Лабораторијске вежбе из Архитектуре рачунара

У следећим табелама дате су адресе релевантних регистара контролера периферија, све адресе су дате у хексадецималном облику. Меморијски адресни простор и улазно излазни адресни простор су меморијски пресликани (мапирани).

Периферија	Control	Status	Entry	Data
KP1 (KP1.1)	F100	F101	F102	F103
KP1.2	F140	F141	F142	F143
KP1.3	F180	F181	F182	F183
KP1.4	F1C0	F1C1	F1C2	F1C3

Периферија	Control	Status	Entry	Data
KP2 (KP2.1)	F200	F201	F202	F203
KP2.2	F240	F241	F242	F243
KP2.3	F280	F281	F282	F283
KP2.4	F2C0	F2C1	F2C2	F2C3

Формат контролних регистара периферија:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												nivo	**start**	**enable**	**ulaz\ilaz**

Бит *start* служи за укључивање периферије вредношћу 1.

Бит *ulaz/izlaz* вредношћу 1 означава да се ради о улазу, а вредношћу 0 да се ради о излазу.

Бит *nivo* мора имати вредност 1 уколико периферија ради коришћењем прекида (уколико *enable* бит има вредност 1), у свим други случајевима има вредност 0.

У статусним регистрима контролера периферија најнижи бит је бит *ready*.

Формат статусних регистара периферија:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															ready

У наредној табели дате су адресе релевантних регистара контролера периферија са директним приступом меморији (DMA).

Контролер	Control	Entry	Count	AR1	AR2
DMA1.1 (DMA)	F000	F002	F004	F005	F006
DMA1.2	F040	F042	F044	F045	F046
DMA1.3	F080	F082	F084	F085	F086
DMA1.4	F0C0	F0C2	F0C4	F0C5	F0C6

Регистар *AR1* је изворишни адресни регистар, а *AR2* је одредишни адресни регистар. Регистар *Count* је бројачки регистар.

Формат контролних регистара контролера периферија са директним приступом меморији:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								burst /cycle_stealing	**inc\dec**	**update**	**mem**	**nivo**	**start**	**enable**	**ulaz\ilaz**

Бит *start* служи за укључивање периферије вредношћу 1.

Бит *ulaz/izlaz* вредношћу 1 означава да се ради о улазу (подаци се пребацују са периферије у меморију), а вредношћу 0 да се ради о излазу (подаци се пребацују из меморије на периферију). Бит *mem* уколико се ради о улазу или излазу мора имати вредност 0. Бит *mem* уколико има вредност један тада се ради о трансферу података из меморије у меморију.

Бит *nivo* мора имати вредност 1 уколико контролер периферије са директним приступом меморији ради коришћењем прекида (уколико *enable* бит има вредност 1).

Бит *burst /cycle_stealing* вредношћу 1 означава да се пренос врши у пакетском режиму, а вредношћу 0 да се пренос ради у појединачним циклусима на магистрали.

Бит *update* вредностима 0 и 1 означава да ли је задат режим рада без или са ажурирањем вредности изворишног адресног регистра *AR1*, респективно. Вредност дестинационог адресног регистра *AR2* увек се ажурира.

Бит *inc/dec* вредностима 0 и 1 означава да ли се вредност регистра *AR1* инкрементира или декрементира, респективно, уколико је бит *update* постављен. Вредност регистра *AR2* увек се инкрементира или декрементира на основу бита *inc/dec*.

Напомене:

- Сва решења су откуцана и преведена помоћу *SPasm* асемблера и могу се извшити коришћењем симулатора *SPECSeo* на начин описан у материјалима за другу лабораторијску вежбу;
- Фајл *VEZBA.ASM* на почетку има део који је ограничен коментарима `! inicijalizacija simulacije` и `! kraj inicijalizacije`, наредбе које се налазе између ова два коментара не треба мењати;
- Периферију (без DMA) која ради коришћењем механизма прекида није могуће искључити у тренутку када пошаље/прими последњи податак, него тек следећи пут када пошаље захтев за прекид;
- Потпрограми се пишу одмах након краја главног програма (испод наредбе `halt`) тако што се наведе лабела која представља име потпрограма;
- Аргументи потпрограму прослеђују се преко стека у обрнутом редоследу од редоследа у декларацији потпрограма, а са стека се уклањају након позива потпрограма (чишћење стека од аргумената врши позивалац, а не позвани);
- Тип `int` је ширине 16 бита, а сви показивачи су ширине 16 бита и адресибилна јединица је ширине 16 бита.
- Стек расте од виших ка нижим локацијама, а регистар SP указује на последњу заузету локацију.

