



ЕЛЕКТРО  
ТЕХНИЧКИ  
ФАКУЛТЕТ  
УНИВЕРЗИТЕТ  
У БЕОГРАДУ

Катедра за рачунарску технику и информатику

Рачунарски ВЛСИ системи (13Е114ВЛСИ)

Школска 2023/2024. година

**ПРОЈЕКАТ**

---

## САДРЖАЈ

---

Садржај .....	2
Домаћи задатак .....	3
<b>(a) Симулација</b> .....	3
<b>(b) Синтеза</b> .....	5
<b>(c) Верификација</b> .....	11
Модификације .....	11
<b>(d) Синтеза</b> .....	11
<b>(e) Верификација</b> .....	11
<b>(f) Ресурси процесора</b> .....	11

---

(a) [5 поена] Симулација

---

Написати комбинациони модул **alu** у оквиру засебне датотеке **alu.v** који има следеће портове:

- тробитни улазни сигнал **oc**
- четворобитни улазни податак **a**
- четворобитни улазни податак **b**
- четворобитни излазни податак **f**

Модул представља аритметичко логичку јединицу (*Arithmetic Logic Unit*). Сигналом **oc** (*Operation Code*) се задаје операција која се извршава над првим и другим операндом који се задају подацима **a** и **b**, респективно. Податак **f** представља резултат задате операције. Модул занемарује сваки пренос и позајмицу који се догоде приликом извршавања операције. Операције које модул подржава задате су у Табели 1.

Табела 1: Операције аритметичко логичке јединице

Код операције	Операција	Оператори	Тип операције	Опис операције
000	ADD	A, B	Аритметичка	Сабирање
001	SUB			Одузимање
010	MUL			Множење
011	DIV			Дељење
100	NOT	A	Логичка	Комплементирање
101	XOR	A, B		Искључиво сабирање
110	OR			Сабирање
111	AND			Множење

---

Написати секвенцијални модул **register** у оквиру засебне датотеке **register.v** који има следеће портове:

- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst\_n**
- једнобитни улазни сигнал **cl**
- једнобитни улазни сигнал **ld**
- четворобитни улазни податак **in**
- једнобитни улазни сигнал **inc**
- једнобитни улазни сигнал **dec**
- једнобитни улазни сигнал **sr**
- једнобитни улазни податак **ir**
- једнобитни улазни сигнал **sl**
- једнобитни улазни податак **il**
- четворобитни излазни податак **out**

Модул представља регистар (*Register*). Сигналима **cl**, **ld**, **inc**, **dec**, **sr** и **sl** се задаје одговарајућа операција, при чему су операције наведене у према приоритету у опадајућем поретку. Подаци **in**, **ir** и **il** садрже додатне информације које су неопходне приликом извршавања операција **ld**, **sr** и **sl**, респективно. Податак **out** представља садржај регистра. Операције које модул подржава задате су у Табели 2.

