

# ***Organisasi Sistem Komputer***

***OSK 12 – Control Unit***

**Muh. Izzuddin Mahali, M.Cs.**





## ❖ CPU must:

- Fetch instructions
- Interpret instructions
- Fetch data
- Process data
- Write data





- ❖ CPU must have some working space (temporary storage)
- ❖ Called registers
- ❖ Number and function vary between processor designs
- ❖ One of the major design decisions
- ❖ Top level of memory hierarchy



# *User Visible Registers*



- ❖ General Purpose
- ❖ Data
- ❖ Address
- ❖ Condition Codes



# General Purpose Registers (1)



- ❖ May be true general purpose
- ❖ May be restricted
- ❖ May be used for data or addressing
- ❖ Data
  - Accumulator
- ❖ Addressing
  - Segment



# General Purpose Registers (2)



- ❖ Make them general purpose
  - Increase flexibility and programmer options
  - Increase instruction size & complexity
- ❖ Make them specialized
  - Smaller (faster) instructions
  - Less flexibility



# How Many GP Registers?



- ❖ Between 8 - 32
- ❖ Fewer = more memory references
- ❖ More does not reduce memory references and takes up processor real estate
- ❖ See also RISC



# How big?



- ❖ Large enough to hold full address
- ❖ Large enough to hold full word
- ❖ Often possible to combine two data registers
  - C programming
  - `double int a;`
  - `long int a;`





# Condition Code Registers



- ❖ Sets of individual bits
  - e.g. result of last operation was zero
- ❖ Can be read (implicitly) by programs
  - e.g. Jump if zero
- ❖ Can not (usually) be set by programs



# *Control & Status Registers*



- ❖ Program Counter
  - ❖ Instruction Decoding Register
  - ❖ Memory Address Register
  - ❖ Memory Buffer Register
- 
- ❖ Revision: what do these all do?



# Program Status Word



- ❖ A set of bits
- ❖ Includes Condition Codes
- ❖ Sign of last result
- ❖ Zero
- ❖ Carry
- ❖ Equal
- ❖ Overflow
- ❖ Interrupt enable/disable
- ❖ Supervisor



# Supervisor Mode



- ❖ Intel ring zero
- ❖ Kernel mode
- ❖ Allows privileged instructions to execute
- ❖ Used by operating system
- ❖ Not available to user programs

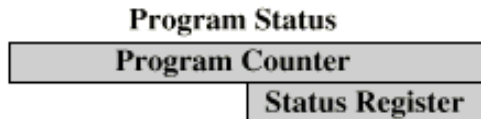
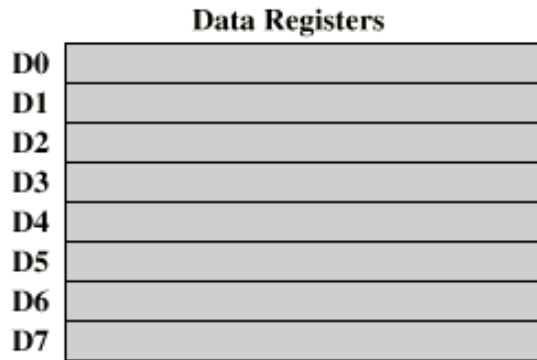
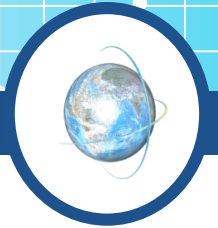




- ❖ May have registers pointing to:
  - Process control blocks (see O/S)
  - Interrupt Vectors (see O/S)
  
- ❖ N.B. CPU design and operating system design are closely linked

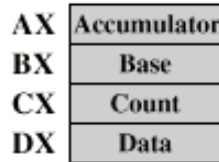


# Example Register Organizations

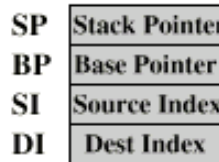


(a) MC68000

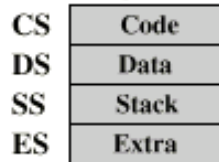
**General Registers**



**Pointer & Index**



**Segment**

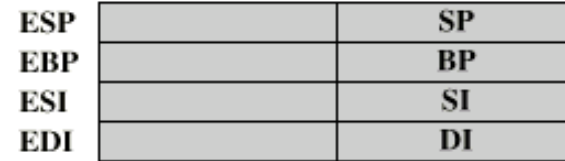
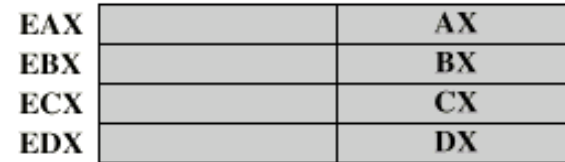


