Computer Organization

Lecture 3

The Interconnection and System Buses

Lecture 3

- What is a program?
- Instruction Cycle
- Interrupts
- Interconnection Structures
- Bus Structures

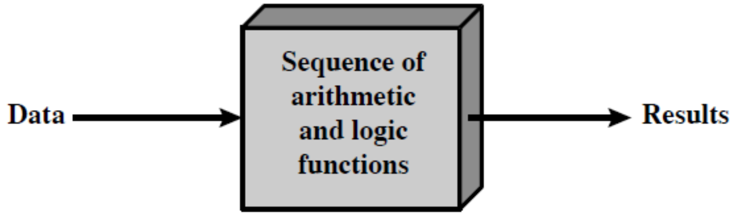
How computer performs its functions

- Basic logic components (e.g., logic gates, capacitors, and transistors) are combined to
  - Store binary data
  - Perform arithmetic and logical operations on data
- A specific configuration of logic components are designated for a particular computation
  - Hardwired program -> Inflexible

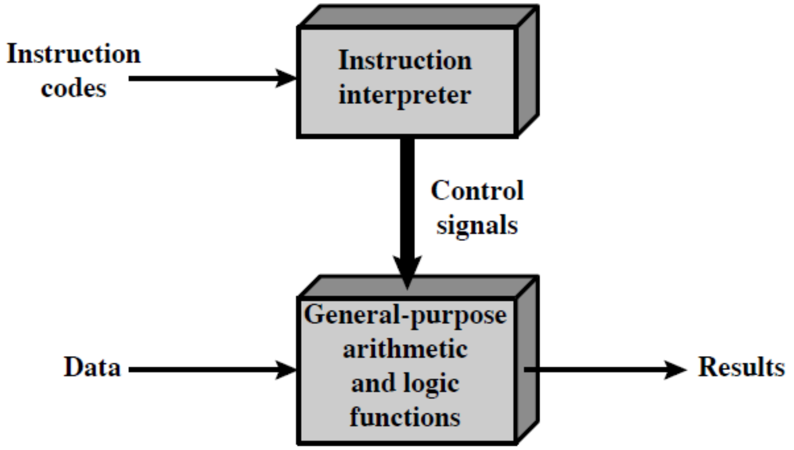
What is a program?

- Instead, a general-purpose configuration of arithmetic and logic functions is constructed
  - Perform various functions based on control signals
- A set of instructions (program codes) are fed to part of the hardware (control unit) for interpretation and producing control signals
- Each instruction carries out a single arithmetic or logical operation
- A program is called *software*

Hardware and Software Approach



(a) Programming in hardware



(b) Programming in software

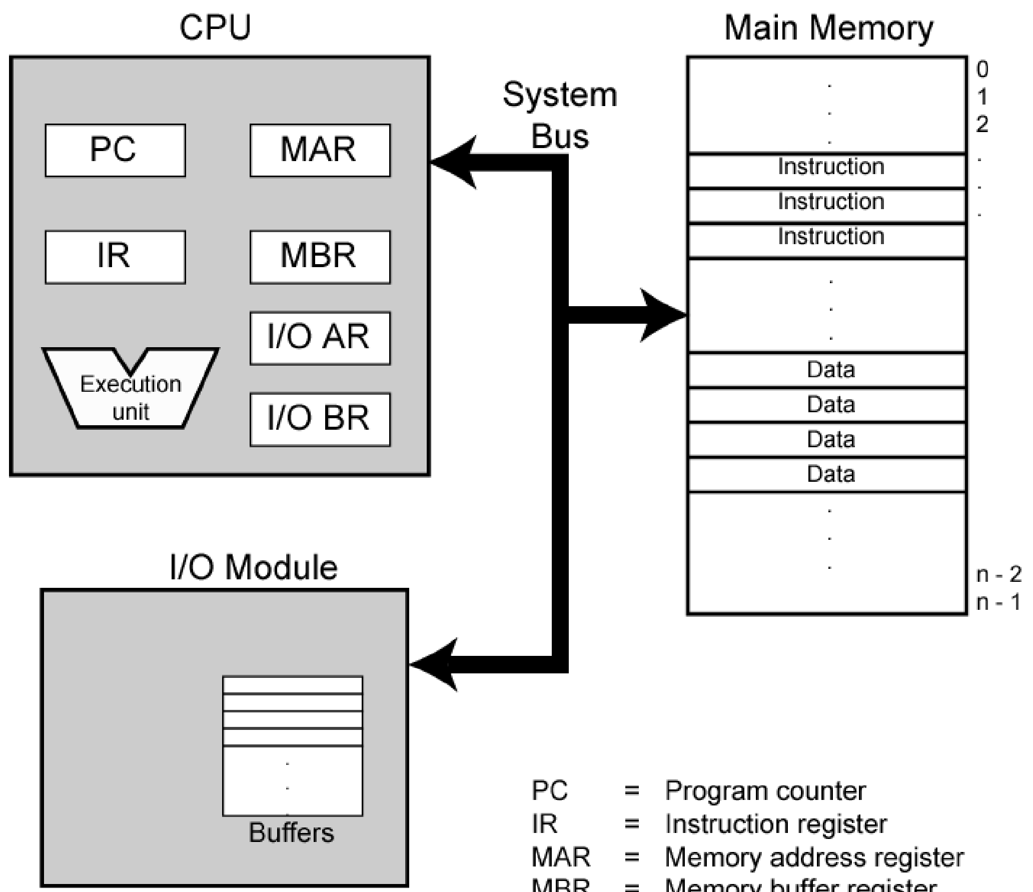
Components

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

Program Formation & Execution – A Simplified View

1. A programmer writes codes/instructions (arranged in a particular sequence to solve a computational problem) to form a program
2. The program is stored/saved in permanent storage medium (e.g., harddisk, floppy disk)
3. When the program is run/executed, the program instructions are fetched to main memory
4. CPU then fetches each instruction from the main memory and executes it

Computer Components: Top Level View

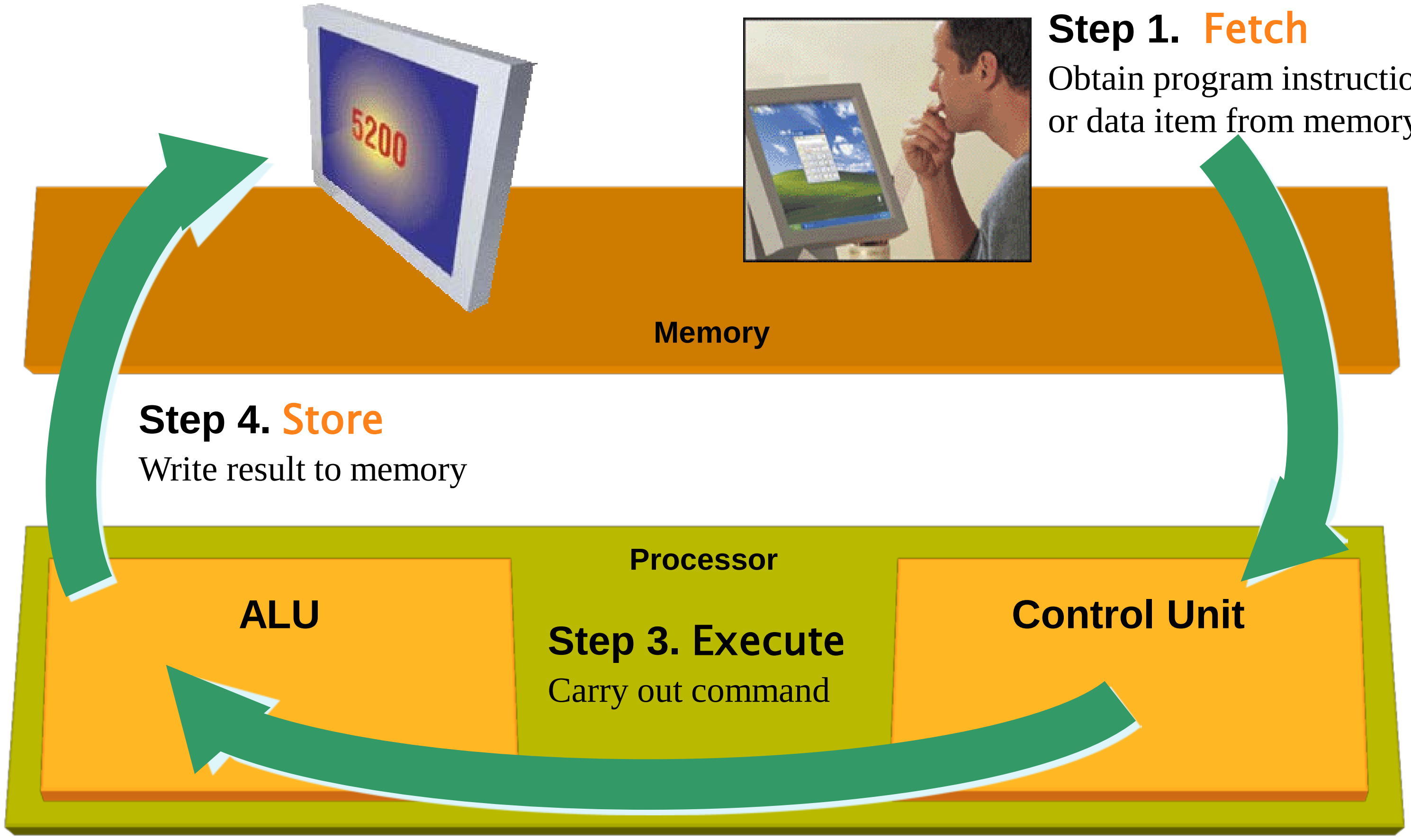


PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

Instruction Cycle

- To execute an instruction, a sequence of steps are carried out by the computer
- Four main steps
  - Fetch
  - Decode
  - Execute
  - Store

