TEORIJSKI DEO – MIKS

Magistrale mikroračunarskih sistema

* Sistemi sa 1 magistralom
* Sistemi sa odvojenom ulazno/izlaznom magistralom
* Sistemi sa više magistrala

Sistem sa 1 magistralom

* Arhitekturno jednostavniji (postoji 1 skup linija preko kojih svi komuniciraju).
* Nedostaci: svi uređaji konkurišu za 1 prolaz, tako da u jednom trenutku samo 1 uređaj može da prenosi svoje podatke
* Drugo, sprežne mreže (elektronika koja omogućava da se moduli priključe na magstralu) moraju da budu složenije, obzirom da treba da podrže različite tipove uređaja i različite tipove komunikacije.

Delovi magistrale (po tipovima podataka)

* Adresna magistrala – služi za izbor memorijske lokacije ili U/I uređaja. Obzirom da imamo više uređaja priključenih na tu magistralu, a svaki od tih uređaja može da ima više lokacija, potrebno je specificirati čemu se pristupa. Adresna ima najčešće najviše linija da bi pokrila sve lokacije
* Magistrala podataka – nakon selekcije određene lokacije, potrebo je preneti podatke sa određene lokacije ili na tu lokacije. Širina magistrale zavisi od sistema (32, 64,...)
* Upravljačka magistrala – služi za prenos upravljačkih signala (signali read, write ili signal kojim se signalizira prekid INTERRUPT ili signal kojim se odobrava prekid, ili DMA request zahteva oslobađanje magistrale)

Tipovi prenosa podataka po magistrali

1. Prenosi pod kontrolom programa
2. Prenosi pod kontrolom prekida
3. Direktan memorijski pristum (DMA)

Sistem sa odvojenom ulazno/izlaznom magistralom

Uvedeni zbog toga što se razlikuje pristup memoriji od pristupa U/I, zbog toga što je potrebno da se rastereti memorijska magistrala od prenosa podataka prema U/I, zbog odvajanja upravljanja tim prenosom, javlja se novi arbitar (DMA modul – direktno bez učešća procesora kopira podatke sa U/I uređaja u memoriju ili iz memorije u memoriju)

Dobra strana: dva nezavisna stream- a podataka, dakle ukoliko se podaci prenose između dva U/I uređaja. To se može odvijati nezavisno od memorijskog podsistema, dakle CPU može pristupati memoriji dok se podaci direktno kopiraju sa jednog na drugi uređaj, ili sa U/I u DMA modul pa kasnije u memoriju. Sprežne mreže su jednostavnije, zato što se sada razlikuje pristup memorije od pristupa U/I uređaja (malo su specifičnije), ceo sistem je generalno složeniji. Arhitektura CPU-a mora da bude kompleksnija da bi podržala te nezavisne magistrale i nezavisan prenos.

Sistemi sa više magistrala

* Ulazno izlazna magistrala se deli na više magistrala pri čemu su neki uređaji specijalizovani za DMA prenos, može se prenositi samo preko DMA. Sa nekih uređaja se podaci kopiraju isključivo na inicijativu interrupta, tako da su sprežne mreže još jednostavnije, a sam sistem je kompleksniji, jer sada svaki od podsistema može da funkcioniše za sebe.
* Broj linija je ograničen i ne mogu se dodavati nove, postoje ograničenja u dužini, što su duže veće je kašnjenje, ne smeju se preklapati.

Nemultipleksirana magistrala

Najosnovniji oblik magistrale, tj. Da svaka linija ima svoju funkciju. Neke su samo za prenos adresa (linije), neke za prenos podataka (bidirekciona). Da bi se smanjio broj linija na ploči, da bi se smanjio broj pinova na integrisanim kolima, vrlo često se vrši multipleksiranje magistrale (isti pinovi se koriste za različite funkcije)

Multipleksirana magistrala

Kod nekih računara se recimo vrši multipleksiranje višeg i nižeg bajta adrese. U prvom taktu se recimo prenosi viši deo adrese (viši bajt), a u drugom taktu niži bajt. To podrazumeva da postoji poseban signal koji će da kaže kada se prenosi šta od adrese i mora da postoji nekakav leč koji će zapamtiti prethodni deo adrese , obzirom da će se na istim linijama sad naći nešto drugo, da bi mogla da se kompletira čitava adresa.

Kod drugih računara multipleksiraju se adrese i podaci (INTEL 8085 ili 8086), prvih 16 bitova adrese se multipleksiraju sa podacima i ponovo je potreban poseban signal, u ovom slučaju Address Letch Enable (ALE) koji služi za aktiviranje kola koji će zapamtiti adresu da bi adresa ostala aktivna i nakon toga prvog takta.

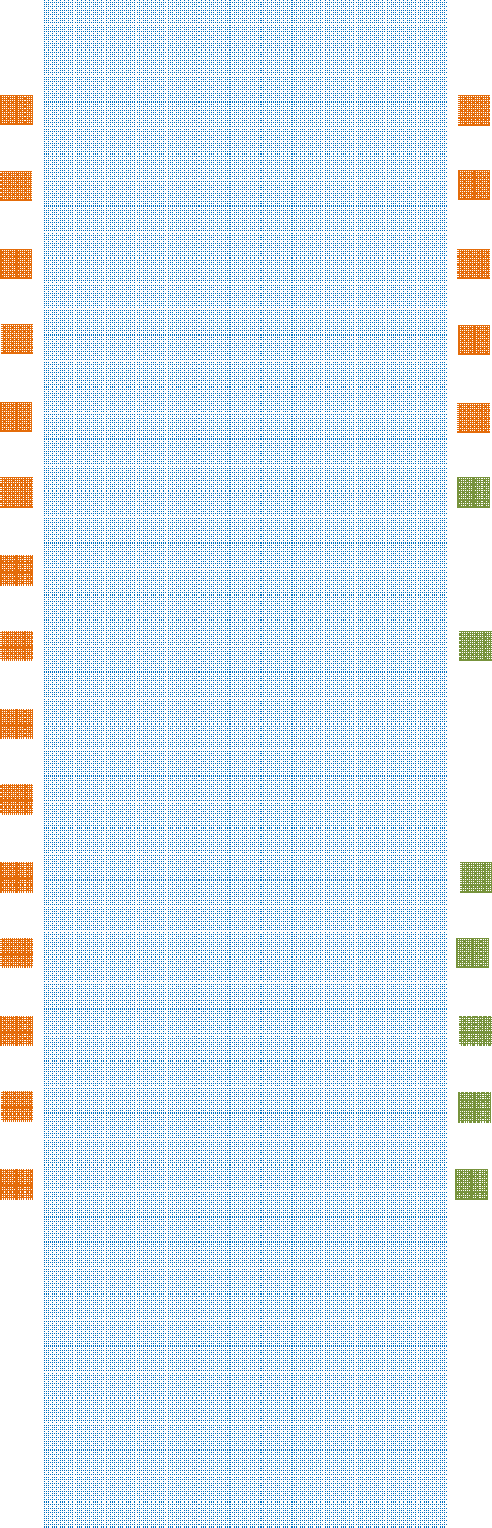
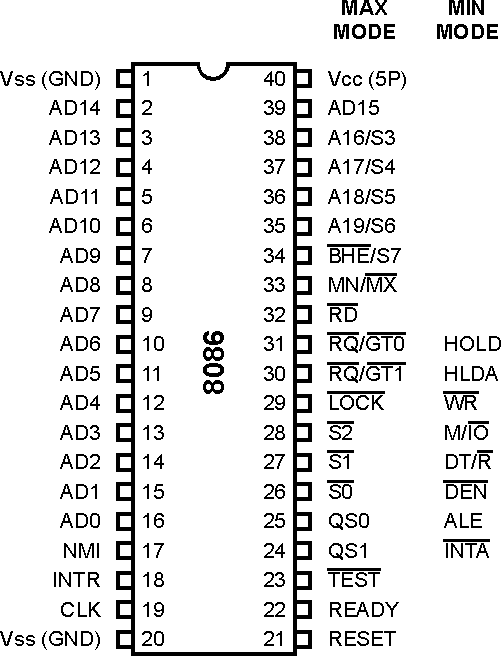
Mogu se multipleksirati podaci i stanja, gde se u jednom trenutku preko magistrale podataka prenosi podaci a u drugom se ostavlja kod u kome je stanje procesor.

Bavimo se samo INTEL 8086.

Potpuno multipleksirane magistrale

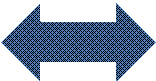
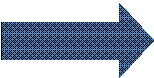
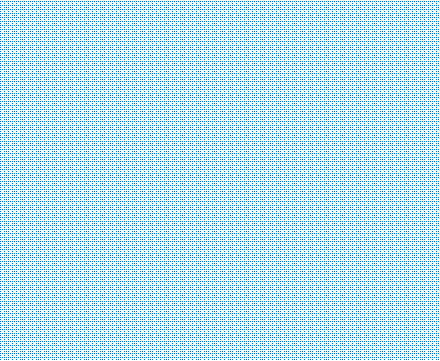
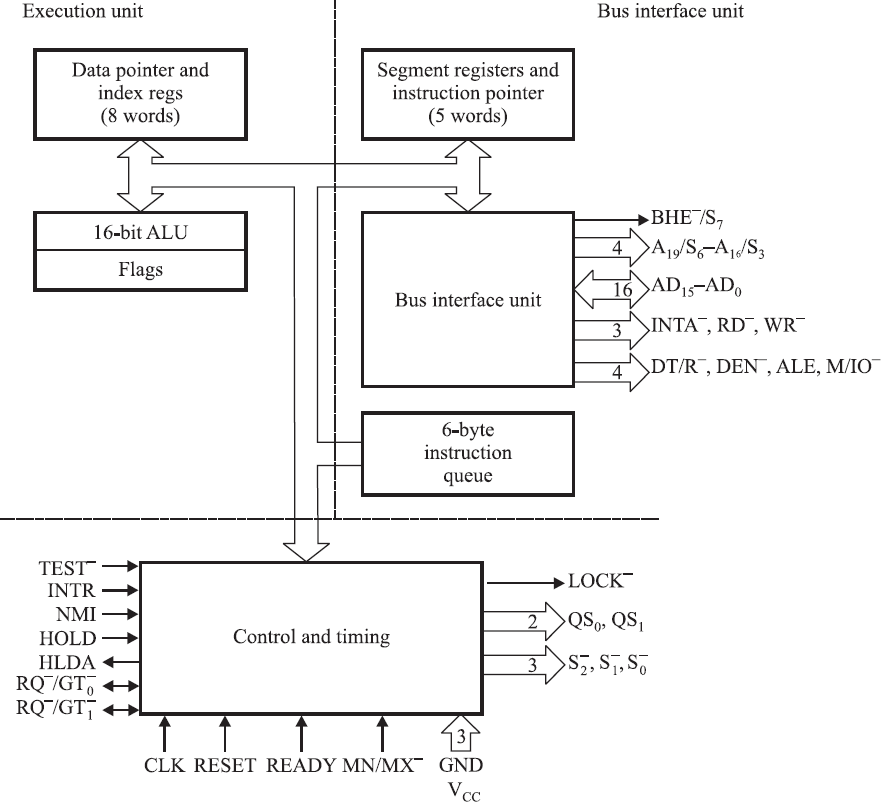
U najopštijem slučaju možemo sve multipleksirati i podatke i adresu preko jedne iste magistrale. Recimo ako je adresa podeljena na viši i niži deo možemo u prvom taktu preneti viši deo koji dolazi do memorije i do jednog i drugog leča ali će posebno kolo vremenski vođeno selektovati samo jedan od dva leča da bi se u njega upisao odgovarajući podatak. Dakle u prvom taktu se generiše signal koji aktivira prvi leč i upisuje se viši deo, u sledećem taktu se upisuje niži deo, a u trećem taktu se postavlja podatak i praktično se generiše signal sa selekciju memorije. Prednost ovde je to što imamo samo jednu magistralu i u okviru te magistrale je multipleksiran prenos adrese i podataka, a loše je to što su nam potrebna 3 takta i da adresiramo ćeliju i da postavimo podatak u nju.

I8086



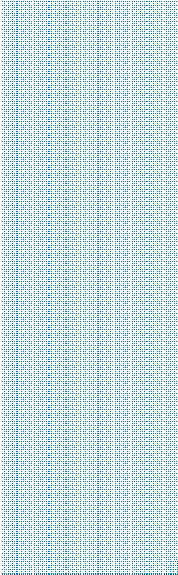
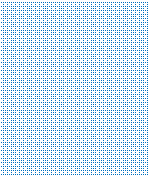
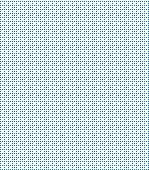
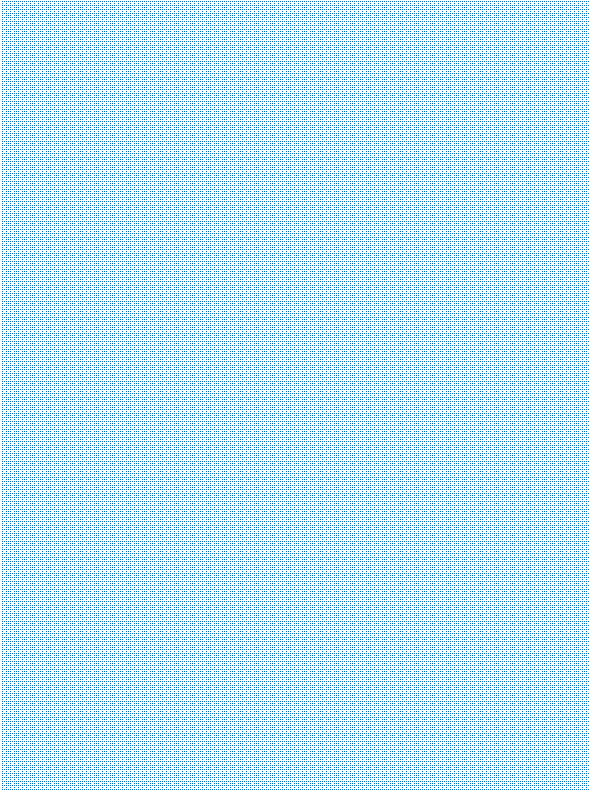
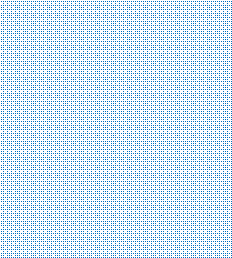
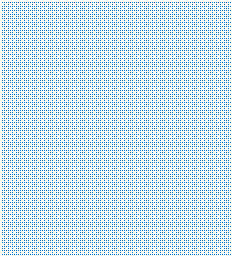
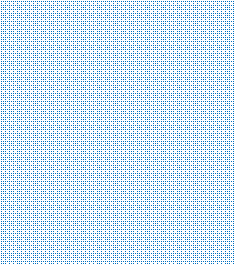
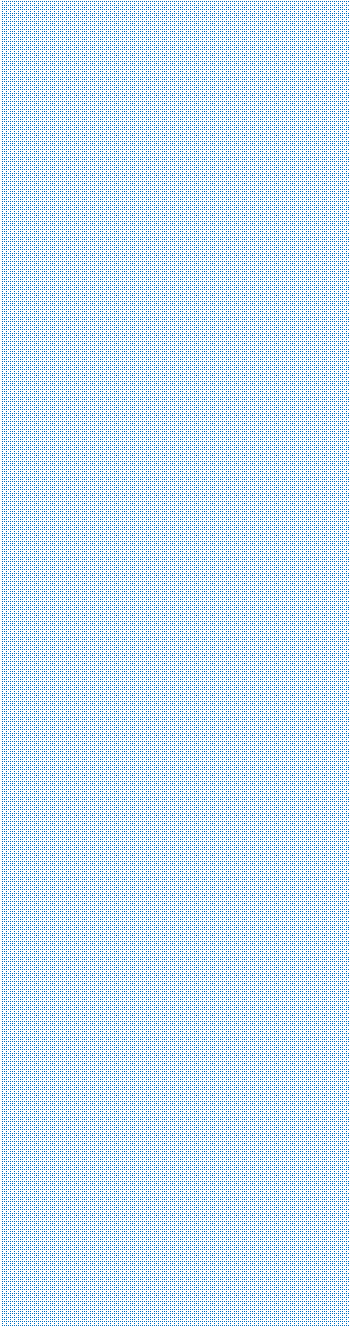
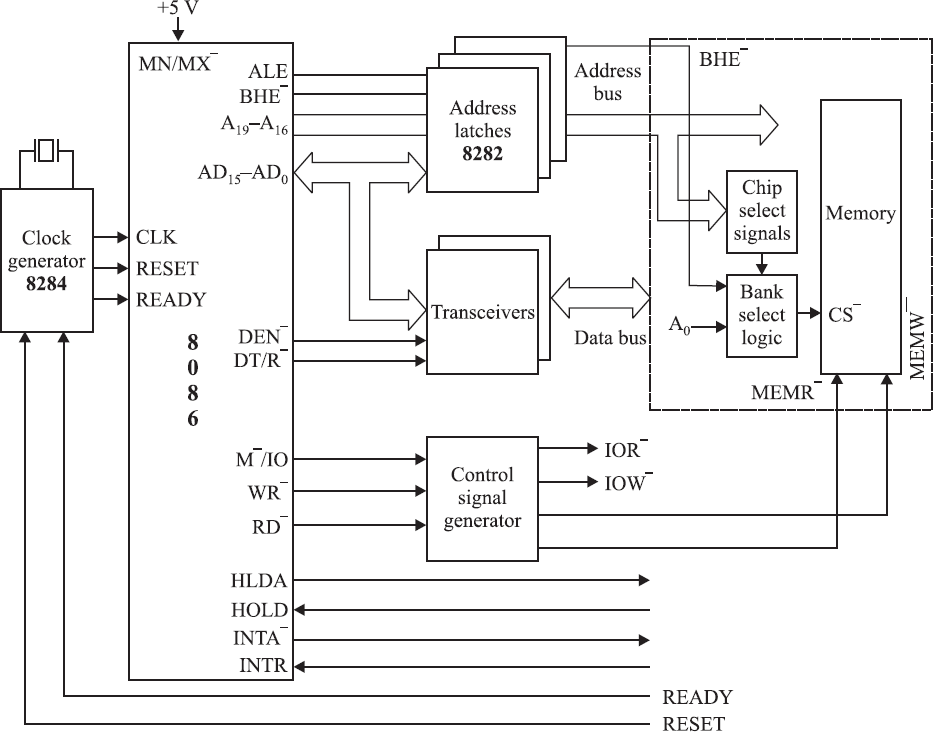
* Najveći broj pinova ( od ukupnih 40) je potrošeno na adrese (50% pinova ukupno 20 je rezervisano za adrese )
* Da ne bismo uzeli svih 20 pinova adresom i podacima, nižih 16 bitova adrese je multipleksirano sa podacima
* Bitni signali su nam RD- čitanje podataka, WR- upis podataka i MEM/IO – definiše da li se pristupa memoriji ili ulazno izlaznom uređaju
* Radićemo u MIN modu, koji podrazumeva da postoji samo jedan CPU u sistemu
* Bitni pinovi kod ovog procesora za komunikaciju sa memorijom je BHE (Bus High Enable) koji omogućuje viši deo data bus- a. Ovaj procesor omogućava da se pristupa memoriji na nivou bajta ali je na nivou 16 bitne reči. Da li je viši deo reči enableovan ili ne zavisi od ovog BHE. On zajedno sa adresnom linijom 0 određuje koja se banka selektuje. Ovaj sistem ima 2 banke: parnu i neparnu. Ako je adresna linija 0 na 0, onda je selektovana parna ili nulta banka, a ako je BHE na nuli onda je selektovana neparna tj prva banka, ako su oba signala na nuli onda je selektovana i parna i neparna, odnosno čita se ili upisuje čitava reč.
* DT/R (Data Transmit/Receive) i DEN (Data Enable) su signali koji upravljaju transmiterom preko koga je ovaj CPU vezan na data bus, a ALE služi da se zapamti adresa kako bi mogli da se postave podaci na pinove AD0 do AD15.
* Sam CPU ima 2 bitna segmenta: Execution Unit koji upravlja procesom izvršenja i Bus Interface Unit koji upravlja interfejsom prema magistrali

I8086 blok dijagram



Niži deo adresne magistrale je istovremeno I DATA magistrala I to je bidirekciona 16 bitna magistrala a viših 4 bita adrese su adresni bitovi u prvom taktu, a kasnije se preko njih definišu stanja procesora.

I8086 minimum mod interfejs prema memoriji



* Instrukcijski ciklus se sastoji od 4 takta:
  + U prvom taktu se postavlja adresa I da bi se ona zapamtila u trenutku kada je definisan podatak, koristi se leč za adrese (8282) I ona se aktivira na ALE, to je signal koji je aktivan u prvom ciklusu I koji omogućava da se adresa zapamti. Pamte se adresa I BHE
* Nakon toga menja se stanje na magistrali

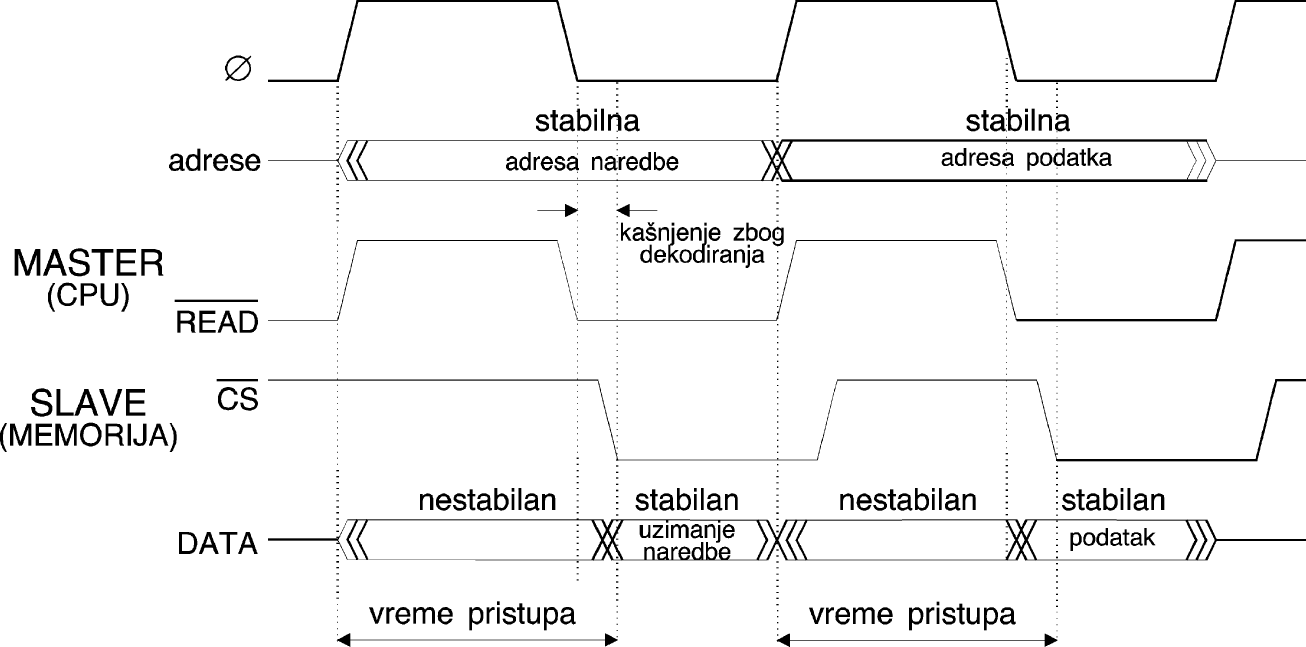
Protokoli za prenos podataka po magistrali

Prema protokolima za prenos podataka po magistrali, magistrale delimo na:

* Sinhrone – ako su svi učesnici upravljani clock- om procesora
* Asinhrone

Sinhrone magistrale

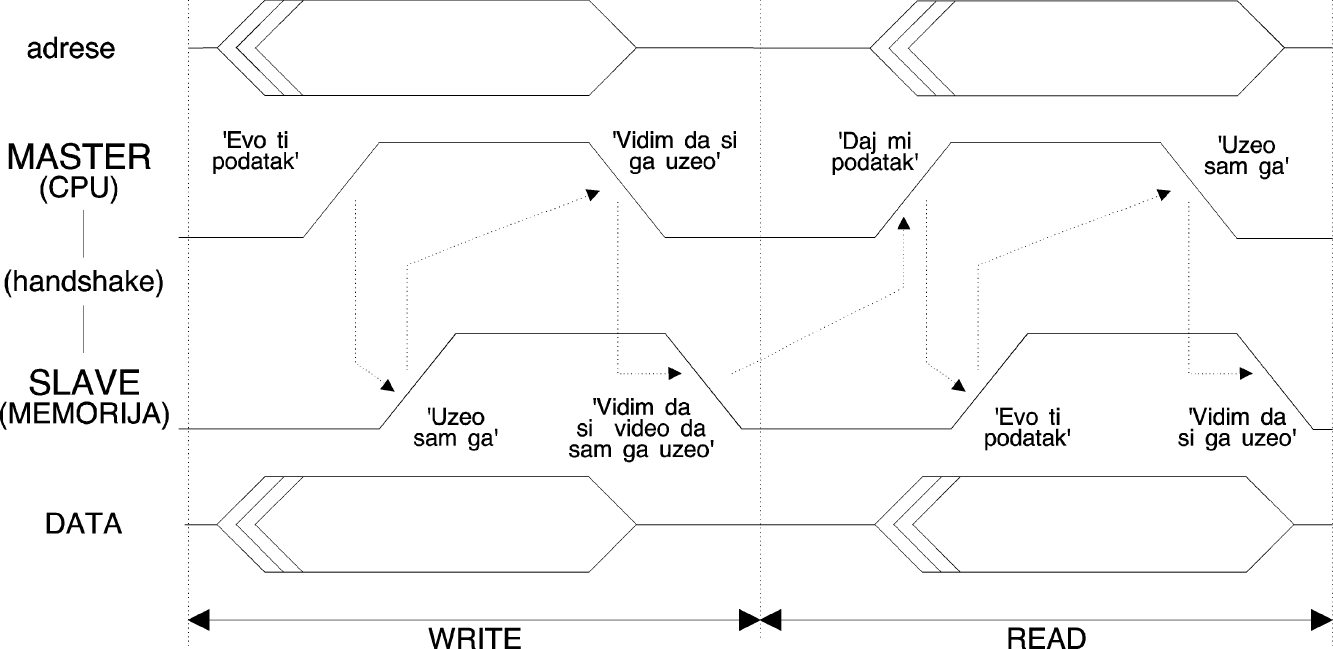
* Magistrale upravljane sistemskim clock- om I podrazumeva se da svi učesnici u komunikaciji poštuju taj clock, da su uvek spremni za komunikaciju I da u određenim trenucima mogu da izvrše operaciju.
* Pretopstavka je da master(CPU) želi da pošalje ili da pročita neki podatak iz memorije, on će najpre postaviti adresu.Recimo da u prvom taktnom intervalu (sa slike) postavlja adresu naredbe i adresa se postavlja na samom početku (na uzlaznu ivicu clocka), zajedno sa read signalom,što označava da procesor želi da pročita nešto iz memorije.
* Potrebno je određeno vreme da se stabilizuje adresa, zato se podatak i ne može čitati u prvom delu taktnog intervala
* Osim signala da li je čitanje ili upis, potrebno je da postoji i chip select signal (selekcija komponente). Obično komponente imaju 3 stanja na izlazima, tako da se one i fizički odvajaju od magistrale (chip select ne radi) ako nije potrebno. Dok je stanje nestabilno, ono što je u tom trenutku na tim linijama je nevalidno. Nakon određenog vremena po aktiviranju cs signala, podatak postaje dostupan (memorija je spremna). Memorija postavlja podatak, u našem primeru to je naredba i CPU može da je pročita sa DATA magistrale. Na primeru je naglašeno da postoji određeno kašnjenje signala na magistrali (recimo CS signal kojim se selektuje neka periferija), mora da zakasni neko vreme obzirom da se uređaji nalaze na različiim rastojanjima u odnosu na magistralu odnosno CPU, i propagacija signala nije ista svuda. Što je memorija udaljenija, to je potrebno više vremena da se dođe do nje.
* CPU preuzima u drugom delu clocka podatak I nakon toga podatak može da se skine sa magistrale (CS postaje neaktivan). Za to vreme, recimo da se postavlja adresa podatka, prvo skidamo naredbu, zatim skidamo podatak koji je ta naredba obradila, opet je potrebno da se aktivira read signal. Neko vreme nakon što je postavljena ta adresa, potrebno je da se ona dekodira I ponovo se aktivira CS I podatak se preuzima sa magistrale
* Ideja sinhrone magistrale, ukratko, da se sve dešava u određenim vremenskim intervalima I da je pretpostavka da je svaka strana u komunikaciji spremna da obavi datu transakciju
* Slika objašnjena prethodnim tekstom



Kašnjenje zbog različitih dužina linija

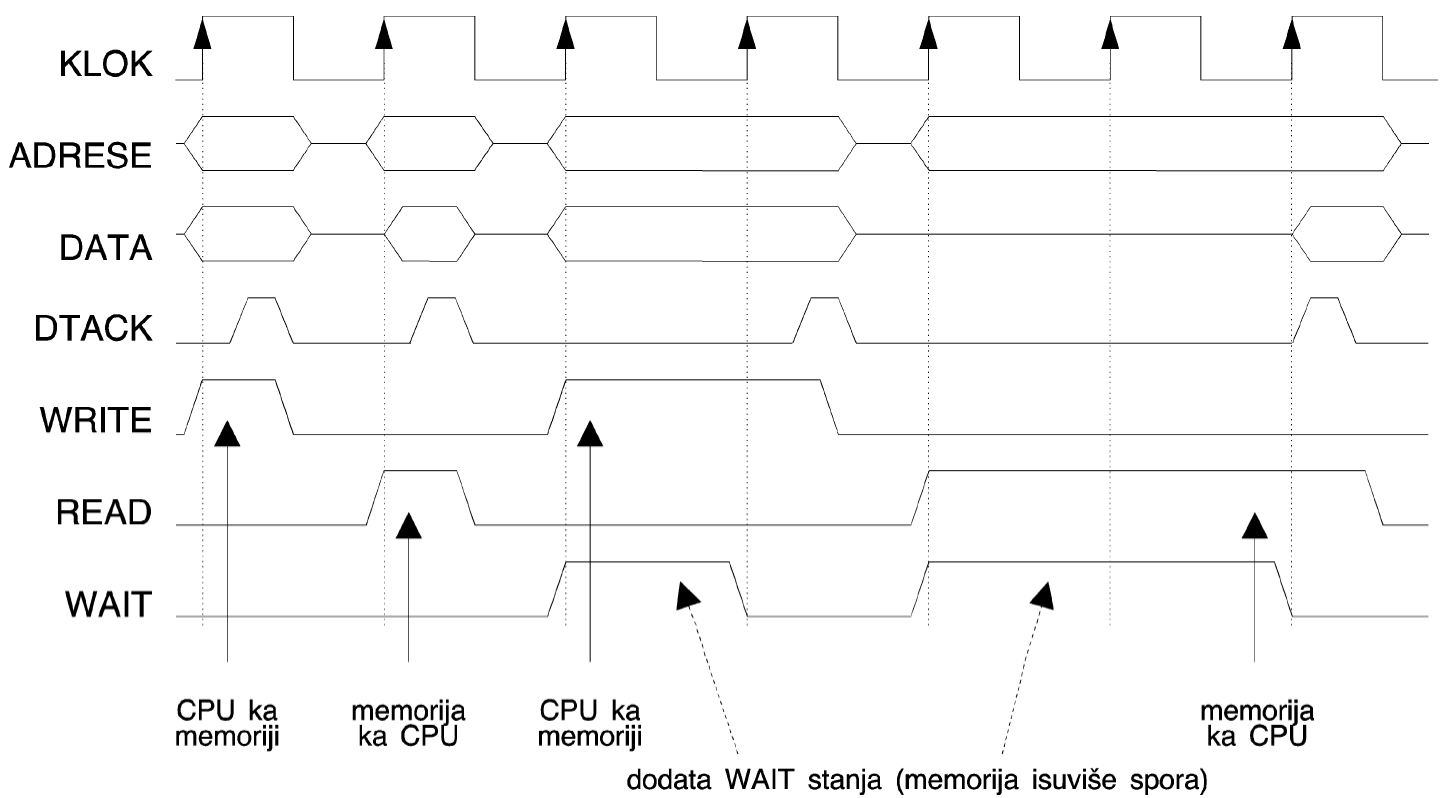
* Periferni uređaj nije selektovan trenutno, već postoji kašnjenje zbog dekodiranja adrese.
* Postoji kašnjenje i zbog udaljenosti od CPU
* Zbog različite dužine linija na štampanoj ploči adresni signali neće doputovati u isto vreme (na visokim učestanostima čak i razlika od nekoliko centimetara može biti značajna)
* Zbog toga postoji i među linijama na samoj magistrali (magistrala od više linija), postoji neka razlika kašnjenja signala po svakoj od tih linija, što otežava projektovanje samih sistema. Veće rastojanje zahteva niži takt da bi sve moglo da stigne u određenom vremenskom intervalu

Asinhrone magistrale

* Prenos nije vezan za takt. Zahtevaju se dodatne “handshake” linije. Pogodne za sisteme sa različitom brzinom CPU i periferija. Ali ne rade baš bez takta, postoji neki sistemski takt jer određene akcije moraju da se izvode u određenom trenutku, ali ne možemo da pretpostavimo kada je neki uređaj spreman za komunikaciju.
* Primer za sliku ispod, gde CPU kao master i kao brz uređaj komunicira sa memorijom. Da bi mogao da upiše ili pročita podatak iz memorije, mora da bude siguran da memorija može da prihvati odnosno da je postavila taj podatak koji se čita. Zbog toga postoje, u ovom slučaju ovaj full interloc assistant gde postoje 2 odvojene linije za komunikaciju (2 handshake linije). Jedna je grant kojom se CPU obraća preiferiji, a druga je ACKNOWLEDGE kojom u ovom slučaju memorija potvrđuje da je prihvatila odgovarajuću akciju.
* Recimo CPU želi da upiše podatak, postavlja podatak na magistralu, postavlja adresu te periferije ili memorije i zadaje signal kojim kaže da je podatak postavljen. I adresa i podatak moraju ostati na magistrali sve dok slave ne kaže da je video taj podatak i da može da ga preuzme. Aktiviranjem ACK signala, on stavlja do znanja da je započeo uzimanje podataka. Na uzlaznu ivicu tog signala master skida grant signal, tj skida i adresu i podatak (memorija preuzela podatak), na šta slave skida svoj ACK signal i time kaže da je video da je podatak skinut i preuzet sa magistrale. Ako želi CPU da pročita podatak, postupak je potpuno isti. Dakle sada će se aktivirati drugi signali, signal READ između ostalog, postavlja se adresa i šalje se zahtev za čitanjem (aktivira se grant). Memorija aktivira ACK kada je spremna, čime kaže da je podatak spreman, u međuvremenu podatak je postavljen na magistralu i CPU nakon tog signala može da preuzme podatak sa magistrale. Kada je dobio ACK on skida svoj grant i nakon toga i memorija skida svoj podatak i svoj ACK.
* Kod ovog sistema moguće je da periferija bude mnogo sporija od CPU
* Negde između ova dva tipa nalaze se semisinhrone magistrale, one su kompromis između ova dva tipa.

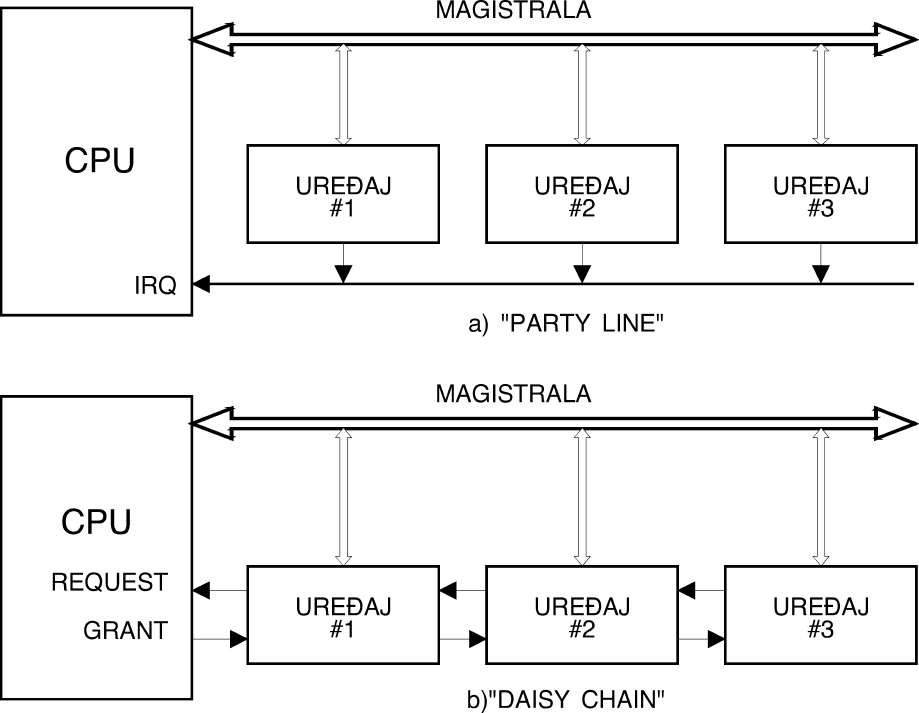
Semisinhrone magistrale

* Koristi brzinu sinhronih, a dozvoljava povezivanje periferija različitih brzina. Rade asinhrono dok periferija nije spremna. Nakon toga se prenos obavlja sinhrono.
* Primer sa slike virtuelni. Postavljaju se adrese i podaci pod upravljanjem takta. Recimo da CPU želi da upiše neki podatak u memoriju, postavlja adresu i podatak i write signal. Ukoliko je memorija spremna i pročitala je podatak, ona postavlja signal DataACK i time stavlja do znanja CPU u da može da pređe na sledeću instrukciju. Nakon ovoga pretpostavimo da CPU želi da čita podatak. Postavlja adresu lokacije i read signal, i kada je memorija spremna ona postavlja signal da je podatak dostupan i podatak će biti pročitan sa magistrale
* Problem nastaje kada memorija nije spremna. U tom slučaju uvode se dodatna wait stanja I dok god CPU očitava odgovarajući signal da taj uređaj nije spreman on će ostati u tekućem stanju. Recimo kod CPU 8086 postoji signal ready kojim se omogućuje ovakav vid komunikacije. Dakle dok god na prednjoj ivici clocka nije aktivan ready, CPU mora da ubacuje wait cikluse, dok ta periferija ne bude spremna da se izvrši ta transakcija. Na taj način se omogućuje da memorija bude I sporija.
* Sinhrono prenose podatke, ali omogućavaju I čekanje na spore periferije time što se ubacuju u stanje čekanja
* Slika sa prethodno objašnjenje



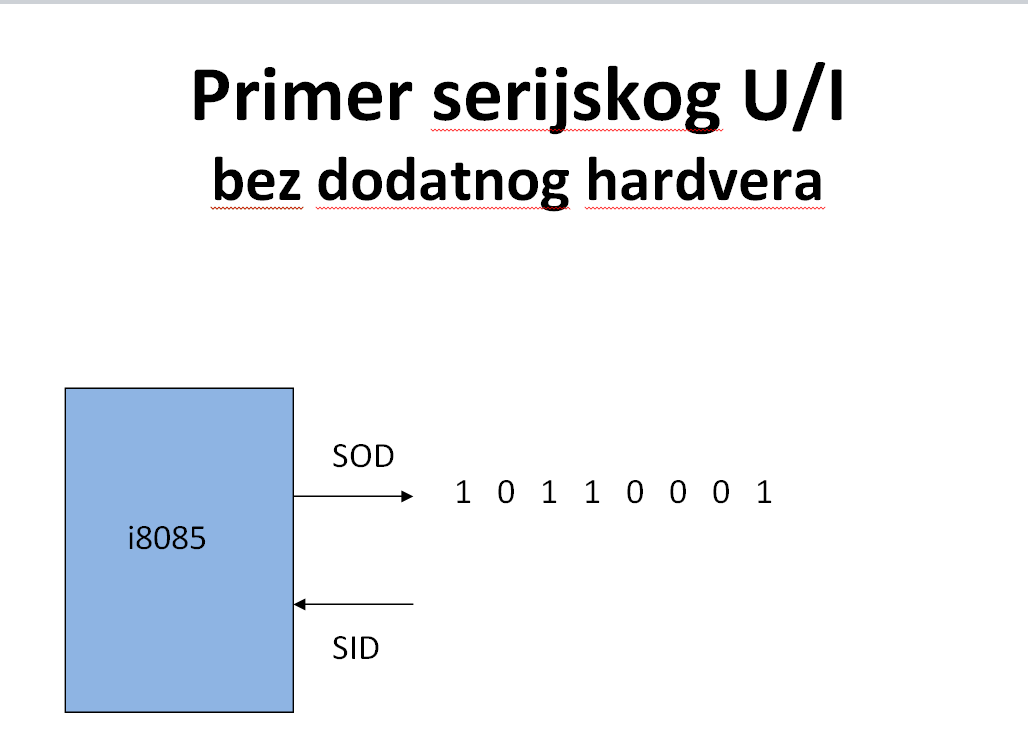
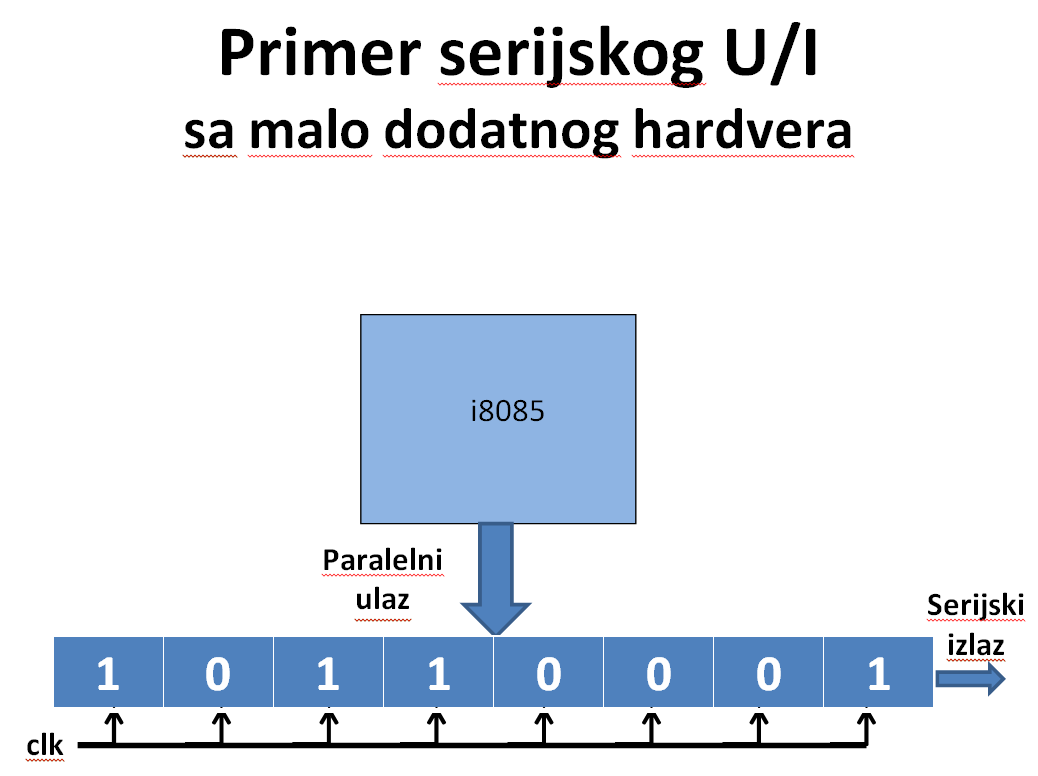
Arbitriranje na magistrali

* Na magistrali može da se pojavi veliki broj uređaja, zato je potrebno da postoji nekakva arbitraža i da se zna ko kada može da prenosi svoje podatke preko magistrale. Dva najčešća načina povezivanja uređaja su:
  + Party line
  + Daisy chain
* Party line – svi uređaji koji su povezani na magistralu povezani su sa CPU sa još jednom linijom, najčešće je to interrupt request linija kojom signaliziraju da žele da prenesu podatke. Problem ovde je sada što sada CPU ne zna ko je želeo da prenese podatke i sad mora da postoji još jedan dodatni mehaizam, neka prozivka recimo koja bi omogućila detekciju uređaja. Jednostavno za implementaciju ali je dosta zamoran za CPU jer nakon tog signala moraju se proveriti svi uređaji da se utvrdi ko zahteva prenos. Tu se može uvesti neki prioritet. Prioritet može da bude fiksan pa da prozivka uvek kreće od prvog uređaja ili rotirajući, da se zapamti dokle se stiglo sa prozivkom pa sledeći put odatle.
* Daisy chain – složenija varijanta, jer zahteva 2 linije za komunikaciju. Jedna linija je request kojom uređaj zahteva magistralu, a druga je grant kojom CPU dozvoljava korišćenje magistrale. Ovde su svi uređaji povezani jedni na druge i tako formiraju ogrlicu (lanac) i praktično redosled uređaja u tom lancu određuje prioritet uređaja. Onaj koji je bliži CPU- u imaće viši prioritet obzirom da će zadržati za sebe pravo na korišćenje magistrale.

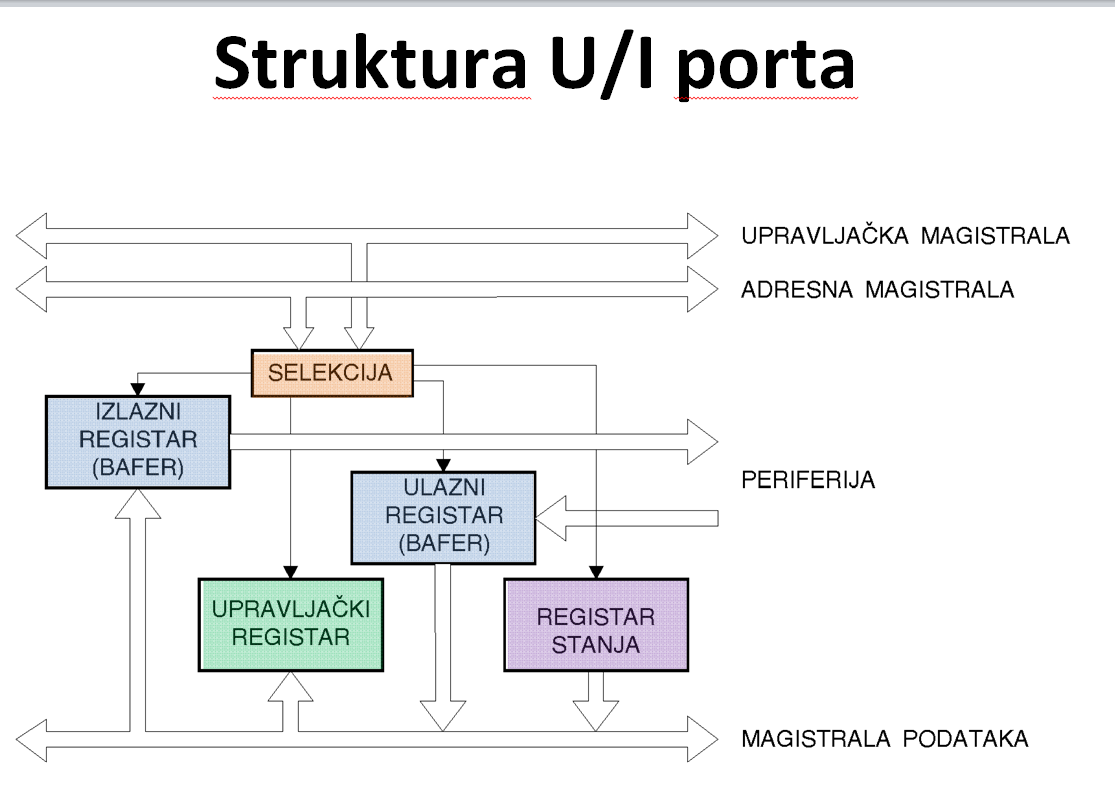


* Primer ako dva uređaja komuniciraju odnosno da prenose podatke preko magistrale. Uređaj koji šalje podatak, aktivira signal Request out prema CPU odnosno prema susednom uređaju ne mora biti CPU. Ako je Request In neaktivno, niko ne zahteva pristup magistrali, tekući uređaj želi pristup, on aktivira svoj Request Out i recimo da dobije mogućnost da komunicira. Tada se javlja Grant In signal i uređaj može da pristupi magistrali. Dakle ukoliko signal želi da komunicira, pošalje zahtev i grant ne prosleđuje. Ukoliko je komunikaciju želeo da ostvari uređaj desno od zadatog uređaja, onda se javlja zahtev na Request In liniji, on se prosleđuje na Request Out prema CPU- u i ukoliko CPU prihvati to, dobija se Grant In i prosleđuje se ka Grant out. Naš zadati uređaj ne želi da komunicira, on samo prosledi ka Grant Out i služi kao relej.
* Ukoliko i uređaj 2 i uređaj 3 sa slike žele da komuniciraju, aktivan je request In i taj zahtev se prosleđuje, ali u tom momentu i drugi uređaj želi da komunicira. U tom slučaju grant In signal se zadržava tj ne prosleđuje ka grant out pinu dok god uređaj 2 ne zavrži svoju komunikaciju. Tek po završetku, on aktivira Grant Out i dozvoljava sledećem ukoliko je Grant In ostao aktivan.