**Program Status**

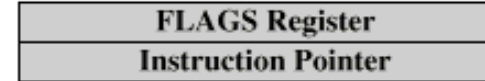


(b) 8086

**General Registers**



**Program Status**



(c) 80386 - Pentium II



# Foreground Reading



- ❖ Stallings Chapter 11
- ❖ Manufacturer web sites & specs





- ❖ Revision
- ❖ Stallings Chapter 3





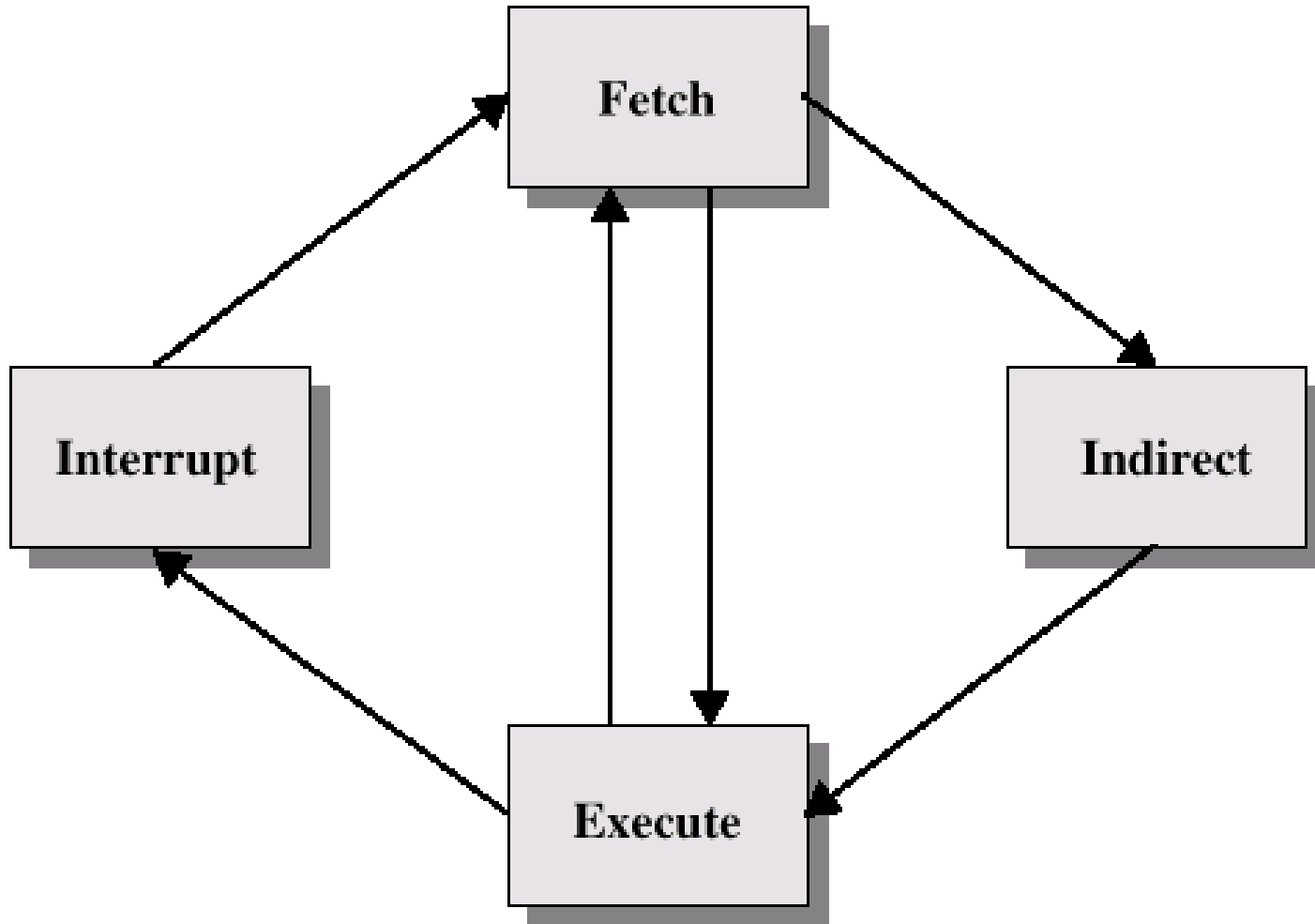
# *Indirect Cycle*



- ❖ May require memory access to fetch operands
- ❖ Indirect addressing requires more memory accesses
- ❖ Can be thought of as additional instruction subcycle



