William Stallings
Computer Organization and Architecture

Chapter 1
Introduction

Architecture & Organization 1

- Architecture is those attributes visible to the programmer
  - Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.
  - e.g. Is there a multiply instruction?
- Organization is how features are implemented
  - Control signals, interfaces, memory technology.
  - e.g. Is there a hardware multiply unit or is it done by repeated addition?

Architecture & Organization 2

- All Intel x86 family share the same basic architecture
- The IBM System/370 family share the same basic architecture

- This gives code compatibility
  - At least backwards
- Organization differs between different versions

Structure & Function

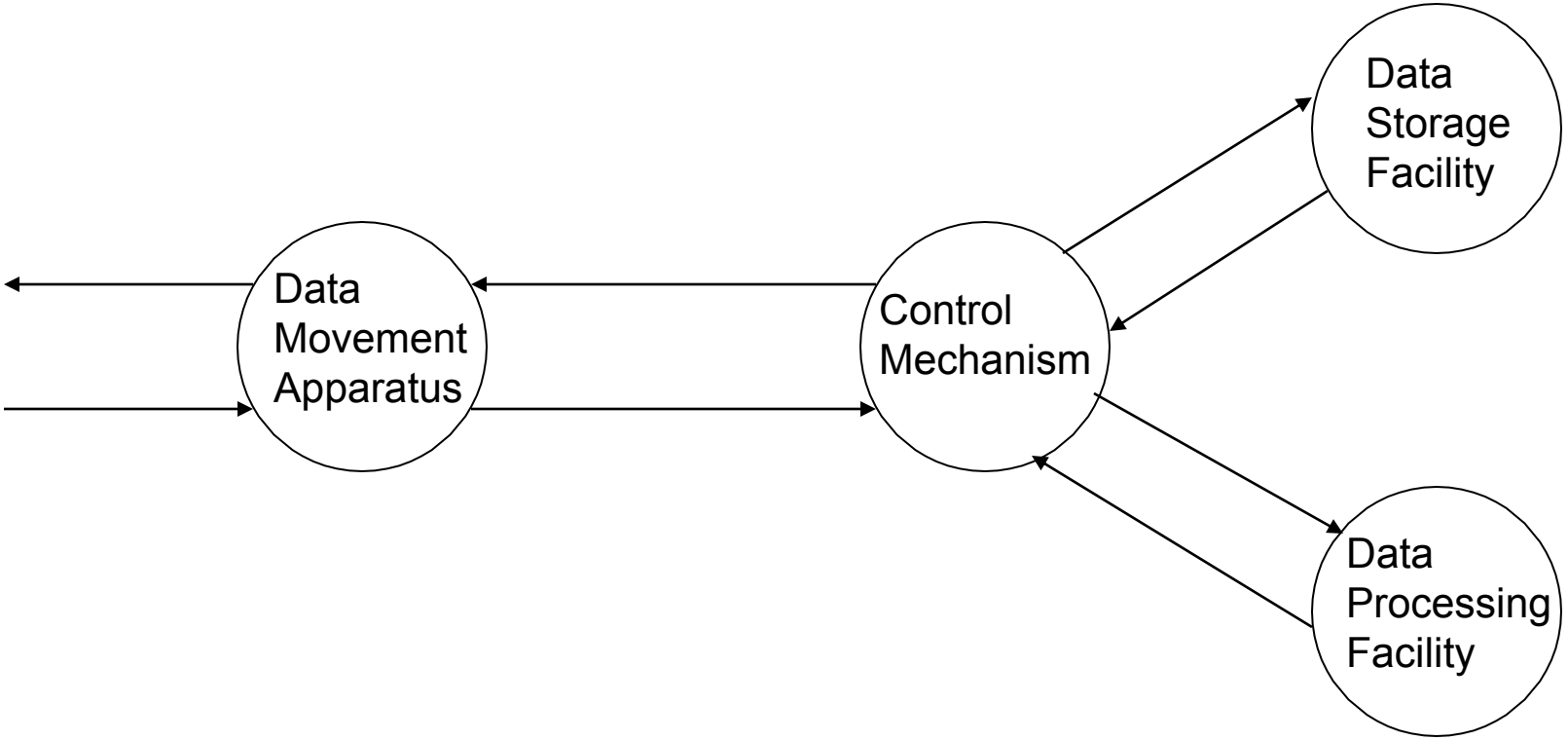
- Structure is the way in which components relate to each other
- Function is the operation of individual components as part of the structure

Function

- All computer functions are:
  - Data processing
  - Data storage
  - Data movement
  - Control

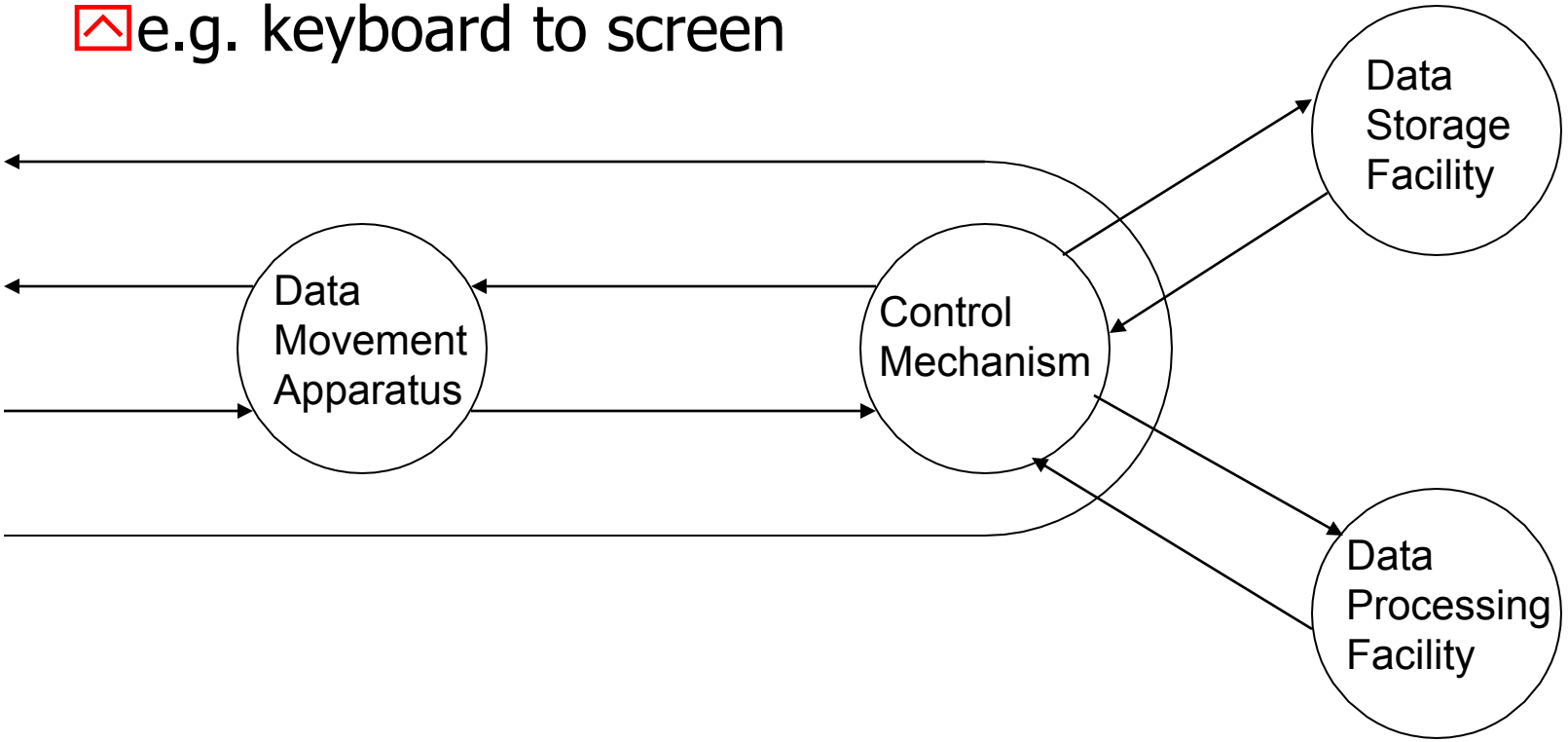
Functional view

- Functional view of a computer



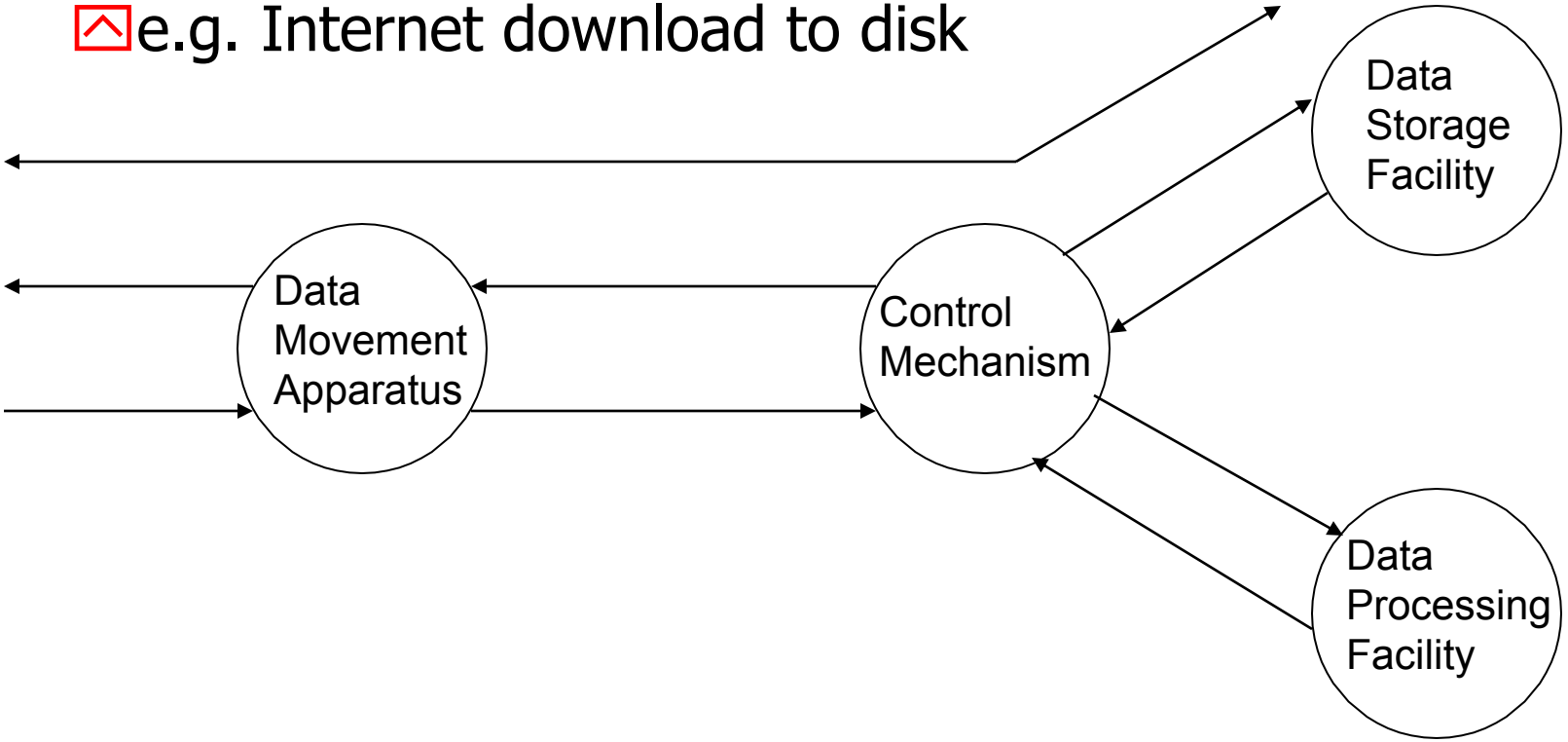
Operations (1)

- Data movement
  - e.g. keyboard to screen



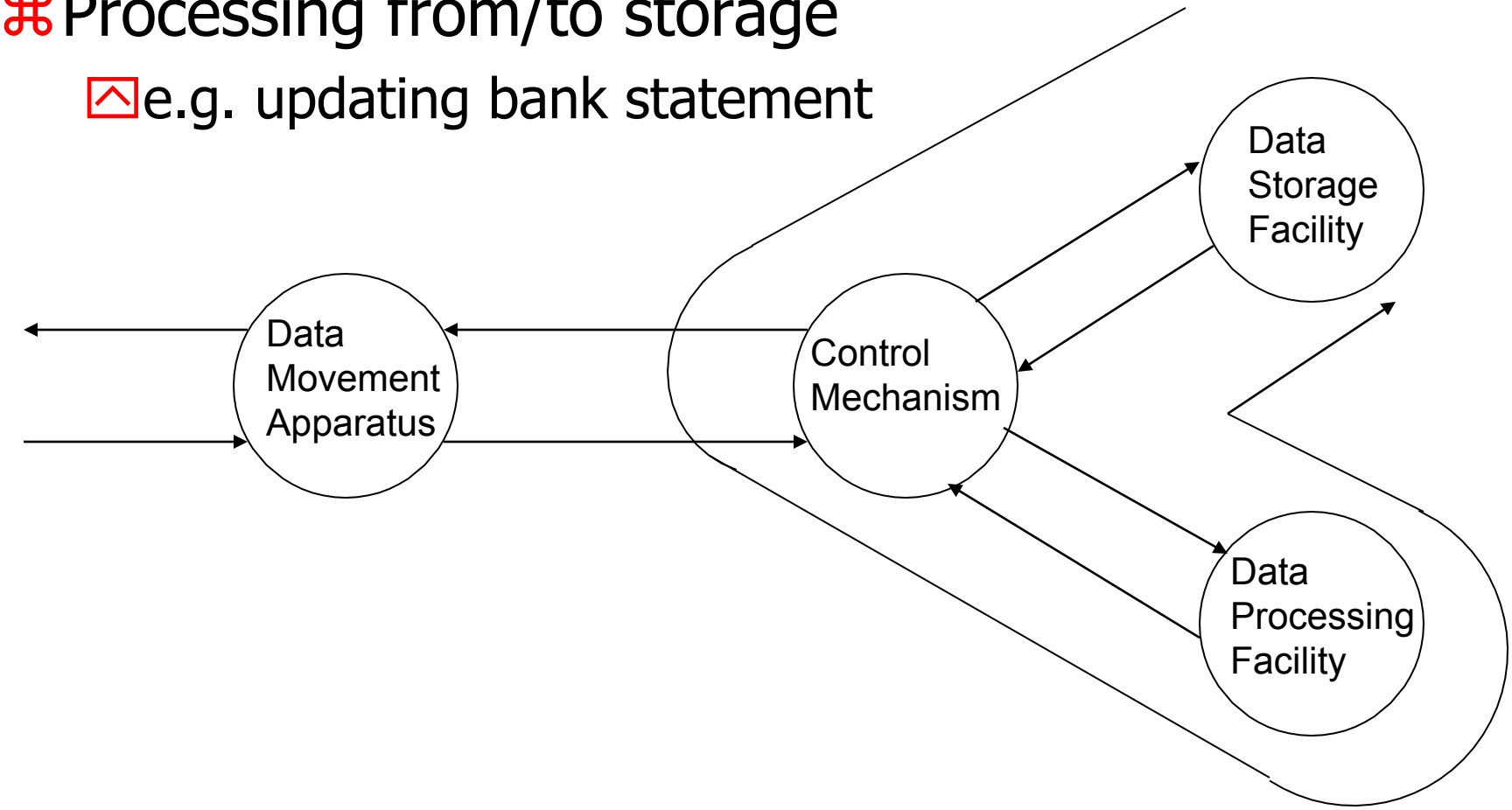
Operations (2)

- Storage
  - e.g. Internet download to disk



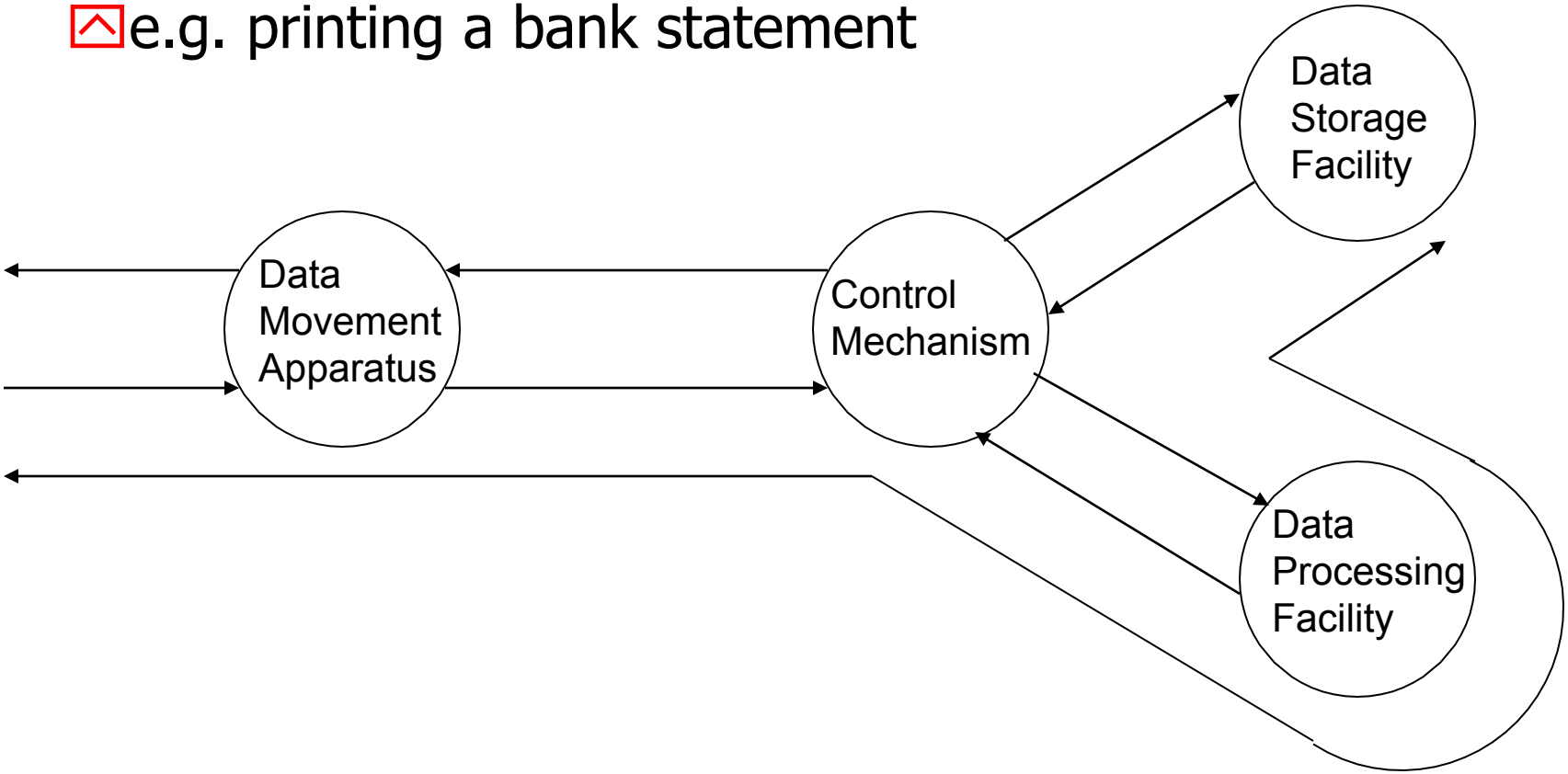
Operation (3)

- Processing from/to storage
  - e.g. updating bank statement

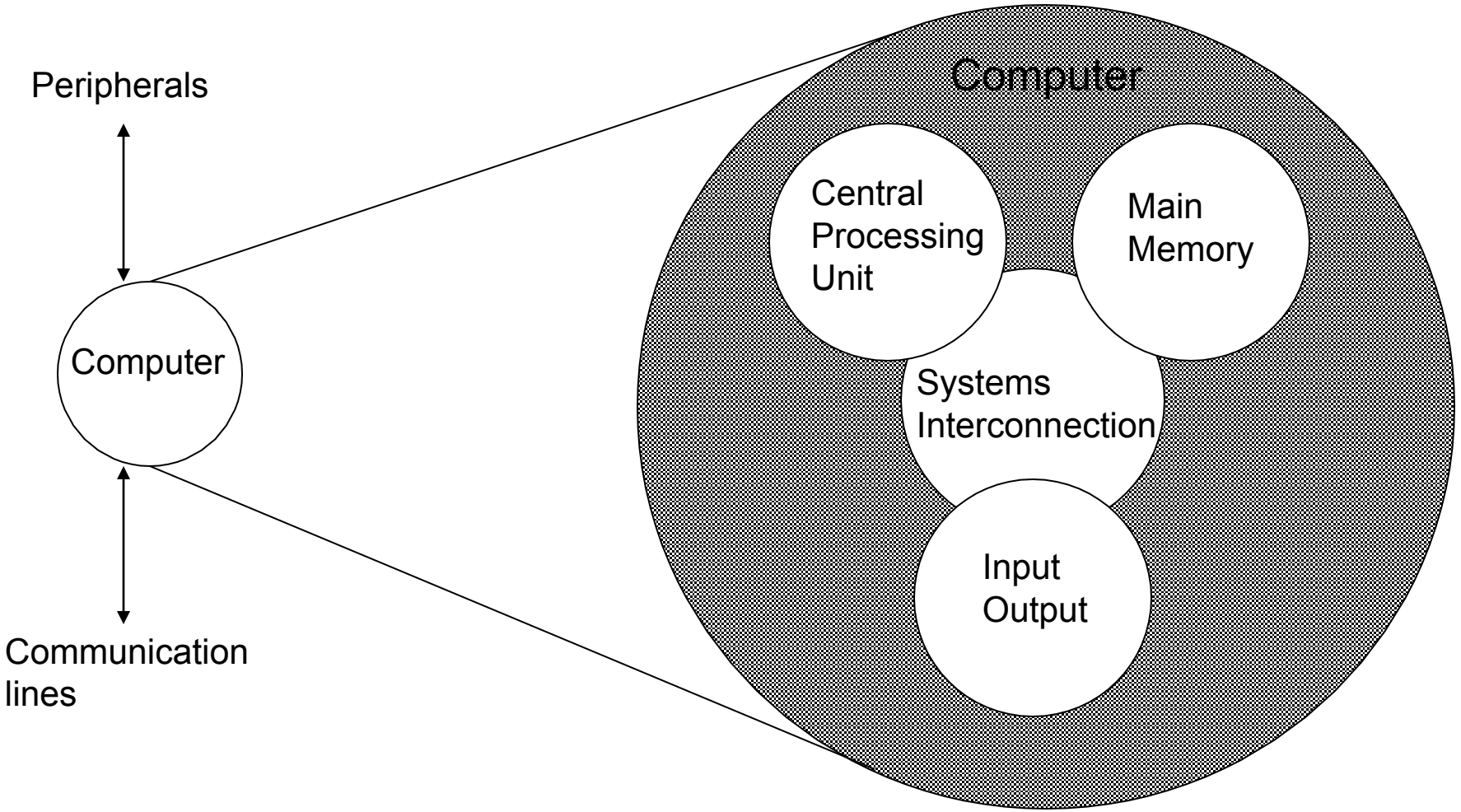


Operation (4)

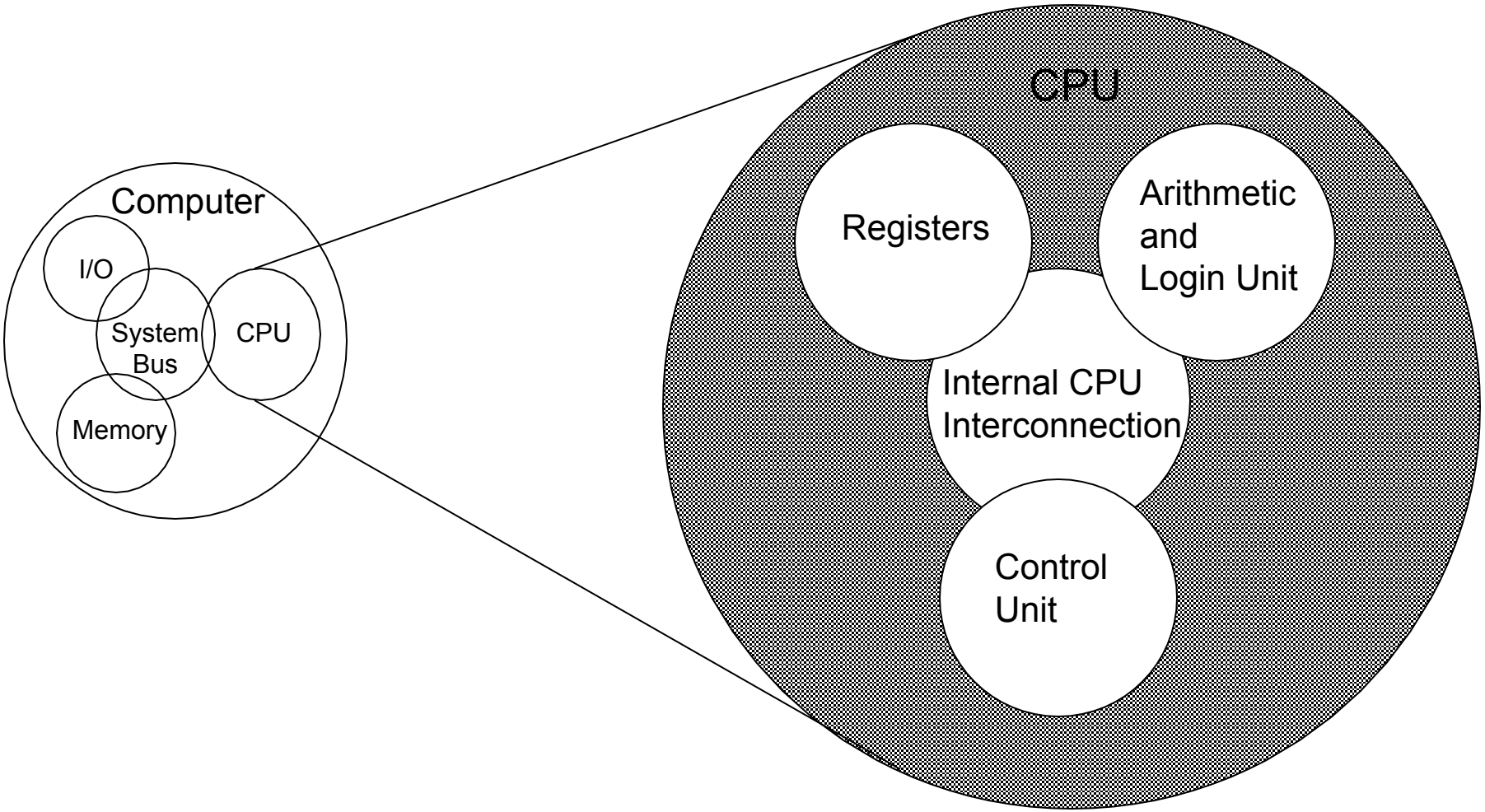
- Processing from storage to I/O
  - e.g. printing a bank statement



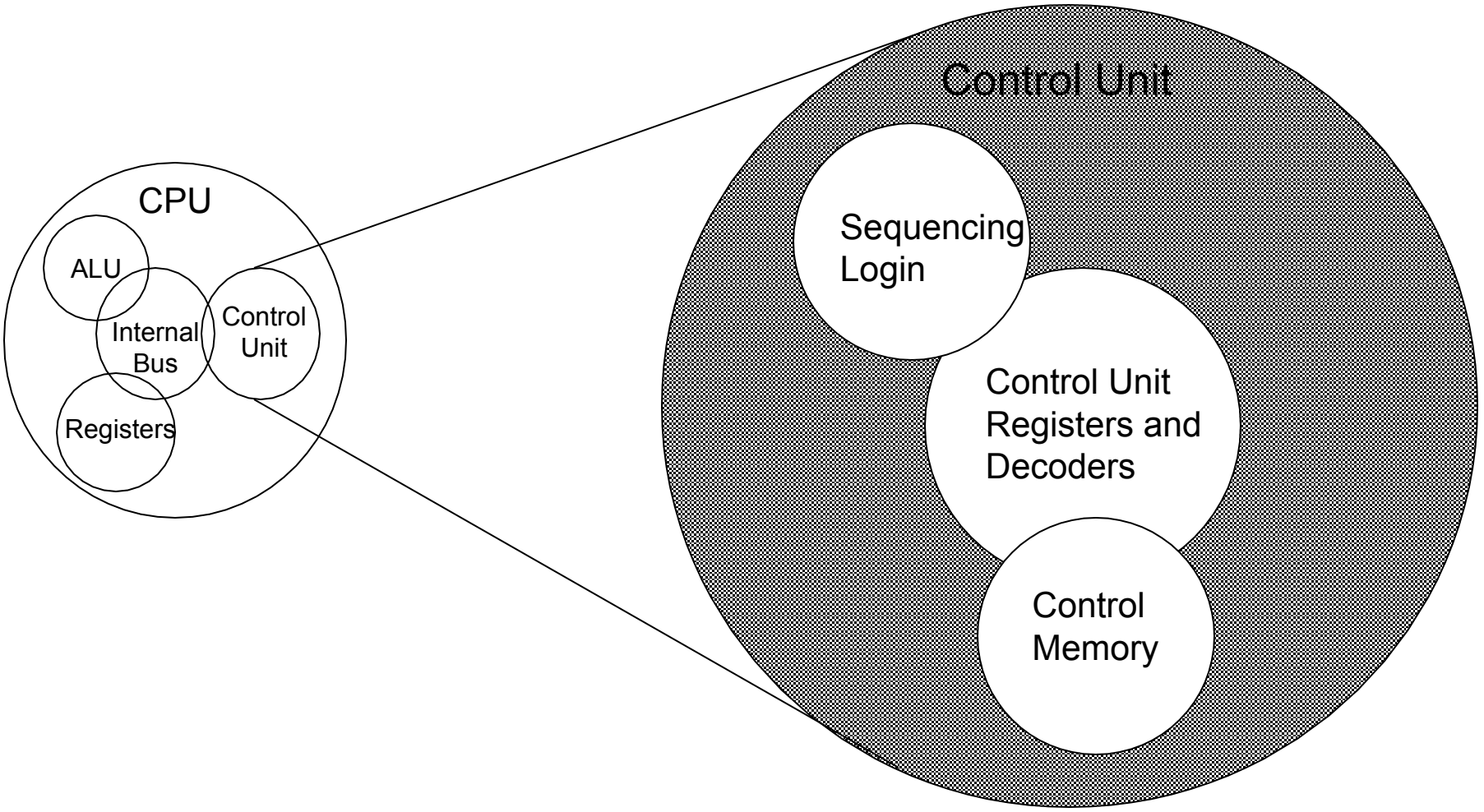
Structure - Top Level



Structure - The CPU



Structure - The Control Unit



Outline of the Book (1)

- Computer Evolution and Performance
- Computer Interconnection Structures
- Internal Memory
- External Memory
- Input/Output
- Operating Systems Support
- Computer Arithmetic
- Instruction Sets

Outline of the Book (2)

- CPU Structure and Function
- Reduced Instruction Set Computers
- Superscalar Processors
- Control Unit Operation
- Microprogrammed Control
- Multiprocessors and Vector Processing
- Digital Logic (Appendix)

Internet Resources - Web site for book

- http://www.shore.net/~ws/COA5e.html
  - links to sites of interest
  - links to sites for courses that use the book
  - errata list for book
  - information on other books by W. Stallings

Internet Resources - Web sites to look for

- WWW Computer Architecture Home Page
- CPU Info Center
- ACM Special Interest Group on Computer Architecture
- IEEE Technical Committee on Computer Architecture
- Intel Technology Journal
- Manufacturer's sites
  - Intel, IBM, etc.

Internet Resources
- Usenet News Groups

- comp.arch
- comp.arch.arithmetic
- comp.arch.storage

William Stallings
Computer Organization and Architecture

Chapter 2
Computer Evolution and Performance

ENIAC - background

- Electronic Numerical Integrator And Computer
- Eckert and Mauchly
- University of Pennsylvania
- Trajectory tables for weapons
- Started 1943
- Finished 1946
  - Too late for war effort
- Used until 1955

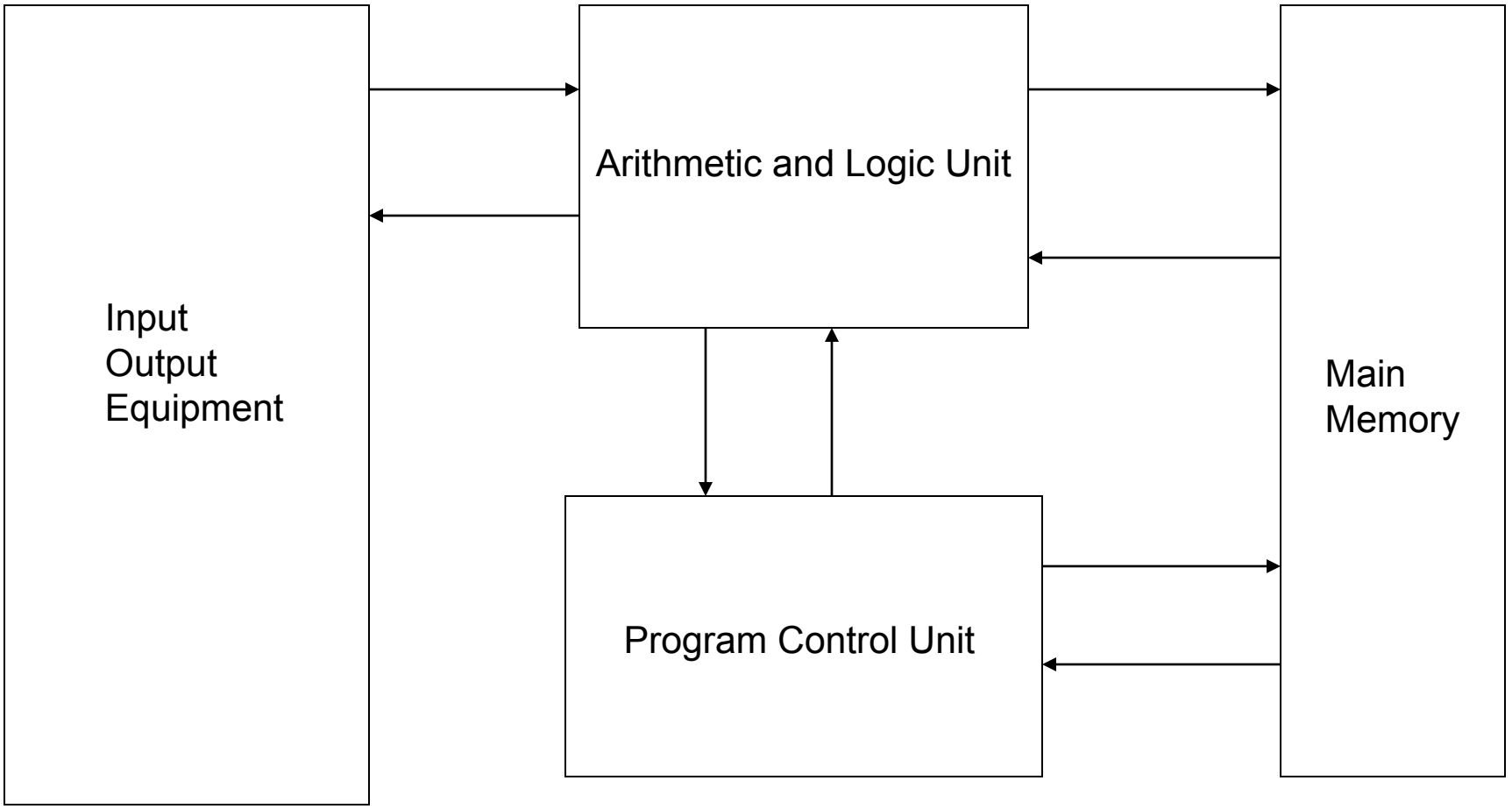
ENIAC - details

- Decimal (not binary)
- 20 accumulators of 10 digits
- Programmed manually by switches
- 18,000 vacuum tubes
- 30 tons
- 15,000 square feet
- 140 kW power consumption
- 5,000 additions per second

von Neumann/Turing

- Stored Program concept
- Main memory storing programs and data
- ALU operating on binary data
- Control unit interpreting instructions from memory and executing
- Input and output equipment operated by control unit
- Princeton Institute for Advanced Studies
  - IAS
- Completed 1952

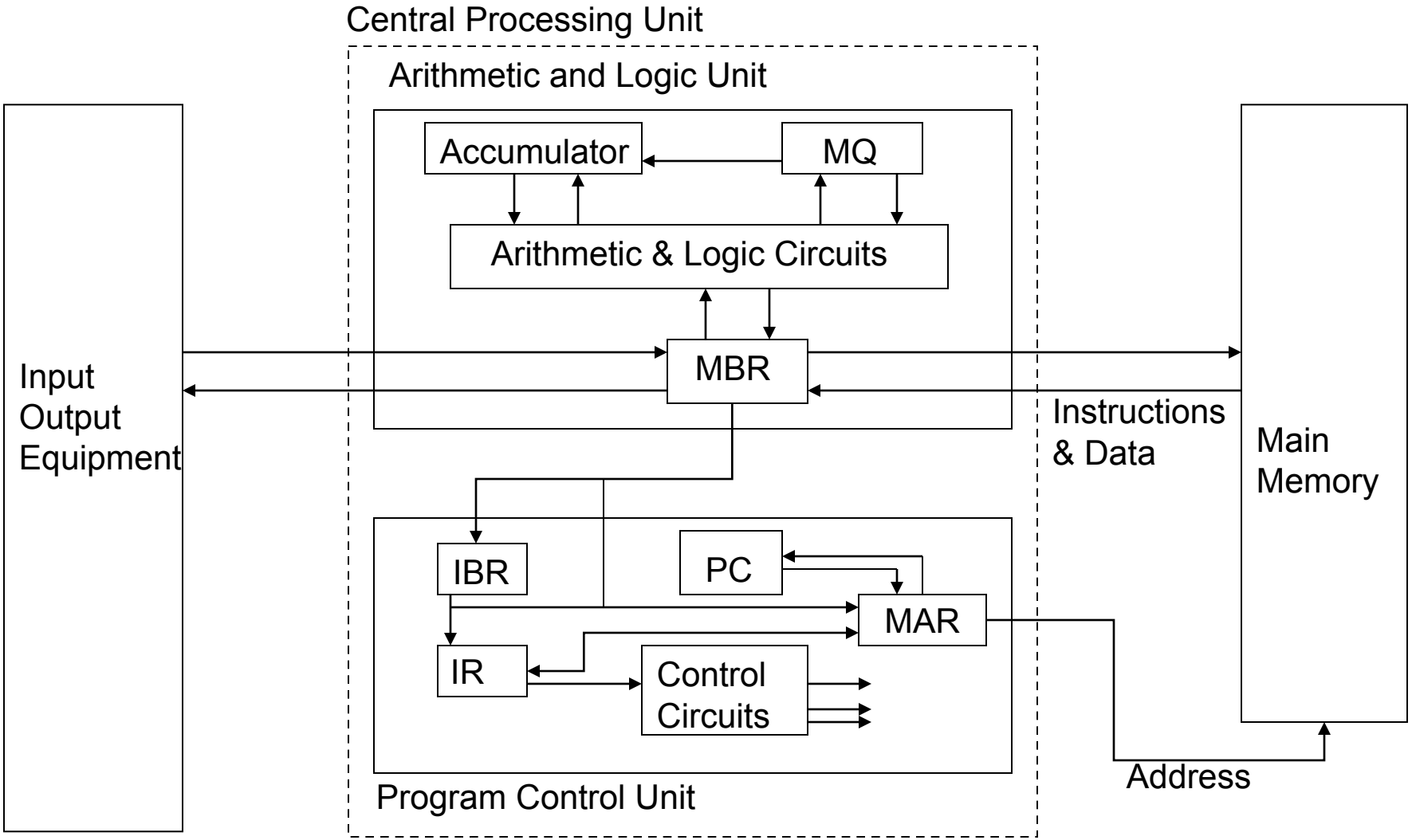
Structure of von Nuemann machine



IAS - details

- 1000 x 40 bit words
  - Binary number
  - 2 x 20 bit instructions
- Set of registers (storage in CPU)
  - Memory Buffer Register
  - Memory Address Register
  - Instruction Register
  - Instruction Buffer Register
  - Program Counter
  - Accumulator
  - Multiplier Quotient

Structure of IAS - detail



Commercial Computers

- 1947 - Eckert-Mauchly Computer Corporation
- UNIVAC I (Universal Automatic Computer)
- US Bureau of Census 1950 calculations
- Became part of Sperry-Rand Corporation
- Late 1950s - UNIVAC II
  - Faster
  - More memory

IBM

- Punched-card processing equipment
- 1953 - the 701
  - IBM's first stored program computer
  - Scientific calculations
- 1955 - the 702
  - Business applications
- Lead to 700/7000 series

Transistors

- Replaced vacuum tubes
- Smaller
- Cheaper
- Less heat dissipation
- Solid State device
- Made from Silicon (Sand)
- Invented 1947 at Bell Labs
- William Shockley et al.

