William Stallings
Computer Organization and Architecture
7th Edition

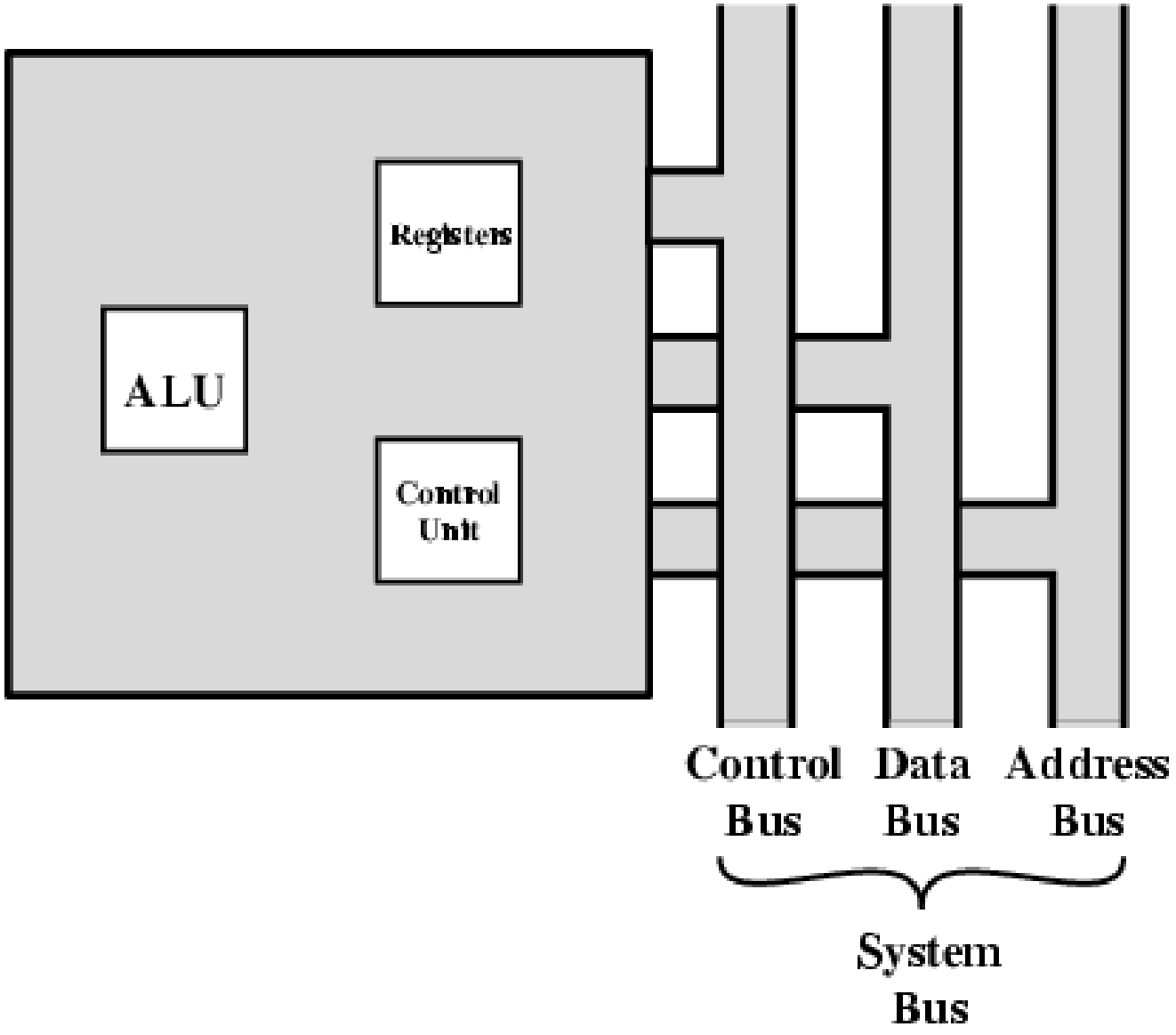
Chapter 12

CPU Structure and Function

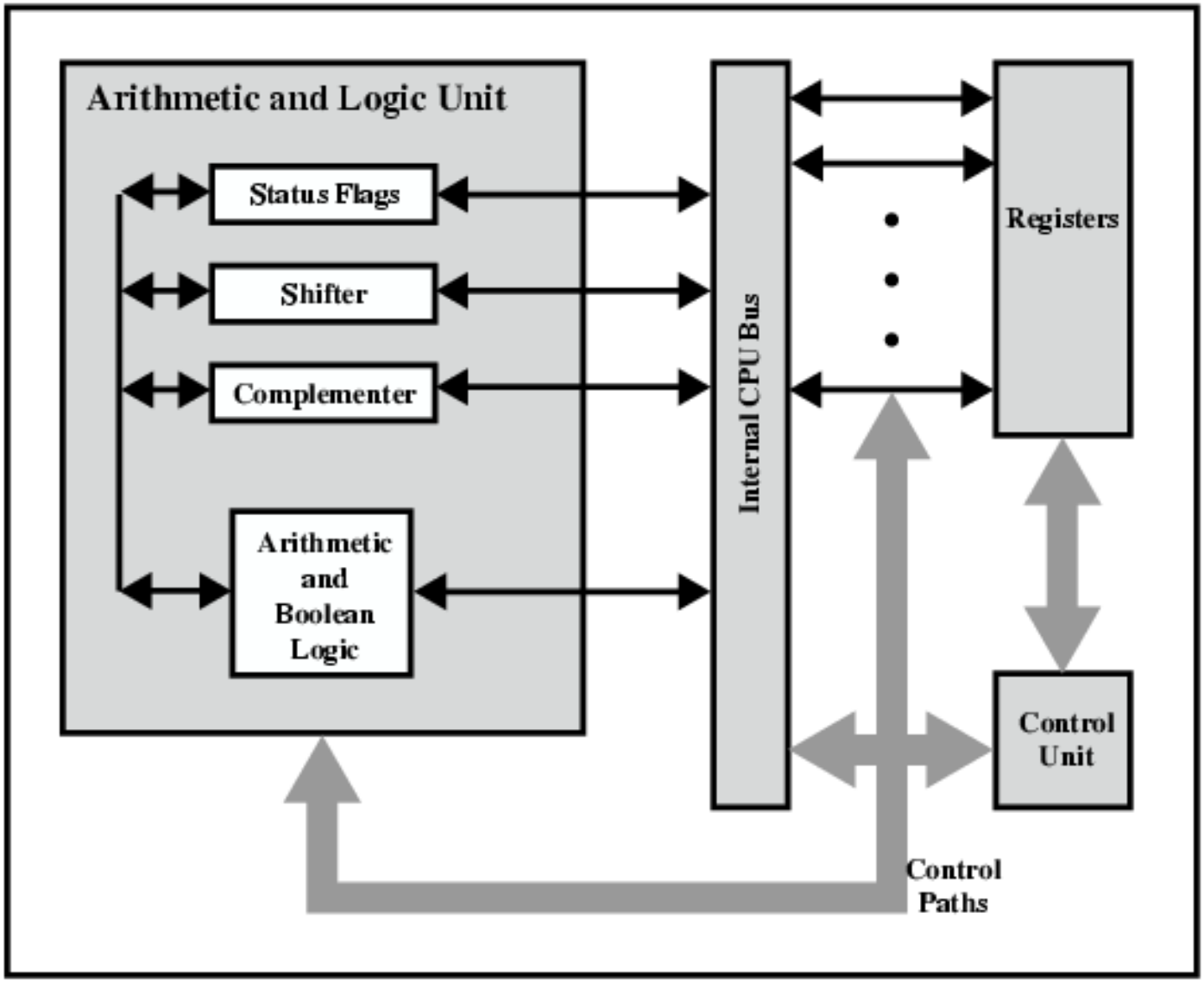
CPU Structure

- CPU must:
  - Fetch instructions
  - Interpret instructions
  - Fetch data
  - Process data
  - Write data

CPU With Systems Bus



CPU Internal Structure



Registers

- 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

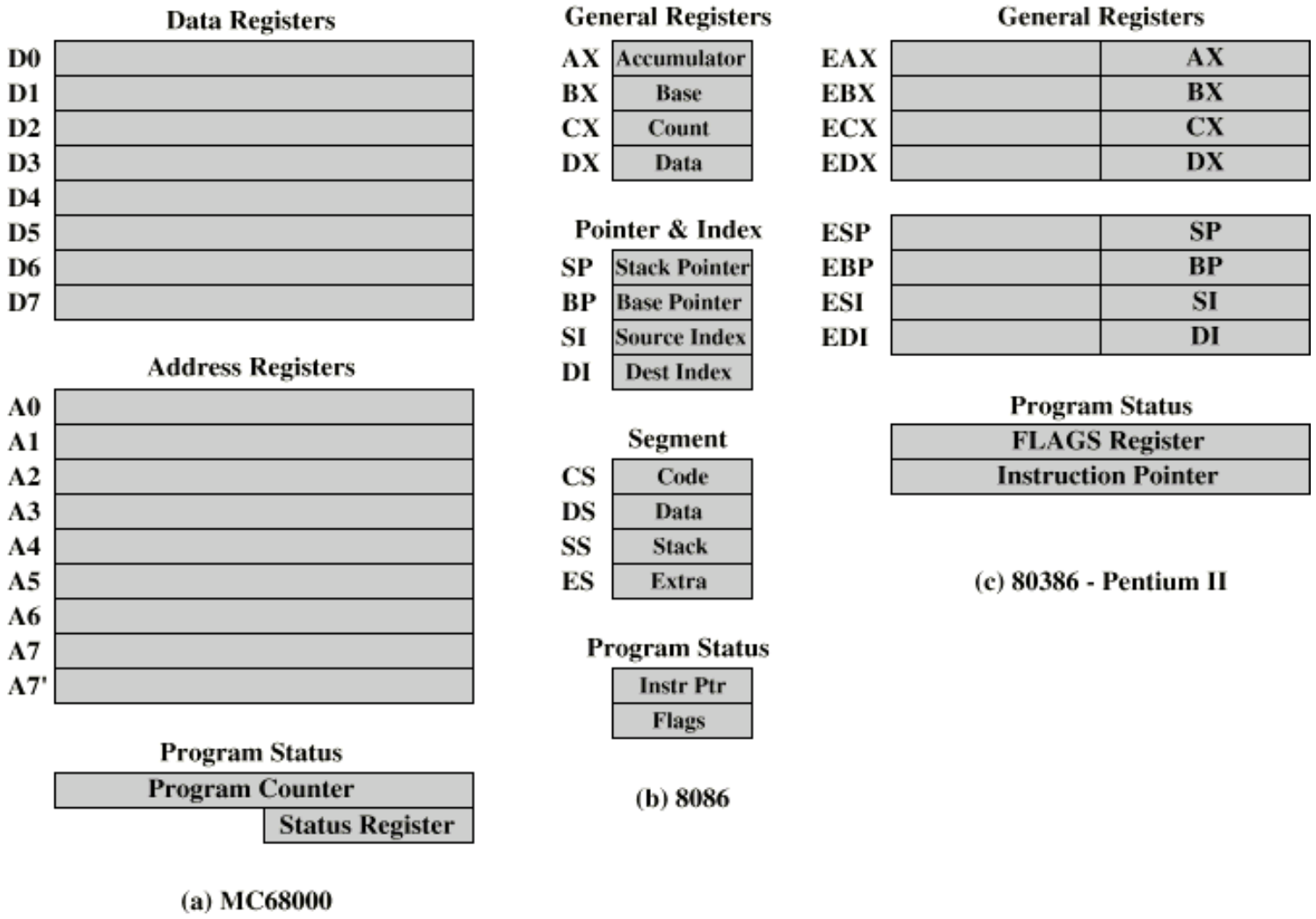
- Intel ring zero
- Kernel mode
- Allows privileged instructions to execute
- Used by operating system
- Not available to user programs

