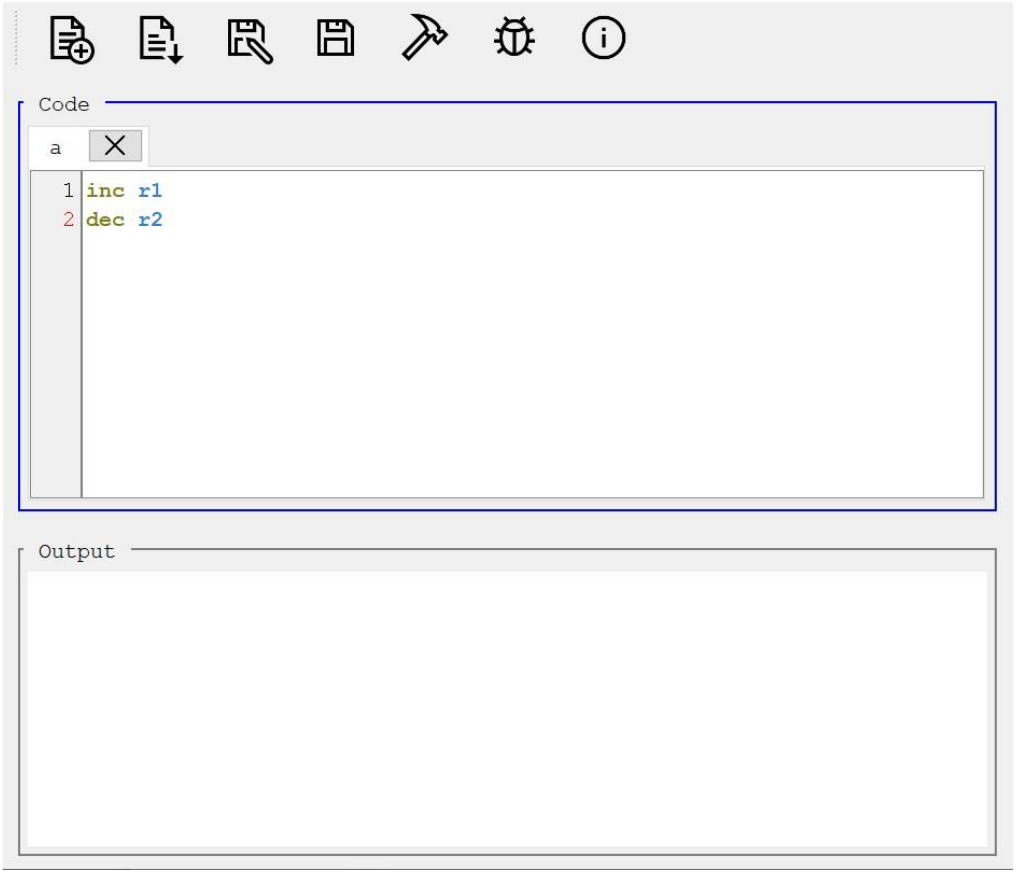
Упутство за лабораторијске вежбе из Архитектуре рачунара

За израду лабораторијских вежби користе се:

