Memorijski sistem

Vladimir Filipović
vladaf@matf.bg.ac.yu

Osnovni pojmovi

Programi i podaci nad kojima se radi se čuvaju u memoriji računara.

U idealnom slučaju, memorija bi bila brza, velika i jeftina. Na nesreću, nemoguće je simultano ispuniti sva tri ova zahteva. U proteklim decenijama, mnogo rada je uloženo u razvoj pametnih struktura koje poboljšati brzinu i kapacitet memorije, a ipak cenu memorije ostaviti na razumnoj meri.

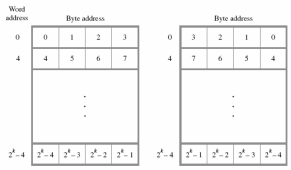
Prvo ćemo opisati najuobičajenije komponente i organizovanja koja se koriste u implementiranju memorije. Potom ćemo ispitati brzinu memorije i razmotriti kako se brzina memorije može povećati korišćenjem keša. Zatim predstavljamo koncept virtuelne memorije koji na izgled uvećava veličinu memorije. Na kraju, razmatramo i uređaje za sekundarnu memoriju, koji omogućuju mnogo veći kapacitet memorije.

Osnovni pojmovi

Za svaki konkretni računar, maksimalna veličina memorije određena je shemom adresiranja. Na primer, 16-bitni računar - koji generiše 16-bitne adrese je u stanju da adresira do 2^{16}=64K memorijskih lokacija. Slično tome, računari čije instrukcije generišu 32-bitne adrese mogu koristiti memoriju koja sadrži do 4G memorijskih lokacija, dok računari sa 40-bitnim adresama mogu pristupiti do 1T lokacija. Broj lokacija predstavlja veličinu adresnog prostora računara.

Najveći broj modernih računara ima bajt-adresibilnu memoriju. Sledeći dijagram pokazuje moguća postavljanja adresa kod bajt-adresibilnog 32-bitnog računara.

68000 – big endian
Intel – little endian
ARM – oboje, po potrebi



(a) Big-endian assignment (b) Little-endian assignment

Osnovni pojmovi

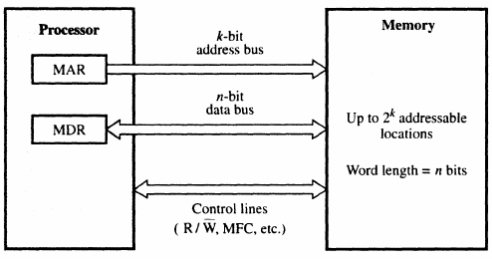
Memorija se obično dizajnira tako da se podaci dobijaju i čuvaju u bitovnim sekvencama čija je dužina u stvari dužina reči. Obično se dužina reči kod računara definiše kao broj bitova kojima se pristupa ili koji bivaju sačuvani tokom jednog pristupa memoriji.

Razmotrimo na primer bajt-adresabilni računar čije instrukcije generišu 32-bitne adrese. Kada je 32-bitna adresa poslata od strane procesora memorijskoj jedinici, viših 30 bitova određuje kojoj reči će se pristupiti. Ako je specificirano da se radi s bitovima, tada dva bita najmanje težine u adresi specificiraju koji će bajt biti uključen. Ako se radi o operaciji čitanja bajta, mogu se dohvatiti i drugi bajtovi prilikom čitanja memorije, ali njih će procesor ignorisati. Međutim, ako se radi o operaciji upisa, memorijska kontrolna kola moraju obezbediti da sadržaj ostalih bitova te iste reči nije promenjen.

Moderne implementacije memorije računara su veoma kompleksne i teške za shvatanje pri prvom upoznavanju. Da bi uprostili uvod u memorijske strukture, prvo će biti predstavljena tradicionalna arhitektura.

Sa tačke gledišta sistema, memorija se može posmatrati kao crna kutija. Prenos podataka između memorije i procesora se realizuje korišćenjem registara MAR i MDR. Ako je MAR dužine k, a MDR dužine n, to znači da ima 2^k memorijskih lokacija, a da se u svakom ciklusu memorije prenese n bitova.

Osnovni pojmovi



Veza između memorije i procesora

Osnovni pojmovi

U ovom slučaju, prenos se izvršava kroz magistralu procesora, koja sadrži k adresnih linija i n linija podataka. Magistrala takođe sadrži i kontrolne linije, "čitaj/ne piši" tj. R/W i "memorijska funkcija završena" tj. MFC, radi koordinacije prenosa. Pored ovih, mogu biti dodate i druge linije koje indiciraju koliko se bitova prenosi.

Ako operacije čitanja ili upisa pogađaju uzastopne memorijske lokacije, tada se može izvršavati "blokovski transfer", pa se memoriji šalje samo adresa prve lokacije bloka. Značaj ovakve vrste prenosa je detaljnije objašnjen pri opisu keširanja.

Pristup memoriji može biti sinhronizovan korišćenjem časovnika, ili može biti kontrolisan korišćenjem specijalnih signala koji kontrolišu prenos na magistrali, korišćenjem prethodno opisanih signalnih shema magistrale.

Korisna mera za brzinu memorijske jedinice je vreme koje je proteklo između iniciranja operacije i njenog završetka – tj. vreme između signala R/W i MFC. Ta veličina se naziva *vreme memorijskog pristupa*.

Druga važna mera je *vreme memorijskog ciklusa*, što predstavlja minimalno vremensko kašnjenje između dve uzastopne operacije čitanja memorije. Ova druga veličina je obično malo duža od prve.

Osnovni pojmovi

Memorijska jedinica se naziva *memorija sa direktnim pristupom* (RAM) ukoliko se ma kojoj lokaciji može pristupiti radi čitanja ili upisa tokom nekog fiksnog vremenskog intervala i ako dužina tog intervala ne zavisi od adrese lokacije. Ovom definicijom se pravi razlika između ovakve memorije i serijskih ili polu serijskih memorijskih uređaja, kod kojih vreme pristupa memoriji zavisi od adrese, odnosno pozicije podataka.

Osnovna tehnologija za implementaciju memorije koristi poluprovodnička integralna kola. Dostupna poluprovodnička integralna kola imaju širok opseg brzina: vreme memorijskog ciklusa im se kreće od 100ns do ispod 10ns. Kada su se pojavila, krajem 60-tih, ona su bila skuplja od memorije sa feromagnetnim jezgrom koja se dotada koristila. Zbog brzog razvoja VLSI tehnologije, cena poluprovodničkih uređaja je dramatično opala, pa se sada ovakvi uređaji skoro svuda koriste za implementaciju memorije.

Unutrašnja organizacija memorijskih čipova

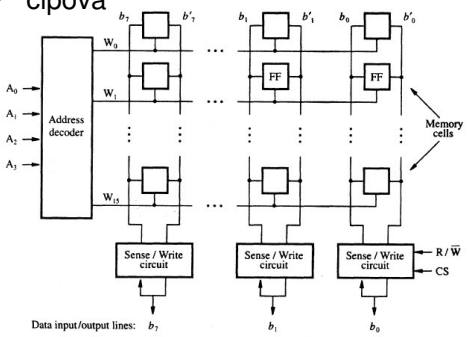
Memorijske ćelije su obično organizovane u formi niza, u kome je svaka ćelija sposobna da čuva jedan bit informacija. Dijagram na sledećem slajdu ilustruje jednu moguću organizaciju.

Svaka vrsta ćelija konstituiše jednu memorijsku reč, pa su ćelije u vrsti povezane na zajedničku liniju, tzv. *liniju reči*, koju aktivira elektronsko kolo dekoder adrese na memorijskom čipu. Sve ćelije u datoj koloni su povezane na logičko kolo Sense/Write preko dve *linije bita*. Logička kola Sense/Write su povezana na ulazno-izlazne linije za podatke koje se nalaze na čipu. Tokom čitanja, Sense/Write kola osluškuju informacije koje su smeštene u ćelijama za koje je aktivirana linija reči i prenose te informacije do izlaznih linija podataka. Tokom operacije pisanja, ova kola prihvataju informacije sa ulaza i smeštaju ih u ćelije izabrane reči.

Dijagram sa sledećeg slajda predstavlja primer veoma malog memorijskog čipa koji se sastoji od 16 reči, a svaka reč od 8 bitova. To se označava kao organizacija 16x8.

Ulazne i izlazne linije svakog Sense/Write kola su povezane sa jedinstvenom dvosmernom linijom koja se može povezati na magistralu podataka računara. Kontrolna linija $R/\overline{W}$ spsecificira zahtevanu operaciju, a ulaz CS (*chip selected*) bira dati čip u memorijskom sistemu sa više čipova.

Unutrašnja organizacija memorijskih čipova



Organizacija bitovnih ćelija na memorijskom čipu

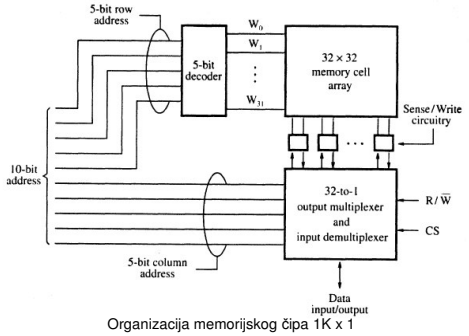
Unutrašnja organizacija memorijskih čipova

Memorijsko kolo opisano u prethodnom dijagramu čuva 128 bitova i zahteva 14 spoljašnjih konekcija za adrese, podatke i kontrolne linije. Naravno, potrebne su i dve linije za struju i uzemljenje.

