

# Computer Organization Project #1

TA in charge: Myeongjae Jang

E-mail: [myeongjae0409@kaist.ac.kr](mailto:myeongjae0409@kaist.ac.kr)

Office Hour: 18:30 ~ 20:00, Tue. / Thu. (N1, 403)

## I. Goal of this project

- ✓ Understand what a computer architecture simulator is
- ✓ Understand what a benchmark is and how benchmarks are different to each other
- ✓ Execute a simulator, 'SimpleScalar-ARM' with a benchmark suite, 'MiBench'
- ✓ Analyze simulation results

## II. Submission and grading

- ✓ You must submit all of followings (**Total 100 points**):
  - A. 4 simulation result files (**20 points**):

After appropriate simulation, there will be four result files  
<adpcm\_result.txt, lame\_result.txt, tiff2rgba\_result.txt, crc32\_result.txt>.  
*You must submit all these results files without any modifications.*
  - B. Analysis report about simulation results (**80 points**):

With simulation result files, you will analyze characteristics of four benchmarks in terms of the number of instructions, IPC, memory utilization, etc. *There are not any specific formats for the report.* You can use any information if it is in result files. You can include anything such as a table, graph, chart, screen captures, etc.

The report file name must be <Report\_StudentID\_NAME.pdf>.  
ex) Report\_20173496\_MyeongjaeJang.pdf  
*You can use either English or Korean as you want.*
- ✓ Compress all these files into a .zip file and upload the compressed file on KLMS.
- ✓ The submitted file name must be <Project1\_StudentID\_NAME.zip>.  
ex) Project1\_20173496\_MyeongjaeJang.zip
- ✓ Project #1 is an individual assignment. You can talk and discuss to each other, but *you cannot simulate benchmarks or write a report with others.*

### III. Due date

- ✓ Oct. 5<sup>th</sup> (Fri.), 23:59
- ✓ Late submission due date: Oct. 6<sup>th</sup> (Sat.), 23:59
- ✓ After the due date, there will be **50% penalty on your project #1 score**.  
If you submit after the due date, your maximum score will be 50 points.
- ✓ After the late submission due date, you will get **0 point**.

### IV. Cheating

- ✓ If there are any cheatings in your submission, you will get **0 point**.
- ✓ *Followings will be regarded as cheating:*
  - A. Copying other students' simulation results or reports
  - B. Modifying other students' results and using them as if they were your own
  - C. Using other sources without any references excluding your own simulation results
  - D. All other sorts of inappropriate behaviors

### V. Prerequisite

- ✓ You will use virtual machine, 'VirtualBox' with a Linux image uploaded on KLMS.
- ✓ *You should follow another guideline, <[linux\\_image\\_setup\\_guide.pdf](#)> to install 'VirtualBox', unzip the Linux image, and execute it on 'VirtualBox'.*
- ✓ TAs have already confirmed that the Linux image works well if you follow the guideline correctly.
- ✓ If there are some problems even though you follow the guideline, please contact to TA.

### VI. Simulator and benchmark

- ✓ Simulator: SimpleScalar-ARM  
'SimpleScalar' is one of the famous simulator used in computer architecture. With this simulator, you can build a modeling applications for program performance analysis.  
Especially, 'SimpleScalar-ARM' is one of various versions of 'SimpleScalar'. It is for testing ARM-core based systems (ARM is one of companies like Intel).  
If you want to know about 'SimpleScalar' simulator, please access <http://www.simplescalar.com/>.

✓ Benchmark suite: MiBench

Basically, benchmarks are a kind of testing programs to quantify relative performance of various computing systems. ‘MiBench’ is one of open source benchmark suites, which is a set of several benchmarks. ‘MiBench’ is supporting ARM embedded processors. It provides 35 benchmarks divided into 6 categories.

In this project, you will use 4 benchmarks among them.

A. ADPCM

‘Adaptive Differential Pulse-Code Modulation (a.k.a. ADPCM)’ is a variation of pulse code modulation algorithm, which is widely used in telecommunication systems.

B. LAME

‘LAME Ain’t an MP3 Encoder (a.k.a. LAME)’ is an open source encoder for encoding MP3 sound files.

C. TIFF2RGBA

It is an image converter. It receives various TIFF images and converts them into RGBA images.

