

Домаћи задатак из Архитектуре рачунара за школску 2019/2020.

Задатак се састоји из три дела А, Б и В. Сваки део има четири варијанте (А0-А3, Б0-Б3, В0-В3). Студент на основу свог броја индекса добија комбинацију делова који чине његов домаћи задатак (од сваког дела добија по једну варијанту на основу броја индекса). Студент са бројем индекса $b_4b_3b_2b_1/g_4g_3g_2g_1$ добија делове на следећи начин:

- Део А као $b_1 \bmod 4$
- Део Б као $b_2 \bmod 4$
- Део В као $b_3 \bmod 4$

Пример: студент 0354/2021 ради домаћи задатак А0 Б1 В3.

Решење је потребно написати унутар фајла *VEZBA.ASM* који садржи неопходна подешавања симулације и исти је за све могуће комбинације домаћег задатка.

Део А

А0. Главни програм сместити од адресе 4000h. Иницијализовати IV табелу и IVTP регистар према следећој табели (колона Уређај садржи ознаку уређаја коме припада улаз):

Број улаза	Садржај	Уређај
0	0500h	KP1.1
1	1000h	KP1.2
2	1500h	KP2.1
3	2000h	DMA1.1
4	2500h	DMA1.2
5	3000h	DMA1.4
IVTP = 0000h		

Потребно је учитати низове А и В и сместити их у меморију почевши од адреса 5000h и 6000h, респективно. Оба низа имају по 8h елемената. Низ А учитати са KP1.1 испитивањем бита спремности, а низ В учитати са KP2.1 коришћењем механизма прекида. Учитавање обавити упоредо са обе периферије.

А1. Главни програм сместити од адресе 4000h. Иницијализовати IV табелу и IVTP регистар према следећој табели (колона Уређај садржи ознаку уређаја коме припада улаз):

Број улаза	Садржај	Уређај
5	0500h	KP1.1
4	1000h	KP1.2
2	1500h	KP2.1
3	2000h	DMA1.1
1	2500h	DMA1.2
0	3000h	DMA1.4
IVTP = 0100h		

Потребно је учитати низове А и В и сместити их у меморију почевши од адреса 5000h и 6000h, респективно. Оба низа имају по 8h елемената. Низ А учитати са KP1.1 коришћењем механизма прекида, а низ В учитати са KP2.1 испитивањем бита спремности. Учитавање обавити упоредо са обе периферије.

A2. Главни програм сместити од адресе 4000h. Иницијализовати IV табелу и IVTP регистар према следећој табели (колона Уређај садржи ознаку уређаја коме припада улаз):

Број улаза	Садржај	Уређај
5	0500h	KP1.1
4	1000h	KP1.2
3	1500h	KP2.1
2	2000h	DMA1.1
0	2500h	DMA1.2
1	3000h	DMA1.4
IVTP = 0200h		

Потребно је учитати низове А и В и сместити их у меморију почевши од адреса 5000h и 6000h, респективно. Оба низа имају по 8h елемената. Низ А учитати са KP1.1 испитивањем бита спремности, а низ В учитати са KP2.1 испитивањем бита спремности. Учитавање обавити упоредо са обе периферије.

A3. Главни програм сместити од адресе 4000h. Иницијализовати IV табелу и IVTP регистар према следећој табели (колона Уређај садржи ознаку уређаја коме припада улаз):

Број улаза	Садржај	Уређај
3	0500h	KP1.1
4	1000h	KP1.2
1	1500h	KP2.1
0	2000h	DMA1.1
2	2500h	DMA1.2
5	3000h	DMA1.4
IVTP = 0300h		

Потребно је учитати низове А и В и сместити их у меморију почевши од адреса 5000h и 6000h, респективно. Оба низа имају по 8h елемената. Низ А учитати са KP1.1 коришћењем механизма прекида, а низ В учитати са KP2.1 коришћењем механизма прекида. Учитавање обавити упоредо са обе периферије.

Део Б

B0. Написати потпрограма `int sumAll(int* arr1, int* arr2, int n)` која рачуна збир елемената два низа и резултат враћа унутар регистра R0. Параметри `arr1` и `arr2` представљају показиваче на низове чије елементе треба сабрати, а параметар `n` представља колико елемената из прослеђених низова треба сабрати. Након пријема низова А и В потребно је позвати функцију `sumAll` тако да се саберу сви елементи низова А и В. Након позива функције, резултат из R0 сачувати и на меморијску локацију 9999h. Потребно је још у меморију почев од локације 6100h прекопирати низ В коришћењем DMA1.4 уређаја у пакетском режиму рада.

B1. Написати потпрограма `void xorArr(int* arr1, int* arr2, int n)` која врши битско ексклузивно или на следећи начин `arr1[i] = arr1[i] xor arr2[i]`. Параметри `arr1` и `arr2` представљају показиваче на низове, а параметар `n` представља број елемената низова. Након пријема низова А и В потребно је позвати функцију `xorArr` тако да се израчуна $A[i] = A[i] \text{ xor } B[i]$. Након позива функције, нулти елемент низа А сместити на локацију 9999h. Потребно је још нулти елемент низа А прекопирати на 8h сукцесивних локација почевши од локације 5100h коришћењем DMA1.4 у пакетском режиму рада.

