JOVAN ĐORĐEVIĆ

ARHITEKTURA I ORGANIZACIJA RAČUNARA

ULAZ/IZLAZ

BEOGRAD, 2014.

DJM

PREDGOVOR

Ova knjiga je napisana kao osnovni udžbenik iz arhitekture i organizacije računara i pokriva osnovne koncepte iz arhitekture i organizacije procesora, memorije, ulaza/izlaza i magistrale.

Autor

Beograd

aprila 2014.

SADRŽAJ

1 ULAZ/IZLAZ

U ovoj glavi se razmatraju neki elementi organizacije ulaza/izlaza. U okviru toga se, najpre, definišu osnovni pojmovi. Potom se prikazuje organizacija ulaza/izlaza korišćenjem kontrolera priferija bez i sa direktnim pristupom memoriji i ulazno/izlaznih procesora. Na kraju se razmatra opsluživanje maskirajućih prekida.

1.1 OSNOVNI POJMOVI

Informacije sa kojima radi računar se ili nalaze na medijumima kao što su diskovi, magnetne i papirne trake, papirne kartice itd. ili se unose sa terminala ili dolaze sa nekih drugih računarskih sistema. Svi oni se u daljim razmatranjima nazivaju ulazni uređaji. S druge strane računar je tako koncipiran da se i instrukcije i operandi uzimaju iz memorijskih lokacija. Zato postoji potreba da se oni sa ulaznih uređaja unose u memorijske lokacije. Rezultati izvršenih instrukcija se, takođe, smeštaju u memorijske lokacije. Da bi se prezentirali korisniku i/ili sačuvali za kasnije korišćenje i/ili predali u neki drugi računarski sistem, oni moraju da se šalju iz memorijskih lokacija na diskove, magnetne i papirne trake, printere itd., terminale ili u druge računarske sisteme. Svi oni se u daljim razmatranjima nazivaju izlazni uređaji.

Postoji veliki broj različitih ulaznih i izlaznih uređaja. Oni se razlikuju po tome kako se informacije u njima smeštaju, po načini kako sa njih dolaze i u njih šalju informacije i po brzini sa kojom se to radi. Međutim, rad sa ulazno/izlaznim uređajima je tako organizovan da se on realizuje jednoobrazno, bez obzira na to o kojoj se vrsti ulaznog ili izlaznog uređaja radi. Da bi se to omogućilo svi uređaji se tako realizuju da se sastoje od kontrolera periferije i periferije.

Kontroleri periferija sadrže određen broj programski dostupnih registara i upravljačku logiku. Programski dostupni registri služe da se kompletna organizacija ulaza/izlaza na programskom nivou svede na upisivanje u ove registre i čitanje ovih registara. Upravljačka logika služi da se na osnovu sadržaja programski dostupnih registara organizuje čitanje podataka iz ulazne periferije i upis podataka u izlaznu periferiju. Programski dostupni registri se, prema svojoj funkciji, mogu svrstati u četiri grupe i to: upravljački registar, statusni registar, registar podatka i registar broja ulaza. Upravljački registar služi da se programskim putem upisivanjem odgovarajućih vrednosti u ovaj registar izvrši inicijalizacija, startovanje i zaustavljanje kontrolera periferije. Upravljačka logika na osnovu sadržaja određenih bitova upravljačkog registra kreće sa prenosom podataka iz ulazne periferije u registar podatka ili iz registra podatka u izlaznu periferiju. Za svaki podatak prenet iz ulazne periferije u registar podatka upravljačka logika postavlja određeni bit u statusnom registru kao indikaciju da sadržaj registra podatka može da se prenese u memorijsku lokaciju. Takođe i za svaki podatak prenet iz registra podatka u izlaznu periferiju upravljačka logika postavlja određeni bit u statusnom registru kao indikaciju da sadržaj sledeće memorijske lokacije može da se prenese u registar podatka. Samo prebacivanje podatka iz registra podatka u memorijsku lokaciju ili iz memorijske lokacije u registar podatka se kod nekih kontrolera periferija realizuje programskim putem čitanjem sadržaja registra podatka ili upisivanjem u registar podatka, dok kod nekih kontrolera periferija to radi sam kontroler. Kod onih kontrolera perfierija kod kojih se to radi programskim putem, mora se čitanjem statusnog registra, najpre, utvrditi da je sledeći podatak prenet iz ulazne periferije u registar podatka, pa ga tek onda prenositi iz

registra podatka u memorijsku lokaciju ili da je prethodni podatak prenet iz registra podatka u izlaznu periferiju, pa tek onda prenositi sledeći podatak iz memorijske lokacije u registar podatka. Kod ovih kontrolera periferija postoji i mogućnost da za svaki podatak prenet iz ulazne periferije u registar podatka upravljačka logika generiše signal prekida i da za svaki podatak prenet iz registra podatka u izlaznu periferiju upravljačka logika takođe generiše signal prekida. Na signal prekida procesor odgovara signalom potvrde. Na ovaj signal kontroler periferije šalje procesoru sadržaj registra broja ulaza. Procesor na osnovu broja ulaza ulazi u tabelu sa adresama prekidnih rutina, čita adresu prekidne rutine i njenim upisivanjem u programski brojač skače na prekidnu rutinu. Sada se u prekidnoj rutini podatak prenosi iz registra podatka u memorijsku lokaciju ili iz memorijske lokacije u registar podatka. Kod onih kontrolera perfierija kod kojih to radi sam kontroler, upravljačke logika kontrolera realizuje ciklus upisa sadržaja iz registra podatka u memorijsku lokaciju ili ciklus čitanja iz memorijske lokacije i prihvatanje očitanog sadržaja u registru podatka. U oba slučaja upravljačka logika kontrolera periferije realizuje određene kontrole rada sa periferijom i u slučaju otkrivanja nekih neregularnosti postavlja određene bitove statusnog registra. Zbog toga se programskim putem u određenim trenucima čita sadržaj statusnog registra i na osnovu provere njegovog sadržaja utvrđuje da li se rad sa periferijom odvija regularno ili ne. Po završenom prenosu ili otkrivenoj neregularnost u odgovarajuće bitove upravljačkog registra se programskim putem upisuje sadržaj na osnovu koga upravljačka logika kontrolera zaustavlja prenos podataka iz ulazne periferije u registar podatka ili iz registra podatka u izlaznu periferiju.

Periferije sa danas tako realizuju da se detalji vezani za samo čitanje iz ulazne periferije ili upis u izlaznu periferiju ne vide, već se pomoću definisanog interfejsa i protokola upravlja čitanjem iz i upisom u periferiju, primaju ili šalju podaci i dobijaju informacije o ispravnosti rada sa periferijom.

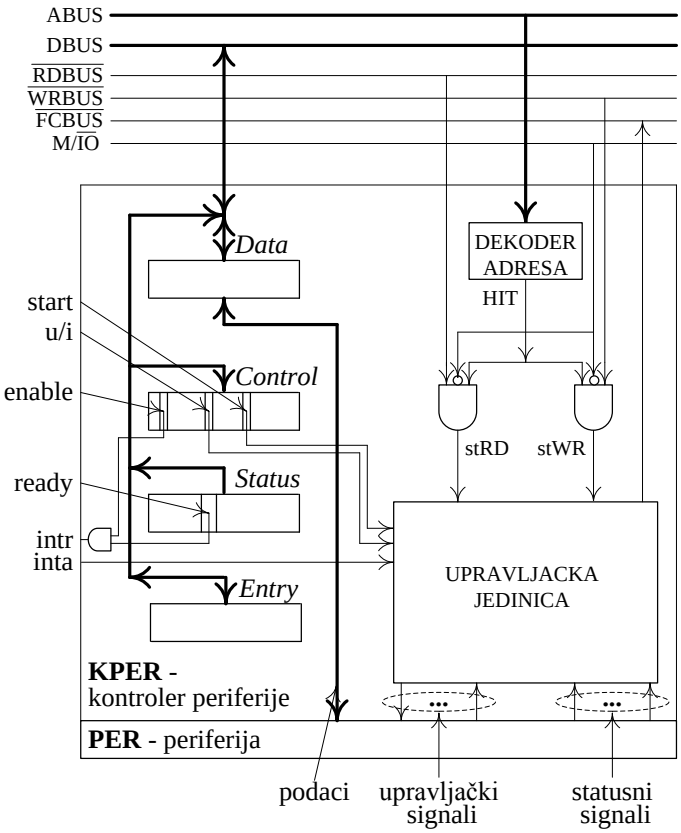
Podaci se obično unose sa periferije blokovski, tako što se u neki deo memorije u niz susednih lokacije prenosi određen broj reči. Isto važi i za slanje podataka iz memorije u izlazne periferije. Za realizaciju blokovskog unosa podataka iz periferije u memoriju i slanja podataka iz memorije u periferiju treba uraditi sledeće: prenositi podatke iz periferije u memoriju ili obratno, voditi evidenciju o adresama memorijskih lokacija u koje se reči upisuju ili iz kojih se reči čitaju i voditi evidenciju o broju prenetih reči. Moguće je da procesor sve ovo radi, da deo poslova radi procesor a deo posebni uređaji i da sve ovo rade posebni uređaji. U zavisnosti od toga kava je podela posla između procesora i posebnih uređaja, koristi se više tehnika organizacije ulaza/izlaza i to ulaz/izlaz sa kontrolerima bez direktnog pristupa memoriji, sa kontrolerima sa direktnim pristupom memoriji i sa ulazno/izlaznim procesorima.

1.2 KONTROLER PERIFERIJE

U ovom odeljku se razmatraju organizacija KPER kontrolera (kontrolera periferije), programiranje KPER kontrolera i povezivanje sa periferijom.

1.2.1 ORGANIZACIJA

U ovom odeljku se razmatra organizacija KPER kontrolera (slika 1). Kontroler se sastoji od operacione jedinice i upravljačke jedinice. Operacionu jedinicu čine registri *Data*, *Control*, *Status* i *Entry* i kombinaciona mreža DEKODER ADRESA za prepoznavanje ciklusa čitanja i upisa koji treba da se realizuju sa ovim registrima. Upravljačka jedinica realizuje čitanje iz i upis u registre *Data*, *Control*, *Status* i *Entry* i prenos podataka iz periferije u registar *Data* i obratno.



Slika 1 Kontroler periferije

Osnovni zadatak kontrolera je da u slučaju ulaza prihvata podatke koji dolaze iz periferije i smešta u registar podatka *Data* i u slučaju izlaza šalje podatke iz registra *Data* u periferiju. Prenos podatka iz registra *Data* u memorijsku lokaciju u slučaju ulaza i iz memorijske lokacije u registar *Data* u slučaju izlaza, realizuje se programskim putem izvršavanjem odgovarajućeg programa.

Kontroler i procesor rade asinhrono, pa mora da postoji mehanizam njihove sinhronizacije. U slučaju ulaza to znači da procesor na neki način mora da ima informaciju da je podatak prenet od strane kontrolera iz periferije u registar *Data* i da može da se pređe na izvršavanje programa kojim će se dati podatak prebaciti iz registra *Data* u odgovarajuću memorijsku lokaciju. U slučaju izlaza procesor mora da ima informaciju da je tekući podatak prenet od strane kontrolera iz registra *Data* u periferiju i da procesor može da pređe na izvršavanje programa kojim će se sledeći podatak prebaciti iz memorijske lokacije u registar *Data*.

Kao podrška za sinhronizaciju između procesora i kontrolera u okviru statusnog registra *Status* postoji poseban bit *ready*. Dvema vrednostima ovog bita kontroler ukazuje na dve situacije koje mogu da se jave u slučaju ulaza. Prva situacija je da je prenos podatka iz periferiju u registar Data od strane kontrolera u toku, pa procesor ne sme da pređe na izvršavanje programa kojim će se preneti podatak iz registra *Data* u memorijsku lokaciju. Druga situacija je da je podatak prenet od strane kontrolera iz periferije u registar *Data*, pa procesor može da pređe na izvršavanje programa kojim će preneti podatak iz registra *Data* u memorijsku lokaciju. Na sličan način dvema vrednostima bita *ready* kontroler ukazuje na dve situacije koje mogu da se jave u slučaju izlaza. Prva situacija je da je prenos podatka iz registra *Data* u periferiju od strane kontrolera u toku, pa procesor ne sme da pređe na izvršavanje programa kojim će preneti sledeći podatak iz memorijske lokacije u registar *Data*. Druga je da je podatak prenet od strane kontrolera iz registra *Data* u periferiju, pa procesor

može da pređe na izvršavanje programa kojim će preneti sledeći podatak iz memorijske lokacije u registar *Data*.

U daljim razmatranjima će se smatrati da vrednost 1 signala *ready* ukazuje da je registar *Data* raspoloživ, a vrednost 0 da nije raspoloživ. Raspoloživost registra *Data* u slučaju ulaza znači da je podatak od strane kontrolera priferije prenet iz periferije u registar *Data* i da procesor može da pređe na izvršavanje programa kojim će ovaj podatak preneti iz registra podatka *Data* u memorijsku lokaciju. Raspoloživost registra *Data* u slučaju izlaza znači da je tekući podatak od strane kontrolera prenet iz registra *Data* u periferiju i da procesor može da pređe na izvršavanje programa kojim će sledeći podatak biti prenet iz memorijske lokacije u registar *Data*. Ovakva sinhronizacija između procesora i kontrolera zahteva da procesor povremeno čita registar *Status* i vrši proveru da li bit *ready* ima vrednost 0 ili 1. Ukoliko se utvrdi da bit *ready* ima vrednost 0, to znači da registar *Data* nije raspoloživ. U tom slučaju ne sme da se pređe na izvršavanje programa kojim će se u slučaju ulaza prenostiti podatak iz registra *Data* u memorijsku lokaciju, a u slučaju izlaza iz memorijske lokacije u registrar *Data*. Ukoliko se utvrdi da bit *ready* ima vrednost 1, to znači da je registar *Data* raspoloživ. U tom slučaju sme da se pređe na izvršavaje programa kojim će se u slučaja ulaza prenositi podatak iz registra *Data* u memorijsku lokaciju, a u slučaju izlaza iz memorijske lokacije u registar *Data*.

Trenutak kada se kreće sa prenosom bloka podataka iz periferije u memoriju, veličina bloka podataka koji se prenosi i deo memorije u koji se blok podataka prenosi su pod programskom kontrolom. To znači da kada se u okviru neke obrade javi potreba za podacima sa neke periferije, mora se data obrada zaustaviti za određeno vreme i preći na izvršavanje programa u okviru koga će se startovati kontroler odgovarajuće periferije, zatim prenositi podaci iz periferije u određeni deo memorije i na kraju zaustaviti kontroler periferije. Po završetku prenosa, obrada, u okviru koje se koriste podaci iz dela memorije u koji su uneti podaci sa periferije, se može nastaviti. Trenutak kada se kreće sa prenosom bloka podataka iz memorije u periferiju, veličina bloka podataka koji se prenosi i deo memorije iz koga se blok podataka prenosi su, takođe, pod programskom kontrolom. U ovom slučaju se podrazumeva da su u okviru neke obrade sračunate vrednosti smeštane u neki deo memorije. Po završetku date obrade prelazi se na izvršavanje programa u okviru koga se staruje kontroler odgovarajuće periferije, zatim prenose podaci iz datog dela memorije u periferiju i na kraju zaustavlja kontroler periferije. Kao podrška za ovakav mehanizam startovanja kontrolera periferije, zatim prenosa bloka podataka i na kraju zaustavljanja kontrolera periferije, u okviru upravljačkog registra *Control* postoji bit *start*. Dvema vrednostima ovog bita se određuje da li je kontroler periferije i prenos bloka podataka startovan ili je kontroler priferije i prenos boka podataka zaustavljen. U daljim razmatranjima se podrazumeva da vrednost 1 bita *start* određuje da je kontroler startovan, a vrednost 0 da je zaustavljen. Startovanje i zaustavljanje kontrolera periferije se realizuje programskim putem upisivanjem u registar *Control* kontrolera periferije vrednosti koja na poziciji bita *start* ima vrednost 1 ili 0, respektivno. Vrednost 1 bita *start* omogućuje da upravljačka jedinica kontrolera priferije prenosi podatke iz periferije u registar *Data* i obratno, dok vrednost 0 onemogućuje taj prenos.

Po starovanju kontrolera ulazne periferije, kreće se sa prenosom podataka iz periferije u registar *Data*, a pri startovanju izlazne periferije sa prenosom podataka iz registra *Data* u periferiju. U slučaju ulazno/izlaznih periferija u okviru registra *Control* postoji bit *u/i*. Dvema vrednostima ovog bita se određuje da li dati ulazno/izlazni kontroler treba da radi u režimu ulaza ili izlaza. U daljim razmatranjima se podrazumeva da vrednost 1 bita *u/i* određuje režim ulaza, a vrednost 0 režim izlaza.

Provera raspoloživosti registra *Data* čitanjem registra *Status* i proverom da li bit *ready* ima vrednost 1 ili 0 je jedna od tehnika kojom se organizuje ulaz/izlaz. Ovo je moguće organizovati na dva načina. Prvi način predviđa da se u petlji čita registar *Status* i proverava bit *ready* i da se u petlji ostaje sve vreme dok bit *ready* ima vrednost 0. Kada se utvrdi da bit *ready* ima vrednost 1, izlazi se iz petlje, realizuje prenos jedne reči, pri čemu tom prilikom kontroler postavlja bit *ready* na vrednost 0, i vraća u petlju radi čekanja da registar *Data* bude ponovo raspoloživ i bit *ready* postane 1. Nezgoda sa ovim načinom je u tome da procesor ne vrši nikakvu drugu obradu i da najveći deo vremena čeka u petlji. Drugi način predviđa da procesor vrši neku drugu obradu, pa u okviru toga povremeno čita registar *Status* i proverava bit *ready*. Ukoliko se utvrdi da bit *ready* ima vrednost 0, produžava se sa tom obradom. Ukoliko se utvrdi da bit *ready* ima vrednost 1, realizuje se prenos jedne reči i produžava sa obradom. Nezgoda sa ovim načinom je u tome da za svaku periferiju treba podešavati učestanost čitanja registra *Status*. Ukoliko se to radi češće, onda će biti manji vremenski razmak između trenutka kada je registar *Data* postao raspoloživ i trenutka kada je to čitanjem registra *Status* utvrđeno. Međutim, u ovom slučaju će biti dosta prethodno neuspešnih čitanja registra *Status*. Ukoliko se to radi ređe, biće manje neuspešnih čitanja registra *Status*, ali će biti veći vremenski razmak između trunutka kada je registar *Data* postao raspoloživ i trenutka kada je to čitanjem registra *Status* utvrđeno.

Indikaciju o tome da je registar *Data* postao raspoloživ moguće je proslediti procesoru generisanjem signala prekida **intr** kad god kontroler upiše vrednost 1 u bit *ready* registra *Status*. U ovom slučaju procesor može dok traje prenos podatka iz periferije u registar *Data* ili obratno da vrši neku drugu obradu. Kada posle jednog prenetog podatka registar *Data* postane raspoloživ, kontroler, uz upisivanje vrednosti 1 u bit *ready*, generiše i signal prekida **intr**. Procesor kao reakcija na signal prekida **intr** na trenutak prekida tekuću obradu, skače na odgovarajuću prekidnu rutinu u okviru koje prenosi podatak iz registra *Data* u memoriju ili obratno, pri čemu tom prilikom kontroler postavlja bit *ready* na vrednost 0, pa se potom procesor vraća na prekinutu tekuću obradu. U okviru registra *Control* postoji i bit *enable* u koji se programskim putem može upisivati vrednost 1 ili 0. Ukoliko je u bitu *enable* vrednost 1, tada se pri upisu vrednosti 1 u bit *ready* registra *Status*, generiše signal prekida **intr**. Ukoliko je u bitu *enable* vrednost 0, tada nema generisanja signala prekida **intr**. Kod inicijalizacije i starovanja kontrolera u registar *Control* je potrebno u bit *enable* upisati vrednost 0 ili 1 u zavisnosti od toga da li je predviđeno da procesor programskim putem, čitanjem registra *Status* i proverom vrednosti bita *ready*, utvrđuje da je registar *Data* raspoloživ ili prijemom signala prekida **intr**.

Registar *Entry* sadrži broj ulaza u IV tabelu. Prilikom inicijalizacije sistema programskim putem se u registar *Entry* upisuje broj ulaza u IV tabelu u kome se nalazi adresa prekidne rutine date periferije. Kao reakcija na signal prekida **intr** procesor u okviru opsluživanja prekida šalje kontroleru vrednost 1 signala potvrde **inta** koji se koristi u kontroleru da se procesoru kao broj ulaza u IV tabelu pošalje sadržaj registra *Entry*.

Kombinaciona mreža za prepoznavanje ciklusa čitanja i upisa stalno proverava vrednost signala $\mathbf{M/\overline{IO}}$ i sadržaja na linijama **ABUS**. Ovde je pretpostavljeno da su ulazno/izlazni i memorijski prostori razdvojeni i da se vrednostima 1 i 0 signala $\mathbf{M/\overline{IO}}$ određuje da se na linijama **ABUS** nalazi adresa iz memorijskog ili ulazno/izlaznog adresnog prostora, respektivno. Ukoliko se pri vrednosti 0 signala $\mathbf{M/\overline{IO}}$ na linijama **ABUS** pojavi adresa nekog od registara kontrolera, signal **HIT** na izlazu DEKODERA ADRESA će imati vrednost 1, pa će pri vrednosti 0 signala $\mathbf{\overline{RDBUS}}$ ili $\mathbf{\overline{WRBUS}}$ vrednost 1 imati signali **stRD** ili **stWR**.