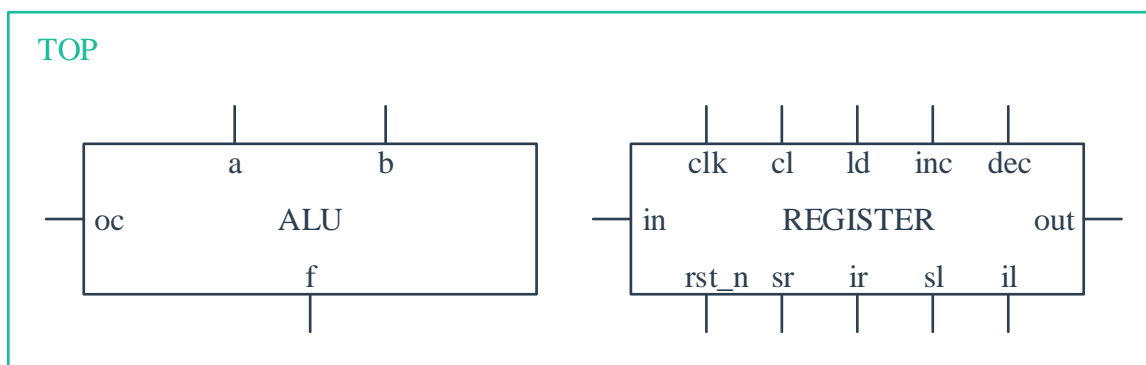
Табела 2: Операције регистра

Операција	Подаци операције	Опис операције
CLEAR	-	Брисање
LOAD	INPUT	Учитавање
INCREMENT	-	Инкрементирање
DECREMENT	-	Декрементирање
SHIFT RIGHT	INFORMATION RIGHT	Померање удесно
SHIFT LEFT	INFORMATION LEFT	Померање улево

Написати модул **top** (*Top Level Entity*) у оквиру засебне датотеке **top.v** који симулира понашање комбинационог модула **alu** и секвенцијалног модула **register**. Изглед модула дат је на Слици 1.

Симулација најпре побуђује комбинациони модул **alu** са свим могућим улазним вредностима, прати вредности свих улазних и излазних портова и у тренутку њихове промене врши испис симулационог тренутка и вредности свих улазних и излазних портова. Након побуђивања са свим могућим улазним вредностима за аритметичке операције, а затим и након побуђивања са свим могућим улазним вредностима за логичке операције, симулација се привремено зауставља. Симулација се наставља позивањем наредбе **run -all**.

Симулација затим побуђује секвенцијални модул **register** са хиљаду псеудослучајних улазних вредности, прати вредност искључиво излазних портова и у тренутку њихове промене врши испис симулационог тренутка и вредности свих улазних и излазних портова осим сигнала такта и асинхроног ресета. Након побуђивања са хиљаду псеудослучајних улазних вредности, симулација се завршава.



Слика 1: Модул за симулацију

---

**(b) [5 поена] Синтеза**

---

Написати модул **bcd** у оквиру засебне датотеке **bcd.v** који има следеће портове:

- шестобитни улазни податак **in**
- четворобитни излазни податак **ones**
- четворобитни излазни податак **tens**

Модул врши конверзију двоцифреног бинарног броја **in** у бинарно кодирани децимални број који чине цифра јединица **ones** и цифра десетица **tens** (*Binary Coded Decimal*).

---

Написати модул **ssd** у оквиру засебне датотеке **ssd.v** који има следеће портове:

- четворобитни улазни податак **in**
- седмобитни излазни податак **out**

Модул омогућава да се податак **in** прикаже на седмосегментном дисплеју (*Seven Segment Display*).

---

Написати модул **alu** у оквиру засебне датотеке **alu.v** који има следеће параметре и портове:

- параметар **DATA\_WIDTH** чија је подразумевана вредност **16**
- тробитни улазни сигнал **oc**
- улазни податак ширине **DATA\_WIDTH** бита **a**
- улазни податак ширине **DATA\_WIDTH** бита **b**
- излазни податак ширине **DATA\_WIDTH** бита **f**

Модул представља аритметичко логичку јединицу (*Arithmetic Logic Unit*) и детаљно је описан у ставци **(a)**.

Написати модул **register** у оквиру засебне датотеке **register.v** који има следеће параметре и портове:

- параметар **DATA\_WIDTH** чија је подразумевана вредност **16**
- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst\_n**
- једнобитни улазни сигнал **cl**
- једнобитни улазни сигнал **ld**
- улазни податак ширине **DATA\_WIDTH** бита **in**
- једнобитни улазни сигнал **inc**
- једнобитни улазни сигнал **dec**
- једнобитни улазни сигнал **sr**
- једнобитни улазни податак **ir**
- једнобитни улазни сигнал **sl**
- једнобитни улазни податак **il**
- излазни податак ширине **DATA\_WIDTH** бита **out**

Модул представља регистар (*Register*) и детаљно је описан у ставци (a).

---

Написати модул **clk\_div** у оквиру засебне датотеке **clk\_div.v** који има следеће параметре и портове:

- параметар **DIVISOR** чија је подразумевана вредност **50 000 000**
- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst\_n**
- једнобитни излазни сигнал **out**

Модул успорава сигнал такта (*Clock Divider*) онолико пута колико је задато параметром **DIVISOR**.

