7. Projektiranje memorijskog sustava

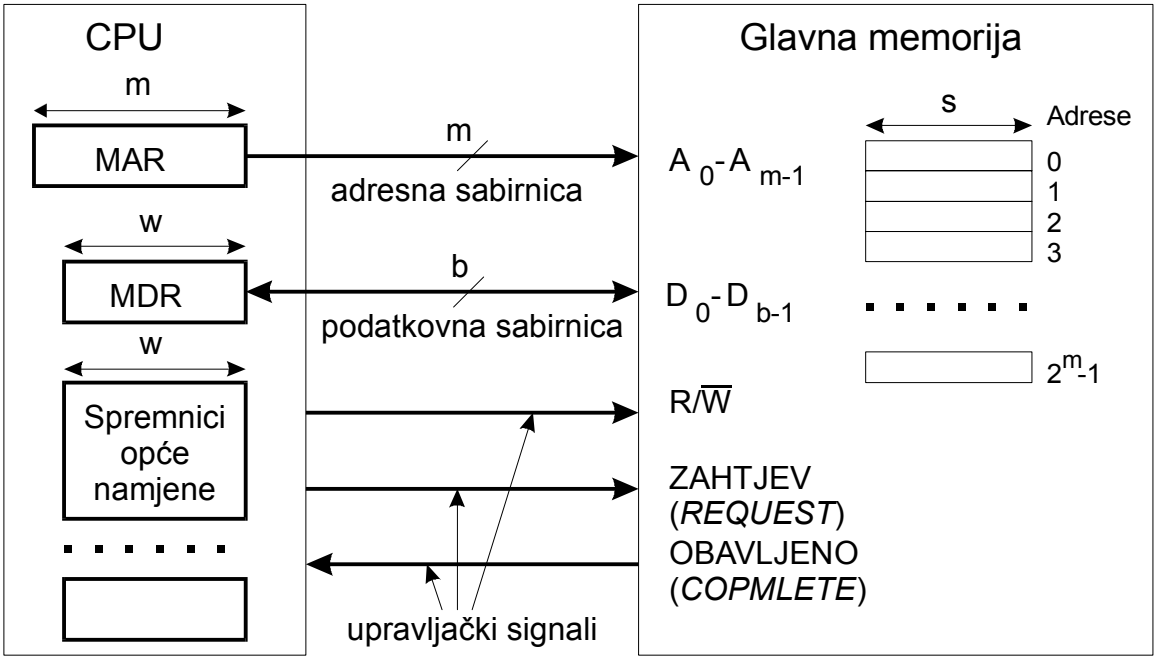
Dosada memorija se tretirala kao monolitni blok s mogućnostima uskladištenja podataka i naredbi, ograničen samo brojem adresnih bita. U memoriju se u jednom taktu upisivao ili iz nje čitao sadržaj cijelog spremnika. U stvarnosti cijena, brzina, potrošnja i mnogi drugi tehnički detalji znatno usložnjavaju ovakav pogled. Ovi detalji predmet su razmatranja ovog poglavlja.

7.1. Uvod: komponente memorijskog sustava

Započeti će se definicijom i analizom parametara koji karakteriziraju memorijski sustav, njegov kapacitet i organizaciju. Slijedi opis svojstava pojedinih dijelova koji su tvore hijerarhijsku strukturu memorije.

7.1.1. Organizacija i veličina glavne memorije

Slika 7.1. i tablica 7.1. prikazuju parametre koji karakteriziraju vezu između procesora i glavne memorije. Glavna memorija može se promatrati kao niz spremnika jednake veličine, *s* bita, koji se nazivaju memorijska riječ (*memory word*). Mora se naglasiti da veličina riječi procesora, *w*, može se razlikovati od veličine memorijske riječi *s*. Općenito, *s* je najmanja riječ kojoj se može pristupati u memoriji. Većina procesora, može pristupati i obrađivati riječi koje su dijelovi riječi procesora, *w*. Tako npr. PowerPC 601 ima 32 bitovne spremnike, memoriji pristupa preko adresa okteta, a može pristupati 8, 16 32 ili 64 bitovnim podacima u glavnoj memoriji. Općenito, procesor koji ima *m* bitovnu adresnu riječ može pristupiti 2^m *s* bitovnim riječima, odnosno može imati kapacitet $2^m \cdot s$ bita.



Slika 7.1. Veza između procesora i memorije.

Simbol	Definicija	Intel 8088	Intel 8086	PowerPC
w	Procesorska riječ	16 bita	16 bita	64 bita
m	Širina adresne sabirnice	20 bita	20 bita	32 bita
s	Broj bita najmanjeg adresabilne podatka	8 bita	8 bita	8 bita
b	Širina podatkovne sabirnice	8 bita	16 bita	64 bita
2^m	Kapacitet memorije u riječima dužine *s*	2^{20} riječi	2^{20} riječi	2^{32} riječi
$2^m \cdot s$	Kapacitet memorije u bitovima	$2^{20} \cdot 8$ bita	$2^{20} \cdot 8$ bita	$2^{32} \cdot 8$ bita

Tablica 7.1. Neka svojstva memorije.

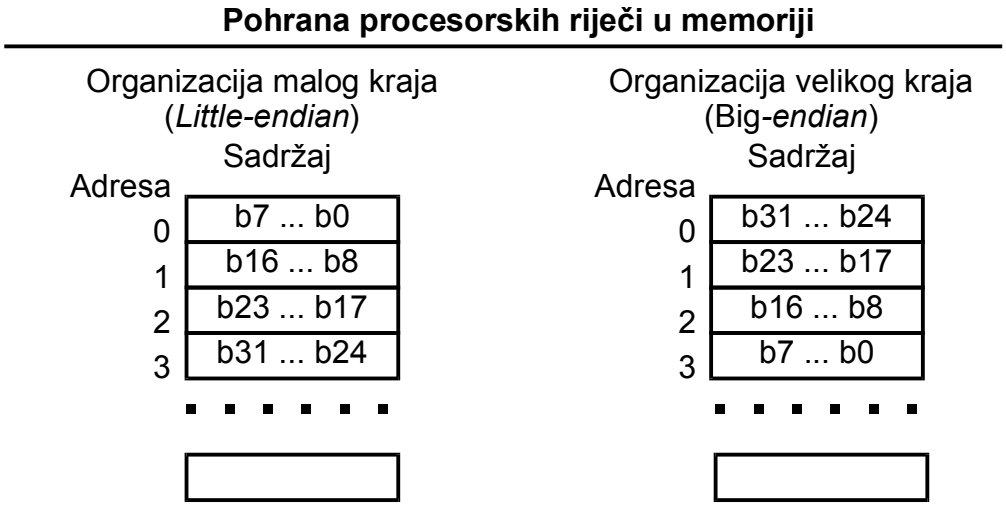
Često, zbog tehničkih razloga, određeni memorijski sustav nije u stanju odjednom prihvatiti ili prenijeti svih *w* bita. Tada se prenose serijski dijelovi procesorske riječi, a procesor ih interno spaja u odgovarajući podatak ili naredbu. Tako se definira najveći broj bita *b* koji se može prenijeti kao jedinstvena cjelina. Drugi slučaj je kada se pristupa najmanje dostupnom podatku u memoriji, veličine *s* bita. Tada je, $s < w$, ali memorijski sustav predaje procesoru svih *w* bita, a sklopovlje procesora izdvaja traženih *s* bita. U oba slučaja proces je transparentan za programera.

Procesor pristupa memoriji postavljajući adresu lokacije na adresnu sabirnicu, te ako je upis u memoriju i podatak na podatkovnu sabirnicu kao i dva upravljačka signala $\mathrm{R}/\overline{\mathrm{W}}$, kao i signal ZAHTJEV (*REQUEST*). Memorijski sustav dojavljuje procesoru da je operacija pristupa obavljena postavljanjem signala OBAVLJENO (*COMPLETE*). U Tablici 7.1. prikazani su podaci za tri procesora: Intel 8088 i 8086 te Motorola PowerPC 601. Procesor Intel 8086 može u jednom memorijskom ciklusu prenijeti cijelu 16 bitovnu riječ preko podatkovne sabirnice, dok procesor Intel 8088 isti podatak prenosi u dva memorijska ciklusa preko 8 bitovne podatkovne sabirnice. Ovo je napravljeno iz ekonomskih razloga s ciljem smanjenja nožica integriranog sklopa, a time i smanjenja cijene sklopa. Ovim je učinjen kompromis između cijene i brzine sklopa.

Organizacija riječi: veliki kraj (*Big Endian*) i mali kraj (*Little Endian*). Ova terminologija organizacije dužih podataka u memoriji ima potječe od Jonathana Swifta koji je u Guliver's Travels opisivao rat između zagovornika da je tvrdo kuhano jaje potrebno razbijati od velikog kraja, *Big Endian*, ili od manjeg kraja, *Little Endian*. Ova terminologija uvriježila se kod opis pohrane podataka dužine *w* u memoriju koja pohranjuje dijelove podatka dužine *s*, gdje je $w > s$. Kod organizacije malog kraja, oktet najmanjeg značaja pohranjuje se na najnižu adresu cijelog podatka, dok organizacija velikog kraja smješta oktet najvećeg značenja na najnižu adresu. Slika 7.2. prikazuje navedene dvije konvencije primijenjene kod procesora PowerPC 601.

Praktički razlike između spomenutih načina pohrane podataka su neznatne. Više je stvar projektanta koji način pohrane će odabrati. Tako npr. Intel 8086 koristi mali kraj, dok PDP11 koristi veliki kraj. Poznavanje načina pohrane podataka značajno je programerima npr. kod korištenja programa za otkrivanje pogrešaka, kada se ispituje sadržaj memorije ili kada se podaci prikazani većim brojem okteta prebacuju s procesora na procesor. Motorola PowerPC 601 procesor ima naredbe koje izvode transformaciju prikaza iz malog u veliki kraj i obratno.

	Riječ procesora			
Bitovi procesorske riječi	b31 ... b24	b23 ... b17	b16 ... b8	b7 ... b0
Adrese okteta kod velikog kraja	0	1	2	3
Adrese okteta kod malog kraja	3	2	1	0



Slika 7.2. Organizacija memorije kod procesora Motorola PowerPC 601; (mali i veliki kraj).

RAM i ROM. RAM je skraćenica od *Random Access Memory*. memorijskim elementima RAM pristupa se proizvoljnim rasporedom i uvijek s istim vremenom pristupa. Prema konvenciji ovaj termin rezerviran je za poluvodičku memoriju u koju se može pisati te iz koje se može čitati. Nasuprot RAMu, ROM je preprogramirana poluvodička memorija iz koje se može samo čitati što opisuje i njen naziv: *Read Only Memory*. Ovi termini su na neki način zbunjujući s obzirom da se ćelijama RAMa i ROMa može pristupati proizvoljnim rasporedom i uvijek s istim vremenom pristupa.

Operacije s memorijom: čitanje (*read*) i pisanje (*write*). Operacija čitanja odnosi se na prijenos podatka iz memorije u procesor ili u neku ulazno/izlaznu jedinicu. Kod ove operacije procesor (ili U/I jedinica) postavlja adresu lokacije čiji sadržaj se očitava na adresnu sabirnicu (sadržaj spremnika MAR), dojavljuje memoriji o kojoj se operaciji radi (čitaj) te može postaviti i neki sinkronizacijski signal npr. ZAHTJEV. Nakon vremena potrebnog da memorija odgovori na zahtjev, koje ovisnosti o brzini kojom memorija može postaviti sadržaj tražene lokacije na podatkovnu sabirnicu (vrijeme odziva), postavlja se sadržaj adresirane lokacije na podatkovnu sabirnicu te memorija dojavljuje procesoru da je podatak spreman aktiviranjem signala ZAVRŠENO. Kod različitih procesora ovaj signal ima različit naziv (Done, Acknowledge, Ready, Accept, $\overline{\text{Wait}}$), ali ima uvijek istu ili sličnu funkciju. Procesor prebacuje podatak s podatkovne sabirnice u međuspremnik MAD. Ovim završava operacija čitanja.

Slično razmatranje vrijedi za operaciju pisanja u memoriju. Postavlja se adresa lokacije u koju se upisuje na adresnu sabirnicu (iz MAR spremnika), podatak koji se upisuje na podatkovnu sabirnicu (iz MAD spremnika) te upravljački signal (signale) piši (write) kao i

ZAHTIJEV. Završetak upisa podatka u memoriju dojavljuje memorijski sustav postavljanjem upravljačkog signala ZAVRŠENO.

7.1.2. Parametri memorijskog sustava

Važan parametar memorijskog sustava je vrijeme pristupa (*access time*), t_a, odnosno vremenski interval od početka memorijske operacije (čitanja ili pisanja) do postavljanja signala od strane memorijskog sustava da je operacija uspješno okončana, kao i vrijeme memorijskog ciklusa (*cycle time*), t_c, odnosno vremenski interval od početka prve do početka sljedeće memorijske operacije. Iako su ova dva podatka dosta povezana, vrijeme ciklusa obično je nešto duže od vremena pristupa.

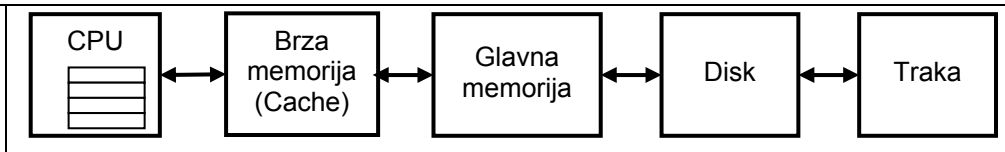
Kod nekih memorijskih sustava ne pristupa se pojedinačnim riječima nego blokovima od k riječi. Kod takvih memorijskih sustava definira se latentnost (*latency*), t_l, odnosno vrijeme potrebno da se pristupi prvoj riječi bloka. Brzina prijenosa (*bandwith*), ω, je brzina kojom se prenose riječi bloka nakon što se pristupilo prvoj riječi bloka. Tako je vrijeme dohvata cijelog bloka $t_{bl} = t_l + k/\omega$. Parametri memorijskih sustava dani su u Tablici 7.1.

Simbol	Naziv	Naziv engl.	Jedinica	Značenje
t_a	Vrijeme pristupa	Access time	s	Vrijeme pristupa riječi
t_c	Vrijeme ciklusa	Cycle time	s	Vrijeme od početka prvog do početka sljedećeg pristupa
k	Veličina bloka	Block size	riječ	Broj riječi po bloku
ω	Brzina prijenosa	Bandwidth	riječ/s	Brzina prijenosa riječi
t_l	Latentnost	Latency	s	Vrijeme potrebno da se pristupi prvoj riječi bloka
$t_{bl} = t_l + k/\omega$.	Vrijeme pristupa bloku	Block access time	s	Vrijeme potrebno da se pristupi cijelom bloku

Tablica 7.2. Parametri memorijskih sustava.

7.1.3. Hijerarhijska organizacija memorije

Obzirom na parametre memorijskog sustava, kao i cijenu vezanu uz pojedina tehnička rješenja, memorija se obično organizira kao hijerarhijski sustav. Tablica 7.3. prikazuje tipičnu hijerarhijsku organizaciju memorije, kao i aproksimacijske vrijednosti kapaciteta, parametara i cijene.



Način pristupa	Direktan pristup	Direktan pristup	Direktan pristup	Direktan pristup	Sekvencijalan pristup
Kapacitet	64 – 1024 B	8 – 256 KB	8 – 64 MB	1 –10 GB	1 TB
Latentnost	1 – 10 ns	20 ns	50 ns	10 ms	10 ms – 10 s
Veličina bloka	1 W	16 W	16 W	4 KB	4 KB
Brzina prijenosa	Sistemski takt	8 MB/s	1 MB/s	1 MB/s	1 MB/s
Cijena/MB	Velika	$500	$30	$0,25	$0,02

Tablica 7.3. Hijerarhijska organizacija memorije, kapacitet, parametri i cijena.

Spremnici opće namjene su prva memorijska razina. Sljedeća je RAM. Današnji računarski sustavi koriste dvije ili više razina RAMa, prva razina je brzi RAM manjeg

kapaciteta ali veće cijene, a druga sporiji RAM većeg kapaciteta. Treća memorijska razina je magnetski disk, a slijedi ga magnetska traka kao sekvencijalni medij za pohranu podataka koji se obično danas koristi za sigurnosnu pohranu podataka (*backup*). Prema konvencijama brza memorija naziva se primarna memorija, diskovi se nazivaju sekundarna, a magnetske trake tercijarna memorija. Za primijetiti je znatne razlike između navedene tri razine u kapacitetu, brzini i cijeni.

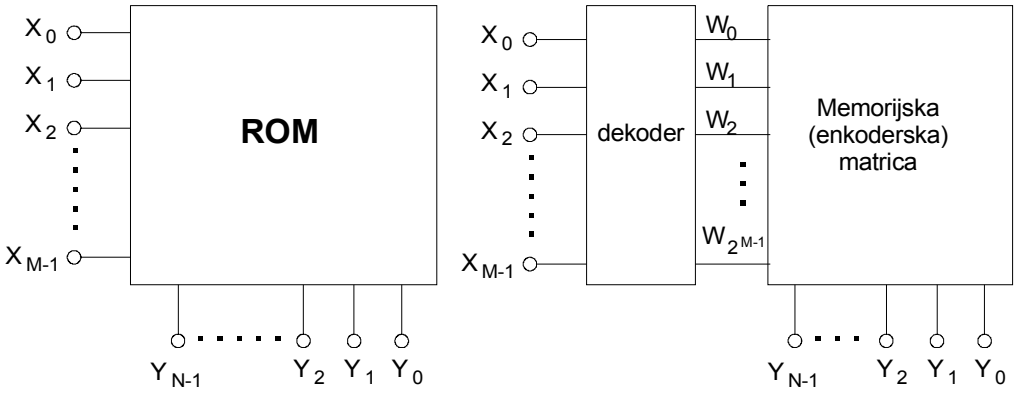
Koncepcijski, svaka memorijska razina mora se povezati sa sljedećom međusklopom sličnim onom na slici 7.1. Razlike postoje u formatu adresa i podataka koji se prenose između različitih razina. Npr. kod pristupa magnetskom disku adresa podatka sastoji se od broja cilindra i broja sektora na cilindru na kojem je pohranjen blok podataka, a prenosi se cijeli blok veličine 0.5 do 4 kokteta. Općenito kod različitih memorijskih razina veličina bloka varira od jednog okteta do nekoliko kokteta. dok vrijeme pristupa varira od reda veličine ns do nekoliko sekunda. Razlika u redu veličina od10^9 puta implicira značenje i složenost projektiranja međusklopova koji povezuju različite memorijske razine.