Upravljačka jedinica se aktivira po prijemu vrednosti 1 signala **stRD**, **stWR**, **inta** ili **start**. U slučaju vrednosti 1 signala **stRD** ili **stWR** upravljačka jedinica generiše signale neophodne da se sadržaj adresiranog registra pusti na linije **DBUS** ili da se sadržaja sa linija **DBUS** upiše u adresirani registar, dok u slučaju vrednosti 1 signala **inta** generiše signale neophodne da se sadržaj registra *Entry* pusti na linije **DBUS**. U sva tri slučaja uz pretpostavku da se radi o asinhronoj magistrali generiše se i odgovarajuća vrednost signala $\overline{\textbf{FCBUS}}$. U zavisnosti od toga da li je u registar *Control* upisana vrednost koja na poziciji bita *start* ima vrednost 1 ili 0, upravljačka jedinica ili generiše upravljačke signale neophodne za čitanje podataka iz periferije i upis u registar *Data* ili čitanje podataka iz registra *Data* i upis u periferiju ili ih ne generiše. Jedan mogući način razmene upravljačkih signala između upravljačke jedinice i periferije prilikom čitanja podataka iz periferije i upisa u registar *Data* i čitanja podataka iz registra *Data* i upisa u periferiju je dat u odeljku 1.2.3.

Pored navedenih bitova *start, u/i* i *enable*, upravljački registar *Control* može da ima i druge bitove. Broj i značenje tih bitova zavisi od karakteristika periferije i mogućih režima rada sa periferijom. Oni se koriste da se prilikom startovanja kontrolera periferije upisivanjem odgovarajućih vrednosti u bitove upravljačkog registra *Control* zada i željeni režim rada sa periferijom.

Pored navedenog bita *ready* statusni registar *Status* može da ima i druge bitove. Broj i značenje tih bitova zavisi od broja i vrste statusnih signala kojima periferija daje informacije kontroleru o mogućim neregularnostima u radu periferije, kao što je informacija da nema papira, da napajanje periferije nije uključeno i slično. Upravljačka jedinica povremeno čita vrednosti statusnih signala koji dolaze iz periferije i njihove vrednosti upisuje u odgovarajuće razrede statusnog registra *Status*. Pretpostavlja se da se programskim putem statusni registar *Status* povremeno čita, vrši analiza vrednosti bitova u statusnom registru *Status* i po otkrivanju vrednosti 1 na nekom od bitova šalje poruka o otkrivenoj neregularnosti.

1.2.2 PROGRAMIRANJE

Programiranje ulaza/izlaza korišćenjem kontrolera periferije je ilustrovano na primeru ulazno /izlazne periferije sa koje se unosi blok podataka u memoriju. U programu je uzeto da se upravljački registar, statusni registar i registar podatka nalaze na adresama koje su simbolički označene sa *Control*, *Status* i *Data*, respektivno. Početna adresa i veličina bloka memorije se zadaju kao neposredne veličine označene sa #blockadr i #blockcount, respektivno. Vrednost koja se upisuje u upravljački registar *Control* prilikom inicijalizacije i startovanja kontrolera periferije se zadaje kao neposredna veličina označena sa #modestart. Vrednost koja se koristi kod provere da li je u statusnom registru *Status* bit *ready* 1 ili 0, zadaje se kao neposredna veličina, označena sa #ready. Vrednost koja se upisuje u upravljački registar *Control* prilikom zaustavljanja kontrolera periferije zadaje se kao neposredna veličina označena sa #modestop. Uzet je dvoadresni procesor sa registrima opšte namene i razdvojenim ulazno/izlaznim i memorijskim adresnim prostorima.

Kod ulaza/izlaza korišćenjem kontrolera periferije, programskim putem se ostvaruje prenos podatka iz registra podatka kontrolera *Data* u memorijsku lokaciju i obratno, vodi evidencija o adresama memorijskih lokacija u koje treba upisivati i iz kojih treba čitati podatke i vodi evidencija o broju podataka koje treba preneti. Problem sinhronizovanja različitih brzina sa kojima podaci mogu da se čitaju sa periferija i upisuju u memorijske lokacije i obratno, rešava se na dva načina i to: programskim putem ispitivanjem bita *ready* statusnog registra kontrolera periferije *Status* (*poling*) i korišćenjem prekida izazvanog od strane kontrolera periferije (*prekid*). Pored toga programskim putem se vrši inicijalizacija i startovanje i zaustavljanje kontrolera periferije.

U ovom odeljku se razmatra programiranje ulaza/izlaza korišćenjem kontrolera bez direktnog pristupa memoriji i to čitanjem statusnog registra (slika 2) i generisanjem prekida (slika 3).

1.2.2.1 ULAZ/IZLAZ ČITANJEM STATUSNOG REGISTRA

Programiranje ulaza/izlaza čitanjem statusnog registra *Status* kontrolera periferije ilustrovano je na primeru ulazno/izlazne periferije sa koje se blok podataka unosi u memoriju (slika 2). Kompletan unos bloka podataka se realizuje programom koji je označen sa Glavni program.

U ovom programu se registar R1 koristi za generisanje adresa memorijskih lokacija u koje se upisuju podaci iz registra podatka *Data* kontrolera periferije, a registar R2 za vođenje evidencije o broju podataka koje treba preneti. Stoga se na početku prenosa početna adresa bloka memorije u koji se unose podaci, zadata kao neposredna veličina #blockadr, upisuje u registar R1, a veličina bloka podataka, zadata kao neposredna veličina #blockcount, u registar R2. Potom se, prebacivanjem neposredne veličine #modestart preko registra R3 u upravljački registar *Control* kontrolera periferije, vrši inicijalizacija i startovanje kontrolera periferije. Vrednost #modestart na pozicijama upravljačkog registra koje odgovaraju bitovima *start, enable* i *u/i*, ima 1, 0 i 1, respektivno, čime se startuje kontroler periferije, zabranjuje generisanje prekida i zadaje režim ulaza. Na pozicijama preostalih bitova su vrednosti koje se razlikuju od periferije do periferije i nisu predmet ovih razmatranja. Potom se u petlji prenosi sadržaj statusnog registra *Status* kontrolera periferije u registar R3 i vrši provera, korišćenjem neposredne veličine označene sa #ready, koja na poziciji bita *ready* statusnog registra *Status* ima 1 a na ostalima 0, da li je bit *ready* statusnog registra *Status* 0 ili 1. Sve dok je bit *ready* 0, što znači da registar podatka *Data* kontrolera periferije nije raspoloživ da se iz njega čita podatak, ostaje se u petlji. Prvi put kada se otkrije da je bit *ready* 1, što će se desiti kada kontroler periferije prenese podatak iz periferije u registar podatka *Data* kontrolera periferije, izlazi se iz petlje. Podatak se potom iz registra podatka preko registra R3 upisuje u memorijsku lokaciju na adresi određenoj sadržajem registra R1, što je označeno sa [R1]. Inkrementiranjem sadržaja registra R1 u njemu se formira adresa sledeće memorijske lokacije u koju treba upisati sledeći podatak. Dekrementiranjem sadržaja registra R2 u njemu se dobija vrednost koja ukazuje koliko još reči treba preneti. Ukoliko tom prilikom sadržaj registra R2 nije postao nula, vraća se na labelu LOOP i prenos sledeće reči.

Treba napomenuti da prilikom izvršavanja instrukcije IN Data, R3 kojom se prebacuje podatak iz registra podatka *Data* kontrolera periferije u registar R3 procesora, kontroler periferije u bit *ready* statusnog registra *Status* kontrolera periferije upisuje vrednost 0. Kontroler periferije će bit *ready* ponovo postaviti na 1 tek kada prenese sledeći podatak iz periferije u registar podatka *Data* kontrolera periferije. Da to nije tako urađeno, onda bi se, u slučaju sporih periferija, po povratku na početak petlje preko labele LOOP i čitanjem statusnog registra *Status* kontrolera periferije utvrđivalo da bit *ready* ima vrednost 1. To bi se dešavalo ne zbog toga što je kontroler periferije u registar podatka *Data* kontrolera periferije preneo novi podatak sa periferije i tom prilikom u bit *ready* upisao vrednost 1, već zbog toga što je vrednost 1 bita *ready* zaostala od prethodnog podatka. U tom slučaju bi se prošlo kroz petlju i prešlo na instrukcije kojima bi se isti podatak ponovo preneo u susednu memorijsku lokaciju.

Iz ovoga se vidi da se program sastoji iz unutrašnje i spoljašnje petlje. Unutrašnja petlja služi za utvrđivanje da li je novi podatak prenet iz periferije u registar podatka *Data* kontrolera periferije. Spoljašnja petlja služi za prenos podatka iz registra podatka *Data* kontrolera periferije u memorijsku lokaciju i ažuriranje sadržaja registra R1 sa adresom

sledeće memorijske lokacije i registra R2 sa brojem podataka koje još treba preneti. U unutrašnjoj petlji se vrti dok novi podatak ne bude raspoloživ u registru podatka *Data* kontrolera periferije. U spoljašnjoj petlji se vrti dok se ne prenesu svi podaci bloka iz periferiju u memoriju.

```
        Glavni program
        . . .
        MOV #blockadr, R1
        MOV #blockcount, R2
        MOV #modestart, R3
        OUT R3, Control
LOOP:   IN Status, R3
        AND R3, #ready
        JZ LOOP
        IN Data, R3
        MOV R3, [R1]
        INC R1
        DEC R2
        JNZ LOOP
        IN Control, R3
        AND R3, #modestop
        OUT R3, Control
        . . .
```

Slika 2 Programirani ulaz čitanjem statusnog registra

Kada sadržaj registra R2 postane nula, izlazi se iz spoljašnje petlje i prelazi na zadnje tri instrukcije kojima se zaustavlja kontroler periferije. Upravljački registar *Control* kontrolera periferije se, najpre, prebacuje u registar R3, zatim se, korišćenjem neposredne veličine označene sa #modestop, koja na pozicija bita *start* upravljačkog registra *Control* ima 0 a na ostalima 1, u registar R3 na poziciji bita *start* upravljačkog registra *Control* upisuje 0, dok se ostali bitovi ne menjaju, i na kraju sadržaj registra R3 upisuje u upravljački registar *Control* kontrolera periferije. S obzirom na to da vrednost koja se upisuje u upravljački registar *Control* kontrolera periferije ima vrednost 0 na poziciji bita *start*, zaustavlja se kontroler priferije. Zaustavljanje kontrolera periferije bi moglo da se realizuje i na taj način što bi se u registar R3 upisao sadržaj sve nule, a zatim sadržaj registra R3 upisao u upravljački registar *Control* kontrolera periferije. U ovom slučaju bi se vrednost 0 upisala u sve bitove upravljačkog registra *Control*, a ne samo u bit *start*, ali to ne bi imalo nikakvog efekta, jer se kontroler periferije zaustavlja.

Programirani izlaz čitanjem statusnog registra se realizuje programom koji je veoma sličan programu za programirani ulaz čitanjem statusnog registra (slika 2). Prva razlika je da neposredna veličina #modestart, čijim se upisivanjem preko registra R3 u upravljački registar *Control* kontrolera periferije vrši inicijalizacija i startovanje kontrolera periferije, mora na pozicijama upravljačkog registra koje odgovaraju bitovima *start*, *enable* i *u/i*, da ima 1, 0 i 0, respektivno, čime se startuje kontroler periferije, zabranjuje generisanje prekida i zadaje režim izlaza. Na pozicijama preostalih bitova su vrednosti koje se razlikuju od periferije do periferije i nisu predmet ovih razmatranja. Potom se u petlji prenosi sadržaj statusnog registra *Status* kontrolera periferije u registar R3 i vrši provera, korišćenjem neposredne veličine označene sa #ready, koja na poziciji bita *ready* statusnog registra *Status* ima 1 a na ostalima 0, da li je bit *ready* statusnog registra *Status* 0 ili 1. Sve dok je bit *ready* 0, što znači da registar podatka *Data* kontrolera periferije nije raspoloživ da se u njega upisuje podatak, ostaje se u

petlji. Prvi put kada se otkrije da je bit *ready* 1, što će se desiti kada prethodni podatak bude prenet iz registra podatka *Data* kontrolera periferije u periferiju, izlazi se iz petlje.

Druga razlika je da je umesto instrukcija

```
IN Data, R3
MOV R3, [R1]
```

kojima se vrši prebacivanje sadržaja registra podatka kontrolera u memorijsku lokaciju, potrebno staviti instrukcije

```
MOV [R1], R3
OUT R3, Data
```

kojima se vrši prebacivanje sadržaja memorijske lokacije u registar podatka kontrolera.

Treba napomenuti da prilikom izvršavanja instrukcije OUT R3, Data kojom se prebacuje podatak iz registra R3 procesora u registar podatka *Data* kontrolera periferije, kontroler periferije u bit *ready* statusnog registra *Status* kontrolera periferije upisuje vrednost 0. Kontroler periferije će bit *ready* ponovo postaviti na 1 tek kada prenese podatak iz registra podatka *Data* kontrolera u periferiju. Da to nije tako urađeno, onda bi se, u slučaju sporih periferija, po povratku na početak petlje preko labele LOOP i čitanjem statusnog registra *Status* kontrolera periferije utvrđivalo da bit *ready* ima vrednost 1. To bi se dešavalo ne zbog toga što je kontroler periferije iz registra podatka *Data* kontrolera periferije preneo podatak u periferije i tom prilikom u bit *ready* upisao vrednost 1, već zbog toga što je vrednost 1 bita *ready* zaostala od prethodnog podatka. U tom slučaju bi se prošlo kroz petlju i prešlo na instrukcije kojima bi se iz susedne memorijsku lokaciju u registar podatka *Data* kontrolera periferije preko prethodnog podatka upisao novi podatak.

U slučaju da se radi o sistemu kod koga je ulazno/izlazni adresni prostor memorijski preslikan, treba u opštem slučaju umesto instrukcija IN i OUT koristiti instrukciju MOV. Međutim, ukoliko se radi sa procesorima kod kojih u instrukciji MOV oba operanda mogu da budu memorijske lokacije, moguće je pri prenosu iz registra podatka *Data* kontrolera periferije u memorijsku lokaciju umesto instrukcija

```
MOV Data, R3
MOV R3, [R1]
```

da se stavi

```
MOV Data, [R1]
```

i pri prenosu iz memorijske lokacije u registar podatka *Data* kontrolera periferije umesto instrukcija

```
MOV [R1], R3
MOV R3, Data
```

da se stavi

```
MOV [R1], Data
```

Nedostatak programiranog ulaza/izlaza čitanjem statusnog registra *Status* kontrolera periferije je čekanje u unutrašnjoj petlji. Ako se radi sa sporim periferijama, najveći deo vremena će se provoditi u unutrašnjoj petlji.

1.2.2.2 ULAZ/IZLAZ GENERISANJEM PREKIDA

Programiranje ulaza/izlaza generisanjem prekida ilustrovano je na primeru ulazno/izlazne periferije sa koje se blok podataka unosi u memoriju (slika 3). Kompletan unos bloka podataka se realizuje programima koji su označeni sa Glavni program i Prekidna rutina. U glavnom programu se zadaju početna adresa dela memorije u koji se unosi blok podataka, veličina bloka podataka i vrši inicijalizacija i startovanje kontrolera periferije. U prekidnoj rutini se ostvaruje prenos podatka iz registra podatka *Data* kontrolera periferije u memorijsku

lokaciju, generisanje adrese memorijske lokacije u koju treba preneti sledeći podatak i vođenje evidencije o tome koliko još podataka bloka preba preneti. U prekidnoj rutini se, po prenosu zadnjeg podatka bloka, vrši i zaustavljanje kontrolera periferije.

U ovom programu se memorijska lokacija *mem1* koristi za generisanje adresa memorijskih lokacija u koje se upisuju podaci iz registra podatka *Data* kontrolera periferije, a memorijska lokacija *mem2* za vođenje evidencije o broju podataka koje treba preneti. Stoga se na početku prenosa početna adresa bloka memorije u koji se unose podaci, zadata kao neposredna veličina #blockadr, upisuje u memorijsku lokaciju *mem1*, a veličina bloka podataka, zadata kao neposredna veličina #blockcount, u memorijsku lokaciju *mem2*. Potom se, prebacivanjem neposredne veličine #modestart preko registra R3 u upravljački registar *Control* kontrolera periferije, vrši inicijalizacija i startovanje kontrolera periferije. Vrednost #modestart na pozicijama upravljačkog registra koje odgovaraju bitovima *start*, *enable* i *u/i*, ima 1, čime se startuje kontroler periferije, dozvoljava generisanje prekida i zadaje režim ulaza. Na pozicijama preostalih bitova su vrednosti koje se razlikuju od periferije do periferije i nisu predmet ovih razmatranja. Zatim se u memorijsku lokaciju *sem* upisuje vrednost 1, koja služi kao indikacija da je unos podataka u blok memorije u toku i da procesor ne sme da koristi podatke iz njega.

Potom se prelazi na izvršavanje programa u kome se ne koriste podaci iz bloka memorije u koji je unos podataka u toku. U toku izvršavanja ovog programa dolaze zahtevi za prekid od kontrolera periferije svaki put kada je novi podatak raspoloživ u registru podatka kontrolera. Kao rezultat toga za svaki podatak prebačen iz periferije u registar podatka *Data* kontrolera periferije skače se na prekidnu rutinu.

Na ulasku u prekidnu rutinu i izlasku iz prekidne rutine registar R3 se stavlja na stek i skida sa steka, respektivno, jer se u njo koristi. U prekidnoj rutini se podatak iz registra podatka *Data* kontrolera periferije preko registra R3 upisuje u memorijsku lokaciju na adresi određenoj sadržajem memorijske lokacije *mem1*, što je označeno sa [mem1]. Inkrementiranjem sadržaja memorijske lokacije *mem1* u njoj se formira adresa sledeće memorijske lokacije u koju treba upisati sledeći podatak. Dekrementiranjem sadržaja memorijske lokacije *mem2* u njoj se dobija vrednost koja ukazuje koliko još reči treba preneti. Ukoliko tom prilikom sadržaj memorijske lokacije *mem2* nije postao nula, vraća se preko labele BACK u prekinuti glavni program. Prilikom zadnjeg ulaska u prekidnu rutinu i prenosa zadnjeg podatka u blok memorije, sadržaj memorijske lokacije *mem2* postaje nula, pa se, umesto skoka na labelu BACK, produžava sa izvršavanjem prekidne rutine. Najpe se u memorijsku lokaciju *sem* upisuje vrednost nula, koja služi kao indikacija da je unos podataka u blok memorije završen i da procesor sme da koristi podatke iz njega. Zatim se prelazi na zadnje tri instrukcije kojima se zaustavlja kontroler periferije. Upravljački registar *Control* kontrolera periferije se, najpre, prebacuje u registar R3, zatim se, korišćenjem neposredne veličine označene sa #modestop, koja na pozicija bita *start* upravljačkog registra ima 0 a na ostalima 1, u registar R3 na poziciji bita *start* upravljačkog registra upisuje 0, dok se ostali bitovi ne menjaju, i na kraju sadržaj registra R3 upisuje u upravljački registar *Control* kontrolera periferije. S obzirom na to da vrednost koja se upisuje u upravljački registar *Control* kontrolera periferije ima nulu na poziciji bita *start*, zaustavlja se kontroler priferije. Potom se vraća u prekinuti glavni program. Zaustavljanje kontrolera periferije bi moglo da se realizuje i na taj način što bi se u registar R3 upisao sadržaj sve nule, a zatim sadržaj registra R3 upisao u upravljački registar *Control* kontrolera periferije. U ovom slučaju bi se vrednost 0 upisala u sve bitove upravljačkog registra *Control*, a ne samo u bit *start*, ali to ne bi imalo nikakvog efekta, jer se kontroler periferije zaustavlja.

```
        Glavni program
        . . .
        MOV #blockadr, mem1
        MOV #blockcount, mem2
        MOV #modestart, R3
        OUT R3, Control
        MOV #1, sem
        ! Program u kome se ne koriste
        ! podaci iz bloka memorije u koji
        ! se unose podaci sa periferije
LOOP:   CMP sem, #0
        JNZ LOOP
        ! Program u kome se koriste
        ! podaci iz bloka memorije u koji
        ! su uneti podaci sa periferije
        . . .
        Prekidna rutina
        PUSH R3
        IN Data, R3
        MOV R3, [mem1]
        INC mem1
        DEC mem2
        JNZ BACK
        MOV #0, sem
        IN Control, R3
        AND R3, #modestop
        OUT R3, Control
BACK:   POP R3
        RTI
```

Slika 3 Programirani ulaz generisanjem prekida

Izvršavanje dela glavnog programa u kome se ne koriste podaci iz bloka memorije u koji se unose podaci, može da traje duže ili kraće od vremena neophodnog za unošenje celog bloka podataka. Zbog toga, kada se u glavnom programu dođe do labele LOOP, mogu da se jave dve situacije.

Ukoliko je to vreme duže, tada će se zahtev za prekid i skok na prekidnu rutinu radi unošenja zadnjeg podatka bloka desiti pre nego što se u glavnom programu stigne na labelu LOOP. U prekidnoj rutini će se, između ostalog, zbog toga što se prenosi zadnji podatak bloka, u memorijsku lokaciju *sem* upisati vrednost nula. Po povratku u glavni program produžiće se njegovo izvršavanje bez daljih prekida od strane kontrolera periferije i stići do labele LOOP. Instrukcijom CMP će se utvrditi da je sadržaj memorijske lokacije *sem* nula, jer je prenos bloka podataka u memoriju kompletiran, pa će se preći na izvršavanje dela glavnog programa u kome se koriste podaci iz bloka memorije u koji su preneti podaci sa ulazne periferije.

