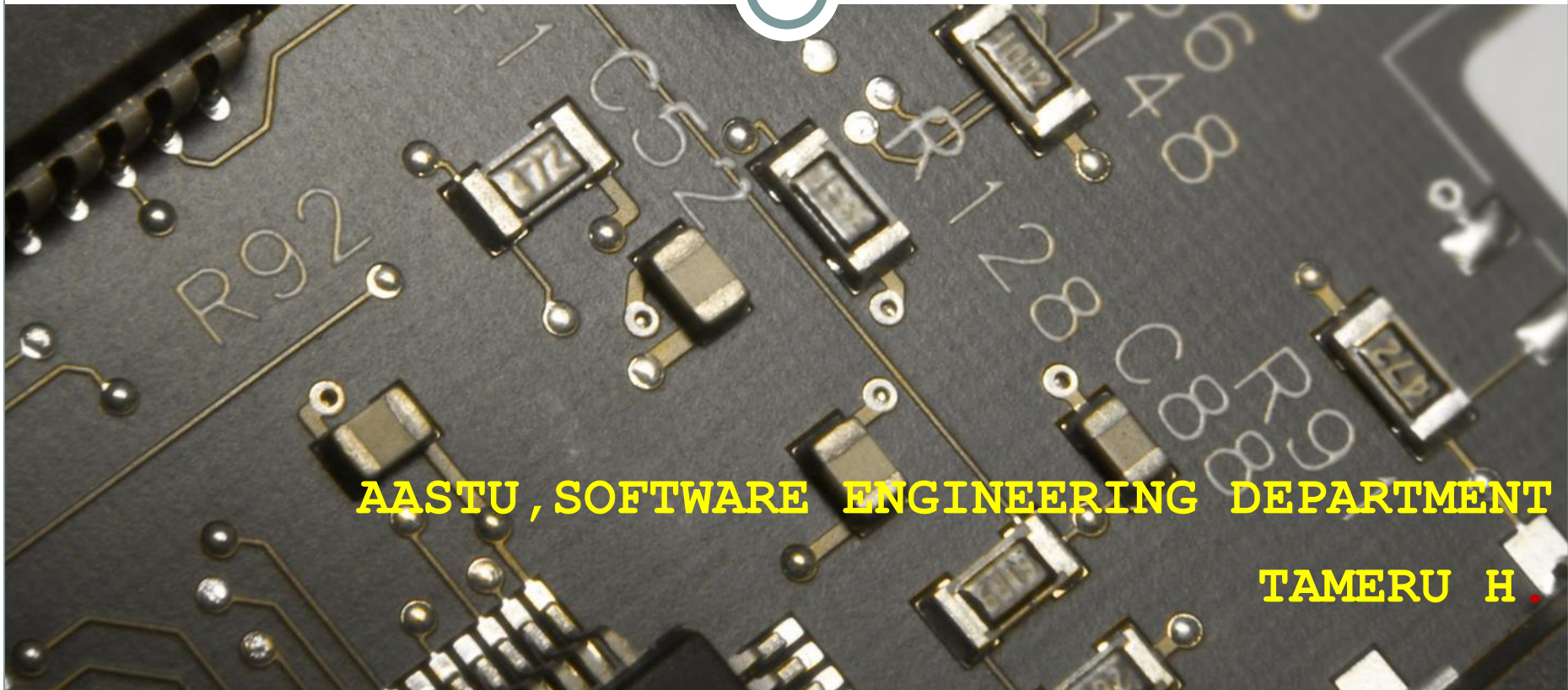


Chapter Three

Part-2



AASTU, SOFTWARE ENGINEERING DEPARTMENT

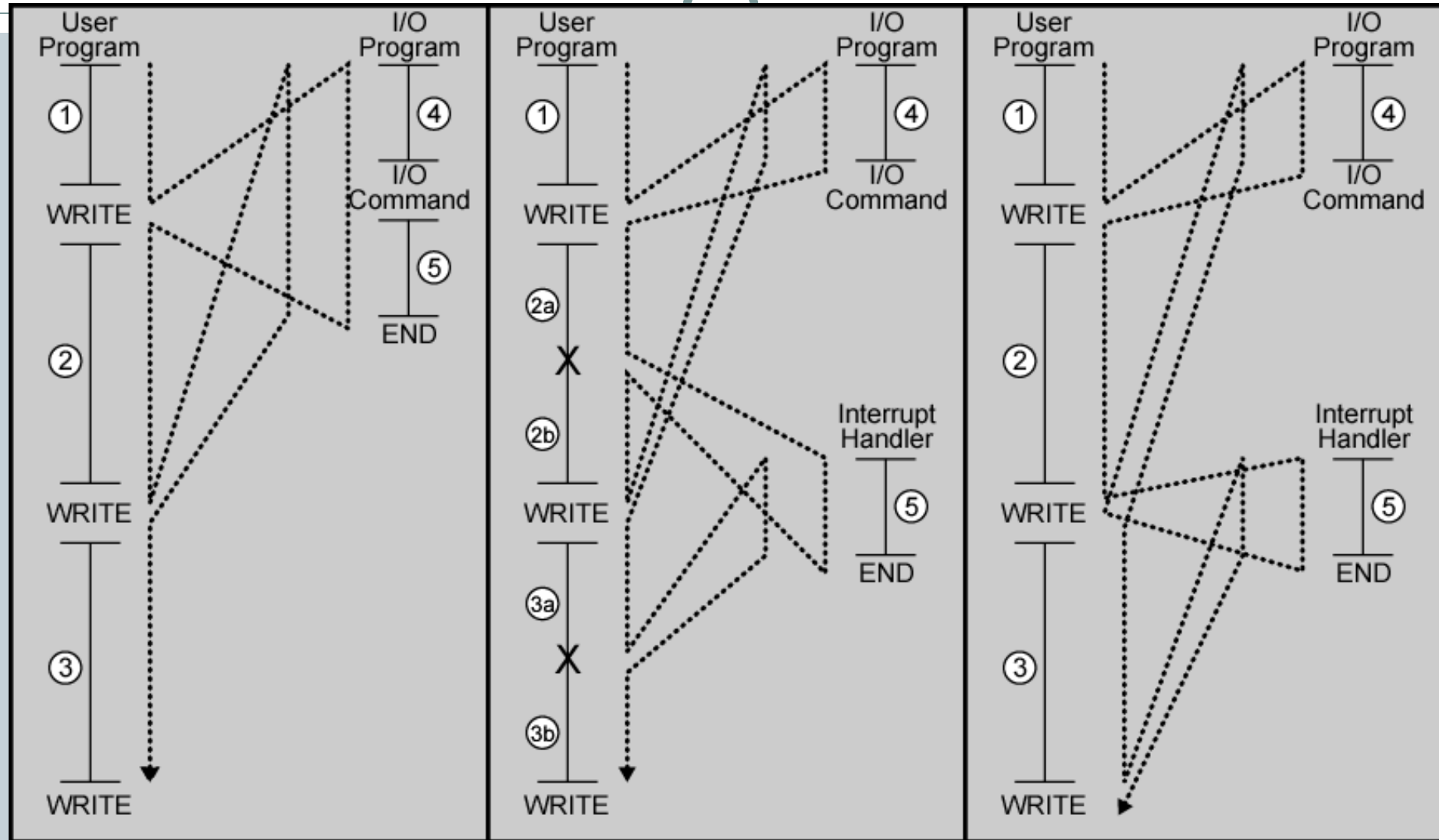
TAMERU H.

