Assignment 1: Branch Predictor on MIPS Simulator

Due date: 12/6 24:00

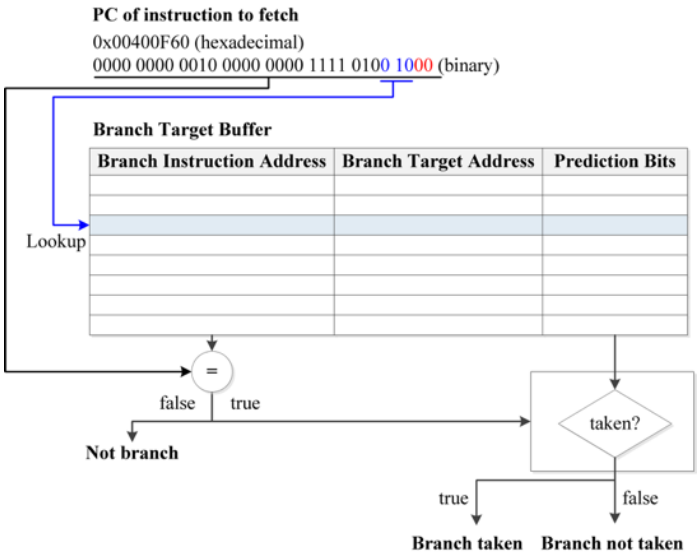
Submission: iCampus (Report), Homework Server (Source Code)

In this homework, you will implement a branch predictor on MIPS pipeline simulator. The target system has to handle the following characteristics:

- **Branch Predictors Running on MIPS Pipeline Simulator**
- **Two Dynamic Branch Predictors**: 1-bit and 2-bit
- **Branch Target Buffer** (Variable Size: 8, 16, 32, 64, 128 entries)

You will be given *a MIPS pipeline simulator with a static branch predictor (Always Not Taken)*. **You should implement two dynamic branch predictors (1-bit and 2-bit) on the simulator.** The dynamic branch predictors should have **a branch target buffer(BTB)**. The branch target buffer is a table, each entry of which is composed of 3 fields; branch instruction address, branch target address, and branch prediction bits.

Assume that the word size of the target system is 32 bits. Since MIPS instruction's addresses are aligned to 4 bytes, the branch target buffer's index should be the least significant bits of branch instruction's address, excluding lower two bits. For example, if target system has a 2-bit branch predictor with branch target buffer of 8 entries, the branch target buffer would work as shown in the following figure. Additionally, the total size of branch target buffer in the example would be 66 bytes. (8 entries * (32 bits + 32 bits + 2 bits) = 528 bits = 66 bytes)



The input to MIPS simulator is a sequence of MIPS instructions. For each instruction, your simulator would run 5 stages of MIPS pipeline. In IF stage, the simulator would predict 'whether branch is taken or not' by looking up the branch target buffer. If the branch is predicted as taken, the control must jump to the predicted branch target address, referring to the branch target buffer. When executing a branch instruction in EXE stage, the

entry of branch target buffer can be updated.

The program should keep track of branch prediction for each combination of branch predictor's type and size (the number of branch target buffer entries).

This assignment covers branch target buffer, not branch history table(BHT). Please be not confused.

You are provided with two program code files written in MIPS instructions (program1.s, program2.s). I also provide a short program code for your test (test.s, loop.s).

The simulator should print the misprediction ratio for the given branch predictor type and branch target buffer's size on the screen(stdout). **Based on the output, your report should include the following tables** for 2 types of branch predictors and various branch target buffer size. Each entry in the table contains the branch misprediction ratio for each combination of branch predictor type and branch target buffer's size. The tables should contain the results for different input programs. Output tables are written in the following format:

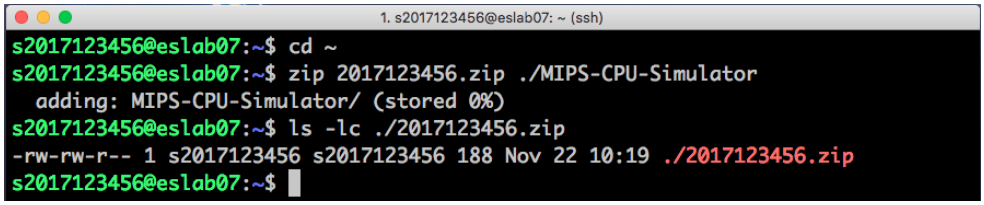
program1.s					
BTB Size (Entries)	**8**	**16**	**32**	**64**	**128**
1-bit					
2-bit					

program2.s					
BTB Size (Entries)	**8**	**16**	**32**	**64**	**128**
1-bit					
2-bit					

Your simulator should be one executable file which can get input arguments of branch predictor's type and size. (Don't make multiple programs each of which can run only one configuration)

What to Submit (Deadline: 12/6 24:00)

- **The report** summarizing the results for the provided two test files. The report file must be a **pdf** file.
  - **How to Submit**: Rename the report into (***yourstudentid.pdf)*** and upload it at iCampus.
  - Explain on how you implemented the simulator. (Descriptions on main data structures and algorithms)
  - Include **the output tables** that were printed from the simulator.
  - If the report file cannot be opened with pdf reader, I will not give any points.
- **The source code** of simulator
  - **How to Submit**: Make an archive file(***yourstudentid.zip***) of your source code and leave it on **your home folder of homework server**. You don't need to submit the code to iCampus.
  - I will regard the **modified time(ctime)** of *yourstudentid.zip* as the submit time.