Б2. Написати потпрограм `void process(int* arr1, int n, Request* r)` која у низу дужине `n` на који показује `arr1` проналази најмањи или највећи елемент на основу вредности поља `operation` у структури на коју показује `r`. Вредност 0 поља `operation` значи да се тражи најмањи, а вредност 1 да се тражи највећи елемент. Пронађени елемент треба сместити у поље `element` структуре на коју показује `r`. Након пријема низова `A` и `B` потребно је позвати потпрограм `process` за оба низа, структуре за ова два позива се налазе у меморији на адресама `9996h` и `9998h`, респективно. Структура је дефинисана као `struct Request { int operation; int element; }`.

Б3. Написати потпрограм `void processElem(int* elem)` која треба да податак на који показује `elem` комплементира. Након пријема низова `A` и `B` за сваки елемент низа `A[i]` позвати потпрограм `processElem` ако је одговарајући елемент низа `B[i]` паран број. Након овога нулти елемент низа `A` сместити на локацију `FFFFh`. Потребно је још у меморију почев од локације `5100h` прекопирати новодобијени низ `A` коришћењем `DMA1.4` уређаја у пакетском режиму рада.

Део В

В0. Након завршеног дела Б задатка, низ `A` послати периферији `KP1.2` испитивањем бита спремности. Вредност са меморијске локације `9999h` послати уређају `DMA1.2` у пакетском режиму раду.

В1. Након завршеног дела Б задатка, низ `A` послати уређају `DMA1.1` у пакетском режиму рада. Вредност са меморијске локације `9999h` послати уређају `DMA1.2` у режиму рада циклус по циклус.

В2. Након завршеног дела Б задатка, низ `A` послати периферији `KP1.2` коришћењем механизма прекида. Вредност са меморијске локације `9999h` послати уређају `DMA1.2` у пакетском режиму раду.

В3. Након завршеног дела Б задатка, низ `A` уређају `DMA1.1` у режиму рада циклус по циклус. Вредност са меморијске локације `9999h` послати уређају `DMA1.2` у пакетском режиму раду.

Документација

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 оредишни 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/izlaz

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*.

Инструкције асемблера

Коментари у асемблеру *PSasm* пишу се након знака !.

Операнд регистра се обележава *rN*, где је *N* индекс регистра у хексадекадном запису. Индекс може имати вредности од 0 до F (15). У табелама које следе је означен са *reg*.

Операнд броја се обележава *x.N*, где је *N* број у хексадекадном запису. Може имати вредности од 0 до FFFF (65535). У табелама које следе је означен са *num*.

Директивом **org x.N** се назначава да све инструкције које следе након ове директиве смештају у меморији почев од адресе *x.N*.

Лабеле у коду се означавају стрингом након кога следи знак :.

У табели 1 су регистри **imr** и **ivtp** означени са **ireg**, а са **preg** су означени регистри **sp** и **psw**.

Табела 1. Инструкције преноса података

Формат инструкције	Опис
ldimm num, reg	У регистар reg уписује вредност num .
ldmem num, reg	У регистар reg уписује вредност из меморије са адресе num .

ldrid [reg1]num, reg2	У регистар reg2 уписује вредност из меморије са адресе reg1+num .
stri [reg1], reg2	Уписује вредност регистра reg2 у меморијску локацију на адреси која се налази у регистру reg1 .
stmem num, reg	Уписује вредност регистра reg у меморијску локацију на адреси num .
mvril reg, ireg	Уписује вредност регистра ireg у регистар reg .
mvrir reg, ireg	Уписује вредност регистра reg у регистар ireg .
mvrri reg1, reg2	Уписује вредност регистра reg2 у регистар reg1 .
mvrpr reg, preg	Уписује вредност регистра reg у регистар preg .
mvrpl reg, preg	Уписује вредност регистра preg у регистар reg .
push reg	Уписује вредност регистра reg на меморијску локацију на врху стека.
pop reg	Уписује вредност меморијске локације врха стека у регистар reg .
clr reg	Уписује вредност 0 у регистар reg .

Табела 2. Аритметичке инструкције

Формат инструкције	Опис
add reg1, reg2, reg3	Уписује збир вредности регистара reg2 и reg3 у регистар reg1 . У регистар psw се уписују одговарајући NZCV битови.
sub reg1, reg2, reg3	Уписује разлику вредности регистара reg2 и reg3 у регистар reg1 . У регистар psw се уписују одговарајући NZCV битови.
cmp reg1, reg2	Израчунава разлику вредности регистара reg1 и reg2 , и у регистар psw уписује одговарајуће NZCV битове.
inc reg	Инкрементира (повећава за 1) вредност регистра reg , и у регистар psw уписује одговарајуће NZCV битове.
dec reg	Декрементира (умањује за 1) вредност регистра reg , и у регистар psw уписује одговарајуће NZCV битове.

