UNIVERZITET MB

POSLOVNI I PRAVNI FAKULTET

MEMORIJSKI SISTEM I MAGISTRALA

- SEMINARSKI RAD -

Profesor:
Prof. dr Dušan Regodić

Kandidat:
Nina Stanković
I 20/2019

Beograd, 2020.

POSLOVNI I PRAVNI FAKULTET

UNIVERZITET MB
POSLOVNI I PRAVNI FAKULTET

Beograd, Knez Mihailova 33

Kandidat: Nina Stanković

Broj indeksa: I 20/2019

Studijski program: Informacione tehnologije

Tema: Memorijski sistem i Magistrala

Nastavnik

Datum odobrenja rada:
Beograd, ___.___._____.

_____

Memorijski sistem i magistrala

Sažetak: U ovom radu bavićmo se memorijskim sistemima i magistralama. U njemu ćemo objasniti kakvi sve tipovi i vrste memorija postoje, koja je primena memorija u radu računara i drugih uređaja, objasniti kakva je uloga magistrala i koje sve magistrale postoje i za šta se koja koristi. Takođe ćemo se baviti i tehnologijama (FIFO, LIFO, Slučajni upis, LRU) upisa podataka u memoriju i ispisa podataka, kao i logičkim i stvarnim adresama na kojima se nalaze podaci. Na kraju rada ćemo objasniti koliko su zapravo brze memorije i kakav je odnos kapaciteta i cene memorije.

Ključne reči: memorijski sistem, magistrala, keš memorija, operativna memorija, virtuelna memorija, adresni prostor, arbitracija, vrste magistrala…

SADRŽAJ

1. UVOD

Programi i podaci nad kojima se radi, čuvaju se u memoriji računara. U idealnom slučaju, memorija bi bila brza, velika i jeftina. Međutim često nije moguće ispuniti sve navedene zahteve istovremeno. U proteklim decenijama, mnogo rada je uloženo u razvoj pametnih struktura koje poboljšati brzinu i kapacitet memorije, a da cena memorije bude svedena na minimum.

Memorija se obično dizajnira tako da se podaci dobijaju i čuvaju u bitovnim sekvencama čija je dužina u stvari dužina reči. Obično se dužina reči kod računara definiše kao broj bitova kojima se pristupa ili koji bivaju sačuvani tokom jednog pristupa memoriji.

Postoje različite vrste memorije, od kojih su najpoznatije Keš memorija, Operativna memorija i Virtuelna memorija, pri čemu svaka od njih ima svoju funkciju u radu računara.

Memorije čine jedan od tri osnovna bloka mikro-računarskog sistema. Veza memorije sa centralnom procesorskom jedinicom je ostvarena putem adresne i magistrale podataka.

Cilj ovog rada jeste detaljnije upoznavanje sa memorijskim sistemom računara. Nakon analize hijerarhijske organizacije memorijskog sistema, detaljnije je opisan princip rada Keš memorije, Operativne memorije i Virtuelne memorije. Kod Keš memorije je opisano nekoliko bitnih tehnika, i to: tehnika preslikavanja blokova, tehnika zamene blokova i tehnika ažuriranja operativne memorije.

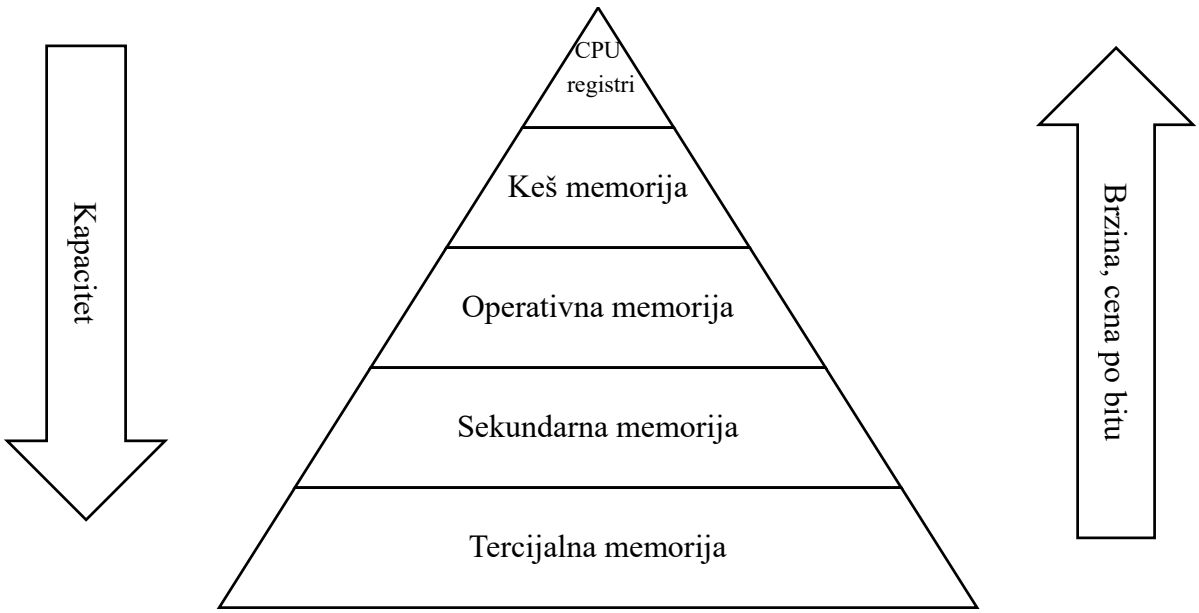
Kada je reč o prenosu podataka unutar računara, u radu je opisana struktura magistrale putem tri vrste linija za prenos, i to: adresnih linija, linija podataka i kontrolnih, odnosno upravljačkih linija. Posebna pažnja je posvećena postupcima arbitracije na magistrali, tj. odlučivanja o tome kome će magistrala biti dodeljena u datom trenutku. Predstavljene su dve vrste magistrala: asinhrona i sinhrona.

Rad je zasnovan na analizi domaće literature, a za bliže pojašnjenje predmetne oblasti, korišćen je tabelarni i slikovit pristup.

2. POJAM I VRSTE MEMORIJE

Memorije služe za smeštanje i čuvanje podataka i programa. U računaru postoji više vrsta memorija sa različitim ulogama. One čine memorijski sistem računara. Memorije su organizovane u hijerarhiju na čijem vrhu se nalaze brze i skupe memorijske jedinice malog kapaciteta, dok su na dnu sporije i jeftinije jedinice koje imaju veliki kapacitet. Ideja o hijerarhijskoj organizaciji memorija potiče još iz 1946.godine, od strane Von Neumann-a.

Na slici 1 prikazana je hijerarhijska organizacija memorijskog sistema. Na vrhu hijerarhije se nalaze procesorski registri koji omogućavaju čuvanje podataka unutar samog procesora. U registrima se nalaze podaci kojima se najbrže pristupa. Zatim sledi brza i skupa keš memorija relativno malog kapaciteta, kao prelaz ka operativnoj memoriji znatno većeg kapaciteta, ali manje brzine. Na sledećem nivou su sekundarne memorije, kao što su hard disk, optički diskovi, kao što su CD, DVD i USB flash. Na kraju hijerahije se nalaze tercijalne memorije, gde se primarno misli na magnetne trake, koje se u današnje vreme dosta retko koriste.



Slika 2.1: Hijerarhijska organizacija memorijskog sistema

Na slici su strelicama označeni parametri po kojima je uspostavljena hijerarhija: brzina, kapacitet i cena po bitu. Smer strelice ukazuje na porast određenog parametra.

Poznavanje memorijske hijerarhije je važno zato što može značajno da utiče na performanse napisanog softvera. Na primer, ukoliko su podaci koje softver koristi smešteni u bržim memorijskim jedinicama, program će se brže izvršavati. Efikasnost hijerarhijske organizacije zasniva se na principu ne tako čestog prenošenja podataka u bržu memoriju uz njihovo višestruko korišćenje pre nego što se zamene novim podacima. Ovaj princip je ostvarljiv

zahvaljujući fenomenu lokalnosti referenci. Lokalnost referenci potiče od činjenice da, u datom vremenskom periodu, programi često pristupaju nekom određenom delu memorije.