Other Registers

- 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



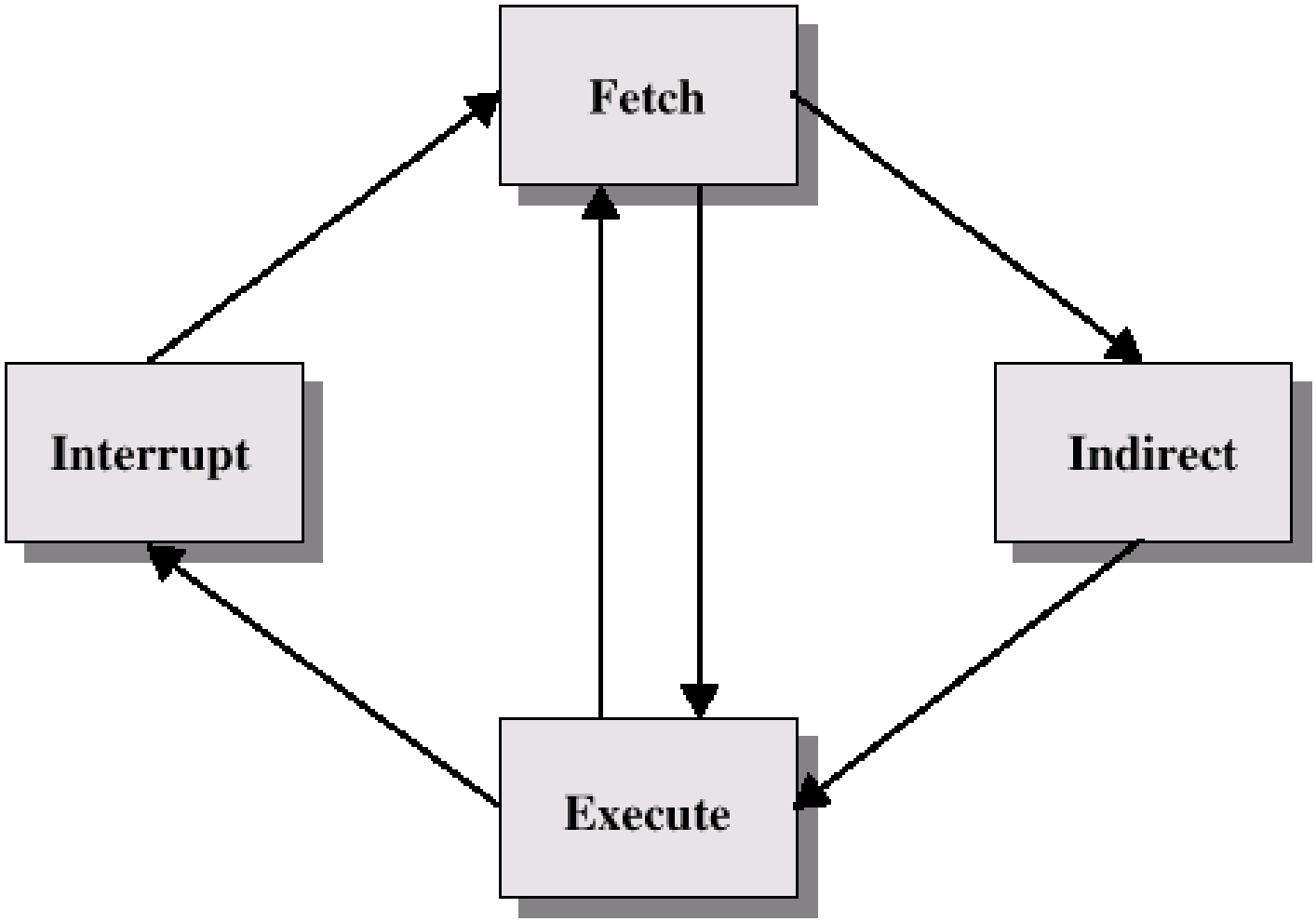
Instruction Cycle

- Revision
- Stallings Chapter 3

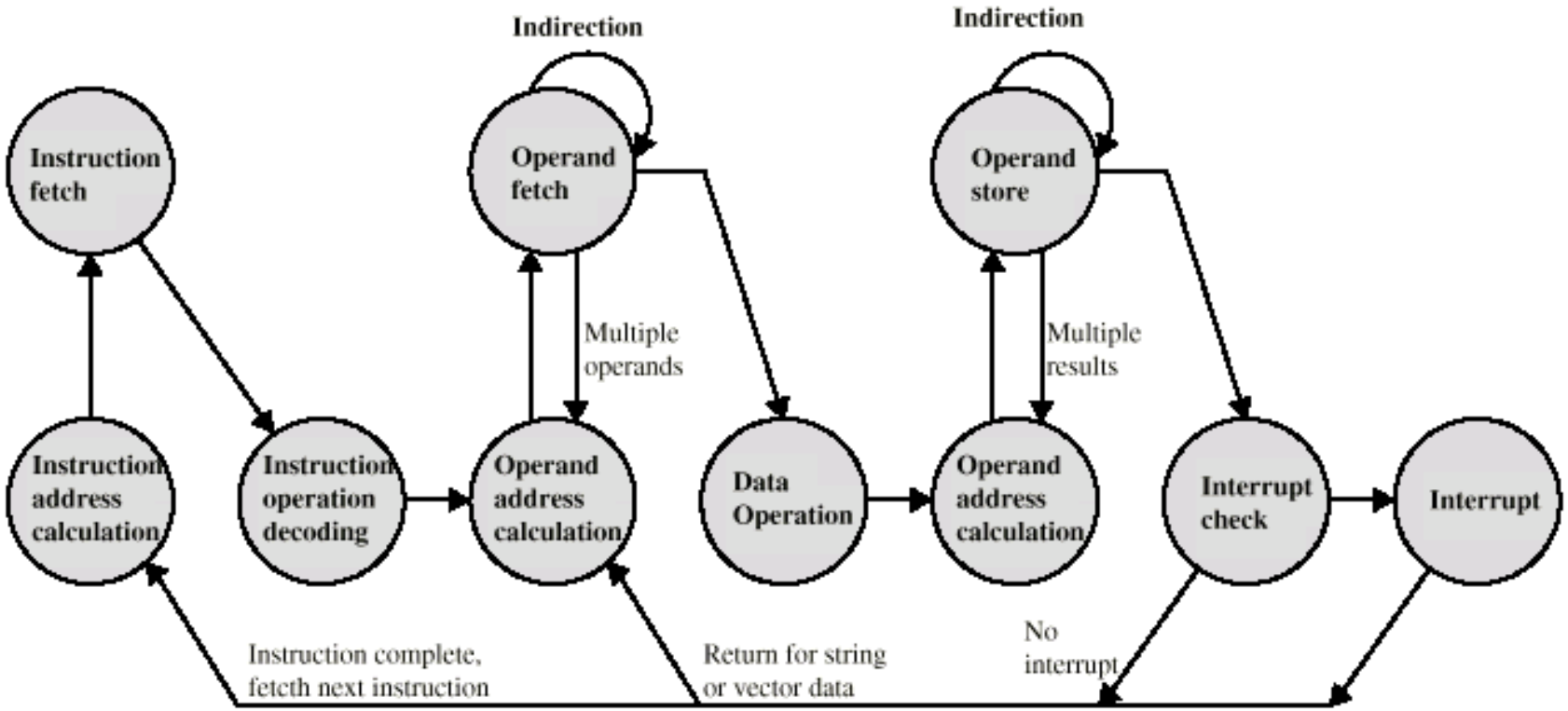
Indirect Cycle

- 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



Instruction Cycle State Diagram



Data Flow (Instruction Fetch)

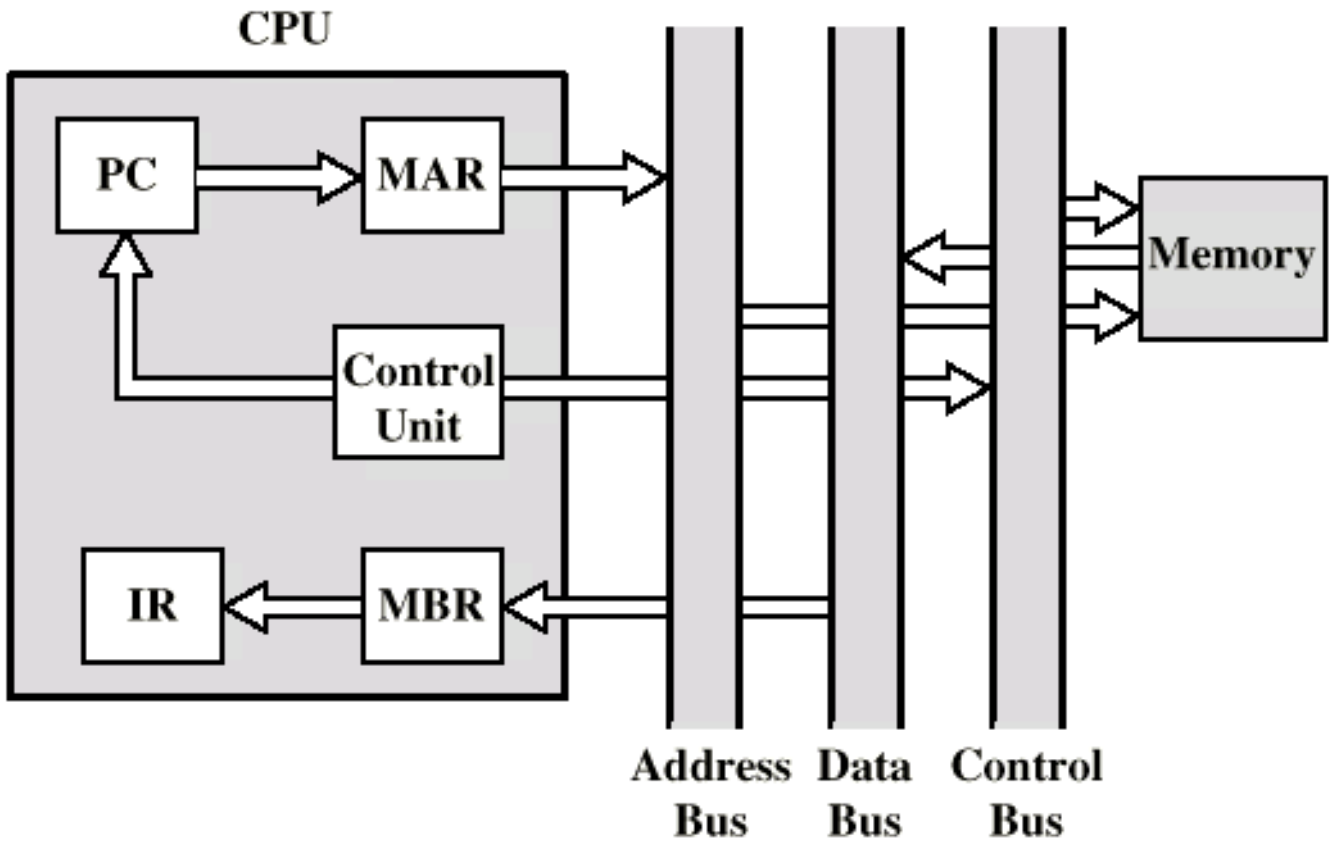
- Depends on CPU design
- In general:

- 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)

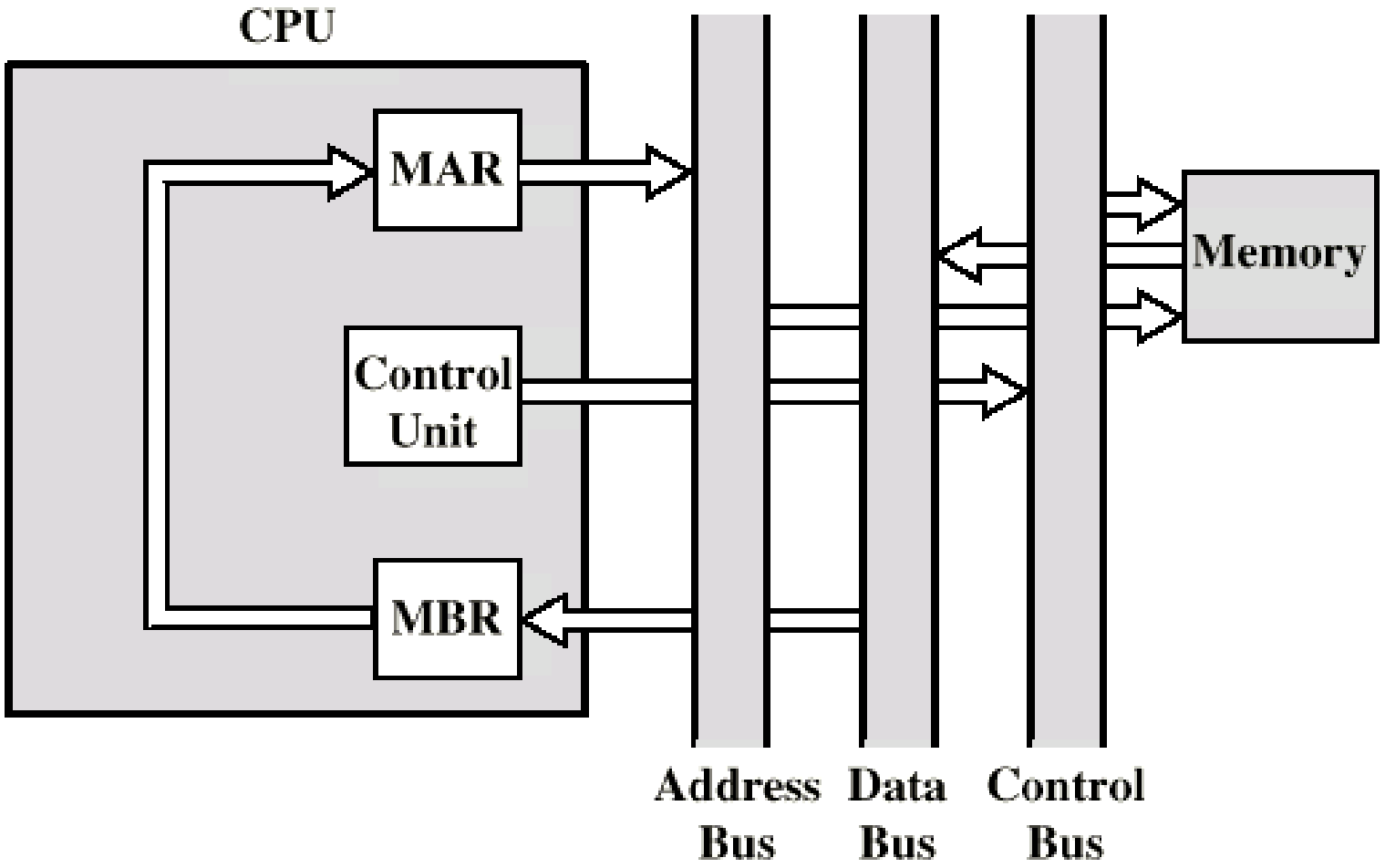
- IR is examined
- If indirect addressing, indirect cycle is performed
  - —Right most N bits of MBR transferred to MAR
  - —Control unit requests memory read
  - —Result (address of operand) moved to MBR

Data Flow (Fetch Diagram)



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

Data Flow (Indirect Diagram)



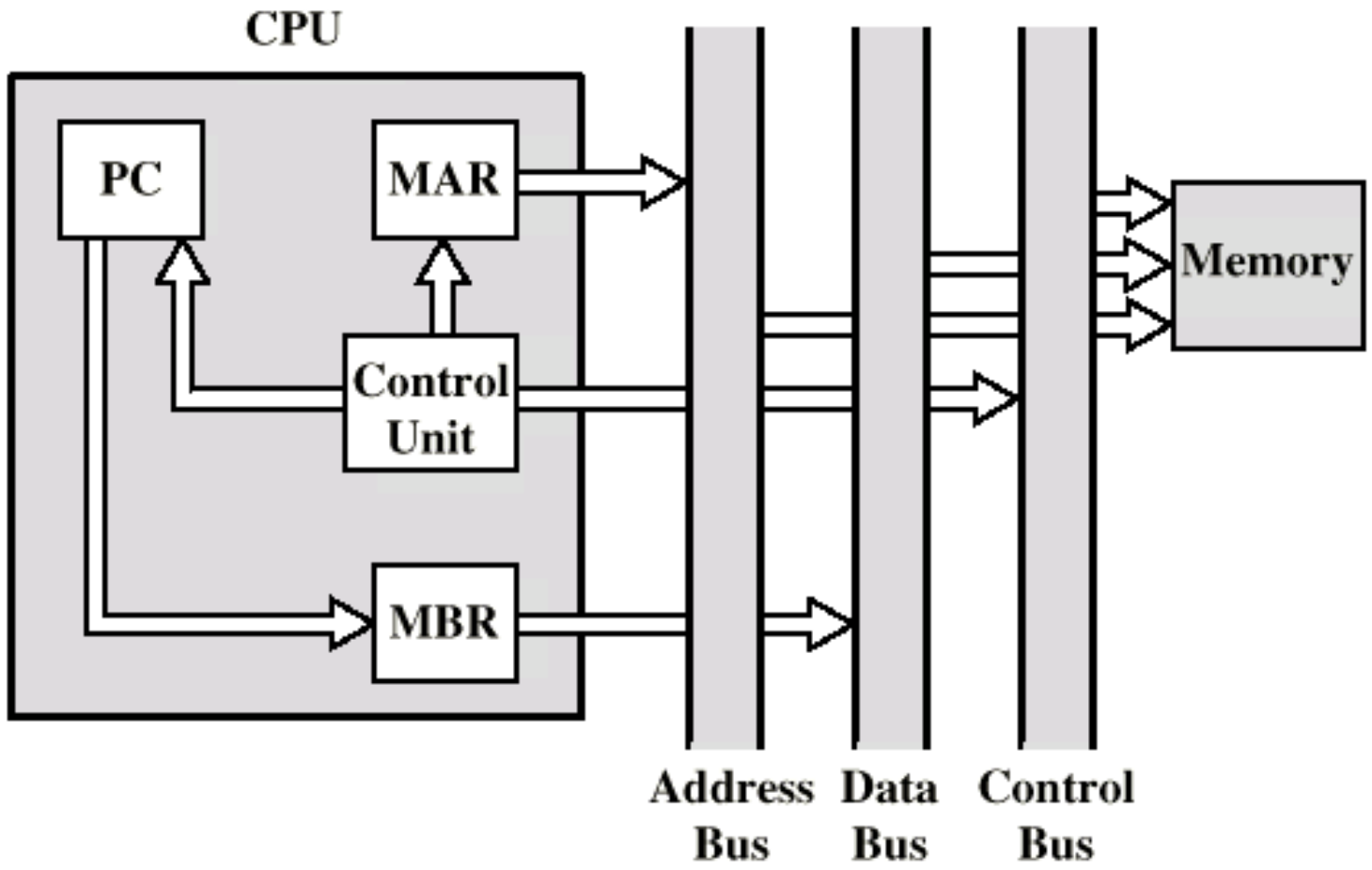
Data Flow (Execute)

- 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)



