COMPUTER ORGANIZATION AND ARCHITECTURE

Course Code : CSE 2151

Credits : 04

INTRODUCTION TO PARALLEL ARCHITECTURE

MODULE 7

PIPELINING: OVERVIEW

- Pipelining is widely used in modern processors.
- Pipelining improves system performance in terms of throughput.
- Pipelined organization requires sophisticated compilation techniques.

PIPELINING: BASIC CONCEPTS

- Pipelined Execution:

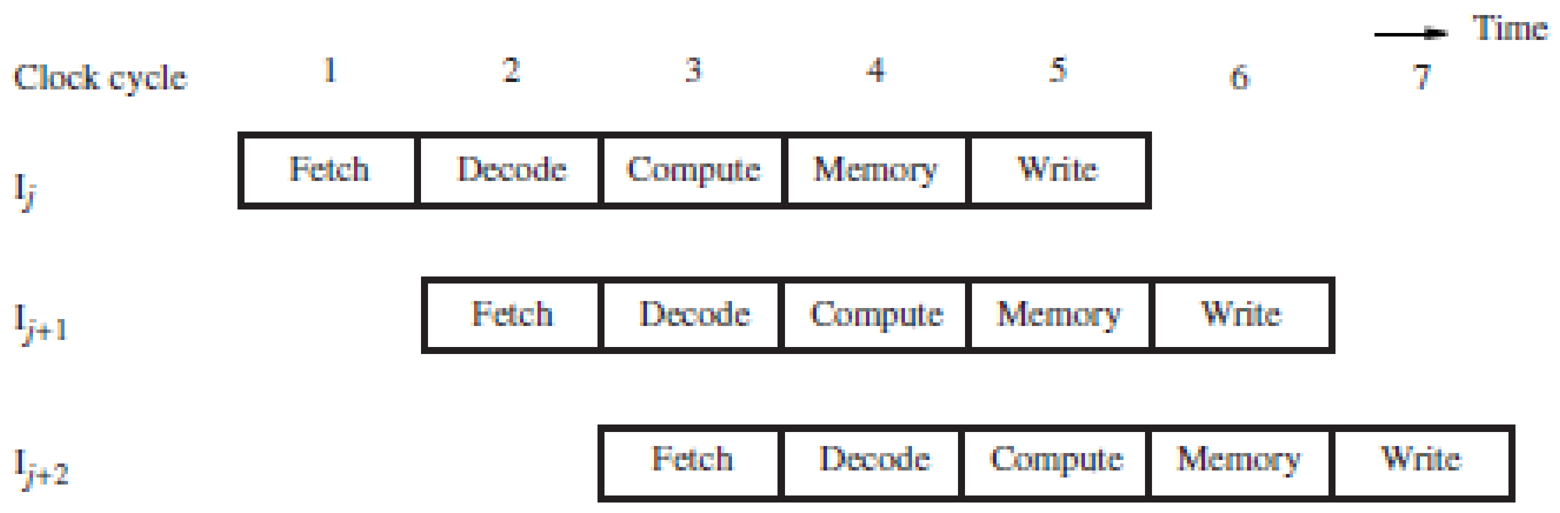


Figure 6.1 Pipelined execution—the ideal case.

PIPELINE ORGANIZATION

- The interstage buffers (for figures 5.7 and 5.8 from the textbook) are used as follows:
  - Interstage buffer B1 feeds the Decode stage with a newly-fetched instruction.
  - Interstage buffer B2 feeds the Compute stage with the operands read from the register file
  - Interstage buffer B3 holds the result of the ALU operation, which may be data to be written into the register file or an address that feeds the Memory stage. In case of a write access to memory, buffer B3 holds the data to be written
  - Interstage buffer B4 feeds the Write stage with a value to be written into the register file.