Postoje dva oblika lokalnosti, i to:

- Prostorna lokalnost,
- Vremenska lokalnost.

Prostorna lokalnost je pojava da, ukoliko se program referiše na jednu adresu, postoji velika verovatnoća da će se uskoro referisati i na obližnje adrese u memoriji (na primer, na uzastopne instrukcije pri sekvencijalnom izvršavanju programa). Vremenska lokalnost je fenomen da ako se program referiše na neku adresu u memoriji, postoji velika verovatnoća da će joj uskoro opet pristupiti (na primer, nekoj instrukciji u programskoj petlji).

3. KEŠ MEMORIJA

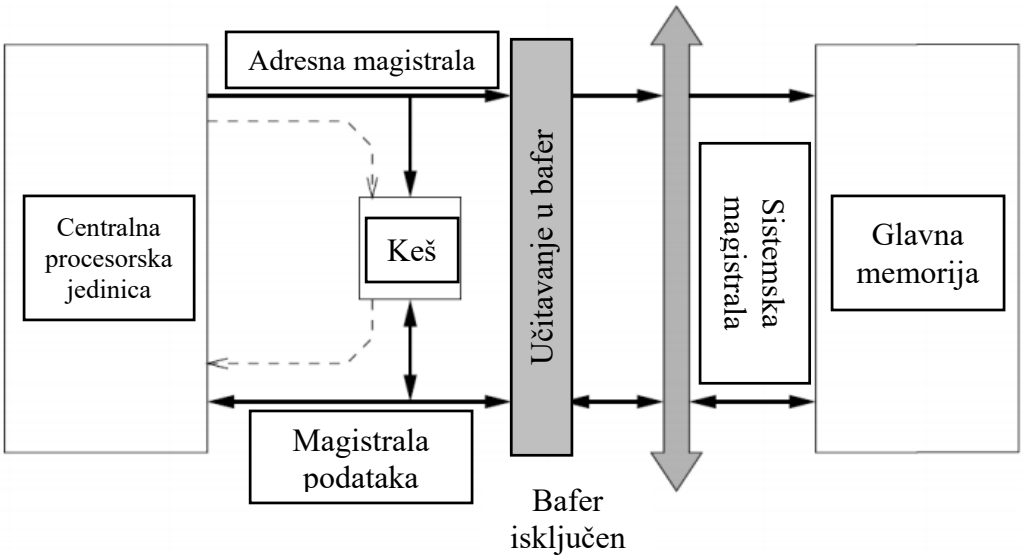
Samu ideju o keš memoriji prvi je predstavio Wilkes 1965. godine, dok je sam termin nastao nešto kasnije. Uvođenje keš memorije je omogućilo brži pristup sadržaju operativne memorije, što je dovelo do značajnog ubrzanja rada računara. Suština ideje je da se mali deo operativne memorije za koji se očekuje da će uskoro biti korišćen, kopira u keš memoriju, koja je znatno brža od operativne, a nalazi se unutar procesora ili vrlo blizu njemu.

Kada procesor zahteva neki podatak, prvo se proverava da li se on nalazi u keš memoriji, pa ako se nalazi, podatak se uzima iz keš memorije, čime se postižu značajne uštede u vremenu.

Bafer je pojam koji se koristi u informacionoj tehnologiji i usko je vezan za računare. Sam pojam je proizašao od engleske reči buffer, što u bukvalnom prevodu znači – amortizer, ublaživač. U našem jeziku, ovaj pojam se uzima i kao sinonim za međuspremnik, pa se bafer, često tako i naziva.

Bafer podrazumeva deo memorije jednog računara, koji je smešten kao izdvojeni deo, i njegova svrha je da beleži potrebne podatke pri izvođenju određenih radnji na računaru. U širem značenju, bafer predstavlja bilo koji medijum na koga se spremaju podaci (koji su već negde, na nekom mestu postojali), i tada ima za svrhu da se i dalje koristi. Bafer nije posebno uslovljen da radi po FIFO principu (First In, First Out, engl.), već redosled kojim se bafer popunjava i čita isključivo zavisi od problema koji se u tom trenutku rešava, a ima veze sa računarskom memorijom. Sam proces kojim se bafer puni podacima se naziva bafovanje, tj. međuspremanje određenih podataka sa računara. Softver u memoriji računara, koji pravi prostor za privremeno smeštanje podataka se, ustvari naziva – bafer.

Na slici 3.1 prikazan je princip rada keš memorije koja se nalazi unutar procesora. Keš memorija je putem linija podataka i adresnih linija povezana sa ostalim komponentama procesora, kao i sa operativnom memorijom.



Slika 3.1: Princip rada keš memorije

Kada procesor želi da pristupi nekom podatku, generiše se adresa lokacije u Operativnu memoriju u kojoj se nalazi taj podatak i prosleđuje je do Keš memorije, gde se proverava da li se traženi podatak nalazi u Keš memoriji. Pošto je u početnom trenutku Keš memorija prazna, podatak se ne nalazi u Keš memoriji, već u Operativnoj memoriji. Zato se taj podatak, kao i podaci na susednim adresama u Operativnoj memoriji, prenose u Keš memoriju. Nakon toga, podatak se čita iz Keš memorije i prosleđuje procesoru. Ovaj postupak se ponavlja za sve podatke potrebne procesoru. Postepeno, Keš memorija se puni podacima, i u jednom trenutku će zahtevani podatak biti u Keš memoriji. Tada će on moći direktno da se pročita iz Keš memorije, bez obraćanja Operativnoj memoriji, pa će pristup podatku biti brži.

Sam procesor generiše jednu adresu za pristup podatku. Međutim, podatak se može nalaziti ili u Keš memoriji ili u Operativnoj memoriji, na različitim adresama zbog razlika u kapacitetima ove dve memorije. Prema tome, da bi se pristupilo podatku, potrebno je obezbediti mapiranje adrese u skladu sa memorijom kojoj se pristupa. Funkciju mapiranja obavlja jedinica za upravljanje memorijom, mada može da bude i posebno integrisano kolo.

Da bi proces keširanja podataka bio moguć, potrebno je rešiti tri problema:

- Problem evidencije blokova Operativne memorije koji se trenutno nalaze u Keš memoriji,
- Problem odlučivanja o tome koji blok treba izbaciti iz pune Keš memorije da bi se oslobodio prostor za upis novog bloka Operativne memorije,
- Problem ažuriranja sadržaja Operativne memorije u slučaju promene sadržaja Keš memorije.

3.1. Tehnike preslikavanja keš memorije

Mehanizam rada Keš memorije zahteva da ona bude podeljena u blokove koji se sastoje od određenog broja uzastopnih memorijskih lokacija. Broj lokacija u bloku se naziva dužinom

bloka. Na sličan način je podeljena i Operativna memorija. Dužine blokova Operativne memorije i Keš memorije su iste. Blokovi obe memorije su numerisani počevši od 0.

Tehnike preslikavanja definišu postupak unosa bloka iz Operativne memorije u Keš memoriju. Pri upisu bloka Operativne memorije u blok Keš memorije, redosled lokacija unutar blokova se ne menja.

U najznačajnije tehnike preslikavanja se ubrajaju:

- Direktno preslikavanje,
- Asocijativno preslikavanje,
- Set-asocijativno preslikavanje.

3.1.1. Direktno preslikavanje

Direktno preslikavanje je najjednostavnija tehnika preslikavanja. Zasniva na tome da se određeni blok Operativne memorije smešta uvek u isti blok Keš memorije.

Pretpostavimo da je u Keš memoriji potrebno upisati i-ti blok Operativne memorije. Neka je NCB ukupan broj blokova u Keš memoriji. Broj bloka j u Keš memoriji u koji treba upisati dolazeći blok Operativne memorije se računa po formuli:

$$\mathbf{j = i \bmod NCB}$$

Više blokova Operativne memorije se preslikava u isti blok Keš memorije, pa je ovo više-na-jedan tehnika mapiranja. Na primer, za NCB = 8, blokovi 14, 22 i 30 iz Operativne memorije se upisuju u blok 6 u Keš memoriji (zato što je 14 mod 8 = 22 mod 8 = 30 mod 8 = 6).

