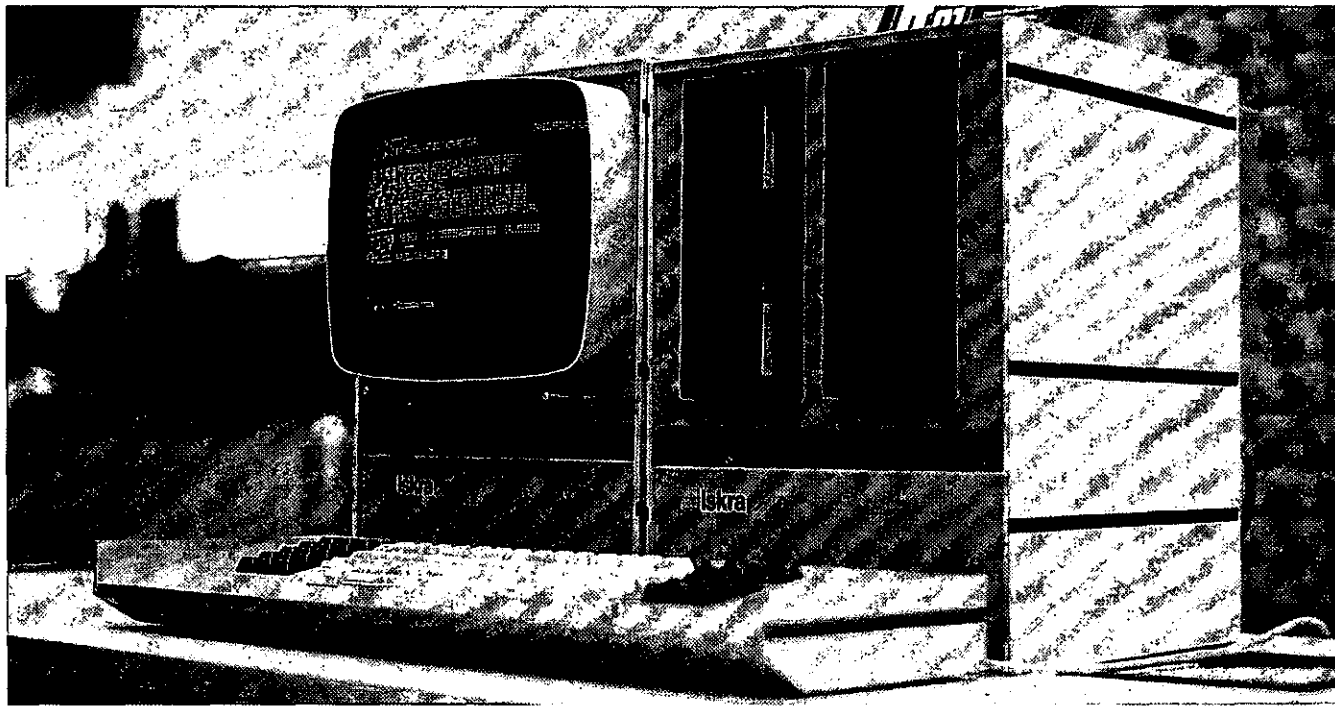
YU ISSN 0350-5596

IskraData

Družina računalniških sistemov za distribuirano obdelavo

ISKRADATA 80-50 **Pisalnik**
ISKRADATA 80-60 **Samostojno delovno mesto**
ISKRADATA 80-70 **Komunikacijsko delovno mesto**
ISKRADATA 80-75 **Sinhroni terminal**
ISKRADATA 80-80 **Samostojni matični sistem**
ISKRADATA 80-90 **Komunikacijski matični sistem**



NAJVAŽNEJŠE ZNAČILNOSTI SISTEMA ISKRADATA

- Modulnost sistema
- Prilagodljivost sistema uporabnikovim zahtevam
- Distribuirana obdelava, ki omogoča popolnoma samostojno delo delovnega mesta in dostop do baze podatkov v matičnem računalniku ali nekem drugem računalniku, ki je vključen v omrežje
- 1—16 inteligenčnih delovnih mest
- Disketna enota omogoča direkten pristop do vsakega podatka
- Za večjo količino podatkov se priključijo na matični računalnik do 4 diskovne enote (40, 80, 160, 200, 300 MB)
- Široka paleta možnosti priključitve perifernih naprav (standardni zaporedni in vzporedni vmesnik RS 232-V 24)
- Sinhroni protokoli ali asinhroni protokoli s kontrolo ali brez nje
- Učinkovita programska jezika BASIC in PASCAL
- Delovno mesto je nezahtevno za okolje
- Večja učinkovitost z manjšimi stroški.

Iskra Elektromehanika Kranj TOZD Tovarna računalnikov

informatica

Časopis izdaja Slovensko društvo INFORMATIKA, 61000 Ljubljana, Parmova 41, Jugoslavija

Uredništvo in uprava: Informatica, Parmova 41, 61000 Ljubljana, telefon (061) 312-988, teleks: 31366 YU DELTA

Letna naročnina za delovne organizacije je 500,00 din, za redne člane 200,00 din, za študente 100,00/50,00 din, posamezne številke 100,00 din
Žiro račun št.: 50101-678-51841
Stališče uredništva se lahko razlikuje od mnenja avtorjev.

Pri financiranju revije sodeluje tudi Raziskovalna skupnost Slovenije.

Na podlagi mnenja Republiškega sekretariata za prosveto in kulturo št. 4210-44/79 z dne 1.2.1979, je časopis oproščen temeljnega davka od prometa proizvodov.

Tisk: Tiskarna KRESIJA, Ljubljana

Grafična oprema: Rasto Kirn

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

YU ISSN 0350-5596

LETNIK 5, 1981 – št. 4

VSEBINA

	1	Pojav informacijskega gospodarstva
A.P.Železnikar	9	Uvod v CP/M II
S. Brajović - Bratanović B. Džonova - Jerman-Blažič	24	Standardi i politika standardizacije u oblasti informatike
M. Kovačević	28	Sistemska obnova u uslovima realnog vremena
S. Prešern I. Ozimek M. Špegel	33	Mikroprocesorsko vodenje senzorskega sistema za robotsko varjenje
I. Lončar	36	Primjena transpozicione metrike za proračun optimalnog balansa
V. Smolej	39	Plus ca change, plus c'est la même chose
M. A. Volk	43	Fortran 8X-revizija Fortran-a 77
I. Lončar	50	Primjena leksikografske metrike za proračun debalansa lopatica zrakoplovnih turbina
N. Hadjina V. Čerić	54	Simulacioni model za evaluaciju performansi računarskih sistema
I. Šilc B. Mihovilović P. Kolbezen	60	Mehurčni pomnilniki - IV.del
I. Tvrdy M. Martinec R. Reinhardt	68	Peto republiško tekmovanje srednješolcev s področja računalništva
	72	Uporabni programi
	80	Novice in zanimivosti

informatica

Published by INFORMATIKA, Slovene Society for Informatics, 61000 Ljubljana, Parmova 41, Yugoslavia

Headquarters: Informatica, Parmova 41, 61000 Ljubljana, Phone: (061) 312-988, Telex: 31366 Delta

Annual subscription rate for abroad is US $ 22 for companies, and US $ 7,5 for individuals.

Printed by: Tiskarna KRESIJA, Ljubljana

DESIGN: Rasto Kirn

JOURNAL OF COMPUTING AND INFORMATICS

YU ISSN 0350-5596

VOLUME 5, 1981 — No. 4

CONTENTS

	1	Emergence of Information Economy
A.P.Železnikar	9	Introduction to CP/M
S.Brajović-Bratanović B.Džonova-Jerman-Blažič	24	Standards and Standardisation Policy in the Field of Informatics and Computer
M. Kovačević	28	System Recovery in Real Time
S. Prešern I. Ozimek M. Špegel	33	Microcomputer Control of Sensing System for a Welding Robot
I. Lončar	36	An Application of Transposition Metric to Calculation of the Optimal Balance
V. Smolej	39	More it changes, more it stays the same
M.A.Volk	43	Fortran 8X-Revision of Fortran 77
I. Lončar	50	An Application of the Lexicografic Metric to Calculation of the Balance of the Airplane Turbine
N. Hadjina V. Čerić	54	A Simulation Model to Evaluation of Computer Systems Performances
I. Šilc B. Mihovilović P. Kolbezen	60	Magnetic Buhble Memories-Part 4
I. Tvrdy M. Martinec R. Reinhardt	68	Fifth Computer Science Contest for High School Students
	72	Programming Quickies
	80	News

POJAV INFORMACIJSKEGA GOSPODARSTVA

UDK: 659.2

Dr. Márc Uri Porat je kot direktor programa "Komuniciranje in družba" leta 1978 objavil v reviji "Economic Impact" v rubriki "Inovacije v komunikacijah" članek z naslovom "Pojav informacijskega gospodarstva" (Emergence of Information Economy). Zaradi izredne aktualnosti članka, predvsem pa zaradi ugotovitev ozirona spoznanj do katerih je prišel avtor v zvezi s prepletenostjo (soodvisnostjo) razvoja informatike ter družbenoekonomskih in družbenopolitičnih odnosov, povzemamo omenjeni članek skoraj v celoti.

Dr. Porat, ki je diplomant stanfordske univerze, z magisterijem iz ekonomije in doktoratom iz komunikacij, je začel in uspel končati pomemben znanstveni projekt "Informacijska ekonomija" za potrebe Ministrstva za trgovino (1975-77). Poleg tega svetuje določenemu številu ameriških mednarodnih skupin v informacijski politiki in stalno opravlja dolžnosti direktorja programa humanističnih študij v zvezi s komunikacijami in družbo na aspenskem institutu.

Kakor je plug napovedal prihod dobe poljedelstva, zdaj računalniki in telekomunikacije potiskajo Združene države v "informacijsko gospodarstvo", pravi avtor, ki opozarja na dejstvo, da je danes približno polovica vseh delavcev v Združenih državah zaposlena v aktivnostih, ki so povezane z informacijami in komunikacijami.

Informacijskemu gospodarstvu ne bo zmanjkalo "naravnih bogastev", te nevarnosti ni, kajti "informacije se obnavljajo v neskončnost", avtor pa nas opozarja na mednarodni problem, ki se je pojavil s širjenjem informacijskih dobrin in storitev; ta problem je širok in se razteza od vprašanja varovanja zasebnosti do občutljivosti neke države v zvezi s "kulturnim imperializmom".

Danes predstavljajo Združene države Amerike gospodarstvo, ki sloni na informacijah. Od leta 1967 izvira 25% bruto nacionalnega proizvoda (BNP) iz proizvodnje, predelave in distribucije informacijskih dobrin in storitev. Poleg tega, več kot 21% BNP izvira iz opravljanja informacijskih storitev v okvirih zasebnih in državnih uradov in namenjenih povsem za notranjo uporabo. Od leta 1970 je skoraj polovica vse delovne sile v Združenih državah klasificirana kot "informacijski delavci", ker se udejstvujejo v službah, kjer je proizvodnja ali distribuiranje s šimboli glavna aktivnost. Zaslužili so preko 53% vsega prihodka od dela. Te ugotovitve so pojasnjene in dokumentirane v poročilu (obsega 9 zvezkov), ki je bilo pred kratkim pripravljeno za Ministrstvo za trgovino Združenih držav Amerike.

Iz te transformacije gospodarstva v Združenih državah pa izhajajo številne mednarodne implikacije. Med važnejše, ki zaskrbljujejo sodijo: Kako se izvoz informacijskih dobrin in storitev prilagaja zunanji politiki? Kaj je "izvoz kulture" in kako nanj gledajo od zunaj? Vprašanje človekovih pravic in uporaba informacijskih tehnologij? Kako pomemben je izvoz tehnoloških in znanstvenih informacij?

Kakorkoli že, preden se lotimo raziskovanj teh vprašanj, bomo orisali osnovne vzroke, ki so nas privedli do zaključka, da so Združene države resnično sredi gospodarske in socialne transformacije, za seboj puščajo industrijsko in vstopajo v "informacijsko družbo".

Domači pregled

Industrijske panoge, ki proizvajajo, predelujejo ali posredujejo znanje, komunikacijske in informacijske dobrine in storitve, smo imenovali "primarni informacijski sektor". Kar zadeva storitve, zajema ta industrija elektronske in tiskarske medije, tu so propaganda, izobraževanje, raziskave in storitve v zvezi z razvojem telekomunikacij, finančne storitve in storitve zavarovanja, knjižnice, svetovalna, raziskovalna in razvojna podjetja. Kar zadeva dobrine, te vključujejo računalnike, komunikacijske in elektronske naprave, pisarniške stroje, merilne in kontrolne naprave, tisk in tiskarske stroje. Podrobno ocenjeni dosežki v letu 1967 pri več kot 70 industrijskih dejavnostih in več kot 6.000 proizvodov (merjenih po standardni industrijski klasifikaciji), so pokazali, da izvira 25% BNP iz primarnega informacijskega sektorja. Tudi druga četrtina BNP je vezana na informacijsko dejavnost in sicer v zvezi s prodajo storitev in dobrin.

Poleg tega vemo, intuitivno, da neinformacijska podjetja in vladne institucije "proizvajajo in konzumirajo" informacije, ki so povsem "internega" značaja. Skoraj vsaka institucija se pri svojem delu poslužuje različnih informacij - s področja raziskovanja in razvoja, načrtovanja, upravljanja, računanja, pravnih in administrativnih storitev ter informacij o položaju na trgu. Podjetja in vlade najemajo "informacijske delavce" (menežerje in sekretarje) in investirajo v "informacijski kapital" (računalništvo, komunikacijske in pisarniške naprave in stroje). To so v bistvu informacijski prispevki neinformacijskim aktivnostim. Netržne informacijske storitve (te niso v obtoku na uradnem tržišču) pa označujemo kot

"sekundarni informacijski sektor". Podrobna ocena nam pove, da so te aktivnosti, pogosto povezane s prej omenjenimi zasebnimi in državnimi uradi, dale v 1967 kakšnih 21% BNP. Tako sta v enem letu primarni in sekundarni informacijski sektor prispevala k formalnemu nacionalnemu prihodku in proizvodu v višini 46% BNP. Predpostavimo pa lahko, da je bila številka v letu 1977 nekoliko višja.

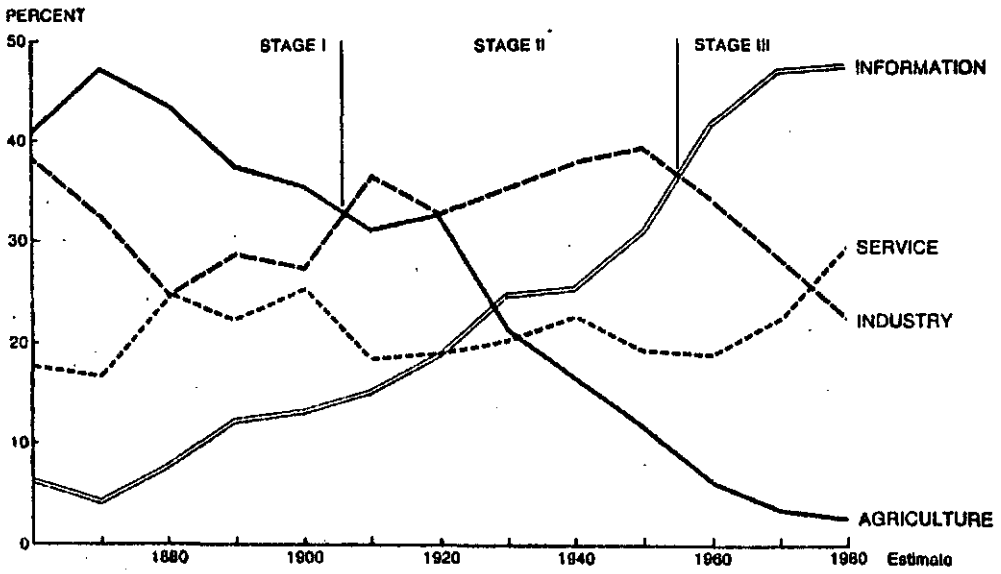
Značilnost gospodarstva pa je lahko ponazorjena tudi s porazdeljenostjo delovne sile v posameznih aktivnostih. Konvencionalna klasifikacija dela je shematično prikazana s tremi sektorji: poljedelstvo, industrija in storitve. Toda, če uvrstimo še četrti sektor - informacije - in prištejemo še vse delavce, ki opravljajo "informacijske posle", se pojavi nadvse zanimiva slika.

"tehnotronska doba", za R.Dahrendorffa je "post-kapitalistična" in za A.Ethioni-ja "post moderna" doba. D.Bell je prvi skoval termin "post-industrijska" doba in jedrnato opisuje bistvo transformacije z naslednjim:

"V predindustrijskih družbah - danes še vedno stanje v katerem je večina sveta- je delovna sila prekomerno zaposlena v izkoriščanju naravnih bogastev: rudarstvu, ribištvu, gozdarstvu, poljedelstvu. Življenje je prvenstveno borba z naravo. Človek dela zgolj fizično, na zastarele načine, kakor v davnini. Njegovo pojmovanje sveta je pogojeno z odvisnostjo od naravnih sil (letni časi, kakovost prsti, količina vode).

Industrijske družbe - posebno tiste okrog severno atlantskega primorja skupaj s Sovjetsko zvezo in Japonsko - pa so družbe, ki proizvajajo dobrine. Življenje (zdaj) je borba s

FOUR SECTOR AGGREGATION OF THE U.S. WORK FORCE BY PERCENT
1860 - 1980
(Using median estimates of information workers)



SOURCE: Porat, "The Information Economy" U.S. Dept. of Commerce based on Census Bureau and Bureau of Labor Statistics.

Diagram prikazuje porazdelitev delovne sile v Združenih državah. Prvo fazo (1860-1906) predstavlja poljedelstvo, ki je hkrati edina in največja skupina. V času te stopnje razvoja so bile Združene države poljedelska družba.

V drugi fazi (1906-1954) prevladuje industrijska delovna sila in dosega vrh s približno 40% v letu 1946. V tem obdobju označujemo Združene države kot industrijsko družbo. V petdesetih letih pa zaznamo začetek upadanja delovne sile v industriji (danes predstavlja okrog 25% vse delovne sile).

V tretji fazi (1954-danes) predstavljajo informacijski delavci najštevilnejšo skupino. Z nizkih 5% delovne sile v 1860, je informacijski sektor v Združenih državah zrasel na okrog 46%. V letu 1967 je ta skupina zaslužila preko 53% vsega dohodka zaposlenih.

Na osnovi predhodnih prepričljivih podatkov o BNP in podatkov o celotni delovni sili, lahko Združene države nedvomno imenujemo "informacijska družba". Za ta pojav obstaja mnogo poimenovanj in izrazov. F.Machlup ga imenuje "ekonomija znanja". Z.Brezenskemu pomeni tehnizirano naravo. Svet se je stehniziral in zracionaliziral.

Post-industrijska družba pa temelji na storitvah. Zato je to borba med ljudmi. Kar šteje ni gola fizična sila ali energija ampak informacija. Osrednja oseba je profesionalec, ki je oborožen z izobrazbo in usposobljenostjo, to pa mu zagotavlja uspešno opravljanje vrsto spretnosti, ki se vedno bolj in bolj zahtevajo v post-industrijski družbi. Če industrijsko družbo definiramo s kvantiteto dobrin kot merilom življenjskega standarda, tedaj definiramo post-industrijsko družbo s kvaliteto življenja, ki jo predstavljajo storitve in udobja (zdravje, izobrazba, rekreacija in umetnost), kar si danes vsakdo želi in tudi lahko ima."

Bistvena razlika med poljedelsko, industrijsko in informacijsko družbo je v tem, da so se gospodarske dejavnosti in tehnološke spremembe preusmerile od izdelovanja "predmetov" k "obravnavanju" informacij in simbolov. Plug in poljedelska tehnika sta napovedala poljedelsko gospodarstvo, parni stroj in industrijske (tovarniške) tehnike so spremenile najprej

Evropo, potem pa še Združene države v industrijsko gospodarstvo, računalniki in telekomunikacije pa zdaj porivajo Združene države v informacijsko gospodarstvo.

Koliko so vredne informacije?

Gospodarstvo Združenih držav proizvaja in konzumira veliko količino informacijskih dobrin in storitev. Človek se sprašuje, kako dolgo bo domače gospodarstvo lahko podpiralo te aktivnosti. Informacije so se vrinile in prodrle v skoraj vse veje trgovine. Vladne institucije zahtevajo njihovo proizvajanje pri podjetjih, podjetja jih zahtevajo od drugih podjetij, vsak želi informacije od partnerja in uporabnika. Mnogo ljudi (in industrijskih podjetij) je obogatelo prav zaradi nenavadne "požrešnosti" družbe za informacijami. Agencije, banke, poslovni svetovalci, strojepisnice - vse te službe in delavci v njih bi bili hudo prizadeti, če bi pretok informacij nenadoma ne bil več moderen. Toda pojavlja se vprašanje ali so informacije resnično koristne, ali predstavljajo le nekakšno "breme" za proizvodne sektorje gospodarstva. Ni dvoma, da so lahko informacije donosen vir za ustvarjanje profita in dela, če pa ne bodo "produktivne" pa slej ko prej tudi ne bodo več potrebne.

Preučimo na kratko informacije kot vir podoben naravnim, delovnim in proizvodnim virom. Podjetje potrebuje za svojo proizvodnjo informacijske dobrine in storitve (računalnike, menežerje). Gospodarno podjetje kupi računalnik ali najame menežerja, le če lahko pričakuje presežek iz te "investicije" in povračilo, ki je večje kot izdatek ali pa vsaj tako veliko kot katerakoli druga investicijska naložba. Informacije tako vstopajo kot produkcijski "input" in povečujejo domači "output". Inovacija in uporaba novih informacij požene v tek doslej nedonosne vire, odpira nova tržišča in izboljšuje učinkovitost že obstoječih procesov. V tej zvezi ustvarjajo informacije blagostanje.

Hkrati pa proizvajanje informacij utegne izkoristiti naložbene in delovne vire, ki jih "jemlje" drugim sektorjem gospodarstva, medtem ko drugim zvišuje ceno za dela in naložbe. Ocenjevanje produktivnosti informacijskih delavcev in informacijskih procesov je resnično zelo težko in iz tega izhaja priljubljena, čeprav nedokazana domneva, da je informiranje lahko brezkoristna aktivnost. V tej zvezi bi informacije utegnile negativno vplivati na domači "output".

Odgovor na vprašanje ali informacije prispevajo še kaj k domači proizvodnji in povečujejo njeno učinkovitost ali pa jo obremenjujejo kot nekakšen neproduktiven vir "odtekanja", lahko da le obsežna in naporna raziskava. Domače naloge še nismo naredili. Kakorkoli že, slika o stanju razporejenosti delovne sile v Združenih državah odkriva, da se je rast informacijske delovne sile naglo umirila. Pričakujemo, da bo informacijski sektor v času od 1970 do 1980 zrasel za 0,04% (nič hitreje kot ostala delovna sila v Združenih državah). Navsezadnje je možno, da bo postalo ameriško uradništvo nasičeno ter nezmožno absorbirati še več informacijskih delavcev.

Obeti za novo rast informacijskega gospodarstva izhajajo iz dveh virov zahtev: domačih in tujih. Doma se bo informacijsko gospodarstvo (t.j. primarni informacijski sektor) razširilo z novimi podjetniškimi tveganji (prodor v poslovno in hišno rabo kot so "informacijske storitve", konference na daljavo; ali pa nove tehnologije kot so izdelava mikroprocesorjev, optičnih vlaken in pomnilnikov z veliko zmogljivostjo).

Tuja tržišča so drugi, prav tako zelo opazen vir zahtev. Vztrajno trgovanje ameriških multinacionalnih družb - hkrati v razvitih deželah (vključujoč vzhodni evropski blok) in v deželah v razvoju, je posledica domačega pritiska za večanje izvoza primarnih informacijskih dobrin in storitev. "Produkti" sekundarnega informacijskega sektorja pa so se do sedaj izvažali kot "črni fond", kot del cene neinformacijske dobrine ali točneje v obliki znanstvenega in tehnološkega znanja.

Izvoz informacijskih dobrin

Industrijska družba se znebi svojega otroštva in postane informacijska, ko vsak vidik njenega gospodarstva postane odvisen od informacijskega stroja. Japonci imenujejo ta proces "informatizacija" - precej okoren prevod za prehod neke dežele iz industrijskega v postindustrijsko stanje. V 19. stoletju, ko je poljedelska družba postala odvisna od strojev pa je bila ključna beseda "mehanizacija".

Združene države so se po japonski terminologiji "informatizirale", pa tudi ostali del sveta stopnjuje rabo informacijskih strojev resnično s hitrimi koraki. Medtem, ko se zahodno evropske industrializirane države še preoblikujejo, industrijske panoge, ki proizvajajo informacijske dobrine v Združenih državah lepo bogatijo. Tuje povpraševanje po ameriških informacijskih strojih presegajo edino zahteve po ameriških puškah in maslu. Kakor hitro so te želje potešene, se vsa pozornost preusmeri k računalnikom in komunikacijam.

Izvoz računalnikov je v letu 1975 presegel vsoto 2,2 milijardi $. Za telekomunikacijske sisteme so izdelovalci v Združenih državah iztržili čedno vsoto 1,1 milijarde $. Po plačilni bilanci so te informacijske tehnologije dale ogromen presežek in sicer v višini 2,4 milijardi $. Edini trgovski problem, ki se pojavlja na obzorju, je "poplava" naprav potrošniške elektronike japonskih izvoznikov na ameriškem tržišču. V letu 1975 so Združene države uvozile radijskih aparatov, televizorjev, hi-fi aparatur in kalkulatorjev v vrednosti 2,067 milijardi $, izvozile pa v vrednosti 625 milijonov $. Toda na področju kalkulatorjev, perifernih enot, mikrovalovne tehnike, stikalne opreme, terminalske opreme, zemeljskih postaj za satelitski prenos, so Združene države v razcvetu. In ta široki pojav bo še dolgo odmeval.

Države v razvoju, ki se spoprijemajo z industrijsko tehnologijo, so v čudnem položaju, ko preskakujejo celo fazo ekonomske evolucije. Država kot n.pr. Brazilija (ali Iran), hkrati razvija industrijsko in informacijsko infrastrukturo. Brez smisla je namreč uvažati moderno industrijsko tehnologijo brez pritiklin: nadzor s procesnimi računalniki, računalniško vodenje tehnike in hitro komuniciranje med vsemi sektorji (podjetja, poslovni sektorji, vladne ustanove, gospodinjstva). Bližč upravljanja s tehniko v moderni industrijski dobi je informacijski in zahteve držav v razvoju po informacijskih dobrinah iz Združenih držav bodo zato še nadalje stopnjevane.

Modernizacija in transfer tehnologije

V okviru tega, da računalniki in komunikacije služijo modernizaciji v gospodarstvu (t.j. podpiranje industrijskega razvoja in proizvodnje), so se Združene države obvezale, da bodo spodbujale takšen razvoj. Gre za vprašanje življenjske ravni, ki je tesno povezano z opredelitvijo ameriške politike glede pomoči tujini - pomagati državam v razvoju pri dvigu njihovega življenjskega standarda.

Vendar modernizacija ali razvoj v tem pomenu nista "nevtralni" besedi. Pogosto ju izenačujejo z izrazom "westernizacija", ki predstavlja večno temo dialogov med severom in jugom. V zvezi z razvojem, je informacijska tehnologija tako politično dejstvo kot gospodarski vir. Vzpostavitev administrativnih postopkov, računovodskih in finančnih služb, kontrole in upravljanja po ameriškem vzorcu - vse to ni čisto nepolitično. V zvezi z računalniki sicer ni ničesar političnega razen dejstva, da je na nesrečo informacijske dobrine moč uporabljati enakovredno za gospodarski razvoj in za vojne namene in da so ameriški strokovni svetovalci del prenosa tehnologije. To pa utegne vplivati oziroma zahtevati politične odločitve.

Z varnostnimi ukrepi pri izvozu računalnikov, je Ministrstvo za zunanje zadeve Združenih držav to točko precej dramatiziralo. Serija 3030 IBM (The International Business Machines) in Cyber 170 CDC (Control Data Corporation) sta tehnološko razvitejša modela med ameriškimi računalniki in ju doma uporabljajo za gospodarske in vojaške namene. Isti računalnik lahko služi "modernizaciji" - načrtuje projekt vodnega namakanja, vodi program nacionalnega cepljenja, koordinira enakomerno medindustrijsko rast. Prav tako pa lahko vodi logistično oskrbovanje armade, pomaga pri načrtovanju nuklearnega orožja, služi kot precizen kontrolni sistem za vodenje izstrelka ali zbira tuje obveščevalne podatke. Na priporočilo Ministrstva za zunanje zadeve je Ministrstvo za trgovino pred kratkim odklonilo izvoz zmogljivejših računalnikov Sovjetski zvezi zaradi njihove uporabnosti pri vojaških strateških nalogah. Poleg tega je Carterjeva administracija omejila prodajo računalnikov nekaterim državam in znova pregleduje celotno politiko v zvezi z vprašanjem človekovih pravic. Takšna politična razmišljanja pa po drugi strani jemljejo domačim izdelovalcem priložnost, da sprostijo izvozni pritisk doma in ne nazadnje odvzemajo gospodarstvu vir za zaposlovanje, dobičke in tujo valuto.

Kratko rečeno torej, računalnik služi gospodarskemu razvoju, je pa tudi potencialno strateško orožje in prav tega temeljnega dualizma njegove uporabe/namena ni lahko rešiti.

Informacijski sistemi in človekove pravice

Uporaba informacijskih sistemov, ki vključuje telekomunikacijsko in računalniško opremo, je sprožila tudi zunanjepolitične probleme na področju človekovih pravic. Nihče ne zanika temeljnih potreb neke dežele po komuniciranju, tako znotraj nje same, kot s sosednjimi deželami. Moderna gospodarstva, še toliko manj zahodno orientirana industrijska gospodarstva, ne morejo več preživeti brez učinkovitih telekomunikacijskih sistemov. Prav zaradi tega je bil pred kratkim odobren izvoz ogromnih količin izpopolnjenih komunikacijskih in računalniških sistemov, največ za dežele tretjega in četrtega sveta.

Vendar se pojavlja vprašanje, kako je z uporabo telekomunikacijskih sistemov v notranjem nadzoru - množična prisluškovanja, snemanja pogovorov na magnetni trak - in sistematično kršenje prav tega, kar Združene države imenujejo osnovne človeške in državljanske svoboščine? Danes je čisto preprosto prisluškovati telefonskim pogovorom, če gre za politiko notranjega nadzora, ki uporablja visoko avtomatizirano opremo. Zato utegne biti v tem primeru prodaja informacijskih sistemov pomembna politična odločitev. Ali imajo Združene države pravico diktirati svojim odjemalcem, kako smejo ali ne smejo uporabljati uvoženi sistem? Kako naj država vnaprej zagotovi, da bo informacijski sistem uporabljala za pospeševanje gospodarskega razvoja, ne pa z njim zlorabljala človekove pravice. Za Združene države predstavlja to spet problem.

Izvoz informacijskih storitev utegne biti prav tako neprijeten kot je izvoz informacijskih dobrin. Te storitve lahko razdelimo na tri osnovne tipe: (1) obravnavanje informacije v zvezi s finančnim, zavarovalnim in računovodskim poslovanjem ter uporaba podatkovnih baz. (2) Izvoz kulture s pomočjo medijev kot so film, televizijski programi, radio, knjige, časopisi in revije. (3) Izvoz znanja preko patentov, avtorskih pravic in obračunavanje storitev in uslug v zvezi z vodenjem in upravljanjem oziroma svetovalnega značaja. Vse te storitve po svoje prispevajo k nastajanju političnih problemov.

Zasebnost in baze podatkov

V Združenih državah in v Evropi proizvajajo podjetja gore in gore finančnih podatkov, ki se prosto pretakajo preko mednarodnih meja. Takšno povezovanje trgovine je tako naravno, kot je bil pretok dobrin v trgovinski in industrijski dobi. Nov vir vrednosti so informacije (fakture, tovorni listi, prevozi tovora, zasebna potovanja, finančne pogodbe, privatni bančni računi, personalni podatki). Že samo evropsko tržišče podatkov je projektirano tako, da bo z letom 1980 preseglo 5 milijard $.

V zvezi z varovanjem zasebnosti pa so se pojavili novi neprijetni politični problemi. Zakoni, ki ščitijo osebne informacije, se precej razlikujejo od države do države in še do sedaj niso usklajeni. Le dve državi, Švedska in ZRN, sta sprejeli stroge zakone, ki ščitijo zasebnost. Evropska gospodarska skupnost in OECD (Organisation for Economic Cooperation and Development) sta zaskrbljeni, ker lahko sedaj neka država s šibkimi zakoni o zasebnosti služi kot "zatočišče podatkov" nekemu nepoštenemu podjetju tako, da ustanovi svojo podružnico v takšni manj strogi državi - gostiteljici, s tem pa izpodkopava stroge zakone o zasebnosti svoje lastne države. Dejstvo je, da veljajo Združene države za posebno privlačno "zatočišče podatkov".

Rešitev tega vprašanja naj bi bila konvencija ali pogodba o "pretoku podatkov preko državnih meja", o kateri trenutno razmišlja OECD. Nekatera ameriška podjetja pa trdijo v zvezi s tem, da je resnični namen dogovora le evropski poskus, izločiti ameriška podjetja iz tekmovanja na donosnih tržiščih podatkov. Takšno razlago pa evropske države odločno zavračajo in zatrjujejo, da jih resnično skrbi le varovanje zasebnosti. Debata se še nadaljuje.

Izvoz kulture ali "kulturni imperializem"

Prodaja izdelkov s področja kulture (televizijski programi, filmi, knjige, revije) predstavlja naraščujoči vir dohodka za te ameriške medije. Pozitivni učinki tega kulturnega izvoza so poleg profita še ti, da preko njega Združene države predstavljajo svetovni skupnosti svoje ideje, mišljenja in poglede. Obenem pa se razvija vse bolj vroča debata o politiki takšnega izvoza. Posebno dežele tretjega sveta, ki so največkrat brez razvite lastne industrije masovnih informacijskih medijev, si skušajo te kulturne izdelke preskrbeti, hkrati pa jih odklanjajo. Vendar se izkaže, da je vedno ceneje kupiti ameriško razvedrilo, kakor narediti televizijski program ali film doma. Minister za kulturo in izobraževanje, ki se srečuje na eni strani z omejenim proračunom, na drugi strani pa z nemogočimi zahtevami, da zapolni vsaj osem ur televizijskega programa dnevno, ima gotovo vse razloge za to, da raje uvaža kot proizvaja doma. To pa pomeni - uvoz izdelkov iz Združenih držav.

Znane so tudi trditve, da so tovrstna sredstva elementi "kulturne propagande", da so v bistvu sila "kulturnega imperializma", da se s prikazom ameriškega načina življenja v televizijskih programih prenaša "ideološka hrana" skupaj z razvedrilom.

Zagovorniki izraza "kulturni imperializem" opozarjajo na vse dominantnejši in močnejši položaj ameriških masovnih informacijskih medijev na svetovnem tržišču medijev. Res je, da levji delež vseh televizijskih in filmskih proizvodov prihaja iz Združenih držav. Zagovorniki "izvoza kulture" pa izjavljajo, da druge države lahko ta uvoz enostransko omejijo s tem, da odklonijo nakup, če ga imajo za politično nesprejemljivega ali propagandnega. Končno je kupovanje TV programov prosta trgovska transakcija. Tukaj pa spet pride do paradoksa: nove tehnologije direktnega oddajanja preko komunikacijskih satelitov utegnejo obiti "cenzuro", ki odklanja uvoz ameriških kulturnih izdelkov. Nekatere države pa se bojijo vdora preko neavtoriziranih oddaj, oddajanja vseh informacij brez razločka ali njihovega nezakonitega sprejema.

Ti problemi so predmet nenehnega razpravljanja na konferencah pod pokroviteljstvom Unesca. Nekatere države v razvoju, ki so v sprejemnem območju komunikacijskih satelitov, so se združile z državami Vzhodnega bloka v protestu proti možni "invaziji" preko telekomunikacijskih satelitov zahodnega izvora. Snop oddajanja telekomunikacijskega satelita je moč brez težav oblikovati tako, da pokriva prav majhno ali pa veliko področje. Prav tako si je težko predstavljati, da bi družina kje v tretjem svetu ali v kateri od komunističnih držav znala postaviti parabolično anteno na streho in na skrivaj sprejemala satelitske prenose. To vprašanje je popolnoma političnega značaja in rešitve še zlahka ne bo.

Izvoz znanja, kot vir dohodkov sekundarnega informacijskega sektorja, se skriva v prodaji patentov, zaračunavanju upravljalskih in svetovalskih storitev in v prodaji licenc. Znanje se prodaja kot "know-how" (znati kako) in "show-how" (pokazati kako), potem kot organizacijske izkušnje, znanstvene in tehnične informacije in znanje v upravljanju in vodenju. Kakor se je že pokazalo - primer televizijskega programa - ima tudi minister za znanost in industrijo prav malo razlogov, da bi investiral v domačo "proizvodnjo pameti". Mnoge države v razvoju se srečujejo s kritičnim pomanjkanjem "tehnokratov" (znanstvenikov, inženirjev, menežerjev) in gotovo nimajo kaj več kot strokovnjake z omejenimi izkušnjami pri vodenju tehničnih in finančnih poslov. Toda trgovanje v najširšem pomenu, tako v vlogi uvoznika kot izvoznika, zahteva resnično dognano (prefinjeno) znanje. Nakup takšnega znanja je vedno cenejši kot poskus pridobiti ga doma. Prav zato so Združene države sprejele program pomoči tujini, posebno na področju znanstvenih in tehničnih informacij. Mnogo odločilnih informacij je moč zelo poceni kupiti. Tudi organizaciji kot Svetovna banka in Mednarodni denarni sklad, ponujata določene informacije skupaj z nizko obrestnimi posojili. Mnoga ameriška podjetja se ukvarjajo s "prodajanjem" informacij in sklepanjem pogodb za upravljanje in svetovanje. In končno, kadar neka multinacionalna delniška družba postavi podružnico kot skupno naložbo v eni od držav v razvoju, predstavljajo pogosto del kupčije tudi patentne pravice in plačila za uporabo ameriške tehnologije in izkušenj.

Koncept "tehnološkega imperializma" pa izhaja iz te zadnje značilnosti. Večina držav v razvoju preprosto nima "človeškega kapitala", ki bi obvladoval razvitejšo tehnologijo. Tako je pritok ameriških usposobljenih tehnikov, menežerjev in znanstvenikov v te dežele pogost in tako zelo očiten, da postaja tarča ostrih kritičnih političnih napadov. Kritiki dokazujejo, da država z nakupom tehničnega ali znanstvenega znanja (v obliki avtorskih pravic, patentov ali plačil storitev) hkrati uvozi tudi določeno obliko gospodarske ureditve, ki karakterizira Zahod. To je skoraj neizogibno. Hierarhija upravne organiziranosti, ekonomski koncept produktivnosti, sistem cen, načini financiranja, spretnosti trgovanja, obvladovanje povpraševanja - to so pojmi zahodne tehnologije. Brž, ko se industrija postavi na noge na podlagi zahodnega znanja, pa to postane dejstvo kulture. Ko se temu prilagodijo še druge, bolj tradicionalne industrijske panoge, je prav verjetno, da se bo slej ko prej spremenila tudi oblika in fizionomija ekonomske in socialne organiziranosti države v razvoju.

Ta problem je manj opazen v trgovanju med Združenimi državami in državami OECD. Tukaj je problem le v neposredni zvezi s ceno. In večini držav se zdi uvoz dobrin ameriškega sekundarnega informacijskega sektorja dobra kupčija.

Ameriški izvoz filmov in televizijskih programov (serij) je v letu 1973 znesel 324 milijonov $. Za primerjavo - izvoz znanja (patenti, plačila za storitve) je prinesel 3.034 milijonov $ - skoraj 10-krat več kot najemnine za filme, avtorske pravice itd. Tiste, ki povzdigujejo glas ogorčenja zaradi invazijskih učinkov tujih TV in filmskih izdelkov, bi morali opozoriti, da je njihova zaskrbljenost usmerjena v napačno smer. Tehnološko znanje daje desetkrat več državnemu dohodku kot izdelki kulture in je mnogo bolj pomemben "skriti" izvoz kulture. In vendar si države v razvoju, včasih celo iste, ki obtožujejo Združene države kulturne invazije preko televizije in filma, vneto prizadevajo dobiti znanstvene in tehnične informacije.

Pogled v prihodnost

Pojav informacijske družbe v Združenih državah - pa tudi drugod - pomeni, da bo imela proizvodnja in distribuiranje znanja ključno vlogo v bodoči ekonomski rasti.

Informacijske družbe bi utegnile postati živ vir novega znanja, inovacij in napredka. Koristi le-teh bi lahko prerasle lastne okvire.

Ena najprepričljivejših in najprivlačnejših domnev o prihodnji ekonomski rasti je, da informacijsko gospodarstvo ni tako vezano na naravna bogastva, kot je to primer z industrijskim gospodarstvom. Vsak dan nas znova opozarjajo, da svetu zmanjkuje virov, ki jih ni moč obnoviti. In spoznali smo že kakšno škodo utegne pomanjkanje teh virov povzročiti v gospodarstvu, sprožiti gospodarske recesije, nezaposlenost in inflacijske pritiske. Združenim državam se odpirajo oči ob dejstvu, da imajo naravni viri in okolje fizične meje in se ne morejo upirati ponavljajoči zlorabi in zanemarjanju.

V takšnem razpoloženju je dobro ponoviti, da je informacija najbolj nenavadna od vseh virov. Neskončno se lahko obnavlja, uporaba oziroma izkoriščanje je ne uniči in uporablja jo lahko nenehno in sočasno več ljudi. Informacija z rabo nič ne izgubi na vrednosti. Prav nasprotno, več je v rabi določena vrsta informacij (znanje, zakoni), dragocenejša postaja. Informacijske dobrine in storitve ne zahtevajo ogromnih vlaganj naravnih bogastev ali energije, prav tako pa ne povzročajo polucije okolja.

Zaupamo lahko v svetlo prihodnost, ki jo utegnejo imeti Združene države kot informacijska družba, seveda s pametnim vodenjem. Imajo dobro proizvodno in človeško infrastrukturo in lahko jo mobilizirajo v veliko silo.

Toda še mnogo notranjih vprašanj, socialnih in gospodarskih je potrebno rešiti, npr.: protislovje med zasebnostjo in prosto rabo informacij; svoboda javnega govora in omejitve v izjavah v zvezi s komercialnimi posli; obremenitve zakonov in predpisov zvezne administracije; nov Akt o avtorskih pravicah, ki daje večjo zaščito avtorjem z omejitvijo proste uporabe njihovih del - vse to so elementi informacijske politike.

Tudi notranja politika se bo morala osredotočiti na strukturo informacijskih industrij, na takšna vprašanja kot so tekma s predpisi v telefonski industriji; meje med komunikacijami in računalništvom; prihodnost sistemov za elektronski prenos monetarnih fondov in elektronske pošte - to so prav tako elementi informacijske politike. In pozornost bo potrebno usmeriti na internacionalne implikacije - izvoz informacijskih dobrin in storitev, izvoz kulture in prenos tehnološke in znanstvene informacije.

Zaenkrat je nemogoče reči kaj določenega o informacijski družbi, kajti to obdobje se še razvija. Toda načrtovalci lahko opazujejo njeno rast in dajejo pametne napotke za njen bodoči potek. Napovedi za informacijsko družbo so različne, rešitve nedokončne in ne bi smeli spregledati mednarodnih implikacij.

Pripravila: A.Jerman-Blažič in
M.Kostevc

U V O D V CP/M* II

ANTON P. ŽELEZNIKAR

SOZD ELEKTROTEHNA, DO DELTA

UDK: 681.3.06 CP/M: 181.4

Članek je drugo nadaljevanje uvoda v CP/M operacijski sistem in opisuje v tem prispevku tri prehodne ukaze, in sicer PIP, ED in DDT. Opis PIP ukaza se nadaljuje iz prvega dela, v celoti pa sta prikazana ED in DDT ukaz. Ukazi PIP, ED in DDT so kompleksni prehodni programi in članek opisuje praktično vse njihove podukaze oziroma podukazne skupine. Ukazi urejevalnika teksta (ED) se obravnavajo v štirih značilnih skupinah, in sicer za prenos teksta, za delo v ED pomnilniškem vmesniku, za iskanje in spreminjanje teksta ter sestavljeni ukazi. DDT je program za oblikovanje in popravljanje različnih programskih paketov, za zasledovanje programskega izvajanja, za prekinjanje uporabniških programov in za prikazovanje pomnilniške vsebine v različnih formatih, od strojnega do zbirniškega.

Introduction to CP/M* II. This article continues the introduction to the CP/M Operating System and describes three further transient commands, namely PIP, ED, and DDT. The description of PIP command is continued from the first part of the article but ED and DDT command are presented completely. PIP, ED, and DDT command are complex transient programs and this article describes all their subcommands and subcommand groups respectively. The commands of the context editor (ED) are presemted in four significant groups, that is for transferring text, working in the edit buffer, searching and changing text, and combining commands. DDT is a program for creating and debugging different program modules, for tracing of user program execution and their interrupting, and for displaying of memory in several formats from machine to assembly code.

4.6. PIP kopiranje med napravami

PIP lahko kopira podatke iz zbirke na napravo, iz naprave v zbirko ter iz naprave na napravo. Imamo tele osnovne PIP ukaze za kopiranje v povezavi s perifernimi napravami:

```
PIP d:ime_zbirke.tip=nap: p ´cr´
        Podatki se kopirajo iz naprave nap: v

        zbirko ime zbirke.tip na disku ´d´
PIP nap:=d:ime_zbirke.tip p ´cr´
        Podatki se kopirajo iz ime_zbirke.tip
        na disku ´d´ v napravo nap:

PIP pri:=odh: p ´cr´
        Podatki se kopirajo v napravo pri:
        (prihodna) iz naprave odh: (odhodna)
```

Parametri (p) so enaki onim za kopiranje zbirk in smo jih že opisali. V primeru ko zunanja naprava ne odda znaka konca zbirke, lahko uporabimo Q parameter.

PIP ukaze lahko obogatimo z imeni "posebnih" naprav:

```
NUL:    To je odhodna "naprava", ki pošlje 40
        ASCII ničel (nevtralni znaki 00H) v na-
        vedeno prihodno napravo

EOF:    To je odhodna "naprava", ki pošlje znak
        konec_zbirke (´CTL z´ ali 1AH) v nave-
        deno prihodno napravo

INP:    Je uporabniško oblikovana priložnostna
        odhodna naprava in PIP moramo z njeno
        vključitvijo modificirati

PRN:    To je posebna oblika LST: naprave, ki
        razširi tabulirne znake, oštevilči vr-
        stice in strani kopije (podobno kot
        LST: NPT8 )
```

Te (umetne) naprave se uporabljajo kot prihodne in odhodne naprave.

4.7. Primeri uporabe PIP ukazov

Uporaba PIP ukazov je zelo raznovrstna, tako da ne bo mogoče pregledati vseh možnosti. Opisali pa bomo najbolj značilne in najpogosteje uporabljane oblike PIP ukazov.

PIP´cr´ Ta ukaz naloži PIP v pomnilnik. Imamo:

```
A > PIP´cr´
*

A > PIP´cr´
*pip_ukazna_vrstica´cr´
*

A > PIP´cr´
* ´cr´
A >
```

---

* CP/M je zaščitni znak ameriškega podjetja Digital Research, Pacific Grove, Ca 93950.

```
A>PIP´cr´
*´CTL c´
A>
```

Tu je ´*´ znak pripravljenosti programa PIP, da sprejme novo ukazno vrstico, ki je v gornjem primeru pip_ukazna_vrstica. Znaka ´cr´ in ´CTL c´ vrneta upravljanje iz PIP programa v CP/M sistem in znak ´ > ´ označuje pripravljenost CP/M sistema za sprejem novega CP/M ukaza.

Naslednji primer z vrsto možnih variacij bodi

```
PIP d:novo.tip=e:staro.top[p]´cr´
```

Ta ukaz kopira staro.top iz diska e: v novo.tip na disku d: z uporabo parametra ´p´. Imamo tele posebne primere:

```
B>PIP zbirka1.doc=zbirka2.doc´cr´
B>
```

Tu se kopiranje opravi na trenutno aktivnem disku, ko se obstoječa zbirka2.doc podvoji (kopira) še z zbirka1.doc.

```
C>D:PIP A:ike2.com=B:ike1.com[V]´cr´
C>
```

Ta primer je že nekoliko bolj zapleten. Program PIP, s katerim bomo kopirali, se nahaja na disku D: in ukazna vrstica kaže, kako ga pokličemo iz stanja z diskom C:. Potem se začne kopiranje z verifikacijo (parameter ´V´), in sicer zbirke ike1.com na disku B: v zbirko ike2.com na disku A: in končno se kontrola vrne zopet na izhodiščni disk C:. V tem primeru so v postopku kopiranja sodelovale štiri različne diskovne enote v zaporedju C:, D:, A:, B:, A: in C:. Prva pojavitev A: v zaporedju je bila potrebna za odprtje zbirke ike2.com na disku A:.

Določene ukaze lahko pišemo tudi okrajšano, ko imamo npr..

```
A>PIP B:=A:dokument.zim[V]´cr´
```

in se zbirka dokument.zim kopira iz diska A: na disk B: z enakim imenom in z verifikacijo. Seveda pa smemo za enak učinek pisati tudi

```
A>PIP B:dokument.zim=A:[V]´cr´
```

kjer smo zbirko navedli samo na levi strani enačaja. Še večji učinek kopiranja bi lahko dosegli z ukazom

```
A>PIP B:=A:*.alg[v]´cr´
COPYING-
JAN.ALG
FEB.ALG
MAR.ALG
A>
```

ko smo kopirali vse zbirke s pripono ´alg´ iz diska A: na disk B: z verifikacijo. Pri tem poudarimo, da smemo uporabljati pri vtipkavanju s tastaturo tako velike kot male črke. CP/M sistem interpretira ukazne vrstice, kot da so zapisane z velikimi črkami. Seveda lahko še z nekoliko splošnejšim ukazom kopiramo na disk vse zbirke drugega diska, ko uporabimo npr. ukazno vrstico

```
C>PIP D:=B:*.*[V]´cr´
```

Ta ukaz se uporablja dokaj pogostoma, npr. ko izločamo iz uporabe nezanesljive ali izrabljene upogljive diske ali ko kopiramo na uporabniške diske določene programske pakete iz arhivskih diskov.

Nadalje lahko s PIP ukazom oblikujemo novo, združeno zbirko iz obstoječih zbirk, ko imamo splošno

```
PIP d:novo.zbi=e:staro1.zba[p],f:staro2.zbe[p]
```

Nova zbirka na disku d: je enaka stiku navedenih zbirk na diskih e: in f:, pri čemer se vsaka od starih zbirk prenese pri določenem parametru (parametri so seveda lahko medsebojno različni).

S PIP ukazi lahko zbirke tudi listamo z različnimi napravami za listanje (tiskanje). V odvisnosti od prireditve fizične naprave logičnemu kanalu LST: bomo dobili izpis na ustrezni fizični napravi. Imeli bomo npr. tole:

```
B>A:PIP LST:=D:CPMINF6.TXT´cr´
```

ko bomo z diska B: poklicali program PIP na disku A: in ta bo izlistal preko kanala LST: tekstovno zbirko CPMINF6.TXT z diska D:. Hkrati lahko izlistamo tudi več zbirk, če uporabimo PIP ukaz tipa

```
B>PIP LST:=*.alg´cr´
```

ko se izlistajo vse zbirke s pripono ALG diska B:.

K pravkar opisanem PIP ukazu imamo tudi obraten ukaz, ki pobere znake iz kanala v zbirko na disku. Zbirko zapremo, ko se iz kanala (npr. iz tastature) pojavi znak ´CTL z´. Imejmo tale primer:

```
C>PIP B:DOK.MAJ=CON:´cr´
```

V zbirko DOK.MAJ na disku B: se prenašajo znaki s konzole CON: (na ta kanal je seveda lahko povezan tudi bralnik traku ali tastatura itn.). Znak ´CTL s´ ustavlja program PIP, tako da lahko na zaslonu opazujemo določene učinke ali čakamo zaradi drugih zunanjih vzrokov.