Tijekom izvođenja programa program i podaci moraju se prenositi između različitih memorijskih razina što je posljedica razlike u njihovim kapacitetima. Prijenos blokova podataka mora biti transparentan programeru. Budući da je vrijeme pristupa brzoj memoriji reda veličine ili jednako brzini rada procesora upravljanje prijenosom blokova između brze memorije i glavne memorije mora se realizirati sklopovski kako bi se ostvarile bolje performanse sustava. S druge strane vrijeme pristupa bloku podataka na disku reda je veličine desetak ms, odnosno desetine ili stotine tisuća procesorskih taktova, pa je dozvoljeno rješenje upravljanja djelomično sklopovski a djelomično programski. Programske rutine sastavni su dio operacijskog sustava.

7.2. Struktura poluvodičke memorije: Pogled projektanta logičkih vrata

/.2.1. Izvedbe memorijskih sklopova

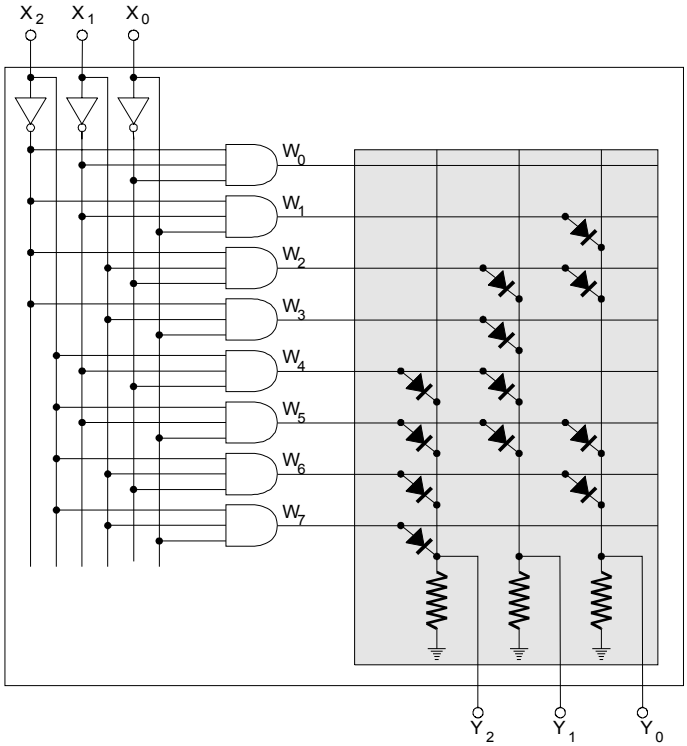
Izvedba poluvodičkih memorija biti će opisana na primjeru ispisne memorije ili ROMa. Prema slici 7.3. ROMse sastoji od dva dijela, dekodera i memorijske (enkoderske) matrice u kojoj su upisane informacije.



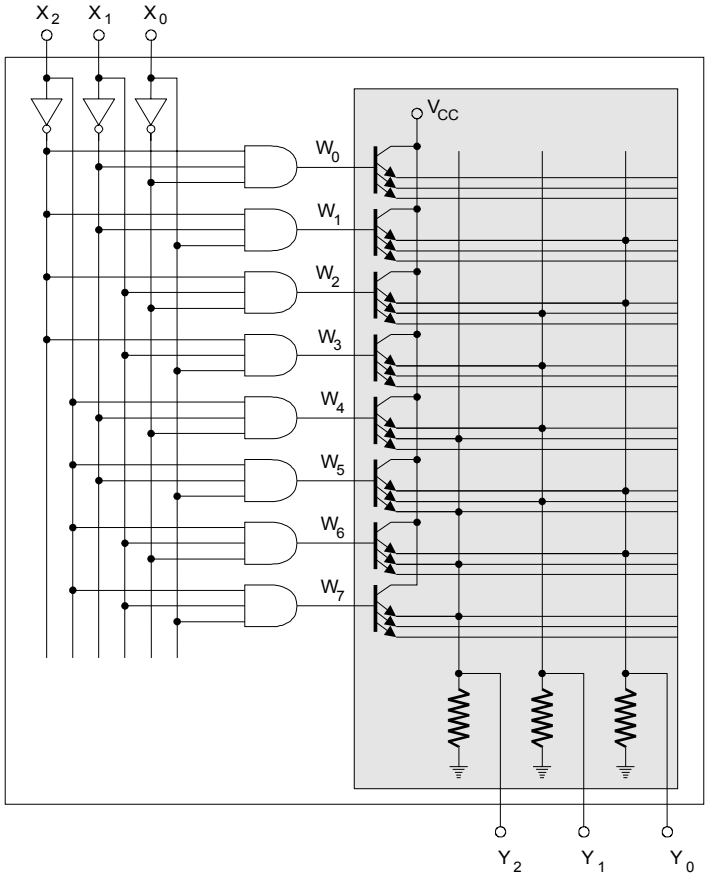
Slika 7.3. Prikaz strukture ROMa.

Dekoder, u ovisnosti o stanju na ulazu X_0 – X_{M-1} odabire jedan od $W_0 - W_{2^M-1}$ redaka memorijske matrice čiji sadržaj se postavlja na izlaze Y_0 – Y_{N-1}. Memorijska matrica može biti realizirana pomoću poluvodičkih dioda, tranzistora ili MOSFETa. Izvedba

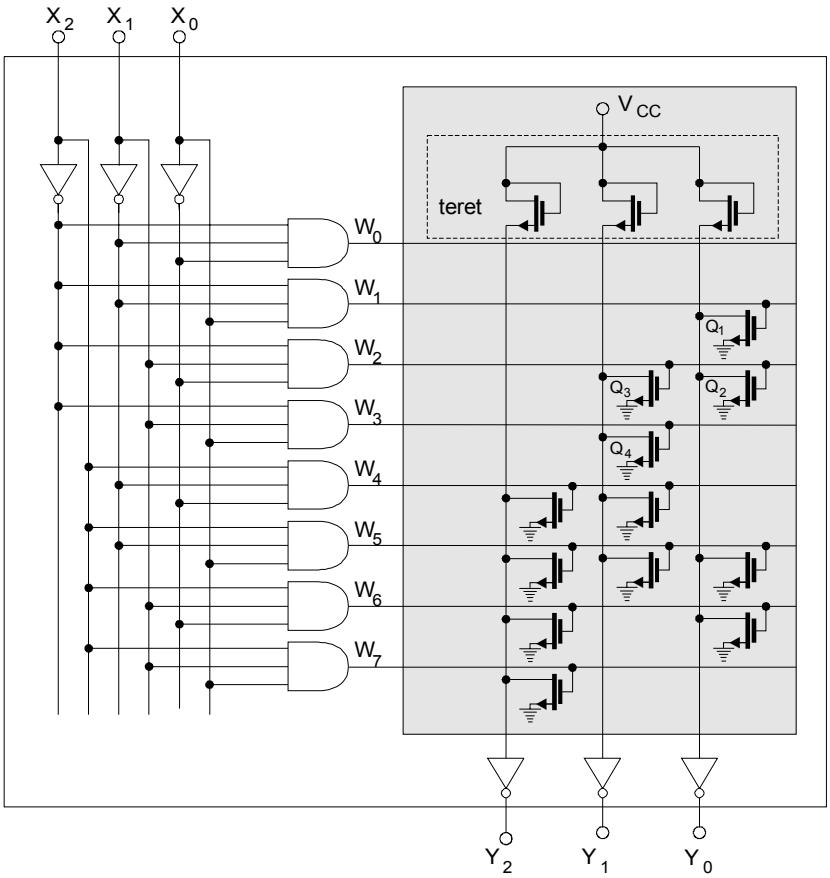
ROMa s osam tro-bitovnim riječima pomoću poluvodičkih dioda prikazana je na slici 7.4., dok je na slici 7.5. prikazana izvedba istog sklopa pomoću bipolarnih tranzistora, a na slici 7.6. prikazana je izvedba pomoću MOSFETa.



Slika 7.4. Izvedba ROMa pomoću poluvodičkih dioda.



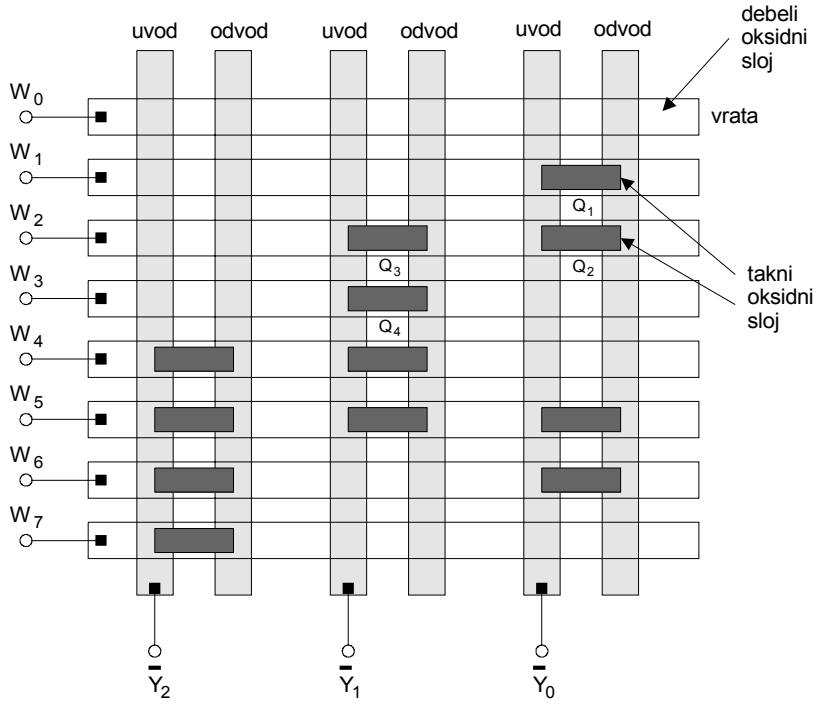
Slika 7.5. Izvedba ROMa pomoću bipolarnih tranzistora.



Slika 7.6. Izvedba ROM-a pomoću unipolarnih tranzistora.

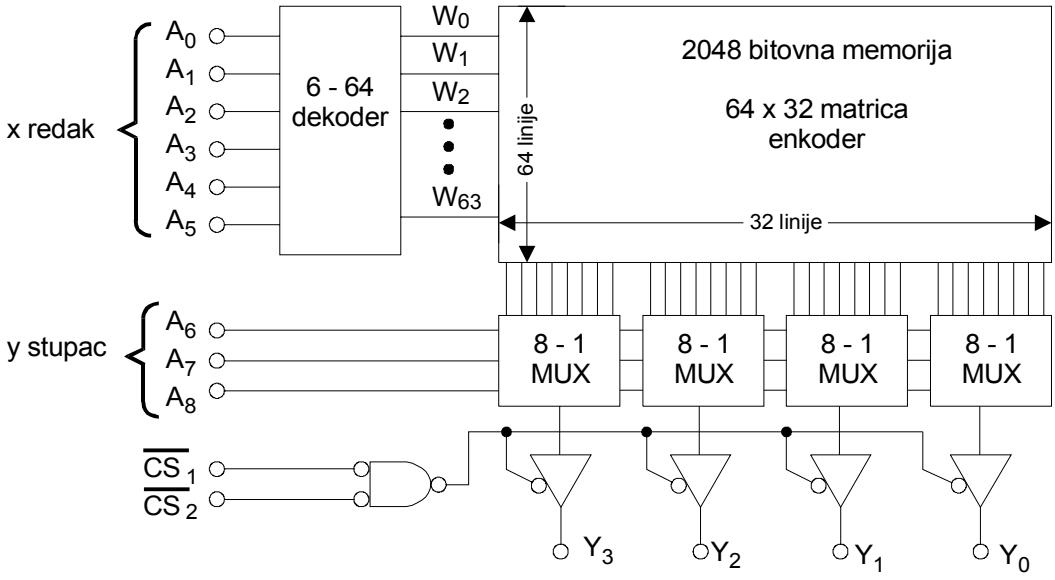
Izvedba ROM-a u NMOS tehnologiji omogućava integraciju znatno većeg broja elemenata po jedinici površine, pa je time moguće realizirati ROM-ove znatno većeg kapaciteta. Prisutnost i položaj svake pojedinačne MOSFET memorijske ćelije određuje se u postupku proizvodnje ROMa, odnosno tijekom generiranja izolacijskog oksidnog sloja između kanala i vrata. Ako je oksidni sloj jako tanak tada je V_T (napon dodira) nizak, odnosno za debeli oksidni sloj V_T je jako veliki. S pozitivnim naponom na vratima provede MOSFET koji ima tanak oksidni sloj spajajući ulaz izlaznog invertora na logičku 0. MOSFET s debelim oksidnim slojem ne može reagirati na napon na vratima i ostaje u stanju ne vođenja. Ukoliko je napon na vratima MOSFETa nizak, u logičkoj nuli MOSFET ne vodi pa je ulaz izlaznog invertora preko tereta (MOSFET tranzistora) spojen na napon napajanja odnosno u logičkoj 1. Memorijska matrica realizira se spajanjem MOSFETa u vezani ILI spoj (*wired OR*). Na slici 7.7. prikazana je izvedba memorijske matrice u integriranoj tehnici.

Kod realizacije ROMova većeg kapaciteta, odnosno ROMova s više ulaznih i više izlaznih linija prisutan je problem relativno velikog sklopa dekodera. Tako npr. za ROM s 9 ulaznih linija i četiri izlazne linije, odnosno 2^9 x 4 bita tj. 512 x 4 bita ROM, potrebno je 512 I vrata s devet ulaza za realizaciju dekodera i 512 tranzistora s četiri emitera za realizaciju enkodera ili memorijske matrice.



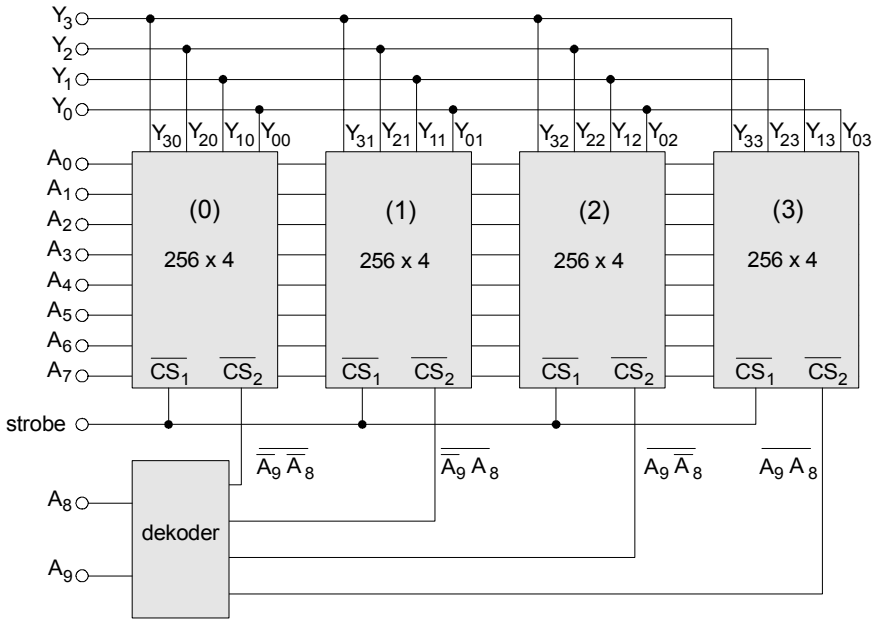
Slika 7.7. Izvedba memorijske matrice pomoću unipolarnih tranzistora.

Na slici 7.8. prikazano je znatno ekonomičnije rješenje. 6 bitova ulaznog podatka (adrese) (x redak) generira 64 izlaza iz dekodera. Ako se enkoder konfigurira da na izlazu ima 32 podatka tada se dobiva 64 x 32 = 2048 bita kao što je i kapacitet predhodno opisanog ROMa. Kako je potrebno imati samo 4 izlazna podatka 4 x 8 izlaznih linija vode se na četiri 8–1 multipleksora. Adresa multipleksora odabire se s tri ulazne (adresne) linije (y stupac). Ovakva izvedba ROMa naziva se xy adresiranje ili dvodimenzionalno adresiranje. Za izvedbo ovakvog sklopa potrebno je 64 I vrata za 6-64 dekoder, po 9 I vrata za 8-1 dekoder što sve ukupno čini 64 + 36 = 100 I vrata. Ovim se postigla značajna ušteda na broju I vrata. Za izvedbu memorijske matrice, enkodera, potrebno je 64 tranzistora s 32 emitera.



Slika 7.8. Dvodimenzionalna izvedba dekodera ROMa.

Izlazi se vode preko odvojnih sklopova (*buffer*) s tri izlazna stanja, a signali $\overline{CS_1}$ i $\overline{CS_2}$ koriste se za odabir sklopa. Naime ako je bilo koji od ovih ulaza u logičkoj 1 tada su izlazi ROMa onemogućeni, odnosno u stanju su visoke impendancije. Tada je ROM neaktivan. Da bi se izlazi mogli proslijediti potrebno je da se oba ova ulaza u logičkoj 0 i tada je ROM aktivan, odnosno kaže se da je taj integrirani sklop odabran ili selektiran. Navedeni ulazi mogu se koristiti kako bi se više ovakvih sklopova kaskadno spajalo kako bi se postigao veći kapacitet memorije, slika 7.9.



Slika 7.9. Kaskadno povezivanje više ROM-ova.