Povezivanje U/I uređaja

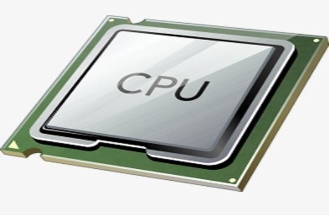
* Predstavlja kompromis između hardvera i softvera. Prednost hardvera je brzina, a mana nefleksibilnost.
* Upravljanje uređajem može biti na strani CPU-a, što je fleksibilnije, možemo u softveru da definišemo rad, ali je brzina manja.
* Postoji nekakav periferijski kontroler koji nije previše složen koji obavlja neke transakcije, a opet CPU je taj koji ga proziva, zadaje mu režim rada i očitava njegovo stanje ili podatke koje su pročitane sa periferije. Najbolji primer za to kakva je razlika između hw i sw, možemo da vidimo na serijskoj komunikaciji
* 
* - CPU 8085 ima 2 osnovna pina koji se koriste za serijsku komunikaciju. To su SOD za serijski output i SID za serijski ulaz. Ako želimo da prenosimo podatke serijski (šaljemo ili primamo bit po bit određenom brzinom) onda je potrebno da se odgovarajućim instrukcijama postavi vrednost na dati pin, da se sačeka određeno vreme, pa da se promeni vrednost da se postavi neka druga. CPU treba da igra ulogu tog serijskog komunikatora i da prosledi poruku. To zahteva mnogo vremena obzirom da je serijska mnogo sporija (bitska brzina prenosa je mnogo manja od takta CPU- a) tako da se više ciklusa troši da se pošalje samo jedan bit. Drugo, CPU vrši sam prenos umesto da obavlja neki drugi posao, tako da je proces vrlo spor. Ovo može da se značajno ubrza samo ako uvedemo shift registar.
* 
* Ovo je minimalan hardver koji omogućuje da se paralelno upiše čitava reč i to se obavlja u jednom taktu, a onda pod dejstvom nekog drugog takta kojim se upravlja preko CPU-a i definiše se njegova brzina, se shiftuje sadržaj i bit po bit prosleđuje sa uređajem sa kojim se komunicira. Na ovaj način CPU je oslobođen od potrebe komunikacije. Sve vreme dok se sadržaj shifta on može da obavlja neke druge instrukcije i obzirom da je sam proces shifta sporiji, može se izvršiti dosta veći deo koda

Strutura U/I porta



* Generalno, za komunikaciju sa perifernim uređajima koristimo portove. Pod pojmom port se podrazumeva neki minimalni hw koji je zapravo interfejs prema nekom uređaju. CPU ga najšeće vidi kao nekoliko registara koji se nalaze na odgovarajućim adresama u koje upisuje podatke ili čita podatke, tj upravlja radom te same komponente definisanjem neke upravljačke reči.
* Na slici imamo 2 bafera u izlazni bafer se upisuje podatak koji se šalje ka periferiji, a u ulazni bafer se učitava podatak sa periferije i prozivkom CPU može da pročita zadavanjem adrese.
* U/I port može da ima različita stanja i upravo zato se koristi upravljački registar kojim se definiše stanje porta. Zavisi od toga koliko je složen sam uređaj, može imati neke dodatne upravljačke signale i mora da postoji mogućnost da se očita njegovo trenutno stanje da bismo znali kako je setovan i u kakvom stanju su U/I baferi, zato je potrebno da postoji i neka dodatna logika koja treba da upravlja čitavom strukturom, odnosno vrši dekodiranje signala, selekciju odgovarajućeg bafera i omogućavanje pristupa magistrali.
* Generalno, periferni uređaji su sporiji od CPU-a i zbog toga je potrebno da postoji mogućnost da se podaci smeštaju u više zasebnih registara koji se postepeno pune, a CPU može da čita iz njih kada je spreman i kada su podaci raspoloživi. Najjednostavniji oblik baferovanja je dvostruko baferovanje

Dvostruko baferovanje



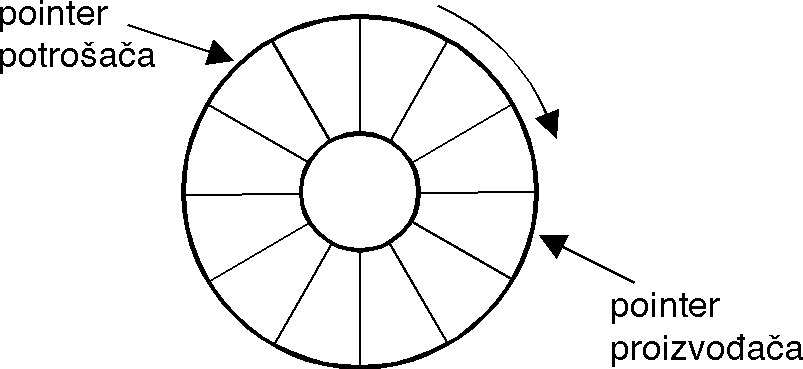
**Bafer B**

**BAFER B**

**Bafer A**

**BAFER A**

* Često se koristi recimo kod pristupa HDD-u, gde postoje dve memorijske lokacije koje mogu da sadrže podatke. Da CPU ne bi čekao dok se jedan bafer napuni podacima, on čita iz drugog, koristi se često u računarstvu, dok uređaj spojen čitanjem, puni drugi bafer. Kada se završi popuna tog bafera sa periferije, zamene se pokazivači, odnosno, taj bafer koji je popunjen podacima (recimo bafer A) postaje dostupan CPU- u, a popunjava se bafer B. Ukoliko je periferija još sporija i ukoliko se ređe čitaju podaci sa nje, još bolja varijanta je CIKLIČNO baferovanje



Ciklično baferovanje

* Postoji više memorijskih lokacija gde se upisuju onako kako pristižu, a kada je CPU spreman za čitanje, on čita s početka pa do lokacije koja je poslednja popunjena.

Načini povezivanja U/I uređaja

U odnosu na način adresiranja, može biti:

* Odvojeni (izolovani) U/I
* U/I preslikan na memoriju

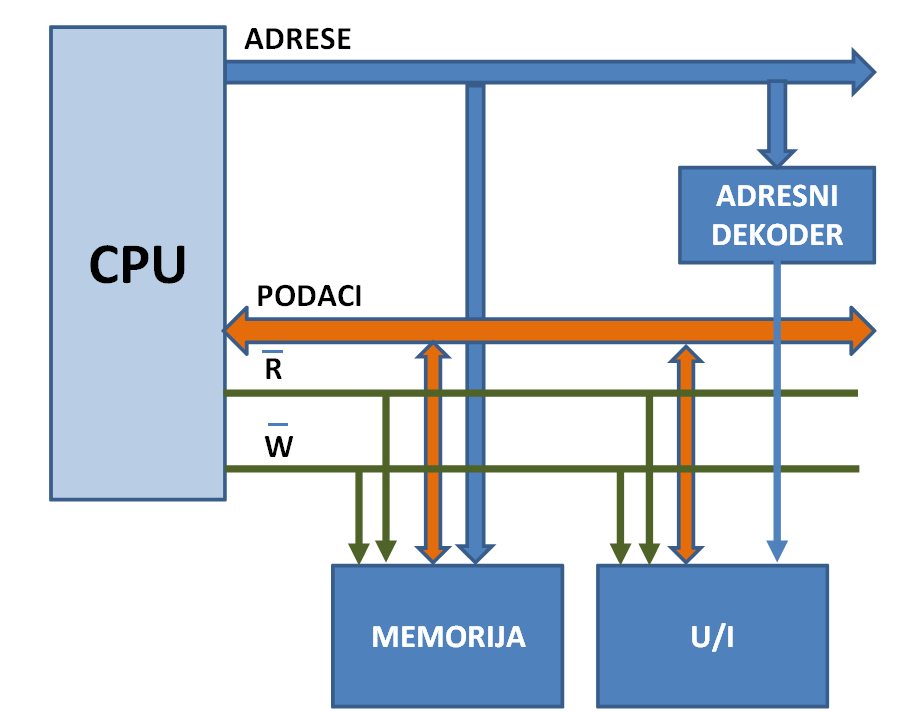
Odvojeni(izolovani) U/I



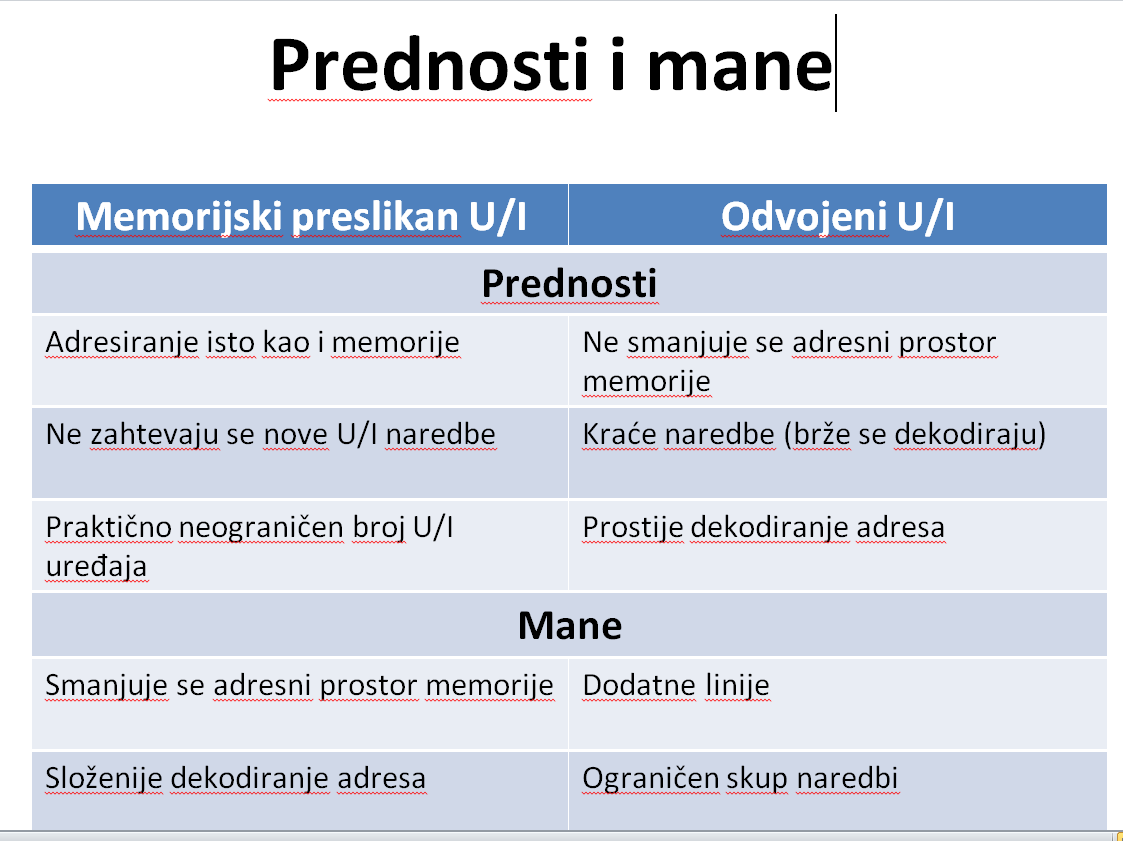
* Može da postoji posebna magistrala za adresiranje U/I uređaja, ili da se koriste adresne linije koje se koriste i za adresiranje memorije, kao na slici iznad, a posebnom linijom da se selektuje da li je pristup U/I uređaju ili memoriji (8086 cpu).

.

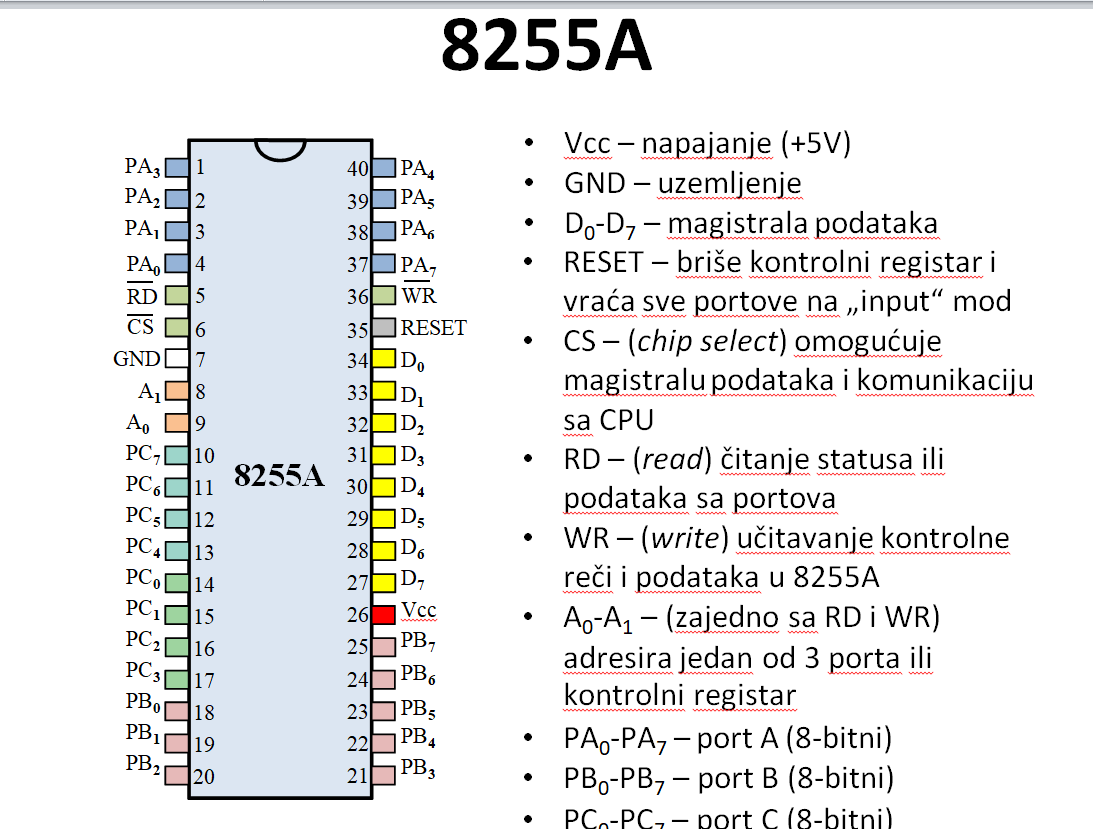
U/I preslikan na memoriju



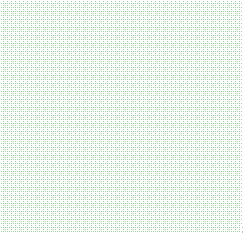
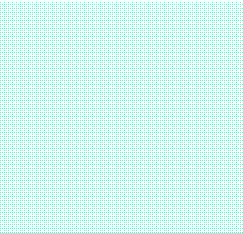
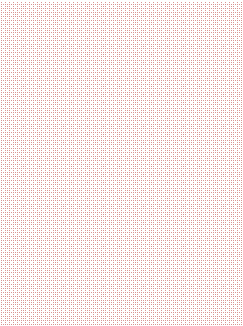
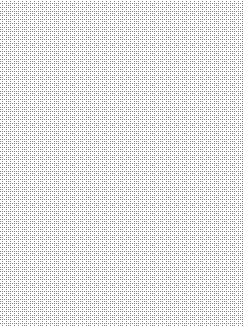
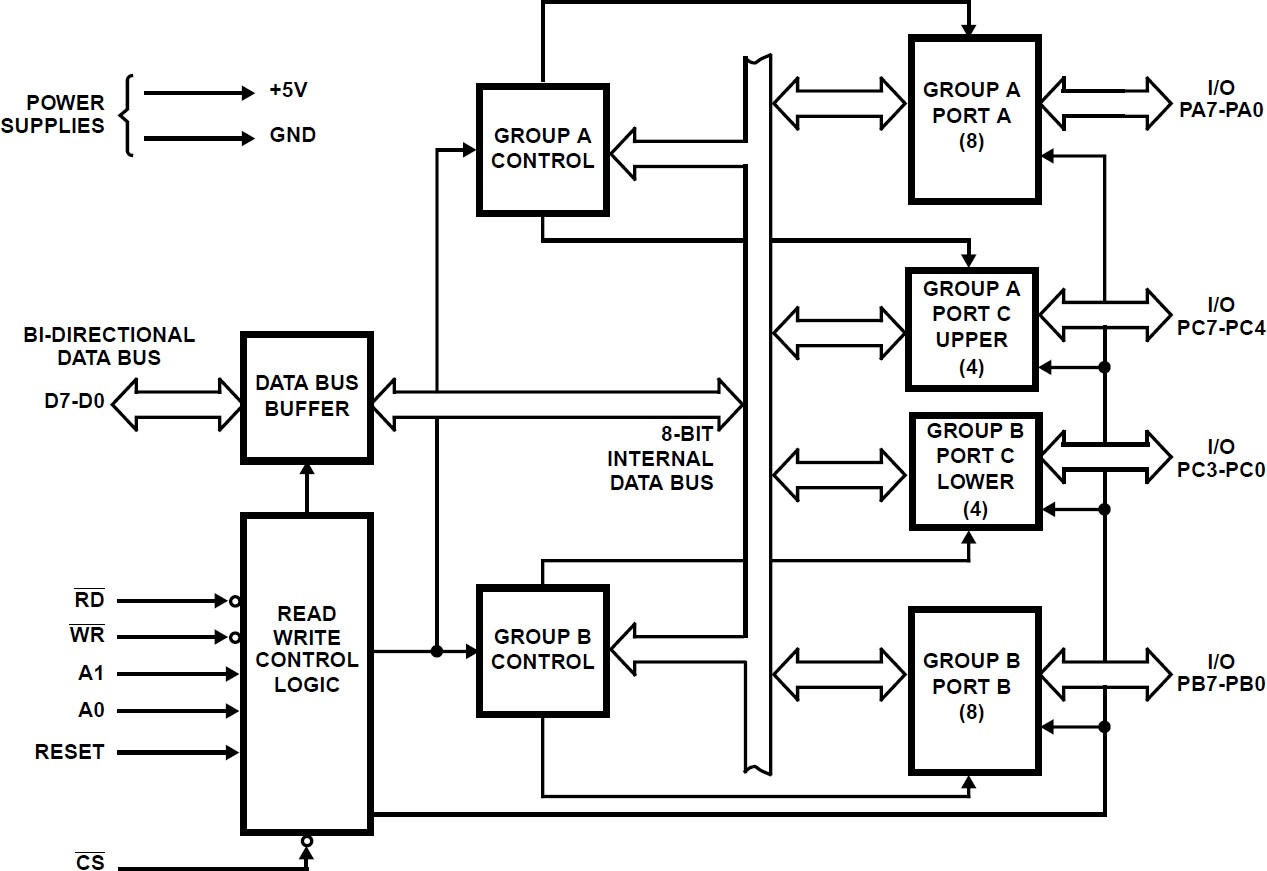
* Nema specijalnog hw oganičenja za pristup U/I uređaju, koriste se standardne adrese za pristup memoriji s tim što se određeni opsezi adresa odvajaju za U/I uređaje. Dakle, nemamo sada kontinuiranu šemu memorije, ne možemom, već ako upisujemo na neku adresu, to se upisuje na neki U/I uređaj ili se čita. Ovo podrazumeva da imamo dodatne adresne dekodere za dekodiranje adresa i selektovati odgovarajuću komponentu za komunikaciju.



8255A – programabilni periferijski interfejs

* Komponenta koja omogućuje da povežemo U/I uređaje na CPU 8086. Ima 24 programabilna U/I pina. Podeljena u 3 porta, ima 3 moda rada i omogućava direktno setovanje nekih pinova. Vrl složena komponenta
* 
* Ima 40 pinova, najveći broj pinova su ulazno-izlazni pinovi(portovi).
* Ima napajanje (5V) i ima uzemljenje.
* Reset- dovođenjem napajanja na ovaj pin, praktično se resetuje sadržaj i briše se upravljačka reč i vraća se na prvobitno podešavanje. Svi portovi se u ovom slučaju vraćaju u INPUT mode. Neophodno da se poveže na uzemljenje da ne bi došlo do slučajnog resetovanja
* Specifično za ovu komponentu su adresne linije A0 i A1. To su pinovi koji selektuuju jedan od registara za upis ili čitanje. S obzirom da ova komponenta ima 3 porta, praktično 3 registra u kojima se upisuje ili iz kojih se čita podatak, potrebne su nam 3 adrese i imamo još 1 upravljački registar kojim se definiše mod rada same komponente.
* Komponenta mora biti povezana na magistralu, zato se koriste pinovi od D0 do D7.
* Pinovi preko kojih vezujemo U/I uređaje i to su PA, PB i PC. Tri porta A B i C. Obično su osmobitni ali port C može da se posmatra i kao 2 odvojena 4 bitna porta.
* CS – dovodi se signal za selekciju čipa, na RD i WR signale za selekciju moda, čitanje ili upis.n

Logička šema



* Pinovi su organizovani u portove pri čemu su portA i viši deo porta C organizovani u grupu A. U nekim modovima rada potrebno je da se upravlja samim portom, nekim spoljašnjim signalima. U tom slučaju ćemo koristiti port C za to upravljanje. To isto važi i za port B. Viši deo porta C je pridružen portu A da bi mogla da se formira grupa A. Svaka od ovih grupa ima svoju kontrolnu logiku, povezana je preko interne magistrale na data bus buffer i na data magistralu sistema. Postoji logika za upis i čitanje koja na osnovu zadate adrese i signala read ili write selektuje odgovarajući registar i čita u data bus buffer ako je read ili iz njega upisuje podatak na zadati port.
* Komponenta radi u 3 moda
  + Mod 0 – osnovni ulaz/izlaz –
  + Mod 1 – strobovani ulaz/izlaz- kada se dodatnim signalima upravlja lečovanje podataka
  + Mod 2 – važi samo za port A, bidirekciona magistrala, kada se port može istovremeno koristiti i za upis i za čitanje.

