



香港中文大學

The Chinese University of Hong Kong

CENG3420

Lab 1-1: RISC-V Assembly Language Programing I

Lancheng ZOU, Shuo YIN

(Original: Chen BAI, modified by Su ZHENG)

Department of Computer Science & Engineering

Chinese University of Hong Kong

{lczou23, syin22}@cse.cuhk.edu.hk

Spring 2023

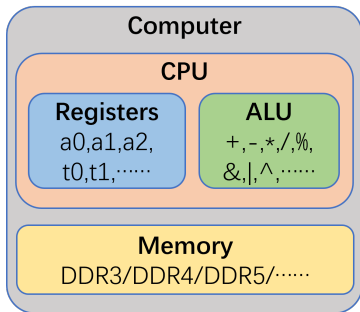
- ① Introduction to Basic RISC-V Assembly Programming
- ② RISC-V ISA Simulator – RARS
- ③ Lab 1-1 Assignment

Introduction to Basic RISC-V Assembly Programming

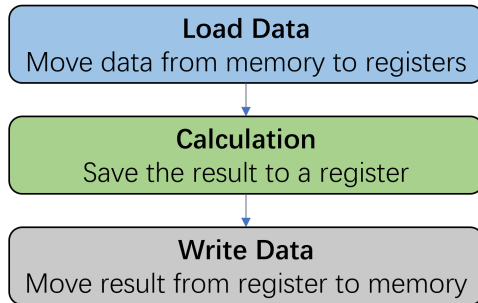
- The RISC-V Instruction Set Manual Volume I: Unprivileged ISA
<https://riscv.org/technical/specifications/>

In all labs. of CENG3420, we focus on RV32I instructions.

- **Computer**, oversimplified.



- **Computing**, oversimplified.



An Example Program

- How to compute " $C = A + B$ "

$\text{resC} = \text{varA} + \text{varB} \Rightarrow \text{resC} = 8$ after execution

```
.globl _start

.data    # global variable declarations follow this line
varA: .word 3 # 1 word = 32 bits
varB: .word 5
resC: .word 0

.text    # instructions follow this line
_start: # a label, marks a position in the code
    la a1, varA # Load varA's address to register a1
    la a2, varB # Load varB's address to register a2
    la a3, resC # Load resC's address to register a3
    lw t1, 0(a1) # Load varA's value to register t1
    lw t2, 0(a2) # Load varB's value to register t2
    add t3, t1, t2 # Register t3 = t1 + t2
    sw t3, 0(a3) # Save register t3 to resC
```

Program Structure I

- Plain text file with data declarations, program code (usually suffixed with *.asm*)
- 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>`
 - `.byte` (1 byte/8 bits), `.2byte`, `.half`, `.short` (2 bytes)
 - `.4byte`, `.word`, `.long` (4 bytes), `.8byte`, `.dword`, `.quad` (8 bytes)
 - `.float`, `.double`,

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

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
```

Data types:

- All instructions are encoding in 32 bits
- Alias: byte (8 bits), halfword (2 bytes), word (4 bytes), double word (8 bytes)

Literals:

- numbers entered as is. *e.g.*, 12 in decimal, and 0xC in hexadecimal
- characters enclosed in single quotes. *e.g.*, 'b'
- strings enclosed in double quotes. *e.g.*, "A string"

- We can manipulate 32 architectural registers in assembly programming directly.
- We prefer using aliases to indicate registers.
- Instructions category
 - Load and store instructions
 - Bitwise instructions
 - Arithmetic instructions
 - Control transfer instructions
 - Pseudo instructions

Register Names and Descriptions

Table: Register names and descriptions

Register Names	ABI Names	Description
x0	zero	Hard-wired zero
x1	ra	Return address
x2	sp	Stack pointer
x3	gp	Global pointer
x4	tp	Thread pointer
x5	t0	Temporary / Alternate link register
x6-7	t1 - t2	Temporary register
x8	s0 / fp	Saved register / Frame pointer
x9	s1	Saved register
x10-11	a0-a1	Function argument / Return value registers
x12-17	a2-a7	Function argument registers
x18-27	s2-s11	Saved registers
x28-31	t3-t6	Temporary registers

Instructions Overview I

LA: The Load Address (*la*) loads the location address of the specified SYMBOL.

Syntax

`la rd, SYMBOL`

Usage

```
.data
NumElements: .byte 6
.text
la x5, NumElements # assign addr[NumElements] to x5
```

LI: The Load Immediate (LI) loads a register (rd) with an immediate value given in the instruction.

Syntax

`li rd, CONSTANT`

Usage