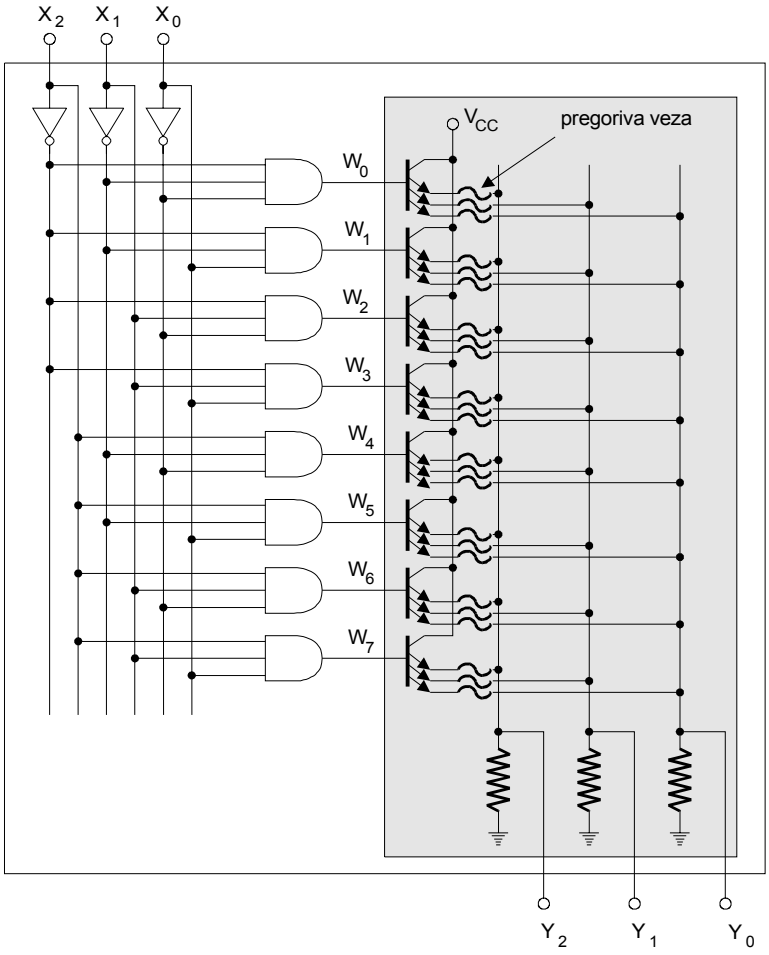
Sklop radi na sljedeći način. Adrese A_0 do A_7 dovode se paralelno na sve ROM-ove čiji izlazi su paralelno vezani na izlaze Y_0 do Y_3. Posljednje dvije adresne linije A_8 i A_9 dovode se na dekoder koji pretvara dvije ulazne u četiri izlazne linije. Svaka izlazna linija povezana je na jedan od ulaza za odabir $\overline{CS_2}$ .Tako npr. ako je A_9 = 0 i A_8 = 1 odabran je ROM (1) je njegov $\overline{CS_2}$ = 0 dok je ostalima $\overline{CS_2}$ = 1. Ovim su, uz pretpostavku da je strobe = 0 izlazi ROMova (0), (2) i (3) u stanju visoke impendancije, a stanja na izlazima jednaka su stanjima na izlazima ROMa (1), tj. $y_0 = y_{01}$, $y_1 = y_{11}$, $y_2 = y_{21}$ i $y_3 = y_{31}$.

Programabilna ispisna memorija (*Programable ROM*) (PROM)

Ispisna memorija programira se tvornički tako da je izrada ovakvih sklopova za manje serije ekonomski neisplativa, a također veći proizvođači teško će se prihvatiti izrade male serije ROM-ova. Naime, za svaku seriju ROM-ova potrebna je izrada maski ili filmova koja se može isplatiti tek kroz veću seriju proizvoda. Također i vrijeme isporuke korisničkih ROM-ova je relativno dugo. Ovi problemi riješeni su sklopovima nazvanim PROM (***P**rogramable **ROM***) koji imaju iste karakteristike kao i ROM, ali ih korisnik sam može programirati. Oni su posebno prikladni za manje serije, a naročito u fazi projektiranja i ispitivanja digitalnih sustava. Kasnije ukoliko se pokaže ekonomski opravdano ovi sklopovi mogu biti jednostavno zamijenjeni tvornički programiranim ROM-ovima.

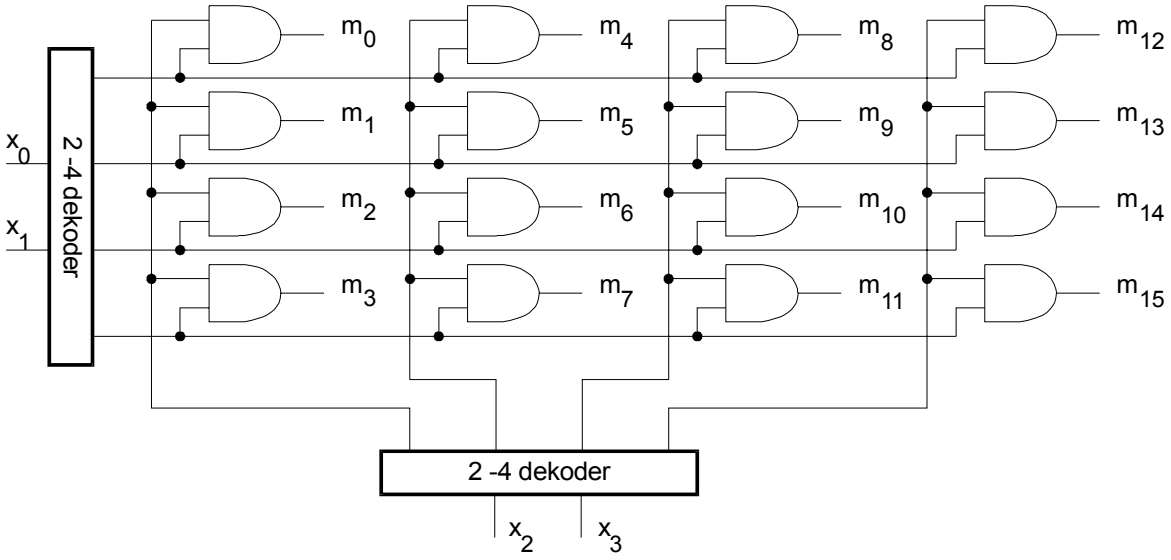
PROM sadrži enkodersku matricu u kojoj su napravljeni svi mogući spojevi. Npr. 32 x 8 bitovni PROM sastoji se od 32 tranzistora s osam emitera (E_0, E_1, ... , E_7). Emiteri E_0 svih tranzistora, njih 32, spojeno je na izlaznu liniju Y_0, emiteri E_1 na Y_1 itd. Spoj svakog emitera na izlaznu liniju ostvaren je preko tankog poluvodičkog spoja koji djeluje kao osigurač. Kada se kroz emiter propusti struja veća od dozvoljene, ovaj spoj pregori i odspaja taj emiter od izlazne linije. Koristeći poseban uređaj za programiranje (PROM programator) korisnik proizvoljno pregara (*burn, blow*) pojedine spojeve kako bi ostvario željeno funkcionalno djelovanje ovog sklopa. Za napomenuti je da jednom programirani PROM nije moguće više preprogramirati.

Logički PROM se može promatrati kao fiksna dekoderska matrica s I logičkim sklopovima i programabilna enkoderska matrica spojena na ILI vrata, slika 7.10.



Slika 7.10. Izvedba programabilne memorije, PROM-a.

Matrični dekoderi. Dekoderi kod većih memorijskih elemenata iz praktičkih razloga nešto su različiti od onih prikazanih na slikama 7.4 –7.6. te slici 7.10. Razlog leži u činjenici da je neprikladno realizirati I vrata s većim brojem ulaza pa se obično koriste tzv. matrični dekoderi, slika 7.11., dok se oni prethodni nazivaju jednodimenzionalni dekoderi. Matrični dekoderi implementiraju se pomoću dvoulaznih I vrata. Oni se nadalje mogu kaskadno proširivati kako bi se realizirali veći dekoderi. Naravno ovakva realizacija rezultira u dužem vremenu odziva sklopa jer se signal mora propagirati kroz veći broj logičkih vrata.



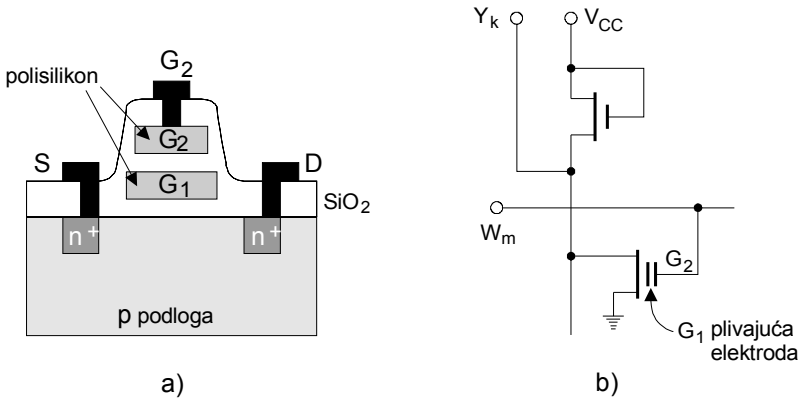
Slika 7.11 . Matrična izvedba dekodera.

Izbrisivi PROM (*Erasable PROM*) EPROM

Danas se proizvode dvije vrste PROMova kod kojih se upisane veze mogu izbrisati a to su: EPROM i električki izbrisiv (*Electrically Erasable*) PROM ili E^2PROM.

EPROM

Programabilne memorije nije bilo moguće preprogramirati zato što su fizički uništene veze emitera tranzistora s izlazima. Izbrisivi PROM zasnovan je MOSFET strukturi prikazanoj na slici 7.12.:



Slika 7.12. a) Izvedba EPROM-a; b) prikaz memorijske ćelije.

Izbrisiva PROM memorija zasnovana je na posebnoj MOS strukturi prikazanoj na slici 7.12.a). Ovaj MOSFET s dvostrukim vratima naziva se FAMOS (***F**loating-gate **A**valanche-injection **M**etal **O**xide **S**emiconductor*). Uvod G_1 je izrađen od polisilikona. Potpuno je okružen izolacijskim oksidnim materijalom SiO_2 i nema omskog kontakta prema vani pa se zato i naziva plivajuća vrata. Kada se između vrata G_2 i uvoda narine relativno veliki napon, npr. 25V, jako električno polje uzrokuje lavinski proboj kroz pn spoj, a posljedica je relativno velika struja kroz komponentu. Elektroni koji dobiju relativno veliku energiju, ubrzani električnim poljem, prodiru u tanki oksidni sloj i skupljaju se oko vrata G_1. Kako je ova elektroda okružena izolacijskim materijalom ne postoji mogućnost pražnjenja nagomilanog naboja tako da nestankom vanjskog napona

ova potencijal vrata G_1 je negativan u odnosu na uvod S. Ovaj negativni potencijal onemogućava stvaranje kanala između uvoda i odvoda čak i kada se na vrata G_2 narine pozitivan napon od 5V. Rezultat opisanog postupka je zapis logičke 1 u ovoj memorijskoj ćeliji. SiO_2 je jako dobar izolacijski materijal tako da akumulirani naboj na vratima G_1 ostaje dugo uskladišten. Ispitivanja su pokazala da 70% uskladištenog naboja ostaje i nakon 10 godina pri temperaturi skladištena od 125 ^{0}C.

Ukoliko se memorijska ćelija izloži ultra-ljubičastom zračenju, SiO_2 postaje djelomično vodljiv i fotoelektrična struja odvede nagomilani naboj s plivajućih vrata G_1. Ovim postupkom briše se sadržaj memorije. Postupak brisanja traje približno 10 minuta.

E^{2}PROM

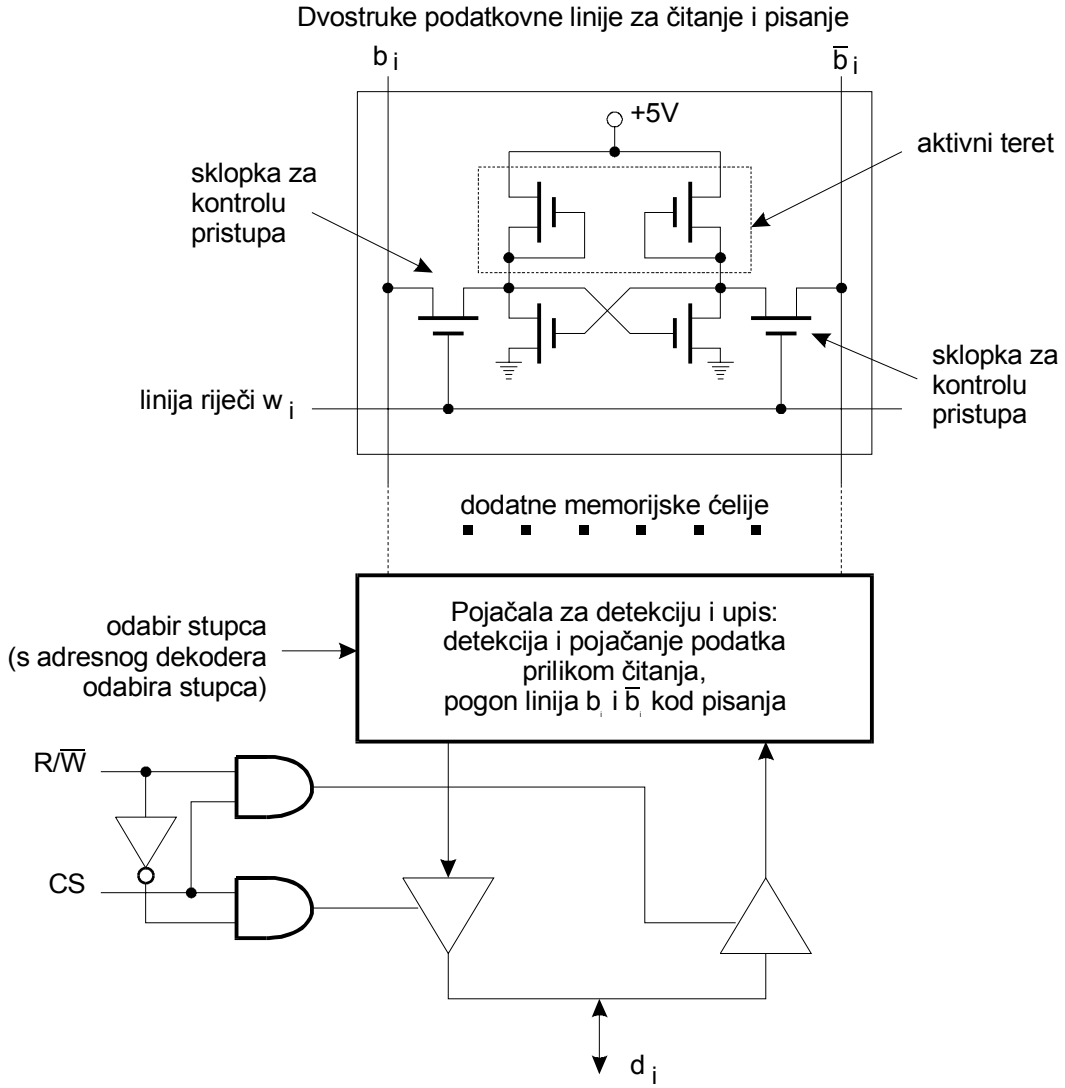
Nedostatak EPROMa je postupak brisanja koji zahtjeva relativno dugo izlaganje komponente UV zračenju. Kada je EPROM sastavni dio digitalnog sklopa, neprikladan je postupak vađenja komponente, njeno stavljanje u uređaj za brisanje, te ponovo programiranje pomoću EPROM programatora. Tako je primjena EPROMa ograničena u aplikacijama koje zahtijevaju brzo i učestalo preprogramiranje memorije. Ovaj nedostatak riješen je električki izbrisivim PROM. Struktura ove memorije slična je strukturi EPROMa. s time da je debljina izolacijskog sloja između vrata G_1 i kanala znatno smanjena (iznosi svega oko 100μm) . Napon od oko 10V, koji je opet viši od normalnog radnog napona od 5V, narinut između vrata i uvoda uvjetuje da elektroni efektom tuneliranje prođu uzak oksidni sloj i nagomilaju se na vratima G_1. Nagomilani naboj u normalnom radu, bez obzira da li je na između vrata G_2 i uvoda naponska razlika od 5V, onemogućava stvaranje vodljivog kanala između uvoda i odvoda te MOSFET nikada ne vodi. U tu memorijsku ćeliju upisana je stalno logička 1. Brisanje sadržaja memorijske ćelije postiže se povećanim negativnim naponom (oko –10V) između vrata G_2 i uvoda.

Statički RAM, SRAM

Slika 7.13. prikazuje memorijsku ćeliju statičkog RAMa. Za razliku od ROMa ćelija RAMa realizirana je sa šest tranzistora spojenih u bistabil. Dva tranzistora tvore aktivni teret, dva tranzistora su u spoju bistabila te dva tranzistora služe kao sklopke koje spajaju ulaz/izlaz memorijske ćelije na zajedničku liniju.

Princip rada RAM memorijske ćelije može se opisati na sljedeći način:

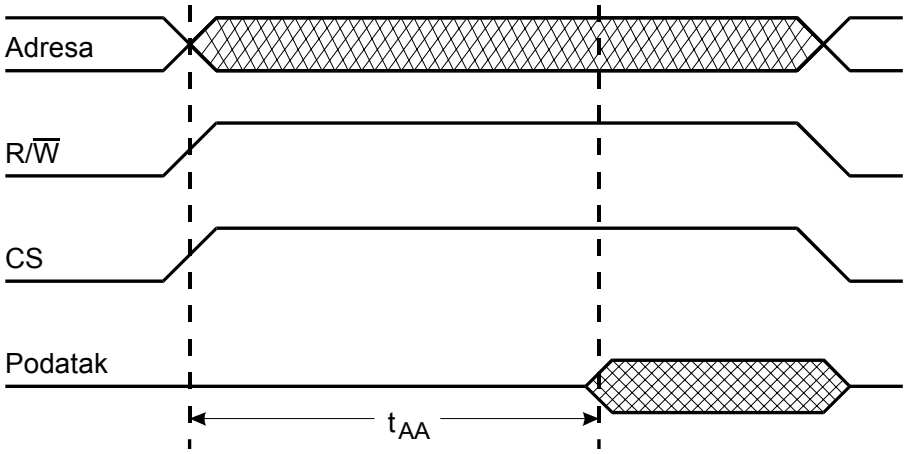
- Memorijska ćelija realizirana je kao direktno vezani bistabil.
- Tranzistori za pristup ćeliji omogućavaju operacije pisanja i čitanja. Ćelija se odabire preko linije za odabir riječi (redak matrice) i u ćeliju je moguće pisati ili iz nje čitati pomoću dva tranzistora.
- Stupac memorijske matrice povezan je dvjema linijama (*dual rail*) na koje se postavlja stvarna b_i i invertirana $\overline{b}_i$ vrijednost stupca.
- Podatak se upisuje u ćeliju na način da se vrijednost b_i i $\overline{b}_i$ narine na linije preko logike za pisanje u ćeliju i odabere se riječ u koju se upisuje.
- Podatak se čita iz ćelije na način da se linije predpolariziraju na napon između logičke nule i jedinice, a zatim se odabere redak i kojeg se čita. Ovaj podatak preko logike za detekciju i pojačanje vodi se na izlaz sklopa.
- Logika na dnu slike 7.13. upravlja operacijama čitanja i pisanja.



