АРХИТЕКТУРА РАЧУНАРА

Верзија 2012 1.1

САДРЖАЈ

Прекиди

Задатак 1.

Посматра се процесор са векторисаним механизмом прекида. Адресе прекидних рутина се чувају у табели адреса прекидних рутина (IV табели) која почиње на адреси 0 и има 256 улаза. Капацитет оперативне меморије је 64KB, ширина речи меморије је један бајт, а 16 битни подаци се у меморију смештају тако да се на нижој адреси налази виши бајт, а на вишој адреси нижи бајт. Прекидне рутине за периферије PER1, PER2 и PER3 почињу на адресама 5678h, 3456h и 1234h, респективно. Улази 5, 3 и 1 у IV табели су додељени периферијама PER1, PER2 и PER3, респективно, а периферије су иницијализоване да као импулс шаљу захтев за прекидом, као и да шаљу број улаза у табели прекидних рутина.

а) Нацртати део оперативне меморије на којима се налазе улази 0 до 5 у IV табели, означити адресе релевантних меморијских локација и попунити их одговарајућим вредностима.

б) Набројати корак по корак шта се све дешава у процесору од тренутка када је стигао захтев за прекид од периферије до тренутка када се у регистру PC налази почетна адреса прекидне рутине.

в) Објаснити за сваки корак набројан у тачки б) да ли га обавља хардвер или софтвер.

г) Објаснити како се обавља повратак из прекидне рутине и набројати кораке који се том приликом изводе.

Решење:

а) Садржај дела оперативне меморије је приказан на слици.

број улаза у IV табелу	меморијска адреса	садржај
5 for PER1	11	78h
	10	56h
4	9	
	8	
3 for PER2	7	56h
	6	34h
2	5	
	4	
1 for PER3	3	34h
	2	12h
0	1	
	0	
Слика.1	1	
	0	

Слика.1.

б) Када стигне захтев за прекид најпре се заврши са извршавањем текуће инструкције. Након тога се наставља са извршавањем додатних корака који су потребни да би се ① сачувао контекст процесора и ② израчунала адреса прекидне рутине. Контекст процесора (PC, PSW и програмски доступни регистри) се чува на врху стека. Адреса прекидне рутине се рачуна следећом секвенцом акција:

процесор шаље **inta** сигнал (interrupt acknowledge) периферији,

периферија одговара тако што процесору пошаље свој број улаза у IV табелу,

процесор израчуна адресу улаза у IV табелу (број улаза се претвори у померај и дода се на садржај IVTP регистра) и

прочита се адреса прекидне рутине из IV табеле и смести се у PC.

Треба додати да веома често након корака ① и ② процесор обавља и корак ③. У овом кораку:

бит I (маскирајући прекиди дозвољени) у регистру PSW се ресетује,

бит T (прекид после сваке инструкције) у регистру PSW се ресетује и

у битове L (приоритет текућег програма) у регистру PSW се уписује приоритет прекидне рутине на коју се скаче.

в) Одговор у тачки б) подразумева да се сви кораци изводе хардверски. Међутим, неко ид корака под ① могу да се изведу софтверски. Ово је случај код програмски доступних регистара и понекад код регистра PSW које у том случају треба програмски сачувати на почетку прекидне рутине. Програмски доступни регистри се обично чувају софтверски ако их је много. У супротном, они се чувају хардверски. Регистар PSW се мора чувати хардверски ако постоји корак ③. У супротном, може се чувати или хардверски или софтверски. У већину случајева регистар PSW се чува хардверски. Кораци ② и ③ се увек изводе хардверски.

г) Повратак из прекидне рутине се изводи посебном инструкцијом **RTI** (return from interrupt). Ова инструкција рестаурира са стека контекст процесора:

програмски доступни регистри се рестаурирају ако су сачувани хардверски,
регистар PSW се рестаурира ако је сачуван хардверски и
регистар PC је рестауриран.

Ако се програмски доступни регистри и PSW не рестаурирају инструкцијом **RTI**, морају се рестаурирати софтверски на крају прекидне рутине пре инструкције **RTI**.

Задатак 2.

Меморија неког рачунара је капацитета 4G (giga) 16 битних речи. Адресибилна јединица је 16 битна реч, а 32 битни бројеви се у меморију смештају тако да је на нижој адреси нижих 16 бита. Процесор је једноадресни, улазно-излазни и меморијски адресни простори су раздвојени, а механизам прекида је векторисан. Интерапт вектор (*IV*) табела заузима најнижи део меморијског адресног простора. На процесор су везане три периферије, PER1, PER2 и PER3 којима треба доделити улазе 3, 5 и 7 у вектор табели, и којима одговарају прекидне рутине на адресама 12345678h, 23456789h и 3456789Ah, респективно. Адресе 16 битних регистара у којима се чувају бројеви улаза су 0h, 4h и 8h, респективно.

а) Написати део програма којим се додељују бројеви улаза наведеним периферијама.

б) Нацртати изглед првих 8 улаза у вектор табели, означити адресе релевантних локација и уписати садржаје у њих.

в) Написати део програма којим се иницијализује улаз 5 у вектор табели.

г) Коју вредност шаље процесору периферија PER2 када јој процесор одобри захтев за прекидом?

д) Описати и представити програмом поступак којим се и периферији PER2 додељује иста прекидна рутина као и периферији PER3.

ђ) Колико улаза има IV табела?

Решење

а)
```
LOAD  #3
OUT   0h
LOAD  #5
OUT   4h
LOAD  #7
OUT   8h
```

в)
```
LOAD  #2345h
STORE 11
LOAD  #6789h
STORE 10
```

г) 5

д)
```
LOAD  #3456h
STORE 11
LOAD  #789Ah
STORE 10
```
ili:
```
LOAD  #7
OUT   4h
```

ђ) 64K улаза

б)

7	PER3	15	3456h
		14	789Ah
6		13	
		12	
5	PER2	11	2345h
		10	6789h
4		9	
		8	
3	PER1	7	1234h
		6	5678h
2		5	
		4	
1		3	
		2	
0		1	
		0	

Задатак 3.

Адресни простор неког рачунара је величине 16GB (giga бајта). Адресибилна јединица је 32 битна реч, а вишеречни бројеви се у меморију смештају тако да је на нижој адреси нижа реч. Процесор је једноадресни, улазно-излазни и меморијски адресни простори су раздвојени, а механизам прекида је векторисан. IV (*Interrupt Vector*) табела има 4 улаза и почиње од адресе 0. Процесор поседује два улаза за спољне маскирајуће прекиде, IRQ0 и IRQ1, при чему је IRQ1 вишег приоритета, на које су везане периферије PER0 и PER1, респективно. У PSW постоји бит I (*Interrupt Enable)* у разреду 1 који се брише у корака за обраду прекида и бит L у разреду 0 који садржи ниво приоритета текућег извршавања. Не постоји селективно маскирање прекида. Процесор прихвата прекид истог нивоа као што је текући. Инструкције INTE и RTI не реагују на прекид. При прекиду се на стеку чувају PC и PSW тим редом. Адресе регистара у којима се чувају бројеви улаза PER0 и PER1 су 02h и 04h, респективно. Садржај дела меморијског адресног простора почев од адресе 0 дат је на Слици 1. Прекидна рутина за PER0 дата је на Слици 2, а за PER1 на Слици 3. Посматра се следећи сценарио извршавања: у току извршавања инструкције `INC` дужине једне речи на адреси 100h главног програма стиже захтев за прекид од PER0 и прихвата се. Инструкција на адреси A00Ah означена је као 1. (прва) по редоследу извршавања, а свака следећа инструкција која се извршава означена је следећим

редним бројем. У току извршавања 2. инструкције стиже захтев за прекид од PER1, у току 6. поново захтев од PER0. Све вредности на сликама су хексадецималне. Напомена на адреси 101h се налази инструкције AND #1 дужине једне речи, Почетни садржај акумулатора је 0. Стек расте од виших ка нижим адресама, а SP указује на прву слободну локацију.

Слика 1

Адреса	Садржај
0	000A
1	000A
2	0A0A
3	A00A
4	000A
5	00A0
6	000A
7	000A

Слика 2

Адреса	Инструкција
A00A	INTE
A00B	INC
A00C	RTI

Слика 3

Адреса	Инструкција
0A0A	POP
0A0B	OR #1
0A0C	PUSH
0A0D	RTI

а) По којим линијама периферија PER1 шаље број улаза у IV табелу?
б) Која вредност се налази на магистрали података у циклусу из тачке а)?
в) Написати секвенцу инструкција којом се периферији PER0 додељује број улаза у IVT.
г) Дати секвенцу адреса инструкција које се редом извршавају по датом сценарију. Резултат дати табеларно тако да табела садржи редни број интрукције, адресу на којој започиње инструкција, саму инструкцију, садржај акумулатора након извршење инструкције, вредности свих познатих бита унутар програмске статусне речи.
д) Приказати садржај свих познатих локација на врху стека непосредно после извршавања 5. инструкције. За сачувану вредност PSW дати само вредност два бита најмање тежине. Означити врх стека.

Решење

а) Ова периферија шаље преко магистрале података свој број улаза у IVT.
б) 2
в) `LOAD #3; OUT 2`
г)

Рб	Адреса	Инструкција	ACC	Стек	I	L	PRIRR1	PRIRR0
0	0100	INC	1	-	1	0	-	1
			1	1	0	0	-	-
1	A00A	INTE	1	1	1	0	-	-
2	A00B	INC	2	1	1	0	1	-
			2	2	0	1	-	-
3	0A0A	POP	XX10b	3	0	1	-	-
4	0A0B	OR #1	XX11b	3	0	1	-	-
5	0A0C	PUSH	XX11b	4	0	1	-	-
6	0A0D	RTI	XX11b	1	1	1	-	1
7	A00C	RTI	XX11b	-	1	0	-	1
8	0101	AND #1	1	-	1	0	-	1
			1	5	0	0	-	-
9	A00A	INTE	1	5	1	0	-	-
10	A00B	INC	2	5	1	0	-	-
11	A00C	RTI	2	-	1	0	-	-
12	0102							

1

0101
I=1, L=0

2

0101
I=1, L=0
A00C
I=1, L=0

3

0101
I=1, L=0
A00C

4

0101
I=1, L=0
A00C
I=1, L=1

5

0102
I=1, L=0

д)

101h
XX10b

	A00Ch
	XX11b
Врх стека->	

Задатак 4.

Меморија неког рачунара је капацитета 64 KB. Адресибилна јединица је бајт, а 16 битни бројеви се у меморију смештају тако да је на нижој адреси виши бајт. Процесор је једноадресни и има наредбе које обављају пренос само једног бајта ка/из меморије, улазно/излазни и меморијски адресни простори су раздвојени, а механизам прекида је векторисан. Интерапт вектор табела има 8 улаза и почиње од адресе 0 оперативне меморије. Процесор има 4 улазне линије IRQ_0 до IRQ_3 за маскирајуће прекиде који се приоритирају, при чему је улаз IRQ0 највишег приоритета. Овим линијама додељени су фиксно улази 4 до 7 у вектор табели, респективно. Немаскирајући и интерни прекиди су вишег приоритета од маскирајућих, и заузимају улазе 0 до 3 у табели. У PSW процесора постоји бит I (*Interrupt Enable)* који се брише у кораку за обраду прекида, а програмски контролише наредбама INTE (*Interrupt Enable*, поставља I на 1) и INTD (*Interrupt Disable*, брише I), као и бити $L_{2...0}$ који садрже ниво прекида који се тренутно опслужује. Такође постоји 4 битни регистар IMR (*Interrupt Mask Register*) код кога јединица на биту *i* значи да је дозвољен прекид са линије IRQ*i*. Овај регистар иницијално садржи све јединице. На линије IRQ1 и IRQ3 су везане периферије PER1 и PER3, респективно, а остале линије су слободне. Треба везати још уређај T—временски бројач који периодично генерише сигнал захтева за прекид. Потребно је обезбедити да се прекид од временског бројача може опслужити и у току извршавања неке од прекидних рутина за периферије PER1 и PER3. Ове рутине почињу на адресама 100h и 300h, респективно.

а) На који улаз $IRQ_{3...0}$ треба везати сигнал захтева за прекид временског бројача?

б) Написати део програма којим се иницијализује улаз периферије PER1 у вектор табели.

в) Прекидна рутина периферије PER1 је приказана на слици.

```
INTH1: IN      FF00h
       STORE (Dest)+
       …
       RTI
```

Ако у току извршавања инструкције STORE стигне сигнал прекида од временског бројача, да ли ће се овај прекид одмах опслужити? Образложити одговор.

г) Изменити дату прекидну рутину периферије PER1 тако да одговор на претходно питање буде супротан.

д) Шта треба урадити у прекидној рутини периферије PER3 тако да се обезбеди да она не буде прекидана од захтева са периферије PER1, али да буде прекидана од захтева са временског бројача?

Решење

а) IRQ0

б)

```
LOAD   #01
STORE  0Ah
LOAD   #00
STORE  0Bh
```

в) Не.

г)

```
INTH1: INTE
       IN      FF00h
       STORE (Dest)+
       …
       RTI
```

д) Постављања бита I на 1, у IMR треба уписати bb01 (бинарно, b означава било коју вредност).

Задатак 5.

Адресни простор процесора је величине 16GB, адресибилна јединица је 32 битна реч, а 64 битни бројеви се смештају тако да је на нижој адреси нижа реч. Процесор је једноадресни, а механизам прекида је векторисан. Интерапт вектор (IV) табела има 4 фиксна улаза и почиње од адресе 2h. Процесор има једну улазну линију IRQM за спољне маскирајуће прекиде и једну улазну линију IRQN за спољне немаскирајуће прекиде. Њима су придружени улази 0 и 1 у IV табелу, респективно. Немаскирајући прекиди су вишег приоритета од маскирајућих. Улаз 2 у IV табели се употребљава у случају прекида после сваке инструкције (TRAP), а улаз 3 у свим осталим случајевима. PSW постоји бит I (*Interrupt Enable)* који се брише у кораку за обраду прекида. При прекиду се на стеку чувају PSW и PC тим редом. Стек расте према нижим локацијама. Акумулатор је 32 битни. Инструкције INTE, INTD, RTI, TRPE и TRPD не реагују на прекиде. Дат је део главног програма на слици 1, прекидне рутине на слици 2, изглед дела меморије почев од адресе 0 дат је на слици 3. Инструкција INTE на адреси 0100h означена је као 1. (прва) по редоследу извршавања, а свака следећа инструкција која се извршава означена је следећим редним бројем. У току

извршавања 2. инструкције стиже захтев за прекид по линији IRQN, а у току 5. по линији IPQM. На почетку су сви бити PSW-а постављени на 0.

Slika 1

Адреса	Наредба
0100h	INTE
0101h	LOAD #1h
0102h	INC
0103h	STORE #1h
0104h	DEC
0105h	INTD

Слика 2

Адреса	Наредба	Адреса	Наредба
1000h	INC	1006h	STORE 1h
1001h	PUSH	1007h	RTI
1002h	POP	1008h	POP
1003h	RTI	1009h	INC
1004h	LOAD 1h	100Ah	PUSH
1005h	OR #FFh	100Bh	RTI

Слика 3

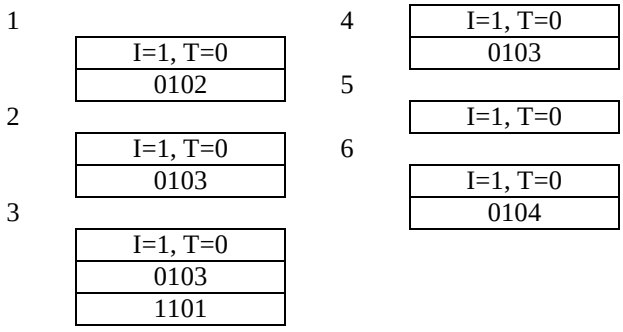
Адреса	Садржај
0000h	1001h
0001h	100Ah
0002h	1000h
0003h	1004h
0004h	1006h
0005h	1008h

а) На којим адресама започињу прекидне рутине за линије IRQM и IRQN, респективно?
б) Написати секвенцу адреса наредби које се редом извршавају, почев од адресе 0100h. Резултат дати табеларно тако да табела садржи редни број интрукције, адресу на којој започиње инструкција, саму инструкцију, садржај акумулатора након извршење инструкције, вредности свих познатих бита унутар програмске статусне речи.
в) Приказати садржај свих познатих локација на врху стека након извршавања 7. инструкције. За сачувану вредност PSW дати само вредност бита I. Назначити у коме смеру расте стек.
г) Која ће се вредност налазити на локацији 1h након извршења секвенце под б)?

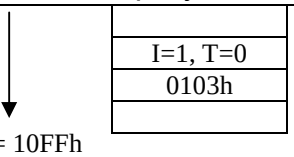
Решење

а) IRQM – 1000h, IRQN – 1004h
б) 0100h, 0101h, 1004h, 1005h, 1006h, 1007h, 0102h, 1000h, 1001h, 1002h, 1003h, 0103h, 1008h, 1009h, 100Ah, 100Bh, 0104h, 0105h

Рб	Адреса	Инструкција	ACC	Стек	I	T	PRIRRN	PRIRRM
0	-	-	?	-	0	0	-	-
1	0100	INTE	?	-	1	0	-	-
2	0101	LOAD #1	1	-	1	0	1	-
			1	1	0	0	-	-
3	1004	LOAD 1	100A	1	0	0	-	-
4	1005	OR #FFh	10FF	1	0	0	-	-
5	1006	STORE 1	10FF	1	0	0	-	1
6	1007	RTI	10FF	-	1	0	-	1
7	0102	INC	1100	-	1	0	-	1
			1100	2	0	0	-	-
8	1000	INC	1101	2	0	0	-	-
9	1001	PUSH	1101	3	0	0	-	-
10	1002	POP	1101	2	0	0	-	-
11	1003	RTI	1101	-	1	0	-	-
12	0103	STORE #1	1101	4	0	0	-	-
13	1008	POP	0103	5	0	0	-	-
14	1009	INC	0104	5	0	0	-	-
15	100A	PUSH	0104	6	0	0	-	-
16	100B	RTI	0104	-	1	0	-	-
17	0104	DEC	0103	-	1	0	-	-
18	0105	INTD	0103	-	0	0	-	-



в)



г) MEM[1] = 10FFh

Задатак 6.