Mod 0

* Dva 8 bitna i dva 4 bitna porta, može i viši i niži deo porta C da se posmatra kao nezavisan port.
* Svaki port može biti ili ulazni ili izlazni, ali ne i jedno i drugo
* Svi izlazi su lečovani – kada se upiše neka vrednost, ona ostaje dok se ne promeni
* Ulazi nisu lečovani samo u trenutku kada se čita, ono što je na magistrali se preuzima i smešta u registar
* 16 mogućih različitih konfiguracija

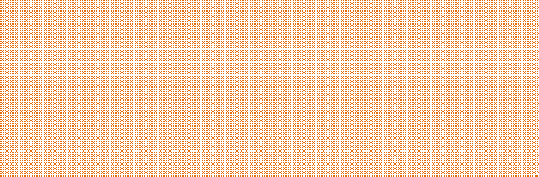
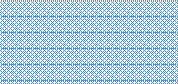
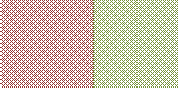
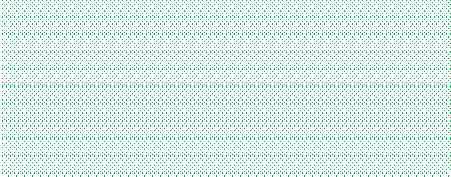
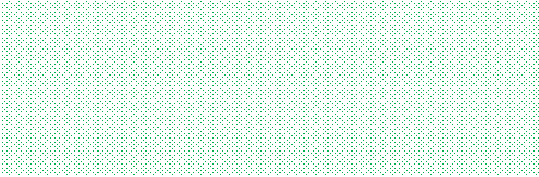
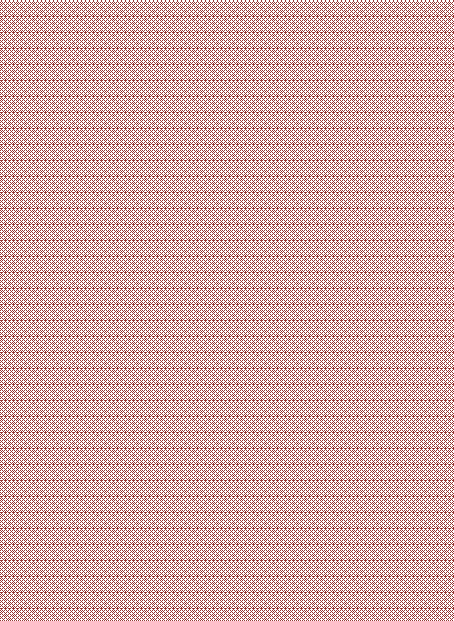
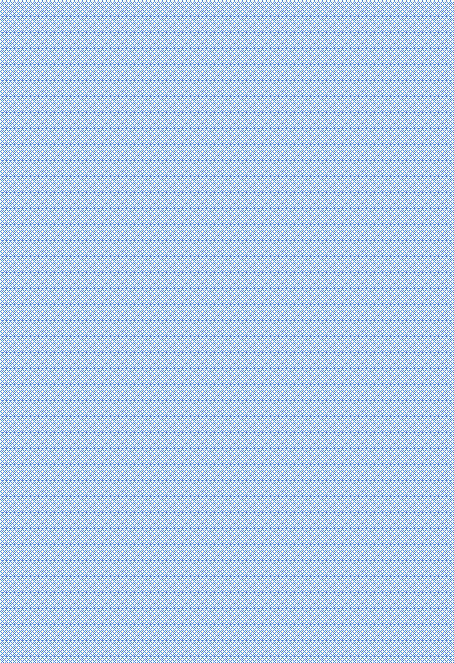
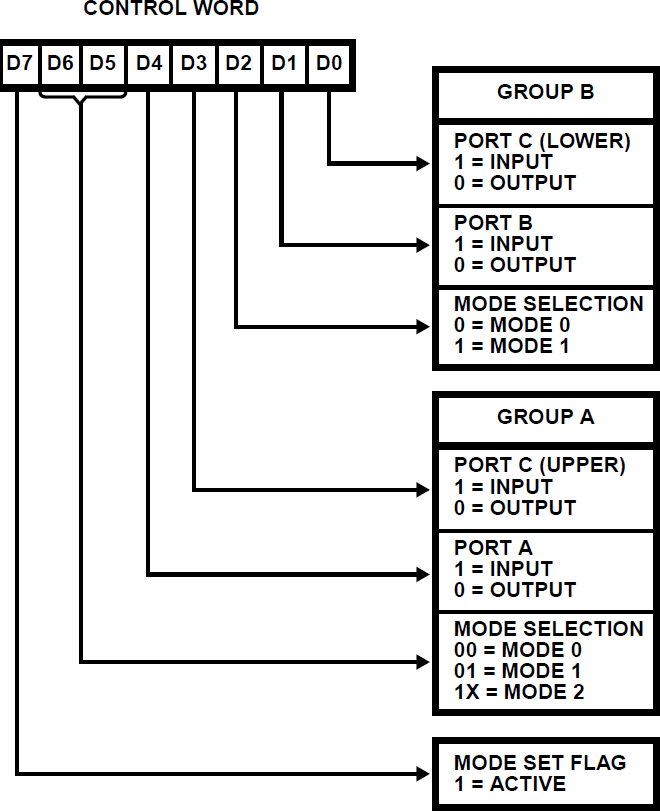
Mod 1

* Omogućuje U/I transfer uz handshake signale
* 2 grupe portova (A i B) – svaka grupa sadrži 8 bit port za podatke i 4 bit za kontrolu
* 8 bitni portovi mogu biti u ulaznom ili izlaznom režimu, i u oba slučaja su lečovani. To znači kada se postavi podatak, on se pamti dok se ne pročita, odnosno dok se ne skine iz odgovarajućih registara.
* 4 bitni portovi se koriste za upravljanje i status 8 bitnih portova. Zbog toga je port C podeljen na 2 dela.

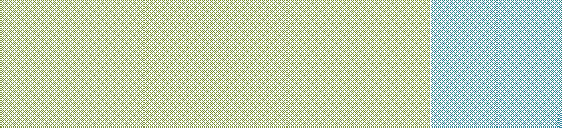
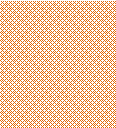
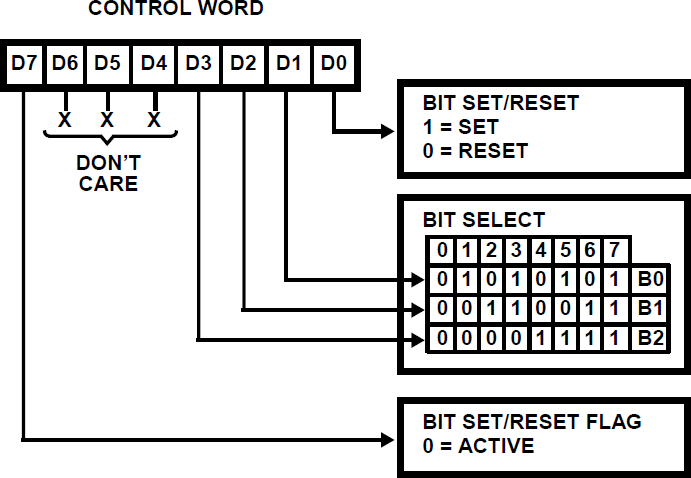
Mod 2

* Omogućuje komunikaciju sa periferijom preko jedne 8 bitne bidirekcione magistrale.
* Samo grupa A može biti u modu 2. Za handshake se koriste 5 linija porta C. I ulazi i izlazi su lečovani. Dozvoljen saobraćaj u oba smera.

Definisanje moda rada



* Za definisanje moda rada komponente koristi se KONTROLNA REČ, posebni registar, koji definiše stanje pojedinih portova i direktnim upisom vrednosti (bajta) u dati registar mi određujemo kako radi ova komponenta.
* 3 bita najmanje težine upravljaju grupom B
* 4 bita veće težine upravljaju portom A
* Bit najveće težine određuje mod rada
  + Ako je 1, definiše se mod rada ove komponente
  + Ako je 0, direktno se setuju neki od pinova na portu C
* D0 određuje niži deo porta C, niža 4 bita, da li se nalaze u ulaznom ili izlaznom režimu. Ako je postavljena 0 znači niži deo porta C je u izlaznom režimu (upisuju se podaci koje čita periferija)
* D1 isto definiše kao i D0 ali za port B, ako je 1 on je ulazni, ako je 0 on je izlazni.
* D2 određuje mod, ako je 0 radi se o modu 0 a ako je 1 radi se o modu 1, jer grupa B može biti samo u modu 0 ili modu 1
* Isto važi i za grupu A, D3 određuje ulazno izlazni mod višeg dela porta C, D4 određuje ulazno izlazni mod porta A, a poslednja 2 bita D5 i D6 određuju mod rada grupe A
* D7 mora biti uvek 1 da bismo definisali kontrolnu reč, tj da bi njeno značenje bilo režim rada komponente.



* Ako je D7 (prvi) bit jednak 0 onda zapravo kontrolna reč postavlja direktno odgovarajući bit na portu C. Zbog toga 3 bita od D4 do D6 nemaju nikakvu funkciju, D3 do D1 selektuju bit koji se menja, odnosno pin koji se menja. Ako želimo da promenimo, recimo, peti pin na portu C, stavljamo 101, a bit D0 određuje da li se on setuje ili resetuje.

Mod 0

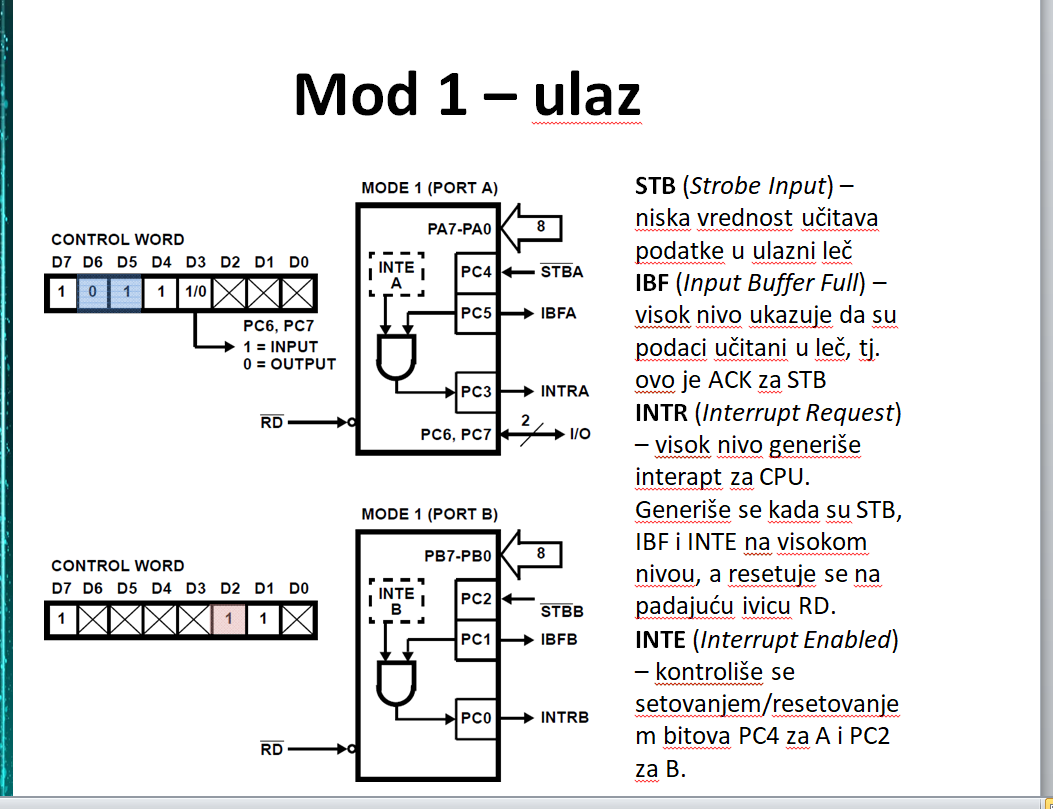
* U modu 0 kada port radi kao ulazni, mi čitamo podatak sa nekog porta, u CPU, zbog toga CPU postavlja adresu, selektuje čip, postavlja adresne linije A0 i A1, da bi pročitao sa odgovarajućeg porta. 00 pristupa portu A, 01 portu B, 10 portu C, a 11 je adresa kontrolne reči. Ako su i A0 i A1 jednake 1, i zadamo na D0 do D7 neki podatak, taj podatak će se direktno upisati u kontrolni registar i definisaće način rada komponente.
* Aktivira se read signal, pošto se čita i vrlo brzo nakon silazne ivice read signala postaje dostupan podatak na data magistrali. Potrebno je da podatak na ulazu bude stabilan pre nego što se aktivira read signal. Dok je read aktivan, validni su podaci, nakon toga nisu.

Mod 0 – primer

* Ako je D7 1 a sve ostalo 0, onda su svi portovi u izalznom režimu.
* Ako je postavljena 1 na bitu D1, port B je ulazni a svi ostali su izlazni

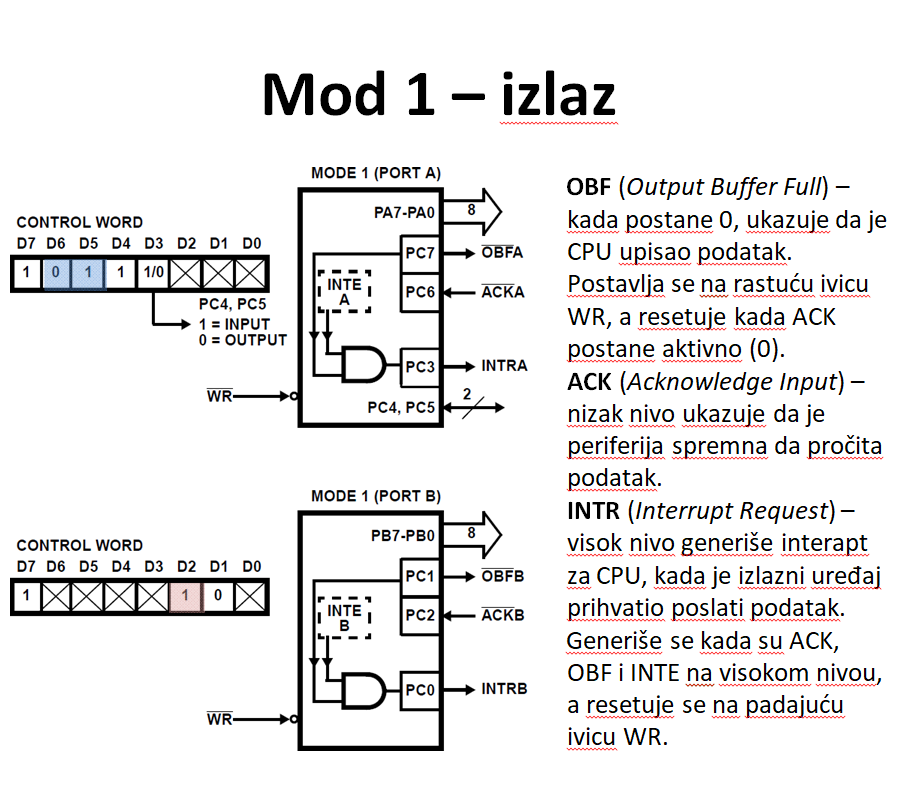
Mod 1- Ulaz

* U modu 1 port može biti ili ulazni ili izlazni i komunikacija se obavlja pod kontrolom nekih od pinova sa porta C

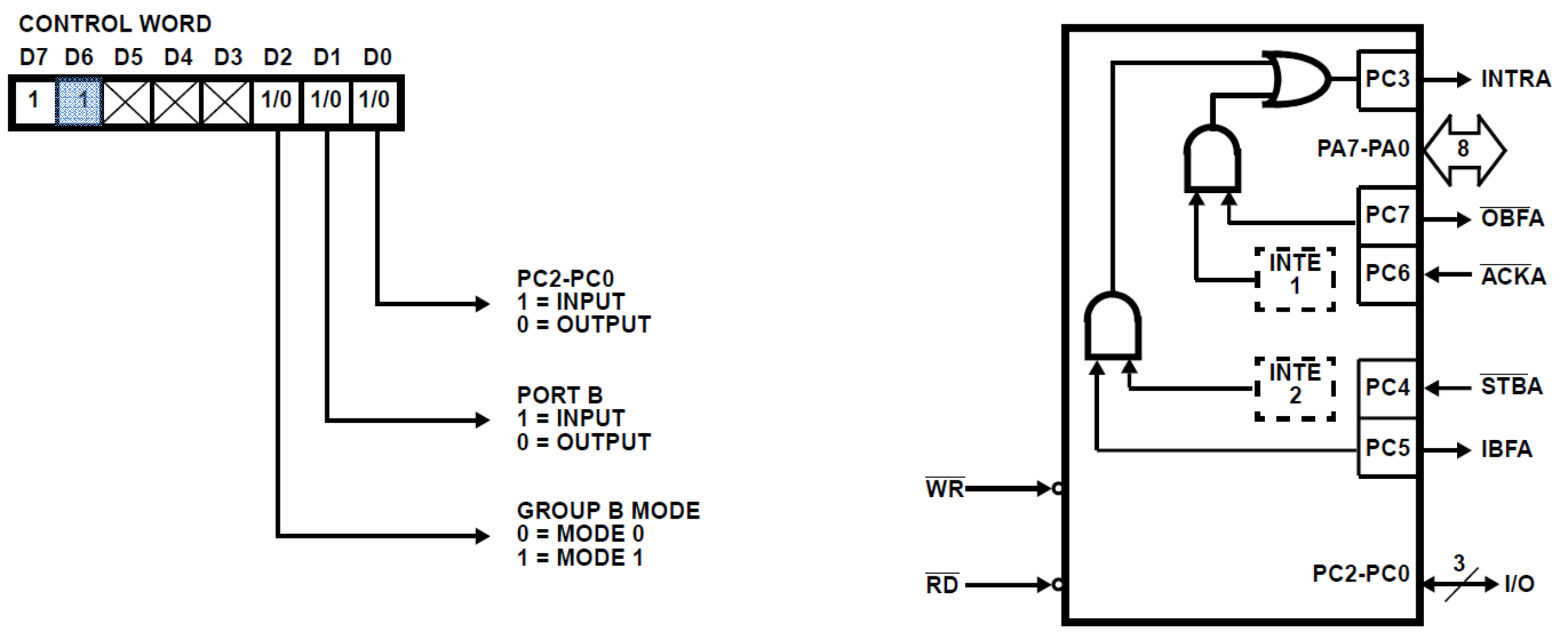


* Recimo ako je port A definisan da radi u modu 1, to znači da su D6,D5 0 i 1. Ako je D4 jednak 1, to znači da je port A u ulaznom režimu u modu 1. Strobovani ulaz znači da se određeni pinovi koriste da jave komponenti da je periferija postavila odgovarajući podatak i da može da se upiše u odgovarajući port. Na pinu 4 porta C, dovodi se signal koji se označava sa Strobe input. On je aktivan sa niskom vrednošću i njime periferija signalizira da je postavljen odgovarajući podatak i da može da se upiše na port. Na to komponenta ogovara signalom Input Buffer Full čime saopštava periferiji da je podatak upisan u bafer (registar komponente) i on će biti lečovan dok ga CPU ne pročita. Kada Strobe postane 1 ako je IBF 1 i ako je dozvoljen interrupt (postavi se 1 na pinu 4 porta C), onda se generiše interrupt prema CPU, čime se kaže da postoji podatak koji treba da pročita
* Ukoliko se radi o ulaznom režimu, recimo ovde je port B postavljen u ulazni režim u modu 1, onda postoje signali Strobe B sa istom funkcionalnošću, jedino se razlikuju pinovi koji učestvuju u definisanju tih kontrolnih signala, dakle to su pinovi Pc0 do Pc2. Ako se naredbom gde je prvi bit 0 postavi da je bit C2 jedinica, kada Strobe B postane neaktivan, a IBF je aktivan, aktivira se interrupt koji obaveštava CPU da treba da pročita taj podatak.
* Kada periferija postavi podatak na ulazni port, aktivira se Strobe signal, to znači da je podatak spreman i da 8255 može da očita podatak. Kada ga očita, setuje se signal Input Buffer full, kojim 8255 ACK periferiji da je podatak pročitan i vrlo brzo nakon toga periferija skida Strobe B. Kada je IBF aktivan i kada Strobe postane 1 (neaktivan), to aktivira interrupt CPU- u koji signalizira da procesor treba da pročita podatak. CPU onda aktivira čitanje i kada signal read postane aktivan, CPU čita podatak, nakon određenog vremena interrupt signal postane neaktivan, što znači da je podatak pročitan i nakon toga se i IBF gasi jer podatak nije validan.

Mod 1 – izlaz



* Ako je portA u modu 1, D6 i D5 su 0 i 1 respektivno Na slici iznad D4 je jednak 1, što je greška, treba da bude 0, da bi port A bio u izlaznom režimu. Pinovi Pc7 Pc6 i Pc3 imaju drugačiju funkciju.
* Obzirom da je port izlazni, potrebno je da 8255 obavesti periferiju da je podatak spreman da ga periferija preuzme. Zato se prvo aktivira OBF (sa niskom vrenošću). Kada periferija dobije taj signal ona čita podatak sa PA0 do PA7 i aktivira signal ACK da je taj podatak pročitan. Čim je podatak pročitan ukoliko je dozvoljen interrupt (dozvoljava se time što se PC6 postavi na 1), aktivira se pin PC3, odnosno signal ka CPU- u, da treba da se upiše sledeći podatak.
* Interrupt se ukida kada se javi novi upis. Isto to važi i za port B.
* Kada je aktivan interrupt, 8255 očekuje da CPU upiše novi podatak koji treba da se pošalje. Kada se javi upis, na prednju ivicu Write signala se praktično ukida interrupt i čeka se da se upiše podatak i da se pročita od strane periferije. Kada je podatak upisan, aktivira se Output Buffer Full (OBF) sa niskom vrednošću i time se signalizira periferiji da treba da očita podatak. Kada periferija pročita podatak, ona setuje signal ACK i time kaže da je videla podatak. Kada ACK postane neaktivan, on ukida OBF i kada su oba na jedinici to trigeruje interrupt ka CPU- u da se upiše sledeći podatak.

Mod 2

* U modu 2 može biti samo port A i to je bidirekcioni prenos. Kombinuje mod 1 u izlaznom i ulaznom režimu. Duplirani su signali kada podatak stiže od periferije on aktivira Strobe A i kada CPU pročita podatak, ACK-uje ga sa IBF. Kada oba postanu 1, javlja se interrupt ka CPU- u da treba da preuzme podatak. Ako je CPU upisao nešto u registar porta, signalizira se sa OBF, kada se preuzme aktivira se ACK, kada su oba 1 signalizira se CPU- u da treba da dostavi sledeći podatak za slanje. Kada se javi interrupt to znači da je podatak ili pročitan ili upisan od strane periferije.

Organizacija U/I aktivnosti

Vrste U/I prenosa

* Programirani U/I (U/I pod kontrolom CPU- a, podrazumevani način komunikacije, kada se iz programskog koda upisuje ili čita iz registra)
  + Bezuslovni – ako je uređaj uvek spreman, npr ako se radi o video kartici tj video izlazu, tu uvek možemo upisati podatak koji će biti prikazan na ekranu
  + Uslovni – proverimo stanja uređaja pre upisa podatka
* Prenos pod kontrolom prekida – napredniji način, jer omogućava CPU-u da obavlja neke druge aktivnosti dok čeka na U/I uređaj
* Direktan pristum memoriji (DMA) – postoji poseban kontroler koji upravlja tim prenosom, DMA kontroler, i CPU je oslobođen tog dela posla.

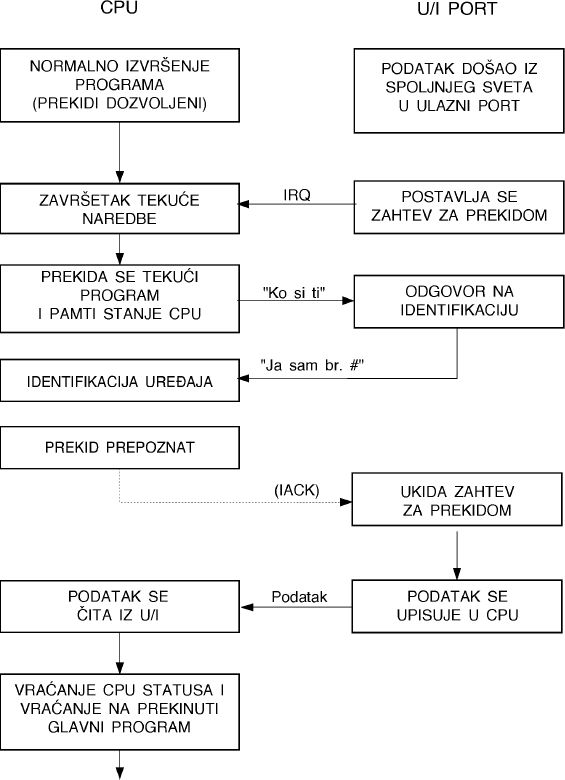
Programirani U/I

* Predstavlja najgore moguće rešenje. Ne zahteva nikakav dodatni hardver, sve obavlja sam CPU, međutim sam postupak komunikacije sa uređajima oduzima dosta vremena CPU-u. On mora stalno da vrši prozivku uređaja (sve uređaje da pita da li ima podatke za slanje). Ta prozivka može da traje relativno dugo, što je nepovoljno, jer se mnogo vremena troši na neke nepotrebne radnje. Dobro je to što nema dodatnog hardvera, ali se CPU vreme troši više na druge radnje.

