SWE3005

SUNG KYUN KWAN UNIVERSITY

Theory on Computer Architectures (Fall 2019)

# PA2: Pipeline & Branch Prediction

Prof. Jinkyu Jeong (jinkyu@skku.edu)

TA – Minwoo Ahn (minwoo.ahn@csi.skku.edu)

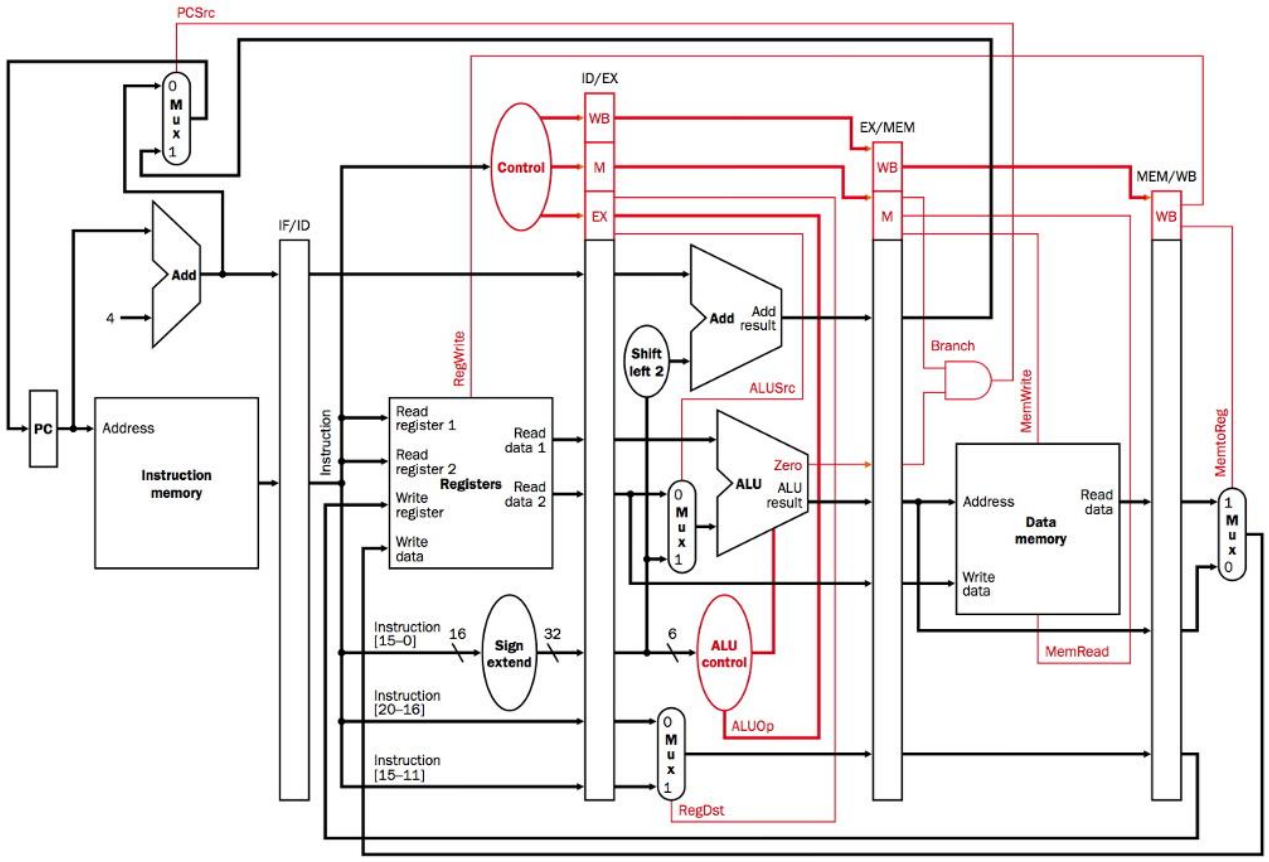
TA – Sunghwan Kim (sunghwan.kim@csi.skku.edu)

Computers Systems and Intelligence Laboratory (http://csi.skku.edu)

2019. 11. 07.

