ЕЛЕКТРО
ТЕХНИЧКИ
ФАКУЛТЕТ
УНИВЕРЗИТЕТ
У БЕОГРАДУ

Катедра за рачунарску технику и информатику

Рачунарски ВЛСИ системи (13С114ВЛСИ)

Школска 2023/2024. година

ПРОЈЕКАТ

САДРЖАЈ

(а) [5 поена] Комбинациони модули

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

(b) [5 поена] Секвенцијални модули

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

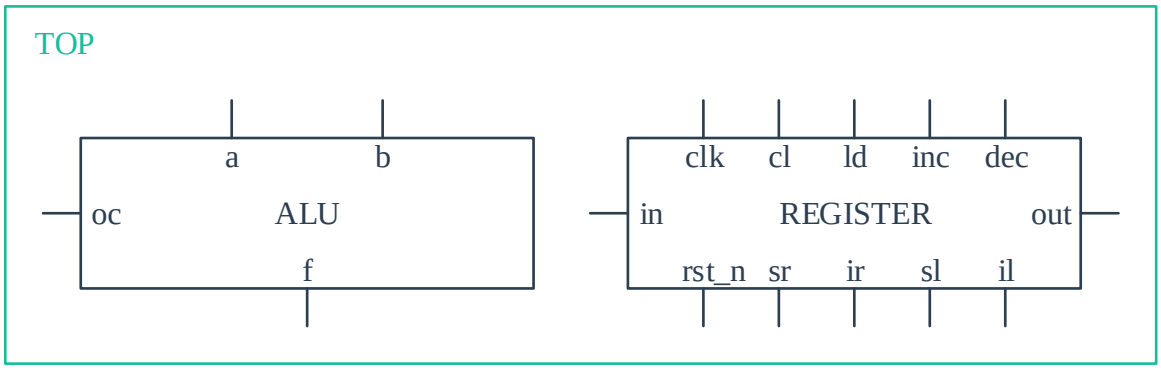
Операција	Подаци операције	Опис операције
CLEAR	-	Брисање
LOAD	INPUT	Учитавање
INCREMENT	-	Инкрементирање
DECREMENT	-	Декрементирање
SHIFT RIGHT	INFORMATION RIGHT	Померање удесно
SHIFT LEFT	INFORMATION LEFT	Померање улево

(c) [5 поена] Симулација

Написати модул **top** (*Top Level Entity*) у оквиру засебне датотеке **top.v** који симулира понашање комбинационог модула **alu** описаног у ставци **(a)** и секвенцијалног модула **register** описаног у ставци **(b)**. 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

(d) [5 поена] Модификација: Синтеза

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

- сигнал такта **clk**
- једнобитни улазни сигнал **we**
- шестобитни улазни податак **addr**
- осмобитни улазни податак **data**
- осмобитни излазни податак **out**

Модул представља меморију (*Memory*). У зависности од вредности сигнала **we** (*Write Enable*) са адресе задате податком **addr** из меморије се чита податак **out** (вредност 0) или се у меморију пише податак **data** (вредност 1). Читање се обавља у истом такту (комбинациона логика), док се упис обавља у наредном такту (секвенцијална логика).

Написати модул **dut** (*Design Under Test*) у оквиру засебне датотеке **dut.v** који има следеће портове:

- сигнал такта **clk**
- једнобитни улазни сигнал **we**
- шестобитни улазни податак **addr**
- шеснаестобитни улазни податак **data**
- шеснаестобитни излазни податак **out**

Модул представља меморију (*Memory*) веће ширине података. Модул написати искључиво инстанцирањем претходно написаног модула **memory**.

Написати модул **tb** (*Testbench*) у оквиру засебне датотеке **tb.v** који симулира понашање модула **dut** (*Design Under Test*).

Симулација најпре редом иницијализује све адресе меморије псеудослучајним вредностима. Након иницијализације симулација се привремено зауставља. Симулација се наставља позивањем наредбе **run -all**.

Симулација затим чита податке са стотину псеудослучајно одабраних адреса. Након читања симулација се завшрава. Модул врши испис симулационог тренутка и вредности свих улазних и излазних портова осим сигнала такта и асинхроног ресета на сваку узлазну ивицу сигнала такта. Испис се врши на самом крају симулационог тренутка.

(е) [5 поена] Помоћни модули

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

- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst_n**
- једнобитни улазни сигнал **in**
- једнобитни излазни сигнал **out**

Модул врши детекцију узлазне ивице улазног сигнала **in** (*Rising Edge Detector*).

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

- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst_n**
- једнобитни улазни податак **in**
- једнобитни излазни податак **out**

Модул исправља сметње на улазном сигналу **in** (*Debouncer*).

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*).

(f) [5 поена] Параметризовани модули

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 **(b)**.

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 регистара 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.

