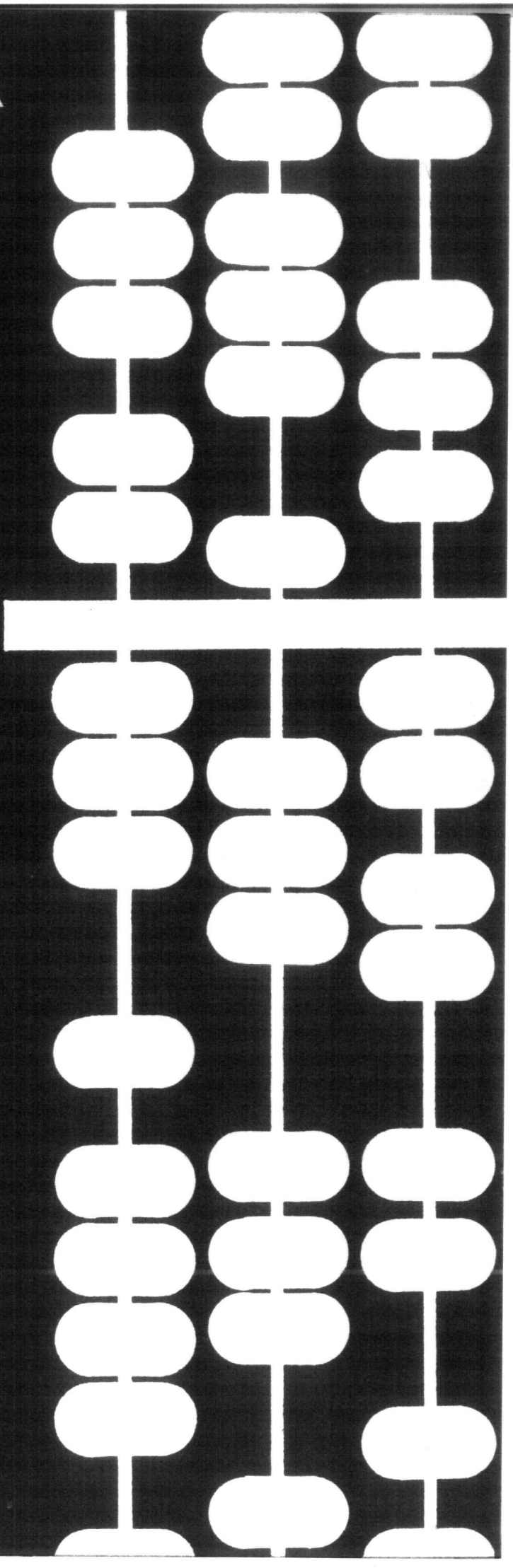
INFORMATICA



2

1977

ŽELITE ?

... svetovalsko pomoč s področja sistemskih programov, daljinske obdelave podatkov in razširitve sistema?

... že izdelane in preizkušene programe, ki obdelujejo podatke posameznih podsistemov podjetja?

... v vaši organizaciji uvesti obdelavo podatkov, pa nimate svojega računalniškega sistema?

... imeti rešene statistične, transportne, linearne ali simulacijske probleme ?

... programe, ki bodo napisani za vašo delovno organizacijo?

... nasvet, kako dobiti od svojega računalniškega sistema kvalitetnejše rezultate s področja poslovne informatike?

... popoln servis obdelave podatkov v vaši delovni organizaciji?

KLIČITE NAS !

UNIVERZA v LJUBLJANI

INSTITUT "JOŽEF ŠTEFAN" odsek za uporabno matematiko

61001 ljubljana jamova 39 jugoslavija

tel.(061) 63-261 tlx: 31-296 yu jostin

INFORMATICA

časopis za tehnologijo računalništva in probleme informatike
časopis za računarsku tehnologiju i probleme informatike
spisanie za tehnologija na smetanjeto i problemi od oblasta na informatikata

Časopis izdaja Slovensko društvo INFORMATIKA,
61000 Ljubljana, Jamova 39, Jugoslavija

Uredništvo in uprava: 61000 Ljubljana, Institut Jožef Stefan, Jamova 39, telefon (061) 63 261, telegram: JOSTIN, telex: 31 296 YU JOSTIN.

Letna naročnina za delovne organizacije 300,00 din, za posamezništva 100,00 din, prodaja posamezne številke 50,00 din
Žiro račun: 50101-678-51841
Stališče uredništva se lahko razlikuje od mnenja avtorjev.

Na podlagi mnenja Republiškega sekretariata za prosveto in kulturo št. 4210-451/77 z dne 4.5.1977, je časopis INFORMATICA strokovni časopis, ki je oproščen temeljnega davka od prometa proizvodov.

Tisk: Tiskarna Kresija, Ljubljana
Grafična oprema: T. Simončič

letnik I. 1977 – št. 2

VSEBINA

INFORMATICA

Journal of Computing and Informatics

Published by INFORMATIKA, Slovene Society for Informatics, 61000 Ljubljana, Jamova 39, Jugoslavija

volume I. 1977 - N° 2

Headquarters: 61000 Ljubljana, Institut Jožef Stefan, Jamova 39, Phone: (061) 63 261, Cable: JOSTIN Ljubljana, Telex: 31 296 YU JOSTIN

Annual subscription rate for abroad is US $ 18 for companies, and US $ 6 for individuals.

Printed by: Tiskarna Kresija, Ljubljana

Design: T. Simončič

VSEBINA

navodilo za pripravo članka

Avtorje prosimo, da pošljejo uredništvu naslov in kratek povzetek članka ter navedejo približen obseg članka (število strani A 4 formata). Uredništvo bo nato poslalo avtorjem ustrezno število formularjev z navodilom.

Članek tipkajte na priložene dvokolonske formularje. Če potrebujete dodatne formularje, lahko uporabite bel papir istih dimenzij. Pri tem pa se morate držati predpisanega formata, vendar pa ga ne vrišite na papir.

Bodite natančni pri tipkanju in temeljiti pri korigiranju. Vaš članek bo s foto postopkom pomanjšan in pripravljen za tisk brez kakršnihkoli dodatnik korektur.

Uporabljajte kvaliteten pisalni stroj. Če le tekst dopušča uporabljajte enojni presledek. Črni trak je obvezen.

Članek tipkajte v prostor obrobljen z modrimi črtami. Tipkajte do črt - ne preko njih. Odstavek ločite z dvojnim presledkom in brez zamikanja prve vrstice novega odstavka.

Prva stran članka :

a) v sredino zgornjega okvira na prvi strani napišite naslov članka z velikimi črkami;
b) v sredino pod naslov članka napišite imena avtorjev, ime podjetja, mesto, državo;
c) na označenem mestu čez oba stolpca napišite povzetek članka v jeziku, v katerem je napisan članek. Povzetek naj ne bo daljši od 10 vrst.
d) če članek ni v angleščini, ampak v katerem od jugoslovanskih jezikov izpustite 2 cm in napišite povzetek tudi v angleščini. Pred povzetkom napišite angleški naslov članka z velikimi črkami. Povzetek naj ne bo daljši od 10 vrst. Če je članek v tujem jeziku napišite povzetek tudi v enem od jugoslovanskih jezikov;
e) izpustite 2 cm in pričnite v levo kolono pisati članek.

Druga in naslednje strani članka:
Kot je označeno na formularju začnite tipkati tekst druge in naslednjih strani v zgornjem levem kotu,

Naslovi poglavij:
naslove ločuje od ostalega teksta dvojni presledek.

Če nekaterih znakov ne morete vpisati s strojem jih čitljivo vpišite s črnim črnilom ali svinčnikom. Ne uporabljajte modrega črnila, ker se z njim napisani znaki ne bodo preslikali.

Ilustracije morajo biti ostre, jasne in črno bele. Če jih vključite v tekst, se morajo skladati s predpisanim formatom. Lahko pa jih vstavite tudi na konec članka, vendar morajo v tem primeru ostati v mejah skupnega dvokolonskega formata. Vse ilustracije morate (nalepiti) vstaviti sami na ustrezno mesto.

Napake pri tipkanju se lahko popravljajo s korekcijsko folijo ali belim tušem. Napačne besede, stavke ali odstavke pa lahko ponovno natipkate na neprozoren papir in ga pazljivo nalepite na mesto napake.

V zgornjem desnem kotu izven modro označenega roba oštevilčite strani članka s svinčnikom, tako da jih je mogoče zbrisati.

Časopis INFORMATICA
Uredništvo, Institut Jožef Stefan, Jamova 39, Ljubljana

Naročam se na časopis INFORMATICA za leto 1977 (štiri številke). Predplačilo bom izvršil po prejemu vaše položnice.
Cenik: letna naročnina za delovne organizacije 300,00 din, za posameznika 100,00 din.

Časopis mi pošiljajte na naslov ☐ stanovanja ☐ delovne organizacije.

Priimek.....

Ime.....

Naslov stanovanja

Ulica.....

Poštna številka_____ Kraj.....

Naslov delovne organizacije

Delovna organizacija.....

.....

Ulica.....

Poštna številka_____ Kraj.....

Datum..... Podpis:

.....

instructions for preparation of a manuscript

Authors are invited to send in the address and short summary of their articles and indicate the opproximate size of their contributions (in terms of A 4 paper). Subsequently they will receive the outor's kits.

Type your manuscript on the enclosed two-column-format manuscript paper. If you require additional manuscript paper you can use similar-size white paper and keep the proposed format but in that case please do not draw the format limits on the paper.

Be accurate in your typing and through in your proof reading. This manuscript will be photographically reduced for reproduction without any proof reading or corrections before printing.

Use a good typewriter. If the text allows it, use single spacing. Use a black ribbon only.

Keep your copy within the blue margin lines on the paper, typing to the lines, but not beyond them. Double space between paragraphs.

First page manuscript:

a) Give title of the paper in the upper box on the first page. Use block letters.
b) Under the title give author's names, company name, city and state - all centered.
c) As it is marked, begin the abstract of the paper. Type over both the columns. The abstract should be written in the language of the paper and should not excesed 10 lines.
d) If the paper is not in English, drop 2 cm after having written the abstract in the language of the paper and write the abstract in English as well. In front of the abstract put the English title of the paper. Use block letters for the title. The lenght of the abstract should not be greater than 10 lines.
e) Drop 2 cm and begin the text of the paper in the left column.

Second and succeeding pages of the manuscript:
As it is marked on the paper, begin the text of the second and succeeding pages in the left upper corner.

Format of the subject headings:
Headings are separated from text by double spacing.

If some characters are not available on your typwriter write them legibly in black ink or with a pencil. Do not use blue ink, because it shows poorly.

Illustrations must be black and white, sharp and clear. If you incorporate your illustrations into the text keep the proposed format. Illustration can also be placed at the end of all text material provided, however, that they are kept within the margin lines of the full size two-column format. All illustrations must be placed into appropriate positons in the text by the author.

Typing errors may be corrected by using white correction paint or by retyping the word, sentence or paragraph on a piece of opaque, white paper and pasting it nearly over errors

Use pencil to number each page on the upper-right-hand corner of the manuscript, outside the blue margin lines so that the numbers may be erased.

Časopis INFORMATICA
Uredništvo, Institut Jožef Stefan, Jamova 39, Ljubljana

Please enter my subscription to INFORMATICA for the volume 1977 (four issues), and send me the bill. Annual subscription price: companies 300,00 din (for abroad US $ 18), individuals 100,00 din (for abroad US $ 6)

Send journal to my ☐ home address ☐ company's address.

Surname.....

Name.....

Home address

Street.....

Postal code _____ City.....

Company address

Company.....

.....

Street.....

Postal code _____ City.....

Date..... Signature

.....

programiranje mikro računalnikov s procesorjem Z80

a. p. železnikar
i. ozimek
m. kovačević
d. novak

UDK 681.3-181.4.06

Odsek za računalništvo
in informatiko
Institut "Jožef Stefan"
Univerza v Ljubljani

Članek opisuje nekatere specifičnosti programiranja na mikro računalnikih s procesorjem Z80. Opisana je globalna shema mikroprocesorskih registrov, načini adresiranja, ukazna razpredelnica in spreminjanje statusne besede pri izvajanju ukazov. Aplikacija posameznih ukazov je prikazana v konkretnih rutinah za manipulacijo tekstov in tabel, pri relativnem naslavljanju, za manipulacijo z biti in pri uporabi indeksnih registrov, pri pomikanju in vrtenju besednih polj ter pri programiranju aritmetičnih, logičnih in izmenjevalnih operacij. Članek daje posamične primerjave za programe z novimi ukazi in brez njih.

Programming of Microcomputers Using Z80 Processor. This article describes particular programming techniques for microcomputers using Z80 processor. A general scheme of processor registers is given; addressing modes, instruction table, and dependence of status word for instructions are presented. The application of several instructions is demonstrated in programs for manipulation of text and tables, in relative addressing, manipulation of bits and useage of index registers, in complex word shifting and rotating, and in programs applying arithmetical, logical and exchange operations. The article gives some comparisions of programs using new instructions and programs without these instructions.

1. Uvod

Programirnaje na mikro računalnikih (kratko na μR) s procesorji tipa 8080, 6800, F8, SC/MP itn. je tudi pri nas že dokaj udomačeno. V μR s procesorjem Z80 je omogočena razen standardnih programirnih pristopov tudi učinkovita tehnika v sistemskih in uporabniških programih. Navadno primerjamo procesor Z80 s procesorjem 8080 predvsem zaradi tega, ker je mogoče programsko opremo za 8080 uporabiti tudi na μR z Z80, seveda pri ustreznih konfiguracijah obeh sistemov (I/O kanali, pomnilnik, operacijski sistem). Procesor Z80 pa vsebuje še taktni generator (z zunanjim kristalom) in generator za transparentno osveževanje dinamičnih pomnilnikov (RAM).

Procesor Z80 izvaja med drugimi tudi ukaze, ki so na področju μR novi pa tudi zelo učinkoviti. Takšne skupine ukazov so: neposredna manipulacija z biti (naslovljivost, primerjava, spreminjanje); avtomatično ponavljanje določenih ukazov, dokler so izpolnjeni ustrezni pogoji (primerjava, nalaganje, vhod in izhod); izmenjevalni ukazi za vrsto registrov z dvojno dolžino besede ter razen absolutnih pogojnih skokov (tudi spremenljivih) še pogojni subrutinski pozivi in vrnitve (te imamo tudi pri procesorju 8080) in pogojni relativni skoki.

2. Zgradba procesorja Z80 in ukazi

Registrska zgradba procesorja je globalno opisana na sliki 1. Akumulator je označen z A, njegov izmenjevalni register pa z A'. V register F se shranjuje t.i. status (zastavica prenosa, ničle, parnosti, polprenosa, prestopa, seštevanja, odštevanja), ki je posledica (ali pa tudi ne) predhodne logične/aritmetične operacije nad različnimi registri.

SP (okrajšava za Stack Pointer) je kazalec, ki kaže v sklad; sklad se gradi v pomnilniku tipa RAM, navadno na njegovem gornjem koncu. Če je SP ime registra, bomo njegovo vsebino označili z (SP); potem je vsebina celice, na katero kaže kazalec (SP), določena z ((SP)). Ta princip označevanja bo veljal za poljubni register pri razlagi pomenov posameznih ukazov. V posameznih ukazih (v zbirnem jeziku procesorja Z80) pa bomo pisali SP za kazalec in (SP) za vsebino celice, na katero kazalec kaže. Npr.

ADD A, (IY+d)

bomo tolmačili s formulo

A ← (A) + ((IY)+d)

kar pomeni: v akumulator 'pride' vrednost, ki je vsota vrednosti v A in vsebine pri naslovu (IY)+d, kjer je (IY) vsebina indeksnega registra IY in d t.i. odmik. Na levi strani znaka ← imamo vselej ime registra (tudi pomnilniški naslov je registrsko ime), na desni strani pa vrednost, ki se imenu prireja.

Na sliki 1 imamo register I, ki je namenjen upravljanju prekinitev. Posledica prekinitve je lahko indirektni poziv na pomnilniško lokacijo; v registru I se tedaj hrani višjih osem bitov indirektnega naslova, nižjih osem bitov pa posreduje prekinjajoča zunanja naprava. Takšen mehanizem omogoča, da shranjujemo prekinitvene servisne subrutine kjerkoli v pomnilniku.

Na sliki 2 imamo grobo strukturo zbirke ukazov procesorja Z80. Iz konkretnega ukaza je tudi razviden način naslavljanja. Takih načinov je šest: direktno, registrsko, registrsko indirektno, takojšnje, indeksno in relativno naslavljanje.

Pri direktnem naslavljanju se nahaja operand na mestu drugega in tretjega zloga (računalniške besede) ukaza. Pri tem se shrani ope-

rand v pomnilniku v t.i. dvozložni inverzni obliki: najprej (na nižjem naslovu) je nižjih 8 bitov operanda, za njimi pa višjih 8 bitov. Imamo primer: ukaz

LD A, (0158H);

se zapiše v pomnilnik kot zaporedje treh zlogov

3A 58 01

in pomeni

A ← ((0158H))

kjer označuje H heksadecimalno predstavitev.

Pri registrskem naslavljanju se nahaja operand v enem od splošnih registrov (glej sliko 1). Vzemimo primer ukaza

BIT 6, E;

ki se zapiše v pomnilnik kot dvozložno zaporedje CB 73 in pomeni preizkus bita 6 registra E, ko velja Z ← $\overline{E6}$, kjer je Z statusni bit (F6) in $\overline{E6}$ negirani šesti bit registra E. Natančneje bi ta ukaz opisali takole:

IF (E6=0) THEN Z ← 1 ELSE Z ← 0

Registrsko indirektno naslavljanje se nanaša na registrske pare (B,C), (D,E) in (H,L), ki so pri tem načinu naslavljanja operandi. Imena teh registrov okrajšamo v BC, DE in HL; najbolj uporaben je register HL. Primer bodi

DEC (HL);

ki pomeni (HL) ← ((HL))-1. Vsebina pomnilniške lokacije (HL) se zmanjša za 1. V tej zvezi so zlasti zanimivi skočni ukazi

JP (HL);

Takojšnje naslavljanje omogoča neposredno uporabo konstant (fiksiranih vrednosti). Npr.

ADD 11;

pomeni A ← (A) + 11D, kjer označuje D decimalno predstavitev. Zlasti pogostni so ukazi tipa 'load', ko imamo takojšnje nalaganje, na primer v:

LD A, 12H; oziroma LD HL, 1234H;

s pomenom A ← 12H oziroma HL ← 1234H.

Pri indeksnem naslavljanju se nahaja operand na pomnilniški lokaciji (naslovu), ki je določena z vsoto tretjega ukaznega zloga (indeksa) z vsebino enega od obeh indeksnih registrov (IX in IY). Ti ukazi so tri- in štirizložni. Tretji zlog ukaza postane v pomnilniku t.i. dvojiški komplement. V primeru

SET 4, (IX+21);

imamo pomen $((IX)+21)_4$ ← 1. Za ukaz

RES 7, (IY+85H);

pa bi imeli $((IY)-7BH)_7$ ← 0. Za prvi ukaz imamo štirizložni kod DD CB 21 E6, za drugi pa FD CB 85 BE. Na sliki 2 označuje vejica sumand z indeksnim registrom, ko imamo FD CB,BE za prejšnji primer.

Pri relativnem naslavljanju se uporablja zlog, ki sledi operacijskemu ukaznemu zlogu,za pomik (relativni skok) od trenutne lokacije (stanja programskega števnika). Ta pomik je dvojiški komplement, ki se prišteva k naslovu operacijskega zloga naslednjega ukaza (tj. naslov zadevnega operacijskega zloga povečan za 2). V zbirnem jeziku uporabljamo za relativni skok predznak (kadar nimamo imena). Npr.

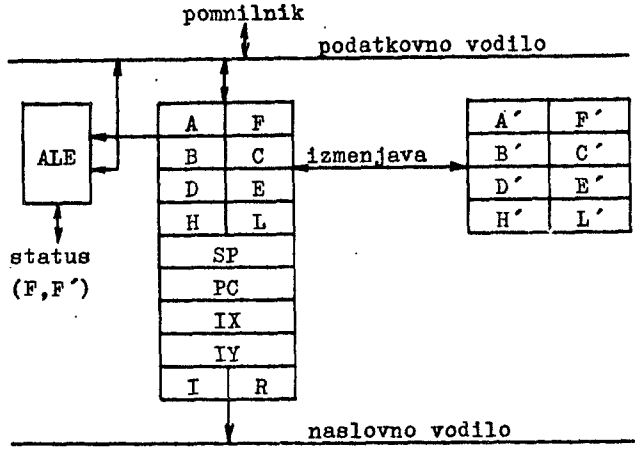
JR +3;

pomeni skok na lokacijo (PC) +2+3, kjer je (PC) naslov operacijskega koda tega ukaza; PC je ime ukaznega (programskega) števnika. Če začenja ta ukaz na lokaciji 835, imamo skok na 83A.

Na sliki 2 imamo le grobi opis ukazov, saj so izpuščeni podatki o spremembi statusa po izvedbi posameznega ukaza. Ti podatki so zbrani na sliki 3.

3. Tekstovne in tabelne manipulacije

Učinkovita manipulacija tekstov (sporočil) in tabel je osnovnega pomena za uporabo procesorja Z80 v poslovnih in komunikacijskih konfiguracijah. Oglejmo si poseben tip ukazov,



Slika 1. Registrska zgradba procesorja Z80: A,B,...,L so primarni 8-bitni registri; A',B',...,L' so njihovi izmenjevalni dvojniki; AF, BC, DE, HL so 16-bitne sestavljenke ustreznih registrov; SP, PC, IX, IY so 16-bitni registri, in sicer skladni kazalec (SP), programski števnik (PC) ter indeksna registra (IX,IY); I je prekinitveni register, ki kaže naslov pomnilnega segmenta in R je osveževalni (naslovni) register; ALE je aritmetična/logična enota; v F (statusni register) se shranjujejo zastavice (bø = carry, b1 = add/sub, b2 = parity/overflow, b4 = half carry, b6 = zero, b7 = sign)

ki jih bomo imenovali ponavljalni ukazi, s katerimi je moč učinkovito uresničiti prenos ali kopiranje tabel, iskanje v tabelah in tekstih ter vhodni in izhodni prenos (nalaganje).

Ponavljalni ukazi so nalagalni oziroma kopirni (LDIR, LDDR), primerjalni (CPIR, CPDR), vhodni (INIR, INDR) ter izhodni (OTIR, OTDR). Ti ukazi so dvozložni, njihova štirimestna mnemonična imena pa končujejo s črko R (repeat). Črka I pred R pomeni inkrementiranje vsebine dvojnega registra HL in/ali ne registra DE,črke D pa ustrezno dekrementiranje. Splošna oblika ponavljalnega ukaza je dountil stavek:

```
DOUNTIL((poseben pogoj)OR (BC)=0)
   akcija ponavljalnega ukaza;
   inkrementiranje ali dekrementiranje (HL)
      in/ali ne (DE);
   dekrementiranje (BC);
   INCLUDE zastavica-(BC)
ENDDO
```

V tem stavku je lahko poseben pogoj prazen,prazen je lahko tudi segment zastavica-(BC).

Če je Z zastavica (šesti bit statusa F na sliki 1), ki je 1 le tedaj, ko je primerjava z vsebino akumulatorja pozitivna in če je P/V zastavica parnosti oziroma prestopa (drugi bit v F), imamo za ukaz CPIR tole:

```
DOUNTIL(Z=1 OR (BC)=0)
   IF ((A)=((HL)))THEN Z ← 1 ELSE Z ← 0
   ENDIF
   HL ← (HL)+1; BC ← (BC)-1
   IF((BC)=0) THEN P/V ← 0 ELSE P/V ← 1
   ENDIF
ENDDO
```

Z ukazom CPIR (oziroma CPDR) primerjamo vsebino akumulatorja z vsebino pomnilniške celice (HL) tako dolgo (inkrementiranje, dekrementiranje), dokler ne naletimo na enakost ali ko postane (BC) = 0. To pa pomeni, da iščemo poljubni znak v danem nizu znakov z enim samim ukazom, seveda pri ustreznem stanju registrov HL, BC in A.

	A	B	C	D	E	H	L	(HL)	Imm	(IX+d)	(IY+d)	
ADD	87	80	81	82	83	84	85	86	C6	DD86	FD86	add r to A
ADC	8F	88	89	8A	8B	8C	8D	8E	CE	DD8E	FD8E	add r to A with CRY
AND	A7	A0	A1	A2	A3	A4	A5	A6	E6	DDA6	FDA6	and r with A
BIT 0	CB47	CB40	CB41	CB42	CB43	CB44	CB45	CB46	-	DDCB,46	FDCB,46	test bit 0
BIT 1	CB4F	CB48	CB49	CB4A	CB4B	CB4C	CB4D	CB4E	-	DDCB,4E	FDCB,4E	test bit 1
BIT 2	CB57	CB50	CB51	CB52	CB53	CB54	CB55	CB56	-	DDCB,56	FDCB,56	test bit 2
BIT 3	CB5F	CB58	CB59	CB5A	CB5B	CB5C	CB5D	CB5E	-	DDCB,5E	FDCB,5E	test bit 3
BIT 4	CB67	CB60	CB61	CB62	CB63	CB64	CB65	CB66	-	DDCB,66	FDCB,66	test bit 4
BIT 5	CB6F	CB68	CB69	CB6A	CB6B	CB6C	CB6D	CB6E	-	DDCB,6E	FDCB,6E	test bit 5
BIT 6	CB77	CB70	CB71	CB72	CB73	CB74	CB75	CB76	-	DDCB,76	FDCB,76	test bit 6
BIT 7	CB7F	CB78	CB79	CB7A	CB7B	CB7C	CB7D	CB7E	-	DDCB,7E	FDCB,7E	test bit 7
CP	BF	B8	B9	BA	BB	BC	BD	BE	FE	DDBE	FDBE	compare r with A
DEC	3D	05	0D	15	1D	25	2D	35	-	DD35	FD35	decrement r
INC	3C	04	0C	14	1C	24	2C	34	-	DD34	FD34	increment r
IN(C)	ED78	ED40	ED48	ED50	ED58	ED60	ED68	-	-	-	-	input to r
LD A	7F	78	79	7A	7B	7C	7D	7E	3E	DD7E	FD7E	load A with r
LD B	47	40	41	42	43	44	45	46	06	DD46	FD46	load B with r
LD C	4F	48	49	4A	4B	4C	4D	4E	0E	DD4E	FD4E	load C with r
LD D	57	50	51	52	53	54	55	56	16	DD56	FD56	load D with r
LD E	5F	58	59	5A	5B	5C	5D	5E	1E	DD5E	FD5E	load E with r
LD H	67	60	61	62	63	64	65	66	26	DD66	FD66	load H with r
LD L	6F	68	69	6A	6B	6C	6D	6E	2E	DD6E	FD6E	load L with r
LD(HL)	77	70	71	72	73	74	75	-	36	-	-	load (HL) with r
LD(IX+d)	DD77	DD70	DD71	DD72	DD73	DD74	DD75	-	DD36	-	-	load (IX+d) with r
LD(IY+d)	FD77	FD70	FD71	FD72	FD73	FD74	FD75	-	FD36	-	-	load (IY+d) with r
OR	B7	B0	B1	B2	B3	B4	B5	B6	F6	DDB6	FDB6	or r with A
OUT(C)	ED79	ED41	ED49	ED51	ED59	ED61	ED69	-	-	-	-	output r
RES 0	CB87	CB80	CB81	CB82	CB83	CB84	CB85	CB86	-	DDCB,86	FDCB,86	reset bit 0
RES 1	CB8F	CB88	CB89	CB8A	CB8B	CB8C	CB8D	CB8E	-	DDCB,8E	FDCB,8E	reset bit 1
RES 2	CB97	CB90	CB91	CB92	CB93	CB94	CB95	CB96	-	DDCB,96	FDCB,96	reset bit 2
RES 3	CB9F	CB98	CB99	CB9A	CB9B	CB9C	CB9D	CB9E	-	DDCB,9E	FDCB,9E	reset bit 3
RES 4	CBA7	CBA0	CBA1	CBA2	CBA3	CBA4	CBA5	CBA6	-	DDCB,A6	FDCB,A6	reset bit 4
RES 5	CBAF	CBA8	CBA9	CBAA	CBAB	CBAC	CBAD	CBAE	-	DDCB,AE	FDCB,AE	reset bit 5
RES 6	CBB7	CBB0	CBB1	CBB2	CBB3	CBB4	CBB5	CBB6	-	DDCB,B6	FDCB,B6	reset bit 6
RES 7	CBBF	CBB8	CBB9	CBBA	CBBB	CBBC	CBBD	CBBE	-	DDCB,BE	FDCB,BE	reset bit 7
RL	CB17	CB10	CB11	CB12	CB13	CB14	CB15	CB16	-	DDCB,16	FDCB,16	rotate left
RLC	CB07	CB00	CB01	CB02	CB03	CB04	CB05	CB06	-	DDCB,06	FDCB,06	rotate left circular
RR	CB1F	CB18	CB19	CB1A	CB1B	CB1C	CB1D	CB1E	-	DDCB,1E	FDCB,1E	rotate right
RRC	CB0F	CB08	CB09	CB0A	CB0B	CB0C	CB0D	CB0E	-	DDCB,0E	FDCB,0E	rotate right circular
SBC	9F	98	99	9A	9B	9C	9D	9E	DE	DD9E	FD9E	subtract r from A with BRW
SET 0	CBC7	CBC0	CBC1	CBC2	CBC3	CBC4	CBC5	CBC6	-	DDCB,C6	FDCB,C6	set bit 0
SET 1	CBCF	CBC8	CBC9	CBCA	CBCB	CBCC	CBCD	CBCE	-	DDCB,CE	FDCB,CE	set bit 1
SET 2	CBD7	CBD0	CBD1	CBD2	CBD3	CBD4	CBD5	CBD6	-	DDCB,D6	FDCB,D6	set bit 2
SET 3	CBDF	CBD8	CBD9	CBDA	CBDB	CBDC	CBDD	CBDE	-	DDCB,DE	FDCB,BE	set bit 3
SET 4	CBE7	CBE0	CBE1	CBE2	CBE3	CBE4	CBE5	CBE6	-	DDCB,E6	FDCB,E6	set bit 4
SET 5	CBEF	CBE8	CBE9	CBEA	CBEB	CBEC	CBED	CBEE	-	DDCB,EE	FDCB,EE	set bit 5
SET 6	CBF7	CBF0	CBF1	CBF2	CBF3	CBF4	CBF5	CBF6	-	DDCB,F6	FDCB,F6	set bit 6
SET 7	CBFF	CBF8	CBF9	CBFA	CBFB	CBFC	CBFD	CBFE	-	DDCB,FE	FDCB,FE	set bit 7
SLA	CB27	CB20	CB21	CB22	CB23	CB24	CB25	CB26	-	DDCB,26	FDCB,26	shift left arithmetic
SRA	CB2F	CB28	CB29	CB2A	CB2B	CB2C	CB2D	CB2E	-	DDCB,2E	FDCB,2E	shift right arithmetic
SRL	CB3F	CB38	CB39	CB3A	CB3B	CB3C	CB3D	CB3E	-	DDCB,3E	FDCB,3E	shift right logical
SUB	97	90	91	92	93	94	95	96	D6	DD96	FD96	subtract r from A
XOR	AF	A8	A9	AA	AB	AC	AD	AE	EE	DDAE	FDAE	exclusive-or r with A

Special load group, coded as LD source, destination

A,I	ED57	A=I
A,R	ED5F	A=R
R,A	ED4F	R=A
I,A	ED47	I=A
(N),A	32	store A
A,(n)	3A	load A
SP,IX	DDF9	SP=IX
SP,IY	FDF9	SP=IY
SP,HL	F9	SP=HL

Slika 2. Ukazi procesorja Z80: mnemonika in kodi. V stolpcu (IX+d) pomeni npr. DDCB,46 , da pride med B in 4 (namesto vejice) vstavljen kod za d.

	(PSW,A)	(B,C)	(D,E)	(H,L)	SP	IX	IV	
ADD HL	-	09	19	29	39	-	-	add pair to HL
ADD IX	-	DD09	DD19	-	DD39	DD29	-	add pair to IX
ADD IY	-	FD09	FD19	-	FD39	-	FD29	add pair to IY
ADC HL	-	ED4A	ED5A	ED6A	ED7A	-	-	add pair to HL with CRY
SBC HL	-	ED42	ED52	ED62	ED72	-	-	subtract pair from HL with BRW
DEC	-	0B	1B	2B	3B	DD2B	FD2B	decrement r pair
INC	-	03	13	23	33	DD23	FD23	increment r pair
LD A,(r)	-	0A	1A	7E	-	-	-	load A indirect
LD (r),A		02	12	77	-	-	-	store A indirect
LXI	-	01	11	21	31	DD21	FD21	load r pair immediate
POP	F1	C1	D1	E1	-	DDE1	FDE1	pop r pair from stack
PUSH	F5	C5	D5	E5	-	DDE5	FDE5	push r pair onto stack
LD r,(n)	-	ED4B	ED5B	2A	ED7B	DD2A	FD2A	load r pair from memory
LD (n),r	-	ED43	ED53	22	ED73	DD22	FD22	store r pair in memory

	Inc	Inc & rep.	Dec	Dec & rep.	
CP	EDA1	EDB1	EDA9	EDB9	compare, inc(dec)HL,dec BC
LD	EDA0	EDB0	EDA8	EDB8	load (DE) with (HL), inc(dec)HL and DE,dec BC
OUT	EDA3	EDB3	EDAB	EDBB	output (HL), inc(dec)HL,dec B
IN	EDA2	EDB2	EDAA	EDBA	input to (HL), inc(dec) HL,dec B

	0	1	2	3	4	5	6	7	
RST	C7	CF	D7	DF	E7	EF	F7	FF	restart call to location i*8

	Unc.	Zero/ Not Zero	Carry/ No Carry	Plus/ Minus	Even Parity/ Odd Parity	
CALL	CD	CC/C4	DC/D4	F4/FC	EC/E4	call subroutine if condition true
JP	C3	CA/C2	DA/D2	F2/FA	EA/E2	jump if condition true
JR	18	28/20	38/30	-	-	jump relative if condition true
RET	C9	C8/C0	D8/D0	F0/F8	E8/E0	return if condition true

CCF	3F	complement carry	IN	DB	input to A
CPL	2F	complement A(1's)	JP(HL)	E9	jump to (HL)
DAA	27	decimal adjust A	JP(IX)	DDE9	jump to IX
DI	F3	disable interrupts	JP(IY)	FDE9	jump to IY
DJNZ	10	decrement B, jump B≠0	NEG	ED44	complement A(2's)
EI	FB	enable interrupts	NOP	00	no operation
EX DE,HL	EB	exchange (D,E) & (H,L)	OUT	D3	output from A
EX AF,AF'	08	exchange (A,F) & (A,F)'	RETI	ED4D	return from interrupt
EXX	D9	exchange (B,C,D,E,H,L) & (B,C,D,E,H,L)'	RETN	ED45	return from NMI interrupt
EX(SP),HL	E3	exchange (H,L) & top of stack	RLA	17	rotate A left thru carry
EX(SP),IX	DDE3	exchange IX & top of stack	RRA	1F	rotate A right thru carry
EX(SP),IY	FDE3	exchange IY & top of stack	RLCA	07	rotate A left circular
HALT	76	halt processor	RRCA	0F	rotate A right circular
IM0	ED46	interrupt mode 0	RLD	ED6F	rotate left digit
IM1	ED56	interrupt mode 1	RRD	ED67	rotate right digit
IM2	ED5E	interrupt mode 2	SCF	37	set carry flag

Slika 2. Ukazi procesorja Z80 (nadaljevanje)

Instruction	C	Z	P/V	S	N	H	Comments
ADD s; ADC s	‡	‡	V	‡	0	‡	8-bit add or add with carry
SUB s; SBC s, CP s, NEG	‡	‡	V	‡	1	‡	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	0	‡	P	‡	0	1	Logical operations
OR s; XOR s	0	‡	P	‡	0	0	And set's different flags
INC s	•	‡	V	‡	0	‡	8-bit increment
DEC s	•	‡	V	‡	1	‡	8-bit decrement
ADD DD, SS	‡	•	•	•	0	X	16-bit add
ADC HL, SS	‡	‡	V	‡	0	X	16-bit add with carry
SBC HL, SS	‡	‡	V	‡	1	X	16-bit subtract with carry
RLA; RLCA, RRA, RRCA	‡	•	•	•	0	0	Rotate accumulator
RL s; RLC s; RR s; RRC s SLA s; SRA s; SRL s	‡	‡	P	‡	0	0	Rotate and shift location s
RLD, RRD	•	‡	P	‡	0	0	Rotate digit left and right
DAA	‡	‡	P	‡	•	‡	Decimal adjust accumulator
CPL	•	•	•	•	1	1	Complement accumulator
SCF	1	•	•	•	0	0	Set carry
CCF	‡	•	•	•	0	X	Complement carry
IN r, (C)	•	‡	P	‡	0	0	Input register indirect
INI; IND; OUTI; OUTD	•	‡	X	X	1	X	Block input and output
INIR; INDR; OTIR; OTDR	•	1	X	X	1	X	Z = 0 if B ≠ 0 otherwise Z = 1
LDI, LDD	•	X	‡	X	0	0	Block transfer instructions
LDIR, LDDR	•	X	0	X	0	0	P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI, CPIR, CPD, CPDR	•	‡	‡	X	1	X	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	•	‡	IFF	‡	0	0	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	•	‡	X	X	0	1	The state of bit b of location s is copied into the Z flag

Slika 3. Tu pomeni: C bit prenosa (carry), Z bit ničle (zero), S bit predznaka (sign), P/V bit parnosti ali prestopa (parity/overflow), H bit prenosa iz spodnje polovice zloga (half-carry) in N bit seštevanja/odštevanja; ‡ nakazuje spremembo bita, • njegovo nespremenjenost, 0 anuliranje in 1 postavitev na vrednost 1; X označuje nepomembnost; nadalje predstavlja r registre A,B,C,D, E,H,L; črka s predstavlja 8-bitno in SS 16-bitno lokacijo.

Ponavljalnim ukazom so enolično prirejeni t.i. koračni ukazi, ki se izvajajo neponavljalno. To so ukazi za nalaganje (LDI, LDD), primerjavo (CPI, CPD), vhod (INI, IND) in izhod (OUTI, OUTD). Splošna shema teh ukazov je:

akcija koračnega ukaza;
inkrementiranje ali dekrementiranje (*HL*) in/ali ne (DE);
dekrementiranje (BC);
INCLUDE zastavica-(BC);

Koračni ukazi predstavljajo tako telesa ustreznih dountil stavkov za ponavljalne ukaze.

Z uporabo ponavljalnih in koračnih ukazov je mogoče doseči dokaj enostavno zgradbo posebnih programov. Vzemimo kot primer subrutino za iskanje podteksta v danem tekstu. Napišimo najprej pomožno subrutino SEBA s temle psevdo kodom:

```
SUBROUTINE SEBA
   poišči prvi znak podteksta v danem tekstu;
   če takega znaka v tekstu ni, izstopi;
   sicer pa shrani: inkrementirano lokacijo
      ujemanja (HL) in dekrementirano stanje
      tekstovnega števnika (BC);
   ugotovi, ali se naslednji znaki podteksta
      pojavljajo zaporedoma v tekstu;
   če je bil pred popolnim ujemanjem podteksta
      in dela teksta dosežen konec teksta (glej (BC)), takoj izstopi;
   sicer pa izstopi zaradi ujemanja ali neujemanja s podatki v ustreznih registrih
ENDSUBROUTINE
```

To subrutino zapišemo konkretno takole:

```
SEBA:;
 LD    A,(IX+00H);LOAD FIRST SUBTEXT CHAR
 CPIR             ;REPEAT SEARCH FOR (A)=((HL))
 RET   PO         ;EXIT IF (BC)=0
 LD    (L6),HL    ;SAVE (HL)
 LD    (L7),BC    ;SAVE (BC)
LOOP:;
 INC   IX         ;PUT NEXT SUBTEXT CHAR
 LD    A,(IX+00H);   INTO ACCUMULATOR
 CPI              ;COMPARE (A)=((HL)), INC (HL),
;                     DEC (BC)
 RET   PO         ;EXIT IF (BC)=0
 JR    Z,LOOP     ;   ELSE GO TO LOOP
 RET              ;EXIT
```

Tu sta L6 in L7 posebni lokaciji. Če je bila primerjava uspešna, imamo npr. po izstopu iz subrutine ((IX)) = 'NULL', kar je znak konca podteksta. V nasprotnem primeru pa inicializiramo IX (na začetek podteksta) ter pokličemo subrutino SEBA z obstoječimi parametri L6,L7 (ju izenačimo z HL, BC) znova. Naslove uspeha (L6)-1 lahko shranjujemo in imamo tako zabeležene vse točke vstopanja podteksta v dani tekst.

Imejmo dve razpredelnici z njunima začetnima naslovoma RAZ1 in RAZ2 ter z enakim obsegom OBS, ki predstavlja število zlogov v posamezni razpredelnici. Podatki RAZ1, RAZ2 in OBS naj se shranjujejo na lokacijah (naslovih) LOC1, LOC2 in LOC3. Podatke prenesemo iz prve v drugo razpredelnico takole:

```
 LD    DE,(LOC2); (DE) je namenski naslov
 LD    HL,(LOC1); (HL) je izvirni naslov
 LD    BC,(LOC3); (BC) je število zlogov
 LDIR           ; izvede se prenos
```

Pri tem se tabeli vobče ne smeta prekrivati. Na podoben način se lahko uporabi tudi ukaz LDDR. Če pa takega ukaza ni, moramo imeti zanj tale *programski segment:*

```
LOOP:;
 LD    A,(DE)    ; v A pride ((DE))
 LD    (HL),A    ; v (HL) pride (A)
 DEC   DE        ; v DE pride (DE)-1
 DEC   HL        ; v HL pride (HL)-1
 DEC   BC        ; v BC pride (BC)-1
 LD    A,B       ; preizkusi (BC)=0 z
 OR    C         ;    B OR C = 0
 JR    NZ,LOOP   ; če je (BC)≠0, pojdi v LOOP
```

Ukaza LDIR in LDDR lahko uporabimo tudi v primerih kopiranja podatkov iz ene v drugo tabelo, ko se le-ti prekrivata; v tem primeru se pokvari izvirna tabela.

Ukaza LDIR in LDDR lahko uporabimo tudi za začetno (ali posebno) nastavitev razpredelnice (npr. za začetno nastavitev simbolne tabele). Npr. za ukaz LDIR imamo psevdo kod:

```
DOUNTIL((BC)=0)
   (DE) <-- ((HL));
   DE <-- (DE)+1; HL <-- (HL)+1;
   BC <-- (BC) -1
ENDDO
```

Tako dobimo:

```
 LD    HL,RAZ    ; v HL je začetek razpredelnice
 LD    (HL),0    ; anuliraj prvo lokacijo
 LD    DE,RAZ+1  ; v DE je začetek + 1
 LD    BC,OBS-1  ; preostali blok je OBS - 1
 LDIR            ; anuliraj preostale lokacije
```

Tako smo celotno razpredelnico anulirali, ko se je prva lokacija, postavljena na nič, kopirala v drugo, druga v tretjo itn.

K osnovnim tekstovnim manipulacijam sodi tudi vpisovanje in izpisovanje teksta (npr. v ASCII kodu). Procesor Z80 omogoča med drugim (zaradi materializiranega sklada) takojšnje rekurzivno pozivanje subrutin. Imejmo določeno Z80 konfiguracijo (npr. z UART v vhodnem/izhodnem kanalu), kjer bodi KONVR ime kontrolnih vrat za serijski prenos podatkov (vsebina vrat je status UARTja) ter TTY ime vhodnih/izhodnih vrat. Subrutina za vpis znaka iz teleprinterja (serijskega kanala) v akumulator A bodi:

```
TTYVP:;
 IN    A,(KONVR);čakalna zanka, ko se preizkuša
 BIT   1,A      ;   kontrolni bit 1 na enico za
 JR    Z,TTYVP  ;   serijski vhod
 IN    A,(TTY)  ;znak pride iz vrat v akumulat.
 RET            ;vrnitev v pozivno točko
```

Podobno je mogoče napisati tudi subrutino za izpis znaka, ki je v A, na teleprinter (oziroma v serijski kanal). Imamo:

```
TTYIZ:;
 PUSH  AF          ;ohrani AKU in status
TEST:;
 IN    A,(KONVR)   ;čakalna zanka: preizkus kon-
 BIT   0,A         ;    trolnega bita 0 na enico
 JR    Z,TEST      ;    za serijski izhod
 POP   AF          ;vrni vrednost v AKU in stat.
 OUT   (TTY),A     ;znak se odda v seriski kan.
 RET               ;vrnitev v pozivno točko
```

Za vpis/izpis (vpis z odmevom) dobimo tole preprosto subrutino:

```
TVHIZ:;
 CALL  TTYVP       ;včitaj znak
 CALL  TTYIZ       ;    izpiši ga
 RET               ;    in vrni se
```

kjer je zadevni znak ob izstopu v A.

Izpišimo iz pomnilnika še poljubno dolgo zaporedje znakov, tj. sporočilo, ki začenja na lokaciji BEG, končuje se pa vselej z ASCII znakom 'NULL', tj. z 00H. Pred pozivom te subrutine naložimo (BEG) v HL z ukazom LD HL,BEG. Tedaj imamo:

```
SPORIZ:;
 LD    A,(HL)      ;v A pride ((HL))
 CALL  TTYIZ       ;poziv izpisa znaka
 OR    A           ;primerjava na 00H v AKU
 RET   Z           ;izstop pri 00H
 INC   HL          ;povečanje (HL)
 JR    SPORIZ      ;izpis naslednjega znaka
```

Podobno imamo za vpis sporočila:

```
SPORVP:;
 CALL  TTYVP       ;v A pride znak iz kanala
 LD    (HL),A      ;v (HL) se vpiše (A)
 OR    A           ;primerjava na 00H
 RET   Z           ;izstop pri 00H
 INC   HL          ;povečanje (HL)
 JR    SPORVP      ;vpis naslednjega znaka
```

Med osnovne tabelne manipulacije sodi tudi nalaganje strojnega koda v pomnilnik (v heksadecimalni obliki) preko teleprinterja (serijskega kanala) ter izpis naloženega koda v obliki pravokotne tabele ali pa celo v t.i. ukaznem formatu. Zbirka teh subrutin za različne µR bo opisana kdaj drugič.

4. Relativno naslavljanje

V primerjavi s procesorjem 8080 ima procesor Z80 tudi nove ukaze za relativne skoke (pogojne in brezpogojni skok), in sicer

JR, JR C, JR NC, JR Z, JR NZ, DJNZ

Ta množica ukazov je pa v primerjavi z drugimi procesorji (npr. F8) še vedno pičla, saj pri teh pogojnih skokih le ničlo in prenos. Procesor F8 lahko preizkuša pri relativnih skokih še prestop, predznak in z dvema logičnima funkcijama (konjunkcijo in disjunkcijo) še kombinacije prestopa, ničle, prenosa, predznaka in njihovih negacij. Vendar velja poudariti, da je s kontekstno ustrezno uporabo kriterijev za ničlo in prenos mogoče rešiti probleme predznaka in prestopa.

Relativne skoke uporabljamo predvsem v lokalnih programskih kontekstih; z njimi je programski kod krajši in ga lažje premeščamo v pomnilniku. Pri daljših skokih (ki presegajo območje -126 do +129 lokacij od lokacije operacijskega koda skočnega ukaza) lahko uporabimo dvojni ali večkratni relativni skok. Pri tem upoštevamo, da je trajanje relativnega skoka nekako za 20% večje od trajanja normalnega skoka (ukazi JP, JP C, JP NC itn.) in da je to časovno povečanje lahko v nekaterih primerih kritično (prekinitvene subrutine, časovna konverzija). Torej moramo upoštevati problem prostora (relativni skok) in problem časa (normalni skok) v posameznem uporabniškem programu.

V okviru relativnih skokov velja posebej obravnavati ukaz DJNZ LOOP. Ta ukaz je zlasti pripraven v t.i. kontrolnih zankah. Njegov psevdo kod je:

IF((B)≠0) THEN GO TO LOOP ELSE ENDIF

Imejmo začetni vmesnik ZVM (tj. prvi naslov tega vmesnika) in končni vmesnik KVM. Želimo subrutino, ki bo pomikala zloge iz ZVM v KVM, dokler ne najde ASCII znaka 'CR' ali dokler ni pomaknila 80 zlogov. Imamo:

```
VMPR:;
 LD    B,80    ; nastavi števnik B
 LD    HL,ZVM  ; nastavi začetni kazalec
 LD    DE,KVM  ; nastavi končni kazalec
ZANKA:;
 LD    A,(HL)  ; prenesi zlog iz začetnega
 LD    (DE),A  ;    v končni vmesnik
 CP    0DH     ; primerjaj na znak 'CR'
 RET   Z       ; prenos je opravljen
 INC   HL      ; sicer pa nadaljuj
 INC   DE      ;    s prenosom naslednega znaka
 DJNZ  ZANKA   ;
```

Ukaz DJNZ nadomešča tako dva ukaza, in sicer DEC in JP oziroma JR; s tem se skrajša kodni zapis za en zlog in čas izvajanja za tri procesorske cikle.

5. Manipulacija z biti in uporaba indeksnih registrov

Procesor Z80 ima ukaze za preizkušanje, anuliranje in postavitev poljubnega besednega bita (0,1,...,7) v registrih A,B,C,D,E,H,L ter v pomnilniku, če se naslovni kazalci nahajajo v registrih HL, IX in IY. Uporaba 'bitnih' ukazov pride v poštev pri konceptih kontrolnih, indikacijskih zastavic pri programiranju zunanjih kanalov, v notranjih programih, pri zgoščevanju podatkov (pakiranju) v pomnilniku ter pri organizaciji in uporabi t.i. binarnih tabel oziroma matrik (preklopne in relacijske matrike v telefoniji, v gramatičnih in drugih uporabniških obdelavah). Z enim samim ukazom je tako mogoče opraviti ustrezno bitno manipulacijo na poljubnem mestu in lokaciji.

Že v dveh prejšnjih primerih (subrutina TTYVP in TTYIZ) smo uporabili ukaz BIT za preizkus bita 0 in 1 v akumulatorju. V ukazih

SET bit,IME; RES bit,IME; BIT bit,IME;

označuje 'bit' element 0,1,...,7 in IME ime registra A,B,...,L oziroma izraze (HL), (IX+D) in (IY+D). Kadar želimo anulirati tretji bit vsebine na naslovu 100AH, uporabimo segment

```
 LD HL,100AH; naloži 100AH v HL
 RES 3,(HL) ; anuliraj bit 3 vsebine (100AH)
```

oziroma tudi

```
 LD  IX,1000H; naloži 1000H v IX
 RES 3,(IX+0AH); prištej A k 1000H in anuliraj
;                    bit 3 vsebine (100AH)
```

ali

```
 LD  IY,100BH  ; naloži 100BH v IY
 RES 3,(IY+FFH); odštej 1 od 100BH in anuliraj
;                    bit 3 vsebine (100AH)
```

Pri indeksnem naslavljanju lahko za D v izrazih IX+D in IY+D določimo posebna imena z uporabo zbirniške direktive EQU. Če pišemo

```
 X   EQU  0 ; X ima vrednost 0
 Y   EQU  1 ; Y ima vrednost 1
 Z   EQU  2 ; Z ima vrednost 2
```

potem smo s segmentom

```
 LD   IX,1E00H ; postavi (IX) na 1E00H
 SET  5,(IX+X) ; bit 5 vsebine (1E00) je 1
 RES  4,(IX+Y) ; bit 4 vsebine (1E01) na 0
 BIT  3,(IX+Z) ; če je bit 3 vsebine (1E02)
 JP   Z,YES    ;    enak 1 pojdi v YES
 ...           ;    sicer ...
```

postavljali in anulirali ter preizkušali bite besed od nekega osnovnega naslova, tj. 1E00H navzgor.

6. Pomikanje in rotiranje besed

Procesor Z80 ima bogato zalogo raznovrstnih pomikalnih in vrtilnih ukazov, ki jih je podobno kot 'bitne' ukaze mogoče uporabiti nad registri A,B,...,L in nad pomnilniškimi besedami (kazalci v HL, IX in IY).

Sestavljena vrtilna ukaza sta npr. RLD in RRD, ko imamo skici:

Tu imamo 4-bitno rotacijo med akumulatorjem A in pomnilno celico z naslovom (HL). Ta ukaza sta npr. pripravna za zbiranje heksadecimalnih številk v heksadecimalno (notranje binarno) število. Pri vpisu heksadecimalnega števila iz periferije, ko prihajajo v μR heksadecimalne številke v obliki ASCII kodov, se te najprej preslikajo v ustrezne 4-bitne ekvivalente in nato zlagajo v pomnilnik. Na ta način je mogoče enostavno včitati poljubno dolgo heksadecimalno število v pomnilnik.

Če označimo s Q virtualni register ter z (HL)' položaj nižjih 4 bitov in z (HL)'' položaj višjih 4 bitov celice (HL), dobimo za ukaz RLD shemo:

$$Q \leftarrow (A);\ A \leftarrow ((HL)'');\quad (HL)'' \leftarrow ((HL)');\ (HL)' \leftarrow (Q);$$

Z zaporedjem ukazov RLD (oziroma RRD), INC HL in DEC HL je mogoče oblikovati poljubno dolg 4-bitni vrtilni register v pomnilniku. Oglejmo si subrutino za vstavitev heksadecimalne številke (na nižjih štirih bitih) v A na najnižje mesto večzložnega operanda. Imamo:

```
INSA:; (HL) je začetni naslov včzl operanda,
;      (C) je dolžina operanda (štev zlogov),
;      (A) je heksadecimalna številka (sp 4 b)
;      register B je aktiven
 LD   B,0  ; nastavi kazalec HL
 ADD  HL,BC;     na konec operan-
 DEC  HL   ;     dnega registra
 LD   B,C  ; nastavi B za izvajanje
 DEC  B    ;     večzložnega pomika
NAZAJ:; vstavi številko (A) z uporabo
 RLD       ;     štiribitnega levega
 DEC  HL   ;     pomika celotnega več-
 DJNZ NAZAJ;     zložnega operanda
 RLD       ;     določene dolžine
 RET       ; vrni se
```

Ko izstopimo iz INSA imamo v A najvišje mesto operanda; tako je mogoče to subrutino uporabiti za oblikovanje večzložnega, v levo vrtilnega ali pomikalnega registra heksadecimalnih številk.

Tudi pomike 16-bitnih registrov lahko dosežemo dokaj enostavno. Za pomik vsebine za en bit v levo v registru DE imamo

```
 SLA   E   ; pomakni (E) v levo z ničlo
 RL    D   ; pomakni (D) v levo s prenosom
```

Za register HL pa je najkrajša oblika levega pomika

```
 ADD   HL,HL   ; pomakni (HL) v levo z ničlo
```

Za desni pomik pa imamo

```
 SRA   H       ; pomakni (H) desno z ničlo
 RR    L       ; pomakni (L) desno s prenosom
```

Dober primer uporabe pomikalnega in vrtilnega ukaza je subrutina za množenje 16-bitnih besed, ko imamo (HL) = (BC).(DE):

```
MNOŽ:;
 LD   HL,0     ; anuliraj (HL)
MZAN:;
 SRL  B        ; pomakni (BC) v desno
 RR   C        ;    z ničlo
 JR   NC,NISE  ; preskoči delni produkt
 ADD  HL,DE    ; (HL)=(HL)+(DE)
NISE:;
 EX   HL,DE    ; pomakni (DE) v levo
 ADD  HL,HL    ;
 EX   HL,DE    ;
 LD   A,B      ; če je (BC)=0, končaj
 OR   C        ;
 JR   NZ,MZAN  ; sicer nadaljuj do konca
 RET           ; izstopi
```

Pomikalni in vrtilni ukazi nad ((HL)), ((IX+D)) in ((IY+D)) so izredno pripravni za konstruiranje aritmetičnih rutin s praktično poljubno stopnjo natančnosti (odvisno od prostora in časa). Brez težav je tako mogoče napisati rutine za 64-bitno aritmetiko (vključno z znanimi standardnimi funkcijami); to pa pomeni, da imamo na μR tudi možnosti največjih računalnikov.

Druga možnost za uporabo vrtilnih in pomikalnih ukazov se ponuja pri prenosu korekcijskih kodov, tj. pri prenosu podatkov na daljavo.

Na koncu tega poglavja si oglejmo še subrutino za izpis akumulatorja v obliki dveh heksadecimalnih številk. V tej subrutini so uporabljeni ukazi RRCA za desno vrtenje vsebine akumulatorja. Imamo

```
IZPA:; heksadecimalni izpis akumulatorja
 PUSH AF       ; ohrani AF
 RRCA          ; zavrti (A) v desno
 RRCA          ;    štirikrat
 RRCA          ;
 RRCA          ;
 CALL INVPR    ; uporabi inverzno pretvorbo
 POP  AF       ; vrni AF
INVPR:; pretvorba heksdec v ASCII
 PUSH AF       ; ohrani AF
 AND  0FH      ; odreži gornje 4 bite
 CP   0AH      ; če je v A vrednost 0,...,9
 JR   C,NIPR   ;    pojdi v NIPR
 ADD  A,7      ;    sicer prištej 7
NIPR:; oblikuj končno obliko ASCII
 ADD  A,30H    ; prištej 30H
 CALL TTYIZ    ; izpiši heks številko
 POP  AF       ; vrni AF, ki je nespremenjen
 RET           ; izstopi
```

Ta subrutina združuje dve subrutini, oziroma ima dve vstopni točki: IZPA in INVPR. Namesto, da bi v IZPA drugič poklicali INVPR, vstopimo v njo neposredno. Tu upoštevamo predvsem prostorsko optimizacijo.