Ukoliko je to vreme kraće, tada će se u glavnom programu stići na labelu LOOP pre kompletiranja prenosa bloka podataka u memoriju. Instrukcijom CMP će se utvrđivati da je sadržaj memorijske lokacije *sem* jedinica. Zbog toga se neće preći na izvršavanje dela glavnog

programa u kome se koriste podaci iz bloka memorije u koji se prenose podaci sa periferije. Umesto toga program će se vrteti u petlji sve vreme dok je sadržaj memorijske lokacije sem jedinica. Za to vreme unos podataka iz periferije u blok memorije će se odvijati do kompletiranja prenosa. Za svaki novi podatak prenet iz periferije u registar podatka *Data* kontrolera periferije, generisaće se prekid i, u zavisnosti od toga kada je zahtev za prekid generisan, iz instrukcije CMP ili JNZ će se skakati u prekidnu rutinu. U prekidnoj rutini će se podatak prenositi u memorijsku lokaciju, ažurirati sadržaji memorijskih lokacija *mem1* i *mem2* i, ukoliko nije preneta zadnja reč, vraćati na instrukciju CMP ili JNZ. Prilikom zadnjeg ulaska u prekidnu rutinu, između ostalog, će se upisati nula u memorijsku lokaciju *sem* i zaustaviti kontroler. Po povratku u glavni program na instrukciju CMP ili JNZ, prvim izvršavanjem instrukcije CMP će se utvrditi da je sadržaj memorijske lokacije *sem* nula, pa će se po izvršavanju instrukcije JNZ izaći iz petlje. Time se prelazi na izvršavanje programa u kome se koriste podaci iz bloka memorije u koji su uneti podaci sa ulazne periferije.

Programirani izlaz generisanjem prekida se realizuje programom koji je veoma sličan programu za programirani ulaz generisanjem prekida (slika 3). Prva razlika je da neposredna veličina #modestart, čijim se upisivanjem preko registra R3 u upravljački registar *Control* kontrolera periferije vrši inicijalizacija i startovanje kontrolera periferije, mora na pozicijama upravljačkog registra koje odgovaraju bitovima *start*, *enable* i *u/i*, da ima 1, 1 i 0, respektivno, čime se startuje kontroler periferije, dozvoljava generisanje prekida i zadaje režim izlaza.. Druga razlika je da je umesto instrukcija

```
IN Data, R3
MOV R3, [mem1]
```

kojima se vrši prebacivanje sadržaja registra podatka *Data* kontrolera periferije u memorijsku lokaciju, potrebno staviti instrukcije

```
MOV [mem1], R3
OUT R3, Data
```

kojima se vrši prebacivanje sadržaja memorijske lokacije u registar podatka *Data* kontrolera periferije.

U slučaju da se radi o sistemu kod koga je ulazno/izlazni adresni prostor memorijski preslikan, treba u opštem slučaju umesto instrukcija IN i OUT koristiti instrukciju MOV. Međutim, ukoliko se radi sa procesorima kod kojih u instrukciji MOV oba operanda mogu da budu memorijske lokacije, moguće je pri prenosu iz registra podatka *Data* kontrolera periferije u memorijsku lokaciju umesto instrukcija

```
MOV Data, R3
MOV R3, [mem1]
```

da se stavi

```
MOV Data, [mem1]
```

i pri prenosu iz memorijske lokacije u registar podatka *Data* kontrolera periferije umesto instrukcija

```
MOV [mem1], R3
MOV R3, Data
```

da se stavi

```
MOV [mem1], Data
```

Prednost programiranog ulaza/izlaza generisanjem prekida u odnosu na programirani ulaz/izlaz čitanjem statusnog registra je da dok traje unošenje podataka iz periferije u memoriju procesor ne čeka u petlji, već izvršava program kome nisu potrebni podaci čije je unošenje u toku. Izvršavanje ovog programa se prekida, radi skoka na prekidnu rutinu,

onoliko puta kolika je veličina bloka podataka u rečima. Ovo prekidanje je vremenski zanemarljivo ukoliko je periferija spora. Ukoliko je periferija brza, može se desiti da su prekidanja toliko česta da procesor više vremena troši na ulazak u prekidnu rutinu i povratak iz prekidne rutine, nego na izvršavanje programa. U slučaju veoma brze periferije može se čak desiti da procesor ne bude u stanju da prati podatke koji dolaze sa ulazne periferije ili koji treba da se šalju u izlaznu periferiju. U tom slučaju se moraju koristiti druga rešenja za ulaz/izlaz, kao što je kontroler sa direktnim pristupom memoriji.

1.2.3 POVEZIVANJE SA PERIFERIJOM

U ovom odeljku se razmatra jedan od mogućih načina povezivanja kontrolera i periferije pomoću paralelnog interfejsa. Kod ovog interfejsa postoji osam linija podataka DATA, jedna statusna linija, jedna upravljačka linija i linija startovanja/zaustavljanja periferije. Postoje dve moguće varijante povezivanja u zavisnosti od toga ko šalje signal po upravljačkoj a ko po statusnoj liniji, pri čemu u obe varijante kontroler vrednošću 1 po liniji *start* startuje ulaznu periferiju da krene sa čitanjem podataka i izlaznu periferiju da krene sa upisom podataka, dok se vrednošću 0 po liniji *start* zaustavlja periferiju.

1.2.3.1 Kontroler statusni signal – periferija upravljački signal

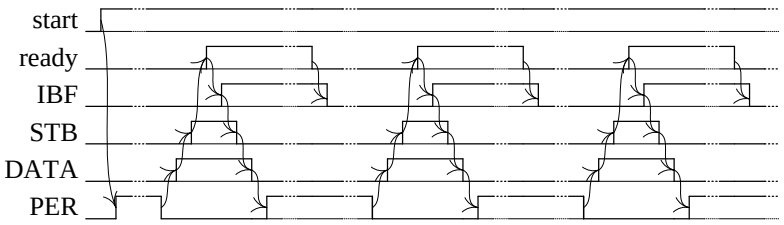
U slučaju ulazne periferije po statusnoj liniji IBF kontroler šalje periferiji indikaciju da li je registar *Data* kontrolera pun ili ne, po linijama DATA periferija šalje podatak kontroleru i po upravljačkoj liniji STB periferija generiše signal upisa sadržaja sa linija podataka DATA u registar *Data* kontrolera. U slučaju izlazne periferije po statusnoj liniji OBF kontroler šalje periferiji indikaciju da li je registar *Data* pun ili ne, po linijama DATA kontroler šalje podatak periferiji i po upravljačkoj liniji ACK periferija generiše signal potvrde da je sadržaj sa linija DATA prihvaćen u periferiju.

Kod ulazne periferije kontroler po statusnoj liniji IBF daje indikaciju da li može da primi sledeći podatak od periferije sa linija DATA. Uzeto je da je vrednost 1 statusnog signala IBF indikacija da je registar *Data* kontrolera pun i da kontroler ne može da primi sledeći podatak dok se tekući podatak ne prebaci iz registra *Data* kontrolera u memorijsku lokaciju, dok je vrednost 0 statusnog signala IBF indikacija da registar *Data* kontrolera nije pun i da kontroler može da primi sledeći podatak. Periferija po završetku čitanja stavlja sledeći podatak na linije DATA i proverava vrednost statusnog signala IBF. Ukoliko statusni signal IBF ima vrednost 0, periferija generiše vrednost 1 upravljačkog signala STB kojim upisuje sledeći podatak sa linija DATA u registar *Data* kontrolera. Ukoliko statusni signal IBF ima vrednost 1, periferija čeka da se tekući podatak prebaci iz registra *Data* kontrolera u memorijsku lokaciju i da kontroler postavi statusni signal IBF na vrednost 0, pa tek onda generiše vrednost 1 upravljačkog signala STB kojim upisuje sledeći podatak sa linija DATA u registar *Data* kontrolera. Po upisu sadržaja sa linija DATA u registar *Data* kontrolera, kontroler postavlja statusni signal IBF na vrednost 1.

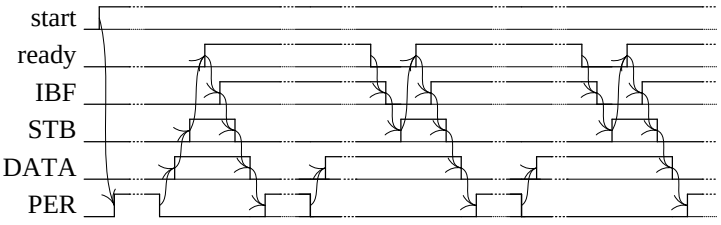
Prebacivanje tekućeg podatka iz registra *Data* iz kontrolera u memorijsku lokaciju i čitanje sledećeg podatka u periferiji se odvijaju paralelno. Prebacivanje tekućeg podatka iz registra *Data* realizuje poseban program, ukoliko se radi o kontroleru bez direktnog pristupa memoriji, ili sam kontroler, ukoliko se radi o kontroleru sa direktnim pristupom memoriji, a po završetku prebacivanja kontroler, bez obzira na to o kom tipu kontrolera se radi, postavlja statusni signal IBF na vrednost 0. Čitanje sledećeg podatka realizuje periferija i po završetku čitanja podatak stavlja na linije DATA i proverava vrednost statusnog signala IBF. Ukoliko se radi o sporoj periferiji, sadržaj registra *Data* kontrolera će tada već biti prebačen u memorijsku lokaciju i vrednost statusnog signala IBF će biti 0, pa periferija odmah generiše

vrednost 1 upravljačkog signala STB kojim upisuje sledeći podatak sa linija DATA u registar *Data* kontrolera. Ukoliko se radi o brzoj periferiji, sadržaj registra *Data* kontrolera još uvek neće biti prebačen u memorijsku lokaciju i vrednost statusnog signala IBF će biti 1, pa periferija čeka dok se tekući podatak ne prebaci iz registra *Data* kontrolera u memorijsku lokaciju i kontroler postavi statusni signal IBF na vrednost 0, pa tek onda generiše vrednost 1 upravljačkog signala STB kojim upisuje sledeći podatak sa linija DATA u registar *Data* kontrolera.

Vremenski oblici signala kontrolera, periferije i između kontrolera i periferije prilikom unosa bloka podataka iz spore i brze periferije u memoriju, dati su na slikama 4 i 5, respektivno. Signal *start* je interni signal kontrolera koji odgovara vrednosti bita *start* registra *Control*. Po startovanju kontrolera programskim putem signal *start* postaje 1 i ostaje 1 sve dok se kontroler programskim putem ne zaustavi. Signal *start* se po liniji *start* vodi iz kontrolera u periferiju. Vrednostima 1 i 0 po liniji *start* kontroler startuje i zaustavlja periferiju. Signal PER je interni signal periferije čija vrednost 1 označava da je čitanje podatka unutar periferije u toku, a vrednost 0 da je podatak pročitan i da se nalazi na linijama DATA spreman za upis u registar *Data* kontrolera. Signal *ready* je interni signal kontrolera koji odgovara vrednosti bita *ready* registra *Status*. Prilikom startovanju kontrolera za unos bloka podataka kontroler postavlja signal *ready* na vrednost 0. Kontroler postavlja signal *ready* na vrednost 1 kada se iz ulazne periferije po linijama DATA u registar *Data* kontrolera prebaci prvi podatak, i na vrednost 0 kada se sadržaj registra *Data* prebaci u memorijsku lokaciju. Statusni signal IBF, koji kontroler šalje periferiji, prati vrednost signala *ready* registra *Control*. Upravljački signal STB, koji ulazna periferija šalje kontroleru, periferija postavlja na vrednost 1 onda kada drži pročitani podatak na linijama DATA i kada statusni signal IBF ima vrednost 0, i služi da se upiše sadržaj sa linija DATA u registar *Data* kontrolera. Periferija postavlja upravljački signal STB na vrednost 0 onda kada kontroler postavi statusni signal IBF na vrednost 0 kao indikaciju da je sadržaj sa linija DATA upisan u registar *Data* kontrolera.



Slika 4 Vremenski oblici signala za sporu ulaznu periferiju



Slika 5 Vremenski oblici signala za brzu ulaznu periferiju

Po startovanju kontrolera situacija je ista i za sporu i za brzu periferiju. Prilikom upisa vrednosti 1 u bit *start* registra *Control* kontroler postavlja signal *start* na vrednost 1, čime se startuje čitanje prvog podatka u periferiji. Za vreme čitanja podatka signal PER ima vrednost 1, a po završetku čitanja signal PER postaje 0. Pročitani podatak periferija stavlja na linije podataka DATA, utvrđuje da statusni signal IBF ima vrednost 0, pa, generisanjem vrednosti 1 upravljačkog signala STB, upisuje sadržaj sa linija podataka DATA u registar *Data* kontrolera.

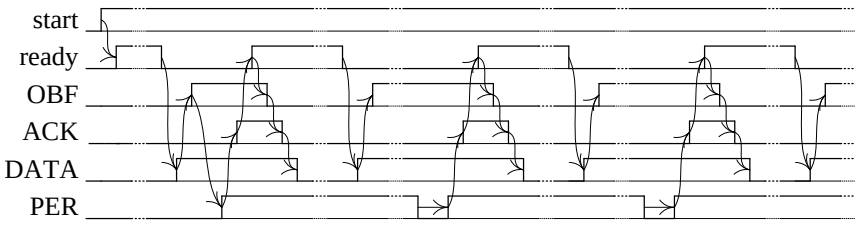
Kao reakcija na vrednost 1 upravljačkog signala periferije STB, kontroler postavlja najpre bit *ready* registra *Control* a zatim i statusni signal IBF na vrednost 1. Kao reakcija na vrednost 1 statusnog signala IBF kontrolera, periferija najpre postavlja upravljački signal STB na vrednost 0, a zatim uklanja sadržaj sa linija podataka DATA. Nadalje se paralelno odvija prebacivanje tekućeg podatka iz registra *Data* kontrolera u memorijsku lokaciju i čitanje sledećeg podatka u periferiji. Od ovog trenutka nastaje razlika između spore i brze periferije. U slučaju spore periferije se prebacivanje tekućeg podatka iz registra *Data* u memorijsku lokaciju i postavljanje najpre bita *ready* a zatim i statusnog signala IBF na vrednost 0, završavaju pre nego što je periferija pročitala sledeći podatak (slika 4). Zbog toga, periferija, čim pročita sledeći podatak i stavi ga na linije DATA, utvrđuje da statusni signal IBF ima vrednost 0, pa, generisanjem vrednosti 1 upravljačkog signala STB, upisuje sadržaj sa linija DATA u registar *Data* kontrolera. U slučaju brze periferije, na linije DATA periferija stavlja pročitani sledeći podatak pre nego što su prenošenje tekućeg podatka iz registra *Data* u memorijsku lokaciju i postavljanje najpre bita *ready* a zatim i statusnog signala IBF na vrednost 0 završeni (slika 5). Zbog toga periferija mora najpre da sačeka da se završi prebacivanje tekućeg podatka iz registra *Data* u memorijsku lokaciju i da najpre bit *ready* a zatim i statusni signal IBF postane 0, pa da zatim, generisanjem vrednosti 1 upravljačkog signala STB, upiše sadržaj sa linija DATA u registar *Data* kontrolera.

Kod izlazne periferije kontroler po statusnoj liniji OBF daje indikaciju da šalje sledeći podatak periferiji po linijama DATA. Uzeto je da je vrednost 1 statusnog signala OBF indikacija da je registar *Data* kontrolera pun i da kontroler šalje sledeći podatak po linijama DATA, dok je vrednost 0 statusnog signala OBF indikacija da registar *Data* kontrolera nije pun i da kontroler ne može da šalje podatak dok se sledeći podatak ne prebaci iz memorijske lokacije u registar *Data* kontrolera. Periferija po završetku upisa tekućeg podatka proverava vrednost statusnog signala OBF. Ukoliko statusni signal OBF ima vrednost 1, periferija prihvata sledeći podatak sa linija DATA i generiše vrednost 1 upravljački signala ACK kao potvrdu kontroleru da je prihvatila sadržaj sa linija DATA. Ukoliko statusni signal OBF ima vrednost 0, periferija čeka dok se sledeći podatak ne prebaci iz memorijske lokacije u registar *Data* kontrolera i postane raspoloživ na linijama DATA i kontroler postavi statusni signal OBF na vrednost 1, pa tek onda prihvata sledeći podatak sa linija DATA i generiše vrednost 1 upravljačkog signala ACK kao potvrdu kontroleru da je prihvatila sadržaj sa linija DATA. Pri pojavi vrednosti 1 upravljačkog signala ACK, kontroler postavlja statusni signal OBF na vrednost 0 i uklanja sadržaj sa linija DATA.

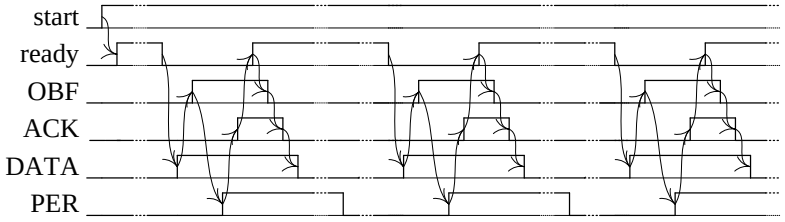
Prebacivanje sledećeg podatka iz memorijske lokacije u registar *Data* kontrolera i upis tekućeg podatka u periferiji se odvijaju paralelno. Prebacivanje sledećeg podatka u registar *Data* realizuje poseban program, ukoliko se radi o kontroleru bez direktnog pristupa memoriji, ili sam kontroler, ukoliko se radi o kontroleru sa direktnim pristupom memoriji, i po završetku prebacivanja kontroler, bez obzira na to o kom tipu kontrolera se radi, postavlja statusni signal OBF na vrednost 1. Upis tekućeg podatka realizuje periferija i po završetku upisa proverava vrednost statusnog signala OBF. Ukoliko se radi o sporoj periferiji, sledeći podatak će već biti prebačen iz memorijske lokacije u registar *Data* kontrolera i raspoloživ na linijama DATA i statusni signal OBF postavljen na vrednost 1 od strane kontrolera, pa periferija odmah prihvata sadržaj sa linija DATA i generiše vrednost 1 upravljačkog signala ACK kao potvrdu kontroleru da je prihvatila sledeći podatak sa linija DATA. Ukoliko se radi o brzoj periferiji, sledeći podatak još uvek neće biti prebačen iz memorijske lokacije u registar *Data* kontrolera i neće biti raspoloživ na linijama DATA i vrednost statusnog signala OBF će biti 0, pa periferija čeka dok se sledeći podatak ne prebaci iz memorijske lokacije u registar *Data* kontrolera i postane raspoloživ na linijama DATA i statusni signal OBF bude postavljen na vrednost 1 od

strane kontrolera, pa tek onda prihvata sadržaj sa linija DATA i generiše vrednost 1 upravljačkog signala ACK kao potvrdu kontroleru da je prihvatila sledeći podatak sa linija DATA.

Vremenski oblici signala kontrolera, periferije i između kontrolera i periferije prilikom slanja bloka podataka iz memorije u sporu i brzu periferiju, dati su na slikama 6 i 7, respektivno. Signal *start* je interni signal kontrolera koji odgovara vrednosti bita *start* registra *Control*. Po startovanju kontrolera programskim putem signal *start* postaje 1 i ostaje 1 sve dok se kontroler programskim putem ne zaustavi. Signal *start* registra *Control* se po liniji *start* vodi iz kontrolera u periferiju. Vrednostima 1 i 0 po liniji *start* kontroler startuje i zaustavlja periferiju. Signal PER je interni signal periferije čija vrednost 1 označava da je upis podatka unutar periferije u toku, a vrednost 0 da je podatak upisan i da je periferija spremna da prihvati sledeći podatak sa linija DATA. Signal *ready* je interni signal kontrolera koji odgovara vrednosti bita *ready* registra *Status* kontrolera. Prilikom startovanju kontrolera za slanje bloka podataka kontroler postavlja signal *ready* na vrednost 1. Kontroler postavlja signal *ready* na vrednost 0 kada se prvi podatak prebaci iz memorijske lokacije u registar *Data* kontrolera i postane raspoloživ na linijama DATA, i na vrednost 1 kada izlazna periferija prihvati podatak sa linija DATA. Statusni signal OBF, koji kontroler šalje periferiji, prati invertovanu vrednost signala *ready* registra *Status*. Upravljački signal ACK, koji izlazna periferija šalje kontroleru, periferija postavlja na vrednost 1 onda kada statusni signal OBF ima vrednost 1 i kada je sadržaj sa linija DATA periferija prihvatila i služi da periferija signalizira kontroleru da joj sadržaj sa linija DATA nije više potreban. Periferija postavlja upravljački signal ACK na vrednost 0 onda kada kontroler postavi statusni signal OBF na vrednost 0 kao indikaciju da se sadržaj registra *Data* kontrolera uklanja sa linija DATA.