Pošto za funkcionisanje keš mehanizma nije dovoljno samo čuvati blokove podataka u Keš memoriji, već se o njima mora voditi i evidencija o poreklu, Keš memorija je podeljena na dva dela. U prvom delu, koji se naziva memorija tagova čuvaju se podaci o evidenciji blokova, dok se u drugom delu, koji se naziva memorija podataka čuvaju blokovi podataka iz Operativne memorije.

Na slici 3 je predstavljen primer direktnog preslikavanja. Neka Operativna memorija sadrži NMB = 4096 blokova, a Keš memorija 128 blokova. Veličina bloka je B = 16 memorijskih lokacija (reči).

TM	KM	DM					OM		
3	0	384	←--------	0	128	256	384	...	3968
1	1	129	←--------	1	129	257	385	...	3969
0	2	...	←--------	2	130	258	386	...	3970
...		...		...					
...	126	...		...					
31	127	4095	←--------	127	255	383	...	...	4095
				0	1	2	3		31

Slika 3.1.1.1: Primer direktnog preslikavanja

Blokovi Operativne memorije su organizovani u 128 vrsta po 32 bloka (4096/128 = 32). Memorija podataka se sastoji od 128 blokova (označenih od 0 do 127) u koje su upisani blokovi Operativne memorije 384, 129,..4095.

Svi blokovi iz prve vrste u Operativnoj memoriji se preslikavaju u blok 0 u Memoriju podataka. Takođe, svi blokovi iz druge vrste Operativne memorije se preslikavaju u blok 1 u Memoriju podataka, itd. S obzirom da se više blokova Operativne memorije iz jedne vrste može upisati u isti blok Keš memorije, neophodno je uvek imati evidenciju o tome koji blok iz vrste je trenutno upisan. To se postiže pomoću Memorije tagova. Za svaki blok u Keš memoriji postoji odgovarajuća lokacija u Memoriji tagova, koja sadrži redni broj bloka Operativne memorije u odgovarajućoj vrsti koji se trenutno nalazi u Keš memoriji. Ovaj redni broj se naziva tag. Na primer, sadržaj prve lokacije u Memoriji tagova je 3 zato što se u bloku 0 u Keš memoriji nalazi blok 384 iz Operativne memorije koji je u trećoj koloni Operativne memorije.

Kod direktnog preslikavanja, adresa podatka dobijena od procesora se deli u tri polja: polje taga (T bitova), polje bloka Keš memorije (CB bitova) i polje memorijske reči unutar bloka (W bitova).

Dužine polja (u broju bitova) određuju se na sledeći način:

$$\mathbf{T = \log_2 (NMB/NCB)}$$
$$\mathbf{CB = \log_2 NCB \; W = \log_2 B}$$

Dužina adrese (A bitova) memorijske lokacije u Operativnoj memoriji koju je generisao procesor, računa se kao:

$$\mathbf{A = \log_2 (B \cdot NMB)}$$

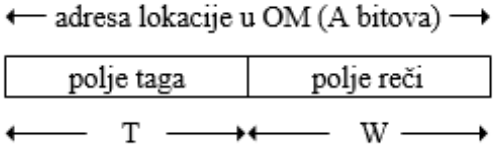
Prednost direktnog preslikavanja je u jednostavnosti postupka i direktnom određivanju mesta u Keš memoriji gde će biti smešten blok Operativne memorije. Nedostatak je u

neefikasnom korišćenju Keš memorije. Česte su situacije da više blokova konkuriše za isto mesto u Keš memoriji, a da su neki drugi blokovi u Keš memoriji prazni.

3.1.2. Asocijativno preslikavanje

Asocijativno preslikavanje je fleksibilnije od direktnog preslikavanja, zato što se kod ove tehnike blok Operativne memorije može smestiti u bilo koji slobodan blok Keš memorije.

U slučaju asocijativnog preslikavanja, adresa koju generiše procesor se interpretira pomoću dva polja, i to: polja taga (*T bitova*) i polja memorijske reči (*W bitova*).



Slika 3.1.2.1: Interpretacija adrese kod asocijativnog preslikavanja

Polje taga jednoznačno identifikuje blok Operativne memorije koji se nalazi u Keš memoriji, dok polje memorijske reči identifikuje traženi podatak unutar bloka.

Dužine polja (u broju bitova) se određuju na sledeći način:

T = log2 NMB(NMB je broj blokova u OM)
W = log2 B (B je broj memorijskih reči u bloku)

NMB predstavlja broj blokova u Operativnoj memoriji, dok *B* označava broj memorijskih reči u bloku.

Dužina adrese (*A bitova*) memorijske lokacije u *OM* koju je generisao procesor, računa se kao:

A = log2 (B·NMB)

Kada od procesora dobije adresu podatka, čija je struktura (*T-polje, W-polje*), Jedinica upravljanja memorijom izvršava sledeći protokol:

- Ispituje da li se vrednost T-polja nalazi među postojećim tagovima u Memoriji tagova,
- Ako je u Keš memoriji, podatak se nalazi u bloku Keš memorije koji odgovara tagu, pa mu se može pristupiti na osnovu sadržaja W-polja,
- Ako podatak nije u Keš memoriji, potrebno je blok iz Operativne memorije preneti u Keš memoriju, podatak proslediti procesoru i ažurirati Memoriju tagova i Memorije podataka.

Suštinu asocijativnog preslikavanja predstavlja pretraživanje tagova u Memoriji tagova. Ukoliko bi ovo pretraživanje bilo sekvencijalno, onda bi trajalo neprihvatljivo dugo. Zato se memorija tagova realizuje kao asocijativna memorija koja omogućava istovremeno pretraživanje.

Paralelno pretraživanje zahteva dodatni hardver, što poskupljuje realizaciju i predstavlja glavni nedostatak tehnike asocijativnog preslikavanja.

Prednost asocijativnog preslikavanja je u efikasnom korišćenju Keš memorije, jer nema restrikcija po pitanju smeštaja dolazećeg bloka Operativne memorije.

3.1.3. Set asocijativno preslikavanje

Set-asocijativno preslikavanje predstavlja kompromis između direktnog i asocijativnog preslikavanja. Ideja je da se zadrže jednostavnost direktnog i efikasnost asocijativnog preslikavanja tako što bi se Keš memorija podelila u setove sa određenim brojem blokova. Blok Operativne memorije bi se uvek preslikavao u isti set u Keš memoriji, ali u bilo koji slobodan blok u tom setu.

Pretpostavimo da je Keš memorija podeljena u NS setova. Blok i u Operativnoj memoriji se mapira u set ***s*** Keš memorije prema formuli:

$$\mathbf{s = i \bmod NS}$$

Unutar seta s, i-ti blok Operativne memorije se može smestiti u bilo koji blok Keš memorije koji pripada tom setu. Unutar seta *s*, i-ti blok Operativne memorije se može smestiti u bilo koji blok Keš memorije koji pripada tom setu.

Kod ove tehnike, adresa podatka koju generiše procesor se interpretira pomoću tri polja: polja taga (T bitova), polja seta (S bitova) i polja memorijske reči (W bitova). Polje taga identifikuje blok u setu, polje seta identifikuje set, dok polje memorijske reči identifikuje traženi podatak unutar bloka.

Ako je broj blokova u setu BS, dužine polja određuju se na sledeći način:

$$\mathbf{T = \log_2 (NMB \cdot BS/NCB)}$$
$$\mathbf{S = \log_2 NS}$$
$$\mathbf{W = \log_2 B}$$

Pri čemu NMB označava broj blokova u Operativnoj memoriji, a NCB u Keš memoriji. B predstavlja broj memorijskih reči u bloku.

Dužina adrese (A bitova) memorijske lokacije u OM koju je generisao procesor, računa se kao:

$$\mathbf{A = \log_2 (B \cdot NMB)}$$

Kada dobije adresu podatka, čija je struktura (T-polje, S-polje, W-polje), Jedinica upravljanja memorijom izvršava sledeći protokol:

- Na osnovu sadržaja S-polja direktno određuje set u koji se blok Operativne memorije mapira,

- Ispituje da li se sadržaj T-polja nalazi među postojećim tagovima u Memoriji tagova za taj set,
- Ako je podatak u Keš memoriji, pronalazi se unutar odgovarajućeg bloka u Memoriji podataka na osnovu vrednosti u W-polju,
- Ako podatak nije u Keš memoriji, potrebno je blok iz Operativne memorije preneti u Keš memoriju, podatak proslediti procesoru i ažurirati Memoriju podataka i Memoriju tagova.

