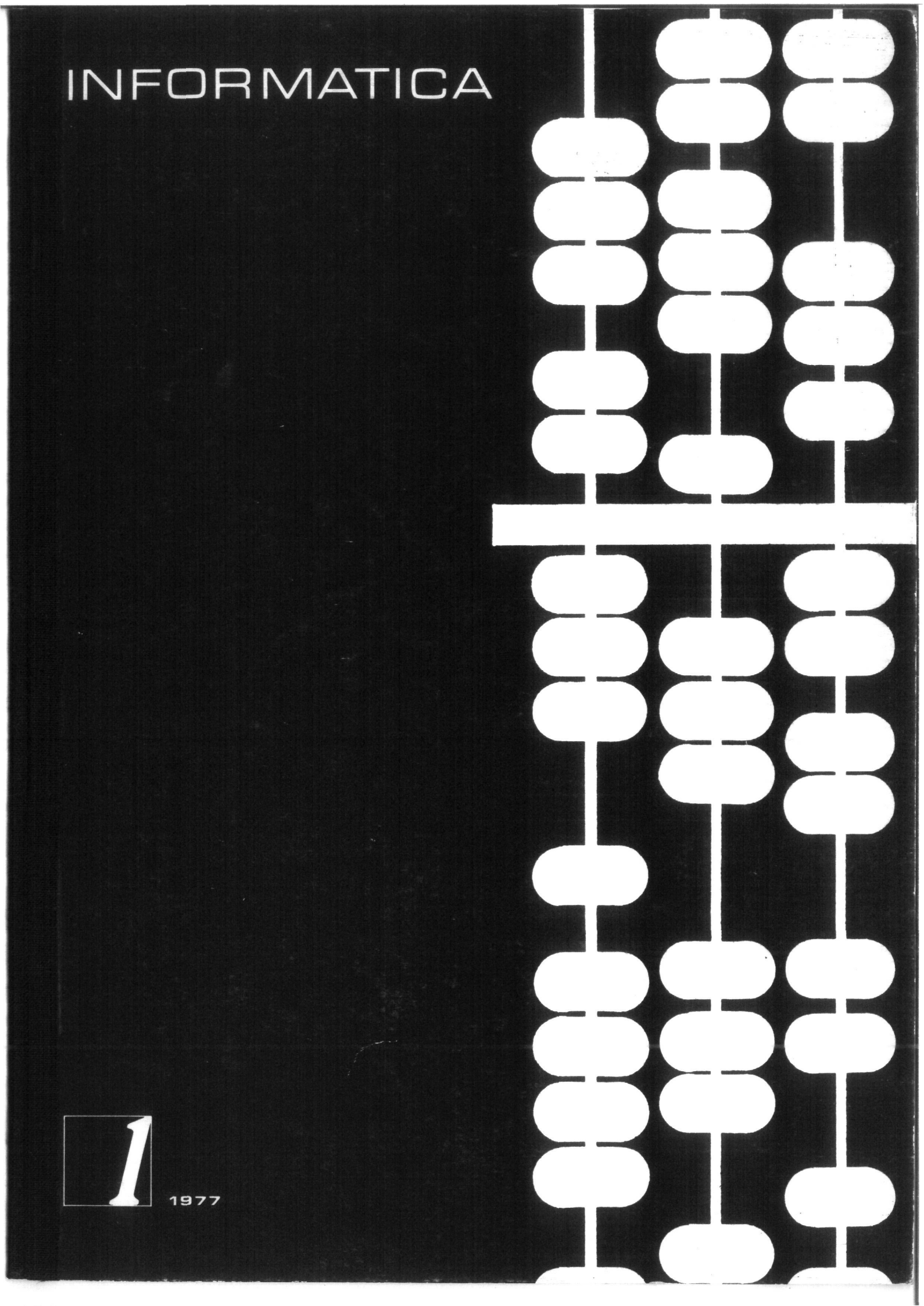
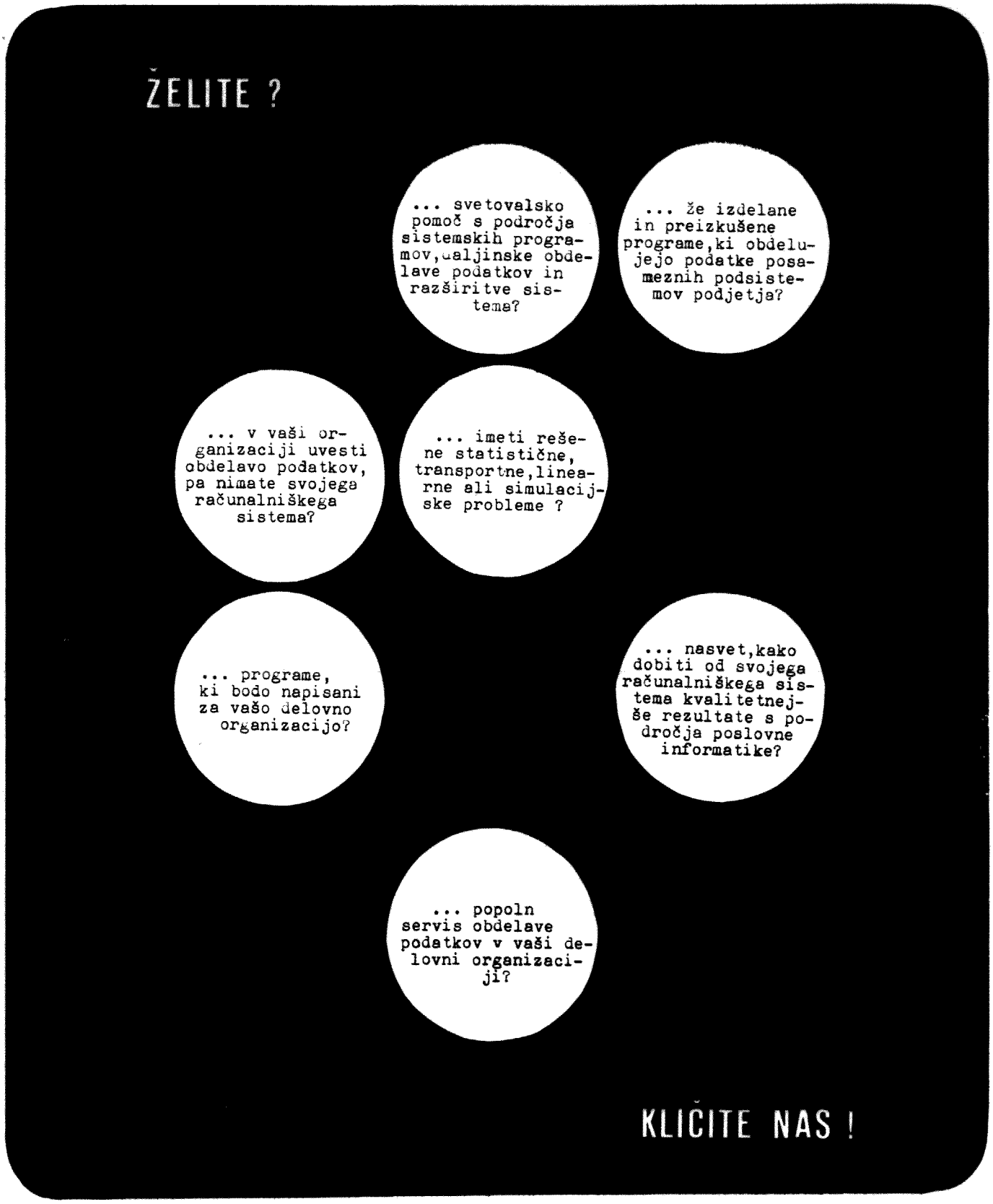
INFORMATICA



1

1977



UNIVERZA v LJUBLJANI

INSTITUT "JOŽEF STEFAN" odsek za uporabno matematiko

61001 ljubljana jamova 39 jugoslavija

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

INFORMATICA

Journal of Computing and Informatics

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

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

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

Printed by: Tiskarna Kresija, Ljubljana

Design: T. Simončič

volume I. 1977 - No 1

CONTENTS

INFORMATICA

Časopis za tehnologijo računalništva in proble - me informatike
Časopis za računarsku tehnologiju i probleme informatike
Spisanie za tehnologija na smetanjeto i problemi od oblasta na informatikata

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

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

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

Na podlagi mnenja Republiškega sekretariata za prosveto in kulturo št. 4210-151/77 z dne 4.5.1977, je časopis INFORMATICA strokovni časopis, ki je oproščen temeljnega davka od prometa proizvodov.

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

letnik I. 1977 - št. 1

VSEBINA

o časopisu informatica

a. p. železnikar

Časopis Informatica naj bi postal strokovno glasilo za računalniško tehnologijo in probleme informatike. Trenutna usmerjenost časopisa odraža naše stanje, potrebe in interese ter se zrcali v naslednjih področjih:

mikro računalniki, programiranje, operacijski sistemi, materialna oprema, terminologija, informacijski sistemi, procesna informatika, robotika, umetna inteligenca, računalništvo v humanističnih in družbenih vedah, vzgoja in izobraževanje, literatura in srečanja, študentska vprašanja, novice in zanimivosti, novice založništva.

Prvotno je bil časopis Informatica zamišljen kot publikacija za tehnike, inženirje, amaterje in druge delavce na področju računalniške tehnologije, ki naj bi obravnaval predvsem majhne sisteme, t.j. take sisteme, ki so lahko predmet domače računalniške proizvodnje, razvoja in raziskav. Ob takih razmišljanih pa se je nujno odprl računalniški in informatični kompleks, ki sega na uporabniška področja tudi v širša vprašanja informacijskih sistemov, kadrov, usmerjenosti in smotrne uporabe.

Strokovni nivo časopisa je sicer večplasten (visok, srednji, nizek), vendar so zaželeni predvsem prispevki srednjega nivoja, povezani s konkretnimi, praktičnimi rezultati in izdelki. Na primer: metodološki opisi programov (zlasti sistemskih, uporabniških s splošnim pomenom), mikroračunalniških programov in vezij, računalniških sistemov, aplikacij v robotiki itd. so v časopisu izredno zaželeni. To velja še zlasti za prispevke, ki obravnavajo pregledno novo tehnologijo ter za novice in zanimivosti iz tuje in predvsem domače proizvodnje računalnikov in sestavnih delov.

Področji materialne (hardware) in programske opreme (software) sta enakopravni: računalniške konfiguracije, arhitektura, procesorji, novi materialni elementi so enako zanimivi kot opisi programov (monitorji, igre, programirna podpora), pri čemer želimo krajše, zanimive programe (zlasti za mikro računalnike) tudi v celoti objaviti.

Pregledni prispevki s področja tehnologije in uporabe na posameznih področjih informatike bodo zanimivi le tedaj, če vključujejo tudi sintezo na naše možnosti, našo prakso in naša stališča. V časopisu želimo imeti polemične prispevke, ki zadevajo zlasti vprašanja izobraževanja, strokovnega izpopolnjevanja, organizacije proizvodnje in informacijskih sistemov ter medsebojnih odnosov. V tem okviru je časopis odprt našim javnim delavcem, gospodarskim orgamizacijam in družbenim ustanovam, ki naj bi usmerjevalno posegale na področje informacijske tehnologije, organizacije in razvoja.

Časopis Informatica se izdaja s finančno podporo Slovenskega društva Informatika, ker z naročnino in oglasi pridobljena sredstva le delno pokrivajo celotne izdatke. Zaradi tega naprošamo naše gospodarske organizacije, zlasti tiste, ki se preusmerjajo v proizvodnjo računalniških sistemov in sestavnih delov, da na samem začetku izdajanja časopisa podprejo našo oglaševalno akcijo. Naprošamo pa tudi naše bralce, da pomagajo razširjati časopis v svoje delovne organizacije in na svoje sodelavce, ker bomo le tako lahko dosegli večjo naklado in s tem gospodarno poslovanje časopisa Informatica.

Članki, poročila in drugi prispevki so lahko napisani v slovenskem, srbsko-hrvaškem, makedonskem, angleškem, ruskem, nemškem ali francoskem jeziku. Uredništvo si bo še nadalje prizadevalo dobiti prispevke iz vseh jugoslovanskih industrijskih in raziskovalnih središč ter tudi iz inozemstva.

Na koncu pozivam v imenu uredništva tudi študente in mlade strokovnjake širom po Jugoslaviji, da pošiljajo v objavo svoje članke, poročila, mnenja, kritične opombe, sugestije in predloge o tem, kako bi časopis približali bralcem, kakšna naj bi bila objavljena problematika in kako bi bilo mogoče izboljšati našo strokovno dejavnost, medsebojne odnose in skupne akcije.

a few notes about informatica

a.p.železnikar

Informatica is intended to become a professional journal for problems of computer technology and informatics. Its present orientation stems from our conditions, needs and interests, and is reflected in the following fields:

microcomputers, programming, operating systems, hardware, terminology, information systems, process informatics, robotics, artificial intelligence, computing in the humanities and social sciences, education, literature and meetings, student problems, publishers and other news, etc.

Our principal goal is to make Informatica the publication of engineers, technicians and hobbyists, as well as other professionals in the field of computer technology whose main interes is in small computer systems that can be researched, developed, and produced in Yugoslavia. Such objectives necessarily open up the entire computer and information complex including broader questions in the fields of information systems, human resources, and computer use.

The professional level of the journal is diverse. Paper for broad as well as selected audiences will be published. The emphasis is on contributions of medium level describing practical results and products. For example: methodological descriptions of programs (especially systems programs as well as application programs of general interest), microcomputer hardware and software, computer systems, applications in robotics, etc. are most welcome in this journal. Surveys of new technologies and news about production of computers and components at home and abroad are particularly desirable.

The two field of computing, hardware and software, are treated equally: computer configurations, computer architectures, processors, new hardware components are just as interesting as descriptions of programs (monitors, games, software support). Particularly interesting short programs (especially for microcomputers) will be published in full.

Survey papers on technology and applicatons will be interesting above all when they are related to the context of Yugoslav possibilities. Also, we would like to publish polemical articles about professional education, production organization, organization of information systems, and mutual relations. In this sense the journal is open to all public workers, companies, and institutions whose contributions might provide further orientation in the fields of information technology, its organization, and its development.

The journal Informatica is published with the financial support of Informatika, the Slovene Society for Computing and Informatics, because the subscriptions and advertizing do not cover all expences. Therefore, further advertizing, especially by manufacturers of computer systems and components, is urgently needed. Furthermore, our readers are kindly requested to help promote Informatica among their colleagues. Getting more subscribers is the surest way to sound financial standing of the journal.

Articles, reports, and other contributions may be written in Slovene, Serbo-Croatian, Macedonian, English, Russian, German, or French language. The editorial board will continue with its efforts to solicit contributions from all Yugoslav industrial and research centers as well as from abroad.

At the end, my editorial colleagues and myself invite students and young scientists from all over Yugoslavia to send their articles, reports, opinions, critical remarks, comments, and suggestions on how the journal could be brought closer to readers, what its contents should be, how to improve our professional activities, as well as on any other issues of our common professional interest.

navodilo za pripravo članka

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

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

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

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

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

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

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

Naslovi poglavij:
naslove ločuje od ostalega teksta dvojni presledek.

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

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

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

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

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

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

Časopis mi pošiljajte na naslov ☐ stanovanja ☐ delovne organizacije.

Priimek.....

Ime.....

Naslov stanovanja

Ulica.....

Poštna številka_____ Kraj.....

Naslov delovne organizacije

Delovna organizacija.....

.....

Ulica.....

Poštna številka_____ Kraj.....

Datum..... Podpis:

.....

instructions for preparation of a manuscript

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

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

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

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

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

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

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

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

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

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

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

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

računalniški center univerze v ljubljani

rado faleskini

UDK 681.3.008:378(497.12 Ljubljana)

RCU LJUBLJANA

Avtor podaja v sestavku nekaj misli o strukturi ciljev in strukturi funkcij kot elementih za sistemski razvoj Računalniškega centra Univerze v Ljubljani.

UNIVERSITY COMPUTER CENTRE, UNIVERSITY OF LJUBLJANA, FUNCTIONS AND GOALS - In the paper the structures of goals and functions for further developing and functioning of UCC as organisational system are given.

1. Uvod

Vrhovni cilj ustanoviteljev Računalniškega centra univerze, zaradi katerega je bilo uvedeno pedagoško in raziskovalno delo za uporabo računalnikov, zaradi katerega so bila sistemizirana nova delovna mesta po fakultetah in zaradi katerega je bil ustanovljen tudi Računalniški center univerze, je razvijanje računalniške kulture v naši družbi. Za doseganje tega vrhovnega cilja je potrebno opredeliti sistem nalog oziroma funkcij in manj kompleksnih ciljev, od katerih se jih nekaj realizira v organizaciji Računalniški center univerze, nekaj pa v visokošolskih temeljnih in delovnih organizacijah, v raziskovalnih temeljnih in delovnih organizacijah kakor tudi v samoupravnih interesnih skupnostih.

2. Dosedanji cilji in naloge

Dosedanji cilji in naloge Računalniškega centra Univerze v Ljubljani so bili le zelo splošno opredeljeni v ustanovitvenih aktih v sklepih samoupravnih organov ustanoviteljev to je Univerze v Ljubljani in Instituta "Jožef Stefan" ter v sklepih Raziskovalne skupnosti Slovenije in Izobraževalne skupnosti Slovenije. Kratkoročne cilje in naloge sta določala tudi poslovni odbor RCU in Strokovni svet za računalništvo Univerze v Ljubljani. Doslej opredeljene cilje in naloge bi lahko razdelili v tri skupine - servisni cilji in naloge, koordinacijski cilji in naloge ter raziskovalni cilji in naloge.

Med servisne cilje in naloge spadajo:

- Priprava predlogov za financiranje in investiranje računalniške izobraževalne in raziskovalne dejavnosti za RIS in RSS.
- Usmerjanje finančnih sredstev za računalniško mrežo in informacijski sistem oz. informacijske sisteme na Univerzi:
  - usmerjanje sredstev za materialne stroške rač. centrov po fakultetah
  - usmerjanje sredstev za odplačevanje anuitet za računalniško in softwarsko opremo in zbiranje amortizacije
  - usmerjanje sredstev za stroške centralnega sistema.
- Izdelava investicijsko tehnične dokumentacije in nabavljanje opreme:
  - nabavljanje opreme, izdelava investicijsko-tehnične dokumentacije
  - priprava različnih izjav, pridobivanje različnih soglasij, ki jih potrebujejo uvozniki in carina
  - izdelava predlogov za finančne konstrukcije, kjer so vključeni tudi stroški centralnega sistema in materialni stroški, ki nastopajo z novim hardwareom in softwareom.
- Generiranje šifer in razporejanje centralnih kapacitet po šolah.
- Vzdrževanje programskih paketov.
- Konzultantsko delo za področje:
  - operacijskih sistemov
  - programskih paketov in zbirk programov
  - programiranje
  - načrtovanje nove opreme
  - dokumentalistike.
- Vzdrževanje terminalne opreme in mini računalnikov.
- Poganjanje centralnega sistema .
- Softwarske usluge za zainteresirane ustanovitelje in univerzitetne delovne organizacije in delovne skupnosti:
  - centralizirana obdelava podatkov o študentih in učiteljih (za RCU)
  - administracija univerze in fakultet
  - obdelave podatkov o izpitih, testih itd. za zainteresirane ustanovitelje
  - pomoč pri izdelavi potr. programov.
- Organizacija izobraževanja:
  - dopolnilno izobraževanje učiteljev, zlasti učiteljev računalništva in informatike
  - dopolnilno izobraževanje, za katero pooblastijo RCU

fakultete.

- Nabavljanje in izmenjava literature in drugih informacij med šolami.
- itd.

Med koordinacijske cilje in naloge spadajo:

- Koordinacija pri izgradnji hardwarskega sistema visokega šolstva in načrtovanje sistema kot celote.
- Koordinacija pri razvoju in nakupovanju softwarea.
- Koordinacija pri uporabi kapacitet in računalniške opreme, in koordinacija pri uporabi specializiranih računalniških hardwarskih in softwarskih kapacitet in njihovo interdisciplinarno načrtovanje.
- Koordinacija pri sistematizaciji delovnih mest.
- Koordinacija pri razvoju CAL, CAI, CAD.
- Koordinacija prizadevanj fakultet in TOZD v zvezi s s standardizacijsko dejavnostjo:
  - standardizacija softwarske dokumentacije
  - izdelava priporočil za pripravo projektov, ki se bodo obravnavali z računalnikom.
- Koordinacija pri uporabi konzultantov po fakultetah.
- Koordinacija pri sodelovanju pri nacionalnih in mednarodnih projektih.
- Koordinacija pedagoškega dela med šolami (skripta, knjige, predmetniki).
- Koordinacija raziskovalnega dela med šolami in fakultetami, za katero so zainteresirani ustanovitelji.
- Povezovanje preko republiških meja.
- Sodelovanje v glasilih za področje informatike in računalništva.
- Izdajanje BILTENA za računalništvo.
- Sodelovanje z delavskimi univerzami.
- Sodelovanje s tiskom in RTV.
- Itd.

Med raziskovalne cilje in naloge spadajo:

- Raziskave o potrebni vsebini izobraževanja in vzgoje za področje računalništva in informatike v srednjem in visokem šolstvu ter v drugih oblikah izobraževanja in vzgoje (DU, tisk).
- Raziskave o obsegu izobraževanja za področje računalništva in informatike v vseh okoljih in o potrebah po različnih kadrih.
- Raziskave s področja računalništva in informatike, za katere so zainteresirani ustanovitelji.
- Itd.

Med cilji in nalogami so tudi mnogi taki, ki jih Računalniški center univerze ni uspel uresničiti in se bo potrebno o njih pri določevanju novega sistema ciljev in funkcij ponovno potrebno pogovarjati.

3. Reorganizacija RCU

Reorganizacija Univerze in nova zakonodaja med katero so zlasti pomembni zakoni o združenem delu, o planiranju, o visokem šolstvu itd. zahteva tudi reorganizacijo ter s tem zvezano nove definicije funkcij in nalog Računalniškega centra univerze. Nova organizacijska oblika bo morala kot temeljne subjekte, ki združujejo svoje interese za realizacijo določenih skupnih ciljev v skladu z zakonodajo vzeti visokošolske temeljne in delovne organizacije. Upoštevati pa bo morala tudi dosedanje subjekte to je ustanovitelje in samoupravne interesne skupnosti za področji raziskovalne in izobraževalne dejavnosti, kakor tudi cilje in plane gospodarstva in družbe. V zvezi s tem bodo morale tudi vse visokošolske temeljne in delovne organizacije opredeliti svoje cilje in interese in tisti njihov del, ki ga bodo realizirale v skupni organizacijski obliki, ki jo bo predstavljal Računalniški center univerze. Podobno velja za univerzo, raziskovalne organizacije in za izebraževalno in raziskovalno skupnost Slovenije. Gre za to, da pridemo do sistema ciljev in funkcij za realizacijo vrhovnega cilja, ki ga predstavlja razvijanje računalniške kulture v naši družbi in za podsistem ciljev in funkcij, ki naj se realizirajo s pomočjo Računalniškega centra univerze, kakor tudi za podsisteme, ki se realizirajo na visokošolskih temeljnih organizacijah v raziskovalnih organizacijah, na univerzi in v samoupravnih interesnih skupnostih.

Pri tem je potrebno upoštevati nekatera dognanja teorije organizacijskih sistemov. Podsistemi ciljev in podsistemi funkcij morajo v vsebini ciljev sistema najti del svojih ciljev. Funkcionalni podsistemi se lahko vključijo v sistem samo, če se bodo njihovi cilji bolje in efektivneje realizirali v sistemu, kot bi se realizirali izven sistema. Brez tega skupnega cilja bi sistem deloval zelo nestabilno, ob stalno prisotni nevarnosti, da razpade. Možnost za del lastnih ciljev podsistemov pa je pogoj za njihovo individualnost in za njihovo pripravljenost, da so vključeni v sistem.

Znotraj danih zakonskih možnosti je potrebno narediti ustrezno konstitucijo Računalniškega centra univerze. Na principih konstitucije počiva struktura organizacijskega sistema in njegove funkcionalne lastnosti. Računalniški center univerze kot tudi vsak drug organizacijski sistem mora biti kompletiran z vsemi potrebnimi podsistemi. Izostanek katerega-koli od konstitucionalno potrebnih elementov naredi sistem defekten, kar ruši oziroma zmanjšuje njegove funkcionalne lastnosti. Konstitucija sistema mora regulirati status vsakega njegovega elementa bodisi strukturalno, bodisi funkcionalno v odvisnosti od ostalih elementov. Nereguliran status posameznih elementov izključuje te elemente iz sistema in s tem razbija sistem. Adekvaten status zahteva tako reguliranj statusa vseh podsistemov in njihovih medsebojnih odnosov, da so čimbolj vsklajeni s cilji sistema.

Treba je najti zadovoljiv kompromis med razvojnimi optimumi podsistemov in sistema kot celote in kompromis med optimumi ciljev, funkcij, upravljanja in kvalitete in glede na ta kompromis optimalno strukturo.
Smisel optimalne funkcionalne strukture je doseganje optimalnega odnosa med elementi rezultatov in elementi vlaganj ob maksimalni funkcionalnosti vloženih elementov. Če pride pri konstituiranju sistema do nepotrebnega viška enih elementov glede na druge, postanejo ti odvečni elementi objektivni faktorji, katerih negativni vpliv na funkcioniranje strukture je trajen.

Tudi pri razmišljanju o ciljih in funkcijah Računalniškega centra univerze nam gre podobno kot pri drugih organizacijskih sistemih za optimalno kvaliteto sistema v dinamiki in za ravnotežje med podsistemi, ko sistem deluje in ne za nek abstrakten model za katerega je dovolj, da ga opredelimo na papirju.

4. Zaključek

Problematika reorganizacije Računalniškega centra univerze je zelo kompleksna in nanjo nikakor ne bi smeli gledati

zgolj kot na formalno pravno problematiko. Za realizacijo družbenega cilja je potrebno upoštevati kompleksnost problematike, sedanje stanje in notranja protislovja v njem ter dolgoročne efekte, ki jih prinaša računalništvo. Iluzorno bi bilo pričakovati, da bo reorganizirani Računalniški center univerze organizacijsko brezhiben. Prizadevati pa si moramo, da bo čimboljši. Poslovne funkcije bodo morale biti načrtane tako, da bo na večino možen permanenten vpliv ustanoviteljev. Zlasti je to pomembno za plansko, razvojno, finančno in samoupravno funkcijo, oziroma lahko bi rekli samoupravno funkcijo v širšem smislu. Funkcija samoupravljanja ima med drugim nalogo, da predvideva obnašanje sistema in da usmerja sistem k ciljem. Kvaliteta samoupravljanja zavisi od stopnje organiziranosti oziroma entropije upravljalnega podsistema. Upravljalni podsistem ima minimalno entropijo in je zato najbolj efektiven, kadar ima adekvatne informacije, kadar daje adekvatne upravljalne impulze in ko je njihova kasnitev sinhronizirana.

Neadekvatne odločitve lahko zelo zmanjšajo efekte sistema. Zamude pri informacijah, pri sprejemanju odločitev in pri realizaciji ravno tako poslabšajo efekte organizacijskega sistema.

Neorganiziranost povzroča lahko tudi samo informacijski podsistem ali podsistem sprejemanja odločitev. V teh primerih govorimo o dezorganizaciji organizacijskega sistema.

Drugi primer možne dezorganizacije je identifikacija ciljev organizacijskega sistema s cilji nosilcev samoupravnih funkcij. V tem primeru lahko pride tudi samo do faktične entropije, ne pa tudi formalne. Sistem v tem primeru efektivno deluje pri realizaciji ciljev nosilcev samoupravnih funkcij, pri realizaciji formalno postavljenih ciljev pa je neefektiven. Zato je pomembno, da so formalno postavljeni cilji razdelani, zapisani in splošno samoupravno sprejeti.

L i t e r a t u r a

Kukoleča, Teorija organizacionih sistema;
Fakultet organizacionih nauka,
Beograd, 1972.

Mulej M., Teorija sistemov-skripta;
Visoka ekonomsko-komercialna šola,
Maribor, 1973.

manipulisanje hijerarhijskom bazom podataka

suad alagić
radovan jović
dženan ridjanović

UDK 681.3.06

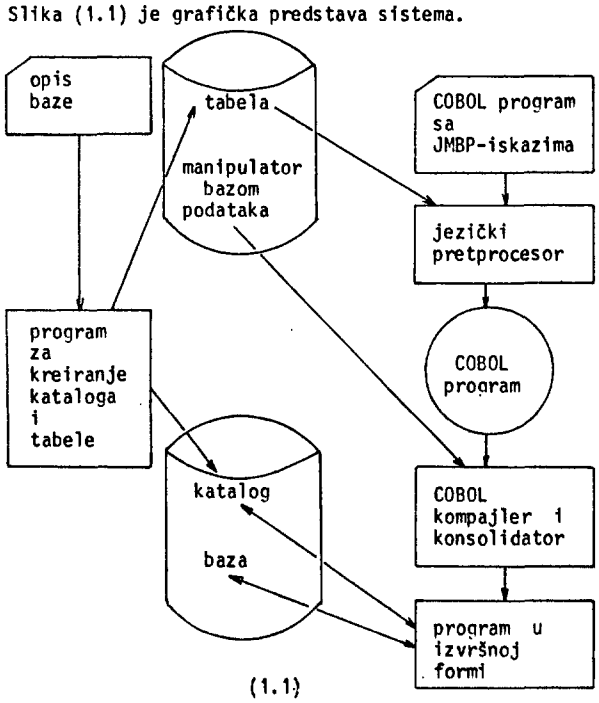
Elektrotehnički fakultet,
71000 Sarajevo-Lukavica

Opisani su,sa korisnikove tačke gledišta,najvažniji aspekti jednog hijerarhijskog sistema za upravljanje bazom podataka.Ovaj sistem je realizovan na računaru ICL 1902A,sa namjerom da se pokaže kako se postojeći softver može proširiti razumnim naporima,tako da se dobije sistem za upravljanje bazom podataka koji je dovoljno moćan,jednostavan za korištenje i ekonomičan u raspolaganju memorijom.U ovom radu opisani su: jezik za manipulisanje bazom podataka i primjeri njegovog korištenja.Drugi aspekti sistema,posebno njegova realizacija,dati su u drugim radovima.

HIERARCHICAL DATABASE MANIPULATIONS: The most important aspects of a hierarchical database management system are described from the user viewpoint.The system was implemented on the ICL 1902A machine,with the intent to show how existing software facilities can be extended by a reasonable effort,giving a database management system which is powerful enough,simple to use and economical in memory utilisation.In this paper the data manipulation language and examples of its use are presented.Other aspects of the system,in particular,its implementation,are reported in other papers.

OSNOVNI OPIS SISTEMA

Slika (1.1) je grafička predstava sistema.



(1.1)

Sistem u osnovi funkcioniše na slijedeći način:
COBOL program sa iskazima jezika za manipulisanje bazom podataka (JMBP-iskazima) prvo prolazi kroz jezički pretprocesor koji prevodi te iskaze u pozive potprograma.Ulazne tačke manipulatora bazom podataka upravo odgovaraju tim potprogramima.Onda COBOL kompajler daje, poslije kompilacije i konsolidacije,program u mašinskom kodu u kojem se u toku izvršavanja pozivaju potprogrami koji odgovaraju JMBP-iskazima.Ti potprogrami komuniciraju sa katalogom (interni opis baze podataka) i bazom podataka i oni su u stvari posrednici izmedju aplikacionog programera i baze podataka.

Najvažniji dio sistema je manipulator bazom podataka. Potprogrami koji sačinjavaju manipulator operišu na posebnoj strukturi koja je veoma pogodna za realizaciju skupova podataka koji su povezani hijerarhijskim vezama.Ta struktura je tzv. B-drvo (Bayer i McCreight (1972)),koje ima takve osobine da omogućuje pretraživanje,umetanje i izbacivanje slogova logaritamskom brzinom,a garantuje bar pedeset postotnu iskorištenost memorije,koja je obično mnogo veća.Važno je i to da je B-drvo vrlo pogodno i za traženje sloga na osnovu njegovog ključa i za sekvencijalnu obradu slogova u rastućem ili opadajućem redoslijedu ključeva. Upravo upotrebljeni način na koji je B-drvo korišteno u realizaciji ovog sistema i algoritmi za realizaciju pojedinih JMBP-iskaza detaljno su prikazani u /5/.

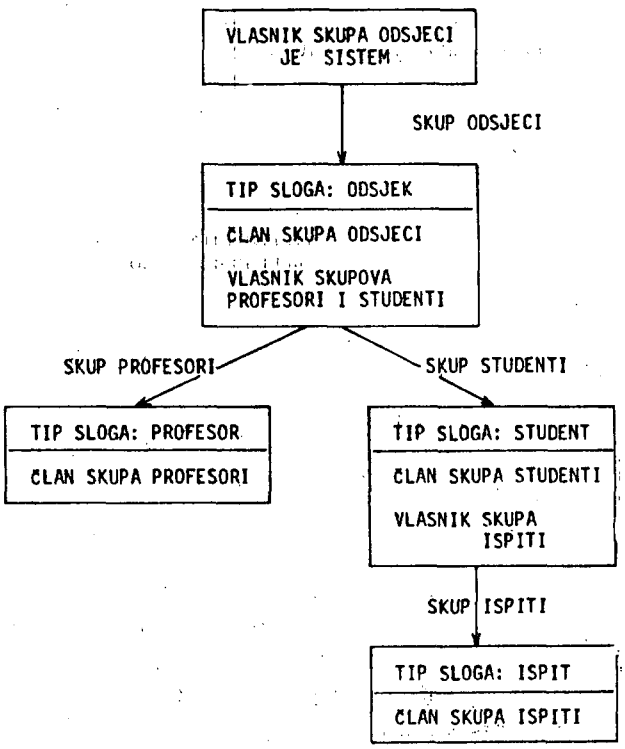
SKUPOVI I HIJERARHIJSKE STRUKTURE

Skup,odredjenog imena,je par tipova slogova.Ovi tipovi se nazivaju tip sloga vlasnik i tip sloga član.
Tip sloga može biti vlasnik jednog ili više skupova, član samo jednog skupa i oboje u isto vrijeme,ali ne može biti vlasnik i član u istom skupu.
Primjerak odredjenog skupa se sastoji od sloga vlasnika i od n ($\geqslant 0$) slogova članova.Za n=0 kažemo da se radi o praznom primjerku skupa.Važno je napomenuti da je postojanje sloga vlasnika uslov za postojanje primjerka skupa.
Slogovi članovi primjerka skupa linearno su uredjeni na bazi vrijednosti odredjenog polja unutar sloga (ključa).
Za svaki primjerak skupa potrebno je poštovati slijedeća pravila:

1. Za dati slog vlasnik primjerka skupa,moguće je doći do slogova članova toga primjerka skupa.
2. Za dati slog član primjerka skupa,moguće je doći do sloga vlasnika toga primjerka skupa.
3. Za dati slog član primjerka skupa,moguće je doći do ostalih slogova članova toga primjerka skupa.
4. Slog član može biti član samo jednog primjerka skupa,odnosno slog član ne može u isto vrijeme imati dva ili više slogova vlasnika.

Osobina hijerarhijskih struktura je ta da je jedan slog vlasnik od 0 do n ($\geq$0) slogova drugog tipa,od kojih je svaki vlasnik od 0 do n slogova trećeg tipa,itd. ali ni jedan slog ne može biti vlasnik sloga (direktno ili indirektno) koji je njegov vlasnik.

Na osnovu ovih objašnjenja grafički ćemo predstaviti hijerarhijsku strukturu baze podataka fakulteta,koja nam je poslužila za testiranje sistema:



(1.2)

Vlasnik skupa ODSJECI je sam sistem,odnosno u sistemu mora postojati indikacija logičkog početka baze podataka.Svaki slog skupa ODSJECI je vlasnik dva primjerka skupa: primjerka skupa PROFESORI (moguće praznog) i primjerka skupa STUDENTI (moguće praznog),tj. svaki slog tipa ODSJEK odredjuje kolekciju slogova tipa PROFESOR (moguće praznu) i kolekciju slogova tipa STUDENT (moguće praznu).Takodje,svaki slog tipa STUDENT odredjuje kolekciju slogova tipa ISPIT (moguće praznu).

PRISTUPANJE BAZI I SAOPSTAVANJE GRESAKA

Naglasimo dvije činjenice koje su bitne za funkcionisanje sistema:
U manipulatoru postoji lokalna memorijska zona koja se zove radni pokazivač.Ovaj radni pokazivač jednoznačno odredjuje jedan slog iz baze,i to onaj sa kojim se trenutno radi.Istovremeno radni pokazivač odredjuje i primjerak skupa kome taj slog pripada.
Efekat skoro svih JMBP-iskaza definisan je u odnosu na trenutno stanje radnog pokazivača,a neki od ovih iskaza ga i ažuriraju.

Kada korisnik želi pristupiti bazi podataka,on to mora najaviti iskazom:[1]

```
#INVOKE.
```

Ovo je obavezno prvi JMBP-iskaz korisnikovog programa, a njegova funkcija je slična funkciji COBOL iskaza OPEN.Napomenimo da su članovi svakog primjerka skupa organizovani u vidu B-drveta,pa ako baza nije prazna, INVOKE-iskaz postavlja radni pokazivač na prvi slog člana korijen stranice primjerka skupa čiji je vlasnik sistem.

Iskazom:

```
#END.
```

odjavljuje se rad sa bazom,i on je obavezno zadnji JMBP -iskaz korisnikovog programa.Funkcija mu je slična funkciji COBOL iskaza CLOSE.U slučaju da programer ne upotrijebi ovaj iskaz,pretprocesor će signalizirati grešku.