Nazadnje si oglejmo še kopiranje iz vhodne naprave v izhodno napravo. Tu se kopirajo znaki iz odhodne v prihodna napravo in imamo npr.

```
A>PIP PTP:=PTR:[EU]´cr´
TO JE PAPIRNI TRAK
A>
```

Tu se kopira vhod iz bralnika papirnega traku PTR: v luknjalnik papirnega traku PTP:, pri čemer se male črke pretvorijo v velike (parameter U) in se prikažejo (na konzoli) kopirani podatki (parameter E). Druga vrstica kaže primer takega prenosa na konzolo.

4.8. Urejevalnik teksta ED

V standarden CP/M paket sodi tudi urejevalnik ED kot prehoden ukaz. Ta urejevalnik ima vrsto vgrajenih ukazov za oblikovanje, spreminjanje in shranjevanje teksta. ED ni posebno močen urejevalnik, vendar ga je mogoče z nekaj vaje in potrpežljivosti uporabljati za vsakršnje urejevanje tekstov. ED ni zaslonsko usmerjen: učinki njegovih ukazov so vidni le post festum. Deluje torej tako, kot večina standardnih urejevalnikov na srednjih in velikih računalniških sistemih.

ED urejevalnik aktiviramo z njegovim pozivom in z obvezno navedbo obstoječe ali nove zbirke (tiste, ki jo želimo začeti oblikovati). Najbolj splošna pozivna oblika bi bila

```
A>B:ED C:ime_zbirke.tip´cr´
```

S tem ukazom pokličemo z diska A: urejevalnik ED na disku B: za urejevanje zbirke ime_zbirke.tip na disku C:. Poziv ED ima za posledico tudi odprtje začasne zbirke ime_zbirke.$$$ na istem disku, kot je zbirka ime_zbirke.tip. Po izstopu iz urejevalnika bo ta začasna zbirka postala nazadnje urejevana zbirka tipa .tip, dočim se bo prvotna (ob vstopu) zbirka preimenovala v tip .BAK. Program ED je shranjen na disku kot zbirka ED.COM, po pozivu pa se prenese v hitri pomnilnik in se locira pod sam operacijski sistem (v vrhnjem delu hitrega pomnilnika).

ED urejevalnik je program, ki sprejema znake s tastature v hitri pomnilnik, od tam pa vpisuje tekst na disk. Ker se tekst spreminja zaradi napak in drugih želenih modifikacij, mora imeti urejevalnik vrsto vgrajenih ukazov za prikazovanje, spreminjanje, brisanje in dodajanje teksta. ED urejevalnik ima dve bistveni lastnosti: je znakovno in vrstično usmerjen, kar pomeni, da delujejo njegovi ukazi nad znaki in vrsticami. Pri tem je vrstica niz znakov, ki končuje z znakom pomika valja.

Kot smo že zapisali, bomo imeli po nekem urejevalnem postopku z ED ukazom na disku vselej dve zbirki, in sicer z neko originalno pripono (.tip) in s pripono .BAK za imenom zbirke. Pri večkratnem urejevanju (ponovnem klicanju programa ED) bosta zbirki .tip in .BAK zamenjevali svoja položaja na disku. Splošno bomo imeli

```
B>ED d:ime_zbirke.tip´cr´
*ed_ukaz´cr´
```

ED ukaze lahko razdelimo v štiri značilne skupine:

1. V to skupino sodijo ukazi za prenos teksta; prenašajo se vrstice ali tekstovni bloki:

   * iz prvotne zbirke v ED pomnilniški vmesnik
   * iz ED pomnilniškega vmesnika v začasno zbirko
   * iz prvotne v začasno zbirko
   * iz ED pomnilniškega vmesnika v drugo zbirko (ki ni prvotna ali začasna temveč je povsem druga zbirka na disku)
   * iz druge zbirke (ki ni prvotna ali začasna zbirka temveč neka druga zbirka) v ED pomnilniški vmesnik

2. Imeli bomo ED ukaze za delo v ED pomnilniškem vmesniku. Pri tem si bomo pomagali z imaginarnim kazalcem, ki kaže na neko lokacijo teksta v ED pomnilniškem vmesniku. Ti ukazi

   * vplivajo na tekst v ED pomnilniškem vmesniku upoštevaje tekstovni kazalec
   * vplivajo na gibanje kazalca

3. ED premore ukaze za iskanje in spreminjanje teksta v okviru ED pomnilniškega vmesnika, in sicer za

   * iskanje posebnega znakovnega niza
   * substitucijo (zámeno) znakovnega niza
   * postavitev preklopnika med znakovnimi nizi
   * pomik teksta iz prvotne zbirke skozi ED pomnilniški vmesnik avtomatično v začasno zbirko med iskanjem

4. Obstajajo sestavljeni ukazi. V ukazno vrstico lahko zapišemo niz ED ukazov in jo zaključimo s ´cr´. Tako dobimo široko izbiro urejevalnih funkcij in takšen ukazni niz lahko tudi poljubno mnogokrat ponovimo.

Opišimo sedaj nekoliko podrobneje posamezne ukazne skupine oziroma konkretne ukaze.

4.8.1. ED ukazi za prenos teksta

Ukazi, ki jih bomo obravnavali v okviru te skupine so: nA, nW, E, H, O, Q, R´cr´, Rime_zbirke´cr´ in nX. Oglejmo si jhi na kratko.

```
********
*  nA  *
********
```

Ta ukaz pridruži (append), pripne ali kopira ´n´ vrstic iz prvotne zbirke v ED pomnilniški vmesnik. Vsak A ukaz velja sam zase (ne glede na prejšnje A ukaze) in štetje se nadaljuje pri ponovnih A ukazih. Za pridružitev ene same vrstice uporabimo kar izraz A (1A ali 1a ni potrebno). Če želimo pridružiti v ED pomnilniški vmesnik vse vrstice iz prvotne zbirke, uporabimo ukaz 0A. Ta ukaz bo pomaknil iz prvotne zbirke vse ukaze, če se pri tem seveda ne napolni polovica ED pomnilniškega vmesnika oziroma če v prvotni zbirki ni več kot 65535 vrstic. Ukaz 0A je tako zelo uporaben. Vrstice, ko so že bile pridružene iz prvotne zbirke v ED pomnilniški vmesnik, se pri naslednjih A ukazih seveda ne upoštevajo za prenos.

```
********
*  nW  *
********
```

Ta ukaz je k prejšnjemu inverzen in z njim prepišemo ´n´ vrstic iz ED pomnilniškega vmesnika v začasno zbirko. Tu imamo še tele posebne oblike:

W	prepiši eno samo vrstico
W	prepiši maksimalmo število vrstic
0W	prepiši takšno število vrstic, da bo ED pomnilniški vmesnik vsaj polovično prazen

W ukaz se začne vselej s prvo vrstico v ED pomnilniškem vmesniku in vpisuje v začasno zbirko vrstice za preje vpisanimi vrsticami. S tem ukazom se v začasno zbirko prepisane vrstice zbrišejo iz ED pomnilniškega vmesnika, ostanek v tem vmesniku pa se pomakne na začetek.

```
*******
*  E  *
*******
```

Ta ukaz končá urejevalni postopek. Pri tem se prenese tekst in preimenujejo se zbirke, in sicer:

1. Tekst, ki je še ostal v ED pomnilniškem vmesniku se pomakne v začasno zbirko podobno kot pri uporabi W ukaza.
2. Tekst, ki je še ostal v prvotni zbirki (ni bil naložen v ED pomnilniški vmesnik) se pomakne v začasno zbirko.
3. Prvotna zbirka dobi pripono (tip) .BAK.
4. Pripona začasne zbirke (ta je .$$$) se spremeni v pripono prvotne zbirke.
5. Zbirka za prenos bloka (nastala z ukazom nX) z imenom X$$$$$$$$.LIB se zbriše z diska.
6. Na zaslonu se pojavi znak pripravljenosti CP/M sistema ( ).

E ukaz se naj uporablja za normalno končanje urejevalnega postopka.

```
*******
*  H  *
*******
```

S tem ukazom se pomaknemo na začetek urejevane zbirke. Pri tem se prenese tekst in se preimenujejo zbirke, in sicer:

1. Tekst, ki je ostal v ED pomnilniškem vmesniku se pomakne v začasno zbirko podobno kot pri uporabi W ukaza.
2. Tekst, ki je še ostal v prvotni zbirki, se pomakne v začasno zbirko.
3. Pripona prvotne zbirke se spremeni v .BAK.
4. Pripona začasne zbirke se spremeni v pripono prvotne zbirke.
5. Oblikuje se nova, prazna začasna zbirka.
6. Tako je vse pripravljeno za urejevanje nove prvotne zbirke.

H ukaz ima dve bistveni uporabi:

1. A, W in N ukazi lahko pomikajo tekst samo naprej ne pa nazaj, ko uporabljajo prvotne in začasne zbirke. Uporaba H ukaza reši urejevano vsebino ter omogoči vrnitev na začetek zbirke z namenom dodatnega urejevanja.
2. Priporočljivo je, da uporabljamo H ukaz med urejevanjem vsakih nekaj minut in tako rešujemo naše urejevalno delo na disk. Tekst v ED pomnilniškem vmesniku se namreč lahko zgubi zaradi napake uporabnika ali računalniškega sistema, dočim lahko tekst v začasni zbirki brez težav prepišemo nazaj. Pogostna uporaba H ukaza je torej še posebej priporočljiva, ko vstavljamo veliko količino novega teksta in opravljamo številne popravke.

```
*******
*  O  *
*******
```

Ta ukaz zbriše urejevano zbirko, tekst pa se prenese takole:

1. Zbriše se vsebina ED pomnilniškega vmesnika (v hitrem pomnilniku) in začasna zbirka.
2. Izvrši se vrnitev na začetek prvotne zbirke.

Pri O ukazu postavi ED vprašanje ´O-(Y/N)?´ pred izvršitvijo O ukaza-, saj se s tem ukazom izgubijo vse spremembe, ki so bile vnešene v zadnjem urejevalnem postopku. Pri odgovoru ´Y´ na vprašanje se vrnemo k prvotni zbirki, pri odgovoru ´N´ pa se urejevanje nadaljuje.

O ukaz se uporablja pri nepravilnih spremembah teksta, ko želimo urejevalni postopek začeti znova.

```
*******
*  Q  *
*******
```

Ta ukaz pomeni končanje urejevanja, ko se ne upoštevajo zadnje spremembe. Po Q ukazu se pojavi vprašanje ´Q-(Y/N)?´ in z odgovorom ´Y´ se vrnemo v CP/M sistem. Q ukaz uporabimo, če želimo začeti urejevanje znova ali če želimo v celoti preklicati zadnji urejevalni postopek.

```
***********
*  R´cr´  *
***********
```

Ta ukaz prebere (včita) zbirko, ki je nastala z uporabo X ukaza, in sicer v ED pomnilniški vmesnik.Z R´cr´ se vstavi vsebina vmesne zbirke X$$$$$$$$.LIB v ED pomnilnjiški vmesnik na mesto za tekstovnim kazalcem v ED pomnilniškem vmesniku. Vmesna zbirka se pri tem ne spremeni (ne zgubi) in jo lahko včitavamo poljubno mnogokrat med urejevalnim postopkom. Ta vmesna zbirka pa se avtomatično zbriše, ko zapustimo ED z E, Q ali ´CTL c´ ukazuom.

```
**********************
*  Rime_zbirke´cr´   *
**********************
```

Včita se knjižnična zbirka ime_zbirke.LIB v ED pomnilniški vmesnik. Ta ukaz vstavi vsebino navedene zbirke v ED pomnilniški vmesnik neposredno za tekstovnim kazalcem. Knjižnična zbirka ostane pri tem nespremenjena.

```
********
*  nX  *
********
```

Zapiši ´n´ vrstic iz ED pomnilniškega vmesnika (neposredno za tekstovnim kazalcem) v vmesno zbirko X$$$$$$$$.LIB. Kasneje je te vrstice možno prenesti v ED pomnilniški vmesnik z uporabo ukaza R´cr´. S kombinacijami X in R´cr´ ukazov lahko poljubno premikamo tekstovne bloke po ED pomnilniškem vmesniku. Zbirka X$$$$$$$$.LIB se avtomatično zbriše po končanem urejevalnem postopku (z uporabo E, Q ali ´CTL c´ ukaza). Z nX ukazom se iz ED pomnilniškega vmesnika ne zbriše ´n´ prepisanih vrstic, ostanejo torej tam, kjer so bile. Brisanje moramo opraviti z uporabo K ukaza.

4.8.2. Ukazi za delo v ED pomnilniškem vmesniku

K tem ukazom spadajo: +/-B, +/-nC, +/-nD, I´cr´, krmilni ASCII znaki, Iniz´CTL z´, Iniz´cr´, +/-nK, +/-nL, +/-nP, +/-nT, +/-U, +/-V, 0V, n:, :m in +/-n. Opišimo kratko njihov pomen.

```
**********
*  +/-B  *
**********
```

S tem ukazom pomaknemo tekstovni kazalec (TK) na začetek ali konec teksta v ED pomnilniškem vmesniku. Predznak + ali - določa začetek ali konec teksta in znak + smemo tudi izpustiti. Ta ukaz uporabljamo zlasti v kombinaciji z I ukazom za vstavljanje teksta na koncu obstoječega teksta in za vračanje na začetek teksta, ko tekst beremo (optično) in popravljamo. Predznak pri tem ukazu je nasproten od predznaka pri drugih ukazih (saj velja ´-´ za konec teksta).

```
***********
*  +/-nC  *
***********
```

Pomakni TK (tekstovni kazalec) za plus ali minus ´n´ znakov v vrstici. Pri tem se ´cr´ in ´lf´ štejeta kot dva znaka (saj se vstavita kot dva znaka na koncu vrstice). Pomik s C ukazom seže tudi preko vrstice in npr. +5C pomeni vobče pomik za 5 znakov proti koncu ED pomnilniškega vmesnika.

```
***********
*  +/-nD  *
***********
```

Ta ukaz zbriše ´n´ znakov neposredno pred (´-´) ali za (´+´) tekstovnim kazalcem (TK). V primeru

```
99: IMAMO PRELEPO JESEN
99: *-3D´cr´
```

ko je TK med črkama E in L v besedi PRELEPO, bi s tem ukazom dobili spremenjeno vrstico

99: IMAMO LEPO JESEN

Seveda pa bi lahko uporabili tudi S ukaz (glej kasneje).

****
* I´cr´ *
****

S tem ukazom nastopi stanje vstavljanja znakov s konzole. ED sprejema znake s tastature in jih vstavlja v ED pomnilniški vmesnik takoj za tekstovnim kazalcem (TK). Z nekaterimi izjemami se tako vsak vtipkani znak vstavi v ED pomnilniški vmesnik. Izjema je le znak ´CTL z´, ki zaključi stanje vstavljanja in na zaslonu se pojavi znak ED pripravljenosti (´*´) za sprejem novega ukaza.

****
* Krmilni ASCII znaki *
****

Krmilni ASCII znaki za delo v ED pomnilniškem vmesniku so tile:

****
* ´CTL h´ ali BACKSPACE *
****

Ta znak zbriše zadnji vtipkani znak in pomakne kurzor (TK) v levo.

****
* ´CTL l´ *
****

S tem znakom vstavimo dva znaka, in sicer ´cr´ in ´lf´ (tj. ´CTL m´ in ´CTL l´).

****
* ´CTL m´ ali RETURN *
****

Ta znak vstavi ´cr´ in ´lf´.

****
* ´CTL r´ *
****

Prikaže na zaslonu tekočo vrstico.

****
* ´CTL u´ ali ´CTL x´ *
****

Ta znak zbriše tekočo vrstico.

****
* RUBOUT ali DELETE *
****

S tem znakom se zbriše zadnji vtipkani znak, ki se prikaže potem še z odmevom.

Tekočo vrstico sestavljajo znaki, ki sledijo znaku ´cr´, vtipkani v stanju vstavljanja (I ukaz). Z I ukazom se oblikuje nova zbirka ali pa se dodajajo vrstice k obstoječi zbirki kjerkoli v tekstu. Kot smo že napisali, je priporočljivo pogostno prekinjanje stanja vstavljanja (znak ´CTL z´) in uporaba H ukaza.

****
* Iniz´CTL z´ *
****

S tem ukazom vstavimo ´niz´ znakov v ED pomnilniški vmesnik neposredno za tekstovni kazalec (TK). Ta ukaz se uporablja za vstavljanje krajših nizov.

****
* Iniz´cr´ *
****

S tem ukazom se vstavi vrstica ´niz´ (torej vključno s ´cr´ in ´lf´) neposredno za TK. Vstavitev ´cr´ in ´lf´ predstavlja edino razliko glede nz ukaz Iniz´CTL z´. Ta ukaz se uporablja za vstavitev ene same vrstice.

****
* +/-nK *
****

Ta ukaz zbriše ´n´ vrstic v ED pomnilniškem vmesniku od tekstovnega kazalca naprej ali nazaj. Te vrstice se ohranijo po izstopu iz urejevalnika v zbirki tipa .BAK. Znak minus je za brisanje pred TK, pri tem moramo vedeti, kje se nahaja TK (na začetku ali nekje v vrstici).

****
* +/-nL *
****

Ta ukaz pomekne TK (tekstovni kazalec: kurzor) naprej ali nazaj (+ ali -) za ´n´ vrstic v ED pomnilniškem vmesniku. Z ukazom 0L pomaknemo TK na začetek vrstice (če se je nahajal v notranjosti vrstice).

********** * +/-nP * **********

Tekstovni kazalec v ED pomnilniškem vmesniku se pomakne za ´n´ strani naprej ali nazaj in natisne se (na zaslonu) ena stran. P je pripraven ukaz za gibanje skozi tekst. Obseg strani je 23 vrstic. Ukaz 0P prikaže trenutno stran brez pomika TK.

****
* +/-nT *
****

T je tiskalni (prikazovalni) ukaz za ´n´ vrstic za ali pred TK. Če je TK v notranjosti vrstice, lahko vidimo tekst vrstice pred TK z ukazom 0T (od začetja vrstice do TK v vrstici). T ukaz ne pomakne TK, tako da ta ostane na svojem mestu.

****
* +/-U *
****

Ta ukaz rabi za prevod malih v velike črke (nekateri zbirniki zahtevajo velike črke). S +U se začne prevajanje znakov, z -U pa konča. Ta prevod se opravi v ED pomnilniškem vmesniku.

****
* +/-V *
****

Ta ukaz uvaja in ukinja oštevilčevanje vrstic. Po vtipkanju V ukaza se vsaka vrstica oštevilči z n:, kjer je ´n´ pripadajoča zaporedna številka vrstice. Oštevilčene vrstice pomagajo pri identifikaciji teksta in pri pomikanju TK.

```
********
* 0V   *
********
```

Posebna funkcija V ukaza je, da pove, koliko prostora v ED pomnilniškem vmesniku je še na razpolago in kako obsežen je ED pomnilniški vmesnik. Razlika med zadnjim in prvim podatkom je zasedenost ED pomnilniškega vmesnika. Npr.

```
*0V´cr´
20349/39863
```

kar pomeni, da je zasedenih 19424 lokacij ED pomnilniškega vmesnika.

```
********
* n:   *
********
```

Ukaz pomeni pomik tekstovnega kazalca (kurzorja) na vrstico s številko ´n´. Če so bile vrstice oštevilčene (uporaba ukaza V), se s tem ukazom lahko gibljemo po začetkih poljubnih vrstic.

```
********
* :m   *
********
```

Ukaz pomeni: začni pri tekstovnem kazalcu in nadaljuj vse do vrstice s številko ´m´. Ta izraz ni ukaz sam po sebi, temveč je predpona k nekemu ukazu. Vzemimo ukazno vrstico

```
73: *:122T´cr´
```

ED bo natisnil vrstice (T ukaz) od vrstice 73 (položaj TK) do vrstice 122. Ta ukaz lahko uporabimo tudi v povezavi z ukazom n: nad določenim območjem vrstic. Tako lahko npr. zbrišemo tekst med vrsticama 15 in 33 neglede na trenutni položaj TK, ko imamo

```
122: *15::33K´cr´
```

```
**********
* +/-n   *
**********
```

Pomaknemo se naprej ali nazaj za ´n´ vrstic in prikažemo ciljno vrstico. Ta ukaz je tako okrajšava ukaza +/-nLT, ko se kazalec pomakne napraj ali nazaj za ´n´ vrstic, nato pa se prikaže vrstica, pri kateri se je kazalec ustavil. Pomik valja je najpreprostejša oblika tega ukaza, ko imamo npr.

```
23: *´cr´
24: to je tekst v vrstici 24
24: *
```

Ta ukaz je enak ukazu LT, ko se pomaknemo na naslednjo vrstico in jo prikažemo.

4.8.3. Ukazi za iskanje in spreminjanje teksta

V tem podpoglavju si bomo ogledali ukaze nFniz´CTL z´, nNniz´CTL z´, nSiskann_niz´CTL z´zamenjalni_niz´CTL z´ in nJ... .

Ukaz za iskanje niza v danem tekstu ima obliko

```
******************
* nFniz´CTL z´   *
******************
```

Ta ukaz poišče v tekstu ´n-to´ pojavitev zaporedja znakov ´niz´, pri čemer se tekstovni kazalec (TK) ustavi neposredno za zadnjo, to je ´n-to´ pojavitvijo niza. Iskanje se začne pri trenutni poziciji TK, zato mora biti TK postavljen pred iskani niz. Če niza v tekstu ni, se bo TK ustavil na koncu ED pomnilniškega vmesnika, tj. na koncu teksta v tem vmesniku.

```
******************
* nNniz´CTL z´   *
******************
```

Ta ukaz pomeni, da se poišče ´niz´, tj. njegova ´n-ta´ pojavitev v ED pomnilniškem vmesniku in tudi na disku. Če namreč te pojavitve ni v ED pomnilniškem vmesniku, se avtomatično naloži nova vsebina z diska, tako da se najde zahtevani niz. Ta ukaz se izvaja nad celotno zbirko, in sicer od začetnega položaja TK. V tem je tudi glavna razlika med ukazoma tipa ´F´ in tipa ´N´.

```
*********************************************
* nSiskani_niz´CTL z´zamenjalni_niz´CTL z´  *
*********************************************
```

Ta ukaz narekuje, da se iskani niz v tekstu ED pomnilniškega vmesnika n-krat zamenja z zamenjalnim nizom. Zamena se opravi od začetnega položaja TK do konca teksta, seveda le pri prvih ´n´ pojavitvah iskanega niza, nakar se TK ustavi za zadnjim zamenjanim (zamenjalnim) nizom. Pri zameni je še posebej treba paziti na enoumnost iskanih nizov, ker lahko sicer dobimo nepredvidene rezultate.

```
*********************************************
nJisk_niz´CTL z´vstav_niz´CTL z´kon_niz´CTL z´
*********************************************
```

Ta ukaz deluje takole: najprej se poišče isk_niz. Ko je ta niz najden, se neposredno za njim vstavi vstav_niz, za njim pa se zbrišejo vsi znaki do niza kon_niz. Ta postopek se ponovi n-krat. Tekstovni kazalec ostane pri tem na koncu niza vstav_niz.

4.8.4. Sestavljeni ukazi

ED dopušča navedbo več ukazov v eni ukazni vrstici, tako da se s tem načinom skrajša čas vpisovanja ukazov s konzole. ED ukaze oziroma podukaze pišemo lahko enostavno enega za drugim, končni ukaz pa končamo z znakom ´cr´. Tako lahko primer

```
22: *0a´cr´
22: *b´cr´
 1: *t´cr´
 1:  To je vsebina prve vrstice.
 1: *
```

zapišemo krajše

```
22: *0abt´cr´
 1:  To je vsebina prve vrstice.
 1: *
```

Imamo le nekaj enostavnih pravil za tipkanje več ukazov v eno ukazno vrstico:

1. Ukazi E, H, O in Q lahko imajo pri tipografičnih (tipkalnih) napakah neprijetne posledice v kombinacijah z drugimi ukazi. Tu je potrebna določena pozornost, da ne pride do napak.

2. Pri uporabi ukazov, ki uporabljajo nize (npr. S ukaz), se namesto znaka ´cr´ (na koncu ukaza) vzame znak ´CTL z´.

ED ima tudi takoimenovane makroukaze oziroma dopušča njihovo definicijo. Ta format je tale:

```
**************************
*  nMukazlukaz2ukaz3´cr´  *
**************************
```

Kot vidimo, omogoča ta ukaz tudi njegovo n-kratno ponovitev. Imejmo tale primer:

```
*MFrom´CTL z´-3DIram´CTL z´0TT´cr´
```

Ta ukaz pomeni tole: poišči niz ´rom´ (From), ko je bil najden, ga zbriši (-3D), vstavi na njegovo mesto niz ´ram´ (Iram) in izpiši celotno vrstico (0TT). Ta sestavljeni ukaz se razlikuje od J ukaza. Če je v ukazu n:... vrednost n = 0, 1 ali izpuščena, se ta ukaz ponavlja do pojavitve napake.

4.9. DDT (Dinamični popravljalnik)

Program ali prehodni ukaz DDT se uporablja za preizkušanje in popravljanje programov, ki so napisani v strojnem jeziku. DDT ukazni vrstici sta le dve, ko imamo:

```
*************
*  DDT´cr´  *
*************
```

Ta ukaz naloži (z diska v hitri pomnilnik) program DDT in čaka na naslednji ukaz (DDT podukaz) s konzole.

```
*******************************
*  DDT d:ime_zbirke.tip´cr´   *
*******************************
```

Ta ukaz naloži najprej DDT v pomnilnik, nato pa naloži še zbirko z imenom ime_zbirke.tip z namenom, da bo ta zbirka preučevana, modificirana, preizkuševana, vendar mora biti tip te zbirke COM ali HEX.

Uporaba DDT prehodnega ukaza je večnamenska in z njim imamo na voljo tele funkcije oziroma opravila:

** Naložitev zbirniško prevedenega programa v pomnilnik
** Enostavna sprememba programa v strojnem jeziku
** Pomoč pri ugotavljanju napak v programu, zapisanem v strojnem jeziku
** Razvoj in vstavitev posebnega perifernega vmesnika, tj. programa za uporabo določene periferije
** Opazovanje in modificiranje vsebine v hitrem pomnilniku
** Vstavljanje kodą (strojnega jezika) v obliki vrstic, zapisanih v zbirnem jeziku
** Oblikovanje zbirniških vrstic iz strojnega koda (obratno zbiranje)
** Opazovanje in spreminjanje vsebin registrov centralnega mikroprocesorja
** Postavljanje prekinjevalnih točk v programih, ko se prekinja izvajanje uporabniških programov, da se tako lahko opazujejo učinki izvajanja programov
** Zasledovanje (koračno opazovanje) programskega izvajanja, opazovanje stanja pomnilnika in registrov

DDT ima 12 osnovnih ukazov, ki jih je moč uporabiti potem, ko je bil DDT naložen. Ti ukazi so:

Assss Vstavi stavke zbirnega jezika z začetkom pri naslovu ssssH
D Prikaži vsebino naslednjih 192 zlogov pomnilnika
Dssss Prikaži vsebino 192 zlogov, začenši od naslova ssssH
Dssss,ffff
Prikaži vsebino pomnilnika med naslovoma ssssH in ffffH
Fssss,ffff,cc
Napolni pomnilnik med naslovoma ssssH in ffffH z osembitno vsebino ccH
G Začni izvajanje programa pri naslovu, ki je vsebovan v programskem števniku
Gssss Začni izvajanje programa pri naslovu ssssH
Gssss,bbbb
Nastavi prekinitev pri naslovu bbbbH za program, ki se bo začel izvajati pri naslovu ssssH
G,bbbb Nastavi prekinitev pri naslovu bbbbH za program, ki se bo začel izvajati pri naslovu v programskem števniku
G,bbbb,cccc
Nastavi prekinitvi pri naslovih bbbbH in ccccH za program, ki se bo začel izvajati pri naslovu v programskem števniku
Iime_zbirke.tip
Nastavi zbirčni krmilni blok za zbirko ime_zbirke.tip, ki bo včitana z R ukazom
L Izlistaj naslednjih 11 vrstic programa v zbirnem jeziku, ko je bil ta program dobljen z obratnim zbirnikom
Lssss Izlistaj 11 vrstic programa v zbirnem jeziku od naslova ssssH dalje, ko je bil ta program dobljen z obratnim zbirnikom
Lssss,ffff
Izlistaj program v zbirnem jeziku med naslovoma ssssH in ffffH, ko je bil ta program dobljen z obratnim zbirnikom
Mssss,ffff,dddd
Pomakni vsebino pomnilniškega bloka med naslovoma ssssH in ffffH na naslov ddddH
R Včitaj zbirko z diska v pomnilnik (s predhodno uporabo I ukaza)
Rnnnn Včitaj zbirko z diska na naslov nnnnH v pomnilnik (s predhodno uporabo I ukaza)
Sssss Prikaži vsebino na naslovu ssssH in po želji spremeni vsebino pri tem naslovu
Tnnnn Zaslediuj izvajanje nnnnH ukazov uporabniškega programa
Unnnn Izvedi nnnnH ukazov uporabniškega programa, se ustavi in prikaži vsebine registrov mikroprocesorja
X Prikaži vsebine registrov mikroprocesorja
Xr Prikaži vsebino registra ali zastavice ´r´ mikroprocesorja

4.9.1. Uporaba nekaterih DDT ukazov

Na preprostih primerih si oglejmo uporabo DDT ukazov.

```
*************
*  DDT´cr´  *
*************
```

Naložimo DDT program (prehodni ukaz) v hitri pomnilnik:

```
A>
A>DDT
DDT VERS 2.2
-
```

V tem primeru se pojavi na zaslonu sporočilo o t. im. verziji programa DDT, v zadnji vrstici sporočila pa se pojavi še znak pripravljenosti DDT sistema za sprejetje novega ukaza (podukaza); ta znak je ´-´.

```
*******************************
*  DDT d:ime_zbirke.tip´cr´   *
*******************************
```

S tem ukazom se naložita DDT in imenovana zbirka v hitri pomnilnik:

```
A>DDT BISTAT.COM
DDT VERS 2.2
NEXT  PC
1580 0100
-
```

V tem primeru se razen verzije CP/M sistema pojavi še sporočilo, ki je sestavljeno iz podatka naslednje proste lokacije v pomnilniku (NEXT) in iz podatka o naslovu programskega števnika.

```
***************
*  Assss´cr´  *
***************
```

S tem ukazom vstavimo ukaz v zbirnem jeziku procesorja 8080A in vsak vstavljeni ukaz se sproti prevede v strojni kod tega procesorja, pri čemer se naslovi pišejo v levem stolpcu. V našem primeru vstavimo dva ukaza za naložitev registrov, pri čemer je dolžina teh ukazov enaka 2, kar se vidi tudi iz levega stolpca našega naslednjega primera:

```
-A100
0100  MOV A,C
0101  MOV B,D
0102
-
```

Kadar DDT ne razpozna nekega zbirnega ukaza (pri uporabi A ukaza), se izpiše vprašaj, kot kaže tale primer (naslov ostane pri tem nespremenjen):

```
-A100
0100  MVO A,C
?
0100
-
```

Ukaz A sprejema samo zbirne stavke procesorja 8080A, vendar pri tem ne moremo uporabljati označitev in jim prirejati vrednosti. Vsi zbirni operandi morajo biti naslovi ali konstante, ki so izraženi s heksadecimalnimi vrednostmi. Operator in operand morata biti ločena s presledkom, vsak zbirni stavek pa zaključen s pomikom valja (´cr´). Zbiranje z A ukazom se konča, če v prazno vrstico vtipkamo ´cr´.

```
************
*  D´cr´   *
************
```

Ta ukaz izpiše navadno 192 zlogov pomnilnika v 12 vrsticah s po 16 zlogi. Tako dobimo tale primer:

```
-D
0100 C3 33 04 20 20 20 43 6F 70 79 72 69 67 68 74 20 .3.   COPYRIGHT
0110 28 63 29 20 31 39 37 39 2C 20 44 69 67 69 74 61 (C) 1979, DIGITA
0120 6C 20 52 65 73 65 61 72 63 68 3F 3F 3F 3F 3F 3F L RESEARCH??????
0130 3F 3F 3F 3F 3F 00 00 43 4F 4E 3A 52 44 52 3A 56 :PUN:LST:DEV:VAL
0140 3A 50 55 4E 3A 4C 53 54 3A 44 45 56 3A 56 41 4C :USR:DSK:TTY:CRT
0150 3A 55 53 52 3A 44 53 4B 3A 54 54 59 3A 43 52 54 :BAT:UC1:TTY:PTR
0160 3A 42 41 54 3A 55 43 31 3A 54 54 59 3A 50 54 52 :URL:UR2:TTY:PTP
0170 3A 55 52 31 3A 55 52 32 3A 54 54 59 3A 50 54 50 :UP1:UP2:TTY:CRT
0180 3A 55 50 31 3A 55 50 32 3A 54 54 59 3A 43 52 54 :LPT:UL1:RAD:A/W
0190 3A 4C 50 54 3A 55 4C 31 3A 52 2F 4F 00 52 2F 57 -SYS-DIR-R/O R/W
01A0 00 53 59 53 00 44 49 52 00 52 2F 4F 20 52 2F 57  SYS DIR ** R/O R/W
01B0 20 53 59 53 20 44 49 52 20 2A 20 20 20 20 20 20  SYS DIR *
-
```

Tu je bil začetni naslov 100H. V levem stolpcu imamo vsakokratni naslov prvega zloga v tej vrstici, dodatno k vsaki vrstici naslovov in zlogov pa se pripišejo še vsi izpisljivi ASCII znaki in naš gornji primer to nazorno kaže. Znaki, ki niso izpisljivi kot ASCII znaki, se izpišejo s piko (´.´).

Naslednji primer kaže uporabo ukaza

```
***************
*  Dssss´cr´  *
***************
```

ko imamo začetni naslov 433H. Prva vrstica, ki ni popolna (nima 16 zlogov), se šteje kot popolna, tako da se naslednja začne z naslovom 440H, kot kaže naslednji primer:

```
-F5000,5100,55
-D5000,501F
5000 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUUUUUUUUUU
5010 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUUUUUUUUUU
-
```

Uporaba D ukaza pokaže (glej zgoraj), da so se lokacije med naslovoma 5000H in 501FH napolnile s 55H.

Vstavitev zbirke v hitri pomnilnik dosežemo z uporabo I in R ukaza, ko imamo splošno

```
*************************
* Iime_zbirke.tip´cr´   *
* Rnnnn´cr´             *
*************************
```

Pri tem lahko imamo R ukaz brez operanda, uporaba operanda pa pomeni pomik nnnnH, in sicer pri intervalu (0000H, 7FFFH) v pozitivno smer pomnilniških naslovov, pri intervalu (8000H, FFFFH) pa v negativno smer glede na nalagalni začetek zbirke ime_zbirke.tip.

Naložimo s tem parom ukazov zbirko STAT.COM, ko imamo:

```
-XP
P=144F 100
-ISTAT.COM
-R
NEXT  PC
1580 0100
-
```

Sporočilo kaže, da se je ta zbirka naložila v naslovnem intervalu (100H, 157FH), ko je naslednja prosta lokacija 1580H (NEXT). To pomeni, da ima zbirka STAT.COM nalagalni začetek 100H. Seveda pa lahko vplivamo na nalagalni začetek tako, da uporabimo R ukaz z operandom, kot kaže naslednji primer:

```
-D433
0433 21 00 00 39 22 32 15 21 54 15 F9 CD ED 1..9"2.!T....
0440 04 FE 20 D2 4F 04 01 11 04 CD D2 04 C3 8B 04 21 ... .O.........!
0450 5D 15 36 01 3A 5C 00 D6 00 D6 01 9F F5 3A 5D 00 ].6.:\.......:].
0460 D6 20 D6 01 9F C1 48 A1 1F D2 72 04 CD 33 0D C3 . ....H...R..3..
0470 8B 04 3A 5C 00 FE 00 CA 80 04 CD 02 14 C3 8B 04 ...:\...........
0480 CD 87 0A 2F 1F D2 8B 04 CD BA 0D 2A 32 15 F9 C9 .../.........*2...
0490 21 22 15 71 2A 22 15 26 00 EB 0E 02 CD 4C 14 C9 !".q*".&.....L..
04A0 0E 0D CD 90 04 0E 0A CD 90 04 C9 0E 20 CD 90 04 ............ ...
04B0 C9 21 24 15 76 2B 71 2A 23 15 7E FE 00 CA D1 04 .!$.v+q*#.~.....
04C0 2A 23 15 4E CD 90 04 2A 23 15 23 22 23 15 C3 B7 *#.N...*#.#"#...
04D0 04 C9 21 26 15 70 2B 71 CD A0 04 2A 25 15 44 4D ..!&.p+q...*%.DM
04E0 CD B1 04 C9 11 00 00 0E 0B CD 4F 14 C9 11 00 00 ..........O.....
-
```

V tem primeru v pomnilniku očitno nimamo ASCII znakov, temveč imamo programski kod, kar se nazorno vidi iz desnega dela v zadnjem primeru. Negrafični ASCII znaki se predstavijo v tem delu s pikami.

Uporabo ukaza

```
**********************
* Dssss,ffff´cr´     *
**********************
```

imamo v tem primeru:

```
-D4ED,507
04ED 11 00 00 ...
04F0 0E 0C CD 4F 14 C9 21 28 15 71 2A 28 15 26 00 EB ...O..!(.q*(.&..
0500 0E 0E CD 4C 14 C9 21 2A ...L..!*
-
```

Tu smo izpisali vsebine naslovov v intervalu (4EDH, 507H), pri čemer sta bili prva in zadnja vrstica nepopolni.

Večkrat se zlasti pri preizkušanju uporabniških programov izkaže, da potrebujemo ukaz

```
************************
* Fxxxx,yyyy,zz´cr´    *
************************
```

Uporaba tega ukaza pokaže, da se naslovi v intervalu (xxxx, yyyy) napolnijo z vsebinami ´zz´, kot je razvidno iz primera:

```
-ISTAT.COM
-RS000
NEXT  PC
6580 0100
-D5100
5100 C3 33 04 20 20 20 43 6F 70 79 72 69 67 68 74 20 -3-  COPYRIGHT
5110 28 63 29 20 31 39 37 39 2C 20 44 69 67 69 74 61 (C) 1979, DIGITA
5120 6C 20 52 65 73 65 61 72 63 68 3F 3F 3F 3F 3F 3F L RESEARCH??????
5130 3F 3F 3F 3F 3F 3F 00 00 43 4F 4E 3A 52 44 52 3A ??????..CON:RDR
5140 3A 50 55 4E 3A 4C 53 54 3A 44 45 56 3A 56 41 4C :PUN:LST:DEV:VAL
5150 3A 55 53 52 3A 44 53 4B 3A 54 54 59 3A 43 52 54 :USR:DSK:TTY:CRT
5160 3A 42 41 54 3A 55 43 31 3A 54 54 59 3A 50 54 52 :BAT:UC1:TTY:PTR
5170 3A 55 52 31 3A 55 52 32 3A 54 54 59 3A 50 54 50 :UR1:UR2:TTY:PTP
5180 3A 55 50 31 3A 55 50 32 3A 54 54 59 3A 43 52 54 :UP1:UP2:TTY:CRT
5190 3A 4C 50 54 3A 55 4C 31 3A 52 2F 4F 00 52 2F 57 :LPT:UL1:R/O.R/W
51A0 00 53 59 53 00 44 49 52 00 2F 4F 20 52 2F 57 20 .SYS.DIR-R/O R/W
51B0 20 53 59 53 20 44 49 52 20 2A 2A 20 41 62 6F 72  SYS DIR ** ABOR
-
```

V tem primeru zmo takoj za dvojico I in R uporabili še D ukaz z operandom (100H + pomik = 5100H) in se tako prepričali, da je bila zbirka STAT.COM v tem primeru naložena s pomikom nalagalnega začetka.

Eden najzanimivejših DDT podukazov je prav gotovo L ukaz, ki povzroči obratno zbiranje, to je preslikavo strojnega koda v zbirni zapis. Splošna oblika tega ukaza je

```
***********
* L'cr' *
***********
```

oziroma

```
***************
* Lssss'cr' *
***************
```

Žal velja ta ukaz samo za obratno zbiranje strojnega koda procesorja 8080A in v primeru uporabe ukaza nad strojnim kodom procesorja Z80 se pojavi sporočilo o napaki za operacijski kod, ki ne pripada procesorju 8080A. V našem primeru imamo:

```
-L4ED
  04ED  LXI  D,0000
  04F0  MVI  C,0C
  04F2  CALL 144F
  04F5  RET
  04F6  LXI  H,1528
  04F9  MOV  M,C
  04FA  LHLD 1528
  04FD  MVI  H,00
  04FF  XCHG
  0500  MVI  C,0E
  0502  CALL 144C
-
```

```
-XP
P=043D
-TS
C0Z0M0E0I0 A=00 B=0000 D=0000 H=1554 S=0100 P=043D SPHL
C0Z0M0E0I0 A=00 B=0000 D=0000 H=1554 S=1554 P=043E CALL 04ED
C0Z0M0E0I0 A=00 B=0000 D=0000 H=1554 S=1552 P=04ED LXI  D,0000
C0Z0M0E0I0 A=00 B=0000 D=0000 H=1554 S=1552 P=04F0 MVI  C,0C
C0Z0M0E0I0 A=00 B=000C D=0000 H=1554 S=1552 P=04F2 CALL 144F*144F
-
```

Primer kaže, da se je izpisalo 11 vrstic zbirnega koda od naslova 4EDH naprej. Z izpisom lahko sedaj enostavno nadaljujemo, tako da uporabimo L ukaz brez operanda in imamo:

```
-L
  0505  RET
  0506  LXI  H,152A
  0509  MOV  M,B
  050A  DCX  H
  050B  MOV  M,C
  050C  LHLD 1529
  050F  XCHG
  0510  MVI  C,0F
  0512  CALL 144F
  0515  STA  1527
  0518  RET
-
```

Seveda pa bi lahko z L ukazom izpisali tudi zbirni program v določenem naslovnem intervalu.

Z 'S' ukazom, katerega splošna oblika je

```
***************
* Sssss'cr' *
***************
```

beremo in spreminjamo (po želji) določeno skupino pomnilniških lokacij. Tako imamo primer:

```
-S100
0100 C3
0101 33 55
0102 04 00
0103 20 0E
0104 20 FF
0105 20 C9
0106 43 .
-
```

Vidimo, da smo spremenili vsebine lokacij v naslovnem intervalu (101H, 105H).

Ukaz za zasledovanje izvajanja nekega uporabniškega programa je bistven pripomoček pri razvoju programov, ko opazujemo, kaj se dejansko dogaja po izvrševanju posameznih ukazov. T ukaz opravlja to nalogo, seveda zopet samo za strojni kod procesorja 8080A. Splošna oblika tega ukaza je

```
***************
* Tnnnn'cr' *
***************
```

pri njegovi uporabi pa predhodno uporabimo še X ukaz, s katerim nastavimo vrednost programskega števnika. Tako dobimo primer:

Tu smo nastavili programski števnik na vrednost 43DH z X ukazom, nato pa smo izvedli pet ukazov od tega naslova naprej (uporaba T5 ukaza). Kot vidimo iz primera, smo pred vsakim ukazom dobili izpis vsebin vseh procesorskih registrov, zbrani ukaz in na koncu (po izvedbi zadnjega ukaza) še končno vrednost programskega števnika. Prva beseda v vsaki vrstici so zastavice statusnega registra, kjer pomenijo črke C, Z, M, E in I po vrsti prenos, ničlo, minus, sodo parnost in vmesni prenos. Nadalje je A akumulator, B sta registra BC, D registra DE, H registra HL, S je skladni kazalec in P programski števnik. Na koncu vrstice je prikazan še zbrani ukaz.

U ukaz omogoča pravkar opisani način opazovanja izvajanega programa po več ukazih (ne po vsakem) v odvisnosti od njegovega operanda. Splošna oblika teha ukaza je

```
***************
* Unnnn'cr' *
***************
```

Z uporabo tega ukaza štedimo s papirjem (če izpisujemo s tiskalnikom) in hitreje izvajamo program, saj opazujemo samo želene naslove oziroma posledice izvajanja ukaznih zaporedij. V našem primeru smo namenoma izbrali operand '1' in tako smo dobili:

```
-U
C0Z0M0E0I0 A=00 B=000C D=0000 H=1554 S=1550 P=0100 JMP  0433*0433
-U
C0Z0M0E0I0 A=00 B=000C D=0000 H=1554 S=1550 P=0433 LXI  H,0000*0436
-U
C0Z0M0E0I0 A=00 B=000C D=0000 H=0000 S=1550 P=0436 DAD  SP*0437
-U
C0Z0M0E0I0 A=00 B=000C D=0000 H=1550 S=1550 P=0437 SHLD 1532*043A
-
```

Tu se nam vsakokrat izpiše stanje programskega števnika po izvedenem zadnjem ukazu.

Za X ukaz imamo tele možnosti:

Xr ukaz	register ali zastavica	število bitov
XC	zastavica prenosa	1
XZ	zastavica ničle	1
XM	zastavica minusa	1
XE	zastavica sode parnosti	1
XI	zastavica vmesnega prenosa	1
XA	akumulator (A register)	8
XB	registrski par B in C	16
XD	registrski par D in E	16
XH	registrski par H in L	16
XS	skladni kazalec	16
XP	programski (ukazni) števnik	16

Kot vidimo je DDT prehodni ukaz močan pripomoček za popravljanje in razvoj programov, še posebej sistemskih. Za procesor Z80 pa imamo na voljo vrsto drugih CP/M pripomočkov in nekatere bomo opisali kasneje.

```
nZ      Počakaj n/2 sekund (pri taktu 2MHz)
n       Pomakni se naprej za 'n' vrstic in izpiši vrstico
'cr'    Pomakni se naprej za eno vrstico in jo izpiši
-       Pomakni se nazaj za eno vrstico in jo izpiši
n:x     Pomakni se za 'n' vrstic in izvedi ukaz 'x'
:mx     Izvedi ukaz 'x' nad vrsticami med trenutno vrstico
          in vrstico 'm'

        Opomba: Znak '-' se lahko uporabi pri vseh pomikalnih
                in prikazovalnih ukazih, ko se želimo pomak-
                niti nazaj (v smeri vrstic)
```

****
* ASM PRAVILA *
****

ASM je zbirnik za procesor 8080A in zanj veljajo tale pravila:

Označitev, ki ji sledi dvopičje ima lahko 1 do 6 alfanumeričnih znakov
Simbol (npr. pri EQU) brez dvopičja mora začenjati s črko, znakom '?' ali znakom '.'

Format zbirnega programa (polja so ločena s presledki) je tale:

```
označitev:  operator   operand(i)    ;komentar
```

Operatorji (brez predznaka) v operandnem polju:

```
a+b       'a' se prišteje k 'b'
a-b       'b' se odšteje od 'a'
+b        0+b (unarno seštevanje)
-b        0-b (unarno odštevanje)
a*b       'a' se pomnoži z 'b'
a/b       'a' se deli z 'b' (celoštevilsko)
a MOD b   Ostanek po deljenju a/b
NOT b     Komplementiranje vseh bitov v 'b'
a AND b   Bitna AND operacija nad 'a' in 'b'
a OR b    Bitna OR operacija nad 'a' in 'b'
a XOR b   Bitna XOR operacija nad 'a' in 'b'
a SHL b   Pomakni 'a' v levo za 'b' bitov z izgubo višjih bitov
            in z ničtimi nižjimi biti (ničta zapolnitev)
a SHR b   Pomakni 'a' v desno za 'b' bitov z izgubo nižjih bitov
            in z ničtimi višjimi biti (ničta desna zapolnitev)
```

Operatorska prednost:

```
najvišja:  *    /    MOD   SHL   SHR
           -    +
           NOT
           AND
najnižja:  OR   XOR
```

Tipi konstant:

numerični (z navedbo baze na koncu konstante):
- B = binarno
- O ali Q = oktalno
- D ali brez navedbe = decimalno
- H = heksadecimalno

nizni (tipa ASCII):
- v enojnih narekovajih (npr. 'aB')

Psevdooperacije zbirnika:

ORG	konst	Nastavitev programskega ali podatkovnega začetka (pri neuporabi ORG velja ORG 0000H)
END	start	Konec programa, ko lahko navedemo začetek (start) izvajanja programa
EQU	konst	Definiraj simbolno vrednost (ki ne more biti spremenjena v programu)
SET	konst	Definiraj simbolno vrednost (ki je lahko kasneje spremenjena)
IF	konst	Pogojno prevedi (zbirniško) blok do ENDIF
DS	konst	Določi pomnilniški prostor za kasnejšo uporabo
DB	zlog ,zlog,...,zlog	Določi zloge kot numerične ali nizne konstante
DW	beseda ,beseda,...,beseda	Določi besede (beseda ima dva zloga)

konst = konstanta; konstanta je logično pravilna, če je njen bit-0 = 1, sicer je nepravilna

---

****
* V/I ZLOG (0003H) *
****

V/I naprava	LST:	PUN:	RDR:	CON:
mesto bita	7 6	5 4	3 2	1 0
bitni par				
0 0	TTY:	TTY:	TTY:	TTY:
0 1	CRT:	PTP:	PTR:	CRT:
1 0	LPT:	UP1:	UR1:	BAT:
1 1	UL1:	UP2:	UR2:	UC1:

TTY: Teleprinter
CRT: Terminal s katodno elektronko
BAT: Paketni proces (RDR = vhod, LST = izhod)
UC1: Uporabniško določena konzola
LPT: Vrstični tiskalnik
UL1: Uporabniško določena naprava za listanje

```
PTR:    Bralnik papirnega traku
UR1:    Uporabniško določena bralna naprava
UR2:    Uporabniško določena bralna naprava
PTP:    Luknjalnik papirnega traku
UP1:    Uporabniško določen luknjalnik
UP2:    Uporabniško določen luknjalnik
```

BIOS VSTOPNE TOČKE

Heks naslov	Vektorsko ime	Funkcija	Vstopna vrednost	Izstopna vrednost
**00	BOOT	Hladen zagon sistema		C=0
**03	WBOOT	Topel zagon sistema		C=št.disk.enote
**06	CONST	Preizkus pripravljenosti konzole		A=const
**09	CONIN	Branje s konzole		A=znak
**0C	CONOUT	Vpis v konzolo	C=znak	
**0F	LIST	Vpis v napravo za listanje	C=znak	
**12	PUNCH	Vpis v napravo za luknanje	C=znak	
**15	READER	Branje iz bralnika		A=znak
**18	HOME	Pomik glave na stezo '0'		
**1B	SELDSK	Izberi disk. enoto ***	C=št.disk.enote	HL=dpg
**1E	SETTRK	Nastavi številko steze	C=štev. steze	
**21	SETSEC	Nastavi številko sektorja	C=štev. sektorja	
**24	SETDMA	Nastavi DMA naslov	BC=DMA	
**27	READ	Beri izbrani sektor		A=stdsk
**2A	WRITE	Vpiši v izbrani sektor		A=stdsk
**2D	LISTST	Vzemi status listanja		A=stlst
**30	SECTRAN	Subrutina za prevod sektorja	BC=lštsek DE=presek	HL=fsek

Pojasnila k tabeli:

```
const = status konzole          lštsek = logična štev. sektorja
   00 = neaktivna               fsek   = fizična štev. sektorja
   FF = pojavitev podatka       presek = preslikani sektorski
  dpg = naslov disk. parametra/          naslov
        /naslovne glave         znak   = ASCII znak
stdsk = status diska            št.    = številka
   00 = ni napake               štev.  = številka
   01 = pojavila se je napaka   disk.  = diskovne
stlst = status listanja         DMA    = DMA naslov
   00 = naprava zasedena        **     = vsebina lokacije 0002H
```



```
   FF = naprava pripravljena        ***    = če je v reg. E bit0 =0
                                             je to prva izbira disk. enote
```

KRMILNI BLOK ZBIRKE

Za zloge krmilnega bloka CP/M zbirke imamo tole:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
dr	f1	f2	f3	f4	f5	f6	f7	f8	t1	t2	t3	ex	s1	s2	rc
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15
32	33	34	35												
cr	r0	r1	r2												

Opis posameznih zlogov:

Zlogi		Funkcija
0	dr	Kod diskovne enote (0 = trenutni, 1 = A, 2 = B, 3 = C, 4 = D itd.)
1-8	f1-f8	Ime zbirke
9-11	t1-t3	Tip zbirke: t1 = 1(R/O), t2 = 1(SYS)
12	ex	Trenutna številka obsega
13	s1	Rezerviran zlog (00H)
14	s2	Vrednost =0 pri BDOS pozivu za OPEN, MAKE, SEARCH
15	rc	Števnik zapisov v obsegu
16-31	d0-dn	Napolni CP/M, rezervirano za sistemsko uporabo
32	cr	Trenutni zapis za branje ali vpis pri zaporedni zbirčni operaciji
33-35	r0-r2	Številka zapisa pri naključnem dostopu

DODELITEV POMNILNIKA

(V tekstu je b = obseg_pomnilnika - 20k zlogov)

Sistemsko pomnilniško (beležno) področje:

Heks lokacija	Vsebina
0-2	Skok v BIOS pri toplem zagonu sistema (vstopna točka)
3	V/I zlog (koncept prireditve periferije logičnim kanalom)
4	Trenutna številka diskovne enote in uporabnika
5-7	Skok v BDOS (v osnovni diskovni operacijski sistem)
8-37	Rezervirano za prekinitvene vektorje in prihodnjo uporabo

```
38-3A   RST7: uporabljata DDT in SID program
3B-3F   Rezervirano za prekinitveni vektor
40-4F   Beležno pomnilno področje za CBIOS (uporabniški V/I)
50-5B   Ni uporabljeno
5C-7C   Območje krmilnega bloka zbirke
7D-7F   Položaj naključnega zapisa
80-FF   DMA vmesnik (128 zlogov) za branje in vpissektorja
```

Območje prehodnih programov (ukazov):

Lokacije COM zbirk: 100H do 33FFH + b

Lokacije CP/M sistema:

```
Konzolni ukazni procesor (CCP):     od 3400H+b  do  3BFFH+b
Diskovni operacijski sistem (BDOS): od 3C00H+b  do  49FFH+b
Vhodni/izhodni sistem (BIOS):       od 4A00H+b  do  4FFFH+b
```

****
* CP/M DISKOVNI FORMAT (enojna gostota) *
****

```
Medij:          8-colski, mehko sektorirani upogljivi disk z e-
                  nojno gostoto zapisa (IBM 3740 standard)
Steze:          77 (oštevilčene z 0 do 76)
Sektorji/steza: 26 (oštevilčeni z 1 do 26)
Zlogi/sektor:   128 podatkovnih zlogov (en logični zapis)
Zlogi/disk:     256256 zlogov (77*26*128)
Obseg zbirke:   Katerokoli število sektorjev med nič in obsegom
                  diska
Zbirčna enota:  1k zlogov oz. 9 sektorjev je najmanjša enota, ki
                  se dodeli zbirki
Preskok:        Standardni preskok je 6 sektorjev (tj. prostor
                  med zaporednimi fizičnimi sektorji na stezi):
                  1-7-13-19-25-5-11-17-23-3-9-15-21-2-8-14-20-26-
                  6-12-18-24-4-10-16-22
CP/M sistem:    Stezi 0 in 1 sta lahko zasedeni s CP/M sistemom:
                  steza-0, sektor-1: navezovalni nalagalnik
                  steza-0, sektorji 2-26: CCP in BDOS
                  steza-1, sektorji 1-17: CCP in BDOS
                  steza-1, sektorji 18-26: uporabniški BIOS
Imenik:         Steza-2, sektorji 1-16, 32 zlogov/vstop,
                  64 vstopov/imenik (64 zbirk/disk)
Območje uporabniških zbirk:
                Od steze-2, sektorja-17 do konca steze-76; obseg
                  zbirke je 2 ali več (v kilozlogih)
```

****
* POZIVI BDOS FUNKCIJ (vstopna točka je 0005H) *
****

Ti pozivi so zahteve, poslane v BDOS, da se opravijo (izvršijo) posebne funkcije.