Prenosi pod kontrolom prekida

* Umesto da proziva U/I uređaje, CPU obavlja druga izračunavanja sve dok ne stigne zahtev (prekid, interrupt)
* Kada stigne zahtev, CPU
  + Mora da zapamti tekuće stanje, pre svega statusni registar, jer mora da zapamti flegove da bi mogao da ispravno nastavi instrukcije. Mora da zapamti brojač naredbi da bi znao od koje instrukcije da nastavi sa izvršenjem.
  + Pre nego što izvrši proceduru za obradu prekida, mora da identifikuje koji uređaj je zahtevao prekid i na osnovu toga selektuje jednu od procedura za opsluživanje

Izvršenje prekida

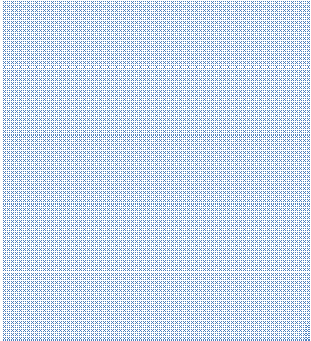
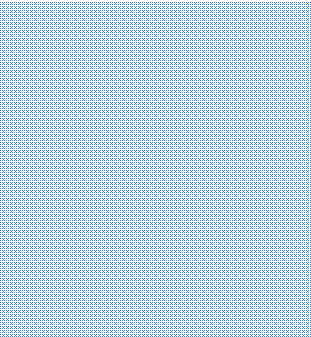
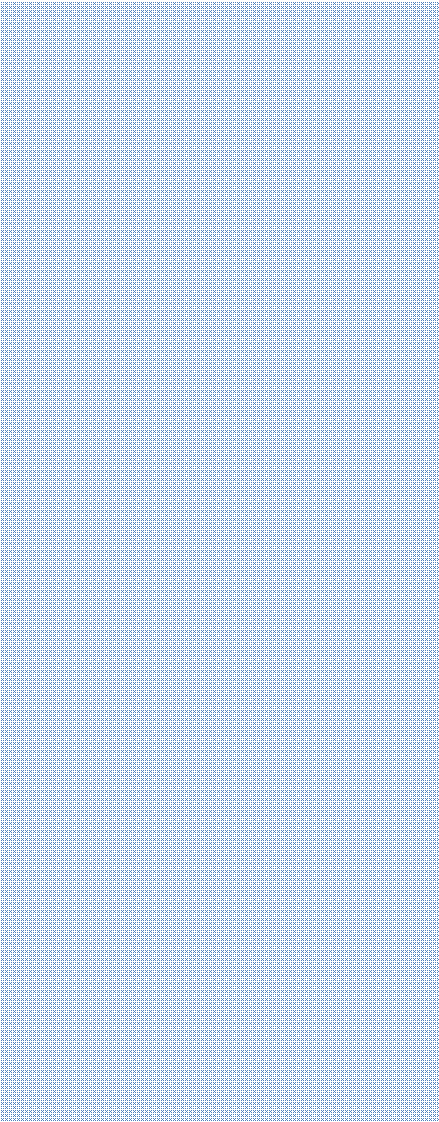
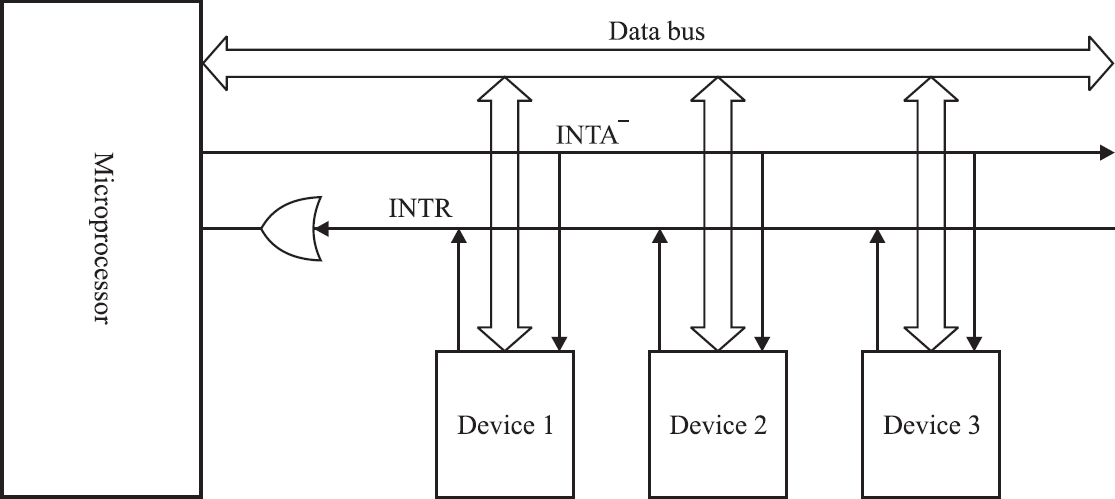


* Pretpostavka je da se na nekom U/I portu pojavio podatak. Kada se to desi, on šalje interrupt request (zahtev za prekidom,preko IRQ linije). Tim se kaže da U/I port ima nešto, što treba da se prenese.
* Za to vreme CPU izvršava normlano svoj program i u to vreme dolazi do zahteva za prekidom. To može da se desi bilo kada i ne sme se instrukcija prekinuti na pola. CPU mora da završi tekuću instrukciju (sve 4 njene faze) i tek nakon toga prelazi na obradu prekida. Pre nego što počne sa obradom mora da zapamti tekuće stanje, flegove, instruction pointer, zatim da pređe na proceduru koja će utvrditi ko je zahtevao prekid. Ukoliko postoji 1 linija za prekid, moramo imati mehanizam koji će omogućiti da se taj uređaj identifikuje. Različite su metode u zavisnosti od CPU-a (na osnovu flegova ili adrese uređaja).
* Nakon što je prekid prepoznat, šalje se acknowledge I sad uređaj može da skine prekid. Postavlja dati podatak na magistralu, CPU ga čita I nakon toga restaurira prethodno stanje sa stacka gde je zapamtio prethodno stanje.
* Prekid je
  + Proces tokom koga CPU suspenduje tekući rad radi izvršenja neke hitne operacije. U ovoj definiciji je stavljen akcenat da CPU prestaje sa izvršavanjem programa koji je započeo radi operacije koja je urgentna.
  + Proces kojim eksterni uređaj skreće pažnju CPU-a. Ova definicija objašnjava da sada CPU ne mora stalno da vrši prozivku, već mu se uređaji sami jave kad su spremni za prenos podataka.
  + Poziv potrograma iniciran od strane spoljašnjeg uređaja kroz odgovarajući hardver (hardverski prekid) ili samog CPU-a (softverski prekid) – Prekid inicira izvršenje procedure. Aktivira se nešto što je organizovano u funkciju. Aktiviranje te funkcije treba da bude odgovor na događaj koji stiže spolja ili samog programa. Iz aplikacije se mogu pozvati funkcije preko prekida, a može i sam program inicirati prekid (desi se overflow steka recimo)
  + Asinhroni hw signal koji skreće pažnju na nešto ili sinhroni softverski događaj koji ukazuje na potrebu za promenom u toku izvršenja. Ova definicija potencira asinhroni prekid spolja, što ima prednosti, jer trenutno kada se desi signal spolja, sistem reaguje na tu promenu, aktiviranjem procedure za opsluženje tog prekida. Sinhrono se javlja ako je zovemo iz koda (same instrukcije aktiviraju proceduru)
* Prednosti
  + Obezbeđuje fleksibilnost da programi višeg prioriteta mogu biti izvršeni van redosleda. Sve ono što je neophodno da se izvrši trenutno, se smešta u okviru procedure za obradu prekida, a aktivira se asinhronim događajem kada se javi potreba za njim. Ovo povećava fleksibilnost, jer ima mogućnost paralelnog (simultanog) izvršenja koda pri čemu se uvode prioriteti.
  + Prekidi ne moraju biti sinhronizovani sa glavnim programom (najčešće i nisu), ovo je istovremeno i mana, jer se teže debagiraju takvi programi, obzirom da se u bilo kom trenutku može desiti prekid i ako se ne predvidi sve moguće situacije, taj program neće da se ponaša kako očekujemo
  + Obezbeđuju odgovarajuće vreme odziva, u roku od nekolikotaktova od trenutka javljanja zahteva za prekidom, aktivira se procedura koja ga opsluži, tako da ako želimo da imamo sistem koji radi u realnom vremenu, da reaguje u kratko vreme na spoljne pobude, jedino putem prekida možemo da obezbedimo

Organizacija prekida

* Jednonivovska
* Višenivovska
* Vektorska

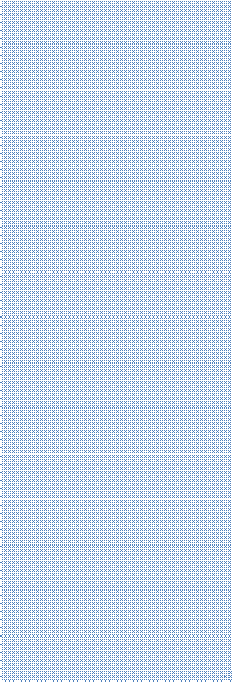
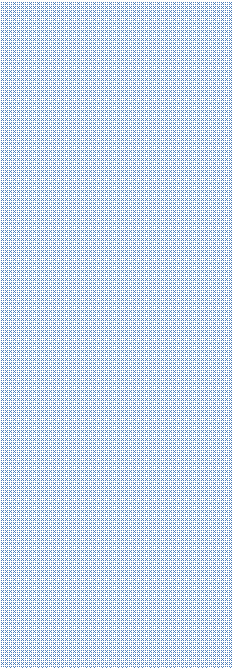
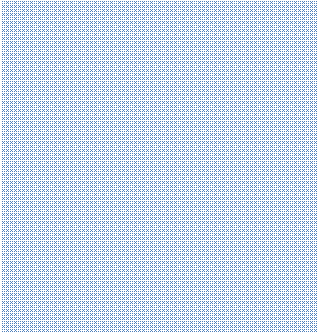
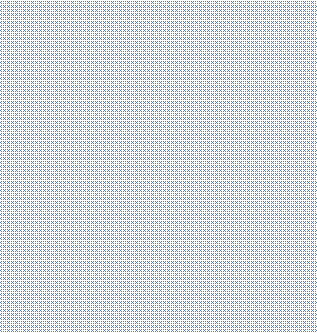
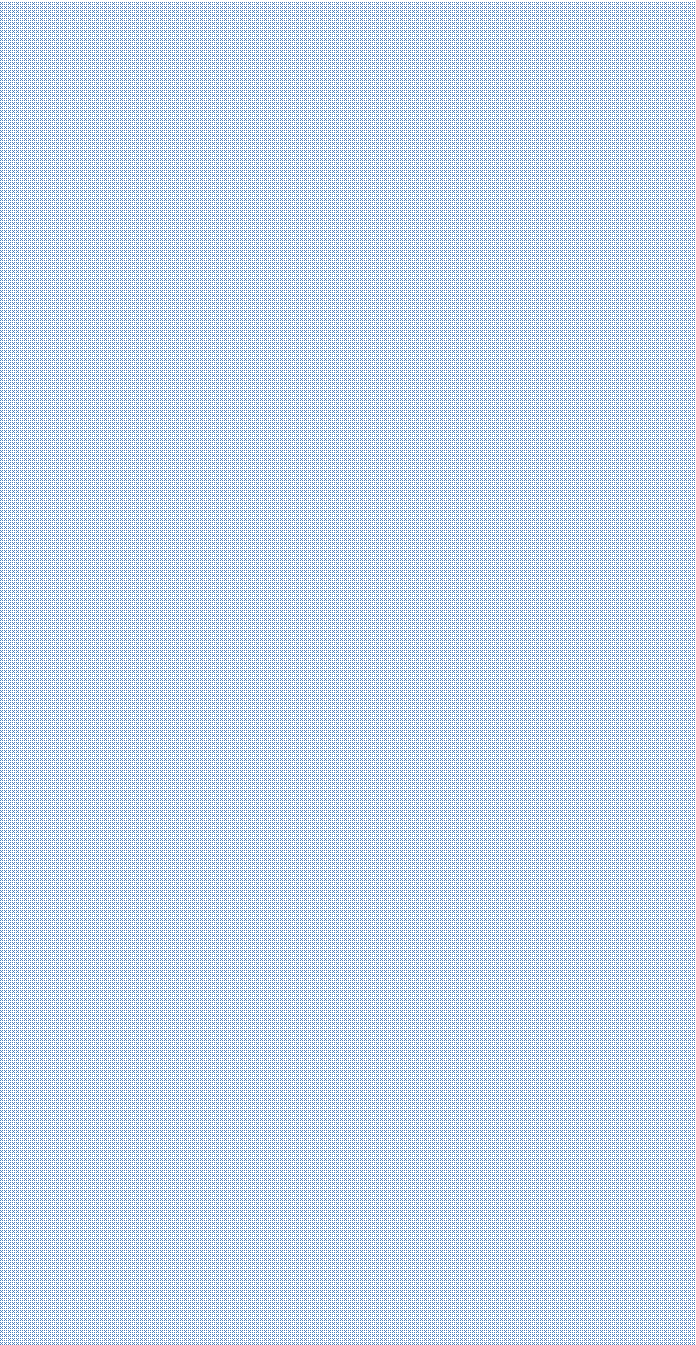
Jednonivovska organizacija



* Svi uređaji su povezani žičanim direktnim spajanjem na odgovarajuću žicu sa interrupt request linijom koja se dovodi na CPU i kada bilo koji od njih želi prenos podataka ili bilo kakvu obradu, onda se javlja signal na INTR liniji. Tada se upravljanje prenosi na posebnu proceduru koja treba da utvrdi koji uređaj inicira prekid. To omogućuje da se uvedu prioriteti, što znači da će uređaj na nižoj adresi (bliži CPU-u) pre biti uslužen. Da se ne bi desilo da neki uređaji umiru od izgladnelosti, tj. Nikada se ne dolazi do njih, može se uvesti rotacija prioriteta, odnosno da se zapamti do koje adrese je CPU stigao pa onda da krene sledeći put od te adrese
* Kada se primi zahtev potrebno je da se utvrdi adresa uređaja, zatim se upravljanje prenosi na proceduru za obradu prekida. Ta procedura zavisi od uređaja i obrađuje neophodno za taj prekid. Da se ne bi desio novi prekid dok se uslužuje jedan od njih, najčešće se zabranjuju svi ostali prekidi. Postoji flag koji to definiše i to nam obezbeđuje da se procedura ne prekine dok traje obrada prekida. Pre nego što se vratimo u glavni program mora se skinuti zabrana (omogućiti prekid), vraća se stanje CPU-a na normalno izvršenje programa

Višenivovska organizacija

* Imamo više linija kojima se signalizira prekid i za to je najčešće potrebno da postoji poseban kontroler, preko koga CPU komunicira sa U/I uređajima. Prednosti: imamo više nivoa prioriteta, koji se mogu dinamički menjati. Pamti se stanje – kada neko zahteva prekid, to se lečuje u nekom registru i po opsluživanju prekida višeg prioriteta može se opslužiti prekid nižeg prioriteta.



Klasifikacija prekida

* Prema načinu aktivacije
  + Aktiviranje ivicom ulaznog signala (koriste se za prekide visokog prioriteta kao što je nemaskirajući hardverski prekid i zahtevaju interni flip-slop da bi se zapamtilo stanje, pošto traje vrlo kratko, ivica aktivira signal. Zašto baš za nemaskirajuće? Ako se desi prekid napajanja, nešto na šta se mora odmah reagovati da bi se računar ugasio regularno i u tom slučaju potrebno je dok se ne završi instrukcija da se zapamti stanje tog ulaza)
  + Aktiviranje nivoom (visokim nivoom najčešće) - zadržava se nivo dok CPU ne potvrdi prihvatanje
* Prema mogućnosti maskiranja
  + Maskirajući – prekidi nižeg nivoa koji se mogu ignorisati, tj. Maskkirati resetovanjem IE flega. Primer INTR)
  + Nemaskirajući – prekidi višeg nivoa koji se ne mogu ignorisati
* Prema izvoru
  + Hardverski (nemaskirajući i maskirajući)
  + Softverski (INT0 – INT255 mogu se pozvati iz koda instrukcijom. Broj određuje koju proceduru za obradu prekida zovemo. INTO (isto kao i INT4 )
* Prema načinu smeštanja adrese rutina za opsluživanje prekida (ISR)
  + Vektorski (adrese su smeštene na određenim lokacijama u okviru nekog vektora)
  + Nevektorski ( kada uređaj koji zahteva prekid nakon što se potvrdi zahtev, šalje adresu procedure koja treba da obradi taj prekid)

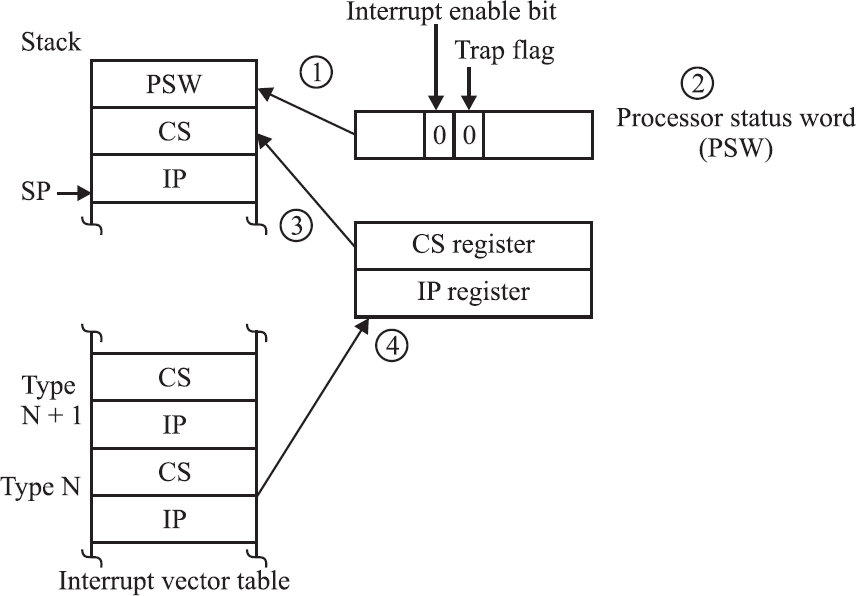
Prekidi procesora 8086

* Unutrašnji ‐ prekidi kao posledica greške izvršenja instrukcije (npr. deljenje 0)
* Hardverski – posledica signala NMI (nemaskirajući) i INTR (maskirajući); obzirom da postoji samo jedna INTR linija, koristi se 8259A programabilni interapt kontroler za upravljanje prekidima
* Softverski – generišu se programski, pozivom INT ili INTO

Prekidna sekvenca

* Kada se desi prekid, izvršava se standardni niz koraka

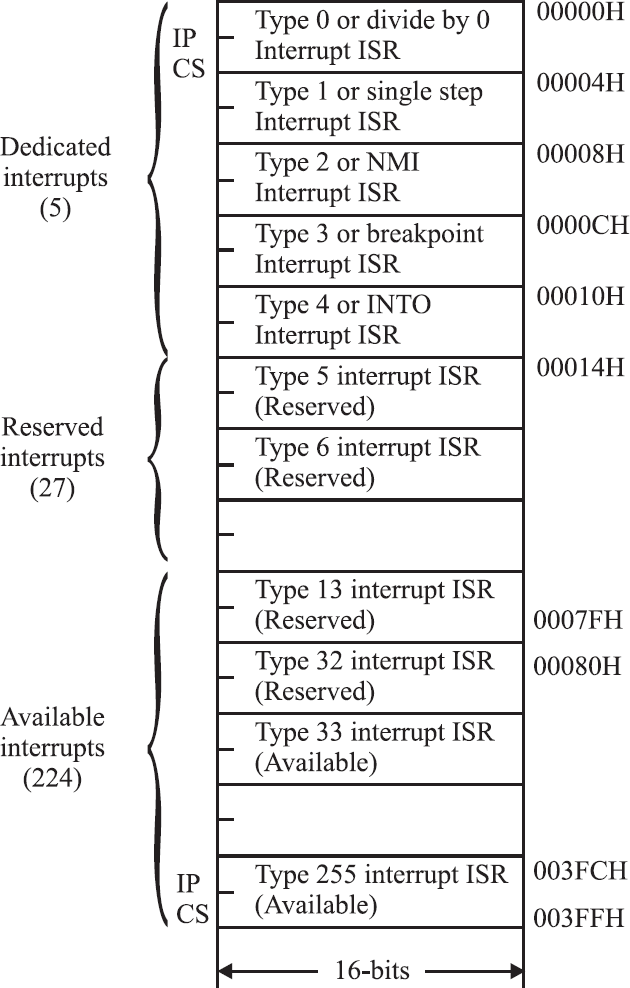
1. Flag registar (Flegovi) se stavlja na stek. Obzirom da ćemo zabraniti dalje prekide dok traje obrada tekućeg, moramo zapamtiti flegove (sa PUSHF)
2. Treba onemogućiti dalje prekide INTR ulaz se onemogućuje (resetovanjem IE flega) i Trap Flag (TF) se resetuje. TF se koristi za step by step izvršenje programa, dakle prekidna procedura se aktivira nakon svake instrukcije, zgodno za debagiranje
3. Code segment (CS) registar se stavlja na stek
4. Instruction Pointer (IP) registar se javlja na stek (u CS i IP se upisuju adrese koje se čitaju iz prekidnog vektora i nakon toga se kreće sa izvršenje procedure sa te adrese)
5. U IP registar se stavlja adresa prekidnog vektora (offset)
6. U CS registar se stavlja adresa segmenta prekidnog vektora
7. ISR se izvršava sa adrese dobijene na osnovu vrednosti CS i IP registra



* Pamti se stanje vekotra na steku, resetuje se IE i Trap fleg, pamti se CS i IP na steku, a iz vektora interrupta se preuzimaju nove vrednosti za CS i IP

Tabela (vektor) interapt pokazivača

* Ova tabela je standardna kod 8086, uvek se nalazi na početku adresnog prostora u prvom kilobajtu i sadrži parove CS i IP vrednosti za svaku od prekidne procedure. Pošto su ovi registri 16-bitni potrebna je jedna 32 bitna vrednost da bi se zapamtilo njihovo stanje. Zbog toga mi zapravo vrednosti CS i IP registara dobijamo tako što indeks u toj tabeli (na osnovu broja interrupta određujemo indeks) i množimo sa 4 da bismo dobili offset u bajtovima odakle počinju ove vrednosti
* Prvih 5 interapta su specijalne namene. Narednih 27 su rezervisani za naredne procesore i ne mogu se koristiti. Preostalih 224 su na raspolaganju korisniku za smeštanje ISR adresa



* INT0 za obradu deljenja nulom je INT0
* INT1 Na 4H je single step interapt koji se aktivira kada je setovan trap fleg, kod izvršava instrukciju po instrukciju.
* INT2 je nemaskirajući hardverski prekid. Na NMI kada dođe signal, skače se na 8H adresu
* Type3 je breakpoint prekid koji se koriti prilikom debagiranja
* Type4 je overflow koji se uslovno izvršava ako je setovan overflow fleg
* Ostali su 27 su za specijalne namene
* CS se nalazi na višoj a IP na nižoj adresi (nulti bajt je niži bajt IP, prvi viši bajt IP, treći bajt je niži bajt CS, četvrti je viši bajt IP)

Povećanje broja INTR prekida (harverskih maskirajućih)

* 8086 obezbeđuje samo jedan INTR ulaz
* 8259 omogućuje da imamo višenivovske maskirajuće prekide hardverske, odgovarajuća INTR linija se vodi na INTR ulaz CPU-a i on je taj koji određje koji će od prekida biti izvršen. Prioriteti mogu biti fiksni ili se mogu menjati tokom izvršenja. CPU odgovara na zahtev u 2 mašinska ciklusa (svaki je po 4 takta) (dvaput šalje int acknowledge) u prvom potvrđuje da je prihvaćen interapt i omogućuje se kontroleru da odredi koji od zahteva trenutno ima najviši prioritet, a u drugom ciklusu kontroler postavlja adresu da bi moglo da se zna na koju CPU da skoči prilikom obrade. Sad možemo da povećamo broj ulaznih linija u startu ih imamo 8, samo jedna se prosleđuje CPU-u ali će se zapamtiti svi ostali zahtevi. Nakon ove obrade ide sledeći i tako dalje. Da bismo znali koju od procedura da aktiviramo, u drugom mašinskom ciklusu se šalje preko data magistrale samo jedan bajt koji određuje kako broj (indeks) u vektoru određuje proceuru za obradu.