Задатак 1

Написати програм који са периферије KP1 учитава низ података дужине Ah и смешта у меморију почев од адресе 1000h, а затим учитани низ шаље периферији KP2. Учитавање реализовати испитивањем бита спремности, а слање коришћењем механизма прекида.

Потребно је на почетку главног програма иницијализовати IV табелу тако да се у улазу 1 налази адреса 2000h и тај улаз одговара периферији KP2. IV табелу сместити у меморију почев од адресе 0h. Главни програм почиње од адресе 100h.

Решење:

```
! inicijalizacija simulacije
onkp false, x.1, x.1
onkp true, x.1, x.1
kpreg 1.1, r0, x.1
kpreg 1.1, r1, x.2
kpreg 1.1, r2, x.3
kpreg 1.1, r3, x.4
kpreg 1.1, r4, x.5
kpreg 1.1, r5, x.6
kpreg 1.1, r6, x.7
kpreg 1.1, r7, x.8
kpreg 1.1, r8, x.9
kpreg 1.1, r9, x.a
dc x.55, x.3000
! kraj inicijalizacije

! resenje
! glavni program
org x.100

! postavljanje IV tabele
clr r0
mvrir r0, ivtp            ! ivtp = 0
ldimm x.2000, r0
stmem x.1, r0             ! postavljanje vrednosti 2000h u ulaz 1 IV tabele

! ucitavanje sa KP1
ldimm x.1000, r0          ! r0 pokazivac gde treba smestiti ucitan podatak
ldimm x.a, r1             ! r1 brojac koliko je jos podataka preostalo
ldimm x.5, r3
stmem x.f100, r3          ! upisivanje vrednosti 5h u kontrolni registar KP1

! ispitivanje biti spremnosti
ldimm x.1, r3                  ! maska za proveru bita spremnosti
loop: ldmem x.f101, r4         ! citanje statusnog registra KP1
      and r4, r4, r3           ! provera statusnog bita
      beql loop                ! ako je rezultat prethodne and instrukcije 0,
                               ! znaci da bit spremnosti nije postavljen
```

```
                              ! i ponovo se ispituje

! bit spremnosti je postavljen
     ldmem x.f103, r5         ! u r5 se prebacuje podatak sa periferije KP1
     stri [r0], r5            ! smesta primljeni podatak u memoriju
     inc r0                   ! povecava se pokazivac
     dec r1                   ! povecava se brojac
     bneq loop                ! ako nije stigao do nule
                              ! ima jos podataka za slanje

! iskljucivanje KP1
clr r0
stmem x.f100, r1

! priprema za transfer na KP2
ldimm x.1000, r0          ! pokazivac na niz
ldimm x.b, r1             ! brojac + 1, jer se na pocetku
                          ! pr. rut. radi dekrementiranje
clr r2                    ! r2 semafor
ldimm x.1, r3             ! r3 broj ulaza za KP2
stmem x.f202, r3          ! slanje broja KP2 u entry registar
ldimm x.e, r3             ! vrednost za pokretanje KP2
stmem x.f200, r3          ! upis u kontrolni registar KP2

ldimm x.1, r3             ! za testiranje semafora
wait: and r2, r2, r3
      beql wait           ! ako je r2 nula jos uvek prenos nije zavrsen

! kraj glavnog programa
halt

! prekidna rutina KP2, nalazi u memoriji od adrese 2000h
org x.2000
dec r1                    ! smanjivanje brojaca
bneq prenos
stmem x.f200, r1          ! iskljucivanje periferike KP2, svi podaci poslati
ldimm x.1, r2             ! postavljanje semafora da je sve poslato
jmp back
prenos:
ldrid [r0]x.0, r4         ! element niza u r4
stmem x.f203, r4          ! upisavanje elementa u data registar KP2
inc r0                    ! povecavanje pokazivaca
back:
rti                       ! izlazak iz pr. rut
```