- Асемблер *PSams*
- Симулатор *SPECS*

Асемблер *PSams*

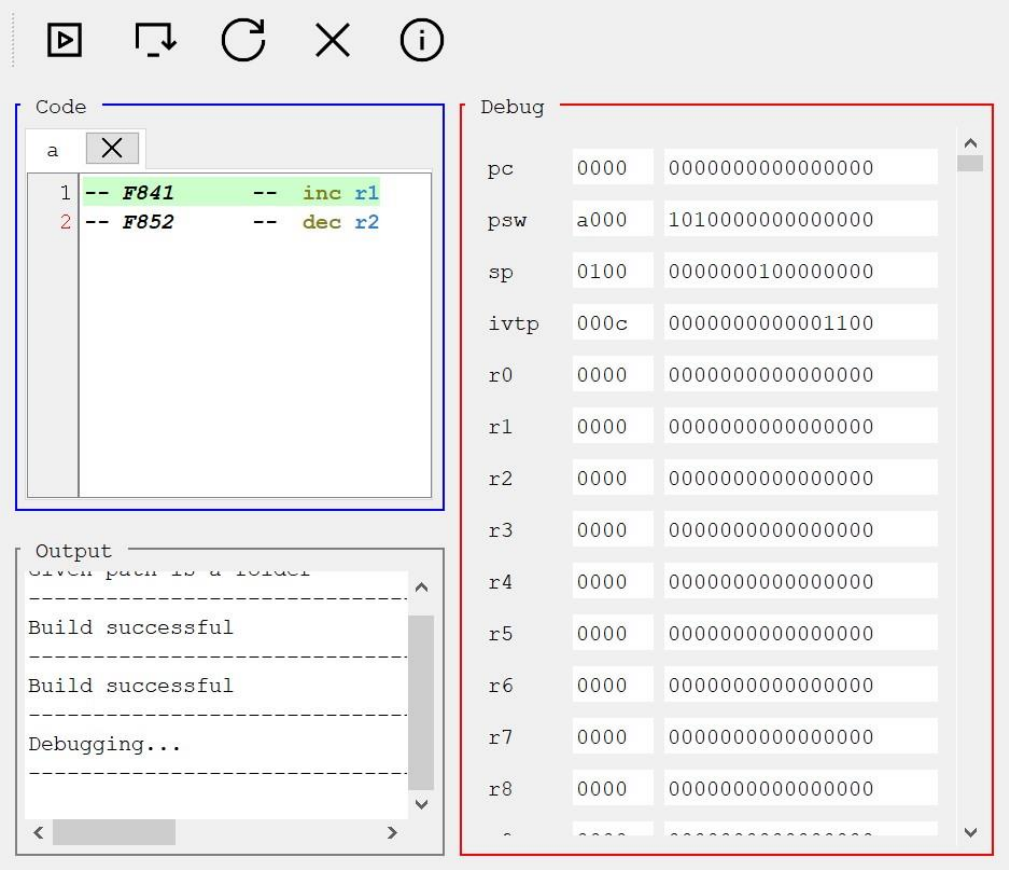
Корисник у програму може да пише код, да од њега ствара излазну датотеку и да емулира написан код ради тражења грешака. У корисничком интерфејсу се налазе следеће компоненте: мени, едитор, испис излаза и приказ стања архитектуре. Програм има два режима, режим писања кода и режим тражења грешака. Компоненте на интерфејсу и њихова функционалност се мењају у зависности од тренутног режима. Изглед прозора програма (у режиму писања кода) приказан је на слици 1.



Слика 1. Изглед асемблера *PSasm*

Притиском на дугме *Open file* треба отворити фајл под називом *VEZBA.ASM* који се налази у директоријуму симулатора *SPECS* у директоријуму *p3lab1*. Након отварања овог фајла могуће је мењати његов садржај додавањем програмског кода. Након измене фајла, треба сачувати фајл притиском на дугме *Save*. Након чувања фајла, могуће је откуцани програмски код асемблирати притиском дугмета *Assemble*.

Уколико се асемблирање заврши успешно, могуће је покренути дебагер (режим рада тражења грешака). У овом режиму се могу посматрати вредности регистара процесора и садржај оперативне меморије процесора након сваке извршене инструкције. У режим тражења грешака се прелази притиском на дугме *Debug* након чега треба унети редни број линије од које се започиње дебаговање програма. На слици 2. приказан је изглед прозора режима тражења грешке.



Слика 2. Режим тражења грешака

Процесор који је имплементиран у симулатору *SPECS* и асемблер *PSasm* подржавају различите типове инструкција и различите типове адресирања. У табелама 1.-7. дат је формат и опис подржаних инструкција.

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**.
`mvrrl 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.
jmprind reg	Врши безусловни скок на задату адресу записану у регистру **reg**.
jsr labela	Врши скок у потпрограм 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.

Симулатор *SPECS*

Након успешног асемблирања фајла *VEZBA.ASM*, који се налази у директоријуму *p3lab1* унутар директоријума *SPECS* симулатора, могуће је покренути симулатор *SPECS* и извршавати код написан у оквиру асемблера *PSasm*. Симулатор се покреће тако што се покрене датотека *SPECSeo.exe* која се налази у директоријуму *p3cfg*. Након покретања притиском на дугме *Simulation* започиње се симулација извршавања написаног програма.

Укратко како покренути симулацију почевши од писања кода:

1. Покренути *PSasm* асемблер
2. Отворити датотеку *VEZBA.ASM* која се налази у *p3lab1* директоријуму симулатора *SPECS*
3. Написати и асемблирати код
4. Покренути симулатор *SPECS* и започети симулацију притиском дугмета *Simulation*