```
li x5,100 # assign 100 to x5
```

LW: The Load Word (LW) instruction does the fetching of 32-bit value from memory and loads into the destination register (rd).

Syntax

```
lw rd, offset(rs1)
```

Usage

```
lw x4, 1352(x9) # assign memory[x9+1352] to x4
```

SW: The Store Word (SW) instruction does the copying of 32-bit value from register (rs2) and loads into the memory(rs1).

Syntax

`sw rs2, offset(rs1)`

Usage

```
sw x4, 1352(x9) # assign x4 to mem[x9+1352]
```

SLL: Shift Logical Left (SLL) performs logical left on the value in register (rs1) by the shift amount held in the register (rs2) and stores in (rd) register.

Syntax

`sll rd, rs1, rs2`

Usage

Instructions Overview IV

```
li x5, 4 # assign 4 to x5
li x3, 2 # assign 2 to x3
sll x1, x5, x3 # assign x5 << x3 to x1
```

SRL: Shift Logically Right (SRL) performs logical Right on the value in register (rs1) by the shift amount held in the register (rs2) and stores in (rd) register.

Syntax

srl rd, rs1, rs2

Usage

```
li x5, 1024 # assign 1024 to x5
li x3, 2     # assign 2 to x3
srl x1, x5, x3 # assign x5 >> x3 to x1
```


SLLI: Shift Logically Left Immediate (SLLI) performs logical left on the value in register (rs1) by the shift amount held in the register (imm) and stores in (rd) register.

Syntax

```
slli rd, rs1, imm
```

Usage

```
slli x1, x1, 3 # assign x1 << 3 to x1
```

SRLI: Shift Logically Right Immediate (SRLI) performs logical Right on the value in register (rs1) by the shift amount held in the register (imm) and stores in (rd) register.

Syntax

`srli rd, rs1, imm`

Usage

```
srli x1, x1, 1 # assign x1 >> 1 to x1
```

For more information about RISC-V instructions and assembly programming you can refer to:

- ① Lecture slides and textbook.
- ② **RARS** Help: F1
- ③ <https://github.com/riscv/riscv-asm-manual/blob/master/riscv-asm.md>
- ④ <https://web.eecs.utk.edu/~smarz1/courses/ece356/notes/assembly/>

RISC-V ISA Simulator – RARS

What is RARS

- **RARS is 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 psuedo-instruction to machine code with intermediate steps.
- You need Java environment to run **RARS**

Download it here:

<https://github.com/TheThirdOne/rars/releases/tag/continuous>

Execute the command to start RARS: `java -jar <rars jar path>`

Launch RARS

```
cbai@hpc1:/research/dept8/gds/cbai/ta/rars$ java -jar rars.jar
```

Launch RARS

RARS Overview

The screenshot displays the RARS application interface. The main window shows assembly code for a MIPS processor. The code includes directives like `.text`, `.globl`, and `.start`, followed by arithmetic tests. The right panel shows the state of registers, floating point registers, control and status registers, and their values. The bottom panel shows messages from the assembler.

Assembly Code:

```
test.asm
88 # 32 "isa/rv64ui/rliw.5" 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 rliw x14, x1, 0
101 li x7, 0xffffffff80000000
102 li w0, 2
103 bne x14, x7, fail
104
105 test_3: li x1, 0xffffffff80000000
106 rliw x14, x1, 1
107 li x7, 0x000000000400000000
108 li w0, 3
109 bne x14, x7, fail
110
111 test_4: li x1, 0xffffffff80000000
112 rliw x14, x1, 1
113
```

Registers:

Name	Number	Value
x0	0	0x0000000000000000
x1	1	0x0000000000000000
x2	2	0x0000000000000000
x3	3	0x0000000000000000
x4	4	0x0000000000000000
x5	5	0x0000000000000000
x6	6	0x0000000000000000
x7	7	0x0000000000000000
x8	8	0x0000000000000000
x9	9	0x0000000000000000
x10	10	0x0000000000000000
x11	11	0x0000000000000000
x12	12	0x0000000000000000
x13	13	0x0000000000000000
x14	14	0x0000000000000000
x15	15	0x0000000000000000
x16	16	0x0000000000000000
x17	17	0x0000000000000000
x18	18	0x0000000000000000
x19	19	0x0000000000000000
x20	20	0x0000000000000000
x21	21	0x0000000000000000
x22	22	0x0000000000000000
x23	23	0x0000000000000000
x24	24	0x0000000000000000
x25	25	0x0000000000000000
x26	26	0x0000000000000000
x27	27	0x0000000000000000
x28	28	0x0000000000000000
x29	29	0x0000000000000000
x30	30	0x0000000000000000
x31	31	0x0000000000000000
pc		0x0000000000000000

Messages:

```
Assembly: assembling F:\Research\misc\TA\CBR3420\tools\test.asm
Parsing in F:\Research\misc\TA\CBR3420\tools\test.asm line 212 column 2: RARS does not recognize the .globl directive. Ignored.
Parsing in F:\Research\misc\TA\CBR3420\tools\test.asm line 318 column 2: RARS does not recognize the .globl directive. Ignored.
Assembly: operation completed successfully.
```

File Edit Bus Settings Tools Help

Run speed at max (no interaction)

Execute

Test Segment

Bit	Address	Code	Basic	Source
0x040000	0x00000013	add, r0, r0, 0	93	_start, nop
0x040004	0x00000014	li, r1, 0xffffffff00000000	99	test, 2, li, r1, 0xffffffff00000000
0x040008	0x00000015	add, r1, r1, 0	100	mov, r14, r1, 0
0x04000c	0x00000016	li, r1, 0xffffffff00000000	101	li, r1, 0xffffffff00000000
0x040010	0x00000017	add, r1, r1, 0	102	li, r1, 0
0x040014	0x00000018	add, r1, r1, 0	103	li, r1, 0
0x040018	0x00000019	add, r1, r1, 0	104	li, r1, 0
0x04001c	0x0000001a	add, r1, r1, 0	105	li, r1, 0
0x040020	0x0000001b	add, r1, r1, 0	106	li, r1, 0
0x040024	0x0000001c	add, r1, r1, 0	107	li, r1, 0
0x040028	0x0000001d	add, r1, r1, 0	108	li, r1, 0
0x04002c	0x0000001e	add, r1, r1, 0	109	li, r1, 0
0x040030	0x0000001f	add, r1, r1, 0	110	li, r1, 0
0x040034	0x00000020	add, r1, r1, 0	111	li, r1, 0

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
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

0x10010000 (data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages Run IO

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

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

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

Assembly: operation completed successfully.

Clear

Registers Floating Point Control and Status

Name	Number	Value
pc	0	0x0000000000000000
r0	1	0x0000000000000000
r1	2	0x0000000000000000
r2	3	0x0000000000000000
r3	4	0x0000000000000000
r4	5	0x0000000000000000
r5	6	0x0000000000000000
r6	7	0x0000000000000000
r7	8	0x0000000000000000
r8	9	0x0000000000000000
r9	10	0x0000000000000000
r10	11	0x0000000000000000
r11	12	0x0000000000000000
r12	13	0x0000000000000000
r13	14	0x0000000000000000
r14	15	0x0000000000000000
r15	16	0x0000000000000000
r16	17	0x0000000000000000
r17	18	0x0000000000000000
r18	19	0x0000000000000000
r19	20	0x0000000000000000
r20	21	0x0000000000000000
r21	22	0x0000000000000000
r22	23	0x0000000000000000
r23	24	0x0000000000000000
r24	25	0x0000000000000000
r25	26	0x0000000000000000
r26	27	0x0000000000000000
r27	28	0x0000000000000000
r28	29	0x0000000000000000
r29	30	0x0000000000000000
r30	31	0x0000000000000000
r31	32	0x0000000000000000

RARS Basic Introduction

The screenshot displays the RARS application window, which is divided into several panels:

- Tools panel:** Located at the top right, it contains a toolbar with icons for file operations (open, save, print, etc.) and a status bar indicating "Run speed at max (no interaction)".
- Source codes panel:** The central area displays the assembly code for a test program. The code includes comments, directives, and instructions for testing arithmetic operations. The code is as follows:

```
test.asm
88 # 32 "isa/rv64ui/rvliw.5" 2
89
90
91 .test
92 .globl _start
93 _start: nop
94
95 #-----
96 # Arithmetic tests
97 #-----
98
99 test_0: li x1, 0xffffffff80000000
100 srlw x14, x1, 0
101 li x7, 0xffffffff80000000
102 li x5, 2
103 bne x14, x7, fail
104
105 test_3: li x1, 0xffffffff80000000
106 srlw x14, x1, 1
107 li x7, 0x000000000400000000
108 li x5, 3
109 bne x14, x7, fail
110
111 test_4: li x1, 0xffffffff80000000
112 li
```
- Registers panel:** Located on the right side, it displays a table of registers. The table has columns for Name, Number, and Value. The registers are listed in hexadecimal format.
- Program information panel:** Located at the bottom, it displays messages and run information. The messages include:

```
Assembly: assembling F:\Research\misc\TA\CBM32420\tools\test.asm
Parsing in F:\Research\misc\TA\CBM32420\tools\test.asm line 312 column 2: RARS does not recognize the .global directive. Ignored.
Parsing in F:\Research\misc\TA\CBM32420\tools\test.asm line 318 column 2: RARS does not recognize the .global directive. Ignored.
Assembly: operation completed successfully.
```

File Edit Bus Settings Tools Help

Run speed at max (no interaction)

Tools panel

Text segment panel

Bit	Address	Code	Basic	Source
0x040000	0x00000013	addi x0, x0, 0	93	_start: nop
0x040004	0x00000014	lui x1, 0xffff0000	99	test_2: li x1, 0xffffffff00000000
0x040008	0x00000015	addi x1, x1, 0	100	movi x14, x1, 0
0x04000c	0x00000016	lui x1, 0xffff0000	101	li x1, 0xffffffff00000000
0x040010	0x00000017	addi x1, x1, 0	102	li gp, 2
0x040014	0x00000018	addi x1, x1, 0	103	hse x14, x1, fail
0x040018	0x00000019	addi x1, x1, 0	105	test_3: li x1, 0xffffffff00000000
0x04001c	0x0000001a	addi x1, x1, 0	106	movi x14, x1, 1
0x040020	0x0000001b	addi x1, x1, 0	107	li x1, 0xffffffff00000000
0x040024	0x0000001c	addi x1, x1, 0	108	li gp, 3
0x040028	0x0000001d	addi x1, x1, 0	109	hse x14, x1, fail
0x04002c	0x0000001e	addi x1, x1, 0	111	test_4: li x1, 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
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (data) Hexadecimal Addresses Hexadecimal Values ASCII

Registers panel

Registers	Floating Point	Control and Status
Name	Number	Value
x0	0	0x0000000000000000
x1	1	0x0000000000000000
x2	2	0x0000000000000000
x3	3	0x0000000000000000
x4	4	0x0000000000000000
x5	5	0x0000000000000000
x6	6	0x0000000000000000
x7	7	0x0000000000000000
x8	8	0x0000000000000000
x9	9	0x0000000000000000
x10	10	0x0000000000000000
x11	11	0x0000000000000000
x12	12	0x0000000000000000
x13	13	0x0000000000000000
x14	14	0x0000000000000000
x15	15	0x0000000000000000
x16	16	0x0000000000000000
x17	17	0x0000000000000000
x18	18	0x0000000000000000
x19	19	0x0000000000000000
x20	20	0x0000000000000000
x21	21	0x0000000000000000
x22	22	0x0000000000000000
x23	23	0x0000000000000000
x24	24	0x0000000000000000
x25	25	0x0000000000000000
x26	26	0x0000000000000000
x27	27	0x0000000000000000
x28	28	0x0000000000000000
x29	29	0x0000000000000000
x30	30	0x0000000000000000
x31	31	0x0000000000000000
pc		0x0000000000000000

Program information panel

Messages Run IO

Assembly: assembling F:\Research\misc\TACB03420\tools\text.asm

Warning in F:\Research\misc\TACB03420\tools\text.asm line 312 column 2: RARS does not recognize the global directive. Ignored.

Warning in F:\Research\misc\TACB03420\tools\text.asm line 312 column 2: RARS does not recognize the global directive. Ignored.

Assembly: operation completed successfully.

Clear

- 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

An Example Program

Hello CENG3420

```
.globl _start

.data    # global variable declarations follow this line
welcome_msg: .asciz "Welcome_to_CENG3420!\n"