Instruction Cycle



Step 1. Fetch
Obtain program instruction or data item from memory

Step 2. Decode
Translate instruction into commands

Step 3. Execute
Carry out command

Step 4. Store
Write result to memory

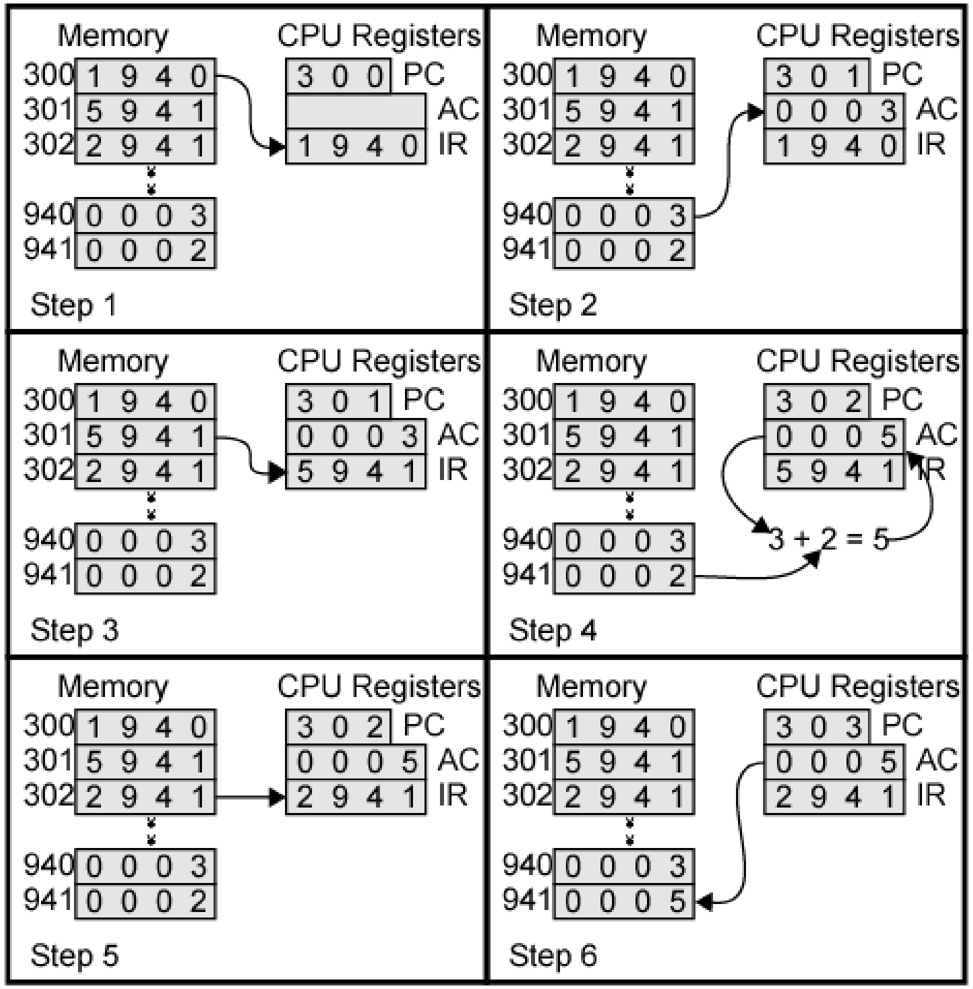
Fetch Cycle

- Program Counter (PC) holds address of next instruction to fetch
- Processor fetches instruction from memory location *pointed to* by PC
- Increment PC automatically
  - Based on the fact that instructions are stored in memory sequentially, with the next instruction being stored in the next memory location
  - Unless told otherwise (e.g., in case of branching)
- Instruction loaded into Instruction Register (IR)

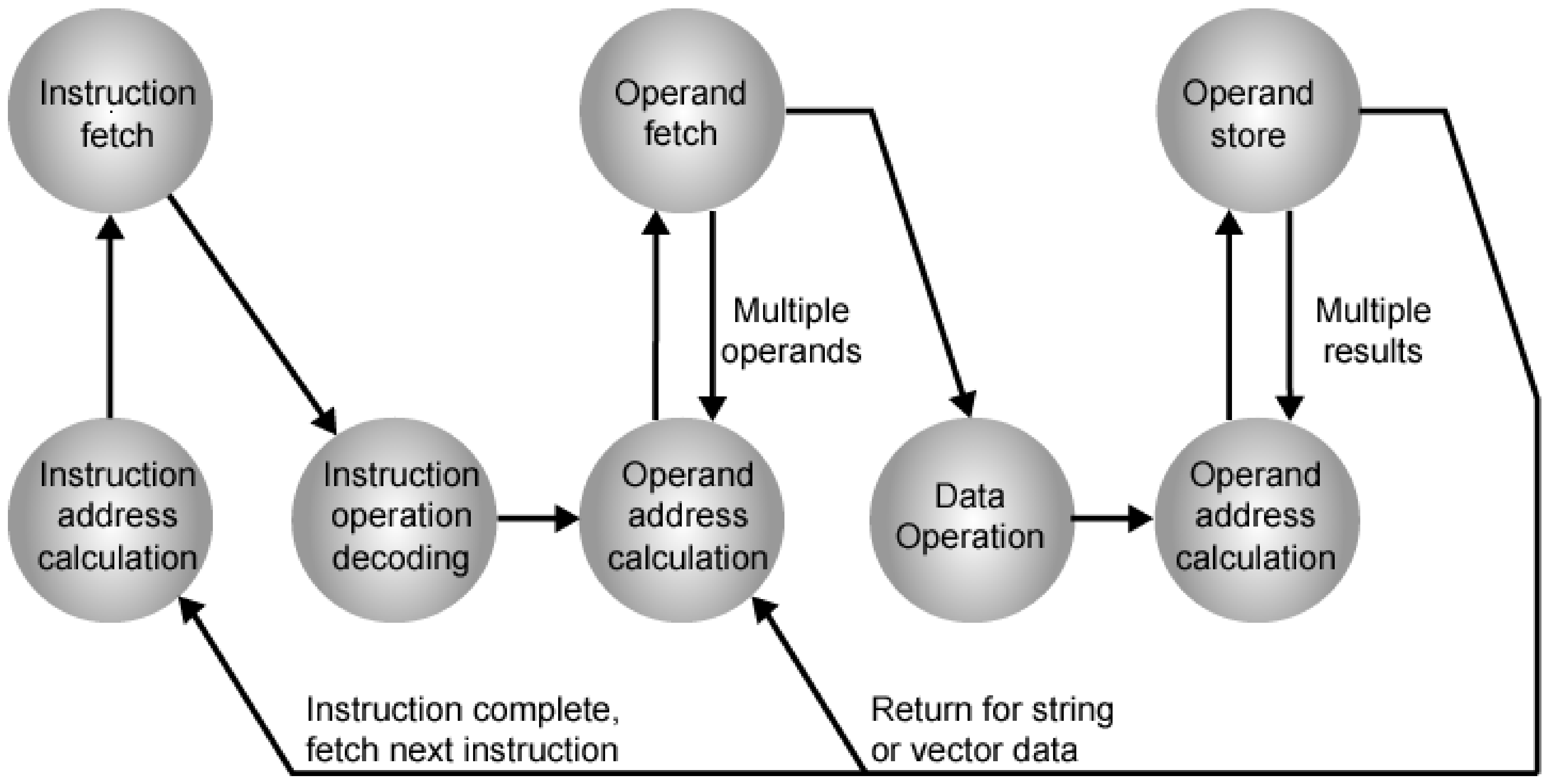
Execute Cycle

- Processor interprets instruction (decoding) and performs required actions (executing)
- Four categories of actions
  1. Processor-memory
     - data transfer between CPU and main memory
  2. Processor-I/O
     - Data transfer between CPU and I/O module
  3. Data processing
     - Some arithmetic or logical operation on data
  4. Control
     - Alteration of sequence of operations
     - E.g., **jump**
  - Combination of above
     - E.g., store the result after calculation

Example of Program Execution



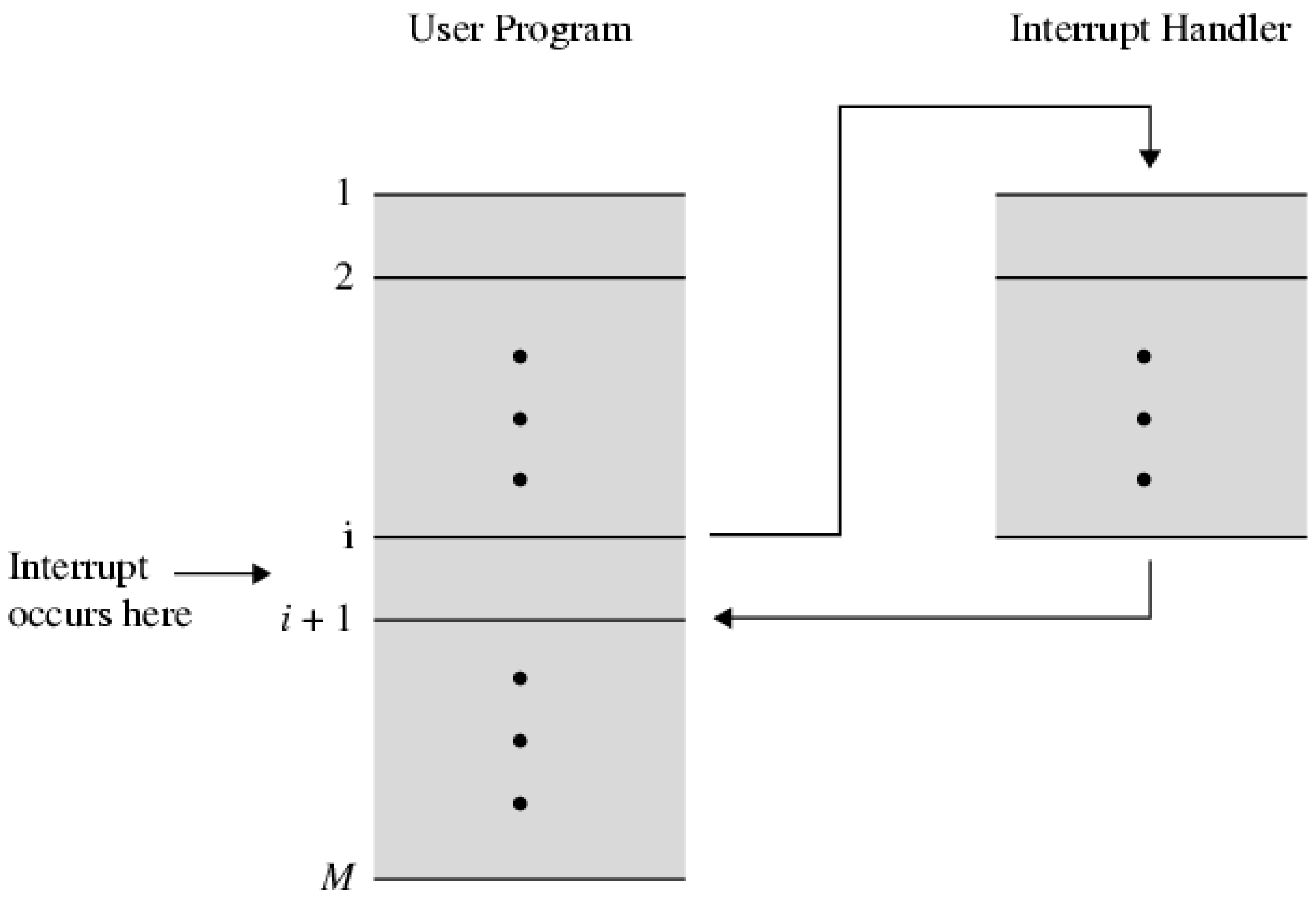
Instruction Cycle State Diagram


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Instruction complete, fetch next instruction
Return for string or vector data