Z uporabo subrutine IZPA lahko zapišemo še subrutino za izpis poljubno dolgega operanda oziroma polja, ko imamo:

```
IZPIS:; (HL) je začetni naslov včzl polja
;       (C) je dolžina polja (štev zlogov)
;       register B je aktiven
 LD   B,C      ; nastavi B za izpis
NAZA:;
 LD   A,(HL)   ; vstavi zlog v AKU
 CALL IZPA     ; izpiši zlog v heksadec obliki
 INC  HL       ; nastavi HL na naslednji zlog
 DJNZ NAZA     ; ponovi izpis do konca
 RET           ; izstopi
```

7. Aritmetični, logični in izmenjevalni ukazi

Oglejmo si najprej 8-bitne aritmetične in logične ukaze. Glede na procesor 8080 je ta zaloga ukazov razširjena na registra IX in IY. Vobče velja za te ukaze

OPERACIJA A, REGISTER;

kjer se A izpušča in se piše okrajšano kar OPERACIJA REGISTER. REGISTER je A,B,C,D,E,H, L,(HL),(IX+D),(IY+D),n, kjer je n 8-bitna konstanta.

Z logičnimi operacijamo lahko registre tudi preizkušamo in jim nastavljamo vrednosti. Za preizkus (BC)=0 uporabimo npr.

```
 LD   A,B      ; (B) pride v A
 OR   C        ; (A) = (B) OR (C)
 JR   Z,YES    ; če (A)=0 pojdi v YES
```

Akumulator anuliramo kratko z

```
XOR  A        ; (A) XOR (A) je vedno =0
```

Čeprav so ukazi z operandi (HL), (IX+D) in (IY+D) medseboj funkcionalno enakovredni, si velja zapomniti, da so ukazi z operandom (HL) enozložni, z (IX+D) in (IY+D) pa trizložni. 8-bitna aritmetična/logična ukazna zaloga je tedaj:

ADD, ADC (s prenosom), SUB, SBC (s prenosom), AND, XOR, OR, CP (primerjava), INC, DEC

Posebni ukazi pa so:

DAA (decimalna nastavitev AKU), CPL (eniški komplement), NEG (dvojiški komplement, ki je dodan), CCF (komplementiranje zastavice prenosa), SCF (postavitev zastavice prenosa)

16-bitni aritmetični ukazi imajo obliko

```
ADD  REGISTER, REGISTER; in
INC/DEC  REGISTER;
```

Tu je register vselej 16-bitni: BC,DE,HL,IX,IY SP. Za seštevanje in odštevanje s prenosom je možna le oblika

```
OPERACIJA  HL, REGISTER;
```

Niso pa možne vse kombinacije registrov. Odštevanje 16-bitnih registrov je vedno odštevanje s prenosom, zato moramo v posameznih primerih anulirati zastavico prenosa. Npr.

```
OR   A        ; (A) se ne spremeni, C-zasta-
;                 vica pa pade ( =0)
SBC  HL,DE    ; (HL)=(HL)-(DE)
```

Večina 16-bitnih aritmetičnih ukazov je glede na procesor 8080 novih (izjema so ukazi INC, DEC in ADD HL, REGISTER;).

Zaloga izmenjevalnih ukazov je občutno narasla. Izmenjava se izvaja med registri R in njihovimi dvojniki R´. Npr.

```
EX   AF,AF´ ; AF <--> AF´
```

Najmočnejši izmenjevalni ukaz je

```
EXX           ; BC <--> BC´; DE <--> DE´;
;               HL <--> HL´
```

ki povzroči izmenjavo med dvanajstimi osnovnimi registri. Zanimiva nova izmenjevalna ukaza sta

EX (SP),IX; in EX (SP),IY;

ki omogočata izmenjavo vsebine dveh lokacij sklada z vsebinama registrov IX in IY. Tako je:

```
EX  (SP),IX ; IX_visoki <--> (SP+1),
;             IX_nizki  <--> (SP)
```

Vrednost (SP) ostane pri tem nespremenjena. Ta ukaza je mogoče uporabiti namesto zaporedja PUSH in POP ukazov.

Imejmo sklad ABC (C je na vrhu sklada), ko želimo C vzeti iz sklada v register IY in naložiti D na njegovo mesto, tako da dobimo sklad ABD. D se nahaja tudi v IY. Za to izmenjavo je potreben en sam EX ukaz. V PUSH/POP tehniki pa izgleda programski segment takole:

```
POP  DE     ; C pride v DE
PUSH IY     ; D gre v sklad
PUSH DE     ; C gre začasno v sklad in
POP  IY     ;   in odtod v IY
```

Uporabili smo pomožni register DE. Procesor 8080 ima samo dva izmenjevalna ukaza:

EX DE,HL; in EX (SP),HL;

8. Ukazi za vhod in izhod

Zaloga teh ukazov je pri Z80 zares pestra. Najprej imamo tele tipe ukazov:

IN A,(n); IN r,(C); OUT (n),A; OUT (C),r;

kjer je r = A,B,...,L, n in (C) pa sta številki vrat (dejansko samo spodnjega dela številke). Za gornje ukaze velja npr.

A <-- (n), r <-- ((C)), n <-- (A), (C) <-- (r)

Procesor Z80 nima svojih vrat. Pri izvedbi IN/OUT ukazov se ustrezno nastavi naslovno vodilo in vpliva tako na dekodiranje vrat. Konstanta n in vsebina (C) oblikujeta naslovne bite A0, ...,A7, registra A in B pa dodatno (po potrebi) bite A8,...,A15. Tako dobimo z n in (C) 256 naslovov vrat, z dodatnimi osmimi biti pa 64K vrat.

Ukazi:INI, INIR, IND, INDR, OUTI, OTIR, OUTD, OTDR. Vsi ti ukazi imajo prenos vsebine med (C) in (HL) torej med periferijo in pomnilnikom. Zadnji ukaz ima tale pomen:

(C) <-- ((HL)), B <-- (B)-1, HL <-- (HL)-1

in se ponavlja do B=0. Tako lahko pošljemo z enim samim ukazom 256 zlogov na zunanjo napravo. Kazalec (HL) se dekrementira ali inkrementira, ukazi brez končnice R pa so ustrezni enokoračni ukazi (neponavljalni ukazi, glej poglavje 3).

9. Sklep

V sklepne opombe sodi vse tisto, kar je za programiranje na sistemih z Z80 bistveno, pa tega vsled pomanjkanja prostora nismo obširneje opisali.

Prva pomanjkljivost zbirnika za Z80 je, da je potrebno pri vseh relativnih skokih JR za operand OZNAČITEV pisati izraz

OZNAČITEV-$

de se relativni prehod (skok) izračuna.

V sestavku nismo opisali možnosti programiranja s prekinitvami (npr. generiranje realnega časa v posebnem kanalu); takšni programi so v sistemih z Z80 možni, če imamo v konfiguraciji element CTC. O tem kaj več drugič.

Pri majhnih konfiguracijah so t.i. sistemski programski dodatki osnovnega pomena zlasti za začetni razvoj programske opreme. Prisiljeni smo, da si večkrat naredimo dodatno orodje sami. Monitorji (majhni operacijski sistemi) so dostikrat pomanjkljivi (zaradi štednje s pomnilnim prostorom) in naravnani na širjenje sistema; takšne možnosti pa so v začetni fazi dostikrat brez praktične vrednosti (različni programski vmesniki za pogon periferije) Navadno tako nimamo na voljo niti zbirnika, premestljivega nalagalnika in urejevalnika. Torej je razvoj nekaterih pomožnih sistemskih dodatkov v začetni fazi nujen. To so programi za nazorno ročno (interaktivno) nalaganje (ko program prvič vpisujemo v pomnilnik), preizkušanje (npr. po posamičnih ukaznih korakih z indikacijo splošnih registrov) in prikazovanje (npr. strojnega koda v ukaznem formatu).

Z80 je najmočnejši mikro procesor in prav je, da ga začnemo uporabljati v procesnih okoljih, še zlasti tedaj, ko je povezan z dinamičnimi pomnilniki (zaradi visoke zanesljivosti transparentnega osveževanja), ko se zahteva hiter odziv in obdelava prekinitev (osnovna taktna frekvenca procesorja je 4 MHz) in ko zahtevamo učinkovito in dovolj prožno jedro operacijskega sistema.

Sistem z Z80 je pripraven tudi za poučevanje inženirskega programiranja zaradi močnih in raznovrstnih ukazov, ki omogočajo strukturiran pristop (rekurzivno pozivanje) ter časovno/prostorsko optimizacijo programov.

Literatura

(1) T.Dollhoff, Techniques for the Zilog Z80, Digital Design, Feb.1977,44.
(2) M.Shima, F.Faggin, R.Ungermann, Z-80 Chip Set Heralds Third Microprocessor Generation, Electronics, Aug.1976,89.
(3) Z80 Technical Manual, Mostek, Aug.1976.
(4) Z80-Assembly Language Programming Manual, Zilog, 1977.

primjena monitorskog koncepta u izgradnji operacionog sistema za periodično aktiviranje programa

branislav popović
matija exel
milan mekinda

UDK 681.3.06

Iskra Telekomunikacije, Kranj
Institut Jožef Stefan, Ljubljana
Iskra Telekomunikacije, Kranj

U članku je prikazan pristup ka izgradnji operacionog sistema koji omogućava periodično aktiviranje programa, pomoću koncepta monitora iz paralelnog programiranja. Takav operacioni sistem može da poslužuje svaki sistem za otkrivanje i obradjivanje velikog broja dogadjaja - signala, kao što su sistemi za upravljanje procesima, a naročito komutacijski sistemi telekomunikacijskih mreža. Algoritam je prikazan u Pascal jeziku za paralelno programiranje (3).

AN APPLICATION OF THE MONITOR CONCEPT TO THE OPERATING SUBSYSTEM FOR PERIODIC PROGRAMS ACTIVATIONS

An approach to operating system design enabling periodic programs activations is given in the paper. In the design , the monitor concept of concurrent programming is used. The described operating system can be used for those real time systems detecting and handling large amounts of events or signals - such as process control systems. But the most representative of such is a stored program controlled switching system treating line and register signalling on enormous number of trunks. Algorithm is shown by Pascal concurrent programming language (3).

1. Uvod

Telekomunikacijski sistemi poseduju veliki broj priključnih tačaka na kojima otkrivaju dogadjaje - signale iz okoline sistema. Priključne tačke posmatramo pomoću skanirnih programa koje uključujemo u odredjenim periodama. Period skaniranja priključnih tačaka zavisi od učestalosti i/ili od trajanja dogadjaja - signala. Učestalošću većom od potrebne ne dobijamo bogatiju informaciju o signalima, a trošimo procesorsku sposobnost, dok suviše mala učestalost prouzrokuje gubitak dogadjaja ili pogrešnu identifikaciju signala. Zato su periodi uključivanja skanirnih programa date vrednosti sistema od kojih ne sme biti odstupanja za vreme rada niti zbog suviše velikog saobračaja kojeg mora sistem posluživati.

U sistemu imamo više tipova priključnih tačaka. Broj tipova je broj različitih linijskih i registarskih signalizacija. Za svaki tip priključne tačke postoji poseban skanirni program.

2. Matrica za periodično aktiviranje programa

Neka u sistemu postoji m skanirnih programa:

$p_1 , p_2 , \ldots , p_i , \ldots , p_m$

koji se moraju izvršavati svakih

$r_1 , r_2 , \ldots , r_i , \ldots , r_m$

taktova. Taktovi su hardverski prekidi koji se deŝavaju u stalnim vremenskim razmacima. Razmak izmedju taktova iznosi T vremenskih jedinica. Program p_i izvodi se dakle svakih r_i T jedinica vremena, gde je r_i prirodan broj za svaki indeks i = 1 .. m.

Aktiviranje programa može se odrediti parom

(identitet programa : p_i , brojač$_i$)

brojač$_i$ povečava se pri svakom taktu za 1. Kad njegova vrednost postaje jednaka faktoru perioda r_i izvodjenja programa p_i , brojač$_i$ stavlja se na 0, a program p_i se aktivira.

Aktiviranje programa može se takodje odrediti Boolovom matricom zahteva. Od prvog načina ka drugom možemo doći jednostavnom transformacijom (1).

U prikazanom pristupu upotrebili smo drugi način.

Matrika zahteva (označimo je na ovom mestu sa M) sastoji iz zahteva δ^i_k, gde je k indeks reda koji je ujedno broj takta, a i je indeks kolone, koji je istovremeno i indeks programa:

$M = [\delta^i_k]$, a δ^i_k odredjen je sa:

$$\delta^i_k = \begin{cases} \underline{\text{false}}, & \text{ako se program } p_i \text{ ne izvodi u taktu k} \\ \underline{\text{true}}, & \text{ako se program } p_i \text{ izvodi u taktu k} \end{cases}$$

Broj redova u matrici ograničavamo tako da uzimamo cikličnu matricu sa brojem redova n, koji je najmanji zajednički sadržioc skupa $r_1, r_2, \ldots, r_m$.

Za primer uzmimo samo dva programa p_1 i p_2 : p_1 se izvršava svaka dva takta ($r_1 = 2$), a p_2 se izvršava svakih tri taktova ($r_2 = 3$). Najmanji zajednički sadržioc je 6, pa je matrica dugačka 6 redova.

Moguća konfiguracija matrice je

(f stoji za false, a t za true) :

ft
tf
ff
tt
ff
tf

Ova matrica odredjuje da se u prvom taktu vrši program p_2, u drugom p_1, u trećem nijedan program, u četvrtom p_1 i p_2, u petom nijedan, a u šestom p_1. Zahteve za sedmi takt možemo naći ponovo u prvom redu, za osmi u drugom redu i tako redom. Znači, broj takta k po modulu od 6 je indeks reda, u kome možemo naći koji programi se moraju izvršiti u taktu k.

Očito važi za elemente matrice sledeća relacija:

ako je $\delta^i_k = \underline{\text{true}}$, onda je $\delta^i_{(k+r_i) \underline{\text{mod}}\ n} = \underline{\text{true}}$

Zato je i - ta kolona u M odredjena sa $sk_i < 0$, gde je $|sk_i| \leq r_i$ te sa r_i po jednačini:

$$\delta^i_k = \begin{cases} \underline{\text{false}} \text{ za k}, & (k-sk_i) \underline{\text{mod}}\ r_i \neq 0 \\ \underline{\text{true}} \text{ za k}, & (k-sk_i) \underline{\text{mod}}\ r_i = 0 \end{cases}$$

za svaki k = 1 .. n.

U gore pokazanom primeru $sk_1 = -1$, dok je $sk_2 = -3$.

Tako je u stvari matrica M odredjena jednolično faktorima r_i i sk_i za sve programe p_i (i = 1 .. m). Pošto su r_i za dati sistem konstante, konfiguracija matrice zavisi samo od sk_i, koji se mogu odrediti tako da imamo ravnomerno opterećenje svakog intervala izmedju taktova (1).

3. Glavne crte operacionog sistema

U ovom poglavju pokazati ćemo grube obrise operacionog sistema, potrebne za razumevanje algoritma za periodično aktiviranje programa. Detaljnija struktura sistema izložena je u (2).

Dotaći ćemo se samo onih delova operacionog sistema koji poslužuju programe za ispitivanje priključnih tačaka.

Sistem je zamišljen kao skup unutrašnjih sekventnih procesa, koji cikliraju beskonačno. Saradjivanje procesa obezbedjeno je skupom hierarhično uredjenih monitora koji upravljaju sredstvima (hardverskim kolima, procesorskim vremenom i slično)(5).

Monitor je sintaksni konstrukt pomoću kojeg definiramo podatkovnu strukturu (pretstavljajuću odredjeno sredstvo) i skup operacija (monitorskih procedura) nad podacima te strukture. Monitorske procedure odredjenog monitora implementirane su tako da omogućavaju istodobno pozivanje sa strane različitih procesa. Monitor takodje definiše početne operacije koje se moraju izvršiti u svrhu inicializacije podatkovne strukture monitora.

Za sve testne procese koji ispituju priključne tačke definišemo tip testniproces.

```
type  testniproces =
process (mon vreme {; monitori za upravljanje
                    ostalim sredstvima } )
  { deklaracije tipova i promenljivih }
cycle
repeat
 {kod za testiranje ispitnih tačaka i za komu-
  nikaciju ostalim procesima }
```

```
until { sve priključne tačke ispitane } ;
monvreme.čekati (indeksprocesa){ monitor monvreme je opisan u poglavju 4 }
end { testniproces }
```

Testniproces ispituje priključne tačke sve dok nije ispitao sve tačke istog tipa. Inače je u stanju čekanja sve dok ga ponovo aktiviramo.

Monitor za upravljanje procesorom nazovimo monproc kojeg definišemo na sledeći način:

```
var monproc { monitor za upravljanje procesorom } :
monitor ({ globalni podaci })
{ deklaracije potrebnih tipova i promenljivih}
procedure entry dodeliti ;
begin
{ izabere sledeći proces maksimalnog prioriteta iz liste "spremnosti", stavlja izabrani proces u stanje "aktuelni" }
end
procedure entry odgoditi;
begin
{ "aktuelni" proces se zaustavlja i stavlja u listu "čekanja" odgovarajućeg prioriteta }
end
procedure entry spremiti (i : indeksprocesa)
begin
{ proces sa indeksom i stavlja se iz liste "čekanja" u listu "spremnosti" najvišeg prioriteta }
end
begin
{ inicializacija lista "čekanja" i "spremnosti"
end { monproc }
```

4. Monitor za aktiviranje testnih procesa u zavisnosti od vremena

Monitor koji aktivira testne procese u zavisnosti od vremena nazovimo monvreme. Ovim monitorom "aktuelni" - tekući - proces se zaustavlja (pomoću pozivanja procedure "čekati") i aktiviraju se procesi (pomoću pozivanje procedure "aktivirati") koji se moraju izvršiti u tekućem taktu.

Proceduru "aktivirati" za aktiviranje programa - procesa - zove proces hardverskih taktnih prekida.

```
var monvreme { vremenski monitor } :
monitor (monproc { ; globalni podaci } );
type boolvekt = array [ 1.. m {jednak broju testnihprocesa}] of Bolean
var zahtevi : array [1.. n {najmanji ukupni sadržioc faktora perioda}] of boolvect
   { to je u stvari matrica za periodično aktiviranje programa };
    dosadzahtevi : boolvekt ;
    završeniciklusi : boolvekt ;
    dozvoljenizahtevi : boolvekt ;
    brojačtaktova : 1 ..n {maksim. dužina matrice } ;
procedure entry čekati (j : integer) ; { zove se iz testnih procesa }
begin
monproc. odgoditi ;
završeniciklusi (j) : = true ;
monproc. dodeliti
end ;

procedure entry aktivirati; { zove se iz procesa hardverskih taktnih prekida }
begin
brojačtaktova : = (brojačtaktova + 1) mod n ;
for i : = 1 to m do
    begin
    dosadzahtevi [i] : = dosadzahtevi [i] or
    zahtevi [brojačtaktova][i]
   {dosadzahtevi sadrži sve zahteve do sada koji se nisu mogli ostvariti jer su bili procesi blokirani, pa se njihov ciklus nije mogao završiti. Ovim zaostalim zahtevima dajemo još zahteve za tekući takt.} ;
    dozvoljenizahtevi[i]: = dosadzahtevi[i]and
    završeniciklusi [i] {dozvoljenizahtevi sada sadrže sve dosadašnje zahteve {dosadzahteve} za procese čiji ciklusi su se završili, pa čekaju na ponovno aktiviranje nakon protečenog perioda vremena.};
    monproc. odgoditi
    {" aktuelni" proces smo prekinuli i stavili ga u odgovarajuću listu "čekanja" };
    if dozvoljenizahtevi [i] then
       begin
       monproc. spremiti [i] ;
       završeniciklusi [i] : = false ;
       dosadzahtevi [i] : = false ;
       end {if}
   end {for}
monproc. dodeliti
 {izabere proces iz liste "spremnosti" sa najvišim prioritetom i stavlja ga u stanje "aktuelni"}
end {aktivirati} ;
```

```
begin {inicializacija monvreme}
for  i  : = 1 to m do
   begin
   završenicikl.si[i]: = true ;
   dosadzahtevi [i]  : = false
   end  {for} ;
brojačzahteva : = 0 ;
{ inicializacija matrice zahteva za periodično
aktiviranje programa }
end { monvreme }
```

5. Diskusija

Ako uredimo indekse programa - procesa u rastućem redu po opadajućem prioritetu, zbog zakonitosti for do petlje, aktiviramo programe takodje u redu odpadajućeg prioriteta. Na taj način obezbedili smo da kritične priključne tačke obilazimo u traženom periodu vremena, dok za procese manje kritičnih priključnih tačaka dopuštamo odstupanje u granicama tolerancija.

Algoritam je prikazan u Pascal jeziku za paralelno programiranje (3, 4). Medjutim primitivi tog jezika mogu se realizovati i u drugim jezicima (6), pa se algoritam može izvesti u drugim programskim jezicima pogodnim za programiranje sistema realnog vremena.

6. Literatura

1) Popović B.; Optimizacija programskog monitora kod sistema realnog vremena; V.Jugoslovensko savetovanje o informacionim sistemima; Zbornik radova, maj 1976.
2) Exel M., Mekinda M., Popović B.;Structuration d'un système telephonique informatise; AFCET, Pariz, 1977.
3) Brinch-Hansen P.; The programming language Concurrent Pascal; IEEE Transactions on Software Engineering, vol. se-1, no.2, june 1975.
4) Brinch-Hansen P.; Concurrent Pascal Report; Information Sciense; California Institute of Technology; june 1975.
5) Hoare C.A.R.; Monitors : an operating system structuring concept; CACM, vol.17,no.10,1974.
6) Mc Gowan C.L., Kelly J.R.; Top-down structured programming techniques; Petrocelli/ Charter; N.Y. 1975.

cleaning up unstructured algorithms using invariants

suad alagić

UDK 681.3.05/. 06

Elektrotehnički fakultet,
71000, Sarajevo-Lukavica

In this paper we attempt to show that the method of invariants is often a very good technique for the design of a well-structured algorithm on the basis of an unstructured one. This transformation is based on a careful analysis of the invariants underlying the original solution. The result of the transformation is an algorithm which is well-structured and whose proof of correctness is immediate from the way this transformation is performed.

ČIŠČENJE NESTRUKTURIRANIH ALGORITAMA POMOĆU INVARIANTI - U ovom članku pokušavamo pokazati kako su invarijantne metode često vrlo dobra tehnika za projektiranje dobro strukturiranih algoritama na osnovi nestrukturiranih. Ova transformacija je zasnovana na pažljivoj analizi invarianti koje baziraju na originalnim rješenjima. Rezultat transformacije je algoritam koji je dobro strukturiran i čiji dokaz korektnosti sljedi direktno iz postupka transformacije.

It has been realised for a while that the method of invariants is a powerful technique for the development of well-structured and correct algorithms (Hoare). However, the majority of well-known and widely used algorithms in computer science literature were invented before the discovery of the main principles of modern programming methodology. Many of them suffer seriously from all sorts of structural irregularities and obscurities. Because of that it is often very difficult to understand why such algorithms work and whether they are really correct. Problems also arise in modifying such algorithms to satisfy specific needs which may be somewhat different from the ones for which these algorithms were originally designed. So what we need are techniques for restructuring poorely structured algorithms into the well-structurd ones.

In this paper we attempt to show that the method of invariants is often a very good technique for the design of a well-structured algorithm on the basis of an unstructured one. This transformation is based on a careful analysis of the invariants underlying the original solution. The result of the transformation is an algorithm which is well-structured and whose proof of correctness is immediate from the way this transformation is performed. This technique is demonstrated here by restructuring the binary algorithm for computing the greatest common divisor of two nonnegative integers.

Recall that if x and y are integers, not both zero, then their greatest common divisor,denoted by gcd(x,y), is the largest integer which divides both of them (that is, without remainder). Observe that this definition makes sense, i.e., such an integer does exist. Indeed, if $y \neq 0$, then no integer greater than $|y|$ (the absolute value of y) can divide y. On the other hand, the integer 1 divides both x and y, and so there must be a largest integer which divides them both. When both x and y are zero, the above definition does not apply, since every integer divides 0. We shall set

$$\gcd(0,0)=0 \qquad (1)$$

It is obvious that the following properties follow from the above definitions:

$$\gcd(u,v)=\gcd(v,u) \qquad (2)$$

$$\gcd(u,v)=\gcd(-u,v) \qquad (3)$$

$$\gcd(u,0)= u \qquad (4)$$

Because of the properties (2) and (3) we can actually concentrate on the problem of finding the greatest common divisor of two non-negative integers a and b. Indeed, if we know how to solve that problem, then in order to find the greatest common divisor of arbitrary integers a and b, we just apply that method to $|a|$ and $|b|$.

A modern version of the Eucledean algorithm for computing the greatest common divisor of non-negative integers a and b is given in (5).

```
begin {a ⩾ 0, b ⩾ 0}
      x:=a; y:=b;
      {gcd(a,b)=gcd(x,y)}
      while y≠0 do
      begin r:=x mod y;
            x:=y;                    (5)
            y:=r
      end
      {x=gcd(a,b)}
end
```

The binary gcd algorithm (Knuth) given in (10) is based on the following four further properties of positive integers u and v:

If u and v are both even, then
gcd(u,v)=2gcd(u div 2, v div 2) (6)

If u is even and v is odd, then
gcd(u,v)=gcd(u div 2, v) (7)

If u>v then gcd(u, v)=gcd(u-v, v) (8)

If u and v are both odd, then u-v is even and |u-v|<max(u,v) (9)

```
begin
  x:=u; y:=v;
  k:=1;
  1: if even(x) ∧ even(y) then
     begin x:=x div 2; y:=y div 2 ;
           k:=k*2;
           goto 1
     end;
     if odd(x) then
     begin t:=-y; goto 3 end
     else t:=x;                          (10)
  2: t:=t div 2;
  3: if even(t) then goto 2;
  if t>0 then x:=t else y:=-t;
  t:=x-y;
  if t≠0 then goto 2 else x:=x*k
  {gcd(u,v)=x}
end
```

The structure of the algorithm (10) can be improved in an obvious manner by discovering the while-do and repeat-until loops in it. This leads to the algorithm (11).

The algorithn (11), although much clearer than the algorithm (10), and without any jumps, is still not too easy to understand. A clearer algorithm is obtained if one tries to derive it directly from the properties (6), (7), (8), and (9). Such an algorithm is presented in (12).

```
begin x:u; y:=v;
      k:=1;
      while even(x) ∧ even(y) do
      begin x:=x div 2; y:=y div 2;
            k:=k*2
      end; {(gcd(u,v)=k*gcd(x,y))
           ∧(odd(x) ∨ odd(y))}
     if odd(x) then t:=-y else t:=x;
     repeat
           while even(t) do t:=t div 2;
           if t>0 then x:=t else y:=-t;
           {odd(x) ∧ odd(y)}
           t:=x-y
     until t=0;
     x:=x*k;
     {gcd(u,v)=x}
end
                    (11)
begin x:=u; y:=v;
      k:=1;
      while even(x) ∧ even(y) do
      begin x:=x div 2; y:=y div 2;
            k:=k*2
      end; {gcd(u,v)=k*gcd(x,y)}
      repeat {odd(x) ∨ odd(y)}
            while even(x) do x:=x div 2;
            while even(y) do y:=y div 2;
            {odd(x) ∧ odd(y)}
            if x>y then x:=x-y
            else y:=y-x
      until (x=0) ∨ (y=0);
      if x=0 then gcd:=y*k else gcd:=x*k
end {gcd =gcd(u,v)}
                    (12)
```

The algorithm (12) is derived on the basis of the following considerations:

i. Because of the property(6) after the loop

```
while even(x) ∧ even(y) do
begin x:=x div 2; y:=y div 2;
      k:=k*2
end                                     (13)
```

gcd(u,v)=k*gcd(x,y) holds.

ii. The loops

```
while even(x) do x:=x div 2;
while even(y) do y:=y div 2              (14)
```

do not affect the value of gcd(x,y).This follows from the property (7) and the fact that whenever the loops (14) are reached odd(x) ∨ odd(y) holds. This is certainly true the first time the loops (14) are reached, since that happens upon exit from the loop (13). Upon completion of execution of the loops (14) odd(x) ∧ odd(y) holds, and after the statement if x>y then x:=x-y else y:=y-x x will be odd and y will be even, or vice versa. This follows from(9). So whenever, later in the repeat-until loop in (12), the loops (14) are reached, either x or y will be odd.

iii. The step if x>y then x:=x-y else y:=y-x is justified by (8) and (9) and of course (2). It follows from (8) and (2) that the statement if x>y then x:=x-y else y:=y-x does not affect the value of gcd(x,y).This statement reduces either x or y , but leaves them non-negative, so that after some number of iterations the condition (x=0) ∨ (y=0) must be satisified.

A further modification of the algorithm (12) is possible. It is based on the substitution of the statement if x>y then x:=x-y else y:=y-x by the loops

```
while x>y do x:=x-y;
while y>x do y:=y-x                      (15)
```

This is again justified by (8) and (2). And now on the basis of gcd(u,u)=u we can substitute in (12) the test (x=0) ∨ (y=0) by x=y, to obtain the following algorithm (16) :

```
begin x:=u; y:=v;
      k:=1;
      while even(x) ∧ even(y) do
      begin x:=x div 2; y:=y div 2;
            k:=k*2
      end; {gcd(u,v)=k*gcd(x,y)}
      repeat while even(x) do x:=x div 2;
             while even(y) do y:=y div 2;
             {odd(x) ∧ odd(y)}
             while x>y do x:=x-y;
             while y>x do y:=y-x
      until x=y
      {gcd(u,v)=x*k}
end
                                         (16).
```

REFERENCES

Alagić, S. (1976): Principles of Programming (In Serbo-Croatian). Svjetlost - Sarajevo.

Gries, D. (1977): On beleiving programs to be correct. Communications of the ACM, Vol. 20, pp. 49-50.

Hoare, C.A. R. (1971): Proof of a program: Find. Communications of the ACM, Vol. 14, pp. 39-45.

Hoare, C.A. R. and Wirth, N.(1973): An axiomatic definition of the programming language Pascal. Acta Informatica 2, pp. 335-355.

Knuth, D. E. (1969): The Art of Computer Programming: Vol. 3/ Sorting and Searching. Adison Wesley.

Wirth, N. (1974): On the composition of well - structured programs. Computing Surveys 6, pp. 247-259.

ACKNOWLEDGMENT

Research reported in this paper was supported by Zajednica za naučni rad SR Bosne i Hercegovine.

Informatica št. 2 letnik 1977

monitori za mikro sisteme sa procesorom 6800

m. kovačević
d. novak
a. p. železnikar

UDK 681.3-181.4.06

Institut "Jožef Stefan",
Ljubljana

U članku se obradjuju različiti koncepti monitora za mikro računarske sisteme sa procesorom 6800. Opisane su ugodnosti koje pruža taj procesor za realizaciju monitora. Članak uključuje dva primjera monitora za istu konfiguraciju sistema od kojih je jedan napisan u asemblerskom jeziku; drugi u obliku pseudo koda.

MONITORS FOR MICROSYSTEMS USING 6800 PROCESSOR - The article deals with several monitor concepts for microcomputer systems using the 6800 processor. Advantages of monitors for such systems are presented in details. This article includes two monitor examples for the same system configuration, the first one being written in the assembly language and the second one in pseudocode.

1. Uvod

Osnovna softverska oprema, koju mikroračunarski sistem mora imati da bi mogao izvršavati naredbe korisnika, mora u najjednostavnijem primjeru imati mogućnost pokazivanje i promjene sadržaja odredjene memorijske lokacije, unošenje objektnog programa u memoriju, ispis sadržaja registara centralne procesne jedinice (CPU) u trenutku prekida izvodjnja korisničkog programa te prenošenja kontrole nad sistemom na korisnički program. Takav monitor nam već omogućava razvijanje i popravljanje programa iako bi to bio mukotrpan posao jer imamo vrlo ograničene mogućnosti za testiranje korisničkih programa i manipulaciju podatcima zapamćenim u memoriji. Monitori predstavljaju nadgradnju hardverskog sistema u mikro računarima. I tako dobre hardverske karakteristike nekoga sistema nije moguće dovoljno iskoristiti bez odgovarajućeg monitora. Posebno važno mjesto medju monitorskim naredbama imaju naredbe za testiranje korisničkih programa (breakpoint, ispis registara CPU) i heksadecimalna aritmetika koju često upotrebljavamo pri popravljanju koristničkih programa.
U skup monitorskih naredbi mikro računara spadaju još i naredba za ispis sadržaja odredjenog memorijskog bloka, naredba za bušenje objektnog koda (mašinskog koda) na papirnoj traki u odgovarajućem formatu, naredba za kopiranje sardžaja odredjenog memorijskog bloka na drugu adresu te naredbe za programiranjePROM memorija (ukoliko konfiguracija sistema sadrži odgovarajuće hardverske elemente). Pored mogućnosti izvršavanja tih osnovnih naredbi namijenjenih komuniciranju čovjek-mašina monitor nam često pruža i druge servise. U korisničkim programima se vrlo često pojavljuje problem čitanja i izpisivanja podataka na različitim perifernim napravama. Monitor treba da sadrži podprograme i drajvere za različite periferne naprave čime je problem čitanja i izpisivanja podataka riješen. Pored podprograma i drajvera za I/O komuniciranje monitor često sadrži i obilje podprograma, namijenjenih za vanjsku upotrebu, koji obradjuju odredjene elemente iz problemskog prostora u kojem djeluje mikro računarski sistem.

2. Različiti koncepti monitora

Minimalni sistem koji je potreban za implementaciju monitora sa opisanim naredbama mora, pored centralne procesne jedinice, imati najmanje 1 K bajtnu ROM (ili EPROM) memoriju, o,5 K bajtnu RAM memoriju te priključen teleprinter ili tastaturu sa monitorom za komuniciranje čovjek-mašina. U odvisnosti od odabrane hardverske konfiguracije odlučujemo se za najpogodniji koncept monitorskog programa. U članku ćemo se ograničiti na monitore pisane za konfiguracije sa ROM ili (EPROM) modulima namijenjenim za pamćenje sistemskih programa, te takvih korisničkih programa koji su često (ili stalno) u upotrebi. Diskovno ili kasetno orientirani monitori zahtijevaju drugačiji pristup.
Razlikujemo dva načina djelovanja monitora s obzirom na način prihvatanja naredbe. U jednom slučaju možemo jasno povući granicu izmedju ciklusa prihvatanja naredbe i ciklusa izvršavanja iste, dok u drugom slučaju ta granica nije tako jasna. Naime, svaki znak, koji se prihvati sa tastature, se interpretira na način kojeg odredjuje sintaksa monitorskih naredbi. Drugim rječima svakim na tastaturi natipkanim znakom biramo jedan od puteva kroz monitorski progam. U prvom slučaju je neophodno imati rezervisano odgovarajuće područje u RAM memoriji(buffer) za formiranje unutrašnje slike naredbe koja se skanira pri izvršavanju iste.
Razlike u sintaksi naredbi sa istim učinkom nam često doprinose prostorskoj optimizaciji monitorskog programa. Pri tome prostorska optimizacija nastaje na uštrb automatiziranosti i brzini izvodjenja naredbe. Na primjer naredba za prenos kontrole nad sistemom na korisnički program može biti sestavljena samo od jednog simbola (G). Dodatna, potrebna informacija za pravilno izvršenje te naredbe je adresa programa na kojeg treba da se prenese kontrola. Tu informaciju posredujemo monitoru odgovarajućom monitorskom naredbom tako da prethodno u odredjeni registar centralnog procesora ili lokaciju ju RAM memorije upišemo adresu programa. Očito je da možemo govoriti o naredbama potpunog oblika i o naredbama koje zahtijevaju predhodne monitorske akcije. Naredbe potpunog oblika sadrže sve informacije potrebne za pravilno

izvršenje naredbe.
Tipični oblik monitorske naredbe ima sljedeći oblik:

⟨mnemonik⟩ ⟨dodatne informacije⟩ ⟨CR⟩

Naredbe koje zahtijevaju predhodne monitorske akcije su obično sastavljene samo iz mnemoničnog dijela.

3. Primjer jednostavnog monitora

Predpostavimo sljedeću konfiguraciju mikro računarskog sistema: 0,5 K bajtna EPROM memorija, 1 K bajtna RAM memorija, teleprinter priključen posredstvom ACIA kompleksnog integriranog vezja (Asynchronous communications interface adapter - MC 6850) i centralna procesna jedinica sa procesorom 6800.
Sa tako skučenim memorijskim prostorom nije moguće realizirati monitor sa bogatom zbirkom naredbi. Zbog toga moramo iskoristiti sve mogućnosti za optimalno korištenje memorijskog prostora, kao na primjer nepotpuni oblik naredbi sa prethodno potrebnim dodatnim akcijama. Maksimalni mogući skup naredbi je sljedeći: pokazivanje i promjena sadržaja memorijske lokacije (M), unošenje objektnog programa u memoriju (L), ispis sadržaja registara centralne procesne jedinice u trenutku prekida izvodjenja korisničkog programa (R), i prenos kontrole nad sistemom na korisnički program (G).
Naredba M ima oblik:

M XXXX YY ZZ

Najprije natipkamo znak M čime odabiremo naredbu, zatim natipkamo četverocifarsku heksadecimalnu adresu memorijske lokacije (XXXX) čiji sadržaj želimo znati ili promijeniti. Automatski se na teleprinteru ispiše sadržaj iste lokacije u heksadecimalnoj notaciji (YY). Ukoliko želimo mjenjati sadržaj te lokacije moramo natipkati znak za prazno mjesto i zatim novi sadržaj te lokacije (ZZ). Po izvršenju te naredbe monitor čeka na sljedeću naredbu.
Naredba L ne zahteva nikakve dodatne informacije ili akcije. Neposredno zatim pošto smo natipkali znak L počinje čitanje papirne trake koja sadrži objektni program izdijeljen u rekorde sa sljedećim formatom:

SXYYAAAAHH.....CC

S je startni znak za rekord
X je identifikator koji označava prvi, zadnji ili rekord koji je izmedju prvog i zadnjeg
YY je heksadecimalni broj bajtova objektnog programa u odgovarjajućem rekordu
AAAA je heksadecimalna adresa lokacije u koju će se upisati prvi bajt iz rekorda
HH je bajt objektnog programa
CC je kontrolna suma svih bajtova objektnog programa iz odgovarajućeg rekorda.

Naredba R se sastoji samo iz znaka R. Neposredno zatim pošto smo natipkali znak R se ispišu, bajt po bajt, sadržine svih registara centralnog procesora u trenutku prekida korisničkog programa po sljedećem redu: CC,B,A,XH,XL,PH, PL,SH,SL
Naredba G se sestoji samo iz znaka G i zahtjeva predhodne akcije. Naime, prije nego što damo naredbu G moramo u odredjeno memorijsko područje upisati početne vrijednosti svih registara centralne procesne jedinice u trenutku starta korisničkog programa. Posebno važno je da upišemo početnu vrijednost programskog brojača (PC) u za to odredjene lokacije (002E, 002F) čime odredimo lokaciju od koje će se korisnički program izvoditi. U trenutku starta korisničkog programa se pomenuto memorijsko područje kopira u registre centralne procesne jedinice.

Monitorski program koji raspoznaje opisane naredbe sadrži tabela 1.
Opisani monitor nema takve osobine kakve često želimo. Glavni nedostatak je prije svega u tome da pri M naredbi moramo za svaku lokaciju tipkati njenu adresu. Poboljšana verzija takvog monitora bi morala imati mogućnost da pri tiskom na jednu tipku dobijemo sadržaj lokacije sa inkrementiranom adresom, a da pritiskom na drugu tipku dobijemo sadržaj lokacije sa dekrementiranom adresom. Dodatne naredbe za poboljšanu verziju monitora bi bile naredba za bušenje objektnog koda na papirnu traku u odgovarajućem formatu te naredba za kopiranje odredjenog memorijskog područja na odredjenu adresu.
Nacrt takvog poboljšanog monitora ima sljedeći oblik u pseudo kodu:

```
program MONITOR
INICIALIZACIJA - inicializiramo ICIA registre

dountil (beskonačna zamka)
: call  ČITAJ-ZNAK
: if  ZNAK JE "M"
: :   then  include M
: :   else
: :     if  (ZNAK JE "L")
: :     : then  include  L
: :     : else
: :     :   if (ZNAK JE "R")
: :     :   : then  include  R
: :     :   : else
: :     :   :   if (ZNAK JE "P")
: :     :   :   : then  include  P
: :     :   :   : else
: :     :   :   :   if (ZNAK JE "C")
: :     :   :   :   : then include C
: :     :   :   :   : else
: :     :   :   :   :   if (ZNAK JE "G")
: :     :   :   :   :   : then izvršimo
: :     :   :   :   :   :      mašinsku naredbu
: :     :   :   :   :   :      RTI (return from
: :     :   :   :   :   :      interrupt)
: :     :   :   :   :   : else
: :     :   :   :   :   endif
: :     :   :   :   endif
: :     :   :   endif
: :     :   endif
: :     endif
: endif
enddo
```

Segmenti L (load) i R (registers) su isti kao kod programa u tabeli 1. Ostale segmente ćemo opisati detaljnije.

```
segment  M

FORMIRAMO ADRESU LOKACIJE KOJU ŽELIMO, TAKO DA
ČITAMO 4 ZNAKA SA TASTATURE;
ZAPAMTIMO FORMIRANU ADRESU LOKACIJE ADDRH  I
ADDRL
```

Tabela 1. Primjer monitorskog programa

```
      TITLE  MICMON
*****************************************
*
*  PRIMJER MALOG MONITORSKOG PROGRAMA
*
*  10 5.1977.
*
*  NAPISAO M.KOVACEVIC
*
*****************************************
*
*  MONITOR ZAHTIJEVA 0.5 K BAJTNI EPROM NA ADRE
*  SI FF00, 0.5 K BAJTNI RAM NA ADRESI 0000 TE
*  ACIA ADRESAMA FB0C I FB0D
*
*****************************************
*
*  DEFINICIJE
*
ACIACS EQU    $FB0C          ACIA KONTROLNI REG
ACIADA EQU    $FB0D          ACIA PODATKOVNI RE
READ   EQU    $D2
STACK  EQU    $28            POKAZIVAC STEKA
       ORG    $0029
CC     RMB    1              CPU REGISTRI
B      RMB    1
A      RMB    1
XH     RMB    1
XL     RMB    1
PH     RMB    1
PL     RMB    1
SH     RMB    1
SL     RMB    1
*
CKSM   RMB    1              KONTROLNA SUMA
XHI    RMB    1              INDEKSNI REGISTAR
XLOW   RMB    1
BEGA   RMB    2              POCETNA ADRESA ZA
*                             BUSENJE
ENDA   RMB    2              KONACNA ADRESA ZA
*                             BUSENJE
TW     RMB    2
TEMP   RMB    1
MCOUNT RMB    1
BYTECT RMB    1              BROJAC BAJTOVA U R
*                             EKORDU
*
*****************************************
*  PODPROGRAM ZA CITANJE ZNAKA
*
*  PROCITANI ZNAK JE ZAPAMCEN U AKUMULATOR A
*
INCH   LDA A  ACIACS         CITAMO STATUS
       ASR A
       BCC    INCH           ZNAK NIJE SPREMAN
*                             ZA CITANJE
       LDA A  ACIADA
       AND A  #$7F
       CMP A  #$7F
       BEQ    OUTLH          EHO
*
*  PODPROGRAM ZA CITANJE HEKSADECIMALNE CIFRE
*  CIFRA JE U NIZIH 4 BITA AKUMULATORA A
*
       BSR    INCH
       CMP A  #$30
       BMI    C1             NEREGULARAN ZNAK
       CMP A  #$39
       BLE    IN1HG          0-9
       CMP A  #$41
       BMI    C1             NEREGULARAN ZNAK
       CMP A  #$46
       BGT    C1
       SUB A  #7             A-F
IN1HG  RTS
*
*  SEGMENT ZA UNOSENJE OBJEKTNOG PROGRAMA U RAM
*
LOAD   LDA A  #$D2           UKLJUCIMO CITAC TR
*                             AK
       STA A  ACIACS
       LDA A  #$11
       BSR    OUTCH
LOAD3  BSR    INCH           CITAMO ZNAK
       CMP A  #'S
       BNE    LOAD3          CEKAMO POCETAK REK
*                             ORDA
       BSR    INCH           CITAMO TIP REKORDA
       CMP A  #'9
       BEQ    LOAD21         ZADNJI REKORD
       CMP A  #'1
       BNE    LOAD3          PRESKOCIMO PRAZNA
*                             MJESTA
       CLR    CKSM
       BSR    BYTE           CITAMO BROJ BAJTOV
*                             U REKORDU
       SUB A  #2
       STA A  BYTECT
       BSR    BADDR          CITAMO ADRESU REKO
*                             RDA
LOAD11 BSR    BYTE           CITAMO JEDAN BAJT
*                             IZ REKORDA
       DEC    BYTECT
       BEQ    LOAD15         PROCITALI SVU REKO
*                             RD
       STA A  X              UPISEMO PROCITANI
*                             BAJT U RAM
       INX
       BRA    LOAD11
LOAD15 INC    CKSM
       BEQ    LOAD3          KONTROLNA SUMA JE
*                             PRAVILNA
LOAD19 LDA A  #'?
       BSR    OUTCH
LOAD21 LDA A  #$81           ISKLJUCIMO CITAC T
*                             RAKE
       STA A  ACIACS
C1     JMP    CONTROL
*
*  PODPROGRAM ZA GRADNJU CETVEROCIFARSKE HEKSAD
*  ECIMALNE ADRESE KOJA SE ZAPAMTI U X REG
*
BADDR  BSR    BYTE           VISIH 8 BITOVA ADR
*                             ESE
       STA A  XHI
       BSR    BYTE           NIZIH 8 BITOVA ADR
*                             ESE
       STA A  XLOW
       LDX    XHI
       RTS
*
*  PODPROGRAM ZA CITANJE VRIJEDNOSTI BYTA ZAPIS
*  ANE U ASCII HEKSADECIMALNOJ NOTACIJI
*
BYTE   BSR    INHEX          CITAMO HEX. CIFRU
       ASL A                 POSTAVIMO JE U VISIH 4 BITA
       ASL A
       ASL A
       ASL A
       TAB
       BSR    INHEX          DRUGA HEX. CIFRA
       AND A  #$0F
       ABA
       TAB
```

Tabela 1. Nadaljevanje

```
        ADD B   CKSM            ODRZAVAMO KONTROLN
*                               U SUMU
        STA B   CKSM
        RTS
*
*   SEGMENT ZA MONITORSKU NAREDBU M
*
CHANGE  BSR     BADDR           FORMIRAMO ADRESU
        BSR     OUTS
        BSR     OUT2HS          ISPIS SADRZAJA LOK
*                               ACIJE
        BSR     BYTE            NOVAVRIJEDNOST
        STA A   X
        BRA     CONTRL
*
*   PODPROGRAM ZA ISPIS HEKSADECIMALNE CIFRE
*   SADRZANE U VISIH 4 BITA ACUMULATORA A
*
OUTHL   LSRA
        LSRA
        LSRA
        LSRA
*
*   PODPROGRAM ZA ISPIS HEKSADECIMALNE CIFRE
*   SADRZANE U NIZIH 4 BITA ACUMULATORA A
*
OUTHR   AND A   #$F
        ADD A   #$30
        CMP A   #$39
        BLS     OUTCH           CIFRA 0-9
        ADD A   #$7             CIFRA A-F
*
*   PODPROGRAM ZA ISPIS ASCII ZNAKA KOJI JE U
*   AKUMULATORU A
*
OUTCH   PSHB
OUTC1   LDA B   ACIACS          CITAMO STATUS
        ASR B
        ASR B
        BCC     OUTC1           IZLAZ ZAUZET
        STA A   ACIADA
        PUL B
        RTS
*
*   PODPROGRAM ZA ISPIS SADRZINE AKUMULATORA A
*   U HEXADECIMALNOJ NOTACIJI U ASCII KODU
*
OUT2H   LDA A   0,X
        BSR     OUTHL           ISPIS VISIH 4 BITA
        LDA A   0,X
        BSR     OUTHR           ISPIS NIZIH 4 BITA
        INX
        RTS
*
*   PODPROGRAM ZA ISPIS SADRZINE AKUMULATORA A U
*   HEKSADECIMALNOJ NOTACIJI U ASCII KODU I
*   PRAZNOG MJESTA
*
OUT2HS  BSR     OUT2H
        LDA A   #$20
        BRA     OUTCH
*
*   SEGMENT ZA ISPIS SADRZAJA SVIH 6 REGISTARA
*   CPU JEDINICE ZAPAMCENIH NA VRHU STEKA
*
PRINT   TSX
        STX     SH
        LDA B   #9
PRINT2  BSR     OUT2HS
        DEC B
        BNE     PRINT2
*
*   INICIALIZACIJA ACIA
*
        LDA A   #$03
        STA A   ACIACS
        DEC A
        STA A   ACIACS
CONTRL  LDS     STACK           INICIALIZACIJA POK
*                               AZIVACA STEKA
        LDA A   #$D
        BSR     OUTCH           ISPIS CR
        LDA A   #$A
        BSR     OUTCH           ISPIS LF
        JSR     INCH            CITAMO KOMANDU
        TAB
        BSR     OUTS
        CMP B   #'L
        BNE     *+5
        JMP     LOAD            L
        CMP B   #'M
        BEQ     CHANGE          M
        CMP B   #'R
        BEQ     PRINT           R
        CMP B   #'G
        BNE     CONTRL
        RTI
        END
```

```
dountil (PROČITANI ZNAK JE TAČKA (.))
:  ISPIŠEMO ADRESU SADRŽANU U LOKACIJAMA ADDRH
                  i ADDRL;
:  TE SADRŽINU TE LOKACIJE ;
:  ČITAMO  ZNAK;
:  if  (PROČITANI ZNAK JE ZNAK ZA PRAZNO MJESTO)
:  :  then
:  :     INKREMENTIRAMO ADRESU U ADDH i ADDL;
:  :  else
:  :    if (PROČITANI ZNAK JE '\' )
:  :    : then
:  :    :    DEKREMENTIRAMO ADRESU U ADDH I
:  :    :                                  ADDL
:  :    : else
:  :    :    FORMIRAMO NOVU VRIJEDNOST ZA
:  :    :     LOKACIJU SA ISPISANOM ADRESOM
:  :    :     I SADRŽAJEM;
:  :    :    UPIŠEMO NOVOFORMIRANU VRIJEDNOST
:  :    :        U ODGOVARAJUĆU LOKACIJU I
:  :    :          ISPIŠEMO JE;
:  :    endif;
:  endif ;
enddo
```

U segmentu P bušimo objektni program na papirnu traku u formatu koji je već opisan pri opisu naredbe L.

```
segment  P
  FORMIRAMO POČETNU I KONAČNU ADRESU OBJEKTNOG
  PROGRAMA U  MEMORIJI TE IH ZAPAMTIMO U LO-
  KACIJE STARTH, STARTL, ENDH I ENDL;
  IZBUŠIMO NULTI REKORD (S0000000);
dountil (ADRESA U LOKACIJAMA STARTH I STARTL
:        JE JEDNAKA KONAČNOJ ADRESI)
:  if  (ENDL-STARTL ≥ 13)
:  : then
:  :    IZBUŠIMO POČETNE ZNAKE REKORDA-S113;
:  :      (13 je heksadecimalni broj bajtova
:  :        u rekordu +3)
:  :    IZBUŠIMO HEKSADECIMALNU  ADRESU PRVOG
:  :      BAJTA U REKORDU SADRŽANU U STARTH I
:  :                                    STARTL;
:  :    IZBUŠIMO SADRŽINE 16 BAJTOVA U HEKSA-
:  :      DECIMALNOJ NOTACIJI POČEVŠI OD ADRE-
:  :              SE STARTH I STARTL;
:  :    INKREMENTIRAMO SADRŽAJ LOKACIJA STARTH
:  :      I STARTL ZA 10 HEKSADECIMALNO;
:  : else
:  :    IZBUŠIMO POČETNE ZNAKE REKORDA-S1;
:  :    IZRAČUNAMO BROJ PREOSTALIH BAJTOVA
:  :      (ENDL-STARTL+3);
:  :    IZBUŠIMO BROJ PREOSTALIH BAJTOVA
:  :    IZBUŠIMO HEKSADECIMALNU ADRESU SADRŽA-
:  :      NU U LOKACIJAMA STARTH I STARTL;
:  :    IZBUŠIMO SADRŽINE PREOSTALIH BAJTOVA;
:  endif
enddo
IZBUŠIMO KONAČNI ZAPIS-S9;
endsegment
```

Ostaje nam još segment za kopiranje memorijskog bloka (START, END) na određenu adresu (ADDR).

```
segment  C
FORMIRAMO POČETNU I KONAČNU ADRESU MEMORIJSKOG
  BLOKA TE ADRESU LOKACIJE NA KOJU SE KOPIRA,
(STARTH, STARTL, ENDH, ADDRH, ADDRL)
if  (ADDR-START < 0)
:   then
:      dountil (START SADRŽI KONAČNU ADRESU)
:      :  KOPIRAMO SADRŽAJ LOKACIJE SA ADRESOM
:      :     U STARTH I STARTL U LOKACIJU SA
:      :     ADRESOM ADDRH I ADDRL;
:      :  INKREMENTIRAMO ADRESE START I ADDR;
:      enddo ;
:   else
:       (END-START)+ADDR → ADDR
:       dountil   (END SADRŽI POČETNU ADRESU)
:      :  KOPIRAMO SADRŽAJ LOKACIJE SA ADRESOM
:      :     U ENDH I ENDL U LOKACIJU SA ADRESOM
:      :     U ADDRH I ADDRL;
:      :  DEKREMENTIRAMO ADRESE END I ADDR;
:      enddo
endif
endsegment
```

4. Mogućnosti za optimizaciju monitorskih programa

Procesor 6800 ima nekoliko registara koji nam omogućavaju vrlo uspješno programiranje. Posebne ugodnosti pruža mogućnost indeksiranog adresiranja te hardverski stek. Pored indeksiranog adresiranja vrlo je korisno direktno adresiranje pri kome upotrebimo samo jedan bajt za adresiranje lokacije na intervalu 0-255.

Organizacija programa mora biti takva da iskoristi sve mogućnosti za optimizaciju koje pruža procesor 6800. Neki od načina kako da racionaliziramo korištenje memorijskog prostora te poboljšamo brzinu izvodjenja programa su sljedeći:

- memorijske lokacije 0-255 upotrebljavamo za variable koje se često pozivaju te pri pozivu podprograma i servisu prekida. Pri tome koristimo direktno adresiranje koje zahtijeva manje prostora za program te manje vremena za izvršavanje.
- po mogućnosti, podprograme postavimo tako da su blizu programima koji ih pozivaju. To omogućava upotrebu branch-to-subroutine naredbi namjesto jump-to-subroutine što također racionalizira upotrebu prostora i vremena.
- upotrebljavamo indeksni registar svugdje gdje je to moguće što reducira potrebu za extended načinom adresiranja
- u svakom programu postavimo vrijednost indeksnog registra tako da pokazuje na odgovarajući blok lokalnih podataka.
- Zanimljive su mogućnosti upotrebe SWI (software interupt) naredbe.

Ta naredba prouzrokuje pamćenje svih registara centralne procesne jedinice u stek i skok na podprogram kojega adresa je u unaprijed određenoj lokaciji. Ta rutina vrši servis koji je u odgovarajućem problemskom prostoru često potreban. Tako sa jednobajtnom naredbom pozivamo podprogram. Koristeći se tim osobinama

SWI naredbe, možemo razviti način pozivanja podprograma zapamćenih u ROM ili PROM memoriji bez upotrebe branc-to-subroutine ili jump-to-subroutine naredbi. Najprije formiramo tabelu koja sadrži adrese svih podprograma u ROM memoriji. Potreban nam je još i kontrolni program koji se izvrši svakom SWI naredbom i koji na osnovu indeksa podprograma, koji se poziva, zapisanog u bajt neposredno za SWI naredbom, prenese kontrolu nad sistemom na odgovarajući podprogram. Takav sistem pozivanja podprograma je vrlo koristan u obimnijim programima gdje je vrlo teško izbjeći jump-to-subroutine pozive. Takvi programi su naprimjer obimniji monitorski programi.

5. Zaključak

Poznavanje monitorskih programa je vrlo korisno za korisnika mikro računarskih sistema.

Monitori su dovoljno kompleksni da zahtijevaju široko i detaljno znanje o organizaciji sistema, o perifernim napravama te o programskim konceptima. Onda kada upoznamo monitor jednog mikro računarskog sistema moći ćemo slijediti svim akcijama koje vršimo na sistemu što nas posebno stimulira pri savladivanju nekog problema.

Literatura

1) M6800 Microprocessor Programming Manual Motorola Inc., 1975

2) M6800 Microprocessor Applications Manual Motorola Inc., 1975

3) T. Dollhof, μP software: How to Optimize Timing Memory Usage, Digital Design, november 1976

Informatica št. 2 letnik 1977

the preference scoring method for decision making- survey, classification and annotate bibliography

j. j. dujmović

UDK 681.3.001.1

University of Belgrade,
Dept. of Eelectrical Engineering
11000 Beograd

THE DEVELOPMENT OF SCORING METHODS FOR DECISION MAKING IN THE COURSE OF THE LAST 25 YEARS IS SURVEYED. VARIOUS VARIANTS OF THIS METHODOLOGY ARE PRESENTED, CLASSIFIED, AND EVALUATED. BOTH THEORETICAL CONTRIBUTIONS AND MAJOR APPLICATIONS IN THE FIELD OF COMPUTER SYSTEM EVALUATION AND SELECTION ARE INCLUDED. THE SELECTED ANNOTATED BIBLIOGRAPHY CONTAINING PAPERS PROPOSING THE SCORING METHODOLOGY AND APPLICATIONS, AS WELL AS THE PAPERS CRITICIZING SCORING, IS PRESENTED AT THE END OF THE PAPER.

ODLUČIVANJE PRIMENOM METODE BODOVANJA PREFERENCI - PREGLED, KLASIFIKACIJA I KOMENTARISANA BIBLIOGRAFIJA. PREZENTIRA SE PREGLED RAZVOJA METODE BODOVANJA PREFERENCI ZA POSLEDNJIH 25 GODINA. RAZNE VARIJANTE OVE METODOLOGIJE SU KLASIFIKOVANE I VREDNOVANE, UKLJUČUJUĆI KAKO TEORETSKE DOPRINOSE, TAKO I VAŽNIJE PRIMENE U OBLASTI VREDNOVANJA RAČUNARSKIH SISTEMA. NA KRAJU RADA PRILOŽENA JE ODABRANA KOMENTARISANA BIBLIOGRAFIJA KOJA OBUHVATA RADOVE KOJI PREDLAŽU METODOLOGIJU BODOVANJA PREFERENCI, A TAKODJE I RADOVE KOJI O OVOJ METODOLOGIJI IZRAŽAVAJU KRITIČKE SUDOVE.

