Š O L S K I C E N T E R P T U J
Poklicna in tehniška elektro šola
Volkmerjeva 19, 2250 Ptuj, ☎ (02) 772-4411, fax: (02) 776-2021

Osnove mikroračunalnika

UČBENIK INTERNO UPORABO

Predmet: DSK4,DIS4,RSM3
ELEKTROTEHNIK ELEKTRONIK,RAČUNALNIŠKI TEHNIK

Pripravil: Rudolf WEINZERL dipl.inž.el. ,Maribor, 2001

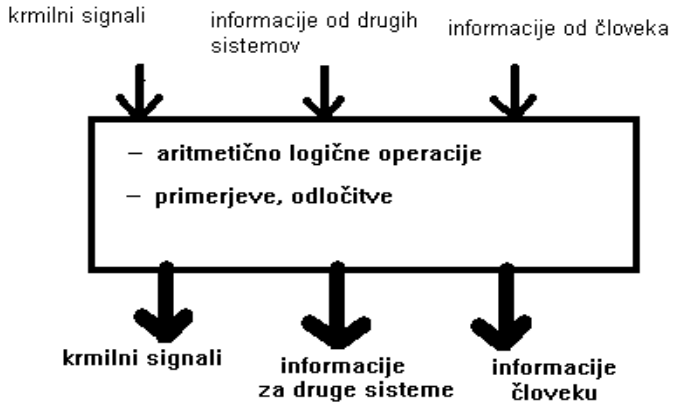
dodatke iz skripte Matjaža Colnariča:Mirkoračunalniki in drugih virov
zbral Slavko Murko

1. Uvod

1.1. Uporabnost računalnika

Računalnik si lahko predstavimo kot črno skrinjo, ki je inteligentni del med vhodnimi in izhodnimi podatki. Od tako predstavljene črne skrinjice zahtevamo:

- da ima procesni del, ki lahko opravi aritmetične in logične operacije
- da ima sklop, ki je sposoben izvajati primerjave in odločitve
- da ima svoj pomnilnik za shranjevanje programa, operandov in rezultatov



Slika 2

Orientacija / člen	Vhodni člen	izhodni člen
Človeku	Tipkovnica, miška, preklopnik, skener	monitor, zvočnik, prikazovalnik
Drugim sistemom	Modem, disketa, kaseta, CD	modem, disketa, kaseta, CD
Tehnološki okolici	Senzor, merilnik, A/D pretvornik	rele, D/A pretvornik

Delitev informacijski, krmilni:

Na primer:

Krmilniki in mikroračunalniki, ki v proizvodnem obratu krmilijo stroje in proizvodne linije spadajo med **krmilne mikroračunalnike.**.

PC-ji ki pa v pisarni pomegajo tajnici pri vodenu poslovanja ter v razvnem oddelku pri razvoju novih izdelkov pa spadajo med **informacijske računalnike.**

Pogosto se tudi za krmilne mikroračunalnike uporabijo PC-ji z ustreznimi programskimi orodji in krmilnimi vmesniki.
Lokalne računaniške mreže pa omogočajo povezavo informacijskih in krmilnih računalnikov v CAD/CAM sisteme.(Computer Aided Desigm/Computer Aided Manufactury) ter še naprej v sisteme za menedžersko spremljanje proizvodnje.

1.2. Zgodovinski razvoj mikroprocesorjev

Računalnik je bil najprej stroj za računanje. Poznali so mehanske izvedbe (zobniki), izvedbe z elektromehanskimi stikali (releji). Z uveljavitvijo elektronike (tranzistorji,čipi) pa je računalnik postal sposoben avtomat , ki je polek računskega stroja uporaben tudi kot pisalni stroj, risalna deska, igralni avtomat, industrijski krmilnik itd.

Osnovni del računalnika je mikroprocesor. Pobuda za razvoj prvega mikroprocesorja je prišla iz Japonske. Sredi leta 1969 je japonska firma Busicom, ki je izdelovala namizne kalkulatorje in je kasneje propadla, ponudila firmi Intel pogodbo za razvoj družine tiskanih vezij. Slednjo bi naj uporabili pri izdelavi programirljivih kalkulatorjev.

Izdelali so prvi mikroprocesor z oznako 4004, ki je bil 4 biten. Pojav tega mikroprocesorja pa še ni povzročil velikega zanimanja na tržišču. Skoraj istočasno je **Intel** razvijal močnejši 8 bitni procesor z oznako 8008, ki je prišel na tržišče 1974 leta.

Leta 1976 je **Zilog** izdelal znani Z-80. Tega leta je bilo na trgu že 54 različnih mikroprocesorjev. Danes verjetno nihče ne ve, koliko jih je.

2. KRMILNI MIKRORAČUNALNIK

2.1 Programirljiv avtomat

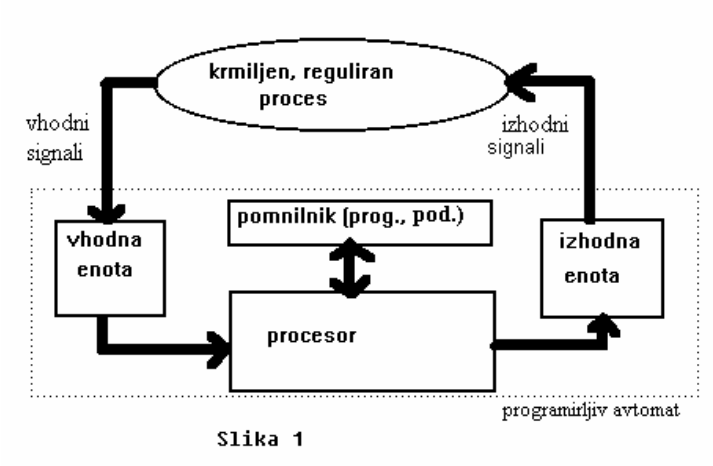
Njegova funkcija je nadomeščanje človeka pri krmijenju in vodenju strojev in proizvodnih procesov v industriji.

Starejša krmilje s trajno ožičenim programom: Krmilja, ki so sestavljena iz osnovnih in sestavljenih logičnih elementov imajo te elemente med seboj povezane po načrtu, ki smo ga dobili na osnovi pravilnostne tabele ali logične funkcije. Medsebojna povezava elementov krmilja je zato zasnovana na programu, ki ga naj to krmilje opravlja. Pravimo, da je program delovanja določen z ožičenjem.

Programirljivi avtomati Pri obsežnih krmiljih za vodenje in nadzor industrijskih procesov pa moramo ponavadi spremeniti program delovanja že v času preizkušanja. Včasih se tudi tehnološki postopek pogosto spreminja in zahteva novi program. Zato prihaja do zahtev po večji prilagodljivosti krmilja različnim potrebam. Takšnim krmiljem bi lahko spreminjali program ne da bi morali spreminjati ožičenje.

Takšni krmilni sistemi so sestavljeni iz aparaturne ali strojne opreme -- **hardware** (merilni pretvorniki, logična vezja, releji, stikala, ožičenje itd.) in iz programske opreme **software** (program delovanja krmilja, ki ga vpišemo v programski del pomnilnika). Sprememba programa delovanja

zahteva le vpis novega programa v programski pomnilnik, medtem ko spremembe v strojni opremi praviloma niso potrebne.



Slika 1

V krmiljenem oz. reguliranem procesu nam dajalniki signalov (senzorji) dajejo podatke o stanju in vrednosti posameznih veličin (temperatura, tlak, hitrost, pretok, pozicija itd.). Ti podatki se v vhodni enoti predelajo v binarne vrednosti, ki jih procesor lahko obdela. Procesor odčituje v skladu s programom posamezne vhodne podatke in ugotavlja stanja v procesu. Na osnovi stanj v procesu in programa daje procesor preko izhodne enote ukaze v obliki napetostnih impulzov na izvršilne člene krmilja (releji, svetlobni in zvočni javljalniki, prikazovalniki, servo motor, koračni motor itd.).

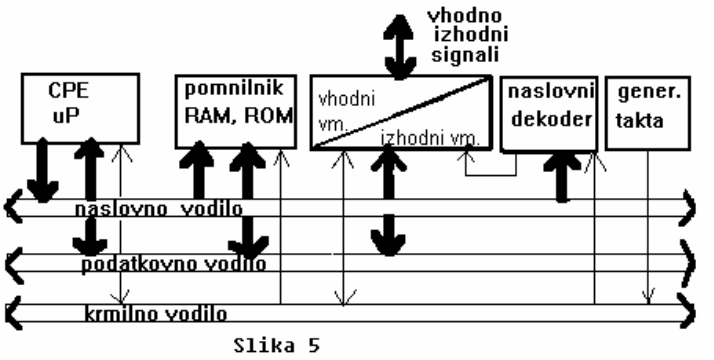
S takšnim programirljivim avtomatom lahko opravljamo naslednje procesne funkcije:
- Zajemanje in urejanje procesnih podatkov
- Izvajanje krmiljenja in regulacije
- Varovanje procesov v konfliktnih situacijah
- Analiza in ovrednotenje rezultatov procesa

2.2. Sestavimo mikroračunalnik

Sestavljajo v glavnem naslednje funkcijske enote:

- centralno procesna enota ali mikroprocesor
- pomnilnik (programski, podatkovni)
- vhodni in izhodni vmesnik

Te enote veže v funkcijsko celoto programska podpora v obliki programa vpisanega v programski pomnilnik.



Slika 5

Vse funkcijske enote so medsebojno povezane preko žic ki jim pravimo vodila (bus).

1. Naslovno vodilo
Preko tega vodila pošilja procesor naslov lokacije v pomnilniku ali naslov registra v vmesniku od koder želi dobiti informacijo ali jo tja odložiti. To vodilo je enosmerno in navadno sestavljeno iz signalov A0-A15. Govorimo o 16 bitnem naslovnem (adress) vodilu, ki lahko naslovi 2^{16} lokacij (65536 ali 64k).

2. Podatkovno vodilo
Služi za prenos podatkov med registri procesorja in pomnilnikom oz. vmesnikom. Je dvosmerno in običajno 8 bitno. Sestavljajo ga signali D0-D7 (data). Zaradi tega govorimo tudi o 8 bitnem uP. Podatki lahko zavzemajo 2^8 različnih vrednosti (0--255).

3. Krmilno vodilo
Po tem vodilu se prenašajo krmilni signali, ki npr. določajo vpisovanje ali čitanje podatkov in programa (RD, WR, R/~~W~~, PSEN), prekinitve delovanja uP (INT0, INT1), signali generatorja takta (XTAL, CLK), signal za reset itd..

Enote desno od CPE sestavljajo periferijo uR. Če nekaj od te periferije integriramo v uP dobimo mikrokontroler oziroma mikrokrmilnik.

2.3. Signali in vodila

Pod izrazom vodilo (bus) razumemo pri uP skupino signalov (žic), ki tvorijo neko celoto. Signali so dostopni preko nožic uP in tvorijo skupaj s povezovalnimi vodniki pot preko katere dobiva in oddaja uP informacije. Fizično je vodilo skupina žic, najpogosteje v obliki tiskanega vezja. Glede na to, ali je neka nožica uporabljena za en ali več signalov razlikujemo:

- navadne ali nemultipleksirane signale

- deljene ali multipleksirane signale

Primer:

Mnogi mikrokontrolerji imajo multipleksirane naslovne in podatkovne signale A0/D0 –A7/D7.

V enem časovnem intervalu je na nožici en signal, v drugem drugi, v tretjem tretji itd.. S posebnimi dodatnimi signali je v vsakem trenutku natanko določeno, kateri signal je trenutno na nožici. Nemultipleksirani signali so preprostejši za uporabo, uP opravila so hitrejša. Multipleksirani signali zahtevajo manjše število nožic, s tem pa zmanjšujejo ceno in fizično velikost uP.

2.3.1. Lastnosti signalov

Če želimo razumeti delovanje uP in sistemov zgrajenih na njegovi osnovi, moramo dobro poznati njegove signale. Signali imajo naslednje lastnosti:

Pomen signala je običajno podan z besedami oz. imenom in pove kako signal vpliva na logično delovanje uP ali kakega drugega elementa oz. kakšno vrsto informacije nosi. Tako razlikujemo naslovne, podatkovne, urine, prekinitvene, statusne in druge signale.

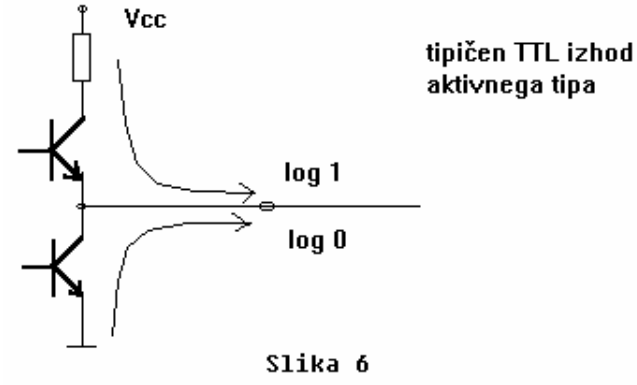
Smer signala gledamo s strani uP je lahko signal izhod (nosi informacijo iz uP) ali vhod (nosi informacijo v uP). Razen teh dveh možnosti poznamo tudi dvosmerne signale. To so tisti, ki so za uP v nekem trenutku izhod v drugem pa vhod.

Stanje signala. Vsak signal je lahko vsak trenutek v enem od dveh možnih stanj:

-- nizkem stanju (Low) , logična 0

-- visokem stanju (High), logična 1

Razen opisanih dveh stanj poznamo pri nekaterih signalih še tretje stanje. To stanje ni namenjeno prenosu informacije in ga imenujemo visoko impedančno stanje Z.

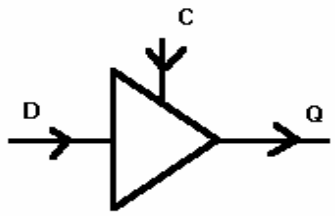


Slika 6

V tem stanju je signal električno prekinjen od nožice oziroma povezovalne žice (oba tranzistorja sta zaprta), njegov napetostni nivo pa nima logičnega pomena. Smisel tega stanja je v tem, da si en povezovalni vodnik lahko deli več signalov oz. vezij. Signale te vrste imenujemo tro-stanjske (TS, Tri-State).

2.3.2. Gonilniki s tremi stanji

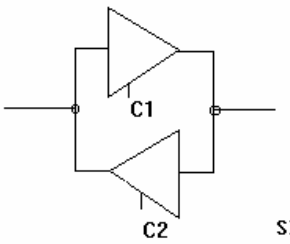
V primeru, ko želimo po eni liniji prenašati podatke v obe smeri, moramo na linijo priključiti logična vrata, s katerimi odrejamo smer pretoka podatkov. Takšna vrata so sestavljena iz dveh gonilnikov s tremi stanji -- tri state buffer .



D	C	Q
0	0	Z
1	0	Z
0	1	0
1	1	1

Slika 7

Kadar je na krmilnem vhodu C stanje 1, se podatki prenašajo od vhoda D na izhod Q. Ko pa je na krmilnem vhodu C stanje 0 se prenos podatkov blokira.



C1	C2	smer pretoka podat.
0	0	ni pretoka
0	1	←
1	0	→
1	1	nedovoljeno stanje

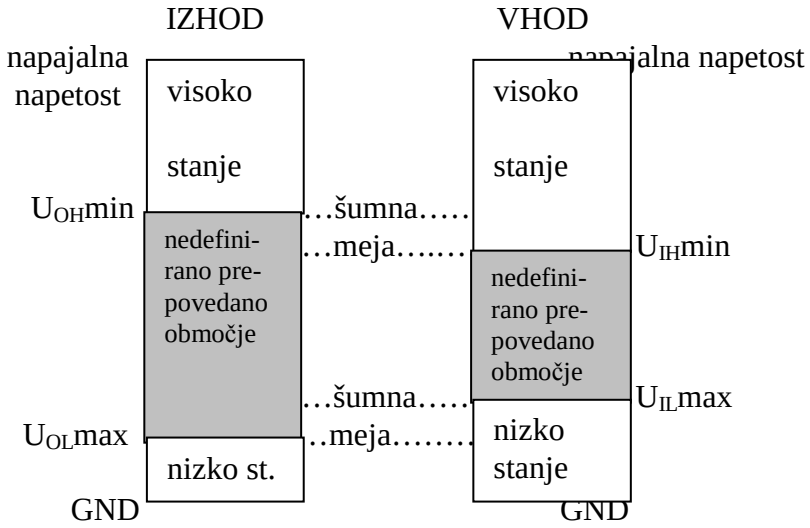
Slika 8

2.3.3. Električna predstavitev signala

Danes praktično vsi uP uporabljajo CMOS logične nivoje, ki se od TTL nivojev razlikujejo.
Oznaka TTL je kratica za bipolarno Transistor Transistor Logic družino digitalnih integriranih vezij, ki so se uveljavila v začeku 70 let. CMOS je oznaka za komplementarno MOS tehnologijo, ki uspešno zamenjuje NMOS tehnologijo, ker ima mnogo manjšo porabo in dosega vse večje hitrosti.
CMOS zahteva in daje višji nivo logične 1 in nivoji so odvisni od napajalne napetosti, medtem ko so garantirani TTL nivoji fiksni, ko je napajanje v okviru TTL specifikacij.

Napetostni nivoji so definirani posebej za izhode in posebej za vhode. Tako mora biti napetost na izhodu vezja v visokem stanju med U_{OH}min in napajalno napetostjo, medtem ko bi vhodno vezje pravilno prepoznalo napetost kot visoko stanje že, če je višja od

U_{IH}min. Razlika med tako definiranimi nivoji daje vezjem, ki jih uporabljajo, zaščito proti šumu.



Slika 9

U_{OH}min = minimalna vrednost izhodne napetosti za logični nivo 1
U_{IH}min = minimalna vrednost vhodne napetosti za logični nivo 1
U_{OL}max = maksimalna vrednost izhodne napetosti za logični nivo 0
U_{IL}max = maksimalna vrednost vhodne napetosti za logični nivo 0

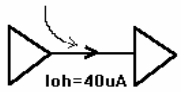
Primerjava logičnih nivojev:

Logično nivo	**Vcc=5V** 74HC	74HCT	LSTTL	8051	80C51
U_{OH}min	4.9V	4.9V	2.7V	2.4V	4.5V
U_{IH}min	3.5V	2.0V	2.0V	2.0V	1.9V
U_{OL}max	0.1V	0.1V	0.5V	0.45V	0.45V
U_{IL}max	1.0V	0.8V	0.8V	0.8V	0.9V

2.3.4. Standardno breme

Razumljivo je, da bo neko vezje vzdrževalo na izhodu visok nivo samo, če ga ne obremenimo preveč. Zato je v podatkih za vsak izhod podano maksimalno dovoljeno breme. Zaradi preprostosti pa pogosto merimo bremena s ti. standardnimi bremeni.

Primer:
Eno standardno TTL breme je tisto, ki v nizkem stanju pošilja v izhod 1,6mA toka, v visokem stanju pa sprejema iz izhoda 40uA toka.



Slika 10

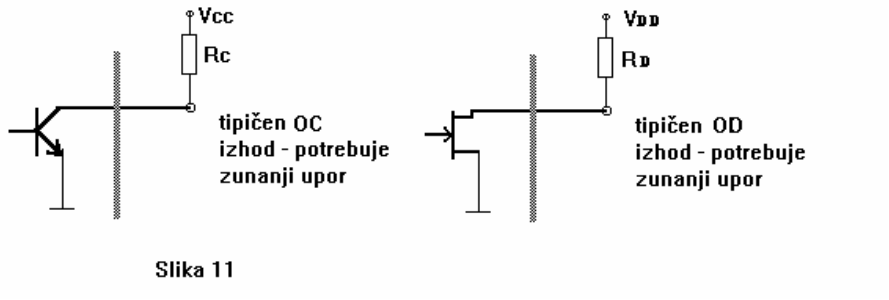
Vezje lahko v visokem stanju obremenimo z 200 uA, v nizkem pa s 4 mA. Določi število standardnih TTL bremen, ki jih lahko priklopimo na takšno vezje.

log1	*200 uA*	*5 standardnih TTL bremen*
log0	*4 mA*	*2 standardni TTL bremeni*

Odgovor: Priključimo lahko samo 2 standardni TTL bremeni.

2.3.5. Odprt izhod (open colector)

Večina IC vezij uporablja izhode aktivnega tipa, pri katerih so izhodni napetostni nivoji določeni ne da bi bilo potrebno karkoli priključiti na izhod. Obstajajo pa tudi vezja, ki imajo ti. **odprte izhode** (OC-Open Collector, OD-Open Drain). Pri teh izhodih so izhodni nivoji določeni šele, ko med izhod in napajalno napetost priključimo ustrezen upor.



Slika 11

Z vezji, ki imajo takšne izhode, lahko prilagajamo napetostne nivoje (iz 5V na 12V ali obratno).

Upor mora biti izbran tako:

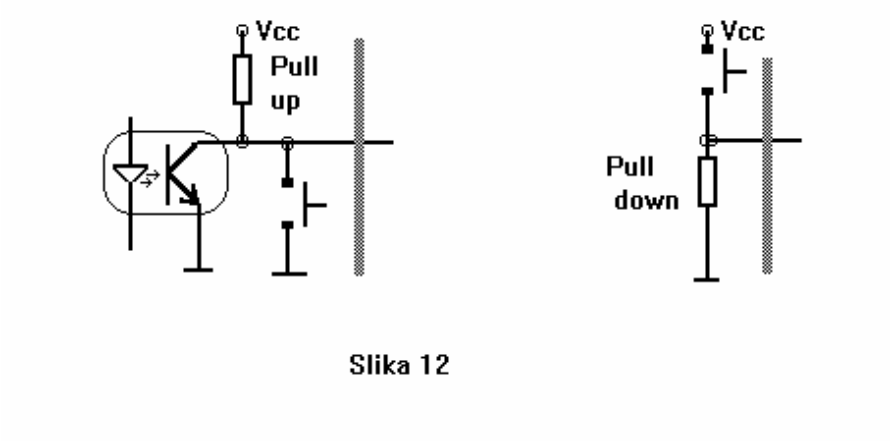
- da v visokem stanju Uizh ne pade pod U_{OH}min (omejitev upornosti navzgor). V tem primeru je izhodna upornost lahko prevelika saj tvori z bremensko upornostjo izhodni delilec napetosti.
- da v nizkem stanju Uizh ni večja od U_{OL}max (omejitev upornosti navzdol). V tem primeru bi lahko bil glede na bazni tok Ib prevelik kolektorski tok Ic. Posledica tega je, da tranzistor ni povsem odprt, kar povzroča na njem prevelik padec napetosti. Pri Fetu pa bi lahko bila upornost kanala glede na Rd prevelika.

2.3.6. Kompatibilnost

Nek vhod ali izhod sta kompatibilna, če sta izpolnjena naslednja dva pogoja:
- oba uporablja enake napetostne nivoje
- izhod dovoljuje obremenitev z najmanj enim standardnim bremenom (vhodom), vhod pa ne obremenjuje izhoda bolj kot eno standardno breme.

2.3.7. Pull up in pull down upor

Na vhodu logičnega vezja mora biti vedno prisotno logično stanje 0 ali 1 oz. vhod naj bo vedno priključen. V nasprotnem primeru je lahko vhod antena za motilne signale. V ta namen lahko uporabimo upor, ki ga vežemo proti potencialu napajanja ali mase. Nekateri logični elementi imajo takšne upore že integrirane (predvsem pull up). V tem primeru priključitev zunanjih ni potrebna. V praksi se pogosteje uporabljajo pull up (dvižni) upori, ki omogočajo preprosto povezavo s transistorji in optospojniki.



Slika 12

2.4. Mikroprocesor

Je osnovni element, okoli katerega je zgrajen mikroračunalnik. Narejen je kot integrirano vezje z visoko stopnjo integracije. V splošnem ga sestavljajo:

- Aritmetično logična enota ALE
- Krmilna enota KE
- Registri
- Notranje vodilo

2.4.1. Aritmetično logična enota

Je enota z naslednjimi funkcijami:
- seštevanje, odštevanje, množenje, deljenje
- pomik vsebin registrov
- komplementiranje vsebin registrov
- povečevanje in zmanjševanje vsebin določenih registrov za 1

- logične operacije (negacija, konjunkcija, disjunkcija, ekvivalenca, antivalenca)

ALE je pridružen register stanj, v katerem so posamezni med seboj neodvisni flip-flopi (zastavice), ki s svojimi stanji 0 ali1 opisujejo stanja v ALE. Ta so lahko:

- rezultat aritmetične ali logične operacije je enak nič (bit Z -- Zero)
- rezultat aritmetične operacije je negativen (bit N -- Negative)
- rezultat aritmetične operacije oziroma njegov bitni zapis presega kapaciteto ciljnega registra (bit OV -- Overflow), bit za prekoračitev
- pri aritmetični operaciji je prišlo do prenosa (bit C -- Carry)
- procesor pri izvajanju programa dovoljuje ali pa ne prekinitve (bit I -- Interrupt)

2.4.2. Krmilna enota

KE vodi delovanje uP, tako da posreduje v skladu s sprejeto inštrukcijo (ukaz programa) krmilne signale ostalim enotam. Izvajanje ene inštrukcije ali ukaza lahko razdelimo na dve fazi:

- faza dostavljanja inštrukcije (prevzem ukaza - fetch)
- faza izvajanja inštrukcije (izvršitev ukaza - execute)

V prvi fazi prinaša KE inštrukcijo v inštrukcijski register in dekodira operacijsko kodo. V drugi fazi pa v odvisnosti od operacijske kode spreminja stanja v uP in pošilja ustrezne krmilne signale.