Transistor Based Computers

- Second generation machines
- NCR & RCA produced small transistor machines
- IBM 7000
- DEC - 1957
  - Produced PDP-1

Microelectronics

- Literally - “small electronics”
- A computer is made up of gates, memory cells and interconnections
- These can be manufactured on a semiconductor
- e.g. silicon wafer

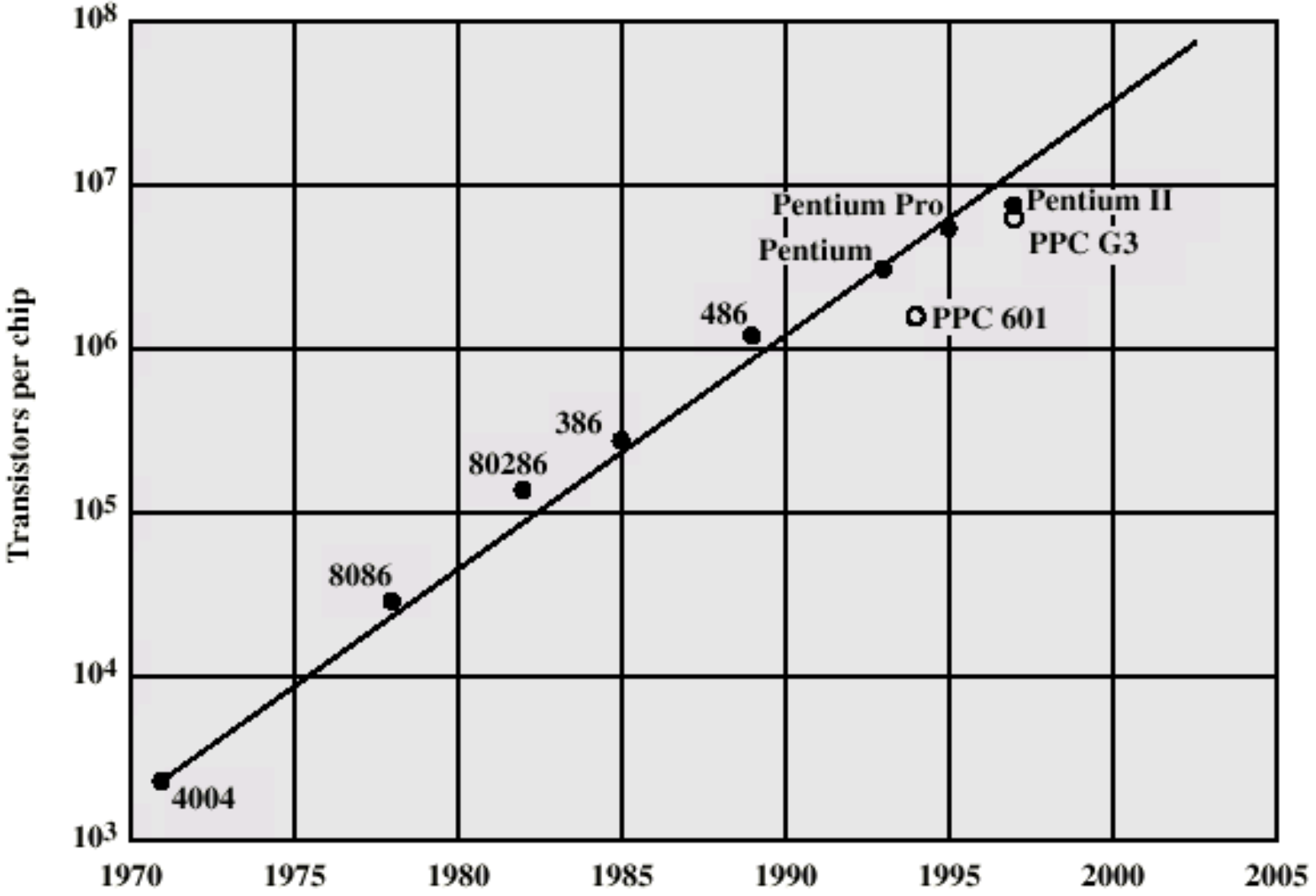
Generations of Computer

- Vacuum tube - 1946-1957
- Transistor - 1958-1964
- Small scale integration - 1965 on
  - Up to 100 devices on a chip
- Medium scale integration - to 1971
  - 100-3,000 devices on a chip
- Large scale integration - 1971-1977
  - 3,000 - 100,000 devices on a chip
- Very large scale integration - 1978 to date
  - 100,000 - 100,000,000 devices on a chip
- Ultra large scale integration
  - Over 100,000,000 devices on a chip

Moore's Law

- Increased density of components on chip
- Gordon Moore - cofounder of Intel
- Number of transistors on a chip will double every year
- Since 1970's development has slowed a little
  - Number of transistors doubles every 18 months
- Cost of a chip has remained almost unchanged
- Higher packing density means shorter electrical paths, giving higher performance
- Smaller size gives increased flexibility
- Reduced power and cooling requirements
- Fewer interconnections increases reliability

Growth in CPU Transistor Count



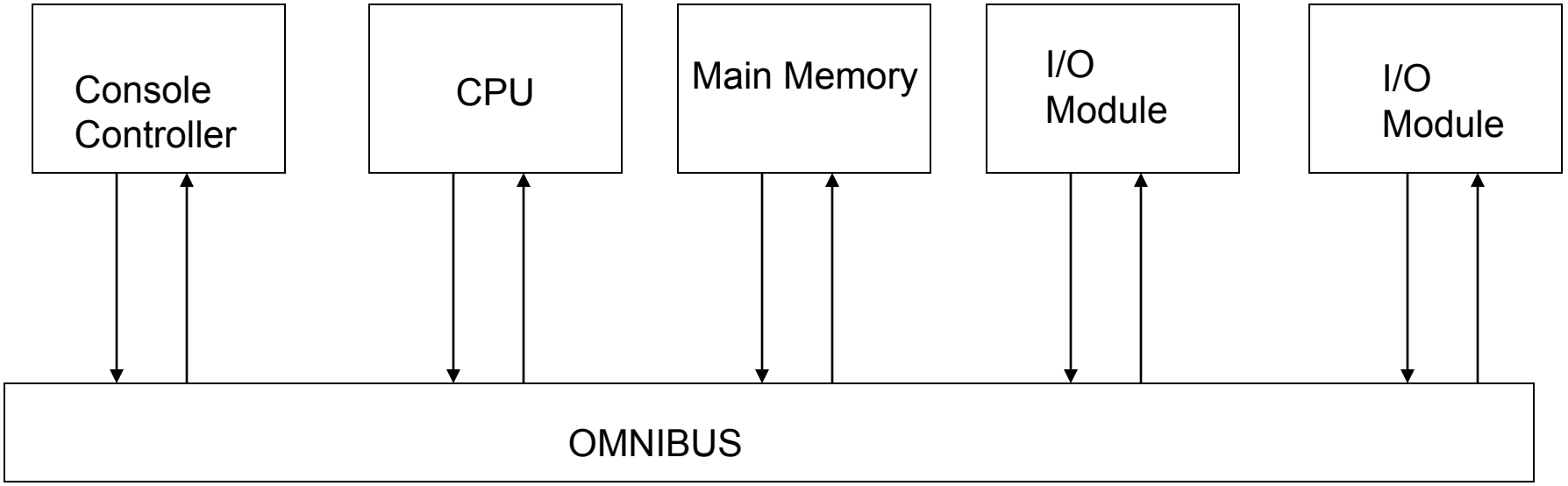
IBM 360 series

- 1964
- Replaced (& not compatible with) 7000 series
- First planned "family" of computers
  - Similar or identical instruction sets
  - Similar or identical O/S
  - Increasing speed
  - Increasing number of I/O ports (i.e. more terminals)
  - Increased memory size
  - Increased cost
- Multiplexed switch structure

DEC PDP-8

- 1964
- First minicomputer (after miniskirt!)
- Did not need air conditioned room
- Small enough to sit on a lab bench
- $16,000
  - $100k+ for IBM 360
- Embedded applications & OEM
- BUS STRUCTURE

DEC - PDP-8 Bus Structure



Semiconductor Memory

- 1970
- Fairchild
- Size of a single core
  - i.e. 1 bit of magnetic core storage
- Holds 256 bits
- Non-destructive read
- Much faster than core
- Capacity approximately doubles each year

Intel

- 1971 - 4004
  - First microprocessor
  - All CPU components on a single chip
  - 4 bit
- Followed in 1972 by 8008
  - 8 bit
  - Both designed for specific applications
- 1974 - 8080
  - Intel's first general purpose microprocessor

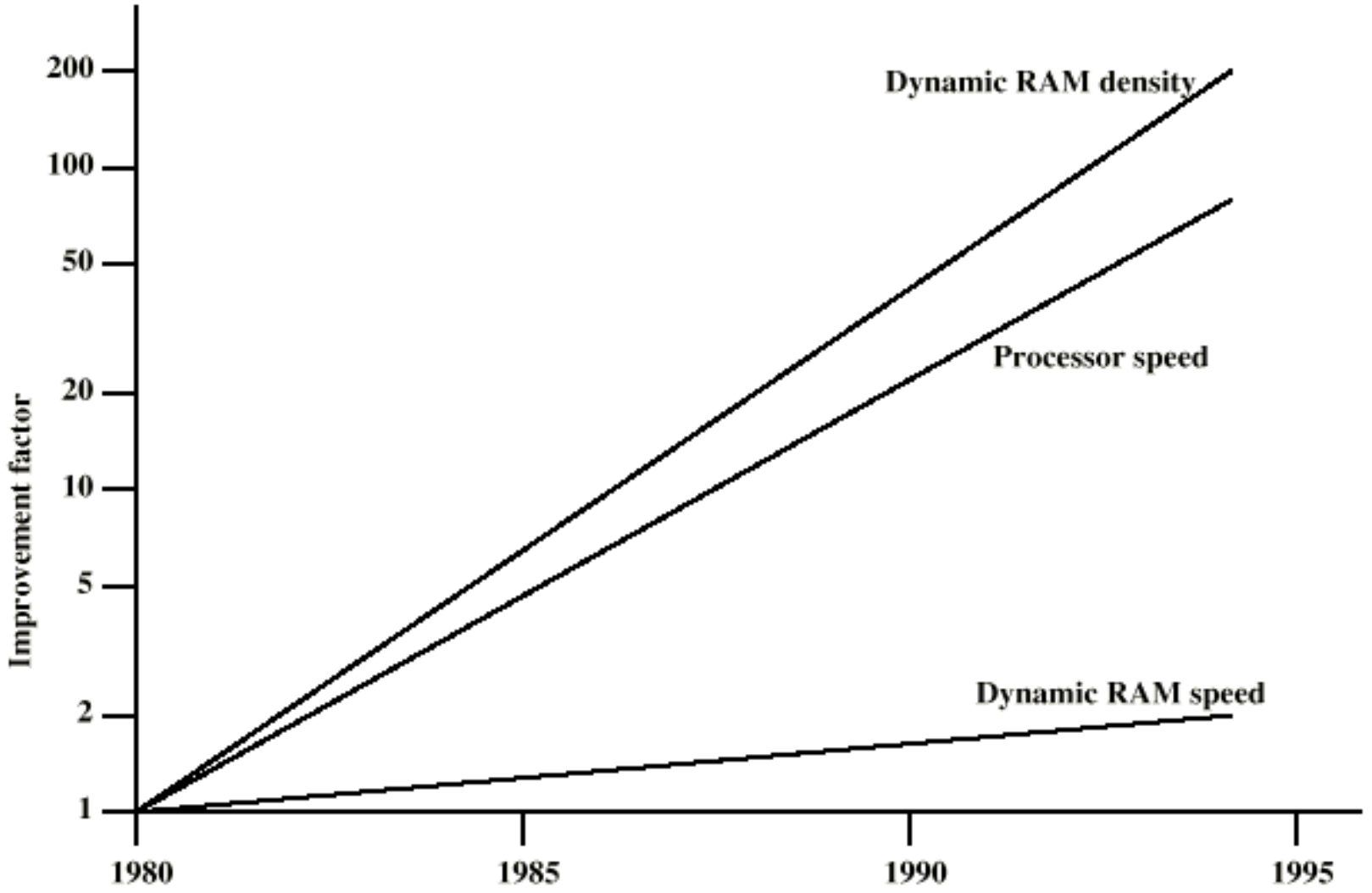
Speeding it up

- Pipelining
- On board cache
- On board L1 & L2 cache
- Branch prediction
- Data flow analysis
- Speculative execution

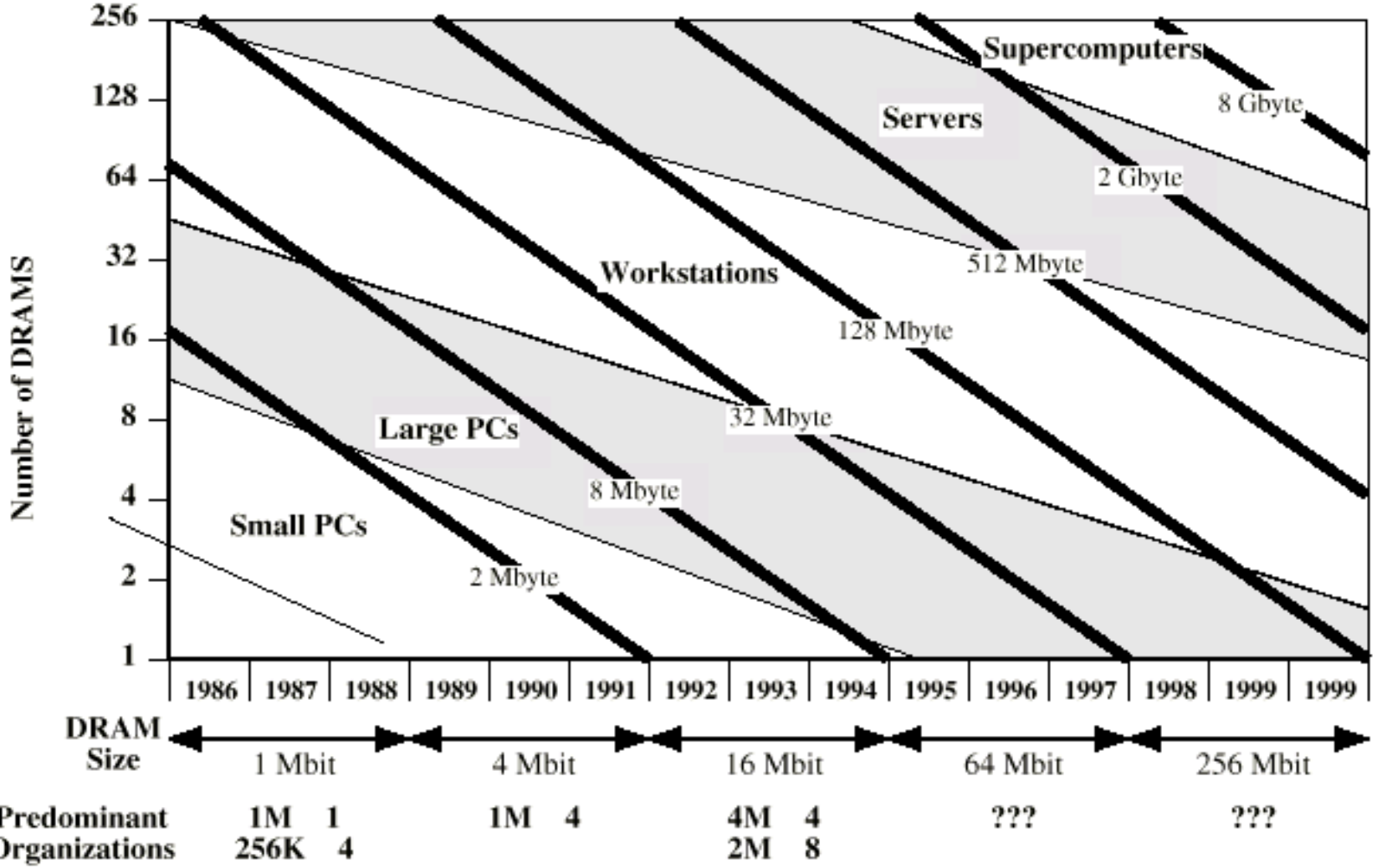
Performance Mismatch

- Processor speed increased
- Memory capacity increased
- Memory speed lags behind processor speed

DRAM and Processor Characteristics



Trends in DRAM use



Solutions

- Increase number of bits retrieved at one time
  - Make DRAM "wider" rather than "deeper"
- Change DRAM interface
  - Cache
- Reduce frequency of memory access
  - More complex cache and cache on chip
- Increase interconnection bandwidth
  - High speed buses
  - Hierarchy of buses

Internet Resources

- http://www.intel.com/
  - Search for the Intel Museum
- http://www.ibm.com
- http://www.dec.com
- Charles Babbage Institute
- PowerPC
- Intel Developer Home

William Stallings
Computer Organization and Architecture

Chapter 3

A top level view of computer function & Interconnection

Program Concept

- Hardwired systems are inflexible
- General purpose hardware can do different tasks, given correct control signals
- Instead of re-wiring, supply a new set of control signals

What is a program?

- A sequence of steps
- For each step, an arithmetic or logical operation is done
- For each operation, a different set of control signals is needed

Function of Control Unit

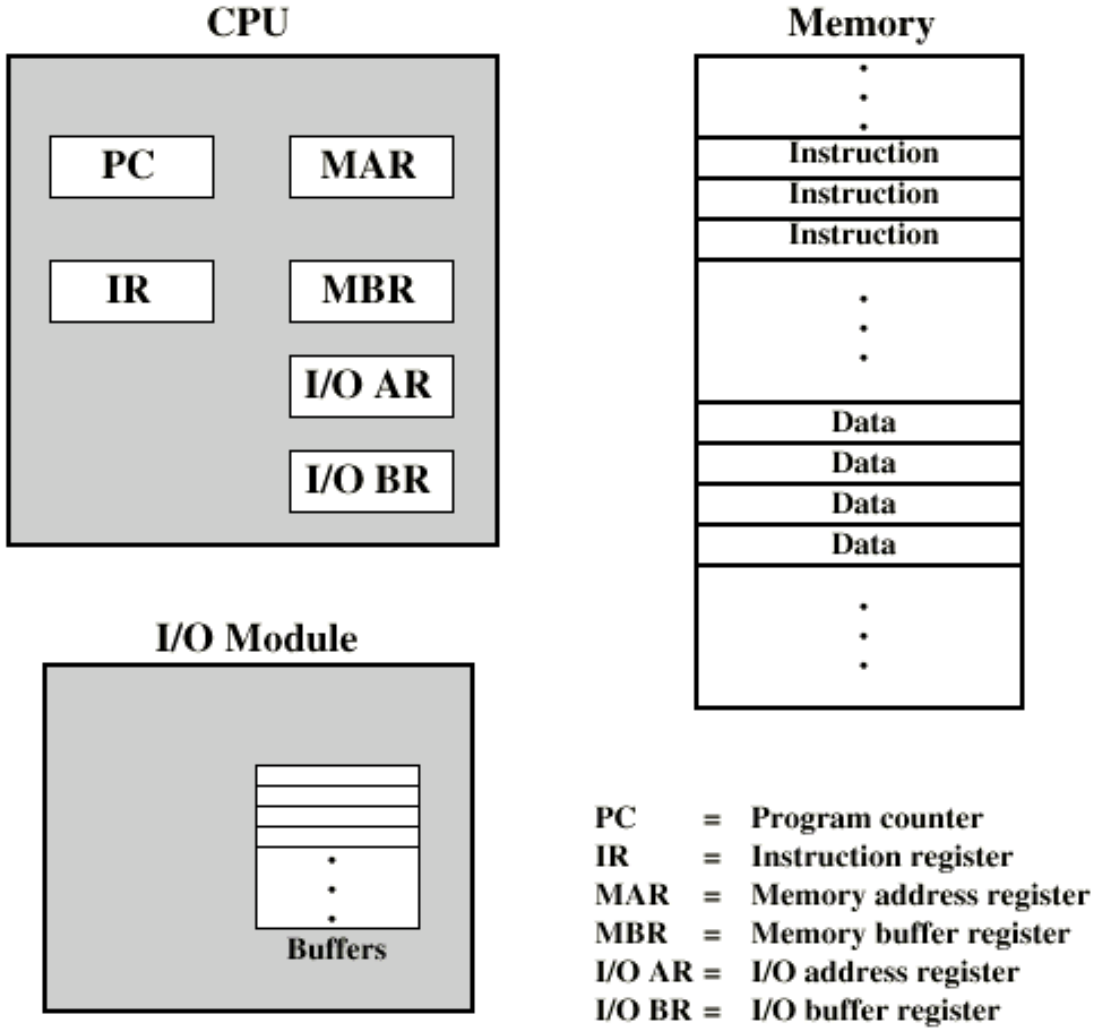
- For each operation a unique code is provided
  - e.g. ADD, MOVE
- A hardware segment accepts the code and issues the control signals

- We have a computer!

Components

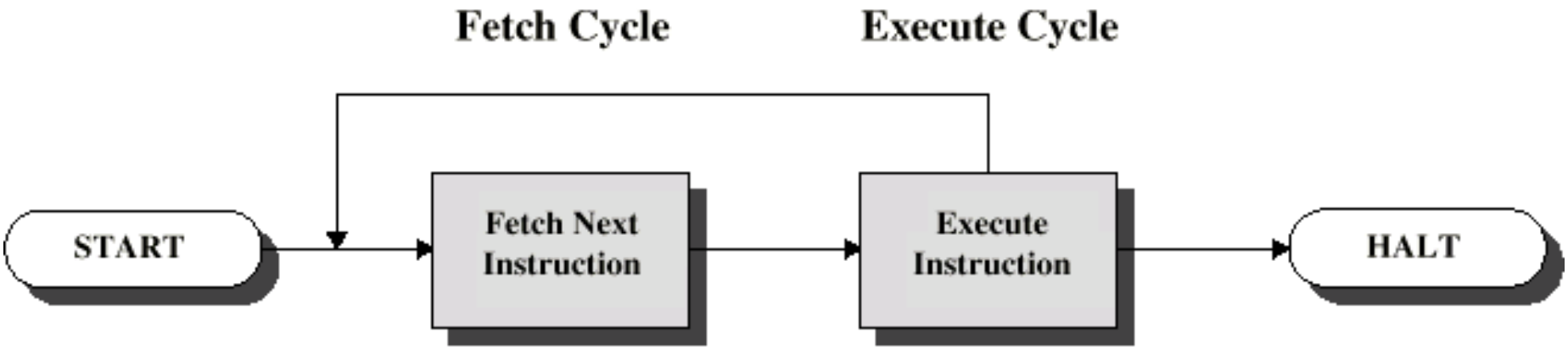
- The Control Unit and the Arithmetic and Logic Unit constitute the Central Processing Unit
- Data and instructions need to get into the system and results out
  - Input/output
- Temporary storage of code and results is needed
  - Main memory

Computer Components: Top Level View



Instruction Cycle

- Two steps:
  - Fetch
  - Execute



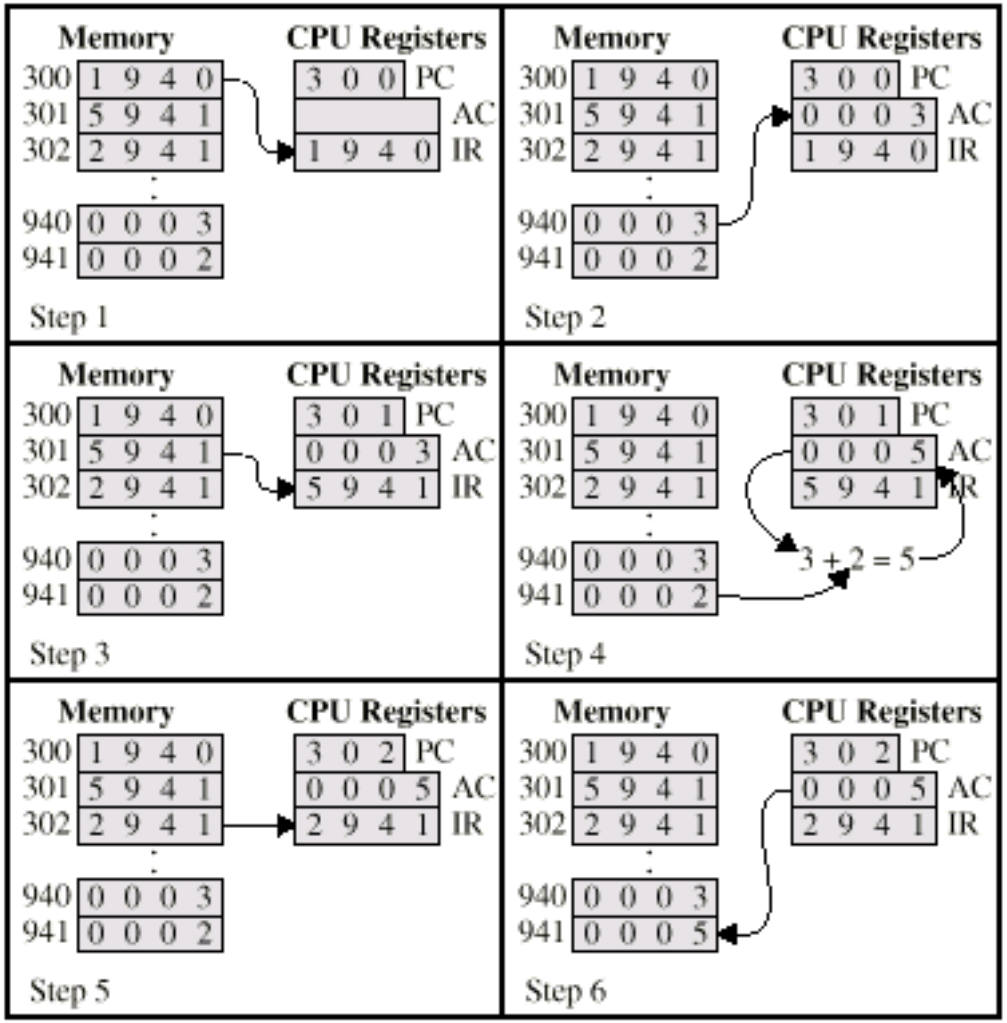
Fetch Cycle

- Program Counter (PC) holds address of next instruction to fetch
- Processor fetches instruction from memory location pointed to by PC
- Increment PC
  - Unless told otherwise
- Instruction loaded into Instruction Register (IR)
- Processor interprets instruction and performs required actions

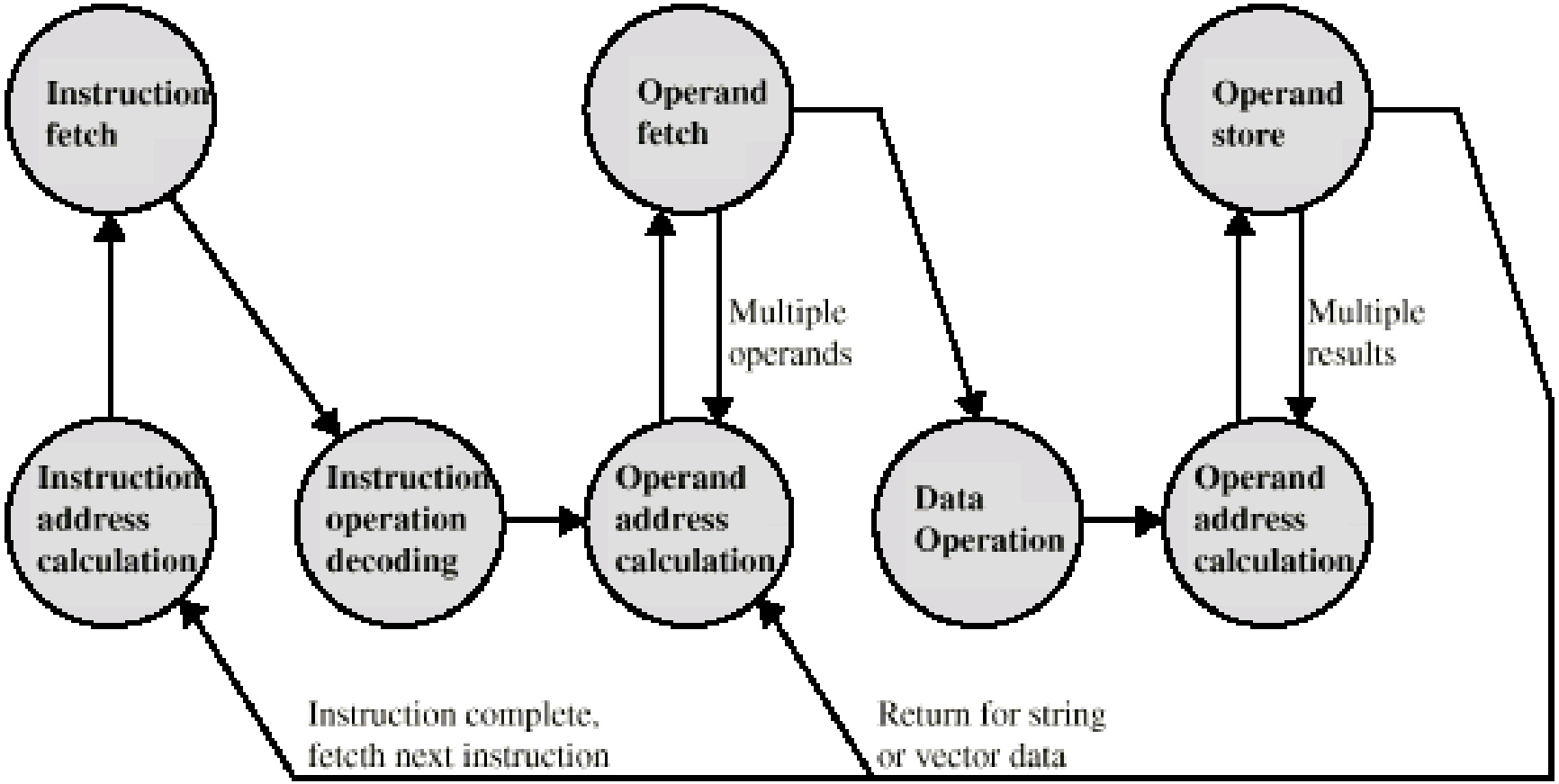
Execute Cycle

- Processor-memory
  - data transfer between CPU and main memory
- Processor I/O
  - Data transfer between CPU and I/O module
- Data processing
  - Some arithmetic or logical operation on data
- Control
  - Alteration of sequence of operations
  - e.g. jump
- Combination of above

Example of Program Execution



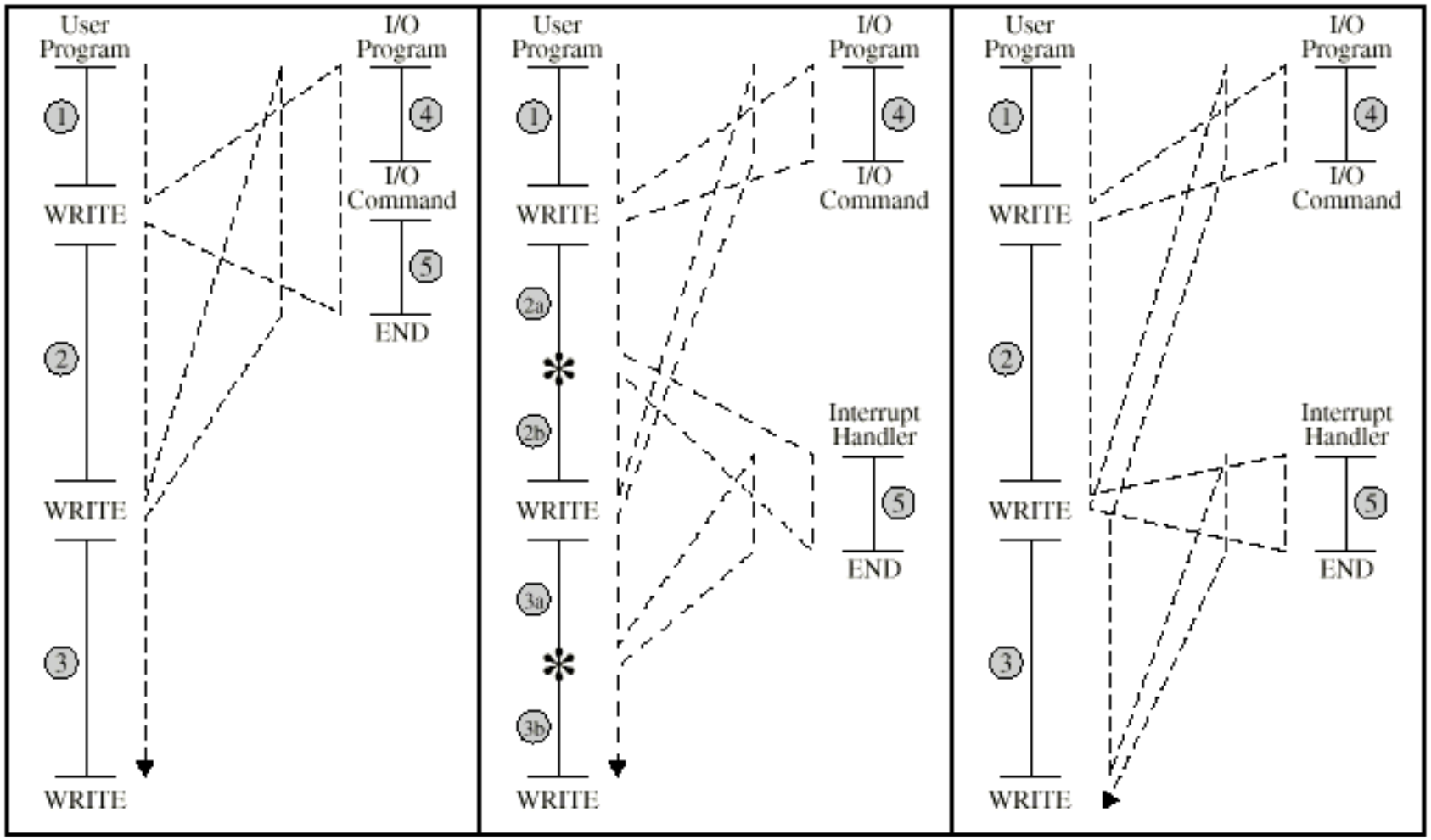
Instruction Cycle - State Diagram



Interrupts

- Mechanism by which other modules (e.g. I/O) may interrupt normal sequence of processing
- Program
  - e.g. overflow, division by zero
- Timer
  - Generated by internal processor timer
  - Used in pre-emptive multi-tasking
- I/O
  - from I/O controller
- Hardware failure
  - e.g. memory parity error

