香 港 中 文 大 學
The Chinese University of Hong Kong

CENG3420
Lab Overview & Introduction to RARS

Mingjun Li

Department of Computer Science & Engineering
Chinese University of Hong Kong
{mjli23}@cse.cuhk.edu.hk

Spring 2025

Outline

Overview of CENG3420 Labs

Assembler & Simulator

- Assembly language – symbolic
- Machine language – binary

- Assembler is a program that
  - turns symbols into machine instructions (e.g., riscv64-unknown-elf-as)

- Simulator is a program that
  - mimics the behavior of a processor
  - usually written in high-level language (e.g., spike)

CENG3420 Lab Contents

We have 3 labs in total, with 2-3 sub-labs for each lab.

- **Lab1**: RISC-V assembly language programming using **RARS** simulator.
- In lab1, we will practice coding in RISC-V assembly language, and understand how our codes run in a RISC-V CPU.
  - **Lab1-1**: basic operators and system call.
  - **Lab1-2**: function call and simple algorithm implementation.
  - **Lab1-3**: stack data structure, recursive function call, more complex algorithm implementation.

CENG3420 Lab Contents

We have 3 labs in total, with 2-3 sub-labs for each lab.

- **Lab2**: build(complete) a C-based RISC-V assembler and simulator.
- Codebase: `https://github.com/MingjunLi99/ceng3420`. We need to implement the assembler and simulator based on the codebase.
    - Lab2-1: implement a RISC-V assembler.
    - Lab2-2: implement a RISC-V ISA simulator with:
        - RISC-V 32 general-purpose registers
        - 32-bit data and address
        - 25+ instructions (including pseudo instructions)

CENG3420 Lab Contents

We have 3 labs in total, with 2-3 sub-labs for each lab.

- **Lab3**: build a more complete C-based RISC-V Simulator based on lab2.
  - Lab3-1: control logic in CPU, finite state machine.
  - Lab3-2: execution model, memory interface.
  - Lab3-3: BUS driver, etc.

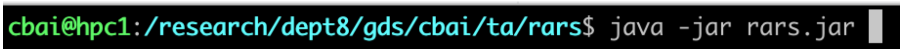
RISC-V ISA Simulator – RARS

What is RARS

- **RARS is the RISC-V Assembler, Runtime and Simulator for RISC-V assembly language programs.**
- We write codes in RISC-V assembly language, then **RARS** translates them into RISC-V instructions and corresponding machine codes, then execute the codes through simulation, like a RISC-V CPU.
- **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.

Launch RARS

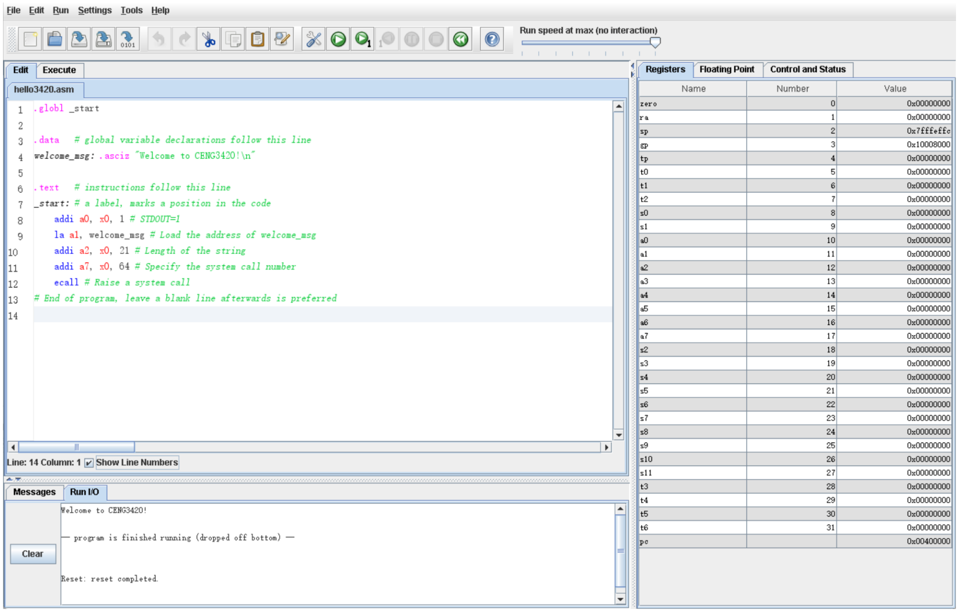
- RARS tutorial: `https://cass-kul.github.io/tutorials/rars/`
- Install Java environment: `https://java.com/en/download/`
- Dowload RARS: `https://github.com/TheThirdOne/rars/releases/tag/continuous`
- Run RARS: run command java -jar <rars jar path> in the command window, under the path where you place rars.jar



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

- We also provide Java install package and RARS in RARS.zip on **Blackboard**.