Адресни простор процесора је величине 16GB, адресибилна јединица је 32 битна реч, а 64 битни бројеви се смештају тако да је на нижој адреси нижа реч. Процесор је једноадресни, а механизам прекида је векторисан. Интерапт вектор (IV) табела има 4 фиксна улаза и почиње од адресе 2h. Процесор има две улазне линију IRQM1 и IRQM2 за спољне маскирајуће прекиде, при чему је IRQM2 вишег приоритета, и једну улазну линију IRQN за спољне немаскирајуће прекиде. Њима су придружени улази 0, 1 и 2 у IV табелу, респективно. Немаскирајући прекиди су вишег приоритета од маскирајућих. Улаз 3 се користи у свим осталим случајевима. У PSW-у постоје бити I (*Interrupt Enable)* и T (*Trap*) који се бришу у кораку за обраду прекида, као и одређен број L бита. При прекиду се на стеку чувају PSW и PC тим редом. Стек расте према нижим локацијама. Акумулатор је 32 битни. Инструкције INTE, INTD, RTI, TRPE и TRPD не реагују на прекиде. Не прихвата се прекид истог нивоа приоритета. Не постоји регистар маске IMR. Дат је део главног програма на слици 1, прекидне рутине на слици 2, изглед дела меморије почев од адресе 0 дат је на слици 3. Инструкција TRPE на адреси 0100h означена је као 1. (прва) по редоследу извршавања, а свака следећа инструкција која се извршава означена је следећим редним бројем. У току извршавања 2. инструкције стиже захтев за прекид по линији IRQM1, у току 5. по линији IRQN, а у току 9. по линији IRQM2. На почетку су сви бити PSW-а постављени на 0.

Слика 1

Адреса	Наредба
0100h	TRPE
0101h	INTE
0102h	LOAD #1h
0103h	INTD
0104h	STORE 1h
0105h	TRPD

Слика 2

Адреса	Наредба	Адреса	Наредба
1000h	INC	1006h	INC
1001h	RTI	1007h	RTI
1002h	POP	1008h	LOAD 1h
1003h	PUSH	1009h	INC
1004h	DEC	100Ah	INC
1005h	RTI	100Bh	RTI

Слика 3

Адреса	Садржај
0000h	1002h
0001h	1008h
0002h	1009h
0003h	1006h
0004h	1004h
0005h	1000h

а) На којим адресама започињу прекидне рутине за линије IRQM1, IRQM2 и IRQN, респективно?
б) Написати секвенцу адреса наредби које се редом извршавају, почев од адресе 0100h. Резултат дати табеларно тако да табела садржи редни број интрукције, адресу на којој започиње инструкција, саму инструкцију, садржај акумулатора након извршење инструкције, вредности свих познатих бита унутар програмске статусне речи.
в) Приказати садржај свих познатих локација на врху стека након извршавања 6. инструкције. За сачувану вредност PSW дати само вредности бита I, T и L. Назначити у коме смеру расте стек.
г) Која ће се вредност налазити на локацији 1h након извршења секвенце под б)?
д) Нацртати структурну шему мреже за разрешавање захтева за маскирајућим прекидом у процесору. Ова мрежа треба да генерише сигнал INTRQ логичког услова који говори да постоји маскирајући прекид који треба опслужити, а помоћу вредности бита из регистара PSW и вредности са улаза IRQM1 и IRQM2.

Решење

а) IRQM1 – 1009h, IRQM2 – 1006h, IRQN – 1004h
б) 0100h, 0101h, 0102h, 1009h, 100Ah, 1004h, 1005h, 100Bh, 0103h, 0104h, 1000h, 1001h, 0105h

Рб	Адреса	Инструкција	ACC	Стек	I	T	L	PRIRRN	PRIRRM2	PRIRRM1
0	-	-	?	-	0	0	00	-	-	-
1	100	TRPE	?	-	0	1	00	-	-	-
2	101	INTE	?	-	1	1	00	-	-	1
3	102	LOAD #1	1	-	1	1	00	-	-	1
			1	1	0	0	10	-	-	-
4	1009	INC	2	1	0	0	10	-	-	-
5	100A	INC	3	1	0	0	10	1	-	-
			3	2	0	0	10	-	-	-
6	1004	DEC	2	2	0	0	10	-	-	-
7	1005	RTI	2	1	0	0	10	-	-	-
8	100B	RTI	2	-	1	1	00	-	-	-
9	103	INTD	2	-	0	1	00	-	1	-
10	104	STORE 1h	2	-	0	1	00	-	1	-
			2	3	0	0	00	-	1	-
11	1000	INC	3	3	0	0	00	-	1	-
12	1001	RTI	3	-	0	1	00	-	1	-
13	105	TRPD	3	-	0	0	00	-	1	-

1

I=1, T=1, L=00
0103

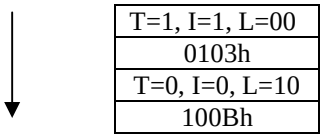
2

I=1, T=1, L=00
0103
I=0, T=0, L=10
100B

3

I=0, T=1, L=00
0105

в)



г) MEM[1] = 2h
д) Види приручник за лабораторијске вежбе

Задатак 7.

Адресни простор неког рачунара је величине 64 KB, адресибилна јединица је бајт, а 16 битни бројеви се у меморију смештају тако да је на нижој адреси нижи бајт. Механизам прекида је векторисан. IV (*Interrupt Vector*) табела почиње од адресе на коју указује регистар IVTP. *Trap* прекиду припада фиксно улаз 2 у IVT. При прекиду се на стеку чувају PSW и PC тим редом. Стек расте према нижим локацијама. Сви регистри опште намене (R*i*) су 8 битни. Процесор поседује инструкције `TRPE` (*Trap Enable)* и `TRPD` (*Trap Disable)* за дозволу, односно забрану *trap*-а. Инструкције `TRPE` и `TRPD` не реагују на прекиде (ни на *trap*). Дат је део програма на слици 1, *trap* прекидна рутина на слици 2, и део меморије почев од адресе 0 на слици 3. За време извршавања датог дела програма, нема других прекида осим *trap*-а.

Слика 1

Адреса	Инструкција
FF00h	TRPD
FF01h	MOV R0, #2
FF03h	TRPE
FF04h	DEC R0
FF05h	JNZ FF03h
FF08h	TRPD

Слика 2

Адреса	Инструкција	
00A0h	POP	R1
00A1h	PUSH	R1
00A2h	RTI	

Слика 3

Адреса	Садржај
0000h	00h
0001h	A0h
0002h	00h
0003h	0Ah
0004h	00h
0005h	A0h
0006h	00h

Слика 4

Адреса	Инструкција
00A0h	POP R1
00A1h	PUSH #2
00A3h	RTI

а) Која се вредност налази на адреси FF07h?
б) Која се вредност налази у регистру IVTP (та вредност је већа или једнака од 0, а мања од 7)?
в) Написати секвенцу адреса наредби које се редом извршавају, почев од FF00h, закључно са FF08h. Резултат дати табеларно тако да табела садржи редни број интрукције, адресу на којој започиње инструкција, саму инструкцију, садржај акумулатора након извршење инструкције, вредности свих познатих бита унутар програмске статусне речи.
г) Која се вредност налази у регистру R1 непосредно пре првог извршавања наредбе `RTI` на адреси 00A2h?
е) Написати секвенцу адреса наредби које се редом извршавају, почев од FF00h, све док су адресе познате, ако се уместо *trap* рутине са слике 2 користи рутина са слике 4.
adresa instrukcija

Решење:

а) FF07h садржај FFh.
б) IVTP једнако 1.
в)

Рб	Адреса	Инструкција	R0	R1	Стек	T
0	-	-	?	?	-	0
1	FF00	TRPD	?	?	-	0
2	FF01	MOV R0, #2	2	?	-	0
3	FF03	TRPE	2	?	-	1
4	FF04	DEC R0	1	?	-	1
			1	?	1	0
5	00A0	POP R1	1	5	2	0

6	00A1	PUSH R1	1	5	1	0
7	00A2	RTI	1	5	-	1
8	FF05	JNZ FF03h	1	5	-	1
			1	5	3	0
9	00A0	POP R1	1	3	4	0
10	00A1	PUSH R1	1	3	3	0
11	00A2	RTI	1	3	-	1
12	FF03	TRPE	1	3	-	1
13	FF04	DEC R0	0	3	-	1
			0	3	5	0
14	00A0	POP R1	0	5	6	0
15	00A1	PUSH R1	0	5	5	0
16	00A2	RTI	0	5	-	1
17	FF05	JNZ FF03h	0	5	-	1
			0	5	7	0
18	00A0	POP R1	0	8	8	0
19	00A1	PUSH R1	0	8	7	0
20	00A2	RTI	0	8	-	1
21	FF08	TRPD	0	8	-	0

1

T=1
FF
05

2

T=1
FF

3

T=1
FF
03

4

T=1
FF

5

T=1
FF
05

6

T=1
FF

7

T=1
FF
08

8

T=1
FF

г) R1 једнака 5.

д) Секвенца адреса је тако: FF00, FF01, FF03, FF04, 00A0, 00A1, 00A3, FF02, даље непознато.

Рб	Адреса	Инструкција	R0	R1	Стек	T
0	-	-	?	?	-	0
1	FF00	TRPD	?	?	-	0
2	FF01	MOV R0, #2	2	?	-	0
3	FF03	TRPE	2	?	-	1
4	FF04	DEC R0	1	?	-	1
			1	?	1	0
5	00A0	POP R1	1	5	2	0
6	00A1	PUSH #2	1	5	3	0
7	00A3	RTI	1	5	-	1
8	FF02	???				

1

T=1
FF
05

2

T=1
FF

3

T=1

FF
02

Задатак 8.

Адресни простор процесора је величине 128KB, адресибилна јединица је 16 битна реч, а вишеречни бројеви се смештају тако да је на вишој адреси нижа реч. Процесор је једноадресни са раздвојеним меморијским и улазно/излазним адресним просторима, а механизам прекида је векторисан. IV (*Interrupt Vector*) табела почиње од адресе на коју указује регистар IVTP (*Interrupt Vector Table Pointer*), а регистар IVTP има вредност 2. Процесор има три улазне линије IRQ0, IRQ1 и IRQ2 за спољне маскирајуће прекиде, при чему је IRQ0 највишег приоритета, а IRQ2 најнижег приоритета, на које су везане периферије PER0, PER1 и PER2, респективно. Њима су придружени улази 2, 3 и 4 у IV табелу, респективно. Не прихвата се прекид истог нивоа приоритета. Адресе 16 битних регистара у којима се чувају бројеви улаза су 10h, 20h и 30h, респективно. У PSW-у постоји бит I (*Interrupt Enable)* који се брише у кораку за обраду прекида, као и одређен број L бита. При прекиду се на стеку чувају PSW и PC тим редом. Стек расте према нижим локацијама. Акумулатор је 16 битни. Инструкције INTE, INTD, RTI и INT не реагују на прекиде. Инструкција INT не мења ниво приоритета текућег програма. Дат је део главног програма на слици 1, прекидне рутине на слици 2, изглед дела меморије почев од адресе 0 дат је на слици 3. Инструкција на адреси 0100h означена је као 1. (прва) по редоследу извршавања, а свака следећа инструкција која се извршава означена је следећим редним бројем. У току извршавања 2. инструкције стиже захтев за прекид по линији IRQ2, а у току 5. по линији IRQ0. На почетку су сви бити PSW-а постављени на 0. Не постоји IMR регистар.

Slika 1

Адреса	Наредба
0100h	LOAD #2h
0102h	INTE
0103h	INC
0104h	ADD #2h
0106h	INT #3h
0108h	INTD

Слика 2

Адреса	Наредба
1000h	PUSH
1001h	INTE
1002h	LOAD 1h
1004h	INC
1005h	STORE 1h
1007h	POP
1008h	RTI
1009h	INTE
100Ah	INC
100Bh	RTI
100Ch	INC
100Dh	RTI

Слика 3

Адреса	Садржај
0000h	0000h
0001h	0001h
0002h	1000h
0003h	1008h
0004h	1000h
0005h	100Ch
0006h	1009h

а) На којим адресама започињу прекидне рутине за линије IRQ0, IRQ1 и IRQ2, респективно?
б) Написати део програма којим се иницијализује улаз 3 у вектор табели.
в) Написати део програма којим се додељују бројеви улаза наведеним периферијама.
г) Дати секвенцу адреса инструкција које се редом извршавају по датом сценарију. Резултат дати табеларно тако да табела садржи редни број интрукције, адресу на којој започиње инструкција, саму инструкцију, садржај акумулатора након извршење инструкције, вредности свих познатих бита унутар програмске статусне речи.
д) Приказати садржај свих познатих локација на врху стека након извршавања 6. инструкције. За сачувану вредност PSW дати само вредности бита I и L. Назначити у коме смеру расте стек.

Решење:

а) IRQ0 – 1000h, IRQ1 – 100Ch, IRQ2 – 1009h

б)
```
LOAD #100Ch
STORE 5h
```

в)
```
LOAD #2h
OUT 10h
LOAD #3h
OUT 20h
LOAD #4h
OUT 30h
```

г) 100h, 102h, 103h, 1009h, 100Ah, 1000h, 1001h, 1002h, 1004h, 1005h, 1007h, 1008h, 100Bh, 104h, 106h, 100Ch, 100Dh, 108h, 109h

Рб	Адреса	Инструкција	ACC	Стек	I	L	PRIRR0	PRIRR1	PRIRR2
0	-	-	?	-	0	00	-	-	-
1	0100	LOAD #2	2	-	0	00	-	-	-
2	0102	INTE	2	-	1	00	-	-	1
3	0103	INC	3	-	1	00	-	-	1
			3	1	0	01	-	-	-
4	1009	INTE	3	1	1	01	-	-	-
5	100A	INC	4	1	1	01	1	-	-
			4	2	0	11	-	-	-

6	1000	PUSH	4	3	0	11	-	-	-
7	1001	INTE	4	3	1	11	-	-	-
8	1002	LOAD 1h	1	3	1	11	-	-	-
9	1004	INC	2	3	1	11	-	-	-
10	1005	STORE 1h	2	3	1	11	-	-	-
11	1007	POP	4	2	1	11	-	-	-
12	1008	RTI	4	1	1	01	-	-	-
13	100B	RTI	4	-	1	00	-	-	-
14	0104	ADD #2	6	-	1	00	-	-	-
15	0106	INT #3	6	4	0	00	-	-	-
16	100C	INC	7	4	0	00	-	-	-
17	100D	RTI	7	-	1	00	-	-	-
18	0108	INTD	7	-	0	00	-	-	-

1

I=1, L=00
0104

2

I=1, L=00
0104
I=1, L=01
100B

3

I=1, L=00
0104
I=1, L=01
100B
0004

4

I=1, L=00
0108

д)

I=1, L=00
0104h
I=1, L=01
100Bh
0004h

Задатак 9.

Једноадресни 16 битни процесор са раздвојеним улазно излазним и меморијским адресним простором поседује само једну линију IRQ за спољашњи прекид коме може да се додели произвољан улаз у IVT. Постоји још и линија INTA којом процесор одобрава прекид и захтева постављање броја улаза на магистралу података. Посебан уређај, контролер прекида, служи за паралелну арбитрацију до 4 захтева за прекид. Контролер не подржава угнежђавање прекида са приоритетом, већ само приоритирање тренутно постојећих захтева. Контролер поседује и регистар маске, који се налази на адреси 0FF0h улазно излазног (I/O) адресног простора и чија 4 најнижа бита маскирају захтеве за прекид на улазима контролера.

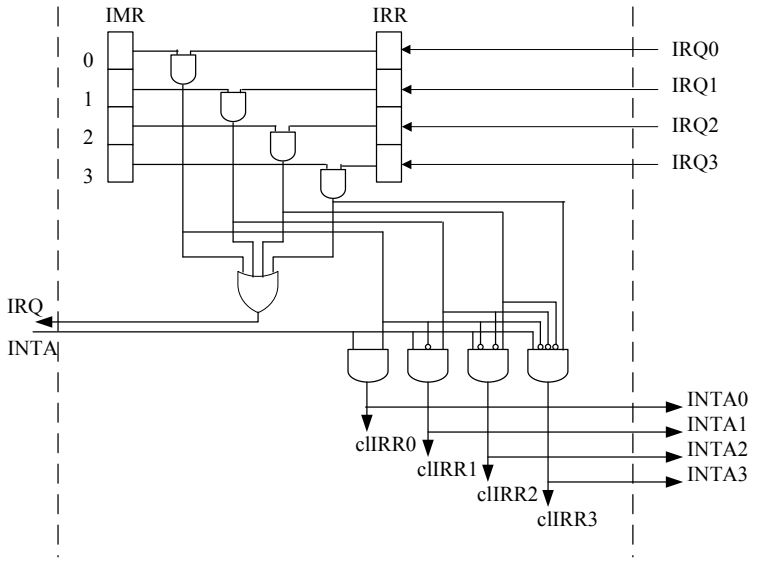
а) Нацртати принципијелну структурну шему контролера.

б) Написати део програма којим се маскирају прекиди на улазима IRQ1 и IRQ2 контролера, а остали дозвољавају.

в) Приказати начин везивања 16 линија захтева за прекид на процесор, помоћу одговарајућег броја датих контролера.