Navedena tehnika nije toliko efikasna kao asocijativno preslikavanje, ali je preuzela jednostavnost direktnog preslikavanja.

3.2. Tehnike zamene Keš memorije

Tehnike zamene rešavaju problem izbora bloka koji će biti izbačen iz pune Keš memorije kako bi se na njegovo mesto upisao dolazeći blok Operativne memorije. Potreba za tehnikama zamene postoji samo kod asocijativnog i set-asocijativnog preslikavanja, dok se u slučaju direktnog preslikavanja unapred zna lokacija upisa dolazećeg bloka.

Izbor bloka koji će biti zamenjen se može sprovesti:

- Po slučajnom izboru,
- Po vremenu koje su upisani blokovi proveli u Keš memoriju (FIFO tehnika),
- Po trenucima poslednjeg korišćenja blokova upisanih u Keš memoriju (LRU tehnika).

3.2.1. Tehnika slučajnog izbora

U računaru postoji generator slučajnih brojeva koji po uključenju računara počne da generiše brojeve iz zadatog opsega. Tehnika slučajnog izbora koristi izlaz ovog generatora u trenutku zamene na osnovu koga određuje broj bloka Keš memorije koga treba zameniti blokom Operativne memorije. Ova tehnika je prvi put bila upotrebljena u Intel-ovoj iAPX seriji mikroprocesora. Tehnika je vrlo jednostavna, a njen glavni nedostatak je u tome što ne uzima u obzir lokalnost referenci, pa se može desiti da se izbaci blok koji bi trebalo uskoro da se koristi.

3.2.2. Fifo tehnika

FIFO (*First-In-First-Out*) tehnika kao kriterijum uvodi vreme koje su blokovi proveli u Keš memoriji od svog unosa do tekućeg trenutka. Izbor se pravi tako što se iz Keš memorije izbacuje onaj blok koji je najviše vremena proveo u njoj. Dakle, to je blok koji je, od svih trenutno raspoloživih blokova u Keš memoriji, prvi unet.

Za sprovođenje ove tehnike, neophodno je vođenje evidencije o vremenima upisa blokova u Keš memoriju, što ovu tehniku čini složenijom od tehnike slučajnog izbora. FIFO tehnika je pogodna za programe koji se pravolinijski izvršavaju jer kod njih lokalnost referenci nije bitna.

3.2.3. LRU tehnika

Najefikasniji postupak zamene je LRU (*Least Recently Used)* tehnika. Ona se zasniva na praćenju istorije korišćenja blokova koji se nalaze u Keš memoriji, za šta je zadužen keš kontroler. Kriterijum za izbacivanje bloka iz Keš memorije je vreme njegovog poslednjeg korišćenja. Naime, iz Keš memorije se izbacuje blok sa najranijim vremenom poslednjeg korišćenja.

Keš kontroler može da prati istoriju korišćenja blokova na različite načine. Jedna moguća realizacija je da se svakom bloku Keš memorije pridruži brojač. Vrednost brojača je veća ako je blok ranije korišćen. Radi lakšeg razumevanja realizacije, može se smatrati da se brojači nalaze u listi po redosledu korišćenja njima odgovarajućih blokova. Na čelu liste je brojač bloka koji je poslednji bio korišćen. Kada procesor želi da pristupi nekom podatku, najpre se proverava da li se blok sa tim podatkom nalazi u Keš memoriji. Ako se nalazi, pročita se trenutna vrednost njegovog brojača b.

Nakon pristupa podatku, brojači blokova Keš memorije se ažuriraju na sledeći način:

- Brojač bloka u kome se nalazi podatak se resetuje na 0,
- Svi brojači sa vrednošću manjom od b se inkrementiraju za 1,
- Svi brojači sa vrednošću većom od b se ne menjaju.

U slučaju da se traženi podatak ne nalazi u Keš memoriji, potrebno je uraditi sledeće:

- Blok Keš memorije sa najvećom vrednošću brojača zameniti blokom Operativne memorije u kome se nalazi podatak,
- Brojač zamenjenog bloka postaviti na 0,
- Vrednosti svih ostalih brojača inkrementirati za 1.

3.3. Tehnike ažuriranja

Tehnike ažuriranja se bave problemom koherencije, odnosno usklađivanja sadržaja Keš memorije i Operativne memorije. U Keš memoriji se u svakom trenutku nalazi određen broj kopija blokova Operativne memorije. Procesor može da pristupa podacima u Keš memoriji i da menja njihove vrednosti. Svaka promena, učinjena samo u bloku Keš memorije, dovodi do neusaglašenosti između sadržaja tog bloka i njegovog originala koji se nalazi u Operativnoj memoriji. Za korektan rad, neophodno je usklađivanje sadržaja Operativne memorije sa promenama u Keš memoriji, što omogućavaju tehnike ažuriranja.

Postoje četiri tehnike ažuriranja:

- Tehnika upisa ako blok postoji u Keš memoriji,
- Tehnika upisa ako blok ne postoji u Keš memoriji,
- Tehnika čitanja ako blok postoji u Keš memoriji,
- Tehnika čitanja ako blok ne postoji u Keš memoriji.

Tehnika upisa ako blok postoji u Keš memoriji se bavi problemom ažuriranja u slučaju upisa podatka u neki blok Keš memorije. Ova tehnika pruža dve mogućnosti ažuriranja. Prva mogućnost je trenutni upis kod koga se svaka operacija upisa realizuje tako što se istovremeno podatak upisuje i u Keš memoriju i u odgovarajući blok Operativne memorije. Druga mogućnost je odloženi upis kod koga se podatak upisuje samo u blok Keš memorije, dok se upis u

Operativnu memoriju odlaže sve dok se ne pojavi potreba da se konkretan blok izbaci iz Keš memorije. U trenutku zamene bloka Keš memorije, ispituje se da li je taj blok menjan. Ukoliko jeste, blok Keš memorije se upisuje u blok Operativne memorije, a ako nije, onda se samo blok Keš memorije zameni dolazećim blokom Operativne memorije. Da bi pomenuto ispitivanje bilo moguće, neophodno je svakom bloku Keš memorije pridružiti jedan bit koji ima vrednost 1, ako je postojao bar jedan upis u taj blok, inače ima vrednost 0.

Tehnika upisa ako blok ne postoji u Keš memoriji se bavi problemom ažuriranja u slučaju upisa podatka u blok Operativne memorije koji nije ranije prenet u Keš memoriju. Ovde, takođe, postoje dve mogućnosti ažuriranja. Prva je dovlačenje bloka, što podrazumeva prenos i upis bloka Operativne memorije u Keš memoriju, nakon čega se primenjuje prethodno opisana tehnika. Druga mogućnost je nedovlačenje bloka, i tada se podatak upisuje samo u blok Operativne memorije.

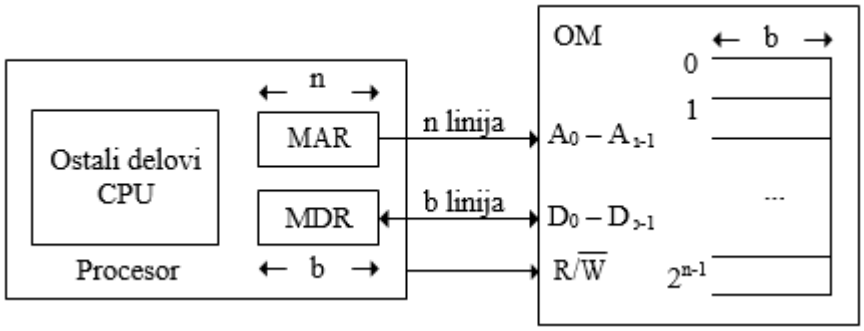
Tehnika čitanja ako blok postoji u Keš memoriji je jednostavna jer se podatak direktno čita iz Keš memorije.

