Lecture-28: Instruction Cycle as pipelining

05 April 2024 14:56

Basic Instruction Cycle:-

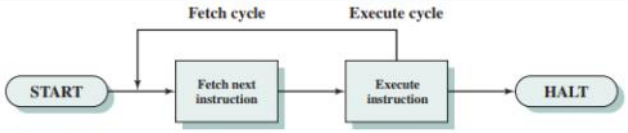


Figure 3.3 Basic Instruction Cycle

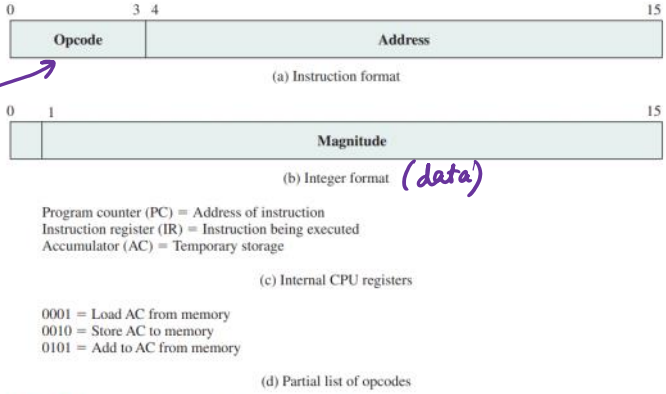
① At the beginning, the processor fetches an instruction from memory.
PC (Program Counter) hold the address of the inst^n to be fetched next.

② The fetched inst^n is loaded into a register in the processor (Inst^n Register IR).
↳ contains bits that specify the action the processor has to take.

③ Processor operations

- Processor-memory: processor ↕ data ↔ memory
- Processor-I/O: processor ↕ data ↔ I/O
- Data Processing: arithmetic/logic operations
- Control: an instruction which is sequence of executions.

Consider,



2^4 opcodes

Figure 3.4 Characteristics of a Hypothetical Machine

Example program execution,

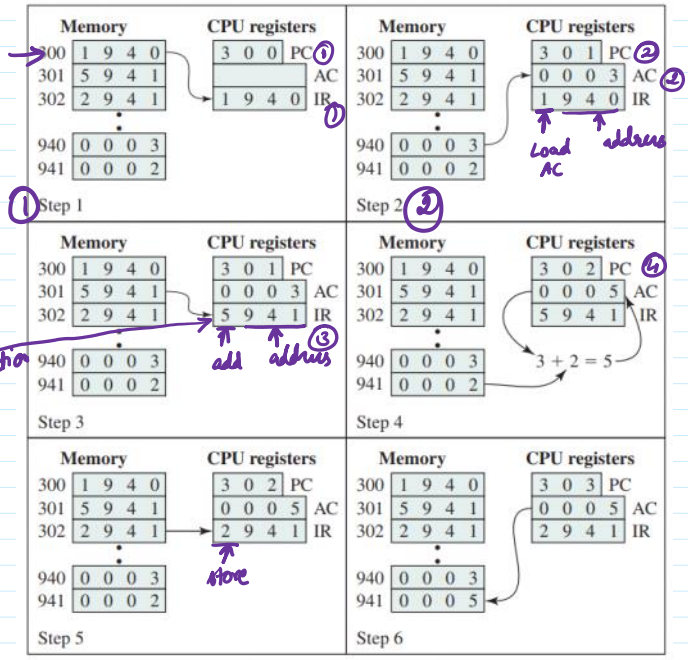
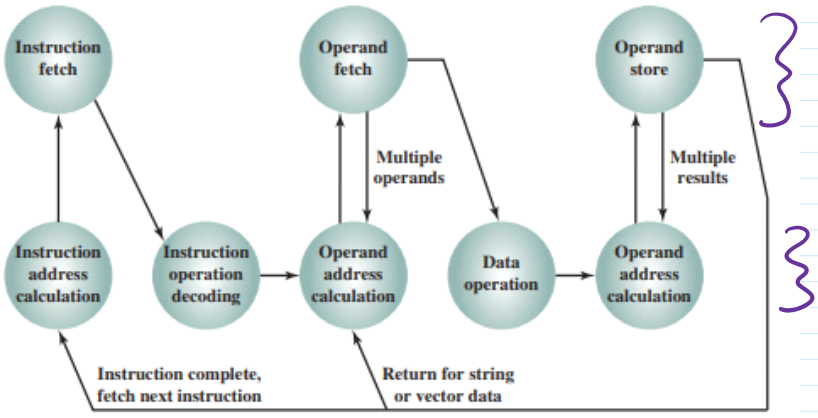


Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)

add(940) + add(941) = store(941)

Adds the contents of address 940 to address 941. and stores the results in 941. using 3 fetches and 3 execute cycles

State diagram of a detailed Instruction cycle :-



involves processor ⟷ m/m or I/O

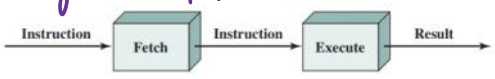
involves internal processor operations

Figure 3.6 Instruction Cycle State Diagram

For any inst^n cycle, some states may be null and others may be visited more than once.