  - The simulator should be written in C++.
  - You should write your C++ program at **Homework Server**.
  - For all the program blocks (functions, loops, if-then-else, etc) and the variables that you implemented, you should add proper comments. If sufficient comments are not provided, I will not give any points. You don't need to add comments on the given source code of MIPS pipeline simulator, which you did not implement.
  - The output of simulator is expected to be **the single value of misprediction ratio** for the given branch predictor type and branch target buffer's size.

◈ Notice

- **If I would find two or more source codes which are much alike, all the corresponding students will fail this course. At the Homework server, we can monitor all your operations such file copy and logging IP. So, we can produce an evidence of your cheating.**
- **If your program satisfies only a subset of the specified requirements, you can get a partial point. To get a partial point, specify the list of complete functions of your program.**
- **Scoring: 1-bit branch predictor is 40 points, 2-bit branch predictor is 60 points.**
- **One day delay has -20 points penalty.**
- **If you have any questions, send email to or visit TA (redcarrottt@gmail.com, room# 85465).**

Development System (Homework Server)

For the fairness, you must use our homework server. All the commands on the homework server will be written in a log file.

Don't upload the final source code after you edit it at your PC. You have to use the Linux editor (vi) in Homework Server. We will check the log file.

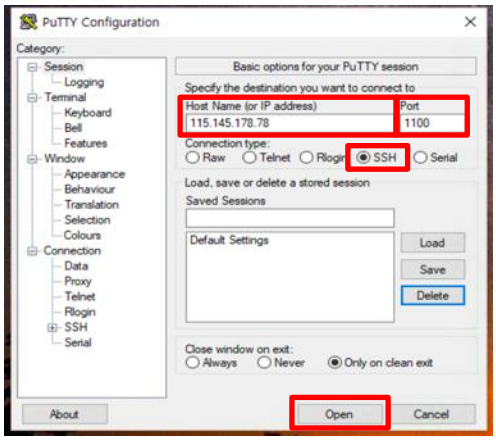
<How to Connect to Homework Server>

You have to use a SSH(Secure SHell) client program to connect to the homework server. You can use any kind of SSH client. Recommended SSH clients are:

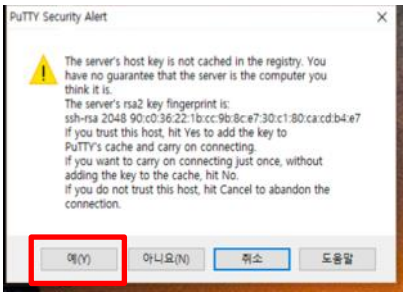
- For Windows, recommended SSH client is putty.
  - Download Link: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
- For macOS or Linux, use "ssh" command on a shell.

You can connect to the homework server as following:

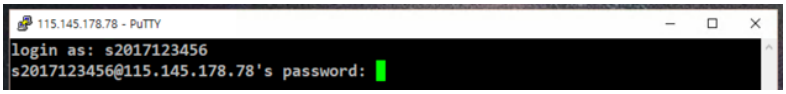
- **In case of putty (Windows):**
  1. Input IP address (115.145.178.78) and port number (1100), then press "Open" button.



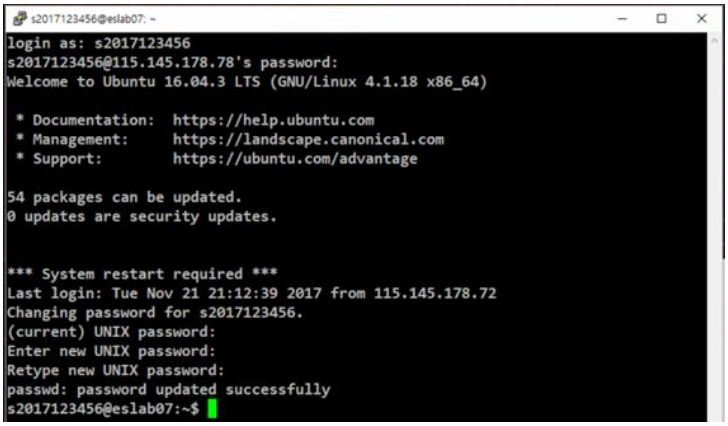
  2. In the case that a security alert appears, press "Yes" or "예" button to allow the connection.



