Computer Architecture

Fall, 2024
Week 12
2024.11.18

組別：_____ 簽名：_____

[group 2]

1. Which of the following statements are correct?

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

(b) Exception only arises from the external I/O controller.

(c) In MIPS, exception managed by a System Control Coprocessor (CP0).

(d) By dynamic pipeline scheduling, there are some instructions that can be executed "out of order" to avoid stalls in MIPS code.

Ans:

(a) False. Shorter clock cycle.
(b) False. Exception arises within the MIPS CPU.
(c) True.
(d) True.

[group 6]

2. Choose the correct answers and also explain if it is false.

A. Branch prediction is more important when pipelines are longer.

B. In branch prediction, we don't need to calculate the target address of the branch.

C. Interrupts originate within the CPU, while exceptions are caused by external I/O controllers.

D. A superscalar processor tends to use compiler-scheduled code.

E. In dynamic multiple issue, it allows the CPU to execute instructions out of order to avoid stalls.

F. In MIPS, exceptions are managed by a System Control Coprocessor (CP0).

Ans:

A. True

B. False (Explanation: Even with prediction, we still need to calculate the target address of the branch.)

C. False. (Explanation: It should be the opposite.)

D. False. (Explanation: Different implementations of ISA have varying latencies and hazards.)

E. True

F. True

[group 1]

3.
(1) What is the difference between an "exception" and an "interrupt"?
(2) How does the processor handle instructions in the pipeline when an exception occurs?

Ans:

(1) An exception is an event generated internally by the CPU, such as an instruction error. An interrupt is a signal generated by an external I/O controller, prompting the processor to take appropriate action.
(2) When an exception occurs in the pipeline, the processor completes the execution of prior instructions, flushes the exception-causing instruction and subsequent instructions, sets the values of Cause and EPC registers, and transfers control to the exception handler.

[group 12]

4. When an overflow exception occurs during the execution of an add $1, $2, $1 instruction in the EX stage, arrange the following steps in the correct order to handle the exception:

A. Flush the add instruction and any subsequent instructions.
B. Complete instructions that were already in the pipeline before the overflow.
C. Prevent $1 from being overwritten by the faulty result.
D. Set the Cause and EPC registers to record the exception information.
E. Transfer control to the exception handler.

Ans:

1. **C** - Prevent $1 from being overwritten by the faulty result.
2. **B** - Complete instructions that were already in the pipeline before the overflow.
3. **A** - Flush the add instruction and any subsequent instructions.
   (1, 2, 3 are executed in the same cycle)
4. **D** - Set the Cause and EPC registers to record the exception information.
5. **E** - Transfer control to the exception handler.

[group 3]

5. 請問下列技術，哪些是主要由 compiler 負責、由 hardware 負責，或是兩者都需要參與：

A. Multiple issue

B. VLIW (Very Long Instruction Word)

C. Out-of-order execution

Ans:

A. Compiler 和 hardware 都需要參與（both）

B. 主要由 compiler 負責（software-based）

C. 主要由 hardware 負責（hardware-based）

[group 11]

6. What is the difference between static multiple issue and dynamic multiple issue? Fill in the blank.

	static multiple issue	**dynamic multiple issue**
Approach		
Hazard Handling		
Require Compiler Scheduling?(Y/N)		
Allow Out-of-order execution?(Y/N)		

Ans:

	static multiple issue	**dynamic multiple issue**
Approach	Compiler-based	Hardware-based
Hazard Handling	Compiler detects and avoids hazards	CPU handles hazards each cycle
Require Compiler Scheduling?(Y/N)	Y	Y
Allow Out-of-order execution?(Y/N)	N	Y

[group 5]

7. Dynamic Scheduling：Why not just let the compiler schedule code?

Ans:

1. Not all stalls are predicable. e.g., cache misses
2. Can't always schedule around branches. Branch outcome is dynamically determined
3. Different implementations of an ISA have different latencies and hazards.

[group 10]

8. Consider a simple branch prediction scenario where you have a **1-bit branch predictor** that predicts **taken (T)** or **not taken (F)** based on the previous branch result (flip if wrong). The initial prediction is set to **not taken (F)**.

For each of the following branch sequences, calculate how many times the predictor will be **incorrect**, resulting in a **pipeline flush**.