Дат је модул **memory** у оквиру засебне датотеке **memory.v** који има следеће параметре и портове:

- параметар **FILE\_NAME** чија је подразумевана вредност "**mem\_init.mif**"
- параметар **ADDR\_WIDTH** чија је подразумевана вредност **6**
- параметар **DATA\_WIDTH** чија је подразумевана вредност **16**
- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst\_n**
- једнобитни улазни сигнал **we**
- улазни податак ширине **ADDR\_WIDTH** бита **addr**
- улазни податак ширине **DATA\_WIDTH** бита **data**
- излазни податак ширине **DATA\_WIDTH** бита **out**

Модул представља меморију (*Memory*). У зависности од вредности сигнала **we** (*Write Enable*) са адресе задате податком **addr** из се меморије чита податак **out** (вредност 0) или се у меморију пише податак **data** (вредност 1).

Меморија се састоји од 64 меморијске речи. Ширина меморијске речи је 16 бита. Меморија је подељена у две логичке целине: фиксну зону података и слободну зону података. Фиксну зону података чини првих 8 меморијских локација и она обавља улогу регистра опште намене (*General Purpose Register*). У слободној зони података налази се програм (машинске инструкције), подаци и стек. Програм почиње од прве меморијске локације у слободној зони. Стек почиње од последње меморијске локације и расте према нижим меморијским локацијама.

---

Написати модул **cpu** у оквиру засебне датотеке **cpu.v** реализован као машину стања који има следеће параметре и портове:

- параметар **ADDR\_WIDTH** чија је подразумевана вредност **6**
- параметар **DATA\_WIDTH** чија је подразумевана вредност **16**
- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst\_n**
- улазни податак ширине **DATA\_WIDTH** бита **mem\_in**
- улазни податак ширине **DATA\_WIDTH** бита **in**
- једнобитни излазни сигнал **mem\_we**
- излазни податак ширине **ADDR\_WIDTH** бита **mem\_addr**
- излазни податак ширине **DATA\_WIDTH** бита **mem\_data**
- излазни податак ширине **DATA\_WIDTH** бита **out**
- излазни податак ширине **ADDR\_WIDTH** бита **pc**
- излазни податак ширине **ADDR\_WIDTH** бита **sp**

Модул представља процесор (*Central Processing Unit*) са *picoComputer* архитектуром.

Подаци су целобројне величине без знака дужине 16 бита.

У процесору постоји регистар програмског бројача PC (*Program Counter*) дужине 6 бита, указивач на врх стека SP (*Stack Pointer*) дужине 6 бита, прихватни регистар инструкције IR (*Instruction Register*) дужине 32 бита и регистар акумулатора A (*Accumulator*) дужине 16 бита. Почетна вредност програмског бројача PC је 8. Указивач на врх стека SP указује на прву слободну меморијску

локацију. Подаци **pc** и **sp** представљају вредност регистара PC и SP, респективно. Регистре реализовати коришћењем модула **register** описаног у ставци (f).

Процесор комуницира са меморијом преко портова **mem\_in**, **mem\_we**, **mem\_addr** и **mem\_data**.

Процесор је са троадресним форматом инструкција. Инструкције су дужине 1 или 2 бајта. Највиша 4 бита првог бајта инструкције представљају код операције. Преосталих 12 бита инструкције се користе за операнде, за сваки операнд по 4 бита. Највиши од 4 бита садржи информацију да ли је адресирање директно (вредност 0) или индиректно (вредност 1), док преостала 3 бита представљају адресу операнда. Изглед првог бајта инструкције приказан је у Табели 3.

Табела 3: Први бајт инструкције

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Код операције				Д/И	Адреса			Д/И	Адреса			Д/И	Адреса		

Други бајт инструкције је опциони и користи се за смештање константе или адресе. Изглед другог бајта инструкције приказан је у Табели 4.

