Computer Architecture

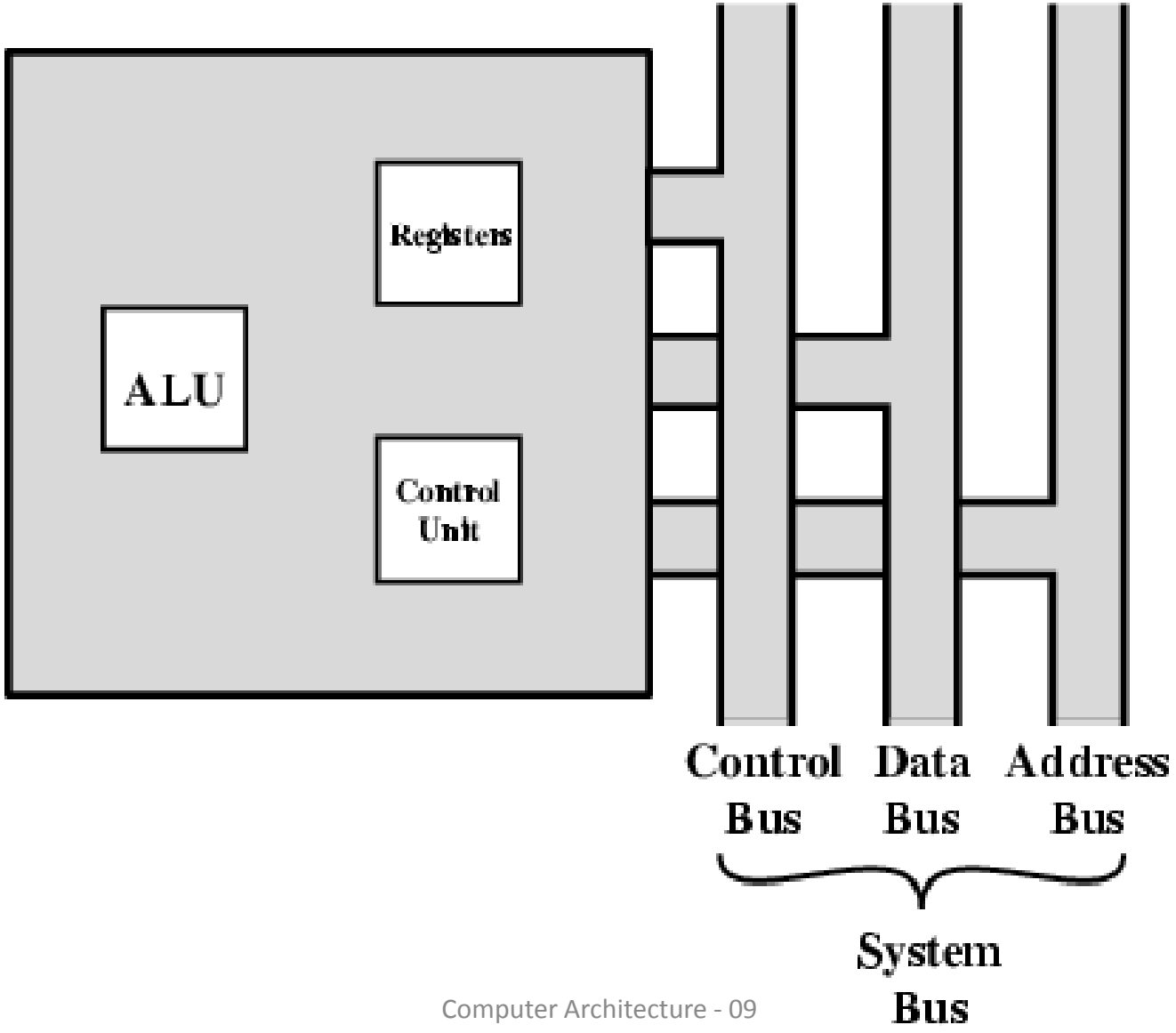
Processor Structure and Functions

ARASH HABIBI LASHKARI
(April- 2010)

