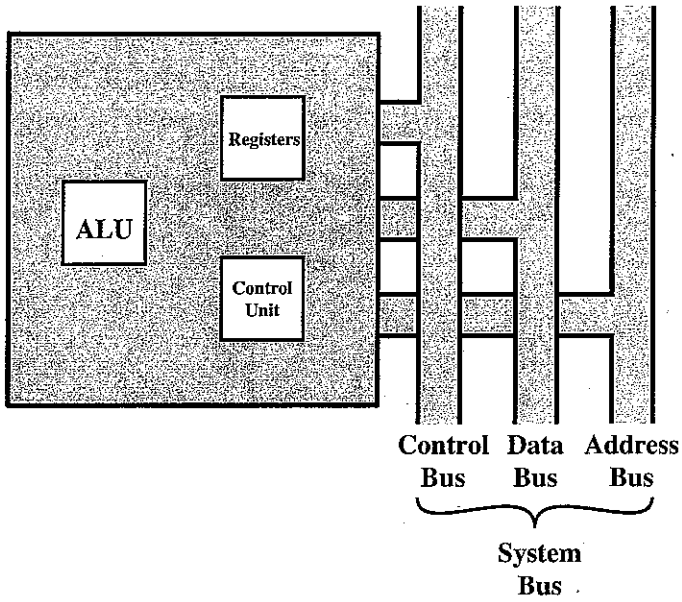
12. Processor (CPU)

12.1 Processor Organization

structure

function

- o Fetch instruction
- o Interpret instruction
- o Fetch data
- o Process data
- o Write data



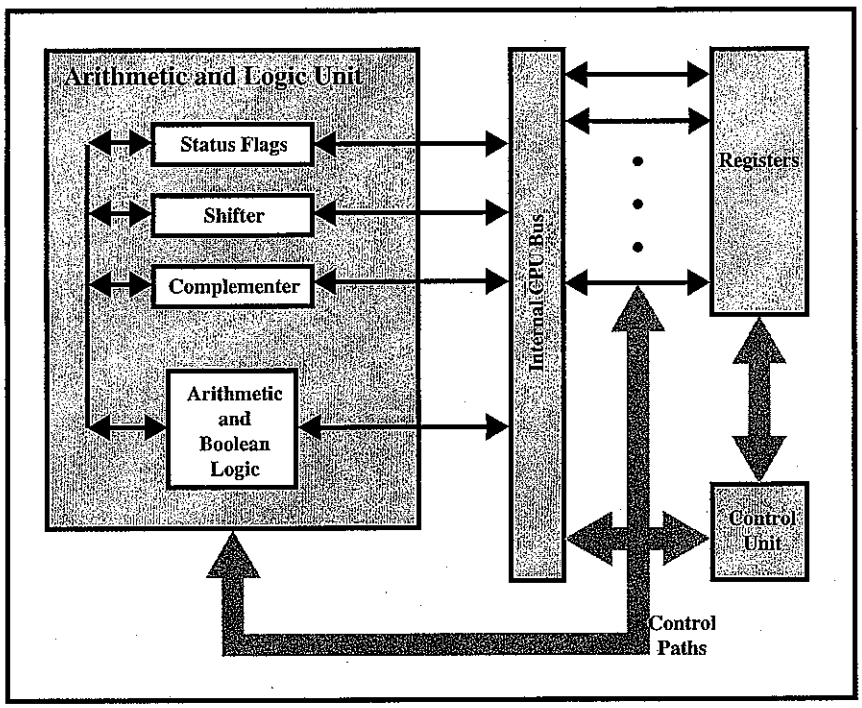


Figure 12.2 Internal Structure of the CPU

12.2 REGISTERS

(1) User-visible Registers

Ex. IBM S/370

- o 16 general-purpose registers – addressing, data, indexing
- o 4 floating-point registers

Ex. Intel Microprocessor

- data register: AX, BX, CX, DX
- address register
  - segment register: SS, DS, CS, ES
  - index register: SI, DI
  - stack pointer: BP

Issues

1. general vs. special - trade-off, no clear-cut
2. number of registers
   - one register – accumulator
   - 8 ~ 32 registers (common)
   - hundreds – RISC
3. length – word, double-word

(2) System Registers

- o control (to execute programs)
  - . PC
  - . IR
  - . MAR – address bus
  - . MBR – data bus
- o status (PSW)
  - sign
  - zero
  - carry
  - overflow
  -