# Instruction Cycle with Indirect





# Data Flow (Instruction Fetch)



- ❖ Depends on CPU design
- ❖ In general:
  - ❖ Fetch
    - PC contains address of next instruction
    - Address moved to MAR
    - Address placed on address bus
    - Control unit requests memory read
    - Result placed on data bus, copied to MBR, then to IR
    - Meanwhile PC incremented by 1



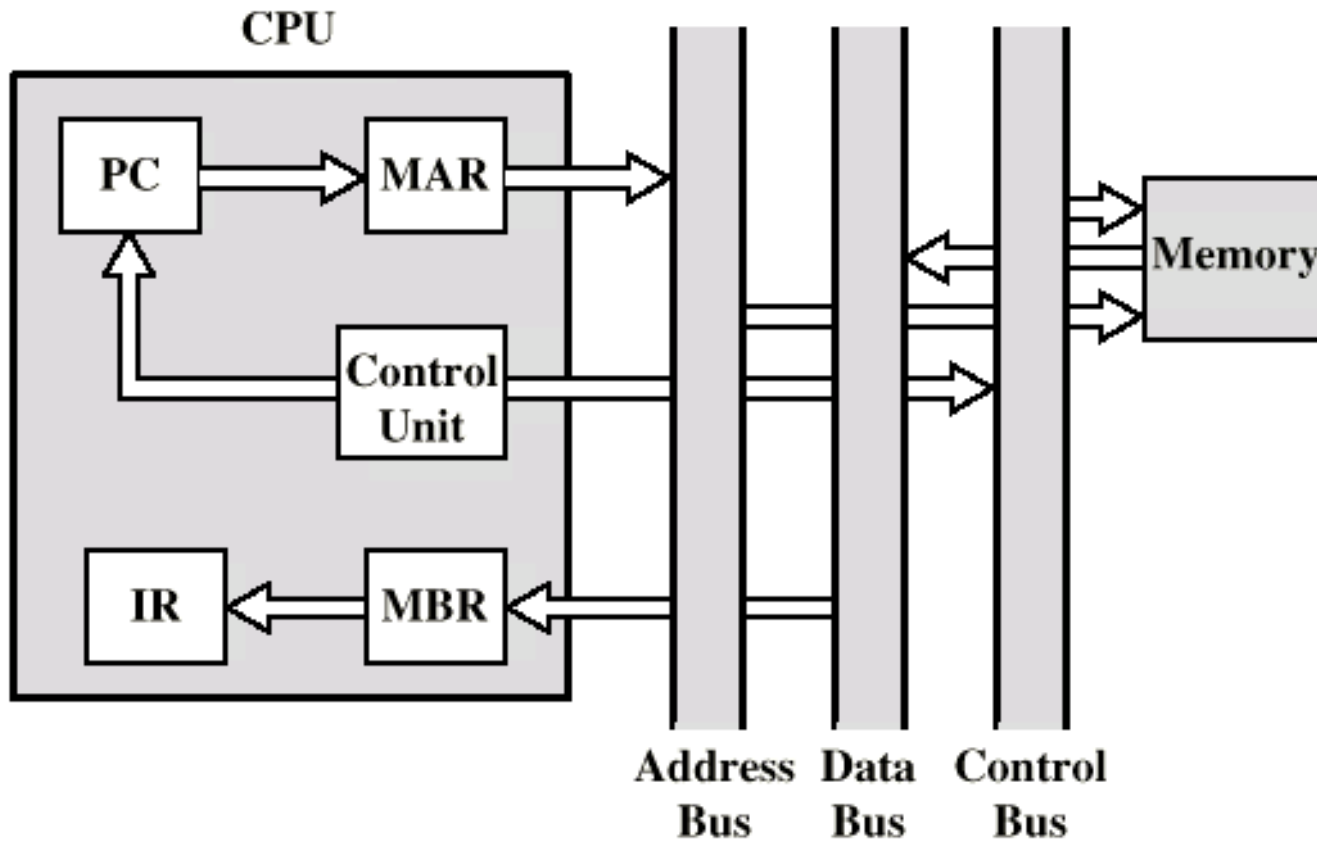
# Data Flow (Data Fetch)



- ❖ IR is examined
- ❖ If indirect addressing, indirect cycle is performed
  - Right most N bits of MBR transferred to MAR
  - Control unit requests memory read
  - Result (address of operand) moved to MBR