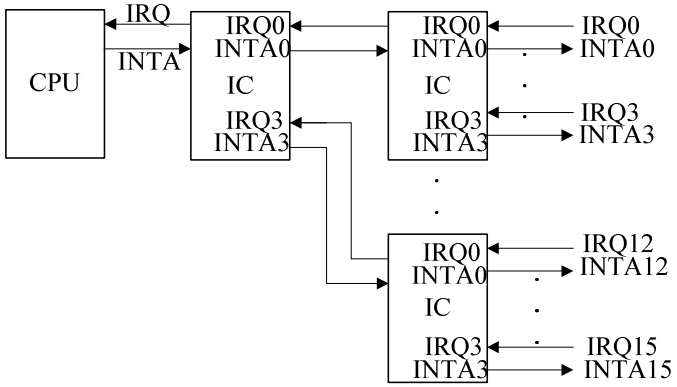
Решење

а)



б)

```
LOAD #1001b
OUT 0FF0h
```



в)

Задатак 10.

Адресни простор процесора је величине 8GB, адресибилна јединица је 16 битна реч, а вишеречни бројеви се смештају тако да је на нижој адреси нижа реч. Процесор је једноадресни са раздвојеним меморијским и I/O адресним просторима, а механизам прекида је векторисан. IV (*Interrupt Vector*) табела почиње од адресе на коју указује регистар IVTP (*Interrupt Vector Table Pointer*), а регистар IVTP има вредност 8. Процесор поседује само једну линију IRQ за спољашњи прекид коме може да се додели произвољан улаз у IV табели. Постоји још и линија INTA којом процесор одобрава прекид и захтева постављање броја улаза на магистралу података. Посебан уређај, контролер прекида, служи за паралелну арбитрацију до 4 захтева за прекид, који стижу по линијама IRQ0 до IRQ3 за спољне маскирајуће прекиде, при чему је IRQ0 највишег приоритета, а IRQ3 најнижег приоритета, на које су везане периферије PER0 до PER3, респективно, којима треба доделити улазе 2, 3, 1 и 4 у вектор табели, и којима одговарају прекидне рутине на адресама 1007h, 1000h, 1010h и 1003h респективно. Адресе 8 битних регистара у којима се чувају бројеви улаза су 10h, 20h, 30h и 40h респективно. Контролер не подржава угнежђавање прекида са приоритетом, већ само приоритирање тренутно постојећих захтева. Контролер поседује и регистар маске, који је се налази на адреси 00ABh I/O адресног простора, и чија 4 најнижа бита маскирају захтеве за прекид на улазима контролера. Почетна вредност овог регистра је Fh. У PSW-у постоји бит I (*Interrupt Enable)* који се брише у кораку за обраду прекида. При прекиду се на стеку чувају ACC, PSW и PC тим редом. Стек расте према нижим локацијама. Акумулатор је 16 битни. Дат је део главног програма на слици 1, прекидне рутине на слици 2. Инструкција на адреси 0100h означена је као 1. (прва) по редоследу извршавања, а свака следећа инструкција која се извршава означена је следећим редним бројем. У току извршавања 3. инструкције стиже захтев за прекид по линији IRQ0, у току 5. по линији IRQ2, а у току 7. по линији IRQ1. На почетку су сви бити PSW-а постављени на 0. Инструкције INTE и INTD не реагују на прекид. Почетни садржај локације 1h је 0.

Слика 1

Адреса	Наредба
0100h	INTE
0101h	LOAD 1h
0104h	DEC
0105h	STORE 1h
0108h	INC
0109h	INTD

Слика 2

Адреса	Наредба
1000h	INTE
1001h	INC
1002h	RTI
1003h	INTE
1004h	DEC
1005h	INTE
1006h	RTI
1007h	INTE
1008h	LOAD 1h
100Bh	INC
100Ch	STORE 1h
100Fh	RTI
1010h	LOAD 1h
1013h	XOR #1h
1016h	STORE 1h
1019h	RTI

а) Написати део програма којим се додељују бројеви улаза наведеним периферијама.

б) Написати секвенцу адреса наредби које се редом извршавају, почев од адресе 0100h. Резултат дати табеларно тако да табела садржи редни број интрукције, адресу на којој започиње инструкција, саму инструкцију, садржај акумулатора након извршење инструкције, вредности свих познатих бита унутар програмске статусне речи.

в) Приказати садржај свих познатих локација на врху стека након извршавања 8. инструкције. За сачувану вредност PSW дати само вредност бита I. Назначити у коме смеру расте стек.

г) Нацртати принципијелну структурну шему контролера.

д) Написати део програма којим се маскирају прекиди на улазима IRQ0 и IRQ3 контролера, а остали дозвољавају.

Решење

а)

```
LOAD #02h
OUTB 0010h
LOAD #03h
OUTB 0020h
LOAD #01h
OUTB 0030h
LOAD #04h
OUTB0040h
```

б) 0100h, 0101h, 0104h, 1007h, 1008h, 1010h, 1013h, 1016h, 1019h, 1000h, 1001h, 1002h, 100Bh, 100Ch, 100Fh, 0105h, 0108h, 0109h

Рб	Адреса	Инструкција	ACC	Стек	I	PRIRR	IRR0	IRR1	IRR2	IRR3
0	-	-	?	-	0	-	-	-	-	-
1	0100	INTE	?	-	1	-	-	-	-	-
2	0101	LOAD 1h	0	-	1	-	-	-	-	-
3	0104	DEC	-1	-	1	1	1	-	-	-
			-1	1	0	-	-	-	-	-
4	1007	INTE	-1	1	1	-	-	-	-	-
5	1008	LOAD 1h	0	1	1	1	-	-	1	-
			0	2	0	-	-	-	-	-
6	1010	LAOD 1h	0	2	0	-	-	-	-	-
7	1013	XOR #1	1	2	0	1	-	1	-	-
8	1016	STORE 1h	1	2	0	1	-	1	-	-
9	1019	RTI	0	1	1	1	-	1	-	-
			0	3	0	-	-	-	-	-
10	1000	INTE	0	3	1	-	-	-	-	-
11	1001	INC	1	3	1	-	-	-	-	-
12	1002	RTI	0	1	1	-	-	-	-	-
13	100B	INC	1	1	1	-	-	-	-	-
14	100C	STORE 1h	1	1	1	-	-	-	-	-
15	100F	RTI	-1	-	1	-	-	-	-	-
16	0105	STORE 1h	-1	-	1	-	-	-	-	-
17	0108	INC	0	-	1	-	-	-	-	-
18	0109	INTD	0	-	0	-	-	-	-	-

1

FFFF
I=1
0000
0105

2

FFFF
I=1
0000

3

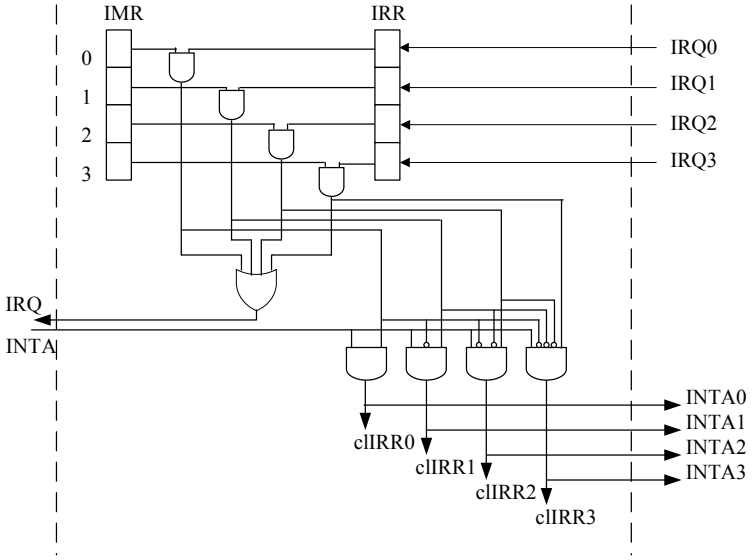
FFFF
I=1
0000
0105
FFFF
I=1
0000
100B

0105
FFFF
I=1
0000
100B

в)

FFFFh
I=1
0000h
0105h
0000h
I=1
0000h
100Bh

г)



д) LOAD #0110b
OUTB 0000 00ABh

Задатак 11.

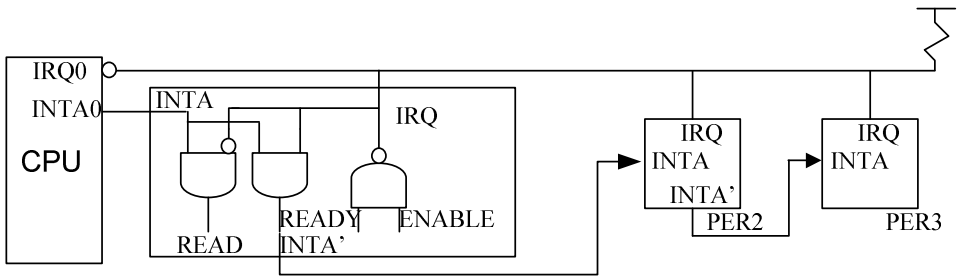
Једноадресни 16 битни процесор са раздвојеним улазно излазним и меморијским адресним простором поседује само једну линију IRQ0 за спољашњи прекид коме може да се додели произвољан улаз у IVT. Постоји још и линија INTA0 којом процесор одобрава прекид и захтева постављање броја улаза на магистралу података. На линију IRQ0 везани су захтеви за прекид од 3 периферије.

а) Нацртати принципијелну структурну шему овог система.

б) У случају да постоји више захтева за прекидом, како се одређује захтев највишег приоритета?

Решење

а) Daisy chainig



б) Приоритете IRQ0 до IRQ3 процесор разрешава интерно, кодером приоритета; када пошаље INTA0 најприоритетнија периферија са линије шаље свој број улаза (активан је њен READ сигнал).

Задатак 12.

Адресни простор процесора је величине 8GB, адресибилна јединица је 16 битна реч, а вишеречни бројеви се смештају тако да је на вишој адреси нижа реч. Процесор је једноадресни са раздвојеним меморијским и U/I адресним

просторима, а механизам прекида је векторисан. IV (*Interrupt Vector*) табела почиње од адресе на коју указује регистар IVTP (*Interrupt Vector Table Pointer*), а регистар IVTP има вредност 16h. Процесор има једну улазну линију IRQM за спољне маскирајуће прекиде и једну улазну линију IRQN за спољне немаскирајуће прекиде, којима се може додели произвољан улаз у IV табели. Постоји још и линија INTA којом процесор одобрава прекид и захтева постављање броја улаза на магистралу података. Не постоји посебан уређај, контролер прекида, већ се приотитирање маскирајућих прекида обавља користећи Daisy chainig. На линију IRQM су везане три периферије, PER1, PER2 и PER3, при чему је PER1 највишег приоритета, а на линију IRQN периферија PERN, којима треба доделити улазе 3, 5, 1 и 7 у вектор табели, и којима одговарају прекидне рутине на адресама 1007h, 1000h, 1003h и 1010h, респективно. Адресе 8 битних регистара у којима се чувају бројеви улаза су 10h, 20h, 30h и 40h респективно. У PSW-у постоји бит I (*Interrupt Enable)* који се брише у кораку за обраду прекида, не постоје L бити. При прекиду се на стеку чувају ACC, PSW и PC тим редом. Стек расте према нижим локацијама. Акумулатор је 16 битни. Дат је део главног програма на слици 1, прекидне рутине на слици 2. Инструкција на адреси 0100h означена је као 1. (прва) по редоследу извршавања, а свака следећа инструкција која се извршава означена је следећим редним бројем. У току извршавања 2. инструкције стиже захтев за прекид од периферије PER1, у току 5. од периферије PER2, а у току 7. од периферије PERN. На почетку су сви бити PSW-а постављени на 0. Инструкције INTE и INTD не реагују на прекид. Почетни садржај локације 1h је 0.

Слика 1

Адреса	Наредба
0100h	INTE
0101h	LOAD 1h
0104h	DEC
0105h	STORE 1h
0108h	INC
0109h	INTD

Слика 2

Адреса	Наредба
1000h	INC
1001h	INC
1002h	RTI
1003h	INTE
1004h	DEC
1005h	INTE
1006h	RTI
1007h	INTE
1008h	LOAD 1h
100Bh	INC
100Ch	STORE 1h
100Fh	RTI
1010h	LOAD 1h
1013h	XOR #1h
1016h	STORE 1h
1019h	RTI

а) Написати део програма којим се додељују бројеви улаза наведеним периферијама.

б) Нацртати изглед првих 8 улаза у вектор табели, означити адресе релевантних локација и уписати садржаје у њих.

в) Нацртати принципијелну структурну шему овог система.

г) Написати секвенцу адреса наредби које се редом извршавају, почев од адресе 0100h. Резултат дати табеларно тако да табела садржи редни број интрукције, адресу на којој започиње инструкција, саму инструкцију, садржај акумулатора након извршење инструкције, вредности свих познатих бита унутар програмске статусне речи.

д) Приказати садржај свих познатих локација на врху стека након извршавања 8. инструкције. За сачувану вредност PSW дати само вредност бита I. Назначити у коме смеру расте стек.

Решење

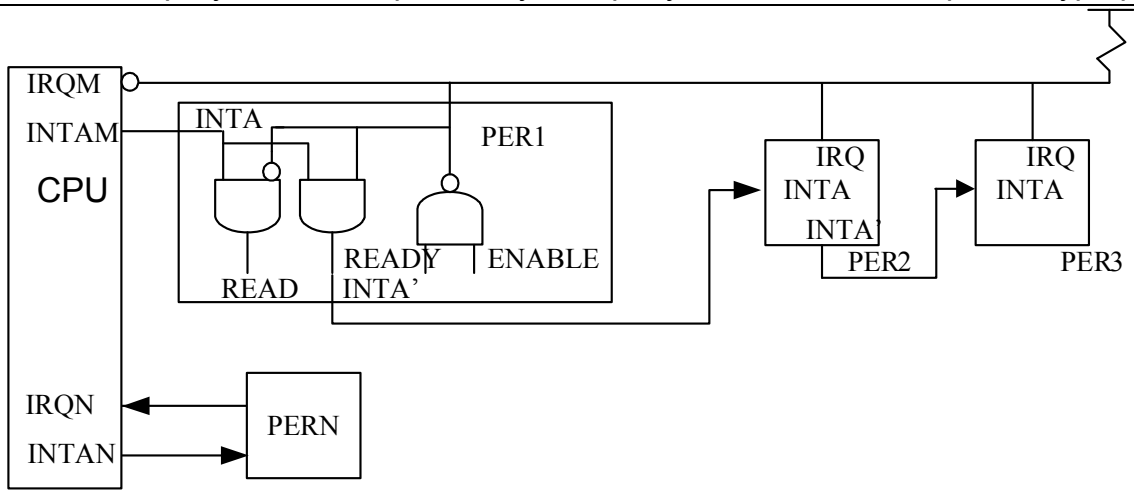
а)

```
LOAD #03h
OUTB 10h
LOAD #05h
OUTB 20h
LOAD #01h
OUTB 30h
LOAD #07h
OUTB 40h
```

б)

број улаза у IV табелу	меморијска адреса	садржај
	25h	1010h
7 PERN	24h	0000h
	23h	
6	22h	
	21h	1000h
5 PER2	20h	0000h
	1Fh	
4	1Eh	
	1Dh	1007h
3 PER1	1Ch	0000h
	1Bh	
2	1Ah	
	19h	1003h
1 PER3	18h	0000h
	17h	
0	16h	

в)



г) 0100h, 0101h, 1007h, 1008h, 100Bh, 1000h, 1001h, 1010h, 1013h, 1016h, 1019h, 1002h, 100Ch, 100Fh, 0104h, 0105h, 0108h, 0109h

Рб	Адреса	Инструкција	ACC	Стек	I	PRIRRN	PRIRRM	IRQM1	IRQM2	IRQM3
0	-	-	?	-	0	-	-	-	-	-
1	0100	INTE	?	-	1	-	-	-	-	-
2	0101	LOAD 1	0	-	1	-	1	1	-	-
			0	1	0	-	-	-	-	-
3	1007	INTE	0	1	1	-	-	-	-	-
4	1008	LOAD 1	0	1	1	-	-	-	-	-
5	100B	INC	1	1	1	-	1	-	1	-
			1	2	0	-	-	-	-	-
6	1000	INC	2	2	0	-	-	-	-	-
7	1001	INC	3	2	0	1	-	-	-	-
			3	3	0	-	-	-	-	-
8	1010	LOAD 1	0	3	0	-	-	-	-	-
9	1013	XOR #1	1	3	0	-	-	-	-	-
10	1016	STORE 1	1	3	0	-	-	-	-	-
11	1019	RTI	3	2	0	-	-	-	-	-
12	1002	RTI	1	1	1	-	-	-	-	-
13	100C	STORE 1	1	1	1	-	-	-	-	-
14	100F	RTI	0	-	1	-	-	-	-	-
15	0104	DEC	-1	-	1	-	-	-	-	-
16	0105	STORE 1	-1	-	1	-	-	-	-	-
17	0108	INC	0	-	1	-	-	-	-	-
18	0109	INTD	0	-	0	-	-	-	-	-

1

0000
I=1
0104
0000

2

0000
I=1
0104
0000
0001
I=1
100C
0000

3

0000
I=1
0104
0000
0001
I=1
100C
0000
0003
I=0
1002
0000

е)

0000h
I=1
0104h
0000h
0001h
I=1
100Ch
0000h
0003h
I=0
1002h
0000h

Задатак 13.

