HW1: Human compiler

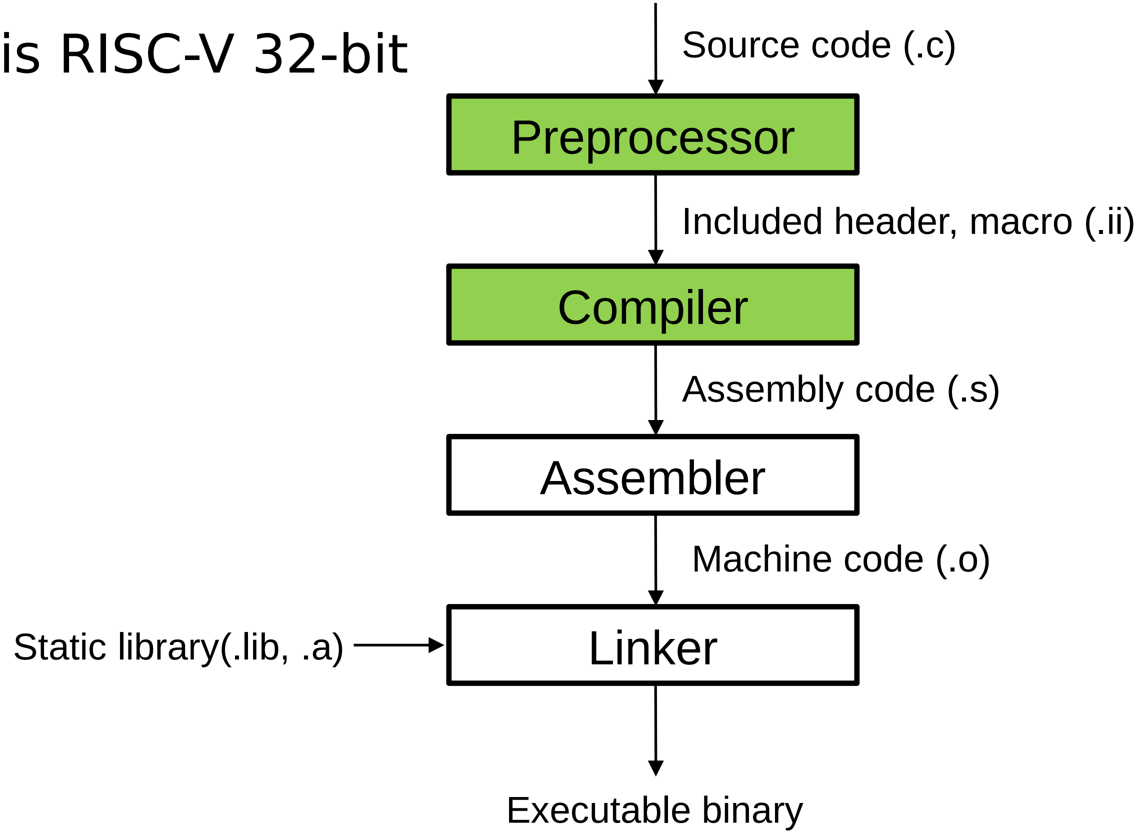
Prof. Jae W. Lee(jaewlee@snu.ac.kr)
Department of Computer Science and Engineering
Seoul National University

TA: Jeonghun Gong, Yunho Jin

Goal of this project

- **You will compile given C source code into assembly code.**

  Target architecture is RISC-V 32-bit



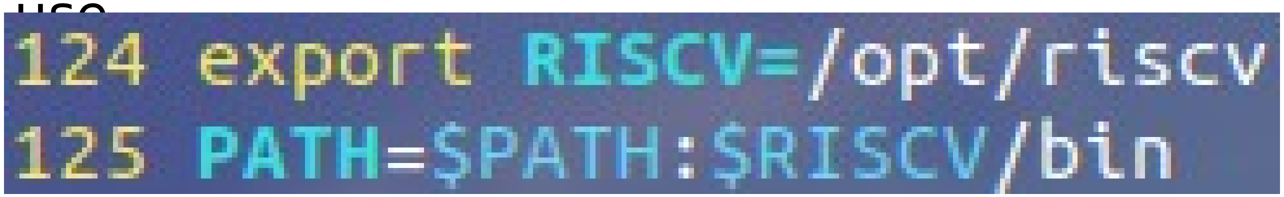
Experimental setup

- **You will use RISC-V ISA simulator on linux.**

  https://github.com/riscv/riscv-isa-sim

  It is already installed on Hardware lab computers.

  Just add these two lines on your `~/.bashrc` before first use.



```
export RISCV=/opt/riscv
PATH=$PATH:$RISCV/bin
```

  Then, type “`source ~/.bashrc`” on your command line.