Prefetch

- 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

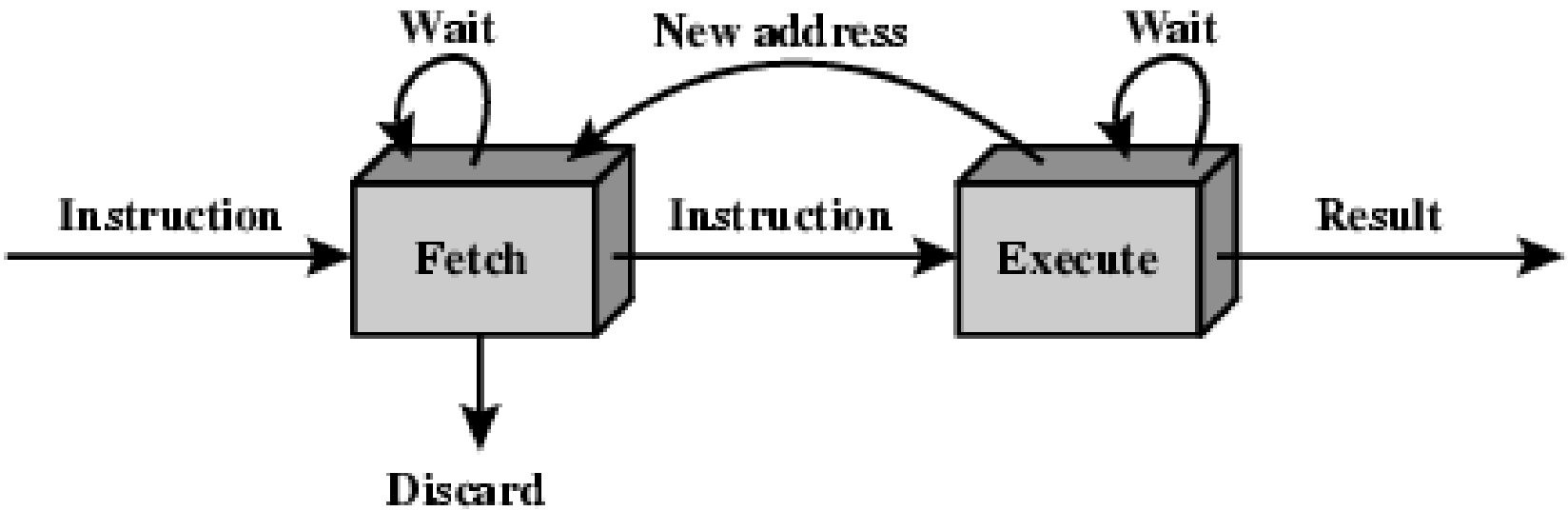
Pipelining

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

- Overlap these operations

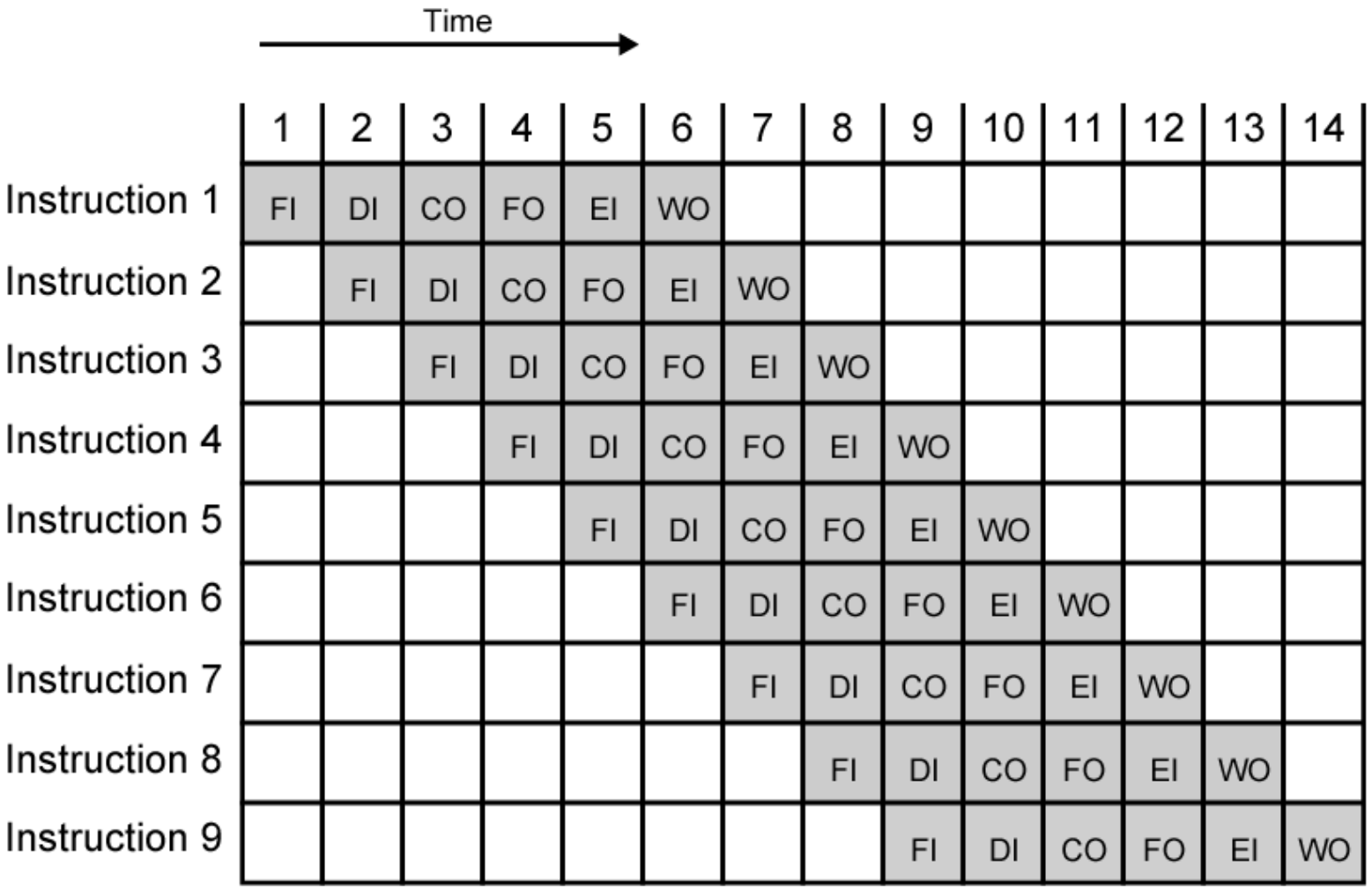
Two Stage Instruction Pipeline

(a) Simplified view

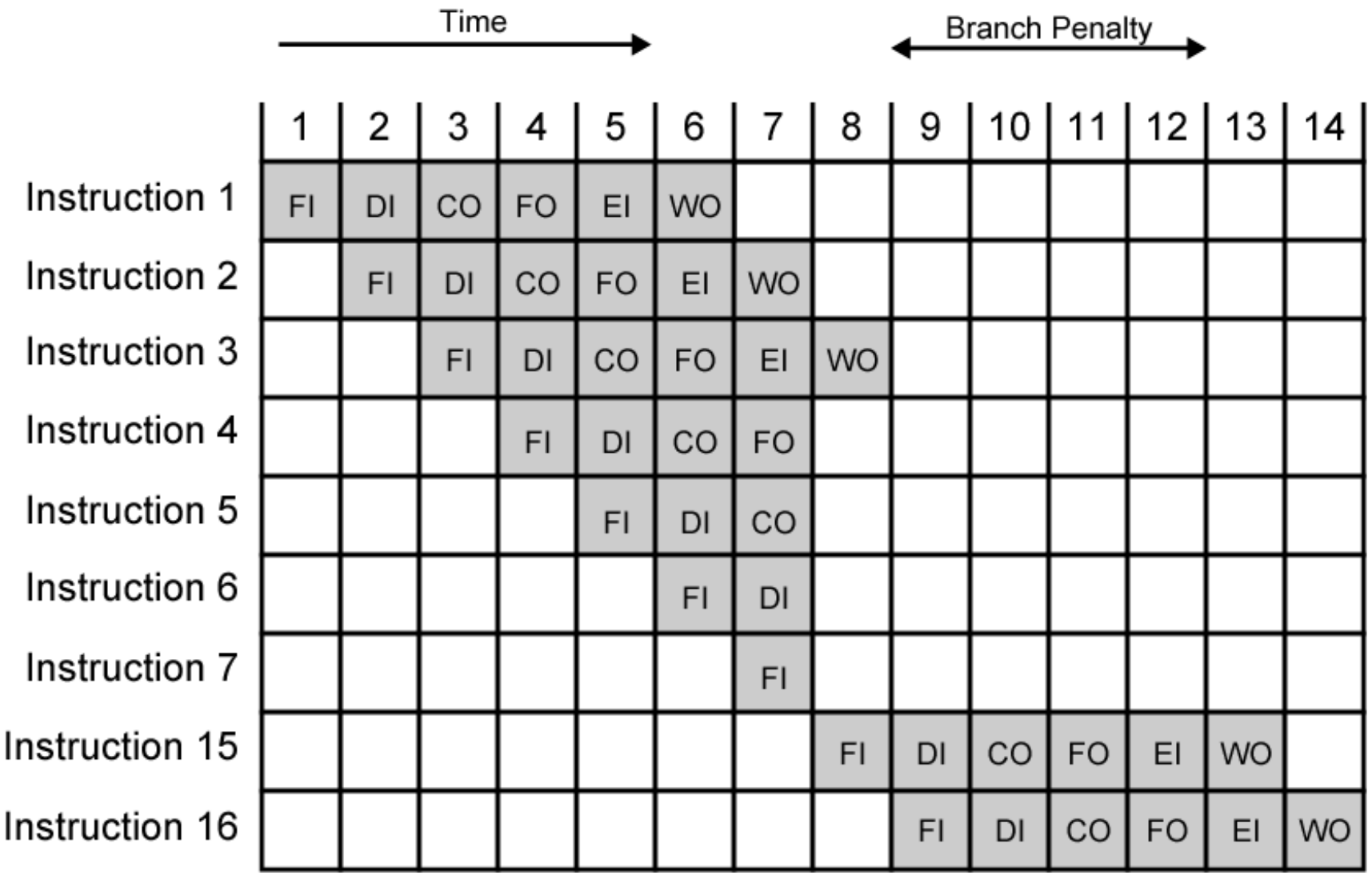


(b) Expanded view

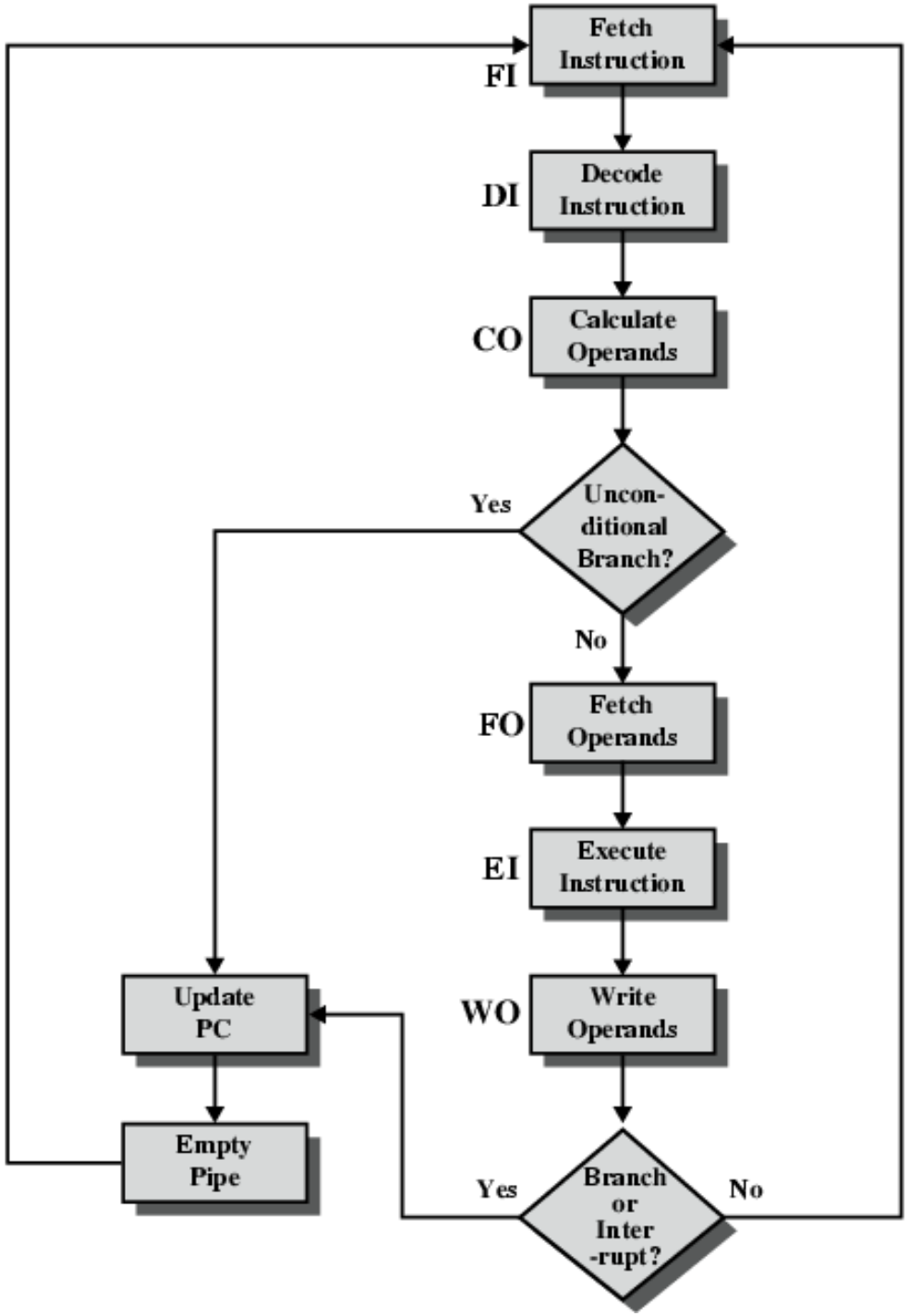