Једноадресни процесор са меморијски мапираним улазно излазим адресним простором поседује само једну линију IRQ за спољашње прекиде, којима одговара фиксан улаз у вектор табели. Адресе су ширине 16, а подаци 8 бита. На линију IRQ везани су захтеви за прекид од 4 периферије PER1…PER4. Сваки контролер периферије генерише захтев за преки када је бит READY у његовом статусном регистру постављен на 1. Статусни регистри за периферије PER1 до PER4 налазе се редом на адресама FF01h, FF02h, FF03h и FF04h, а бит READY су редом на позицијама 7, 3, 5 и 1. Прекидне рутине ових периферија почињу на адресама 100h, 200h, 300h и 400h, и завршавају се инструкцијом RTI. Периферија PER4 је највишег приоритета, а PER1 најнижег. При прекиду се на стеку чувају ACC, PSW и PC тим редом.

а) Написати "главну" прекидну рутину, на коју процесор прелази по пријему захтева за прекид по линији IRQ.

б) Какве измене су потребне у прекидној рутини из тачке а) ако се прекидне рутине завршавају наредбом RTS (повратак из потпрограма)?

Решење

а)

```
INTH: LOAD FF04h;   PER4
      AND #02h
      JNZ 400h
      LOAD FF03h;   PER3
      AND #20h
      JNZ 300h
      LOAD FF02h;   PER2
      AND #08h
      JNZ 200h
      LOAD FF01h;   PER1
      AND #80h
      JNZ 100h
      RTI;          ELSE
```

б) Уместо JUMP-а у потпрограму треба извршити JSR (Jump to Subroutine), а одмах иза RTI за сваку периферију посебно

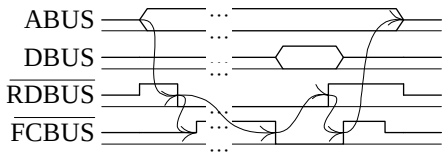
МЕМОРИЈА И МАГИСТРАЛА

Задатак 14.

За асинхрону магистралу нацртати и објаснити временски облик свих сигнала на магистрали (адресних, управљачких и линија података) за случај:
а) циклуса читања,
б) циклуса уписа и
в) циклуса прихватања броја улаза.
Сигнали на контролној магистрали су активни у логичкој нули.

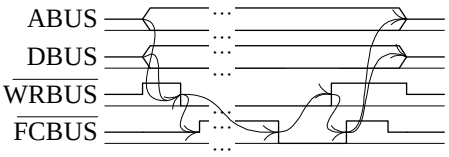
Решење:

а) Временски облици сигнала ABUS, DBUS, $\overline{\text{RDBUS}}$ и $\overline{\text{FCBUS}}$, које на магистрали размењују газда и слуга приликом реализације циклуса читања, дати су на слици.



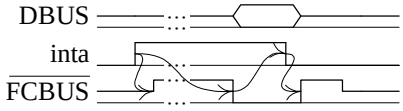
Временски облици сигнала за циклус читања на асинхроној магистрали

б) Временски облици сигнала ABUS, DBUS, $\overline{\text{WRBUS}}$ и $\overline{\text{FCBUS}}$, које на магистрали размењују газда и слуга приликом реализације циклуса уписа, дати су на слици.



Временски облици сигнала за циклус уписа на асинхроној магистрали

в) Временски облици сигнала DBUS и $\overline{\text{FCBUS}}$, које на магистрали размењују процесор као газда и уређај као слуга, и сигнала потврде inta, који по посебној линији која не припада магистрали процесор шаље улазно/излазном уређају приликом реализације циклуса прихватање броја улаза, дати су на слици.



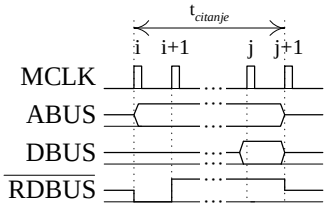
Временски облици сигнала за циклус прихватање броја улаза на асинхроној магистрали

Задатак 15.

За синхрону магистралу нацртати и објаснити временски облик свих сигнала на магистрали (адресних, управљачких и линија података) за случај:
а) циклуса читања,
б) циклуса уписа и
в) циклуса прихватања броја улаза.
Сигнали на контролној магистрали су активни у логичкој нули.

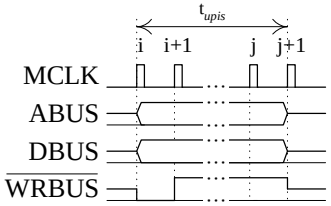
Решење

а) Временски облици сигнала ABUS, DBUS и $\overline{\text{RDBUS}}$, које на магистрали размењују газда и слуга приликом реализације циклуса читања, дати су на слици.



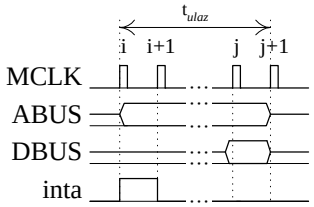
Временски облици сигнала за циклус читања на синхроној магистрали

б) Временски облици сигнала ABUS, DBUS и $\overline{\text{WRBUS}}$, које на магистрали размењују газда и слуга приликом реализације циклуса уписа, дати су на слици.



Временски облици сигнала за циклус уписа на синхроној магистрали

в) Временски облици сигнала DBUS које на магистрали размењују процесор као газда и уређај као слуга и сигнала потврде inta, који по посебној линији која не припада магистрали, процесор шаље улазно/излазном уређају приликом реализације циклуса прихватање броја улаза, дати су на слици.



Временски облици сигнала за циклус прихватање броја улаза на синхроној магистрали.

Задатак 16.

Адресни простор једноадресног процесора је 64KB. Из меморије се чита и у меморију уписује бајт по бајт. Улазно/излазни адресни простор је раздвојен. Највиших 32KB меморије је резервисано за ROM меморију, а најнижих 32KB за RAM меморију. Физичка RAM меморија заузима најнижих 8KB адресног простора резервисаног за RAM меморију, а физичка ROM најнижих 8KB адресног простора резервисаног за ROM меморију. На располагању су 256x4-бита RAM чипови (контролни улази су R, W и CS) и 1024x4-бита ROM чипови (контролни улази су R и CS).

а) Назначити адресне опсеге простора резервисаних за меморијски адресни простор, адресни простор резервисан за RAM, адресни простор резервисан за ROM меморију, физичку RAM меморију, физичку ROM меморију. Резултат представити табеларно.

б) Колико је RAM и ROM чипова потребно за реализацију меморије? Нацртати везу између чипова и магистрале микропроцесора.

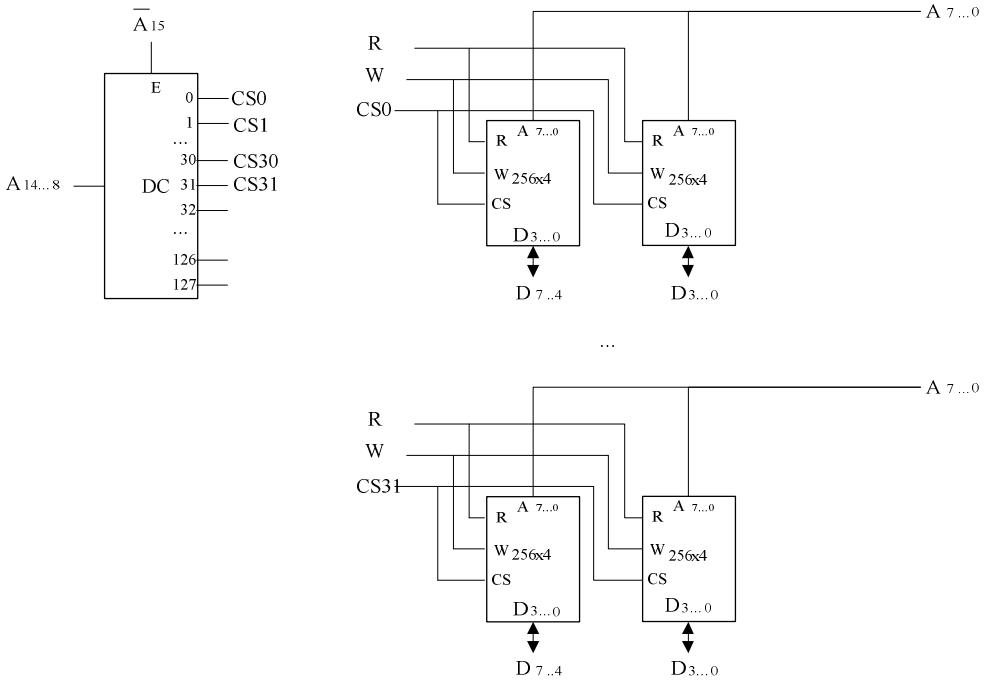
Решење

а)

Шта?	Прва адреса	Последња адреса
Меморијски адресни простор	0000h	FFFFh
Адресни простор резервисан за RAM	0000h	7FFFh
Адресни простор резервисан за ROM	8000h	FFFFh
Физичка RAM меморија	0000h	1FFFh
Физичка ROM меморија	8000h	9FFFh

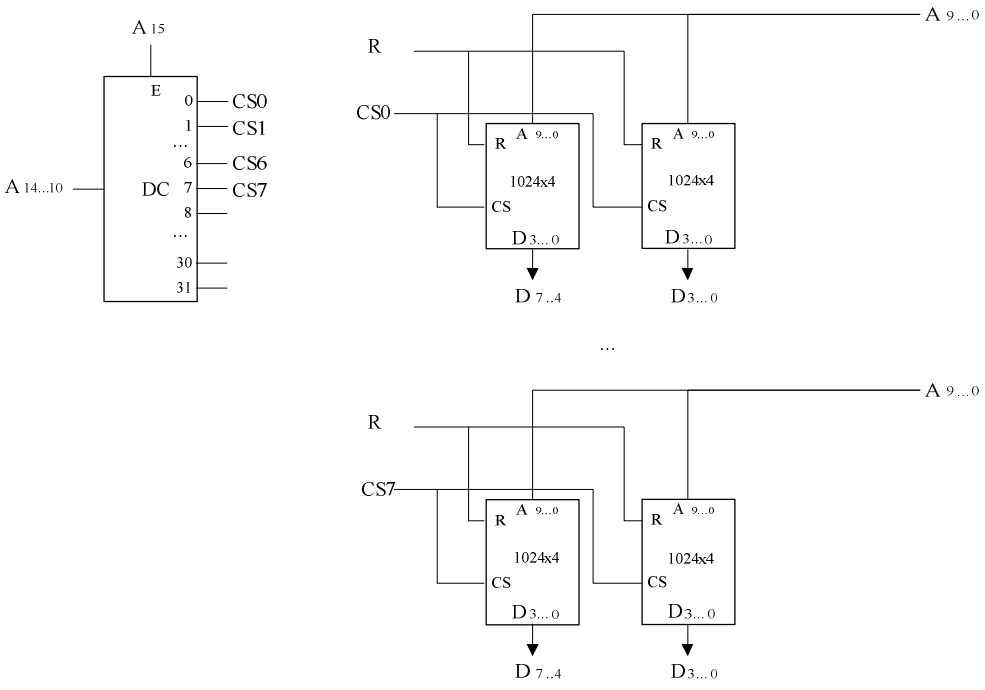
б) Реализација RAM меморије је представљена на слици.

$$\text{numRam} = \frac{\text{RAMSize}}{\text{RAMChipSize}} = \frac{8\text{Kx8бита}}{256\text{x4бита}} = \frac{8\text{Kx8бита}}{2\cdot 128\text{x4бита}} = \frac{64\cdot 128\text{x8бита}}{128\text{x8бита}} = 64$$



Реализација ROM меморије је представљена на слици.

$$\text{numRom} = \frac{\text{ROMSize}}{\text{ROMChipSize}} = \frac{8\text{Kx8бита}}{1024\text{x4бита}} = \frac{8\text{Kx8бита}}{2 \cdot 512\text{x4бита}} = \frac{16 \cdot 512\text{x8бита}}{512\text{x8бита}} = 16$$



Задатак 17.

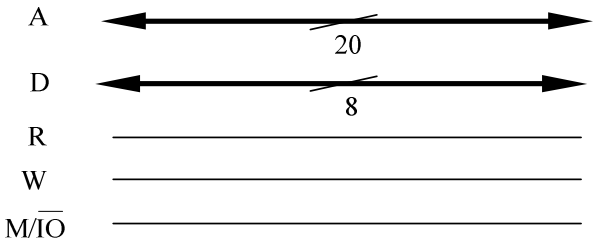
Адресни простор једноадресног процесора је 1MB, а адресирање је бајтовско. Процесор поседује 16-битни акумулатор. Спрежни део процесора према магистрали садржи регистар MAR и 8-битни регистар MDR. Улазно/излазни и меморијски адресни простори су раздвојени. Процесор је повезан са меморијом MEM и периферијом PER преко синхроне магистрале. Величина физичке меморије је 256KB и заузима највиши део адресног простора.

а) Нацртати све релевантне линије системске магистрале и прецизно назначити ширине адресне и магистрале података.

б) Приказати реализацију физичке меморије ако на располагању стоје меморијски модули 128Kx4 бита (контролни улази су R, W и CS).

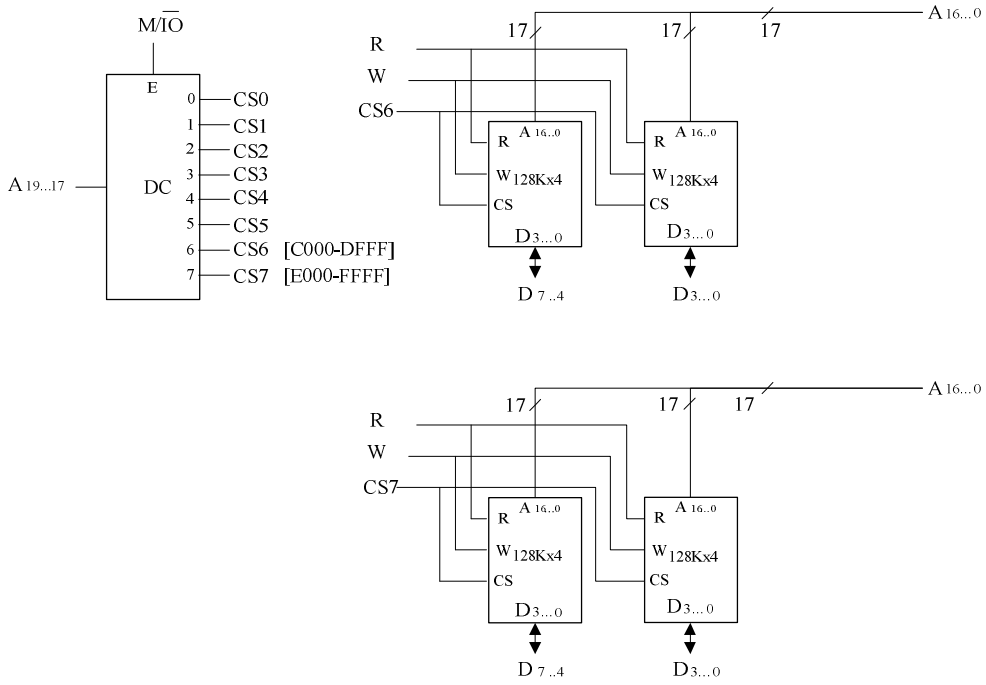
Решење

а) Релевантне линије системске магистрале су приказане на слици.



б) Реализација физичке меморије је представљена на слици.

$$\text{numRam} = \frac{\text{RAMSize}}{\text{RAMChipSize}} = \frac{256\text{Kx8бита}}{128\text{Kx4бита}} = \frac{256\text{Kx8бита}}{2 \cdot 64\text{Kx4бита}} = \frac{256\text{Kx8бита}}{64\text{Kx8бита}} = \frac{4 \cdot 64\text{x8бита}}{64\text{x8бита}} = 4$$



Задатак 18.

Посматра се двоадресни процесор са раздвојеним меморијским и улазно/излазним адресним просторима. Адресибилна јединица је 16-битна реч (W). Магистрала је асинхрона, а механизам прекида векторисан. Капацитет меморијског адресног простора је 8GB, при чему је највиших 2GB резервисано за ROM меморију. За адресирање улазно/излазних уређаја резервисана је 1GW почев од адресе 0h.

а) Нацртати све релевантне линије системске магистрале и прецизно назначити ширине адресне и магистрале података.

б) Назначити адресне опсеге простора резервисаних за меморијски адресни простор, улазно-излазни адресни простор, адресни простор резервисан за RAM и адресни простор резервисан за ROM меморију. Резултат представити табеларно.

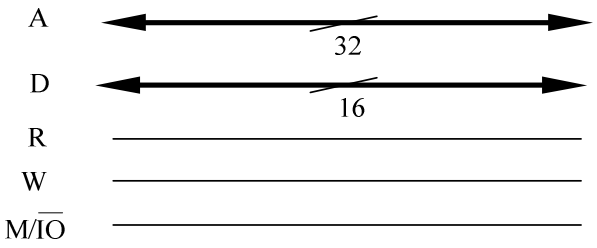
в) Приказати реализацију физичке RAM меморије која заузима највиших 4GB у адресном простору резервисаном за RAM меморију, ако су на располагању меморијски SRAM чипови капацитета 512Mx8 бита (контролни улази су **R**, **W** и **CS**).

г) Приказати реализацију физичке ROM меморије која заузима најнижих 1GB у адресном простору резервисаном за ROM меморију, ако су на располагању ROM меморијски чипови капацитета 256Mx8бита (контролни улази су **R** и **CS**).

д) Шта ће се десити у посматраном систему током циклуса уписа на магистрали који је иницран инструкцијом **STORE** C000 0000h, R1 (мем.дир). Образложити. Предложити решење за отклањање проблема, уколико постоји.

Решење

а) Релевантне линије системске магистрале су приказане на слици.