Табела 4: Други бајт инструкције

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

Инструкција преноса података MOV (*Move*) умножава податак из једне меморијске локације у другу. Код операције је 0000<sub>2</sub>. Ако трећи операнд има вредност 0000<sub>2</sub>, тада се податак са адресе другог операнда умножава на адресу првог операнда. Ако трећи операнд има вредност 1000<sub>2</sub>, тада се константа задата другим бајтом инструкције умножава на адресу првог операнда, док се други операнд не користи.

Улазно-излазна инструкција IN (*Input*) учитава податак са стандардног улаза. Податак са стандардног улаза **in** се учитава на адресу првог операнда. Други и трећи операнд се не користе.

Улазно-излазна инструкција OUT (*Output*) исписује податак на стандардни излаз. Податак **out** се исписује на стандардни излаз са адресе првог операнда. Други и трећи операнд се не користе.

Аритметичке инструкције ADD (*Addition*), SUB (*Subtraction*), MUL (*Multiplication*) и DIV (*Division*) извршавају редом операције сабирања, одузимања, множења и дељења. Кодови операција су редом 0001<sub>2</sub>, 0010<sub>2</sub>, 0011<sub>2</sub> и 0100<sub>2</sub>. Операција се извршава између података са адреса другог и трећег операнда, а резултат се смешта на адресу првог операнда. Аритметичке инструкције реализовати коришћењем модула **alu** описаног у ставци (f). Аритметичка инструкција DIV не ради.

Контролна инструкција STOP (*Stop*) зауставља извршавање програма и може опционо да испише до 3 податка на стандардни излаз. Код операције је 1111<sub>2</sub>. Уколико било који операнд има вредност различиту од 0000<sub>2</sub>, тада се податак са адресе операнда исписује на стандардни излаз.



Написати модул **top** у оквиру засебне датотеке **top.v** реализован као машину стања који има следеће параметре и портове:

- параметар **DIVISOR** чија је подразумевана вредност **50 000 000**
- текстуални параметар **FILE\_NAME** чија је подразумевана вредност **mem\_init.mif**
- параметар **ADDR\_WIDTH** чија је подразумевана вредност **6**
- параметар **DATA\_WIDTH** чија је подразумевана вредност **16**
- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst\_n**
- тробитни улазни податак **btn**
- деветобитни улазни податак **sw**
- десетобитни излазни податак **led**
- двадесетосмобитни излазни податак **hex**

Модул представља главну шему (*Top Level Entity*). Меморија и процесор раде на успорен сигнал такта који је 1Hz и који се формира преко сигнала такта **clk** који је 50MHz. Уређај се асинхроно ресетује преко прекидача **sw[9]**. Стандардни улаз представљају прекидачи **sw[3:0]** са којих се читава податак. Стандардни излаз представљају диоде **led[4:0]**. Вредности регистра PC и SP се приказују преко седмосегментних дисплеја **hex[27:0]**. Изглед модула дат је на *Слици 2*.