Timing Diagram for Instruction Pipeline Operation



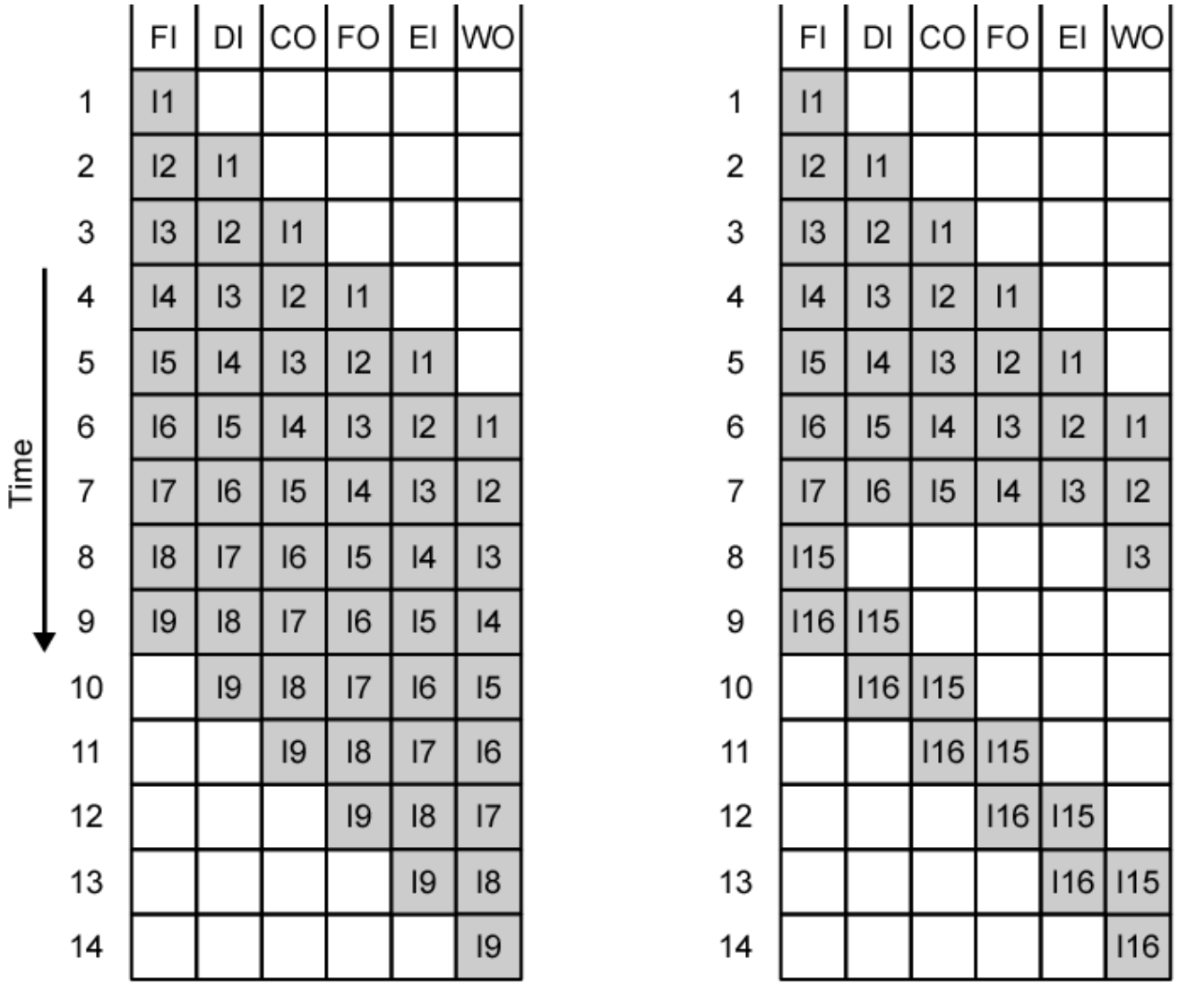
The Effect of a Conditional Branch on Instruction Pipeline Operation



Six Stage Instruction Pipeline



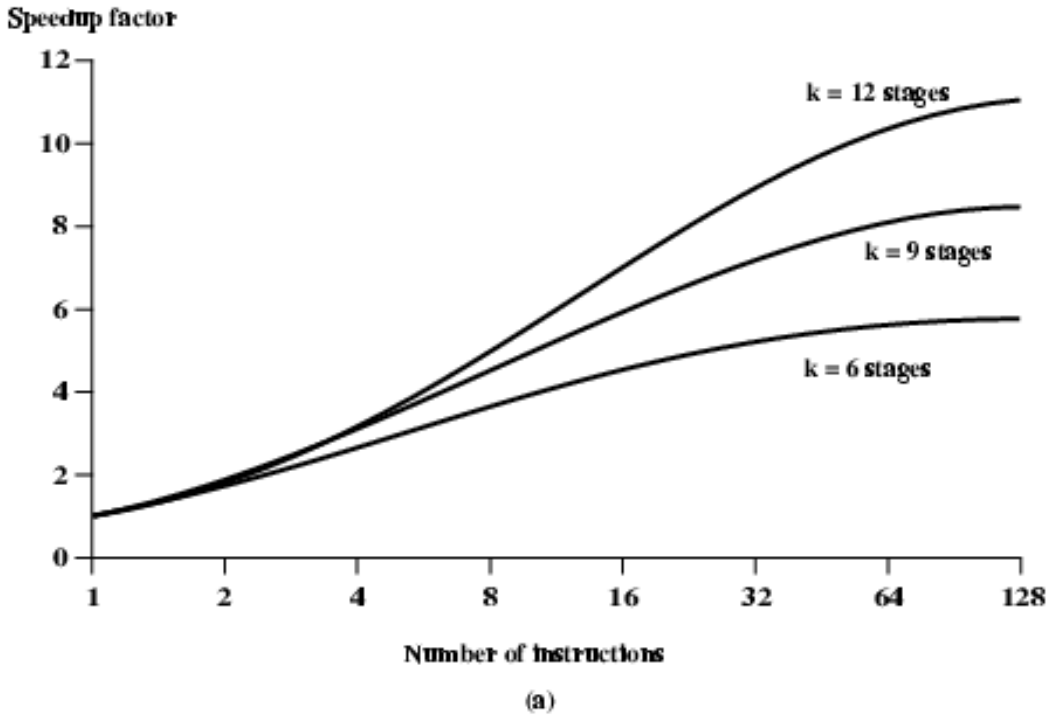
Alternative Pipeline Depiction



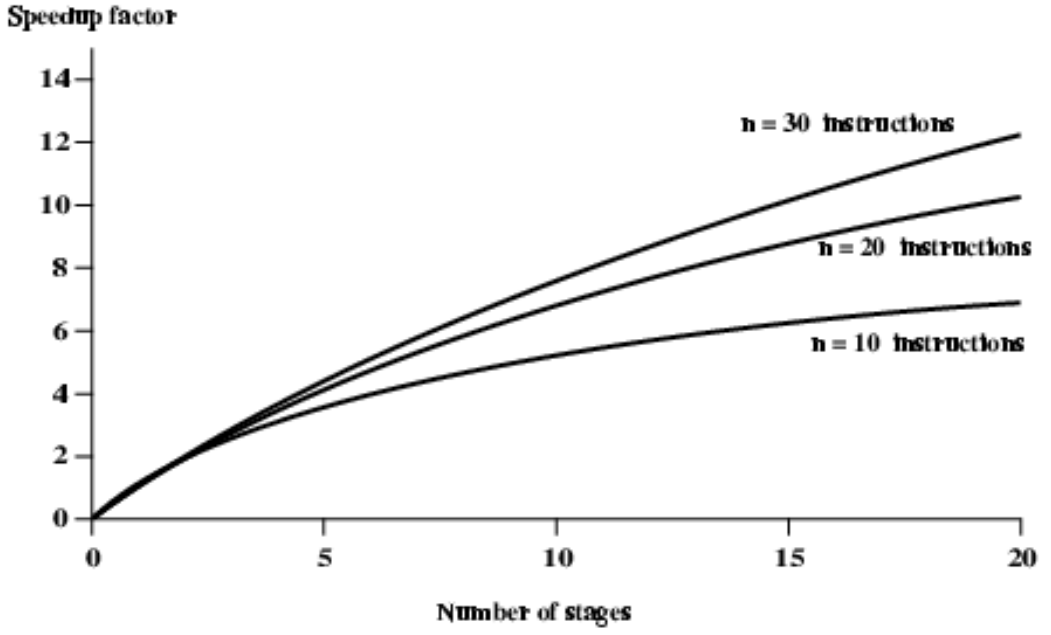
(a) No branches

(b) With conditional branch

Speedup Factors with Instruction Pipelining



(a)