2.4.3. Registri

Število registrov in njihove oznake so pri raznih izvedbah uP različne. V grobem ločimo registre, ki so programsko dostopni in programsko nedostopni. Programsko dostopne registre lahko v programu naslavljamo in tudi spreminjamo njihove vsebine.

1. Akumulator ACC

V 8 bitnem uP je 8 bitni register , preko katerega poteka največ operacij v procesorju:

- shranjevanje operandov nad katerimi izvaja uP aritmetične in logične operacije
- shranjevanje rezultatov teh operacij
- posredništvo pri prenosu iz ene v drugo pomnilniško lokacijo ali iz vmesnikov v pomnilnik oz. obratno

2. Programski števec - PC

Ta 16 bitni register (Program Counter) vodi procesor skozi program. V njem je vpisan naslov naslednje inštrukcije, ki jo bo začel izvajati procesor, ko bo zaključil tekočo inštrukcijo.

3. Kazalec sklada - SP

Poseben prostor v podatkovnem pomnilniku, ki se začenja z vnaprej določeno lokacijo imenujemo sklad. Kazalec sklada (Stack Pointer) je 16 bitni register, v katerem je vpisan naslov prve prazne lokacije v skladu pri vpisovanju in zadnje polne pri čitanju iz sklada.

4. Indeksni register - IX

Je 16 bitni register, ki ga uporabljamo za:

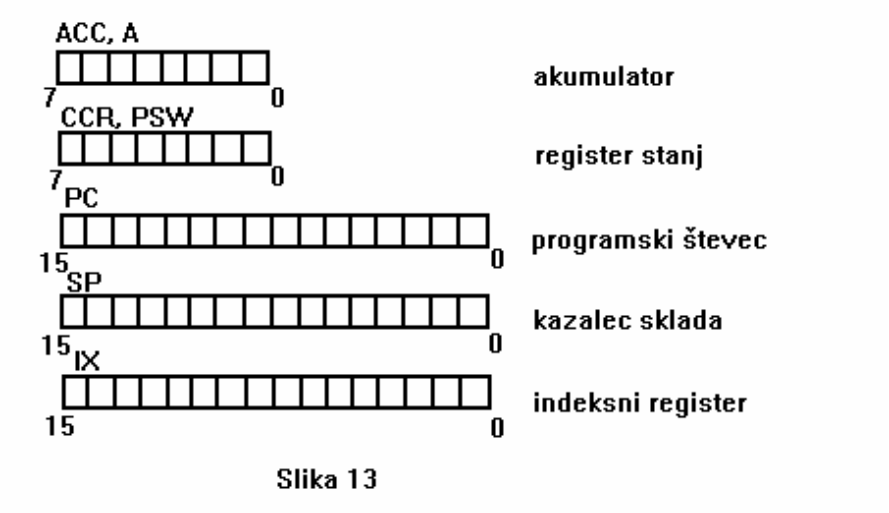
- shranjevanje naslovov pri indeksnem naslavljanju pomnilniških lokacij
- števec programskih obhodov v zanki določenega programa

Primer:

Sešteti moramo N podatkov. V tem primeru bomo v IX vpisali začetni naslov in ga po vsaki izvršeni operaciji povečali za 1, dokler ne bomo sešteli vseh N podatkov.

5. Register stanj - CCR, PSW

Je 8 bitni register, ki opisuje stanje ALE (Condition Code Register, Program Status Word). Glej pri ALE.



Slika 13

2.4.4. Prekinitve (interrupt)

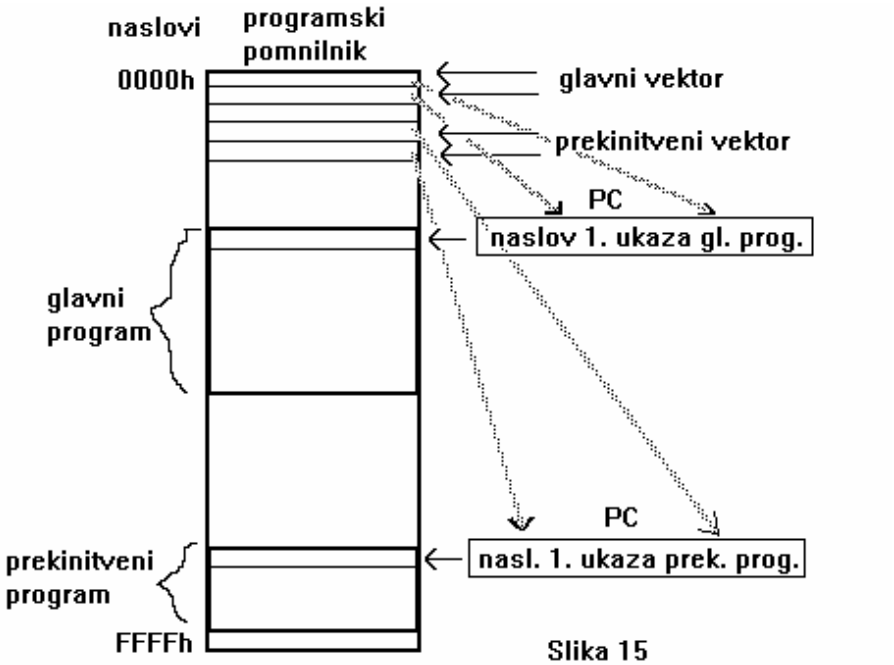
Ena od zelo pomembnih lastnosti vsakega uP je njegova sposobnost, da na zunanjo ali notranjo zahtevo prekine izvajanje trenutno izvajajočega programa in prične izvajati **prekinitveni servisni program**. Ob zaključku slednjega se vrne na prekinjeno mesto in nadaljuje z izvajanjem prekinjenega programa.

Slika 14

Osnovna zahteva pri prekinitvah je, da se po vrnitvi iz prekinitvenega servisnega programa v CPE vzpostavi stanje, ki je popolnoma enako tistemu ob nastopu prekinitve. Pravimo, da morajo biti prekinitve **transparentne - nevidne**. S tem dosežemo, da se prekinjeni program prekinitev ne zaveda in da so njegovi rezultati enaki ne glede na to ali je bil prekinjen ali ne in tudi ne glede na to , v kateri točki je bil prekinjen.

Vektorske prekinitve

Večina uP uporablja to zvrst prekinitev, kjer je vektor pomnilniški naslov v programskem pomnilniku, na katerem je shranjen naslov prekinitvenega servisnega programa. Ker gre za naslov, se imenuje tudi **prekinitveni naslovni vektor**. Če je možnih več prekinitvenih virov, ima vsak vir svoj prekinitveni vektor.



Slika 15

Viri prekinitev
Zahtevo po prekinitvi lahko uK sporočajo različni viri. Tako poznamo:
- zunanje vire (signali)
- notranje vire (časovniki, števci)
- serijsko komunikacijo

Maskiranje prekinitev
Zahteva za prekinitev se servisira le, če je to dovoljeno (prekinitev omogočena). Če ni, se zahteva ignorira in govorimo o maskiranju prekinitve. Ali neko prekinitev dovolimo ali ne, je odvisno, kaj v programu vpišemo v register za dovolitev prekinitev IE (Interrupt Enable), ki je sestavljen iz posameznih omogočitvenih bitov za posamezni prekinitveni vir.

Prioriteta prekinitev
V primeru istočasnega nastopa dveh ali več zahtev po prekinitvi imajo nekateri viri prednost pred drugimi. To podaja tabela prioritete proizvajalca uP. Ta določa tudi, ali lahko zahteva iz določenega vira prekine delovanje uP, če je v teku že določeni prekinitveni servisni program. Tudi prioriteto lahko določamo programsko v registru IP (Interrupt Priority).

Servisiranje prekinitve
Če je prekinitev zahtevana in omogočena se začne servisiranje prekinitve:
- na sklad se shrani vsebina vseh pomembnih registrov razen SP:
  - vsebina PC, da shranimo naslov ukaza, pri katerem je prišlo do prekinitve oz. naslov naslednjega ukaza prekinjenega programa
  - vsebina ACC, da shranimo trenutni rezultat oz. operand pred prekinitvijo
  - vsebina registra stanj, da shranimo trenutno stanje ALE pred prekinitvijo
  - vsebina IX, da shranimo v njem zapisan naslov pred prekinitvijo
- v PC se prenese vsebina viru pripadajočega prekinitvenega naslovnega vektorja, ki je zahteval prekinitev
- izvrševanje ukazov viru in vektorju pripadajočega prekinitvenega servisnega programa

Mikroprocesor ob vsaki prekinitvi sam shrani na sklad vsebino pomembnih registrov in jih tudi iz sklada vrne v registre. S tem je transparentnost zagotovljena ne da bi moral uporabnik o njej posebej razmišljati.

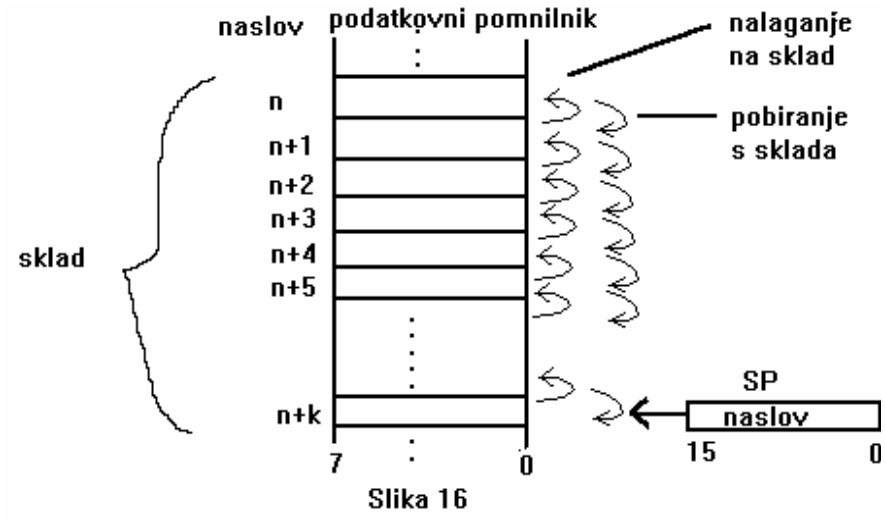
- Z ukazom, ki je zadnji v prekinitvenem programu in pomeni vrnitev iz prekinitve (RTI-Return from Interrupt) uP vrne podatke iz sklada nazaj v registre in vzpostavi stanje pred prekinitvijo.

2.4.5.Sklad

Pri večini uP se nekatere operacije vedno nanašajo na sklad. To so predvsem **prekinitve in klici podprogramov**. Sklad je pomemben tudi kot orodje za realizacijo nekaterih programskih rešitev (rekurzija). Sklad definiramo na naslednji način:

- rezerviramo del podatkovnega pomnilnika za sklad . Kako velik del pomnilnika bomo rezervirali, je odvisno od obsega uporabe sklada, ta pa od vrste programov, ki jih bo uP izvajal. V vsakem primeru pa moramo zagotoviti, da se prostor rezerviran za sklad ne uporabi za nič drugega.
- v kazalec sklada vpišemo vrednost, ki je enaka najvišjemu naslovu pomnilnika rezerviranega za sklad

Značilno za sklad je
- da se širi od višjih naslovov proti nižjim (ali obratno)
- da SP kaže vedno na naslednjo prazno lokacijo v skladu, v katero lahko uP opravi naslednji vpis oziroma na zadnjo polno pri branju iz sklada
- delovanje sklada je podobno delovanju registra LIFO (last in first out) - podatek, ki smo ga nazadnje shranili na sklad, je prvi, ki ga lahko preberemo.

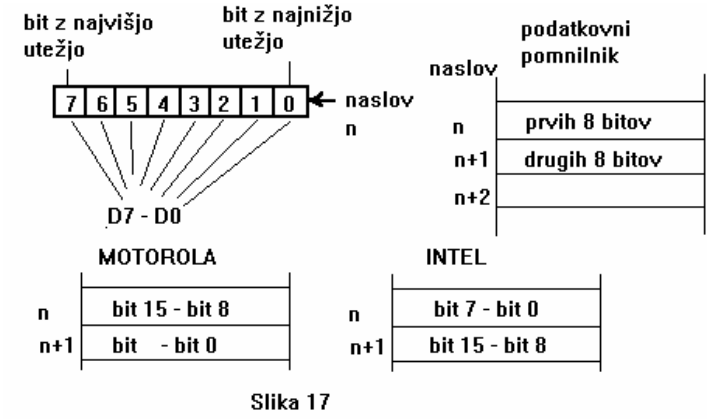


Slika 16

2.4.6. Organizacija podatkov v pomnilniku

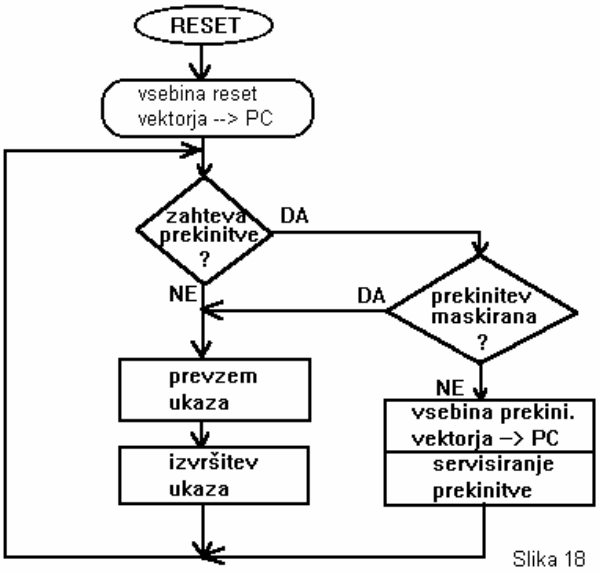
Gledano s strani uP je pomnilnik videti kot množica 8 bitnih besed (bytov). Kako velika je lahko ta množica, je odvisno od števila lokacij, ki jim uP lahko dodeli naslov. V primeru naslovnega vodila A0-A15 lahko uP naslovi 65536 različnih lokacij. Biti v teh besedah so uteženi in ustrezajo signalom podatkovnega vodila D0-D7.

Takšna organizacija bi zadoščala, če bi vsi ukazi uporabljali 8 bitne operande. Ker pa poznamo ukaze s 16 bitnimi operandi moramo definirati zgradbo teh v pomnilniku. Takšne operande razdelimo na dve polovici (spodnjo in zgornjo) ter jim dodelimo dve lokaciji. Naslov operanda je spodnji naslov n.



Slika 17

2.4.7. Delovanje mikroprocesorja



Slika 18

Reset vektor (glavni vektor) je rezervirana lokacija v programskem pomnilniku, kamor programator vpiše naslov prvega ukaza v glavnem programu, v katerem program steče po resetu (slika 15). Zaradi tega kaže PC po resetu na glavni vektor. Reset uP pomeni, da se njegovi registri postavijo v točno določena začetna stanja. Prekinitveni vektor določenega vira prekinitve pa je rezervirana lokacija v programskem pomnilniku, kamor programator vpiše naslov prvega ukaza prekinitvenega programa. Ker so naslovi običajno 16 bitni, lokacije programskega pomnilnika pa 8 bitne, zavzemajo vektorji dve ali več lokacij.

Mikroprocesor se pri svojem delovanju ravna po signalih ure (clock) ali takta. Njegova opravila so organizirana v takoimenovanih **strojnih ciklih**. En strojni cikel je lahko dolg

eno ali več urinih period, kar je odvisno od izvedbe uP. Strojni cikli se imenujejo glede na dogajanja znotraj njih. Tako poznamo:

- bralni cikel
- pisalni cikel
- prekinitveno prevzemni cikel
- čisti strojni ali izvršbeni cikel

Bralni cikel
Pri tem ciklu uP pošlje na naslovne signale A0-A15 naslov lokacije, iz katere želi prebrati ukaz ali podatek. Na krmilno vodilo pošlje signal za branje (RD -- read). Na podatkovnih signalih D0-D7 pa pričakuje ukaz ali podatek.

Pisalni cikel
Pri tem ciklu uP pošlje na naslovne signale A0-A15 naslov lokacije, v katero želi vpisati podatek. Na krmilno vodilo pošlje signal za pisanje (WR -- write). Ta podatek pošlje na podatkovne signale D0-D7.

Program je sestavljen iz ukazov ali inštrukcij, ki povedo uP kaj mora narediti. Izvajanje vsake inštrukcije poteka preko več strojnih ciklov, ki sestavljajo **inštrukcijski cikel**. **Prevzem ukaza** ali fetch je eden ali več bralnih ciklov, v katerih uP bere iz lokacij programskega pomnilnika. **Izvršitev ukaza** ali execute pa je korak realizacije ukaza.

Primer: inštrukcija c=a+b
- fetch
1. bralni cikel operacijske kode
- execute
1. bralni cikel operanda a
2. bralni cikel operanda b
3. strojni cikel seštevanja
4. pisalni cikel shranjevanja rezultata

2.5 Pomnilnik

Idealen pomnilnik bi moral imeti vse naslednje lastosti:

- iz njega bi lahko izvajali neomejeno število branj (to lastnost imajo pomnilniki v obliki IC)
- vanj bi lahko opravljali neomejeno število vpisovanj (to lastnost ima samo RAM)
- ob izpadu napajalne napetosti bi moral vsebino zadržati (to lastnost imajo ROM in Flash RAM)
- vpisovanje bi moralo potekati enako hitro kot branje (samo RAM)

- kapaciteta pomnilnika in hitrost dostopa bi morala biti dovolj velika pri sorazmerno nizki ceni

V grobem delimo pomnilnike v obliki IC na bralne in bralno-zapisovalne. Iz vseh je možno neomejeno branje, število vpisov pa se razlikuje. V skupino, ki imajo končno število vpisov in zadržujejo vsebino tudi ob izpadu napajalne napetosti, sodijo:

- ROM (read only memory)
  Vpis vanj izvrši proizvajalec. Uporaben je za program, konstante in tabele.
- PROM (programable ROM)
  Omogoča enkraten vpis vsebine, kar lahko izvrši programer s pomočjo programatorja. Uporabnost je takšna kot pri ROM-u.
- EPROM (erasable PROM)
  S pomočjo UV svetlobe je možno njegovo vsebino izbrisati, tako da omogoča nekaj deset vpisov s pomočjo programatorja. Uporabnost je podobna kot pri PROM-u.
- EEPROM (electric EPROM)
  Vsebino je možno izbrisati kar v ciljnem sistemu, vendar le nekaj 100000 krat. Zaradi tega nima enake uporabne zmožnosti kot RAM. Uporabljamo ga lahko za podatke, ki se redkeje spreminjajo oz. jih spreminja uporabnik s pomočjo tipkovnice.
- FLASH RAM
  Je nižje cenovna verzija EEPROM-a in omogoča do 1000 vpisov.

Vpisovanje v zgoraj naštete pomnilnike poteka po določeni proceduri, vsekakor pa počasneje kot pri pomnilniku, v katerega lahko neomejeno vpisujemo. Takšen pomnilnik je:

- RAM (random access memory)
  Svojo vsebino ob izpadu napajalne napetosti izgubi. Uporaben je za shranjevanje podatkov. Če želimo te podatke trajno ohraniti, moramo uporabljati sistem neprekinjenega napajanja ali pa ob izpadu napajanja podatke prepišemo v EEPROM. Če želimo RAM uporabljati za programski pomnilnik, moramo vsakič ob vklopu naložiti program.

2.6. Vhodno izhodni vmesnik

Poznamo več vrst vhodno izhodnih vmesnikov. Nekateri od njih imajo programsko dostopne registre, s katerimi lahko uP komunicira. Razen podatkov, ki se prenašajo preko vmesnikov, pošilja uP ukaze, s katerimi določa, kako naj takšen vmesnik deluje (smer pretoka podatkov, usklajenost delovanja, možnost prekinitev ...). Takšne vmesnike imenujemo **PIA (paralel interface adapter).** Pomembno je, da ima vsak register vmesnika svoj naslov ali pa da register določimo s pomočjo naslovnega dekoderja. Tako lahko v programu vmesnik obravnavamo kot običajno pomnilniško lokacijo, preko katere komuniciramo z zunanjim svetom.

Za vmesnike pa največkrat uporabljamo registre, ki jim pravimo zadrževalniki (latch). Tem moramo pošiljati kontrolne signale, ki te vmesnike odpirajo in zapirajo. Dobimo jih s pomočjo naslova preko naslovnega dekoderja. Imajo naslednje vhodne kontrolne signale:

- CE (chip enable)
  Vmesnik je omogočen, oz. pripravljen sprejemati podatke (vpisovanje), kadar je na tem vhodu 1log. Različica tega signala je CS (chip select).
- OE (output enable)
  Omogočeno je branje iz vmesnika oz. na izhodih vmesnika se pojavi v registru zapisana vsebina, kadar je na tem vhodu 1log.

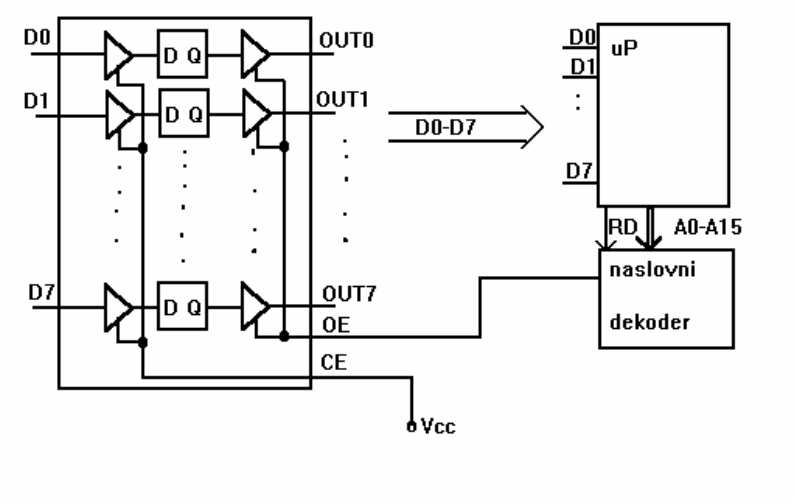
Takšne vmesnike ne moremo uporabljati kot dvosmerne, kajti smer je določena z ožičenjem. Dvosmerni vmesniki imajo še dodaten kontrolni signal, s katerim določamo smer. Naslovni dekoder kreiramo s pomočjo logične enačbe.

2.6.1. Primer vhodnega registra

Napiši enačbo za kontrolna signala vhodnega registra na naslovu 0000h.

CE = 1 CE zvežemo na napetost napajanja

OE = $\overline{A15}$ & $\overline{A14}$ & $\overline{A13}$ & $\overline{A12}$ & $\overline{A11}$ & $\overline{A10}$ & $\overline{A9}$ & $\overline{A8}$ & $\overline{A7}$ & $\overline{A6}$ & $\overline{A5}$ &
& $\overline{A4}$ & $\overline{A3}$ & $\overline{A2}$ & $\overline{A1}$ & $\overline{A0}$ & RD



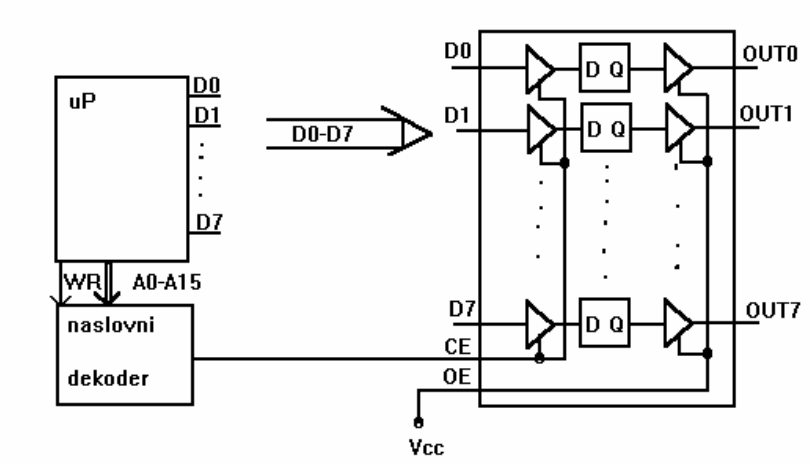
Vmesnik sprejema signale iz okolice neprenehoma, saj je signal CE zvezan na potencial napajalne napetosti. Pri branju iz vmesnika je potrebno setirati signal OE preko naslovnega dekoderja pri točno določenem naslovu v sodelovanju s krmilnim signalom RD (read) za branje.

2.6.2. Primer izhodnega registra

Napiši enačbo za kontrolna signala izhodnega registra na naslovu 0001h.