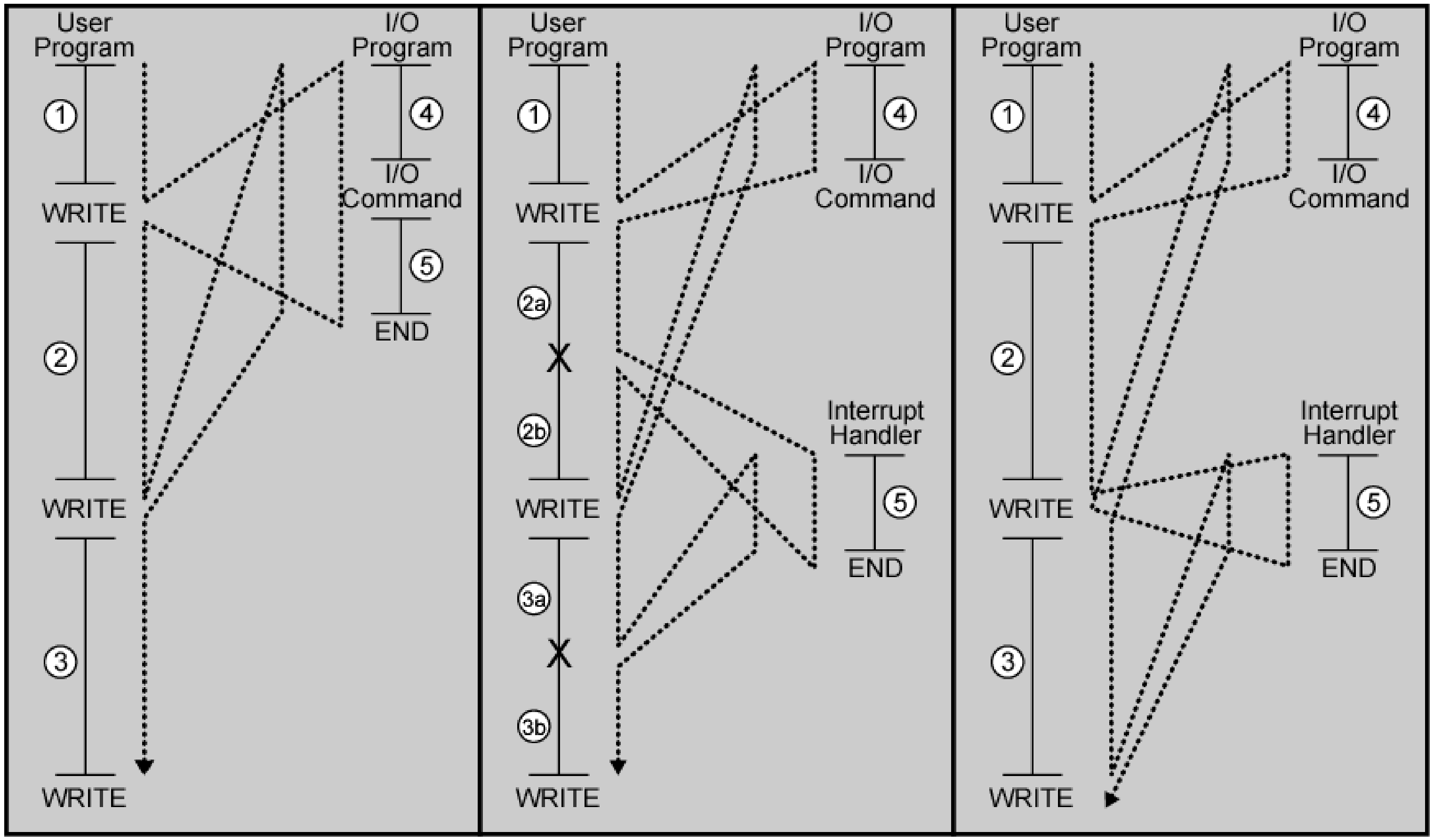
Interrupts

- Mechanism by which other modules (e.g., I/O) may interrupt normal sequence of processing
  - Suppose we have two instruction cycles, A and B, in which B is executed after A according to the pre-defined sequence
  - An interrupt may occur during A
  - The interrupt instructions, instead of B, are then executed after completion of A
- Types of interrupts
  - 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

Transfer of Control via Interrupts



Program Flow Control

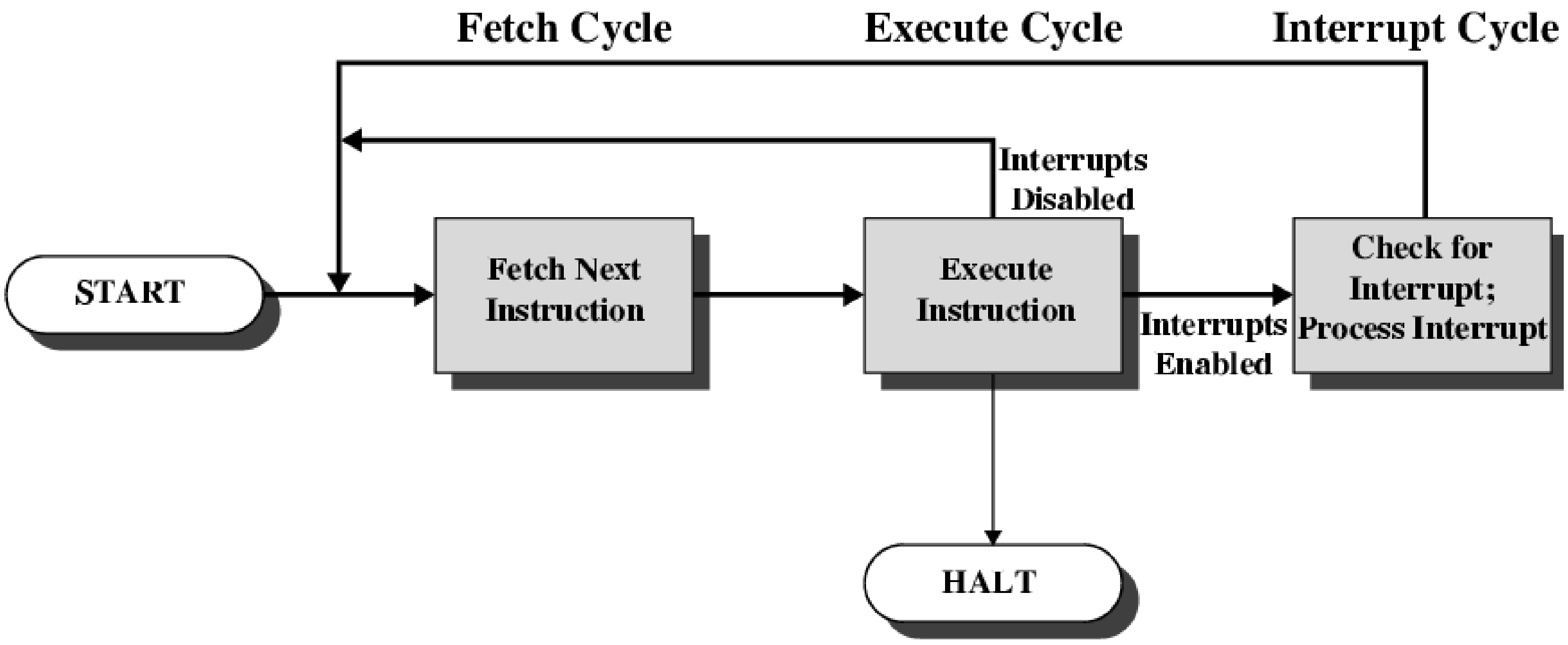


(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

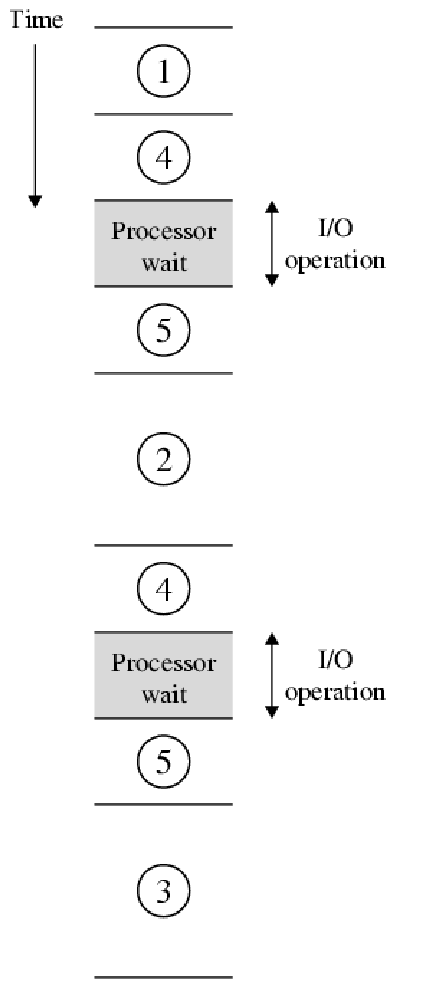
Interrupt Cycle

- Added to instruction cycle
- Processor checks for interrupt
  - Indicated by an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending
  1. Suspend execution of current program
  2. Save context
     - E.g., values in PC, AC and registers
  3. Set PC to start address of interrupt handler routine
  4. Process interrupt
  5. Restore context and continue the interrupted program