Slika 7.13. RAM memorijska ćelija realizirana pomoću šest tranzistora.

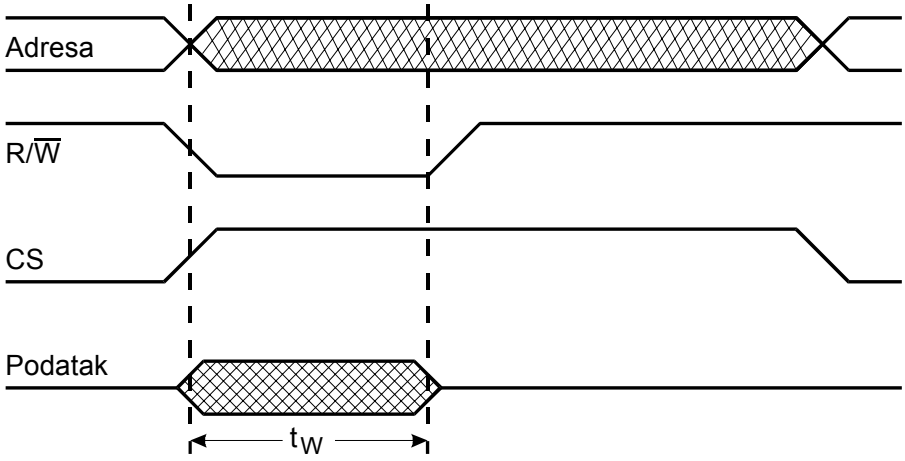
7.2.2. Vremenski dijagram statičkog RAM-a

Slijedi analiza vremenskog odziva statičkog RAM-a. Promatrati će se vrijeme potrebno da se očita sadržaj niza ćelija, odnosno podatak. Pretpostavka je da procesor istovremeno postavlja na adresnu sabirnicu adresu podatka kao i upravljački signal za čitanje, READ, te da se istovremeno preko adresne logike sustava postavlja i signal CS. Vrijeme pristupa podatku uključuje vrijeme potrebno da se adresa i upravljački signal propagiraju kroz dekodersku i upravljačku logiku, te vrijeme potrebno da se sadržaj ćelije postavi na internu sabirnicu te da se isti podatak propagira kroz izlazni multipleksor (kod 2D realizacije adresnog dekodera slika 7.8.) na izlaze sklopa $Y_0 – Y_3$. Slika 7.14 prikazuje vremenski dijagram statičkog RAM-a za operaciju čitanja.



Slika 7.14. Vremenski dijagram čitanja iz statičkog RAM-a.

Vremenski dijagram upisa u RAM sličan je vremenskom dijagramu čitanja iz RAM-a uz razliku da procesor postavlja podatke na podatkovnu sabirnicu istovremeno s adresom i upravljačkim signalom, te da dekoder za odabir integriranog sklopa postavlja istovremeno i CS signal. Slika 7.15. prikazuje vremenski dijagram upisa u RAM.



Slika 7.15. Vremenski dijagram pisanja u statički RAM-a.

7.2.3. Dinamički RAM

Umjesto da se informacija pohranjuje u bistabil moguće je kao memorijski element koristiti kondenzator, slika 7.16.

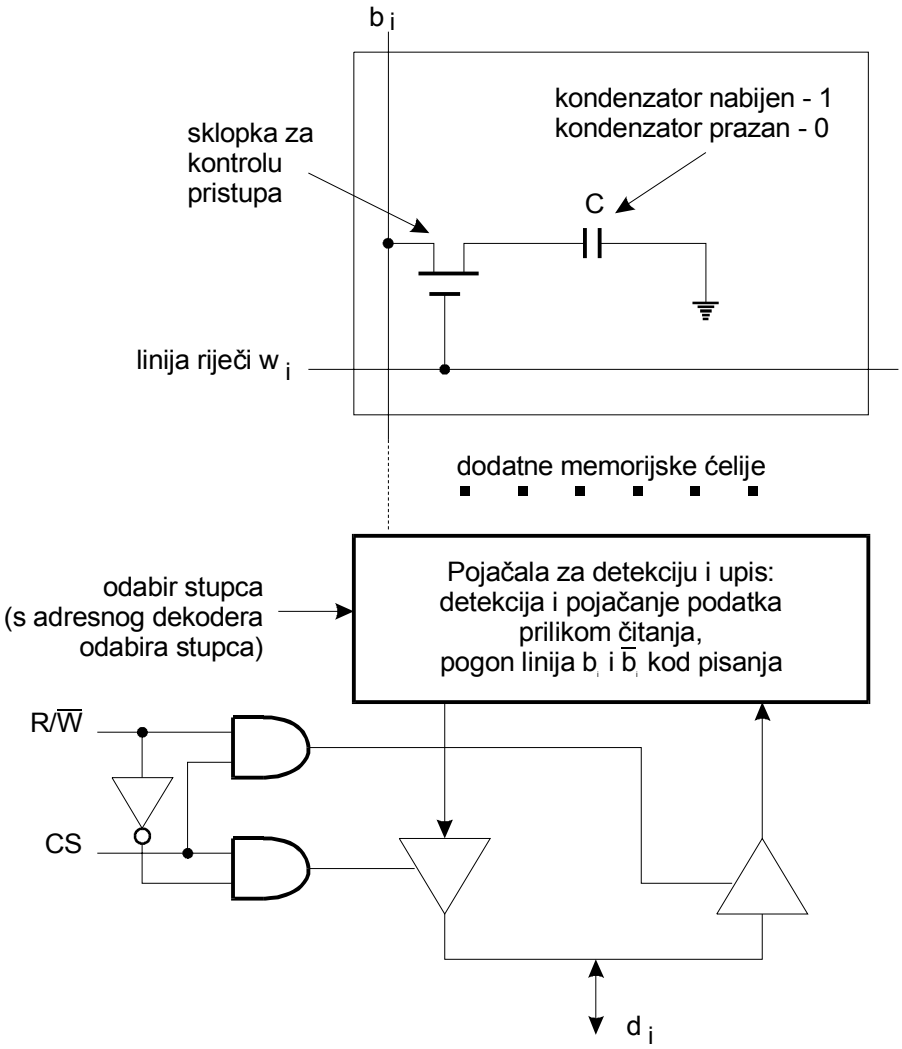
Za primijetiti je da ovom realizacijom jedan tranzistor i kondenzator zamjenjuju šest tranzistora. Ukoliko se kondenzator realizira pomoću tranzistora slijedi zaključak da je potrebno tri puta manje komponenata za realizaciju ovakve memorije. Ovakva memorija radi na sljedeći način:

- Za upis podatka u memorijsku ćeliju, podatak se postavlja na b_i liniju i odabire se riječ u koju se podatak upisuje. Kondenzator se nabija ili izbija u ovisnosti o vrijednosti bita koji se upisuje.
- Čitanje podataka ostvaruje se na sličan način kao i čitanje kod statičkog RAMa. Linija za podatak se predpolariziraja na napon između logičke nule i jedinice, a zatim se odabere redak i kojeg se čita. Podatak iz kondenzatora postavlja se na liniju podatka te se preko logike za detekciju i pojačanje vodi na izlaz sklopa. U ovom

procesu ukoliko je nabijen kondenzator se isprazni. Zato je potrebno osvježiti njegovo stanje.

- Sklop za detekciju i pojačanje postavlja ponovo podatak na liniju podatka i upisuje podatak u ćeliju, odnosno osvježava naboj kondenzatora. Ovaj proces naziva se ciklus osvježavanja (*refresh cycles*).

Zbog potrebe osvježavanja sadržaja memorijskih ćelija ovakva memorija naziva se dinamički RAM ili DRAM.



Slika 7.16. Memorijska ćelija realizirana pomoću kondenzatora.

Dodatno vrijeme potrebno za osvježavanje. Ako se duže vrijeme ne čita iz kondenzatora, on se preko parazitnih otpora isprazni. Zato je potrebno čitati sadržaje svih memorijskih lokacija u određenim vremenskim intervalima kako bi se sačuvala zapisana informacija. Ovo donekle komplicira samu izvedbu memorijskog sklopa. Ipak sama realizacija olakšava sklopovsko rješenje, budući da se cijeli redak memorijske matrice može osvježiti u jednom ciklusu. Na primjer za 4 x 1Mbit DRAM, koji je realiziran kao 2048 x 2048 matrica, potrebno je izvesti 2048 osvježavanja svakih 4 ms. Ukoliko je vrijeme osvježavanja 80 ns, vrijeme potrošeno na osvježavanje iznosi:

$$t_{refresh}\% = \frac{2048 \times 80}{4 \times 10^6} \times 100 = 4.1\%,$$

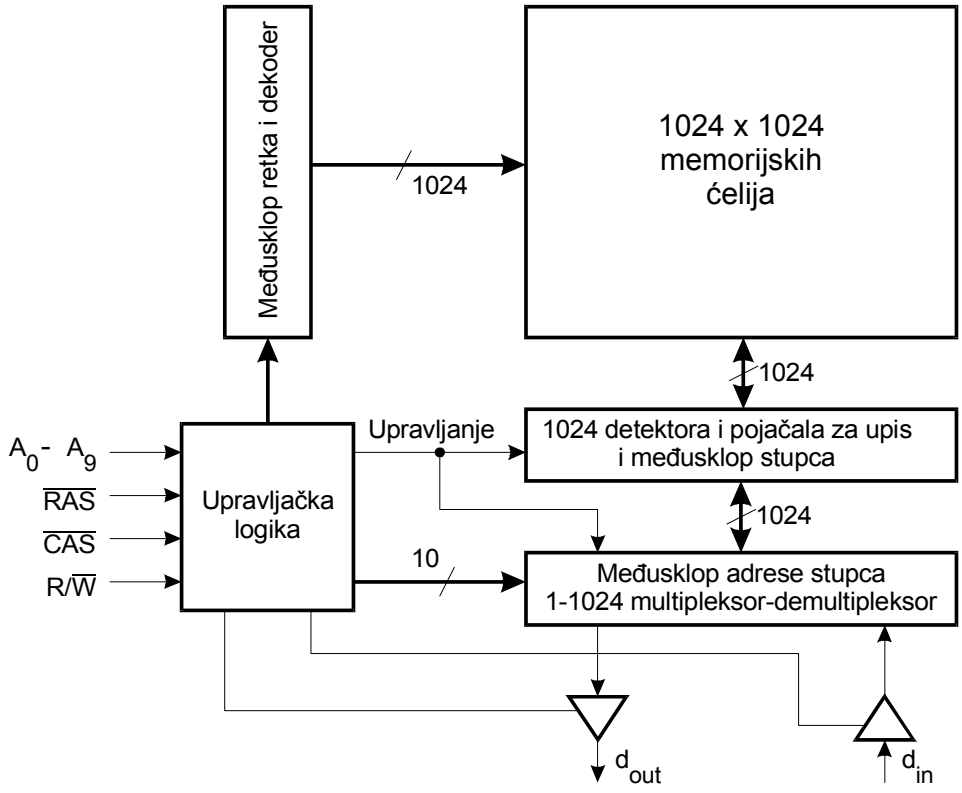
što je sasvim prihvatljivo.

Organizacija DRAM-a

Više faktora utječu na projektiranje i organizaciju DRAM-a. Najvažniji čimbenici su broj nožica integriranog sklopa i mehanizam osvježavanja. Ukoliko se pretpostavi da se troškovi istraživanja i razvoja (R&D) proizvoda relativno brzo isplate, troškovi pakovanja su praktički veći od cijena silicija. Nadalje troškovi pakovanja direktno su vezani uz broj nožica integriranog sklopa. Tako npr. za 16M x 1bit DRAM potrebno je 24 adresne, 1 ili 2 podatkovne, 2 upravljačke i 2 za napajanje nožice, sveukupno 30 nožica.

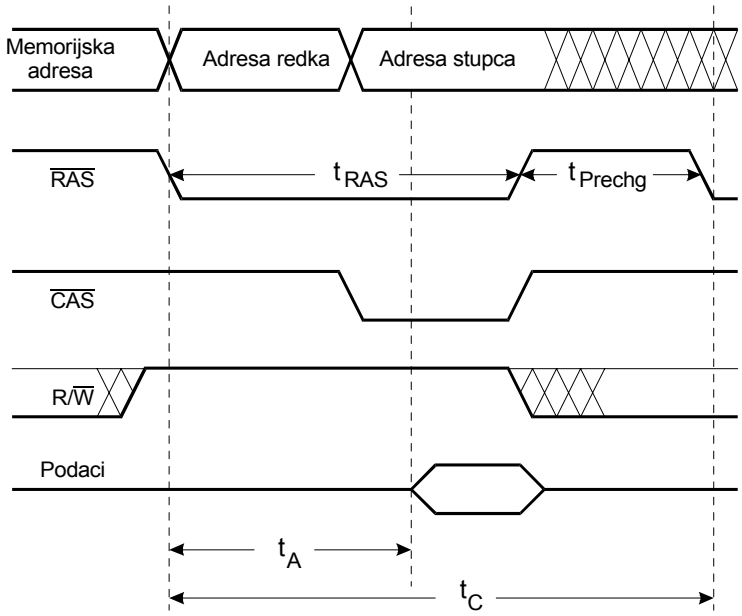
Jedan od načina za smanjenje broja nožica je prijenos adrese retka matrice i adrese stupca matrice preko istih nožica jedne iza druge. Ovo vremensko multipleksiranje dviju adresa smanjuje broj adresnih nožica na pola, ali zahtjeva dodatne dvije upravljačke nožice, jednu za dojavu da je adresa retka važeća, RAS (*Row Address Strobe*), i druga za dojavu da je adresa stupca važeća, CAS (*Column Address Strobe*). Ipak moguće je uštedjeti na jednoj upravljačkoj nožici ukoliko se ista nožica koristi za CAS i CS. U ovom slučaju moguće je čitav redak osvježiti postavljanjem RAS signala bez postavljanja CAS signala. Ovi signali su obično aktivni u logičkoj nuli pa se označavaju: $\overline{\mathrm{RAS}}, \overline{\mathrm{CAS}}, \overline{\mathrm{CS}}$.

Slika 7.17. prikazuje realizaciju 1M x 1bit DRAM-a. Ovaj sklop ima 10 adresnih linija s kojih se sadržaj jednom pohranjuje u međuspremnik retka i koristi za odabir retka, a drugi put u međuspremnik stupca i koristi za odabir stupca. Upravljački signali $\overline{\mathrm{RAS}}$ i $\overline{\mathrm{CAS}}$ upisuju sadržaj sa adresnih ulaza u odgovarajući međuspremnik. Upravljačkim signalom $\mathrm{R}/\overline{\mathrm{W}}$ upravlja se smjerom podatka.



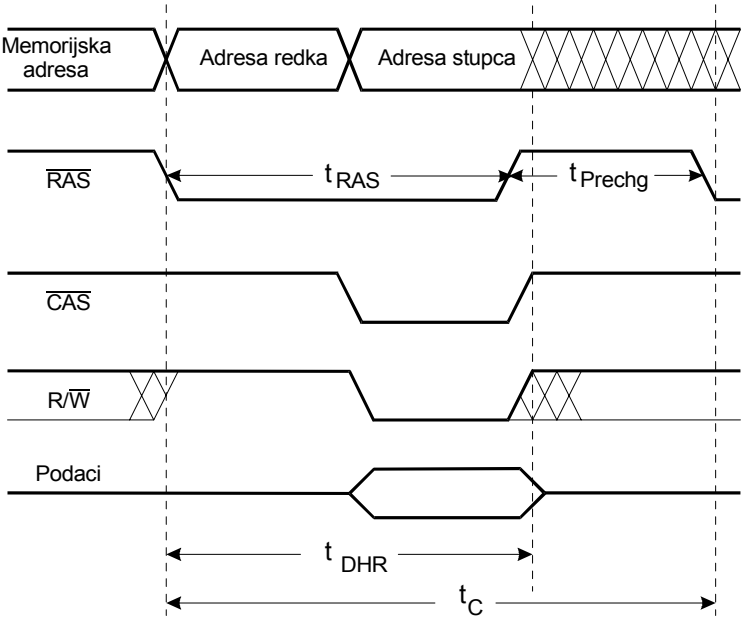
Slika 7.17. Organizacija DRAM-a.

Slika 7.18. prikazuje vremenski dijagram čitanja podatka iz DRAM-a. Trajanje ciklusa čitanja, t_C, sastoji se od minimalnog trajanja signala $\overline{RAS}$, t_{RAS}, vremena potrebnog da sklopovi osvježe stanje na unutarnjim podatkovnim linijama, t_{Prechg}, odnosno vremena u kojem RAS signal mora biti neaktivan. Podatak je dostupan nakon vremena pristupa t_A. Aktiviranjem signala $\overline{CAS}$ podatak se postavlja na izlaz sklopa, a ciklus čitanja praktički završava deaktiviranjem signala RAS i CAS.



Slika 7.18. Vremenski dijagram čitanja iz DRAM-a.

Sličan vremenski dijagram vrijedi za operaciju upisa podatka u DRAM, a prikazan je na slici 7.19. Ovdje je potrebno naglasiti značenje vremenskog intervala t_{DHR} za vrijeme kojeg je potrebno zadržati podatke aktivne na ulazu DRAM-a kako bi oni bili ispravno upisani.



Slika 7.19. Vremenski dijagram upisa u DRAM-a.

DRAM integrirani sklopovi imaju mogućnost osvježavanja aktiviranjem samo signala $\overline{\text{RAS}}$ (*RAS-only refresh*). U ovom slučaju postavlja se samo adresa retka na adresnu sabirnicu te se sadržaj cijelog retka osvježi aktiviranjem $\overline{\text{RAS}}$ signala ne aktivirajući signal CAS. Odsustvo signala CAS signalizira DRAM-u da se radi o ciklusu osvježavanja. U ovom slučaju podaci se ne postavljaju na izlaz sklopa.