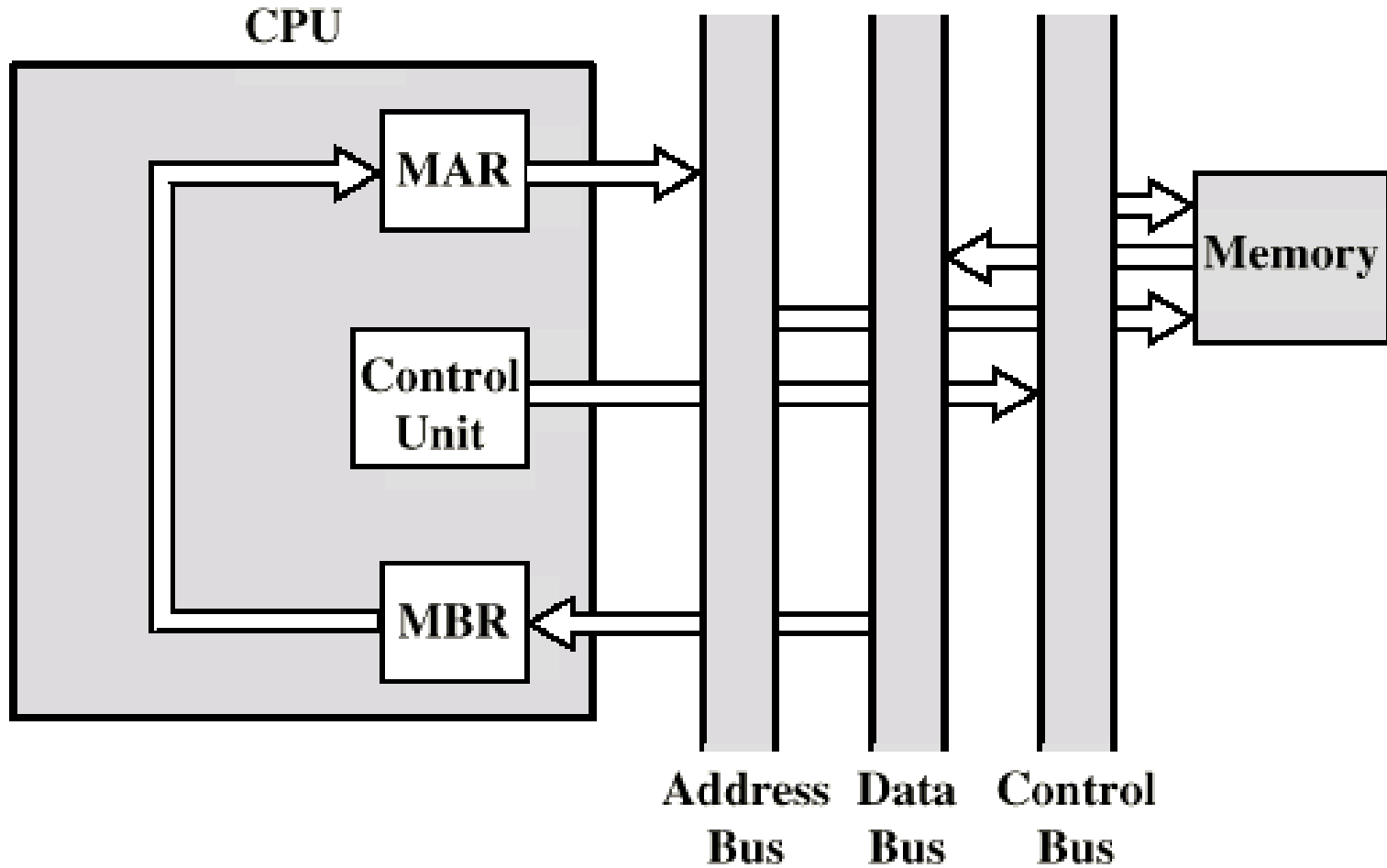
# Data Flow (Fetch Diagram)



MBR = Memory buffer register  
MAR = Memory address register  
IR = Instruction register  
PC = Program counter



# Data Flow (Indirect Diagram)



# Data Flow (Execute)



- ❖ May take many forms
- ❖ Depends on instruction being executed
- ❖ May include
  - Memory read/write
  - Input/Output
  - Register transfers
  - ALU operations