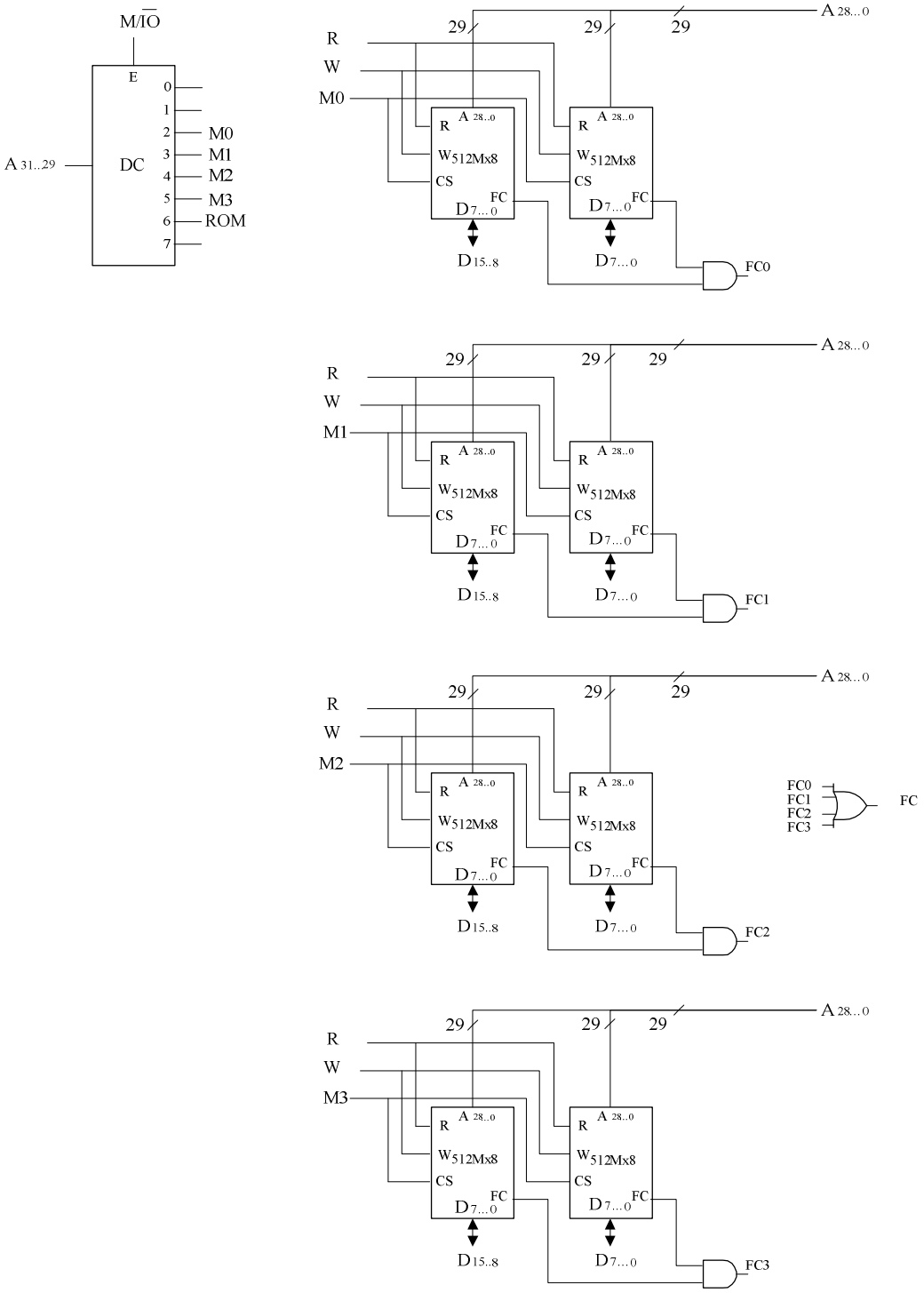


б)

Шта?	Прва адреса	Последња адреса
Меморијски адресни простор	0000 0000h	FFFF FFFFh
Улазно-излазни адресни простор	0000 0000h	3FFF FFFFh
Адресни простор резервисан за RAM	0000 0000h	BFFF FFFFh
Адресни простор резервисан за ROM	C000 0000h	FFFF FFFFh

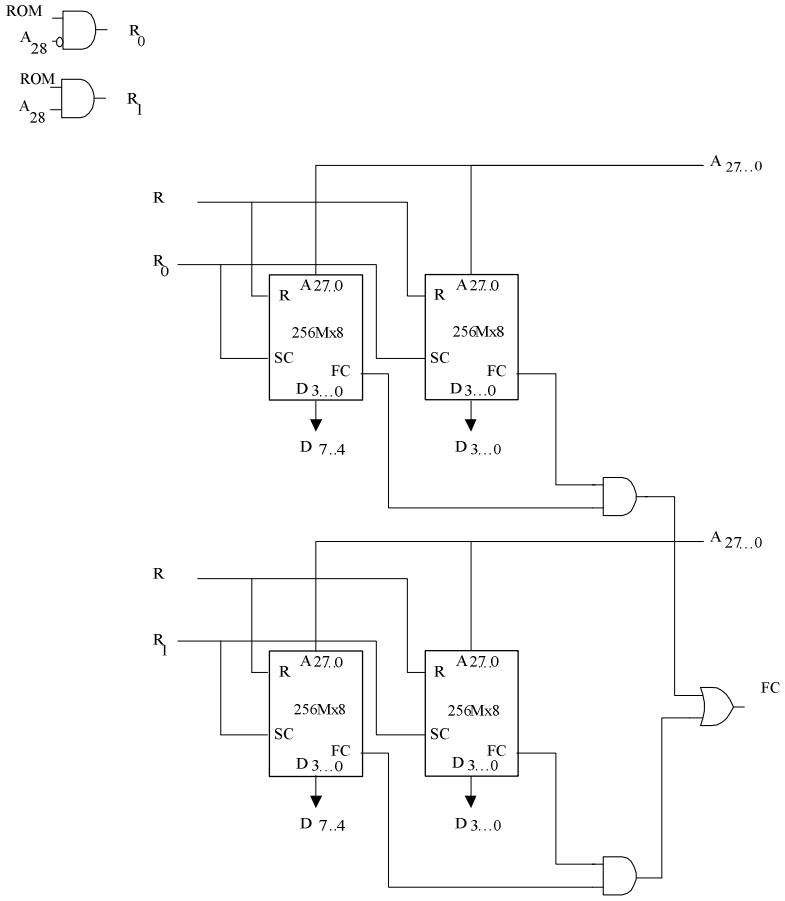
в) Реализација RAM меморије је представљена на слици.

$$\text{numRam} = \frac{\text{RAMSize}}{\text{RAMChipSize}} = \frac{\text{2Gx16бита}}{\text{512Mx8бита}} = \frac{\text{2Gx16бита}}{2 \cdot \text{256Mx8бита}} = \frac{\text{2Gx16бита}}{\text{256Mx16бита}} = \frac{8 \cdot \text{256Mx16бита}}{\text{256Mx16бита}} = 8$$



г) Реализација ROM меморије је представљена на слици.

$$\text{numRom} = \frac{\text{ROMSize}}{\text{ROMChipSize}} = \frac{512\text{Mx16бита}}{256\text{Mx8бита}} = \frac{512\text{Mx16бита}}{2 \cdot 128\text{Mx8бита}} = \frac{4 \cdot 128\text{Mx16бита}}{128\text{Mx16бита}} = 4$$



д) У фази извршења инструкција STORE иницира упис у меморију. Међутим, одредишна адреса C000 0000 припада опсегу адреса које припадају физичкој ROM меморији из које се може само читати. ROM меморија неће генерисати FC сигнал, тако да процесор не може да заврши циклус уписа на магистрали и остаје блокиран. Да би се то спречило потребно је додати једно коло које ће препознати да је дошло до уписа у ROM меморију (довољно је двоулазно I коло, на чије улазе долазе W и сигнал са адресног декодера који је активан за адресе резервисане за ROM меморију) и генерисати FC. Исти сигнал се може водити и на улаз за прекиде, да би се извршила прекидна рутина која ће сигнализирати недозвољен упис у ROM меморију.

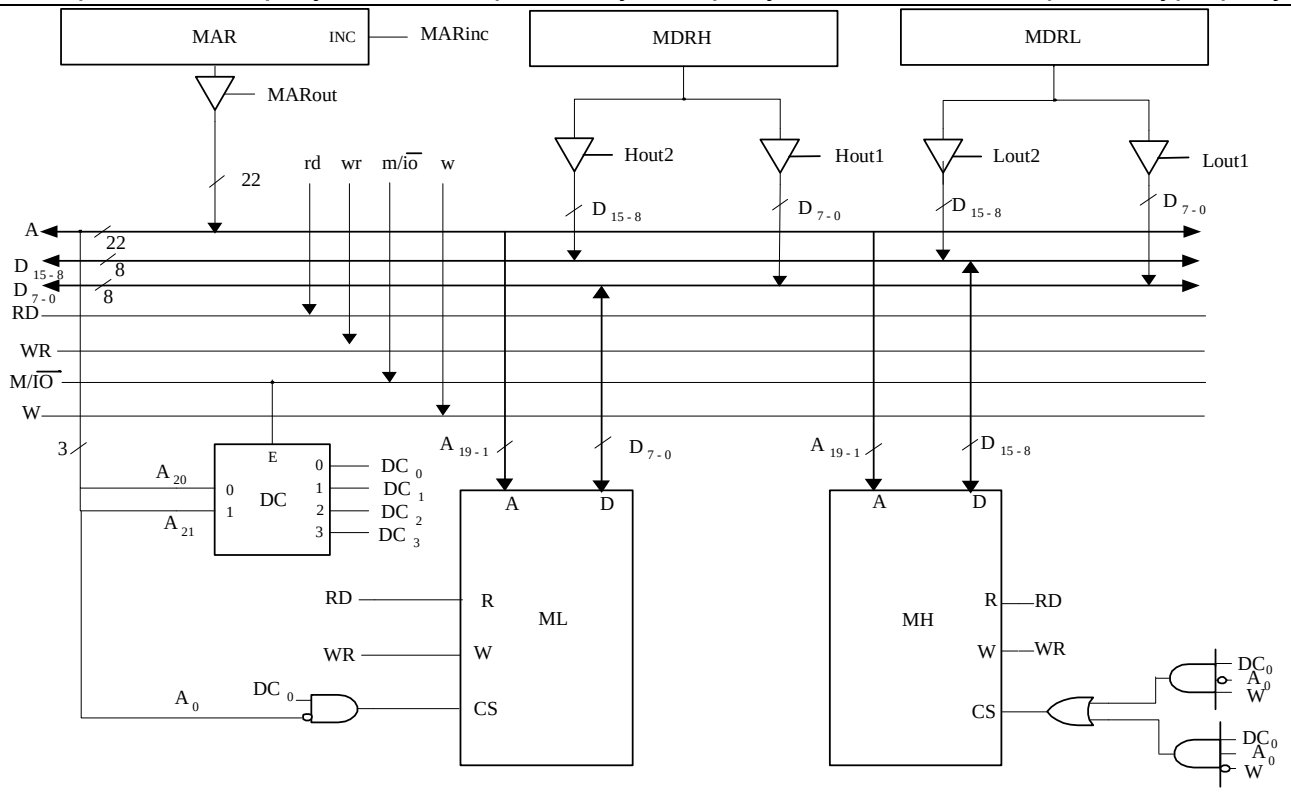
Задатак 19.

Једноадресни процесор са раздвојеним адресним просторима комуницира са меморијом преко синхроне магистрале приказане на слици. Управљачка линија W има вредност 1 када се обавља трансфер 16 битне речи. Инструкција STOREB преноси садржај регистра MDRL у меморију на адресу која се налази у регистру MAR. Инструкција STOREW преноси садржај из регистара MDRH (виши бајт) и MDRL (нижи бајт) у меморију на адресу која се налази у регистру MAR. 16 битна реч се смешта тако да се нижи бајт смешта на нижу адресу. Претпоставити да циклус уписа у меморију траје један циклус такта.

а) Колика је величина целог меморијског адресног простора? Колика је величина меморијских модула ML и MH?

б) Које адресе обухвата меморијски модул ML, а које меморијски модул MH?

в) Колико циклуса такта траје фаза извршења следећих инструкција: (*i*) `STOREW 8h`, (*ii*) `STOREW 5h`.



Решење

а) 4MB, 512KB, 512KB
б) ML: 0h, 2h, ... (парне адресе), MH: 1h, 3h,... (непарне адресе).
в) 1; 2.

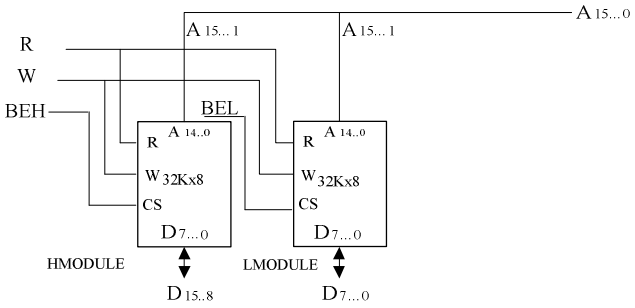
Задатак 20.

Посматра се 16-битни little-endian процесор (старији бајт се смешта на вишу адресу). Минимална адресибилна јединица је један бајт. Процесор је са меморијом повезан преко магистрале са 16 адресних и 16 линија за податке. Не постоји улазно-излазни адресни простор.
Меморија је подељена на два модула виши (HMODULE) и нижи (LMODULE) и важе следећа правила:
• Дохватање 16-битног операнда који се налази на парној адреси траје један циклус
• Дохватање 16-битног операнда који се налази на непарној адреси траје два циклуса
• Процесор шаље меморији следеће сигнале:
• R: читање из меморије
• W: упис у меморију
• BEH(Byte Enable High): дозвола за приступ меморијском модулу повезаном на линије D15..8
• BEL (Byte Enable Low): дозвола за приступ меморијском модулу повезаном на линије D7..0
а) Нацртати меморијске модуле и њихову везу са магистралом. Назначити најнижих пет адреса у сваком од модула. Који контролни сигнали се користе за селектовање сваког меморијског модула?
б) Посматра се 16-битни операнд ABCDh. Ако се операнд смешта у меморију почевши од адресе 1234h,
1. Колико циклуса на магистрали је потребно за дохватање операнда из меморије?
2. Приказати (хексадецимално) вредности које се појављују на магистрали података.
3. Које адресе (хексадецимално) процесор шаље меморији?
4. Које контролне сигнале поставља процесор у циљу одабира одговарајућег меморијског модула?
в) Ако се предходно поменути операнд смешта у меморију почевши од адресе 1235h, одговорити на питања од 1. до 4.
Процесор може да прихвата и да поставља податке на линије података D15..8 и D7..0 и да аутоматски реорганизује бајтове по потреби било да представљају виши или нижи бајт.

Решење

а) Најнижа линија адресне магистрале нема утицај на декодовање адреса због постојања додатних управљачких линија BEH и BEL, па неки произвођачи у циљу смањења броја адресних линија изостављају ову линију адресне магистрале. Управљачки сигнал BEH омогућава селектовање модула HMODULE, а управљачки сигнал BEL селектовање модула LMODULE. Непарне меморијске локације се пресликавају у HMODULE, па је најнижих пет адреса у том модулу 1h, 3h, 5h, 7h, 9h (A0=1). Парне меморијске локације се пресликавају у LMODULE, па је

најнижих пет адреса у том модулу 0h, 2h, 4h, 6h, 8h (A0=0). Меморијски модули и њихова веза са магистралом је представљена на слици.



б) Операнд се смешта у меморију на парну адресу, па је потребан један циклус на магистрали да би се обавио упис. На адресну магистралу се поставља адреса на коју је потребно извршити упис податка односно 1234h, млађи бајт податка се поставља на нижих 8 линија магистрале података, а старији бајт на виших 8 линија магистрале података. У питању је циклус уписа па ће управљачки сигнал W бити активан. Пошто се упис обавља и у LMODULE и у HMODULE оба управљачка сигнала BEL и BEH ће такође бити активна. Вредности које се појављују на адресној магистрали, магистрали података као и сви релевантни контролни сигнали приказани су у табели.

Цк	Адресна маг.	D_{0-7}	D_{15-8}	R	W	BEH	BEL
1	1234h	CDh	ABh	0	1	1	1

в) Операнд се смешта у меморију на непарну адресу, па ће бити потребна два циклуса на магистрали да би се обавио упис. У првом циклусу ће се у модул HMODULE уписати млађи бајт податка, а у другом циклусу у модул LMODULE се уписује старији бајт податка. Вредности које се појављују на адресној магистрали, магистрали података као и сви релевантни контролни сигнали приказани су у табели.

Цк	Адресна маг.	D_{0-7}	D_{15-8}	R	W	BEH	BEL
1	1234h	XX	CDh	0	1	1	0
2	1236h	ABh	XX	0	1	0	1

Задатак 21.