Slika 6 Vremenski oblici signala za sporu izlaznu periferiju



Slika 7 Vremenski oblici signala za brzu izlaznu periferiju

Po startovanju kontrolera situacija je ista i za sporu i za brzu periferiju. Prilikom upisa vrednosti 1 u bit *start* registra *Control* kontroler postavlja bit *ready* registra *Status* na vrednost 1, čime se prelazi na prebacivanje prvog podatka iz memorijske lokacije u registar *Data* kontrolera, i signal *start* na vrednost 1, čime se startuje prihvatanje prvog podatka u periferiji. Po prebacivanju prvog podatka kontroler postavlja bit *ready* registra *Status* na vrednost 0, pušta sadržaj registra *Data* na linije DATA i postavlja statusni signal OBF na vrednost 1. Po startovanju periferija je spremna da prihvati prvi podatak, na šta ukazuje vrednost 0 signala PER, ali to ne sme da učini sve dok statusni signal OBF ima vrednost 0. Kao reakcija na postavljanje statusnog signala OBF na vrednost 1, periferija prihvata sadržaj sa linija DATA i

kreće sa upisom u periferiju, na šta ukazuje vrednost 1 signala PER. Pored toga, periferija, generisanjem vrednosti 1 upravljačkog signala ACK, ukazuje kontroleru da joj sadržaj sa linija DATA nije više potreban. Kao reakcija na vrednost 1 upravljačkog signala periferije ACK, kontroler postavlja najpre bit *ready* na vrednost 1 a zatim i statusni signal OBF na vrednost 0. Kao reakcija na vrednost 0 statusnog signala OBF kontrolera, periferija postavlja upravljački signal ACK na vrednost 0, a na to kontroler uklanja sadržaj sa linija DATA. Nadalje se paralelno odvija prebacivanje sledećeg podatka iz memorijske lokacije u registar *Data* kontrolera i upis tekućeg podatka u periferiji. Od ovog trenutka nastaje razlika između spore i brze periferije. U slučaju spore periferije prebacivanje sledećeg podatka iz memorijske lokacije u registar *Data* kontrolera, postavljanje najpre bita *ready* registra *Status* na 0, puštanje sledećeg podatka iz registra *Data* kontrolera na linije DATA i postavljanje statusnog signala OBF na vrednost 1, se završavaju pre nego što je periferija upisala prethodni podatak, na šta ukazuje vrednost 1 signala PER (slika 6). Zbog toga, periferija, čim završi sa upisom prethodnog podatka, na šta ukazuje vrednost 0 signala PER, prihvata sadržaj sa linija DATA i kreće sa upisom u periferiju, na šta ukazuje vrednost 1 signala PER. Pored toga, periferija, generisanjem vrednosti 1 upravljačkog signala ACK, ukazuje kontroleru da joj sadržaj sa linija DATA nije više potreban. U slučaju brze periferije, periferija završava sa upisom, na šta ukazuje vrednost 0 signala PER, pre nego što je završeno prebacivanje sledećeg podatka iz memorijske lokacije u registar *Data* kontrolera, postavljanje najpre bita *ready* registra *Status* na 0, puštanje sledećeg podatka iz registra *Data* kontrolera na linije DATA i postavljanje statusnog signala OBF na vrednost 1 (slika 7). Zbog toga periferija mora da čeka da vrednost statusnog signala OBF postane 1, pa da, najpre prihvati sadržaj sa linija DATA i krene sa upisom, na šta ukazuje vrednost 1 signala PER, i da zatim, generisanjem vrednosti 1 signala ACK, ukaže kontroleru da joj sadržaj sa linija DATA nije više potreban.

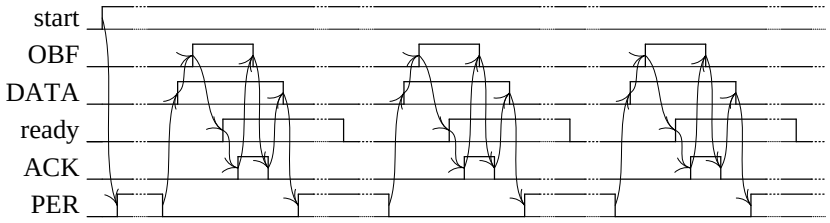
1.2.3.2 Periferija statusni signal – kontroler upravljački signal

U prethodnom odeljku je razmotran pristup povezivanja kontrolera i periferije, kod koga kontroler vrednošću 0 statusnog signala IBF šalje indikaciju ulaznoj periferiji da periferija može da upiše sledeći podatak u registar *Data* kontrolera i vrednošću 1 statusnog signala OBF šalje indikaciju izlaznoj periferiji da može da prihvati sledeći podatak iz registra *Data* kontrolera. Periferija je ta koja, ukoliko je ulazna, kada ima podatak i kada utvrdi da statusni signal IBF ima vrednost 0, upravljačkim signalom STB upisuje podatak sa linija DATA u registar *Data* kontrolera. Na sličan način, periferija je ta koja, ukoliko je izlazna, kada utvrdi da statusni signal OBF ima vrednost 1 i kada utvrdi da nema podatka čiji je upis u periferiji toku, prihvata podatak sa linija DATA registra *Data* kontrolera i o tome upravljačkim signalom ACK obaveštava kontroler.

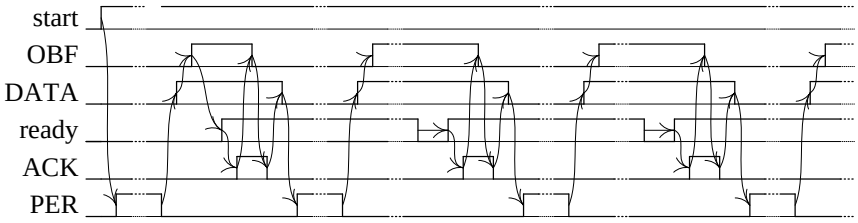
Povezivanje kontrolera i periferije je moguće realizovati i na taj način da ulazna periferija daje indikaciju kontroleru da na linijama DATA drži spreman podatak, a da kontroler realizuje prihvatanje podatka sa linija DATA i upis u registar *Data* kontrolera, kao i da izlazna periferija daje indikaciju kontroleru da može da primi sledeći podatak, a da kontroler preko linija DATA realizuje slanje podatka iz registra *Data* kontrolera u periferiju. Ulazna periferija vrednošću signala 1 statusnog signala OBF šalje indikaciju kontroleru da na linijama DATA drži sledeći podatak, a izlazna periferija vrednošću 0 statusnog signala IBF šalje indikaciju kontroleru da može da primi sledeći podatak sa linija DATA. Kontroler je sada taj koji, ukoliko je periferija ulazna, kada može da primi novi podatak iz periferije i kada utvrdi da statusni signal OBF ima vrednost 1, prihvata podatak sa linija DATA i upisuje u registar *Data* kontrolera i o tome signalom ACK obaveštava kontroler i, ukoliko je periferija izlazna, kada ima podatak u registru *Data* kontrolera koji drži na linijama DATA i ukoliko utvrdi da statusni signal IBF ima vrednost 0, signalom STB upisuje podatak sa linija DATA u periferiju. Signali start i PER

imaju isto značenje kao i odgovarajući signali u slučaju pristupa povezivanja kontrolera i periferije razmotrenog u prethodnom poglavlju za ulaznu i izlaznu periferiju.

Vremenski oblici signala kontrolera, periferije i između kontrolera i periferije prilikom unosa bloka podataka iz spore i brze periferije u memoriju, dati su na slikama 8 i 9, respektivno.



Slika 8 Vremenski oblici signala za sporu ulaznu periferiju

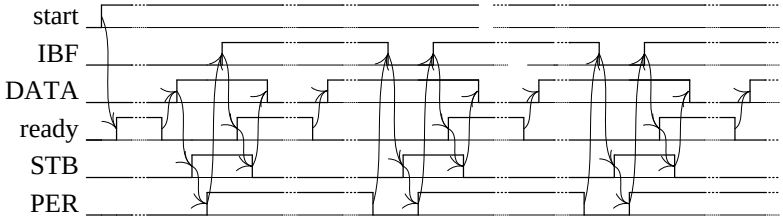


Slika 9 Vremenski oblici signala za brzu ulaznu periferiju

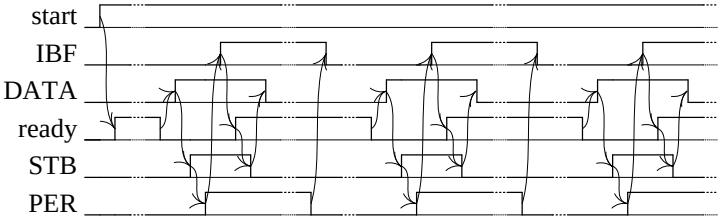
Po startovanju kontrolera situacija je ista i za sporu i za brzu periferiju. Prilikom upisa vrednosti 1 u bit *start* registra *Control* kontroler postavlja signal *start* na vrednost 1, čime periferija prelazi na čitanje prvog podatka, i bit *ready* registra *Status* na vrednost 0, čime kontroler prelazi na čekanje da ulazna periferija pročita podatak. Po startovanju čitanja prvog podatka u periferiji signal PER dobija vrednost 1. Po završetku čitanja signal PER dobija vrednost 0, pa periferija pušta pročitani podatak na linije DATA i postavlja statusni signal OBF na vrednost 1. Po startovanju kontroler je spreman da prvi podatak upiše sa linija DATA u registar *Data*, na šta ukazuje vrednost 0 signala *ready* registra *Status*, ali to ne sme da učini sve dok statusni signal OBF ima vrednost 0. Kao reakcija na postavljanje statusnog signala OBF na vrednost 1, kontroler prihvata sadržaj sa linija DATA i bit *ready* registra *Status* postavlja na vrednost 1. Pored toga, kontroler, generisanjem vrednosti 1 upravljačkog signala ACK, ukazuje periferiji da mu sadržaj sa linija DATA nije više potreban. Kao reakcija na vrednost 1 upravljačkog signala periferije ACK, periferija postavlja OBF na vrednost 0. Kao reakcija na vrednost 0 statusnog signala OBF kontrolera, kontroler postavlja upravljački signal ACK na vrednost 0, a na to periferija uklanja sadržaj sa linija DATA. Nadalje se paralelno odvija prebacivanje tekućeg podatka iz registra *Data* kontrolera u memorijsku lokaciju i čitanje sledećeg podatka u periferiji. Od ovog trenutka nastaje razlika između spore i brze periferije. U slučaju spore periferije prebacivanje tekućeg podatka iz registra *Data* kontrolera u memorijsku lokaciju i postavljanje bita *ready* registra *Status* na vrednost 0 se završavaju pre nego što je periferija pročitala sledeći podatak i pustila ga na linije DATA, na šta ukazuje vrednost 0 statusnog signala OBF (slika 8). Zbog toga kontroler mora da sačeka da periferija završi čitanje sledećeg podatka, pusti ga na linije DATA i statusni signal OBF postavi na vrednost1, pa da, najpre prihvati sadržaj sa linija DATA, postavi bit *ready* registra *Status* na vrednost 1 i da zatim, generisanjem vrednosti 1 signala ACK, ukaže periferiji da mu sadržaj sa linija DATA nije više potreban. U slučaju brze periferije, periferija završava čitanje sledećeg podatka, pušta ga na linije DAT i postavlja statusni signala OBF na vrednost 1, pre nego što je završeno prebacivanje tekućeg podatka iz registra *Data* kontrolera u memorijske

lokaciju i kontroler postavio bit *ready* registra *Status* na 0 (slika 9). Zbog toga, kontroler, čim završi sa prebacivanjem tekućeg tekućeg podatka, na šta ukazuje vrednost 0 bita *ready* registra *Status*, prihvata sadržaj sa linija DATA, postavlja bit *ready* registra *Status* na vrednost 1 i generisanjem vrednosti 1 upravljačkog signala ACK, ukazuje periferiji da mu sadržaj sa linija DATA nije više potreban.

Vremenski oblici signala kontrolera, periferije i između kontrolera i periferije prilikom slanja bloka podataka iz memorije u sporu i brzu periferiju, dati su na slikama 10 i 11, respektivno.



Slika 10 Vremenski oblici signala za sporu izlaznu periferiju



Slika 11 Vremenski oblici signala za brzu izlaznu periferiju

Po startovanju kontrolera situacija je ista i za sporu i za brzu periferiju. Prilikom upisa vrednosti 1 u bit *start* registra *Control* kontroler postavlja signal *start* na vrednost 1, čime se startuje prihvatanje prvog podatka u periferiji, i bit *ready* registra *Status* na vrednost 1, čime se prelazi na prebacivanje prvog podatka iz memorijske lokacije u registar *Data* kontrolera. Po startovanju periferija je spremna da prihvati prvi podatak, na šta ukazuju vrednosti 0 internog signala periferije PER i statusnog signala IBF. Po prebacivanju prvog podatka kontroler postavlja bit *ready* registra *Status* na vrednost 0, pušta sadržaj registra *Data* na linije DATA, utvrđuje da statusni signal IBF ima vrednost 0 i generiše vrednost 1 upravljačkog signala STB. Kao reakcija na vrednost 1 upravljačkog signala kontrolera STB, periferija upisuje podatak sa linija podataka DATA u ulazni registar periferije, postavlja interni signal PER na vrednost 1 kao indikaciju da je započet upis podatka iz ulaznog registra podatka periferije u periferiju i postavlja statusni signal IBF na vrednost 1 kao indikaciju da periferija nije spremna da prihvati sledeći podatak sve dok se ne završi upis podatka u periferiju i signal bude postavljen na vrednost 0. Kao reakcija na vrednost 1 statusnog signala IBF periferije, kontroler najpre postavlja bit *ready* registra *Status* na vrednost 0, zatim postavlja upravljački signal STB na vrednost 0 i na kraju uklanja sadržaj sa linija podataka DATA. Nadalje se paralelno odvija upis tekućeg podatka u periferiji i prebacivanje sledećeg podatka iz memorijske lokacije u registar *Data* kontrolera. Od ovog trenutka nastaje razlika između spore i brze periferije. U slučaju spore periferije se prebacivanje sledećeg podatka iz memorijske lokacije u registar *Data*, postavljanje bita *ready* na vrednost 0 i slanje sledećeg podatka iz registra *Data* po linijama podataka DATA desiće se pre nego što je periferija završila upis tekućeg podatka u periferiju (slika 10). Zbog toga kontroler mora najpre da sačeka da se završi upis tekućeg podatka u periferiju i da najpre interni signal PER a zatim i statusni signal IBF postanu 0, pa da zatim, generisanjem vrednosti 1 upravljačkog signala STB, upiše sadržaj sa linija DATA u

ulazni registar periferije. U slučaju brze periferije upis tekućeg podatka u periferiju će se završiti i najpre interni signal PER a zatim i statusni signal IBF postaviti na vrednost 0 pre nego što kontroler prebaci sledeći podatak iz memorijske lokacije u registar *Data* (slika 11). Zbog toga, kontroler, čim prebaci sledeći podatak iz memorijske lokacije u registar *Data*, postavlja bit *ready* registra *Status* na vrednost 0, pušta sadržaj registra *Data* na linije DATA, utvrđuje da statusni signal IBF ima vrednost 0 i generiše vrednost 1 upravljačkog signala STB i upisuje sadržaj sa linija DATA u ulazni registar periferije.

1.3 KONTROLER PERIFERIJE SA DIREKTNIM PRISTUPOM MEMORIJI

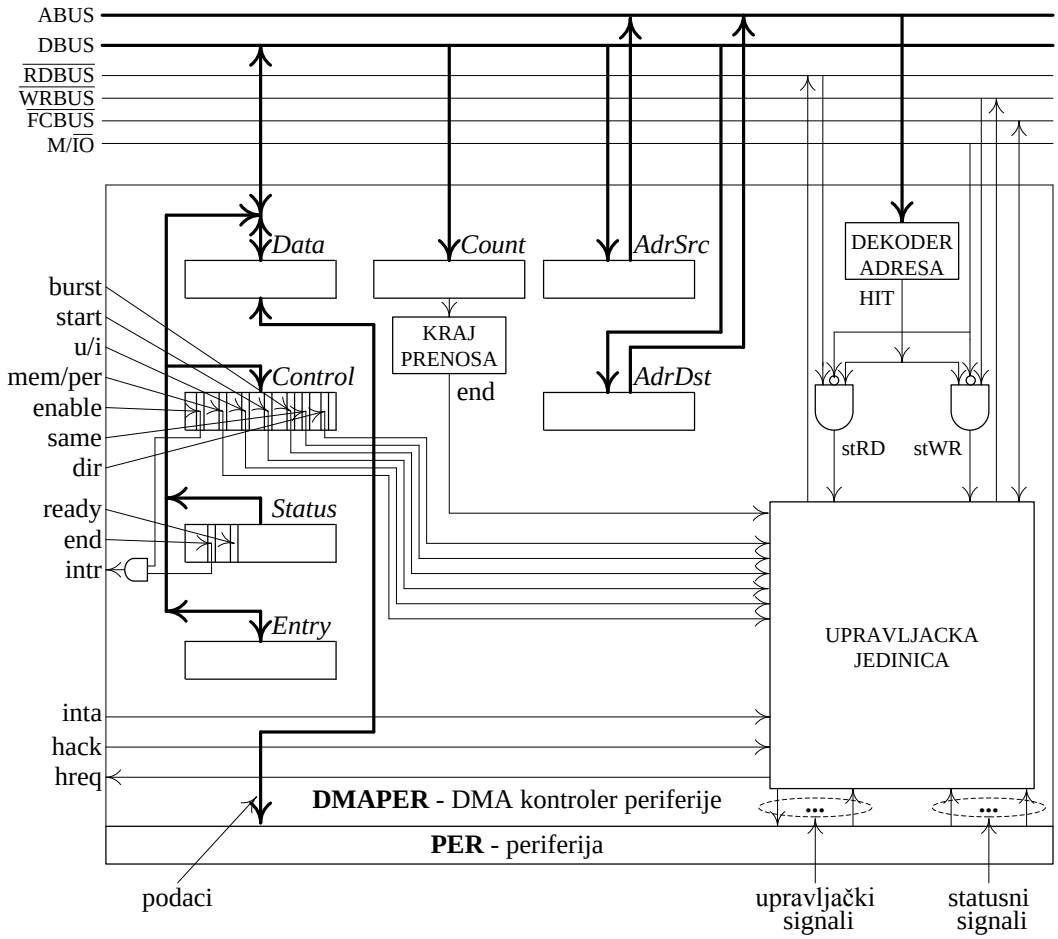
U ovom odeljku se razmatraju organizacija kontrolera periferije sa direktnim pristupom memoriji, njegovo programiranje radi organizacije ulaza/izlaza i povezivanje periferije korišćenjem kontrolera bez direktnog pristupa memoriji i kontrolera sa direktnim pristupom memoriji.

1.3.1 ORGANIZACIJA

U ovom odeljku se razmatra organizacija kontrolera periferije sa direktnim pristupom memoriji (slika 12). Kontroler se sastoji od operacione jedinice i upravljačke jedinice. Operacionu jedinicu čine registri *Data*, *Control*, *Status*, *Entry*, *AdrSrc*, *AdrDst* i *Count* i kombinaciona mreža DEKODER ADRESA za prepoznavanje ciklusa čitanja i upisa koji treba da se realizuju sa ovim registrima. Upravljačka jedinica realizuje čitanje iz i upis u registre *Data*, *Control*, *Status*, *Entry*, *AdrSrc*, *AdrDst* i *Count*, prenos podataka iz periferije u registar *Data* i obratno, arbitraciju za izlazak na magistralu, cikluse upisa u memoriju i čitanja iz memorije, vođenje evidencije o adresama memorijskih lokacija iz kojih treba čitati ili u koje treba upisivati i vođenje evidencije o broju podataka koje treba preneti između periferije i memorije i obratno.

U odnosu na kontroler bez direktnog pristupa memoriji, kontroler sa direktnim pristupom memoriji pri prenosu iz periferije u memoriju ne samo da čita podatak iz periferiju i upisuje u registar *Data* kontrolera, već i podatak iz registra *Data* kontrolera upisuje u memorijsku lokaciju na adresi određenoj sadržajem registra *AdrDst* kontrolera, a pri prenosu iz memorije u periferiju podatak prvo čita iz memorije sa adrese određene sadržajem registra *AdrSrc* kontrolera i upisuje u registar *Data* kontrolera, a zatim podatak iz registra *Data* kontrolera upisuje u periferiju. Pored toga, kontroler sa direktnim pristupom memoriji može da prenosi podatke iz jednog dela memorije u drugi deo memorije i pri tome podatak prvo čita iz memorije sa adrese određene sadržajem registra *AdrSrc* kontrolera i upisuje u registar *Data* kontrolera, a zatim podatak iz registra *Data* kontrolera upisuje u memorijsku lokaciju na adresi određenoj sadržajem registra *AdrDst* kontrolera.

Pre svakog upisa iz registra *Data* kontrolera u memorijsku lokaciju, kontroler najpre postavljanjem signala *hreq* na 1 traži od procesora dozvolu da koristi magistralu, pa upis iz registra *Data* kontrolera u memorijsku lokaciju realizuje tek kada mu procesor postavljanjem signala *hack* na 1 da dozvolu. Sve vreme dok traje upis iz registra *Data* kontrolera u memorijsku lokaciju na adresi *AdrDst* kontroler drži vrednost 1 signala *hreq*, a procesor vrednost 1 signala *hack*. Po završetku upisa kontrol ukida zahteva za korišćenje magistrale tako što signal *hreq* postavlja na 0, a procesor ukida dozvolu korišćenja magistrale tako što signal *hack* postavlja na 0.



Slika 12 Kontroler periferije sa direktnim pristupom memoriji

Na sličan način pre svakog čitanja iz memorijske lokacije i upisa u registar *Data* kontrolera, kontroler najpre postavljanjem signala *hreq* na 1 traži od procesora dozvolu da koristi magistralu, pa čitanje iz memorijske lokacije i upisa u registar *Data* kontrolera realizuje tek kada mu procesor postavljanjem signala *hack* na 1 da dozvolu. Sve vreme dok traje čitanje iz memorijske lokacije sa adrese *AdrSrc* i upis u registar *Data* kontroler držim vrednost 1 signala *hreq*, a procesor vrednost 1 signala *hack*. Po završetku čitanja kontrol ukida zahteva za korišćenje magistrale tako što signal *hreq* postavlja na 0, a procesor ukida dozvolu korišćenja magistrale tako što signal *hack* postavlja na 0.

Ovo važi za slučaj kada je magistrala u posedu procesora i kada kontroler od procesora traži dozvolu da koristi magistralu i kada procesor, ukoliko njemu nije potrebna magistrala, daje dozvolu korišćenja magistrale kontroleru. Na sličan način se, ukoliko postoji arbitrator magistrale od koga ne samo kontroleri već i procesor traži dozvolu korišćenja magistrale, postavljanjem signala *hreq* na 1 traži od arbitratora dozvola za korišćenje magistrale, a arbitrator postavljanjem signala *hack* na vrednost 1 daje dozvolu korišćenja magistrale. Takođe se postavljanjem signala *hreq* na 0 ukida arbitratoru zahtev za korišćenje magistrale, a arbitrator postavljanjem signala *hack* na vrednost 1 ukida dozvolu korišćenja magistrale.