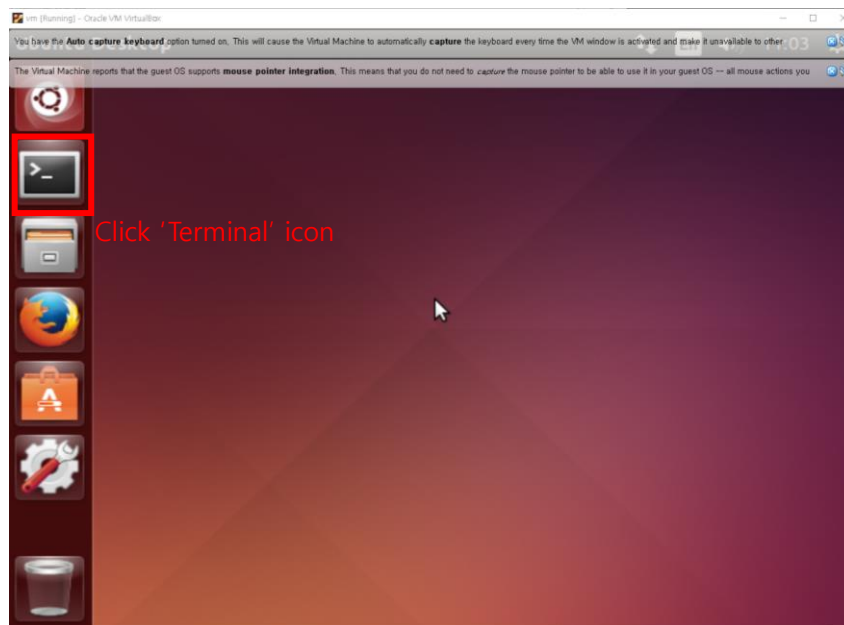
D. CRC32

It performs 32-bit ‘Cyclic Redundancy Check (a.k.a. CRC)’ to input files. CRC is usually used to detect errors in files or data packets.

If you want to know more about ‘MiBench’, please read <MiBench.pdf> uploaded on KLMS.

## VII. Execution example

- ✓ After setting simulation environment with < [linux\\_image\\_setup\\_guide.pdf](#) >, you may be able to use Linux Ubuntu 14.04 operating system on the virtual machine.  
In the Linux image, *there is pre-installed 'SimpleScalar-ARM' with 'MiBench'*.
- ✓ As following this execution example, you can simulate benchmarks and get 4 result files.
  - A. If you follow < [linux\\_image\\_setup\\_guide.pdf](#) > appropriately, you may see a screen as below.
  - B. Click left 'Terminal' icon or press 'Ctrl' + 'Alt' + 't' to open new terminal.



- C. New terminal will be open. If there are not any problems, it shows the followed sentence.

```
cs311@cs311:~$
```

It means that your user name is 'cs311' and now you are in a home directory, '~'.

- D. Next to the dollar mark, '\$', *write the followed command*.

```
cs311@cs311:~$ cd simplescalar-arm
```

'cd' is a command to change the current directory. 'simplescalar-arm' is a directory name, which has pre-installed 'SimpleScalar-ARM' simulator.

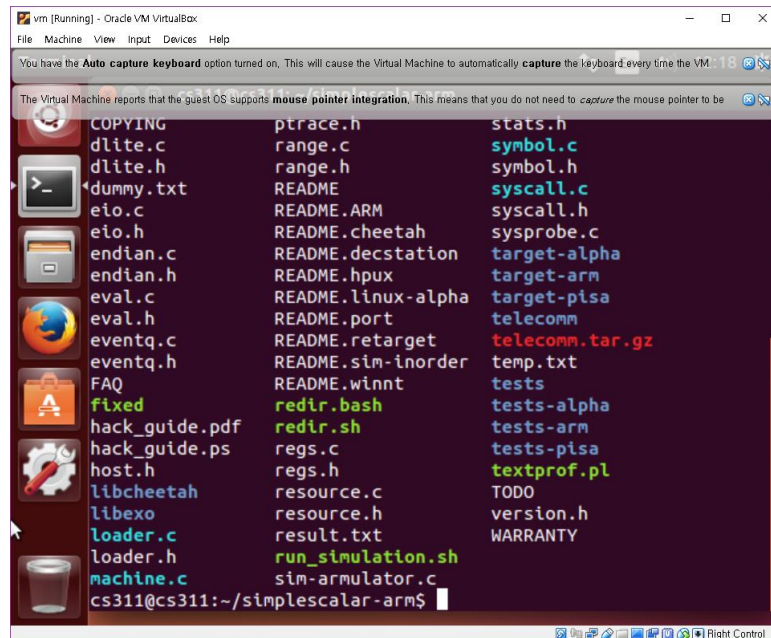
*After writing the command, press 'Enter'*. Then, the terminal shows new sentence as follows.

```
cs311@cs311:~/simplescalar-arm$
```