Prilikom manipulisanja bazom podataka postoje situacije kada je potrebno uspostaviti komunikaciju izmedju korisnika i sistema.Iskaz:

```
#ERROR FIELD IS <identifikator>.
```

omogućava programeru da dobije obavještenje o nekoj izuzetnoj situaciji,i to u polje radne zone navedeno ovim iskazom.Ovo polje se mora sastojati od četiri alfanumerička karaktera.Izbor karaktera je proizvoljan. Na primjer:

```
WORKING-STORAGE SECTION.
77  PORUKA PIC X(4).
.
.
PROCEDURE DIVISION.
POCETAK.
.
.
    #INVOKE.
    #ERROR FIELD IS PORUKA.
    .
    .
    #END.
```

Nakon pojave izuzetne situacije zabranjena je upotreba nekih JMBP-iskaza,a ako se oni ipak upotrijebe program će biti zaustavljen i poruka o grešci će se pojaviti na konzoli (kažnjavanje pogrešnih postupaka programera zaustavljanjem programa i štampanjem koda greške na konzoli vrši se kod grešaka koje bi mogle dovesti do poremećaja strukture baze podataka ili do gubljenja kontrole nad programom).Prema tome,informacija o izuzetnoj situaciji omogućava programeru da,poduzimanjem dozvoljenih koraka,a dozvoljeni koraci su oni koji imaju smisla u odredjenoj situaciji,izbjegne neželjeni prekid programa.Preporučljivo je da se ovaj iskaz piše u svakom programu za manipulisanje bazom podataka,i to odmah iza INVOKE-iskaza.

PRETRAZIVANJE BAZE

Jezik za manipulisanje bazom podataka ima šest iskaza za pretraživanje,od kojih se pet odnosi na interno uredjenje skupova,a šesti na veze izmedju skupova.Svi ovi iskazi postavljaju radni pokazivač na pronadjeni slog.

Slogovi članovi primjeraka skupova uredjeni su po rastućem nizu ključeva.Prilikom sekvencijalne obrade često se polazi od prvog ili posljednjeg člana niza.Iskazom:

```
#FIND FIRST.
```

se na najbrži način pronalazi prvi član niza,odnosno slog član primjerka skupa sa najmanjim ključem,a iskazom:

```
#FIND LAST.
```

slog član primjerka skupa sa najvećim ključem.

U sekvencijalnoj obradi efikasan je i koristan iskaz:

```
#FIND NEXT.
```

koji pronalazi slijedeći slog u nizu.U stvari,on pronalazi slijedeći slog u odnosu na slog odredjen radnim pokazivačem.Prilikom upotrebe ovog iskaza može se pojaviti jedna izuzetna situacija.To je slučaj kada se traži slijedeći slog u odnosu na slog sa najvećim ključem

1) Svi JMBP-iskazi počinju sa # i pišu se u odvojenom redu.Ostala sintaksička pravila data su u /2/.

u primjerku skupa.Takav slog naravno ne postoji i to se saopštava programeru upisivanjem koda "NESL" (nema slijedećeg) u polje radne zone zadato ERROR-iskazom.U ovakvoj situaciji nije dozvoljena upotreba slijedećih JMBP-iskaza:
FIND NEXT,FIND PRIOR,FIND SET <ime-skupa> ,
SAVE <ime-pokazivača> ,GET <ime-sloga> ,
PUT <ime-sloga> ,INSERT <ime-sloga> INTO SET <ime-skupa> ,DELETE ALL BASE,DELETE RECORD,
DELETE SET <ime-skupa> ,DETACH <ime-strukture> ,
DETACH <ime-strukture> USING SET <ime-skupa> ,
ATTACH <ime-strukture> USING SET <ime-skupa> .
Ovi iskazi nisu dozvoljeni jer rade sa slogom na koga radni pokazivač trenutno pokazuje,a u navedenom slučaju radni pokazivač ne pokazuje ni na jedan slog.Pošto ćemo se na ovu grupu JMBP-iskaza pozivati nekoliko puta,nazovimo je grupa JMBP-iskaza koji rade sa trenutnim slogom.Ako se ovi iskazi ipak upotrijebe,program će se zaustaviti a na konzoli će se pojaviti poruka o grešci.
To se neće desiti ukoliko se upotrijebe slijedeći JMBP-iskazi (i na njih ćemo se nekoliko puta pozivati pa ih nazovimo grupa JMBP-iskaza koji rade na trenutnom primjerku skupa):
FIND FIRST,FIND LAST,FIND <identifikator> ,
FIND <literal> ,ATTACH <ime-strukture> ,
UNSAVE <ime-pokazivača> .
Ovi iskazi imaju smisla dokle god ima članova u primjerku skupa.Zbog svega ovog,preporučljivo je iskaz FIND NEXT pisati u sprezi sa if rečenicom:

```
#FIND NEXT.
IF PORUKA = "NESL" THEN imperativni iskaz.
```

Iskaz:

```
#FIND PRIOR.
```

ima sličnu funkciju kao FIND NEXT,s tom razlikom što se traži prethodni slog u odnosu na slog odredjen radnim pokazivačem.Izuzetna situacija se javlja kada se traži prethodni slog,na osnovu uredjenja članova primjerka skupa,u odnosu na slog član sa najmanjim ključem u primjerku skupa.Kod ovakve situacije,koga sistem saopštava programeru u spomenutu lokaciju radne zone,je "NEPR" (nema prethodnog).U tom slučaju je zabranjena upotreba grupe JMBP-iskaza koji rade sa trenutnim slogom,a dozvoljena je upotreba grupe JMBP-iskaza koji rade na trenutnom primjerku skupa.Ovaj FIND-iskaz je,takodje, preporučljivo pisati u sprezi sa if rečenicom.

Slijedeći iskaz pretraživanja

$$\#\text{FIND}\left\{\begin{matrix}<\text{identifikator}>\\<\text{literal}>\end{matrix}\right\}$$

omogućava da se,na osnovu zadanog ključa,pronadje slog član u primjerku skupa,odredjenom radnim pokazivačem. Ključ se zadaje kao literal ili se zadaje lokacija u radnoj zoni u kojoj se nalazi ključ.U slučaju da se slog ne pronadje upisuje se kod "SLNN" (slog nije nadjen) u polje odredjeno ERROR-iskazom (nazovimo ga u ovom radu PORUKA).U tom slučaju nije dozvoljena upotreba grupe JMBP-iskaza koji rade sa trenutnim slogom, a može se upotrijebiti grupa JMBP-iskaza koji rade na trenutnom primjerku skupa.Zbog toga preporučujemo da se nakon ovog iskaza koristi if rečenica,kojom bi se provjerilo da li je slog nadjen.

Iskaz:

```
#FIND SET <ime-skupa> .
```

u slogu vlasniku,odredjenom radnim pokazivačem,traži pokazivač na članove primjerka skupa,zadatog identifikatorom ime-skupa.Ako ga nadje,postavlja radni pokazivač na korijen stranicu (i to na prvi slog član te stranice) članova primjerka skupa.Rezultat ovog pretraživanja može biti negativan iz dva razloga.Prvi je, ako programer ne vodi računa o hijerarhijskom uredjenju skupova,tj. ako u iskazu nije naveo odgovarajuće ime skupa.U tom slučaju program bi se zaustavio,a na konzoli bi se pojavila poruka o grešci.Drugi razlog je,ako je primjerak skupa,označen identifikatorom ime-skupa, prazan.Tada će se u polje radne zone PORUKA smjestiti kod "PSJP" (primjerak skupa je prazan).U ovoj situaciji jedino je dozvoljena upotreba UNSAVE-iskaza.Ako se upotrijebi bilo koji drugi JMBP-iskaz,program će se zaustaviti,a na konzoli će se pojaviti poruka o grešci. Zbog toga je,u situacijama kada programer nije siguran da li je traženi primjerak skupa prazan (dilema je da li nema ili ima slogova članova u tom primjerku skupa), preporučljivo ovaj iskaz pisati u sekvenci:

```
#INVOKE.
#ERROR FIELD IS PORUKA.
.
.
#SAVE POKAZIVAC.
#FIND SET STUDENTI.
IF PORUKA = "PSJP" THEN
#UNSAVE POKAZIVAC
imperativni iskaz.
.
.
#END.
```

Iskaz:

```
#SAVE <ime-pokazivača> .
```

omogućava programeru da,u toku izvršavanja svog programa,zapamti neke tačke u bazi,na koje se kasnije može veoma brzo vratiti upotrebom iskaza:

```
#UNSAVE <ime-pokazivača> .
```

U stvari,SAVE-iskaz sačuva trenutnu vrijednost radnog pokazivača,a UNSAVE-iskaz postavi radni pokazivač na neku,prethodno sačuvanu,vrijednost,koja je poznata sistemu pod imenom navedenim u ovim iskazima.U jednom programu može se upotrijebiti maksimalno deset različitih imena pokazivača (za veći broj potrebno je i više memorijskog prostora).Pod jednim imenom može se sačuvati proizvoljno mnogo tačaka baze,s tim da se nova tačka pamti preko stare,pa se u tom slučaju UNSAVE-iskazom može pristupiti samo posljednjoj zapamćenoj tački.Ako se slog izbaci sa nekog mjesta iz baze,njegov pokazivač prethodno sačuvan SAVE-iskazom,biće postavljen na nulu. Ukoliko programer pokuša UNSAVE-iskazom pristupiti takvom pokazivaču,program će se zaustaviti i na konzoli će se pojaviti poruka o grešci.Iskazima SAVE i UNSAVE se može realizovati traženje sloga vlasnika članova nekog primjerka skupa.Ovo se radi tako da se prije prelaska na niži hijerarhijski nivo zapamti pokazivač na slog vlasnika,a kasnije,kada je za vrijeme rada sa članovima primjerka skupa potrebno pronaći vlasnika,jednostavno se UNSAVE-iskazom vrati prethodno zapamćeni pokazivač.

AZURIRANJE SLOGOVA I STAMPANJE IZVJESTAJA

Ako programer želi dobiti u svojoj radnoj zoni sadržaj nekog sloga iz baze,on prvo mora pronaći taj slog,a onda ažurajućim iskazima postaviti radni pokazivač na taj slog),a zatim upotrijebiti iskaz:

```
#GET <ime-sloga> .
```

Ovaj iskaz prenosi slog odredjen radnim pokazivačem u lokaciju radne zone zadatu identifikatorom ime-sloga. Ažurirani slog se vraća na svoje mjesto u bazi iskazom:

```
#PUT <ime-sloga> .
```

Pri tome,ovaj iskaz provjerava da li je ključ sloga u programerovoj radnoj zoni identičan ključu sloga u bazi.Ako to nije slučaj,zaustavlja se izvršavanje programa i na konzoli se štampa poruka o grešci.
Podatke studenta SRNA AZRE korisnik može ažurirati na slijedeći način:

```
WORKING-STORAGE SECTION.
77  PORUKA  PIC X(4).
01  STUDENT.
    02  PODACI-S-A  PIC X(11).
    02  KLJUC-S     PIC X(29).
    02  PODACI-S-B  PIC X(120).
```

```
.
.
#INVOKE.
#ERROR FIELD IS PORUKA.
#FIND "INFORMATIKA      ".
#FIND SET STUDENTI.
MOVE "SRNA AZRA" TO KLJUC-S.
#FIND KLJUC-S.
#GET STUDENT.
ažuriranje podataka.
#PUT STUDENT.
.
.
#END.
```

Ako programer želi imati u svojoj radnoj zoni istovremeno dva sloga iz baze,onda on mora rezervisati dvije odgovarajuće lokacije,i ako vrši ažuriranje tih slogova, iskazima SAVE i UNSAVE mora obezbjediti njihovo vraćanje u bazu:

```
WORKING-STORAGE SECTION.
77  PORUKA  PIC X(4).
01  ODSJEK.
    02  PODACI-O-A  PIC X(4).
    02  KLJUC-O     PIC X(16).
    02  PODACI-O-B  PIC X(108).
01  STUDENT.
    02  PODACI-S-A  PIC X(11).
    02  KLJUC-S     PIC X(29).
    02  PODACI-S-B  PIC X(120).
.
.
PROCEDURE DIVISION.
.
.
    #INVOKE.
    #ERROR FIELD IS PORUKA.
    #FIND "INFORMATIKA      ".
    #SAVE INFORMATIKA.
    #GET ODSJEK.
    #FIND SET STUDENTI.
    MOVE "MANDIĆ PETAR" TO KLJUC-S.
    #FIND KLJUC-S.
    #GET STUDENT.
    ažuriranje učitanih slogova.
    #PUT STUDENT.
    #UNSAVE INFORMATIKA.
    #PUT ODSJEK.
    .
    .
    #END.
```

Očigledno je da je i za više slogova postupak analogan. Ako bi iz baze podataka prikazane na slici (1.2) bilo potrebno izlistati imena svih studenata četvrte godine Odsjeka za informatiku,program bi izgledao ovako:

```
DATA DIVISION.
FILE SECTION.
FD  STAMPANJE.
01  SLOG-S.
    02  FILLER  PIC X(11).
    02  IMENA   PIC X(29).
    02  FILLER  PIC X(80).
WORKING-STORAGE SECTION.
77  PORUKA  PIC X(4).
01  STUDENT.
    02  PODACI-A         PIC X(10).
    02  GODINA-STUDIJA   PIC 9.
    02  IME-STUDENTA     PIC X(29).
    02  PODACI-B         PIC X(120).
PROCEDURE DIVISION.
POCETAK.
    OPEN OUTPUT STAMPANJE.
    MOVE ALL SPACES TO SLOG-S.
    #INVOKE.
    #ERROR FIELD IS PORUKA.
    #FIND "INFORMATIKA      ".
    #FIND SET STUDENTI.
    #FIND FIRST.
    PERFORM STAMPANJE-STUDENTA
    UNTIL PORUKA = "NESL".
    #END.
    CLOSE STAMPANJE.
    STOP RUN.
STAMPANJE-STUDENTA.
    #GET STUDENT.
    IF GODINA-STUDIJA = 4 THEN
    MOVE IME-STUDENTA TO IMENA
    WRITE SLOG-S AFTER 2.
    #FIND NEXT.
****
```

Ažuriranje podataka o svim studentima ETF-a moglo bi se izvesti na slijedeći način:

```
.
.
#INVOKE.
#ERROR FIELD IS PORUKA.
#FIND FIRST.
PERFORM AZURIRANJE-STUDENATA
UNTIL PORUKA = "NESL".
#END.
.
.
AZURIRANJE-STUDENATA.
    #SAVE ODSJEK.
    #FIND SET STUDENTI.
    #FIND FIRST.
    PERFORM AZURIRAJ-STUDENTA
    UNTIL PORUKA = "NESL".
    MOVE ALL ZEROS TO PORUKA.
    #UNSAVE ODSJEK.
    #FIND NEXT.
AZURIRAJ-STUDENTA.
    #GET STUDENT.
    ažuriranje podataka o studentu.
    #PUT STUDENT.
    #FIND NEXT.
****
```

UMETANJE SLOGOVA

Umetanje sloga člana u primjerak skupa vrši se iskazom:

#INSERT <ime-sloga> INTO SET <ime-skupa>.

U iskazu je potrebno navesti lokaciju u kojoj se nalazi slog koji se umeće i ime skupa u čiji primjerak treba umetnuti slog.Slog član se umeće u primjerak skupa na kojeg radni pokazivač trenutno pokazuje.Ovo umetanje se vrši na bazi ključa,i to na mjesto koje na osnovu uredjenja članova primjerka skupa pripada spomenutom slogu. Ukoliko utvrdi da slog član sa takvim ključem već postoji u primjerku skupa,sistem ga ne umeće,već štampa njegov ključ na konzoli (bez zaustavljanja programa) i saopštava programeru poruku "DPSL" (dupli slog) u lokaciju odredjenu ERROR-iskazom.Ovo se radi zbog toga,što medju članovima primjerka skupa ne smiju postojati dva sloga sa istim ključem.U slučaju da programer ne poštuje hijerarhijske odnose,program će biti zaustavljen i na konzoli će se pojaviti poruka o grešci.
Članovi primjerka skupa ODSJECI,ETF-a (pogledati sliku (1.2)),mogli bi se kreirati ovako:

```
.
.
#INVOKE.
#ERROR FIELD IS PORUKA.
PERFORM UMETANJE-ODSJEKA
VARYING I FROM 1 BY 1 UNTIL I>5.
#END.
.
.
STOP RUN.
UMETANJE-ODSJEKA.
    READ KARTICE AT END imperativni iskaz.
    MOVE PODACI TO ODSJEK.
    #INSERT ODSJEK INTO SET ODSJECI.
****
```

U jednom programu bi se mogli umetnuti studenti svih odsjeka ETF-a:

```
DATA DIVISION.
FD  KARTICE.
01  PODACI.
```

```
        02  TERMINATOR  PIC X(4).
        02  FILLER      PIC X(76).
    WORKING-STORAGE SECTION.
    77  PORUKA  PIC X(4).
    01  STUDENT.
        02  PODACI-S-A  PIC X(11).
        02  KLJUC-S     PIC X(29).
        02  PODACI-S-B  PIC X(120).
    PROCEDURE DIVISION.
    POCETAK.
        OPEN INPUT KARTICE.
        #INVOKE.
        #ERROR FIELD IS PORUKA.
        #FIND FIRST.
        PERFORM UMETANJE-STUDENATA
        UNTIL PORUKA = "NESL".
        #END.
        CLOSE KARTICE.
        STOP RUN.
    UMETANJE-STUDENATA.
        #SAVE ODSJEK.
        PERFORM UMETNI-STUDENTA
        UNTIL TERMINATOR = "////".
        #UNSAVE ODSJEK.
        #FIND NEXT.
    UMETNI-STUDENTA.
        READ KARTICE AT END imperativni iskaz.
        IF TERMINATOR ≠ "////" THEN
        MOVE PODACI TO STUDENT
        #INSERT STUDENT INTO SET STUDENTI.
****
```

Prilikom umetanja članova primjerka skupa kartice sa *podacima ne moraju biti sortirane po ključu.*čak je i poželjno da ne budu (zbog bolje popunjenosti stranice). Napomenimo da ovaj iskaz postavlja radni pokazivač na umetnuti slog.

IZBACIVANJE SLOGOVA

Da bi se izbacio slog iz baze potrebno je upotrijebiti iskaz:

```
#DELETE RECORD.
```

Ovaj iskaz,u stvari,izbacuje slog na koga radni pokazivač trenutno pokazuje,kao i čitavu podstrukturu za koju je ovaj slog vlasnik.Nakon toga,radni pokazivač se postavlja na slijedeći slog,što omogućava vrlo efikasno brisanje sekvence slogova:

```
        .
        .
        #FIND "INFORMATIKA      ".
        #FIND SET STUDENTI.
        IF PORUKA ≠ "PSJP" THEN
        #FIND FIRST
        PERFORM BRISI-STUDENTA
        VARYING I FROM 1 BY 1 UNTIL I>5.
        .
        .
    BRISI-STUDENTA.
        #DELETE RECORD.
    PARAGRAF.
        .
        .
```

Prilikom upotrebe ovog iskaza mogu se pojaviti tri izuzetne situacije.Prva je,kada se izbriše slog član sa najvećim ključem u primjerku skupa.U tom slučaju se u lokaciju zadanu ERROR-iskazom smješta kod "NESL" i u slijedećem koraku dozvoljena je upotreba jedino grupe JMBP-iskaza koji rade na trenutnom primjerku skupa.Pretpostavlja se da brisanjem sloga primjerak skupa nije postao prazan.Upotreba drugih JMBP-iskaza biće kažnjena zaustavljanjem programa i štampanjem poruke o grešci na konzoli.Zbog toga je preporučljivo koristiti ovaj DELETE-iskaz u sprezi sa <u>if</u> rečenicom:

```
        .
        .
        #DELETE RECORD.
        IF PORUKA = "NESL" THEN imperativni iskaz.
```

Brisanje sekvence slogova počevši od sloga sa ključem "PERIĆ PERO " pa do sloga sa najvećim ključem vrši se ovako:

```
        .
        .
        #FIND "PERIĆ PERO                ".
        PERFORM BRISANJE UNTIL PORUKA = "NESL".
        .
        .
    BRISANJE.
        #DELETE RECORD.
    PARAGRAF.
        .
        .
```

Primijetimo da se,ako je potrebno brisati prvi,zadnji ili slog sa zadatim ključem,DELETE RECORD koristi u sprezi sa odnovarajućim FIND-iskazom.
Druga izuzetna situacija je kada se izbaci i posljednji slog član primjerka skupa.Tada će se programeru saopštiti poruka "IZSC" (izbačen zadnji slog član) i u slijedećem koraku dozvoljava se samo upotreba UNSAVE-iskaza. Upotreba bilo kojeg drugog JMBP-iskaza izazvaće zaustavljanje programa uz štampanje poruke o grešci na konzoli.Ukoliko želimo da brišemo (izbacimo) sve članove nekog primjerka skupa,preporučljivo je to uraditi iskazom DELETE SET <ime-skupa> ,koji to mnogo brže radi nego iskaz DELETE RECORD.
Treća izuzetna situacija se javlja prilikom izbacivanja posljednjeg sloga iz baze.Poruka programeru je "IZSB" (javlja se u lokaciji odredjenoj ERROR-iskazom).Medjutim,efikasniji iskaz za brisanje cijele baze je DELETE ALL BASE.

Svi članovi primjerka skupa mogu se efikasno izbrisati iskazom:

```
#DELETE SET <ime-skupa> .
```

Prethodno radni pokazivač mora biti postavljen na sloga vlasnika članova primjerka skupa.Na primjer studenti Odsjeka za energetiku mogli bi se izbrisati na slijedeći način (takodje bi se izbrisali i njihovi ispiti):

```
        .
        .
        #FIND "ENERGETIKA       ".
        #DELETE SET STUDENTI.
        .
        .
```

Ukoliko programer ne vodi računa o hijerarhijskom uredjenju baze,kao na primjer:

```
        .
        .
        #FIND "ENERGETIKA       ".
        #DELETE SET ISPITI.
        .
        .
```

izvršavanje programa će se zaustaviti,a na konzoli će se pojaviti poruka o grešci.Ako je,pak,primjerak skupa prazan,prekinuće se dalje izvršavanje ovog iskaza (prekinuće se u smislu prelaska na slijedeći iskaz programa), a programeru će se saopštiti poruka "PSJP" u lokaciju odredjenu ERROR-iskazom.U ovom slučaju efekat je isti kao da je DELETE SET <ime-skupa> do kraja izvršen.Ne postoji nikakvo ograničenje na upotrebu slijedećeg JMBP-iskaza,jer se radni pokazivač ne mijenja (i dalje ostaje na slogu vlasniku).

Cijela baza se može efikasno izbrisati iskazom:

```
#DELETE ALL BASE.
```

ODVAJANJE I PRIPAJANJE PODSTRUKTURA

Ako programer želi premjestiti neki slog iz jednog primjerka skupa u drugi primjerak istog skupa,onda on prvo mora odvojiti taj slog iskazom:

```
#DETACH <ime-strukture>
```

i zatim ga pripojiti na željeno mjesto iskazom:

#ATTACH <ime-strukture> .

U DETACH-iskazu se navodi ime pod kojim odvojeni slog postaje poznat sistemu i pod kojim mu se može pristupiti ATTACH-iskazom.Prilikom premiještanja sloga premiješta se i čitava podstruktura za koju je on vlasnik.DETACH--iskaz odvaja slog na koga radni pokazivač trenutno pokazuje i postavlja radni pokazivač na slijedeći slog član primjerka skupa.ATTACH-iskaz pripaja odvojeni slog članovima primjerka skupa na koje radni pokazivač trenutno pokazuje i postavlja radni pokazivač na pripojeni slog.Naglasimo da se premiještanje slogova može vršiti samo unutar istih skupova.U jednom programu moguće je istovremeno odvojiti više slogova,a odvojeni slog se ne može pripojiti strukturi baze na više od jednog mjesta. Pretpostavlja se da nema smisla odvajati posljednji slog iz baze,jer se on ne može nigdje ni pripojiti.Prilikom upotrebe ovog DETACH-iskaza mogu se pojaviti dvije izuzetne situacije.Prva je,kada se odvoji slog član sa najvećim ključem u primjerku skupa,a primjerak skupa ne postane prazan.U tom slučaju se u lokaciju zadanu ERROR--iskazom smiješta kod "NESL" i u slijedećem koraku je dozvoljena upotreba jedino grupe JMBP-iskaza koji rade na trenutnom primjerku skupa.Upotreba drugih JMBP-iskaza biće kažnjena zaustavljanjem programa i štampanjem poruke o grešci na konzoli.Druga izuzetna situacija je kada se odvoji posljednji slog član primjerka skupa.Tada će se programeru saopštiti poruka "IZSC" i u slijedećem koraku dozvoljava se samo upotreba UNSAVE-iskaza.Upotreba bilo kojeg drugog JMBP-iskaza izazvaće zaustavljanje programa uz štampanje poruke o grešci na konzoli.Ukoliko se pri upotrebi opisanog ATTACH-iskaza pogriješi skup, program će se zaustaviti i na konzoli će se pojaviti poruka o grešci,a ukoliko se medju članovima primjerka skupa već nalazi slog sa istim ključem kao u slogu koji se želi pripojiti,na konzoli će se bez zaustavljanja programa štampati ključ sloga i programeru će se u lokaciju zadanu ERROR-iskazom saopštiti poruka "DPSL". Prebacivanje jednog studenta sa Odsjeka za energetiku na Odsjek za informatiku moglo bi se izvesti ovako:

```
.
.
#FIND "ENERGETIKA      ".
#SAVE ODSJEK.
#FIND SET STUDENTI.
#FIND FIRST.
#DETACH PRVOG-STUDENTA.
#UNSAVE ODSJEK.
#FIND "INFORMATIKA     ".
#FIND SET STUDENTI.
#ATTACH PRVOG-STUDENTA.
.
.
```

Zajedno sa studentom premješteni su i njegovi ispiti.

Odvajanje članova primjerka skupa,na čijeg vlasnika radni pokazivač trenutno pokazuje,vrši se iskazom:

#DETACH <ime-strukture> USING SET <ime-skupa>

a pripajanje tih članova novom slogu vlasniku,koji je odredjen radnim pokazivačem,iskazom:

#ATTACH <ime-strukture> USING SET <ime-skupa> .

Ime skupa odredjuje skup unutar kojeg se vrši premiještanje,a ime strukture ukazuje sistemu na članove koji se odvajaju i pripajaju.Naglasimo da se prilikom ovog premiještanja premiješta i čitava podstruktura za koju su spomenuti članovi vlasnici.Ovi iskazi ne mijenjaju radni pokazivač.Kao i kod prethodno opisanog DETACH i ATTACH-iskaza i ovdje se premiještanje može vršiti samo unutar istih skupova.U jednom programu se mogu istovremeno odvojiti članovi više primjeraka skupova,a odvojena struktura se ne može pripojiti strukturi baze na više od jednog mjesta.Ukoliko se pri upotrebi ovih iskaza ne vodi računa o hijerarhiji skupova na konzoli će se javiti poruka o grešci i program će se zaustaviti.Ako je,pri upotrebi opisanog DETACH-iskaza,primjerak skupa,čije bi članove trebalo odvojiti,prazan,programeru se saopštava poruka "PSJP" u lokaciju odredjenu ERROR-iskazom,a ako se,prilikom upotrebe opisanog ATTACH-iskaza,utvrdi da se članovi primjerka skupa ne mogu pripojiti slogu vlasniku jer odredjeni primjerak skupa nije prazan,tada se u spomenutu lokaciju smiješta poruka "PSNP".U oba slučaja ne postoji nikakvo ograničenje na upotrebu slijedećeg JMBP-iskaza.Takodje,ukoliko se pri upotrebi ovog ATTACH--iskaza pogriješi skup,na konzoli će se pojaviti poruka o grešci i program će se zaustaviti.
Premiještanje svih studenata Odsjeka za energetiku na Odsjek za informatiku moglo bi se obaviti na slijedeći način:

```
.
.
#FIND "ENERGETIKA      ".
#DETACH STUDENTE USING SET STUDENTI.
#FIND "INFORMATIKA     ".
#ATTACH STUDENTE USING SET STUDENTI.
.
.
```

ZAKLJUČCI

Realizacija čitavog sistema izvedena je metodom od vrha na dolje.Prvo je,u narativnoj formi,razradjena osnovna koncepcija,a zatim su,u nešto modifikovanom jeziku visokog nivoa PASCAL-u,definisane strukture podataka i apstraktni algoritmi.Nakon toga su ovi algoritmi razradjivani do detalja,da bi na kraju bili realizovani u asemblerskom jeziku PLAN fakultetskog računara ICL 1902A. Metoda od vrha na dolje pokazala se veoma korisnom,omogućivši podjednako dobar uvid u opšte kao i u pojedinačne probleme realizacije sistema.Takodje je omogućila uspostavljanje vrlo dobre kontrole nad izuzetno složenim i dugim asemblerskim programima.Sve ovo je dozvoljavalo da se,u toku rada na realizaciji sistema,uočavaju nedostaci i moguća poboljšanja,kao i da se,odgovarajuće izmjene,veoma efikasno i bezbolno unose u sistem.Pored toga ovakav način rada omogućava matematičko dokazivanje korektnosti asemblerskih programa (posredno preko programa napisanih u PASCAL-u),što je od izuzetnog značaja s obzirom na dužinu,kompleksnost i namjenu ovih programa.

Ovaj sistem proširuje mogućnosti struktuiranja podataka koje pruža COBOL i istovremeno oslobadja COBOL program od niza iskaza kojima je,inače,potrebno odrediti fizičku organizaciju podataka (dovoljno je u programu sa JMBP-iskazima navesti u odjeljku WORKING-STORAGE SECTION opise tipova logičkih slogova).Korisnik,poznavajući dobro hijerarhijske odnose u svojoj bazi i poznavajući naravno COBOL,može vrlo brzo savladati jezik za manipulisanje bazom podataka.Na taj način on može pisati jednostavne i elegantne programe za najkomplikovanije manipulacije sa podacima u bazi.

Struktura B-drveta,koja je uvedena kao fizička predstava članova pojedinih primjeraka skupova,pokazala se veoma pogodna za dvonivoovsku memoriju,omogućivši podjednako dobar direktni pristup slogu na osnovu zadatog ključa,kao i sekvencijalnu obradu članova primjerka skupa u odnosu na uredjenje tih članova.

U sistem je ugradjena vrlo jednostavna i efikasna zaštita koja onemogućava narušavanje strukture baze i ne dozvoljava programeru da izgubi kontrolu nad svojim programom.Medjutim,povećanjem kompleksnosti jezičkog pretprocesora i proširenjem sintakse nekih JMBP-iskaza,zaštita bi postala efikasnija,a mogućnost pojave greške u toku izvršavanja programa znatno bi se smanjila.Korisno bi bilo uvesti slijedeća proširenja:

```
#FIND NEXT AT END imperativni iskaz.
#FIND PRIOR AT END imperativni iskaz.
#FIND <identifikator> INVALID KEY imperativni iskaz.
#INSERT <ime-sloga> INTO SET <ime-skupa>
  DOUBLE KEY   imperativni iskaz.
#DELETE RECORD AT END imperativni iskaz.

  itd.
```

Medjutim,ova proširenja,koliko god u nekim situacijama doprinose sigurnosti u programiranju,isto toliko pred-

stavljaju nepotrebna ograničenja u nekim drugim situacijama.Zbog toga nam se čini da bi najbolje bilo ova proširenja uvesti kao opcionalna.Tako bi pojedini iskazi zadržali oba oblika (skraćeni i prošireni),a programer bi morao voditi računa o tome da upotrijebi odgovarajući oblik.

Postojeći sistem bi se takodje mogao proširiti uvodjenjem makro instrukcija za operacije na čitavim skupovima.Prema potrebi,mogao bi se ugraditi i poseban mehanizam za obezbjedjenje tajnosti i privatnosti podataka.

Na kraju,naglasićemo da bi se ovaj sistem,inače predvidjen samo za ugradnju hijerarhije skupova,mogao proširiti i na jedan opštiji mrežni sistem.

LITERATURA

1. ICL-ovi priručnici 1900 serije (1971):
   1.1 PLAN REFERENCE MANUAL.
   1.2 DIRECT ACCESS.
   1.3 UNIFED DIRECT ACCESS STANDARD UTILITES.
   1.4 COMPILING SYSTEMS.
   1.5 COBOL.
   1.6 COBOL PROGRAMMING.

2. Vojislav Maksimović (1976):
   REALIZACIJA HIJERARHIJSKOG SISTEMA ZA MANIPULISANJE BAZOM PODATAKA ZASNOVANOG NA B-DRVEĆU:
   JEZIČKI PRETPROCESOR.
   Diplomski rad.ETF Sarajevo.

3. Dženan Ridjanović (1976):
   REALIZACIJA HIJERARHIJSKOG SISTEMA ZA MANIPULISANJE BAZOM PODATAKA ZASNOVANOG NA B-DRVEĆU:
   ALGORITMI ZA PRETRAŽIVANJE I AŽURIRANJE.
   Diplomski rad.ETF Sarajevo.

4. Radovan Jović (1976):
   REALIZACIJA HIJERARHIJSKOG SISTEMA ZA MANIPULISANJE BAZOM PODATAKA ZASNOVANOG NA B-DRVEĆU:
   ALGORITMI ZA UMETANJE I IZBACIVANJE.
   Diplomski rad.ETF Sarajevo.

5. S.Alagić,R.Jović,Dž.Ridjanović (1976):
   IMPLEMENTING HIERARCHIES OF SETS USING B-TREES.
   Simpozijum Informatica 76 - Bled.

6. S.Alagić,A.Kulenović,M.Sarajlić (1976):
   STRUCTURED EXTENSION OF COBOL FOR HANDLING DATA BASES.
   International Journal of Information Systems, Vol. 2, No. 1.

7. R.Bayer,E.McCreight (1972):
   ORGANIZATION AND MAINTENANCE OF LARGE ORDERED INDEXES.
   Acta Informatica 1, 173-189.

8. R.W.Taylor,R.L.Frank (1976):
   CODASYL DATA-BASE MANAGEMENT SYSTEMS.
   ACM Computing Surveys (Special Issue: Data-Base Management Systems), Vol. 8, No. 1.

9. D.C.Tsichritzis,F.H.Lochovsky (1976):
   HIERARCHICAL DATA-BASE MANAGEMENT.
   ACM Computing Surveys (Special Issue: Data-Base Management Systems) Vol. 8, No. 1.

10. CODASYL DATA BASE TASK GROUP REPORT,April 1971 report,ACM, New York.

11. C.J.Date (1976):
    AN INTRODUCTION TO DATABASE SYSTEMS (The System Programming Series).
    Addison-Wesley Publishing Company.
    Reading(Massachusetts),Menlo Park(California), London,Amsterdam,Don Mills(Ontario),Sydney.

12. N.Wirth (1976):
    ALGORITHMS+DATA STRUCTURES=PROGRAMS.
    Prentice-Hall,Inc.
    Englewood Cliffs(New Jersey).

13. O.-J.Dahl,E.W.Dijkstra,C.A.R.Hoare (1972):
    STRUCTURED PROGRAMMING.
    Academic Press. London,New York.

nekateri vidiki uporabe računalnikov v sociologiji in politologiji

c. trampuž
a. ferligoj

UDK 681.3:(301 + 32)

Fakulteta za sociologijo,
politične vede in novinarstvo
Univerze v Ljubljani

Podan je kratek pregled problemov pri uporabi računalnikov v sociologiji in politologiji. Opisane so smeri razvoja bank podatkov in metod.

SOME ASPECTS OF COMPUTER APPLICATION IN SOCIAL SCIENCES. A short survey of the problems involved in the application of computers in social sciences is given and some perspectives in the concepts of data archives and methods are described.

Ena izmed karakterističnih posebnosti sedanje znanstveno-tehnične revolucije je rastoča uporaba matematičnih metod na različnih področjih znanosti in tehnike. Ta proces, ki je upravičeno dobil naziv matematizacija znanstveno tehničnega znanja, obsega sedaj tudi taka področja znanosti, za katera je veljalo, da zaradi težavnosti in kompleksnosti problematike, raziskave z matematičnimi metodami niso možne (biologija, lingvistika, ekonomija,...). Dejansko pa težavnost proučevanja pojavov v takih znanostih ne le, da ne izključuje, temveč nasprotno, zahteva oziroma predpostavlja uporabo točnih in abstraktnih matematičnih metod (Ruzavin, 1977).

V zadnjem času je tudi v družboslovju vedno bolj čutiti potrebo po kvantitativnem preverjanju predpostavk o družbenih zakonitostih. Še do nedavnega je bilo empirično raziskovanje v sociologiji in politologiji omejeno le na zelo preproste statistične obdelave na maloštevilnih podatkih, saj bi kompleksne analize večjega števila informacij zahtevale ogromno naporov in časa. Ugodnejše pogoje za kvantitativno proučevanje pojavov v družbenih znanostih nudi šele sodobna računalniška tehnologija, ki omogoča:

a. sistematično urejanje in izmenjavo velikega števila informacij, ki se pojavljajo vzporedno z vsakdanjim delom ljudi in se lahko zbirajo več let ali celo desetletij;

b. uporabo ustreznih računalniško orientiranih matematičnih metod.

Problemi pri uporabi računalnikov

Družbene znanosti se ukvarjajo z realnimi vsakdanjimi pojavi, ki jih je težko prevesti iz vsakdanjega nedoločenega jezika v natančni jezik znanosti. "Meritve", v kolikor so sploh mogoče, so obremenjene z vrsto slučajnosti in mnogimi napakami, ki jih je potrebno oceniti. To je zelo težka naloga, saj naletimo na vrsto omejitev pri izvedbi eksperimentov in jih v večini primerov ne moremo ponoviti pod enakimi pogoji. Poleg tega se družbeni pojavi spreminjajo s časom in so v svoji medsebojni povezanosti in raznovrstnosti manifestiranja "neskončni". Neposredno je pri raziskovanju določenih pojavov mogoče opazovati le majhne izseke, za kar pa je največkrat potrebno zbrati veliko število podatkov (od katerih se nekateri izkažejo kot nepomembni), da dobimo relevantne informacije v največji možni meri.

Obilje računalnikov, različnih matematičnih metod za statistično obdelavo podatkov in podatkov samih, lahko pripelje do formalne uporabe matematičnih metod in do situacij, ko raziskovalec izgubi iz vida osnovni namen numeričnih manipulacij. Obdelava empiričnega gradiva ne more nadomestiti globokega poznavanja raziskovanih družbenih pojavov. Predstavlja le dopolnilo k premišljeno postavljeni nalogi o proučevanem pojavu in izboru ustreznih metod. Žal so te teze v družboslovnih raziskavah včasih neupravičeno pozabljene (Ajvazjan, 1974).

vanje skupine sorodnih problemov iz določenega področja.