Razmotrimo sada nešto veće memorijsko kolo, ono koje sadrži 1K memorijskih ćelija. To kolo može biti organizovano kao 128x8 memorija i tada zahteva ukupno 19 spoljašnjih konekcija. Alternativno, isti broj ćelija može biti organizovan u formatu 1Kx1. U tom slučaju, potrebna je desetobitna adresa, ali samo jedna linija za podatke, što rezultuje sa 15 spoljašnjih konekcija. Takva organizacija je prikazana dijagramom na sledećem slajdu. Zahtevana desetobitna adresa je podeljena u dve grupe od po 5 bitova, gde jedna formira adresu vrste, a druga adresu kolone. Adresa vrste bira vrstu od 32 ćelije, gde se svima njima pristupa paralelno. Međutim, u skladu sa adresom kolone, samo jedna od ovih ćelija biva povezana na spoljašnju liniju podataka preko izlaznog multipleksera i ulaznog demultipleksera.

Komercijalni memorijski čipovi sadrže mnogo veći broj memorijskih ćelija nego što je to slučaj u prethodnim primerima. Veliki čipovi suštinski imaju istu organizaciju, samo što koriste veću matricu memorijskih ćelija i imaju više spoljašnjih konekcija. Na primer, 4M čip može imati organizaciju 512Kx8, u kom slučaju je potrebno 19 adresa i 8 ulazno-izlaznih pinova za podatke.

Unutrašnja organizacija memorijskih čipova

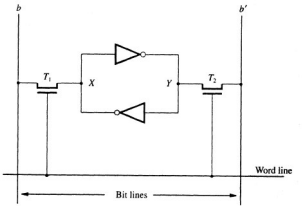


Organizacija memorijskog čipa 1K x 1

Statičke memorije

Memorije koje se sastoje od logičkih kola sposobnih da zadrže svoje stanje dok god ima struje se nazivaju *statičke memorije*. Sledeći dijagram ilustruje kako može biti implementirana statička RAM (tj. SRAM) ćelija. Dva invertora su unakrsno povezani i formiraju bravu (*latch*). Brava je dvema bitovnim linijama povezana sa tranzistorima T_1 i T_2. Ovi tranzistori se ponašaju kao prekidači koji mogu biti otvoreni ili zatvoreni pod kontrolom linije reči. Kad je linija reči na nivou uzemljenja, tranzistori su isključeni i brava zadržava svoje stanje.

Statička RAM memorijska ćelija



Statičke memorije

Operacija čitanja

Da bi se pročitalo stanje SRAM ćelije, linija reči se aktivira kako bi zatvorila prekidače T_1 i T_2. Ako je ćelija u stanju 1, signal na liniji b je visok i signal na liniji b' je nizak. U slučaju kada je ćelija u stanju 0, dešava se suprotno. Stoga b i b' su komplementarne jedna drugoj. Logička kola Sense/Write na kraju linije bitova nadgledaju stanja b i b' i postavljaju izlaz na odgovarajući način.

Operacija upisa

Stanje ćelije se postavlja tako što se odgovarajuća vrednost postavi na bit liniju b, a njen komplement na liniju b', a potom se aktivira linija reči. Ovo prebacuje ćeliju u odgovarajuće stanje. Zahtevane signale na bitovnim linijama su generisane od strane Sense/Write logičkih kola.

CMOS ćelije

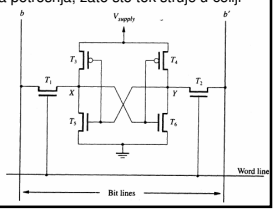
Dijagram koji sledi predstavlja CMOS realizaciju ćelije sa prethodnog dijagrama. Parovi tranzistora T_3 i T_5, odnosno T_4 i T_6 formiraju invertore u bravi. Stanje ćelije se čita ili upisuje na prethodno opisan način.

Nivo napona V_{supply} je 5V kod starijih CMOS SRAM-ova ili 3.3V kod novijih verzija niže voltaže. Uočavamo da je potreba neprekidan tok struje da bi ćelija zadržala stanje – ako se struja prekine, sadržaj ćelije će se izgubiti. Dakle, SRAM je *ranjiva* memorija.

Glavna prednost CMOS SRAM je veoma mala potrošnja, zato što tok struje u ćeliji postoji samo kada se pristupa ćeliji.

CMOS SRAM je vrlo brz – reda nekoliko ns.

Primer CMOS memorijske ćelije



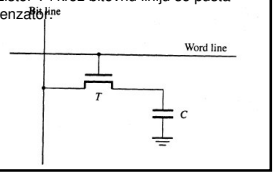
Asinhroni DRAM

Statički RAM je brz, ali skup – jer jedna njegova ćelija zahteva nekoliko tranzistora. Ako bi se koristile jednostavnije ćelije, mogao bi se implementirati jeftiniji RAM. Međutim, takav RAM nije u stanju da neograničeno dugo zadržava svoje stanje. Stoga se on zove *Dinamički RAM* (DRAM).

Informacija se čuva u dinamičkoj memorijskoj ćeliji u obliku struje u kondenzatoru, i ta struja može biti održavana samo tokom desetak milisekundi. Stoga njen sadržaj mora biti periodično osvežavan, tako što se kapacitet kondenzatora restauriše do njegove pune vrednosti.

Sledeći dijagram predstavlja primer dinamičke memorije. Da bi se smestila informacija u ovu ćeliju, uključuje se tranzistor T i kroz bitovnu liniju se pušta odgovarajuća voltaža, čime se puni kondenzator.

Dinamička memorijska ćelija od jednog tranzistora



Asinhroni DRAM

Po isključenju tranzistora, kondenzator počinje da se prazni. Ovo je prouzrokovano smanjenjem otpornosti kod samog kondenzatora i činjenicom da tranzistor, čak i kada je isključen, nastavlja da provodi malu količinu struje. Stoga, informacija koja je smeštena u ćeliju može da se korektno čita samo ako je očitana pre nego što se struja u kondenzatoru spusti ispod neke granične vrednosti.

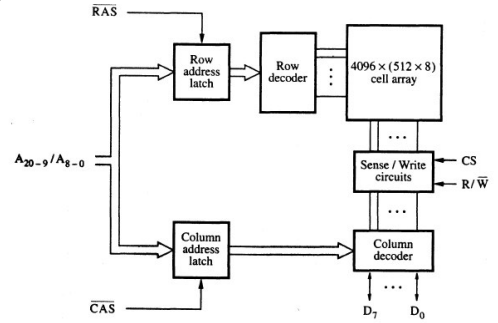
Tokom operacije čitanja, tranzistor u izabranoj ćeliji biva uključen. Osluškivač povezan na liniju bita detektuje da li je nivo struje u kondenzatoru iznad granične vrednosti. Ako jeste, on dovodi liniju bita na puni napon, koji predtavlja jedinicu. Ovaj napon ponovo puni kondenzator, tako da on predstavlja logičku jedinicu. Ako osluškivač detektuje da je nivo u kondenzatoru ispod granične vrednosti, on spušta nivo napona na liniji bita na nulti nivo, što prazni kondenzator, a to predstavlja logičku nulu. Na ovaj način, čitanjem sadržaja taj sadržaj automatski biva osvežen. Sve ćelije u izabranoj vrsti se čitaju istovremeno, i sadržaj cele vrste biva osvežen.

Asinhroni DRAM

Dijagram na sledećem slajdu prikazuje 16M bitni DRAM čip, konfigurisan kao 2Mx8. Ćelije su organizovane kao matrica 4Kx4K. 4096 ćelija u svakoj od vrsta su podeljene u 512 grupa po 8, tako da svaka vrsta može da čuva 512 bajtova podataka. Dakle, 12 linija je potrebno za izbor vrste. Nadalje, potrebno je još 9 bitova da bi se specificirala grupa od 8 bitova u izabranoj vrsti. Prema tome, potrebna je 21-bitna adresa za pristup bajtu u memoriji. Viših 12 bitova i nižih 9 bitova predstavljaće adresu vrste i adresu kolone za traženi bajt.

Tokom operacije čitanja ili upisa, prvo se radi sa adresom vrste. Ona se, kao odgovor na ulazni signalni puls RAS (*Row Access Strobe*) sa čipa, učita u bravu za adresu vrste. Potom se inicira operacija čitanja, pri čemu se sve ćelije u izabranoj vrsti pročitaju i osveže. Kratko po učitavanju adrese vrste, u bravu za adresu kolone su učita adresa kolone – to je aktivirano kontrolnim signalom CAS (*Column Access Strobe*) sa čipa. Potom se dekodira informacija iz ove brave i aktivira se izabrana grupa od 8 Sense/Write elektronskih kola. Ako signal R/W ukazuje na operaciju čitanja, izlazne vrednosti izabranih kola se prenose na linije podataka D_{7-0}. Za operaciju pisanja, informacije sa linija D_{7-0} se prenose u izabrana kola, a potom se te informacije koriste kako bi "pregazile" sadržaj izabranih ćelija u 8 odgovarajućih kolona. Signali RAS i CAS u komercijalnim procesorima imaju invertovanu shemu reprezentacije – RAS i CAS.