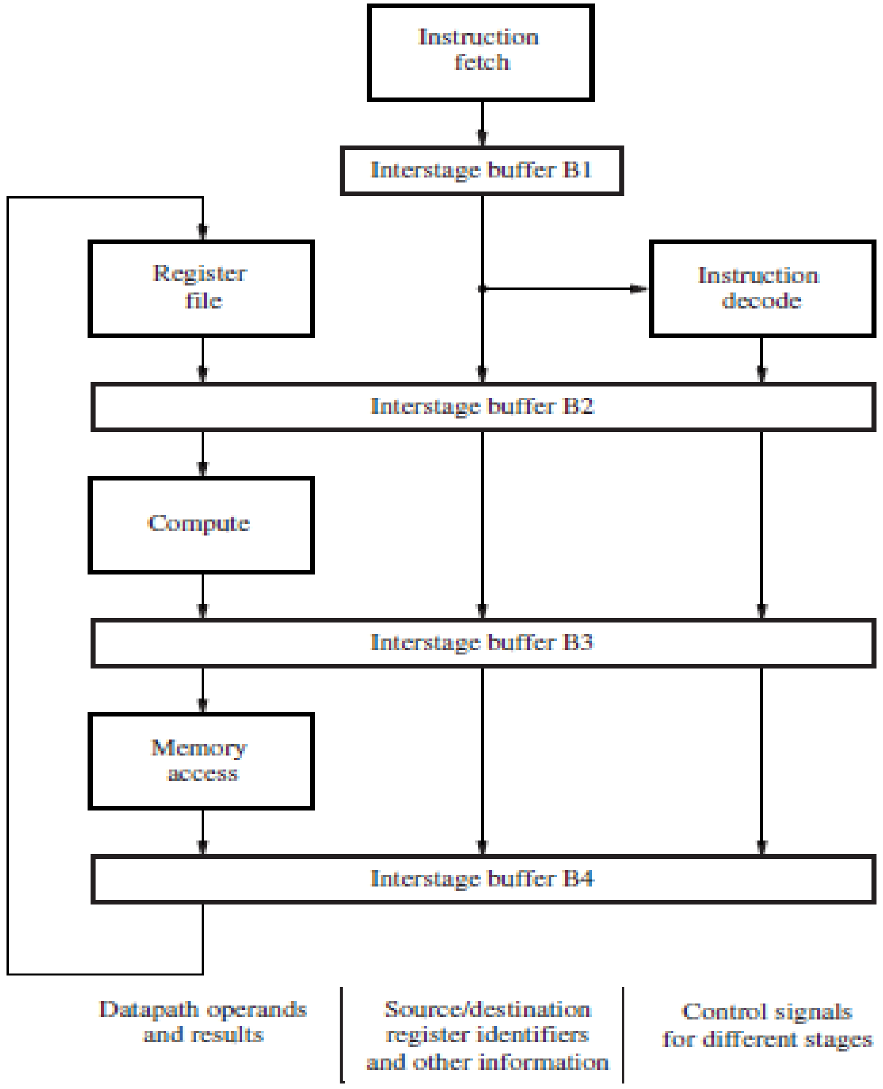


Figure 6.2 A five-stage pipeline.

PIPELINING ISSUES

- Consider the case of two instructions, I_j and I_{j+1}, where the destination register for instruction I_j is a source register for instruction I_{j+1}.
- To obtain the correct result it is necessary to wait until the new value is written into the register by instruction I_j .
- Hence, instruction I_{j+1} cannot read its operand until cycle 6, which means it must be stalled in the Decode stage for three cycles. While instruction I_{j+1} is stalled, instruction I_{j+2} and all subsequent instructions are similarly delayed. New instructions cannot enter the pipeline, and the total execution time is increased.

HAZARD

- Any condition that causes the pipeline to stall is called a hazard.
  - Data Hazard
  - Memory delays
  - Branch instructions

DATA HAZARDS

- Consider the two instructions in Figure 6.3:

  Add R2, R3, #100

  Subtract R9, R2, #30

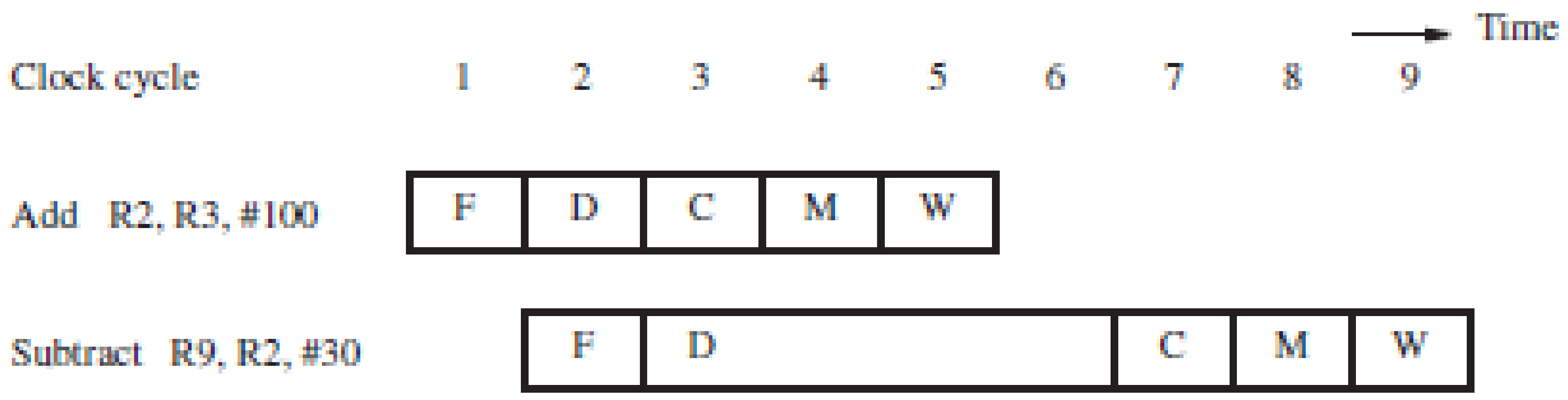


Figure 6.3 Pipeline stall due to data dependency.