Periferni vhod/izhod:

0	00	sistemski reset	--	--
1	01	konzolno branje	--	A = znak
2	02	konzolni zapis	E = znak	--
3	03	bralniško branje	--	A = znak
4	04	luknjalni zapis	E = znak	--
5	05	listni zapis	E = znak	--
6	06	direktni konzolni V/I	E = FFH (vhod)	A=00H (ni pripravljen)
			E = znak (izhod)	A = znak
7	07	vzemi V/I zlog	--	A = V/I zlog
8	08	postavi V/I zlog	E = V/I zlog	--
9	09	tiskaj niz	DE = naslov niza	--
10	0A	beri konzolni vmesnik	DE = naslov podat.vmesnika	A = znaki v vmesniku
11	0B	vzemi konzolni status	--	A=00H (ni pripravljen) A=FFH (je pripravljen)

Diskovni vhod/izhod:

12	0C	vzemi CP/M verzijo	--	HL = verzija
13	0D	resetiraj disk	--	--
14	0E	izberi disk	E = štev. diska	--
15	0F	odpri zbirko	DE = FCB naslov	A = kod imenika
16	10	zapri zbirko	DE = FCB naslov	A = kod imenika
17	11	poišči zbirko	DE = FCB naslov	A = kod imenika
18	12	poišči naslednjo	--	A = kod imenika
19	13	zbriši zbirko	DE = FCB naslov	A = kod imenika
20	14	beri naslednji zapis	DE = FCB naslov	A = kod napake
21	15	vpiši naslednji zapis	DE = FCB naslov	A = kod napake
22	16	oblikuj zbirko	DE = FCB naslov	A = kod imenika
23	17	preimenuj zbirko	DE = FCB naslov	A = kod imenika
24	1	vrni login vektor	--	HL= kod d.enote
25	19	vrni številko diska	--	HL= štev. diska
26	1A	nastavi DMA naslov	DE = DMA naslov	--
27	1B	vzemi prir. vektor	--	HL= naslov prired. vektorja

Sistemske funkcije:

28	1C	vpisna zaščita	--	--
29	1D	vzemi R/O vektor	--	HL= R/O biti
30	1E	nastavi zbir. pridevke	DE = FCB naslov	A = kod imenika
31	1F	vzemi disk. parametre	--	HL= DPB naslov
32	20	nastavi/vzemi upor.kod	E = FFH	A = tr.up.štev.
			E = upor. kod	--
33	21	beri naključno	DE = FCB naslov	A = kod napake
34	22	vpiši naključno	DE = FCB naslov	A = kod napake

35	23	izrač.obseg zbirke	DE = FCB naslov	naklj.polje po.
36	24	nastavi naklj.zapis	DE = FCB naslov	naklj.polje po.
37	25	resetiraj d.enoto	DE = št. enote	A = 0
38	26	ni v uporabi		
39	27	ni v uporabi		
40	28	vpiši naključno z ničelno polnitvijo	DE = FCB naslov	A = kod vrnitve

Pojasnila:

znak = ASCII znak
V/I zlog = poseben mehanizem za prirejanje periferije z vsebino zloga na naslovu 0003H
FCB naslov = naslov krmolnega bloka zbirke
DPB naslov = naslov bloka diskovnih parametrov
tr.up.štev. = trenutna številka uporabnika
naklj.polje po. = postavitev poja v naključnem zapisu
št.enote = številka diskovne enote

kod napake:
01 = čitanje nezapisanih podatkov
03 = zaprtje trenutnega obsega ni mogoče
04 = poišči nezapisani obseg
05 = prestop imenika (samo pri zapisu)
06 = poišči prejšnji fizični konec diska

TEX UKAZI

TEX je enostaven procesor teksta in njegovi ukazi so tipa .ab
Vsak ukaz začenja v prvem stolpcu z znakom '.' (v spodnji tabeli so pike na začetku ukaza izpuščene).

Ukaz	Prekinitev ?	Zač. vrednost	Brez argumenta	Funkcija ukaza
AD	da	vkl		Nastavitev levega in desnega roba
BP +-n	da		+1 *	Začni novo stran
BR	da			Prekinitev (izpis nove vrstice)
CE n	da		1	Centriranje vrstice
CP n	da **			Pogojno začni novo stran
DS	da	izkl		Dvojni vrstični presledek
FT niz	ne			Zapis niza kot opombe na dnu str. (dolzina niza ni daljša od vrst.)
FT	ne			Izključitev prejšnje opombe
HE s	ne	izkl	prazno	Naslovna glava
HM +-n	ne	2		Nastavitev naslovnega prostora
IG	ne			Ne upoštevaj (ignoriraj)
IN +-n	da	0	0	Umaknitev teksta
LI	da			Dobesedni (izvirni) izpis

LL +-n	ne	70		Dolžina vrstice
LS n	da	1		Nastavitev presledkov med vrst.
MB +-n	ne	5		Spodnji rob strani
MT +-n	ne	6		Zgornji rob strani
NA	da	izkl		Ni nastavitve stranskih robov
OP	ne	izkl		Izpustitev številke strani
PA n	da		1	Izdaja 'n' praznih strani
PL +-n	ne	66		Dolžina strani
PN +-n	ne	1	1	Številka strani
PO +-n	da	8	0	Umaknitev levega roba
PP n	da	6	zadnji n	Naslov poglavja (umaknitev in presledek)
QI	da			Ukinitev veljavne umaknitve
SP n	da	1	1	Vstavitev 'n' praznih vrstic
SS	da	vkl		Enojni vrstični presledek
TI +-n	da	0		Začasna (enkratna) umaknitev

Opombi:
* velja le, če oštevilčevanje ni aktivirano
** velja le pri izdaji strani

UPORABA TEX PREHODNEGA UKAZA

Imamo tele oblike uporabe TEX prehodnega ukaza:

TEX ime_zbirke.tip
Imamo tekstovno obdelavo zbirke ime_zbirke tip in rezultat je zbirka ime_zbirke.PRN

TEX ime_zbirke.tip novo_ime
Obdela se ime_zbirke.tip, rezultat je zbirka novo-ime.PRN

TEX ime_zbirke.tip novo_ime.tap
Obdela se ime_zbirke.tip, rezultatna zbirka pa je v tem primeru novo-ime.tap

TEX ime-zbirke $X
To je obdelava s parametri izvajanja prehodnega ukaza TEX, ko imamo za X šest različnih parametrov, in sicer:

$C Konzolno listanje
$E Listanje napak na napravi za listanje
$F Uporaba oblikovnega krmiljenja na tiskalniku (seveda če ta možnost obstaja)
$L Izhod se izda na napravo za listanje
$P Tiskanje se ustavi na koncu strani, tako da je mogoče zamenjati papir; tiskanje se nadaljuje z znakom 'cr' ('CTL m')
$S Izhodna zbirka se ne oblikuje, sporočila o napakah se tiskajo

INFORMATICA 4/1981

STANDARDI I POLITIKA STANDARDIZACIJE U OBLASTI INFORMATIKE

S. BRAJOVIĆ – BRATONOVIĆ
B. DŽONOVA – JERMAN – BLAŽIČ

UDK: 389.6:681.3

SAVEZNI ZAVOD ZA STANDARDIZACIJU, BEOGRAD
INSTITUT JOŽEF STEFAN, JAMOVA 39, LJUBLJANA

Razmatrana je problematika i ciljevi standardizacije za područje informatike i računarske tehnike. Razradjena je metodologija rada na standardima u oblasti informatike kao i srednjeročni plan rada Grupe za standardizaciju Saveznog zavoda za standardizaciju u oblasti informatike.

STANDARDS AND STANDARDISATION POLICY IN THE FIELD OF INFORMATICS AND COMPUTER SCIENCE. The paper discuss the problems encountered in the development of standards in the field of informatics and computer science as well as the standardisation policies. The areas and procedures of standardisation are treated too. The working plans of the Yugoslav Committee for standardisation in the field of informatics for the next five years are presented.

1. UVODNE MISLI O STANDARDIZACIJI

Racionalno korišćenje savremenih rezultata nauke i tehnologije bez razvijanja i korišćenja funkcija i instrumentarija standardizacije praktično je nemoguće ili, u najboljem slučaju, osetno reducirano. Negativne posledice tehničko-tehnološkog razvoja nemoguće je držati pod kontrolom bez konstantne i smišljene politike, odnosno bez korišćenja bogatog instrumentarija standardizacije kao produkta i pratioca razvijenog društva. Borba za kvalitet proizvoda zahteva utvrdjivanje kvalitetnih indikatora i metoda kao i njihovu egzaktnu proveru, koju upravu utvrdjuje i definiše standardizacija u svim oblastima nauke i tehnike.

Privreda Jugoslavije se razvija pod jakim uticajem privreda razvijenih industrijskih zemalja. Naša zemlja je preuzimala i preuzimaće znanje i tehnologiju višeg nivoa.
Medjutim, ekonomski je neopravdano, spontano i nekontrolisano unošenje tehnologija raznih nivoa, različitih, nekompatibilnih tehničkih sistema, uz pritisak velikog broja suprotnih interesa medjunarodno organizovanih grupacija krupnog kapitala radi prisustva na našem tržištu. Problemi tipizacije i unifikacije sredstava, opreme, zamenljivosti delova su tehnički i ekonomski problemi razvoja zemlje, njene razvojne orijentacije i ekonomske sposobnosti da se uspešno razvija u granicama svojih mogućnosti i efikasno uključuje u medjunarodnu podelu rada.

Brojni problemi ubrzanog razvoja ne mogu se rešiti samo standardima odnosno tehničkim normativima, normama kvaliteta, vancarinskom zaštitom i uklanjanjem tehničkih barijera u medjunarodnom prometu roba, all se ceni da se dobrih 50 % problema vezanih za korišćenje, eksploataciju i ekonomičnije privredjivanje mogu eliminisati blagovremeno donetom i koncepcijski savremenom tehničkom regulativom. Ekonomske efekte nije jednostavno izračunati, all se oni sigurno odnose na milijarde novih dinara svake godine.

Poslednje decenije se u industrijski razvijenim zemljama, i ne samo u njima, stvaraju novi, širi pogledi u odnosu na standardizaciju. Osnovnim ciljevima standardizacije se smatraju uprošćavanje rastućeg broja varijanti proizvoda i postupaka u životu čoveka, olakšano sporazumevanje, opšta ekonomičnost, bezbednost, zdravlje i zaštita života, zaštita interesa potrošača i interesa društva i odstranjivanje prepreka za trgovinsku razmenu.
Osnovni elementi ovog koncepta sistema integralne standardizacije jesu:

1. izrada programa i propisa u vezi sa unifikacijom i tipizacijom, transferom tehnologije i sl.,
2. definisanje društveno opravdanog kvaliteta i propisivanje ovog kvaliteta,
3. garantovanje, nadzor i kontrolisanje kvaliteta, posebno u slučajevima bezbednosti i zaštite.

U Jugoslaviji su stvoreni preduslovi za ostvarenje sistema integralne standardizacije (Zakon o standardizaciji, Zakon o mernim jedinicama i merilima itd.). Ovaj sistem treba da bude povezan sa: obezbedjivanjem jedinstva tehničkih i tehnoloških sistema, obezbedjenjem jedinstva jugoslovenskog tržišta, jačanjem odbrambene sposobnosti naše zemlje, zaštitom života i zdravlja ljudi, životne sredine, društvenih sredstava i imovine gradjana, razvojem i unapredjivanjem proizvodnje i prometa, tipizacijom i unifikacijom, racionalnim korišćenjem energije, razvojem i unapredjenjem kvaliteta proizvoda, otklanjanjem tehničkih prepreka poslovno-tehničkoj saradnji, kooperaciji, specijalizaciji i prometu roba, zaštitom potrošača, naročito u pogledu bezbednosti pri upotrebi proizvoda, pouzdanosti i trajnosti proizvoda itd.
Da bi se postigli ovi ciljevi, potrebna je uska povezanost i koordinacija sledećih aktivnosti: izrade standarda, tehničkih propisa i ostalih sličnih akata, primene standarda, tehničkih propisa i ostalih akata, razvoja i istraživanja u oblasti standardizacije, uključujući i medjunarodnu saradnju, kontrole kvaliteta proizvoda pri izvozu i uvozu, ispitivanja, metrologije, pregleda merila i merne opreme, inspekcijskog nadzora nad sprovodjenjem i primenjivanjem propisa i standarda povezanih sa standardizacijom.

2. NIVOI I NOSIOCI STANDARDIZACIJE

Sistem standardizacije sastoji se od više nivoa aktivnosti: internog, granskog i aktivnosti velikih tehničkih sistema, nacionalnog, regionalnog i medjunarodnog. Nosioci aktivnosti su: na internom nivou – organizacije udruženog rada, na granskom nivou – organizacije udruženog rada preko svojih

asociacija i zajednica, na nacionalnom nivou - federacija preko Saveznog zavoda za standardizaciju, na regionalnom nivou - grupe zemalja organizovane u regionalne celine, na medjunarodnom nivou - specijalizovane organizacije za standardizaciju, kao i pojedina tela mnogobrojnih medjunarodnih vladinih i nevladinih organizacija.

Sve pomenute nivoe standardizacije su jednako značajne za sam sistem standardizacije. Razvijati i jačati istovremeno sve nivoe standardizacije znači obezbedjivati ekonomičan i usklađen sistem standardizacije. Standardi nižeg nivoa po sadržaju moraju biti u skladu sa standardima višeg nivoa. standardizacije. Oni obično imaju i mnogo više detalja, a i broj standarda idući ka nižim nivoima raste. U svom radu Savezni zavod za standardizaciju (SZS) aktivno uključuje stručnjake iz organizacija udruženog rada, koji su i stručni nosioci poslova u svim telima za donošenje standarda i brojnih dokumenata na svim nivoima.

Savezni zavod za standardizaciju, kao nosilac jugoslovenske standardizacije, ima sledeće zadatke:
- da organizuje rad na donošenju jugoslovenskih standarda, tehničkih normativa i normi kvaliteta proizvoda i usluga od značaja za jugoslovensku samoupravnu zajednicu,
- da inicira i pomaže razvoj standardizacije na nižim nivoima standardizacije, sa ciljem da se što potpunije i skladnije razvija jedinstveni sistem jugoslovenske standardizacije na svim nivoima, u svim oblastima i u svim regionima zemlje,
- da organizuje i kontroliše sprovodjenje sistema atestiranja u Jugoslaviji,
- da organizuje koordinirano učešće jugoslovenskih sistema standardizacije u medjunarodnom radu na standardizaciji, sa ciljem da se obezbedi uticaj jugoslovenske privrede i društva na razvoj medjunarodne standardizacije, a u skladu sa našim potrebama i mogućnostima,
- da ostvari odgovarajuću saradnju sa sistemima standardizacije drugih zemalja, u cilju medjusobne pomoći i saradnje.

3. STANDARDIZACIJA U OBLASTI INFORMATIKE I RAČUNARSKE TEHNIKE

3.1. Sadašnje stanje kod nas i ciljevi standardizacije

Razmatrajući pitanje standardizacije u oblasti informatike u našoj zemlji, nemoguće je oteti se utisku da ovom pitanju nije posvećena dovoljna pažnja. Već sasvim površan pregled o delatnosti u informatici u drugim zemljama, kakav je dat u tabeli 1 govori o znatnom zaostajanju u ovoj oblasti. Ova zaostajanje može se delimično objasniti i činjenicom da je razvoj informatike u našoj zemlji u odredjenoj meri zaostao za razvojem informatike u zemljama pomenutim u tabeli 1, koje su na znatno višem stupnju tehnološkog razvoja.

Danas, medjutim, kad u našoj zemlji već postoji prilično veliki broj instalacija računara srednjeg i velikog kapaciteta i kada mnogi od velikih korisnika već prevazilaze nivo obrade po partijama, dalje zaostajanje u ovoj oblasti moglo bi imati ozbiljnih posledica na razvoj tako značajnih disciplina kao što su informatika i računarska tehnika.

Većina većih organizacija u privredi, bankarstvu, administraciji i uslužnim delatnostima poseduje računare različitih kapaciteta na kojima se odvija obrada velikog broja podataka od nesporne važnosti za obezbedjenje normalnog procesa privredjivanja u zemlji. Ustanovljeno je da vrlo često dolazi do podvajanja podataka u srodnim i različitim institucijama odnosno da se isti podatci skupljaju i obradjuju na više različitih mesta bez svake potrebe.
Zato obezbedjenje uslova za razmenu informacija na magnetnim nosiocima na klasičan način ili uz upotrebu savremenih komunikacionih sredstava predstavlja preku potrebu.

Standardizacija u ovoj oblasti nužna je i radi obezbedjenja kompatibilnosti, koherentnosti, konzistentnosti i medjusobne povezanosti sistema naročito u situaciji kad mnoge od ovih velikih sistema treba uskladiti u smislu zakona o društvenom sistemu informisanja.

Nedostatak standarda* u oblasti informatike mogao bi postati stvarna kočnica usklađenog razvoja velikih informacionih sistema u našoj zemlji takodje iz sledećih razloga. Budući da je postojanje odredjenih standarda preduslov daljeg razvoja informacionih sistema, a da standardi koji su uradjeni od kompetentnih stručnjaka i preporučeni od ovlašćene i autoritativne institucije ne postoje, vrlo je verovatno da će
- doći do standardizacije na internom nivou koja neće biti u skladu sa ostalima pa će kao takva u budućnosti predstavljati kočnicu razvoja,
- specifikacije internih standarda koji, usled nedostatka vremena i neobaveštenosti, neće predstavljati optimalna rešenja i odstupaće od uobičajene prakse,
- početi korišćenje standarda drugih zemalja, verovatno zemalja isporučilaca opreme, koji mogu biti i kontradiktorni,
- kvalitet standarda direktno će zavisiti od sposobnosti raspoloživih subjekata pojedinim računskim centrima itd.

Posebni napori i troškovi kod prelaska sa internih na opšte standarde će u ovim slučajevima predstavljati i stvarnu prepreku uspešnoj standardizaciji na višem nivou.

* U oblasti informatike je registrovano 2 standarda: A.F0.004 koji se odnosi na simbole dijagrama sistema obrade informacija i A.F0.039 koji se odnosi na nebušene papirne kartice.

Tabela 1.

Standardi u oblasti informatike u nekim tehnološki razvijenim zemljama i našoj zemlji

	Obrada podataka	Oprema	Dokumentacija	Programski jezici	Prenos podataka	Raspoznavna oblika	Kartice	Papirna traka	Diskovi	Kasete	Diskete	Magn. trake	Ostalo	Ukupno
SAD	6	4	13	7	11	8	2	5	2	2	-	8	-	68
Vel. Britanija	6	4	6	-	3	2	2	1	5	2	2	8	4	45
Francuska	7	2	18	4	9	6	4	10	5	6	1	16	1	89
Nemačka	11	8	19	9	21	2	7	4	11	9	11	21	7	140
ISO - postojeći	6	1	17	4	10	4	4	11	5	4	-	14	1	81
ISO - u pripremi	9	11	10	5	9	1	1	1	5	-	4	4	-	60
Jugoslavija	-	-	1	-	-	-	1	-	-	-	-	-	3	5

SAD - ANSI, Vel. Britanija - BSI, Francija - NF, Nemačka - DIN

Zbog toga je neophodno pristupiti intenzivnoj standardizaciji u oblasti informatike koja će omogućiti postizanje sledećih ciljeva:

- efikasnije iskorišćenje opreme za elektronsku obradu podataka obezbedjenjem potrebnog nivoa kompatibilnosti računara i perifernih uredjaja,
- zaštitu jugoslovenskog tržišta i obezbedjenju normalnih uslova rada za jugoslovenske korisnike,
- efikasnu razmenu podataka na odgovarajućim magnetnim medijumima na unapred utvrdjen način,
- povezivanje računara u cilju optimalnog iskorišćenja resursa,
- teleprenos i teleobradu podataka i poboljšanje kvaliteta informisanja,
- nesmetan razvoj velikih informacionih sistema sa centralizovanim i/ili distribuiranim bazama podataka,
- uputstva i smernice za rad na razvoju informacionih sistema,
- edukativnu funkciju u sredinama gde nije došlo do intenzivnog razvoja informatike,
- smernice za razvoj domaće proizvodnje sredstava za obradu i prenos podataka

3.2. Problem standardizacije za područje informatike i računarske tehnike

Informatika je multidisciplinarna nauka. Oblast informatike, je definisana od strane ISO-a (International Standard Organization) postojanjem tehničkog komiteta 97 sa svojih 16 radnih grupa. Obuhvata preseke naučnih disciplina elektrotehnike, telekomunikacija, numeričke matematike, teorije informacija, algebre, teorije redova, ekonomije, teorije upravljanja, dokumentalistike itd. Oblasti rada pojedinih radnih grupa su takodje multidisciplinarne. Teško je verovati da može postojati stručnjak koji u dovoljnoj meri vlada svim ovim oblastima da bi mogao sam raditi valjane standarde, kako je to u okviru Saveznog zavoda za standardizaciju (SZS) uobičajeno za druge oblasti standardizacije. Rad u informatici prema tome i pre svega mora da bude timski.

Informatika kao oblast je izuzetno dinamična. Nagli razvoj tehnologije i značaj koji je informatika u svetu dobila poslednjih godina ima za posledicu da se u ovoj oblasti neprekidno dešavaju značajni prodori tehnike i nauke tako da rešenja veoma brzo tehnički zastarevaju. U poslednjih 25 godina arhitektura, organizacija, oblast i mogućnosti primene računara četiri puta su doživeli dramatične promene. To praktično znači da znanja u odredjenoj oblasti informatike zastarevaju u roku 5-7 godina. Jasno je da i ova činjenica ima velikog uticaja na pristup standardizaciji, kako na brzinu donošenja standarda tako i na potrebu da se pojedini standardi često revidiraju. Više nego u drugim oblastima proces standardizacije i destendardizacije mora biti kontinualan i vodjen od strane stručnjaka koji dobro poznaju konkretnu problematiku predmeta standarda.

Novi prodori u oblasti tehnologije dovode do toga da se problematika i oblasti kojima se informatika bavi pomeraju. Ilustracija ove činjenice prikazana je u tabeli 2.

Kako se iz pregleda može sagledati, na primer papirna traka kao medijum računara nije nigde standardizovana posle 1975. godine budući da je prevazidjena iako je oko 1970 to bila jedna od glavnih oblasti standardizacije. Sa druge strane diskete su počele da bivaju predmet standarda tek od 1979. itd.

Imajući u vidu teškoće u radu zbog multidisciplinarnog karaktera informatike i nauke o računarima kao i teškoće nastale usled raznog menjanja oblasti standardizacije, mnoge zemlje su se u ovoj oblasti pretežno oslonile na medjunarodnim standardima.

Tabela 2.

Godine donošenja standarda u pojedinim oblastima

	Papirna traka	Kartice	Trake	Prenos podataka	Kasete	Diskete
ANSI	71-74	67-70	73-77	75-79	78-80	-
BS	70	71	72-75	76-79	78-80	-
NF	67-72	70-71	71-74	72-74	77-79	79-80
DIN	67-69	75-77	74-77	77-80	77-80	79-80
ISO	72-75	74-77	75-78	76-80	77-79	80

Medjunarodna standardizacija u oblasti informatike se odvija preko Medjunarodne Organizacije za Standardizaciju (ISO), koja donosi medjunarodne standarde za sve oblasti osim za oblast elektrotehnike* i preko Medjunarodnog saveza za telekomunikacije (prenos podataka), koji u okviru svoje delatnosti utvrdjuje standarde, uputstva, preporuke ili slične dokumente koji po svom karakteru odgovaraju medjunarodnim standardima. U početnom periodu ovog rada ove dve organizacije nisu mogle, da igraju značajniju ulogu zbog toga što je bilo teško usaglašavati već razvijene i stvorene sisteme u pojedinim zemljama.

ISO nije vladna organizacija, ona skuplja nacionalne instrukcije za standardizaciju koje su u većini kapitalističkih zemalja stvorene inicijativom privrede. Za ilustraciju u tabeli 3 navodimo područje rada najpoznatijih nacionalnih i medjunarodnih institucija koje rade na standardima za oblast informatike i računarske tehnike.

Poznato je, da na medjunarodnom tržištu ne može opstati onaj koji ne usvaja medjunarodne standarde, takodje je poznato da najveći profit ostvaruje onaj čija rešenja udju u medjunarodne standarde. Zato u poslednjih petnaeset godina dolazi do značajnog širenja područja medjunarodne standardizacije, a i osetnog porasta interesa za rad na medjunarodnoj standardizaciji. Pored industrijskih zemalja, koje su tradicionalno učestvovale u radu ISO, sve više i više zemalja u razvoju nalaze svoj interes u radu ISO, IEC i upravo te zemlje danas predstavljaju većinu u radu ISO. Doprinos zemalja u razvoju tehničkim aktivnostima ISO, IEC je možda ograničen, ali je za njih od velikog interesa da medjunarodni standardi budu prihvaćeni i, da je moguće njihovo korišćenje, budući da ti standardi predstavljaju politički i ekonomski neutralan način za transfer tehnologije koji je za njih od izuzetnog značaja.

Posebno u informatici, koja je oblast izuzetno kompleksna i dinamična, mnoge zemlje se direktno uključuju u rad na ISO standardima pa ISO standarde posle prihvataju bilo direktno bilo kao osnov za svoje nacionalne standarde (Nemačke, Holandija, Francuska itd.). Čista nacionalna standardizacija obavlja se obično u onim oblastima informatike gde pojedine zemlje imaju neki poseban interes.

Tabela 3.

1. British Standard Institut

oblasti izrade standarda:

- kodovi
- papirna traka
- prenos podataka
- reprezentacija podataka
- povezivanje sredstava za obradu podataka
- definicija i analiza problema
- numeričko upravljanje
- uredjaji za administrativno poslovanje (office machines)

*Taj zadatak izvršava Medjunarodna elektrotehnička komisija (IEC).

Tabela 3. (produžetak)

- bušene papirne kartice
- magnetne trake i magnetni diskovi
- magnetne trake za potrebe instrumentarija
- struktura znakova i datoteka
- programski jezici
- prepoznavanje znakova i ADP
- rečnici pojmova
- sigurnost i zaštita uredjaja za obradu podataka

2. American National Standards Institute

(ne radi sam na standardima već organizuje rad uz pomoć stručnih udruženja i vladnih organizacija)

- optičko prepoznavanje znakova
- magnetna traka
- bušene kartice
- programski jezici (Cobol, Algol, Basic)
- simboli za dijagrame sistema obrade informacija
- računarske mreže
- prenos podataka
- tastature
- bušena papirnata traka
- magnetni nosioci podataka
- Fortran, APL
- šifriranje
- dokumentacija projekata
- rečnici pojmova
- kodovi
- reprezentacija podataka
- alfanumerički uredjaji

3. International Organization for Standardization

Članice ISO su nacionalne institucije za standardizaciju i druge medjunarodne organizacije, koje se bave ovim pitanjima, za informatiku je zadužen tehnički komitet 97 sa sledećim radnim grupama:

- rečnici pojmova
- prepoznavanje znakova
- prenos podataka
- uredjaji za numeričko upravljanje
- magnetni diskovi
- magnetne trake za računare
- povezivanje uredjaja
- skupovi znakova i kodiranje
- struktura datoteka
- programski jezici
- dokumentacija informacionih sistema
- programski jezici za numeričko upravljanje
- magnetna traka za registrovanje merenja
- reprezentacija podataka
- traka za štampanje,
- alfanumerički uredjaji za administrativno poslovanje (alphanumeric office machines)
- povezivanje otvorenih sistema

4. ECMA, European Computer Manufactors Association razvija standarde za udruženje evropskih proizvodjača računara. Ima sledeće tehničke komitete:

- medijumi nosioci podataka
  - magnetni diskovi
  - magnetne trake
  - diskete
  - kasete
- programski jezici
- prepoznavanje znakova
- prenos podataka
  - protokoli
  - osnovne kontrolne procedure
  - kontrolne procedure visokog nivoa

Tabela 3. (produžetak)

- reprezentacija podataka
  - kodiranje
  - formati
  - labelisanje
- dijagrami toka
- tastature
- matrični štampači
- sigurnost i elektromagnetna interferencija

5. CCITT

Iz oblasti informatike radi na preporukama u vezi sa prenosom podataka preko sledećih tehničkih komiteta:

- prenos podataka telefonskim mrežama
  - opšte
  - interfejsi i modemi
  - kontrola grešaka
  - kvalitet prenosa
- prenos podataka u mrežama za prenos podataka
  - usluge i osobine mreža
  - interfejsi u mrežama
  - signalizacija u mrežama
  - kvalitet prenosa
  - klase korisnika u mrežama

6. International Electrotechnical Commission (IEC)

radi u koordinaciji sa ISO za oblast elektrotehnike i u oblasti računskih mašina pokriva sledeće oblasti:

- simboli i dijagrami povezivanja strujnih kola
- elektronska instrumentacija
- magnetne memorije - električne specifikacije
- sigurnost uredjaja.

Kako se ni razvoj jugoslovenske privrede ne može drugačije posmatrati već kao deo u mehanizmu svetskog razvoja, to se i standardizacija u našoj zemlji mora posmatrati kako u kontekstu donošenja standarda potrebnih našem sadašnjem stupnju razvoja tako u sklopu medjunarodne standardizacije. Rad na jugoslovenskoj nacionalnoj standardizaciji mora se meriti ne samo po broju i potrebi posebnih "jugoslovenskih rešenja" (na primer u oblasti informatike, to su rečnici pojmova, jugoslovenski latinički i ćirilički kodovi, izbor opreme, dokumentacija informacionih sistema i sl.) već i po broju rešenja koja pogoduju našoj privredi i društvu i koja su prihvaćena od drugih zemalja na nivou medjunarodnih organizacija za standardizaciju.

4. REFERENCE

U pripremi materijala korišćeni su sledeći dokumenti:

1. Politika standardizacije u Jugoslaviji, Beograd 1979, izdavač SZS, odgovorni urednik Milan Krajnović.
2. Srednjeročni plan rada na standardima u oblasti informatike, jun 1981, SZS, autor S. Brajović-Bratanović.
3. P. Wolley, Standards in Computing, S. Tech. College, 1928, London.

(II. deo članka, biće objavljen u sledećem broju časopisa Informatica)

INFORMATICA 4/1981

SISTEMSKA OBNOVA U USLOVIMA REALNOG VREMENA

MARKO KOVAČEVIĆ, dipl. ing.

DO DELTA, SOZD ELEKTROTEHNA

UDK: 681.326.7

Kompleksni računarski sistemi, uključeni u procese realnog vremena moraju zadovoljavati uslove integriteta, sigurnosti i kontinuiteta rada. Kompleksnost, konstrukcijske greške, nepredvidjene okolnosti te fizičke greške u sistemu mogu biti uzrok za neregularan i neželjen tok procesa. Zato su u ovakve sisteme ukomponirani materijalni i programski dijagnostični elementi što stvara uslove za održavanje kontinuiteta rada sistema i pored grešaka bilo kakve prirode. U članku je dana osnova za praktičnu realizaciju materijalne i programske opreme za obnovu sistema u realnom vremenu tako da su zadovoljeni navedeni uslovi rada sistema u realnom vremenu. Posebno je obradjena problematika obnove u multiprocesorskim sistemima

SYSTEM RECOVERY IN REAL TIME: Computer systems included into real time processes must provide integrity, sequriti and continuity of work. Construction bugs, physical errors and unforeseen conditions can cause irregular and undesirable flow of process. Therefore, we must include hardware and software diagnostic tools into real time systems which make the base for system recovery in real time. In this article the practical basis for implementation of recovery procedures are given, especially the recovery in multiprocessor systems.

1. Uvodna besjeda

Bitna osobina savremenih digitalnih sistema je tolerancija grešaka i automatska obnova sistema do regularnog djelovanja ili bezbjednog deaktiviranja poslije detekcije grešaka. Tolerancija grešaka logičkih mašina predstavlja tehnološko oživljanje riječi Šekspirovog junaka: "Biti ili ne biti! Pitanje je sad", dakle predstavlja pitanje opstanka sistema. Dodatna materijalna i programska oprema potrebna za primjenu principa tolerancije grešaka je redundančna u toku normalnog djelovanja logične mašine. Tolerancija grešaka ima za cilj da zaštiti logičku mašinu od nesavršenosti fizičkog sistema na kojemu je zasnovana te od konstrukcijskih grešaka u materialnoj ili programskoj opremi, kao i od neprimjernih akcija operatera u toku eksploatacije i održavanja sistema. Metodologija povećanja sigurnosti sistema koja je u proteklosti bila više primjenjivana od principa tolerancije se može označiti kao metodologija izbjegavanja. Izbjegavanje grešaka u digitalnim sistemima povlači za sobom visoku cijenu komponenata te klasični način lokacije i otklanjanje grešaka u, ipak mogućim slučajevima. Glavni argument u prilog principa tolerancije je to što početna investicija za realizaciju tolerantnosti može reducirati doživotnu cijenu razvoja eksploatacije i održavanja sistema. Dodatni početni troškovi su posljedica napora realizacije tolerantnosti u toku razvoja sistema te cijenom dodatnih materijalnih i programskih komponenata koje vrše funkciju detekcije grešaka i obnove sistema. Niži doživotni troškovi eksploatacije i održavanja sistema imaju dva osnovna razloga. Prvi se sastoji u nižoj cijeni komponenti sistema koji ne moraju ispunjavati uslove izbjegavanja grešaka, dakle visoke sigurnosti rada. Drugi razlog je logična posljedica organizacije sistema sa tolerancijom grešaka, koji će i poslije greške automatski nastaviti regularan rad, tako da nije nužna fizička prisutnost servisnog osobija uz sam sistem.

Veoma je širok spektar aplikacija digitalnih računarskih sistema gdje je primjena principa tolerancije grašaka i neprekidnosti rada sistema od izuzetnog značaja. Prije svega to su aplikacije kod kojih greške mogu imati za posljedicu opasnost za ljudske živote. Takve su aplikacije kontrole avionskog, željezničkog, drumskog, svemirskog prometa, kontrola

nuklearnih energetskih objekata i odbrambenih sistema i slično. Aplikacije gdje čak i kratak ispad računarskog sistema može imati teške ekonomske posljedice moraju biti obezbijedjene od grešaka fizičkog ili ljudskog porijekla. Takvi su na primjer veliki vremensko dijeljeni sistemi, elektronski telegrafski i drugi komunikacijski sistemi, kontrola proizvodnje i potrošnje energije, procesna kontrola u automatiziranim fabrikama itd. Zaštita sistema od grešaka bilo kakve vrste po principu tolerancije je od izuzetnog značaja u uslovima koji ne omogućavaju ručno održavanje kao što su sateliti, medjuplanetarni svemirski sistemi, teško-dostupni udaljeni sistemi za raznovrsna osmatranja itd. Zajednička osobina svih sistema koji zahtijevaju toleranciju grešaka svih vrsta je rad u realnom vremenu. Dakle ovi sistemi moraju ispunjavati osnovni uslov da u potpunosti slijede i drže pod kontrolom proces kojemu su pridruženi, te da u slučaju greške obnove regularan rad u dovoljno kratkom vremenu.

2.Osnove za realizaciju tolerantnosti

U proces razvoja sistema sa zahtijevima tolerancije grešaka moramo uključiti i posebne module o specifikaciji grešaka koje je potrebno tolerisati, o postupcima za detekciju grešaka te o postupcima obnove koji se aktiviraju na osnovu signala detekcije, a imaju zadatak vratiti sistem na neki nivo normalnog djelovanja. Opseg signaliziranе neregularnosti materijaalne opreme, podataka ili programa te brzina kojom normalno stanje mora biti uspodtavljeno diktiraju izbor tehnike obnove. Obnova se sastoji iz svih akcija koje mogu biti izvršene poslije identifikacije greške. To može ukljućivati korekciju greške, lokalizaciju greške, isključenje ili zamjena pokvarenih elemenata, zapis poduzetih akcija u cilju kasnije rekonstrukcije te ponavljanje starta normalnog rada ili bezbjedne obustave rada. Detekcijski algoritmi su implementirani u materijalnoj ili programskoj opremi, najčešće kombinirano. Prema vremenu njihove aktivnosti razlikujemo nekoliko vrsta detekcijskih algoritama.

Početno testiranje se vrši prije normalne upotrebe sistema, a ima za cilj otkriti kako fizičke greške u materijalnoj opremi tako i konstrukcijske greške u programskoj i materijalnoj opremi.

Konkurenčna detekcija se vršl simultano sa osnovnom djelatnošću sistema Ova metoda je najuspješnija i najkorisnija pošto neprestano prati rad sistema te signalizira u slučaju neregulanosti. U sljedećim glavama će bi najviše riječi o konkurenčnoj metodi detekcije grešaka.

Detekcija u pasivnom stanju sistema se vrši onda kada je noramlan rad privremeno preklnut. Ovakav način detecije ne zadovoljava uslove tolerancije i obnove u sistemima realnog vremena.

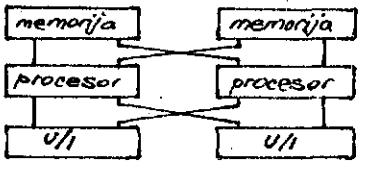
Testiranje redundanci ima za cilj verifikaciju ispravnosti redundantnih elemenata u sistemu, a može se vršiti konkurenčno ili za vrijeme pasivnog stanja sistema.

Detekciji greške u procesu tolerancije slijede algoritmi obnove sistema. Uopšte, ovi algoritmi mogu biti automatski ili ručni. Automatski algoritmi obnove ne zahtijevaju posredovanja operatera u toku procesa obnove. Izključenje operatera iz ovog procesa je posljedica potrebe za ekstremnim smanjenjem vremena potrebnog za kompletiranje procesa obnove. Prema stanju sistema poslije procesa automatske obnove razlikujemo potpunu obnovu, djelimičnu obnovu te bezbjedno deaktiviranje sistema. Pri djelimičnoj obnovi sistem ponovo prelazi u stanje normalnog rada ali sa reduciranim kapacitetima. Ukoliko u procesu obnove ne uspijemo oživiti osnovne funkcije sistema pristupa se procesu bezbjednog deaktiviranja sistema, što znači takav prestanak djelovanja koji nebi imao destruktivne posljedice na okolicu sistema, pa ni na sistem sam. Ručni algoritmi obnove zahtijevaju posredovanje operatera što umanjuje njihovu vrijednost u sistemima realnog vremena pošto najčešće nije obezbijedjena dovoljno brza reakcija na signaliziranu grešku. Algoritmi obnove mogu biti realizirani materijalno ili programski. Materijalna obnova temelji na specijalnoj materijalnoj opremi čiji zadatak je sakupljati signale o greškama te inicializirati proces obnove. Programska obnova temelji na posebnim programima koji imaju zadatak inicializirati i kontrolirati proces obnove. Pri tome su signali o greškama obezbijedjeni putem materijalnih ili programskih modula kao što je kontrola parnosti, zaštita upisa u memoriju, komparatori, watchdog časovnici, testni programi i slično.

2.1. Multiprocesiranje i tolerancija

Organizacija multiprocesorskih sistema (tu ubrajam i distribuirane sistome) nudi vrlo pogodno tlo za primjenu metoda tolerancije grešaka. Sasvim je opravdana tvrdnja da je integritet (sigurnost, kontinuitet rada, neosjetljivost na greške) multiprocesorskih sistema bolji od integriteta jednoprocesorskih sistema. Glavni razlog za takvu tvrdnju je znatno veća mjera prisutnosti redundantnosti u sistemima sa više procesora nega u sistemima sa jednim procesorom. Isto tako, ostvarena autonomnost pojedinih modula sistema koji komuniciraju sa drugim modulima sprečava širenje greške preko cijelog sistema. Time se problem obnove svodi na problem obnove podsistema. Tako je tolerancija grešaka i povećanje sigurnosti sistema, pored povećanja sposobnosti i fleksibilnosti, jedan od glavnih razloga za gradnju multiprocesorskih sistema. Sistemi sa više procesora, koji nude potporu principu tolerancije grešaka se pojavljaju u sljedećim konfiguracijama:

Dvojni sistemi imaju podvojene sve elemente sistema uključujući i centralnu procesorsku jedinicu. Oba sistema sakupljaju, pamte i obradjuju sve podatke i istovremeno uporedjuju podatke i rezultate. Kontrolu nad procesom ima samo jedan od dvaju sistema ili neizmjenično oba sistema. Pri ispadu trenutno vodećeg sistema preuzima njegovu ulogi drugi sistem. Pri tome se signalizira ispad jednog od sistema, pristupa se testiranju i lokalizaciji uzroka ispada te, eventualno, obnovi sistema. Potpuna dvojnost je vrlo skupo rješenje tolerantnih multiprocesorskih sistema. Tako se ovakve konfiguracije upotrebljavaju u veoma osobitim prilikama.



Slika 1: Sigurnost i integritet multiprocesorskih sistema je veća.

Pasivni dvojni sistemi su oni kod kojih je jedan od dvaju računarskih sistema, pasivna rezerva radnom sistemu. U slučaju ispada radnog sistema uključuje se rezervni sistem koji kontinuirano obezbjeđuje kontrolu nad procesom na osnovu medjurezultata i "svježih" podataka o procesu koje je radni sistem prije ispada periodično posredovao rezervnom sistemu. Ovakav sistem je znatno jeftiniji od potpunog dvojnog sistema ali zato zahtijeva više vremena za preklop na rezervni sistem.

Sistemi gospodar-sluga predpostavljaju dijeljenje funkcija medju dvjema računarskim sistemima. Gospodar vrši funkcije optimizacije, protokoliranja itd. Ispad gospodara u sistemu gospodar-sluga nema fatalne posljedice na tok procesa. Sluga vrši funkcije koje su neophodno potrebne za regularan tok procesa. U slučaju ispada sluge u sistemu "gospodar-sluga" preuzima gospodar sve funkcije sluge pri čemu zanemaruje neke od svojih primarnih djelatnosti. Ispad sluge identificira gospodar sam ili je to signalizirano od strane sluge. Ovaki sistemi su jeftiniji od svih dvojnih sistema i vrlo često se susreću u praktičnoj primjeni.

Distribuirani sistemi su ekonomični, veoma sposobni i hierarhično gradjeni sistemi čija je osobina takodjer i visoka sigurnost. Pri ispadu na nekom hierarhijskom nivou nesmetano rade podsistemi na nižim nivoima.

Identifikacija neregularnosti u distribuiranom sistemu vrši se na svakom nivou provjeravanjem regularnosti rada nižeg nivoa. Obnova podsistema se vrši lokalno ili na osnovu distribuirane baze podataka (i programske opreme) iz sistema na višem hierarhijskom nivou. U slučaju lokalne obnove mora imati podsistem koji je u procesu obnove na raspolaganju odgovarajuće vanjske memorijske kapacitete koji sadrže podatkovnu i programsku bazu koja se obnavlja. U slučaju centralne obnove (obnove na osnovu sistema na višem hierarhijskom nivou) uspostavlja se komunikacija izmedju dva nivoa kojim putem se prenose informacije obnove u podsistem koji se obnavlja.

3. Struktura procesa obnove

Pogledajmo primjer realizacije sistema za obnovu u uslovima realnog vremena. Sistem predstavlja kombinaciju identifikacije grešaka i obnove u materijalnoj i programskoj opremi. Sistem omogućava centralnu ili lokalnu obnovu. Identifikacija grešaka u sistemu realizirana je na osnovi višenivojskog sistema materijalnih klopki (trap). Sistem klopki moguće je realizirati putem nemaskiranog prekidnog procesorskog signala. Dakle, uvijek kada se sistem klopki aktivira izvrši se prekidni servisni program koji na bazi strategije obnove aktivira odgovarajuće akcije obnove. Sistem klopki se aktivira na osnovu sljedećih diagnostičnih elemenata sistema.

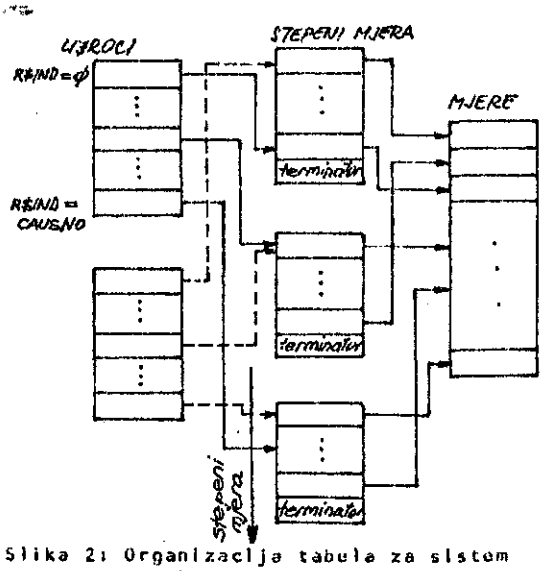
Kontrola parnosti omogućava stalnu kontrolu parnosti u sistemu pri svim pozivima memorijskih lokacija. Pri upisivanju podataka ili programa u memoriju generira se za svaki slog (ili riječ) bit parnosti kao dodatni bit koji se takodjer zapamti kao pridružena informacija za taj slog. Pri čitanju podataka ili naredbe iz memorije ponovo se generira bit parnosti čija se vrijednost uporedjuje sa onom vrijednošću koja je upisana u bit parnosti pri upisu podatka odnosno naredbe. U slučaju da se te dvije vrijednosti razlikuju aktivira se klopka parnosti.

Klopka za neprepoznate adrese omogućava stalnu kontrolu odziva memorijskih ili perifernih poziva centralne procesne jedinice za sve pozive koje ova generira. U slučaju da na neki memorijski poziv nema odziva (signal READY na vodilu sistema) aktivira se klopka neprepoznatih adresa.

Zaštita memorije pred upisom omogućava stalnu zaštitu memorijskih lokacija pred upisom. Svakoj memorijskoj lokaciji je pridružen zaštitni bit čija vrijednost odlučuje o mogućnosti upisa u pridruženu lokaciju. U slučaju pokušanja upisa u zaštićenu lokaciju aktivira se klopka memorijske zaštite.

Vremenska kontrola je veoma uspješna metoda zaštite sistema pred beskonačnim cikličnim programskim sekvencama. Sistem sadrži jedan ili više intervalnih časovnika koji paralelno osnovnoj djelatnosti sistema mjere vremenski interval u kojem se neka programska sekvenca mora zaključiti. U slučaju isteka takve vremenske kontrole aktivira se klopka vremenske kontrole.

Svaki od opisanih diagnostičnih elemenata sistema možemo smatrati za uzroke za obnovu sistema.



Slika 2: Organizacija tabela za sistem za obnovu

Važan podatak za odluku o izboru akcije za obnovu sistema je i učestanost uzroka za kojeg se akcija bira. Uopšte treba razlikovati više stepeni mjera koje se poduzimaju za neki uzrok za obnovu. Ukoliko se neki uzrok u nekom vremenskom intervalu češće pojavljuje utoliko se dže stepen mjera obnove. Isto tako mora biti moguće sniziti stepen mjera u slučaju da se neki od uzroka rjedje pojavljuje. Tako nastaje dinamična struktura sistema za obnovu u kojoj će za isti uzrok u različitim situacijama biti poduzete različite mjere obnove. Time postižemo vrlo fleksibilan sistem obnove koji na najoptimalniji način dovodi sistem do normalnog rada. Za programsku realizaciju ovakog sistema obnove moramo definirati sljedeće tabele:

Tabela uzroka je tabela koja za svaki uzrok za obnovu sadrži pokazivač (pointer) na tabelu stepeni mjera za taj uzrok.

