**Ј. ЂОРЂЕВИЋ, З. РАДИВОЈЕВИЋ, М. ПУНТ,
Б. НИКОЛИЋ, Д. МИЛИЋЕВ, Ј. ПРОТИЋ,
А. МИЛЕНКОВИЋ**

# АРХИТЕКТУРА И ОРГАНИЗАЦИЈА РАЧУНАРА

## ПРЕКИДИ, МАГИСТРАЛА И УЛАЗ/ИЗЛАЗ

### ЗБИРКА РЕШЕНИХ ЗАДАТАКА

**Београд 2013.**

# САДРЖАЈ

# 1 УЛАЗ/ИЗЛАЗ

## 1.1 ЗАДАТАК

Дат је рачунарски систем који чине процесор, меморија и контролери периферија PER0 и PER1 повезани магистралом.

Процесор је са једноадресним форматом инструкција које се извршавају над 16 битним целобројним величинама. Од програмски доступних регистара постоји само 16 битни акумулатор ACC и програмска статусна реч PSW са индикаторима N, Z, C и V који се постављају приликом извршавања инструкције преноса у акумулатор и аритметичких, логичких и померачких инструкција. При прекиду хардверски се на стеку чувају програмски бројач PC и програмска статусна реч PSW. Меморијски и улазно-излазни адресни простори су раздвојени.

Меморијски адресни простор је величине $2^{16}$ меморијских локација, при чему је ширина меморијске локације 16 бита.

Улазно-излазни адресни простор је величине $2^{16}$ регистара, при чему је ширина регистра 16 бита. У улазно-излазном адресном простору контролери периферија PER0 и PER1 имају управљачке регистре (*Control*), статусне регистре (*Status*) и регистре података (*Data*) и то редом на адресама FF00h, FF01h и FF02h за контролер периферије PER0 и FF10h, FF11h и FF12h за контролер периферије PER1. У регистрима контролера периферија највиши бит је означен са 15, а најнижи са 0. У управљачким регистрима *Control* бит 0 је бит *enable* којим се дозвољава прекид, бит 1 је бит *u/i* којим се задаје режим улаза или излаза (0—улаз, 1—излаз) и бит 15 је бит *start* којим се дозвољава почетак операције. У статусним регистрима *Status* бит 0 је бит спремности *ready*.

Адресне линије и линије података магистрале рачунара су широке по 16 бита.

Написати програм којим се блок од 200h података учитава са PER0, смешта у меморију од локације F000h, обрађује процедуром *Obrada* и резултат шаље на PER1. Операције улаза и излаза реализовати техникама програмираних улаза и излаза испитивањем бита спремности *ready* статусних регистара *Status*. Процедуру *Obrada* не треба реализовати, већ је само позвати на одговарајућем месту у програму помоћу наредбе `CALL Obrada`. Процедура *Obrada* не мења место ни дужину блока података.

**Решење:**

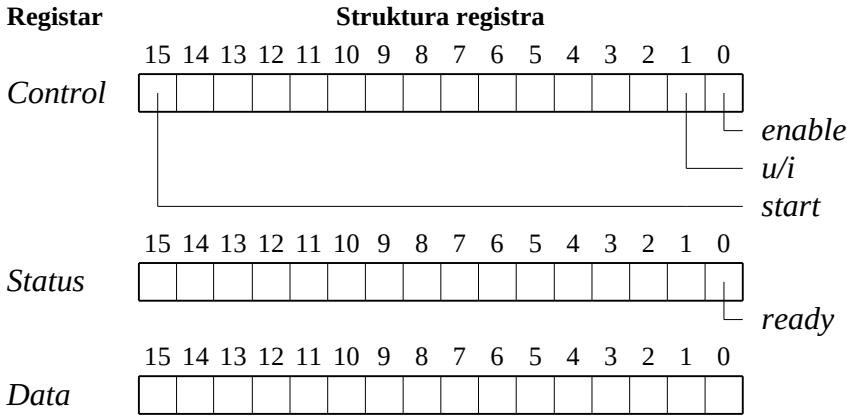
Адресе и структура регистара контролера периферија PER0 и PER1 су дати на слици1.

Kontroler periferije PER0

| Registar | Control | Stataus | Data |
|---|---|---|---|
| Adresa | FF00h | FF01h | FF02h |

Kontroler periferije PER1

| Registar | Control | Stataus | Data |
|---|---|---|---|
| Adresa | FF10h | FF11h | FF12h |




Слика 1 Адресе и структура регистара контролера периферија PER0 и PER1

Тражени програм је приказан на слици 2.

```
;unos bloka podataka sa periferije PER0 u memoriju
  ;inicijalizacija prenosa iz periferije PER0
       LOAD   #200h     ;upis veličine bloka podataka za PER0 preko ACC u
       STORE  MemCnt    ;mem. lok. na adr. MemCnt
       LOAD   #F000h    ;upis adr. bloka podataka za PER0 preko ACC u
       STORE  MemAdr0   ;mem. lok. na adr. MemAdr0
  ;startovanje kontrolera periferije PER0
       LOAD   #8000h    ;upis režima rada i startovanja PER0 (start=1,
       OUT    FF00h     ;u/i=0,enable=0)preko ACC u reg. Control PER0
  ;unos bloka podataka
    ;provera da li podatak može da se prenese iz registra Data
Loop0: IN     FF01h     ;upis sadrž. reg. Status PER0 u ACC
       AND    #1        ;ispitivanje bita ready
       JZ     Loop0     ;povratak na Loop0 ukoliko je ready 0
    ;prenos podatka iz registra Data u memorijsku lokaciju
       IN     FF02h     ;upis sadrž. reg. Data PER0 preko ACC u mem. lok.
       STORE  (MemAdr0) ;na adresi datoj sadrž. mem. lok. na adr. MemAdr0
    ;inkrementiranje sadržaja memorijske lokacije MemAdr0
       LOAD   MemAdr0   ;upis sadrž. mem.lok. sa adr. MemAdr0 u ACC
       INC              ;inkrementiranje sadrž. ACC
       STORE  MemAdr0   ;upis sadrž. ACC u mem.lok. na adr. MemAdr0
    ;dekrementiranje sadržaja memorijske lokacije MemCnt
       LOAD   MemCnt    ;upis sadrž. mem.lok. sa adr. MemCnt u ACC
       DEC              ;dekrementiranje sadrž. ACC
       STORE  MemCnt    ;upis sadrž. ACC u mem.lok. na adr. MemCnt
    ;provera da li je prenet poslednji podatak
       JNZ    Loop0     ;povratak na Loop0 ukoliko nije poslednji podatak
  ;zaustavljanje kontrolera periferije PER0
       LOAD   #0        ;upis režima zaustavljanja PER0 (start=0, u/i=0,
       OUT    FF00h     ;enable=0)preko ACC u reg. Control PER0
;obrada
       CALL   Obrada    ;obrada bloka od 200h podataka u memorijskim
                        ;lokacijama od adrese F000h do adrese F1FFh
;slanje bloka podataka iz memorije u periferiju PER1
  ;inicijalizacija prenosa u periferiju PER1
       LOAD   #200h     ;upis veličine bloka podataka za PER1 preko ACC u
       STORE  MemCnt    ;mem. lok. na adr. MemCnt
       LOAD   #F000h    ;upis adr. bloka podataka za PER1 preko ACC u
       STORE  MemAdr1   ;mem. lok. na adr. MemAdr1
  ;startovanje kontrolera periferije PER1
       LOAD   #8002h    ;upis režima rada i startovanja PER1 (start=1,
       OUT    FF10h     ;u/i=1,enable=0) preko ACC u reg. Control PER1
```

```
  ;slanje bloka podataka
    ;provera da li podatak može da se prenese u registar Data
Loop1: IN     FF11h     ;upis sadrž. reg. Status PER1 u ACC
       AND    #1        ;ispitivanje bita ready
       JZ     Loop1     ;povratak na Loop1 ukoliko je ready 0
    ;prenos podatka iz memorijske lokacije u registar Data
       LOAD   (MemAdr1) ;upis sadrž. mem. lok. sa adr. date sadrž. mem.
       OUT    FF12h     ;lok. sa adr. MemAdr1 preko ACC u reg. Data PER1
    ;inkrementiranje sadržaja memorijske lokacije MemAdr1
       LOAD   MemAdr1   ;upis sadrž. mem.lok. sa adr. MemAdr1 u ACC
       INC              ;inkrementiranje sadrž. ACC
       STORE  MemAdr1   ;upis sadrž. ACC u mem.lok. na adr. MemAdr1
    ;dekrementiranje sadržaja memorijske lokacije MemCnt
       LOAD   MemCnt    ;upis sadrž. mem.lok. sa adr. MemCnt u ACC
       DEC              ;dekrementiranje sadrž. ACC
       STORE  MemCnt    ;upis sadrž. ACC u mem.lok. na adr. MemCnt
    ;provera da li je prenet poslednji podatak
       JNZ    Loop1     ;povratak na Loop1 ukoliko nije poslednji podatak
  ;zaustavljanje kontrolera periferije PER1
       LOAD   #0        ;upis režima zaustavljanja PER0(start=0, u/i=0,
       OUT    FF10h     ;enable=0)preko ACC u reg. Control PER1
```

Слика 2 Програм

Приликом уноса блока података са периферије PER0 у меморију бит спремности *ready* статусног регистра *Status* контролера периферије PER0 се поставља на вредности 0 и 1 на начин приказан у даљем тексту. Унос блок података са периферије PER0 у меморију започиње тако што се извршавањем одговарајућег програма у управљачки регистар *Control* контролера периферије PER0 упише садржај којим се контролер стартује (*start*=1) за рад у режиму улаза (*u/i*=0) и без генерисања прекида (*enable*=0). Том приликом контролер поставља на вредност 0 бит спремности *ready* статусног регистра *Status* и започиње читање података из периферије PER0 и пренос у регистар података *Data*. Контролер, приликом уписа података из периферије PER0 у регистар *Data*, поставља бит *ready* на вредност 1. Када се, у оквиру извршавања одговарајућег програма којим се преноси садржај регистра *Data* у меморијску локацију, чита регистар *Data*, контролер поставља на вредност 0 бит *ready* и започиње читање следећег података из периферије PER0 и пренос у регистар података *Data*.

Приликом слања блока података из меморије у периферију PER1 бит спремности *ready* статусног регистра *Status* контролера периферије PER0 се поставља на вредности 1 и 0 на начин приказан у даљем тексту. Слање блока података из меморију у периферију PER1 започиње тако што се извршавањем одговарајућег програма у управљачки регистар *Control* контролера периферије PER1 упише садржај којим се контролер стартује (*start*=1) за рад у режиму излаза (*u/i*=1) и без генерисања прекида (*enable*=0). Том приликом контролер поставља на вредност 1 бит спремности *ready* статусног регистра *Status*. Када се, у оквиру извршавања одговарајућег програма којим се преноси садржај меморијске локације у регистар *Data*, врши упис у регистар *Data*, контролер поставља на вредност 0 бит *ready* и започиње пренос података из регистра података *Data* у периферију PER1. Контролер, када заврши пренос података из регистра података *Data* у периферију PER1, поставља бит *ready* на вредност 1.

Програм се састоји из три дела. У првом делу се из периферије PER0 техником програмираног улаза испитивањем бита спремности *ready* статусног регистра *Status* контролера периферије PER0 уноси блок од 200h података и смешта у меморијске локације од адресе F000h до адресе F1FFh. У другом делу се позива процесура *Obrada* која врши одређену обраду над унетим подацима у блоку података у меморијским локацијама од адресе F000h до адресе F1FFh и резултат смешта у исте меморијске локације од адресе F000h до адресе F1FFh. У трећем делу се у периферију PER1 техником програмираног излаза испитивањем бита спремности *ready* статусног регистра *Status* контролера периферије PER1 шаље блок од 200h података прочитаних из меморијских локација од адресе F000h до адресе F1FFh.

