



National Yang Ming Chiao Tung University
Computer Architecture & System Lab

Lab1: RISC-V Assembly Language Programming

CS10014 Computer Organization

Tsung Tai Yeh
Department of Computer Science
National Yang Ming Chiao University



Acknowledgements and Disclaimer

- Slides were developed in the reference with
 - RISC-V Programming
 - <https://riscv-programming.org/book/riscv-book.html>
 - CENG3420, CUHK
 - <https://www.cse.cuhk.edu.hk/~byu/CENG3420/2022Spring/slides/ab1-1.pdf>



Outline

- RISC-V Assembly Programming
 - Labels
 - Symbols
 - Directives
- RARS RISC-V Assembly Simulator



Generating native programs

- A native program is a program
 - Encoded using instructions that can be directly executed by the computer hardware

C Code

```
int main ()
{
    int r = func (10);
    return r+1;
}
```

RISC-V assembly code

```
.text
.align 2
main:
    addi sp, sp, -16
    li   a0, 10
    sw   ra, 12(sp)
    jal  func
    lw   ra, 12(sp)
    addi a0, a0, 1
    addi sp, sp, 16
    ret
```



Generating native programs

- **A compiler**

- translate a program from one language to another
- `riscv64-unknown-elf-gcc -mabi=ilp32 -march=rv32i -S main.c -o main.s`
- The RV32I assembly program will be stored on the main.s file

- **An Assembler**

- A assembler is a tool that translates a program in assembly language into a program in machine language
- The GNU assembler tool (`as`) is an assembler
- The assembler produces object files (`.o`) that are encoded in binary and contains code in machine language
- `riscv64-unknown-elf-as -mabi=ilp32 -march=rv32i main.s -o main.o`



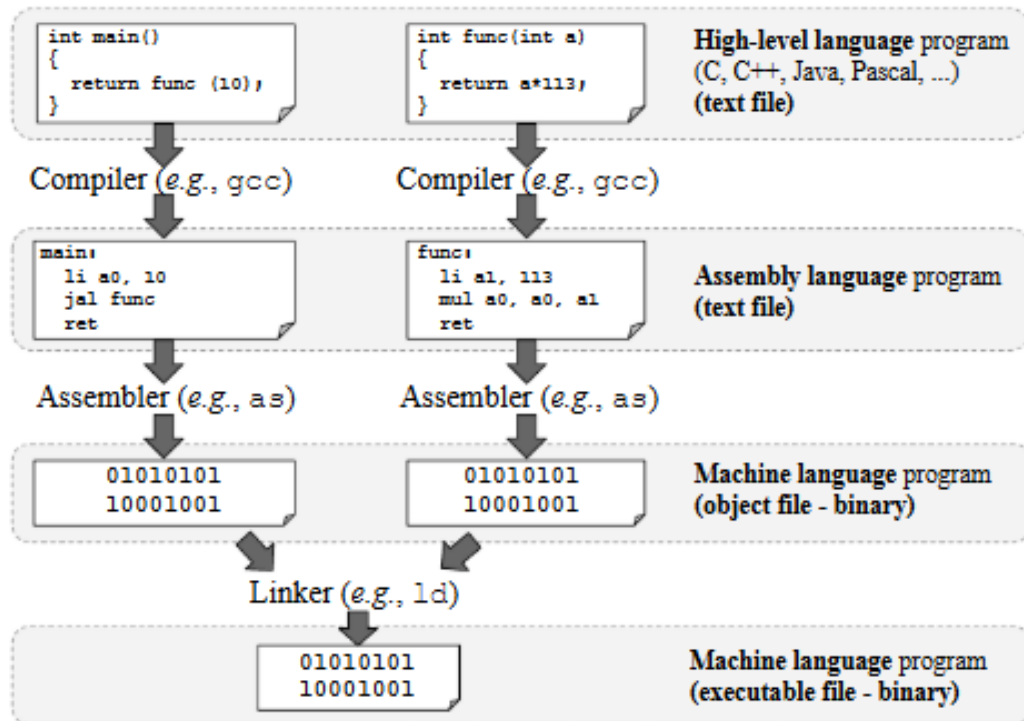
Generating native programs

- **A linker**

- A tool that 'links' together one or more object files
- Produces an executable file

