JOVAN ĐORĐEVIĆ

ARHITEKTURA I ORGANIZACIJA RAČUNARA

ULAZ/IZLAZ

BEOGRAD, 2006.

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.

Sistemi.

Autor

Beograd

avgusta 2005.

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 registat 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.

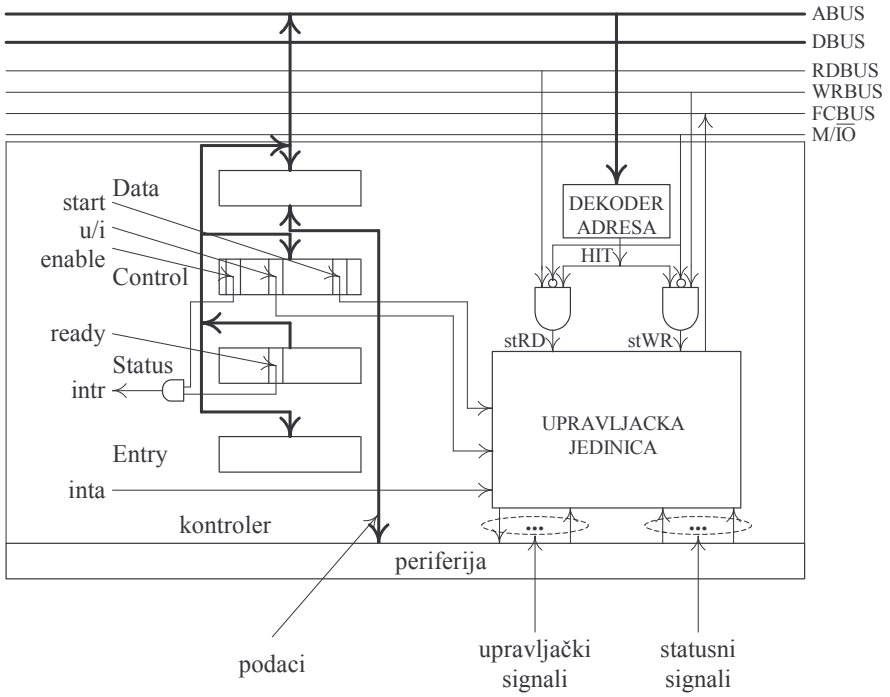
Podaci se obično unose sa periferije blokovski, tako što se u neki deo memorije u niz susednih lokacije prenosi određen broj reči. Isto važi i za slanje podataka iz memorije u izlazne periferije. Za realizaciju blokovskog unosa podataka iz periferije u memoriju i slanja podataka iz memorije u periferiju 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 KONTROLERI BEZ DIREKTNOG PRISTUPA MEMORIJI

U ovom odeljku se razmatraju organizacija kontrolera, programiranje kontrolera i povezivanje sa periferijom.

1.2.1 ORGANIZACIJA

U ovom odeljku se razmatra organizacija kontrolera bez direktnog pristupa memoriji (slika 1). Kontroler se sastoji od operacione jedinice i upravljačke jedinice. Operacionu jedinicu čine registri Data, Control, Status i Entry i kombinaciona mreža 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 bez direktnog pristupa memoriji

Osnovni zadatak kontrolera periferije 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 periferije 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 program kojim će se datak podatak prebaciti iz registra Data u odgovarajuću memorijsku lokaciju. U slučaju izlaza procesor mora da ima informaciju da je prethodni 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 ulazuje na dve situacije koje mogu da se jave u slučaju ulaza. Prva je da je prethodni podatak od strane procesora prenet iz registra Data u memorijsku lokaciju, da je prenos sledećeg podatka iz periferiju u registar Data od strane kontrolera u toku i da procesor ne sme da pređe na izvršavanje programa kojim će se preneti sledeći podatak iz registra Data u memorijsku lokaciju. Druga je da je sledeći podatak prenet od strane kontrolera iz periferije u registar Data ali ne i u memorijsku lokaciju, pa procesor može da pređe na izvršavanje programa kojim će preneti taj podatak iz registra Data u memorijsku lokaciju. Na sličan način dvema vrednostima ovog bita kontroler ukazuje na dve situacije koje mogu da se jave u slučaju izlaza. Prva je da je prethodni podatak prenet od strane procesora iz memorijske lokacije u registar Data, da je prenos tog podatka iz registra Data u periferiju od strane kontrolera u toku i da procesor ne sme da pređe na izvravanje programa kojim će preneti sledeći podatak iz memorijske lokacije u registar Data. Druga je da je prethodni 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 aktivna vrednost signala ready ukazuje da je registar Data raspoloživ, a neaktivna da nije raspoloživ. Raspoloživost registra Data u slučaju ulaza znači da je sledeći 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 prethodni podatak od strane kontrolera prenet iz registra Data u periferiju i da procesor može da pređe na izvršavanje programa kojim će sledeći podatak biti prernet iz memorijske u registar Data. Ovakva sinhronizacija između procesora i kontrolera zahteva da procesor povremeno čita registar Status i vrši proveru da li je bit ready neaktivan ili aktivan. Ukoliko se utvrdi da je bit ready neaktivan, to znači da 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 je bit ready aktivan, 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 bloka 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 kontroleom. 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 starovanja kontrolera periferije, zatimi 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 aktivna vrednost bita Start određuje da je kontroler startovan, a neaktivna 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 aktivnu i neaktivnu vrednost, respektivno. Aktivna vrednost bita Start omogućuje da upravljačka logika kontrolera priferije prenosi podatke iz periferije u registar Data i obratno, dok neaktivna vrednost 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 razmatranjim se podrazumeva da aktivna vrednost bita u/i određuje režim ulaza, a neaktivna vrednost režim izlaza.

