Chapter 06: Instruction Pipelining and Parallel Processing

Lesson 10:
In-order execution

Objective

- To understand two ways of executing instructions using a pipeline
- In-order execution
- Out-of-order

In-Order and out-of-order Execution

Two ways to issue instructions to a pipeline

(i) The processor can execute instructions in the order that they appear in the program (in-order execution), or

Two ways to issue instructions to a pipeline

(ii) The processor can execute instructions in any order that does not change the result of the program (out-of-order execution)

Out-of-order execution

- Can provide much better performance than in-order but requires more complex hardware to implement

Execution Time in Pipelined Processor

Execution time of programs on pipelined processor

- Divides into the time to issue all of the instructions in the program and pipeline latency of the processor
- Execution Time (in Cycles) = Pipeline Latency + Issue Time –1 for one pipeline processor
- Superscalar parallel pipelines
- Issue time calculation more complex in superscalar

Example

Example of a Program in 5 stage pipeline

- Assume─ instruction latencies of 3 cycles for non-branch and 4 cycles for branch instructions
- Assume─ instruction sequence

  ADD r1, r2, r3 ; arithmetic add

  SUB r4, r5, r6; arithmetic subtract

  MUL r8, r2, r1; arithmetic multiply

  ASH r5, r2, r1; arithmetic shift

  OR r10, r11, r4; logical OR r11 and r4

Execution time computations

- Assume—

(i) The ADD issues on cycle number *n*

(ii) The SUB— independent of the ADD

- SUB can issue on cycle *n* + 1, the cycle after the previous instruction in the program issues

Execution time computations

(iii) The MUL─ depends on the ADD

- Can't issue until cycle n + 3, because the ADD has a latency of 3 cycles

(iv) The ASH─ also dependent on the ADD

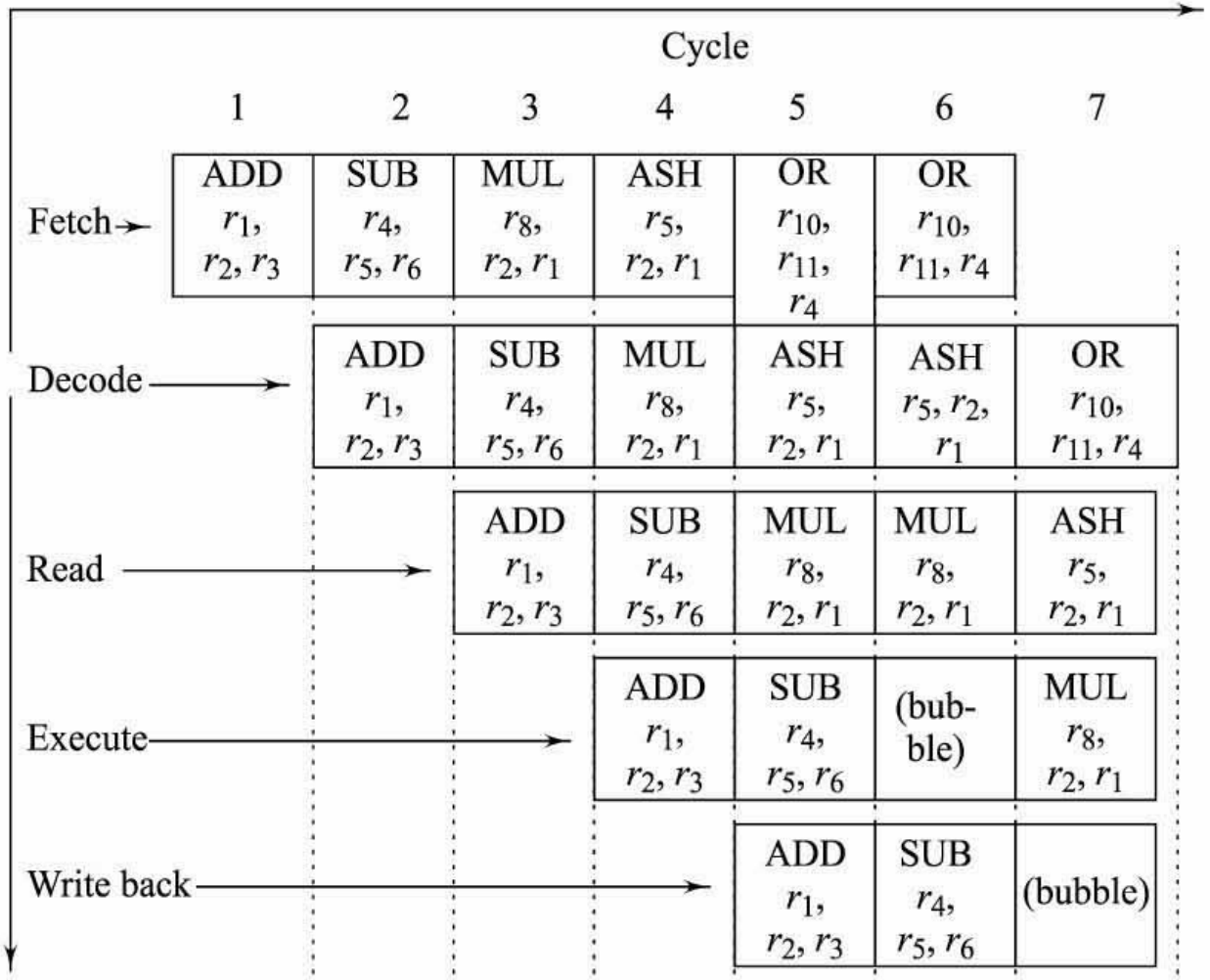
- Can't issue until cycle n + 4 because the MUL issues on cycle n+3