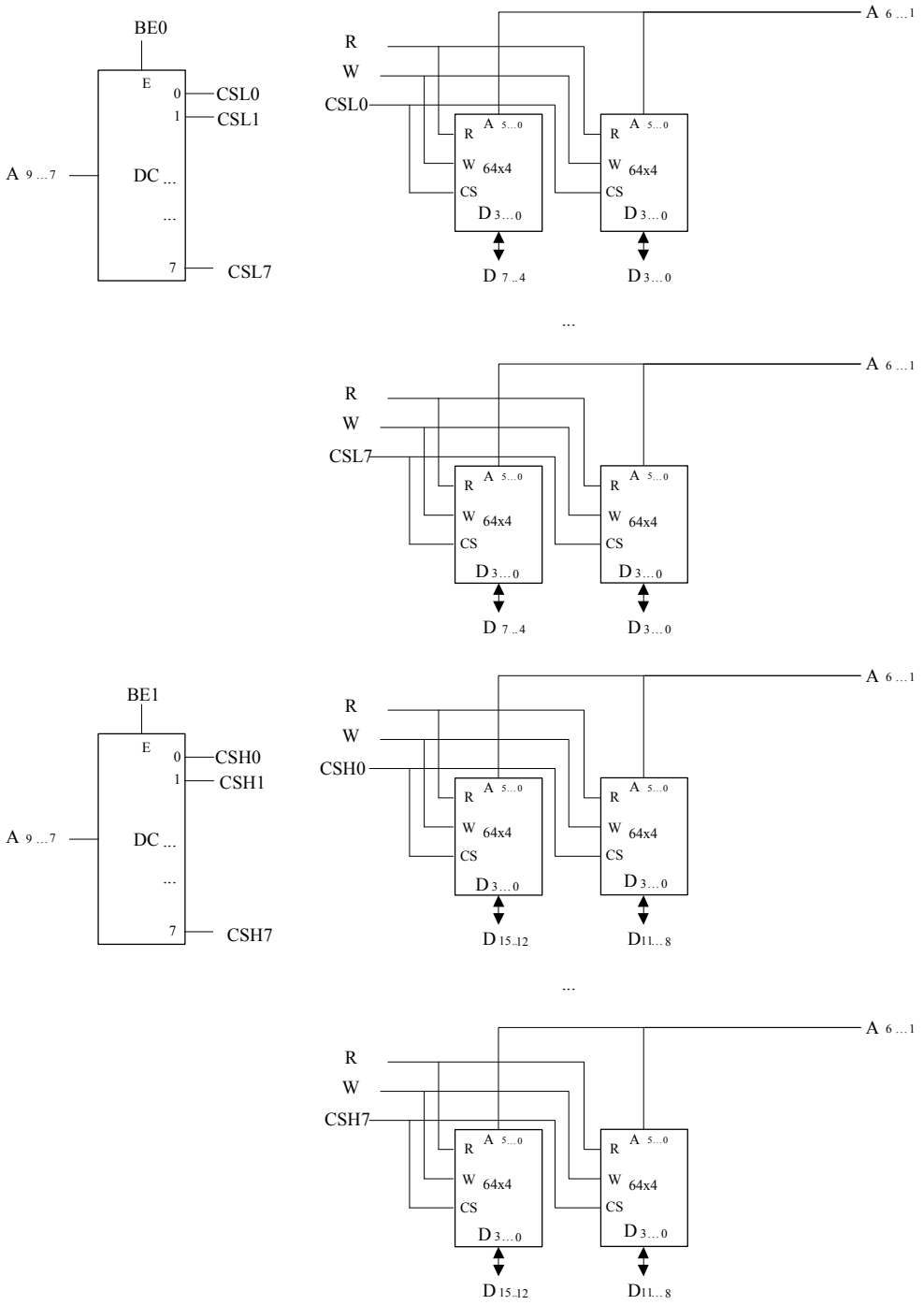
Пројектовати 16-битну меморију капацитета 8192 бита која заузима највиши део меморијског простора. Користити SRAM чипове величине 64x4 бита (контролни улази су **R**, **W** и **CS**). Процесор је *little-endian*. Приказати везу меморијских модула са магистралом. Потребно је омогућити трансфер и 8-битних и 16-битних података. Не постоји улазно-излазни адресни простор. Најмања адресибилна јединица је бајт.

Решење

Сигнал BE1 омогућава везу меморијског модула и линија података D15..8, а сигнал BE0 везу меморијског модула и линија података D7..0. (BEx - Byte Enable x).

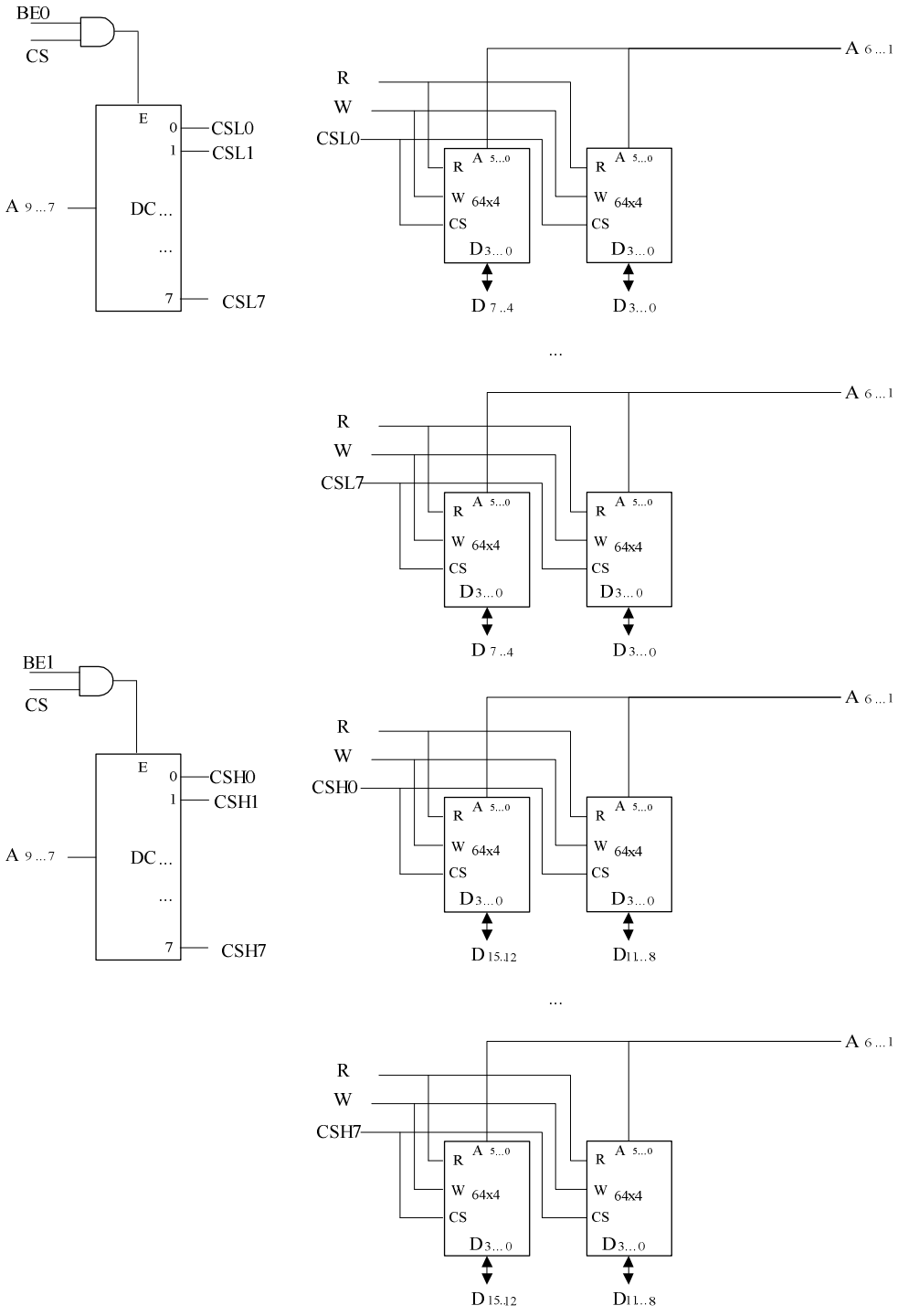
Реализација RAM меморије је представљена на слици.

$$\text{numRam} = \frac{\text{RAMSize}}{\text{RAMChipSize}} = \frac{8192\text{бита}}{64\text{x}4\text{бита}} = \frac{1024\text{x}8\text{бита}}{2 \cdot 32\text{x}4\text{бита}} = \frac{1024\text{x}8\text{бита}}{32\text{x}8\text{бита}} = 32$$



У зависности од тога да ли се приступа 8-битном или 16-битном податку на парној односно непарној адреси процесор генерише сигнале дозволе BE0 и BE1.

Ако заузима највиши део меоријског простора модул се може користити за креирање других меморијских чипова, у том случају реализација је представљена на слици.


BE0
CS
E
0
1
7
CSL0
CSL1
CSL7
A 9 ...7
DC ...
...
A 6 ...1
R
W
CSL0
CSL7
A 5...0
64x4
CS
D3 ... 0
D 7 ..4
D3...0
BE1
CS
CSH0
CSH1
CSH7
CSH0
CSH7
D 15..12
D11..8

Задатак 22.

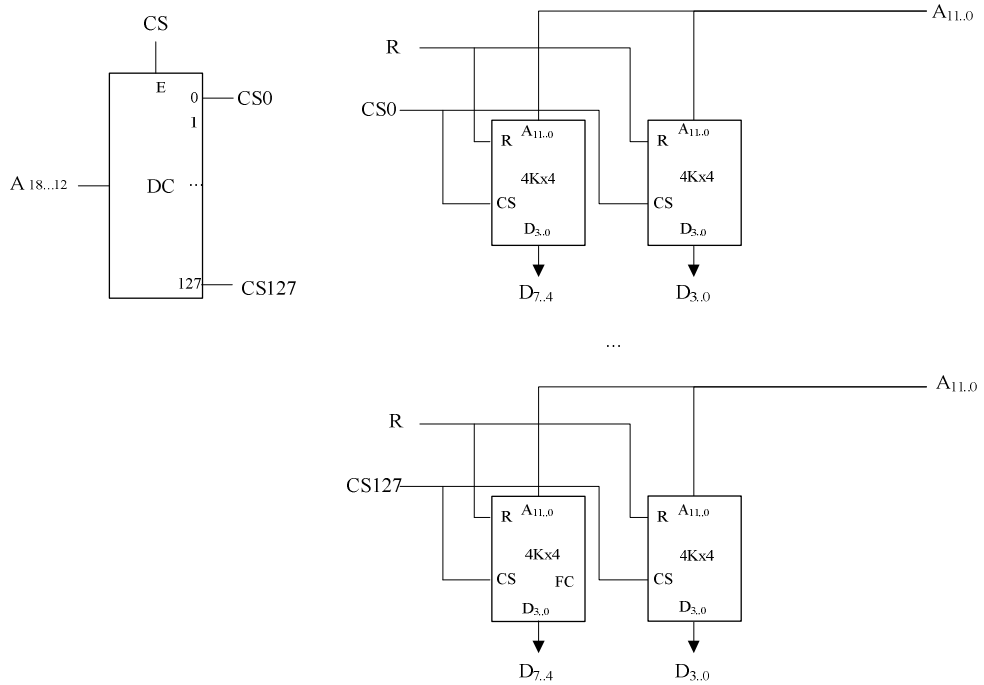
Посматра се 32-битни *big-endian* (старији бајт се смешта на нижу адресу) процесор (са 32-битном адресном магистралом и 32-битном магистралом података). Најмања адресибилна јединица је бајт. Потребно је пројектовати ROM и RAM меморију. ROM меморија заузима најниже адресе у меморијском адресном простору, а RAM меморија заузима адресе које следе ROM меморију. ROM меморија је капацитета 1MB и ширине 16 бита, користити 4Kx4 бита чипове. RAM меморија је капацитета 8MB и ширине 32-бита, користити 16Kx2 бита чипове. Приказати везу меморијских модула са адресном и магистралом података. Не постоји улазно-излазни адресни простор.

Решење

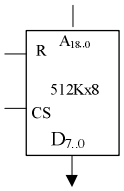
Шта?	Прва адреса	Последња адреса
Адресни простор резервисан за ROM	00000000h	000FFFFFh
Адресни простор резервисан за RAM	00100000h	008FFFFFh

Прво ћемо креирати ROM модул капацитета 512Kx8 бита коришћењем модула капацитета 4Kx4 бита. Реализација је представљена на слици.

$$numRom = \frac{ROMSize}{ROMChipSize} = \frac{512Kx8бита}{4Kx4бита} = \frac{512Kx8бита}{2Kx8бита} = 256$$

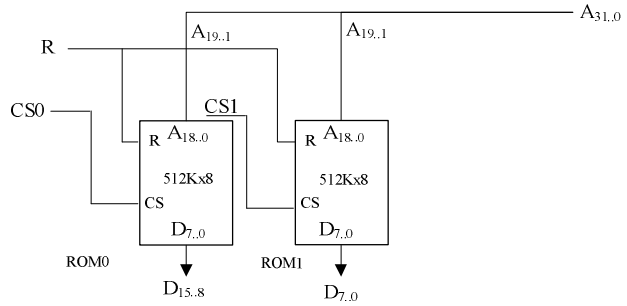


Креирани модул капацитета 512Kx8 бита је приказан на слици.



Коришћењем креираног модула креира се ROM меморија на начин приказан на слици. У случају када је генерисан циклус читања и на адресним линијама A_{20} до A_{31} се налази вредност нула у питању је циклус читања из ROM меморије и активан је сигнал **ROMENABLE**. Сигнал **BE1** омогућава везу меморијског модула и линија података

$D_{15..8}$, а сигнал **BE0** везу меморијског модула и линија података $D_{7..0}$. Непарне меморијске локације се пресликавају у модул ROM0 (1h, 3h, 5h, 7h, 9h,..), а парне меморијске локације се пресликавају у ROM1 (0h, 2h, 4h, 6h, 8h,..).



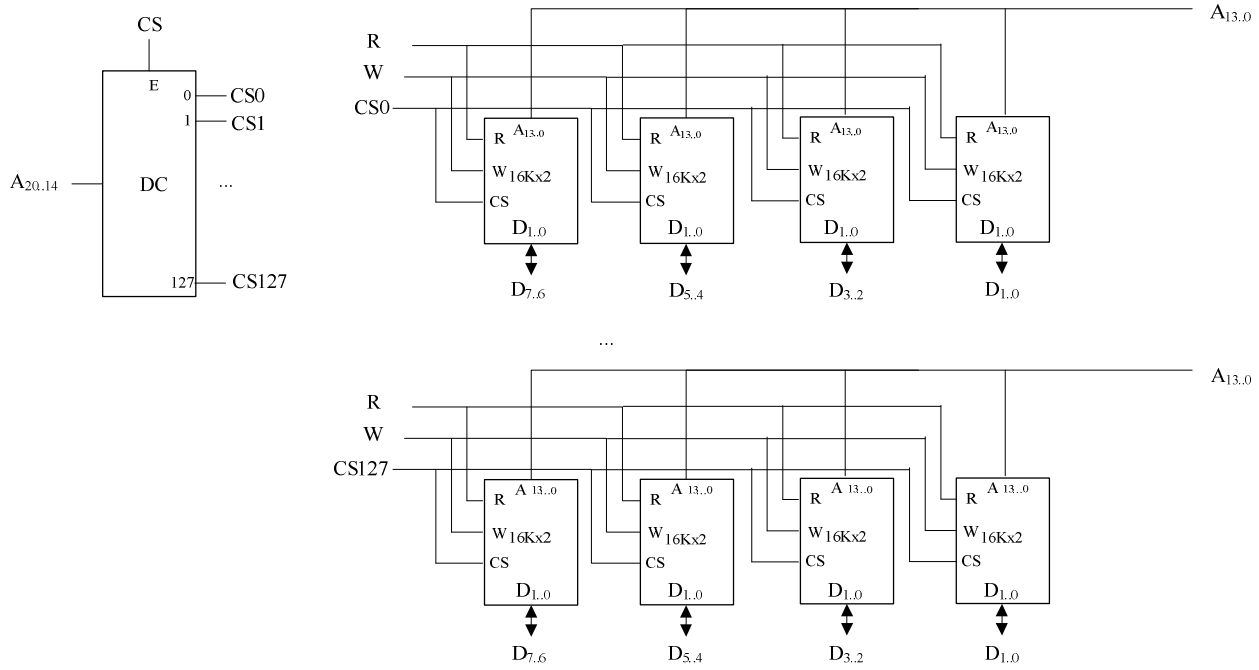
ROMENABLE = $\overline{A}_{31}...\overline{A}_{20}$

CS0 = ROMENABLE · BE1

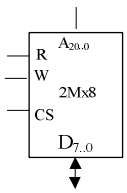
CS1 = ROMENABLE · BE0

Креирање RAM модула 2Mx8бита коришћењем модула 16Kx2 бита приказано је на слици.

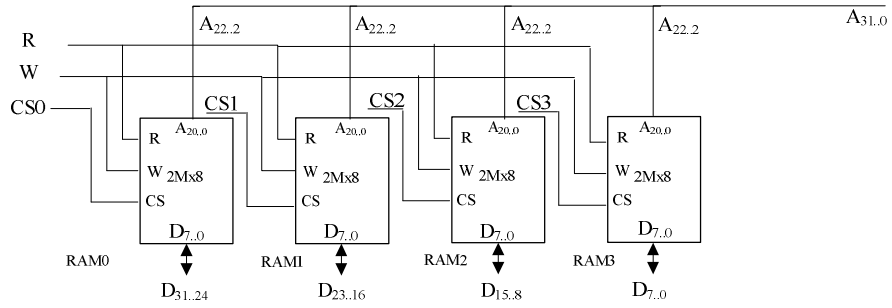
$$\text{numRam} = \frac{\text{RAMSize}}{\text{RAMChipSize}} = \frac{\text{2Mx8бита}}{\text{16Kx2бита}} = \frac{\text{2Mx8бита}}{\text{4Kx8бита}} = 512$$



Креирани модул 2Mx8бита RAM меморије је представљен на слици.



Коришћењем креираног модула креира се RAM меморија приказана на слици.



$$\text{RAMENABLE} = \overline{A}_{31}...\overline{A}_{24}(\overline{A}_{23}A_{22} + \overline{A}_{23}A_{21} + \overline{A}_{23}A_{20} + A_{23}A_{22}\overline{A}_{21}\overline{A}_{20})$$

CS0 = RAMENABLE · BE3

CS1 = RAMENABLE · BE2

CS2 = RAMENABLE · BE1

CS3 = RAMENABLE · BE0

Задатак 23.

Посматра се big-endian процесор повезан на магистралу са 24-битним адресним линијама и 16-битним линијама података. Најмања адресибилна јединица је бајт.