Provera raspoloživosti registra Data čitanjem registra Status i proverom da li je bit ready aktivan ili neaktivan je jedna od tehnika kojom se organizuje ulaz/izlaz. Ovo je moguće organizovati na dva načina. U prvom slučaju se u petlji čita registar Status i proverava bit ready sve vreme dok je bit ready neaktivan. Kada se utvrdi da je bit ready aktivan, izlazi se iz petlje, realizuje prenos jedne reči i vraća u petlju radi čekanja da registar Data bude ponovo raspoloživ. Nezgoda sa ovim načinom je u tome da procesor ne vrši nikakvu obradu već u petlji čeka. U drugom slučaju procesor vrši neku obradu, pa u okviru toga povremeno čita registar Status i proverava bit ready. Ukoliko se utvrdi da je bit ready neaktivan, produžava se sa obradom. Ukoliko se utvrdi da je bit ready aktivan, 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 aktivnu vrednost u bit ready registra Status. U ovom slučaju procesor može dok traje prenos podatka iz periferije u registar Data ili obratno da vrši neku obradu. Kada posle jednog prenetog podatka registar Data postane raspoloživ, kontroler, uz upisivanje aktivne vrednosti u bit ready, generiše i signal prekida intr. Procesor kao reakcija na signal prekida na trenutak prekida tekuću obradu, skače na odgovarajuću prekidnu rutinu u okviru koje prenosi podatak iz registra Data u memoriju ili obratno, pa se potom vraća na prekinutu tekuću obradu. U okviru registra Control postoji i bit enable u koji se programskim putem može upisivati aktivna ili neaktivna vrednost. Ukoliko je u bitu enable aktivna vrednost, tada se pri upisu aktivne vrednosti u bit ready registra Status, generiše signal prekida intr. U suprotnom slučaju nema generisanja signala prekida intr.Kod inicijalizacije i starovanja kontrolera u registar Control je potrebno upisati takvu vrednost koja će na poziciji bita enable imati aktivnu ili neaktivnu vrednost u zavisnosti od toga da li je programskim putem predviđeno da procesor utvrđuje da je registar Data raspoloživ čitanjem registra Status ili prijemom signala prekida.

Kombinaciona mreža za prepoznavanje ciklusa čitanja i upisa stalno proverava vrednost signala M/IOBUS i sadržaja na linijama ABUS. Ovde je pretpostavljeno da su ulazno/izlazni i memorijski prostor razdvojeni i da se aktivnom i neaktivnom vrednošću signala M/IOBUS određuje da se na linijama ABUS nalazi adresa iz memorijskog ili ulazno/izlaznog adresnog prostora, respektivno. Ukoliko se pri neaktivnoj vrednosti signala M/IOBUS na linijama ABUS pojavi adresa nekog od registara kontrolera, signal HIT na izlazu DEKODERA ADRESA će biti aktivan, pa će signal RDBUS ili WRBUS dati aktivnu vrednost signala RD ili WR.

Upravljačka jedinica se aktivira po prijemu aktivne vrednosti signala RD, WR, inta ili start. U slučaju aktivne vrednosti signala RD ili WR 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 aktivne vrednosti signala 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 FCBUS. U zavisnosti od toga da li je u registar Control upisana vrednost koja na poziciji bita start ima aktivnu ili neaktivnu vrednost, 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.