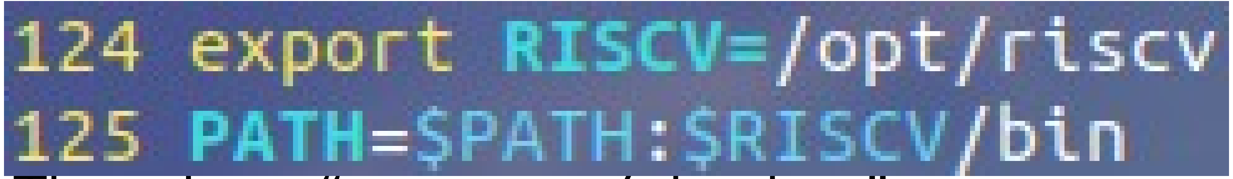
  Now, you’re good to go with your code!

Experimental setup

- **Self setup (On Ubuntu (Debian) linux)**

  Before start, add these two lines on your ~/.bashrc

  You can use other 'RISCV' installation path if you want.



```
export RISCV=/opt/riscv
PATH=$PATH:$RISCV/bin
```

  Then, type "source ~/.bashrc" on your command line

  Make directory using mkdir command.

```
$> sudo mkdir $RISCV
$> sudo chown –R [your_username] $RISCV
```

Experimental setup

Self setup (On Ubuntu (Debian) linux)

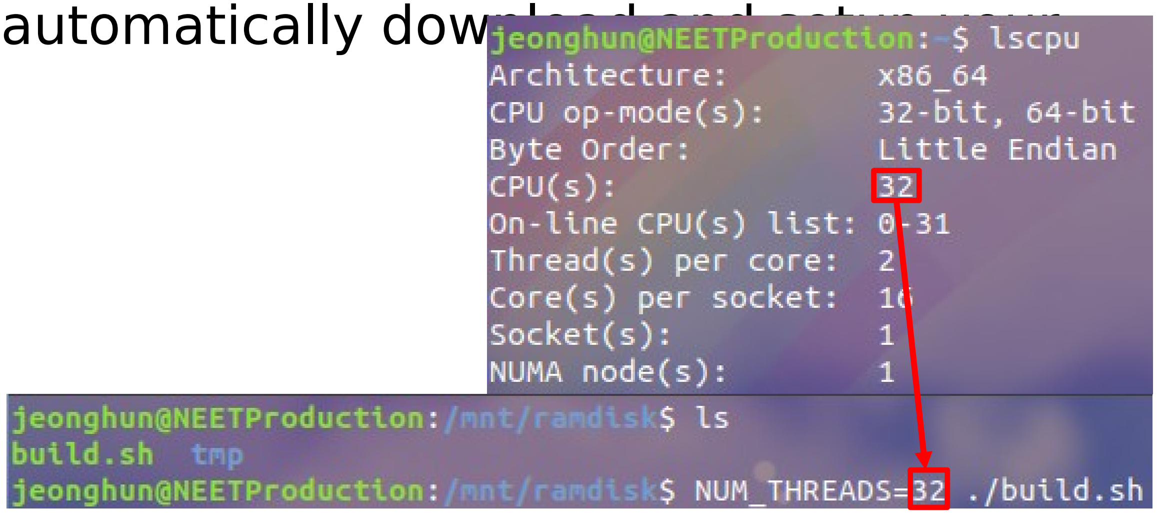
Download setup.sh from eTL.

Before get started, check the number of CPU cores of your PC

with `lscpu` command.

Run build.sh with argument `NUM_THREADS=[core count]`

This script will automatically download and setup your environment.



```
jeonghun@NEETProduction:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    2
Core(s) per socket:    16
Socket(s):             1
NUMA node(s):          1
```

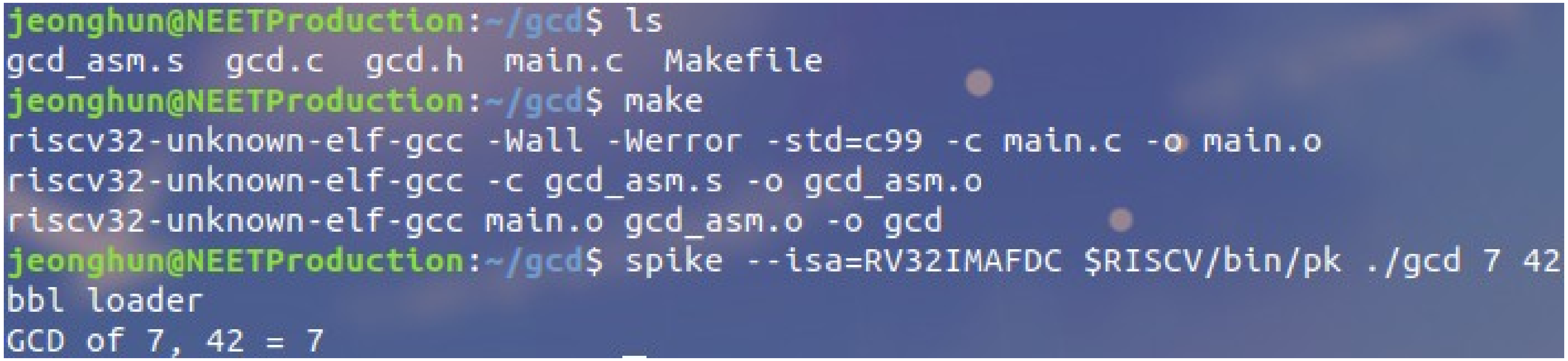
```
jeonghun@NEETProduction:/mnt/ramdisk$ ls
build.sh  tmp
jeonghun@NEETProduction:/mnt/ramdisk$ NUM_THREADS=32 ./build.sh
```

