Computer Architecture

Fall, 2026

Week 12

2026.11.23

組別：_____ 簽名：_____

[group 8]

1. What is a branch hazard in a pipelined processor, and how does branch prediction help reduce its performance impact?

Ans:

A branch hazard occurs when the processor does not yet know which instruction to execute next because the outcome of a branch instruction is uncertain. Branch prediction reduces delays by predicting whether the branch will be taken and continuing execution. If the prediction is incorrect, the processor flushes the incorrectly fetched instructions and resumes execution from the correct address.

[group 12]

2. 請問 Exception 跟 Interrupt 的觸發來源有什麼差異？

Ans:

Exception 是由 CPU 內部在執行指令時觸發（如溢位、未定義指令）；Interrupt 則是由 CPU 外部的 I/O controller 或硬體裝置所發出。

[group 8]

3. For each of the following statements, answer True or False.

   1. If a branch prediction is incorrect, the processor may need to flush incorrectly fetched instructions from the pipeline.
   2. Exceptions are always caused by external hardware devices, rather than events within the CPU.
   3. A superscalar processor can issue multiple instructions in a single clock cycle.

Ans:

1. True, 2. False, 3. True

[group 2]

4. what is the differences between branch from fall through and branch from target

Ans:

Fall-Through (Not Taken): When a conditional branch evaluates to false, execution continues sequentially to the very next instruction in memory (PC + 4, assuming 4-byte instructions). Because the program does not jump, it simply "falls through" to the subsequent line of code.
Branch Target (Taken): When a conditional branch evaluates to true—or when an unconditional jump occurs—execution transfers to a specific destination address encoded in the instruction or held in a register. The CPU redirects its Program Counter (PC) to this new target address to fetch the next block of code.

[group 7]

5. When an exception occurs in a MIPS processor, what operations are performed to handle the exception?

Ans:

Save PC of offending (or interrupted) instruction in the Exception Program Counter(EPC).
Save indication of the problem in the Cause register.
Jump to handler at 0x80000180.

[group 9]

6. When a branch decision is made in the MEM stage and the pipeline needs to flush instructions, which pipeline registers are flushed and how is it done?

Ans:

Instructions in IF/ID and ID/EX are flushed by changing their control values to 0.

[group 9]

7. When a branch is resolved in the ID stage, how does the hardware check whether two registers are equal without waiting for the main ALU in the EX stage?

Ans:

It compares the two register values read during ID using XOR logic.

[group 1]

8. Consider a MIPS 5-stage pipeline (IF → ID → EX → MEM → WB) that always predicts branches as not taken.
   The following instructions are executed:
   beq $t0, $t1, LABEL
   add $t2, $t3, $t4
   sub $t5, $t6, $t7
   LABEL: or $t8, $t9, $s0
   Assume $t0 = $t1, and the branch decision is made in the ID stage.
   (a) Is the branch prediction correct or incorrect? Explain why.
   (b) Which instruction needs to be flushed?
   (c) What is the purpose of the IF.Flush control signal?

Ans:

(a) Incorrect. Since $t0 = $t1, the branch is taken, but the processor predicts not taken.
(b) The add instruction must be flushed because it was fetched from the incorrect path.
(c) IF.Flush clears the instruction in the IF/ID pipeline register, converting it into a NOP to prevent incorrect execution.

[group 13]

9. Consider a 5-stage MIPS pipeline:
   IF → ID → EX → MEM → WB
   Assume:
   Branches are resolved in the EX stage.
   The processor uses Predict Not Taken.
   The branch is actually Taken.
   beq $t0, $t1, LABEL
   add $t2, $t3, $t4
   sub $t5, $t6, $t7
   LABEL:
   and $s0, $s1, $s2
   Questions:
   (a) How many instructions must be flushed?
   (b) What is the branch misprediction penalty in clock cycles?

Ans:

(a) 2 instructions (add and sub).
(b) 2 clock cycles, because the branch outcome is determined in the EX stage.

[group 14]

10. A branch prediction buffer stores the last outcome of each branch (taken / not taken). It predicts the same outcome next time, and if the prediction is wrong, it flips the prediction. Given a loop branch is taken 9 times, then not taken once (loop exit). The loop is executed twice in a row.
    (a) How many mispredictions happen during the second execution of the loop? Which iterations?
    (b) Why is this predictor not ideal for loops?

Ans:

(a) There are 2 mispredictions. The first iteration is wrong because the buffer still holds “not taken” from the previous loop exit. The last iteration (the exit) is wrong because the buffer holds “taken”.
(b) A loop branch is almost always taken, so it is mispredicted twice every time the loop runs (once on entry and once on exit)

[group 2]

11. In clock cycle 4, the ID stage detects an undefined opcode in some instruction. In clock cycle 5, a lw that was fetched before that instruction causes a memory exception in the MEM stage. Which exception should the processor handle, and why can’t it simply handle exceptions in the order they are detected?

Ans:

A: The lw exception, because lw is the older instruction. To keep exceptions precise, every instruction before the faulting one must complete and every instruction after it must be flushed. If the processor took the undefined-opcode exception first, the older lw would be flushed while a younger instruction appeared to have run past it, leaving an inconsistent state. The younger instruction is simply flushed; after the handler returns, it is fetched again and raises its own exception then. For this reason, many designs record exceptions as they are detected but only act on them when the instruction reaches the end of the pipeline, in program order.