RARS Overview



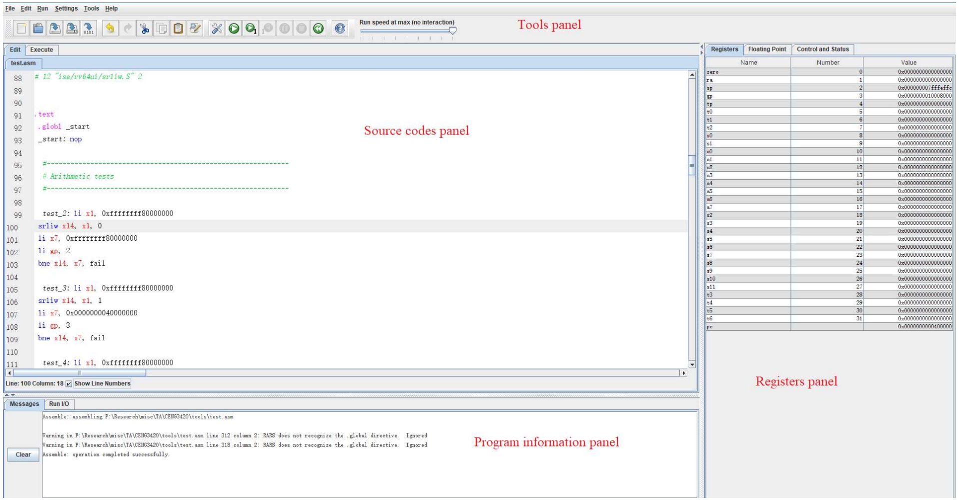
```
.globl _start

.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

```

Name	Number	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
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400000

```
Welcome to CENG3420!

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


Reset: reset completed.
```

RARS edit panel

RARS Overview

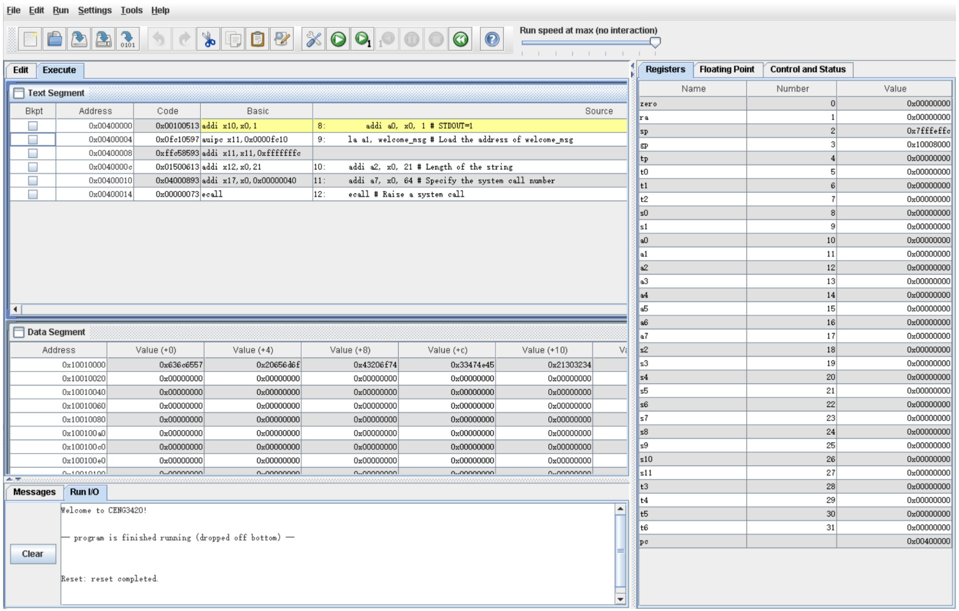

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Tools panel
Edit Execute
test.asm
Source codes panel
Line: 100 Column: 18 ✓ Show Line Numbers
Messages Run I/O
Clear
Program information panel
Registers Floating Point Control and Status
Registers panel

```
# 12 "isa/rv64ui/srliw.S" 2


.text
 .globl _start
 _start: nop

  #-------------------------------------------------------------
  # Arithmetic tests
  #-------------------------------------------------------------

  test_2: li x1, 0xffffffff80000000
 srliw x14, x1, 0
 li x7, 0xffffffff80000000
 li gp, 2
 bne x14, x7, fail

  test_3: li x1, 0xffffffff80000000
 srliw x14, x1, 1
 li x7, 0x0000000040000000
 li gp, 3
 bne x14, x7, fail

  test_4: li x1, 0xffffffff80000000
```

```
Assemble: assembling F:\Research\misc\TA\CENG3420\tools\test.asm

Warning in F:\Research\misc\TA\CENG3420\tools\test.asm line 312 column 2: RARS does not recognize the .global directive.  Ignored.
Warning in F:\Research\misc\TA\CENG3420\tools\test.asm line 318 column 2: RARS does not recognize the .global directive.  Ignored.
Assemble: operation completed successfully.
```