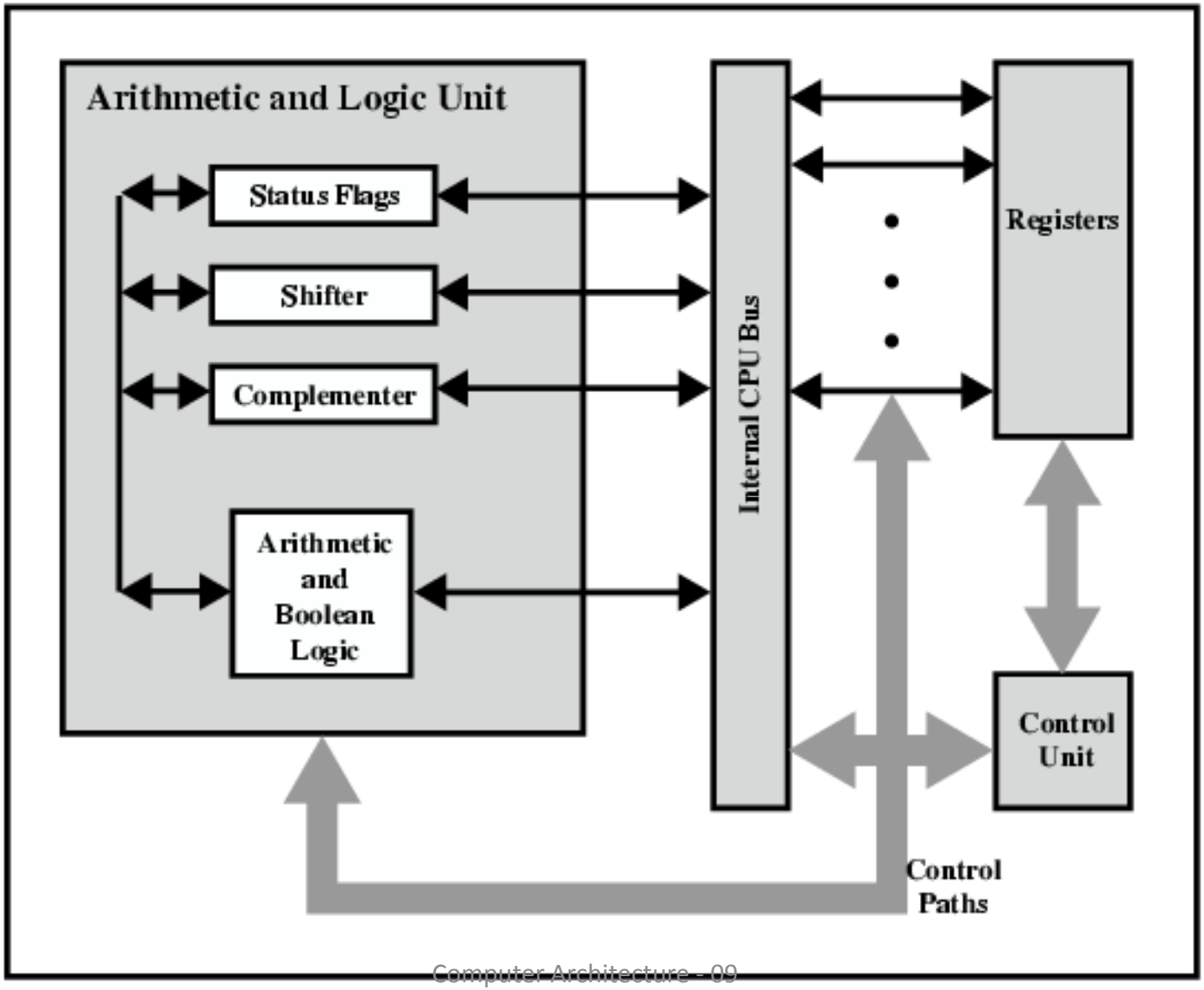
CPU Structure

- 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

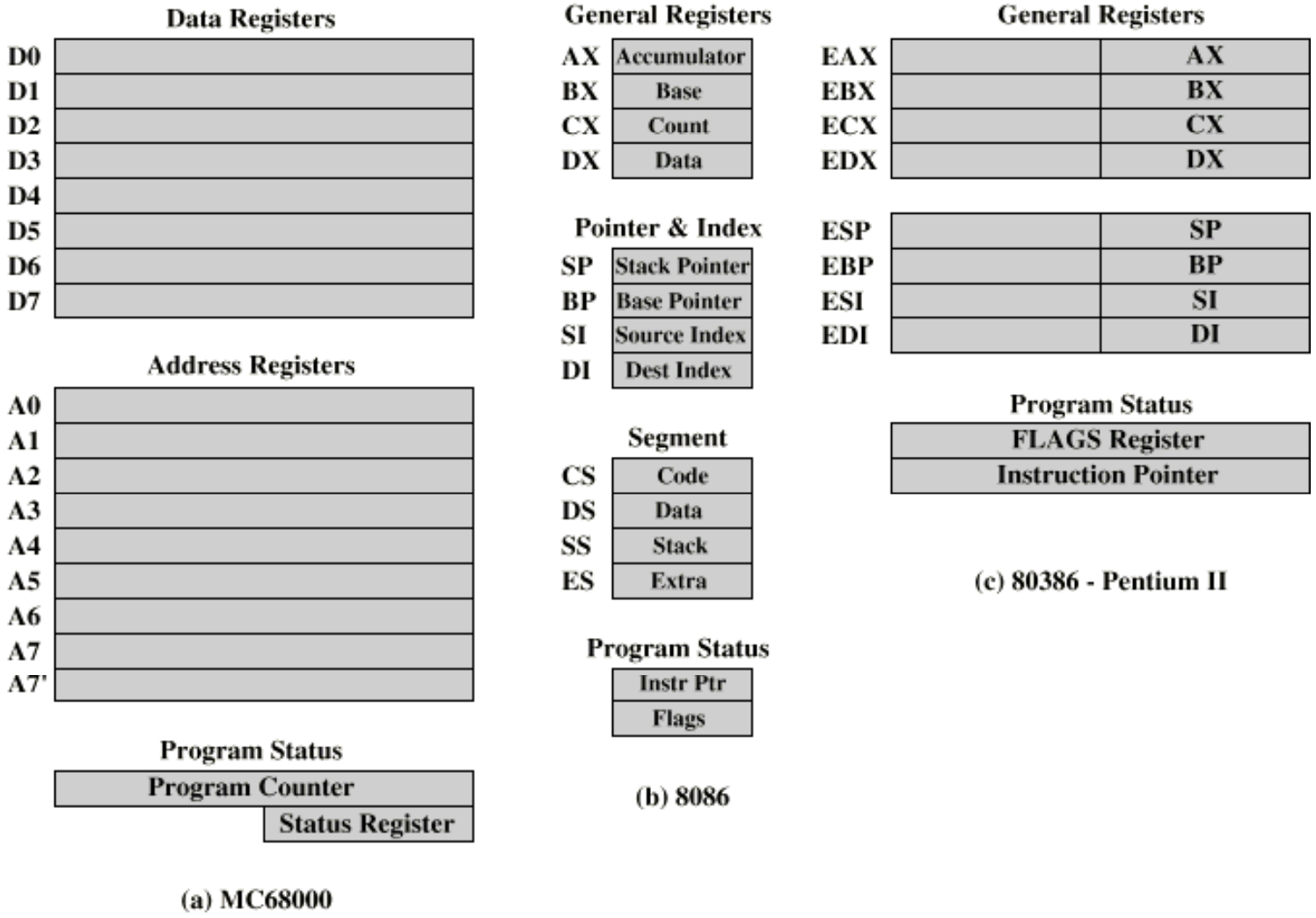
- Intel ring zero
- Kernel mode
- Allows privileged instructions to execute
- Used by operating system
- Not available to user programs