Posle svakog upisa u memoriju/čitanja iz memorije sadržaj registra *AdrDst*/*AdrSrc* kontrolera se inkrementira, a sadržaj registra *Count* kontrolera dekrementira.

Ukoliko sadržaj registra *Count* dekrementiranjem nije stigao do nule, a radi se o prenosu iz periferije u memoriju, nastavlja se najpre sa čitanjem podatka iz periferije i upisom u registar *Data*, postavljanjem signala *hreq* na 1 i po dobijanju vrednosti 1 signala *hack* upisom podatka

iz registra *Data* u memorijsku lokaciju na adresi *AdrDst*, a po završetku upisa sa postavljanjem signala *hreq* na 0 i po dobijanju vrednosti 0 signala *hack* sa inkrementiranje sadržaja registra *AdrDst* i dekrementiranjem sadržaja registra *Count*. Ukoliko sadržaj registra *Count* dekrementiranjem nije stigao do nule, a radi se o prenosu iz memorije u periferiju, nastavlja se najpre sa postavljanjem signala *hreq* na 1 i po dobijanju vrednosti 1 signala *hack* čitanjem podatka iz memorije sa adrese *AdrSrc* i upisom u registar podatka *Data*, a po završetku čitanja sa postavljanjem signala *hreq* na 0 i po dobijanju vrednosti 0 signala *hack* sa slanjem podatka iz registra *Data* u periferiju i inkrementiranje sadržaja registra *AdrSrc* i dekrementiranjem sadržaja registra *Count*. Ukoliko sadržaj registra *Count* dekrementiranjem nije stigao do nule, a radi se o prenosu iz memorije u memoriju, nastavlja se sa čitanjem iz memorije sa adrese *AdrSrc* i upisom u registar podatka *Data* i upisom podatka iz registra *Data* u memorijsku lokaciju na adresi *AdrDst* kao i pri prenosu podatka iz memorije u periferiju i periferiju u memoriju, respektivno. Prenos podataka se završava kada sadržaj registra *Count* dekrementiranjem postane 0. Tada se u bit *end* registra *Status* kontrolera upisuje vrednost 1 i, ukoliko je bit *enable* registra *Control* postavljen na 1, generiše signal prekida **intr**.

Prenos podataka iz periferije u memoriju ili obratno i iz memorije u memoriju se realizuje tako što se izvršavanjem programa izvrši inicijalizacija kontrolera, u okviru koje se u registre *AdrDst*/*AdrSrc* kontrolera upišu početne adrese memorije a u registar *Count* kontrolera količina podataka koju treba preneti, i startovanje kontrolera, u okviru koga se u registar *Control* kontrolera upiše takav sadržaj da je na pozicija bita *start* vrednost 1. Kompletan prenos podataka realizuje sam kontroler na prethodno opisani način. Po završetku prenosa kontroler u bit *end* registra *Status* kontrolera upisuje vrednost 1 i, ukoliko je bit *enable* registra *Control* kontrolera postavljen na 1, generiše signal prekida **intr**. Procesor utvrđuje da je prenos završen ili čitanjem bita *end* registra *Status* kontrolera ili prijemom signala prekida **intr** od kontrolera i izvršavanjem programa zaustavlja kontroler tako što u registar *Control* kontrolera upiše takav sadržaja da je na pozicija bita *start* vrednost 0.

U upravljačkom registru *Control* kontrolera bit *mem/per* vrednostima 1 i 0 određuje da se radi o prenosu iz memorije u memoriju ili o prenosu između memorije i periferije, respektivno. Ukoliko se radi o prenosu iz memorije u memoriju vrednost bita *u/i* nije bitna. Ukoliko se radi o prenosu između memorije i periferije bit *u/i* vrednostima 1 i 0 određuje da se radi o prenosu iz periferije u memoriju ili o prenosu iz memorije i periferiju, respektivno.

U upravljačkom registru *Control* kontrolera bit *burst* vrednostima 0 i 1 određuje da se radi o pojedinačnom ili paketskom prenosu podataka, respektivno. Kod pojedinačnog prenosa podataka, kontroler za svaki podatak traži dozvolu korišćenja magistrale, dok kod paketskog prenosa podataka kontroler traži dozvolu za prenos celog bloka podataka.

U upravljačkom registru *Control* kontrolera postoji bit *same* čija je vrednost bitna jedino ukoliko je vrednošću 1 bita *mem/per* registra *Control* kontrolera zadat prenos iz memorije u memorije. Ukoliko bit *same* ima vrednost 0, vrši se prebacivanje iz jednog dela memorije u drugi deo memorije onako kako je objašnjeno. Ukoliko bit *same* ima vrednost 1, tada se izvorišni adresni registar *AdrSrc* koristi samo za jedno čitanje iz memorije i upis u registar podatka *Data*, a posle se isti sadržaj registra podatka *Data* upisuje u memoriju počev od adrese koja je data odredišnim adresnim registrom *AdrDst* i to onoliko puta kolika je vrednost *Count* registra.

U upravljačkom registru *Control* kontrolera postoji bit *dir* koji vrednostima 0 i 1 određuje da li se prenos realizuje prema višim ili nižim memorijskim lokacijama, respektivno. U skladu sa ovim vrednošću 0 ovog bita se određuje da sadržaj adresnih registara *AdrSrc* i *AdrDst* treba

inkrementirati, a vrednošću 1 da treba dekrementirati. Ovo važi i za prenose između periferije i memorije i za prenose iz memorije u memoriju.

1.3.2 PROGRAMIRANJE

Programiranje ulaza/izlaza korišćenjem kontrolera sa direktnim pristupom memoriji je ilustrovano na primeru ulazno /izlazne periferije sa koje se unosi blok podataka u memoriju. U programu je uzeto da se upravljački registar, statusni registar, adresni registar i registar veličine bloka podataka nalaze na adresama koje su simbolički označene sa *Control*, *Status*, *AdrSrc*, *AdrDst* i *Count*, respektivno. Početna adresa i veličina bloka memorije se zadaju kao neposredne veličine označene sa #blockadr i #blockcount, respektivno. Vrednost koja se upisuje u upravljački registar prilikom inicijalizacije i startovanja kontrolera periferije se zadaje kao neposredna veličina označena sa #modestart. Vrednost koja se upisuje u upravljački registar prilikom zaustavljanja kontrolera periferije zadaje se kao neposredna veličina označena sa #modestop. Uzet je dvoadresni procesor sa registrima opšte namene i razdvojenim ulazno/izlaznim i memorijskim adresnim prostorima.

Kod ulaza/izlaza korišćenjem kontrolera sa direktnim pristupom memoriji, programskim putem se vrši inicijalizacija i startovanje i zaustavljanje kontrolera periferije, dok sam kontroler ostvaruje prenos podataka iz periferije u registar podatka kontrolera i dalje u memorijsku lokaciju i obratno, vodi evidenciju o adresama memorijskih lokacija u koje treba upisivati i iz kojih treba čitati podatke i vodi evidencija o broju podataka koje treba preneti. Utvrđivanje da li je prenos bloka podatak kompletiran se realizuje ili programskim putem ispitivanjem bita *end* statusnog registra kontrolera periferije ili korišćenjem prekida izazvanog od strane kontrolera periferije.

Programiranje ulaza/izlaza generisanjem prekida ilustrovano je na primeru ulazno/izlazne periferije sa koje se blok podataka unosi u memoriju (slika 13). Programi koji se tom prilikom koriste su označeni sa Glavni program i Prekidna rutina. U glavnom programu se kontroleru zadaju početna adresa dela memoriju u koji se unosi blok podataka, veličina bloka podataka i režim rada i vrši startovanje kontrolera periferije. U prekidnoj rutini se, po prenosu zadnjeg podatka bloka, vrši zaustavljanje kontrolera periferije.

Prilikom prenosa bloka podataka kontroler koristi adresni registar za generisanje adresa memorijskih lokacija u koje se upisuju podaci iz registra podatka kontrolera periferije, a registar veličine bloka podataka za vođenje evidencije o broju podataka koje treba preneti. Stoga se na početku glavnog programa početna adresa bloka memorije u koji se unose podaci, zadata kao neposredna veličina #blockadr, upisuje u odredišni adresni registar kontrolera, a veličina bloka podataka, zadata kao neposredna veličina #blockcount, u registar veličine bloka podataka. Potom se, prebacivanjem neposredne veličine #modestart preko registra R3 u upravljački registar kontrolera periferije, vrši zadavanje režima rada i startovanje kontrolera periferije. Vrednost #modestart na pozicijama upravljačkog registra koje odgovaraju bitovima *start*, *enable* i *u/i*, ima 1, 1 i 1, a na pozicijama koje odgovaraju bitovima *mem/per*, *burst*, *same* i *dir*, ima 0, čime se startuje kontroler periferije, dozvoljava generisanje prekida i zadaje režim ulaza. Na pozicijama preostalih bitova se vrednosti koje se razlikuju od periferije do periferije i nisu predmet ovih razmatranja. Zatim se u memorijsku lokaciju *sem* upisuje vrednost 1, koja služi kao indikacija da je unos podataka u blok memorije u toku i da procesor ne sme da koristi podatke iz njega.

Potom se prelazi na izvršavanje programa u kome se ne koriste podaci iz bloka memorije u koji je unos podataka u toku. U toku izvršavanja ovog programa kontroler pebacuje podatak iz periferije u registar podatka i zatim iz registra podatka upisuje u emorijsku lokaciju na adresi

određenoj sadržajem adresnog registra kontrolera. Potom kontroler inkrementira sadržaj adresnog registra i dekrementira sadržaj registra veličine bloka. Na kraju kontroler proverava sadržaj registra veličine bloka. Ukoliko sadržaj registra nije nula, na isti način se prenosi sledeći podatak iz periferije u memorijsku lokaciju. Ukoliko je nula, u bit *ready* statusnog registra se upisuje 1 i generiše prekid, ukoliko je bit *enable* upravljačkog registra jedinica.

Glavni program

```
        . . .
        MOV #blockadr, R1
        OUT R1, AdrDst
        MOV #blockcount, R1
        OUT R1, Count
        MOV #modestart, R1
        OUT R1, Control
        MOV #1, sem
        ! Program u kome se ne koriste
        ! podaci iz bloka memorije u koji
        ! se unose podaci sa periferije
LOOP:   CMP sem, #0
        JNZ LOOP
        ! Program u kome se koriste
        ! podaci iz bloka memorije u koji
        ! su uneti podaci sa periferije
        . . .
```

```
        Prekidna rutina
        PUSH R1
        MOV #0, sem
        IN Control, R1
        AND R1, #modestop
        OUT R1, Control
BACK:   POP R1
        RTI
```

Slika 13 Programirani ulaz/izlaz sa prekidom

Na ulasku u prekidnu rutinu i izlasku iz prekidne rutine registar R1 se stavlja na stek i skida sa steka, respektivno, jer se u njoj koristi. U prekidnoj rutini se u memorijsku lokaciju *sem* upisuje vrednost nula, koja služi kao indikacija da je unos podataka u blok memorije završen i da procesor sme da koristi podatke iz njega. Zatim se prelazi na zadnje tri instrukcije kojima se zaustavlja kontroler periferije. Upravljački registar kontrolera se, najpre, prebacuje u registar R1, zatim se, korišćenjem neposredne veličine označene sa #modestop, koja na pozicija bita *start* upravljačkog registra ima 0 a na ostalima 1, u registar R1 na poziciji bita *start* upravljačkog registra upisuje 0, dok se ostali bitovi ne menjaju, i na kraju sadržaj registra R1 upisuje u upravljački registar kontrolera. S obzirom na to da vrednost koja se upisuje u upravljački registar kontrolera ima nulu na poziciji bita *start*, zaustavlja se kontroler priferije. Potom se vraća u prekinuti glavni program

Izvršavanje dela glavnog programa u kome se ne koriste podaci iz bloka memorije u koji se unose podaci, može da traje duže ili kraće od vremena neophodnog za unošenje celog bloka podataka. Zbog toga kada se u glavnom programu dođe do labele LOOP mogu da se jave dve situacije.

Ukoliko je to vreme duže, tada će se zahtev za prekid i skok na prekidnu rutinu desiti pre nego što se u glavnom programu stigne na labelu LOOP. U prekidnoj rutini će se, između ostalog, u memorijsku lokaciju *sem* upisati vrednost nula. Po povratku u glavni program produžiće se njegovo izvršavanje i stići do labele LOOP. Instrukcijom CMP će se utvrditi da je sadržaj memorijske lokacije *sem* nula, jer je prenos bloka podataka u memoriju kompletiran, pa će se preći na izvršavanje dela glavnog programa u kome se koriste podaci iz bloka memorije u koji su preneti podaci sa ulazne periferije.

Ukoliko je to vreme kraće, tada će se u glavnom programu stići na labelu LOOP pre kompletiranja prenosa bloka podataka u memoriju. Instrukcijom CMP će se utvrđivati da je sadržaj memorijske lokacije *sem* jedinica. Zbog toga se neće preći na izvršavanje dela glavnog programa u kome se koriste podaci iz bloka memorije u koji se prenose podaci sa periferije. Umesto toga program će se vrteti u petlji sve vreme dok je sadržaj memorijske lokacije *sem* jedinica. Za to vreme unos podataka u blok memorije će se odvijati do kompletiranja prenosa. Kada kontroler prenese zadnji podatak iz periferije u memoriju, generisaće se prekid i, u zavisnosti od toga kada je zahtev za prekid generisan, iz instrukcije CMP ili JNZ će se skakati u prekidnu rutinu. U prekidnoj rutini će se upisati nula u memorijsku lokaciju *sem* i zaustaviti kontroler. Po povratku u glavni program na instrukciju CMP ili JNZ, prvim izvršavanjem instrukcije CMP će se utvrditi da je sadržaj memorijske lokacije *sem* nula, pa će se po izvršavanju instrukcije JNZ izaći iz petlje. Time se prelazi na izvršavanje programa u kome se koriste podaci iz bloka memorije u koji su uneti podaci sa ulazne periferije.

Utvrđivanje da je kontroler sa direktnim pristupom memoriji preneo ceo blok podataka može se realizovati i povremenim čitanje statusnog registra kontrolera i proverom bita *end*. Prelazak najpre na program u kome će se zaustavlja kontroler a zatim i na program u kome se koriste podaci iz bloka memorije u koji su uneti podaci, se realizuje po otkrivanju vrednosti 1 u bitu *end* satusnog registra.

Programirani izlaz generisanjem prekida se realizuje programom koji je veoma sličan programu za programirani ulaz generisanjem prekida (slika 3). Razlika je da se početna adresa bloka memorije iz koga se šalju podaci, zadata kao neposredna veličina #blockadr, upisuje u izvorišni adresni registar kontrolera i da neposredna veličina #modestart, čijim se upisivanjem preko registra R3 u upravljački registar kontrolera periferije vrši inicijalizacija i startovanje kontrolera periferije, mora na pozicijama upravljačkog registra koje odgovaraju bitovima *start*, *enable* i *u/i*, da ima 1, 1 i 0, respektivno, a na pozicijama koje odgovaraju bitovima *mem/per*, *burst*, *same* i *dir*, ima 0, čime se startuje kontroler periferije, dozvoljava generisanje prekida i zadaje režim izlaza.

U slučaju da se radi o sistemu kod koga je ulazno/izlazni adresni prostor memorijski preslikan, treba u opštem slučaju umesto instrukcija IN i OUT koristiti instrukciju MOV. Međutim, ukoliko se radi sa procesorima kod kojih u instrukciji MOV oba operanda mogu da budu memorijske lokacije, moguće je pri prenosu neposredne veličine #blockadr u adresni registar kontrolera umesto instrukcija

```
MOV #blockadr, R1
OUT R1, AdrSrc
```

da se stavi

```
MOV #blockadr, AdrSrc
```

Isto važi i za prenos neposredne veličine #blockcount u registar broja reči i neposredne veličine #modestart u upravljački registar.

Prednost programiranog ulaza/izlaza korišćenjem kontrolera sa direktnim pristupom memoriji u odnosu na programirani ulaz/izlaz korišćenjem kontrolera bez direktnog pristupa memoriji i generisanjem prekida je da dok kontroler ne kompletira unos celog boka podataka iz periferije u memoriju nema nema prekidanja izvršavanja programa kome nisu potrebni podaci čije je unošenje u toku.

Kontroler pored prenosa podataka iz periferije u memoriju i iz memorije u periferiju može da realizuje i prenos podataka iz memorije u memoriju, pri čemu se vrsta prenosa zadaje bitovima *mem/per* i *u/i* u upravljačkom registru *Control*. Ukoliko bit *mem/per* ima vrednost 0, kontroler realizuje prenosa podataka iz periferije u memoriju ili iz memorije u periferiju u zavisnosti od vrednosti bita *u/i*. Ukoliko bit *mem/per* ima vrednost 1, kontroler realizuje prenosa podataka iz memorije u memoriju i tada vrednost bita *u/i* nije bitna. U ovom kontroleru postoje dva adresna registra i to izvorišni i odredišni adresni registri *AdrSrc* i *AdrDst*. Pri prenosu iz periferije u memoriju koristi se samo odredišni adresni registar *AdrDst*, dok se pri prenosu iz memorije u periferiju koristi samo izvorišni adresni registar *AdrSrc*. Pri prenosu iz memorije u memoriju adresni registar *AdrSrc* se koristi za čitanje iz memorije i prebacivanje u registar podatka *Data*, a adresni registar *AdrDst* za upis iz registra podatka *Data* u memorijsku lokaciju. Na magistrali se realizuju dva posebna ciklusa i to ciklus čitanja sa adrese *AdrSrc* i ciklus upisa na adresi *AdrDst*. Prenos iz memorije u memoriju se realizuje onoliko puta kolika je vrednost *Count* registra.

U slučaju veoma brze i vremenski kritične periferije može se čak desiti da kontroler ne bude u stanju da prati podatke koji dolaze sa periferije ili koji treba da se šalju u periferiju. To je zbog toga što za svaki podatak koji treba kontroler da prenese iz registra podatka u

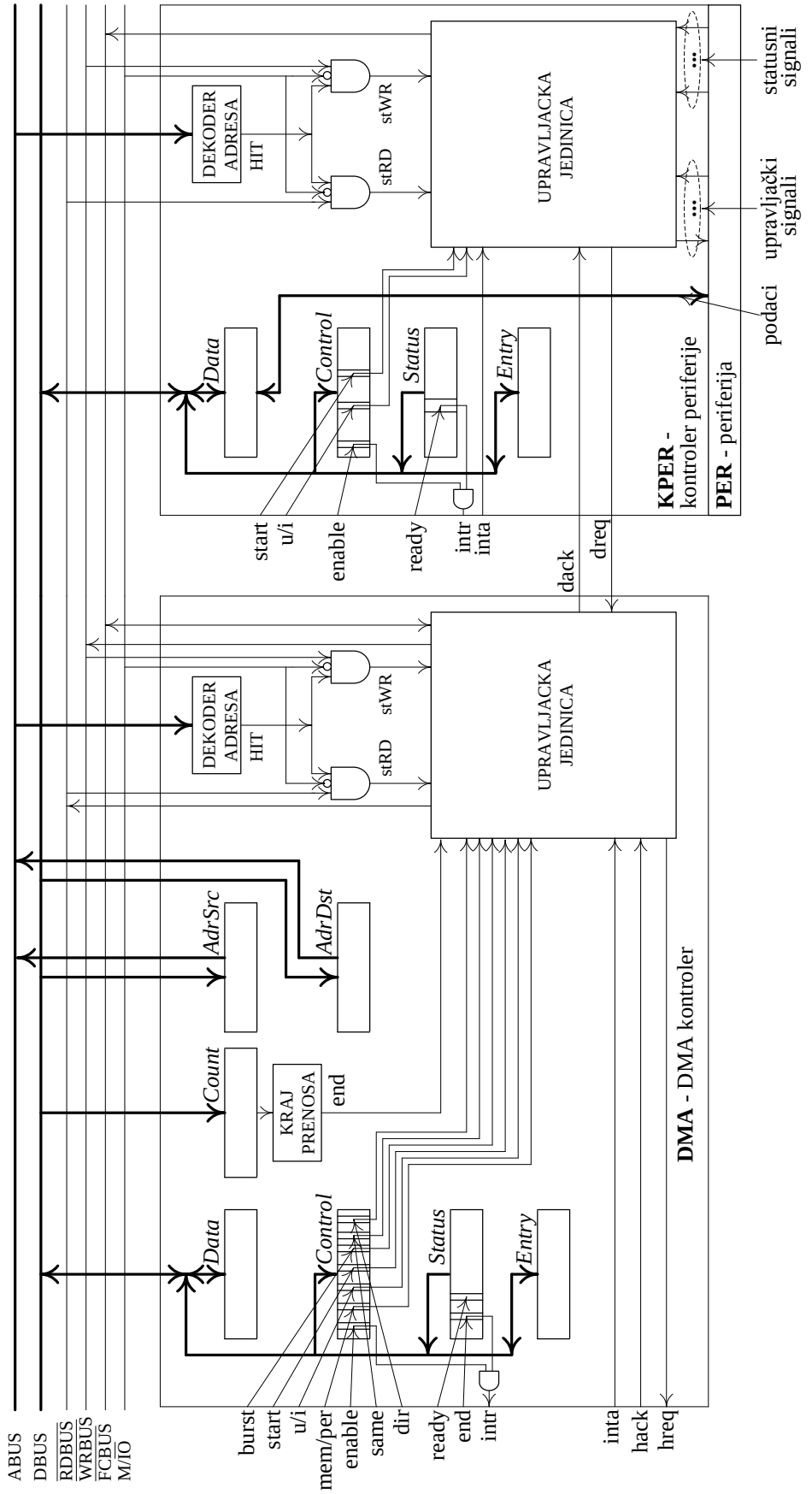
memorijsku lokaciju ili obratno, kontroler mora da traži dozvolu za korišćenje magistrale i da tek po dobijanju dozvole realizuje ciklus na magistrali. Da bi se ovaj problem rešio, kontroler se može, upisivanjem vrednosti 1 u bit *burst* upravljačkog registra, programirati da prenos bloka podataka realizuje u paketskom režimu rada. U ovom režimu rada kontroler, po dobijanju dozvole korišćenja magistrale, drži magistralu zauzetu sve vreme dok ne prenese ceo blok podataka i tek po završetku prenosa ukida zahtev za korišćenje magistrale.