(b)

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

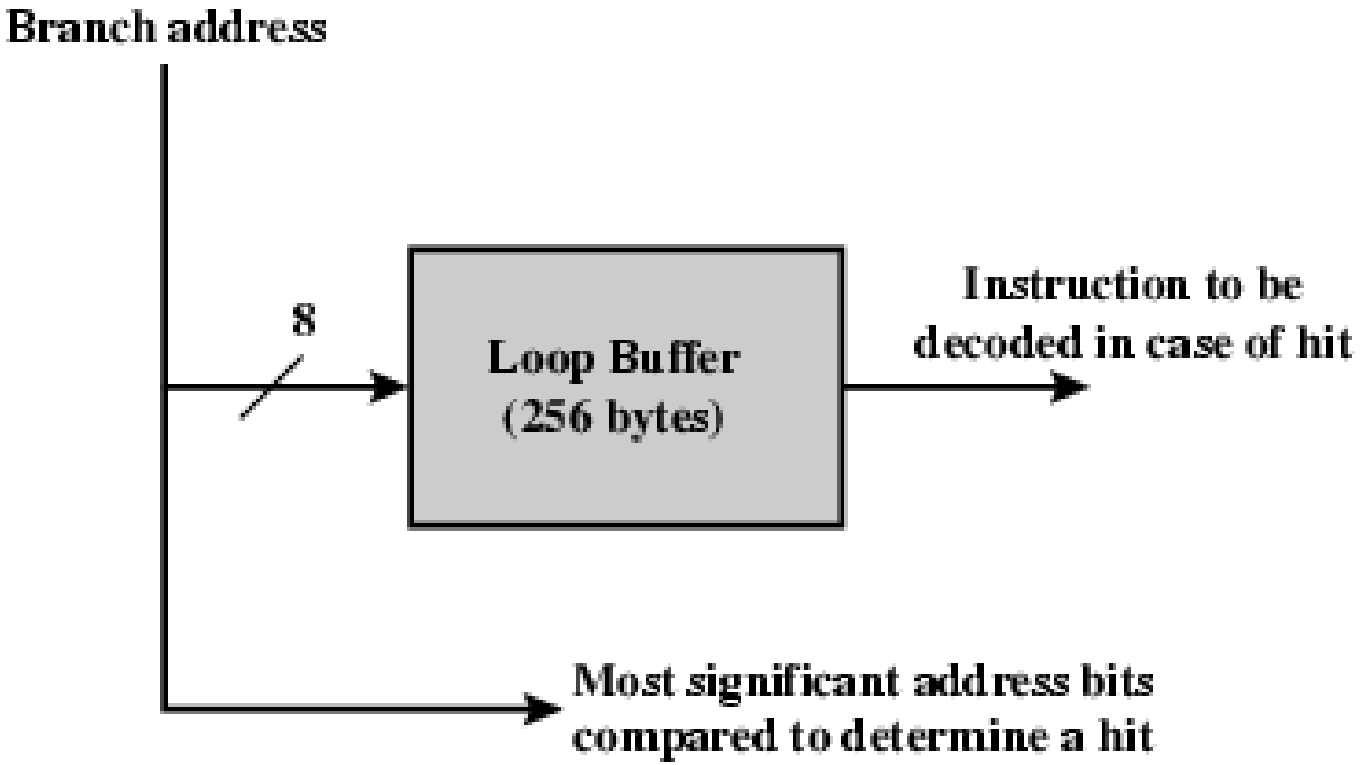
Prefetch Branch Target

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

Loop Buffer

- 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

Loop Buffer Diagram



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

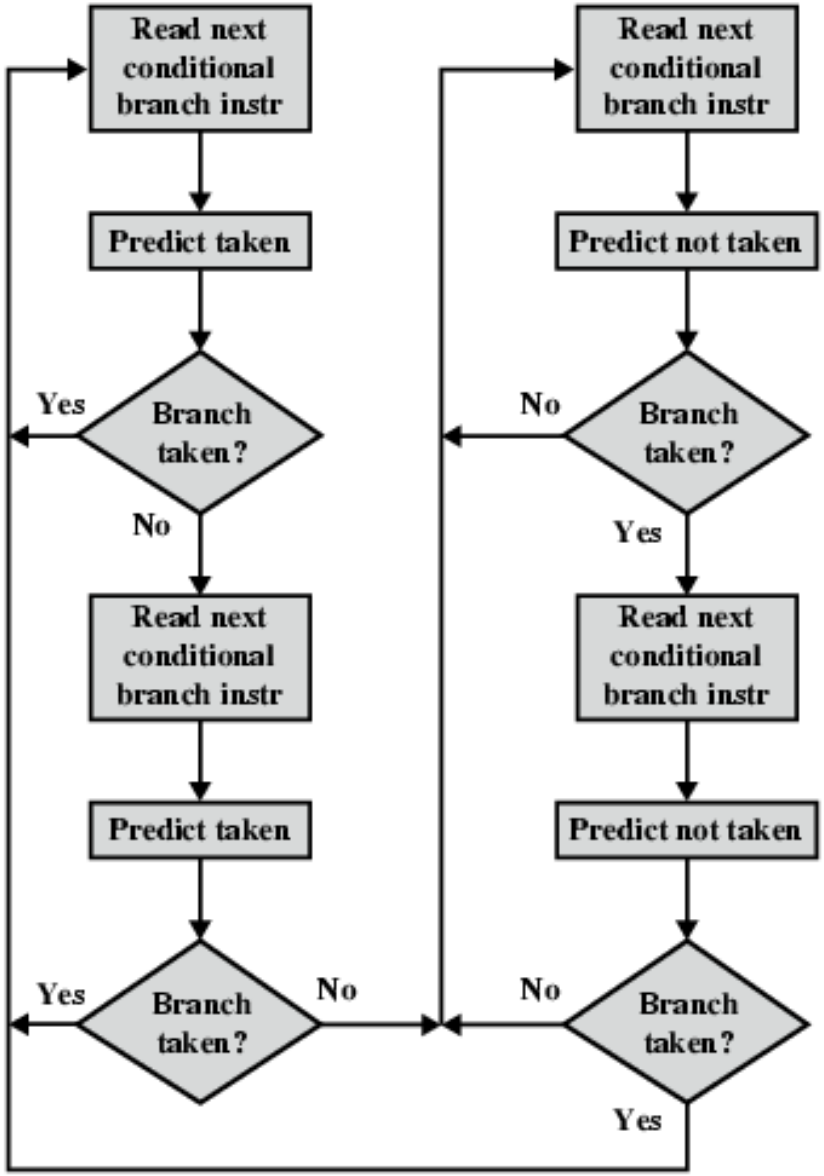
Branch Prediction (2)

- Predict by Opcode
  - —Some instructions are more likely to result in a jump than thers
  - —Can get up to 75% success
- Taken/Not taken switch
  - —Based on previous history
  - —Good for loops

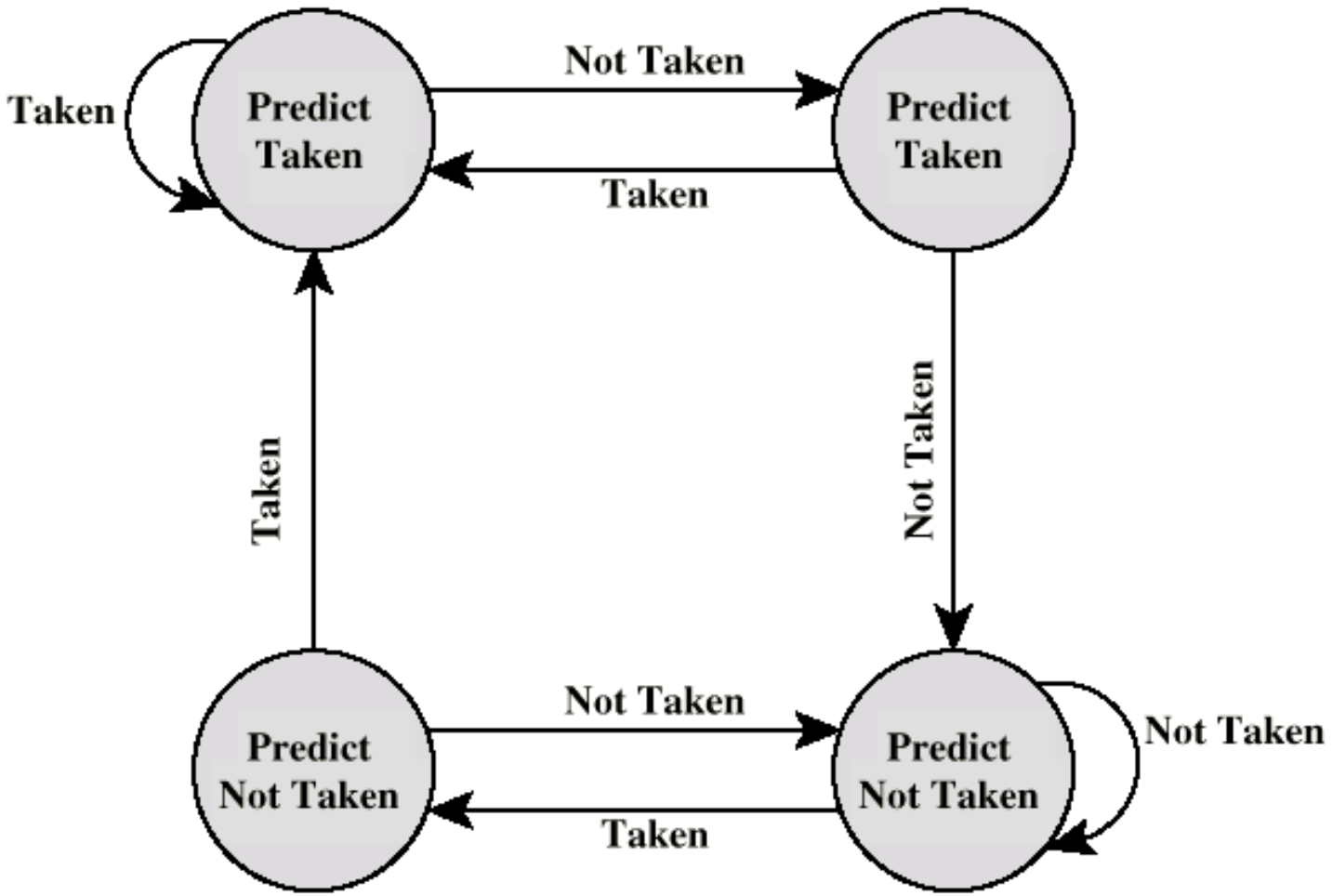
Branch Prediction (3)

- Delayed Branch
  - —Do not take jump until you have to
  - —Rearrange instructions