Tehnika čitanja ako blok ne postoji u Keš memoriji, se bavi problemom čitanja podatka iz nekog bloka Operativne memorije. Tehnika pruža dve mogućnosti. Prva mogućnost je direktno prosleđivanje, pri čemu se blok Operativne memorije prenosi u Keš memoriju, a podatak se odmah po pristizanju u Keš memoriju prosleđuje procesoru. Druga mogućnost je prosleđivanje sa zadrškom, pri čemu se prvo ceo blok Operativne memorije upiše u Keš memoriju, pa se tek onda pročita podatak iz njega i prosledi procesoru.

4. OPERATIVNA MEMORIJA

Operativna memorija (*Main Memory*) je radna memorija računara u kojoj se čuvaju trenutno aktivni programi i podaci. Po aktiviranju, aplikacija se upisuje u operativnu memoriju, nakon čega može da se koristi. Pri izlasku iz aplikacije, deo memorije u kome je bila smeštena aplikacija se oslobađa i stavlja na raspolaganje drugim programima.

Komunikacija između procesora i operativne memorije je prikazana na slici 5. Veza između ove dve komponente se obično ostvaruje preko dva registra, MAR (prihvata adrese memorijskih lokacija) i MDR (prihvata podatke). Veličina MAR (n bitova) je usklađena sa kapacitetom Operativne memorije (2n ćelija), dok veličina MDR (b bitova) odgovara veličini ćelije u memoriji (b bitova). MAR je povezan sa Operativnom memorijom preko adresnih linija memorije, a MDR preko linija za podatke.



Slika broj 4.1: Povezivanje procesora i Operativne memorije

Unutrašnja struktura Operativne memorije se može vizualizovati u vidu mreže osnovnih ćelija (*BC - Basic Cells*) raspoređenih po vrstama i kolonama. Svaka osnovna ćelija sadrži informaciju od 1 bita. Sadržaji svih ćelija u jednoj vrsti čine jednu memorijsku reč, tj. sadržaj jedne memorijske lokacije.

Sastavni deo Operativne memorije je i adresni dekoder čiji su ulazi direktno povezani na adresne linije memorije. Na osnovu prispele adrese, adresni dekoder generiše signale na selekcionim linijama koje su povezane sa osnovnim ćelijama. Takođe, na osnovne ćelije se dovode i linije podataka memorije (vertikalne linije).

U datom trenutku, aktivira se samo jedan izlaz adresnog dekodera, tj. jedna selekciona linija, dok su sve ostale selekcione linije neaktivne. Aktivna selekciona linija se koristi za upis podatka koji se trenutno nalazi na linijama podataka u sve osnovne ćelije u vrsti, ili za čitanje podataka iz svih osnovnih ćelija u vrsti i njihovo postavljanje na linije podataka.

Svaka osnovna ćelija se realizuje skupom tranzistora povezanih na odgovarajući način, tako da može da bude samo u dva stabilna stanja (0 ili 1). Zbog načina realizacije, osnovne ćelije su obično priključene na dve linije podataka, pri čemu jedna služi za ulaz sadržaja u ćeliju, a druga za izlaz.

Operativna memorija se realizuje u integrisanoj tehnologiji u vidu čipova. Memorijski čipovi istog kapaciteta mogu da imaju različit broj pinova u zavisnosti od unutrašnje organizacije memorije.

Pri projektovanju memorijskog sistema računara, Operativna memorija se realizuje u vidu više čipova određenog kapaciteta. Kapacitet čipa može da bude ograničavajući faktor koji utiče na realizaciju.

5. VIRTUELNA MEMORIJA

Kada procesor zahteva neku programsku instrukciju ili neki podatak, oni bi trebalo da se nalaze u operativnoj memoriji. Međutim, s obzirom da Operativna memorija ima ograničen

kapacitet, može se desiti da u njoj nema dovoljno slobodnog prostora za smeštaj željenog programa i podataka.

Povećanje kapaciteta Operativne memorije bi značajno povećalo njenu cenu. U praksi, nije neophodno da se ceo program koji se izvršava ili svi potrebni podaci nalaze u Operativnoj memoriji. Delovi programa koji se trenutno ne izvršavaju mogu da budu na hard disku, dok su delovi programa koji se trenutno izvršavaju u Operativnoj memoriji. Ovakav način rada zahteva uvođenje mehanizama za prenos delova programa ili podataka iz Operativne memorije na hard disk i obrnuto. Ovi mehanizmi čine koncept virtuelne memorije.

Tehnika Virtuelne memorije služi za proširivanje fizičke veličine Operativne memorije na potrebnu veličinu. Prenos delova programa između Operativne memorije i hard diska je transparentan za korisnika i obavlja ga operativni sistem.

Princip rada Virtuelne memorije je sličan principu rada keš memorije. Razlike su uglavnom implementacione. Međutim, cena promašaja u virtuelnom je od 100 do 200 puta veća od cene promašaja u keš pristupu. To je razumljivo s obzirom da se radi o različitim nivoima u memorijskoj hijerarhiji. Ovde se pojam cene pre svega odnosi na vreme potrebno za pristup podatku.

Kod savremenih računara, prilikom pisanja programa, programeri imaju na raspolaganju celokupnu memoriju, tj. Operativnu memoriju i hard disk. Adrese koje se generišu u programima nazivaju se virtuelnim adresama, a njihov opseg, virtuelni adresni prostor (*VAP*). Ove adrese ne odgovaraju fizičkim adresama u Operativnoj memoriji. Opseg adresa u Operativnoj memoriji predstavlja realni adresni prostor (*RAP*), a adresa lokacije u Operativnoj memoriji se naziva realnom adresom. Za prevođenje virtuelne adrese u odgovarajuću realnu adresu u Operativnoj memoriji, zadužena je jedinica za upravljanje memorijom (*MMU*).

Da bi virtuleni pristup mogao da funkcioniše, neophodno je voditi evidenciju o tome koji delovi programa se trenutno nalaze u Operativnoj memoriji i u kojem njenom delu. Za vođenje evidencije se koriste tabele preslikavanja koje se nalaze u Operativnoj memoriji.

Prema načinu podele virtuelnog adresnog prostora, postoje tri tipa Virtuelne memorije:

- Virtuelna memorija sa straničnom organizacijom,
- Virtuelna memorija sa segmentnom organizacijom,
- Virtuelna memorija sa segmentno-straničnom organizacijom.

5.1. Stranična organizacija

Stranična organizacija Virtuelne memorije podrazumeva podelu Virtuelnog adresnog prostora na delove fiksne veličine koji se nazivaju stranicama i podelu Realnog adresnog prostora na delove fiksne veličine koji se nazivaju blokovima. Veličina stranice je jednaka veličini bloka (obično 2 do 16 KB).

Pri izboru veličine stranice (a samim tim i bloka), treba voditi računa da ona ne bude ni previše mala (često bi se pristupalo hard disku, a to dugo traje), ni previše velika (mnogi preneti delovi stranice se ne bi koristili). Stranice se smeštaju u blokove Operativne memorije. Kada se

Operativna memorija napuni, neka od stranica se prenosi na hard disk, kako bi se na njeno mesto upisala nova stranica sa hard diska (koja sadrži traženi podatak).

Kod stranične organizacije Virtuelne memorije, virtuelna adresa sadrži dva polja:

- Page (broj stranice, dužina ovog polja je p bitova),
- Word (adresa reči unutar stranice, dužina ovog polja je w bitova).

Realna adresa, takođe, ima dva polja:

- Block (broj bloka, dužina ovog polja je b bitova),
- Word (adresa reči unutar bloka, dužina ovog polja je w bitova).

Dakle, veličina Virtuelnog adresnog prostora je 2^{p+w}, a veličina Realnog adresnog prostora je 2^{b+w}adresa. Evidencija o trenutnom sadržaju Operativne memorije se vodi pomoću tabele stranica (*PT – Page Table*). PT se nalazi u Operativnoj memoriji, a njena početna adresa u posebnom procesorskom registru PTBR (*Page Table Base Register*). Pošto u Virtuelnom adresnom prostoru ima 2^p stranica, PT ima 2^p ulaza (vrsta). Svakom ulazu odgovara po jedan deskriptor stranice, koji sadrži sve potrebne informacije o toj stranici. Na osnovu sadržaja PTBR i broja stranice, Jedinica upravljanja memorijom može da pristupi deskriptoru te stranice.