У првом делу се уноси блок података са периферије PER0 у меморију тако што се најпре се врши иницијализација преноса из периферије PER0 и стартовање контролера периферије PER0, затим сам унос блока података и на крају заустављање контролера периферије PER0.

У оквиру иницијализације преноса се најпре у меморијску локацију чија је адреса симболички означена са MemCnt уписује вредност 200h која представља величину блока података који треба пренети, а затим се у меморијску локацију чија је адреса симболички означена са MemAdr0 уписује вредност F000h која представља почетну адресу дела меморије у који треба пренети блок података. У оквиру стартовања контролера периферије PER0 у управљачки регистар *Control* контролера периферије PER0, који се налази на адреси FF00h у улазно/излазном адресном простору, се уписује вредност 8000h чиме се контролер стартује (*start*=1) за рад у режиму улаза (*u/i*=0) и без генерисања прекида (*enable*=0).

У оквиру уноса блока података се најпре у унутрашњој петљи проверава да ли податак може да се пренесе из регистра *Data* у меморијску локацију, а затим се у спољашњој петљи најпре реализује пренос податка из регистра *Data* у меморијску локацију, потом инкрементира садржај меморијске локације MemAdr0 и декрементира садржај меморијске локације MemCnt и на крају на основу провере да ли је пренет последњи податак или остаје у петљи и продужава са преносом података или излази из петље и завршава са преносом података. Провера да ли податак може да се пренесе из регистра *Data* у меморијску локацију се реализује тако што се у унутрашњој петљи најпре садржај статусног регистра *Status* контролера периферије PER0, који се налази на адреси FF01h у улазно/излазном адресном простору, преноси у акумулатор ACC и затим реализује логичка И операција са непосредном величином (#1h) која на позицији бита *ready* има вредност 1 а на осталим битовима има вредност 0. Уколико бит *ready* има вредност 0, резултат И операције ће бити уписивање вредности 0h у акумулатор ACC и постављање индикатора Z програмске статусне речи PSW на вредност 1, док у случају да бит *ready* има вредност 1, резултат И операције ће бити уписивање вредности 1h у акумулатор ACC и постављање индикатора Z програмске статусне речи PSW на вредност 0. На крају унутрашње петље се реализује условни скок на лабелу Loop0 и остаје у унутрашњој петљи уколико индикатор Z има вредност 1 и излази из унутрашње петље уколико индикатор Z има вредност 0. Пренос податка из регистра *Data* у меморијску локацију се реализује тако што се најпре садржај регистра податка *Data* контролера периферије PER0, који се налази на адреси FF02h у улазно/излазном адресном простору, преноси у акумулатор ACC и затим из акумулатора ACC уписује у меморијску локацију на адреси одређеној садржајем меморијске локације MemAdr0. Садржај меморијске локације MemAdr0 представља показивач на меморијску локацију у коју треба пренети следећи податак, па се зато њен садржај инкрементира после преноса сваког податка. Како је усвојено да је могуће инкрементирати само садржај акумулатора ACC, најпре се садржај меморијске локације MemAdr0 пребацује у акумулатор ACC и затим се инкрементирани садржај акумулатора ACC враћа у меморијску локацију MemAdr0. Садржај меморијске локације MemCnt представља преостали број података из блока података које треба пренети, па се зато њен садржај декрементира после преноса сваког податка. Како је усвојено да је могуће декрементирати само садржај акумулатора ACC, најпре се садржај меморијске локације MemCnt пребацује у акумулатор ACC и затим се декрементирани садржај акумулатора ACC враћа у меморијску локацију MemCnt. Уколико је као резултат декрементирања садржај акумулатор ACC постао 0, индикатор Z програмске статусне речи PSW ће се поставити на вредност 1, док ће се у супротном случају поставити на вредност 0. Инструкција STORE којом се декрементирани садржај акумулатора ACC враћа у меморијску локацију MemCnt не мења вредност индикатора Z постављену на основу резултата декрементирања. На крају спољашње петље се реализује условни скок на лабелу Loop0 и остаје у спољашњој петљи уколико индикатор Z има вредност 0 и излази из спољашње петље уколико индикатор Z има вредност 1.

У оквиру заустављања контролера периферије PER0 у управљачки регистар *Control* контролера периферије PER0 се уписује вредност 0h, што има за последицу да се контролер зауставља (*start*=0). При томе уписивање вредности 0 у битове којима се задаје режим рада (*u/i*=0 и *enable*=0) нема никаквог значаја.

У другом делу се позива процесура *Obrada* која врши одређену обраду над унетим подацима у блоку података у меморијским локацијама од адресе F000h до адресе F1FFh и резултат смешта у исте меморијске локације од адресе F000h до адресе F1FFh.

У трећем делу се шаље блок података из меморије у периферију PER1 и то на сличан начин као што се шаље блок података из периферије PER0 у меморију. Разлика је у самом преносу јер се сада садржај меморијске локације са адресе одређенс садржајем меморијске локације MemAdr1

преноси у акумулатор ACC и затим из акумулатора ACC уписује у регистар податка *Data* контролера периферије PER1, који се налази на адреси FF12h у улазно/излазном адресном простору.

**Дискусија:**

Процесор је једноадресни, па се сви преноси података, који могу да се нађу у меморијским локацијама, регистрима контролера периферија или у самој инструкцији, уколико се користе непосредно адресирање, реализују преко акумулатора ACC. Поред тога, меморијски и улазно-излазни адресни простори су раздвојени, па се регистрима у контролерима периферија PER0 и PER1 приступа искључиво инструкцијама IN и OUT. У складу са тим уписивање вредности 200h у меморијску локацију на адреси MemCnt се реализује инструкцијама LOAD #200h и STORE MemCnt, а уписивање вредности #8000h у регистар *Control* контролера периферије PER0 на адреси FF00h инструкцијама LOAD #8000h и OUT FF00h.

Усвојено је да су инструкције INC и DEC безадресне и да се њима инкрементира и декрементира садржај регистра акумулатора ACC. Због тога, када се јави потреба да се садржај неке локације инкрементира или декрементира, садржај дате локације се најпре пребацује у акумулатор ACC, затим се садржај акумулатора инкрементира или декрементира и на крају добијена вредност враћа у локацију. У складу са тим инкрементирање садржаја меморијске локације на адреси MemAdr0 се реализује инструкцијама LOAD MemAdr0, INC и STORE MemAdr0. Међутим, инструкције INC и DEC могу да имају и једноадресни формат, па би се тада инкрементирање садржаја меморијских локација MemAdr0 и MemAdr1 и декрементирање садржаја меморијске локације MemCnt реализовало једном инструкцијом (INC MemAdr0, INC MemAdr1 и DEC MemCnt).

Провера вредности бита *ready* статусног регистра *Status* контролера периферије мора да се реализује логичком инструкцијама AND или TST (логичко поређење), а не аритметичком инструкцијама SUB или CMP (аритметичко поређење).

Могуће је и да се одмах на почетку стартују оба контролера, али се тиме не добија ништа.

## 1.2 ЗАДАТАК

Дат је рачунарски систем из задатка 1.1.

Написати програм којим се блок од 200h података учитава са PER0, смешта у меморију од локације F000h, обрађује процедуром *Obrada* и резултат шаље на PER1. Операцију улаза са периферије PER0 реализовати техником програмираног улаза испитивањем бита спремности *ready* статусног регистра *Status*, а операцију излаза на периферију PER1 техником програмираног излаза са прекидом. Процедуру *Obrada* не треба реализовати, већ је само позвати на одговарајућем месту у програму помоћу наредбе `CALL Obrada`. Процедура *Obrada* не мења место ни дужину блока података.

**Решење:**

Тражени програм је приказан на слици 3.

```
;glavni program
;unos bloka podataka sa periferije PER0 u memoriju
  ;inicijalizacija prenosa iz periferije PER0
       LOAD   #200h     ;upis veličine bloka podataka za PER0 preko ACC u
       STORE  MemCnt    ;mem. lok. na adr. MemCnt
       LOAD   #F000h    ;upis adr. bloka podataka za PER0 preko ACC u
       STORE  MemAdr0   ;mem. lok. na adr. MemAdr0
  ;startovanje kontrolera periferije PER0
       LOAD   #8000h    ;upis režima rada i startovanja PER0 (start=1,
       OUT    FF00h     ;u/i=0,enable=0)preko ACC u reg. Control PER0
  ;unos bloka podataka
    ;provera da li podatak može da se prenese iz registra Data
Loop0: IN     FF01h     ;upis sadrž. reg. Status PER0 u ACC
       AND    #1        ;ispitivanje bita ready
       JZ     Loop0     ;povratak na Loop0 ukoliko je ready 0
    ;prenos podatka iz registra Data u memorijsku lokaciju
       IN     FF02h     ;upis sadrž. reg. Data PER0 preko ACC u mem. lok.
       STORE  (MemAdr0) ;na adresi datoj sadrž. mem. lok. na adr. MemAdr0
    ;inkrementiranje sadržaja memorijske lokacije MemAdr0
       LOAD   MemAdr0   ;upis sadrž. mem.lok. sa adr. MemAdr0 u ACC
       INC              ;inkrementiranje sadrž. ACC
       STORE  MemAdr0   ;upis sadrž. ACC u mem.lok. na adr. MemAdr0
    ;dekrementiranje sadržaja memorijske lokacije MemCnt
       LOAD   MemCnt    ;upis sadrž. mem.lok. sa adr. MemCnt u ACC
       DEC              ;dekrementiranje sadrž. ACC
       STORE  MemCnt    ;upis sadrž. ACC u mem.lok. na adr. MemCnt
    ;provera da li je prenet poslednji podatak
       JNZ    Loop0     ;povratak na Loop0 ukoliko nije poslednji podatak
  ;zaustavljanje kontrolera periferije PER0
       LOAD   #0        ;upis režima zaustavljanja PER0(start=0, u/i=0,
       OUT    FF00h     ;enable=0)preko ACC u reg. Control PER0
;obrada
       CALL   Obrada    ;obrada bloka od 200h podataka u memorijskim
                        ;lokacijama od adrese F000h do adrese F1FFh
;slanje bloka podataka iz memorije u periferiju PER1
  ;inicijalizacija prenosa u periferiju PER1
       LOAD   #200h     ;upis veličine bloka podataka za PER1 preko ACC u
       STORE  MemCnt    ;mem. lok. na adr. MemCnt
       LOAD   #F000h    ;upis adr. bloka podataka za PER1 preko ACC u
       STORE  MemAdr1   ;mem. lok. na adr. MemAdr1
  ;postavljanje "semafor"-a na 1
       LOAD   #1h       ;upis vrednosti 1 preko ACC u
       STORE  MemSem1   ;mem. lok. na adr. MemSem1
```

```
  ;startovanje kontrolera periferije PER1
       LOAD   #8003h    ;upis režima rada i startovanja PER1 (start=1,
       OUT    FF10h     ;u/i=1,enable=1)preko ACC u reg. Control PER1
  ;čekanje na završetak slanja bloka podataka iz memorije u periferiju
  ;PER1 proverom vrednosti "semafor"-a
Wait1: LOAD   MemSem1   ;upis sadrž. mem.lok. sa adr. MemSem1 u ACC
       CMP    #0        ;ispitivanje da li je "semafor" postao 0
       JNZ    Wait1     ;povratak na Wait1 ukoliko "semafor" nije 0
. . .
;prekidna rutina periferije PER1
;slanje bloka podataka iz memorije u periferiju PER1
PER1:  PUSHA             ;akumulator ACC na stek
  ;prenos podatka iz memorijske lokacije u registar Data
       LOAD   (MemAdr1) ;upis sadrž. mem. lok. sa adr. date sadrž. mem.
       STORE  FF12H     ;lok. sa adr. MemAdr1 preko ACC u reg. Data PER1
  ;inkrementiranje sadržaja memorijske lokacije MemAdr1
       LOAD   MemAdr1   ;upis sadrž. mem.lok. sa adr. MemAdr1 u ACC
       INC              ;inkrementiranje sadrž. ACC
       STORE  MemAdr1   ;upis sadrž. ACC u mem.lok. na adr. MemAdr1
  ;dekrementiranje sadržaja memorijske lokacije MemCnt
       LOAD   MemCnt    ;upis sadrž. mem.lok. sa adr. MemCnt u ACC
       DEC              ;dekrementiranje sadrž. ACC
       STORE  MemCnt    ;upis sadrž. ACC u mem.lok. na adr. MemCnt
  ;provera da li je prenet poslednji podatak
       JNZ    Back      ;prelaz na Back ukoliko nije poslednji podatak
  ;zaustavljanje kontrolera periferije PER1
       LOAD   #0        ;upis režima zaustavljanja PER1(start=0, u/i=0,
       STORE FF10H      ;enable=0)preko ACC u reg. Control PER1
  ;postavljanje "semafor"-a na 0
       LOAD   #0        ;upis vrednosti 0 preko ACC u
       STORE  MemSem1   ;mem. lok. na adr. MemSem1
  ;izlazak iz prekidne rutine
Back:  POPA              ;restauracija akumulatora ACC sa steka
       RTI               ;povratak iz prekidne rutine
```