  3. Input your ID and password. ID is a string combined with 's' and your student ID (ex. s2017123456). Initial password is same as your ID.



4. If it is first connection, you have to change your password.

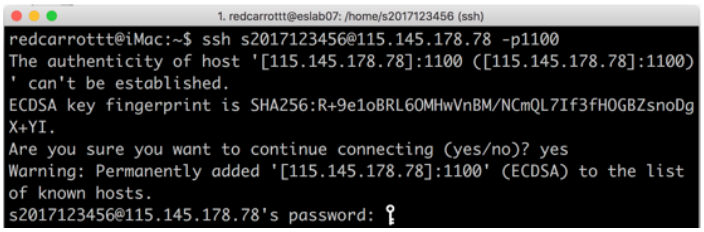


- **In case of "ssh" command (macOS, Linux):**

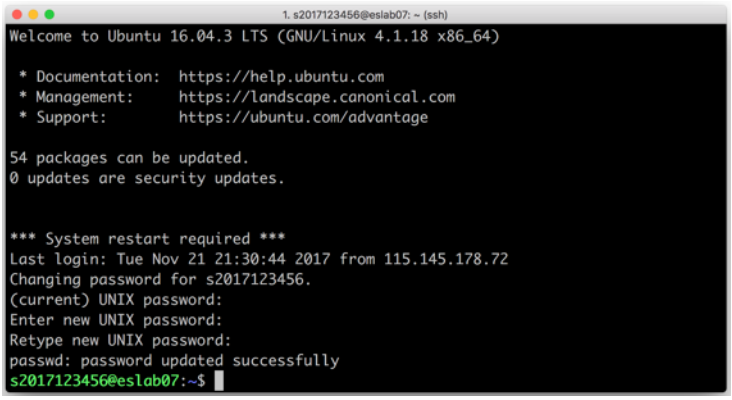
1. Run "ssh" command with three arguments (ID, IP address and port number) on a shell. ID is a string combined with 's' and your student ID (ex. s2017123456).

2. In the case that a security alert appears, input "yes" to allow the connection.

3. Input your password. Initial password is same as your ID.

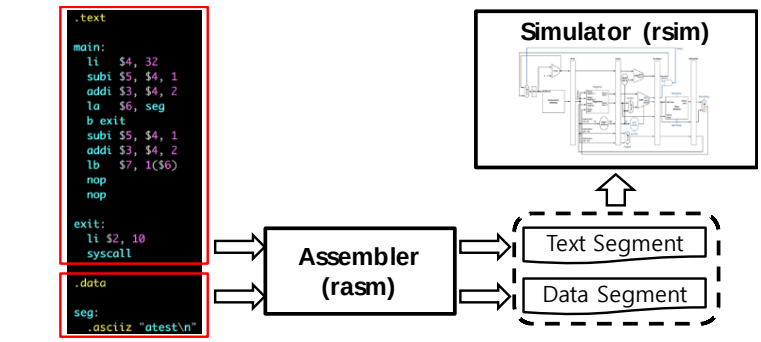


4. If it is first connection, you have to change your password.



MIPS Pipeline Simulator

I will give you the MIPS Pipeline Simulator that runs on a Linux machine. It simulates the execution of a subset of 32-bit 5-stage MIPS CPU pipeline described in your textbook; "Computer Organization and Design (COD)" by Patterson & Hennessy. It is composed of two binaries; a MIPS assembler(rasm) and a MIPS simulator(rsim).



MIPS Assembly (program1.s)

<Source Code of MIPS Pipeline Simulator>

I will give you following code in "`MIPS-CPU-Simulator/sim`" directory. You'd better to implement your branch predictor by modifying "`stages.cc`", "`stages.h`", and "`instruction.h`".

- `instruction.h`
  - Data structure of MIPS instructions that can run in this simulator
- `stages.cc, stages.h`
  - Logic of each MIPS pipeline stage
  - There are 5 functions running 5 stages of MIPS pipeline. You can implement your branch predictor by modifying these functions.
    - `void InstructionFetchStage::Execute()`
    - `void InstructionDecodeStage::Execute()`
    - `void ExecuteStage::Execute()`
    - `void MemoryStage::Execute()`
    - `void WriteBackStage::Execute()`
  - If you want to implement jump (changing program counter), use given `jump_to()` function.