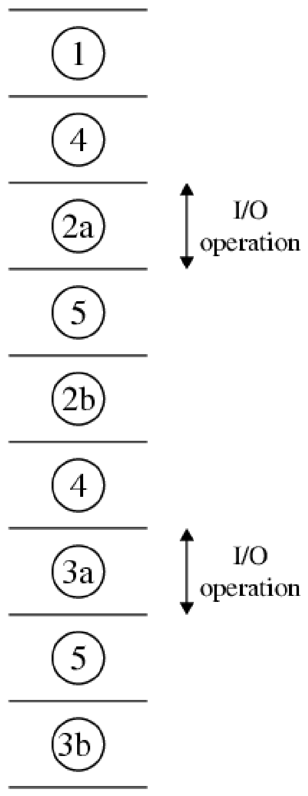
Instruction Cycle with Interrupts



Program Timing – Short I/O Wait

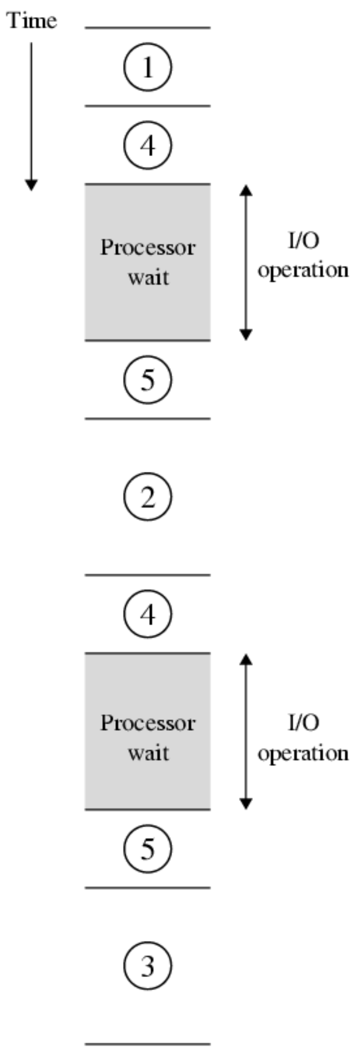


(a) Without interrupts

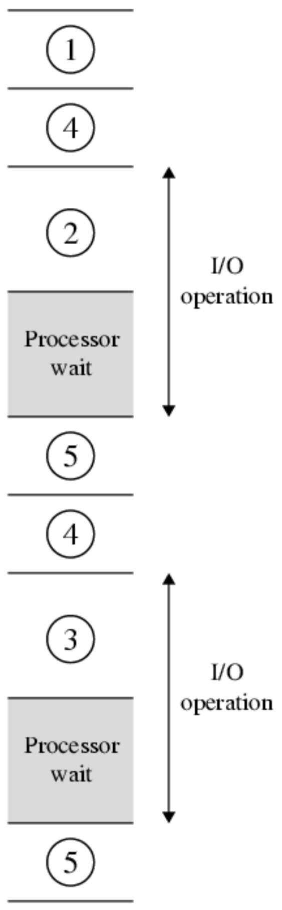


(b) With interrupts

Program Timing – Long I/O Wait

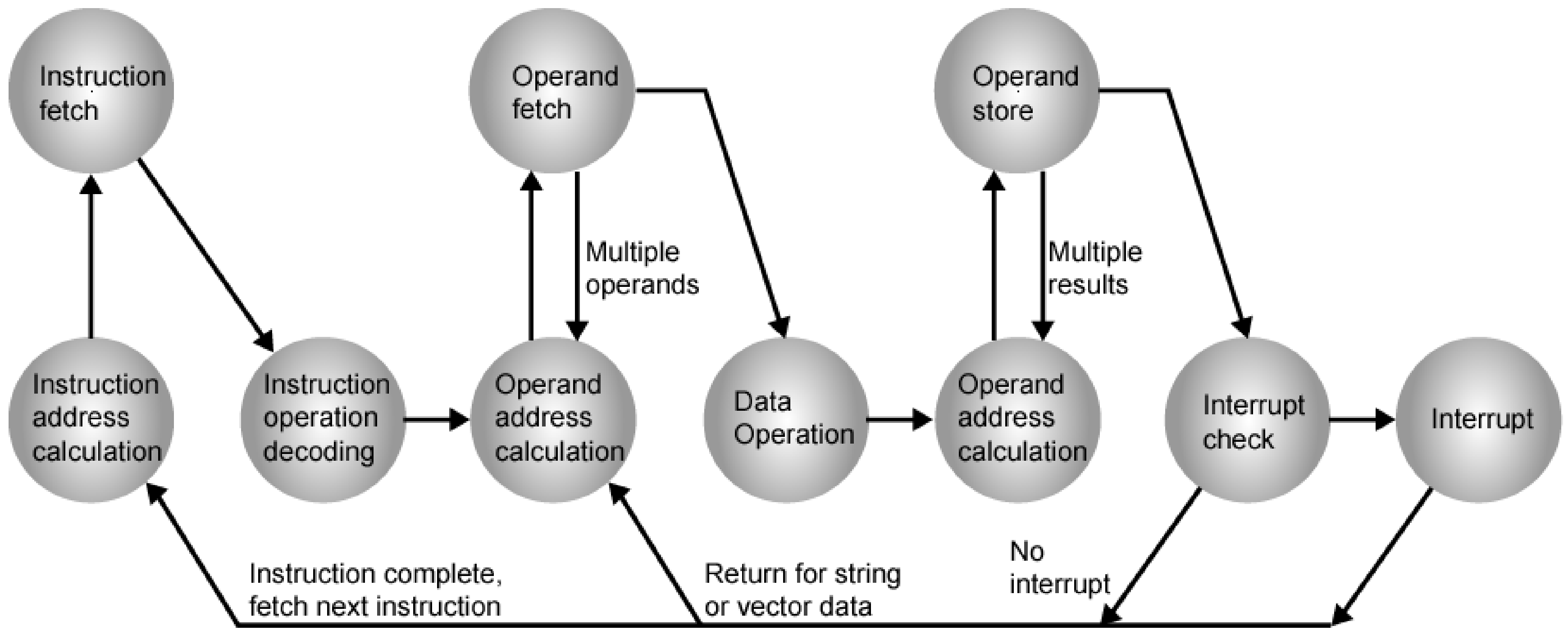


(a) Without interrupts



(b) With interrupts

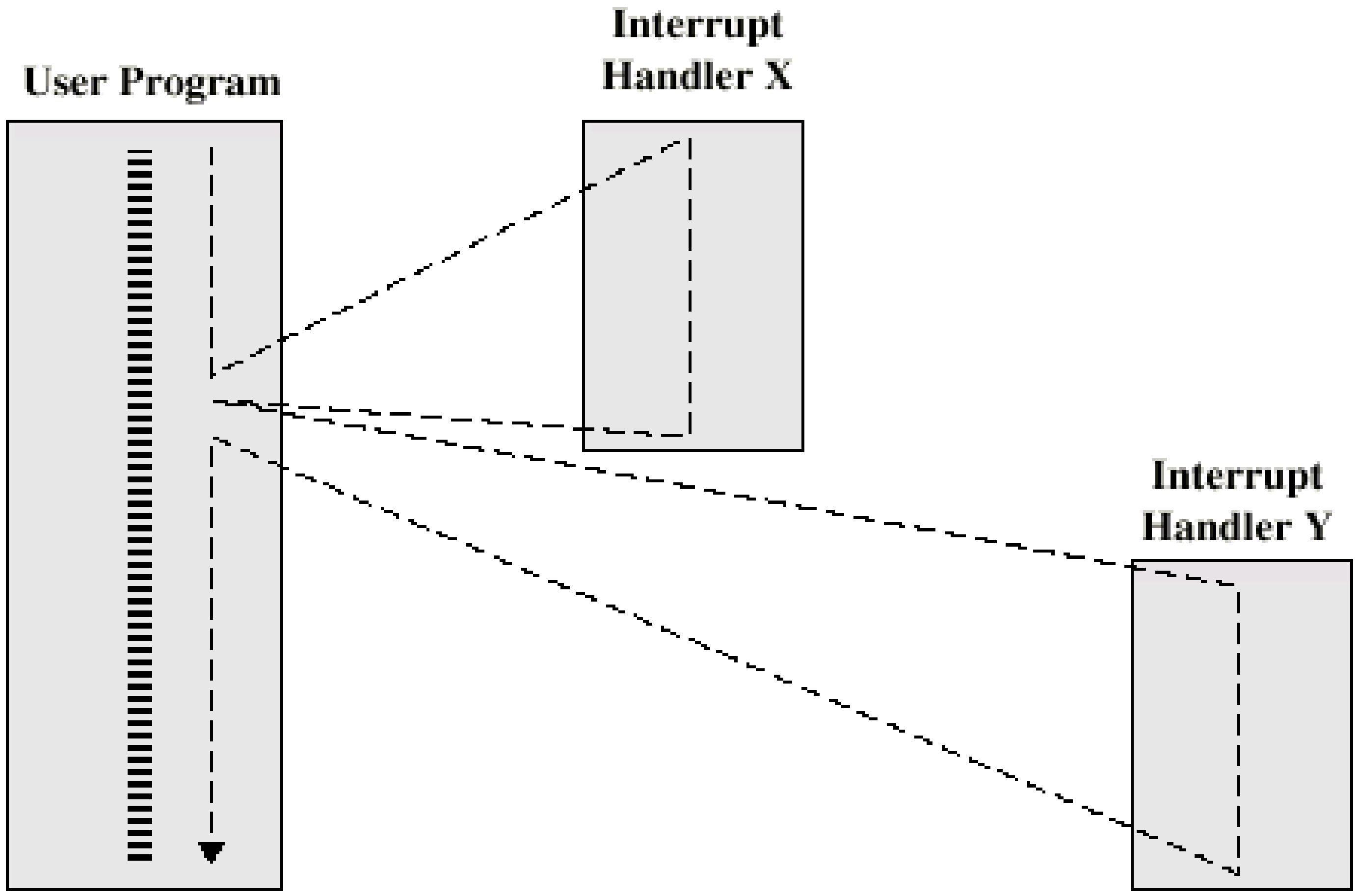
Instruction Cycle (with Interrupts) State Diagram


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Interrupt check
Interrupt
Instruction complete, fetch next instruction
Return for string or vector data
No interrupt