Слика 3 Програм

Програм се састоји из три дела. У првом делу се из периферије PER0 техником програмираног улаза испитивањем бита спремности *ready* статусног регистра *Status* контролера периферије PER0 уноси блок од 200h података и смешта у меморијске локације од адресе F000h до адресе F1FFh. У другом делу се позива процесура *Obrada* која врши одређену обраду над унетим подацима у блоку података у меморијским локацијама од адресе F000h до адресе F1FFh и резултат смешта у исте меморијске локације од адресе F000h до адресе F1FFh. У трећем делу се у периферију PER1 техником програмираног излаза са прекидом шаље блок од 200h података прочитаних из меморијских локација од адресе F000h до адресе F1FFh. Прва два дела програма су идентична као прва два дела програма у задатку 1.1. У трећем делу постоји разлика јер се за слање блока података из меморије у периферију PER1 уместо технике програмираног излаза испитивањем бита спремности *ready* статусног регистра *Status* контролера периферије PER1 користи техника техником програмираног излаза са прекидом.

Део програма којим се шаље блок података из меморије у периферију PER1 има два дела и то један део који се извршава у главном програму и други део који се извршава у прекидној рутини периферије PER1. У оквиру дела програма који се извршава у главном програму врши се иницијализација преноса у периферију PER1, постављање "semafor"-a na 1, стартовање контролера периферије PER1 и чекање на завршетак слања блока података из меморије у периферију PER1. У оквиру дела програма који се извршава у прекидној рутини периферије PER1 врши се пренос податка из меморијске локације у регистар *Data* контролера периферије PER1, инкрементирање садржаја меморијске локације MemAdr1 и декрементирање садржаја меморијске локације MemCnt и, уколико је пренет последњи податак, заустављање контролера периферије PER1 и постављање "semafor"-a na 0.

У оквиру иницијализације преноса у периферију PER1 се најпре у меморијску локацију чија је адреса симболички означена са MemCnt уписује вредност 200h која представља величину блока

података који треба пренети, а затим се у меморијску локацију чија је адреса симболички означена са MemAdr1 уписује вредност F000h која представља почетну адресу дела меморије из кога треба пренети блок података. У оквиру постављања "semafor"-а на 1, у меморијску локацију MemSem1 се уписује вредност 1 која служи као индикација да је пренос података у току и да се не сме продужити са извршавањем главног програма. У оквиру стартовања контролера периферије PER1 у управљачки регистар *Control* контролера периферије PER1, који се налази на адреси FF10h у улазно/излазном адресном простору, се уписује вредност 8003h чиме се контролер стартује (*start*=1) за рад у режиму излаза (*u/i*=1) са генерисањем прекида (*enable*=1).

Од овог тренутка процесор и контролер периферије PER1 почињу да раде паралелно. Процесор чека завршетак преноса блока података из меморије у периферију PER1 у петљи са лабелом Wait1. Контролер периферије PER1, најпре, пошто је стартован (*start*=1) за рад у режиму излаза (*u/i*=1), поставља на вредност 1 бит спремности *ready* статусног регистра *Status*, а потом, пошто је стартован (*start*=1) за рад са генерисањем прекида (*enable*=1), генерише прекид и чека да се податак пренесе из меморијске локације у регистар *Data*.

На прекид генерисан од стране контролера периферије PER1, процесор излази из петље са лабелом Wait1 и прелази на извршавање инструкција прекидне рутине периферије PER1 на чијем почетку преноси податак из меморијске локације у регистар *Data*. Од овог тренутка процесор и контролер периферије PER1 настављају да раде паралелно. Процесор наставља са извршавањем инструкција прекидне рутине тако што инкрементира садржај меморијске локадине MemAdr1, декрементира садржај меморијске локације MemCnt и, уколико није пренет последњи податак, враћа се у петљу са лабелом Wait1 прекинутог главног програма. Контролер поставља на вредност 0 бит *ready* и започиње пренос податка из регистра података *Data* у периферију PER1 и по његовом завршетку, поставља бит *ready* на вредност 1 и поново генерише прекид.

Рад процесора и контролера приказан у претходном параграфу се наставља до преласка процесора на прекидну рутину у оквиру којег се преноси последњи податак. Када садржај меморијске локације MemCnt декрементирањем постане 0, процесор најпре изврши инструкције прекидне рутине којима се постављање "semafor"-а на 0 и зауставља контролера периферије PER1, па се затим враћа се у петљу са лабелом Wait1 прекинутог главног програма.

Током наставка извршавања програма у петљи са лабелом Wait1 утврдиће се да "semafor"-а сада има вредност 0. Ова вредност служи као индикација да је пренос података завршен, па се излази из петље са лабелом Wait1 и продужава извршавање главног програма.

## 1.3 ЗАДАТАК

Дат је рачунарски систем из задатка 1.1.

Написати програм којим се блок од 200h података учитава са PER0, смешта у меморију од локације F000h, обрађује процедуром *Obrada* и резултат шаље на PER1. Операцију улаза са периферије PER0 реализовати техником програмираног улаза са прекидом, а операцију излаза на периферију PER1 техником програмираног излаза са испитивањем бита спремности *ready* статусног регистра *Status*. Процедуру *Obrada* не треба реализовати, већ је само позвати на одговарајућем месту у програму помоћу наредбе `CALL Obrada`. Процедура *Obrada* не мења место ни дужину блока података.

**Решење:**

Тражени програм је приказан на слици 4.

```
;glavni program
;unos bloka podataka iz periferije PER0 u memoriju
  ;inicijalizacija prenosa iz periferije PER0
       LOAD   #200h     ;upis veličine bloka podataka za PER0 preko ACC u
       STORE  MemCnt    ;mem. lok. na adr. MemCnt
       LOAD   #F000h    ;upis adr. bloka podataka za PER0 preko ACC u
       STORE  MemAdr0   ;mem. lok. na adr. MemAdr0
  ;postavljanje "semafor"-a na 1
       LOAD   #1h       ;upis vrednosti 1 preko ACC u
       STORE  MemSem0   ;mem. lok. na adr. MemSem0
  ;startovanje kontrolera periferije PER0
       LOAD   #8001h    ;upis režima rada i startovanja PER0 (start=1,
       OUT    FF00h     ;u/i=0,enable=1)preko ACC u reg. Control PER0
  ;čekanje na završetak unosa bloka podataka iz periferiju PER0 u memoriju
  ;proverom vrednosti "semafor"-a
Wait0: LOAD   MemSem0   ;upis sadrž. mem.lok. sa adr. MemSem0 u ACC
       CMP    #0        ;ispitivanje da li je "semafor" postao 0
       JNZ    Wait0     ;povratak na Wait0 ukoliko "semafor" nije 0
;obrada
       CALL   Obrada    ;obrada bloka od 200h podataka u memorijskim
                        ;lokacijama od adrese F000h do adrese F1FFh
;slanje bloka podataka iz memorije u periferiju PER1
  ;inicijalizacija prenosa u periferiju PER1
       LOAD   #200h     ;upis veličine bloka podataka za PER1 preko ACC u
       STORE  MemCnt    ;mem. lok. na adr. MemCnt
       LOAD   #F000h    ;upis adr. bloka podataka za PER1 preko ACC u
       STORE  MemAdr1   ;mem. lok. na adr. MemAdr1
  ;startovanje kontrolera periferije PER1
       LOAD   #8002h    ;upis režima rada i startovanja PER1 (start=1,
       OUT    FF10h     ;u/i=1,enable=0)preko ACC u reg. Control PER1
  ;slanje bloka podataka
    ; provera da li podatak može da se prenese u registar Data
Loop1: IN     FF11h     ;upis sadrž. reg. Status PER1 u ACC
       AND    #1        ;ispitivanje bita ready
       JZ     Loop1     ;povratak na Loop1 ukoliko je ready 0
    ;prenos podatka iz memorijske lokacije u registar Data
       LOAD   (MemAdr1) ;upis sadrž. mem. lok. sa adr. date sadrž. mem.
       OUT    FF12h     ;lok. sa adr. MemAdr1 preko ACC u reg. Data PER1
    ;inkrementiranje sadržaja memorijske lokacije MemAdr1
       LOAD   MemAdr1   ;upis sadrž. mem.lok. sa adr. MemAdr1 u ACC
       INC              ;inkrementiranje sadrž. ACC
       STORE  MemAdr1   ;upis sadrž. ACC u mem.lok. na adr. MemAdr1
```

```
     ;dekrementiranje sadržaja memorijske lokacije MemCnt
       LOAD   MemCnt    ;upis sadrž. mem.lok. sa adr. MemCnt u ACC
       DEC              ;dekrementiranje sadrž. ACC
       STORE  MemCnt    ;upis sadrž. ACC u mem.lok. na adr. MemCnt
     ;provera da li je prenet poslednji podatak
       JNZ    Loop1     ;povratak na Loop1 ukoliko nije poslednji podatak
   ;zaustavljanje kontrolera periferije PER1
       LOAD   #0        ;upis režima zaustavljanja PER0(start=0, u/i=0,
       OUT    FF10h     ;enable=0)preko ACC u reg. Control PER1
. . .
;prekidna rutina periferije PER0
;unos bloka podataka iz periferiju PER0 u memoriju
PER0:  PUSHA            ;akumulator ACC na stek
     ;prenos podatka iz registra Data u memorijsku lokaciju
       IN     FF02h     ;upis sadrž. reg. Data PER0 preko ACC u mem. lok.
       STORE  (MemAdr0) ;na adresi datoj sadrž. mem. lok. na adr. MemAdr0
     ;inkrementiranje sadržaja memorijske lokacije MemAdr0
       LOAD   MemAdr0   ;upis sadrž. mem.lok. sa adr. MemAdr0 u ACC
       INC              ;inkrementiranje sadrž. ACC
       STORE  MemAdr0   ;upis sadrž. ACC u mem.lok. na adr. MemAdr0
     ;dekrementiranje sadržaja memorijske lokacije MemCnt
       LOAD   MemCnt    ;upis sadrž. mem.lok. sa adr. MemCnt u ACC
       DEC              ;dekrementiranje sadrž. ACC
       STORE  MemCnt    ;upis sadrž. ACC u mem.lok. na adr. MemCnt
   ;provera da li je prenet poslednji podatak
       JNZ    Back      ;prelaz na Back ukoliko nije poslednji podatak
   ;zaustavljanje kontrolera periferije PER0
       LOAD   #0        ;upis režima zaustavljanja PER1(start=0, u/i=0,
       STORE FF10H      ;enable=0)preko ACC u reg. Control PER0
   ;postavljanje "semafor"-a na 0
       LOAD   #0        ;upis vrednosti 0 preko ACC u
       STORE  MemSem0   ;mem. lok. na adr. MemSem0
   ;izlazak iz prekidne rutine
Back:  POPA             ;restauracija akumulatora ACC sa steka
       RTI              ;povratak iz prekidne rutine
```