Za statistične analize v družboslovju je v svetu in pri nas že vrsta "programskih paketov" kot so na primer: SPSS, STATJOB, P-STAT, SS, OSIRIS itd. Večina naštetih programov omogoča raznovrstne transformacije in izbiro podatkov; pogojno, brezpogojno in slučajno, izbiro enot in vrsto statističnih in drugih analiz, ki so natančnejše naštete v naslednjem poglavju. Natančen pregled drugih programov, primernih za družboslovne raziskave je podan v reviji: Social Science Information 13(1974)3 na straneh 105-146.

2. Metode in programi za obdelavo podatkov v družboslovju

Analiza podatkov brez uporabe razvitejših matematičnih metod, omejena le na tabele in grafične ponazoritve, največkrat ne izčrpa vse informacije v podatkih o proučevanih družbenih pojavih. Zato si vedno več družboslovcev prizadeva poiskati primernejše tehnike za kvantitativno analizo podatkov. Računalnik omogoča uporabo postopkov z zahtevnejšimi numeričnimi prijemi (določitev lastnih vrednosti, reševanje enačb, ...) pa tudi uporabo postopkov, ki so osnovani na heurističnih metodah, kjer je potrebno pregledati veliko možnih izidov.

Statistične metode, ki so primerne za analizo podatkov v družboslovju, je mogoče urediti v skupine podobnih metod. Tak izčrpen pregled metod primernih za analizo podatkov v političnih vedah je podal Alker (1975). Podobna klasifikacija je bila izdelana v letu 1976 na Mednarodnem delovnem seminarju SOECO v Ljubljani in na Simpoziju računalniške statistike COMPSTAT v Berlinu. Na grobo lahko metode empiričnega proučevanja v družboslovju razdelimo na dve skupini: metode za merljive spremenljivke in metode za kvalitativne značilnosti.

a. Metode za analizo merljivih spremenljivk

Statistična analiza ene ali dveh merljivih spremenljivk je poznana iz klasičnih učbenikov statistike (Blejec, 1973, Jamnik, 1976). Vse pogosteje se v družboslovju za preverjanje zapletenejših trditev o povezanostih večih merljivih spremenljivk uporabljajo multivariatne metode kot parcialna in multipla korelacija, multipla linearna regresija, analiza variance, faktorska analiza, kanonska korelacijska analiza in diskriminacijska analiza (Anderson, 1958, Cooley in Lohnes, 1971, ...). Za omenjene analize imamo na voljo v Računskem centru na FSPN nekaj računalniških programov kot SPSS, SS, STATJOB, P-STAT, BMD, ... Testiranje razlik faktorskih struktur in podobnih zapletenejših hipotez (Jöreskog, 1970) omogočata programa SIFASP in ACOVS. Pomemben dosežek za empirično proučevanje v družboslovju so vzročni linearni modeli (Blalock, 1964, 1971, Golderger in Duncan, 1973, ...), s katerimi je mogoče preverjati zapletenejše vzročne zakonitosti v družboslovju. Tak, dokaj splošen vzročni model, ki vključuje kot posebne primere rekurzivni, nerekurzivni in faktorski model, je vgrajen v program LISREL.

b. Metode za analizo kvalitativnih lastnosti

Empirično proučevanje družboslovnih zakonitosti je za razliko od naravoslovnih težavnejše predvsem zato, ker so ponavadi proučevane lastnosti le kvalitativnega značaja (ordinalne ali nominalne lestvice) in zanje povečini še niso izdelani primerni matematični pristopi. Nekaj klasičnih neparametričnih testov za take lastnosti je vključenih v paketih SPSS, STATJOB in P-STAT. V zadnjem času pa je tudi na tem področju predlaganih nekaj pomembnih rešitev. Psihologi Togerson (1958), Coombs (1964) in drugi so izdelali metode lestvičenja (SPSS, SCALE). Tem so sledile metode večrazsežnostnega lestvičenja (Torgerson, 1958, Shepard, 1962. Kruskal, 1964), ki iz danih podobnosti med enotami določijo lego teh enot v Evklidskem prostoru s čim manj razsežnostmi tako, da je napetost med njima čim manjša. Za te metode sta dostopna programa MDSCAL in MINISSA. Za družboslovne raziskave je še posebej zanimiva splošnejša metoda večrazsežnostnega lestvičenja Carolla in Changa (1970), ki analizira spremembe v zaporedju matrik podobnosti (INDSCAL). Za proučevanje kvalitativnih značilnosti se vse bolj uveljavljajo tudi metode določanja skupin (Cluster metode). Programi zanje so ponavadi hitri in jih lahko uporabljamo za večje količine podatkov. Najpogosteje se uporabljajo postopki hierarhičnega določanja skupin, pri katerih je mogoče proces združevanja enot v skupine grafično predstaviti z drevesom - dendrogramom. Za te postopke so dostopni trije programi: HICLA, HICAJOHN in CLUSE. Slednji temelji na Lance-Williamsovem

Pregled in perspektive razvoja banke podatkov in metod

Izkušnje kažejo, da se razvija uporaba računalnikov v družbenih vedah med drugim tudi v naslednjih smereh:

1. Formiranje računalniško vodenih bank podatkov

Dosedanja praksa v svetu in posebno pri nas kaže, da so podatki iz večine raziskav neurejeni, neustrezno arhivirani in niso na razpolago vsem uporabnikom na tem področju. Ker ostajajo tako podatki kot rezultati (pri tem zlasti številni materiali, ki niso vključeni oz. upoštevani v tekstovnih interpretacijah) raztreseni med številnimi posamezniki, institucijami in disciplinami, prihaja do ponavljanja istovrstnih vprašanj v raznih anketah in do medsebojno nepovezanega proučevanja teh vprašanj, kar vodi do nesmiselnega trošenja energije raziskovalcev. Sistematizacija podatkov vodi k večji selektivnosti tako, da se jasneje nakazujejo predvsem tista vprašanja, ki so še nepojasnjena in tista področja, pri katerih bi ponovno zbiranje informacij pomenilo dodatno trošenje finančnih sredstev.

Urejeni arhivi naj bi napravili korak naprej k temu, da bi tudi empirične ugotovitve lahko privedle do pomembnejših teoretičnih posplošitev tj. do sinteze empiričnega in teoretičnega obravnavanja posameznih vprašanj. Sedanja razpršenost in izredno težka dostopnost rezultatov in podatkov iz že opravljenih raziskav obenem določa meje možnega teoretičnega posploševanja. Vpogled v rezultate dosedanjega dela je odvisen povsem od številnosti osebnih stikov in poznanstev, ne naslanja pa se na kakršen koli sistematično organiziran in preverjen fond podatkov. Posamezen raziskovalec mora torej nesorazmerno veliko količino časa posvetiti povsem slučajnostnemu neorganiziranemu iskanju in preverjanju relevantnih virov.

Poseben vidik, ki zaostruje potrebo po standardizaciji podatkov in njihovem sistemiziranju predstavlja proučevanje dinamičnih sistemov, ugotavljanje različnih trendov oz. zakonitosti glede povezanosti med pojavi oz. procesi v času in prostoru. Dinamičnih sistemov ni mogoče obravnavati brez kontinuitete virov informacij oz. načinov konceptualizacije v posameznih časovnih točkah. Tudi s tega vidika in zlasti z vidika marksističnega metodološkega pristopa se pojavlja upravičena kritika sedanje raziskovalne prakse, ki se predvsem omejuje na statične preseke družbenega dogajanja in s tem vodi do enostranskih interpretacij.

Sistematično urejeni in dosegljivi podatki predstavljajo skupaj z računalniško vodeno dokumentacijo tudi študijsko gradivo pri pouku sociologije in političnih ved ter informacijsko bazo za druge uporabnike.

Nadaljnja naloga arhivov podatkov je standardizacija informacij na področju sociologije, politologije in komunikologije. Za razliko od sedanjega stanja, ko vsak raziskovalec po svoje opredeljuje pojme in spremenljivke, ki jih uporablja, naj bi v prihodnje poiskušali vsaj do neke mere standardizirati tiste spremenljivke, ki se najpogosteje pojavljajo v empirično raziskovalnem delu. Le v tej meri lahko iščemo možnosti zagotavljanja širših teoretičnih posploševanj na osnovi pri nas ugotovljenih empiričnih podatkov. Urejeni arhivi bodo prispevali k zmanjšanju količine potrebnih informacij tudi glede na to, ker bodo evidentirali uporabljena kompleksna in sintetična merila posameznih sklopov medsebojno povezanih pojavov in pomembnost posameznih indikatorjev oz. spremenljivk. Selektivnost v tej smeri bo torej pomenila, da bi z manjšim številom podatkov dosegli enako ali celo pomembnejše znanje in lažjo možnost obravnave dinamičnih sistemov oz. predvidevanja sprememb v prihodnosti. Nadaljnji razvoj naj bi šel v smeri tesnejšega povezovanja številnih podatkov iz anketnih raziskav (predvsem informacij o različnih percepcijah in dispozicijah posameznikov) z obsežnim gradivom, ki ga nudi bodisi redna statistična služba v posameznih sektorjih, bodisi občasne statistične akcije in drugi arhivi pri nas in v inozemstvu.

V razširjeni koncept arhivov podatkov spadajo tudi priprave in razvoj sistemov za shranjevanje in izkazovanje podatkov. Posebno važna nadaljna naloga arhivov je zbiranje in razvoj programov za računalnike, tako za manipulacije s podatki, kot tudi za reševanje statističnih in drugih obdelav podatkov. Programi morajo biti pripravljeni tako, da so na najenostavnejši način dostopni vsem interesentom kot so npr. učitelji in raziskovalci, študentje in drugi. V zadnjem času teži razvoj k izdelavi in izpopolnjevanju že obstoječih programov, ki omogočajo reše-

(1967) obrazcu in vsebuje zmožnosti prvih dveh programov (Johnson, 1967) kot poseben primer. Obetajoči so postopki, ki temeljijo na razpoznavanju vzorcev (Žuravljov, Kamilov in Tuljaganov, 1974), za katere je v pripravi precej splošen računalniški program, katerega del je dostopen tudi pri nas (PARE). Ti postopki rešujejo za kvalitativne podatke podobne probleme kot znani multivariatni postopki za merljive spremenljivke.

Z obema skupinama metod se delno prepletajo še druge metode, kot na primer slučajni procesi (Coleman, 1964, Kemeney in Snell, 1962, Bartholomew, 1967, Fararo, 1973), simulacije (Dutton in Starbuck, 1971), ...

L i t e r a t u r a :

Ajvazjan C.A., Bežaeva Z.I., Staroverov O.V.: Klasifikacija mnogomernih nabljudenij, Statistika, Moskva 1974

Alker Jr. H.R.: An overview of polimetrics, Handbook of political science, Volume 7, 1975

Anderson T.W.: An Introduction to Multivariate Statistical Analysis, John Wiley, New York, 1958

Blalock H.M.: Causal Inferences in Non-experimental Research, Chapel Hill: University of North Carolina Press, 1964

Blalock, H.M.: Causal Models in the Social Sciences, Aldine Publishing Company, Chicago, 1971

Blejec M.: Statistične metode za ekonomiste, Univerza v Ljubljani, Ljubljana, 1973

Caroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via a N-way generalization of "Eckart-Young" decomposition, Psychometrika, 1970

Coleman, J.S.: An Introduction to Mathematical Sociology, The Free Press, New York, 1964

Cooley W.W., Lohnes P.R.: Multivariate Data Analysis, John Wiley, New York, 1971

COMPSTAT 1976 Proceedings in Computational Statistics, Physica-Verlag, Wien 1976

Coombs, C.H.: A Theory of Data, John Wiley, New York, 1964

Dutton, J.M., Starbuck, W.H.: Computer simulation of human behaviour, New York, John Wiley, 1971

Fararo, T.J.: Mathematical Sociology, John Wiley, New York, 1973

Goldberger, A.S., Duncan, O.D.: Structural Equation Models in the Social Sciences, Seminar Press, New York, 1973

Jamnik R.: Uvod v matematično statistiko, DMFA, Ljubljana, 1976

Johnson, S.C.: Hierarchical Clustering Schemes, Psychometrika, 1967

Lance, G.N., Williams, W.T.: A general theory of classificatory sorting strategies, 1. Hierarchical systems, The Computer Journal, 9 (1967) 4, 373 - 380

Kemeney, J.G., Snell, J.L.: Mathematical Models in the Social Sciences, Ginn, Boston, 1962

Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, Psychometrika (I in II), 1964

Mlinar Z.: Dialektika, empirizem in spekulacija v socioloških in politoloških raziskavah. Referat na posvetovanju: "Teoretično-metodološke osnove raziskovanja v družbenih vedah", ki ga je organiziral Marksistični center CK ZK Srbije v Beogradu, decembra 1976

Ruzavin, G.I.: Matematizacija naučnogo znanija, "Znanie" (Serije "Filozofija"), Moskva 1977

Shepard, R.N.: The Analysis of proximities: Multidimensional scaling with an unknown distance function, Psychometrika (I in II), 1962

SOECO, Report of the methodological Group, FSPN, Ljubljana 1976

Torgerson, W.S.: Theory and Methods of Scaling, John Wiley, New York, 1958

Toš N.: Metode družboslovnega raziskovanja, FSPN, 1975

Žuravljov, Ju,I., Kamilov, M.M., Tuljaganov, Š.E.: Algoritmi vičislenija ocenok i ih primenenie, "Fan" Taškent, 1974

Bartholomew, D.J.: Stochastic Models for Social Processes, John Wiley, New York, 1967

Jöreskog, K.G.: Simultaneous Factor Analysis in Several Populations, Research Bulletin 70-61, Princeton, N.J.: Educational Testing Service, 1970

the microcomputer families 8080, 6800, F8, Z80

bojan ponebšek

UDK 681.3-181.4

MOSTEK GmbH
7024 Filderstadt - 1
West Germany

MIKRORAČUNALNIČKE DRUŽINE 8080, 6800, F 8, Z 80. Članek daje pregled mikroračunalniških družin, ki trenutno dominirajo na tržišču. Ocenjene so potrebe svetovnega tržišča, pri čemer je vključenih tudi nekaj napovedi o rasti tega tržišča do leta 1980. Ocene so podane na podlagi pregleda možnosti uporabe mikroračunalnikov. Opisani sta potrebna sparaturna zgradba mikroračunalnika in programska oprema, ki bosta določenemu mikroračunalniku omogočili prodor in konkurenčnost na tržišču. Članek je zaključen s primerjavo zmogljivosti mikroračunalnikov: 8080, 6800, F 8, Z 80.

The article gives an overview over microcomputer families that dominate the market at the moment. Some estimations of world market demands and its growth until 1980 are given. These estimations are based on an overview of microcomputer application possibility. The necessary hardware architecture of microcomputer and its system software that will enable certain microcomputers a break trough and competition on the market, are describet. The article closes with the comparison of efficiency of following microcomputers: 8080, 6800, F 8, Z 80.

The microcomputer market is growing very fast.

Five years ago there was only one microprocessor on the market, delivered from a single supplier. Last year there were 55 different microprocessors delivered from 2o suppliers, and now there are 125 different microprocessors delivered from 65 semiconductor houses.

It is a very hard job in the short time to give an overview over all microcomputer families which are on the market today.

By analysing some popular microcomputer systems we will try to determine the direction in which the development will move in the near future.

By near future we mean the time up to 198o. The development is always faster than we expect! For this reason, it is not interesting to look 1o or 2o years in the future. All we can say for so long a time in the future is that it will be totally different as we know it today.

Today are some hundreds of thousands of general purpose computers installed wordlwide. There are already more than a million microprocessors in different applications today.

How many millions will be on the market in five years from now? And who can tell me how many will there be in 2o years?

Twenty years ago the typical computer application was the solution of numerical problems in the scientific laboratory. This would have been a pretty black future for the computer. The most optimistic people expected that worldwide only one or two thousand scientific labs could use a computer. Nobody anticipated the fact that the market would go in totally different directions. The hundreds of thousands of computers today do more data moving, controlling and searching as computing. The computer does today not the computing but the controlling.

Where will the market for all millions of the microcomputers in the next four or five years be?

A major semiconductor factory can today, without any problems, produce five hundred thousand microcomputers monthly. This gives you in total 6 million microcomputers yearly, at a selling price in large quantities of 5 dollars each. This means for less than an average dinner for one person in a restaurant you can have one microcomputer. But you cannot eat the microcomputers. There are today not enough people who understand how to apply a microcomputer.

The microcomputer is a complex tool. The development people must understand hardware and softwere. There is today a shortage in the market of people who can understand computer systems, and who are willing to work with microcomputers.

Today all computer manufacturers together could use for their terminals, peripheral controllers and test equipments maybe one or two million microcomputers, but 6o% of this market is not the free market. The total demand for these users together can be satisfied by one or two months at full production from one semiconductor producer.

We need desperately to look for some new users of microcomputers.

Marketing research people have discovered, or invented, more than 3o.ooo potential applications for the microcomputer.

If we suppose the price for the microcomputer with a program of up to 2 K Byte and some

input-output capabilities will cost in large quantities between 5 and 1o dollars in next year, then we can very quickly see some big volume applications.

First, the applications in which the microcomputers have already been used. Up to 5 million microcomputers will be used next year in TV game applications.

Some millions will be in the sophisticated pocket calculators. Some other millions will be in the microwave ovens, sewing machines, other programmable kitchen equipment, air conditioning, and burglar alarms.

The consumer market is already developing equipment with microcomputer, for example video recorders, movie and photo equipment, hi fi stereo tape recorders etc.

Actual industrial applications are gas pump control, motor control, and information terminal equipment. This market includes avionics and similar high reliability applications.

With up to 2o million cars production yearly, the biggest market in the near future will be the automotive market. There will be at least one microcomputer in each car.

We can see the enormous demand this will create for the people to design, development and maintenance systems. Of course, the software for all these applications must also be written.

In the computer applications you can, perhaps, use existing software, but in a market with entirely new applications the programs must be written from scratch.

The foregoing is enough to give an insight into the market which will be served by micro computers in the near future.

We will now investigate the microcomputer, what it is and how it will look in the next years.

Just to be sure that we all speak about the same items, we will now repeat some definitions:

Under the term "microprocessor" we understand a monolithic piece of silicon produced by some large scale integration technology which is packaged in an appropriate housing (often dual in line package or DIP), with some pins for interfacing with additional circuits or equipment. If we speak about a microprocessor, the central processor unit must be in this DIP. As an example of a microprocessor we have the F 8 MK 385o CPU.

The next bigger unit is the "microcomputer". Today, the microcomputer consists of a number of packages. In these DIP's we can find the CPU, some memory, and some input-output (I/O) circuits for the interface to the peripheral equipment. The minimum microcomputer today available on the market consists of two DIP's. These are the F 8 MK 385o CPU and the F 8 MK 3851 PSU (program storage unit). Both large scale integrated (LSI) circuits together have an 8 bit parallel CPU with 72 different powerful instructions, memory addressing capability up to 64 K byte (1 K is 1o24 bytes), and 256 I/O ports, with external and programmable timer interrupt, 64 Byte random access memory (RAM) in the CPU, and 1 K Byte read only memory (ROM) program storage in PSU.

In the very near future, there will be a one-chip F 8 microcomputer with all these features and program storage will be expanded to 2 K byte.

There will be several one chip microcomputers delivered from different suppliers in the next few months.

The next step which should be defined is the microcomputer system which consists of the microcomputer with some peripherals and, of course, there must be some software.

The last step in our definition is the microcomputer systems family. The members of the family are a variety of microcomputer systems which can be in one compatible family if there is some kind of software compatibility. The programs written for one microcomputer system would, almost without any changes, run on another microcomputer system. For example, the programs which are written in machine language for the 8080 microcomputer can run on the Z 80 microcomputer system. Because the 8080 instruction set is a real subset of the Z 80 instruction set, we speak about the compatibility on the machine language level in one direction.

We can speak about bidirectional compatibility of some high level language if, for example, there exist interpreters or compilers for two different microcomputer systems. The user can, for the execution of his application software written in BASIC, use different microcomputer systems with different speeds and at different costs.

We can finish now with the basic definitions.

We will here speak just about some 8 bit parallel microcomputer systems in the N-channel MOS (metal oxide semiconductor) LSI technology.

On the market there are serial microcomputers, 4 bit, 8 bit, and 16 bit parallel microcomputers in various MOS technologies and sliced 2 bit and 4 bit parallel microcomputers in bipolar technology.

All 8 bit microcomputers can, with these 16 bit address registers, address up to 64 K memory locations which usually have 1 byte (8 bits) organisation. Memory can be programmable read only memory (PROM), read only memory (ROM), random access memory (RAM), if you wish, the core memory.

The signals are bus oriented and memory and peripheral controllers can be connected on this CPU controlled bus.

The peripheral can have the 8 bit parallel or the serial data path organisation.

Because the peripheral usually has both these data organisations the 8 bit parallel microcomputer is so successful as a peripheral control.

The 4 bit is not effective enough, and the 16 bit microcomputer does not have any advantage yet costs more than the 8 bit one.

The speed of the 8 bit parallel microcomputer is somewhere between 1 and 2 microseconds for the fastest instructions.

Almost all microcomputers have the multilevel interrupt structures and direct memory access (DMA) controllers. There are today some generations of general purpose computers on the market. For general purpose computers we have

had several generations if we evaluate the technological process but in this last generation of the GPC we have had several generations of the minicomputers and microcomputers. All microcomputers are produced by some form of LSI technology, and the power of the microcomputer depends on this technology and density more than on the architecture.

The power from the microcomputer today can be compared with the power of the minicomputers which had been developed 5 years ago (PDP 11, NOVA 1200).

But we can expect in the next years that some microcomputers will be as powerful as the minicomputers. The gap between the minis and the micros is disappearing already.

Microcomputer advances are terminated by a technological revolution and not by their architecture.

We can see the most successful minicomputers in their microcomputer version PDP 11 as LSI 11 and NOVA as LSI NOVA, and we can find the 8080 in the bipolar LSI version.

This shows the overlapping of computers in different technologies at different cost and power for different applications.

The fact which makes the microcomputers so extremely interesting is that the computer power can be concentrated on a piece of silicon which can be produced extremely cost effectively.

The price for microcomputer depends, of course, on production quantity and from the yield of the production. It is totally independant from instruction set, so long as it is possible to implement these instruction sets on the area of the silican which can be optimally produced.

Today it is possible to produce economically in mass production an area between 20 and 25 square millimeters (up to 200 mills on the side).

About 8000 to 9000 MOS transistors on this area can be put with up to 40 input-output signals, if there is some random logic in the CPU. More transistors can be implemented in RAM circuits because the density is higher. But we will spend some word about memories later.

We will look now at the 4 microcomputers which are today the most popular on the market. These are the

8080 - first developed and produced by Intel and now produced by different suppliers

6800 - developed and produced by Motorla and other sources

F 8 - developed and produced by Fairchild and now produced as second source by MOSTEK

Z 80 - developed by Zilog and now produced as second source by MOSTEK.

All these microcomputers have 8 bit parallel structure and are produced in the N-channel MOS technology.

Some of the important features can we see in the table at the end of this article.

We have in this table some typical features of three microcomputers of the second generation (8080, 6800 and F8) and Z 80, the first of the third generation.

What makes the Z 80 the first of one new generation is the implementation of complex instructions such as the string move, search and string input-output, the bit manipulation and the register set architecture.

But we should compare some important features from all these microcomputers.

First, we can see that about three years ago there appeared a successor for the 8008 - the 8080.

The CPU architecture:

8080 - Register oriented, distributed in more chips because the circuits were developed in 1973 when the technology was not advanced as today, with high level clock and three power supply voltages. - This is the first really successful microcomputer and all later development must be compared with this one.

6800 - Register A and B; this CPU is more memory oriented with good addressing features, such as indexing or shift in the memory locations. The first with just one power supply. The CPU is only one chip.

F 8 - The CPU development had been concentrated on a system with absolutely minimal external circuitry. The CPU has one accumulator and a 64 byte RAM, the instruction set is ideal for these minimal systems. There are novel features in the CPU such as on chip refreshing of dynamic memory, and programmable timer interrupts, very usefull in control applications.

Z 80 - The designer had learned a lot from other designs. Well balanced design with registers in CPU to control the memory and I/O. The CPU is a combination of many good features from other microcomputers. The simple clock generation, one power supply voltage, and the possibility to stop the CPU for testing. All instructions of the 8080 are included in the Z 80.

The design had been done with the objective to be better than the 8080 and to give to the customer the possibility to use 8080 software plus new instructions.

What can we learn from this short comparison of these microcomputers? - The designers of microcomputers learn fast and they are willing to give the programmer better instructions, and the user the possibility of designing simpler systems.

All microcomputers must have a lot of additional LSI controllers. The parallel and serial I/O controllers, direct memory access circuitry and special controllers such as floppy disk, communications line etc.

In the next year we will have some one chip microcomputers. It is possible to have the CPU, up to 2 K byte ROM for programs, some RAM's and good I/O features. The circuits will go into low cost one chip designs and into sophisticated 8 bit and 16 bit microcomputers.

The memory development shows extremely good progress. We have now the start of the production of 16 K bit dynamic memory on one

chip and of 4 K bit static memory, we can expect in some years the 64 K bit memory on one chip. If we put all these facts together we can expect in 1980 a one chip microcomputer with the power of today's minicomputer, 4 K Byte ROM and 4 K Byte RAM. The problem will be to attach the peripheral to have some other combination of the memory in the system.

But the real revolution in the microcomputer architecture will first come when we have microprogramming in the microcomputer.

We can expect the same development as with general purpose and minicomputers, the big step forward can be done with implementing of complex software direct in the microcomputer, used in language interpreters, operating systems etc.

If we look at the progress in the bigger computers, we can anticipate the direction in which the microcomputers will be developed.

There is nothing new in the architecture of the microcomputer, it is the progress in the technology which has given us the possibility to repeat all features which exist in the classical computers at very low cost on a single piece of silicon.

Finally, we could perhaps ask ourselves, and the question is not too phantastic:

When will it be possible to copy the IBM 370 computer on one piece of silicon?

Table to page 32

	8080	6800	F 8	Z 80
Production start	4th Quarter 73	3rd Quarter 74	3rd Quarter 75	1st Quarter 76
Technology	N-channel MOS	N-channel MOS	N-channel MOS	N-channel MOS
Power Supply	-5, +5, +12 V	+ 5V	+ 5, + 12 V	+ 5 V
Clock generation	2 Ø special high level	2 Ø special generator	1 Ø only crystal	1 Ø TTL level
Clock frequency	2 MHz	2 MHz	2 MHz	DC - 2,5 MHz
Clock time	5oo ns	5oo ns	5oo ns	4oo ns
Shortest instruction	2 microsends	2 microseconds	2 microseconds	1,6 microseconds
Addressability	64 K	64 K	64 K	64 K
Register in CPU	A, F B, C D, E H, L	A B	A 64 Byte	A, F; A, F B, C; B, C D, E; D, E H, L; H, L
	Stack P PC	Stack P PC IX	PCO, PC 1 DCO, DC1 in PSU, SMI, DMI	Stack P PC IX IY
Memory refresh	-	-	On chip	in CPU
Instructions	78	72	72	158
String instructions	No	No	No	Yes
I/O instructions	Yes	No	Yes	Yes
Relative jumps	No	Yes	Yes	Yes
Bit manipulation	No	No	No	Yes
Indexing	No	Yes	No	Yes
Move Byte	27 microseconds 54 cycles 8 Bytes	52 microseconds 52 cycles 28 Bytes	38 microseconds 76 cycles 1o Bytes	8.4 microseconds 21 cycles 2 Bytes
Search Byte	26 microseconds 52 cycles 13 Bytes	38 microseconds 38 cycles 21 Bytes	32 microseconds 64 cycles 9 Bytes	8.4 microseconds 21 cycles 2 Bytes
I/O Byte	18,5 microseconds 37 cycles 8 Bytes	3O microseconds 3o cycles 16 Bytes	2o microseconds 4o cycles 9 Bytes	8 microseconds 2o cycles 2 Bytes

okvir za izučavanje informacionih sistema

d. ćećez - kecmanović

UDK 659.2.001.1

Odsjek za informatiku
Elektrotehnički fakultet
Univerzitet u Sarajevu

Mnogobrojni, a i u mnogočemu sporni, pravci razvoja oblasti informacionih sistema, kao i nepostojanje čvrste teorijske osnove za praktične realizacije informacionih sistema, nametnulo je potrebu za definisanjem okvira za izučavanje, u kojem bi se na izvjestan način integrisali oni teorijski rezultati koji su izdržali kritiku vremena i dobili potvrdu prakse. U nastojanju da se formuliše jedan takav okvir u radu se razmatraju tri značajna koncepta: 1) koncept objektnog sistema, 2) ciklus razvoja informacionog sistema i 3) metodske oblasti u ciklusu razvoja. Prikazane su osnovne teorijske predpostavke ovog koncepta, kao i relevantni aspekti njihove primjene. Ovi koncepti su sagledani u kontekstu razvoja teorije informacionih sistema.

Teorija informacionih sistema nije dovoljno razvijena da bi pružala jasan, koncepcijski i metodski definisan, opšte prihvaćen okvir. Postoji, medjutim, nekoliko značajnih koncepata, čija je vrijednost potvrdjena u praksi, koje ni jedan informacioni stručnjak -bilo teoretičar, bilo praktičar- nebi mogao previdjeti. Uvrštavanje značajnih koncepata u jedan teorijski okvir implicira neminovno i primjenu odredjenih kriterija relevantnosti. Svjesni ograničenosti svakog ovakvog pokušaja, želimo istaći da ovaj rad nema pretenziju da bude iscrpan, već da reflektuje neke osnovne tendencije u uspostavljanju ovog okvira.

Odredjenje sistema za koji se projektuje informacioni sistem tj. njegovog odredišnog sistema, od suštinskog je značaja za definisanje ciljeva informacionog sistema i bitno utiče na efekte informacionog sistema. Koncept "objektnog sistema" tj. sistemsko razmatranje dijela realnog svijeta — objekta za koji se gradi informacioni sistem, jedan je od fundamentalnih koncepata u izučavanju informacionih sistema, te se kao dio teoretskog okvira razmatra u ovom radu.

Ciklus razvoja informacionog sistema je proces koji mnogi mjerodavni autori prepoznaju kao specifičan u pogledu principa odvijanja i kontrole, strukturnih standarda i metoda razvoja. Na ovom konceptu osniva se definisanje metodskih oblasti u razvoju informacionih sistema, čime se zaokružuje teoretski okvir, koji je po našem mišljenju čvrsta osnova za dalji razvoj teorije informacionih sistema i u isto vrijeme nezaobilazna podloga svakom praktičnom radu u oblasti razvoja informacionih sistema.

1. OBJEKTNI SISTEM vs. INFORMACIONI SISTEM

Sistemsko mišljenje o složenim fenomenima koji se javljaju u funkcionisanju organizacija ispoljava se u a) izolaciji dijela realnosti koji je objekat posmatranja, b) posmatranju tog dijela realnosti kao komponente šireg sistema kome pripada i c) analizi objekta posmatranja kroz njegove sastavne dijelove i njihove relacije. Omedjeni dio realnosti -objekat posmatranja značajan za našu radnu namjeru- definišemo kao sistem, koji ima uspostavljene relacije sa širim sistemom, odnosno, i sam se sastoji od medjusobno povezanih dijelova, podsistema. Opisom ovog sistema definišemo objektni sistem što je oznaka za "sistemsko tretiranje objekta", /1/. Objektni sistem definiše se sa ciljem izgradnje informacionog sistema, kao njegovog budućeg dijela. Dakle, objektni sistem je širi sistem kome informacioni sistem pripada, odn. kome će pripadati.

Pojam objektnog sistema u oblasti informacionih sistema uveo je Langefors, /2/,:

> "Informacioni sistem je potreban kao pomoćni za drugi sistem, objektni sistem tj. sistem kojim se upravlja. Informacioni sistem treba da obezbjedi informacije potrebne na bilo kom mjestu, u bilo koje vrijeme, u objektnom sistemu. Objektnim sistemom često se smatra organizacije, preduzeće ili upravljačko tijelo."

Vrijednost "pridruženog" informacionog sistema sagledava se u kontekstu objektnog sistema tj. u poboljšanju efikasnosti "upravljanog sistema", /3/.

Sobzirom na našu radnu namjeru -izgradnju informacionog sistema- opis objektnog sistema može biti više ili manje

dobar,odgovarajući.Šta obuhvatiti opisom,odn. sa kojom širinom i dubinom izolacije opisati realnost, i koji je zadovoljavajući nivo izomorfizma,dva su suštinska pitanja u definisanju objektnog sistema,sa značajnim posljedicama na razvoj informacionih sistema.

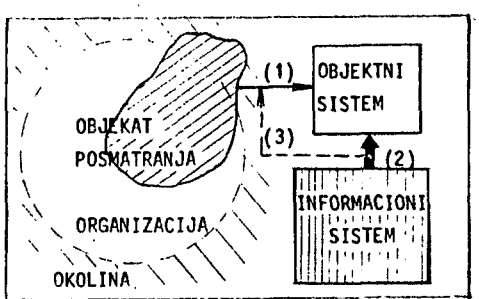
Primjena opšte teorije sistema u studiranju realnih sistema,prema autorima Matejiću i Petroviću,/4/, "u suštini nije ništa drugo već stvaranje uprošćenih,šematizovanih portreta izvjesnih aspekata sistema i okoline u kojoj on egzistira".Slika objektnog sistema uvijek je uprošćena predstava realnosti.Više ili manje formalizovana,ova slika postaje sredstvo za razvoj informacionih sistema.Nivo izomorfizma između slike objektnog sistema i realnog svijeta manifestuje se već u procesu razvoja informacionog sistema,u toku kog se u iterativnom postupku može korigovati.

Odredjenje širine i dubine izolacije Langefors označava kao probleme "spoljašnje" i "unutrašnje" granice,/5/.Granica izmedju objekta posmatranja,tj. "cjeline",koja je u fokusu našeg interesovanja,i ostalog dijela realnosti , naziva se spoljašnja granica.Ono što nije u sistemu, što nije obuhvaćeno spoljašnjom granicom,a što ima značajne posljedice za objektni sistem,pripada okolini (okruženju) sistema.Endogene varijable su one koje opisuju resurse objektnog sistema i dostupne su kontroli u okviru sistema. Relevantne interakcije izmedju objektnog sistema i okoline,koje se smatraju nekontrolabilnim u okviru objektnog sistema,opisuju se tzv. egzogenim varijablama.Endogene i egzogene varijable,sa zajdničkom karakteristikom da su značajne za sistem,razlikuju se dakle u stepenu kontrolabilnosti.

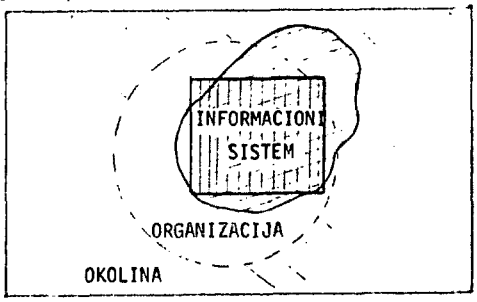
Na analogan način definiše se unutrašnja granica.Objektni sistem sastoji se od medjusobno povezanih podsistema,komponenti,koje se takodje sastoje od podsistema itd.Tako se objektni sistem opisuje hijerarhijskom strukturom.Komponente sistema koje su najniže u hijerarhijskoj strukturi i koje ne opisujemo uz pomoć subkomponenti,nego samo pomoću ulaz/izlaz karakteristika,smatramo elementarnim komponentama.Elementarne komponente odredjuju unutrašnju granicu ,jer njihova struktura nije značajna prilikom opisa objektnog sistema,za potrebe razvoja informacionog sistema.Odredjenje unutrašnje granice je pitanje o kojem tim informacionih stručnjaka mora donijeti odluku.Granice izmedju podsistema,koji čine objektni sistem,nazivaju se medjugranice.Problemi odredjivanja spoljne granice bitno se razlikuju od onih koji se javljaju kod definisanja medjugranica ili unutrašnje granice,i po načinu rješavanja i po posljedicama.

Definisanje spoljašnje granice,medjugranica i unutrašnje granice predstavlja efikasno sredstvo u razvoju sistema uopšte.Npr. izvodjenje uslova na medjugranicama iz uslova koji se postavljaju na spoljnoj granici,pomoću formalnih metoda,jedan je od ciljeva teorije sistema.

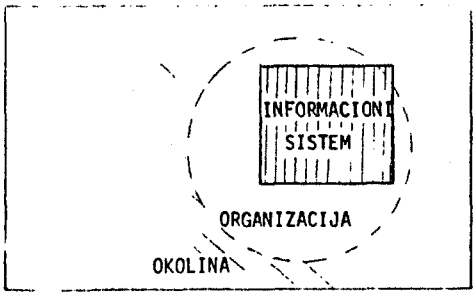
Budući "sistem nije nešto dato u prirodi stvari",/6/, tj. nije inherentna osobina realnog svijeta,odredjenje spoljašnje i unutrašnje granice,kao i medjugranica, je proizvoljno.Definisanje objektnog sistema,dakle,u sebi implicitno sadrži problem relevantnosti i problem vrijednosti.Kriterij za izbor granica treba da izražava cilj posmatranja realnog fenomena kao sistema (u objektni sistem preslikavamo ono što je važno za našu radnu namjeru),i da sadrži mjerila relevantnosti pojedinih činjenica.Kriteriji relevantnsti su nužno subjektivni i velikim dijelom zavise od znanja i iskustva informacionih stručnjaka.Kriteriji se mijenjaju ne samo tokom radnog vijeka stručnjaka,već i u toku jedne aplikacije može da dodje do njihove korekcije.



a) (1)definisanje objektnog sistema, (2)analiza i projektovanje informacionog sistema, (3)redefinisanje objektnog sistema.



b) implementacija informacionog sistema



c) funkcionisanje informacionog sistema

Slika 1. UGRADNJA INFORMACIONOG SISTEMA U ORGANIZACIJU POMOĆU OBJEKTNOG SISTEMA

Na sl.1. ilustrovana je upotreba koncepta objektnog sistema kao sredstva u razvoju informacionog sistema.Objekt posmatranja se opisuje -na sl.1.a) označeno sa (1)- kao

sistem sa ciljem da informacioni sistem,projektovan (2) za taj objektni sistem,bude dovoljno dobar u realnom kontekstu objekta posmatranja.Koliko će informacioni sistem biti dobar,koliku vrijednost će predstavljati za korisnike zavisi i od toga kakav je kvalitet objektnog sistema, te prema tome i od kriterija relevantnosti.