CE = $\overline{A15}$ & $\overline{A14}$ & $\overline{A13}$ & $\overline{A12}$ & $\overline{A11}$ & $\overline{A10}$ & $\overline{A9}$ & $\overline{A8}$ & $\overline{A7}$ & $\overline{A6}$ & $\overline{A5}$ &
& $\overline{A4}$ & $\overline{A3}$ & $\overline{A2}$ & $\overline{A1}$ & A0 & WR

OE=1 OE zvežemo na napetost napajanja



Signal OE lahko povežemo na napetost napajanja in s tem omogočimo, da je signal na izhodu vedno prisoten. Če pa ga zvežemo na signal /RESET, onemogočimo prisotnost signalov na izhodu ob vklopu napajanja. Zapis v vmesnik je možen, če je signal CE v visokem stanju. V enačbo dekoderja lahko vključimo tudi krmilni signal WR (write) za vpisovanje.

2.7. Programiranje

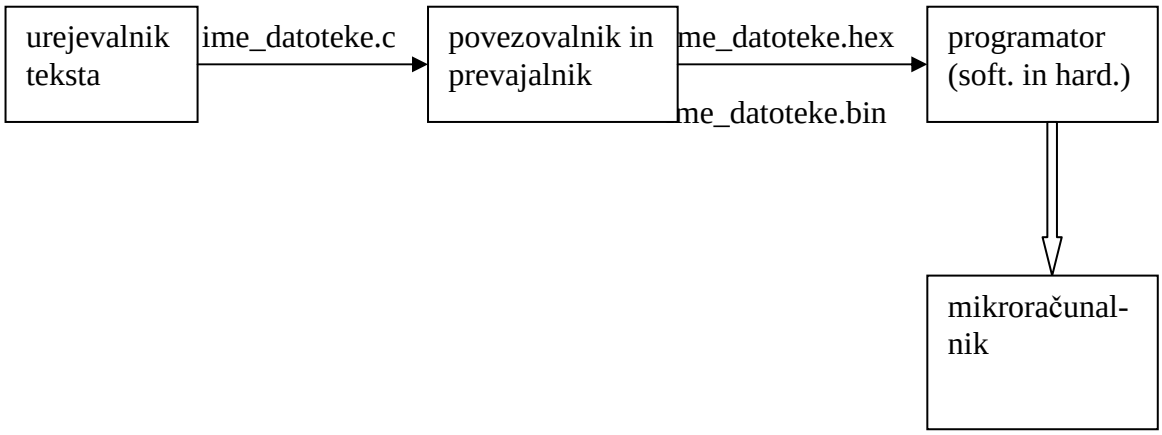
Vsak mikroprocesor pozna določene **ukaze**. Množico ukazov za nek uP imenujemo **nabor ukazov.** Uporabnik uporablja nabor ukazov, tako da z njimi sestavi neko smiselno zaporedje. Temu zaporedju pravimo **program**, njegovemu sestavljanju pa **programiranje.**

Programiranje lahko opravljamo na **strojnem ali zbirniškem nivoju**, kjer neposredno uporabljamo nabor ukazov (zbirnik ali assembler). Druga možnost je programiranje v **višjem programskem jeziku**, ki je človeku bližji (Basic, Fortran, Pascal, C…).

Programer mora do podrobnosti razumeti problem, za katerega piše program. Njegovo delo lahko razčlenimo v naslednja opravila:

- **Analiza problema** je razvoj postopka, ki korak za korakom reši dani problem. Temu postopku pravimo **algoritem**.
- **Organizacija programa**. Algoritem organiziramo v zaporedje operacij, pri čemer uporabljamo **diagrame poteka**. Delo lahko opravljamo s pomočjo osebnega računalnika, če imamo orodje za risanje diagramov poteka.
- **Kodiranje**. Diagram poteka zapišemo v zaporedje ukazov–program s pomočjo **urejevalnika teksta** za osebni računalnik. Shranimo ga v ti. izvorni datoteki.
- **Povezovanje in prevajanje**. Program povežemo z drugimi deli programa, ki smo jih napisali že v preteklosti ali pa s programi, ki jih dobimo v knjižnici (napisal jih je proizvajalec prevajalnika in povezovalnika). Povezavo opravi **povezovalnik - linker**. Celoten program prevedemo s **prevajalnikom - compilerjem** v obliko, ki jo razume uP. Povezovalnik in prevajalnik sta programski opremi za osebni računalnik.

- **Vstavljanje programa v mikroračunalnik** opravljamo s pomočjo **programatorja**, ki ga priključimo na osebni računalnik s pripadajočo programsko opremo.
- **Preizkušanje**. Program preizkusimo tako, da z izvajanjem na uR ugotovimo, ali da ustrezne rezultate. V ta namen izdelamo testno kartico. Napake, ki jih pri tem odkrijemo odpravimo tako, da se vrnemo na kodiranje in ponovimo postopek.
- **Dokumentiranje**. Za kasnejšo dodelavo ali spremembo programa je potrebno program dokumentirati. Dokumentacijo s potrebnimi komentarji uredimo pri vseh opravilnih točkah (tehnični opis, poročilo, elaborat).



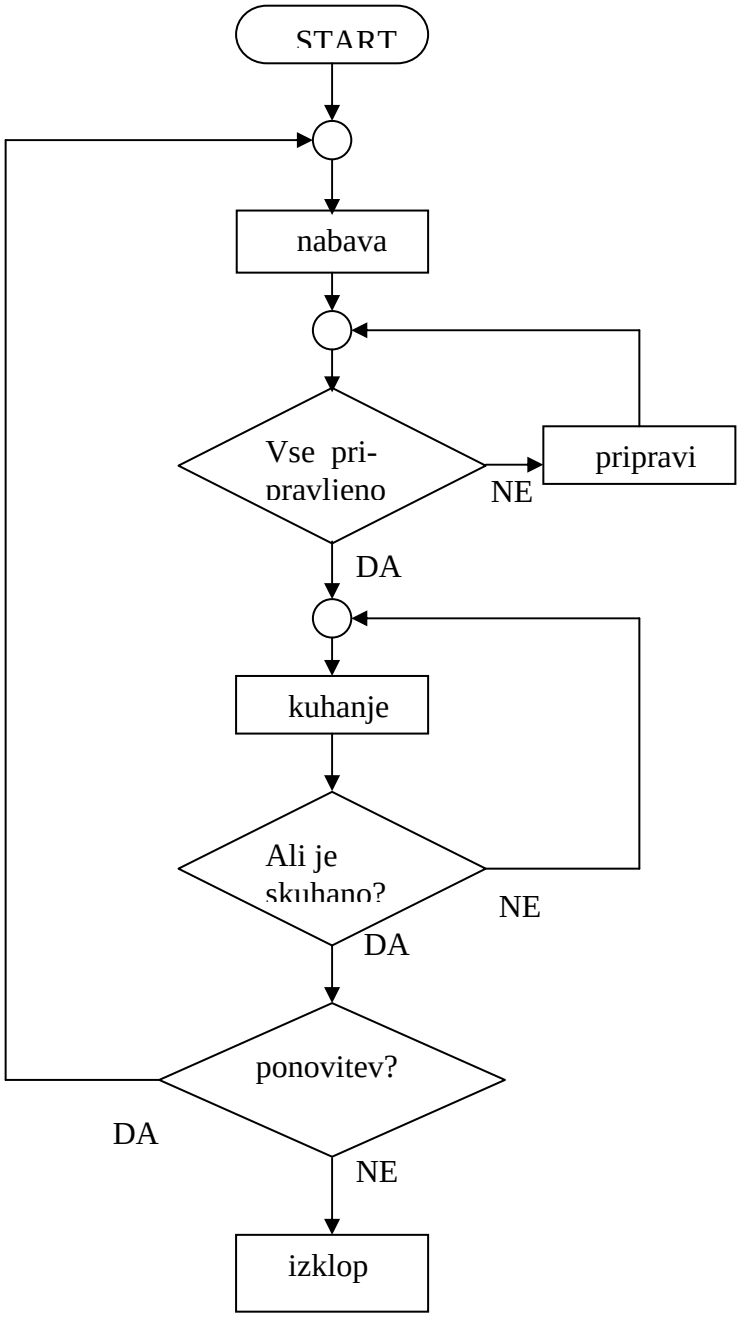
Primer:

Realiziraj prvi dve točki za problem kuhanja kave.

1. Analiza problema

a) nabava potrebščin (kava, sladkor in voda), opreme(štedilnik, žlica, kuhalna posoda) in energije (elektrika ali plin) in njihova priprava
b) kuhanje
c) testiranje (ali je že kuhano)
d) testiranje ponovitve
e) izklop

2. Organizacija programa
diagram poteka:


START
nabava
Vse pri-
pravljeno
NE
pripravi
DA
kuhanje
Ali je
skuhano?
NE
DA
ponovitev?
DA
NE
izklop

3. Programiranje v zbirnem jeziku

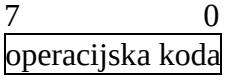
Zbirni jezik (zbirnik) ali **assembler** je jezik določenega procesorja oziroma družine mikrokontrolerjev, ki bazirajo na istem mikroprocesorju. Uporablja nabor oziroma zbirek ukazov, ki jih uP pozna.

V vsakem ukazu mora biti prisotna informacija o dveh stvareh:
- operaciji
- operandih

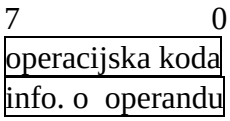
Informacija o operaciji je vsebovana v ti. **operacijski kodi**. Informacija o operandih pa je lahko podana na več načinov. V mikroračunalniku so operandi lahko shranjeni v registrih procesorja, v pomnilniku ali v registrih vhodnih naprav. V vsakem ukazu je zato potrebno na nek način povedati, kje se operandi nahajajo. Ker to običajno povemo z naslovom, na katerem se operand nahaja, pravimo temu naslavljanje. Kje se operandi nahajajo, lahko povemo na več načinov in tem pravimo **načini naslavljanja**.

Ukazi lahko v programskem pomnilniku zasedajo eno ali več lokacij. Tako poznamo eno, dve ali več zložne ukaze (zlog=byte=8bitov). Ukaz se prične z operacijsko kodo.

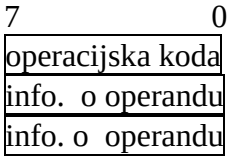
- enozložni ukaz



- dvozložni ukaz



- trozložni ukaz



4. Pomnilnik

4.1. Uvod

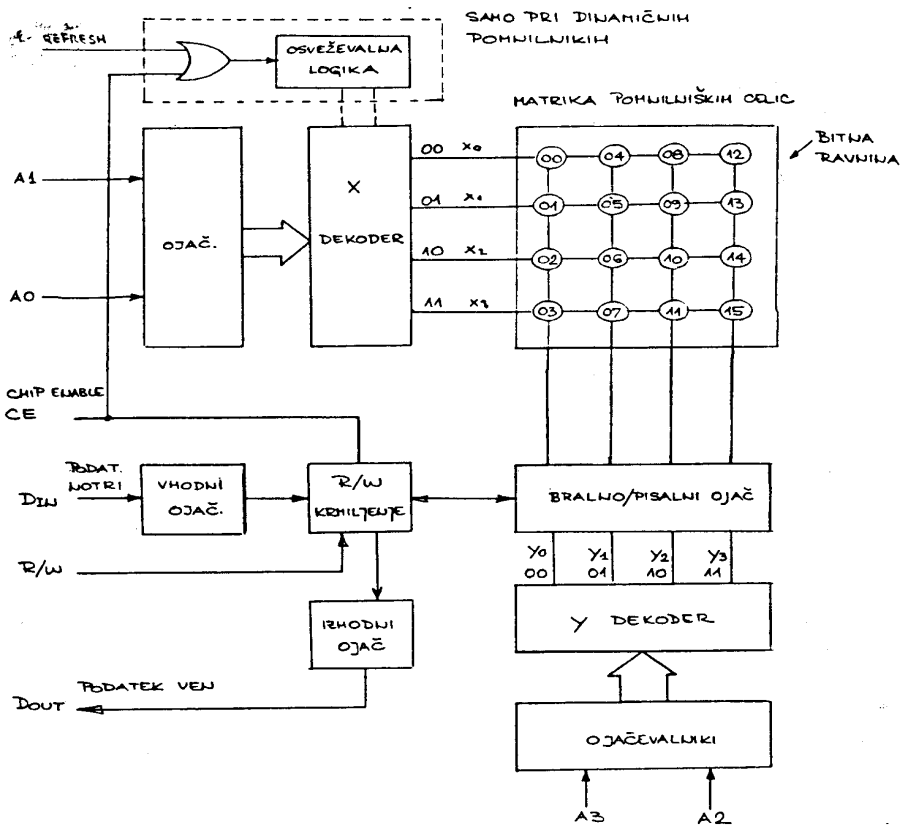
V pomnilniku so shranjeni ukazi in podatki. Največji del prenosa signalov v mikroračunalniku se nanaša na prenose med uP in pomnilnikom, medtem ko na prenose med uP in vhodno/izhodnimi napravami odpade razmeroma majhen del.

4.2. Načini dostopa do pomnilnikov

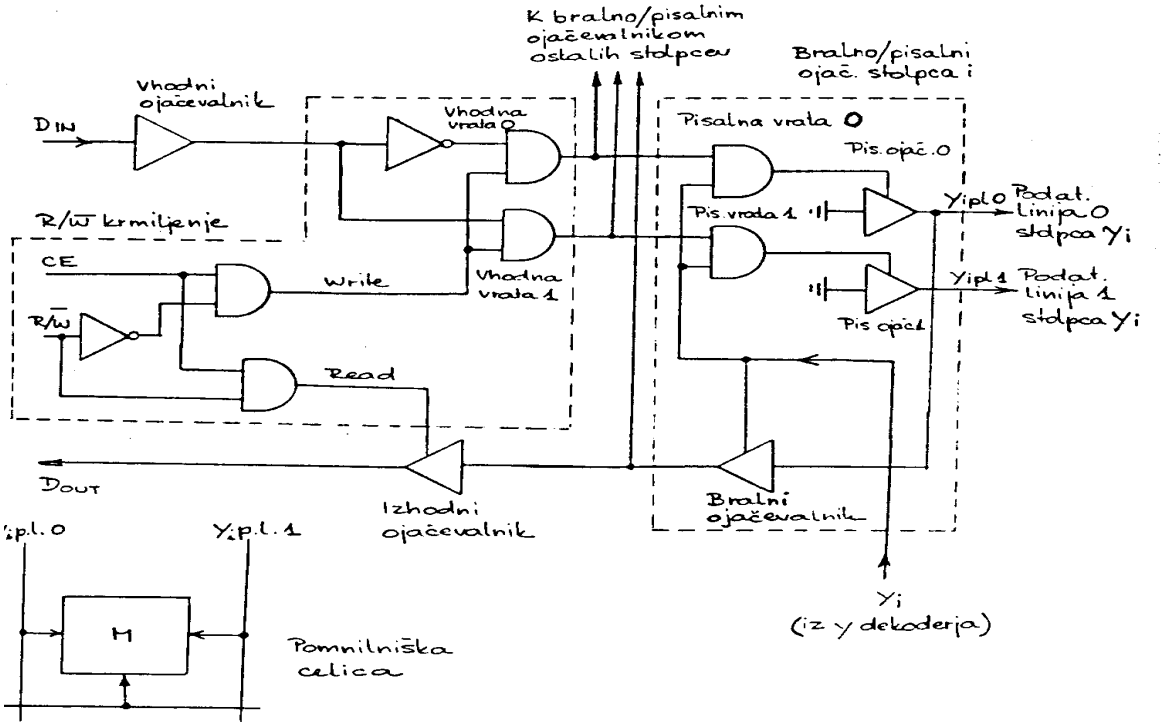
Glede na način kako hitro lahko dobimo podatke ali ukaze oziroma kako hitro jih lahko v pomnilnik shranimo, poznamo naslednje načine dostopa:

- **Naključni dostop (random access)**
  Pri tem načinu imamo dostop do katerekoli lokacije časovno neodvise od predhodnega naslova dostopanja. **Čas dostopa t_{ACC}** je konstanten in vnaprej znan. Glavni pomnilniki računalnikov so vedno te vrste in so realizirani kot integrirana vezja. Kratica RAM pomeni ravno ta dostop in ne kot jo napačno uporabljamo za pomnilnik branja in vpisovanja, kajti tudi ROMi lahko imajo takšen dostop.
- **Serijski – zaporedni dostop (serial access)**
  Čas dostopa do neke lokacije je odvisen od naslova prejšnje lokacije dostopanja. Če je bi prejšnji naslov N, sta takoj dostopna samo naslova N+1 in N-1. Za dostop do poljubnega naslova M pa moramo izvršiti dostope do vseh lokacij med N in M. Torej je dostopni čas močno odvisen od zaporedja naslovov. Tipični predstavniki teh so magnetni trak, pomikalni register in zakasnilna linija.
- **Neposredni dostop (direct access)**
  To so tisti načini dostopa, pri katerih je čas za dostop do neke lokacije sicer odvisen od naslova prejšnje dostopane lokacije, vendar je ta odvisnost bistveno manjša kot pri zaporednem dostopu. Predstavniki so magnetni disk in CD.

4.3. Zgradba pomnilnikov z naključnim dostopom



Pomnilnik ima organizacijo 16x1, kar pomeni, da vsebuje 16 enobitnih besed. Vsaka kombinacija signalov A0, A1, A2, A3 bo povzročila odprtje natanko ene celice (presek vrstice in stolpca). Tak način naslavljanja (z x/z dekodiranjem) je znan kot **dvodimenzionalno naslavljanje**, ki potrebuje manj povezovalnih vodnikov. Pri enodimenzionalnem naslavljanju ima vsaka celica svojo naslovno linijo.



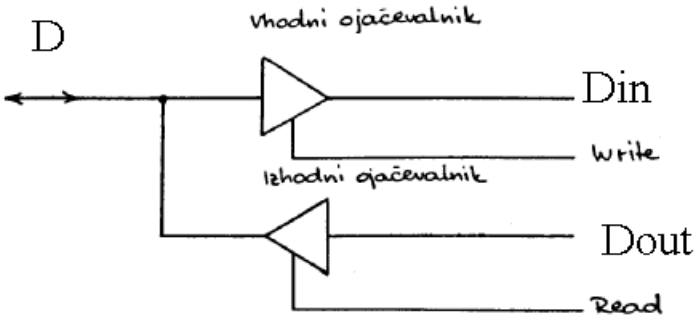
- **Vpisovanje**

Pri vpisovanju se podatek na vhodni podatkovni sponki Din preko vhodnega ojačevalnika prenese originalen in negiran na vhodna vrata. Če je IC omogočen (CE=1) in je signal R/~~W~~ nič, imamo notranji signal *Write=1*. Vhodna vrata 0 in 1 prepuščajo podatek k bralno pisalnim ojačevalnikom vseh stolpcev. Ker je samo en stolpec določen (signal Yi=1), se podatek preko pisalnih vrat 0 in 1 preseli na kontrolni vhod pisalnega ojačevalnika 0 in 1 tega stolpca. Če je kontrolni vhod pisalnega ojačevalnika log. 0 dobimo na izhodu stanje Z, če pa je log.1 pa je na izhodu log.0. Tako imamo na dveh podatkovnih linijah 0 in 1 tega stolpca stanji 0 in Z ali Z in 0, kar je odvisno od vhodnega podatka. Od te kombinacije je odvisno ali se bo v pomnilniško celico vpisala 1 ali 0. Katera pomnilniška celica tega stolpca bo doživela vpis pa je odvisno od stanja Xi določene vrstice.

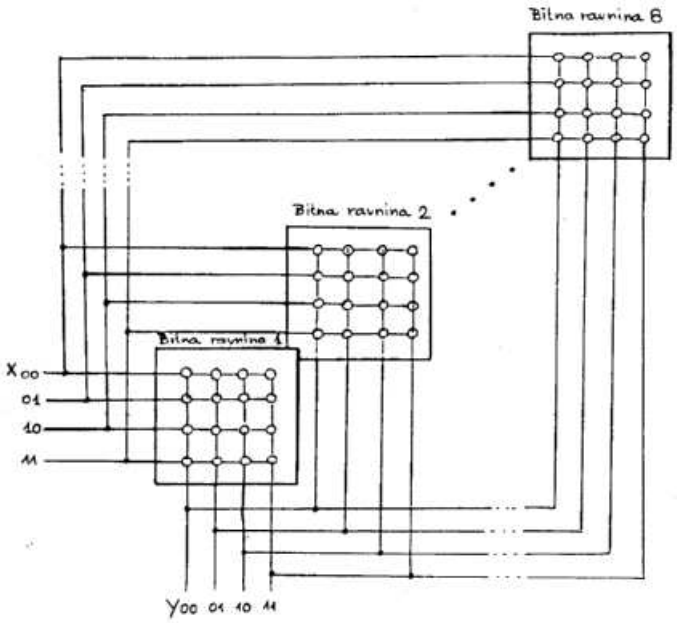
- **Branje**

Pomnilniške celice določene vrstice Xi bodo dale na svojih podatkovnih linijah 0 stanja svoje vsebine (0 ali 1). Iz katere od teh celic želimo podatek določa signal Yi, ki prepušča podatek skozi bralni ojačevalnik natanko ene celice. Če je CE=1 in R/~~W~~=1 se postavi notranji signal *Read* in podatek iz celice se preko izhodnega ojačevalnika pojavi na signalu Dout.

Ker so signali podatkovnega vodila dvosmerni, je potrebno izvesti združitev signalov Din ter Dout.



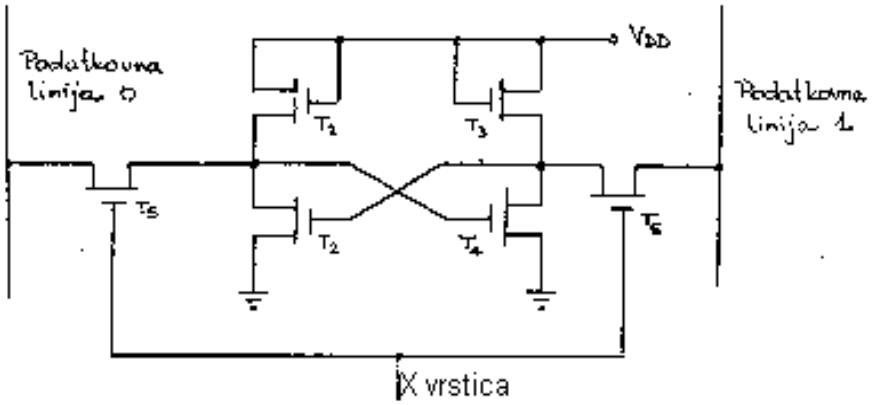
Za 8 bitne uR je značilna 8 bitna organizacija podatkov v pomnilniku. Zato mora pomnilnik vsebovati 8 bitnih ravnin.



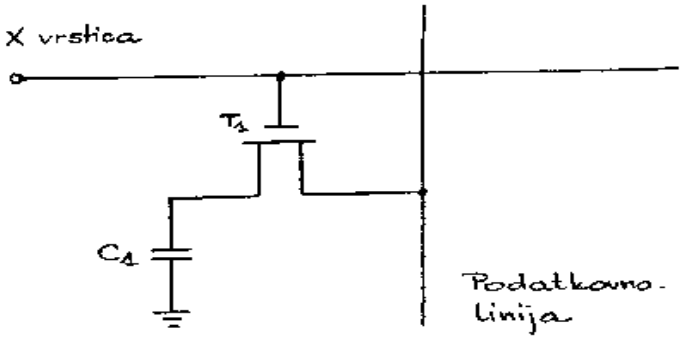
4.4. Zgradba pomnilniških celic

4.4.1. Statična RAM celica

Je bistabilni multivibrator, ki ga sestavljajo tranzistorji T1 do T4. Kadar je celica izbrana (visok nivo na signalu Xi), se pri branju preko T5 stanje, v katerem je MV, pojavi na podatkovni liniji 0. Pri pisanju pa stanje podatkovnih linij 0 in 1 povzroči, da se MV postavi v ustrezno stanje.



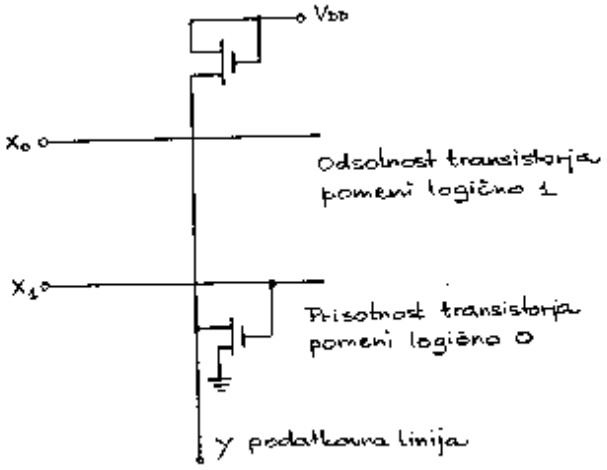
4.4.2. Dinamična RAM celica



Celico sestavljata kondenzator C in tranzistor T1. Konstrukcija je mnogo manjša od statične RAM celice, zato ti pomnilniki dosegajo večjo kapaciteto pomnenja. Pri pisanju se stanje podatkovne linije prenese na kondenzator. Če je podatkovna linija v stanju log.1 se kondenzator nabije v nasprotnem primeru pa se izprazni. Informacija je shranjena v obliki naboja na C. Kapacitivnost znaša manj kot 0.1 pF. Ker je naboj zelo majhen, s časom hitro izginja. Da preprečimo izgubo naboja moramo vsako celico od časa do časa osvežit. Pri osveževanju se nabiti kondenzatorji ponovno nabijejo in prazni ponovno izpraznijo. Večina teh elementov zahteva, da se vsaka celica osveži najmanj 2x na vsako milisekundo.