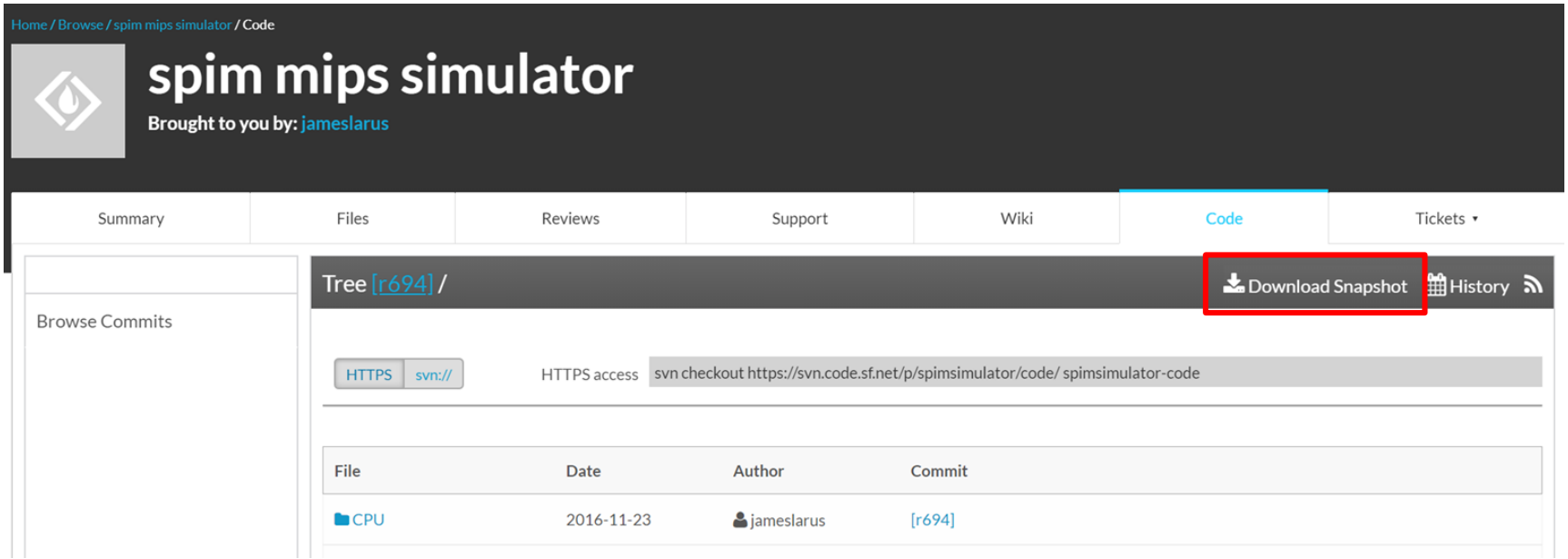
# Overview: What to do?

- Implement pipeline execution and branch prediction on SPIM simulator



# Overview: Environment

- Linux Ubuntu
  - VirtualBox is already installed in Workstation Lab. #400212, #400222, Semiconductor Bld.
- SPIM simulator (r694)
  - https://sourceforge.net/p/spimsimulator/code/694/tree/



# Overview: Environment (PL)

- Use the 5-stage (IF, ID, EX, MEM, and WB) in-order single issue MIPS pipelined architecture described in the textbook.
- Assume no cache miss, i.e., the latency of each instruction is 5 cycles if there are no pipeline stalls by hazard.
- Arithmetic and logical instructions determine the value of destination register in the EX stage.
- Load and store instructions determine the value of destination register in the MEM stage.
- The decision of branch (direction/target) and jump (target address) are determined in the ID stage.