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

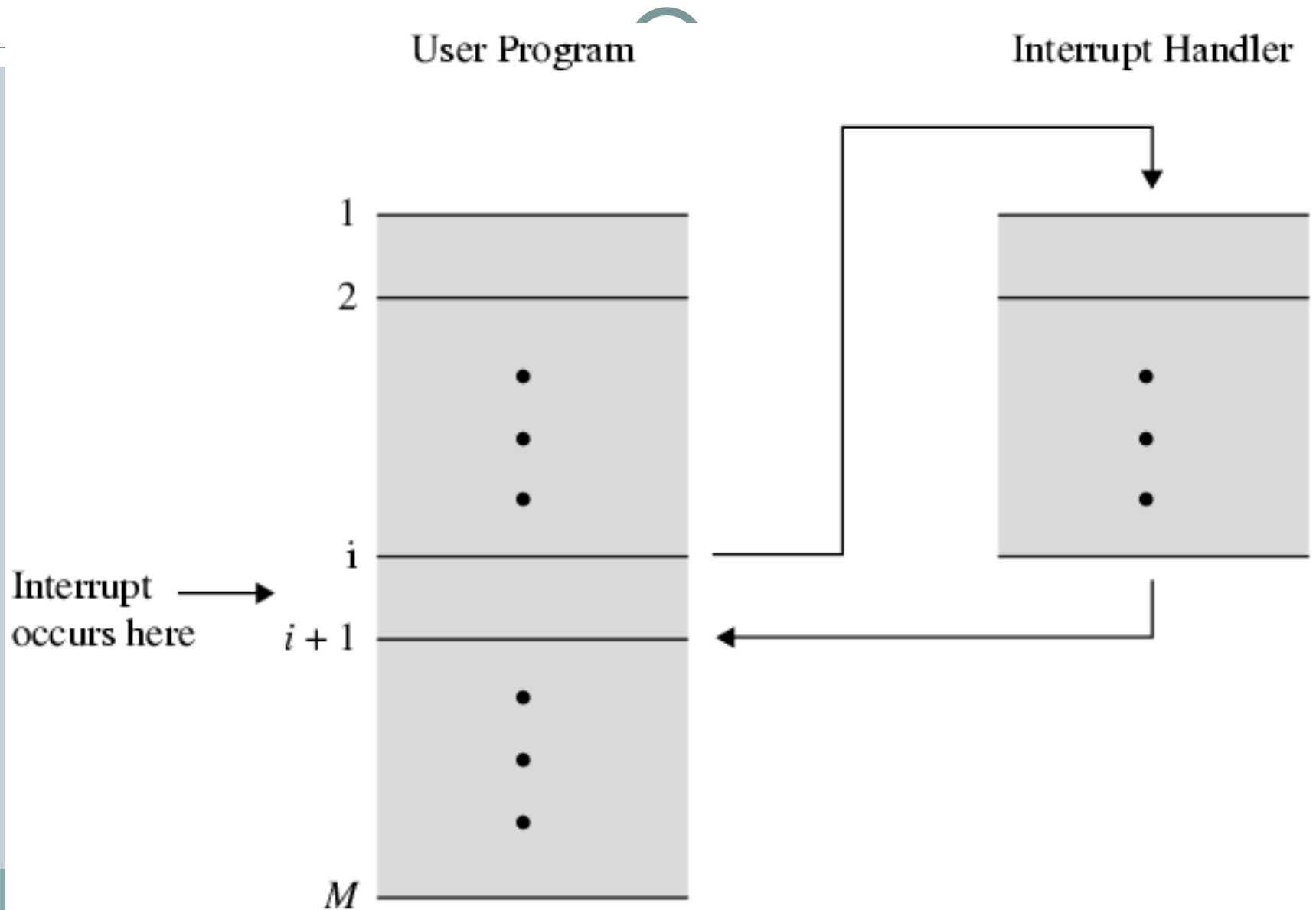
(c) Interrupts; long I/O wait

Interrupt Cycle

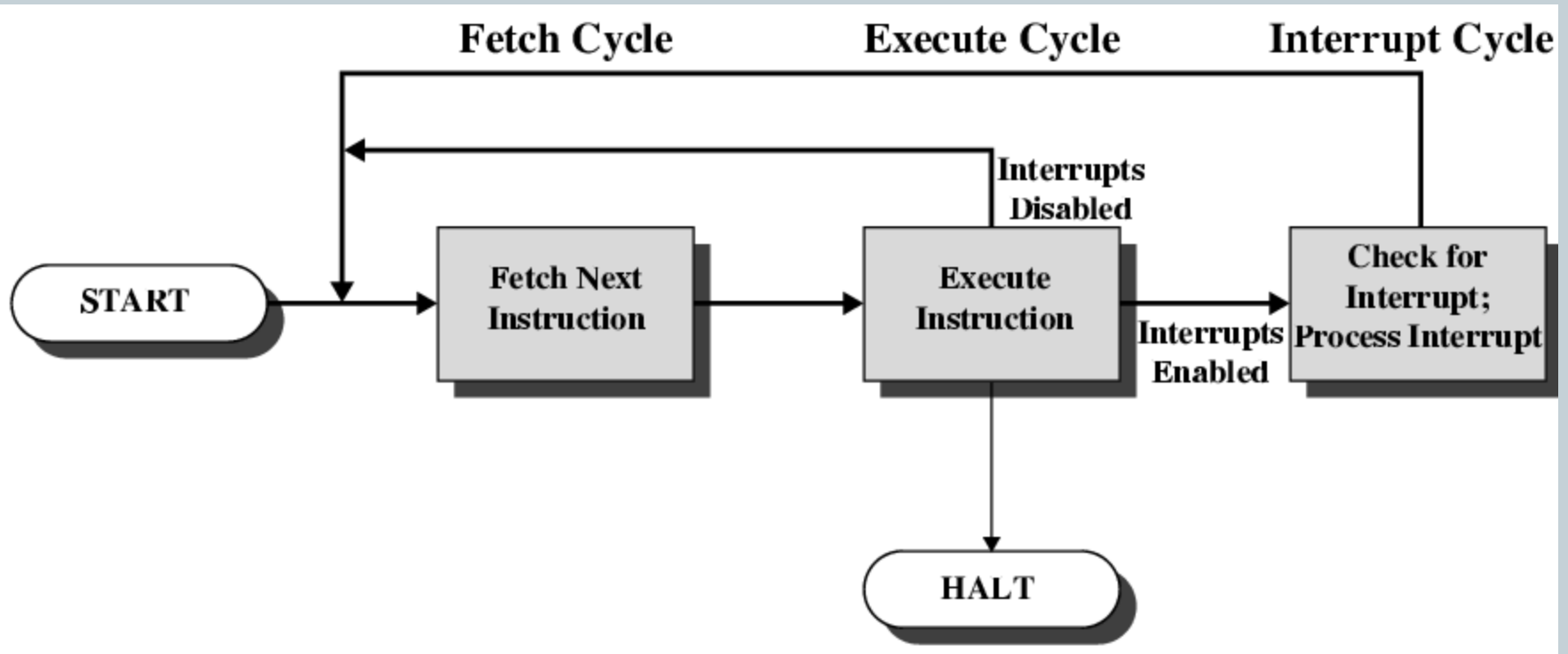


- Added to instruction cycle
- Processor checks for interrupt
 - Indicated by an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending:
 - Suspend execution of current program
 - Save context
 - Set PC to start address of interrupt handler routine
 - Process interrupt
 - Restore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