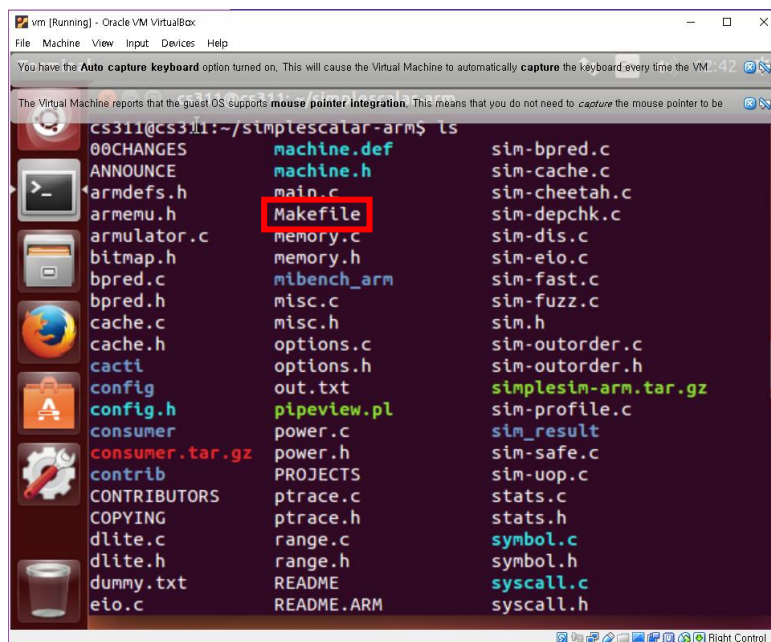
E. Write the command and press 'Enter'.

```
cs311@cs311:~/simplescalar-arm$ ls
```

'ls' shows the components list of the current directory. The terminal shows many files and directories like the next screen capture.



F. With a scroll, you should find a file named 'Makefile'.

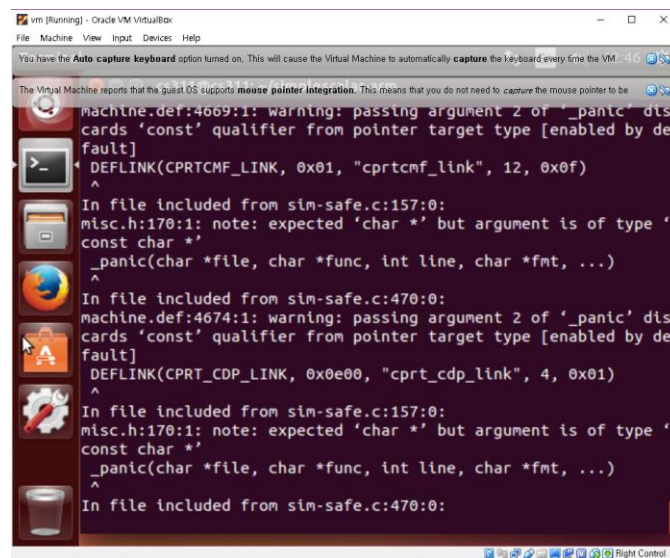


- G. In fact, the current 'SimpleScalar-ARM' is just in the state of several source codes. You must compile and build them to simulate. This process is quite complex, but 'Makefile' can help you. *You can compile and build 'SimpleScalar-ARM' just by writing the command below.*

```
cs311@cs311:~/simplescalar-arm$ make
```

'make' calls 'Makefile' and make executable simulation files. In the 'Makefile' there are several complex commands to compile and build the simulator. You do not need to know all these commands.

When you write 'make' command and press 'Enter', the terminal shows continuous complex explanations as follows.