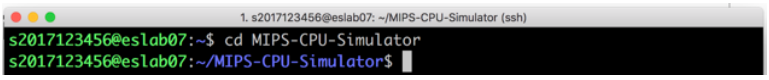
```
static void inline jump_to(uint32_t* pc, uint32_t immediate) {
  if(immediate < 0x00400000)
    *pc = 0x00400000 + immediate; // load the target into the PC.
  else
    *pc = immediate;
}
```

- `simulator.cc`
  - Main function
- `cpu.cc, cpu.h`
  - Main logic of CPU simulation

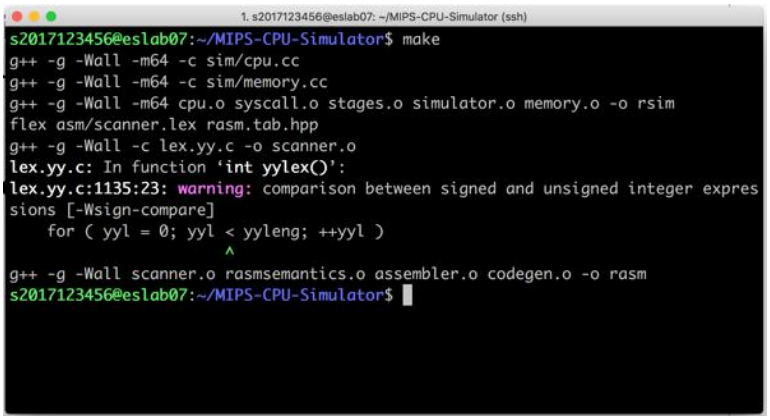
- `memory.cc, memory.h`
  - Main logic of memory simulation (code segment, data segment)
- `syscall.cc, syscall.h`
  - Main logic of system call simulation

<How to Build MIPS Pipeline Simulator>

1. After connecting to the homework server, navigate to the directory of MIPS simulator.

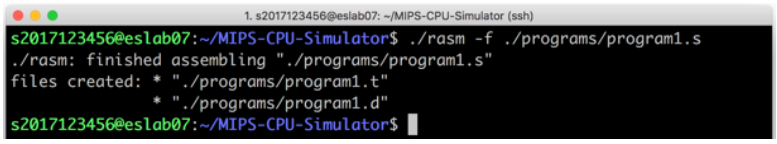


2. Run "`make`" command.



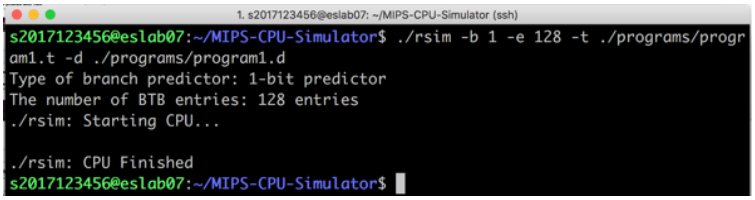
<How to Use MIPS Pipeline Simulator>

1. Run "`rasm`" command (MIPS Assembler) with a parameter; MIPS assembly code file path (-f option). This command convert MIPS assembly code into two MIPS binary files; text segment (*.t) and data segment (*.d).

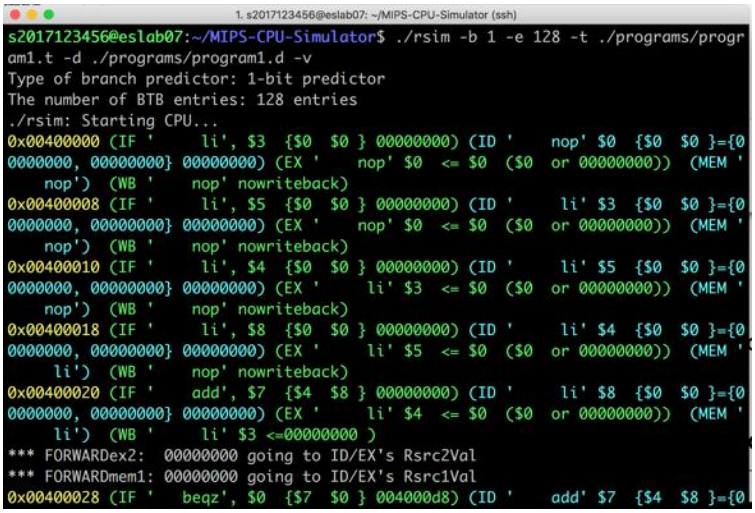


2. Run "`rsim`" command (MIPS Simulator) with parameters; branch predictor's type (-b option), branch predictor's size (-e option), text segment file path (-t option), and data segment file path (-d option). This command executes MIPS binary files on simulated MIPS pipeline.

   Caution: You have to print the single value of misprediction ratio for the given branch predictor type and branch target buffer's size.



3. If you want to see more detailed pipeline behavior, run the "`rsim`" command with "-v" option.



<How to Recover Original Code of MIPS Pipeline Simulator>

If you have trouble in doing your homework and want to recover the original code, follow the below commands:

Caution: If you input these commands, you will lose your homework source code. Please backup them before doing it.