Tabela stepeni mjera sadrži pokazivače na tabelu mjera odnosno akcija. Za svaki stepen mjere imamo jedan pokazivač na jedan od elemenata tabele akcija za obnovu sistema.

```
00004                   *
00005                   *
00006                   *
00007                   ************************************************
00008                   *                                              *
00009                   *   TRHND: TRAP HANDLER                        *
00010                   *                                              *
00011                   ************************************************
00012                   *
00013                   * FUNCTION : THIS HANDLER INCREMENTS THE DEGREE
00014                   *          OF ACTION WHEN TWO TRAPS OCCUR DURING
00015                   *          5 MINUTES. DEGREE OF ACTION IS SET TO
00016                   *          THE LOWEST VALUE IF THERE IS A PERIOD
00017                   *          OF 9 MINUTES WITHOUT TRAPS. ACCORDING
00018                   *          TO THE CAUSE OF THE TRAP AND ACCORDING
00019                   *          TO CURRENT DEGREE THE HANDLER PICKS UP
00020                   *          APPROPRIATE ACTION AND STARTS IT.
00021                   *
00022                   *
00023                   * ENTRY CONDITIONS: R$IND - CODE FOR TRAP CAUSE
00024                   *                   COUNTT - STOPWATCH
00025                   *
00026                   * EXIT CONDITIONS: RECOVERY ACTION IS STARTED
00027                   *
00028                   *


00030                           XDEF    TRHND,UPDATE

00032                           XREF    ANY:TEMP3,TEMP4,TEMP5,TRY,INDIK,R$IND,COUNT
00033                           XREF    ANY:DEVET,CAUSNO
00034                           XREF    ANY:TABIND,STRPTR,MOVSTR


00036P 0000                     PSCT

00038P 0000 FE 0000   A TRHND   LDX     COUNTT  IF STOPWATCH=0 THEN GOTO
00039P 0003 8C 0000   A         CPX     #0      SKEL1 ELSE CONT.
00040P 0006 27 4E 0053          BEQ     SKEL1
00041P 0008 7A 0000   A         DEC     TRY     DECREMENT TRY COUNTER
00042P 000B 26 60 008D          BNE     SKEL4   IF TRY<>0 THEN GOTO SKEL4
00043P 000D CE 0000   A         LDX     #TABIND
00044P 0010 B6 0000   A         LDAA    R$IND   SAVE TRAP CAUSE
00045P 0013 B7 0000   A         STAA    INDIK
00046P 0016 7D 0000   A SKEL9   TST     INDIK   MOVE POINTER IN TABIND
00047P 0019 27 07 0022          BEQ     SKEL8   TABLE ACCORDING TO CAUSE
00048P 001B 08                  INX             OF TRAP
00049P 001C 08                  INX
00050P 001D 7A 0000   A         DEC     INDIK
00051P 0020 20 F4 0016          BRA     SKEL9

00053P 0022 FF 0000   A SKEL8   STX     TEMP3
00054P 0025 EE 00     A         LDX     0,X
00055P 0027 FF 0000   A         STX     TEMP4
00056P 002A EE 02     A         LDX     2,X
00057P 002C 8C FFFF   A         CPX     #$FFFF  IF TERMINATOR SKIP INCREMENT
00058P 002F 27 15 0046          BEQ     SKEL10
```

Tabela mjera(akcija) je tabela koja sadrži sve mjere obnove koje u sistemu mogu biti poduzete poslije identificiranja nekog uzroka za obnovu. Element tabele mjera je pokazivač na podprogram koji ima kontrolu nad procesom obnove za odgovarajući uzrok i stepen mjere. Elementi tabele mjera mogu biti zajednički za različite uzroke i stepene mjera. Priloženi program predstavlja osnovu sistema za start i obnovu. Klopke koje se aktiviraju, kada dođe do neke greške u sistemu postave odgovarajuću vrijednost indikatora R$IND uz pomoć klopkama pridruženih servisnih programa. Indikator R$IND određuje uzrok te element u tabeli uzroka. Vrijednost indikatora mora biti u granicama od 1 do broja uzroka grešaka u sistemu koje je točno određeno (parametar CAUSNO). R$IND je ulazni parametar programa TRHND. Servisiranje klopki znači određivanje indikatora R$IND i skok u program TRHND. Postupak programa TRHND je sljedeći:

Ako se u intervalu od 9 min aktiviraju tri klopke povećava se stepen mjere za uzrok određen indikatorom R$IND. Zatim se aktivira mjera odgovarajućeg stepena. Ako se u intervalu od 9 min nije aktivirala niti jedna klopka pristupa se sniženju stepeni mjera za sva uzroka na minimum kada se aktivira prva sljedeća klopka istovremeno se aktivira mjera najnižeg stepena za odgovarajući uzrok (R$IND). Za pravilno djelovanje programa potrebno je osigurati periodično aktiviranje programa UPDATE svake sekunde, time se obezbjeđuje mjerenje vremena. Program UPDATE je moguće aktivirati periodičnim prekidnim (interrupt) vremenskim signalom.

4. Zaključak

Sistemi sa tolerancijom grešaka, a time u vezi i sistemske obnova u uslovima realnog vremena, dobivaju izuzetno na važnosti u savremenim uslovima masovne automatizacije industrijskih, informacijskih, prometnih, energetskih i drugih objekata. Masovna primjena mikroprocesora za ovakve aplikacije zahtjeva posebna konstrukcijska rješenja koja obezbjeđuju kontinuitet i sigurnost procesa. Metode obnove sistema imaju pri tome posebno mjesto posebno u multiprocesorskim konfiguracijama te distribuiranim sistemima.

Literatura:

1. A. Avižienis: Fault-Tolerance: The Survival Attribute of Digital Systems, Proceedings of IEEE, October 1978, vol.66, No. 10

2. D.Novak: Sistemi z več procesorji, Informatica, 3, 1980, Ljubljana.

```
00059P 0031 FE 0000  A        LDX    TEMP4
00060P 0034 08               INX
00061P 0035 08               INX
00062P 0036 FF 0000  A        STX    TEMP4      INCREMENT POINTER
00063P 0039 FE 0000  A        LDX    TEMP3      TO ACTION POINTER TABLE
00064P 003C B6 0000  A        LDAA   TEMP4      FOR THIS CAUSE
00065P 003F F6 0001  A        LDAB   TEMP4+1
00066P 0042 A7 00    A        STAA   0,X
00067P 0044 E7 01    A        STAB   1,X

00069P 0046 CE 0000  A SKEL10 LDX    #DEVET     START STOPWATCH
00070P 0049 FF 0000  A        STX    COUNTT
00071P 004C 86 02    A        LDAA   #2         INIT TRY COUNTER
00072P 004E B7 0000  A        STAA   TRY
00073P 0051 20 1A 006D       BRA    SKEL4

00075P 0053 CE 0000  A SKEL1  LDX    #DEVET     START STOPWATCH
00076P 0056 FF 0000  A        STX    COUNTT
00077P 0059 86 02    A        LDAA   #2
00078P 005B B7 0000  A        STAA   TRY        INIT TRY COUNTER
00079P 005E CE 0097  P        IDX    #TABSTR    COPY TABSTR TABLE INTO
00080P 0061 FF 0000  A        STX    STRPTR     TABIND TABLE
00081P 0064 CE 0000  A        LDX    #TABIND
00082P 0067 C6 00    A        LDAB   #CAUSNO
00083P 0069 58               ASLB
00084P 006A BD 0002  A        JSR    MOVSTR+2 (MOV1)

00086P 006D CE 0000  A SKEL4  LDX    #TABIND
00087P 0070 7D 0000  A SKEL5  TST    R$IND      FIND THE TABLE OF ACTIONS
00088P 0073 27 07 007C       BEQ    SKEL11     GIVEN CAUSE
00089P 0075 08               INX
00090P 0076 09               INX
00091P 0077 7A 0000  A        DEC    R$IND
00092P 007A 20 F4 0070       BRA    SKEL5

00094P 007C EE 00    A SKEL11 LDX    0,X        START ACTION
00095P 007E EE 00    A        LDX    0,X
00096P 0080 EE 00    A        LDX    0,X
00097P 0082 6E 00    A        JMP    0,X



00099                *
00100                ************************************************
00101                *                                              *
00102                *  UPDATE: DECREMENT STOPWATCH                 *
00103                *                                              *
00104                ************************************************
00105                *
00106                * FUNCTION: THIS SUBROUTINE DECREMENTS THE
00107                *           STOPWATCH IF IT IS RUNNING
00108                *
00109                * ENTRY COND  : COUNTT - VALUE OF STOPWATCH
00110                *
00111                * EXIT COND   : COUNTT - NEW VALUE OF STOPWATCH
00112                *
00113P 0084 FF 0000  A UPDATE STX    TEMP5      SAVE INDEX REGISTER
00114P 0087 FE 0000  A        LDX    COUNTT
```



```
00115P 008A 8C 0000  A        CPX    #0         IF COUNTT=0 THEN GOTO UPD1
00116P 008D 27 04 0093       BEQ    UPD1       ELSE DECREMENT COUNTT
00117P 008F 09               DEX
00118P 0090 FF 0000  A        STX    COUNTT
00119P 0093 FE 0000  A UPD1   LDX    TEMP5
00120P 0096 39               RTS



00122P 0097    0000  A TABSTR FDB    0          TABLE OF POINTERS TO
00123P 0099    0000  A        FDB    0          TOP OF ACTION POINTER TABLES
00124P 009B    0000  A        FDB    0
00125P 009D    0000  A        FDB    0
00126                *
00127                ************************************************
00128                *
00129                *  THE STRUCTURE OF TABSTR SHOULD BE THE FOLLOWING:
00130                *
00131                * TABSTR FDB  TABAC0
00132                *        FDB  TABAC1
00133                *        FDB  TABAC2
00134                *          .
00135                *          .
00136                *          .
00137                *        FDB  TABACN        N=CAUSENO
00138                * TABAC0 FDB  ACTN4
00139                *        FDB  ACTN2         TABLE OF POINTER TO ACTIONS
00140                *          .                FOR BD$FF=1
00141                *          .
00142                *        FDB  $FFFF
00143                * TABAC1 FDB  ACTN1         TABLE OF POINTERS TO ACTIONS
00144                *        FDB  ACTN0         FOR CAUSE 1 (R$IND=1)
00145                *          .
00146                *          .
00147                *        FDB  $FFFF
00148                *          .
00149                *          .
00150                *          .
00151                * TABACN FDB  ACTNX
00152                *          .
00153                *          .
00154                *        FDB  $FFFF
00155                *
00156                      END
TOTAL ERRORS 00000
```

Prilog: program TRHND: Program TRHND ima sljedeću ulogu u sistemu za obnovu: Na osnovu vrijednosti indikatora uzroka R$IND i tabele stepeni mjera za taj uzrok odrediti mjeru (akciju) koja ima biti poduzeta u cilju obnove sistema.

INFORMATICA 4/1981

MIKROPROCESORSKO VODENJE SENZORSKEGA SISTEMA ZA ROBOTSKO VARJENJE

S. PREŠEREN
I. OZIMEK
M. ŠPEGEL

UDK: 681.3:621.791

INŠTITUT JOŽEF STEFAN, LJUBLJANA

Članek opisuje izdelavo laboratorijskega prototipa robotskega sistema za avtomatsko varjenje, ki omogoča avtomatsko programiranje varilnega avtomata na podlagi baze znanja o trenutnem stanju sistema, ki se zbira s senzorji, in baze podatkov o kontrolnem algoritmu. Povdarjena je modularnost robotskega senzorskega sistema in opisana je funkcija posameznih modulov. Posebna pozornost je posvečena kontrolnem algoritmu, ki ga izvaja mikroračunalniški modul z mikroračunalnikom Z80.

MICROCOMPUTER CONTROL OF SENSING SYSTEM FOR A WELDING ROBOT

The paper describes the development of a laboratory prototype of a robotic system for automatic arc welding. It is designed for automatic programming of robot actions on the bases of knowledge base about the present state of the system, which is obtained through sensors, and on data base of control algorithm. Modularity of the system is stressed out and the function of different modules is described. The microcomputer modul and the control algorithm are described in detail.

1. UVOD

V zadnjih letih je prišlo do potrebe po večji avtomatizaciji obločnega in točkastega varjenja. Tako so nastali sodobni sistemi za avtomatsko varjenje in varilni roboti, ki popolnoma samostojno izvajajo proces varjenja po v naprej predpisani tirnici. Ti roboti pa se niso sposobni prilagajati dejanski tirnici, ki ni vedno enaka predpisani. Zaradi potrebe po avtomatskem sledenju reže, ki jo robot vari, smo na Inštitutu J. Stefan razvili mehansko tipalo z dvema prostostnima stopnjama (1). Avtomat za varjenje ima tri prostostne stopnje, ki pozicionirajo varilno šobo glede na režo. Tipalo je vpeto približno tri centimetre pred varilno šobo v smeri varjenja. Robot vleče tipalno iglo po reži. Z odklanjanjem tipalne igle v smer reže, se registrira tirnica reže. Ko varilna šoba pripotuje do točke, kjer je tipalo izmerilo odklon, se izvede pozicioniranje šobe.

2. MODULI ROBOTSKEGA SISTEMA

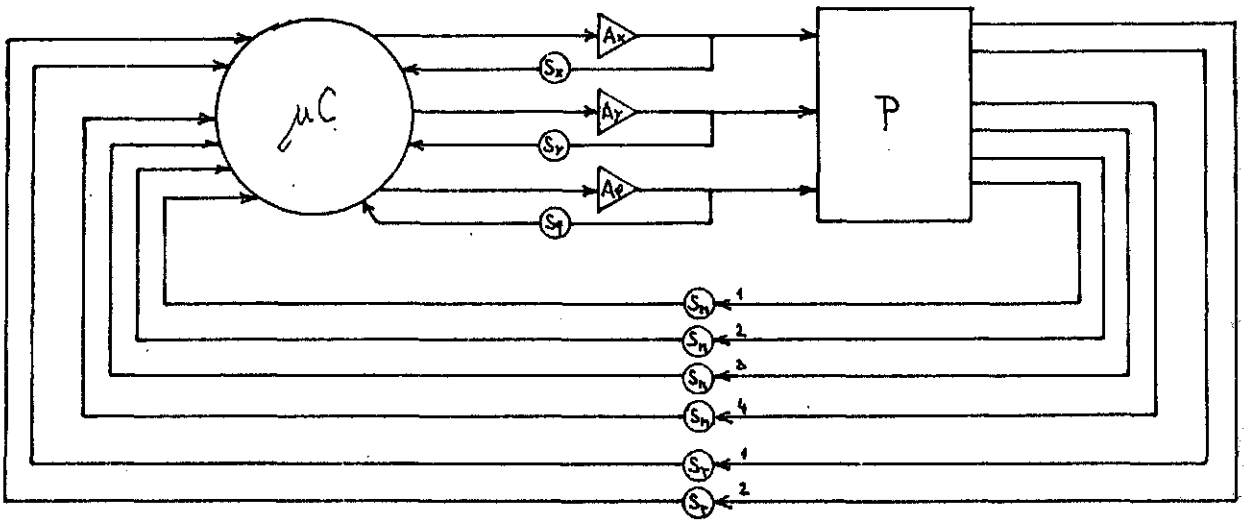
Avtomatsko programiranje robotskega sistema za varjenje smo izvedli v obliki multivariabilnega senzorskega sistema (2). Senzorski sistem je sestavljen iz senzorskega modula S, motornega modula A in mikroprocesorja μC, ki kontrolira proces pozicioniranja P. (Slika 1).

Senzorski modul $S = \{S_x, S_y, S_t, S_{m1}, S_{m2}, S_{m3}, S_{m4}, S_{t1}, S_{t2}\}$ je sestavljen iz tipala z dvema notranjima prostostnima stopnjama, od katerih ima vsaka po šest senzorjev za identifikacijo položaja tipalne igle (1). Poleg tega sestavljajo senzorski modul še štiri mikro stikala S_{m1}, S_{m2}, S_{m3} in S_{m4}, ki varujejo sistem pred odklonom držala varilne šobe v ekstremni položaj v $\pm x$ in $\pm y$ smeri. Na motorjih imamo senzorje obratov motorjev S_x, S_y in S_t.

Motorski modul $A = \{A_x, A_y, A_t\}$ sestavljajo trije DC motorji, ki služijo za pozicioniranje varilne šobe glede na varjenec. Motor A_x in A_y premikata varilno šobo v ravnini xy, motor A_t pa vrti podnožje, kjer je pritrjen varjenec.

Proces P je sestavljen iz dveh delov:
- odmikanja tipalne igle v smer tirnice in
- pozicioniranja varilne šobe s pomočjo motorjev A.

V kontrolni modul je vključen mikroračunalnik



Slika 1.: Moduli robotskega sistema za varjenje.

Z80 z 2K besedami EPROMa in 1K besedami RAMa ter kontrolni algoritem.

3. KONTROLNI ALGORITEM

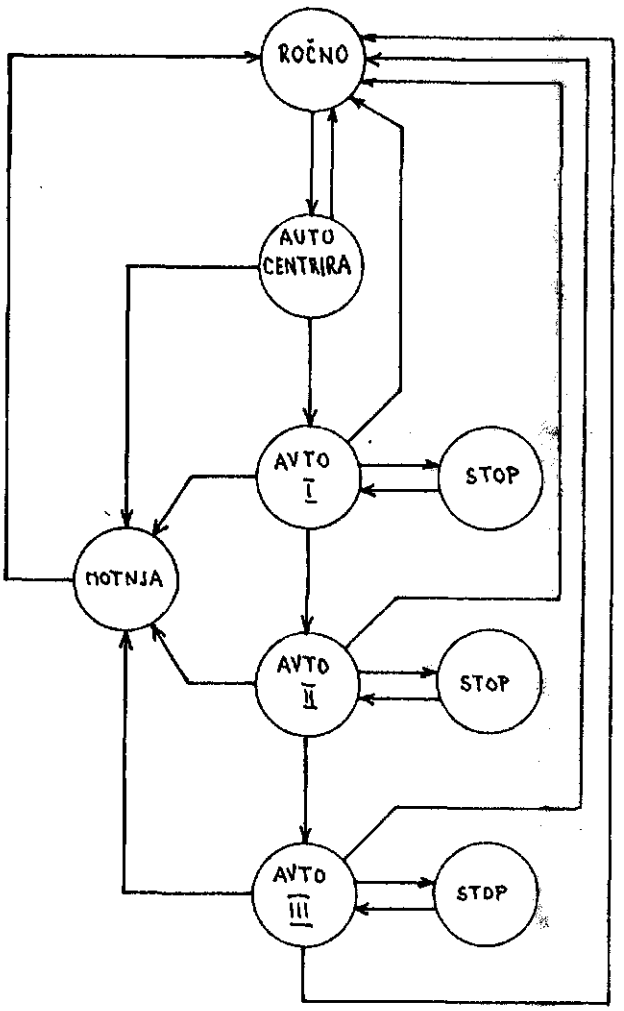
Kontrolni algoritem vodi avtomatski varilni sistem. Na podlagi baze znanja o postopku tipanja in varjenja, ter informacij o trenutnem stanju, v katerem se nahaja varilni sistem, se izbere ustrezna faza kontrolnega algoritma. Kontrolni algoritem (Slika 2) je razdeljen na dva režima delovanja:

- ročni in
- avtomatski.

Ročni režim omogoča ročno vodenje varilne šobe s tipalom. V tem režimu je tipalo izključeno. Avtomatski režim pa omogoča detekcijo tirnice reže s tipalom in ustrezno avtomatsko pozicioniranje varilne šobe.

Iz grafa kontrolnega algoritma je razvidno, da je iz vsake stopnje avtomatskega režima možen skok v ročni režim. Prav tako preskoči sistem v ročni režim, če pride pri gibanju vpenjala z varilno šobo do nepredvidenih premikov preko limitnih stikal. V tem primeru skoči sistem najprej v stanje "motnja" in to stanje signalizira na komandni plošči s prižigom opozorilne lučke.

Avtomatski režim ima štiri stopnje: centriranje in tri stopnje za različne faze delovanja avtomatskega režima. V fazi "centriranje" se tipalo, ki je že nameščeno v reži, pozicionira v centralni položaj, tako, da je pri poznejšem sledenju možen maksimalen odmik tipalne igle v $\pm x$ in $\pm y$ smer. Ko je faza centriranja zaključena, je sistem pripravljen za sledenje.



Slika 2.: Graf kontrolnega algoritma.

Prva stopnja v fazi avtomatskega delovanja je namenjena zapisovanju začetka trajektorije v spomin. Ta stopnja je potrebna, ker je varilna šoba približno 3cm za tipalom. Pozicioniranje šobe se začne šele, ko tipalo prepotuje to razdaljo in je varilna šoba prispela do začetka reže. Tu se začne druga stopnja avtomatske faze delovanja, ki zajema čitanje merskih podatkov iz spomina in zapisovanje novih lokacij trajektorije. Tretja stopnja nastopi, ko je tipalo že izven reže, s preostalimi podatki iz spomina pa se še izvaja pozicioniranje šobe. Iz vsake stopnje v fazi delovanja je možen skok v čakalno rutino in povratek v isti del faze avtomatskega delovanja.

Odločitev za prehod iz enega stanja v drugo v grafu kontrolnega algoritma se izvrši na podlagi:

- ukazov s pomočjo komandnih stikal na komandni plošči,
- baze znanja o trenutnem stanju sistema, ki se zbira s senzorji (tipalo, limitna mikrostikala) in
- baze podatkov, ki vsebuje možne prehode v grafu kontrolnega algoritma in delovnega programa.

Delovni program ima splošni del, ki se opravlja pri vsakem izvajanju programa, in procesiranje trenutnega stanja, ki je različno za vsako fazo v grafu kontrolnega algoritma. Splošni del vključuje sledeče rutine (Slika 3)

- tipalo: prebere položaj tipalne igle v x in y smeri,
- gumbi: prebere stanje komandnih stikal na komandni plošči,
- limite: prebere eventualne limitne vrednosti iz mikrostikal v primeru, če je odmik vpenjala s šobo izven dovoljenih meja,
- lučke: prižge ustrezne signalne lučke na komandni plošči,
- motorji: požene motorje z zaželjeno hitrostjo
- čas: skrbi za sinhronizacijo delovanja celotnega sistema.

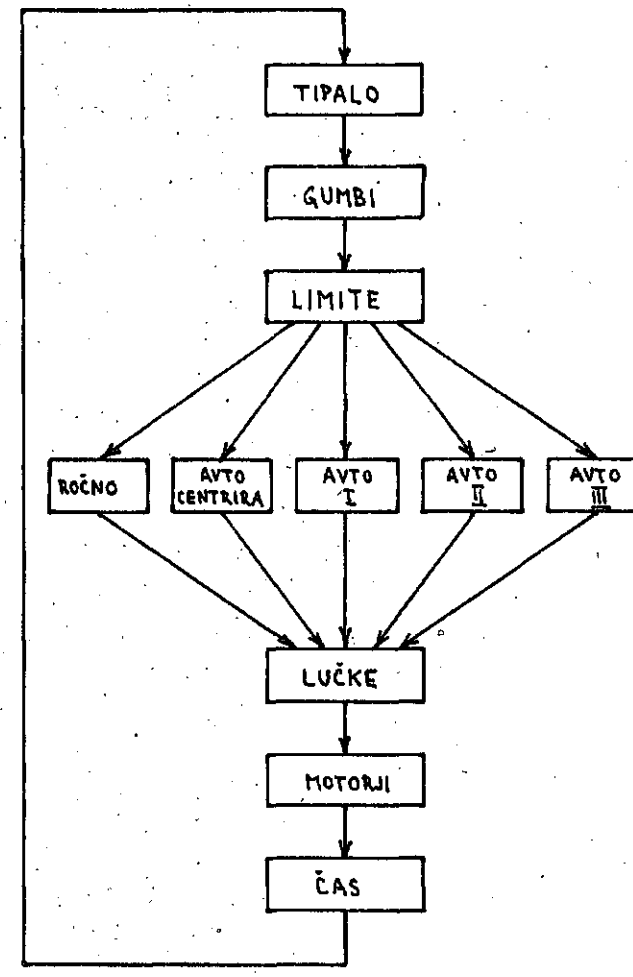
Frekvenca ponavljanja programa je 10 Hz.

4. RAZVOJNO OKOLJE

Delovni program je napisan v programskem jeziku CORAL, ki omogoča enostavno vnašanje asemblerskih inštrukcij za Intelov mikroračunalnik med visiko programski jezik. Prevajanje CORALa v asembler je izvedeno na sistemu INTELEC.

5. ZAKLJUČEK

Podali smo strukturo in glavne specifikacije



Slika 3.: Blok diagram delovnega programa.

robotskega sistema za avtomatsko varjenje, ki je opremljeno z digitalnim tipalom. Tipalo lahko uporabljamo za identifikacijo nove neznane trajektorije ali pa za merjenje odstopanj dejanske trajektorije reže od idealne krivulje. Pri definiranju kontrolnega algoritma in razvoju delovnega programa smo pazili na modularnost, kar nam omogoča fleksibilno spreminjanje in dopolnjevanje obstoječega sistema. V načrtu imamo dve izpopolnitvi: tabelarično regulacijo hitrosti motorjev za pozicioniranje prijemala z varilno šobo in vključitev ene aktivne notranje prostostne stopnje tipala, kar bo omogočilo avtomatsko iskanje začetka reže.

6. LITERATURA

(1) S. Prešern et al.: Razvoj digitalnega tipala in mikroračunalniškega kontrolnega sistema za obločno varjenje, Informatica 3, 1980.

(2) S. Prešern, Mathematical and methodical aspects of computerused measurement architecture, Proc. Symp. on Computerized Measurement, Dubrovnik, 1981.

INFORMATICA 4/1981

PRIMJENA TRANSPOZICIONE METRIKE ZA PRORAČUN OPTIMALNOG BALANSA

JOVAN LONČAR

VIŠA ZRAKOPLOVNA ŠKOLA – ZAGREB

UDK: 681.3:51

U radu se u prostor permutacija uvodi transpoziciona metrika, koja se koristi za nalaženje optimalne vrijednosti debalansa.

APLICATION OF TRANSPOSITION METRIC TO CALCULATION OF THE OPTIMAL BALANCE: In space of permutation transposition metric is introduced and used for calculation of the optimal balance.

UVOD

U [1] [2] u prostor permutacija $\mathcal{P}$ uvedene su lančana odnosno inverziona metrika i pomoću njih računate su vrijednosti debalansa. Medjutim pokazuje se potreba da se i dalje uvode nove metrike koje omogućavaju na jednostavniji način formuliranje raznih probleme i ograničenja što se pojavljuju pri nalaženju ekstrema funkcionala na skupu permutacija [3]. Zato se u ovom radu uvodi transpoziciona metrika u prostor permutacija $\mathcal{P}$.

TRANSPOZICIONA METRIKA

Udaljenost u transpozicionoj metrici izmedju dvije permutacije p_1 i p_2 od n-simbola definira se kao najmanji mogući broj transpozicija nužnih za prelaz od p_1 u p_2. Nalaženje udaljenosti izmedju dviju permutacija od n-simbola u transpozicionoj metrici je dosta kompliciran problem i autoru nije poznat algoritam za rješavanje ovog problema. Medjutim, nas to ne zabrinjava, jer kod traženja minimuma funkcionala mi ne formiramo permutaciju koja leži na lopti već u kugli zadanog radiusa. Budući se od proizvoljne $p_1 \in \mathcal{P}$ od n-simbola može prijeći u proizvoljnu $p_2 \in \mathcal{P}$ za (n-1) transpozicija, to je $\max d(p_i, p_j) \leq n-1$. Prednje se može dokazati pomoću matematičke indukcije. Naime, ako imamo permutacije od dva simbola, onda je očito da iz $p_1=(1,2)$ možemo prijeći u $p_2=(2,1)$ za (n-1) = 1 transpoziciju. Recimo da je tvrdnja istinita za n=k tj. da iz proizvoljne permutacije $p_i \in \mathcal{P}$ od k-simbola možemo prijeći u drugu $p_j \in \mathcal{P}$, takodjer od k-simbola za k-1 transpoziciju. Promatrajmo sada dvije permutacije od (k+1) simbola $p_1=(i_1 i_2 \ldots i_{k+1})$, $p_2=(j_1, j_2 \ldots j_{k+1})$. Ako je $i_{k+1}=j_{k+1}$ onda njih nije potrebno transponirati, pa se prijelaz iz p_1 u p_2 može izvršiti za (k-1) transpozicija. Ako je $i_{k+1} \neq j_{k+1}$ dovoljno je dodati samo jednu transpoziciju. Znači iz p_1 u p_2 možemo prijeći za k-transpozicija. Provjerimo aksiome udaljenosti za transpozicionu metriku.

A.1. $d(p_1,p_2) \geq o$.
Ovaj aksiom je toliko očit da se nema što provjeravati.

A.2 $d(p,p) = o$

A.3 $d(p_1,p_2) = o \Rightarrow p_1=p_2$
Drugi i treći aksiom su ispunjeni jer odsutnost transpozicija označava nepromjenljivost permutacija.

A.4. $d(p_1,p_2) = d(p_2,p_1)$.
Aksiom je ispunjen, jer koliko je potrebne transpozicija za prijelaz iz p_1 u p_2 toliko je potrebno i za prijelaz iz p_2 u p_1.

A.5. $d(p_1,p_3) \leq d(p_1,p_2)+d(p_2,p_3)$
Ovaj aksiom je takodjer ispunjen. Naime, $d(p_1,p_2)+d(p_2,p_3)$ možemo smatrati kao broj transpozicija pomoću kojih od p_1 možemo prijeći u p_3. Kod toga p_2 smatramo samo prolaznom etapom. Medjutim, $d(p_1,p_3)$ je najmanji broj transpozicija koji osigurava prijelaz od p_1 u p_3. Slijedi $d(p_1,p_3) \leq d(p_1,p_2)+d(p_2,p_3)$.

Kao što vidimo svi su aksiomi zadovoljeni. Nama treba postupak za generiranje $p \in \pi$ koja leži u kugli radiusa R s centrom u permutaciji p_0. Kod toga R ispunjava uvjet $1 \leq R \leq n-1$. To se može realizirati pomoću slijedećeg algoritma.

ALGORITAM

Prvi korak: Stavi k=1, a $p=p_0$.

Drugi korak: Iz segmenta $(1, n]$ izaberemo dva slučajna broja „i", „j". U permutaciji p zamjenimo mjesta znakova i sa j te dobivenu permutaciju označimo sa p_0.

Treći korak: Stavi k=k+1. Ako je $k \leq R$, vrati se na korak 2, inače se zaustavi.

Na osnovu prednjeg algoritma sačinjen je program za elektronski računar. U svrhu usporedbe dobivenih rezultata pomoću transpozicione metrike s rezultatima dobivenim pomoću lančane i inverzione metrike uzeti su isti podaci za statičke momente iz tabele 1. Za $p_0 \in \pi$ uzeli smo onu permutaciju na kojoj je funkcional

$$f(i_1 i_2 \ldots i_n) = \left[\left(\sum_{i=1}^{n} M_{ik} \cos\frac{2\pi k}{n}\right)^2 + \sum_{i=1}^{n} M_{ik} \sin\frac{2\pi k}{n}\right)^2\Big]^{\frac{1}{2}} \quad (*)$$

imao minimalnu vrijednost, kada su eksperimenti vršeni metodom MONTE-KARLO [1].

U okolini R=5 s centrom u navedenoj p_0, birano je 100 permutacija $p \in \pi$ i računate vrijednosti funkcionala - debalansa danog sa (*).

Rezultate vidimo na slici 1. Dobiveno je:
Minimum debalansa = 36,17
Maksimum debalansa = 1824,88
Matematičko očekivanje 668,58
Raspored lopatica koji odgovara minimumu debalansa vidimo u tabeli 2, a raspored koji odgovara maksimumu u tabeli 3.

ZAKLJUČAK

Vidimo da su za 100 pokusa u transpozicionoj metrici dobiveni bolji rezultati nego što su oni dobiveni metodom MONTE-KARLO, kao i oni u lančanoj i inverzionoj metrici [1], [2]. To nas upućuje da treba i dalje nastaviti s uvodjenjem novih metrika u prostor permutacija.

TABELA 1

Broj lopatice	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Statički moment	0	109	-309	-31	-411	-179	420	-145	-301	61	181	231	-162	-341	-281	175
Broj lopatice	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Statički moment	49	0	-69	-31	-46	176	-511	-412	-317	470	79	13	-142	-35	-152	-88
Broj lopatice	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Statički moment	13	-625	- 6	21	-259	- 53	17	11	-259	15	91	-121	475	- 9	-191	81
Broj lopatice	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Statički moment	15	42	3	-11	-249	22	5	-11	- 5	- 2	17	-113	17	-17	-39	408
Broj lopatice	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
Statički moment	5	25	15	41	5	-26	22	-11	27	-23	-30	75	49	31	-457	-45
Broj lopatice	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
Statički moment	-57	- 9	9	0	-11	-25	- 7	31	65	31	44	21	61	21	51	31

SLIKA 1

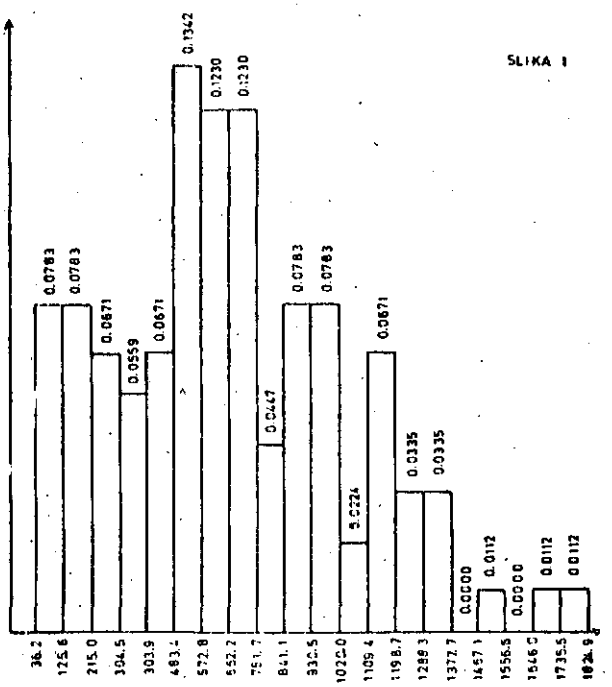


TABELA 2

Broj mjesta	1	2	3	4	5	6	7	8	9	1o	11	12	13	14	15	16
Broj lopatice	25	73	27	37	13	54	79	8o	45	83	6	17	4	87	88	65
Statički moment	-317	27	79	-259	-162	22	-457	-45	91	9	-179	49	-31	- 7	31	-11
Broj mjesta	17	18	19	2o	21	22	23	24	25	26	27	28	29	3o	31	32
Broj lopatice	89	23	55	49	51	19	96	28	11	68	77	32	12	72	2	82
Statički moment	65	-511	5	15	3	-69	31	13	181	41	49	-88	231	-11	1o9	- 9
Broj mjesta	33	34	35	36	37	38	39	4o	41	42	43	44	45	46	47	48
Broj lopatice	78	84	93	75	56	3o	39	6o	41	52	71	6	38	81	91	3
Statički moment	31	0	61	-58	-11	35	17	-113	-259	-11	22	-145	-53	-57	44	-309
Broj mjesta	49	5o	51	52	53	54	55	56	57	58	59	6o	61	62	63	64
Broj lopatice	59	9o	67	33	69	35	57	1	15	26	2o	9	33	1o	31	14
Statički moment	17	31	15	-249	5	-6	5	0	-281	47o	-31	-3o1	13	61	-152	-341
Broj mjesta	65	66	67	68	69	7o	71	72	73	74	75	76	77	78	79	8o
Broj lopatice	63	36	22	34	44	46	5o	92	64	58	16	48	7	21	5	42
Statički moment	-39	21	176	-625	-121	- 9	42	21	488	- 2	175	81	42o	-46	-411	15
Broj mjesta	81	82	83	84	85	86	87	88	89	9o	91	92	93	94	95	96
Broj lopatice	61	86	29	24	4o	45	66	18	62	65	95	47	7o	74	94	76
Statički moment	17	-25	-142	-412	11	475	25	0	-17	5	51	-191	-26	-23	21	75

TABELA 3

Broj mjesta	1	2	3	4	5	6	7	8	9	1o	11	12	13	14	15	16
Broj lopatice	25	73	16	37	13	54	79	80	94	83	86	51	4	87	88	28
Statički moment	-317	27	175	-259	-162	22	-457	-45	21	9	-25	3	-31	- 7	31	15
Broj mjesta	17	18	19	2o	21	22	23	24	25	26	27	28	29	30	31	32
Broj lopatice	89	23	55	92	34	3	96	85	11	53	77	32	12	72	59	82
Statički moment	65	-511	5	21	-625	-3o9	31	-11	181	-249	49	-88	231	-11	17	- 9
Broj mjesta	33	34	35	36	37	38	39	4o	41	42	43	44	45	46	47	48
Broj lopatice	78	84	93	75	56	3o	39	6o	63	52	71	8	38	81	91	19
Statički moment	31	0	61	-58	-11	35	17	-113	-39	-11	22	-145	-53	-57	44	-69
Broj mjesta	49	5o	51	52	53	54	55	56	57	58	59	6o	61	62	63	64
Broj lopatice	66	24	68	67	69	35	57	1	15	26	2o	9	33	1o	3	14
Statički moment	25	-412	41	15	5	- 6	5	0	-281	47o	-31	-3o1	13	61	1o9	-341
Broj mjesta	65	66	67	68	69	7o	71	72	73	74	75	76	77	78	79	8o
Broj lopatice	41	36	22	17	44	46	5o	49	64	58	27	48	7	21	5	42
Statički moment	-259	21	176	49	-121	- 9	42	15	488	- 2	79	81	42o	- 46	-411	15
Broj mjesta	81	82	83	84	85	86	87	88	89	9o	91	92	93	94	93	96
Broj lopatice	61	6	29	9o	4o	45	31	18	62	65	95	47	7o	74	45	76
Statički moment	17	-179	-142	31	11	475	-152	0	-17	5	51	-191	- 26	- 23	91	75

LITERATURA:

1. J.Lončar: O jednoj metodi proračuna optimalnog balansa. Informatica broj 3 Vol 5, Ljubljana, 1981.
2. J.Lončar: Primjena inverzione metrike kod optimalnog balansiranja. Informatica broj 3 Vol 5 Ljubljana 1981.
3. Kaspšickaja M.F. i drugi. Ob odnompodhode k rešeniju zadać razmešenija. Kibernetika 1974 N° 5

INFORMATICA 4/1981

PLUS CA CHANGE, PLUS C'EST LA MÊME CHOSE

VITOMIR SMOLEJ

UDK: 681.3.06

HIGH MANAGEMENT SCHOOL IN KRANJ,
UNIVERSITY OF MARIBOR

ABSTRACT: by desk-top analysis of program specifications for a medium-size software project (about 65.000 lines of COBOL code to be produced in a year) it has been shown that a significant amount of programming effort can be saved by using pre-fabricated prototype programs and sensibly selected primitives (for instance I/O modules). The prerequisites for success in the approach proposed are discussed.

BOLJ KO SE SPREMINJA, BOLJ SI JE PODOBNO: analiza specifikacij za programe v srednje velikem programskem projektu (okoli 65 tisoč vrstic COBOLske kode v enem letu) je pokazala, da si lahko olajšamo posel v veliki meri z uporabo prefabriciranih prototipnih programov in pametno izbranih primitivnih modulov (recimo moduli za V/I operacije). Obrazloženi so pogoji, ki so potrebni za uspešno uvedbo novega pristopa v proizvodnjo programske opreme.

INTRODUCTION

A group of programmers has obtained a large contract for coding programs for commercial applications (stock control plus accounts receivable/payable).The code is to be produced according to programming specifications of the future user of the software. The specifications are very detailed. Let me quote:

> SECTION READ-INP-S.
> Move 0 to EOF-INP and read a record from input file. In the case that end-of-file has been read, place 1 into EOF-INP and leave the section. If, however, no eof has been read, test whether the sort keys (PARTNO, USERNO) has changed since the last read. If they are greater, place 1 to EOL-INP. If they are equal, place 0 into EOL-INP. If, however, the sort keys last read are smaller than the previous values, display "wrong sort before this program" and end the program abnormally..."

Every specification has a VTOC attached, which displays the tree of the sections used. Some of the sections are standard (for instance checking the modulo-10 digit, checking the correctness of the date, printing the error summary and printing the basic statistics of the records read or written). A large amount of data division structures is available in the development library: all the record descriptions are already in the library. Also, all the selects for files are available.

All in all, 41 program specifications, authorized by the future user of the software, have been first discussed for dubious points and then used for designing and coding the programs.

DEJA VU EFFECT

After browsing through specifications, an unpleasant feeling of deja-vu has occured to the author of this article. Namely, forgetting that the program A had the section READ-A-S (with sort keys being PARTA AND personA), while program B dealt with READ-B-S (with PARTB and PERSONB from some other file), quite a lot seemed to be the same in both programs. To check where the differences lie and how big they are, the following strategy has been accepted:

1. draw the top-most structure of every program (if possible) in the form of syntax graph
2. check for most frequent types of trees that arise
3. for every program find the most similar tree - the prototype program
4. count the number of sections (using VTOC) which can be attributed to the prototype program and number of lowest-level sections, which deal with the input/output of data.

PROTOTYPES

According to ideas of the top-down programming (see for instance Dijkstra, Notes on structured programming) every program can at

first be thought of as a one-verb program. For instance: a program to do payrolls: would at first look very simple and powerfull:

```
do payroll
```

As unfortunately we do not yet have such a combination of hardware and software: which would understand this verb: we start to detail it using sequence: selection and iteration as the basic constructs of structured programming.

The tree bellow (fig.1) shows in what prototype categories the programs could be divided on the basis of program specifications. 8 prototypes have been found for 41 programs to be developed. It must be pointed out: that not less than five upper-most sections in every program have been found to coincide with a given prototype (see Table 1). If we take into account the number of possible structures built of six blocks: a reduction in number of prototype programs is very large indeed.

It is no surprise that out of 41 programs about 75% can be developed from a prototype program for sequential input. Its pseudo code can be written as follows:

```
do something at the beginning;
read file;
while not eof
  do something with the record read
  read file;
do something at the end;
stop
```

The other two upper-most branches deal with problems: which do not have a sequential nature. The first is the problem of sorting: the second one the problem of matching two files on the basis of record-by-record key comparison. Note that this two branches are rather thin: (sort having 9: matching problem only two descendants in the set of programs to be written).

Further differentiation of prototypes is made on the basis of developing the instruction

```
do something with the record read
```

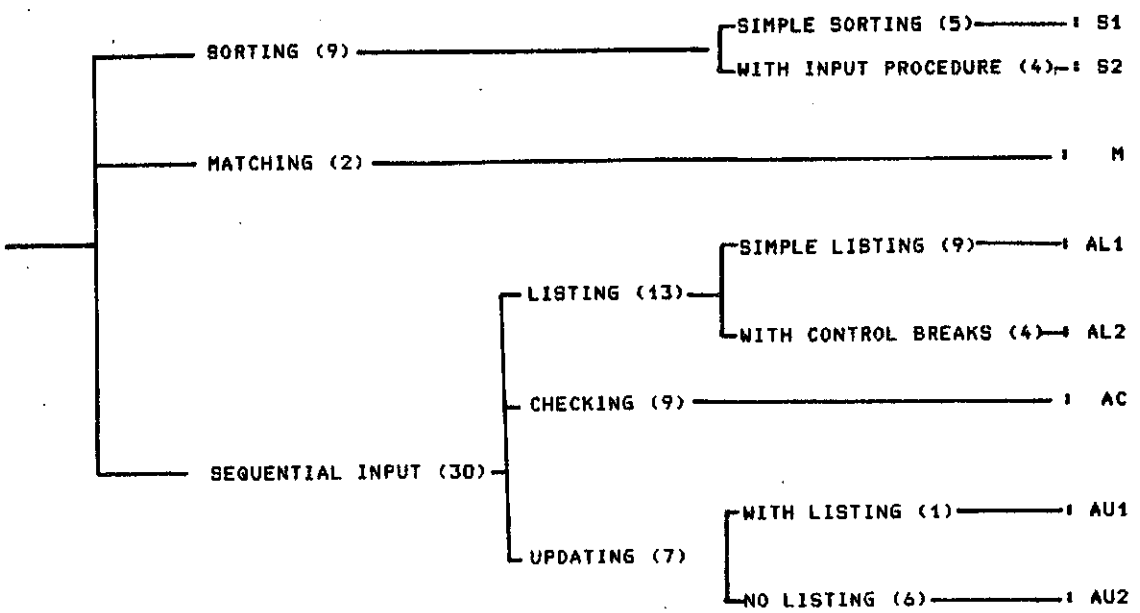
Three sub-branches have been found:

1. listing a) a simple list or b) a list with control breaks
2. checking the input record
3. updating a master file using the record read

Corresponding pseudo programs would appear as:

```
LISTING:
      edit the input record;
      if new page required
            print the heading;
      print the edited record;
```

(the two sub-braches differ as regards the request for new page : simple lists print new headings when end of page has been reached: while list with control breaks may request new heading because of the control break that has occurred in the stream of input records)



number of programs falling into the given prototype family is given by the number in parentheses

```
CONTROL PROGRAMS:
      check the input record;
      if record ok
        write the record
      else
        edit and print it
        with corresponding diagnosis;

UPDATE PROGRAMS:
      check for transaction type;
      if      a do A
      else if b do B
      else if c do C
      ....
      else if z do Z;
      clean up;
```

(the branch with listing has an "else if" line like: else if l do listing)

THE BOTTOM-UP SIDE OF THE STORY

Reduction in the amount of code to be written can also be made by strict standardisation of lowest-level modules. Desk top inspection of VTOCs showed that they nearly exclusively deal with the I/O operations. Their function is clear (which is one of the most stringent requests for modularity of software) and does not differ much from file to file; further, because of user's request, all files have direct-access organisation; thus every record in every file has at least three occurences: when it is written, read, and deleted (not to talk about possible corrections of the record). Usually this instances in record's life occur in different programs, which is a reason per se to standardize the modules in a form of prefabricated sections - with prototype sections serving as starting points for final versions of modules -.

STATISTICS

For every program out of 41 under scrutiny the following numbers have been collected:

1. total number of sections
2. number of sections coinciding with the prototype
3. number of files used by the program

The following table summarizes what we have found out:

TABLE I

program	I	II	III	IV	V	prototype
NPSC30	-	-	-	-	-	S2
NPSC10	-	-	-	-	-	S2
NPSP10	-	-	-	-	-	S2
NASO20	-	-	-	-	-	S2
NASZ20	-	-	-	-	-	S1
ZASP10	-	-	-	-	-	S1
NPSC20	-	-	-	-	-	S1
NPSP20	-	-	-	-	-	S1
NPSO30	-	-	-	-	-	S1
NAKO22	26	7	8	27	54	M
ZAUP10	24	8	6	33	54	M
NALZ20	9	6	4	67	100	AL1
NPLO20	9	5	5	56	100	AL1
NPLO30	7	5	3	71	100	AL1
NPLC20	10	5	4	50	80	AL1
NPLP10	9	5	3	56	78	AL1
NPLC40	9	6	2	67	78	AL1
NALZ21	12	6	3	50	67	AL1
NPLC30	13	5	4	39	62	AL1
NAUL14	53	7	9	13	28	AL1
NARF11	11	6	5	55	91	AL2
NAUZ10	14	8	6	57	79	AL2
NAUL12	15	7	4	47	67	AL2
NAUL13	17	7	6	41	71	AL2
NAKF10	16	9	8	56	100	AC
NPKP10	16	8	7	50	88	AC
NPKC10	21	12	6	57	81	AC
NAKF11	17	10	3	59	71	AC
NAKF12	17	9	3	53	65	AC
ZAAP10	28	10	9	36	64	AC
NAKO21	18	9	3	50	61	AC
ZAKP10	35	12	9	34	57	AC
NAKO20	42	10	9	24	43	AC
NAUL10	26	5	6	19	39	AU1
NPLC10	13	10	4	77	100	AU2
NPAP10	17	10	8	59	100	AU2
NPAC10	16	10	6	63	94	AU2
NAAF10	15	8	5	53	80	AU2
ZAAP10	28	10	9	36	64	AU2
ZARP10	21	7	6	33	57	AU2
NAUL11	28	5	8	18	43	AU2
NAAO20	51	8	8	16	29	AU2

(LEGEND:
col I : tot. number of sections in the program
col II : number of sections in the prototype
col III: number of files used by the program
col IV : II / I in percents
col V : II + III - 1 / I in percents
(- 1 because of a read module
among the top-level modules)

It is evident from the table above that some of the programs are already very well suited to act as prototype programs (col V). They have a moderate number of sections which are partly top-level modules (second column) and/or bottom-level modules (third column). Thus they could be used, without much additional fuss, as templates for the programs which fall into the same prototype class of programs.

The column V shows us that a large portion of program code (30% for larger programs and up to 100% for smaller) can be produced without starting from scratch; a suitable prototype program can be used instead.

Of course there is no proof that the amount of work requested for a CORRECT program is a direct function of number of sections the program has. A lot of other factors is also present. Usually the programming effort is measured in the number of lines of code to be produced. If we, however, assume that the function of sections is well-determined and and that sections are loosely connected (simple interfaces), the correlation between the quantities oblained from specifications (which are given in the table above) and actual effort of the team will be significant.

WHY IT WILL NOT WORK?

The difficulties which the application of prototype programming will meet in this case, can be divided in two broad categories:

1. objective factors
2. internal factors

OBJECTIVE FACTORS:

A decent time-sharing system with effective editor and librarian is needed. Also a full-level COBOL (with copy verb fully incorporated) is a prerequisite! otherwise some other copy utility must be prepared, which will serve as an interface between the development library and the source program files.

The programs discussed here are being produced on a system with following negative points:

1. program library can not be used simultanously by several users
2. only one terminal is available either as RJE or screen terminal
3. the firmware in the terminal is defective and consistently forgets to show the last three or four output lines
4. editor is line-oriented, which, together with the 3rd negative point cited above, makes the whole job a real nightmare

INTERNAL FACTORS

Far more important from objective factors are the factors connected with the team which is working on the project. If the new technology is to suceed the following prerequisite should be fulfilled:

1. Programming standards should be enforced
2. The project should be planned as any other manufacturing process
3. The progress of the project should be followed and controled
4. The interaction between the members of the team should be encouraged to enable the dissemination of new ideas

The team which is working on the software project used as a work case here consists of app 15 to 20 members.This is definitely too large a number. The result is a loosely connected team where news propagate very slowly. Further, the team is organized very democratic1y which is just an euphemism for saying that everybody does just as he pleases. Although standards exist, no mechanism is available to force programmers to stick to them.As the progress of the project is not being followed, the critical points are not identified and the programmers spend 90% of their time waiting for somebody else to do his job.

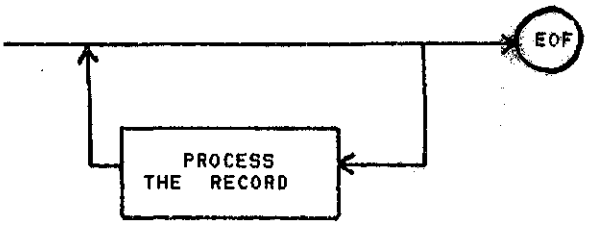
Thus instead of following the cornerstones on the road to successful end of the story, the project is moving slowly from one tombstone to another on the way into troubled waters of failure.

CONCLUSIONS

It has been shown that a significant amount of programming effort can be reduced by identifying redundant parts of programs to be produced and using prototype programs as starting point of programming.

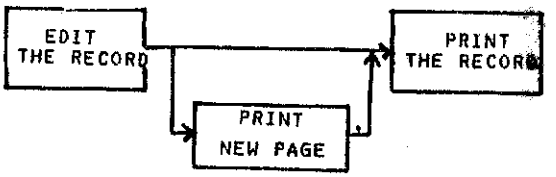
However, no success is guaranteed without sensible management of the project. This is nothing new to people dealing with software technologies may come and go, but problems remain the same. Which explains the truth written in the title.