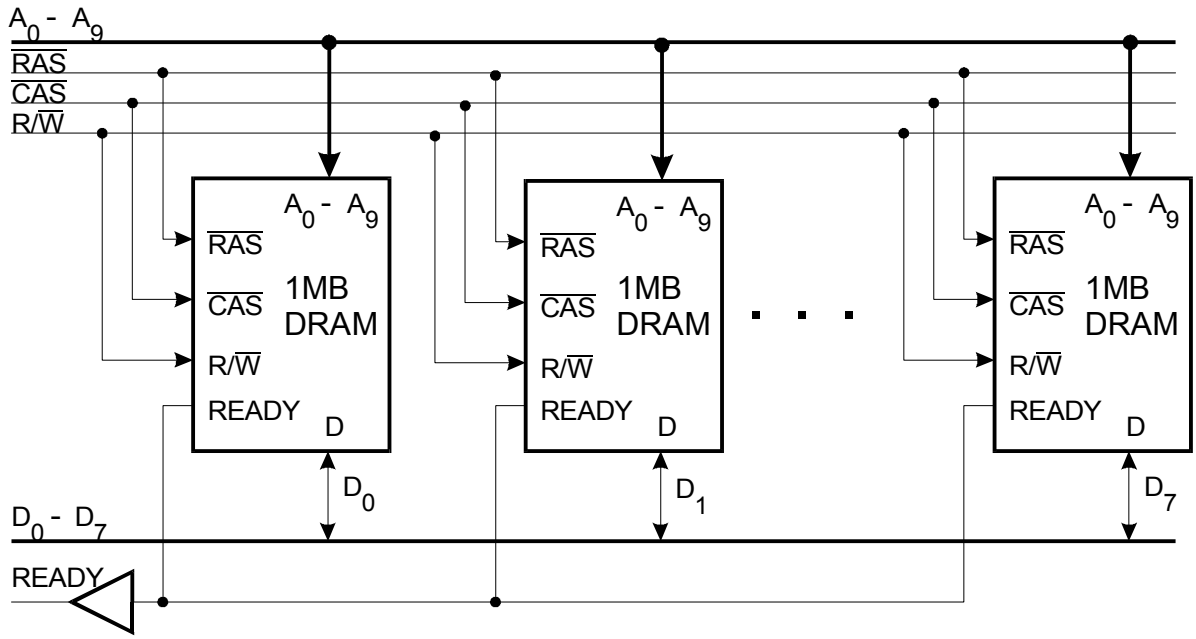
Mnogi DRAM sklopovi koriste princip CAS prije RAS (*CAS before RAS*). Sklop sadrži r bitovno brojilo, gdje je r broj adresnih bitova redaka. U slučaju kada sklop detektira aktiviranje CAS signala prije RAS signala započinje se ciklus osvježavanja sadržaja retka na koje pokazuje unutarnje brojilo. Po završetku ciklusa brojilo se inkrementira za jedan. U nadležnosti posebnog vanjskog sklopa je upravljanje ciklusima osvježavanja. Većina proizvođača DRAM-ova integriraju upravljački sklop zajedno s memorijom.

/.3. Memorijski moduli

Veliki memorijski moduli su znatno više od samih memorijskih integriranih sklopova. Oni obuhvaćaju adresne dekodere, multipleksore i međusklopove. integrirani sklopovi organiziraju se na način da se povećava broj bita po riječi i ukupni kapacitet memorije. Računarski sustav može imati više memorijskih modula spojenih na način ili da povećaju sveukupni memorijski kapacitet ili da ubrzaju vrijeme pristupa.

Memorijski moduli koji koriste SRAM integrirane sklopove relativno se jednostavno povezuju bez značajnih promjena u vremenskim dijagramima. Kod modula koji koriste DRAM integrirane sklopove, sama složenost upravljanja sklopom značajno usložnjava vremenske dijagrame ovakvih memorijskih modula. Razlozi su u vremenski multipleksiranim adresama redaka i stupaca kao i u ciklusima osvježavanja.

Na slici 7.20. prikazan je memorijski modul kapaciteta 1 Moktet sastavljen od 8 DRAM integriranih sklopova kapaciteta 1 Mbit.

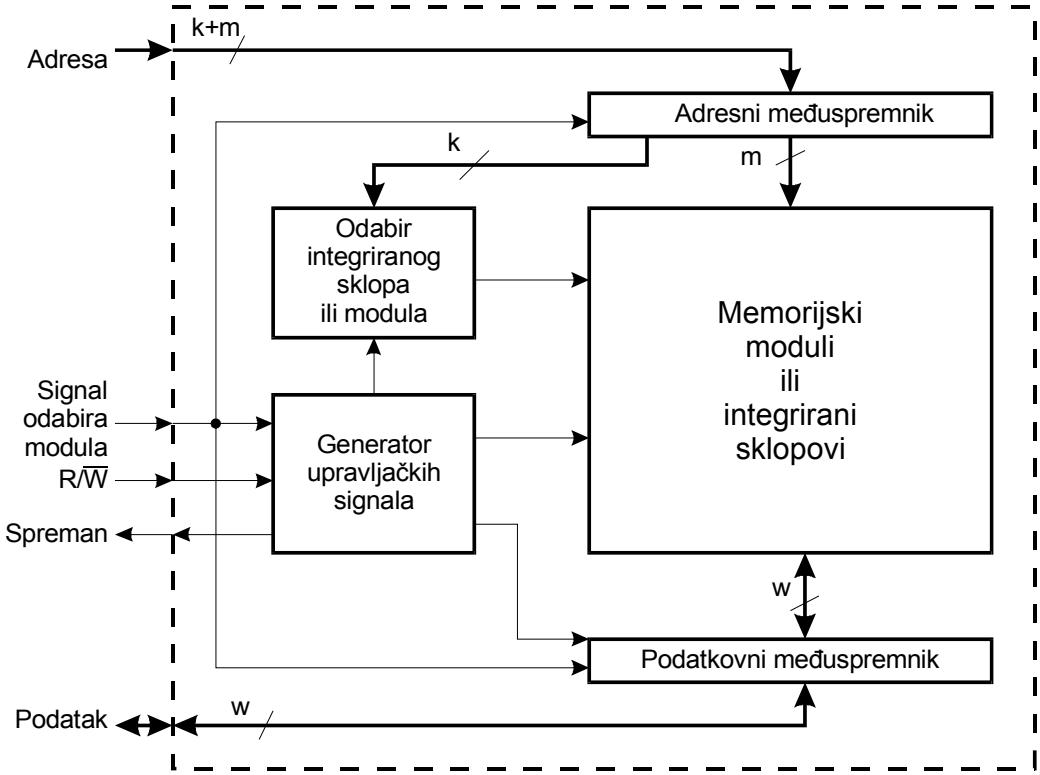


Slika 7.20. 1 Moktet DRAM modul.

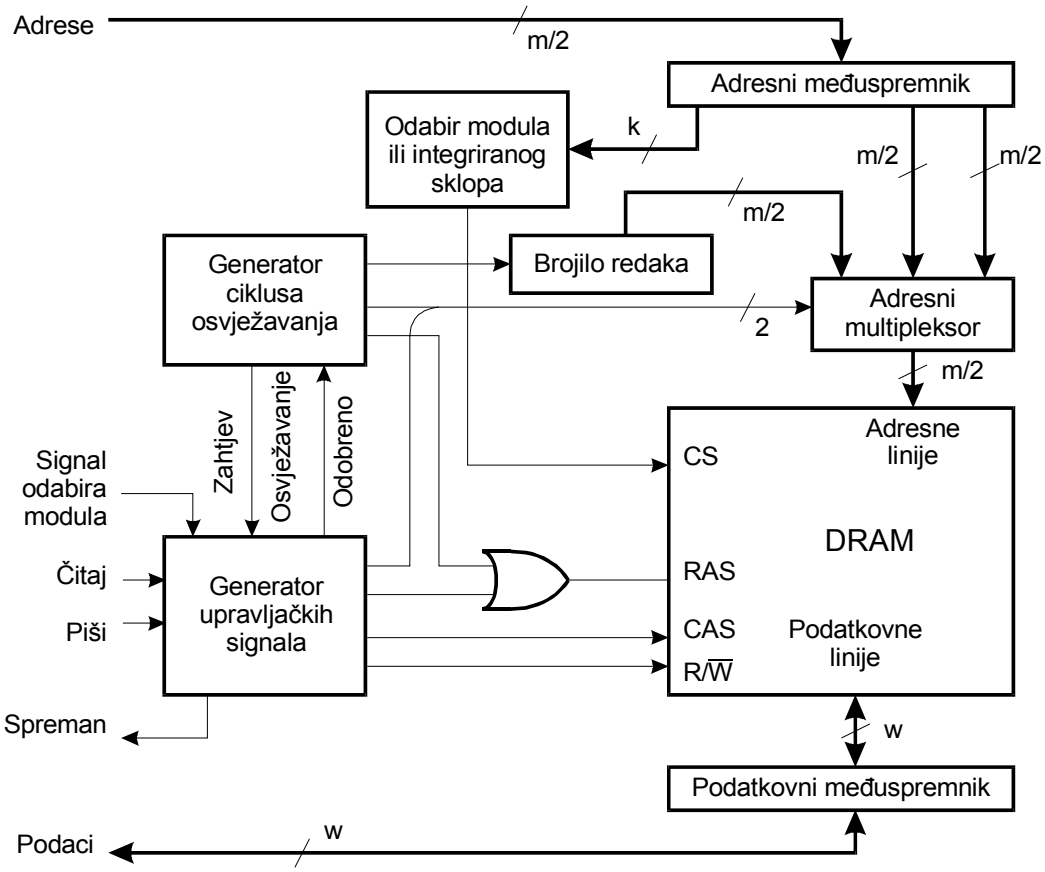
READY izlaz DRAM integriranog sklopa je izlaz s otvorenim kolektorom i ukoliko je u logičkoj nuli nije završen upis u memoriju ili se izvodi osvježavanje. Postavljanjem ovog

izlaza u logičku jedinicu integrirani sklop spreman je za sljedeći memorijski ciklus. Prema prikazanom rješenju tek kada su svi integrirani sklopovi uspješno obavili započetu memorijsku operaciju i ni jedan ne izvodi operaciju osvježavanja, dozvoljen je sljedeći memorijski ciklus. Ukoliko integrirani sklopovi imaju interni nezavisni sklop za upravljanje ciklusom osvježavanja tada je vjerojatnost osam puta veća da neki od sklopova nije spreman za novi memorijski ciklus. Ovo značajno usporava prosječnu brzinu odziva memorijskog modula pa projektanti običavaju koristiti rješenje s memorijskim integriranim sklopovima bez internog upravljačkog sklopa, a koriste zaseban sklop koji upravlja istovremeno ciklusima osvježavanja za sve memorijske integrirane sklopove.

Memorijski moduli osim memorijskih integriranih sklopova sadrže i međuspremnike za adrese i podatke s ciljem da procesor ili neka druga jedinica ne treba stalno održavati sadržaje na tim linijama tijekom cijelog memorijskog ciklusa. Modul radi nezavisno prihvaćajući adresu, upisujući je u adresni međuspremnik kao i upravljačke signale čitaj/piši, te podatak u slučaju operacije pisanja, odnosno predajući podatka u slučaju operacije čitanja. Po završetku operacije dojavljuje se procesoru ili drugoj jedinici koja pristupa memorijskom modulu da je operacija uspješno okončala. Ukoliko se koristi više modula u nekim slučajevima potrebno je posebnom logikom odabirati module (isto kao i odabir integriranih sklopova) ili to može raditi nezavisna logika na modulu (adresni dekoder) temeljen postavljene adrese. Moduli sa SRAM-ovima nešto su jednostavniji od modula s DRAM-ovima zbog izostanka logike za osvježavanje. Na slici 7.21. prikazan je memorijski modul s SRAM-om, a na slici 7.22. s DRAM-om.



Slika 7.21. Memorijski modul s međusklopovima.



Slika 7.22. DRAM memorijski modul sa sklopom za osvježavanje.

Vremenski dijagram sabirničkog ciklusa za memorijski modul prikazan na slici 7.21. ne mora biti striktno vezan uz brzinu odziva memorije koju modul koristi. Npr. za operaciju čitanja, procesor na adresnu sabirnicu može kratkotrajno postaviti adresu lokacije iz koje se čita i upravljački signal ČITAJ. Memorijski modul upisuje adresu u međuspremnik i aktivira preko upravljačke logike signale integriranim sklopovima na modulu. Podatak se prebacuje u podatkovni međuspremnik, a procesoru se dojavljuje preko SPREMAN upravljačke linije da je podatak spreman za prijenos. Generator upravljačkih signala sinkronizira sabirničke cikluse i cikluse memorijskog modula. Naravno ovaj opći koncept koji je ovdje izložen potrebno je adaptirati prema stvarnim integriranim sklopovima i procesoru koji se koristi.

Memorijski modul s DRAM-om, prikazan na slici 7.22., nešto je složeniji zbog logike za osvježavanje kao i dijeljenje adrese na adresu retka i adresu stupca memorijske matrice. Upravljački sklop generira upravljačke signale RAS, CAS, $R/\overline{W}$, multipleksira adresne linije te upravlja ciklusima osvježavanja. Poseban vremenski sklop generira zahtjev za osvježavanje retka matrice na koji pokazuje brojilo redaka. Multipleksor s 3 x m/2 ulaza određuje da li se na adresne ulaze dovodi adresa retka, adresa stupca ili adresa retka kojega je potrebno osvježiti. Adrese retka i stupca memorijske matrice dobivaju se dijeljenjem memorijske adrese na dva jednaka dijela, adresa retka je gornja polovica adrese dok je adresa stupca donja polovica adrese. Upravljački signali CAS kao i signali odabira memorijskih integriranih sklopova (CS signali) generiraju se samo temeljen memorijske adrese i signala čitaj/piši, a ne ovise o sklopu za osvježavanje. RAS signal generira se ili kod memorijskog ciklusa temeljen adrese i upravljačkih signala čitaj/piši ili za vrijeme ciklusa osvježavanja njenom upravljačkom logikom. Ukoliko vremenski sklop za upravljanje ciklusima osvježavanja postavi zahtjev tijekom

memorijskog ciklusa, zahtjev će biti odgođen (neće se odobriti) sve do završetka memorijskog ciklusa. Slično, ukoliko se postavi zahtjev za memorijskom operacijom tijekom ciklusa osvježavanja on će biti odgođen dok se započeta operacija ne završi. Prioritet u slučaju istovremenog zahtjeva za memorijskom operacijom i ciklusom osvježavanja ima ciklus osvježavanja. Signal spremna aktivira se tek po završetku obiju operacija.

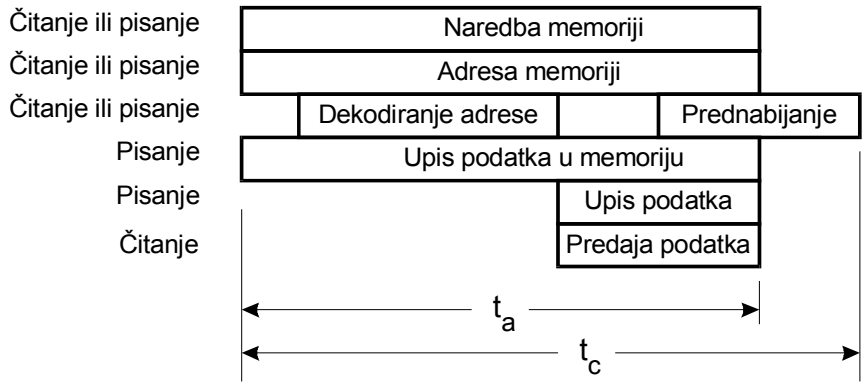
7.3.1. Performanse i kompromisi u projektiranju memorijskih modula

Različite domene implementacije memorijskih modula, kao i višestruka mogućnost njihovog sklopovskog povezivanja rezultira brojnim rješenjima koja se moraju usporediti s gledišta performansi i cijene.

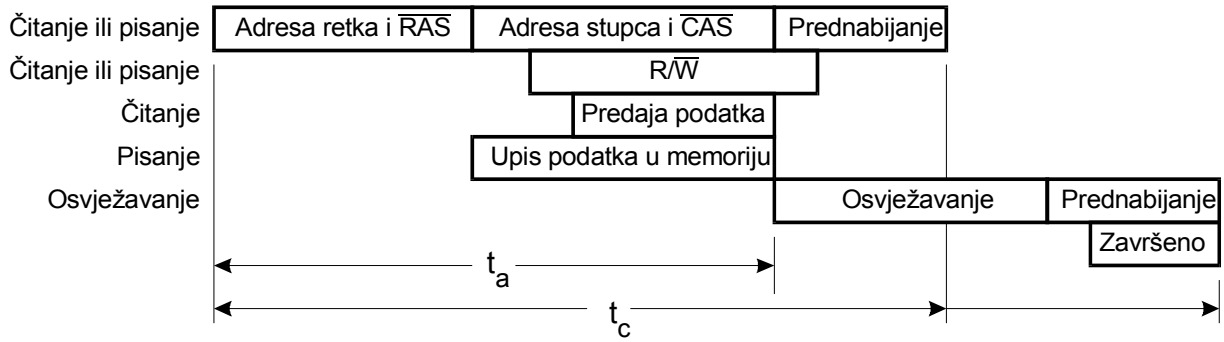
Performanse: Vrijeme pristupa. Kako je vidljivo iz dosadašnjih razmatranja vezanih uz projektiranje memorijskih modula, moguća se brojna sklopovska rješenja koja je potrebno analizirati kako bi se odredile njihove performanse. Više čimbenika određuje vrijeme pristupa memorijskom modulu. Brojne aktivnosti mogu se paralelno izvoditi tako da određivanje ovog parametra vezano je uz domenu implementacije, naročito da li se koriste statički ili dinamički RAMovi. Na najvišoj razini izvode se sljedeće operacije:

- U svakom pristupu memoriji:
  - prijenos adrese do memorije,
  - prijenos naredbi (čitaj/piši) do memorije,
  - dekodiranje adrese od strane memorije.
- Kod čitanja iz memorije:
  - predaja podatka upisanog na odabranoj lokaciji,
  - predaja signala o okončanju operacije (spreman).
- Kod pisanja u memoriju:
  - predaja podatka memoriji,
  - upis podatka u odabranu lokaciju,
  - predaja signala o okončanju operacije (spreman).