`riscv64-unknown-elf-ld -m elf32lriscv
main.o mylib.o -o main.x`

`main.x` is an executable file





Labels

- A **label** in assembly language
 - As a marker that represent program location
 - 'x:' label identifies a program location that contains a variable, which is allocated and initialized by the directive `.word 10`
 - 'sum10:' label identifies the program location that contains the first instruction of the sum10 routine

RISC-V assembly code

```
x:  
    .word 10  
sum10:  
    lw    a0, x  
    addi a0, a0, 10  
    ret
```



Program Symbol

- Program **symbols**
 - “names” that are associated with numerical values
 - The “symbol table” is a data structure that maps each program system to its value
 - “nm” tool helps us to inspect the symbol table of a program

```
$ riscv64-unknown-elf-nm sum10.o  
00000004 t .L0  
00000004 t sum10  
00000000 t x
```



Program Symbol

- Using the **“.set”** directive
 - Explicitly define symbols
 - The follow example that uses **.set** directive to define a symbol named **answer** and assign value 42 to it
 - Two symbols: **answer**, **get_answer**

```
.set    answer, 42  
get_answer  
    li    a0, answer  
    ret
```



Global vs local Symbols

- Local symbols
 - Only visible on the same file
 - By default, the assembler registers labels as local symbols
- The `.global` directive
 - Instructs the assembler to register a label as a global symbol

```
.global  exit
exit:
    li    a0, 0
    li    a7, 93
    ecall
```



Program Entry Point

- The entry point is defined by an address
 - The address of the first instruction that must be executed
 - The linker sets the entry point field on the executable file and looks for a symbol named start
 - The linker sets the entry point to a default value (the address of the first instruction of the program) if the linker cannot find “start” symbol

```
.global  start
start:
    li    a0, 10
    li    a1, 20
    jal   exit
```



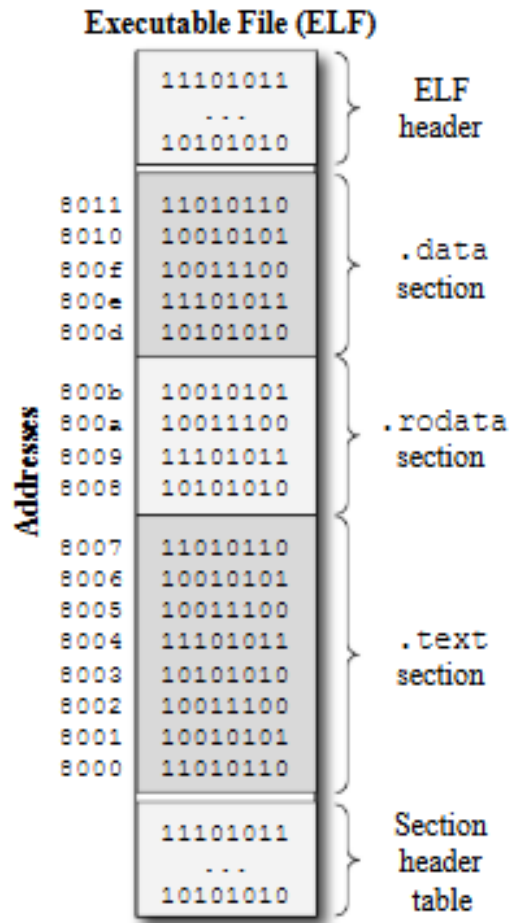
Program Section

- The assembly program is usually organized in ‘sections’
 - A section may contain data or instructions
 - The contents of each section are mapped to a set of consecutive main memory addresses
 - **.text**
 - Store the program instructions
 - **.data**
 - Store initialized global variables
 - **.bss**
 - Store uninitialized global variables
 - **.rodata**
 - Store constants



Executable File (ELF)