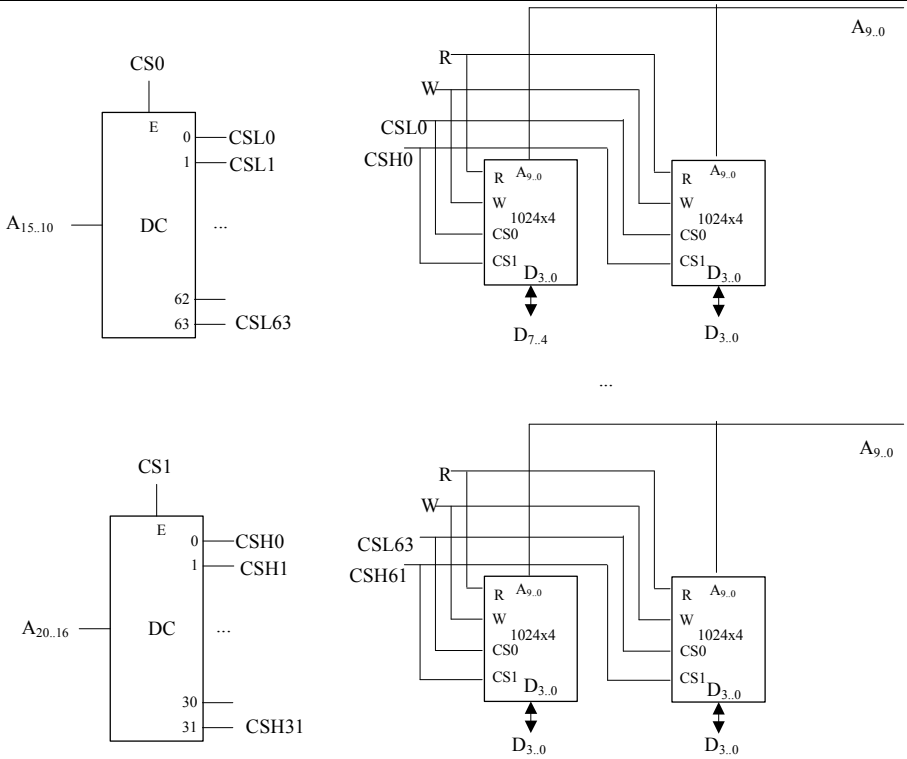
а) Пројектовати 2 MB меморије користећи 1024x4бита SRAM чипове распоређених у две димензије. Контролни улази су R, W и два улаза за селекцију чипа CS0 и CS1.

б) Користећи меморију пројектовану у тачки а) пројектовати 4MB меморије и приказати њену везу са магистралом.

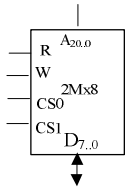
Решење:

а) Креирање RAM меморијског модула од 2Mx8бита коришћењем модула 1024x4бита приказано је на слици.

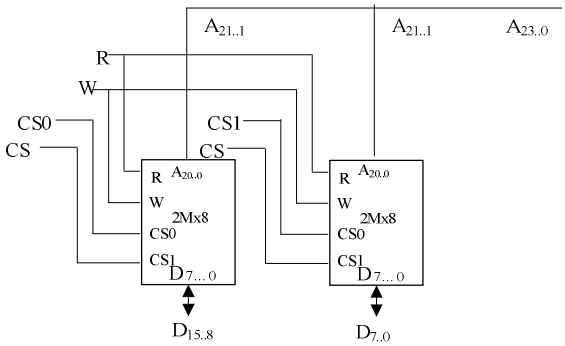
$$\text{numRam} = \frac{\text{RAMSize}}{\text{RAMChipSize}} = \frac{\text{2Mx8бита}}{\text{1024x4бита}} = \frac{\text{2Mx8бита}}{\text{512x8бита}} = 2048$$



Креирани модул од од 2Mx8бита представљен је на слици.



б) Користећи два модула од 2Mx8бита креирана у тачки а) креирана је меморија капацитета 4Mx8бита и приказана на слици.



CS0 = BE1

CS1 = BE0

RAM0 (0h, 2h, 4h, 6h, 8h, …)

RAM1 (1h, 3h, 5h, 7h, 9h, …)

Задатак 24.

Једноадресни процесор, меморија и периферија повезани су 16-битном адресном и 8-битном магистралом података. Садржај дела оперативне меморије дат је на слици.

Адреса	0000h	0001h	0002h	0003h	0004h	0005h	0006h	0007h	0008h	0009h	000Ah
Садржај	14h	0Ah	05h	00h	32h	15h	31h	FFh	0Ah	00h	16h
Адреса	000Bh	000Ch	000Dh	000Eh							
Садржај	01h	F0h	00h	10h							

Процесор поседује 16-битне регистре PC (програмски бројач), SP (показивач на прву слободну локацију стека који расте према нижим адресама), A (акумулатор), и AR (адресни регистар). При позиву потпрограма на стеку се чува само PC. Први бајт инструкције увек садржи само кôд операције, а други начин адресирања. Главни програм и потпрограм који се извршава позивом JSR дати су на слици. Претпоставити да је пре почетка извршавања главног програма SP = DEE0h и AR = 9, а да се виши бајт 16-битне речи смешта на нижу адресу.

```
адреса   инструкција          коментар
0000h    LOADW PC(05h)        ; relativno adresiranje sa 8-bitnim pomerajem
0003h    ADDB(AR)+            ; autoinkrement adresiranje
0005h    JSR(AR)              ; skok u potprogram, registarsko indirektno
0007h    HALT                 ; zaustavljanje procesora

;potporogram
X        OUTB F000h           ; memorijsko direktno
X+4      RTS                  ; povratak iz potprograma
```

а) Чему је једнако X?

б) Навести секвенцу садржаја на адресној магистрали, магистрали података и контролној магистрали за сваки циклус на магистрали при извршавању програма приказаног на слици.

Решење:

а) Потпрограм почиње на адреси која представља садржај адресног регистра AR при извршавању инструкције JSR. Претходна инструкција је инкрементирала овај регистар, док је AR на почетку имао вредност 9. Према томе, X = 10 = 000Ah.

б) Секвенца адреса које се генеришу на адресној магистрали при извршавању датог дела програма је дата у табели.

Цк	Адресна маг.	Маг. података	R	W	M/$\overline{IO}$	Коментар
1	0000h	14h	1	0	1	Фаза читања инструкције LOADW PC(5), први бајт
2	0001h	0Ah	1	0	1	Фаза читања инструкције LOADW PC(5), други бајт
3	0002h	05h	1	0	1	Фаза читања инструкције LOADW PC(5), трећи бајт
4	0008h	0Ah	1	0	1	Фаза читања операнда LOADW PC(5) , виши бајт
5	0009h	00h	1	0	1	Фаза читања операнда LOADW PC(5), нижи бајт ACC=0A00h
6	0003h	00h	1	0	1	Фаза читања инструкције ADDB(AR)+, први бајт
7	0004h	32h	1	0	1	Фаза читања инструкције ADDB(AR)+, други бајт
8	0009h	00h	1	0	1	Фаза читања операнда ADDB(AR)+
9	0005h	15h	1	0	1	Фаза читања инструкције JSR(AR), први бајт
10	0006h	31h	1	0	1	Фаза читања инструкције JSR(AR), други бајт
11	DEE0h	07h	0	1	1	Фаза извршавања инструкције JSR(AR), млађи бајт PC на стек
12	DEDFh	00h	0	1	1	Фаза извршавања инструкције JSR(AR), старији бајт PC на стек
13	000Ah	16h	1	0	1	Фаза читања инструкције OUTB F000h, први бајт
14	000Bh	01h	1	0	1	Фаза читања инструкције OUTB F000h, други бајт
15	000Ch	F0h	1	0	1	Фаза читања инструкције OUTB F000h, трећи бајт
16	000Dh	00h	1	0	1	Фаза читања инструкције OUTB F000h, четврти бајт
17	F000h	00h	0	1	0	Фаза извршавања инструкције OUTB F000h
18	000Eh	10h	1	0	1	Фаза читања инструкције RTS
19	DEDFh	00h	1	0	1	Фаза извршавања инструкције RTS, млађи бајт PC са стека
20	DEE0h	07h	1	0	1	Фаза извршавања инструкције RTS, старији бајт PC са стека
21	0007h	FFh	1	0	1	Фаза читања инструкције HALT

Задатак 25.

Посматра се двоадресни процесор који је повезан са меморијом и улазно/излазним уређајима преко асинхроне магистрале. Адресни простор је капацитета 32MB, а адресибилна јединица је 16-битна реч (W). Улазно/излазни адресни простор је меморијски раздвојен. Садржај дела оперативне меморије дат је на слици.

Адреса	0000h	0001h	0002h	0003h	0004h	0005h	0006h	0007h	0008h	0009h	000Ah
Садржај	A000h	0000h	0000h	E004h	F000h	E028h	0000h	0009h	0500h	0050h	0000h

Навести секвенцу садржаја на адресној магистрали, магистрали података и контролној магистрали за сваки циклус на магистрали при извршавању програма приказаног на слици. Регистри опште намене су дужине 16 бита.

```
адреса   инструкција          коментар
0000h   LOAD R1,0           ; memorijsko direktno adresiranje
0003h   OUT PC(F000h),R1    ; relativno adresiranje sa 16-bitnim pomerajem
0005h   STORE (9h),R1       ; memorijsko indirektno adresiranje.
```

Решење

Секвенца адреса које се генеришу на адресној магистрали при извршавању датог дела програма је дата на слици.

Цк	Адресна маг.	Маг. података	R	W	M/$\overline{IO}$	Коментар
1	00 0000h	A000h	1	0	1	Фаза читања инструкције LOAD R1,0, прва реч
2	00 0001h	0000h	1	0	1	Фаза читања инструкције LOAD R1,0, друга реч
3	00 0002h	0000h	1	0	1	Фаза читања инструкције LOAD R1,0, трећа реч
4	00 0000h	A000h	1	0	1	Фаза читања операнда LOAD R1,0
5	00 0003h	E004h	1	0	1	Фаза читања инструкције OUT PC(F000h),R1, прва реч
6	00 0004h	F000h	1	0	1	Фаза читања инструкције OUT PC(F000h),R1, друга реч
7	FF F005h	A000h	0	1	0	Фаза извршавања инструкције OUT PC(F000h),R1
8	00 0005h	E028h	1	0	1	Фаза читања инструкције STORE (9h),R1, прва реч
9	00 0006h	0000h	1	0	1	Фаза читања инструкције STORE (9h),R1, друга реч
10	00 0007h	0009h	1	0	1	Фаза читања инструкције STORE (9h),R1, трећа реч
11	00 0009h	0050h	1	0	1	Фаза дохватања адресе операнда STORE (9h),R1, старија реч
12	00 000Ah	0000h	1	0	1	Фаза дохватања адресе операнда STORE (9h),R1, млађа реч
13	50 0000h	A000h	0	1	1	Фаза извршавања инструкције STORE (9h),R1

Задатак 26.

Двоадресни процесор са 4 16-битна регистра опште намене (R0-R3) повезан је са меморијом преко асинхроне магистрале са 16 адресних линија, 8 линија за податке и одговарајућим управљачким линијама. Адресибилна јединица је бајт. Садржај дела оперативне меморије дат је на слици.

Адреса	0000h	0001h	0002h	0003h	0004h	0005h	0006h	0007h		
Садржај	00h	30h	30h	0Eh	0Eh	30h	30h	00h		
Адреса	3000h	3001h	3002h	3003h	3004h	3005h	3006h	3007h	3008h	3009h
Садржај	05h	11h	0Ah	30h	05h	22h	0Ch	44h	02h	FFh
Адреса	300Ah	300Bh	300Ch	300Dh	300Eh	300Fh	3010h	3011h	3012h	3013h
Садржај	00h	00h	05h	30h	00h	02h	04h	F0h	12h	C0h

При позиву потпрограма или прекидне рутине на стеку се чува само PC. Стек расте ка вишим меморијским адресама а SP указује на прву слободну локацију на врху стека. Претпоставити да је пре почетка извршавања сегмента програма са слике, SP=4000h, IVTP=0h, PC=3000h. Навести секвенцу садржаја на адресној магистрали, магистрали података и контролној магистрали за сваки циклус на магистрали при извршавању тог програма.

```
адреса   инструкција          коментар
3000h    LOAD R1, (300Ah)     ; memorijski indirektno adresiranje
3004h    LOAD R2, (R1)0Ch     ; registarski indirektno sa 8-bitnim pomerajem
3007h    INT #2               ; softverski prekid
3009h    HALT                 ; zaustavljanje procesora
...
300Eh    ADD R2, #4           ; neposredno adresiranje
3011h    PUSH R2              ; registarski direktno adresiranje
3013h    RTI                  ; povratak iz prekidne rutine
```

Решење

Цк	Адресна маг.	Маг. података	R	W	M/$\overline{IO}$	Коментар
1	3000h	05h	1	0	1	Фаза читања инструкције LOAD R1, (300Ah), први бајт
2	3001h	11h	1	0	1	Фаза читања инструкције LOAD R1, (300Ah), други бајт
3	3002h	0Ah	1	0	1	Фаза читања инструкције LOAD R1, (300Ah), трећи бајт
4	3003h	30h	1	0	1	Фаза читања инструкције LOAD R1, (300Ah), четврти бајт
5	300Ah	00h	1	0	1	Фаза читања адресе операнда LOAD R1, (300Ah), први бајт
6	300Bh	00h	1	0	1	Фаза читања адресе операнда LOAD R1, (300Ah), други бајт
7	0000h	00h	1	0	1	Фаза читања операнда LOAD R1, (300Ah), млађи бајт
8	0001h	30h	1	0	1	Фаза читања операнда LOAD R1, (300Ah), старији бајт R1=3000h

9	3004h	05h	1	0	1	Фаза читања инструкције LOAD R2, (R1)0Ch, први бајт
10	3005h	22h	1	0	1	Фаза читања инструкције LOAD R2, (R1)0Ch, други бајт
11	3006h	0Ch	1	0	1	Фаза читања инструкције LOAD R2, (R1)0Ch, трећи бајт
12	300Ch	05h	1	0	1	Фаза читања операнда LOAD R2, (R1)0Ch, млађи бајт
13	300Dh	30h	1	0	1	Фаза читања операнда LOAD R2, (R1)0Ch, старији бајт R2=3005h
14	3007h	44h	1	0	1	Фаза читања инструкције INT #2, први бајт
15	3008h	02h	1	0	1	Фаза читања инструкције INT #2, други бајт
16	4000h	09h	0	1	1	Опслуживање инструкције прекида, млађи бајт PC-а на стек
17	4001h	30h	0	1	1	Опслуживање инструкције прекида, млађи бајт PC-а на стек
18	0004h	0Eh	1	0	1	Читање млађег бајта адресе почетка прекидне рутине
19	0005h	30h	1	0	1	Читање старијег бајта адресе почетка прекидне рутине
20	300Eh	00h	1	0	1	Фаза читања инструкције ADD R2,#4, први бајт
21	300Fh	02h	1	0	1	Фаза читања инструкције ADD R2,#4, други бајт
22	3010h	04h	1	0	1	Фаза читања инструкције ADD R2,#4, трећи бајт R2=3009h
23	3011h	F0h	1	0	1	Фаза читања инструкције PUSH R2, први бајт
24	3012h	12h	1	0	1	Фаза читања инструкције PUSH R2, други бајт
25	4002h	09h	0	1	1	Фаза извршавања инструкције PUSH R2, млађи бајт R2 на стек
26	4003h	30h	0	1	1	Фаза извршавања инструкције PUSH R2, старији бајт R2 на стек
27	3013h	C0h	1	0	1	Фаза читања инструкције RTI
28	4003h	30h	1	0	1	Фаза извршавања инструкције RTI, старији бајт PC са стека
29	4002h	09h	1	0	1	Фаза извршавања инструкције RTI, млађи бајт PC са стека
30	3009h	FFh	1	0	1	Фаза читања инструкције HALT

Задатак 27.

Посматра се 32-битни *big-endian* процесор (млађи бајтови се смештају на вишу адресу), са 32-битном магистралом података и 29-битном адресном магистралом. Најмања адресибилна јединица је бајт. Не постоји улазно-излазни адресни простор. Процесор извршава инструкцију **STORE** којом се операнд AABBCCDDh смешта на меморијску локацију 12345676h.

а) Навести секвенцу садржаја на адресној магистрали, магистрали података и контролној магистрали у току фазе извршавања дате инструкције када је меморија повезана на линије података $D_{31..0}$.

б) Навести секвенцу садржаја на адресној магистрали, магистрали података и контролној магистрали у току фазе извршавања дате инструкције када је меморија повезана на линије података $D_{15..0}$.

Решење

MEM[12345676] = AAh
MEM[12345677] = BBh
MEM[12345678] = CCh
MEM[12345679] = DDh

а)

Цк	Адресна маг.	$D_{0..7}$	$D_{15..8}$	$D_{23..16}$	$D_{31..24}$	R	W	$M/\overline{IO}$	BE3	BE2	BE1	BE0
1	12345674h	BBh	AAh	XX	XX	0	1	1	0	0	1	1
2	12345678h	XX	XX	DDh	CCh	0	1	1	1	1	0	0

б)

Цк	Адресна маг.	D_{0-7}	D_{15-8}	D_{23-16}	D_{31-24}	R	W	$M/\overline{IO}$	BE3	BE2	BE1	BE0
1	12345676h	BBh	AAh	XX	XX	0	1	1	0	0	1	1
2	12345678h	DDh	CCh	XX	XX	0	1	1	0	0	1	1

Задатак 28.

