Computer Architecture

Fall, 2020

Week 14

2020.12.14

組別：__________ 簽名：____

[group3]

1. When handling Branch Hazard, if pipelines are deep, how can we solve the problem of significant branch penalty? How?

ANS:

By dynamic branch prediction, we can guess the branch base on history (branch prediction buffer, branch history table), if last time branch was

(1) Taken → predict taken

(2) Not taken → predict not taken

If the prediction is wrong → flush pipeline and flip prediction

[group14] (對抗賽)

2. Which statement are true?

(a) In deeper and superscalar pipelines, branch penalty is more significant.

(b) If the branch prediction is wrong, we will always flush pipeline and flip prediction.

(c) If the branch prediction is true and no using the approach of branch target buffer, there is no penalty for a taken branch.

(d) Interrupt only arises within the MIPS CPU.

(e) Longer clock cycle will help to increase ILP (Instruction-Level Parallelism).

Answer

(a)

(b) depends on the design of branch predictor

(c) still need 1-cycle penalty to calculate the target address

(d) Exception

(e) Shorter clock cycle

[group12] (對抗賽)

3. Given a MIPS instruction sequence shown below. Assume this code is executed on a pipelined MIPS CPU with a five-stage (IF, ID, EXE, MEM, WB) pipeline, full forwarding. In addition, assume this CPU can finish the register write in the first half cycle and the register reads in the second half cycle. If the prediction is wrong and the branch is not taken, please redraw the simple (traditional) multiple-clock-cycle pipeline execution diagram of the first 5 executed instructions, including all necessary stalls.

`L3: add $s6, $s6, $s5`										
`addi $s5, $s5, 2`										
`slti $t0, $s5, 10`										
`bne $t0, $zero, L3`										
`add $s6, $s6, $s5`										
`sw $s6, 100($gp)`										

Ans:

`L3: add $s6, $s6, $s5`	IF	ID	EX	ME	WB					
`addi $s5, $s5, 2`		IF	ID	EX	ME	WB				
`slti $t0, $s5, 10`			IF	ID	EX	ME	WB			
`bne $t0, $zero, L3`				IF	ID	ID	EX	ME	WB	
`add $s6, $s6, $s5`					IF					
`sw $s6, 100($gp)`						IF	ID	EX	ME	WB

[group9] (對抗賽)

4. True or False? If False, why?
    A. Handling branch do not need any hardware when it need to flush instruction if it predicts wrong.
    B. We can minimize the instruction that needs to flush to one instruction if we made branch decision at IF.
    C. In MIPS, interrupt managed by a System Control Coprocessor(CP0)
    D. Exceptions arise within the CPU.

Ans:

A. False, it needs hardware to flush instructions.

B. False, should made branch decision at ID stage.

C. False, exceptions managed by a System Control Coprocessor (CP0)

D. True.

[group6] (對抗賽)

5. Which statements are false, choose and correct it.

(a) By using dynamic prediction, we do not need to calculate the target address because we already have predictor.

(b) Unexpected events arise within the CPU called exception, from an external I/O controller called interrupt.

(c) Dynamic multiple issue use compiler to group instructions into issue slots.

(d) Dynamic multiple issue is the simplest version of superscalar processors.

(e) Pipelining is not easy because detecting data hazards is difficult.

A:

(a)need to calculate the target address.

(c)Static multiple issue use compiler to group instructions into issue slots.

[group13] (對抗賽)

6.

```
Begin:  add     $t1, $zero, $zero
        addi    $t0, $zero, 10
Loop:   beq     $t0, $zero, L1
        sub     $t1, $t1, $t0
        addi    $t0, $t0, -2
        j       Loop
L1:
```

(1) According to the above MIPS code, how many times should these instructions need to flush pipeline by using 1 bit Dynamic Branch Prediction? Assume the value of branch history table is taken.

(2) If predict branch always taken, how many times should these instructions need to flush pipeline?

Ans:

(1)2 times, one is when entering Loop, the other time is when entering to L1

(2)5 times, when $t0=10,8,6,4,2,beq will not taken. However, the predict is always taken.

[group7] (對抗賽)

7. Which of the following state about step of exception in a pipeline is true?

a. exception in pipeline is another form of control hazard
b. consider overflow on add in EX stage: add \$1, \$2, \$1. we should prevent \$1 from being clobbered and complete previous instruction.
c. after (b), we can transfer control to handler directly.
d. use much of the same hardware with mispredicted branch.

Ans: abd

c. we should flush related instruction and set Cause and EPC register before transfer control to handler

[group2] (對抗賽)

8. By dynamic pipeline scheduling, please write down whether there are some instructions that can be executed “out of order” to avoid stalls in MIPS code. If we don’t need this scheduling, please also explain why.

```
lw      $t0, 20($s2)
addu    $t1, $t0, $t2
add     $s3, $s3, $t3
slti    $t5, $s3, 20
```

Ans:

Yes, we can execute “add” while addu is waiting for lw.