- When linking multiple object files
 - The linker groups information from sections with the same name and places them together into a single section on the exec. File
 - .text are mapped to addresses 8000 to 8007
 - .data are mapped to addresses 800d to 8011





Assembly Language

- Assembly program contains
 - **Comment**
 - **Labels**
 - Usually defined by a name ended with the suffix “:”
 - **Assembly instructions**
 - Converted by the assembler into machine instructions
 - E.g. `addi a0, a1, 1`
 - **Assembly directives**
 - Commands used to coordinate the assembling process
 - E.g. `.word 10`
 - Instruct the assembler to assemble a 32-bit value (10) into the program



Program Structure I

- Plain text file with data declarations
- Data declaration section is followed by program code section

Data Declarations

- Identified with assembler directive **.data**
- Declares variable names used in program
- Storage allocated in main memory (*e.g.*, RAM)
- `<name>:   .<datatype> <value>`



Program Structure II

Code

- placed in section of text identified with assembler directive **.text**
- contains program code (instructions)
- starting point for code e.g. execution given label **start:**

Comments

Anything following # on a line



Program Structure III

The structure of an assembly program looks like this:

Program outline

```
# Comment giving name of program and description
# Template.asm
# Bare-bones outline of RISC-V assembly language program

.globl _start

.data    # variable declarations follow this line
        # ...
.text    # instructions follow this line

_start: # indicates start of code
# ...

# End of program, leave a blank line afterwards is preferred
```



An Example RISC-V Assembly Program

```
1  .global _start
2
3  .data
4  welcome_msg: .asciz "Welcome!"
5
6  .text
7  _start:
8      # STDOUT = 1
9      addi a0, x0, 1
10     # Load the address of 'welcome_msg'
11     la a1, welcome_msg
12     # Length of the string
13     addi a2, x0, 8
14     # Linux write system call
15     addi a7, x0, 64
16     # Call linux service to output the string
17     ecall
```



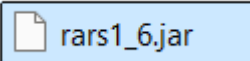
RISC-V ISA Simulator - RARS

- **RARS**
 - The RISC-V Assembler, Runtime, and Simulator for RISC-V assembly language programs
 - **RARS** supports RISC-V IMFDN ISA base (riscv32 & riscv64)
 - **RARS** supports debugging using breakpoints like ebreak
 - **RARS** supports side by side comparison from pseudo-instruction to machine with intermediate steps
 - You need to Java environment to run RARS
 - https://www.java.com/download/ie_manual.jsp
 - Download RARA
 - <https://github.com/TheThirdOne/rars/releases/tag/v1.6>



RISC-V ISA Simulator - RARS

- **How to run RARS?**

- Execute the command to start RARS: `java -jar <rars jar path>`
- In Windows OS
 - Ensure you have installed Java Environment
 - Right click  `rars1_6.jar`
 - Open with -> Java (TM) SE Platform



RISC-V ISA Simulator - RARS

The screenshot displays the RARS (RISC-V Assembler, Reverser, and Simulator) interface. The main window shows assembly code for a test program. The code includes comments, directives, and instructions for loading, shifting, and branching. The right panel shows the state of the registers, and the bottom panel shows the assembly process messages.

Assembly Code (test.asm):

```
88 # 12 "isa/rv64ui/rliw.S" 2
89
90
91 .text
92 .globl _start
93 _start: nop
94
95 -----
96 # Arithmetic tests
97 -----
98
99 test_2: li x1, 0xffffffff80000000
100 srlw x14, x1, 0
101 li x7, 0xffffffff80000000
102 li gp, 2
103 bne x14, x7, fail
104
105 test_3: li x1, 0xffffffff80000000
106 srlw x14, x1, 1
107 li x7, 0x0000000040000000
108 li gp, 3
109 bne x14, x7, fail
110
111 test_4: li x1, 0xffffffff80000000
```

Registers:

Name	Number	Value
zero	0	0x0000000000000000
ra	1	0x0000000000000000
sp	2	0x00000000ffffefff
gp	3	0x0000000010000000
tp	4	0x0000000000000000
t0	5	0x0000000000000000
t1	6	0x0000000000000000
t2	7	0x0000000000000000
t3	8	0x0000000000000000
t4	9	0x0000000000000000
t5	10	0x0000000000000000
t6	11	0x0000000000000000
t7	12	0x0000000000000000
t8	13	0x0000000000000000
t9	14	0x0000000000000000
s0	15	0x0000000000000000
s1	16	0x0000000000000000
s2	17	0x0000000000000000
s3	18	0x0000000000000000
s4	19	0x0000000000000000
s5	20	0x0000000000000000
s6	21	0x0000000000000000
s7	22	0x0000000000000000
s8	23	0x0000000000000000
s9	24	0x0000000000000000
s10	25	0x0000000000000000
s11	26	0x0000000000000000
s12	27	0x0000000000000000
s13	28	0x0000000000000000
s14	29	0x0000000000000000
s15	30	0x0000000000000000
s16	31	0x0000000000000000
pc		0x0000000000000000

Messages:

```
Assembly: assembling F:\Research\misc\TA\CEBS3420\tools\test.asm
Warning in F:\Research\misc\TA\CEBS3420\tools\test.asm line 312 column 2: RARS does not recognize the .global directive. Ignored.
Warning in F:\Research\misc\TA\CEBS3420\tools\test.asm line 318 column 2: RARS does not recognize the .global directive. Ignored.
Assembly: operation completed successfully.
```



RISC-V ISA Simulator - RARS

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Test Segment

Bitpt	Address	Code	Basic	Source
93	0x00000000	0x00000013	addi x0,x0,0	93: _start: nop
94	0x00000004	0x00000017	liu x1,0xffff0000	99: test_2: li x1, 0xffffffff00000000
95	0x00000008	0x0000001b	addiu x1,x1,x1,0	
96	0x0000000c	0x00000017	oriw x14,x1,0	100: oriw x14, x1, 0
97	0x00000010	0x0000001b	liu x7,0xffff0000	101: li x7, 0xffffffff00000000
98	0x00000014	0x0000001b	addiu x7,x7,x1,0	
99	0x00000018	0x0000001b	addi x2,x0,2	102: li gp, 2
100	0x0000001c	0x34771e53	liu x14,x1,x7,0x00000000	103: liu x14, x1, x7
101	0x00000020	0x0000001b	liu x1,0xffff0000	105: test_3: li x1, 0xffffffff00000000
102	0x00000024	0x0000001b	addiu x1,x1,x1,0	
103	0x00000028	0x00000017	oriw x14,x1,x1,1	106: oriw x14, x1, 1
104	0x0000002c	0x0000001b	liu x7,0x00040000	107: li x7, 0x0000000004000000
105	0x00000030	0x0000001b	addiu x7,x7,x1,0	
106	0x00000034	0x0000001b	addi x2,x0,3	108: li gp, 3
107	0x00000038	0x2271e53	liu x14,x1,x7,0x00000034	109: liu x14, x1, x7
108	0x0000003c	0x0000001b	liu x1,0xffff0000	111: test_4: li x1, 0xffffffff00000000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010004	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010008	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001000c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010010	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010014	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010018	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001001c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010024	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010028	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001002c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010030	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010034	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010038	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001003c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers Floating Point Control and Status

Name	Number	Value
zero	0	0x0000000000000000
ra	1	0x0000000000000000
sp	2	0x0000000000000000
gp	3	0x0000000000000000
tp	4	0x0000000000000000
t0	5	0x0000000000000000
t1	6	0x0000000000000000
t2	7	0x0000000000000000
t3	8	0x0000000000000000
t4	9	0x0000000000000000
t5	10	0x0000000000000000
t6	11	0x0000000000000000
t7	12	0x0000000000000000
t8	13	0x0000000000000000
t9	14	0x0000000000000000
s0	15	0x0000000000000000
s1	16	0x0000000000000000
s2	17	0x0000000000000000
s3	18	0x0000000000000000
s4	19	0x0000000000000000
s5	20	0x0000000000000000
s6	21	0x0000000000000000
s7	22	0x0000000000000000
s8	23	0x0000000000000000
s9	24	0x0000000000000000
s10	25	0x0000000000000000
s11	26	0x0000000000000000
s12	27	0x0000000000000000
s13	28	0x0000000000000000
s14	29	0x0000000000000000
s15	30	0x0000000000000000
s16	31	0x0000000000000000
pc		0x0000000000000000

Messages Run IO

Assembly: assembling F:\Research\misc\JA\CEB03420\tools\test.asm

Turning in F:\Research\misc\JA\CEB03420\tools\test.asm line 312 column 2: RARS does not recognize the global directive. Ignored.