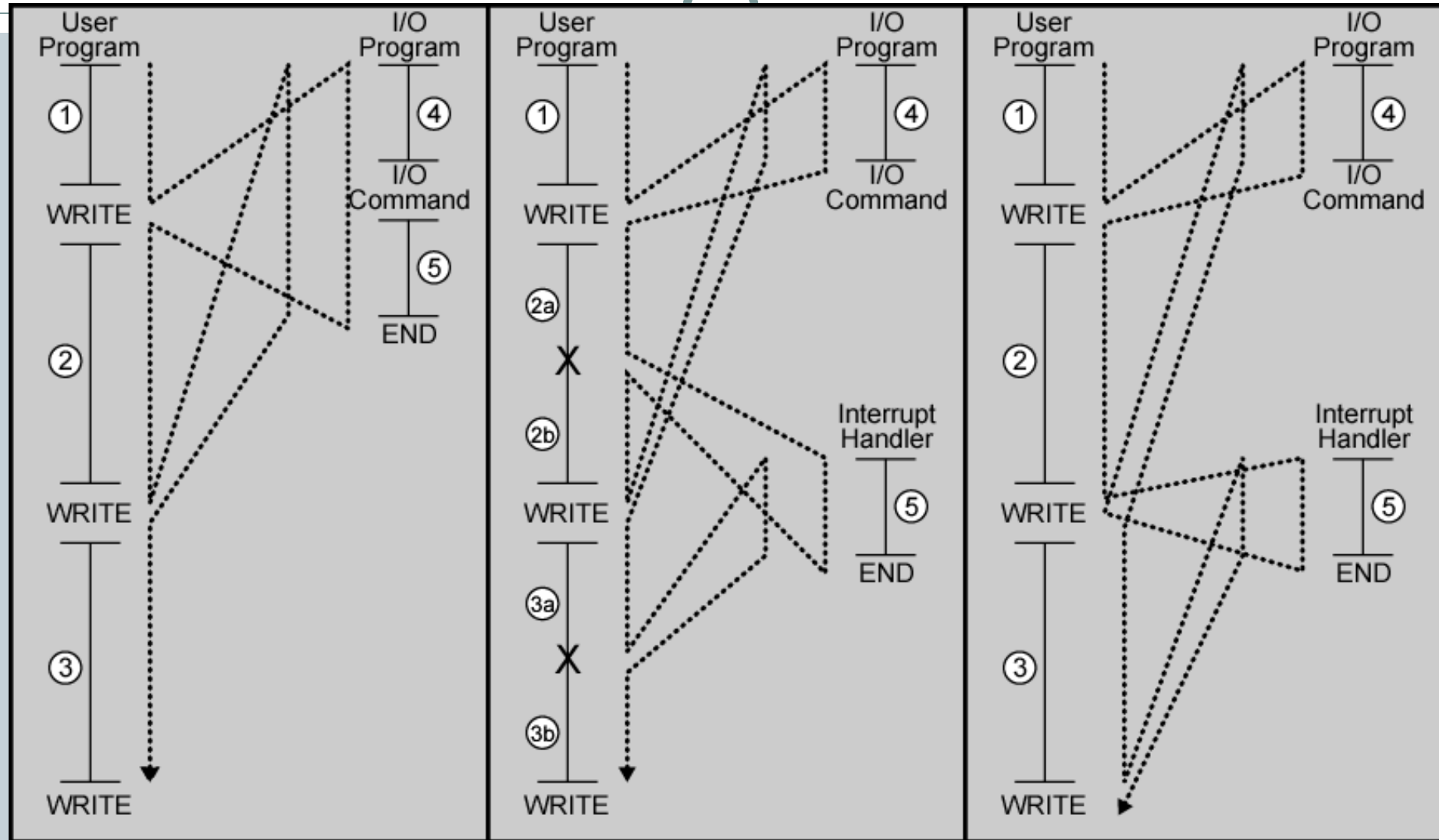
Transfer of Control via Interrupts



Instruction Cycle with Interrupts



Program Flow Control

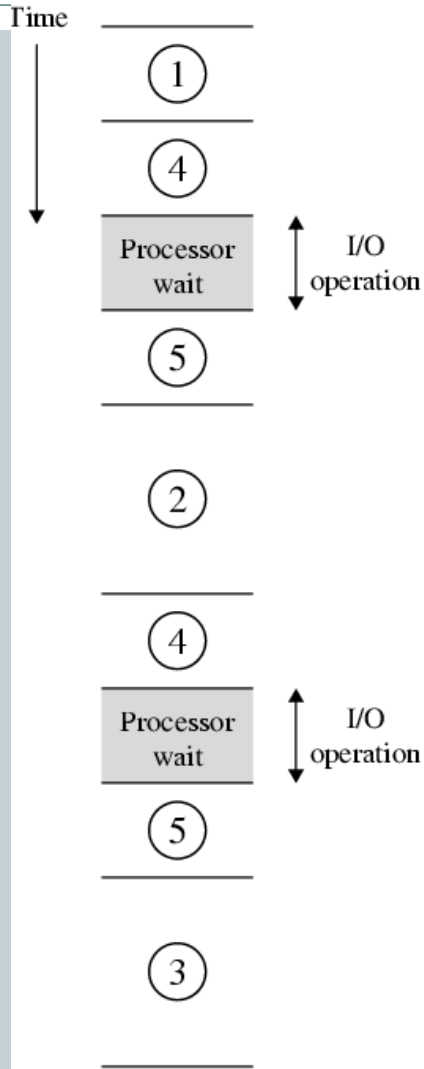


(a) No interrupts

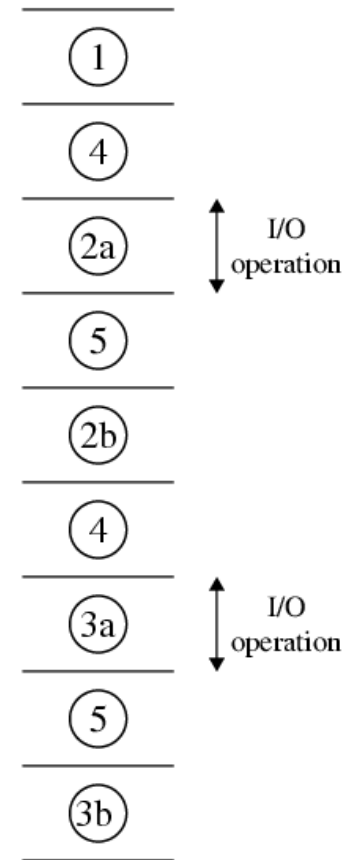
(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait

Program Timing Short I/O Wait

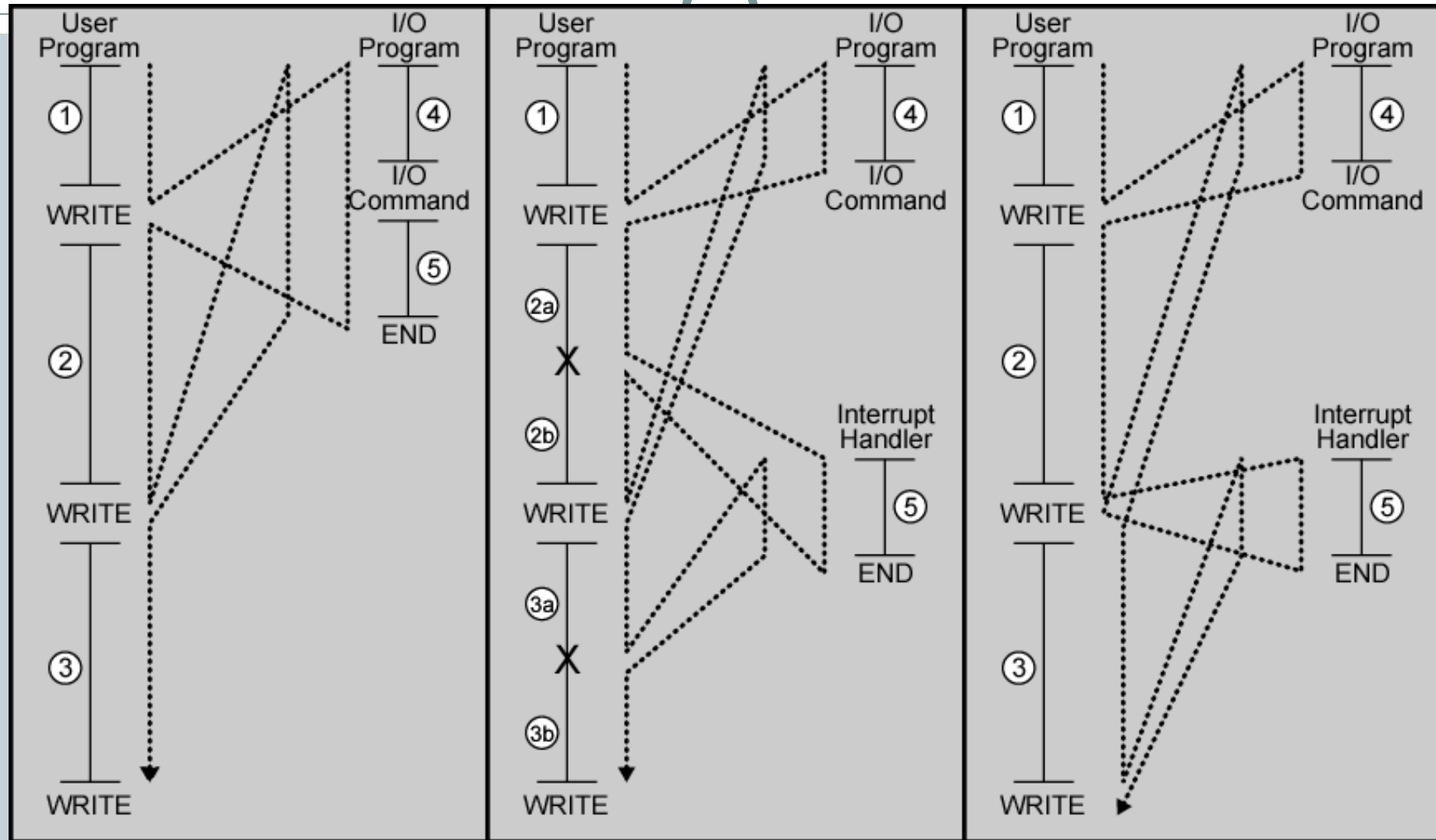


(a) Without interrupts



(b) With interrupts

Program Flow Control



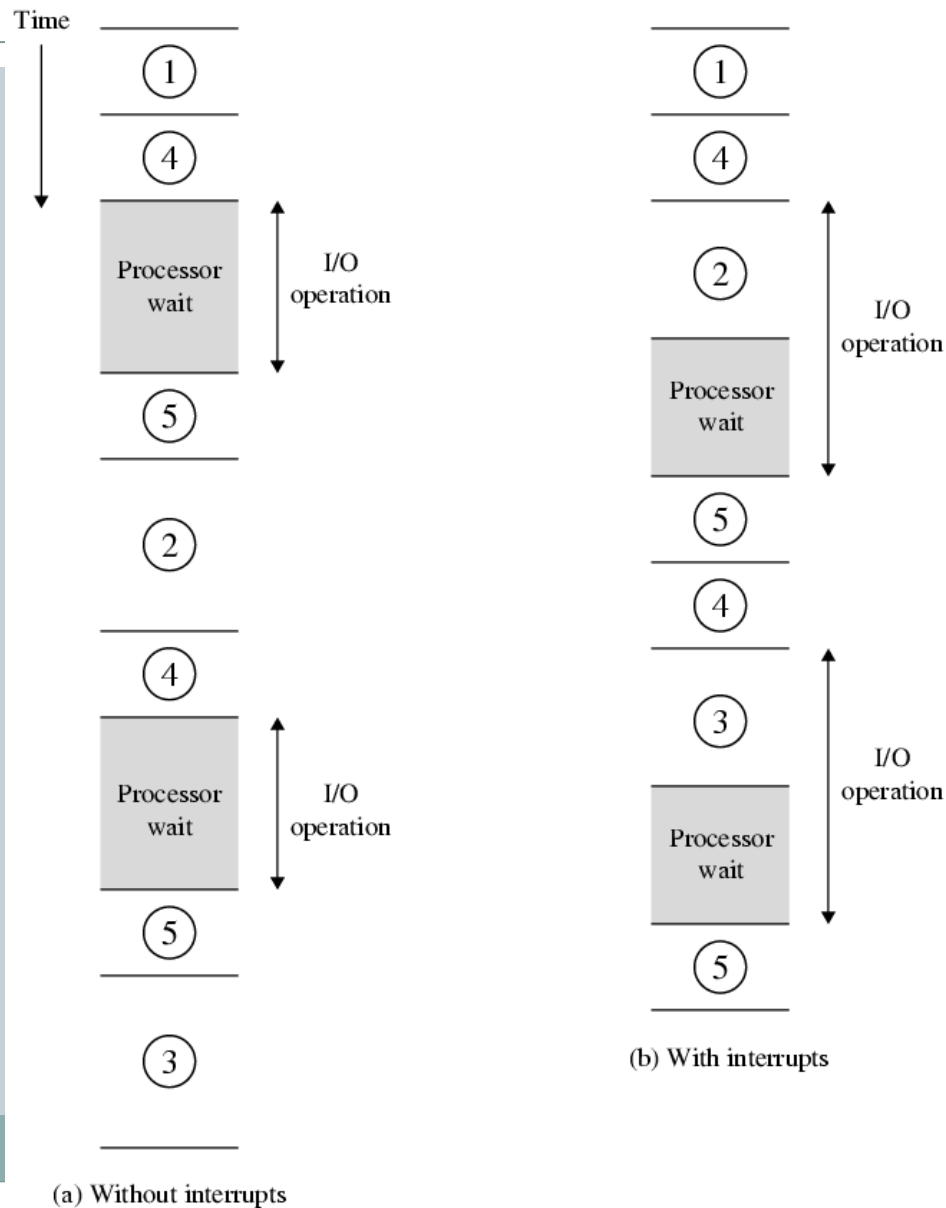
(a) No interrupts

(b) Interrupts; short I/O wait

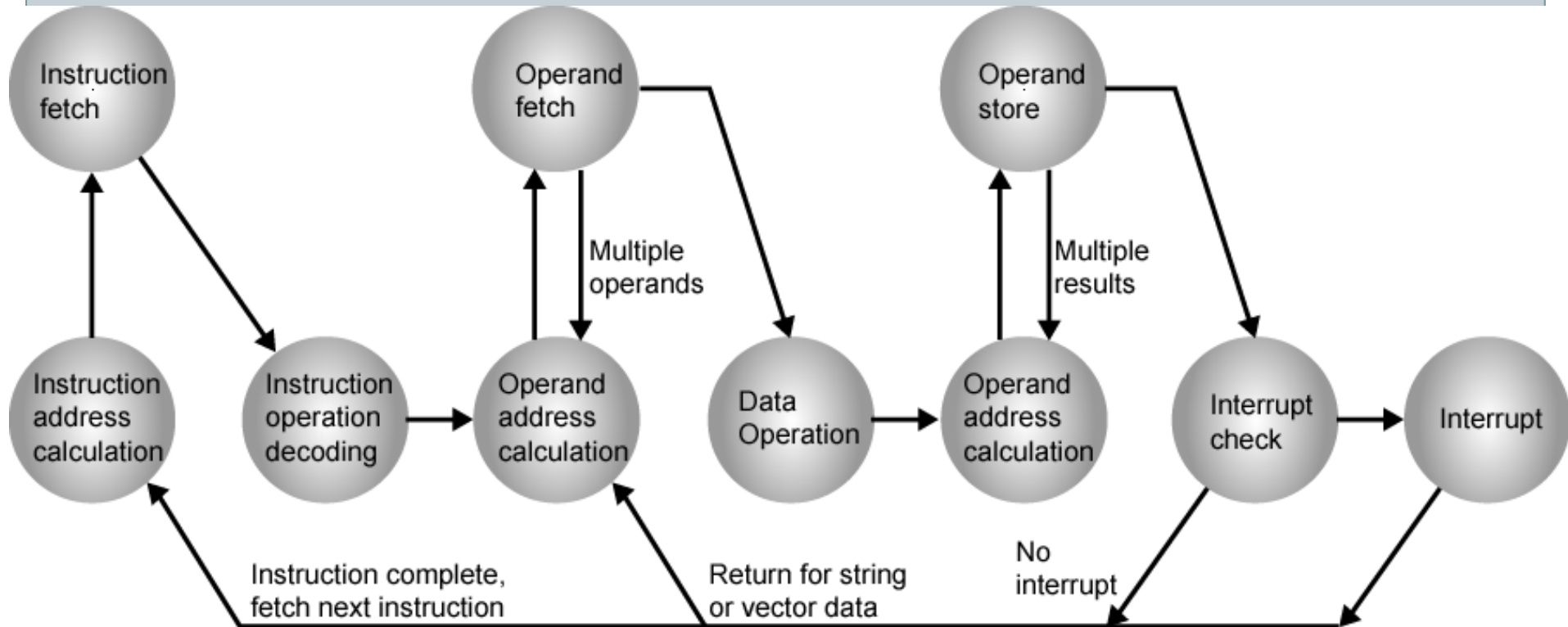
(c) Interrupts; long I/O wait

Program Timing

Long I/O Wait



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

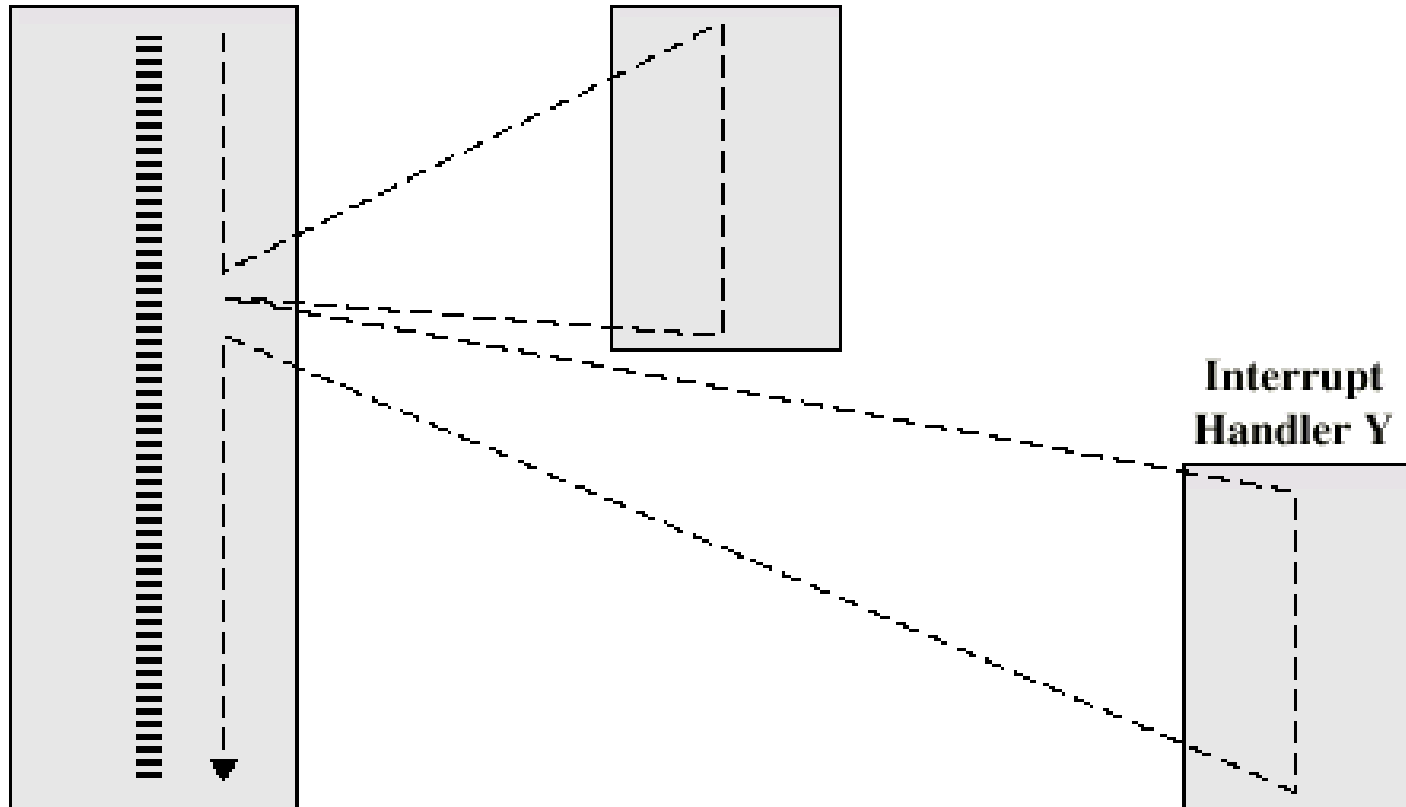
Multiple Interrupts - Sequential



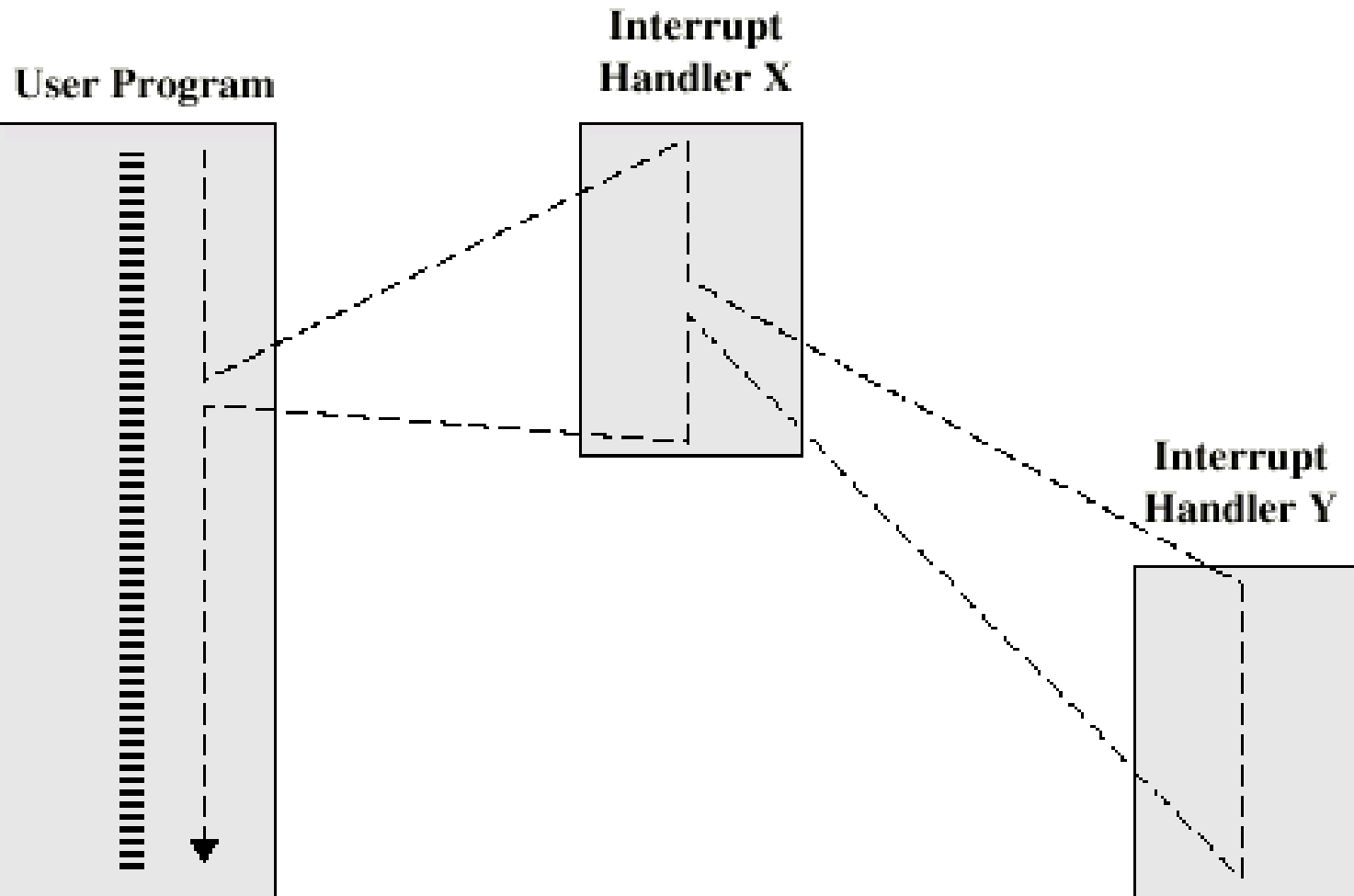
User Program

**Interrupt
Handler X**

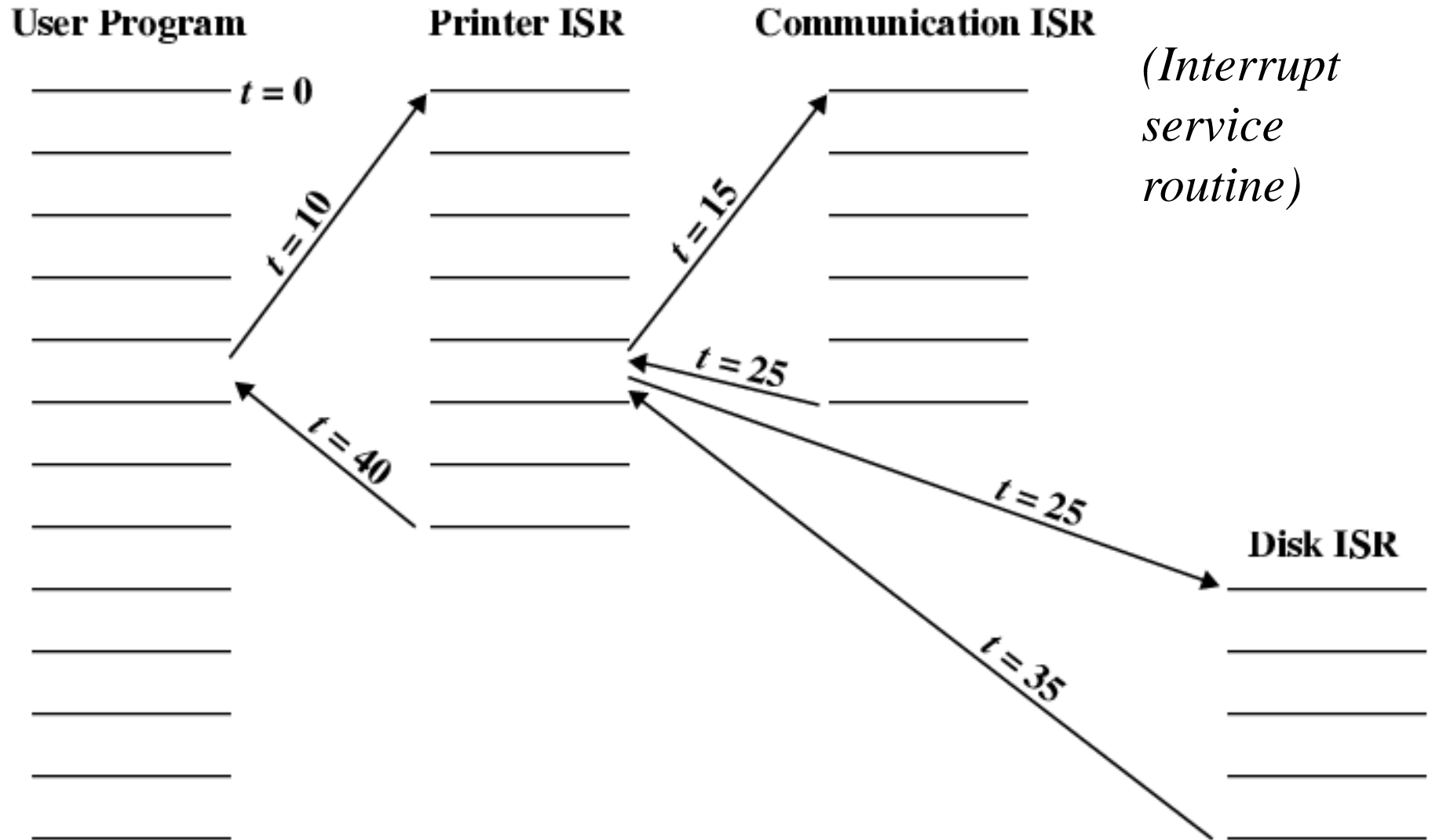
**Interrupt
Handler Y**



Multiple Interrupts – Nested



Time Sequence of Multiple Interrupts