Slika 7.23. prikazuje kako se spomenute operacije uklapaju u memorijski ciklus, što je moguće izvoditi istovremeno, a što sekvencijalno. Aktivnosti uključuju i cikluse osvježavanja kod DRAM-a i prednabijanje podatkovnih linija kod SRAM-ova. Najgori slučaj određuje vrijeme pristupa t_a i vrijeme ciklusa t_c za svaki slučaj. Kod DRAM-ova moguća su dva slučaja u ovisnosti da li je ciklus osvježavanja u tijeku ili ne. Ukoliko memorijski modul ima i adresne i podatkovne međusklopove, potrebni su dodatni koraci za upis i čitanje iz njih.



a) Vremenski dijagram SRAMa



b) Vremenski dijagram DRAMa

Slika 7.23. Slijed koraka tijekom pristupa memoriji.

Primjer: 70ns SRAM ne znači 70ns memorijski modul. U ovom primjeru proračunati će se vrijeme ciklusa čitanja modula sa 70 ns 4k x 8bita SRAM-ovima. Neka je vrijeme kašnjenja veznih sklopova CPUa s adresnom sabirnicom 40 ns, a vrijeme propagacije signala preko adresne sabirnice 10 ns, tako da adresa ja prisutna na ulazu memorijskog modula nakon 50 ns. Sklopovlje za odabir modula (adresni dekoder) neka ima vrijeme odziva 20 ns te sklopovlje za odabir memorijskog integriranog sklopa dodatnih 20 ns. Praktički zahtjev i adresa dolaze do memorijskog integriranog sklopa sa zakašnjenjem od 90 ns.

Kako je već zadano memorijski integrirani sklop ima vrijeme pristupa kod čitanja podatka od 70 ns. Ovo vrijeme posljedica je vremena odziva dekoderske logike retka (6-64 linije) i ulaznog ili izlaznog multipleksora stupca (6-64 linije) te vremena upisa u memorijsku ćeliju. Ukoliko je dekoder/multipleksor realiziran kao matrična struktura prema slici 7.10. (četiri kaskadno spojena 4-16 multipleksora preko jednog 2-4 multipleksora) signal se propagira kroz tri razine logičkih vrata. Ukoliko je vrijeme propagacije kroz logička vrata 10 ns, vrijeme propagacije kroz multipleksor je 30 ns. Neka je i vrijeme odziva memorijske ćelije 10 ns, podatak se pojavljuje na ulazima izlaznog multipleksora u operaciji čitanja tek nakon 40 ns. Izalzni multipleksor ima također kašnjenje od 30 ns (tri razine logičkih vrata) pa se podatak pojavljuje na izlazu memorijskog integriranog sklopa 70 ns nakon postavljanja adrese i naredbe za čitanje.

Prema ovim proračunima podatak je na izlazu memorijskog integriranog sklopa 90 + 70 = 160 ns nakon što je procesor započeo operaciju čitanja. Ali ovaj podatak se prebacuje u izlazni međuspremnik memorijskog modula koji ima dodatno kašnjenje od 30 ns i tek

nakon 160 + 30 = 190 ns prisutan je na podatkovnoj sabirnici sustava. Uz kašnjenje sabirnice od 10 ns i 40 ns vremena odziva ulaznog veznog sklopa procesora podatak je pročitan nakon 240 ns. Ovo vrijeme je znatno duže od 70 ns što je deklarirano kao vrijeme odziva memorijskog integriranog sklopa.

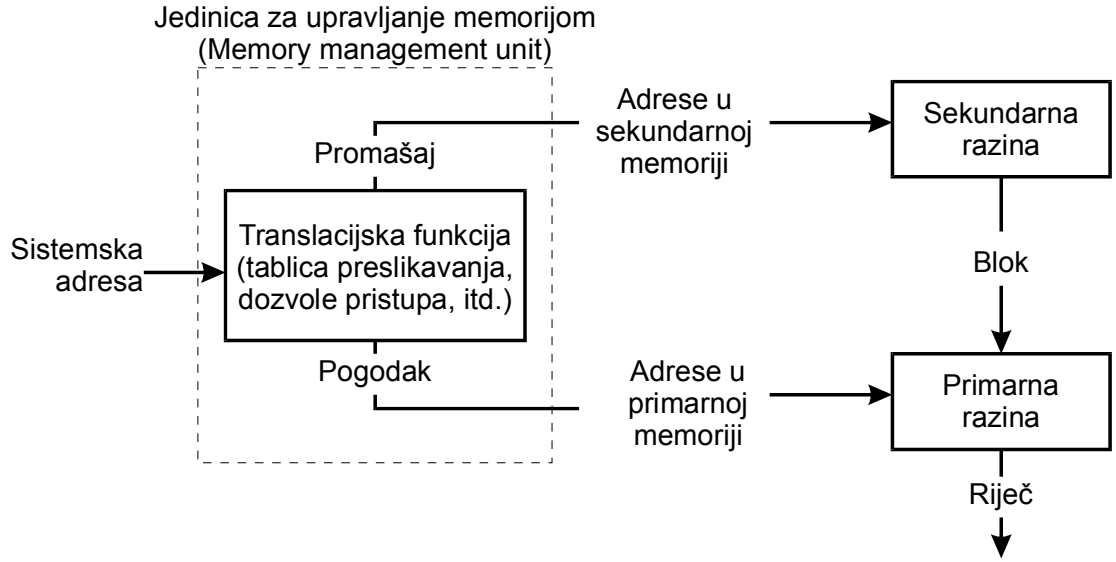
Provedena razmatranja pokazuju važnost pažljive analize utjecaja svih komponenata sustava na vremenski dijagram te performanse sustava.

7.4. Podjela memorije na dvije razine

Opis izvedbe memorijskih sklopova ukazuje da je sklopovska realizacija brzih memorija većeg kapaciteta u jednom integriranom sklopu praktički teško ostvariva. Razlozi su višestruki. Prvenstveno veći dekoderi zahtijevaju više razina logičkih vrata, a time se i povećava vrijeme propagacije podatka. Također brži sklopovi zahtijevaju veću potrošnju, a to je u suprotnosti s velikom gustoćom pakovanja. U svakom slučaju, brze memorije izrađuju se u integriranim sklopovima manjeg kapaciteta, a time se i povećava značajno njihova cijena. Kako je memorija važan čimbenik računarskog sustava, jer procesor u svakom svom ciklusu pristupa jednom a nekad i više puta memoriji s ciljem dohvata naredbe i podataka. Zato brzina memorije ima značajan utjecaj na sveukupne performanse računarskog sustava. Da bi se navedeno ostvarilo uz prihvatljivu cijenu memorija se dijeli na dvije ili više razina s obzirom na brzinu i kapacitet.

Hijerarhijski koncept memorije podrazumijeva razinu brze memorije (*cache*) kojoj procesor pristupa u jednom taktu, te razinu sporije memorije koja je znatno većeg kapaciteta. Procesor pristupa informacijama, naredbe i podaci, koje su samo u brzoj memoriji. Ukoliko informacija nije u brzoj memoriji, potrebno ju je prebaciti iz sporije u brzu memoriju. Ova operacija zahtijeva dodatno vrijeme i smanjuje brzinu izvođenja obrade. Tako je učinkovitost ovog koncepta obrnuto proporcionalna frekvenciji prebacivanja podataka iz sporije u bržu memoriju. Odnosno, što procesor češće nalazi potrebne informacije u brzoj memoriji, rjeđe je potrebno iste prebacivati iz spore u brzu memoriju te sustav radi bliže maksimalno mogućoj brzini obrade. U prilog ovom konceptu ide princip lokaliteta (*principle of locality*), odnosno činjenica da program u nekom vremenskom intervalu pristupa relativno uskom memorijskom području. Ovo nije teoretska postavka već činjenica koju su uočili inženjeri i programeri tijekom promatranja izvođenja stvarnih programa.

Lokalitet može biti prostorni (*spatial*) i vremenski (*temporal*). Prostorni lokalitet podrazumijeva da ukoliko se u jednom trenutku pristupilo jednoj memorijskoj lokaciji da je velika vjerojatnost u sljedećim trenutcima pristupa njoj susjednim lokacijama. Vremenski lokalitet je posljedica činjenice da ukoliko se pristupi jednoj memorijskoj lokaciji da je velika vjerojatnost ponovnog pristupa istoj unutar kratkog vremenskog intervala. Uz pojam lokaliteta vezuje se radni skup memorijskih lokacija (*working set of memory locations*) koji se odnosi na skup memorijskih lokacija kojima se pristupa u određenom vremenskom intervalu. Kod većine programa radni skup se relativno sporo mijenja s vremenom.

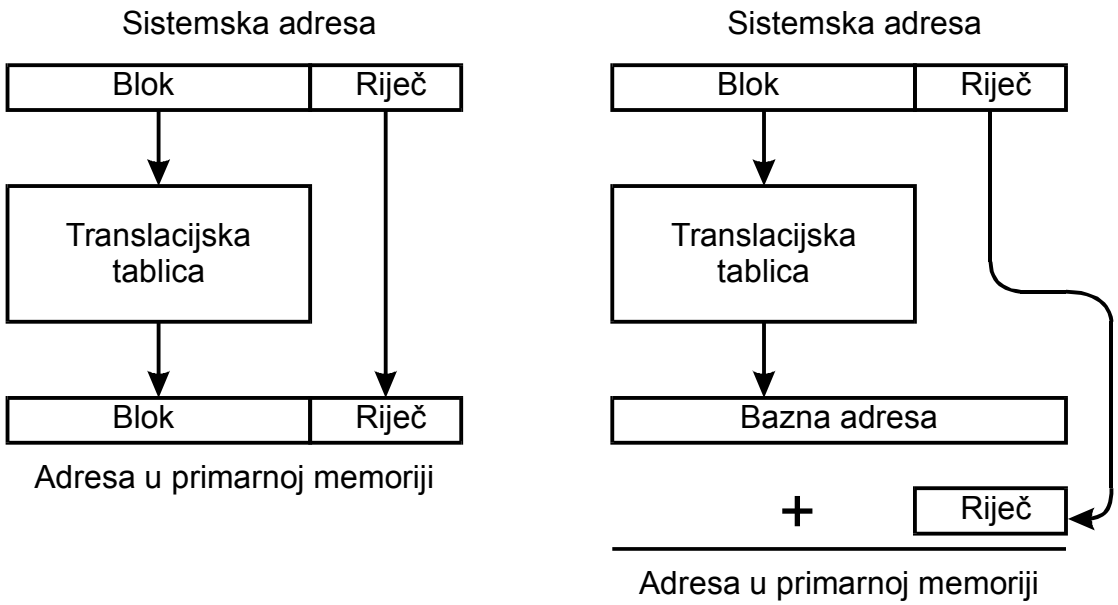


Slika 7.24. Adresiranje i pristup dvo-razinskoj memoriji.

Sistemska adresa, npr. adresa iz procesora, dovodi se na jedinicu za upravljanje memorijom (*Memory Management Unit skr. MMU*). Ova jedinica izvodi funkciju preslikavanja za određeni par primarna-sekundarna memorija. Uz ovu funkciju moguće je dodati i neke dopunske funkcije kao što je pravo pristupa određenim memorijskim lokacijama. Ukoliko je informacija u primarnoj razini tada MMU generira adresu lokacije u kojoj je informacija upisana, a ukoliko nije izdaje nalog za po potrebi izbacivanje jednog bloka iz primarne i unos novog bloka iz sekundarne memorije. MMU mora funkciju detekcije prisutnosti informacije u primarnoj razini i proračun njene adrese temeljen sistemske adrese izvesti jako brzo kako se ostvarile što bolje performanse sustava. Razlog je što procesor praktički u svakom ciklusu pristupa preko MMUa primarnoj memoriji. Promašaji koji moraju biti relativno rijetki ne zahtijevaju tako brz servis. Zato se zamjena blokova između dviju razina može realizirati i nešto sporije. Ovo posebice vrijedi ukoliko je sekundarna memorija magnetski disk.

Zahtjev za brzom translacijom adrese znatno je utjecao na rješenje funkcije preslikavanja iz sistemske adrese u adresu primarne memorije. Na slici 7.25 prikazana su dva moguća rješenja translacija adresa. Početna adresa bloka u primarnoj memoriji dobiva se pomoću tablice preslikavanja (*lookup table*). Kod straničenja (*paging*), bitovi većeg značenja određuju početnu adresu bloka preko tablice preslikavanja, dok bitovi manjeg značaja određuju položaj (pomak od početka) riječi unutar bloka kao što je prikazano na slici 7.25.a. Adresa u primarnoj memoriji dobiva se povezivanjem (konkatencijom) ova dva podatka. Ovakav princip određivanja primarne adrese koristi se kod brzih memorija. U ovom slučaju blok se naziva linija (*cache-line*)

Kod sustava koji koriste segmentaciju, bitovi većeg značenja sistemske adrese preko tablice preslikavanja određuju baznu adresu bloka kojoj se pribraja redni broj riječi unutar bloka, kao što je prikazano na slici 7.25.b. Ovakav princip koristi se kad je primarna memorija glavna memorija, a sekundarna memorija je magnetski disk. Kod straničenja, blokovi ne mogu počinjati na proizvoljnim adresnim lokacijama u primarnoj memoriji, dok kod segmentacije je to dozvoljeno.



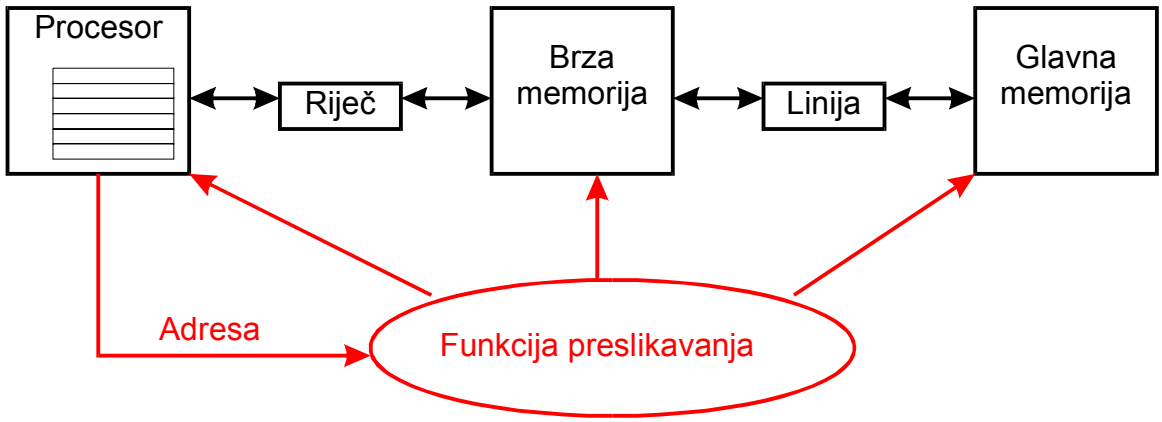
Slika 7.25. Određivanje primarne adrese.

7.5. Brza memorija (*Cache*)

Rad brze memorije transparentan je programeru. Program generira efektivnu adresu (prije nazvana sistemska adresa) te definira operaciju (čitanje ili pisanje). Memorijski sustav mora realizirati ovu operaciju neovisno da li je informacija u primarnoj ili samo u sekundarnoj memoriji. Način realizacije memorijske transakcije nije vidljiv programu i programeru. Između procesora i primarne memorije prenose se riječi dok se između primarne i sekundarne memorije prenose linije (blokovi) riječi.

Funkcije preslikavanja (*Mapping function*)

Funkcije preslikavanja između različitih memorijskih razina prikazane su na slici 7.26.



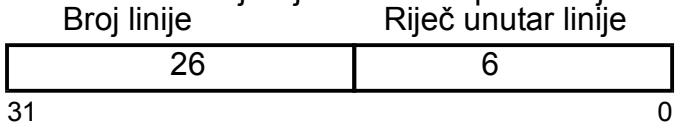
Slika 7.26. Funkcija preslikavanja kod brze memorije.

Funkcije preslikavanja odgovorne su za funkcioniranje više-razinske memorije. Zbog brzine rada ove funkcije su sklopovski realizirane i određuju sljedeće:

- Strategiju unosa linije - gdje u brzu memoriju pohraniti liniju iz glavne memorije,
- Strategiju zamjene – koju liniju iz brze memorije zamijeniti ako adresirana linija nije u brzoj memoriji (*cache miss*),
- Strategiju čitanja i pisanja – kako izvoditi operacije čitanja i pisanja ukoliko je linija u brzoj memoriji (*cache hit*) ili nije u njoj (*cache miss*).

Danas se susreću tri različite funkcije preslikavanja: asocijativno (*associative*), direktno (*direct*), asocijativno po skupinama blokova (*block-set-associative*).

Kako bi se realizirala funkcija preslikavanja memorijska adresa dijeli se na dva dijela: broj linije i pomak riječi unutar linije. U sljedećem primjeru 32 bitovna memorijska adresa podijeljena je na 26 bita za broj linije i 6 bita za pomak riječi unutar linije veličine:

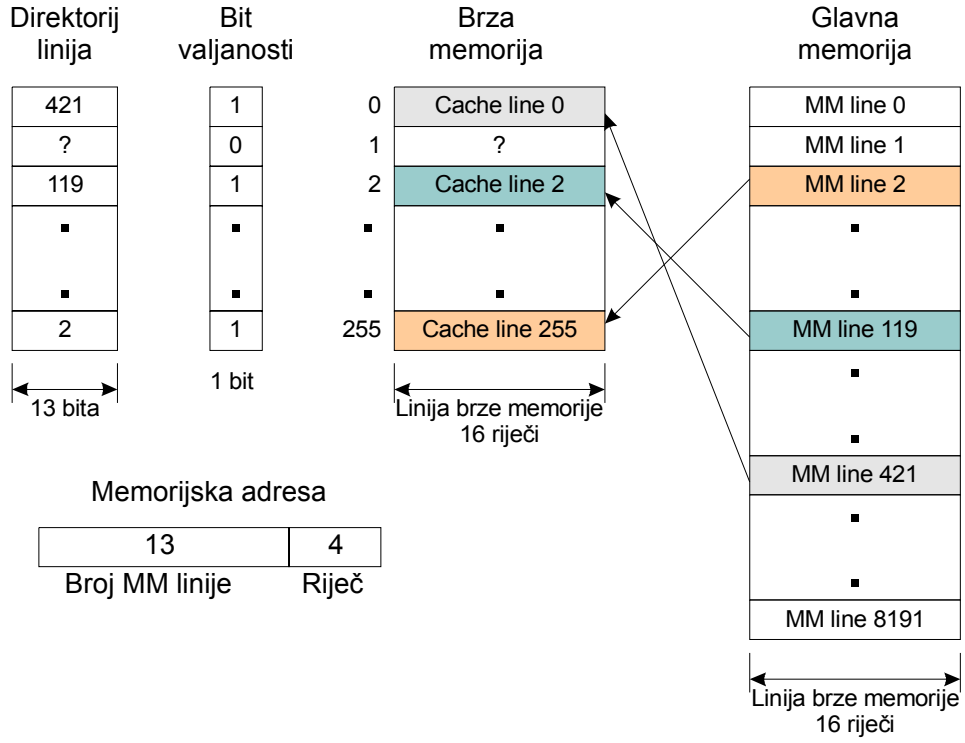


Na ovaj način moguće je adresiranje 2^{28} = 256 M linija veličine 2^4 = 16 riječi.