INTRODUCTION

DECISION ANALYSIS CONSISTS OF A NUMBER OF METHODS USED FOR DECISION MAKING. THE PREFERENCE SCORING METHOD IS ONE AMONG THESE METHODS. WE USE THE TERM 'PREFERENCE SCORING METHOD' ('PSM' HENCEFORTH) FOR A METHODOLOGY COMMONLY CALLED 'SCORING SYSTEM', 'WEIGHTED FACTOR METHOD', 'WEIGHTED RANKING', ETC., IN AN ATTEMPT TO RELATE THE TITLE OF THE METHOD WITH THE TYPE OF DECISION SITUATION IN WHICH THE METHOD CAN BE APPLIED. THIS DECISION SITUATION PERTAINS TO THE DECISION CATEGORY KNOWN AS 'DECISION UNDER CERTAINTY', AND IS SPECIFIED IN MORE DETAIL BY THE FOLLOWING CHARACTERISTICS

1. THERE EXISTS A SYSTEM TO BE EVALUATED, AND AN INDIVIDUAL (OR A TEAM), CALLED 'EVALUATOR', BEING IN CHARGE OF JUDGING, FROM A GIVEN STANDPOINT, THE WORTH OF THE SYSTEM.

2. MOST FREQUENTLY, THE 'GIVEN STANDPOINT' IS DIRECTLY DERIVED FROM THE REQUEST THAT THE SYSTEM ATTAINS SOME PRESCRIBED GOALS.

3. THE EVALUATION PROCESS CONSISTS OF THE STEP-BY-STEP DETERMINATION OF A QUANTITATIVE GLOBAL DEGREE OF PREFERENCE, OFTEN CALLED 'GLOBAL EFFECTIVENESS', ASSOCIATED WITH THE EVALUATED SYSTEM.

4. AS A CONSEQUENCE OF THE GIVEN EVALUATION STANDPOINT, ASSOCIATED WITH THE SYSTEM IS A SET OF VARIABLES, CALLED 'PERFORMANCE VARIABLES' OR 'COMPONENTS FOR EVALUATION', WHOSE VALUES INFLUENCE THE RESULTING EVALUATOR'S DEGREE OF PREFERENCE FOR THE SYSTEM.

5. THE EVALUATOR MUST BE ABLE TO EXPRESS ITS EVALUATION STANDPOINT IN THE FORM OF THE SET OF REQUIREMENTS IMPOSED TO THE VALUES OF PERFORMANCE VARIABLES. THESE REQUIREMENTS ARE DEFINED AS MAPPINGS OF THE VALUES OF PERFORMANCE VARIABLES INTO THE VALUES OF SO CALLED 'ELEMENTARY PREFERENCES' (OR 'ELEMENTARY EFFECTIVENESSES') - THESE MAPPINGS ARE CALLED 'ELEMENTARY CRITERIA'. AFTER THE DEFINITION OF ELEMENTARY CRITERIA, THE EVALUATOR DEFINES A PROCEDURE FOR AGGREGATING THE ELEMENTARY PREFERENCES INTO THE UNIQUE GLOBAL DEGREE OF PREFERENCE FOR THE WHOLE SYSTEM. THE MAPPING OF THE VALUES OF PERFORMANCE VARIABLES INTO THE GLOBAL DEGREE OF PREFERENCE IS CALLED THE 'GLOBAL CRITERION FOR SYSTEM EVALUATION.'

6. THE DECISION PROCESS OF SELECTING THE BEST AMONG COMPETITIVE SYSTEMS (ALTERNATIVES, OR COURSES OF ACTION) REDUCES TO THE SYNTHESIS OF A GLOBAL CRITERION FOR EVALUATION, THE CALCULATION OF THE GLOBAL DEGREE OF PREFERENCE OF EACH SYSTEM, AND FINALLY TO THE SELECTION OF THE MOST PREFERRED SYSTEM.

7. THE SYSTEM IS HERE CONSIDERED IN A GENERAL SENSE, AS A SET OF INTERACTING OBJECTS, SO THAT THE PSM CAN BE APPLIED IN A VERY LARGE NUMBER OF COMPLEX DECISION SITUATIONS.

IF A DECISION PROBLEM IS DEFINED AS ABOVE, ONE CAN GUARANTEE THAT THE PSM PROVIDES THE MOST ADEQUATE WAY FOR ITS SOLUTION. OBVIOUSLY, IF SOME OTHER DECISION PROBLEM IS UNDER CONSIDERATION, THE PSM NEED NOT TO BE APPLIED (AND THIS IS NOT A DRAWBACK OF THE PSM). UNFORTUNATELY, THIS SIMPLE FACT IS OFTEN NEGLECTED BY SOME AUTHORS CRITICIZING THE PSM. THE EXAMPLES OF SUCH A CRITICISM WILL NOT BE DISCUSSED IN THIS PAPER.

THERE ARE SEVERAL THEORIES TO WHICH THE PSM IS RELATED. THE PSM APPERTAINS TO MULTIPLE CRITERIA DECISION METHODS, AND IS CLOSELY RELATED TO VARIOUS ECONOMIC DECISION METHODS, ESPECIALLY TO THE UTILITY THEORY. BECAUSE OF THE USE OF PSM TO RECOGNIZE SYSTEMS SATISFYING

THE EVALUATOR'S CRITERION IN SOME PRESCRIBED EXTENT, THE PSM IS ALSO RELATED TO THE PATTERN RECOGNITION METHODS. FINALLY, SINCE PREFERENCES ARE CONCERNED, THE PSM IS ALSO RELATED TO THE LOGIC OF PREFERENCE.

THE APPLICATIONS OF THE PSM ARE INDEED NUMEROUS. ALL EVALUATION AND/OR SELECTION PROBLEMS HAVING PREFERENCE AS ITS NATURAL RESULT, LEAD TO THE USE OF THE PSM. EVALUATION AND SELECTION OF COMPUTERS AND OTHER TECHNICAL DEVICES, EVALUATION AND SELECTION OF RESEARCH AND DEVELOPMENT PROJECTS, AND CONSUMER SELECTION OF GOODS AND SERVICES, ARE FEW OF SUCH EXAMPLES. EVEN IN THE SITUATIONS INVOLVING RISK AND UNCERTAINTY, THE PSM CAN BE SUCCESSFULLY USED AS A FIRST STEP TOWARD SOLUTION.

THE HISTORY OF THE PSM USE IS VERY LONG AND SOME AUTHORS QUOTE EXAMPLES OF ITS USE WHICH ARE MORE THAN HUNDRED YEARS OLD. SINCE FIFTIES UNTILL NOW THE MOST PROMOTING FIELDS OF THE PSM THEORY AND APPLICATIONS WERE THE EVALUATION AND SELECTION OF COMPUTERS AND THE EVALUATION AND SELECTION OF RESEARCH AND DEVELOPMENT PROJECTS.

THE SCOPE OF THIS PAPER WILL BE CONCENTRATED ON THE DEVELOPMENT OF THE PSM IN THE COURSE OF THE LAST 25 YEARS, AND ON THE PSM APPLICATIONS IN THE FIELD OF COMPUTER SYSTEM EVALUATION AND SELECTION. HOWEVER, SOME IMPORTANT WORKS IN THE FIELDS RELATED TO PSM, AS WELL AS SOME IMPORTANT APPLICATIONS DIFFERENT FROM COMPUTER SELECTION, WILL BE ALSO PRESENTED.

SURVEY OF THE BASIC PSM CONCEPTS

LET BE GIVEN THE VECTOR

$$x = (x_1,\dots,x_n) ,$$

$$x_{i\,min} \leq x_i \leq x_{i\,max} , \quad i=1,\dots,n ,$$

CONTAINING ALL PERFORMANCE VARIABLES OF THE EVALUATED SYSTEM. DETERMINATION OF THE CORRESPONDING VECTOR OF ELEMENTARY PREFERENCES (EFFECTIVENESSES):

$$E = (E_1,\dots,E_n) , \quad 0 \leq E_i \leq 1, \quad i=1,\dots,n ,$$

CAN BE ORGANIZED EITHER BY SOME INFORMAL, INTUITIVE SCORING, OR (WHAT IS BETTER) BY THE USE OF A SET OF ELEMENTARY CRITERIA WHICH CAN BE DEFINED IN ANALYTICAL OR TABULAR FORM:

$$E_i = g_i(x_i) , \quad i=1,\dots,n .$$

THE OLDEST WAY FOR AGGREGATING ELEMENTARY PREFERENCES IN ORDER TO CALCULATE THE GLOBAL PREFERENCE OF A SYSTEM, IS BY THE USE OF THE WEIGHTED ARITHMETIC MEAN:

$$E_o = \sum_{i=1}^{n} W_i E_i , \quad 0 \leq E_o \leq 1 .$$

THE VECTOR OF WEIGHTS

$$W = (W_1,\dots,W_n) , \quad \sum_{i=1}^{n} W_i = 1 ,$$

$$0 < W_i < 1, \quad i=1,\dots,n ,$$

CAN BE DETERMINED EITHER BY A DIRECT ONE-LEVEL PROCEDURE, OR IN A FAR BETTER WAY, BY THE USE OF MULTI-LEVEL PROCEDURE ORGANIZING THE WEIGHTS IN A HIERARCHICAL TREE-LIKE STRUCTURE.

THE NEXT POSSIBILITY FOR AGGREGATION OF ELEMENTARY PREFERENCES IS BASED ON THE USE OF THE WEIGHTED GEOMETRIC MEAN:

$$E_o = \prod_{i=1}^{n} E_i^{W_i} , \quad 0 \leq E_o \leq 1 .$$

THE ESSENTIAL AND SERIOUS DRAWBACK OF BOTH THE ARITMETIC MEAN APPROACH AND THE GEOMETRIC MEAN APPROACH IS THE COMPLETE LACK OF POSSIBILITY TO EXPRESS THE NECESSARY LOGIC RELATIONSHIPS AMONG ELEMENTARY PREFERENCES AND THEIR GROUPS. IN FACT, ARITHMETIC AND GEOMETRIC MEAN PERMIT ONLY THE 'FIXED LOGIC POLARIZATION' OF A COMPLEX CRITERION. THIS CAN CAUSE A NUMBER OF ABSURD CONCLUSIONS. FOR EXAMPLE, USING THE ARITHMETIC MEAN, IF THE MOST IMPORTANT PREFERENCE, SAY THE PREFERENCE OF THE THRUPUT OF A COMPUTER, IS ZERO. THIS WILL NOT AUTOMATICALLY CAUSE THE ZERO GLOBAL PREFERENCE (WHAT IS ABSOLUTELY NECESSARY), AND EVEN IT WILL BE POSSIBLE TO COMPENSATE THE UNACCEPTABLE THRUPUT BY A SIMPLE INCREASING OF, SAY, THE NUMBER OF TAPES OR THE TRAINING OF PERSONEL. IN A SIMILAR WAY, USING THE GEOMETRIC MEAN, IF SOME UNIMPORTANT PREFERENCE (E.G. THE PREFERENCE OF THE INTEGER DEVIDE TIME) IS ZERO, THIS WILL AUTOMATICALLY CAUSE THE ZERO PREFERENCE OF THE WHOLE SYSTEM, WITHOUT ANY POSSIBILITY OF COMPENSATION.

THE DRAWBACKS OF WEIGHTED ARITHMETIC MEAN AND WEIGHTED GEOMETRIC MEAN CAN BE OVERCOMED BY INTRODUCING THE MULTI-LEVEL LOGIC AGGREGATION OF ELEMENTARY PREFERENCES:

$$E_o = L(E_1,\dots,E_n) = L(g_1(x_1),\dots,g_n(x_n)) =: g(x_1,\dots,x_n) .$$

FOR THIS PURPOSE, ALL PREFERENCES ARE INTERPRETED AS DEGREES OF TRUTH IN THE CORRESPONDING STATEMENTS WHICH HAVE THE FORM OF ASSERTION THAT 'THE SYSTEM (OR SOME ITS PART) COMPLETELY FULFILS ALL GIVEN REQUIREMENTS.' THEREFORE, L REPRESENTS THE LOGIC FUNCTION WHICH IS REALIZED BY AN ADEQUATE SUPERPOSITION OF THE SO-CALLED 'EXTENDED CONTINUOUS LOGIC FUNCTIONS.' THE ELEMENTARY FUNCTIONS IN THE EXTENDED CONTINUOUS LOGIC ARE QUASI-CONJUNCTION AND QUASI-DISJUNCTION, WHICH ARE REALIZED AS THE WEIGHTED POWER MEANS:

$$e_o = \left(\sum_{i=1}^{k} W_i e_i^R \right)^{1/R} , \quad -\infty \leq R \leq +\infty ,$$

WHOSE REQUESTED LOGIC PROPERTIES CAN BE ADJUSTED BY THE APPROPRIATE SELECTION OF THE PARAMETER R ($e_1,\dots,e_k$ DENOTE SOME INPUT PREFERENCES, AND e_o IS THE CORRESPONDING OUTPUT PREFERENCE IN SOME AGGREGATION POINT WITHIN THE MULTI-LEVEL PREFERENCE AGGREGATION STRUCTURE). SINCE THE ARITHMETIC AND GEOMETRIC MEAN ARE THE SPECIAL CASES OF THE WEIGHTED POWER MEANS, IT FOLLOWS THAT ALL PREVIOUSLY PRESENTED COMPLEX CRITERIA ARE THE SPECIAL CASES OF THE CRITERION BASED ON THE MULTI-LEVEL LOGIC AGGREGATION OF ELEMENTARY PREFERENCES.

AFTER THE REALIZATION OF THE GLOBAL CRITERION FOR SYSTEM EVALUATION, WHICH ENABLES THE CALCULATION OF THE GLOBAL DEGREE OF PREFERENCE OF A SYSTEM DIRECTLY FROM THE VALUES OF ITS PERFORMANCE VARIABLES, I.E.

$$E_o = g(x_1,\dots,x_n) ,$$

IT IS POSSIBLE TO PERFORM, IN ADDITION TO THE IMMEDIATE SYSTEM EVALUATION AND COMPARISON, THE FOLLOWING SEVERAL VERY USEFUL ANALYSES.

THE SENSITIVITY ANALYSIS INVESTIGATES HOW VARIOUS INPUTS AFFECT VARIOUS OUTPUTS IN COMPLEX CRITERION. A NUMBER OF USEFUL SENSITIVITY INDICATORS CAN BE DEFINED.

THE TRADEOFF ANALYSIS INVESTIGATES THE PROBLEM OF COMPENSATING THE LACK OF SOME SYSTEM

PERFORMANCE BY INCREASING SOME OTHER SYSTEM PERFORMANCE. VARIOUS TRADEOFF INDICATORS CAN BE INTRODUCED.

SYSTEM OPTIMIZATION INTRODUCES THE ALGORITHMS FOR OPTIMIZATION OF THE COMPLEX SYSTEM CONFIGURATION, GIVING ANSWERS TO THE FOLLOWING QUESTIONS - (1) WHAT IS THE OPTIMAL SYSTEM CONFIGURATION, YIELDING MAXIMAL GLOBAL PREFERENCE FOR CONSTRAINED COST OF THE SYSTEM, AND (2) WHAT IS THE MINIMAL COST OF THE SYSTEM, NECESSARY TO ATTAIN A GIVEN DEGREE OF THE EVALUATOR'S PREFERENCE.

THE RELIABILITY ANALYSIS QUANTITATIVELY INVESTIGATES THE RELIABILITY OF RESULTS OBTAINED FROM EVALUATION STUDIES, TAKING INTO ACCOUNT THAT ALL COMPLEX CRITERIA ARE SUBJECTIVELY SYNTHESIZED BY CORRESPONDING EXPERTS AND THEREFORE CONTAIN INHERENT ESTIMATION ERRORS. THE RANGES OF POSSIBLE ERRORS ARE INVESTIGATED AND SOME QUANTITATIVE RELIABILITY INDICATORS ARE INTRODUCED.

COST-PREFERENCE ANALYSIS INTRODUCES THE TECHNIQUES FOR COMPARING COMPETITIVE SYSTEMS ON THE BASIS OF RELATIONS BETWEEN THE COSTS OF THE SYSTEMS AND THEIR GLOBAL DEGREES OF PREFERENCE.

AN ANNOTATED BIBLIOGRAPHY OF THE PSM WILL BE PRESENTED IN THE SEQUEL. AS FAR AS WE KNOW, THIS IS THE FIRST ATTEMPT TO COMPILE AND ANNOTATE A BIBLIOGRAPHY DEVOTED ESPECIALLY TO THE PSM. THEREFORE, WE CAN NOT GUARANTEE THAT THE BIBLIOGRAPHY IS COMPLETE AND WITHOUT SOME ERRORS. MOREOVER, THE NOTES VARY IN SIZE AND CONTENT, AND SOME ENTRIES ARE LEFT WITHOUT COMMENT. THE SIZE OF NOTE BY NO MEANS IS PROPORTIONAL TO THE IMPORTANCE OF THE ITEM UNDER CONSIDERATION (IN FACT, THIS SIZE IS PROPORTIONAL TO THE AUTHOR'S AVAILABLE TIME FOR PROCESSING THE ITEM).

THE PRESENTED BIBLIOGRAPHY IS CLASSIFIED. EACH ENTRY CONTAINS THE CLASSIFICATION CODE WHICH IS COMPOSED OF ONE OR MORE CLASSIFICATION CHARACTERS. EACH CLASSIFICATION CHARACTER DENOTES THAT THE GIVEN BIBLIOGRAPHIC ENTRY APPERTAINS TO THE CORRESPONDING CLASS OF BIBLIOGRAPHIC ENTRIES. ALL CLASSES ARE DEFINED IN THE FOLLOWING TABLE, AND AT THE END OF THE PAPER, FOR EACH CLASS, THE REFERENT NUMBERS OF ALL BELONGING BIBLIOGRAPHIC ENTRIES ARE PRESENTED.

CLASSIFICATION CODES FOR PAPERS PRESENTING THE PREFERENCE SCORING METHOD

---

1 = PAPER PROPOSING THE ONE-LEVEL LIST OF THE PERFORMANCE VARIABLES OF SOME GIVEN SYSTEM
2 = PAPER PROPOSING THE HIERARCHICALLY ORGANIZED LIST OF PERFORMANCE VARIABLES FOR THE EVALUATION OF SOME GIVEN SYSTEM
3 = INFORMAL (INTUITIVE) SCORING OF THE VALUES OF PERFORMANCE VARIABLES
4 = ELEMENTARY CRITERIA DEFINED IN THE ANALYTICAL OR TABULAR FORM
5 = THE ONE-LEVEL POINT-ADDITIVE CRITERIA AGGREGATION STRUCTURE (WITH OR WITHOUT THE EXPLICIT USE OF WEIGHTS)
6 = THE MULTI-LEVEL ADDITIVE CRITERIA AGGREGATION STRUCTURE (BASED ON THE WEIGHTED ARITHMETIC MEAN)
7 = THE ONE- OR MULTI-LEVEL MULTIPLICATIVE (OR GEOMETRIC MEAN) CRITERIA AGGREGATION STRUCTURE
8 = THE MULTI-LEVEL CRITERIA AGGREGATION STRUCTURE BASED ON THE USE OF THE EXTENDED CONTINUOUS LOGIC FUNCTIONS
9 = SPECIAL AGGREGATION STRUCTURES (E.G. FIGURES OF MERIT, SPECIAL UTILITY FUNCTIONS, ETC.)
A = SURVEY PAPER OR BOOK INCLUDING THE PRESENTATION OF THE PSM
B = PAPER CONTAINING BIBLIOGRAPHICAL RESEARCH RELATED TO THE PSM
C = PAPER PROPOSING THE COMPLETE COMPUTER EVALUATION CRITERION, AND/OR THE CASE STUDY OF COMPUTER EVALUATION AND SELECTION BASED ON THE PSM
D = THE PSM USED IN THE EVALUATION AND SELECTION OF THE COMPETITIVE RESEARCH AND DEVELOPMENT PROJECT PROPOSALS
E = PAPER CONTAINING AN ILLUSTRATIVE EXAMPLE OF AN ARBITRARY SYSTEM EVALUATION AND/OR SELECTION PROCEDURE BASED ON THE USE OF THE PSM
F = THE FUZZY SET APPROACH TO PREFERENCE SCORING
G = THE ANALYTICAL MODELS FOR DERIVING PARAMETERS OF THE CRITERION FOR SYSTEM EVALUATION STARTING DIRECTLY FROM GOALS TO BE ATTAINED BY THE SYSTEM
M = THE MATHEMATICAL PROBLEMS RELATED TO THE PSM
O = SYSTEM OPTIMIZATION BASED ON THE PSM CRITERION
P = PAPER PRESENTING OR QUOTING PROGRAM PRODUCTS APPLIED WHEN EVALUATING AND SELECTING SYSTEMS USING THE PSM
Q = COST - PREFERENCE ANALYSIS, WHERE THE PREFERENCE IS OBTAINED BY THE PSM CRITERION
R = RELIABILITY ANALYSIS OF THE PSM CRITERIA
S = SENSITIVITY ANALYSIS OF THE PSM CRITERIA
T = TRADEOFF ANALYSIS OF THE PSM CRITERIA
U = FORMAL MODELS FOR THE MULTIPLE CRITERIA DECISION MAKING DEVELOPED WITHIN THE UTILITY THEORY AND INTERESTING FROM THE PSM STANDPOINT
W = THE PROCEDURES FOR DETERMINING WEIGHTS BY THE GROUP OF EXPERTS, AND THE TECHNIQUES USED BY THE GROUP MEMBERS IN ORDER TO NEGOTIATE THE GROUP CONSENSUS
X = PAPERS INDIRECTLY RELATED TO THE PSM
Y = PRESENTATION OF EXPERIENCES DERIVED FROM THE PRACTICE OF SYSTEMS AND COMPUTERS EVALUATION AND SELECTION USING THE PSM
Z = PAPER CRITICIZING SOME ASPECTS OF THE PSM

AN ANNOTATED BIBLIOGRAPHY OF THE PREFERENCE SCORING METHOD

---

1. ********** BEFORE 1962 **********
VON NEUMANN, J. AND O. MORGENSTERN, 'THEORY OF GAMES AND ECONOMIC BEHAVIOR' REVISED EDITION (PRINCETON, N.J. / PRINCETON UNIVERSITY PRESS, 1947). (CODE=UX)

2. THRALL, R.M., COOMBS, C.H. AND DAVIS, R.L., 'DECISION PROCESSES', WILEY AND CHAPMAN AND HALL, NEW YORK 1954. (CODE=UX)

3. CHURCHMAN, C.W., ACKOFF, R.L., 'AN APPROXIMATE MEASURE OF VALUE', JOURNAL OF THE OPERATIONS RESEARCH SOCIETY OF AMERICA, MAY 1954., 172-187.
--- THE PAPER PRESENTS THE ADDITIVE PSM MODEL WHERE THE WEIGHTS ARE CALCULATED DIRECTLY FROM THE OUTCOMES TO BE ATTAINED. (CODE=35G)

4. CHURCHMAN, C.W., ACKOFF, R.L. AND ARNOFF E.L., 'INTRODUCTION TO OPERATIONS RESEARCH', CHAPTER 6, WILEY, NEW YORK, 1957. (CODE=G)

5. AUMANN, P.J. AND I.B. KRUSKAL, 'ASSIGNING QUANTITATIVE VALUES TO QUALITATIVE FACTORS IN THE NAVAL ELECTRONICS PROBLEM,' NAVAL RESEARCH LOGISTICS QUARTERLY, VOL.6, NO.1, PP. 1-16, MARCH, 1959 (CODE=X)

6. YNTEMA, D.B., TORGERSON, W.S., 'MAN-COMPUTER COOPERATION IN DECISIONS REQUIRING COMMON SENSE.' IRE TRANS. ON HFE, MARCH 1961. (CODE=X)

7. RAIFFA, H., SCHLAIFER, R., 'APPLIED STATISTICAL DECISION THEORY.' THE M.I.T. PRESS, 1961. (CODE=MUX)

8. ********** 1962 **********
STEERE, R.E. JR., 'HOW TO SELECT THE RIGHT DATA PROCESSING EQUIPMENT.' ADMINISTRATIVE MANAGEMENT, AUGUST 1962, PP. 26-28.
--- THE 17 PERFORMANCE VARIABLES OF AN EARLY DATA PROCESSING EQUIPMENT ARE PROPOSED. THE PRESENTED ADDITIVE PSM MODEL WAS SUCCESSFULLY APPLIED IN PRACTICE. (CODE=135CY)

9. CASTILLO-FERNANDEZ, J.A., RIVERA-SANTANA, E., 'TECHNIQUE FOR EVALUATING ELECTRONIC COMPUTERS.' DATA PROCESSING, SEPTEMBER 1962. (CODE=X)

10. NOVOSEL, P., 'EVALUATION OF THE EFFECTIVENESS OF WORKERS' (IN SERBO - CROATIAN). ZAGREB, 1962.
--- THE BOOK REVIEWS VARIOUS TECHNIQUES FOR EVALUATION OF THE EFFECTIVENESS OF WORKERS. AMONG OTHER TECHNIQUES, THE PSM HAS BEEN EXTENSIVELY APPLIED FOR THIS PURPOSE. (CODE=5EX)

11. BLUMENTHAL, P.L. JR., 'APPLYING AN OR TECHNIQUE IN SELECTING DATA PROCESSING HARDWARE'. THE JOURNAL OF ACCOUNTANCY, AUGUST 1962., 82-83.
--- THE AUTHOR PRESENTS A PSM MODEL FOR COMPUTER EVALUATION. THE MODEL IS BASED ON THE CHURCHMAN-ACKOFF METHOD. (CODE=35G)

12. ********** 1963 **********
GREGORY, R.H., AND VAN HORN, R.L., 'AUTOMATIC DATA PROCESSING SYSTEMS.' WADSWORTH PUB. CO., BELMONT, 1963. (CODE=36)

13. FRIEDLAND, E.I., 'HOW TO SELECT THE BEST COMPUTER / A CONCEPTUAL OUTLINE.' MITRE WORKING PAPER, W-6283, JULY 1963. (CODE=X)

14. ABRAHAMS, P.W., LIPP, M.F., AND HARLOW, J., 'QUANTITATIVE METHODS OF INFORMATION PROCESSING SYSTEM EVALUATION.' ITT DATA AND INFORMATION SYSTEMS, REPORT NO. EDS-TDR-63-670, OCTOBER, 1963. (CODE=X)

15. WILLIAMS, PERROTT, WEITZMAN, MURRAY AND SHOBER, 'A METHODOLOGY FOR COMPUTER SELECTION STUDIES.' COMPUTERS AND AUTOMATION, MAY 1963., 18-22.
--- THE MULTI-LEVEL ADDITIVE PSM MODEL FOR COMPUTER EVALUATION IS PRESENTED. (CODE=236C)

16. WHITE, D.R.J., SCOTT, D.L., AND SCHULZ, R.N., 'POED - A METHOD OF EVALUATING SYSTEM PERFORMANCE.' IEEE/TEM, DECEMBER 1963, PP. 177-182.
--- THE PAPER PRESENTS ONE OF THE MOST IMPORTANT EARLY CONTRIBUTIONS TO THE PSM. ELEMENTARY CRITERIA, THE USE OF GEOMETRIC MEAN, AND SENSISIVITY AND CONFIDENCE ANALYSIS OF THE PSM RESULTS ARE PROPOSED. AN ILLUSTRATIVE EXAMPLE IS INCLUDED. (CODE=247ERS)

17. ********** 1964 **********
FISHBURN, P.C., 'DECISION AND VALUE THEORY.' WILEY, 1964.
--- THE SUBJECT OF THIS BOOK IS A PERSONALISTIC PRESCRIPTIVE THEORY OF CHOICE FOR INDIVIDUAL DECISIONS (AUTHOR'S STATEMENT). CHAPTER 10 (INDEPENDENCE AND CERTAINTY) INTRODUCES 'PERFORMANCE VARIABLES' AND, ASSUMING THAT THE PERFORMANCE VARIABLES ARE VALUEWISE INDEPENDENT, DESCRIBES AN ADDITIVE MODEL FOR THE GLOBAL PEFROMANCE ('RELATIVE VALUE') OF A SYSTEM. (CODE=M45UE)

18. ROSENTHAL, S., 'ANALYTICAL TECHNIQUE FOR AUTOMATIC DATA PROCESSING EQUIPMENT ACQUISITION.' PROCEEDINGS - SPRING JOINT COMPUTER CONFERENCE 1964, PP.359-366.
--- THE COMPLETE TECHNIQUE, BASED ON THE PSM, FOR ADP EQUIPMENT ACQUISITION IS PROPOSED. THE TECHNIQUE WAS USED BY THE U.S. AIR FORCE. THE HIERARCHICAL ADDITIVE EVALUATION MODEL WITH RELATIVE SCORING OF THE ELEMENTARY PREFERENCES IS PROPOSED. (CODE=36)

19. JOSLIN, E.O., AND MULLIN, M.J., 'COST-VALUE TECHNIQUE FOR EVALUATION OF COMPUTER SYSTEM PROPOSALS.' PROCEEDINGS-SPRING JOINT COMPUTER CONFERENCE, 1964, PP. 367-381.
--- THE AUTHOR CRITICIZES THE ADDITIVE PSM APPROACH TO COMPUTER SELECTION, AND PROPOSES THE COST-VALUE TECHNIQUE AS AN ALTERNATIVE. (CODE=QZ)

20. ********** 1965 **********
MARKEL, G.A., 'TOWARD A GENERAL METHODOLOGY FOR SYSTEMS EVALUATION.' HRB - SINGER, INC., JULY 1965. (NTIS AD 619373)
--- THE REPORT PRESENTS AN ANNOTATED BIBLIOGRAPHY OF SYSTEMS EVALUATION LITERATURE INCLUDING ALSO THE PRESENTATION OF PSM. (CODE=AB)

21. ECKENRODE, R.T., 'WEIGHTING MULTIPLE CRITERIA.' PP. 180-192, MANAGEMENT SCIENCE, VOL. 12, NO. 3, NOVEMBER, 1965 (CODE=X)

22. DEAN, B.V., NISHRY, M.J., 'SCORING AND PROFITABILITY MODELS FOR EVALUATING AND SELECTING ENGINEERING PROJECTS.' OPERATIONS RESEARCH, VOL 13, NO. 4, JULY - AUGUST 1965, PP. 550-569. (CODE=5D)

23. BROMLEY, A.C., 'CHOOSING A SET OF COMPUTERS.' DATAMATION, PP. 37-40, AUGUST 1965.
--- THE AUTHOR PRESENTS A CASE STUDY OF THE APPLICATION OF PSM BASED ON THE CHURCHMAN-ACKOFF APPROACH TO WEIGHTS DETERMINATION. THE PROCEDURE IS APPLIED FOR SELECTION OF A SET OF COMPUTERS FOR A LARGE COMPANY. (CODE=36GE)

24. ********** 1966 **********
MILLER, J.R., 'THE ASSESSMENT OF WORTH / A SYSTEMATIC PROCEDURE AND ITS EXPERIMENTAL VALIDATION'. PH. D. DISSERTATION, M.I.T., 1966.
--- VERY RIGOROUS TREATMENT OF A DECISION PROCESS BASED ON THE PSM IS PRESENTED. VARIOUS ANALYTICAL MODELS OF ELEMENTARY CRITERIA ARE INTRODUCED. AN EXPERIMENTAL VALIDATION OF THE PROPOSED PROCEDURE IS ALSO PRESENTED. (CODE=246ABCEY)

25. CANNING, R.G., 'EQUIPMENT SELECTION.' DATA PROCESSING DIGEST, NO. 6, JUNE 1966, PP. 1-9. (CODE=36)

26. ********** 1967 **********
DOWKANT, A.J., MORRIS, W.A., BUETTELL, T.D., 'A METHODOLOGY FOR COMPARISON OF GENERALIZED DATA MANAGEMENT SYSTEMS / PEGS (PARAMETRIC EVALUATION OF GENERALIZED SYSTEMS), EDS-TR-67-2, MARCH, 1967, AD 811 682.
--- THIS EXTENSIVE AND IMPORTANT STUDY PRESENTS IN DETAIL THE USE OF AN ADDITIVE VERSION OF THE PSM FOR EVALUATION AND COMPARISON OF COMPUTER SYSTEMS. DETAILED LISTS OF PERFORMANCE VARIABLES ARE ALSO PRESENTED. (CODE=246EC)

27. CETRON, M.J., MARTINO, J., ROEPCKE, L., 'THE SELECTION OF R AND D PROGRAM CONTENT - SURVEY OF QUANTITATIVE METHODS.' IEEE/TEM NO.1, MARCH, 1967.
--- AN EXTENSIVE BIBLIOGRAPHY ON THE SELECTION OF R AND D PROGRAMS IS PRESENTED. A NUMBER OF EARLY REFERENCES RELATED TO THE PSM USE IS INCLUDED. (CODE=ABD)

28. STEFFES, S.P., 'HOW THE AIR FORCE SELECTS COMPUTERS.' BUSINESS AUTOMATION , PP. 30-35, AUGUST 1967. (CODE=36)

29. ********** 1968 **********
FISHBURN, P.C., 'UTILITY THEORY.' MANAGEMENT SCIENCE NO. 5, FEBRUARY 1968, PP. 335-378. (CODE=MUX)

30. RIGGS, J.L., 'ECONOMIC DECISION MODELS', MC GRAW-HILL, NEW YORK 1968. (CODE=EQX)

31. PORTER, J.D., RUDWICK, B.H., 'APPLICATION OF COST-EFFECTIVENESS ANALYSIS TO EDP SYSTEM SELECTION'. THE MITRE CORPORATION, BEDFORD, MASS., 1968. (CODE=QB)

32. JOSLIN, E.O., 'COMPUTER SELECTION.' ADDISON-WESLEY PUBLISHING COMPANY, INC., READING, MASSACHUSETTS, U.S.A., 1968.
--- THE COST-VALUE TECHNIQUE FOR COMPUTER SELECTION IS PRESENTED. PERFORMANCE VARIABLES FOR COMPUTER SYSTEM EVALUATION ARE DEFINED. (CODE=2QZ)

33. JOHNSEN, E., 'STUDIES IN MULTIOBJECTIVE DECISION MODELS', STUDENTLITTERATUR, LUND 1968. (CODE=9ABUQMP)

34. FIFE, D.W., 'ALTERNATIVES IN EVALUATION OF COMPUTER SYSTEMS'. THE MITRE CORPORATION, BEDFORD, MASS., 1968.
--- THE REPORT DESCRIBES VARIOUS APPROACHES TO COMPUTER EVALUATION. THE ADDITIVE PSM MODEL IS PRESENTED AND CRITICIZED. (CODE=45ABZ)

35. SCHWARTZ, E.S., 'COMPUTER EVALUATION AND SELECTION.' DATA MANAGEMENT, PP. 58-62, JUNE 1968.
--- THE MULTI-LEVEL ADDITIVE PSM MODEL FOR COMPUTER EVALUATION IS PROPOSED. PERFORMANCE VARIABLES AND THEIR RATING SCALES ARE ALSO PRESENTED. (CODE=235BC)

36. KELLER, R.F., AND DENHAM, C.R., 'COMPUTER SELECTION PROCEDURES', PROCEEDINGS - 1968 ACM NATIONAL CONFERENCE, PP. 679-683.
--- THE PAPER PRESENTS A CASE STUDY OF COMPUTER SELECTION FOR AN UNIVERSITY USER. THE SELECTION IS REALIZED USING THE ADDITIVE VARIANT OF THE PSM. NUMERICAL RESULTS ARE ALSO PRESENTED. (CODE=1345EC)

37. KEENY, R.L.; 'QUASI - SEPARABLE UTILITY FUNCTIONS.' NAVAL RESEARCH LOGISTICS QUARTERLY 15, 1968.
--- THE UTILITY FUNCTIONS OF THE FORM U(X,Y) = U1(X) + U2(Y) + K*U1(X)*U2(Y), K=CONST., ARE PROPOSED AND THEIR PROPERTIES INVESTIGATED. (CODE=9MUX)

38. ********** 1969 **********
STEFFY, W.; DARBY, D.; 'PERFORMANCE EVALUATION SYSTEMS.' THE UNIVERSITY OF MICHIGAN, 1969.
--- THE BOOK IS IMPORTANT FOR THE USE OF PSM IN GENERAL INDUSTRIAL ENVIRONMENT. AN EXAMPLE OF A BUYER EVALUATION IS PRESENTED IN DETAIL. VARIOUS METHODOLOGICAL ISSUES ARE ALSO DISCUSSED. (CODE=346AEYWR)

39. SEILER, K.; 'INTRODUCTION TO SYSTEMS COST-EFFECTIVENESS.' WILEY - INTERSCIENCE, NEW YORK, 1969. (CODE=QX)

40. MITRINOVIC, D.S., VASIC, P.M.; 'MEANS' (IN SERBO-CROATIAN), BELGRADE, 1969.
--- THE BOOK PRESENTS WEIGHTED POWER MEANS AND GENERALIZED MEANS, WITH THE SPECIAL EMPHASIS ON INEQUALITIES RELATED TO MEANS. (CODE=M)

41. RAIFFA, H.; 'PREFERENCES FOR MULTI-ATTRIBUTED ALTERNATIVES', (SANTA MONICA, CALIF. / RAND, RM-5868, 1969).
--- THE EVALUATION PROCEDURE DEFELOPED BY J.R. MILLER III IS CRITICALLY REVIEWED. VARIOUS UTILITY THEORY APPROACHES ARE ALSO PRESENTED. (CODE=146MUZ)

42. SHARPE, W.F.; 'THE ECONOMICS OF COMPUTERS', A RAND CORPORATION RESEARCH STUDY, COLUMBIA UNIVERSITY PRESS, NEW YORK 1969.
--- AMONG THE OTHER TOPICS, THIS IMPORTANT BOOK PRESENTS THE METHODS FOR COMPUTER SELECTION. ADDITIVE AND MULTIPLICATIVE SCORING SYSTEMS ARE DISCUSSED IN DETAIL. THE PROBLEM OF INCLUDING COST INTO THE SCORING SYSTEM IS ALSO CONSIDERED. (CODE=357AEQST)

43. HILLEGASS, J.R.; 'SYSTEMATIC TECHNIQUES FOR COMPUTER EVALUATION AND SELECTION', MANAGEMENT SCI., JULY-AUGUST 1969., 36-40.
--- VARIOUS COMPUTER EVALUATION METHODS ARE REVIEWED. THE ADDITIVE VARIANT OF THE PSM IS CRITICIZED BECAUSE OF SUBJECTIVITY WHEN DEFINING WEIGHTS AND SCORES. (CODE=35AZ)

44. MILLER, W.G.; 'SELECTION CRITERIA FOR COMPUTER SYSTEM ADOPTION', EDUCATIONAL TECHNOLOGY, OCTOBER 1969., 71-75.
--- A LIST OF COMPUTER SYSTEM PERFORMANCE VARIABLES IS PROPOSED. (CODE=135C)

45. SCHARF, T.; 'WEIGHTED RANKING BY LEVELS,' IAG QUARTERLY JOURNAL, PP. 7-22, VOL. 2, NO. 2, 1969.
--- SECOND PRESENTATION OF THE POPULAR METHOD DEVELOPED IN 1968. A SAMPLE FROM COMPUTER EVALUATION STUDY IS INCLUDED. (CODE=236EY)

46. MOORE, J.R.; BAKER, N.R.; 'AN ANALYTICAL APPROACH TO SCORING MODEL DESIGN - APPLICATION TO RESEARCH AND DEVELOPMENT PROJECT SELECTION.' IEEE/TEM NO.3, AUGUST 1969.
--- RATIONALE FOR THE PSM USE WHEN SELECTING RESEARCH AND DEVELOPMENT PROJECTS IS PRESENTED. STATISTICAL TECHNIQUE IS INTRODUCED IN PSM MODELS. (CODE=BDR)

47. SCHWARTZ, M.H.; 'COMPUTER PROJECT SELECTION IN THE BUSINESS ENTERPRISE.' JOURNAL OF ACCOUNTANCY, NO. 4, APRIL 1969, PP. 35-43. (CODE=36)

48. DUJMOVIĆ, J.J.; PJEVIĆ, N.; 'AN ALGORITHM FOR COMPARISON AND SELECTION OF DIGITAL COMPUTERS' (IN SERBO - CROATIAN), PROCEEDINGS OF THE FIRST YUGOSLAV ADP CONGRESS, ZAGREB 1969, PAPER 34C.
--- THE PROPOSED ALGORITHM IS BASED ON THE IVANOVIC'S DISTANCE DEVELOPED IN DISCRIMINANT ANALYSIS. WEIGHTED FACTORS ARE CALCULATED AUTOMATICALLY FROM THE CORRELATION MATRIX OF COMPONENTS FOR EVALUATION. (CODE=9W)

49. ********** 1970 **********
MILLER III, J.R.; 'PROFESSIONAL DECISION-MAKING.' PRAEGER PUBLISHERS, NEW YORK, 1970.
--- EXTENDED VERSION OF THE AUTHOR'S DOCTORAL THESIS. ELEMENTARY CRITERIA AND ADDITIVE EVALUATION MODEL ARE USED. INTERACTIVE COMPUTER PROGRAM FOR DECISION MAKING IS DESCRIBED. (CODE=462EBP)

50. FISHBURN, P.C.; 'UTILITY THEORY FOR DECISION MAKING.' WILEY, NEW YORK, 1970. (CODE=45UM)

51. GINI, C.; 'MEANS' (ORIGINAL IN ITALIAN, TRANSLATED IN RUSSIAN), RUSSIAN EDITION PUBLISHED BY 'STATISTIKA', MOSCOW 1970.
--- THE BOOK PRESENTS AN EXTENSIVE ANALYSIS OF VARIOUS MEANS AND THEIR PROPERTIES. (CODE=M)

52. RAU, J.G.; 'OPTIMIZATION AND PROBABILITY IN SYSTEMS ENGINEERING', VAN NOSTRAND REINHOLD COMPANY, NEW YORK 1970. (CODE=Q)

53. RAIFFA, H.; 'DECISION ANALYSIS', ADDISON - WESLEY, READING 1970. (CODE=UX)

54. SCHARF, T.; 'WEIGHTED RANKING BY LEVELS - ONE YEAR OF EXPERIENCE', IAG QUARTERLY JOURNAL, PP. 71-91, VOL. 3, NO. 3, 1970.
--- THIRD PRESENTATION OF THE METHOD. SOME IDEAS FOR IMPROVING THE PROCEDURE ARE PROPOSED. THE COMPLETE MULTI-LEVEL PSM MODEL FOR COMPUTER SYSTEM EVALUATION IS PRESENTED. THE MODEL CONTAINS 147 PERFORMANCE VARIABLES. (CODE=236CHY)

55. OLLIVER, R.T.; 'A TECHNIQUE FOR SELECTING SMALL COMPUTERS'. DATAMATION, JANUARY 1970., 141-145.
--- THE PRACTICAL APPROACH TO COST-EFFECTIVENESS ANALYSIS USING AN ADDITIVE PSM MODEL IS PRESENTED IN DETAIL. AN EXAMPLE OF COMPUTER SELECTION IS INCLUDED. (CODE=35CQ)

56. CLERCQ, H. DE; 'CHOOSING A COMPUTER / A METHOD FOR QUANTIFYING DATA PROCESSING REQUIREMENTS AND EVALUATION CRITERIA.' IAG QUARTERLY JOURNAL, PP. 1-21, VOL. 3, NO. 1, 1970.
--- THE PSM MODEL FOR COMPUTER EVALUATION IS PRESENTED. PERFORMANCE EVALUATION RESULTS ARE INCLUDED IN THE PSM MODEL. (CODE=2346C)

57. KANTER, J.; 'MANAGEMENT GUIDE TO COMPUTER SYSTEM SELECTION AND USE.' PRENTICE - HALL, INC., ENGLEWOOD CLIFFS, NEW JERSEY, 1970, PP. 136- 155.
--- THE AUTHOR CONSIDERS THE PSM TO BE A STANDARD METHOD FOR EVALUATION AND SELECTION OF COMPUTER SYSTEMS, AND EXEMPLIFIES ITS USE. (CODE=26C)

58. DUJMOVIĆ, J.J.; BATAVELJIĆ, P.; 'AN ALGORITHM FOR DIGITAL COMPUTER COST - EFFECTIVENESS ANALYSIS' (IN SERBO - CROATIAN), PROCEEDINGS OF THE SECOND YUGOSLAV ADP CONGRESS, 1970, PP. B20 - B27.
--- A PROCEDURE FOR SCORING EFFECTIVENESS AND COST-EFFECTIVENESS OF A COMPUTER IS PROPOSED. AN ANALYSIS OF COST-EFFECTIVENESS OF COMPUTER INPUT AND OUTPUT UNITS IS PRESENTED. (CODE=Q)

59. ********** 1971 **********
LINDLEY, D.; 'MAKING DECISIONS', WILEY - INTERSCIENCE, LONDON 1971. (CODE=UX)

60. GILB, T.; 'RELIABLE DATA SYSTEMS.' UNIVERSITETSFORLAGET, OSLO, 1971.
--- THE BOOK INCLUDES THE AUTHOR'S PAPERS PRESENTING THE 'WEIGHTED RANKING BY LEVELS' METHOD. (CODE=236AC)

61. VAN OORSCHOT, J.M.; AND HOFTIJZER, W.D.; 'SOME REMARKS ON COMPUTER CHOICE.' PROCEEDINGS - ADP SEMINAR, ZAGREB, 1971.
--- THE PROCEDURE OF COMPUTER SELECTION BY THE DUTCH GOVERNMENT CENTRE FOR OFFICE MACHINES IS DESCRIBED. AN ABSTRACT FROM THE QUESTIONNAIRE CONTAINING AN EXTENSIVE LIST OF COMPONENTS FOR EVALUATION IS INCLUDED. (CODE=26Y)

62. KEENEY, R.L.; 'UTILITY INDEPENDENCE AND PREFERENCES FOR MULTIATTRIBUTED CONSEQUENCES', OPERATIONS RESEARCH, 19,1971, 875-893. (CODE=MUX)

63. DEMOODY, H.C.; BASSLER, R.A.; 'COMPUTER SYSTEM EVALUATION AND SELECTION', COLLEGE READINGS INC., ARLINGTON, U.S.A. 1971.
--- AN EXTENSIVE BIBLIOGRAPHIC RESEARCH INCLUDING AN ANNOTATED BIBLIOGRAPHY AND KEYWORD INDEX. VARIOUS ASPECTS OF COMPUTER EVALUATION ARE PRESENTED. THE SIX PSM REFERENCES ARE INCLUDED AND ANNOTATED. (CODE=B)

64. KREIBEL, C.H.; 'THE EVALUATION OF MANAGEMENT INFORMATION SYSTEMS.' SELECTED PAPERS FROM MIS COPENHAGEN 70 - AN IAG CONFERENCE, STUDENTLITTERATUR - AUERBACH, SWEDEN, PP. 227- 250. 1971. (CODE=Z)

65. DWORATSCHEK, S.; 'MANAGEMENT INFORMATION SYSTEMS' (IN GERMAN), WALTER DE GRUYTER AND CO., BERLIN AND NEW YORK 1971, PP.169- 175.
--- CHAPTER 9 OF THIS BOOK PROPOSES AND EXEMPLIFIES A COMPUTER SELECTION PROCEDURE BASED ON AN ADDITIVE PSM MODEL. THE DETERMINATION OF ELEMENTARY PREFERENCES IS DISCUSSED AND A PROCEDURE FOR DERIVING WEIGHTED FACTORS DIRECTLY FROM THE GOALS OF THE SYSTEM IS INCLUDED. (CODE=2346CQ)

66. LOPUHIN, M.M., 'PATTERN - A METHOD FOR PLANNING AND FORCASTING RESEARCH AND DEVELOPMENT PROGRAMS' (IN RUSSIAN). PUBLISHED BY 'SOVETSKOE RADIO', MOSCOW, 1971.
--- THE BOOK PRESENTS A REVIEW OF LITERATURE DESCRIBING THE METHOD 'PATTERN' AND ITS APPLICATIONS. (CODE=26D)

67. GILBERT, E.J., AND DEMMERLE, A.M., 'THE VALUE OF INCREASED PERFORMANCE AS APPLIED TO SYSTEM PROCUREMENTS.' DATA MANAGEMENT, PP. 26-29, MAY 1971.
--- THE PAPER PRESENTS A PSM MODEL FOR COMPUTER EVALUATION AND ITS USE IN THE COST - PREFERENCE ANALYSIS. (CODE=135CQ)

68. RAYMONT, P.G., 'THE COMPARATIVE EVALUATION OF COMPUTER SYSTEMS', SEMINAR ON THE APPLICATION OF COMPUTERS AS AN AID TO MANAGEMENT, GENEVA, 11-15 OCTOBER 1971. U.N. ECONOMIC COMMISSION FOR EUROPE.
--- THE COST - VALUE TECHNIQUE VERSUS PSM IS ANALYSED. THESE TWO METHODS ARE COMBINED, AND THE COMPOSITE METHOD IS PROPOSED. (CODE=3570)

69. BROCATO, L.J., 'GETTING THE BEST COMPUTER SYSTEM FOR YOUR MONEY.' COMPUTER DECISIONS, SEPT. 1971, PP 12-16. (CODE=36)

70. TURBAN, E., AND METERSKY, M.L., 'UTILITY THEORY APPLIED TO MULTIVARIABLE SYSTEM EFFECTIVENESS EVALUATION.' MANAGEMENT SCIENCE, NO. 12, AUG. 1971, PP. B817-B828. (CODE=UX)

71. ROY, B., 'PROBLEMS AND METHODS WITH MULTIPLE OBJECTIVE FUNCTIONS.' MATHEMATICAL PROGRAMMING 1(1971), PP. 239-266. (CODE=BMUX)

72. ********** 1972 **********
GROCHOW, J.M., 'A UTILITY THEORETIC APPROACH TO EVALUATION OF A TIME-SHARING SYSTEM.' IN 'STATISTICAL COMPUTER PERFORMANCE EVALUATION', EDITED BY W. FREIBERGER, ACADEMIC PRESS, 1972, PP. 25-50. (CODE=9U)

73. BOTTLER, J., HORVATH, P., AND KARGL, H., 'METHODS FOR THE COMPUTATION OF THE EFFECTIVENESS OF DATA PROCESSING' (IN GERMAN). VERLAG MODERNE INDUSTRIE, MUNCHEN, 1972. (CODE=36G)

74. LISITSCHKIN, V.A., 'THE THEORY AND PRACTICE OF FORCASTING' (IN RUSSIAN). PUBLISHED BY 'NAUKA', MOSCOW, 1972.
--- THE BOOK SYSTEMATICALLY INTRODUCES MATHEMATICAL METHODS FOR FORCASTING. THE TECHNIQUE FOR ESTIMATING FORCASTING ERRORS IS PRESENTED. A FORCASTING-BASED AUTOMATIZED TECHNIQUE USED FOR THE RESEARCH AND DEVELOPMENT DECISSION MAKING IS ALSO DISCUSSED. (CODE=W)

75. CHERNOUSKO, F.L., 'WEIGHTING FACTORS IN EXPERT JUDGEMENTS' (IN RUSSIAN). KIBERNETIKA, NO. 6, 1972, PP. 128-130.
--- AN ANALYTICAL MATRIX METHOD FOR DETERMINATION OF WEIGHTING FACTORS IS PROPOSED. THE METHOD IS VERY INTERESTING FROM THE PSM POINT OF VIEW. (CODE=W)

76. DE NEUFVILLE, R. AND R.L. KEENEY, 'USE OF DECISION ANALYSIS IN AIRPORT DEVELOPMENT IN MEXICO CITY.' IN A.DRAKER ET AL., EDS.,'ANALYSIS FOR PUBLIC SYSTEMS' (CAMBRIDGE, MASS./ MIT PRESS, 1972). (CODE=UX)

77. GOODWIN, P.G., 'A METHOD FOR EVALUATION OF SUBSYSTEM ALTERNATE DESIGNS', IEEE/TEM 1(1972), PP. 12-21. (CODE=459SRE)

78. JOSLIN, E.O., 'ANALYSIS DESIGN AND SELECTION OF COMPUTER SYSTEMS.' COLLEGE READINGS, 1972.
--- THIS SELECTION OF ARTICLES CONTAINS A SPECTRUM OF IDEAS RELATED TO THE EVALUATION OF COMPUTER SYSTEMS. THE PSM IS PRESENTED AND SOME EXAMPLES OF ITS USE ARE INCLUDED. (CODE=36AZ)

79. GOEFFRION, A.M., DYER, J.S., FEINBERG, A., 'AN INTERACTIVE APPROACH FOR MULTI-CRITERION OPTIMIZATION, WITH APPLICATION TO THE OPERATION OF AN ACADEMIC DEPARTMENT.' MANAGEMENT SCIENCE 4(1972), PP. 357-369. (CODE=XUMTOR)

80. DUJMOVIĆ, J.J., 'SOME ASPECTS OF ERROR ANALYSIS WHEN SELECTING COMPUTERS BY WEIGHTED SCORING TECHNIQUE' (IN SERBO-CROATIAN). PROCEEDINGS OF THE 7-TH INFORMATICA CONGRESS, BLED, YUGOSLAVIA, OCTOBER 1972, PAPER C4.
--- THE PAPER ANALYZES AND SYSTEMATIZES SOURCES OF ERRORS INHERENT TO WEIGHTED SCORING TECHNIQUES. VARIOUS ERRORS WHEN SELECTING COMPONENTS FOR EVALUATION, WEIGHTS, ELEMENTARY SCORES, AND AVERAGING FUNCTIONS ARE IDENTIFIED AND CLASSIFIED. SOME HINTS FOR ERROR REDUCTION ARE PROPOSED. (CODE=2367AZ)

81. DUJMOVIĆ, J.J., 'DIGITAL COMPUTER EVALUATION AND SELECTION METHODS - SURVEY AND APPRAISAL' (IN SERBO - CROATIAN). PRAKSA (YUGOSLAV JOURNAL) NO. 9 (FIRST PART) AND NO. 10 (SECOND PART), 1972.
--- AN EXTENSIVE SURVEY OF COMPUTER EVALUATION AND SELECTION METHODS IS PRESENTED. THE ARTICLE REVIEW AND COMPARE INFORMAL METHODS, FIGURES OF MERIT, KERNEL TIMING ESTIMATES, INSTRUCTION MIXES, BENCHMARKS, THRUPUT MEASUREMENTS, HARDWARE AND SOFTWARE MONITORING, COST - VALUE TECHNIQUE, SCORING METHODS, SIMULATION, AND MATHEMATICAL PROGRAMMING METHODS. THE COMPARISON OF THESE METHODS SHOWED THAT ONLY THE SCORING METHODS CAN SUPPLY THE USER WITH A COMPLETE ANSWER TO THE PRACTICALLY FORMULATED COMPUTER SELECTION PROBLEM. (CODE=AB)

82. DUJMOVIĆ, J.J., 'SOME CORRELATIONAL ASPECTS OF THE COMPENSATION OF ERRORS WHEN SELECTING DATA PROCESSING SYSTEMS BY THE WEIGHTED SCORING METHOD' (IN SERBO - CROATIAN). PROCEEDINGS OF THE 7-TH INFORMATICA CONGRESS, 1972.
--- MOST FREQUENTLY, THE PERFORMANCE VARIABLES USED FOR THE EVALUATION OF COMPUTERS ARE POSITIVELY CORRELATED. THE PAPER PRESENTS AN ANALYTICAL PROOF THAT THIS FACT CAUSE THE CORRESPONDING COMPENSATION OF ERRORS IN ESTIMATING WEIGHTS WITHIN THE SCORING MODELS. (CODE=6WM)

83. ********** 1973 **********
HANSSMANN, F., (EDITOR) 'OPERATIONAL RESEARCH IN THE DESIGN OF ELECTRONIC DATA PROCESSING SYSTEMS.' THE ENGLISH UNIVERSITIES PRESS, 1973.
--- THIS BOOK REPRESENTS THE PROCEEDINGS OF A NATO CONFERENCE AND CONTAINS NUMEROUS CONTRIBUTIONS IN THE FIELD OF COMPUTER PERFORMANCE EVALUATION AND COST - EFFECTIVENESS ANALYSIS. (CODE=QB)

84. BESHELEV, S.D., GUREVICH, F.G., 'EXPERT JUDGEMENTS' (IN RUSSIAN). PUBLISHED BY 'NAUKA', MOSCOW 1973.
--- THE BOOK REVIEWS THE PROBLEMS OF COLLECTIVE WORK OF A GROUP OF EXPERTS. MATHEMATICAL METHODS FOR AVERAGING DIFFERENT EXPERT OPINIONS ARE PRESENTED. (CODE=W)

85. YEARSLEY, R.B., AND GRAHAM, G.M.R., 'HANDBOOK OF COMPUTER MANAGEMENT.' HALSTED PRESS, NEW YORK, 1973. (CODE=36)

86. COCHRANE, J.L., ZELENY, M., (EDITORS) 'MULTIPLE CRITERIA DECISION MAKING.' UNIVERSITY OF SOUTH CAROLINA PRESS, 1973.
--- THE BOOK CONTAINS A NUMBER OF IMPORTANT CONTRIBUTIONS RELATED TO THE UTILITY THEORY AND PSM. (CODE=ABMU)

87. MAC CRIMMON, K.R., 'AN OVERVIEW OF MULTIPLE OBJECTIVE DECISION MAKING.' PUBLISHED IN 'MULTIPLE CRITERIA DECISION MAKING', EDITED BY J.L. COCHRANE AND M. ZELENY, UNIVERSITY OF SOUTH CAROLINA PRESS, 1973. PP. 18-44. (CODE=AB)