Program Flow Control



(a) No interrupts

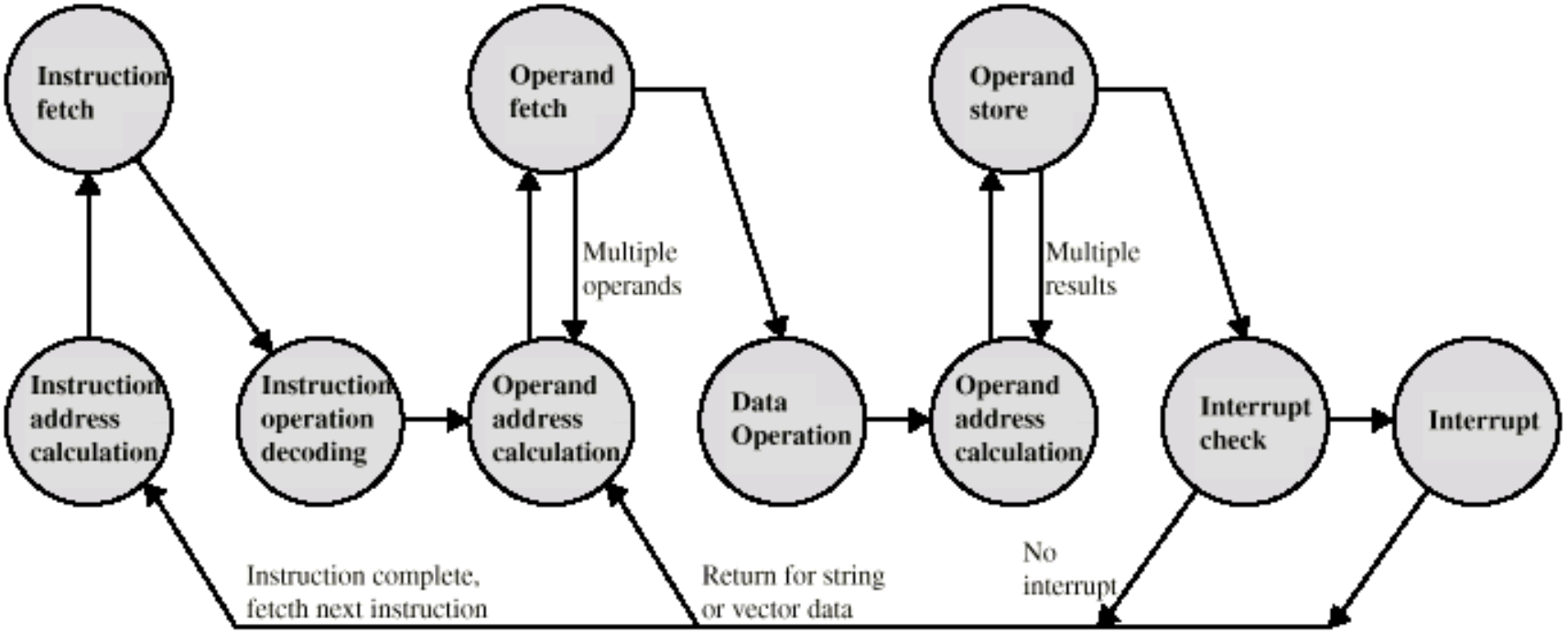
(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait

Interrupt Cycle

- Added to instruction cycle
- Processor checks for interrupt
  - Indicated by an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending:
  - Suspend execution of current program
  - Save context
  - Set PC to start address of interrupt handler routine
  - Process interrupt
  - Restore context and continue interrupted program

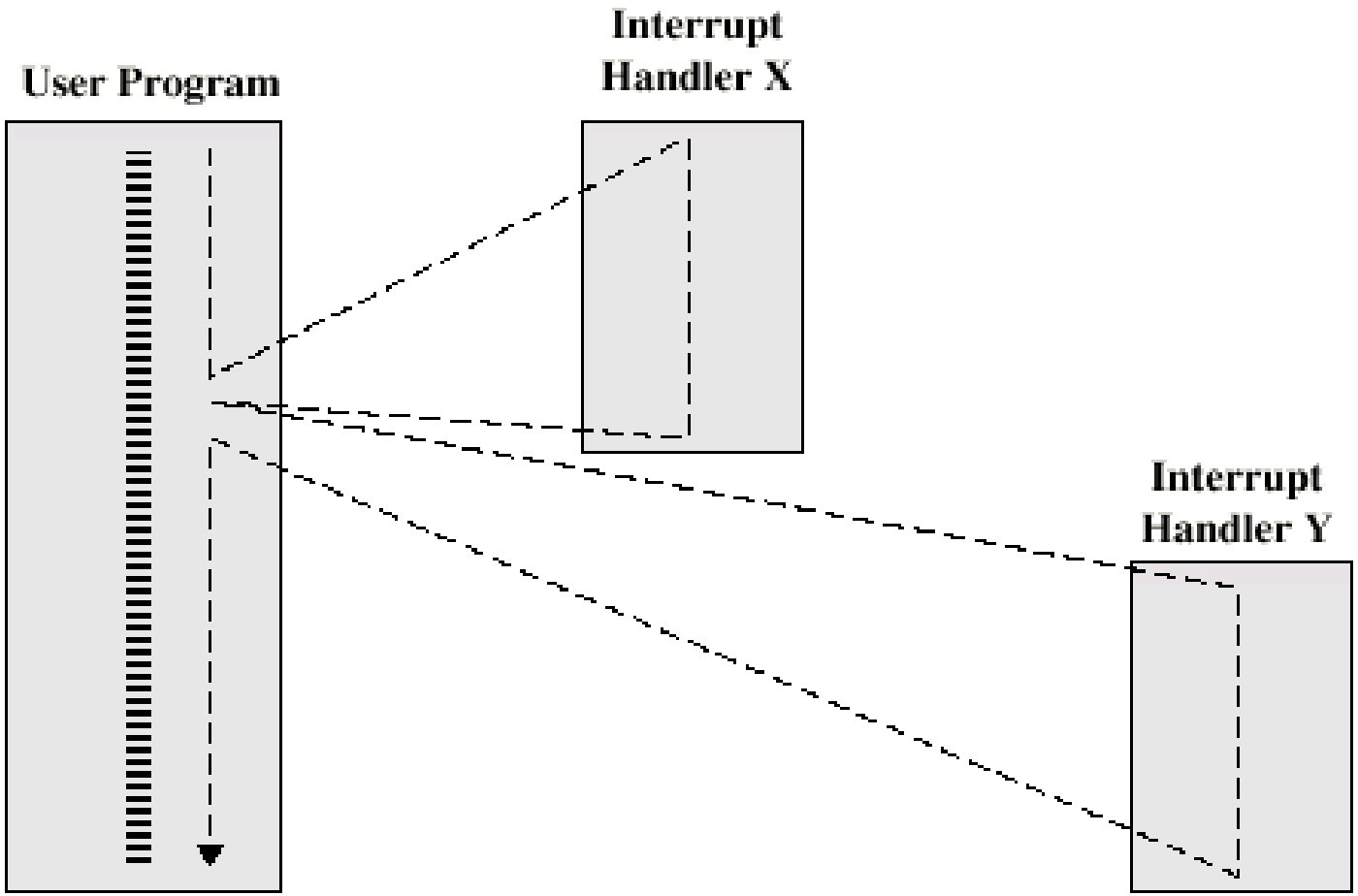
Instruction Cycle (with Interrupts) - State Diagram



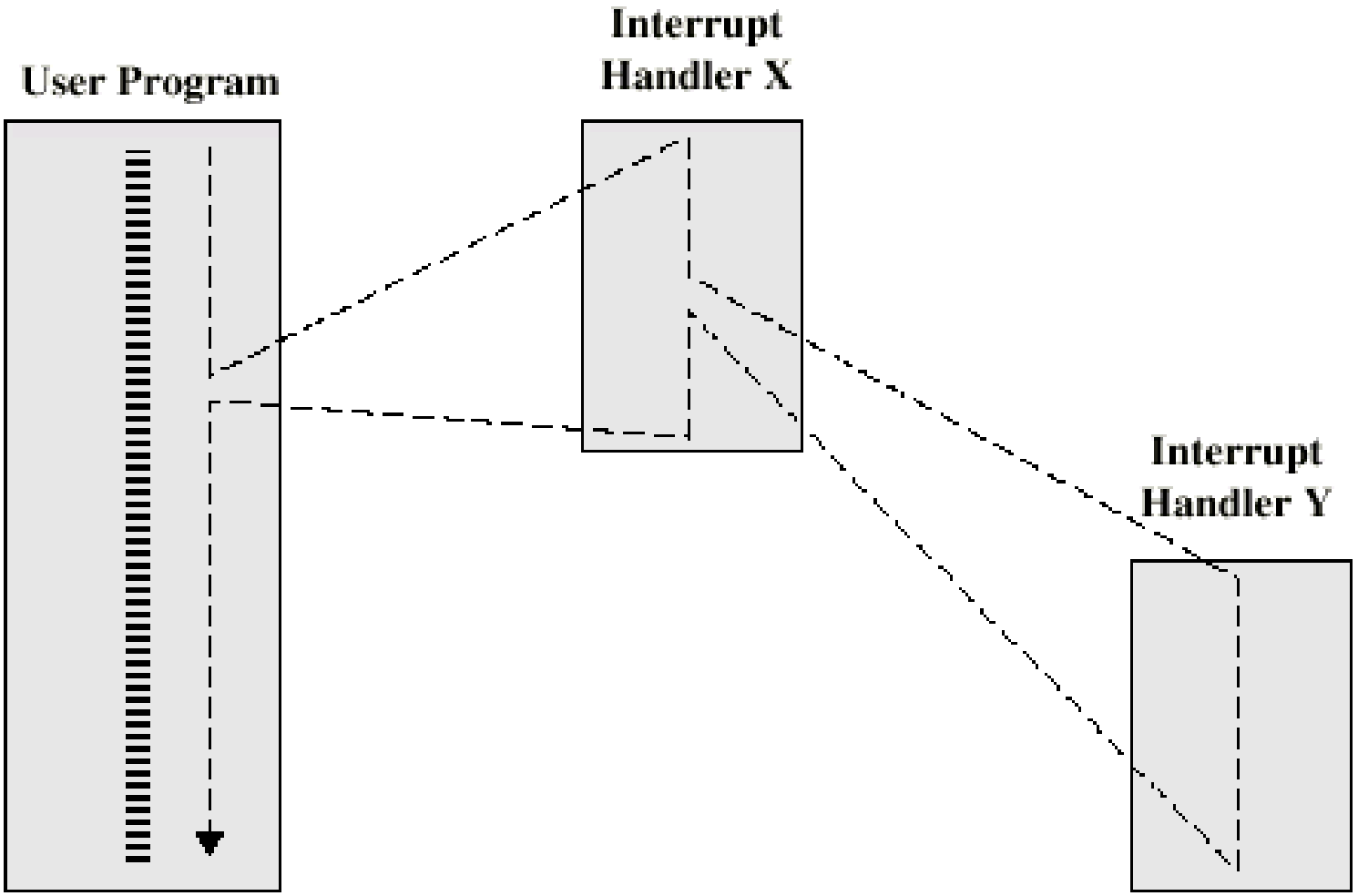
Multiple Interrupts

- Disable interrupts
  - Processor will ignore further interrupts whilst processing one interrupt
  - Interrupts remain pending and are checked after first interrupt has been processed
  - Interrupts handled in sequence as they occur
- Define priorities
  - Low priority interrupts can be interrupted by higher priority interrupts
  - When higher priority interrupt has been processed, processor returns to previous interrupt

Multiple Interrupts - Sequential



Multiple Interrupts - Nested



Connecting

- All the units must be connected
- Different type of connection for different type of unit
  - Memory
  - Input/Output
  - CPU

Memory Connection

- Receives and sends data
- Receives addresses (of locations)
- Receives control signals
  - Read
  - Write
  - Timing

Input/Output Connection(1)

- Similar to memory from computer's viewpoint
- Output
  - Receive data from computer
  - Send data to peripheral
- Input
  - Receive data from peripheral
  - Send data to computer

Input/Output Connection(2)

- Receive control signals from computer
- Send control signals to peripherals
  - e.g. spin disk
- Receive addresses from computer
  - e.g. port number to identify peripheral
- Send interrupt signals (control)

CPU Connection

- Reads instruction and data
- Writes out data (after processing)
- Sends control signals to other units
- Receives (& acts on) interrupts

Buses

- There are a number of possible interconnection systems
- Single and multiple BUS structures are most common
- e.g. Control/Address/Data bus (PC)
- e.g. Unibus (DEC-PDP)

What is a Bus?

- A communication pathway connecting two or more devices
- Usually broadcast
- Often grouped
  - A number of channels in one bus
  - e.g. 32 bit data bus is 32 separate single bit channels
- Power lines may not be shown

Data Bus

- Carries data
  - Remember that there is no difference between “data” and “instruction” at this level
- Width is a key determinant of performance
  - 8, 16, 32, 64 bit

Address bus

- Identify the source or destination of data
- e.g. CPU needs to read an instruction (data) from a given location in memory
- Bus width determines maximum memory capacity of system
  - e.g. 8080 has 16 bit address bus giving 64k address space

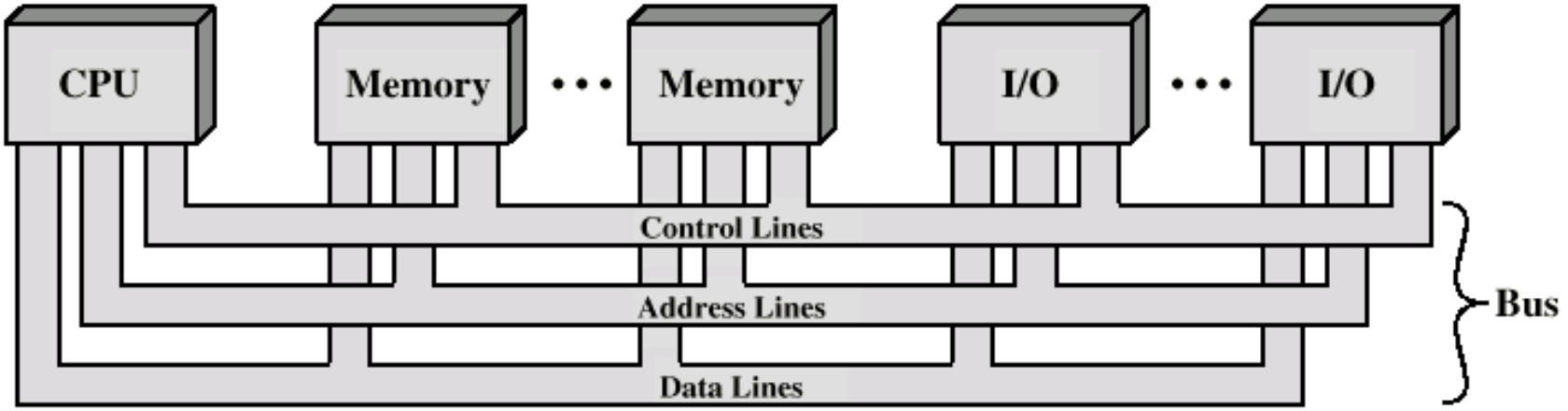
Control Bus

- Control and timing information
  - Memory read/write signal
  - Interrupt request
  - Clock signals

Control Bus

- Typical control lines include
  - Memory write
  - Memory read
  - I/O write
  - I/O read
  - Transfer ACK
  - Bus request
  - Bus grant
  - Interrupt request
  - Interrupt ACK
  - Clock
  - Reset

Bus Interconnection Scheme



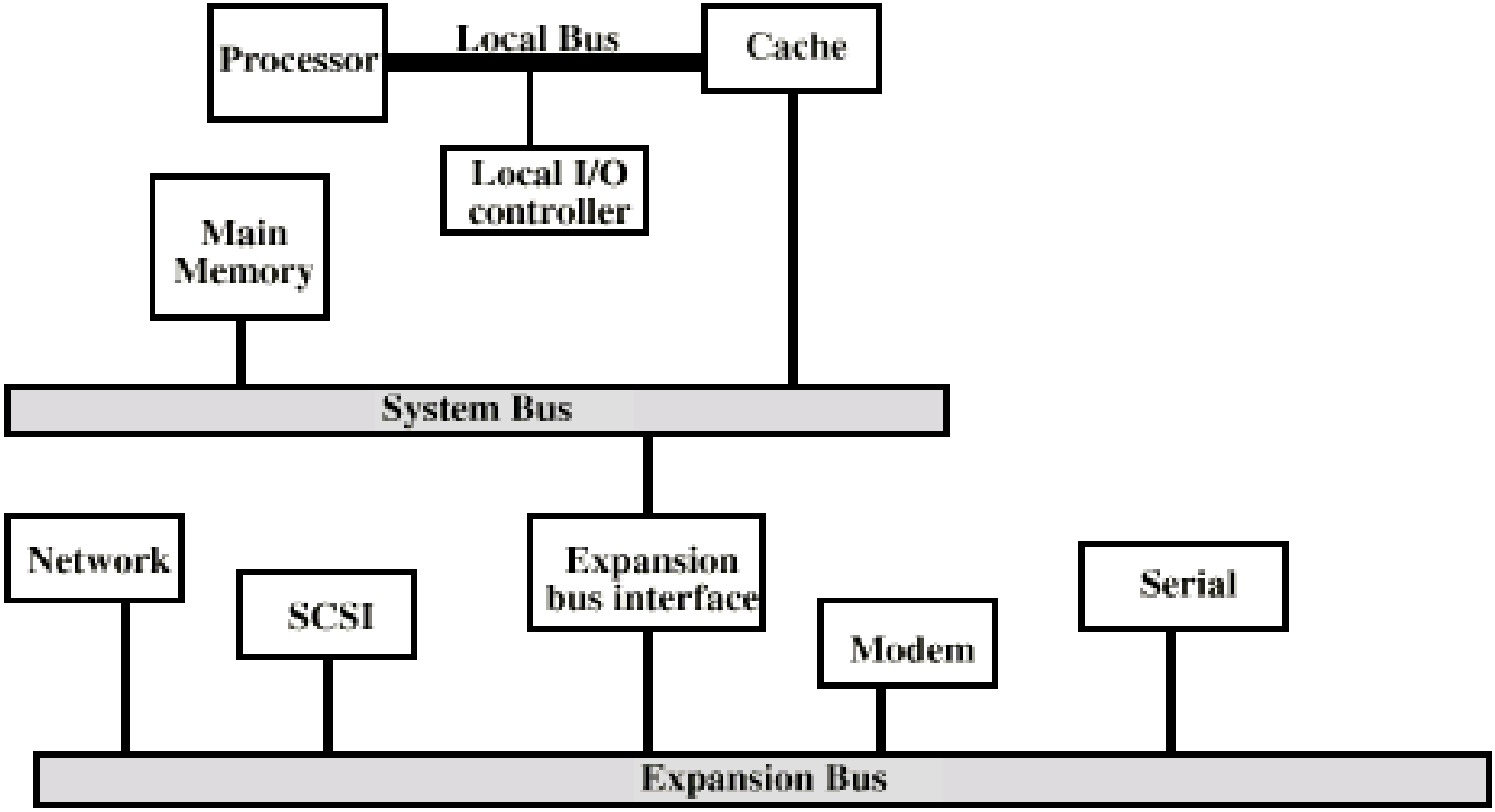
Big and Yellow?

- What do buses look like?
  - Parallel lines on circuit boards
  - Ribbon cables
  - Strip connectors on mother boards
    - e.g. PCI
  - Sets of wires

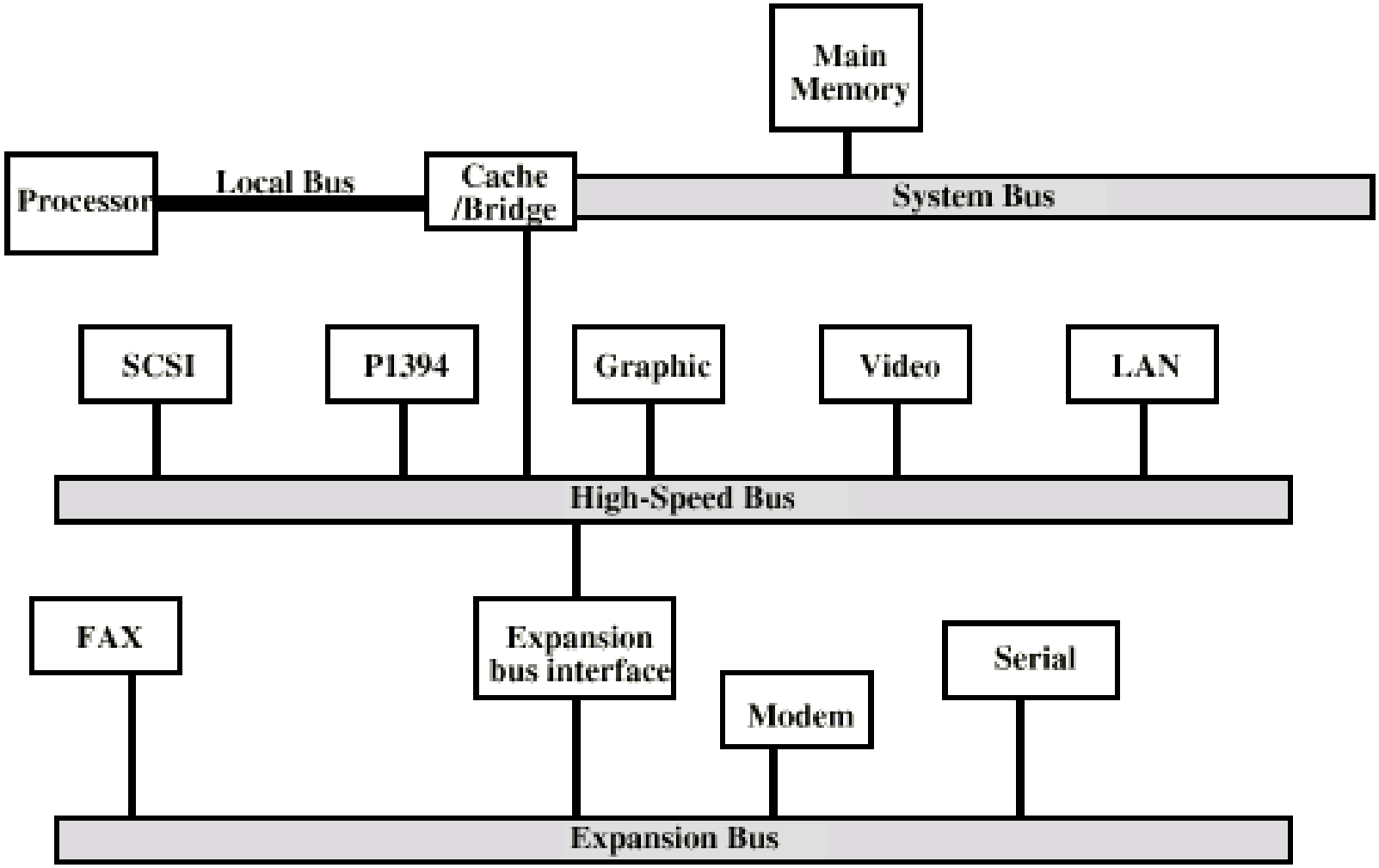
Single Bus Problems

- Lots of devices on one bus leads to:
  - Propagation delays
    - Long data paths mean that co-ordination of bus use can adversely affect performance
    - If aggregate data transfer approaches bus capacity
- Most systems use multiple buses to overcome these problems

Traditional (ISA) (with cache)



High Performance Bus



Bus Types

- Dedicated
  - Separate data & address lines
- Multiplexed
  - Shared lines
  - Address valid or data valid control line
  - Advantage - fewer lines
  - Disadvantages
    - More complex control
    - Ultimate performance

Elements of Bus Design

- Type
  - Dedicated Address
  - Multiplexed Data
- Bus Width
  - Address
  - Data
- Method of Arbitration
  - Centralized Read
  - Distributed Write
- Timing Read-modify-write
  - Synchronous Read-after-write
  - Asynchronous Block

Elements of Bus Design

- **Data Transfer Type**
  - Read
  - Write
  - Read-modify-write
  - Read-after-write
  - Block

Bus Arbitration

- More than one module controlling the bus
- e.g. CPU and DMA controller
- Only one module may control bus at one time
- Arbitration may be centralised or distributed

Centralised Arbitration

- Single hardware device controlling bus access
  - Bus Controller
  - Arbiter
- May be part of CPU or separate

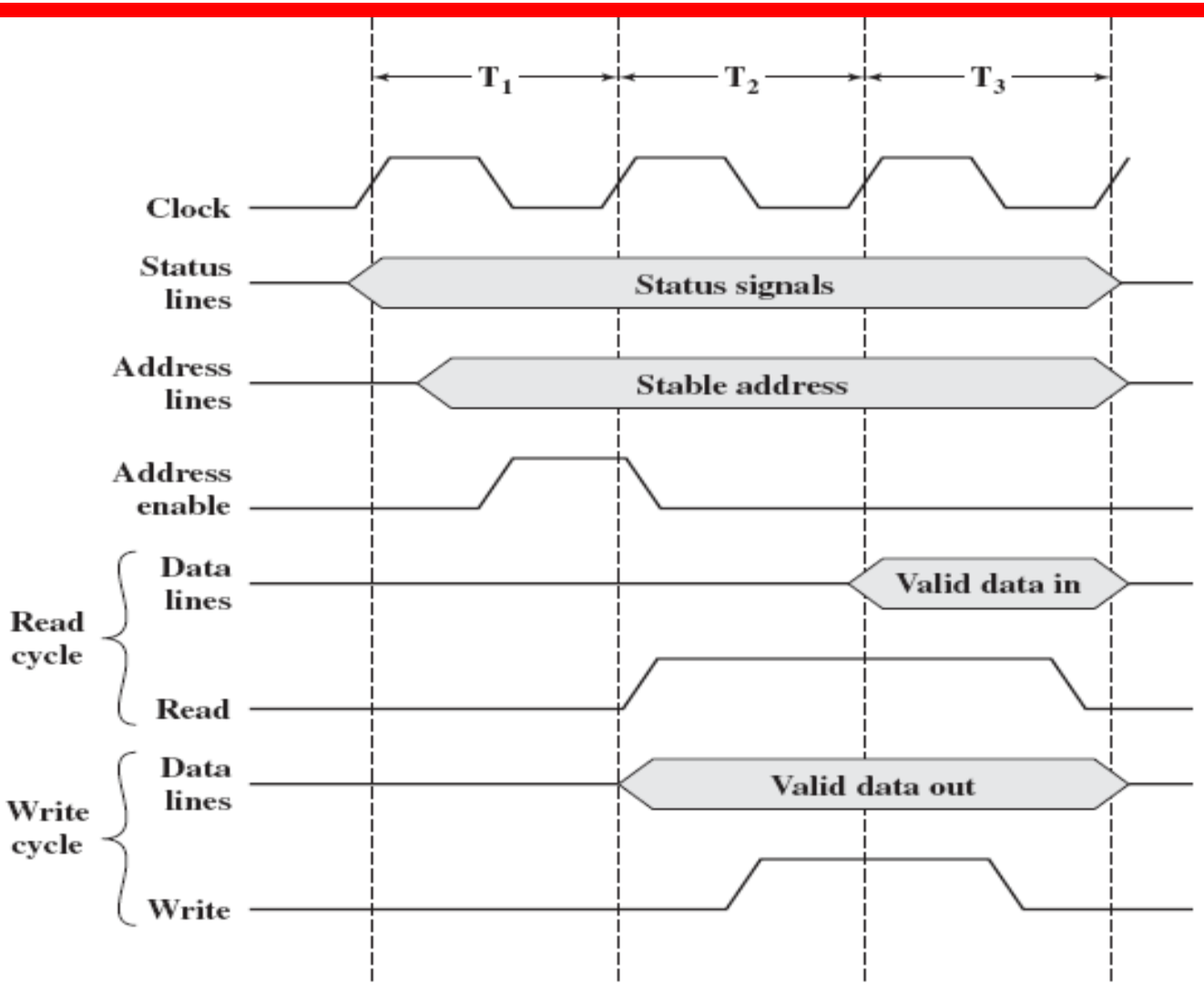
Distributed Arbitration

- Each module may claim the bus
- Control logic on all modules

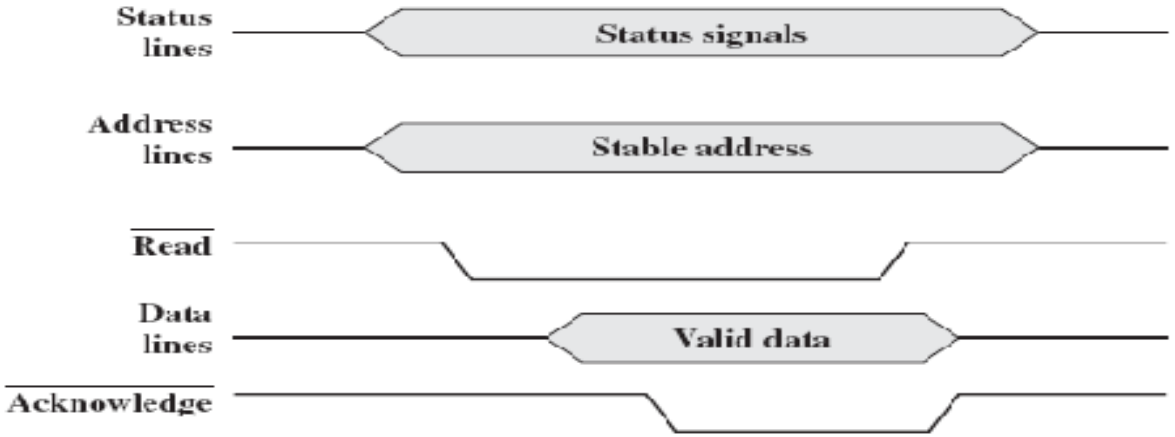
Timing

- Co-ordination of events on bus
- Synchronous
  - Events determined by clock signals
  - Control Bus includes clock line
  - A single 1-0 is a bus cycle
  - All devices can read clock line
  - Usually sync on leading edge
  - Usually a single cycle for an event

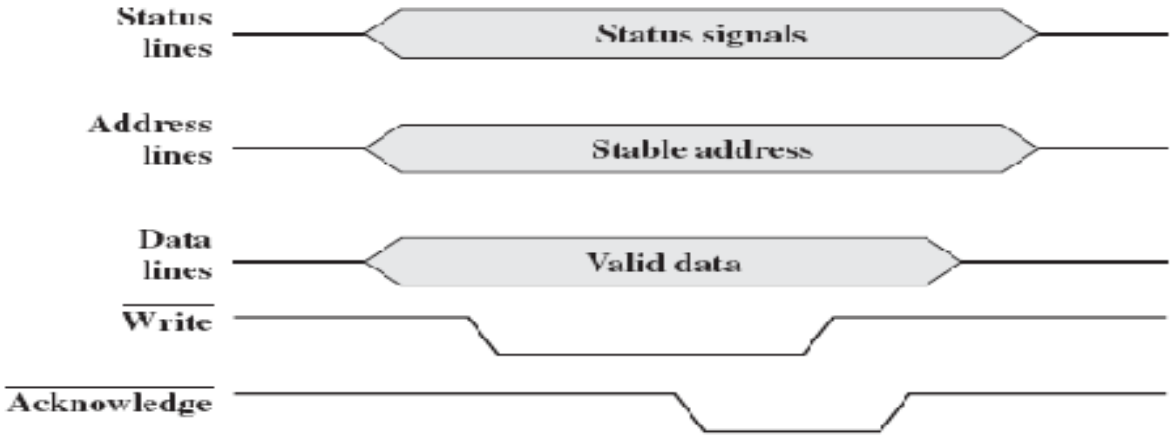
Synchronous Timing Diagram



Asynchronous Timing Diagram



(a) System bus read cycle



(b) System bus write cycle

PCI Bus

- Peripheral Component Interconnection
- Intel released to public domain
- 32 or 64 bit
- 50 lines

PCI Bus Lines (required)

- Systems lines
  - Including clock and reset
- Address & Data
  - 32 time mux lines for address/data
  - Interrupt & validate lines
- Interface Control
- Arbitration
  - Not shared
  - Direct connection to PCI bus arbiter
- Error lines

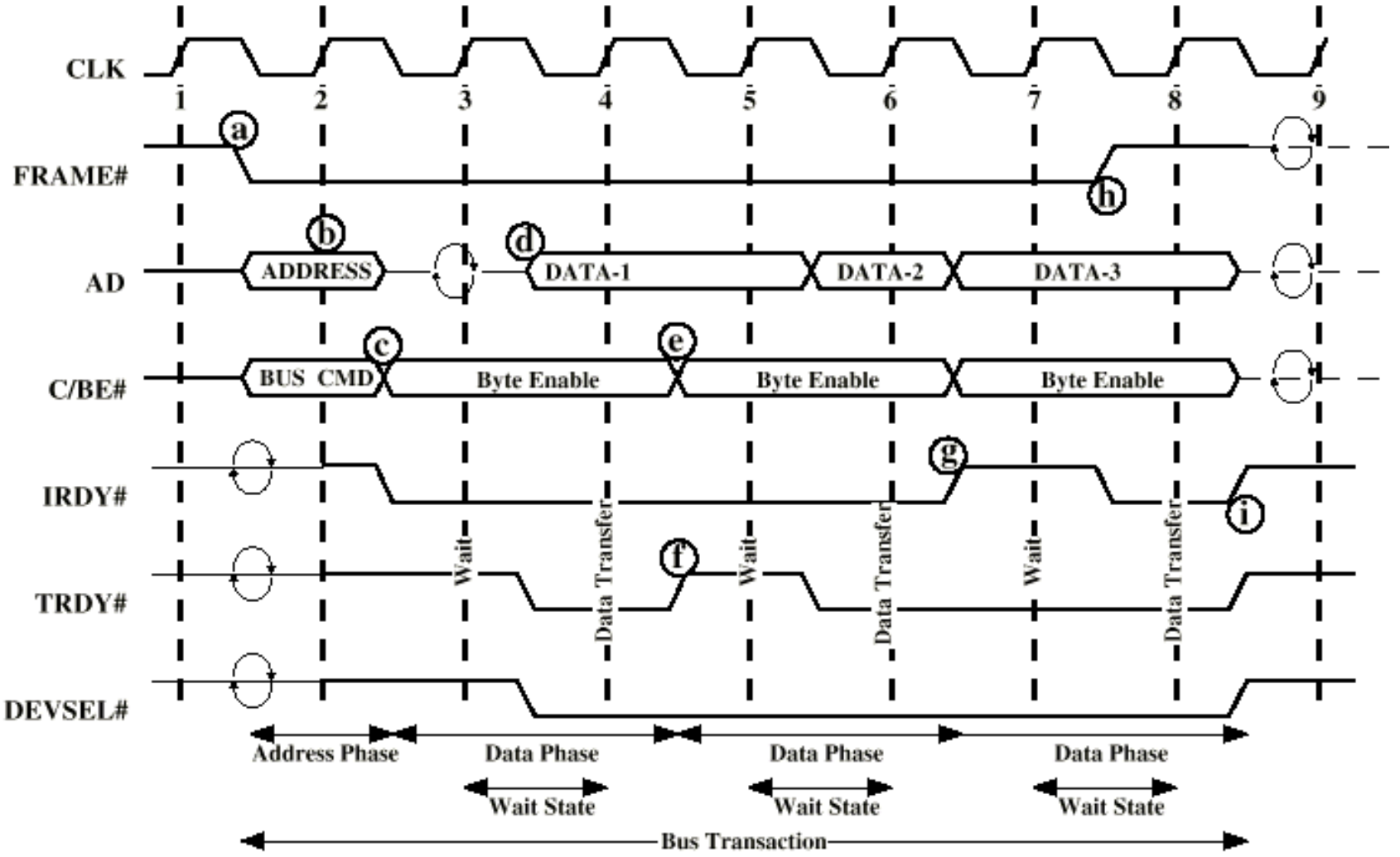
PCI Bus Lines (Optional)

- Interrupt lines
  - Not shared
- Cache support
- 64-bit Bus Extension
  - Additional 32 lines
  - Time multiplexed
  - 2 lines to enable devices to agree to use 64-bit transfer
- JTAG/Boundary Scan
  - For testing procedures

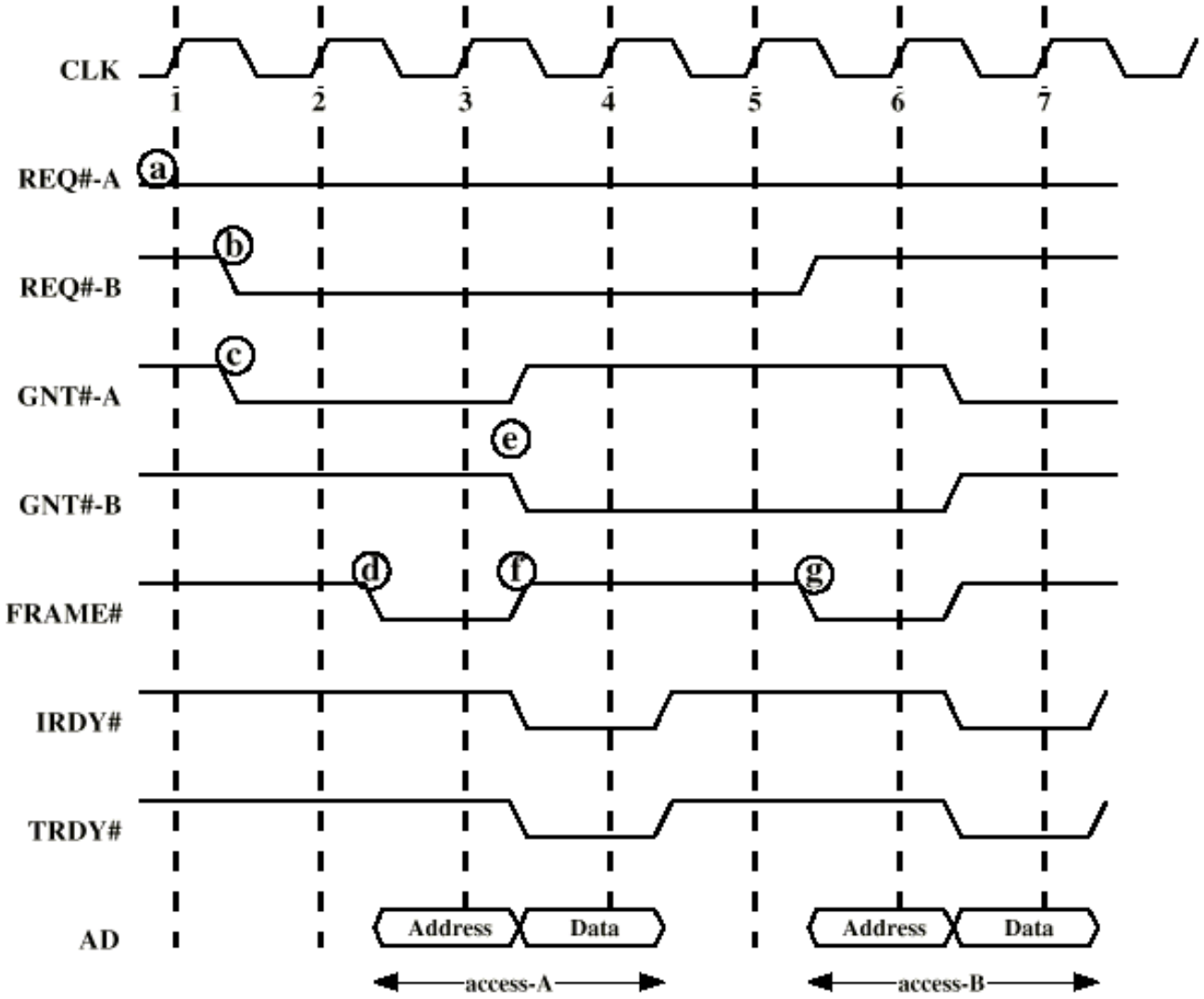
PCI Commands

- Transaction between initiator (master) and target
- Master claims bus
- Determine type of transaction
  - e.g. I/O read/write
- Address phase
- One or more data phases

PCI Read Timing Diagram



PCI Bus Arbitration



Foreground Reading

- Stallings, chapter 3 (all of it)
- www.pcguide.com/ref/mbsys/buses/

- In fact, read the whole site!
- www.pcguide.com/

William Stallings Computer Organization and Architecture

Chapter 4

Internal Memory

Characteristics

- Location
- Capacity
- Unit of transfer
- Access method
- Performance
- Physical type
- Physical characteristics
- Organisation

Location

- CPU
- Internal
- External

Capacity

- Word size
  - The natural unit of organisation
- Number of words
  - or Bytes

Unit of Transfer

- Internal
  - Usually governed by data bus width
- External
  - Usually a block which is much larger than a word
- Addressable unit
  - Smallest location which can be uniquely addressed
  - Word internally
  - Cluster on M$ disks

Access Methods (1)

- Sequential
  - Start at the beginning and read through in order
  - Access time depends on location of data and previous location
  - e.g. tape
- Direct
  - Individual blocks have unique address
  - Access is by jumping to vicinity plus sequential search
  - Access time depends on location and previous location
  - e.g. disk

Access Methods (2)

- Random
  - Individual addresses identify locations exactly
  - Access time is independent of location or previous access
  - e.g. RAM
- Associative
  - Data is located by a comparison with contents of a portion of the store
  - Access time is independent of location or previous access
  - e.g. cache

Memory Hierarchy

- Registers
  - In CPU
- Internal or Main memory
  - May include one or more levels of cache
  - “RAM”
- External memory
  - Backing store

Performance

- Access time
  - Time between presenting the address and getting the valid data
- Memory Cycle time
  - Time may be required for the memory to "recover" before next access
  - Cycle time is access + recovery
- Transfer Rate
  - Rate at which data can be moved

Physical Types

- Semiconductor
  - RAM
- Magnetic
  - Disk & Tape
- Optical
  - CD & DVD
- Others
  - Bubble
  - Hologram

Physical Characteristics

- Decay
- Volatility
- Erasable
- Power consumption

Organisation

- Physical arrangement of bits into words
- Not always obvious
- e.g. interleaved

The Bottom Line

- How much?
  - Capacity
- How fast?
  - Time is money
- How expensive?