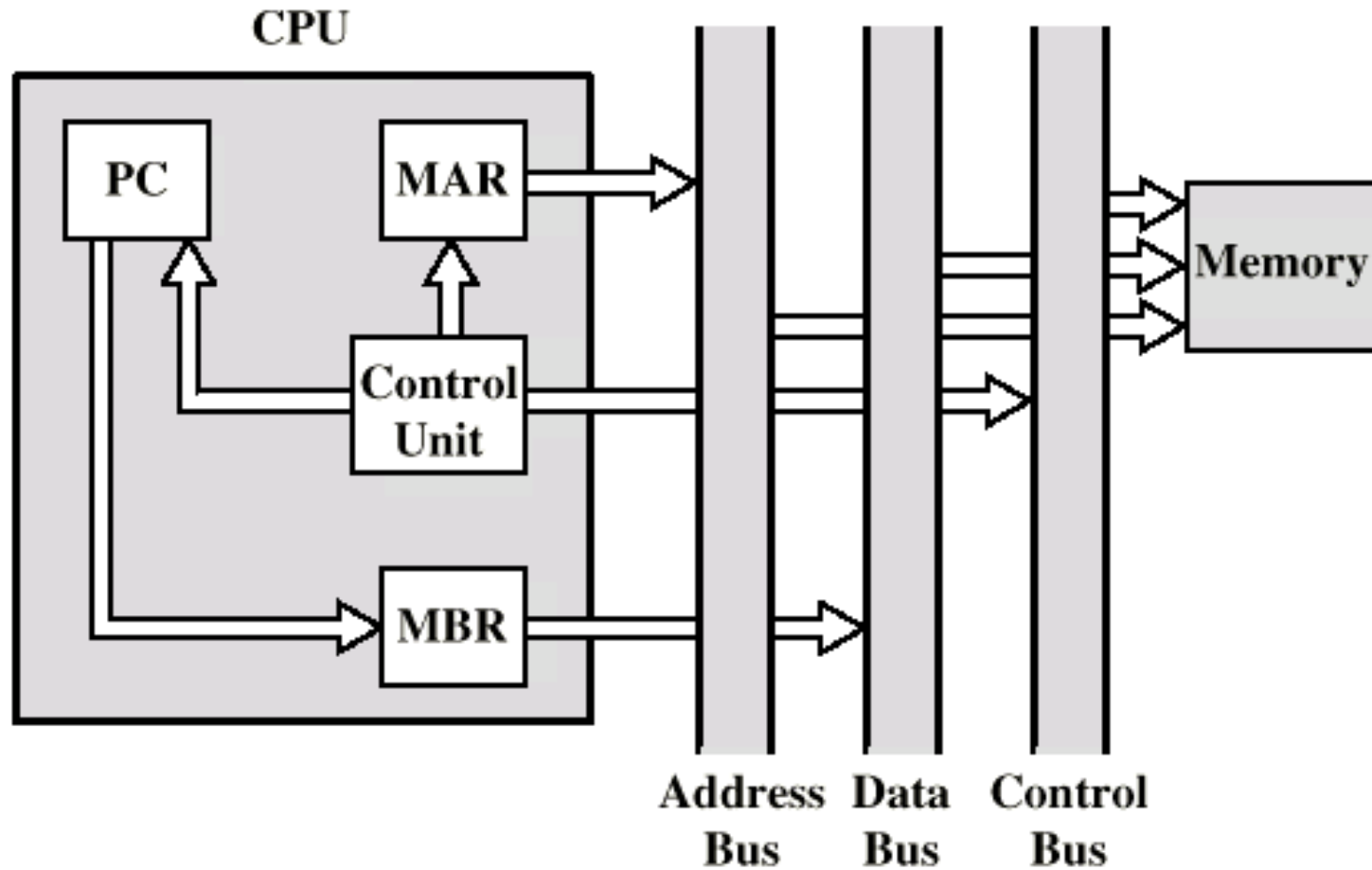
# Data Flow (Interrupt)



- ❖ Simple
- ❖ Predictable
- ❖ Current PC saved to allow resumption after interrupt
- ❖ Contents of PC copied to MBR
- ❖ Special memory location (e.g. stack pointer) loaded to MAR
- ❖ MBR written to memory
- ❖ PC loaded with address of interrupt handling routine
- ❖ Next instruction (first of interrupt handler) can be fetched



# Data Flow (Interrupt Diagram)





- ❖ Fetch accessing main memory
- ❖ Execution usually does not access main memory
- ❖ Can fetch next instruction during execution of current instruction
- ❖ Called instruction prefetch



# Improved Performance



- ❖ But not doubled:
  - Fetch usually shorter than execution
    - Prefetch more than one instruction?
  - Any jump or branch means that prefetched instructions are not the required instructions
- ❖ Add more stages to improve performance