88. BAUGUT, G., 'MODELS FOR COMPUTER SELECTION' (IN GERMAN). PUBLISHED BY R. MULLER, KOLN - BRAUNSFELD, 1973.
--- VARIOUS EXAMPLES OF THE PSM USE FOR COMPUTER SELECTION ARE PRESENTED. (CODE=236CQAB)

89. TIMRECK, E.M., 'COMPUTER SELECTION METHODOLOGY'. COMPUTING SURVEYS, VOL. 5, NO. 4, DECEMBER 1973, PP. 199-222. (CODE=ABZ)

90. DUJMOVIĆ, J.J., 'TWO INTEGRALS RELATED TO MEANS', PUBLIKACIJE ELEKTROTEHNICKOG FAKULTETA, SERIJA MATEMATIKA I FIZIKA, NO. 412 - NO. 460, PP. 231 - 232, BEOGRAD 1973.
--- THE PAPER PRESENTS THE CALCULATION OF THE DEFINITE INTEGRALS WITHIN THE N-DIMENSIONAL UNIT HYPERCUBE FOR THE FUNCTION F(X1,...,XN)**K , BOTH FOR F=MIN AND F=MAX. (CODE=M)

91. DUJMOVIĆ, J.J., 'A GENERALIZATION OF SOME FUNCTIONS IN CONTINUOUS MATHEMATICAL LOGIC' (IN SERBO CROATIAN). PROCEEDINGS OF THE INFORMATICA CONGRESS (BLED, YUGOSLAVIA), 1973, PAPER D27.
--- THE FUNCTION ENABLING A CONTINUOUS TRANSFORMATION FROM DISJUNCTION TO CONJUNCTION AND REPRESENTING THEIR GENERALIZATION IS PROPOSED. THIS WAS THE FIRST ATTEMPT TO INTRODUCE QUASI-CONJUNCTION AND QUASI-DISJUNCTION IN THE CONTINUOUS LOGIC MODELS FOR SYSTEM EVALUATION. (CODE=M)

92. DUJMOVIĆ, J.J., 'MIXED AVERAGING BY LEVELS (MAL) - SYSTEM AND COMPUTER EVALUATION METHOD' (IN SERBO - CROATIAN). PROCEEDINGS OF THE INFORMATICA CONGRESS (BLED, YUGOSLAVIA), 1973, PAPER D28.
--- THE MULTI-LEVEL EXTENDED CONTINUOUS LOGIC MODEL OF A COMPLEX CRITERION FOR SYSTEM EVALUATION IS PROPOSED. THE MAL METHOD FOR SYSTEM EVALUATION REPRESENTS A GENERALIZATION OF EXISTING SCORING METHODS ELIMINATING THEIR MAIN SHORTCOMINGS. PRACTICAL OPPORTUNITIES FOR COMPUTER SELECTION USING THE MAL METHOD ARE ALSO CONSIDERED. (CODE=248C)

93. DUJMOVIĆ, J.J., 'CONTRIBUTIONS TO THE WEIGHTED SCORING METHOD FOR COMPUTER EVALUATION' (IN SERBO - CROATIAN). UNPUBLISHED M.S. THESIS, UNIVERSITY OF BELGRADE, DEPARTMENT OF ELECTRICAL ENGINEERING, 1973. (CODE=2369CGWM)

94. ********** 1974 **********
COUGER, J.D., KNAPP, R.W., (EDITORS) 'SYSTEM ANALYSIS TECHNIQUES.' WILEY, 1974.
--- PART III OF THE BOOK CONTAINS VARIOUS PRESENTATIONS OF THE COST - EFFECTIVENESS ANALYSIS. (CODE=Q)

95. SHERIDAN, T.B., AND FERRELL, W.R., 'MAN-MACHINE SYSTEMS / INFORMATION, CONTROL AND DECISION MODELS OF HUMAN PERFORMANCE.' THE M.I.T. PRESS, 1974. (CODE=WUX)

96. ELDIN, H.K., AND CROFT, F.M., 'INFORMATION SYSTEMS - A MANAGEMENT SCIENCE APPROACH.' PETROCELLI BOOKS, NEW YORK, 1974. (CODE=36)

97. HUBER, G.P., 'METHODS FOR QUANTIFYING SUBJECTIVE PROBABILITIES AND MULTI-ATTRIBUTE UTILITIES.' DECISION SCIENCES, NO. 3, JULY 1974, PP. 430-458. (CODE=9UXR)

98. MIRKIN, B.G., 'PROBLEMS OF EXPERT JUDGEMENT' (IN RUSSIAN). PUBLISHED BY 'NAUKA', MOSCOW, 1974.
--- MATHEMATICAL THEORY OF AGGREGATING THE INDIVIDUAL PREFERENCE INFORMATION IS PRESENTED IN THIS BOOK. THE THEORY CAN BE USED IN PRACTICAL WORK OF EXPERT TEAMS. (CODE=W)

99. BESHELEV, S.D., GUREVICH, F.G., 'MATHEMATICAL AND STATISTICAL METHODS FOR EXPERT TEAMS' (IN RUSSIAN). PUBLISHED BY 'STATISTIKA', MOSCOW, 1974.
--- THE BOOK PRESENTS A NUMBER OF STATISTICAL TECHNIQUES FOR OBTAINING QUANTITATIVE INFORMATION FROM THE EXPERT TEAM JUDGEMENTS. THE ROLE OF EXPERT - BASED METHODS IN OPERATIONS RESEARCH IS DISCUSSED. THE TECHNIQUES FOR ORGANIZING THE WORK OF EXPERT TEAMS AND FOR PROCESSING INFORMATION OBTAINED FROM A GROUP OF EXPERTS ARE PRESENTED. THE CONCORDANCE ANALYSIS OF EXPERT JUDGEMENTS IS DESCRIBED IN DETAIL. (CODE=W)

100. WILLIAMS, G.Z., WILLIAMS, R.L., 'CLINICAL LABORATORY SUBSYSTEM, SECTION F - EQUIPMENT EVALUATION AND SELECTION'. CHAPTER 7 IN M. COLLEN (ED.), 'HOSPITAL COMPUTER SYSTEMS', J. WILEY 1974, PP. 174-193.
--- ALL COMPETITIVE SYSTEMS ARE 'FILTERED' THROUGH THE 'COST FILTER', 'TECHNICAL FILTER' AND 'SUPPLIER FILTER'. THE TECHNICAL FILTER CONSISTS OF 62 'TECHNICAL' COMPONENTS FOR EVALUATION OF A CLINICAL LABORATORY COMPUTER. IN ADDITION, TEN SUPPLIER COMPONENTS FOR EVALUATION ARE PROPOSED. THE WEIGHTING IS CONSIDERED ARBITRARY. ONE SMALL 10 - COMPONENT POINT - ADDITIVE EXAMPLE IS INCLUDED. PURCHASE PROCEDURE AND CONTRACTUAL AGREEMENTS ARE ALSO CONSIDERED. (CODE=735EY)

101. RAKOV, G.K., 'METHODS FOR THE OPTIMIZATION OF COMPUTER SYSTEM STRUCTURE' (IN RUSSIAN). PUBLISHED BY 'ENERGIA', MOSCOW, 1974.
--- AT THE VERY BEGINNING OF THIS BOOK THE AUTHOR CRITICIZES THE ONE-LEVEL ADDITIVE SCORING MODEL AS INADEQUATE TO BE APPLIED IN THE OPTIMIZATION OF COMPUTER SYSTEM STRUCTURE. INTERESTING ENOUGH, IN THE AUTHOR'S OPINION 'IT IS NOT ACCEPTABLE TO AGGREGATE WITHIN A SINGLE CRITERION SOME INCOMMENSURABLE PERFORMANCE VARIABLES (E.G. THE MEMORY SPACE AND THRUPUT).' (CODE=5Z0)

102. BURCH, J.G. JR., STARTER, F.R. JR., 'INFORMATION SYSTEMS / THEORY AND PRACTICE', HAMILTON PUBLISHING CO., SANTA BARBARA, CALIFORNIA, 1974.
--- CHAPTER 12 OF THE BOOK PRESENTS THE USE OF PSM FOR SYSTEM EVALUATION. (CODE=5A)

103. DUJMOVIĆ, J.J., 'COMPONENTS FOR EVALUATION OF COMPUTER SYSTEMS' (IN SERBO - CROATIAN). PRAKSA (YUGOSLAV JOURNAL), OCTOBER 1974.
--- AN EXTENSIVE ANNOTATED LIST CONTAINING COMPONENTS FOR EVALUATION OF MODERN DIGITAL COMPUTERS IS PRESENTED. COMPONENTS FOR EVALUATION OF SUPPLIER, HARDWARE AND SOFTWARE ARE INCLUDED. (CODE=1)

104. DUJMOVIĆ, J.J., KLEM, N., 'AN APPROACH TO EVALUATION AND COMPARISON OF FORTRAN AND ALGOL COMPILERS' (IN SERBO - CROATIAN). PROCEEDINGS OF THE INFORMATICA CONGRESS (BLED, YUGOSLAVIA), 1974, PAPER 4.39.
--- A PROCEDURE FOR EVALUATION AND COMPARISON OF FORTRAN AND ALGOL COMPILERS IS PROPOSED. THE PROCEDURE IS BASED ON REGRESSION ANALYSIS AND THE MAL METHOD, AND IS ILLUSTRATED BY A COMPARISON OF FORTRAN AND ALGOL COMPILERS FOR THE IBM 1130 COMPUTER. (CODE=E9)

105. SLAVIĆ, D.V., DUJMOVIĆ, J.J., 'NUMERICAL COMPUTATION OF WEIGHTED POWER MEANS', PUBLIKACIJE ELEKTROTEHNICKOG FAKULTETA BEOGRAD, SERIJA MATEMATIKA I FIZIKA, BEOGRAD, 1974.
--- THE ERRORS ARISING IN COMPUTATION OF VALUES OF THE WEIGHTED POWER MEANS ARE ANALYSED AND A PROCEDURE FOR THEIR COMPENSATION IS PROPOSED. (CODE=M)

106. DUJMOVIĆ, J.J., 'WEIGHTED CONJUNCTIVE AND DISJUNCTIVE MEANS AND THEIR APPLICATION IN SYSTEM EVALUATION', PUBLIKACIJE ELEKTROTEHNICKOG FAKULTETA BEOGRAD, SERIJA MATEMATIKA I FIZIKA, BEOGRAD 1974.
--- THE DEFINITION OF WEIGHTED CONJUNCTIVE AND DISJUNCTIVE MEANS IS PROPOSED, AND THEIR BASIC PROPERTIES INVESTIGATED. THESE MEANS ARE DERIVED FROM WEIGHTED POWER MEANS FOR USE IN FORMAL MODELS FOR ESTIMATION OF THE TRUE VALUE OF A CLASS OF COMPLEX STATEMENTS IN CONTINUOUS LOGIC. (CODE=MB)

107. DUJMOVIĆ, J.J., 'NEW RESULTS IN DEVELOPMENT OF THE 'MIXED AVERAGING BY LEVELS' METHOD FOR SYSTEM EVALUATION' (IN SERBO - CROATIAN). PROCEEDINGS OF THE INFORMATICA CONGRESS (BLED, YUGOSLAVIA), 1974, PAPER 4.36.
--- THE PROPOSED IMPROVEMENT OF THE MAL METHOD CONSISTS IN INTRODUCING THE CONJUNCTIVE AND DISJUNCTIVE MEANS, NEW ASYMETRIC AVERAGING OPERATORS, AND A SERIES OF SPECIFIC SYSTEM INDICATORS NECESSARY FOR SYSTEM EVALUATION. (CODE=48S)

108. DUJMOVIĆ, J.J., 'OPTIMIZATION OF COMPLEX SYSTEMS USING THE MAL METHOD' (IN SERBO - CROATIAN). PROCEEDINGS OF THE INFORMATICA CONGRESS (BLED, YUGOSLAVIA), 1974, PAPER 4.38.
--- AN ALGORITHM FOR OPTIMIZATION OF ARBITRARY COMPLEX SYSTEM, WHOSE EFFECTIVENESS CAN BE EXPRESSED BY THE MAL METHOD, IS PROPOSED. THE ALGORITHM SERVES TO MAXIMIZE THE EFFECTIVENESS FOR A GIVEN COST, OR TO MINIMIZE THE SYSTEM COST FOR A GIVEN EFFECTIVENESS, AND TO DERIVE THE COMPLEX SYSTEM OPTIMAL CONFIGURATION. (CODE=48EO)

109. DUJMOVIĆ, J.J., DŽIGURSKI, O.D., 'EVALUATION AND COMPARISON OF ANALOG COMPUTERS' (IN SERBO - CROATIAN). PROCEEDINGS OF THE INFORMATICA CONGRESS (BLED, YUGOSLAVIA), 1974, PAPER 4.37.
--- THE COMPLETE GLOBAL CRITERION FOR EVALUATION OF MODERN ANALOG COMPUTERS IS PROPOSED. THE GLOBAL CRITERION INCLUDE 39 ELEMENTARY CRITERIA AND THE CRITERIA AGGREGATION STRUCTURE BASED ON THE USE OF EXTENDED CONTINUOUS LOGIC FUNCTIONS. (CODE=248CS)

110. ********** 1975 **********
KITAEV, N.N., 'EXPERT GROUP JUDGEMENTS' (IN RUSSIAN). PUBLISHED BY 'ZNANIE', MOSCOW, 1975.
--- THE BOOK DISCUSSES USEFULNESS AND TECHNIQUES OF FORCASTING USING EXPERT JUDGEMENTS. PROCEDURES FOR ACQUISITION AND PROCESSING OF EXPERT INFORMATION ARE PRESENTED. (CODE=W)

111. BAKER, N., FREELAND, J. 'RECENT ADVANCES IN R AND D BENEFIT MEASUREMENT AND PROJECT SELECTION METHODS.' MANAGEMENT SCIENCE 10(1975) 1164-1175.
--- AN EXTENSIVE SURVEY INCLUDING THE ROLE OF PSM IN R AND D BENEFIT MEASUREMENT IS PRESENTED. (CODE=ABD)

112. VAN OORSCHOT, J.M., 'NATIONAL POLICIES FOR THE ACQUISITION OF HARDWARE, SOFTWARE AND SYSTEMS', PP. 280 - 285 IN 'ECONOMICS OF INFORMATICS' EDITED BY A.B. FRIELINK, NORTH - HOLLAND 1975.
--- COMPUTER EVALUATION PROCESS PRACTICED BY THE DUTCH GOVERNMENT OFFICE FOR AUTOMATION AND MECHANIZATION (KMC) IS PRESENTED. THE PSM IS USED FOR MEDIUM AND LARGE COMPUTER SYSTEM EVALUATION. (CODE=Y)

113. FRIELINK, A.B. (EDITOR), 'ECONOMICS OF INFORMATICS.' NORTH - HOLLAND, 1975. (CODE=QUX)

114. ZANGEMEISTER, C.: 'MEASUREMENT OF EFFECTIVENESS OF COMPUTERIZED INFORMATION SYSTEMS FROM A MANAGEMENT POINT OF VIEW THROUGH UTILITY ANALYSIS', PP. 440 - 451 IN 'ECONOMICS OF INFORMATICS' EDITED BY A.B. FRIELINK, NORTH - HOLLAND 1975.
--- REVIEW OF AN ELABORATED AND SOFTWARE SUPPORTED UTILITY THEORETICAL SYSTEM FOR COMPUTERIZED INFORMATION SYSTEM EVALUATION IS PRESENTED. (CODE=6GPSUY)

115. KURZBAN, S.A., HEINES, T.S., AND SAYERS, A.P., 'OPERATING SYSTEMS PRINCIPLES', PETROCELLI, NEW YORK 1975.
--- APPENDIX C ENTITLED 'OPERATING SYSTEM FUNCTIONS' DESCRIBES AN EXTENSIVE HIERARCHICALLY ORGANIZED LIST OF COMPONENTS FOR EVALUATION OF A MODERN OPERATING SYSTEM, REPRESENTING USEFUL BASIS FOR COMPARING DIFFERENT SYSTEMS IN A CONSISTENT WAY. (CODE=2)

116. CHURCHMAN, C.W., WESNER, R.W., (EDITORS) 'SYSTEMS AND MANAGEMENT ANUAL 1975.' PETROCELLI, NEW YORK, 1975.
--- CHAPTER 7 PRESENTS SOME RELATIONS AMONG PATTERN RECOGNITION METHODS AND PSM. CHAPTER 10 PRESENTS A REVIEW OF MULTI-ATRIBUTE UTILITY MODELS. MULTIPLICATIVE UTILITY FUNCTIONS ARE INTRODUCED IN CHAPTER 12. (CODE=MUX)

117. KEENY, R.L., NAIR, K., 'DECISION ANALYSIS FOR THE SITING OF NUCLEAR POWER PLANTS - THE RELEVANCE OF MULTIATRIBUTE UTILITY THEORY,' PROCEEDINGS OF THE IEEE 3(1975) 494-501. (CODE=9UE)

118. WHITE, D.J., 'DECISION METHODOLOGY', J. WILEY, LONDON, 1975. (CODE=BUX)

119. KAHNE, S., 'A CONTRIBUTION TO DECISION MAKING IN ENVIRONMENTAL DESIGN.' PROCEEDINGS OF THE IFEE 3(1975) 518-528.
--- A FUZZY SET ORIENTED PSM MODEL, AND THE CASE STUDY OF ITS APPLICATION IN THE SITE SELECTION FOR AN URBAN INSTITUTION ARE PRESENTED. (CODE=5FR)

120. KAHNE, S., 'A PROCEDURE FOR OPTIMIZING DEVELOPMENT DECISIONS,' AUTOMATICA, VOL. 11, PP. 261-269, PERGAMON PRESS, 1975.
--- THE PAPER INTRODUCES THE FUZZY SET THEORY IN WEIGHTED SCORING. ACCORDING TO THE AUTHOR, 'IN ANY REALISTIC PROBLEM FORMULATION, THE CRITERIA ARE NOT PRECISELY DEFINED, THEY ARE FUZZY, THE RELATIVE IMPORTANCE OF EACH CRITERION IS ALSO FUZZY, THE TECHNIQUE PROPOSED IN THE PAPER ACCOUNTS FOR THIS UNCERTAINTY IN ALL ASPECTS OF THE PRORLEM AND YIELDS PROBABILISTIC ANSWERS'. (CODE=5FR)

121. DUJMOVIĆ, J.J., 'GRAPHIC APPROACH TO WEIGHTED CONJUNCTIVE AND DISJUNCTIVE MEANS CALCULATION' PUBLIKACIJE ELEKTROTEHNICKOG FAKULTETA BEOGRAD, SERIJA MATEMATIKA I FIZIKA, NO. 498-541 (1975), 191-196. (CODE=M)

122. DUJMOVIĆ, J.J., 'EXTENDED CONTINUOUS LOGIC AND THE THEORY OF COMPLEX CRITERIA', PUBLIKACIJE ELEKTROTEHNICKOG FAKULTETA BEOGRAD, SERIJA MATEMATIKA I FIZIKA, NO. 498-541 (1975), 197-216.
--- THE EXTENDED CONTINUOUS LOGIC (ECL), A GENERALIZATION OF CONTINUOUS LOGIC BASED ON THE INTRODUCTION OF VARIABLE DEGREE OF CONJUNCTION AND DISJUNCTION IN COMPLEX CONJUNCTIVE AND DISJUNCTIVE STATEMENTS, IS PROPOSED IN THIS PAPER. THE PROPERTIES OF ECL, AS AN ALGEBRAIC STRUCTURE, ARE ANALYSED AND VARIOUS ECL FUNCTIONS ARE DEFINED AND CLASSIFIED. (CODE=48M)

123. DUJMOVIĆ, J.J. 'EVALUATION OF DIGITAL COMPUTERS USING THE SYSTEM EVALUATION METHOD MAL' (IN SERBO - CROATIAN), PROCEEDINGS OF THE INFORMATICA CONGRESS (BLED, YUGOSLAVIA), 1975, PAPER 6.9.
--- A CRITERION FOR EVALUATION OF MEDIUM AND LARGE DIGITAL COMPUTERS IS PROPOSED. THE CRITERION INCLUDE 113 ELEMENTARY CRITERIA, THEREOF 43 ARE RELATED TO COMPUTER SUPPLIER, 36 TO HARDWARE, AND 34 TO SOFTWARE. (CODE=248C)

124. DUJMOVIĆ, J.J., 'EVALUATION OF COMPLEX SYSTEMS' (IN SERBO - CROATIAN), UNPUBLISHED DOCTORAL DISSERTATION, UNIVERSITY OF BELGRADE, DEPARTMENT OF ELECTRICAL ENGINEERING, 1975. (CODE=248CMOPS)

125. ********** 1976 **********
ZELENY, M. (EDITOR), 'MULTIPLE CRITERIA DECISION MAKING (KYOTO 1975)', SPRINGER - VERLAG, 1976.
--- THE PROCEEDINGS OF THE KYOTO CONGRESS CONTAIN A NUMBER OF IMPORTANT CONTRIBUTIONS TO THE DECISION ANALYSIS. ESPECIALLY INTERESTING IS THE EXTENSIVE BIBLIOGRAPHIC RESEARCH BY M. ZELENY PRESENTED AT THE END OF THE BOOK. (CODE=BUM)

126. KEENEY, R.L., AND RAIFFA, H., 'DECISIONS WITH MULTIPLE OBJECTIVES / PREFERENCES AND VALUE TRADEOFFS.' WILEY, NEW YORK, 1976. (CODE=69UTX)

127. GILB, T., 'SOFTWARE METRICS.' STUDENTLITTERATUR, LUND, 1976.
--- THE USE OF THE PSM FOR EVALUATING THE COMPLEX SOFTWARE SYSTEMS IS PRESENTED. INCLUDED ARE THE SAMPLES FROM THE PSM MODELS RELATED TO THE EVALUATION AND COMPARISON OF OPERATING SYSTEMS, DATA BASE MANAGEMENT SYSTEMS, PROGRAMMING LANGUAGES, ETC. (CODE=236EC)

128. ZIONTS, S. WALLENIUS, J., 'AN INTERACTIVE PROGRAMMING METHOD FOR SOLVING THE MULTIPLE CRITERIA PROBLEM.' MANAGEMENT SCIENCE 6(1976) 652-663.
--- LINEAR ADDITIVE UTILITY MODELS AND THEIR OPTIMIZATION ARE PRESENTED. (CODE=MUXOE)

129. THIRIEZ, H., AND ZIONTS, S., (EDITORS) 'MULTIPLE CRITERIA DECISION MAKING.' SPRINGER VERLAG, BERLIN, 1976. (CODE=UX)

130. DUJMOVIĆ, J.J., 'CRITERIA AGGREGATION TECHNIQUE FOR EVALUATION, OPTIMIZATION AND SELECTION OF COMPUTER SYSTEMS,' PAPER PRESENTED AT THE OECD ANKARA MEETING, MARCH 1976 (PUBLISHED ALSO IN GREEK IN THE BULLETIN OF THE GENERAL DIRECTORATE OF PUBLIC ADMINISTRATION, NO. 3, PP. 51-70, ATHENS 1976).
--- SURVEY OF A COMPLETE PROCEDURE FOR COMPUTER SYSTEM ACQUISITION USING THE PREFERENCE SCORING APPROACH (METHOD MAL) IS PRESENTED. (CODE=48A)

131. DUJMOVIĆ, J.J., 'SYSTEM EVALUATION LANGUAGE (SEL) - PROGRAMMING LANGUAGE FOR EVALUATION, COMPARISON AND OPTIMIZATION OF COMPLEX SYSTEMS' (IN SERBO - CROATIAN), PROCEEDINGS OF THE INFORMATICA CONGRESS (BLED, YUGOSLAVIA), 1976, PAPER 1/121.
--- THE SYNTAX AND THE BASIC SEMANTICS OF THE PROGRAMMING LANGUAGE SEL ARE DESCRIBED. SEL IS A PROBLEM - ORIENTED LANGUAGE DESIGNED TO EVALUATE, COMPARE AND OPTIMIZE COMPLEX SYSTEMS USING THE MAL METHOD. COMPUTER IMPLEMENTATION OF THIS LANGUAGE IS MADE POSSIBLE BY THE SEL INTERPRETER WRITTEN IN BASIC FORTRAN IV. (CODE=P)

132. DUJMOVIĆ, J.J., TOMAŠEVIĆ, I.D., 'CRITERIA DATA BASE SYSTEM (CDBS)' (IN SERBO - CROATIAN), PROCEEDINGS OF THE INFORMATICA CONGRESS (BLED, YUGOSLAVIA), 1976, PAPER 1/122.
--- THE PAPER PRESENTS CDBS - AN INTERACTIVE PROGRAMMING SYSTEM DESIGNED FOR CREATING, UPDATING AND EDITING OF THE DATA BASE CONTAINING (1) ELEMENTARY CRITERIA FOR SYSTEM EVALUATION, AND (2) THE RELATED CRITERIA AGGREGATION STRUCTURE. CDBS AND THE SYSTEM EVALUATION LANGUAGE (SEL) ARE TWO COMPLEMENTARY BASIC SOFTWARE TOOLS DESIGNED FOR COMPLEX SYSTEMS EVALUATION, COMPARISON, AND OPTIMIZATION USING THE MAL METHOD. (CODE=P)

133. DUJMOVIĆ, J.J., 'A PROCEDURE FOR DETERMINING OPTIMAL CONFIGURATIONS OF THE COMPUTING UNITS OF ANALOG AND HYBRID COMPUTERS' (IN SERBO - CROATIAN), PROCEEDINGS OF THE ETAN CONGRESS 1976, PP. 1181 - 1198.
--- A PROCEDURE FOR DETERMINING OPTIMAL NUMBERS OF INTEGRATORS, TRACK-STORES, SUMMERS, INVERTERS, DIGITAL COEFFICIENT UNITS, MANUAL POTENTIOMETERS, UNIVERSAL FUNCTION GENERATORS, FIXED FUNCTION GENERATORS, MULTIPLIERS, D/A SWITCHES, FUNCTIONAL RELAYS, COMPARATORS, AND VARIOUS PARALLEL LOGIC UNITS FOR AN ANALOG OR HYBRID COMPUTER IS PROPOSED. OPTIMIZATION IS CARRIED OUT USING THE SYSTEM EVALUATION METHOD MAL, AND PROGRAMMING LANGUAGE SEL. (CODE=248CO)

134. DUJMOVIĆ, J.J., 'EVALUATION, COMPARISON AND OPTIMIZATION OF HYBRID COMPUTERS USING THE THEORY OF COMPLEX CRITERIA,' IN 'SIMULATION OF SYSTEMS' EDITED BY L. DEKKER, NORTH - HOLLAND, 1976, PP. 353 - 366.
--- THE CRITERION FOR MEDIUM-SIZED HYBRID COMPUTERS EVALUATION, COMPARISON, AND OPTIMIZATION IS PROPOSED. THE CRITERION INCLUDES SUBCRITERIA FOR EVALUATION OF ANALOG COMPUTER, DIGITAL COMPUTER HARDWARE, HYBRID INTERFACE UNIT, AND HYBRID SOFTWARE. (CODE=248CEOY)

135. ********** 1977 **********
HENN, R., MOESCHLIN, O., (EDITORS) 'MATHEMATICAL ECONOMICS AND GAME THEORY.' SPRINGER - VERLAG, BERLIN, 1977, PP. 346 - 369. (CODE=5MU)

136. BAAS, S.M., AND KWAKERNAAK, H., 'RATING AND RANKING OF MULTIPLE-ASPECT ALTERNATIVES USING FUZZY SETS.' AUTOMATICA, VOL. 13, PP. 47-58. PERGAMON PRESS, 1977.
--- A METHOD FOR SOLVING THE MULTIPLE - ALTERNATIVE DECISION PROBLEMS UNDER UNCERTAINTY IS PROPOSED. THE RATINGS AND WEIGHTS ARE CONSIDERED AS FUZZY QUANTITIES. (CODE=5F)

137. KLEIJNEN, J.P.C., 'SCORING METHODS, MULTIPLE CRITERIA AND UTILITY ANALYSIS.' CHAPTER 4 IN THE AUTHOR'S BOOK 'COMPUTERS AND PROFIT' (TO APPEAR).
--- VARIOUS SCORING METHODS ARE CRITICALLY REVIEWED. THE RELATIONS OF SCORING METHOD WITH MULTIPLE CRITERIA DECISION METHODS AND UTILITY ANALYSIS ARE PRESENTED. THE POSSIBILITIES FOR COMPUTER SELECTION USING THESE METHODS ARE ANALYZED. (CODE=ABRUTZ)

138. DUJMOVIĆ, J.J., 'PROFESSIONAL EVALUATION AND SELECTION OF COMPUTER SYSTEMS.' PROCEEDINGS OF THE INFORMATICA CONGRESS, BLED (YUGOSLAVIA), 1977.
--- THE PAPER PRESENTS A CONDENSED SURVEY OF A MULTI-STEP PROCEDURE FOR COMPUTER SYSTEM EVALUATION, OPTIMIZATION AND SELECTION. THE PROPOSED TECHNIQUE REPRESENTS A PART OF AN INTEGRAL PROCEDURE FOR REALIZATION OF ADP CENTERS, AND HAS BEEN PARTICULARLY DESIGNED FOR PROFESSIONAL USE BY ORGANIZATIONS SPECIALIZED FOR SYSTEMS AND COMPUTERS EVALUATION AND SELECTION. (CODE=48ABC)

CLASSIFICATION OF BIBLIOGRAPHIC ENTRIES

CODE = 1
8 36 41 44 67 103

CODE = 2
15 16 24 26 32 35 45 49 54 56
57 60 61 65 66 80 88 92 93 100
109 115 123 124 127 133 134

CODE = 3
3 8 11 12 15 18 23 25 28 35
36 38 42 43 44 45 47 54 55 56
60 65 67 68 69 73 78 80 85 88
93 96 100 127

CODE = 4
16 17 24 26 34 36 38 41 49 50
56 65 77 92 107 108 109 122 123 124
130 133 134 138

CODE = 5
3 8 10 11 17 22 34 35 36 42
43 44 50 55 67 68 77 100 101 102
119 120 135 136

CODE = 6
12 15 18 23 24 25 26 28 38 41
45 47 49 54 56 57 60 61 65 66
69 73 78 80 82 85 88 93 96 114
126 127

CODE = 7
16 42 68 80

CODE = 8
92 106 107 108 109 122 123 124 130 133
134 138

CODE = 9
33 37 48 72 77 93 97 104 117 126

CODE = A
20 24 27 33 34 38 42 43 60 78
80 81 86 87 88 89 102 111 130 137
138

CODE = B
20 24 27 31 33 34 35 46 49 54
63 71 79 81 83 86 87 88 89 111
118 125 137 138

CODE = C
8 15 24 26 35 36 44 54 55 56
57 60 65 67 88 92 93 109 123 124
127 133 134 138

CODE = D
22 27 46 66 111

CODE = E
10 16 17 23 24 26 30 36 38 42
45 49 77 100 104 108 117 127 128 134

CODE = F
119 120 136

CODE = G
3 4 11 23 73 93 114

CODE = M
7 17 29 33 37 40 41 50 51 62
71 79 82 86 90 91 93 105 106 116
121 122 124 125 128 135

CODE = O
79 101 108 124 128 133 134

CODE = P
33 49 114 124 131 132

CODE = Q
19 30 31 32 33 39 42 52 55 58
65 67 68 83 88 94 113

CODE = R
16 38 46 77 97 119 120 137

CODE = S
16 42 77 107 109 114 124

CODE = T
42 79 126 137

CODE = U
1 2 7 17 29 33 37 41 50 53
59 62 70 71 72 76 79 86 95 97
113 114 116 117 118 125 126 128 129 135
137

CODE = W
38 48 74 75 82 84 93 95 98 99
110

CODE = X
1 2 5 6 7 9 10 13 14 21
29 30 37 39 53 59 62 70 71 76
79 95 97 113 116 118 126 128 129

CODE = Y
8 24 38 45 54 61 100 112 114 134

CODE = Z
19 32 34 41 43 64 78 80 89 101
137

ACKNOWLEDGMENT

THE AUTHOR IS INDEBTED TO MR. TOM GILB AND TO DR. J.P.C. KLEIJNEN FOR THE MOTIVATION TO WRITE THIS PAPER, AND FOR HELP AND USEFUL CRITICISM DURING ITS PREPARATION.

rešitve nekaterih problemov krmiljenja objekta z mikro procesorjem v realnem času

p. kolbezen
b. mihovilović
z. milavc

UDK 681.3 - 181.4

Institut J.Stefan,
Univerza v Ljubljani, Ljubljana

Članek obravnava nekatere probleme komuniciranja mikro računalnika s perifernimi napravami med izvajanjem glavnega opravila in sočasnem generiranju linearne funkcije časa. Ta je realizirana s programsko opremo brez ure v materialni opremi. Posebej so opisane možnosti programskih prekinitev pri nekaterih popularnejših mikro računalnikih in posebnosti, ki jih mora programer upoštevati pri programiranju takšnih prekinitev. K splošni obravnavi so dodane rešitve na konkretnem primeru krmiljenja objekta z mikro procesorjem v realnem času.

SOLUTIONS OF SOME PROBLEMS OF THE REAL-TIME MICROPROCESSOR CONTROL. This article discusses some problems of the microprocessor communication with the peripheral devices during the general function execution and the generation of the linear time-function. This function is realised in software without the hardware clock device. The possibilities of the program interrupts of some more popular microprocessors are treated. A particularities of the interrupt programming to which the programmer must be known are pointed out. To the general treatment solutions of some problems of the real-time microprocessor control are added.

1. UVOD

Intel je leta 1974 realiziral mikro računalnik INTEL 8080. Ta je v preteklih nekaj letih predstavljal, lahko bi rekli, nekakšen industrijski standard za mikro računalnik, kot ga je pred tem že predstavljal INTEL 1103 za pomnilnik. Minimalni sistem INTEL 8080 sestavljata dva posebna čipa in pet TTL sklopov. Ta sistem je približno petkrat hitrejši od njegovega uspelega predhodnika 8008, ki zavzema tudi 60 % večjo površino čipov. Novi Intel je bil ob svojem rojstvu najhitrejši mikro procesor na tržišču. Ostal je prilagodljiv običajnim ROM in RAM pomnilnikom ter programski opremi svojega prednika, ima pa več instrukcij (74) ter omogoča večkratno programsko prekinitev. Zaradi premajhne zmogljivosti in hitrosti je ostala njegova uporaba omejena. Zato so Intel in drugi proizvajalci tedaj obstoječih mikro procesorjev kaj kmalu dobili tekmece. Večjo zmogljivost so dosegli s spremembami v sistemski arhitekturi in naprednejšo tehnologijo. Slednja bo tudi v bodoče omogočala vse večje hitrosti. Med več ali manj konkurenčnimi proizvodi, ki so se kmalu pojavili na tržišču, štejemo predvsem AMI 6800, IMP-16C, Mostek F8 in Z80. Opažamo pa, da aplikacije le-teh še vedno zaostajajo, kot je bilo to dokaj izrazito tudi za Intel 8080 v prvem obdobju njegovega obstoja. Del problema je v standardizaciji vezij, primanjkuje pa tudi boljša materialna in programska oprema. Vendar je Intel v tem pogledu uspešnejši, moral pa bi poskrbeti za še širšo uporabo svojih izdelkov. Pri tem je pomembno, da je njegova dokumentacija skrbno izdelana in kot taka dobro uporabljiva za povprečnega sistemskega načrtovalca.

Ena od pomanjkljivosti centralne procesne enote in osnovnega sistema Intel 8080 je, da nima posebne ure, ki bi omogočala generiranje časovnih funkcij. Ta pomanjkljivost se na primer kaže, kadar moramo časovno funkcijo generirati pri obratovanju mikroprocesorskega sistema v realnem času. V določenih okoliščinah, ki so odvisne zlasti od časovnih zahtev krmiljenja objekta, ali drugače rečeno, od časovnih karakteristik glavnega opravila, in od zahteve po točnosti generiranja časovne funkcije, je mogoče problem tudi programsko rešiti. V referatu je prikazan primer takšne rešitve, medtem ko so obravnavani tudi nekateri povsem splošni problemi komuniciranja računalnika s perifernimi napravami. Te so uporabljene za vnašanje podatkov, ki so potrebni za generiranje časovne funkcije. Tako so poleg časovnega generatorja na konkretnem primeru podane tudi rešitve problemov, ki jih najčešče srečujemo pri mikroprocesorskem krmiljenju objekta v realnem času.

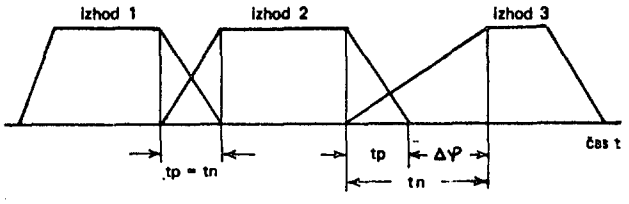
2. FUNKCIJA ČASOVNEGA GENERATORJA

Predno si ogledamo konfiguracijo uporabljenega računalniškega sistema, si oglejmo vlogo časovnega generatorja v glavnem opravilu krmiljenja objekta v realnem času.

Krmiljeni objekt ima množico analognih izhodov, katerih vrednost lahko zvezno spreminjamo. Vrednost vsakega izhoda, ki ga moramo adresirati, določimo z digitalnim podatkom na vhodu krmiljenega objekta. Imamo torej množico vhodnih podatkov, ki določajo izhodno stanje objekta. Zaradi serijske narave vhoda in analognega pomnenja podatkov v objektu samem, se morajo vhodni podatki periodično obnavljati. Na ta način se ohranja določeno izhodno stanje ali kratko rečeno izhod objekta.

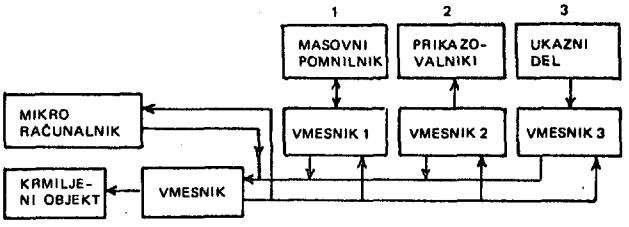
Izhod, ki je različen od nič, ne moremo dobiti v trenutku, ampak le-ta postopoma narašča od vrednosti 0. Narašča torej iz stanja, ko imajo vsi analogni izhodi istočasno vrednost 0, do stanja, ko ima vsak od analognih izhodov določeno vrednost. Čas, v katerem naraste jakost vseh analognih izhodov od nič do predpisanih vrednosti, imenujemo čas naraščanja izhoda. Čas, v katerem pa preide določeno izhodno stanje v stanje 0, imenujemo čas padanja izhoda. Izhodna stanja si slede zaporedoma tako, da istočasno, ko začne vrednost prisotnega izhoda upadati, začne novi izhod naraščati. Navadno je čas padanja starega izhoda enak času naraščanja novega izhoda. Prehod, pri katerem sta časa enaka, imenujemo simetrični prehod (CRS) iz enega v drug izhod. Kadar je prehod avtomatičen, je le-ta linearen, nelinearen pa praktično vedno, kadar ga upravljamo ročno s pomočjo potenciometrov.

Avtomatska prehoda: simetrični prehod med izhodom 1 in 2, ter asimetrični prehod med izhodoma 2 in 3, prikazuje slika 1. Pri prvem prehodu je $\Delta\varphi = 0$, pri drugem pa $\Delta\varphi \neq 0$.



Slika 1.

Nalogo krmiljenja izhodov objekta lahko razdelimo v dva dela. Prvi del predstavlja določanje izhoda, to je nastavljanje posameznih analognih izhodov na določeno vrednost, določanje vrstnega reda izhodov, ter časov naraščanja in padanja le-teh. Drugi del predstavlja avtomatično ali ročno krmiljenje izhodov v določenem vrstnem redu, izvajanje prehodov, to je naraščanje in padanje izhodov, ter mešanje več izhodov. Napravo, ki je namenjena reševanju teh nalog, lahko prikažemo s posplošeno shemo na sliki 2.



Slika 2.

Vidimo, da v našem primeru periferne naprave mikro računalnika sestavljajo: krmiljeni objekt, dodatni masovni pomnilnik, prikazovalniki (indikatorji) in ukazni del naprave. Iz slike je razvidno, da te naprave glede na smer komuniciranja z mikro računalnikom pripadajo vsem trem možnim tipom. Probleme komuniciranja teh enot prištevamo k bistvenim problemom krmiljenja objekta v realnem času. V naslednjem poglavju bomo zato z vidika organizacije tej problematiki posvetili nekaj več pozornosti.

3. KOMUNICIRANJE PROCESORJA S PERIFERIJO

Večina digitalnih sistemov komunicira med sistemskimi enotami tako, da na sprejeto informacijo odgovarja z ustreznim izhodom. Obstajajo zelo preprosti sistemi, pri katerih program v zanki periodično testira vhod, medtem ko ga pričakuje (npr. pritisk na tipko). Procesor med odtipavanjem vhodov ne more opravljati nobeno drugo koristno delo. Taka tehnika je zato potratna. Vendar obstajajo specifični primeri, kjer je opravičljiva. Druga tehnika, to je tehnika programske prekinitve, omogoča vhodno/izhodnim ali drugim sistemskim enotam, da prekinjajo glavni program le, kadar je to potrebno. Prekinitev je možna, če je procesor na to pripravljen. Prav tako, procesor more komunicirati z ostalimi enotami le, če so tudi te pripravljene na prenos informacij.

Programska prekinitev (INT) povzroči, da se trenutna adresa v programskem števniku vloži v sklad, medtem ko se v programski števnik vnese specifična lokacija. Pred tem se tekoča instrukcija še normalno izvrši. Nova lokacija se imenuje lokacija programske prekinitve in običajno vsebuje brezpogojno razvejitev na servisno rutino programske prekinitve (INSR). Ta rutina izvrši želeno nalogo. Navadno je to vnašanje ali izpisovanje podatkov in omenjeni rutini ustrezno procesiranje. Po opravljeni servisni rutini se z instrukcijo RETURN nadaljuje izvajanje glavnega programa. Tako je poslednja instrukcija vsake INTSR vedno RETURN.

Pri rabi INT pa nastopa vrsta problemov. INTSR mora biti taka, da se vsi parametri glavnega programa, ki so pomembni za nadaljno izvajanje glavnega opravila, ohranijo. Zato mora INTSR najpreje shraniti vsebine vseh uporabljenih registrov, vključno z zastavicami (kot so prenosi, predznaki, parnost, ničle in dr.). Na koncu INTSR se vsa stanja registrov in zastavice zapišejo v prvotne lokacije, še predno se prične izvajanje glavnega programa. Raba sklada "prvi v - zadnji iz" (push-down stack) je v ta namen idealna.

Operacije reševanja vsebin posameznih registrov niso potrebne, če so uporabljeni registri v INTSR za to posebej rezervirani in neuporabljeni v drugih delih programa. Operacije v zvezi s shranjevanjem zastavic pa so še vedno potrebne, ker je program lahko prekinjen med aritmetično operacijo. Razvejitve v programu aritmetične operacije so namreč odvisne od pogojev, ki se generirajo med izvajanjem operacij. Enaki problemi morejo nastopati tudi pri rabi subrutin. Ker pa se subrutine ne kličejo naključno, je odstranjevanje možnih tovrstnih napak veliko lažje.

Iz gornjega sledi, da moramo INT programirati zelo skrbno. Morebitna napaka v programu se namreč lahko pokaže le pri določeni lokaciji in pri določenih kombinacijah podatkov. Taki primeri so včasih možni le enkrat na mesec ali celo na leto. Če, na primer, pozabimo shraniti in ponovno restavrirati zastavico, se napaka pokaže, kadar nastopi INT med aritmetično operacijo. Tedaj namreč zavisi pogojna razvejitev od trenutnega rezultata operacije. Kljub temu pa ostane taka napaka prekrita, če je stanje (prenosov, ničel, itd.) po zadnji aritmetični operaciji isto, kot je bilo ob prekinitvi glavnega programa. Problem obstaja tudi v tem, da so simptomi takšnih napak vsakokrat različni. Zato je skrbno pisanje prekinitvenih programov edina rešitev.

Pri programiranju programskih prekinitev so možne tudi druge napake. Če INTSR spremeni pomnilniško lokacijo, ki jo uporablja tudi glavni program, se lahko le-ta povrne na začetek. Pri možnem INT v programu ne smemo uporabiti zakasnitveno zanko. Sicer pa, če jo uporabimo, se čas izvrševanja INT rutine doda programirani zakasnitvi. Včasih pa se INT v zakasnitveni zanki mora dopuščati. Tedaj obstaja rešitev v štetju impulzov zunanje ure.

Pri krmiljenju časovno odvisnih operacij v sistemu se pogosto uporablja programska prekinitev z uro v realnem času (Real-Time Clock Interrupt ali RTCINT). Na primer, INT moremo generirati s frekvenco 50 Hz, to je vsakih 20 msek. Rutina INT pa lahko med štetjem impulzov RTC hrani kritične informacije na določenih pomnilniških lokacijah ali registrih toliko časa, kolikor je potrebno za izvajanje posameznih operacij med programsko prekinitvijo. Če s posebno subrutino shranimo trenutno stanje števnika RTC impulzov, moremo kasneje določiti čas izvajanja operacije. Določimo ga preprosto z odštevanjem shranjenega časa od časa, ki je določen s koncem operacije. Na ta način lahko v glavnem programu upoštevamo izgubo časa, ki jo doprinese vsakokratna prekinitvena rutina.

Včasih se morajo nekatere operacije izvrševati periodično na vsak urin INT ali morda na vsakih deset urinih prekinitvenih impulzov. Rutina RTC INT shranjuje štete impulze in pravočasno, periodično opravlja določena opravila. Sistem pa lahko zahteva tudi informacijo o absolutnem času in periodično dostavljanje podatkov. Navadno je frekvenca omrežja dovolj natančna in visoka, da je takšen tip ure primeren tudi za štetje daljših period.

Drug problem programske prekinitve se pojavlja pri določenih operacijah, ki se morajo izvršiti neprekinjeno z veliko hitrostjo, čim se enkrat že pričnо izvajati. Na primer, če zapisujemo na magnetni trak ali disk, bi bile posledice programske prekinitve med takšnim izhodom nepopravljive. Problem rešimo s posebnim bistabilnim elementom (interrupt enable flip-flop), ki omogoči INT. Navadno je to zasta-

vica (flag). Pred operacijo oziroma delom programa, ki ne sme biti prekinjen, vstavimo posebno instrukcijo. Ta prepreči programsko prekinitev dokler takšna kritična operacija ni izvršena, nakar se prekinitev ponovno omogoči.

Bolj kompleksni sistemi uporabljajo prioritetne programske prekinitve. Najbolj kritičnim opravilom dajemo najvišjo prioriteto. Kadar se pojavi zahteva po takšnem opravilu, more le-ta prekiniti izvajanje rutine INT z nižjo prioriteto. Hkrati se med tem onemogoči kakršenkoli INT z nižjo prioriteto. Ta se ponovno omogoči, kadar je rutina INT z višjo prioriteto končana.

S prioritetnimi prekinitvami lahko mehaniziramo vhod in izhod vmesnikov glavnega pomnilnika. Glavni program upravlja prenos vhodnih podatkov iz polja vmesniških registrov v glavni pomnilnik. Zapis v vmesnik poteka avtomatično z rutino INT vsakokrat, ko se le-ta pojavi. Oba procesa: vpis in izpis iz vmesnika lahko potekata neodvisno drug od drugega. Posebni registri ali pomnilniške lokacije hranijo vsakokratno naslednjo adreso informacije, ki se zapisuje ali čita iz vmesnika. Podobno lahko izhodne prekinitvene rutine čitajo pomniški vmesnik neodvisno od glavnega programa, kadarkoli je tudi izhodna naprava pripravljena na takšno opravilo.

Sposobnosti programskih prekinitev so pri posameznih mikro procesorjih zelo različne. Pri INTEL 8008 in 8080 je potreben dodaten, zunanji hardware, ki omogoča INT. Dasi obstaja poseben prekinitveni vhod, ki se v primeru uporabe po stanju HALT ponovno restavrira, ne omogoča programsko prekinitev v pravem pomenu besede. Med stanjem, ki je določeno s tem vhodom, namreč ni možno nikakršno procesiranje. Resnični INT generiramo pri INTEL-ovem procesorju na tak način, da najpreje sinhroniziramo signal zahteve za prekinitev oziroma z uro procesorja tako, da je prekinitev možna le ob pravem času. Pri naslednjem dostavnem ciklu instrukcije onemogočimo delovanje pomnilnika in pošljemo na podatkovno vodilo instrukcijo RESTART (00AAA101). Ta vstavi (push) trenutno vsebino programskega števnika v sklad in preide na tekočo prekinitveno adreso: AAA000. Dočim mora biti INT pri 8008 omogočen od zunaj z INTEN (interrupt enable), je le-ta v 8080 že vključen. Za prioritetne prekinitvene verige, posamezne dvojčke, ki omogočajo INT in za identifikacijo vira prioritetnih prekinitev moramo poskrbeti preko I/O kanala od zunaj. V prekinitvenih sposobnostih je INTEL 8008 popolnoma omejen, saj ne omogoča shranitev in ponovno restavriranje zaznambnih bitov (zastavic). INTEL 8080 pa že razpolaga z instrukcijami PUSH A in POP A, skupaj z instrukcijami za manipuliranje z zastavicami.

IMP-16C ima večje prekinitvene sposobnosti. Te dosega z avtomatičnim vstavljanjem vsebine programskega števnika v sklad, razvejanjem na lokacijo 1 in preprečevanjem ponovnega INT, ko je le-ta že prisoten. Šestnajst statusnih zastavic (ki se lahko avtomatično vstavljajo in odvzemajo iz sklada) je namenjenih za to, da omogočijo programsko prekinitev (ali druga opravila za bitni izhod). Tudi v tem primeru se mora preko I/O kanalov identificirati napravo, ki povzroči INT, pri pogoju, da zahtevek po prekinitvi izvira od ene same naprave. Instrukcija RETURN FORM INTERRUPT prepiše povratno adreso (in kontrolno polje) iz sklada v programski števnik in omogoči prekinitve avtomatično.

Večji računalniki uporabljajo načine, ki še povečujejo učinkovitost obravnavanja prekinitev. Dosežejo jo z rabo instrukcij, ki omogočajo avtomatično razvejitev na različne lokacije. Te lokacije zavise od posameznih naprav, ki povzroče INT. Včasih uporabljajo tudi posebne instrukcije, ki shranijo ter na ta način rešijo in ponovno restavrirajo stanja vseh pomembnih registrov. Nastopajo primeri, ko lahko postane prekinitveni čas (interrupt response time) kritičen, ker morajo biti nekatere naprave dovolj hitro servisirane (na primer, sprejeti podatke) za tem, ko povzroče INT. Tako je pri nekaterih mikroprocesorjih čas, ki ga porabijo za iskanje prekinitvenega vira in reševanje stanj posameznih registrov za nekatere vhodno-izhodne naprave še vedno prevelik.

V našem primeru pripada k perifernim napravam prvega tipa kaseta (ali disk). Ta predstavlja dodatni, masovni pomnilnik k tako imenovanem delovnem pomnilniku mikro računalnika. Slednji je sestavljen iz petih strani pomnilnika RAM, ki sestavlja vmesnik krmiljenega objekta. Zato bomo nadalje govorili o petih delovnih pomnilnikih, ki lahko istočasno hranijo podatke o petih izhodih objekta. Podatki o časih tn in tp teh delovnih strani so shranjeni v posebnih registrih, medtem ko so na magnetnem traku pridruženi ostalim podatkom posameznih izhodov. Komuniciranje mikro računalnika s periferno napravo takšnega tipa je običajno realizirano s programsko prekinitvijo. Ta je podrobneje obravnavana na primeru v delu [1].

Periferne naprave tipa 2 so prikazovalniki. To so lahko signalne lučke, svetleče diode ali katodna cev. K takšnim napravam prištevamo tudi dajalnike zvočnih signalov. Prikazovalne naprave krmilimo preko vmesnikov. Bistveni elementi le-teh so pomnilniki, preko katerih aktiviramo svetlobni ali zvočni signal z mikro računalnikom. Na enak način ga tudi prekinemo. Med drugim so prikazovalniki pogosto pomožni elementi ukaznega dela. Take uporabljamo tudi v našem primeru in jih bomo podrobno opisali v naslednjem poglavju.

Naprave tipa 3 so vhodne naprave, ki so namenjene ročnemu krmiljenju sistema. Te predstavljajo ukazni del naprave in jih sestavljajo tipke, stikala in potenciometri. Organizacija komuniciranja teh naprav z mikro računalnikom je lahko preprosta s testiranjem posebnih dvojčkov (skip flip-flop) ali s programsko prekinitvijo. V naslednjem poglavju se bomo srečali s preprostejšo organizacijo ukazne enote, kakršna je uporabljena v našem primeru.

4. ORGANIZACIJA NAPRAVE

Opisali bomo organizacijo enot mikro računalnika in perifernih naprav. Med prvimi bomo obravnavali centralno procesno enoto, krmilni pomnilnik in krmilnik vhodno-izhodnih enot, ojačevalnik izhodnih in multipleks vhodnih vodil, pomnilnika tipa PROM in RAM, adresno, dvosmerno izhodno vodilo ter podatkovno vodilo pomnilnika. Med drugimi enotami pa bomo obravnavali ukazno enoto s prikazovalniki, masovni pomnilnik in krmiljeni objekt.

4.1. Mikro računalnik INTEL 8080

Računalnik sestavlja pet enot:

1. Centralna procesna enota (CPU) je osembitni paralelni procesor s šestimi 8-bitnimi delovnimi registri: B(0-7), C(0-7), D(0-7), E(0-7), H(0-7), L(0-7), 8-bitnim akumulatorjem A(0-7) in 16-bitnim kazalcem SP(0-15) vrha sklada. Poleg teh imamo še ukazni register UR(0-7), programski števnik PC(0-15) ter adresni AR(0-15) in podatkovni register PRP(0-7). Poleg večbitnih registrov imamo še enobitne registre zastavic prenosa CARRY (ali kraje CY), ničle ZERO in parnosti PARITY. Med temi registri lahko definiramo še podregister AR(SP) = AR(0-7) adrese zlogov in register AR(ZG) = AR(0-15) adrese strani. Stik registrov B in C je delovni register BC(0-15) = = B-C, ki je tudi register za naslavljanje. Podobno sta definirana tudi registra DE in HL, medtem ko slednji rabi tudi za seštevanje.

Centralno enoto krmili dvofazna ura s frekvenco 2 MHz. Povprečni čas, v katerem se izvrši en ukaz, je od 5,5 μsek do 7 μsek.

2. Krmilnik pomnilnika in vhodno-izhodnih enot vsebuje dvofazno uro frekvence 2 MHz in pomnilnik,

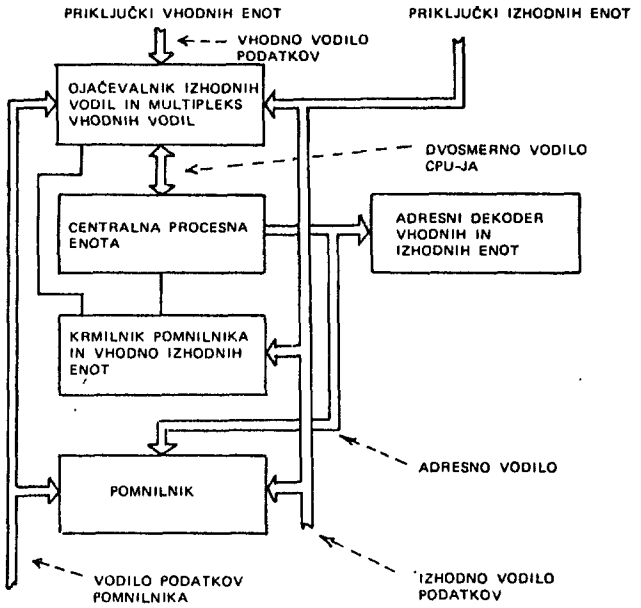
ki hrani informacije o stanju centralne enote. Ta informacija določa, kateri podatki se vnesejo v CPU, oziroma, če je CPU izvor podatkov, kam se le-ti prenesejo. V krmilnem delu je še logično vezje, ki vsklajuje hitrost CPU s hitrostjo pomnilnika in hitrostjo vmesnikov zunanjih enot.

3. Ojačevalnik izhodnih vodil in multipleks vhodnih vodil. Multipleks vhodnih vodil prepušča podatke v CPU iz zunanjih enot ali iz pomnilnika.

4. Pomnilnik tipa PROM je M(AR) = M(0-17 377$_{(8)}$, 0-7), torej kapacitete 4K osembitnih zlogov. Vanj se vpiše program.

5. Pomnilnik tipa RAM je M(AR) = M(20 000 - 27 377$_{(8)}$, 0-7) je kapacitete 1K osembitnih zlogov. Vanj se vpisujejo podatki, ki se med izvajanjem programa spreminjajo.

Povezave med temi elementi podaja slika 3.



Slika 3.

Vodila, ki povezujejo računalniške enote so:

1. Adresno vodilo, ki ima 16 bitov. Izvira iz AR v centralni enoti. Običajno se po tem vodilu pri pomnilniških ukazih prenašajo adrese posameznih zlogov, pri vhodno-izhodnih ukazih pa adrese, ki pripadajo vhodnim oziroma izhodnim enotam.

2. Dvosmerno vodilo centralne enote prenaša podatke iz CPU na izhodna vodila in podatke iz vhodnega vodila ali vodila pomnilnika v CPU.

3. Izhodno vodilo prenaša podatke iz centralne enote v pomnilnik in v izhodne enote. Po njem se prenaša tudi podatek o stanju centralne enote v register krmilnika.

4. Podatkovno vodilo pomnilnika prenaša podatke iz pomnilniške lokacije, katere adresa se nahaja na adresnem vodilu, v CPU.

5. Vhodno vodilo prenaša podatke iz vhodnih enot v CPU.

4.2. Periferne naprave

Poleg ukazne enote s prikazovalnikl in masovnega pomnilnika lahko med periferne naprave prištevamo tudi krmiljeni objekt. Slednjemu smo posvetili že nekaj pozornosti. Na tem mestu bomo dodali le še nekatere nadrobnosti, ki se nanašajo na vmesnik krmiljenega objekta. Te so še posebej pomembne za razumevanje časovnega generatorja.