1. Sequence 1: T T F T F T F T T F
2. Sequence 2: T F T T F F T F T T

Ans:

1. Sequence 1: 8
2. Sequence 2: 7

[group 8]

9.
For the MIPS code below:

```
Loop:
        lw $t0, 8($s1)
        add $t0, $t0, $s2
        sw $t0, 12($s1)
        add $s3, $t0, $s2
        sub $s1, $s1, $s2
        bne $s1, $zero, Loop
```

Q: Use static multiple issue. There are two issue packets, “ALU/branch” and “Load/Store”. Schedule this and fill in the blank. Also, if we use dynamic multiple issue, would it be faster? Why or why not?

	ALU/branch	Load/Store	cycle
Loop			1
			2
			3
			4
			5

Ans:

	ALU/branch	Load/Store	cycle
Loop	nop	lw $t0, 8($s1)	1
	nop	nop	2
	add $t0, $t0, $s2	nop	3
	add $s3, $t0, $s2	sw $t0, 12($s1)	4
	sub $s1, $s1, $s2	nop	5
	bne $s1, $zero, Loop	nop	6

Yes, it would be faster, add $s3, $t0, $s2 and sub $s1, $s1, $s2 can be processed earlier, which can save 1 cycle.

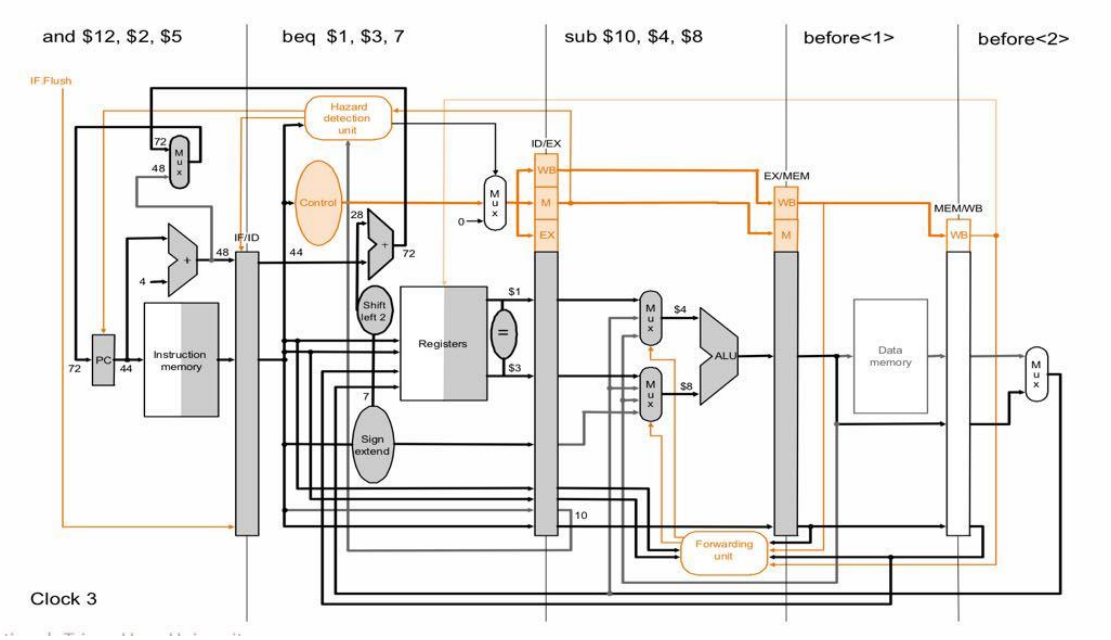
Dynamic:

	~~ALU/branch~~	~~Load/Store~~	cycle
Loop	nop	lw $t0, 8($s1)	1
	nop	nop	2
	add $t0, $t0, $s2	add $s3, $t0, $s2	3
	sub $s1, $s1, $s2	sw $t0, 12($s1)	4
	bne $s1, $zero, Loop	nop	5

[group 4]

10.

Please answer: what will be the next instruction in ID stage when (1)beq taken, (2)beq not taken. Additionally, what outcome does this hardware always predict when executing branch instructions?



Ans:

(1)bubble(NOP)

(2)add $12, $2, $5

Predict branch always not taken

[group 13]

11. What do we have to do to handle exceptions before jumping to OS?

Ans:

1. Save PC of offending (or interrupted) instruction
2. Save indication of the problem
3. Jump to handler at 8000 00180