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



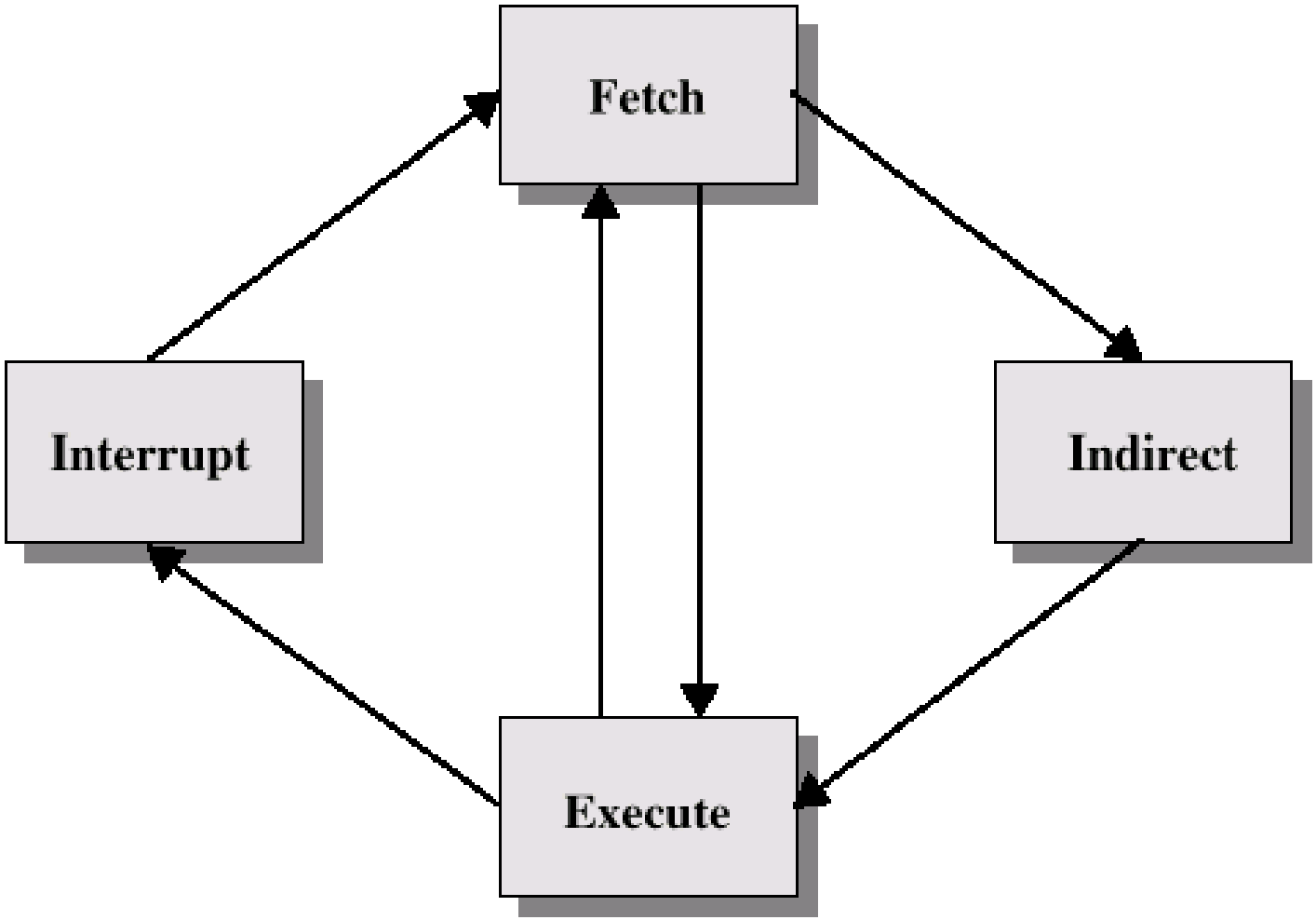
Instruction Cycle

- Revision
- Stallings Chapter 3

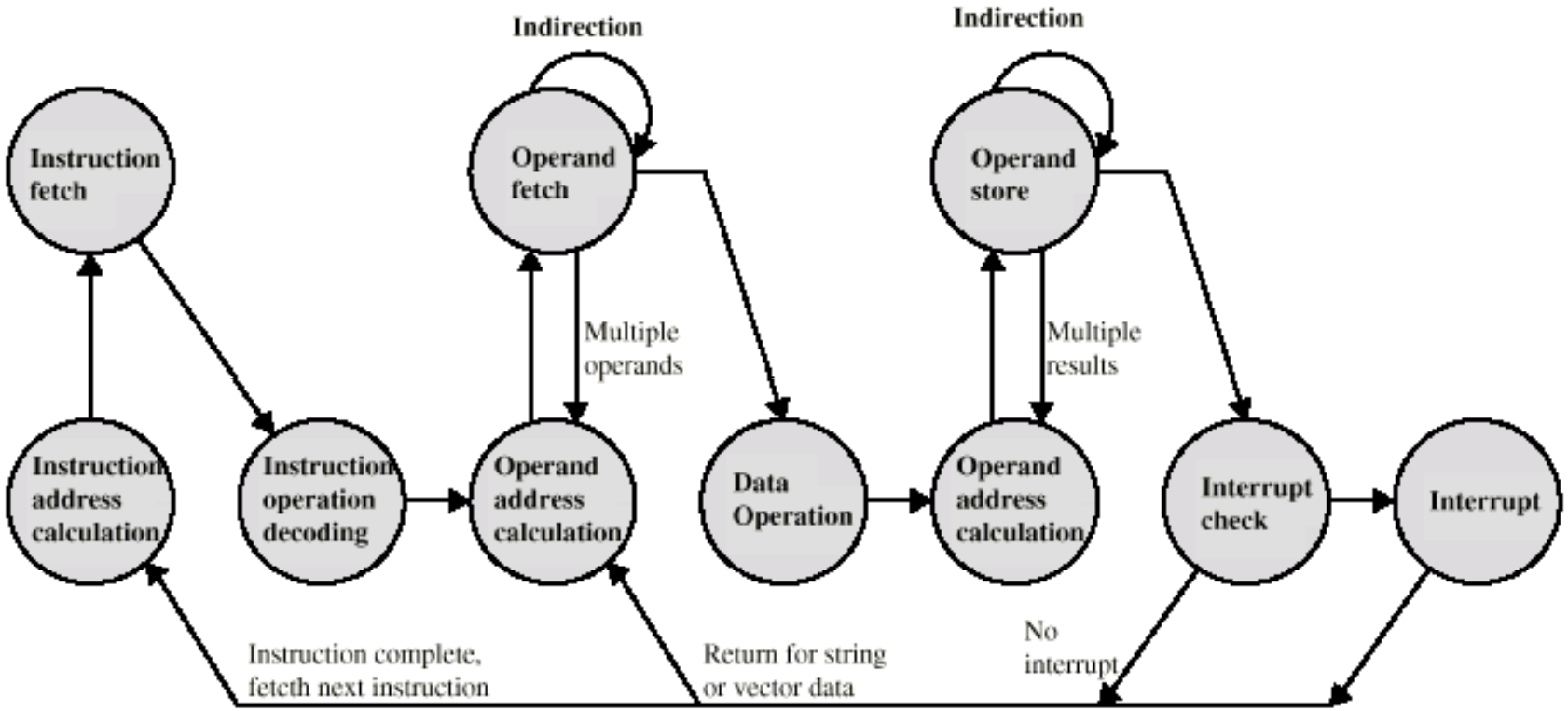
Indirect Cycle

- 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)