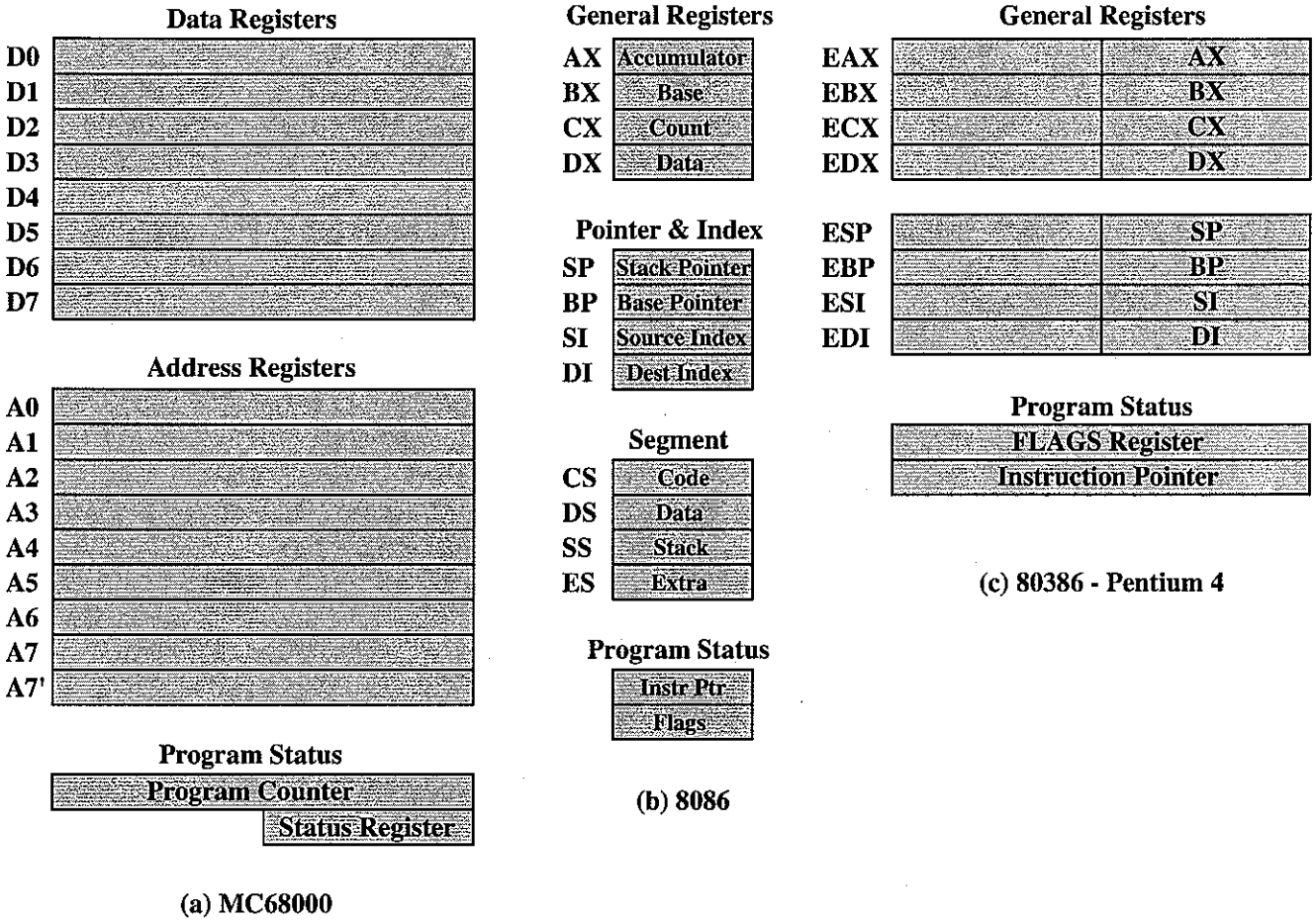


Figure 12.3 Example Microprocessor Register Organizations

12.3 INSTRUCTION CYCLE

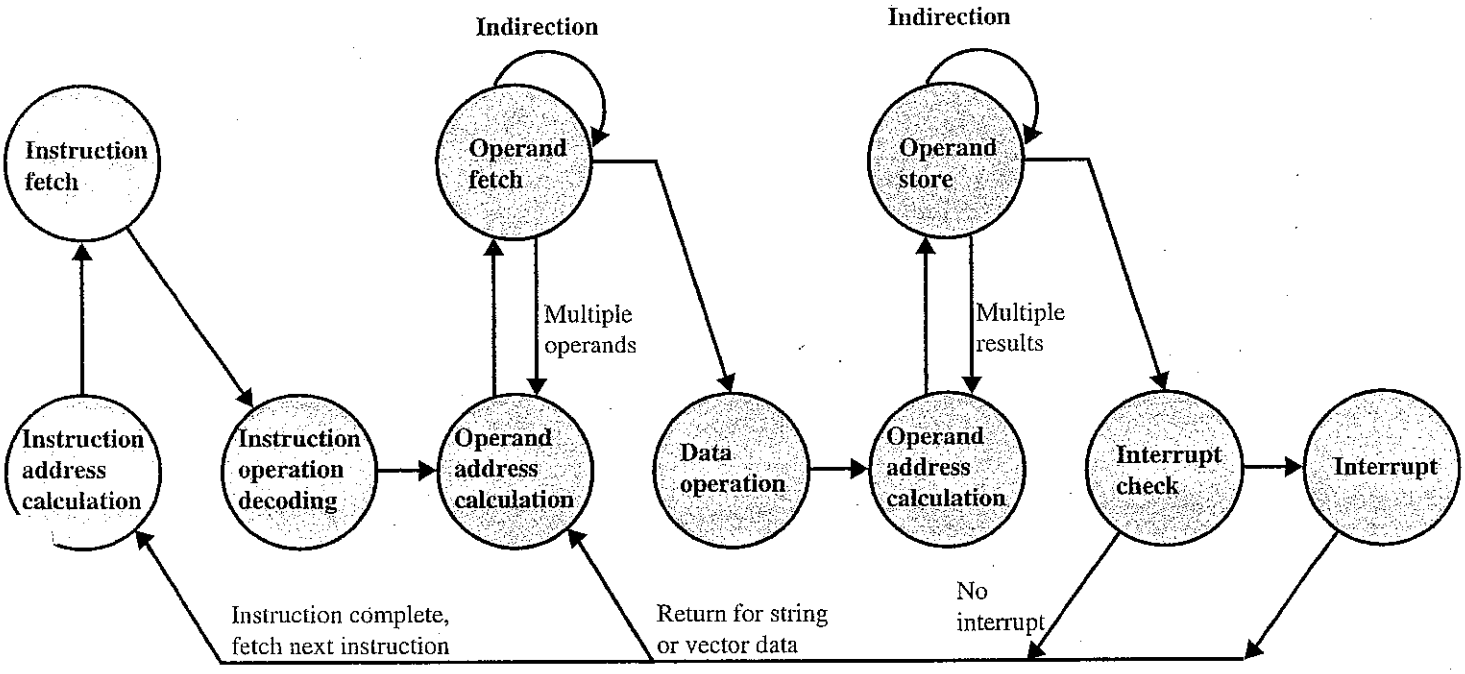


Figure 12.5 Instruction Cycle State Diagram

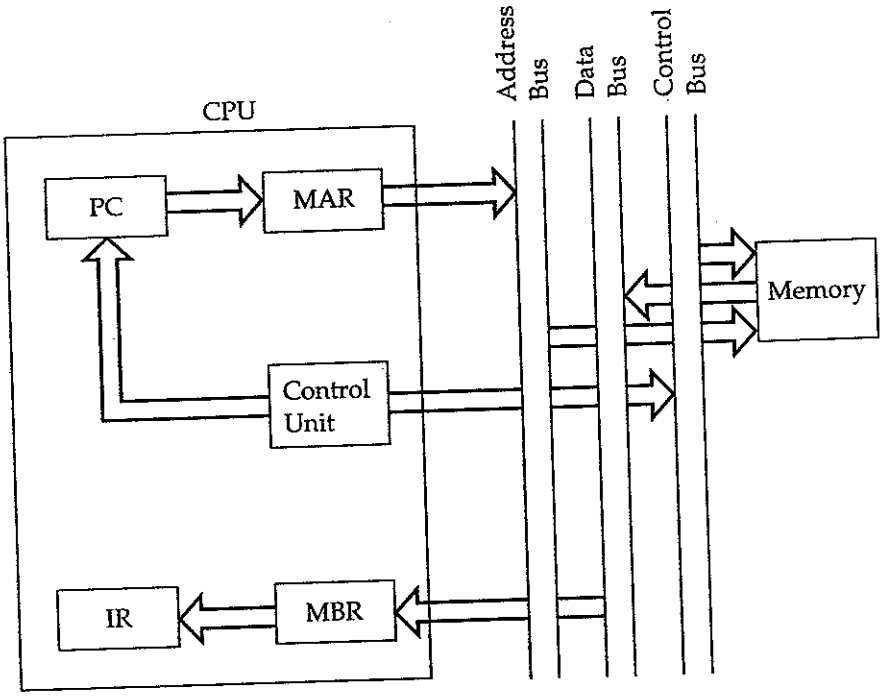


FIGURE 12.6 Data flow, fetch cycle