Табела 3. Логичке инструкције

Формат инструкције	Опис
and reg1, reg2, reg3	Уписује резултат операције битског „И” над регистрима reg2 и reg3 у регистар reg1 . У регистар psw се уписују одговарајући NZ битови, а V бит се брише.
or reg1, reg2, reg3	Уписује резултат операције битског „ИЛИ” над регистрима reg2 и reg3 у регистар reg1 . У регистар psw се уписују одговарајући NZ битови, а V бит се брише.
xor reg1, reg2, reg3	Уписује резултат операције битског „ексклузивног ИЛИ” над регистрима reg2 и reg3 у регистар reg1 . У регистар psw се уписују одговарајући NZ битови, а V бит се брише.
tst reg1, reg2	Израчунава вредност операције битског „И” над регистрима reg1 и reg2 , и у регистар psw уписује одговарајуће NZ битове, а V бит

	се брише.
not reg	Уписује комплементирану вредност регистра reg у тај регистар. У регистар psw се уписују одговарајући NZ битови, а V бит се брише.

Табела 4. Инструкције померања и ротирања

Формат инструкције	Опис
asr reg	Врши аритметичко померање удесно битова регистра reg . У регистар psw се уписују одговарајући NZ битови, C бит добија вредност претходног најнижег бита reg регистра, а V бит се брише.
lsr reg	Врши логичко померање удесно битова регистра reg . У регистар psw се уписују одговарајући NZ битови, C бит добија вредност претходног најнижег бита reg регистра, а V бит се брише.
ror reg	Врши ротирање удесно битова регистра reg . У регистар psw се уписују одговарајући NZ битови, C бит добија вредност претходног најнижег бита reg регистра, а V бит се брише.
rorc reg	Врши ротирање удесно битова регистра reg , с тиме да се бит C регистра psw посматра као бит на 16. позицији. У регистар psw се уписују одговарајући NZ битови, а V бит се брише.
sl reg	Врши померање улево битова регистра reg . У регистар psw се уписују одговарајући NZ битови, C бит добија вредност претходног највишег бита reg регистра, а V бит се брише.
rol reg	Врши ротирање улево битова регистра reg . У регистар psw се уписују одговарајући NZ битови, C бит добија вредност претходног највишег бита reg регистра, а V бит се брише.
rolc reg	Врши ротирање улево битова регистра reg , с тиме да се бит C регистра psw посматра као бит на 16. позицији. У регистар psw се уписују одговарајући NZ битови, а V бит се брише.

Табела 5. Инструкције скокова

Формат инструкције	Опис
beql labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на релацију „једнако” (Z = 1).
bneq labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на релацију „неједнако” (Z = 0).
bgrt labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на релацију „веће” ((N xor V) or Z = 0).
bgre labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на релацију „веће или једнако” (N xor V = 0).
blss labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на релацију „мање” ((N xor V) = 1).
bleq labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на релацију „мање или једнако” ((N xor V) or Z = 1).

bgrtu labela	Врши скок на задату лабела у случају да битови NZCV регистра psw указују на релацију „веће” у случају посматрања регистара као неозначених вредности (C or Z = 0).
bgreu labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на релацију „веће или једнако” у случају посматрања регистара као неозначених вредности (C = 0).
blssu labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на релацију „мање” у случају посматрања регистара као неозначених вредности (C = 1).
blequ labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на релацију „мање или једнако” у случају посматрања регистара као неозначених вредности (C or Z = 1).
bneq labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на негативну вредност (N = 1).
bnng labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на ненегативну вредност (N = 0).
bovf labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на то да је дошло до прекорачења (V = 1).
bnvf labela	Врши скок на задату лабелу у случају да битови NZCV регистра psw указују на то да није дошло до прекорачења (V = 0).
jmp labela	Врши безусловни скок на задату лабелу.
jmpind reg	Врши безусловни скок на задату адресу записану у регистру reg .
jsr labela	Врши скок у потпрограма на задатој лабели.
rts	Врши повратак из потпрограма.
int num	Num је у интервалу [0x00, 0xFF] хексадекадних вредности, врши скок у прекидну рутину чији је број улаза једнак num .
rti	Врши повратак из прекидне рутине.

Табела 6. Инструкције постављања индикатора у PSW регистар

Формат инструкције	Опис
intd	Поставља вредност 0 на биту I регистра psw .
inte	Поставља вредност 1 на биту I регистра psw .
trpd	Поставља вредност 0 на биту T регистра psw .
trpe	Поставља вредност 1 на биту T регистра psw .

Табела 7. Инструкција заустављања

Формат инструкције	Опис
halt	Зауставља процесор.

Напомене:

- За израду домаћег задатка користити асемблер *SPasm* и симулатор *SPECSeo* на начин описан у материјалима за лабораторијске вежбе;
- 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.