Задатак 2

Написати програм који упоредо са периферија КР1.1 и КР1.2 учитава два низа A(i) (i=0..4) и B(j)(j=0..4). Низ A(i) учитава се са КР1.1 испитивањем бита спремност и смешта у меморију почев од адресе 100h. Низ B(i) учитава се са КР1.2 коришћењем механизма прекида и смешта у меморију почев од адресе 200h. Прекидну рутину КР1.2 сместити у меморију почев од адресе 2000h. Након пријема низова потребно је позвати процедуру `saberi` коју треба написати и сместити у меморију од адресе 3000h. Ова процедура треба да израчуна низ C(k)(k=0..4) који се смешта у меморију почев од адресе 300h, где је C(k)=A(k)+B(k). Након позива процедуре `saberi` потребно је низ C(k) послати периферији КР2 испитивањем бита спремности.

Потребно је на почетку главног програма иницијализовати IV табелу тако да се у улазу 2 налази адреса 2000h и тај улаз одговара периферији КР1.2. IV табелу сместити у меморију почев од адресе 0h. Главни програм почиње од адресе 1000h.

Решење:

```
! inicijalizacija simulacije
onkp false, x.1, x.1
onkp false, x.1, x.1
onkp true, x.1, x.1
kpreg 1.1, r0, x.1
kpreg 1.1, r1, x.2
kpreg 1.1, r2, x.3
kpreg 1.1, r3, x.4
kpreg 1.1, r4, x.5
kpreg 1.2, r0, x.6
kpreg 1.2, r1, x.7
kpreg 1.2, r2, x.8
kpreg 1.2, r3, x.9
kpreg 1.2, r4, x.a
reg pc, x.1000
reg sp, x.500
! kraj inicijalizacije

! resenje
! glavni program
org x.1000

! postavljanje IV tabele
clr r0
mvrir r0, ivtp          ! ivtp = 0
ldimm x.2000, r0
stmem x.2, r0           ! postavljanje vrednosti 2000h u ulaz 1 IV tabele

! pokretanje KP1.1
ldimm x.100, r0         ! r0 pokazivac gde treba smestiti niz A
ldimm x.5, r1           ! r1 brojac koliko je jos elemenata niza A preostalo
ldimm x.5, r3
```

```
stmem x.f100, r3          ! upisivanje vrednosti 5h
                          ! u kontrolni registar KP1.1

! porketanje KP1.2
ldimm x.2, r3             ! broj ulaza u IV za KP1.2
stmem x.f142, r3          ! upisivanje broja ulaza u entry registar KP1.2
ldimm x.200, r4           ! r4 pokazivac gde treba smestiti niz B
ldimm x.6, r5             ! r5 broj + 1 koliko
                          ! je jos elemenata niza B preostalo
ldimm x.f, r3             ! vrednost za starovanje KP1.2
clr ra                    ! semafor
stmem x.f140, r3          ! upis u kontrolni registar KP1.2

! ispitivanje biti spremnosti KP1.1
ldimm x.1, r3                 ! maska za proveru bita spremnosti
loop: ldmem x.f101, rc        ! citanje statusnog registra KP1
      and rc, rc, r3          ! provera statusnog bita
      beql loop               ! ako je rezultat prethodne and instrukcije 0,
                              ! znaci da bit spremnosti nije postavljen i
                              ! ponovo se ispituje

! bit spremnosti je postavljen
     ldmem x.f103, r7         ! u r7 se prebacuje podatak sa periferije KP1
     stri [r0], r7            ! smesta primljeni podatak u memoriju
     inc r0                   ! povecava se pokazivac