Execution time computations

(v) The OR─ independent of the previous instructions

- Issues on cycle $n + 5$
- Therefore, it takes 6 cycles to issue all of the instructions in the program
- Using the formula, the execution time of the program is 5 cycles (pipeline latency) + 6 cycles (time to issue the instructions in the program) –1 = 10 cycles

Pipeline stall in cycle 6 between SU and MULL



Lesson from the example

Result

- Illustrates an important factor in achieving good performance on pipelined processors—scheduling instructions to avoid pipeline stalls
- Because the SUB instruction did not use the result of the ADD, it was able to execute on the cycle immediately following the ADD

Effect of In order and Out of Order

- If the SUB and MUL instructions had been reversed, the MUL instruction would still have had to wait until three cycles after the ADD executed for its input data to be ready, and the SUB would have been unable to issue until four cycles after the ADD

... Result

- Compilers for pipelined processors must understand the details of the pipeline to be able to place instructions in an order, which maximizes performance

Instruction window in In-order Executing processor

Instruction window in In-order Executing processor

- Instruction window size = the number of instructions the processor examines to select instructions to issue in each cycle
- Processor is not allowed to issue an instruction until all of the instructions that appear before it in the program have been issued
- Small instruction window size

A processor with *n* execution units

- Only the next *n* instructions in the program can possibly be issued in a given cycle
- So an instruction window length of *n* instructions is generally sufficient

In-order superscalar processors and pipelined processors

- Assume─ execution of only one instruction per cycle
- The issue time of a program can be determined by stepping sequentially through the code to determine when each instruction can issue

Superscalar processor

- Can issue an instruction in the same cycle as the previous instruction in the program if the data dependencies allow, as long as the number of instructions issued in the cycle does not exceed the number of instructions that the processor can execute simultaneously

When some or all of the execution units can only execute some instructions

- The set of instructions issued on a given cycle must match the limitations of the execution units

Summary

We learnt

- In-order execution
- How to compute execution time
- Out-of-order benefits
- Instruction Window in In-order processor

End of Lesson 10 on

In-order execution