Asinhroni DRAM



Interna organizacija dinamički memorijski čip 2M x 8

Asinhroni DRAM

Postavljanje adrese vrste dovodi do toga da sve ćelije u odgovarajućem redu budu pročitane i osvežene i tokom čitanja i tokom upisa. Da bi obezbedili da sadržaj DRAM bude korektan, svakoj vrsti se mora periodično pristupati. *Kola za osvežavanje* se obično koriste u tu svrhu. Mnogi DRAM čipovi u okviru sebe već sadrže komponente za osvežavanje i dinamička priroda takvih čipova biva skoro nevidljiva korisniku.

Kod prethodno opisanog DRAM-a, vremenski sled za ovaj memorijski uređaj je asinhrono kontrolisan, a vremenima upravljaju signali RAS i CAS, koje obezbeđuje specijalizovani kontroler memorije. Ovakva memorija se zato često označava kao *asinhroni DRAM*.

DRAM se široko koristi zato što obezbeđuje veliku gustinu podataka i nisku cenu.

Mod brzih stranica

Jednostavnom modifikacijom čipa opisanog prethodnim dijagramom, može se postići da se pristupa i drugim bitovima u vrsti, bez potrebe da se ponovo vrši selekcija vrste. Na izlaz osluškivača kod svake kolone može da se doda brava. Sada se, prilikom primene adrese vrste, novododate brave napune sadržajem bitova izabrane vrste. Potom samo treba postavljati različite adrese kolona kako bi se dobili različiti podaci u linijama podataka.

Asinhroni DRAM

Kod ovih DRAM, uobičajen period za osvežavanje svih vrsta je 16ms.

Najkorisnija primena moda brzih stranica je prenos bajtova u sekvencijalnom redosledu, koji se postiže primenom sekvence uzastopnih adresa kolona, pod kontrolom sukcesivnih CAS signala. Ova shema omogućuje prenos bloka podataka mnogo većom brzinom nego što bi se postiglo kod slučajnih adresa. Sposobnost blokovskog transfera se još naziva i karakteristika *moda brzih stranica*. (U žargonu se manja grupa bitova često naziva blok, a veća grupa kao stranica).

Veća brzina blokovskog transfera može biti eksploatisana u aplikacijama u kojim memorijski pristup prati regularne obrasce, kao što je to slučaj kod grafičkih terminala. Ova karakteristika je takođe korisna i kod računara opšte namene pri prenosu blokova podataka između glavne memorije i keša, o čemu će kasnije biti više reči.

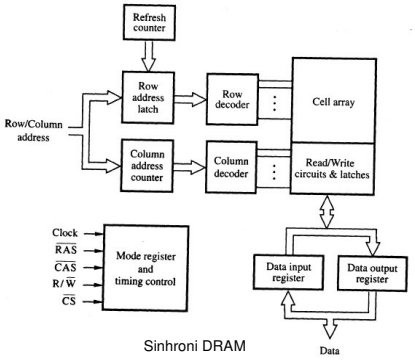
Sinhronizovani DRAM

Noviji tehnološki razvoj kod memorije je doveo do kreiranja DRAM-ova čiji je rad direktno sinhronizovan sa signalom časovnika sistemske magistrale. Takve memorije su poznate kao *sinhronizovani DRAM*, tj. SDRAM. Dijagram na sledećem slajdu ukazuje na strukturu SDRAM-a.

Matrica ćelija je ista kao kod asinhronog DRAM-a. Adresne konekcije i konekcije podataka su baferisane korišćenjem registara. Naročito ističemo da je izlaz svakog osluškivača povezan na bravu. Operacija čitanja dovodi do toga da sadržaj svih ćelija u izabranoj vrsti bude učitan u te brave. Ali, ako se pristup vrši samo radi osvežavanja, neće se menjati sadržaj tih brava; isključivo će se osvežavati sadržaj ćelija. Podaci koji se čuvaju u bravama koje odgovaraju izabranoj koloni se prenose u registar za izlaz podataka i tako bivaju dostupni na izlaznim pinovima za podatke.

SDRAM ima nekoliko različitih modova rada, koji se mogu izabrati upisom kontrolne informacije u *registar modova*. Na primer, mogu se specificirati rafalne operacije različite dužine. Rafalna operacija koristi blokovski transfer i mod brzih stranica. Pri tome, kod SDRAM nije potrebno obezbeđivati spolja generisani puls na CAS liniju za izbor uzastopnih kolona. Potrebni kontrolni signali se obezbeđuju interno, korišćenjem brojača kolona i signala časovnika, pa se novi podaci smeštaju na liniju za podatke pri svakom ciklusu časovnika (sinhronizovano na podignutu ivicu časovnika).

Sinhronizovani DRAM



Sinhroni DRAM

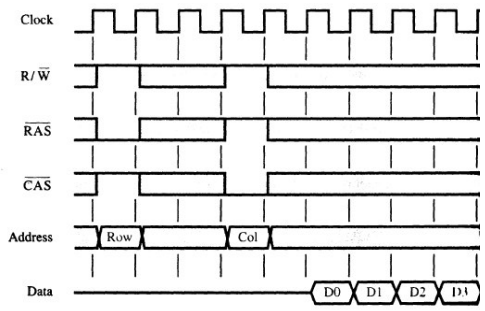
Sinhronizovani DRAM

Dijagram na sledećem slajdu prikazuje vremensku shemu za tipično rafalno čitanje dužine 4. Prvo se adresa vrste smesti u brave, pod kontrolom signala $\overline{RAS}$. Memoriji obično treba 2 do 3 ciklusa (na dijagramu su prikazana 2 ciklusa) da bi se aktivirala izabrana vrsta. Potom se adresa kolone smesti u odgovarajuće brave, pod kontrolom CAS signala. Posle kašnjenja od koje traje jedan ciklus časovnika, prvi skup bitova podataka biva smešten na linije za podatke. SDRAM automatski uvećava adresu kolone, kako bi pristupio sledećim trima skupovima bitova u izabranoj vrsti, i ti skupovi bitova se smeštaju na linije podataka u sledeća tri ciklusa časovnika.

SDRAM poseduje ugrađena kola za osvežavanje. Deo kola za osvežavanje je i brojač osvežavanja, koji obezbeđuje adrese za vrste koje su izabrane za osvežavanje. U tipičnoj SDRAM memoriji, svaka vrsta mora biti osvežena najduže u svakih 64 ms.

Komercijalni SDRAM radi na brzini od preko 100MHz. SDRAM čipovi su dizajnirani tako da ispune zahteva komercijalno dostupnih široko korišćenih procesora. Na primer, Intel je definisao specifikacije sistemske magistrale (na koju se povezuje memorija) PC100 i PC133 koje su kontrolisane od strane časovnika sa frekvencijom 100MHz i 133MHz respektivno. Stoga, glavni proizvođači memorijskih čipova proizvode 100MHz i 133MHz SDRAM čipove.

Sinhronizovani DRAM



Rafalno čitanje dužine 4 u SDRAM-u

Sinhronizovani DRAM

Kašnjenje i propusni opseg

Termin *memorijsko kašnjenje* se koristi da bi označio količinu vremena koja je potrebna za prenos jedne reči podataka u/iz memorije. U slučaju čitanja ili upisa jedne reči podataka, kašnjenje obezbeđuje kompletnu indikaciju memorijskih performansi.

Kod rafalnih operacija, gde se prenose blokovo podataka, vreme potrebno za završetak operacije još zavisi i od brzine kojom se mogu prenositi uzastopne reči, kao i od veličine bloka. U tom slučaju termin kašnjenje se koristi radi označavanja vremena potrebnog za prenos prvog bloka podataka – a to vreme je obično duže od vremena potrebnog za prenos podataka koji slede (što se može videti i sa prethodnog dijagrama).

Pri merenju brzine prenosa podataka, dobro je znati i koliko je vremena potrebno za prenos celog bloka. Kako veličina blokova može varirati, korisno je da se definiše mera performansi kao broj bitova koji mogu biti preneseni u jednoj sekundi. Ta mera se često označava kao *propusni opseg memorije*. Na njega utiče brzina prenosa i broj bitova kojima se može pristupiti paralelno.Efektivni propusni opseg računarskog sistema (što uključuje prenos podataka između memorije i procesora) ne zavisi samo od memorije – on zavisi i od mogućnosti za prenos veza koje povezuju memoriju i procesor, što se obično svede na brzinu magistrale.

Sinhronizovani DRAM

SDRAM sa dvostrukom brzinom

Standardni SDRAM izvršava sve operacije na podižućoj ivici signala časovnika. Kreiran je i sličan uređaj, koji pristupa matrici ćelija na isti način, ali prenosi podatke na obema ivicama signala: podižućoj i padajućoj. Kašnjenje ovog uređaja je isto kao kod standardnog SDRAM. Ali, budući da on prenos podatke na obema ivicama, to je njegov propusni opseg dupliran u slučaju rafalnih prenosa.

Takav uređaj se naziva SDRAM sa dvostrukom brzinom, (*Double Data Rate SDRAM - DDR SDRAM*).

Da bi se omogućilo da se podacima pristupi dovoljno velikom brzinom, matrica ćelija se organizuje u dve banke. Svakoj banci se odvojeno pristupa. Uzastopne reči datog bloka se smeštaju u različitim bankama. Takvo *preplitanje* reči dopušta simultani pristup dvema rečima koje se prenose na uzastopnim ivicama časovnika, što će kasnije biti detaljnije opisano.