(g) [5 поена] Коначни аутомати (*Finite-State Machines*)

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, адресни регистар меморије MAR (*Memory Adress Register*) дужине 6 бита, прихватни регистар података меморије MDR (*Memory Data Register*) дужине 16 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_2. Ако трећи операнд има вредност 0000_2, тада се податак са адресе другог операнда умножава на адресу првог операнда. Ако трећи операнд има вредност 1000_2, 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 када сигнал **control** има вредност 1. Други и трећи операнд се не користе. Инструкција је блокирајућа.

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

Аритметичке инструкције ADD (*Addition*), SUB (*Substraction*), MUL (*Multiplication*) и DIV (*Division*) извршавају редом операције сабирања, одузимања, множења и дељења. Кодови операција су редом 0001_2, 0010_2, 0011_2 и 0100_2. 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_2. Уколико било који операнд има вредност различиту од 0000_2, тада се податак са адресе операнда исписује на стандардни излаз.

(h) [5 поена] Синтеза

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

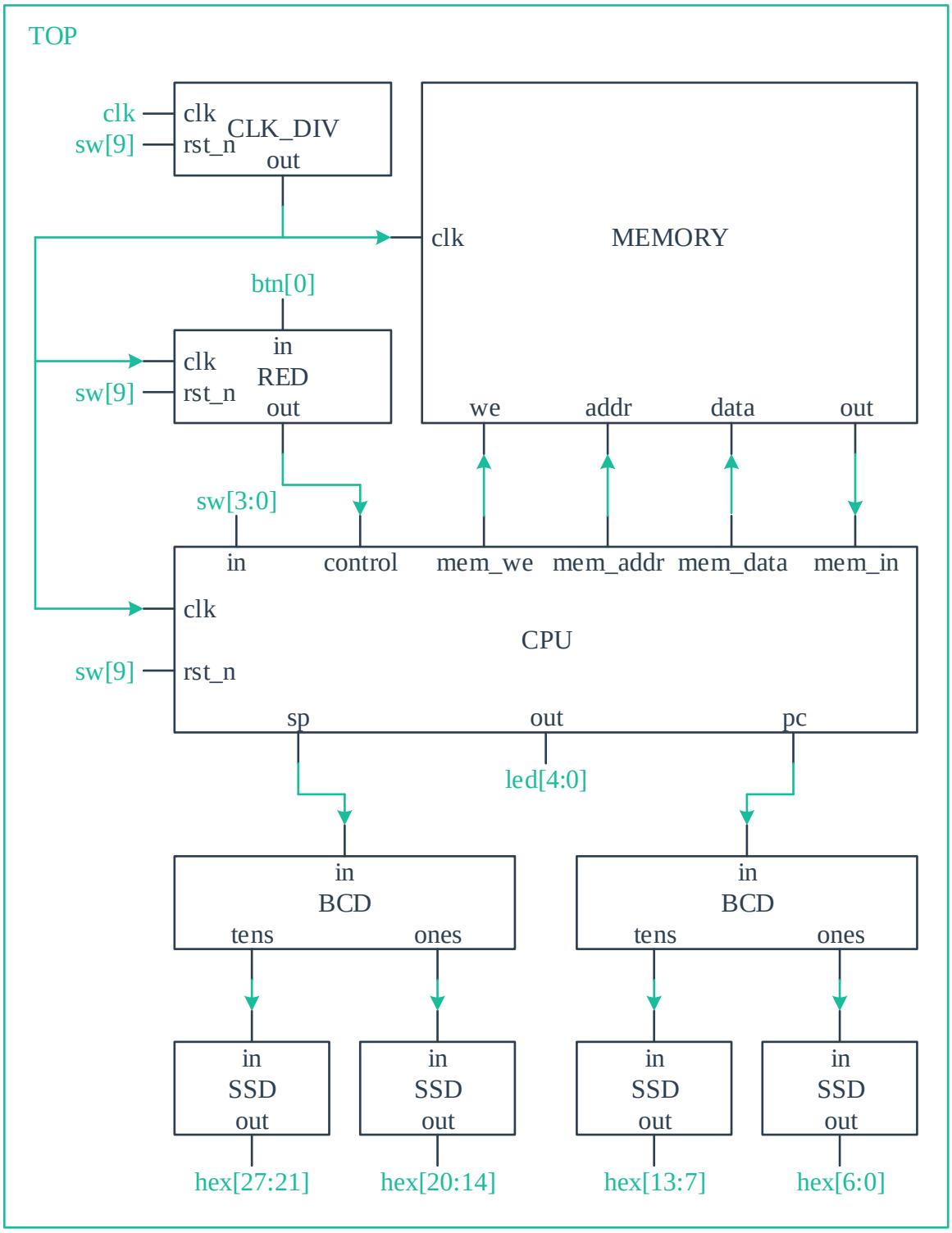
- параметар **DIVISOR** чија је подразумевана вредност **50 000 000**
- текстуални параметар **FILE_NAME** чија је подразумевана вредност **mem_init.mif**
- параметар **ADDR_WIDTH** чија је подразумевана вредност **6**
- параметар **DATA_WIDTH** чија је подразумевана вредност **16**
- сигнал такта **clk**
- 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 учитава податак. Стандардни излаз представљају диоде **led[4:0]**. Вредности регистара PC и SP се приказују преко седмосегментих 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