Postoji i varijanta kontrolera koji istu vrednost pročitanu iz jedne memorijske lokadije prebacuje u niz susednih memorijskih lokacija. Taj režim rada se zadaje dodatnim bitom *memsame* u upravljačkom registru *Control*. Ukoliko bit *mem/per* ima vrednost 0, kontroler realizuje prenosa podataka iz periferije u memoriju ili iz memorije u periferiju u zavisnosti od vrednosti bita *u/i*, i tada vrednost bita *memsame* nije bitna. Ukoliko bit *mem/per* ima vrednost 1, kontroler realizuje prenosa podataka iz memorije u memoriju, i tada je vrednost bita *memsame* bitna. Ukoliko bit *memsame* ima vrednost 0, vrši se prebacivanje iz jednog dela memorije u drugi deo memorije onako kako je objašnjeno u prethodnom pasusu. Ukoliko bit *memsame* ima vrednost 1, tada se izvorišni adresni registar *AdrSrc* koristi samo za jedno čitanje iz memorije i upis u registar podatka *Data*, a posle se isti sadržaj registar podatka *Data* upisuje u memoriju počev od adrese koja je data odredišnim adresnim registrom *AdrDst* i to onoliko puta kolika je vrednost *Count* registra.

Postoji i varijanta kontrolera kod koje se može dadati režim prenosa prema višim i nižim memorijskim lokacijama. Postoji poseban bit *dir* u upravljačkom registru *Control*, pa se vrednošću 0 ovog bita određuje da sadržaj adresnih registara *AdrSrc* i *AdrDst*, treba inkrementirati, a vrednošću 1 da treba dekrementirati. Ovo je moguće i za prenose između periferije i memorije i za prenose iz memorije u memoriju.

1.3.3 POVEZIVANJE KONTROLERA

Postoje realizacije kontrolera sa direktnim pristupom memoriji kod kojih ne postoji direktna veza između kontrolera i periferije, pa se tada periferija povezuje na uobičajeni način sa kontrolerom bez direktnog pristupa memoriji (slika 14). U slučaju ulazne periferije nonDMA kontroler (kontroler bez direktnog pristupa memoriji) prihvata podatak iz periferije i smešta u svoj registar podatka *Data*, dok DMA kontroler (kontroler sa direktnim pristupom memoriji) realizuje upis podatka iz registra podatka *Data* nonDMA kontrolera u memorijsku lokaciju. U slučaju izlazne periferije DMA kontroler realizuje čitanje podatka iz memorije i smeštanje u registar podatka *Data* nonDMA kontrolera, dok nonDMA kontroler realizuje slanje podataka iz svog registra podatka *Data* u periferiju. Sinhronizacija rada ova dva kontrolera se realizuje razmenom signala po linijama dreq i dack.



Slika 14 Povezivanje DMA kontrolera i kontrolera periferije

Programskim putem se vrši inicijalizacija, startovanje i zaustavljanje oba kontrolera. U okviru inicijalizacije se u registre *AdrDst* i *Count* DMA kontrolera upisuje adresa memorijske lokacije i veličina bloka podataka. U okviru startovanja se u upravljačke registre *Control* DMA i nonDMA kontrolera upisuju vrednosti koje na poziciji bitova *start* i *u/i* imaju vrednost 1.

U slučaju ulazne periferije nonDMA kontroler čita podatak iz periferije, smešta ga u registar podatka *Data* nonDMA kontrolera i o tome obaveštava DMA kontroler postavljanjem signala dreq na vrednost 1. DMA kontroler vrednošću 1 signala **hreq** od procesora traži dozvolu korišćenja magistrale. Pošto vrednošću 1 signala **hack** od procesora dobije dozvolu korišćenja magistrale, DMA kontroler najpre vrednošću 1 signala AE pušta sadržaj registra *AdrDst* na adresne linije magistrale ABUS, zatim vrednošću 1 signala dack omogućava da nonDMA kontroler pusti sadržaj registra *Data* na linije podataka magistrale DBUS i na kraju startuje ciklus upisa tako što upravljačku liniju magistrale $\overline{\textbf{WRBUS}}$ postavlja na vrednost 0. Pošto od memorije dobije vrednost 0 signala $\overline{\textbf{FCBUS}}$ kao indikaciju da je upis završen, DMA kontroler postavlja signal $\overline{\textbf{WRBUS}}$ na vrednost 1, a potom memorija na vrednost 1 signala $\overline{\textbf{WRBUS}}$ postavlja signal $\overline{\textbf{FCBUS}}$ na vrednost 1. Na vrednost 1 signala $\overline{\textbf{FCBUS}}$ DMA kontroler najpre vrednošću 0 signala AE uklanja sadržaj registra *AdrDst* sa adresnih linija magistrale ABUS, a zatim vrednošću 0 signala dack omogućava da nonDMA kontroler ukloni sadržaj registra *Data* sa linija podataka magistrale DBUS i da signal dreq postavi na vrednost 0. Na vrednost 0 signala dreq DMA kontroler postavlja signal zahteva za korišćenje magistrale **hreq** na 0, a na to procesor postavljanjem signal **hack** na vrednost 0 ukida dozvolu koriščenja magistrale. Potom DMA kontroler inkrementira registar *AdrDst* i dekrementira registar *Count*, dok nonDMA kontroler čita sledeći podatak iz periferije. Kada se podatak upiše iz periferije u registar *Data*, nonDMA kontrolera o tome obaveštava DMA kontroler postavljanjem signala dreq na vrednost 1, pa se prethodno opisani postupak upisa podatka iz registra *Data* nonDMA kontrolera u memorijsku lokaciju ponavlja sve dok registar broja *Count* ne dođe do nule.

U slučaju izlazne periferije nonDMA kontroler postavljanjem signala dreq na vrednost 1 obaveštava DMA kontroler da treba da krene sa čitanjem podatka iz memorijske lokacije i prebacivanjem u registar podatka *Data* nonDMA kontrolera. Na vrednost 1 signala dreq DMA kontroler vrednošću 1 signala **hreq** od procesora traži dozvolu korišćenja magistrale. Pošto vrednošću 1 signala **hack** od procesora dobije dozvolu korišćenja magistrale, DMA kontroler najpre vrednošću 1 signala AE pušta sadržaj registra *AdrSrc* na adresne linije magistrale ABUS, a zatim startuje ciklus ćitanja tako što upravljačku liniju magistrale $\overline{\textbf{RDBUS}}$ postavlja na 0. Pošto od memorije dobije vrednost 0 signala $\overline{\textbf{FCBUS}}$ kao indikaciju da je čitanje završeno, DMA kontroler vrednošću 1 signala dack omogućava da nonDMA kontroler najpre upiše sadržaj sa linija podataka magistrale DBUS u registar *Data* i da potom signal dreq postavi na vrednost 0. Na vrednost 0 signala dreq DMA kontroler postavlja na vrednost 0 signal DACK i na vrednost 1 signal $\overline{\textbf{RDBUS}}$. Memorija na vrednost 1 signala $\overline{\textbf{RDBUS}}$ postavlja signal $\overline{\textbf{FCBUS}}$ na vrednost 1. Na vrednost 1 signala $\overline{\textbf{FCBUS}}$ DMA kontroler najpre vrednošću 0 signala AE uklanja sadržaj registra *AdrSrc* sa adresnih linija magistrale ABUS, a zatim postavlja signal zahteva za korišćenje magistrale **hreq** na 0, a na to procesor postavljanjem signal **hack** na vrednost 0 ukida dozvolu koriščenja magistrale. Potom DMA kontroler inkrementira registar *AdrSrc* i dekrementira registar *Count*, dok nonDMA kontroler prebacuje podatak iz registra *Data* u periferiju. Kada se podatak upiše u periferiju, nonDMA kontrolera o tome obaveštava DMA kontroler postavljanjem signala dreq na vrednost 1, pa se

prethodno opisani postupak čitanja podatka iz memorijske lokacije i upis u registar *Data* ponavlja sve dok registar broja *Count* ne dođe do nule.

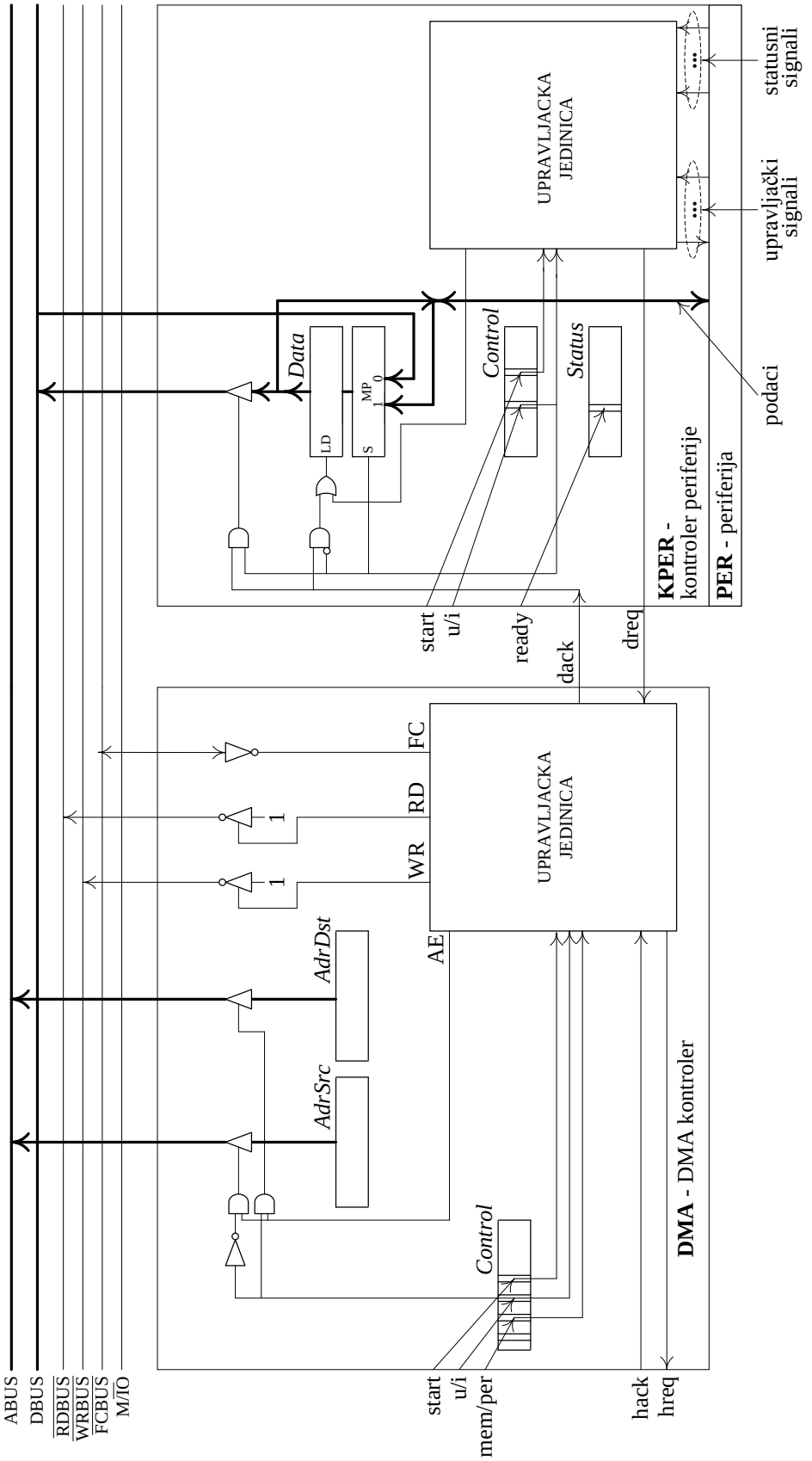
U oba slučaja se po završetku prenosa bloka podataka u bit *end* registra *Status* DMA kontrolera upisuje 1 i, ukoliko bit *enable* registra *Control* DMA kontrolera ima vrednost 1, generiše prekid **intr** DMA kontrolera. Završetak prenosa bloka podataka se utvrđuje programskim putem i to ili čitanjem registra *Status* DMA kontrolera i proverom vrednosti bita *end* ili generisanjem prekida **intr** DMA kontrolera. Po utvrđivanju završetka prenosa programskim putem se vrši zaustavljanje oba kontrolera tako što se u upravljačke registra *Control* DMA i nonDMA kontrolera upisuju vrednosti koje na poziciji bitova *start* imaju vrednost 0.

Programiranje ulaza/izlaza generisanjem prekida ilustrovano je na primeru ulazno/izlazne periferije sa koje se blok podataka unosi u memoriju (slika 15).

```
        Glavni program
        . . .
        MOV #blockadr, R1
        OUT R1, AdrDst
        MOV #blockcount, R1
        OUT R1, Count
        MOV #modestartdma, R1
        OUT R1, Controldma
        MOV #modestartper, R1
        OUT R1, Controlper
        MOV #1, sem
        ! Program u kome se ne koriste
        ! podaci iz bloka memorije u koji
        ! se unose podaci sa periferije
LOOP:   CMP sem, #0
        JNZ LOOP
        ! Program u kome se koriste
        ! podaci iz bloka memorije u koji
        ! su uneti podaci sa periferije
        . . .
        Prekidna rutina
        PUSH R1
        MOV #0, sem
        MOV #0, R1
        OUT R1, Controldma
        OUT R1, Controlper
BACK:   POP R1
        RTI
        . . .
```

Slika 15 Programirani ulaz/izlaz sa prekidom

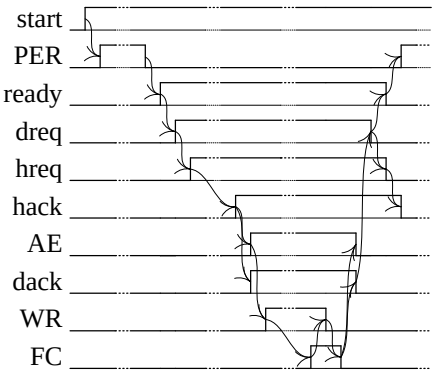
Delovi kontrolera sa direktnim pristupom memoriji i bez direktnog pristupa memoriji koji učestvuju u realizaciji čitanja iz memorije i upisu u memoriju su dati na slici 16.



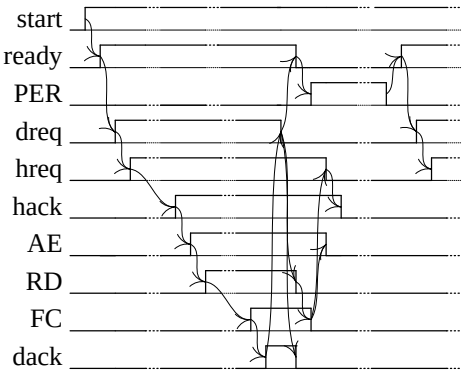
Slika 16 Čitanje iz memorije i upis u memoriju sa DMA kontrolerom i kontrolerom periferije

Vremenski oblici signala prilikom realizacije čitanja iz periferije i upisa u memoriju i čitanja iz memorije i upisa u periferiju u slučaju povezivanja periferije korišćenjem kontrolera

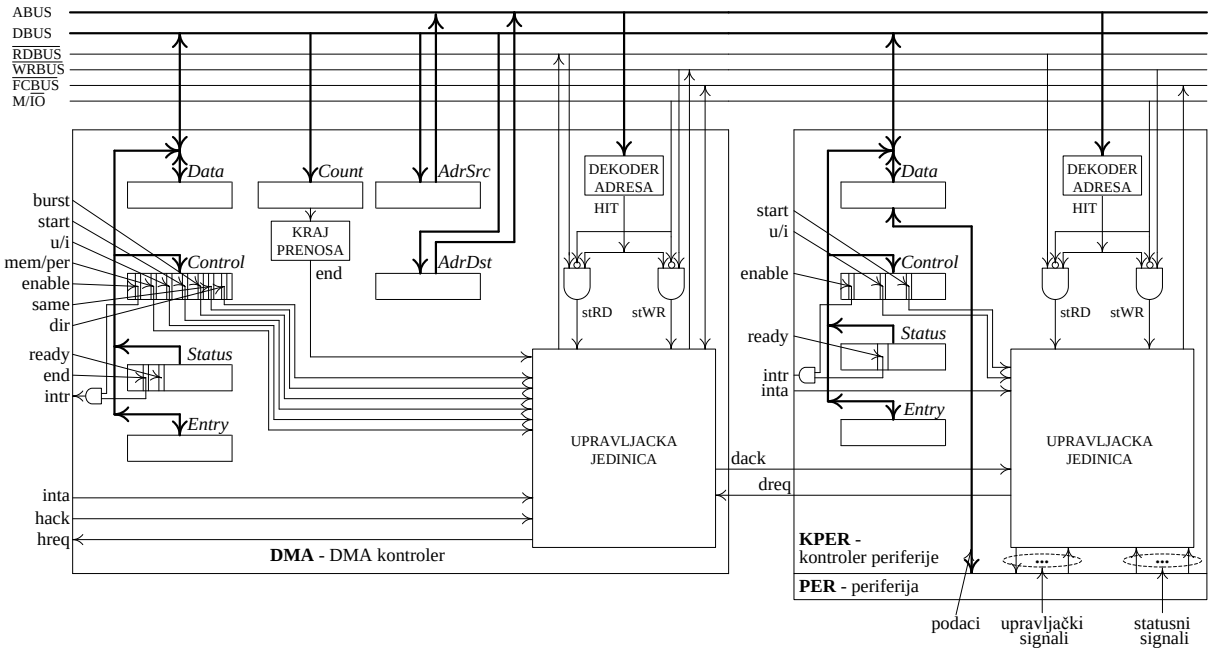
bez direktnog pristupa i kontrolera sa direktnim pristupom memoriji su dati na slikama 17 i 18, respektivno.

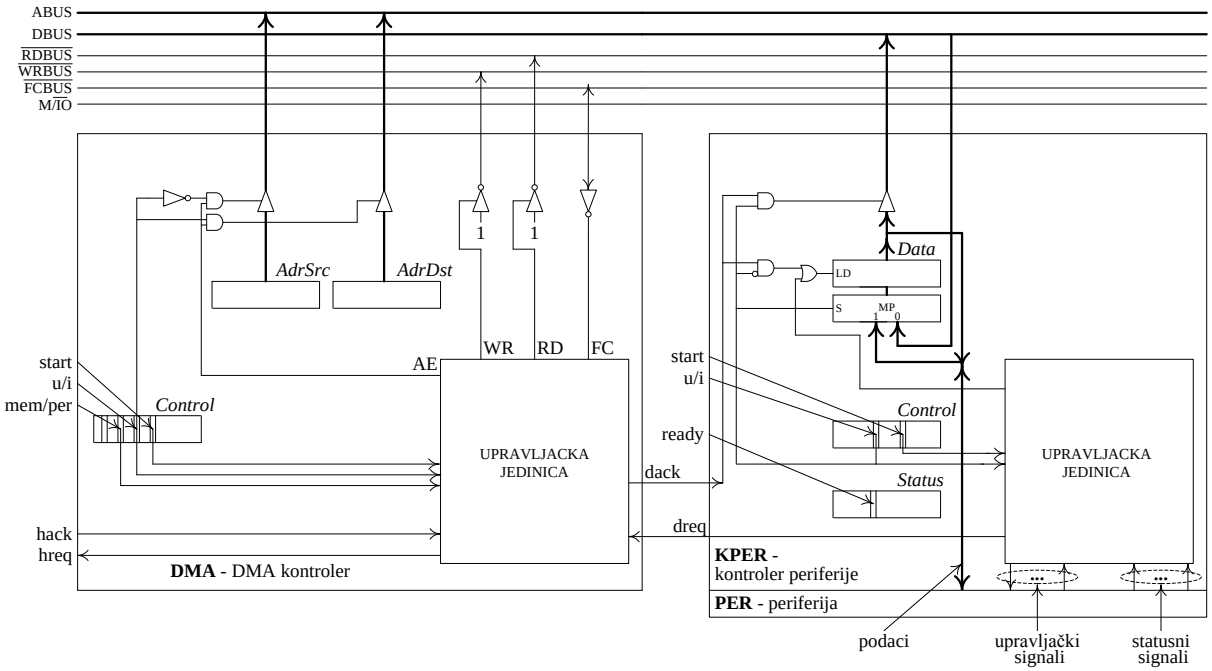


Slika 17 Vremenski oblici signala za čitanje iz periferije i upis u memoriju



Slika 18 Vremenski oblici signala za čitanje iz memorije i upis u periferiju


ABUS
DBUS
RDBUS
WRBUS
FCBUS
M/IO
Data
Count
AdrSrc
AdrDst
DEKODER ADRESA
HIT
KRAJ PRENOSA
end
burst
start
u/i
mem/per
enable
same
dir
Control
ready
end
intr
Status
Entry
stRD
stWR
UPRAVLJACKA JEDINICA
inta
hack
hreq
dack
dreq
DMA - DMA kontroler
KPER - kontroler periferije
PER - periferija
podaci
upravljački signali
statusni signali


ABUS
DBUS
RDBUS
WRBUS
FCBUS
M/IO
1
1
AdrSrc
AdrDst
WR
RD
FC
AE
start
u/i
mem/per
Control
UPRAVLJACKA JEDINICA
hack
hreq
dack
dreq
DMA - DMA kontroler
Data
LD
S
MP
ready
Status
KPER - kontroler periferije
PER - periferija
podaci
upravljački signali
statusni signali

1.4 ULAZNO/IZLAZNI PROCESORI

U ovom odeljku se razmatra organizacija ulaza/izlaza korišćenjem ulazno/izlaznih procesora.