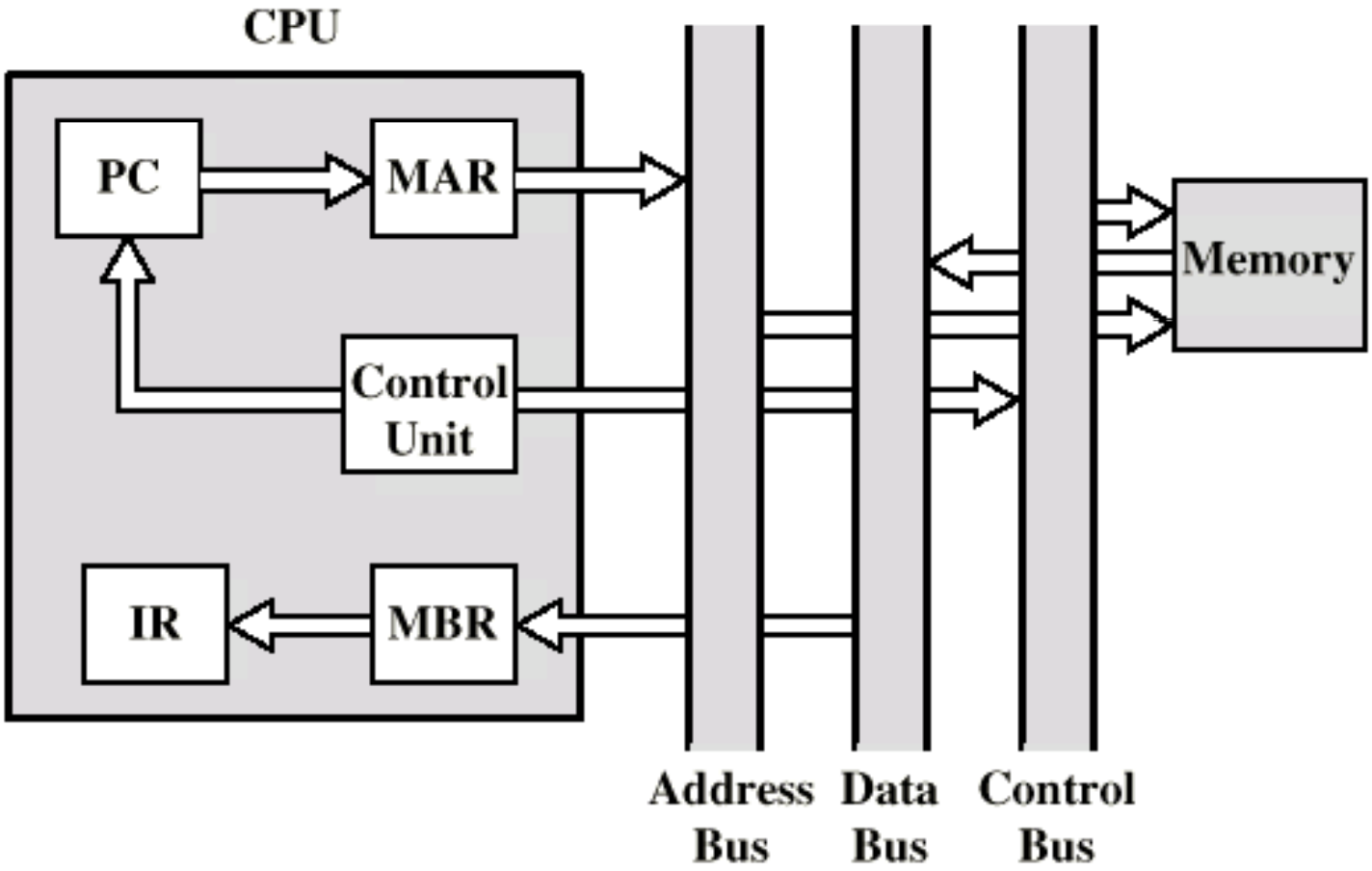
- Depends on CPU design
- In general:

- 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)

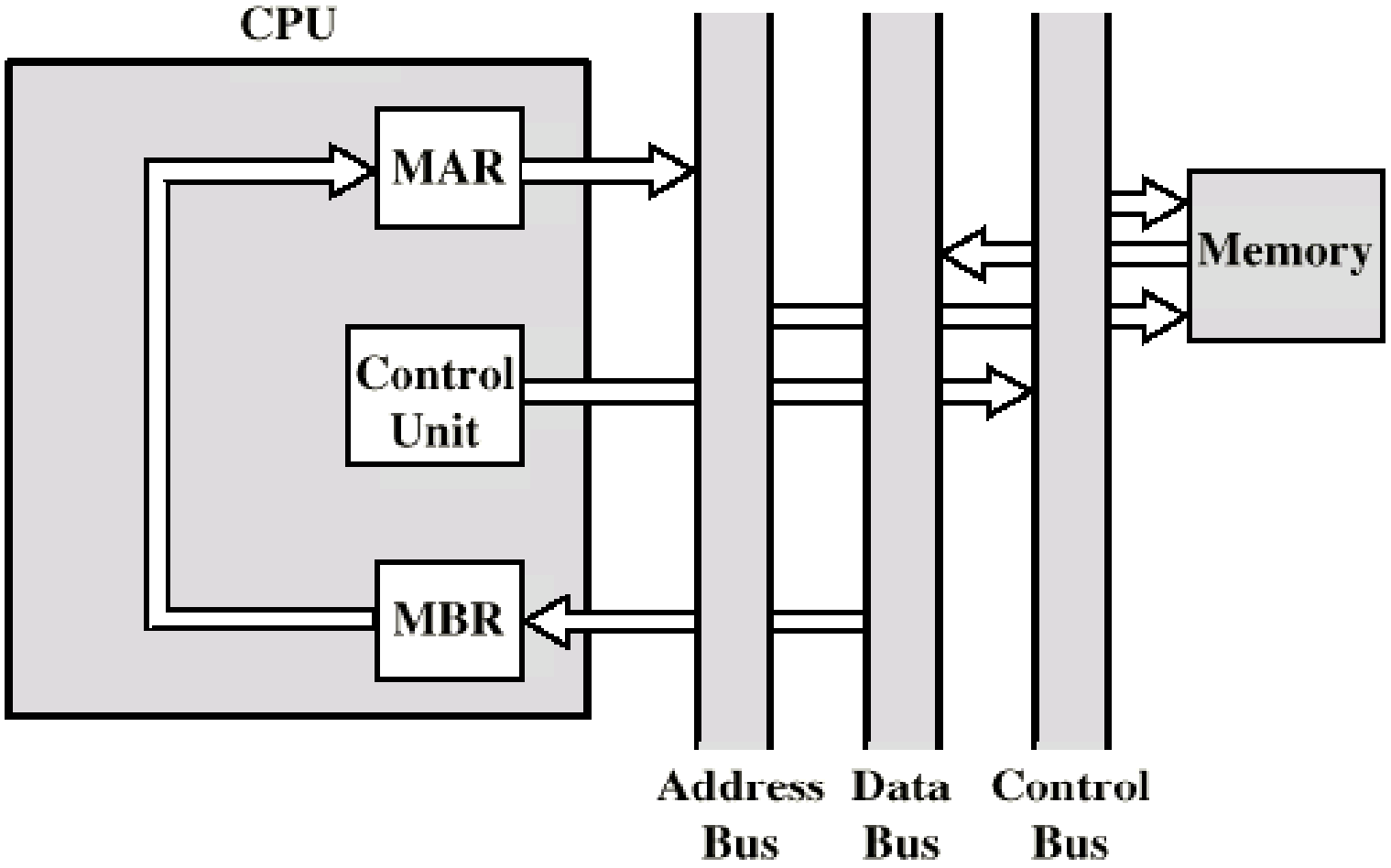
- IR is examined
- If indirect addressing, indirect cycle is performed
  - Right most N bits of MBR transferred to MAR
  - Control unit requests memory read
  - Result (address of operand) moved to MBR