SEQUENTIAL INPUT

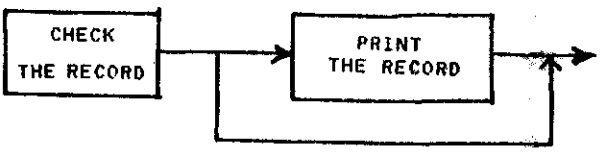


PROCESS THE RECORD:

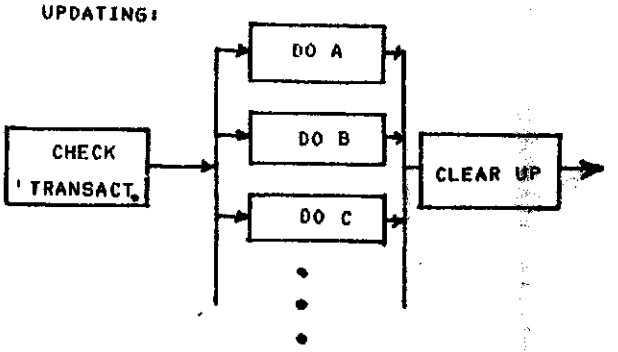
LISTING:

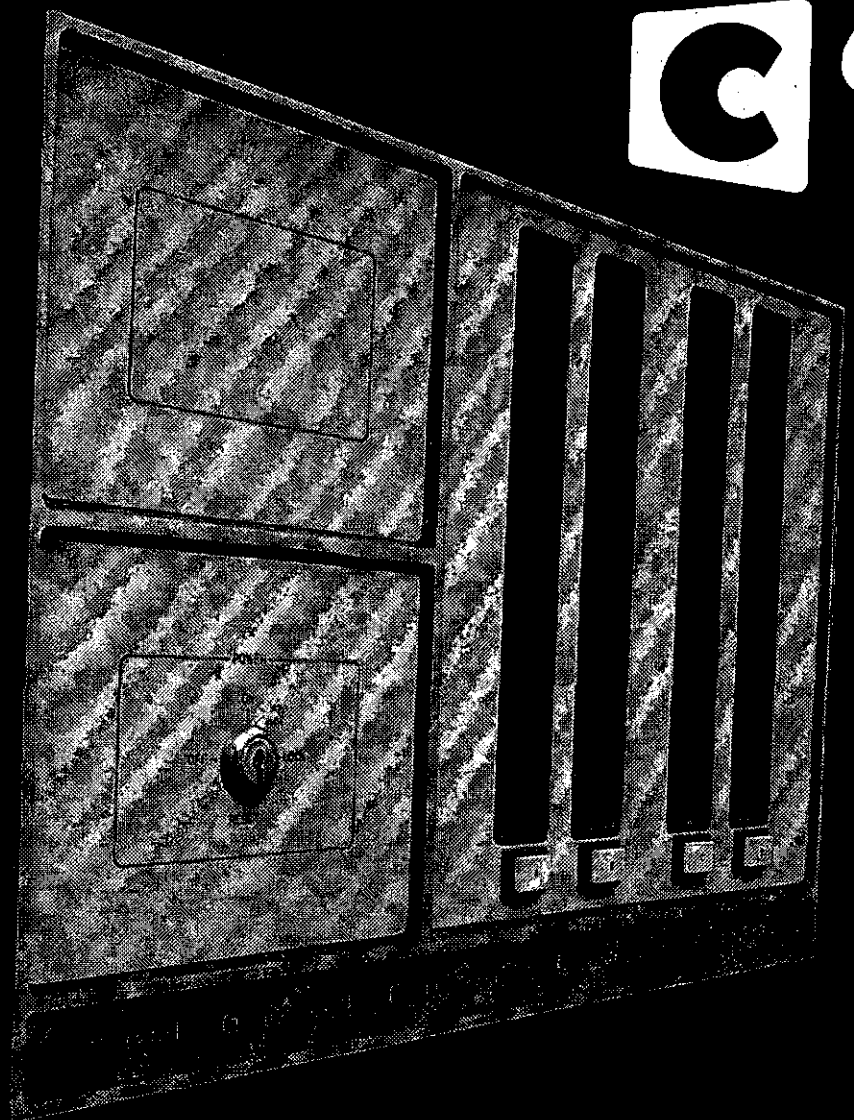


CHECKING:



UPDATING:

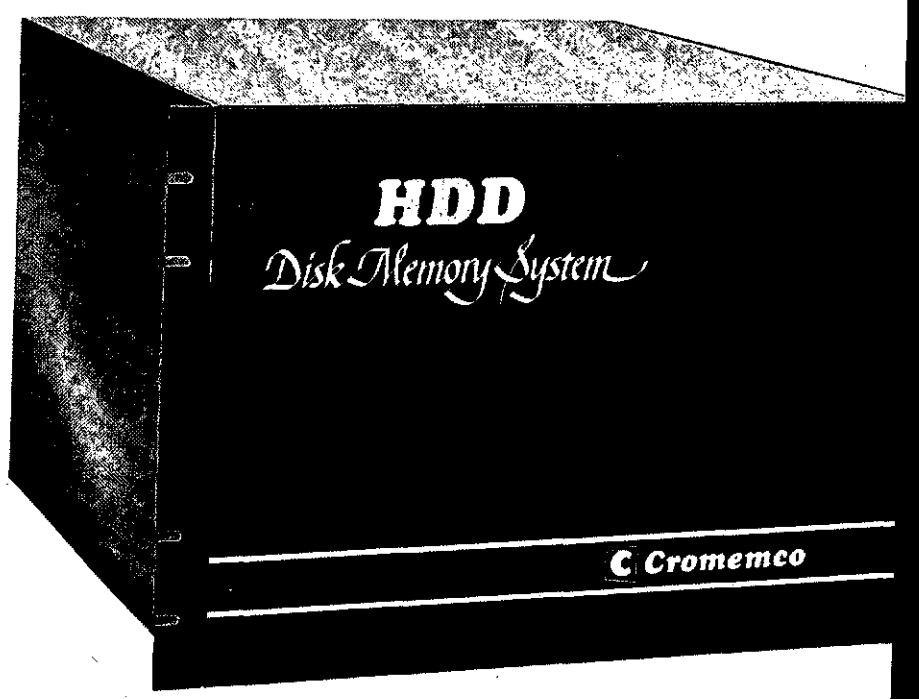




Cromemco
System Three Computer

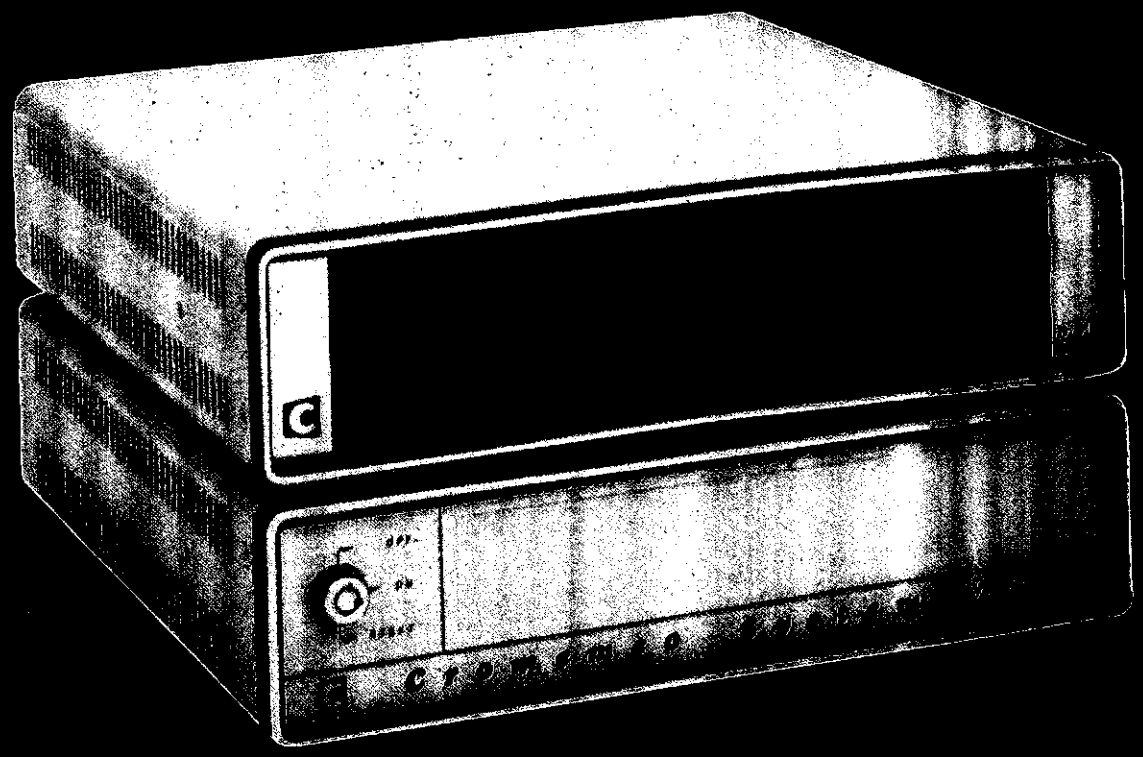
Mikro-kompjutor za profesionalnu upotrebu

11 ili 22 Megabyte Winchester Hard disk



Kolor grafički sistem visoke rezolucije

System zero – namijenjen za specijalne aplikacije



Ekskluzivni zastupnik za SFR Jugoslaviju:

agromarketing

41000 Zagreb, B. Adžije 7/1, P.P. 5
Telefon: (041) 417 662, telex: 21741

INFORMATICA 4/1981

FORTRAN 8X – REVIZIJA FORTRAN-A 77

MLADEN A. VOUK

UDK: 519.682 FORTRAN

SVEUČILIŠNI RAČUNSKI CENTAR
ENGELSOVA bb, 41000 ZAGREB

Opisani su najvažniji trenutno prihvaćeni prijedlozi za reviziju važećeg FORTRAN-skog standarda (ANSI X3.9-1978) poznatog i kao FORTRAN 77. Nova verzija standarda za sada je poznata kao FORTRAN 8X i donosi značajne izmjene u arhitekturi jezika (predlaže se modularizacija) i formi izvornog koda, značajna proširenja u kontrolnim strukturama jezika (npr. CASE-block), mogućnostima manipulacije poljima (npr., processiranje segmenata polja), uvodjenje velikog broja novih intrinzičnih funkcija, dinamičku alokaciju memorije itd. Ukratko su prodiskutirani plan rada na novom standardu i mogući utjecaj na korisnike FORTRAN-a.

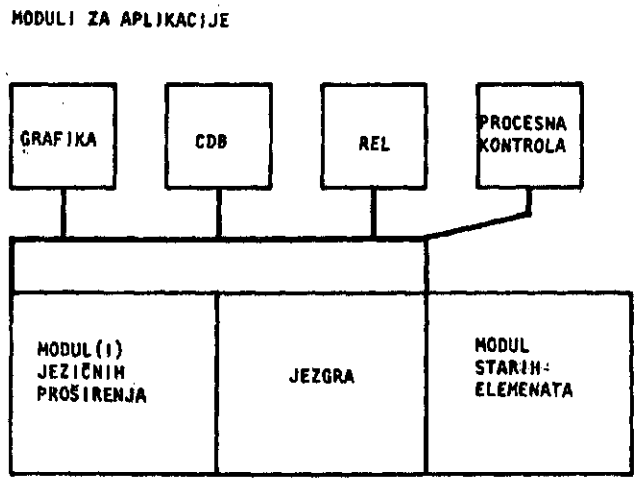
FORTRAN 8X - REVISION OF FORTRAN 77: More important currently accepted proposals for revision of the FORTRAN 77 standard (ANSI X3.9-1978) are described. The working name for the new version of the standard is FORTRAN 8X. Significant changes are proposed for the language architecture (modularization) and the FORTRAN source form (free form). New control structures, such as the CASE-block are introduced. Also proposed are special array processing facilities, a number of new intrinsic functions, dynamic memmory allocation etc. New FORTRAN standard development schedule and its future impact on the FORTRAN users are briefly discussed.

1. UVOD

Historija FORTRAN-a poznata je danas manje više svakom tko ozbiljnije koristi taj viši programski jezik i može se naći u skoro svakom udžbeniku tog jezika (npr. Tucker 1977, Katzan 1978, Meissner i Organick 1980). I danas je, kao i pred više od četvrt stoljeća, taj jezik prvenstveno namijenjen numeričkim proračunima iako nedavno prihvaćeni standard (npr. ANSI 1978, Brainerd 1978) legalizira manipulaciju znakovima, strukturirano programiranje i sl. i tako znatno proširuje područje njegove primjene. Osim općih jezičnih standarda (npr. ANSI 1966, ANSI 1978, ISO 1980) postoje i razni specifični standardi za proširenje tog jezika za procesne aplikacije (npr. IPW/EWICS 1980) te razna proširenja koja su uveli pojedini proizvodjači računala i prevodioca (npr. UNIVAC 1973, 1979, CDC 1976).

Iako je danas konkurencija već vrlo jaka taj se jezik još uvijek u svijetu i kod nas koristi za proizvodnju enormne količine numeričkih programskih proizvoda. U Sveučilišnom računskom centru (SRCE) u Zagrebu, na primjer, oko 80 % poziva programskim procesorima čine pozivi FORTRAN-skim kompilatorima (Cvitaš 1978).

Osnovu još uvijek predstavlja standardni FORTRAN 66 a ne novi FORTRAN 77, što je prilično razumljivo jer se na taj način prilično povećava prenosivost proizvoda s obzirom da je FORTRAN 66 danas implementiran na praktički svakom tipu računala u formi koja, ako već nije standard ANSI 1966, dovoljno je blizu standarda da konverzija ne predstavlja veliki problem. Medjutim, očito je da u usporedbi s jezicima kao što su ALGOL, PASCAL i slični, FORTRAN-u manjka još mnogo poželjnih karakteristika. Najnoviju konkurenciju, barem u domenu aplikacija u realnom vremenu, predstavlja nedavna standardizacija programskog jezika ADA (ADA 1979, DOD 1980) i podrška koju jeziku ADA pruža Američko ministarstvo obrane. Zato nije iznenadjujuće da pobornici FORTRAN-a konstantno razmišljaju o daljnjoj modernizaciji tog jezika i uvodjenju, po njihovom mišljenju, potrebnih i korisnih konstrukcija, naredbi, funkcija itd. Koliko od tih promjena predstavlja "prirodnu" evpluciju jezika da bi se zadovoljile stvarne potrebe većine sadašnjih i budućih korisnika, a koliko je forsirano unošenje karakteristika koje zastupa jedna veća ili više manjih ali utjecajnih korisničkih grupacija, ili čak pojedini proizvodjači računala i kompilatora, teško je uvijek reći. Sigurno je jedno, a to je da će u slijede-



Slika 1. Šematski prikaz arhitekture jezika FORTRAN 8X. Blok između aplikacijskih modula i jezgre i modula proširenja označava potpunu kompatibilnost a ne neka dodatna svojstva.

ćoj dekadi numerički programski proizvodi, ukoliko budu pisani u FORTRAN-u, sve više koristiti mogućnosti "novih" FORTRAN-a kao što je FORTRAN 77, još neiskristalizirani FORTRAN 8X i IRTF (IPW/EWICS 1980) i slični.

U ovom radu su opisani najvažniji trenutno prihvaćeni prijedlozi za reviziju FORTRAN-a 77 i ukratko je prodiskutiran mogući utjecaj tih izmjena na proizvodnju numeričkih i ostalih programskih proizvoda.

2. FORTRAN 8X

Rad na skupljanju i razmatranju prijedloga za izmjenu i dopunu standarda FORTRAN 77 te na formiranju prijedloga za slijedeću reviziju tog jezika obavlja tzv. X3J3 tehnička komisija Američkog nacionalnog instituta za standarde. Komisija radi u grupama i periodično izdaje izvještaje s prijedlozima izmjena i dopuna. Trenutno prihvaćeni prijedlozi uključuju, na primjer, modularizaciju cijelog jezika, standardizaciju NAMELIST naredbe, uvodjenje novih kontrolnih struktura, procesiranja polja i sl. Plan rada X3J3 komisije predvidja završavanje prvog prijedloga kompletnog prednacrta novog standarda negdje u siječnju 1982. godine, te nacrta za javnu raspravu u studenom 1982. godine. Medjutim, očekuje se da će konačna forma standarda biti gotova tek u 1986. godini.

Ovdje su u glavnim crtama opisane interesantnije izmjene predvidjene za slijedeću reviziju FORTRAN-skog standarda, no treba napomenuti da je materijal uzet iz radnog dokumenta (ANSI 1981, Meissner 1981) i da konačni standard može, ali ne mora u cijelosti ili čak uopće sadržavati izmjene i dopune koje su ovdje opisane ukoliko ih komisija u medjuvremenu modificira ili odbaci.

2.1. Arhitektura jezika

Osnovni koncept revizije FORTRAN-a je postojanje *jezgre* jezika sa dodatnim modulima. Šema arhitekture jezika prikazana je na slici 1.

Jezgra je potpuni i konsistentni jezik koji odgovara nekim dogovorenim kriterijima i koji ima funkcionalne mogućnosti koje nisu lošije od važećeg FORTRAN-a 77, i u prvoj iteraciji je ustvari nešto modificirani FORTRAN 77.

Modul_starih_elemenata jezika u principu sadrži jezične elemente koje X3J3 komisija smatra prevazidjenim, ali koji su neophodni za održavanje kompatibilnosti sa prijašnjim verzijama standarda.

Modul(i)_jezičnih_proširenja sadrži(e) dodatke jeziku koji su sintaktički, semantički i idejno konsistentni sa FORTRAN jezgrom. Elementi proširenja mogu se slobodnije koristiti svim pozitivnim specifičnostima arhitekture računala na kojem se jezik implementira, mogu biti raščeni za usko specijalizirane aplikacije i mogu predstavljati korisne eksperimente u konstrukciji i razvoju jezika, koji medjutim, ne moraju biti pogodni za uvrštavanje u Jezgru.

Moduli_za_aplikacije su standardizirana proširе-

nja programskog jezika FORTRAN za specifične aplikacije kao što su baze podataka (npr. relaciona, CODASYL), grafika ili procesna kontrola. Svi moduli moraju biti potpuno kompatibilni s Jezgrom i Modulima_proširenja. Sami aplikacijski moduli pišu se na osnovu dodatnih standarda koji nisu obuhvaćeni standardom (u ovom slučaju 8X) jezika FORTRAN i koji moraju biti odobreni od X3J3 komisije.

Trenutni sadržaj Jezgre je FORTRAN 77 iz kojeg su izbačene neke naredbe i konstrukcije (Tabela 1). Medjutim, sigurno je da će u taj modul biti dodani neki elementi koji se momentalno nalaze u Modulu_proširenja. Trenutni sadržaj Modula_starih_elementa čine svi elementi FORTRANA 77 brisani iz Jezgre. U principu ovaj modul treba sadržavati sve one elemente koji jesu ili će biti izbačeni iz Jezgre ili Modula_proširenja. Za sada komisija ne predvidja posebni modul koji bi sadržavao elemente FORTRAN-a 66 što bi eventualno moglo prouzročiti probleme kada su u pitanju programski proizvodi koji ovise o nekim specifičnostima standarda FORTRAN 66 kao što je na primjer jedan obavezni prolaz kroz DO petlju kada program dosegne bilo koju DO naredbu. S druge strane skoro je sigurno da će proizvodjači kompilatora, barem još slijedećih desetak godina, ugradjivati mogućnost za rad po standardu 66 odnosno standardu 77 (npr. UNIVAC 1979, 1981, NCSU 1981) jer ako se novi standard i prihvati do kraja 1987. godine onda će, sudeći po dosadašnjem iskustvu, trebati barem četiri do pet godina da se kompilatori koji su u potpunosti suglasni sa novim standardom rašire po svijetu i počnu upotrebljavati u aktivnoj proizvodnji. To u principu znači da će barem do godine 2000. procesori koji su u skladu sa standardom primati sve standardne FORTRAN 77 programe. Uvodjenje modula rezultirati će u dodatnim standardnim naredbama za kontrolu kompilatora kao što je npr. USING. Svaki standardno prihvaćeni modul imao bi neko prihvaćeno ime i neku skraćenicu odnosno prefix za upotrebu kod kodiranja. Moguća imena nekih modula ilustrirana su u tabeli 2. Primjer upotrebe modula dan je na slici 2.

2.2. Kontrolne strukture

Predvidja se postojanje tri kontrolne strukture

IF - blok
DO - blok
CASE - blok

Tabela 1. Trenutni sadržaj jezgre (travanj 1981) FORTRAN-a 8X*

Jezgra je FORTRAN 77 iz kojega su brisani slijedeći elementi:
1. Kolona 6 za nastavak i C za komentar
2. EQUIVALENCE
3. COMMON i BLOCK DATA
4. Predavanje elemenata polja ili substring-a "DUMMY" polju
5. Aritmetički IF
6. Izračunati GO TO
7. Alternativni RETURN
8. ASSIGN i asajnirani GO TO
9. Funkcije definirane aritmetičkim izrazom
10. DOUBLE PRECISION
11. ERR = i END = specifikacije
12. H, X i D format deskriptori
13. Specifična imena intrinzičnih funkcija
14. Asocijacija ENTRY imena

* trenutni sadržaj Modula_starih_elemenata čine elementi FORTRAN-a 77 koji su navedeni kao brisani iz Jezgre. U Jezgru će, medjutim, biti uključeni neki novi elementi kao što je CASE-block, interne procedure, dijelovi prijedloga za procesiranje polja, generičke intrinzične funkcije, izmjene u formi izvornog koda, specifikacije točnosti, GLOBAL deklaracija itd. Svi ostali prijedlozi pripadati će modulima proširenja.

If- blok strukturu već poznajemo iz FORTRAN-a 77 i trenutno nije predvidjeno da se forma te strukture mijenja.

DO-blok ima slijedeću opću formu:

```
DO (kontrola)
   blok
REPEAT
```

gdje je blok, kao IF blok, slijed naredbi koje se mogu početi izvršavati samo od početka, dok je iskakanje iz bloka dozvoljeno. Kontrola može biti izražena na dva načina:

(1) $i = e_1, e_2 [, e_3]$
(2) e_4 TIMES

gdje e_i označava neki realni ili cjelobrojni izraz. Brojač prolaza kroz DO-blok se evaluira na početku bloka i nula prolaza kroz blok je dozvoljeni slučaj. Trenutno se ne predvidja da

Tabela 2. Primjeri imena modula i prefiksa (u zagradama)

MODUL_STARIH_ELEMENATA:	OBSOLETE
MODUL_PROŠIRENJA:	ARRAY_PROCESSING
APLIKACIJSKI_MODUL:	CODASYL_DATA_BASE(CDB)
APLIKACIJSKI_MODUL:	RELATIONAL_DATA_BASE (REL)
APLIKACIJSKI MODUL:	GRAPHICS (GRAF)
APLIKACIJSKI MODUL:	NUMERICAL_ANALYSIS (NAG)

```
USING CODASYL_DATA_BASE  PREFIX = CDB, &
    & NUMERICAL ANALYSIS PREFIX = NAG &
   & PROCLIB = NAG*NAGASCII, OBSOLETE
      :
    CDB_FETCH (NEXT,SET=SET1, END = 123) X, Y, Z
    CALL FETCH (SIZE)
    CALL NAG_E04FAF (A,B ...)
      :
    END
```

Slika 2. Primjer upotrebe modula i prefiksa u FORTRANskim naredbama. CDB_FETCH označava CDB specifičnu naredbu, dok FETCH može predstavljati korisnički potprogram. Imenovanje OBSOLETE modula znači da se bilo kakve nekompatibilnosti između Modula_starih_elemenata i modula proširenja odnosno Jezgre rješavaju u korist Modula_starih_elemenata.

bi DO blok mogao zamijeniti klasičnu (FORTRAN 77) DO petlju nego bi predstavljao jednu dodatnu konstrukciju.

Case-blok ima slijedeću opću formu:

```
SELECT CASE (exp)
CASE blok-selektor
   blok
[CASE blok-selektor
   blok
   :       ]
END SELECT
```

gdje je exp neki izraz a blok-selektor ima slijedeću formu

(selektor_raspona ili vrijednosti [, selektor_raspona ili vrijednosti ...])

ili

DEFAULT

selektor_vrijednosti je izraz (konstanta) istog tipa kao exp, a selektor_raspona je oblika [exp1] : [exp2] i označava raspon unutar kojega se smatra da CASE izraz (exp) aktivira taj blok _selektor. Ukoliko je selektor_raspona izostavljen uzima se minimum i maximum implementirane vrijednosti za tip exp-a.

Primjeri kontrolnih struktura dani su na slici 3.

2.3. Procesiranje polja

Za numeričku analizu jedan od najinteresantnijih predloženih noviteta je veliko proširenje mogućnosti FORTRAN-a za procesiranje polja. Osim skalara i polja uvodi se još jedan koncept tzv. sekcija polja, tj. podsekvenca elemenata polja. Opća forma adresiranja sekcija polja je:

a(ss, [,ss ...])

gdje je a ime polja, (ss [,...]) je tzv. subskript sekcije, a ss je izraz za sekcijski subskript.

```
!KOD ZA PROVJERU
!NEUPARENIH ZAGRADA
      :
CHARACTER LINE (80)
      :
LEVEL = 0
DO (I = 1,80)
        SELECT CASE (LINE (I:I))
        CASE ('(')
           LEVEL = LEVEL + 1
        CASE (')')
           LEVEL = LEVEL -1
           IF (LEVEL.LT.0) THEN
            PRINT*,'NEOČEKIVANA DESNA ZAGRADA'
            EXIT
           ENDIF
        CASE DEFAULT
           ! IGNORIRAJ SVE OSTALE ZNAKOVE
        END SELECT
REPEAT
IF (LEVEL.GT.0) THEN
        PRINT*,'NEUPARENA LIJEVA ZAGRADA'
ENDIF
      :
```

Slika 3. Primjer upotrebe novih kontrolnih struktura

Broj ss-a mora biti jednak broju dimenzija u deklaraciji polja. Sam ss može biti posebni znak i za ilustraciju neka je to zvjezdica

(tzv. "forward section selector), zvjezdica sa predznakom minus ("revers ss"), konstrukcija ss1 : ss2 [: ss3] gdje je ssi neki izraz ("indexed section selector") ili izraz koji sadrži jednodimenzionalno polje ("vector-valued ss"). Primjeri sekcije polja dani su na slici 4.

```
1. A(*,2,*) je dvodimenzionalna sekcija
           trodimenzionalnog polja
2. INTEGER A(4:7,3),B(4)
   B(-*) = A(*,3)
   ta jednakost uparuje slijedeće elemente:

   B(4)   B(3)    B(2)    B(1)
   sa
   A(4,3) A(5,3)  A(6,3)  A(7,3)
3. Ako je definirano
   INTEGER Z(5,7), V(3)
   i V ima slijedeće vrijednosti
   V = 2,1,1,3  onda sekcija Z (3,V) sadrži
              ove elemente polja Z:
   Z(3,2),  Z(3,1),  Z(3,1),  Z(3,3)
```

Slika 4. Primjeri sekcija polja

Paralelno sa sekcijom polja uvode se i izrazi i operacije sa poljima. Tako na primjer aritmetički izraz odnosno pridruženje C = A + B (gdje su A, B i C recimo dvodimenzionalna polja) predstavlja formiranje sume odgovarajućih elemenata od A i B da bi se dobili elementi od C. Jasno A, B i C moraju imati kompatibilne dimenzije. Predlaže se uvodjenje nove funkcijske definicije:

ELEMENTAL EXTERNAL fun [,fun ...]

kojom se funkcija označava kao ona kod koje se, ukoliko su argumenti polja, funkcija koju obavlja vrši pojedinačno na svim elementima polja. Predlaže se uvodjenje novih funkcija PACK i UNPACK za kompresiju podataka, te blok strukture WHERE-blok i OTHERWISE-blok prvenstveno namijenjene za rad sa poljima. Uvode se brojne nove intrinzične funkcije za manipulaciju s poljima, kao što su funkcije za redukciju polja (npr. MASK_SUM (C,C.GT.0.), PRODUCT (A,4)), razne manipulacije matricama i vektorima (npr. TRANSPOSE (A) ili DOTPRODUCT (A,B)) itd. Za definiciju sekcije polja koja sadrži medjusobno odvojene elemente izvornog polja predlaže se naredba IDENTIFY čija se konačna forma još razradjuje.

Primjeri nekih novih intrinzičnih funkcija dani su u tabeli 3.

Tabela 3. Primjeri novih intrinzičnih funkcija za rad s poljima *

Intrinzična funkcija	Generičko ime	Broj argumenta
SKALARNI PRODUKT VEKTORA (A1,A2)	DOTPRODUCT	2
TRANSPONIRANJE MATRICE	TRANSPOSE	1
SUMA SVIH ELEMENATA POLJA A1 PO DIMENZIJI A2	SUM	1 ili 2
SUMA SVIH ELEMENATA POLJA A1 PO NEKOJ OD DIMENZIJA A3 KADA JE A2 .TRUE.	MASK_SUM	2 ili 3
PRODUKT SVIH ELEMENATA POLJA A1 PO DIMENZIJI A2	PRODUCT	1 ili 2
PRODUKT MATRICA (A1 sa A2)	MATMUL	2
STVARANJE DIJAGONALNOG POLJA	DIAGONAL	2
KRUŽNI POMAK POLJA	CSHIFT	3
MINIMUM OD SVIH ELEMENATA OD POLJA A1 PO DIMEZIJI A2	MINVAL	1 ili 2

* A1, A2, A3 označavaju prvi, drugi odnosno treći argument

2.4. Dinamička alokacija polja

Isto tako interesantan i koristan prijedlog je uvodjenje dinamičke alokacije memorije kroz tzv. automatska polja koja bi se medjutim mogla deklarirati samo u potprogramima. Ta bi deklaracija dozvoljavala uvodjenje potpuno novog polja čije su dimenzije potprogramu predane kao argumenti ili kroz COMMON naredbu odnosno ekvivalent COMMON naredbe.

2.5. Izmjene u formi izvornog koda i neke druge izmjene

Predlažu se temeljne izmjene u dozvoljenoj formi FORTRAN-skog izvornog koda. Tako se na primjer planira uvodjenje delimitera za naredbu što bi dozvolilo pisanje jedne ili više naredbi po slogu (kartici). Labela bi bila prvi broj odvojen blankom od ostatka naredbe a komentar bi započinjao posebnim komentar-delimiterom u bilo kojoj koloni ulaznog sloga. Naredba bi se nastavljala u slijedećem slogu tako da bi se u prethodnom završavala i u novom počinjala posebnim znakom za nastavak. Zatim se predlaže povećanje dozvoljenog FORTRAN-skog seta znakova kao i povećanje duljine FORTRAN-skog imena na 31 alfanumerički znak (prvi mora biti slovo) i

legaliziranje upotrebe podvlake u imenima (npr. NOVI_SLOG bi bilo legalno FORTRAN-sko ime). Predloženi posebni delimiteri i znakovi dani su u tabeli 4.

Tabela 4. Novi delimiteri i dodatni znakovi

Svrha	Opis
DELIMITER NAREDBE	TOČKA-ZAREZ (;)
DELIMITER ZA KOMENTAR	USKLIČNIK (!)
DELIMITER ZA NASTAVAK	"AMPERSAND" (&)
ZNAK	DVOSTRUKI NAVODNIK (")
ZNAK	POSTOTAK (%)
ZNAK	MANJE (<)
ZNAK	VIŠE (>)
ZNAK	UPITNIK (?)
ZNAK	PODVLAKA(_)

Ostale dopune i izmjene čine prijedlozi za uvodjenje argumenata ključeva u procedure (npr. PROC (BETA = 8.0) predstavlja legalnu referencu procedure PROC koja ima jedan formalni argument sa imenom BETA), uvodjenje internih procedura koje bi zamijenile funkcije definirane aritmetičkim izrazom, te brojne nove intrinzične funkcije vezane za točnost brojeva (npr. PRECISION(X), TINY(X)) i za uvodjenje novog tipa varijable BIT za manipulaciju bitovima (npr. INTB (b,k), SHIFTB (b,n)). Forma naredbe i način definiranja preciznosti realnih brojeva još se raspravlja. Planira se uvodjenje deklaracije globalnih podataka kroz naredbu GLOBAL DEFINITION i GLOBAL kojima bi se zamijenila današnja COMMON naredba, uvodjenje nekih dopuna ulazno/izlaznim naredbama itd.

3. DISKUSIJA I ZAKLJUČAK

Većina predloženih izmjena i dopuna sigurno će doprinjeti većim mogućnostima i većoj upotrebljivosti programskog jezika FORTRAN. Tako je, na primjer, neosporno da je uvodjenje ALGOL-u sličnih karakteristika, kao što su dinamička alokacija memorije i nova forma izvornog koda, poželjno sa stanovišta fleksibilnosti pisanja koda i efikasnosti korištenja memorije. Isto tako procesiranje polja će sigurno olakšati pisanje numeričkih programskih proizvoda koji se koriste vektorima, matricama ili poljima višeg ranga.

Standardizacija aplikacijskih modula za grafiku, procesiranje u realnom vremenu itd, doprinjeti će povećanju portabilnosti proizvoda koji koriste te ili druge standardizirane aplikacijske module. Medjutim, postoji opasnost da FORTRAN postane prekompliciran i preglomazan i za korisnika i za impelmentatora i da definitivno izgubi bitku sa modernijim jezicima kao što su ALGOL , PASCAL ili ADA.

S obzirom da su kompilatori koji potpuno podržavaju FORTRAN 77 tek počeli ozbiljnije proditati na tržište, portabilnost programa koji koriste neke egzotičnije mogućnosti tog jezika, koliko god dobre i poželjne one bile, smanjena je u usporedbi sa FORTRAN 66 programima. Skoro je sigurno da će se još dugi niz godina većina proizvoda pisanih u FORTRAN-u, ukoliko se želi da su portabilni, biti pisani u FORTRANU 66 ili nekom relativno skromnom supersetu FORTRAN-a 66 ili subsetu FORTRAN-a 77. Tako recimo jedan od velikih proizvodjača numeričkih programskih proizvoda, Numerical Algorithms Group Ltd (NAG), nema namjeru, osim u izuzetnim slučajevima, brzati u FORTRAN 77 implementacije svojih dobro poznatih proizvoda (Du Croz 1980) kao što su NAG biblioteka (NAG 1981a), GENSTAT (NAG 1977) itd. Njihovi novi proizvodi, koji bi u principu morali odgovarati aplikacijskim modulima FORTRAN-a 8X, kao Grafička biblioteka (NAG 1980) i FE (Finite element) biblioteka (NAG 1981b, NAG 1981c) takodjer su pisani u standardnom ANSI 1966 FORTRAN-u. NAG naime smatra da je revizija FORRAN-a 77 neminovna i ne žele se upuštati u konverzije svojih proizvoda prije nego se odobre novi standardi ili FORTRAN 77 kompilatori budu pristupačniji nego što su sada (Du Croz 1980). Na sličan način reagiraju i individualni korisnici i proizvodjači (istraživači) numeričkih programskih proizvoda ili se barem tako čini na osnovu algoritama i opisa proizvoda koji se susreću u literaturi.

Navedeni prijedlozi za reviziju FORTRAN-skog standarda ne obuhvaćaju sve prijedloge koje je X3J3 komisija prihvatila i koji se nalaze u daleko opširnijoj formi u radnim materijalima te komisije. Činjenica da je komisija trenutno prihvatila neki prijedlog za dopunu ili izmjenu ne znači da taj prijedlog prije formulacije konačne forme standarda (koja se očekuje tokom 1986. godine) neće pretrpjeti veće ili manje izmjene ili da eventualno neće u cjelini biti odbačen. Tako su, recimo , na zasjedanju komisije održanom u kolovozu 1981. godine eliminirani ranije prihvaćeni prijedlozi da se u dio standarda za manipulaciju polja uvrste intrinzične funkcije za izračunavanje determinante i inverza matrice. Smatra se, naime, da je za pisanje dobrih potprograma iz linearne algebre

potrebno veliko iskustvo i s obzirom da postoje i sve više se koriste specijalizirane veoma dobro testirane i izradjene biblioteke numeričkih potprograma (npr. NAG, IMSL), uvodjenje intrinzičnih funkcija za te operacije ne bi služilo svrsi. Moguće je, medjutim, provesti standardizaciju biblioteka za numeričku i statističku analizu i uvrstiti ih u standard kao aplikacijske module. U svakom slučaju komisija je spremna razmotriti sve dokumentirane prijedloge koje joj dostave individualni korisnici, grupe korisnika ili institucije. Prijedlozi i komentari se mogu dostaviti komisiji preko sekretara komisije (adresa u prilogu) ili se mogu adresirati direktno na komisiju pri Američkom nacionalnom institutu za standarde. Vrijedno je napomenuti da se periodično izdaje jedna publikacija koja se zove "FORTRAN Newsletter" u kojoj se objavljuju i raspravljaju novosti vezane za programski jezik FORTRAN. Publikaciju izdaje Lawrence Berkley Laboratory, a urednik je L.P. Meissner. Publikacija se distribuira besplatno i za uvrštavanje u "mailing"-listu potrebno je poslati pismeni zahtjev uredniku (adresa u prilogu).

Jugoslavenski standardi za programske jezike ne postoje no razumno je očekivati da će se, kada budu formulirani, oslanjati na standarde za programske jezike Internacionalne organizacije za standardizaciju. Za FORTRAN važeći Internacionalni standard (ISO 1980) je ustvari ANSI FORTRAN standard X3.9-1978. Prema tome rad ANSI X3J3 komisije na reviziji FORTRAN-a 77 sigurno će se odraziti i u budućem ISO standardu za taj programski jezik. Kao što FORTRAN 77 postaje FORTRAN 1980-tih godina tako je za očekivati da će slijedeća verzija tog jezika, FORTRAN 8X, postati FORTRAN 1990-tih godina (Meissner 1981).

4. ZAHVALA

Htio bih se zahvaliti Dr B.T. Smith-u, Argonne National Laboratory (USA) i Dr.B. Ford-u, NAG Central Office (Oxford) na susretljivosti i trudu oko nabavke materijala X3J3 komisije.

5. PRILOG

Adresa sekretara X3J3 i urednika publikacije FORTRAN Newsletter

Loren P. Meissner
CSAM - 50B
Lawrence Berkley Laboratory
University of California
Berkley CA 94720, U S A

n a p o m e n a:Ukoliko ste zainteresirani za pojedine prijedloge ili komplet radnih materijala X3J3 dostupnih autoru molimo vas da zahtjev za kopiju uputite, preko vaše biblioteke, biblioteci Sveučilišnog računskog centra u Zagrebu, sa napomenom "FORTRAN 8X - radni materijal".

6.LITERATURA

1. ADA, 1979. Preliminary ADA reference manual, SIGPLAN notices (ACM), Vol 14 (6)
2. ANSI (American National Standards Institute), 1966, Standard FORTRAN, ANSI X2.9-1966, ANSI, New York
3. ANSI, 1979, Programming language FORTRAN, ANSI X3.9-1978, ANSI New York
4. ANSI, 1981, Proposals approved for FORTRAN 8X, X3J3/S6, version 1.77, April 1981.
5. Brainerd W., 1978, FORTRAN 77, Communications of the ACM, Vol, 21 (10), 806-820
6. CDC (Control Data Corporation), 1976, FORTRAN extended 4, Reference Manual, CDC publication no. 84000009, CDC, Mineapolis
7. Cvitaš V., 1978. Magistarski rad, EIT, Sveučilište u Zagrebu
8. DOD (U.S. Department of Defence), 1980, Reference Manual for the ADA Programming Language, proposed standard document
9. Katzan M.Jr, 1978, FORTRAN 77, Van Nostrand Reinhold Co, New York
10. Du Croz J.J, 1980, The Impact of FORTRAN 77 on the NAG Library, NAG Newsletter, 1/80, 4-7
11. IPW/EWICS (International Purdue Workshop and Industrial Computer Systems and European Workshop on Industrial Computer System), 1980. IRTF - Industrial Real Time FORTRAN, TC1, 2.2/80, Draft Standard
12. ISO (International Standard Organization), 1980, International Standard ISO 1593-1980, Programming Languages: FORTRAN, ISO, Geneva
13. Meissner L.P. i Organick E.I., 1980, FORTRAN 77 Featuring Structured Programming, Addison Wesley
14. Meissner L.P., ed. 1981, FORTRAN Newsletter "FOR-WORD", Vol 7 (4), 13 - 16
15. NAG (Numerical Algorithms Group Ltd), 1977, GENSTAT, A General Statistical Program, Manual, Rothamsted Experimental Station, NAG, Oxford
16. NAG, 1980, The Development of a Graphical Supplement for the NAG Library, NAG Newsletter, 2/80, 11 - 14
17. NAG, 1981a, FORTRAN Library Manual, Mark 8, NAG, Oxford
18. NAG, 1981b, The Finite Element Library, NAG Information Note, NAG, Oxford
19. NAG, 1981c, NAG Newsletter, 2/81
20. NCSU (North Carolina State University), 1981, FORTRAN 77 ready testing, NCSU Computing Centre Newsletter, 1981-08,4
21. Tucker A.B. Jr, 1977, Programming Languages, McGraw-Hill Book Co
22. UNIVAC, 1973, FORTRAN V Programmer Reference, SPERRY UNIVAC, UP-4060 rev.2
23. UNIVAC, 1979, FORTRAN (ASCII) 9R1, Programmer Reference, SPERRY-UNIVAC, UP-8244.1
24. UNIVAC, 1981, FORTRAN (ASCII 10R1 Release Descr., SPERRY-UNIVAC, SRD-288.3, February 1981.

INFORMATICA 4/1984

PRIMJENA LEKSIKOGRAFSKE METRIKE ZA PRORAČUN DEBALANSA LOPATICA ZRAKOPLOVNIH TURBINA

JOVAN LONČAR

VIŠA ZRAKOPLOVNA ŠKOLA – ZAGREB

UDK: 681.3:51

U radu se uvodi leksikografska metrika u prostor permutacije koja se zatim koristi za proračun debalansa.

APLICATION OF THE LEXICOGRAFIC METRIC TO CALCULATION OF THE BALANCE OF THE AIRPLANE TURBINE SHOVELS: The lexicografic metric is introduced in space of permutation and used for the calculation of the balance of the airplane turbine shovels.

UVOD

U [3] [4] [5] u prostor permutacija uveli smo lančanu, inverzionu i transpozicionu metriku i u njima dobili različite vrijednosti debalansa. U ovom radu uvodimo leksikografsku metriku i u njoj računamo vrijednost debalansa.

LEKSIKOGRAFSKA METRIKA

Na skupu permutacija uvedimo tkz. leksikografsku uređenost. Uzmimo da permutacija $p_2=(j_1 \ldots j_n)$ slijedi iza permutacije $p_1=(i_1 i_2 \ldots i_n)$ i označimo $p_1 \prec p_2$ ako postoji takav cijeli broj α $(1 \le \alpha \le n)$ da imamo $j_1=i_1, j_2=i_2 \ldots j_{\alpha-1}=i_{\alpha-1}$, no $j_\alpha \neq i_\alpha$. Uvjeti refleksivnost i antisimetričnosti su ispunjeni. Pokažimo da je ispunjen i uvjet tranzitivnosti. Neka je $p_3=(k_1 k_2 \ldots k_n)$ i $p_1 \prec p_2$. Zapis $p_2 \prec p_3$ označava da postoji takav cijeli broj β $(1 \le \beta \le n)$ da je $j_1=k_1, j_2=k_2 \ldots j_{\beta-2}=k_{\beta-2}$, $j_{\beta-1}=k_{\beta-1}$ a $j_\beta < k_\beta$. Neka je $\gamma = \min(\alpha,\beta)$ tada je $i_1=k_1$, $i_2=k_2 \ldots i_{\gamma-1}=k_{\gamma-1}$, $i_\gamma \le k_\gamma$ odnosno $i_\gamma < k_\gamma$, budući je $i_\gamma < j_\gamma$ (za $\gamma=\alpha$) ili $j_\gamma < k_\gamma$ za $\gamma=\beta$. Slijedi da $p_1 \prec p_3$, čime je tranzitivnost dokazana. Za bilo koje različite permutacije p_1, p_2 uvjek vrijedi $p_1 \prec p_2$ ili $p_2 \prec p_1$, jer uvijek postoji indeks α za koji je $i_\alpha \neq j_\alpha$. Prema tome leksikografska uređenost je linearna uređenost skupa permutacija.

Prednji princip uredjaja iskoristimo i za uvođenje metrike. Naime, svakoj $p \in \pi$ pridružimo broj $Z(p)$ koji u stvari predstavlja redni broj permutacije kad je π leksikografski uredjen.

Udaljenost $d(p_1,p_2)=|Z(p_1)-Z(p_2)|$. Da su ispunjeni aksiomi metrike nema potrebe provjeravati jer je sada prostor permutacija π s uvedenom udaljenošću izometričan skupu prirodnih brojeva od 1 do $n!$. (n-faktorijela) s običnom Euklidskom udaljenošću. Krajnji su slučajevi $p=(1,2 \ldots n)$ kojoj odgovara $z(p)=1$, a permutaciji $p=(n,n-1, \ldots 3,2,1)$ broj $Z(p)=n!$. Znači da je $d_{max}(p_i,p_j)=n!-1$. Da bi izračunali $Z(p)$ za proizvoljni $p \in \pi$ uvedimo na π razbijanje na klase m-tog reda [1]. Za $p_1=(i_1 \ldots i_n)$ te $p_2=(j_1 j_2 \ldots j_n)$ kažemo da ulaze u istu klasu reda m $(m=1,n)$ ako je $i_1=j_1 \ldots i_m=j_m$. Skup π se sastoji od n-klasa I reda, $n(n-1)$ klasa II reda itd. Za proizvoljni m, razbijanje m-tog reda predstavlja razbijanje na disjunktne klase čija unija čini skup π. Ako je $\ell > m$ onda razbijanje ℓ-tog reda predstavlja usitnjavanje razbijanja m-tog reda tj. za $\ell > m$ bilo koje permutacije iz jedne klase ℓ-tog reda obavezno ulaze u jednu klasu n-tog reda. Sve klase istog reda sadrže jednak broj permutacija. Postoji hijerarhija klasa. Čitav skup π koji se sastoji od $n!$ permutacija ima n-klasa I reda, a u svakoj klasi ima $(n-1)!$ elemenata. Svaka klasa I reda sadrži (n-1) klasa II reda od kojih svaka ima $(n-2)!$ permutacija itd. Demonstrirajmo prednje za slučaj n=5. Tada imamo: 5 klasa I reda, svaka klasa ima 24 permutacije

20 klasa II reda, svaka ima 6 permutacija,
60 klasa III reda, svaka ima 2 permutacije,
120 klasa IV reda, svaka ima 1 permutaciju.
Unutar svake klase m-tog reda, klase (m-1) reda možemo prenumerirati na slijedeći način:
Prvom klasom (m+1) reda nazivamo klasu sa najmanjim elementom na (m+1) mjestu, drugom klasom nazivamo klasu sa slijedećim rastućim elementom na (m+1) mjestu itd.
Opća formula za izračunavanje broja permutacije u zavisnosti od broja klasa različitih redova, u koje ulazi dana permutacija ima slijedeći oblik. Postoje i drugi izrazi pomoću kojih se može izračunati leksikografski broj permutacije Z(p).
Označimo ℓ_i sa ℓ brojeve klasa m-tih redova u klasama (n-1) redova, onda je broj permutacija u nizu od n permutacija dan izrazom:

$$Z(p)=(\ell_1-1)(n-1)!+(\ell_2-1)(n-2)!+\ldots(\ell_m-1)(n-m)!+\ldots(\ell_{n-2}-1)2!+(\ell_{n-1}-1)+1 \qquad (1)$$

Ako je zadana fiksna permutacija p_0, onda se može generirati permutacija iz kruga radiusa R, s centrom u p_0, pomoću slijedećeg algoritma [1] :

ALGORITAM 1

Prvi korak: Pomoću formula (1) odrediti Z(p)
Drugi korak: Iz segmenta [1,R] slučajno izabrati cijeli broj r i predznak plus ili minus. Izračunati $Z(p)=Z(p_0)\pm r$
Treći korak: Ako je $1\le Z(p)\le n!$ prijeći na slijedeći korak inače se vratiti na korak 2.
Četvrti korak: Postupno odrediti brojeve k_i (i=1,2...n-2) i to po pravilu:
k_1 - je ostatak od djelenja $|Z(p)-1|$ na $(n-1)!$
k_i-(i=2,3...n-2) je ostatak od dijeljenja k_{i-1} na $(n-i)!$
Peti korak: Odredite ℓ_i (i=1,2...n-1) pomoću

$$\ell_1=N\left[\frac{Z(p)-1}{(n-1)!}\right]+1 \qquad l_i=N\left[\frac{K_{i-1}}{(n-i)!}\right]+1$$

gdje je: i=(2,3...n-2), N[ ]cijeli dio izraza u[ ].
Šesti korak: Na osnovu dobivenog niza $\{\ell_i\}$ formirati permutaciju $p=(i_1 i_2 \ldots i_n)$ na slijedeći način: staviti $i_1=\ell_1$ i precrtati i_1 u nizu (1,2,3...n). Staviti i_2=broju koji stoji na prvom mjestu u proređenom nizu i precrtati i_2 iz niza i.t.d.
i_n= jedinom preostalom neprecrtanom elementu u nizu (1,2,3,...n). Nedostatak prednjeg algoritma je u tome što se on može primjeniti za relativno male n, tako da već kod n=20 dolazi do poteškoća. Naime, u zavisnosti od duljine memorijske riječi računara, ograničena je i veličina broja koji se može zapisati. Tako za n=21 imamo $21!\simeq 10^{20}$ što prelazi mogućnosti nekih računara.

Za formiranje permutacija koje leže u zadanom krugu za (n>20) razradjen je algoritam zasnovan na faktorijelnom sistemu računanja. Recimo da imamo permutacije $p=(i_1 i_2 \ldots i_n)$ za svaki i_α (osim i_n) stavimo: $s_\alpha = i_\alpha - \ell_\alpha$, gdje je ℓ_α-broj elemenata skupa $\{i_\beta : \beta<\alpha, i_\beta \le i_\alpha\}$ (3)
Tada leksikografski broj permutacije možemo zapisati

$$Z(p)=\sum_{\alpha=1}^{n-1} s_\alpha\,(n-\alpha)! \qquad (4)$$

Neka je $p_0\in\pi$ centar neke okoline radiusa R, a r- udaljenost do neke permutacije p_1 $(r\le R)$. Tada je:

$$\left|Z(p_0)-Z(p_1)\right| = r \text{ ili}$$

$$\left|\sum_{\alpha=1}^{n-1} s_\alpha^0\,(n-\alpha)! - \sum_{\alpha=1}^{n-1} s_\alpha^1 (n-\alpha)!\right| = r \qquad (5)$$

gdje je s_α^0 - koeficient s_α u(4) za p_0, a

s_α^1 - koeficient u (4) za p_1.

Nakon prenumeriranja dobijemo

$$\left|\sum_{\alpha=1}^{n-1} s_{n-\alpha}^0\cdot\rho! - \sum_{\alpha=1}^{n-1} s_{n-\alpha}\cdot\rho!\right| = r \qquad (6)$$

U (6) su nam poznati s_α^0 i r. Treba odrediti s_α^1, a zatim normirati permutaciju p_1.

Izraz $Z(p)=\sum_{\alpha=1}^{n-1} Z_{n-\alpha}\cdot\rho!$ je konvolucija nekog pozicionog zapisa broja Z(p) u kojoj α-ti razred ima težinu $\alpha!$, po čemu je sistem i dobio naziv faktorijelni [2].
Faktorijelni sistem je pozicioni jer vrijednost cifre zavisi od položaja unutar broja. Kod toga je težina α-te cifre (računajući s desna), cijelog faktorijelnog broja jednaka $\alpha!$ dok je težina α-te cifre poslije zareza, koji dijeli cijeli i razlomljeni dio mješovitog broja jednaka $1/(\alpha+1)!$.

Uzmemo li da se cifra broja može predstaviti, ne jednim simbolom, već nizom simbola, onda nas može zadovoljiti deset simbola 0,1...9 s tim da se predvidi simbol za odvajanje jednih od drugih faktorjelnih cifara. Budući je težina

α-tog razreda jednaka α-1, u njemu ne treba biti više od α-jedinica prvog razreda, u protivnom pojavljuje se jedinica prenosa u (α+1) razred. Najveći faktorjelni broj po analogiji s dekadskim ...999,999... je oblika...11 10 9 8 7 6 5 4 3 2 1, 1 2 3...

Budući je po poznatoj permutaciji lako dobiti cifre faktorijelnog broja, te da nije težak prijelaz od faktorjelnog broja na samu permutaciju, možemo brojevima permutacija operirati, a da ne prelazimo u dekadski sustav. Formiranje permutacije p_1 udaljene r od p_o, vrši se pomoću slijedećeg algoritma:

ALGORITAM 2

Prvi korak: Za p_o naći faktorijelni broj $Z(p_o)$
Drugi korak: Broj r-prevesti u faktorijelni broj
Treći korak: Za r>o naći $Z(p_1)=Z(p_o)\oplus r$
Za r<o naći $Z(p_1)=Z(p_o)\ominus r$
Četvrti korak: Naći $p_1\in\Pi$ po njenom faktorijelnom broju $Z(p_1)$.