1. Vmesnik krmiljenega objekta sestavlja pet delovnih strani: DSi(PAR) = DSi(6i 000$_{(8)}$ - 6i 377$_{(8)}$, 0-7), kjer je i$\in${1,...,5} in PAR = PAR(0-15) = P - AR(ZG), adresni register petih delovnih strani. V registru P je spodnji, v registru AR(ZG) pa gornji del adrese. Delovne strani se razlikujejo od strani računalniškega pomnilnika po tem, da so izhodi delovnih strani vezani le na digitalno-analogni pretvornik (DAP). Zaradi tega je možno v delovni strani podatke le vpisovati, čitati pa jih ni mogoče. Vsaki od prvih štirih strani pripada po en potenciometer. Poleg teh potenciometrov X1, X2, X3 in X4 obstaja še glavni potenciometer Z, ki paralelno napaja vse štiri potenciometre. Delovnim stranem pripadajoči potenciometri rabijo za prehajanje izhoda krmiljenega objekta iz enega stanja v drugo, medtem ko omogoča potenciometer Z nastavljanje posameznih analognih izhodov izhodnega stanja objekta. Vsi našteti potenciometri imajo servosistem. Vmesnik vsebuje poleg že omenjenih petih digitalno-analognih pretvornikov še enoto za analogno množenje in seštevanje, multipleksor in dva demultipleksorja. Ker te enote nimajo pomembnejše vloge pri generiranju časovne funkcije, nadalje ne bodo deležni naše pozornosti.

2. Masovni pomnilnik je lahko različnega tipa. Izbira zavisi od zahtev krmiljenega objekta. V našem primeru je uporabljen magnetni trak - kaseta, ki ustreza po ceni, hitrosti in zanesljivosti. Problemi komuniciranja te naprave z mikro procesorjem, ki zajemajo zlasti probleme kasetnega vmesnika in vključevanja kasetnih programov v glavni program, so bile že podrobneje opisani v delu [1].

3. Ukazno enoto sestavljajo stikala, signalne lučke in prikazovalniki. Prikazovalniki so običajno katodne cevi ali polvodniške svetleče diode. V naši napravi so uporabljene slednje z dekoderjem (HP-5082-7300).

Poleg stikala POWER (ON, OFF) z dvema položajema za vklop in izklop naprave nahajamo tipko kot enopoložajno stikalo za generiranje impulza, ki postavi programski števnik v začetno stanje 0, in polja stikal, s katerimi določamo posamezna opravila.

Polje stikal lahko zapišemo kot stik stikal. Eno takšnih polj je stik:

Casswitch, SSS(AV,R,0-4) = SSZ-SSX1-SSX2-SSX3-SSX4

Stikalo SSX podaja stanje potenciometra servosistema X, ki pripada X-ti delovni strani.

Polje numeričnih tipk NT sestavljajo tipke, s katerim zapišemo v prikazovalnik PNT numeričnih tipk števila od 0-9:

Casswitch, POLNT(ON,0-9) = NT1-NT2-NT3-NT4-NT5-NT6-NT7-NT8-NT9

Stanje tipk enega polja lahko dobimo v akumulator z enim ukazom. Zato pravimo, da tvorijo takšne tipke polje tipk, ki ga določimo kot stik. Naj omenimo še nekatera takšna polja:

Casswitch, POLB(ON,0-6) = SCN-ŽAR-TN-TP-BNT-SSCN-BFT

Polje tipk B tvorijo torej tipke: SCN(ON), preko katere zahtevamo vnašanje izhoda iz perifernega (masovnega) pomnilnika v mikro računalnik in pove, da je število v PNT številka izhodnega stanja, ki ga želimo. S tipko ŽAR(ON) zahtevamo nastavljanje analognega izhoda, katerega številka je zapisana v PNT. TN(ON) oziroma TP(ON) pove, da je število v PNT čas naraščanja oziroma padanja. BNT(ON) zbriše število PNT, BFT(ON) pa vse funkcijske tipke. SSCN(ON) pa zahteva vnašanje izhodnega stanja objekta.

Številka tega stanja je v PNT in se zapiše v delovno stran DS5.

Casswitch, POLD(ON,0-5) = CRS - ZP1 - ZP2 - ZP3 - ZP4 - BS

CRS(ON) zahteva simetričen prehod, medtem ko zahtevajo stikala ZP(ON) prehod na izhodno stanje objekta, ki je zapisano v določeni delovni strani DS1 do DS4. Na tipko BS(ON) pritisnemo, kadar hočemo sprostiti eno od DS za vnašanje novega izhoda.

Poleg omenjenih polj imamo še polja tipk PP1(ON) do PP5(ON), ki adresirajo ustrezne delovne strani. V posebna polja so vključene tudi tipke drugih opravil, ki jih v članku ne bomo obravnavali. Ta opravila namreč ne dotikajo problematike, kateri je članek posvečen, ali pa so bila že obravnavana v delu [1]. Omenim naj le še tipko IZV(ON), ki zahteva izvršitev opravila, ki je določeno s poprej pritisnjenimi tipkami.

Večini zgoraj opisanih tipk pripadajo ustrezne signalne lučke. Tako pripišemo polju B funkcijskih tipk polje signalnih lučk LUCB :

Caslight, LUCB(ON,OFF,0-4) = LSCN - LŽAR - LTN - LTP - - LSSCN

Za lučke enega polja je značilno, da so pri prižiganju (npr. LTN(ON) in ugašanju (npr. LTN(OFF) dostopne vse hkrati.

Prikazovalnik popišemo kot stik signalnih lučk z devetimi različnimi signali:

Caslight, PNT(0,1,2,3,4,5,6,7,8,9,0-2)

PNT je torej stik treh devetsignalnih lučk za vpis tromestnega števila. Delovnim stranem DS1 do DS4 pripadajo vsaki po dva prikazovalnika. Zato jih popišemo kot dve polji, katerih elementi so razvrstitve prikazovalnikov.

Array - Caslight, PTDS(0,1,2,3,4,5,6,7,8,9,0-2, 0-3)

Stavek popisuje polje, ki ga tvorijo štirje prikazovalniki, kamor vpisujemo čase prehodov izhodnih stanj, zapisane v pripadajoči delovni strani.

Array - Caslight, PSSDS (0,1,2,3,4,5,6,7,8,9,0-2, 0-3)

To je polje, kamor vpisujemo številko izhodnega stanja, ki je trenutno zapisano v pripadajoči delovni strani.

4.3. Nekatere povezave med elementi sistema

Vsakemu polju tipk, ki ima največ sedem tipk, pripada en enobitni register. Imenovali ga bomo SKIPpl, kjer je pl oznaka polja (B,C,...). Ta dvojček zaseda osmi bit zloga. Vsakemu polju pripada ustrezno polje signalnih lučk, polju numeričnih tipk pa tromestni, desetvrednostni prikazovalnik za vsako številčno mesto.

Stikala, signalne lučke, prikazovalniki in ADP so povezani z akumulatorjem preko vhodnih in izhodnih vodil podatkov. Prenos iz vhodne enote preko vhodnega vodila v akumulator dosežemo z ukazom INPa, kjer je a adresa vhodne enote, ki se vstavi v AR. Pri tem ukazu se pojavi signal INP = 1. Prenos iz akumulatorja do izhodne enote se izvrši z ukazom OUTa, kjer je a adresa izhodne enote, ki se pojavi v AR. Pri tem ukazu se generira signal OUT = 1. Te prenose opisujejo sledeči stavki:

IF(INP=1 AND AR(0-7) = $25_{(8)}$) THEN(IF(POLB(I) = ON) THEN(A(I) ← 1) ELSE(A(I) ← 0)), A(7) ← SKIPB) ,

kjer je $25_{(8)}$ adresa polja B in I=0,1,...,6. Na enak način se z naslavljanjem drugih polj prenesejo stanja ostalih tipk. V AR(8-15) je pri INP in OUT ukazih enaka vsebina kot v AR(0-7).

Ker so vhodi v prikazovalnike kodirani v BCD kodu, se stanje tipk numerične tastature prenaša v akumulator takole:

IF(INP=1 AND AR(0-7) = $20_{(8)}$) THEN(IF(PLNT(I) = ON) THEN(A(0-3) ← $1_{(2)}$, A(4-7) ← 0) ELSE(A(I) ← 0)), A(7) ← SKIPNT),

kjer je $20_{(8)}$ adresa polja NT in $I_{(10)}$ = 0,1,...,9.

Stanje stikal servosistemov v akumulator preko dekoderja poteka takole:

IF(INP=1 AND AR(0-7) = $61_{(8)}$) THEN(IF(SSX(I) = AV) THEN(A(I) ← 1) ELSE(A(I) ← 0)), A(5-7) ← 0) ,

kjer je I=Z,1,2,3,4 in $61_{(8)}$ adresa polja stikal servosistemov.

Brisanje enobitnih registrov SKIPpl dosežemo z ukazi OUT. Na primer za SKIPNT z adreso $21_{(8)}$ velja:

IF(OUT=1 AND AR(0-7) = $21_{(8)}$ THEN(SKIPNT ← 0).

Lučke v tipkah prižigamo in ugašamo tako, da vsebino akumulatorja prenesemo preko vodil izhodnih podatkov do pomnilnika lučk. Sledeči stavek opisuje prižiganje in ugašanje stika lučk LUCB.

IF(OUT=1 AND AR(0-7) = $27_{(8)}$) THEN(IF(A(I) = 1) THEN(LUCB(J) ← ON) ELSE(LUCB(J) ← OFF)) .

kjer je (I,J) = (0,0), (1,1), (2,2), (3,3), (5,4) in vrednost para (I,J) zavisi od razmestitve tipk in njim ustreznih lučk po bitih.

Kot primer zapisovanja v prikazovalnik vzemimo prikazovalnik enic numeričnih tipk, ki ima adreso $22_{(8)}$.

IF(OUT=1 AND AR(0-7) = $22_{(8)}$) THEN(PNT(0) ← A(0-3))

Analogno velja za ostale prikazovalnike. Pri tem upoštevamo, da podregister A(0-3) hrani enice, A(4-7) desetice, podregister A(0-3) naslednjega zloga pa stotice. Pri prikazovalniku časa so v prvem zlogu zakodirane sekunde, v drugem zlogu na prvih štirih bitih pa minute.

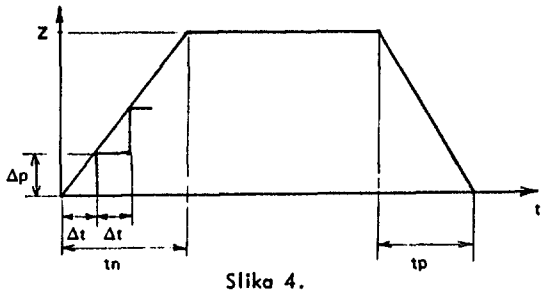
5. IZRAČUN ČASOVNE FUNKCIJE

Časovni generator generira binarno zakodirane amplitude do vnaprej predpisane vrednosti Z, kot linearne funkcije časa. Avtomatsko prehajanje izhoda krmiljenega objekta iz enega stanja v drugo je izvedeno tako, da se pri naraščanju potenciometrov v "enakih" časovnih intervalih Δt prišteva k vsebini lokacije, ki krmili servosistem potenciometra, prirastek Δp. To traja, dokler vsebina lokacije ne doseže vrednost potenciometra Z. Pri tem doseže krmiljeni potenciometer skrajno zgornjo lego. Kadar vrednost potenciometra pada, je postopek enak, le da se Δp odšteva od vsebine lokacije, dokler ni njena vrednost nič. Na ta način je realizirana stopničasta funkcija, ki je približek linearne funkcije. Čas prehajanja je določen s časom naraščanja pri naraščanju scene in s časom padanja pri upadanju scene. Časovni interval, v katerem se izvrši eno prištevanje oziroma odštevanje, je enak povprečnemu času, v katerem se enkrat izvede program ZANK. Ta znaša približno 20 msek. Prirastek Δp izračuna podprogram CZP pred začetkom prehoda na sledeči način:

$$p = \frac{Z}{tn}\,\Delta t\ , \quad \text{oziroma} \quad p = \frac{Z}{tp}\,\Delta t\ , \qquad (1,2)$$

če gre za naraščanje oziroma padanje potenciometra (sl.4). Časa tn in tp določa operater. Podatka vnese za vsak izhod posebej preko tastature ukaznega dela sistema.
Ko CZP izračuna Δp uredi še spremenljivko $M(20117_{(8)}, 0\text{-}7) = ZP(0\text{-}7)$. Posamezni biti v ZP(1-4) pripadajo zaporedoma potenciometroma X1,...,X4. V tiste bite v ZP(1-4), ki pripadajo potenciometrom, za katere je CZP izračunala Δp, se zapišejo enice. Prehajanje potenciometrov pa krmili rutina TSCG.

Kadar hoče operater izvesti simetričen prehod dveh potenciometrov (eden narašča, drugi upada in je tn=tp), potem pritisne tipko CRS in še tisti dve tipki izmed ZP1,...,ZP4, ki pripadata omenjenima potenciometroma. S tipko IZV se spro-



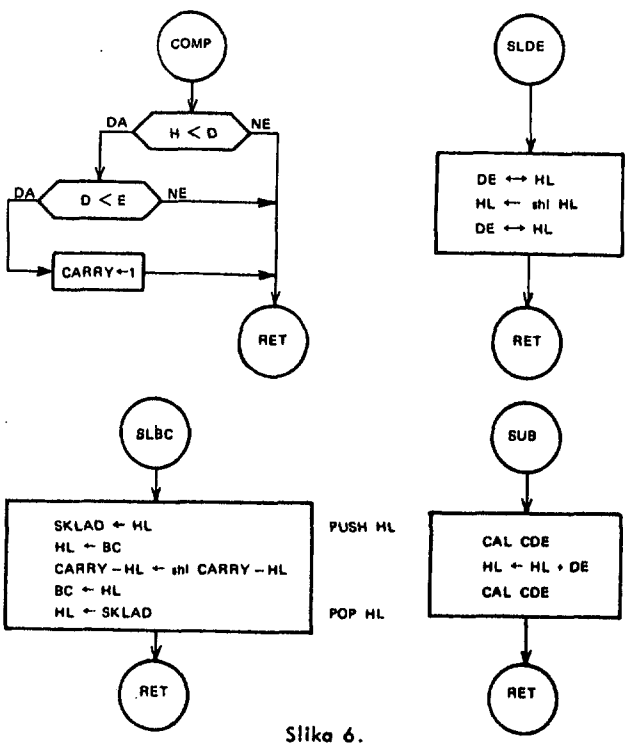
Slika 4.

ži IZŠ in s tem CZP. Ta ugotovi, da je bila pritisnjena tipka CRS, zato pregleda, kateri dve tipki izmed ZP1,...,ZP4 sta bili pritisnjeni in lego obeh pripadajočih potenciometrov. Pri tem mora biti eden v skrajni zgornji, drugi pa v skrajni spodnji legi. CZP iz časa naraščanja, ki pripada potenciometru v spodnji legi, izračuna Δp. Za tolikšno vrednost po vsakim Δt lega potenciometra korakoma narašča, drugega pa pada. CZP nato postavi še ustrezne bite v ZP(1-4) na 1. INIT nastavi na začetku ZP = 0.

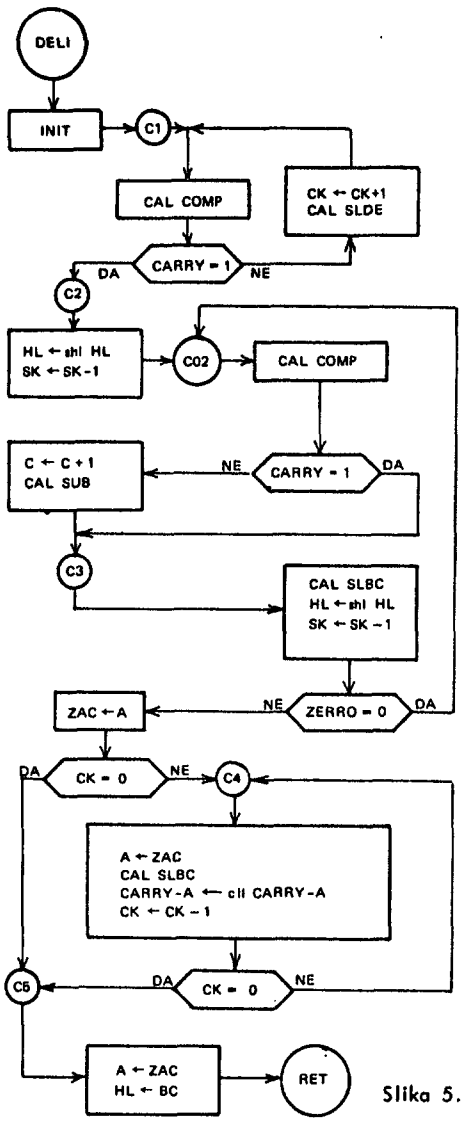
5.1. Podprogrami za generiranje časovne funkcije

Algoritem je osnovan na enačbah 1 in 2. Realiziran je s programom CZP, ki uporablja pri izračunu p podprograma za deljenje in množenje. Podprogram DELI deli vsebino na lokaciji Z z vsebino registrov HL. Registra DE hranita deljitelja, HL deljenca, BC pa delne kvociente. Lokacija SK rabi kot števnik pomikov deljenca, CK pa kot števnik celih mest kvocienta. Rezultat se nahaja: v A celi del, ki se prepiše iz pomožnega registra ZAC, v HL pa ulomljeni del kvocienta, ki se prepiše iz BC. Podprogram DELI poteka po diagramu na sliki 5. Vanj vstopajo rutine INIT, COMP, SLDE, SLBC in SUB. Te so podane na sliki 8. V DELI se rutina C1 izvaja, dokler ni HL manjši od DE, cikl v rutini C02 se izvaja 16-krat, rutina C4 pa 4-krat. Deljenje po programu DELI se na mikro procesorju INTEL 8080 izvrši v 5 do 8 nsek.

Rutina COMP primerja vsebini registrov HL in DE. Če je vsebina v HL manjša od vsebine v DE, potem postavi zastavico CARRY. Rutina SLBC pomakne vsebino v BC za en bit v levo. Na najnižji bit (najbolj desni bit) zavzame vrednost 0, najvišji bit pa se pomakne v CARRY. Rutina SCDE



Slika 6.



Slika 5.

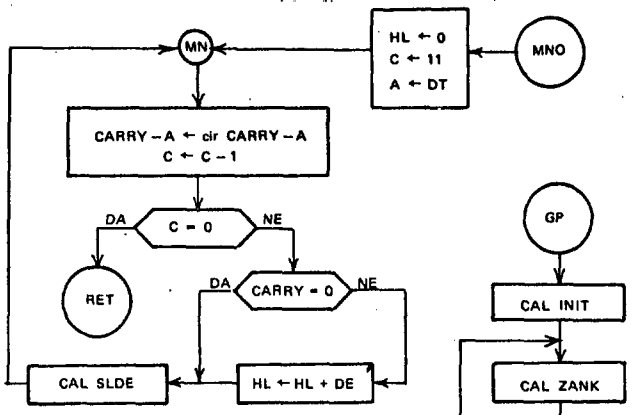
pomakne vsebino v DE za en bit v levo. Najnižji bit prevzame 0, najvišji pa se pomakne v CARRY. Rutina SUB odšteje vsebino v DE od vsebine v HL. Vsebina v DE ostane nespremenjena. Podrutina CDE izračuna k vsebini v DE dvojiški komplement in ga postavi v DE.

Rutina MNO množi podatek v registru DE s podatkom na lokaciji DT. Rezultat se naračuna v HL. Register C je uporabljen kot števnik pomikov množitelja. Vanj vstavimo 11. V DE naložimo množenec. Množenje se izvrši v času T_{MNO} = 676 μsek + 12,5 x število enic množitelja, kar znaša 676 do 776 μsek.

5.2. Povezava časovnega generatorja z glavnim programom

Glavni program sestavljata dva podprograma INIT in ZANK. Diagram poteka podaja slika 10. Ob vklopu naprave se programski števnik postavi na nič in začne se izvajati podprogram INIT, ki inicializira napravo. Nato se ciklično izvaja podprogram ZANK, ki teče, dokler naprave ne izklopimo. Z besedo CAL, ki ji sledi ime podprograma, so označeni klici podprogramov. Povratek v program iz klicanega podprograma je v podprogramu podan z ukazom RET.

Podprogram INIT vzpostavi začetno stanje naprave. Najprej nastavi kazalec sklada. Ker sklad narašča od višje adrese proti nižjim adresam, postavi dno sklada na najvišjo loka-

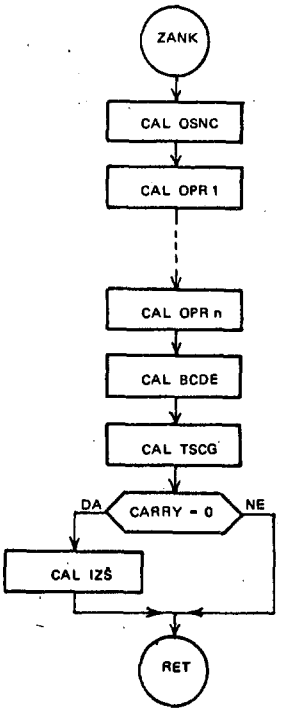


Slika 7.

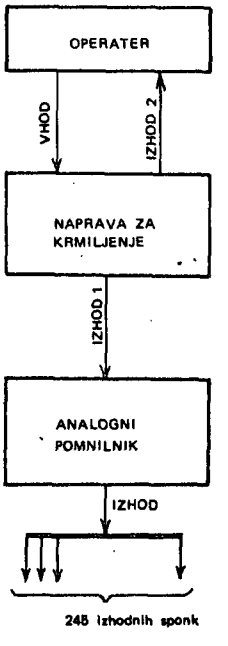
Slika 8.a

cijo: SP ← 27 377(8). Nato nastavi izhodno stanje na vrednost 0 tako, da vpiše ničle v vse delovne strani, in nastavi lokacije, ki krmilijo servosisteme potenciometrov X1, X2, X3 in X4. INIT nadalje ugasne signalne lučke v tipkalu ukaznega dela, ugasne numerične prikazovalnike, briše "skip" registre, prečita prve podatke iz masovnega pomnilnika ter nastavi začetno vrednost še nekaterim spremenljivkam, ki zasedajo en ali več zlogov v pomnilniku (RAM).

Podprogram ZANK na sliki 8b lahko v grobem razdelimo na dva dela. Razmejitev pojasnjuje slika 9. Podprogram OSNC krmili izhod 1 s tem, da donaša in obnavlja podatke o izhodnem stanju v analognem pomnilniku. To obnavljanje se mora ponoviti najkasneje v 20 msek.



Slika 8.b



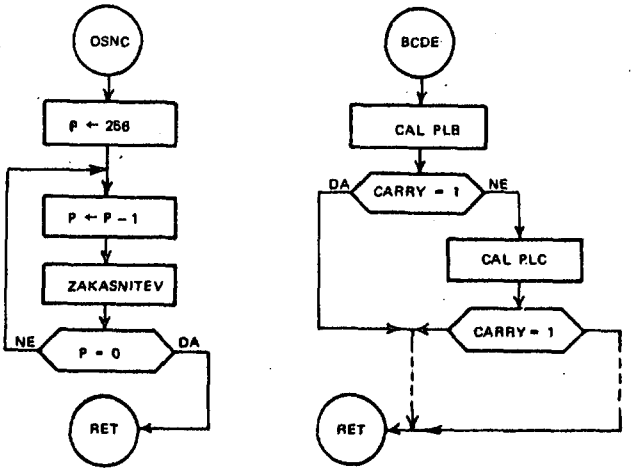
Slika 9.

OSNC traja 10 msek, zato imajo preostali podprogrami v ZANK največ 10 msek časa za svoje izvajanje. Potem mora ponovno priti na vrsto OSNC. Ti preostali podprogrami tvorijo drugi del ZANK, ki uravnava komuniciranje z operaterjem. Od operaterja sprejme ukaze (VHOD), jih razpoznava, izvršuje,ter javlja svoje stanje operaterju (IZHOD 2). Med podprograme opravil OPR1 do OPRn štejemo npr. podprogram, ki pregleduje stanje servosistemov in opravi dela v zvezi s potenciometri, podprogram, ki omogoča nastavljanje posameznih analognih izhodov, ter tiste, ki skrbe za komuniciranje z masovnim pomnilnikom (npr. kaseto, diskom, ipd.). Operater daje napravi ukaze tako, da pritisne določeno kombinacijo tipk na ukaznem delu. Ukaz zaključi s pritiskom na tipko IZV, ki pove, naj se opravilo, ki ga zahteva kombinacija tipk, izvrši. Podprogram BCDE pregleduje polja tipk ukaznega dela in si zapomni pritisnjeno kombinacijo. Ob pritisku na tipko IZV postavi 0 v CARRY, s čemer sproži izvajanje podprograma IZŠ. IZŠ prevzame kombinacijo tipk, ki jo shrani podprogram BCDE, ugotovi kakšno opravilo je zahtevano in ga izvrši. *Postavi pa tudi ustrezen signal, če se opravilo izvaja dlje* časa. Eno takih opravil, ki trajajo dlje časa je komuniciranje *s počasnimi perifernimi napravami - masovnimi pomnilniki,* drugo takšno opravilo pa je avtomatski prehod potenciometrov. TSCG pregleduje, če je prisoten signal (postavi ga IZŠ), ki zahteva prehod potenciometra. Če najde tak signal, izračuna novo lego pripadajočega potenciometra in jo javi servosistemu. Med izvajanjem takšnih počasnih opravil se podprogram ZANK večkrat izvaja.

OSNC obnavlja podatke o izhodih tako, da postavlja ciklično v register P adrese analognih izhodov med 0 in 245. Adresa posameznega izhoda mora biti prisotna v registru P vsaj 40 μsek, da se lahko obnovi analogni pomnilnik. To dosežemo z zakasnitvijo, pri kateri računalnik izvaja ukaz NOP. Diagram poteka kaže slika 10.

Podprogram BCDE sestavljajo podprogrami PLB, PLC, ..., od katerih vsak ustreza določenemu polju tipk (B,C,itd.). Ti podprogrami pregledujejo polja tipk POLB, POLC,... in ugotavljajo, če je v pregledovanem polju pritisnjena katera od tipk, potem postavijo 1 v CARRY, sicer pa 0 v CARRY, in skočijo iz podprograma BCDE. Vanj se vračajo v naslednji zanki, če je CARRY = 1. Na ta način se v enem ciklu lahko obdela največ ena pritisnjena tipka. Tipka IZV postavi 0 v CARRY.

Med podprogrami programa BCDE naj omenim le podprogram PLB, ki urejuje poleg tipk POLB tudi tipke numerične tastature PLNT in pripadajoči prikazovalnik PNT. Tipke SCN, ŽAR, TN, TP in SSCN določajo namen numerične tastature. Tako pomeni število PNT: številko izhodnega stanja, če gori lučka v tipki SCN, številko posameznega analognega izhoda, če gori lučka v tipki ŽAR, ter čas naraščanja oziroma padanja, če gori lučka v tipki TN oziroma TP. Številka v PNT pomeni številko izhodnega stanja objekta tudi v primeru, ko gori lučka v tipki SSCN. Omenjene tipke se med seboj izključujejo, kar pomeni, da lahko gori vedno le ena od signalnih lučk, ki pripada tem tipkam in to tista, ki je bila zadnja pritisnjena. Ta funkcijska tipka tudi določa namen numerične tastature. Numerično tastaturo in prikazovalnik urejamo le, kadar je določen namen numerične tastature.

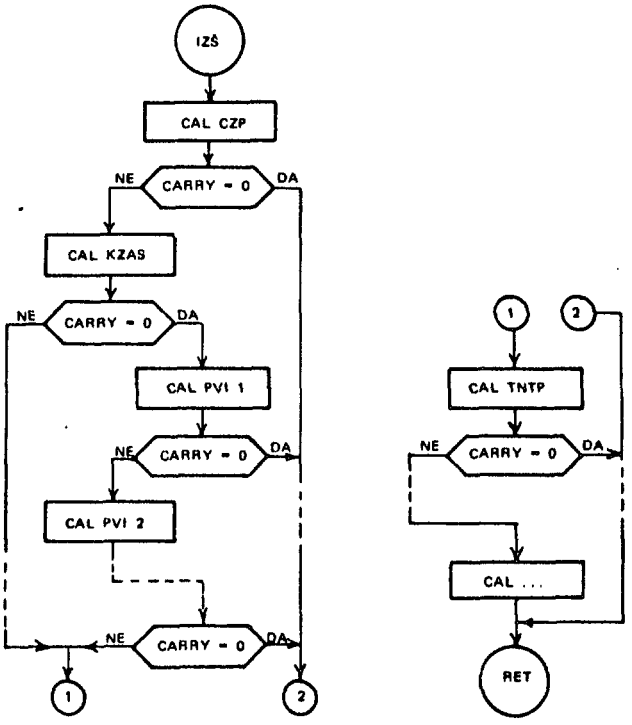


Slika 10.

Tedaj se število, ki je odtipkano na njej, vpiše v prikazovalnik in v pomnilniško lokacijo, ki je prirejena funkcijski tipki. Omenjeno lokacijo imenujemo tudi register tipke. Ti registri se brišejo s podprogramom INIT.

Vsak podprogram v IZŠ pripada enemu od opravil, ki jih zahteva operater. Med te podprograme sta vključena tudi podprograma CZP in TNTP, ki sta tesno povezana z generiranjem časovne funkcije. Vsak od podprogramov v IZŠ najprej preveri, če kombinacija tipk zahteva opravilo, kateremu podprogram pripada. Če kombinacijo razpozna, opravilo izvrši in pred vrnitvijo v GP zapiše ničle v spremenljivke, iz katerih je razbral kombinacijo tipk, ugasne lučke v odgovarjajočih tipkah, CARRY pa postavi v 1.

Podprogram KZAS pogleda v posebnem registru SKAS, če je vhodno-izhodna naprava zasedena s kakšnim opravilom. Če je, izvrši CARRY ← 1, sicer pa CARRY ← 0 in se vrne. V primeru zasedenosti naprave IZŠ ignorira zahteve za delo z njo in preskoči programe PV1. Diagram poteka IZŠ kaže slika 11.

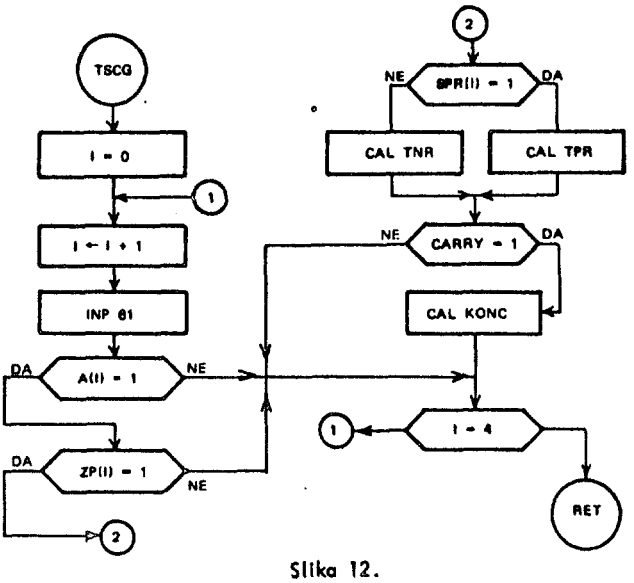


Slika 11.

Podprogram TNTP preveri, ali hoče operater predpisati čas naraščanja ali čas padanja izhoda, katerega podatki se nahajajo v eni od delovnih strani. Operater postavi to zahtevo tako, da pritisne tipko TN ali TP, nato odtipka čas prehoda na numerični tastaturi. S pritiskom na eno od tipk PP1, ..., PP5 pove, kateri strani naj pripadajo podatki o izhodu, ki določajo čas naraščanja ali čas padanja izhoda. S pritiskom na tipko IZV se sproži IZŠ in s tem TNTP. Ta razpozna pritisnjeno kombinacijo tipk in zapiše čas v ustrezen prikazovalnik, ki pripada izbrani delovni strani.

Diagram poteka TSCG prikazuje slika 12. TSCG najprej z ukazom INP 61(8) prenese stanje potenciometrov v A. Če je stanje opazovanega potenciometra ročno, TSCG pregleda naslednji potenciometer, sicer pa pogleda, če je postavljena zahteva za avtomatski prehod potenciometra. To pove ustrezni bit v registru ZP (zlog v pomnilniku RAM). Pri avtomatskem prehodu pove ustrezen bit v registru SPR (posebni lokaciji v RAM), če je potrebno naraščanje ali padanje izhoda. Temu ustrezno TSCG pokliče podprogram TNR ali TPR.

TNR izvaja naraščanje tako, da prišteje Δp k vsebini lokacije, ki krmili servosistem opazovanega potenciometra. Na



Slika 12.

enak način izvaja TPR padanje tako, da odšteva Δp. Oba podprograma pregledujeta, če je prehod končan (potenciometer je v eni od skrajnih leg). Če je, javita to tako, da izvršita CARRY ← 1. To pa sproži podprogram KONC, ki ob končanem prehodu potenciometra postavi ustrezni bit v ZP na 0. KONC postavi še 1 v določeni bit registra SPR, če je potenciometer dosegel skrajno zgornjo lego, oziroma 0, če je dosegel skrajno spodnjo lego. Podprogram TSCG se vrne v GP, ko pregleda vse potenciometre. V diagramu poteka je z I označen indeks, ki ga hranimo v registru C.

6. ZAKLJUČEK

Programiranje opisanega primera krmiljenega objekta v realnem času je potekalo v zbirnem jeziku na mikro računalniku INTEL 8080. Pri testiranju sta bila uporabljena dva računalnika. Prvi, ki je bil kasneje vgrajen v vmesnik krmiljenega objekta, je imel priključeno kot vhodno-izhodno enoto ukazni del naprave (funkcijske tipke in numerično tastaturo). Drugi mikro računalnik pa je imel kot vhodno-izhodni enoti priključen tiskalnik z luknjačem TELETYPE ASR 33 in kaseto MFE 250. Ta je omogočala hitrejše vnašanje programov, zbirnika in ODT programov. Vnašanje le-teh preko čitalnika traku ASR 33 je namreč mnogo počasnejše, ne le zaradi čitalnika samega, ampak tudi zaradi zbirnika, ki je tro-prehoden in zahteva trikratno čitanje. Programiranje bi bilo še hitrejše z rabo križnega zbirnika.

Gradnja algoritma krmiljenja objekta je potekala z navzdolnjim razvijanjem programa, kar omogoča dober pregled nad celotnim sistemom. Pri tem so bila predvsem na zgornjih nivojih v veliki meri upoštevana pravila strukturiranega programiranja. Tako programiranje in navzdolnji razvoj programa sta omogočala enostavno preizkušanje in povezovanje enot, ki tvorijo program.

Univerzalni mikro računalnik INTEL 8080 se je izkazal kot primeren za reševanje zgoraj opisanih nalog. Tam, kjer je proces dovolj počasen, je bil uporabljen način ugotavljanja sprememb s cikličnim odtipavanjem. Programska prekinitev, ki jo 8080 omogoča, je bila uporabljena le pri komuniciranju s kaseto.

7. LITERATURA

[1] B.Mihovilović, R.Murn, T.Pisanski, Z.Milavc, P.Kolbezen, Komuniciranje mikro računalnika s kaseto v realnem času, Zbornik radova Jurema 1977, svezak II, Zagreb, 1977.

[2] J.Polajnar, Krmiljenje objekta z mikro računalnikom v realnem času, Diplomsko delo, FE v Ljubljani, 1976.

Informatica št. 2 letnik 1977

a programming system for editing annotated bibliographies

j. j. dujmović

UDK 681.3.06:[02 + 002]

University of Belgrade,
Dept. of Eelectrical Engineering
11000 Beograd

ABSTRACT. A PROGRAMMING SYSTEM FOR EDITING, CLASSIFYING AND PRINTING OF SIMPLE OR ANNOTATED BIBLIOGRAPHIES AND SIMILAR TEXTS IS PRESENTED. THE SYSTEM IS PRIMARILY DESIGNED FOR THE INDIVIDUAL USE BY RESEARCHERS AND IS SIMPLE FOR MANIPULATION. IT CAN BE USED IN THE FOLLOWING APPLICATIONS - (1) EDITING AND PRINTING GENERAL PURPOSE TEXTS, (2) PRODUCING LISTS OF REFERENCES TO BE PUBLISHED AT THE END OF PAPERS, (3) PRODUCING ANNOTATED BIBLIOGRAPHIES, AND (4) PRODUCING CLASSIFIED BIBLIOGRAPHIES. ALL FORTRAN PROGRAMS AND THE CORRESPONDING USER MANUAL ARE INCLUDED IN THE PAPER.

PROGRAMSKI SISTEM ZA EDITIRANJE KOMENTARISANIH BIBLIOGRAFIJA. U RADU JE PRIKAZAN PROGRAMSKI SISTEM ZA EDITIRANJE, KLASIFIKACIJU I ŠTAMPANJE PROSTIH ILI KOMENTARISANIH BIBLIOGRAFIJA I SLIČNIH TEKSTOVA. SISTEM JE PRVENSTVENO PREDVIDJEN ZA INDIVIDUALNO KORIŠĆENJE OD STRANE ISTRAŽIVAČA I JEDNOSTAVAN JE ZA RUKOVANJE. MOŽE SE KORISTITI U SLEDEĆIM PRIMENAMA - (1) EDITIRANJE I ŠTAMPANJE PROIZVOLJNIH TEKSTOVA, (2) FORMIRANJE LISTA REFERENCI ZA PUBLIKOVANJE NA KRAJEVIMA RADOVA, (3) REALIZACIJA KOMENTARISANIH BIBLIOGRAFIJA I (4) REALIZACIJA KLASIFIKOVANIH BIBLIOGRAFIJA. U RADU SU PRILOŽENI SVI POTREBNI FORTRAN PROGRAMI I ODGOVARAJUĆI PRIRUČNIK ZA KORISNIKA.

INTRODUCTION

BIBLIOGRAPHIC RESEARCH IS REGULARLY AN IMPORTANT PART OF EACH RESEARCH EFFORT. DURING RESEARCH IN SOME FIELD, RESEARCHERS OFTEN MAINTAIN AND UPDATE COMPREHENSIVE LISTS OF REFERENCES. IN A NUMBER OF CASES IT IS NECESSARY TO SUPPLY REFERENCES WITH ABSTRACTS AND/OR COMMENTS. MOREOVER, SOMETIMES IT IS ESSENTIAL TO CLUSTER BIBLIOGRAPHIC ENTRIES FORMING CLASSES OF RELATED ITEMS, AND TO PROVIDE THE POSSIBILITY FOR AN ITEM TO BELONG SIMULTANEOUSLY TO SEVERAL CLASSES. THE AUTOMATIZATION OF THESE ACTIVITIES IS DOUBTLESSLY AN USEFUL AND TIME-SAVING JOB.

THE BASIC PROBLEM ASSOCIATED WITH THE REALIZATION OF BIBLIOGRAPHIC DATA BASE SYSTEMS DESIGNED FOR PERSONAL USE, IS TO PROPERLY MEASURE OUT THE DEGREE OF SOPHISTICATION OF SUCH SYSTEMS CONSIDERRING BOTH THE SCOPE AND COMPLEXITY. SYSTEMS FOR PERSONAL USE DIFFER CONSIDERABLY FROM THE SYSTEMS DESIGNED FOR PUBLIC USE (E.G. LIBRARY INFORMATION RETRIEVAL SYSTEMS ETC.), SINCE THE LATER ARE AS A RULE VERY SOPHISTICATED. ALL EXPERIENCES WITH BIBLIOGRAPHIC SYSTEMS FOR PERSONAL USE SHOWED THAT THE ESSENTIAL PREREQUISITE FOR THE APPLICABILITY OF SUCH SYSTEMS IS TO KEEP THEM SIMPLE, BOTH FOR UNDERSTANDING AND USE.

THE SYSTEM PRESENTED IN THIS PAPER IS PARTIALLY RELATED TO SYSTEMS DESCRIBED IN REFERENCES 1, 2, AND 3. IT IS DESIGNED IN VIEW OF THE FACT THAT IN A RELATIVELY NARROW FIELD OF RESEARCH, A RESEARCHER DOES NOT NEED MORE THAN SEVERAL HUNDREDS OF 'ACTIVE' BIBLIOGRAPHIC RECORDS. BECAUSE OF THE LIMITED NUMBER OF RECORDS AND THE NEED FOR THEIR OFF-LINE ACCESSIBILITY AND READABILITY, IN THIS CASE IT IS CONVENIENT TO KEEP BIBLIOGRAPHIC RECORDS ON PUNCHED CARDS. ACCORDING TO THE GIVEN SPECIFIC NEEDS, VARIOUS SETS OF RECORDS TAKEN FROM THE BIBLIOGRAPHIC RECORD DATA BASE CAN BE READ, PROCESSED, EDITED, AND PRINTED IN REQUIRED FORM BY A COMPUTER, USING THE 'AB' PROGRAMMING SYSTEM DESCRIBED IN THE SEQUEL.

THE 'AB' PROGRAMMING SYSTEM

ALL TEXTS PRODUCED BY THE 'AB' PROGRAMMING SYSTEM HAVE ADJUSTABLE WIDTH IN ORDER TO FIT THE AVAILABLE SPACE FOR PRINTING, AND ARE BOTH LEFT AND RIGHT JUSTIFIED. BEFORE PROCESSING, THESE TEXTS MUST BE PREPARED ON PUNCHED CARDS IN A NON-EDITED FORM WITH ARBITRARY SPACING BETWEEN WORDS. THEREFORE, THE POSSIBILITIES FOR EASY COMBINING VARIOUS SEGMENTS OF THE TEXT, AND FOR THE TEXT MODIFICATION AND UPDATING ARE SUFFICIENTLY PROVIDED. FOLLOWING ARE THE FOUR BASIC OUTPUTS OF THE 'AB' SYSTEM:

(1) ARBITRARY SEGMENTS OF THE TEXT SEPARATED BY BLANK LINES. THE PRESENT PAPER IS AN EXAMPLE OF SUCH A TEXT.

(2) LISTS OF REFERENCES. THE USER MUST PREPARE THE DESIRED INPUT SEQUENCE OF REFERENCES ON PUNCHED CARDS. REFERENCES AT THE END OF THIS PAPER CAN SERVE AS AN EXAMPLE OF SUCH AN OUTPUT.

(3) ANNOTATED BIBLIOGRAPHIES. EACH RECORD IN AN ANNOTATED BIBLIOGRAPHY MAY BE FOLLOWED BY AN ARBITRARY TEXT SERVING USUALLY AS ABSTRACT, COMMENT, OR CRITICISM. AN EXAMPLE OF ANNOTATED BIBLIOGRAPHY CAN BE FOUND IN REFERENCE 4.

(4) CLASSIFIED BIBLIOGRAPHIES. IF THE USER DEFINES A SET OF CLASSES AND ASSIGNS THE ONE-CHARACTER CLASSIFICATION CODE TO EACH CLASS, IT IS POSSIBLE TO ASSOCIATE THE CORRESPONDING MULTI-CHARACTER RECORD CLASSIFICATION CODE TO EACH RECORD ACCORDING TO ITS APPURTENANCE TO VARIOUS CLASSES. FOR EACH GIVEN INPUT CLASSIFICATION CODE THE EXISTING RECORDS CAN BE TESTED IN ORDER TO ESTABLISH THE LIST OF RECORDS WHOSE RECORD CLASSIFICATION CODE CONTAINS THE INPUT CLASSIFICATION CODE. AN EXAMPLE OF THE USE OF CLASSIFIED BIBLIOGRAPHIES IS PRESENTED IN REFERENCE 4.

THE 'AB' PROGRAMMING SYSTEM CONSISTS OF FOUR SUBROUTINES (INPUT, REDUC, CODE, AND ROW) AND THE MAIN PROGRAM. SUFFICIENTLY COMMENTED TEXTS OF ALL PROGRAMS FOLLOWED BY THE CORRESPONDING USER MANUAL ARE PRESENTED IN THE SEQUEL.

```
C*****************************************
      SUBROUTINE INPUT(L,NL,LDIM)
C*****************************************
C --- INITIALIZATION OF THE CHARACTER-VECTOR L
C --- NL = INDEX OF THE LAST NON-BLANK
C      CHARACTER WITHIN THE VECTOR L
C --- IF L CONTAINS ONLY BLANKS, THEN NL=0, AND
C      NL=-1 IF 'END' IS DETECTED AT THE END OF
C      VECTOR L
C --- LDIM = DIMENSION OF VECTOR L
C-----------------------------------------
      DIMENSION L(1),IEND(3)
      COMMON KIN,KOUT
      DATA IBL/' '/,IEND/'E','N','D'/

      DO 10 I = 1,LDIM,80
      NL=I+79
      READ(KIN,5)(L(J),J=I,NL)
    5 FORMAT(80A1)
      IF(L(NL)-IBL) 16,10,16
   10 CONTINUE
      WRITE(KOUT,15)
   15 FORMAT(' *** RECORD LENGHT ERROR'///)
      PAUSE

   16 IF(L(NL)-IEND(3)) 21,17,21
   17 IF(L(NL-1)-IEND(2)) 21,18,21
   18 IF(L(NL-2)-IEND(1)) 21,19,21
   19 NL=-1
   20 RETURN

   21 NL=NL-1
      IF(NL) 20,20,25
   25 IF(L(NL)-IBL) 20,21,20
      END
C*****************************************
      SUBROUTINE REDUC(L,NL)
C*****************************************
C --- REDUCTION OF THE NL-DIMENSIONAL
C      CHARACTER-VECTOR L BY REDUCING THE
C      NUMBER OF BLANKS BETWEEN WORDS TO ONE
C-----------------------------------------
      DIMENSION L(1)
      DATA IBL/' '/
      INDEX=0
      INDIC=1
      DO 30 I=1,NL

      IF(L(I)-IBL) 20,5,20
    5 IF(INDIC) 30,10,30

   10 INDIC=1
      GO TO 25
   20 INDIC=0

   25 INDEX=INDEX+1
      L(INDEX)=L(I)
   30 CONTINUE
      NL=INDEX
      RETURN
      END
```

```
C*****************************************
      SUBROUTINE ROW(L,NL,NCHAR,LINI,K,KEND)
C*****************************************
C      SELECTION OF NCHAR CHARACTERS FROM THE
C      INPUT VECTOR L, STARTING WITH L(LINI),
C      AND THEIR TRANSFER IN THE OUTPUT VECTOR
C      K .  DURING THE TRANSFER BLANKS ARE
C      INSERTED BETWEEN WORDS, SO THAT THE
C      TEXT IN VECTOR K IS BOTH LEFT AND RIGHT
C      JUSTIFIED, AND READY FOR PRINTING. THE
C      LAST NON-BLANK CHARACTER IN VECTOR L IS
C      L(NL). IF THE OUTPUT INDICATOR KEND=1,
C      THEN THE VECTOR K CONTAINS THE LAST
C      SEQUENCE OF CHARACTERS FROM THE VECTOR L
C      (OTHERWISE, KEND=0). AFTER THE TRANSFER
C      OF CHARACTERS THE POINTER LINI IS
C      AUTOMATICALLY INCREMENTED.
C      INITIALIZATION LINI=1 MUST BE DONE IN
C      THE PROGRAM WHICH CALLS SUBROUTINE ROW.
C-----------------------------------------
      DIMENSION L(1),K(1)
      DATA IBL/' '/
      IF(NCHAR-(NL-LINI+1)) 25,5,5
C---  LAST ROW
    5 IND=1
      DO 10 I=LINI,NL
      K(IND)=L(I)
   10 IND=IND+1
      IF(IND-NCHAR) 15,15,20
   15 DO 19 I=IND,NCHAR
   19 K(I)=IBL
   20 KEND=1
      RETURN
C---  NON-LAST ROW
   25 LEND=LINI+NCHAR-1
      IF(L(LEND+1)-IBL) 30,26,30

C---  NO MODIFICATION
   26 IND=0
      DO 28 I=LINI,LEND
      IND=IND+1
   28 K(IND)=L(I)
      GO TO 160

C---  SEARCHING THE LAST WORD IN A ROW
   30 DO 50 I=1,NCHAR
      LEND=LEND-1
      IF(L(LEND+1)-IBL) 50,70,50
   50 CONTINUE
      WRITE(3,60)
   60 FORMAT(' TEXT LENGHT ERROR'///)
      RETURN
C---  NBL = NUMBER OF BLANKS TO BE INSERTED
C---  MBL = NUMBER OF BLANKS BETWEEN WORDS
   70 NBL=I
      MBL=0
      DO 80 I=LINI,LEND
      IF(L(I)-IBL) 80,75,80
   75 MBL=MBL+1
   80 CONTINUE
      NUMB=NBL/MBL
      KEND=MBL-NBL+NUMB*MBL

C---  BLANKS INSERTION
      JB=0
      INK=0
      DO 150 I=LINI,LEND
      IF(L(I)-IBL) 90,95,90
   90 INK=INK+1
      K(INK)=L(I)
      GO TO 150
   95 JB=JB+1
      IF(JB-KEND) 100,100,105
  100 MMM=NUMB+1
      GO TO 106
  105 MMM=NUMB+2
  106 DO 110 J=1,MMM
      INK=INK+1
  110 K(INK)=IBL
  150 CONTINUE
  160 LINI=LEND+2
      KEND=0
      RETURN
      END
```

```
C*******************************************
      SUBROUTINE CODE(L,NL,KOD,NKOD,ID)
C*******************************************
C --- DETECTION OF THE TEXT (CODE=A......H)
C     WITHIN THE NON-EMPTY CHARACTER-VECTOR L
C --- NL = INDEX OF THE LAST NON-BLANK
C     COMPONENT OF VECTOR L
C --- KOD = OUTPUT VECTOR CONTAINING THE
C     CLASSIFICATION CODE A......H
C --- NKOD = INDEX OF THE LAST NON-BLANK
C     COMPONENT OF VECTOR KOD
C --- ID = 1 IF (CODE=A......H) IS THE ONLY
C     CONTENT OF VECTOR L
C --- ID = 2 IF (CODE=A......H) IS PLACED
C     AFTER SOME TEXT AT THE END OF VECTOR L
C --- ID = 3 IF THE TEXT DOES NOT CONTAIN
C     THE CODE
C-------------------------------------------
      DIMENSION L(1),KOD(8),IT(6)
      DATA IT,IPAR/'(','C','O','D','E','=',')'/

C --- SEARCHING THE BEGINNING OF VECTOR L
      DO 10 I=1,6
      IF(L(I)-IT(I)) 32,10,32
   10 CONTINUE
      NKOD=0
      DO 20 I=7,14
      IF(L(I)-IPAR) 15,25,15
   15 NKOD=NKOD+1
   20 KOD(NKOD)=L(I)
   25 ID=1
      RETURN

C --- SEARCHING THE END OF VECTOR L
   32 IF(L(NL)-IPAR) 45,35,45
   35 IP=NL-1
      DO 40 I=1,8
      IP=IP-1
      IF(L(IP)-IT(6)) 40,50,40
   40 CONTINUE
   45 ID=3
      RETURN

   50 IP=IP-5
      DO 55 I=1,5
      IF(L(IP)-IT(I)) 45,55,45
   55 IP=IP+1
      IP=IP+1
      IK=NL-1
      NKOD=0
      DO 60 I=IP,IK
      NKOD=NKOD+1
   60 KOD(NKOD)=L(I)
      ID=2
      RETURN
      END

C*******************************************
C     M A I N   P R O G R A M   'AB'
C*******************************************
C --- LDIM = DIMENSION OF VECTORS L1 AND L2
C --- KIN = CARD READER CODE
C --- KOUT = LINE PRINTER CODE
C --- MXREC = MAXIMAL NUMBER OF BIBLIOGRAPHIC
C     RECORDS
C --- MAXIMAL NUMBER OF CARDS IN A HALF-RECORD
C     IS LDIM/80 (HERE LDIM/80 = 25)
C-------------------------------------------
      DIMENSION L1(2000),L2(2000),KOD(8),
     *,MAKOD(500,8), IROW(120), IEND(3)
      COMMON KIN,KOUT
      DATA IBL/' '/,IEND/'E','N','D'/

C --- INITIALIZATIONS
      LDIM = 2000
      KIN = 2
      KOUT = 3
      MXREC = 500

      DO 5 I=1,MXREC
      DO 5 J=1,8
    5 MAKOD(I,J) = IBL
      WRITE(KOUT,7)
    7 FORMAT(1H1)

C --- TEXT WIDTH AND PRINTING OPTIONS DATA
      READ(KIN,10) CM,NUMB,LINE1,LINE2,ICLAS,I
   10 FORMAT(F5.0,5I1)
      NCHAR=3.92*CM-(2+I)*NUMB

C --- TITLES AND COMMENTS
   15 READ(KIN,20)(L1(I),I=1,80)
   20 FORMAT(80A1)
      WRITE(KOUT,25) (L1(I),I=1,79)
   25 FORMAT(5X,79A1)
      IF(L1(80) - IBL) 30,15,30
   30 WRITE(KOUT,35)
   35 FORMAT(//)

C --- RECORD PROCESSING CYCLE
      DO 180 NREC=1,MXREC

C --- READING OF THE FIRST HALF-RECORD
   40 CALL INPUT(L1,NL1,LDIM)
      IF(NL1) 200,45,55
   45 WRITE(KOUT,50)
   50 FORMAT(' *** EMPTY RECORD'///)
      PAUSE
      GO TO 40
   55 CALL REDUC(L1,NL1)

C --- READING OF THE SECOND HALF-RECORD
      CALL INPUT(L2,NL2,LDIM)
      IF(NL2) 85,85,60
   60 CALL REDUC(L2,NL2)
      CALL CODE(L2,NL2,KOD,NKOD,IDENT)
      GO TO(65,75,85),IDENT

C --- ADDITION OF THE CODE AT THE END
C     OF VECTOR L1
   65 NL1=NL1+1
      L1(NL1)=IBL
      DO 70 I=1,NL2
      NL1=NL1+1
   70 L1(NL1)=L2(I)
      NL2=0

C --- INITIALIZATION OF THE CODE MATRIX
   75 DO 80 I=1,NKOD
   80 MAKOD(NREC,I)=KOD(I)

C --- PRINTING OF THE FIRST HALF-RECORD
   85 IF(LINE1) 90,110,90
   90 WRITE(KOUT,100)
  100 FORMAT(1X)
  110 LBEGI=1
      INDIC=0
  111 CALL ROW(L1,NL1,NCHAR,LBEGI,IROW,LAST)
      IF(INDIC) 135,120,135
  120 INDIC=1
      IF(NUMB) 125,135,125
  125 WRITE(KOUT,130) NREC,(IROW(I),I=1,NCHAR)
  130 FORMAT(I5,'. ',103A1)
      GO TO 145
  135 WRITE(KOUT,140) (IROW(I),I=1,NCHAR)
  140 FORMAT(7X,103A1)
  145 IF(LAST) 150,111,150

C --- PRINTING OF THE SECOND HALF-RECORD
  150 IF(NL2) 200,180,155
  155 IF(LINE2) 160,170,160
  160 WRITE(KOUT,100)
  170 LBEGI=1
  171 CALL ROW(L2,NL2,NCHAR,LBEGI,IROW,LAST)
      WRITE(KOUT,140)(IROW(I),I=1,NCHAR)
      IF(LAST) 180,171,180
  180 CONTINUE
C --- CLASSIFICATION
  200 NREC=NREC-1
      WRITE(KOUT,202)
  202 FORMAT(1H1,'CLASSIFICATION OF ',
     *,'BIBLIOGRAPHIC ENTRIES'/40('=')/)
      IF (ICLAS) 203,208,203
  203 WRITE(KOUT,204)
  204 FORMAT(//' LIST OF RECORD ',
     *,'CLASSIFICATION CODES'/)
      WRITE(KOUT,205)
     *, (I,(MAKOD(I,J), J=1,8), I=1,NREC)
  205 FORMAT(5(I5,1X,8A1))
```

```
  208 WRITE(KOUT,100)
C --- READING THE CLASSIFICATION CODE
      READ(KIN,20) KOD,(IROW(I),I=1,72)
      IF(IROW(70)-IEND(1)) 230,210,230
  210 IF(IROW(71)-IEND(2)) 230,220,230
  220 IF(IROW(72)-IEND(3)) 230,222,230
  222 CALL EXIT

  230 LAST=72
  235 IF(IROW(LAST)-IBL) 244,240,244
  240 LAST=LAST-1
      IF(LAST) 250,250,235
  244 WRITE(KOUT,246) (IROW(I),I=1,LAST)
  246 FORMAT(' CLASS = ',72A1)

  250 NKOD=8
  255 IF(KOD(NKOD)-IBL) 265,260,265
  260 NKOD=NKOD-1
      IF(NKOD) 208,208,255
  265 WRITE(KOUT,268) KOD
  268 FORMAT(' CODE = ',8A1)

C --- SEARCHING THE RECORDS BELONGING TO
C     THE GIVEN CLASS
      NL1=0
      L1(1)=0
      DO 300 I=1,NREC
      LAST=8
  270 IF(MAKOD(I,LAST)-IBL) 280,275,280
  275 LAST=LAST-1
      IF(LAST) 300,300,270
  280 DO 290 IKOD=1,NKOD
      DO 285 J=1,LAST
      IF(KOD(IKOD)-MAKOD(I,J)) 285,290,285
  285 CONTINUE
      GO TO 300
  290 CONTINUE
      NL1=NL1+1
      L1(NL1)=I
  300 CONTINUE
      WRITE(KOUT,310)(L1(I),I=1,NL1)
  310 FORMAT(1014)
      GO TO 208
      END
```

PROGRAMMING SYSTEM 'AB' FOR EDITING AND PRINTING ANNOTATED BIBLIOGRAPHIES / USER MANUAL

'AB' IS A PROGRAMMING SYSTEM FOR EDITING AND PRINTING ANNOTATED BIBLIOGRAPHIES AND SIMILAR TEXTS. ALL INPUT DATA MUST BE PUNCHED ON CARDS AND FOLLOW THE INITIAL 'EXECUTE AB' CARD. THE INPUT DECK CONTAINS (1) TEXT WIDTH AND PRINTING OPTION DATA, (2) COMMENTS, (3) BIBLIOGRAPHIC RECORDS (OR SIMILAR DATA), AND (4) DATA FOR CLASSIFIED BIBLIOGRAPHIES. A DETAILED DESCRIPTION OF THESE INPUTS IS PRESENTED IN THE SEQUEL.