U toku analize i projektovanja informacionog sistema,mogu se uočiti slabosti,netačnosti i nedovoljna preciznost (tj. neodgovarajući stepen izomorfizma) u opisivanju objekta posmatranja.Intervencije mogu da se odnose na korekciju kriterija relevantnosti,što je prikazano isprekidanom strelicom (3).

Na sl.1.b) prikazana je implementacija projektovanog informacionog sistema u dio realnosti koji je predstavljen objektnim sistemom.Projektovan prema potrebama objektnog sistema,implementirani informacioni sistem ostvaruje svoju vrijednost kroz zadovoljenje potreba,koje proizilaze, ne samo iz izolovanog dijela realnosti,nego i iz šireg sistema,sl.1.c).Dok ostajemo u okvirima objektnog sistema pitanje vrijednosti ostaje jednodimenzionalno: informacioni sistem naspram objektnog sistema.Pogledamo li,međutim,šire,objekat posmatranja kao komponentu šireg sistema, npr. organizaciju kao dio privrednog sistema, pitanje vrijednosti postaje višedimenzionalno.Funkcionisanje komponente uvijek odražava djelovanje cjeline kojoj pripada,tj. šireg sistema.U opisu komponente nužno su sadržane predpostavke o prirodi šireg sistema,koje su najčešće invarijantne u odnosu na promjene šireg sistema.Izdvajanje komponente-podsistema iz cjeline odn. šireg sistema,uvijek je uprošćenje,više ili manje krektno. Zbog toga je valjanost objektnog sistema ograničavajući faktor vrijednosti informacionog sistema.

Prije nego razmotrimo posljedice primjene koncepta objektnog sistema,daćemo primjere njegove upotrebe.

Kao objektni sistem može se opisati jedan ili više funkcionalnih dijelova organizacije uključujući operativni i/ili više nivoe upravljanja.Npr. za izgradnju proizvodnog informaiconog sistema objektni sistem može biti slika proizvodne radne jedinice sa odlukama koje se donose u okviru samoupravnih organa radne jedinice,samoupravnih organa na nivou OOUR-a i,eventualno,na nivou radne organizacije.U objektni sistem mogu biti uključene djelomično i funkcije nabave i prodaje,zavisno od toga kolike su ambicije korisnika u pogledu izgradnje proizvodnog informacionog sistema.Ukoliko se želi izgraditi "totalni" informacioni sistem radne organizacije tada se u objektni sistem preslikava radna organizacija sa dijelom svoje okoline (interakcije sa drugim radnim organizacijama u okviru složene organizacije i izvan nje).No budući se totalni informacioni sistem formira integracijom pojedinih sistema,za definisanje objektnog sistema,u ovom slučaju,veoma je važno određenje i njegovih međugranica.

Posljedice usvajanja koncepta objektnog sistema, kao sredstva u razvoju informacionog sistema, su dvojake: 1) Prilikom razvoja informacionog sistema problemi samog objekta eksplicitno se razmatraju,i 2) analiza objektnog sistema postaje dio ciklusa razvoja informacionog sistema, i usljed metodskih specifičnosti predstavlja posebnu metodsku oblast.

Analizom objektnog sistema otkrivaju se slabosti i neracionalnosti u osnovnom radnom procesu organizacije (to može bi proizvodnja, uslužna djelatnost, transportna djelatnost, hospitalizacija, obrazovni proces i sl.) kao i u realizaciji samoupravnog sistema odlučivanja. Rješenja uočenih problema nisu samo u poboljšanju informacionog sistema. Ciljevi kao npr. povećanje ekonomičnosti proizvodnje ili povećanje efikasnosti samoupravnog sistema odlučivanja, mogu se ostvariti (ostvarivati) poboljšanjima organizacije rada i samoupravnih procedura, sa jedne strane, i obezbjedjenjem informacione podloge odgovarajućeg kvaliteta,sa druge. Tako se uloga informacionih stručnjaka ne može ograničiti samo na utvrdjivanje informacionih potreba i projektovanje informacionog sistema koji će te potrebe zadovoljiti (takav pristup bi se opravdano mogao nazvati "tehnokratskim"). Uloga informacionih stručnjaka je daleko šira i daleko odgovornija. U saradnji sa budućim korisnicima informacionog sistema - upravljačima i rukovodiocima- informacioni stručnjaci moraju identifikovati probleme i istraživati rješenja kroz organizacione, tehničko--tehnološke i dr. promjene,imajući u vidu mogućnosti informacione tehnologije. Tako koncept objektnog sistema omogućuje eksplicitno tretiranje ovih problema u svjetlu razvoja informacionih sistema i utvrdjivanje ambicija korisnika u pogledu promjena u objektnom sistemu. U definiciji objektnog sistema implicitno je sadržana i odluka o radikalnosti zahvata u realni sistem. Ambicije korisnika mogu da variraju od zahtjeva za projektovanjem novog, poboljšanog objektnog sistema,do insistiranja na minimalnim promjenama i tretiranja zatečenog stanja kao polaznog u projektovanju informacionih sistema. Na taj način,u toku analize objektnog sistema više ili manje je uključeno projektovanje samog objekta.Kao rezultat dobije se objektni sistem za koji treba da se projektuje informacioni sistem.

U praktičnim realizacijama informacionih sistema ovi problemi su se rješavali i bez formalnog isticanja i izdvajanja analize objektnog sistema. Koncept objektnog sistema bio je ustvari intuitivno korišten. I u svakodnevnom životu, u razumjevanju naše okoline, analizu situacija pravimo na osnovu naših predstava o realnim situacijama. Formulacija koncepta objektnog sistema nije ništa drugo već ekspliciranje i formalizovanje izražavanja određenih spoznajnih procesa pri projektovanju informacionog sistema za, projektantima nepoznati, realni objekt.

Eksplicitno razmatranje objektnog sistema i njegovo djelomično projektovanje, po metodama,modelima i sredstvima ra-

zlikuje se od faza koje ga slijede (analiza i projektovanje informacionog sistema). Zato se analiza i (djelomično) projektovanje objektnog sistema tretira, kao što ćemo kasnije vidjeti, kao posebna metodska oblast razvoja informacionog sistema.

2. CIKLUS RAZVOJA INFORMACIONOG SISTEMA

Osnovni koraci u razvoju informacionih sistema -inicirranje, analiza, projektovanje, konstrukcija, testiranje, implementacija, funkcionisanje i modifikacija- uočeni su još u najranijim primjenama računara u organizacionim problemima, /7/. Pojavila se potreba za definisanjem "ciklusa razvoja informacionog sistema", /8/, za ispitivanjem zakonitosti njegovog odvijanja i instrumenata kontrole.

Dosadašnji razvoj oblasti informacionih sistema doveo je do spoznaje da je ciklus razvoja informacionog sistema specifičan i složen proces,u kome su dominantna dva problema: 1. strukturiranje ciklusa za potrebe efikasnog izvodjenja i upravljanja,
2. metodski aspekti razvoja informacionog sistema.
Modeli ciklusa razvoja informacionog sistema razmatraju se u ovom odjeljku. Drugo pitanje je predmet razmatranja u narednom.

Jedan od osnovnih principa ciklusa razvoja informacionog sistema je distinkcija "ŠTA" vs. "KAKO" tj. ŠTA informacioni sistem treba da realizuje, za razliku od načina KAKO će to da realizuje. Zajednični imenitelj problema koji se pojavljuju u odredjenju "ŠTA informacioni sistem treba da pruži ?" je identifikovanje informacija potrebnih korisniku, te se ova kategorija problema, prema Langeforsu, naziva "informatološkim" problemima. Za razliku od informatološkog nivoa, na "datološkom" nivou posmatraju se problemi kreiranja sistema za obradu podataka i strukturiranje podataka tj. oni koji daju odgovore na "KAKO realizovati informacione zahtjeve?", /9/.Iako su oba aspekta podjednako važna u ciklusu razvoja informacionog sistema,datološki problemi bili su u centru interesovanja,dok su informatološki bili ili samo površno razmatrani ili potpuno ignorisani. Realtivno česti neuspjesi u realizaciji informacionih sistema i nezadovoljstvo korisnika zbog neodgovarajućih efekata informacionih sistema ukazali su na značaj informatoloških problema. U teoriji informacionih sistema došlo je do pomjeranja interesa od datološkog prema informatološkom nivou.

Eksplicitno izdvajanje i predhodno izučavanje pitanja "ŠTA" (u odnosu na "KAKO") omogućilo je i računarski nezavisan pristup analizi informacionih sistema.Upravo,kako će odredjeni zahtjevi biti realizovani (i da li uopšte mogu biti realizovani) zavisi prije svega od tehnologije, zatim i od opreme,odn. postavlja odredjene zahtjeve na računarsku instalaciju.Pitanja tehnologije,kao i izbora (ili adaptacije postojeće) opreme rješava se,dakle,u fazi projektovanja i "odgodjeno" je što je moguće kasnije u procesu razvoja informacionog sistema.

Distinkcija "ŠTA/KAKO" u razvoju informacionih sistema smatra se značajnim dostignućem.Na njoj se bazira strukturiranje ciklusa razvoja.

Ciklus razvoja je u znatnoj mjeri kreativan zadatak te,po svojoj prirodi,teži slobodi forme i vremenskoj neodredjenosti.Definisanjem specifičnih grupa zadataka tj. faza u razvoju,ciklusu se nameće struktura: svaka faza posmatra se sa stanovišta odnosa prema susjednim fazama,ulazne i izlazne dokumentacije,metoda razvoja,vremena i troškova odvijanja.

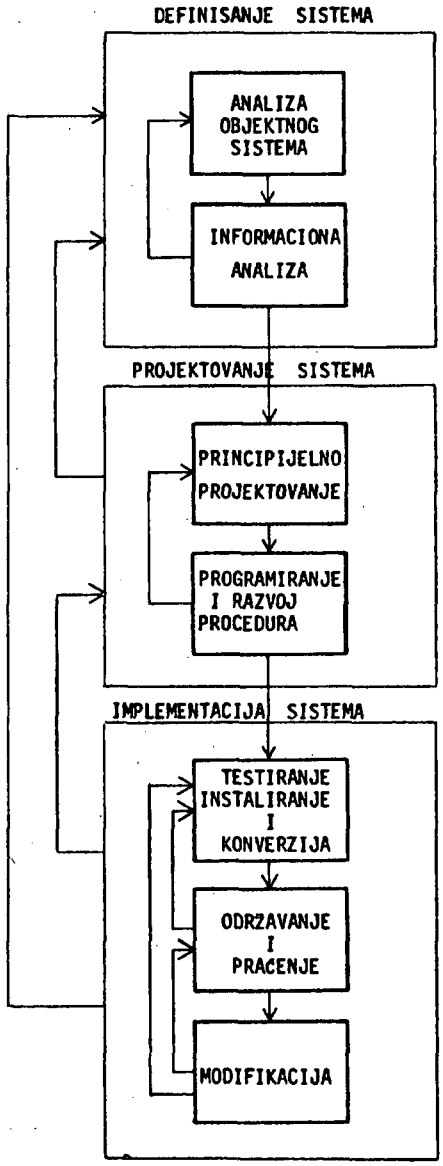
U razvoju informacionog sistema učestvuju dvije bitno različite grupe ljudi,informacioni stručnjaci koji razvijaju sistem i samoupravljači i rukovodioci - budući korisnici sistema.Razumijevanje ciklusa razvoja i zajednički prepoznavanje faza razvoja neophodna je predpostavka saradnje ovih dviju grupa.Pored toga,grupa informacionih stručnjaka sastoji se od različitih vrsta specijalista: informacionih analitičara,projektanata sistema,programera i dr.Ove grupe stručnjaka obavljaju pojedine vrste zadataka u slijedu u kojem se oni pojavljuju u ciklusu razvoja (analiza,projektovanje,implementacija).Odredjenje faza i odgovarajuće dokumentacije razvoja informacionog sistema u okviru faza omogućuje komunikaciju izmedju ovih grupa stručnjaka i koordinaciju aktivnosti u okviru ciklusa.

Može se zaključiti da su strukturiranjem ciklusa razvoja tj. definisanjem faza razvoja stvoreni instrumenti za upravljanje ciklusom i kontrolu realizacije informacionog sistema.

Polazeći od distinkcije "ŠTA/KAKO" ciklus razvoja se dijeli na tri globalne etape : -definisanje sistema
-projektovanje sistema
-implementacija sistema
koje su prikazane na sl.2. U etapi definisanja sistema utvrdjuje se šta informacioni sistem treba da bude. Ova etapa se dijeli na dvije faze: analizu objektnog sistema i informacionu analizu.Analiza objektnog sistema obuhvata analizu sistema upravljanja i djelomično projektovanje upravljačkih procedura,kao što smo vidjeli u predhodnom odjeljku.Rezultat ove faze je slika objektnog sistema i utvrdjene ambicije korisnika u pogledu promjena i usavršavanja u sistemu upravljanja.Informaciona analiza specificira informacione potrebe objektnog sistema dajući sliku željenog informacionog sistema,u formi razumljivoj za korisnika.Ona uključuje i analizu ostvarljivosti u tehničkom,operativnom i ekonomskom smislu.

U etapi projektovanja sistema informacioni zahtjevi iz etape definisanja prevode se u fizički sistem programa, procedura i podataka, tzv. sistem za obradu podataka. Etapa se odvija kroz dvije faze: faza principijelnog pro-

jektovanja, u kojoj se definiše sistem za obradu podataka, uključujući strukture podataka u bazi podataka, i detaljne specifikacije za programiranje i razvoj procedura; faza detaljnog projektovanja obuhvata programiranje i testiranje programa, detaljnu razradu i testiranje procedura.



Slika 2. CIKLUS RAZVOJA INFORMACIONOG SISTEMA

Etapa implementacije počinje fazom testiranja, instaliranja i konverzije na novi sistem za obradu podataka, nakon čega dolazi faza održavanja sistema i praćenja njegovog funkcionisanja. U toku rada jednog informacionog sistema pojavljuje se i potreba za njegovom modifikacijom, te se modifikacija sistema smatra fazom u ciklusu razvoja informacionog sistema. Tako se s pravom ciklus razvoja naziva i "životnim ciklusom informacionog sistema".

Razvoj informacionog sistema nije "gladak" proces, kao što bi to moglo izgledati iz prethodnog pasusa. Svaki daljI korak u razvoju sistema verifikuje postavke iz prethodnih koraka, na koji način "slika" informacionog sistema u toku ciklusa razvoja postaje realističnija, ostvarljivija i adekvatnija zahtjevima. Mnoge greške ili pogrešne predpostavke napravljene u prvim fazama ciklusa otkriju se tek u kasnim fazama, (npr.tokom implementacije) što neminovno dovodi do vraćanja na prethodne faze. Ciklus razvoja informacionog sistema je iterativan proces *i na planu etapa razvoja i u okviru etapa - izmedju faza*, što je prikazano na sl.2.

Upravo davanje većeg ili manjeg značaja iterativnosti postupka razvoja informacionego sistema opredjelilo je autore u pogledu definisanja strukture ciklusa razvoja tj. izdvajanju faza razvoja i oštrine granica izmedju faza.

Veoma oštru granicu izmedju analize i projektovanja uočavaju mnogi autori, definišući analizu kao "odozgo prema dole" postupak, a sintezu (projektovanje) kao izgradnju "odozdo prema gore", /10/. Bez obzira koliko iterativnost doprinosi sukcesivnim izmjenama ovih postupaka, lako se uočava da se postupak analize suštinski razlikuje od projektovanja. Ovi autori prave sličnu distinkciju izmedju projektovanja i implementacije.

Neki autori, medjutim, upravo u iterativnosti vide argument za eliminisanje granica izmedju faza. Npr. Ogdin, /11/, ukazuje na neopravdanost razdvajanja projektovanja i implementacije videći ih kao jedinstven "oscilatorni proces izmedju tehnika "odozgo prema dole" i "odozdo prema gore"".

Imajući u vidu da iterativnost ciklusa razvoja bitno utiče na efikasnost, kao jedan od ciljeva upravljanja ciklusom postavlja se smanjenje iterativnosti do tolerantnih granica. *Prema tome, iterativnost ne smije biti tretirana kao odredjujući faktor, već kao osobina na koju se može uticati.* Ovaj zaključak ima kao posljedicu favorizovanja više strukturisanih pristupa ciklusu razvoja.

Prvi korak u primjeni strukturiranog ciklusa razvoja informacionog sistema bio je definisanje principa i pravila za izvodjenje pojedinih faza ciklusa. Izdati su mnogi udžbenici i priručnici koji na ovakav način razmatraju ciklus razvoja. Iako pojedini autori različito definišu strukturu ciklusa -što je rezultat ponekad koncepcijskih, a češće formalnih razlika- uočljivo je opšte slaganje u pogledu vrste i toka aktivnosti u ciklusu, /12/.

Naredni korak predstavlja definisanje standarda i manuelnih, više i manje formalnih, procedura za ciklus razvoja informacionog sistema. Uočena je tendencija razvoja internih *manuelnih procedura u pojedinim organizacijama*, djelomično usljed nedostatka opšte prihvatljivih pristupa i definicija strukture ciklusa, a djelomično usljed otpora prema upotrebi sredstava koje su razvili drugi, izvan odjeljenja ili organizacije, /7/.

Ipak potreba za sveobuhvatnijim i opštijim procedurama, za većom formalizacijom, brzo je uočena. Pojavile su se značajne knjige koje razmatraju strukturne standarde ciklusa razvoja, definišu faze razvoja sa stanovišta ulazno/izlazne dokumentacije, preciziraju procedure u formi dijagrama toka, /13/.

Na ovom nivou analize ciklusa moramo uzeti u razmatranje i metodsku stranu razvoja informacionog sistema. Dok su metode i tehnike razvoja sistema bile manuelne, za upravljanje ciklusom razvoja korištena su uspješno i manuelna sredstva. Sa povećanjem pomoći računara u razvoju informacionog sistema (povećane mogućnosti programskih jezika, pojava sistema za upravljanje bazom podataka) manuelni standardi postaju sve manje upotrebljivi, a potreba za formalizovanijim tretiranjem ciklusa sve uočljivija. Neobičan je nesklad, kako zapaža Teichroew, /7/, izmedju tendencije informacionih stručnjaka da automatizuju sve procedure kod svojih klijenata i korištenja manuelnih procedura u sopstvenom radu. ("Oni su tolerisali svoje manuelne metode koje bi inače kod klijenata smatrali netolerantnim").

Istraživanje metoda razvoja ukazalo je na potrebu (re)definisanja metodskih oblasti. Savremeni pristupi ciklusu razvoja kao i tehnološka sredstva koja se koriste odrazili su se i na odredjenje metodskih oblasti u ciklusu.

3. METODSKE OBLASTI U CIKLUSU RAZVOJA INFORMACIONOG SISTEMA

Slijedeći proces ciklusa razvoja informacionog sistema uočavaju se različite problemske oblasti. Različite po vrsti objekta koji se posmatra, po sredstvima koja se koriste, po znanjima i umijećima koja su za to potrebna.

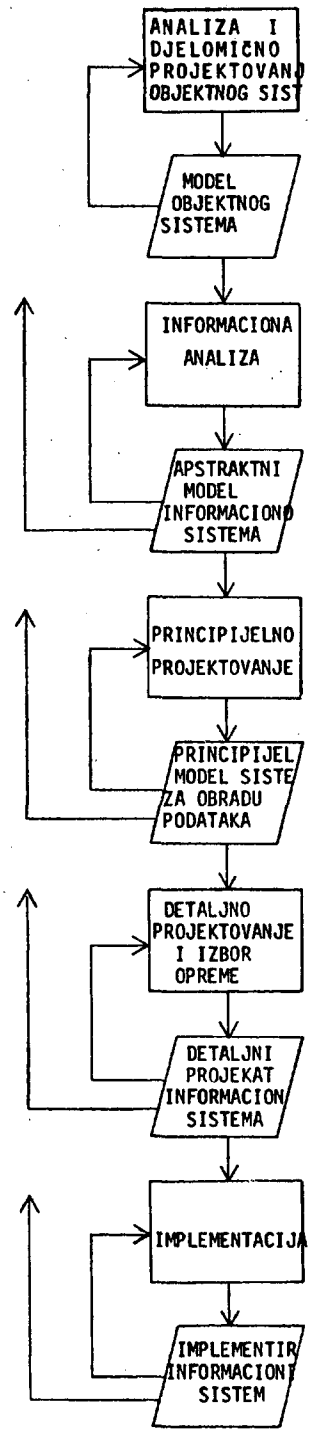
Počevši od analize objektnog sistema pa do implementacije informacionog sistema, objekat rada informacionih stručnjaka se mijenja. U fazi analize objektnog sistema to je živi socio-ekonomski sistem, upravljačke procedure, ponašanje ljudi i njihove mogućnosti korištenja informacione tehnologije. U analizi objektnog sistema sagledavaju se i mogući efekti informacionog sistema. Kao sredstvo u formiranju slike objektnog sistema koriste se tehnike ankete i intervjua, tehnike analize dokumenata, normativnih akata i sl. Pored osnovnih znanja iz tehnologije i metoda razvoja informacionih sistema, veoma su značajna znanja i iskustva iz organizacionih nauka, ekonomskih nauka i bihejvorističkih disciplina.

Već u narednoj fazi, informacionoj analizi, objekat je sam informacioni sistem, jer se, polazeći od opštih zahtjeva, definisanih u objektnom sistemu, analizira potrebna informaciona podrška u funkcionisanju i upravljanju u objektnom sistemu. U identifikovanju informacionih zahtjeva kao sredstva se koriste tehnike intervjua i ankete. Za strukturiranje sistema za obradupodataka koriste se manje ili više formalizovane metode dekompozicije informacionih skupova (zahtjeva) sve do baznog skupa podataka i skupa procesa. Budući opis sistema za obradu podataka ostaje na apstraktnom nivou -opis se odnosi na podatke, relacije medju njima, operacije koje se nad njima vrše- problemi u ovoj oblasti ostaju informatološke prirode, nezavisni od računara i opreme. Potrebna znanja u ovoj oblasti su znanja iz upravljanja i korištenja informacija u upravljanju, modela operacionih istraživanja, pored onih iz informacionih struktura, tehnika projektovanja informacionih sistema, posebno, tehnika informacione analize.

U fazi principijelnog projektovanja na osnovu apstraktnog modela sistema za obradu podataka, vrši se strukturisanje datoteka i informacionih procesa prema kriteriju maksimalne efikasnosti budućeg sistema za obradu podataka. U ovoj fazi vrši se izbor tipova obrade podataka (sekvencijalni/direktni pristup) i savršenosti sistema komunikacije korisnik-računar (off line/on line). Odluka o opremi odgodjena je za fazu detaljnog projektovanja u kojoj se vrši programiranje i izbor gotovih software-skih sistema i definišu manuelne procedure. Objekt interesa u ove dvije faze je sistem za obradu podataka, a cilj je konstrukcija takvog sistema koji će na najefikasniji način (npr. min. troškovi) realizovati zahtjeve postavljene u apstraktnom modelu informacione analize. Za razliku od problema u etapi definisanja sistema, koji su vezani za korisnika informacija, problemi u projektovanju sistema su datološke prirode i odnose se na podatke, njihovo strukturiranje i obradu, prema kriteriju maksimalne efikasnosti.Sredstva koja se koriste u ovoj etapi doživjela su znatan razvoj posljednjih godina. Intuitivan izbor "najboljih" struktura podataka, manuelne upute za projektovanje, zamjenjuju se više formalizovanim sredstvima kao što su standardi za projektovanje i dokumentaciju sistema, simulatori, sistemi za upravljanje bazom podataka i td. Potrebna znanja informacionih stručnjaka - projektanata sistema obuhvataju metode projektovanja, programske jezike, strukture podataka i sisteme za upravljanje bazom podataka.

U etapi implementacije objekat postaje informacioni sistem sa svim programima, bazom podataka i manuelnim procedurama,koji na kraju faze instaliranja postaje dio objektnog sistema odn. dio realnog sistema. U ovoj etapi informatološki i datološki problemi se isprepliću odn. pojavljuju se zajednički efekti informatološkog i datološkog projektovanja. Sredstva koja se koriste su raznolika, od ad hoc testiranja i standardnih testova do kontrole i procedura rada informacionih sistema uz pomoć računara. Potrebna znanja su,prije svega,iz hardware-a i software-a, zatim iz programskih jezika, pored opštih znanja iz razvoja informacionih sistema.

Metodske oblasti definišu se prema grupama problema specifičnih po objektu, načinu komunikacije sa objektom, sredstvima rada, tehničko-tehnološkoj pomoći u radu, skupo-



Slika 3. METODSKE OBLASTI U CIKLUSU I NJIHOVI REZULTUJUĆI MODELI (Adaptirano prema Lundebergu,/14/)

nu definisanosti problema i sl.Na slici 3. prikazan je globalan metodološki pristup razvoju informacionih sistema.Izdvojene su metodske oblasti i modeli koji su rezultat razvoja u pojedinim oblastima.

Distinkcija metodskih oblasti omogućuje :
- razbijanje kompleksnog zadatka razvoja informacionih sistema na dijelove koje jedan čovjek,stručnjak,može da obuhvati,
-specijalizaciju informacionih stručnjaka po metodskim oblastima
-koncentraciju istraživanja metoda po pojedinim oblastima.

Metode razvoja u pojedinim oblastima ciklusa definišu : a) "jezik" za prezentiranje opisa - modela odgovarajućeg sistema, b) tehnike generisanja modela odn. transformisanja jednog (ulaznog) modela u drugi (izlazni) model, c) sistem dokumentacije modela.

Novija istraživanja metoda razvoja pokazuju tendencije povećanog učešća računara u razvoju informacionih sistema. Istražuju se jezici za opis modela koji su razumljivi za računar, tehnike razvoja modela koje se koriste uz pomoć računara ili su potpuno automatizovane. Najavljena IV generacija metoda razvoja informacionih sistema, već po svojim prvim koncepcijama i prvim dilemama, oglasila se kao nosilac formalizacije i automatizacije metoda u okviru ciklusa razvoja. Kao cilj se postavlja skraćenje ciklusa i izgradnja informacionih sistema koji zadovoljavaju potrebe korisnika na "optimalan" način, /15/.

Automatizacijom metoda u pojedinim metodskim oblastima i njihovim (automatskim) povezivanjem vrši se integracija ciklusa na jednom višem nivou.Na putu da se ostvari potpuna automatizacija ciklusa stoje još mnogi neriješeni problemi, mahom oni informatološkog karaktera.

* * *

Strukturiranjem ciklusa razvoja informacionog sistema i omedjenjem metodskih oblasti stvoreni su neophodni uslovi za razvoj discipline informacionih sistema. Istraživanja metoda su u vrhu naučnog interesa u ovoj disciplini. To svakako svjedoči da je disciplina informacionih sistema na samom početku razvoja. Madjutim, veliki broj opsežnih istraživanja koja se posljednjih godina obavljaju u svijetu svjedoče o potrebi da se taj početak prevazidje.

Reference

1.Orchard,R.,"ON an Approach to General Systems Theory", in Klir,G.J.,ed.,Trends in General Systems Theory, John Wiley and Sons,New York,1972, p.3.

2.Langefors,B., Theoretical Analysis of Information Systems, Auerbach Pub.Inc.Phil.,1973,(sec.ed),p. 199.

3.Langefors,B.,op. cit. ,p. 224.

4.Matejić,V. i R. Petrović, Kompleksni transportni sistem u svjetlu opšte teorije sistema,"Mihajlo Pupin", Beograd,1973.p.14.

5.Langefors,B.,op. cit., p.56-58.

6.Beer,S., Decision and Control, John Wiley and Sons,New York,1966,p.242.

7.Teichroew,D.,"Improvements in the System Life Cycle",

IFIP Congress 1974, Stockholm, p.972-979.

8. Termin "information system development life cycle" pojavljuje se kod :
Benjamin,R.I.,Control of the Information System Development Cycle, John Wiley and Sons, New York 1971,
Teichroew,D.,ed.,"Education Related to the Use of Computers in Organizactions", Comm. of ACM , sept. 1971,vol.14,No.9.,p.573-588.
a kasnije i kod drugih autora.

9. Distinkcija "STA/KAKO" u ciklusu razvoja informacionog sistema samo je uprošćena interpretacija "Fundamentalnog principa sistemskog rada" koji je formulisao Langefors u knjizi /2/. Odgovarajuća distinkcija "informatološki/datološki" (infological/datalogical) eksplicirana je kasnije u radovima :
Langefors,B.,"Information Systems", IFIP Congress 1974, Stockholm, p.937-945.
Langefors,B.,"On Information Structure and Data Structures",Report,TRITA-IBADB-1019, Symp. "Informatica 76",Bled.

10. Gregory,R.H. and R.L. VanHorn, Automatic Data Processing Systems, Wodsworth,Belmont,Calif.,1963.
Nunamaker,J.F.,Jr.,"A Methodology for the Design and Optimization of Information Processing Systems" Proc.of the 1971 Spring Joint Comp. Conf.,AFIPS Press,vol.38.,may 1971,p.283-294.
i drugi autori.

11. Ogdin,J.L.,"Designing Reliable Software",Datamation, july 1972, p.71-78.

12. Uporedni prikaz ciklusa razvoja daje Teichroew,D., "Education Related to the Use of Computers in Organizations",Comm.of ACM,sept.1971,vol.14, No.9.,p.573-588.

13. Dvije zapažene knjige preporučamo čitaocu:
Benjamin,R.I.,ibid.
Krik,F.G., Total System Development for Information Systems,John Wiley and Sons,NewYork,1973.

14. Lundeberg,M.,"Information Analysis",Royal Institute of Technology,Stockholm,TRITA-IBADB-4401, 1971.

15. Teichroew,D. and H.Sayani,"Automatization of System Building",Datamation,aug.15.,1971, p.25-30.

A FRAMEWORK FOR INFORMATION SYSTEM RESEARCH - As information system application grow in number and variety the need for theoretical foundation of information systems work increasingly emerges. A number of concepts have appeared in the information system theory. The framework integrating those concepts that has been varified in practice, is needed. Attempting to contribute the formulation of such a framework, this paper discusses three concepts: a) the concept of an object system, b) the life cycle of the information system development, and c) method areas in the life cycle. The theoretical basis of these concepts, their interconnections and some important aspects of their applications are considered.

a model of secure computer system

sead muftić

UDK 681.3.004.58

Zavod za ekonomsko planiranje
Sarajevo

MODEL SIGURNOG KOMPJUTERSKOG SISTEMA - Zaštita kompjuterskih sistema i informacija postaju veoma značajna područja naučnog rada u oblasti kompjuterskih nauka. Jedan od najvažnijih problema u tim oblastima je realizacija sigurnog kompjuterskog sistema. Rješenje tog problema zahtjeva najprije specificiranje svih mogućih opasnosti i listu karakteristika sigurnog kompjuterskog sistema. Sljedeći korak bila bi realizacija sistema. Opis operacija u njemu i praktična provjera rada takvog sistema.
U ovom članku dat je konceptualan opis sigurnog kompjuterskog sistema, t.j. njegov model. Model sugeriše globalne komponente sistema i njihovo madjusobno djelovanje. Kao jedan od prvih članaka koji se bavi ovom problematikom, on specificira više problematična pitanja nego što nudi finalna rješenja. Osnovna svrha ovog članka bila je da istakne ozbiljnost problema i da podstakne dalji naučni rad na konačnoj realizaciji sigurnog kompjuterskog sistema.

Computer sicurity and information privacy are becoming very important areas for scientific research in computer science. One of the outstanding problems is the design of secure computer system. A solution of that problem requires the specification of accepteble threats together with a list of characteristics of secure computer system. Then the next step should be the design of the system, description of its implementation of such a system. This paper gives a general description of a secure computer system, i.e. the model of the system. It just suggest certain components and their interaction. As one of the first papers of this type in the field of secure computer system, it opens more questions than it offers solutions. Therefore the main purpose of this paper is to point out the severity of the problem and to initiate further research towards the realization of a secure computer system.

KEYWORDS : Security, access control, cryptography, secure computer system.

1. INTRODUCTION

With the recent development of large computer systems, multiprocessors, computer networks and large data bases, chances that information stored and used in the computer system will be retrieved illegally are much higher. Also computer data are often classified with respect to privacy considerations into various categories (special, private, confidential, secret, etc.), and that requires special handling and use of such data.

In large and general purpose computing facilities the question naturally arises of protecting the user's stored programs and data against unauthorized access by other users and programs. The same is the case with system control programs, utility programs, and library routines. During the operations of the computer system they are exposed to accidental disclosures, hardware failures or deliberate efforts for passive or active illegal infiltrations into the system. All these are possible threats to information privacy and system security. How serious any one of them might be, depends on the sensitivity of the information being handled, the authorization of users, the operating environment, and certainly, on the skill with which the whole system has been designed.

Because of all these growing requirements and threats against computerized information systems, there is a growing need for all possible means of resource protection and control of resource usage in such systems, known under the common name : computer privacy and security.

If a computer system is a special purpose system, then its configuration is in fact a modified configuration of a general purpose computer system. In such special purpose computer systems it is possible sometimes to establish a good protection mechanism. An example might be the system for text editing described in /1/. But the disadvantages of such special purpose systems are not concerned with the design and implementation of their protection mechanisms, but with their operational capabilities. What is needed, in fact, is a computer system without degraded operational capabilities or reduced hardware structure as a consequence of the implementation of some protection scheme. Therefore the computer

system must be a general computing facility as far as its operational capabilities are concerned. The system consists of a central processor, main memory, external memory devices, local terminals, and remote stations connected to the central processor by communication lines. The operating system is general, time sharing, multiprogramming operating system. The whole computer system must be able to operate in a larger, computer network configuration /2/.

In order to create a model of a secure computer system we give the definition (or, in fact, the list of threats which are assumed) for a described computer system. That list will contain most of the threats to the secure operations of the system one may think of ; therefore, at the same time it may be used as the list of requirements for the design of the protection scheme. Perhaps the list itself is not complete and exhaustive, but it will be accepted just for the needs of this paper. The reader may extend or modify this list if, in his opinion, it seems necessary. We assume that the following accidential or illegal activities may be dangerous to computer operations and that should be considered when designing a secure computer system /3/ :

1. Theft of recording media (magnetic tapes, disks, cards, cassettes, microfilms, etc.);

2. Illegal copying of data sets or programs;

3. Unauthorized access to a database ;

4. Software failures (errors in programs, access control, user identification procedure, etc.) ;

5. Hardware failures (protection circuits, bound registers, memory read/write protects, privileged mode, etc.) ;

6. Radiation (communication lines, the CPU, terminal connections) ;

7. Crosstalk (communication lines, device connections) ;

8. Personnel (operators, systems and application programmers, maintenance people, managers and other authorized personnel) ;

9. Terminals (illegal recordings, illegal users, bugs).

Because of all these threats to computer programs and data, extensive research has been recently developed in this area, so that computer security and information privacy are becoming very important areas of computer science. They are concerned with all aspects of computer protection : secure interactive and time sharing systems (user identification and authentication), computer networks (data communication), processing security, data base security and physical protection. The common goal of this research is to eliminate the leakage of information to the minimal possible extent and provide secure and reliable computer operations.

2. A MODEL OF A SECURE COMPUTER SYSTEM

The purpose of this paper is to design a model of a secure computer system according to the definition of such system given in the introduction, i.e. one which allows legal user to perform various activities in a secure way and the actions mentioned in the definition of the secure computer system either cannot be executed at all or their illegal execution does not reveal any protected information in the system. Such a system will be the next step in the area of computer security and data privacy, where the ultimate goal is to create a system in which illegal activities do not give intruder any significant results.

In order to introduce a model of a secure computer system consider problems of wiretapping communication lines and stealing computer media on which data are recorded (magnetic tapes, disks, cassettes, etc.). It seems that cryptography appears as the only way of data protection during their transmission and against thefts of tapes and disks. So an encoding/decoding (E/D) device will be placed at each terminal for which a user provides the secrecy key /1/. Data will be entered at the terminal in a clear form, transformed and then transmitted across the communication line. The reverse transformation will be performed by that E/D device on data coming from the central site. This provides an adequate protection from wiretappers, but someone who attacks the integrity of transmitted messages can still represent a problem. Instead of stealing messages, he can destroy them or insert false messages in the communication system. One method to determine if that has occured (without visual control of data) is to use error detection and correction schemes with already encoded data. If the error occurs that scheme will notify an operator that either data has been changed by an unauthorized user or that some

serious transmission error has occurred. This kind of protection is called end-to-end encryption /4/. Another method to protect against the same threats is so called message chaining /5/ : a part of each message is included in the next message. So the messages are chained and destruction of a legal message or insertion of a false message can be easily detected.

Data will be kept in the encoded form not only during transmission, but also in main memory or in the external files. That provides certain security at these points of a computer system. The only problem with such data is their processing. In general, data cannot be processed in the encoded form. Processing includes arithmetic and logical operations. In order to perform them, an internal decoding will be used which will apply to all informations for processing purposes. After processing, data will be encoded again and returned to main memory or to the external file. In order to perform this internal transformation, the processor should be "bracketed" by the decoding and encoding device.