Algoritam 2 razlikuje se od algoritma 1 po tome što se ovdje radi s faktorijeln m sistemom računanja.
Treba voditi računa da kod zbrajanja (oduzimanja) dvaju faktor jelnih brojeva (α+1) - jedinica α- tog faktorjelnog razreda daje jedinicu prijenosa u (α+1) razred (odnosno posudjena jedinica iz (α+1) razreda daje (α+1) jedinicu u α- razredu).
Za rad u leksikografskoj metrici sačinjen je program za računar. Usporedbe radi računate su vrijednosti funcionala-debalansa

$$f(i_1 \dots i_n)=\left[\left(\sum_{i=1}^{n} M_{ik}\cos\frac{2\pi k}{n}\right)^2+\left(\sum_{i=1}^{n} M_{ik}\sin\frac{2\pi k}{n}\right)^2\right]^{1/2}$$

Naime, uzeti su podaci iz tabele 1, zatim je uzeta ona $p\in\Pi$ na kojoj je u 1oo pokusa metodom MONTE-KARLO postignuta minimalna vrijednost [3]. Iz okoline te permutacije s radiusom R=5, u leksikografskoj metrici birano je 1oo $p\in\Pi$ i dobiveni su rezultati koje vidimo na slici1
Dobiveno je:
Minimum debalansa = 1o4,4
Maksimum debalansa = 122,79
Matematičko očekivanje = 113,25

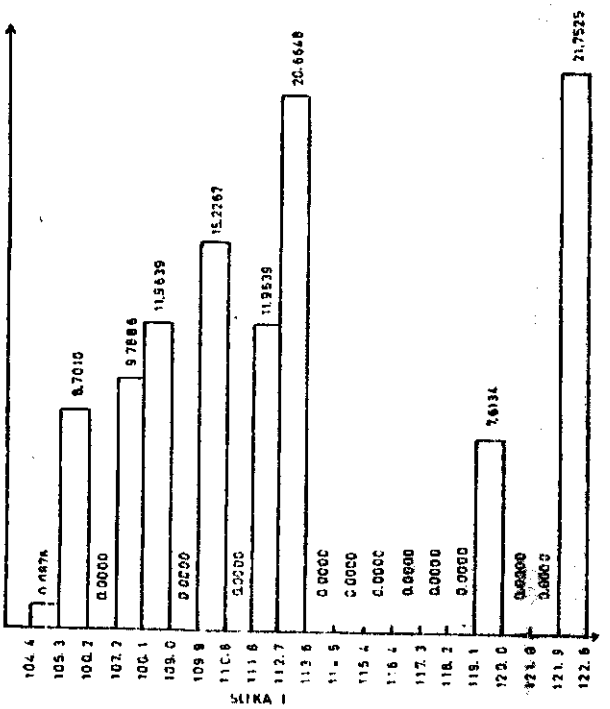
Raspored lopatica koji odgovara minimumu dan je u tabeli 2, a onaj koji odgovara maksimumu u tabeli 3.

ZAKLJUČAK

Vidimo da smo u leksikografskoj metrici dobili slabije rezultate nego u lančanoj i transpozicionoj metrici. Medjutim to nije općenit slučaj. Naime, postoje klase drugih problema gdje ova metrika daje bolje rezultate od ostalih metrika.

TABELA 1

	1	2	3	4	5	6	7	8	9	1o	11	12	13	14	15	16
Broj lopatica	0	1o9	-3o9	-31	-411	-179	42o	-145	-3o1	61	181	231	-162	-341	-281	175
Statički moment	17	18	19	2o	21	22	23	24	25	26	27	28	29	3o	31	32
Broj lopatica	49	0	-69	-51	-46	176	-511	-412	-317	47o	79	13	-142	35	-152	-88
Statički moment	33	34	35	36	37	38	39	4o	41	42	43	44	45	46	47	48
Broj lopatica	13	-625	- 6	21	-259	- 53	17	11	-259	15	91	-121	475	- 9	-191	81
Statički moment	49	5o	51	52	53	54	55	56	57	58	59	6o	61	62	63	64
Broj lopatica	15	42	3	-11	-249	22	5	-11	- 5	- 2	17	-113	17	-17	-39	4o8
Statički moment	65	66	67	68	69	7o	71	72	73	74	75	76	77	78	79	8o
Broj lopatica	5	25	15	41	5	-26	22	-11	27	-23	-58	75	49	31	-457	-45
Statički moment	81	82	83	84	85	86	87	88	89	9o	91	92	93	94	95	96
Broj lopatica	-57	- 9	9	0	-11	-25	- 7	51	65	31	44	21	61	21	51	31



SLIKA 1

TABELA 2

Broj mjesta	1	2	3	4	5	6	7	8	9	1o	11	12	13	14	15	16
Broj lopatice	52	73	16	37	13	54	79	8o	81	83	86	51	4	87	88	85
Statički momenti	-11	27	175	-259	-162	22	-457	-45	-57	9	-25	3	-31	- 7	31	-11
Broj mjesta	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Broj lopatice	89	90	91	92	17	19	23	28	11	53	77	32	12	72	59	75
Statički moment	65	31	44	21	49	-69	-511	13	181	-249	49	-88	231	-11	17	-58
Broj mjesta	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Broj lopatice	78	84	93	82	56	3o	39	60	63	25	74	8	38	94	55	69
Statički moment	31	0	61	-9	-11	35	-17	-113	-39	-317	-23	-145	-53	21	5	5
Broj mjesta	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Broj lopatice	95	96	47	67	2	35	57	1	15	26	2o	29	33	7	5	14
Statički moment	51	31	-191	15	1o9	- 6	5	0	-281	47o	-31	-142	13	420	-411	-341
Broj mjesta	65	66	67	68	69	7o	71	72	73	74	75	76	77	78	79	8o
Broj lopatice	34	36	22	41	44	46	50	49	64	58	27	48	10	21	31	42
Statički moment	-625	21	176	-259	-121	- 9	42	15	488	-2	79	81	61	-46	-152	15
Broj mjesta	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
Broj lopatice	61	6	9	24	40	45	3	18	62	65	66	43	76	68	71	70
Statički moment	17	-179	-301	-412	11	475	-309	0	-17	5	25	41	75	41	22	-26

TABELA 3

Broj mjesta	1	2	3	4	5	6	7	8	9	1o	11	12	13	14	15	16
Broj lopatice	52	73	16	37	13	54	79	8o	81	83	86	51	4	87	88	85
Statički moment	-11	27	175	-259	-162	22	-457	-45	-57	9	-25	- 3	-31	- 7	31	-11
Broj mjesta	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Broj lopatice	89	90	91	92	17	19	23	28	11	53	77	32	12	72	59	75
Statički moment	65	31	44	21	49	-69	-511	13	181	-249	49	-88	231	-11	17	-58
Broj mjesta	33	34	35	36	37	38	39	4o	41	42	43	44	45	46	47	48
Broj lopatice	78	84	93	82	56	3o	39	6o	63	25	74	8	38	94	55	69
Statički moment	31	0	61	- 9	-11	35	-17	-113	-39	-317	-23	-145	-53	21	5	5
Broj mjesta	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Broj lopatice	95	96	47	67	2	35	57	1	15	26	2o	29	33	7	5	14
Statički moment	51	31	-191	15	109	- 6	5	0	-281	470	-31	-142	13	420	-411	-341
Broj mjesta	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
Broj lopatice	34	36	22	41	44	46	50	49	64	58	27	48	10	21	31	42
Statički moment	-625	21	176	-259	-121	- 9	42	15	488	- 2	79	81	61	-46	-152	15
Broj mjesta	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
Broj lopatice	61	6	9	24	40	45	3	18	62	65	66	68	43	7o	71	76
Statički moment	17	-179	-301	-412	11	475	-309	0	-17	5	25	41	91	-26	22	75

LITERATURA:

1. Golenko B.I. - Statističesnie modeli v upravlenii proizvodstvom. Statistika 1973.g.
2. Pomanarenko L.D. - Ob ispolzovanii alfavitnoj metriki v zadaćah razmešćenija geometričesnih objektov. Kiev 1973.
3. J.Lončar - O jednoj metodi proračuna optimalnog balansa. Informatika br.3 Vol.5 Ljubljana, 1981. god.
4. J.Lončar - Primjena inverzione metrike za proračun optimalnog balansa. Informatika br. 3 Vol. 5, Ljubljana 1981. god.
5. J.Lončar - Primjena transpozicione metrike za proračun optimalnog debalansa. Informatica, br.4, Vol.5, str.32, Ljubljana 1981 god.

INFORMATICA 4/1981

SIMULACIONI MODEL ZA EVALUACIJU PERFORMANSI RAČUNARSKIH SISTEMA

N. HADJINA
V. ČERIĆ

UDK: 519.682.6

SVEUČILIŠNI RAČUNSKI CENTAR, ZAGREB

U radu je opisan diskretni simulacioni model za evaluaciju performansi računarskih sistema. Model omogućuje usporedbu rada različitih konfiguracija sistema, detekciju i otklanjanje uskih grla u sistemu, te usporedbu rada istog sistema sa različitim mješavinama radnog opterećenja sistema. Model je programiran u jeziku GPSS, te verificiran na realnom sistemu UNIVAC 1100.

A SIMULATION MODEL FOR EVALUATION OF COMPUTER SYSTEMS PERFORMANCES

The paper includes a description of the discrete simulation model for evaluation of computer systems performances. The model enables the comparation of performances for various systems configurations, detection and elimination of systems bottlenecks and comparation of performance of the same system with various mixes of system workloads. The model is programmed in GPSS language, and its verification is done on the real UNIVAC 1100 system.

1. UVOD

Računarski sistemi su vrlo složeni sistemi, pa je i evaluacija njihovih performansi složen zadatak. Cilj evaluacije performansi računarskih sistema treće generacije obično uključuje maksimiziranje propusnosti uz odgovarajuća ograničenja na vrijeme odziva sistema. Sistemi za obradu u realnom vremenu (Real-time systems) zahtijevaju završetak obrade do nekog vremena, nakon kojeg je završetak obrade beskoristan. Mnogi veliki računarski sistemi nastoje minimizirati vrijeme odziva na račun zakašnjenja u obradi drugih poslova. Poslovi za obradu u interaktivnim sistemima (Time sharing) zahtijevaju odziv unutar nekoliko sekundi, dok neki poslovi za serijsku obradu (Batch systems) mogu zahtijevati završetak obrade unutar nekoliko minuta, nekoliko sati ili čak nekoliko dana. Neki sistemi moraju garantirati i minimalno vrijeme pristupa do nekih sredstava računarskog sistema nakon što su uzeti u obradu. Takvi sistemi osiguravaju nekoliko nivoa pristupa, gdje svaki nivo garantira pristup sredstvu, vrijeme odziva i cijenu rasporedivanja za korištenje tog sredstva[1].

Zadovoljavanje zahtjeva korisničkih poslova na sredstva računarskog sistema predstavlja primarni zadatak operacionog sistema. Performansa operacionog sistema ili računarskog sistema se izjednačuje sa izvršenom obradom nad korisničkim poslovima uz zadovoljavanje ograničenja na vrijeme odziva sistema. Performansa računarskog sistema, u tom smislu, zavisi od četiri različita aspekta sistema i njihovog rada:

1. Raspoloživa sredstva sistema,
2. Organizacija ovih sredstava unutar sistema,
3. Zahtjevi radnog opterećenja na ta sredstva,
4. Strategija pridjeljivanja sredstava sistema.

Mogućnost provođenja ekonomične procjene (predviđanja) performanse sistema kao funkcije prethodno navedenih aspekata vrlo je korisna. Tako npr. neki načini poboljšanja rada postojećih sistema mogu biti provedeni variranjem raspoložive količine postojećih sredstava sistema, organizacijom postojećih sredstava, kontroliranjem radnog opterećenja i procedurama pridjeljivanja sredstava unutar operacionog sistema. Posupak evaluacije moguć je izgradnjom i analizom rada hipotetskih sistema (modela) te usporedivanjem performansi različitih sistema kod obrade istog radnog opterećenja. Model pridjeljivanja sredstava računarskog sistema koji je ovdje opisan treba omogućiti predviđanje performansi sistema kao funkcije opisa gore navedenih aspekata operacija sistema.

2. OPIS RAČUNARSKOG SISTEMA

Struktura sistema

Osnovne karakteristike rada računarskog sistema koji će ovdje biti razmatran prikazane su na slici 1. Svaki posao koji se obrađuje na sistemu opisan je skupom zadataka. Zadatak predstavlja konstantan skup zahtjeva za sredstvima, te se naziva aktivnim ukoliko njegovi zahtjevi mogu biti zadovoljeni. Struktura posla određuje poredak u kojem zadaci mogu biti zadovoljeni, tj. aktivni.

Operacioni sistem (OS) sastoji se od skupa OS funkcija koje pridružuju sredstva zadacima i odgovaraju na prekide u radu sistema. Prekidi u radu sistema odgovaraju referenciranju procesa nepristupačnoj informaciji ili završetku intervala za vrijeme kojega je sredstvo bilo pridjeljeno nekom zadatku. Pozivom funkcije operacionog sistema, koja unosi značajan utrošak sistemskog vremena, iniciraju se sistemske radnje. Zahtjevi sistemskog posla (sistemski "overhead") mogu se opisati na isti način kao i zahtjevi korisničkog posla.

Korisnički poslovi se izabiru iz radnog opterećenja prema kategorijama koje se razlikuju prema prioritetu, nivou željene usluge, te prema izvoru posla.

Usporedba performansi između različitih sistema koji obrađuju isto radno opterećenje zahtijeva opis sistema koji je nezavisan od karakteristika radnog opterećenja. Da bi se to postiglo, u modelu na slici 1 provedeno je odjeljivanje modela zahtjeva na sredstva od modela računarskog sistema. Parcijalni raspored zadataka posla odražava strukturu posla i odnosi se u biti na izvođenje programa. Stvarni poredak zadovoljavanja zahtjeva određen je samim sredstvima sistema, te strategijom pridjeljivanja. Ovu nezavisnost strukture zahtjeva i strukture sredstava teže je postići sa djeljivim sredstvima, kao što je npr. memorija, tj. zahtjev korisničkog posla za memorijom ne može zavisiti o tome da li se memorija pridjeljuje u stranicama fiksne veličine ili u blokovima varijebilne veličine.

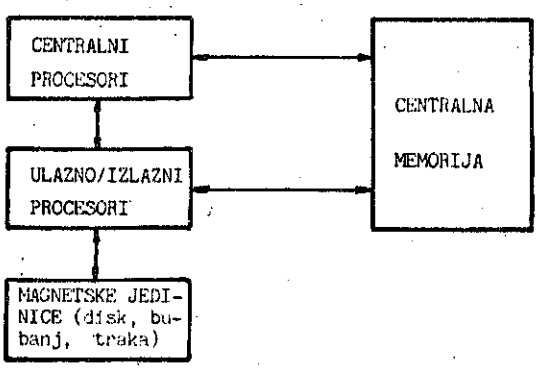
Način rada sistema

U ovom radu promatramo neinteraktivni, multiprogramski i multiprocesorski računarski sistem sa slijedećim sklopovskim komponentama (sredstvima) kao na slici 2. Ostali uzlazno/izlazni uređaji (čitači, štampači i dr.) neće se razmatrati u ovom modelu, jer je evaluacija tih komponenti puno jednostavnija od evaluacije navedenih sredstava računarskog sistema.

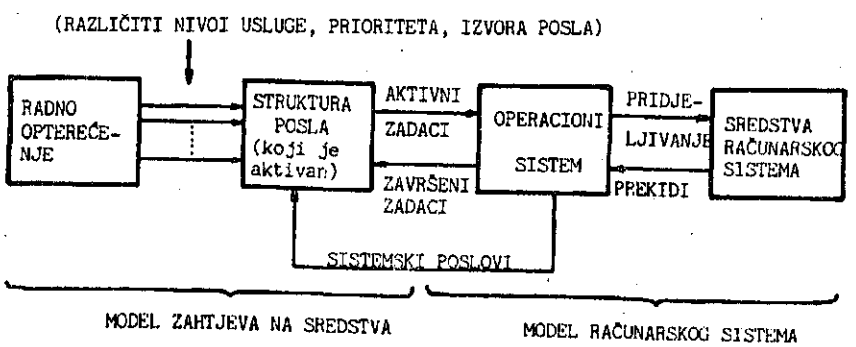
Radno opterećenje čine svi programi učitani u sistem, tj. oni koji čekaju na izvođenje obrade. Svaki program sastoji se od skupa zadataka koji predstavljaju samostalnu cjelinu koja se može izvesti u računarskom sistemu. Programi su pohranjeni na vanjsku memoriju (disk, bubanj) , te se ovisno o prioritetu i veličini raspoloživе centralne memorije izvlače iz vanjske memorije i pohranjuju u centralnu memoriju, kako bi bili spremni za izvođenje.

Multiprogramski nivo određen je veličinom raspoložive memorije i distribucijom zahtjeva za centralnom memorijom. Svi napunjeni zadaci se takmiče za vrijeme centralnog procesora. U računarskim sistemima koji rade na raspodjeli vremena, a takvih je većina, niti jedan zadatak ne dobiva odjednom kvantum zahtijevanog vremena procesa, nego dobiva zahtijevano vrijeme u inkrementima. Po svakom utrošenom inkrementu vremena centralnog procesora ponovo se ispituje prioritetna lista za dodjelu centralnog procesora. Kad god zadatak izvodi ulazno/izlazni zahtjev na magnetsku periferiju oslobađa se centralni procesor, a kad zadatak završi svoju obradu oslobađa se i zauzeta memorija koja je bila pridjeljena zadatku. Budući da u sistemu može postojati više od jednog zadatka može se pojaviti slučaj da neki zadatak zahtijeva sredstvo koje nije trenutačno raspoloživo. Dakle, mora postojati rep čekanja za svako sredstvo, tj. za centralne procesore, centralnu memoriju, magnetsku periferiju (disk, bubanj, traka). Kad god neki zadatak zahtijeva sredstvo koje se već koristi ili raspoloživi prostor sredstva nije dovoljan, zadatak ulazi u odgovarajući rep čekanja. Zadatak može biti samo u jednom repu čekanja istovremeno, i on se ne izvodi dok je u repu.

Strategija dodjele sredstava računarskog sistema, kao i prioriteta zadovoljavanja zahtjeva, ugrađena je u sam operacioni sistem. Prioriteti izvođenja zadataka mogu biti nametnuti i od strane radnog opterećenja sistema.



Slika 2. Sredstva računarskog sistema



Slika 1. Osnovne karakteristike računarskog sistema

Radno opterećenje

Radno opterećenje instalacije dobiva se najčešće snimanjem utroška sredstava sistema u nekom periodu. Iz mjerenih rezultata statističkim metodama određuju se funkcije distribucije vremena pristizanja i veličine zahtjeva. Za svaku radnu instalaciju može se odrediti broj karakterističnih grupa zadataka koje su karakterizirane specifičnim zahtjevima na sredstva sistema. Npr. to mogu biti memorijski zadaci, U/I zadaci i različite mješavine jednih i drugih. Ovu problematiku treba posebno razmatrati, što nije predmet ovog rada.
Svaki zadatak obavezno zahtijeva centralni prostor i memoriju, a po potrebi i neku od magnetskih jedinica.

3. SIMULACIONI MODEL

Osnovne karakteristike simulacionog modela su slijedeće:
Radno opterećenje sistema predstavljeno je kao mješavina tri tipa zadataka: znanstveni, razvojni i poslovni, od kojih svako ima svoju raspodjelu pojavljivanja i zahtjeve za sredstvima računarskog sistema.
Zadatak (task) je zaokružena cjelina za obradu, i ne dijeli se u finije elemente. Zadatak ima različite zahjeve za CPU, memorijom, perifernim jedinicama i sl. U simulacionom modelu zadatak je prikazan pomoću transakcije koja je pokretač svih događaja u modelu.
Sistem se sastoji od primarne i proširene memorije, te dva ili više CPU procesora. Zadaci koji su u primarnoj memoriji koriste CPU u jednom vremenskom odsječku, dok zadaci iz proširene memorije koriste CPU u prekidnom načinu rada. Zadaci iz primarne memorije su višeg prioriteta od ovih u proširenoj memoriji, tako da u slučaju konflikta zadatak iz proširene memorije mora čekati na korištenje svog slijedećeg dijela potrebnog CPU vremena. Ukoliko su i primarna i proširena memorija zauzete kada zadatak postavi zahtjev na njih, tada se zadatak veže u tzv. korisnički lanac. Do aktiviranja tog zadatka dolazi u času kada neki od prethodnih zadataka napušta glavnu ili proširenu memoriju. Zadaci iz korisničkog lanca aktiviraju se po slijedećem rasporedu: prvi idu zadaci koji koriste najviše memorije (uz uvjet da traže najviše onoliko memorije koliko u tom času ima na raspolaganju), a među njima idu prvi oni koji su prvi vezani u lanac.
Na slici 3 prikazan je blok dijagram rada simulacionog modela.

Najznačajniji dio modela, oponašanje dodjele, zauzeća i izvođenja zadataka u primarnoj i proširenoj memoriji prikazan je u Dodatku 1.

Model omogućuje variranje slijedećih parametara:
a) Radno opterećenje
   - učestalost pristizanja zadataka,
   - distribucija zahtjeva za resurse sistema (CPU, memorije diska, bubnja, trake),
   - odnos tipova opterećenja.
b) Parametri konfiguracije sistema
   - broj CPU-ova (1-6),
   - veličina proširene memorije (131K-524K),
   - veličina primarne memorije (32K-160K),
   - veličina raspoložive memorije za taskove (kapacitet STORAGE-a), umanjena je za memoriju potrebnu za operacioni sistem,
   - broj kanala za priključak diskova,
   - broj i vrsta jedinice diskova po kanalu,
   - broj i vrsta magnetskih traka po kanalu,
   - broj i vrsta magnetskih bubnjeva po kanalu.

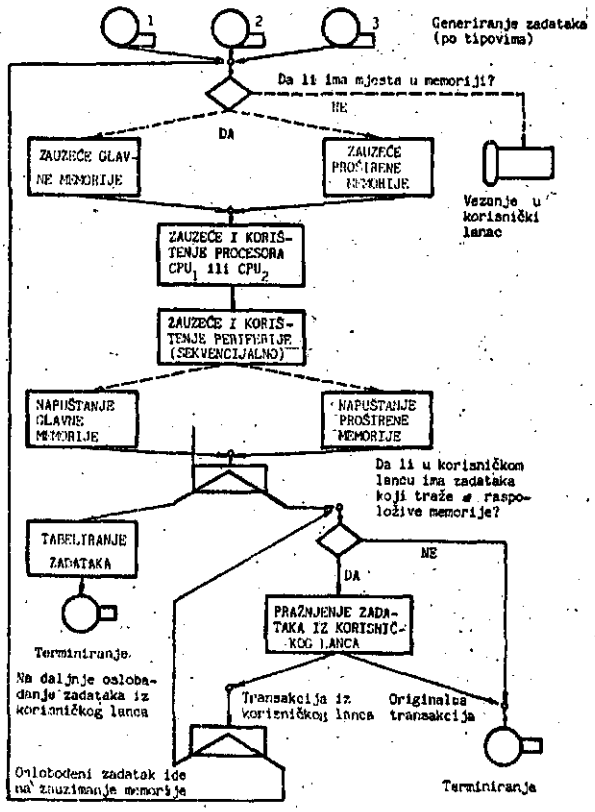
Izlazni rezultati modela obuhvaćaju:
   - iskorištenje CPU-ova,
   - iskorištenje memorije,
   - prosječan broj zadataka u memoriji,
   - iskorištenje kanala,
   - repovi za CPU, memoriju, diskove, trake, bubnjeve.

Vremenska jedinica modela je 1 milisekunda.

4. VREDNOVANJE MODELA

Analiza radnog opterećenja realnog sistema UNIVAC 1100 u Sveučilišnom računskom centru u Zagrebu 1977. godine pokazala je da postoje tri različite grupe zadataka, i to:
   - poslovni ,
   - znanstveno-istraživački,
   - razvoj programa .

Mjerenja su pokazala da se te grupe pojavljuju sa slijedećim postocima: 45%, 36%, 19% (prema redoslijedu navođenja).
Na osnovu rezultata istraživanja[2], mjerenja provedenih na realnom sistemu[3], te rezultata dobivenih za utrošak sistemskog vremena operacionog sistema dobivenih od UNIVAC-a, sintetizirano je radno opterećenje od 300 zadataka, tako da maksimalno aproksimira radno opterećenje upotrebljeno za mjerenje u radu[2].
Primjer izlaznih rezultata rada simulacionog modela za konfiguraciju prikazanu na slici 4 dan je u Dodatku 2.
U izlaznim rezultatima prikazano je korištenje opreme (FACILITY, STORAGE), stanje repova (QUEUE) pred sredstvima za posluživanje (oprema), stanje korisničkih lanaca (USER CHAIN) prikazanih u opisu simulacionog modela, te osnovni podaci o tabelama (TABLE) vremena prolaza kroz različite grupe zadataka kroz pojedine dijelove računarskog sistema.

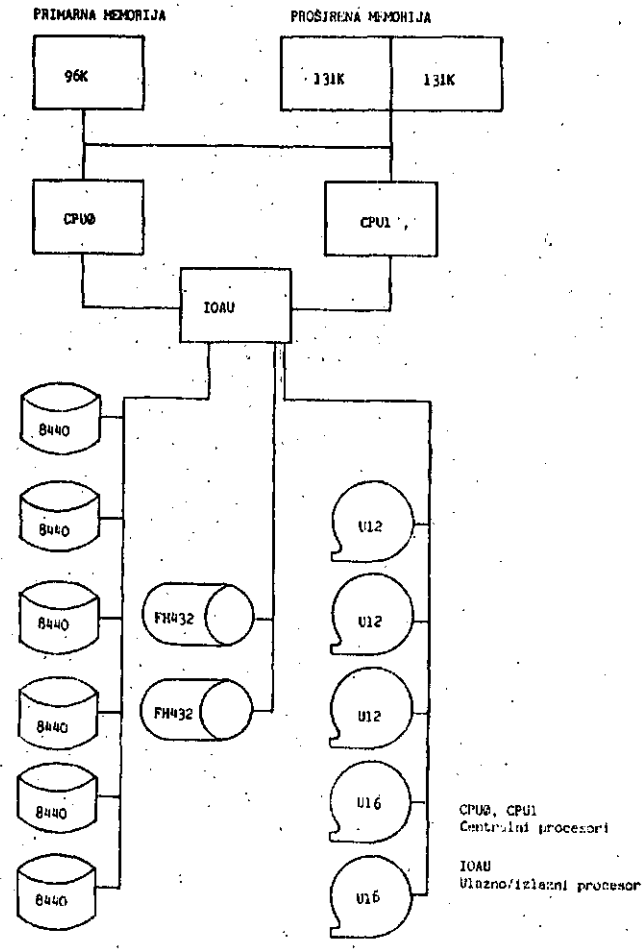


Slika 3. Blok dijagram rada simulacionog modela

Napravljena su mjerenja slijedećih veličina, koje usporedujemo sa rezultatima dobivenim simulacijom.

	Vrijeme izvođenja (minute)	Prosječan broj zadataka u memoriji	Iskorištenje procesora	Iskorištenje memorije	
				Ukupne	Proširene
Mjerenje	18' 22"	7,50	93%	89,90%	89,30%
Simulacija	18' 25"	7,88	94,5%	88,5%	90,1%

Vidi se da postoji vrlo dobro slaganje rezultata što ukazuje na to da model vrlo dobro opisuje računarski sistem.



Slika 4. Konfiguracija računala UNIVAC 1110 (centralni kompleks)

5. ZAKLJUČAK

Diskretni simulacioni model, opisan u ovom radu, može poslužiti kao dobar alat za evaluaciju performansi računarskih sistema. Model je, zbog potrebe vrednovanja, prilagođen sistemu UNIVAC 1100, ali se može prilagoditi i za ostale sisteme sa srodnim karakteristikama.
Za izvođenje simulacije na sistemu UNIVAC 1100/42 potrebno je reda veličine 1 minuta za 100 zadataka, uz zauzeće od oko 50K riječi memorije.
Predviđaju se daljnja poboljšanja modela sa ciljem detaljnog opisa korištenja centralne memorije.

DODATAK 1

```
*
*  GET.MEM  MACRO      OPONAŠANJE  DODJELE,
*     ZAUZEĆA  I  IZVODENJA  PROGRAMA  U
*     GLAVNOJ  I  EXTENDIRANOJ  MEMORIJI.
*        PARAMETRI:
* A= TIP ZADATKA(SCI,BUS,PRO(),
* B=VELIČINA MEMORIJE KOJU ZAHTIJEVA ZADATAK,
* C= ZAHTIJEVANO VRIJEME CPU-A ZA IZVODJENJE
*              ZADATKA U GLAVNOJ MEMORIJI,
* D= ZAHTIJEVANO VRIJEME ZADATKA ZA I/O,
* G= OZNAKA MIXA (SCI,BUS,PRO),
* H= ZAHTIJEVANO VRIJEME CPU-A ZA
*   IZVODENJE ZADATKA U PROŠIRENOJ MEMORIJI.
*
GET.MEM   STARTMACRO
      ASSIGN  MNAME,XXX=G
      ASSIGN  CNAME,XAAX=G
      ASSIGN  PJOB,TYPE,=A
      ASSIGN  MEMORY,FN$=P
      ASSIGN  CAU,VS=C
      ASSIGN  CUU,VS=H
      ASSIGN  IOU,VS=D
*
*  ODREDIVANJE PRIORITETA  ZADOVOLJAVANJA
*  ZAHTJEVA ZA GLAVNU I PROŠIRENU MEMORIJU
*  ORIG = USER CHAIN ZA GLAVNU MEMORIJU
*         SORTIRAN PO SCAU
*  GIRO = USER CHAIN ZA PROŠIRENU MEMORIJU
*         SORTIRAN PO SCAU
*  SVAKA TRANSAKCIJA ULAZI I U JEDAN
*  I U DRUGI CHAIN PA SE OSLOBADJA
* OVISNO O VELIČINI RASPOLOŽIVE MEMORIJE.
*  PO ZADOVOLJENJU ZAHTJEVA ZA MEMORIJOM
*  OTPUŠTA SE TA TRANSAKCIJA IZ DRUGOG CHAIN-A
*
      ASSIGN  MCPUC,(P$CAU)*(.737/1000)
      ASSIGN  ECPUC,(P$CUU)*(.737/1000)
      ASSIGN  ARTC,((P$CAU+P$IOU)*P*MEMORY*
+(20340/(30*20*3600*1000)))*5000
      ASSIGN  ERTC,((P$CUU+P$IOU)*P*MEMORY*
**(2698.4/(30*20*3600*1000)))*5000
   ASSIGN  SSCAU,-(P1MRTC/(P$ERTC-P1MRTC))*1E
      ASSIGN  SCAU,-((P$ERTC-P$MRTC)/P$MRTC)*
**1000  GOTO(+4)
T1C=G  LINK,U  ORIG,P$SCAU
T9=G   LINK,U  GIRO,P$SSCAU
       SPLIT  1,T9=G  GOTO(T1C=C)
*
*  ZAUZEĆE  GLAVNE ILI EXTENDIRANE MEMORIJE
*
      ADVANCE  GOTO(+1,+5,-1)
XXX=G  ENTER  CPSTORE,PSMEMORY
     INQUEUE  QPEM,PPEM    D FJERE JE BROJA
*             ZADATAKA U MEMORIJI
       PRIORITY  1
       MARK  GOTO(+4)
XAAX=G ENTER  CPSTORE,PSMEMORY
     INQUEUE QPEM,PPEM
       MARK  GOTO(+2)
       ASSIGN MEM.GOT,1      GOTO(+2)
       ASSIGN MEM.GOT,2
*
*  OTVARANJE REDA ZA PROCESORE
*  MCPU= BROJ KONFIGURIRANIH CPU-OVA,
*  ECPU= BROJ AKTIVNIH ZADATAKA NAD CPU-OM
*        ZA PROŠIRENU MEMORIJU,
*  MLCPU= BROJ AKTIVNIH ZADATAKA NAD CPU-OM
*         ZA GLAVNU MEMORIJU.
*
      INQUEUE CPU,U,PROCESSOR
      QUEUE  ECPU
=F    COMPARE X*MECPU LT 2
      SAVEX  MECPU,XSHECPU+1
      ADVANCE  GOTO(+1,+3)
      COMPARE P$PEM,COT EQ 1  GOTO(+2)
      COMPARE P$MEM,COT EQ 2  GOTO(+2)
      SAVEX  MCPU,XSHCPU+1
      ADVANCE  GOTO(+1 THRU +3)
      COMPARE P*PJOB.TYPE EQ 1  GOTO(+2)
      COMPARE P*PJOB.TYPE EQ 2  GOTO(+11)
*
*  ZAUZEĆE PROCESORA OVISNO JE O VRSTI
*  MEMORIJE U KOJU JE SMJEŠTEN ZADATAK,
*  O  TIPU ZADATKA(BUS,SCI,PRO),
*  TE O ZAUZEĆU PROCESORA (1 ILI 2).
*  -   KONFIGURACIJA IMA 2 PROCESORA  -
*
*  ZAUZEĆE  PROCESORA ZA IZVODENJE U GLAVNOJ
*  MEMORIJI
*
      ADVANCE  GOTO(+1  THRU +2)
HOLD  CPU1  TIME(VSPRO.MAIN.TIM)   GOTO(+14)
HOLD  CPU2  TIME(VSPRO.MAIN.TIM)   GOTO(+13)
HOLD  CPU3  TIME(VSPRO.MAIN.TIM)   GOTO(+12)
HOLD  CPU4  TIME(VSPRO.MAIN.TIM)   GOTO(+11)
      ADVANCE  GOTO(+1 THRU +2)
HOLD  CPU1  TIME(VSSCI.MAIN.TIM)   GOTO(+9)
HOLD  CPU2  TIME(VSSCI.MAIN.TIM)   GOTO(+8)
HOLD  CPU3  TIME(VSSCI.MAIN.TIM)   GOTO(+7)
HOLD  CPU4  TIME(VSSCI.MAIN.TIM)   GOTO(+6)
      ADVANCE  GOTO(+1 THRU +2)
HOLD  CPU1  TIME(VSBUS.MAIN.TIM)  GOTO(+4)
HOLD  CPU2  TIME(VSBUS.MAIN.TIM)  GOTO(+3)
HOLD  CPU3  TIME(VSBUS.MAIN.TIM)  GOTO(+2)
HOLD  CPU4  TIME(VSBUS.MAIN.TIM)  GOTO(+1)
         SAVEX  MCPU,XSHCPU-1
         SAVEX  MECPU,XSHECPU-1    GOTO(+36)
*
*  -   ZAUZEĆE PROCESORA ZA IZVODENJE U
*         PROŠIRENOJ MEMORIJI  -
*  MAKSIMALNI BROJ ZADATAKA U PROŠIRENOJ
*  MEMORIJI  LLDH=10 , ZA SVAKI ZADATAK
*  DOBIVA DESETINU TRAŽENOG VREMENA .
*
         SAVEX  ECPU,XSECPU+1
         ADVANCE   GOTO(+1 THRU +3)
         COMPARE   P*PJOB.TYPE EQ 1  GOTO(+1*)
         COMPARE   P*PJOB.TYPE EQ 2  GOTO(+75)
         ASSIGN  PCPUTIME,VSPRO.EXT.TIM
         ASSIGN  LLDH,10
      ADVANCE   GOTO(+1 THRU  +2)
HOLD  CPU1  TIME(P*PCPUTIME/10)   GOTO(+4)
HOLD  CPU2  TIME(P*PCPUTIME/10)   GOTO(+3)
HOLD  CPU3  TIME(P*PCPUTIME/10)   GOTO(+2)
HOLD  CPU4  TIME(P*PCPUTIME/10)   GOTO(+1)
         SAVEX  ECPU,XSECPU-1
      ASSIGN  LLDH,P*LLDH-1
         SAVEX  MECPU,XSECPU-1
         ADVANCE  GOTO(+1,+71)
      COMPARE  P*LLDH NE  0
      LOOP
         QUEUE  ECPU
         COMPARE  X*MECPU  LT  2
         SAVEX   MLCPU,XSLCPU +1
         SAVEX   ECPU,XSECPU+1     GOTO(-14)
         ASSIGN  SCPUTIME,VSSCI.EXT.TIM
      ASSIGN  LLDH,10
      ADVANCE  GOTO(+1 THRU  +2)
HOLD  CPU1  TIME(P*SCPUTIME/10)   GOTO(+4)
HOLD  CPU2  TIME(P*SCPUTIME/10)   GOTO(+3)
HOLD  CPU3  TIME(P*SCPUTIME/10)   GOTO(+2)
HOLD  CPU4  TIME(P*SCPUTIME/10)   GOTO(+1)
         SAVEX  ECPU,XSECPU-1
         SAVEX  MLCPU,XSLCPU-1
      ASSIGN  LLDH,P*LLDH-1
         ADVANCE  GOTO(+1,+24)
      COMPARE  P*LLDH NE  0
      LOOP
```

```
        QUEUE   CCPU
        COMPARE X$MECPU  LT  2
        SAVEX   MECPU,X$MECPU+1
        SAVEX   ECPU,X$ECPU+1   GOTO(-14)
    ASSIGN  BCPUTIME,V$BUS.EXT.TIM
     ASSIGN  LLDH,10
     ADVANCE GOTO(+1 THRU +2)
HOLD    CPU1-TIME(P$BCPUTIME/10)    GOTO(+4)
HOLD    CPU2 TIME(P$BCPUTIME/10)    GOTO(+3)
HOLD    CPU3 TIME(P$BCPUTIME/10)    GOTO(+2)
HOLD    CPU4 TIME(P$BCPUTIME/10)    GOTO(+1)
        SAVEX   ECPU,X$ECPU-1
        SAVEX   MECPU,X$MECPU-1
     ASSIGN  LLDH,P$LLDH-1
        ADVANCE   GOTO(+1,+7)
     COMPARE  P$LLDH  NE  0
     BUFFER
        QUEUE   CCPU
        COMPARE X$MECPU LT 2
     SAVEX   MECPU,X$MECPU+1
     SAVEX   ECPU,X$ECPU+1      GOTO(-14)
     OUTQUEUE   CPU.Q,PROCESSOR  GOTO(PERIF)
        ENDMACRO
```

DODATAK 2

FACILITY NAME	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRANS
CPU1	.9611	431	2466.64
CPU2	.9309	443	2324.52
CPU3	.0000	0	.00
CPU4	.0000	0	.00
D432.1	.0496	96	571.11
D432.2	.0493	94	579.74
L8440.1	.2048	26	8714.73
CH.1	.4046	190	2355.70
L8440.2	.2652	33	8888.73
L8440.3	.2248	28	8879.54
L8440.4	.2166	27	8675.70
L8440.5	.2210	33	7408.36
L8440.6	.3841	43	9881.07
DTAPE.1	.0670	28	2647.68
DTAPE.2	.1058	45	2600.07
DTAPE.3	.0973	38	2851.76
DTAPE.4	.0602	36	1849.28
DTAPE.5	.0648	43	1665.86

STORAGE NAME	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	TOTAL TRANS
MMSTORE	67	56.46	2944	114
EMSTORE	204	184.36	2525	76

QUEUE NAME	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES
QMEM	12	7.80	190
CPU.Q	12	5.63	190
CCPU	10	3.74	874
Q432.1	1	.00	96
Q432.2	1	.00	94
Q8440.1	1	.01	26
Q8440.2	2	.05	33
Q8440.3	3	.05	28
Q8440.4	1	.02	27
Q8440.5	2	.06	33
Q8440.6	2	.04	43
QTAPE.1	1	.00	28
QTAPE.2	1	.01	45
QTAPE.3	1	.00	38
QTAPE.4	1	.00	36
QTAPE.5	1	.00	43

USER CHAIN NAME	MAXIMUM LENGTH	AVERAGE LENGTH	TOTAL ENTRIES
ORIG	11	9.45	139
GIRO	11	9.45	139

TABLE NAME	WEIGHTED NO. OF ENTRIES	WEIGHTED MEAN ARGUMENT
ALL.JOBS	190	45050.955
MAIN.MEM.JOB	114	21645.219
EXT.MEM.JOB	76	82159.565
SCI.JOB	87	46373.356
BUS.JOB	64	48891.766
PROG.JOB	39	39787.025
WAIT.CPU	190	20017.442

LITERATURA:

1. Ferrari, D., Computer System Performance Model, Prentice-hall, 1978.
2. Cvitaš, V., Analiza iskorištenja multiprogramskih sistema, magistarski rad, Elektrotehnički fakultet u Zagrebu, 1977.
3. Hadjina, N. i Gaćeša M., Mjerenje utjecaja komponenti sistema UNIVAC 1100 na propusnost u batch modu, Informatica, Bled, 1977.

INFORMATICA 4/1981

MEHURČNI POMNILNIKI – IV DEL

J. ŠILC
B. MIHOVILOVIĆ
P. KOLBEZEN

UKD: 681.327.664.4

INŠTITUT JOŽEF STEFAN, LJUBLJANA

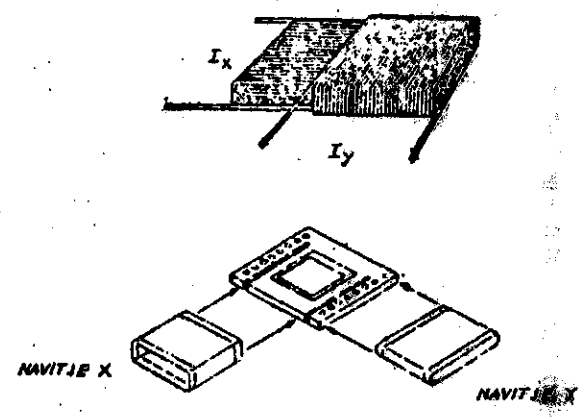
Zadnji v seriji člankov o mehurčnih pomnilnikih nas seznanja z mehurčnimi pomnilniškimi sistemi. Njihova zgradba je zelo kompleksna, saj vsebuje vmesnik za priključitev na sistemsko vodilo računalnika, krmilnik, ki generira potrebne krmilne signale, funkcijski gonilnik za oskrbovanje pomnilniškega elementa z ustreznimi tokovnimi impulzi, ki generirajo, podvajajo in prenašajo magnetne mehurčke, nadalje gonilnik navitij, ki s pomočjo tuljavic generira rotirajoče magnetno polje ter bralno – oblikovalni ojačevalnik za ojačevanje in oblikovanje izhodnega signala pomnilniškega elementa.

MAGNETIC BUBBLE MEMORIES - PART 4. In the last in a series of articles on magnetic bubble memories basic bubble memory system is presented. A typical bubble memory system consists of an interface to the host computer's bus, a controller to produce the necessary control - signal enables, a data buffer, redundancy logic for bad - loop definition and error detection, a function timing generator to control the sequencing and duration of the control pulses required during each bubble shift cycle, function drivers to produce control pulses of the proper amplitude, coil drivers to energize the bubble - shifting coils, and sense amplifiers to detect the presence or absence of a bubble at specific points in the bubble - access cycle.

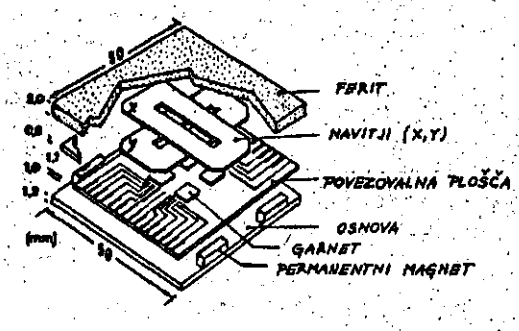
1. UVOD

Mehurčni pomnilniški element nima gibljivih delov, v čemer je njegova glavna prednost pred drugimi pomnilniškimi mediji (tračne enote, diski, gibki diski). Kljub temu pa ima precej zahtevno konstrukcijo

Najpomembnejši sestavni del elementa je pomnilniški čip, ki je sestavljen iz nemagnetne garnetne osnove, magnetnega garnetnega filma, ki je nosilec mehurčkov, vzorcev prevodnih plasti in vzorcev permalojskih plasti, ki definirajo pomnilniške zanke. Podroben opis vseh naštetih komponent je podan v [1,2]. Vertikalno magnetno polje, ki je potrebno za obstoj stabilnih cilindričnih domen, je realizirano s permanentnim magnetom, kar zagotavlja v pomnilniku stalno informacijo (nonvolatility), ki ni odvisna od izpada napajanja. Rotirajoče magnetno polje je realizirano z dvema pravokotno postavljenima ploščatima tuljavama, ki sta napajani s fazno premaknjenima tokovoma trikotniških oblik. (Slika 1.a)
Realizacija in namestitev tuljavic je dokaj zahteven postopek, saj od njega v veliki meri zavisi kvaliteta pomnilniškega elementa. Tuljave omejujejo maksimalno frekvenco rotirajočega polja in povzročajo motnje zaradi induciranih napetosti v vodnikih. Dodatni problem nastopi zaradi segrevanja tuljavic, kar v veliki meri vpliva na karakteristike pomnilniškega elementa. Celoten pomnilniški element je obdan še z magnetnim ščitom, ki preprečuje nezaželjene vplive magnetnih polj na okolico. Zunanje dimenzije sestavljenega elementa, ki ima kapaciteto 10^6 bitov, so 35 x 30 x 10 mm.



Slika 1.a



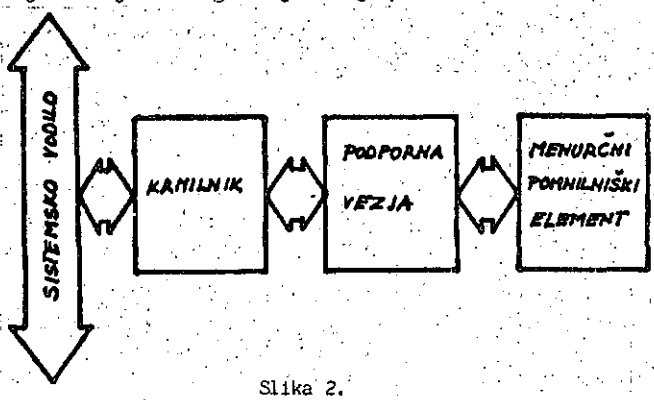
Slika 1.b

Konstrukcija pomnilniškega elementa je prikazana na sliki 1.b.

2. OSNOVNA ARHITEKTURA MEHURČNEGA POMNILNIŠKEGA SISTEMA

Mehurčni pomnilniški sistem (magnetic bubble device) sestavljajo tri osnovne funkcionalne enote in sicer:
- mehurčni pomnilniški element (magnetic bubble memory),
- LSI podporna vezja in
- krmilnik (Bubble memory controller).

Najosnovnejša konfiguracija MPS je prikazana na sliki 2.



Slika 2.

Prvo funkcionalno celoto mehurčnega pomnilniškega sistema predstavlja mehurčni pomnilniški element. Fizikalne principe delovanja pomnilniškega elementa, kakor tudi tehnologijo izdelave smo si ogledali v predhodnih nadaljevanjih, zato si na tem mestu osvežimo le najvažnejše in najznačilnejše lastnosti pomnilniškega elementa.

Kapaciteta. Kapacitete pomnilniških elementov se giblejo od 64 Kbitov do 1 Mbit na čip, vendar lahko trdimo, da je kapaciteta 1 Mbit danes že standardna.

Organizacija. Organizacije današnjih pomnilniških elementov so večinoma v obliki minor zank in major trakov (podvajanje bloka oz. block replication). Starejše izvedbe pomnilniških elementov pa še vedno uporabljajo major/minor zančno organizacijo. Zasledimo pa celo še organizacijo v obliki dolgega premikalnega registra.

Število minor zank se giblje v razredu od 300 do 600, pač odvisno od kapacitete. Število bitov v posameznih minor zankah pa je od 1024 do 4096.

Premer magnetnega mehurčka. LPE garnetni filmi, ki se trenutno uporabljajo, omogočajo stabilne cilindrične domene premera 2 do 3 μm.

Časi dostopa. Povprečni časi dostopa, ki so odvisni od organizacije in kapacitete pomnilniškega elementa, se giblejo v območju od 4 do 40 ms.

Propagacijska frekvenca. Praktično vsi pomnilniški elementi delujejo s propagacijsko frekvenco 100 KHz.

Hitrost prenosa podatkov. Hitrosti prenosa podatkov so v območju od 50 do 150 Kbitov/s.

Vplivi okolice. Magnetni mehurčni pomnilniki delujejo v temperaturnem območju od 0° C do + 50° C. Informacija ostane ohranjena v nekoliko širšem temperaturnem območju in sicer od - 50° C do + 100° C. Pomnilniški elementi brez posledic prenašajo 95% relativno vlažnost in motilna zunanja magnetna polja jakosti 3 do 4 KA/m.

Dimenzije in teža. Zaradi velikih gostot mehurčnih pomnilnikiov (10^6 bitov /cm^2) so tako volumen, kot teža pomnilniških elementov relativno majhni. Dimenzije pomnilniškega elementa so 35 x 30x10 mm, teža pa je približno 40 g.

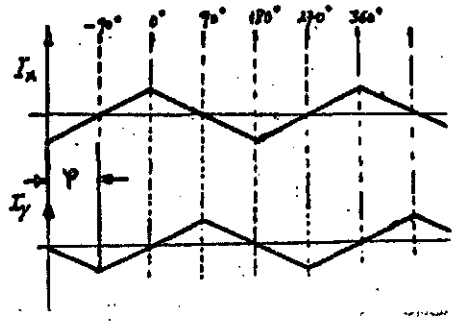
Drugo pomembno funkcionalno celoto mehurčnega pomnilniškega sistema tvorijo LSI podporna vezja.
Podporna vezja sestojijo iz naslednjih elementov:

- gonilnikov navitij (coil drivers)
- funkcijskega gonilnika (function driver) in
- bralno - oblikovalnega ojačevalnika (sense amplifier)

Gonilnika navitij imata nalogo, da tvorita fazno premaknjena tokova trikotniških oblik, ki napajata pravokotno nameščeni X in Y tuljavici ter tako ustvarjata rotirajoče magnetno polje. Povsem jasno je, da je za ustvarjanje rotirajočega magnetnega polja v pravokotno nameščenih tuljavicah potrebno generirati tokove trikotnih oblik, časovno med seboj premaknjene za 90°. (Slika 3.)

Gonilnik navitij je v grobem sestavljen iz dveh delov: logičnega in močnostnega. Logični del gonilnika sprejema logične signale iz krmilnika, pri čemer sta najpomembnejša:

A (XA oz. YA) : začetek naraščanja toka v tuljavici X



Slika 3.

oziroma Y (coil rise enable)

B (XB oz. YB): začetek upadanja toka v tuljavici X oziroma Y (coil fall enable)

Naloga močnostnega dela je, da izoblikuje na podlagi signalov A in B napetost stopničaste oblike, ki napaja tuljavico X oziroma Y. Ker je tok skozi tuljavico definiran z izrazom:

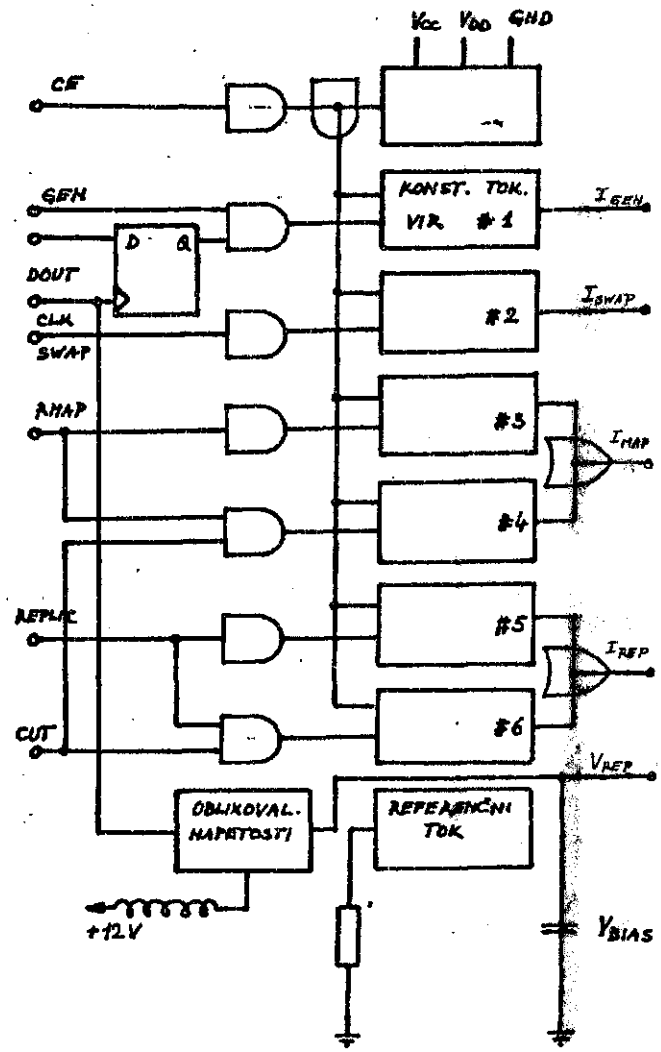
$$i(t) = \frac{1}{L}\int_0^t u(\tau)\,d\tau \qquad (1)$$

je pri stopničasti obliki napetosti, tok linearno naraščajoč odnosno padajoč (odvisno od polaritete napetosti na tuljavici).