(1) TEXT WIDTH AND PRINTING OPTION DATA ARE PUNCHED IN THE FIRST DATA CARD ACCORDING TO THE FORMAT F5.0, 5I1. THE FIRST FIELD CONTAINS THE REQUESTED TEXT WIDTH OF EDITED AND PRINTED BIBLIOGRAPHIC RECORDS EXPRESSED IN CENTIMETERS. THE NEXT FOUR FIELDS CONTAIN BINARY DATA (0 OR 1) GIVING RESPECTIVELY THE POSSIBILITY TO OMIT OR TO INCLUDE THE FOLLOWING OPTIONS - (I) NUMERATING OF BIBLIOGRAPHIC RECORDS, (II) SINGLE LINE SPACING BETWEEN RECORDS, (III) SINGLE LINE SPACING BETWEEN HALF-RECORDS (THE TERM HALF-RECORD IS DEFINED LATER IN THIS TEXT), AND (IV) PRINTING THE LIST OF RECORD CLASSIFICATION CODES. THE LAST FIELD CONTAINS THE MAXIMAL NUMBER OF DIGITS USED FOR RECORD NUMERATING (USUALLY 1,2 OR 3).

(2) A NON LIMITED NUMBER OF FIRST INPUT CARDS MAY CONTAIN ARBITRARY COMMENTS (MOST FREQUENTLY TITLES AND RELATED INFORMATION). THESE COMMENTS ARE PUNCHED IN COLUMNS 1-79, WHILE THE 80-TH COLUMN MUST BE LEFT BLANK. ALL COMMENTS ARE PRINTED BY THE LINE PRINTER EXACTLY AS THEY ARE PUNCHED ON CARDS. BLANK COMMENT CARDS MAY BE USED FOR SPACING. THE LAST COMMENT CARD MUST BE DENOTED BY A NON-BLANK CHARACTER IN THE 80-TH COLUMN. AFTER THE LAST COMMENT CARD THE SEQUENCE OF BIBLIOGRAPHIC RECORDS MAY FOLLOW.

(3) BIBLIOGRAPHIC RECORD IS A SET OF PUNCHED CARDS CONTAINING ARBITRARY BIBLIOGRAPHIC OR OTHER DATA IN COLUMNS 1-79. THE RECORD IS DIVIDED IN TWO PARTS, HEREAFTER REFERRED AS HALF-RECORDS. THE LAST CARD OF EACH HALF-RECORD MUST CONTAIN A NON-BLANK CHARACTER IN THE 80-TH COLUMN. IF THE 'AB' SYSTEM IS USED FOR EDITING ANNOTATED BIBLIOGRAPHIES, THE FIRST HALF-RECORD CONTAINS THE NAME OF THE AUTHOR, TITLE, JOURNAL OR PUBLISHER AND THE RANGE OF PAGES. THE SECOND HALF-RECORD CONTAINS ABSTRACT AND/OR SOME COMMENTS. IF THE BIBLIOGRAPHY IS CLASSIFIED, THE SECOND HALF-RECORD MUST BE ENDED BY THE RECORD CLASSIFICATION CODE STATEMENT HAVING THE FORM (CODE=A.....H) , WHERE A.....H DENOTES ONE-TO-EIGHT-CHARACTER RECORD CLASSIFICATION CODE. EACH CLASSIFICATION CHARACTER REPRESENTS THE SYMBOL OF A CLASS, AND DENOTES THAT THE RECORD IS ASSOCIATED WITH THE CORRESPONDING CLASS OF RECORDS. IF THE RECORD IS NOT CLASSIFIED, THE RECORD CLASSIFICATION CODE STATEMENT MUST BE OMITTED. IF THE RECORD CLASSIFICATION CODE IS THE ONLY CONTENT OF THE SECOND HALF-RECORD, IT WILL BE AUTOMATICALLY ADDED TO THE FIRST HALF-RECORD. IF THE SECOND HALF-RECORD IS EMPTY, THE BLANK CARD HAVING A NON-BLANK CHARACTER IN THE 80-TH COLUMN MUST BE INSERTED AFTER THE FIRST HALF-RECORD. THE SECOND HALF-RECORD IS PRINTED BEGINNING AT A NEW ROW. ALL RECORDS WILL BE EDITED AND PRINTED IMMEDIATELY AFTER READING, FORMING A COLUMN OF THE REQUESTED WIDTH. THE END OF A SEQUENCE OF BIBLIOGRAPHIC RECORDS MUST BE DENOTED BY THE CARD HAVING CHARACTERS 'END' PUNCHED IN COLUMNS 78-80.

(4) DATA FOR CLASSIFIED BIBLIOGRAPHIES CONSIST OF A NUMBER OF DATA CARDS ENDED BY THE CARD HAVING CHARACTERS 'END' PUNCHED IN COLUMNS 78-80. EACH DATA CARD MAY HAVE AN ARBITRARY INPUT CLASSIFICATION CODE PUNCHED IN COLUMNS 1-8, LEFT JUSTIFIED. COLUMNS 9-80 MAY BE USED FOR THE TITLE OF THE CLASS ASSOCIATED WITH THE INPUT CLASSIFICATION CODE. AFTER THE READING OF A DATA CARD, REFERENT NUMBERS OF ALL RECORDS WHOSE RECORD CLASSIFICATION CODES CONTAIN THE GIVEN INPUT CLASSIFICATION CODE WILL BE PRINTED ON A LINE PRINTER.

REFERENCES

1. BASSLER, R.A., DEMOODY, H.C., 'COMPUTER SYSTEM EVALUATION AND SELECTION - AN ANNOTATED BIBLIOGRAPHY AND KEYWORD INDEX.' COLLEGE READINGS INC., ARLINGTON, VIRGINIA, U.S.A., 1971.

2. MILLER, E.F. JR., 'BIBLIOGRAPHY ON TECHNIQUES OF COMPUTER PERFORMANCE ANALYSIS.' COMPUTER, SEPTEMBER - OCTOBER 1972, PP. 39-47.

3. KRAUS, L.L., 'PRIVATE INFORMATION SYSTEM FOR LITERATURE CATALOGING.' PART II IN THE AUTHOR'S UNPUBLISHED B.S. THESIS, UNIVERSITY OF BELGRADE, DEPARTMENT OF ELECTRICAL ENGINEERING, 1973. (IN SERBO - CROATIAN)

4. DUJMOVIĆ, J.J., 'THE PREFERENCE SCORING METHOD FOR DECISION MAKING - SURVEY, CLASSIFICATION AND ANNOTATED BIBLIOGRAPHY'. INFORMATICA (YUGOSLAV JOURNAL), NO. 2, 1977.

upis kontinuiranih signala u mikroračunalo

v. zgaga

UDK 681.3-181.4:615.84

Elektrotehnički fakultet,
41000 Zagreb

Za potrebe obrade bioelektričnih signala (EKG,EEG) u realnom vremenu napravljen je uredjaj AS (analogni subsistem), predviđen za rad sa procesorom 8080. Parametri analogno-digitalne pretvorbe (period uzorkovanja, mjerni opseg) zadaju se programom. Postoje i 16-kanalni ulazni multipleksor i D/A pretvarač, također kontrolirani programom. Osim pod kontrolom računala uredjaj može raditi i autonomno : kao digitalni voltmetar, generator zadane frekvencije, te programirani izvor referentnog napona - tada se podaci unose ručno, preko decimalne tastature.

THE INPUT OF ANALOG SIGNALS INTO MICROCOMPUTER: Hardware realization of the programmable A/D converter, designed as a part of the 8080 microcomputer sistem, is described. The aim of this unit is to make possible real-time EEG and EKG analysis. Conversion parameters, such as conversion period, input voltage range and number of input channels (up to 16), may be modified by a program. In addition, the unit could be used autonomically as a digital voltmeter. In this case, it is programmed through its own keyboard.

1. UVOD

Ovdje opisani uredaj, AS (Analogni Subsistem), nastao je u težnji za realizacijom sistema za obradu EKG i EEG signala u realnom vremenu. Obrada u realnom vremenu može zahtjevati i promjenu parametara A/D pretvorbe (period uzorkovanja, pojačanje signala prije pretvorbe, broj kanala koji se učitavaju) za vrijeme rada, što znači da se navedeni parametri trebaju moći kontrolirati programom. AS zadovoljava taj uvjet. U nedostatku računala uredjaj može raditi i samostalno - tada se programiranje rada vrši malom tastaturom na prednjoj ploči.

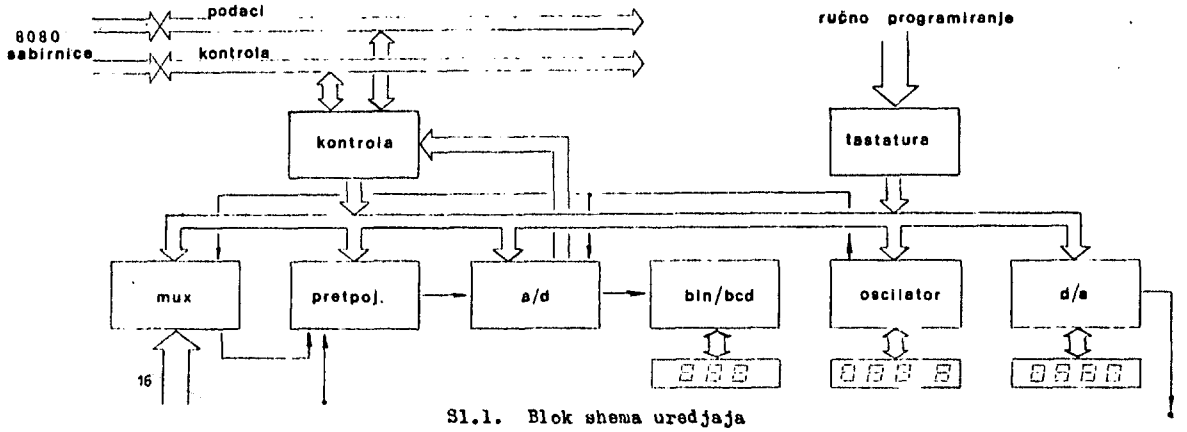
Za obradu signala je predviđeno mikroračunalo sa procesorom 8080, pa je AS predviđen za direktno spajanje na sabirnice tog sistema. U slijedećem poglavlju opisani su glavni sklopovi uredjaja, a na kraju je dan i pojednostavljeni primjer programa koji prvo zadaje period uzorkovanja, a zatim u prekidnom modu (interrupt) učitava rezultate A/D pretvorbe.

2. OPIS UREĐAJA

Blok shema uredaja prikazana je na sl.1. Kontrolni sklop povezuje AS sa mikroračunarskim sistemom. Naredbe i podaci koji stižu iz mikroračunala primaju se, po potrebi dekodiraju i prepakiraju, te šalju u AS internim kontrolnim sabirnicama. Preko tih sabirnica zadaje se period uzorkovanja, definira se rad analognog multipleksora, pojačanje pretpojačala, mod rada A/D pretvarača, te kod D/A pretvorbe mod rada i binarna ili NBCD vrijednost napona za D/A pretvarač. Izlaz iz kontrolnog sklopa na interne kontrolne sabirnice može se dovesti u "treće stanje", tj. odspojiti od sabirnica. Kontrolu tada preuzima tastatura sa prednje ploče uredjaja. Na taj način može se ručno upravljati svim funkcijama uredjaja.

Analogni multipleksor (16:1) spaja, prema instrukcijama koje dobija, jedan od 16 ulaznih kanala na pretpojačalo za A/D pretvorbu. Može biti izabran bilo koji od ulaza, ili se sekvencijalno mogu očitavati ulazi od prvog do izabranog. Pretpojačalo može biti spojeno na izlaz iz multipleksora ili na zasebni ulaz (kad AS radi kao digitalni voltmetar), a pojačanje se može zadati u opsegu 10^6 (za vanjski ulaz ± 5KV do ± 5mV). Postoji mogućnost automatskog namještanja nule, te mjerenja vršne i srednje vrijednosti za izmjenični signal (kad radi kao DVM).

Izlaz pretpojačala vodi se na ulaz A/D pretvarača. Upotrebljen je modularni pretvarač (30 μs, 10 bita). Izlaz iz pretvarača (10 bita, paralelno) vodi se preko internih A/D sabirnica u kontrolni sklop, koji podatke proslijeđuje u mikroračunalo. Serijski izlaz iz pretvarača vodi se u binarno/NBCD pretvornik,



Sl.1. Blok shema uredjaja

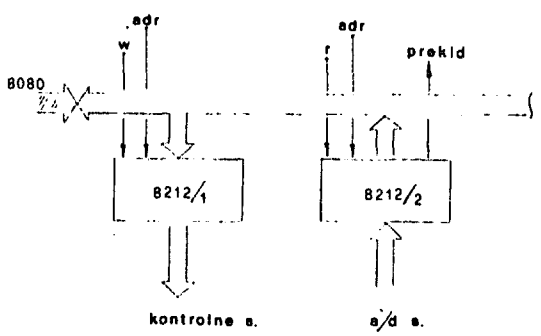
koji binarnu vrijednost mjerenog napona pretvara u tri BCD znamenke. Tako je u svakom trenutku dostupan rezultat pretvorbe (na prednjoj ploči su 7-segmentni indikatori).

Period konverzije odredjuje se programabilnim NBCD brojilom. Brojilo sadrži tri znamenke (od 000 do 999), a jedna znamenka definira eksponent (0 do 6). Sve 4 znamenke prikazuju se indikatorima na prednjoj ploči ($001 \cdot (10)^0$ μs do $999 \cdot (10)^6$ μs). Ako se sekvencijalno očitava više kanala, onda se brojilom zadaje period izmedju dva očitanja istog kanala. Uz naponom kontrolirani oscilator i petlju zatvorene faze (PLL) može se realizirati generator programom zadane frekvencije.

Za potrebe simulacije analognih signala AS-u je dodan i D/A pretvarač (modularni 12-bitni, 5 μs). Napon se može numerički zadati programom: 8 bita binarno, ili u NBCD kodu sa 4 znamenke. Zadani napon prikazuje se na 7-seg. indikatorima na prednjoj ploči.

2.1. KONTROLNI SKLOP I TASTATURA

Na slici 2.1. je prikazan kontrolni sklop. Razmjena podataka sa mikroračunalom vrši se pomoću Intelovih integriranih krugova 8212 (8-bitni ulazno izlazni sklop). Svaki 8212 adresira se zasebno. Ulaz prvoga i izlaz drugoga spajaju se na sabirnice podataka procesora, tj. na mikroračunalo. Prvog 8212 procesor aktivira ulazno/izlaznom naredbom I/O WRITE (W), uz odgovarajuću adresu.



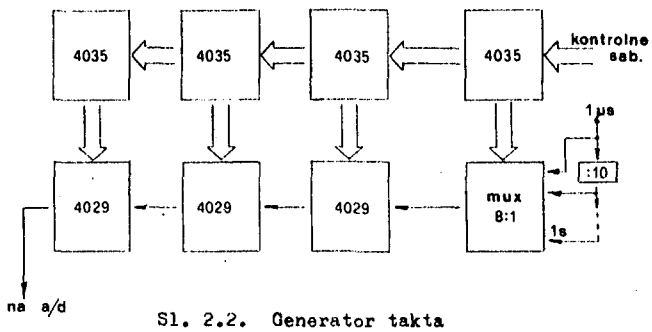
Sl. 2.1. Kontrolni sklop

Tom naredbom se sadržaj akumulatora (u procesoru) pojavljuje na sabirnicama podataka. Impulsi dekodirane adrese i "W" upisuju taj podatak u $8212/_1$, gdje on ostaje sačuvan do slijedeće izlazne naredbe na istoj adresi. Izlaz iz $8212/_1$ spojen je direktno na interne kontrolne sabirnice. Taj izlaz se prekidačem sa prednje ploče može "odspojiti" od sabirnica. Time se ujedno na sabirnice spaja dekoder tastature. Tastatura je decimalna, a izlaz iz dekodera u BCD kodu. $8212/_2$ procesor aktivira ulazno/izlaznom naredbom I/O READ (R), opet uz odgovarajuću adresu. Tom naredbom se izlaz 8212/2 priključuje na sabirnice podataka procesora, a procesor taj podatak sprema u akumulator. Ulaz u $8212/_2$ su A/D sabirnice, koje sadrže rezultat posljednje A/D pretvorbe.

Od tri osnovna načina razmjene podataka sa računalom (ispitivanjem statusa A/D pretvarača, prekidno, te direktni pristup memoriji(DMA)), izabran je prekidni (interrupt) način rada. Po završetku pojedine A/D pretvorbe postavlja se bistabil "zahtjev za prekid" (unutar $8212/_2$). Procesor završava tekuću instrukciju, te prelazi u prekidni potprogram (npr. učitavanje rezultata A/D pretvorbe iz $8212/_2$ instrukcijom IN (R), te spremanje tog podatka u neku memorijsku lokaciju).

2.2. GENERATOR TAKTA ZA UZORKOVANJE

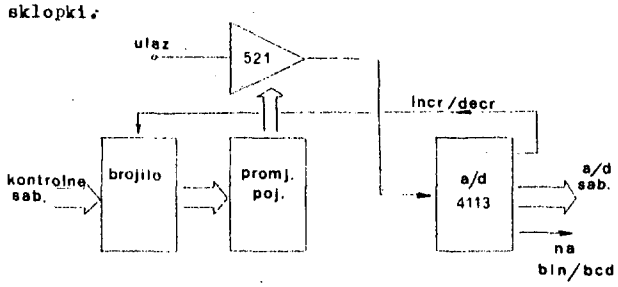
Blok shema generatora takta dana je na sl.2.2. Četiri 4-bitna registra (4035) primaju preko kontrolnih sabirnica 4 BCD znamenke koje definiraju period oscilacija. Sadržaj sva 4 registra se preko BCD/7-seg. dekodera prikazuje na četiri 7-seg. LED indikatora. Prva tri registra upravljaju programabilnim 4-bitnim BCD brojilima (4029), a četvrti, koji sadrži eksponent, upravlja multiplekserom 8:1. Na ulaze multipleksora vodi se osnovni takt od 1 μs podijeljen sa 10^N (N=0,1,..6). Impulsi na izlazu multipleksora traju 1 μs, 10 μs, .. 1 sekundu - ovisno o sadržaju zadnjeg registra. Ti impulsi se u brojilima dijele sa sadržajem prva 3 registra (:1 do :999). Tako se na izlazu zadnjeg brojila dobiju impulsi od 1 μs do 999 sekundi.



Sl. 2.2. Generator takta

2.3. A/D PRETVARAČ

Za kontrolu pojačanja pretpojačalu koristi se binarno "up-down" brojilo i BCD/decimalno dekoder. Početni mjerni opseg zada se preko kontrolnih sabirnica tako da se brojilo postavi na željenu vrijednost. Ako tokom rada A/D pretvarač poprimi maksimalnu vrijednost, brojilo će se dekrementirati (time se pojačanje smanjuje 10 puta). Obrnuto, ako je izlaz manji od 10% maksimalnog, brojilo će se inkrementirati. Ova automatika se naredbom može isključiti.
Za pretpojačalo je upotrebljeno diferencijalno instrumentaciono pojačalo AD 521, a faktor pojačanja se bira uključivanjem odgovarajućih otpornika ovisno o sadržaju brojila i stanju dekodera, a pomoću CMOS sklopki.

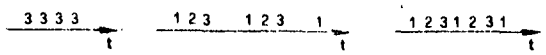


Sl. 2.3. A/D pretvarač

Za A/D pretvarač iskorišten je modul Teledyne Philbrick-a 4113 (ulazni raspon $\pm 5V$, 10 bita, vrijeme jedne konverzije 30 μs). Izlaz iz pretvarača je preko "three state" logike spojen na interne A/D sabirnice.

2.4. ANALOGNI MULTIPLEKSOR

Kako bi se direktno mogli upisivati izlazi iz 16-kanalnog EEG-a, uredjaj sadrži i 16:1 analogni multipleksor. Programabilno brojilo može adresirati bilo koji od 16 ulaza, ili redom od prvog do N-tog. Modovi rada prikazani su na slici 2.4. :



3.1. SAMOSTALNI RAD

Ako uređaj nije priključen na mikroračunalo, internim kontrolnim sabirnicama upravlja vlastita tastatura. Tastaturom treba samo definirati vrijeme konverzije, pa se uređaj može koristiti kao automatski digitalni voltmetar. Može se izabrati za mjerenje bilo koji od 16 priključenih napona. Istovremeno se mogu koristiti i programabilni izvor referentnog napona i programabilni oscilator, što uređaj čini vrlo pogodnim za razna mjerenja i ispitivanja.

3.2. RAD U MIKRORAČUNARSKOM SISTEMU

Kod rada u μR sistemu sve funkcije uredjaja mogu se definirati programom. Npr. kod zadavanja vremena izmedu dvije konverzije (period uzorkovanja) potrebno je na izlaznu adresu za $8212/_1$ poslati prvo internu adresu programabilnog oscilatora (80_{16}), zatim četiri BCD znaka. To se postiže slijedećim programskim isječkom (assembler za 8080):

```
MVI  A,80
OUT  01
MVI  A,02
OUT  01
MVI  A,05
OUT  01
MVI  A,00
OUT  01
MVI  A,03
OUT  01
```

Naredba MVI A,XX upisuje 8-bitni broj XX_{16} u akumulator, a naredba OUT NN taj isti broj ispisuje u izlaznu jedinicu čija je adresa NN. Ako je adresa od $8212/_1$ baš 01, po izvršenju ovog programskog odsječka programabilni oscilator će oscilirati sa periodom $250 \cdot 10^3$ μs.
Upis rezultata konverzije obično se vrši periodički (A/D pretvarač se aktivira oscilatorom - npr. svakih 250 ms, a po završetku pojedine pretvorbe postavi se "zahtjev za prekid" procesoru). Kod obrade zahtjeva z za prekid koristi se posebna CALL instrukcija, RST ("restart"). Ta instrukcija je realizirana hardwareski, tako da procesor automatski prelazi na početak prekidnog potprograma. Primjer za upis rezultata A/D pretvorbe (nakon što je definiran period uzorkovanja) izgledao bi ovako :

IN 03 upisuje rezultat konverzije iz $8212/_2$ u akumulator, a sa MOV M,A se taj podatak sprema u memoriju (adresa je u registrima H,L). INX H,L inkrementira taj par registara. Slijedi EI (enable int.) kako bi se ponovo dozvolio prekid, te RET (return), povratak u glavni program.

komunikacija med sekvenčnimi procesi - pregled del II.

m. exel

Institut "Jožef Stefan", Ljubljana

UDK 681.3.01

Članek obravnava probleme, ki jih zastavlja organizacija medprocesnih komunikacij. Podane so definicije osnovnih pojmov nakar sledi opis (s primeri) obstoječih jezikovnih konstruktov za medprocesno komunikacijo: semaforjev,(pogojno) kritičnih delov, monitorjev in procesnih poti.

COMMUNICATIONS AMONG SEQUENTIAL PROCESSES - A SURVEY (PART II.). The problems arising from the organization of interprocess communication are briefly sruveyed. The definitions of basic concepts are given. Descriptions (with examples) of the following primitives and language constructs for interprocess communication are presented: semaphores,(conditional) critical regions, monitors and path expressions.

V prvem delu (glej prvo številko Informatice) smo si ogledali nekatere splošne pojme,zadevajoče medprocesno komunikacijo in semaforje. V tem delu bomo opisali še druge jezikovne konstrukte za medprocesno komunikacijo.

4.2. (Pogojno) kritični del procesa

Prvo notacijo za kritične dele procesa je predlagal Hoare (1). Za sočasnost izvajanja procesov P1, ... Pn predlaga notacijo

{P1//P2 ...//Pn} ;

odvisnost procesov od skupnega vira r pa opiše z:

{resource r;P1// ...//Pn } .

Pri tem je hardverski ali pa logični vir r definiran oz. deklariran izven konstrukta { ... } oz. implicitno (npr. rezervirano ime za čitalnik, luknjalnik ipd.).

V procesih P1, ... Pn se kritični deli,ki zadevajo vir r opisujejo s konstruktom

with r do kritični -del;

Konstrukt za pogojno kritični del je:

with r when pogoj do kritični - del;

Pri tej obliki moramo paziti na implementacijo (primerjaj s 4.1.). Če pogoj ni izpolnjen se proces postavi v vrsto vezano na vir r; če je pogoj izpolnjen se izvrši kritični del in ob koncu tega dela se pogoj na novo preverja za vsak proces v vrsti vezani na vir r.

Kadar kritični del procesa Pi zadeva le kako komponento vira r,bomo zapisali:

with komponenta-vira-r...do krit.-del;

Uporabo zgornjih konstruktov bomo ilustrirali s Hoare-jevo rešitvijo za problem petih filozofov (glej 4.1.3. s sliko). Če je Pi proces,ki opisuje dejavnost filozofa i,bo sistem petih procesov zapisan takole:

{resource r; P0//P1//P2//P3//P4 };

vir r bo definiran takole (Pascal):

```
r: record
   vilice-na-razpolago:array 0..4 of 0..2;
   vilica-A, vilica-B,vilica-C, vilioa-D,
   vilica-E: tip-vilica
   end;
```

Prva tabela v viru r daje informacijo o stanju procesov: vilice-na-razpolago (i) = število vilic, ki so na razpolago filozofu i; začetna vrednost bo za vse filozofe 2.

Elementi vilica-A do vilica-E pa označujejo vilice kot posamezne delno kritične vire, ki sestavljajo vir r.

Rešitev za filozofski proces P3, na primer, bo sledeča:

```
P3: cycle
    begin
    misliti;
    with r. vilice-na-razpolago
    when r. vilice-na-razpolago(3)=2do
      begin r. vilice-na-razpolago(2):=
            r. vilice-na-razpolago(2)-1;
            r. vilice-na-razpolago(4):=
            r. vilice-na-razpolago(4)-1
      end;
    with r. vilica-D do
      with r. vilica-E do jesti;
    with r. vilice-na-razpolago do
      begin r. vilice-na-razpolago(2):=
            r. vilice-na-razpolago(2)+1;
            r. vilice-na-razpolago(4):=
            r. vilice-na-razpolago(4)+1
      end
    end cycle.
```

Drugo notacijo za kritične dele procesov je predlagal Brinch Hansen (14). Viri se pri njemu deklarirajo kot deljena informacija:

```
var v: shared tip.
```

Kritični deli imajo naslednje oblike:

```
region v do S;
region v when B do S; in
region v do S await B;
```

V svojem članku (14) Brinch Hansen tudi natančneje nakazuje (po Hoare-ju v 1) princip dokazovanja pravilnosti rešitve danega problema sinhronizacije procesov.

Dokazovanje bazira na takozvanih invarijantih ali nespremenljivih trditvah. Invarijanti so trditve, ki izražajo kriterije pravilnosti dane rešitve. Brinch Hansen omenja štiri kriterije izražajoče:

- pravilnost reaktiviranja (skedjuliranja) čakajočih procesov,
- spoštovanje zahtevanih sočasnih izključitev procesov,
- odsotnost mrtve točke ("deadlock"a) in
- spoštovanje zahtevanih relativnih prioritet procesov.

Ob primerjavi semaforjev s predlaganimi konstrukti za (pogojne) kritične dele zaključuje naslednje:

- s programskega stališča je uporaba slednjih preglednejša in jasnejša,
- s stališča dokazovanja s pomočjo invarijantov je stvar s slednjimi enostavnejša,
- pri uporabi kritičnih delov se programerju ni treba ukvarjati eksplicitno z "obujanjem" čakajočih procesov,
- s stališča implementacije je pri konstruktih za kritične dele možna določena neučinkovitost zaradi potrebe ponovnega preračunavanja pogojev (glej zgoraj opombo o implementaciji konstrukta with).

Kar se tiče zadnjega zaključka je nedavno Schmid (15) pokazal, da se lahko statično določi invarijantne relacije med (pogojno) kritičnimi deli procesov glede na dani vir v. Te relacije so neodvisne od poteka sistema procesov in so dveh vrst:

- usposabljajoče: izvedba (pogojnega) kritičnega dela kl lahko izpolni pogoj za izvedbo pogojnega kritičnega dela k2,
- neusposabljajoče: analogno definirane.

Določitev teh relacij se nato uporabi za učinkovitejšo implementacijo pogojnih kritičnih delov procesov.

Stvar si bomo pobliže ogledali: Schmid (15) definira skupne vire takole:

```
var v shared record v1...vn:
    ⟨tip⟩ end;
```

kjer so v_i spremenljivke stanj sistema procesov (glej 4.1.3.);
pogojno kritični del pa ima naslednjo splošno obliko:

```
region v do begin await pogoj1; op1;
                  ..
                  ..
              await pogojm; opm
          end;
```

kjer pogoj$_i$ lahko testira spremenljivke v_j, $1 \leq j \leq n$, op_i pa je lahko zaporedje (možno pogojnih) nastavitev spremenljivk v_j, $1 \leq j \leq n$.

Opisana implementacija zgornjih pogojnih kritičnih delov daje tudi možnost eksplicitnega skedjuliranja čakajočih procesov. Pri tem se uporabljajo spremenljivke tipa vrste:

```
var r: vrsta
```

in nekatere standardne procedure nad vrstami:

enter(r): postavi proces v vrsto r,

unblock-one(r), unblock-all(r): reaktiviraj enega ali vse procese čakajoče v vrsti r.

Oglejmo si sedaj primer čitalcev in piscev, ki si delijo skupen vir: čitalci lahko dosegajo vir sočasno, pisci pa seveda samo sekvenčno; pri tem imajo pisci prioriteto (ta primer je bil opisan že v 14).

Stanje sistema procesov je opisano v naslednji skupni informaciji:

```
var v: shared record ap,rp,rč: integer
    end,
```

kjer je ap število piscev, ki želijo doseči vir, rp in rč pa število piscev oz. čitalcev, ki razpolagajo z virom.

Rešitev za čitalce je sledeča:

```
...
region v do ⎧begin await ap ⩽ 0;
            ⎪      rč:=rč+1
        čl: ⎩end;
čitaj;
region v do
        č2: ⎧begin rč:=rč-1 end;
...
```

rešitev za pisce pa je:

```
...
region v do
     p1:   begin ⎧ap:=ap+1;
     p2:         ⎪await rč ⩽ 0 and
                 ⎪      rp ⩽ 0;
                 ⎩rp:=rp+1
           end;
piši;
region v do
     p3: ⎧begin rp:=rp-1;ap:=ap-1
         ⎩end;
...
```

Statično lahko ugotovimo naslednje usposabljajoče relacije: (č2,p2), (p3,čl), (p3,p2)
in naslednjo neusposabljajočo relacijo: (p2,p2).

Na primer (č2,p2) pomeni, da z zmanjšanjem rč za 1 v č2 lahko izpolnimo pogoj rč $\leq$ 0 kritičnega čela p2 in da torej lahko omogočimo izvedbo p2.

V implementaciji bomo potrebovali dve vrsti za čakanje:

```
var včit, vpis:vrsta;
```

Implementacija s skedjuliranjem bo za čitalca naslednja:

```
    ...
    region v do
    begin if not ap ≤ 0
začit:      then enter(včit);
            rč:=rč+1
    end;
    čitaj;
    region v do
    begin rč:=rč-1;
          com skedjuliraj;
kočit:    if rč ≤ 0 and rp ≤ 0
          then unblock-one(vpis)
    end;
    ...
```

za pisca pa sledeča:

```
          ...
          region v do
          begin ap:=ap+1;
                if not (rč ≤ 0 and rp ≤ 0)
začpis:         then enter (vpis)
                rp:=rp+1
          end;
          piši;
          region v do
          begin rp:=rp-1; ap:=ap-1;
                com skedjuliraj;
kopis:          if ap ≤ 0
                then unblock-all (včit)
                else if rč ≤ 0 and rp ≤ 0
                     then unblock-one(vpis)
          end;
          ...
```

Uporaba relacije (č2,p2), na primer, se odraža v implementaciji čitalca (glej del kočit) z retestiranjem pogoja kritičnega dela p2 ob koncu kritičnega dela č2. Če relacij ne bi statično ugotovili, bi bili prisiljeni testirati oba pogoja kritičnih delov č1, p2 ob koncu vsakega od kritičnih delov č1,č2,p1,p2 in p3.

4.3. Monitorji in hierarhija monitorjev

Intuitivno smo se s pojmom tajnika (Dijkstra: "secretary") oz. monitorja (glej 5 in 4) seznanili že v 4.1.3. Ugotovili smo, da nek skupek procesov medsebojno komunicira v (pogojnih) kritičnih delih in pri tem uporablja določene skupne informacije (spremenljivke stanj). Pojem monitorja bomo konkretizirali z uvedbo programskega konstrukta monitor, ki bo vseboval lokalne spremenljivke predstavljajoče vse informacije, ki so skupne danemu skupku procesov in pa procedure pred stavljajoče vse kritične dele tega skupka procesov. V samih procesih bodo kritični deli nadomeščeni s klici odgovarjajočih monitorskih procedur.

Programski konstrukt monitor bo imel naslednjo obliko (glej 4):

```
monitor
var v1:t1;... vk:tk;
com skupne informacije procesov com;
shared procedure p1(...); begin ...
end;
...
shared procedure pm(...); begin ...
end;
deklaracija lok.proc.;
inicijalizacija spremenljivk v1,...vk;
end.
```

Vsak od procesov, ki medsebojno komunicirajo preko zgornjega monitorja pa bo imel naslednjo obliko:

```
process
environment p1, ... pn, v1, ... vm;
... ... p1(...); ... pj(...);
com klici monitorskih proc. com;
end;
```

kjer so p1, .. pn deljene procedure monitorja in v1, .. vm skupne informacije monitorja, ki jih dani proces uporablja.

Stvar bomo ilustrirali s primerom čitalcev in piscev iz 4.2. Monitor bo izgledal takole:

```
monitor
var ap, rp, rč: integer;
var včit, vpis:vrsta;
shared procedure začit;com glej 4.2.
com;
shared procedure kočit;com glej 4.2.
com;
shared procedure začpis;com glej 4.2.
com;
shared procedure kopis;com glej 4.2.
com;
procedure unblock-one(vrsta);..;
procedure enter(vrsta);...;
com za skedjuliranje com;
procedure unblock-all(vrsta);..;
ap:=rp:=rč:=0;
end;
```

Čitalni proces bo zapisan:

```
process
environment začit,kočit,
  ap,rp,rč;
...
začit;com monitorski klici com;
čitaj;
kočit;
...
end;
```

pisalni proces pa:

```
process
environment začpis, kopis, ap, rp, rč;
...
začpis;
piši;
kopis;
...
end;
```

Pri tem mora biti implementacija tako urejena, da:
- je v določenem trenutku možna izvedba le ene od shared procedures monitorja (pri tem lahko uporabimo binarne semaforje);
- ob izvedbi klica enter(vrsta) v proceduri začit ali začpis, ki jo kliče proces p, se proces p postavi v "vrsto", nakar so monitorske procedure na razpolago ostalim procesom;
- ob izvedbi klica unblock-one(vrsta) ali unblock-all(vrsta) v proceduri kočit ali kopis, ki jo kliče proces p, se proces p vrne iz monitorske procedure in nadaljuje z izvedbo; obenem pa procesi, ki so na ta način "prebujeni" nadaljujejo z izvedbo inštrukcije, ki sledi inštrukciji (v tem primeru "enter"), s katero so bili "uspavani".

Naslednji primer je par procesov proizvajalec-uporabnik (glej 4.1.2.), ki si pošiljata "sporočila" preko buferja, ki lahko vsebuje samo eno sporočilo (primer je iz 4):

```
proizvajalec:
process
environment pošlji, sporočilo;
var s: sporočilo;
begin .. proizvedi s ; pošlji (s) end;
uporabnik:
process
environment sprejmi,sporočilo;
var r: sporočilo;
begin ... sprejmi (r); uporabi r... end;
monitor:
monitor
type sporočilo = array (1..8) of integer,
var bufer: sporočilo; poln:boolean;
    proizv, upor:vrsta;
```

```
shared procedure pošlji (m:sporočilo);
 begin  if poln then enter proizv;
        bufer:=m; poln:=true;
        unblock-one upor;
 end;
shared procedure sprejmi (m:sporočilo);
 begin  if not poln then enter upor;
        m:=bufer; poln:=false;
        unblock-one proizv;
 end;
inicijalizacija:poln:=false
end monitor;
```

Programski konstrukt monitor nam torej dovoljuje, da v visokem programskem jeziku pregledno opišemo komunikacije med procesi in skedjuliranje procesov.

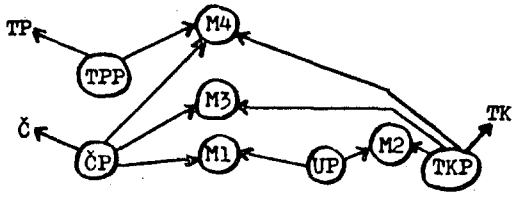
Brinch Hansen (4) je pokazal, da je uporaba monitorskega konstrukta primerna za opis in zgradbo operacijskih sistemov s stališča:

- dokazljivosti pravilnega sodelovanja procesov sistema;
- dokazljivosti določenih lastnosti sistema (npr. odsotnost mrtvih točk);
- učinkovitosti uporabe virov in
- možnosti postopnega izgrajevanja sistema.

Z uporabo konstruktov za pojem procesa in monitorja lahko strukturo danega operacijskega sistema opišemo kot hierarhijo procesov in monitorjev.

Tako bi del nekega operacijskega sistema lahko prikazali z naslednjo strukturo monitorjev, procesov in virov:

naj TP označuje teleprinter, Č čitalnik kartic, TK tiskalnik ("printer"), ČP, UP, TKP in TPP čitalniški, uporabniški, tiskalniški in teleprinterski proces, M1 do M4 pa monitorje:



V zgornjem acikličnem grafu krogi predstavljajo procese in monitorje, puščice pa možne dosege.

Monitor M1, npr., ureja delovanje para procesov ČP, UP tipa proizvajalec-uporabnik, monitor M4 pa ureja delovanje dveh "proizvajalcev" sporočil ČP, TKP in "uporabnika" sporočil TPP oziroma enega "proizvajalca" sporočil TPP in dveh "uporabnikov" sporočil ČP in TKP.

5. PROCESNE POTI

Campbell in Habermann (10) predlagata mehanizem za opisovanje sinhronizacije in komunikacije med procesi, ki se precej razlikuje od pristopa opisanega v poglavju 4. V tem poglavju bomo na kratko opisali ta mehanizem.

Proces je pojmovan kot sekvenčno zaporedje osnovnih dejanj, pri čemer je osnovno dejanje izvedba neke procedure. Opis sinhronizacije procesov je podan s procesnimi potmi. Neka procesna pot je izraz, ki predstavlja množico dovoljenih zaporedij osnovnih dejanj. Sinhronizacija je torej podana na nivoju izvedbe procedure. Glede implementacije tega opisa sinhronizacije se predlaga po en kontrolor za vsako procesno pot: kadar proces želi izvesti neko proceduro r, vsebovano v procesni poti p, potem o izvedbi procedure r odloča kontrolor vezan na procesno pot p. Zaradi enostavnosti kontrolorja se zahteva, da se dana procedura pojavi samo enkrat v samo eni procesni poti.

Za opis procesnih poti so uvedene naslednje oznake (kjer so p,q,r,.. imena procedur):

- sekvenčna izvedba:

  p;q;r

pomeni, da morajo biti procedure izvedene v tem vrstnem redu, ena za drugo. Pri tem ni važna identiteta procesov oz. procesa, ki izvaja(jo) posamezne procedure;

- selektivna izvedba:

  p,q;r

pomeni, da se lahko izvede ena od naštetih procedur: dokler se izvaja npr. q, ne moremo začeti izvajanja procedure p ali r;

- ponavljanje:

  path P end ≡ P;P;P;...

kjer je P procesna pot;

- sočasna izvedba:

  {P} , kjer je P procesna pot

pomeni, da se lahko poti P sočasno izvajajo s strani različnih procesov. Dokler je vsaj ena izvedba poti P v teku, lahko nekinov proces začne z novo izvedbo poti P. Izvedba poti {P} je končana v trenutku, ko so končane vse izvedbe poti P.

Razne načine izvedb lahko kombiniramo z uporabo oklepajev npr.:

  path p,(q; {r} ; s) end.

Konstrukti procesnih poti so kombinirani z naslednjim pojmom oz. konstruktom tipa: definicija tipa opisuje

- strukturo podatka danega tipa in
- operacije nad podatki danega tipa.

Podatke tipa t lahko manipuliramo samo s pomočjo operacij definiranih v definiciji tipa t.

Uporabo procesnih poti kot sinhronizacijskega konstrukta bomo opisali s primerom piscev in čitalcev, ki komunicirajo s pomočjo krožnega buferja.

Element R(i) krožnega buferja R bo opisan z naslednjim tipom:

```
type bufel;
message vsebina;
com edina komponenta neke spremenljivke
    tipa bufel je spremenljivka vsebina
    com;
path piši; čitaj end;
operations
   procedure čitaj (returns message m):
   m:= vsebina;
   procedure piši (accepts message m):
   vsebina:= m;
endtype;
```

Procesna pot path piši; čitaj end pomeni, za dano spremenljivko tipa bufel naslednje:

- vsaki izvedbi "piši" sledi izvedba "čitaj",
- vsaki izvedbi "čitaj" sledi izvedba "piši" in
- izvedbi "piši" in "čitaj" ne moreta biti sočasni.

Krožni bufer bo opisan z naslednjim tipom:

```
type kbuf;
array 0 to N-1 bufel R;
com prva komponenta spremenljivk tipa
kbuf com;
 type kazalec; com lokalna definicija
      tipa com;
 integer P=0; path naslednji end;
 com, kar pomeni za dano spremenljiv-
      ko tipa kazalec, da izvedbe "na-
      slednji" ne morejo biti sočasne
      com;
 operations
 procedure naslednji (returns integer
                      I):
   begin P:=(P+1)mod N; I:=P end;
 endtype;
kazalec zapisanje, začitanje;
com druga in tretja komponenta spre-
    menljivk tipakbuf com;
operations
 procedure pošlji (accepts message m):
   begin integer J; J:= zapisanje.
                        naslednji;
                 R(J).piši(m)
   end;
 procedure sprejmi (returns message m):
   begin integer J; J:= začitanje.
                        naslednji;
                     m:= R(J).čitaj;
   end;
endtype;
```

S temi definicijami lahko uporablja neki krožni bufer KB:

```
kbuf KB;
```

kakršnokoli število čitalnih procesov:

```
process
message M;
M:=KB.sprejmi;  uporabi M
end;
```

in pisalnih procesov:

```
process
message M;
 ustvari M ; KB.pošlji (M)
end;
```

in to v principu na kakršenkoli način, kajti procesne poti opisane v tipih bufel in kazalec skrbijo za pravilno sinhronizacijo.

6. ZAKLJUČEK

Za zaključek lahko rečemo, da obseg recentne literature s področja organizacije medprocesne sinhronizacije in komunikacij kaže na zelo živahen razvoj raziskav, ki se bavijo s to problematiko. Raznolikost pristopov pa med drugim dokazuje, da zadnja beseda na tem področju še dolgo ne bo izrečena.

7. BIBLIOGRAFIJA

(1) Hoare C.A.R.: Towards a theory of parallel programming, in Operating systems techniques, Academic press 1972

(2) Wegner P.: Programming languages, information structures, and machine organization, MCGraw-Hill 1968.

(3) Dijkstra E.W.: Cooperating sequential processes, in Programming languages, ed. Genuys, Academic press 1968.

(4) Brinch-Hansen P.: A programming methodology for OS design, IFIP 1974,2.

(5) ... : Monitors (special discussion), in OS techniques, Academic press, 1972.

(6) Owicki S., Gries D.: An axiomatic proof technique for parallel programs I, Acta informatica v.6, pp 319-40., 1976.

(7) Peterson J.L., Bredt T.H.: A comparison of models of parallel computation. IFIP 1974, 3.

(8) Kahn G.: The semantics of a simple language for parallel programming, IFIP 1974, 3.

(9) Dijkstra E.W.: Hierarchical ordering of sequential processes, in OS techniques, Academic press, 1972.

(10) Campbell R.H., Habermann A.N.: The specification of process synchronization by path expressions, in O.S., Lecture notes on computer science 16, 1974.

(11) Lauer P.E., Campbell R.H.: Formal semantics of a class of high level primitives for coordinating concurrent processes, Acta informatica, v.5.pp. 297-332, 1975.

(12) ... : Algol 68 report, Hermann, 1972.

(13) Dijkstra E.W.: The structure of the THE multiprogramming system, CACM 11, pp. 341-347, 1968.

(14) Brinch-Hansen P.: A comparison of two synchronizing concepts, Acta informatica, v.1., pp. 190-199, 1972.

(15) Schmid H.A.: On the efficient implementation of conditional critical regions and the construction of monitors, Acta informatica, v. 6, pp. 227-250, 1976.

(16) Hoare C.A.R.: An axiomatic basis for computer programming, CACM 12, pp. 576-580, 1969.

(17) Lautenbach K., Schmid H.A.,: Use of Petri nets for proving correctness of concurrent process systems, IFIP 1974, 2.

dinamični mos pomnilniki

r. trobec
j. korenini
f. novak

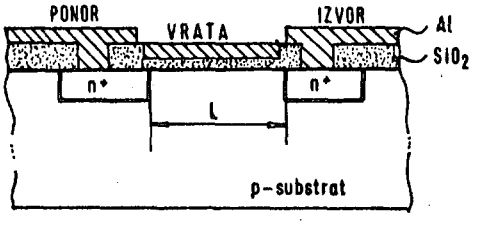
UDK 621.377.622.25

Institut "Jožef Stefan",
Ljubljana

V članku želimo seznaniti bralca z natančnejšim opisom izvedbe in delovanja dinamičnih MOS pomnilnikov. Najprej je opisana tehnologija, ki dopušča veliko gostoto podatkov, sledi opis osnovnih funkcij pomnilnikov (čitanje, pisanje), končno pa so predstavljene sheme važnejših funkcionalnih blokov na nivoju tranzistorjev. Opisane so pomnilne lokacije, ojačevalniki informacije (sense amplifier) in vhodno-izhodni vmesniki (buffer), ki so specifično zasnovani zaradi TTL kompatibilnosti in optimalnega razmerja poraba moči/hitrost.
Na koncu podajamo časovne diagrame za tipične pomnilniške cikle (čitanje, pisanje, način zaporednega vpisovanja (čitanja) v vrstice (page mode), osvežitev informacije (refresh)), s poudarkom na važnih časovnih parametrih.
DYNAMIC MOS MEMORIES The aim of this article is to inform the reader with a detailed description of the architecture and the performance of dynamic memories. At first we describe the technology that permits so much information gathered on a single chip. The description includes basical memory functions (read, write) and some diagrams of important functional blocks on transistor level. Basic memory cell and its environment (sense amplifiers, decoder and I/O buffers necessary for TTL compatibility) is described as well as internal timing and problems of power dissipation and access time. Finally timing diagrams for typical memory cycle (read, write, page mode, refresh) are given and attention is paied to important timing parameters.

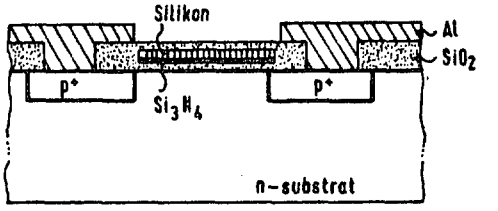
1. Pregled osnovnih tehnologij, ki se uporabljajo pri izdelavi dinamičnih MOS pomnilnikov

Pri izdelavi dinamičnih pomnilnikov se danes zaradi razvoja tehnologije večinoma uporabljajo tranzistorji z n-kanalom. Ti imajo zaradi boljše gibljivosti elektronov večjo hitrost (2x) in manjšo površino kot tranzistorji s p-kanalom. Nova silikonska vrata (gate) vse bolj nadomeščajo metalno elektrodo.



Slika 1. MOS FET+z metalnimi vrati in n-induciranim kanalom.

Če je na vratih pozitiven potencial, se inducira pod njimi prevodni n-kanal, skozi katerega teče tok, odvisen od napetosti med ponorom (drain) in izvorom (source). Za izdelavo so potrebne 4 maske. Ponor in izvor lahko izvedemo z difuzijo ali ionsko implantacijo.



Slika 2. FET s silikonskimi vrati in induciranim p-kanalom.

Bistvena prednost te izvedbe je v nizki napetosti praga, kar je posledica velike dielektričnosti kombinacije plasti SiO_2 - Si_3N_4.
Iz enačbe za pragovno napetost (V_T):

$$V_T = -\frac{\sigma_{ss}\, x_{oks}}{\varepsilon_{oks}} + U_B + 2(V_F - V_{Fi})$$

je neposredno viden vpliv ε_{oksida} na V_T. Tipične vrednosti so 1,5 - 2V. Taka napetost je potrebna za majhno porabo moči in za kompatibilnost s TTL logiko. Tranzistor na sliki 2. ima silikonska vrata.**

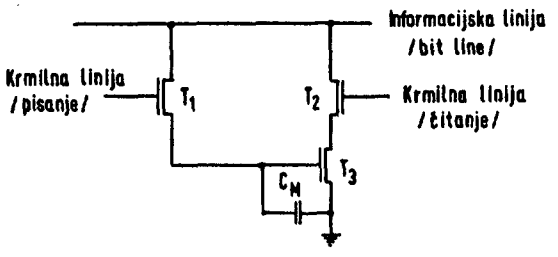
2. Princip delovanja dinamične pomnilne celice.

V pomnilni celici na sliki 3. se shrani naboj na kapacitivnosti med vrati in ponorom tranzistorja T_3.

MOS+ - metal oksid polprevodnik (Metal Oxid Semiconductor)

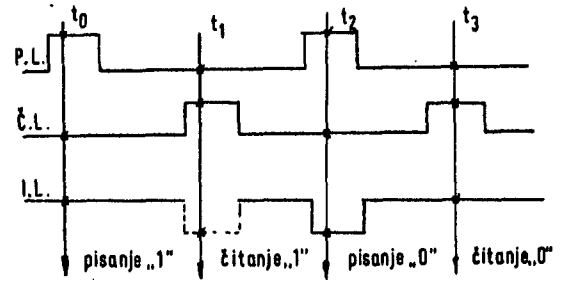
FET - tranzistor, ki deluje zaradi polja v polprevodniku (Field Efect Transistor)

** Poleg metalne in silikonske povezave obstaja še tretja možnost - difundirana povezava. Vsaka ima svoje prednosti in pomanjkljivosti, ki odločajo o specifičnosti njihove uporabe.



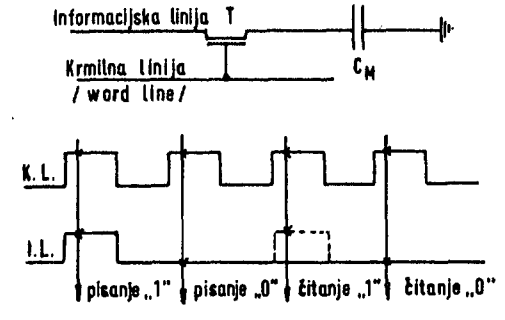
Slika 3. Tritranzistorska pomnilna lokacija.

Pri taki zasnovi služita tranzistorja T_1 in T_2 le kot stikali pri čitanju oziroma pisanju informacije, ki se hrani na kapacitivnosti C_M. Kadarkoli želimo spremeniti naboj (informacijo), je potrebno aktivirati eno ali drugo stikalo (T_1 ali T_2) glede na zahtevano operacijo (pisanje, čitanje). Rezultat se odčita ali spremeni s stanjem na informacijski liniji (bit line) - (slika 4).



Slika 4. Principielni časovni diagram tritranzistorske celice.

Ponuja se ekstremna poenostavitev pomnilne celice. Uporabljen je le en tranzistor (stikalo) in en kondenzator (spominski element).



Slika 5. Enotranzistorska pomnilna lokacija in pripadajoči časovni diagram

Čitanje in pisanje je omogočeno s krmilno linijo (word line), ki odpira vrata tranzistorja. Stanje celice pa sledi stanju informacijske linije (bit line) - (slika 5).
Ta celica ima številne pomanjkljivosti:

- čitanje povzroči izgubo informacije
- hitrost je omejena
- signali so zaradi majhnih dimenzij kondenzatorja zelo majhni;

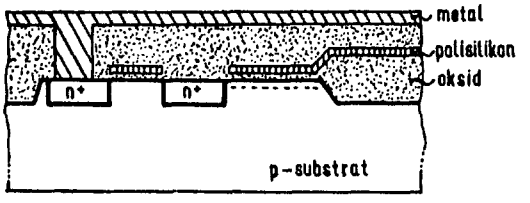
po drugi strani pa je celica:

- enostavna
- zahteva malo povezav (odpade ena linija)
- zavzema malo prostora.

Sodobni dinamični 4 K+ in 16 K bitni pomnilniki so danes že večinoma izvedeni z enotranzistorsko pomnilno celico. Majhne signale je potrebno ojačevati in obnavljati (refresh) vsaki 2 ms s posebnimi ojačevalniki informacije (sense amplifier), o katerih bomo še govorili.

3. Tehnološke izvedbe enotranzistorske pomnilne celice

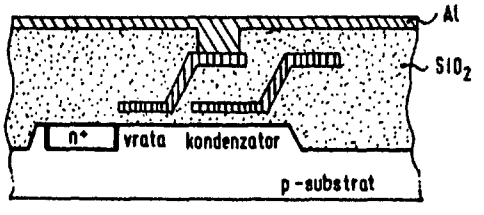
-Prva zelo široko uporabljena izvedba je tehnologija z eno plastjo silikona (single-level polysilicon), ki potrebuje pet standardnih mask.+



Slika 6. Prerez pomnilne celice, izvedene z eno plastjo silikona.

Električna shema je predstavljena na sliki 5. Tranzistor s silikonskimi vrati služi kot stikalo, drugi del silikona (v isti plasti) pa je uporabljen za kondenzator, ki hrani naboj. Tipične velikosti pomnilnih celic, ki jih dobimo na ta način, so 800 μm^2, kar dopušča izvedbe 4 K in celo 16 K pomnilnikov. Primeri so pomnilniki: MOSTEK MK 4096, MK 4027, oba 4 K in TI 4070 firme Texas Instruments, ki je primer 16 K pomnilnika (tako velikost so dosegli z racionalnejšo razporeditvijo, manjšim kondenzatorjem in kontaktom).

- Tehnologija, ki je resnično odprla pot do 16 K pomnilnikov, je izvedba z dvojno plastjo silikona (double-level polysilicon). Osvojili so jo že vsi vodilni proizvajalci pomnilnikov. Postopek omogoča manjše pomnilne celice, ker je kondenzator, ki hrani informacijo o stanju pod priključkom za preklopni tranzistor. Prva silikonska plast je uporabljena za spominski kondenzator, druga pa za vrata preklopnega tranzistorja. Dodatna prednost je tudi v tem, ker je pri tej izvedbi informacijska linija difundirana. S tem pa se zmanjšajo parazitne kapacitivnosti in število povezav. Na površini ostane le še krmilna linija, ki pa se lahko uporabi za dve sosednji celici. Po drugi strani pa je ta tehnologija bolj kompliciran. Potrebni sta dve dodatni maski (skupaj 7 mask: difuzija za izvor in ponor, epitaksijska plast, obe silikonski plasti, kontakti in metal).

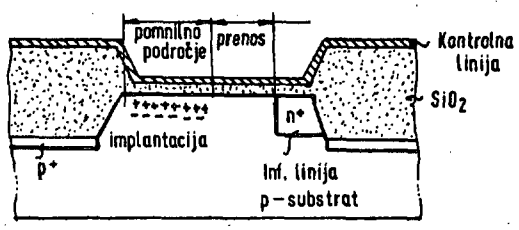


Slika 7. Prerez pomnilne celice, izvedene z dvema plastema silikona

+ 4 K bit - 4.2^{10} = 4096 pomnilnih celic
+ maske določajo aktivni del substrata, na katerem se izvršujejo posamezni tehnološki postopki.

Primer 16 K bitnega pomnilnika, izvedenega s tem tehnološkim postopkom je MOSTEK MK 4116.

- Pokažimo še zamisel celice, s katero bi bilo možno realizirati 64 K bitne pomnilnike. Za tak pomnilnik je potrebna celica velikosti 100 -200 μm^2, ki jo je mogoče izvesti s CCD (charge-coupled) tehnologijo.
Taka celica (slika 8.) nima tranzistorja ampak le preklopno kondenzatorsko področje, ki je pod CCD vrati. Ta zelo kompaktna celica je hitra kot posamezni tranzistor. Za shranjevanje naboja je dovolj prostora, potrebni sta le dve liniji (krmilna in informacijska linija).



Slika 8. Prerez pomnilne celice, izvedene s CCD tehnologijo

Oba nivoja informacije sta dobljena z ionsko implantacijo. Podatek je spravljen v enem od nivojev. Aktivna krmilna linija odpira vrata, ki omogočajo pretok informacije v obe smeri (čitanje in pisanje). Informacijska linija pa določa stanje celice.