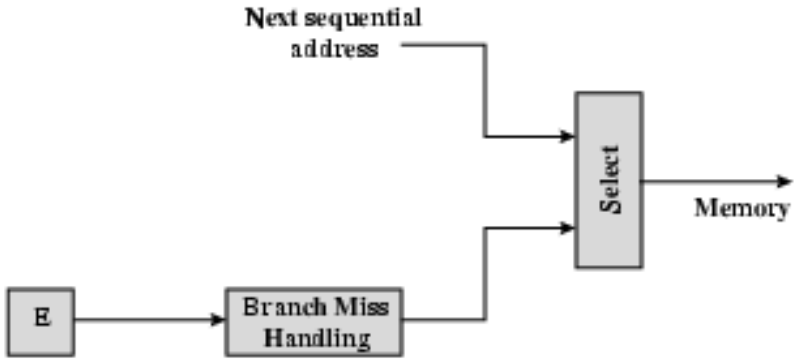
Branch Prediction Flowchart



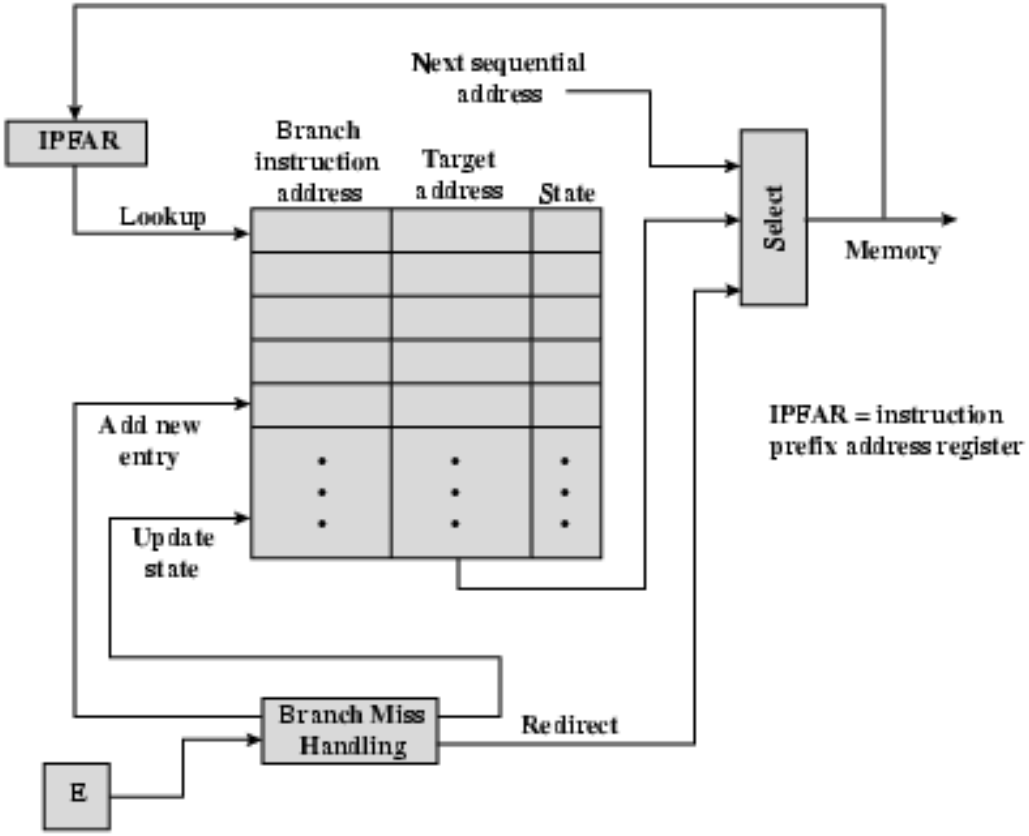
Branch Prediction State Diagram



Dealing With Branches



(a) Predict never taken strategy

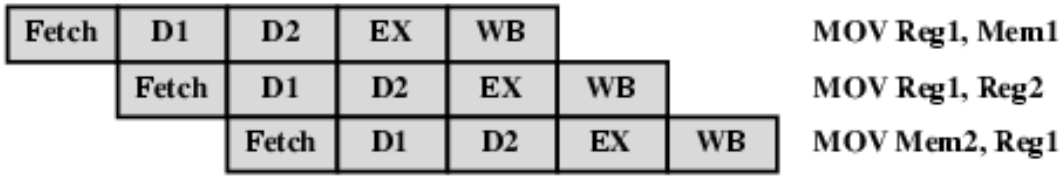


(b) Branch history table strategy

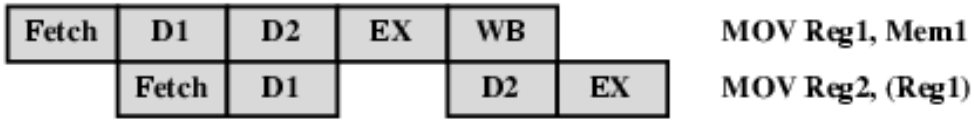
Intel 80486 Pipelining

- Fetch
  - From cache or external memory
  - Put in one of two 16-byte prefetch buffers
  - Fill buffer with new data as soon as old data consumed
  - Average 5 instructions fetched per load
  - Independent of other stages to keep buffers full
- Decode stage 1
  - Opcode & address-mode info
  - At most first 3 bytes of instruction
  - Can direct D2 stage to get rest of instruction
- Decode stage 2
  - Expand opcode into control signals
  - Computation of complex address modes
- Execute
  - ALU operations, cache access, register update
- Writeback
  - Update registers & flags
  - Results sent to cache & bus interface write buffers

80486 Instruction Pipeline Examples



(a) No Data Load Delay in the Pipeline



(b) Pointer Load Delay

(c) Branch Instruction Timing

Pentium 4 Registers

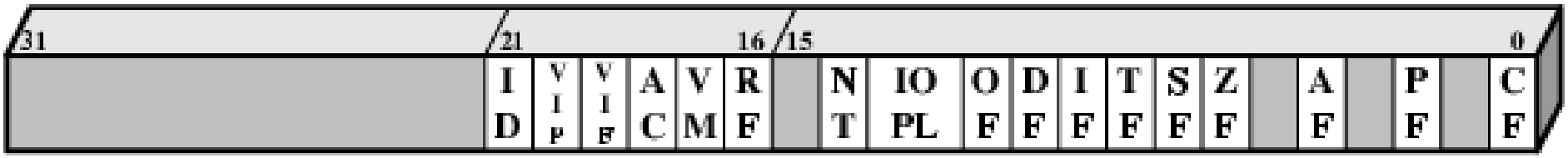
(a) Integer Unit

Type	Number	Length (bits)	Purpose
General	8	32	General-purpose user registers
Segment	6	16	Contain segment selectors
Flags	1	32	Status and control bits
Instruction Pointer	1	32	Instruction pointer

(b) Floating-Point Unit

Type	Number	Length (bits)	Purpose
Numeric	8	80	Hold floating-point numbers
Control	1	16	Control bits
Status	1	16	Status bits
Tag Word	1	16	Specifies contents of numeric registers
Instruction Pointer	1	+8	Points to instruction interrupted by exception
Data Pointer	1	+8	Points to operand interrupted by exception

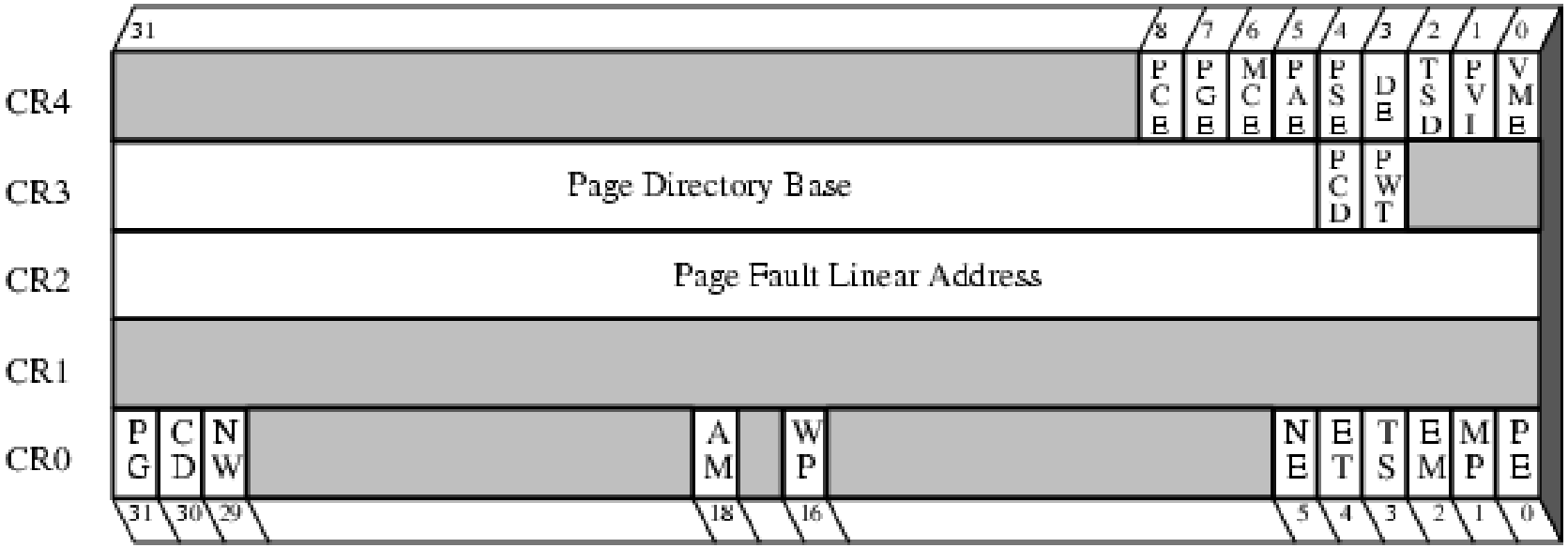
EFLAGS Register



ID = Identification flag
VIP = Virtual interrupt pending
VIF = Virtual interrupt flag
AC = Alignment check
VM = Virtual 8086 mode
RF = Resume flag
NT = Nested task flag
IOPL = I/O privilege level
OF = Overflow flag

DF = Direction flag
IF = Interrupt enable flag
TF = Trap flag
SF = Sign flag
ZF = Zero flag
AF = Auxiliary carry flag
PF = Parity flag
CF = Carry flag