Instruction pipelining :-

Let 2 stage inst^n pipeline :-



(a) Simplified view

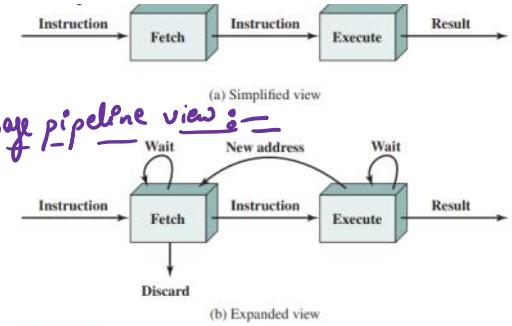


Figure 14.9 Two-Stage Instruction Pipeline

IF IE

To gain further speed up, the pipeline must have more stages.

① Fetch Instⁿ (FI) : read the next expected Instⁿ into a register.

② Decode Instⁿ (DI) : determine opcode & operand specifiers.

③ Calculate Operands (CO) : calculate effective address of each source operand. indirect, register indirect, etc.

④ Fetch Operands (FO) : fetch each operand from memory.

⑤ Execute Instⁿ (EI) : perform indicated operation and store the result.

⑥ Write Operand (WO) : store the result in memory

For the sake of illustration, let stages take equal duration.

Time

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO	EI	WO					
Instruction 5					FI	DI	CO	FO	EI	WO				
Instruction 6						FI	DI	CO	FO	EI	WO			
Instruction 7							FI	DI	CO	FO	EI	WO		
Instruction 8								FI	DI	CO	FO	EI	WO	
Instruction 9									FI	DI	CO	FO	EI	WO

Figure 14.10 Timing Diagram for Instruction Pipeline Operation

Without pipelining 6×9 = 54 time units
With pipelining 14 time units.

Various factors :—

① six stages are unequal duration.

② each instruction does not goes through all six stages.

(eg. LD) WO

③ Memory conflicts → FI, FO, WO involves memory access. Cannot access simultaneously.

④ Conditional branch instructions.

Let Inst^{n} 3 is a conditional branch to inst^{n} 15.

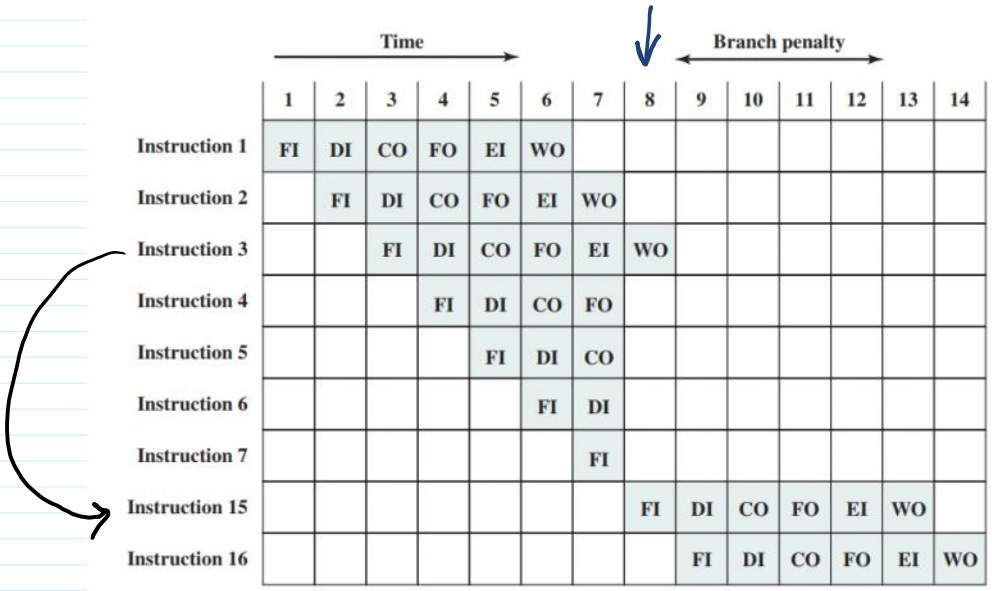


Figure 14.11 The Effect of a Conditional Branch on Instruction Pipeline Operation

Pipelining logic accounting for branches and interrupts

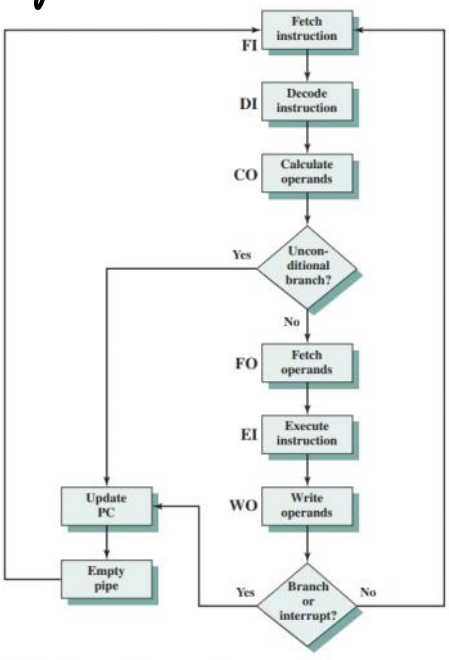


Figure 14.12 Six-Stage CPU Instruction Pipeline

The pipelining is a powerful technique for enhancing

performance but requires careful design to achieve optimum results with reasonable complexity.

"Pipelining Hazards"