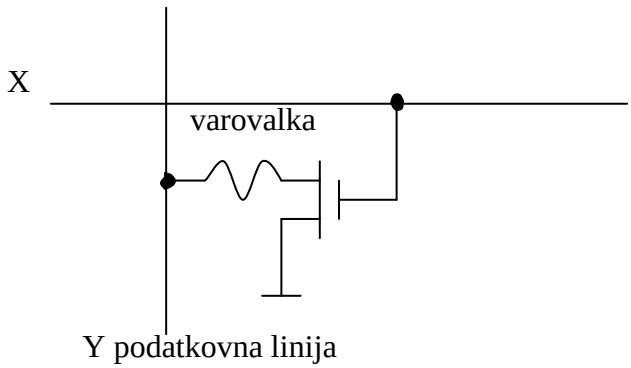
4.4.3. ROM celica

Prisotnost oziroma odsotnost tranzistorja je določena v tovarni z masko. Te vrste pomnilniki so zanimivi , kadar potrebujemo veliko število elementov z enako vsebino, kajti pri zelo velikih količinah postanejo ROM elementi cenejši od vseh drugih bralnih pomnilnikov.



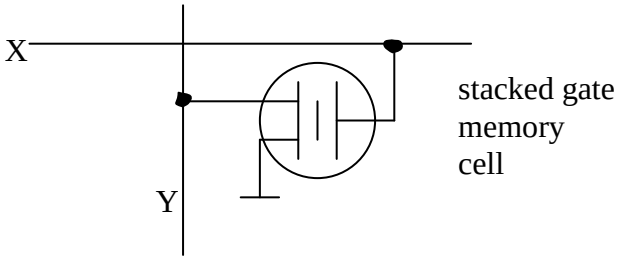
4.4.4. PROM celica

Vsaka celica vsebuje varovalko, ki jo je s pomočjo programatorja možno prežgati in tako vpisati logično 1. Prežgane varovalke ni mogoče popraviti, zato lahko te elemente programiramo samo enkrat. Uporabljamo jih, ko količina ne opravičuje uporabe ROM elementov.



4.4.5. EPROM celica

Posebni tranzistor se s pomočjo višje napetosti (12V) postavi v prevodno stanje. S pomočjo UV svetlobe lahko vedno ponovno vzpostavimo neprevodno stanje. Elemente lahko preprogramiramo večkrat (do 100x) in so zelo primerni pri razvoju prototipov in maloserijski proizvodnji.



4.4.6. EEPROM celica

Tako kot pri eprom celici gre tudi tukaj za poseben element, v katerega lahko vpisujemo že z enakimi napetostnimi nivoji , kot jih ima uP, le da vpisovanje zahteva nekaj milisekund za vsako lokacijo. Število vpisov pa je omejeno do milijon. Ponoven vpis je hkrati tudi izbris prejšnje vsebine. Te celice uporabljajo serijski EEPROM pomnilniki v chip karticah.

4.4.7. FLASH RAM celica

Je cenejša varianta EEPROMA in omogoča do 1000 vpisov.

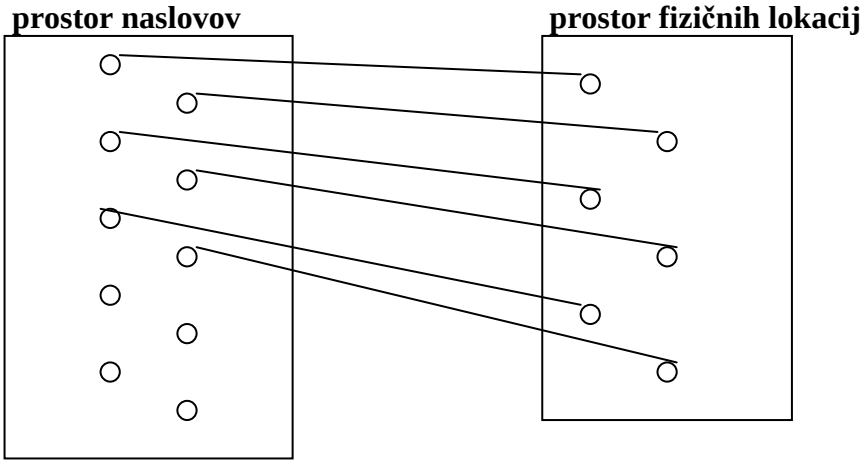
Novejši mikrokontrolerji imajo ponavadi integrirane naslednje pomnilnike:
- Statični RAM pomnilnik kot podatkovni (nekaj 10 B)
- Flash RAM pomnilnik kot programski (nekaj kB)
- EEPROM pomnilnik za podatke, ki se obdržijo brez napajalne napetosti (nekaj 10 B)

4.5. Naslovni dekoder

Značilno za uR je, da v glavnem pomnilniku srečujemo več tipov pomnilnikov z naključnim dostopom. Razen tega imajo tudi vhodno/izhodni vmesniki svoje pomnilniške lokacije.

Način razporeditve naslovov na pomnilniške elemente imenujemo tudi **naslovna organizacija pomnilnika.** Ta je odvisna od razpoložljivega naslovnega prostora uP, od vrste uporabljenih pomnilniških elementov in od namenov, za katere gradimo uR.

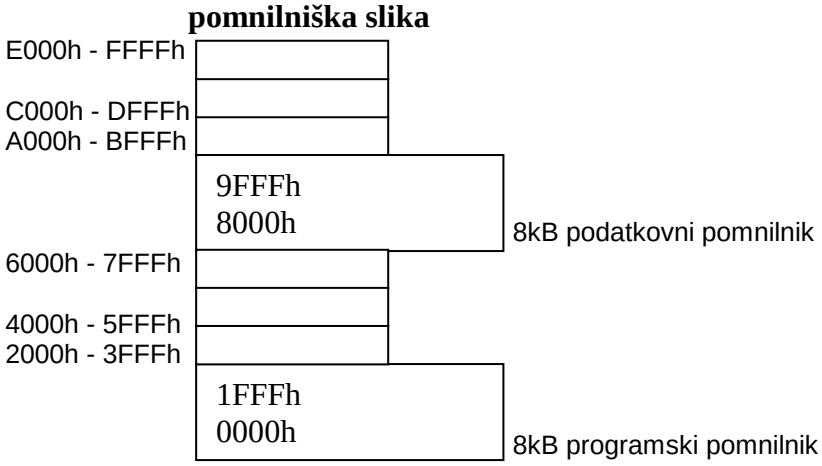
Zveza med naslovi in fizičnimi pomnilniškimi lokacijami mora biti enolična. Vsak naslov preslikamo v največ eno lokacijo oziroma naslov vsake lokacije mora biti različen. Realizacija neke organizacije pomnilnika ni nič drugega kot **prepoznavanje ali dekodiranje ustreznih naslovov.**



4.5.1. Primer popolnega in nepopolnega naslovnega dekodiranja

V mikroračunalniku, ki uporablja mikroprocesor z naslovnim prostorom 64kB in glavnim vektorjem na naslovu 0000h, želimo imeti naslednjo organizacijo pomnilnika:

- SRAM podatkovni pomnilnik 8kB
- EPROM programski pomnilnik 8kB



Naslov Pomnilniška enota		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
naslovni prostor		0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1
EPROM	začetni naslov	*0*	*0*	*0*	0	0	0	0	0	0	0	0	0	0	0	0	0
	končni naslov	*0*	*0*	*0*	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM	začetni naslov	*1*	*0*	*0*	0	0	0	0	0	0	0	0	0	0	0	0	0
	končni naslov	*1*	*0*	*0*	1	1	1	1	1	1	1	1	1	1	1	1	1

Imamo primer, da sta oba tipa elementov enako velika (8kB) in imata zaradi tega le 13 naslovnih signalov (A0-A12). Preostalim signalom A13, A14, A15 pravimo **redundantni** naslovni signali. Ti so konstantni v celotnem področju lokacij posameznega pomnilniškega elementa in določajo kje v pomnilniški sliki se ta nahaja. Ti trije signali lahko zavzamajo 8 kombinacij, kar pomeni tudi 8 možnih pozicij v pomnilniški sliki (za vsak element smo izbrali le eno pozicijo).

Če te signale pripeljemo na naslovni dekoder, nam ta preko svoje logične enačbe odredi pozicijo v pomnilniški sliki s pomočjo izhodnih signalov, ki ju pripeljemo na kontrolna vhoda pomnilniških elementov CEeprom in CEram. Enačbi naslovnega dekoderja se glasita:

$$CEeprom = \overline{A15}*\overline{A14}*\overline{A13}$$

$$CEram = A15*\overline{A14}*\overline{A13}$$

Izbira pozicije programskega pomnilnika je odvisna od naslova glavnega vektorja, medtem ko je izbira pozicije podatkovnega pomnilnika prosta.

Če za naslovno dekodiranje uporabimo vse redundantne ali preostale naslovne signale, govorimo o **popolnem naslovnem dekodiranju**, kjer ima vsaka lokacija le en točno določen naslov.

V primeru, da imamo večino naslovnega prostora neizkoriščenega (v našem primeru je tako), lahko izvedemo tudi **nepopolno dekodiranje**. Vzamemo samo tiste redundantne signale, ki se razlikujejo pri pomnilniških elementih. S tem poenostavimo enačbo naslovnega dekoderja in njegovo vezje. Enačbi se glasita:

$$CEeprom = \overline{A15} \qquad CEram = A15$$

Sedaj ima določena fizična lokacija več možnih naslovov (4 kombinacije A14, A13). Kateregakoli uporabimo bo določena pravilna želena lokacija, vendar je zaradi preglednosti in programskega reda dobro, če se odločimo uporabljati samo en naslov.

5. Serijska komunikacija

5.1. Uvod

Če se naj dve napravi razumeta, morata uporabljati isti prenosni medij, imeti enake električne in druge fizikalne lastnosti, uporabljati enak nabor znakov in enak jezik. Kadar sodeluje več naprav, pa moramo upoštevati še dodatne dogovore:

- kako naj naprava vzpostavi zvezo z naslovnikom
- kako se naj naprava predstavi

Poseben problem predstavlja pogoj, da ne sme nikdar istočasno oddajati več naprav. Nastala je cela vrsta pravil in postopkov, ki določajo posamezne načine prenosa. Nabor pravil in postopkov, ki urejajo prenos informacij med ljudmi, napravami in procesi imenujemo protokol.

Da bi lahko prišlo do komuniciranja med vsemi zainteresiranimi udeleženci, morajo vsi uporabljati enak komunikacijski protokol. Te probleme rešuje standardizacija. Standard je eden ali več formalno ali neformalno sprejetih protokolov, ki jih razumejo vsi zainteresirani ljudje, stroji in naprave. Neformalni standardi so tisti, ki so močno razširjeni in za katerimi stojijo velike korporacije. Formalni standardi pa so tisti, ki jih sprejmejo običajno na podlagi neformalnih posebne organizacije za standardizacijo:

- ISO – International Standards Organization
- ANSI – American National Standards Institute
- DIN – Deutschhe Industrie Norm
- IEEE – Institute of Electrical and Electronics Engineers
- EIA – Electronics Industry Association

Žal najboljšega protokola ni. Obstaja le bolj ali manj optimalen protokol glede na naše potrebe in okolje, v katerem bo deloval. Protokol, ki deluje v enem okolju zelo dobro, je lahko v drugem popolnoma neučinkovit.

5.2. Standard RS-232 ali V.24

Je najbolj pogost protokol asinhrone serijske (bit za bitom) komunikacije. Podatki med dvema enotama se lahko prenašajo v obe smeri istočasno (full duplex). Najenostavnejša konfiguracija zahteva tri signale:

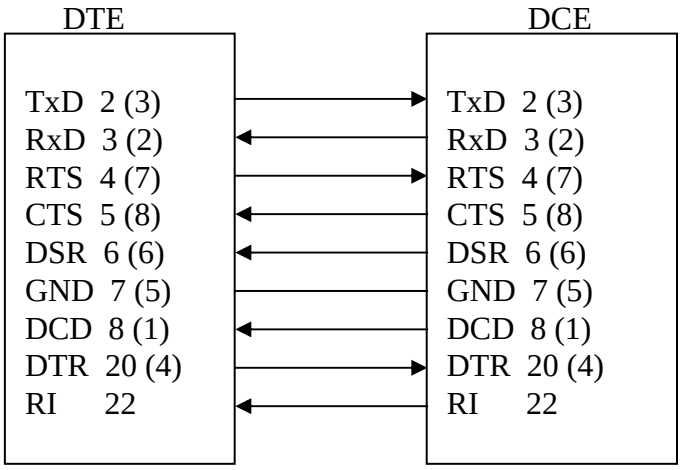
- TxD – Transmit Data (oddajanje podatkov)
- RxD – Receive Data (sprejemanje podatkov)
- GND – Ground (skupna signalna masa)

V tem primeru je prenos podatkov kontroliran s ti. Xon /Xoff načinom, ki softversko določa, kdaj je prenos dovoljen in kdaj ne.

Če je kontrola prenosa (handshake) izvedena hardversko, so potrtebne dodatne kontrolne in statusne linije. Narisan je primer, ko imamo DTE (data terminal equipment) in DCE (data

communication equipment) napravi, sicer moramo vse signalne linije prekrižati. Z oznako DTE označujemo naprave, ki sprejemajo, oddajajo in z danim protokolom omogočajo prenos podatkov (računalnik). Z oznako DCE pa imenujemo naprave, ki podatke posredujejo (modem).

- RTS (Request To Send) – signal, ki nadzoruje pretok podatkov iz DTE v DCE
- CTS (Clear To Send) – signal, ki nadzoruje pretok podatkov iz DCE v DTE
- DSR (Data Set Ready) – signal DTE enoti posreduje informacijo o pripravljenosti DCE
- DCD (Data Carrier Detect) – signal posreduje enoti DTE informacijo o napačno prenesenem podatku
- DTR (Data Terminal Ready) – signal, ki enoti DCE dopušča, da se lahko odzove na klic ali pa se priključi v omrežje
- RI (ring indicator) – signal pove DTE enoti, če je modem zaznal prihajajoč klic



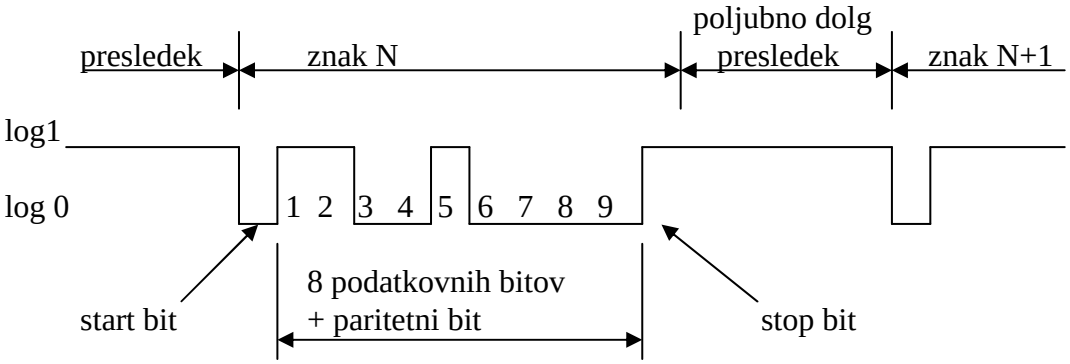
Napetostni nivoji so od –3 do –15V za logično 1 in +3 do +15V za logično 0. Na kontrolnih in statusnih linijahh je situacija obrnjena (+ potencial je logična 1). Hitrosti prenosa so od 75 do več 100kBaudov (bit/sek, bps - bits per seconds). Priporočljiva dolžina kabla je do 15 metrov.

5.2.1. Način prenosa

Podatke pošiljamo v obliki znakov. Vsak znak je sestavljen iz treh delov:

- startni bit
- podatkovni biti
- stop biti

Zaradi start in stop bitov se asinhroni način imenuje tudi start-stop način.



Start bit je vedno različen od nivoja mirovanja linije, ki je označeno kot 1. Njegov namen je, da sporoči sprejemniku začetek znaka. Podatkovni biti, ki jim lahko neobvezno sledi paritetni bit, določajo znak, ki se pošilja. Njihovo število je odvisno od nabora znakov (abecede) in je običajno 8. Na linijo se najprej pošlje bit z najmanjšo utežjo, zadnji pa bit z največjo. Stop bit je vedno enak mirovnemu stanju linije. Včasih imamo namesto enega stop bita dva. Stop bit zagotavlja, da bo pred naslednjim znakom linija v mirovnem stanju najmanj en bitni interval. Lahko mu takoj sledi naslednji znak, lahko pa je med njima poljubno dolg presledek, v katerem je linija v mirovnem stanju. Oddajnik in sprejemnik morata biti v času oddajanja znaka v sinhronizmu. Sprejemnik mora vedeti, s kakšno hitrostjo, koliko podatkovnih bitov, kakšno pariteto in koliko stop bitov pošilja oddajnik.

5.2.2. Pariteta

S pomočjo neobveznega paritetnega bita lahko ugotavljamo ali je prišlo pri prenosu znaka oz. podatkovnih bitov do podatkovne napake (motnje, neprimerna linija, prekinjajoča linija, nesinhronizem ...). Oddajnik in sprejemnik morata delovati v istem načinu paritete, ki je lahko soda, liha in brez paritete.

Oddajnik pri oddajanju prešteje število enic med podatkovnimi biti in postavi dodatni paritetni bit v stanje:

- liha pariteta
  - liho število enic --- paritetni bit v 1
  - sodo število enic --- paritetni bit v 0
- soda pariteta
  - liho število enic --- paritetni bit v 0
  - sodo število enic --- paritetni bit v 1
- brez paritete
  - paritetni bit ne obstaja

Sprejemnik pri sprejemu ponovno prešteje število enic v podatkovnih bitih, pregleda paritetni bit in ugotavlja resničnost glede na zgoraj napisana pravila. Če pri določeni pariteti paritetni bit kaže na obratno parnost kot pa je ugotovljena, sporoči preko posebne zastavice v statusnem registru napako pri prenosu. Ugotavljanje napake je pri določenih komunikacijskih adapterjih avtomatično, v nasprotnem primeru pa mora napako ugotoviti sam program.

5.2.3 Komunikacijski adapter za RS-232

Pri mnogih uK je komunikacijski adapter že vgrajen. Pravimo mu serijski port in operira z enakimi nivoji kot uK. Zaradi tega mu moramo v določenih primerih (komunikacija s PC) dodati element za pretvorbo nivojev (MAX232).
Večina proizvajalcev uP pa ponuja posebne elemente, ki jim pravimo asinhroni komunikacijski adapterji in so znani pod skupnim imenom UART (Universal Asynchronous Receiver Transmitter). Vsak UART vsebuje oddajnik in sprejemnik, ki sta med seboj neodvisna in lahko delujeta sočasno. Hitrost oddajanja in sprejemanja je določena z oddajno oz. sprjemno uro. S programom jim lahko določimo naslednje parametre prenosa:

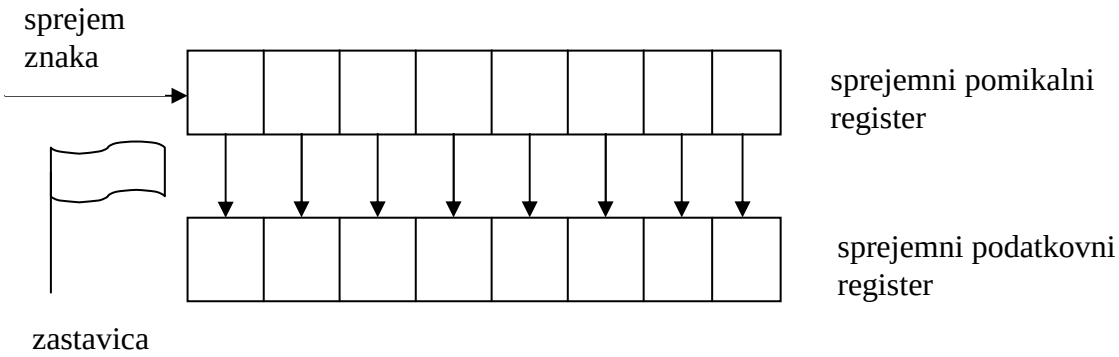
- način kontrole prenosa (softverski, hardverski)
- hitrost prenosa (110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200)
- pariteta (liha, soda, nobena)
- število podatkovnih bitov (5, 7, 8, 9)
- število stop bitov (1, 2)
- stanja na kontrolnih izhodih
- zahteva za prekinitev ob oddaji znaka
- zahteva za prekinitev ob sprejemu znaka

UART pri oddaji avtomatsko pretvori znak, ki ga dobi od uP v prenosni format za katerega je inicializiran. Pri sprejemu pa avtomatsko pretvori zaporedne bite v paralelno obliko in istočasno preveri pariteto in pravilno zgradbo zaporedja bitov.

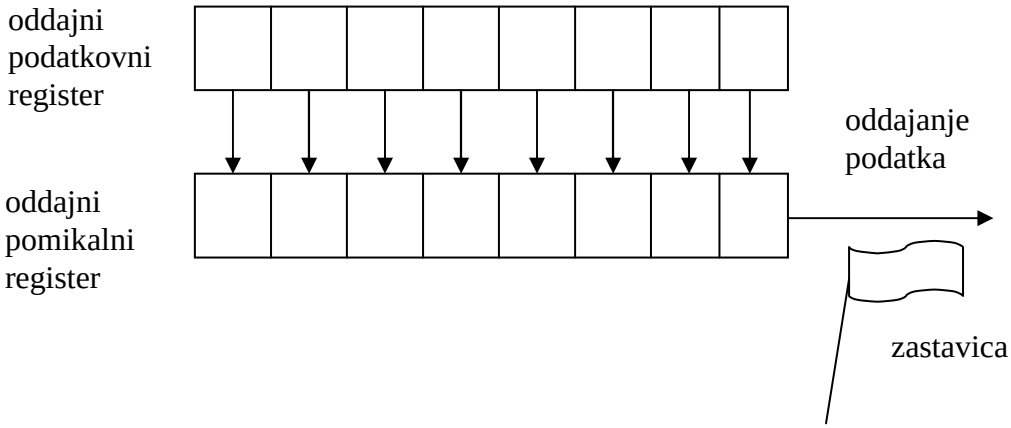
Pri sprejemu in oddaji največkrat uporablja ti. dvojno izravnavanje (double buffering). Pri sprejemu se biti pomikajo v sprejemni pomikalni register tako kot prihajajo. Ko so vsi biti znaka sprejeti se zgodi naslednje:

- podatek se avtomatsko prenese v sprejemni podatkovni register
- v statusnem registru se postavi zastavica, ki pove uP, da ga čaka prispeli podatek.

Sprejemnik lahko takoj prične sprejemati naslednji znak, ko je sprejemni pomikalni register prost.



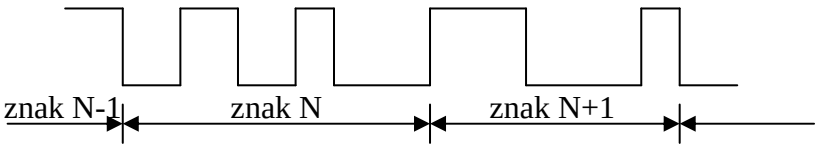
Pri oddaji uP vpiše podatek v oddajni podatkovni register , od koder se avtomatsko prenese v oddajni pomikalni register in prične oddajati bit za bitom. V oddajni podatkovni register lahko uP vpiše nov znak, ki čaka dokler se oddajni pomikalni register ne izprazni. Ko je oddan zadnji bit, se namreč postavi posebna zastavica.



Slaba lastnost asinhronskega prenosa je v slabi izkoriščenosti prenosne linije. Zaradi start in stop bitov je linija izkoriščena za 20% slabše kot bi sicer lahko bila.

5.3. Sinhronska komunikacija

Pri sinhronskem načinu prenosa nimamo start in stop bitov. Na liniji ni presledkov. Kadar oddajniku zmanjka podatkov prične avtomatsko vstavljati posebne mirovne znake, ki jih sprejemnik izpušča. Sprejemnik in oddajnik morata ostati v sinhronizmu preko celotnega sporočila. Ta so lahko dolga od nekaj znakov do nekaj tisoč znakov, zato morata biti frekvenci oddajnika in sprejemnika bistveno bolj natančni kot pri asinhronskem prenosu.