DDRSDRAM i standardni SDRAM su najefikasnije korišćeni u aplikacijama gde dominira blokovski prenos, a to je slučaj i sa računarima opšte namene.

Struktura većih memorija

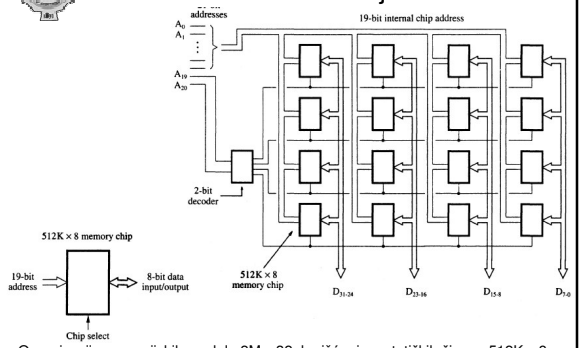
Do sada smo posmatrali osnovnu organizaciju memorijskih kola, kao da će se ona implementirati na jednom čipu. Sada ćemo ispitati kako se čipovi mogu povezati da bi oformili mnogo veću memoriju.

Sistemi sa statičkom memorijom

Razmotrimo memorijski sistem sa 2M reči dužine 32 bita. Sledeći dijagram pokazuje kako se takva memorija može implementirati korišćenjem 512Kx8 statičkih memorijskih čipova. Svaka od kolona na slici sadrži po četiri čipa, gde jedan čip implementira jednu bajt-pozicija. Četiri takva skupa obezbeđuju zahtevanu 2Mx32 memoriju. Svaki čip ima kontrolni ulaz nazvan Chip Select, tj. CS. Kada je CS postavljeno na 1, čip je osposobljen za prijem podataka sa linije ili smeštaj podataka na liniju. Izlaz od svakog čipa je tip sa tri stanja. Stoga, samo izabrani čip smešta podatke na izlazne linije za podatke, dok ostali smeštaju na linju sa visokom impendansom.

Da bi se izabrala 32-bitna reč iz memorije, potrebna su 21 adresna bita. Dva najviša bita adrese se dekodiraju da bi se odredilo koji se CS signal aktivira, a preostalih 19 bitova se koristi za pristup konkretnoj lokaciji unutar svakog od čipova izabrane vrste.

Struktura većih memorija



Organizacija memorijskih modula 2M x 32, korišćenjem statičkih čipova 512K x 8

Struktura većih memorija

Sistemi sa dinamičkom memorijom

Organizacija velikih dinamičkih memorijskih sistema je esencijalno ista kao na prethodnom dijagramu. Međutim fizička implementacija je mnogo pogodnije izvedena - u obliku *memorijskih modula*.

Današnja memorija je jako velika. Ako bi se jako velika memorija direktno stavila na matičnu ploču, ona bi zauzela neprihvatljivo veliki deo ploče. Nadalje, bilo bi dobro da se omogući dalje proširenje memorije u budućnosti.

Ovo je dovelo do razvoja velikih memorijskih jedinica, nazvanih SIMM (*Single In-Line Memory Modul*) i DIMM (*Double In-Line Memory Modul*).

Ti moduli predstavljaju sklop nekoliko memorijskih čipova koji se nalaze na odvojenoj memorijskoj ploči, a ta ploča se vertikalno priključuje preko jedinstvenog priključnog sklopa (*socket*) na matičnoj ploči. SIMM-ovi i DIMM-ovi različitih veličina su dizajnirani tako da koriste iste priključne sklopove. Tako, na primer, 4Mx32, 16Mx32 i 32Mx32 soketi koriste isti 100-pinski priključni sklop. Slično tome, 8Mx64, 16Mx64, 32Mx64 i 64Mx72 DIMM-ovi koriste priključni sklop od 168 pinova.

Ovakvi moduli zauzimaju manji prostor na matičnoj ploči i olakšavaju proširenje memorije ukoliko veći modul koristi istu vrstu priključnog sklopa koju je koristio manji modul.

Razmatranja memorijskih sistema

Izbor RAM čipa za datu primenu zavisi od nekoliko faktora. Najznačajniji među ovim faktorima su cena, brzina, trošenje energije i veličina čipa.

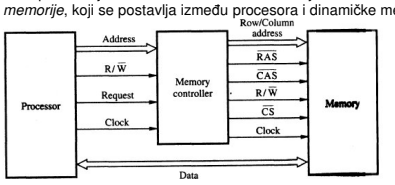
Statički RAM se u opštem slučaju koristi samo kada je primarni zahtev koji se postavlja pred memoriju veoma brz rad, tj. veoma brzo izvršenje operacija. Na cenu i veličinu statičkog RAM-a dominantno utiče složenost elektronskih kola kojima se realizuje osnovna ćelija. Ova memorija se najčešće koristi u keš memorijama.

Dinamički RAM je preovladavajući izbor za implementaciju glavne memorije računara. Činjenica da su kod ovih čipova postignute velike gustine smeštaja čine da su velike memorije oformljene na ovakav način ekonomski veoma povoljne.

Kontroler memorije

Čipovi sa dinamičkom memorijom koriste multipleksirane adresne ulaze i na taj način smanjuju broj pinova. U tom slučaju, adresa se deli u dva dela. Adresni bitovi višeg reda, tj. oni bitovi koji biraju vrstu u matrici ćelija, se prvi obezbede i oni se smeste u bravu memorijskog čipa pod kontrolom RAS signala. Potom se bitovi manje težine, oni koji biraju kolonu, dovode na iste adresne pinove i smeštaju se u bravu pod kontrolom signala CAS.

Procesor obično istovremeno prosledi sve bitove adrese na adresne linije magistrale. Zahtevano multipleksiranje adresnih bitova obično realizuju elektronska kola *kontrolera memorije*, koji se postavlja između procesora i dinamičke memorije.



Korišćenje kontrolera memorije

Kontroler memorije

Kontroler prihvata kompletnu adresu i signal R/$\overline{W}$ od procesora, pod kontrolom signala *Request*, koji ukazuje da je potrebno izvršiti memorijsku operaciju. Kontroler potom prosleđuje memoriji delove adrese koji predstavljaju vrstu i kolonu i generiše signale $\overline{RAS}$ i $\overline{CAS}$. Na taj način, kontroler (pored funkcije multipleksiranja) obezbeđuje RAS-CAS vremensko podešavanje signala. Kontroler takođe i šalje R/W i CS signale u memoriju. Kako je kod CS signala aktivnost obično označena niskim nivoom napona, to je on na prethodnom dijagramu označen sa $\overline{CS}$. Linije za podatke direktno povezuju procesor i memoriju. Uočavamo da je za SDRAM čipove neophodan i signal časovnika magistrale.

Kada se radi sa DRAM čipovima koji nemaju sposobnost samoosvežavanja, kontroler memorije mora da obezbedi sve informacije potrebne za kontrolu procesa osvežavanja. U tom slučaju, kontroler sadrži brojač osvežavanja koji prosleđuje uzastopne lokacije vrsta. Njegova funkcija je da prouzrokuje da osvežavanje svih vrsta bude završeno tokom perioda koji je karakterističan za određeni uređaj.

Kašnjenje zbog osvežavanja

Sve dinamičke memorije moraju da se osvežavaju. Kod starijeg DRAM-a, uobičajen period za osvežavanje svih vrsta je bio 16 ms. Kod tipičnih SDRAM-ova, uobičajen period je 64 ms.

Razmotrimo SDRAM čije su ćelije organizovane u matricu sa 8K (=8192) vrste. Pretpostavimo da su potrebna četiri ciklusa da bi se pristupilo svakoj vrsti. Stoga, potrebno je 8192 x 4 = 32768 ciklusa kako bi se osvežile sve vrste. Ako je frekfenca časovnika 133 MHz, vreme potrebno za osvežavanje svih vrsta iznosi $32768/(133x10^{-6})$ sekundi. Prema tome, osvežavanje zauzima 0,246 ms u svakom vremenskom intervalu od 64 ms. Dakle, kašnjenje zbog osvežavanja je 0,246/64=0,0038 – što znači da se nešto manje od 0,4% ukupnog vremena kada procesor može pristupiti memoriji troši za osvežavanje memorije.

Memorije samo za čitanje

I statičke i dinamičke memorije su ranjive (*volatile*), što znači da će u slučaju nestanka napajanja izgubiti podatke koji su smešteni u njima. Postoji mnogo primena koje zahtevaju memorijske uređaje sposobne da zadrže informacije koje čuvaju i kada nestane napajanje.

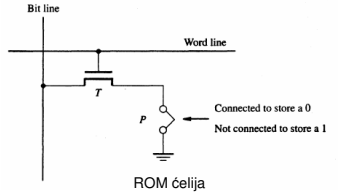
Na primer, kod tipičnog računara se koristi hard disk za smeštaj velike količine informacija, uključujući i operativni sistem. Kada se uključi računar, operativni sistem mora da bude učitan sa diska u memoriju. Ovo zahteva da se izvrši program koji "podiže" operativni sistem. Kako je taj program veliki, to se njegov najveći deo nalazi na disku. Procesor mora da izvrši neke instrukcije koje učitavaju program za podizanje u memoriju. Ako bi se cela memorija sastojala samo od ranjivih memorijskih čipova, procesor ne bi imao načina da pristupi tim instrukcijama. Praktično rešenje je da se obezbedi mala količina neranjive memorije koja čuva instrukcije čijim izvršavanjem se program za podizanje učita sa diska.