(i) [5 поена] Модификација: Синтеза

(j) [5 поена] Модификација: Синтеза

(k) [5 поена] Комуникација са тастатуром

Написати модул **ps2** у оквиру засебне датотеке **ps2.v** реализован као машину стања који има следеће портове:

- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst_n**
- једнобитни улазни сигнал **ps2_clk**
- једнобитни улазни податак **ps2_data**
- шеснаестобитни излазни податак **code**

Модул представља контролер за комуникацију са тастатуром коришћењем PS/2 протокола (*IBM Personal System/2*). Сигнал **ps2_clk** представља сигнал такта тастатуре, док податак **ps2_data** представља податак који се шаље. Податак **code** садржи информацију о последња два успешно прочитана бајта, при чему се последњи бајт чува на нижих осам бита, а претпоследњи на виших осам бита.

(l) [5 поена] Комуникација тастатура-процесор

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

- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst_n**
- шеснаестобитни улазни податак **code**
- једнобитни улазни сигнал **status**
- једнобитни излазни сигнал **control**
- четворобитни излазни податак **num**

Модул преводи притисак тастера на тастатури **code** у одговарајући број **num**. Прати се искључиво отпуштање тастера и искључиво тастери који означавају цифре. Модул преводи само када сигнал **status** има вредност један. Сигнал **control** добија вредност један када се преведе нови број и има ту вредност све док сигнал **status** не добије вредност нула.

(o) [5 поена] Комуникација са процесором

Ажурирати модул **cpu** написан 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 улазни сигнал **control**
- једнобитни излазни сигнал **status**
- 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 и детаљно је описан у ставци **(g)**.

Улазно-излазна инструкција IN (*Input*) учитава податак са стандардног улаза. Податак са стандардног улаза **in** се учитава на адресу првог операнда када сигнал **control** има вредност један, при чему сигнал **status** има вредност један када је процесор спреман да учита податак. Други и трећи операнд се не користе. Инструкција је блокирајућа.

(n) [5 поена] Комуникација процесор-монитор

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

- шестобитни улазни податак **num**
- двадечетворобитни излазни податак **code**

Модул преводи сваку цифру двоцифреног броја **num** у одговарајућу боју **code**. Виших дванаест бита податка **code** представља боју цифре десетица, док нижих дванаест бита представља боју цифре јединица. Свака цифра је представљена различитом бојом. Групе од четири бита у дванаестобитној боји, редом од највишег бита, означавају присуство црвене, зелене и плаве боје.

(m) [5 поена] Комуникација са монитором

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

- сигнал такта **clk**
- асинхрони ресет активан за вредност нула **rst_n**
- двадесетчетворобитни улазни податак **code**
- једнобитни излазни сигнал **hsync**
- једнобитни излазни сигнал **vsync**
- четворобитни излазни податак **red**
- четворобитни излазни податак **green**
- четворобитни излазни податак **blue**

Модул представља контролер за комуникацију са монитором коришћењем VGA протокола (*Video Graphics Array*). Податак **code** садржи информацију о две боје које треба да се прикажу на монитору. Виших дванаест бита означава боју која се приказује на левој половини монитора, а нижих дванаест бита боју која се приказује на десној страни монитора. Сигнали **hsync** и **vsync** редом представљају сигнале за хоризонталну и вертикалну синхронизацију са монитором. Подаци **red**, **green** и **blue** се редом користе за приказивање црвене, зелене и плаве боје на монитору.

(p) [5 поена] Синтеза

Ажурирати модул **top** написан 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**
- двобитни улазни податак **kbd**
- тробитни улазни податак **btn**
- деветобитни улазни податак **sw**
- четрнаестобитни излазни податак **mnt**
- десетобитни излазни податак **led**
- двадесетосмобитни излазни податак **hex**

Модул представља главну шему (*Top Level Entity*) и детаљно је описан у ставци **(h)**. Нижи бит података **kbd** представља сигнал такта тастатуре док виши бит представља податак. Прекидачи **sw[3:0]** се више не користе за учитавање података. Информација да ли је процесор спреман да прими податак приказује се преко диоде **led[5]**. Највиша два бита података **mnt** редом представљају сигнале за хоризонталну и вертикалну синхронизацију са монитором. Преосталих дванаест бита података **mnt** у групама од четири бита, почевши од највишег бита се користе редом за приказивање црвене, зелене и плаве боје на монитору.

(q) [5 поена] Верификација

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

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

(г) [5 поена] Модификација: Синтеза

(s) [5 поена] Модификација: Верификација

(t) [5 поена] Модификација: Ресурси процесора