Hierarchy List

- Registers
- L1 Cache
- L2 Cache
- Main memory
- Disk cache
- Disk
- Optical
- Tape

So you want fast?

- It is possible to build a computer which uses only static RAM (see later)
- This would be very fast
- This would need no cache
  - How can you cache cache?
- This would cost a very large amount

Locality of Reference

- During the course of the execution of a program, memory references tend to cluster
- e.g. loops

Semiconductor Memory

- RAM
  - Misnamed as all semiconductor memory is random access
  - Read/Write
  - Volatile
  - Temporary storage
  - Static or dynamic

Dynamic RAM

- Bits stored as charge in capacitors
- Charges leak
- Need refreshing even when powered
- Simpler construction
- Smaller per bit
- Less expensive
- Need refresh circuits
- Slower
- Main memory

Static RAM

- Bits stored as on/off switches
- No charges to leak
- No refreshing needed when powered
- More complex construction
- Larger per bit
- More expensive
- Does not need refresh circuits
- Faster
- Cache

Read Only Memory (ROM)

- Permanent storage
- Microprogramming (see later)
- Library subroutines
- Systems programs (BIOS)
- Function tables

Types of ROM

- Written during manufacture
  - Very expensive for small runs
- Programmable (once)
  - PROM
  - Needs special equipment to program
- Read "mostly"
  - Erasable Programmable (EPROM)
    - Erased by UV
  - Electrically Erasable (EEPROM)
    - Takes much longer to write than read
  - Flash memory
    - Erase whole memory electrically

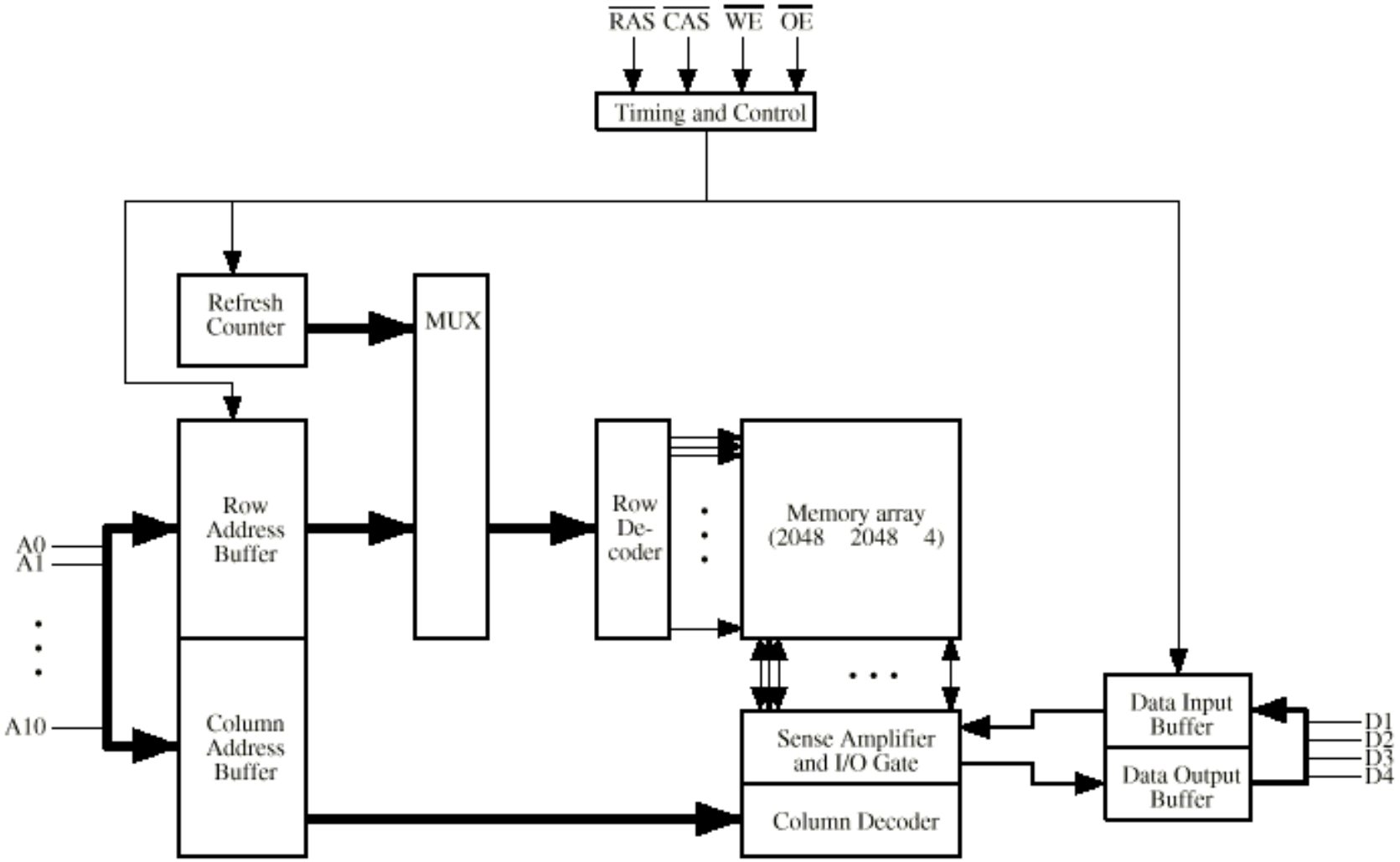
Organisation in detail

- A 16Mbit chip can be organised as 1M of 16 bit words
- A bit per chip system has 16 lots of 1Mbit chip with bit 1 of each word in chip 1 and so on
- A 16Mbit chip can be organised as a 2048 x 2048 x 4bit array
  - Reduces number of address pins
    - Multiplex row address and column address
    - 11 pins to address (2^{11}=2048)
    - Adding one more pin doubles range of values so x4 capacity

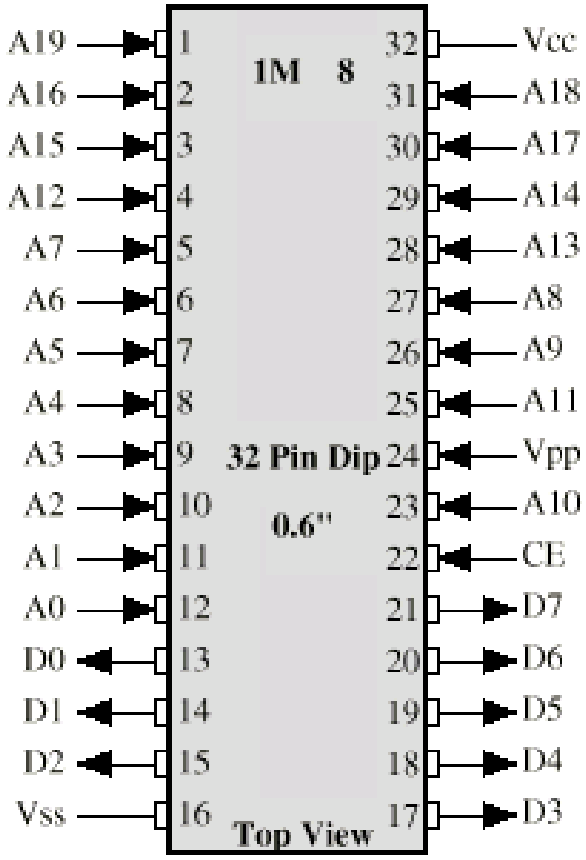
Refreshing

- Refresh circuit included on chip
- Disable chip
- Count through rows
- Read & Write back
- Takes time
- Slows down apparent performance

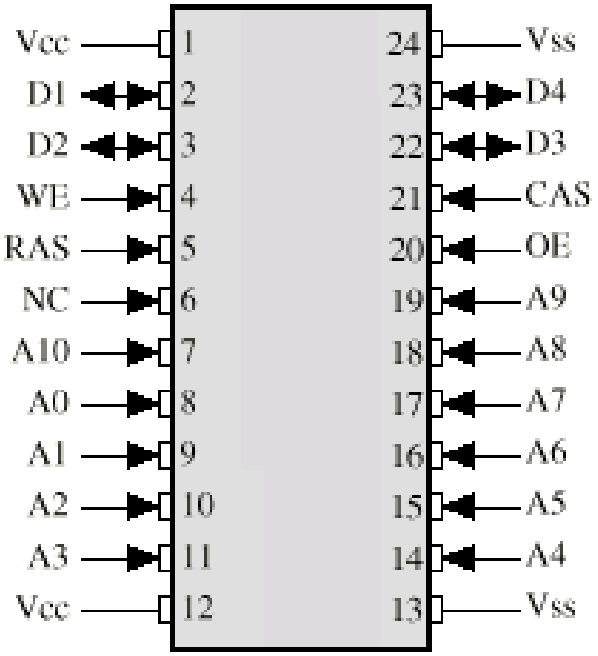
Typical 16 Mb DRAM (4M x 4)



Packaging

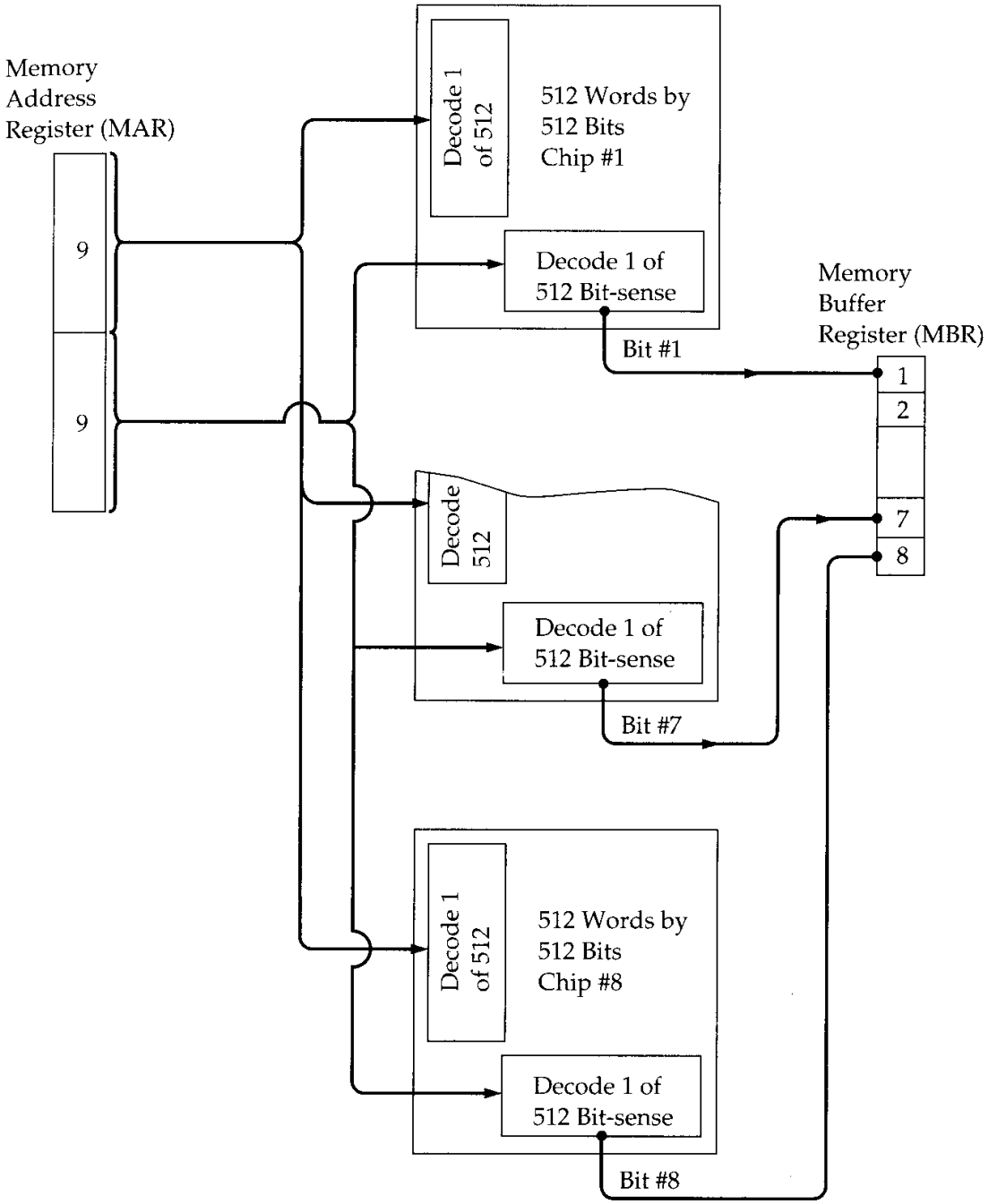


(a) 8 Mbit EPROM

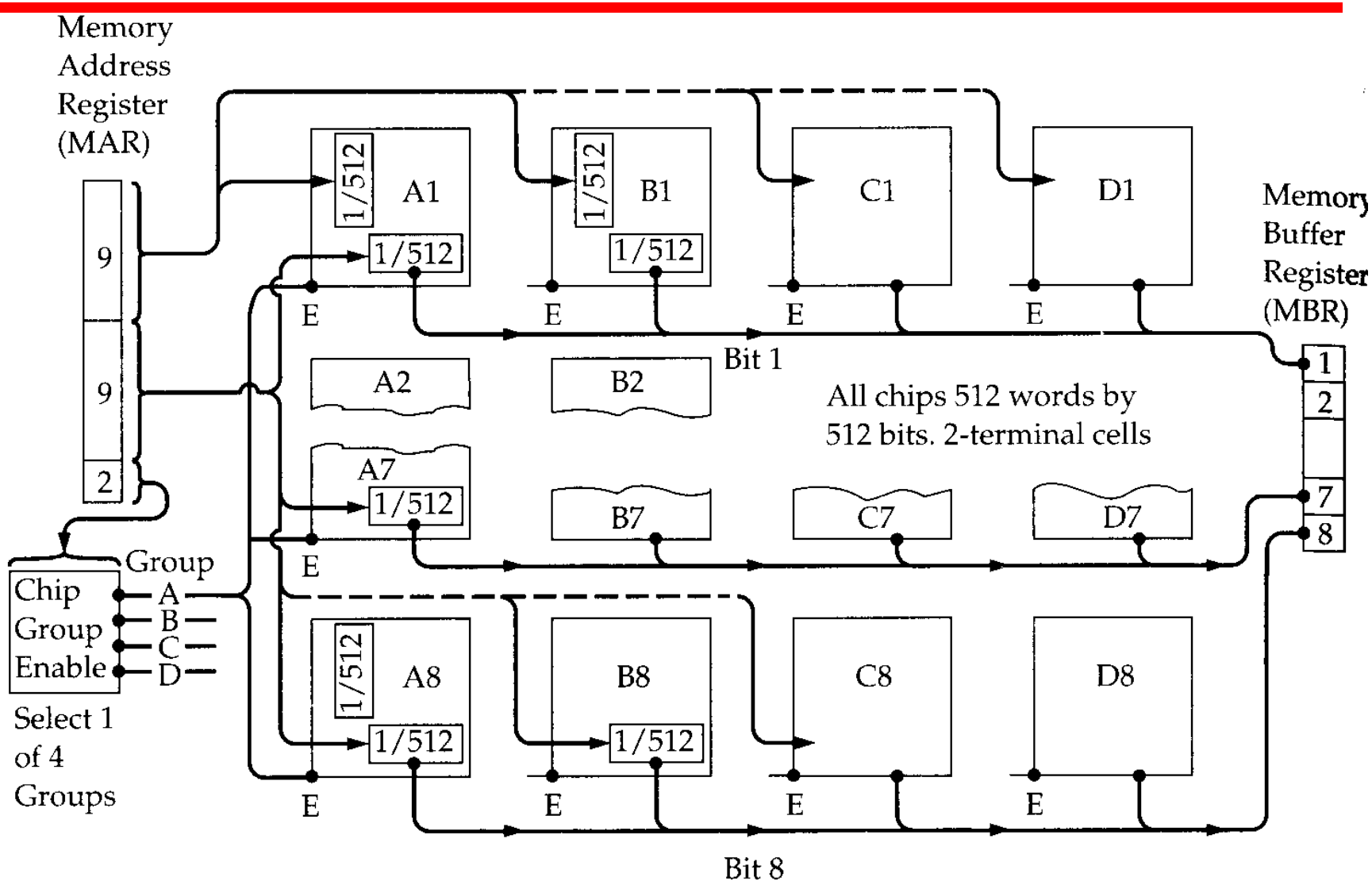


(b) 16 Mbit DRAM

Module Organisation



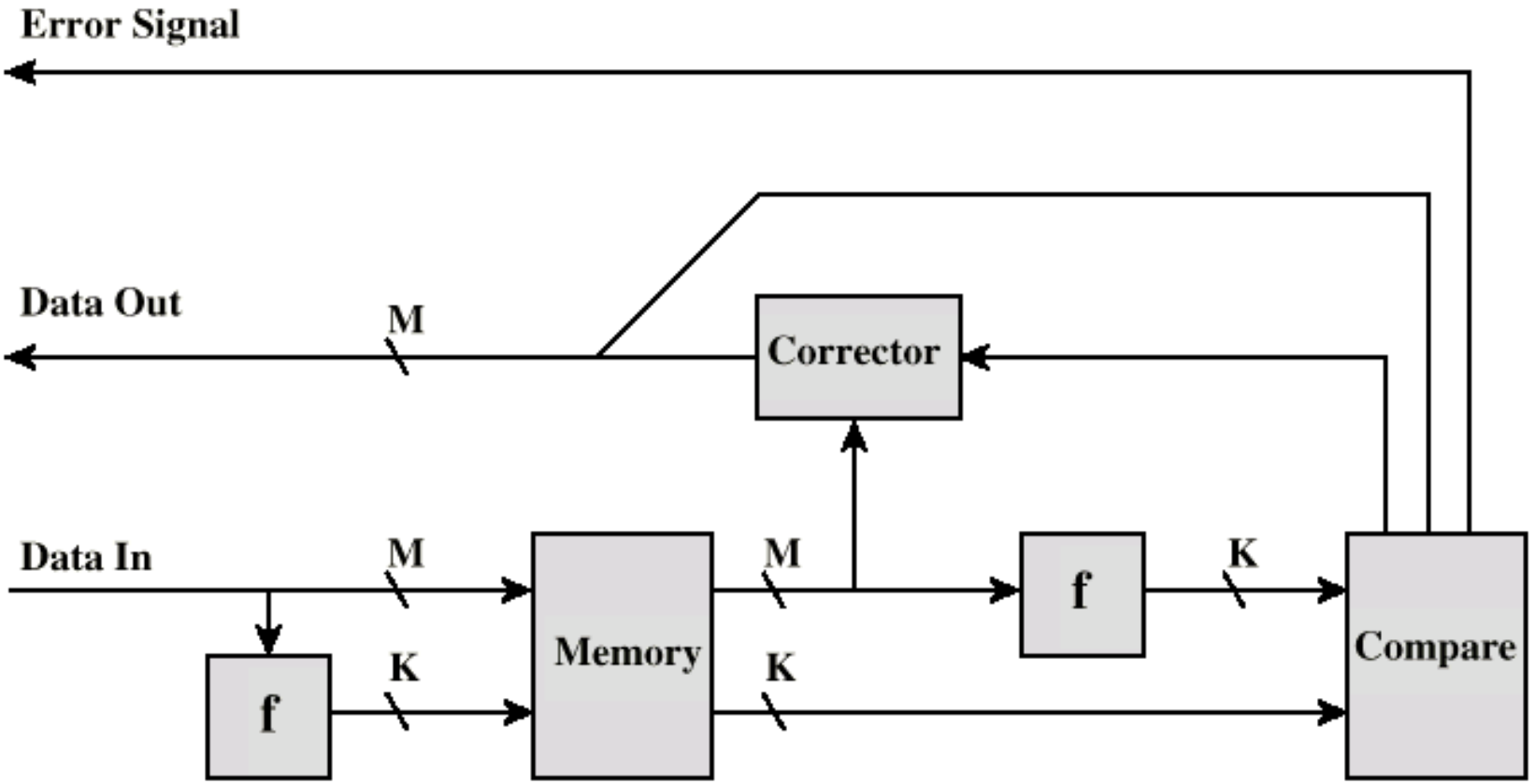
Module Organisation (2)



Error Correction

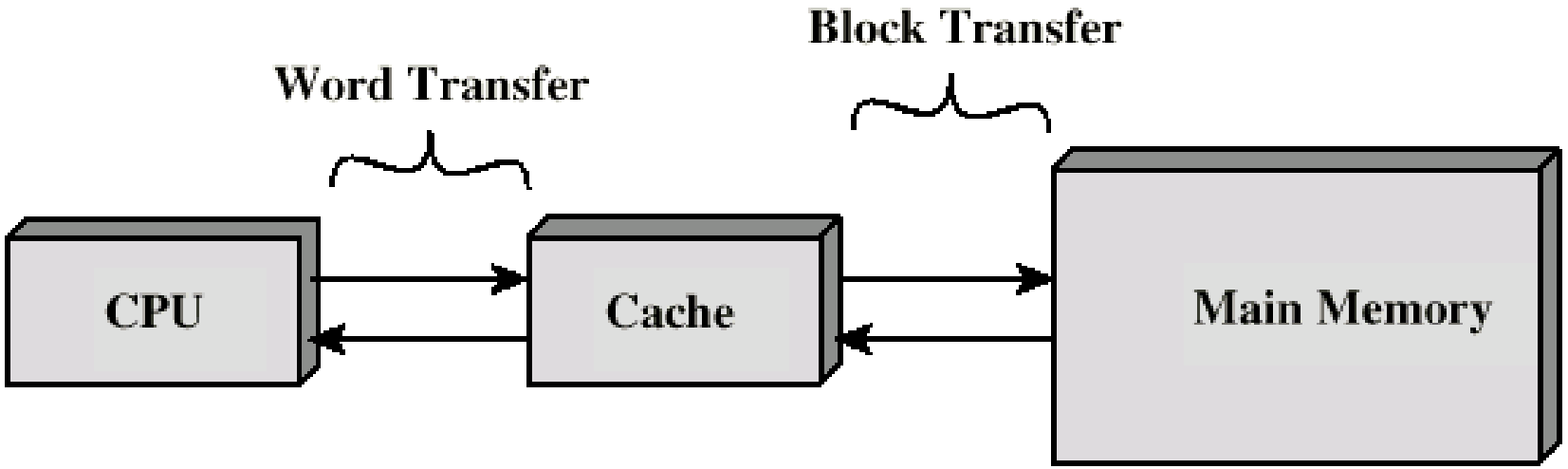
- Hard Failure
  - Permanent defect
- Soft Error
  - Random, non-destructive
  - No permanent damage to memory
- Detected using Hamming error correcting code

Error Correcting Code Function



Cache

- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module



Cache operation - overview

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

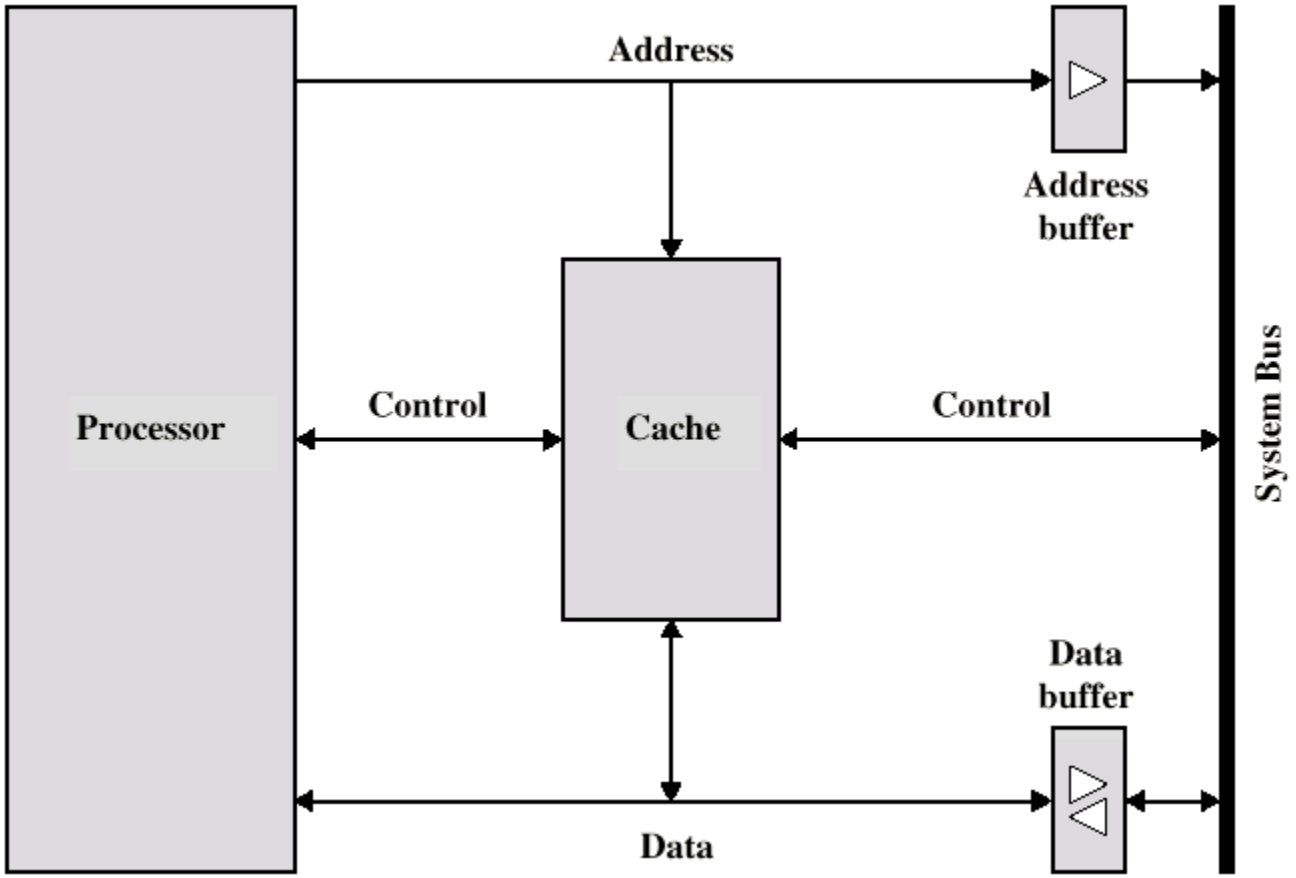
Cache Design

- Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

Size does matter

- Cost
  - More cache is expensive
- Speed
  - More cache is faster (up to a point)
  - Checking cache for data takes time

Typical Cache Organization



Mapping Function

- Cache of 64kByte
- Cache block of 4 bytes
  - i.e. cache is 16k (2^{14}) lines of 4 bytes
- 16MBytes main memory
- 24 bit address
  - (2^{24}=16M)

Direct Mapping

- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
- Address is in two parts
- Least Significant w bits identify unique word
- Most Significant s bits specify one memory block
- The MSBs are split into a cache line field r and a tag of s-r (most significant)

Direct Mapping Address Structure

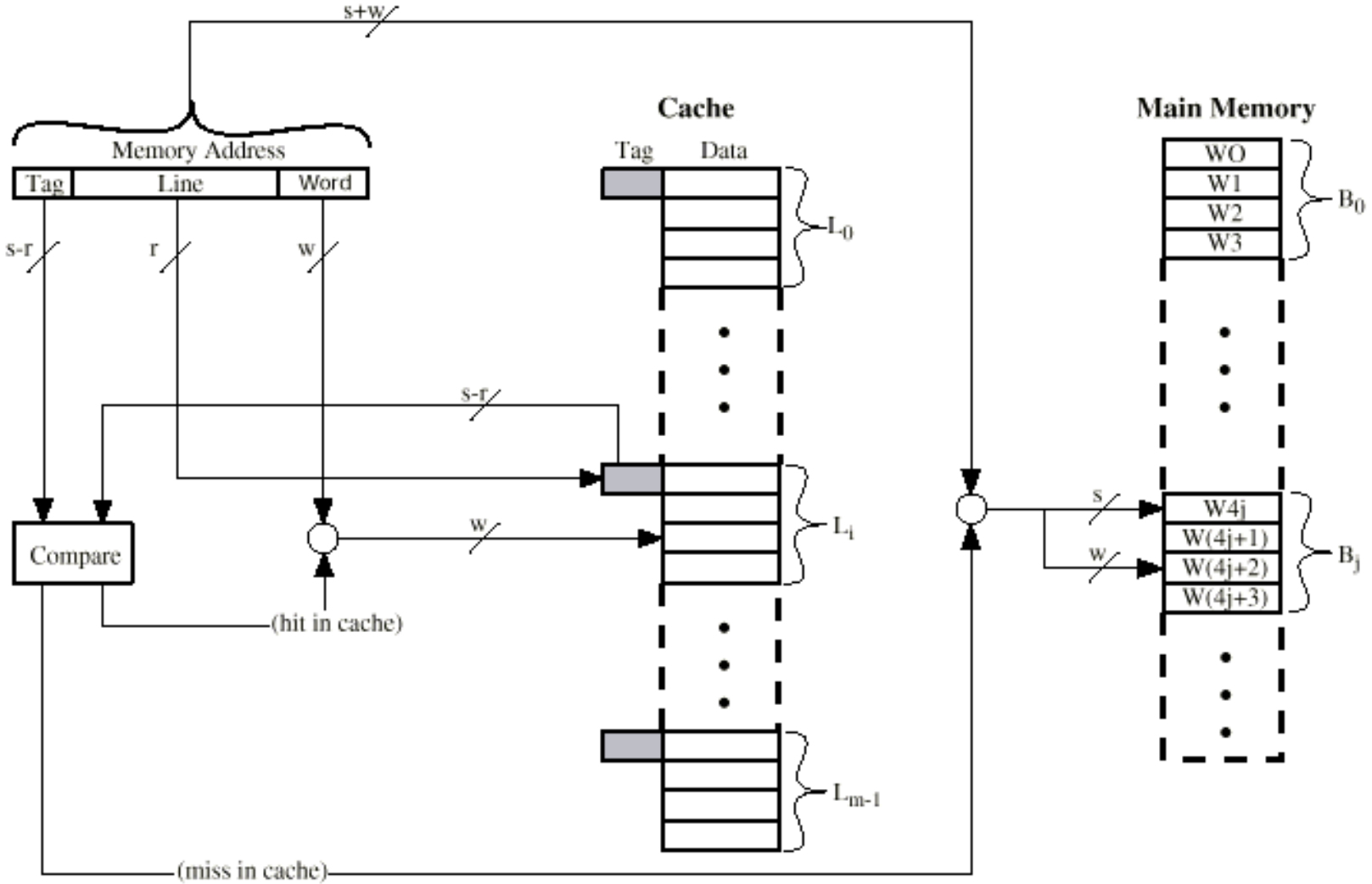
Tag s-r	Line or Slot r	Word w
8	14	2

- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

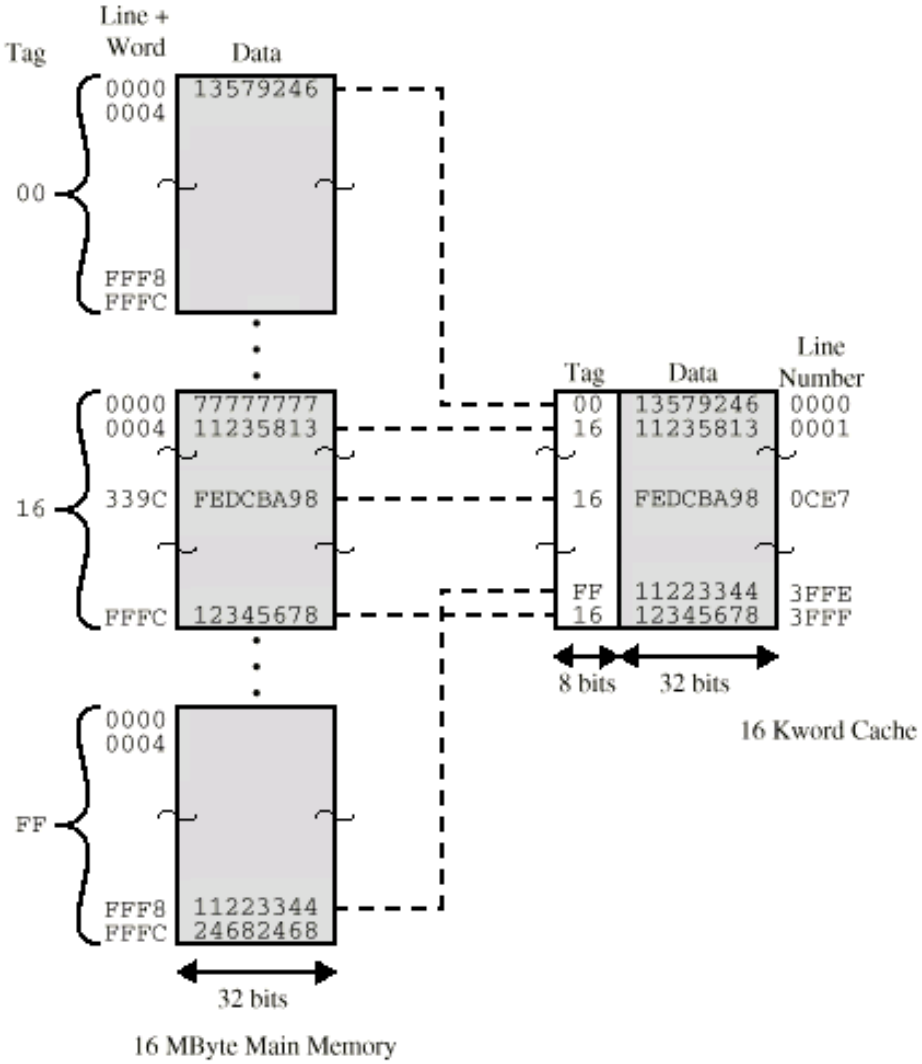
Direct Mapping Cache Line Table

Cache line	Main Memory blocks held
0	0, m, 2m, 3m…2^s-m
1	1,m+1, 2m+1…2^s-m+1
m-1	m-1, 2m-1,3m-1…2^s-1