.text    # instructions follow this line
_start: # a label, marks a position in the code
    addi a0, x0, 1 # STDOUT=1
    la a1, welcome_msg # Load the address of welcome_msg
    addi a2, x0, 21 # Length of the string
    addi a7, x0, 64 # Specify the system call number
    ecall # Raise a system call
# End of program, leave a blank line afterwards is preferred
```

An Example Program

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x00109513	addi x10,x0,1	9: addi a0, x0, 1
	0x00400004	0x0fc10597	auipc x11,0x0000fc10	11: la a1, welcome_msg
	0x00400008	0xffc58593	addi x11,x11,0xffff...	
	0x0040000c	0x01500613	addi x12,x0,21	13: addi a2, x0, 21
	0x00400010	0x04000893	addi x17,x0,0x00000040	15: addi a7, x0, 64
	0x00400014	0x00000073	ecall	17: ecall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00400000	0x00109513	0x0fc10597	0xffc58593	0x01500613	0x04000893	0x00000073	0x00000000	0x00000000
0x00400004	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400008	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0040000c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400010	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400014	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400018	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0040001c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400024	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400028	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0040002c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400030	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400034	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400038	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0040003c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400044	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400048	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0040004c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400050	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400054	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400058	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0040005c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400064	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400068	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0040006c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400070	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400074	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400078	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0040007c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400084	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400088	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0040008c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400090	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400094	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400098	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x0040009c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000a4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000a8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000ac	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000b0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000b4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000b8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000bc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000c4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000c8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000cc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000d0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000d4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000d8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000dc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000e4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000e8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000ec	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000f0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000f4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000f8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000fc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Control and Status

Registers Floating Point

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffffc
gp	3	0x10000000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
s0	10	0x00000015
a1	11	0x10010000
a2	12	0x00000015
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000040
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		0x0040001c

Messages Run I/O

Welcome to ENG3420!

-- program is finished running (dropped off bottom) --

Clear

RARS provides a small set of operating system-like services through the system call (`ecall`) instruction. Register contents are not affected by a system call, except for result registers in some instructions.

- Load the service number (or number) in register `a7`.
- Load argument values, if any, in `a0`, `a1`, `a2` ..., as specified.
- Issue `ecall` instruction.
- Retrieve return values, if any, from result registers as specified.

System Calls in RARS II

Name	Number	Description	Inputs	Outputs
PrintInt	1	Prints an integer	a0 = integer to print	N/A
PrintFloat	2	Prints a float point number	fa0 = float to print	N/A
PrintString	4	Prints a null-terminated string to the console	a0 = the address of the string	N/A
ReadInt	5	Reads an int from input console	a0 = the int	N/A
ReadFloat	6	Reads a float from input console	fa0 = the float	N/A
ReadString	8	Reads a string from the console	a0 = address of input buffer, a1 = maximum number of characters to read	N/A
Open	1024	Opens a file from a path Only supported flags (a1), read-only (0), write-only (1) and write-append (9)	a0 = Null terminated string for the path, a1 = flags	a0 = the file decriptor or -1 if an error occurred
Read	63	Read from a file descriptor into a buffer	a0 = the file descriptor, a1 = address of the buffer, a2 = maximum length to read	a0 = the length read or -1 if error
Write	64	Write to a filedescriptor from a buffer	a0 = the file descriptor, a1 = the buffer address, a2 = the length to write	a0 = the number of charcters written
LSeek	62	Seek to a position in a file	a0 = the file descriptor, a1 = the offset for the base, a2 is the beginning of the file (0), the current position (1), or the end of the file (2)}	a0 = the selected position from the beginning of the file or -1 is an error occurred

An Example of System Calls in RARS I

An example shows how to use system calls in RARS

Using system call

```
# Comment giving name of program and description
# sys-call.asm
# Bare-bones outline of RISC-V assembly language program
    .globl _start

    .data
msg: .asciz "Hello, _world!\n"

    .text
_start:
li a7, 4      # system call code for PrintString
la a0, msg    # address of string to print
ecall          # Use the system call
# End of program, leave a blank line afterwards is preferred
```

You can check the output in Run/IO of the program information panel.

An Example of System Calls in RARS II

- *li* loads a register with an immediate value given in the instruction.
- *la* loads an address of the specified symbol.
- *.asciz* emits the specified string within double quotes and includes the terminated zero character at the end.

Lab 1-1 Assignment

Write a RISC-V assembly program step by step as shown below:

- ① Define three variables `var1`, `var2` and `var3` which will be loaded from **terminal** using `syscall`.
- ② Increase `var1` by 3, multiply `var2` by 2.
- ③ increase `var3` by `var1 + var2`.
- ④ print `var1`, `var2` and `var3` to **terminal** using `syscall`.

Submission Method:

Submit the source code and report to **Blackboard**.

★: Submit a **YourStudentID.zip** must contain **lab1-1.asm** and **lab1-1_report.pdf**.

Example: For a student with ID 123456, you should submit a 123456.zip file.

```
./
├── lab1-1.asm
└── lab1-1_report.pdf
```

If your submitted files do not satisfy the above structure, your lab score will be zero.

Some Tips

- 1 Variables should be declared following the `.data` identifier.
- 2 `<name>: .<datatype> <value>`
- 3 Use `la` instruction to access the RAM address of declared data.
- 4 Use system call to read and print from the terminal.
- 5 Do not forget `\n`.
- 6 Do not forget exit system call.

Example:

Input:

1
2
3

Output:

4
4
11