1.2.2 PROGRAMIRANJE

Programiranje ulaza/izlaza korišćenjem kontrolera bez direktnog pristupa 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 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 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 bit ready 1 ili 0, zadaje se kao neposredna veličina, označena sa #ready. 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 bez direktnog pristupa memoriji, programskim putem se ostvaruje prenos podatka iz registra podatka kontrolera 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 lokacij i obratno, rešava se na dva načina i to: programskim putem ispitivanjem bita ready statusnog registra kontrolera periferije (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 kontrolera ilustrovano je na primeru ulazno/izlazne periferije sa koje se blok podataka unosi u memoriju unosi (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 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 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 se vrednosti koje se razlikuju od periferije do periferije i nisu predmet ovih razmatranja. Potom se u petlji prenosi sadržaj statusnog registra kontrolera periferije u registar R3 i vrši provera, korišćenjem neposredna veličine označene sa #ready, koja na pozicija bita ready statusnog registra ima 1 a na ostalima 0, da li je bit ready statusnog registra 0 ili 1. Sve dok je bit ready 0, što znači da podatak nije raspoloživ u registru podatka, ostaje se u petlji. Prvi put kada se otkrije da je bit ready 1, šro

će se desiti da poddatak postane raspoloživ u registru podatka, 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 LOOPi prenos sledeće reči.

```
        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

Treba napomenuti da se prilikom prebacivanje podatka iz registra podatka kontrolera periferije u registar R3, podrazumeva da kontroler periferije hardverski u bit ready statusnog registra upisuje vrednost 0. Ovaj bit će ponovo postati 1 tek pošto sledeći podatak bude prenet iz periferije u registar podatka kontrolera periferije. Da to nije tako urađeno, onda bi se, u slučaju sporih periferija, dešavalo da se, povratkom na početak petlje preko labele LOOP, u statusnom registru nalazilo da je bit ready jedinica, ali ne zbog toga što je u registar podatka unet novi podatak sa periferije i bit ready postavljen na jeda, već zbog toga što je vrednost jedan 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 kontrolera. Spoljašnja petlja služi za prenos podatka iz registra podatka kontrolera u memorijsku lokaciju

i ažuriranje registara sa adresom i brojem podatak koje još treba preneti. U unutrašnjo petlji se vrti dok novi podatak ne bude raspoloživ u registru podatka kontrlera. U spoljašnjoj petlji se vrti dok se ne prenesu svi podaci bloka iz periferiju u memoriju.

Kada sadržaj registra R2 postane nula, izlazi se iz spoljašnje petlje i prelazi na zadnje tri instrukcije kojima se zaustavlja kontroler periferije. Upravljački registar kontrolera 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 kontrolera. S obzirom na to da vrednost koja se upisuje u upravljački registar kontrolera ima nulu na poziciji bita start, zaustavlja se kontroler priferije.

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 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. 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.

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 kontrolera 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 kontrolera umesto instrukcija

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

da se stavi

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

Nedostatak programiranog ulaza/izlaza čitanjem statusnog registra kontrolera 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 memoriju 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 kontrolera u memorijsku lokaciju, generisanje adrese memorijske lokacije u koju treba preneti sledeći podatak i vođenje evidencije o tome koliko ješ podataka boka 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 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 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 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 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 kontrolera skače se na prekidnu rutinu.

Na ulasku u prekidnu rutinu i izlasku iz prekidne rutine registar R1 se stavlja na stek i skida sa steka, respektivno, jer se u no koristi. U prekidnoj rutini se podatak iz registra podatka preko registra R1 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 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 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

```
        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 R1, [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 dalji prekida od strane kontrolera 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. Za svaki novi podatak prenet iz periferije u registar podatka kontrolera, 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 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 kontrolera u memorijsku lokaciju, potrebno staviti instrukcije

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

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

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 kontrolera 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 kontrolera umesto instrukcija