4. Preostala vezja v pomnilniku

Večino površine aktivnega silikona zavzema matrika pomnilnih celic. Najboljše razmerje poraba moči/hitrost in zahteve po odčitavanju majhnih signalov, narekujejo simetrično zgradbo pomnilne matrike. Med obema polovicama so ojačevalniki informacije (sense amplifier) in vhodno-izhodne linije, ki vodijo od vsakega ojačevalnika do vhodno-izhodnih vmesnikov (buffer).
- Ojačevalniki informacije so dinamično aktivni tako, da se ne troši nobena enosmerna moč. Zaradi take zasnove so podatki v eni polovici pomnilnih lokacij invertirani, kar pa je internega značaja, ker dobijo pri izpisu zopet pravo vrednost.
V pomnilni matriki najdemo še posebne celice (dummy cell), ki dajejo pri detekciji signala referenčne napetosti za logično "1" in "0".
Pri 4 K bitnem pomnilniku je na vsako stran ojačevalnika informacije vezano 32 pomnilnih lokacij in ena referenčna celica (slika 9).
Željena lokacija se izbere preko krmilne linije za dostop do lokacije (word line), ki vodi iz dekoderja. Celotna vrstica se prenese v ojačevalnike informacije, kjer povzroči naboj iz pomnilnega kondenzatorja prehod le-teh v nestabilno stanje, ki je določeno z diferenco tokov I_s in $I_{\bar{s}}$ skozi tranzistorja T_s in $T_{\bar{s}}$. Ta dva toka pa sta odvisna od diference nivojev v pomnilniški lokaciji in referenčni celici.
Zaradi omejenih dimenzij kondenzatorja je signal, ki ga je potrebno zaznati, zelo majhen (200 mV). Za detektiranje tako majhnih signalov je potrebna velika občutljivost diferencialnih ojačevalnikov. Če vzamemo za merilo občutljivosti:

$$S/_{\text{ob času delovanja}} = \frac{I_s - I_{\bar{s}}}{I_s}$$

in

$$I_s = \frac{C_o \mu_n Z}{L} (V_{DS} - V_T)^2 = K (V_s - V_o - V_T)^2$$

kjer je K odvisna od tehnologije in velikosti tranzistorja (dolžina in širina kanala), V_T pa je napetost praga, ki je za določeno tehnologijo konstanta, sledi

$$S = \frac{K(V_s - V_o - V_T)^2 - K(V_{\bar{s}} - V_o - V_T)^2}{K(V_s - V_o - V_T)^2} = \frac{2(V_s - V_{\bar{s}})}{(V_s - V_o - V_T)}.$$

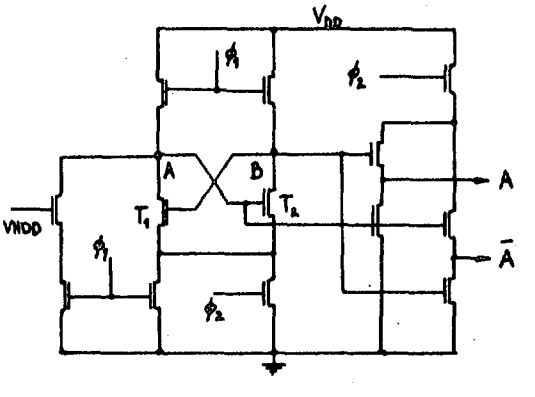
Vidimo, da je občutljivost večja, čim večja je napetost V_o, oziroma čim manjša je napetost V_{DS}. Isti pogoj ustreza tudi minimalni porabi moči, saj je v prvi aproksimaciji moč:

$$I_S V_{DD} = P_S = V_{DD} K(V_s - V_o - V_T)^2 .$$

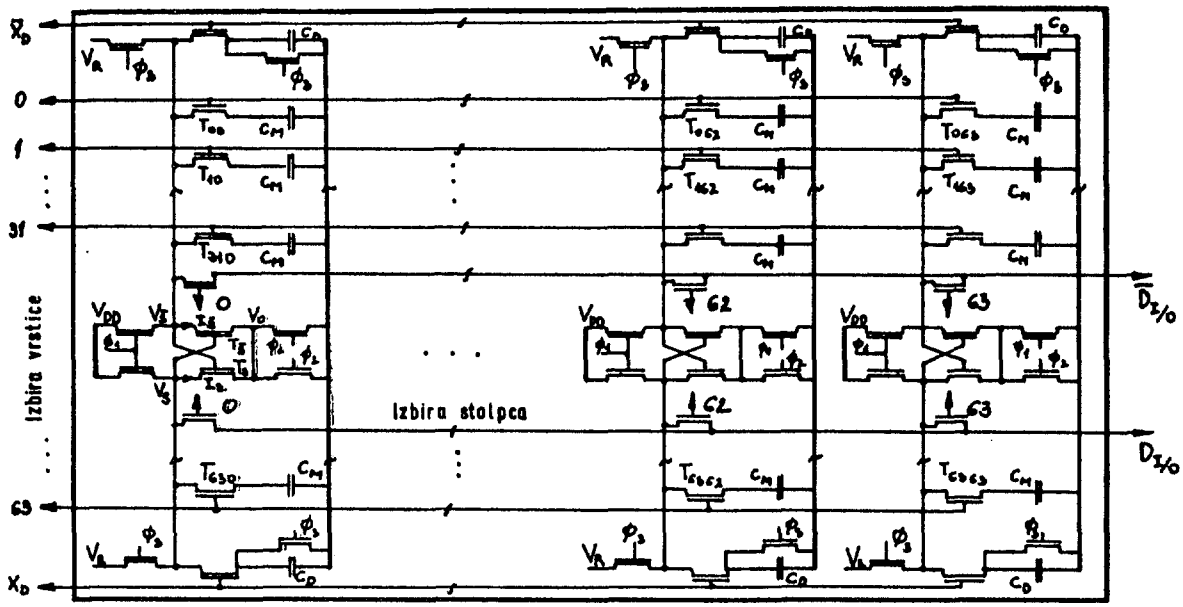
Zaradi navedenih vzrokov tranzistor T_2 često spremljajo še paralelni tranzistorji (multigrounding path), s katerimi se doseže višja napetost V_o. Med odčitavanjem je aktivirana le ena pot. Pš določeni zakasnitvi, ko je stanje ojačevalnika informacije definirano, pa se aktivirajo še druge poti, ki znižajo V_o in jasno določijo izhod ojačevalnika.
- Za pravilno delovanje dinamičnega pomnilnika je potrebna množica internih signalov, ki jih dobimo s krmilnimi in urinimi vezji. Na teh vezjih se generirajo določene zakasnitve za zunanjimi signali, ki krmilijo pomnilnik:
$\overline{RAS}$, $\overline{CAS}$, $\overline{CS}$, $\overline{WRITE}$. Njihovo vlogo bomo še opisali.

- Dekoder je standarden, ima pa dodatne novosti, ki jih zahteva moderna arhitektura dinamičnega pomnilnika (RAM). Uporabljen je le en dekoder za dekodiranje vrstic in stolpcev. Adresiranje je razdeljeno (časovno multipleksirano) tako, da se ob prvem signalu $\overline{RAS}$ (Row address select) izbere preko dekoderja odgovarjajoča vrstica v pomnilni matriki. Nato pa se ista vezja ob nastopu zunanjega signala $\overline{CAS}$ (Column address select) uporabijo za izbiro zahtevanega stolpca (na križišču stolpca in vrstice je željena pomnilna lokacija).

- Vhodni vmesniki (buffer) sprejemajo adreso in druge krmilne signale s standardnim TTL nivojem. Zato je potrebna posebna zgradba teh vezij (slika 10.), ki omogočajo enostavno uporabo dinamičnih MOS pomnilnikov v standardnih sistemih.



Slika 10. Vhodni vmesnik dinamičnega pomnilnika



Slika 9. Shematični prikaz pomnilniške matrike

Kanala tranzistorjev T_1 in T_2 imata različne dimenzije. To omogoča detekcijo logične "1" oziroma "0". Če je na vhodu logični nivo "1", dobi točka A potencial mase, kar določi stanje bistabilnega vezja. Ob vhodnem nivoju "0" pa je stanje določeno z razliko v velikosti kanalov. Kanal tranzistorja T_2 je večji (tok čez tranzistor je odvisen od dimenzij), zato se T_2 odpre in točka B dobi potencial mase. Preostali del vezja služi za invertiranje signalov, ker dekoder potrebuje tudi inverzne vrednosti.

- Pomembno vlogo v pomnilniku imajo tudi izhodni ojačevalniki, ki detektirajo in ojačijo informacije iz dinamičnega pomnilnika. Njihova hitrost je odločilnega pomena pri skupnem času dosega (access time) celotnega pomnilnika. Osnova izhodnega ojačevalnika je podobno kot pri ojačevalniku informacije (sense amplifier) balansirani dvojček (flip-flop) z razliko, da je tu smer preklopa določena s potencialom na vratih bremenskih tranzistorjev.

- Edino vezje, ki stalno troši moč, je vhodna stopnja za prvi signal $\overline{RAS}$, s katerim se začne pomnilni cikel. To vezje pretvori TTL (5V) v MOS (12V) nivo in s tem naznači začetek za kakršno koli aktivnost pomnilnika.

5. Funkcionalni opis delovanja dinamičnega pomnilnika

Zunanji signal $\overline{RAS}$ označi začetek vsakega pomnilnega cikla. $\overline{RAS}$ se na MOS nivo prilagodi in generira več internih urinih signalov, ki pripravijo pomnilnik za nadaljnje operacije. Prvi signal aktivira vhodni register za prvo polovico adres. Po dekodiranju in izbiri željene vrstice v pomnilni matriki se aktivirajo še referenčne (dummy) celice na nasprotni polovici pomnilnika. Vhodni register se resetira. Zadnji signal v prvi sekvenci pa aktivira še ojačevalnike informacij, ki se postavijo v pravilna stanja in s tem osvežijo (refresh) vsebino celotne vrstice.
Medtem je potrebno zunaj pomnilnika z dodatnim multipleksorjem pripeljati na vhod drugi del adres, ki bodo izbrale zahtevani stolpec. Drugi zunanji signal $\overline{CAS}$ resetira izhodni register in povzorči visoko impedanco na izhodu (prekine povezavo med pomnilnikom in zunanjim svetom). Zopet se aktivira zaporedje urinih impulzov, ki povežejo zahtevano pomnilno celico z vhodno/izhodno linijo.
Zadnji zunanji signal $\overline{CS}$ (chip select) izbere del pomnilnika, na katerem se bo izvršilo čitanje ali pisanje. Če tega signala ni, se nadaljnje operacije v pomnilniku onemogočijo. Do te časovne točke je torej aktiven celoten pomnilnik. Zato dekodiranje $\overline{CS}$, ki izbere željeni del pomnilnika ne vpliva na skupni čas dosega.
Signal $\overline{WRITE}$ izbira med čitanjem ali pisanjem v pomnilno lokacijo.
Če gre za pisanje, se sproži paralelno zaporedje signalov, ki resetira in aktivira vhodne vmesnike in sprejme podatke, kateri se preko ojačevalnikov informacije (sense amplifier) zapišejo v izbrano lokacijo na križišču stolpca in vrstice.
Pri čitanju je smer pretoka podatkov obratna. Ko je na izhodni liniji prava vrednost, jo izhodni ojačevalnik sprejme in po določeni zakasnitvi prezentira zunanjemu svetu.
Konec cikla označuje neaktivni $\overline{RAS}$ ne glede na stanje $\overline{CAS}$.

- Poleg navedenih običajnih operacij je možno pri sodobnih pomnilnikih še zaporedno pisanje (čitanje) v isto vrstico. To je izvedljivo zato, ker $\overline{CAS}$ po končani operaciji inicializira vhodni vmesni register in pri tem ne vpliva na izbrano vrstico, ki je določena s stanji ojačevalnikov informacij, kar omogoča ponovni $\overline{CAS}$ cikel (page mode).

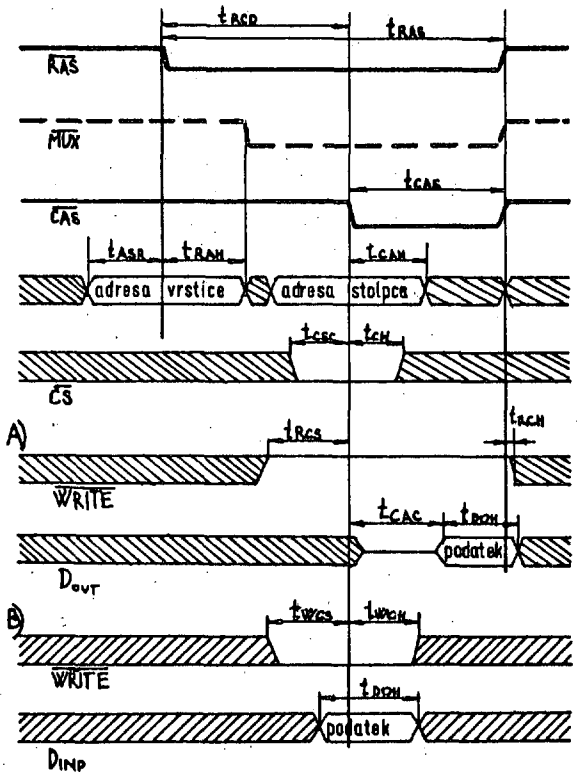
- V določenih časovnih intervalih obnavljamo vsebino pomnilnika z aktivnim $\overline{RAS}$, ki je edino potreben pri tej operaciji. Novejši mikroračunalniki (Z-80) interno generirajo ta signal, tako da v tak sistem lahko brez dodatnih logičnih vezij vključimo velike pomnilnike.

+ RAM (Random Access Memory) - ta izraz pomeni, da je možno ob vsakem trenutku adresirati katerokoli pomnilno lokacijo.

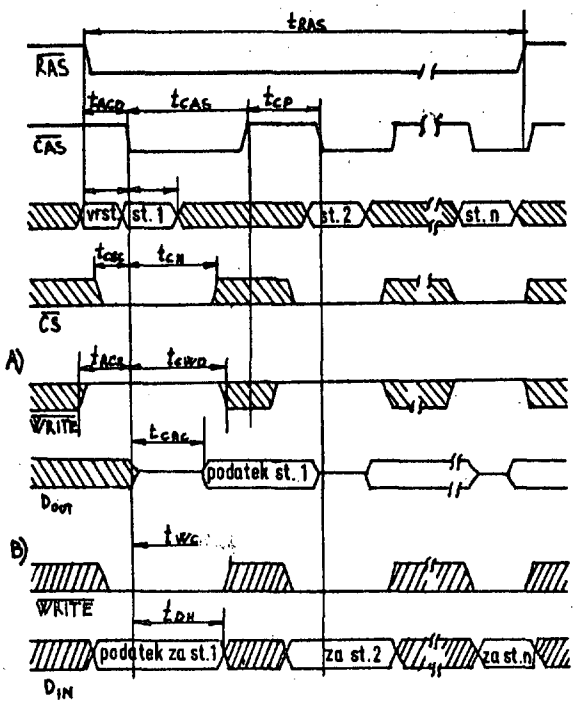
Pomnilnik se osvežuje v času dekodiranja operacijske kode instrukcije (to je v dostavnem ciklu mikroračunalnika), tako da se zaporedoma obnavljajo vsebine celih vrstic. Za ta proces napetost V_{cc} (+5V) ni potrebna.
Če želimo, da nam pomnilnik opravlja samo funkcijo ohranjevanja informacije, lahko napetost V_{cc} izključimo.

6. Časovni diagrami za dinamični pomnilnik

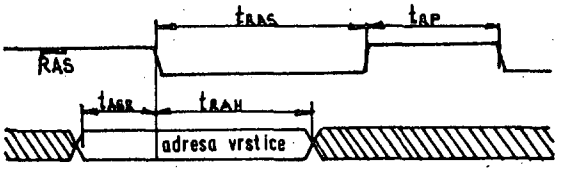
Za konec poglejmo še tipične časovne diagrame, ki so potrebni za pravilno delovanje pomnilnika. To poglavje naj nam približa in uskladi prejšnje opise, zato jih podajamo brez dodatnih pojasnil. V časovne diagrame so vnešeni le važnejši časovni intervali. Natančne opise diagramov pa je mogoče najti v priročnikih za dinamične pomnilnike.



Časovni diagram I : A) čitanje B) pisanje



Časovni diagram II : A) zaporedno čitanje
B) zaporedno pisanje



Časovni diagram III : Osvežitveni cikel

7. Zaključek

Razumevanje delovanja dinamičnih MOS pomnilnikov je v pomoč vsakemu načrtovalcu, predvsem pa tistemu, ki se ukvarja s posebnimi načini uporabe pomnilnikov (testiranje, DMA, itd...). V članku je podana osnova, ki je potrebna za dobro projektiranje in razumevanje delovanja dinamičnih pomnilnikov.

8. Literatura

1. C. Kuo, N. Kitagawa, D. Ogden, J. Hewkin: 16-K RAM built with proven process may offer high start-up reliability, Electronics, Maj 1976, str. 81-86
2. G. Landers: Choosing among 4-K MOS RAM's, Electronic Design, Junij 1976, str. 138-142
3. Semiconductor Memories (edited by David A. Hodges), Port III, str. 69-146
4. MOSTEK 1977 Memory products catalog, Copyright 1977 by Mostek Corporation.

linking fortran and assembly language programs

hervé tireford

Motorola Inc.

1211 Geneve 20, Switzerland

UDK 681.3.06

This article describes three methods of solution of program linkage for programs written in Fortran and in assembly language.Such problems arise when control programs for peripherals (A/D converters,line printers,etc.) are to be written.These programs are usually written in an assembly language.This article gives a method of calling a Fortran program from an assembly language program,two methods of calling an assembly routine from a Fortran program are described.Besides,the way of data (arguments) exchange between the two types of program is explained.

POVEZAVA FORTRANSKIH IN ZBIRNIH PROGRAMOV. V članku so opisane tri metode za reševanje problema povezovanja programov,zapisanih v Fortranu in v zbirnem jeziku.S takimi problemi se srečujemo,ko moramo sami pisati kontrolne programe za periferne naprave (A/D pretvornike,vrstične tiskalnike ipd.).Te programe pišemo običajno v zbirnem jeziku. Opisana je metoda klicanja fortranskega programa iz programa napisanega v zbirnem jeziku in dve metodi klicanja zbirnih rutin iz fortranskega programa.Poleg tega je opisan način izmenjave podatkov(argumentov)med obema vrstama programov.

1. PROBLEM DEFINITION

Since the Fortran language is not always suited to handle the many tasks to be performed in a program, it is often desirable to use one or several assembly language routines to implement them. This is especially the case when dealing with peripherals whose drivers are not included in the compiler (peripherals other than the console, line printer, disk). The user is then required to write his own assembly program which must make provision for initializing the I/O adapters specific for this application and for exchanging data through them. Furthermore, in order to keep consistent with the data types defined in the M6800 Fortran (16-bit integers and 32-bit real numbers), the assembly routine is also required to format the data exchanged with the peripherals. Therefore, the linkage implies that a technic be used which, on one hand, allows for one program to be called by the other, and on the other hand, which permits the mutual exchange of data (arguments) between each other.

This paper describes three solutions which may be used to cope with the linkage problem; two of them explain how to call for an assembly routine from a Fortran program, while the third solution presents how to face the inverse situation, i.e. how to call for a Fortran program from an assembly language calling program. Due to the fact that one of the problems involved in the linkage is to make provision for formatting data, it looks useful to recall the data structures which agree to the computer.

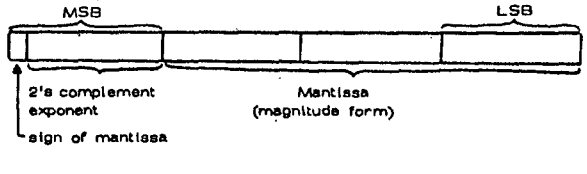
2. FORTRAN DATA TYPES

The M6800 Fortran compiler handles two types of numbers:

- integers.
- real or floating-point numbers.

Each data is implicitly defined as an integer or as a real depending on the first character of the symbolic name attached to it. By convention, any symbol starting with I, K, J, L, M, or N defines the data it represents as an integer which consequently is internally stored as a 16-bit quantity in the 2's complement form.

Any other symbol defines a data as being a real number which is stored on 4 bytes according to the floating-point representation illustrated below (note that the compiler makes use of the floating-point math package listed in program No. 26 of the User Group Library).



Floating-point representation

The base being used is hexadecimal, hence any real number X may be expressed as a equal to:

$$X = (+/-\cdot X_1X_2X_3X_4X_5X_6)_{16}16^{E}$$

where X stands for an hexadecimal digit and E for the signed exponent of X. The actual value represented is:

$$X = +/-\,(X_1{\cdot}16^{-1}+X_2{\cdot}16^{-2}+X_3{\cdot}16^{-3}+X_4{\cdot}16^{-4}+X_5{\cdot}16^{-5}+X_6{\cdot}16^{-6})\cdot 16^{E}$$

3. CALLING AN ASSEMBLY ROUTINE FROM A FORTRAN PROGRAM.

This method utilizes the Fortran "CALL" statement to call for the assembly language routine from a Fortran calling program unit (main program or subprogram); consequently, the assembly segment is regarded as a subroutine and must terminate with the RTS instruction (Return from Subroutine).

The arguments (data needed to the assembly program or returned by it) may be passed in either of two ways.

3.1. Passing the arguments in the argument list

The general form of the CALL statement is as follows:

CALL NAME (Arg.# 1, Arg.# 2, ... Arg.#n)

where NAME is the label used in the label field of the first instruction in the PSCT (program section) of the assembly language program, and Arg.# i are the actual arguments needed to or returned by the assembly subprogram. This latter has the following structure:

```
            NAM     FILE
            OPT     REL
            XDEF    NAME
            XREF    RUN
            DSCT
LABEL 1     RMB 2
LABEL 2     RMB 2
LABEL N     RMB 2
              :
optional reservation in this section for the temporary storage of
data, if required
              :
            PSCT
              :
NAME        JSR     RUN+15
            FDB     LABEL 1
            FDB     LABEL 2
              :       :
```

```
        FDB     LABEL N
        FDB     0
          :
instructions to perform the desired task and to pass arguments
          :
        RTS
        END
```

The rules which apply are listed below:

1. The assembly routine is to be processed by the macro-assembler since the generation of a relocatable object code is required (OPT REL).

2. The programmer must indicate to the assembler that the label NAME is defined within the assembly segment so that it may be referenced in the Fortran module (XDEF NAME).

3. The RUN package of the Fortran library being used during the execution of the final program, one must indicate to the assembler that this package is external to the assembly segment (XREF RUN).

4. The programmer must, in the data section (DSCT), reserve two bytes per argument, that will hold the address of the argument at the time of execution (and not the argument itself).

5. The starting address of the program section (PSCT) must be identified with a label which is the same as the one used as subroutine name in the Fortran calling statement (NAME).

6. Right after the JSR RUN+15 instruction, the programmer must reserve and initialize (FDB LABEL 1,..., LABEL N) two bytes per argument to indicate where the address of the argument may be found. The list ends with an FDB 0 directive.

Appendix F of the M68FTN resident Fortran manual shows an example where this method is utilized.

3.2. Passing the arguments via the common section.

Declaring an array by the COMMON statement causes the compiler to reserve the corresponding amount of bytes in the common section (CSCT). A straight forward fashion of getting hold of the arguments in the assembly language subroutine is thus to provide, within the assembly module, a CSCT directive to direct the macro-assembler to reserve the same amount of bytes as the one needed in the Fortran calling program. It should be noted that, this time, the reservation applies to the arguments themselves and not to their address as previously (see 3.1). Therefore 2 bytes will be reserved for integer, whereas 4 bytes are needed to code an argument defined as real in the Fortran section.

The general form of the calling sequence is as follows:

```
        COMMON A, N, ITEM(2), REAL(2)
          :
        CALL ASSEM
          :
```

Where ASSEM is the label used in the label field of the first instruction in the PSCT of the assembly language subroutine and A, N, ITEM (2), REAL (2) are the arguments that may be shared among the assembly and the Fortran modules. Assuming that all of them are shared between the two modules, the assembly subroutine will be written as indicated below:

```
        NAM     FILE
        OPT     REL
        XDEF    ASSEM
        DSCT

optional reservation if required within this module

        CSCT
A       RMB 4       real argument
N       RMB 2       integer argument
ITEM    RMB 4       2-integer table
REAL    RMB 8       2-real table

        PSCT

ASSEM

        Get or return arguments
        and perform the required
        task

        RTS
        END
```

As an example, consider an application in which 10 16-bit values are read from a counter connected to a PIA and processed by the Fortran section which, for instance, stores them in a table to be sorted later on.

The two modules may be written as follows:

```
        COMMON ICOUNT
        DIMENSION ITEM (10)
        DO 10 I = 1,10
        CALL COUNTR
10      ITEM (I) = ICOUNT
          :
        sort routine
          :
```

Fortran calling program

```
        NAM ASMBLY
        OPT REL
        XDEF COUNTR
PIADA   EQU  $ 8004
PIACA   EQU  $ 8005
PIADB   EQU  $ 8006
PIACB   EQU  $ 8007
        CSCT
ICOUNT  RMB 2
        PSCT
COUNTR  LDA A #4          initialize PIA
        STA A PIACA
        STA A PIACB
        LDA A PIADA       read MSB
        LDA B PIADB       read LSB
        STA A ICOUNT
        STA B ICOUNT + 1
        RTS
        END
```

assembly language subroutine

4. CALLING A FORTRAN SUBROUTINE FROM AN ASSEMBLY LANGUAGE PROGRAM.

The arguments, if needed, are again passed via the common section, the Fortran program unit is a subroutine.

The general forms of the modules are the following:

a) Assembly language program:

```
        NAM PGM
        OPT REL
        XREF FORSUB
        CSCT
    optional reservation if required
        PSCT
START   LDS  STACK
    put argument if any in CSCT
        JSR FORSUB
    retrieve arguments if any in CSCT
          :
        END
```

b) Fortran subroutine:

```
        SUBROUTINE FORSUB

        COMMON -, -, -,     (declare arguments involved
                             in linkage)
        RETURN
        END
```

As an example consider a program in which the assembly language main routine reads a 16-bit value from a PIA and calls for a Fortran subroutine which prints out the message "OVERFLOW" whenever the data just read is greater than a reference value.

A. Assembly language program:

```
        NAM PGM
        OPT REL
        XREF TEST

PIADA   EQU $ 8004
PIACA   EQU $ 8005
PIADB   EQU $ 8006
PIACB   EQU $ 8007

        CSCT
ITEM    RMB  2
        DSCT
        RMB  40
STACK   EQU  *
        PSCT
START   LDS  STACK
        LDA A#4
        STA A PIACA
        STA A PIACB
LOOP    LDA A PIADA
        LDA B PIADB
        STA A ITEM
        STA B ITEM + 1
        JSR TEST
        BRA LOOP
        END
```

b) Fortran subroutine

```
        SUBROUTINE TEST
        COMMON ITEM
        IF (ITEM - 5000) 1, 2, 2
2       PRINT 9
9       FORMAT ('OVERFLOW')
1       RETURN
        END
```

fonction de tri croissant et de tri decroissant apl mitra 15

d. davčev

UDK 681.327

Elektrotehnički fakultet ,
91000 Skopje

Le but de cet article est de présenter la réalisation d'opérateurs de tri croissant et de tri décroissant APL Mitra 15, dans les conditions d'une mémoire virtuelle implantée sur un petit calculateur (32K mots-CII-Mitra 15).Nous avons utilisé un algorithme de tri interne.Les mesures qui caractérisent les opérateurs de tri croissant et de tri décroissant sont résumées sur la fig.9 et 10.

FUNKCIJA NA RASTEČKO I NAMALUVAČKO SORTIRANJE APL MITRA 15: Celta na ovoj trud e da ja opiše realizacijata na operatorite na rastečko i namaluvačko sortiranje APL Mitra 15 ,vo uslovi na virtuelna memorija implantirana na eden mal kalkulator (32K zbora-CII-Mitra 15).Pri toa e upotreben eden algoritam na interno sortiranje.Merenjata koi gi karakteriziraat operatorite na rastečko i namaluvačko sortiranje se rezimirani na sl.9 i 10.

INTRODUCTION

Si X est un vecteur, Z←⍋X représente un vecteur d'indices de la même dimension que X. Les valeurs du vecteur Z sont des indices des éléments de X dans un ordre croissant. Soit ,par exemple :

```
    X←2.3 -5 8 1 100
    Z←⍋X
    Z
2 4 1 3 5
```

Si X contient des éléments identiques, les indices de ces éléments sont déterminés selon la position de ces éléments dans X. Par exemple :

```
     X←2 3 2 8
    ⍋X
1 3 2 4
```

La fonction de tri décroissant (⍒) fait un classement dans un ordre descendant.Dans l'exemple précédent, on a :

```
   ⍒X
4 2 1 3
```

Donc, le résultat de ces opérateurs représente toujours une permutation dont les éléments sont des indices des éléments de l'opérande droit.
Puisqu'on travaille dans les conditions d'une mémoire virtuelle, nous allons utiliser un algorithme de tri interne qui doit avoir les caractéristiques suivantes :
1.La seule variable APL construite qui sera utilisée pour toutes les transformations doit être la variable du résultat.
2.Le nombre de comparaisons doit être minimal
3.L'opérande droit doit être sauvegardé sans aucun changement de ses éléments.
4.L'ordre original des éléments identiques doit être préservé.
5.Le nombre de défauts de pages doit être minimal.

Brawn et Gustavson (B.3.,B.4.) ont montré qu'un programme de tri doit avoir le plus petit "Working set" (D.3.,D.4.) possible , pour obtenir des performances satisfaisantes dans une mémoire virtuelle.La fusion de la liste d'indices peut satisfaire les exigences qui ont été posées pour la réalisation d'opérateurs de tri croissant et de tri décroissant. Tous les échanges sont effectués dans la liste, l'ordre original des éléments est préservé et l'espace supplémentaire utilisé est égal à n + 2 ,ce qui est le minimum possible.

Après avoir obtenu la liste d'indices, il faut résoudre le problème inverse d'une permutation.Nous montrerons un algorithme qui fait l'inverse d'une permutation avec la même variable.Le temps d'exécution de cet algo - rithme est proportionnel à "n".

Pour éviter le plus grand nombre de défaut de pages, on va transférer l'opérande droit et ses pages sur disque car on n'a plus besoin de cette variable : toutes les transformations seront effectuées sur la variable du résultat.

L'ALGORITHME

La première partie du programme contient un sous-programme qui va déterminer les parcours qui existent dans l'opérande droit (séquences dans l'ordre croissant).De cette manière, il va fixer le nombre de séquences pour la fusion.Si les éléments sont déjà dans un ordre croissant (par exemple un J-vecteur APL), la sortie du programme est immédiate (il n'y a pas de séquences à fusionner).

Soit (L_i) ; i = 0,1,2,...,N,N + 1 l'ensemble des éléments de la liste qui sera construite sur la variable du résultat ;$(\mathcal{E}_i)$; i = 1,2, ...,N l'ensemble des éléments à trier de l'opérande droit.
La liste est obtenue de la manière suivante :

Si $\mathcal{E}_i \leq \mathcal{E}_{i+1} \Rightarrow L_i \leftarrow i+1$

Si $\mathcal{E}_i > \mathcal{E}_{i+1}$ on a la fin d'un parcours;on va stocker dans L_i une valeur négative dont la valeur absolue va déterminer la tête du parcours suivant.On va constituer deux listes d'éléments dont la fin est marquée par un zéro, chaque liste contenant plusieurs parcours qui finissent par un pointeur négatif.Le sous programme utilise deux pointeurs p et t.Le pointeur t détermine la fin de chaque parcours, le pointeur p passe la liste entière.L_1 et L_{N+1} contiennent les têtes de deux listes dont les parcours seront fusionnés.Soit : $N \geq 2$.
L'organigramme de ce programme est présenté sur la fig.**1**.
Soit, par exemple : $\mathcal{E} \leftarrow$? 20 ; 100

49 75 35 37 3 85 74 30 47 35 40 21 31 50 20 1

83 21 58 99

Le résultat du programme précédent est présenté sur la fig.**2**.
On a obtenu deux listes de quatre parcours qui seront fusionnés entre eux :

Première liste		Deuxième liste
1,2	↔	3,4
5,6	↔	7
8,9	↔	10,11
12,13,14	↔	15
16,17,0	↔	18,19,20,0

L'étape suivante sera la fusion de ces parcours.L'échange sera effectuée avec les éléments de la liste, en fonction de la fusion des parcours.L'organigramme de ce programme est présenté sur la fig.**4**.Il utilise quatre pointeurs p,t,s et q.Les pointeurs p et q pointent sur les deux parcours qui sont en cours de fusion.
Dans notre exemple le développement est présenté sur la fig.**3**
Après L_1 on a les parcours suivants pour la fusion :

Première liste		Deuxième liste
3,4,1,2	⟷	5,7,6
8,10,11,9	⟷	15,12,13,14,0
16,18,19,17,20,0		

L_1 représente les deux listes qui doivent être fusionnées

5,15,12,8,13,3,10,4,11,9,1,14,7,2,6,0 ↔ 16,

18,19,17,20,0

Maintenant, il faut tracer la chaîne du début jusqu'à la fin en remplaçant les pointeurs par leurs positions relatives dans la liste (le premier élément de la liste est remplacé par 1 ,le deuxième par 2, etc.).Le résultat est une permutation P .Il faut réaliser aussi l'inverse de cette permutation.Pour obtenir cet inverse, on a besoin de mettre des valeurs négatives au lieu des valeurs positives dans tous les cas où $i \neq P_i$.
L'organigramme de ce programme est présenté sur la fig.**6**.
Dans notre exemple ,on va obtenir la permutation P_1 (fig. **5**.).
Donc,on a obtenu une permutation P telle que

$$\mathcal{E}'(P) = \mathcal{E} \qquad (1)$$

où $\mathcal{E}'$ contient les éléments de $\mathcal{E}$ dans l'ordre croissant.Il faut obtenir une permutation P' telle que :

$$\mathcal{E}(P') = \mathcal{E}' \qquad (2)$$

Soit ivP' l'inverse de P' ,où P'(ivP') représente une permutation identique (1,2,3,... etc).Alors, $\mathcal{E}'$(ivP') = $\mathcal{E}$ .Puisque nous avons trouvé une permutation P telle que $\mathcal{E}'(P) = \mathcal{E}$, la relation (2) sera satisfaite si P' = ivP.
Donc, il faut réaliser l'inverse du résultat actuel.Les valeurs négatives vont indiquer les composantes des cycles qui ne sont pas encore inversés.Dans tous les cas où $i = P_i$ il n'y aura aucun changement.L'organigramme de cet algorithme est présenté sur la fig.**8**.
Dans notre exemple on va obtenir le résultat qui est présenté sur la fig.**7**.
On a obtenu une liste d'indices d'éléments de l'opérande droit dans l'ordre croissant :

16,5,15,12,18,8,13,3,10,4,11,9,1,14,19,7,2,17,6,20

Pour éliminer les premiers (i=0) et dernier (i=N+1) éléments de cette liste, on a modifié le descripteur (voir D.2.) de l'opérande droit

NBEL,RVEC ⟵ N

ABASE ⟵ 1

Le même programme a été utilisé pour l'opérateur de tri décroissant ; son résultat est le "miroir" du résultat du tri croissant , sauf pour les éléments égaux, qui doivent rester dans leurs positions originales: pour cela, on fait un test de l'opérateur avant la comparaison des éléments de l'opérande droit.Pour le tri décroissant, à la fin du programme, le descripteur de l'opérande droit est modifié de la manière suivante :

DEL ⟵ -DEL

ABASE ⟵ N

Les mesures qui caractérisent les opérateurs de tri croissant et de tri décroissant sont résumées sur les fig. **9** et **10**.

CONCLUSION

On a vu que le nombre de comparaisons nécessaires pour trier n éléments n'est pas un seul critère de l'efficacité d'une méthode de tri. Le nombre d'échanges ou de déplacements d'éléments ou d'indices et l'espace nécessaire sont également des mesures de l'efficacité d'une méthode.D'autre part, dans les conditions d'une mémoire virtuelle le taux de défaut de pages est une mesure très importante qui caractérise un algorithme de tri.Si dans un intervalle de temps virtuel, l'ensemble des pages référencées n'est pas trop grand,c'est-à-dire le "Working set" n'est pas grand, les défauts de pages seront moins nombreux.
Le taux de défauts de pages a montré qu'il existe une taille de mémoire en dessous de laquelle le nombre de défauts de page est très élevé par unité de temps;la taille de la mémoire est un critère plus important que l'algorithme choisi.

BIBLIOGRAPHIE

A.1.ACM Sort symposium,Princeton,New Jersey, Sorting on Computer,Comm.of the ACM 6,n°5 , May 1965
B.1.Belady,L.A.,Palermo,F.P.,On line Measurement of Paging Behavior by the Multivalued MIN Algorithm,IBM J.Res.Develop.,Jan.1974
B.2.Bose,R.C.,Nelson,R.J.,A sorting problem, Journal of the ACM 9,n°2 ,April 1962

B.3.Brawn,B.G.,Gustavson,F.G.,Mankin,E.S., Sorting in a Paging Environment,Comm.of the ACM 13, n°8.1970
B.4.Brawn,B.S.,Gustavson,F.G.,Program behavior in a paging environment,Fall Joint Computer Conference, 1968
D.1.Davčev,D., Thèse de docteur ingénieur : Intwrpréteur APL sur Mitra 15,Université de Paris,1975
D.2.Davčev,D.,Mémoire virtuelle et la technique d'optimisation adaptée à l'interpréteur APL sur Mitra 15,Informatika n°1,1977
D.3.Denning,P.J.,The Working set model for programming behavior,Comm.of the ACM 5,18,1968
D.4.Denning,P.J.,Schwartz,S.C.,Properties of the Working set Model,Comm.of the ACM 15,3, 1972
F.1.Flores,I.,Computer Sorting,Prentice-Hall, Inc.,Englewood Cliffs,N.J.,1969
F.2.Flores,I.,Analysis of internal computer sorting,Journal of the ACM 8,n°1,1961
G.1.Glicksman,S.,Concerning the marging of Equal length tape files,Journal of the ACM 12,n°2,1965
K.1.Knuth,D.E.,The art of computer programming,vol.3,Add-Wesley pub.Company, 1973
L.1.Lorin,H.,A guided bibliography to sorting,IBM syst.Journal 10,n°3,1971
P.1.Prieve,B.G.,Using Page Residency to Select the Working set Parameter,Comm.of the ACM 16,n°10 ,1973
W.1.Woodrum,L.J.,Internal sorting with minimal comparing,IBM Syst.J., n°3,1969

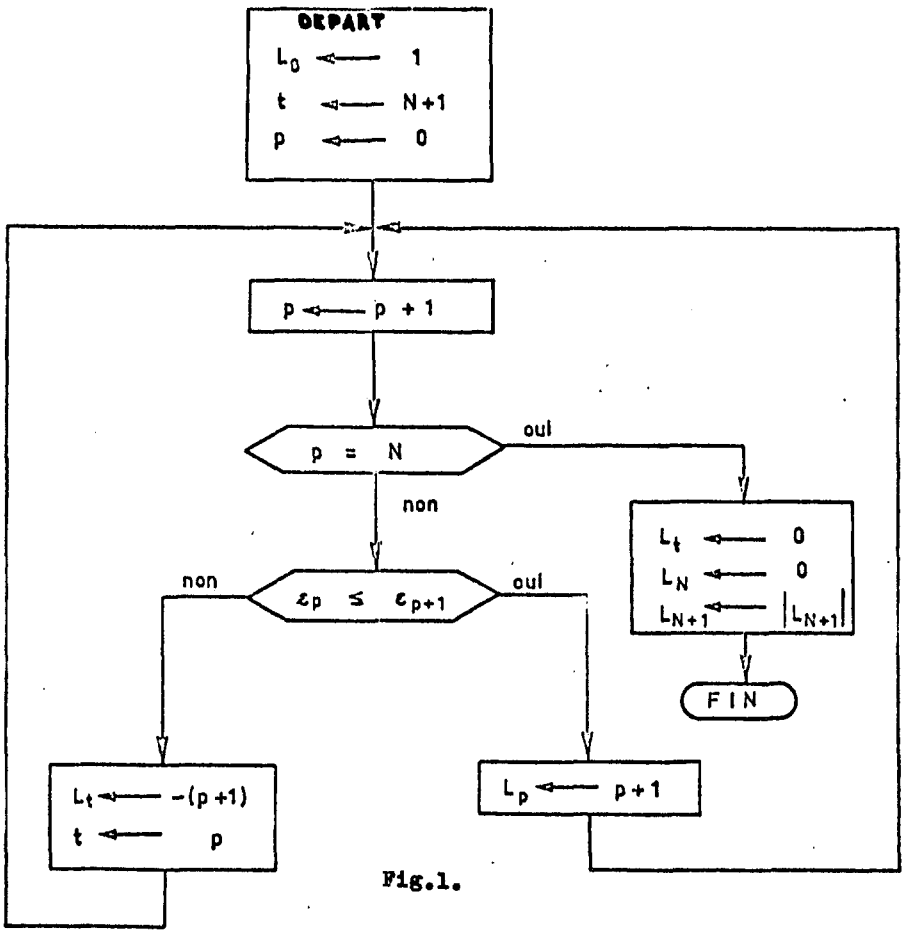


Fig.1.

Fig.2.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
ε_i		49	75	35	37	3	85	74	30	47	35	40	21	31	50	20	1	83	21	58	99	
L_i	1	2	-5	4	-7	6	-8	-10	9	-12	11	-15	13	14	-16	-18	17	0	19	20	0	

Fig.3

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
ε_i		49	75	35	37	3	85	74	30	47	35	40	21	31	50	20	1	83	21	58	99	
L_i	1	2	-5	4	-7	6	-8	-10	9	-12	11	-15	13	14	-16	-18	17	0	19	20	0	3
L'_i	3	2	-8	4	1	7	-15	6	10	-13	11	9	13	14	0	12	18	20	19	17	0	5
L''_i	5	7	6	4	1	3	-16	2	13	14	11	9	8	10	0	12	18	20	19	17	0	15
L'''_i	5	14	6	10	11	15	0	2	13	1	4	9	8	3	7	12	18	20	19	17	0	16
L^{NZAS}_i	16	14	17	10	11	15	20	2	13	1	4	9	18	3	19	12	5	6	3	7	0	0

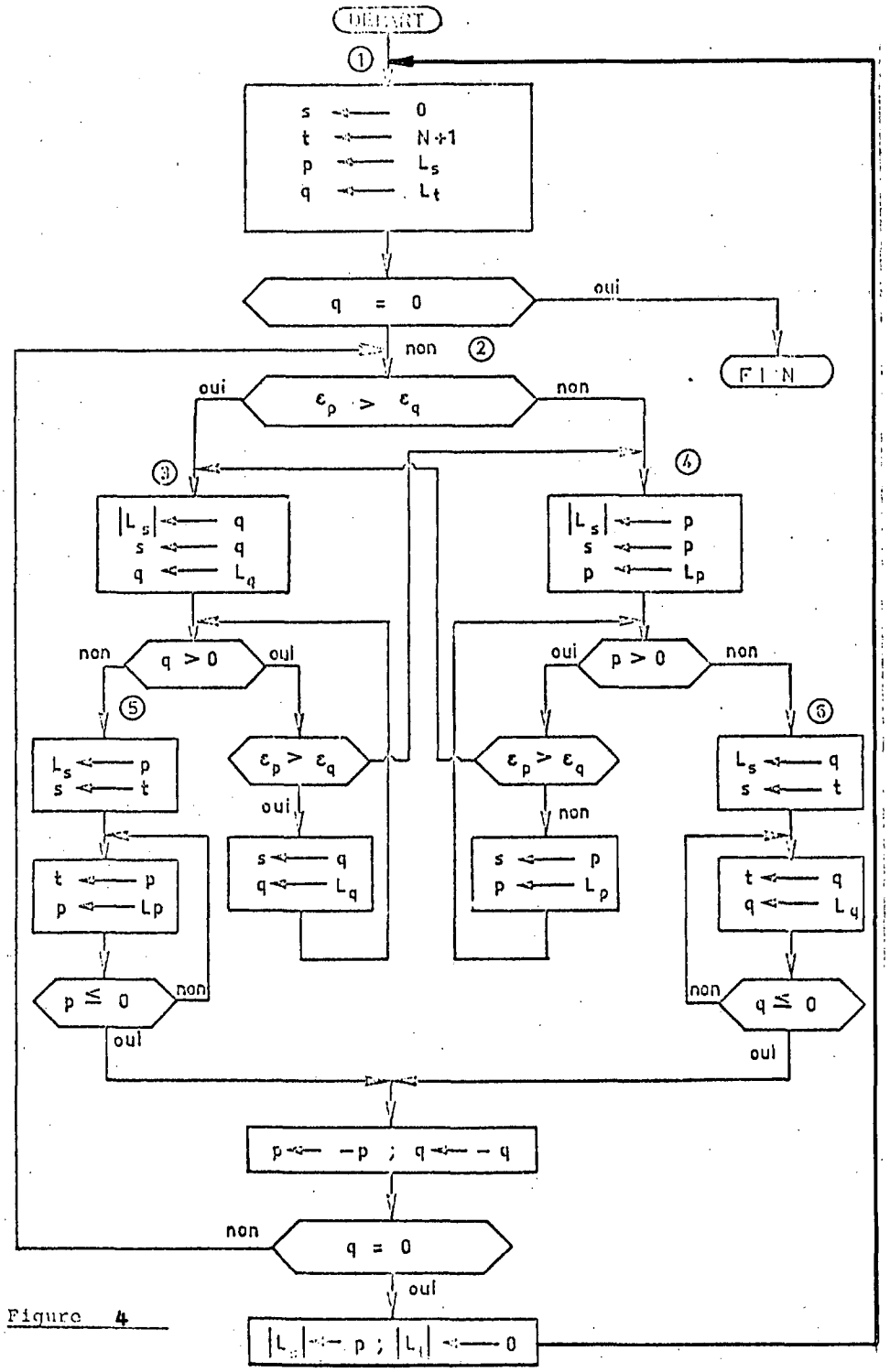


Figure 4

Fig.5.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
L'_i	16	14	17	10	11	15	20	2	13	1	4	9	18	3	19	12	5	6	8	7	0	0
P_i	16	-13	-17	-8	-10	-2	-19	-16	-6	-12	-9	11	-4	-7	14	-3	-1	-10	-5	-15	20	0

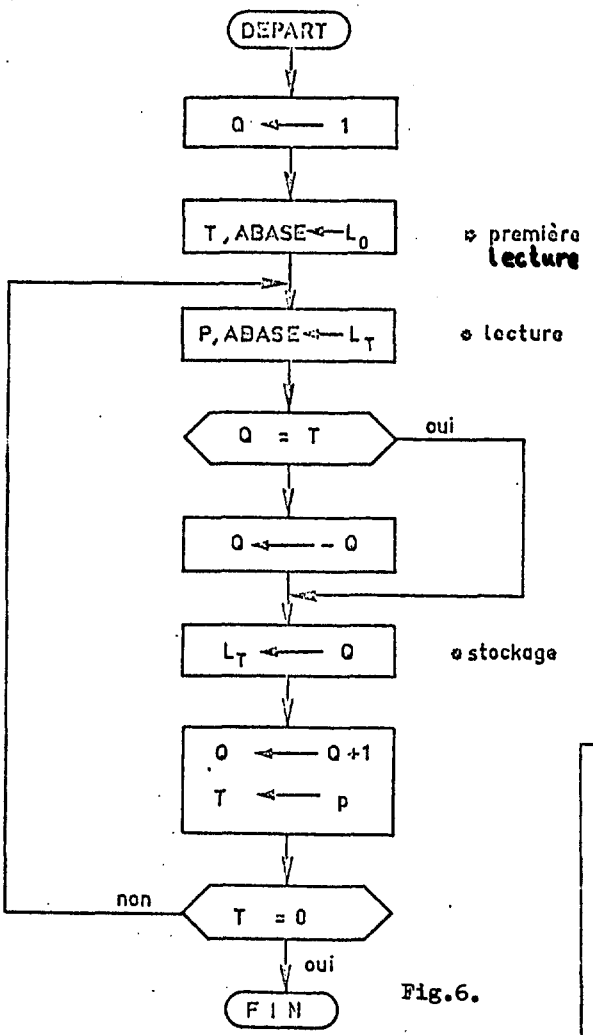


Fig.6.

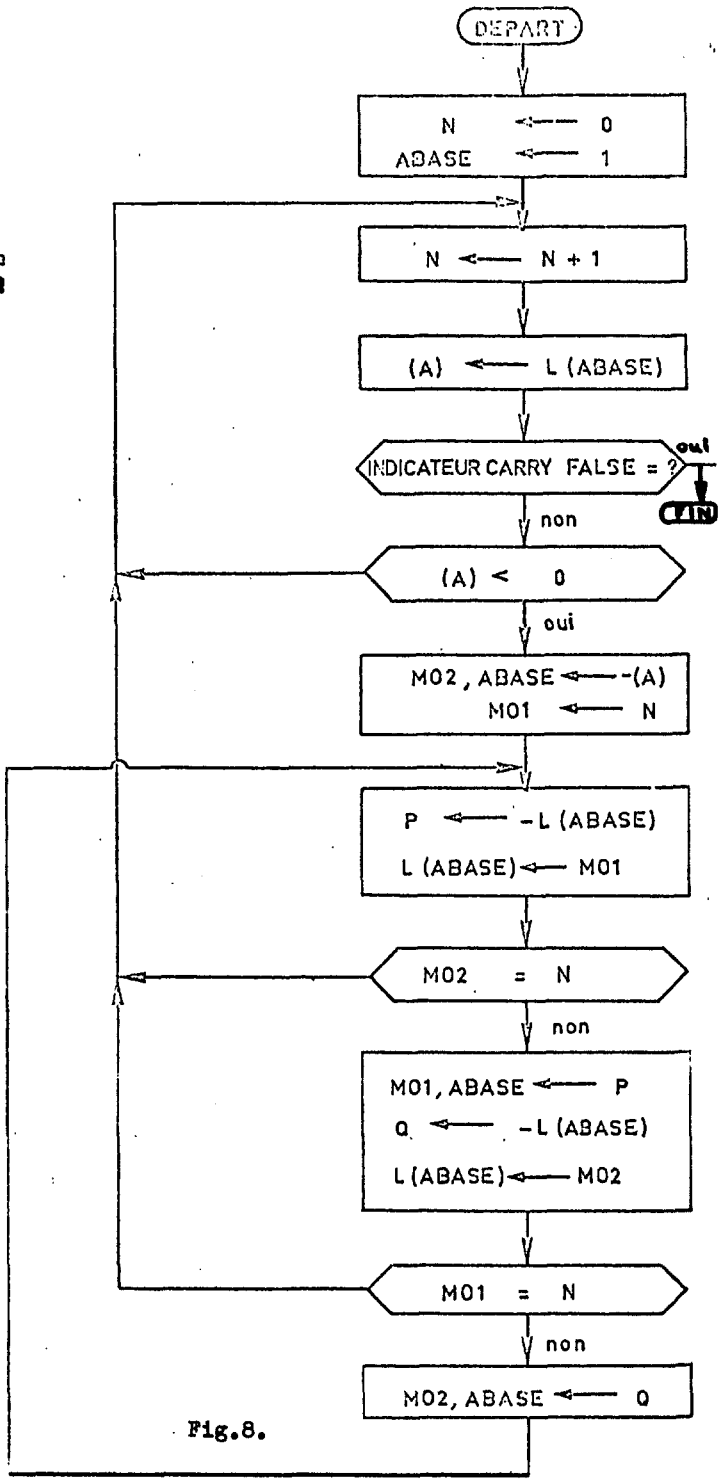


Fig.8.

Fig.7.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P_i	16	-13	-17	-8	-10	-2	-19	-16	-6	-12	-9	11	-4	-7	14	-3	-1	-13	-5	-15	20	0
		16						13						1			7					
			5			18												2	17			
				15			3		3							19				6		
					12					10	4		9									

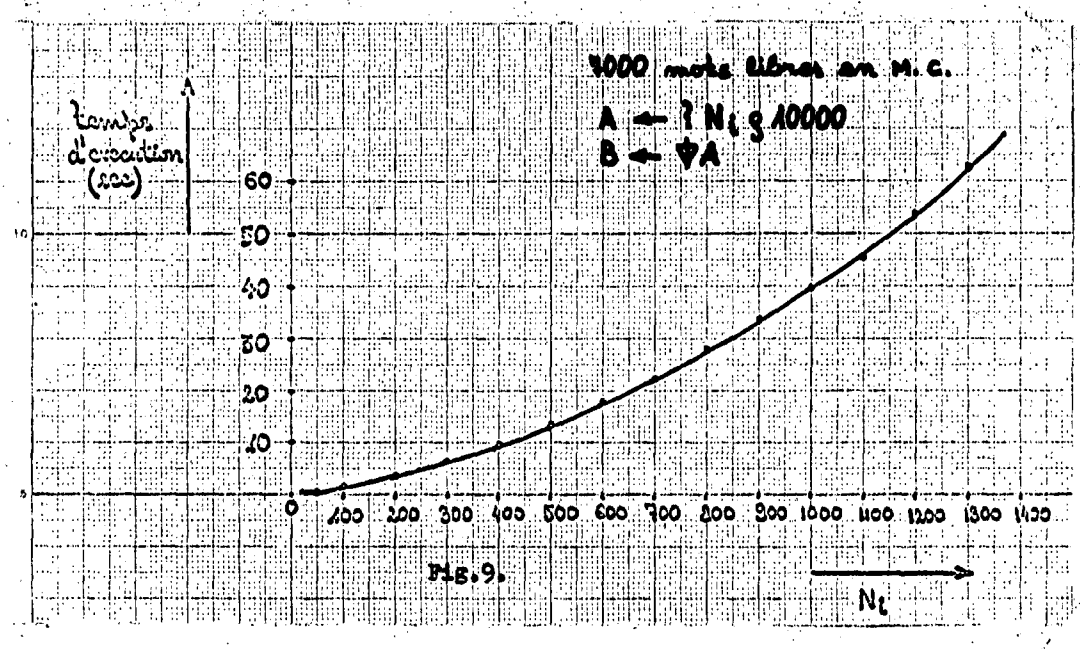

1000 mots libres en M.C.
A ← ? Nℓ ≤ 10000
B ← ΨA
Temps d'exécution (sec)
60
50
40
30
20
10
0
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
Nℓ

Fig.9.

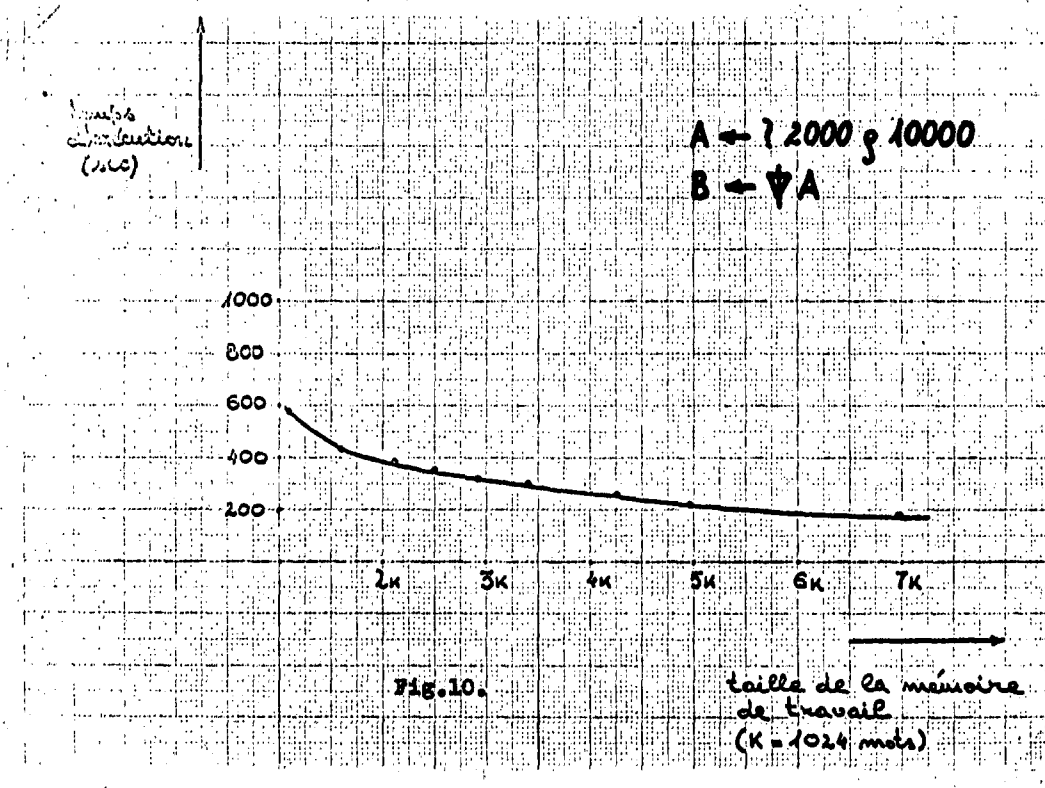

A ← ? 2000 ≤ 10000
B ← ΨA
1000
800
600
400
200
2K
3K
4K
5K
6K
7K
taille de la mémoire de travail (K = 1024 mots)

Fig.10.

linguistics and automatic processing of texts

petr sgall

UDK 681.3:801

Charles University
Prague, Czechoslovakia

A system designed to construct and update an automatic encyclopedia must be based i.a. on a systematic description of natural language, including a specification of a disambiguated (tectogrammatical) language and algorithms of translation between this language and the surface structure of natural languages. The paper gives a characterization of some of the properties of the tectogrammatical language constructed by the Prague group of algebraic linguistics.

LINGVISTIKA IN AVTOMATSKO PROCESIRANJE TEKSTOV. Sistem za konstruiranje in ažuriranje avtomatske enciklopedije mora temeljiti na sistemskem opisu naravnega jezika, ki vsebuje specifikacijo nedvoumnega (tektogramatičnega) jezika in algoritmih za prevajanje med tem jezikom in površinsko strukturo naravnih jezikov. Članek podaja karakterizacijo nekaterih lastnosti tektogramatičnega jezika, razvitega v praški šoli algebrske lingvistike.

It is possible to imagine a situation where the authors writing about, say, electronics, or chemistry, etc., will be advised not to use sentences longer than twenty words, the pronoun which without a preposition, the conjunction as in positions where because can be used instead, etc.; otherwise their texts will not be understood by the system designed to construct and update the automatic encyclopaedia of the given branch of technology, and customers using this system as a source of their information probably will not find and use the author's results. The general shape of such a system is more or less clear today, as far as its division into three main parts is concerned: (a) the brain, which organizes the whole repertory of data (including definitions, theorems, fact as well as bibliographical information, etc.), finds the proper place for new bits of knowledge, if they are included in a text correctly presented, and finds proper answers to the questions asked by users of the system; (b) the analysis, which takes, as its input, text and also questions - both in natural language mixed with formal notation, to which we are used from the present-day technological periodicals - and presents, at the output, their disambiguated translations in a language the brain can directly use; (c) the synthesis, translating the answers delireved by the brain from that language into English, Czech, or Chinese, according to the needs of the customer. As for the analysis and the synthesis, they must, we suppose, be accounted for first of all by linguistic means.