Слика 4 Програм

Програм се састоји из три дела. У првом делу се из периферије PER0 техником програмираног улаза са прекидом уноси блок од 200h података и смешта у меморијске локације од адресе F000h до адресе F1FFh. У другом делу се позива процесура *Obrada* која врши одређену обраду над унетим подацима у блоку података у меморијским локацијама од адресе F000h до адресе F1FFh и резултат смешта у исте меморијске локације од адресе F000h до адресе F1FFh. У трећем делу се у периферију PER1 техником програмираног излаза испитивањем бита спремности *ready* статусног регистра *Status* контролера периферије PER1 шаље блок од 200h података прочитаних из меморијских локација од адресе F000h до адресе F1FFh. Други и трећи део су идентични као други и трећи део у задатку 1.1. У првом делу постоји разлика јер се за унос блока података из периферије PER0 у меморију уместо технике програмираног излаза испитивањем бита спремности *ready* статусног регистра *Status* контролера периферије PER0 користи техника техником програмираног излаза са прекидом.

Део програма којим се уноси блок података из периферије PER0 у меморију има два дела и то један део који се извршава у главном програму и други део који се извршава у прекидној рутини периферије PER0. У оквиру дела програма који се извршава у главном програму врши се иницијализација преноса из периферију PER0, постављање "semafor"-a na 1, стартовање контролера периферије PER0 и чекање на завршетак уноса блока података из периферије PER0 у меморију. У оквиру дела програма који се извршава у прекидној рутини периферије PER0 врши се пренос податка из регистра *Data* контролера периферије PER0 у меморијску локацију, инкрементирање садржаја меморијске локације MemAdr0 и декрементирање садржаја меморијске локације MemCnt и, уколико је пренет последњи податак, заустављање контролера периферије PER0 и постављање "semafor"-a na 0.

У оквиру иницијализације преноса у периферију PER0 се најпре у меморијску локацију чија је адреса симболички означена са MemCnt уписује вредност 200h која представља величину блока података који треба пренети, а затим се у меморијску локацију чија је адреса симболички означена са MemAdr0 уписује вредност F000h која представља почетну адресу дела меморије у који треба пренети блок података. У оквиру постављања "semafor"-а на 1, у меморијску локацију MemSem0 се уписује вредност 1 која служи као индикација да је пренос података у току и да се не сме продужити са извршавањем главног програма. У оквиру стартовања контролера периферије PER0 у управљачки регистар *Control* контролера периферије PER0, који се налази на адреси FF00h у улазно/излазном адресном простору, се уписује вредност 8001h чиме се контролер стартује (*start*=1) за рад у режиму улаза (*u/i*=0) са генерисањем прекида (*enable*=1).

Од овог тренутка процесор и контролер периферије PER0 почињу да раде паралелно. Процесор чека завршетак преноса блока података из периферију PER0 у меморију у петљи са лабелом Wait0. Контролер периферије PER0, најпре, пошто је стартован (*start*=1) за рад у режиму улаза (*u/i*=0), поставља на вредност 0 бит спремности *ready* статусног регистра *Status*, затим, започиње пренос податка из периферије PER0 у регистар податка *Data*, потом, по упису податка у регистар *Data*, поставља бит *ready* на вредност 1 и, пошто је стартован (*start*=1) за рад са генерисањем прекида (*enable*=1), генерише прекид и на крају чека да се податак пренесе из регистра *Data* у меморијску локациј.

На прекид генерисан од стране контролера периферије PER0, процесор излази из петље са лабелом Wait0 и прелази на извршавање инструкција прекидне рутине периферије PER0 на чијем почетку преноси податак из регистра *Data* у меморијску локације. Од овог тренутка процесор и контролер периферије PER0 настављају да раде паралелно. Процесор наставља са извршавањем инструкција прекидне рутине тако што инкрементира садржај меморијске локадине MemAdr0, декрементира садржај меморијске локације MemCnt и, уколико није пренет последњи податак, враћа се у петљу са лабелом Wait0 прекинутог главног програма. Контролер периферије PER0, најпре, поставља на вредност 0 бит спремности *ready* статусног регистра *Status*, затим, започиње пренос података из периферије PER0 у регистар података *Data*, потом, по упису података у регистар *Data*, поставља бит *ready* на вредност 1 и генерише прекид и на крају чека да се податак пренесе из меморијске локације у регистар *Data*.

Рад процесора и контролера приказан у претходном параграфу се наставља до преласка процесора на прекидну рутину у оквиру којег се преноси последњи податак. Када садржај меморијске локације MemCnt декрементирањем постане 0, процесор најпре изврши инструкције прекидне рутине којима се постављање "semafor"-а на 0 и зауставља контролера периферије PER0, па се затим враћа се у петљу са лабелом Wait0 прекинутог главног програма.

Током наставка извршавања програма у петљи са лабелом Wait0 утврдиће се да "semafor"-а сада има вредност 0. Ова вредност служи као индикација да је пренос података завршен, па се излази из петље са лабелом Wait0 и продужава извршавање главног програма.

Прекидна рутина на улазу чува акумулатор на стеку (то је обавезно јер се хардверски чувају једино PC и PSW, а главни програм користи регистар A при испитивању семафора *MemSem*).

## 1.4 ЗАДАТАК

Процесор, меморија и контролери периферија PER0, PER1 и PER2 су повезани магистралом.

Процесор је са једноадресним форматом инструкција. Од програмски доступних регистара постоји 16 битни акумулатор ACC и програмска статусна реч PSW. Типови података који се користе су 8-битне и 16-битне целобројне величине са знаком и без знака. Инструкције преноса са акумулатором, аритметичке, логичке и померачке инструкција се извршавају и над 8-битним и над 16-битним целобројним величинама. Приликом извршавања инструкција над 8-битним целобројним величинама користи се само нижих 8 битова акумулатора, а приликом извршавања инструкција над 16-битним бинарним речима свих 16 битова. Програмска статусна реч PSW има индикаторе N, Z, C и V који се постављају приликом извршавања инструкција преноса у акумулатор и аритметичких, логичких и померачких инструкција. Улазно-излазни адресни простор је меморијски пресликан.

Меморијски адресни простор је величине $2^{16}$ меморијских локација, при чему је ширина меморијске локације 8 бита.

Контролери периферија PER0, PER1 и PER2 имају управљачке регистре (*Control*), статусне регистре (*Status*) и регистре података (*Data*) и то редом на адресама FF10h, FF11h и FF12h за контролер периферије PER0, FF20h, FF21h и FF22h за контролер периферије PER1 и FF30h, FF31h и FF32h за контролер периферије PER2. У регистрима контролера периферија највиши бит је означен са 7, а најнижи са 0. У управљачким регистрима *Control* бит 0 је бит *enable* којим се дозвољава прекид, бит 3 је бит *u/i* којим се задаје режим улаза или излаза (1—улаз, 0—излаз) и бит 7 је бит *start* којим се дозвољава почетак операције. У статусним регистрима *Status* бит 7 је бит спремности *ready*.

Написати главни програм и одговарајуће прекидне рутине којима се истовремено са периферија PER0 и PER1 учитавају блокови од по 100h бајтова и смештају у меморију почев од адресе 1000h за PER0 и адресе 1100h заPER1, затим блок од унетих 200h бајтова обрађује процесуром Obrada и на крају врши пренос обрађеног блока од 200h бајтова из меморије у периферију PER2. Операције улаза са периферија PER0 и PER1 реализовати техником програмираног улаза са прекидом, а операцију излаза на периферију PER2 техником програмираног излаза испитивањем бита спремности *ready* статусних регистара *Status*. Процедуру Obrada не треба реализовати, већ је само позвати на одговарајућем месту у програму помоћу наредбе CALL Obrada. Процедура Obrada не мења место ни дужину блока података

**Решење:**

Адресе и структура регистара контролера периферија PER0 и PER1 су дати на слици5.

Kontroler periferije PER0

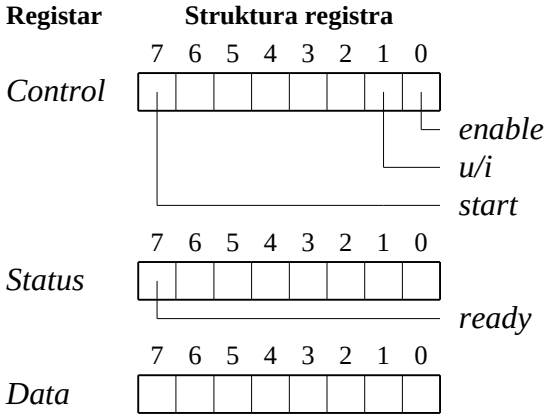
| Registar | Control | Stataus | Data |
|---|---|---|---|
| Adresa | FF10h | FF11h | FF12h |

Kontroler periferije PER1

| Registar | Control | Stataus | Data |
|---|---|---|---|
| Adresa | FF20h | FF21h | FF22h |

Kontroler periferije PER2

| Registar | Control | Stataus | Data |
|---|---|---|---|
| Adresa | FF30h | FF31h | FF32h |