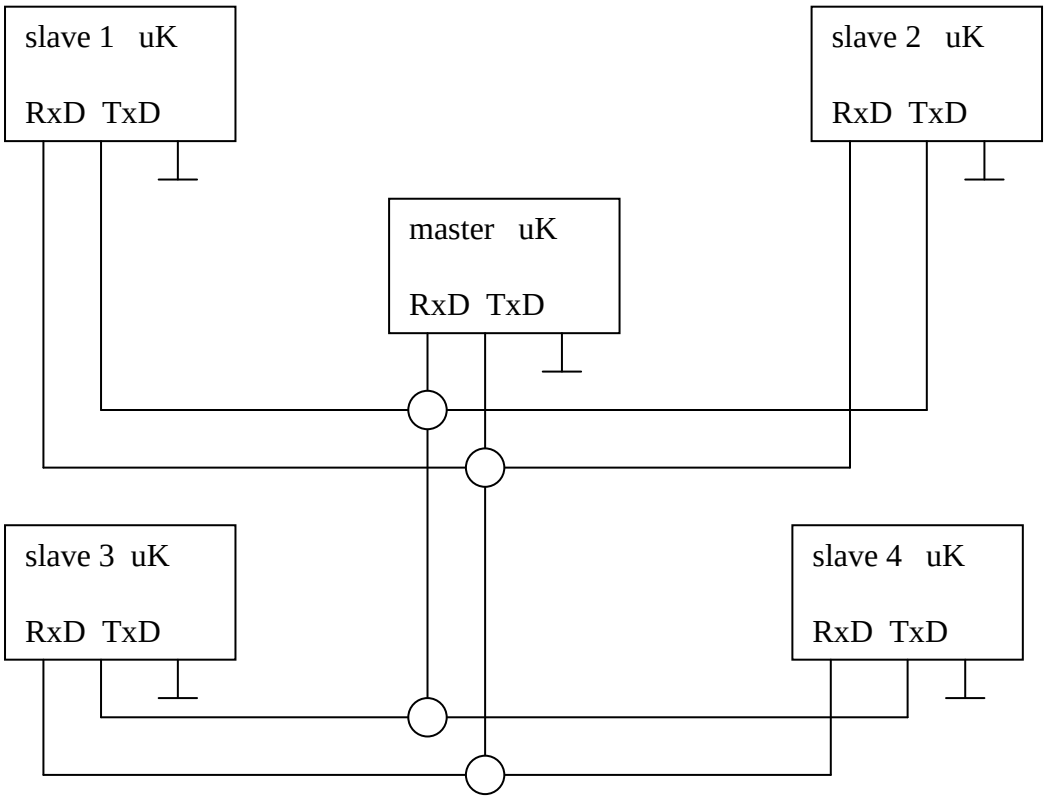
Poznamo dve glavni vrsti protokolov sinhronskega prenosa:

- znakovno orientirani protokol, kjer so sporočila grajena iz znakov neke abecede
- bitno orientirani protokol, kjer je sporočilo sestavljeno iz poljubnega zaporedja bitov, ki so lahko znaki v neki abecedi lahko pa tudi ne

V praksi se pri komunikacijah preko telefonskih linij skoraj vedno uporablja sinhronski prenos, kjer je hitrost prenosa običajno določena z modemom.

5.4. RS485 serijska komunikacija med več enotami

Pri takšnem komuniciranju je potrebno izbrati procesor, ki bo vodil "konferenco". Pravimo mu master uK.



Vsi ostali uK so podrejeni (slave) in vsak ima svoj naslov (šifro, kodo). Če želi master uK komunicirati z določenim slavom mora poslati najprej njegov naslov. Da je master poslal naslov, vidijo slavi po 9 podatkovnem bitu, ki se postavi v določeno stanje. Vsi slavi pregledajo naslov. V naslednjem koraku se lahko prične komuniciranje med masterjem in z naslovom določenim slavom, kar se spet vidi po 9 podatkovnem biti (njegovo stanje je nasprotno).
Slave uK ne morejo klicati master uK in ostalih slave uK.

6. Ostale enote periferije

Periferne enote funkcijsko obogatijo uporabnost uP. Nameščene so lahko skupaj z njim v integrirano celoto-uK ali pa jih dodamo na isto oz. dodatno kartico TIV. Med te enote spadajo:

- števci/časovniki
- vhodno/izhodne enote
- * D/A pretvornik
- * analogni komparator

- watch-dog timer (časovnik stražnik) * namenski IC-ji prilagojeni uP
- serijski adapter
- A/D pretvornik

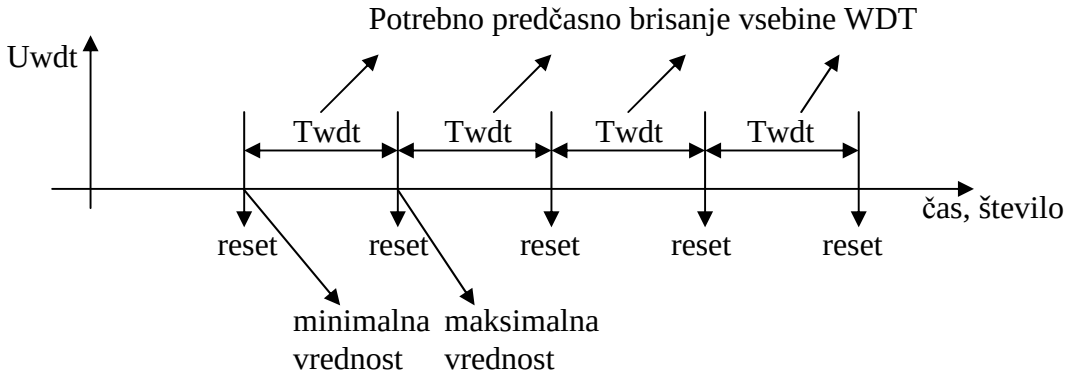
6.1. Watch-dog timer (časovnik-stražnik) WDT

Je časovnik, ki povzroča hardvarski reset uK, če doseže maksimalno vrednost. S tem zavarujemo mikroračunalnik pred programskim zazankanjem, ki je lahko posledica :

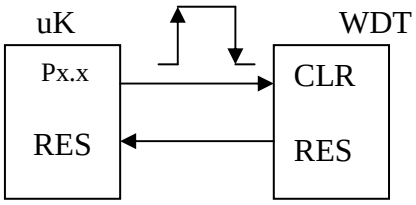
- nespretno napisanega programa
- napake pri vodenju uP skozi program (motnje na vodilih).

Da nam ta časovnik ne bi periodično resetiral uP pri normalnem delovanju, je potrebno njegovo vsebino izbrisati še preden doseže maksimalno vrednost. Njegovo varovanje je najbolj učinkovito, če mu vsebino izbrišemo v glavni zanki programa in mu s tem javljamo, da je program še vedno "živ". V primeru, da je perioda WDT-ja krajša od enega cikla programa, moramo njegovo brisanje izvršiti na dodatnih primernih mestih v programu. Zaradi tega je njegova zanesljivost zmanjšana.

WDT je lahko 8 do 16 bitni števec, ki šteje impulze določenega generatorja takta. Nameščen je lahko znotaj uK, njegovo vrednost brišemo s programskim vnosom vrednosti 0 v števni register.



Druga možnost je posebno IC vezje WDT, ki je lahko združeno še z vezjem za reset in preklop na pomožno napajanje. To vezje priklopimo na uK preko vhodnega signala za reset in poljubnega izhodnega digitalnega signala Px.x. Stanje zadnjega je potrebno spremeniti, pred iztekom Twdt, kajti sprememba stanja tega signala zbriše vsebino števnega registra WDT.



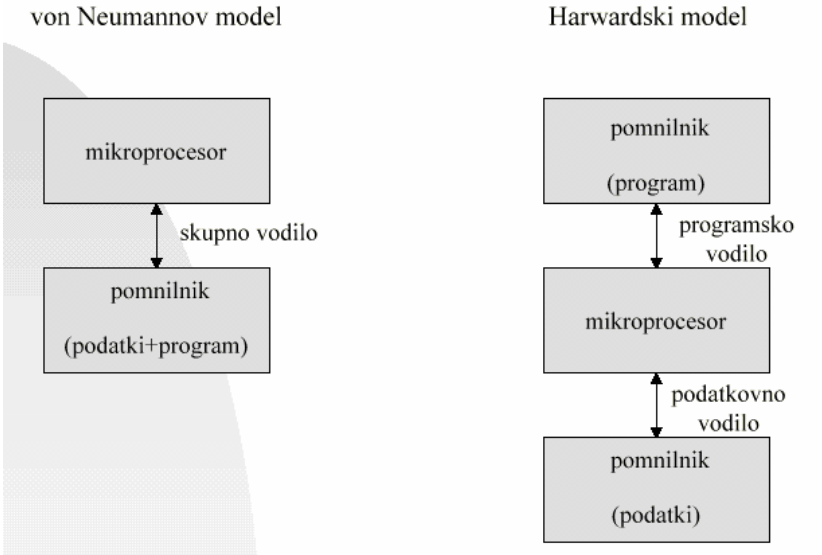
Dodatki:

1. Zgradba mikroprocesorjev

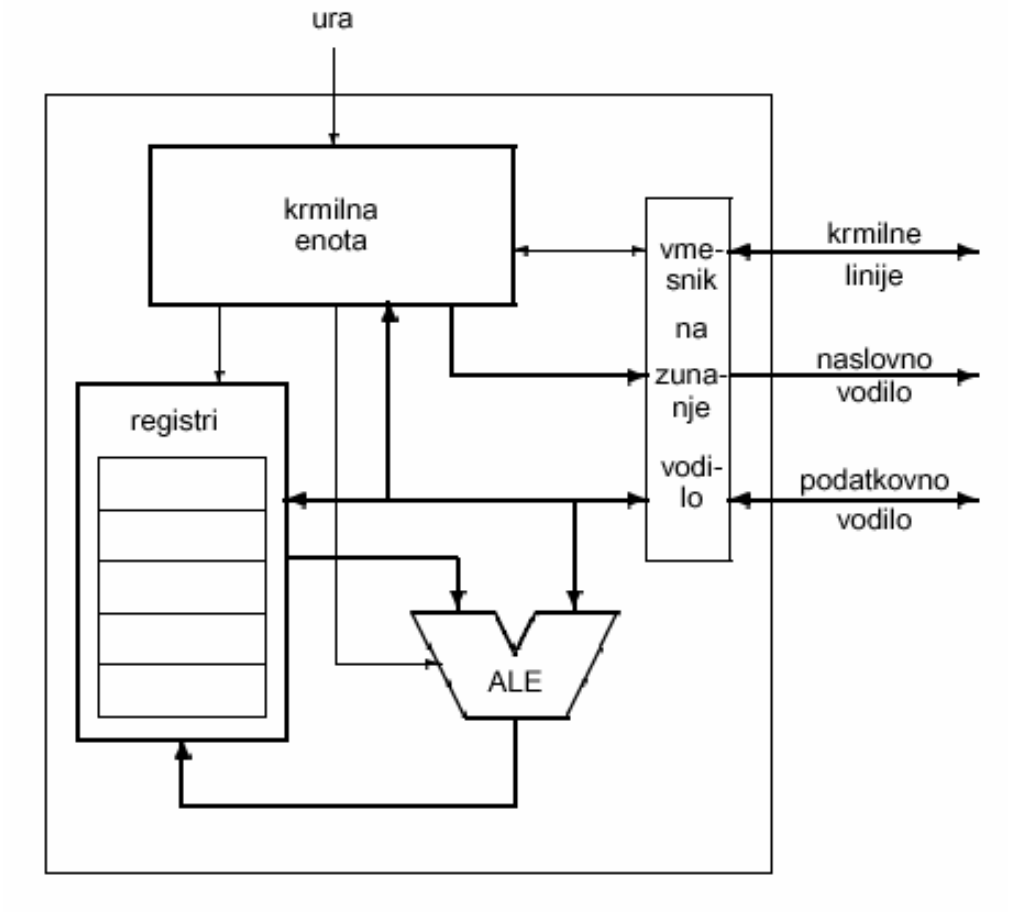
Za razumevanje delovanja mikroprocesorjev bomo v tem poglavju postavili preprost model, v katerem bomo upoštevali le njihove najbolj splošne in pomembne funkcije. Resnična izvedba tipičnega mikroprocesorja je veliko bolj kompleksna; gradnikov je več, v njih se prepletajo funkcije različnih funkcionalnih enot.
.
Osnovne funkcije mikroprocesorja lahko povzamemo zelo na kratko: prenos, hranjenje in obdelava podatkov, kakor zahteva uporabniški program. Gledano funkcionalno, sestavljajo model mikroprocesorja tri enote: krmilna enota, aritmetičnologična enota in nabor registrov.

7.1 Model mikroprocesorja

Arhitekture mikroprocesorjev se zelo razlikujejo med seboj, večina pa jih temelji na von Neumannovem modelu, kjer se podatki in programska koda nahajajo skupaj v istem pomnilniškem naslovnem področju. To je bila nesporno ustrezna in zelo koristna rešitev v časih, ko je obdelava ukazov zahtevala bistveno več časa kot njihov prenos. V zadnjem času se je ozko grlo preselilo na slednje. Von Neuman-nova arhitektura je postala šibka točka, ki jo predvsem v RISC mikroprocesorjih nadomešča harwardska: ta ima ločeni pomnilniki področji in podatkovne poti za podatke in ukaze. Prenos enih in drugih se zato lahko vrši popolnoma vzporedno.



Arhitekturo mikroprocesorja lahko v grobem razdelimo na tri glavne sestavne dele: krmilno enoto, aritmetično-logično enoto in nabor registrov.



Najpomembnejši del mikroprocesorja predstavlja krmilna enota, ki usklajuje delo-vanje posameznih delov mikroprocesorja po navodilih programske kode. Krmilna enota bere ukaz za ukazom iz programa, zapisanega v strojni obliki, jih dekodira in preko množice krmilnih signalov krmili njihovo izvajanje.

Za izvajanje operacij nad podatki, ki jih obdeluje program, skrbi aritmetično-logična enota (ALE). Ta je sposobna opraviti različne matematične in logične operacije nad podatki različnih tipov.

Registri sluzijo za začasno hranjenje in obdelavo podatkov in njihovih naslovov. Izvedeni so kot zelo hitre pomnilne celice (običajno 5 do 10 krat hitrejše od zu-nanjega – delovnega). Imamo več vrst registrov, od katerih so nekateri dostopni programerju, drugi pa so namenjeni internemu delovanju mikroprocesorja. Neka-teri registri nastopajo tudi kot del krmilne oziroma aritmetične in logične enote. Posamezne enote mikroprocesorja so med seboj povezane preko več internih vodil. Širina teh vodil običajno po širini ustreza velikosti registrov. Prav tako je mikroprocesor je z okolico povezan preko zunanjih vodil, ki jih pri večini mikroprocesorjev sestavljajo:

naslovno vodilo, ta določa naslov pomnilniške lokacije, do katere želi mikroprocesor dostopati;

podatkovno vodilo, preko njega poteka izmenjava vsebine (programskih ukazov in podatkov) med registri in celicami pomnilnika, torej v ali iz mi-kroprocesorja;

ter

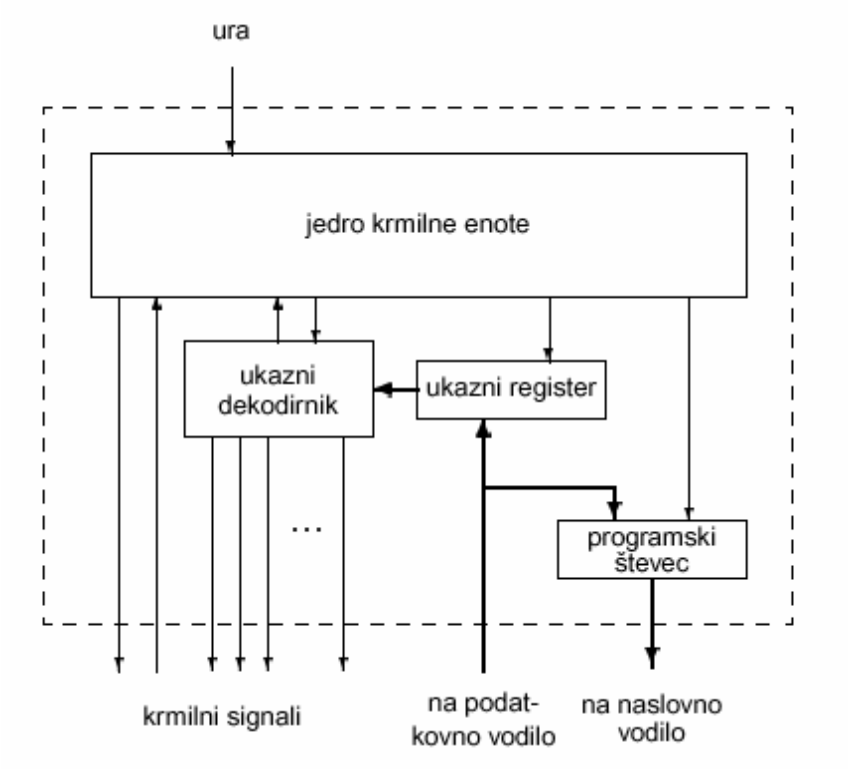
krmilne linije, ki krmilijo komunikacijo z okolico (povejo na primer, ali zeli mikroprocesor brati ali pisati v pomnilniško področje).

Zaradi tehnoloških omejitev so bile širine zunanjih vodil pri starejših mikroprocesorjih pogosto manjše od širine internih vodil. To je pomenilo, da je bilo potrebno za prenos ene besede izvesti dva ali več bralnih oz. pisalnih ciklov. V sodobnih mikroprocesorskih arhitekturah težimo k čim večji prepustnosti podatkov in nimajo takšnih omejitev. Zunanje in notranje vodilo sta povezana preko posebnega vmesnika, katerega delovanje je pod nadzorom krmilne enote.

Nekateri preprostejši mikroprocesorji in mikroračunalniki imajo programski in podatkovni pomnilnik vgrajen na isti silicijevi rezini. Takšni mikroprocesorji se-veda zunanjega naslovnega in podatkovnega vodila ne potrebujejo.

7.1.1 Krmilna enota

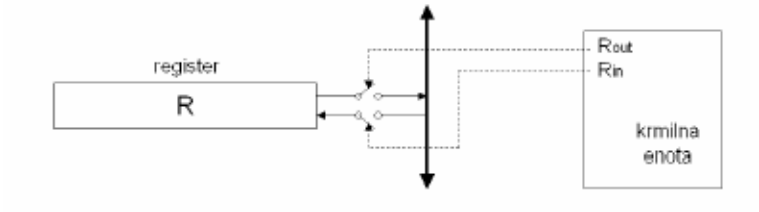
Najpomembnejši del mikroprocesorja je krmilna enota. To je sinhrono vezje, ki deluje po taktu sistemske ure (system clock). Preko posebnih krmilnih linij, po katerih se ne prenašajo podatki, upravlja z delovanjem posameznih pod-enot in krmili komunikacijo z okolico mikroprocesorja.
Na spodnji sliki je prikazan primer, ko krmilna enota krmili dostop do enega iz-med registrov. Potrebna sta dva signala. Prvi skrbi za prenos podatkov v register (R_{IN}), drugi pa za prenos podatkov iz registra (R_{OUT}).



Podobni signali vodijo v ALE in določajo katera aritmetično/logična operacija se

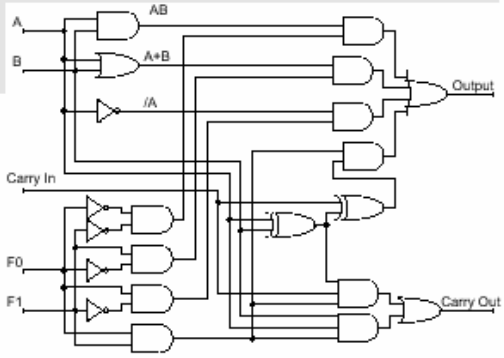
mora izvesti.

Na naslednji sliki je prikazana podrobnejša zgradba krmilne enote.



Osnovo krmilne enote predstavlja njeno jedro, ki vodi njeno delovanje. Jedro je tesno povezano z dekodirnikom ukazov ter ukaznim registrom. V ukazni register se zapiše koda ukaza, ki ga mora mikroprocesor izvesti, dekodirnik ukazov pa v skladu s tem ukazom generira krmilne signale in tako poskrbi za njegovo izvedbo. Naslov pomnilniške lokacije, v kateri se nahaja naslednja koda ukaza za izvajanje, je podan v programskem števcu (PŠ). Vrednost tega registra spreminja običajno le krmilna enota. Praviloma deluje tako, da se programski števec med dekodiranjem vsakega ukaza avtomatsko poveča in kaze na naslednji ukaz v programu. Programer spreminja njegovo vrednost samo preko ukazov skokov in klicev pod-programov. Ob vklopu se vrednost PŠ postavi na vnaprej definirano vrednost in kaze na začetek zagonskega dela programa.

Tukaj prikazujemo model logičnega vezja, ki predstavlja preprosto enobitno aritmetično/logično enoto; ta zna izračunati logično konjunkcijo (IN), disjunkcijo (ALI), negacijo in aritmetično vsoto dveh bitov A in B z upoštevanjem prenosov. Želeno funkcijo bi krmilna enota izbirala s krmilnima signaloma F0 in F1. S paralelno vezavo bi lahko dobili ALE za poljubno dolžino besede. Prave ALE so seveda neprimerno bolj kompleksne.



F_0	F_1	Output	CarryOut
0	0	$A \wedge B$	0
0	1	$A \vee B$	0
1	0	$\neg A$	0
1	1	A+B+CarryIn	CarryOut

7.1.2 Aritmetično-logična enota

Za izvajanje operacij nad operandi ukazov mikroprocesorja skrbi *aritmetična/logična enota, ALE* (Arithmetic/Logic Unit). Sodobni procesorji imajo običajno več aritmetično logičnih enot, ki skrbijo za računanje z različnimi tipi podatkov: ene skrbijo za celoštevilčne podatke, druge pa za računanje s podatki s pomično vejico. Nekatere ALE so sposobne izvajati tudi kompleksnejše računske operacije, kot so npr. logaritmi, kotne funkcije ipd. Nekatere od njih so prirejene tudi za izvajanje operacij posebnega tipa npr. za obdelavo signalov,slik ipd.

ALE je običajno izvedena je kot čisto preklopno vezje, ki operira nad enim ali dvema vhodnima podatkoma in generira tretjega – rezultat. Krmilna enota s krmilnimi signali izbira poljubne aritmetične ali logične operacije nad njimi, ALE pa poleg rezultata generira še dodatne informacije v obliki t.i. zastavic (flags) o tem, ali je bil rezultat zadnje operacije negativen ali enak nič, ali je pri operaciji prišlo do prekoračitve območja in podobno. Glede na te informacije lahko krmilna enota izvede ali ne pogojne skoke in podobne ukaze. Podrobnejši opis operacij, ki jih lahko izvaja ALE, in pomenov zastavic je podan v naslednjem poglavju.

7.1.3 Registri

Registri so začasne shrambe za podatke, največkrat izvedene na samem čipu in z zelo kratkim dostopnim časom. Njihova izbira poteka neposredno s krmilnimi signali in ne z naslavljanjem kot pri zunanjem pomnilniku.

V mikroprocesorju nastopa več vrst registrov; v grobem jih delimo na programerju dostopne in nedostopne. Osnovne skupine registrov, ki jih srečamo v skoraj vseh mikroprocesorjih so naslednje:

Podatkovni registri, ki služijo za vmesno hranjenje podatkov med obdelavo. Tipični podatkovni registri so bili pri zgodnjih mikroprocesorjih t.i.
akumulatorji ali zbirni registri; ti so sluzili pri aritmetično/logičnih operacijah za nosilce enega vhodnega operanda in rezultata. Namesto akumulatorjev imajo sedaj mikroprocesorji več neodvisnih podatkovnih registrov.
Podatkovni registri običajno hranijo celoštevilčne informacije, v novejših mikroprocesorjih pa srečamo tudi posebno skupino podatkovnih registrov, ki so namenjeni za izvajanje operacij nad števili s pomično vejico.

Naslovni registri vsebujejo naslove operandov ali dele, iz katerih se le–ti izračunajo. Naslovni registri nastopajo v različnih načinih naslavljanja, ki bodo podrobneje opisana v naslednjem poglavju.

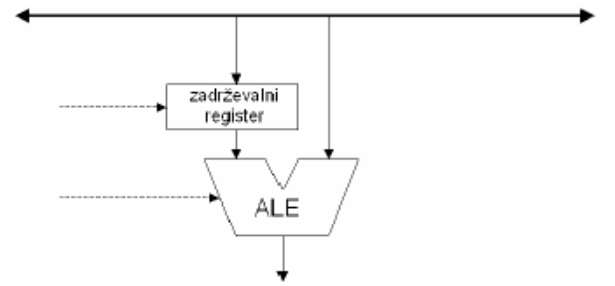
Posebni registri imajo v mikroprocesorjih ozko specialne funkcije. Nekateri med njimi, t.i. specialni registri, so programerju nedostopni oz. le deloma dostopni in sluzijo za pomožne shrambe. K tem prištevamo zgoraj opisana pomožna registra v procesni enoti, ukazni register in programski števec.
V zgodneših tipih mikroprocesorjev so določeni registri nastopali le v določenih ukazih mikroprocesorja. Danes težimo k čim večji univerzalnosti. Tako v sodobnih mikroprocesorjih ne ločimo več podatkovnih in naslovnih registrov in lahko vse splošno namenske registre uporabljamo za ene ali druge namene. Razvijalci CISC mikroprocesorjev so težili k čim večjemu številu registrov, da bi lahko hitro posegali do čim več spremenljivk. Vendar je to tedanja tehnologija to onemogočala.Dodatni problem je predstavljala tudi širina ukaza. V osem bitnih mikroprocesorjih lahko identifikaciji določenega registra namenimo le manjše število bitov.
Ko je tehnologija omogočila na čipu realizirati veliko registrov in ko se je veli-kost operacijske kode povečala, je nastopil problem razporejanja spremenljivk na

registre: kako čim bolj optimalno razporediti spremenljivke programa, posebej v višjem programskem jeziku, v posamezne registre, tako da bo dostop do njih kar najhitrejši. Optimizacija uporabe registrov je zapletena, posebej, ko gre za multi-programiranje ali rekurzivne klice programov. Zaradi tega jo običajno prepustimo prevajalniku.