1.5 MASKIRAJUĆI PREKIDI

Mehanizam prekida uključuje određene aktivnosti u preocesoru koje se mogu grupisati na opsluživanje prekida i povratak iz prekidne rutine. U okviru opsluživanja prekida čuva se kontekst procesora i prelazi na prekidnu rutinu. U okviru povratka iz prekidne rutine restaurira se kontekst procesora i vraća na prekinuti program. Opsluživanje prekida se delimično realizuje hardverski u okviru faze opsluživanje prekida instrukcije iz koje se prelazi na prekidnu rutinu i delimično softverski instrukcijama na početku prekidne rutine. Povratak iz prekidne rutine se realizuje softverski instrukcijama na kraju prekidne rutine.

Čuvanje konteksta procesora i povratak iz prekidne rutine se za određeni procesor realizuju na isti način bez obzira na vrstu prekida. Prelazak na prekidnu rutinu se za prekid zbog instrukcije prekida (PRINS), unutrašnje prekide zbog greške u kodu operacije (PRCOD) i grešeke u adresiranju (PRADR), spoljašnji nemaskirajući prekid (PRINM) i unutrašnji prekid zbog zadatog režima rada prekid posle svake instrukcije (PSWT) realizuje na isti način hardverski tehnikom vektorisanja prekida u okviru faze opsluživanje prekida instrukcije iz koje se prelazi na prekidnu rutinu uz korišćenje IV (Interrupt Vector) tabele (IV tabela) sa adresama prekidnih rutina i broja ulaza BRU (Broj Ulaza) u IV tabelu. Pri tome je broj ulaza za PRINS promenljiv i određen adresnim delom instrukcije prekida, dok je za PRCOD, PRADR, PRINM i PSWT fiksan i generisan unutar procesora u skladu sa tim koji su ulazi u IV tabeli predviđeni prilikom projektovanja procesora za ove prekide. Prelazak na prekidnu rutinu se za spoljašnje maskirajuće prekide realizuje ili kompletno hardverski, tehnikom vektorisanja prekida kojom se iz faze opsluživanje prekida instrukcije direktno prelazi na prekidnu rutinu, ili kombinovano hardverski, tehnikom vektorisanja prekida kojom se iz faze opsluživanje prekida instrukcije najpre prelazi na zajedničku prekidnu rutinu, i softverski, tehnikom poliranja kojom se iz zajedničke prekidne rutine prelazi na prekidnu rutinu. Pri tome realizacija prelaska na prekidnu rutinu za spoljašnje maskirajuće prekide zavisi najpre od toga da li postoji jedna ili više linija za slanje zahteva za prekid, zatim da li je realizovano slanje potvrde za prekid ili nije, potom, ukoliko je realizovano slanje potvrde za prekid, da li postoji jedna ili više linija za slanje potvrde za prekid i na kraju da li je broj ulazno/izlaznih uređaja koji šalju maskirajuće zahteve za prekid manji ili veći od broja linija po kojima mogu da se šalju zahtevi za prekid.

U ovom odeljku se razmatraju tehnike prelaska na prekidnu rutinu za maskirajuće prekide i to kompletno hardverski vektorisanjem prekida i kombinovano hardverski i softverski vektorisanjem i poliranje prekida.

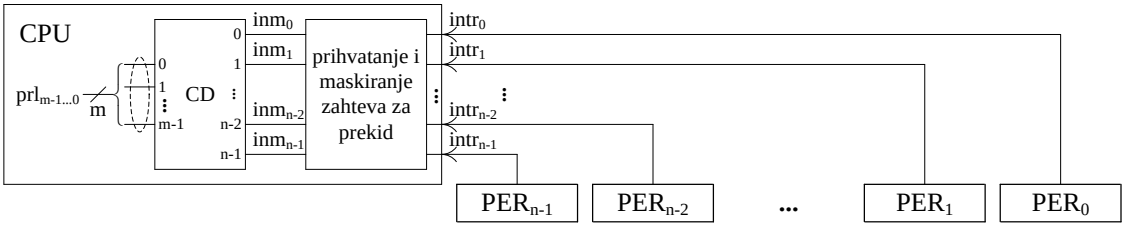
1.5.1 VEKTORISANJE PREKIDA

Ovom tehnikom se kompletno hardversko u okviru faze opsluživanje prekida instrukcije realizuje prelazak na prekidnu rutinu ulazno/izlaznog uređaja najvišeg prioriteta koji je generisao zahtev za prekid. Ovde se podrazumeva da se u okviru inicijalizacije sistema formira IV tabela, u kojoj za svaki od uređaja postoji poseban ulaz sa adresom prekidne rutine, i da se početna adresa IV tabele, koja može da se smesti u bilo koji deo memorije, upiše u poseban programski dostupan registar procesora IVTP (Interrupt Vector Table Pointer). Prelazak na prekidnu rutinu se realizuje tako što se za ulazno/izlazni uređaj najvišeg prioriteta koji je generisao zahtev za prekid najpre na odgovarajući način utvrđuje broj ulaza, zatim se broj ulaza pretvara u pomeraj, potom se sabiranjem pomeraja i sadržaja registra IVTP dobija adresa ulaza u IV tabeli iz koga se na kraju čita adresa prekidne rutine i upisuje u PC. Pri tome se broj ulaza u tabelu sa adresama prekidnih rutina može utvrditi na četiri načina u zavisnosti od toga da li po jednoj ili više linija dolaze zahtevi za prekid od ulazno/izlaznih

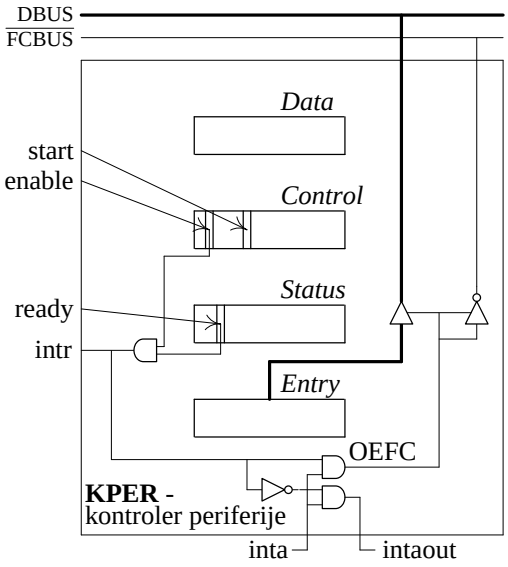
uređaja, da li ne postoje ili postoje linije za slanje signala potvrda i ukoliko postoje da li postoji jedna ili više linija.

Zahteve za prekid generišu kontroleri periferija ulazno/izlaznih uređaja. Kontroler periferije KPER koji nema direktan pristup memoriji generiše zahtev za prekid prilikom prenosa svakog podatka onda kada se u bit ready statusnog registra *Status* upiše vrednost 1, dok kontroler periferije sa direktnom pristupom memoriji DMAPER generiše zahtev za prekid na kraju prenosa bloka podataka onda kada se u bit end statusnog registra *Status* upiše vrednost 1. Pri tome je u oba slučaja potrebno da je generisanje zahteva za prekid dozvoljeno tako što je prilikom starovanja kontrolera u bit enable upravljačkog registra Control upisana vrednost 1. Zahtev za prekid se generiše kao impuls određenog trajanja ili drži kao nivo. Zahtevi za prekid koji dolaze kao impuls po linijama $intr_{n-1}$ do $intr_0$ se pamte u posebnim flip-flopovima $PRIRR_{n-1}$ do $PRIRR_0$ procesora koji se pojedinačno brišu po prihvatanju određenog zahteva za prekid i prelaska na prekidnu rutinu. Zahtevi za prekid koji se drži kao nivo na linijama $intr_{n-1}$ do $intr_0$ nema potrebe da se pamte, ali se pojedinačno ukidaju po prihvatanju određenog zahteva za prekid i prelaska na prekidnu rutinu.

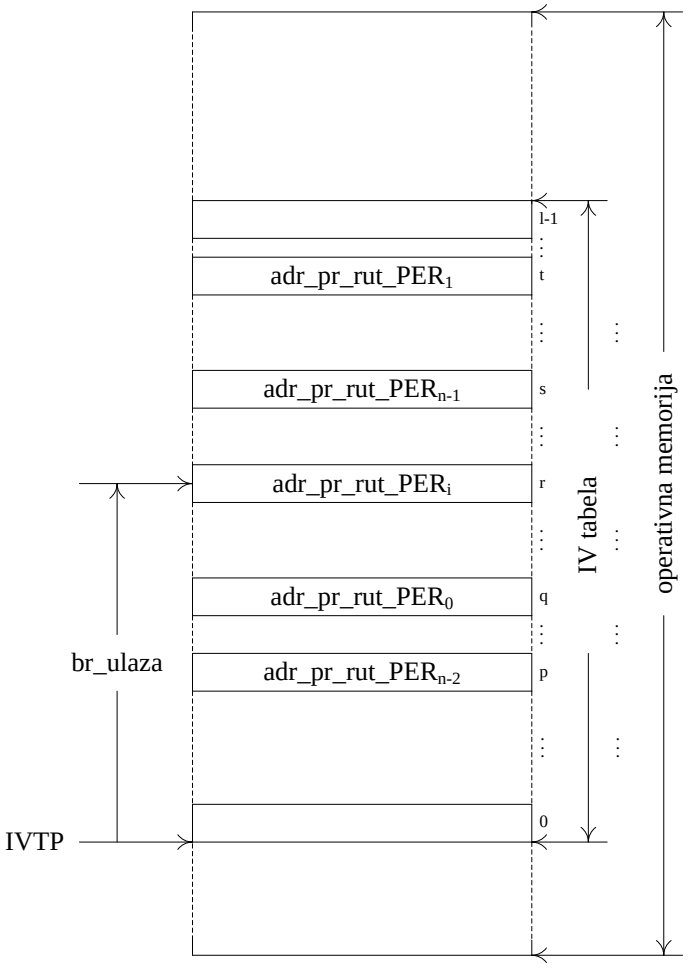
Na slici je prikazan slučaj kada zahtevi za prekid dolaze po linijama $intr_{n-1}$ do $intr_0$, pri čemu se uzima da je $n=2^m$. U ovom slučaju se za zahteve za prekid koji dolaze po linijama $intr_{n-1}$ do $intr_0$ ulazi u IV tabelu za adrese prekidnih rutina fiksno definišu prilikom projektovanja procesora. Pri tome se dodeljivanje ulaza u IV tabelu može realizovati na dva načina.



Slika 19 Zahtevi za prekid dolaze po više linija



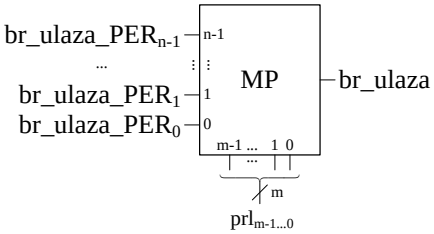
Slika 20 Generisanje zahteva za prekid



Slika 21 Operativna memorija sa IV tabelom

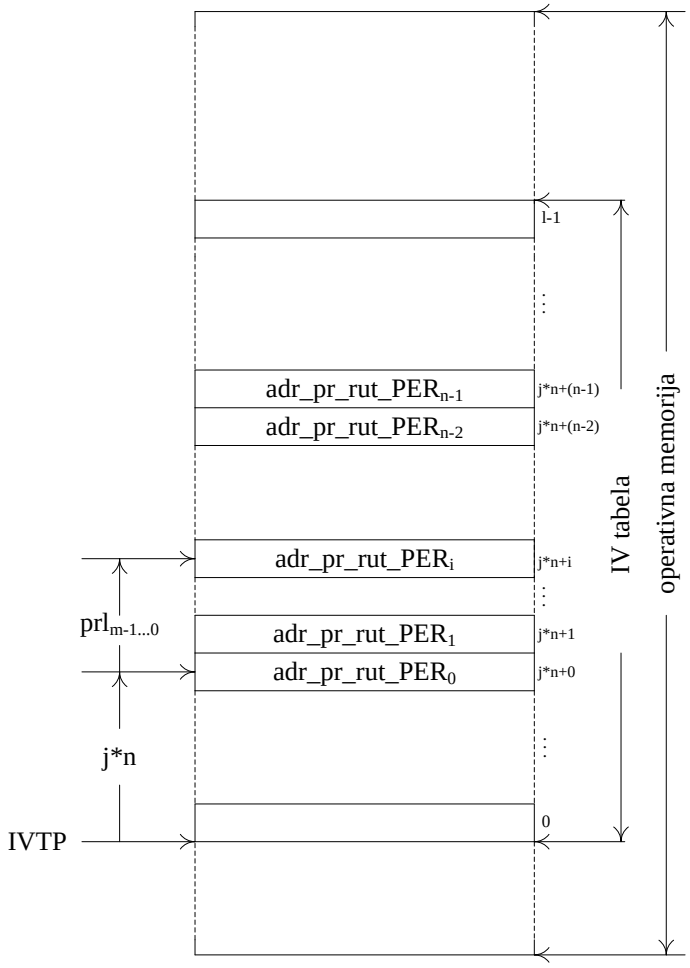
Prvi način dodeljivanja ulaza u IV tabelu je prikazan na slici na kojoj je dat deo operativne memorije sa IV tabelom. Uzeto je da IV tabela ima l ulaza pri čemu je $l=2^k$. Na deo operativne memorije sa IV tabelom ukazuje registar IVTP. Adrese prekidnih rutina za zahteve za prekid koji dolaze po linijama $intr_{n-1}$ do $intr_0$ se nalaze u ulazima IV tabele koji su proizvoljno dodeljeni. Tako je za adresu $adr\_pr\_rut\_PER_0$ prekidne rutine uređaja PER_0 koji šalje zahtev za prekid po liniji $intr_0$ dodeljen ulaz q, za adresu $adr\_pr\_rut\_PER_1$ prekidne rutine uređaja PER1 koji šalje zahtev za prekid po liniji $intr_1$ dodeljen ulaz t i tako redom do adrese $adr\_pr\_rut\_PER_{n-1}$ prekidne rutine uređaja PER_{n-1} koji šalje zahtev za prekid po liniji $intr_{n-1}$ kojoj je dodeljen ulaz s.

Broj ulaza u IV tabelu se generiše hardverski u fazi opsluživanje prekida u skladu sa tim kako su ulazi u IV tabelu dodeljeni zahtevima za prekid koji dolaze po linijama $intr_{n-1}$ do $intr_0$. Zahtevi za prekid mogu da dođu istovremeno po većem broju linija $intr_{n-1}$ do $intr_0$, a treba da se prihvati zahtev za prekid koji je navišeg prioriteta. Zbog toga za ove linije mora da budu definisani prioriteti i da se u procesoru generiše binarna vrednost $prl_{m-1...0}$ linije $intr_{n-1}$ do $intr_0$ najvišeg prioriteta na kojoj postoji zahtev za prekid a koji nije selektivno maskiran odgovarajućim razredom registra maske (slika). Binarna vrednost $prl_{m-1...0}$ se formira na izlazima kodera prioriteta CD na osnovu signala inm_{n-1} do inm_0 koji predstavljaju signale zahteva za prekid sa linija $intr_{n-1}$ do $intr_0$ upamćene u flip-flopovima $PRIRR_{n-1}$ do $PRIRR_0$ i selektivno maskirane razredima IMR_{n-1} do IMR_0 registra maske. Sama binarna vrednost broja ulaza br_ulaza se dobija sa izlaza multipleksera MP na koje se signalima $prl_{m-1...0}$ selektuje jedna od binarnih vrednosti brojeva ulaza $br\_ulaza\_PER_0$ do $br\_ulaza\_PER_{n-1}$ sa ulaza (slika).



Slika 22 Generisanje broja ulaza

Drugi način dodeljivanja ulaza u IV tabelu je prikazan na slici na kojoj je dat deo operativne memorije sa IV tabelom. Uzeto je da IV tabela ima l ulaza pri čemu je $l=2^k$. Na deo operativne memorije sa IV tabelom ukazuje registar IVTP. Adrese prekidnih rutina za zahteve za prekid koji dolaze po linijama $intr_{n-1}$ do $intr_0$ se nalaze u n susednih ulaza IV tabele počev od ulaza j·n, gde je j=0,..., l/n-1. Tako je za adresu $adr\_pr\_rut\_PER_0$ prekidne rutine uređaja PER_0 koji šalje zahtev za prekid po liniji $intr_0$ dodeljen ulaz j·n+0, za adresu $adr\_pr\_rut\_PER_1$ prekidne rutine uređaja PER1 koji šalje zahtev za prekid po liniji $intr_1$ dodeljen ulaz j·n+1 i tako redom do adrese $adr\_pr\_rut\_PER_{n-1}$ prekidne rutine uređaja PER_{n-1} koji šalje zahtev za prekid po liniji $intr_{n-1}$ kojoj je dodeljen ulaz j·n+(n-1).

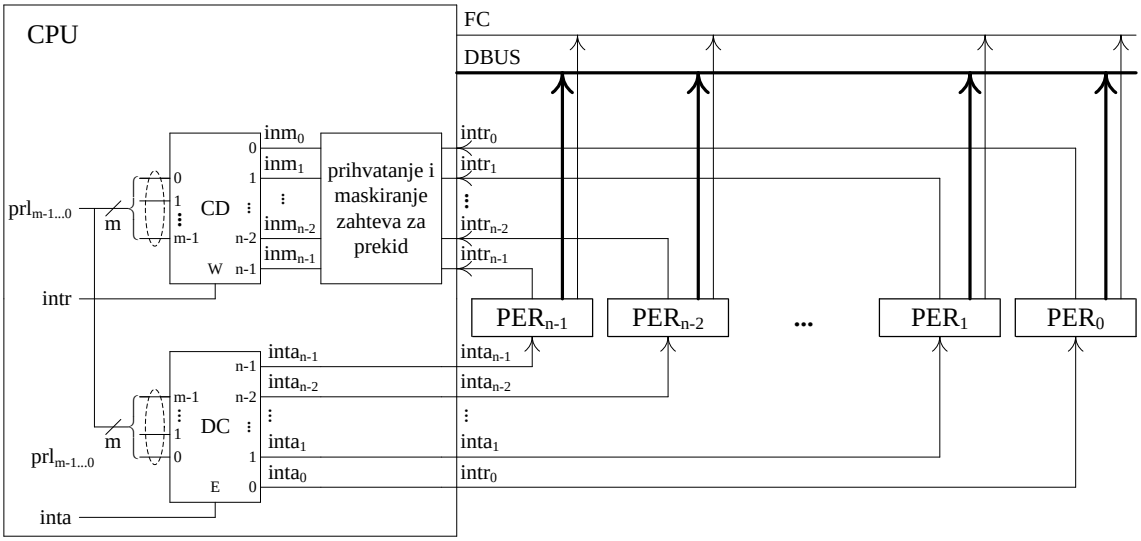


Slika 23 Operativna memorija sa IV tabelom

Broj ulaza u IV tabelu se generiše hardverski u fazi opsluživanje prekida na osnovu broja linije $intr_{n-1}$ do $intr_0$ najvišeg nivoa prioriteta po kojoj dolazi zahtev za prekid koji nije selektivno maskiran odgovarajućim razredom registra maske. Zahtevi za prekid mogu da dođu istovremeno po većem broju linija $intr_{n-1}$ do $intr_0$, a treba da se prihvati zahtev za prekid koji je navišeg prioriteta. Zbog toga za ove linije mora da budu definisani prioriteti i da se u

procesoru generiše binarna vrednost $prl_{m-1...0}$ linije $intr_{n-1}$ do $intr_0$ najvišeg prioriteta na kojoj postoji zahtev za prekid a koji nije selektivno maskiran odgovarajućim razredom registra maske (slika). Binarna vrednost $prl_{m-1...0}$ se formira na izlazima kodera prioriteta CD na osnovu signala inm_{n-1} do inm_0 koji predstavljaju signale zahteva za prekid sa linija upamćene u flip-flopovima i selektivno maskirane razredima IMR do IMR registra maske. Sama binarna vrednost broja ulaza se dobija sabiranjem j·n, gde je j=0,..., l/n-1, i $prl_{m-1...0}$.

Na slici je prikazan slučaj kada zahtevi za prekid dolaze po linijama $intr_{n-1}$ do $intr_0$, pri čemu se uzima da je $n=2^m$, ali i kada postoje linije $inta_{n-1}$ do $inta_0$ po kojima se šalju potvrde prihvatanja zahteva za prekid. U ovom slučaju se za zahteve za prekid koji dolaze po linijama $intr_{n-1}$ do $intr_0$ ulazi u IV tabelu za adrese prekidnih rutina proizvoljno dodeljuju prilikom inicijalizacije sistema.