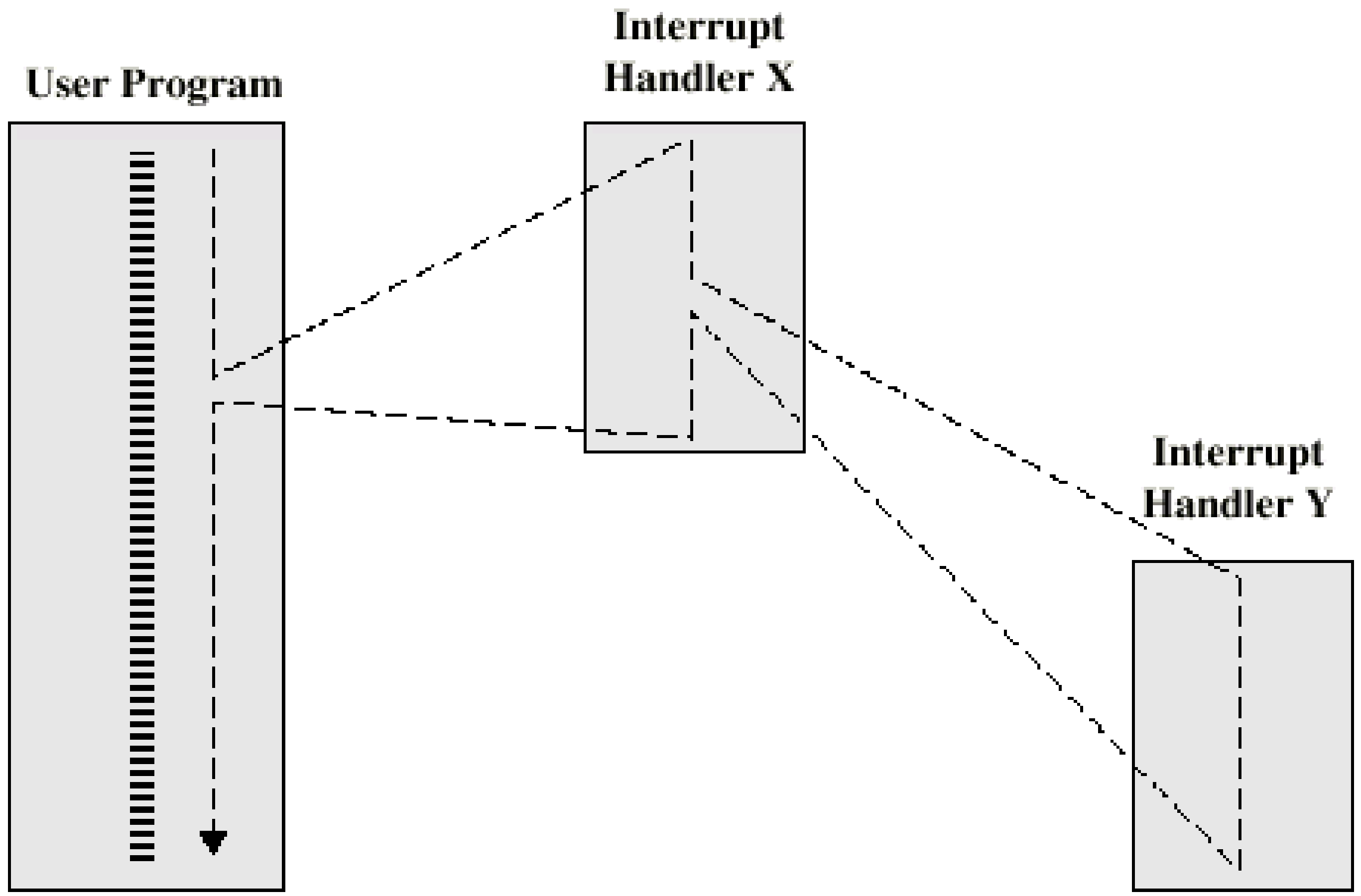
Multiple Interrupts

- Two approaches to handle multiple interrupts
  1. Interrupts are executed one after another sequentially
     - 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
  2. Based on priorities
     - Low priority interrupts can be interrupted by higher priority interrupts
     - When higher priority interrupt has been processed, processor returns to previous interrupt

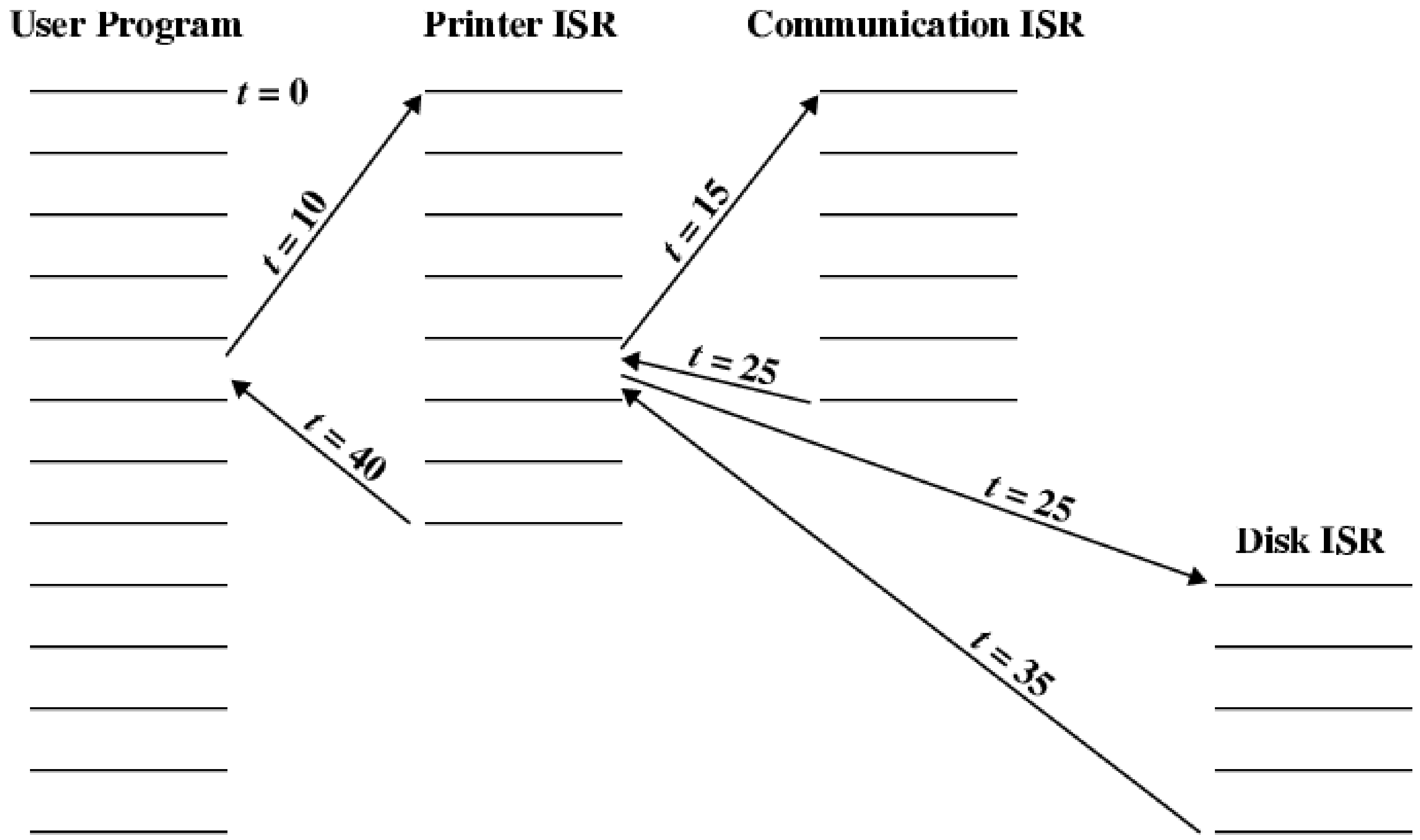
Multiple Interrupts – Sequential



Multiple Interrupts – Priority



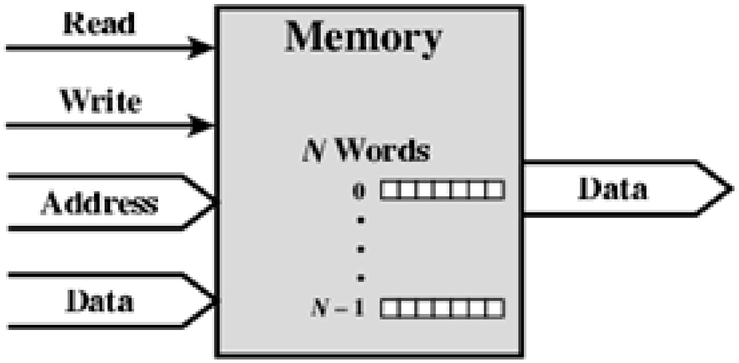
Time Sequence of Multiple Interrupts

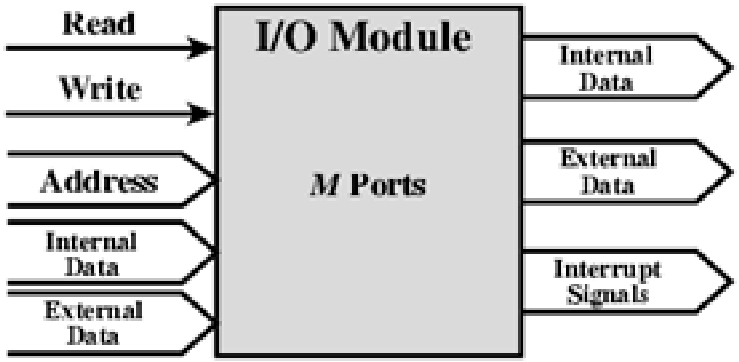


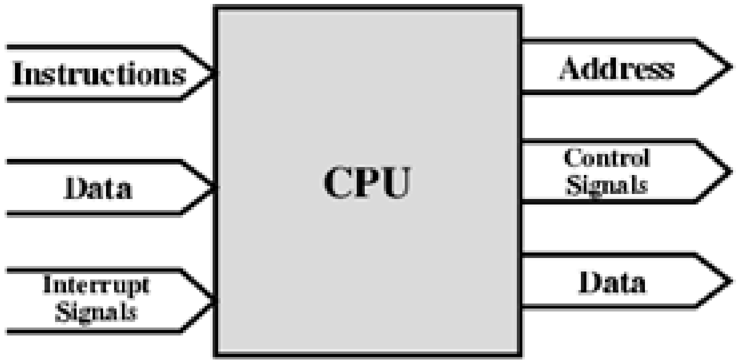
Interconnection Structure

- A computer consists of a set of components or modules
  - Processor
  - Memory
  - I/O
- Communications occur among these components
- Through communication paths
- The collection of paths connecting the various modules is called the *interconnection structure*