Neranjiva memorija se mnogo koristi u umetnutim sistemima, koji obično nemaju hard diskove, pa su programi smešteni na neranjivu poluprovodničku memoriju.

Ova memorija može da se čita na isti način kao SRAM ili DRAM memorija. Međutim, da bi se smestila informacija u memoriju, potrebno je realizovati specijalan proces upisivanja. Kako njihovo normalno operisanje uključuje samo čitanje smeštenih podataka, ova memorija se naziva *memorija samo za čitanje* (ROM).

ROM

Dijagram jasno pokazuje moguću konfiguraciju ROM ćelije. Logička vrednost nula je smeštena u ćeliju ukoliko je na tački P tranzistor povezan sa uzemljenjem. U suprotnom, u ćeliju je smeštena vrednost 1. Linija bita je povezana preko otpornika na izvor napajanja. Da bi se pročitalo stanje ćelije, aktivira se linija reči. Ako postoji veza sa uzemljenjem, tranzistor se zatvori i napon na liniji bita padne blizu 0. Ako nema veze sa uzemljenjem, linija bita ostane pod visokim naponom, što ukazuje na jedinicu. Elektronska kola za osluškivanje na kraju linije bita generišu odgovarajući izlazni napon. Podaci se u ROM upisuju prilikom proizvodnje.



ROM ćelija

PROM

Neke vrste ROM-a dopuštaju da korisnik unese podatke, i na taj način obezbeđuju *programabilni ROM* (PROM). Programabilnost se postiže ubacivanjem osigurača na tačku P sa prethodnog dijagrama. Pre programiranja memorija sadrži sve nule. Korisnik može postaviti jedinicu na konkretnu lokaciju tako što će impulsom jake struje istopiti osigurač na odgovarajućoj poziciji. Naravno, ovaj proces je nepovratan.

PROM-ovi obezbeđuju fleksibilnost i udobnost u radu koja sa ROM-ovima nije bila dostupna. ROM je ekonomski atraktivnija opcija za smeštaj fiksnih programa i podataka u situacijama kada se proizvodi veliki broj memorijskih jedinica. Međutim, cena pripreme maski koja su neophodne za smeštaj konkretnog obrasca informacija u ROM čini tu memoriju veoma skupom u slučaju kada je potreban mali broj čipova. Dakle, u takvim slučajevima PROM predstavlja brži i jeftiniji pristup, jer njega može programirati i korisnik.

EPROM

Druga vrsta ROM čipa dopušta da smešteni podaci budu obrisani i da budu uneseni novi podaci. Takav obrisivi, reprogramibilni ROM naziva se *EPROM*. On obezbeđuje povećanu fleksibilnost tokom faze razvoja digitalnog sistema. Budući da je EPROM sposoban da tokom dugo vremena zadrži informacije koje su smeštene u njega, on se tokom razvoja softvera može koristiti umesto ROM-a. Na ovaj način se mogu lako vršiti promene i ažuriranja u memoriji.

Struktura EPROM ćelije je slična strukturi ROM ćelije sa prethodnog dijagrama. Međutim, kod EPROM ćelije na tački P uvek postoji uzemljenje, a koristi se specijalni tranzistor, koji ima mogućnost da radi bilo kao normalan tranzistor ili kao onemogućeni tranzistor (koji je uvek isključen). Ovaj specijalni tranzistor može biti programiran da se ponaša kao otvoreni prekidač, tako što se u njega ubaci struja koja ostaje zarobljena unutar tog tranzistora. Dakle, EPROM može da se koristi za konstrukciju memorije na isti način kao i prethodno opisana ROM ćelija.

Važna prednost EPROM čipova je to što njihov sadržaj može biti obrisan i reprogramiran. Brisanje zahteva rasipanje struje koja je zarobljena unutar tranzistora; ovo se može postići tako što se čip izloži ultraljubičastom zračenju. Stoga se EPROM čipovi montiraju u paketima koji imaju prozirne prozore.

EEPROM

Značajan nedostatak EPROM-a je činjenica da je za reprogramiranje čipa potrebno da se on fizički izvadi i izloži ultraljubičastom zračenju. Moguće je implementirati još jednu verziju izbrisivog PROM-a koji može biti i programiran i izbrisan uz pomoć struje. Takvi čipovi, nazvani EEPROM-i, ne moraju da se vade da bi bili obrisani. Nadalje, moguće je selektivno brisati sadržaj pojedinih ćelija.

Jedina mana ove memorije u odnosu na EPROM je da su potrebni različiti nivoi napona za brisanje, za upis i za čitanje smeštenog podatka.

Fleš memorija

Pristup koji je sličan kao kod EEPROM tehnologije je u poslednje vreme doveo do pojave uređaja sa fleš memorijom. Fleš ćelija je zasnovana na jednom tranzistoru koji kontroliše napon zarobljen u njemu, baš kao što je slučaj sa EEPROM ćelijom. Dok je kod EEPROM-a moguće čitati i pisati u jednu ćeliju, kod fleš memorije je moguće čitati sadržaj jedne ćelije, ali je samo moguće upisati ceo blok podataka. Pre upisa podataka, dotadašnji sadržaj celokupnog bloka se briše.

Fleš uređaji imaju veću gustinu, što dovodi do većeg kapaciteta i niže cene po bitu. Fleš uređaji troše manje struje za svoj rad.

Niska potrošnja struje čini ovu vrstu memorije veoma privlačnom za male uređaja koji rade na baterije. Tipična primena uključuje PDA-ove, mobilne telefone, digitalne kamere i MP3 plejere. Svi prethodno pomenuti uređaji predstavljaju dobar primer umetnutih sistema.

Jedan fleš čip ne sadrži dovoljno prostora za korišćenje u prethodno pobrojanim situacijama, već su nam za tu svrhu potrebni veći memorijski moduli koji se sastoje od većeg broja čipova.

Postoje dva popularna izbora za implementaciju takvih modula: fleš kartice i fleš uređaji.

Fleš kartice

Jedan način za kreiranje većih modula je montiranje fleš čipova na maloj kartici.

Takva fleš kartica sadrži standardni interfejs koji je čini korisnom u različitim proizvodima. Ona se stoga jednostavno ubaci u odgovarajući slot za pristup.

Fleš kartice se proizvode u većem broju memorijskih veličina. Ako se koristi MP3 format, minut muzike zauzima otprilike 1Mb memorije. Sada je lako proceniti koliko muzike može da se smesti na konkretan MP3 plejer sa memorijom datog kapaciteta.

Fleš uređaji

Razvijeni su i veći memorijski moduli kako bi zamenili hard diskove. Takvi fleš uređaji su dizajnirani tako da u potpunosti emuliraju hard diskove. Međutim, kapacitet fleš uređaja je značajno niži.

Činjenica da fleš uređaji nemaju pokretnih delova obezbeđuje neke važne prednosti. Ovakvi uređaji imaju kraća vremena traženja i vremena pristupa, što rezultuje bržim odgovorom. Oni imaju manju potrošnju struje, što ih čini pogodnim za primenu u uređajima na baterije. Nadalje, oni su neosetljivi na vibracije.

Nedostatci u poređenju sa hard diskovima su njihov manji kapacitet i viša cena po bitu. Naime, hard diskovi imaju ekstremno nisku cenu po bitu.

Još jedan nedostatak je to što se fleš memorija kvar ako se u nju piše određeni broj puta. Na sreću, taj broj je veliki – ne manji od milion.

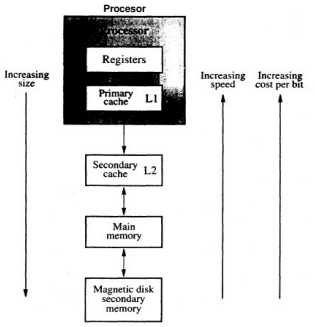
Brzina, veličina i cena

Idealna memorija treba da bude brza velika i jeftina.

Iz prethodnih razmatranja je jasno da se veoma brza memorija može implementirati pomoću SRAM čipova. Ali, zbog cene je nepraktično da se velika memorija gradi korišćenjem SRAM čipova.

Alternativa je korišćenje dinamičkih DRAM čipova, sa jednostavnijim osnovnim ćelijama, sa manjom cenom i manjom brzinom.

Veoma veliki diskovi su dostupni po povoljnoj ceni, i ovi se u veoma velikoj meri koriste u računarima Međutim, oni su mnogo sporiji od poluprovodničkih memorija.

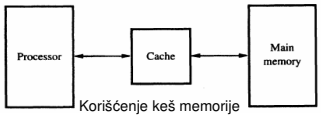


Hijerarhija memorija

Keš memorije

Veoma je važno da se razvije shema koja smanjuje vreme potrebno procesoru za pristup potrebnim informacijama. Kako je brzina glavne memorije ograničena elektronskim ograničenjima i ograničenjima pakovanja, rešenje je nađeno u arhitektonskim podešavanjima. To rešenje je korišćenje brze *keš memorije* koja omogućuje da procesoru glavna memorija izgleda bržom nego što u stvari jeste.