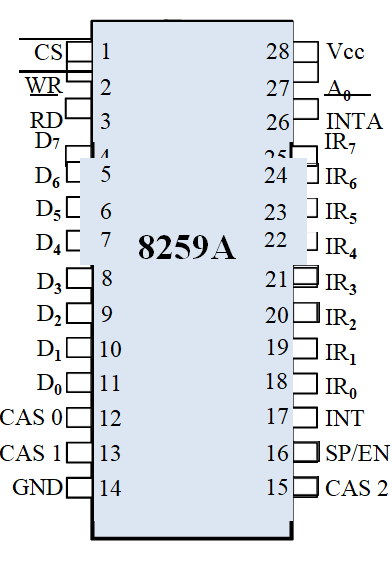
Softverski prekidi

* Aktiviraju se pozivom INTn ili INTO
* Broj odgovara rednom broju u tabeli interapt pokazivača
* Postoji 256 bezuslovnih (INTn) i jedan uslovni (INTO) prekid (INTO izvršava INT4 ako je setovan OF)
* Koriste se za:
  + Testiranje hardverskih prekida – da vidimo kako se ponaša neki prekid. Poziva se sinhrono
  + Poziv opštih proceura
  + Poziv sistemskih I/O procedura, kad se komunicira sa uređajem spolja

Prioriteti prekida

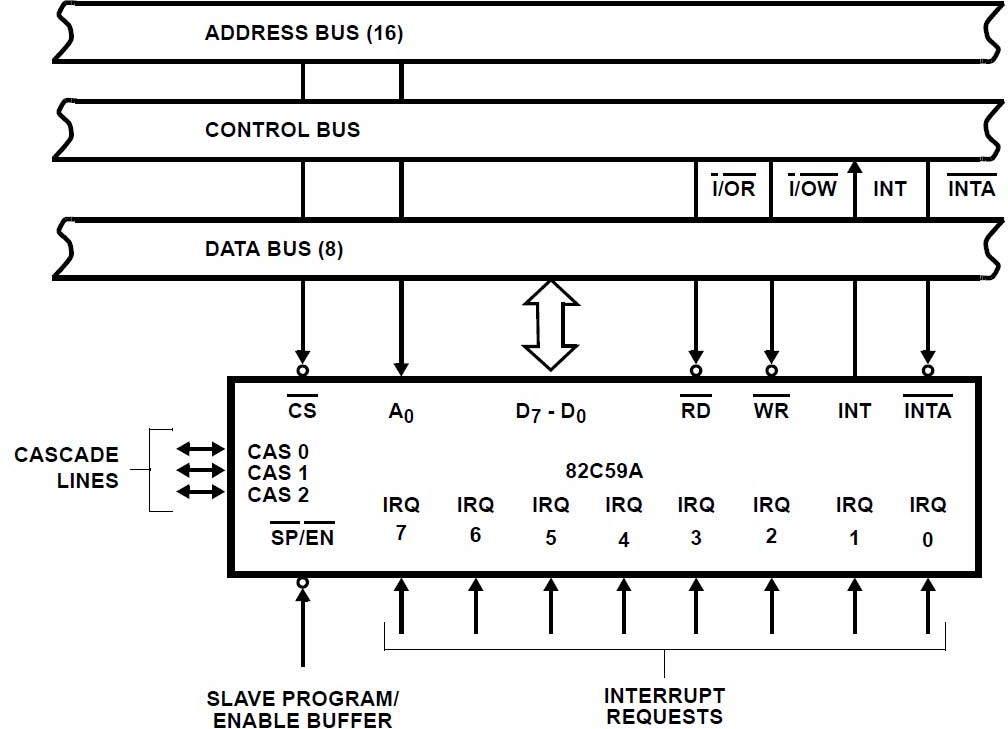
1. Unutrašnji prekidi i izuzeci
2. Softverski prekidi – INTn, INTO
3. Nemaskirajući hardverski prekidi – preko NMI pina
4. HW maskirajući prekidi – preko INTR
5. Single step

8259A



* Omogućava 8 nivoa prekida, može se proširiti na 64, kaskadnim povezivanjem. Može se programirati način rada i maskirati svaki od zahteva pojedinačno
* Ima 28 pinova
* Od D0 do D7 služe za povezivanje na magistralu podataka, bidirekciona magistrala i služi za prijem upravljačkih instrukcija za podešavanje načina rada, preko njih se čitaju statusi (recimo stanje aktivnih prekida) takođe se preko njih šalje deo adrese procedure za obradu prekida.
* Linije CAS0 CAS1 i CAS2 služe za kaskadno povezivanje ovih kontrolera i u slučaju master komponente na ovim pinovima se postavlja adresa slejva, a u slučaju slejv komponente to su ulazni pinovi i adresa koja je postavljena njima se poredi sa sopstvenom adresom da bi se selektovao odgovarajući čip
* SP/EN je pin koji određuje mod rada, ako se koriste u nebaferovanom režimu onda jedinica na ovom pinu kaže da se radi o master, a nula da se radi o slejvu. Ukoliko se radi u baferovanom režimu onda se ovim pinom enabluje bafer koji je transiver preko koga je ova komponenta vezana na data magistralu. Ako na data magistrali ima više uređaja koji su priključeni da bi mogli da se svi oni napajaju, potrebno je da postoji odgovarajući bafer, jer izlazi samih komponenti nisu dovoljno snažni da mogu da napajaju veći broj uređaja
* Interapt request linije su osnovne linije preko kojih se primaju zahtevi za prekidima i to su linije koje se vezuju na periferiju od kojih dolazi zahtev za komunikaciju sa CPU. Mogu da se aktiviraju ivicom ili nivoom i to se određuje u režimu rada ove komponente. Kada se javi zahtev na nekoj o INT pinova ova komponenta prosledi zahtev prema CPU linijom interapt, CPU odgovara u 2 instrukcijska ciklusa. U prvom potvrdi zahtev i selektuje jedan od ulaza, a u drugom se na data linijama postavlja offset ili indeks odgovarajuće procedure u okviru tabele procedura za obsluživanje prekida
* A0 selektuje komandnu reč koja se postavlja i najčešće se vezuje na adresnu liniju A1 procesora 8086.

Interfejs prema standardnoj sistemskoj magistrali



* Preko neke dekoderske mreže povezuje se CS na adresne linije, pri čemu ćemo liniju A1 da dovedemo na A0 komponente 8259A da bismo izabrali koji registar postavljamo, data pinove direktno povezujemo na data magistralu, a preko neke dekoderske mreže se povezuju read i write signali uz kombinaciju sa mem/io pinom da bismo izabrali opciju čitanja statusa ili upisa neke kontrolne reči. Ako se javlja zahtev na nekoj INT request linija aktivira se odmah INT linija koja se vezuje na INT ulaz procesora, a INTA se vezuje na 8259 direktno. Za povezivanje više kontrolera međusobno koriste se linije CAS0 – CAS2 gde se postavlja adresa sledeće komponente koja treba da postavlja adresu prekida i SP/N pin kaže da li se radi o masteru ili slejvu. Izlazna je komponenta ako radimo u baferovanom režimu ovde nije taj režim ako napravimo veći broj uređaja preko data magistrale onda je potrebno da postoji neki transiver koji će sa dovoljno struje napajati data magistralu

8259A blok šema



* Kada se javi zahtev na nekom od ulaza, bilo da se radi o prednjoj ivici ili naponskom nivou, to se upisuje u interapt request registar (IRR). Ovaj registar sadrži jedinice na svakoj poziciji gde se javlja zahtev za prekidom. Kada god postoji neki zahtev, ukoliko nije maskiran prosleđuje se interapt request ka CPU- u, odnosno aktivira se interapt linija. Interapt mask registar (IMR) služi za maskiranje pojedinih ulaza. Gde god se nalazi jedinica u interapt mask reg, taj ulaz je onemogućen. Može se vrlo lako zabraniti neko od njih dinamički tokom izvršenja programa. Ako nije maskiran neki od ulaza jedinica na bilo kom od tih ulaza će aktivirati interapt request.
* Kada stigne INTA od CPU- a i to prvi INTA, na osnovu priority resolvera, određuje se koji od tih zahteva ima najviši prioritet i on se upisuje u IN-Service registar (ISR). ISR registar ima jedinicu na poziciji interapta koji se trenutno opslužuje.
* U drugom INTA ciklusu se odgovarajuća adresa postavlja na data magistralu da bi se selektovala procedura koja treba da opsluži zahtev. Rewrite logic služi da se izabere jedan od registra u koji će se upisati kontrolna reč ili iz koga će se pročitati status. Može se pročitati IRR da vidimo šta je sve setovano, ili IMR da vidimo šta je maskirano.
* A0 služi da se izabere jedna od kontrolnih reči. Postoji interni brojač i u zavisnosti od toga od prve kontrolne reči koji je ciklus, zavisiće kako će se tumačiti svaka od tih reči koja je upisana
* Kada se setuje bit u ISR automatski se resetuje IR bit, da bi se stavilo do znanja da se krenulo u opsluživanje tog prekida i sledeća promena nivoa na ulazu se tumači kao novi zahtev.
* Bit ISR ostaje sve dok ne naiđe zadnja ivica drugog INTA signala kada se briše, ukoliko je setovano automatsko resetovanje tog bita. Ako nije to setovano, to se radi iz prekidne procedure eksplicitnim brisanjem tog bita.

Dodatni opisi blokova šeme

* IRR (Interrupt request register) – smešta sve zahteve
* ISR (In-service register) – smešta interapte koji se trenutno opslužuju
* PR (Priority Resolver) utvrđuje prioritet bitova setovanih u IRR i setovani bit najvišeg prioriteta prosleđuje u odgovarajući bit ISR tokom INTA impulsa
* IMR (Interrupt mask register) maskira bitove u IRR
* Read/Write logic – prihvata komande od CPU i šalje status. Sadrži Initialization Command Word (ICW) i Operation Command Word (OCW) reigstre koji upravljaju radom komponente.
* Cascade Buffer/Comparator – smešta i poredi ID jeve svih 8259A u sistemu. Master šalje ID slejva na CAS0-CAS2 linije, čime se selektuje odgovarajući slejv koji šalje adresu prekidne procedure na magistralu podataka tokom narednog jednog ili dva uzastopna INTA impulsa .

Prekidna sekvenca

1. Jedna ili više IR linija prelaze na visok nivo, čime se postavljaju odgovarajući bitovi u IRR
2. 8259A evaluira zahteve i šalje INT ka CPU
3. CPU potvrđuje INT slanjem INTA impulsa (aktivna niska vrednost)
4. Po pristizanju INTA impulsa, bit najvišeg prioriteta u ISR se postavlja, a odgovarajući bit u IRR resetuje. Magistrala podatak nije aktivna u ovom ciklusu. U kaskadnom režimu, master koristi ovaj ciklus za selekciju slave‐a.
5. 8086 šalje drugi INTA impuls, tokom koga 8259A šalje 8‐ bitnu adresu (pointer) na magistralu podataka.
6. U AEOI modu, ISR bit se resetuje na kraju drugog INTA impulsa. U protivnom, ISR bit ostaje postavljen dok se ne izda EOI komanda na kraju rutine za obradu prekida.

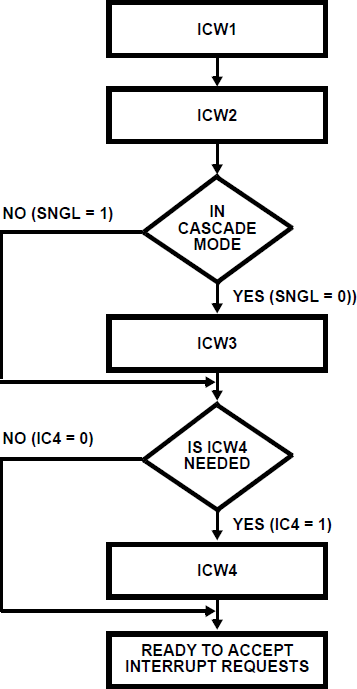
* Ako nijedna IR linija nije aktivna u koraku 4 (zahtev je trajao suviše kratko), 8259A izdaje interapt level 7.
* Čim pristigne neki IR, INT postaje aktivan.
* Ako naiđe prekid višeg prioriteta između dva INTA impulsa, INT postaje neaktivan neposredno nakon drugog INTA impulsa.
* Nakon neodređenog vremena INT ponovo postaje aktivan, da ukaže da prekid višeg prioriteta čeka na obradu. To neaktivno vreme varira.

Programiranje 8259A

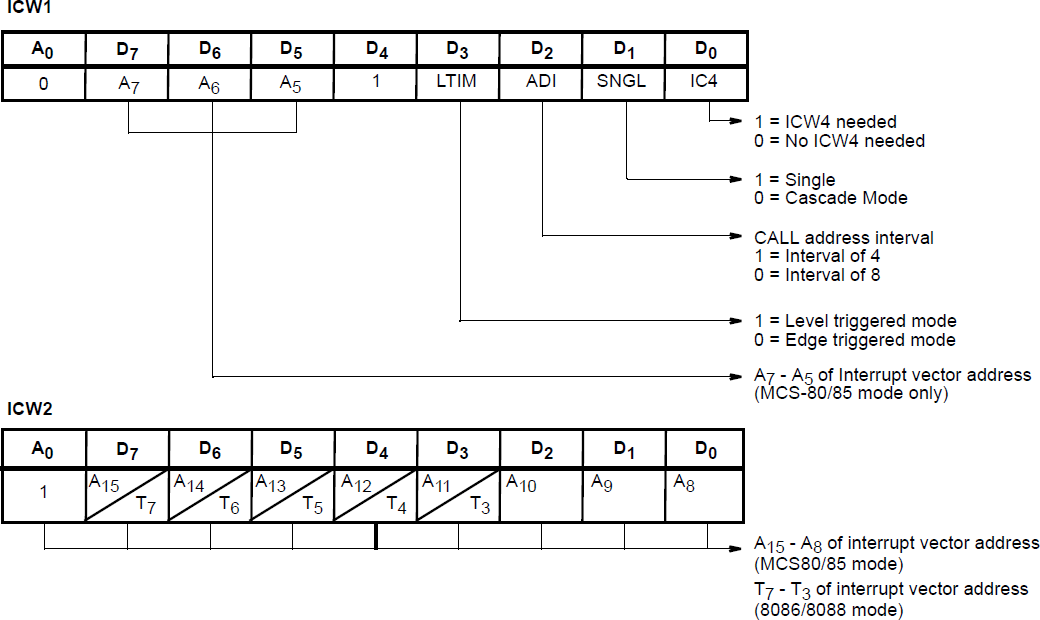
2 tipa komandi

* ICW – inicijalizacija se obavlja sekvencom od 2 ili 4 bajta, taktovanih WR impulsima. Njima se definiše rad same komponente. Javljaju se na početku rada, a nakon toga u proizvoljnom trenutku mogu da se jave OCW reči
* OCW – definišu mod rada komponente
  + Fully nested mode
  + Rotating priority mode
  + Special mask mode
  + Polled mode

Inicijalizaciona sekvenca



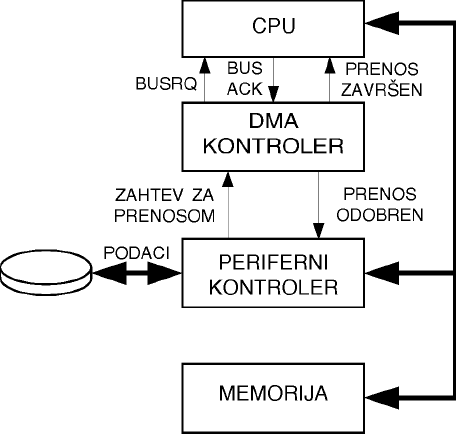
* Uvek se javljaju 2 reči na početku ICW1 i IC2. Ako se radi o kaskadnom modu onda postoji i ICW3 reč. Ukoliko se javi potreba za ICW4 onda se i ona javlja. Za 8086 ICW4 je obavezan.
* ICW1 – karakteriše se time što je na adresnoj liniji 0 (A0=0) a na data liniji 4 (D4 = 1), time se resetuje counter za inicijalizaciju. Svaka sledeća write naredba će značiti da prelazi u sledeću inicijalizaciju reči.
* ICW1 definiše:
  + Kako se 8259A koristi u sistemu (pojedinačno ili kaskadno),
  + Da li se prekidi okidaju ivicom ili nivoom
  + Da li je potrebna ICW4
* Kada se izvrši:
  + Resetuju se ulazna kola za IR (interapt request). Ako je bio setovan neki interapt, odgovarajući flip flop će biti postavljen na nisku vrednost i potreban je da se javi novi impuls da bi ga okinuo.
  + IMR se briše, omogućavaju se svi ulazi
  + IR7 ulazu je dodeljen prioritet 7 (najniži), a IR0 prioritet 0 (najviši).
  + Slave mod adresa je postavljena na 7.
  + Special Mask Mode se briše, a Status Read se postavlja na IRR. Ukoliko pročitamo status. Pročitaćemo zapravo interapt request registar
  + Ako je IC4=0 (bit najmanje težine u ICW1, tj. vrednost na D0 liniji), tada se sve funkcije selektovane u ICW4 resetuju.
  + D0/IC4 – definiše da li je potrebna ICW4 (ako je IC4=0 nije potrebna; za 8086 je uvek 1).
  + D1/SNGL – (single) definiše da li je samo jedna 8259A komponenta u sistemu (SNGL=1) ili je više kaskadno vezanih (SNGL=0), pa je potrebna i ICW3.
  + D3/LTIM – (level triggered interrupt mode) definiše da li se zahtev za prekidom okida prednjom ivicom (LTIM=0) ili visokim nivoom (LTIM=1); nivo mora biti spušten pre izvršenja EOI komande, inače će biti protumačen kao novi zahtev
  + Ovo ne mora za ispit, samo orijentaciono
* ICW2
  + Zadaje se uvek nakon ICW1
  + A0 mora biti postavljeno na 1
  + Definiše adresu ISR u memoriji
  + Za 8085 to je viši bajt ISR adrese
  + Za 8086/88 koriste se samo D7‐D3 koji zajedno sa rednim brojem IR (pošto ih ima 8, to su zapravo D2‐D0) da formiraju 8‐bitni broj (u okviru vektora adresa ISR) koji definiše redni br. prekida
  + Npr. ako je D7‐D3=00011, a zahtev je stigao sa IR6, dobija se 00011110 tj. 1EH (30), što znači da se CS i IP za ISR počinju od adrese 78H(120).



DMA

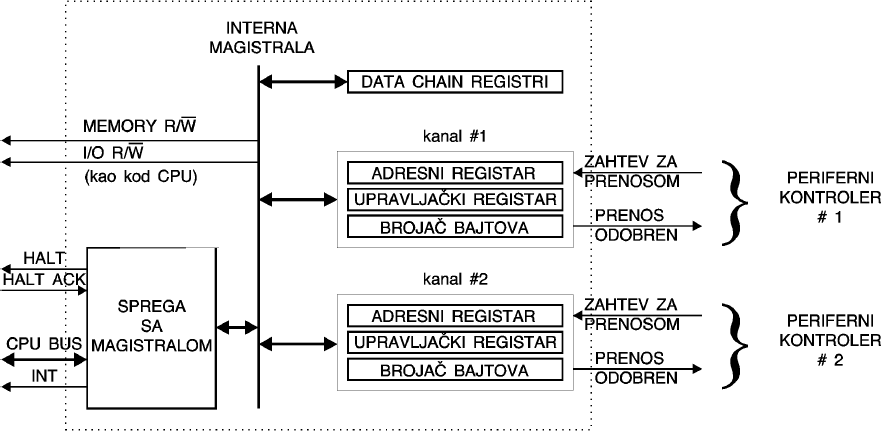
* Prekidi obezbeđuju brz odgovor perifernom uređaju, ali se opsluživanje obavlja softverski
* Nekada je potrebno da se prenos obavi brže nego što to omogućuje procedura za obradu prekida
* Nekada se čitav prekidni program izvršava samo da bi se preneo jedan bajt (prekida se rad CPU, snima stanje, uzima podatak iz izvora, upisuje se u odredište, vraća se prethodno stanje)
* Prekidi su dobri samo za spore periferije
* DMA:
  + Za brze periferije, prenos se prepušta specijalizovanom hardveru
  + On obezbeđuje brz transfer podataka, zaobilazeći CPU
  + DMA kontroler preuzima upravljanje sistemskom magistralom za vreme prenosa podataka i vraća upravljanje CPU nakon završetka
  + DMA po svojoj prirodi bi trebalo da bude brži transfer zbog toga što nemamo onaj klasični pristum (fečovanje, dekodiranje instrukcija, pribavljanje podataka), to se sve radi automatizovano sa svakim ciklusom se vrši inkrementiranje adresa, automatski bez pribavljanja novih naredbi, smanjuje se broj podataka koji treba da se prenese i generišu se signali za čitanje ili upis. Sve ovo radi hardver nezavisno od CPU-a.

Blok šema DMA



* Kada periferni kontroler ima podatke za prenos, recimo neki kontroler hard diska, napunjen je izlazni bafer i želi da prenese te podatke u memoriju. Sve ovo je pod kontrolom CPU- a. CPU je taj koji je inicirao prenos, a on će se obaviti kada se za to steknu uslovi
* Kada se steknu uslovi, taj kontroler šalje zahtev DMA kontroleru koji je arbitar i koji može da primi zahteve od više periferija. Svaka periferija ima svoju adresu. Sve periferije za brzi transport se povezuju na DMA kontroler. On određuje ko ima najveći prioritet.
* DMA zatim šalje BUS request (BUSRQ) CPU-u da oslobodi magistralu. Kada CPU završi sa tekućom instrukcijom ili kada smatra da treba da prenese upravljanje, aktivira BUS ACK i time dozvoljava kontroleru da on postane arbitar na magistrali
* Kontroler dozvoljava prenos perifernom kontroleru i obavlja se direktni prenos između porta perifernog uređaja i nekih memorijskih lokacija
* Po završetku prenosa, skida se BUSRQ, i vraća se upravljanje magistralom CPU- u

Šta CPU šalje DMA kontroleru da bi ga inicijalizovao da obavi jedan takav transfer

* Početnu adresu bloka koji treba da se prenese, najčešće se prenos vrši u blokovima.
* Veličinu bloka (u bajtovima ili rečima)
* Smer prenosa (memorija‐ka‐uređaju, uređaj‐ka‐memoriji)
* Port ID (adresa U/I uređaja koji zahteva prenos)
* Akcija na kraju prenosa (da li se zahteva po završetku prekid ili ne, samo da bi se obavestio CPU)

Struktura DMA kontrolera

* Mora da ima mogućnost sprege sa magistralom, da može da generiše HALT zahtev (HOLD) signal kojim se zahteva magistrala i HALT ACK da može da se preuzme upravljanje magistralom.
* Kada se preuzme magistrala, potrebno je da mogu da se prenose podaci, pri čemu ukoliko je prenos između 2 mem lokacije, ti podaci ne protiču kroz DMA kontroler već idu direktno od izvora ka odredištu. Po završetku se generiše interapt koji obaveštava CPU da je prenos završen. Dok se vrši prenos preko magistrale, DMA kontroler je arbitar i on mora da generiše memori R/W i I/O R/W signale, jer on upravlja ko kad postavlja i ko treba da prihvati podatak.
* Podaci vezani za prenos se smeštaju u pomoćne strukture koji se nazivaju DATA CHAIN registri, jer možemo da imamo više DMA prenosa koji čekaju na obradu. Oni koji su trenutno aktivni imaju svoje kanale. Svaki DMA kontroler ima više DMA kanala. Obavlja se samo jedan prenos preko aktivnog kanala, zavisi od prioriteta
* Svaki kanal prima zahtev i kada se odobri prenos šalje odgovor da je odobren prenos
* Za svaki od kanala postoji adresni registar za adresu sa koje se čitaju podaci ili gde se upisuju. Brojač bajtova se dekrementira do 0 odnosno dok se prenos ne završi

Rad DMA kontrolera

* CPU inicijalizuje DMA prenos (šalje početnu adresu, veličinu bloka, smer, port, tip završetka)
* CPU se vraća svojim zadacima, a DMA kontroler preuzima prenos
* Svaki put kada DMA kontroler pristupa memoriji, to mora sinhronizovati sa idle periodom procesora, na jedan od sledećih načina:
  + Trenutno onemogućavanjem CPU da pristupa magistrali
  + Slanjem halt signala i čekanjem potvrde
  + Usklađivanjem DMA pristupa sa određenim intervalom kloka ili statusnih signala procesora (kojim se signalizira idle ciklus)
  + Kada DMA kontroler pristupa U/I portu ili memoriji, on koristi iste signale kao CPU (pristup U/I portu može biti ostvaren preko posebnih linija/magistrale, i tada ne mora biti sinhronizovan sa CPU)
  + Po završetku prenosa, aktivira zahtev za prekidom ili postavlja odgovarajući status u svom status registru (zavisno koja je akcija selektovana da označi kraj prenosa)
  + Pocesor prepoznaje kraj U/I aktivnosti na osnovu prekida ili čitanjem status registra DMA kontrolera

Podela DMA

* Burst ili blok transfer DMA
* „Krađa ciklusa“ ili „prenos jednog bajta“ DMA
* Transparentni ili skriveni DMA – kad god CPU signalizira stanje u kome se koristi magistrala, DMA kontroler to iskoristi da prenese jedan podatak. Najsporiji metod, ali ne zaustavlja CPU