In addition to a cryptographic data protection, computer system should also use a certain access control mechanism. It is used for verification of all user identifications and requests and for manipulations of secrecy keys for data processing. Each user is assigned a set of access privileges, and data are organized into so called logical segments. For each user request the access mechanism will verify, based on certain decision policies, whether the request is legal or not. It will have at its disposal the list of secrecy keys which should be also hardware implemented (as ROM or in a similar form). The access control mechanism and the key list should consist of users' identifications, users' secrecy keys, secrecy keys for all segments in the system and the list of pointers to corresponding segments. Therefore the configuration of the secure computer system is depicted in the following figure :

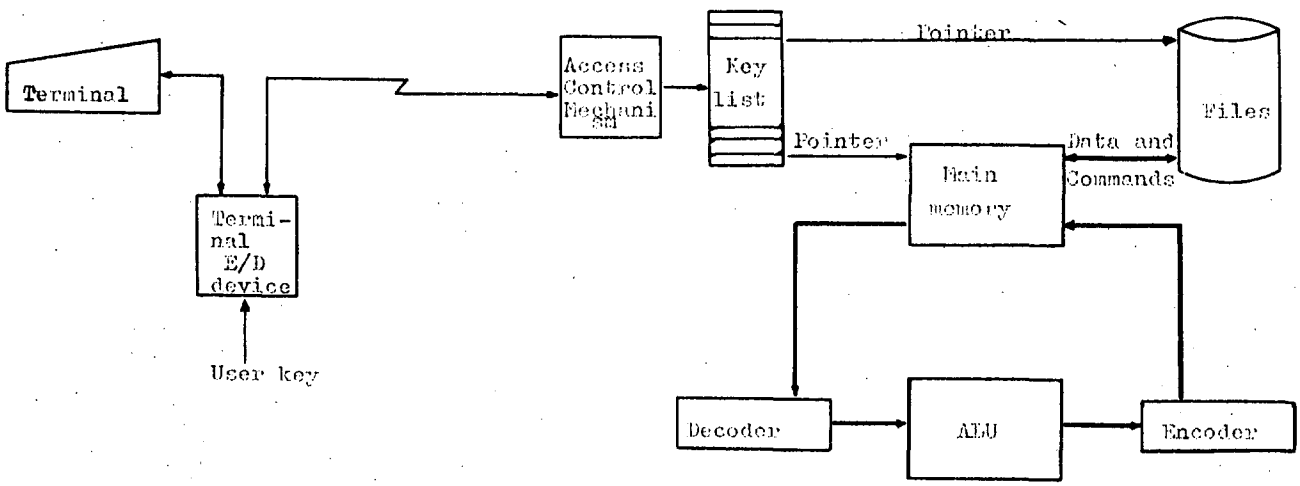


FIGURE 1. A Model of A Secure Computer System

3. DISCUSSION AND CONCLUSIONS

The model presented in the previous section is very broad and general. In fact only its main components and their interactions are indicated without discussing any details. It can be verified that this model of a secure computer system can cope with all the problems mentioned in the list of acceptable threats. However, all the necessary details must be further developed, and at this point most of them still remain as research problems. In this section some of them will be indicated in order to inform the readers of the current progress and to initialize further research in the field of secure computer systems.

On the global basis the model needs major work to include some type of a test in a form of a mathematical proof to somehow verify the design. Another type of the same test would be an actual implementation of the system. The general approach presented in this paper should be compared with similar attempts, for instance with the system described in /6/. Their system has more details, as the purpose of the authors was not just to describe a model of a secure computer system in general terms, but actually to give the design of such a system.

Secondly, when one proposes a model of a secure computer system, one is faced with a difficulty of ensuring that the implementation is successful in meeting the design specifications. Each commercially available system attempts to implement some security policies, and the provider of the system assert that the systems are secure. However, it is commonly believed that not only are these systems not secure in the sense given in the introduction of this paper, but they have not been able to implement the much weaker policies specified in the sales brochures.

The given description of the model includes only hardware components. But much of the problem is the result of difficulties in designing software even in regular computer system, let alone the problem of implementing something as complex as a secure operating system. Therefore the implementation of the model must be also concerned with software components of the computer system, and reliability of software in a computer system still remains a problem. These issues must be addressed before there can be any serious discussion of building a secure computer system.

As far as the details of the design are concerned they include hardware vs. software implementation, security trade-offs, considerations of cryptographic data protection, design of the access control mechanism, mutual suspicion problem, etc. The interesting detail of the design is the use of cryptography. Some of the interesting problems to be solved in this area are :

1. How data are encrypted/decrypted ?

2. What techniques are used for security of the communication system ?

3. How performance of the CPU is affected by encryption/decryption of instructions and data ?

4. How key lists and pointers are maintained ? How this can be done with minimum software activity ?

5. What hardware is necessary to manipulate secrecy keys reliably ?

6. Which cryptographic technique to use ?

Some of these problems have been discussed in the literature (/6/, /7/, /8/, and /9/), but there are still many open questions concerning the application of cryptography for computer security.

The relation between hardware and software implementation has several aspects. The major reason for hardware implementation of the model is efficiency. It does not provide, necessarily, reliability or flexibility. Certainly the most flexible sorts of protection are provided by user constructed software systems. Some discussions may be found in /lo/. The issue of protection against operators and system administrators is somewhat different than protection against regular users. The real issue is at what level and at what time the people with special privileges should intervene in the system. For example, illegal access may be necessary to some special system administrators in the case of creating back-up libraries, performing system measurement, and helping out users who have messed up.

Security/cost trade-offs is another problem in the design of a secure computer system. For an interesting discussion see /11/ and /12/.

The access control mechanism represents a very important component of a security mechanism. There are still many interesting problems concerned with its design and opera-

tions. First, the entries in capability lists or access lists must be clearly specified. They must reflect various user privileges (read, write, execute, modify, block, delete, etc.). One specially interesting and important problem is the mutual suspicion :

1. The executing program must have the access to the segments not accessable to the caller.

2. The executing program must not automatically gain access to the segments accessable to the caller.

A good discussion of this problem is given in /13/.

To conclude, this paper was structured in such a way that it gives first a list of needs and requirements for a secure computer system and then a broad and general model of such a system. The model includes cryptographic protection of communication lines, user identification procedure, user authentication, protection of data in main memory, on external files and during their processing. Finally, an important component of the protection scheme is the access control mechanism. It solves various problems such as user authorization, mutual suspicion, transfer of access attributes, etc.

This model is created by using already established ideas in the field of computer security. Its main purpose is to indicate the main components of the overall protection scheme in the computer system as well as their function and interactions. However, there are still many important and interesting problems to be solved. Some of them are mentioned in this part of the paper and the others can be found in the papers from the enclosed reference list.

REFERENCES

/1/ Stahl, F.
ON COMPUTATIONAL SECURITY
Dr. disseratation, Jan 1974
Report R-637, University of Illinois, Urbana, Illinois, USA

/2/ Petersen, H., Turn, R.
SYSTEM IMPLICATIONS OF INFORMATION PRIVACY
Proceedings of SJCC, 1967, USA

/3/ Ware, W.
SECURITY AND PRIVACY IN COMPUTER SYSTEMS
Proceedings of SJCC. 1967. USA.

/4/ Bartek, D.
ENCRYPTION FOR DATA SECURITY
Honeywell Comp. Jour., 8(2), 1974, USA

/5/ Feistel, H., Notz, W., Smith, J.
CRYPTOGRAPHIC TECHNIQUES FOR MACHINE TO MACHINE DATA COMMUNICATIONS
IBM Report RC 3663, Dec 1971, USA

/6/ Saltzer, J., Schroeder, M.
THE PROTECTION OF INFORMATION IN COMPUTER SYSTEMS
Proceedings of IEEE, 63(9), Sept 1975

/7/ Saltzer, J.
PROTECTION AND CONTROL OF INFORMATION SHARING IN MULTICS
CACM, 17(7), July 1974, USA

/8/ Muftić, S.
DESIGN AND OPERATIONS OF A SECURE COMPUTER SYSTEM
Dr. dissertation, May 1976
The Ohio State University, Columbus, USA

/9/ Muftić, S.
TEORIJA SIGURNOSTI KOMPJUTERSKIH SISTEMA
In preparation,
Zavod za ekonomsko planiranje Sarajevo

/1o/ Jones, A.
PROTECTION IN PROGRAMMED SYSTEMS
Dr. dissertation, Carnegie-Mellon Univ., Pittsburgh, USA

/11/ Rotenberg, S.
MAKING COMPUTERS KEEP SECRETS
Dr. dissertation, MIT, TR-116, Sept 1973

/12/ Myers, E.
COMPUTER SECURITY : EACH CASE IS DIFFERENT
Datamation, April 1975, USA

/13/ Schroeder, M., Saltzer, J.
A HARDWARE ARCHITECTURE FOR IMPLEMENTING PROTECTION RINGS
CACM, 15(3), March 1972, USA

reševanje problemov računalniškega kadra in kadra informatike v sr sloveniji

jernej virant

UDK 681.3-05(497.12)

Fakulteta za elektrotehniko, Ljubljana

V sestavku nahajamo pregled prizadevanj za reševanje problemov, ki tarejo računalniški kader in kader informatike v SR Sloveniji.

COMPUTER AND INFORMATICS PERSONNEL: SOLUTION OF PROBLEMS IN SR SLOVENIA

This report presents survey of work done towards the solution of problems, concerning computer and informatics personnel in SR Slovenia.

1. ZAKAJ SO POTREBNE SISTEMSKE REŠITVE?

V tem času se v SR Sloveniji približujemo 200 računalnikom (veliki, srednji, mali in birojski z diskovno usmerjenostjo) v OZD, predvsem v SIS pa se bližamo 30 registriranim računalniško usmerjenim informacijskim sistemom. Eno in drugo predstavlja za republiko našega obsega precejšnjo investicijo, obratovalne stroške in preobremenitev kadra. Medtem ko lahko računalniško opremo kupimo, si jo sposodimo, zamenjamo itd., kaj takega pri kadrih ne moremo storiti. Omenjeni računalniki in informacijski sistemi že danes zahtevajo 1% vseh zaposlenih v SR Sloveniji, kar kaže na nemogočo situacijo saj gospodarstvo in družba upravičeno zahtevata kadre za druga tudi pomembna področja dela. Investicije v računalništvo v SR Sloveniji so že tolikšne, da je potrebna dokaj natančna analiza ali nam računalniška dejavnost vrača vložena sredstva ali ne.

Računalništvo in informatika sta stroki, ki se razvijata s to posebnostjo, da sta potrebni vsem ostalim strokam in se temu primerno tudi redno izobražuje na visokih, višjih in srednjih šolah. Ravno zaradi velike širine je možno, da izobraževalni zavodi niso med seboj uravnani niti po kvaliteti in niti po tem, da bi bili vsi poklici in vsa delovna mesta primerno izobraževalno zajeta. Problem bi bil takoj nekoliko lažji, če bi imeli vsaj dovolj dobro razrešen kompleks delovna mesta - poklici - nomenklatura poklicev. S tem pa bi bilo rešenih tudi mnogo drugih problemov na področju računalništva in informatike.

Omenili smo že, da je računalniški kader v SR Sloveniji preobremenjen. Narava dela na področju računalništva, še neprimerno bolj pa preobremenjenost pri takšnem delu, vodita v 6% fluktuacijo in beg na delovna mesta, ki glede na obremenjenost niso kritična. Tako imamo v gospodarstvu poprečni staž za programerja/koderja samo 2 leti, medtem ko velja za vodjo RC že 7 let.

V našem gospodarstvu je 42% delovnega časa programerjev in organizatorjev potrošeno za vzdrževanje obstoječega stanja AOP. Zadnje zelo povečuje potrebo po dodatnem številu omenjenega kadra oziroma večji avtomatizaciji in mehanizaciji ažuriranja.

V SR Sloveniji imamo danes resda skoraj 25% vsega jugoslovanskega računalniškega kadra, vendar je njegova struktura slaba. Z največjim inkrementom narašča kader z osnovno šolsko izobrazbo, srednješolsko izobražen kader pa seže predaleč na delovna mesta, na katerih je potrebna kreativnost, analitični pristop, poznavanje metodologije itd., kar srednješolsko izobražen kader praviloma ne obvlada. Zadnji čas je, da se posodobi pripravo podatkov in rezultatov (na ustreznih mestih je zaposlen kader z osnovno šolsko izobrazbo) ter odpre možnost, da se srednješolsko izobražen kader dovolj izobrazi, VDO pa povečajo usmerjenost v računalništvo in informatiko.

Čim bolj je računalniški sistem poraz[illegible]en, obsežen, učinkovit itd., tem manj je pričakovati zadostno pomoč s strani ponudnika računalniške opreme. Ta je pogodbeno obvezan, da je oprema v stanju obratovanja, ni pa obvezan da je za družbo in gospodarstvo dovolj učinkovita.

Naj ne bi več naštevali problemov, ki so našim strokovnjakom znani. Naj povemo le še to, da so nas v sistemsko reševanje kadrovskih problemov pahnile tudi strokovne ocene kvantitete kadra. Trenutno imamo v SR Sloveniji zaposlenih v računalništvu in informatiki približno 3800 oseb, v l. 1976 pa bi potrebovali 4400-6900 ter v l. 1980 9000-12600 oseb [8]. Kako priti do tega kadra, če so velike potrebe tudi na drugih področjih dela?

2. DOSEDANJA PRIZADEVANJA ZA REŠEVANJE PROBLEMOV

Do sedaj je resnejše tekla beseda o splošnih problemih računalniškega kadra v Komisiji za vprašanja računalništva in informatike, IS SRS [1], v Komisiji za obdelavo podatkov GZ, SRS [2, 3], v študiji [4], v okviru Strokovnega sveta za računalništvo na Univerzi v

Ljubljani [5], ob oblikovanju pogodbe CDI, ob oblikovanju projekta za uvajanje računalništva v srednje šole [6], na raznih simpozijih, seminarjih in v več člankih ter v okviru študijskega programa Katedre za računalništvo in informatiko, FE. Največ dela pa je bilo dosedaj opravljenega v okviru delovne skupine za pripravo republiškega programa izobraževanja kadrov za potrebe računalništva in informatike, ki deluje pri Republiškem komiteju za družbeno planiranje in informacijski sistem IS SRS že od l. 1975 naprej. Pravzaprav je bila delovna skupina imenovana že s strani Komisije za vprašanje računalništva in informatike pri IS SRS hkrati z drugimi delovnimi skupinami, ki pa po večini niso začele s prepotrebnim sistematičnim odpravljanjem problemov v našem računalništvu in informatiki. V manj kot dveh letih je imela imenovana delovna skupina 29 delovnih sestankov in en dvodneven seminar v Škofji loki.

Delovna skupina, ki jo sestavljajo strokovnjaki iz izobraževalnih bazenov Ljubljana, Maribor in Kranj, je svoje delo razdelila na 4 dele:

1) delovna mesta, poklici, profili poklicev, nomenklatura poklicev,
2) kurikulum SR Slovenije in pokritje tega z našim rednim izobraževalnim sistemom,
3) dodatno firmno odvisno in firmno neodvisno izobraževanje na nivoju republike in
4) "računalniška kultura" v SR Sloveniji.

Do sedaj je izdelala Prvo delovno poročilo [7], in ga dala v obravnavo pomembnejšim računskim centrom, izobraževalnim enotam ter Komiteju za družbeno planiranje in informacijski sistem IS SRS z namenom, da dobi sugestije za nadaljnje delo. V tem poročilu je obdelana točka 1), pripravljeni pa so tudi elementi za računalniške obdelave, ki nas lahko privedejo brez subjektivnih vplivov do točke 2). Poročilo obravnava:

- relacije med znanji neke stroke, znanji informatike in znanji računalništva,
- opredelitev potrebnih znanj računalništva in informatike tako po številu kot po globini,
- definicije delovnih mest,
- definicije poklicev,
- presek delovnih mest s poklici,
- določanje profilov poklicev ter
- oceno potreb računalniškega kadra in kadra informatike za SR Slovenijo.

Popoln spekter delovnih mest se nanaša na 26 delovnih mest, ki do določene mere kažejo tudi na nadaljnji razvoj računalništva in informatike (na primer: analitik informacijskega sistema, vodja računalniške mreže, firmno neodvisen svetovalec, organizator zbirke podatkov itd.). Teh 26 delovnih mest, ki še niso gradirani (na primer: mlajši programer, starejši programer) pokriva 11 poklicev, ki jih naj bi dal naš redni izobraževalni sistem. Ta je v poročilu predviden kot harmonična celota enega izvora čistih računalnikarjev in informatikov, 14 izvorov x - informatikov (x = medicina, poslovnost, tehnika, družboslovje itd.) ter 7 tipov izvora kadra na nivoju srednje izobrazbe in manj. K omenjenim izvorom se dodaja še nefirmno in firmno (dodatno) izobraževanje.

Definicije delovnih mest, poklicev in profilov poklicev se nanašajo na 125 različnih znanj računalništva in informatike, na 9 temeljnih in na 34 aplikativnih znanj. Vse gradivo je osnovnega pomena, da s strani Skupnosti za zaposlovanje SR Slovenije pristopijo k nomenklaturi poklicev, ki naj bi dosegla kvaliteto nomenklatur drugih, bolj klasičnih panog. Šifrant poklicev sicer že sedaj vsebuje nekaj poklicev iz računalništva, vendar so se ti pojavili prej, preden sta se računalništvo in informatika pojavila kot samostojno področje dela.

3. PREDVIDENO NADALJEVANJE REŠEVANJA PROBLEMOV

V kolikor bo prvo poročilo [7] ugodno sprejeto bo delovna skupina nadaljevala z delom v okviru naštetih točk 2), 3) in 4) po že začrtani metodologiji. Drugo poročilo bi vsebovalo vsaj predlog kurikuluma SR Slovenije, ki bi upošteval naše specifične razmere in zadnjo metodologijo v strokovni literaturi. Podan bi bil tudi postopek, ki bi na najenostavnejši način vpeljal omenjeni izobraževalni koncept v prakso.

Tretje poročilo, ki ga namerava izdelati delovna skupina, bi zajelo dopolnilno izobraževanje, ki ga firme ne dajo več, je pa nujno potrebno ter, kaj storiti v SR Sloveniji, da se dvigne splošna "računalniška kultura" in odnos do računalništva in informatike na sploh. Morda bo problematika za dopolnilno izobraževanje tako široka, da bi omenjeno poročilo morali deliti v dva dela.

V okviru pogodbe CDI (Control Data Institute) je Fakulteta za elektrotehniko v Ljubljani prišla do zadostne izkušenosti, da se z njeno pomočjo organizira nefirmno dopolnilno izobraževanje na nivoju republike. Že v tem času je omenjena VDO prešla iz CDI na DIR (dopolnilno izobraževanje iz računalništva), kar je potrebno podpreti s strani raznih republiških organov, da to izobraževanje ali preusmerjeno takšno izobraževanje v resnici odpravlja probleme na nivoju republike, ki nastajajo v zvezi z ažuriranjem znanja, potrebnim dnevnim znanjem, podajanjem rešitev uspešnih domačih informacijskih sistemov itd. Vsekakor pa firmnim šolam ostane izobraževanje, ki se nanaša na firmne posebnosti aparaturne opreme, programske opreme, firmne posebnosti pri postavljanju informacijskih sistemov, vzdrževanju itd., kar zanesljivo ne bo predmet nefirmnega dopolnilnega izobraževanja.

Da je potrebno imeti v življenju in pri delu primeren odnos do računalništva in informatike, nam je vsem jasno. Kako to doseči, pa ni tako jasno, vsekakor pa so zametki reševanja že prisotni [6]. Delovni človek oz. občan se bo znašel pred republiškimi, občinskimi bankami podatkov, pred bazami podatkov OZD itd., kar kaže, da bo obdan vedno bolj z računalniškimi podatki in obdelavami (register prebivalcev, register vozil, register vojnih obveznikov, plačilo vodarine, plačilo elektrike itd.). Zadnje zahteva posebno informiranost o podatku, o strogosti njegovega zapisa itd. že v okviru obveznega izobraževanja. Tovrstna problematika sodi bolj v okvir raziskav naših pedagogov in sociologov in zato pričakujemo primerne rešitve tudi z njihove strani.

V okviru elaborata [9] nahajamo 12 smernic, ki naj bi prišle do izraza pri nadaljnjem kreiranju kadrovske politike na področju računalništva in informatike. Področna raziskovalna skupnost za avtomatiko, računalništvo in informatiko, RSS, se je odločila, da bo kadrovski problematiki posvetila precej več pozornosti kot v preteklosti, kar se naj bi odrazilo tudi v bodočem programu področja.

4. ZAKLJUČEK

S tem sestavkom nismo želeli podati konkretne delovne rezultate, temveč smo želeli računalniško javnost le informirati o akcijah in prizadevanjih za zboljšanje stanja računalniškega kadra in kadra informatike v SR Sloveniji.

Literatura

[1] J.Virant, Izobraževanje kadrov za področje računalništva in informatike ter vpeljava računalništva in informatike v splošni izobraževalni sistem. Poročilo za Komisijo za vprašanja računalništva in informatike pri IS SRS, Ljubljana 1972.

[2] M.Bradaška, H.Bajda, J.Virant, Nomenklatura poklicev v računalništvu. Poročilo za Komisijo za obdelavo podatkov GZ SRS, Ljubljana 1972.

[3] M.Bradaška, Anketa 50 RC v DO v Sloveniji. Komisija za obdelavo podatkov GZ SRS, Ljubljana 1972.

[4] J.Virant, Opredelitev potreb računalniških kadrov v SR Sloveniji. RSS, Ljubljana 1974.

[5] Bilten za računalništvo, Univerza v Ljubljani, št. 6, leto III, februar 1973.

[6] Projekt, Pouk računalništva v usmerjenem izobraževanju, RCPU-FNT, RSS, Ljubljana 1975.

[7] Delovna skupina za pripravo rep. programa izobraževanja kadrov za potrebe računalništva in informatike, Prvo delovno poročilo. Komite za družbeno planiranje in informacijski sistem IS SR Slovenije, Ljubljana, december 1976.

[8] J.Virant in ostali, Prispevek k republiškemu načrtu izobraževanja računalniških kadrov in kadrov informatike. Zaključno poročilo naloge na RSS, FE, december 1976.

[9] Delovna grupa, Smerniki za investiranje v računalništvo v SR Sloveniji, naloga LB, RSS, GZ SRS, v zaključni fazi dela.

komunikacija med sekvenčnimi procesi - pregled del I.

m. exel

Institut "Jožef Stefan", Ljubljana

UDK 681.3.01

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

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

1. UVOD

Pričujoči pregled obravnava nekatere probleme ki jih zastavlja organizacija medprocesnih komunikacij.

Za začetek bomo poskušali razčistiti nekatere pojme, ki se pojavljajo v zvezi z medprocesno komunikacijo.

Predpostavimo, da nam je intuitivno jasno, kaj je nek sekvenčen proces: to je določeno zaporedje elementarnih dejanj, kjer je dejanje izvedba (eksekucija) neke elementarne inštrukcije ali stavka. Ne bomo se nadalje spuščali v razpravo, kaj pomeni "elementarnost" v tej definiciji; naj tudi ne bo važno kdaj se dejanja nekega sekvenčnega procesa v času odvijajo. Med dvema zaporednima dejanjima lahko preteče nek (realen) čas $t > 0$; važno je le, da se dejanje, ki v danem zaporedju sledi dejanju d, odvija (v celoti) kasneje kot d.

Sedaj lahko definiramo proces kot nek skupek sekvenčnih procesov, ki so medsebojno logično povezani tako, da nam proces, ki ga sestavljajo predstavlja neko zaključeno, logično celoto. Kakršenkoli program v nekem programskem jeziku lahko definira (statično) nek proces. Proces je dinamičen pojem; v primeru programa je to vsaka izvedba tega programa; komponentni sekvenčni procesi, ki dan proces (izvedbo programa) sestavljajo se lahko v tem procesu večkrat izvedejo (pojem zanke) ali pa se lahko izvajajo tudi sočasno, če programski jezik to dovoljuje (prisotnost konstruktov za paralelnost stavčnih izvedb). Čeprav program predstavlja s svojimi različnimi možnimi izvedbami več procesov, pa so slednji sestavljeni iz istega skupka sekvenčnih procesov; vsak sekvenčni proces skupka lahko namreč enolično vežemo na nek "elementarni blok" programa, kjer "elementarni blok" pomeni vsako maksimalno zaporedje programskih inštrukcij (na nivoju osnovnih programskih konstruktov jezika), zaporedje, ki se vedno izvaja sekvenčno. Procesi, ki odgovarjajo vsem možnim izvedbam danega programa bodo seveda imeli isto predstavitev; ta predstavitev bo neka struktura sekvenčnih procesov, katerih skupek je enolično določen z danim programom. Tak strukturiran skupek sekvenčnih procesov predstavlja torej nek razred procesov (, ki lahko vsebuje enega ali več članov).

Odslej naprej bomo pod pojmom proces podrazumevali nek razred procesov. Odvijanje procesa p bo torej pomenilo odvijanje nekega procesa iz procesnega razreda p.

Predpostavimo nek koneksen računalniški sistem (računalnik ali več povezanih računalnikov plus vsa priključena periferija). Na tem sistemu se odvijajo razni procesi: nekateri so permanentni (se stalno odvijajo) in ustrezajo predvsem sistemskim programom, nekateri so začasni in ustrezajo predvsem uporabniškim programom. Med temi procesi obstajajo ali pa ne določene zveze v času in/ali v prostoru. Glede na vrsto zvez lahko govorimo o treh vrstah procesov:

- neodvisni procesi so logično neodvisni in niso sočasni;
- paralelni procesi (glej 1) so logično neodvisni (delno) sočasni procesi brez prostorskih (hardverskih) konfliktov (tj. prostorsko neodvisni);
- komunicirajoči ali odvisni procesi so po Wegnerju (2) procesi, ki lahko spreminjajo informacije, ki so jim skupne.

Odvisne procese bomo nadalje razdelili v:

- medlo odvisne procese ("loosely connected"), glej Dijkstra(3)in
- sodelujoče procese (kooperacija).

Medlo odvisni procesi so logično neodvisni; odvisnost se izraža v dejstvu, da si morajo deliti različne računalniške vire. Pojavlja se torej potreba sinhronizacije medlo odvis-

nih procesov in eventuelno, glede na dan vir, potreba vzajemne izključenosti ("mutual exclusion",3) uporabe tega vira ali, bolje rečeno, izključenosti sočasne uporabe tega vira.

Tipičen primer te vrste procesov je skupek programov, ki se odvijajo na nekem multiprogramiranem računalniškem sistemu. Pri tem ni bistveno, če je ta sistem tudi multiprocesorski; če je več procesorjev, se s tem pač samo poveča število virov, ki so intervali časa centralnih računalniških enot. V zgornjem primeru skrbi navadno za sinhronizacijo in izključenost sočasne uporabe virov takozvani skedjuler.

S pojmom izključenosti sočasne uporabe virov se pojavi tudi pojem kritičnega dela nekega procesa ("critical section", 3). To je tisti delček nekega procesa, ki ob uporabi določenega vira spreminja informacije (oziroma jih dosega), ki so lahko skupne več procesom (klasičen primer: pisanje v pomnilnik).

Pojem vira zgoraj nismo točno definirali; naj omenimo, da za določene vire - primerno definirane - izključenost sočasne uporabe ni nujna: npr., če za vir vzamemo del pomnilnika, ki ga skupek procesov vedno lahko samo čita.

Preidimo sedaj na sodelujoče procese: to so odvisni procesi, ki so tudi medsebojno logično povezani.

Tipičen primer je par simetričnih procesov tipa proizvajalec-uporabnik. Proizvajalec naprimer ustvarja v nekem buferju slike kartic, uporabnik pa jih jemlje iz buferja in jih luknja. Očitno je, da tako v proizvajalcu kot v uporabniku obstaja kritični del: to je tisti del v vsakem od obeh procesov, ki manipulira bufer in s tem spreminja informacijo skupno obema procesoma (; bufer-komunikacijsko "polje" - torej tukaj izpade kot "logičen" vir, ki si ga delita oba procesa). Bistvo pravilnega sodelovanja je v izključenosti sočasne uporabe buferja; ali nekritični deli obeh procesov potekajo sočasno ali ne pri tem ni bistveno. Ni nujno, da so sodelujoči procesi simetrični. Nek proces lahko logično obravnavamo kot pomožen proces nekega glavnega procesa. Klasičen primer je naslednji nesimetričen par procesov: glavni proces se odvija na centralni računalniški enoti, pomožni proces pa se sočasno odvija na vhodno-izhodnem kanalu in izvaja vhodno-izhodne operacije za glavni proces. V tem primeru želimo izkoristiti možnost sočasnega odvijanja obeh procesov. Glavni proces sproži pomožni proces in kasneje ali čaka - če je potrebno - na zaključek pomožnega procesa ali pa je o njegovem zaključku obveščen.

Zgornji način logičnega obravnavanja procesov (glavni proces, pomožni proces) je bil zadnje čase kritiziran (glej 4). Priporoča se programska uporaba nekega specijalno definiranega kontrolnega procesa, ki naj ureja sinhronizacijo glavnega in pomožnega procesa, ki tako s stališča kontrolnega procesa izpadeta kot medsebojno enakovredna sodelujoča procesa. V zgornjem primeru (kanalni proces kot pomožni proces) se v klasičnih večjih računalniških sistemih imenuje ta kontrolni proces supervizor. Videli bomo, da se v novejših pristopih tak kontrolni proces definira kot monitor (glej 5).

V nadaljevanju se bomo predvsem zanimali za odvisne procese. Ker smo definirali proces kot skupek sekvenčnih procesov lahko torej zožimo problem medprocesne komunikacije na problem komunikacije med sekvenčnimi procesi.

Odslej naprej bomo torej pod pojmom proces podrazumevali nek sekvenčen proces.

2. MEDPROCESNA KOMUNIKACIJA

Komunikacija med procesi nastopa v različnih kontekstih:

- v organizaciji in strukturiranju operacijskih sistemov,
- v aplikacijah v realnem času,
- v simulacijskih študijah,
- v kombinatoričnem in hevrističnem programiranju.

Opis in implementacija medprocesne komunikacije se rešujeta na različne načine, glede na vrsto aplikacije:

- v operacijskih sistemih se za opis in implementacijo uporabljajo npr. kanalne komande, posebne inštrukcije za testiranje in otipavanje raznih indikatorjev; komunikacija pa se realizira s pomočjo prekinitev, prioritet in s pomočjo supervizorskega sistema (glej 4);
- v simulacijskih jezikih se pojavlja pojem dogodka ("event") in z njim v zvezi inštrukcije za aktiviranje, zadrževanje, prekinitev dogodkov in za čakanje na dogodek;
- v nekaterih splošno-problemskih programskih jezikih (denimo PL/1) se pojavlja pojem dejavnosti ali opravila ("task") in inštrukcije za aktiviranje opravil in za čakanje na zaključek opravil.

Glede na raznovrstnost aplikacij in na raznolikost opisov in implementacij medprocesne komunikacije se danes pojavlja potreba bolj teoretičnega in bolj enotnega pristopa k problemom medprocesne komunikacije (glej 1). Eden od pobudnikov takega pristopa je bil nedvomno Dijkstra s svojim člankom o kooperaciji sekvenčnih procesov (3). Potrebo po novem pristopu na področju snovanja in razvoja operacijskih sistemov je povdaril predvsem Brinch Hansen (4); pokazal je, da so sedanji pristopi nejasni in nezanesljivi.

Naj omenimo, da se v literaturi problematika medprocesne komunikacije pojavlja v različnih kontekstih, glede na aspekte, ki jih v tej problematiki želimo povdariti. Često je naprimer povdarek na možnosti sočasnega odvijanja procesov; od tod zanimanje za problem kontrole odvijanja različnih procesov in od tod pojem teorije paralelnega programiranja (glej 1, 6, 8) in pojem modelov za paralelno izračunavanje (7). V pričujočem pregledu smo zavzeli stališče, da vsaka teorija medprocesne komunikacije vsebuje tudi teorijo paralelnega programiranja.

V literaturi se trenutno problem medprocesne komunikacije obravnava nekako na dveh nivojih:

- v snovanju visokega programskega jezika za paralelno programiranje in v iskanju osnovnih primitivnih konstruktov za opisovanje sinhronizacije procesov;
- v preučevanju in snovanju raznih modelov paralelnega izračunavanja (glej 7).

V nadaljevanju se bomo zanimali predvsem za programski nivo.

3. PROGRAMIRANJE MEDPROCESNE KOMUNIKACIJE

Po Hoare-ju (1) in Dijkstri (9) bomo povzeli zastavljene cilje visokega programskega jezika za paralelno programiranje in medprocesno komunikacijo.

Tak jezik naj bi imel naslednje lastnosti:

- vseboval naj bi programske konstrukte za opisovanje sočasnosti procesov, za opisovanje sinhronizacije procesov in za opisovanje medprocesne komunikacije;
- vseboval naj bi enostavne, jasne in pregledne koncepte, primerne za strukturirano in hierarhično programiranje;
- dopuščal naj bi možnost enostavnega in metodičnega dokazovanja lastnosti programov, napisanih v tem jeziku (lastnosti, ki nas zanimajo so predvsem korektnost, odsotnost mrtve točke - "deadlock"a- in končnost -"termination");
- omogočal naj bi dovolj učinkovito implementacijo.

Kar se tiče snovanja takega programskega jezika smo v literaturi zasledili nekako dva pristopa:

- prvi pristop, katerega glavni pobudniki so Dijkstra, Hoare in Brinch Hansen se vključuje v metode strukturiranega programiranja in uvaja ter razvija naslednje pojme:
  - pojem binarnega in splošnega semaforja,
  - pojem (pogojnega) kritičnega dela procesa,
  - pojem monitorja ali tajnika in
  - pojem hierarhije sekvenčnih procesov in monitorjev;
- drugi pristop opisujeta Campbell in Habermann (10, 11); glavna razlika glede na prvi pristop je predvsem v odsotnosti hierarhije procesov in v baziranju opisa sinhronizacije procesov na opisu možnih zaporedij izvajanj sekvenčnih procesov.

Predno preidemo na opis teh dveh novejših pristopov si bomo zelo na kratko ogledali nekatere klasične programske konstrukte oz. posebne inštrukcije, ki se danes uporabljajo v zvezi z medprocesno sinhronizacijo.

Področje sistemskega programiranja:

sodelovanje med kanalom oz. procesom na kanalu in centralno računalniško enoto oz. procesom na njej se odvija s pomočjo (2):

- komande za aktiviranje kanala:

  primer: WDBA Y : piši na disk v binarni obliki preko kanala A; pri tem uporabi informacijo (npr. program za kanal), ki se začenja na naslovu Y.

Ta komanda je primer inštrukcije tipa razcep ("fork"): ustvari in spočne namreč nek sočasen proces (na kanalu A);

- inštrukcije za testiranje kanala:

  primer: TCA * : ponavljaj inštrukcijo vse dokler se proces na kanalu A odvija, potem pa nadaljuj sekvenčno.

Ta inštrukcija ("busy waiting" ali aktivno čakanje) je primer inštrukcije tipa združitev ali zlitje ("join"): "končuje" namreč nek sočasen proces (na kanalu);

Področje višjih splošnih programskih jezikov:

"multitasking" v Pl/1. Prevedimo "task" s procesom; "multitasking" torej omogoča, da neki glavni proces lahko definira in sproži nek podproces ("subtask"), ki se lahko odvija sočasno z glavnim procesom. Pregled inštrukcij:

- inštrukcija tipa razcep:

  primer: CALL PROCA TASK(T3) EVENT(ET3) PRIORITY(-2);

  sprožen je sočasni podproces imenovan T3, ki je izvedba - sočasna s procesom, ki izvaja CALL- procedure PROCA; dogodek ET3 je vezan na zaključek procesa T3 (ob zaključku procesa T3 je spremenljivka ET3 postavljena na binarno vrednost "1"B); prioriteta procesa T3 je za 2 manjša od prioritete glavnega procesa;

- inštrukcija tipa združitev:

  primer: WAIT (ET2, ET3) (1); proces, ki izvaja WAIT čaka na zaključek enega od dogodkov ET2 in ET3. Dogodki so lahko vezani na zaključek naslednjih procesov: na izvedbe procedur in na izvedbe vhodno-izhodnih inštrukcij READ, WRITE, DELETE in DISPLAY;

- inštrukcija za zaustavitev dejavnosti:

  primer: DELAY(10); proces, ki izvaja DELAY je zaustavljen za 10 milisekund.

Področje simulacijskih jezikov:

komande za skedjuliranje ali razvrščanje "aktivnosti" v jeziku SIMULA (glej 2):

- activate aktivnost čas-aktiviranja; tipa razcep (podobna komanda v jeziku SIMSCRIPT je CAUSE);
- reactivate aktivnost čas-reaktiviranja;
- reactivate tekoča-aktivnost delay t: tekočo aktivnost zaustavi za čas t ;
- cancel(A); terminate(A); ustavi aktivnost A;
- wait(S) : zaustavi tekočo aktivnost in jo postavi v vrsto S.

Naj za konec omenimo, da so bile inštrukcije tipa razcep in združitev zadnji čas kritizirane s stališča jasnega in zanesljivega sistemskega programiranja (glej 4).

4. OD SEMAFORJA DO HIERARHIJE MONITORJEV

4.1. Pojem semaforja in uporaba

Za sinhronizacijo in komunikacijo med procesi je Dijkstra predlagal (glej 3) pojem semaforja. Semafor je posebna cela spremenljivka, nad katero so možne samo naslednje operacije: operacija P ali čakanje, operacija V ali signaliziranje in inicijalizacija.

Razlikujemo binarne in splošne semaforje, glede na uporabo pa govorimo tudi o privatnih semaforjih.

Binarni semafor ima lahko samo dve vrednosti: 0 in 1, splošni semafor pa ima lahko katerokoli celo vrednost.

Oglejmo si operaciji P in V. Ti dve operaciji morata biti, po definiciji, implementirani tako, da jih nikakor ni možno prekiniti.

Sprva (glej 13) sta bili ti dve operaciji definirani tako, da je njun pomen bil naslednji:

P`(sem): sem: = sem-1; if sem<0 then com proces izvajajoč P je prekinjen in postavljen v čakajočo vrsto vezano na semafor sem com; skip;

V`(sem): sem: = sem+1; if sem<0 then com aktiviraj, če obstaja, enega izmed procesov v čakajoči vrsti vezani na semafor sem com;

Zahteva neprekinljivosti za operacijo P`(sem) pomeni naslednje: v danem trenutku je možno, da je več procesov "načelo" operacijo P`(sem)

(sem je torej v tem pomenu lahko imel tudi negativne vrednosti) vendar, v trenutku, ko se vrednost sem-a poveča za 1 (po izvedbi neke V`(sem) operacije), se lahko "zaključi" (zaključitev pomeni izvedbo prazne inštrukcije skip) le ena izmed načetih P`(sem) operacij.

Kasneje je bila semantika obeh operacij redefinirana takole:

P(sem):await sem>0 then sem:=sem-1;
V(sem):await true then sem:=sem+1;

Naj pri tem ne bo bistveno, kako sta obe operaciji dejansko implementirani. V nadaljevanju bomo uporabljali ta drugi pomen obeh operacij. Naj še omenimo, da je v tem pomenu vrednost semaforja lahko le nenegativna.