Efikasnost mehanizma keširanja je zasnovana na osobini računarskih programa koja je nazvana *lokalnost referenci*. Lokalnost referenci se manifestuje u dva oblika: vremenskom i prostornom. Prvo znači da će nedavno izvršene instrukcije vrlo verovatno biti ponovo izvršavane. Prostorna lokalnost znači da će instrukcije čije su adrese blizu adresa nedavno izvršenih instrukcija vrlo verovatno biti ubrzo izvršavane. Lokalnost referenci se odnosi i na instrukcije i na programe nad kojima se instrukcije izvršavaju.



Korišćenje keš memorije

Keš memorije

Dakle, ako se aktivni segmenti programa mogu smestiti u brzu keš memoriju, tada se ukupno vreme izvršavanja može značajno skratiti.

Konceptualno, rad keš memorije je vrlo jednostavan i prikazan je na prethodnom dijagramu.

Za referisanje na niz uzastopnih memorijskih lokacija iste veličine koristićemo termin *blok*. Još jedan termin koji se koristi za označavanje ovog pojma je *keš linija*.

Obično keš memorija može sadržavati veći broj blokova u datom trenutku, ali taj broj je mali u odnosu na ukupan broj blokova u glavnoj memoriji. Veza između blokova u glavnoj memoriji i blokova u kešu data je *funkcijom mapiranja*. Kada je keš pun a referiše se na memorijsku reč (instrukciju ili podatak) koja nije u kešu, tada kontrolni hardver keša mora da odredi koji će blok da bude uklonjen, kako bi se kreirao prostor za novi blok koji sadrži memorijsku reč na koju se referiše. Skup pravila za donošenje ove odluke oformljuje *algoritam zamene*.

Procesor ne mora eksplicitno da zna o postojanju keša. On samo izdaje zahteve za čitanje i za upis koji referišu na lokacije u memoriji. Kontrolna elektronska kola u kešu određuju da li zahtevana reč postoji u kešu ili ne. Ako je reč u kešu, tada se operacije čitanja i pisanja obavljaju nad odgovarajućom lokacijom keša. U tom slučaju, došlo je do *pogotka pri čitanju* ili *pogotka pri pisanju*, zavisno od toga koja se operacija izvršava.

Keš memorije

Pri pogotku pri čitanju, glavna memorija ne biva uključena u aktivnosti. Međutim, ako se radilo o pogotku pri upisivanju, sistem može da nastavi sa radom na dva načina. Kod prve tehnike, nazvane *protokol piši-kroz* (*write-through*), lokacija u keš memoriji i lokacija u glavnoj memoriji se simultano ažuriraju. Druga tehnika ažurira samo lokaciju u kešu i označava je kao ažuriranu korišćenjem pridruženog fleg-bita, nazvanog *bit prljavosti* ili *bit modifikacije*. Lokacija te reči u glavnoj memoriji se ažurira kasnije, onda kada dođe do uklanjanja bloka koji sadrži tu markiranu reč iz keša, kako bi se napravio prostor za novi blok. Ova druga tehnika se naziva *protokol povratnog pisanja* (*write-back*) ili *protokol povratnog kopiranja* (*copy-back*). Protokol piši-kroz je jednostavniji, ali on dovodi do nepotrebnih operacija upisa u slučaju kada se podatak menja više puta tokom boravka u kešu. Uočimo da i protoko povratnog pisanja može dovesti do nepotrebnih upisivanja – kada se blok iz keša upisuje natrag u memoriju, upisuju se sve reči bloka, čak i u slučaju kada je samo jedna reč promenjena tokom vremena boravka tog bloka u keš memoriji.

Kada se pri operaciji čitanja referiše na reč koja nije u kešu, dolazi do *promašaja pri čitanju*. Tada se blok reči koji sadrži zahtevanu reč kopira iz glavne memorije u keš. Pošto se ceo blok učita u keš, zahtevana reč se prosleđuje procesoru. Alternativa tome je da se reč pošalje procesoru čim bude donesena iz memorije.

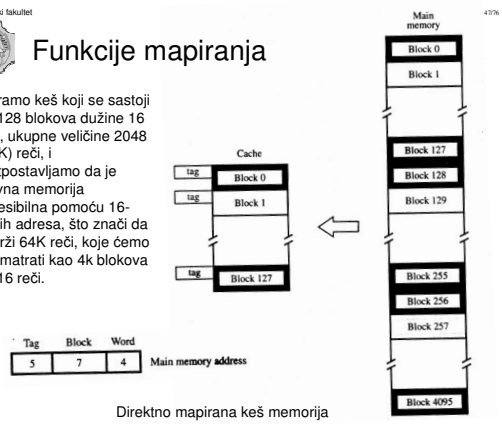
Keš memorije

Taj drugi pristup, nazvan *učitaj-kroz* (*load-through*) ili *rano restartovanje* u nekoj meri smanjuje period čekanja procesora, ali po cenu izgradnje složenijih elektronskih kola.

Tokom operacije pisanja, ako reč koja se adresira nije u kešu, dolazi do *promašaja pri upisu*. Tada, ako se koristi protokol piši-kroz, informacija biva upisana direktno u memoriju. U slučaju protokola povratnog pisanja, blok koji sadrži reč na koju se referiše se smešta u keš i potom se u odgovarajuću reč iz keša upišu nove informacije.

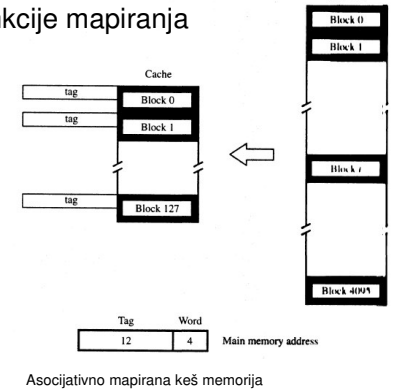
Funkcije mapiranja

Razmatramo keš koji se sastoji od 128 blokova dužine 16 reči, ukupne veličine 2048 (=2K) reči, i pretpostavljamo da je glavna memorija adresibilna pomoću 16-bitnih adresa, što znači da sadrži 64K reči, koje ćemo posmatrati kao 4k blokova od 16 reči.



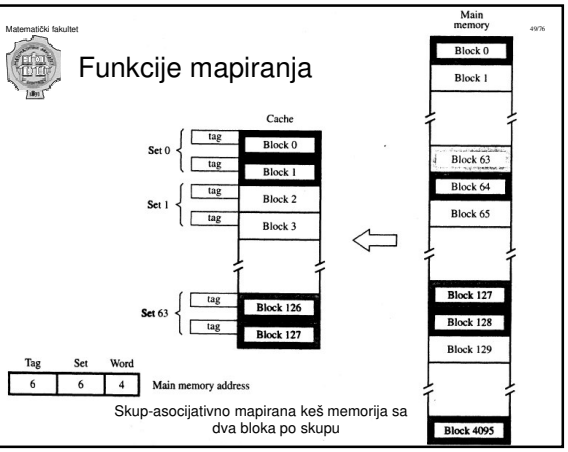
Direktno mapirana keš memorija

Funkcije mapiranja



Asocijativno mapirana keš memorija

Funkcije mapiranja



Skup-asocijativno mapirana keš memorija sa dva bloka po skupu

Algoritmi zamene

U direktno mapiranom kešu, pozicija svakog bloka je unapred određena. Stoga ne postoji strategija zamene. Kod asocijativnog i skup-asocijativnog keširanja postoji neka fleksibilnost. Kada se novi blok treba smestiti u keš, tada keš kontroler mora da odluči koji od starih blokova će da bude "pregažen".

Ovo je važna odluka zato što u velikoj meri utiče na performanse sistema. Uopšteno govoreći, cilj je da se u kešu zadrži blok na koji će se veoma verovatno referisati u bliskoj budućnosti. Osobina lokalnosti referenci ukazuje na razumnu strategiju. Naime, postoji velika verovatnoće da će blokovi koji su bili nedavno referisani biti uskoro referisani ponovo. Stoga će kontroler keša uklanjati blokove koji su bili najdavnije referisani (*least reacent used*) LRU.

Da bi se koristio LRU algoritam, keš kontroler mora pratiti reference na sve blokove tokom izvršavanja.

Pretpostavimo da je potrebno određivati LRU blok u skupu od četiri bloka unutar skup-asocijativnog keša. Za svaki blok skupa koristi se dvobitni brojač. Kada dođe do pogotka, brojač bloka koji je referisan se postavlja na 0. Brojači sa vrednostima koje su originalno bile niže od referenciranog se uvećavaju za 1, a sve ostale ostaju nepromenjene. Kada dođe do promašaja, a skup nije pun, brojač koji je pridružen novoučitanom bloku se postavlja na 0, a vrednosti svih ostalih brojača se uvećavaju za 1.

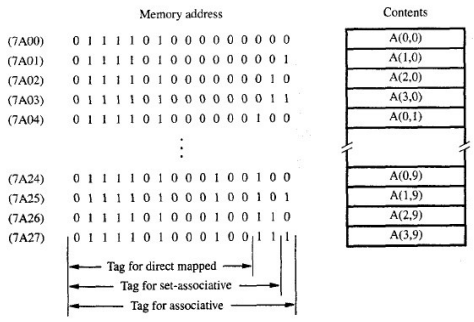
Algoritmi zamene

Kada dođe do promašaja, a skup jeste prazan, uklanja se blok kome je pridružen brojač sa vrednošću 3, postavlja se na njegovo mesto novi blok i njegov brojač se postavlja na 0. Preostali brojači se uvećavaju za 1.