Deskriptor stranice (*DS*) se sastoji od četiri polja:

- V (1 bit)- statusni bit koji pokazuje da li je stranica u Operativnoj memoriji ili nije, ovaj bit postavlja operativni sistem prilikom dovlačenja stranice sa hard diska u Operativnu memoriju, bit omogućava Operativnom sistemu da stranicu proglasi nevalidnom bez njenog uklanjanja iz Operativne memorije,
- D (1 bit)- statusni bit koji pokazuje da li je stranica modifikovana ili nije, ovaj bit postavlja Jedinicu upravljanja memorijom prilikom operacije upisa, a koristi ga Operativni sistem da odluči da li stranicu treba upisati na hard disk pre izbacivanja iz Operativne memorije,
- Blok (b bitova)- broj bloka Operativne memorije u kome se nalazi data stranica (važi ako je V = 1), polje postavlja Operativni sistem prilikom dovlačenja stranice sa hard diska, a koristi ga Jedinica upravljanja memorijom za formiranje realne adrese,
- Disk (d bitova)-adresa stranice na hard disku.

Postupak preslikavanja se odvija u sledećim koracima:

- Polje page u virtuelnoj adresi predstavlja broj stranice, a istovremeno i broj ulaza u Page Table-u, u kome se nalazi deskriptor te stranice, ako svaki Deskriptor stranice zauzima po 2l memorijskih reči, sadržaj polja page treba pomeriti za l mesta ulevo (što odgovara množenju sa 2l) kako bi se dobio ofset (pomeraj) početka deskriptora u odnosu na početak Page Table-a,
- U registru PTBR se nalazi adresa početka Page Table-a, kada se sadržaj ovog registra sabere sa ofsetom dobijenim u koraku 1, dobija se adresa početne lokacije deskriptora stranice,
- Počev od adrese dobijene u koraku 2, iz Deskriptora stranice se čita reč koja odgovara polju

V i na osnovu nje proverava da li se stranica trenutno nalazi u Operativnoj memoriji,

- Ako je stranica u Operativnoj memoriji, iz Deskriptora stranice se čita sadržaj polja blok koji predstavlja broj bloka Operativne memorije u kome se stranica nalazi, konkatenacijom (spajanjem) sadržaja polja blok i polja word iz virtuelne adrese, formira se realna adresa,
- Ako stranica nije u Operativnoj memoriji, generiše se prekid u cilju dovlačenja stranice sa hard diska, ovaj deo posla odrađuje Operativni sistem softverskim putem.

6. MAGISTRALA

Osnovna uloga magistrale je da omogući prenos sadržaja između komponenata računara. Sadržaj se obično prenosi između procesorskih registara, memorijskih lokacija i registara U/I uređaja. Osim ove, uloga magistrale je i da obezbedi takt sistemu, prenosi upravljačke signale, vrši arbitraciju pristupa raznih jedinica na magistralu, omogući konfigurisanje naprednijih U/I mehanizama prenosa, itd.

Zbog raznolikosti zahteva u pogledu unosa podataka u računar i prikaza rezultata obrade, postoji veliki broj U/I uređaja koji se međusobno razlikuju po funkcionalnosti, načinu transfera podataka, brzini rada i sl. Međutim, bez obzira na razlike, rad sa svim U/I uređajima je organizovan na isti način. To je postignuto zahvaljujući uvođenju univerzalnog modela U/I uređaja.

Svi U/I uređaji se sastoje od periferije i kontrolera periferije. Periferija realizuje osnovnu funkcionalnost uređaja. Kontroler služi za prihvat podataka, upravljanje periferijom i komunikaciju sa ostatkom računara. Dakle, računar pristupa U/I uređaju preko kontrolera periferije, dok mu je sama periferija transparentna.

Kontroler periferije sadrži upravljačku jedinicu (upravljačka logika) i operacionu jedinicu (određen broj registara). Uloga upravljačke logike je da na osnovu sadržaja registara operacione jedinice organizuje preuzimanje podataka iz ulazne periferije ili upis podataka u izlaznu periferiju, kao i njihovo prosleđivanje do/od procesora/memorije. Registri kontrolera omogućavaju da se kompletna organizacija ulaza/izlaza na programskom nivou svede na upis i čitanje sadržaja ovih registara.

Važan pojam u funkcionisanju ulaza/izlaza predstavlja spremnost podatka. Podatak se smatra spremnim za upis u memoriju kada je prenet iz ulazne periferije u registar podatka kontrolera. Slično, podatak je spreman za upis u izlaznu periferiju kada je prenet iz memorije u registar podatka kontrolera. Prenos podatka između registra podatka kontrolera i memorije se može realizovati na dva načina: programski (čitanjem ili upisom u registar podatka kontrolera) ili radom samog kontrolera.

Uobičajeni scenario koji ilustruje princip rada na magistrali sadrži sledeće korake:

- Gazda šalje adresu memorijske lokacije ili registra U/I uređaja na ABUS,
- U slučaju upisa, gazda šalje podatak na DBUS,
- Sve sluge koje su povezane na ABUS dobijaju poslatu adresu,

- Pomoću svojih dekodera adresa, sve sluge proveravaju da li adresa odgovara nekoj od njihovih adresa,
- Samo jedan sluga prepoznaje da se poslata adresa odnosi na njega, sa malim zakašnjenjem (dovoljnim da sluge provere adresu), gazda šalje na CBUS signal upisa/čitanja svim slugama,
- Upis/čitanje izvršava samo sluga koji je adresiran poslatom adresom.
- U slučaju čitanja, pročitani podatak se šalje na DBUS i gazda ga prihvata.

6.1. Adresni prostor

Memorijski adresni prostor predstavlja opseg adresa koje se mogu koristiti za adresiranje memorijskih lokacija. *U/I* adresni prostor čini opseg adresa za adresiranje registara unutar U/I uređaja.

Ukoliko se registrima *U/I* uređaja i memorijskim lokacijama pristupa pomoću istih programskih instrukcija, kaže se da je U/I adresni prostor memorijski preslikan. U slučaju da postoje posebne instrukcije za pristup memorijskim lokacijama, a posebne za pristup registrima U/I uređaja, kaže se da su *U/I* i memorijski adresni prostori razdvojeni.

Kontrolna magistrala (*CBUS*) je razložena na tri linije: RDBUS, WRBUS i FCBUS. RDBUS/WRBUS su kontrolne linije po kojima gazda šalje signale upisa/čitanja koji se dovode na ulaze logičkih množača u slugi. Po ABUS stiže adresa koja se dovodi na ulaze dekodera. Ako dekoder prepozna da je posmatrani sluga adresiran, na svom izlazu će generisati signal HIT = 1 i jedno od logičkih kola će generisati signal upravljačkoj jedinici.

Upravljačka jedinica prepoznaje o kom kolu se radi i generiše odgovarajući signal za upis/čitanje memorijske lokacije. U slučaju upisa, podatak se uzima sa DBUS i upisuje u ćeliju, dok se u slučaju čitanja, sadržaj adresirane lokacije prosleđuje na DBUS. Nakon toga, upravljačka jedinica generiše signal završetka upisa/čitanja i šalje ga na FCBUS. Signal sa FCBUS primaju svi u sistemu, ali samo gazda reaguje na njega tako što sadržaj sa DBUS upisuje u svoj MDR i završava čitanje (u slučaju čitanja) ili završava upis (u slučaju upisa).

Pošto se memorija i *U/I* uređaj mogu adresirati istim adresama, neophodno je odrediti kome se gazda obraća. Zato je uvedena nova linija $M/\overline{IO}$ u CBUS. Aktivna vrednost signala na $M/\overline{IO}$, znači da se gazda obraća memoriji, a neaktivna, da se obraća U/I uređaju. U realizaciji, neophodno je logičkim kolima dodati još po jedan ulaz na koji se dovodi $M/\overline{IO}$ signal. Kod U/I uređaja, ovaj ulaz mora biti invertovan, da bi izlaz kola bio aktivan kada je signal $M/\overline{IO}$ neaktivan.