Direct Mapping Cache Organization



Direct Mapping Example



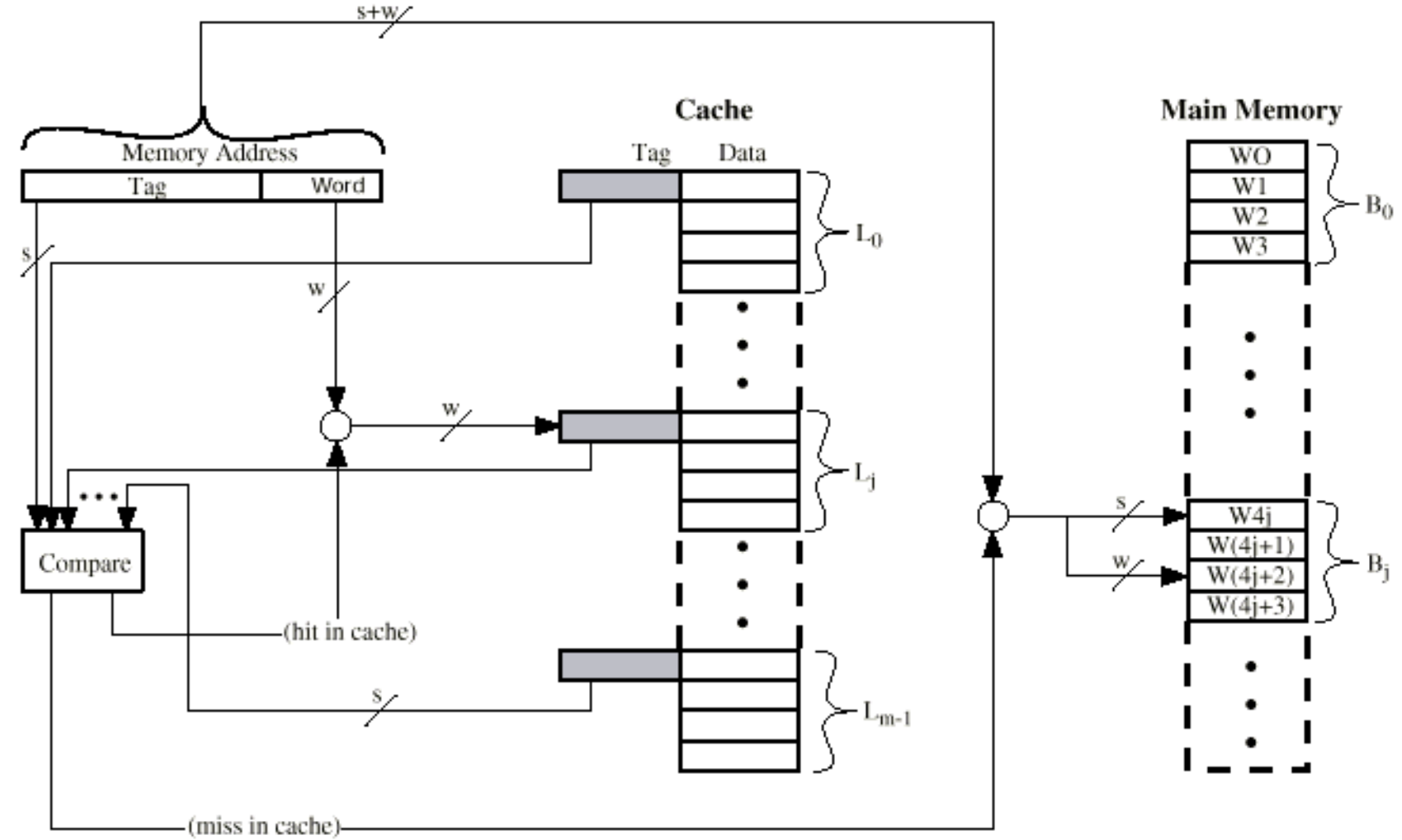
Direct Mapping pros & cons

- Simple
- Inexpensive
- Fixed location for given block
  - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

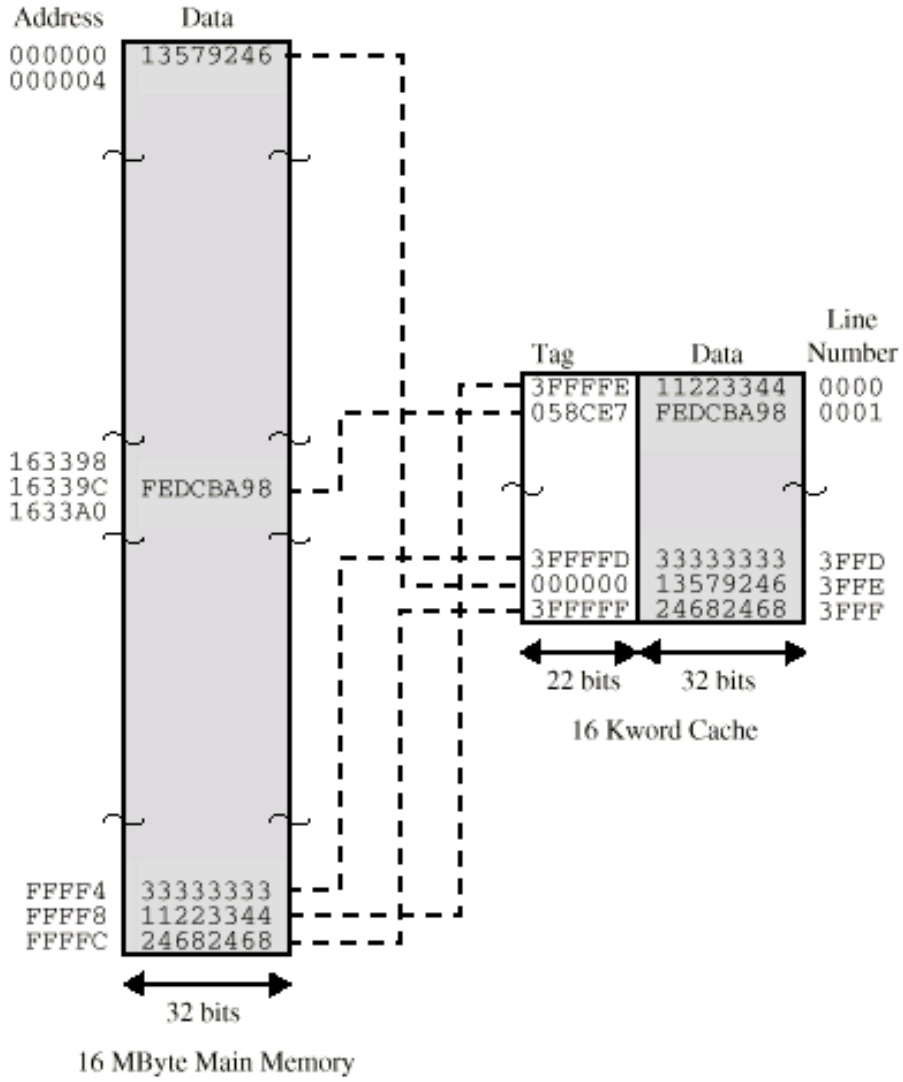
Associative Mapping

- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line's tag is examined for a match
- Cache searching gets expensive

Fully Associative Cache Organization



Associative Mapping Example



Associative Mapping Address Structure

- 22 bit tag stored with each 32 bit block of data
- Compare tag field with tag entry in cache to check for hit
- Least significant 2 bits of address identify which 16 bit word is required from 32 bit data block
- e.g.

Address	Tag	Data	Cache line
FFFFFC	FFFFFC	24682468	3FFF

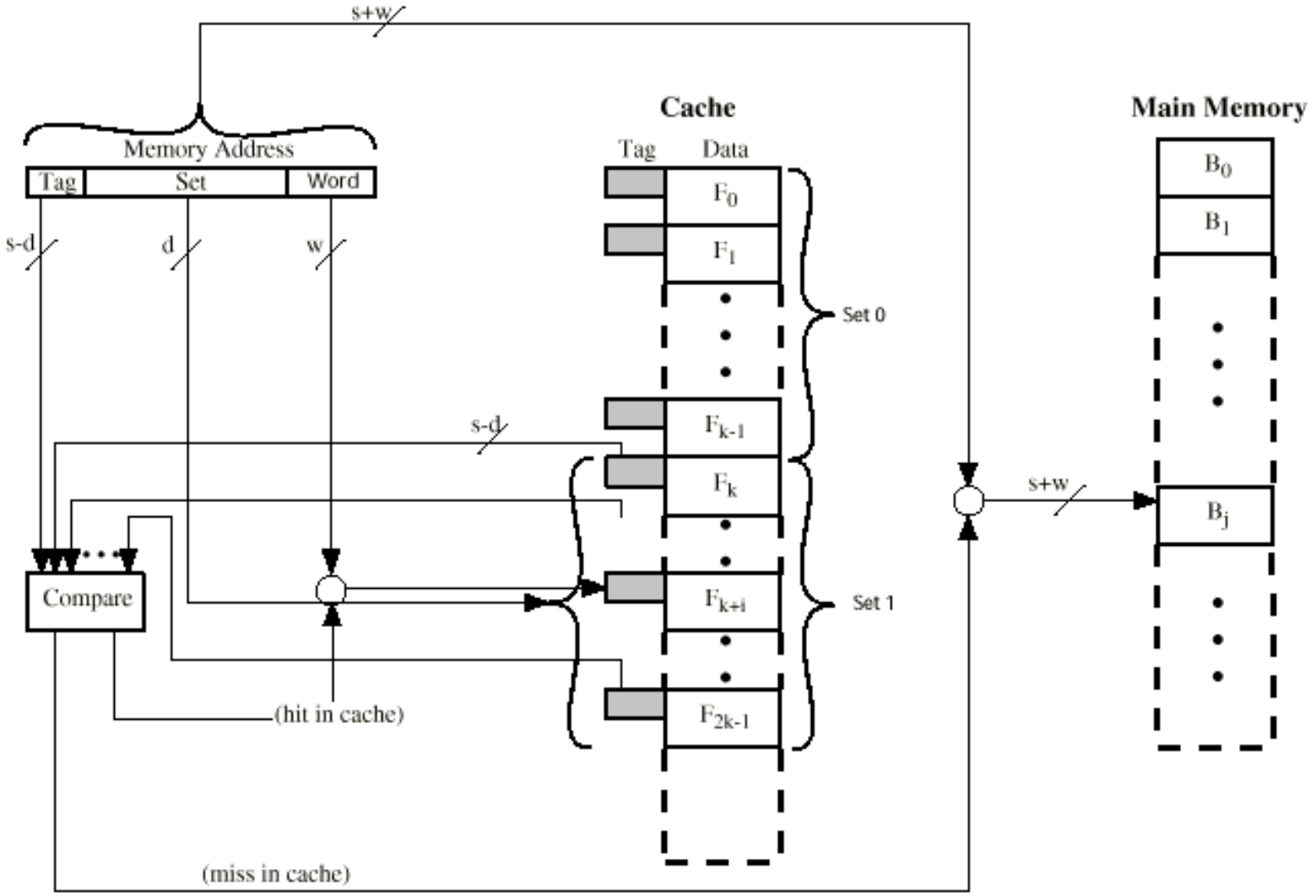
Set Associative Mapping

- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - e.g. Block B can be in any line of set i
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

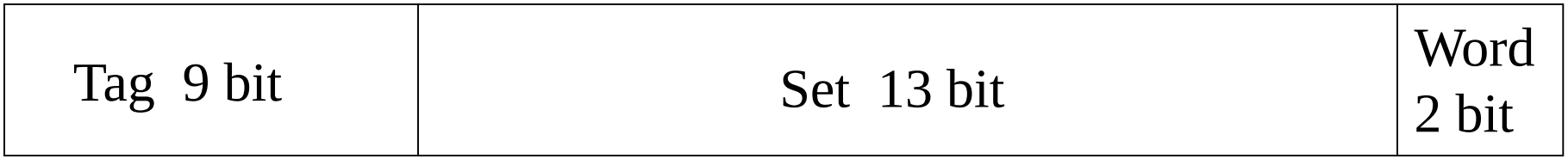
Set Associative Mapping Example

- 13 bit set number
- Block number in main memory is modulo 2^{13}
- 000000, 00A000, 00B000, 00C000 … map to same set

Two Way Set Associative Cache Organization

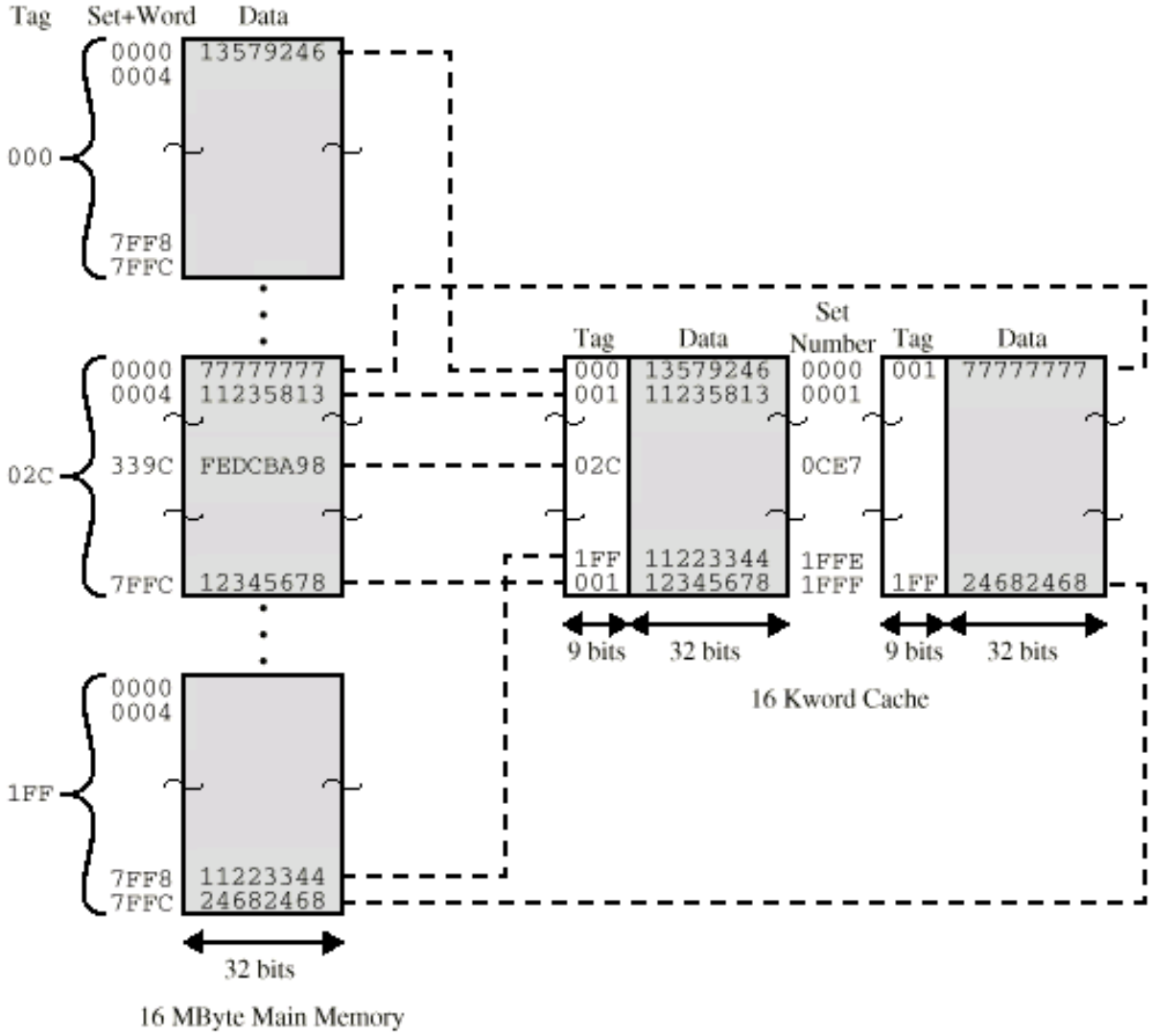


Set Associative Mapping Address Structure



- Use set field to determine cache set to look in
- Compare tag field to see if we have a hit
- e.g
  - Address Tag Data Set number
  - 1FF 7FFC 1FF 12345678 1FFF
  - 001 7FFC 001 11223344 1FFF

Two Way Set Associative Mapping Example



Replacement Algorithms (1) Direct mapping

- No choice
- Each block only maps to one line
- Replace that line

Replacement Algorithms (2) Associative & Set Associative

- Hardware implemented algorithm (speed)
- Least Recently used (LRU)
- e.g. in 2 way set associative
  - Which of the 2 block is lru?
- First in first out (FIFO)
  - replace block that has been in cache longest
- Least frequently used
  - replace block which has had fewest hits
- Random

Write Policy

- Must not overwrite a cache block unless main memory is up to date
- Multiple CPUs may have individual caches
- I/O may address main memory directly

Write through

- All writes go to main memory as well as cache
- Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date
- Lots of traffic
- Slows down writes

- Remember bogus write through caches!

Write back

- Updates initially made in cache only
- Update bit for cache slot is set when update occurs
- If block is to be replaced, write to main memory only if update bit is set
- Other caches get out of sync
- I/O must access main memory through cache
- N.B. 15% of memory references are writes

Pentium Cache

- Foreground reading
- Find out detail of Pentium II cache systems
- NOT just from Stallings!

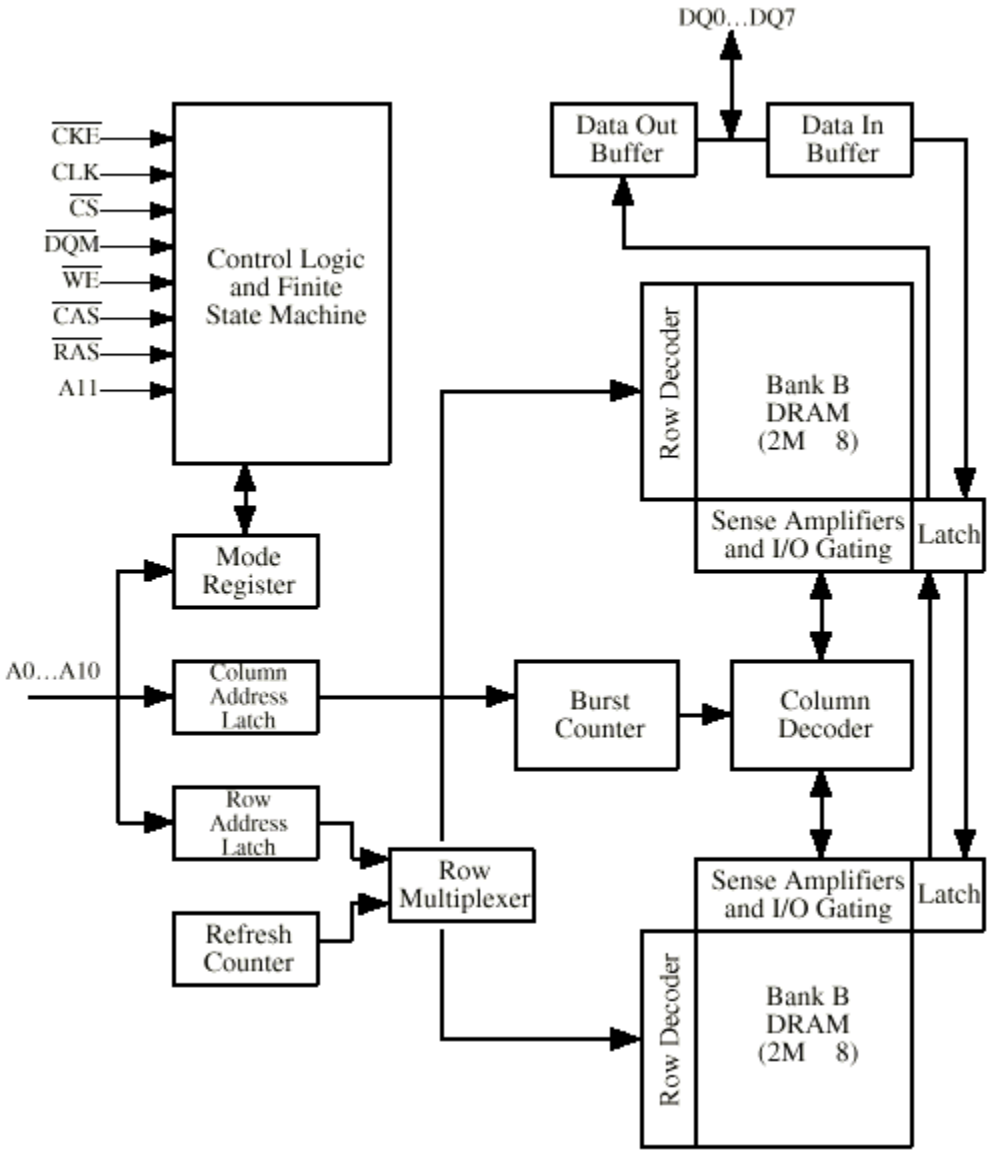
Newer RAM Technology (1)

- Basic DRAM same since first RAM chips
- Enhanced DRAM
  - Contains small SRAM as well
  - SRAM holds last line read (c.f. Cache!)
- Cache DRAM
  - Larger SRAM component
  - Use as cache or serial buffer

Newer RAM Technology (2)

- Synchronous DRAM (SDRAM)
  - currently on DIMMs
  - Access is synchronized with an external clock
  - Address is presented to RAM
  - RAM finds data (CPU waits in conventional DRAM)
  - Since SDRAM moves data in time with system clock, CPU knows when data will be ready
  - CPU does not have to wait, it can do something else
  - Burst mode allows SDRAM to set up stream of data and fire it out in block

SDRAM


DQ0…DQ7
Data Out Buffer
Data In Buffer
CKE
CLK
CS
DQM
WE
CAS
RAS
A11
Control Logic and Finite State Machine
Row Decoder
Bank B DRAM (2M 8)
Sense Amplifiers and I/O Gating
Latch
Mode Register
A0…A10
Column Address Latch
Burst Counter
Column Decoder
Row Address Latch
Row Multiplexer
Refresh Counter
Sense Amplifiers and I/O Gating
Latch
Row Decoder
Bank B DRAM (2M 8)

Newer RAM Technology (3)

- Foreground reading
- Check out any other RAM you can find
- See Web site:
  - The RAM Guide

William Stallings Computer Organization and Architecture

Chapter 5

External Memory

Types of External Memory

- Magnetic Disk
  - RAID
  - Removable
- Optical
  - CD-ROM
  - CD-Writable (WORM)
  - CD-R/W
  - DVD
- Magnetic Tape

Magnetic Disk

- Metal or plastic disk coated with magnetizable material (iron oxide…rust)
- Range of packaging
  - Floppy
  - Winchester hard disk
  - Removable hard disk

Data Organization and Formatting

- Concentric rings or tracks
  - Gaps between tracks
  - Reduce gap to increase capacity
  - Same number of bits per track (variable packing density)
  - Constant angular velocity
- Tracks divided into sectors
- Minimum block size is one sector
- May have more than one sector per block

Disk Data Layout

Fixed/Movable Head Disk

- Fixed head
  - One read write head per track
  - Heads mounted on fixed ridged arm
- Movable head
  - One read write head per side
  - Mounted on a movable arm

Fixed and Movable Heads

Removable or Not

- Removable disk
  - Can be removed from drive and replaced with another disk
  - Provides unlimited storage capacity
  - Easy data transfer between systems
- Nonremovable disk
  - Permanently mounted in the drive

Floppy Disk

- 8″, 5.25″, 3.5″
- Small capacity
  - Up to 1.44Mbyte (2.88M never popular)
- Slow
- Universal
- Cheap

Winchester Hard Disk (1)

- Developed by IBM in Winchester (USA)
- Sealed unit
- One or more platters (disks)
- Heads fly on boundary layer of air as disk spins
- Very small head to disk gap
- Getting more robust

Winchester Hard Disk (2)

- Universal
- Cheap
- Fastest external storage
- Getting larger all the time
  - Multiple Gigabyte now usual

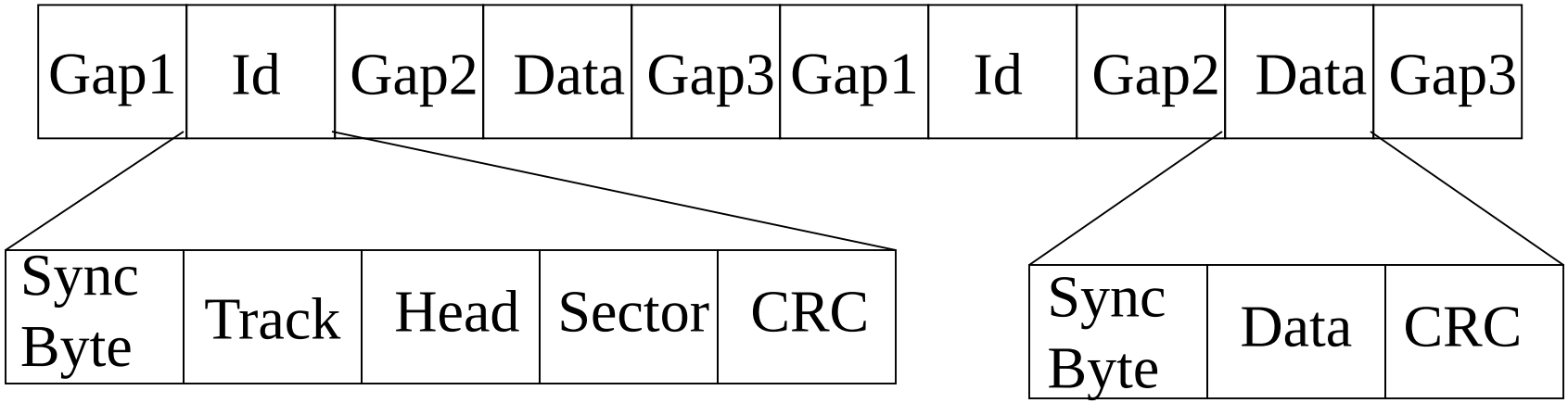
Removable Hard Disk

- ZIP
  - Cheap
  - Very common
  - Only 100M
- JAZ
  - Not cheap
  - 1G
- L-120 (a: drive)
  - Also reads 3.5” floppy
  - Becoming more popular?

Finding Sectors

- Must be able to identify start of track and sector
- Format disk
  - Additional information not available to user
  - Marks tracks and sectors

ST506 format (old!)



- Foreground reading
  - Find others

Characteristics

- Fixed (rare) or movable head
- Removable or fixed
- Single or double (usually) sided
- Single or multiple platter
- Head mechanism
  - Contact (Floppy)
  - Fixed gap
  - Flying (Winchester)

Multiple Platter

- One head per side
- Heads are joined and aligned
- Aligned tracks on each platter form cylinders
- Data is striped by cylinder
  - reduces head movement
  - Increases speed (transfer rate)

Speed

- Seek time
  - Moving head to correct track
- (Rotational) latency
  - Waiting for data to rotate under head
- Access time = Seek + Latency
- Transfer rate

RAID

- Redundant Array of Independent Disks
- Redundant Array of Inexpensive Disks
- 6 levels in common use
- Not a hierarchy
- Set of physical disks viewed as single logical drive by O/S
- Data distributed across physical drives
- Can use redundant capacity to store parity information

RAID 0

- No redundancy
- Data striped across all disks
- Round Robin striping
- Increase speed
  - Multiple data requests probably not on same disk
  - Disks seek in parallel
  - A set of data is likely to be striped across multiple disks

RAID 1

- Mirrored Disks
- Data is striped across disks
- 2 copies of each stripe on separate disks
- Read from either
- Write to both
- Recovery is simple
  - Swap faulty disk & re-mirror
  - No down time
- Expensive

RAID 2

- Disks are synchronized
- Very small stripes
  - Often single byte/word
- Error correction calculated across corresponding bits on disks
- Multiple parity disks store Hamming code error correction in corresponding positions
- Lots of redundancy
  - Expensive
  - Not used

RAID 3

- Similar to RAID 2
- Only one redundant disk, no matter how large the array
- Simple parity bit for each set of corresponding bits
- Data on failed drive can be reconstructed from surviving data and parity info
- Very high transfer rates

RAID 4

- Each disk operates independently
- Good for high I/O request rate
- Large stripes
- Bit by bit parity calculated across stripes on each disk
- Parity stored on parity disk

RAID 5

- Like RAID 4
- Parity striped across all disks
- Round robin allocation for parity stripe
- Avoids RAID 4 bottleneck at parity disk
- Commonly used in network servers

- N.B. DOES NOT MEAN 5 DISKS!!!!!

Optical Storage CD-ROM

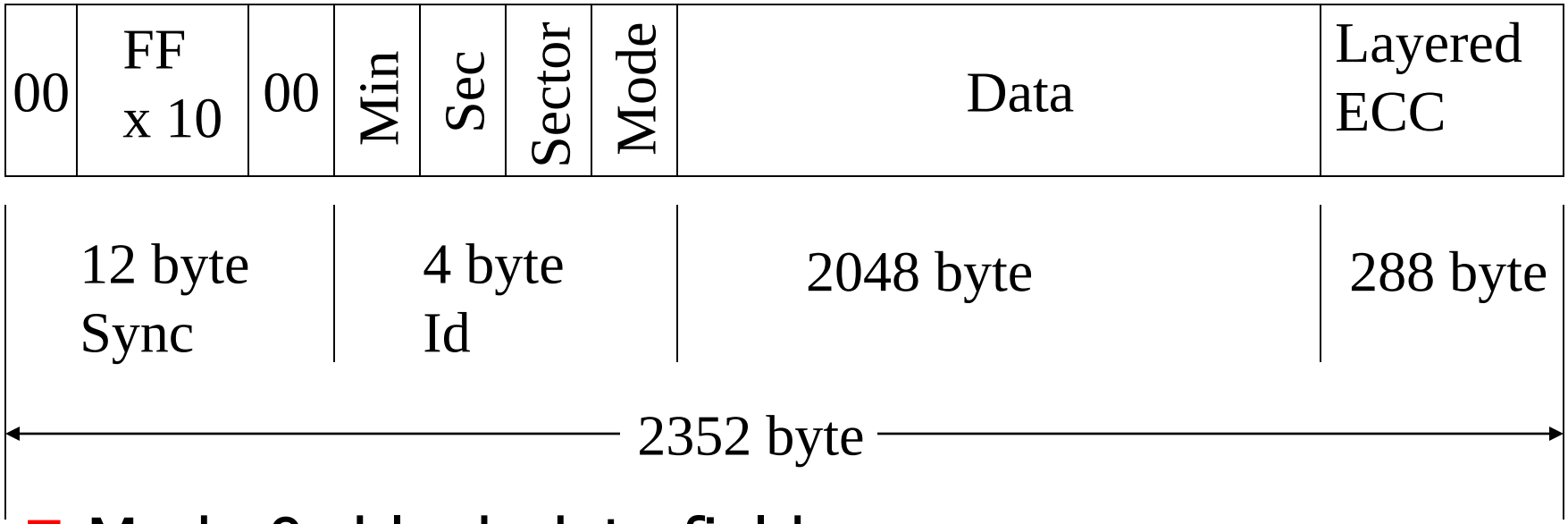
- Originally for audio
- 650Mbytes giving over 70 minutes audio
- Polycarbonate coated with highly reflective coat, usually aluminum
- Data stored as pits
- Read by reflecting laser
- Constant packing density
- Constant linear velocity

CD-ROM Drive Speeds

- Audio is single speed
  - Constant linier velocity
  - 1.2 ms^{-1}
  - Track (spiral) is 5.27km long
  - Gives 4391 seconds = 73.2 minutes
- Other speeds are quoted as multiples
- e.g. 24x
- The quoted figure is the maximum the drive can achieve

Comparison of Disk Layouts

CD-ROM Format



- Mode 0=blank data field
- Mode 1=2048 byte data+error correction
- Mode 2=2336 byte data

Random Access on CD-ROM

- Difficult
- Move head to rough position
- Set correct speed
- Read address
- Adjust to required location
- (Yawn!)

CD-ROM for & against

- Large capacity (?)
- Easy to mass produce
- Removable
- Robust

- Expensive for small runs
- Slow
- Read only

Other Optical Storage

- CD-Writable
  - WORM
  - Now affordable
  - Compatible with CD-ROM drives
- CD-RW
  - Erasable
  - Getting cheaper
  - Mostly CD-ROM drive compatible

DVD - what's in a name?

- Digital Video Disk
  - Used to indicate a player for movies
    - Only plays video disks
- Digital Versatile Disk
  - Used to indicate a computer drive
    - Will read computer disks and play video disks
- Dogs Veritable Dinner
- Officially - nothing!!!

DVD - technology

- Multi-layer
- Very high capacity (4.7G per layer)
- Full length movie on single disk
  - Using MPEG compression
- Finally standardized (honest!)
- Movies carry regional coding
- Players only play correct region films
- Can be “fixed”

DVD - Writable

- Loads of trouble with standards
- First generation DVD drives may not read first generation DVD-W disks
- First generation DVD drives may not read CD-RW disks
- Wait for it to settle down before buying!

Foreground Reading

- Check out optical disk storage options
- Check out Mini Disk

Magnetic Tape

- Serial access
- Slow
- Very cheap
- Backup and archive

Digital Audio Tape (DAT)

- Uses rotating head (like video)
- High capacity on small tape
  - 4Gbyte uncompressed
  - 8Gbyte compressed
- Backup of PC/network servers

William Stallings
Computer Organization and Architecture

Chapter 6
Input/Output

Input/Output Problems

- Wide variety of peripherals
  - Delivering different amounts of data
  - At different speeds
  - In different formats
- All slower than CPU and RAM
- Need I/O modules

Input/Output Module

- Interface to CPU and Memory
- Interface to one or more peripherals
- GENERIC MODEL OF I/O DIAGRAM 6.1

External Devices

- Human readable
  - Screen, printer, keyboard
- Machine readable
  - Monitoring and control
- Communication
  - Modem
  - Network Interface Card (NIC)

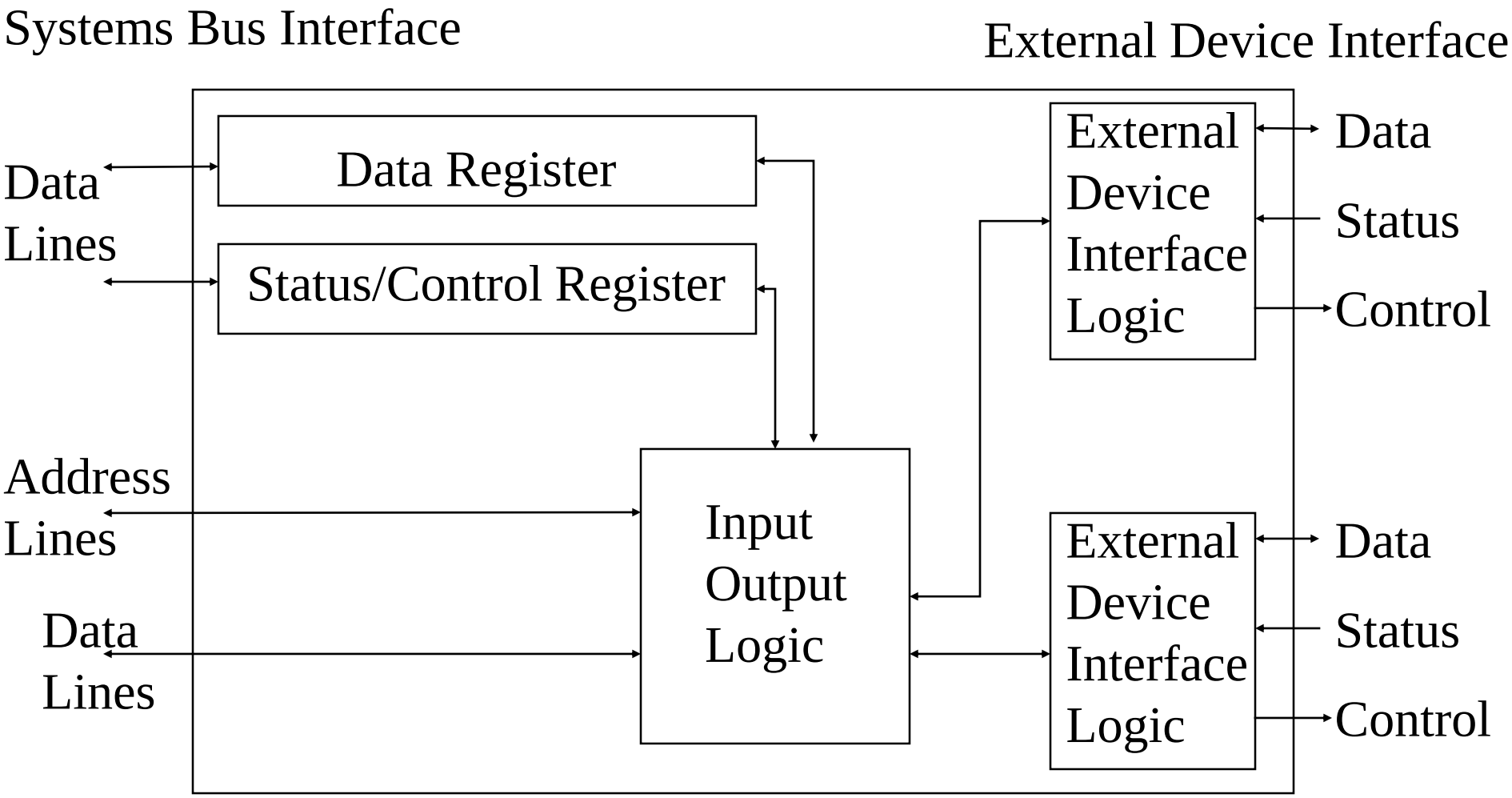
I/O Module Function

- Control & Timing
- CPU Communication
- Device Communication
- Data Buffering
- Error Detection

I/O Steps

- CPU checks I/O module device status
- I/O module returns status
- If ready, CPU requests data transfer
- I/O module gets data from device
- I/O module transfers data to CPU
- Variations for output, DMA, etc.