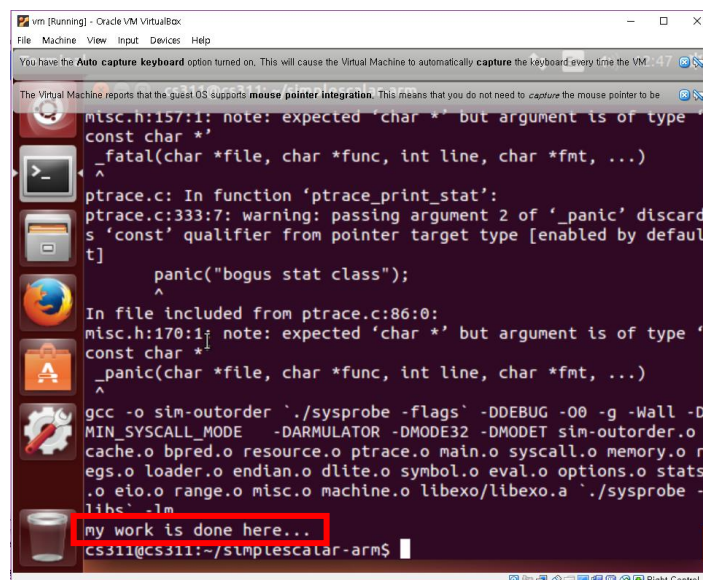


```

The Virtual Machine reports that the guest OS supports mouse pointer integration. This means that you do not need to capture the mouse pointer to be
Machine.def:4669:1: warning: passing argument 2 of '_panic' discards 'const' qualifier from pointer target type [enabled by default]
^
DEFLINK(CPRTCMF_LINK, 0x01, "cprtcmf_link", 12, 0x0f)
^
In file included from sim-safe.c:157:0:
misc.h:170:1: note: expected 'char *' but argument is of type 'const char *'
_panic(char *file, char *func, int line, char *fmt, ...)
^
In file included from sim-safe.c:470:0:
machine.def:4674:1: warning: passing argument 2 of '_panic' discards 'const' qualifier from pointer target type [enabled by default]
DEFLINK(CPRT_CDP_LINK, 0x0e00, "cpirt_cdp_link", 4, 0x01)
^
In file included from sim-safe.c:157:0:
misc.h:170:1: note: expected 'char *' but argument is of type 'const char *'
_panic(char *file, char *func, int line, char *fmt, ...)
^
In file included from sim-safe.c:470:0:

```

Do not worry about them. It means the simulator is being compiled and built. There are some warnings, but it does not affect to your simulator. *Compiling and building take some times.* Please be patient, and when it finishes, the terminal shows as follows. *Please check 'my work is done here...' sentence.*



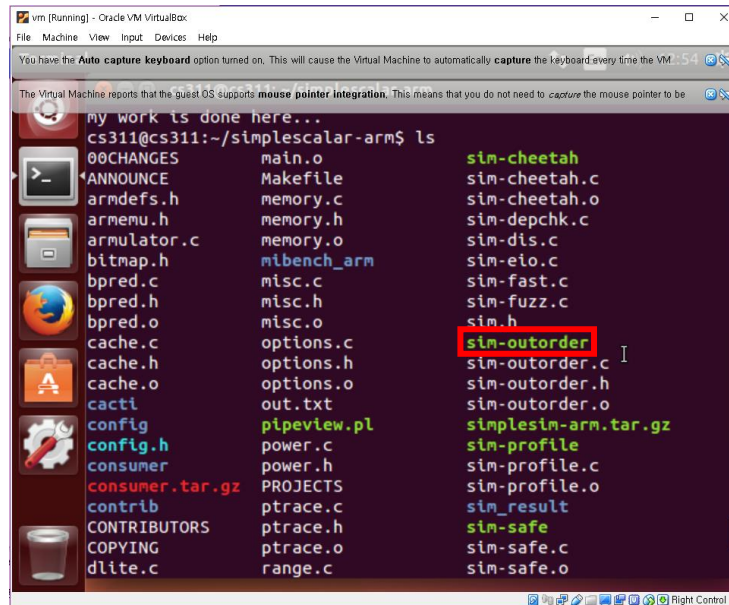
```

The Virtual Machine reports that the guest OS supports mouse pointer integration. This means that you do not need to capture the mouse pointer to be
misc.h:157:1: note: expected 'char *' but argument is of type 'const char *'
_fatal(char *file, char *func, int line, char *fmt, ...)
^
ptrace.c: In function 'ptrace_print_stat':
ptrace.c:333:7: warning: passing argument 2 of '_panic' discards 'const' qualifier from pointer target type [enabled by default]
panic("bogus stat class");
^
In file included from ptrace.c:86:0:
misc.h:170:1: note: expected 'char *' but argument is of type 'const char *'
_panic(char *file, char *func, int line, char *fmt, ...)
^
gcc -o sim-outorder ./sysprobe -flags -DDEBUG -O0 -g -Wall -D MIN_SYSCALL_MODE -DARMULATOR -DMODE32 -DMODET sim-outorder.o
cache.o bpre.o resource.o ptrace.o main.o syscall.o memory.o regs.o loader.o endian.o dlite.o symbol.o eval.o options.o stats.o
eio.o range.o misc.o machine.o libexio/libexio.a ./sysprobe -libs -lm
my work is done here...
cs311@cs311:~/simplescalar-arm$