Name	Number	Value
zero	0	0x0000000000000000
ra	1	0x0000000000000000
sp	2	0x000000007fffeffc
gp	3	0x0000000010008000
tp	4	0x0000000000000000
t0	5	0x0000000000000000
t1	6	0x0000000000000000
t2	7	0x0000000000000000
s0	8	0x0000000000000000
s1	9	0x0000000000000000
a0	10	0x0000000000000000
a1	11	0x0000000000000000
a2	12	0x0000000000000000
a3	13	0x0000000000000000
a4	14	0x0000000000000000
a5	15	0x0000000000000000
a6	16	0x0000000000000000
a7	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
t3	28	0x0000000000000000
t4	29	0x0000000000000000
t5	30	0x0000000000000000
t6	31	0x0000000000000000
pc		0x0000000000400000

RARS Overview


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Registers Floating Point Control and Status
Messages Run I/O
Clear

Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x00100513	addi x10, x0, 1	8: addi a0, x0, 1 # STDOUT=1
☐	0x00400004	0x0fc10597	auipc x11, 0x0000fc10	9: la a1, welcome_msg # Load the address of welcome_msg
☐	0x00400008	0xffc58593	addi x11, x11, 0xfffffffc	
☐	0x0040000c	0x01500613	addi x12, x0, 21	10: addi a2, x0, 21 # Length of the string
☐	0x00400010	0x04000893	addi x17, x0, 0x00000040	11: addi a7, x0, 64 # Specify the system call number
☐	0x00400014	0x00000073	ecall	12: ecall # Raise a system call

Data Segment

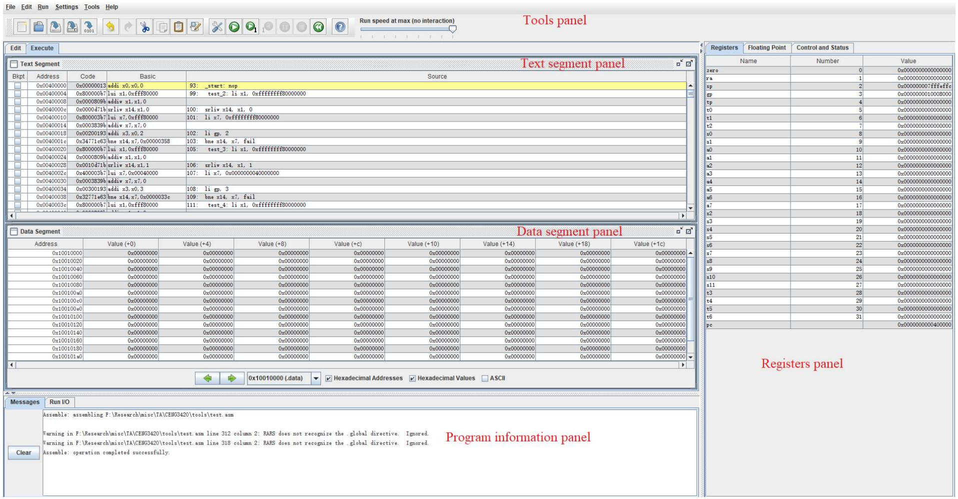
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	0x636c6557	0x20656d6f	0x43206f74	0x33474e45	0x21303234
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

```
Welcome to CENG3420!

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


Reset: reset completed.
```

Name	Number	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
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400000