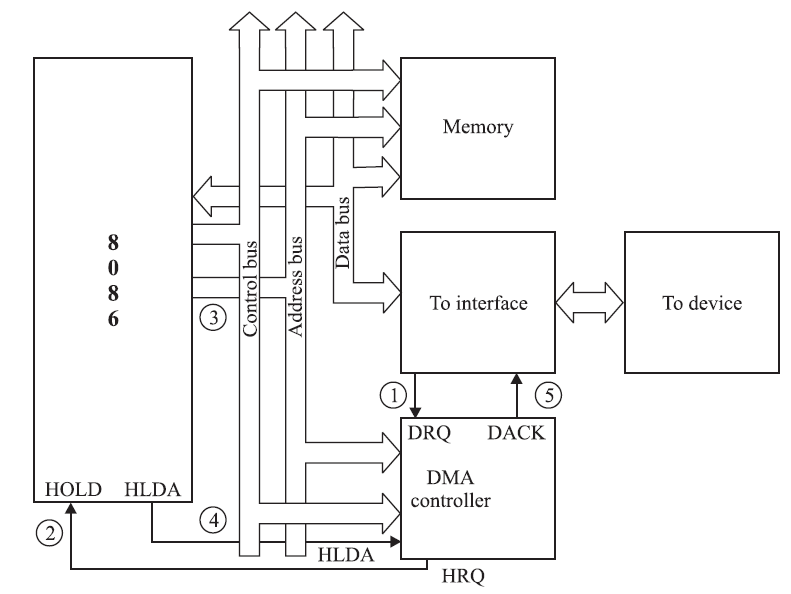
Burst ili blok transfer DMA

* DMA kontroler zadržava pravo da koristi magistralu sve dok ne prenese podatke koje su potrebni.
* To je najbrži DMA mod
* Podaci se prenose kontinuirano u jednoj HOLD operaciji
* CPU se povlači sa magistrale dok traje prenos (dakle, ne može da obavlja svoj posao)
* Za prenos N bajtova (ili reči) podataka N DMA ciklusa se umeće u mašinske cikluse CPU
* U realnom vremenu čekanje baš i nije zgodno, ako se desi prekid dok traje DMA, neće biti opsluženi

Krađa cujkzsa DMA

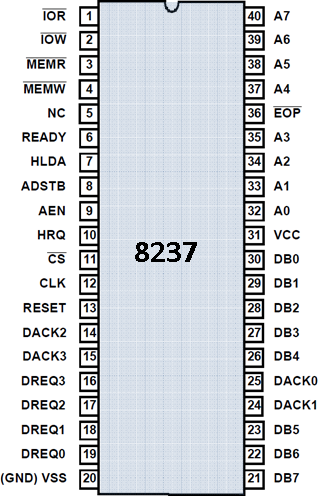
* Samo 1 bajt se prenosi u jednoj HOLD operaciji
* Nakon prenosa 1 bajta, DMA kontroler ukida HOLD signal i vraća CPU kontrolu nad magistralom (naizmenično koriste magistralu)
* Ovaj mod je sporiji od burst moda
* Ali CPU brže izvršava svoj program

Prenos podataka detaljno



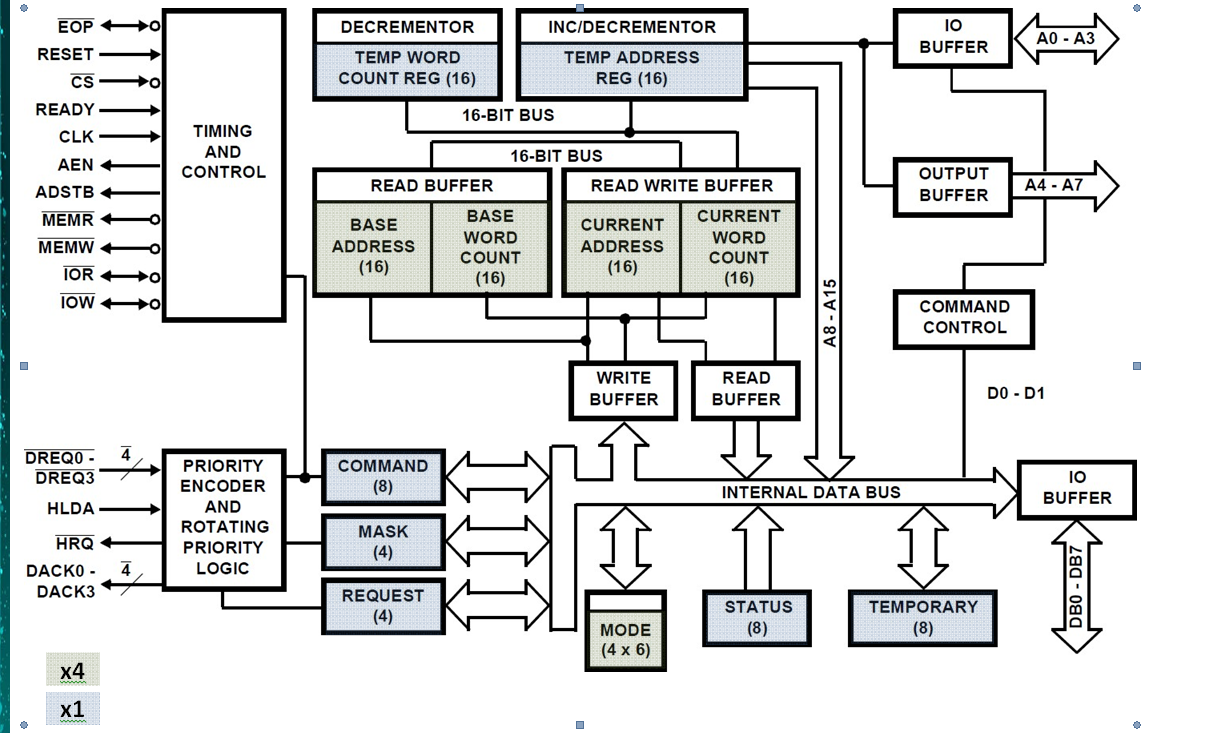
* Pretpostavka da je CPU inicijalizovao DMA kontroler. Desiće se zahtev od periferije za obavljanje transfera. Potrebu za prenosom signalizira slanjem DMA request signala (DRQ). Broj tih signala može biti veći, kontroler može imati više kanala. Kada stigne zahtev, recimo DRQ2, DMA proverava da li je dati kanal konfigurisan i da li u tom trenutku postoji signal kanala sa višim prioritetom da se izvrši prenos. Ako jeste konfigurisan i ne postoji zahtev drugog kanala višeg prioriteta, kreće se u obradu tog zahteva. Šalje se HOLD request ka CPU, da se oslobodi magistrala. CPU završava instrukciju i nakon toga oslobađa magistralu i šalje HLDA (Hold Acknowledge). Sada DMA preuzima magistralu i postaje arbitar, a uređaju šalje DACK gde mu kaže da može početi sa prenosom. Nakon toga adrese koje se budu generisale, generisaće DMA kontroler i definisaće memorijsku lokaciju u koju će se upisati podatak, kad god DMA postavi IO/R i MEM/WR čime omogućava upis podatka sa magistrale u memoriju.
* Kada U/I uređaj primi DACK, on postavlja podatak na magistralu. Ne može se direktno povezati na magistralu, već se povezuje sa komponentom recimo 8255 ili slično. Recimo čip koji će služiti kao port i moći ćemo da čitamo podatak sa tog porta
* Sa svakim prenosom podatka, adresni registar u DMA kontroleru se inkrementira ili dekrementira (smer može da se poešava), a brojač bajtova (reči) se dekrementra
* Kada brojač bajtova dostigne 0, javlja se EOP signal koji označava kraj prenosa
* Kontroler U/I uređaja deaktivira DRQ2 kada se završi prenos
* Kao odgovor na ukidanje DRQ2, DMA kontroler ukida HRQ (Hold request)
* Nakon ukidanja HRQ, CPU preuzima kontrolu nad magistralom i deaktivira HLDA
* Kao odgovor na to, DMA kontroler ukida DACK

8237



* 40 pinova
* Opslužuje spoljašnje uređaje koji zahtevaju DMA prenos. Pinovi koji se za ovo koriste su DREQ0-DREQ3 (DMA request). Na njih se vezuje periferija. 4 kanala, za svaki od kanala po 1 pin, inicijalno postoje prioriteti linija 0 najviši, linija 3 najniži, mogu da se rotiraju prioriteti ili da se programski menjaju
* Kad god se javi zahtev, DMA prosledi ka CPU i zato se koristi HRQ signal – izlazni signal, vezuje se ovaj kontroler na odgovarajući pin CPU
* CPU aktivira HLDA, to je ulazni pin za ovaj kontroler koji mu kaže da može da preuzme magistralu
* DB0-DB7-bidirekcione 3 state linije za pristup data magistrali. Koriste se za postavljanje statusa ove komponente i za čitanje statusa. Imamo registre koji definišu rad same komponente.
* Pinovi DB0 do DB7 se koriste za prenos višeg bajta adrese. Adresa će na pinovima od A0 do A7 biti aktivna sve vreme dok traje transfer. Pošto se data deo može koristiti u samom procesu prenosa, onda je neophodno da se podatak sa data linija lečuje, zato koristimo osmobitni leč koji se omogućuje signalom ADSTB. Ovaj signal omogučuje da se viši bajt adrese zapamti. Javlja se samo u prvom ciklusu
* Ukoliko se vrši prenos iz memorije u memoriju onda se na data magistrali nalaze i podaci koji se čitaju/upisuju. U normalnoj komunikaciji U/I uređaja i memorije sam kontroler ne učestvuje u razmeni podataka (nema ništa validno na data linijama), međutim kada se prenosi iz memorije u memoriju potrebno je da se podatak privremeno smesti u kontroler i u tom slučaju se na data pinovima nalaze podaci koji se upisuju u interni pomoćni registar, da bi se u sledećem podciklusu podatak preneo na drugu mem lokaciju
* A0 – A3 niža 4 bita adresne linije. Kada se postavlja adresa lokacije za upic/čitanje, to je deo adrese. Ulazno izlazne linije
* A4-A7 viši polubajt adrese, izlazne linije.
* AEN – (Address Enable) omogućuje izlaz iz 8‐ bitni leča koji sadrži i aktivan je sve vreme dok su postavljene adrese na adresnoj magistrali. Aktivne dok traje DMA prenos da bi se selektovala memorijska lokacija za upis podatka. Signal se uparuje sa lečom u koji se upisuje viši deo adrese
* ADSTB – (Address Strobe) aktivira spoljašnji leč. Aktivan samo u prvom podciklusu, služi da se podaci sa data linija upišu u adresni leč
* READY – ulazni signal koji se koristi da produži impulse za pristup memoriji (čitanje i upis) kod sporih memorija
* IOR – (I/O Read) kao ulazni signal omogućuje CPU da pročita stanje kontrolnih registara (Idle ciklus), kao izlazni omogućuje 8237A da čita sa periferije tokom DMA Write transfera (Active ciklus)
* IOW – (I/O Write) kao ulazni signal omogućuje CPU da upiše informacije u kontrolne registre (Idle ciklus), kao izlazni omogućuje 8237A da upiše podatke u periferije tokom DMA Read transfera (Active ciklus)
* U idle ciklusu ova 2 pina IOR i IOW se koriste kao ulazni i omogućavaju čitanje ili upis kontrolnih registara
* DACK0‐DACK3 – (DMA Acknowledge) obaveštava periferije da je odobren DMA ciklus
* MEMR – (Memory Read) koristi se za pristup memoriji radi čitanja
* MEMW – (Memory Write) koristi se za pristup memoriji radi upisa
* EOP – (End of Process) eksterni signal koji prekida aktivni DMA ili izlazni signal kojim se označava kraj DMA (kada se dostigne terminal count (TC) za bilo koji kanal, osim za kanal 0 u prenosu iz memorije u memoriju
* NC – (No Connect)

Blok dijagram



* Current address – adresa mem lokacije sa koje se čita/upisuje podatak tokom DMA prenosa, da li se čita ili upisuje zavisi od smera. 16 bit registar. Njegova vrednost se menja nakon svakog transfera. Kada se inicijalizuje rad komponente, početna adresa se upisuje u base address, a početna vrednost veličine bloka se upisuje u base word count. Istovremeno se ta vrednost upisuje i u current address i current word count, kako se bude odvijao transfer tako će se ova adresa povećavati ili smanjivati, zavisi kako je definisan prenos od poč ili kraj adrese, a broj će se smanjivati i kada dostigne 0 završiće se transfer.
* Zbog čega postoji potreba i za baznom adresom i za tekućom adresom?
  + To je posledica ubrzavanja prenosa. Ukoliko je definisana auto inicijalizacija kao mod rada tog kanala, po završetku orenosa će se izvršiti reinicijalizacija tog prenosa. Vrednosti koje su upisane u baznu adresu i inicijalni broj reči koju treba preneti će se automatski preneti u current word count bez posredovanja CPU- a. Možemo da restarujemo komunikaciju.
* Ako nije uključena auto inicijalizacija, po završetku transfera, broj reč se premotava na FFFFh i ostaje u tom stanju, ako je uključena, po završetku se reinicijalizuje na početne vrednosti i možemo ponovo startovati za prenos podataka, bez da se CPU uopšte uključuje.
* Pomoćni registri TEMP WORD COUNT i TEMP ADDRESS REG, koji direktno čitaju adrese (radni registri) povezani na bafere, postoji po 1 registar, tu se upisuje ono što se trenutno koristi.
* Command registar služi za definisanje rada čitave komponente. To je jedan osmobitni registar.
* Mode registar služi za sve vezano za pojedinačne kanale (konfigurisano je u Mode registru). Moći ćemo sva 4 kanala odjednom da konfigurišemo ali zapravo za svaki od kanala postoji poseban mode registar. Postoji Naredba kojom se upravlja modom rada odgovarajućeg kanala, a mi ćemo u okviru te naredbe da selektujemo koji kanal konfigurišemo
* Za svaki kanal postoji Mask i Request registar
* Mask definiše koji su kanali maskirani odnosno sa kojih kanala ne želimo da primamo zahteve. Svugde de je 1 je maskirani kanal
* Request ima po 1 bit po kanalu u kome se signalizira da postoji zahtev na odg kanalu. Može i softverski da se proveri. Može i softverski da se postavi. Kod mem to mem transfera ne postoji periferija koja inicijalizuje prenos nego CPU, jedini način da se signalizira DMA kontroleru je da se setuje odgovarajući bit u ovom registru i emulira zahtev koji bi došao sa spolj uređaja
* Status registar (read only registar), za svaki kanal ima po 2 bita. Oni definišu da li postoji zahtev i da li je odgovarajući zahtev opslužen (da li je dostignut terminal count).
* Temporary registar se koristi kod mem to mem transakcija 8-bit, privremeno smešta 1 bajt koji se prenosi. Kod U/i to mem ili obrnuto, komunikacija se obavlja preko magistrale, tu nema konflikta i DMA kontroler se ne uklučuje u sam prenos. Na data pinovima se ništa ne postavlja. Kada se vrši prenos mem to mem za taj prenos se koristi ista magistrala tako da bismo u jednom delu morali da selektujemo mem modul iz koga čitamo, a u sledećem modul gde će da se upiše podatak. U međuvremenu podatak mora da bude negde smešten. Zato se koristi temporary registar

Stanja i ciklusi

* DMAC radi u dva ciklusa: Idle i Active
* U Idle ostaje dok se ne javi zahtev na DREQ. DMA je neaktivan i čeka da se javi zahtev.
* Tada DMAC zahteva magistralu i prelazi u Active ciklus koji se sastoji od nekoliko unutrašnjih stanja, zavisno od selektovane operacije (burst DMA, single byte DMA,….) i zahtevanog prenosa (mem‐IO, IO‐mem, mem‐ mem)
* 8237A ima 7 stanja, svako u trajanju od jednog taktnog intervala
* Stanje I (SI) je Idle stanje. U njega ulazi kada nema DREQ zahteva, nakon DMA ili nakon Reset ili Master Clear. Početno stanje.
* U SI stanju očitava DREQ linije na opadajuću ivicu takta (ako nema takta, zahtevi se ignorišu)
* Kada je CS aktivan i HLDA neaktivan, 8237A ulazi u Program Condition, kada CPU može da postavi ili pročita stanja registara

Modovi rada

* Single transfer mod
* Block transfer mod
* Demand transfer mod
* Cascade mod

Single transfer mod

* U ovom režimu, komponenta se programira da prenese samo jedan bajt
* Nakon prenosa 1B, ukida se HRQ, bez obzira da li je Current Word Count dostigao 0 (tj. FFFFH)
* Ovo je zapravo režim „krađe ciklusa“
* Ako je DREQ osta aktivan nakon jednog DMA ciklusa, ponovo se postavlja HRQ i zahteva se sledeći ciklus

Block transfer mod

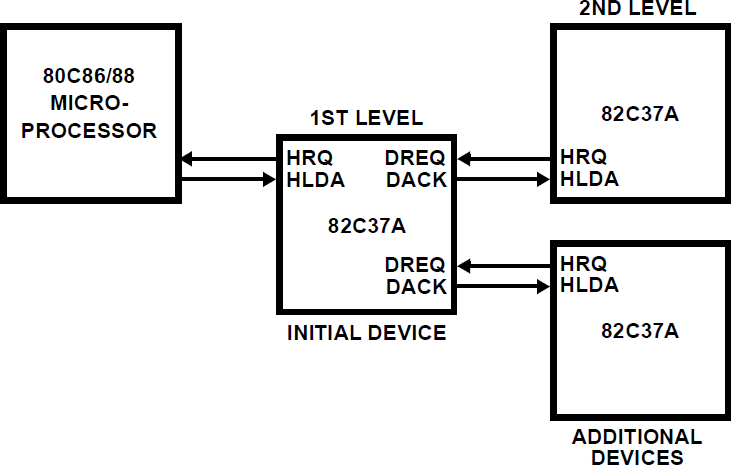
* U ovom režimu, bajtovi se prenose kontinuirano
* HRQ ostaje aktivno tokom celog prenosa (svih DMA ciklusa)
* Prenos se završi teka kada se javi TC ili spoljašnji EOP

Demand transfer mod

* U ovom modu, broj prenetih bajtova zavisi od U/I uređaja
* DMA se prekida u jednom od 3 slučaja:
  + dostignut TC,
  + spoljašnji EOP ili
  + DREQ postaje neaktivan
* Koristi se kada treba omogućiti DMA dok U/I uređaj ne iscrpi podatke za prenos
* Po prestanku DREQ, zadržavaju se vrednosti adrese i brojača, osim ako nije postavljen autoinicijalizacioni mod, kod koga EOP resetuje registre na startne vrednosti

Kaskadni mod

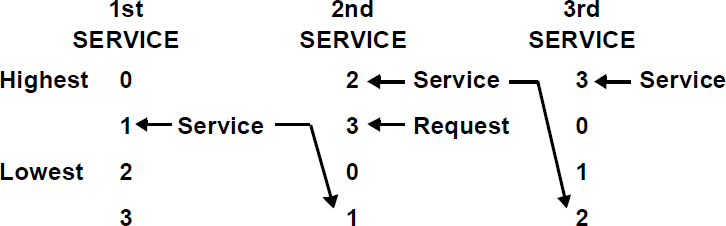
* Koristi se kod kaskadnog povezivanja više 8237 komponenti, kako bi se povećao broj DMA kanala
* Prioriteti se propagiraju kroz kaskadnu vezu
* Komponenta u 1. nivou ne postavlja adrese niti kontrolne signale (MEMW, MEMR, IOR, IOW), već to radi komponenta narednog nivoa
* Komponenta na 1. nivou postavlja samo HREQ i DACK (izlazne) signale (READY ignoriše)
* Nema ograničenja u broju nivoa



* Kada stigne zahtev sa neke periferije, komponenta drugog nivoa će proslediti HRQ (Hold request) kao DMA request komponenti prvog nivoa, a ona će proslediti CPU-u kroz svoj HRQ. Kad dobije HLDA od CPU-a on prosleđuje komponenti drugog nivoa.

Prioriteti zahteva

* 8237A ima dva tipa određivanja prioriteta:
  + fiksni i
  + rotirajući
* Kod fiksnog prioriteta 0 je kanal sa najvišim, a 3 sa najnižim prioritetom
* Kod rotirajućeg prioriteta, poslednji kanal koji je uslužen dobija najniži prioritet, a sledeći po redu najviši (vidi narednu sliku); rotacija se dešava nakon što se upravljanje magistralom vrati CPU



* Ovo levo je inicijalno stanje. 0 najviši, 3 najniži prioritet. Ako se javio zahtev po kanalu 1, nakon njegove obrade, kanal 1 će biti najnižeg prioriteta, a kanal 2 najvišeg.
* Ako se javi na kanalu 0 ili 1 zahtev, kanal 0 će biti opslužen,a kanal 1 nakon toga postaje najviši prioritet.

Tipovi prenosa

* Postoje 3 tipa prenosa podataka od/ka periferijama:
* DMA Read – prenosi od memorije ka U/I uređaju (aktivni MEMR i IOW)
* DMA Write – prenosi od U/I uređaja ka memoriji (aktivni MEMW i IOR)
* DMA Verify – pseudo prenos (8237 radi kao da zaista postoji prenos, ali MEMX i IOX signali nisu aktivni, tako da se zapravo ništa ne prenosi; nije dozvoljen za mem‐mem prenos; READY se ignoriše)

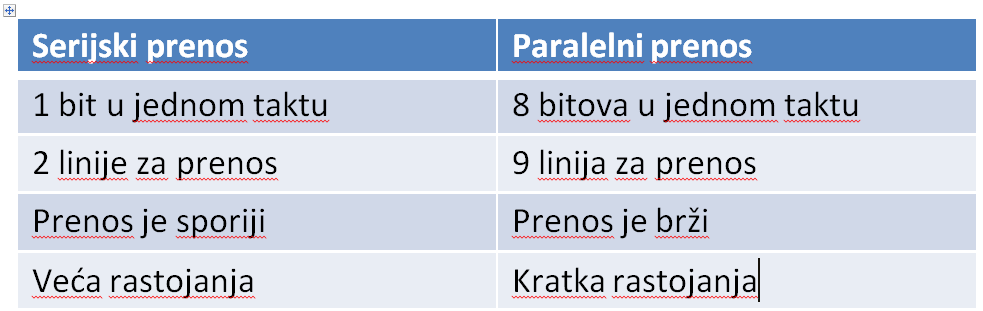
Memory to memory prenos

* Koriste se dva kanala: 0 i 1
* Kanal 0 definiše adrese sa kojih se čita (DMA Read), a kanal 1 u koje se upisuje (DMA Write)
* Radi jedino u blok modu
* Signali DACK, IOR i IOW nisu aktivni (prepoznaje se po aktivnom AEN i neaktivnim DACK linijama)
* Inicira se postavljanjem softverskog DREQ za kanal 0
* Čitanje se obavlja u 4 ciklusa (S11, S12, S13 i S14) i vrednost smešta u privremeni registar u okviru 8237A, a zatim u 4 ciklusa prenosi u odredište (S21, S22, S23 i S24)
* Kada brojač reči u kanalu 1 pređe sa 0 na FFFFH, generiše se TC i postavlja EOP
* Kanal 0 može biti programiran da ne menja adresu, što omogućuje da se čitav blok ispuni jednim podatkom

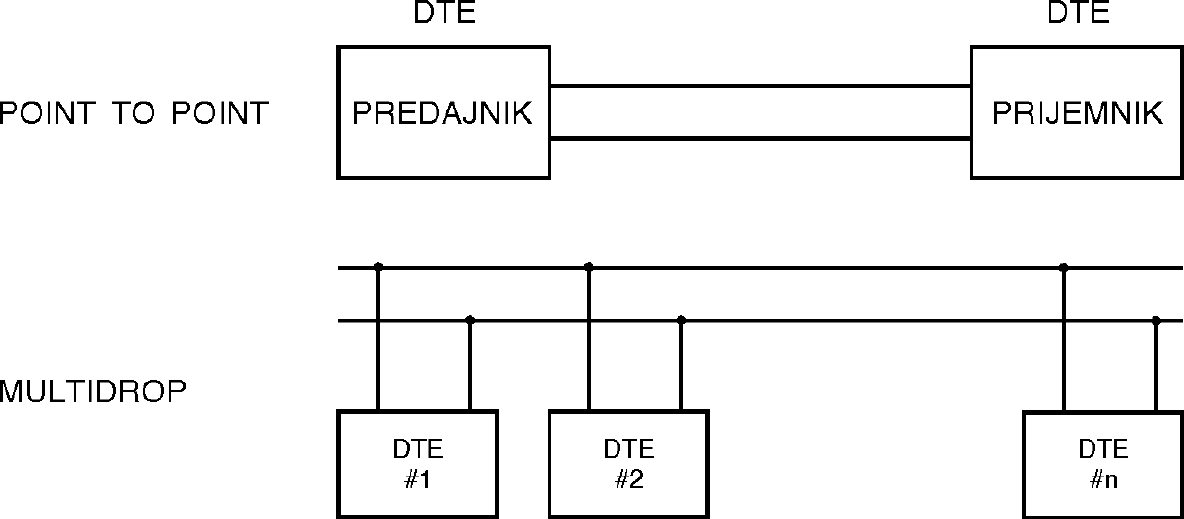
Softverske komande

* Postoje dodatne softverske komande, koje ne zavise od stanja na adresnoj magistrali:
* Clear first/last flip‐flop – mora biti izdata pre upisa ili čitanja adrese i brojača jer se svakim upisom ili čitanjem menja stanje tog flip‐flopa, kako bi se pristupilo narednoj lokaciji (nakon reseta prvo se očitava/upisuje niži bajt)
* Master clear – ima isti efekat kao hardverski RESET
* Clear Mask register – briše Mask registar i omogućuje sva četiri kanala

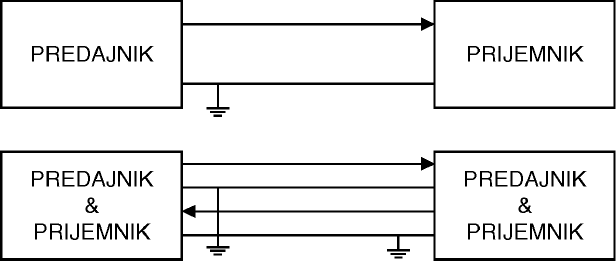
Serijski U/I



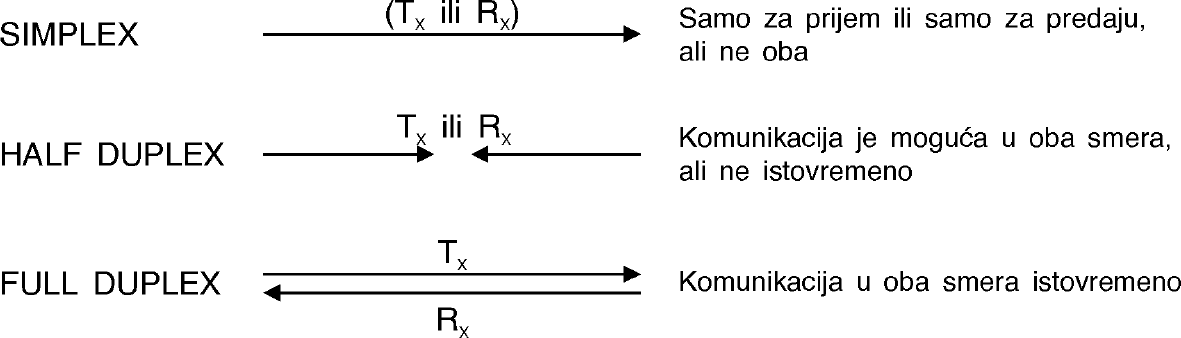
* slabljenje signala ‐ energija signala duž propagacionog puta se smanjuje
* distorziono kašnjenje signala ‐ različite Fourier‐ove komponentesignala se prostiru različitim brzinama (što dovodi do izobličenja signala)
* šum ‐ nepoželjna energija koja potiče od drugih izvora
  + termički šum je uzrokovan slučajnim kretanjem elektrona kroz provodnike,
  + preslušavanje je uzrokovano induktivnom spregom izmešu dva provodnika,
  + Impulsni šum posledica je uticaja indukovanih smetnji energetskog razvoda (uključenje/isključenje raznih motora ili velikih potrošača).