Слика 5 Адресе и структура регистара контролера периферија PER0 и PER1

Тражени програм је приказан на слици 6

```
;glavni program
;unos bloka podataka iz periferija PER0 i PER1 u memoriju
  ;inicijalizacija prenosa iz periferija PER0 i PER1
       LOADW  #100h     ;upis veličine bloka podataka preko ACC u
       STOREW MemCnt0   ;mem. lok. na adr. MemCnt0 za PER0
       STOREW MemCnt1   ;mem. lok. na adr. MemCnt1 za PER1
       LOADW  #1000h    ;upis adr. bloka podataka za PER0 preko ACC u
       STOREW MemAdr0   ;mem. lok. na adr. MemAdr0
       LOADW  #1100h    ;upis adr. bloka podataka za PER1 preko ACC u
       STOREW MemAdr0   ;mem. lok. na adr. MemAdr1
  ;postavljanje "semafor"-a za periferije PER0 i PER1 na 1
       LOADB  #1        ;upis vrednosti 1 preko ACC u
       STOREB MemSem0   ;mem. lok. na adr. MemSem0 za PER0
       STOREB MemSem1   ;mem. lok. na adr. MemSem1 za PER1
  ;startovanje kontrolera periferija PER0 i PER1
       LOADB  #89h      ;upis režima rada i startovanja PER0 i PER1
                        ;(start=1,u/i=1,enable=1)preko ACC
       STOREB FF10h     ;u reg. Control PER0
       STOREB FF20h     ;u reg. Control PER1
  ;čekanje na završetak unosa bloka podataka iz periferije PER0 u memoriju
  ;proverom vrednosti "semafor"-a za PER0
Wait0: LOADB  MemSem0   ;upis sadrž. mem.lok. sa adr. MemSem0 u ACC
       CMPB   #0        ;provera da li je "semafor" za PER0 postao 0
       JNZ    Wait0     ;povratak na Wait0 ukoliko "semafor" nije 0
  ;čekanje na završetak unosa bloka podataka iz periferije PER1 u memoriju
  ;proverom vrednosti "semafor"-a za PER1
Wait1: LOADB  MemSem1   ;upis sadrž. mem.lok. sa adr. MemSem1 u ACC
       CMPB   #0        ;provera da li je "semafor" za PER1 postao 0
       JNZ    Wait1     ;povratak na Wait0 ukoliko "semafor" nije 0
;obrada
       CALL   Obrada    ;obrada bloka od 200h podataka u memorijskim
                        ;lokacijama od adrese F000h do adrese F1FFh
;slanje bloka podataka iz memorije u periferiju PER2
  ;inicijalizacija prenosa u periferiju PER2
       LOADW  #200h     ;upis veličine bloka podataka za PER2 preko ACC u
       STOREW MemCnt2   ;mem. lok. na adr. MemCnt2
       LOADW  #1000h    ;upis adr. bloka podataka za PER2 preko ACC u
       STOREW MemAdr2   ;mem. lok. na adr. MemAdr2
  ;startovanje kontrolera periferije PER2
       LOADB  #80h      ;upis režima rada i startovanja PER2 (start=1,
       STOREB FF30h     ;u/i=0,enable=0)preko ACC u reg. Control PER2
  ;slanje bloka podataka
```

```
     ;provera da li podatak može da se prenese u registar Data
Loop2: LOADB  FF31h     ;upis sadrž. reg. Status PER2 u ACC
       ANDB   #80h      ;ispitivanje bita ready
       JZ     Loop2     ;povratak na Loop2 ukoliko je ready 0
     ;prenos podatka iz memorijske lokacije u registar Data
       LOADB  (MemAdr2) ;upis sadrž. mem. lok. sa adr. date sadrž. mem.
       STOREB FF32h     ;lok. sa adr. MemAdr1 preko ACC u reg. Data PER2
     ;inkrementiranje sadržaja memorijske lokacije MemAdr2
       LOADW  MemAdr2   ;upis sadrž. mem.lok. sa adr. MemAdr2 u ACC
       INCW             ;inkrementiranje sadrž. ACC
       STOREW MemAdr2   ;upis sadrž. ACC u mem.lok. na adr. MemAdr2
     ;dekrementiranje sadržaja memorijske lokacije MemCnt
       LOADW  MemCnt2   ;upis sadrž. mem.lok. sa adr. MemCnt2 u ACC
       DECW             ;dekrementiranje sadrž. ACC
       STOREW MemCnt2   ;upis sadrž. ACC u mem.lok. na adr. MemCnt2
     ;provera da li je prenet poslednji podatak
       JNZ    Loop2     ;povratak na Loop2 ukoliko nije poslednji podatak
  ;zaustavljanje kontrolera periferije PER2
       LOADB  #0        ;upis režima zaustavljanja PER2(start=0, u/i=0,
       STOREB FF30h     ;enable=0)preko ACC u reg. Control PER2

. . .

;prekidna rutina periferije PER0
;unos bloka podataka iz periferiju PER0 u memoriju
PER0:  PUSHA            ;akumulator ACC na stek
     ;prenos podatka iz registra Data u memorijsku lokaciju
       LOADB  FF12h     ;upis sadrž. reg. Data PER0 preko ACC u mem. lok.
       STOREB (MemAdr0) ;na adresi datoj sadrž. mem. lok. na adr. MemAdr0
     ;inkrementiranje sadržaja memorijske lokacije MemAdr0
       LOADW  MemAdr0   ;upis sadrž. mem.lok. sa adr. MemAdr0 u ACC
       INCW             ;inkrementiranje sadrž. ACC
       STOREW MemAdr0   ;upis sadrž. ACC u mem.lok. na adr. MemAdr0
     ;dekrementiranje sadržaja memorijske lokacije MemCnt
       LOADW  MemCnt    ;upis sadrž. mem.lok. sa adr. MemCnt u ACC
       DECW             ;dekrementiranje sadrž. ACC
       STOREW MemCnt    ;upis sadrž. ACC u mem.lok. na adr. MemCnt
  ;provera da li je prenet poslednji podatak
       JNZ    Back      ;prelaz na Back ukoliko nije poslednji podatak
  ;zaustavljanje kontrolera periferije PER0
       LOADB  #0        ;upis režima zaustavljanja PER0(start=0, u/i=0,
       STOREB FF10h     ;enable=0)preko ACC u reg. Control PER0
  ;postavljanje "semafor"-a na 0 za PER0
       LOADB  #0        ;upis vrednosti 0 preko ACC u
       STOREB MemSem0   ;mem. lok. na adr. MemSem0
  ;izlazak iz prekidne rutine
Back:  POPA             ;restauracija akumulatora ACC sa steka
       RTI              ;povratak iz prekidne rutine

. . .

;prekidna rutina periferije PER1
;unos bloka podataka iz periferiju PER1 u memoriju
PER1:  PUSHA            ;akumulator ACC na stek
     ;prenos podatka iz registra Data u memorijsku lokaciju
       LOADB  FF22h     ;upis sadrž. reg. Data PER1 preko ACC u mem. lok.
       STOREB (MemAdr1) ;na adresi datoj sadrž. mem. lok. na adr. MemAdr1
```

```
    ;inkrementiranje sadržaja memorijske lokacije MemAdr1
      LOADW  MemAdr1   ;upis sadrž. mem.lok. sa adr. MemAdr1 u ACC
      INCW             ;inkrementiranje sadrž. ACC
      STOREW MemAdr1   ;upis sadrž. ACC u mem.lok. na adr. MemAdr1
    ;dekrementiranje sadržaja memorijske lokacije MemCnt
      LOADW  MemCnt1   ;upis sadrž. mem.lok. sa adr. MemCnt1 u ACC
      DECW             ;dekrementiranje sadrž. ACC
      STOREW MemCnt1   ;upis sadrž. ACC u mem.lok. na adr. MemCnt1
  ;provera da li je prenet poslednji podatak
      JNZ    Back      ;prelaz na Back ukoliko nije poslednji podatak
  ;zaustavljanje kontrolera periferije PER1
      LOADB  #0        ;upis režima zaustavljanja PER1(start=0, u/i=0,
      STOREB FF20h     ;enable=0)preko ACC u reg. Control PER1
  ;postavljanje "semafor"-a na 0 za PER1
      LOADB  #0        ;upis vrednosti 0 preko ACC u
      STORE  MemSem1   ;mem. lok. na adr. MemSem1
  ;izlazak iz prekidne rutine
Back: POPA             ;restauracija akumulatora ACC sa stek
      RTI              ;povratak iz prekidne rutine
```

Слика 6 Програм

Програм се састоји из три дела. У првом делу се из периферија PER0 и PER1 техником програмираног улаза са прекидом уносе два блока од по 100h података и смештају у меморијске локације од адресе 1000h до адресе 10FFh за периферију PER0 и од адресе 1100h до адресе 11FFh за периферију PER1. У другом делу се позива процесура *Obrada* која врши одређену обраду над унетим подацима у блоковима од по 100h података у меморијским локацијама од адресе 1000h до адресе 10FFh и од адресе 1100h до адресе 11FFh и резултат блок од 200h података смешта у исте меморијске локације од адресе 1000h до адресе 11FFh. У трећем делу се у периферију PER2 техником програмираног излаза испитивањем бита спремности *ready* статусног регистра *Status* контролера периферије PER2 шаље блок од 200h података прочитаних из меморијских локација од адресе 1000h до адресе 11FFh.

Уноси података из периферија PER0 и PER1 у меморију се реализује упоредо техником програмираног улаза са прекидом на начин објашњен у задатку 1.3. Због тога се у првом делу главног програма иницијализују преноси из обе периферије, постављају на 1 "semafor"-и за обе периферије и стартују контролери обе периферије. Прелазак на други део програма у коме се реализује обрада унетих блокова података не сме да се дозволи док се не унесу блокови података из обе периферије. Чекање на завршетак уноса блокова од по 100h података из обе периферије се реализује најпре у петљи са лабелом Wait0 за периферију PER0 а затим и у петљи са лабелом Wait1 за периферију PER1, а сами уноси података из периферија PER0 и PER1 у меморију се реализују у прекидним рутинама периферија PER0 и PER1 на које се прелази када контролери периферија PER0 и PER1 генеришу прекиде. Том приликом могу да се јаве два случаја у зависности од брзине периферија PER0 и PER1. Први случај се јавља када је периферија PER0 бржа од периферије PER1, па ће се унос података из периферије PER0 завршити пре од уноса података из периферије PER1 и прво "semafor" периферије PER0 поставити на вредност 0 а затим и "semafor" периферије PER1 поставити на вредност 0. Због тога ће се по изласку из петље са лабелом Wait0 најпре у петљи са лабелом Wait1 чекати да се заврши и унос података из периферије PER1 и да се "semafor" периферије PER1 поставити на вредност 0 и затим прећи на други део програма у коме се реализује обрада унетих блокова података. Други случај се јавља када је периферија PER0 спорија од периферије PER1, па ће се унос података из периферије PER1 завршити пре од уноса података из периферије PER0 и прво "semafor" периферије PER1 поставити на вредност 0 а затим и "semafor" периферије PER0 поставити на вредност 0. Због тога се по изласку из петље са лабелом Wait0 у петљи са лабелом Wait1 неће чекати, јер ће унос података из периферије PER1 бити завршен и "semafor" периферије PER1 постављен на вредност 0, па ће одмах прећи на други део програма у коме се реализује обрада унетих блокова података.

Други део у коме се врши обрада и трећи део у коме се у периферију PER2 техником програмираног излаза испитивањем бита спремности *ready* статусног регистра *Status* контролера периферије PER2 шаље блок података се реализује на начин објашњен у задатку 1.3

**Дискусија:**

Неопходно је покренути обе улазне операције, па онда чекати на завршетак обе. Погрешно је чекати на завршетак прве, пре него што се покрене друга. То би значило непотребну секвенцијализацију ових операција. Битан ефекат је управо у томе што контролери обе периферије паралелно извршавају своје операције, чиме се добија на времену. Ипак, процесор не може истовремено преносити саме податке из оба контролера у меморију, већ се то обавља секвенцијално. Добитак на времену је у томе што саме периферије и њихови контролери раде паралелно и независно.

Улазно-излазни адресни простор меморијски пресликан, па се користе LOAD и STORE.

Инструкције за рад са 8-битним величинама се користе за преносе 8-битних вредности у/из меморије и регистара контролера периферија и када су 8-битне подаци довољни. Инструкције за рад са 16-битним величинама за рад са адресама и када 8-битне подаци нису довољни већ морају да се користе 16-битни подаци (величина блока података)

## 1.5 ЗАДАТАК

Једноадресни процесор са раздвојеним адресним просторима, меморија, периферије PER0 и PER1 повезани су системском магистралом са 16 адресних линија и 16 линија за податке. Адресирање је на нивоу 16 битних речи. Механизам прекида је векторисан а број улаза у IV табелу је одређен фиксно (0 за PER0, 1 за PER1). Адресе релевантних регистара су:

| | | | |
|---|---|---|---|
| PER0_CONTROL | FF00h | PER1_CONTROL | FF10h |
| PER0_STATUS | FF01h | PER1_STATUS | FF11h |
| PER0_DATA | FF02h | PER1_DATA | FF12h |

У управљачким регистрима бит 0 је Start којим се дозвољава почетак операције, бит 1 одређује смер операције (0-улаз, 1-излаз), бит 4 је Enable којим се дозвољава прекид, а у статусним регистрима бит 0 је Ready који сигнализира спремност контролера. Написати главни програм и одговарајућу прекидну рутину којима се: упоредо врши учитавање низа A(i) (i=0…FF) са PER0 у меморијски блок који почиње од адресе 1000h, и низа B(i) (i=0,..FF) са PER1 у меморијски блок почев од адресе 1100h, затим изврши сабирање унетих низова (B(i) = A(i) + B(i)) и резултујући низ шаље на периферију PER0. Улаз са PER0 реализовати коришћењем механизма прекида, улаз са PER1 реализовати испитивањем бита спремности, а излаз на PER0 коришћењем механизма прекида. Контролни регистар PER0 може да се чита.