t_1: MAR ← (PC)
t_2: MBR ← Memory
PC ← (PC) + 1
t_3: IR ← (MBR)

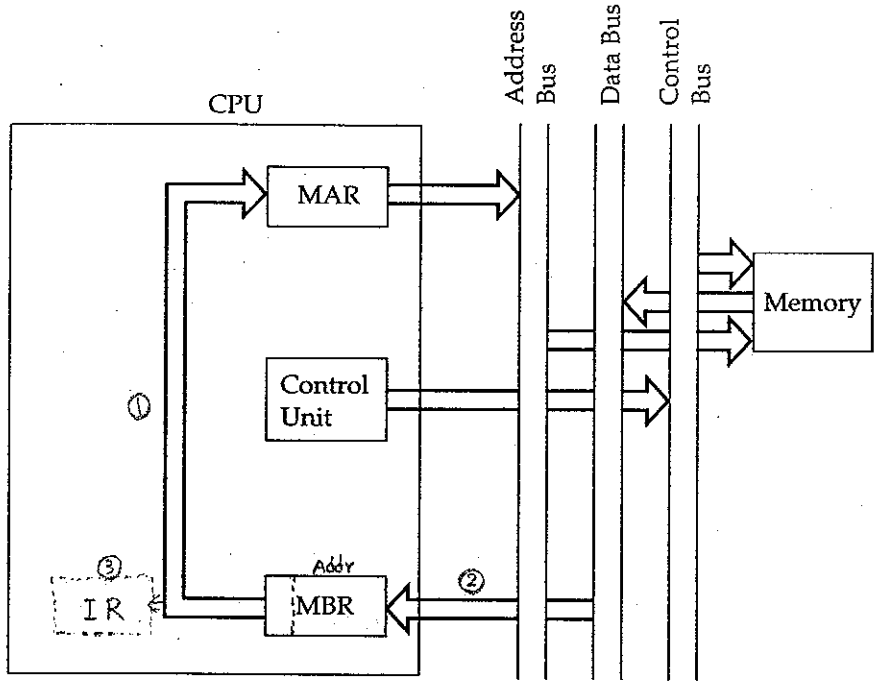


FIGURE 12.7. Data flow, indirect cycle

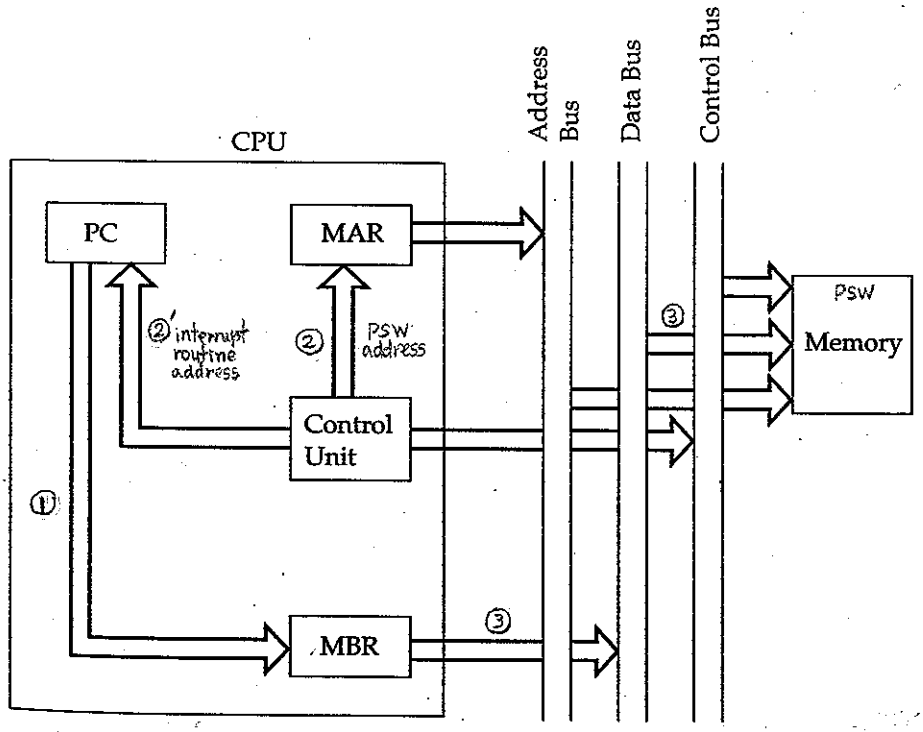


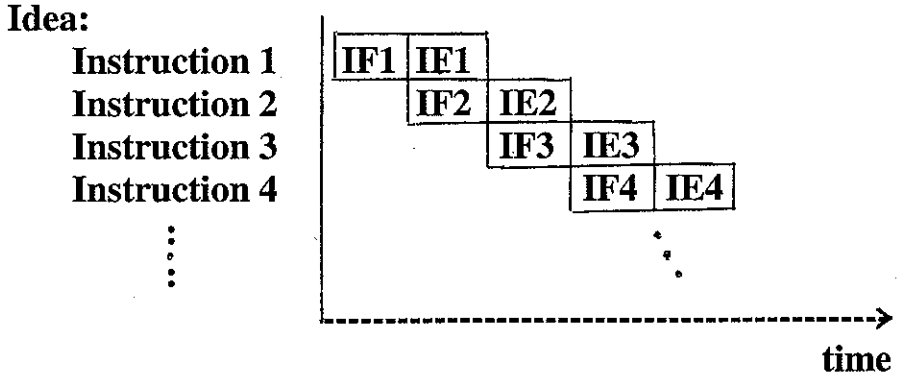
FIGURE 12.8. Data flow, interrupt cycle.