I/O Module Diagram



I/O Module Decisions

- Hide or reveal device properties to CPU
- Support multiple or single device
- Control device functions or leave for CPU
- Also O/S decisions
  - e.g. Unix treats everything it can as a file

Input Output Techniques

- Programmed
- Interrupt driven
- Direct Memory Access (DMA)

Programmed I/O

- CPU has direct control over I/O
  - Sensing status
  - Read/write commands
  - Transferring data
- CPU waits for I/O module to complete operation
- Wastes CPU time

Programmed I/O - detail

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically
- I/O module does not inform CPU directly
- I/O module does not interrupt CPU
- CPU may wait or come back later

I/O Commands

- CPU issues address
  - Identifies module (& device if >1 per module)
- CPU issues command
  - Control - telling module what to do
    - e.g. spin up disk
  - Test - check status
    - e.g. power? Error?
  - Read/Write
    - Module transfers data via buffer from/to device

Addressing I/O Devices

- Under programmed I/O data transfer is very like memory access (CPU viewpoint)
- Each device given unique identifier
- CPU commands contain identifier (address)

I/O Mapping

- Memory mapped I/O
  - Devices and memory share an address space
  - I/O looks just like memory read/write
  - No special commands for I/O
    - Large selection of memory access commands available
- Isolated I/O
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O
    - Limited set

Interrupt Driven I/O

- Overcomes CPU waiting
- No repeated CPU checking of device
- I/O module interrupts when ready

Interrupt Driven I/O
Basic Operation

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

CPU Viewpoint

- Issue read command
- Do other work
- Check for interrupt at end of each instruction cycle
- If interrupted:-
  - Save context (registers)
  - Process interrupt
    - Fetch data & store
- See Operating Systems notes

Design Issues

- How do you identify the module issuing the interrupt?
- How do you deal with multiple interrupts?
  - i.e. an interrupt handler being interrupted

Identifying Interrupting Module (1)

- Different line for each module
  - PC
  - Limits number of devices
- Software poll
  - CPU asks each module in turn
  - Slow

Identifying Interrupting Module (2)

- Daisy Chain or Hardware poll
  - Interrupt Acknowledge sent down a chain
  - Module responsible places vector on bus
  - CPU uses vector to identify handler routine
- Bus Master
  - Module must claim the bus before it can raise interrupt
  - e.g. PCI & SCSI

Multiple Interrupts

- Each interrupt line has a priority
- Higher priority lines can interrupt lower priority lines
- If bus mastering only current master can interrupt

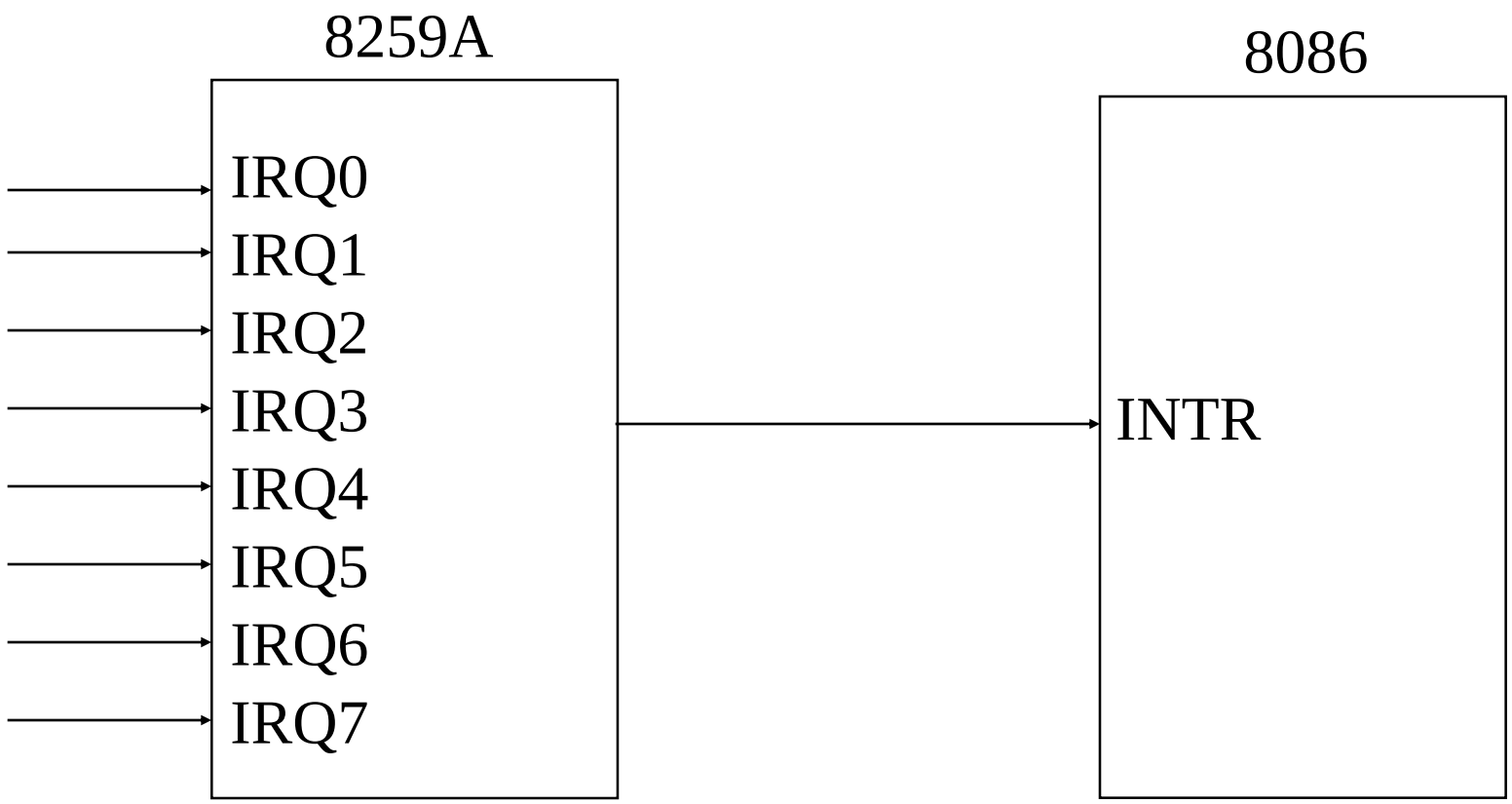
Example - PC Bus

- 80x86 has one interrupt line
- 8086 based systems use one 8259A interrupt controller
- 8259A has 8 interrupt lines

Sequence of Events

- 8259A accepts interrupts
- 8259A determines priority
- 8259A signals 8086 (raises INTR line)
- CPU Acknowledges
- 8259A puts correct vector on data bus
- CPU processes interrupt

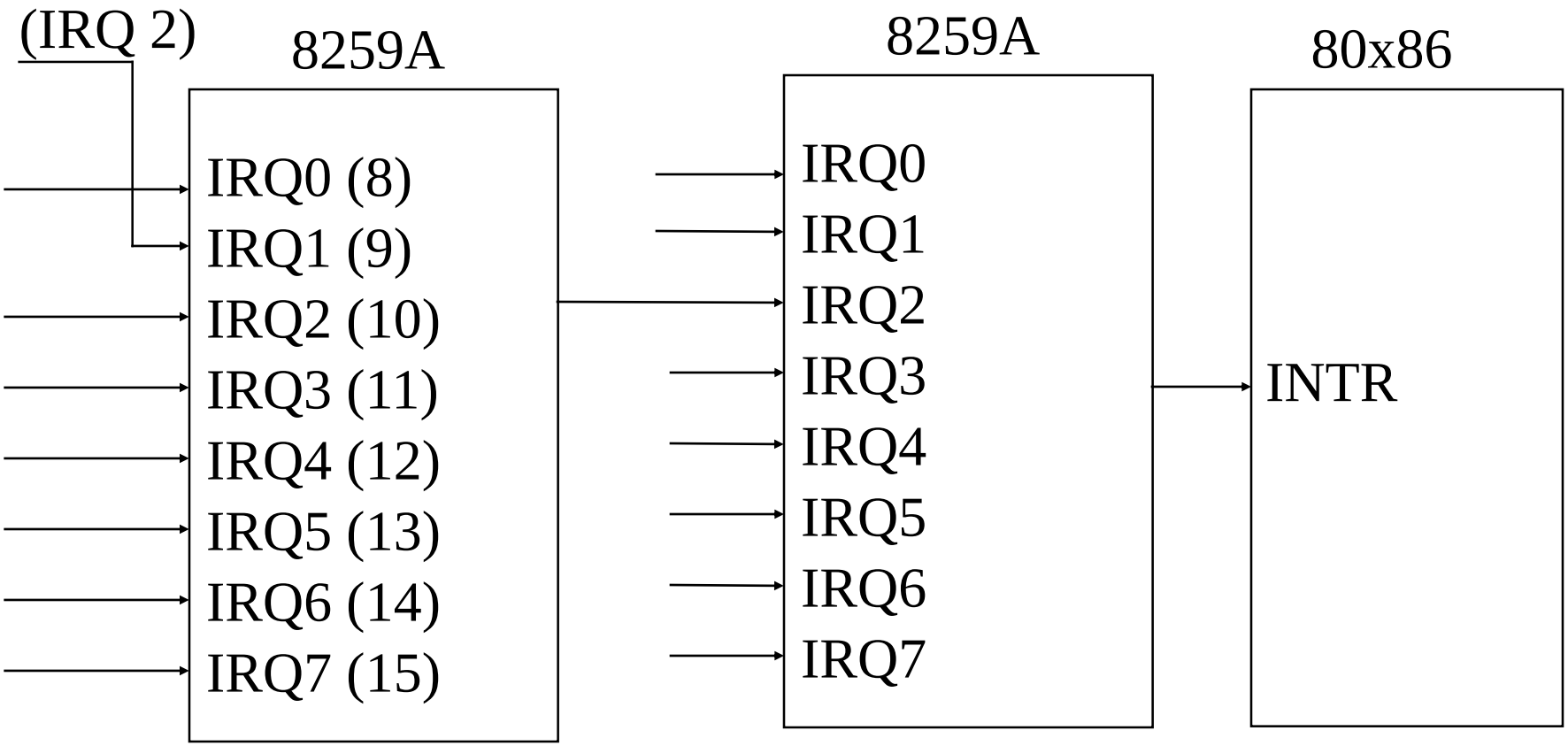
PC Interrupt Layout



ISA Bus Interrupt System

- ISA bus chains two 8259As together
- Link is via interrupt 2
- Gives 15 lines
  - 16 lines less one for link
- IRQ 9 is used to re-route anything trying to use IRQ 2
  - Backwards compatibility
- Incorporated in chip set

ISA Interrupt Layout



Foreground Reading

- http://www.pcguide.com/ref/mbsys/res/irq/func.htm
- In fact look at http://www.pcguide.com/

Direct Memory Access

- Interrupt driven and programmed I/O require active CPU intervention
  - Transfer rate is limited
  - CPU is tied up
- DMA is the answer

DMA Function

- Additional Module (hardware) on bus
- DMA controller takes over from CPU for I/O

DMA Operation

- CPU tells DMA controller:-
  - Read/Write
  - Device address
  - Starting address of memory block for data
  - Amount of data to be transferred
- CPU carries on with other work
- DMA controller deals with transfer
- DMA controller sends interrupt when finished

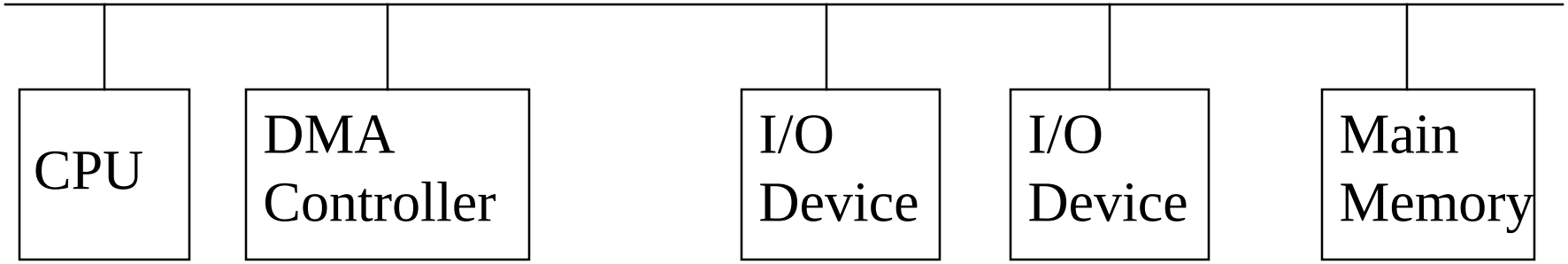
DMA Transfer
Cycle Stealing

- DMA controller takes over bus for a cycle
- Transfer of one word of data
- Not an interrupt
  - CPU does not switch context
- CPU suspended just before it accesses bus
  - i.e. before an operand or data fetch or a data write
- Slows down CPU but not as much as CPU doing transfer

Aside

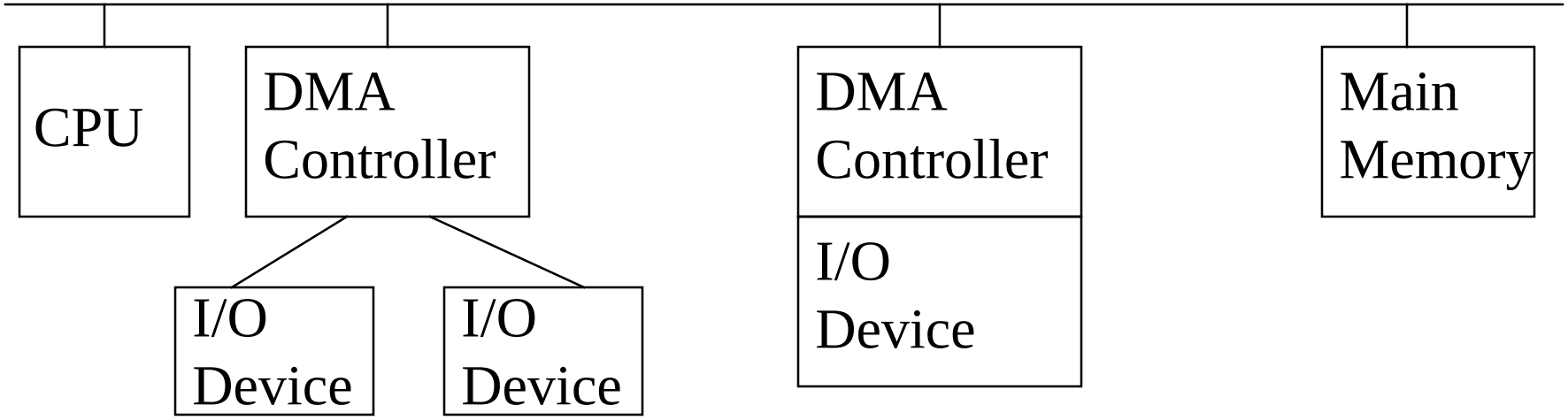
- What effect does caching memory have on DMA?
- Hint: how much are the system buses available?

DMA Configurations (1)



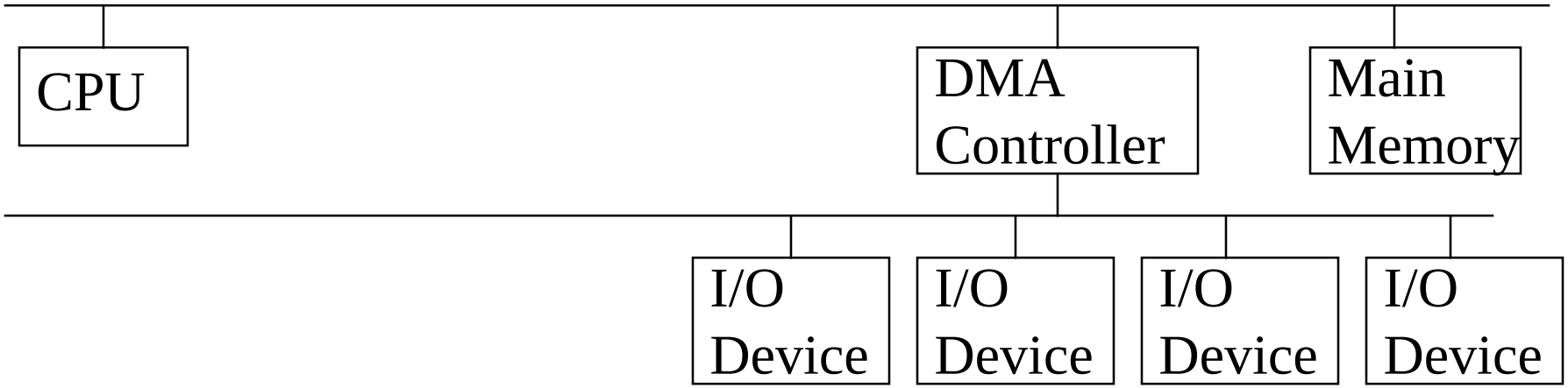
- Single Bus, Detached DMA controller
- Each transfer uses bus twice
  - I/O to DMA then DMA to memory
- CPU is suspended twice

DMA Configurations (2)



- Single Bus, Integrated DMA controller
- Controller may support >1 device
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

DMA Configurations (3)



- Separate I/O Bus
- Bus supports all DMA enabled devices
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

I/O Channels

- I/O devices getting more sophisticated
- e.g. 3D graphics cards
- CPU instructs I/O controller to do transfer
- I/O controller does entire transfer
- Improves speed
  - Takes load off CPU
  - Dedicated processor is faster

Interfacing

- Connecting devices together
- Bit of wire?
- Dedicated processor/memory/buses?
- E.g. SCSI, FireWire

Small Computer Systems Interface (SCSI)

- Parallel interface
- 8, 16, 32 bit data lines
- Daisy chained
- Devices are independent
- Devices can communicate with each other as well as host

SCSI - 1

- Early 1980s
- 8 bit
- 5MHz
- Data rate 5MBytes.s^{-1}
- Seven devices
  - Eight including host interface

SCSI - 2

- 1991
- 16 and 32 bit
- 10MHz
- Data rate 20 or 40 Mbytes.s^{-1}

- (Check out Ultra/Wide SCSI)

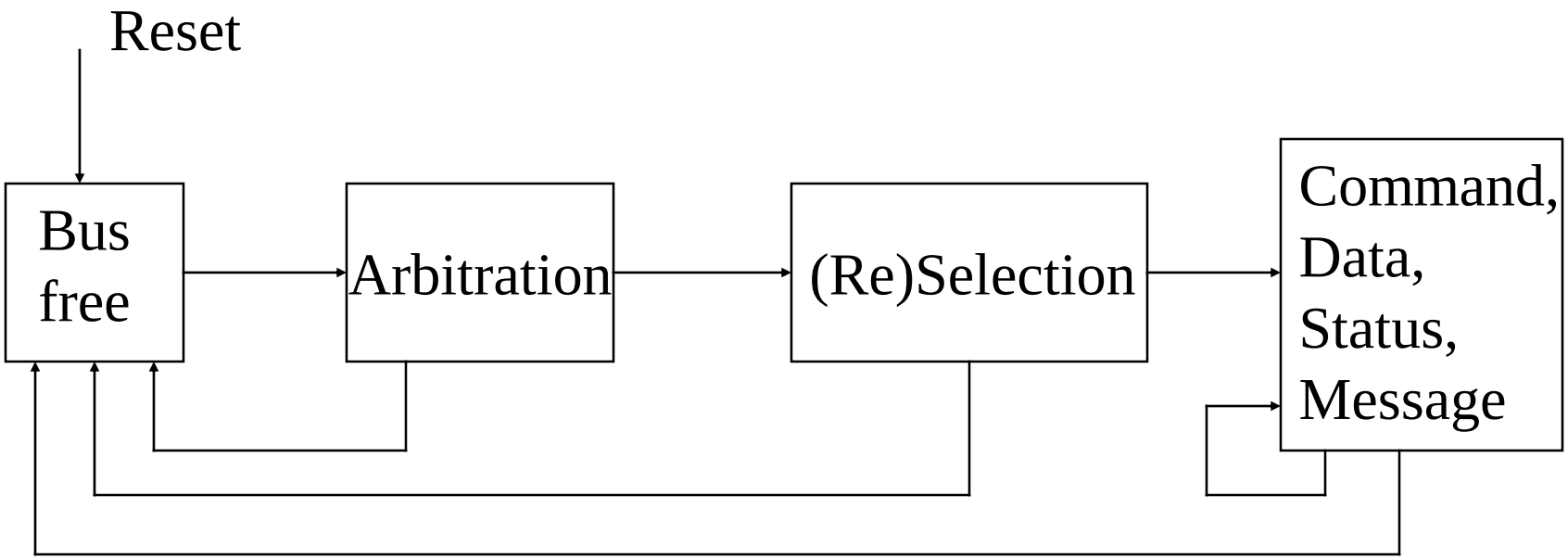
SCSI Signaling (1)

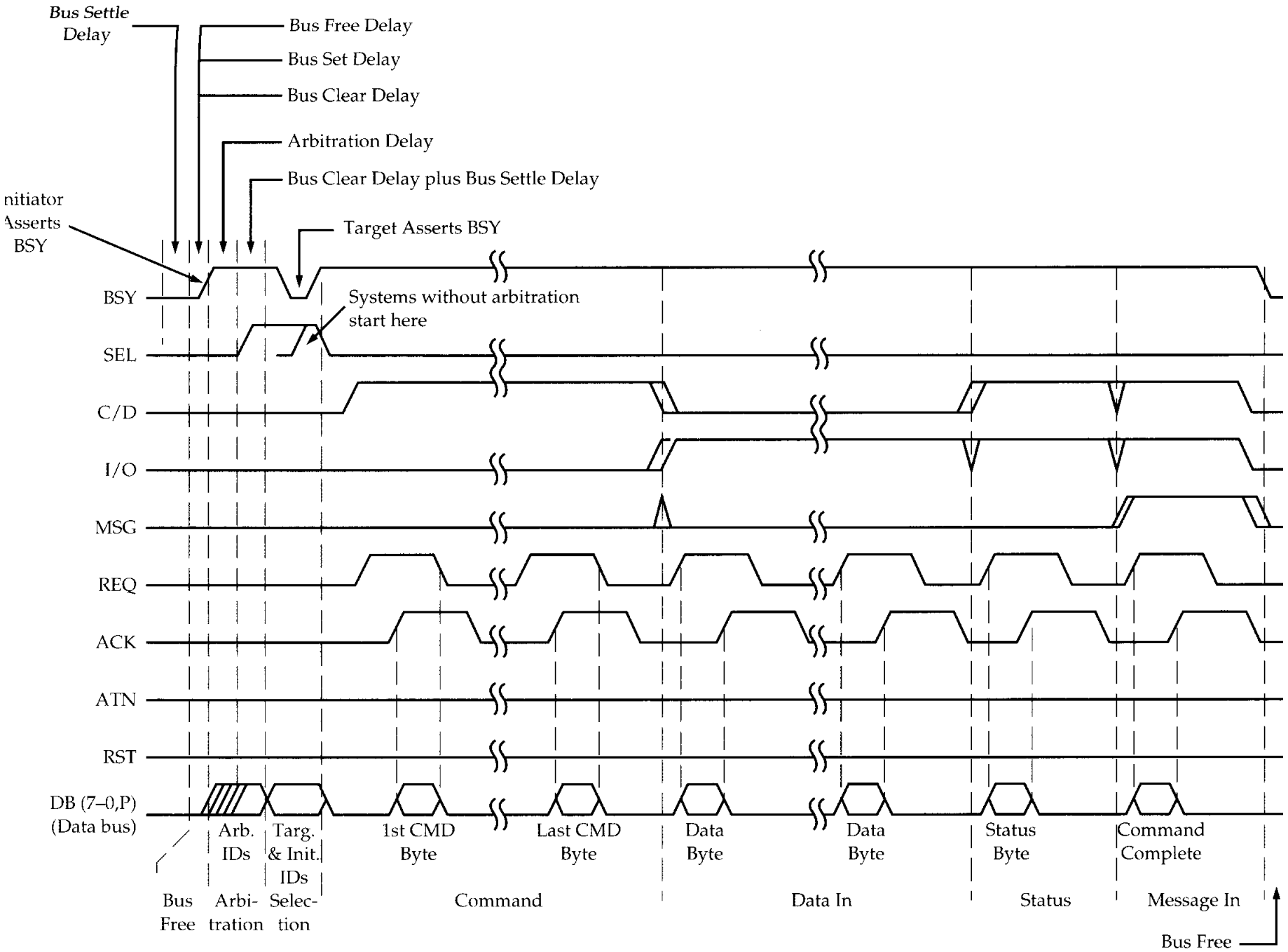
- Between initiator and target
  - Usually host & device
- Bus free? (c.f. Ethernet)
- Arbitration - take control of bus (c.f. PCI)
- Select target
- Reselection
  - Allows reconnection after suspension
  - e.g. if request takes time to execute, bus can be released

SCSI Signaling (2)

- Command - target requesting from initiator
- Data request
- Status request
- Message request (both ways)

SCSI Bus Phases




Bus Settle Delay
Bus Free Delay
Bus Set Delay
Bus Clear Delay
Arbitration Delay
Bus Clear Delay plus Bus Settle Delay
nitiator Asserts BSY
Target Asserts BSY
Systems without arbitration start here
BSY
SEL
C/D
I/O
MSG
REQ
ACK
ATN
RST
DB (7–0,P) (Data bus)
Bus Free
Arb. IDs
Arbi-tration
Targ. & Init. IDs
Selec-tion
1st CMD Byte
Last CMD Byte
Data Byte
Data Byte
Status Byte
Command Complete
Command
Data In
Status
Message In
Bus Free

Configuring SCSI

- Bus must be terminated at each end
  - Usually one end is host adapter
  - Plug in terminator or switch(es)
- SCSI Id must be set
  - Jumpers or switches
  - Unique on chain
  - 0 (zero) for boot device
  - Higher number is higher priority in arbitration

IEEE 1394 FireWire

- High performance serial bus
- Fast
- Low cost
- Easy to implement
- Also being used in digital cameras, VCRs and TV

FireWire Configuration

- Daisy chain
- Up to 63 devices on single port
  - Really 64 of which one is the interface itself
- Up to 1022 buses can be connected with bridges
- Automatic configuration
- No bus terminators
- May be tree structure

FireWire v SCSI

FireWire 3 Layer Stack

- Physical
  - Transmission medium, electrical and signaling characteristics
- Link
  - Transmission of data in packets
- Transaction
  - Request-response protocol

FireWire - Physical Layer

- Data rates from 25 to 400Mbps
- Two forms of arbitration
  - Based on tree structure
  - Root acts as arbiter
  - First come first served
  - Natural priority controls simultaneous requests
    - i.e. who is nearest to root
  - Fair arbitration
  - Urgent arbitration

FireWire - Link Layer

- Two transmission types
  - Asynchronous
    - Variable amount of data and several bytes of transaction data transferred as a packet
    - To explicit address
    - Acknowledgement returned
  - Isochronous
    - Variable amount of data in sequence of fixed size packets at regular intervals
    - Simplified addressing
    - No acknowledgement

FireWire Subactions

Foreground Reading

- Check out Universal Serial Bus (USB)
- Compare with other communication standards e.g. Ethernet

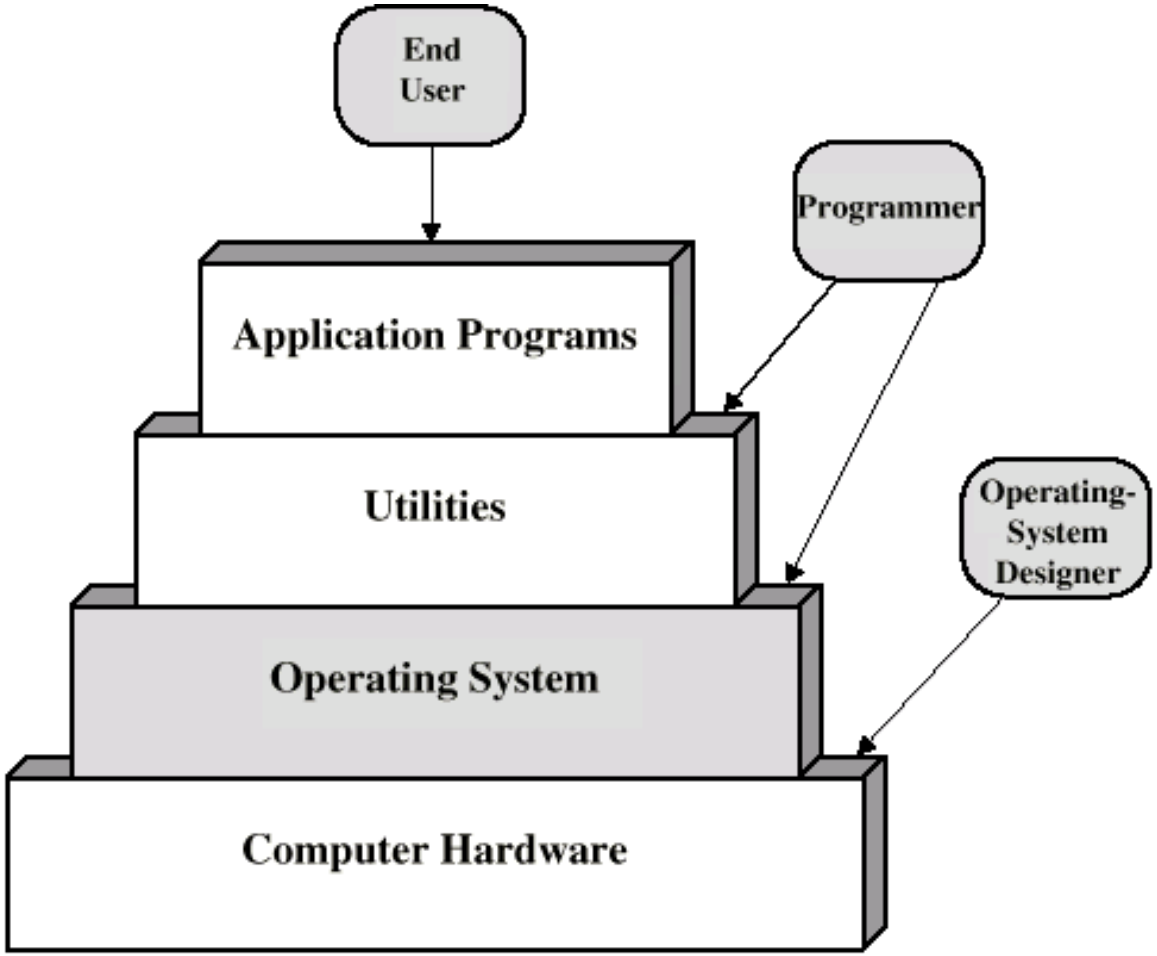
William Stallings Computer Organization and Architecture

Chapter 7
Operating System Support

Objectives and Functions

- Convenience
  - Making the computer easier to use
- Efficiency
  - Allowing better use of computer resources

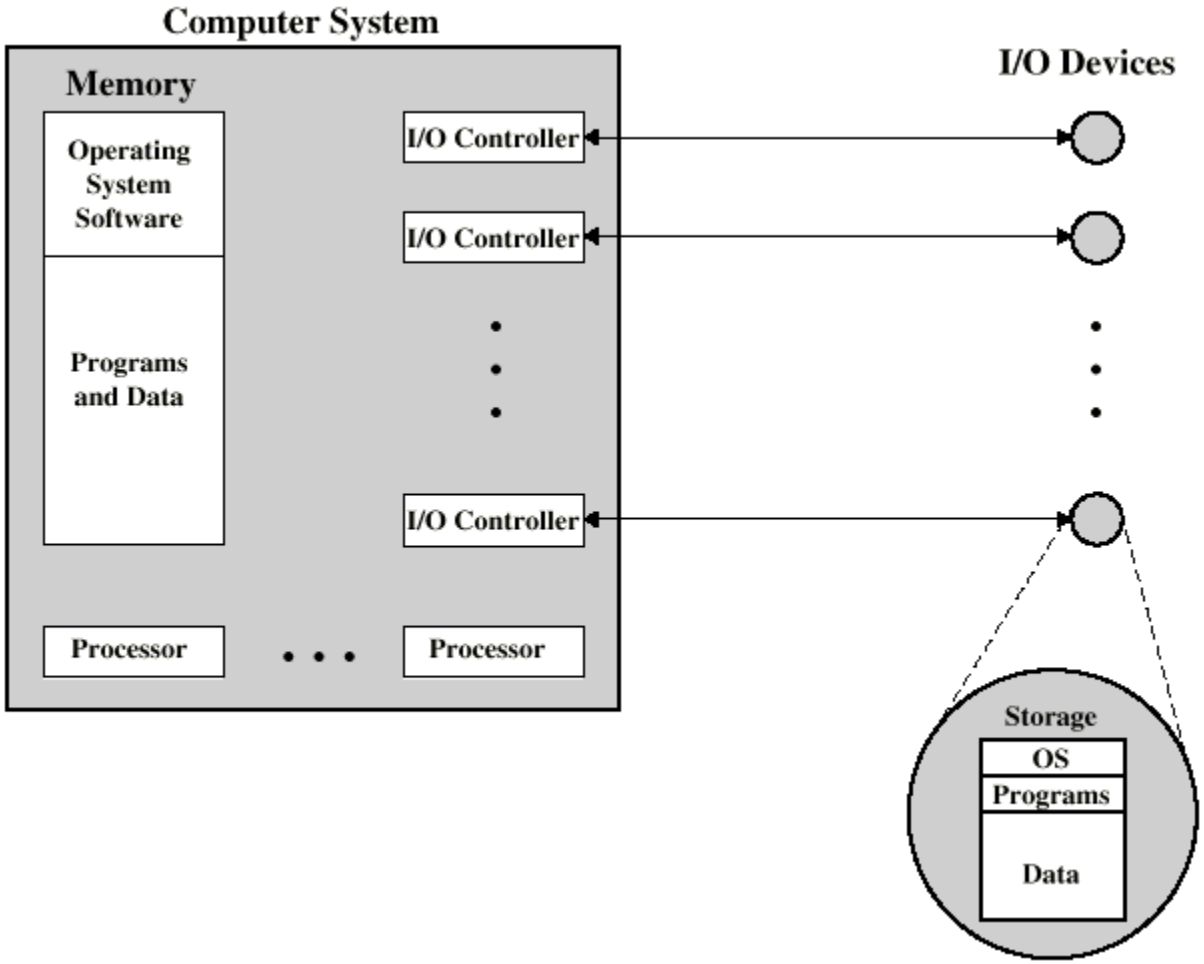
Layers and Views of a Computer System



Operating System Services

- Program creation
- Program execution
- Access to I/O devices
- Controlled access to files
- System access
- Error detection and response
- Accounting

O/S as a Resource Manager



Types of Operating System

- Interactive
- Batch
- Single program (Uni-programming)
- Multi-programming (Multi-tasking)

Early Systems

- Late 1940s to mid 1950s
- No Operating System
- Programs interact directly with hardware
- Two main problems:
  - Scheduling
  - Setup time

Simple Batch Systems

- Resident Monitor program
- Users submit jobs to operator
- Operator batches jobs
- Monitor controls sequence of events to process batch
- When one job is finished, control returns to Monitor which reads next job
- Monitor handles scheduling

Job Control Language

- Instructions to Monitor
- Usually denoted by $
- e.g.
  - $JOB
  - $FTN
  - ... Some Fortran instructions
  - $LOAD
  - $RUN
  - ... Some data
  - $END

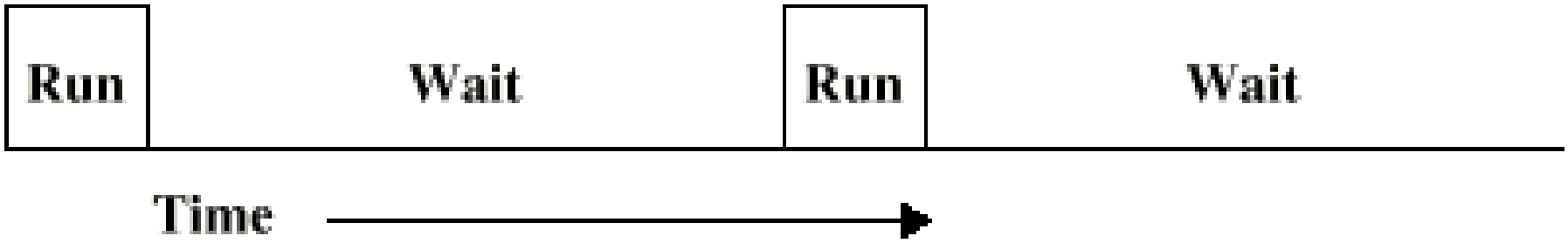
Desirable Hardware Features

- Memory protection
  - To protect the Monitor
- Timer
  - To prevent a job monopolizing the system
- Privileged instructions
  - Only executed by Monitor
  - e.g. I/O
- Interrupts
  - Allows for relinquishing and regaining control

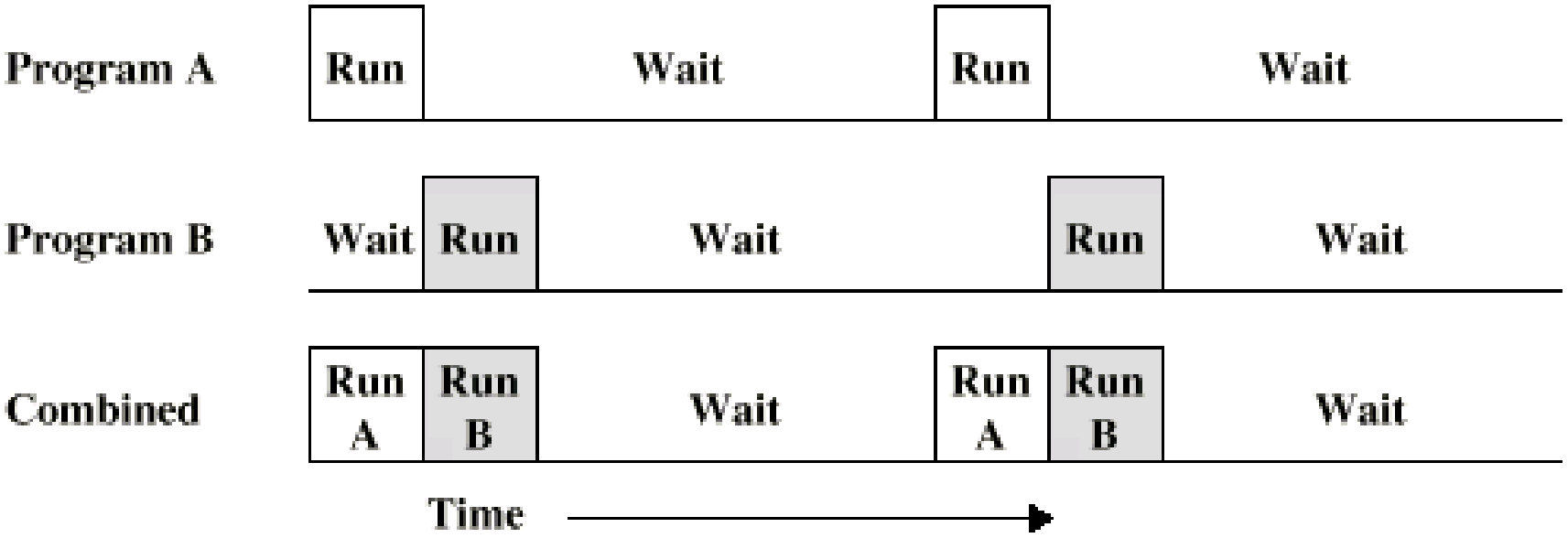
Multi-programmed Batch Systems

- I/O devices very slow
- When one program is waiting for I/O, another can use the CPU

