UNIVERSITY OF CALIFORNIA, DAVIS

Department of Electrical and Computer Engineering

EEC 170 Introduction to Computer Architecture Fall 2019

Getting Started with RARS

(RISC-V Assembler, Runtime and Simulator)

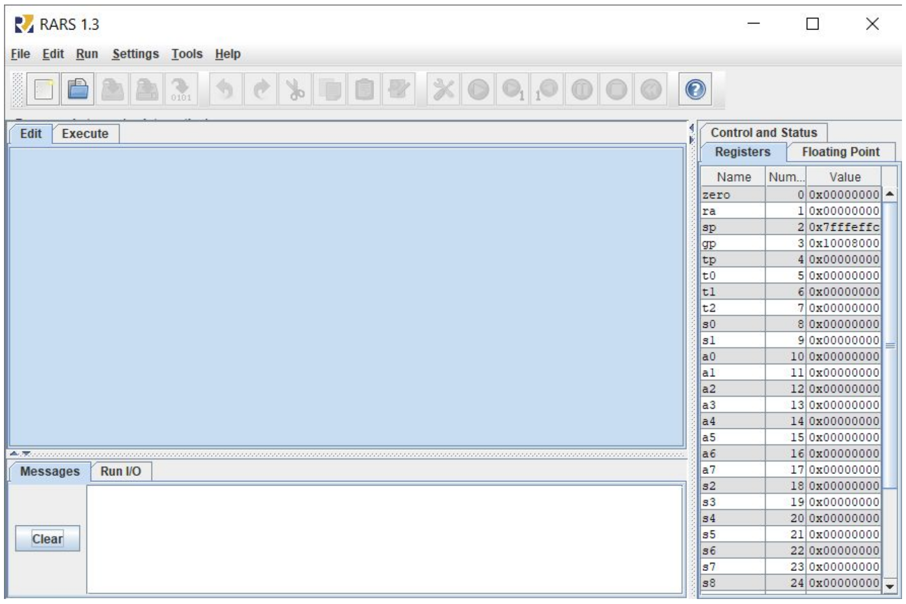
Setting up the Environment

1. Download the java executable for a recent release of RARS from https://github.com/TheThirdOne/rars/releases/download/v1.3.1/rars1_3_1.jar
2. RARS is distributed as an executable jar so, you will need at least Java 1.8 to run it. Install both of these packages Java Development kit and Java Runtime Environment. Update the environment variables in your computer by adding the path to binaries in these packages.
3. Our department workstations have these packages are already installed, go to */software/classtools/EEC170* directory and type the following command in the terminal to launch the IDE.

```
java -jar rars1_3_1.jar
```

Usage

1. Run "rars_1_3_1.jar" to open the IDE as shown picture below.
2. The IDE provides basic editing, assembling and execution capabilities. Refer to help section for detailed explanation of the features. **Help > RARS > IDE**
3. Optional: RARS can also be used through command line, for this, you need to download the source code from https://github.com/TheThirdOne/rars/releases. Then, run "build-jar.sh" file in the folder to build the repository. After a successful build, "rars.jar" will be created.
   Refer to https://github.com/TheThirdOne/rars/blob/master/help/Command.html for usage directives.



Name	Num...	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000

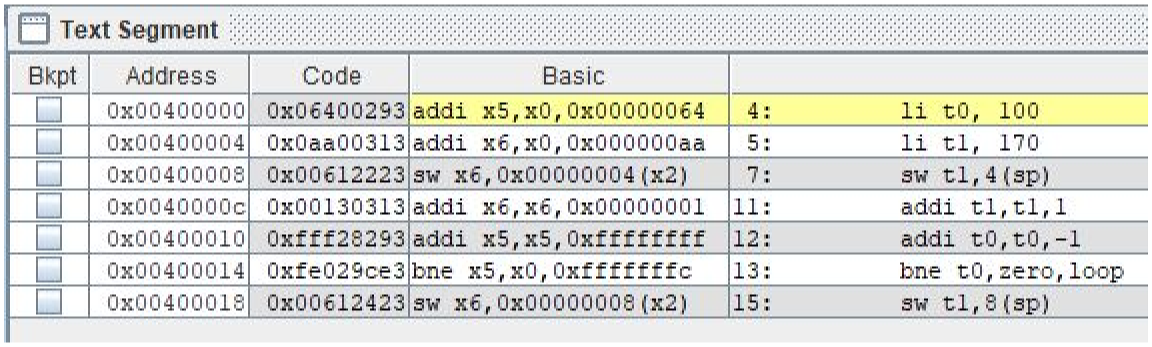
Running and Debugging

Let us work with a simple assembly code, which increments a register for a certain number of times and saves it in the stack.