Methods that improves Performance

- bit-parallel arithmetic (carry look-ahead adder)
- separate I/O processor – interrupt-driven (DMA)
- memory interleaving – lower memory contentions
- associative memory – cache
- multiple registers
- instruction pipelining – IBM 7090 (1959)
- multiple functional unit – CDC 6600/7600
- pipelined functional unit
  - STAR 100 – pipelined adder (4 pipes)
  - Vector supercomputers (Cray, Fujitsu, NEC)

Instruction Pipeline

Idea:



Note. Pipeline lengths may not be the same. Usually, |IE| > |IF|
Conditional branch instruction
The more stages, the higher speedup
As the number of pipes increases, overhead increases, too.
(logic controlling the gating between stages)

Ex. Six-stage pipeline

FI → DI → CO → FO → EI → WO

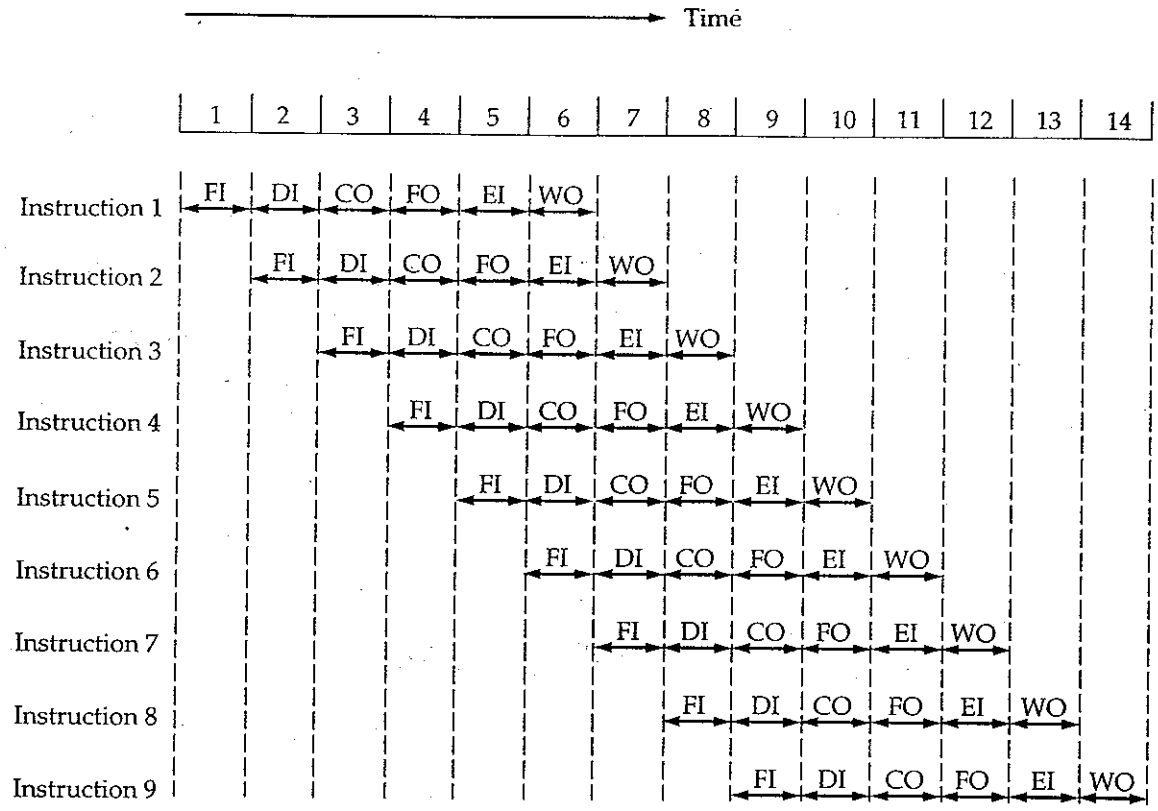


FIGURE 12.10. Timing Diagram for instruction pipeline operation

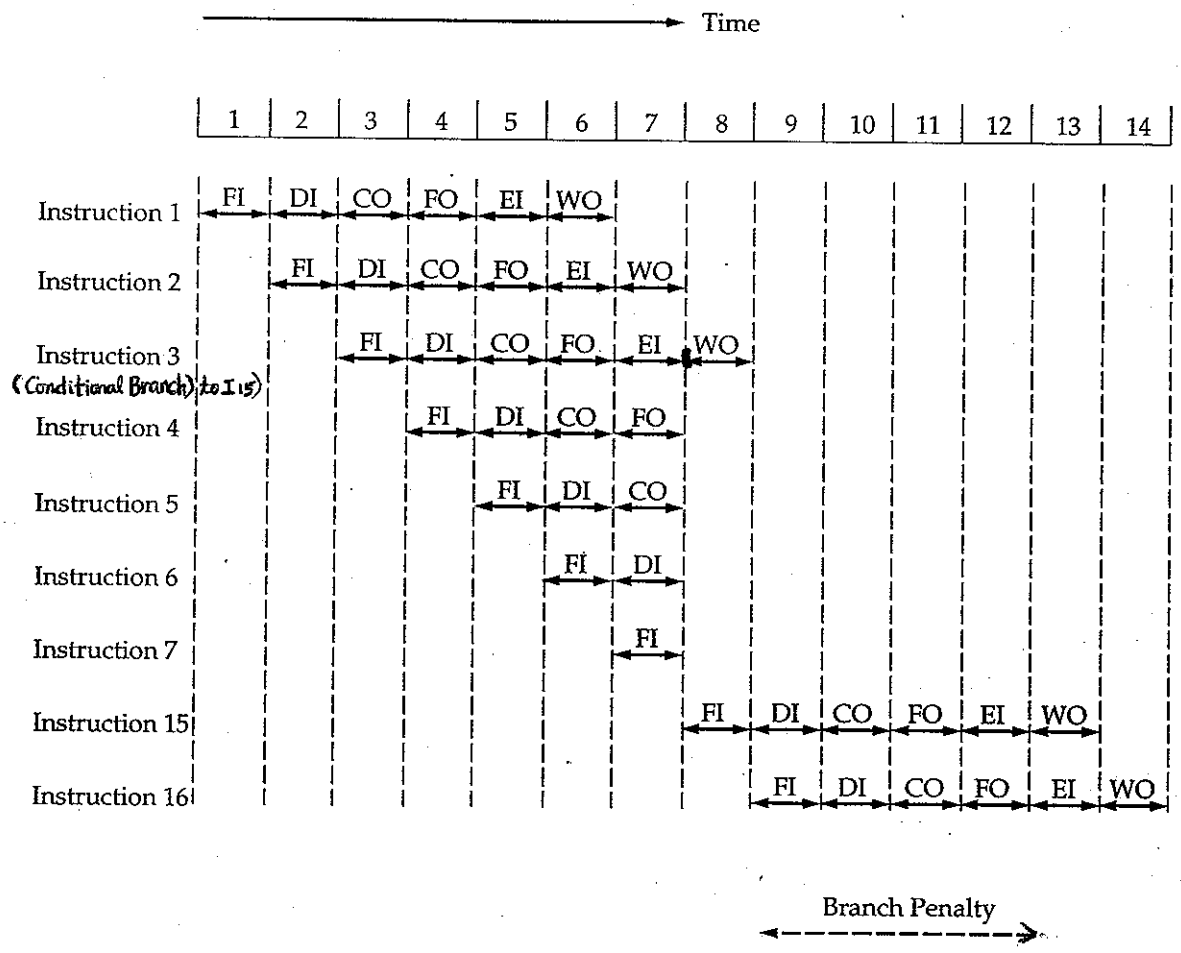


FIGURE 12.11. The effect of a conditional branch on instruction pipeline operation

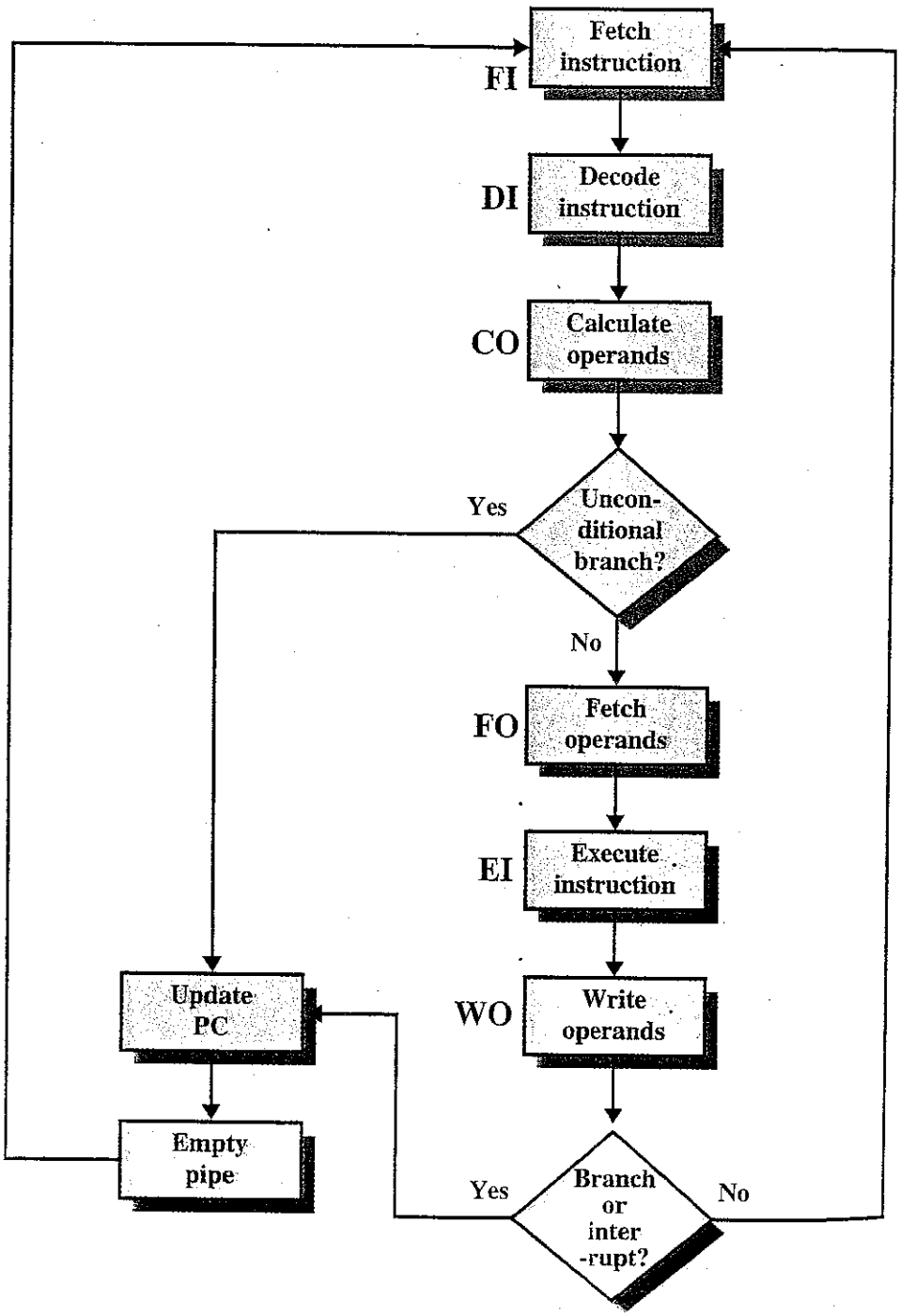
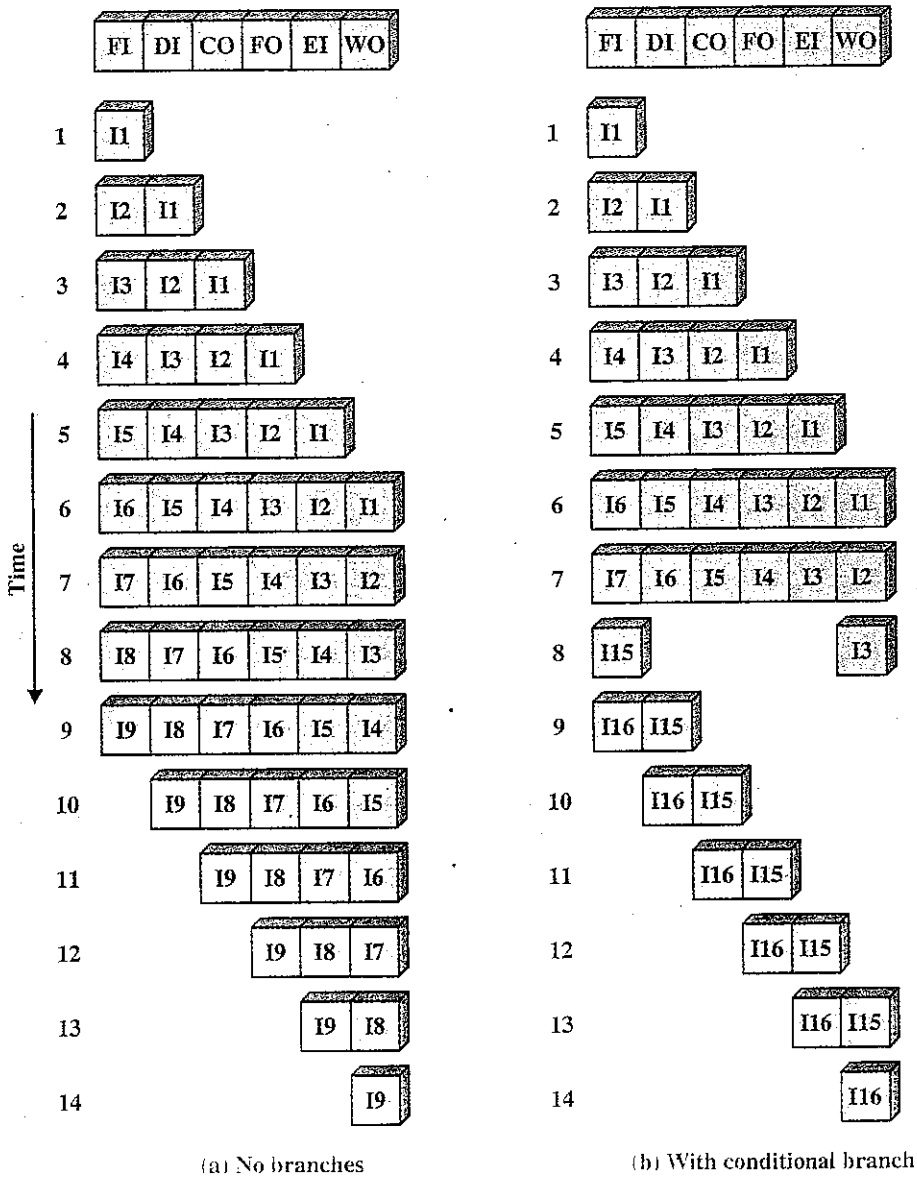