Computer Modules







Memory Connection

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

I/O Connection

- Similar to memory
- Always think about it from computer’s viewpoint
- Output
  - Receive data from computer
  - Send data to peripheral
- Input
  - Receive data from peripheral
  - Send data to computer

I/O Connection

- CPU sends control signals to peripherals
  - E.g., spin disk
- Peripheral receives addresses from CPU
  - E.g., port number to identify peripheral
- Peripheral sends interrupt signals (control) to CPU
  - E.g., a printer tells the CPU it finishes its print job

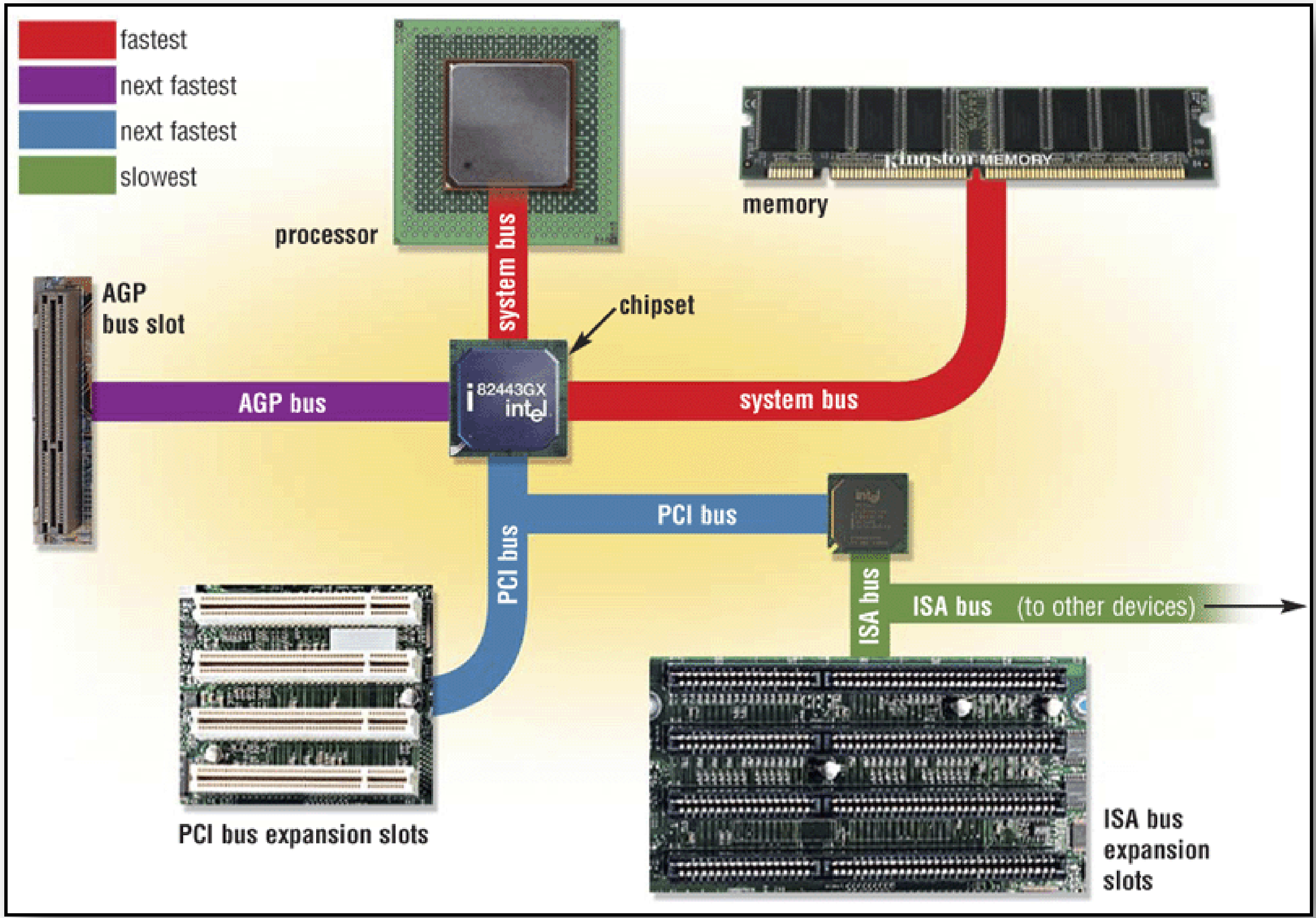
CPU Connection

- Reads instructions and data
- Writes out data (after processing)
- Sends control signals to other units
- Receives interrupts

What is a Bus?

- **Not a vehicle!**
- A *communication pathway* connecting two or more devices
- Usually broadcast
  - All devices sharing the same bus will receive the data/signals from any one of the devices
- Often grouped
  - A number of channels in one bus
  - E.g., 32-bit data bus is 32 separate single bit channels
- A bus that connects major computer components is called a *system bus*

What is a Bus?



🟥 fastest
🟪 next fastest
🟦 next fastest
🟩 slowest
processor
memory
system bus
chipset
AGP bus slot
AGP bus
82443GX intel
system bus
PCI bus
PCI bus
ISA bus
ISA bus (to other devices)
PCI bus expansion slots
ISA bus expansion slots

Data Bus

- Carries data
  - Remember that there is no difference between "data" (e.g., an integer, or a character) and "instruction" (e.g., an ADD operation, or a binary shift operation) at this level
- Width (represents how many bits can be transferred at a time) is a key determinant of performance
  - 8, 16, 32, 64-bit

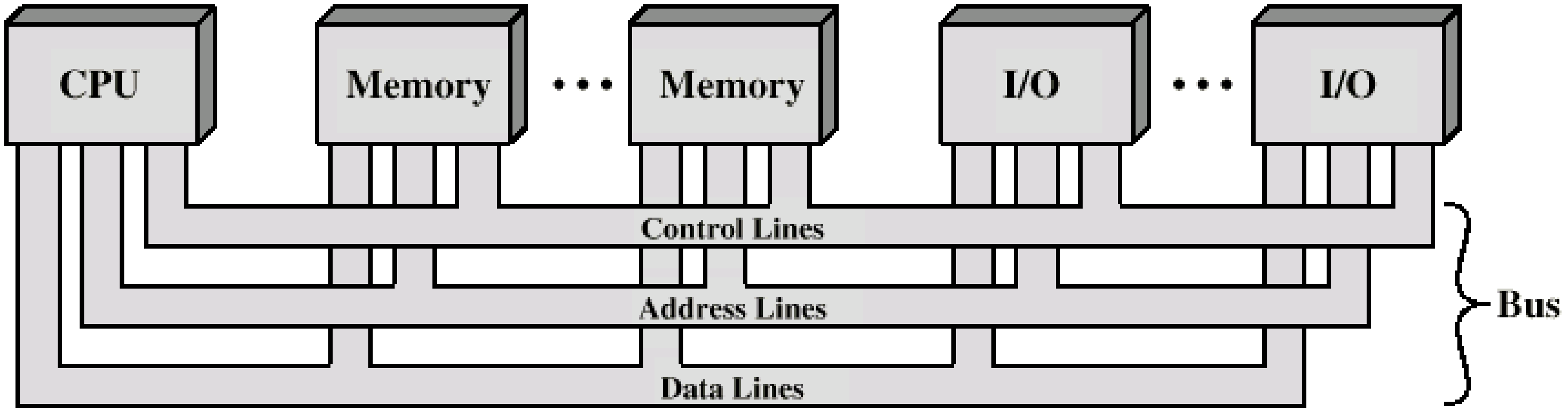
Address Bus

- Identify the source or destination of data on the data bus
  - E.g., CPU needs to read an instruction (data) from a given location in memory
- Bus width of address bus normally determines maximum memory capacity of system
  - E.g., 8080 has 16-bit address bus giving 64k (2^{16}) address space
- NOTE: Throughout this course, storage sizes/address spaces are expressed in terms of powers of 2
  - E.g., k or K refers to 1024 (2^{10})