Kada upravljačka jedinica dobije signal sa izlaza nekog logičkog množača, prepoznaje da li se radi o upisu ili čitanju, i generiše odgovarajući signal *RD/WR*. Ovaj signal se dovodi do memorijskih lokacija ili registara, i obavlja se odgovarajuća operacija. Po završetku operacije, upravljačka jedinica generiše signal završetka upisa/čitanja i šalje ga na *FCBUS*.

6.2. Arbitracija

Arbitracija podrazumeva donošenje odluke o tome koja komponenta u datom trenutku može da realizuje ciklus na magistrali. Mehanizam arbitracije je bitan u računarskim sistemima u kojima postoji više jedinica koje mogu da imaju ulogu gazde. U savremenim računarskim sistemima, obično je više procesora i *U/I* uređaja sa DMA prenosom priključeno na magistralu, pa je neophodno u svakom trenutku znati koja jedinica raspolaže magistralom.

Procesom arbitracije upravlja kontroler magistrale ili arbitrator. Značajnu ulogu u ovom procesu imaju tri kontrolna signala:

- BB (*Bus Busy*), koji indicira zauzeće magistrale,
- BR (*Bus Request*), koji predstavlja zahtev za magistralom upućen kontroleru magistrale od strane neke jedinice,
- BG (*Bus Grant*), koji predstavlja signal dozvole za korišćenje magistrale poslat od strane kontrolera magistrale.

Kada neka jedinica treba da realizuje ciklus na magistrali, ona šalje kontroleru zahtev za magistralom. Kontroler joj dodeljuje magistralu, ukoliko ona nije trenutno zauzeta. Postoji više tehnika arbitracije:

- Tehnika ulančavanja (*Daisy chain*),
- Tehnika prozivanja (*Polling*),
- Tehnika nezavisni zahtev/dozvola (*Independent request/grant*).

6.2.1. Tehnika ulančavanja

Tehnika ulančavanja podrazumeva da su jedinice ulančane po prioritetu tako da se jedinica sa najvišim prioritetom nalazi na čelu lanca.

Proces arbitracije otpočinje tako što neke od jedinica šalju kontroleru zahteve za magistralom po BR liniji. Ako je magistrala trenutno slobodna (BB signal je neaktivan), kontroler magistrale šalje signal po BG liniji koji najpre dolazi do jedinice na početku lanca. Kada signal BG stigne do neke jedinice, ona može da reaguje na dva načina:

- Ukoliko nije poslala zahtev za magistralom, da propusti signal do sledeće jedinice u lancu,
- Ukoliko jeste poslala zahtev za magistralom, da zaustavi dalje prosleđivanje signala BG i postavi aktivnu vrednost na BB liniju, čime postaje gazda na magistrali.

Ovakvim postupkom je postignuto da magistralu uvek zauzme jedinica sa najvišim prioritetom.

6.2.2. Tehnika prozivanja

Kod tehnike prozivanja, jedinice nisu međusobno povezane, već je svaka jedinica pojedinačno priključena na kontrolne linije BB, BR i BG. Osim toga, uvedene su i posebne linije

za prozivanje po kojima se šalju adrese jedinica. Broj linija za prozivanje zavisi od broja jedinica. Ako je broj jedinica N, broj adresnih linija je $Log_2 N$.

Kada neka od jedinica želi da ostvari ciklus na magistrali, ona po BR liniji šalje zahtev kontroleru magistrale. U istom trenutku, osim ove jedinice, zahteve mogu da upute i druge jedinice u sistemu. Kada dobije aktivan signal po BR liniji, kontroler magistrale najpre proverava da li je magistrala slobodna ispitivanjm signala na BB liniji. Ako jeste, kontroler proziva jedinice tako što šalje njihove adrese (jednu po jednu) po linijama za prozivanje.

Redosled adresa je u skladu sa unapred utvrđenim prioritetima jedinica (u kontroler magistrale je ugrađena lista adresa po prioritetima). Od svih jedinica koje su poslale zahteve, prva se proziva ona koja ima najviši priritet. Ona prepoznaje svoju adresu i postaje selektovana, a prozivanje se prekida. Nakon toga, kontroler šalje signal BG koga prihvata samo selektovana jedinica. Ona generiše signal BB i postaje gazda na magistrali.

6.2.3. Tehnika nezavisni zahtev/dozvola

Tehnika nezavisni zahtev/dozvola podrazumeva da svaka jedinica ima nezavisnu BR i BG liniju do kontrolera magistrale.

Arbitracija počinje kada jedna ili više jedinica pošalju zahteve po linijama BR kontroleru magistrale. Kontroler određuje koja od jedinica koje su poslale zahteve ima najviši prioritet, pa samo njoj šalje signal dozvole po njenoj BG liniji. Jedinica prima BG signal, postavlja BB signal i postaje gazda na magistrali. Da bi ova tehnika bila izvodljiva, logika koja određuje prioritete jedinica mora biti ugrađena u kontroler magistrale.

6.2.4. Procesor kao arbitrator

U sistemima u kojima procesor ima ulogu arbitratora, magistrala je uvek u posedu procesora. *U/I* uređaj mora procesoru da pošalje zahtev, pa tek kada od njega dobije dozvolu, može da izvrši ciklus na magistrali. Komunikacija između procesora i *U/I* uređaja se ostvaruje preko dve linije:

- *Hreq*, po kojoj *U/I* uređaj šalje zahtev za korišćenje magistrale,
- *Hack*, po kojoj procesor šalje dozvolu za korišćenje magistrale.

Komunikacija počinje tako što ***U/I*** uređaj postavlja signal na hreq liniji na aktivnu vrednost, a zatim čeka dovolu za pristup magistrali u vidu aktivnog signala na hack. Nakon dobijanja dozvole, *U/I* uređaj realizuje ciklus na magistrali, sve vreme držeći aktivnu vrednost signala na hreq. Po završetku ciklusa, *U/I* uređaj postavlja signal na hreq liniji na neaktivnu vrednost. Zatim, procesor postavlja neaktivnu vrednost signala na hack liniju i *U/I* gubi dozvolu za korišćenje magistrale.

U komunikaciji su moguće tri situacije:

- Kada ni procesor ni *U/I* uređaj ne čekaju na realizaciju ciklusa na magistrali. U ovom slučaju, *U/I* uređaj upućuje zahtev u trenutku kada je magistrala slobodna, tako da odmah (sa malim zakašnjenjem) dobija dozvolu i realizuje ciklus,

- Situacija u kojoj *U/I* uređaj upućuje zahtev u trenutku kada procesor već koristi magistralu. Zato, procesor mora prvo da završi svoj ciklus na magistrali, pa tek onda da pošalje dozvolu *U/I* uređaju.
- Moguća je i situacija u kojoj procesor čeka da *U/I* uređaj završi ciklus na magistrali.

Iz ovih situacija se može zaključiti da kada neka jedinica dobije magistralu, ona joj ne može biti oduzeta sve dok ne završi svoj ciklus na magistrali. Na magistrali se mogu realizovati: ciklus upisa, ciklus čitanja i ciklus prihvatanja koda prekida.

6.3. Vrste magistrala

U zavisnosti od dinamike aktivnosti na magistrali, postoje dve vrste magistrala:

- Asinhrone,
- Sinhrone.

Asinhronim se nazivaju one magistrale na koje su priključuju jedinice koje pri radu koriste sopstveni signal takta. Stoga se na asinhrone magistrale mogu priključivati vrlo raznovrsne jedinice. Za sinhronizaciju ovih jedinica, koriste se *handshake* protokoli.

Sinhronim se smatraju one magistrale na koje se priključuju jedinice koje koriste isti (zajednički) signal takta. Ove magistrale imaju fiksni protokol komunikacije, relativno u odnosu na takt. To znači da se unapred zna koliko taktova traje upis/čitanje, pa se nakon tog vremena smatra da je operacija upisa/čitanja završena, tj. pri upisu, da je podatak smešten na odredište, a pri čitanju, da je podatak raspoloživ na DBUS. Pošto je protokol unapred definisan, sinhrona magistrala može biti vrlo brza. Glavni nedostatak sinhrone magistrale je taj što ona mora da bude relativno kratka kako bi prenos u taktu mogao regularno da se obavi.