7.2 Prenos podatkov znotraj mikroprocesorja

Prenos podatkov znotraj mikroprocesorja poteka preko internih vodil. Glede na izvedbo ločimo dva načina prenosa podatkov:

Skupno notranje vodilo. Pri takšni organizaciji poteka celotni promet znotraj mikroprocesorja po eni skupini povezav. Težave nastopijo v primeru,če je potrebno izvesti operacijo ki zahteva več kot en operand (npr. seštevanje dveh števil). V tem primeru moramo izvesti operacijo v dveh korakih kot je to prikazano na spodnji sliki.



V prvem koraku vsebino prvega operanda prenesemo v posebni zadrževalni register s čemer ga pripravimo za izvedbo operacije. V drugem koraku po vodilu prenesemo vrednost drugega operanda in izvedemo operacijo.

Ločena notranja vodila Seveda je realizacija prenosa podatkov po skupnem vodilu neučinkovita, vendar se je pri zgodnejših mikroprocesorjih zaradi tehnoloških omejitev pogosto uporabljala. Zaradi težnje po čim hitrejšem delovanju, uporabljajo vse novejše mikroprocesorske arhitekture več notranjih vodil za komunikacijo med posameznimi segmenti mikroprocesorja.
Zgled uporabe takšnega pristopa bo podan v naslednjem podpoglavju.

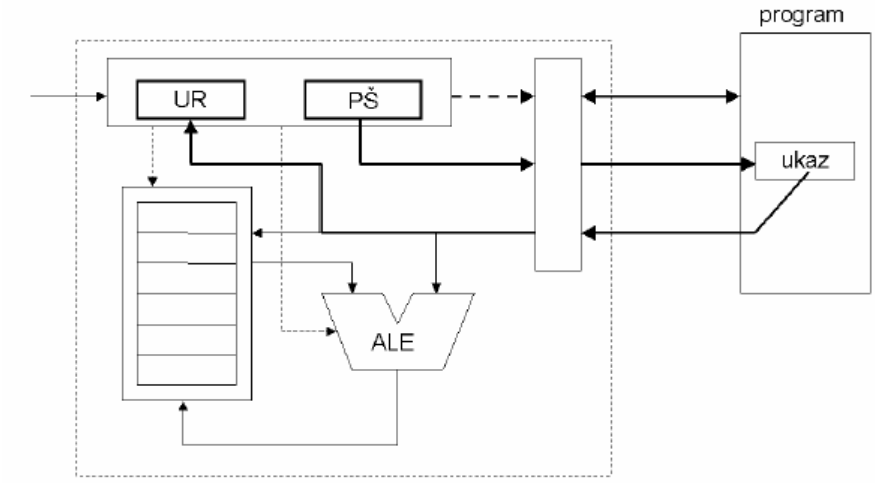
7.3 Delovanje mikroprocesorja

V tem podpoglavju si bomo na primeru hipotetičnega mikroprocesorja ogledali osnovne korake v njegovem delovanju. Model smo zaradi lažjega razumevanja zelo poenostavili. čeprav sodobni mikroprocesorji temeljijo na podobnem modelu delovanja, pa je njihova zgradba veliko kompleksnejša. Spodaj opisane faze delovanja se v vseh sodobnih mikroprocesorjih medsebojno prepletajo, več faz (nad različnimi ukazi) se izvaja vzporedno, itd. Najprej bo delovanje mikroprocesorja predstavljeno bolj splošno, na koncu pa bomo podrobneje predstavili še podrobnejši primer izvedbe hipotetičnega mikroprocesorja.

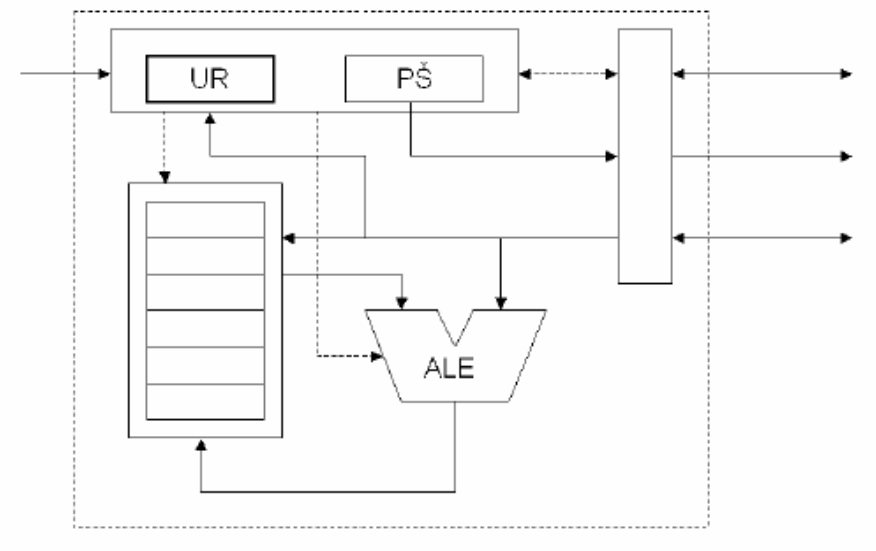
7.3.1 Faze delovanja

Izvajanje ukazov mikroprocesorja lahko razdelimo na več zaporednih faz:

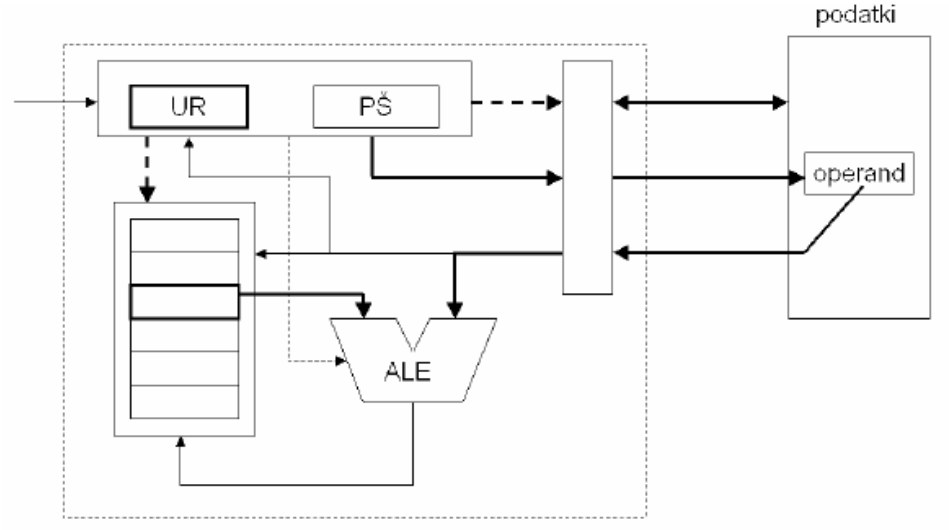
1. *Prevzem ukaza*
V tej fazi jedro krmilne enote poskrbi, da se vrednost programskega števca prenese na naslovno vodilo in s podatkovnega vodila prečita vrednost z lokacije, na katero kaze. Prečitana vrednost se prenese v ukazni register. Po opravljenem prenosu se vrednost programskega števca poveča tako, da kaze na naslednji ukaz.



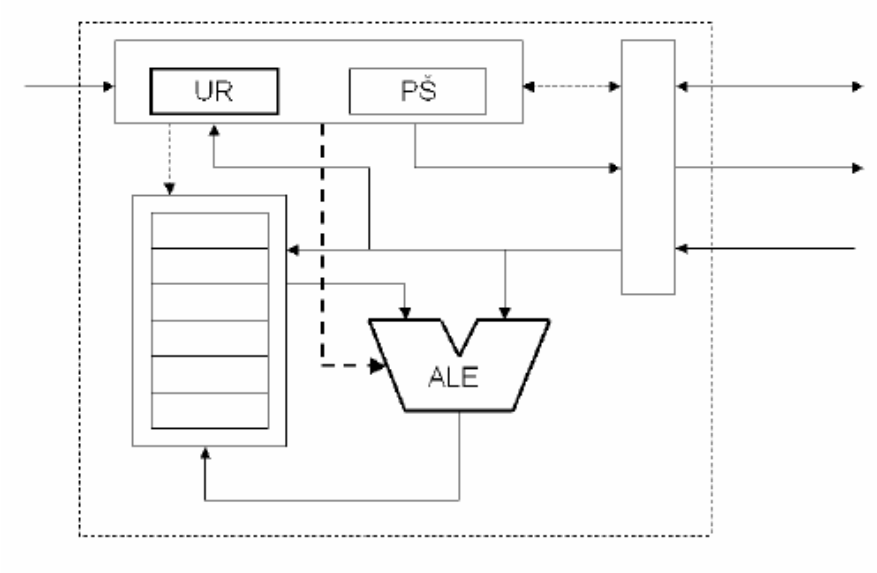
2. *Dekodiranje ukaza*
Vrednost, zapisano v ukazni register, prevzame ukazni dekodirnik. Ukaz se prevede v sekvenco ustreznih krmilnih signalov, ki vklapljajo in izklapljajo posamezne dele mikroprocesorja. Ta sekvenca signalov se izvede v naslednjih fazah delovanja.



3. *Priprava parametrov*
če zahteva ukaz vhodne parametre (podatke), jih je potrebno predhodno pripraviti. če se podatek nahaja v zunanjem pomnilniku, je potrebno na naslovno vodilo pripeljati ustrezni naslov in s podatkovnega vodila prečitati njihovo vrednost. če pa se parameter nahaja v registru, mora krmilna enota samo odpreti vrata med vodilom in enim izmed registrov. čitanje zunanjih operandov običajno poteka poteka bistveno dalj časa od čitanja vrednosti registrov. Ta čas bi bil še daljši, če bi za izvedbo operacije potrebovali dva operanda iz zunanjega pomnilnika. V tem primeru bi potrebovali dodatni zadrževalni register (kot v primeru skupnega internega vodila za prenos podatkov), sama priprava parametrov pa bi zahtevala še en dodaten korak.

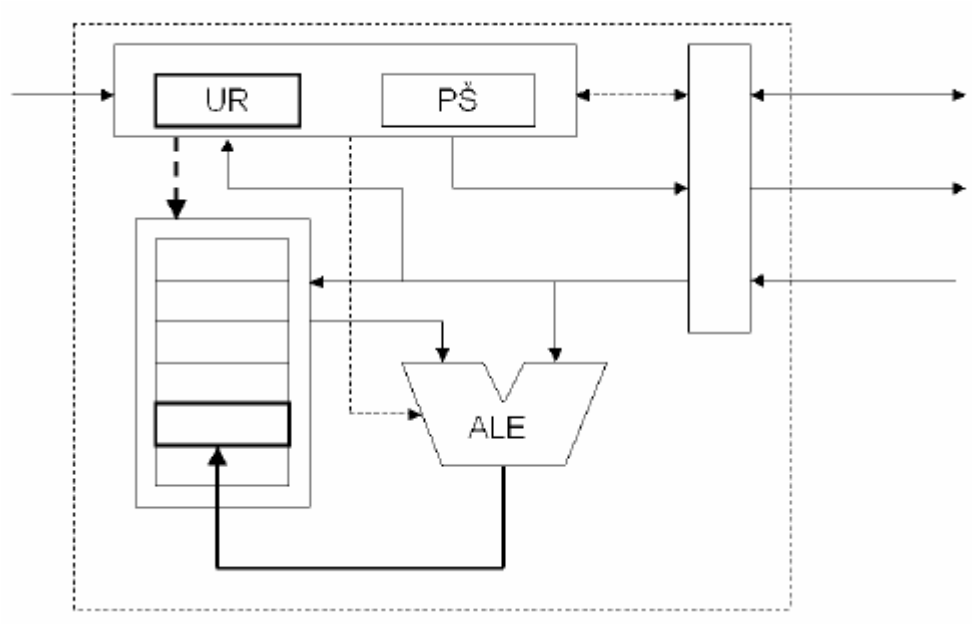


4. *Izvedba ukaza*
če gre za ukaze za obdelavo podatkov, za njegovo izvedbo običajno poskrbi ALE. Na vhodu sprejme do dva parametra in na izhodu generira rezultat. Tudi delovanje ALE je pod nadzorom ukaznega dekodirnika.



V določenih primerih ta izvajalna faza ni potrebna. če smo zeleli samo prenesti vsebino pomnilniške lokacije v enega izmed registrov, lahko to fazo izpustimo.

5. *Shranjevanje rezultatov*
Vrednost rezultata je potrebno shraniti. če je cilj operacije zunanji pomnilnik, je potrebno na naslovno vodilo pripeljati ustrezni naslov, na podatkovno vodilo pa prenesti izhodno vrednost ALE. če je cilj operacije register, krmilna enota interno poveze izhod ALE z ustreznim registrom. V našem hipotetičnem modelu predpostavljamo, da je možno rezultat aritmetično/logične operacije shraniti samo v enega izmed registrov.

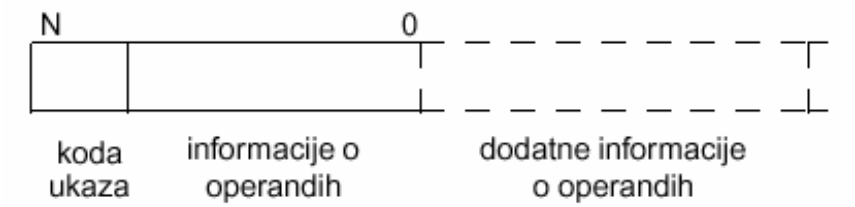


Fazi 3 in 5 sta odvisni od vrste ukaza in nista zmeraj prisotni. Da bi pospešili njihovo

delovanje, so skoraj vsi sodobni mikroprocesorji organizirani tako, da se več ukazov znotraj mikroprocesorja obdeluje sočasno. Ko se npr. izvaja dekodiranje enega ukaza, se sočasno že izvaja čitanje naslednjega ukaza. Podrobneje bomo o tej tehniki, imenovani cevenje (pipelining) in o drugih tehnikah za pohitritev delovanja mikroprocesorja govorili v 9. poglavju. Naš hipotetični model mikroprocesorja je zgrajen po filozofiji RISC mikroprocesorjev: vse aritmetično/logične operacije se izvajajo samo nad vsebino registrov; edina ukaza za povezavo z zunanjim podatkovnim pomnilnikom sta ukaza za shranjevanje in čitanje podatkov.

7.3.2 Oblika strojnih ukazov

Ukazi mikroprocesorja so zapisani v strojni obliki (binarno). Ukaz je lahko se-stavljen iz ene besede ali več (razširitvenih) besed. Velikost besede je običajno enaka velikosti zunanjega podatkovnega vodila, lahko pa je tudi manjša.



V prvi besedi se običajno nahajata koda ukaza in informacija o operandih, v raz-širitvenih besedah pa so podane dodatne informacije o operandih (npr. naslov operanda v pomnilniku).

Krmilna enota najprej izlušči informacijo o operandih ukaza in jih po potrebi pre-nese v pomozne reistre ali na notranje vodilo procesorja, nato pa na osnovi kode ukaza sprozi potrebno zaporedje signalov za njegovo izvedbo.

CISC mikroprocesorji uporabljajo ukaze različne dolzine (z različnim številom besed v operacijski kodi), RISC mikroprocesorji pa tezijo k ukazom dolzine ene besede. S tem se poenostavi dekodiranje ukazov in izveča učinkovitost mikropro-cesorja saj ne potrebujemo večkratnih posegov v programski pomnilnik. Nekatere mikroprocesorske arhitekture (npr. Intel) uporabljajo tudi t.i. predpone. To so po-sebne kode, ki jih navedemo pred ukazom in ki spremenijo njegov pomen. Na ta način so omogočili razširitev nabora ukazov pri novejših verzijah mikroprocesor-jev in hkrati ohranili kompatibilnost strojne kode s starejšimi verzijami.

S povečanjem dolzine besede na 64 bitov se v sodobnih mikroprocesorskih ar-hitekturah pojavlja tudi moznost predstavitve več ukazov znotraj iste besede. V tem primeru se več ukazov mikroprocesorja dejansko izvede vzporedno. Seveda je zato potrebna tudi ustrezna mikroprocesorska arhitektura. Potrebujemo pa tudi poseben prevajalnik, saj ukazov ne moremo poljubno zdruzevati. Veliko krat je potrebno, da se izvede najprej en ukaz in šele zatem se lahko izvede drugi.

7.3.3 Implementacija krmilne enote

Krmilna enota sodobnih mikroprocesorjev je lahko zasnovana na dva načina: s *kombinacijsko logiko* ali na *principu mikroprogramiranja*. Večina zgodnjih mikroprocesorjev je uporabljala *kombinacijsko logiko*. Pri tem načinu se vsi ukazi interpretirajo z logičnimi vezji. Dekodirnik ukazov je sestavljen iz dekoderja, ki prepozna ukaz in ga razgradi v komponente, ter enkoderja, ki na njihovi osnovi generira krmilne signale za krmiljenje posameznih enot mikroprocesorja.

Mikroprocesorji iz skupine CISC so največkrat *mikroprogramirani*. Ta tehnika se je razvila iz zelje po bogatem naboru kompleksnih ukazov, ki jih je bilo težko izvesti v diskretnih vezjih. Uporabili so osnovno idejo, ki je botrovala nastanku mikroprocesorjev: za izvedbo kompleksnih digitalnih vezij izdelamo univerzalno vezje, katerega obnašanje programiramo.

Dekodirnik ukazov deluje kot mikroprocesor v malem, saj vsebuje tako *mikroprogramsko kodo* kot tudi *mikroprogramski števec*. Njegova izvedba je zasnovana na preprosti kombinacijski logiki. Na ta način se en ukaz mikroprocesorja prevede v sekvenco mikro ukazov, ki so shranjeni kot *mikroprogram ali mikrokoda* v ROM-u (*firmware*). Mikro ukazi so precej enostavnejši od samih ukazov mikroprocesorja. Namesto operandov so deli mikro ukazov navodila za generiranje krmilnih signalov za ostale enote znotraj mikroprocesorja.

Tehnologija mikrokodiranih mikroprocesorjev omogoča kompleksnejše nabore ukazov in naslavljanj. Nabor ukazov lahko proizvajalec relativno preprosto spre-meni ali razširi z zamenjavo vsebine mikro ROM-a brez spremembe kombinacijske logike. Ob revizijah pri izdaji nove verzije procesorjev preprosteje optimizirajo mikroprogramirano izvajanje ukazov. Tudi snovanje procesorja je preprostejše, hitrejše in ima jasno strukturo, posamezne komponente se lahko razvijajo vzporedno, napak je zaradi manjše kompleksnosti manj.

Obstajajo tudi *mikroprogramirljivi procesorji*, pri katerih je mikroprogram shranjen na posebnem programirljivem notranjem ali zunanjem pomnilniku in ga je mogoče spreminjati, graditi svoje ukaze ipd. Takšne tipe procesorjev danes le še redko srečujemo.

Slabost mikroprogramirane izvedbe izvajanja ukazov je v tem, da je delež krmilne enote na čipu v primerjavi z aktivnimi komponentami (ALE, registri) velik (v kompleksnih CISC mikroprocesorjih tudi več kot 60%), dekodiranje pa zaradi razkošnega nabora ukazov razmeroma počasno. Za izvršitev enega ukaza je potrebnih mnogo korakov. Zato so se v sodobnih mikroprocesorjih tipa RISC omejili na najpotrebnejše ukaze in tako spet omogočili izvedbo s kombinacijsko logiko, ki zavzema največ 10% površine čipa. Ob preprostejših optimirajočih prevajalnikih so s tem dosegli bistveno večje hitrosti izvajanja istih programov v višjem programskem jeziku.
V sodobnih mikroprocesorskih arhitekturah je krmilna enota lahko zelo kompleksna in organizirana tako, da se lahko več ukazov izvaja sočasno.

8. Sodobni pristopi pri izdelavi mikroprocesorjev

- tehnološki pristopi (pospeševanje ure, povečevanje gostote, nižanje napajalne napetosti)
- Uporaba RISC filozofije
- Predpomnilnik
- Prekrivanje faz delovanja (cevenje)
- Vzporedno izvajanje ukazov (ILP- instruction-level parallelism)
- Koprocesorji in posebni ukazi
- Večprocesorski sistemi
- Trendi razvoja
- Pregled lastnosti nekaterih sodobnih mikroprocesorjev

Tehnološki pristopi

- Povečanje gostote elementov na isti površini silicij: sedaj 0.25µ
- Povečanje števila elementov mikroprocesorja

  Mooreov zakon:Število tranzistorjev na mikroprocesorju se podvoji vsakih 18 mesecev.

4004	2300 tranzistorjev
Pentium III	$8{,}2*10^6$ tranzistorjev

Nižanje napajalne napetosti

- Omogoča hitrejši preklop med logičnim stanji
- Manjša poraba energije

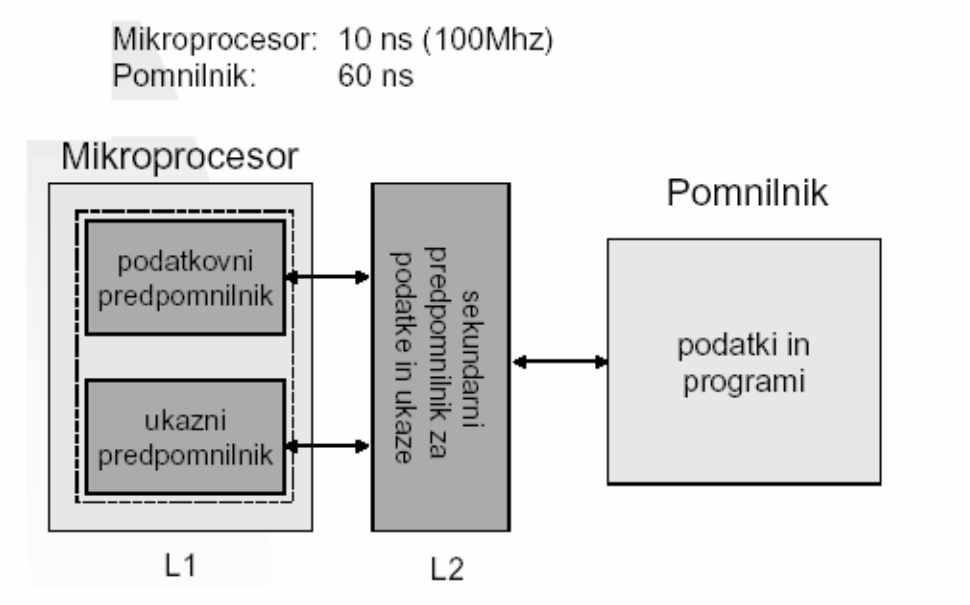
Pentium III	3,3V/1,5V

Uporaba RISC filozofije

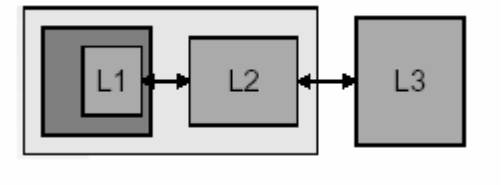
CISC	RISC
– bogat nabor ukazov različne širine	– omejen nabor ukazov enake širine
– bogat nabor naslavljanj	– omejen nabor naslavljanj
– kompleksna implementacija	– enostavna implementacija
– počasnejše izvajanje	– zelo hitro izvajanje
– enostavnejše prevajanje	– kompleksno prevajanje
– krajša koda	– obsežnejša koda

Predpomnilnik (cache)

- Izravnajo nesorazmerje med hitrostjo delovanja jedra mikroprocesorja in pomnilnika

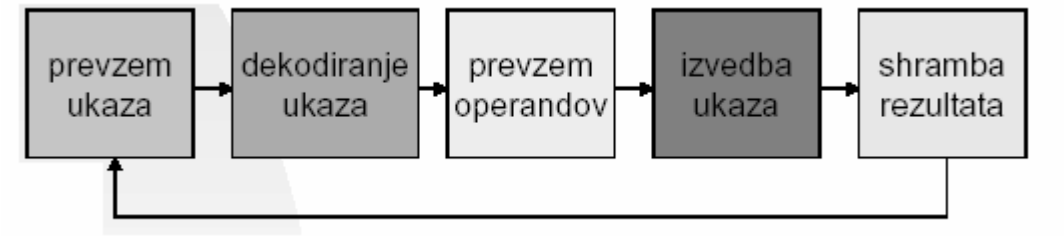


- Omogoča sočasni pristop do podatkov in ukazov (Harvardska arhitektura)
- Sodobni mikroprocesorji imajo sekundarni predpomnilnik vgrajen v istem ohišju