Načini serijskog povezivanja

Fizička veza

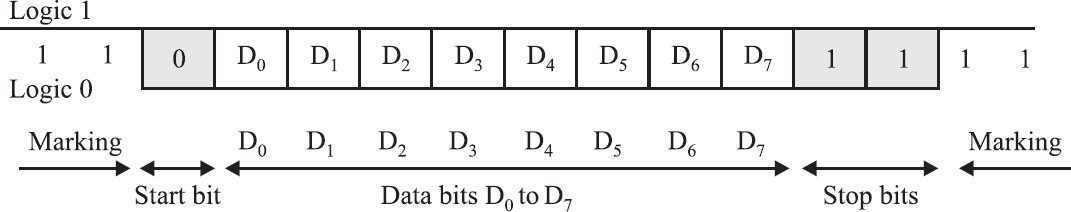


Logičke veze

Tipovi modulacije

* Amplitudna
* Frekventna
* Fazna

Asinhroni prenos

* Svaki bajt se „omotava“ da bi omogućio sinhronizaciju prijemnika i predajnika
* Dodaju se start i stop bitovi
* Linija je inicijalno na „visokom“ nivou
* Kada treba da se prenese 1B, u trajanju od 1 klok‐ intervala, prelazi na nisku vrednost – start bit
* Nakon prenosa poslednjeg bita podatka (D7) linija prelazi na visok nivo za određeni period (1, 1.5 ili 2 klok‐intervala) – stop bitovi
* Pogodan je za manje brzine prenosa, do 32kb/s, na svaki bajt troši još bar 2 dodatna bita.
* Nema kontrolu grešaka.

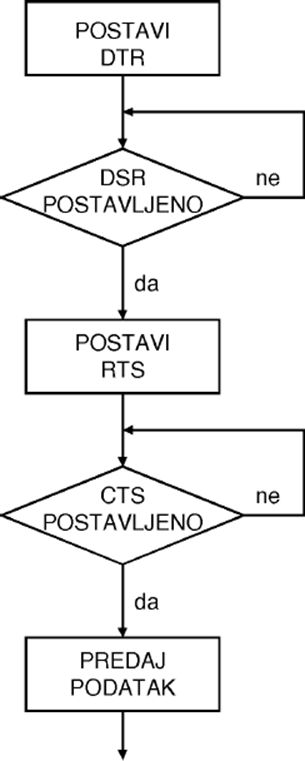
Sinhroni prenos

* Zahteva prenos takt‐signala i master/slave organizaciju (master je onaj koji šalje i obezbeđuje takt)
* Prenos takta obezbeđuje se posebnom linijom ili stalnim prenosom podataka
* Povećava efikasnost grupisanjem podataka
* Dodaje se okvir (početak, zaglavlje, kontrolna suma, kraj okvira)
* Može biti
  + Karakter‐orijentisan
  + Bit‐orijentisan

Serijski protokoli

* RS‐232
  + Kratko rastojanje (<15m) i mala brzina prenosa (20kb/s)
* RS-423
  + Veće rastojanje (<1200m) i veće brzine prenosa (100kb/s na 15m, 1kb/s na 1200m)
  + Adaptabilna brzina promene signala (može se postaviti zavisno od dužine kabla i brzine prenosa i time smanjiti uticaj šuma; izražava se u V/s; za RS‐232 max. 30 V/s)
* RS‐422
  + Veće rastojanje (<1200m) i veća brzina prenosa (10Mb/s na 15m , 100kb/s na 1200m)
* RS‐485
  + Sličan RS‐422, ali dozvoljava više i primalaca i pošiljaoca (više mastera na magistrali)

RS-232

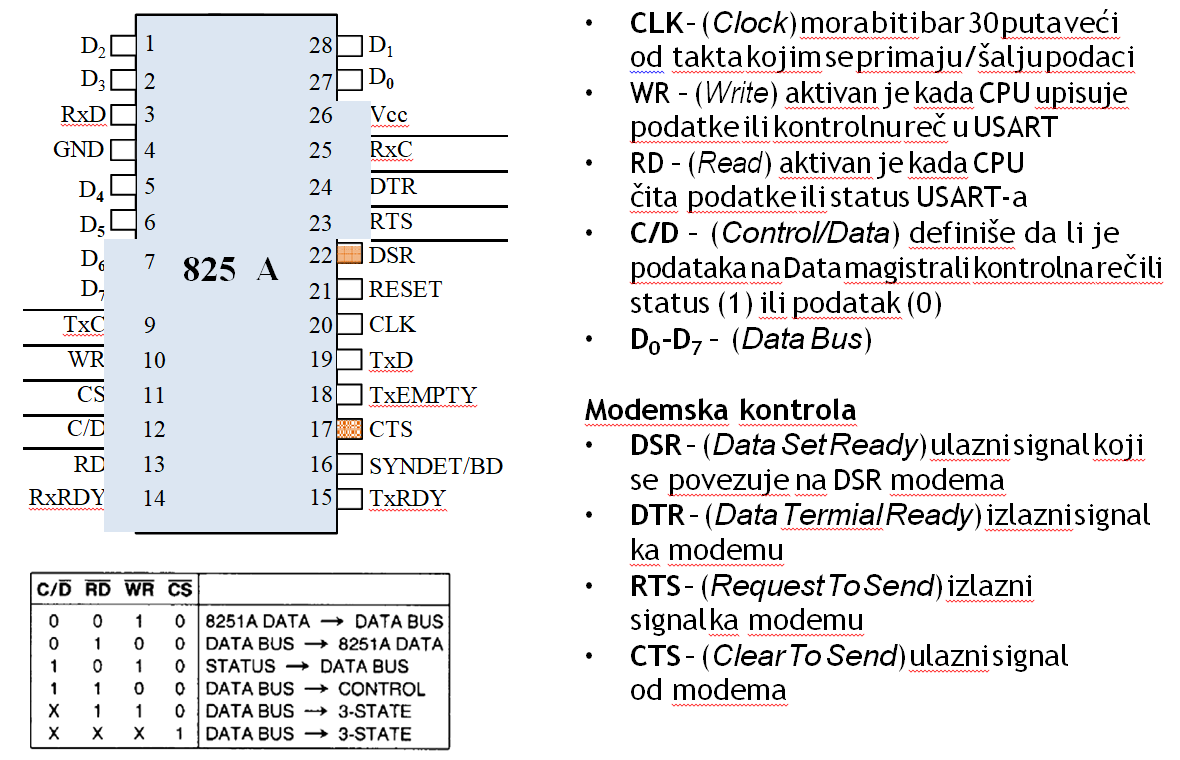
* DTR (Data Terminal Ready) se koristi za upravljanje priključenjem DCE na komunikacioni kanal.
* DSR (Data Set Ready) se koristi da prikaže stanje lokalnog DCE (modema). Ako je mark znači da je spreman za korišćenje.
* DSR i DTR se često vezuju na izvor napajanja u DCE i DTE respektivno, tako da pokazuju da je svaki od uređaja uključen i spreman za rad.
* RTS (Request To Send), šalje DTE (računar ili terminal) ka DCE (modemu) da bi pripremio DCE za prenos. Ovaj signal se takođe koristi da odredi smer prenosa u half duplex režimu. Prelazi na aktivni nivo kad god DTE ima spreman karakter za slanje.
* CTS (Clear To Send) generiše DCE da bi obavestio DTE da prenos može da počne kad god DTE bude spreman. Kod full duplex linija RTS i CTS se prosto vezuju zajedno.
* Ring (CE) ukazuje da je prisutan zvuk zvona na telefonskoj liniji. Indicira uspostavljanje veze onom ko je zvao.
* DCD (Data Carrier Detect) znači da je DCE primio signal koji odgovara modemskim specifikacijama. Ako ga nema, znači da signal nije primljen ili da je nepogodan za demodulaciju od strane DCE.
* 

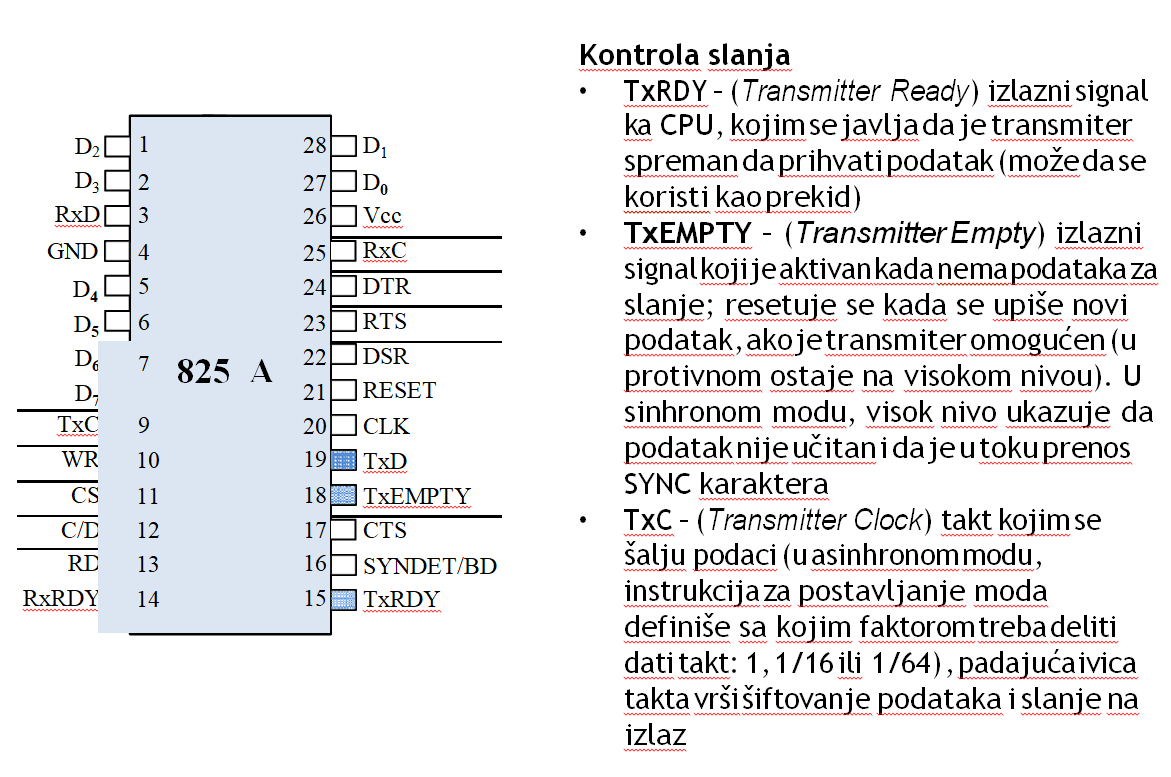
Serijski U/I korišćenjem paralelnog porta

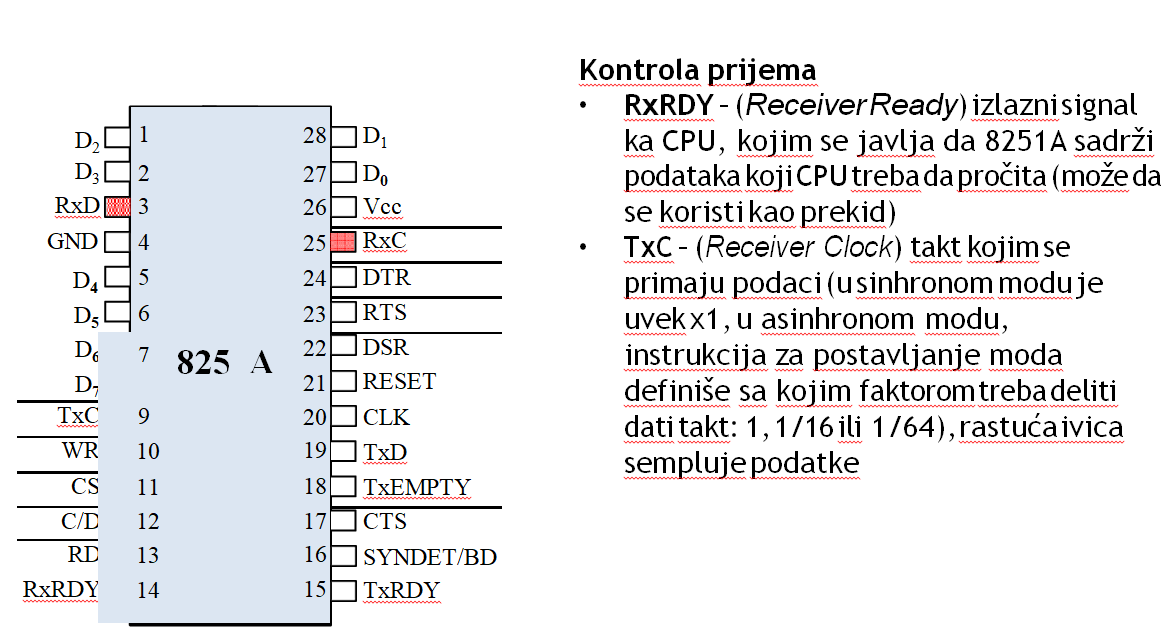
* Pretpostavimo da je potrebno implementirati softverski zasnovan serijski U/I, korišćenjem 8086 procesora
* Na raspolaganju je 8255 programabilni periferijski interfejs
* Postaviti port A (PA) u mod 0 i ulazni smer, a port B (PB) u mod 0 i izlazni smer
* Neka se bit PA0 koristi kao serijski ulaz, a PB0 kao serijski izlaz
* Smestiti podatke koje treba preneti u memoriju, počev od određene lokacije
* Smestiti dužinu u odgovarajući brojač
* Inicijalizovati 8255 tako da radi u odgovarajućem režimu
* Dok nisu preneti svi bajtovi
  + Učitati jedan bajt
  + Poslati 0 na PA (start bit)
  + Dok nisu poslati svi bitovi (8)
    - Poslati na PA očitani bajt (samo PA0 se zaista šalje)
    - Šiftovati sadržaj pročitanog podatka
  + Poslati 1 na PA (stop bit)
  + Dekrementirati brojač i inkrementirati adresu sa koje se čitaju podaci

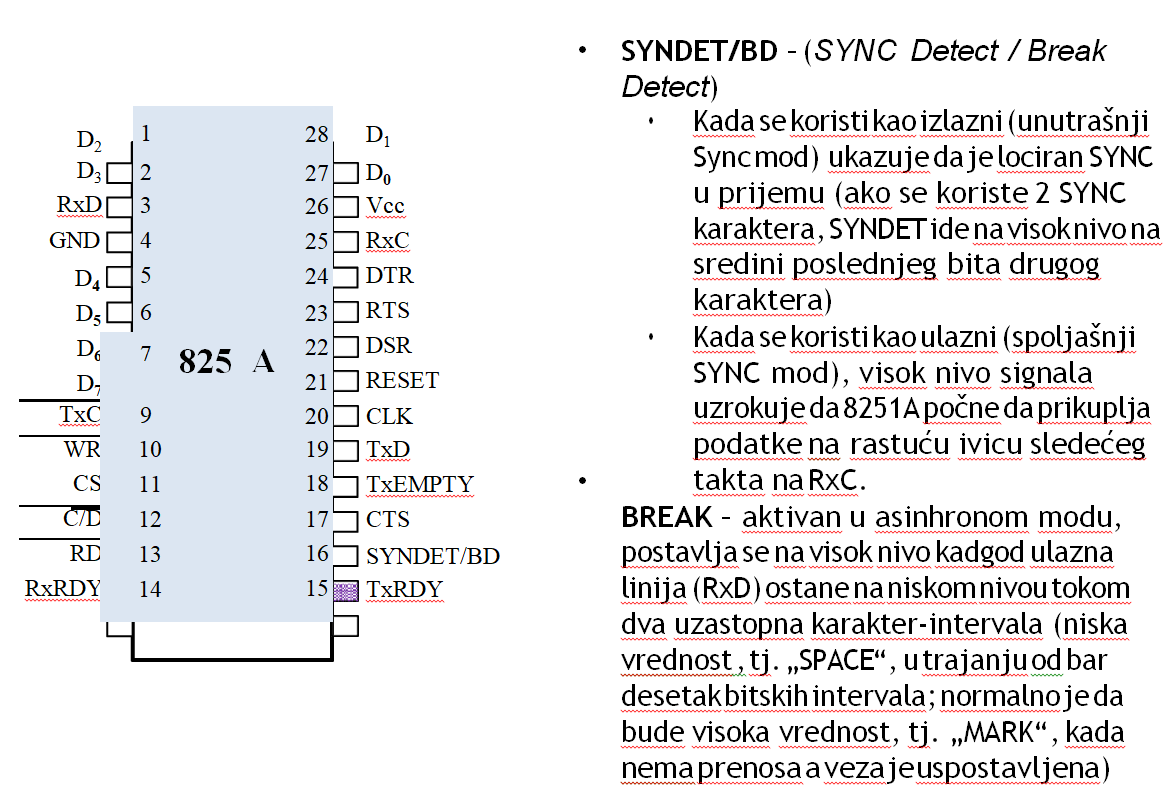
USART 8251

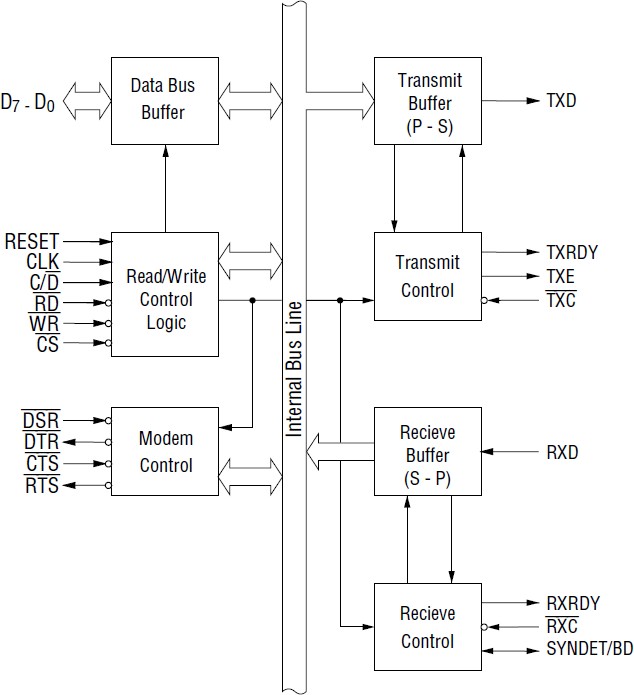
* Ako se koristi 8251, CPU treba samo da inicijalizuje komponentu i pošalje odgovarajući podatak
* 8251 prihvata „paralelni“ podatak od CPU, konvertuje ga i šalje serijski
* Takođe, prihvata serijske podatke na svom ulazu i konvertuje ih u „paralelne“ i prosleđuje CPU





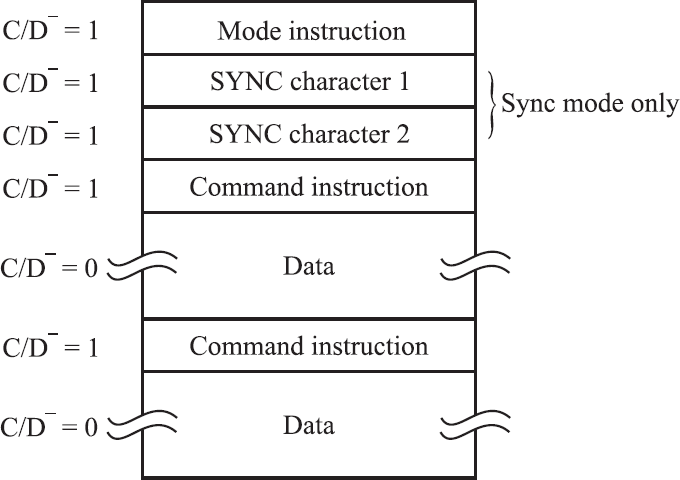




Funkcionalni blok dijagram

* Transmit Buffer – prihvata paralelno podatak iz Data Bus Buffer‐a, pretvara ga u serijski niz bitova, umećući odgovarajuće karaktere ili bitove (zavisno od tehnike) i šalje ih na TxD izlazni pin, na padajuću ivicu TxC signala.
* Prenos počinje nakon što je omogućen, ukoliko je CTS = 0.
* Receive Buffer – prihvata serijske podatke sa RxD, pretvara ih u paralelni oblik, pronalazi umetnute bitove ili karaktere i sklapa karakter koji se šalje ka CPU; taktuje se rastućom ivicom RxC

Tipični prenos podataka



1. Resetovati komponentu (hardverski ili slanjem odgovarajuće komande)
2. Postaviti mod rada (posle reseta prva komanda, C/D=1, se tumači kao postavljanje moda rada)
3. Ako je sinhroni mod rad, sledeća komanda definiše SYNC1 karakter (C/D=1)
4. Ako je sinhroni mod rad i selektovana su dva SYNC karaktera u postavkama moda rada, sledeća komanda definiše SYNC2 karakter (C/D=1)
5. Izdati komandu (uključiti prijem i/ili predaju, ući u HUNT mod, resetovati greške, …) (C/D=1)
6. Zadavati karaktere koji se prenose pod dejstvom prethodne komande (C/D=0)
7. Izdati novu komandu…