# Overview: Environment (PL)

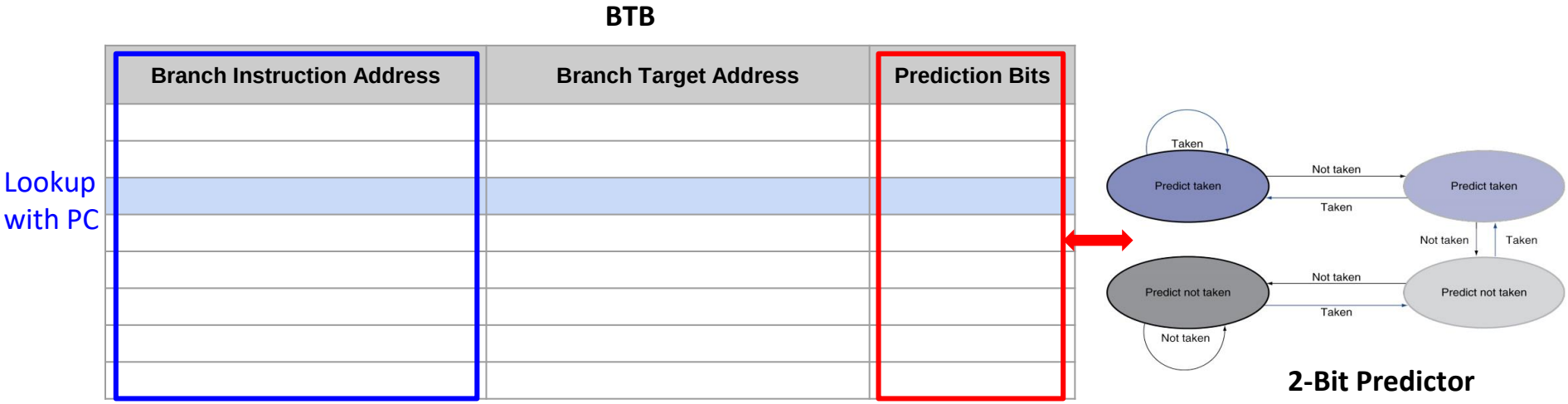
- Use the ‘static not-taken’ policy for branch prediction.
- The pipelined version must detect data and control hazards and must calculate the required execution cycles considering the forwarding and pipeline stalls.
- Treat the syscall as nop. Even if the last instruction does not update any register at the WB stage (e.g. syscall, sw, b, and j), the program can be complete after the WB stage of the instruction.
- Don’t count the cycles consumed before entering the main function.
- Process only the core instructions.
  - ADD, ADDI, SUB, AND, ANDI, NOR, OR, ORI, LUI, SLT, SLTI, SLL, SRL, LW, SW, BEQ, BNE, J, JAL, JR, SYSCALL

# Implementation of Pipeline

- When running the test code by spim, the total number of cycles of pipelined architecture, data hazard, data forwarding, stall by data hazard and stall by control hazard (branch and jump) should be output.
- Currently, spim processes 1 instruction per 1 cycle.
- # of cycle = 4 + # of Instruction + # of Stall by hazard
- You have to implement detection of hazards.
- Data forwarding does not affect # of cycles.
- In ‘static not-taken’ policy, when branch taken, it need to add 1 cycle to stall because of control hazard.
- jump instruction (j, jal, jr) must need to add 1 cycle to stall.

# Overview: Environment (BP)

- 2-bit dynamic branch predictor on your pipelined spim.
- Output format is same as pipelined spim.
- The dynamic branch predictors should have a branch target buffer(BTB).




- BTB has 8 entries.
  - 8 entries * (32 bits + 32 bits + 2 bits)

# Overview: Given Files for Modify

- CPU/pipeline.h
  - printResult()
- CPU/pipeline.c
  - printResult()
- CPU/run.c (You have to modify this file!)
  - When you submit
    1. Pipeline Implementation: change the file name run.c to run_pl.c
    2. Branch Prediction Implementation: change the file name run.c to run_bp.c
- spim/*.s
  - 5 test cases (df.s, lw.s, beq.s, lw_b.s, test_bp.s)
  - You can test with following command
    - $ ./spim -f [filename].s

# How to build SPIM

1. Download SPIM and unzip.

```
sung@ubuntu:~/Downloads$ unzip spimsimulator-code-r694.zip
```

2. Add pipeline.c & pipeline.h & run.c files in *CPU* directory
3. Go to the *spim* directory
4. Add input files (*.s files) and Modify the “Makefile”

```
126 OBJS = spim.o spim-utils.o run.o mem.o inst.o data.o sym-tbl.o parser_yacc.o lex.yy.o pipeline.o \
127        syscall.o display-utils.o string-stream.o
```

```
269 run.o: $(CPU_DIR)/syscall.h
270 run.o: $(CPU_DIR)/pipeline.h
271 run.o: $(CPU_DIR)/run.h
```

```
71 # Full path for the directory that will hold the exception handler:
72 EXCEPTION_DIR = ../CPU
```

5. Run “make” command

```
sung@ubuntu:~/Downloads/spimsimulator-code-r694/spim$ make
```

6. When rebuild the SPIM, run “make clean” command before run “make” command

# Example (1)

- df.s (Detection of data forwarding)

```
        .text
main:
        add $t0, $zero, $zero
        addi $t1, $t0, 3
        sub $t2, $t1, $t0
        addi $v0, $zero, 10
        syscall         # exit()
```