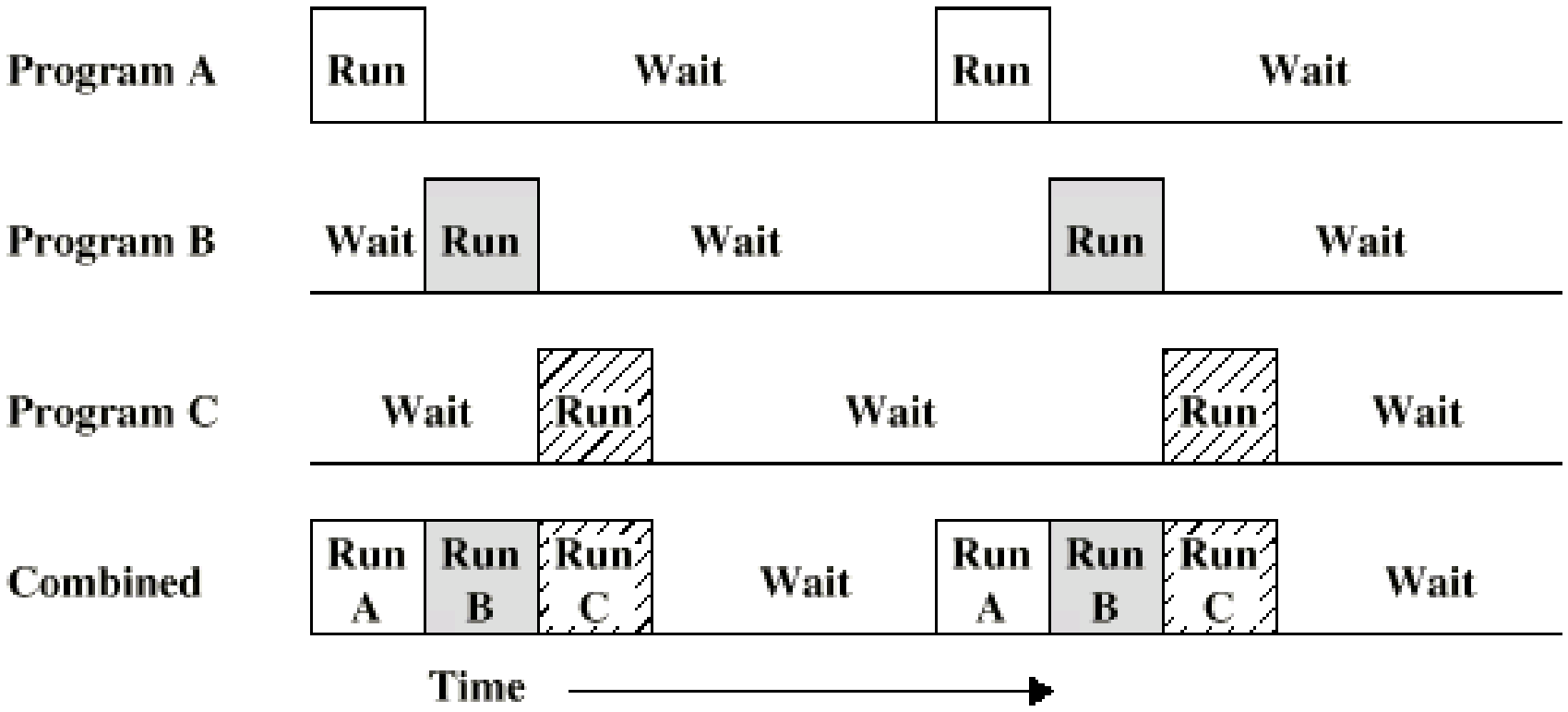
Single Program



Multi-Programming with Two Programs



Multi-Programming with Three Programs



Time Sharing Systems

- Allow users to interact directly with the computer
  - i.e. Interactive
- Multi-programming allows a number of users to interact with the computer

Scheduling

- Key to multi-programming
- Long term
- Medium term
- Short term
- I/O

Long Term Scheduling

- Determines which programs are submitted for processing
- i.e. controls the degree of multi-programming
- Once submitted, a job becomes a process for the short term scheduler
- (or it becomes a swapped out job for the medium term scheduler)

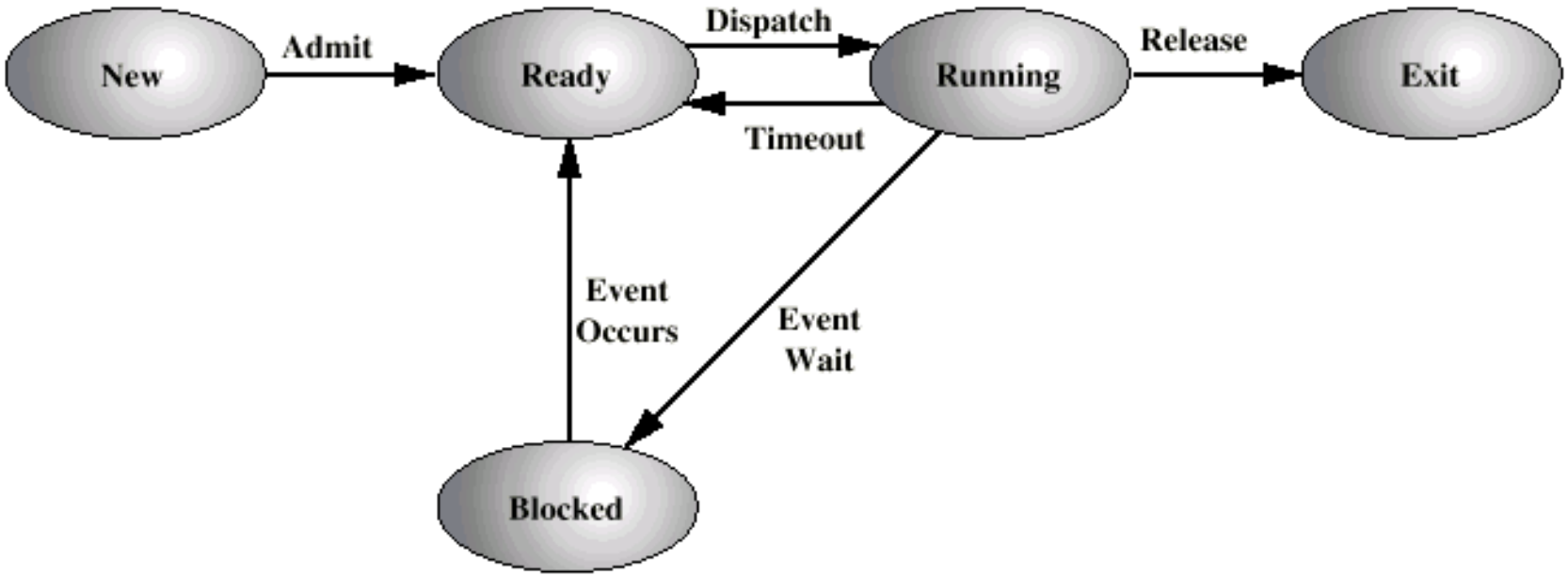
Medium Term Scheduling

- Part of the swapping function (later…)
- Usually based on the need to manage multi-programming
- If no virtual memory, memory management is also an issue

Short Term Scheduler

- Dispatcher
- Fine grained decisions of which job to execute next
- i.e. which job actually gets to use the processor in the next time slot

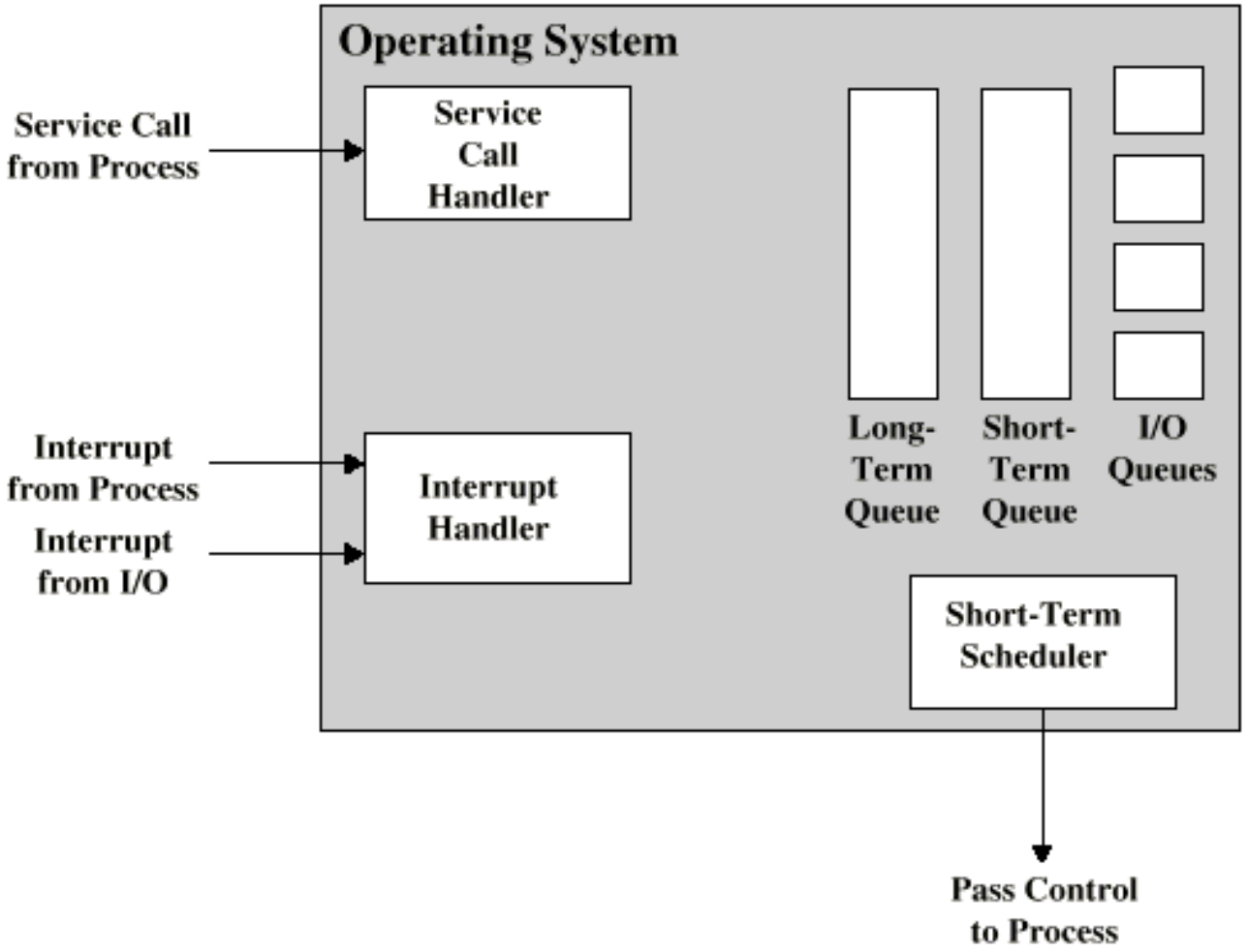
Process States



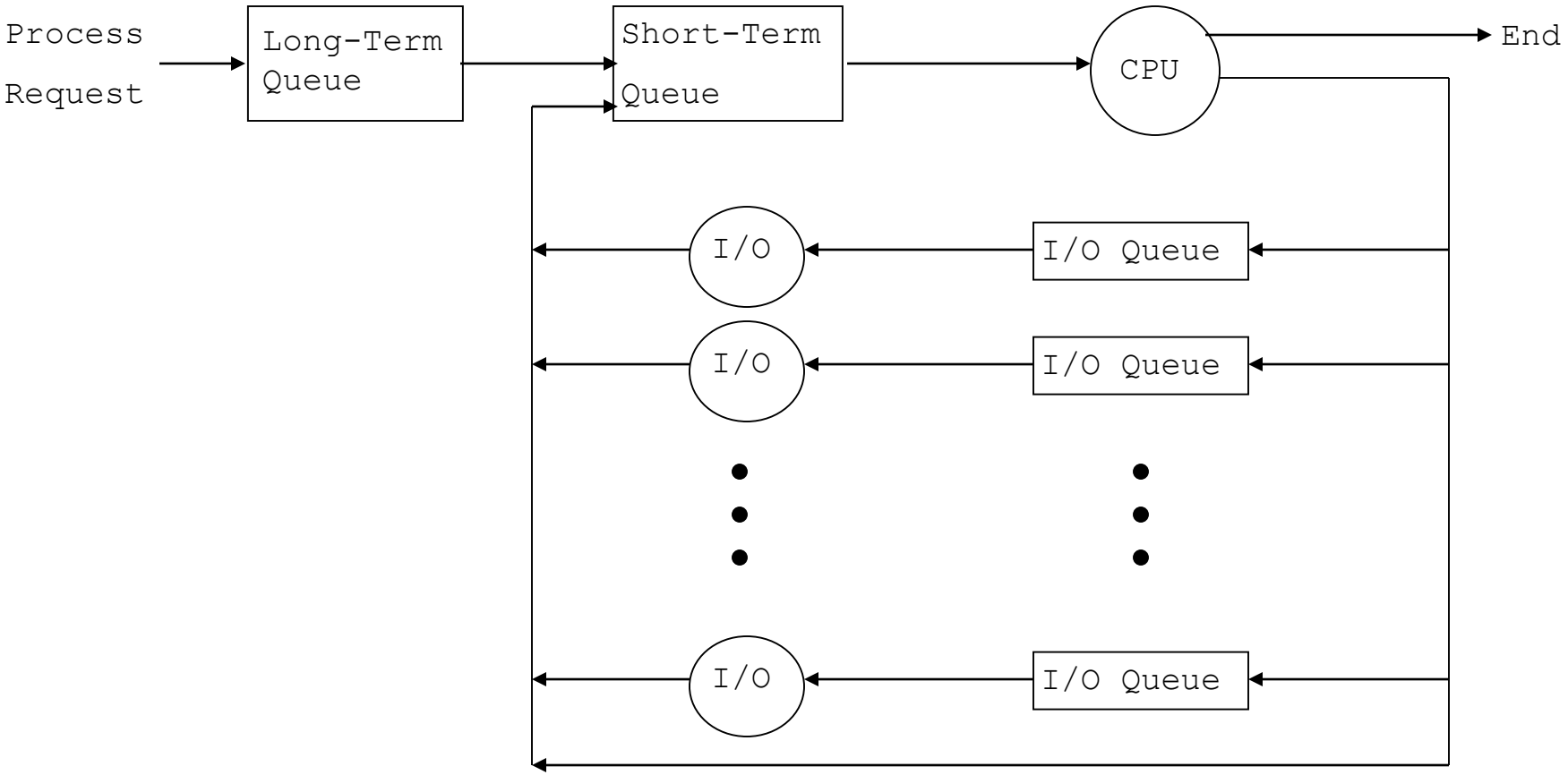
Process Control Block

- Identifier
- State
- Priority
- Program counter
- Memory pointers
- Context data
- I/O status
- Accounting information

Key Elements of O/S



Process Scheduling



Memory Management

- Uni-program
  - Memory split into two
  - One for Operating System (monitor)
  - One for currently executing program
- Multi-program
  - “User” part is sub-divided and shared among active processes

Swapping

- Problem: I/O is so slow compared with CPU that even in multi-programming system, CPU can be idle most of the time
- Solutions:
  - Increase main memory
    - Expensive
    - Leads to larger programs
  - Swapping

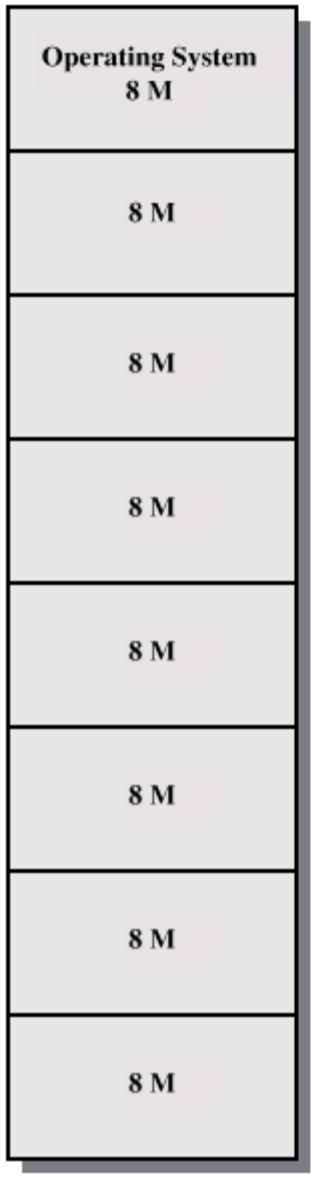
What is Swapping?

- Long term queue of processes stored on disk
- Processes "swapped" in as space becomes available
- As a process completes it is moved out of main memory
- If none of the processes in memory are ready (i.e. all I/O blocked)
  - Swap out a blocked process to intermediate queue
  - Swap in a ready process or a new process
  - But swapping is an I/O process...

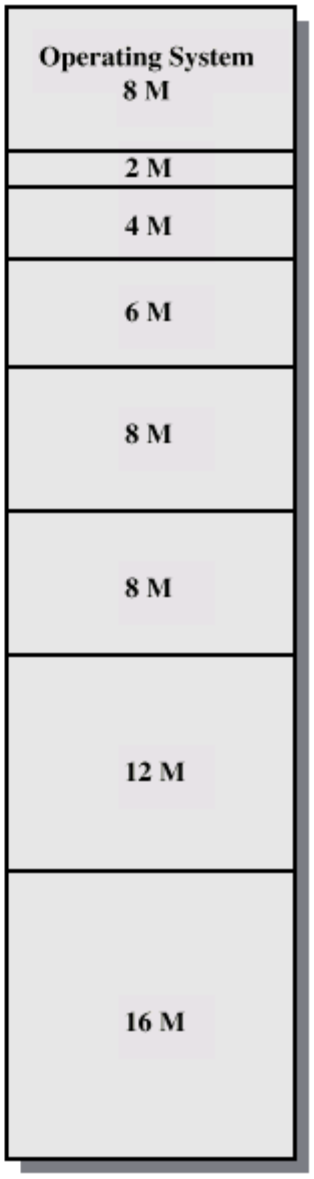
Partitioning

- Splitting memory into sections to allocate to processes (including Operating System)
- Fixed-sized partitions
  - May not be equal size
  - Process is fitted into smallest hole that will take it (best fit)
  - Some wasted memory
  - Leads to variable sized partitions

Fixed Partitioning



(a) Equal-size partitions



(b) Unequal-size partitions

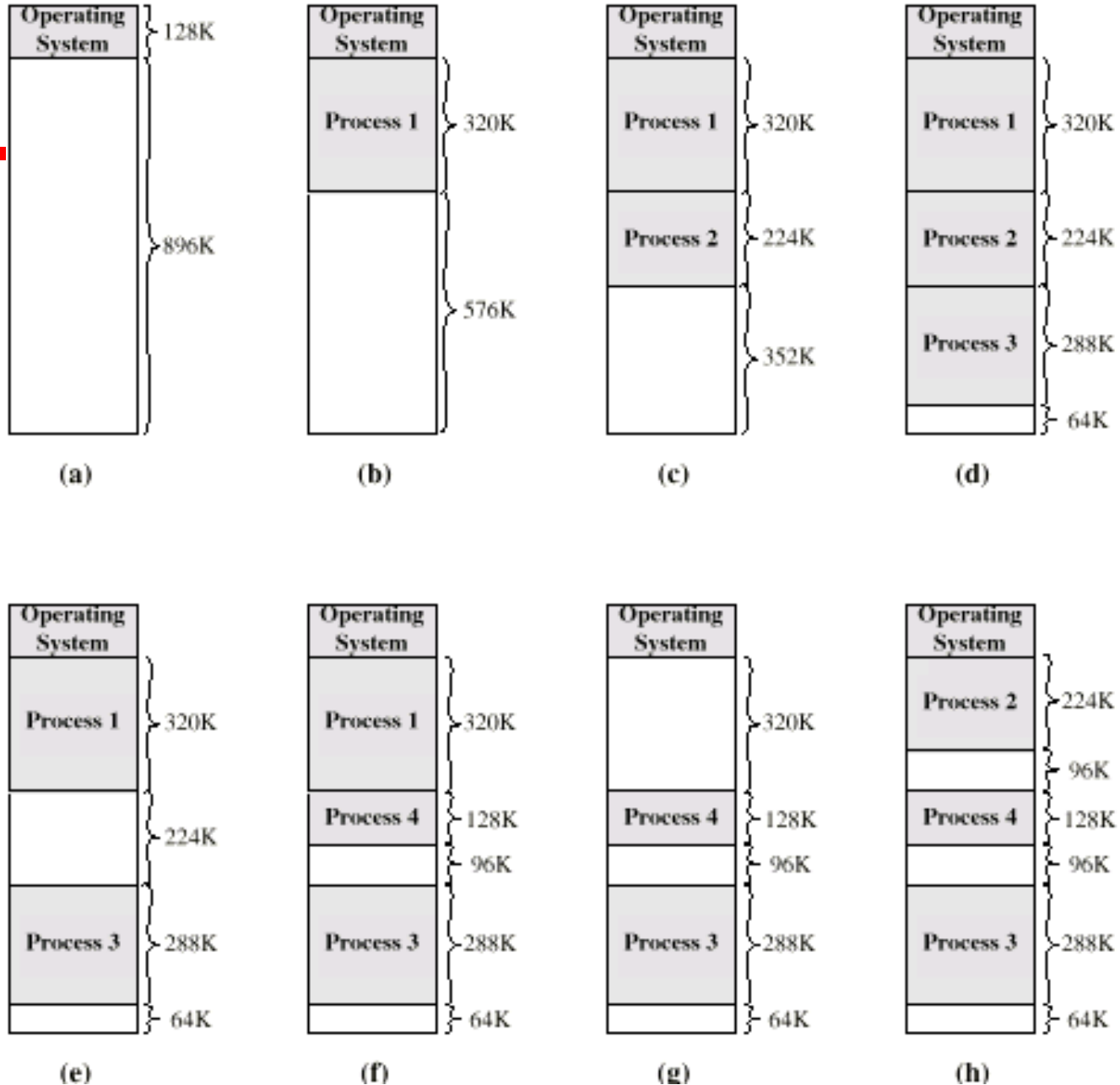
Variable Sized Partitions (1)

- Allocate exactly the required memory to a process
- This leads to a hole at the end of memory, too small to use
  - Only one small hole - less waste
- When all processes are blocked, swap out a process and bring in another
- New process may be smaller than swapped out process
- Another hole

Variable Sized Partitions (2)

- Eventually have lots of holes (fragmentation)
- Solutions:
  - Coalesce - Join adjacent holes into one large hole
  - Compaction - From time to time go through memory and move all hole into one free block (c.f. disk de-fragmentation)

Effect of Dynamic Partitioning



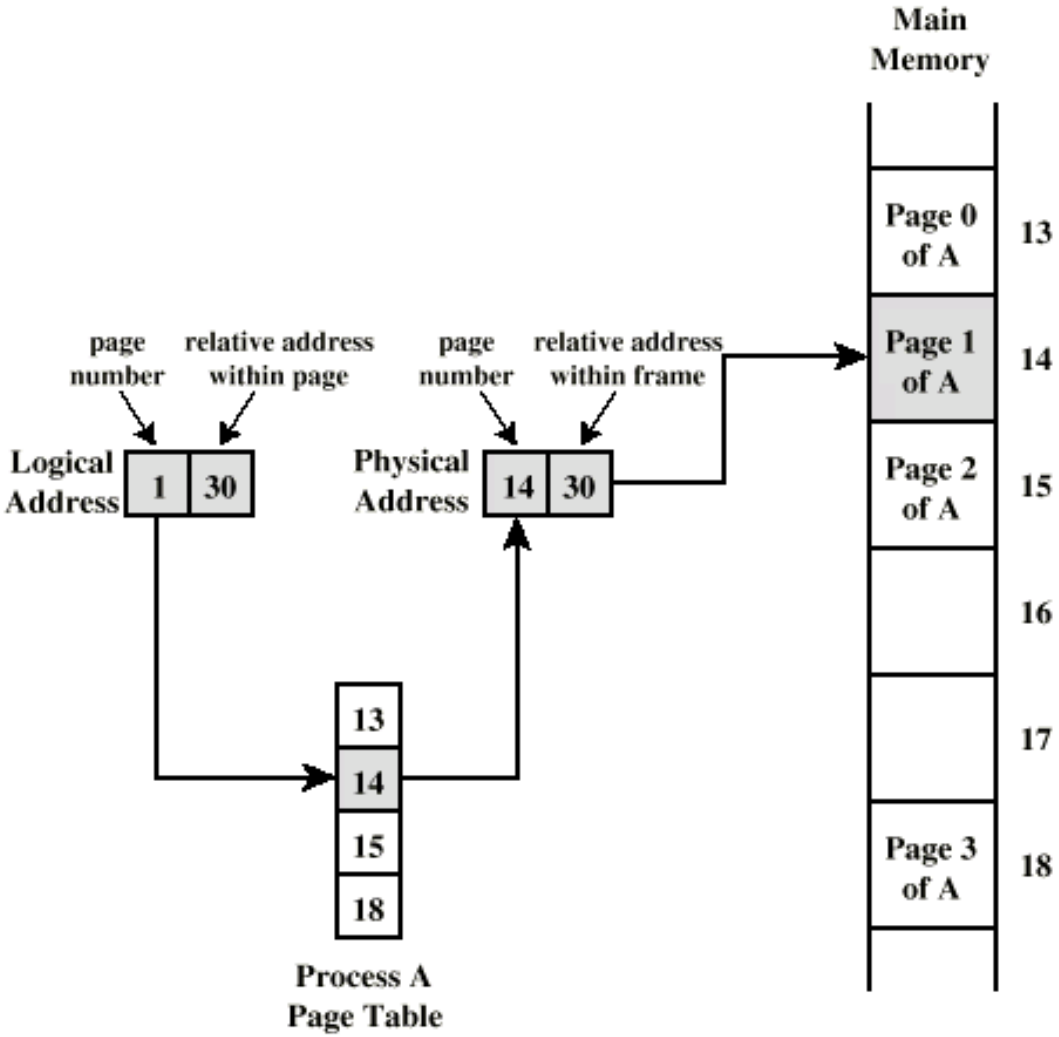
Relocation

- No guarantee that process will load into the same place in memory
- Instructions contain addresses
  - Locations of data
  - Addresses for instructions (branching)
- Logical address - relative to beginning of program
- Physical address - actual location in memory (this time)
- Automatic conversion using base address

Paging

- Split memory into equal sized, small chunks - page frames
- Split programs (processes) into equal sized small chunks - pages
- Allocate the required number page frames to a process
- Operating System maintains list of free frames
- A process does not require contiguous page frames
- Use page table to keep track

Logical and Physical Addresses - Paging



Virtual Memory

- Demand paging
  - Do not require all pages of a process in memory
  - Bring in pages as required
- Page fault
  - Required page is not in memory
  - Operating System must swap in required page
  - May need to swap out a page to make space
  - Select page to throw out based on recent history

Thrashing

- Too many processes in too little memory
- Operating System spends all its time swapping
- Little or no real work is done
- Disk light is on all the time

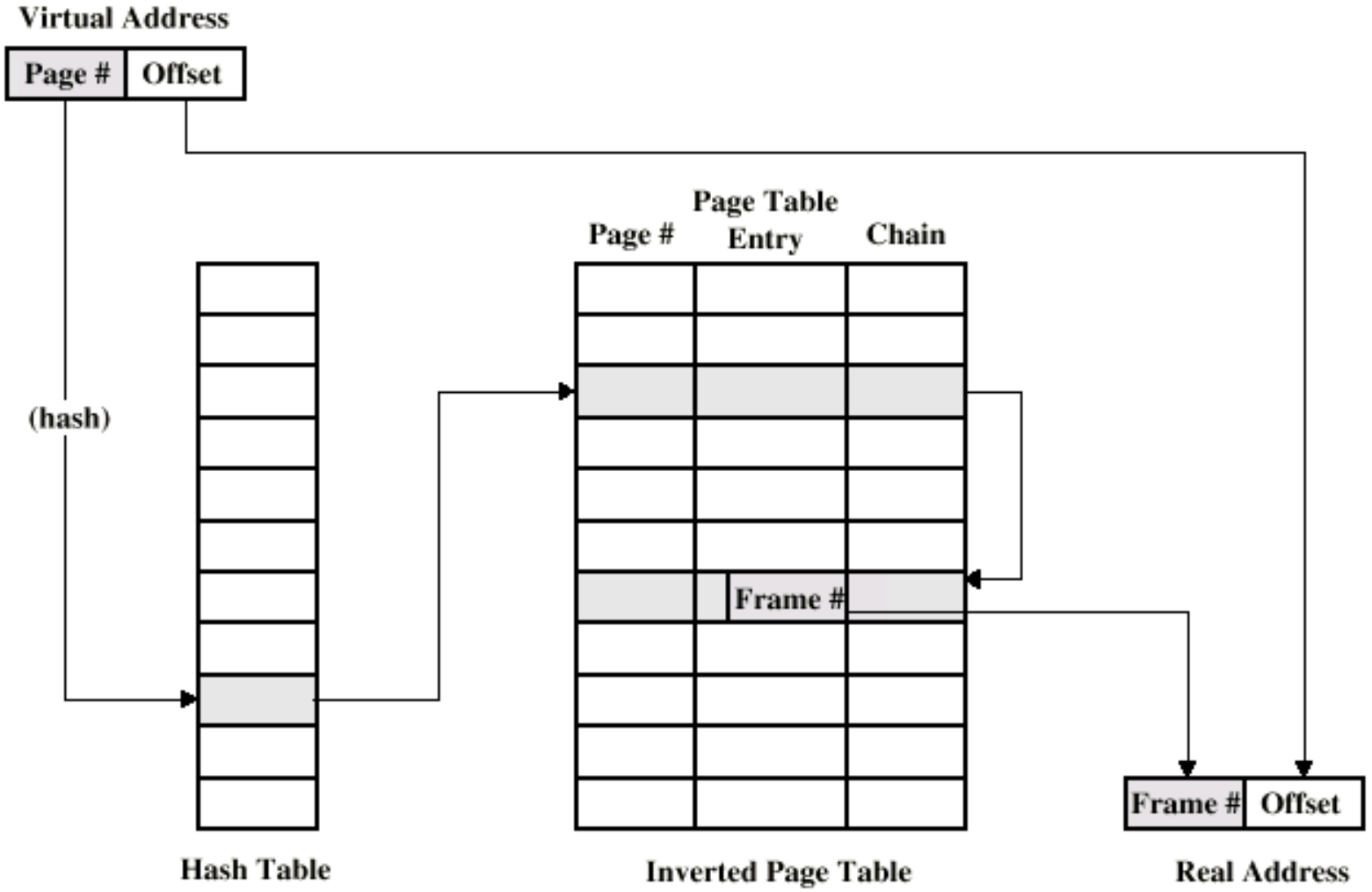
- Solutions
  - Good page replacement algorithms
  - Reduce number of processes running
  - Fit more memory

Bonus

- We do not need all of a process in memory for it to run
- We can swap in pages as required
- So - we can now run processes that are bigger than total memory available!

- Main memory is called real memory
- User/programmer sees much bigger memory - virtual memory

Page Table Structure



Segmentation

- Paging is not (usually) visible to the programmer
- Segmentation is visible to the programmer
- Usually different segments allocated to program and data
- May be a number of program and data segments

Advantages of Segmentation

- Simplifies handling of growing data structures
- Allows programs to be altered and recompiled independently, without re-linking and re-loading
- Lends itself to sharing among processes
- Lends itself to protection
- Some systems combine segmentation with paging

Required Reading

- Stallings chapter 7
- Stallings, W. Operating Systems, Internals and Design Principles, Prentice Hall 1998
- Loads of Web sites on Operating Systems

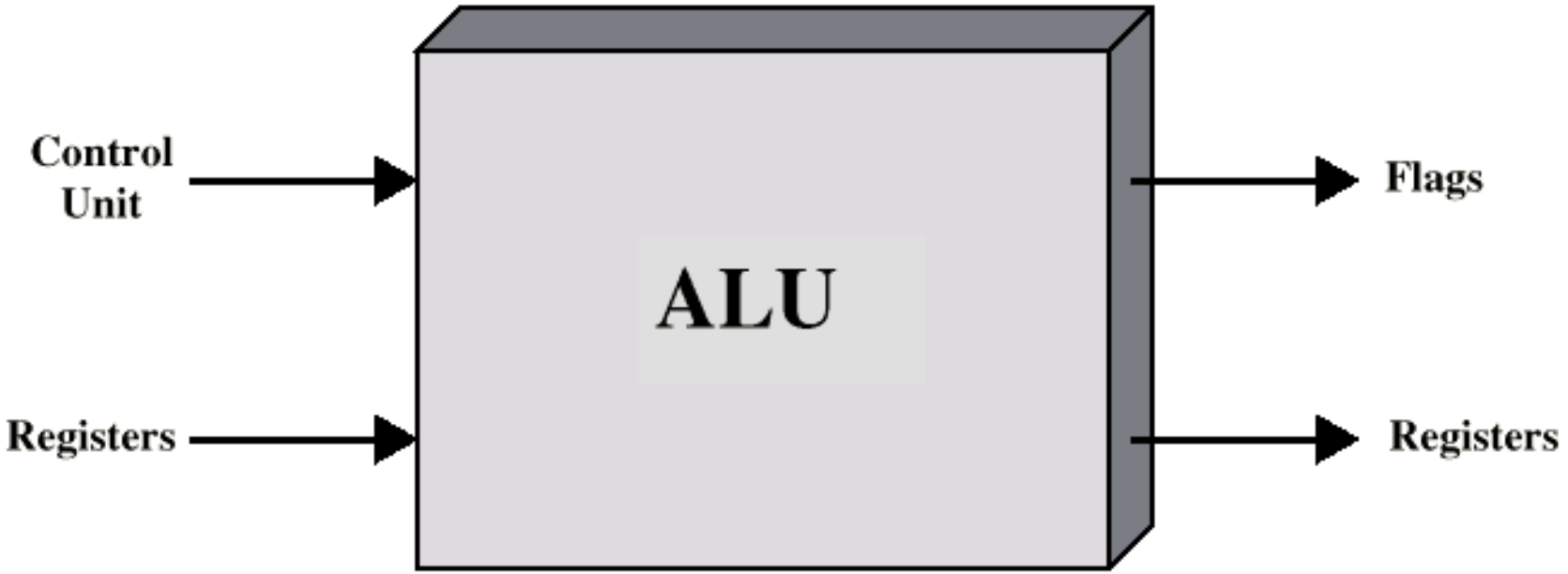
William Stallings
Computer Organization and Architecture

Chapter 8
Computer Arithmetic

Arithmetic & Logic Unit

- Does the calculations
- Everything else in the computer is there to service this unit
- Handles integers
- May handle floating point (real) numbers
- May be separate FPU (maths co-processor)
- May be on chip separate FPU (486DX +)

ALU Inputs and Outputs



Integer Representation

- Only have 0 & 1 to represent everything
- Positive numbers stored in binary
  - e.g. 41=00101001
- No minus sign
- No period
- Sign-Magnitude
- Two’s compliment

Sign-Magnitude

- Left most bit is sign bit
- 0 means positive
- 1 means negative
- +18 = 00010010
- -18 = 10010010
- Problems
  - Need to consider both sign and magnitude in arithmetic
  - Two representations of zero (+0 and -0)

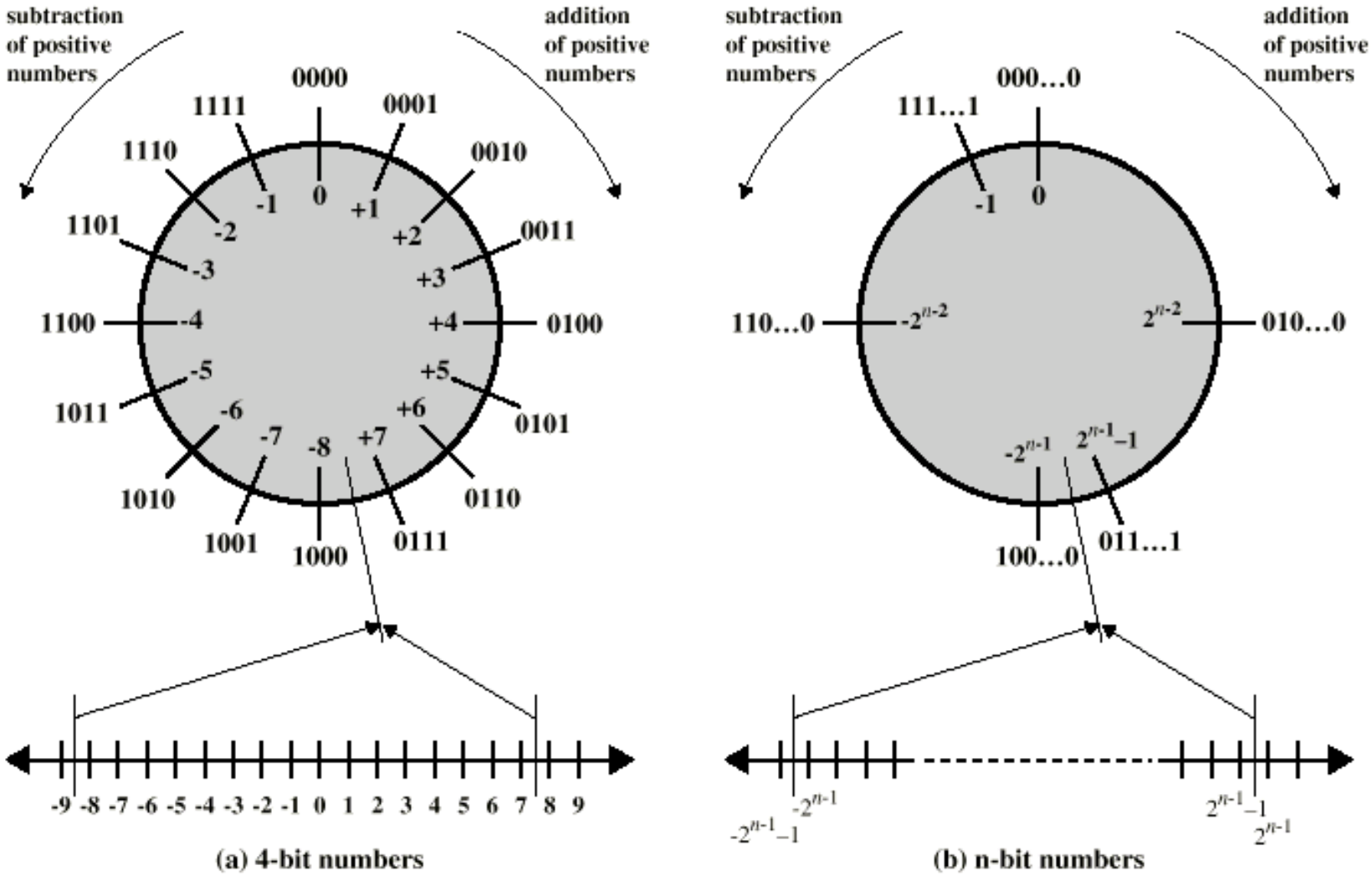
Two's Compliment

- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

Benefits

- One representation of zero
- Arithmetic works easily (see later)
- Negating is fairly easy
  - 3 = 00000011
  - Boolean complement gives 11111100
  - Add 1 to LSB 11111101