LRU algoritam se najčešće koristi u ovu svrhu. Ipak, u nekim slučajevima on može dovesti do loših performansi (npr. kada se pristupa sukcesivnim lokacijama niza koji je malo veći od veličine keša). Performanse LRU algoritma mogu da se poprave dodavanjem male količine slučajnosti.

U praksi se koristi i još nekoliko algoritama zamene: intuitivno jasno uklanjanje "najstarijeg" bloka iz punog skupa blokova, uklanjanje slučajno izabranog bloka itd.

Primeri tehnika mapiranja



Niz koji je smešten u glavnoj memoriji

Primeri tehnika mapiranja

$$A(0,i) \leftarrow \frac{A(0,i)}{\left(\sum_{j=0}^{9} A(0,j)\right) / 10} \qquad \text{for } i = 0, 1, \ldots, 9$$

```
SUM := 0
for j:= 0 to 9 do
        SUM := SUM + A(0,j)
end
AVE := SUM / 10
for i:= 9 downto 0 do
        A(0,i) := A(0,i) / AVE
end
```

Primer zadatka i način njegove realizacije

Primeri tehnika mapiranja

Block position	Contents of data cache after pass: $j = 1$	$j = 3$	$j = 5$	$j = 7$	$j = 9$	$i = 6$	$i = 4$	$i = 2$	$i = 0$
0	A(0,0)	A(0,2)	A(0,4)	A(0,6)	A(0,8)	A(0,6)	A(0,4)	A(0,2)	A(0,0)
1									
2									
3									
4	A(0,1)	A(0,3)	A(0,5)	A(0,7)	A(0,9)	A(0,7)	A(0,5)	A(0,3)	A(0,1)
5									
6									
7									

Sadržaj direktno mapiranog keša za podatke

Primeri tehnika mapiranja

Block position	Contents of data cache after pass: $j = 7$	$j = 8$	$j = 9$	$i = 1$	$i = 0$
0	A(0,0)	A(0,8)	A(0,8)	A(0,8)	A(0,0)
1	A(0,1)	A(0,1)	A(0,9)	A(0,1)	A(0,1)
2	A(0,2)	A(0,2)	A(0,2)	A(0,2)	A(0,2)
3	A(0,3)	A(0,3)	A(0,3)	A(0,3)	A(0,3)
4	A(0,4)	A(0,4)	A(0,4)	A(0,4)	A(0,4)
5	A(0,5)	A(0,5)	A(0,5)	A(0,5)	A(0,5)
6	A(0,6)	A(0,6)	A(0,6)	A(0,6)	A(0,6)
7	A(0,7)	A(0,7)	A(0,7)	A(0,7)	A(0,7)

Sadržaj asocijativno mapiranog keša za podatke

Primeri tehnika mapiranja

	Contents of data cache after pass: $j = 3$	$j = 7$	$j = 9$	$i = 4$	$i = 2$	$i = 0$
Set 0	A(0,0)	A(0,4)	A(0,8)	A(0,4)	A(0,4)	A(0,0)
	A(0,1)	A(0,5)	A(0,9)	A(0,5)	A(0,5)	A(0,1)
	A(0,2)	A(0,6)	A(0,6)	A(0,6)	A(0,2)	A(0,2)
	A(0,3)	A(0,7)	A(0,7)	A(0,7)	A(0,3)	A(0,3)
Set 1						

Sadržaj skup-asocijativno mapiranog keša za podatke

Primeri keširanja kod komercijalnih procesora

Razmotrićemo implementaciju keša kod procesora 68040, ARM710T, te Pentium III i Pentium 4.

Keš kod procesora 68040

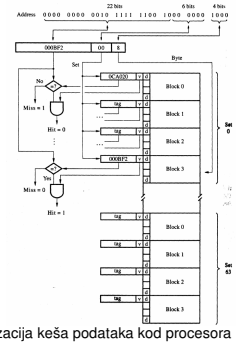
Procesor Motorola 68040 sadrži dva keša koji se nalaze na čipu – jedan koji se koristi za instrukcije i drugi za podatke. Svaki keš ima kapacitet od 4K bajtova i koristi četvorostranu skup-asocijativnu organizaciju koja je prikazana na sledećem slajdu. Keš sadrži 64 skupa, a svaki skup sadrži četiri bloka. Svaki blok ima dužinu od četiri reči, a svaka reč se sastoji od četiri bajta.

Poslednja četiri bita najmanje težine određuju poziciju bajta unutar bloka. Sledećih šest bitova identifikuju jedan od 64 bloka. Najviših 22 bita obrazuju oznaku (*tag*). Da bi notacija na dijagramu ostala kompaktna, sadržaj svakog od ovih polja se prikazuje u heksadekadnom obliku.

Mehanizam za kontrolu keša sadrži i jedan bit validnosti po bloku i jedan bit prljavosti po svakoj reči unutar bloka. Bit validnosti se postavlja na 1 kada se odgovarajući blok učita u keš. Bit prljavosti se postavlja na 1 kada god odgovarajuća reč biva promenjena tokom operacije upisa. Bit prljavosti ostaje postavljen sve dok se sadržaj bloka ne upiše nazad u memoriju.

Keš podataka može koristiti i protokol povratnog upisa i protokol piši-kroz, zavisno od operativnog sistema. Algoritam zamene na slučajan način bira blok koji se izbacuje.

Primeri keširanja kod komercijalnih procesora



Organizacija keša podataka kod procesora Motorola 68040

Primeri keširanja kod komercijalnih procesora

Keš kod procesora ARM710T

Kao što je već istaknuto, ARM familija procesora ima RISC arhitekturu, malu cenu i malu potrošnju energije. Procesor ARM710T koristi jedan keš i za instrukcije i za podatke. Keš kod procesora ARM710T je organizovan kao četvorostrani skup-asocijativni keš. Svaki blok sadrži 16 bajtova, tj. četiri 32-bitne reči.

Pri upisu u keš koristi se protokol piši-kroz. Za odlučivanje koji će blok biti "pregažen" kada se bude zahtevao novi blok, a ne bude slobodnog prostora, koristi se algoritam slučajnog izbora.

Keš kod procesora Pentium III

Pentium III je procesor visokih performansi. On sadrži dva nivoa keša.

Nivo 1 se sastoji od keša za instrukcije veličine 16Kb i keša za podatke veličine 16Kb. Keš za podatke ima četvorostranu skup-asocijativnu organizaciju i može koristiti bilo politiku povratnog pisanja, bilo politiku piši-kroz. Keš za instrukcije ima dvostranu skup-asocijativnu organizaciju. Kako se instrukcije obično ne modifikuju tokom izvršenja programa, nema potrebe za politikom upisa kod keša za instrukcije.

Keš nivoa 2 je mnogo veći. On sadrži i instrukcije i podatke i povezan je na keš-magistralu (koja je bitno brža od sistemske magistrale računara) preko jedinice za interfejs sa magistralom. Kod nekih verzija procesora (Katmai) L2 keš je van čipa procesora, sadrži 256 Kb implementiranih kao SRAM memorija.

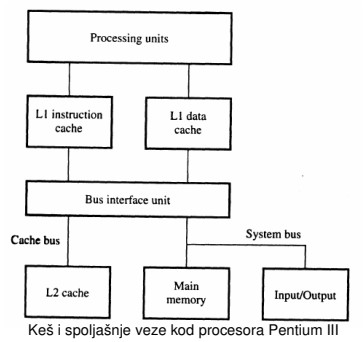
Primeri keširanja kod komercijalnih procesora

Organizacija ovakvog čipa je četvorostrana skup-asocijativna i koristi bilo protokol piši-kroz, bilo protokol povratnog pisanja, što se može programirati na nivou bloka. Magistrala za keš je širine 64 bita.

Poboljšanja kod VLSI tehnologije su omogućila da se L2 integriše na čip procesora, što je i urađeno kod čipa Pentium III verzija Coppermine. Veličina keša je 256K, koristi se osomostrana skup-asocijativna organizacija, a kako je L2 keš na procesorskom čipu, to je magistrala za keš širine 256 bita.

Katmai L2 keš radi polovinom brzine procesora, dok Coppermine L2 keš radi przinom procesora. Smeštanje L2 keša u procesor smanjuje kašnjenje pri pristupu i povećava propusnost, jer se podaci prenose kroz šire puteve, što rezultuje boljim performansama. Glavni problem kod pristupa u kome se L2 keš integriše u procesor je to što čip procesora postaje mnogo veći, pa se mnogo teže može proizvoditi.

Primeri keširanja kod komercijalnih procesora



Keš i spoljašnje veze kod procesora Pentium III

Primeri keširanja kod komercijalnih procesora

Keš kod Pentium 4 procesora

Pentium 4 ima do tri nivoa keša. L1 keš se sastoji od odvojenog keša za instrukcije i keša za podatke. Keš za podatke ima kapacitet od 8Kb, organizovanih na četvorostrani skup-asocijativni način. Svaki blok keša sadrži 64 bajta. Za upis podataka na keš koristi se politika piši-kroz. Celobrojnim podacima u kešu podataka se može pristupiti u dva ciklusa časovnika, što sa frekvencijom procesora od preko 1.3 GHz dovodi do vremena pristupa manjeg od 2 ns. Keš za instrukcije ne sadrži normalne mašinske instrukcije, već verzije tih instrukcija koje su već dekodirane.