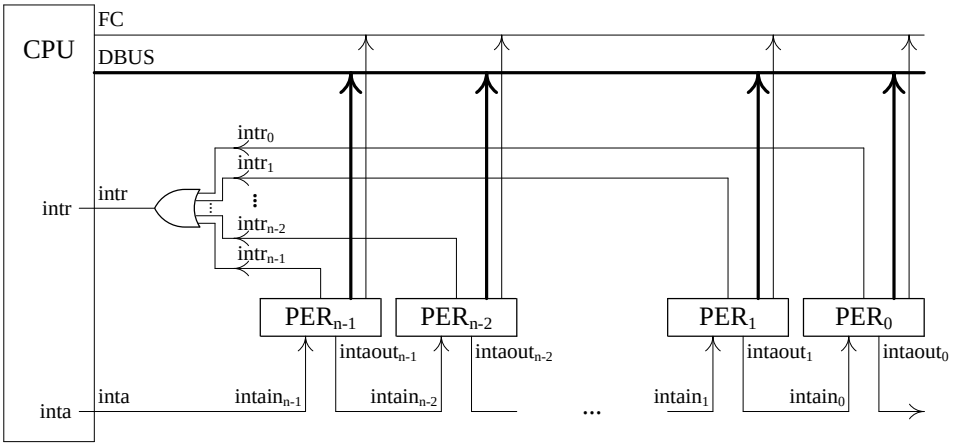
Slika 24 Paralelno slanje signala zahteva i potvrda

Dodeljivanje ulaza u IV tabelu je prikazano na slici na kojoj je dat deo operativne memorije sa IV tabelom. Uzeto je da IV tabela ima l ulaza pri čemu je $l=2^k$. Na deo operativne memorije sa IV tabelom ukazuje registar IVTP. Adrese prekidnih rutina za zahteve za prekid koji dolaze po linijama $intr_{n-1}$ do $intr_0$ se nalaze u ulazima IV tabele koji su proizvoljno dodeljeni. Tako je za adresu adr_pr_rut_PER_0 prekidne rutine uređaja PER_0 koji šalje zahtev za prekid po liniji $intr_0$ dodeljen ulaz q, za adresu adr_pr_rut_PER_1 prekidne rutine uređaja PER1 koji šalje zahtev za prekid po liniji $intr_1$ dodeljen ulaz t i tako redom do adrese adr_pr_rut_PER_{n-1} prekidne rutine uređaja PER_{n-1} koji šalje zahtev za prekid po liniji $intr_{n-1}$ kojoj je dodeljen ulaz s. Prilikom inicijalizacije sistema se adrese prekidnih rutina upisuju u ulaze koji su dodeljeni njima dodeljeni, a brojevi ulaza upisuju u registre Entry kontrolera.

Broj ulaza u IV tabelu šalje onaj uređaj koji u fazi opsluživanje prekida po prijemu u skladu sa tim kako su ulazi u IV tabelu dodeljeni zahtevima za prekid koji dolaze po linijama $intr_{n-1}$ do $intr_0$. Zahtevi za prekid mogu da dođu istovremeno po većem broju linija $intr_{n-1}$ do $intr_0$, a treba da se prihvati zahtev za prekid koji je navišeg prioriteta. Zbog toga za ove linije mora da budu definisani prioriteti i da se u procesoru generiše binarna vrednost $prl_{m-1...0}$ linije $intr_{n-1}$ do $intr_0$ najvišeg prioriteta na kojoj postoji zahtev za prekid a koji nije selektivno maskiran odgovarajućim razredom registra maske (slika). Binarna vrednost $prl_{m-1...0}$ se formira na izlazima kodera prioriteta CD na osnovu signala inm_{n-1} do inm_0 koji predstavljaju signale zahteva za prekid sa linija upamćene u flip-flopovima i selektivno maskirane razredima IMR do IMR registra maske. Pri vrednosti 1 signala inta se na osnovu binarne vrednosti $prl_{m-1...0}$ na jednom od izlaza dekodera DC $inta_{n-1}$ do $inta_0$ formira vrednost 1. Time se daje dozvola kontroleru ulazno/izlaznog uređaja najvišeg prioriteta koji je poslao zahtev za prekid a koji nije selektivno maskiran odgovarajućim razredom registra maske da pošalje procesoru po linijama podataka magistrale sadržaj svog registra Entry koji predstavlja broj ulaza.

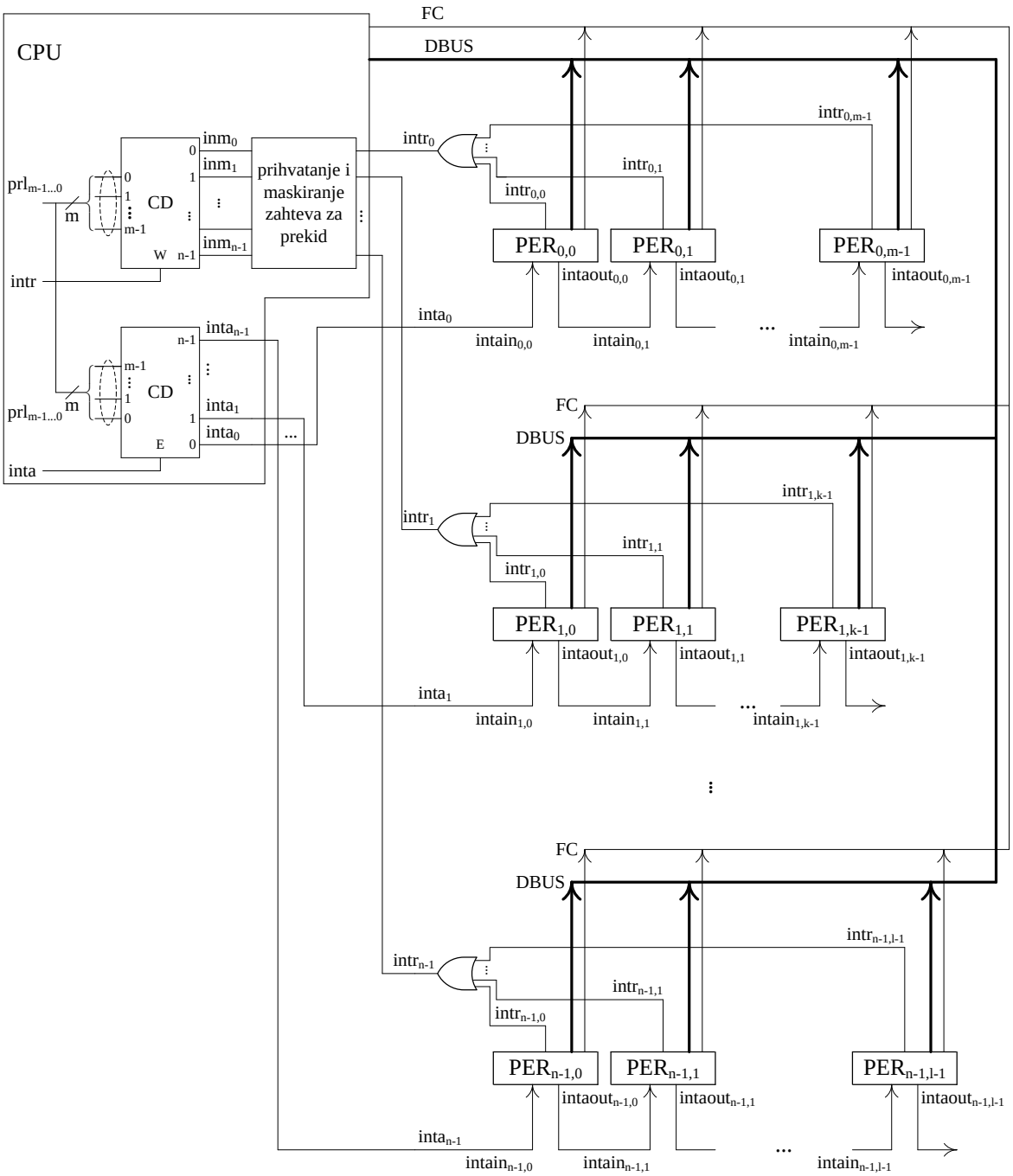
Na slici je prikazan slučaj kada u procesoru postoji samo jedna linija intr po kojoj može da stigne zahtev za prekid i jedan linija inta po kojoj se šalje dozvola. Signal intr se formira kao unija signala koji dolaze po linijama $intr_{n-1}$ do $intr_0$, dok se dozvola inta preosleđuje serijski u lancu od uređaja najvišeg prioriteta PER_{n-1} do uređaja najnižeg prioriteta PER_0. U uređaju koji dobije vrednost 1 signala inta mogu da nastanu dve situacije. Prva situacija je da njegov signal intr ima vrednost 1. Tada dati uređaj otvara bafere sa tri stanja i na linije podatak DBUS pušta sadržaj svog registra Entry i upravljačku liniju FCBUS na vrednost 0. Pored toga uređaj postavlja liniju intaout na vrednost 0, čime će na linijama inta i intaout svih uređaja u lancu da bude vrednost 0. Druga situacija je da njegov signal intr ima vrednost 0, pa dati uređaj postavlja liniju intaout na vrednost 1, čime dozvolu prosleđuje sledećem uređaju u lancu.

U ovom slučaju se za zahteve za prekid koji dolaze po linijama $intr_{n-1}$ do $intr_0$ ulazi u IV tabelu za adrese prekidnih rutina dodeljuju na isti način kao u prethodno razmatranom slučaju kada postoje posebne linije za $intr_{n-1}$ do $intr_0$ i $inta_{n-1}$ do $inta_0$. Razlika između ova dva slučaja je samo u tome da se potvrde inta na šalju paralelno već serijski.



Slika 25 Serijsko slanje signala zahteva i potvrda

Na slici je prikazan slučaj kada kao u slučaju 2 u procesoru postoje posebne linije za $intr_{n-1}$ do $intr_0$ i $inta_{n-1}$ do $inta_0$. Međutim, zbog toga što je broj uređaja veći od broja linija po kojima mogu da se šalju zahtevi za prekid i dobijaju dozvole, u ovom slučaju se kombinuju pristupi za slučajeve 2 i 3. Uzeto je da kao u slučaju 3 uređaji $PER_{0,0}$ do $PER_{0,m-1}$ šalju zahteve za prekid po liniji $intr_0$ i da se dozvola $inta_0$ serijski prosleđuje od uređaja $PER_{0,0}$ do $PER_{0,m-1}$. Na sličan način i preostali uređaji po presotalim linijama šalju zahteve za prekid i dobijaju dozvole. U procesoru se kao u slučaju 2 u zavisnosti od toga na kojoj liniji $intr_{n-1}$ do $intr_0$ postoji zahtev dozvola šalje po jednoj liniji $inta_{n-1}$ do $inta_0$. i $inta_{n-1}$ do $inta_0$. U ovom slučaju se za zahteve za prekid koji dolaze po linijama $intr_{n-1}$ do $intr_0$ ulazi u IV tabelu za adrese prekidnih rutina dodeljuju na isti način kao u dva prethodno razmatrana slučaja.

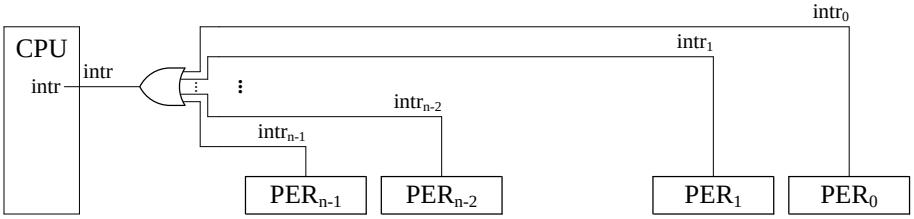


Slika 26 Vektorisanje prekida sa promenljivim ulazima i serijsko prosleđivanje potvrda

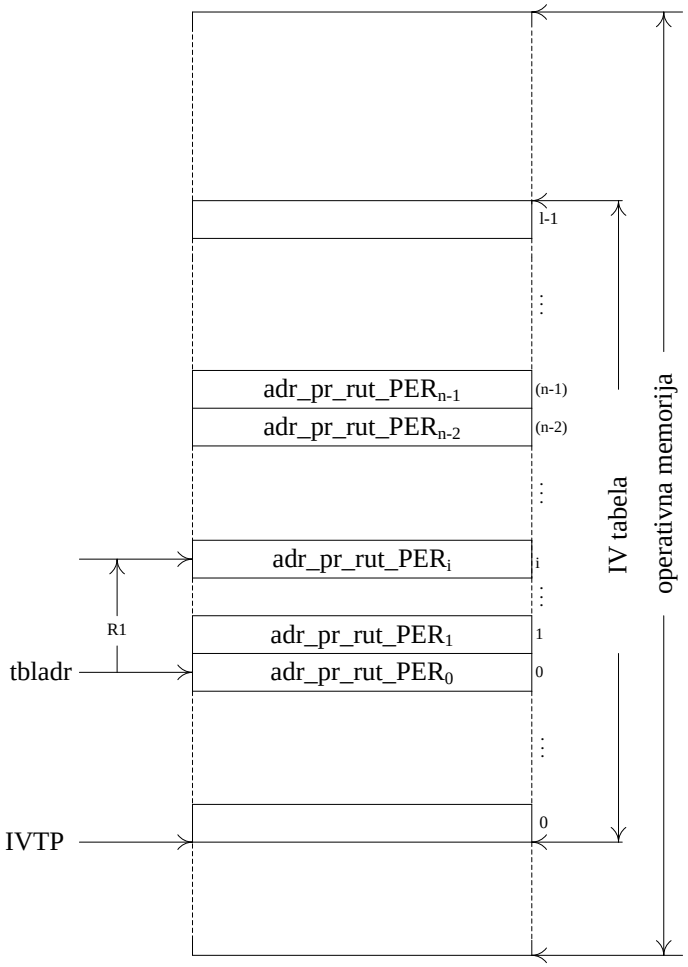
1.5.2 VEKTORISANJE I POLIRANJE PREKIDA

Ova tehnika podrazumeva kombinovano hardversko i softversko utvrđivanje adrese prekidne rutine.

Prva situacija je da postoji jedna linija po kojoj se šalje zahtev za prekid.



Slika 27 Poliranje zahteva



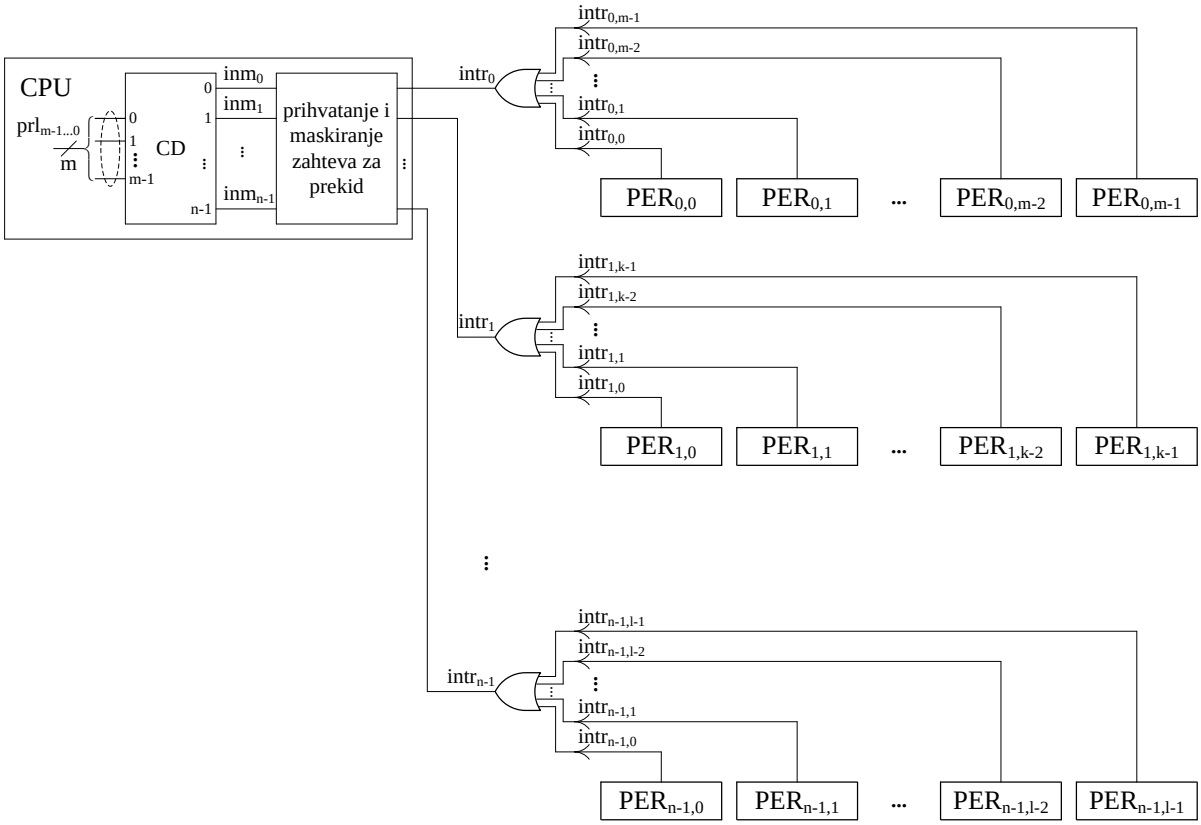
Zahtev za prekid dolazi po liniji intr. U fazi opsluživanje prekida se vektorisanjem prelazi na zajedničku prekidnu rutinu u kojoj se poliranjem po opadajućim prioritetima programskim putem prelazi na prekidnu rutinu uređaja najvišeg nivoa prioriteta koji je uputio zahtev za prekid. U registru R1 je broja ulaz uređaja najvišeg nivoa prioriteta, a u lokaciji tbladr početna adresa dela IV tabele sa adresam prekidnih rutina.

Zajednička prekidna rutina

```
                PUSH R1
                PUSH R2
! Poliranje periferije PER(n-1)
                MOV #(n-1), R1          ! broj ulaza za PER(n-1) u R1
    PER(n-1):   IN R2, ControlPER(n-1)  ! Control za PER(n-1) u R2
                TST R2, #startPER(n-1)  ! provera bita start u Control za PER(n-1) i
                JZ PER(n-2)             ! prelazak na PER(n-2) ukoliko start 0
                TST R2, #enablePER(n-1) ! provera bita enable u Control za PER(n-1)
                JZ PER(n-2)             ! prelazak na PER(n-2) ukoliko enable 0
                IN R2, StatusPER(n-1)   ! Status za PER(n-1) u R2
                TST R2, #readyPER(n-1)  ! provera bita ready u Status za PER(n-1) i
                JZ PER(n-2)             ! prelazak na PER(n-2) ukoliko ready 0
                JMP TBL                 ! prelazak na zajednički prelazak na prekidnu rutinu
! Poliranje periferije PER(n-2)
    PER(n-2):   DEC R1                  ! broj ulaza za PER(n-2) u R1
                IN R2, ControlPER(n-2)  ! Control za PER(n-2) u R2
                TST R2, #startPER(n-2)  ! provera bita start u Control za PER(n-2) i
                JZ PER(n-3)             ! prelazak na PER(n-3) ukoliko start 0
                TST R2, #enablePER(n-2) ! provera bita enable u Control za PER(n-2)
                JZ PER(n-3)             ! prelazak na PER(n-3) ukoliko enable 0
                IN R2, StatusPER(n-2)   ! Status za PER(n-2) u R2
                TST R2, #readyPER(n-2)  ! provera bita ready u Status za PER(n-2) i
                JZ PER(n-3)             ! prelazak na PER(n-3) ukoliko ready 0
                JMP TBL                 ! prelazak na zajednički prelazak na prekidnu rutinu
! Poliranje periferije PER(n-3)
    PER(n-3):   DEC R1                  ! broj ulaza za PER(n-3) u R1
                IN R2, ControlPER(n-3)  ! Control za PER(n-3) u R2
                TST R2, #startPER(n-3)  ! provera bita start u Control za PER(n-3) i
                JZ PER(n-4)             ! prelazak na PER(n-4) ukoliko start 0
                ...
! Poliranje periferije PER1
    PER1:       DEC R1                  ! broj ulaza za PER1 u R1
                IN R2, ControlPER1      ! Control za PER1 u R2
                TST R2, #startPER1      ! provera bita start u Control za PER1 i
                JZ PER0                 ! prelazak na PER0 ukoliko start 0
                ...
! Poliranje periferije PER0
    PER0:       DEC R1                  ! broj ulaza za PER0 u R1
                IN R2, ControlPER0      ! Control za PER0 u R2
                TST R2, #startPER0      ! provera bita start u Control za PER0 i
                JZ BACK                 ! prelazak na BACK ukoliko start 0
                TST R2, #enablePER0     ! provera bita enable u Control za PER0
                JZ BACK                 ! prelazak na BACK ukoliko enable 0
                IN R2, StatusPER0       ! Status za PER0 u R2
                TST R2, #readyPER0      ! provera bita ready u Status za PER0 i
                JZ BACK                 ! prelazak na BACK ukoliko ready 0
! Prelazak na prekidnu rutinu periferije na osnovu broja ulaza iz R1 i početne adrese tabele
! sa adresama prekidnih rutina iz tbladr
    TBL:        SHL R1                  ! broj ulaza iz R1 u pomeraj u R1
                ADD R1, tbladr          ! adresa lokacije sa adresom prekidne rutine u R1
                MOV [R1], R2            ! adresa prekidne rutine u R2
                JSR [R2]                ! adresa prekidne rutine u PC
    BACK:       POP R2
                POP R1
                RTI
```

Slika 28 Utvrđivanje adrese prekidne rutine poliranjem

Druga situacija je da ima više linija po kojima može da se šalje zahtev za prekid.



Slika 29 Vektorisanje prekida sa fiksnim ulazima i poliranje

Ovo je kombinacija situacije 1 i prethodne situacije. Kao u situaciji 1 skače se na jednu od n zajedničkih rutina. U svakoj zajedničkoj prekidnoj rutini se kao u prethodnom slučaju poliranjem prelazi na odgovarajuću prekidnu rutinu.

2