RARS execute panel

RARS Overview



Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x00000013	addi x0, x0, 0	93: _start: nop
☐	0x00400004	0x800000b7	lui x1, 0xfff80000	99: test_2: li x1, 0xffffffff80000000
☐	0x00400008	0x0000809b	addiw x1, x1, 0	
☐	0x0040000c	0x0000d71b	srliw x14, x1, 0	100: srliw x14, x1, 0
☐	0x00400010	0x800003b7	lui x7, 0xfff80000	101: li x7, 0xffffffff80000000
☐	0x00400014	0x0003839b	addiw x7, x7, 0	
☐	0x00400018	0x00200193	addi x3, x0, 2	102: li gp, 2
☐	0x0040001c	0x34771c63	bne x14, x7, 0x00000358	103: bne x14, x7, fail
☐	0x00400020	0x800000b7	lui x1, 0xfff80000	105: test_3: li x1, 0xffffffff80000000
☐	0x00400024	0x0000809b	addiw x1, x1, 0	
☐	0x00400028	0x0010d71b	srliw x14, x1, 1	106: srliw x14, x1, 1
☐	0x0040002c	0x400003b7	lui x7, 0x00040000	107: li x7, 0x0000000040000000
☐	0x00400030	0x0003839b	addiw x7, x7, 0	
☐	0x00400034	0x00300193	addi x3, x0, 3	108: li gp, 3
☐	0x00400038	0x32771e63	bne x14, x7, 0x0000033c	109: bne x14, x7, fail
☐	0x0040003c	0x800000b7	lui x1, 0xfff80000	111: test_4: li x1, 0xffffffff80000000

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
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

Name	Number	Value
zero	0	0x0000000000000000
ra	1	0x0000000000000000
sp	2	0x000000007fffeffc
gp	3	0x0000000010008000
tp	4	0x0000000000000000
t0	5	0x0000000000000000
t1	6	0x0000000000000000
t2	7	0x0000000000000000
s0	8	0x0000000000000000
s1	9	0x0000000000000000
a0	10	0x0000000000000000
a1	11	0x0000000000000000
a2	12	0x0000000000000000
a3	13	0x0000000000000000
a4	14	0x0000000000000000
a5	15	0x0000000000000000
a6	16	0x0000000000000000
a7	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
t3	28	0x0000000000000000
t4	29	0x0000000000000000
t5	30	0x0000000000000000
t6	31	0x0000000000000000
pc		0x0000000000400000

```
Assemble: assembling F:\Research\misc\TA\CENG3420\tools\test.asm

Warning in F:\Research\misc\TA\CENG3420\tools\test.asm line 312 column 2: RARS does not recognize the .global directive.  Ignored.
Warning in F:\Research\misc\TA\CENG3420\tools\test.asm line 318 column 2: RARS does not recognize the .global directive.  Ignored.
Assemble: operation completed successfully.
```

Shortcuts in Windows

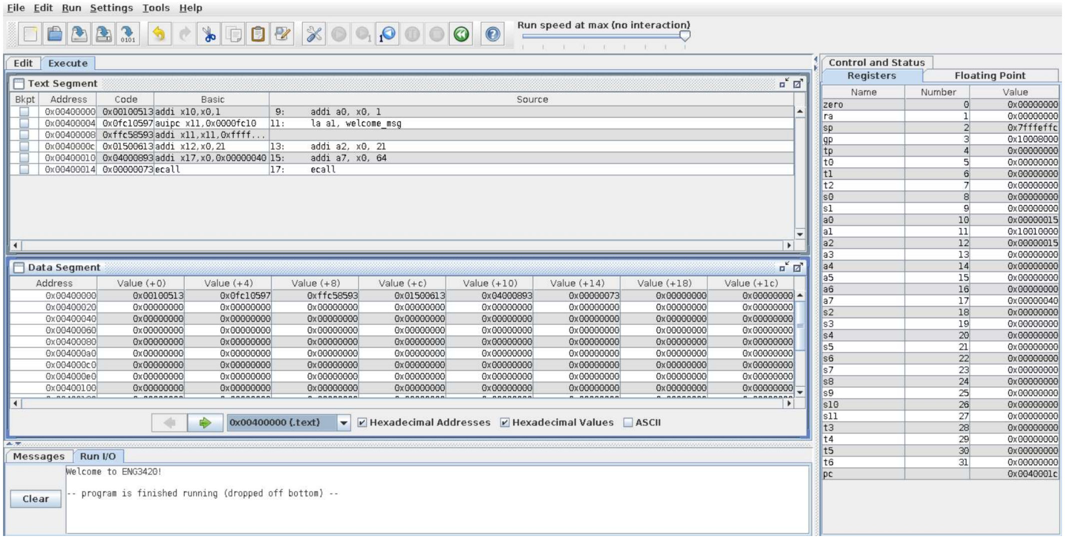
- Create a new source file: Ctrl + N
- Close the current source file: Ctrl + W
- Assemble the source code: F3
- Execute the current source code: F5
- Step running: F7
- Instructions & System call query: F1

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



Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x00100513	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	0x00100513	0x0fc10597	0xffc58593	0x01500613	0x04000893	0x00000073	0x00000000	0x00000000
0x00400020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x00400000 (.text) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Name	Number	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	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

```
Welcome to ENG3420!

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

System Calls in RARS

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

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 begining 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

THANK YOU!