MOV [mem1], R3

MOV R3, Data

da se stavi

MOV [mem1], Data

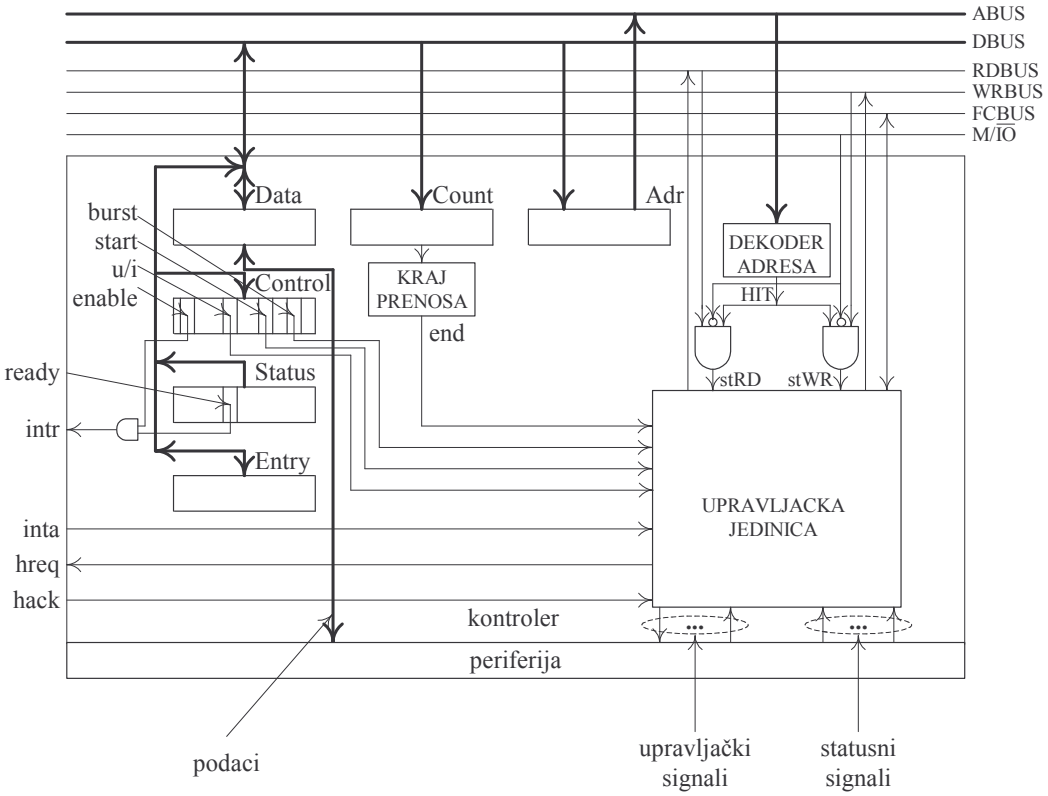
Prednost programiranog ulaza/izlaza generisanjem prekida u odnosu na programirani ulaz/izlaz čitanjem statusnog registra je da dok traje unošenje podataka iz periferije u memoriju procesor ne čeka u petlji, već izvršava program kome nisu potrebni podaci čije je uošenje u toku. Izvršavanje ovog programa se prekida, radi skoka na prekidnu rutinu, onoliko puta kolika je veličina bloka podataka. 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 periferije ili koji treba da se šalju u periferiju. U tom slučaju se moraju koristiti druga rešenja za ulaz/izlaz, kao što je kontroler sa direktnim pristupom memoriji.

1.3 KONTROLERI SA DIREKTNIM PRISTUPOM MEMORIJI

U ovom odeljku se razmatraju organizacija kontrolera 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 sa direktnim pristupom memoriji (slika 4). Kontroler se sastoji od operacione jedinice i upravljačke jedinice. Operacionu jedinicu čine registri Data, Control, Status, Entry, Adr i Count i kombinaciona mreža 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, Adr 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 upisivati i vođenje evidencije o broju podataka koje treba preneti između periferije i memorije i obratno.



Slika 4 Kontroler sa direktnim pristupom memoriji

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, Adr 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 ready 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 5). 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 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, 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 reistra 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, Adr
        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 5 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 ready. 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 ready satusnog registra.

Programirani izlaz generisanjem prekida se realizuje programom koji je veoma sličan programu za programirani ulaz generisanjem prekida (slika 3). Razlika je 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, 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, Adr

da se stavi

MOV #blockadr, Adr

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.

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 koriščenje 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.

Režim penosa memorija-memorija. Zbog toga postoje i dodatni adresni registar i dodatni bit mem u upravljačkom registru. Sada postoje izvorišni i odredišni adresni registri. Jedan adresni registar se koristi za čitanje iz memorije i prebacivanje u registar podatka i drugi za upis iz registra podatka u memorijsku lokaciju. Na magistrali se realizuju dva posebna ciklusa i to ciklus čitanja i ciklus upisa.

Režim slanja iste vrednosti. Tada se izvorišni adresni registar koristi samo za jedno čitanje i upis u data registar, a posle se isti sadržaj data registra upisuje u memoriju počev od adrese koja je data odredišnim adresnim registrom i to onoliko puta k olika je vrednost Count registra.

Režim prenosa prema višim i nižim memorijskim lokacijama. Postoji poseban bit direction u upravljačkom registru, pa se dvema vrednostima ovog bita određuje da li sadržaj adresnog registra treba inkrementirati ili dekrementirati. Ovo je moguće i za prenose između periferije i memorije i za prenose iz memorije u memoriju.