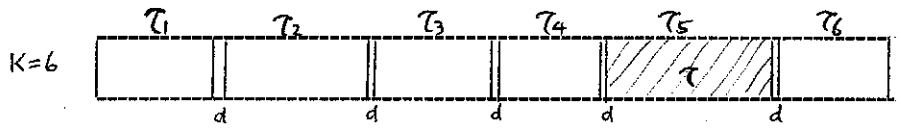
Figure 12.12 Six-Stage CPU Instruction Pipeline



(a) No branches (b) With conditional branch

Figure 12.13 An Alternative Pipeline Depiction

Instruction Pipelining Performance



$\tau = \max_i [\tau_i] + d, \quad 1 \le i \le k$

$= \tau_m + d$ // d is time delay between stages – to move signals/data //

In general, $\tau_m >> d \rightarrow \tau \doteq \tau_m$

Let Tk,n be total time taken for n instructions on k-stage pipeline

$T_{k,n} = [k + (n-1)]\,\tau$ // $(n-1)\tau$ for initial pipe filling / pipe flushing //

Example. 8th instruction completion time (Fig 12.10)

Here, k = 6, n = 8

T6,8 = [6 + 7] = 13

Def. Speedup

$$S_k = \frac{T_{1,n}}{T_{k,n}} = \frac{n\,k\,\tau}{[k + (n-1)]\,\tau} = \frac{nk}{k + (n-1)}$$

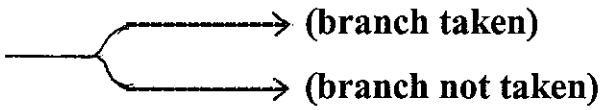
Note. $\lim_{n \to \infty} S_K \doteq K$

- The larger the number of pipeline stages, the larger the speedup. (Fig. 12.14)
- As the number of stages increases, the complexity/cost increases.

Conclusion. Instruction pipelining is a powerful technique for enhancing performance, but requires careful design to achieve optimum results with reasonable complexity.

Conditional Branch

(1) Multiple pipelines

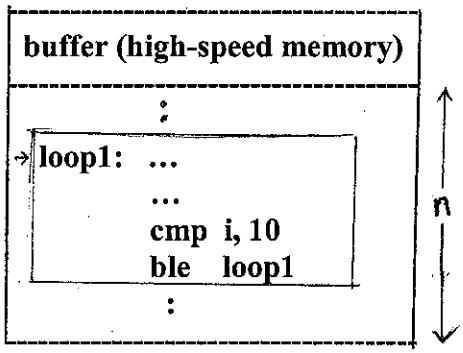


- IBM System 370/168, IBM 3033
- expensive

(2) Pre-fetch branch target

- IBM System 360/91

(3) Loop buffer



- n most recently used instructions – pre-fetched (only once)
- consecutive instructions, such as loop
- similar to cache
- CDC 6600, STAR 100, Cray 1

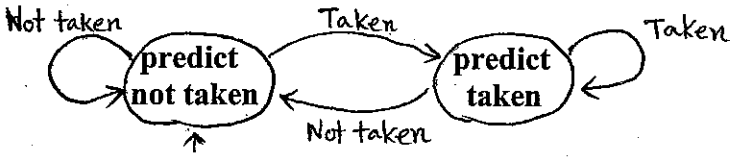
(4) Branch prediction

<u>Static</u>

- branch never taken – Vax 11/780
- branch always taken
  (not applied when page fault expected)
- predict by operation code – 75%