     dec r1                   ! smanjuje se brojac
     bneq loop                ! ako nije stigao do nule
                              ! ima jos podataka za slanje
! iskljucivanje KP1
clr r0
stmem x.f100, r0

! cekanje da se sigurno primi niz B
ldimm x.1, r3                 ! za testiranje semafora
wait: and ra, ra, r3
      beql wait

jsr saberi                    ! poziv potprograma

! pokretanje KP2
ldimm x.300, r0           ! r0 pokazivac gde se nalazi niz C
ldimm x.5, r1             ! r1 brojac koliko je jos elemenata niza C preostalo
ldimm x.4, r3
stmem x.f200, r3          ! upisivanje vrednosti 4h u kontrolni registar KP2
```

```
ldimm x.1, r3             ! maska za proveru bita spremnosti
loop2: ldmem x.f201, r4   ! citanje statusnog registra KP1
       and r4, r4, r3     ! provera statusnog bita
       beql loop2         ! ako je rezultat prethodne and instrukcije 0,
                          ! znaci da bit spremnosti nije postavljen i
                          ! ponovo se ispituje

! bit spremnosti je postavljen
     ldrid [r0]x.0, r7        ! u r7 se prebacuje jedan element niza C
     stmem x.f203, r7         ! element se prebacuje u data registar KP2
     inc r0                   ! povecava se pokazivac
     dec r1                   ! smanjuje se brojac
     bneq loop2               ! ako nije stigao do nule ima
                              ! jos elemenata za slanje
! iskljucivanje KP2
clr r0
stmem x.f200, r0

! kraj glavnog programa
halt

! procedura saberi
saberi:
push r0                       ! cuvanje registara koji se menjaju
push r1
push r2
push r3
push r5
push r6
push r7
ldimm x.100, r0               ! ucitavanje adrese niza A
ldimm x.200, r1               ! ucitavanje adrese niza B
ldimm x.300, r2               ! ucitavanje adrese niza C
ldimm x.5, r3                 ! broj elemenata niza

loop3: ldrid [r0]x.0, r5      ! ucitava se element niza A
      ldrid [r1]x.0, r6       ! ucitava se element niza B
      add r7, r5, r6          ! sabira se a[i] + b[i]
      stri [r2], r7           ! smesta se formirani element niza C
      inc r0                  ! povecava se pokazivac niza A
      inc r1                  ! povecava se pokazivac niza B
      inc r2                  ! povecava se pokazivac niza C
      dec r3                  ! smanjuje se brojac
      bneq loop3              ! ako ima jos elemenata skace se na loop3

! povratak iz potprograma (procedure)
pop r7
pop r6
pop r5
```

```
pop r3
pop r2
pop r1
pop r0
rts

! prekidna rutina KP1.2, nalazi u memoriji od adrese 2000h
org x.2000
dec r5              ! smanjivanje brojaca
bneq prenos
stmem x.f140, r5    ! iskljucivanje periferike KP2, svi podaci poslati
ldimm x.1, ra       ! postavljanje semafora da je sve poslato
jmp back

prenos:
ldmem x.f143, rb    ! element niza B u rb
stri [r4], rb       ! smestanje elementa niza u memorju
inc r4              ! povecavanje pokazivaca
back:
rti                 ! izlazak iz pr. rut
```