Data Flow (Fetch Diagram)



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

Data Flow (Indirect Diagram)



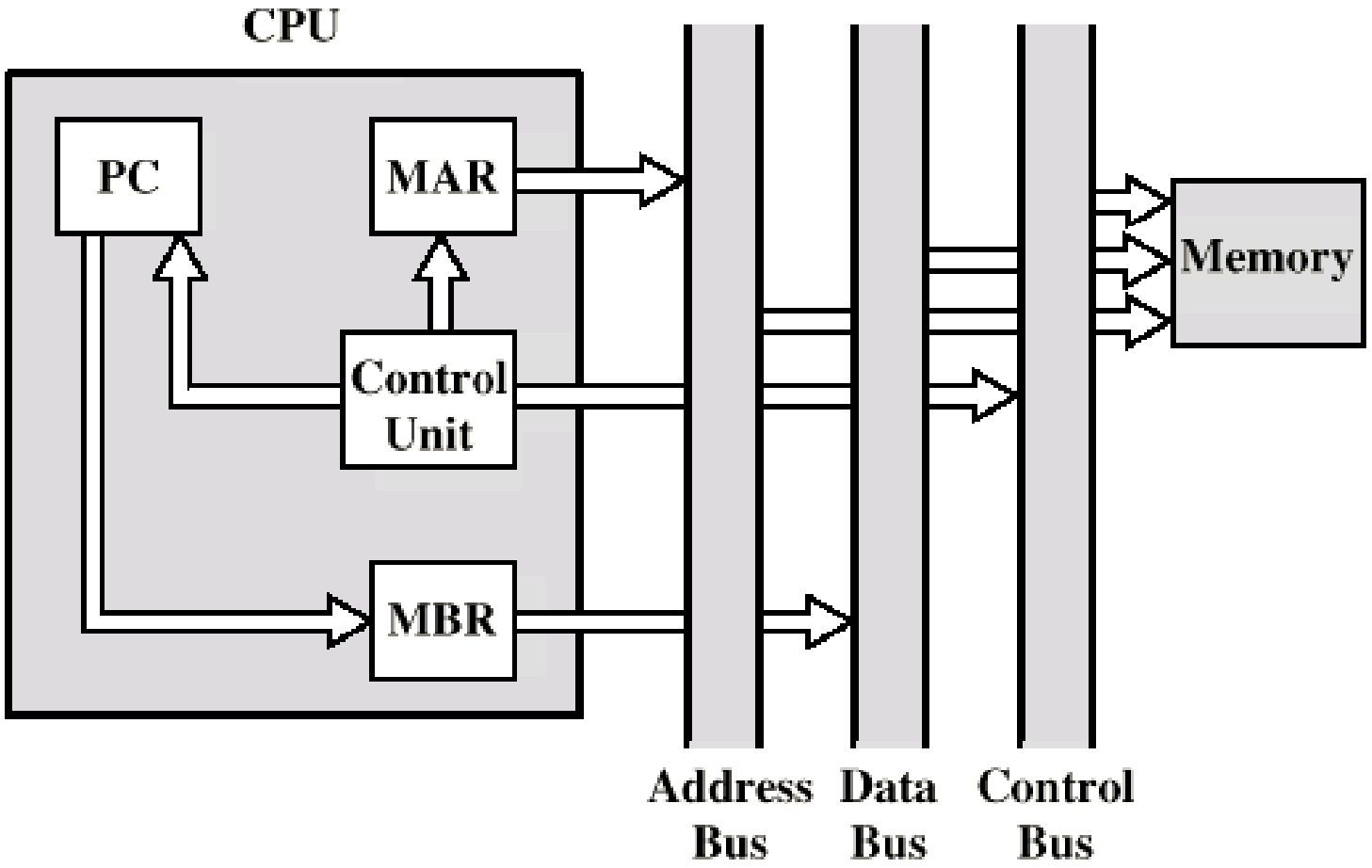
Data Flow (Execute)