```



- H. If you use 'ls' command again, you will see new several executable files. Executable files are shown as light green. In project #1, you will use 'sim-outorder' executable file for simulation. Please check 'sim-outorder'.



- I. Before starting simulation for project #1, you should test 'sim-outorder' whether it works well or not. Write the command below, and press 'Enter' to test. The command is quite long, so you can easily make some errors. Please write carefully, and do not just copy and paste. Copying sentences from PDF file can make some duplicate words.

```
cs311@cs311:~/simplescalar-arm$ ./sim-outorder -redir:sim adpcm_result.txt \
-max:inst 10000000 mibench_arm/adpcm/rawcaudio.arm \
< mibench_arm/adpcm/large.pcm > dummy.txt
```

Be careful about several dots (.), bars (-), and under-bars (\_). There are 7 zeros. With Korean keyboard, you can write back-slash (\) as 'won (₩)' key. When you press 'Enter' after writing back-slash, the terminal automatically makes inequality sign (>). It just means that the command is not finished yet. Ignore it, and write the next command line.

After writing all commands, you should have the following.

```

cs311@cs311:~/simplescalar-arm$ ./sim-outorder -redir:sim adpcm_result.txt \
> -max:inst 10000000 mibench_arm/adpcm/rawaudio.arm \
> < mibench_arm/adpcm/large.pcm > dummy.txt

```

- J. If you press 'Enter', the cursor moves the next line and nothing is occurred. *Do not do anything, and wait until 'cs311@...' line is shown again.* During this time, 'SimpleScalar-ARM' simulates with 'MiBench' based on the command. If you do something during that time, a simulation result will be wrong.
- K. After testing simulation, write the command to check the simulation result.

```
cs311@cs311:~/simplescalar-arm$ vim adpcm_result.txt
```

'vim' is one of famous text editors like 'notepad' in Windows. You can see contents of 'adpcm\_result.txt' text file as follows.

```

sim-outorder: SimpleScalar/ARM Tool Set version 3.0 of November
, 2000.
Copyright (c) 1994-2000 by Todd M. Austin. All Rights Reserved
.
This version of SimpleScalar is licensed for academic non-comme
rcial use only.

sim: command line: ./sim-outorder -redir:sim adpcm_result.txt -
max:inst 10000000 mibench_arm/adpcm/rawaudio.arm

sim: simulation started @ Wed Sep 5 13:24:28 2018, options fol
low:

sim-outorder: This simulator implements a very detailed out-of-
order issue
superscalar processor with a two-level memory system and specul
ative
execution support. This simulator is a performance simulator,
tracking the
latency of all pipeline operations.

# -config                                # load configuration from a file
"adpcm_result.txt" 259L, 16273C          1,1          Top

```

The first several lines are explanation of simulator configuration. You can scroll down and find the sentence, 'sim: \*\* starting performance simulation \*\*'.



```

Or, a fully unified cache hierarchy (il1 pointed at dl1):
-cache:il1 dl1
-cache:dl1 ul1:256:32:1:l -cache:dl2 ul2:1024:64:2:l

sim: ** starting performance simulation **

sim: ** simulation statistics **
sim_num_insn             10000001 # total number of instructions committed
sim_num_uops              11469106 # total number of UOPs executed
sim_avg_flowlen           1.1469 # uops per instruction
sim_num_refs              1370935 # total number of loads and stores committed
sim_num_loads              1277030 # total number of loads committed
sim_num_stores             93905.0000 # total number of stores committed

131,0-1 51%

```

After that line, it shows performance simulation results. *Results are composed of three parts.*

Simulation parameter	(Blank)	Simulation result	#	Explanation
----------------------	---------	-------------------	---	-------------

- L. You can use these simulation results to analyze and write a report. After confirming simulation results, *you can exit 'vim' editor with the command.*

```
:q
```

When you write the command, it shows the last line of the terminal. If you press 'Enter', 'vim' editor will be closed and the terminal will be back to the state before starting 'vim' editor.