Experimental setup

- **Execution of your code.**

  $> Make

  $> spike --isa=RV32IMAFDC $RISCV/bin/pk ./binary [arg1] [arg2] …



```
jeonghun@NEETProduction:~/gcd$ ls
gcd_asm.s  gcd.c  gcd.h  main.c  Makefile
jeonghun@NEETProduction:~/gcd$ make
riscv32-unknown-elf-gcc -Wall -Werror -std=c99 -c main.c -o main.o
riscv32-unknown-elf-gcc -c gcd_asm.s -o gcd_asm.o
riscv32-unknown-elf-gcc main.o gcd_asm.o -o gcd
jeonghun@NEETProduction:~/gcd$ spike --isa=RV32IMAFDC $RISCV/bin/pk ./gcd 7 42
bbl loader
GCD of 7, 42 = 7
```

Problem 1. Greatest common divisor

- **Calculate the Greatest common divisor (GCD) of two integers.**

  Write your code on `gcd_asm.s`

  Refer to `gcd.c` for algorithm.

  Operands are stored at register a0, a1.

  Store the answer to register a0 and return.

  Execution:

  ```
  $> spike --isa=RV32IMAFDC $RISCV/bin/pk ./gcd [lhs]
  [rhs]
  ```

Problem 2. Fibonacci sequence

- **Store the given count of Fibonacci sequence on specified memory location.**

  Write your code on `fibonacci_asm.s`

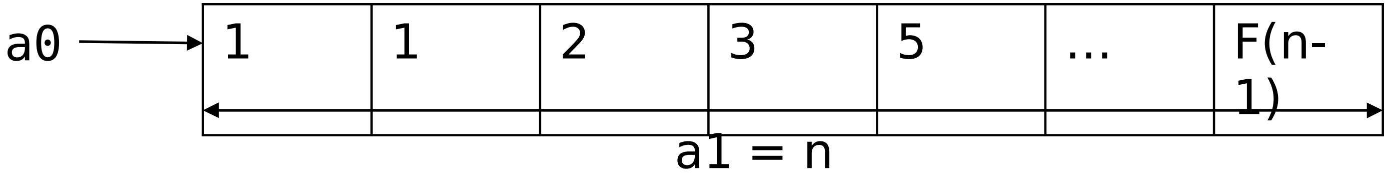
  Memory address is stored on register a0

  Count (number to calculate) is stored on register a1.

  Return value is the memory address having answer.

  Execution:

  ```
  $> spike --isa=RV32IMAFDC $RISCV/bin/pk ./fibonacci
  [count]
  ```



Problem 3. Maze solving

- **Find out the length of the shortest path to solve given maze.**

  Maze is stored in array (reg a0).

  Width (reg a1) and height (reg a2) of array are given.

  Each entry of array represents the state of pix

  (1: Blocked, 0: Opened)

  Starting point is (0,0) of array.

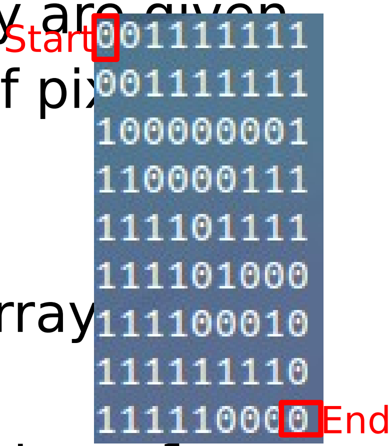
  Ending point is (width – 1, height – 1) of array

  Refer to `maze.c` for algorithm.

  If this maze can't be solved in 20 steps, return -1.

  Execution:

```
$> spike --isa=RV32IMAFDC $RISCV/bin/pk ./maze [file-
name]
```



```
001111111
001111111
100000001
110000111
111101111
111101000
111100010
111111110
111110000
```

Submission

- **Report**
  - Briefly describe your implementation within 5 pages.
  - Filename: [student_id].pdf (example: 2019-12345.pdf)
  - **Please** submit it in **PDF** format. Other formats are not accepted.
- **Compress your source code and report into single zip file.**
  - Compress gcd_asm.s, fibonacci_asm.s, maze_asm.s and your report.
  - Filename should be [student_id].zip (example: 2019-12345.zip).
  - **Please** submit it in **ZIP** format. Other formats are not accepted.
- **Submission deadline: Before 2019. 9. ?? 23:59**