OPERAND FORWARDING

- Instead of from the register file, the second instruction can get data directly from the output of ALU after the previous instruction is completed.
- A special arrangement needs to be made to "forward" the output of ALU to the input of ALU.

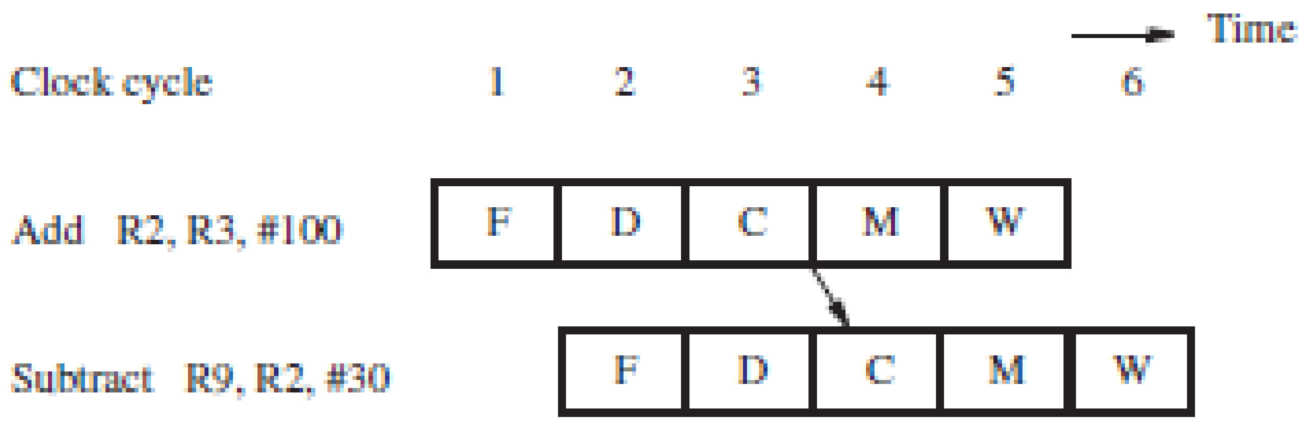


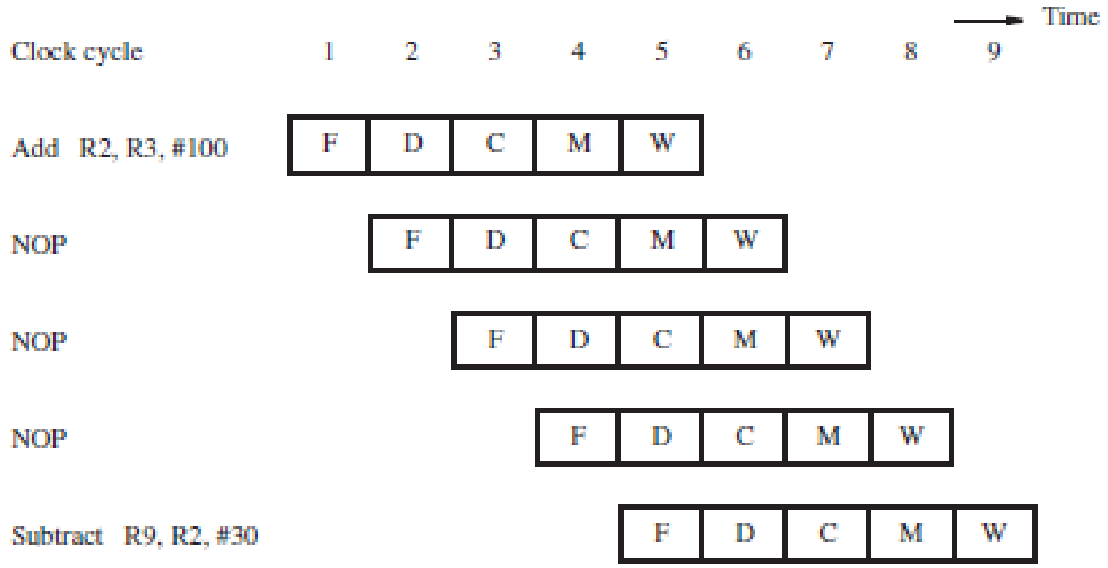
Figure 6.4 Avoiding a stall by using operand forwarding.

HANDLING DATA DEPENDENCIES IN SOFTWARE

- Compiler identifies a data dependency between two successive instructions I_j and I_{j+1}, it can insert three explicit NOP (No-operation) instructions between them.
- The NOPs introduce the necessary delay to enable instruction I_{j+1} to read the new value from the register file after it is written.
- The compiler can reorder the instructions to perform some useful work during the NOP slots

```
Add        R2, R3, #100
NOP
NOP
NOP
Subtract   R9, R2, #30
```

(a) Insertion of NOP instructions for a data dependency



(b) Pipelined execution of instructions

TOPICS COVERED FROM

- Textbook 1:
  - Chapter 6: 6.1- 6.4