Operacije mehurčnega pomnilniškega elementa, kot so generacija mehurčkov, podvajanje, prenos mehurčkov itd., se izvajajo s pomočjo ustrezno amplitudno in časovno oblikovanih tokovnih impulzov. Vezje, ki oskrbuje pomnilniški element s potrebnimi tokovi se imenuje funkcijski gonilnik. Tokovne generatorje znotraj gonilnika krmilijo logični signali iz krmilnika:

GEN :	Omogočitev generacijskega tokovnega impulza
SWAP :	Omogočitev tokovnega impulza za prenos major trak- minor zanke
RMAP :	Omogočitev tokovnega impulza za podvajanje mehurčkov iz kontrolne zanke (pri prenosu kontrola zanka-major trak)
REPLIC:	Omogočitev tokovnega impulza za podvajanje mehurčkov iz minor zank (pri prenosu minor zanka - major trak)
CUT :	Omogočitev tokovnega impulza za pretrganje mehurčka pri podvajanju mehurčkov (pri prenosu minor zanka - major trak, ali kontrolna zanka - major trak)
DOUT :	Podatki, ki se serijsko prenašajo iz krmilnika in vpisujejo v mehurčni pomnilniški element.

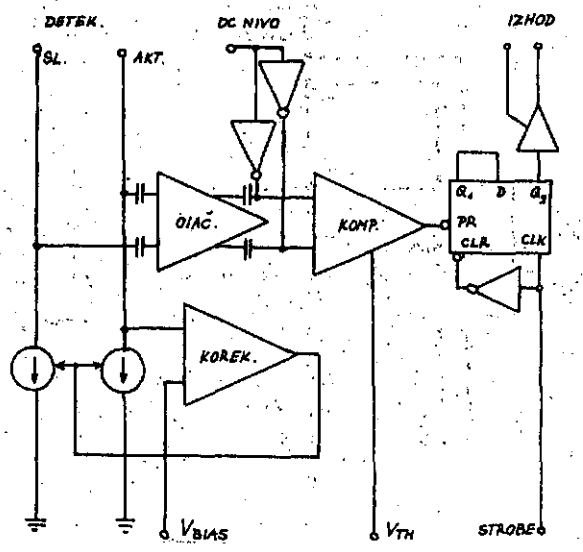
Primer funkcijskega gonilnika prikazuje slika 4.



Slika 4.

Naloga bralno-oblikovalnega ojačevalnika je da pretvarja nizko nivojske signale, ki prihajajo od magnetoreziste-nčnega detektorja mehurčnega pomnilniškega elementa, v TTL kompatibilne izhodne nivoje. Vezje vsebuje ojačevalnik hitri precizni komparator, dva flip-flopa in tri - state izhodna vrata. (Slika 5)

Osnovo bralno-oblikovalnega vezja predstavlja mostiček, ki je v primeru, ko ni prisoten mehurček v ravnotežju za kar skrbi korekcijski ojačevalnik. Prisotnost mehurčka pomeni napetostni signal na vhodu ojačevalnika, katerega le-ta primerno ojači. Predno ojačani signal na komparatorju



Slika 5.

primerjamo z v naprej določenim progom (V_{TH}), le-tega postavimo na določen enosmerni nivo. Pragovno napetost V_{TH} lahko spreminjamo. Časovno uokvirjanje izhodnih podatkov dosežemo z dvema flip-flopoma in logičnim signalom STROBE dobljenim iz krmilnika. Tretjo funkcionalno celoto tvori krmilnik, ki je najkompleksnejša enota mehurčnega pomnilniškega sistema. Krmiljenje funkcij in časovnega poteka je odvisno tako od zgradbe samih komponent MPS, kakor tudi od organizacije mehurčnega pomnilniškega elementa, kar vse se odraža v krmilniku. Običajno izvaja krmilnik kontrolo nad:

- časovnimi poteki sistema
- upravljanjem z vodilom
- redundančnimi zankami
- korekcijo napak
- pretvorbo adres in
- podporo.

Krmilnik je zgrajen tako, da v sebi združuje tako osnovne funkcije potrebne za nemoteno obratovanje mehurčnega pomnilniškega sistema, kakor tudi dodatne opcijske funkcije, ki omogočajo v programski povezavi z okolico pomnilnika gradnjo učinkovitejših sistemov (DMA, Error corection itd.). Funkcije krmilnika so:

- krmiljene gonilnikov navitij in bralno - oblikovalnega ojačevalnika

Del krmilnika, ki oblikuje logične signale za gonilnik navitij (XA, XB, YA, YB) in bralno oblikovalni ojačevalnik (STROBE), sestavlja nekaj logičnih vrat, števcev, izhodnih latch-ov in ROM pomnilnik v katerem je shranjen mikroprogram, ki oblikuje časovni potek teh signalov.

- krmiljenje funkcijskega gonilnika

Prej opisani del krmilnika oblikuje tudi logične signale za funkcijski gonilnik (GEN, SWAP, REPLIC, CUT), katerih časovni potek je tako kot v prejšnjem primeru mikrogramsko oblkovan.

- naslavljanje podatkov, (velikost bloka, formatiranje podatkov)

Z enim krmilnikom je mogoče krmiliti enega ali več mehurčnih pomnilniških elementov. Informacijo o tem koliko pomnilniških elementov je združeno v takoimenovano pomnilniško banko, hrani programljiv register SFR (System Features Register). Krmilnik nam omogoča dostop do ene ali več strani. Adreso željene strani podamo v adresnem registru strani SAR (Sector Address Register). Število strani, ki jih želimo prebrati (pri čemer povemo adreso le prve v seriji strani) pa prav tako podamo v posebnem MSR registru (Multiple Sector Register).

Uporabnik poda ukaze krmilniku preko ukaznega registra CMDR (Command Register), krmilnik pa podaja informacijo o delovanju preko statusnega registra STR (Status Register) v katerem podamo dodatne opcijske funkcije krmilnika (Write protect, Stop if Error Detected,...). Za izravnavanje različnih hitrosti med uporabniškim in pomnilniškim vmesnikom je uporabljen 16bitni FIFO pomnilnik. Naslavljanje kateregakoli od naštetih registrov, kakor tudi FIFO pomnilnika, je mogoče preko adresnih linij.

- detekcija in korekcija napak

Krmilnik vsebuje tudi vezje, ki odkriva in popravlja določeno število napačnih bitov v besedi. Pri vpisu podatkov v pomnilnik, generira krmilnik zaporedje korekcijskih bitov, ki sledijo bloku podatkov. Med čitanjem podatkov iz pomnilnika, posebno vezje preverja vsebino korekcijske besede tako, da lahko odkrije in popravi napako. Še več, uporabniku je dana možnost, da izbira med različnimi učinkovitostmi korekcijskega vezja. Podatkovni blok v katerem se je pojavila napaka ponovno preberemo. V primeru, ko je drugič prebrani blok v redu, govorimo o mehki napaki, če pa vezje ponovno detektira isto napako, govorimo o trdi napaki, ki izvira iz pomnilniškega elementa. V tem primeru krmilnik napako popravi in popravljen blok ponovno zapiše v pomnilnik. Pripomniti velja, da je lahko detekcijsko - korekcijsko vezje združeno v krmilniku ali pa v bralno - oblikovalnem ojačevalniku.

- upravljanje z redundančnimi zankami

Informacija o tem katera minor zanka je pokvarjena je shranjena v kontrolni zanki (Map loop). Vsebina kontrolne zanke se prenese v RAM pomnilnik, ki je vsebovan v krmi-

lniku in na podlagi katerega krmilnik oblikuje format podatkov, tako da upošteva zgolj dobre zanke (defektne nadomesti z redundančnimi).

- prenos podatkov v/iz pomnilniškega sistema

Poleg klasičnega I/O prenosnega mehanizma (prenos podatkov v smeri periferni pomnilniški sistem - hitri pomnilnik in obratno ob angažiranju CPU-ja) omogoča krmilnik priključitev na DMA (Direct Memory Access) krmilnik in s tem gradnjo učinkovitejšega sistema, saj pri samem prenosu podatkov CPU ni udeležen. Da pa je izvajanje DMA prenosa možno je med DMA krmilnikom in krmilnikom mehurčnega pomnilnika vzpostavljena povezava preko ukaznih in statusnih signalov. Vobče poteka vpisovanje in izpisovanje podatkov pri mehurčnem pomnilniškem elementu serijsko po blokih tako, da v grobem lahko ta pomnilniški medij primerjamo s tradicionalnimi elektromehanskimi pomnilniškimi mediji (tračna enota, diski). Vendarle pa obstajajo razlike, ki jih ne moremo in ne smemo zanemariti. Mehurčni pomnilniki omogočajo start-stop na bit natančno in ne potrebujejo za ostale pomnilniške medije značilnih adresirnih pripomočkov kot so optična detekcija začetka in konca, sektorski markerji in podobno.

3. VEČJI MEHURČNI POMNILNIŠKI SISTEMI

Osnovo za gradnjo večjih in sposobnejših mehurčnih pomnilniških sistemov predstavlja pomnilniški modul (PM), ki vsebuje:

- mehurčni pomnilniški element
- podporno opremo (gonilnika navitij, funkcijski gonilnik, bralno- oblikovalni ojačevalnik)

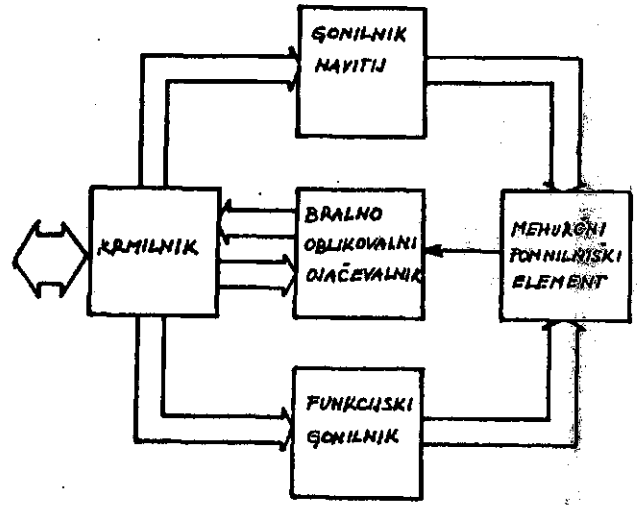
Za delovanje PM je potreben še krmilnik tako,da dobimo minimalen pomnilniški sistem MPS, ki ga prikazuje slika 6.

Povečevanje mehurčnih pomnilniških sistemov je mogoče na več načinov:

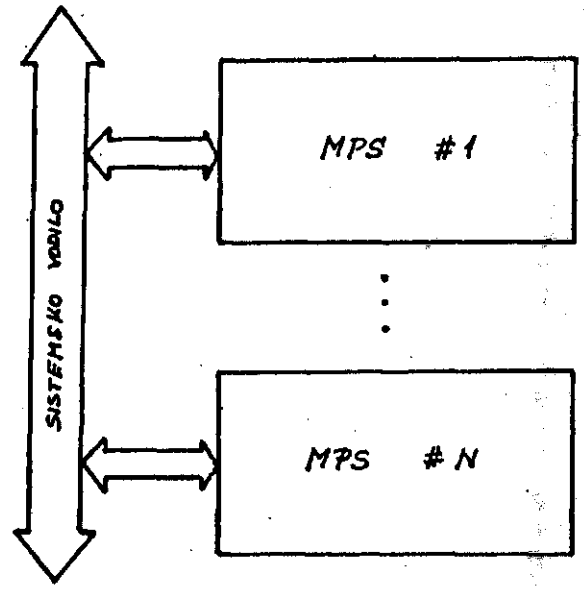
- paralelno povezovanje MPS

Takšna razširjava pomnilniškega sistema omogoča večanje kapacitete, dočim ostaneta povprečni čas dostopa in hitrost prenosa podatkov nespremenjena. Konfiguracijo takšnega sistema prikazuje slika 7. Takoj velja pripomniti, da ima tak pomnilniški sistem nekaj slabih lastnosti, kot so: veliko število krmilnikov, relativno veliko porabo energije in veliko število tiskanin.

- multi modulni sistem



Slika 6.

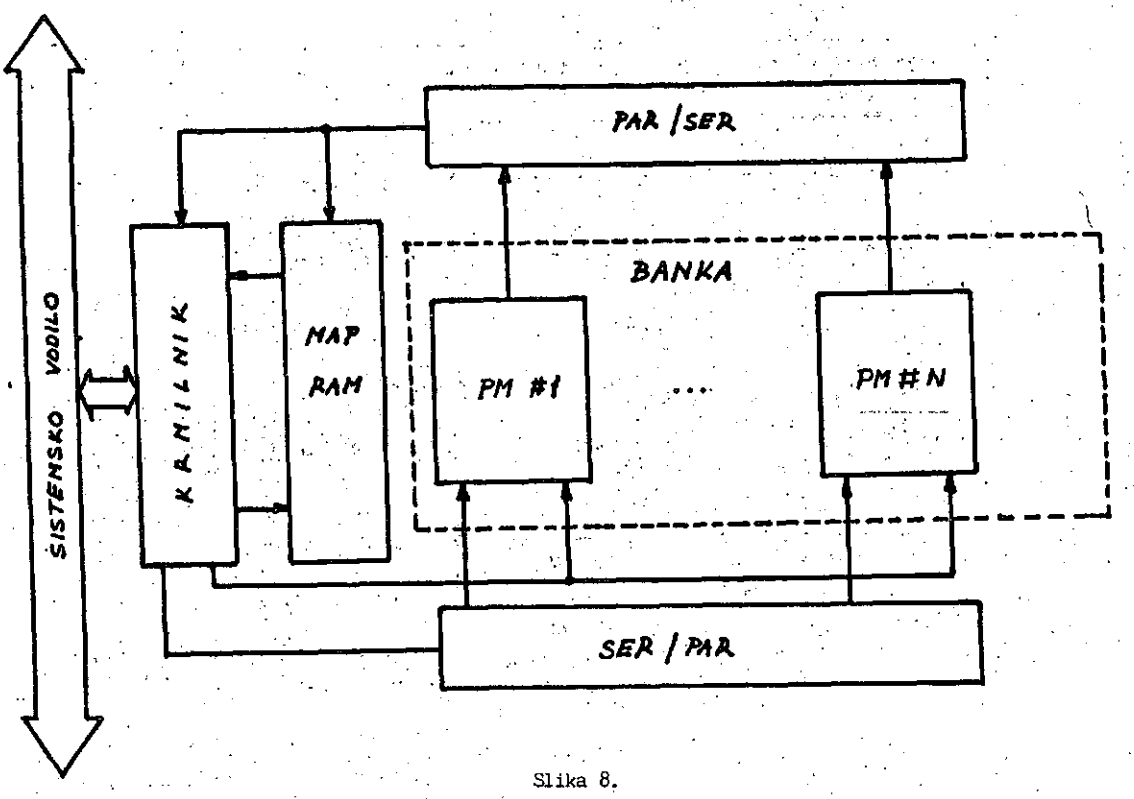


Slika 7.

Če želimo poleg kapacitete povečati tudi hitrost prenosa podatkov, uporabimo konfiguracijo mehurčnega pomnilniškega sistema, ki jo prikazuje slika 8.
Z enim krmilnikom nadzorujemo več paralelno delujočih pomnilniških modulov (banka). Prednost multi modulnega sistema je v tem, da ni potrebna predhodna inicializacija (glej naslednji pomnilniški sistem), kar prinaša krajše čase dostopa. Hitrost prenosa podatkov se napram paralelni vezavi MPS poveča tolikokrat, kolikor pomnilniških modulov vsebuje banka, saj delujejo vsi PM istočasno.
Slaba stran takšnega pomnilniškega sistema je v tem, da je poraba energije relativno velika.

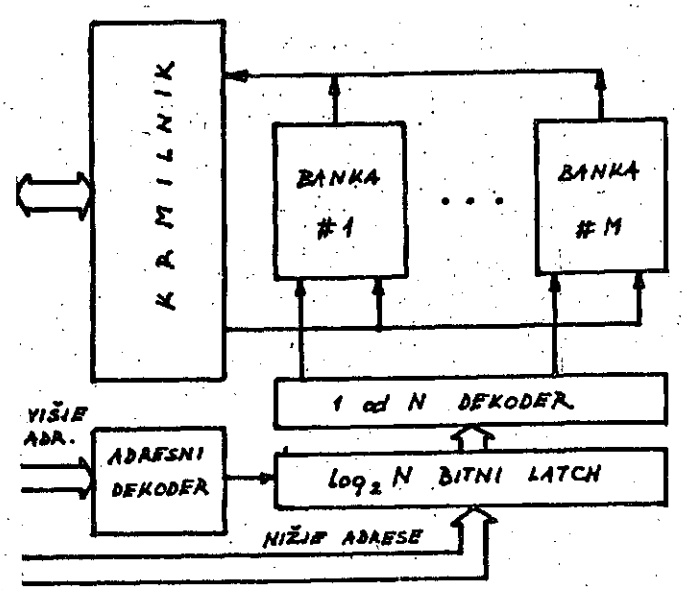
- multi bančni sistem

Obstaja tudi možnost gradnje mehurčnih pomnilniških siste-



Slika 8.

mov z multipleksiranjem večjega števila bank, ki jih krmilimo z enim krmilnikom. Takšna možnost je prikazana na sliki 9.



Slika 9.

Kapaciteta sistema je enaka M krat kapaciteta banke. Hitrost prenosa podatkov je enaka kot je hitrost prenosa podatkov ene banke. Časi dostopa so nekoliko daljši, saj je potrebno pri vsaki izbiri nove banke ažurirati "map" pomnilnik, kar imenujemo inicializacija. Čas inicializacije je reda nekaj deset ms. Velika prednost takšnega sistema je, da poraba energije ne narašča proporcionalno s številom bank, saj je istočasno aktivna le ena banka.

Značilnosti različnih mehurčnih pomnilniških sistemov so povzete v tabeli 1. Pomnilniški sistem pa je mogoče še povečevati tako, da sestavljamo multi modulne ali multi bančne sisteme v kompleksnejši sistem (KPS). Združuje jih posében krmilnik (mikroprocesorski), vmesnik, RAM pomnilnik in DMA krmilnik. Preko internega vodila je mogoče priključiti večje število KPS. Takšen sistem nam omogoča zelo velike kapacitete in hitrosti prenosa podatkov.

4. NEKAJ BESED O PROGRAMSKI OPREMI

Iz same organizacije mehurčnega pomnilniškega elementa [3] je razvidno, da lahko na nivoju programske opreme obravnavamo mehurčne pomnilnike enako kot vse elektromehanske diske. Z minimalno spremembo programske opreme, je mogoče zamenjati disk z mehurčnim pomnilniškim sistemom.

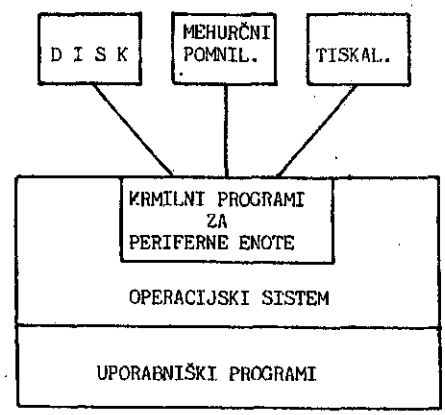
Oglejmo si najprej odnose med posameznimi elementi programske opreme računalniškega sistema. Razmere so prikazane na sliki 10. Vidimo, da lahko z preoblikovanjem krmilnega programa za ustrezno periferno enoto (v našem

	minimalen pomnilniški sistem (MPS)	paralelno povezovanje MPS	multi modulni sistem (MMS)	multi bančni sistem (MBS)
Kapaciteta	C	N x C	N x C	M x N x C
povprečni čas dostopa	ta	ta	ta	ta + ti
hitrost prenosa podatkov	V	V	N x V	N x V
prednosti		- velika kapaciteta	- velika kapaciteta - velika hitrost prenosa podatkov	- zelo velika kapaciteta - Velika hitrost prenosa podatkov - relativno majhna poraba energije
slabosti	- majhna kapaciteta - majhna hitrost prenosa podatkov	- veliko število krmilnikov - majhna hitrost prenosa podatkov	- relativno velika poraba energije	- daljši čas dostopa

N... število minimalnih pomnilniških sistemov oz. pomnilniških modulov
M... število bank
t_i...čas inicialitacije MAP RAM-a

tabela 1.

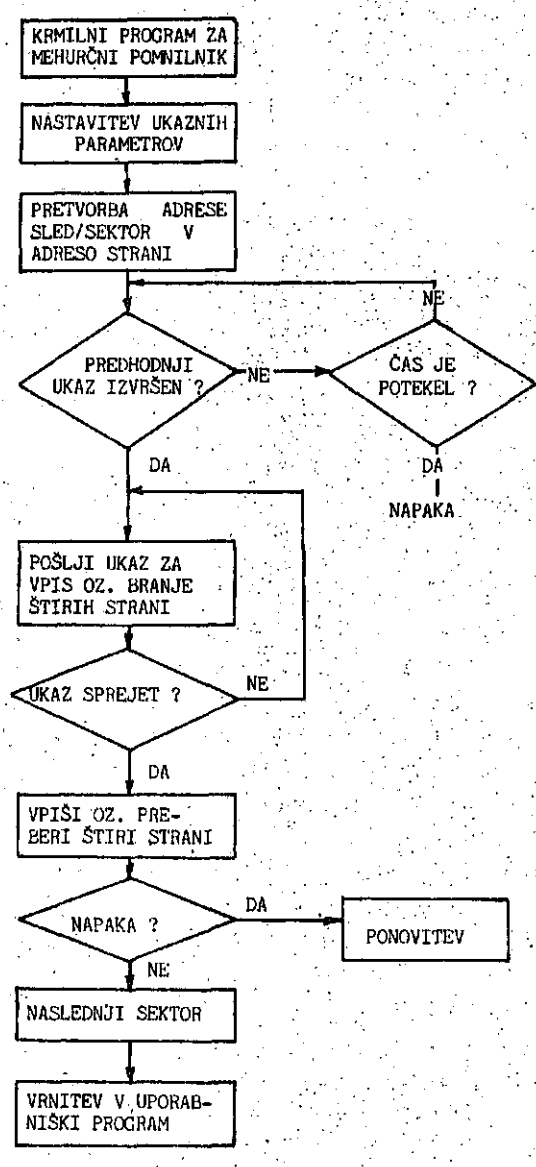
primeru diskovno) zagotovimo, da je z nivoja uporabniških programov mehurčni pomnilniški sistem povsem enak disku. Torej se odnos med operacijskim sistemom in uporabniškimi programi nič ne spremeni.
Kakšne spremembe je potrebno izvesti v krmilnem programu si najlaže ogledamo na konkretnem primeru. Vzemimo 256 Kbitni pomnilniški element, ki vsebuje 282 minor zank s po 1024 biti na zanko. Na podlagi "map" zanke je izbranih 262 dobrih minor zank. Vsi istoležeči biti v dobrih minor zankah tvorijo blok podatkov imenovan tudi stran, kjer je 256 bitov izrabljenih za podatek, 6 bitov pa služi za detekcijo in korekcijo napak. Dolžina strani je torej 32 bytov, kar je dolžina sektorja standardega diska. Nadalje je definirana tudi logična sled diska, ki jo sestavlja 32 sektorjev. Dolžina sektorja je 4 Kbyte. Tako bi imel obravnavani mehurčni pomnilniški element organiziran enako kot disk, skupno 8 sledi s po 32 sektorji. Torej je naloga krmilnega programa (handlerja) da pretvarja adreso sledi in sektorja diska v ustrezno adreso strani mehurčnega pomnilnika. Tako velja sled nič sektor nič je stran nič, itd... Diagram poteka krmilnega programa (handlerja), za mehurčni pomnilniški sistem je prikazan na sliki 11.



Slika 10.

5. ZAKLJUČEK

Namen pričujoče serije člankov o mehurčnih pomnilnikih je bil seznaniti bralca z osnovnimi fizikalno - tehnološkimi principi delovanja mehurčnih pomnilnikov, podati njihovo organizacijsko zgradbo, tako na nivoju pomnilniškega elementa, kakor pomnilniškega sistema ter nenazadnje nakazati možnosti uporabe tega novega pomnilniškega medija.



Slika 11.

Od vseh tehnologij, ki so bile nakazane v desetih letih burnega razvoja mehurčnih pomnilnikov je danes realizirana le tehnologija permalojnih propagacijskih vzorcev. Čeprav imajo mehurčni pomnilniki, ki so realizirani v tej tehnologiji, številne prednosti pred elektromehanskimi pomnilniškimi mediji, se zaradi relativno visoke cene še ne morejo širše uveljaviti. Trenutno so njihove aplikacije omejene na področja, kjer cena ni bistvena postavka, temveč se izrabljajo njihove lastnosti kot so integriranost, ohranitev informacije, majhna teža in volumen in zanesljivost, itd... Takšna področja so predvsem letalstvo in vesoljka tehnika.

Kaj lahko rečemo o razvoju in uporabi mehurčnih pomnilnikov v naslednjih letih? Na področju tehnologije pomnilniškega elementa še ni izrečena zadnja beseda. Kristalični garnetni substrati, katerih priprava je zelo draga, bodo nadomeščeni z amorfnimi materiali. Stični diski in mrežica mehurčkov sta potencialna naslednika permalojnih vzorcev, to pa pomeni večje gostote integracije in s tem občutno večanje kapacitete mehurčnega pomnilnika. Izboljšave se pričakujejo tudi na področju organizacije pomnilnika, kjer naj bi dinamično urejeni podatkovni bloki nadomestili major/minor zančno organizacijo, kar pomeni skrajševanje časa dostopa. Tudi na nivoju pomnilniškega sistema se pričakujejo izboljšave v smislu vse bolj integrirane podporne materialne opreme, tako že danes nekaj proizvajalcev nudi VLSI krmilnike. V kolikor bo proizvajalcem uspelo zniževati ceno mehurčnih pomnilniških sistemov kakor so napovedovali, bomo v naslednjih letih vse pogosteje srečevali v računalniških sistemih tudi mehurčne pomnilnike.

6. LITERATURA

[1] P. KOLBEZEN, R. TROBEC, J. ŠILC, B. MIHOVILOVIĆ: Mehurčni pomnilniki, IJS Ljubljana, junij 1981

[2] J. ŠILC, B. MIHOVILOVIĆ, P. KOLBEZEN: Mehurčni pomnilniki - II del, Informatica 4/1980 pp. 47- 55

[3] B. MIHOVILOVIĆ, J. ŠILC, P. KOLBEZEN: Mehurčni pomnilniki - III del, Informatica 1/1981 pp. 47 - 55

[4] R. MACDONALD: Bubble memory circuits promote 3 - dimensional stacking, Computer Desigen, pp. 135 - 141, June 1981

[5] A. C. FOREMAN: Bubble memory mass storage for microcomputers, Digital Design pp. 26 - 37, June 1981

[6] R. LUTHRA, G. REYLING Jr.: Control chip and driver program unlock magnetic - bubble potential, Electronics, pp. 138 - 143, February 1981

[7] C. E. LETOCQ: Software driver lets CP/M address bubbles as a disk, Electronics, pp. 143 - 148, February 1981

PETO REPUBLIŠKO TEKMOVANJE SREDNJEŠOLCEV S PODROČJA RAČUNALNIŠTVA

I. TVRDY
M. MARTINEC
R. REINHARDT

UDK: 371.27-681.3

INSTITUT JOŽEF STEFAN, LJUBLJANA

Povzetek. Prispevek podaja poročilo o petem republiškem tekmovanju srednješolcev s področja računalništva, ki ga je organiziralo Slovensko društvo Informatika 30. maja 1981. V prispevku so vse naloge z rešitvami in pregled rezultatov.

FIFTH COMPUTER SCIENCE CONTEST FOR HIGH-SCHOOL STUDENTS. The article gives a report on the Fifth Computer Science Contest. It includes the complete set of problems with their solutions and a short overview of contest results.

1. Uvod

Ena od rednih dejavnosti Slovenskega društva Informatika je tudi popularizacija računalništva in informatike med srednješolsko mladino. Komisija za popularizacijo računalništva je zato skupaj z Institutom Jožef Stefan, Fakulteto za elektrotehniko in Fakulteto za naravoslovje in tehnologijo organizirala že peto republiško tekmovanje srednješolcev s področja računalništva.

Tekmovanje je bilo 30. maja 1981 na Fakulteti za elektrotehniko v Ljubljani. Udeležilo se ga je 67 tekmovalcev po enem letu pouka in 34 tekmovalcev po dveh letih pouka računalništva.

Tisti, ki so imeli za sabo že tri leta pouka računalništva, so bili prikljubeni tekmovalcem, ki so poslušali računalništvo samo dve leti, saj jih je bilo premalo za samostojno skupino.

Tekmovalna komisija je ob pripravljanju in izbiranju nalog upoštevala neizenačenost učnih programov, različno znanje dijakov in različne možnosti za delo na računalniku. Zato je poskušala izbrati naloge, ki ne zahtevajo posebnega znanja, niti ne temeljijo preveč na praktičnih izkušnjah učencev. Naloge poskušajo pokriti najvažnejše učne smotre, zato zahtevajo od tekmovalca predvsem razvit smisel za algoritmično razmišljanje.

Programi v nalogah in v rešitvah nalog so v tem članku zapisani v pascalu, tekmovalci pa so imeli možnost opravljati tekmovanje tudi v fortranu ali basicu.

Čas tekmovanja je bil za obe skupini omejen na 2 uri 30 minut, tekmovalci pa so smeli uporabljati poljubno literaturo.

2. Naloge za učence po enem letu pouka računalništva

1. Barvna slika, ki jo pošilja Voyager 1 na zemljo je sestavljena iz treh črno-belih slik. Vsaka slika je sestavljena iz 256 x 256 točk, v vsaki točki pa ločimo 256 različnih svetlostnih jakosti. Kolikšna je hitrost prenosa slikovnih podatkov (koliko bitov na sekundo), če je potrebnih 67 minut (5249 sekund) za prenos cele barvne slike (to je vseh treh slik)?

2. Imamo spodnji program

```
program Hamming (input, output);
 var x, y, d: integer;
 begin { Hamming }
  read(x, y);
  d := 0;
  while (x <> 0) or (y <> 0) do
   begin
    if (x mod 2) <> (y mod 2) then
     d := d + 1;
    x := x div 2;
    y := y div 2
   end;
  writeln(d);
 end. { Hamming }
```

a. Na nekaj primerih izračunaj, kaj izpiše program Hamming.

b. Program Hamming iz podatkov x in y izračuna rezultat d. Napiši program AntiHamming, ki prečita x in d in izračuna tak y, da bo program Hamming iz x in y izračunal prav d. (y ni enolično določen; AntiHamming naj izračuna poljuben tak y.)

3. Na računalnik imamo priključen grafični terminal, kjer lahko na vsako točko zaslona vplivamo s klicem podprograma

pobarvaj (x, y, barva);

kjer sta x in y koordinati točke med 0 in 500, barva pa ima vrednosti 1, če želimo, da bo točka bela in vrednost 0, če želimo točko črno.

Poleg tega imamo na razpolago še škatlico, ki jo z roko premikamo po mizi, s programom pa lahko odčitamo, za koliko smo škatlico (ki ji rečemo "miš") premaknili v x in y smeri od zadnjega odčitka. Za to uporabljamo podprogram

miš (dx, dy, tipka);

ki v dx in dy vrne, za koliko sta se spremenili x in y koordinata (dx in dy sta lahko tudi negativna). Spremenljivka tipka pa ima vrednost 0, če tipka, ki je na miši, ni pritisnjena in 1, če je.

Napiši program, ki bo omogočal risanje belih črt po zaslonu s pomočjo premikanja miši po mizi. Miš naj pušča sled samo takrat, ko je tipka pritisnjena. Program naj se razumno obnaša tudi, ko odpeljemo miš iz območja zaslona. Pri pisanju tega programa lahko upoštevaš, da je izvajanje programa hitro v primerjavi s premikanjem miši, tako da sta vrednosti dx in dy vedno -1, 0 ali 1.

4. Opiši postopek, s katerim preverimo, ali sta dve besedi anagrama (anagrama sta besedi, ki ju sestavljajo iste črke v pomešanem vrstnem redu - npr. DESTILARNA / NADREALIST).

5. Napiši program, ki bo izpisal vse kvadrate naravnih števil do N (podatek), ki so palindromi. Palindrom je beseda (število), ki se naprej in nazaj bere enako.

III. Naloge za učence po dveh letih pouka računalništva

1. V ravnini imamo dva pravokotnika, katerih stranice so vzporedne koordinatnima osema. Vsak izmed pravokotnikov je podan s koordinatama spodnjega levega in zgornjega desnega oglišča.

a. Napiši podprogram, ki poišče najmanjši možni pravokotnik, ki vsebuje oba pravokotnika.

b. Napiši podprogram, ki ugotovi, ali imata notranjosti pravokotnikov kakšno skupno točko.

2. Za kakšne podatke izpiše naslednji program rezultat 0? Najprej izračunaj nekaj primerov!

Opomba: komentarji v zavitih oklepajih so invariante in jih v originalni nalogi ni bilo.

```
program modul (input, output);
 var ai k : integer;
 begin { modul }
  readln (a);
  k := 10;
  while 2*k <= a do  k := 2*k;
  { (k <= a < 2*k) & (∃n)(k = 10*2^n) }
  while a >= 10 do
   begin
    { a < 2*k) & (∃n)(k = 10*2^n) }
    if a >= k then
     { vetja: k <= a < 2*k }
     a := a - k;
     { vetja: a < k }
    k := k div 2;
    { vetja: a < 2*k }
   end;
  writeln (a)
 end. { modul }
```

3. Vesoljski dolničkek (space shuttle) je zelo zapleteno vesoljsko vozilo, ki ga je možno voditi samo s pomočjo računalnika. Zaradi večje zanesljivosti je na krovu 5 enakih računalnikov, ki izvajajo enake programe. Računalniki tudi nadzorujejo drug drugega in če jih večina ugotovi, da kdo od njih daje napačne ukaze, lahko glasujejo za njegovo izključitev.

Napiši tisti del programa (ki bo tekel hkrati na vseh računalnikih), ki primerja akcije ostalih računalnikov in če je potrebno, glasuje za izključitev enega ali več izmed njih. Izključitev računalnika opravi posebna elektronika, če je večina glasov za njegovo izključitev. Na voljo so naslednji podprogrami:

kdosem(i)	i postane številka računalnika (od 1 do 5), ki izvaja ta program.
kajpravi(i,akcija)	i je številka računalnika, akcija pa je ukrep, ki ga i-ti računalnik predlaga v danem trenutku. Če je akcija enaka 0, je i-ti računalnik že izključen.
glasujproti(i)	i je številka računalnika.

Kako bi napisal ta del programa, če ne bi imel podprograma kdosem?

Opomba: ta naloga je poenostavitev dejanskega stanja, ki se od opisane situacije nekoliko razlikuje.

4. Palindrom je zaporedje znakov, ki se ga enako bere z leve in z desne strani (npr. PERICAREZERACIKEP).

Napiši program, ki iz danega niza znakov, ki ga zaključuje presledek, poišče najdaljši palindrom, ki se začne s prvim znakom.

5. V pravokotnem koordinatnem sistemu imamo podanih n točk, med njimi imamo izbrano začetno in končno točko. Opiši postopek, ki določi, ali lahko povežemo začetno in končno točko z daljicami, ki niso daljše od a. Vse daljice imajo za krajišča podane točke. (Ni potrebno poiskati zaporedja točk, ki povezujejo začetno in končno točko, ampak samo odločiti, ali je to možno, ali ne.)

IV. Rezultati prvouvrščenih tekmovalcev v vsaki skupini

PO ENEM LETU POUKA RAČUNALNIŠTVA

nagrada	št. točk	tekmovalec; šola
I.	98	Tomi VEBER Gimnazija Ljubljana Vič
I.	97	Dean MOŽETIČ Gimnazija Koper
I.	93	Igor KOKAVICA I. gimnazija Ljubljana Bežigrad
II.	90	Alen VARŠEK Gimnazija Ljubljana Vič
II.	88	Aram KARALIČ Gimnazija Ljubljana Šentvid
II.	84	Zoran KOZINA računalniški krožek ISKRA
III.	81	Tomaž DEBAŠEK računalniški krožek ISKRA
III.	61	Leon CIZELJ Tehniška šola Celje
III.	78	Mitja BENSA Gimnazija Nova Gorica

PO DVEH LETIH POUKA RAČUNALNIŠTVA

nagrada	št. točk	tekmovalec; šola
I.	100	Ivan PEPELNJAK I. gimnazija Ljubljana Bežigrad
I.	95	Matjaž KAUFMAN I. gimnazija Ljubljana Bežigrad
I.	85	Matjaž KALOŽA Gimnazija M. Zidanška Maribor
II.	81	Branko DERNAČ Gimnazija Brežice
II.	73	Robert BANULA I. gimnazija Ljubljana Bežigrad
II.	70	Maja PERLE Gimnazija M. Zidanška Maribor
III.	59	Alfred ANŽLOVAR I. gimnazija Ljubljana Bežigrad
III.	55	Aleksander BAJISTIČ Gimnazija Nova Gorica
III.	54	Nevenka PUŠTAVRH Gimnazija B. Ziherl Škofja Loka

V. rešitve nalog za učence po enem letu pouka računalništva

1. Za reprezentacijo svetlostnega nivoja v eni točki slike potrebujemo $\log_2 256 = 8$ bitov (binarni zapis števila!). V vsaki sliki je 256 x 256 točk. Zato je hitrost prenosa

$$\frac{4 \times 256 \times 256 \times 8 \text{ bit}}{5243 \text{ s}} = 500 \text{ bit/s}$$

2. a) program Hamming presteje število mest, na katerih se razlikujeta binarna zapisa števil x in y. Odtod tudi ime "Hammingova razdalja".

Zanka v programu se izvaja tako dolgo, da postaneta x in y oba enaka 0. Ob vsakem prehodu skozi zanko se x in y celoštevilčno razpolovita. Označimo vrednost x v k-tem prehodu skozi zanko z x(k) in vrednost y z y(k). Tedaj velja, kot se vidi iz programa:

```
   x(0) = x                    y(0) = y
x(k+1) = x(k) div 2      y(k+1) = y(k) div 2
```

d enostavno šteje, kolikokrat je po parnosti različnih od y(k).

b) za program AntiHamming imamo precej možnosti. V tej rešitvi binarnemu zapisu števila x zamenjamo na zadnjih d mestih vse ničle z enicami in obratno.

```
program AntiHamming (input, output);
 var
  x, y, d: integer;
  i,j: integer;
 begin   { AntiHamming }
  readln (x, d);
  j := 1; y := 0;
  { z naslednjo zanko prepišemo zadnjih
    d cifer; zamenjane ničle in enice   }
  for i := 1 to d do
   begin
    if not odd (x) then y := y + j;
    j := 2*j; x := x div 2
   end;
  { z naslednjo zanko prepišemo ostanek }
  while x <> 0 do
   begin
    if odd (x) then y := y + j;
    j := 2*j; x := x div 2
   end;
  writeln (y)
 end.   { AntiHamming }
```

Če bi imeli na voljo vse potrebne operacije (potenciranje ali generiranje maske, logične operacije s celimi števili), bi lahko ves postopek zapisali takole:

```
...
m := (2^d - 1);
y := (x and (not m)) or ((not x) and m);
...
```

```
3. program črta (output);
   var
    x, y, dx, dy: integer;
    tipka: integer;
   procedure pobarvaj (x,y,barva: integer);
    external;
   procedure miš (var dx,dy,tipka: integer);
    external;

   begin  { črta }
    x := 250; y := 250;
    while true do { forever }
     begin
      miš (dx, dy, tipka);
      x := x + dx; y := y + dy;
      if tipka = 1 then
       if (x >= 0) and (x <= 500) and
          (y >= 0) and (y <= 500) then
          pobarvaj (x, y, 1)
     end
   end.  { črta }
```

4. Rešitev 1:
Za vsako besedo naredimo tabelo pogostosti črk. Če sta tabeli enaki, sta besedi anagrama, sicer nista.

Rešitev 2:
črke v vsaki besedi uredimo po abecedi. Če sta sedaj besedi enaki, sta prvotni besedi anagrama, sicer nista.

```
5. program Palindrom (output);
   var
    N, m: integer;
    mm, a, b: integer;

   begin { Palindrom }
    readln (N);
    for m := 1 to N do
     begin
      mm := sqr (m);
      { obrnemo cifre v stevilu mm }
      a := mm; b := 0;
      while a > 0 do
       begin
        b := 10 * b + a mod 10;
        a := a div 10
       end;
      if b = mm then writeln (mm)
     end
   end.  { Palindrom }
```

VI. rešitve nalog za učence po dveh letih pouka računalništva

```
1. ...
   type
    pravokotnik = record
     xmin, ymin, xmax, ymax: integer
    end;
    ...
   procedure objemi ( a, b: pravokotniki
                      var p: pravokotnik);
    begin
     if a.xmin < b.xmin then p.xmin := a.xmin
                        else p.xmin := b.xmin;
     if a.ymin < b.ymin then p.ymin := a.ymin
                        else p.ymin := b.ymin;
     if a.xmax > b.xmax then p.xmax := a.xmax
                        else p.xmax := b.xmax;
     if a.ymax > b.ymax then p.ymax := a.ymax
                        else p.ymax := b.ymax
    end;
    ...
   function presek (a,b:pravokotnik): boolean;
    begin
     presek := not ((a.xmax < b.xmin)
                 or (a.xmin > b.xmax)
                 or (a.ymax < b.ymin)
                 or (a.ymin > b.ymax))
    end;
```

2. program izpiše ostanek pri deljenju podatka s številom 10.

V nalogi sami so kot komentarji pripisane invariante, torej pogoji, ki vedno veljajo na tistem koraku programa. Od nenegativne začetne vrednosti števila a odštevamo mnogokratnike števila 10. Program se ustavi v končno mnogo korakih: k se med izvajanjem zanke razpolavlja in ko postane k <= 5 in ker je a < 2*k, pade vrednost a pod 10, kar je pogoj za ustavitev zanke.

Rezultat 0 dobimo natanko pri tistih nenegativnih podatkih, ki so deljivi z 10.

```
3. ...
   var jaz, prad, kaj: integer;
   ...
    kdosem (jaz); kajpravi (jaz, a);
    for prad := 1 to 5 do
     begin
      kajpravi (prad, kaj);
      if kaj <> a then glasujproti (prad)
     end;
```

Rešitev brez uporabe podprograma kdosem:

```
   ...
   var
    akc, frekv, a: array [1..5] of integer;
    prad, indeks, n, j, min: integer;
   ...
    n := 0;
    { napravimo seznam različnih akcij, ki
      jih predlagajo računalniki }
    for prad := 1 to 5 do
     begin  kajpravi (prad, a[prad]);
      if a[prad] <> 0 then
       begin  indeks := 0;
        for j := 1 to n do
         { ali je akcija a[prad] že predlagana }
          if akc[j] = a[prad] then indeks := j;
          { shrani akcijo a[prad] k ostalim }
         if indeks > 0 then
          frekv[indeks] := frekv[indeks] + 1
         else
          begin  n := n + 1;
           akc[n] := a[prad]; frekv[n] := 1
          end
       end { if }
     end; { for }
    if n >= 2 then { predlagani sta vsaj dve
                     različni akciji }
     begin
      min := 1; { poiščemo akcijo, za katero
        se je odločilo najmanj računalnikov }
      for j := 2 to n do
       if frekv[j] < frekv[min] then min := j;
       { izločimo računalnike, ki predlagajo
         tako akcijo }
      for prad := 1 to 5 do
       if a[prad] = akc[min] then
        glasujproti (prad)
     end; { if }
   ...
```

```
4. program Palindrom (input, output);
   const maxniz = 80;
   var
    niz: array [1..maxniz] of char;
    konec, pal: boolean;
    n, i, j: integer;
   begin   { Palindrom }
    n := 0;
    while (input^ <> ' ') and (n < maxniz) do
     begin n := n + 1; read (niz[n]) end;
    konec := false; pal := false;
    while not konec do
     if n < 1 then konec := true
     else
      begin pal := true;
       i := 1; j := n;
       while (i <= j) and pal do
        if niz[i] <> niz[j] then pal := false
        else
         begin i := i + 1; j := j - 1 end;
       if pal then konec := true
       else n := n - 1
      end;
    if pal then
     begin
      for i := 1 to n do write (niz[i]);
      writeln
     end
   end.  { Palindrom }
```

5. V množico dosegljivih točk damo začetno točko. Dokler v množici ni končne točke in dokler obstaja še kakšna točka, ki je v množici še ni in je dosegljiva iz vsaj ene točke iz množice, potem takšno točko dodamo množici. Če je v množici končna točka, potem je možno povezati začetno in končno točko, sicer to ni možno.

UPORABNI PROGRAMI

Poziv časopisa Informatica bralcem, da objavljajo svoje uporabne programe, se nadaljuje. Katera so programirna uporabna področja, ki nas še posebej zanimajo?

Naštejmo le nekaj področij:

poslovnometodološki programi
ekonometrični izračuni
napovedovanje dogodkov
vrednotenje proizvodnje
planiranje proizvodnje
mrežno planiranje
računovodski programi
statistični izračuni
poslovodne igre
programi za preizkušanje sposobnosti
obdelava podatkovnih zbirk
tehnološki programi
bančna in finančna aritmetika
programi za učenje
itd.

Konkretni uporabni programi so npr. programi za mali poslovni sistem, za glavno poslovno knjigo, za evidenco prihodkov in izdatkov, za obračunavanje osebnih dohodkov, za izračun različnih statističnih kazalcev, tudi krajši zabavni programi (igre), dalje programi za preizkušanje računalniških virov in naprav, za kriptografiranje (šifriranje podatkov), za slikovito prikazovanje rezultatov na zaslonu, za urejanje in upravljanje podatkovnih zbirk itn.

Rubrika "Uporabni programi" naj bi vzpodbujala bralce, da pišejo kratke prispevke v dokaj standardni obliki, ki obsega

kratek opis področja, na katerem se program uporablja, s pripadajočo metodologijo (pojasnili)

programsko listo s komentarji v visokem programirnem ali zbirnem jeziku (PL/I, FORTRAN, COBOL, PASCAL, BASIC, ALGOL, FORTH, LISP, ADA, MODULA in zbirni jeziki mikroprocesorjev)

izvajanje (izvršitev) programa na dovolj slikovitem primeru, po možnosti v realnem okolju

Uredništvo bo vsakemu prispevku dodalo določen označevalnik. Tekst in formati programskih list se morajo prilegati avtorskim formularjem, ki jih dobite v uredništvu časopisa Informatica, Parmova 41, 61000 Ljubljana.

```
**************************************
*  EKONOMSKA KOLIČINA PROIZVODNJE    *
**************************************

******************************************
*  Informatica UP 4                      *
*  Ekonomska količina proizvodnje        *
*  november 1981                         *
*  Izvor: Practical Basic Programs       *
*         (ed. Lon Poole, McGraw-Hill)   *
*  modificiral A.P.Železnikar            *
*  sistem CP/M, CBASIC2                  *
*  računalnik Delta 323/M1               *
******************************************
```

1. Področja uporabe programa

Večkrat želimo vedeti, kakšna naj bi bila optimalna količina prizvodov neke vrste glede na minimalne stroške. Ta program izračuna to vrednost, vključuje pa tudi prodajni izračun (ko se proizvodi že prodajajo, toda še več se jih proizvaja). Program ima tako zelo široko področje uporabe, saj ga je mogoče uporabiti pri kalkulacijah proizvodnje različnih izdelkov.

Program uporabimo tako, da vstavimo proizvodno razmerje (število izdelkov na dan), prodajno ali uporabno razmerje (povprečno število izdelkov, ki jih dnevno vzamemo iz zaloge), celotno število prodanih izdelkov na leto, stroške skladiščenja (v dinarjih na eno to oziroma na izdelek) in proizvodne stroške. Program bo izračunal optimalno število izdelkov na leto in optimalno proizvodno količino v vsaki proizvodni seriji. Optimalna količina izdelkov je tista, pri kateri so proizvodni in vzdrževalni stroški najmanjši.

2. Kratek opis programa

Program za izračun ekonomske proizvodne količine nekega izdelka je prikazan v listi 1. Vhodni podatki se vstavijo v vrsticah 40 (proizvodno razmerje), 110 (prodajno ali uporabno razmerje), 180 (letna prodaja ali uporaba), 250 (stroški skladiščenja na enoto) in 320 (proizvodni stroški). Optimalno število izdelkov se izračuna v vrstici 390, v vrstici 400 pa imamo izračun ekonomske proizvodne količine. Program je dokaj enostaven in pojasnjuje sam sebe.

3. Izvajanje programa

Izvajanje programa si oglejmo na nekaj primerih, ki jih imamo na listi 2. Kot vidimo je program narejen tako, da omogoča ponavljanje postopka izračuna z novimi vhodnimi podatki, dokler to želimo. Lista 2 prikazuje šest primerov, ki lahko pomenijo različne situacije. Vzemimo tele primere:

Primer 1: Tovarna lakov proizvaja lake v posodah (enotah), in sicer v različnih barvah (različne serije). Za to svoje izdelovanje ima na razpolago en sam mešalni in polnilni stroj. Ta stroj proizvaja 400 posod (enot) laka dnevno in tovarna lakov odpremlja 130 posod vsake barve (seveda tistih, ki jih izdeluje) dnevno, vendar 36000 posod letno. Stroški skladiščenja posode znašajo 50 din na leto. Za vsako serijo

```
B>TYPE PROIZ1.BAS
10 PRINT "EKONOMICNA KOLICINA PROIZVODNJE"
20 PRINT
30 PRINT "VSTAVI PROIZVODNO RAZMERJE (ENOTE/DAN)          ";
40 INPUT R
50 IF R>0 THEN 100
60 PRINT
70 PRINT "PROIZVODNO RAZMERJE MORA BITI VECJE OD NIC"
80 PRINT
90 GOTO 30
100 PRINT "VSTAVI PRODAJNO ALI UPORABNO RAZMERJE (ENOTE/DAN)";
110 INPUT U
120 IF U>0 THEN 170
130 PRINT
140 PRINT "PRODAJNO (UPORABNO) RAZMERJE MORA BITI VECJE OD NIC"
150 PRINT
160 GOTO 100
170 PRINT "VSTAVI LETNO PRODAJO ALI UPORABO               ";
180 INPUT H
190 IF H>U THEN 240
200 PRINT
210 PRINT "LETNO RAZMERJE MORA BITI VECJE OD DNEVNEGA"
220 PRINT
230 GOTO 170
240 PRINT "VSTAVI STROSKE SKLADISCENJA ENOTE (DIN NA ENOTO) ";
250 INPUT J
260 IF J>0 THEN 310
270 PRINT
280 PRINT "STROSKI SKLADISCENJA MORAJO BITI VECJI OD NIC"
290 PRINT
300 GOTO 240
310 PRINT "VSTAVI STROSKE MENJAVE SERIJE (DIN)            ";
320 INPUT S
330 PRINT
340 IF S>0 THEN 380
350 PRINT "PROIZVODNI STROSKI MORAJO BITI VECJI OD NIC"
360 PRINT
370 GOTO 310
380 REM IZRACUNAJ REZULTATE
390 N=INT(SQR(((J*H)/(2*S))*(1-(U/R)))+.5)
400 PRINT "OPTIMALNO STEVILO SERIJ NA LETO ZNASA ";N
410 PRINT "EKONOMSKA KOLICINA PROIZVODOV SERIJE JE";INT(H/N)
420 REM PONOVI ALI KONCAJ PROGRAM
430 PRINT "ALI ZELIS PONOVITI PROGRAM Z NOVIMI PODATKI (DA/NE) ?"
440 INPUT Z$
450 IF Z$="DA" THEN 20
460 IF Z$<>"NE" THEN 440
470 END
```

Lista 1. Ta kratek program izračunava primere, ki jih srečamo v proizvodnji in so prikazani v listi 2. Vhodni podatki se vstavijo s programskimi vrsticami 40, 110, 180, 250 in 320. Namesto sestavljenih INPUT stavkov so pri tem uporabljeni pari (PRINT, INPUT) stavkov. Rezultata se izračunavata v vrsticah 390 in 410.

(menjavo barve) mora biti stroj očiščen in stroški čiščenja znašajo 20000 din. Ob tem se postavlja vprašanje, koliko različnih serij (barvnih vrst) naj tovarna lakov letno proizvede.

Odgovor: Tovarna lakov naj proizvede letno 6 serij po 6000 posod. To pomeni, da lahko proizvede 6 različnih barv.

Priner 2: Ta primer je ponovitev primera 1 ob povečanih stroških čiščenja mešalnega stroja, ki so zdaj dvainpolkrat večji.

Odgovor: Tovarna lakov naj proizvede letno le 3 serije po 12000 posod.