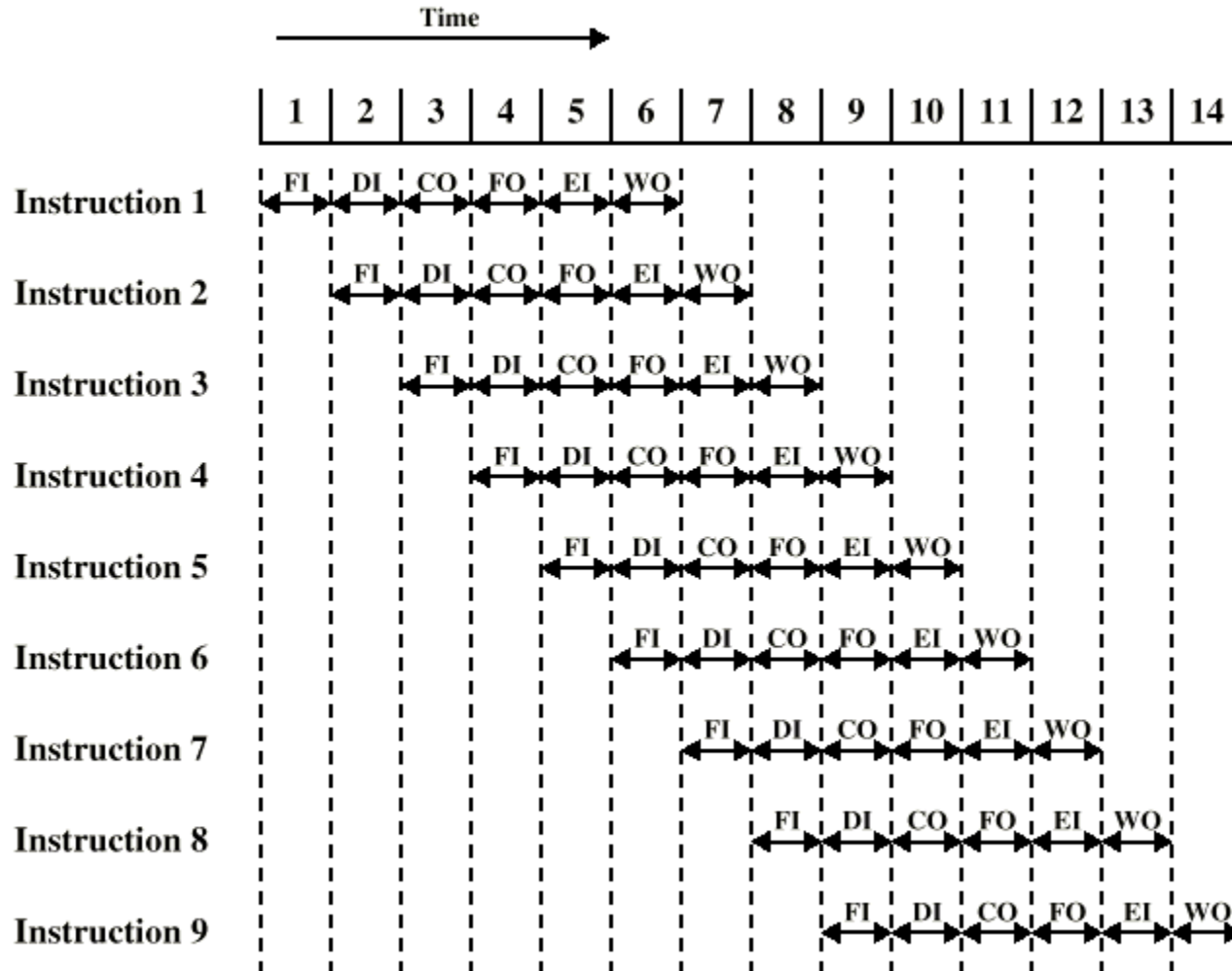




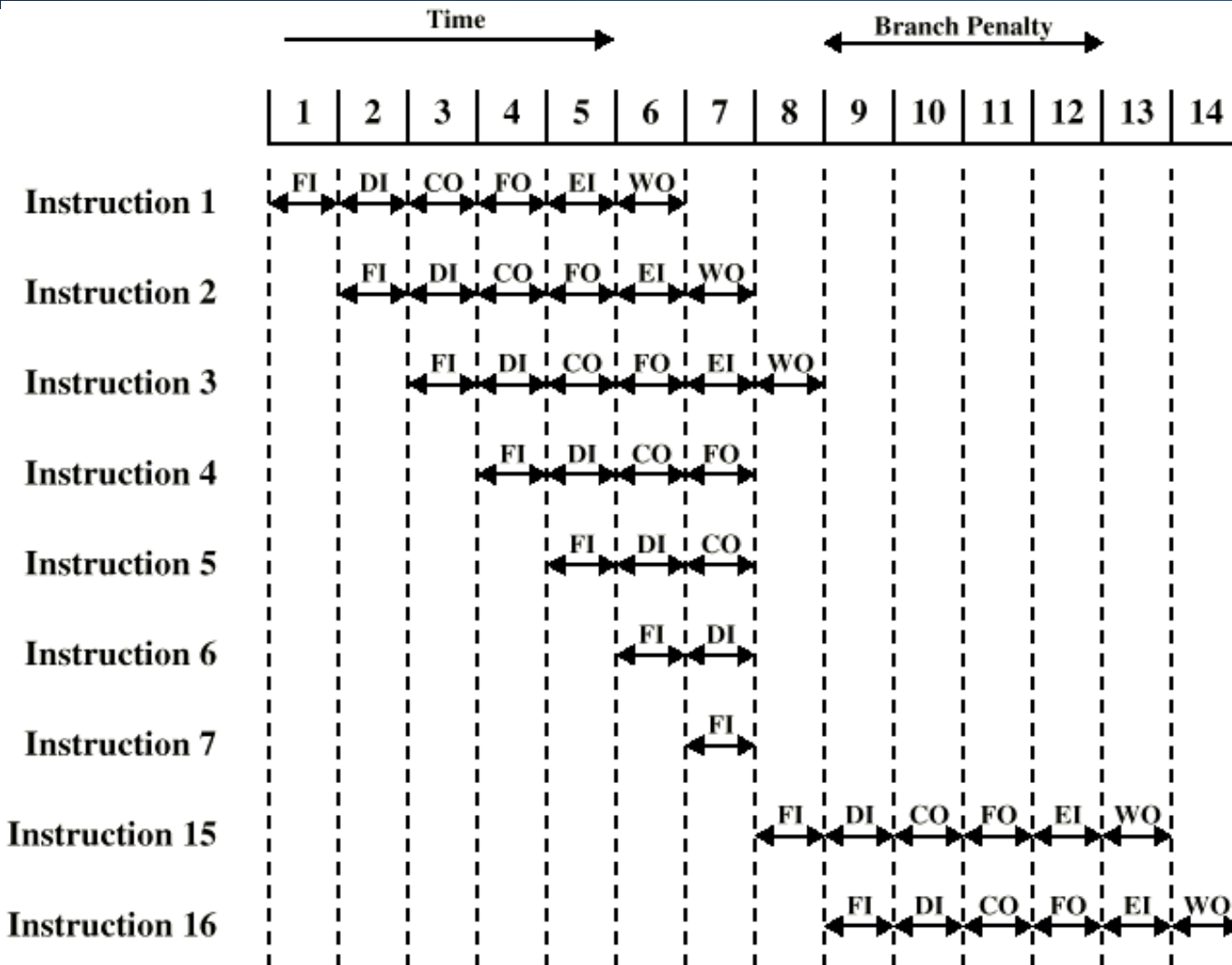
- ❖ Fetch instruction
- ❖ Decode instruction
- ❖ Calculate operands (i.e. EAs)
- ❖ Fetch operands
- ❖ Execute instructions
- ❖ Write result
  
- ❖ Overlap these operations



# Timing of Pipeline



# Branch in a Pipeline



# Dealing with Branches



- ❖ Multiple Streams
- ❖ Prefetch Branch Target
- ❖ Loop buffer
- ❖ Branch prediction
- ❖ Delayed branching





# Multiple Streams



- ❖ Have two pipelines
- ❖ Prefetch each branch into a separate pipeline
- ❖ Use appropriate pipeline
  
- ❖ Leads to bus & register contention
- ❖ Multiple branches lead to further pipelines being needed



# *Prefetch Branch Target*



- ❖ Target of branch is prefetched in addition to instructions following branch
- ❖ Keep target until branch is executed
- ❖ Used by IBM 360/91





- ❖ Very fast memory
- ❖ Maintained by fetch stage of pipeline
- ❖ Check buffer before fetching from memory
- ❖ Very good for small loops or jumps
- ❖ c.f. cache
- ❖ Used by CRAY-1



# Branch Prediction (1)



## ❖ Predict never taken

- Assume that jump will not happen
- Always fetch next instruction
- 68020 & VAX 11/780
- VAX will not prefetch after branch if a page fault would result (O/S v CPU design)