Control Registers

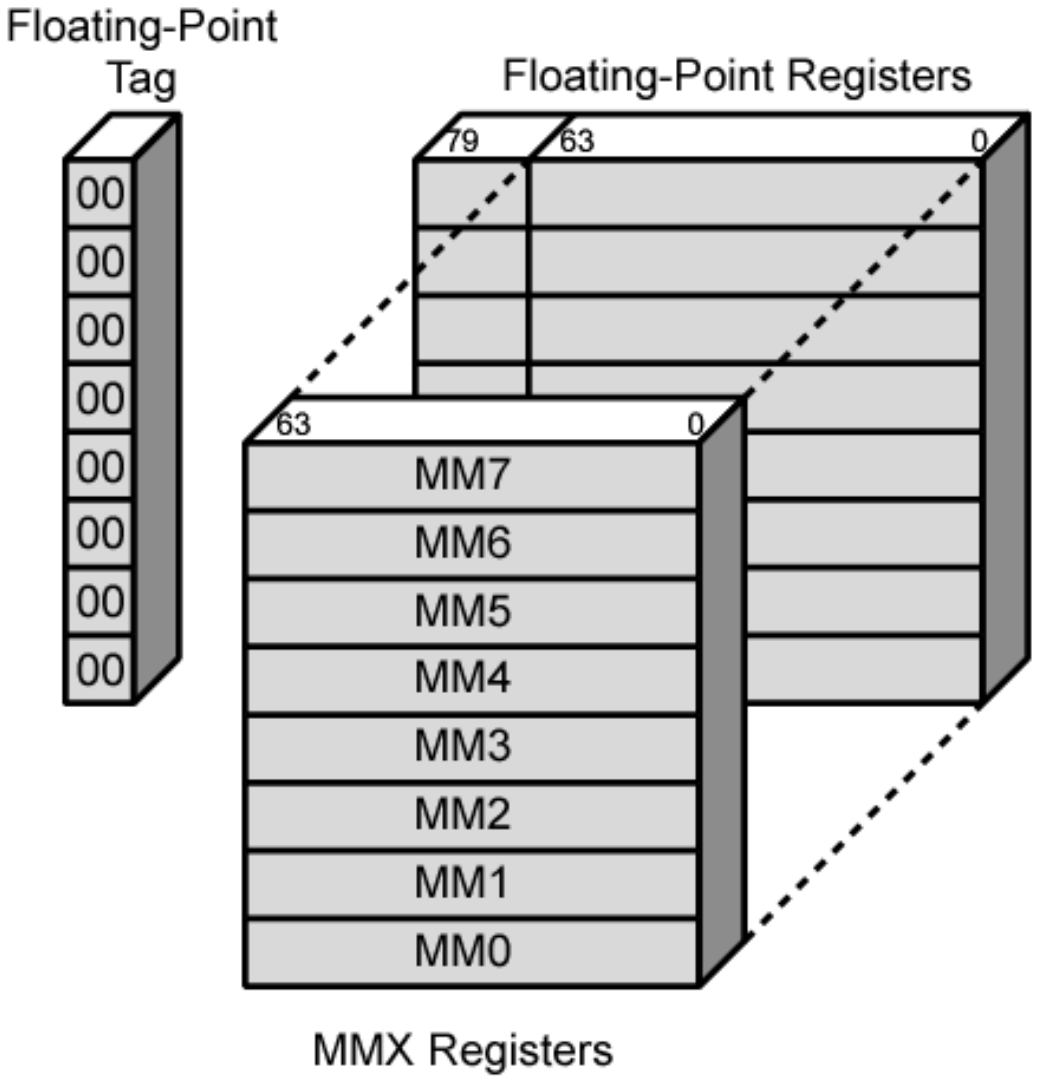


PCE	=	Performance Counter Enable	PG	=	Paging
PGE	=	Page Global Enable	CD	=	Cache Disable
MCE	=	Machine Check Enable	NW	=	Not Write Through
PAE	=	Physical Address Extension	AM	=	Alignment Mask
PSE	=	Page Size Extensions	WP	=	Write Protect
DE	=	Debug Extensions	NE	=	Numeric Error
TSD	=	Time Stamp Disable	ET	=	Extension Type
PVI	=	Protected Mode Virtual Interrupt	TS	=	Task Switched
VME	=	Virtual 8086 Mode Extensions	EM	=	Emulation
PCD	=	Page-level Cache Disable	MP	=	Monitor Coprocessor
PWT	=	Page-level Writes Transparent	PE	=	Protection Enable

MMX Register Mapping

- MMX uses several 64 bit data types
- Use 3 bit register address fields
  - —8 registers
- No MMX specific registers
  - —Aliasing to lower 64 bits of existing floating point registers

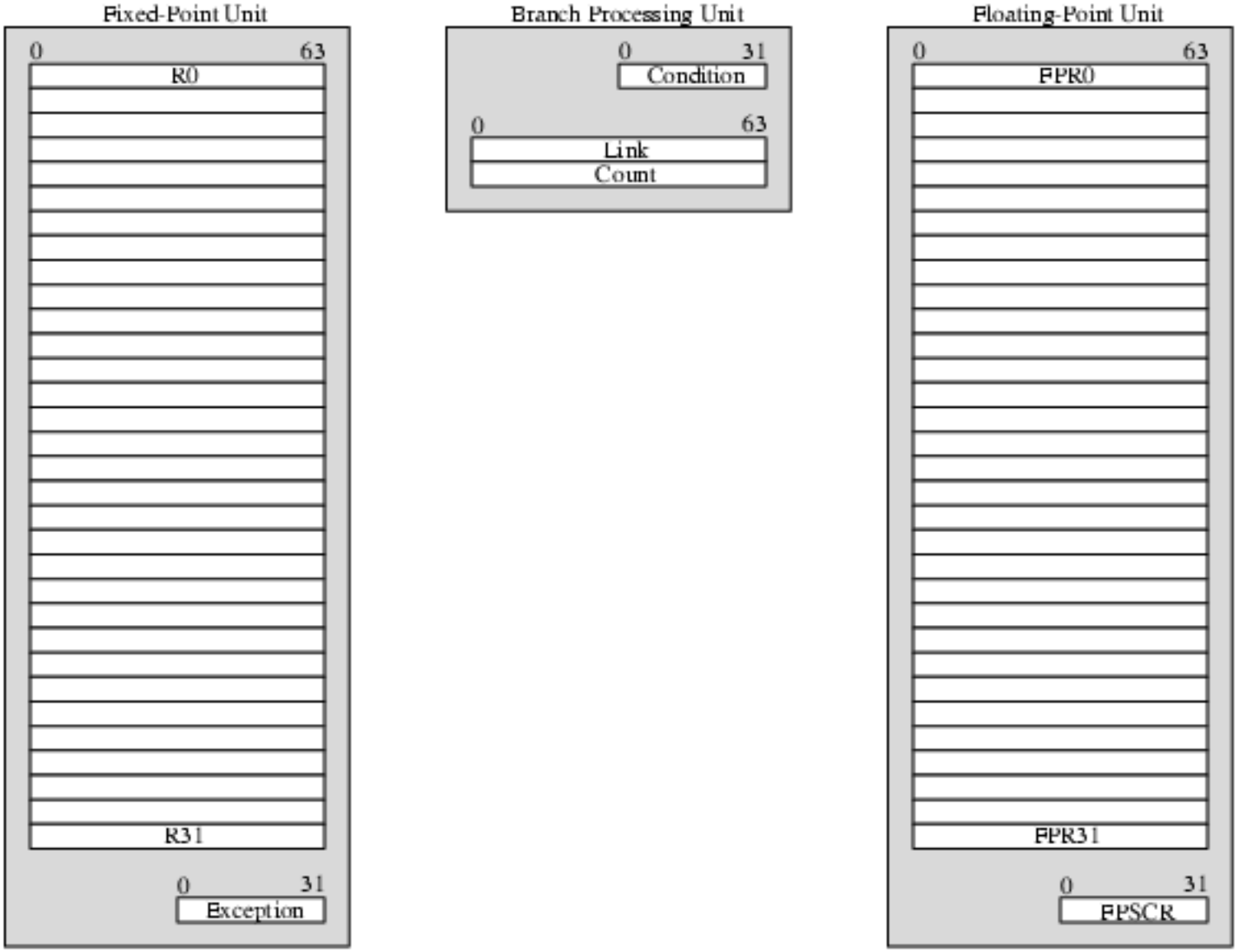
Mapping of MMX Registers to Floating-Point Registers



Pentium Interrupt Processing

- Interrupts
  - —Maskable
  - —Nonmaskable
- Exceptions
  - —Processor detected
  - —Programmed
- Interrupt vector table
  - —Each interrupt type assigned a number
  - —Index to vector table
  - —256 * 32 bit interrupt vectors
- 5 priority classes

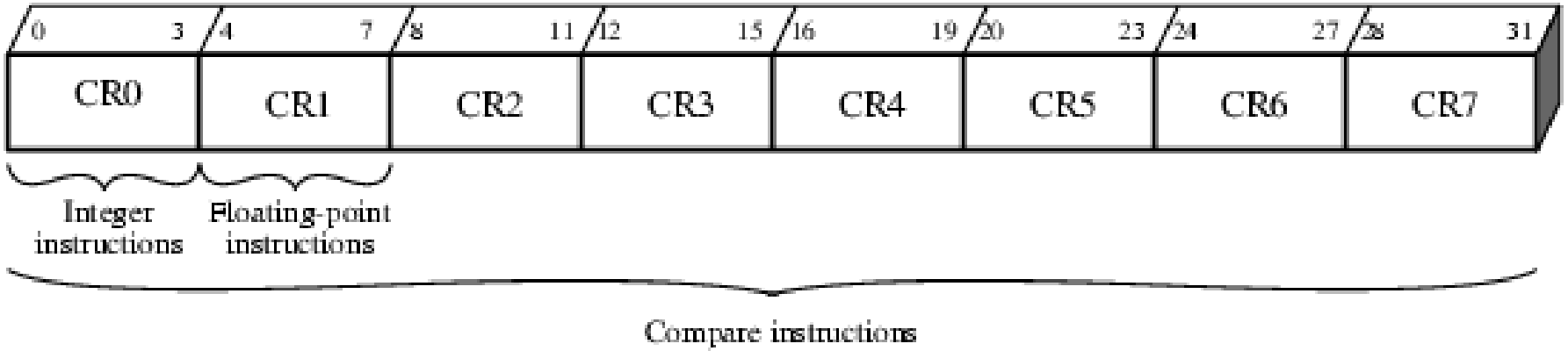
PowerPC User Visible Registers



PowerPC Register Formats

SO = Summary overflow: set to 1 to indicate an overflow occurred during the exection of an instruction; remains 1 until reset by software
OV = Overflow: set to 1 to indicate an overflow occurred during the exection of an instruction; res to 0 by next instruction if there is no overflow
CA = Carry: set to 1 to indicate carry out of bit 0 during the execution of an instruction
Byte Count = Specifies number of bytes to be transferred by Load/Store String indexed instruction

(a) Fixed-Point Exception Register (XER)



(b) Condition Register

Foreground Reading

- Processor examples
- Stallings Chapter 12
- Manufacturer web sites & specs

- Arrsitektur cpu-32bit