Turning in F:\Research\misc\JA\CEB03420\tools\test.asm line 318 column 2: RARS does not recognize the global directive. Ignored.

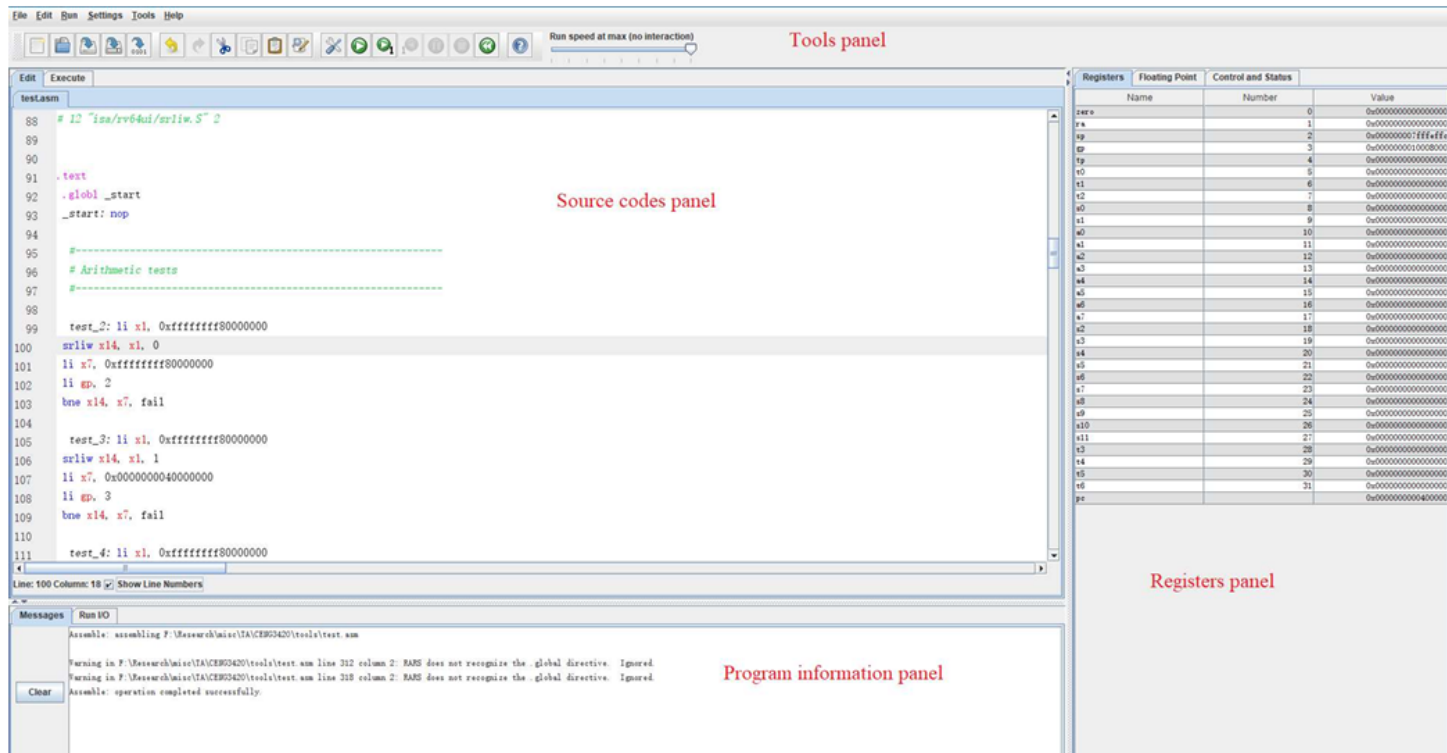
Assembly: operation completed successfully