Kaže, da je bil Algol 68 (12) prvi jezik, ki je vključeval semaforje ; semafor je v Algolu 68 struktura

struct sema = (ref integer rezervirano-ime-F);

Operacije pa so

op / = (integer a) sema: c za inicijalizacijo na vrednost a c;

op ↓ = (sema s): c operacija P(s) c;

op ↑ = (sema s): c operacija V(s) c;

4.1.1. Binarni semafor

Binarni semafor se tipično uporablja za rešitev problema kritičnih delov procesov oz. izključenosti sočasne uporabe virov ("mutual exclusion problem").

Tipičen primer (glej 3): konstrukcija N cikličnih procesov, vsak vsebujoč kritičen del, katerega izvedba je dovoljena v danem časovnem intervalu samo enemu izmed N procesov se rešuje trivijalno z uporabo enega binarnega semaforja sem, katerega pomen bo naslednji:

sem=1 pomeni, da se nobeden od N procesov ne nahaja v svojem kritičnem delu;

sem=0 pomeni, da se eden izmed N procesov nahaja v svojem kritičnem delu.

Shema rešitve:

```
begin int sem; sem:=1;
 parbegin proces1: begin ... end;
          ...
          ...
          procesN: begin ... end;
 parend
end;
```

Konstrukt parbegin p1; p2;...pN parend označuje sočasno izvedbo stavkov p1, p2,,,pN.

Shema procesa i:

```
procesi:begin Li: P(sem); kritični del i;
              V(sem); ostanek cikla procesa
              i; goto Li
        end.
```

4.1.2. Splošni semafor

Povzeli bomo primer iz uvoda: par cikličnih procesov proizvajalec-uporabnik komunicira preko skupne informacije, ki bodi nek neomejen bufer (za enostavnost primera). Elementi buferja so slike kartic. Primer rešimo z uporabo splošnega semaforja štel, ki skrbi za pravilno sosledje obeh procesov in obenem za štetje elementov v buferju ter z uporabo binarnega semaforja bin, ki skrbi za sočasno izključenost uporabe skupnega vira - buferja.

Shema rešitve (glej 3):

```
begin integer štel, bin; štel:=0; bin:=1;
 parbegin
  proizvajalec:
     begin zopet1: proizvede naslednjo sliko
                   kartice:
                   P(bin);
                   doda ta element buferju;
                   V(bin);
                   V(štel);
                   goto zopet1;
     end.
  uporabnik:
     begin zopet2: P(štel);
                   P(bin);
                   vzame element iz buferja;
                   V(bin);
                   uporabi vzeti element;
                   goto zopet2;
     end.
 parend
end.
```

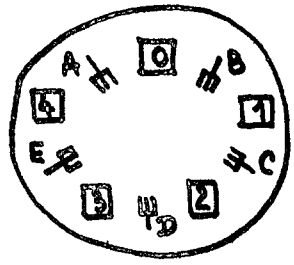
Dijkstra je pokazal (3), da splošni semafor v bistvu ni potreben; zgornji primer lahko rešimo s pomočjo dveh binarnih semaforjev in z uporabo spremenljivke število-elementov-buferja, ki si jo delita ("shared") oba procesa. V bistvu gre torej za to, da informaciji bufer, ki si jo delita oba procesa - ki je skupni "logičen" vir (glej uvod) obeh procesov - dodamo še informacijo število-elementov-buferja.

4.1.3. Privatni semafor in spremenljivka stanja

S pojmom privatnega semaforja se bomo seznanili v primeru petih filozofov (glej 9): dejavnost vsakega filozofa bodi opisana z naslednjim procesom:

```
cycle begin misliti;
           jesti
      end.
```

Sodelovanje vseh petih procesov se izraža v dejstvu, da vseh pet filozofov jé ob isti okrogli mizi in pri tem mora vsak uporabiti, kadar jé, dve vilici obenem. Vilice in filozofi so razporejeni takole:



Problem je v tem, da dva soseda ne moreta jesti obenem. Pri rešitvi moramo paziti tudi na možnost mrtve točke ("deadlock"): če bi vsi filozofi naenkrat pograbili vsak svojo levo vilico, na primer, ne bi mogel jesti nihče. Procesi so "zablokirani".
Dijkstra (9) predlaga rešitev, v kateri uporablja naslednje elemente:

- izključitveni binarni semafor bin z začetno vrednostjo 1;
- tabelo petih celih spremenljivk integer arrayC(0:4), ki predstavljajo stanje procesov:

  C(i)=0 pomeni: filozof i misli,
  C(i)=1 pomeni: filozof i je lačen,
  C(i)=2 pomeni: filozof i je;

začetna vrednost C-ja je 0;

- tabelo petih privatnih semaforjev priv(0:4) z začetno vrednostjo 0;
- proceduro

```
procedure test (integer value k);
if C ((k-1) mod 5) ≠ 2
   and C(k)=1
   and C((k+1)mod 5) ≠ 2
then begin C(k): = 2;
           V(priv(k))

     end.
```

S temi elementi rešitev za filozofa f izgleda takole:

```
cycle begin misliti;
          P(bin);
     krl: C(f):=1; test(f);
          V(bin);
     kr2: P(priv(f));jesti;
          P(bin);
     kr3: C(f):=0; test((f+1) mod 5);
                   test((f-1) mod 5);
          V(bin);

     end.
```

Poskušajmo analizirati zgornji primer s stališča skupnih virov: očitno je namreč, da je ta primer bolj kompliciran zaradi tega, ker ne moremo vseh 5 vilic identificirati kot en sam vir, skupen vsem petim procesom. Tukaj gre za pet različnih virov (vilic) od katerih je vsak skupen samo dvema procesoma. To je imelo za posledico uvedbo spremenljivk stanj C in semaforjev priv. S tem smo ustvarili nov logičen vir, skupen vsem petim procesom, sestavljen iz spremenljivk C ter iz procedure test. Uporaba tega skupnega vira se v glavnem ureja z binarnim semaforjem bin, slično kot v primeru v 4.1.1. V vsakem od procesov se ta skupni vir uporablja v kritičnih delih krl in kr3. Akcija "jesti" je kritična samo relativno na dva delno kritična vira (ki sta leva in desna vilica filozofa, ki želi jesti); z relativnostjo nameravamo izraziti dejstvo, da obstaja možnost sočasnosti akcije "jesti" za dva nesosedna filozofa. Torej je akcija "jesti" samo pogojno kritična; pogoj je izražen s privatnim semaforjem priv(i). Kr2 je torej pogojno kritični del vsakega procesa: od tod je le še korak do uvedbe programskega konstrukta za ta pojem pogojno kritičnega dela procesa - podrobneje si ga bomo ogledali v 4.2.

Oglejmo si sedaj kritične dele krl in k3 vseh procesov. Ugotovimo lahko, da se v teh delih analizira in ureja stanje vseh petih procesov. Preko procedure test se v teh delih tudi realizirajo pogoji za izvedbo pogojno kritičnih delov kr2 vseh procesov. Dejavnosti delov krl in kr3 vseh procesov lahko interpretiramo torej kot nek nov proces, ki ima vlogo tajnika ("secretary", glej 9) oz. monitorja (glej 5) glede na sistem petih procesov. Od tod ideja za uvedbo programskega konstrukta, ki bi odgovarjal pojmu tajnika oz. monitorja - kasneje o tem v 4.3.

Nadaljevanje pregledabo objavljeno v naslednji številki.

primer preprostega mikroračunalnika z mikroprocesorjem motorola 6800

d. kodek

UDK 681.3-181.4

Fakulteta za elektrotehniko, Ljubljana

Večina proizvajalcev mikroprocesorskih vezij podaja skupaj z opisom lastnosti vezij tudi tako imenovane minimalne sistemske konfiguracije. Te konfiguracije so običajno sestavljene iz 2 do 5 LSI vezij in postanejo uporabne šele, ko jih opremimo z ustreznim programom. V članku je opisan primer take konfiguracije na osnovi mikroprocesorja MOTOROLA 6800 in kontrolni program, ki omogoča njeno uporabo za različne namene.

AN EXAMPLE OF SIMPLE MICROCOMPUTER BASED ON MOTOROLA 6800: Most manufacturers of microprocessor circuits include so called minimum system configurations in their product description sheets. These configurations usually consist of 2 to 5 LSI circuits and are usable only if we support them with some kind of a program. The paper gives an example of such configuration based on MOTOROLA 6800 microprocessor and control program, which allows its use for different purposes.

1. UVOD

Danes je že povsem jasno, da predstavljajo mikroprocesorji veliko prelomnico v razvoju in uporabi digitalnih sistemov raznih vrst. Upoštevajoč vse prednosti programsko vodenega delovanja in današnje cene mikroprocesorskih elementov, je pravzaprav že težko najti primere, v katerih z mikroprocesorji tako ali drugačne vrste ne bi mogli zamenjati fiksno ožičene logike. To še posebej velja za počasnejše digitalne naprave, ki jih je mogoče zelo uspešno in hitro realizirati z uporabo MOS mikroprocesorjev.

Kljub veliki aparaturni preprostosti z mikroprocesorji narejenih naprav in kljub dejstvu, da lahko isto napravo s preprosto zamenjavo programa uporabimo za povsem različne namene, pa so uporabniki kmalu spoznali, da uvajanje mikroprocesorjev ni vedno preprosto. Problemi, ki se pojavljajo, so predvsem posledica pomanjkanja izkušenj pri delu s programsko vodenimi sistemi in razmeroma skromne opreme, ki jo ima na voljo večina uporabnikov. Kot pomoč pri reševanju teh problemov je v tem članku opisan preprost mikroračunalnik s kontrolnim programom, ki omogoča prikaz in študij lastnosti mikroprocesorja MOTOROLA 6800 in ki je istočasno uporaben tudi pri realizaciji in preizkušanju različnih kontrolnih naprav.

2. KONFIGURACIJA

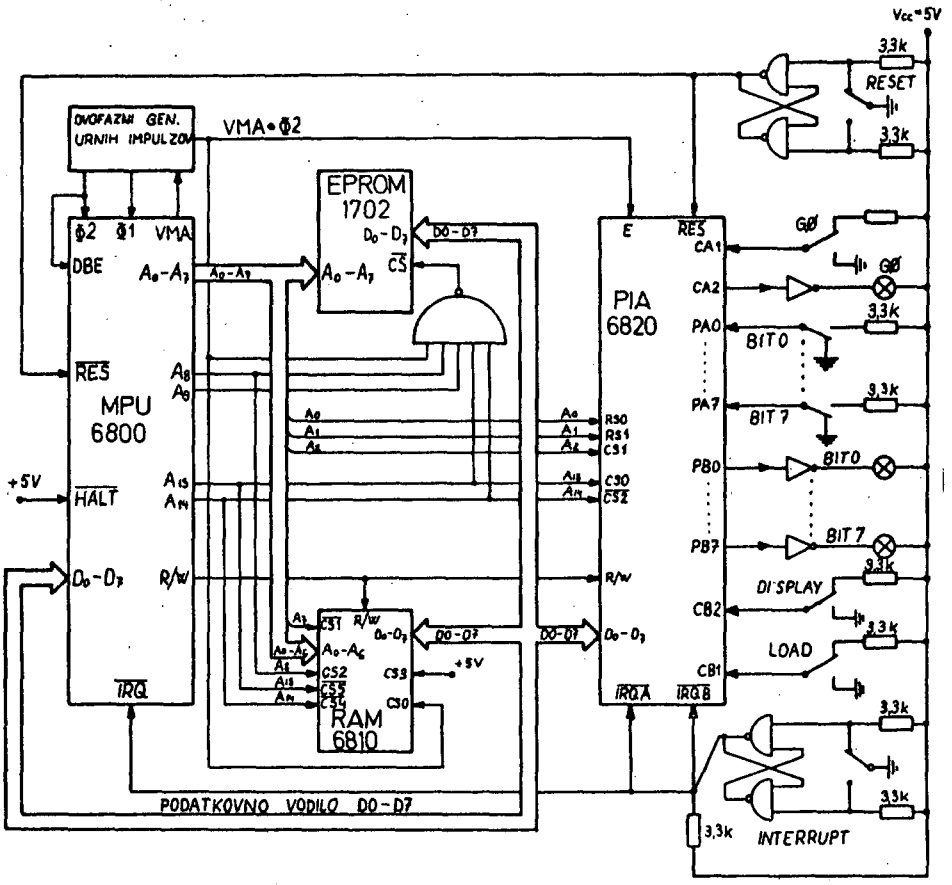
Pri izbiri mikroračunalniške konfiguracije so bili najpomembnejši kriteriji njena preprostost, cenenost in možnost za hitro realizacijo. Vse to ob zahtevi, da je omogočeno preprosto programiranje in da je po potrebi možno dodati še dodatne elemente za posebne namene.

Tem zahtevam je najlaže zadostiti z izpeljanko ene od minimalnih sistemskih konfigiracij, ki jih podaja proizvajalec / 1 /. Značilno za te konfiguracije je, da zahtevajo zelo malo vmesnih TTL vezij in so zato zelo primerne za naš namen.

Uporabljena konfiguracija je prikazana na sliki 1. Namesto težko dosegljivega in razmeroma dragega EPROM pomnilnika S6834, ki bi po proizvajalčevih napotkih sodil v to konfiguracijo, je bil uporabljen starejši in cenejši Intelov 1702 EPROM pomnilnik. Čeprav je kapaciteta tega pomnilnika samo 256 x 8 bitov, je to za naše namene dovolj. Tudi programatorji za te elemente pri nas niso več redkost in zato njihova uporaba ni problematična.

Kot je razvidno iz slike 1 vsebuje mikroračunalnik poleg pomožnih TTL vezij samo 4 LSI vezja. Pri današnjih cenah elementov znaša vrednost sestavnih delov nekaj več kot 1000 dinarjev, pri čemer odpade več kot 80% na LSI vezja. Uporabnik ima na voljo 128 8-bitnih besed v RAM pomnilniku (adrese 0 do 7F), 256 8-bitnih besed v EPROM pomnilniku (adrese FF00 do FFFF), 8 stikal kot vhodno enoto, 8 svetlečih diod kot izhodno enoto, 5 komandnih tipk (RESET, INTERRUPT, LOAD, DISPLAY in GO) in 1 svetlečo diodo kot kontrolni indikator. Potrebujemo napajalni napetosti 5V in -9V; pri tem je napetost -9V potrebna samo za EPROM pomnilnik.

Zizjemo tipk RESET in INTERRUPT, ki delujeta neposredno na vhode mikroprocesorja, so vse ostale tipke, stikala in svetleče diode priključene preko perifernega adapterja PIA (adrese 2004 do 2007). Način njihovega delovanja in s tem njihova funkcija, je zato povsem pod programsko kontrolo in jo lahko z zamenjavo programa spremenimo. Zaradi preprostosti tudi ni bilo realizirano popolno deko-



Slika 1: Konfiguracija

diranje adres in so ostali adresni biti A10, A11, A12 in A15neuporabljeni. Pri razširjanju konfiguracije je zato potrebno upoštevati to zoženost adresnega prostora.

3. KONTROLNI PROGRAM

Omenili smo, da je z izjemo tipk RESET in INTERRUPT, delovanje vseh ostalih tipk in vseh stikal in svetlečih diod, pod programsko kontrolo. Brez ustreznega programa se bo mikroračunalnik s slike 1 odzval vedno samo na tipko RESET, na tipko INTERRUPT pa samo v primeru, ko maskirni bit I v mikroprocesorju ni postavljen. Ker je poleg brisanja registrov v mikroprocesorju edina posledica RESET signala ta, da se programski števec v mikroprocesorju napolni z vsebino adres FFFF in FFFE, je naš mikroračunalnik brez kontrolnega programa povsem neuporaben.

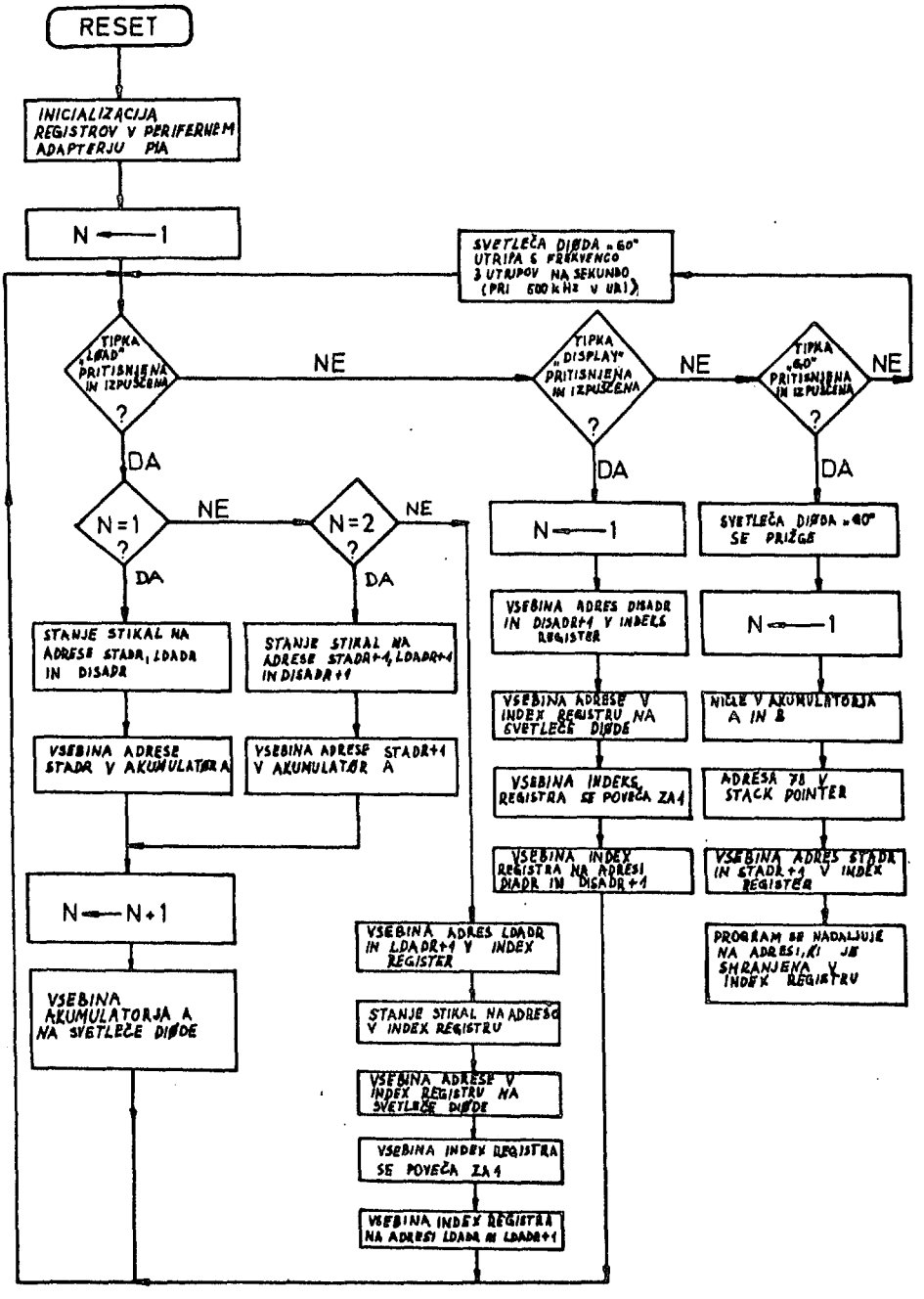
V našem primeru mora kontrolni program omogočati, da lahko uporabnik preko tipk, stikal in svetlečih diod vpisuje in izvaja poljubne programe. Razumljivo je, da bo pri tako skromni konfiguraciji uporabnik moral programirati v strojnem jeziku. Nujno je tudi, da se kontrolni program, ki to programiranje in izvajanje omogoča, nahaja v EPROM pomnilniku, saj bi se sicer ob izklopu napetosti izgubil. Uporabnik ima torej za svoje programe na voljo 128 8-bitnih besed v RAM pomnilniku, kar zadošča za spoznavanje mikroprocesorja, obenem pa omogoča realizacijo in preizkušanje raznih manjših kontrolnih naprav.

Princip delovanja kontrolnega programa je prikazan na sliki 2. Program poženemo s pritiskom na tipko RESET in se prične z inicializacijo registrov v perifernem adapterju PIA. Periferne priključke PA0 do PA7 konfiguriramo kot vhode (ničle v data direction registru A), priključke PB0 do PB7 pa kot izhode (enice v data direction registru B). Kontrolne priključke CA1, CB1 in CB2 konfiguriramo kot vhode, ki reagirajo na zadnjo fronto, (bit CRA1 = CRB1 = 0, CRA4 = CRB4 = 0, CRB5 = 0), medtem ko je priključek CA2 konfiguriran kot izhod (bit CRA5 = 1). Izhoda IRQA in IRQB onemogočimo (CRA0 = CRB0 = 0, CRA3 = CRB3 = 0), ker želimo imeti detekcijo aktiviranja tipk brez programskih prekinitev.

Po inicializaciji vpišemo v spremenljivko N vrednost 1 in pričnemo z otipavanjem stanja tipk. Ker so tipke LOAD, DISPLAY in GO priključene neposredno na vhode CB1, CB2 in CA1, je s posebno subrutino potrebno preprečiti, da se odskakovanje kontaktov v tipkah ne detektira kot večkratno aktiviranje tipke. Ker pride do odskakovanja tako takrat, ko pritisnemo na tipko, kot takrat, ko jo izpustimo, vsebuje subrutina po detekciji zadnje fronte (ob pritisku na tipko) najprej zakasnitev, ki izloči fronte zaradi odskakovanja, nato pa čaka na pozitivno fronto (ob izpustitvi tipke) in se šele po njeni detekciji in ponovni zakasnitvi zaradi odskakovanja, vrne v glavni program.

Aktivnosti, ki se sprožijo ob aktiviranju posameznih tipk so razvidne iz diagrama na sliki 2. S prvim in drugim aktiviranjem tipke LOAD vpišemo preko stikal 16-bitno adreso v spremenljivke STADR, LDADR in DISADR. Ob nadaljnih aktiviranjih tipke LOAD se vsebina stikal prične vpisovati na adreso, ki se nahaja v spremenljivki LDADR in ki se po vsakem vpisu poveča za 1.

Ob aktiviranju tipke DISPLAY se na svetlečih diodah prikaže vsebina, ki se nahaja na adresi shranjeni v spre-



Slika 2: Princip delovanja kontrolnega programa

menljivki DISADR. Ob vsakem prikazu se vrednost adrese v spremenljivki DISADR poveča za 1. Ob aktiviranju tipke GO se prižge svetleča dioda GO, mikroračunalnik pa nadaljuje z izvajanjem programa na adresi, ki se nahaja v spremenljivki STADR. Ta adresa je običajno začetek uporabnikovega programa. Aktiviranje tipke GO zato pomeni konec delovanja kontrolnega programa in s tem možnost, da se stikala, tipke in svetleče diode z novim konfiguriranjem perifernega adapterja PIA uporabijo za povsem druge namene.

Kontrolni program je napisan tako, da za skoke v subrutine ne uporablja ukazov JSR ali BSR. S tem smo se izognili uporabi sklada (stack) in dosegli, da program teče tudi takrat, kadar v mikroračunalniku ni RAM pomnilnika. V tem primeru programiranje seveda ni možno, lahko pa preizkušamo pravilnost delovanja kontrolnega programa. Med izvajanjem kontrolnega programa utripa svetleča dioda GO (pri frekvenci generatorja urinih impulzov 500 kH_Z približno 3 krat v sekundi), kar služi za indikacijo pravilnosti delovanja kontrolnega programa in celotnega mikroračunalnika.

4. ZAKLJUČEK

Opisani mikroračunalnik je bil realiziran v ISKRI Elektromehanika, Kranj in se uspešno uporablja za študij in za preizkušanje različnih razvojnih hipotez. Avtor bi se rad ob tem zahvalil mag. A.Uratniku, ki je dal pobudo zaizdelavo tega dela in diplomiranim inženirjem M.Rogaču, J.Kožuhu in in L.Peternelju, ki so sodelovali pri realizaciji.

5. LITERATURA

/1/ Motorola Semiconductor Products Inc., "M6800 APPLICATION MANUAL", Motorola Inc., 1975.

memoire virtuelle et la technique d'optimisation adaptee a l'interpreteur apl sur mitra 15

d. davčev

UDK 681.327

Faculté d'Eléctronique
de Skopje,Skopje,Yougoslavie

Le but de cet article est d'exposer une réalisation d'une mémoire virtuelle utilisée pour l'interpréteur APL sur Mitra 15,(1 mémoire de 32K mots de 16 bits,1 disque de 800K octets) et les conceptions de la technique d'optimisation utilisée pour la réalisation des opérateurs APL.Le système permet d'amener les pages des mémoires virtuelles implantées sur disque en mémoire centrale et de faire les transformations adresse virtuelle à adresse réelle en cours d'exécution.

VIRTUELNA MEMORIJA I TEHNIKA NA OPTIMIZACIJA PRIMENETA NA INTERPRETATOROT APL NA MITRA 15: Celta na ovoj trud e da ja izloži realizacijata na virtuelnata memorija upotrebena za interpretatorot APL na Mitra 15 (1 memorija od 32K zbora od po 16 bita,1 disk od 800K bajta) i koncepciite na tehnikata na optimizacija upotrebena za realizacijata na operatorite APL. Sistemot dozvoluva prenos na stranici na podatoci od virtuelnite memorii na diskot vo centralnata memorija i transformacii na virtuelnata adresa vo realna adresa za vreme na rabotata.

INTRODUCTION

L'APL Mitra 15 (D.1.) fonctionne sous système MTRD,E dont la configuration est la suivante:1 Mitra 15,1 mémoire de 32K mots de 16 bits,1 opérateur virgule flottante, 1 unité de visualisation Tektronix type 4015 et equipée d'un "Hardcopy",disque rapide à tête fixe de 256 pistes,800K octets.

L'interpréteur APL est écrit en assembleur Mitra 15.La mémoire de travail est allouée dynamiquement d'après la méthode de "First fit"(D.1.).Comme une extension en temps partagéest prévue pour l'APL Mitra 15,la mémoire de travail de l'utilisateur est mise sur le disque.Une copie intégrale de sa zone peut être transférée en mémoire centrale.

MEMOIRE VIRTUELLE

L'accès à un nom est réalisé par "hash coding" grâce à la table des entrées(16 entrées).Chaque entrée de cette table indique une page de la table des noms externes de 256 octets.Un nom contient 8 caractères maximum(8 octets);six octets sont utilisés pour le pointeur au descripteur d'une variable ou d'une ligne zéro d'une fonction.Un indicateur précise s'il s'agit d'un nom de fonction ou d'un nom de variable(fig.1.).

Une variable APL est représentée par un descripteur ,pointé éventuellement par la page de la table des entrées ,et les pages de données pointées par ce descripteur.La taille d'une page est de 256 octets (un secteur sur le disque).

Le descripteur de tableau contient une table de pages.Pour chaque page pointée par ce descripteur il y a trois mots réservés:pour le nombre d'éléments mis dans la page, pour le numéro de secteur sur le disque et pour l'adresse de la page en mémoire centrale.(fig.2.).Le descripteur de J vecteur et le descripteur de scalaire ne contiennent pas la liste des pages car un J vecteur n'a pas de pages et le scalaire se trouve directement dans le descripteur de scalaire (D.1.). la note 1.).
Chaque bloc contient des listes de pointeurs arrière.Pour les descripteurs ce sont des pointeurs vers la table de noms:un descripteur peut être pointé par plusieurs noms de la table de noms.Pour les pages de données ,ce sont des pointeurs vers les descripteurs:les pages de données peuvent être partagées entre plusieurs descripteurs (fig.1.).

Pour chaque fonction on a un descripteur qui contient une liste de pointeurs aux lignes de la fonction(fig.1.).

L'algorithme de remplacement utilisé est "LRU".Deux mots dans chaque bloc sont réservés pour la pile "LRU"(note 2.):
"Chaînage amont"-pointeur au bloc précédent
"Chaînage aval"-pointeur au bloc suivant.
Un descripteur ne peut être "éjecté" sur disque que si les pages pointées par lui sont déjà "éjectées" sur le disque.Si on considère le chaînage:

table de noms-descripteurs-pages

comme un graphe "l'éjection" doit être commencée par ses derniers éléments."L'éjection" d'un bloc sur le disque sera nécessaire s'il n'y a plus de place en mémoire centrale et dans les cas où ce bloc a été modifié.S'il n'a pas été modifié,il peut être effacé car une copie de ce bloc existe

sur le disque.Plusieurs compteurs sont utilisés pour déterminer les relations de chaque bloc avec les autres à tout instant.

Au début du travail,la table des noms est amenée en mémoire centrale.Si on a besoin d'une opération sur un descripteur,on va vérifier dans la table des noms si le descripteur est en mémoire centrale et le travail pourra s'effectuer.Si on a besoin de pages de données ,le procédé précédent est continué en recherchant les adresses des pages dans la liste des pages du descripteur.Les pages sont amenées en mémoire centrale,modifiées éventuellement et puis "éjectées" sur le disque.Dans chaque bloc un bit indique si le bloc a été modifié.

ADRESSAGE D'UN ELEMENT DANS UNE VARIABLE APL

L'analyse des opérateurs APL a montré que certains opérateurs peuvent être réalisés sans aucune opération sur les pages de données d'un tableau APL.Il suffit de modifier les éléments qui caractérisent ce tableau dans son descripteur.Pour appliquer cette méthode,il était nécessaire de définir la représentation des tableaux APL.

Les éléments d'un tableau en mémoire sont rangés dans l'ordre des indices croissants ("row major order").

Soit A un tableau de rang N et L un vecteur dont chaque élément est la valeur d'un indice en indexage d'origine zéro.L'élément A[;/L] aura alors la position suivante dans la mémoire:

$$VBASE + (\rho A) \perp L \qquad ...(1)$$

où VBASE est l'adresse de A[0;...;0], c'est-à-dire l'adresse de premier élément dans la première page de données.Cette représentation ne favorise aucune coordonnée d'un tableau,ce qui est essentiel pour APL.

Puisque le vecteur DEL (voir le schéma du descripteur) représente un vecteur-facteur de pondération,il peut être calculé de la manière suivante:

$$DEL[N] \leftarrow 1$$
$$DEL[I] \leftarrow DEL[I+1] \times (\rho A)[I+1]$$

pour tous les I = 0,1, ... ,N-1

En conséquence, l'expression $(\rho A) \perp L$ peut être transformée en +/DEL x L, d'après la définition de la fonction de décodage.(1) devient alors:

$$VBASE + ABASE ++/ DEL \times L \qquad ...(2)$$

où ABASE est une constante additive, dans ce cas égale à zéro,mais qui peut avoir n'importe quelle valeur.

La réalisation d'opérateurs APL a exigé une rationalisation de l'espace de mémoire utilisé.Par example, si on a un tableau A :

```
      A ← 2 3⍴5
      A
5 5 5
5 5 5
```

il n'est pas nécessaire de stocker dans la mémoire six fois la valeur du scalaire 5. En conséquence, un mot "OPTIM" dans le descripteur sert à indiquer le nombre d'éléments stockés effectivement dans la mémoire.La fonction d'accès (2) est modifiée alors en (voir la note 3.) :

$$OPTIM|VBASE + ABASE ++/DEL \times L \qquad ...(3)$$

EXEMPLES

I.Fonction de symétrie : L'expression Z←Φ[J]X qui représente la fonction de symétrie signifie que les éléments de Z sont les éléments de X, mais dans un ordre inverse.Pour un tableau l'ordre des éléments est inversé le long de la J-ième coordonnée.Cette fonction est réalisée par la transformation suivante du descripteur de l'opérande droit:

$$ABASE \leftarrow ABASE + DEL[J] \times (RVEC[J] - 1)$$
$$DEL[J] \leftarrow -DEL[J]$$

où J est la coordonnée de travail.

II.Restructuration : L'APL utilise des variables de types scalaire,vecteur,matrice ou tableau.La fonction APL "RHO" dyadique est utilisée pour une génération de variable de rang et de dimensions spécifiées: l'opérande droit est restructuré en fonction de l'opérande gauche qui détermine les dimensions et le rang du résultat.

Si l'opérande gauche contient au moins un zéro ,le résultat sera une variable vide.

D'après la définition du "RHO" dyadique,il est évident que le résultat est une variable différente de l'opérande gauche ou de l'opérande droit, mais avec les mêmes éléments que l'opérande droit .

En conséquence, il faut construire un descripteur du résultat qui va utiliser les mêmes pages de données que le descripteur de l'opérande droit.

Dans le cas où le résultat est soit un tableau ou un vecteur vide, soit un scalaire, il est suffisant de construire un descripteur du résultat parce que ces variables n'ont pas de pages de données.Sinon,le calcul suivant est effectué:

1.Dans le cas où l'opérande droit est J vecteur le résultat sera également une J variable.On a ici deux cas :
a)$NBEL \geqslant NBEL_{RES}$

La valeur du $NBEL_{RES}$ sera stockée dans le mot "NBEL" du descripteur du résultat.

Par exemple:

```
   2 2⍴i6
1 2
3 4
```

b)$NBEL_{OD} < NBEL_{RES}$

La valeur du $NBEL_{RES}$ sera également stockée dans le mot "NBEL" du descripteur du résultat,et la valeur du $NBEL_{OD}$ dans le mot "OPTIM"du même descripteur.Les éléments seront trouvés en accord avec la relation (3).Donc,les éléments de l'opérande droit seront répétés.

Par exemple :

```
   2 4⍴i6
1 2 3 4
5 6 1 2
```

2.Si l'opérande droit est un scalaire,il est suffisant de construire une page contenant un élément pour le descripteur du résultat. Dans le mot "OPTIM" du descripteur du résultat on va stocker la valeur "réelle" du $NBEL_{RES}$.

De cette manière, le scalaire sera répété jusqu'à le $NBEL_{RES}$.

Par exemple :

```
    2 2 ⍴5
5 5
5 5
```

3.Si l'opérande droit est un vecteur ou un tableau, il faut utiliser les pages mêmes de données de l'opérande droit pour le descripteur du résultat.Ce descripteur contiendra les pointeurs à ces pages.Les valeurs du mot "OPTIM" et du mot NBEL sont les mêmes que pour le cas d'un J vecteur comme opérande droit.

III.Transposition : Soit V un vecteur 1 5 3 4 2 6 et U un autre vecteur 1 1 1 2 3 3.Si V représente une permutation des dimensions d'un tableau Z de rang 6, le tableau Z sera transformé en un tableau T tel que :

$$T[I;M;K;L;J;N] \leftarrow Z[I;J;K;L;M;N]$$

Le vecteur U va effectuer la transformation suivante:

$$T[I;J;K] \leftarrow Z[I;I;I;J;K;K]$$

On a vu que le vecteur RVEC du descripteur d'un tableau représente les dimensions de ce tableau.Donc, il doit être transformé en fonction du vecteur de permutation. D'autre part, l'ordre des éléments du tableau est également changé.

La réalisation de la fonction de transposition a été faite d'après la fonction APL suivante :

```
RANG ← ⌈/A
R    ← RVEC_OD
D    ← DEL_OD
DEL  ← RANG ↑ DEL_OD
RVEC ← RANG ↑ RVEC_OD

RVEC[I]←L/(I=A)/R          ...(4)
DEL [I]←+/(I=A)/D          ...(5)
```

où I= 1,2,...,RANG ,pour l'origine 1.

A représente l'opérande gauche et les vecteurs RVEC et DEL représentent l'opérande droit.

De cette manière ,nous devrons modifier les éléments de l'opérande droit et utiliser aussi les pages de données de l'opérande droit pour le descripteur du résultat.En conséquence ,nous avons fait une copie de ce descripteur et toutes les modifications ont été effectuées sur cette copie.

Pour réaliser les relations (4) et (5), nous avons utilisé l'opérateur de compression.

Si le rang du résultat est égal au rang de l'opérande droit , les tailles du vecteur RVEC et du vecteur DEL restent les mêmes. Mais,si le rang du résultat est plus petit que le rang de l'opérande droit (un rang plus grand n'est pas possible), il faut "compresser" les vecteurs RVEC et DEL du descripteur de résultat.

Il y a deux cas :

1.L'opérande droit n'est pas un J descripteur.Dans ce cas ,nous devrons connaître le nombre de pages qui sont associées au descripteur : le contenu du mot "PAGE" dans le descripteur.Pour chaque page , il y a trois mots réservés dans la liste des pages ,qui suit le vecteur DEL.Il faut décaler vers le haut (fig.3) ces (nombre de pages x 6)octets de (la différence du rang) x 4 octets.Les mots qui suivent la liste des pages décalée seront remis à zéro jusqu'aux listes des pointeurs arrière.

Soit par exemple :

```
      A← ? 2 3 4 ⍴ 100
      A
 5  10   3  99
 4   8  25  31
28   9  18   7

51  60  27   1
32  88  96  54
16  43  77  11
```

$$RVEC^{(1)} = \begin{bmatrix}2\\3\\4\end{bmatrix} \quad ; \quad DEL^{(1)} = \begin{bmatrix}12\\4\\1\end{bmatrix}$$

et soit :

```
      B← 1 2 1 ⍉A
      B
 5   4  28
60  88  43

2 3 ⍴B
```

$$RVEC^{(2)} = \begin{bmatrix}2\\3\end{bmatrix} \quad ; \quad DEL^{(2)} = \begin{bmatrix}13\\4\end{bmatrix}$$

La figure 3. représente la modification de la copie du descripteur de l'opérande droit.

2.L'opérande droit est un J descripteur.Les pages de données n'existent pas dans ce cas et nous allons décaler seulement les valeurs de "a" et "b" de la même façon que pour le descripteur qui n'est pas un J tableau.

Dans les deux cas , il faut modifier le rang du tableau.A la fin du programme, nous allons indiquer dans le mot "OPTIM" que les éléments du résultat ne sont pas dans un ordre séquentiel.

CONCLUSION

Nous avons exposé les conceptions générales de la réalisation de la mémoire virtuelle adaptée à l'interpréteur APL sur Mitra 15. Tous les problèmes qui se sont posés lors de sa conception ont été dictés par deux critères :

- Place mémoire d'occupation ;
- Temps d'exécution

Les expériences ont montré que la réalisation d'un interpréteur APL n'est pas incompatible avec une mémoire virtuelle implantée sur un petit calculateur et que cela n'entraîne pas des temps d'exécution trop longs.