## ❖ Predict always taken

- Assume that jump will happen
- Always fetch target instruction





## ❖ Predict by Opcode

- Some instructions are more likely to result in a jump than others
- Can get up to 75% success

## ❖ Taken/Not taken switch

- Based on previous history
- Good for loops



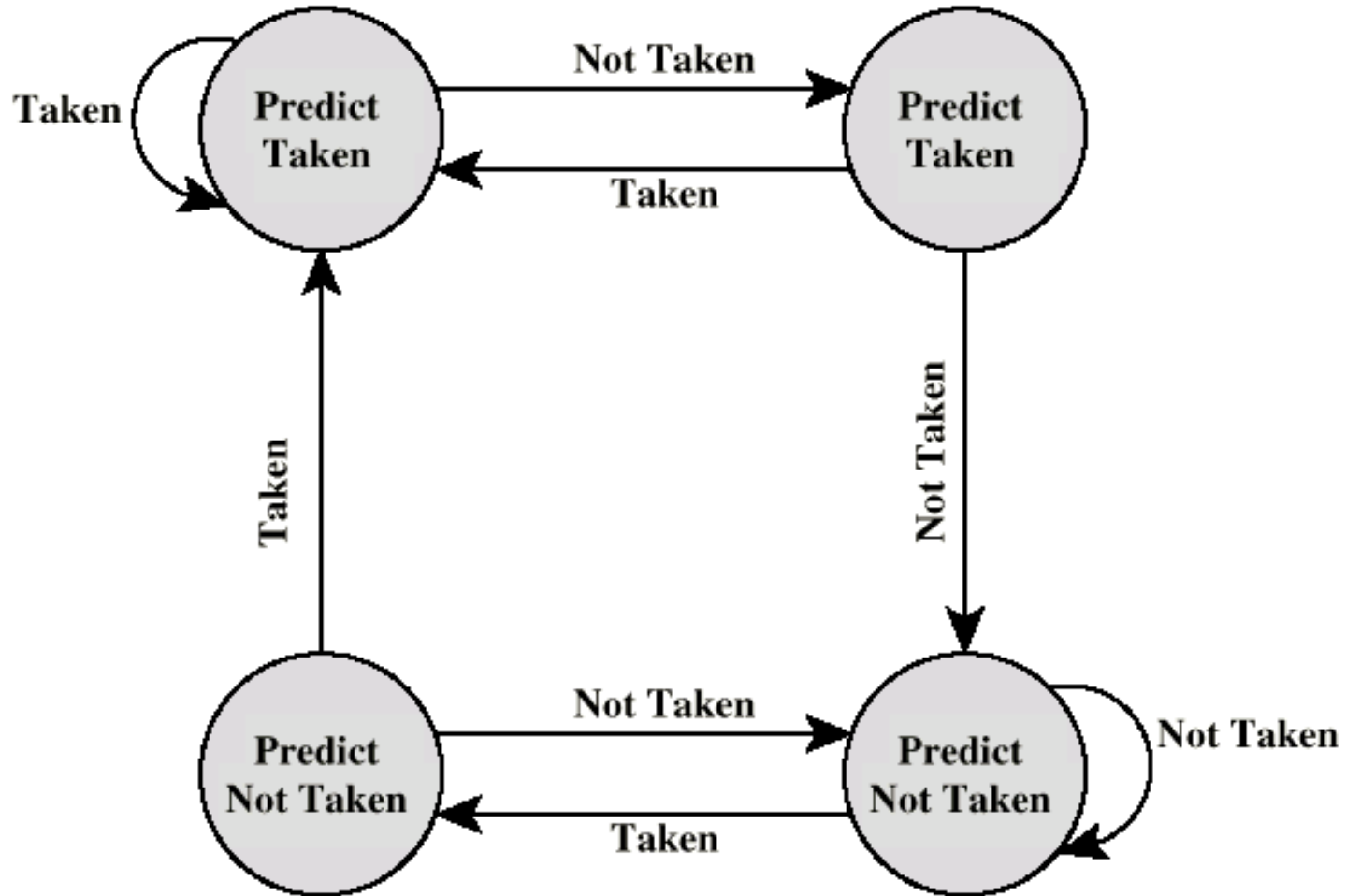


## ❖ Delayed Branch

- Do not take jump until you have to
- Rearrange instructions



# Branch Prediction State Diagram



# Foreground Reading



- ❖ Processor examples
- ❖ Stallings Chapter 11
- ❖ Web pages etc.