L2 keš je unificirani keš kapaciteta 256Kb, organizovan na osmostrani skup-asocijativni način. Svaki njegov blok sadrži 128 bajtova. Za upis u keš se koristi politika povratnog pisanja. Kašnjenje pri pristupu ovom kešu iznosi sedam ciklusa časovnika.

I keš nivoa 1 i keš nivoa 2 su implementirani na čipu procesora. Ova arhitektura još dopušta i uključenje L3 keša na čipu. To nije implementirano na procesorima za desktop računare, već je omogućeno za procesorske čipove koji će se koristiti na serverskim sistemima.

Razmatranje performansi

Dva ključna faktora za uspeh su cena i performanse.

Uobičajena mera za uspešnost dizajna kojim se poboljšavaju performanse sistema bez značajnog uvećavanja cene je *odnos cena/performanse*.

Hijerarhija memorije opisana ovde se stvarala tokom potrage za optimalnim odnosom cena/performanse.

U postizanju optimalnog odnosa, od velikog značaja je da se prenos prema bržim jedinicama ili od njih vrši brzinom tih bržih jedinica. Ovo nije moguće ako se i bržim i sporijim memorijskim jedinicama pristupa na isti način, ali to može biti postignuto kada se koristi paralelizam u pristupu sporijoj jedinici.

Preplitanje predstavlja jedan efikasan način uvođenja paralelizma.

Preplitanje

Dva dijagrama na sledećem slajdu prikazuju dva načina strukturisanja memorijskih adresa. U prvom slučaju, *k* bitova veće težine označavaju jedan od *n* modula, a *m* bitova manje težine ukazuju na konkretnu reč u tom modulu. U slučaju kada se pristupa uzastopnim lokacijama, sve se dešava u jednom modulu.

U drugom slučaju, gde se radi o tzv. *memorijskom preplitanju*, nižih *k* bitova u adresi služi za izbor modula, a viših *m* bitova određuje lokaciju unutar modula. Da bi se impementirala preplićuća struktura, mora biti 2^k modula – inače će biti biti rupa sa nepostojećim lokacijama u adresnom prostoru.

Primer. Prenosi se blok od 8 reči iz glavne memorije u keš, zato što se dogodio promašaj. Pretpostavimo da hardveru treba jedan ciklus da pošalje adresu u memoriju, da je DRAM memorija spora, pa se prvoj reči pristupa za 8 ciklusa, a svakoj sledećoj za 4 ciklusa i da je jedan ciklus potreban za slanje redne reči u keš. Broj ciklusa potrebnih za prenos u slučaju jednog memorijskog modula je

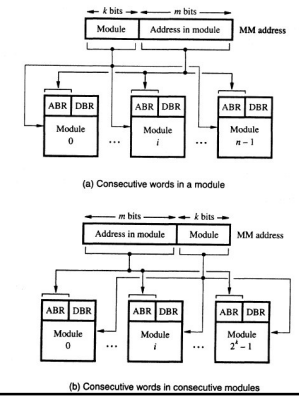
1+8+(7*4)+1 = 38

Broj ciklusa u slučaju preplićuće strukture memorije sa 4 modula je

1+8+4+4 = 17

Preplitanje

Adresiranje kod memorijskih sistema sa više modula



Nivo pogodaka i kazna za promašaj

Nivo pogodaka je količnik između broja pogodaka i ukupnog broja pokušanih pristupa, a *nivo promašaja* je količnik između broja promašaja i ukupnog broja pokušanih pristupa. Visoki nivoi pogotka, dosta iznad 0.9 su esencijalni za računare visokih performansi.

Kazna za promašaj je veličina koja predstavlja dodatno vreme potrebno za dovođenje nove informacije u keš. Termin kazna se koristi zato što je u slučaju promašaj procesor zaustavljen u svom radu. Kao što je već istaknuto, preplitanje smanjuje veličinu kazne.

$$t_{ave} = hC+(1-h)M$$

$$t_{ave} = h_1C_1+(1-h_1)h_2C_2+(1-h_1)(1-h_2)M$$

Primer. Odrediti grubu aproksimaciju za poboljšanje performansi kod istog slučaja kao prilikom preplitanja, pri čemu je pretpostavka da na 100 izvršavanja memorijskih instrukcija ima 30 operacija čitanja ili upisa podataka, da je nivo pogodaka keša za instrukcije 0.9, a keša za podatke 0.95.

$$\frac{vreme\,bez\,kesa}{vreme} = \frac{130(1+8+1)}{100(0.95\times1+0.05\times17)+30(0.9\times1+0.1\times17)} = 5.04$$

Nivo pogodaka i kazna za promašaj

Nivo pogodaka je količnik između broja pogodaka i ukupnog broja pokušanih pristupa, a *nivo promašaja* je količnik između broja promašaja i ukupnog broja pokušanih pristupa. Visoki nivoi pogotka, dosta iznad 0.9 su esencijalni za računare visokih performansi.

Kazna za promašaj je veličina koja predstavlja dodatno vreme potrebno za dovođenje nove informacije u keš. Termin kazna se koristi zato što je u slučaju promašaj procesor zaustavljen u svom radu. Kao što je već istaknuto, preplitanje smanjuje veličinu kazne.

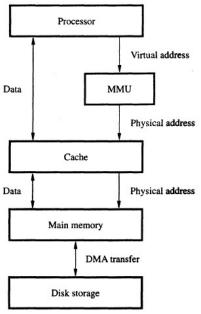
$$t_{ave} = hC+(1-h)M$$

$$t_{ave} = h_1C_1+(1-h_1)h_2C_2+(1-h_1)(1-h_2)M$$

Primer. Odrediti grubu aproksimaciju za poboljšanje performansi kod istog slučaja kao prilikom preplitanja, pri čemu je pretpostavka da na 100 izvršavanja memorijskih instrukcija ima 30 operacija čitanja ili upisa podataka, da je nivo pogodaka keša za instrukcije 0.9, a keša za podatke 0.95.

$$\frac{vreme\,bez\,kesa}{vreme} = \frac{130(1+8+1)}{100(0.95\times1+0.05\times17)+30(0.9\times1+0.1\times17)} = 5.04$$

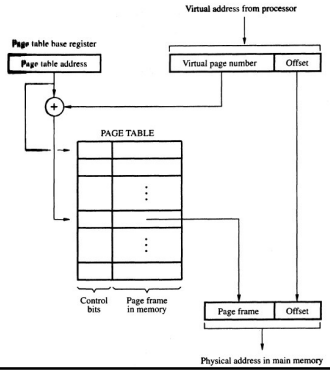
Virtuelna memorija



Organizacija virtuelne memorije

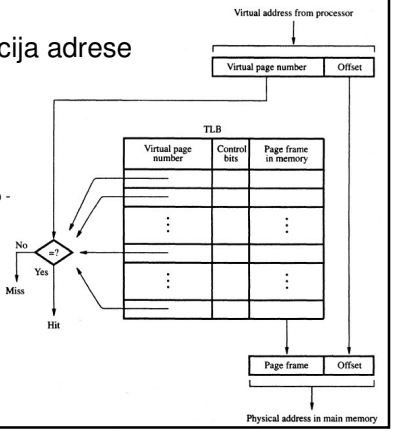
Translacija adrese

Translacija adrese kod virtuelne memorije



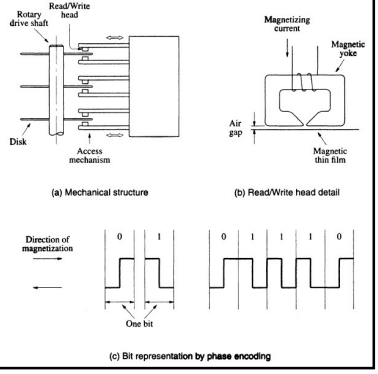
Translacija adrese

Korišćenje asocijativno - mapiranog TLB

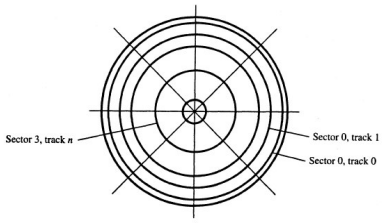


Magnetni diskovi

Principi rada magnetnih diskova

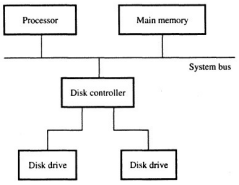


Magnetni diskovi



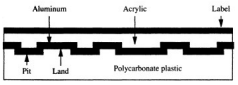
Organizacija jedne površi diska

Magnetni diskovi



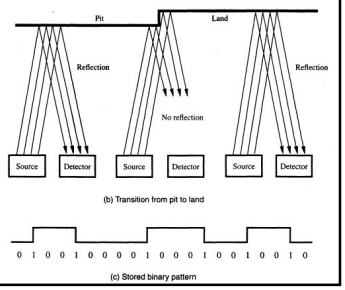
Diskovi koji su povezani na sistemsku magistralu

Optički diskovi

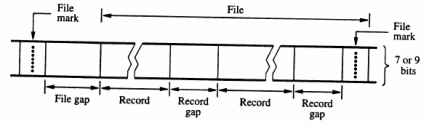


(a) Cross-section

Optički disk



Sistemi sa magnetnim trakama



Organizacia podataka na magnetnoj traci

Zadaci

-
-
-