1. Click on **File > New** to create a new file. Type the following code into editor and save it.

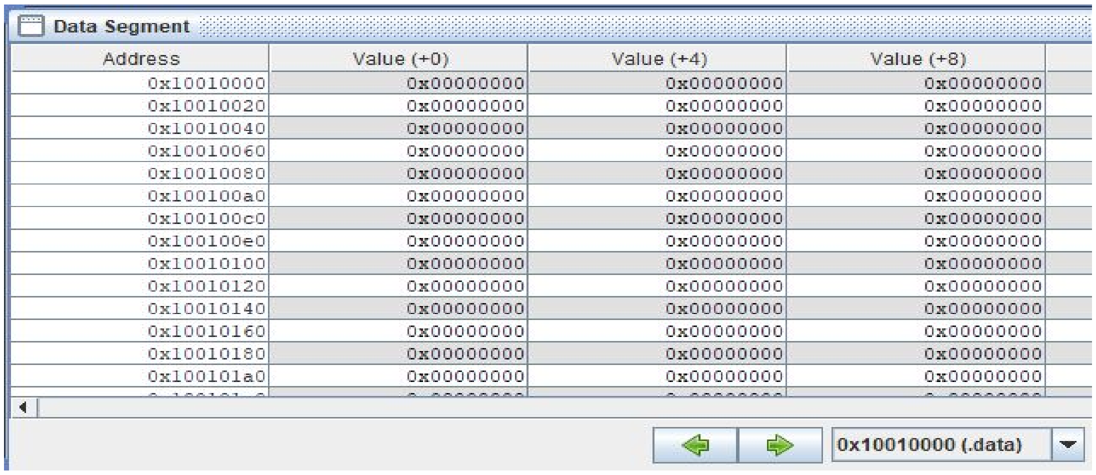
```
# Declare the listed label(s) as global to enable referencing from
other files
.globl  main
main: # The Program execution starts from here
        li t0, 100
        li t1, 170
# Initial Value of t1 is pushed into the stack
        sw t1,4(sp)

# Loop which increments t1 for a certain number of times
loop:
        addi t1,t1,1
        addi t0,t0,-1
        bne t0,zero,loop
# Final Value of t1 is pushed into the stack
        sw t1,8(sp)
```

2. To assemble the code, click on, **Run > Assemble.** Once the program successfully assembles, the registers are initialized and three windows in the Execute tab are filled with: *Text Segment*, *Data Segment*, and *Program Labels*.



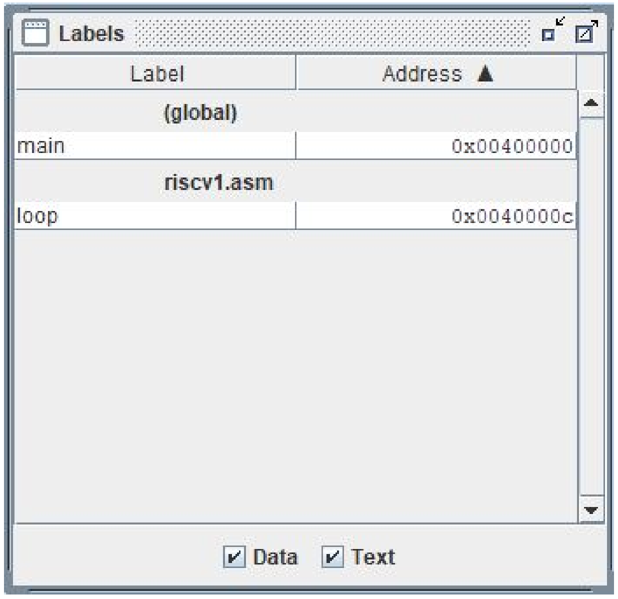
Text Segment

Bkpt	Address	Code	Basic		
☐	0x00400000	0x06400293	addi x5,x0,0x00000064	4:	li t0, 100
☐	0x00400004	0x0aa00313	addi x6,x0,0x000000aa	5:	li t1, 170
☐	0x00400008	0x00612223	sw x6,0x00000004(x2)	7:	sw t1,4(sp)
☐	0x0040000c	0x00130313	addi x6,x6,0x00000001	11:	addi t1,t1,1
☐	0x00400010	0xfff28293	addi x5,x5,0xffffffff	12:	addi t0,t0,-1
☐	0x00400014	0xfe029ce3	bne x5,x0,0xfffffffc	13:	bne t0,zero,loop
☐	0x00400018	0x00612423	sw x6,0x00000008(x2)	15:	sw t1,8(sp)



Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000



Labels

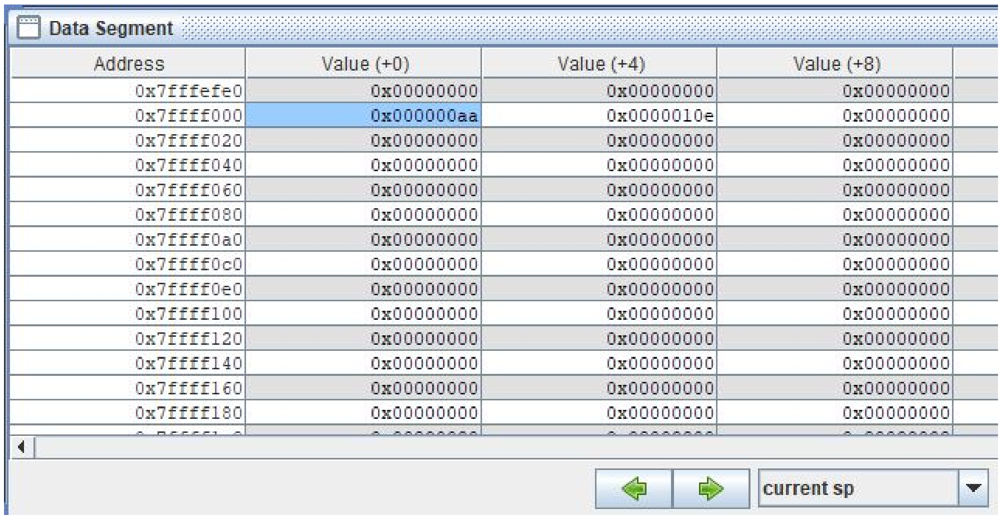
Label	Address ▲
(global)	
main	0x00400000
riscv1.asm	
loop	0x0040000c