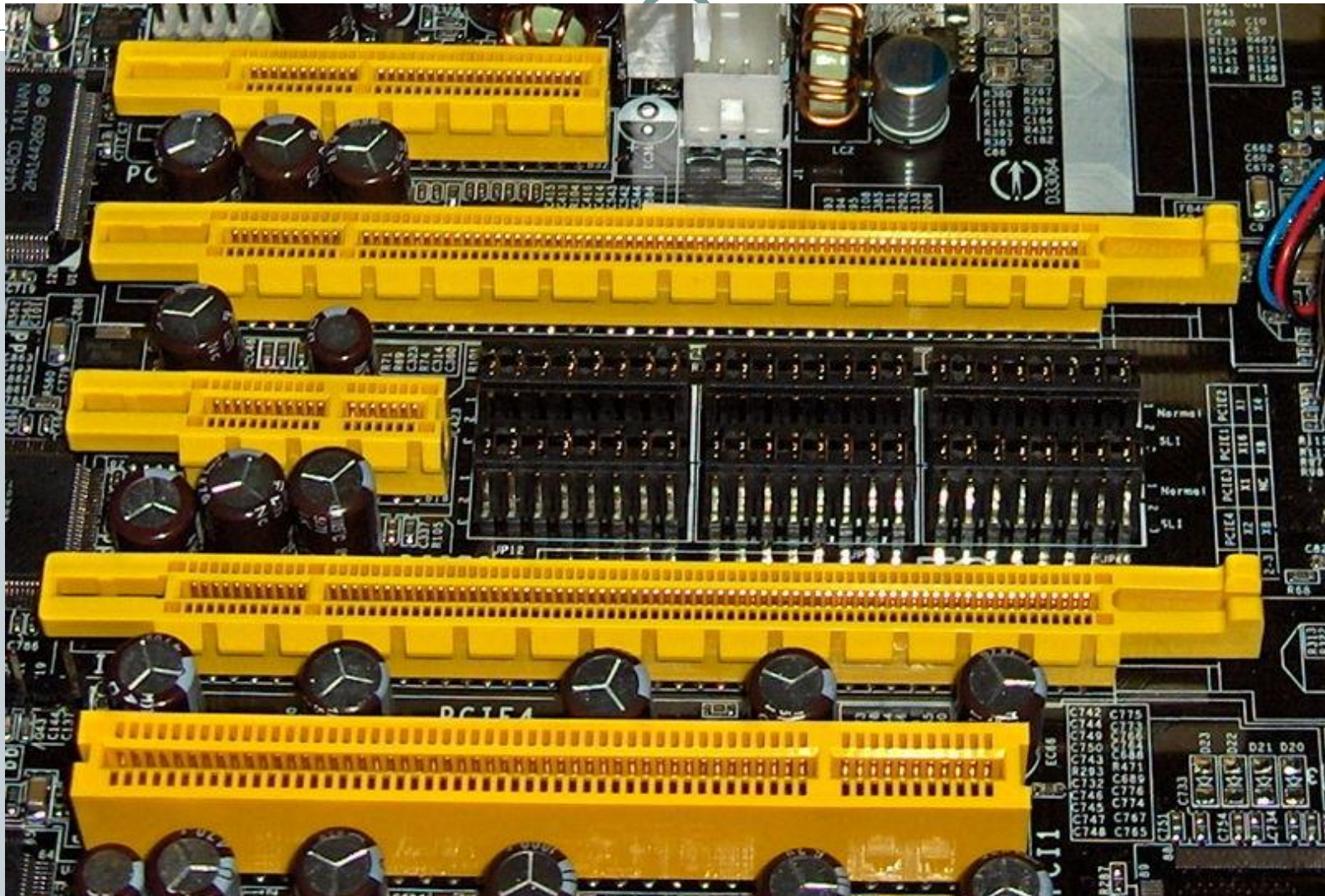


Connecting

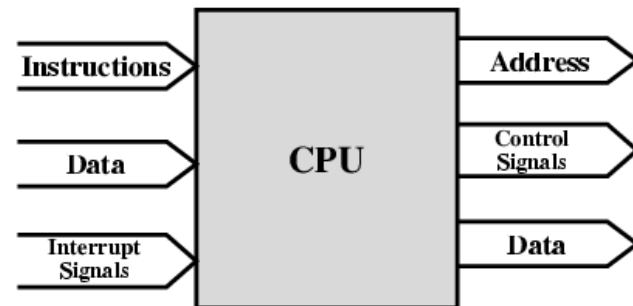
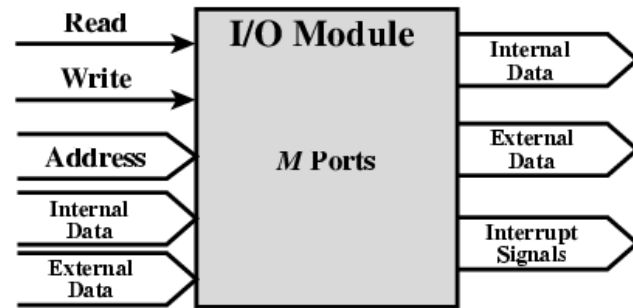
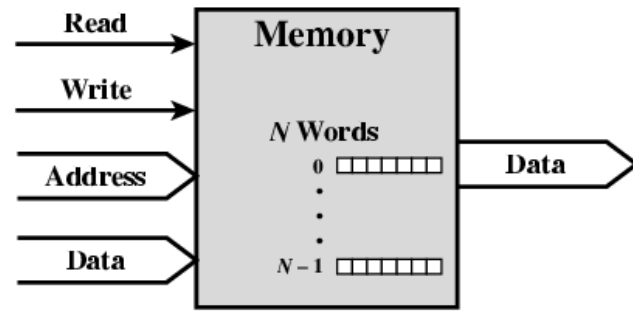


- All the units must be connected
- Different type of connection for different type of unit
 - Memory
 - Input/Output
 - CPU

PCI Express bus card slots (from top to bottom: x4, x16, x1 and x16), compared to a traditional 32-bit PCI bus card slot (bottom). (PCI = Peripheral Component Interconnect)



Computer Modules

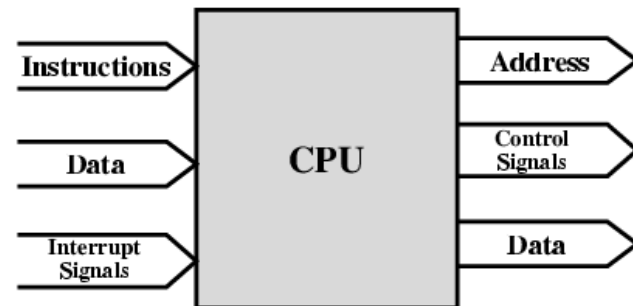
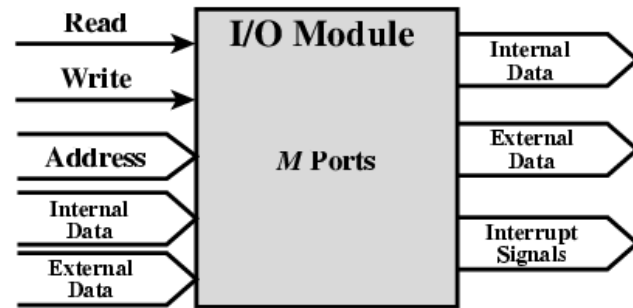
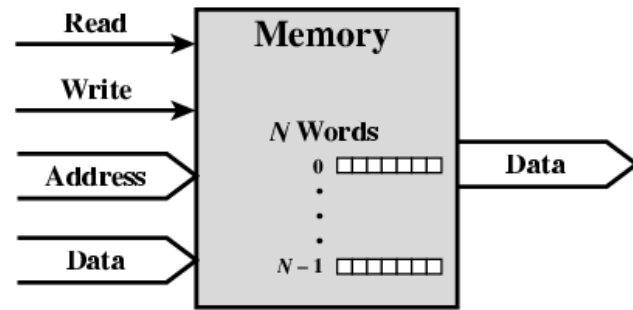


Memory Connection



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

Computer Modules