Control Bus

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

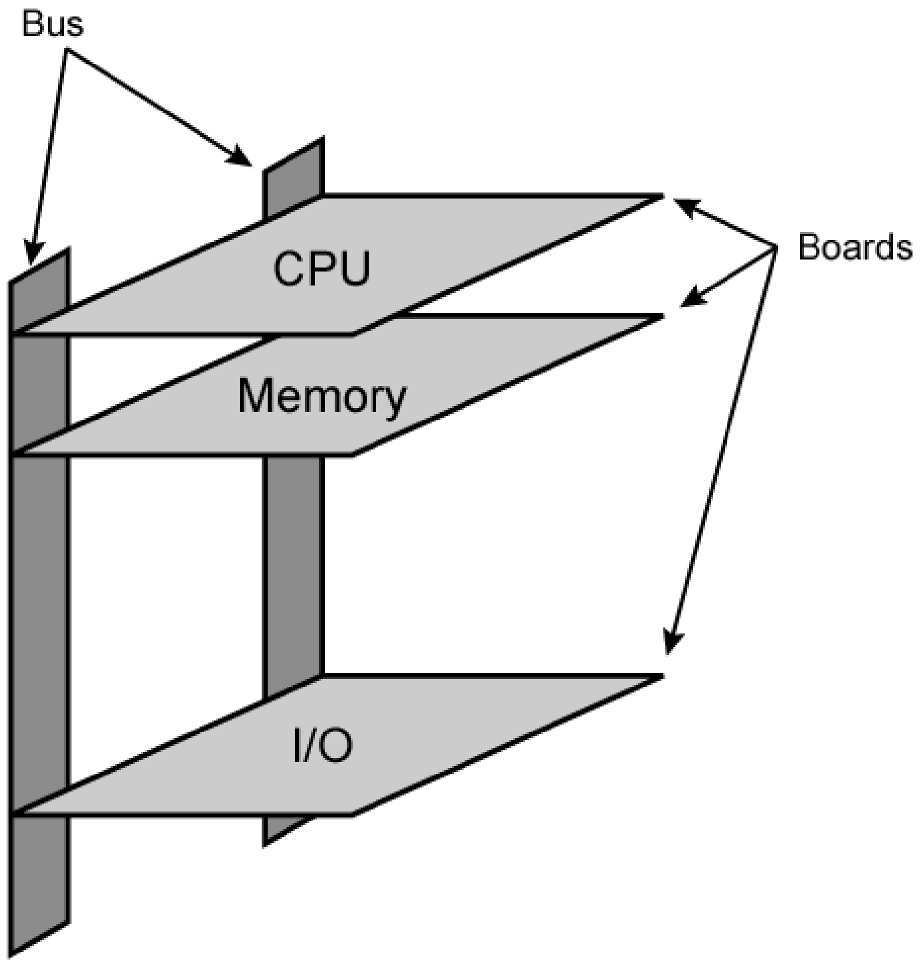
Bus Interconnection Scheme



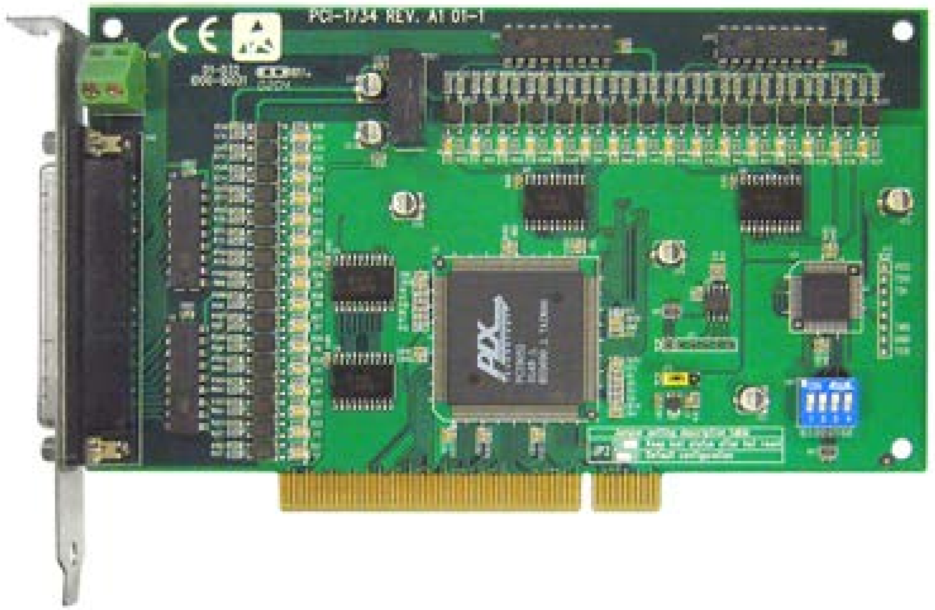
What do buses look like?

- A number of parallel electrical conductors
- Metal line etched in card or circuit
- Example depiction in next slide
  - Two vertical columns of conductors
  - Slots extending out horizontally to support printed-circuit boards
  - Convenient for scalability (a computer can extend its functions)
  - Damaged board can be replaced

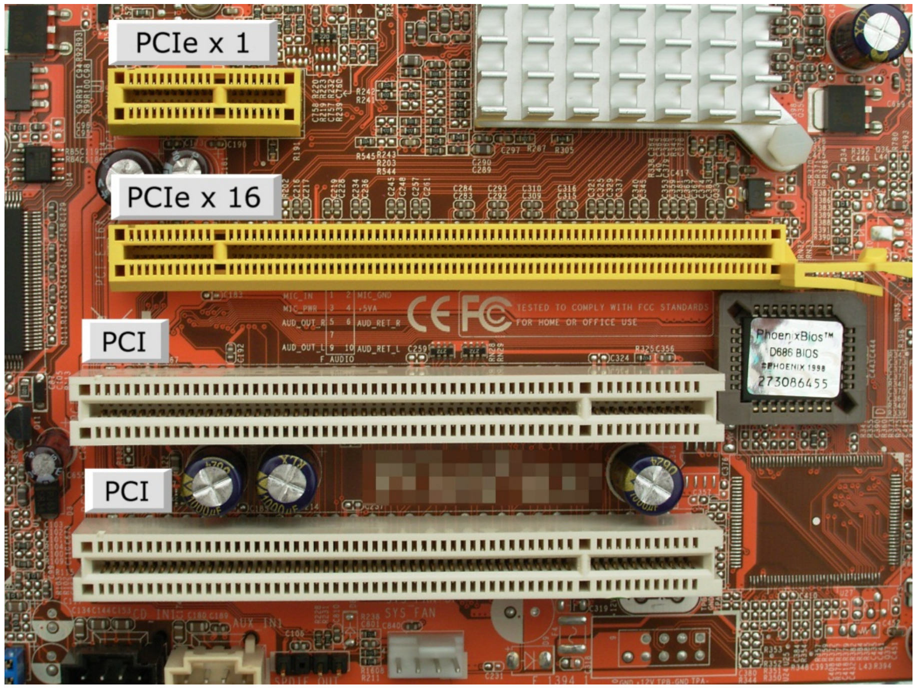
Physical Realization of Bus Architecture



Bus Example: PCI Family



http://downloadt.advantech.com/ProductFile/Downloadfile2/1-CUB1ZW/PCI-1734_03_B.jpg