Prekrivanje faz delovanja (cevenje)

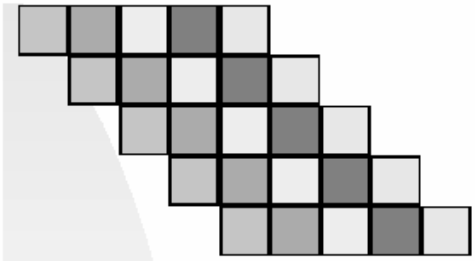
- Izvedba ukaza znotraj mikroprocesorja poteka v več zaporednih fazah



- Vsaka faza se mora izvesti do konca preden se lahko začne izvajati naslednja

⇒ izvedba enega ukaza traja 5 urinih ciklov
⇒ izvedba petih ukazov traja 25 urinih ciklov

- Z uporabo cevenja se istočasno obdeluje več ukazov hkrati

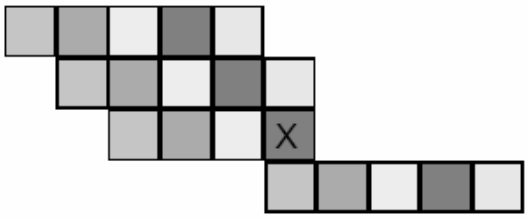


⇒ izvedba enega ukaza traja 5 urinih ciklov
⇒ izvedba petih ukazov traja 9 urinih ciklov

- Za vsako fazo skrbi ena enota znotraj mikroprocesorja – enote delujejo vzporedno

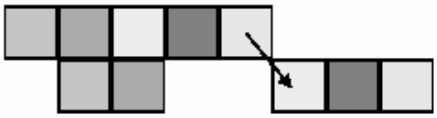
- **Zastoji pri cevenju**
- (pogojni) skok

```
cmp   r0,r1
jz    enaka
mov   r1,r2
```



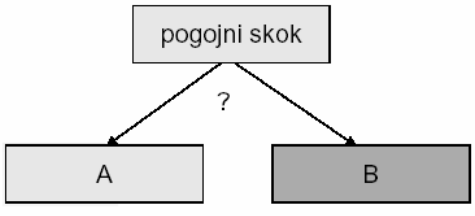
- **Odvisnost med ukaza**

```
add   r0,r1   ; r1=r1+r0
mov   r1,r2   ; r2=r1
...
```

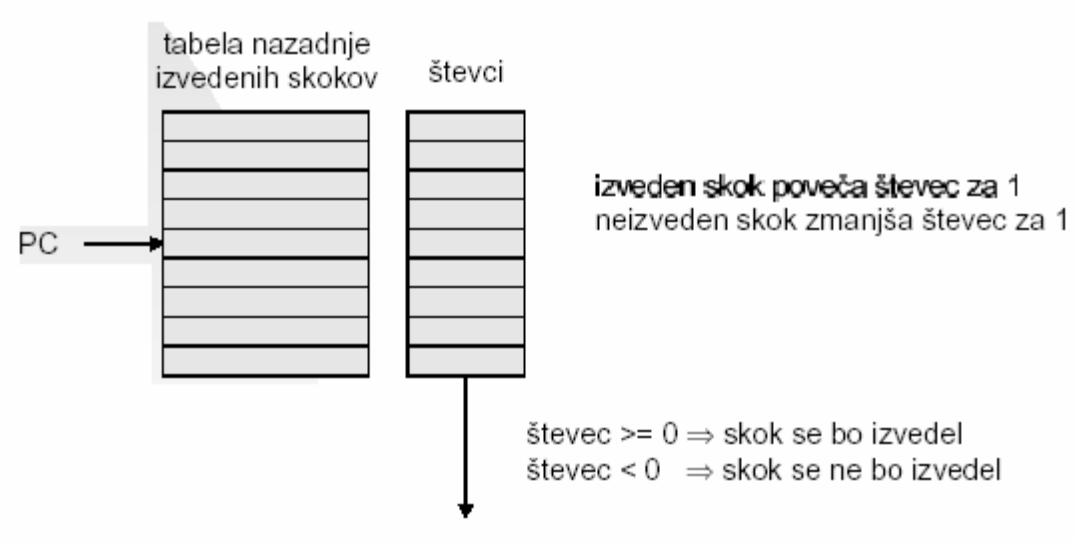


Napovedovanje skokov

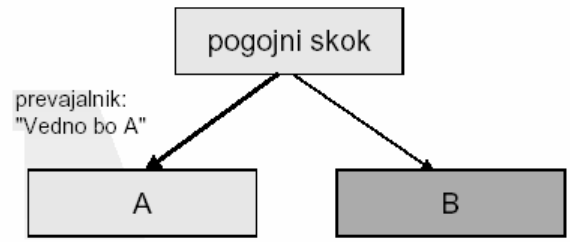
- ***Dinamično napovedovanje*** izvedbe skokov
  (Dynamic branch prediction)



- Posebna krmilna logika »ugane« smer skoka
- V primeru napake se izkoristek bistveno zmanjša: 5-10% napaka lahko 40-50% zmajša izkoristek.
- Zgled implementacije dinamičnega napovedovanja skokov

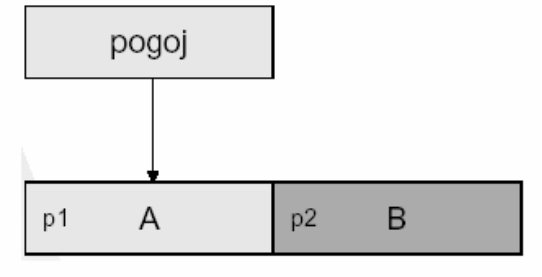


- ***Statično napovedovanje*** skokov (Static branch prediction)



- Prevajalnik določi katera pot bo skoraj vedno izbrana
- V primeru napačne napovedi se izkoristek bistveno ne zmanjša
- Ne potrebujemo (zahtevne) krmilne logike za ugibanje

- ***Uporaba predikatov*** (Predication)

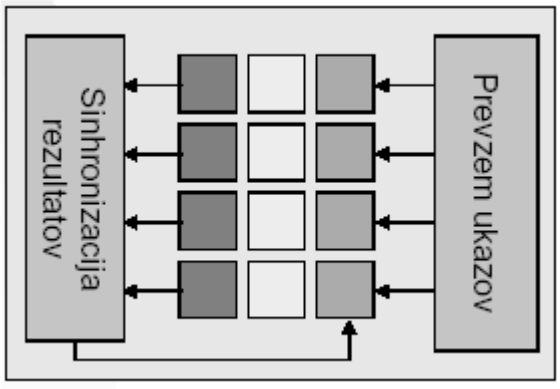


- Odstrani pogojni skok in izvede A in B vzporedno
- Ne more priti do napake v napovedovanju

- Predikata (bita) p1 in p2 določata rezultat ukaza, ki bo ostal oz. bo zavržen
- Koristna v primeru, ko je zelo težko predvideti smer skoka (npr. sortiranje)

Vzporedno izvajanje ukazov (ILP – instruction-level parallelism)

- Predstavlja razširitev filozofije cevenja
- Istočasno se prečita in obdeluje več ukazov hkrati (superskalarne arhitekture)
- Za izvedbo vsake faze imamo na razpolago več neodvisnih enot



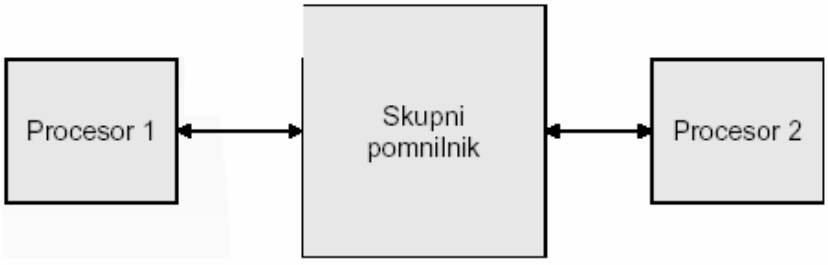
Koprocesorji in posebni ukazi

- Pomožne procesne enote za izvajanje posebnih nalog:
- računanje z realnim števili
- upravljanje z vhodno/izhodnimi napravami
- obdelava grafičnih podatkov
- Dodatni ukazi za pohitritev specifičnih operacij (obdelava slik, zvoka, …)

Primer: MMX in SIMD nabor ukazov pri Pentium III

Večprocesorski sistemi

- Porazdeljeno izvajanje nalog na več procesnih enotah hkrati.
- Običajno niso namenjeni za vzporedno izvajanje enega programa temveč za sočasno izvajanje večih programov (multitasking).



Trendi razvoja mikroprocesorjev

- Povečanje hitrosti preko 1Ghz
- Povečanje gostote na 0.18µ in več
- Izboljšave pri vzporednem izvajanju ukazov
- Večprocesorski sistemi na isti silicijevi rezini

Alternativna arhitektura:

- optični računalniki
- nevronski računalniki

Primerjava procesorjev:

Pentium III

- 16 Kb podatkovnega in 16 Kb ukaznega predpomnilnika (L1)
- 256/512 Kb sekudarnega predpomnilnika v istem ohišju
- hitrost delovanja do 1 Ghz
- CISC/RISC arhitektura: vsak CISC ukaz se prevede v enega ali več RISC ukazov
- V enem urinem ciklu se lahko izvede do 5 RISC ukazov
- Dinamično napovedovanje skokov in spekulativno izvajanje ukazov
- Nabor MMX ukazov in dodatnih 70 SIMD ukazov za multimedijske aplikacije

IA-64 (Merced-Itanium)

- Popolna 64 bitna arhitektura z možnostjo izvajanja IA-32 ukazov
- 128 registrov za cele in realna števila, 64 predikatnih registrov (bitov)
- Statično napovedovanje skokov in uporaba predikatov
- Spekulativno izvajanje ukazov

Alpha 21264

- Superskalarni procesor s pretočnim delovanjem
- Izvede do 4 ukaze v enem urinem ciklu
- Štiri računske enote za cela števila in dve računski enoti za realna števila (IEEE in VAX)
- 64 Kb podatkovnega in 64 Kb ukaznega predpomnilnika
- Večstopenjsko dinamična napovedovanja skokov
- Frekvenca ure nad 1 Ghz

9. Navidezni pomnilnik

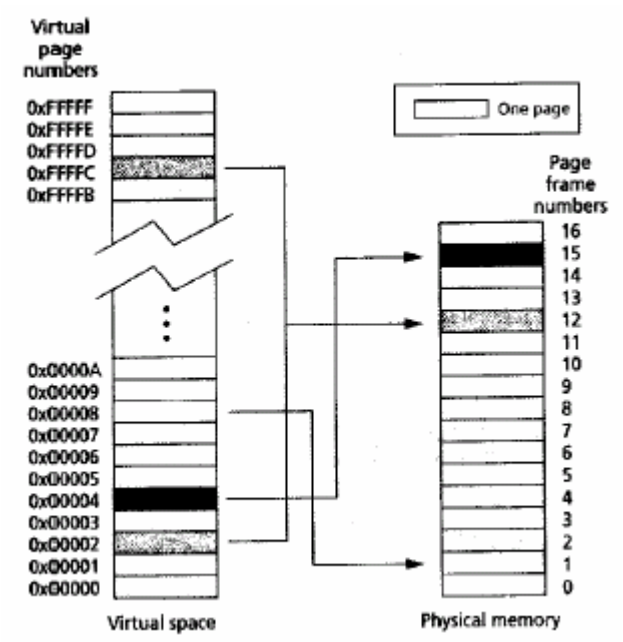
V kompleksnejših mikroračunalnikih je ugodno posebej poskrbeti za upravljanje s pomnilnikom (Memory Management). To porabniku omogoča preslikavo med njegovimi logičnimi naslovnimi področji in fizično shrambo v pomnilniku ter zaščito podatkov na teh področjih preko dodeljenih atributov.

Posebej pomembna funkcija upravljavca pomnilnika je vzdrževanje navideznega pomnilnika. Navidezni pomnilnik je bil razvit z namenom, da bi avtomatiziral premeščanje programov in podatkov med hitrim pomnilnikom in zunanjo shrambo, kadar prvega ni dovolj na razpolago. S tem je ustvarjen občutek velikega enovitega pomnilnika.

Princip delovanja je v tem, da se v fizičnem pomnilniku vzdržujejo tisti podatki, do katerih v nekem trenutku v resnici dostopamo. Tisti, ki jih takrat ne potrebujemo, so medtem spravljeni na disku. Ko pride do preklopa konteksta, se slednji naložijo v prosti fizični pomnilnik, če pa ga ni, spravimo tiste, kijih najdlje nismo uporabljali, na disk in jih preložimo z aktualnimi.

Strojna oprema in operacijski sistem omogočata, da se preslikave in prelaganje področij pomnilnika izvajajo sproti med tem, ko procesor posega na svoje logično naslovno področje. Preslikava se izvaja po straneh, najmanjših enotah, s katerimi lahko manipuliramo.

Virtualni naslovni prostor, ki ga naslavlja uporabnik, se deli na virtualne strani, ki imajo vsaka svoj naslov. Fizični pomnilnik je razdeljen na enako velike naslovljene okvirje. Najpreprosteje rečeno je torej virtualno naslavljanje preslikava virtualnih naslovov na fizične okvirje.



Informacije o preslikavah so zbrane v tabeli strani (page table); te so sestavljene iz posameznih vpisov tabele strani (page table entries, PTE), ki se nanašajo na posamezne strani. Vsak vpis (PTE) vsebuje vsaj podatke o virtualnem in fizičnem naslovu v pomnilniku ali na disku, lahko pa tudi

dodatne informacije, ki služijo za zaščito podatkov, na primer, ali je na stran mogoče vpisovati podatke, ali se na njej nahaja program, ipd. V njih je spravljena tudi informacija o tem, kdaj je bila stran nazadnje uporabljena. To informacijo upošteva sistem, ko se odloča, katere strani v fizičnem pomnilniku bo prekril z zahtevanimi, potem, ko je v njem zmanjkalo prostora.

Nekaterih podatkov nam ni treba eksplicitno vzdrževati v tabeli; z ustrezno organizacijo jih lahko implicitno ugotovimo iz položaja v tabeli.

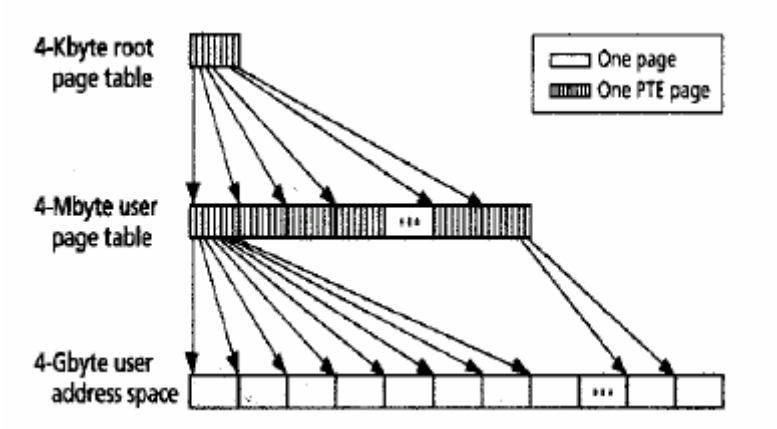
Za hitrejše delovanje preslikave večina sodobnih sistemov uporablja hitre pomnilnike (translation lookaside buffer, TLB), v katerih vzdržuje najbolj sveže podatke o zadnjih preslikavah. Če ob posegu v virtualno stran njen fizični naslov najdemo v TLB, ga takoj uporabimo. Kadar pa ga ni, ga moramo ugotoviti iz tabel.
V preteklosti, ko so bili pomnilniki manjši, pa tudi v preprostejših sodobnih sistemih, se je tabela strani nahajala v t.i. direktni tabeli (direct table). V direktni tabeli strani so zvezno navedeni podatki za vse virtualne strani. Dokler je ta razmeroma majhna, je upravljanje s pomnilnikom lahko izvedeno neposredno v strojni opremi.

Logični naslov je razdeljen na dva dela, na naslov virtualne strani in na odmik (notranji naslov) na strani. Naslov strani izbere vpis v tabeli strani (PTE), ki vsebuje atribute te strani za nadzor dostopa in naslov fizične strani. Iz slednjega in iz odmika se sestavi fizični naslov podatka v pomnilniku, iz atributov pa se razbere, ali je ta stran na razpolago, ali jo je treba naložiti, ter privilegije, potrebne za njen dostop.

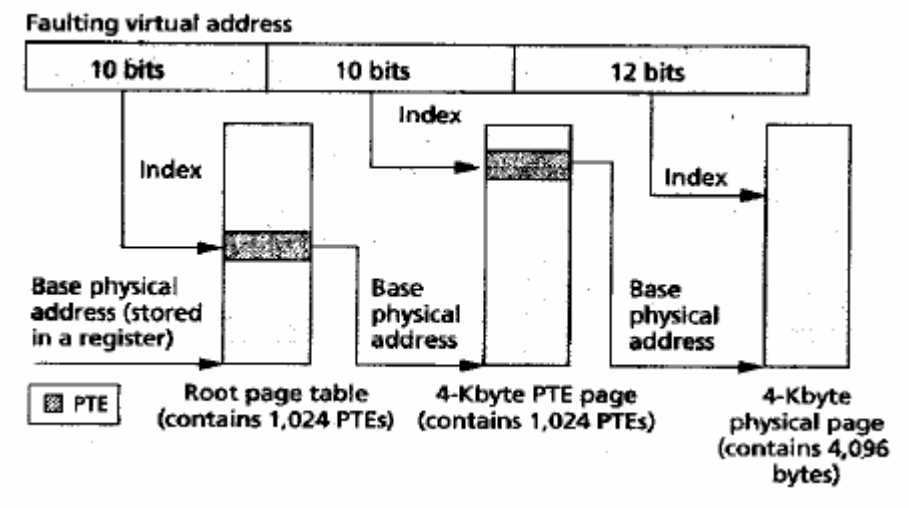
S povečevanjem pomnilniškega prostora je bilo potrebno to tabelo preseliti v zunanji pomnilnik, njeno obdelavo pa izvesti s programiranjem; upravljanje s pomnilnikom postane funkcija operacijskega sistema. To pa pomeni, daje izvedba te funkcije bistveno počasnejša.
Tabele strani sedaj ne morejo več vsebovati podatke o vseh virtualnih straneh, ker jih je preveč; to pa pomeni, da naslov strani ne določa več pozicije vnosa v tabeli strani (PTE). Podatke o preslikavi je zato potrebno poiskati v teh tabelah.

Preiskovanje velikih tabel je posebej časovno zahtevna funkcija, zato so bili razviti posebni algoritmi. Običajni pristopi upoštevajo hierarhično zgradbo oz. segmentacijo pomnilnika: ta je razdeljen na segmente, ti pa dalje na strani. Preslikava sedaj poteka v dveh korakih: najprej se poišče naslov fizičnega segmenta, nato pa naslov strani znotraj njega. Ker je segmentov bistveno manj kot strani, znotraj segmentov pa spet bistveno manj strani, kot jih je vseh v pomnilniku, je iskanje preslikav precej hitrejše.
Slika prikazuje naslednji primer: pri 32-bitnem naslavljanju, ki omogoča 4 Gbyte (2^{32}) prostora imamo pomnilnik organiziran po 4kbytnih (2^{12}) straneh. Takšnih strani je torej 1M ($2^{32}/2^{12} = 2^{20}$). Za preiskovanje takšne tabele potrebujemo veliko časa. Zato virtualni naslov strani sestavimo iz dveh delov po 10 bitov: prvi določa vnos v korenski tabeli strani dolžine 4-Kbyte, ki podaja začetek uporabniške tabele strani, drugi del naslova pa iz slednje tabele poišče fizični naslov strani. Na tej strani velikosti 4kbyte s pomočjo 12-bitnega notranjega naslova določimo končni fizični naslov.
Korenska tabela strani je pri tem kratka in zato brez večje izgube vedno prisotna v pomnilniku. Od ostalih tabel pa so prisotne le tiste, ki so v resnici uporabljene. Običajno tudi pripadajo enemu opravilu; fizični okvirji, na katere preslikujejo virtualne naslove, morajo običajno biti istočasno prisotni v pomnilniku, kar tudi poenostavi mehanizem prenosa fizičnih strani med diskom in pomnilnikom: podatek o razpoložljivosti podatkov v okvirju fizičnega pomnilnika je lahko vpisan

že v korenski tabeli preslikav, med diskom m pomnilnikom pa lahko naenkrat prenesemo večjo količino pomnilnika.



Na spodnji sliki je za ta primer prikazan postopek sprehoda po tabeli (tablewalking); uporabljen je način preiskovanja od zgoraj navzdol (Top-down traversal, Forward-mapped page table):



poleg tega obstajata še sodobnejša načina preiskovanja od spodaj navzgor in inverzne preslikave.

10. Operacijski sistemi

KAJ JE TO OPERACIJSKI SISTEM ?

CP/M, VMS, OS/9, Unix, Lynux, Solaris, AIX, Apple System 7, Multics, Windows NT, OS/2, … so imena, ki so (vsaj nekatera) vsakemu, ki se vsaj malo ukvarja z računalništvom, dobro znana. Vsa označujejo različne operacijske sisteme. In kaj sploh je to - operacijski sistem? Na kratko rečeno gre za zadevo, ki nam omogoča, da z računalnikom počnemo koristne stvari. Ko računalnik pride iz tovarne, je le za kup komponent (elektronskih in tudi drugih), s katerimi pa si ne moremo kaj dosti pomagati. Prav vseeno bi bilo, če bi bil sestavljen le iz ohišja.

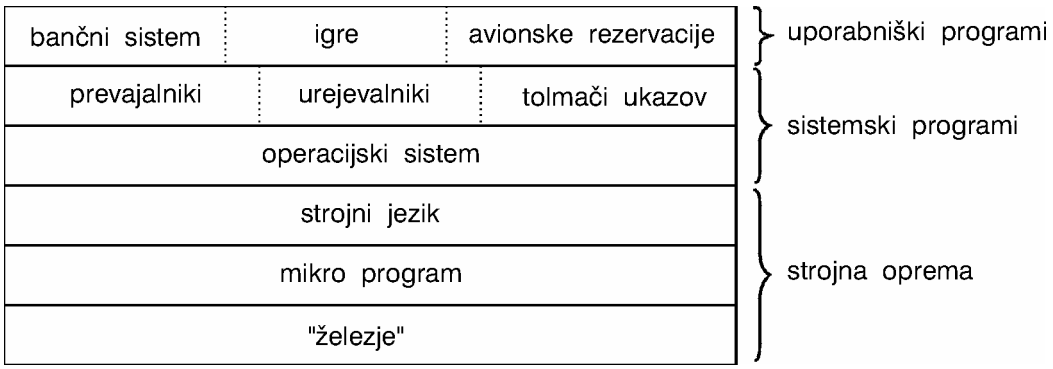
Definicija

- Operacijski sistem je sklop ročnih in avtomatskih postopkov, ki uporabniku omogočajo uspešno uporabo računalnika (Per Brinch Hansen: *Operating system principles*).
- Operacijski sistem je osnovni sistemski program, ki nadzira vsa računalnikova sredstva in predstavlja osnovo, nad katero nastajajo uporabniški programi. (A. Tanenbaum: *Operating systems: Design and Implementation*)
- Operacijski sistem je skupek programov, ki leži med strojno opremo in uporabniškimi programi. (H.L. Capron: *Computers: Tools for an Information Age*)

Definicij bi lahko našteli še več, vendar je pravzaprav težko točno opredeliti, kaj operacijski sistem je. Recimo kar po domače:

- Operacijski sistem je tisti nepogrešljivi del programske opreme, ki skrbi za to, da z računalnikom sploh kaj lahko počnemo.

Kaj OS počne?
Ko ugotavljamo, kakšne so naloge, ki jih opravlja operacijski sistem je važno, s katerega zornega kota gledamo - v tem leži tudi večplastnost definicije operacijskega sistema.