Clear

RAS Exec. Panel



RISC-V ISA Simulator - RARS





RISC-V ISA Simulator - RARS

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Tools panel

Text segment panel

Byte	Address	Code	Basic	Source
0x04000000	0x00000012	addi a0, a0, 0	93	_start: .nop
0x04000004	0x00000013	li a1, 0xffffffff	99	test_2: li a1, 0xffffffff00000000
0x04000008	0x00000014	addi a1, a1, 0		
0x0400000c	0x00000015	ori a1, a1, 0	100	ori a1, a1, 0
0x04000010	0x00000016	li a7, 0xffffffff	101	li a7, 0xffffffff00000000
0x04000014	0x00000017	addi a7, a7, 0		
0x04000018	0x00000018	addi a5, a5, 2	102	li gp, 2
0x0400001c	0x34771453	bne a14, a7, 0x000000358	103	bne a14, a7, fail
0x04000020	0x00000019	li a1, 0xffffffff	105	test_3: li a1, 0xffffffff00000000
0x04000024	0x0000001a	addi a1, a1, 0		
0x04000028	0x0000001b	ori a1, a1, 1	106	ori a1, a1, 1
0x0400002c	0x0000001c	li a7, 0x00004000	107	li a7, 0x0000000004000000
0x04000030	0x0000001d	addi a7, a7, 0		
0x04000034	0x0000001e	addi a3, a3, 3	108	li gp, 3
0x04000038	0x37714534	bne a14, a7, 0x000000334	109	bne a14, a7, fail
0x0400003c	0x0000001f	li a1, 0xffffffff	111	test_4: li a1, 0xffffffff00000000

Data segment panel

Address	Value (+0)	Value (+4)	Value (+8)	Value (+C)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010004	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010008	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001000c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010010	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010014	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010018	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001001c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010024	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010028	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001002c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010030	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010034	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010038	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001003c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010044	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010048	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001004c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010050	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers panel

Name	Number	Value
zero	0	0x0000000000000000
ra	1	0x0000000000000000
sp	2	0x0000000000000000
gp	3	0x0000000000000000
tp	4	0x0000000000000000
t0	5	0x0000000000000000
t1	6	0x0000000000000000
t2	7	0x0000000000000000
a0	8	0x0000000000000000
a1	9	0x0000000000000000
a2	10	0x0000000000000000
a3	11	0x0000000000000000
a4	12	0x0000000000000000
a5	13	0x0000000000000000
a6	14	0x0000000000000000
a7	15	0x0000000000000000
s0	16	0x0000000000000000
s1	17	0x0000000000000000
s2	18	0x0000000000000000
s3	19	0x0000000000000000
s4	20	0x0000000000000000
s5	21	0x0000000000000000
s6	22	0x0000000000000000
s7	23	0x0000000000000000
s8	24	0x0000000000000000
s9	25	0x0000000000000000
s10	26	0x0000000000000000
s11	27	0x0000000000000000
s12	28	0x0000000000000000
s13	29	0x0000000000000000
s14	30	0x0000000000000000
s15	31	0x0000000000000000
pc		0x0000000000000000

Program information panel

Assembly: assembling F:\Research\misc\TAUCERO3420\tools\test.asm

Turning in F:\Research\misc\TAUCERO3420\tools\test.asm line 312 column 2: RARS does not recognise the global directive. Ignored.

Turning in F:\Research\misc\TAUCERO3420\tools\test.asm line 315 column 2: RARS does not recognise the global directive. Ignored.

Assembly: operation completed successfully.



RISC-V ISA Simulator - RARS

- **RARS shortcut in windows OS**
 - Create a new source file: Ctrl + N
 - Close the current source file: Ctrl + W
 - Assemble the source code: F3
 - Execute the current source code: F5
 - Step running: F7
 - Instructions & System call query: F1