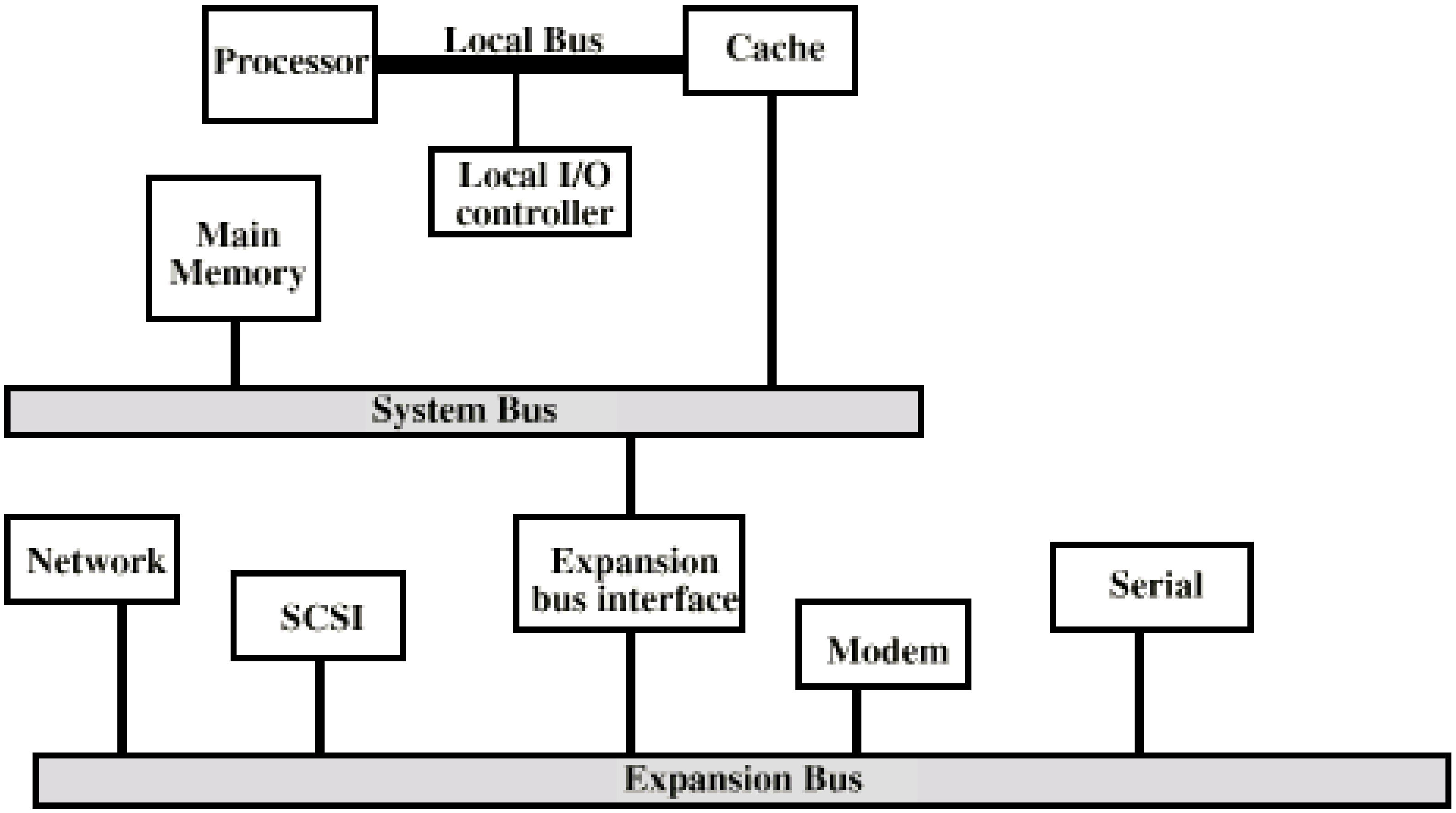


http://upload.wikimedia.org/wikipedia/commons/0/0c/PCI_und_PCIe_Slots.jpg

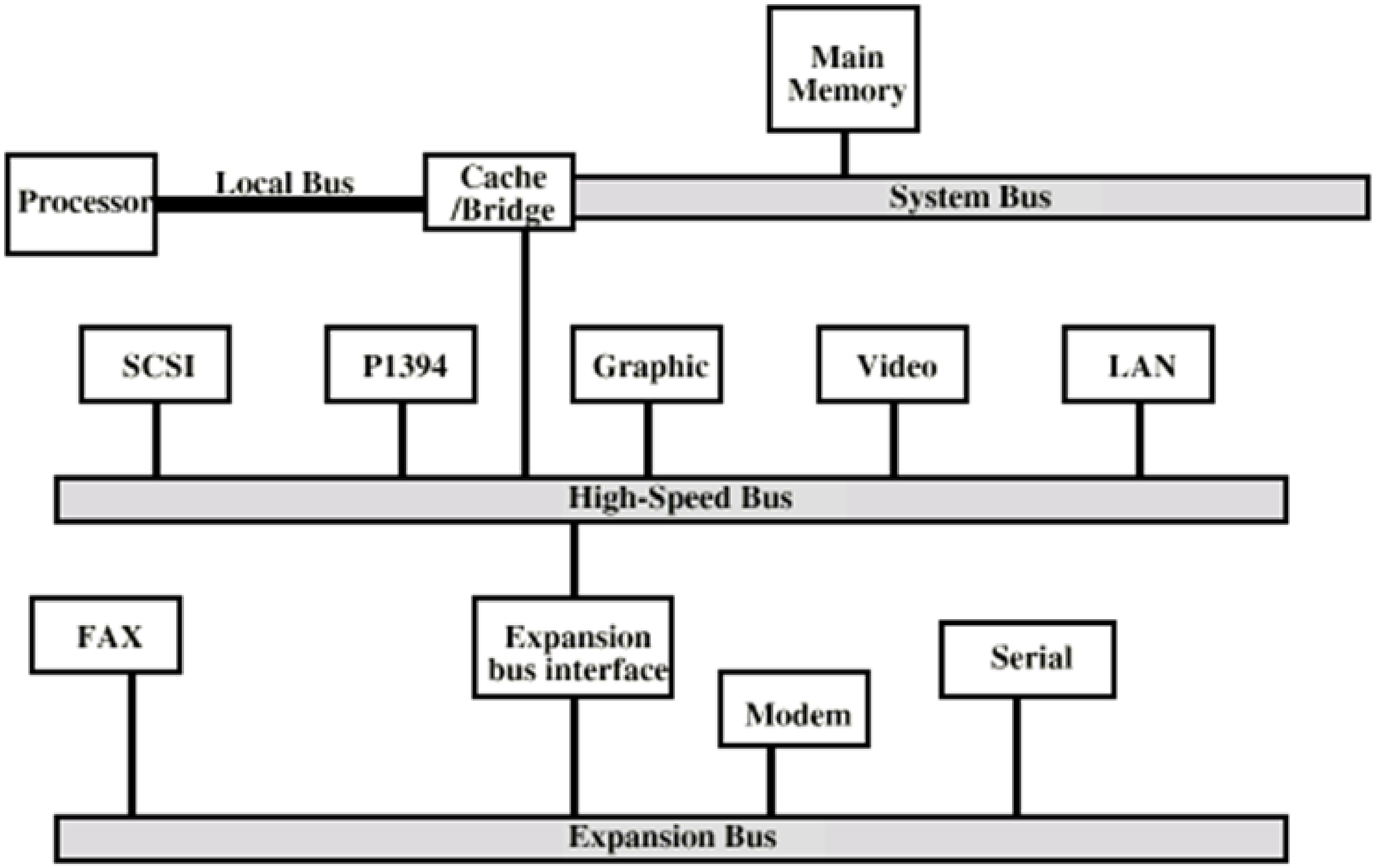
Single Bus Problems

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

Traditional Bus Architecture



High Performance Bus



Bus Types

- Dedicated
  - Separate data & address lines
- Multiplexed
  - Shared lines
  - An address is first placed on the bus to address the module
  - Same bus is used for subsequent data transfer
  - Advantage - fewer lines
  - Disadvantages
    - More complex circuitry
    - Address and data cannot be transferred in parallel

Bus Arbitration

- Only one unit at a time can successfully transmit over the bus. Why?
- Centralized
  - A single hardware device – bus controller/arbiter
  - Allocates time on the bus
- Distributed
  - Each module contains access control logic
- Ultimate aim is to designate a single device to initiate data transfer

Timing

- Co-ordination of events on bus
- Synchronous
  - The bus includes a clock line
  - A single 1-0 is a clock/bus cycle (a time slot)
  - All devices can read clock line and start at the beginning of a clock cycle
  - Usually a single cycle for an event
- Asynchronous
  - Occurrence of one event on a bus follows and depends on the occurrence of a previous event

PCI Bus

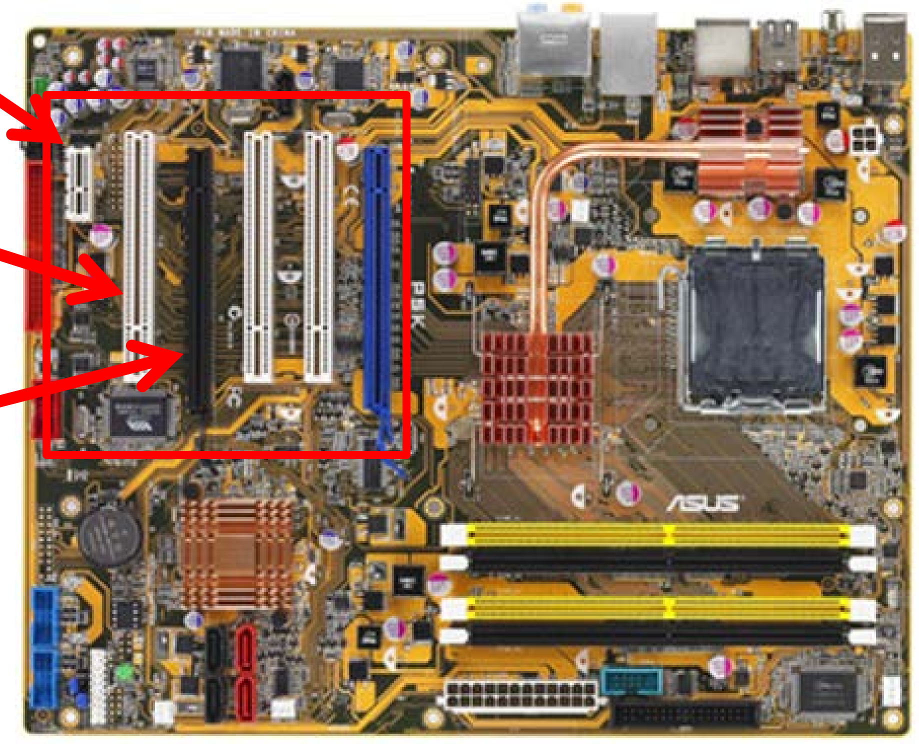
- Peripheral Component Interconnection
- Created by Intel
- 32-bit bus width (133 MB/s)
  - Synchronous transfer
- Superseded by *PCI Express* (2004), PCI Express 2.0 (2007) and PCI Express 3.0 (2010)
- *PCI-E x16* has a transfer rate of 4 GB/s
  - x16 means sixteen lanes, with each lane of 250 MB/s

PCI Slots on Motherboard

PCI-E x1 slot

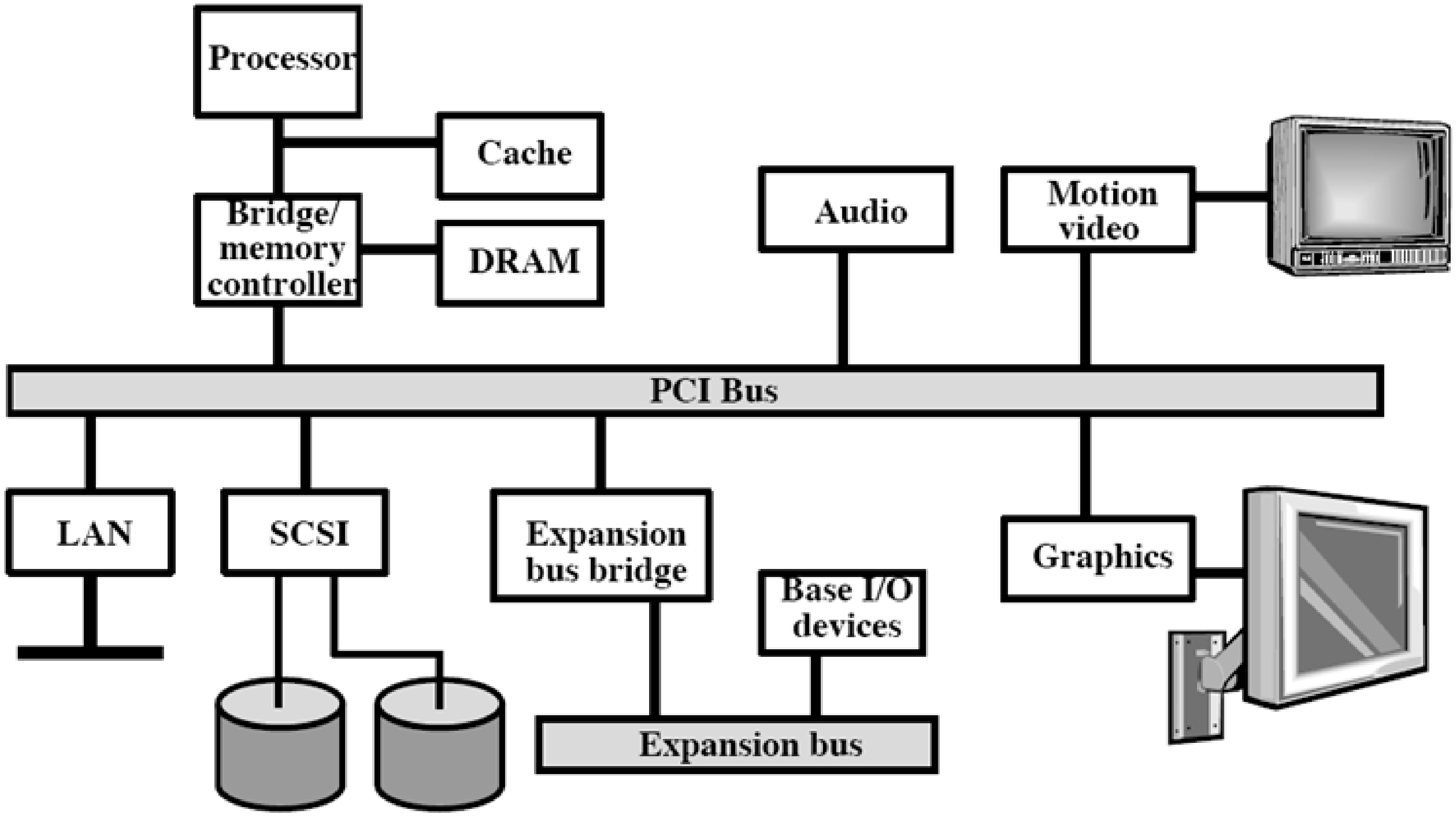
PCI slot

PCI-E x16 slot



http://www.asus.com/Motherboards/Intel_Socket_775/P5K/

Typical Desktop System


Processor
Cache
Bridge/ memory controller
DRAM
Audio
Motion video
PCI Bus
LAN
SCSI
Expansion bus bridge
Base I/O devices
Graphics
Expansion bus