BIBLIOGRAPHIE

A.1.APL Congress-Copenhague,1973
A.2.Abrams,P.S.,An APL machine,SLAC Report n° 114,Stanford University,1970
A.3.Auslander,M.A.,Virtual storage operating systems,IBM-S.J.,vol.12,n° 4,1973
B.1.Boote,W.P.,Clark,S.R.,Rourke,T.A.,Simulation of a paging computer system,The computer Journal,vol.15,n° 1,1972
B.2.Bensonssan,A.,Clingen,C.T.,The multics virtual memory,ACM symposium,1969
B.3.Belady,L.A.,A study of remplacement algorithms for a virtual storage computer, IBM Syst.J. 5,n° 2,1966
C.1.Christensen,C.,Hanse,A.D.,A multiprogramming virtual system for a small computer, Spring Joint Computer Conference,1970
D.1.Davčev,D.,Thèse de docteur ingénieur: Interpréteur APL sur Mitra 15,Université de Paris,Centre d'Orsay, 1975
D.2.Denning,P.J.,Virtual memory,Computing surveys 2,n° 3,1970
D.3.Denning,P.J.,Third Generation Computer Systems,Computing Surveys 3,n° 4 ,1971
D.4.Demars,G.,Rault,J.C.,Ruggin,G.,Le langage et les systèmes APL,Masson et Cie,Paris, 1974
F.1.Falkoff,A.D.,Iverson,K.E.,The Design of APL,IBM J.Res. Develop.,July 1973
W.1.Wilkes,M.V.,The dimamics of paging,The Comp.Journal,vol.16,n° 1,1973

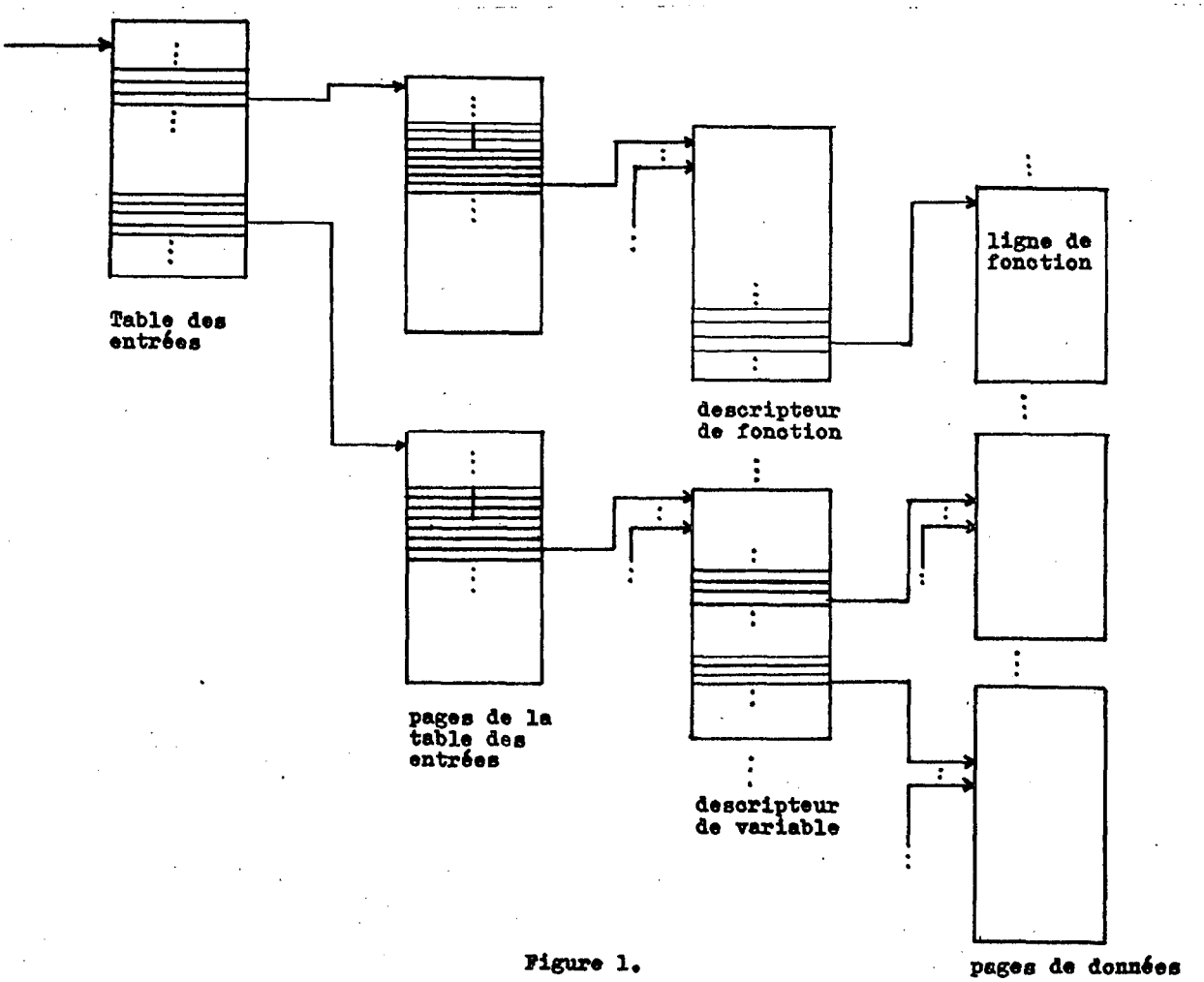


Figure 1.

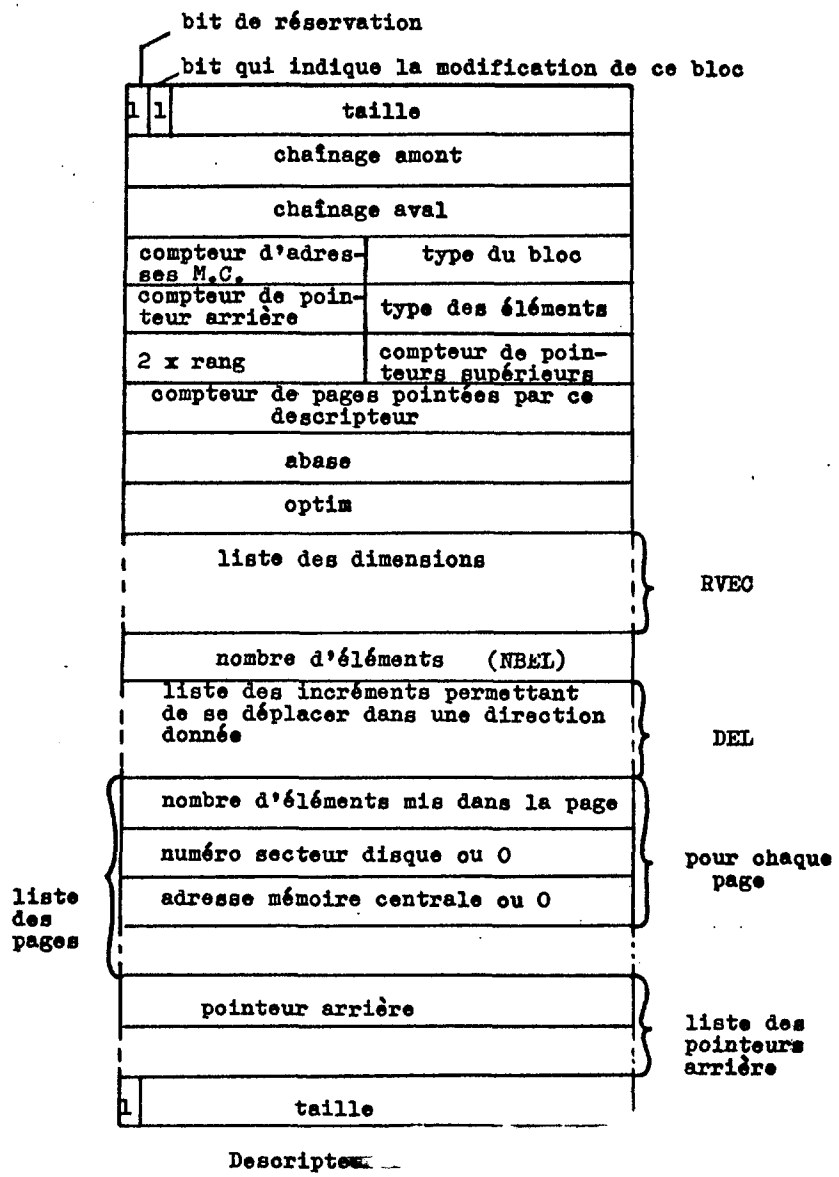


Figure 2.

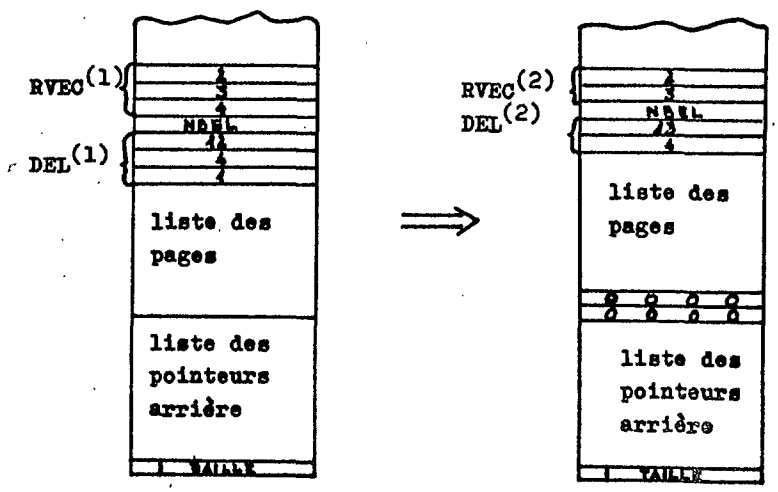


Figure 3.

Note 1. J vecteur représente le générateur d'indices "i".Si N est un entier non négatif alors iN (iota N) correspond à un vecteur des N premiers entiers.Le descripteur de J vecteur contient les valeurs "a"(origine) et "b"(pas) : x-ième élément = a+bx .

Note 2. "LRU" ("Least Recently Used").La page qui n'a pas été référencée depuis le plus long temps est transférée sur le disque.

Note 3. Pour la signification de la fonction de résidu dyadique APL ("|") voir (D.4.). Si la valeur du mot OPTIM n'est pas égale à zéro le programme de lecture doit utiliser l'expression (3).Dans ce cas les éléments du résultat ne sont pas dans un ordre séquentiel.Dans les cas où l'optimisation n'est pas possible (pour certains opérateurs APL) le mot OPTIM est égal à zéro et les éléments du résultat sont dans un ordre séquentiel.Le programme de lecture peut éviter le calcul d'après l'expression (3) et la lecture est faite dans un ordre séquentiel.

urejanje zaporedij

v. batagelj

UDK 681.3.06/07

FNT, VTO matematika in mehanika
Ljubljana

Namen sestavka je seznaniti bralca z dvema učinkovitima primerjalnima postopkoma za urejanje zaporedij znotraj pomnilnika. To sta "heap sort" in "quick sort". Poleg splošnega uvoda v postopke urejanja zaporedij, so podani tudi izpisi ustreznih podprogramov, napisanih v structranu, in časovne značilnosti le-teh pri urejanju slučajno zgeneriranih zaporedij na računalniku Cyber.

INTERNAL SORTING METHODS - In the paper two efficient internal sorting methods, "heap sort" and "quick sort", are presented. Their time characteristics for sorting random generated data on Cyber are also given.

Imejmo zaporedje n podatkov iz linearno urejene množice $(P , \leqslant)$

$$p_1 , p_2 , p_3 , \cdots , p_n$$

Urediti dano zaporedje, pomeni poiskati tako permutacijo indeksov π , da bo zaporedje

$$p_{\pi(1)}, p_{\pi(2)}, p_{\pi(3)}, \cdots , p_{\pi(n)}$$

urejeno v nepadajočem vrstnem redu; kar pomeni, da za vsak par indeksov i in j velja:

$$i < j \Longrightarrow p_{\pi(i)} < p_{\pi(j)}$$

Pogosto pri urejanju preuredimo kar samo zaporedje (permutacija je določena implicitno).

Postopke urejanja zaporedij delimo na
- zunanje : zaporedje je datoteka na zunanjem pomnilniku
- notranje : zaporedje se v celoti nahaja v hitrem pomnilniku.

Postopke urejanja razvrščamo naprej na
- postopke, ki upoštevajo notranjo strukturo podatkov
- primerjalne postopke, ki upoštevajo samo linearno urejenost množice P

V tem sestavku se bomo omejili na primerjalne postopke urejanja znotraj pomnilnika. Ti postopki se v glavne uporabljajo kot sestavni deli drugih postopkov; morda najpogosteje pred izpisom podatkov.

V programih najpogosteje srečujemo takoimenovani "bubble sort" postopek

```
k ← n
for ( i = 1, n-1 ) do
    k ← k - 1
    for ( j = 1, k ) do
        if ( p(j) > p(j+1) ) then
        {premenjamo podatka}
            t ← p(j)
            p(j) ← p(j+1)
            p(j+1) ← t
        endif
    endfor
endfor
```

ali njegove inačice oziroma izboljšave (glej primer v dodatku).

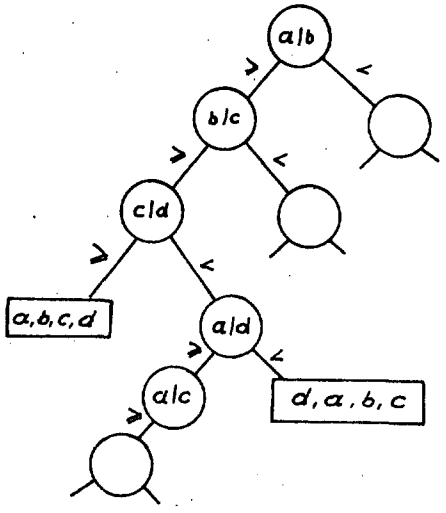
Kompleksnost postopkov urejanja je odvisna od večih parametrov. Za primerjalne postopke je najznačilnejši parameter število primerjanj.

Poglejmo kolikšno je število primerjanj za gornjo verzijo "bubble sort" postopka. Iz programa vidimo, da se primerjanje izvrši

$$\sum_{i=1}^{n-1} \sum_{j=1}^{n-i} 1$$

krat. Torej $n(n-1)/2$ krat. Izboljšani program iz dodatka potrebuje v splošnem manj primerjanj, vendar je v povprečju še vedno reda n^2. V najslabšem primeru, ko je zaporedje urejeno v nenaraščajočem vrstnem redu, pa je enakovreden gornjemu postopku.

Postavi se vprašanje: koliko najmanj primerjanj je potrebnih pri urejanju zaporedja n podatkov z "najboljšim" primerjalnim postopkom? Do odgovora na zastavljeno vprašanje pridemo z naslednjim razmislekom. Vse možne izvršitve poljubnega primerjalnega postopka, lahko prikažemo z binarnim drevesom, katerega notranje točke so primerjanja, končne točke (listi) pa permutacije.
Primer dela mogočega drevesa izvršitve pri urejanju zaporedja a,b,c,d je prikazan na sliki.



Torej je kompleksnost postopka povezana z dolžinami poti po tem drevesu od korena do lista. Zato se vprašamo: koliko je najmanj največja dolžina poti po drevesu? Število listov drevesa (permutacij) je vsaj $n!$. Po poteh dolžine k lahko pridemo v največ 2^k listov. Torej mora veljati

$$2^k \geq n!$$

oziroma po Stirlingovem približku

$$k \geq n \log_2 n - n/\ln 2 + \log_2 n/2 + c$$

Torej je

$$k_{opt} \approx n \log_2 n$$

Od tu vidimo, da je kompleksnost "najboljših" primerjalnih postopkov urejanja zaporedij reda $n \log n$.
Poleg časovne kompleksnosti (število operacij) je pomembna tudi prostorska kompleksnost (zahteve po pomnilniku). Najboljši postopki urejanja bodo za urejanje potrebovali le prostor, ki ga zaseda zaporedje podatkov (in program), drugi pa zahtevajo še dodaten prostor za shranjevanje pomožnih rezultatov.

V dodatku sta opisana dva izmed "najboljših" postopkov.
Prvi postopek, ki ga v literaturi srečamo pod imenom "heap sort", ima časovno kompleksnost (največjo dolžino poti po drevesu) reda $n \log n$ in ne potrebuje dodatnega prostora.

Drugi postopek, imenovan "quick sort", pa je primer postopka, ki potrebuje še $c \log n$ dodatnega prostora in ima povprečno časovno kompleksnost (povprečna dolžina poti po drevesu) reda $n \log n$.
Poskusi na slučajnih podatkih (za dane programe glej tabelo v dodatku), kažejo, da je "quick sort" (v povprečju) boljši od "heap sort-a"; obstajajo pa tudi zaporedja, za katera je reda n^2. Zato previdni programerji dajejo prednost zanesljivemu "heap sort-u".

Več o postopkih urejanja zaporedij lahko bralec prebere v knjigah:

D.E. Knuth: The Art of Computer Programming; vol.3/ sorting and searching; Addison-Wesley, 1973

in

A.V. Aho, J.E. Hopcroft, J.D. Ullman: The Design and Analysis of Computer Algorithms; Addison-Wesley, 1974

T A B E L A : časovne značilnosti za slučajne podatke za Cyber (čas je merjen v sekundah)

dolžina zaporedja	BUBBLE meritve	F_b	HEAP meritve	F_h	QUICK meritve	F_q
0						
10	.001	.001	.001	.001	.002	.001
50	.019	.018	.010	.010	.008	.010
100	.074	.072	.023	.024	.018	.023
200	.27	.29	.055	.055	.045	.052
500	1.8	1.8	.16	.16	.14	.15
1000	7.2	7.2	.36	.36	.34	.34
2000	28.	29.	.79	.79	.78	.74
3000	64.	65.	1.25	1.25	1.2	1.2
5000		180.	2.2	2.2	1.7	2.1
10000		720.	4.8	4.8	4.5	4.5
20000		2900.	10.	10.	9.8	9.7
30000		6500.	16.	16.	15.	15.
100000		72000.		60.		56.

$F_b = 7.2\ n^2\ 10^{-6}$ $F_h = 3.6\ n \log_2 n\ 10^{-5}$ $F_q = 3.4\ n \log_2 n\ 10^{-5}$

B U B B L E - SORT

```
      SUBROUTINE SORT(TAB,LENGTH)
      INTEGER TAB    ( 1       )
      INTEGER BOUND , CHANGE
*
*  TABELO  TAB(1..LENGTH)  UREJAMO TAKO, DA NA KONEC NEUREJENEGA DELA
*  TABELE SPRAVIMO NJEGOV NAJVECJI ELEMENT. TO PONAVLJAMO VSE DOKLER NI
*  TABELA UREJENA.
*
      IND = LENGTH
      REPEAT
*
*     NEUREJENI DEL TABELE PREGLEDUJEMO OD ZACETKA PROTI KONCU. CE TE-
*     KOCA ZAPOREDNA ELEMENTA NISTA V PRAVEM VRSTNEM REDU, JU PREMENJAMO
*
         BOUND = IND - 1
         IND   = 0
         FOR (I = 1,BOUND) DO
            NEXT = I + 1
            IF (TAB(I).GT.TAB(NEXT)) THEN
*
*              TEKOCA DVOJICA NI V PRAVEM VRSTNEM REDU, ZATO ELEMENTA
*              PREMENJAMO
*
                 CHANGE    = TAB(I)
                 TAB(I)    = TAB(NEXT)
                 TAB(NEXT) = CHANGE
*
*              ZAPOMNIMO SI INDEKS PREMENE. PO KONCANEM PREGLEDOVANJU
*              DOLOCA INDEKS ZADNJE PREMENE KONEC NEUREJENEGA DELA TABELE.
*
                 IND = I
              ENDIF
           ENDFOR
      UNTIL (IND.LE.1) ENDREPEAT
*
*  TABELA JE UREJENA
*
      RETURN
*
      END
```

H E A P - SORT

```
      SUBROUTINE SORT(TAB,LENGTH)
      INTEGER CHANGE, FATHER, ROOT , SON
      INTEGER TAB    ( 1      )
      DATA    ROOT   / 1      /
*
*  NAD TABELO TAB(1..LENGTH) RAZPNEMO BINARNO DREVO, TAKO DA VELJA:
*  ROOT = 1  IN ZA POLJUBNO TOCKO  NODE  :  FATHER = ENTIER(NODE/2) ,
*  LEFTSON = 2*NODE  TER  RIGHTSON = LEFTSON + 1 . DREVO, V KATEREM JE
*  VRH VSAKEGA PODDREVESA VECJI OD VSEH SVOJIH NASLEDNIKOV IMENUJEMO
*  KOPICA (HEAP). TABELO  TAB  UREJAMO TAKO, DA NAJPREJ SESTAVIMO IZ
*  NJE KOPICO. NA VRHU KOPICE JE NAJVECJI ELEMENT TABELE. ODSTRANIMO GA.
*  PREOSTALE ELEMENTE ZOPET PREUREDIMO V KUPICO. TO PONAVLJAMO VSEDOKLER
*  NI TABELA UREJENA.
*
*  SESTAVIMO KOPICO
*
      FOR (NODE = 2*LENGTH) DO
*
*     PREDPOSTAVIMO, DA JE  TAB(1..NODE-1)  KOPICA. PREDELAJMO V KOPICO
*     TAB(1..NODE) , TAKO DA POMAKNEMO ELEMENT  TAB(NODE)  NAVZGOR NA
*     NJEGOVO MESTO V HIERARHIJI.
*
        SON = NODE
        REPEAT
          FATHER = SON/2
          IF (TAB(FATHER).GE.TAB(SON)) EXIT
*
*       POMAKNEMO SE NAVZGOR
*
          CHANGE      = TAB(FATHER)
          TAB(FATHER) = TAB(SON)
          TAB(SON)    = CHANGE
          SON         = FATHER
        UNTIL (SON.LE.ROOT) ENDREPEAT
      ENDFOR
*
*  UREDIMO TABELO
*
      NODE = LENGTH
      LOOP
*
*     PREMENJAMO  TAB(NODE)  IN TAB(ROOT) . DEL TABELE  TAB(NODE..
*     LENGTH)  JE TAKO UREJEN.
*
        CHANGE      = TAB(NODE)
        TAB(NODE)   = TAB(ROOT)
        TAB(ROOT)   = CHANGE
*
*     ELEMENT  TAB(ROOT)  POMAKNEMO PO DELU TABELE  TAB(1..NODE-1)
*     PRIPADAJOCEM DREVESU NAVZDOL, TAKO DA ZASEDE V HIERARHIJI
*     PRIPADAJOCE MU MESTO. DEL TABELE  TAB(1..NODE-1)  JE POSTAL
*     KOPICA.
*
        NODE = NODE - 1
        IF (NODE.LE.ROOT) RETURN
        FATHER = ROOT
        LOOP
          SON = 2*FATHER
          IF (SON.GT.NODE) EXIT
*
*       DOLOCIMO VECJEGA OD BRATOV
*
          IF ((TAB(SON).LT.TAB(SON+1)).AND.(SON.LT.NODE)) SON = SON+1
          IF (TAB(FATHER).GE.TAB(SON)) EXIT
*
*       POMAKNEMO SE NAVZDOL
*
          CHANGE      = TAB(FATHER)
          TAB(FATHER) = TAB(SON)
          TAB(SON)    = CHANGE
          FATHER      = SON
        ENDLOOP
      ENDLOOP
*
      END
```

```
      Q U I C K - SORT


      SUBROUTINE SORT(TAB,LENGTH)
      INTEGER CHANGE, DELTA , DEPTH , RIGHT
      INTEGER TAB    ( 1     ), STACK ( 3     )
*
*  TABELO  TAB(1..LENGTH)  UREJAMO TAKO, DA JO GLEDE NA IZBRANI DELILNI
*  ELEMENT RAZBIJEMO NA DVA DELA, PRI ČEMER DELILNI ELEMENT ZASEDE SVOJE
*  MESTO V UREJENI TABELI IN JE VSAK ELEMENT PRVEGA DELA MANJŠI OD VSA-
*  KEGA ELEMENTA IZ DRUGEGA DELA. ZA VSAK DEL POSEBEJ, ČE NI UREJEN, PO-
*  STOPEK PONOVIMO. TABELA JE UREJENA, KO SO UREJENI VSI DOBLJENI DELI.
*
      DEPTH =                 ... GLOBINA LASTNEGA SKLADA
      LEFT  = 1               ... SPODNJI INDEKS DELA TABELE, KI GA UREJAMO
      RIGHT = LENGTH          ... ZGORNJI INDEKS DELA TABELE, KI GA UREJAMO
      LOOP
*
*     ČE TEKOČI DEL TABELE SE NI UREJEN, GA RAZBIJEMO IN NADALJUJEMO Z
*     UREJENJEM KRAJŠEGA DELA. DALJŠI DEL SI ZAPOMNIMO; DRUGAČE NADALJU-
*     JEMO Z UREJANJEM ENEGA IZMED PREOSTALIH NEUREJENIH DELOV; ČE NI
*     NOBENEGA VEČ, JE TABELA UREJENA.
*
         DELTA = RIGHT - LEFT
         IF (DELTA.GT.1) THEN
*
*        TEKOČI DEL TABELE VSEBUJE VEČ KOT 2 ELEMENTA. DELITEV NADA-
*        LJUJEMO. SIRŠI DEL TABELE SI ZAPOMNIMO (MEJI SPRAVIMO V SKLAD)
*
            CALL SPLIT(TAB,LEFT,RIGHT,NLEFT,NRIGHT)
            IF (NLEFT.LT.NRIGHT) THEN
               DEPTH = DEPTH + 2
               STACK(DEPTH-1) = NLEFT
               STACK(DEPTH)   = NRIGHT
            ENDIF
         ELSE
*
*        TEKOČI DEL TABELE VSEBUJE NAJVEČ 2 ELEMENTA. ČE STA 2,
*        JU, ČE JE POTREBNO, UREDIMO.
*
            IF ((DELTA.EQ.1).AND.(TAB(LEFT).GT.TAB(RIGHT))) THEN
               CHANGE      = TAB(LEFT)
               TAB(LEFT)   = TAB(RIGHT)
               TAB(RIGHT)  = CHANGE
            ENDIF
*
*        PRIPRAVIMO NOV DEL TABELE (MEJI VZAMEMO Z VRHA SKLADA); ČE GA
*        NI KONČAMO
*
            IF (DEPTH.LE.0) RETURN
            LEFT  = STACK(DEPTH-1)
            RIGHT = STACK(DEPTH)
            DEPTH = DEPTH - 2
         ENDIF
      ENDLOOP
*
      END
```

```
      SUBROUTINE SPLIT(TAB,LEFT,RIGHT,NLEFT,NRIGHT)
      INTEGER TAB(  1     )
      INTEGER CHANGE, CUT   , RIGHT , RSCAN , TEMP
*
*  SPLIT JE POMOZNA RUTINA PODPROGRAMA (QUICK)SORT. DEL TABELE
*  TAB(LEFT..RIGHT) RAZBIJE NA DALJSI DEL  TAB(NLEFT..NRIGHT)  IN
*  KRAJSI DEL  TAB(LEFT..RIGHT) .
*
*  DEL TABELE  TAB(LEFT..RIGHT)  PREUREDIMO TAKO, DA SO NA LEVI STRANI
*  DELILNEGA ELEMENTA VSI (OD NJEGA) MANJSI; NA DESNI PA VSI VECJI ALI
*  ENAKI ELEMENTI. S TEM DELILNI ELEMENT ZASEDE MESTO, KI GA ZASEDA V
*  UREJENI TABELI.
*
      CUT    = (LEFT + RIGHT)/2
      TEMP   = TAB(CUT)          ... DELILNI ELEMENT
      RSCAN  = RIGHT             ... ZGORNJI INDEKS SE NEPREUREJENEGA DELA
      LSCAN  = LEFT              ... SPODNJI INDEKS SE NEPREUREJENEGA DELA
*
*  TABELO PREUREJAMO TAKO, DA POISCEMO PRVI ELEMENT Z LEVE, KI JE VECJI,
*  IN PRVI ELEMENT Z DESNE, KI JE MANJSI OD DELILNEGA. ELEMENTA PREME-
*  NJAMO. TO PONAVLJAMO VSE DOKLER OBSTAJA NEPREUREJENI DEL TABELE.
*
      LOOP
        WHILE (TAB(LSCAN).LT.TEMP) DO
          LSCAN = LSCAN + 1
        ENDWHILE
        WHILE (TAB(RSCAN).GT.TEMP) DO
          RSCAN = RSCAN - 1
        ENDWHILE
*
*       DOBILI SMO DVA ELEMENTA, KI STOJITA NA NAPACNI STRANI DELILNEGA
*       ELEMENTA. PREMENJAMO JU.
*
        IF (LSCAN.GT.RSCAN) EXIT
        CHANGE     = TAB(RSCAN)
        TAB(RSCAN) = TAB(LSCAN)
        TAB(LSCAN) = CHANGE
        RSCAN      = RSCAN - 1
        LSCAN      = LSCAN + 1
      ENDLOOP
*
*  DOLOCIMO NOVA DELA TABELE
*
      IF ((LEFT+RIGHT).LT.(LSCAN+RSCAN)) THEN
*
*       LEVI DEL JE VECJI
*
        NLEFT  = LEFT
        NRIGHT = RSCAN
        LEFT   = LSCAN
      ELSE
*
*       DESNI DEL JE VECJI
*
        NLEFT  = LSCAN
        NRIGHT = RIGHT
        RIGHT  = RSCAN
      ENDIF
      RETURN
*
      END
```

kako dobiti aktualne informacije o mikro računalnikih ?

Strokovnjak iz prakse, ko se prvič srečuje s tem novim področjem, potrebuje osnovne aktualne informacije o mikro računalnikih oziroma mikro procesorjih. Prvi korak je predvsem spoznavanje novih tehnoloških elementov, svojstvenih postopkov ter novih izrazov in definicij. Bralcem želimo v kratkem pokazati nekaj poti, ki lahko vodijo do podrobnejših inforamcij iz omenjene problematike.

Kot prvo omenjamo informacije, ki jih podajajo proizvajalci z opisom svojih izdelkov. V večina primerov so ti opisi lahko razumljivi s številnimi ilustracijami. Med najbolj poznanimi so: Intel-ov "8080 Users Manual", Motorolina vêlika priročnika "Mikroprocessor Applications Manual" in opis sistema 6800, potem Fairchild-ov "F-8 Circuit Data Book", priročnik tvrdke Signetic (2650), opisi mikroprocesorjev tvrdke Rockwell, opisi tvrdke National kot so SC/MP, PACE in IMP-16, prav tako Zilog-ov " Z 80-MBC Hardware User's Manual" in Z 80 -MBC Software User's Manual.

Drugi viri informacij proizvajalcev se občasno pojavljajo v periodičnih strokovnih revijah kot so na primer Electronics, Electronic Design, Computer itd., vendar so te informacije lahko na spremenljivem strokovnem nivoju. Koristna je mesečna revija, ki objavlja različne probleme iz področja mikro procesorjev to je "New Logic Note Book". Zanimiv je tudi mesečnik "Microcomputer Digest" (Copertino, Calif.).

Za tiste, ki imajo doma svoj mikro računalnik kot konjiček pa sta zanimivi reviji: "Personal Computing" in "Byte". Reviji sta seveda zanimivi tudi za profesionalce, saj ni strokovni nivo nič nižji kot pri zgoraj naštetih revijah.

Na zahodu je izšlo več knjig iz tega področja. Opaziti je, da je veliko od njih z zelo podobnim načinom podajanja snovi z manj ali več podobno vsebino. V ZDA ocenjujejo kot enega od boljših priročnikov s področja mikro računalnikov broširano knjigo "An Introduction to Microcomputers" avtorski kolektiv Adam Osborne in sodelavci (2950 Seventh st., Berkeley, Calif. 94710). Omenimo tudi knjigo "Microprocessors" iz "Electronics Book"serije, ki predstavlja zbirko v reviji Electronics objavljenih člankov. Nova knjiga s tega področja je knjiga jugoslovanskega avtorja B. Součka z naslovom "Microprocessors and Microcomputers" izdana v letu 1976. Ena boljših knjig je tudi "Microcomputers/Microprocessors: Hardware, Software and Applications" avtorjev J.L. Hiburne-a in P.M. Julich-a (Prentice-Hall, Inc.).

Dober izbor aktualnih informacij predstavljajo organizirani seminarji,predvsem neodvisni, ki običajno v nekaj dnevnih predavanjih posredujejo snov iz področja mikro računalnikov. V Evropi je npr. poznana tvrdka SYBEX (313, Rue Lecourbe 75015 Paris), ki v več evropskih mestih organizira seminarje iz področja mikro računalniške tehnike. Seveda se z organizacijo tečajev in seminarjev iz mikro računalniške problematike ukvarjajo tudi proizvajalci in številne univerze.

študentska vprašanja

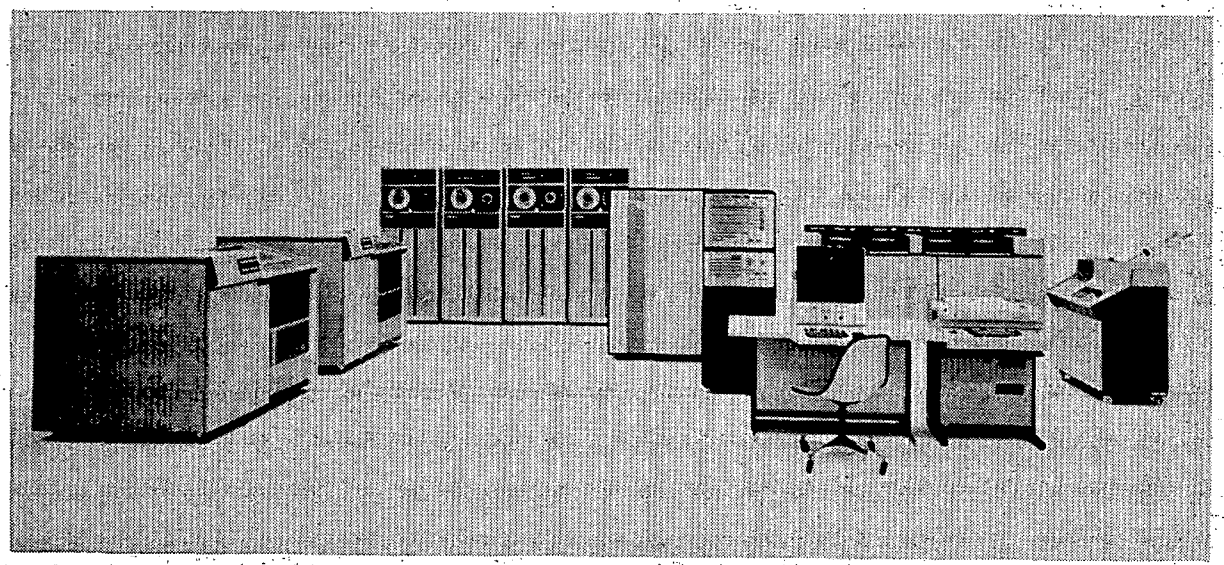
Zakaj uvajamo to rubriko?

Študiramo vedo, ki je sedaj v intenzivnem razvoju in vejanju. Seveda to zelo čutimo. Programi predmetov, ki jih poslušamo, se dopolnjujejo vsako leto, uvajajo se novi pojmi in novi predmeti skladno s trenutnim razvojem doma in po svetu. To na žalost ne gre vedno dovolj hitro in temeljito.
Tudi mi občutimo trenutne probleme visokega šolstva: najbolj nam manjka praksa. Po končanem študiju se počutimo nekoliko premalo pripravljeni za praktično delo. Razvijamo programsko opremo, nekaj slišimo o materialni opremi, dejansko pa o velikih računalnikih vemo zelo malo. Program vstavimo v stroj in čakamo kaj bo. Ali je rezultat odvisen tudi od stroja? Vemo,da ni pa vseeno čutimo odpor. Sovražimo stroj, če kaj ni prav. Odvisni smo od stroja in mogoče bi zato bilo dobro, da ga bolje spoznamo, ne samo iz knjig, marveč tudi v neposrednem stiku. Takoj dobimo odgovor na vse to: denarja ni, enkrat pa bo in bodo terminali ter vse ostalo. Kaj pa mi, ki smo sedaj tu? Mogoče bi se med tem časom lahko spoznali z malimi in mikro sistemi, kar pa je vprašanje zagnanosti pedagoških delavcev in elestičnosti administracije. Kdo bo predaval o tem, kje dobiti sredstva in material za eventualno praktično izvajanje, čemur sledi še administrativna obravnava in priznavanje predmeta itd.
Tako kot nas sedaj izobražujejo, smo lahko samo raziskovalci, razvijalci raznih teorij, v glavnem orientirani na področje programske opreme.
Vprašanje je le ali bo za raziskovalce dovolj delovnih mest? V gospodarstvu gotovo ne, na fakultetah tudi ne in tudi ne na institutih. Zakaj bi namreč neka organizacija imela raziskovalca-računalnikarja, ki se bo lahko šolal naprej le kot elektrotehnik, ekonomist, kibernetik in podobno ali pa bo svoje izobraževanje nadaljeval v tujini, ker še ni podiplomskega študija. Nas študentov računalništva in informatike je sedaj malo, planiran pa je vse večji vpis. Ali bo dovolj pedagoškega kadra za take načrte? Od kje bo prišel? Kot v vsem visokem šolstvu je tudi pri nas potrebno spreminjati sistem. Poslušanje predavanj na stari način, vmes učenje in kolokviji in pisanje programov brez dejanskega "študiranja", iz vsega tega ne bo velikih rezultatov.
Tudi ni nikjer jasno definirano, kaj gospodarstvo od nas pričakuje:eni želijo to, drugi drugo. Verjetno je potrebno narediti resno zasnovo študija računalništva, tako za dvoletni kot štiriletni s predvidenimi usmeritvami. Mogoče bi se takrat lahko bolj konkretno pogovarjali, ker bi točno vedeli, kakšna je naloga posameznika in kaj bo kdo delal.
To je samo nekaj vprašanj, ki tarejo ne samo nas študente ampak tudi pedagoške delavce in gospodarstvo.
Študentje smo neposredno prizadeti, vendar gledamo na problematiko kompleksno. Zato smo vpeljali to rubriko, v kateri naj se objavljajo mnenja in misli študentov smeri računalništva in informatike.