**Решење**

Главни програм:

```
        LOAD #100h
        STORE MemCnt0
        STORE MemCnt1
        LOAD #1000h
        STORE MemAdr0
        LOAD #1100h
        STORE MemAdr1
        LOAD #0
        STORE MemSem0
        LOAD #0 ; smer: ulaz
        STORE MemDir0
        LOAD #11h
        OUT FF00h
        LOAD #01h
        OUT FF10h
Crdy:   IN FF11h
        AND #01h
        JZ Crdy
        IN FF12h
        STORE (MemAdr1)
        INC MemAdr1
        DEC MemCnt1
        JNZ Crdy
        LOAD #0
        OUT FF10h
Wait:   LOAD MemSem0
        CMP #1
        JNZ Wait
        LOAD #100h
        STORE MemCnt
        LOAD #1000h
        STORE MemA
        LOAD #1100h
        STORE MemB
Loop:   LOAD (MemA)
        ADD (MemB)
        STORE (MemB)
        INC MemA
        INC MemB
        DEC MemCnt
        JNZ Loop

        LOAD #100h
        STORE MemCnt0
        LOAD #1100h
        STORE MemAdr0
        LOAD #0
        STORE MemSem0
        LOAD #1   ; smer: izlaz
        STORE MemDir0
        LOAD #13h
        OUT FF00h
        …
Wait1:  LOAD MemSem0
        CMP #1
        JNZ Wait1
        HALT
```

```
Прекидна рутина:
        PUSH
        LOAD MemDir0      ; koji smer?
        CMP #1
        JZ Out
        ; input
        IN FF02h
        STORE (MemAdr0)
Inc:    INC MemAdr0
        DEC MemCnt0
        JNZ Back
        LOAD #1
        STORE MemSem0
        LOAD #0
        OUT FF00h
        JMP Back
        ;output
Out:    LOAD (MemAdr0)
        OUT FF02h
        JMP Inc
Back:   POP
        RTI
```

## 1.6 ЗАДАТАК

Једноадресни процесор са раздвојеним адресним просторима, периферије PER0 и PER1, и меморија повезани су магистралом са 16 адресних линија и 16 линија за податке. Адресирање је на нивоу 16 битних речи. Адресе релевантних регистара дате су на слици.

| | | | |
|---|---|---|---|
| PER0_CONTROL | FF00h | PER1_CONTROL | FF10h |
| PER0_STATUS | FF01h | PER1_STATUS | FF11h |
| PER0_DATA | FF02h | PER1_DATA | FF12h |

Бит 0 контролних регистара је Start бит, бит 1 дефинише смер операције (0—улаз, 1—излаз), а бит 2 је Enable којим се омогућује прекид. Бит 4 статусних регистара је Ready бит. Написати програм и одговарајуће прекидне рутине којима се реализује: учитавање низа A(i), i = 0, …, 999 са PER0 у меморијски блок који почиње од адресе 2000h и слање квадрираног низа (A(i)*A(i)) на PER1. Пријем са PER0 и слање на PER1 реализовати упоредо, тј. омогућити слање елемента низа на PER1 чим је то могуће. Улаз реализовати механизмом прекида, а излаз испитивањем бита спремности.

**Решење**

```
        Главни програм:
        LOAD #2000h       ;početna adresa ulaznog bafera
        STORE MemWP       ;pokazivač koji prati učitavanje sa PER0
        STORE MemRP       ;pokazivač koji prati slanje na PER1
        LOAD #1000        ;broj elemenata niza
        STORE CntW
        STORE CntR
        LOAD #5
        OUT FF00h         ;start PER0
        LOAD #3
        OUT FF10h         ;start PER1
Chck:IN FF11h
        AND #10h
        JZ Chck
;provera da li postoji element spreman za slanje
        LOAD MemWP
        SUB  MemRP
        JLE Chck          ;skok ako je rezultat oduzimanja =< 0
        LOAD (MemRP)
        MUL (MemRP)       ;kvadriranje elementa niza
        OUT FF12h         ;slanje rezultujućeg elementa na PER1
        INC  MemRP
        DEC  CntR
```

```
        JNZ   Chck
        LOAD #0
        OUT FF10h           ;Stop PER1
        HALT
```

Прекидна рутина периферије PER0:

```
Per1: PUSH
        IN     FF02h
        STORE (MemWP)
        INC   MemWP
        DEC  CntW
        JNZ   Back
        LOAD #0
        OUT FF00h           ;Stop PER0
Back:POP
        RTI
```

## 1.7 ЗАДАТАК

Двоадресни процесор са 16 регистара опште намене поседује меморијски пресликан улазно/излазни адресни простор. Процесор, меморија, периферије PER0, PER1 и PER2 повезани су системском магистралом са 16 адресних линија и 16 линија за податке. Адресирање је на нивоу 16 битних речи. Адресе релевантних регистара су приказане на слици.

| | | | | | |
|---|---|---|---|---|---|
| PER0_CONTROL | FF00h | PER1_CONTROL | FF10h | PER2_CONTROL | FF20h |
| PER0_STATUS | FF01h | PER1_STATUS | FF11h | PER2_STATUS | FF21h |
| PER0_DATA | FF02h | PER1_DATA | FF12h | PER2_DATA | FF22h |

У управљачким регистрима бит 15 је Start којим се дозвољава почетак операције, бит 0 одређује смер операције (0—улаз, 1—излаз), бит 7 је Enable којим се дозвољава прекид, а у статусним регистрима бит 0 је Ready који сигнализира спремност контролера периферије. Написати главни програм и одговарајуће прекидне рутине којима се упоредо: врши учитавање низа A(i) (i = 0, …, 99) са PER0 у меморијски блок који почиње од адресе 1000, и низа B(i) (i = 0, …, 99) са PER1 у меморијски блок почев од адресе 2000, формирање резултујућег низа C(i) (C(i) = A(i) + B(i), i = 0, …, 99) који се смешта у меморијски блок почев од адресе 3000, и слање резултујућег низа C на периферију PER2. Формирање низа C тече упоредо са учитавањем, тј. чим се учита i-ти елемент низа A(B) формира се i-ти елемент низа C под условом да је учитан i-ти елемент низа B(A). Такође, слање низа C на PER2 треба започети пре него што је цео низ C формиран, тј. чим је неки елемент низа C формиран треба омогућити његово слање на PER2. Улаз са PER0 и PER1 реализовати коришћењем механизма прекида, а излаз на PER2 испитивањем бита спремности.

**Решење:**

```
        MOV R0,   #-1
        MOVmcA, R0
        MOV mcB, R0
        MOV mwcC, R0
        MOV R1,   R0    ;R1 is read pointer
        MOV FF00h, #8080h   ;start PER0
        MOV FF10h, #8080h   ;start PER1
        MOV FF20h, #8001h   ;start PER2
Wait:   MOV R0,   FF21h     ;read status
        AND R0,   #1
        JZ    Wait
        MOVR2, mwcC
        CMP R2, R1
        BLE  Wait
        INC  R1
        MOVFF22h, (R1)3000
        CMP R1,   #99
        JNZ  Wait
        MOV FF20h, #0
        HALT

Прекидна рутина за PER0:
        INTD
        PUSH R0
        PUSH R1
        PUSH R2
        MOVR0, FF02h
        MOVR1, mcA
        INC  R1
        MOV mcA, R1
        MOV(R1)1000, R0
        CMP R1, mcB
        JG    SkipA
        ADD R0,  (R1) 2000
        MOVR2, mwcC
        INC  R2
        MOVmwcC, R2
        MOV (R2) 3000, R0
SkipA:  CMP R1, #99
        JNZ  BackA
        MOVFF00h, #0
BackA:  POP R2
        POP  R1
        POP  R0
        RTI
Прекидна рутина за PER1:
        INTD
        PUSH R0
        PUSH R1
        PUSH R2
        MOVR0, FF12h
        MOVR1, mcB
        INC  R1
        MOVmcB, R1
        MOV (R1) 2000, R0
        CMP R1, mcA
```

```
        JG     SkipB
        ADD R0,  (R1) 1000
        MOVR2, mwcC
        INC   R2
        MOVmwcC, R2
        MOV (R2) 3000, R0
SkipB: CMP R1, #99
        JNZ   BackB
        MOV FF10h,      #0
BackB:POP R2
        POP   R1
        POP   R0
        RTI
```

## 1.8 ЗАДАТАК

Двоадресни процесор са раздвојеним I/O и меморијско адресним просторима везан је на 16 битну адресу и 8 битну магистралу података. На магистралу су везана два контролера периферије, чији се регистри налазе на следећим адресама: Data (0026h, 0028h), Control (0030h, 0032h) и Status (0020h, 0022h). Сви регистри су 8 битни, адресирање је бајтовско. У статусним регистрима бит 2 је Ready, а у управљачким регистрима бит 0 је Start. Написати програм који учитава вредности са обе периферије, техником испитивања бита Ready, и пристигле вредности сумира, посебно за сваку периферију у локацијама SUM1 и SUM2. Улаз се прекида када са било које периферије стигне вредност 0. Уређаји шаљу податке различитим брзинама, а податак који је пристигао треба учитати што пре.

**Решење**

```
        …
        MOV SUM1, #0
        MOV SUM2, #0
        OUT 30h, #1    ; Start PER1
        OUT 32h, #1    ; Start PER2
LOOP:   IN R0, 20h     ; Test PER1
        AND R0, #4
        JNZ PER1
TSTP2: IN R0, 22h      ; Test PER2
       AND R0, #4
       JZ LOOP
PER2:      IN R0, 28h ;Input from PER2
      OR R0, R0
      JZ END
      ADD SUM2, R0
      JMP LOOP
PER1:      IN R0, 26h ;Input from PER1
      OR R0, R0
      JZ END
      ADD SUM1, R0
      JMP TSTP2
END:OUT 30h, #0;     Stop PER1
      OUT 32h, #0;     Stop PER2
      …
```

## 1.9 ЗАДАТАК

Једноадресни процесор са меморијски мапираним улазно/излазним адресним простором, меморија, периферија PER0 (adrCR=FF10h, adrSR= FF11h, adrDR=FF12h), периферија PER1 (adrCR=FF20h, adrSR=FF21h, adrDR=FF22h) са придруженим контролором периферије DMA1 (adrCR=FF00h, adrSR=FF01h, adrDR=FF02h, adrCNT=FF03h, adrAs=FF04h, adrAd=FF05h) и периферија PER2 (adrCR=FF30h, adrSR=FF31h, adrDR=FF32h) повезани су системском магистралом са 16 битном адресном и 16 битном магистралом података. Адресирање је на нивоу 16 битних речи. У управљачким регистрима бит 0 је Start којим се дозвољава почетак операције, бит 1 одређује смер операције (0-улаз, 1-излаз), бит 2 је Enable којим се дозвољава прекид, а у статусним регистрима бит 4 је Ready који сигнализира спремност контролера. Бит 3 управљачког регистра DMA контролера задаје режим рада (0-блоковски (burst), 1-циклус по циклус (cycle stealine)).