Задатак 3

Написати програм који учитава низ са периферије KP1 коришћењем механизма прекида. Низ треба сместити у меморију почев од адресе 1000h. Периферија прво шаље број елемената низа (величину низа), а након тога елементе. Након пријема потребно је позвати функцију **int** `max(`**int*** `array,` **int** `size)`, први параметар је адреса низа, а други број елемената у низу. Повратна вредност функције је највећи број у низу. Повратну вредност функција оставља у регистру R0. Након позива функције max треба послати највећи елемент периферији KP1

Поред главног програма, потребно је написати прекидну рутину која одговара периферији KP1 и функцију `max`. Главни програм треба сместити почев од адресе 100h, а прекидну рутину од адресе 2000h. Табела прекидних рутина је већ учитана у меморију, а периферији KP1 је већ постављен број улаза унутар *entry* регистру.

Решење:

```
! inicijalizacija simulacije
onkp false, x.1, x.1
kpreg 1.1, r0, x.a
kpreg 1.1, r1, x.11
kpreg 1.1, r2, x.53
kpreg 1.1, r3, x.22
kpreg 1.1, r4, x.24
kpreg 1.1, r5, x.23
kpreg 1.1, r6, x.ad
kpreg 1.1, r7, x.78
kpreg 1.1, r8, x.35
kpreg 1.1, r9, x.12
kpreg 1.1, ra, x.64
reg pc, x.100
reg ivtp, x.0
dc x.2000, x.1
kpreg 1.1, interrupt, x.1
! kraj inicijalizacije

! resenje
! glavni program
org x.100
ldimm x.1, ra       ! fleg da li se radi o ulazu ili izlazu (1 ulaz, 0 izlaz)

! startovanje KP1
clr rb              ! fleg da li je primljena velicina niza
ldimm x.1000, r5    ! u r5 pokazivac na niz
clr rc              ! semafor, da li je primljen niz
ldimm x.f, r1       ! u r1 konstanta za starovanje KP1
stmem x.f100, r1    ! upis u kontrolni registar KP1

ldimm x.1, rd
wait1: cmp rc, rd   ! ceka se da se primi niz
bneq wait1
```

```
! priprema steka za poziv funkcije max(int* array, int size)
! parametri sa desna na levo se stavljaju na stek
push rf
ldimm x.1000, r0
push r0
! poziv funkcije
jsr max
! nakon poziva funkcije ocistiti stek od argumenata
pop rc
pop rc    ! sa steka su skinuta dva argumenta

clr rc              ! semafor, da li je poslat najveci element
clr ra              ! fleg da se sada radi o izlazu
ldimm x.2, r4       ! broj podataka za slanje uvecan za 1
ldimm x.e, r1       ! u r1 konstanta za starovanje KP1
stmem x.f100, r1    ! upis u kontrolni registar KP1

ldimm x.1, rd
wait2: cmp rc, rd   ! ceka se da se posalje najveci element
bneq wait2

! kraj glavnog programa
halt

! funkcija int max(int* array, int size)
! pre poziva na steku se moraju postaviti parametri funkcije
! sa desna na levo
max:
                    ! cuvaju se registri koji se menjaju
push r4             ! koristi se r4 kao bazni registar
mvrpl r4, sp        ! u r4 sp, na steku imamo sada
                    ! r4, retPC, array i size

push r1             ! koristi se r1
push r2             ! koristi se r2
push r3             ! koristi se r3

ldrid [r4]x.2, r1   ! r4 + 2 je adresa na kojoj se nalazi argument array
                    ! r1 = array
ldrid [r4]x.3, r2   ! r4 + 3 je adresa na kojoj se nalazi argument size
                    ! r2 = size

ldrid [r1]x.0, r0   ! u r0 se cuva najveci element
loop:
inc r1              ! prelazi se na sledeci element
dec r2              ! smanjuje se brojac
beql izadji         ! proslo se kroz ceo niz
ldrid [r1]x.0, r3   ! u r3 naredni element
```

```
cmp r0, r3
blss noviMax          ! nadjen je veci element
jmp loop              ! nazad na ispitivanje sledeceg elementa

noviMax:
mvrrl r0, r3          ! u r0 upisuje novi najveci element iz r3
jmp loop              ! nazad na ispitivanje sledeceg elementa



! izlaz iz funkcije
izadji:
               ! u obrnutom redosledu skidamo registre
pop r3
pop r2
pop r1
pop r4
rts


! prekidna rutina KP1
org x.2000
! neke registre koji se menjaju u prekidnoj rutini,
! a potrebni su i van nje, cuvati na steku

push r0               ! koristi se kao pomocni registar pa se cuva na steku
ldimm x.1, r0
cmp ra, r0            ! da li se radi ulaz ili izlaz
bneq izlaz            ! radi se o izlazu

! ulaz
cmp rb, r0            ! da li je primljena velicina niza
beql primiElement     ! skoci na prijem sledeceg elementa

! primanje velicine niza
ldmem x.f103, r4      ! broj elemenata niza u r4, sluzi kao brojac
mvrrl rf, r4          ! u rf se cuva velicina niza,
                      ! bice potrebna kada se poziva funkcija max
inc r4                ! brojac uvecan za 1
ldimm x.1, rb         ! rb na vrednost 1 jer je primljena velicina niza
jmp kraj

! prijem elemenata
primiElement:
dec r4                ! smanjuje se brojac
beql iskljuci         ! svi elementi su primljeni, iskljucuje se periferija
ldmem x.f103, r0      ! citanje jednog elementa
stri [r5], r0         ! smestanje elementa u memoriju
inc r5                ! povecavanje pokazivaca
```

```
jmp kraj              ! skok na povratak iz prekidne rutine

! izlaz
izlaz:
dec r4                ! smanjimo brojac
beql iskljuci         ! ako je poslat podatak
pop r0                ! na pocetku je registar r0 stavljen na stek
stmem x.f103, r0      ! salje se periferiji najveci element
push r0               ! vratimo r0 jer se na kraju skida sa steka
jmp kraj              ! skok na kraj

! iskljucivanje KP1
iskljuci:
stmem x.f100, r4      ! u r4 je nula pa se koristi za iskljucivanje KP1
ldimm x.1, rc         ! postavlja se semafor, prenos zavrsen

! kraj
kraj: pop r0
rti
```

Задатак 4

Написати програм који учитава низ дужине Ah са DMA1.1 контролера у меморију почев од адресе 3000h. DMA контролер ради у пакетском режиму рада. Након тога учитани низ послати DMA1.2 контролеру у појединачном режиму рада.