Dymanic

- **one-bit dynamic branch prediction** (switch)



- **Two-bit dynamic branch prediction** (last 2 branch history)

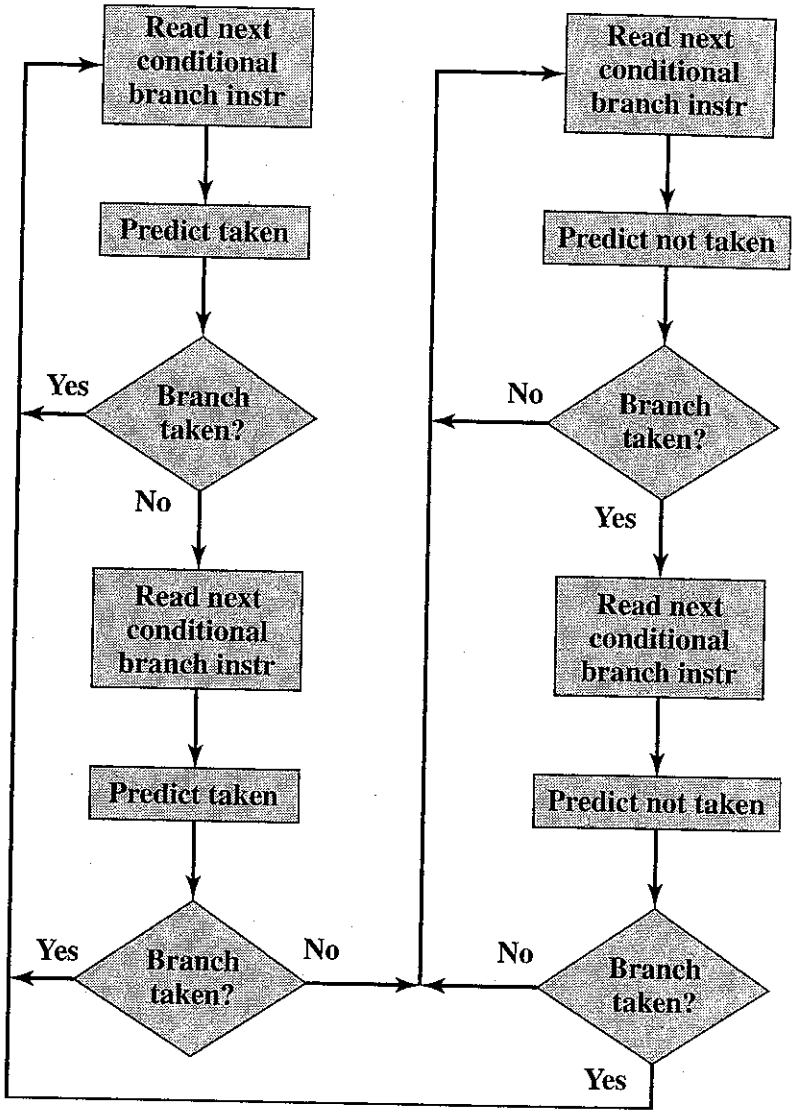


Figure 12.16 Branch Prediction Flowchart

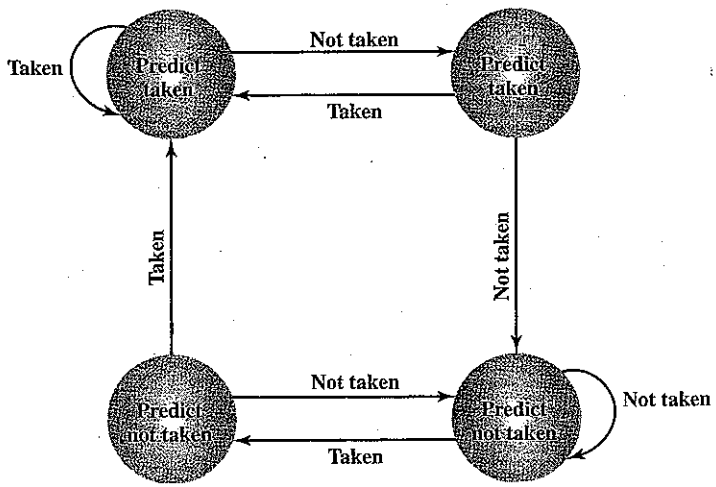


Figure 12.17 Branch Prediction State Diagram

Example - nested loop

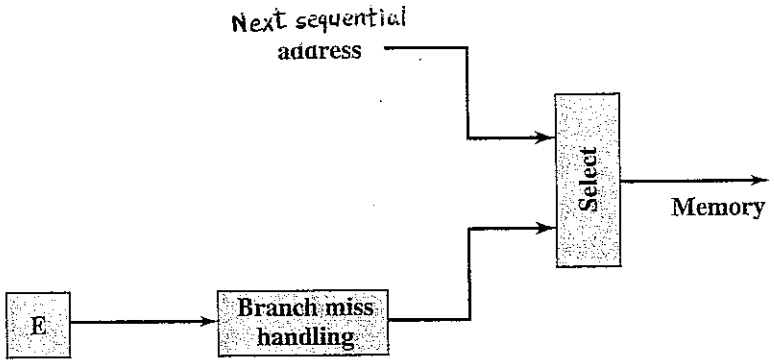
```
for i:= 1 to n do
    for j:= 1 to 4 do
```