- M. If you confirm the simulation result, it means there are not any problems on the simulator. Now, you can simulate with selected 4 benchmarks. *There are two ways to simulate these benchmarks.*

*✂ Real simulations needs much longer time than the testing simulation done before. Please confirm again whether you can simulate now or not.*

1. Simulate each benchmarks one by one with commands:  
You can simulate with following commands. *It is little bit different to testing commands, so please write them carefully.*

```

<Command for 'ADPCM' benchmark simulation>
cs311@cs311:~/simplescalar-arm$ ./sim-outorder -redir:sim sim_result/adpcm_result.txt \
-config config/test.cfg -max:inst 10000000 mibench_arm/adpcm/rawaudio.arm \
< mibench_arm/adpcm/large.adpcm > dummy.txt

```

```

<Command for 'LAME' benchmark simulation>
cs311@cs311:~/simplescalar-arm$ ./sim-outorder -redir:sim sim_result/lame_result.txt \
-config config/test.cfg -max:inst 10000000 mibench_arm/lame/lame.arm \
mibench_arm/lame/large.wav > dummy.txt

```

<Command for 'TIFF2RGBA' benchmark simulation>

```
cs311@cs311:~/simplescalar-arm$ ./sim-outorder -redir:sim sim_result/tiff2rgba_result.txt \
-c none ./mibench_arm/tiff2rgba/large.tif \
./mibench_arm/tiff2rgba/output_largergba.tif > dummy.txt
```

<Command for 'CRC32' benchmark simulation>

```
cs311@cs311:~/simplescalar-arm$ ./sim-outorder -redir:sim sim_result/crc32_result.txt \
-c none ./mibench_arm/crc/crc.arm \
./mibench_arm/adpcm/large.pcm > dummy.txt
```

2. Simulate four benchmarks at once with a shell script:

If you want to simulate four benchmarks at once, you can use a shell script, 'run\_simulation.sh'. A shell script is a file which has a set of commands. *You can use same commands with only calling the shell script as follows.*

```
cs311@cs311:~/simplescalar-arm$ ./run_simulation.sh
```

It is better because there are less possibility to make errors, but you must wait much longer since it simulates four benchmarks at once.

Even though it simulates at once, simulation results are not affected. *There are not any differences in results, so do not worry about using it.*

- N. After the end of simulation, the result files are stored in 'sim\_result' directory.

*You can go to 'sim\_result' directory, and confirm results with 'vim' editor.*

There will be four simulation results named:

< **adpcm\_result.txt**, **lame\_result.txt**, **tiff2rgba\_result.txt**, **crc32\_result.txt** >.

```
cs311@cs311:~/simplescalar-arm$ cd sim_result
cs311@cs311:~/simplescalar-arm/sim_result$ vim OO.txt
<Write each result file names instead of OO.txt>
```

### VIII. Tips

- ✓ Please read all this guideline carefully. Most of you can do project #1 without any difficulties if you read it carefully.
- ✓ There are some **red-highlighted words** and *italic sentences* in this guideline. They are very important information for your project #1. Do not ignore these highlighted contents.
- ✓ If you have some questions, please contact to the charge TA. You can use both an e-mail and a KLMS QnA board. Both English and Korean can be used for a question.
- ✓ TA also can help you in office hour if you bring a notebook. It can be much helpful than sending an e-mail or questioning on KLMS. However, TA will help only setting simulation environment. Simulation and analysis must be done by yourself.
- ✓ From 22<sup>nd</sup>, Sept. to 26<sup>th</sup>, Sept., there will be a long holiday. TA can be hard to reply fast during this holiday. TA will try to reply on time, but please understand that there can be some delays.
- ✓ Project will continuously use 'SimpleScalar-ARM' and 'MiBench'. If you fail project #1, it will affect to project #2 and #3. Project #1 is not quite difficult, so please do not give up!