Прекидне рутине за DMA1.1 и DMA1.2 се налазе на адресама 2000h и 2100h, респективно. IV табела је подешена, као и бројеви улаза унутар *entry* регистара ова два контролера. Главни програм треба сместити почев од адресе 1000h.

Решење:

```
! inicijalizacija simulacije
onkp false, x.1, x.1
ondma x.1, x.1
ondma x.1, x.1
ondma x.1, x.1
ondma x.1, x.1
dmareg 1, r0, x.a
dmareg 1, r1, x.b
dmareg 1, r2, x.c
dmareg 1, r3, x.d
dmareg 1, r4, x.e
dmareg 1, r5, x.f
dmareg 1, r6, x.10
dmareg 1, r7, x.11
dmareg 1, r8, x.12
dmareg 1, r9, x.13
reg pc, x.1000
reg ivtp, x.0
dc x.2000, x.1
dc x.2100, x.2
dmareg 1, interrupt, x.1
dmareg 2, interrupt, x.2
! kraj inicijalizacije

! resenje
! glavni program
org x.1000
! inicijalizacija DMA1.1
ldimm x.a, r0            ! koliko podataka treba preneti u r0
stmem x.f004, r0         ! upis u count registar DMA1.1
ldimm x.3000, r0         ! adresa od koje se smestaju podaci u memoriju
stmem x.f006, r0         ! upis u destinacioni adresni registar AR2 DMA1.1
clr r0                   ! pocetna vrednost semafora nula
stmem x.4000, r0         ! odabrana slobodna lokacija x.4000
                         ! da cuva vrednost semafora
                         ! moze se koristiti i neki registar
                         ! da se cuva vrednost semafora
ldimm x.8f, r0           ! vrednost za startovanje DMA1.1 kontrolera
```

```
stmem x.f000, r0          ! upis vrednosti u kontrolni registar DMA1.1

! ceka se da DMA1.1 zavrsi slanje
ldimm x.1, r1             ! jedinica za testiranje semafora
wait1: ldmem x.4000, r0   ! cita se vrednost semafora
cmp r1, r0                ! da li je semafor postao jedan
bneq wait1                ! ako nije nazad na wait1

! inicijalizacija DMA1.2
ldimm x.a, r0             ! koliko podataka treba preneti u r0
stmem x.f044, r0          ! upis u count registar DMA1.2
ldimm x.3000, r0          ! adresa od koje DMA1.2 cita iz memorije podatke
stmem x.f045, r0          ! upis u izvorisni adresni registar AR1 DMA1.2
clr r0                    ! pocetna vrednost semafora nula
stmem x.4000, r0          ! odabrana slobodna lokacija x.4000
                          ! da cuva vrednost semafora,
                          ! nema veze sto je ista kao za DMA1.1
                          ! posto je DMA1.1 sada iskljucen
                          ! moze se koristiti i neki registar
                          ! da se cuva vrednost semafora

ldimm x.0e, r0            ! vrednost za startovanje DMA1.2 kontrolera
stmem x.f040, r0          ! upis vrednosti u kontrolni registar DMA1.2

! ceka se da DMA1.2 zavrsi slanje
ldimm x.1, r1             ! jedinica za testiranje semafora
wait2: ldmem x.4000, r0   ! cita se vrednost semafora
cmp r1, r0                ! da li je semafor postao jedan
bneq wait2                ! ako nije nazad na wait2

halt                      ! kraj programa

! prekidna rutina DMA1.1
org x.2000
push r0                   ! koristi se r0 pa se njegova vrednost cuva na steku
ldimm x.1, r0             ! jedinica za semafor
stmem x.4000, r0          ! postavljanje semafora
clr r0                    ! nula za kontrolni registar DMA za iskljucivanje
stmem x.f000, r0          ! upis u kontrolni registar
pop r0                    ! vracanje stare vrednosti r0
rti                       ! povratak iz prekidne rutine

! prekidna rutina DMA1.2
org x.2100
push r0                   ! koristi se r0 pa se njegova vrednost cuva na steku
ldimm x.1, r0             ! jedinica za semafor
stmem x.4000, r0          ! postavljanje semafora
clr r0                    ! nula za kontrolni registar DMA za iskljucivanje
stmem x.f040, r0          ! upis u kontrolni registar
```

```
pop r0                  ! vracanje stare vrednosti r0
rti                     ! povratak iz prekidne rutine
```