Input/Output Connection(1)



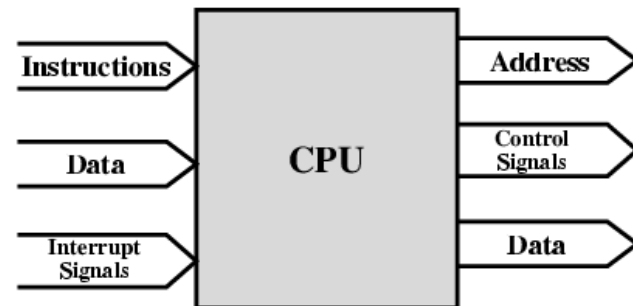
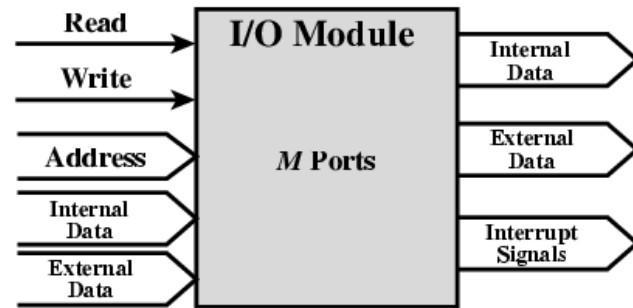
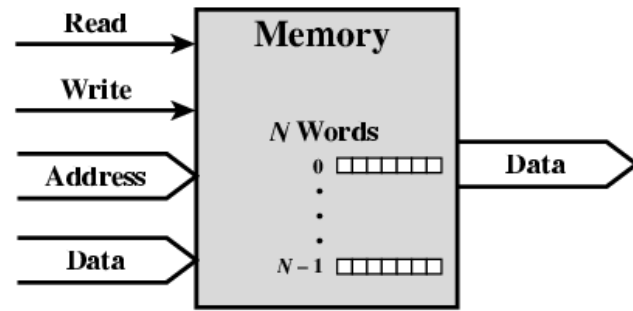
- Similar to memory from computer's viewpoint
- Output
 - Receive data from computer
 - Send data to peripheral
- Input
 - Receive data from peripheral
 - Send data to computer

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)

Computer Modules



CPU Connection



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

Buses



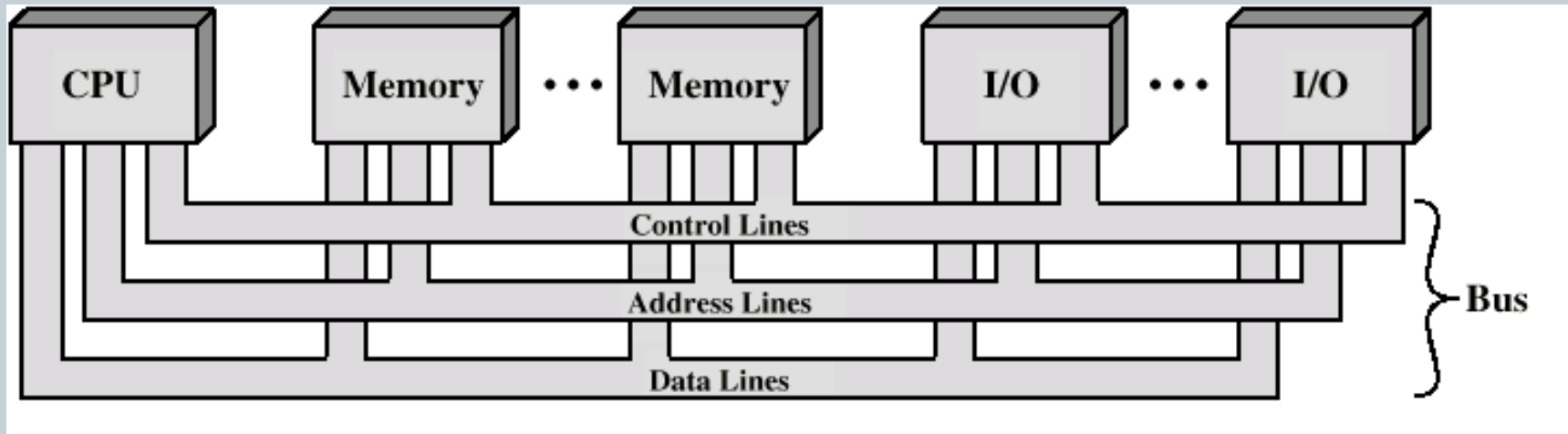
- There are a number of possible interconnection systems
- Single and multiple BUS structures are most common
- e.g. Control/Address/Data bus (PC)
- e.g. Unibus (DEC-PDP)