# Example (1)

- Result
  - Pipeline (static not-taken)

```
sung@ubuntu:~/Downloads/spimsimulator-code-r694s/spim$ ./spim -f df.s
Loaded: /usr/share/spim/exceptions.s
Number of Cycle : 9
Number of Data Hazard : 2
Number of Data Forwarding : 2
Number of Stall by Data Hazard : 0
Number of Stall by Branch (Jump) : 0
```

# Example (2)

- lw.s (Detection of load data stall)

```
        .text

main:
        lui     $3, 0x1000
        lw      $2, 0($3)
        and     $4, $2, $5
        or      $8, $2, $6
        add     $9, $4, $2
        slt     $3, $6, $7
        addi    $v0, $zero, 10
        syscall         # exit()


.data 0x10000000
        .word   1, 2, 0
```

# Example (2)

- Result
  - Pipeline (static not-taken)

```
sung@ubuntu:~/Downloads/spimsimulator-code-r694s/spim$ ./spim -f lw.s
Loaded: /usr/share/spim/exceptions.s
Number of Cycle : 13
Number of Data Hazard : 3
Number of Data Forwarding : 2
Number of Stall by Data Hazard : 1
Number of Stall by Branch (Jump) : 0
```

# Example (3)

- beq.s (Detection of control hazard)

```
        .text
main:
        add $t0, $zero, $zero
        add $t1, $zero, $zero
        beq $t0, $t1, B
        add $t2, $t0, $t1
B:      addi  $v0, $zero, 10
        syscall          # exit()
```

# Example (3)

- Result
  - Pipeline (static not-taken)

```
sung@ubuntu:~/Downloads/spimsimulator-code-r694s/spim$ ./spim -f beq.s
Loaded: /usr/share/spim/exceptions.s
Number of Cycle : 11
Number of Data Hazard : 1
Number of Data Forwarding : 0
Number of Stall by Data Hazard : 1
Number of Stall by Branch (Jump) : 1
```

# Example (4)

- lw_b.s (Detection of load data stall and control hazard)

```
        .text

main:
        lui     $3, 0x1000
        lw      $2, 0($3)
        bne     $2, $3, B
        add     $3, $0, $0
B:
        and     $4, $2, $5
        or      $8, $2, $6
        add     $9, $4, $2
        slt     $3, $6, $7
        addi    $v0, $zero, 10
        syscall         # exit()


.data 0x10000000
        .word   1, 2, 0
```

# Example (4)

- Result
  - Pipeline (static not-taken)

```
sung@ubuntu:~/Downloads/spimsimulator-code-r694s/spim$ ./spim -f lw_b.s
Loaded: /usr/share/spim/exceptions.s
Number of Cycle : 16
Number of Data Hazard : 3
Number of Data Forwarding : 2
Number of Stall by Data Hazard : 2
Number of Stall by Branch (Jump) : 1
```

# Example (5)

- test_bp.s (Branch prediction)

```
        .text
main:                       # execution starts here
        addi $t0, $0, 30    # i = 30
LOOP:   addi $t0, $t0, -1   # i --
        bne $t0, $0, LOOP   #if ( i != 0 )
        addi  $v0, $zero, 10
        syscall             # exit()
```

# Example (5)

- Result
  - Only pipeline (static not-taken)

```
sung@ubuntu:~/Downloads/spimsimulator-code-r694s/spim$ ./spim_pl -f test_bp.s
Loaded: /usr/share/spim/exceptions.s
Number of Cycle : 126
Number of Data Hazard : 31
Number of Data Forwarding : 1
Number of Stall by Data Hazard : 30
Number of Stall by Branch (Jump) : 29
```

  - 2-bit Branch Prediction

```
sung@ubuntu:~/Downloads/spimsimulator-code-r694s/spim$ ./spim_bp -f test_bp.s
Loaded: /usr/share/spim/exceptions.s
Number of Cycle : 99
Number of Data Hazard : 31
Number of Data Forwarding : 1
Number of Stall by Data Hazard : 30
Number of Stall by Branch (Jump) : 2
```

# Submission

- Compress your modified files only (run_pl.c & run_bp.c)
  - Do not change file name if it is modified
  - Without subdirectories
  - Submitted file name should be YourStudentID.zip
    - ex) 2018000000.zip
  - Please follow this format
- Upload your zip file to I-Campus *Assignments* bulletin
- DO NOT COPY
  - If you have any questions please use your TAs
- Due date:
  - -20% per day for delayed submission

# Questions

- You are free to ask questions to TAs
  - Please send email before visit the office
  - Email: minwoo.ahn@csi.skku.edu / sunghwan.kim@csi.skku.edu
  - Office: Semiconductor Building #400509