- 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

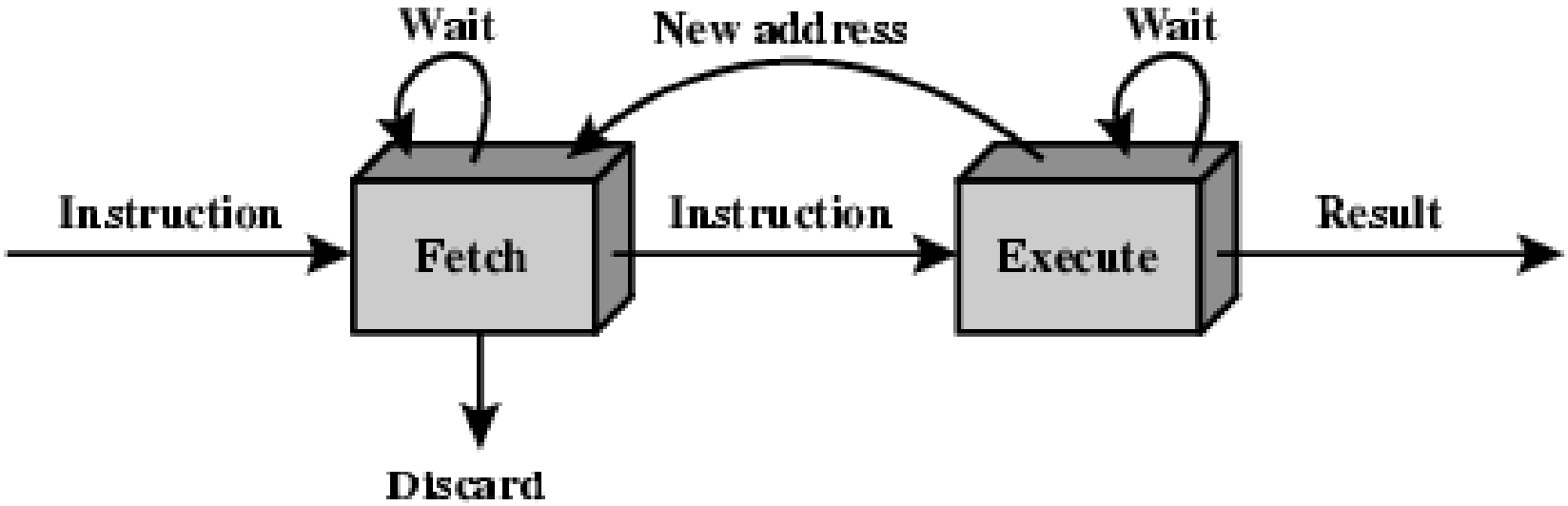
Pipelining

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

- Overlap these operations

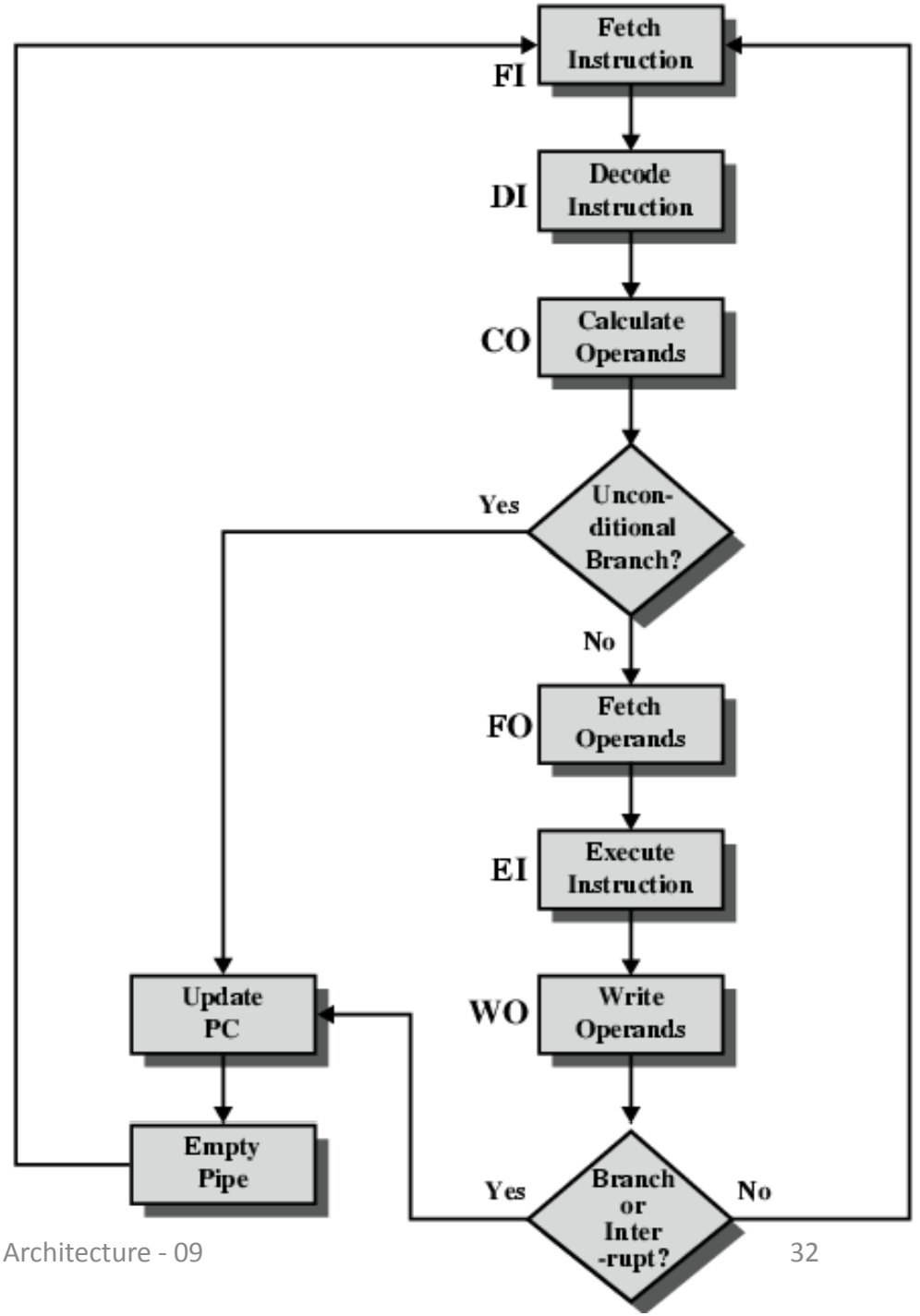
Two Stage Instruction Pipeline

(a) Simplified view



(b) Expanded view

Six Stage Instruction Pipeline