а) Написати главни програм и одговарајуће прекидне рутине којима се обавља следећи пренос. Са периферије PER1 учитава се низ података A(i) дужине 80h и смешта у меморију почев од адресе 1000h коришћењем DMA контролера у byrst режиму рада, па се по завршетку преноса учита низ података B(i) исте дужине са периферије PER1 и смешта у меморију почев од адресе 2000 коришћењем DMA контролера у циклус по циклус режиму рада. По завршетку уноса низова врши се упоредо слање података на периферије PER0 и PER2, и то тако што се на бржу периферију шаље податак A(i)/B(i), а на спорију A(i)-B(i) (i=1, ..., 80h). Излаз на периферију PER0 реализовати коришћењем механизма прекида, а излаз на периферију PER2 испитивањем бита спремности.

б) Да ли процесор може приступити регистру података периферије PER1 током учитавања низа података A(i) са ове периферије? Образложити одговор.

**Решење:**

а) Главни програм:

```
        LOAD #0
        STORE MemSem
        STORE MemSem0
        STORE MC0
        STORE MC2
        LOAD #80h
        STORE FF03h
        LOAD #1000h
        STORE FF05h
        STORE MAA0
        STORE MAA2
        LOAD #1h
        STORE FF20h
        LOAD #5h
        STORE FF00h
Wait:   LOAD MemSem
        CMP #1
        JNZ Wait
        LOAD #0
        STORE MemSem
        LOAD #80h
        STORE FF03h
        LOAD #2000h
        STORE FF05h
        STORE MAB0
        STORE MAB2
        LOAD #1h
        STORE FF20h
        LOAD #13
        STORE FF00h
Wait2:  LOAD MemSem
        CMP #1
        JNZ Wait2
        LOAD #3h
        STORE FF30h
        LOAD #7h
        STORE FF10h
ChRd:   LOAD FF31h
        AND #10h
        JZ ChRd
        INTD
        LOAD MC2
        SUB MC0
        JL Skip
        LOAD (MAA2)
        DIV (MAB2)
        JMP IncMa
Skip:   LOAD (MAA2)
```

```
        SUB (MAB2)
IncMa:STORE FF32h
        INC MC2
        INTE
        INC MAA2
        INC MAB2
        LOAD MC2
        CMP #80h
        JNZ ChRd
        LOAD #0h
        STORE FF30h
        ...
ChPer0:LOAD MemSem0
        CMP #1
        JZ ChPer0
        HALT

DMA_Int: PUSH
      LOAD 0
      STORE FF20h
      STORE FF00h
      INC
      STORE MemSem
      POP
        RTI
PER0_Int:PUSH
        LOAD MC0
        SUB MC2
        JL Skip0
        LOAD (MAA0)
        DIV (MAB0)
        JMP IncMa0
Skip0:LOAD (MAA0)
        SUB (MAB0)
IncMa0:STORE FF12h
        INC MC0
        INC MAA0
        INC MAB2
        LOAD MC0
        CMP #80h
        JNZ Back
        LOAD #0h
        STORE FF10h
        INC
      STORE MemSem0
Back:POP
        RTI
```

б) НЕ. Процесор не може извршити циклус читања на магистрали јер DMA контролер ради у блоковском режиму и држи магистралу заузету до завршетка преноса целог блока података

## 1.10 ЗАДАТАК

Једноадресни процесор са раздвојеним адресним просторима, меморија, периферија PER0, периферије PER1 и PER2 са придруженим контролерима за директан приступ меморији DMA1 и DMA2, редом, повезани су системском магистралом са 16 битном адресном и 16 битном магистралом података. Адресирање је на нивоу 16 битних речи. Адресе релевантних регистара су:

| | | | |
|---|---|---|---|
| PER0_CONTROL | FF10h | PER1_DATA | FF22h |
| PER0_STATUS | FF11h | PER2_CONTROL | FF30h |
| PER0_DATA | FF12h | PER2_STATUS | FF31h |
| PER1_CONTROL | FF20h | PER2_DATA | FF32h |
| PER1_STATUS | FF21h | | |
| DMA1_CONTROL | FF00h | DMA2_CONTROL | FF06h |
| DMA1_ADDRESS | FF01h | DMA2_ADDRESS | FF07h |
| DMA1_COUNT | FF02h | DMA2_COUNT | FF08h |
| DMA1_DATA | FF03h | DMA2_DATA | FF09h |
| DMA1_STATUS | FF04h | DMA2_STATUS | FF0Ah |

У управљачким регистрима бит 0 је Start којим се дозвољава почетак операције, бит 1 одређује смер операције (0-улаз, 1-излаз), бит 2 је Enable којим се дозвољава прекид, а у статусним регистрима бит 4 је Ready који сигнализира спремност контролера. Бит 3 управљачког регистра DMA контролера задаје режим рада (0-блоковски (burst), 1-циклус по циклус (cycle stealine)). Написати главни програм и одговарајуће прекидне рутине којима се обавља следећи пренос. Са периферије PER0 се прихвата бесконачни низ података и упоредо прослеђује периферијама PER1 и PER2. На располагању су два бафера BP0 и BP1 који се пуне наизменично подацима са периферије PER0. Наиме, упоредо са пуњењем бафера BP0 одвија се пражњење бафера BP1 слањем података на периферије PER1 и PER2 и обратно, док се пуни бафер BP1 празни се бафер BP0 слањем података на периферије PER1 и PER2. Величине бафера су 100h речи а почетне адресе бафера BP0 и BP1 су 1000h и 1100h, редом. Улаз са периферије PER0 реализовати коришћењем механизма прекида, излаз на периферију PER1 коришћењем DMA контролера који ради у циклус-по-циклус режиму, а излаз на периферију PER2 коришћењем DMA контролера који ради у блоковском режиму.

**Решење**

```
; initialize input from PER0 (fill BP0)
        LOAD #0
        STORE Mfin
        LOAD #1000h
        SOTRE Ain
        LOAD #100h
        STORE Cntin
        LOAD #5
        OUT FF10h
            ; ...
Wait1:LOAD Mfin
        AND #1
        JZ Wait1
; initialize output from BP0 and input to BP1
Swap:LOAD #1000h
        STORE Aout
        LOAD #1100h
        STORE Ain
        LOAD #1
        STORE Dir
Init:   LOAD #0
        STORE Mfin
        STORE MFout1
        STORE MFout2
        LOAD #100
        STORE Cntin
; init DMAs
        LOAD Aout
        OUT FF01h
        OUT FF07h
        LOAD #100h
        OUT FF02h
        OUT FF08h
; start controllers
        LOAD #5
```

```
        OUT FF10h
        LOAD #Fh
        OUT FF00h
        LOAD #7
        OUT FF06h
        LOAD #3h
        OUT FF20h
        LOAD #3h
        OUT FF30h

;               ....
Wait2:LOAD Mfin
        AND MFout1
        AND MFout2
        JZ Wait2
        LOAD Dir
        AND #1
        JZ Swap
; initialize output from BP1 and input to BP0
        LOAD #1100h
        STORE Aout
        LOAD #1000h
        STORE Ain
        LOAD #0
        STORE Dir
        JMP Init
```

Прекидна рутине:

```
DMA1Int:PUSH
        LOAD #0
        OUT FF20h
        OUT FF00h
        INC
        STORE MFout1
        POP
        RTI

DMA2Int:PUSH
        LOAD #0
        OUT FF30h
        OUT FF06h
        INC
        STORE MFout2
        POP
        RTI

Per0Int:PUSH
          IN FF12h
          STORE (Ain)
          INC Ain
          DEC Cntin
          JNZ Back0
          LOAD #0
          OUT FF10h     ;stop PER0
          INC
          STORE MFin
Back0:POP
          RTI
```

## 1.11 ЗАДАТАК

Двоадресни процесор са меморијски раздвојеним улазно/излазним адресним простором, меморија, и периферије PER0, PER1, PER2 повезани су системском магистралом са 16 адресних линија и 16 линија за податке. Адресирање је на нивоу 16 битних речи. Адресе релевантних регистара су:

| | | | | | |
|---|---|---|---|---|---|
| PER0_CONTROL | F010h | PER1_CONTROL | F020 | PER2_CONTROL | F030h |
| PER0_STATUS | F011h | PER1_STATUS | F021 | PER2_STATUS | F031h |
| PER0_DATA | F012h | PER1_DATA | F022 | PER2_DATA | F032h |

У управљачким регистрима бит 0 је *Start* којим се дозвољава почетак операције, бит 3 одређује тип трансфера података (0-улаз, 1-излаз), бит 7 је *Enable* којим се дозвољава прекид, а у статусним регистрима бит 15 је *Ready* који сигнализира спремност контролера периферије.

Механизам прекида је векторисан, а број улаза у IV табелу је одређен фиксно и износи 1 и за PER1 и за PER2. Периферија PER2 има већи приоритет од периферије PER1.

Написати главни програм и одговарајуће прекидне рутине којима се са периферије PER0 прихвата бесконачни низ *a(i)* и упоредо шаљу одговарајући низови на периферије PER1 и PER2. Размена података између периферија се одвија преко кружног бафера, организованог у меморији са почетном адресом 1000h, капацитета 100h речи. Периферија PER0 смешта нове елементе низа у бафер, а на периферије PER1 и PER2 се шаље податак који је најдуже у баферу. Уколико је бафер пун, пристигли елемент низа се смешта на место податка који је најдуже у баферу. Улаз са PER0 реализовати испитивањем бита спремности, а излаз на PER1 и PER2 реализовати коришћењем механизма прекида.

**Решење:**

```
        MOV head, #1000h
        MOV tail, #1000h
        MOV cnt, #0h
        AND IMR, #FFF9h
        OUT F010h, #1
        OUT F020h, #89h
        OUT F030h, #89h
Wait:   IN R0, F011h
        AND R0, #8000h
        JZ Wait
        IN R0, F012h
        INTD
        MOV (head), R0
        OR IMR, #6h
        ADD head, #1
        CMP head, 1100h
        JNZ Skip0
        MOV head, 1000h
Skip0:  CMP cnt, #100h
        JZ Tail0
        ADD cnt, #1h
        JMP Skip1
Tail0:  ADD tail, #1h
        CMP tail, #1100h
```

```
        JNZ Skip1
        MOV tail, 1000h
Skip1:INTE
        JMP Wait
```

**Прекидна рутина за PER1 и PER2:**

```
        INTD
        PUSH R0
        IN R0, F031h
        AND R0, #8000h
        JNZ Per2
Per1: OUT F022h, (tail)
        JMP Tail1
Per2: OUT F032h, (tail)
Tail1:ADD tail, #1h
        CMP tail, 1100h
        JNZ Skip2
        MOV tail, 1000h
Skip2:SUB cnt, #1h
        JNZ Back
        AND IMR, #FFF9h
Back:POP R0
        INTE
        RTI
```

## 1.12 ЗАДАТАК

Двоадресни процесор са меморијски мапираним улазно/излазним адресним простором, меморија, и периферије PER0 и PER1 повезани су системском магистралом са 16 адресних линија и 16 линија за податке. Стек расте од виших ка нижим адресама, SP указује на последњу заузету локацију. Приликом позива подпрограма на стеку се чува само PC. Адресирање је на нивоу 16 битних речи. Адресе релевантних регистара су:

| | | | |
|---|---|---|---|
| PER0_CONTROL | F010h | PER1_CONTROL | F020h |
| PER0_STATUS | F011h | PER1_STATUS | F021h |
| PER0_DATA | F012h | PER1_DATA | F022h |

У управљачким регистрима бит 0 је *Start* којим се дозвољава почетак операције, бит 3 одређује тип трансфера података (0-улаз, 1-излаз), бит 7 је *Enable* којим се дозвољава прекид, а у статусним регистрима бит 15 је *Ready* који сигнализира спремност контролера периферије.