Ptičji pogled (navidezni računalnik)

Ena od opredelitev OS je, da ugotovimo, kakšne usluge nam nudi OS. Večine se jih pravzaprav sploh ne zavedamo, ampak jih jemljemo kot same po sebi umevne. Denimo, da želimo prepisati datoteko. Operacijski sistem nam omogoča, da to izvedemo z enostavnim ukazom (COPY, poteg miške, PIP, besedni ukaz, …). Če OS ne bi bilo, bi ta zadeva zgledala v najboljšem primeru nekako takole: shrani prvih 128 zlogov na cilinder 7, glava 2, sektor 3. Popravi tabelo zasedenosti sektorjev na disku. Poišči naslednje prazno mesto. Shrani naslednjih 128 zlogov na … ali pa še bolj zapleteno. Tako zna denimo tipičen čip, ki nadzira delovanje disketne enote, izvajati 10 do 20 ukazov (branje, pisanje podatkov, premikanje roke, kalibracija, …) Vse te ukaze bi za preprosto prepisovanje morali poznati. Če bi računalniku dodali nov tip zunanjega pomnilnika, ali pa zamenjali kontrolni čip, bi morali poznati nove ukaze. Jasno je, da običajen uporabnik ne želi vedeti nič o programiranju diskovnih krmilnikov. Namesto tega potrebuje enostavno, abstraktno sliko diskete, kot recimo skupka poimenovanih datotek. Ta slika mora biti enaka za vse različne oblike zunanjega pomnilnika - čeprav je dejanski način zapisovanja lahko (in je) med različnimi enotami zelo različen. In naloga OS je, da uporabniku tako abstraktno sliko nudi. To sliko uporabljata tako "končni" uporabnik pri uporabi programov, kot tudi programer pri pisanju programske opreme.
Takih nalog (servisov) je kar nekaj - delo s pomnilnikom, nadzor tipkovnice in ostalih zunanjih enot in še cel kup podobnih zadev, ki se neposredno tičejo strojne opreme in njenega delovanja.
Operacijski sistem torej "skriva resnico" o strojni opremi in uporabniku te opreme kaže navidezni pogled na računalnik. Tak navidezni računalnik je enostavneje uporabljati. Omenimo še to, da ima pri tem "skrivanju" za uporabnika veliko vlogo tudi **uporabniški vmesnik**. Tega sicer pogosto ne štejemo za del operacijskega sistema, vendar je z njim tesno povezan.

Žabji pogled (dodeljevanje zmogljivosti)

Sodobni računalniški sistem sestavljajo procesorji, pomnilnik, diski, različne izhodne in vhodne enote, omrežni vmesniki, tiskalniki ter druge enote. Naloga operacijskega sistema je, da različnim programom, ki se izvajajo v računalniku, omogoči urejen in nadzorovan dostop do teh sestavnih delov. Tako večina sodobnih operacijskih sistemov omogoča sočasno izvajanje različnih programov. OS mora omogočiti, da pri takem delovanju ne pride do zapletov. Vsak program mora teči, kot da je računalnik samo njegov. Denimo, da se v računalniku sočasno izvajajo trije programi, ki vsi pišejo na tiskalnik. OS mora zagotoviti, da se na tiskalniku ne bo pojavilo nekaj vrstic izpisa prvega programa, nekaj vrstic drugega in nekaj vrstic izpisa tretjega programa.
Pomen ustreznega dodeljevanja sredstev je še bolj izrazit pri operacijskih sistemih, ki omogočajo, da računalniški sistem sočasno uporablja več uporabnikov.
Naloga OS torej je, da **urejuje zahteve po sredstvih**, omogoči in vodi nadzor nad njihovo uporabo.

Glavne funkcije OS

- upravljanje s sredstvi računalnika kot so procesor, pomnilnik, zunanje pomnilniške enote, tiskaniki, ...
- postavitev uporabniškega vmesnika
- izvajanje in podpora storitev za uporabniško programsko opremo

S stališča uporabnika je eno glavnih opravil današnjega operacijskega sistema delo z datotekami. Te operacije so "navzven" najbolj vidne. Zato ne preseneča, da je pogost laični odgovor na vprašanje, kaj dela operacijski sistem ravno ta, da skrbi za datoteke. Operacijski sistem omogoča, da uporabnik vidi datoteke v skladu z možnostmi, ki jih ponuja datotečni sistem. Posamezni operacijski sistemi omogočajo delo z različnimi datotečnimi sistemi (npr. Windows NT omogoča uporabo sistema FAT in NTFS).

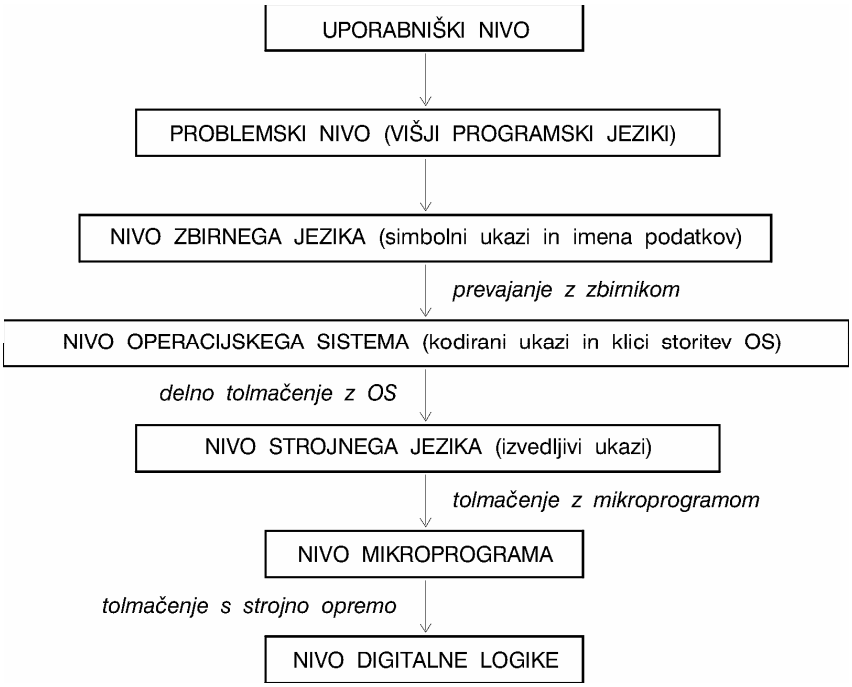
Uporabniški vmesnik

Uporabniški vmesnik je program, ki uporabniku omogoča komunikacijo z računalnikom - poganjanje programov, nadzor zunanjih enot, … Posamezni avtorji ga štejejo za del operacijskega sistema, drugi pa ga štejejo med pomožne sistemske programe (kot so npr. protivirusni programi, programi za arhiviranje, …) Z razvojem strojne opreme so se razvijali tudi uporabniški vmesniki. Načeloma lahko nad istim operacijskim sistemom uporabljamo različne uporabniške vmesnike.
Prvotni uporabniški vmesniki so omogočali ukazovanje le z vnosom besednih ukazov (CP/M, VMS, DOS, ...). Nato so se pojavili znakovni izbirni sistemi (DOS 4, PC SHELL, Norton Commander, …), za njimi pa še grafični uporabniški vmesniki. Pojavili so se tudi zvočni uporabniški vmesniki, kjer uporabnik z računalnik upravlja z govorjenimi ukazi (npr. zadnja različica OS/2).

Glavna težava pri uporabi različnih uporabniških vmesnikov je ravnotežje med zmogljivostjo in "prijaznostjo". Če pogledamo to na konkretnem primeru para DOS/Windows: veliko stvari v DOSu opravite hitreje (zbriši eno datoteko, formatiraj disketo, …) vendar ste v težavah že, če se zmotite pri eni črki v ukazu. Ukaze si morate tudi zapomniti. Pri okolju Windows je poudarek na tem, da si slikovno predstavljeno informacijo (npr. ikone), laže zapomnimo in povežemo z določenim opravilom, kot točno zaporedje črk nekega ukaza (ki običajno izvira iz nam tujega jezikovnega okolja). Žal pa s tem hitreje uporabimo tudi takšne ukaze, ki nam ob napačni rabi lahko povzročijo probleme (denimo prestavimo program iz ene skupine v drugo). Ravno tako večina opravil zahteva, da vsaj potrdite, da se strinjate z vsemi nastavitvami, kar je včasih zamudno. Dodatni problem uporabe okolja Windows v šoli je še v tem, da je to enouporabniški sistem, ki si zapomni vse nastavitve. Te so zaradi "prijaznosti" dosegljive vsem in jih je relativno težko ustrezno zaščititi. To pomeni, da si jih vsakdo poskuša prilagoditi po svoje, kar lahko povzroča hude težave naslednjemu uporabniku.

Navidezni računalnik

Vsak računalnik lahko razdelimo na več nivojev oziroma plasti. Vsakemu nivoju ustreza navidezni računalnik, ki razume le svoj jezik.



DELITEV OPERACIJSKIH SISTEMOV

Število uporabnikov

Enouporabniški sistemi

Velika večina operacijskih sistemov, ki tečejo na osebnih računalnikih, je enouporabniških. To pomeni, da računalnik hkrati uporablja le en uporabnik. Primeri takih operacijskih sistemov so MS-DOS, OS/2, Windows 95, Windows NT in drugi.

Večuporabniški sistemi

Značilnost teh sistemov je, da na njih hkrati dela večje število uporabnikov, ki se preko terminalov povezujejo na osrednji računalnik. Vsi sodobni večuporabniški operacijski sistemi delujejo po načelu časovnega deljenja sredstev, tako da (če uporabnikov le ni preveč, oz. če njihve zahteve po sredstvih niso prehude) ima vsak posameznik občutek, da ima na voljo računalnik le zase. Primer VMS,OS9,UNIX itd.

En program

V to skupino štejemo tiste operacijske sisteme, ki omogočajo, da je v pomnilniku računalnika hkrati le en program, ki se izvaja. Tipičen primerek takega operacijskega sistema je MSDOS (odmislimo pritajene programe). Opozoriti velja, da tudi ti operacijski sistemi dopuščajo izvajanje več nalog hkrati (npr. procesiranje in branje z diska), še zlasti, ker običajno ne preprečujejo, da se določene vhodno/izhodne operacije opravijo mimo operacijskega sistema.

Večopravilni - multitzaking

Vsak moderen operacijski sistem, ki da nekaj nase, mora podpirati večopravilnost - torej možnost uporabnika, da sočasno izvaja več programov. Seveda lahko hkrati procesor (če je ta en sam) uporablja le en uporabniški program, vendar pod sočasnost razumemo to, da v istem časovnem okviru izvajamo več programov, ki si med sabo delijo sredstva. Gre za nadgradnjo ideje multiprogramiranja. Glede na način, kako si delijo glavno sredstvo - procesor - pa v grobem poznamo dve skupini. Pri prvi se operacijski sistem zanaša na "vljudnost" programov, da občasno odstopijo procesor tudi drugim. Taka je večopravilnost, ki jo imajo običajna Okna (Windows). S tujko temu rečemo cooperative multitasking. Vsak malce bolj izkušeni uporabnik ve, da v praksi to pomeni le to, da imamo v pomnilniku hkrati več programov, dejansko pa se sočasno izvaja en sam. Windows NT pa že pozna večopravilnost s časovno delitvijo, ko operacijski sistem nadzira procesorski čas, ki ga posamezno opravilo porabi (pre-emptive multitasking).

Povezanost

"Običajni" OS

Tu mislimo na tiste OS, ki ne podpirajo dela v omrežju. Vsak računalnik je "osamljen otok" in ima na voljo le tista sredstva, ki so sestavni del tega računalnika. Taki operacijski sistemi v splošno namenskih računalnikih počasi izginjajo, oziroma jih dopolnjujejo dodatki, ki omogočijo delo v omrežju.

Omrežni OS

Kakor hitro povežemo dva ali več računalnikov med sabo, dobimo omrežje. Običajni operacijski sistem ne zna poskrbeti za delitev sredstev med posameznimi računalniki (npr. da bi en računalnik uporabljal disk drugega). Zato potrebujemo omrežne operacijske sisteme. Ti omogočajo delitev sredstev preko omrežja in običajno tudi to, da običajni OS delujejo na znan način brez spremembe. Poleg tega omrežni operacijski sistemi omogočajo varnost, upravljanje z omrežji in opravljanje še drugih nalog.
Omrežni operacijski sistem pravzaprav sestavljata dva operacijska sistema - prvi teče na strežniku, nadzira njegovo delovanje strežnika in upravlja z tam shranjenimi datotekami, drugi pa je operacijski sistem, ki teče na odjemalcu, ki dostopa do omrežja in tamkajšnjih sredstev. Opozoriti velja, da je, glede na tip omrežja, isti računalnik lahko hkrati strežnik, kot tudi odjemalec. Kvaliteten in preizkušen mrežni OS je UNIX kot tudi različice za PC LINUX. Tudi Windows bolj ali manj uspešno obvalduje mrežne zahteve.

Operacijske sisteme bi lahko delili tudi drugače. Nismo omenili delitev na porazdeljene operacijske sisteme, OS, ki omogočajo večnitnost, večprocesorske operacijske sisteme, na napake odporne operacijske sisteme, operacijske sisteme za delo v realnem času, ...

Zgodovinski razvoj operacijskih sistemov za PC:

Avgust 1981	DOS 1.0.	prvi OS za IBM PC, združljiv z CP/M, le en imenik
oktober 1982	DOS 1.1	podpora disketam vel. 320K
marec 1983	DOS 2.0	praktično nov program, združljiv s starim, podpora disku, disketam 360K, CONFIG.SYS, osnovan na UNIXu
Marec 1984	DOS 2.11	podpora različnim jezikom
Avgust 1984	DOS 3.0	podpora procesorju 80286,
November 1984	DOS 3.1	podpora omrežjem (se ni uveljavilo)
1985	Windows 1.0.	strojna oprema premalo zmogljiva - pravzaprav ni bilo uporabnikov - potrebe
Januar 1986	DOS 3.2	podpora 3.5” disketnim enotam (720K)
April 1987	DOS 3.3	1.44M disketne enota
1987	Windows 2.0.	nima še grafičnega uporabniškega vmesnika, omogoča zložiti več programov v pomnilnik (nekakšna večopravilnost) - preklapljanje med posli
1987	Windows 286	za procesor Intel 80286, zaradi boljšega procesorja omogoča boljše preklapljanje med posli
Julij 1988	DOS 4.0	podpora diskom, večjim od 32M, DOS lupina
1988	Windows 386	Intel 386, večopravilnost (cooperative multitasking)
Maj 1990	Windows 3.0	grafika, ikone, večopravilnost, prva komercialno uspešna različica
April 1991	DOS 5.0	izboljšano delo s pomnilnikom, na novo napisana večina kode
April 1992	Windows 3.1	ne teče več na 8088, izboljšano delo s pomnilnikom, True Type nabori pisav
November 1992	Windows for Workgroups 3.1	delo z omrežjem, izboljšave - uporabljajo tudi tisti, ki niso na omrežju
1993	Windows for Workgroups 3.11	odpravljene številne napake različice 3.1
1993	DOS 6	stiskanje podatkov
1993	DOS 6.2	odpravljene napake različice 6, spor zaradi uporabljene tehnologije
1993	Windows 3.11	
1993	Windows NT 3.5	
1995	Windows NT 3.51	
Avgust 1995	Windows 95	
Avgust 1996	Windows NT 4 (Server, Workstation)	
1997 - 1998	Linux ,Windows 98	
1999	Linux +, Windows2000 beta	

Značilnosti posameznih sistemov

DOS

Namenjen je izključno enemu uporabniku. Ta lahko počne vse - ni zaščite posameznih datotek. Uporabnik ima precejšen nadzor nad sistemom - lahko npr. sam napiše program, kako ukrepati od procesorskih prekinitvah (DOSKEY). Deli OS so zamenljivi - npr. uporabniška lupina z vmesnikom vred (`command.com`) (Pozor! Dejali smo že, da ta del pravzaprav ne spada v osrednje jedro OS). Programom omogoča, da se obračajo naravnost na strojno opremo. To prinaša s seboj prednosti - povečano hitrost in slabosti - zabrisan je koncept navideznega računalnika, pravilnost programov je odvisna od strojne opreme, pri menjavi strojne opreme je delovanje programa vprašljivo.

DOS ne omogoča multiprogramiranja, čeprav tudi ni čisti enoprogramski operacijski sistem. Omogoča sicer, da glavni proces lahko naredi podrejene procese, vendar se oče .... - povejmo raje po domače: program lahko požene drug program, a s svojim delom nadaljuje šele, ko slednji svoje delo konča. Zaradi omejitev, ki jih je postavljal prvi procesor, na katerem je DOS tekel (in kasnejše zahteve po popolni združljivosti z obstoječimi programi), je zelo zapleteno delo s pomnilnikom - kako uporabiti pomnilnik nad 640K.

Obstaja nekaj različic tega operacijskega sistema - MS DOS, PC DOS, DR DOS, vendar gre v glavnem za “kozmetične” razlike, ki ne vplivajo ne na uporabnikov ne na programerjev pristop.

WINDOWS

Windows kot taki pravzaprav sploh niso operacijski sistem - so le različica uporabniškega vmesnika. Po drugi strani pa od DOSa prevzamejo cel kup nalog in programerju omogočajo drugačen pogled na računalnik - nov navidezni računalnik. Tako so operacijski sistem, kar pa spet niso, saj niso samozadostni - za delovanje še vedno potrebujejo DOS. Skratka - Windows so grafični uporabniški vmesnik za računalnike z operacijskim sistemom DOS, ki pa so od DOSa prevzeli določene naloge operacijskega sistema. So razširitev operacijskega sistema DOS.

Omogočajo t.i. cooperative multitasking - dogovorno večopravilnost. To pomeni, da v računalniku lahko sočasno izvajamo več programov. Vendar ti sami povedo, kdaj ne potrebujejo npr. procesorja - da ga bo lahko uporabil drug program. V praksi to pomeni, da je prave večopravilnosti ni.

WINDOWS for Workgroups

Gre pravzaprav za program Windows, ki so mu dodali podporo dela v omrežju. Podpira deljenje sredstev (tiskalnik, disk, ...) in različne omrežne protokole. V osnovni različici nima podpore za protokol TCP/IP, obstajajo pa brezplačni dodatki, ki omogočijo, da tak računalnik s tem OS povežemo v Internet (za kar potrebujemo TCP/IP).

WINDOWS 95

Je že “pravi” OS. V svoji zasnovi je to 32bitni operacijski sistem, vendar ima zaradi zahtev po popolni združljivosti z DOSom in Windowsi, velike dele še 16 bitne. Še vedno je to sistem za enega uporabnika s primesmi možnosti dela več ljudi (prijavljanje, osebne nastavitve, ni pa zaščite datotek, ...)

WINDOWS NT

Pri tem operacijskem sistemu so se pri Microsoftu odločili, da bodo malce spremenili strategijo. Tako niso več vztrajali po popolni združljivosti s prejšnjimi operacijskimi sistemi. To jim je omogočilo, da so operacijski sistem pisali zelo na novo, brez prehudih omejitev, ki jih postavlja prejšnje stanje. Nastal je sodoben operacijski sistem, ki ima večino značilnosti sodobnih, "spodobnih" operacijskih sistemov. Je prenosni - teče na računalnikih s procesorji serij Alpha (Digital Equipment Corporation), PowerPC (IBM, Motrola, Apple), x86 in Pentium (Intel), Mips (Mips). Je povsem 32 bitni omrežni operacijski sistem. Omogoča večopravilnost, ki jo imenujemo preemptive mutitasking. Tu operacijski sistem prevzame vlogo, kot jo imajo semaforji na cesti - odloča, kdaj ima posamezni proces (program) dostop do kakšnega sredstva in to počne izmenjaje - nekaj časa en proces, nekaj časa drugi, pa spet tretji, … Obvezno zahteva prijavljanje uporabnika, čigar datoteke so zaščitene pred drugimi uporabniki.
Podpira tudi računalnike z več procesorji (in jih zna izrabiti), Je bistveno stabilnejši kot Windows 95 (16 bitni programi (vsi DOSovski programi) tečejo vsak v svoljem pomnilniškem prostoru - ne morejo zrušiti ostalih!) In kaj so njegove slabosti? Predvsem zahteva računalnik z več pomnilnika (16M je minimum), določeni programi, napisani za operacijski sistem DOS in Windows (kot tudi Windows 95), v tem operacijskem sistemu ne tečejo - predvsem igrice (direktni dostop do strojne opreme). Prav tako zahteva več računalniškega znanja pri nameščanju na nov računalnik in ob dodajanju nove ali zamenjavi obstoječe strojne opreme.

LINUX, FREEBSD, ...

Poleg Microsoftovih operacijskih sistemov, se na šolah kot operacijski sistem pogosteje pojavlja le še Unix, običajno v eni od prostih različic (Linux, FreeBSD). Gre za zmogljive operacijske sisteme, saj gre za polne izvedbe Unixa (in ne denimo okrnjenje), pa še brezplačni so.
Značilnosti:

- omrežni operacijski sistem
- večuporabniški, večopravilni
- potrebujejo sistemskega administratorja
- zaščita datotek pred drugimi uporabniki
- podpora protokolu TCP/IP
- elektronska pošta, FTP, DNS, TELNET, ...
- možnost različnih uporabniških vmesnikov
-

Kaj pa komercialne različice Unixa (Solaris, AIX, IRIX, …)? Vsekakor gre za odlične operacijske sisteme. Vendar so običajno vezani na dokaj zmogljivo (in drago) strojno opremo. Vprašljiva je tudi "učna podpora" - priročniki, programi za rabo v šoli, revije, …

11. Datotečni sistemi

Microsoft-ovi DATOTEČNI SISTEMI

Del operacijskega sistema OS so seveda datotečni sistemi. So odvisni in seveda na vsakem OS različni. Poglavitna naloga datotečnega sistema je skrb za delo z datotekami. Zato vsak datotečni sistem shrani datoteko v svojem formatu. Spodaj bomo opisali posamezne datotečne formate. Najprej pa si poglejmo lastnosti datotek:

Vsaka datoteka ima svoje ***ime*** in zaseda določen ***prostor na disku*** (različno od OS). Datoteka je lahko različnega tipa - torej lahko vsebuje besedilo (TXT datoteka, DOC kot dokument datoteka kreirana v urejavalniku besedila Word), lahko so izvršljive (EXE, COM, BAT …) in podobno. Nekateri OS predvsem DOS in Windows NT prepoznavajo datoteke po njeni končnici. (**ime.končnica, npr. "naloga.doc"**) drugi pa po opisu v njegovi glavi datoteke.

OPIS DATOTEČNIH SISTEMOV (DOSov in seveda tudi Windows Windows 95, 98 - FAT, FAT 32, Windows NT – NTFS)

FAT in FAT 32

Microsoft-ov »novi« produkt Porazdelitvena tabela datotečnega sistema (File Allocation Table File System) je strukturirana tako, da bolje izkoristi prostor na disku s tem, ko zmanjša velikost skupkov. Če ste kupil disk večji od 2GB, ste dobili zraven še zaganjalnik OEM Service Release 2 (SR2) različico za Win95, katera vključuje sistem FAT32. Novejša različica Win98 pa popolnoma podpira datotečni sistem FAT32.

PREDNOSTI: Velikost samostojne particije ima lahko do 2 TB (navadni »FAT« oz. FAT16 pa le do 2 GB), kot pa smo že prej povedali uporablja manjše skupke in zato boljše izkorišča prostor na disku.

SLABOSTI: Za uporabo FAT32 morate nadgraditi vsa diskovna orodja in antivirusne programe. Pri nekaterih starejših programih, tudi MS Office95 in Office 4.3, je možno, da ne bodo delovali pravilno pod SR2. FAT32 particije ne bodo vidne v starejših verzijah DOS-a, v Win95 »Classic« ali v Win NT.

FAT32 je razširjen FAT datotečni sistem, ki podpira večje trde diske, z izboljšano prostorsko izkoriščenostjo

Datotečni sistem	Uporablja se v:	Značilnosti/uporaba
FAT ali FAT16	DOS 1-7.0, Windows95 (vse vezije) WindowsNT4	To je najbolj razširjen primer ki se uporablja. Na žalost FAT16 ne podpira diskov nad 2GB in prostorska izkoriščenost je zelo slaba na velikih diskih.
FAT32	Windows 95 OSR/2, DOS 7.1	FAT32 je najboljši Windows95 datotečni sistem. Podprti so diski nad 20GB. Diski formatirani s sistemom FAT32 ne morejo biti brani s statimi verzijami Windowsev.
HPFS	OS/2 Warp	HPFS (High Performance File System) je tudi dober sistem in je podprt samo v OS/2 Warp.
NTFS	WindowsNT 3.51, Windows NT 4.x	NTFS (NT File System) je zelo dobro oblikovan datotečni sistem sistema Windows NT Sprotna kompresija (stiskanje) datotek je ena glavnih značilnosti tega sistema poleg drugih.

Tabela datotečnih sistemov

DOSTOPI

Imamo naslednje pravice dostopa do datotek:

- No Access (ni dostopa)
- Read (bralni)
- Change (spreminjevalni)
- Full Control (popolen dostop)
- Special Access (poseben dostop)

Permission	R	X	W	D	P	O
No Access						
Read	Yes	Yes				
Change	Yes	Yes	Yes			
Full Control	Yes	Yes	Yes	Yes	Yes	Yes
Special Access (any combination)	Yes	Yes	Yes	Yes	Yes	Yes

- **R** - izpišemo podatke datoteke, njegove atribute, lastnika in pravice dostopa
- **X** - lahko izvršujemo datoteko
- **W** - pišemo v datoteko in lahko spreminjamo njena atribute
- **D** - lahko zbrišemo datoteko
- **P** - lahko spremenimo pravice dostopa
- **O** - lahko prevzamemo lastništvo nad datoteko

Razlaga: če ima datoteka pravico »Change«, potem lahko z njo delamo 'operacije' R,X,W, R pomeni izpis lastnosti datoteke, W pomeni da lahko vanjo pišemo …

Rešitve v UNIX / LINUX operacijskem sistemu

Dober opis v slovenščini je na razpolago na http://os.tscng.net.
Za boljše razumevanje opisanega je priporočljivo je primerjati rešitve v Windows in v UNIX.