Pentium 4 Registers

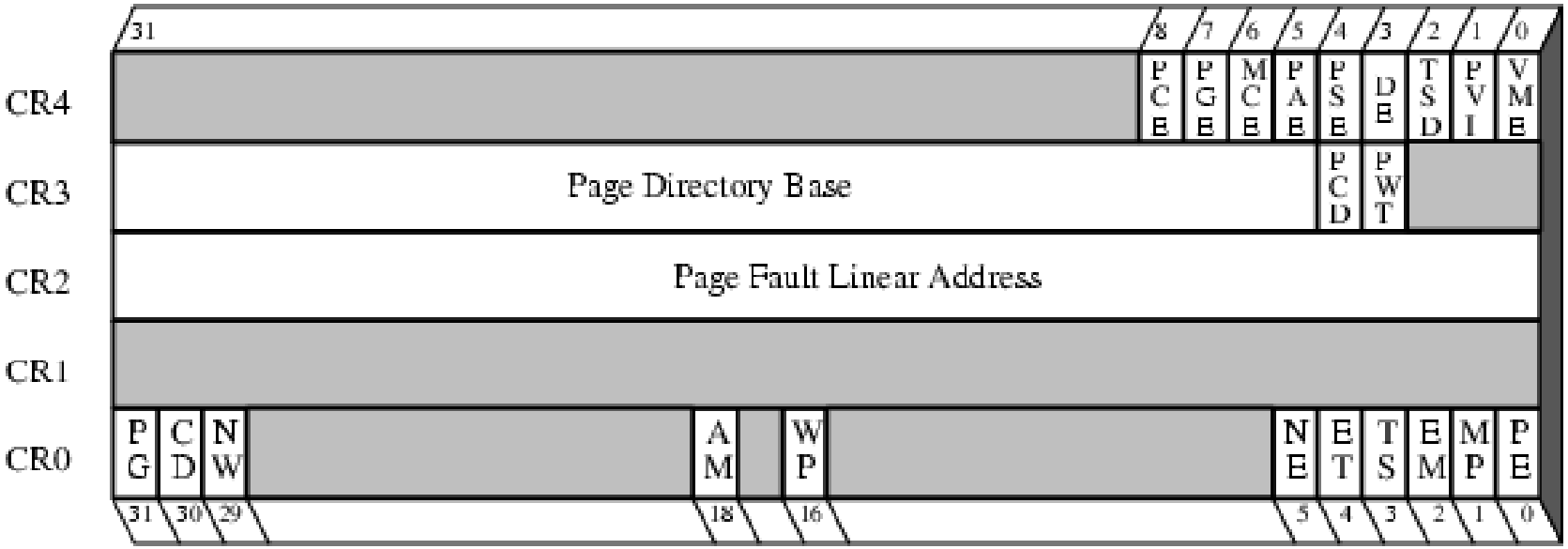
(a) Integer Unit

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	48	Points to instruction interrupted by exception
Data Pointer	1	48	Points to operand interrupted by exception

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

Questions