**а)** Написати подпрограм и одговарајуће прекидне рутине којима се са периферије PER0 прихвата бесконачни низ *a(i)* и шаље на периферију PER1. Размена података између периферија се одвија преко FIFO (First In First Out – податак који је пре уписан у бафер, треба да се пре и пошаље из бафера) бафера. Почетна адреса и капацитет бафера се прослеђују као аргументи подпрограма (прототип: **void** transferData(**void** * buffer, **int** size);). Захтев се периферији задаје следећом структуром: struct IOStruct {**void*** buffer; **int** size; **int** dir; **int*** memSem; };

Периферија PER0 смешта нови елемент низа у бафер, а периферија PER1 чита елементе из бафера. Улаз са PER0 реализовати испитивањем бита спремности, а излаз на PER1 коришћењем механизма прекида.
**б)** Колики је укупни број захтева за прекид које прими процесор током преноса првих 800h речи низа *a* са PER0 на PER1?

**Решење:**
**а)**
**Подпрограм:**

```
                PUSH BP
                MOV BP, SP
                PUSH R0
                PUSH R1
                PUSH R2

                MOV R0, struct0
                MOV (R0)0h, (BP)2h; // buffer
                MOV (R0)1h, (BP)3h; // size
                MOV Full, #0h
                MOV WrP, #0h

                MOV R1, struct1
                MOV (R1)0h, (BP)2h; // buffer
                MOV (R1)1h, (BP)3h; // size
                AND IMR, #FFFDh; //disabled PER1
                MOV Empty, #1h
                MOV RdP, #0h

                MOV F010h, #1h      ;// start PER0
                MOV F020h, #89h     ;// start PER1

      Check0:   TST F011h, #8000h
                JZ Check0

Wait_for_Place: CMP Full, #1h
                JZ Wait_for_Place; //wait(Full);
                MOV R2, (R0)0h
                ADD R2, WrP
                MOV (R2), F012h

                MOV Empty, #0h
                OR IMR, #2h      ;// enabled PER1

              ;WrP:=(WrP+1) mod bufferSize
                ADD WrP, #1h
                CMP WrP, (R0)1h
                JNZ Skip0
                MOV WrP, #0h
      Skip0:    INTD
                CMP WrP, RdP
```

```
            JNZ Skip1
            MOV Full, #1h
Skip1:      INTE
            JMP Check0
            MOV F010h, #0h
            MOV F020h, #0h
            POP R2
            POP R1
            POP R0
            POP BP
            RTS
```

**Прекидна рутина за PER1:**

```
IntPER1:    PUSH R0
            PUSH R1
            PUSH R2
            MOV R0, struct0
            MOV R1, struct1
            MOV R2, (R1)0h
            ADD R2, RdP
            MOV F022h, (R2)

                MOV Full, #0h
                ;RdP:=(RdP+1) mod bufferSize
                ADD RdP, #1h
                CMP RdP, (R1)1h
                JNZ Skip2
                MOV RdP, #0h
          Skip2: CMP RdP, WrP
                JNZ Back
                MOV Empty, #1h
                AND IMR, #FFFDh; //disabled PER1
          Back:  POP R2
                POP R1
                POP R0
                RTI
```

**б)** 800h пута

## 1.13 ЗАДАТАК

Двоадресни процесор са меморијски раздвојеним улазно/излазним адресним простором, меморија, и периферије PER0, PER1 и PER2 повезани су системском магистралом са 16 адресних линија и 16 линија за податке. Стек расте од виших ка нижим адресама, SP указује на последњу заузету локацију. Приликом позива подпрограма на стеку се чува само PC. Адресирање је на нивоу 16 битних речи. Адресе релевантних регистара су:

| | | | | | |
|---|---|---|---|---|---|
| PER0_CONTROL | F010h | PER1_CONTROL | F020 | PER1_CONTROL | F030h |
| PER0_STATUS | F011h | PER1_STATUS | F021 | PER1_STATUS | F031h |
| PER0_DATA | F012h | PER1_DATA | F022 | PER1_DATA | F032h |

У управљачким регистрима бит 0 је *Start* којим се дозвољава почетак операције, бит 3 одређује тип трансфера података (0-улаз, 1-излаз), бит 7 је *Enable* којим се дозвољава прекид, а у статусним регистрима бит 15 је *Ready* који сигнализира спремност контролера периферије.

Написати главни програм и одговарајуће прекидне рутине којима се са периферије PER0 прихвата низ од num података учитава *a(i)* и шаље на периферије PER1 и PER2. Размена података између периферија се одвија преко бафера, организованог у меморији. Периферија PER0 смешта нови елемент низа у бафер када има слободног места, на периферију PER1 послати елемент који је најдуже био у баферу, а на периферију PER2 елемент који је најкраће био у баферу. Улаз са PER0 реализовати испитивањем бита спремности, а излазе коришћењем механизма прекида.
Почетна адреса капацитет бафера, и број података које је потребно обрадити се прослеђују као аргументи подпрограма (прототип: **void** transferData(**void** *buffer, **int** size, **int** num);). Захтев се периферији задаје следећом структуром: **struct** IOStruct {**void*** buffer; **int** size; **int** dir; **int** memSem; };

**Решење:**

```
PUSH BP
MOV BP, SP
PUSH R0
PUSH R1
PUSH R2
PUSH R3
PUSH R4

MOV cnt0, #0h
MOV R0, struct0
MOV (R0)0h, (BP)2h; // buffer
MOV (R0)1h, (BP)3h; // size
MOV (R0)3h, 0h; // Sem0 = #0
MOV WrP, #0

MOV cnt1, #0h
MOV R1, struct1
MOV (R1)0h, (BP)2h; // buffer
MOV (R1)1h, (BP)3h; // size
MOV (R1)3h, 0h; // Sem1 = #0
AND IMR, #FFFDh; //disabled PER1
MOV RdP, #0h

MOV cnt2, #0h
```

```
            MOV R2, struct2
            MOV (R2)0h, (BP)2h;     // buffer
            MOV (R2)1h, (BP)3h;     // size
            MOV (R2)3h, 0h;   // Sem2 = #0
            AND IMR, #FFFBh;       //disabled PER2

            MOV cnt, #0h
            MOV cntx, #0h
            MOV num, (BP)4h

            OUT F010h, #1h    ;       //start PER0
            OUT F020h, #89h;  //start PER1
            OUT F030h, #89h;  //start PER2

      Check0:IN R4, F011h
            AND R4, #8000h
            JZ Check0

Wait_for_Place:  CMP (R0)1h, cnt
            JZ Wait_for_Place;
            INTD
            MOV R3, (R0)0h
            ADD R3, WrP
            IN (R3), F012h
            ADD cnt, #1h
            ADD cnt0, #1h
            OR IMR, #6h;    //enabled PER1, enabled PER2

            ;WrP:=(WrP+1) mod bufferSize
            ADD WrP, #1h
            CMP WrP, (R0)1h
            JNZ Skip0
            MOV WrP, #0h
Skip0:      INTE
            CMP num, cnt0h
            JNZ Check0
            OUT F010h, #0h

Wait1:      CMP (R1)3h, #1h
            JNZ Wait1
Wait2:      CMP (R2)3h, #1h
            JNZ Wait2

            POP R4
            POP R3
            POP R2
            POP R1
            POP R0
            POP BP
            RTS
```

**Prekidna rutina za PER1:**

```
IntPER1:  PUSH R0
          PUSH R1
          PUSH R2
          MOV R0, struct2
          MOV R1, struct1

          MOV R2, (R1)0h
          ADD R2, RdP
          OUT F022h, (R2)

          INTD
          SUB cnt, #1h
          ADD cnt1, #1h
          ADD cntx, #1h

          ;RdP:=(RdP+1) mod bufferSize
          ADD RdP, #1h
          CMP RdP, (R1)1h
          JNZ Skip1
          MOV RdP, #0h

Skip1:    CMP num, cntx
          JNZ Ret1
          OUT F020h, #0h
          MOV (R1)3h, #1h
          OUT F030h, #0h
          MOV (R0)3h, #1h
          JMP Back1
Ret1:     CMP cnt, #0h
          JNZ Back1
          AND IMR, #FFF9h;
Back1:    POP R2
          POP R1
          POP R0
          RTI
```

**Prekidna rutina za PER2:**

```
IntPER2:  PUSH R0
          PUSH R1
          PUSH R2
          MOV R0, struct1
          MOV R1, struct2

          INTD
          ;WrP:=(WrP -1) mod bufferSize
          CMP WrP, #0h
          JN Rol2
          SUB WrP, #1h
```

```
            JMP Skip2
Rol2:       MOV WrP, (R1)3h

Skip2:      MOV R2, (R1)0h
            ADD R2, WrP
            OUT F032h, (R2)

            SUB cnt, #1h
            ADD cnt2, #1h
            ADD cntx, #1h

            CMP num, cntx
            JNZ Ret2
            OUT F020h, #0h
            MOV (R1)3h, #1h
            OUT F030h, #0h
            MOV (R0)3h, #1h
            JMP Back2
Ret2:       CMP cnt, #0h
            JNZ Back2
            AND IMR, #FFF9h;
Back2:      POP R2
            POP R1
            POP R0
            RTI
```

# 2 ЛИТЕРАТУРА

1. B. Lazić, *Logičko projektovanje računara*, Nauka—Elektrotehnički fakultet, Beograd, 1994.

2. D. Živković, M. Popović, *Impulsna u digitalna elektronika*, Nauka—Elektrotehnički fakultet, Beograd, 1992.

3. J. Djordjevic, A. Milenkovic, N. Grbanovic, "*An Integrated Educational Environment for Teaching Computer Architecture and Organisation*, " IEEE MICRO, May 2000.pp. 66-74.

4. J. Djordjevic, M. R. Barbacci, B. Hosler, *A PMS Level Notation for the Description and Simulation of Digital Systems*, The Computer Journal, Vol. 28, No. 4, pp. 357-365, 1985.

5. S. Miladinović, J. Đorđević, A. Milenković, *Programski sistem za grafički opis u simulaciju digitalnih sistema*, Zbornik radova ETRAN 1997, Zlatibor, Jugoslavija, Jun 1997.

6. N. Grbanovic, J. Djordjevic, B. Nikolić, *The Software Package for an Educational Computer System*, International Journal он Electrical Engineering Education, Vol. 40, No. 4, Oct 2003, pp. 270-284.

7. J. Djordjevic, A. Milenkovic, I. Todorovic, D. Marinov, "CALKAS: *A Computer Architecture Learning and Knowledge Assessment System*, " IEEE TC Computer Architecture Newsletter, March 1999.

8. J. Đorđević, *Priručnik из arhitekture računara*, Elektrotehnički fakultet, Beograd, 1997.

9. J. Đorđević, *Priručnik из arhitekture u organizacije računara*, Elektrotehnički fakultet, Beograd, 1997.

10. J. Đorđević, *Arhitektura računara*, *Edukacioni računarski sistem*, *Arhitektura u organizacija računrskog sistema,* Elektrotehnički fakultet, Beograd, 2002.

11. J. Đorđević, N. Grbanović, B. Nikolić, Z. Radivojević, *Arhitektura računara*, *Edukacioni računarski sistem*, *Priručnik za simulaciju ca zadacima,* Elektrotehnički fakultet, Beograd, 2004.

12. J. Djordjevic, B. Nikolic, A. Milenkovic, "*Flexible Web-based Educational System for Teaching Computer Architecture and Organization*, " IEEE, Transactions он Education, Vol. 48, No. 2, 2005.

13. J. Djordjevic, B. Nikolic, M. Mitrovic, "*A Memory System for Education*, " Computer Journal, (to appear)