Kada je reč o načinu rada asinhrone magistrale pri realizaciji različitih ciklusa na magistrali, ciklus čitanja podatka se realizuje tako što gazda šalje adresu podatka na ABUS i aktivan signal na RDBUS, čime zahteva čitanje podatka u slugi. Po završetku čitanja, sluga šalje podatak na DBUS i signal FCBUS koji gazdi signalizira da je podatak raspoloživ.

Ciklus upisa podatka se realizuje tako što gazda šalje adresu na ABUS, podatak na DBUS i aktivan signal na WRBUS, čime startuje upis. Nakon upisa podatka, sluga šalje gazdi signal FCBUS, čime signalizira da mu podatak i adresa više nisu potrebni.

Ciklus prihvatanja koda prekida se realizuje tako što procesor šalje signal potvrde prekida inta i startuje čitanje registra ER u *U/I* uređaju. Nakon čitanja, *U/I* uređaj šalje pročitani sadržaj na DBUS i signal FCBUS da je kod prekida raspoloživ.

Kada je reč o radu sinhrone magistrale pri realizaciji ciklusa čitanja, upisa i prihvatanja koda prekida, ciklus čitanja podatka se realizuje tako što gazda šalje adresu podatka na ABUS i aktivan signal na RDBUS, čime zahteva čitanje u slugi. Pošto je vreme čitanja podatka fiksno,

nakon tog vremena, gazda očekuje da je podatak raspoloživ na DBUS linijama, pa ga upisuje u prihvatni registar podatka, a adresu sa ABUS uklanja.

Pri realizaciji ciklusa upisa, gazda šalje adresu na ABUS, podatak na DBUS i aktivan signal na WRBUS, čime zahteva upis. Pošto je vreme upisa fiksno, posle određenog vremena, gazda očekuje da je podatak upisan i uklanja adresu i podatak sa ABUS i DBUS, respektivno.

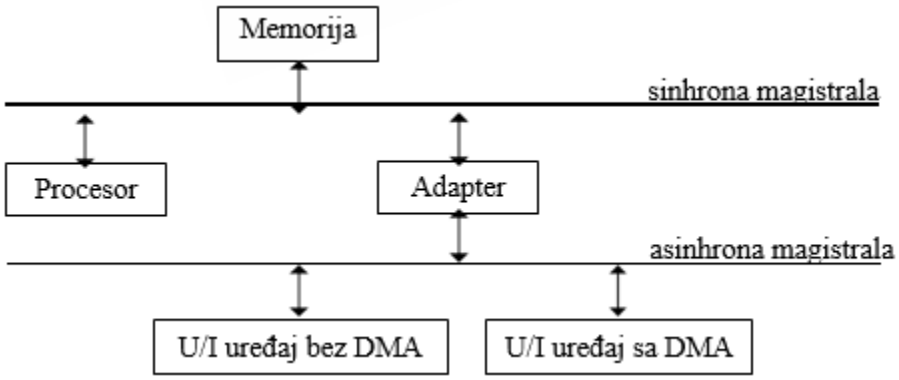
Ciklus prihvatanja koda prekida se realizuje tako što procesor šalje signal potvrde prekida inta i zahteva čitanje registra ER u U/I uređaju. Pošto je vreme čitanja fiksno, nakon tog vremena, gazda očekuje da je kod prekida na DBUS i upisuje ga u svoj prihvatni registar.

Mnogi savremeni računarski sistemi se projektuju tako da imaju više magistrala. Na taj način se može smanjiti vreme čekanja na realizaciju ciklusa na magistrali, a takođe, mogu se efikasno kombinovati osobine asinhrone i sinhrone magistrale.

Uobičajena struktura sistema sa dve magistrale se sasatoji iz sistemske i lokalne magistrale, koje se često realizuju kao asinhrone.

Neki od uređaja su priključeni na sistemsku, a neki na lokalnu magistralu, dok je procesor priključen na obe magistrale. U zavisnosti od toga kako su priključeni uređaji, adresni prostor je podeljen na dva dela. Jedan deo čine adrese za adresiranje lokacija u uređajima priključenim na sistemsku, a drugi deo na lokalnu magistralu. Procesor na osnovu adrese određuje na kojoj magistrali se ciklus realizuje. Zbog postojanja dve magistrale, procesor kao gazda, brže realizuje svoje cikluse na magistrali jer može da koristi obe magistrale.

U nekim konfiguracijama je moguće kombinovati magistrale različitog tipa.



Slika broj 6.3.1: Sistem sa asinhronom i sinhronom magistralom

Adresni prostor je i u ovom slučaju podeljen na dva dela. Jedan deo čine adrese memorijskih lokacija, dok drugi deo čine adrese registara u *U/I* uređajima. Procesor sve cikluse realizuje na sinhronoj magistrali. Na adrese iz *U/I* adresnog prostora reaguje adapter, koji realizuje cikluse sa *U/I* uređajima na asinhronoj magistrali.

7. ZAKLJUČAK

Memorija kod računara je organizovana na hijerahijskom principu. Osnovna podela je na osnovu mogućnosti pristupa od strane procesora, pa se memorija deli na: primarnu i sekundarnu.

Primarna memorija je memorija koja je direktno dostupna procesoru računara preko njegovih glavnih magistrala (adresnih i za podatke), bez potrebe za alternativnim ulazno-izlaznim i često značajno sporijim načinima komunikacije.

Primarna memorija poznata je i kao unutrašnja memorija i obuhvata:

- Procesorske register,
- Internu memoriju procesora keš (cache),
- RAM (glavnu ili radnu memorija),
- ROM, CMOS.

Sekundarna memorija je memorija koja nije direktno dostupna procesoru računara preko njegovih glavnih magistrala (adresnih i za podatke)- preko sistemskog BUS-a. Da bi se došlo do sadržaja sekundarne memorije on prvo mora da se (privremeno) prebaci u primarnu (radnu) memoriju, a po završetku obrade (ukoliko je to potrebno) „vrati“, odnosno adekvatno ažurira odgovrajući sadržaj sekundarne memorije.

Za prisup sekundarnoj memoriji potrebni su dodatni sistemi i često mnoštvo procesorskih instrukcija radi kontrole tih dodatnih sistema. Sadržaj sekundarnih memorija trajno čuvaju (ne gubi se nakon isključivanja napajanja).

Postoji nekoliko različitih vrsta medija koji mogu trajno čuvati podatke. Podaci se mogu trajno pohraniti na:

- Magnetne medije,
- Optičke medije,
- Sekvencijalne poluprovodničke (engl. flash) medije.

Dve bitne osobine svake memorije: kapacitet i vreme pristupa.

Kapacitet predstavlja količinu podataka koji mogu biti smešteni u memoriju. Osnovna jedinica za kapacitet memorije je bajt (B)

- bajt (B) 1B=8b, bit je digitalna cifra i može biti 0 ili 1,
- kilobajt (KB), 1 KB = 1024 b,
- megabajt (MB), 1 MB = 1024 KB,
- gigabajt (GB) 1 GB = 1024 MB,
- terabajt (TB) 1 TB = 1024 GB.

Standardni kapaciteti kod današnjih operativnih memorija su 4GB, 8 GB i više. (Za rad na računaru koji ima Windows 7 neophodno je barem 2 GB memorije. Uz veći kapacitet, rad je brži i lakši. Za ilustraciju količine podataka vezanih za ove jedinice, recimo da je za smeštanje knjige od 200 stranica teksta, dovoljan 1MB, dok samo jedna digitalna fotografija može zahtevati veći kapacitet. Za čuvanje 1 minuta muzike u CD formatu, potrebno je čak 10 MB).

Vreme pristupa označava vreme koje protekne od momenta kada se podatak zatraži iz memorije, do trenutka kada ga ona i isporuči (često se kaže i brzina memorije). Današnje memorije imaju vreme prilaza od 5 do 10 ns (nanosekunda).

Poredeći je sa ostalim (spoljnim) memorijama, mogli bismo da kažemo da je dobra osobina operativne memorije to što je brza, a loše što je srazmerno malog kapaciteta, što je skupa (oko 0,2 evra za 1 MB), i što gubi sadržaj nakon isključenja računara iz struje.

LITERATURA

[1]Tomašević V. „*Osnovi arhitekture i organizacije računara*“, (2019), Univerzitet Singidunum, Beograd.