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

Bus Interconnection Scheme



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

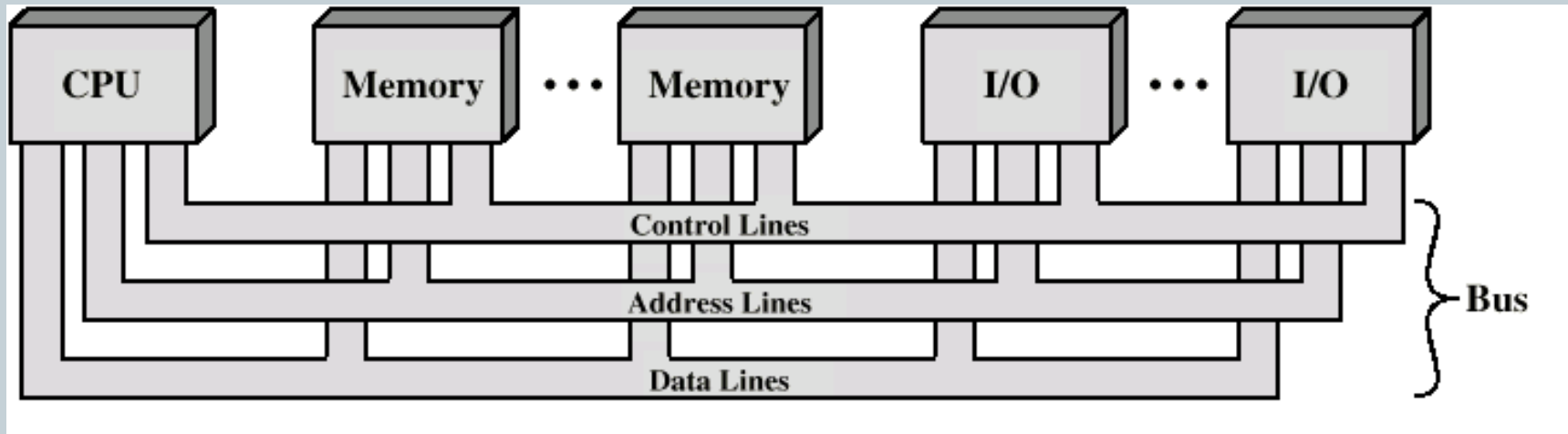
$$\begin{aligned} \text{○ } 2^{16} &= 2^{10} \times 2^6 \\ &= 2^6 \times 2^{10} \\ &= 64 \text{ k} \end{aligned}$$

Control Bus



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

Bus Interconnection Scheme



Big and Yellow?

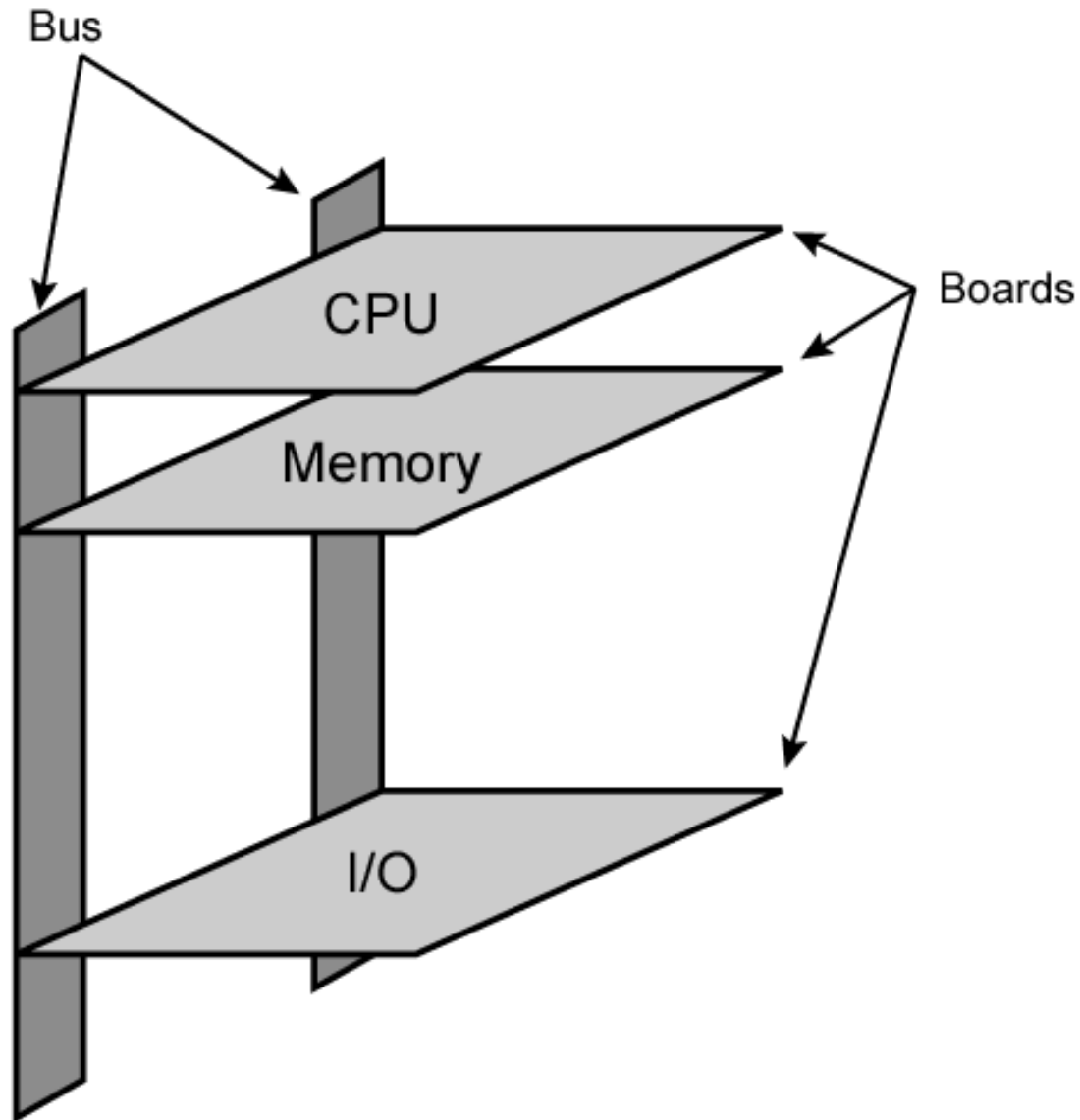


- What do buses look like?
 - Parallel lines on circuit boards
 - Ribbon cables
 - Strip connectors on mother boards
 - ✦ e.g. PCI
 - Sets of wires