Посматра се једноадресни *little-endian* процесор повезан на магистралу са 32-битним адресним линијама и 64-битним линијама података. Најмања адресибилна јединица је бајт. Процесор извршава инструкцију **LOAD**, која врши учитавање 16-битног операнда са меморијске локације 1234567Ah. Ако је операнд AABBh навести секвенцу садржаја на адресној магистрали, магистрали података и контролној магистрали у току фазе дохватања операнда ове инструкције.

Решење:

Цк	Адресна маг.	$D_{0..7}$	$D_{15..8}$	$D_{23..16}$	$D_{31..24}$	$D_{39..32}$	$D_{47..40}$	$D_{55..48}$	$D_{63..56}$	Активни сигнали
1	12345676h	XX	XX	XX	AAh	BBh	XX	XX	XX	RD, BE2, BE3

Линије података од 0 до 15 и од 23 до 63 се не користе у овом трансферу.

Задатак 29.

У случају магистрале са већим бројем газди :

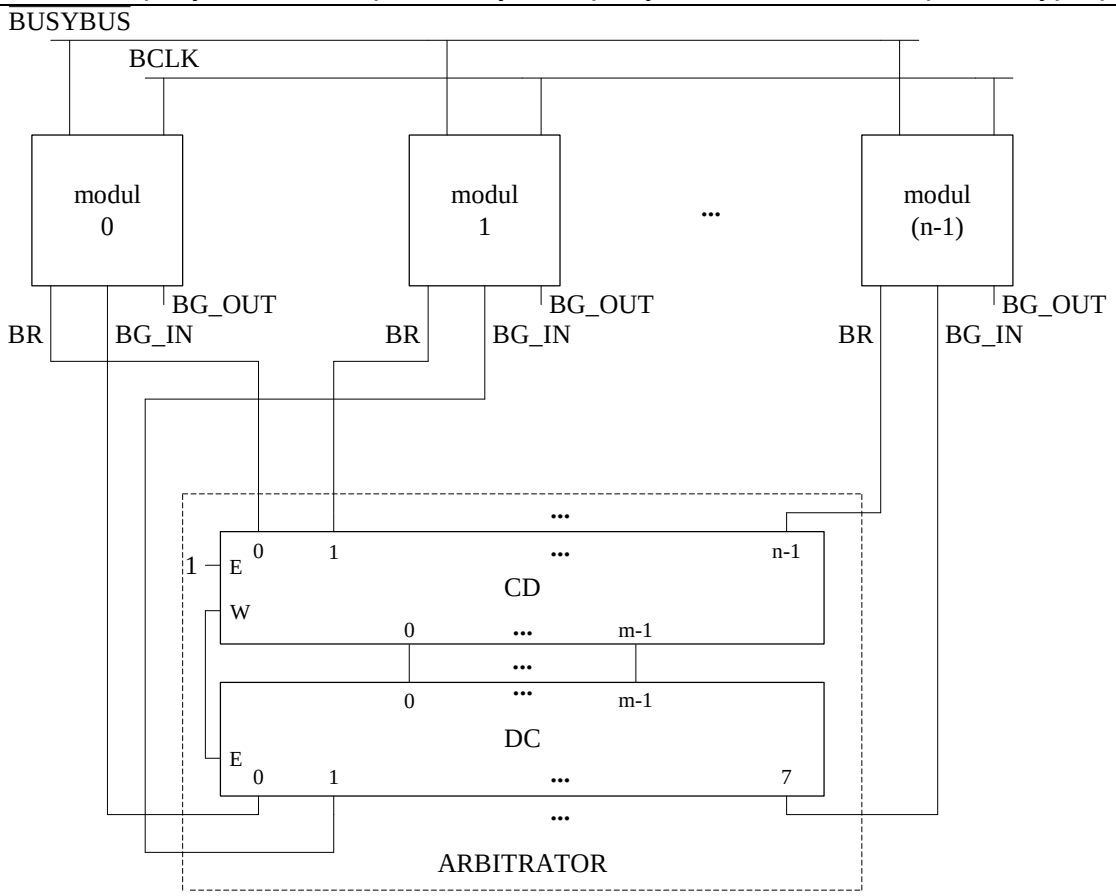
а) Нацртати и објаснити како се реализује паралелна шема за арбитрацију.

б) Нацртати и објаснити како се реализује серијска шема за арбитрацију.

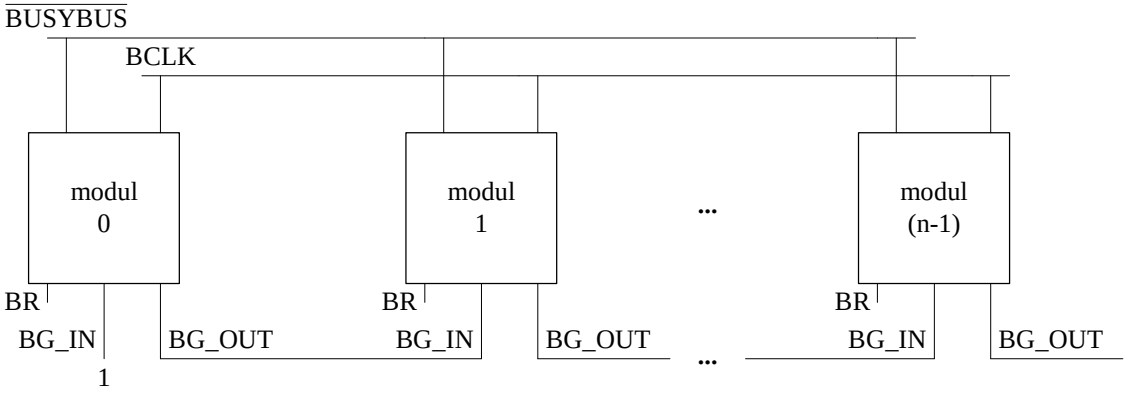
в) Арбитрација о следећем кориснику магистрале се обавља док текући корисник обавља пренос на магистрали. С обзиром да арбитрација траје краће од преноса, како се обезбеђује да онај који је добио дозволу да буде следећи корисник магистрале не крене са преносом док текући корисник магистрале не заврши свој пренос?

Решење

а)



б)

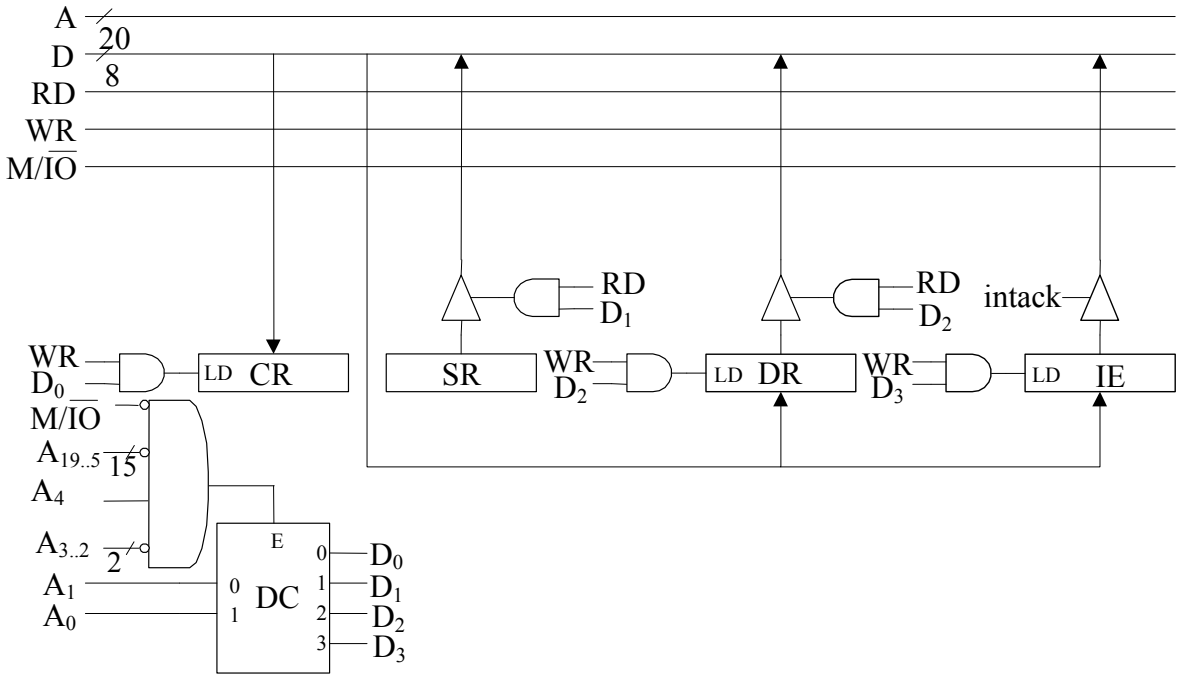


в) BUSYBUS забрањује излазак на магистралу док је активан.

Задатак 30.

Адресни простор једноадресног процесора је 1MB, а адресирање је бајтовско. Процесор поседује 16-битни акумулатор. Спрежни део процесора према магистрали садржи регистар MAR и 8-битни регистар MDR. Улазно/излазни и меморијски адресни простори су раздвојени. Процесор је повезан са меморијом MEM и периферијом PER преко синхроне магистрале. Нацртати интерну структуру контролера периферије и везу контролера са системском магистралом. Управљачки регистар CR се налази на адреси 10h, статусни регистар SR се налази на адреси 11h, регистар података DR на адреси 12h, а регистар IE у коме се чува број улаза у табелу прекидних рутина налази се на адреси 13h. Садржај регистра IE се износи на магистралу података ако је активан сигнал *intack*.

Решење



Задатак 31.

Посматра се двоадресни процесор са раздвојеним меморијским и улазно/излазним адресним просторима. Адресибилна јединица је 16-битна реч (W). Магистрала је асинхрона, а механизам прекида векторисан. Капацитет меморијског адресног простора је 8GB. За адресирање улазно/излазних уређаја резервисана је 1GW почев од адресе 0h. Навести релевантне регистре DMA контролера који је преко системске магистрале везан у посматрани систем. Прецизно назначити дужину сваког регистра; сваком регистру придружити адресе, ако се зна да је за овај контролер резервисано првих 10 адреса у адресном простору. Напомена: није потребно цртати везу DMA контролера са системском магистралом.

Решење

Име регистра	Дужина		Адресе
Адресни изворишни регистар (*Source*)	32	ARSH, ARSL	0h, 1h
Адресни одредишни регистар (*Destination*)	32	ARDH, ARDL	2h, 3h
бројачки регистар	32	CNTH, CNTL	4h, 5h
регистар података	16	DR	6h
статусни регистар	16	SR	7h
управљачки регистар	16	CR	8h
број улаза у IVT	16	IE	9h

Улаз/Излаз

Задатак 32.

Адресне линије и линије података магистрале посматраног рачунара су широке по 16 бита. Процесор је једноадресни и оперише само над 16 битним подацима. Улазно излазни и меморијски адресни простори су раздвојени. Периферије PER0 и PER1 имају управљачке, статусне и регистре података редом на следећим адресама: FF00h, FF01h, FF02h (PER0) и FF10h, FF11h, FF12h (PER1). У управљачким регистрима најмлађи бит је бит Enable Interrupt којим се дозвољава прекид, најстарији је бит Start којим се дозвољава почетак операције, а бит 1 је бит смера операције (0—улаз, 1—излаз). У статусним регистрима најмлађи бит је бит спремности Ready. 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 испитивањем бита спремности. Процедура Obrada не мења место ни дужину блока података. Ову процедуру не треба реализовати, већ је само позвати на одговарајућем месту у програму помоћу наредбе CALL Obrada.

Решење:

```
        LOAD   #200h        ;brojač u MemCnt
        STORE  MemCnt
        LOAD   #F000h       ;adresa u MemDst
        STORE  MemDst
        LOAD   #8000h       ;Start=1, Enable=0, Direction=0
        OUT    FF00h        ;pokreni kontroler
LOOP1:IN       FF01h        ;ispitivanje bita spremnosti
        AND    #1
        JZ     LOOP1        ;ako nije spreman, čekaj
        IN     FF02h        ;ulaz podatka
        STORE  (MemDst)     ;i smeštanje u memoriju
        LOAD   MemDst       ;ažuriranje pokazivača
        INC
        STORE  MemDst
        LOAD   MemCnt       ;i brojača
        DEC
        STORE  MemCnt
        JNZ    LOOP1        ;ako nije poslednji, ponovi
        LOAD   #0           ;zaustavi kontroler
        OUT    FF00h        ;Stop PER0
        CALL   Obrada       ;obrada
        LOAD   #200h        ;brojač u MemCnt
        STORE  MemCnt
        LOAD   #F000h       ;pokazivač u MemSrc
        STORE  MemSrc
        LOAD   #8002h       ;Start=1, Enable=0, Direction=1
        OUT    FF10h        ;pokretanje kontrolera
LOOP2:IN       FF11h        ;ispitivanje bita spremnosti
        AND    #1
        JZ     LOOP2        ;ako nije spreman, čekaj
        LOAD   (MemSrc)     ;podatak na izlaz
        OUT    FF12h
        LOAD   MemSrc       ;ažuriranje pokazivača
        INC
        STORE  MemSrc
        LOAD   MemCnt       ;i brojača
        DEC
        STORE  MemCnt
        JNZ    LOOP2        ;ako nije poslednji, ponovi
        LOAD   #0           ;zaustavi kontroler
        OUT    FF10h        ;Stop PER1
```

Задатак 33.

Једноадресни рачунар са 16 битном адресном и 8 битном магистралом података поседује меморијски пресликан улазно/излазни адресни простор. Периферије PER0, PER1 и PER2 имају управљачке, статусне и регистре података редом на следећим адресама: FF10h, FF11h, FF12h (PER0), FF20h, FF21h, FF22h (PER1) и FF30h, FF31h, FF32h (PER2). У управљачким регистрима најстарији бит је Start којим се дозвољава почетак операције, најмлађи је бит Enable којим се дозвољава прекид, а бит 3 је бит смера операције (1—улаз, 0—излаз). У статусним регистрима најстарији бит 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 локација 1000h (PER0) и 1100h (PER1), затим изврши обрада унетих блокова података позивом процедуре Obrada (CALL Obrada) и потом врши пренос обрађеног блока од 200h бајтова, почев од адресе 1000h из меморије у PER2. Обе улазне операције извршити коришћењем прекида, а излазну операцију испитивањем бита Ready.

Решење

```
        LOAD   #1000h;pokretanje obe ulazne operacije
        STORE  MemB0
        LOAD   #1100h
        STORE  MemB1
        LOAD   #100h
        STORE  MemCnt0
        STORE  MemCnt1
        LOAD   #0
        STORE  MemSem0
        STORE  MemSem1
        LOADB  #89h
        STOREB         FF10h
        STOREB         FF20h
Wait0:  LOAD MemSem0;čekanje na završetak sa PER0
        AND    #1
        JZ     Wait0
Wait1:  LOAD MemSem1;čekanje na završetak sa PER1
        AND    #1
        JZ     Wait1
        CALL   Obrada  ;obrada
        LOAD   #1000h  ;izlazna operacija
        STORE  MemB
        LOAD   #200h
        STORE  MemCnt
        LOADB  #80h
        STOREB         FF30h
Loop:   LOADB  FF31h
        AND    #80h
        JZ     Loop
        LOADB  (MemB)
        STOREB         FF32h
        INC    MemB
        DEC    MemCnt
        JNZ    Loop
        LOAD   #0
        STOREB         FF30h
Прекидне рутине:
Per0:   PUSH
        LOADB  FF12h
        STOREB         (MemB0)
        INC    MemB0
        DEC    MemCnt0
        JNZ    Back0
        LOAD   #0
        STOREB         FF10h
        INC
        STORE  MemSem0
Back0:  POP
        RTI

Per1:   PUSH
        LOADB  FF22h
        STOREB         (MemB1)
        INC    MemB1
        DEC    MemCnt1
        JNZ    Back1
        LOAD   #0
        STOREB         FF20h
        INC
        STORE  MemSem1
Back1:  POP
        RTI
```

Задатак 34.

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
```

Задатак 35.

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

Задатак 36.

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
        MOV   mcA,     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
        MOV   R2,      mwcC
        CMP   R2,      R1
        BLE   Wait
        INC   R1
        MOV   FF22h, (R1)3000
        CMP   R1,      #99
        JNZ   Wait
        MOV   FF20h,   #0
        HALT
```

Прекидна рутина за PER0:

```
        INTD
        PUSH  R0
        PUSH  R1
        PUSH  R2
        MOV   R0,      FF02h
        MOV   R1,      mcA
        INC   R1
        MOV   mcA,     R1
        MOV   (R1)1000,          R0
        CMP   R1,      mcB
        JG    SkipA
        ADD   R0,      (R1) 2000
        MOV   R2,      mwcC
        INC   R2
        MOV   mwcC,    R2
        MOV   (R2) 3000,         R0
SkipA:  CMP   R1,      #99
        JNZ   BackA
        MOV   FF00h,   #0
BackA:  POP   R2
        POP   R1
        POP   R0
        RTI
```

Прекидна рутина за PER1:

```
        INTD
        PUSH  R0
        PUSH  R1
        PUSH  R2
        MOV   R0,      FF12h
        MOV   R1,      mcB
        INC   R1
        MOV   mcB,     R1
        MOV   (R1) 2000,         R0
        CMP   R1,      mcA
        JG    SkipB
        ADD   R0,      (R1) 1000
        MOV   R2,      mwcC
        INC   R2
        MOV   mwcC,    R2
        MOV   (R2) 3000,         R0
SkipB:  CMP   R1,      #99
        JNZ   BackB
        MOV   FF10h,   #0
BackB:  POP   R2
        POP   R1
        POP   R0
        RTI
```

Задатак 37.

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
        …
```

Задатак 38.

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 stealing)).

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 burst 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

Задатак 39.

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
```

Задатак 40.

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
```

Задатак 41.

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

Задатак 42.

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
```

Copyright © 2012 Електротехнички факултет Универзитета у Београду