---

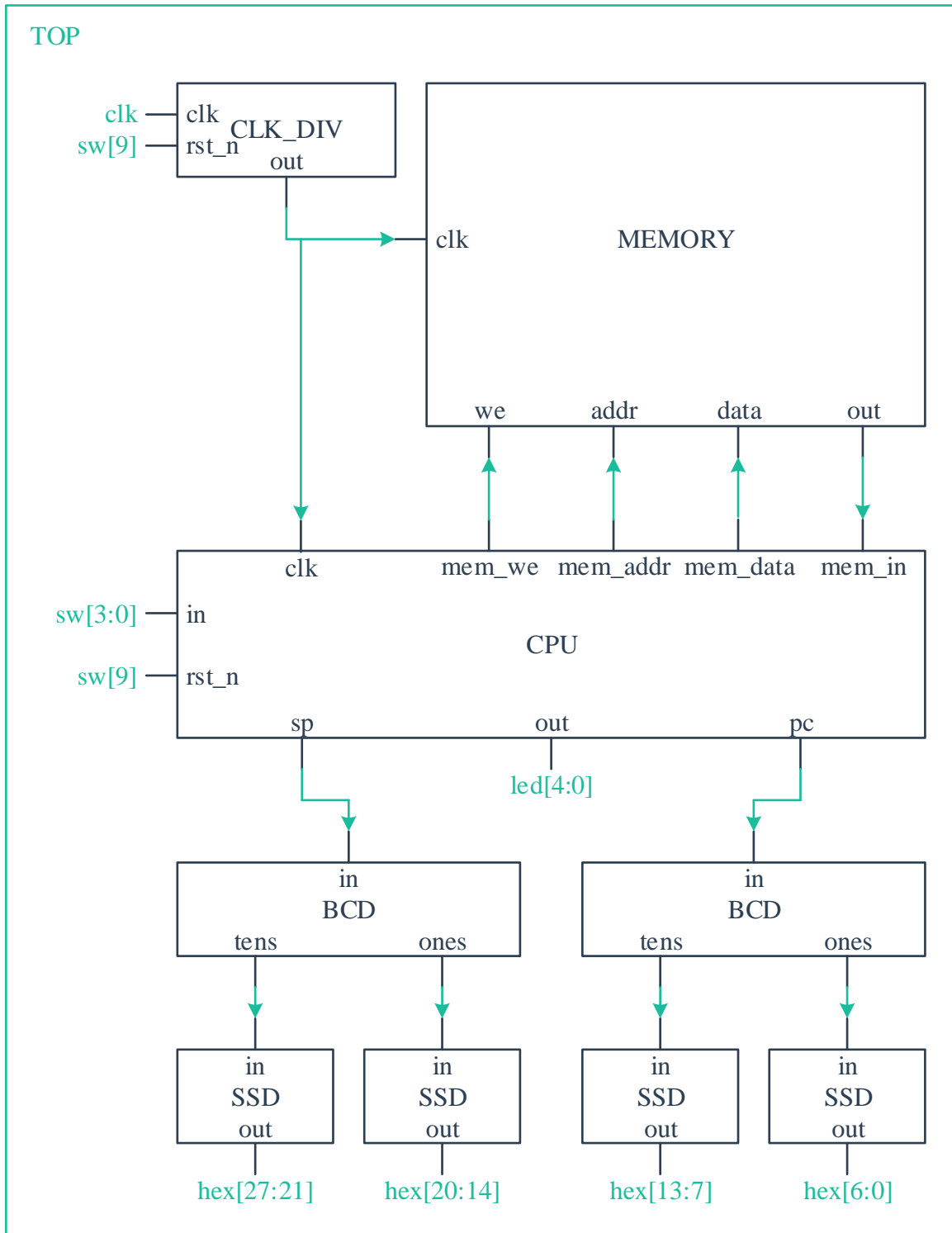
Дат је модул **DE0\_TOP** у оквиру засебне датотеке **DE0\_TOP.v** који има портове који омогућавају комуникацију са **DE0** плочицом која поседује **Cyclone III FPGA** чип. Модул инстанцира главну шему **top** (*Top Level Entity*).

---

Дат је модул **DE0\_CV\_TOP** у оквиру засебне датотеке **DE0\_CV\_TOP.v** који има портове који омогућавају комуникацију са **DE0** плочицом која поседује **Cyclone V FPGA** чип. Модул инстанцира главну шему **top** (*Top Level Entity*).

---

Дата је датотека за иницијализацију меморије **mem\_init.mif** (*Memory Initialization File*) на основу које се попуњавају меморијске локације тако да се смести програм написан на машинском језику за *picoComputer*.



Слика 2: Модул за синтезу

---

**(c) [5 поена] Верификација**

---

Написати модул **top** (*Top Level Entity*) у оквиру засебне датотеке **top.sv** који верификује понашање секвенцијалног модула **register** описаног у ставци (a) коришћењем **UVM** (*Universal Verification Methodology*) стандарда.

Утврдити успешност и квалитет верификације формирањем извештаја о покривености кода (*Code Coverage*).

---

**(d) [5 поена] Модификација: Синтеза**

---

**(e) [5 поена] Модификација: Верификација**

---

**(f) [5 поена] Модификација: Ресурси процесора**

---