There are three main questions concerning the linguistic aspects of systems of the kind characterized above: (i) the language of disambiguated information, used as input and output language of the brain, must be designed properly; (ii) an algorithm for the synthesis, i.e. for the translation from this to the natural language, as well as (iii) for the analysis of the text in natural language and its translation into the artificial language must be constructed.

The following three hierarchies involved in the syntax of the sentence of a natural language are semantically relevant and must be rendered unambiguously in the tectogrammatical language:

(1) the hierarchy of predications (and depredications), or of predicates and their arguments (participants, cases): in Mother read the letter aloud to the children, who wondered it really contained all the news, we have the main predicate read with its participants - actor Mother, objective letter, manner aloud and addressee children, while this last item itself governs the embedded phrase having wonder as its predicate, children (substituted by the pronoun who) as its first participant with the second one having the shape of another clause with its own predicate, etc.;

(2) the hierarchy of topic and focus, including a scale of "communicative dynamism", which is not exactly identical to the distinctions between what is "given" and "new", since, for instance in It's better to give it to her than to him, the pronoun her refers to someone known, but the pronoun itself is included in that part of the sentence which is presented as a piece of new information, and not mentioned only to be activated in the hearer's memory and connected with some new information;

(3) the hierarchy of the so-called delimiting features of noun phrases, including their definiteness, generic use, indefiniteness, etc.

In order to account for these three hierarchies, the tectogrammatical language employs the following assumptions:

(1) The representation of the sentence on the

tectogrammatical level (henceforth the semantic representation, SR) has the form of a dependency graph, with the verb as its root. The nodes correspond to word forms and the edgex (connecting the two members of a syntactic pair) to the relations of syntactic dependency (e.g. actor, object, dative, instrument, adverbial of cause, locative, etc.)

(2) It is assumed that the participants of a verb are ordered, essentially (i.e. in sentences where intonation is not superimposed on word order, so that the intonation centre is carried by the last lexical item), according to three factors:

(i) an inherent order of the participants determined by the language system;

(ii) topicalization of a participant which, according to (i) would occupy a right -hand position in the ordering, but, being only mentioned as contextually bound, i.e. already known from the previous context or situation, carries a lower degree of the so-called communicative dynamism, which can be regarded as the "deep word order"; in this way the left-to-right ordering of major symbols in an underlying P-marker would get an appropriate interpretation (it would not be left uninterpreted, as in one variant of Fillmore's case grammar);

(iii) rules of grammar, according to which an adjunct NP must follow its head NP, an object follows its verb (if not passivized), etc.

(3) The SR of each sentence is marked (by the superscript b) as to which elements are contextually bound and which are contextually non-bound.

(4) As for the position of the verb in the SR, we assume that it always stands between its contextually bound and contextually non-bound participants. Therefore it is possible to employ a notation in which the position of the verb does not correspond directly to its degree of CD but the verb, as a predicate, is put to the left of its arguments, i.e. of the participants and adverbials:

$$V(A_1^b, A_2^b, \ldots A_j^b, A_{j+1}, \ldots, A_n).$$

An experimental version of generative rules has been used to derive automatically the semantic representations. The generative component has the shape of a context-free phrase structure grammar; it is simplified from the empirical point of view in that the variation in communicative dynamism (the various types of topic/comment articulation of a sentence) is neglected and only one "deep word-order" for each type of construction is generated.

Complex non-terminal symbols of the grammar are ordered n-tuples $(X_1, X_2, \ldots, X_n)$ where X_1 is the so-called name-symbol, i.e. a name shared by a certain class of non-terminal symbols of that class, $X_2, \ldots, X_n$ are indices specifying individual non-terminal symbol-names. Similar structure characterizes a terminal symbol, as well.

The syntactic relations between complex symbols (each of which comprises a lexical unit together with its indices or grammatemes) are accounted for by means of functors denoted by $\underline{R}$ with a subscript. In the experimental version of the description six such functors are distinguished: $\underline{R}_a$ for the relation of agent to the verb, $\underline{R}_p$ for that of patient, $\underline{R}_i$ for the indirect object (dative), $\underline{R}_n$ for the so-called second object, $\underline{R}_d$ for the free (adverbial) expansions, and $\underline{R}_v$ for a zero relation (as e.g. with the vocative case). In a more recent version of the tectogrammatical representation the functors are differentiated to a greater extent, the adverbial relationship (formerly denoted only by different grammatemes of the shape det_i, accompanying the lexical symbol connected with the verb by $\underline{R}_d$) being denoted by different functors, such as $\underline{R}_{loc}$, $\underline{R}_{dir}$, $\underline{R}_{caus}$, etc. The grammatemes of the shape det_i now account only for semantic variation inside a given adverbial (e.g. a noun connected with its governing verb by means of $\underline{R}_{loc}$ can be accompanied by one of the grammatemes det_{inside}, det_{above}, det_{below}, etc.; the functors for temporal adverbial can be combined with a semantic variation comprising simultaneity, precedence and subsequence.

The expansions of the verb are divided into inner and free ones; the former are either obligatory or optional. On the base of whether a verb has a given expansion as an inner one (included in its verbal frame, in Fillmorian terms), whether this expansion is obligatory or optional, and to what type of expansion (with what functor) it belongs, we distinguish, in the experimental version, 15 classes of Czech verbs. The classes are mutually disjunct (so that a partition on the class of all described verbs is defined), and the membership of a verb in a class is denoted by the value of one of the indices accompanying the lexical symbol of the verb.

The translation of the semantic representation to the graphemic level, i.e. its transduction to a sequence of Czech word-forms is realized by means of several steps. In the first of them, the semantic representation is translated to the phenogrammatical (surface syntactic) level; the functors, corresponding to the relations between a governing and a dependent lexical item at the tectogrammatical level, are changed here into symbols (grammatemes) for sentence parts; in this connection, the choice between active and passive is made, as well as between a dependent clause, an infinitive, a nominalization, etc.; the semantic functions (meanings) are substituted by the means realizing them at the next lower level.

The second step consists in the translation of the representation to the morphemic level; the meanings of place, time, cause etc. are changed here into the morphemic forms realizing them (prepositional phrases etc.); furthermore, the morphemic units of tense, aspect, gender, number etc. are chosen here. The ru-

les of morphemic synthesis translate then these morphemic representations into sequences of Czech word forms (with case inflections, personal encings etc.) corresponding to grammatical sentences; at last, the graphemic shape of a sentence is achieved, which expresses the meaning that was represented by the given input string of the transductive components.

The sequence of computer programs performing this transduction is based on the formal pattern of pushdown transducers. The main program of each step is constructed on the basis of the defining function of such a transducer (see SGALL et al. /1/, pp. 40f, 60f, 76 ff), where by means of a single passing through the given representation of a sentence the changes necessary for the transduction to the next lower level are ensured, while every dependent word (rectum) is confronted here with its governing word (regens). The results of such a confrontation (first of all, modifications of the dependent word according to relevant properties of its governor) are given in the form of large tables, represented as subroutines of the main program, activated always when the two members of a syntactic pair are confronted, one of them being read at the input of the transducer and the other being at the accessible end of the pushdown store. The large size of the tables makes it necessary to have specific subroutines (a) for the identification of the types of word forms figuring as names of rows and (b) as names of lines and (c) for the identification of the result found in the table (at the crossing of the given line and row), i.e. the value of the function for the given values of its two arguments.

A detailed account of the tectogrammatical language and of the program of random generation of SR's of Czech sentences may be found in /2/. A description of the programs of synthesis and of the - now being prepared - algorithms of analysis of Czech sentences will be published in the next two volumes of the same series.

References:

/1/ Sgall, P., Nebeský, L., Goralčíková, A., Hajičová, E. : Functional Approach to Syntax, New York, 1969

/2/ Explizite Beschreibung der Sprache und automatische Textbearbeitung. III. Die tektogrammatische Ebene der funktionellen generativen Beschreibung. Praha 1977

študentska vprašanja

Kot vedno, ko se jeseni ponovno srečamo na fakulteti, se pogovarjamo o prijetnih počitnicah in seveda tudi o tem kje in kako smo opravljali poletno prakso.
Skoraj vsi smo štipendisti in moramo na prakso v organizacije, ki nas štipendirajo. Zelo zanimivo in koristno prakso smo imeli na Institutu Jožef Stefan. Nekatera podjetja so zelo dobro poskrbela za svoje bodoče strokovnjake. Ti bodo delali na obdelavi podatkov v komerciali in so ta mesec izkoristili, da so se spoznali z obdelavo datotek v jeziku COBOL, ki se ga nismo učili na fakulteti. Imeli smo zagotovljene prakse tudi v tujini, ki pa na žalost niso bile ravno za računalnikarje. Imam kolegico, ki se je vrnila iz tujine že drugi dan, ker ni dobila potrebnega dovoljenja za opravljanje prakse. Ob tem smo se spomnili tudi kolegov, ki so navijali kondenzatorje, ali opravljali administrativna dela.

Če že moramo vsako leto imeti en mesec prakse, bi bilo dobro, da ja ta resnično strokovna (ker radi delamo, če se ob tem lahko kaj koristnega tudi naučimo), kot naprimer na fakulteti, institutu ali v podjetju, ki je zainteresirano za naše delo.

Vprašanje pa je, če se bodo študentje v bodoče boljše vključevali v raziskovalno delo in stalno delali v zadnjih letnikih ob študiju. Ali je taka praksa za računalnikarja resnično potrebna, ali ni to samo formalnost, ki jo moramo opraviti? Ob eventuelnem raziskovalnem delu ali delu na aplikativnih nalogah bi lahko naredili poročilo o opravljenem delu in bi se to priznalo na fakulteti kot praksa, brez pisanja dnevnika prakse (dan za dnem) na osnovnošolski način.

Sigurno pa je, da bi vsi študenti morali biti vključeni v delo na nalogah na fakulteti, institutu ali v zainteresiranih podjetjih.

N.P.

bajke o izdelavi programskih projektov

Vrsta projektantov programske opreme ni uspela, ker so predpostavljali nekaj, kar se sicer zdi prav, a je že v osnovi narobe. Nekaj najpogostejših predpostavk bomo predstavili v naslednjih vrsticah.

Prva napaka se pokaže že v načrtovanju projektov v razmerju človek-mesec. Napaka se z lahkoto razloži: "Stroški so res sorazmerni produktu števila ljudi in števila mesecev, napredek projekta pa ne. Človek-mesec je kot enota za merjenje količine dela nevaren in varljiv mit. Namiguje namreč na to, da se ljudje in meseci lahko izmenjujejo!"

Možnost napake lahko uvidimo, če premišljamo o možnosti, da bi pri zakasnelem projektu dodali še človeško moč. Kalkulacija števila dodatnih ljudi, ki so potrebni, da se razmerje človek-mesec zopet uravna, navadno ne uspe, ker moramo predvideti čas, ki ga bodo novi ljudje porabili za vpeljavo v delo in zanimanje tega osebja za obstoječi projekt. To nas privede do Brookovega izreka, ki stvar poenostavi, ko pravi: "Če dodamo človeško moč zakasnelemu projektu, ga še bolj zakasnimo!" Edina možnost je, da dodamo čas, ne pa človeško moč.

Naslednja napaka je posledica t.i. efekta drugega sistema. V začetku je razumljivo, da predvidimo, da bodo ljudje, ki drugič načrtujejo isti tip sistema, opravili delo bolje kot tisti, ki opravljajo to prvič. V resnici pa ga bodo opravili slabše.

Da razumemo zakaj, je potrebno, upoštevati stališče načrtovalca. Pri prvem poskusu na danem tipu sistema se načrtovalec zaveda, da se spoprijema z novim tipom problema in zato je njegov pristop izredno previden in končni produkt je pičel in omejen. Vsi popravki in bistri domisleki se ne uporabijo in se prihranijo za drugič za "izboljšano" verzijo.

Drugič, ko isti konstrukter dobi priložnost razvoja podobnega sistema, bodo vse te "izboljšave" uporabljene. Rezultat bo verjetno konstrukcija, ki jo lahko opišemo kot končno. Še več, verjetno bo mnogo olepšav osnovanih na domnevah (glede tehnologije in metod dela), ki bodo takrat, ko začne drugi sistem eksistirati, že neveljavne. Zato izboljšave niso samo zamotane in nenavadne ampak tudi nepotrebne.

Ko pride načrtovalec do tretjega podobnega sistema, se je verjetno že izučil na napakah v drugem sistemu. Izhod je torej v tem, da je konstrukter nekdo, ki je konstruiral vsaj dva podobna sistema prej, ali nekdo, ki se dobro zaveda efekta drugega sistema.

Tretja bajka pa je, da manjši neuspehi v projektu niso važni. Zakaj nastane zakasnitev pri izvajanju projekta?

Večina projektantov se zaveda pomembnosti načrtovanih rokov: vendar pa nimamo zagotovila, da se dajo projektni roki dobro preveriti. Zaključek projektne specifikacije ni natanko določljiv in ga lahko razglasimo kakor želimo; dejstvo, da so bile specifikacije podpisane obojestransko je posebna, prepričljiva točka v projektu. Tedaj morajo biti roki dovolj pogosti, da pokažejo napako kmalu potem, ko nastane; dejansko potrebujemo roke na vsak centimeter projekta.

To so samo trije elementi v bajeslovju izdelave programske opreme in ti so tudi najbolj pogosti.

Po članku Software Myths, Infotech Update, Vol.2, No.8

D.N.

mnenja naročnikov o časopisu

... Koristim ovu priliku da izrazim svoje zadovoljstvo zbog pokretanja ovog časopisa koji će po mom mišljenju znatno doprineti razvoju svih oblasti kojima je posvećena pažnja u našoj zemlji. Uredništvu želim mnogo uspeha u budućem radu.

Dipl. Ing. M.P., Titograd

novice in zanimivosti

4 K Dinamična bipolarna RAM memorija sa vremenom dostupa od 100 n s, uključuje dvoje čip select linije, mogućnost kontrole prihvata podataka te brzu tehniku straničenja (paging). Brža verzija, 93481A, ima maksimalno vrijeme dostupa od 100 ns i vrijeme ciklusa od 240 n s. Standardna verzija ima maksimalno vrijeme dostupa od 120 ns i vrijeme ciklusa od 280 ns. Oba čipa regularno djeluju od 0° C do 70° i zahtijeva izvor energije od +5V±5%. Aktivna potrošnja energije je 350 mW, statična 70 mW te 500 mW pri djelovanju u page načinu. Pri page načinu djelovanja vrijeme dostupa i vrijeme ciklus je 75 ns za 93481 i 65 ns za 93481A. Cijena je $ 24 za 93481 i $ 31,20 za 93481A. Fairchild Camera and Instrument Corp., 464 Ellis st., ms 20-1050, Mountain View, CA 94042, (415) 962-3951.

Zahvatanje podataka pisanih slobodnom rukom je moguće upotrebom takozvanih Datapad sistema za zahvatanje podataka. Takav sistem se sastoji od specijalne ploče za pisanje, male displaj jedinice i mini računara sa specijalnim softverom. Nije potrebna posebna obuka kadrova za zahvatanje podataka ovim putem; podatke unosimo tako da slobodnom rukom pišemo po ploči za pisanje. Displaj jedinica verificira raspoznane podatke. Mini računar može da opslužuje do 16 terminala producirajući magnetni ili papirni trak koji će sadržavati određene ulazne podatke.

Novi, 16-bitni mikroprocesor u IIL tehnologiji (Integrated Injection Logic) sa oznakom SBP 9900 je objelodanila firma Texas Instruments. Direktna TTL ulazno/izlazna kompatibilnost omogućava gradnju sistema na osnovu čipa 9900 sa standardnim memeorijskim elementima, bez potrebe za specijalnim klok generatorima ili prilagođavajućim elementima.
SBP 9900 je u pogledu softvera kompatibilan sa svim članovima familije 9900, te sa nekim hardverskim i softverskim elementima Teksasovih mini računara familije 990.
Teksas Instrument Inc, Inquiry Answering Service, P.O. Box 5012, M/S 303 (Attn:SBP 9900), Dallas, TX 75222. (713) 494-5115, Ext. 2621

Necol je program koji izvršava disk-na-disk konverziju programa pisanih u jeziku Neat/3 u ekvivalentni program u jeziku Cobol/74.
Neat/3 je programski jezik koji se upotrebljava u NCR Century računarima. Korisnici koji žele da prenesu svoje programe na računare sa osnovnim jezikom Cobol/74 (npr. NCR 8250 i Criterion serije) mogu uz pomoć priručnika dosta jednostavno prilagoditi svoje programe novoj mašini.
Neke naredbe jezika Neat/3 ne mogu biti prevedene u Cobol. U takvim slučajevima mora programer dopuniti transformirani program što zahtijeva neznatne napore i vrijeme.
Computer Facilities Software Ltd., Cleckheaton, W. Yarks, tel. (0724) 63167

Console 6800 je kontrolna ploča koja pojednostavljuje operiranje , testiranje programa i otkrivanje grešaka na mikro sistemima koji baziraju na procesorima Motorola 6800. Kontrolna ploča je bus kompatibilna. Od posebne važnosti za otkrivanje grešaka u programima, je mogućnost djelovanja sistema korak po korak (single step) te zaustavljanje izvršavanja programa pri određenoj naredbi. Po zaustavljanju izvršavanja programa je moguće dobiti informacije o sadržaju internih registara centralne procesne jedinice ili memorijskih lokacija te modificirati te sadržaje.
Di-An Data Systems Ltd, 70-74 Princess Street, Stockport, Cheshire SK1 1RJ. Tel: 061-236 2321

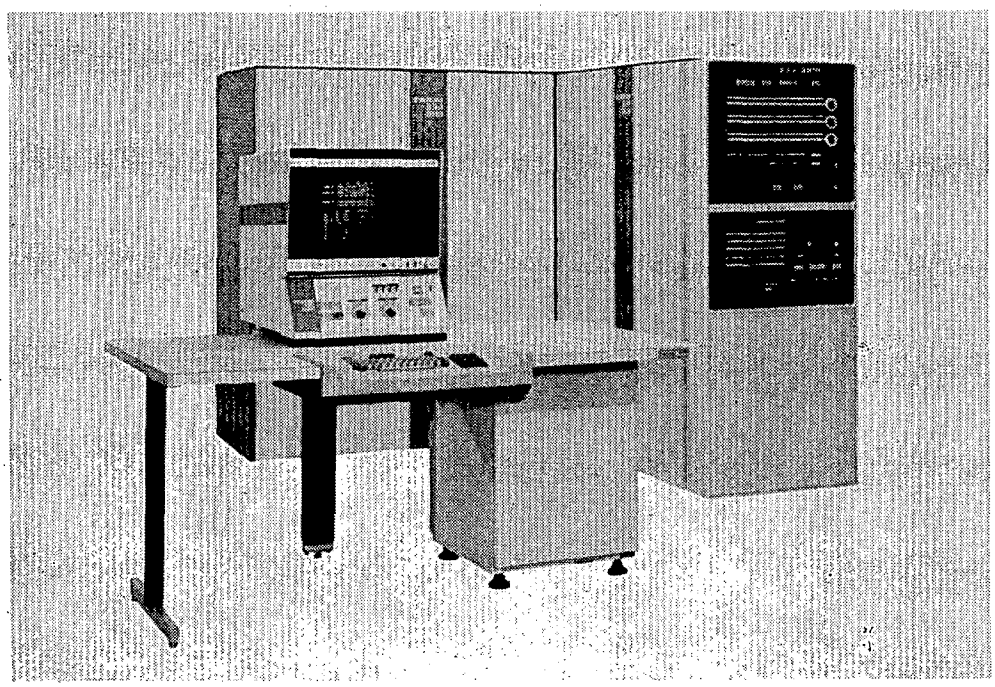
Digital Equipment naznanja prihod svojega prvega 32 bitnega miniračunalnika na trg ob koncu oktobra letos. Uporabili so tehnologijo za velike hitrosti operacij, znano kot ECL (Emitter-coupled logic). Procesor bo izvrševal vse ukaze računalnika PDP-11 ter svoje lastne. Zato je izveden z mikrokodiranjem in je zmožen emuliranja drugih kodov. Glavna prednost daljše besede je v večji hitrosti procesiranja in večji aritmetični natančnosti. Poleg tega pa omogoča bolj svobodno naslavljanje, ker 16 bitni procesorji močno omejujejo območje direktnega naslavljanja zaradi majhnega števila bitov za naslove.

Signetics Corporation iz ZDA nudi na tržišču emulator za MOS mikroprocesorje, ki jih želimo pohitriti do hitrosti bipolarne tehnologije. Emulator je namenjen predvsem za MOS mikroprocesor 8080A. Signetics 8080A emulator oprema sestoji iz plošče tiskanega vezja in potrebnih komponent za sestavo Schottky bipolarne verzije sistema 8080A; vsebuje tudi vsa vmesna vezja za kanale ter krmilna in časovna vezja. Vendar je emulator možno prirediti tudi drugim mikroprocesorskim sistemom ali pa za posebne nabore ukazov, ker ima na voljo 150 mikroukaznih lokacij. Tako lahko emulator uporabimo tudi kot procesor z operacijami bitov vezij znotraj besed. Z emulatorjem postane sistem 8080A približno petkrat hitrejši. Cena opreme in priročnika z navodili za uporabo je 304 $.

V svetu intenzivno razvijajo magnetne mehurčne pomnilnike predvsem za uporabo pri mikroračunalnikih. Naštejmo nekaj tvrdk, ki pripravljajo te nove pomnilnike: Bell Laboratories Inc., Rockwell International Corp., IBM, Hewlett-Packard, Nippon Telegraph and Telephone (Tokyo), Fujitsu. Magnetne pomnilnike bodo uporabili kot periferne pomnilnike za mikroprocesorje in so že sedaj glede na cene v prednosti pred disketami. Resnično prednost pa vidijo, ker lahko mehurčni pomnilnik namestijo zaradi njegove majhne velikosti na isto ploščo tiskanega vezja, kjer je centralna procesorska enota.
Pri tem pa ne potrebujejo dodatna vmesna elektronska vezja ter povezovalnih kablov.

Hewlett-Packard bo v naslednjih mesecih ponudil računalnik 21 MX s polprevodniškim pomnilnikom enega milijona zlogov za ceno 59.800 $, To je rezultat uporabe nove polprevodniške tehnologije SOS (silicon-on-saphire). Uporabo tega "velikega" miniračunalnika vidijo v dejavnostih, kjer magnetni koluti ne zadovoljujejo niti s hitrostjo niti z zanesljivostjo. En sam integrirani element RAM vsebuje 16 K spomina. Zaradi take koncentracije spomina so megazlogovni pomnilnik opremili z vezji za odpravo napak na osnovi Hammingovega kodiranja. Sistem za odpravo napak s petimi korekcijskimi biti odpravlja napake enega bita ter zazna napake dveh ali treh bitov v besedi. S tem da je vsak bit ene besede v drugem RAM-u, pomeni napaka treh bitov hkratno odpoved treh RAM-ov, kar je zelo malo verjetno. Tako je zanesljivost tega pomnilnika šest do osem tisoč ur kot srednjim časom med odpovedmi.

PORODICA FACOM FIRME FUJITSU



Radeći bez mnogo buke, ali marljivo poslijednje četiri godine, FUJITSU je zajedno sa svojim zastupnikom ZPR-om (Zavod za primjenu elektroničkih računala) uspješno sklopila ugovore za više od 50 FACOM računala u Jugoslaviji.
Iznenadjeni? Ne morate biti. FUJITSU, povrh toga što je vodeći proizvodjač sistema za elektroničku obradu podataka na Japanskom tržištu, vrlo brzo preuzima jedno od vodećih mjesta i na svjetskom tržištu. Tajna uspjeha firme FUJITSU je u tome što ima vodeću tehnologiju u kombinaciji sa velikom pouzdanošću sistema i dobro organiziranom službom za održavanje i stručnu pomoć. Ne smijemo zaboraviti ni konkurentne cijene koje će vam dati najbolji mogući odnos cijena/performance.
Ako razmišljate o uvodjenju elektronske obrade podataka u vašoj organizaciji ili želite da poboljšate svoj sadašnji sistem, obratite se predstavnicima firme FUJITSU da vas upoznaju sa svim novostima.

FUJITSU proizvodi sve - sastavne dijelove, memorije, off i on-line uredjaje za prikupljanje podataka, inteligentne terminale, micro procesore i micro računala, malih, srednjih, velikih i super velikih kompjutera, uključujući najsnažnije kompjutore za opću svrhu koji se mogu kupiti na tržištu.
FUJITSU je poznata i u području telekomunikacija. To je razumljivo zbog toga što je FUJITSU jedan od vodečij proizvodjača telefona i telekomunikacija u Japanu.
FUJITSU je u Jugoslaviji izabrana da snabdije i pomogne kod razvoja najveće on-line real-time mreže koja je do sada ugovorena, uključujući oko 300 terminala.
Mislimo da biste sebi i svojoj organizaciji učinili mnogo, ako saznate više o tome što Vam firma FUJITSU može ponuditi.

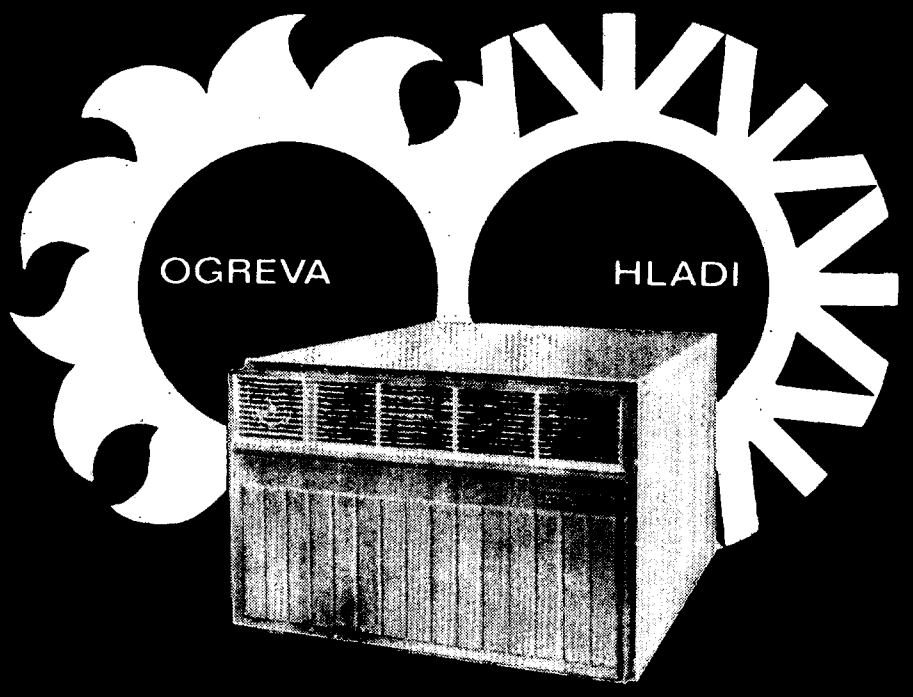
Servisni centri i uredi su u Ljubljani, Mariboru, Beogradu i Zagrebu

ZPR
ZAGREB, Savska 56
tel. 518-706
Telex 21689 YU ZPR FJ

FACOM

LJUBLJANA, Topniška 45
tel. 311-059

FUJITSU LIMITED
Communications and Electronics



VEDNO PRIJETNO POČUTJE
Rešitev vseh problemov klimatiziranja:

KLIMATIZER TOBI 32

KLIMA OMARE KO

- Samodejno uravnavanje temperature v prostoru
- Enostavna montaža in vzdrževanje
- Servisna mreža in rezervni deli na celotnem področju SFRJ

INFORMACIJE O PRODAJI IN INŽENIRINGU

EMO
63000 CELJE
Mariborska 86
tel.: (063) 23-921

ISKRA - TOZD INŽENIRINGI
61000 LJUBLJANA
Kotnikova 6
tel.: (061) 312-322

večjezični slovar izrazov

Objavljamo kratek večjezični slovar izrazov, ki se v računalništvu in informatiki, še posebej pa pri mikroračunalnikih, najbolj pogosto uporabljajo. Slovar še zdaleč ni popoln, prepričani smo tudi, da marsikateri predloženi izraz ne ustreza in bi bil zaželen ustreznejši.

Bralce vabimo k sodelovanju z novimi predlogi za izraze ter, da povedo svoje mnenje o nekaterih vprašljivih izrazih (zlasti onih, ki so označeni z vprašajem). Slovenske in srbohrvaške izraze sta pripravila L.Pipan in Z. Salčić .

ANGLEŠKI	NEMŠKI	SLOVENSKI	SRBOHRVATSKI
A			
Acknowledge	Bestätigen	potrditev, dovoljenje	odobrenje
Accumulator	Akkumulator	akumulator, zbiralnik	akumulator
Aging	Alterung	staranje	starenje
ALU	Zentraleinheit	ALE (aritmetična-logična enota)	aritmetička-logička jedinica
Assembler	Assembler	zbirnik	asembler
Assign	Zuteilen	prirediti, pridati	pridružiti
Asynchronous	Asynchron	asinhron, nesočasen	asinhroni
Available	Vorhanden	razpoložljiv, dosegljiv, na voljo	razpoloživ
B			
Battery	Batterie	baterija, vir	baterija
Bi-directional	Beidseitig gerichtet	dvosmeren	dvosmjeran
Bootstrap	Urlader	pra(Bootst.Loader-pranalagalnik)	automatski punilac
Branch	Verzweigung	razvejitev, kretnica?	grananje
Buffer	Puffer	vmesni pomnilnik, medpomnilnik	bafer
Bus	Bus	vodilo	sabirnica
Byte	Byte	zlog	bajt
C			
Cabinet	Schrank	ohišje, omara	kabinet
Capacitor	Kondensator	kondenzator	kondenzator
Card	Lochkarte	kartica, luknjana kartica	karta
Carry	Übertrag	prenos	prenos
Cartridge	Spule	ovitek, saržer, nabojnica,	punjenje
Channel	Kanal	kanal	kanal
Chip	Chip	tableta, tabletka, gosenica	čip
Clock	Uhr	ura	sat
Command	Befehl	ukaz	komanda
Compiler	Compiler	sestavljalnik, prevajalnik	kompajler
Computer	Computer	računalnik	računar
Console	Konsole	konzola, zaslon	konzola
Controller	Steuerung	krmilnik	upravljač
Converter	Konverter	pretvornik	pretvarač
Core	Kern	jedro	jezgro
CPU	Zentralsteuereinheit	CPE(centralna procesna enota)	centralna procesna jedinica
Cross-Assembler	Cross-Assembler	prečni zbirnik	kros-asembler
C.R.	Kartenleser	čitalnik kartic	čitač kartica
CRT	Bildsichtgerät	prikazovalnik	ekran
D			
Data	Daten	podatki	podaci
Data Collection	Datensammlung	zbirka podatkov	prikupljanje podataka
Debugger	Fehlerbeseitiger	čistilec, čistilnik	program za provjeru
Device	Gerät	naprava	uredjaj
Disk	Platte	disk	disk
DMA	Direkter Speicherzugriff	neposreden dostop do pomnilnika	direktan memorijski pristup
Drive	Antrieb	gonilo, gnati	pojačavati
Driver	Treiber	gonilnik	drajver
Dynamic	Dynamisch	dinamičen	dinamički

E			
Editor	Editor	izdajalnik	editor
Extension card	Erweiterungskarte	kartica za razširitev, plošča za razširitev	karta za proširenje
F			
Failure	Fehler	izpad, odpoved	kvar
Fetch	Holen	dostava	priprema, dostava
File	Datenblock	niz, kup	datoteka
Flag	Flagge	zastavica	zastavica
Flip-flop	Flip-flop	bistabilni multivibrator, bistabilno vezje	flip-flop
Floppy-Disk	Floppy Disk	gibek disk	flopi-disk
Formatter	Formatierer	formatnik?	formater
G			
Gate	Gatter	vrata	kapija
GP	General Purpose	splošen	opšte namjene
Ground	Erde	masa, zemlja	zemlja
H			
Hardware	Hardware	aparaturna oprema, hardver?	hardware
Hard-wired	Verdrahtet	ožičen	ožičen
High Level Language	Hohe Programmiersprache	visokonivojski jezik	jezik visokog nivoa
Hole	Loch	luknja	rupa
I			
Indexed	Indiziert	indeksiran	indeksiran
Indirect	Indirekt	posreden, indirekten	indirektan
Immediate	Unmittelbar	neposreden	neposredan
Input/Output	Eingang/Ausgang	vhod/izhod	ulaz/izlaz
Instruction-Set	Befehlssatz	nabor instrukcij, množica instr.	skup instrukcija
Insulation	Isolation	osamitev, ločitev, izolacija	izolacija
Interface	Interface	vmesnik	interfejs
Interrupt	Unterbrechung	prekinitev	prekid
J			
Jump	Sprung	skok	skok
K			
Key	Schlüssel, Taste	ključ, tipka	šifra
Keyboard	Tastenfeld	tipkovnica, tastatura?	tastatura
L			
Layer	Lage	sloj	sloj
Level	Ebene	nivo, raven	nivo
Library	Bibliothek	knjižnica	biblioteka
Line	Leitung, Zeile	vod, vrstica	linija
Line Printer	Zeilendrucker	vrstični tiskalnik	linijski štampač
Load	Laden	naložiti	puniti
Loader	Lader	nalagalnik, vlagalnik	punilac
Logical	Logisch	logičen	logički
Loop	Schleife	zanka	petlja
M			
Mask	Maske	maska	maska
Microcomputer	Mikrocomputer	mikro računalnik	mikro računar
Microprocessor	Mikroprozessor	mikroprocesor? mikroobravnovalnik	mikroprocesor
Monitor	Monitor	monitor? povezovalnik, zveznik	monitor
Moving-Head Disk	Platte mit beweglichem Kopf	disk z gibljivimi glavami	disk sa pokretnom glavom
Multiplexer	Multiplexer	multipleksor	multipleksor
N			
Network	Netzwerk	omrežje	mreža

Noise	Störung	šum	šum
O			
Opcode	Befehlsmode, -code	koda operacije, kod operacije	kod operacije
Output	Ausgang	izhod	izlaz
Overlap	Überlappen	prekrivati se	preklapanje
P			
Package	Paket	paket	paket
Panel	Bedienungspult	pult? panel?	panela
Paper-tape	Lochstreifen	luknjani trak	papirna traka
Parity	Parität	parnost	paritet
Paripheral	Peripheres Gerät	periferna naprava, zunanja naprava	periferal
Pin	Anschlusztift	nožica, priključek	izvod
Plotter	Zeichengerät	risalnik	crtač
Pointer	Zeiger	kazalec (kazalnik?)	pokazivač
Power	Leistung	moč	napajanje, snaga
Power-Failure	Stromausfall	izpad napajanja	pad, kvar napajanja
Power Start	Einschalten	vklop napajanja	uključivanje napajanja
Power Supply	Spannungsversorgung	napajalnik	napajanje
Program Counter	Programmzähler	programski števnik	programski brojač
Punch	Stanzer, stanzen	luknjati	bušiti
R			
Rack	Gerätegestell	stojalo	okvir
RAM	Schreib-Lesespeicher	pomnilnik(z naključnim dostopom)	memorija sa slučajnim pristupom
Range	Bereich	obseg	opseg
Reader	Leser	bralnik, čitalnik?	čitač
Real-time Clock	Echtzeituhr	ura za realni čas	sat realnog vremena
Reed Relay	Schutzgasrelais	hermetični kontaktnik	zaštiteni rele
Register	Register	register	registar
Reliability	Zuverlässigkeit	zanesljivost	pouzdanost
Request	Anfrage	zahteva	zahtjev
Restart	Restart	ponovitev, ponovni start	ponovni start
ROM	Festwertspeicher, Lesespeicher	bralni pomnilnik	memorija za očitavanje
S			
Saving	Ersparnis	prihranek	sačuvati
Scheduling	Arbeitszeiteinteilung	razdeljevanje(delovnega časa)	rasporedjivanje
Sensor	Fühler	senzor	senzor
Sequencing	Nachfolgend	razvrščanje	sekvenciranje
Sequential	Aufeinanderfolgend	razvrščen	sekvencijalni
Serial Transfer	Serielle Übertragung	zaporeden prenos	serijski prenos
Shared	Geteilt	porazdeljen	zajednički
Shift	Schieben	pomik	pomjeraj
Silicon	Silizium	silicij	silikon
Simulator	Simulator	simulator	simulator
Slot	Steckerplatz	reža	priključnica
Software	Software	programska oprema, softver?	software
Speed	Gechwindigkeit	hitrost	brzina
Static	Statisch	statičen, trajen	statički
Stack	Stapel	kopica, odlagalnik	stek
Start	Start	štart	start
State/Status	Zustand	stanje	stanje
Subsystem	Untersystem	podsistem	podsistem
Switching	Schaltend	preklapljanje, preklopen	prekidanje
Synchronous	Synchron	sočasen	sinhroni
T			
Tape	Band	trak	traka
Tape Driver	Bandantrieb	tračni gonilnik	jedinica za pogon trake
Task	Aufgabe	naloga	posao
Terminal	Terminal	terminal, priključek	terminal
Threshold	Schwelle	prag	prag
Time-sharing	Time-sharing	delitev časa	dijeljenje vremena
Track	Spur	sled	trag

Trap	Falle	past	trap
Transfer Rate	Übertragungsgeschwindigkeit	prenosna hitrost, hitrost prenosa	brzina prenosa
U			
Utility	Dienstprogramm	služnost, služnosten	uslužni program
V			
Vectored	Vektorisiert	vektoriziran	vektorski
Voltage	Spannung	napetost	napon
W			
Way	Weg	pot	način
Width	Weite	širina	širina
Word	Wort	beseda	riječ
Y			
Yield	Ausbeute	izplen	davanje

kratice, ki jih pogosto srečujemo

RTL - Resistor transistor logic, Integrirana vezja
DTL - Diode transistor logic ,
TTL - Transistor, transistor logic ,
ECL - Emitter coupled logic,
HTL - High threshold logic,
SSI - Small scale integration, manj kot 25 logičnih vrat v enem čipu.
MSI - Medium scale integration, običajno 25 ali več logičnih vrat v enem čipu.
LSI - Large scale integration, več kot 100 logičnih vrat na čip
I^2L - Integrated injection logic, bipolarna oblika LSI.
CCD - Charge coupled device,
MOS - Metal Oxide Semiconductor
NMOS - N channel MOS.
PMOS - P channel MOS.
CMOS - Complementary MOS, tako N kot P vrste.

literatura in srečanja

OKTOBER

oktober, Beograd, Jugoslavija

NOVI PRAVCI RAZVOJA HIBRIDNIH RAČUNARSKIH MAŠINA
NEW TRENDS OF HYBRID COMPUTERS DEVELOPMENT
Organizator in informacije: Institut Mihailo Pupin, Volgina 15, 11000 Beograd

oktober, Beograd, Jugoslavija

SEMINAR O INDUSTRIJSKIM ROBOTIMA I MANIPULATORIMA
INDUSTRIAL ROBOTS AND MANUPULATORS
Organizator in informacije: Institut Mihailo Pupin, Volgina 15, 11000 Beograd

oktober, Bled, Jugoslavija

SIMPOZIJUM O SASTAVNIM DELOVIMA
SYMPOSIUM ON THE COMPONENTS
Organizator in informacije: Jugoslovenski komitet za ETAN, Kneza Miloša 9, 11000 Beograd, telef. 011 33 957

3-5 oktober, Bonn, ZRN

THIRD INTERNATIONAL SYMPOSIUM ON MODELING AND PERFORMANCE AND EVALUATION COMPUTER SYSTEMS
Organizator: GMD/IRIA/IFIP
Informacije: Mr.P.Hayderhoff-G.M.D. Postfach 1240 d-5205 St. Augustin, Germany

3-5 okt. Düsseldorf, ZRN

SECOND SYMPOSIUM ON CONTROL IN POWER ELECTRONICS AND ELECTRICAL DRIVES
Organizator: IFAC
Informacije: VDI/VDE Gesellschaft Mess-und Regelungstechnic, Postfach 1139, D-4000 Düsseldorf 1, Federal Republic of Germany

3-6 okt. Amsterdam, Nizozemska

EUROMICRO 3rd Symposium on Microprocessing and Microcomputing
Organizator: EUROMICRO
Informacije: J.D. Nicoud LCD-EPFL Bellerive 16 CH-1007 Lausanne, Switzerland

3-8 okt. Bled, Jugoslavija

SIMPOZIJ IN SEMINARJI INFORMATICA 77
Organizator in informacije: Slovensko društvo Informatika, Jamova 39, 61000 Ljubljana, telef. 061/ 63 261 Jùgoslavija

5-7 okt. Niagara Falls, New York, ZDA

MICRO 10: TENTH ANNUAL WORKSHOP ON MICROPROGRAMMING
Organizator: ACM SIGMICRO, IEEE-CS
Informacije: ACM HQ, 1133 Avenue Of the Americas, New York, NY 10036,USA

6-8 okt. Tokyo, Japonska

THIRD INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES
Organizator in informacije: IFIP

6-12 okt. Düsseldorf, ZRN

INTERKAMA 1977
INTERNATIONAL EXHIBITION OF MEASURING TECHNIQUES AND AUTOMATICS, CONFERENCE
Organizator in informacije: Düsseldorfer Messegesellschaft m6H-NOWEA 4 Düsseldorf 30, Postfach 320 203 BRD

9-12 okt. Washington, DC, ZDA

INFO /EXPO 77
Organizator: Data Processing Management Association
Informacije: DPMA, 505 Busse Highway, Park Ridge, IL 60068, USA

10-12 okt. Chicago, Illinois, ZDA

NATIONAL ELECTRONICS CONFERENCE AND NATIONAL COMMUNICATIONS FORUM
Organizator: National Engineering Consortium Inc.
Informacije: National Electronics Conference, Oak Brook Executive Plaza 1, 1301 West 22nd Street, Oak Brook, IL 60521, USA

10-13 okt. Varna, Bolgarija

IFAC WORKSHOP ON CONTROL OF MANAGEMENT SYSTEMS
Organizator: National Centre for Cybernetics and C omputer Techniques, Bulgaria
Informacije: National Centre for Cybernetics and Computer Techniques of the Committee for Science, Technical Progress and Higher Education, 8, Slavyanska Street,Sofia, Bulgaria

16-19 okt. Seattle, Washington, ZDA

ACM 77
Organizator: ACM
Informacije: ACM Headquarters,1133 Avenue of the Americas, New York, NY 10036, USA

17-20 okt. Tokyo, Japonska

IFAC SYMPOSIUM ON INFORMATION-CONTROL PROBLEMS IN MANUFACTURING TECHNOLOGY
Organizator: IFAC
Informacije: IFAC Manufacturing Technology Symposium c/o The Society of Instrument and Control Engineers 39 Shiba Kotohira-Cho Minata-Ku, Tokyo, Japan

17-21 okt. Sao Paulo, Brazilija

TENTH NATIONAL CONGRESS OF DATA PROCESSING
Organizator: SUCESU
Informacije: Renato A Mazzola, Director Executivo, Av Paulista 1159-14^{O} andar-conjs. 1404/5, Sao Paulo, Brazil

17-21 okt. München, ZRN

INTERNATIONAL SEMINAR AND PROFESSIONAL EXHIBITION -COMPUTER SYSTEMS AND ITS APPLICATIONS
Organizator in informacije:Münchner Messe und Ausstellungsgeselshaft mbH 8 Postfach 121 009, BRD

19. okt. Herceg Novi, Jugoslavija

SIMPOSIJ O PRIMENI OPERACIONIH ISTRAŽIVANJA
Organizator in informacije: Fakultet organizacionih nauka (biblioteka), 11091 Beograd, Ul. Oslobodjenja 1

20-21 okt. Rocquencourt, Francija

SEMINAR ON STANDARDIZATION IN INFORMATICS
Organizator: IRIA, GMD,NCC
Informacije: IRIA, Service des Relations Extérieurs, Domaine de Voluceau, Rocquencourt, BP 5, 78150 Le Chesnay, France

24-25 okt. Turku, Finska

SIXTH NORDIC CONGRESS ON OPERATIONS RESEARCH
Organizator: Finish Operations Research Society, Abo Swedish University-School of Economics
Informacije: The Secretary, Programme Committee, Handelshögskolan vid Abo Akademi, Henriksgatan 7, 20500 Abo 50, Finland

24 okt.-27 nov. Philippines, Japonska in Hong Kong

ADVANCED COURSE ON COMMUNICATION AND INFORMATION
Organizator: Asian Productivity Organization (APO)
Informacije: APO, 4-14 Aksaka 8-chome, Minato-Ku, Tokyo 197, Japan

24-28 okt. Paris, Francija

FOURTEENTH SESSION OF THE ICIREPAT TECHNICAL COMMITTEE FOR STANDARDIZATION
Organizator: World Intellectual Property Organization (WIPO)
Informacije: WIPO, 32 Chemin des Colombettes, 1211 Geneva 20, Switzerland

25-27 okt. Compiègne, Francija

IFAC WORKSHOP ON INFORMATION AND SYSTEMS
Organizator: IFAC
Informacije: Prof B Dubuisson, Université de Technologie de Compiègne, Département MAI, BP 233, 60206 Compiègne, France

NOVEMBER

9-11 nov. Berlin, ZRN

1977 EUSIDIC CONFERENCE
Organizator: European Association of Scientific Information Dissemination Centres
Informacije: Dr C Weiske, Chemie Information und Dokumentation Berlin, Postfach 126050, Steinplatz 2, 1000 Berlin 12, BDR

14-15 nov. Austin, Texas, ZDA

SIXTH TEXAS CONFERENCE ON COMPUTING SYSTEMS
Organizator: University of Texas, ACM, IEEE-CS
Informacije: ACM HQ, 1133 Avenue of the Americas, New York, NY 10036, USA

16-18 nov. West Lafayette, Indiana, ZDA

SIXTH SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES
Organizator: ACM SIGOPS
Informacije: Saul Rosen (Gen Chm), Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA

22-24 nov. Versailles, Francija

MODELISATION ET MAITRISE DES SYSTEMS TECHNIQUES, ECONOMIQUES, SOCIAUX
Organizator in informacije: AFCET, 156, Bulevard Pereire, 75017 Paris, France

26-29 nov. Cairo, Egipt

SECOD IFAC INTERNATIONAL CONFERENCE ON SYSTEMS APPROACH FOR DEVELOPMENT
Organizator: IFAC
Informacije: Eng. Sayed Abdel Kader El Sheshe Secretary of Egyptian High Committee of Automatic Control (EHCAC), 6Khail Agha Street, Garden City, Cairo, Egypt

28-30 nov. Amsterdam, Nizozemska

MANAGERS, DATABASES AND INFORMATION SYSTEMS
Organizator: The IFIP group for Applied Information Processing in Management and Administration
Informacije: IAG Headquarters, 40, Paulus Potterstraat 1071 DB Amsterdam, The Netherlands

29 nov.-2dec. Washington, DC, ZDA

1977 SIGMETRICS/CMG VIII: AN INTERNATIONAL CONFERENCE ON COMPUTER PERFORMANCE
Organizator: ACM Special Interest Group on Measurement and Evaluation, the Computer Measurement Group Inc
Informacije: Robert L Morrison, IBM Corp (B617072), PO Box 390, Poughkeepsie, NY 12602, USA

november, Enschede, Nizozemska

IFAC WORKSHOP ON CASE STUDIES IN THE HUMANIZATION OF AUTOMATION
Organizator: IFAC
Informacije: Koninklijk Instituut van Ingenieurs, Afdeling voor Regeltechniek, c/o Ir.F. Meiring, vakroep NR gebouws WandS, Technische Hogeschool, Eindhoven, The Netherlands

november, New Brunswick, New Jersey, ZDA

IMACS/IFAC WORKSHOP ON PHYSIOLOGICAL SYSTEM MODELS
Organizator: IMACS, IFAC
Informacije: American Automatic Control Council, c/o Mr M A Keyes, Vice-President, Bailey Meter Company, 29801 Euclid Avenue, Wickliffe, Ohio 44092, USA

DECEMBER

5-7 dec. Gaithersburg, Maryland, ZDA

WINTER SIMULATION CONFERENCE
Organizator: National Bureau of Standards-Us Department of Commerce
Informacije: Dr Robert G Sargent (Gen Chm), Syracuse University, Syracuse, NY 13210, USA

5-9 dec. Rocquencourt, Francija

THIRD INTERNATIONAL SYMPOSIUM ON COMPUTING METHODS IN APPLIED SCIENCES AND ENGINEERING
Organizator: IRIA-LABORIA
Informacije: IRIA, Service des Relations Extérieures, Domaine de Voluceau, Rocquencourt, BP 5, 78150.Le Chesnay, France

5-9 dec. Versailles, Francija

THIRD INTERNATIONAL COLLOQUIUM ON METHODS OF SCIENTIFIC AND TECHNICAL CALCULATION
Organizator: IRIA
Informacije: IRIA, Service des Relations Extérieures, Domaine de Voluceau, Rocquencourt, BP 5, 78150 Le Chesnay, France

LETO 1978

23-25 jan. Tuscon, Arizona, ZDA

FIFTH ANNUAL ACM SIGACT-SIGPLAN SYMPOSIUM ON PRINCIPLES OF PROGRAMMING LANGUAGES
Organizator: ACM
Informacije: Prof Larry Reeker and Prof David Ripley, Department of Computer Science, University of Arizona, Tuscon, AZ 85721, USA

23-27 jan. Valparaiso, Čile

FIFTH PANEL DISCUSSION ON COMPUTATION TOPICS
Organizator: Computation Centre-Valparaiso Catholic University
Informacije: Aldo Migliaro, Director, Centro de Computación, Universidad Católica de Valparaiso, Casilla 4059, Valparaiso, Chile

13-15 feb. Liège, Belgija

SYMPOSIUM ON COMPUTER NETWORK PROTOCOLS
Organizator: The University of Liège
Informacije: A Danthine, Symposium on Computer Network Protocols, Avenue des Tilleuls 49, B-4000 Liège, Belgium

23-24 feb. Detroit, Michigan, ZDA

SIGCSE/CSA SYMPOSIUM
Organizator: ACM Special Interest Group on Computer Science Education, The Computer Science Association
Informacije: Kenneth Williams, SIGCSE/CSA Symposium Chairman, Computer Science Group, Western Michigan University, Kalamazoo, MI 49008, USA

7-10 marec Paris, Francija

INTERNATIONAL CONGRESS ON THE CONTRIBUTION OF COMPUTERS TO THE DEVELOPMENT OF CHEMICAL ENGINEERING AND INDUSTRIAL CHEMISTRY
Informacije: Monsieur Le Professeur A Brusset, Congrès International 1978, Societé de Chemie Industrielle 28, rue Saint Dominique F-75007 Paris, France

15-17 marec Tampa, Florida, ZDA

ELEVENTH ANNUAL SIMULATION SYMPOSIUM
Organizator: ACM-SIGSIM, IEEE-CS
Informacije: ACM HQ, 1133 Avenue of the Americas, New York, NY 10036, USA

27-30 marec, New York, ZDA

INTERNATIONAL CONVENTION AND EXHIBITION OF THE INSTITUT OF ELECTRICAL AND ELECTRONICS ENGINEERS
Organizator: IEEE
Informacije: J.H. Shumacher, IEEE, 345 East, 47th Street, New York, NY 10017, USA

9-12 maj, London, Velika Britanija

EUROCOMP 78
Organizator: Online Conferences Limited
Informacije: EUROCOMP 78, Online, Cleveland Road, Uxbridge UB8 2DD, England

22-25 maj, Taormina, Sicilija

SIXTH INTERNATIONAL CODATA CONFERENCE
Organizator: CODATA
Informacije: CODATA Secretarat, 51 Boulevard de Montmorency, 75016 Paris, France

maj, Bled, Jugoslavija

CONFERENCE ON COMPUTERS IN BANKING AND FINANCE
Organizator: IAG
Informacije: IAG HQ, Paulus Potterstraat 40, Amsterdam -1007, The Netherlands

11-17 junij, Helsinki, Finska

IFAC CONGRESS
Organizator: Finish Society of Automatic Cont
Informacije: Mr Olli Pezolanti, Höyläämotie 1 Helsinki 38, Finland

21-23 junij, Toulouse, Francija

1978 INTERNATIONAL SYMPOSIUM ON FAULT TOLERANT COMPUTING-FTCS-8
Organizator: FTC Technical Committee of the Institute for Electrical and Electronics Engineers Computer Society
Informacije: IEEE, 345 East 47th Street, New York, NY 10017, USA

11-15 junij, Prague, Češkoslovaška

IFAC/IFIP 2nd INTERNATIONAL SYMPOSIUM "SOFTWARE FOR COMPUTER CONTROL"
Organizator in informacije: IFAC/IFIP TC 5

6-8 avg. Jerusalem, Izrael

JERUSALEM CONFERENCE ON INFORMATION TECHNOLOGY
Informacije: IFIP Secretariat,3 rue de Marché, 1204 Geneva, Switzerland

21-25 avg. Leiden, Nizozemska

COMPSTAT 1978
Organizator: Vereniging voor Statistiek, Netherlands Society for Statistics, Biometrics, Econometrics and Operational Research
Informacije: COMPSTAT 1978, c/o Central Reken Institut, University of Leiden, Wassenaarseweg 80, Leiden, The Netherlands

september, Berlin, ZRN

THIRD INTERNATIONAL CONGRESS ON ELECTRONIC INFORMATION PROCESSING (IKD)
Organizator: AMK- Berlin, IKD Professional Commission
Informacije: AMK-Berlin, Ausstellungs-Mess-Kongress-GmbH, Messedamm 22, D-1000, Berlin 19, Germany

4-8 sept. MEDICAL INFORMATICS EUROPE: FIRST CONGRESS OF THE EUROPEAN FEDERATION FOR MEDICAL INFORMATICS
Organizator: European Federation for Medical Informatics
Informacije: Dr B Barber, Management Services Division, St Faith's Hospital, London Road, Brentwood, Essex, UK

4-8 sept. Manila, Filipini

SOUTH EAST ASIA REGIONAL COMPUTER CONFERENCE 1978 (SEARCC 78)
Organizator: Singapore Computer Society
Informacije: Robert Iau, President, Singapore Computer Society, c/o Central Provident Fund Board, Robinson Road, Singapore 1, Rep. of Singapore

avtorji in sodelavci

SUAD ALAGIĆ (1946)

Diplomirao na Elektrotehničkom fakultetu u Sarajevu marta 1970. Magistrirao na Masačusetskom univerzitetu u Amherstu, SAD, septembra 1972. Tema: Semantika algoritamskih jezika. Doktorirao na Masačusetskom univerzitetu u Amherstu februara 1974. Tema: Algebarski aspekti programskih i formalnih jezika. Docent Elektrotehničkog fakulteta u Sarajevu. Bavi se računarskim softverom, posebno programskim jezicima i metodologijom programiranja, zatim strukturama podataka i bazama podataka. Objavio je nekoliko radova u eminentnim svjetskim časopisima i veći broj radova u zemlji. Autor je knjige "Principi programiranja", koju je izdala "Svjetlost" 1976.

DANČE DAVČEV (1949)

Diplomirao na Elektrotehničkom fakultetu u Beogradu 1972 godine. Nakon odbrane teze "Interpretator APL na Mitri 15" na univerzitetu u Parizu-Centar za elektroniku ORSAY 1975 godine,stekao je naziv doktora-inženjera. Glavna dosadašnja delatnost bila je na području informatike. Radio je na interpretatoru APL, asemblerskim jezicima i na problemima upravljanja terminala Tektronix preko računala. Trenutno radi kao asistent na Elektrotehničkom fakultetu u Skopju. Bavi se sa problemima vodjenja elektroenergetskog sistema SRM preko procesnog računala, sa problemom pretvaranja neelektričnih veličina u električne u sistemima za automatsku obradu podataka i sa istraživanjem vrednovanja APL sistema u odnosu na druge sisteme.

JOZO DUJMOVIĆ (1941)

Diplomirao, magistrirao i doktorirao na Elektrotehničkom fakultetu Univerziteta u Beogradu. Početkom 1964. godine zaposlio se u Institutu "Mihailo Pupin" u Beogradu, a početkom 1967. godine prešao je na Elektrotehnički fakultet Beograd, gde je sada zaposlen kao docent za oblast Računarska tehnika i informatika. Publikovao do sada preko 50 naučnih i stručnih radova iz oblasti računarske tehnike, informatike, primenjene matematike i elektronike. Nagradjivan za naučni rad: Aprilska nagrada Beogradskog univerziteta (kao student) i nagrade kongresa ETAN 1970 i INFORMATICA (FCIP) 1970. Od 1969. godine intenzivno se bavi razvojem metodologije za vrednovanje računara i šire klase srodnih složenih sistema. Metoda koju je razvio uspešno se primenjuje u oblasti profesionalnog vrednovanja i izbora računarskih sistema.

MARKO KOVAČEVIĆ (1953)

Diplomirao 1976 godine na Fakulteti za elektrotehniko v Ljubljani, smjer računalništvo in informatika. Zaposlen na odjeljenju za elektroniku instituta Jožef Stefan u Ljubljani. Dosadašnje i trenutno područje rada su operacijski sistemi te softwerska i hardverska problematika mikro računara. Tema diplomskog rada bila je : Izrada makro procesora za asemblerski jezik mikro računara. Saraduje sa našim uredništvom, posebno pri rubriki novice in zanimivosti.

NEDA PAPIĆ (1951)

Studentka IX semestra Fakultete za elektrotehniko v Ljubljani, smeri računalništvo in informatika. Trenutno

dela na področju mikro računalnikov. Pri Slovenskem društvu Informatika je generalna tajnica in sodeluje pri vseh akcijah društva. Na Fakulteti za elektrotehniko v Ljubljani je član OOZSMS. V časopisu vodi rubriko "študentska vprašanja