Задатак 5

Дата је структура **struct** `InputRequest { int* address; int size; InputRequest* next; }` која представља један захтев за улазном операцијом коришћењем DMA контролера у пакетском режиму рада. Поље `address` садржи адресу почев од које треба сместити учитане податке, поље `size` садржи колико података треба учитати. Овакве структуре су уланчане у једноструку листу (поље `next` садржи адресу наредног захтева), а адреса прве структуре у листи је 2000h. Последња структура у листи у свом пољу `next` има вредност 0. Сва поља структуре су ширине 16 бита и у листи постоји бар један захтев. Написати програм који обрађује један по један захтев све док се не обради последњи у листи.

Прекидна рутина за DMA контролер налази се на адреси 3000h. IV табела је подешена, као и број улаза унутар *entry* регистра DMA контролера. Главни програм треба сместити почев од адресе 1000h.

Решење:

```
! inicijalizacija simulacije
dc x.4000, x.2000
dc x.2, x.2001
dc x.2100, x.2002
dc x.5000, x.2100
dc x.3, x.2101
dc x.2200, x.2102
dc x.6000, x.2200
dc x.4, x.2201
dc x.0, x.2202
ondma x.1, x.1
dmareg 1, r0, x.a
dmareg 1, r1, x.b
dmareg 1, r2, x.c
dmareg 1, r3, x.d
dmareg 1, interrupt, x.1
reg pc, x.1000
reg ivtp, x.0
dc x.3000, x.1
! kraj inicijalizacije

! resenje
! glavni program
org x.1000
! inicijalizacija DMA1.1
ldimm x.2000, r0              ! adresa prve strukture u r0
loop: ldrid [r0]x.0, r1       ! u r1 adresa od koje treba smestati podatke
                              ! u memoriju, polje address se nalazi na
                              ! adresi r0+0h

stmem x.f006, r1              ! uspis u destinacioni adresni registar
                              ! AR2 DMA
ldrid [r0]x.1, r1             ! sledece polje strukture (size)
                              ! se nalazi na adresi r0+1h
                              ! koja se dobija sabiranje r0 i pomeraja 1h
```

```
                                ! u r1 se ucitava broj koliko
                                ! treba podataka preneti
stmem x.f004, r1                ! upis u count registar DMA
clr r2                          ! pocetna vrednost semafora nula,
                                ! koristi se r2 registar kao semafor
ldimm x.8f, r1                  ! vrednost za startovanje DMA kontrolera
stmem x.f000, r1                ! upis vrednosti u kontrolni registar DMA

! ceka se da DMA zavrsi obradu jednog zahteva
ldimm x.1, r1                   ! jedinica za testiranje semafora
wait: cmp r1, r2                ! da li je semafor postao jedan,
                                ! da li je u r2 upisana jedinica
bneq wait                       ! ako nije nazad na wait

! prelazi se na sledeci zahtev
ldrid [r0]x.2, r0               ! u r0 se smesta vrednost next polja,
                                ! adresa polja next se dobija kao r0+2h
                                ! jer se u r0 nalazi adresa
                                ! upravo obradjenog zahteva

clr r1                          ! u r1 nula
cmp r1, r0                      ! da li je r0 nula, to znaci
                                ! da je polje next jednako nuli
                                ! i da nema vise zahteva
bneq loop                       ! ako nije nula prelazi se na obradu
                                ! sledeceg zahteva


halt                            ! kraj programa

! prekidna rutina DMA
org x.3000
push r0                  ! koristi se r0 pa se njegova vrednost cuva na steku
ldimm x.1, r2            ! u r2 ide jedinica, postavlja se semafor
clr r0                   ! nula za kontrolni registar DMA za iskljucivanje
stmem x.f000, r0         ! upis u kontrolni registar
pop r0                   ! vracanje stare vrednosti r0
rti                      ! povratak iz prekidne rutine
```