Asocijativno preslikavanje

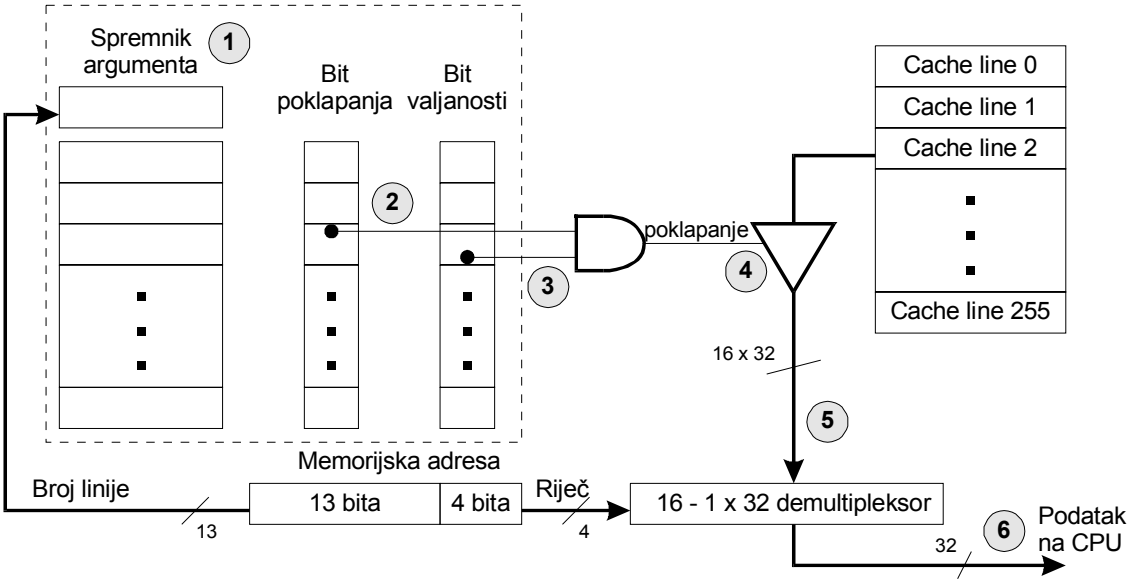
Kod asocijativnog preslikavanja svaka linija iz glavne memorije može se smjestiti bilo gdje u brzoj memoriji. Nakon što se unese u brzu memoriju linija je jedinstveno identificirana brojem linije ili znakom (*tag*) koji se upisuje u posebni dio brze memorije direktorij blokova (*tag memory*). Također bez obzira na vrstu brze memorije linija upisana u brzoj memoriji može ali i ne mora sadržavati valjane podatke. Npr. prilikom ukapčanja sustava svi podaci u brzoj memoriji su nevažeći. To je razlog da se uz direktorij linija uvodi i bitovi valjanosti (*valid bit*) koji označavaju da li je dana linija važeća ili ne.

Na slici 7.27. prikazan je primjer realizacije asocijativne memorije s 256 linija veličine 16 riječi.



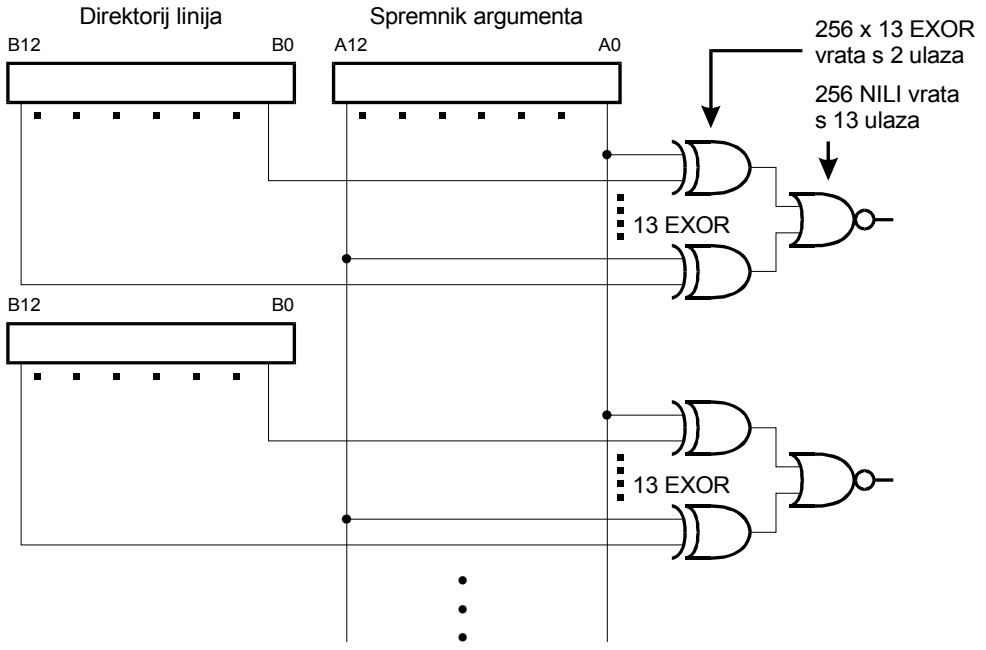
Slika 7.27. Asocijativna brza memorija.

Princip rada asocijativne memorije prikazan je slikom 7.28.



Slika 7.28. Princip rada asocijativne memorije.

Prilikom pristupa brzoj memoriji potrebno je prvo provjeriti da li se tražena linija nalazi u brzoj memoriji. Broj linije se prebacuje u spremnik adresa i uspoređuje sa sadržajem direktorija linija ①. Zbog brzine obrade pretraživanje je potrebno izvršiti paralelno. Zato se direktorij linija pohranjuje u asocijativnu memoriju. Sadržaj svih asocijativnih spremnika istovremeno se uspoređuje sa sadržajem spremnika argumenta te se rezultat usporedbe pohranjuje u bit poklapanja. Asocijativna memorija realizira se pomoću XOR vrata, slika 7.29. Svaki bit asocijativnih spremnika se uspoređuje s odgovarajućim bitom spremnika argumenta XOR vratima. Ako su bitovi isti izlaz XOR vrata je 0 inače je 1. Izlazi svake riječi (za ovaj primjer 13 izlaza) vode se na NILI vrata čiji izlaz je 1 samo ako su svi bitovi spremnika koji se uspoređuju jednaki. Rezultat se pohranjuje kao bit poklapanja.



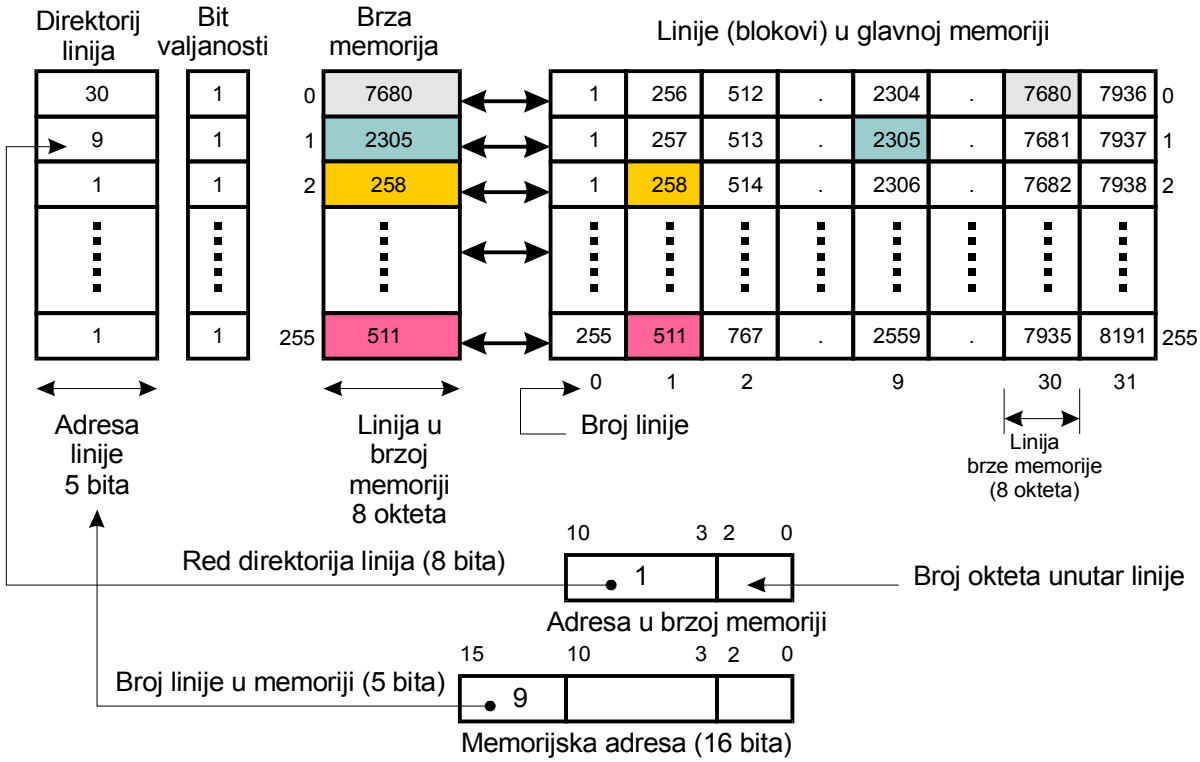
Slika 7.29. Izvedba asocijativne memorije.

Ovakvo rješenje očito zahtjeva dosta sklopova pa se asocijativni spremnici realiziraju samo u manjim kapacitetima.

Rezultat pretraživanja ② uspoređuje se s bitom valjanosti ③ pomoću I vrata. Ukoliko je blok u brzoj memoriji i ujedno je označen kao valjan generira se signal propuštanja (*Gate*) izlaznim međuspremnicima odgovarajuće linije brze memorije (*Cache line*) ④. Cijela linija 16 x 32 bita se prebacuje u spremnik za odabiranje podatka te se iz linije selektira tražena riječ ⑤ koja se postavlja na podatkovnu sabirnicu procesora ⑥.

Direktno mapirana brza memorija (*Direct-Mapped Cache*)

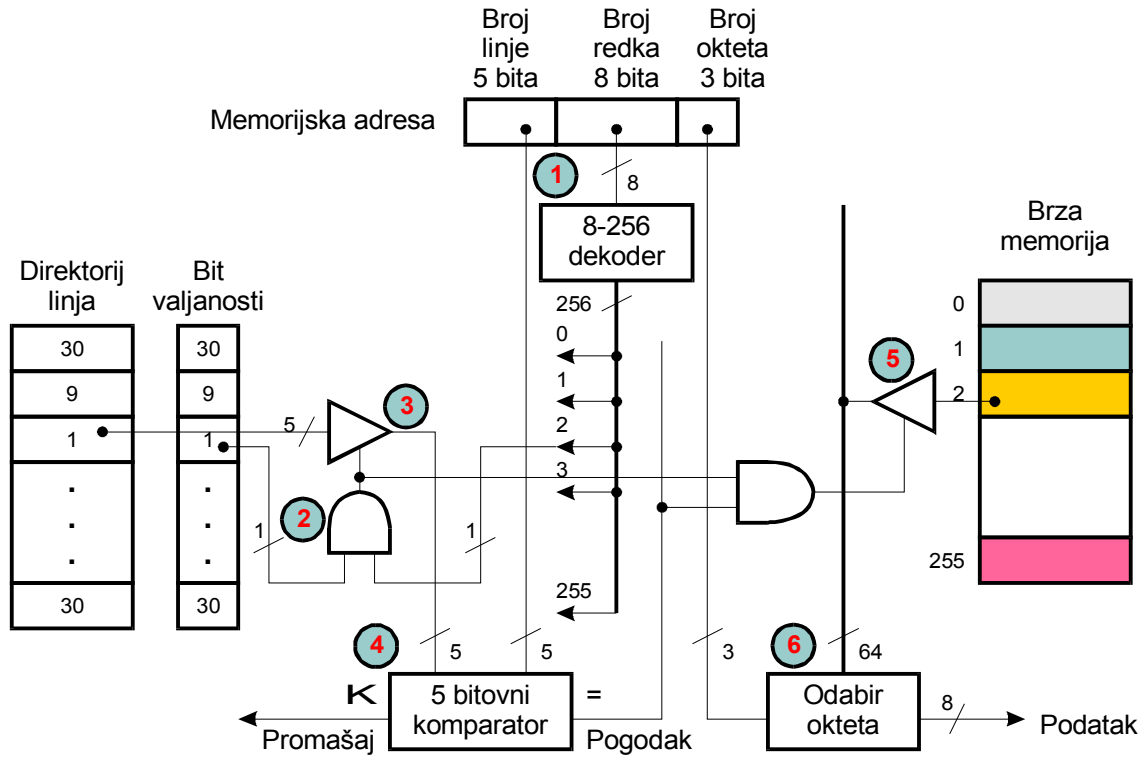
Direktno mapirana memorija je sasvim različit pristup pohrane memorijske linije. Za razliku od asocijativne memorije gdje se memorijska linija mogla upisati na proizvoljno mjesto u brzoj memoriji, kod direktno mapirane brze memorije linija se može nalaziti samo na određenom mjestu u brzoj memoriji. Na slici 7.30. je prikazan primjer direktno mapirane memorije.



Slika 7.30. Direktno mapirana brza memorija.

Glavna memorija je podijeljena na 8191 linija s 8 okteta. Blokovi su postavljeni u 256 redova s 32 linija. Brza memorija ima 256 linija (koliko ima redova glavne memorije) te se u brzu memoriju prebacuje samo po jedna linija iz svakog pojedinog reda iz glavne memorije. U direktorij linija upisuje se broj stupca u kojem je linija pohranjena u glavnoj memoriji a koji je upisan u brzoj memoriji. Tako bitovi 3 – 10 memorijske adrese (8 bita) određuje koji se red adresira, bitovi 11 – 15 stupac u kojem je memorijska linija (5 bita), a bitovi 0 – 2 (3 bita) oktet unutar linije. Na primjeru na slici u prvom redu se nalaze linije 0, 256, 512, ..., n·256, n $\in$ 0, 31, u drugom 1, 257, ..., n·256+1, itd. U brzoj memoriji je upisana 30. linija iz prvog reda (linija broj 30·256 = 7680) u drugom 9. linija iz drugog reda (linija broj 9·256 + 1 = 2305), itd.

Na slici 7.31.prikazan je princip rada direktno mapirane brze memorije.



Slika 7.31: Princip rada direktno mapirane brze memorije.

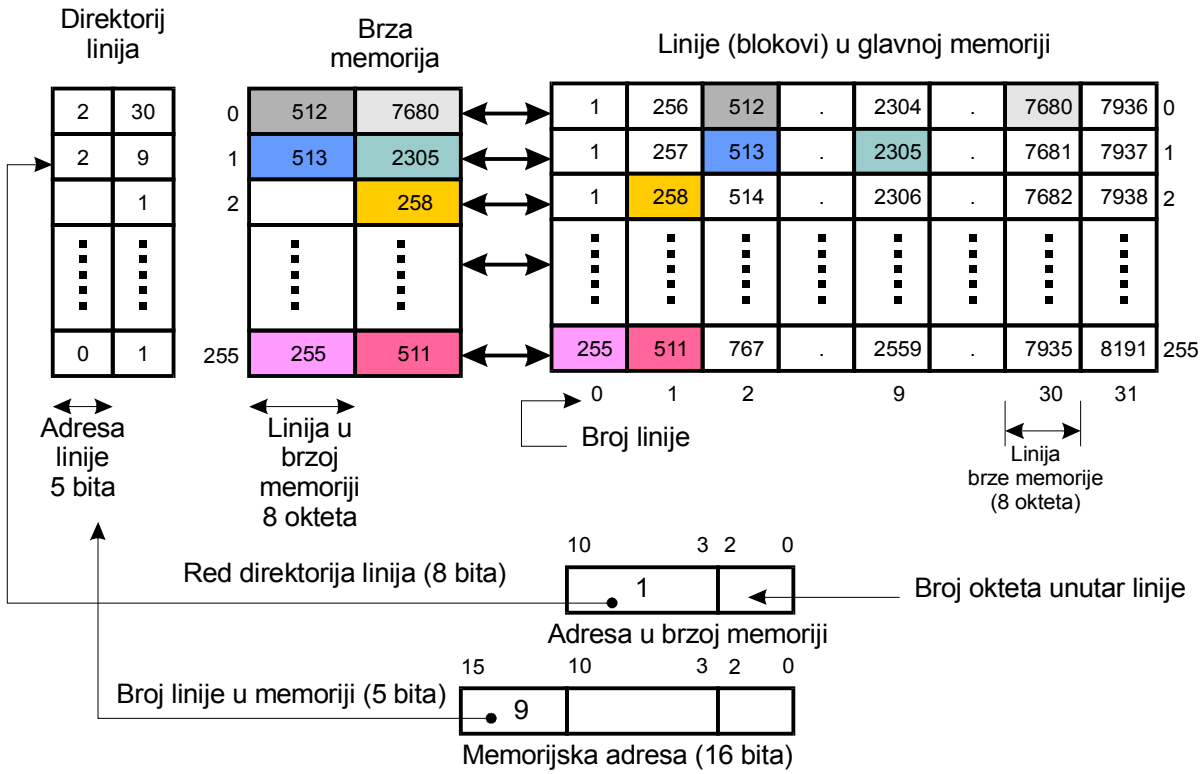
Bitovi 3 – 11 16 bitovne memorijske adrese određuju redak u kojem se nalazi memorijska linja. Ovi bitovi dovode se na dekoder koji 8 bita dekodira na 256 linja koji preko I vrata selektiraju određeni redak ①. Ukoliko je linja valjana ②, sadržaj spremnika iz direktorija linja (5 bita) se prosljeđuje preko međusklopa s tri stanja na jedan od ulaza digitalnog komparatora ③. Ovaj podatak se uspoređuje s 5 bita najveće važnosti memorijske adrese kako bi se ustanovilo da li je tražena linja u brzoj memoriji ④. Ukoliko linja nije u brzoj memoriji (vrijednost u direktoriju linja je različita od vrijednosti u memorijskoj adresi) signalizira se promašaj. Ukoliko su uspoređene vrijednosti iste, linja je u brzoj memoriji. Tada se ta linija iz brze memorije (8 okteta), preko međusklopa s tri stanja⑤ prosljeđuje na sklop za odabir okteta koji temeljen tri bita najmanjeg značaja memorijske adrese odabire podatak iz linje ⑥ i prosljeđuje ga procesoru.