Neda Papić

FACOM 230-38S
FACOM 230-38

Snažni novi kompjuterski sistemi srednje veličine tvrtke FUJITSU



Oba elektronička računala posjeduju vrlo efikasni sistemski software s potpuno modularnim operativnim sistemom OSII/VS i multi-virtualnom memorijom, što omogućava multi-programiranje koje nije ograničeno veličinom memorije. Četiri on-line softwareska paketa omogućavaju stvaranje praktično svih vrsta on-line sistema, kao i sistema koji zadovoljavaju sve zahtjeve aplidvije vrste banke podataka komunikacionih sistema koji zadovoljavaju sve zahtjeve aplikacionih sistema i omogućavaju korisniku stvaranje svojeg vlastitog informacionog sistema.

FACOM

Svaki korisnik ima potpunu slobodu u projektiranju batch ili on-line sistema obrade, budući da postoji široka mogučnost izbora perifernih jedinica i terminala, a isto tako i mogučnost velikog proširenja glavne memorije. Za FACOM 230-38 od 96 do 512 KB, a za FACOM 230-38S do 96 do 256 KB. Također nema problema što se tiče kompatibilnosti s ostalim sistemima, budući da je moguće koristiti sve više programske jezike: PLI, COBOL, FORTRAN.
Posljednji benchmark rezultati pokazali su da su FACOM 230-38/38S iznad ostalih kompjuterskih sistema u toj klasi.

Za daljnja obavještenja o novim kompjuterskim sistemima tvrdke FUJITSU -vodećeg proizvođača elektroničkih sistema u Japanu - molimo obratite se:

ZPR - Zavod za primjenu elektroničkih računala i ekonomski inženjering

ZAGREB, Savska 56
Tel. 510-670
Telex: 21689 YU ZPR FJ

LJUBLJANA, Topniška 45
Tel. 311-059

富士通

FUJITSU LIMITED

Communications and Electronics
Marunouchi, Tokyo, Japan

novice in zanimivosti

Celo najnovejši algoritmi za računanje z matrikami so še vedno prepočasni. Operacije pri matrikah porabijo veliko računalniškega časa zaradi nezmožnosti procesorjev, da bi obdelovali in naslavljali z enim samim ukazom. V ta namen so v IMB razvili procesor polj 3838, ki se lahko uporablja z modeloma 158 in 168 računalnika IBM 370. V tem procesorju en sam ukaz izvede zaporedje operacij nad poljem ali množico sorodnih podatkov. Uporabnik ima dostop do 50 algoritmov, ki se v grobem delijo v bazične in kompleksne algoritme množenja, ukaze prenosa podatkov, hitre Fourijerjeve transformacije ter specialne funkcije. Hitra Fourierjeva transformacija se tako izvrši nad 1024 točkami v 2,95 msek.

Kasete na malih sistemih bodo počasi izpodrinjene z minidisketami. Te se že pojavljajo v sistemu SA 400 proizvajalca CPU *Computers of Woking*. *Minidisketna enota* zavzema isto prostornino kot povprečna kaseta. Ker je dostop podatkov nakljućen, je obdelava le-teh občutno hitrejša. SA 400 uporablja disketo v velikosti 5,25 inč.

Magnetni računalniški trakovi in diski so zanesljivi pomnilniki, na katerih nastanejo napake skoraj vedno zaradi fizičnega uničenja in ne zaradi prisotnosti motilne energije. Edina energija katere vpliv pokvari podatke, je magnetna energija, pa še pri tej mora biti direkten kontakt med magnetom in medijem. Zgoraj navedeno je rezultat raziskave vplivov na magnetne medije, ki je bila izvedena v National Bureau of Standards v ZDA. Magnetne trakove so izpostavili različnim vplivom kot so: detektorji kovin na letališčih, mikrovalovi v mikrovalovni pečici, radarski signali v sami bližini anten, X-žarki in visoke napetosti. *Izpostavljeni so bili tudi vplivom generatorjev in* napetostnih tuljav v avtomobilih ter jederskim sevanjem, toda podatki so bili nespremenjeni. Prav tako na magnetne medije ne vpliva svetlobna energija, celo v laserski obliki ne. Šele ko je bil laserski žarek fokusiran, je seveda nastala poškodba zaradi toplotnega učinka. Magnetni mediji prenašajo tudi velike pritiske (900 kg so dali na magnetne kartice) ter vročino. Obdržijo namreč podatke celo do 675^{O} C - Curijeve temperature. Celo če so bili magnetni trakovi izpostavljeni temperaturi -51^{O} C za 24 ur, so jih zlahka prebrali brez izgub podatkov, potem ko so jih odtajali.

V raziskovalnem centru NASA v Langley-u preskušajo prototip mehurčnega pomnilnika. Celico pomnilnika so razvili v Rockwell International's Automatics Group in vsebuje več kot 1,6 *milijonov* bitov. *Za pomnilnik* bodo uporabili 64 celic in naj bi bil pripravljen za vesoljske polete v naslednjem letu. Za te so mehurčni pomnilniki bolj privlačni kot elektromehanski zaradi daljše življenjske dobe in večje zanesljivosti. Če jih primerjamo z običajnimi magnetno tračnimi pomnilniki, je njihov srednji čas med odpovedmi dva do trikrat daljši. Ocena za srednji čas odpovedi znaša 40000 ur. Sama mehurčna celica meri 3,81 krat 7,62 krat 1,27 centimetrov in sestoji iz para plošč tiskanega vezja. Ne vedo še cene tovrstnih pomnilnikov, toda v NASA in v Rockwell menijo, da bodo precej cenejši od tračnih pomnilnikov. Kljub odpovedi posameznih celic pomnilnik še vedno deluje. Povprečni dostopni čas registracije je 0,5 sek. Napolnimo oziroma preberemo ga v eni sekundi.

V Telesensory Systems, Inc. of Palo v ZDA so razvili kalkulator za slepe, ki potrjuje vnesene podatke in podaja rezultat v govorni obliki. To je omogočeno z integriranim vezjem za sintetiziranje človeškega glasu, ki so ga izdelali v Silicon Systems.

Novi mikro računar koji može biti upotrebljen kao neodvisni računar ili kao periferni procesor za računar GI 16-bitni CP 1600 će razviti General Instrument's European Centar u Škotskoj. Potrebne su neznatne promijene na čipu koje ga prilagodjivaju za jednu od tih primjena. Čip sadrži 512 12-bitnih lokacija ROM memorije, dvonivojni stek, 8-bitna registra, tri I/O kanala te clock oscilator. Sve I/O linije su TTL kompatibilne.
Skup instrukcija tog mikro računara je usmerjen na manipulaciju sa bitima te na prenos medju registrima. Svaki bit nekog od 32 registra sadržanih u čipu može biti setiran ili resetiran posebnom neredbom. Isto tako svaki bit može biti testiran na "1" ili "0" tako da se sljedeća naredba preskoči u slučaju "pravilno".

Nova prekidačka tehnika za tastaturni sistem RS 76 firme RAFI (Ravensburg - D 7980, Zapadna Njemačka), bazira na Halovom efektu što mu osigurava visoku pouzdanost. Cijena je za sada oko 60% viša od klasičkih sistema.

Čip TMS 9900 firme Texsas Instruments je iznenadjujuće močan, brz i fleksibilan 16-bitni mikro procesor. Njegov repertoar različitih instrukcija i prekidne karakteristike obezbedjuju mogućnosti računanja , kakve obično povezujemo sa 16-bitnima TTL mini računarima.

WL-3DS Environmental System razvijen u Beukers Laboratories, Bohemia, NY, koji upotrebljava pet 6800 mikro procesora omogućava detalina atmosferska mjerenja. Mikro procesori su upotrebljeni za predprocesiranje meteoroloških parametara, prije no što su isti zapisani na magnetnu traku i procesirani na Nova mini računaru. Sistem neprekidno semplira brzinu i smjer vjetra, temperaturu, vlažnost i pritisak preko padajućih sondi koje se izbacuju iz aviona na odredženoj visini i slobodno padaju uz pomoć padobrana. Sonde se obično puštaju u pet-minutnim intervalima iz aviona iznad područja mjerenja. Beakers sistem određuje poziciju sondi te njihovu brzinu i smjer kretanja uz pomoć navigacijskih signala koji se obradžuju u mini računaru.

Intelov 8085 mikro računar je dobijen proširenjem popularnog 8080 mikro računara tako da ima kompatibilan bus sa ostalim 8080 komponentama. Isto tako su svi 8080 programi izvodljivi na novom sistemu bez prethodnog modificiranja. Glavna prednost novog sistema je u osjetnoj redukciji broja elemenata. Tipični 8085 sistem sa tri čipa može se zamjeniti i do više od 10 komponentni

8080 sistem. Brzina operiranja je 3 MHz umjesto 2 MHz. Sve komponente napajamo jednim izvorom sa +5 V, uključujući i izbrisive PROM-e (EPROM). Tipični tročipni sistem može biti sestavljen od 8085 CPU, 8155 2 K RAM I/O i tajmera te 8355 16 K ROM i I/O ili 8755 16 K EPROM i I/O. Vrijeme jednog cikla je 1.3 µS.

Mala 5,5 in. disketa, zvana Micro-Floppy, firme Wango Inc., 5404 Jandy Pl., Los Angeles, CA 90066, ima kapacitet od 498,6 hiljada bajtova. Kapacitet za neformatirane podatke je 109,4 hiljade bajtova na 35 tragova. Prosječno vrijeme traženja je 370 ms. Dimenzije diskete su 3,25 x 5,75 x 7,95 in.

Vrijeme pristupa Mostekovog 16-K RAM memorijskog čipa MK 4116 P-2 je 150 ns; vrijeme ciklusa je isto tako veoma malo i iznosi 375 ns. Taj dinamični RAM je smješten u 16-pinski DIP. 4116 je male snage (462 mW); tolerira izvore energije u granicama od 10% (+12, ± 5 V); zahtijeva 128 ciklova za osvježenje; sadrži adresni i podatkovni registar; uključuje dvije metode selektiranja čipa. Pored običnih ciklova za čitanje, pisanje te čitanje-modificiranje-pisanje, čip uključuje i ciklus pisanje sa zadrškom. Jedinica je sposobna operirati u "page" načinu što omogučava sukcesivne memorijske operacije na različitim adresama kolone (column adress) pri istim adresama reda (row adress) tako da se poveća brzina. Vrijeme pristupa u toku " page" operiranja je 100 ns.

Novi kompjutorizovani komandni policijski sistem u Glazgovu, koji pokriva područje od 5500 kvadratnih milja sa više od 2,5 miliona žitelja, zapošljava samo 10000 ljudi. Sistem smješten u ultramodernom kompleksu, je najsavršeniji u Evropi, ako ne i u svijetu.
Novi sistem pomaže u upravljanju i razvoju policijskih kapaciteta u Glazgovu, najvećem urbanom području Škotske.
Oko 800 incidenata i 300 raporta o kriminalnim akcijama dnevno se obraduje na sistemu. Iz različitih dijelova grada stižu raporti preko teleprinterskih linija. Sistem nudi online kontrolu nad prestrojavanjem raspoložljivih ljudi i sredstava. Kontrolori u centralnoj sobi su opremljeni sa dva terminala. Jedan je standardna vizualna jedinica sa ekranom na kojem se ispisuju detalji o incidentu. Na ekranu drugog terminala se avtomatično pokaže plan dijela grada sa kojeg je stigao raport. Svaka policijska mobilna jedinica je opremljena malim terminalom za unošenje podataka o statusu i poziciji jedinice u računarski sistem. Sistem sadrži i veliki laboratorij sa svim potrebnim uredjajima uključujući kompjutorizovani kromatograf za ispitivanje procenata alkohola u krvi, koji za samo tri minuta ispiše rezultate analize.

izdelki domačih tovarn

Kartonažna tovarna Ljubljana, Resljeva 14, si prizadeva svoj obsežni proizvodni program papirnatih **žvitkov** za telexe (na osnovi domačih papirjev) razširiti tudi na perforirane trakove, ki se uporabljajo za vpis in izpis podatkov v računalniški tehniki.

literatura in srečanja

JUNIJ 1977

junij, Chicago, ZDA

"DPMA 77" INTERNATIONAL DATA PROCESSING CONFERENCE AND BUSINESS EXHIBITION
Organizator in informacije: Data Processing Management Assoc. (DPMA), 505 Busse, Highway Park Ridge I 1160068 USA

junij, Beograd, Jugoslavija

2. SIMPOSIUM: MAŠINE, AUTOMATI, ROBOTI I MANIPULATORI
SECOND SYMPOSIUM: AUTOMATIC MACHINES, ROBOTS AND MANIPULATING MECHANISMS
Organizator in informacije: Jugoslovanski nacionalni komitet IFTO MM-a Ul. 27. marta br. 80, 11000 Beograd, tel. 329-212

1-6 junij, Bytom, Polska

IFAC WORKSHOP ON SYSTEM ANALYSIS APPLICATION
Organizator: IFAC, The Polish Academy of Sciences
Informacije: Dr. Krzysztof Cichocki, Programme Committee, Institute of Organization, Management and Control Sciences, Polish Academy of Sciences, 55 KRN St, 00-818 Warsaw, Poland

7-9 junij, Zürich, Švica

THIRD ANNUAL INTERNATIONAL SYMPOSIUM AND EXHIBITION ON MINI- AND MICROCOMPUTERS AND THEIR APPLICATION · MIMI 77
Organizator: ICORD, The International Society for Mini- and Microcomputers (ISMM)
Informacije: MIMI 77, P.O. Box 354, 8053, Zürich, Switzerland

9-10 junij, Helsinki, Finska

CHEMDATA 77, COMPUTER APPLICATIONS IN THE ANALYSIS OF CHEMICAL DATA AND PLANTS
Organizira: European Federationof Chemical Engineering
Informacije: Chemdata 77, P.O. Box 28, SF-00131 Helsinki.13, Finland

13-16 junij, Dallas, Texas, ZDA

NATIONAL COMPUTER CONFERENCE
Organizator: AFIPS, ACM, DPMA
Informacije: Dr. Robert R. Korfhage, Department of Computer Science, Southern Methodist University, Dallas TX 75275, USA

14-17 junij, Hague, Nizozemska

5th IFAC/IFIP INTERNATIONAL CONFERENCE ON DIGITAL COMPUTER APPLICATIONS TO PROCESS CONTROL
Organizator ininformacije: IFAC/IFIP 1977 c/o Klvl, 23 Prinsessegracht, The Hague, The Netherlands

14-17 junij, Gif sur Yvette, Francija

APPLICATION D'APL EN FRANCE
Organizator: AFCET, Division Theorie et Techniques de l'informatique
Informacije: AFCET, 156 Boulevard Pereire, 75017 Paris

22-24 junij Montreux, Švica

INTERNATIONAL SYMPOSIUM ON SIMULATION
Organizator: ICORD, The International Association of Science and Technology for Development, IACSS, SEV, SVOR
Informacije: Chairman of the Symposium Simulation 77 P.P. Box 354, 8053 Zürich, Switzerland

28-30 junij, Los Angeles, California, ZDA

1977 INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING-FTCS-7
Organizator in informacije: The Institute of Electrical and Electronics Engineers, Computer Society, USA

28 junij-1 julij Coventry, Velika Britanija

2nd IFAC/IFIP SYMPOSIUM "CONTROL OF DISTRIBUTET PARAMETER SYSTEMS"
Organizator in informacije: IFAC/IFIP TC 7

JULIJ

1-8 julij, Fredericton, Canada

FOURTH IFAC SYMPOSIUM ON MULTIVARIABLE TECHNOLOGICAL SYSTEMS
Organizator: The National Research Council of Canada, The Associate Committee on Automatic Control and The University of New Brunswick
Informacije: The Secretary, IFAC MVTS Symposium, Electrical Engineering Department, University of Brunswick, Frederickton, New Brunswick, Canada E3B 5A3

4-8 julij, Brisbane, Australia

15th INTERNATIONAL SYMPOSIUM ON THE APPLICATION OF COMPUTERS AND MATHEMATICS IN THE MINERAL INDUSTRIES
Organizator in informacije: Sec. Australasian Inst. of Mining and Metallurgy, P.O. Box 310, Carlton South, Victoria 3053, Australia

12-16 julij, Barcelona, Španija

FIRST WORLD CONFERENCE ON MATHEMATICS AT THE SERVICE OF MAN
Organizator: Instituto de Matemática Apliada
Informacije: c/o Mrs Roser Luch, Callcerola, 25 Barcelona -6, Spain

26-29 julij, Grandfield, Bedfordshire, Velika Britanija

SIX INTERNATIONAL CONFERENCE ON MECHANIZED INFORMATION STORAGE AND RETRIEVAL SYSTEMS
Informacije: Cyril Cleverdon, Librarian, Granfield Institute of Technology, Granfield, Bedford MK 43 OAL Great Britain

AVGUST

1-5 avg., Kyoto, Japonska

IFAC SYMPOSIUM ON ENVIRONMENTAL SYSTEMS PLANNING DESIGN AND CONTROL
Organizator: IFAC Environmental Systems Symposium Committee on Systems Engineering and Applications
Informacije: The Secretariat of IFAC Environmental Systems Symposium c/o prof. Y.Sauaragi, Department of Applied Mathématics and Phisycs, Faculty of Engineering, Kyoto University, Kyoto 606, Japan

2-5 avg. Waterloo, Ontario, Kanada

THIRD INTERNATIONAL CONFERENCE ON COMPUTING IN THE HUMANITIES
Organizator: University of Montreal and University of Waterloo
Informacije: Prof. J.N.North, Department of English University of Waterloo, Waterloo, Ontario, Canada N 2 1 361

8-12 avg. Toronto, Kanada

MEDINFO 77
Organizator: IFIP TC 4 (Medicine)
Informacije: MEDINFO 77, Dr.J.Brandejs, The Canadian Medical Association, P.O. Box 8650, Ottawa, Ontario, Canada

8-12 avg. Toronto, Kanada

IFIP CONGRES
Organizator: IFIP
Informacije: Mr.J.H. Finch, Canadian Information Processing Society, 212 King street West, 501, Toronto, Ontario, Canada M5H 1K5

15-16 avg. Rochester, New York, ZDA

SYMPOSIUM ON ARTIFICIAL INTELLIGENCE AND PROGRAMMING LANGUAGES
Organizator: ACM SIGART, SIGPLAN
Informacije: Dr. Vincent Suyer, Director, Computing Centre, University of Rochester, Rochester, NY 14627 USA

15-19 avg. Binghamton, New York, ZDA

CONFERENCE ON APPLIED GENERAL SYSTEMS RESEARCH: RECENT DEVELOPMENTS AND TRENDS
Informacije: Dr.G.Kliz, School of Advanced Technology State University of New York at Binghamton, Binghamton, NY 13901, USA

16-18 avg. Yorktown Heights, ZDA

3rd INTERNATIONAL SYMPOSIUM ON COMPUTER PERFORMANCE MODELING, MEASUREMENT AND EVALUATION
Organizator in informacije: IFIP WG 7.3

22-25 avg. Bangkok, Tajska

INTERNATIONAL CONFERENCE ON COMPUTER APPLICATIONS IN DEVELOPING COUNTRIES
Organizator: Regional Computer Centre, Asian Institute of Technology
Informacije: Dr. Fook Loy Ng, Secretary ICCA, Asian Institute of Technology, P.O. Box 2754, Bangkok, Thailand

avgust-september, Pisa, Italija

4th INTERNATIONAL SUMMER SCHOOL "COMPUTATIONAL AND MATHEMATICAL LINGUISTICS"
Organizator in informacije: Prof. Zampolli, Director of

the International Summer School CNUCE - Via S. Maria 36, 56100 Pisa, Italia

SEPTEMBER

5-9 sept. Bergamo, Italija

CONFERENCE ON EVALUATION AND PLANNING OF INTERPERSONAL TELECOMMUNICATION SYSTEMS
Informacije: Dr. M.Elton, Joint limit for Planning Research, Communications Studies Group, Wates House, 22 Gordon Street, London, WCIH, /QB, United Kingdom

7-9 sept. Rocquencourt, Francija

DATA AND INFORMATICS ANALYSIS
Organizator: IRIA, Domaine de Voluceau, Rocquencourt, 78150 Le Chesnay, France
Informacije: Secrétatiat des Journées, Service de Relations Extérieures, IRIA, Domaine de Voluceau, Rocquencourt, 78150 Le Chesnay, France

12-16 sept. Leipzig, NDR

IFAC SYMPOSIUM ON CONTROL MECHANISMS IN BIO- AND ECOSYSTEMS
Organizator in informacije: IFAC, EKONO, P.O. Box 27 SF- 00131 Helsinki, Finland

19-23 sept. Prague, Češkoslovaška

COMPUTER APPLICATIONS TO DISCRETE MANUFACTURING
Organizator in informacije: IFAC, Symposium on Discontinous Computer Control System, Organizing Committee, Suchbartova 4, 160 00 Prague 6-Dejvice, Czechoslovakia

19-23 sept. Paris, Francija

CONVENTION INFORMATIQUE
Organizator: SYNTEC, INFORMATIQUE
Informacije: Convention Informatique, 6 Place de Valois 7500, Paris, France

26-28 sept. Toronto, Kanada

INTERNATIONAL ELECTRICAL, ELECTRONICS CONFERENCE AND EXHIBITION
Organizator in informacije: Institute Electrical and Electronics Engineers Technical Activities Board, 345 East 47th Street, New York, NY 10017, USA

sept. - 1 okt. Chicago, ZDA

AMERICAN SOCIETY FOR INFORMATION SCIENCE, ANNUAL MEETING
Organizator in informacije: ASIS, 1155, Sixtenth Street, N.W. Washington, D.C. 2036 USA

OKTOBER

oktober, Beograd

NOVI PRAVCI RAZVOJA HIBRIDNIH RAČUNARSKIH MAŠINA
NEW TRENDS OF HYBRID COMPUTERS DEVELOPMENT
Organizator in informacije: Institut Mihajlo Pupin, Volgina 15, 11000 Beograd

oktober, Beograd, Jugoslavija

SEMINAR O INDUSTRIJSKIM ROBOTIMA I MANIPULATORIMA
INDUSTRIAL ROBOTS AND MANIPULATORS
Organizator in informacije: Institut "Mihajlo Pupin", Volgina 15, 11000 Beograd

oktober, Bled, Jugoslavija

SIMPOZIJUM O SASTAVNIM DELOVIMA
SYMPOSIUM ON THE COMPONENTS
Organizator in informacije: Jugoslovenski komitet za ETAN, Kneza Miloša 9 11000 Beograd, telef. 011/ 33 957

3-5 okt. Bonn, ZRN

THIRD INTERNATIONAL SYMPOSIUM ON MODELING AND PERFORMANCE AND EVALUATION COMPUTER SYSTEMS
Organizator: GMD/IRIA/IFIP
Informacije: Mr. P.Hayderhoff - G.M.D. Postfach 1240 d-5205 St. Augustin, Germany

3-6 okt. Amsterdam, Nizozemska

EUROMICRO 3rd Symposium on Microprocessing and Microcomputing
Organizator: EUROMICRO
Informacije: J.D. Nicoud LCD-EPFL Bellerive 16, CH-1007 Lausanne, Switzerland

3-8 okt. Bled, Jugoslavija

SIMPOZIJ IN SEMINARJI INFORMATICA 77
Organizator in informacije: Slovensko društvo Informatica, Jamova 39, 61000 Ljubljana, Jugoslavija

6-8 okt. Tokyo, Japonska

THIRD INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES
Organizator in informacije: IFIP

6-12 okt. Düseldorf, ZRN

INTERKAMA 1977
INTERNATIONAL EXHIBITION OF MEASURING TECHNIQUES AND AUTOMATICS, CONFERENCE
Organizator in informacije: Düseldorfer Messegesellschaft m6H-NOWEA 4 Düseldorf 30, Postfach 320 203 BRD

17-19 okt. Seattle, Washington, ZDA

ACM
Organizator: ACM
Informacije: ACM Headquarters, 1133 Avenue of the Americas, New York, NY 10036, USA

17-20 okt. Tokyo, Japonska

IFAC SYMPOSIUM ON INFORMATION-CONTROL PROBLEMS IN MANUFACTURING TECHNOLOGY
Organizator: IFAC Manufacturing Technology Symposium c/o The Society of Instrument and Control Engineers 39 Shiba Kotohira-Cho Minata-Ku, Tokyo, Japan

17-21 okt. München, ZRN

INTERNATIONAL SEMINAR AND PROFESSIONAL EXHIBITION - COMPUTER SYSTEMS AND ITS APPLICATIONS
Organizator in informacije: Münchner Messe und Ausstellungsgeselshaft mbH 8 Postfach 121 009 , FRG

19 okt. Herceg Novi, Jugoslavija

SIMPOSIJ O PRIMENI OPERACIONIH ISTRAŽIVANJA
Organizator in informacije: Fakultet organizacionih nauka (biblioteka), 11091 Beograd, Ul. Oslobodjenja 1

NOVEMBER

22-24 nov. Versailles, Francija

MODELISATION ET MAITRISE DES SYSTEMS TECHNIQUES,

ECONOMIQUES, SOCIAUX
Organizator in informacije: AFCET, 156, Bulevard Pereire, 75017 Paris, France

26-29 nov. Cairo, Egipt

SECOND IFAC INTERNATIONAL CONFERENCE ON SYSTEMS APPROACH FOR DEVELOPMENT
Organizator: IFAC
Informacije: Eng. Sayed Abdel Kader El Sheshe Secretary of Egyptian High Committee of Automatic Control (EHCAC), 6 Khalil Agha Street, Garden City, Cairo, Egypt

28-30 nov. Amsterdam, Nizozemska

CONFERENCE ON DATA MANAGEMENT TECHNOLOGY-LATEST METHODOLOGY IN DATA SYSTEMS
Organizator in informacije: IAG, IFIP, The Netherlands

DECEMBER

5-9 dec. Paris, Francija

3rd INTERNATIONAL SYMPOSIUM ON COMPUTING METHODS IN APPLIED SCIENCE AND ENGINEERING
Organizator in informacije: IFIP WG 7.2 (co-sponsor)

5-11 dec. Gaithersburg, Maryland, ZDA

WINTER SIMULATION CONFERENCE
Organizator: National Bureau of Standards
Informacije: Dr.J. William Schmidt (Prog. Chm), Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 - USA

LETO 1978

21-23 feb. London, Velika Britanija

FIRST EUROPEAN CONFERENCE ON PRAGMATIC PROGRAMMING AND SENSIBLE SOFTWARE
Informacije: Jerry Weinberg, Online, Cleveland Road Uxbridge UB8 2DD, England

7-10 marec, Paris, Francija

INTERNATIONAL CONGRESS ON THE CONTRIBUTION OF COMPUTERS TO THE DEVELOPMENT OF CHEMICAL ENGINEERING AND INDUSTRIAL CHEMISTRY
Informacije: Monsieur Le Professeur A. Brusset, Congrès International 1978, Societé de Chemie Industrielle 28, rue Saint Dominique F-75007 Paris, France

27-30 marec, New York, ZDA

INTERNATIONAL CONVENTION AND EXHIBITION OF THE INSTITUT OF ELECTRONICAL AND ELECTRONICS ENINEERS
Organizator: IEEE
Informacije: J.H. Shumacher, IEEE, 345 East, 47th Street, New York, NY 10017, USA

Maj

4th INTERNATIONAL COMPUTERS AND COMPUTING CENTERS, EQUIPMENT EXHIBITION
Organizator: Beogradski sajam

11-17 junij Helsinki, Finska

IFAC CONGRESS
Organizator: Finish Society of Automatic Control
Informacije: Mr. Olli Pezolanti, Höyläämotie 18, 00380 Helsinki 38, Finland

7-10 nov. Kyoto, Japonska

4th INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION
Informacije: Prof. T. Sakai, Kyoto University, Department of Information Science, Kyoto, Japan

21-23 junij Toulouse, Francija

1978 INTERNATIONAL SYMPOSIUM ON FAULT TOLERANT COMPUTING-FTCS-8
Organizator: FTC Technical Committee of the Institute for Electrical and Electronics Engineers Computer Society
Informacije: IEEE, 345 East 47th Street, New York, NY 10017, USA

11-15 junij Prague, Češkoslovaška

IFAC/IFIP 2nd INTERNATIONAL SYMPOSIUM "SOFTWARE FOR COMPUTER CONTROL"
Organizator in informacije: IFAC/IFIP TC 5

naslovi

Za tiste, ki imajo mikro računalnike in sploh računalnike za svoj konjiček, smo pripravili nekaj naslovov klubov iz ZDA. Na vaša vprašanja s področja programske in materialne opreme ter računalniške tehnike vam bodo člani teh klubov radi odgovorili.

Bay Area Microprocessor Users Group
4565 Black Avenue
Pleasanton
CA 94 566

Computer Org. of L.A. (COLA)
Box 43677
Los Angeles
CA 90043

Homebrew Computer Club
Box 626
Mountain View
CA 94040

Litton Calculator/Computer Club
MS 78/31
5500 Canoga Avenue
Woodland Hills
CA 91364

Miami Computer Club
John Lynn
13431 S.W. 79th
Miami
FL 33138

Sacramento Minicomputer Users Group
Box 741
Citrus Heights
CA 95610

San Gabriel SCCS
c/o Dan Erickson
400 S.Cataline Avenue
Pasadena
CA 91106

Southern California Computer Society
Box 3123
Los Angeles
CA 90051

Denver Amateur Computer Society
Box 6338
Denver
CO 80206

autorji in sodelavci

DUBRAVKA ĆEĆEZ-KECMANOVIĆ (1946), završila Elektrotehnički fakultet u Sarajevu 1970 godine i magistrirala na Centru za multidisciplinarne studije, Univerzitet u Beogradu 1974 godine. Radi kao asistent na Elektrotehničkom fakultetu u Sarajevu. Težište dosadašnjeg rada bilo je iz oblasti razvoja informacionih sistema, sada se bavi metodijskim pitanjima razvoja informacionih sistema, posebno metodama analize objektnog sistema, informacione analize i vrednovanja informacionih sistema. U našom časopisu vodi područje Informacioni sistemi.

MATIJA EXEL (1947), diplomiral za inženirja računalništva na Univerzi v Ljubljani leta 1970, doktorat 3-ga cikla univerze v Grenoblu dosegel v letu 1975. Zaposlen na Institutu Jožef Stefan v Ljubljani. Ukvarjal se je predvsem s programskimi jeziki (podatki in podatkovne strukture) in prevajalniki. Sedanja področja udejstvovanja so predvsem: principi operacijskih sistemov, paralelno programiranje ter metodologija sistemov programske opreme. Je sodelavec našega uredništva in vodi področje Operacijski sistemi.

SEAD MUFTIĆ (1948), diplomirao 1971 godine na odsjeku za matematiku Prirodno-matematičkog fakulteta u Sarajevu, magisterij (1974 godine) i doktorat (1976 godine) dosegao na Ohio State University. Predavač je na drugom stupnju Prirodno-matematičkog fakulteta u Sarajevu te na podiplomskom studiju, Odsjek za informatiku Elektrotehničkog fakulteta u Sarajevu. Zaposlen je u Zavodu za ekonomsko planiranje Sarajevo. Glavne oblasti naučnog rada su, prije svega, Sigurnost kompjutorskih sistema, Baze podataka, Kompjutorske mreže, Projektovanje kompjutorskih sistema te Informacioni sistemi. Društveno je aktivan i nosioc brojnih nagrada.

RADO FALESKINI (1944), diplomiral za inženirja elektrotehnike na Fakulteti za elektrotrotehniko v Ljubljani leta 1970. Zaposlen v računskem centru Univerze v Ljubljani, kjer se v glavnem ukvarja s vprašanji planiranja, projektiranja in investiranja ter z ekonomsko in organizacijsko problematiko. V okviru dosedanjega strokovnega udejstvovanja je omeniti tudi delo na problemih prenosa podatkov, telegrafijo ter problematiko frekvenčno multipliciranih žičnih in radijskih zvez.

DŽENAN RIPANOVIĆ (1953), diplomirao na Elektrotehničkom fakultetu u Sarajevu, Odsjek za informatiku 1976 godine. Zaposlen je u Računskom centru Privredne banke Sarajevo. Dosadašnje i trenutno područje njegovog rada su sistemi za manipulisanje (upravljenje) bazama podataka. Tema diplomskog rada bila je: Realizacija hijerarhijskog sistema za manipulisanje bazom podataka zasnovanog na B-drveću.

DUŠAN KODEK (1946), je diplomiral na Fakulteti za elektrotehniko v Ljubljani leta 1970, na isti je v letu 1975 promoviral za doktorja elektrotehniških znanosti. Zaposlen je kot vodja računskega centra na Fakulteti za elektrotehniko v Ljubljani. Pri svojem dosedanjem delu se je zlasti ukvarjal s projektiranjem sistemov za prenos podatkov in načrtovanjem in realizacijo raznih digitalnih naprav, z digitalnimi filtri ter mikro računalniki. Vodi organizacijo seminarjev za dopolnilno izobraževanje v računalništvu.

I N F O R M A T I C A 77
Bled, 3.-8. oktober 1977

simpozij
12. jugoslovanski mednarodni simpozij o obravnavanju podatkov
Bled, 3.-8. oktober 1977

seminarji
izbrana poglavja računalniških znanosti
Bled, 4.-7. oktober 1977

razstava
računalniška oprema in literatura
Bled, 2.-8. oktober 1977

organizator:
Slovensko društvo Informatica v sodelovanju z Institutom Jožef Stefan in Fakulteto za elektrotehniko, Ljubljana

roki
30. maj 1977 - zadnji rok za sprejem formularja s prijavo in 2 izvodov razširjenega povzetka
30. junij 1977 - pošiljanje rezultatov recenzije in avtorskega kompleta
15. avgust 1977- zadnji rok za sprejem končnega teksta prispevka

nadaljnje informacije:
INFORMATICA 77
Institut Jožef Stefan, 61001 Ljubljana, pp 199
telefon: (061) 63 261, telegram: JOSTIN Ljubljana
telex: 31 296 YU JOSTIN

S tradicionalnim posvetovanjem "Simpozij in seminarji INFORMATICA" slovensko društvo Informatica nadaljuje aktivnost Zveznega strokovnega odbora za obravnavanje podatkov pri Jugoslovanskem komiteju za ETAN.
To posvetovanje je postalo tako po udeležbi kot po tehtnosti objavljenih del osrednje jugoslovansko srečanje teoretikov in praktikov s področja obravnavanja podatkov. Lanskemu simpoziju, na katerem je bilo predstavljenih 278 tujih in domačih del, je prisostvovalo 459 strokovnjakov. Zaradi hitrega vzpona znanstvenih in strokovnih računalniških moči v Jugoslaviji in sosednjih deželah pričakujemo, da bo na letošnjem simpoziju in seminarjih sodelovalo še večje število predavateljev in poslušalcev. Tudi letošnji simpozij bo mednarodnega značaja. V vabljenih uvodnih predavanjih in na seminarjih, ki bodo potekali vzporedno s simpozijem, bodo priznani tuji in domači strokovnjaki pregledno predstavili najnovejše dosežke iz izbranih področij računalništva in informatike.
Za naše strokovnjake bo simpozij priložnost, da v teku uradnega dela simpozija v neformalnih srečanjih podvržejo svoja dognanja kritični oceni priznanih tujih in domačih strokovnjakov. Upravičeno smemo pričakovati, da bo simpozij pomemben prispevek k medsebojnemu povezovanju in izmenjavi izkušenj na področju računalništva.

Vabimo vas, da se aktivno udeležite letošnjega posvetovanja.

jezik simpozija
Angleščina je uradni jezik simpozija. Zaradi mednarodnega značaja simpozija vabimo avtorje, da predložijo in predstavijo svoja dela v angleškem jeziku. Seveda bodo v program uvrščena tudi dela, napisana v kateremkoli izmed jugoslovanskih jezikov.

zbornik del
Vsak udeleženec bo prejel zbornik del ob prihodu na simpozij, vsak avtor bo poleg tega prejel še pet kopij svojega prispevka.

prijavnina
Za simpozij: 1.800 din
Za seminarje: 400 din (velja za cikel treh seminarjev)
Za študente velja 50% popust
Avtorji plačajo enak znesek kot drugi udeleženci.

I N F O R M A T I C A 77
Bled, October 3-8, 1977

Symposium
12th Yugoslav International Symposium on Information Processing
Bled, October 3-8, 1977

Seminars
Selected topics in Computer Science
Bled, October 4-7, 1977

Exhibition
Computer Equipment and Literature
Bled, October 2-8, 1977

Organizer
INFORMATICA, Slovene Computer Society in Co-operation with Jožef Stefan Institute, and Faculty of Electrical Engineering, University of Ljubljana

Deadlines
May 30, 1977 - submission of the application form and 2 copies of the extended summary,
June 30, 1977 - mailing out of the summary reviews and author kits,
August 15, 1977 - submission of the full text of contribution.

Further Information
INFORMATICA 77
Institut Jožef Stefan, 61001 Ljubljana, pp 199, Jugoslavija
Phone: (061) 63 261, Cable: JOSTIN Ljubljana,
Telex: 31 296 YU JOSTIN

With the traditional conference "Symposium and Seminars INFORMATICA" the Slovene Computer Society Informatica carries on the tradition of the Federal Professional Committee for Information Processing at the Yugoslav Committee ETAN. This conference has become a major Yugoslav meeting of professionals in the field of information processing. Last year, 459 experts from Yugoslavia and abroad attended the meeting and presented 278 papers. In view of the recent developments of scientific and technical computing in Yugoslavia and neighbouring countries, an even greater numbers of authors and other participants is expected to attend the symposium and seminars this year. As in previous years, this will be an international symposium. In their invited papers and at the seminars, eminent foreign experts will present surveys of the latest achievements in selected fields of computer science in informatics.
The present symposium will also provide Yugoslav experts with an opportunity to submit their achievements to critical appraisal by eminent foreign and Yugoslav experts during scientific sessions and informal meetings. We believe that this symposium will be a significant contribution to mutual cooperation and exchange of experience in the field of information processing.

You are invited to take part in the symposium

Language
English is the official language of the symposium. In view of the international character of the symposium, authors are invited to write and present their contributions in English. However, papers written in any of the Yugoslav language will also be included in program.

Proceedings
Participants will receive the proceedings upon arival. Each autor will also receive five copies of his paper.

Registration Fee
For symposium: 1.800 din
For seminars: 400 din (for cycle of three seminars)
Student registration will be half price.
Authors pay full registration fee.