(1) 1-bit dynamic branch prediction

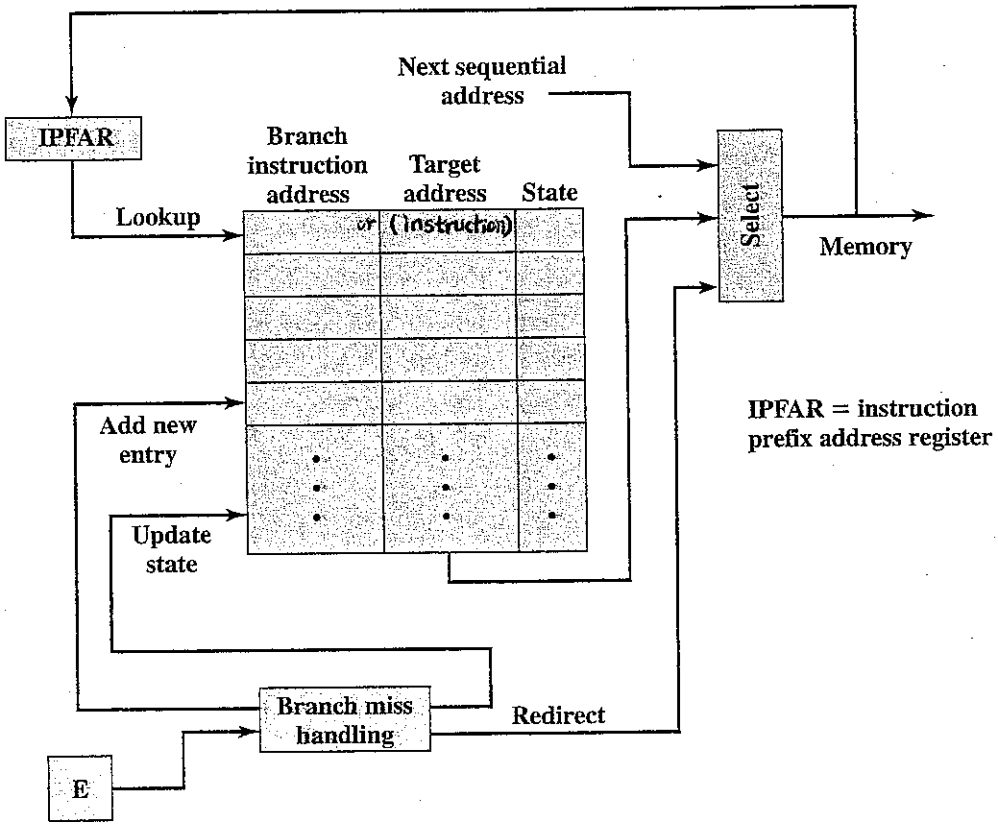
Prediction	N	Y	Y	Y	Y	N	Y	Y	Y	Y	N	Y	Y	Y	Y	...
Taken	Y	Y	Y	Y	N	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	...
incorrect	x				x	x				x	x					

(2) 2-bit dynamic branch prediction

Prediction	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	...
Taken	Y	Y	Y	Y	N	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	...
incorrect	x	x			x					x						

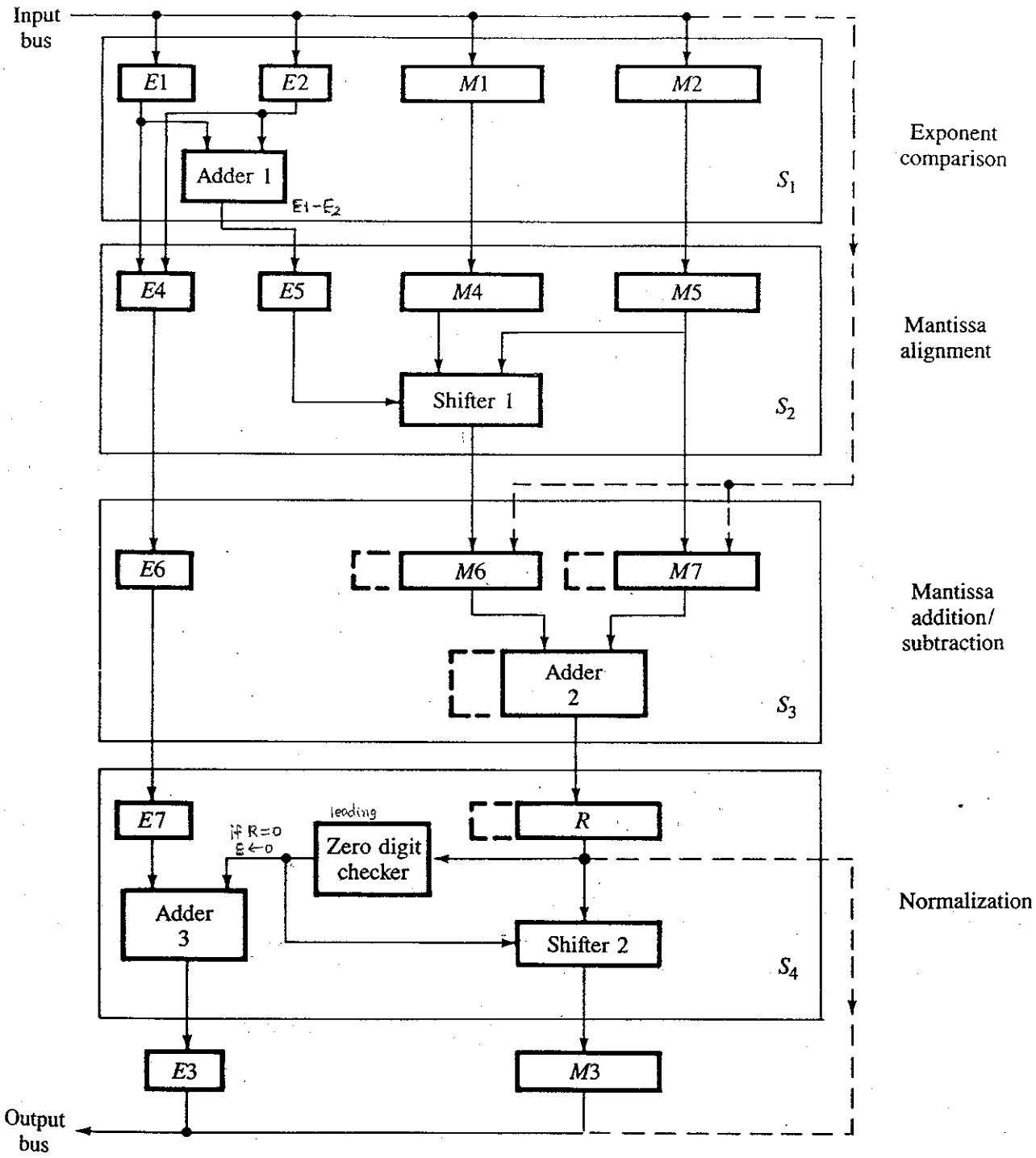


(a) Predict never taken strategy



(b) Branch history table strategy

Figure 12.18 Dealing with Branches AMD 29000



Pipelined version of the floating-point adder