Chapter 06: Instruction Pipelining and Parallel Processing

Lesson 12:

Out-of-order execution

Objective

- To understand ways of executing instructions using a pipeline—out-of-order

Out-of-order Executing processor

Example

```
LD      r1,  (r2)
ADD     r3, r1, r4
SUB     r5, r6, r7
MUL     r8, r9, r10
```

- Assume─ Load operations have a latency of 2 cycles, and all other operations have a latency of 1 cycle

Example

- Finding how long would the instructions take to issue on an out-of-order processor with two execution units, each of which can execute any instruction

Solution

- The only dependency in this sequence is between the LD and the ADD instructions (a RAW dependency)
- The ADD instruction must issue at least two cycles after the LD

… Solution

- The SUB and the MUL could both issue in the same cycle as the LD
- Using our greedy (to get higher performance) assumption, the SUB and the LD issue in cycle *n*, the MUL issues in cycle *n* + 1, and the ADD issues in cycle *n* + 2, giving a three-cycle issue time for this program

Instruction window in Out-of--order Executing processor

Instruction window

- Much larger instruction windows than in-order processors
- As much opportunity as possible to find instructions that can issue in a given cycle

Size of the instruction logic

- Increases quadratically with the number of instructions in the instruction window
- Each instruction in the window must be compared to all other instructions to determine the dependencies between them

Large instruction windows

- Expensive to implement in terms of the amount of hardware required

Determining the execution time of an instruction

Determining the execution time of an instruction

- Sequence on an out-of-order processor
- Assume— the processor's instruction window large enough to allow the processor to examine all of the instructions in the sequence simultaneously
- Execution Time (in Cycles) = Pipeline Latency + Issue Time –1 for each pipeline of processor

Instruction Window not large enough

- Predicting execution time becomes much more difficult, as it becomes necessary to keep track of which instructions are contained within the instruction window on any given cycle and only select instructions to issue from within that set

Problems in Out-of--order Executing processor

Difficult implementation issue in out of order case

- Handling interrupts and program exceptions
- Very difficult to determine exactly which instructions have executed when an instruction takes an exception or when an interrupt occurs

Difficult implementation issue in out of order case

- Difficult for the programmer to determine the cause of an exception and makes it hard for the system to return to execution of the original program when an interrupt handler completes

Solution in case of Problems in Out-of--order Executing processor

Out-of-order processors using in-order retirement

- When an instruction generates its result, the result is only written into the register file if all earlier instructions in the program have completed
- Otherwise, the result is saved until all earlier instructions have completed, and only then written into the register file

Out-of-order processors using in-order retirement

- Since results are written into the register file in order, the hardware can simply discard all results that are waiting to be written into the register file when an exception or interrupt occurs

Out-of-order processors using in-order retirement

- Presents the illusion that instructions are being executed in order
- Resume execution of the program at the next instruction when an interrupt handler completes

Out-of-order processors using in-order retirement

- Use bypassing logic to forward the result of an instruction to dependent instructions before the result is written into the register file

Out-of-order processors using in-order retirement

- Allows dependent instructions to issue as soon as an instruction generates its result, rather than having to wait until the instruction's result is written back into the register file

Summary

We learnt

- Out-of-Order Execution
- Instruction window larger in out-of-order
- Handling Interrupts and program exceptions
- In-order retirement

End of Lesson 12 on

Out-of-order execution