Ovakva organizacija brze memorije je relativno jednostavna i praktično je dosta efikasna ukoliko se program izvodi skoro sekvencijalno i dosta lokalno. Ako su učestali skokovi izvan granica kapaciteta brze memorije (256 x 8 = 2 kokteta) može doći do učestalih promašaja što rezultira znatnim smanjenjem performansi sustava.

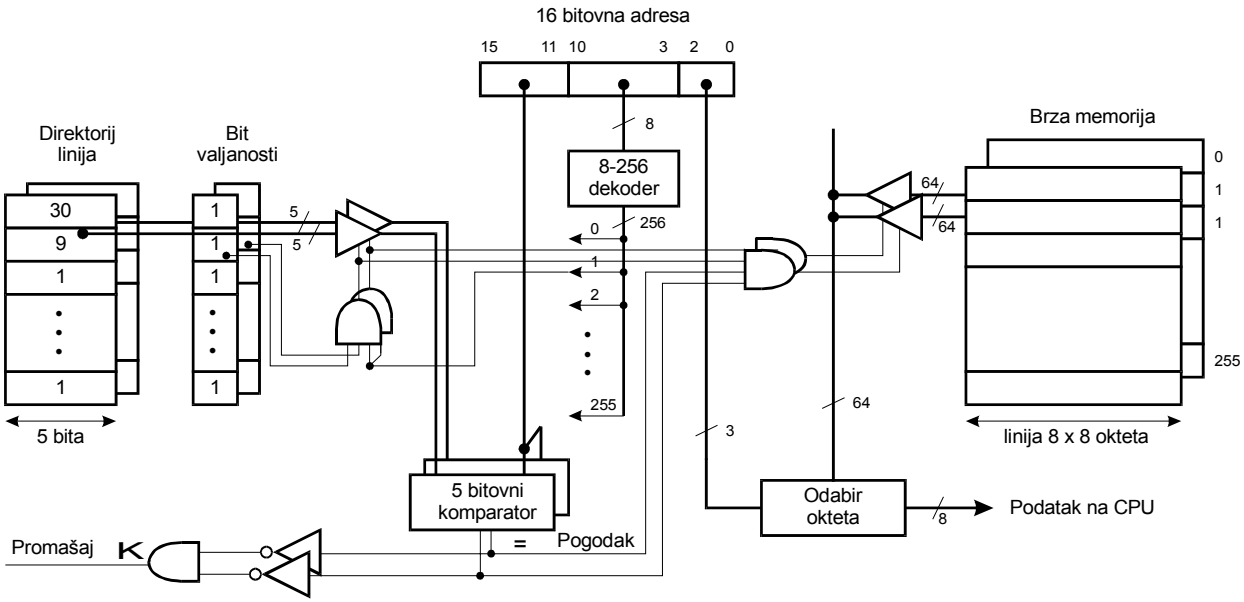
Memorijske linije asocijativno preslikane po skupinama

(*Block-Set-Associative Caches*)

Asocijativno preslikavanje unutar skupina linija je kombinacija dviju prethodno opisanih metoda. Direktorij linija kod direktno preslikane memorije se proširuje dodatnim stupcima. Ukoliko se proširi na dva stupca naziva se dvostruka, s četiri stupca četverostruka itd. brza memorija s linijama asocijativno preslikanim po skupinama (*Two-Way Block-Set-Associative Caches, Four-Way ..., itd.*). Primjer brze memorije s dvije skupine linija asocijativno preslikanih po skupinama prikazan je slikom 7.32.



Slika 7.32: Brza memorija s dvije skupine linija koji su asocijativno preslikani po skupinama.



Slika 7.33: Izvedba brze memorije s dvije linije asocijativno preslikane po skupinama.

Glavna memorija je i dalje podijeljena na retke i stupce po linijama kao i kod direktnog preslikavanja. Direktorij linija, kao i brza memorija su prošireni na dva stupca. U svaki od redaka brze memorije može se upisati do dvije linije iz odgovarajućeg retka glavne memorije. Brojevi upisanih linija upisuju se u pripadajući redak i stupac direktorija linija. Memorijska adresa je podijeljena na isti način kao i kod direktno preslikane memorije. Prva tri bita određuju riječ unutar linije. Sljedećih osam bita određuje redak u kojem se linija nalazi a preostalih pet bita stupac linije u glavnoj memoriji. Kod pretraživanja

direktorija brze memorije osam bitova (3 – 10) direktno selektiraju redak direktorija dok se moguća dva polja asocijativno pretražuju uspoređujući se s pet bita najvećeg značenja memorijske adrese. Zbog jednostavnosti prikaza na slici su izostavljeni bitovi valjanosti. Izvedba dvostruke brze memorije s linijama asocijativno preslikanim po skupinama prikazana je na slici 7.33.

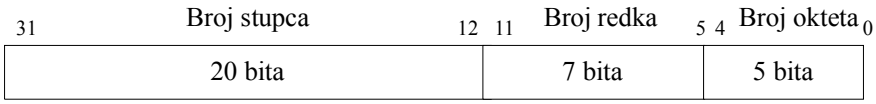
Jedinstvena brza interna memorija za naredbe i podatke

Kao što je već napomenuto procesor ne pristupa samo memoriji tijekom dohvata naredbe nego i tijekom dohvata operanada i pohrane rezultata. Zbog navedenoga važno je i da se podaci nalaze u brzoj memoriji. Većina suvremenih procesora ima jedinstvenu internu brzu memoriju za naredbe i podatke. Tako jedinica za preddohvat naredbi preko interne brze memorije pristupa naredbama, a jedinica za obradu cjelobrojnih ili brojeva s pomičnim zarezom pristupa operandima i pohranjuje rezultate u internu brzu memoriju. U slučaju kada se bilo linija naredbi ili linija podataka ne nalaze u internoj brzoj memoriji potrebno ga je unijeti iz glavne memorije.
Ovakav pristup ima određenih nedostataka. Kako internoj brzoj memoriji pristupa više različitih jedinica procesora tada je otežana realizacija njihova paralelnog rada. Naime dok npr. jedinica za obradu cjelobrojnih brojeva dohvata operande, jedinica za preddohvat mora čekati. Ovim se usporuje cjelokupni rad sustava. Također, npr. u slučaju promašaja podataka može se izbaciti iz brze memorije jedna linija naredbi kako bi se napravilo mjesto za novu liniju podataka. Ovim je memorija nejednoliko dodijeljena naredbama i podacima.

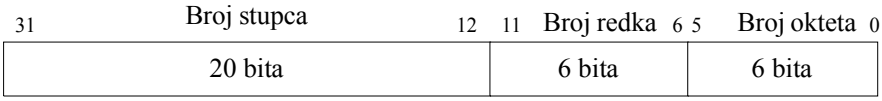
Posebna brza interna memorija za naredbe a posebna za podatke
Navedeni nedostaci jedinstvene interne brze memorije mogu se riješiti uporabom posebne interne brze memorije u koju su upisane naredbe (program), a posebne za podatke. Ovakvom arhitekturom jedinica za preddohvat naredbi istovremeno može pristupati informacijama s jedinicom za obradu.

Kod različitih procesora susreću se i različita rješenja:

- Intel 486 ima jedinstvenu internu četverostruku brzu memoriju s linijama asocijativno preslikanim po skupinama kapaciteta 8 k okteta.
- Intel Pentium ima podijeljenu internu dvostruku brzu memoriju s linijama asocijativno preslikanim po skupinama kapaciteta 8 k okteta za naredbe i 8 k okteta za podatke.



- PowerPC 601 ima jedinstvenu internu osmostruku brzu memoriju s linijama asocijativno preslikanim po skupinama kapaciteta 32 k okteta.



Ako bi se napravila usporedba koliko je sklopova potrebno za realizaciju stvarne brze memorije kod Intel Pentium procesora i koliko bi sklopova bilo potrebno ukoliko se ista memorija realizira kao potpuno asocijativna dobivaju se sljedeći rezultati:

	Kapacitet direktorija (bita)	Sklopovi za usporedbu (XOR)
Two-Way Block-Set-Associative Caches	20 x 128 x 2 = 5120	20 x 2 = 40
Associative Caches	27 x 256 = 6912	27 x 256 = 6912

Vanjska brza memorija (*External, Level Two (L2) Cache*)

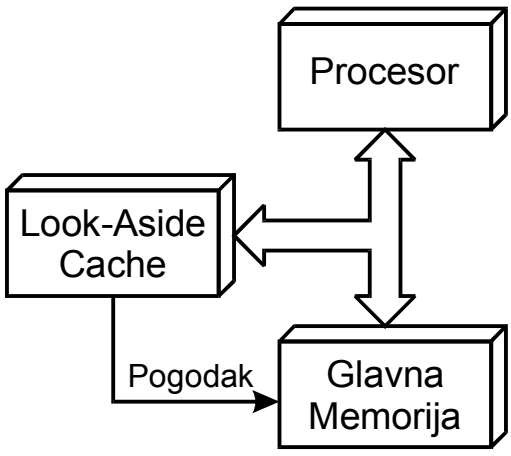
Interna brza memorija ima značajan učinak na karakteristike sustava, ali zbog relativno malog kapaciteta učestali pristupi glavnoj memoriji ponovo su ograničavajući faktor u radu sustava. U smislu poboljšanja karakteristika sustava, projektanti ubacuju još jednu razinu brze memorije koja je smještena između procesora (njegove interne brze memorije) i glavne memorije. Ova memorija je statička (SRAM), i može raditi maksimalnom frekvencijom kojom procesor pristupa vanjskim sklopovima. Obično eksterna brza memorija radi bez umetnutih stanja čekanja (*zero wait state*). To je sporije nego pristup internoj brzoj memoriji, ali je znatno brže nego pristup glavnoj memoriji.

Eksterna brza memorija ima funkciju provjere da li je informacija kojoj se pristupa upisana u njoj kao i način prijenosa bloka informacija iz i u glavnu memoriju. Prema načinu realizacije razlikuje se:

- paralelni spoj (*look-aside*),
- serijski spoj (*look-trough*).

Paralelni spoj eksterne brze memorije (*Look-Aside Cache*)

Slika 7.34. prikazuje način spajanja paralelne vanjske brze memorije s procesorom i glavnom memorijom.



Slika 7.34.: Look-Aside Cache.

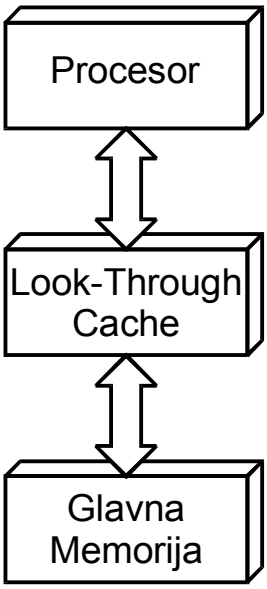
Brza memorija smještena je paralelno procesorskoj vanjskoj sabirnici i prati aktivnosti procesora. Kada procesor pristupa vanjskoj memoriji (promašaj u brzoj unutarnjoj memoriji) vanjska brza memorija provjerava da li je tražena informacija pohranjena u njoj ili ne. Ako je prebacuje se linija iz eksterne u internu brzu memoriju. U slučaju promašaja linija se prebacuje iz glavne memorije što rezultira usporenjem rada sustava.

Paralelno kako se prebacuju informacije u internu brzu memoriju iste se upisuju u eksternu brzu memoriju i ažurira se njen direktorij.
Na prvi pogled izgleda da je L2 brza memorija nepotrebna jer se identični sadržaj nalazi i u L1 brzoj memoriji. Pri tome je ispušteno iz razmatranja da je L2 brza memorija znatno većeg kapaciteta od L1 memorije. Tako s vremenom kako L1 memorija postaje popunjena potrebno je izbaciti pojedinu liniju iz nje. Ta ista linija ostaje u L2 brzoj memoriji pa u slučaju ponovnog pristupa informaciji iz izbačene linije ista se brže prebacuje ponovo u L1 brzu memoriju. S vremenom L1 brza memorija sadrži podskup informacija upisanih u brzu L2 memoriju.

Nedostatak paralelnog spoja eksterne brze memorije i glavne memorije je u tome što se svaki pristup vanjskoj memoriji (L2 brzoj memoriji i glavnoj memoriji) odvija preko iste sabirnice. Ovo je neprikladno u višeprocesorskim sustavima ili u sustavima s inteligentnim sklopom za upravljanje sabirnicom kada ovo rješenje umanjuje iskoristivost procesorske sabirnice.

Serijski spoj eksterne brze memorije (*Look-Through Cache*)

Slika 7.35. prikazuje serijski spoj eksterne brze memorije s procesorom i glavnom memorijom. Za razliku od paralelnog spoja u serijskom spoju eksterna brza memorija je smještena između procesora i radne memorije. Zadatak eksterne brze memorije je da snabdije internu brzu memoriju informacijom koja nije u njoj pohranjena. Ukoliko je ta informacija upisana u eksternoj brzoj memoriji, linija se maksimalnom brzinom prebacuje u internu brzu memoriju. U ovom slučaju nema nikakvih aktivnosti na sabirnici glavne memorije i ona je dostupna ostalim procesorima ili uređajima.



Slika 7.35: Look-Through Cache.

U slučaju promašaja (informacija nije pohranjena u eksternoj brzoj memoriji, ali je u glavnoj memoriji) linija se prvo preko sistemske sabirnice iz glavne memorije prebacuje u L2 brzu memoriju i istovremeno kroz nju prosljeđuje internoj brzoj memoriji.

Nedostatak ovakvog rješenja je unošenje dodatnog kašnjenja koje je posljedica ispitivanja da li je potrebna informacija u L2 brzoj memoriji. Ali u višeprocesorskim sustavima, gdje svaki procesor ima vlastitu L1 i L2 brzu memoriju, oslobađanje

sistemske sabirnice od nepotrebnih operacija značajno pridonosi povećanju sveukupnih performansi sustava.

Upis u brzu memoriju

Kod upisa u brzu memoriju (L1 ili L2) javlja se problem konzistentnosti podataka. Naime podatak upisan u brzu memoriju na određenoj razini ne odgovara njegovoj slici na drugoj razini. Npr. podatak upisan u L1 brzu memoriju nije isti onom u L2 brzoj memoriji ni onome u glavnoj memoriji. Kako bi se riješila konzistentnost podataka u memoriji na svim razinama koriste se sljedeće metode:

- pisanje kroz (*write through*),
- upis natrag (*write back*).

Pisanje kroz (*write through*)

Kada procesor upisuje rezultat operacije natrag u memoriju izvodi to na sljedeći način. Ako je blok prisutan u L1 brzoj memoriji, podatak se tamo ažurira i ujedno se ažurira i u L2 brzoj memoriji i u glavnoj memoriji. Ovakav pristup osigurava da su podaci na svim memorijskim razinama u suglasnosti.

Ukoliko podatak nije u L1 brzoj memoriji ne prebacuje se memorijski blok u L1 brzu memoriju nego se upisuje samo u L2 brzu memoriju (ako je u njoj podatak prisutan) i u glavnu memoriju.

Upis natrag (*write back*)

Kod ovakvog pristupa prilikom upisa podatka u memoriju provjerava se prvenstveno da li je podataka u L1 brzoj memoriji te ako je on se ažurira. Njegova kopija na ostalim memorijskim razinama ostaje neažurirana. Tako samo interna brza memorija ima najnoviju kopiju podatka dok su kopije na ostalim razinama zastarjele. L1 brza memorija obilježava memorijski blok kao blok koji je mijenjan i da je različit od onih na drugim razinama.

Ukoliko lokacija u koju se upisuje nije u L1 brzoj memoriji moguća su sljedeća rješenja:

- Podatak se upisuje samo u sljedeću memorijsku razinu u kojoj je ta lokacija prisutna (L2 brza memorija ili glavna memorija).
- Učitava se linija u L1 brzu memoriju te nakon njenog prebacivanja izvodi se izmjena sadržaja adresirane memorijske lokacije i linija se označava kao promijenjena.

Strategije zamjene linija

Do sada je razmatran samo slučaj kada se unosi linija kada postoji slobodnog mjesta u brzoj memoriji. Naravno, s vremenom se brza memorija popuni te je kod promašaja potrebno isprazniti određenu liniju kako bi se oslobodilo mjesto potrebnim podacima. Potrebno je odrediti algoritam ili strategiju koja će se linija odabrati za zamjenu, odnosno za izbacivanje iz brze memorije.
Najjednostavniji pristup je slučajan odabir linije koja će se zamijeniti. Nepovoljno je ukoliko se za zamjenu odabere linija koja će se u skoroj budućnosti koristiti. Zato bi bilo dobro na određeni način procijeniti koja linija se neće u budućnosti koristiti ili koja će se najkasnije koristiti. Predviđanje se može izvoditi temeljem informacija o korištenju linije u prošlosti. Tako se može za zamjenu odabrati linija koja se u prošlosti najkasnije koristila (*Last recently used*). Sklopovski ovo se rješava na način da se svakoj liniji brze memorije dodijeli jedno brojilo koje se inkrementira npr. svaku ms, a postavi se u nulu prilikom pristupa liniji. Ona linija čije brojilo ima najveću vrijednost se odabire za zamjenu.