3. You can run complete program by clicking on **Run > Go** or step by step **Run > Step.**
    - In **Step** mode, the next instruction to be simulated is highlighted and memory content displays are updated at each step.Select the **Go** option if you want to simulate continually. It can also be used to continue simulation from a paused (step, breakpoint, pause) state.
    - Breakpoints are easily set and reset using the checkboxes next to each instruction displayed in the Text Segment window.
    - When running in the **Go** mode, you can select the simulation speed using the Run Speed slider.
    - You can also pause or stop simulation at any time using the **Pause** or **Stop** features.
    - You have the ability to interactively step "backward" through program execution one instruction at a time to "undo" execution steps.
    - When program execution is paused or terminated, select **Reset** to reset all memory cells and registers to their initial.

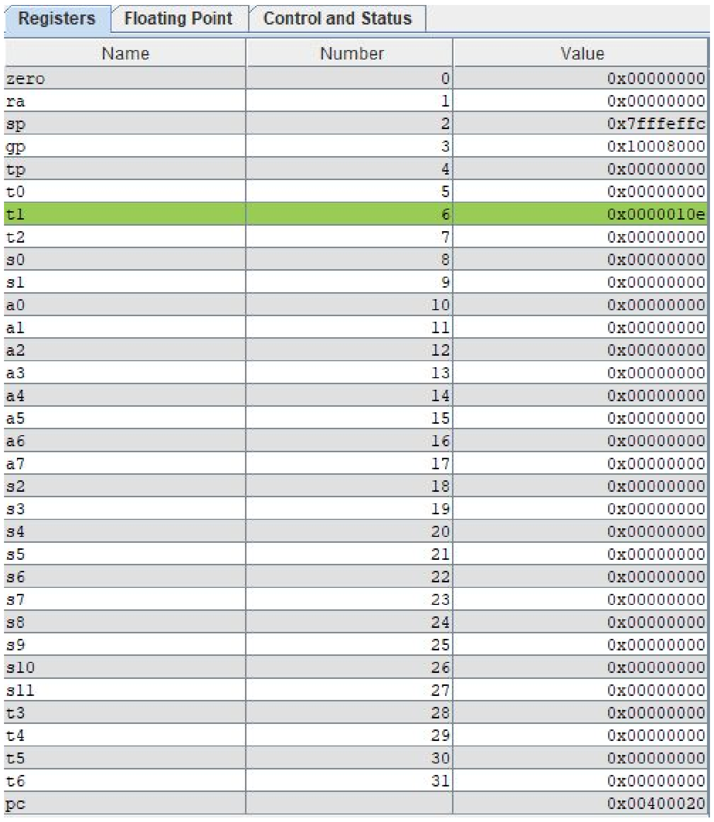
For more information refer to click on **Help > IDE > Debugging.**

4. Your values in registers and memory locations should match the following after execution.

    Address 0x7ffff000 corresponds to a location in stack, it contains the initial value of t1 i.e 170 or 0x0aa and the address 0x7ffff004(0x7ffff000 + Value(+4)) contains final value of t1 i.e 270 or 0x10e.



Address	Value (+0)	Value (+4)	Value (+8)
0x7fffefe0	0x00000000	0x00000000	0x00000000
0x7ffff000	0x000000aa	0x0000010e	0x00000000
0x7ffff020	0x00000000	0x00000000	0x00000000
0x7ffff040	0x00000000	0x00000000	0x00000000
0x7ffff060	0x00000000	0x00000000	0x00000000
0x7ffff080	0x00000000	0x00000000	0x00000000
0x7ffff0a0	0x00000000	0x00000000	0x00000000
0x7ffff0c0	0x00000000	0x00000000	0x00000000
0x7ffff0e0	0x00000000	0x00000000	0x00000000
0x7ffff100	0x00000000	0x00000000	0x00000000
0x7ffff120	0x00000000	0x00000000	0x00000000
0x7ffff140	0x00000000	0x00000000	0x00000000
0x7ffff160	0x00000000	0x00000000	0x00000000
0x7ffff180	0x00000000	0x00000000	0x00000000



Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x0000010e
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400020

5. You can use an editor of your choice to write a code, to import the code click on, **File > Open.**
6. To close a file on the RARS editor, select a file and click on **File > Close**.