Primer 3: Obrat, ki izdeluje računalniške terminale, izdela dnevno 20 terminalov in 10 terminalov (različnih vrst) jih dnevno odpremi naročnikom. Letno zagotovljena prodaja (po pogodbah) znaša 3000 terminalov. Stroški skladiščenja enega terminala so 5000. Pri menjavi proivoda (terminala druge vrste) je potrebno vselej preurediti proizvodno linijo in stroški te preureditve znašajo 1000000 din. Vprašanje je, koliko različnih vrst terminalov naj ta obrat proizvede na leto.

Odgovor: Obrat terminalov naj proizvede letno 2 seriji terminalov po 1500 kosov. To pomeni, da naj izdela ta obrat le dva tipa terminalov.

Primer 4: Ta primer je ponovitev primera 3 z drugimi vhodnimi podatki.

Primer 5: Tovarna računalnikov proizvede dnevno tri računalniške sisteme pri dnevni prodaji oziroma uporabi 2 sistemov (ta tovarna je namreč še v izgradnji in v svojih proizvodnih postopkih uporablja lastne računalniške sisteme). Letna prodaja in uporaba znaša 300 računalniških sistemov pri minimalnih stroških skladiščenja 1000 din na sistem. Tudi menjava proizvoda je cenena, saj je avtomatizirana z lastnimi sistemi in znaša le 10000 din za serijo. Kakšno je optimalno število serij (tipov sistemov) na leto?

Odgovor: Tovarna računalnikov naj proizvede 2 seriji po 150 računalniških sistemov na leto.

Primer 6: Ta primer je ponovitev primera 5 s spremenjenimi vhodnimi podatki. Ob zmanjšani dnevni prodaji in uporabi ter pri desetkrat višjih stroških skladiščenja je odogovr tale:

Odgovor: Tovarna računalnikov naj proizvede 10 serij po 30 različnih računalniških sistemov.

Primer 1

```
A>B:CRUN2 PROIZI

CRUN VER 2.07P
EKONOMICNA KOLICINA PROIZVODNJE

VSTAVI PROIZVODNO RAZMERJE (ENOTE/DAN)              ? 400
VSTAVI PRODAJNO ALI UPORABNO RAZMERJE (ENOTE/DAN)? 130
VSTAVI LETNO PRODAJO ALI UPORABO                    ? 36000
VSTAVI STROSKE SKLADISCENJA ENOTE (DIN NA ENOTO)    ? 50
VSTAVI STROSKE MENJAVE SERIJE (DIN)                 ? 20000

OPTIMALNO STEVILO SERIJ NA LETO ZNASA  6
EKONOMSKA KOLICINA PROIZVODOV SERIJE JE 6000
```

Primer 2

```
ALI ZELIS PONOVITI PROGRAM Z NOVIMI PODATKI (DA/NE) ?
? DA

VSTAVI PROIZVODNO RAZMERJE (ENOTE/DAN)              ? 400
VSTAVI PRODAJNO ALI UPORABNO RAZMERJE (ENOTE/DAN)? 130
VSTAVI LETNO PRODAJO ALI UPORABO                    ? 36000
VSTAVI STROSKE SKLADISCENJA ENOTE (DIN NA ENOTO)    ? 50
VSTAVI STROSKE MENJAVE SERIJE (DIN)                 ? 50000

OPTIMALNO STEVILO SERIJ NA LETO ZNASA  3
EKONOMSKA KOLICINA PROIZVODOV SERIJE JE 12000
```

Primer 3

```
ALI ZELIS PONOVITI PROGRAM Z NOVIMI PODATKI (DA/NE) ?
? DA

VSTAVI PROIZVODNO RAZMERJE (ENOTE/DAN)              ? 20
VSTAVI PRODAJNO ALI UPORABNO RAZMERJE (ENOTE/DAN)? 10
VSTAVI LETNO PRODAJO ALI UPORABO                    ? 3000
VSTAVI STROSKE SKLADISCENJA ENOTE (DIN NA ENOTO)    ? 5000
VSTAVI STROSKE MENJAVE SERIJE (DIN)                 ? 1000000

OPTIMALNO STEVILO SERIJ NA LETO ZNASA  2
EKONOMSKA KOLICINA PROIZVODOV SERIJE JE 1500
```

Primer 4

```
ALI ZELIS PONOVITI PROGRAM Z NOVIMI PODATKI (DA/NE) ?
? DA

VSTAVI PROIZVODNO RAZMERJE (ENOTE/DAN)              ? 10
VSTAVI PRODAJNO ALI UPORABNO RAZMERJE (ENOTE/DAN)? 8
VSTAVI LETNO PRODAJO ALI UPORABO                    ? 1500
VSTAVI STROSKE SKLADISCENJA ENOTE (DIN NA ENOTO)    ? 2000
VSTAVI STROSKE MENJAVE SERIJE (DIN)                 ? 100000

OPTIMALNO STEVILO SERIJ NA LETO ZNASA  2
EKONOMSKA KOLICINA PROIZVODOV SERIJE JE 750
```

```
ALI ZELIS PONOVITI PROGRAM Z NOVIMI PODATKI (DA/NE) ?
? DA
```

Primer 5

```
VSTAVI PROIZVODNO RAZMERJE (ENOTE/DAN)              ? 3
VSTAVI PRODAJNO ALI UPORABNO RAZMERJE (ENOTE/DAN)? 2
VSTAVI LETNO PRODAJO ALI UPORABO                    ? 300
VSTAVI STROSKE SKLADISCENJA ENOTE (DIN NA ENOTO)    ? 1000
VSTAVI STROSKE MENJAVE SERIJE (DIN)                 ? 10000

OPTIMALNO STEVILO SERIJ NA LETO ZNASA  2
EKONOMSKA KOLICINA PROIZVODOV SERIJE JE 150
```

Primer 6

```
ALI ZELIS PONOVITI PROGRAM Z NOVIMI PODATKI (DA/NE) ?
? DA

VSTAVI PROIZVODNO RAZMERJE (ENOTE/DAN)              ? 3
VSTAVI PRODAJNO ALI UPORABNO RAZMERJE (ENOTE/DAN)? 1
VSTAVI LETNO PRODAJO ALI UPORABO                    ? 300
VSTAVI STROSKE SKLADISCENJA ENOTE (DIN NA ENOTO)    ? 10000
VSTAVI STROSKE MENJAVE SERIJE (DIN)                 ? 10000

OPTIMALNO STEVILO SERIJ NA LETO ZNASA  10
EKONOMSKA KOLICINA PROIZVODOV SERIJE JE 30
```

```
ALI ZELIS PONOVITI PROGRAM Z NOVIMI PODATKI (DA/NE) ?
? NE

A>
```

Lista 2. Ta lista kaže šest primerov, ki so podrobneje pojasnjeni (interpretirani) v tekstu. Kot vidimo, vplivajo na število serij najbolj odločilno stroški, ki nastanejo zaradi menjave serije, npr. zaradi spremljanja proizvodnje linije, nameščanja drugačnega orodja, prilagajanja, tudi čiščenja strojev (v primeru proizvodnje različnih barv) itn. Ob tem se upoštevajo tudi stroški skladiščenja, ki nastanejo zaradi proizvodnje, ki presega trenutno porabo (plasma na tržišče). Ti primeri kažejo, kako koristno je imeti tovrstne kratke programe za ocenjevanje podobnih situacij in prihaja čas, ko bodo tudi naši vodilni delavci morali ocenjevati operativne situacije na ta način. K temu bodo prav gotovo lahko pripomogli osebni računalniki na delovnem mestu in tudi ta program je bil napisan za mikroračunalnik Delta 323/M1, ki je za takšne aplikacije kot naročen.

INFORMACIJSKI SISTEM
NAROČNIKOV ČASOPISA INFORMATICA

Informatica UP 5
INFO
december 1981
B.Blatnik, J.Šantl
sistem DELTA-N V1.2,
FMS-11, BASIC2

1.Področje uporabe programa

Program INFO (UP 5) in indeksna datoteka CASOPI.IDX sestavljata informacijski sistem naročnikov časopisa Informatica.

Program INFO omogoča naslednje operacije nad datoteko CASOPI.IDX:
- informacije o posameznih naročnikih,
- vnašanje novih naročnikov,
- brisanje naročnikov,
- popravljanje informacij o naročnikih,
- izpis spiska naročnikov z osnovnimi podatki,
- izpis naslovov naročnikov na nalepke.

Poleg tega nudi še vsa potrebna navodila za uporabo.

Program INFO dela interaktivno - na bazi dialoga z uporabnikom - in omogoča dostop do posameznega naročnika po dveh ključih - priimku in ustanovi. Vezan je na programski paket FMS-11 (Form Management System) v sklopu operacijskega sistema Delta-N V1.2. Moduli FMS-11 omogočajo razvoj uporabniških form in njihovo izvajanje na terminalih tipa VT100 (Kopa 1000).

2. Programska lista s komentarji

Prikazana je celotna programska lista programa INFO, napisanega v visokem programirnem jeziku BASIC2-Plus. Iz nje je razvidna tudi zgradba indeksne datoteke CASOPI.IDX.

Kaj želite...:

navodila.....: 1
informacije..: 2
vnos.....: 3
brisanje.....: 4
popravljanje.: 5
izpise.....: 6
nalepke.....: 7
konec.....: 8

Vnesite zahtevo.....: 2

Slika 1. Menu

3. Izvajanje programa

Po klicu programa se vzpostavi dialog med programom in uporabnikom. Najprej program zahteva šifro. Po prejemu pravilne šifre se na zaslon izpiše spisek nalog, ki jih lahko program opravlja - menu (slika 1).

Poglejmo si primer popravljanja informacij o naročniku časopisa. Vnesemo zahtevo <5>, nakar nas program vpraša, po kakšnem ključu želimo iskati naročnika (slika 2). Recimo, da hočemo najti naročnika po priimku. Pritisnemo tipko <1> - ali pa kar <CR>, ker je ta zahteva predpostavljena. Pokaže se osnovna forma (slika 3). Ker smo izbrali dostop po priimku, je odzivni znak (prompt) že v levem delu prvega polja. Vnesemo priimek iskane osebe ali samo del priimka. Program nam vrne izpolnjeno celotno formo. Če oseba, ki je prikazana v formi, ni iskana oseba, pritisnemo <CR> in program nas vpraša, če želimo še iskati. V primeru pritrdilnega odgovora nam program izpolni formo s podatki o naslednjem naročniku, ki ima enak priimek oz. del priimka, kar smo pač podali kot vzorec za iskanje. Ko najdemo iskano osebo, se lahko lotimo popravljanja podatkov. Priimek je zaščiten in ga ne moremo spremeniti. Spreminjamo tako, da enostavno prepisujemo. Iz enega polja v drugega prehajamo z znakoma <TAB> - naprej - in <BCKSP> - nazaj. Popravljanje zaključimo s <CR> in sistem nas zopet vpraša, če želimo nadaljevati - naslednji naročnik - ali zaključiti popravljanje.

Kako želite iskati naročnika ?

Naročnika časopisa lahko iščete
na dva načina:

1. po priimku
2. po ustanovi

Odgovorite z ustrezno številko [1,2]....: 1

Slika 2. Podati moramo ključ za iskanje.

Priimek.....: Ime.....:
Ustanova.....:
Ulica.....:
Pošta.....: Kraj.....:
Država.....:

Telefon.....:

Število izvodov: 1 Naročnina.....:

Naročnina plačana do vklj. številke....: Letnik..:

Slika 3. Osnovna forma - podatki o naročniku

Podoben je dialog za iskanje informacij o naročnikih oz. brisanje naročnikov iz registra. Pri vnašanju novih naročnikov v register program seveda ne vpraša po ključu, saj v tem primeru ključ ni važen.

Manj radoveden je program, če želimo navodilo za delo s programom ali izpise, ki jih napiše na datoteko PRINT.TEM oz. NALEP.TEM. V teh primerih samo vnesemo ustrezno zahtevo.

Pri vnašanju podatkov v posamezna polja v formah program izvaja določene kontrole, kot na primer na tip vnesenega znaka, če je polje izpolnjeno, ipd. Če ugotovi neskladje, izpiše ustrezno sporočilo in čaka na pravilen vnos. Atribute posameznih polj vnaprej poljubno izberemo in definiramo z vsako posamezno formo. Poglejmo za zgled opis polja IZV (slika 4). Opis nam pove naslednje: Podatkovno polje ima dolžino enega znaka, vnos ni potreben (predpostavljena vrednost je ena), vnesen podatek mora biti dekadna cifra, pomeni pa število naročenih izvodov.

Po izstopu iz vsake od obdelav (zahteve 1 do 7) program ponovno izpiše menu. Interaktivno delo zaključimo tako, da vnesemo zahtevo <8>. Program se poslovi s formo, prikazano na sliki 5.

```
Field Descriptions

 6 23
Field PRI    of length 20
Display attributes: None
Field Type: Alphabetic
Clear character: ' '
Help text: 'priimek narocnika'
Picture value: 'AAAAAAAAAAAAAAAAAA'

 6 61
Field IME    of length 20
Display attributes: None
Field Type: Any character
Clear character: ' '
Help text: 'ime narocnika'
Picture value: 'XXXXXXXXXXXXXXXXXXX'

 8 23
Field ULI    of length 30
Display attributes: None
Field Type: Any character
Clear character: ' '
Picture value: 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'

 9 23
Field POSTA  of length 20
Display attributes: Some Required
Field Type: Any character
Clear character: ' '
Help text: 'postna stevilka'
Picture value: 'XXXXXXXXXXXXXXXXXXX'

 9 61
Field KRAJ   of length 20
Display attributes: Some Required
Field Type: Any character
Clear character: ' '
Picture value: 'XXXXXXXXXXXXXXXXXXX'
```

Slika 4. Primeri opisov polj

```
10 23
Field DRZ    of length 20
Display attributes: None
Field Type: Alphabetic
Clear character: ' '
Picture value: 'AAAAAAAAAAAAAAAAAA'

12 23
Field TEL1   of length 3
Display attributes: Autotab
Field Type: Numeric
Clear character: ' '
Picture value: '999'

12 27
Field TEL2   of length 6
Display attributes: None
Field Type: Numeric
Clear character: ' '
Picture value: '999999'

14 23
Field IZV    of length 1
Display attributes: None
Field Type: Numeric
Clear character: ' '
Help text: 'stevilo narocenih izvodov'
Picture value: '9'

14 61
Field NAR    of length 4
Display attributes: Some Required
Field Type: Mixed Picture
Clear character: ' '
Help text: 'narocnina-glede na vrsto narocnika'
Picture value: '999X'

16 47
Field STEV   of length 1
Display attributes: Full Required, Some Required
Field Type: Numeric
Clear character: ' '
Picture value: '9'

16 61
Field LET    of length 2
Display attributes: Full Required, Some Required
Field Type: Numeric
Clear character: ' '
Picture value: '99'
```

Slika 4. (nad.) Primeri opisov polj

```
Pozdravljeni!

Casopis INFORMATICA
Uredništvo
Ljubljana, Parmova 41
```

Slika 5. Pozdrav

```
10 !=============================================================!&
   !                                                             !&
   !                                                             !&
   !                                                             !&
   !        i n f o r m a c i j s k i     s i s t e m          !&
   !                                                             !&
   !                  NAROCNIKI CASOPISA INFORMATICA             !&
   !                                                             !&
   !                                                             !&
   !                     B.Blatnik, J.Santl                      !&
   !                                                             !&
   !                  Ljubljana, december 1981                   !&
   !                                                             !&
   !=============================================================!&
100!-------------------------------------------------------------!&
   !ZACETNA INICIALIZACIJA:                                      !&
   !                        napake                               !&
   !                        mape,baferji                         !&
   !                        FMS-ini                              !&
   !                        zacetne slike                        !&
   !                        vnos in test sifre                   !&
   !                                                             !
    ON ERROR GOTO 19000                          !napake        !
109 MAP (MAPA) A$=205                            !dolzina buferja!
110 MAP (MAPA) PRI$=20,                          !priimek       !&
               IME$=20,                          !ime           !&
               UST$=5#,                          !ustanova      !&
               ULI$=30,                          !ulica         !&
               POSTA$=20,                        !postna stevilka!&
               KRAJ$=20,                         !kraj          !&
               DRZ$=20,                          !drzava        !&
               TEL1$=8,                          !telefon       !&
               TEL2$=6,                          !telefon       !&
               IZV$=1,                           !stev.izvodov  !&
               NAR$=4,                           !narocnina     !&
               STEV$=1,                          !placano do stev.!&
               LET$=2                            !letnik        !
115 P$=STRING$(60%,32%)                          !buf           !\&
    D$=STRING$(2#,32%)                           !buf           !\&
120 CALL WTQIO(7@8%,5%,5%)                       !ataciram      !\&
    DIM I%(2000)                                 !bufer         !\&
    CALL FINIT(I%(),2000%) \ GOSUB 18000         !def.bufferja  !\&
    CALL FCHAN(6%) \ GOSUB 18000                 !kanal         !\&
    CALL FLOPEN("SY:PISIPA.FLB") \ GOSUB 18000   !odp.lib       !\&
    CALL FSHOW("PISSIF") \ GOSUB 18000                          !\&
    CALL FGET(P$,T%,"SIFRA") \ GOSUB 18000                      !\&
    SIFRA$=TRM$(P$)                              !sifra dostopa !\&
    IF SIFRA$="" THEN 124 ELSE                                  !\&
       CALL FPUTL("vasa sifra ni pravilna")                     !\&
       GOSUB 18000 \ GOSUB 2000                                 !\&
       GOTO 700
124 OPEN "CASOPI.IDX" AS FILE #3,                !podatkovni file!&
         ORGANIZATION INDEXED FIXED,             !organizacija  !&
         MAP MAPA,                               !mapa          !&
         PRIMARY PRI$ DUPLICATES,                !primarni kljuc!&
         ALTERNATE UST$ DUPLICATES CHANGES       !sekund. kljuc !&
   !.........................................................!&
   !                                                         !
130 !-------------------------------------------------------!&
   !glavni blok programa                      zacetek       !&
   !=====================                      =======       !&
   !                                                        !&
   !                    dispecer zelja                      !&
   !                                                        !&
   CALL FCLRSH("PISOZA") \ GOSUB 18000        !ozadje       !\&
   CALL FSHOW ("PISIZ1") \ GOSUB 18000        !menu         !\&
   CALL FGET  (D$,T%,"ZAHTVA") \ GOSUB 18000  !zahteva obdel.!\&
   K%=VAL(D$)                                 !kazalec obdel.!\&
   IF K%<1% OR K%>6% THEN 1000                !napaka       !
140 ON K% GOTO 200,300,400,500,600,800,900,700 !vejanje obdel.!&
   !.........................................................!&
   !                                                         !
200 !-------------------------------------------------------!&
   !navodilo                                  ========      !&
   !========                                  prikaz navodila!&
   !                                                        !&
   CALL FSHOW ("PISNAZ") \ GOSUB 18000        !navodilo     !\&
   CALL FGET  (D$,T%,"ZAHTVA") \ GOSUB 18000  !odgovor      !\&
   GOTO 130                                   !nazaj na menu!&
```

```
   !                                                         !
300 !-------------------------------------------------------!&
   !informacije                               ============  !&
   !============                              prikaz inform.!&
   !                                                        !
310 CALL FSHOW("PISINF") \ GOSUB 18000         !prikaz forme !\&
    CALL FSHOW("PISIZ2") \ GOSUB 18000         !izbor kljuca !\&
    CALL FGET(D$,T%,"ZAHTVA") \ GOSUB 18000                  !\&
    K%=VAL(D$)                                               !\&
    IF K%<1% OR K%>2% THEN                                   !\&
       CALL FPUTL("vasa zahteva ni pravilna")                !\&
       GOSUB 2000                                            !\&
       SLEEP 2%                                              !\&
       GOTO 310                                              !\&
          ELSE                                               !\&
    CALL FSHOW("PISSL1") \ GOSUB 18000         !slika 1      !
315 IF K%=1% THEN CALL FGET(P$,T%,"PRI")\GOSUB 18000
318 IF K%=2% THEN CALL FGET(P$,T%,"UST")\GOSUB 18000
320 PP$=TRM$(P$)                               !kraj. kljuca !\&
    KK%=K%-1%                                                !\&
    GET #3, KEY #KK% GE PP$                    !citanje prvega!\&
    GOTO 340                                   !preskok      !
330 GET #3                                     !citanje nasl.!\&
    CALL FSHOW("PISSL1") \ GOSUB 18000         !prikaz forme !
340 CALL FPUTAL(A$) \ GOSUB 18000              !prikaz starih!\&
    CALL FSHOW("PISDNO") \ GOSUB 18000         !forma da/ne  !\&
    CALL FGET(D$,T%,"DANE") \ GOSUB 18000      !se?-odgovor  !\&
    T$=TRM$(D$)                                !trancate     !\&
    IF T$="N" THEN 130 ELSE                    !NE           !&
       IF T$="D" THEN 330 ELSE 1020            !DA           !&
   !.........................................................!&
   !                                                         !
400 !-------------------------------------------------------!&
   !vnos                                      ============  !&
   !=====                                     vnos informacij!&
   !                                                        !&
    CALL FSHOW("PISVNO") \ GOSUB 18000         !for.glave    !\&
    CALL FSHOW("PISSL1") \ GOSUB 18000         !forma vnos   !\&
    CALL FGETAL(A$) \ GOSUB 18000              !vnos         !\&
    PUT #3                                     !zapis v file !\&
    CALL FSHOW("PISDNO") \ GOSUB 18000         !forma da/ne  !\&
    CALL FGET(D$,T%,"DANE") \ GOSUB 18000      !se?-odgovor  !\&
    T$=TRM$(D$)                                !trancate     !\&
    IF T$="N" THEN 130 ELSE                    !NE           !&
       IF T$="D" THEN 400 ELSE 1020            !DA           !&
   !.........................................................!&
   !                                                         !
500 !-------------------------------------------------------!&
   !brisanje                                  ============  !&
   !========                                  brisanje inform.!&
   !                                                        !&
    CALL FSHOW("PISBRI") \ GOSUB 18000         !prikaz brisanja!\&
    CALL FSHOW("PISIZ2") \ GOSUB 18000         !izbor kljuca !\&
    CALL FGET(D$,T%,"ZAHTVA") \ GOSUB 18000                  !\&
    K%=VAL(D$)                                               !\&
    IF K%<1% OR K%>2% THEN                                   !\&
       CALL FPUTL("vasa zahteva ni pravilna")                !\&
       GOSUB 2000 \ SLEEP 2% \ GOTO 500                      !\&
          ELSE                                               !\&
    CALL FSHOW("PISSL1") \ GOSUB 18000         !slika 1      !
505 IF K%=1% THEN CALL FGET(P$,T%,"PRI")       !citanje kljuca!\&
    GOSUB 18000
506 IF K%=2% THEN CALL FGET(P$,T%,"UST")                     !\&
    GOSUB 18000
507 PP$=TRM$(P$)                               !kraj. kljuca !\&
    KK%=K%-1%                                                !\&
    GET #3, KEY #KK% GE PP$                    !citanje prvega!\&
    GOTO 520                                   !preskok      !
510 GET #3                                     !citanje nasl.!\&
    CALL FSHOW("PISSL1") \ GOSUB 18000         !forma brisi  !
520 CALL FPUTAL(A$) \ GOSUB 18000              !prikaz starih!\&
    CALL FSHOW("PISDN1") \ GOSUB 18000         !forma da.ne  !\&
    CALL FGET(D$,T%,"DANE") \ GOSUB 18000      !se?-odgovor  !\&
    T$=TRM$(D$)                                !trancate     !\&
    IF T$="N" THEN 530 ELSE                    !NE           !&
       IF T$<>"D" THEN 1020 ELSE               !<>DA         !\&
       CALL FSHOW("PISDN2") \ GOSUB 18000      !forma da/ne  !\&
       CALL FGET(D$,T%,"DANE") \ GOSUB 18000   !se?-odgovor  !\&
       T$=TRM$(D$)                             !trancate     !\&
```

Slika 6. Programska lista programa INFO. Celotna datoteka CASOPI.IDX je razvidna iz MAP in ORDER stavkov.

```
      IF T$="N" THEN 130 ELSE                          !NE              !&
         IF T$<>"D" THEN 1020 ELSE                     !<>DA            !\&
         DELETE #3                                     !brisanje        !\&
         CALL FSHOW("PISDNO") \ GOSUB 18000            !forma da/ne     !\&
         CALL FGET(D$,T%,"DANE") \ GOSUB 18000         !se?-odgovor     !\&
         T$=TRM$(D$)                                   !trancate        !\&
         IF T$="N" THEN 130 ELSE                       !NE              !&
            IF T$="D" THEN 500 ELSE 1020               !DA              !
530 CALL FSHOW("PISDN3") \ GOSUB 18000                 !forma da/ne     !\&
   CALL FGET(D$,T%,"DANE") \ GOSUB 18000               !se?-odgovor     !\&
   T$=TRM$(D$)                                         !trancate        !\&
   IF T$="N" THEN 130 ELSE                             !NE nazaj na men!&
      IF T$="D" THEN 510 ELSE 1020                     !DA              !&
   !-------------------------------------------------------------------!&
   !                                                                   !
600 !-------------------------------------------------------------------!&
   !popravki                                           ----------------!&
   !=======                                            popravljanje inf!&
   !                                                                   !&
   CALL FSHOW("PISPOP") \ GOSUB 18000                  !prikaz forme    !\&
   CALL FSHOW("PISIZZ") \ GOSUB 18000                  !izbor kljuca    !\&
   CALL FGET(D$,T%,"ZAHTVA") \ GOSUB 18000                              \&
   K%=VAL(D$)                                                           \&
   IF K%<1% OR K%>2% THEN                                               &
      CALL FPUTL("vasa zahteva ni pravilna")                            \&
      GOSUB 2000 \ SLEEP 2% \ GOTO 600                                  &
         ELSE                                                           &
   CALL FSHOW("PISSL1") \ GOSUB 18000                  !slika 1         !
605 IF K%=1% THEN CALL FGET(P$,T%,"PRI")               !citanje kljuca  !\&
   GOSUB 18000                                                          \&
606 IF K%=2% THEN CALL FGET(P$,T%,"UST")                                \&
   GOSUB 18000
607 PP$=TRM$(P$)                                       !kraj. kljuca    !\&
   KK%=K%-1%                                                            \&
   GET #3, KEY #KK% GE PP$                             !citanje records!\&
   GOTO 620                                            !preskok         !
610 CALL FSHOW("PISSL1") \ GOSUB 18000                 !prikaz forme    !
620 CALL FPUTAL(A$) \ GOSUB 18000                      !prikaz starih   !\&
   TEMP$=A$                                            !zacasno         !\&
   CALL FGETAL(A$) \ GOSUB 18000                       !vnos novih      !\&
   IF TEMP$<>A$ THEN 630 ELSE                          !popravek        !&
      CALL FSHOW("PISDN3") \ GOSUB 18000               !forma da/ne     !\&
      CALL FGET(D$,T%,"DANE") \ GOSUB 18000            !se?-odgovor     !\&
      T$=TRM$(D$)                                      !trancate        !\&
      IF T$="N" THEN 130 ELSE                          !nazaj na menu   !&
         IF T$<>"D" THEN 1020 ELSE                     !<>DA            !&
         GET #3                                        !citanje         !\&
         GOTO 610                                      !naprej          !
630 UPDATE #3                                          !update          !\&
   CALL FSHOW("PISDNO") \ GOSUB 18000                  !forma da/ne     !\&
   CALL FGET(D$,T%,"DANE") \ GOSUB 18000               !se?-odgovor     !\&
   T$=TRM$(D$)                                         !trancate        !\&
   IF T$="N" THEN 130 ELSE                             !NE              !&
      IF T$="D" THEN 600 ELSE 1020                     !<>DA            !&
   !-------------------------------------------------------------------!&
   !                                                                   !
700 !-------------------------------------------------------------------!&
   !konec                                              ----------------!&
   !=====                                              konec obdelav   !&
   !                                                                   !&
   CALL FSHOW("KONEC")                                 !forma konec     !\&
   GOTO 32000                                          !na END          !&
   !-------------------------------------------------------------------!&
   !                                                                   !
800 !-------------------------------------------------------------------!&
   !izpis podatkov                                                     !&
   !=============                                                      !&
   !                                                                   !

810   OPEN "PRINT.TEM" FOR OUTPUT AS FILE #2%
813   STEV%=0%                                         !zap.st.narocnika!
815   RESTORE #3%
820   GET #3% \ STEV%=STEV%+1%                         !nov zapis, inkr.!
821   X$=TRM$(IME$)
822   Y$=TRM$(PRI$)
824   S$=" "\X$=X$+SP$+Y$
826   PRINT #2%\PRINT #2%\PRINT #2%
828   PRINT #2%,X$                                     !ime in priimek  !
830   X$=TRM$(UST$)
      IF LEN(X$) > 1 THEN PRINT #2%,X$                 !ustanova        !
832   X$=TRM$(POSTA$)
833   Y$=TRM$(KRAJ$)
834   Z$=EDIT$(ULI$,16%)
835   X$="  "\YY$=X$+SP$+Y$+B$+Z$
836   PRINT #2%, YY$                                   !posta, naslov   !
838   X$=TRM$(DRZ$)
839   IF LEN(X$)>0 THEN PRINT #2%,X$                   !drzava          !
842   Y$="/"
843   X$=STEV$+Y$+LET$
844   PRINT #2%, "zap.st.narocn...",STEV%
845   PRINT #2%, "stev.izvodov... ",IZV$
846   PRINT #2%, "narocnina..... ",NAR$
847   PRINT #2%, "placano do..... ",X$
870   GOTO 820                                                           &
   !-------------------------------------------------------------------!&
   !                                                                   !
900 !-------------------------------------------------------------------!&
   !izpis naslovov na nalepke                                          !&
   !=========================                                          !&
   !                                                                   !&
      OPEN "NALEP.TEM" FOR OUTPUT AS FILE #2%
905   RESTORE #3%
910   GET #3%
912   IJ%=VAL(IZV$)
913   FOR J%=1% TO IJ%                                 !st.izpis.nalepk!
915   PRINT #2%\ PRINT #2%\ PRINT #2%                  !zacetna praznina!
920   X$=IME$+SPACE$(1%)+PRI$                          !ime in priimek  !\&
      Y$=EDIT$(X$,16%)                                                   \&
      PRINT #2%, Y$                                                      \&
      XX$=TRM$(UST$)                                   !ustanova        !\&
      IF LEN(XX$) > 1 THEN PRINT #2%,XX$
925   X$=EDIT$(ULI$,16%)                               !ulica in stev.  !\&
      PRINT #2%, X$                                                      \&
      X$=POSTA$+SPACE$(1%)+KRAJ$                       !kraj            !\&
      Y$=EDIT$(X$,16%)                                                   \&
      PRINT #2%, Y$                                                      \&
      X$=TRM$(DRZ$)                                    !drzava          !\&
      PRINT #2%,X$
930   FOR I=1 TO 4                                                       \&
      PRINT #2%                                                          \&
      NEXT I
932   IF LEN(XX$) < 2 THEN PRINT #2%
933   NEXT J%
940   GOTO 910                                                           &
   !-------------------------------------------------------------------!&
   !
1000 !                                                 nap.odgovori     !
1010  CALL FPUTL("vasa zahteva ni pravilna")           !napaka          !\&
      GOSUB 18000                                      !test napake     !\&
      GOSUB 2000                                       !zvocni signal   !\&
      SLEEP 2%                                         !pauza           !\&
      GOTO 130                                         !nazaj na menu   !
1020  CALL FPUTL("vas odgovor ni pravilen-[D/N]")      !napaka          !\&
      GOSUB 18000                                      !test napake     !\&
      GOSUB 2000                                       !zvocni signal   !\&
      SLEEP 2%                                         !pauza           !\&
      GOTO 130                                         !nazaj na menu   !
1030  CALL FPUTL("vasa sifra ni pravilna")             !napaka          !\&
      GOSUB 18000                                      !test napake     !\&
      GOSUB 2000                                       !zvocni signal   !\&
      SLEEP 2%                                         !pauza           !\&
      GOTO 700                                         !konec           !
2000                                                   !zvocni signal   !
2010  FOR ZVOK=1 TO 2                                  !zanka-st.zvok   !\&
      PRINT CHR$(7%);                                  !zvok            !\&
      NEXT ZVOK                                        !end zanke       !\&
      RETURN                                           !povratek iz sub!
18000 !-------------------------------------------------------------------!&
      !TMS napake                                      napake           !&
      !==========                                      ----------------!&
      !          testiramo status                                       !&
      !          teksti napak so v formah:                              !&
      !                    FSTST1.FRM in                                !&
      !                    FSTST2.FRM                                   !&
      !                                                                 !
18010 CALL FSTAT(S%)                                   !test statusa    !\&
      IF S%=0% THEN RETURN                             !ni napake       !
18030 S$=NUM1$(S%)                                     !pretvorba v str!\&
      CALL FPUTL(S$)                                   !izpis           !\&
```

Slika 6. (cont.) Programska lista programa INMO

```
      FOR ZVOK=1 TO 2                          !zanka-st.zvok   !\&
      PRINT CHR$(78);                          !zvok            !\&
      NEXT ZVOK                                !end zanke       !\&
      SLEEP 2%                                 !pauza           !\&
      GOTO 700                                 !konec           !
18040 RETURN                                   !povratek        !&
18050 !.....................................................     !&
      !                                                        !
19000 !--------------------------------------------------------!&
      !basic plus 2 napake                     napake         !&
      !========================                ---------------!&
      !                                                        !
19010 IF (ERR=11 AND ERL=320) OR               !konec datoteke !&
         (ERR=11 AND ERL=330) OR               !konec datoteke !&
         (ERR=11 AND ERL=500) OR               !konec datoteke !&
         (ERR=11 AND ERL=510) OR               !konec datoteke !&
         (ERR=11 AND ERL=600) OR               !konec datoteke !&
         (ERR=11 AND ERL=820) OR               !konec datoteke !&
         (ERR=11 AND ERL=910) OR               !konec datoteke !&
         (ERR=11 AND ERL=620) THEN             !konec datoteke !&
        CALL FPUTL("konec datoteke!")          !izpis napake   !\&
        GOSUB 18000                            !test napake    !\&
        GOTO 19990                             !zvocni signal  !
19020 IF (ERR=155 AND ERL=320) OR              !ni records     !&
         (ERR=155 AND ERL=330) OR              !ni records     !&
         (ERR=155 AND ERL=500) OR              !ni records     !&
         (ERR=155 AND ERL=510) OR              !ni records     !&
         (ERR=155 AND ERL=600) OR              !ni records     !&
         (ERR=155 AND ERL=620) THEN            !ni records     !&
        CALL FPUTL("tega podatka ni!")         !izpis napake   !\&
        GOSUB 18000                            !test napake    !\&
        GOTO 19990                             !zvocni signal  !
19030 IF ERR=130 AND ERL=630 THEN              !sprem.primka   !&
        CALL FPUTL("primka ne smete spremeniti!")!izpis napake!\&
        GOSUB 18000                            !test napake    !\&
        GOTO 19990                             !zvocni signal  !
19040 IF ERR=143 AND ERL=320 THEN              !null karakter  !&
        CALL FPUTL("podajte primek!")          !izpis napake   !\&
        GOSUB 18000                            !test napake    !\&
        GOTO 19990                             !zvocni signal  !
19130 TEXT$="napaka:"+NUM1$(ERR)+" v vrstici:" !opis           !&
      +NUM1$(ERL)                              !napake         !\&
      CALL FPUTL(TEXT$)                        !prikaz         !\&
      GOSUB 18000                              !test napake    !
19990 GOSUB 2000                               !zvocni signal  !\&
      SLEEP 2%                                 !pauza          !\&
      RESUME 130                               !nazaj menu     !
19999 !.....................................................     !&
      !                                                        !
32000 END
>
```

Slika 6.(nad.) Programska lista prog. INFO

NOVICE IN ZANIMIVOSTI

OPERACIJSKI SISTEM SHIVA

SHIVA, produkt kalifornijske firme Omega Research, je prvi sprotni mikroračunalniški multiprogramski multiuporabniški virtualni operacijski sistem. S prehodom na SHIVA operacijski sistem lahko še naprej uporabljamo vse programe, ki so prej tekli na CP/M ali CDOS operacijskem sistemu, tudi podatkovne zbirke so kompatibilne. Podpira tako nove 16-bitne mikroprocesorje (8086,8088,Z8000-1 in Z8000-2) kot standardne 8-bitne mikroprocesorje (8080,8085,6800,6502 in Z80).

SHIVA ima tri načine delovanja (polled, interrupt-driven, oboje), kar omogoča šestnajstim uporabnikom, sprotnim poslom ali mešanici uporabnikov in poslov optimalno učinkovitost in hitrost. Vsak izmed šestnajstih uporabnikov ali poslov lahko uporablja virtualni RAM - pomnilnik nad RAMom, dostopen z direktnim naslavljanjem, sistem ga dinamično dodeljuje.

Uslužnostni programi, ki se najpogosteje uporabljajo, so rezidentni v RAMu. Hitrost izvajanja programov je zato v splošnem mnogo boljša kot pri drugih multiuporabniških sistemih, ki bazirajo na popravljenih in dopolnjenih enouporabniških. SHIVA je bil že od začetka zamišljen kot multiuporabniški multiprogramski sistem.

SHIVA dovoljuje izbor do 16 8M-zlogovnih bank pomnilnika tipa RAM v SHIVA 1.6 Z8000-1 verziji oz. 16 64K-zlogovnih bank za Z80 in druge 8-bitne procesorje. Tako so lahko posamezni programi na Z8000-1 veliki do 128M zlogov. Multiuporabniške ali multiprogarmske operacije se lahko odvijajo v isti fizični ali logični RAM banki. Jedro sistema navadno zaseda pomnilnik od naslova E000h navzgor, lahko pa ga uporabnik prekonfigurira na katerikoli podsklop velikosti 2K banke 00.

Vse novejše verzije SHIVA (od 1.2/1.52 navzgor, vse 1.6) so kompatibilne z bodočim mrežnim operacijskim sistemom SHIVANET. SHIVANET bo dovoljeval 1024 podmrež, vsaka podmreža pa bo lahko sestavljena iz 16 multiuporabniških sistemov, od katerih bo vsak sistem lahko obvladal 16 uporabnikov ali poslov. Mrežni sistem bo omogočal sprotno multiprocesiranje, zajemanje podatkov in visoko hitrost prenosa med uporabniki (do 79 KBaudov) in bo dovoljeval obravnavanje 262144 uporabnikov ali poslov na celotno mrežo. SHIVANET bo dovoljeval X.25, BISYNC in SDLC protokole in s tem komunikacije z že obstoječimi mrežami. SHIVANET bi naj bil predstavljen v začetku leta 1982.

SHIVA vključuje tudi večnivojski varnostni sistem z gesli za zbirke, podatke in vzpostavljanje zveze s sistemom.

SHIVA vsebuje celo vrsto komand za razne uslužnostne dejavnosti, kar omogoča uporabniško manipuliranje in nadzorovanje programov in podatkovnih zbirk. Imamo sistemske komande za nadzor izvrševanja (vejanje,zanke), brisanje, prenos, kopiranje in preimenovanje zbirk, oblikovanje formatiranje diska, inicializacija, preimenovanje celotnega diska, vrstično orientiran urejevalni editor z velikimi in malimi črkami, imena zbirk dolga 16 znakov, možnost paketne obdelave, foreground/background izvrševanje, branje, pisanje in izvajanje zbirk in programov.

SHIVA ukazi pokrivajo tudi nadzor nad vhodom in izhodom, dodeljevanje in sprostitev [illegible], dodeljevanje in konfiguracija [illegible] RAM, povezovanje, nalaganje [illegible] programov, dodeljevanje in sprostitev [illegible] uporabnikom in poslom ter [illegible] dostopom do podatkovnih zbirk.

SHIVA podpira ločene direktorije [illegible] uporabnika ali posel) kakor [illegible] torij. Direktoriji so zavarovani [illegible] da vsakomur niso dostopni.

SHIVA uporabnikom je na voljo [illegible] vedena ura, po želji pa [illegible] tudi programsko izvedeno.

SHIVA daje zaenkrat najnočnejše [illegible] koriščа moč in hitrost [illegible] winchesterske tehnologije [illegible] riran pristop za bodoče [illegible] so 32-bitni in večji mikroprocesorji.

B. [illegible]

PROCESORJI Z OPERACIJSKIM SISTEMOM

Intel je predstavil dva procesorja z operacijskim sistemom: iAPX 86/30 in iAPX 88/30. Gre za dva para čipov, ki vsebujeta [illegible] sor 8086 ali 8088 in koprocesor [illegible] sistemom. Nova komponenta [illegible] eden od obeh procesorskih [illegible] hardwarsko in softwarsko [illegible] gljivega sprotnega večprocesnega [illegible] sistema.

iAPX 86/30 in 88/30 razširjata [illegible] turo procesorjev 8086 in [illegible] polnoma pretestiranih [illegible] sistema, ki so implemen[illegible]

Izbrane operacije tvorijo majhno vendar izjemno močno množico ukazov, ki lahko služijo pri enostavnih proizvodih kot kompletni operacijski sistem ali kot osnova za Intelov sprotni operacijski sistem iRMX 86 oz.lasten operacijski sistem. Te osnovne operacije kreirajo, brišejo in obdelujejo pet novih sistemskih podatkovnih tipov in s tem dodajo k obstoječim zmožnostim sistemov iAPX 86 oz. iAPX 88 še obdelavo prekinitev, posredovanje sporočil, sinhronizacijo procesov in upravljanje s pomnilnikom,

80130 vsebuje še prekintveni kontroler, 16bitni sistemski časovnik, 16bitni časovnik za zakasnitve in generator hitrosti serijskega prenosa (baud rate). Te hardwarske funkcije zamenjajo v povprečju 10 LSI vezij oz. več deset TTL komponent in zagotavljajo vso hardwarsko podporo za operacije operacijskega sistema.

Ta Intelov pristop je izposojen pri programskih konceptih velikih računalnikov. Mikroračunalniški operacijski sistem se gradi okrog izbranih osnovnih operacij jedra. Sedaj lahko namesto velikega programa, ki mora koordinirati in obdelovati N dogodkov, N programerjev ločeno napiše vsak po en program za posamezni dogodek. Ti programi pa uporabljajo skupne operacije jedra, ki koordinirajo sprotno obdelavo vseh dogodkov.

Ta pristop daje več ugodnosti. Prva je le skrajšanje časa razvijanja. Ker razvijalcem ni treba pisati in testirati multiprogramskega jedra, se lahko posvetijo samo uporabniškim programom.

Druga ugodnost je ta, da omogoča tak pristop kasnejše preoblikovanje in izboljševanje programskih paketov.

Vezje 80130 se priključi direktno na lokalno vodilo mikroprocesorjev iAPX 86/10 ali iAPX 88/10 in avtomatsko ugotovi s katerim delu. Vsi procesorjevi dosegi se izvršijo brez čakalnih ciklov pri procesorjevi uri 5 ali 8 MHz. Na voljo je vseh 24 načinov naslavljanja procesorjev iAPX 86/88.

Programski paket "Operating System Processor Support Package" poenostavlja konfiguriranje sistema s tem novim vezjem (80130). Ta paket teče na sistemu Intellec III oz. na vsakem sistemu z iRMX 86 operacijksim sistemom in gibkimi diski.

D.Novak

VSEBINA LETNIKA 1981

Batista P.,Kraigher M. št.1,str.12;
Paralelno procesiranje IBM-SNA rač.mreži.

Brajović S.-Bratanović,Džonova B.-Jerman - Blažič,št.4, str.24;
Standardi i politika standardizacije u oblasti informatike.

Exel M.,Prijatelj F. št.2,str.8;
Programiranje sprotnih in vgnezdenih sistemov:procesi v ADI.

Hadjina N. št.3,str.47;
Postupci za sprečavanje potpunog zastoja i permanentnog blokiranja sistema na modelu grafa sistema.

Hadjina N.,Čerić V. št.4,str.54;
Simulacioni model za evaluaciju performansi računarskih sistema.

Holodkov V. št.2,str.46;
Prikaz razvoja operativnih sistema.

Holodkov V. št.2,str.61;
Jedna metoda procene vremena trajanja prenosa podataka.

Ivančić N.,Krtolica B.,Zakrajšek E. št.1,str.56
Cromemco CS-3 mikroračunalo.

Jefić M.V. št.1,str.34;
Otkrivanje i popravljanje grešaka u računskim memorijama(ECC).

Kovačević M. št.4, str.28;
Sistemska obnova u uslovima realnog vremena.

Kuštrin B.,Barle J. št.1,str.28;
Elementi paralelnega procesiranja v operacijskem sistemu računalnika IBM 8100.

Lenarčič J. št.3,str.25;
Avtomatsko prepoznavanje predmetov v robotiziranem procesu emajliranja.

Lončar j. št.3,str.33;
O jednoj metodi proračuna optimalnog balansa.

Lončar J. št.3,str.60;
Primjena inverzione metrike kod optimalnog balansiranja.

Lončar J. št.4, str.50;
Primjena leksikografske metrike sa proračun debalansa lopatica zrakoplovnih turbina.

Lončar J. št.4, str.36;
Primjena transpozicione metrike sa proračun optimalnog balansa.

Mahnič V.,Vilfan B. št.1,str.42;
Program za oblikovanje besedil.

Mihovilović B.,Šilc J.,Kolbezen P. št.1,str.47;
Mehurčni pomnilniki III.

Mijajlović Ž.,Jovanović A. št.3,str.38;
Turing Machines as a Prog.Language.

Murn R.,Peček D. št.2,str.66;
Sodobni dinamični pomnilniki za mikrorač.

Novaković Ž.,Popović B. št.2,str.56;
Pristop k rešitvi medračunalniške komunikacije v sistemih z distribuir.krmiljenjem.

Novak F.,Dobrin A.,Ropret B.,Blaznik T.št.2,59;
Testiranje enot mikroračunalnika v proizv.

Popović B. št.2,str.19;
Organizacija distribuiranega krmiljenja sistema Iskra 2000.

Prešern S.,Ozimek I.,Špegel M. št.4,str.33;
Mikroprocesorsko vodenje senzorskega sistema za robotsko varjenje.

Pyle I.C. št.3,str.4;
Using ADA for Specification and Design.

Rogač M.,Hafner D., št.2,str.44;
Mali razvojni sistem za mikroproc.68000.

Smailagić A. št.3,str.51;
On Solutions for Some Open Problems in the Design of Multiproc.Operating Systems.

Smolej V. št.4, str.39;
Plus ca change,plus c'est la meme chose.

Šilc J.,Mihovilović B.,Kolbezen P.št.4,str.60;
Mehurčni pomnilniki IV del.

Šubelj M.,Trobec R.Korenini J. št.3,str.55;
Primer komunikacijskega protokola.

Švab B. št.1,str.61;
Mikroračunalniško krmiljenje avto.motorja.

Turski V.M. št.2,str.3;
Issues in Large Program Design and Implem.

Tvrdy I.,Martinec M.,Reinhardt R. št.4, str.68;
Peto republiško tekmovanje srednješolcev s področja računalništva.

Uratnik A.,Peternelj L.,Kožuh J. št.2,str.26;
Novi računalniški sistem.

Vitas D. št.2,str.11;
Podela na slogove srpskohrvatskih reči.

Volk M.A. št.4,str.43;
Fortran 8X - Revizija Fortran-a 77.

Železnikar A.P. št.1,str.4;
Možnosti razvoja mikroračunalniške tehnologije v SFRJ.

Železnikar A.P. št.1,str.16;
Jezik PL/I in mikroračunalniki II.

Železnikar A.P. št.2,str.32;
Jezik PL/I in mikroračunalniki III.

Železnikar A.P. št.3,str.3;
Tiha revolucija.

Železnikar A.P. št.3,str.63;
Uvod v CP/M.

Železnikar A.P. št.4,str.9;
Uvod v CP/M.

št.4,str.1;
Pojav informacijskega gospodarstva.

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 polja oštevilčite strani članka s svinčnikom, tako da jih je mogoče zbrisati.

INSTRUCTIONS FOR PREPARATION OF A MANUSCRIPT

Authors are invited to send in the address and short summary of their articles and indicate the approximate 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 apaque, 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.

INFORMATICA, Journal Headguarters
Parmova 41, 61000 Ljubljana, Yugoslavia

Please enter my subscription to INFORMATICA and send me the bill.

Annual subscription price: companies US $ 22, individuals US $ 7,5.

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

.....

SLOVENSKO DRUŠTVO I N F O R M A T I K A , Parmova 41, 61000 Ljubljana

I Z J A V A

Podpisani ..... izjavljam, da želim vstopiti v Slovensko društvo I N F O R M A T I K A in da sprejemam Statut društva (objavljen v časopisu Informatica štev. 1, stran 67, letnik 1981).

P o d p i s :

.....

Ime in priimek: .....

Točen naslov: .....

Datum: .....

informatica 82

Mednarodna razstava računalniške in informacijske tehnologije

Gospodarsko razstavišče Ljubljana
10. do 14. maj 1982

Informatica '82 bo skupna sejemsko-strokovna manifestacija raziskovalnih, proizvajalnih in zastopniških delovnih organizacij, uporabnikov in strokovnih društev s področja računalništva in informatike.

Razstava Informatica je namenjena
- proizvajalcem računalniških naprav in sistemov
- uporabnikom informacijskih sistemov in
- vsem, ki uvajajo automatizacijo z uporabo računalniških sistemov.

RAZSTAVNI PROGRAM

1. Elementi naprav za obdelavo podatkov
- mikroprocesorji
- periferni procesorji
- integrirana vezja in drugi aktivni elementi
- konektorji, kabli, podnožja
- pasivni elementi

2. Enote sistemov za obdelavo podatkov
- centralne enote
- pomnilniške enote (dinamične, statične, mehanične)
- vhodno/izhodno kontrolne enote
- druge podsistemske enote
- usmerniki

3. Periferne naprave in terminali
- enote za čitanje luknjanih kartic
- enote za čitanje in luknjanje traku
- magnetnotračne enote
- enote za pogon magnetnih kaset
- diskovne enote
- pogoni za gibke diske
- optični čitalniki znakov
- čitalniki rokopisov
- znakovni, vrstični in bločni tiskalniki
- video terminali

4. Sistemi za obdelavo podatkov
- sistemi za splošno obdelavo podatkov
- pisarniški sistemi
- razvojni sistemi
- poslovni sistemi
- laboratorijski sistemi
- procesni sistemi
- vojni sistemi
- osebni in domači sistemi
- sistemi za zajemanje podatkov

. Programska oprema

osnovna in sistemska programska opremá
uporabniška programska oprema za poslovne sisteme
uporabniška programska oprema za vodenje procesov
komunikacijska programska oprema
uporabniška programska oprema za tehnične in znanstvene namene

. Aplikacije računalniških sistemov

- telekomunikacijski sistemi
- vodenje elektroenergetskih sistemov
- zajemanje podatkov
- računalniška grafika
- robotika
- vodenje tehnoloških procesov
- bančni sistemi

. Naprave za zbiranje in prenos podatkov

- modemi za prenos podatkov
- multiplekserji
- kontrolne in merilne naprave
- naprave za komutacijo

8. Oprema za proizvodnjo računalnikov

- kazalčni merilni instrumenti
- elektronski merilni instrumenti
- logični analizatorji
- sistemi za načrtovanje vezij
- sistemi za izdelavo tiskanih vezij
- sistemi za testiranje tiskanih vezij
- sistemi za proizvodnjo integriranih vezij

9. Mediji za zapis podatkov

- magnetni trakovi
- magnetni diski, gibki diski
- magnetne kasete
- formularji na neskončnem traku
- papirni trak

10. Strokovna literatura

- knjige
- revije in časopisi
- uporabniška dokumentacija

Simpozij INFORMATICA 82 je mednarodni simpozij za računalniško tehnologijo in probleme informatike, ki bo v organizaciji Slovenskega društva Informatika, Elektrotehniške zveze Slovenije in Gospodarskega razstavišča v dneh od 10. do 14. maja 1982, obravnaval pereče problematike s področja računalniških znanosti, tehnologije in uporabe. Sodelovanje z mednarodnimi strokovnimi organizacijami bo zagotovilo visoko kakovost referatov in posvetov, kjer bodo sodelovali ugledni mednarodni in domači izvedenci.

Tako bosta razstava in simpozij Informatica '82 srečanje strokovnjakov, proizvajalcev, uporabnikov in drugih interesentov z računalniške informacijske panoge.

INFORMACIJE IN PRIJAVE:
Gospodarsko razstavišče Ljubljana
61000 Ljubljana, poštni predal 413, Titova 50
telefon (061) 311-022, 310-930, 311-232
telex 31 127 gr yu