Geometric Depiction of Twos Complement Integers



Negation Special Case 1

- 0 = 00000000
- Bitwise not 11111111
- Add 1 to LSB +1
- Result 1 00000000
- Overflow is ignored, so:
- - 0 = 0 √

Negation Special Case 2

- -128 = 10000000
- bitwise not 01111111
- Add 1 to LSB +1
- Result 10000000
- So:
- -(-128) = -128 X
- Monitor MSB (sign bit)
- It should change during negation

Range of Numbers

- 8 bit 2s compliment
  - +127 = 01111111 = 2^7 -1
  - -128 = 10000000 = -2^7
- 16 bit 2s compliment
  - +32767 = 011111111 11111111 = 2^{15} - 1
  - -32768 = 100000000 00000000 = -2^{15}

Conversion Between Lengths

- Positive number pack with leading zeros
- +18 = 00010010
- +18 = 00000000 00010010
- Negative numbers pack with leading ones
- -18 = 10010010
- -18 = 11111111 10010010
- i.e. pack with MSB (sign bit)

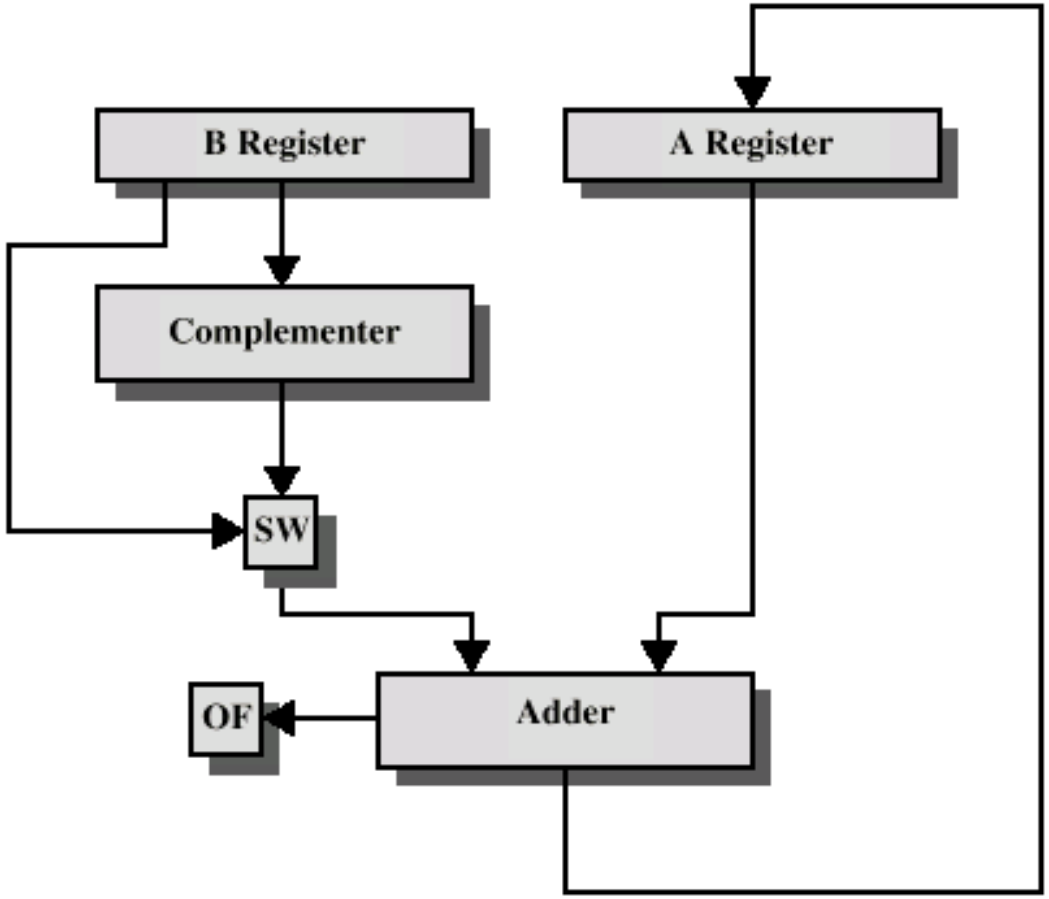
Addition and Subtraction

- Normal binary addition
- Monitor sign bit for overflow

- Take twos compliment of substahend and add to minuend
  - i.e. a - b = a + (-b)

- So we only need addition and complement circuits

Hardware for Addition and Subtraction



OF = overflow bit
SW = Switch (select addition or subtraction)

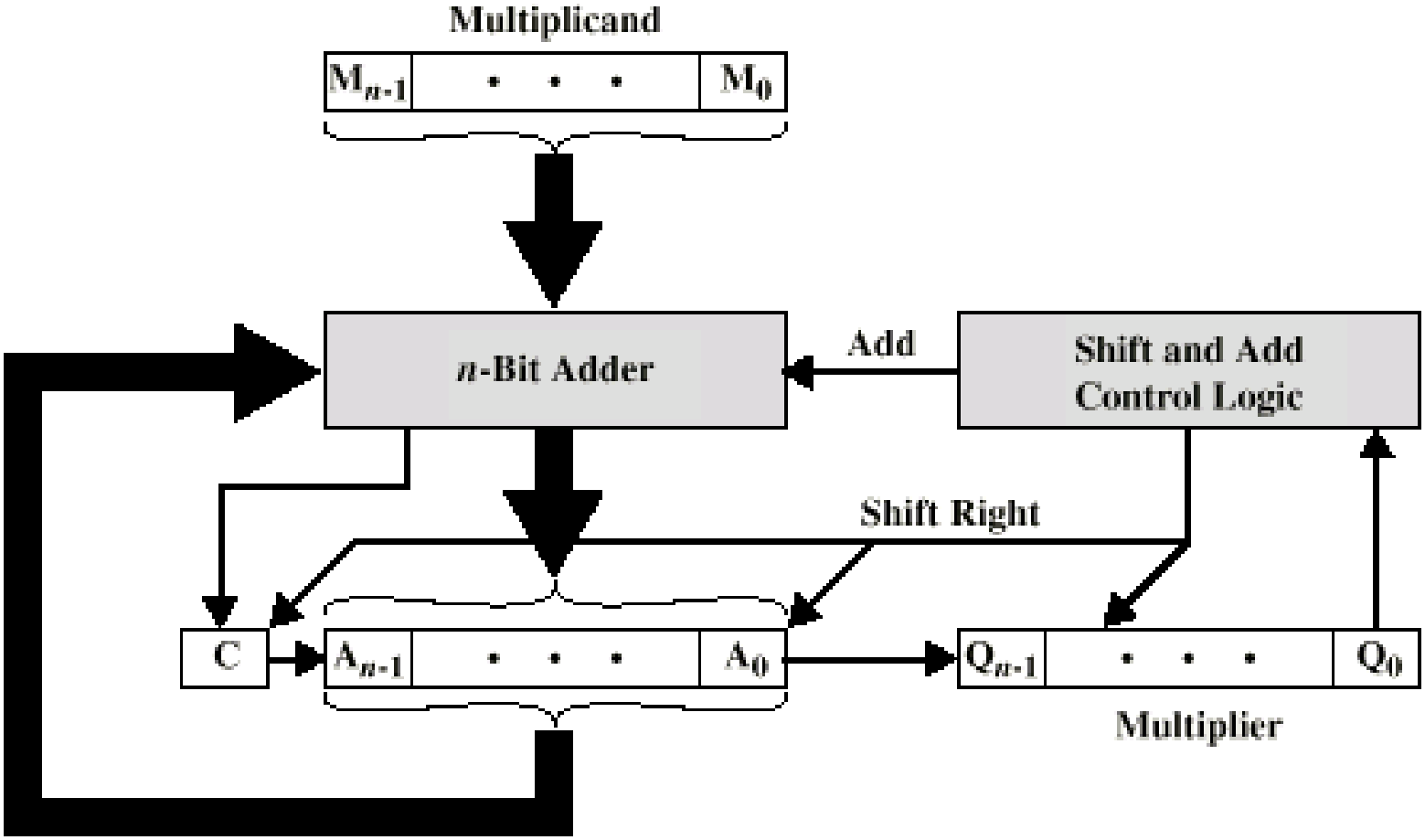
Multiplication

- Complex
- Work out partial product for each digit
- Take care with place value (column)
- Add partial products

Multiplication Example

- 1011 Multiplicand (11 dec)
- x 1101 Multiplier (13 dec)
- 1011 Partial products
- 0000 Note: if multiplier bit is 1 copy
- 1011 multiplicand (place value)
- 1011 otherwise zero
- 10001111 Product (143 dec)
- Note: need double length result

Unsigned Binary Multiplication

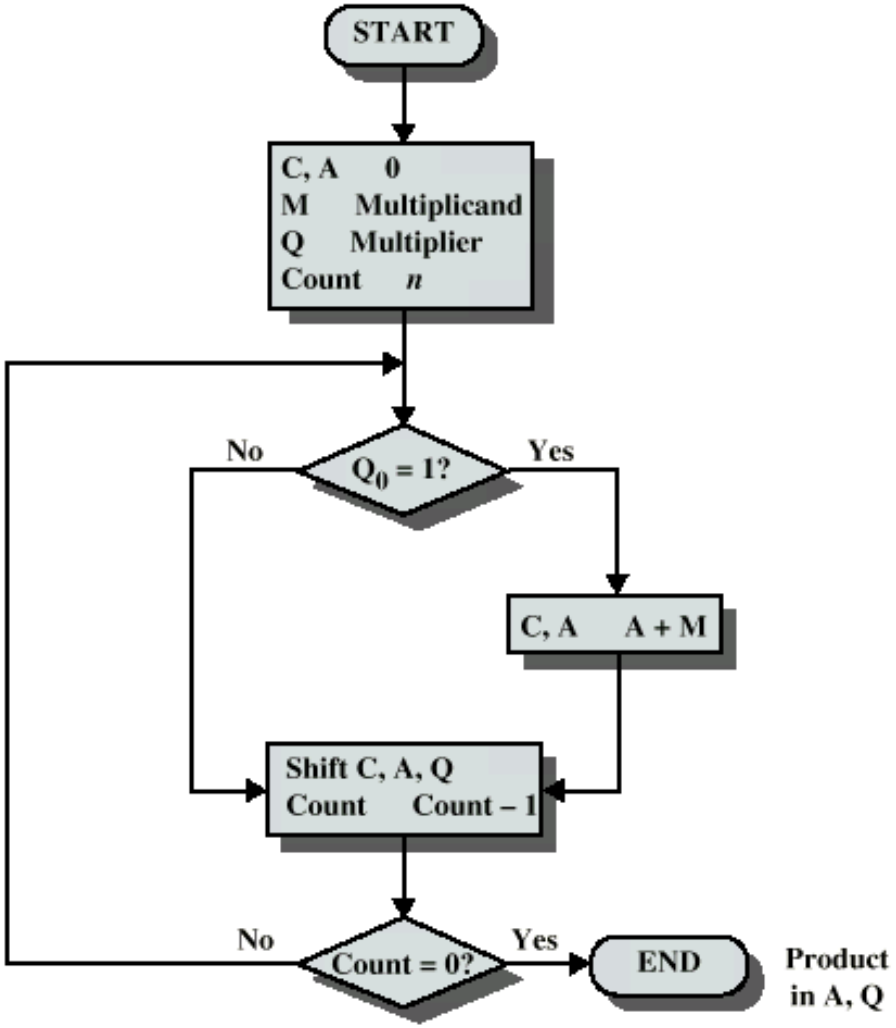


(a) Block Diagram

Execution of Example

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

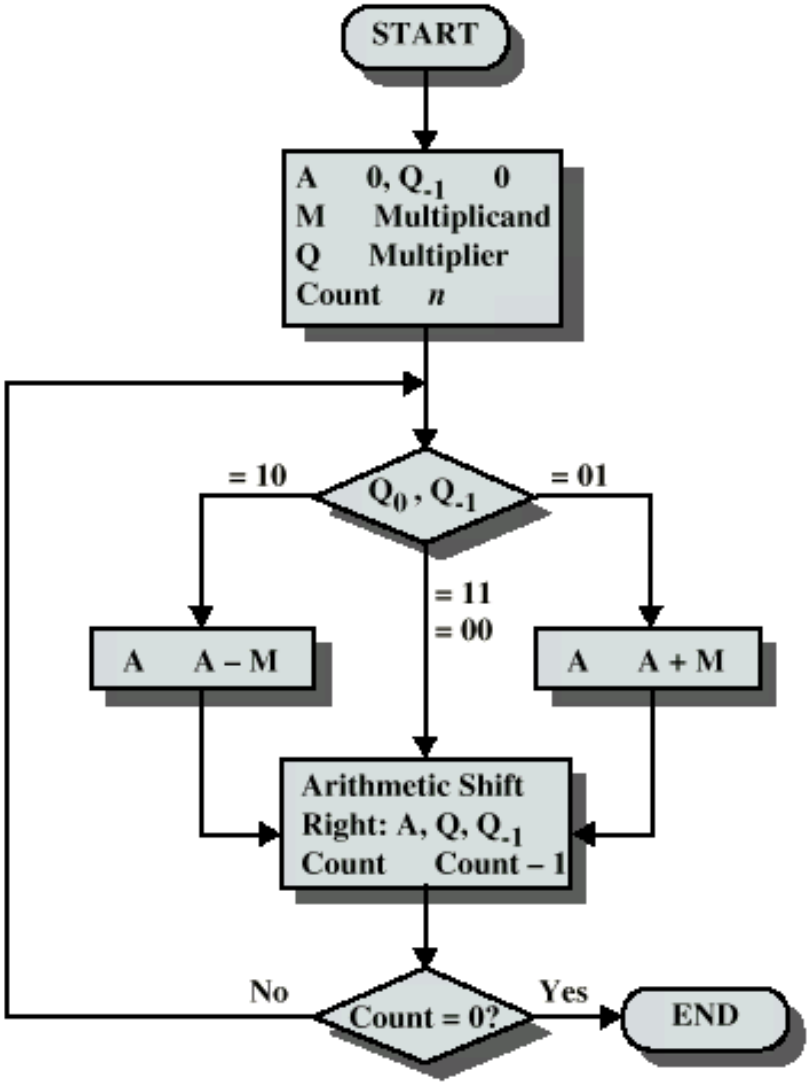
Flowchart for Unsigned Binary Multiplication



Multiplying Negative Numbers

- This does not work!
- Solution 1
  - Convert to positive if required
  - Multiply as above
  - If signs were different, negate answer
- Solution 2
  - Booth's algorithm

Booth's Algorithm



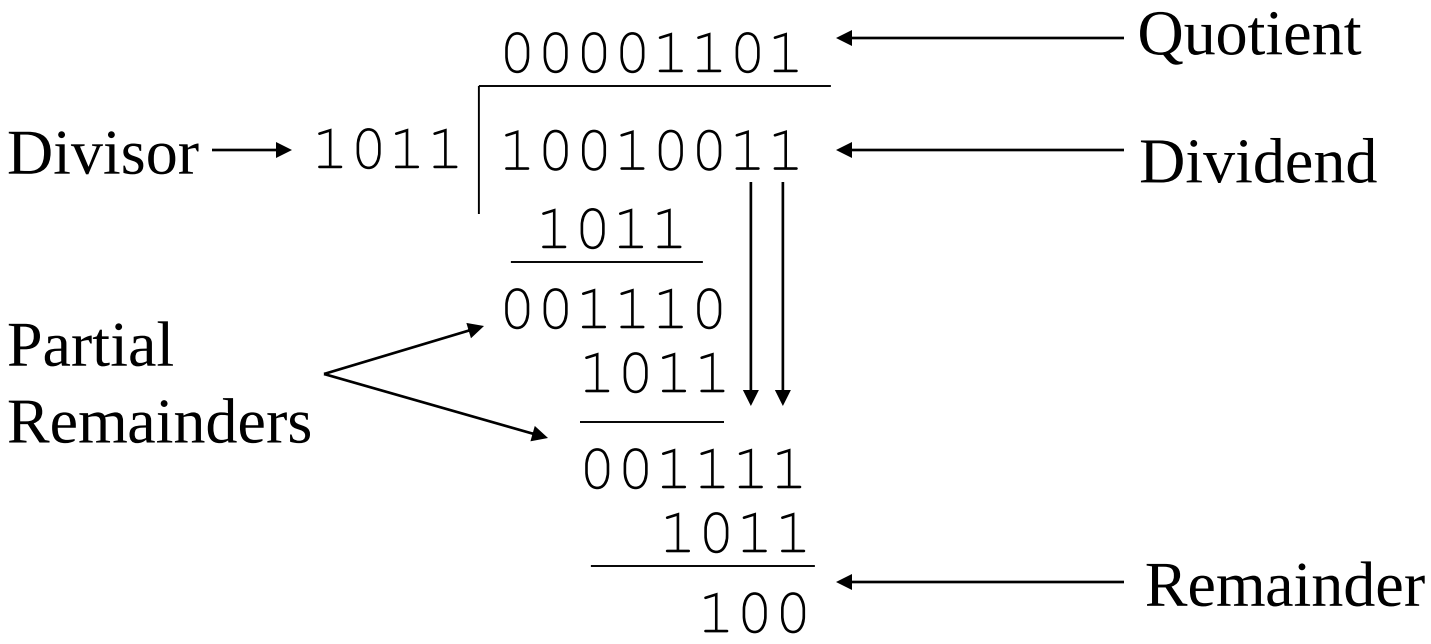
Example of Booth's Algorithm

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A A - M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

Division

- More complex than multiplication
- Negative numbers are really bad!
- Based on long division

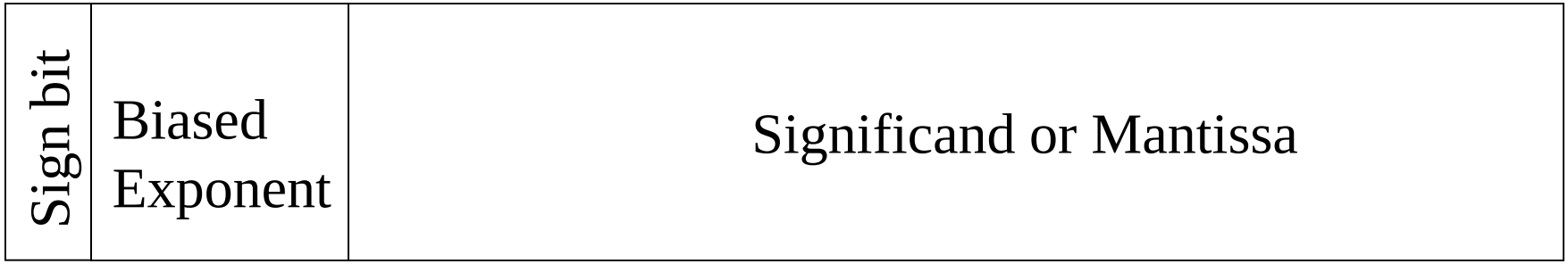
Division of Unsigned Binary Integers



Real Numbers

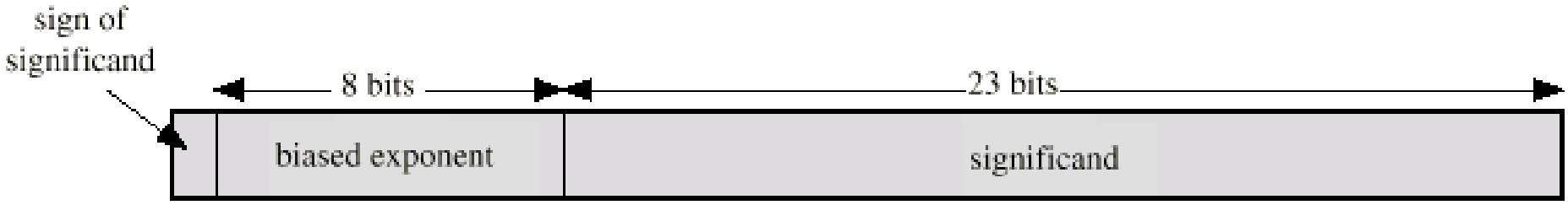
- Numbers with fractions
- Could be done in pure binary
  - $1001.1010 = 2^4 + 2^0 + 2^{-1} + 2^{-3} = 9.625$
- Where is the binary point?
- Fixed?
  - Very limited
- Moving?
  - How do you show where it is?

Floating Point



- +/- .significand x $2^{exponent}$
- Misnomer
- Point is actually fixed between sign bit and body of mantissa
- Exponent indicates place value (point position)

Floating Point Examples



(a) Format

0.11010001	2^{10100}	=	0	10010011	10100010000000000000000
-0.11010001	2^{10100}	=	1	10010011	10100010000000000000000
0.11010001	2^{-10100}	=	0	01101011	10100010000000000000000
-0.11010001	2^{-10100}	=	1	01101011	10100010000000000000000

(b) Examples

Signs for Floating Point

- Mantissa is stored in 2s compliment
- Exponent is in excess or biased notation
  - e.g. Excess (bias) 128 means
  - 8 bit exponent field
  - Pure value range 0-255
  - Subtract 128 to get correct value
  - Range -128 to +127

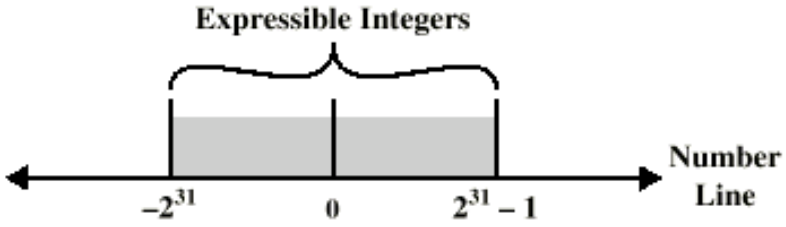
Normalization

- FP numbers are usually normalized
- i.e. exponent is adjusted so that leading bit (MSB) of mantissa is 1
- Since it is always 1 there is no need to store it
- (c.f. Scientific notation where numbers are normalized to give a single digit before the decimal point
- e.g. 3.123×10^3)

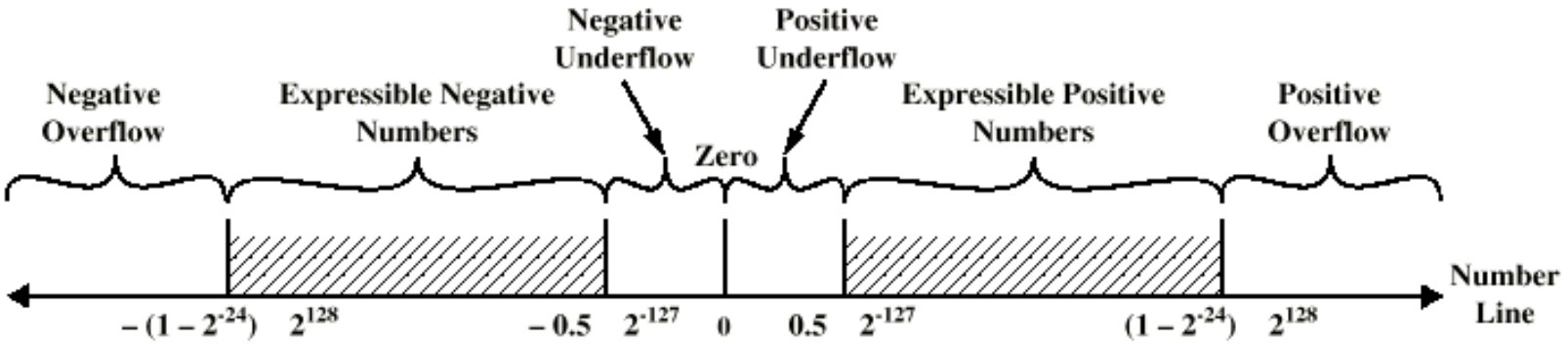
FP Ranges

- For a 32 bit number
  - 8 bit exponent
  - +/- $2^{256} \approx 1.5 \times 10^{77}$
- Accuracy
  - The effect of changing lsb of mantissa
  - 23 bit mantissa $2^{-23} \approx 1.2 \times 10^{-7}$
  - About 6 decimal places

Expressible Numbers



(a) Twos Complement Integers



(b) Floating-Point Numbers

IEEE 754

- Standard for floating point storage
- 32 and 64 bit standards
- 8 and 11 bit exponent respectively
- Extended formats (both mantissa and exponent) for intermediate results

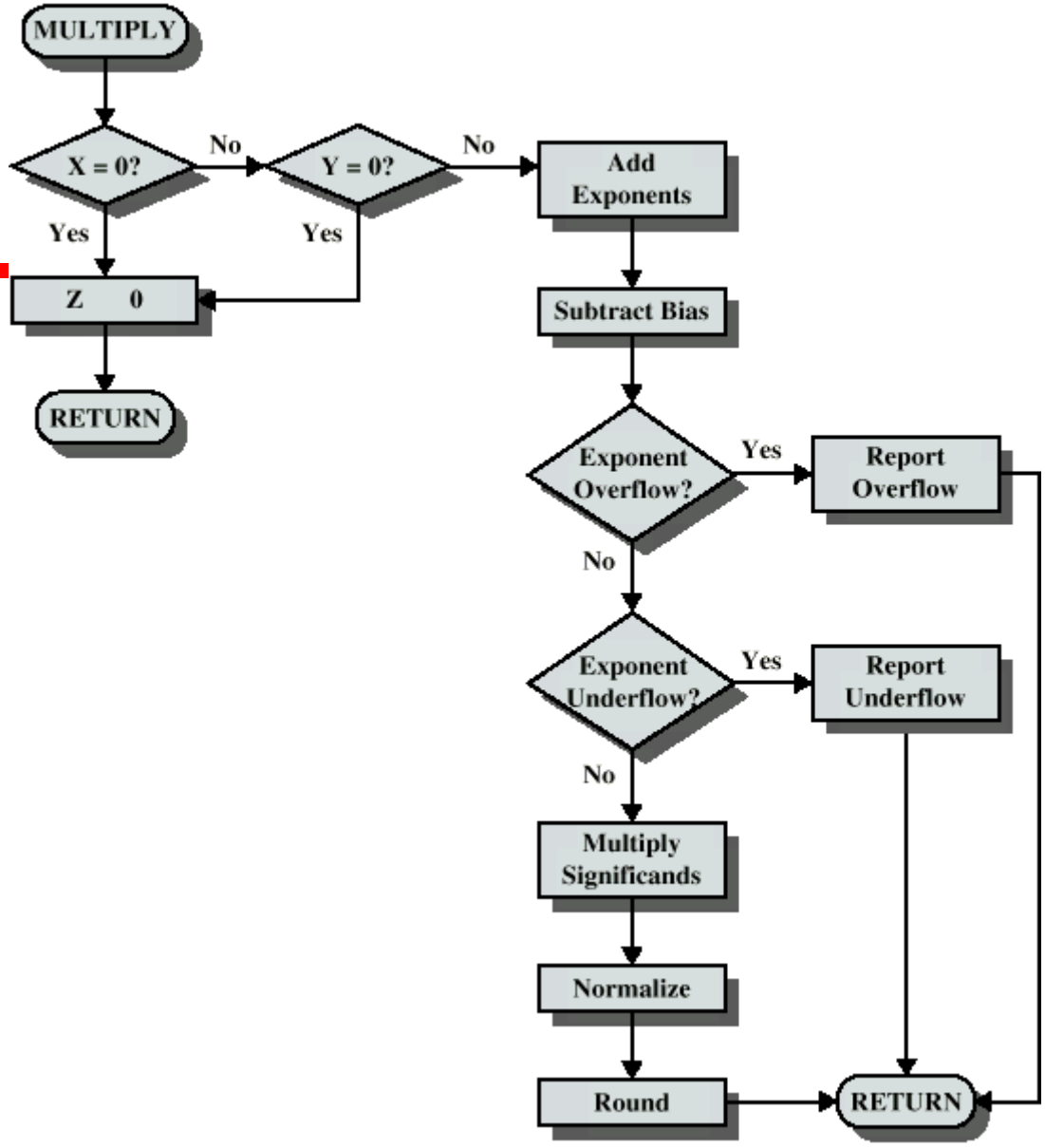
FP Arithmetic +/-

- Check for zeros
- Align significands (adjusting exponents)
- Add or subtract significands
- Normalize result

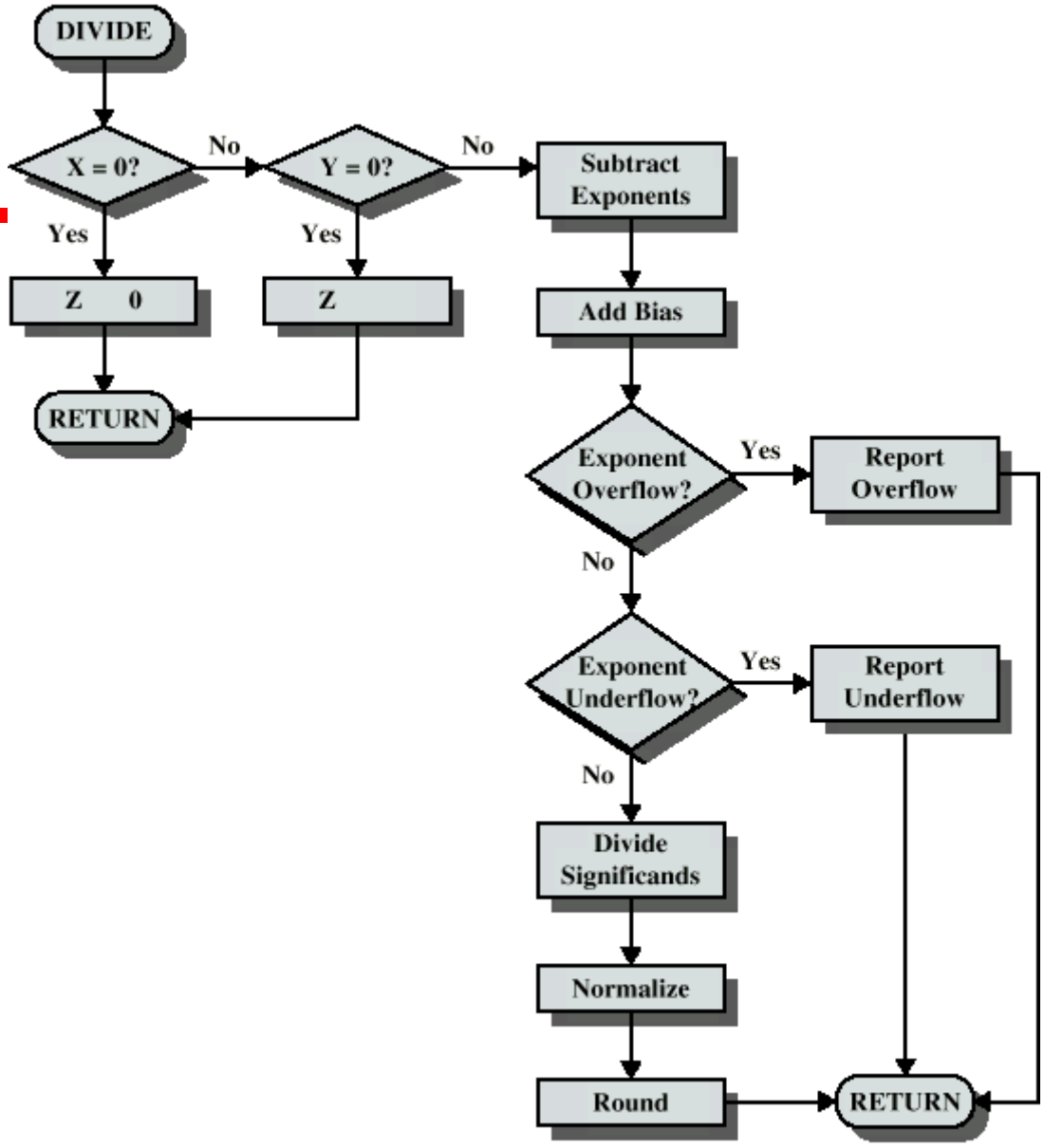
FP Arithmetic x/÷

- Check for zero
- Add/subtract exponents
- Multiply/divide significands (watch sign)
- Normalize
- Round
- All intermediate results should be in double length storage

Floating Point Multiplication



Floating Point Division



Required Reading

- Stallings Chapter 8
- IEEE 754 on IEEE Web site

William Stallings
Computer Organization and Architecture

Chapter 9
Instruction Sets: Characteristics and Functions

What is an instruction set?

- The complete collection of instructions that are understood by a CPU
- Machine Code
- Binary
- Usually represented by assembly codes

Elements of an Instruction

- Operation code (Op code)
  - Do this
- Source Operand reference
  - To this
- Result Operand reference
  - Put the answer here
- Next Instruction Reference
  - When you have done that, do this...

Where have all the Operands gone?

- Long time passing….
- (If you don’t understand, you’re too young!)
- Main memory (or virtual memory or cache)
- CPU register
- I/O device

Instruction Representation

- In machine code each instruction has a unique bit pattern
- For human consumption (well, programmers anyway) a symbolic representation is used
  - e.g. ADD, SUB, LOAD
- Operands can also be represented in this way
  - ADD A,B

Instruction Types

- Data processing
- Data storage (main memory)
- Data movement (I/O)
- Program flow control

Number of Addresses (a)

- 3 addresses
  - Operand 1, Operand 2, Result
  - a = b + c;
  - May be a forth - next instruction (usually implicit)
  - Not common
  - Needs very long words to hold everything

Number of Addresses (b)

- 2 addresses
  - One address doubles as operand and result
  - a = a + b
  - Reduces length of instruction
  - Requires some extra work
    - Temporary storage to hold some results

Number of Addresses (c)

- 1 address
  - Implicit second address
  - Usually a register (accumulator)
  - Common on early machines

Number of Addresses (d)

- 0 (zero) addresses
  - All addresses implicit
  - Uses a stack
  - e.g. push a
  - push b
  - add
  - pop c

  - c = a + b

How Many Addresses

- More addresses
  - More complex (powerful?) instructions
  - More registers
    - Inter-register operations are quicker
  - Fewer instructions per program
- Fewer addresses
  - Less complex (powerful?) instructions
  - More instructions per program
  - Faster fetch/execution of instructions

Design Decisions (1)

- Operation repertoire
  - How many ops?
  - What can they do?
  - How complex are they?
- Data types
- Instruction formats
  - Length of op code field
  - Number of addresses

Design Decisions (2)

- Registers
  - Number of CPU registers available
  - Which operations can be performed on which registers?
- Addressing modes (later…)
- RISC v CISC

Types of Operand

- Addresses
- Numbers
  - Integer/floating point
- Characters
  - ASCII etc.
- Logical Data
  - Bits or flags
- (Aside: Is there any difference between numbers and characters? Ask a C programmer!)

Pentium Data Types

- 8 bit Byte
- 16 bit word
- 32 bit double word
- 64 bit quad word
- Addressing is by 8 bit unit
- A 32 bit double word is read at addresses divisible by 4

Specific Data Types

- General - arbitrary binary contents
- Integer - single binary value
- Ordinal - unsigned integer
- Unpacked BCD - One digit per byte
- Packed BCD - 2 BCD digits per byte
- Near Pointer - 32 bit offset within segment
- Bit field
- Byte String
- Floating Point

Pentium Floating Point Data Types

- See Stallings p324

Types of Operation

- Data Transfer
- Arithmetic
- Logical
- Conversion
- I/O
- System Control
- Transfer of Control

Data Transfer

- Specify
  - Source
  - Destination
  - Amount of data
- May be different instructions for different movements
  - e.g. IBM 370
- Or one instruction and different addresses
  - e.g. VAX

Arithmetic

- Add, Subtract, Multiply, Divide
- Signed Integer
- Floating point ?
- May include
  - Increment (a++)
  - Decrement (a--)
  - Negate (-a)

Logical

- Bitwise operations
- AND, OR, NOT

Conversion

- E.g. Binary to Decimal

Input/Output

- May be specific instructions
- May be done using data movement instructions (memory mapped)
- May be done by a separate controller (DMA)

Systems Control

- Privileged instructions
- CPU needs to be in specific state
  - Ring 0 on 80386+
  - Kernel mode
- For operating systems use

Transfer of Control

- Branch
  - e.g. branch to x if result is zero
- Skip
  - e.g. increment and skip if zero
  - ISZ Register1
  - Branch xxxx
  - ADD A
- Subroutine call
  - c.f. interrupt call

Foreground Reading

- Pentium and PowerPC operation types
- Stallings p338 et. Seq.

Byte Order
(A portion of chips?)

- What order do we read numbers that occupy more than one byte
- e.g. (numbers in hex to make it easy to read)
- 12345678 can be stored in 4x8bit locations as follows
-

Byte Order (example)

Address	Value (1)	Value(2)
184	12	78
185	34	56
186	56	34
186	78	12

- i.e. read top down or bottom up?

Byte Order Names

- The problem is called Endian
- The system on the left has the least significant byte in the lowest address
- This is called big-endian
- The system on the right has the least significant byte in the highest address
- This is called little-endian

Standard…What Standard?

- Pentium (80x86), VAX are little-endian
- IBM 370, Moterola 680x0 (Mac), and most RISC are big-endian
- Internet is big-endian
  - Makes writing Internet programs on PC more awkward!
  - WinSock provides htoi and itoh (Host to Internet & Internet to Host) functions to convert