Bus



Physical Realization of Bus Architecture

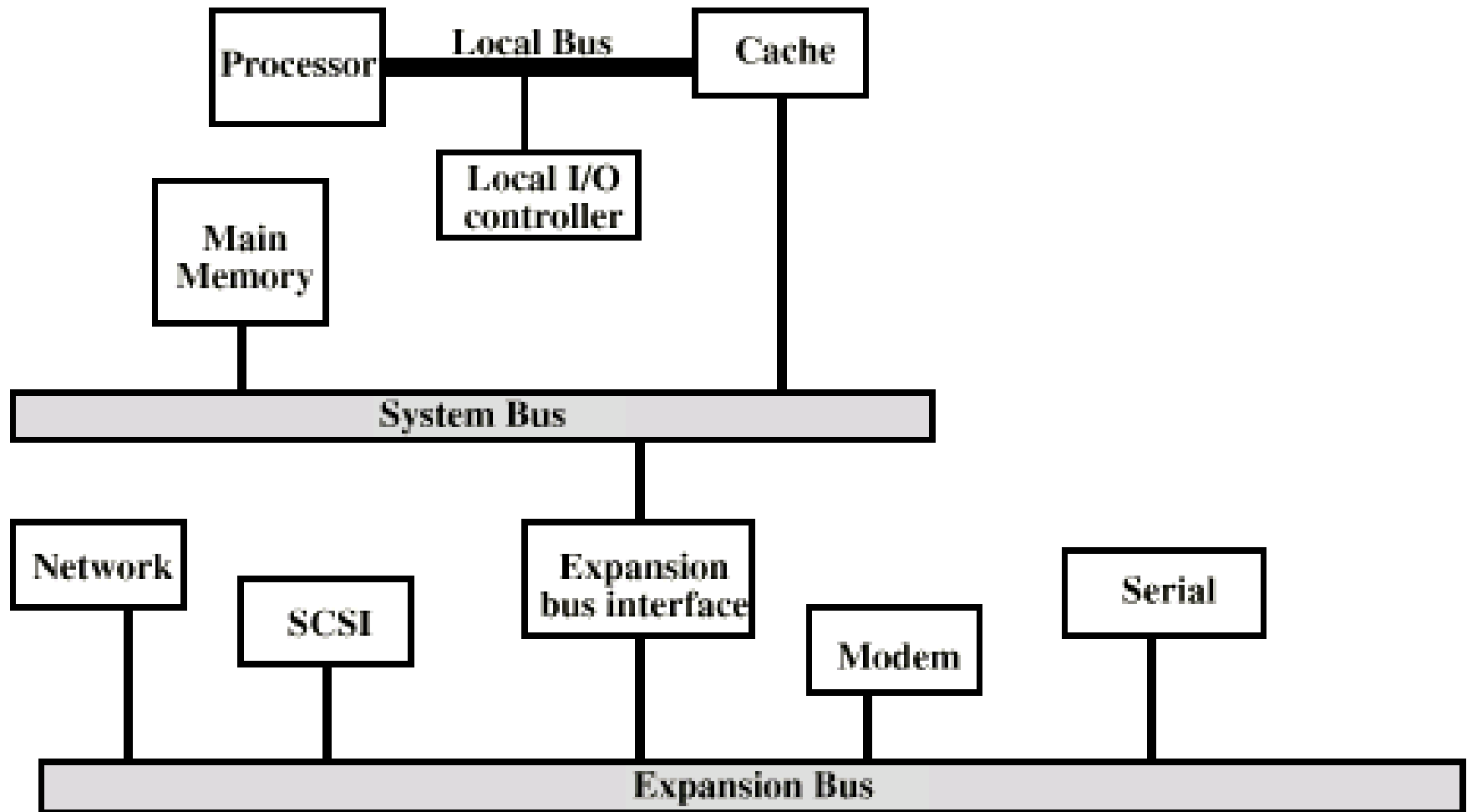


Single Bus Problems

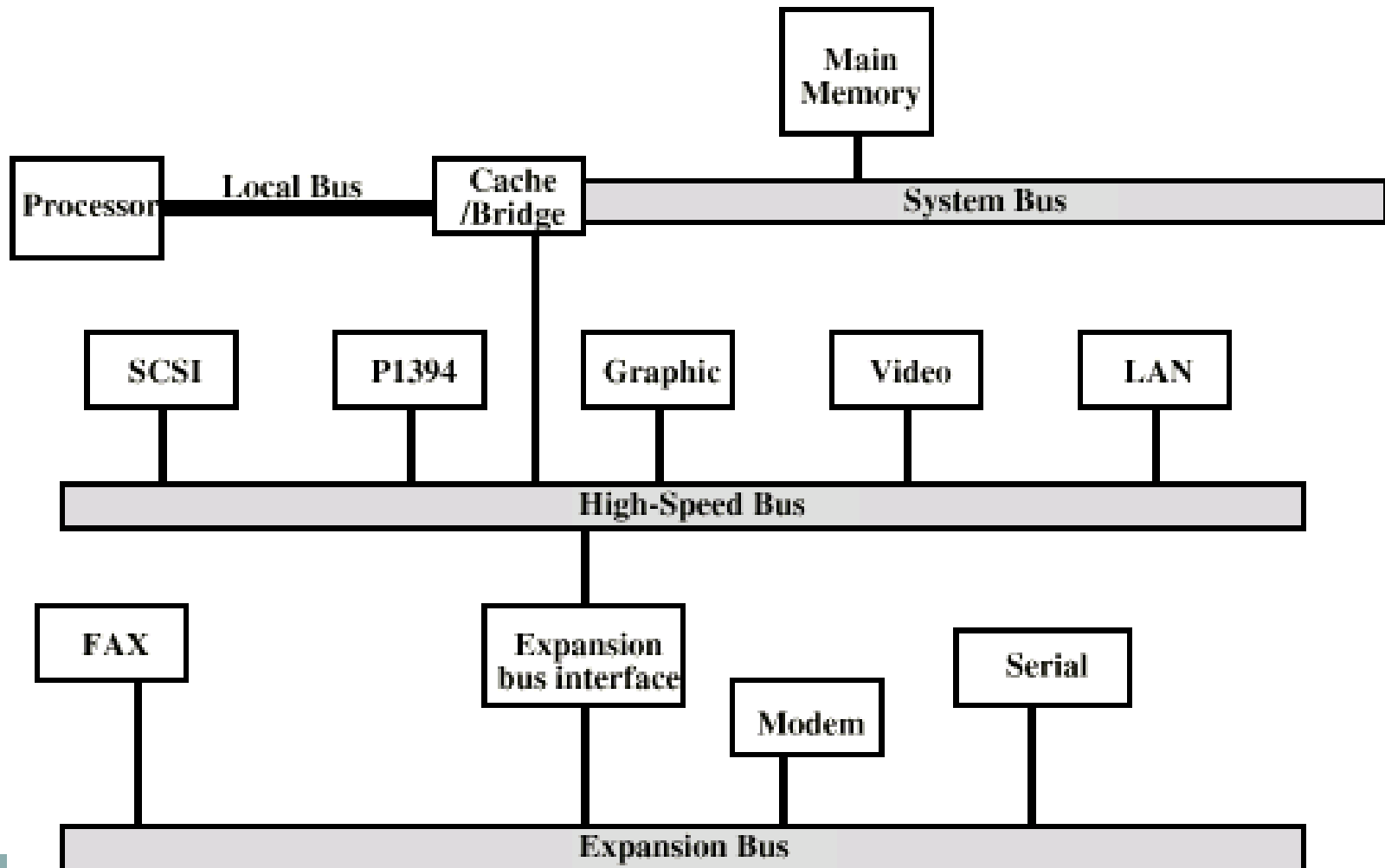


- Lots of devices on one bus leads to:
 - Propagation delays
 - ✦ Long data paths mean that co-ordination of bus use can adversely affect performance
 - ✦ If aggregate data transfer approaches bus capacity
- Most systems use multiple buses to overcome these problems

Traditional (ISA — *Industry Standard Architecture*) (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

Reading Assignment



- Bus Arbitration

Quiz



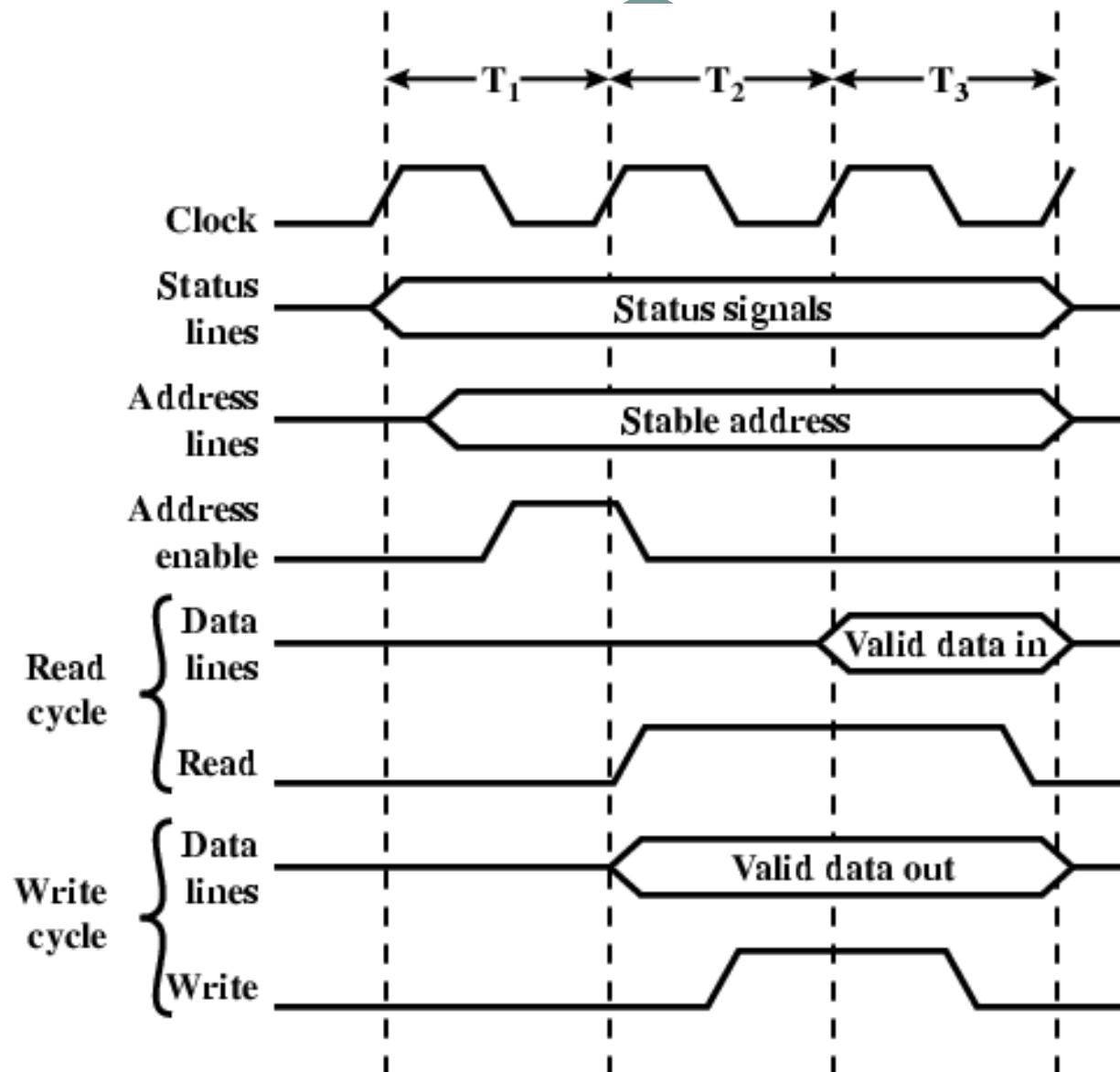
- Define A system Bus and discuss types of Buses? (10%)

Timing

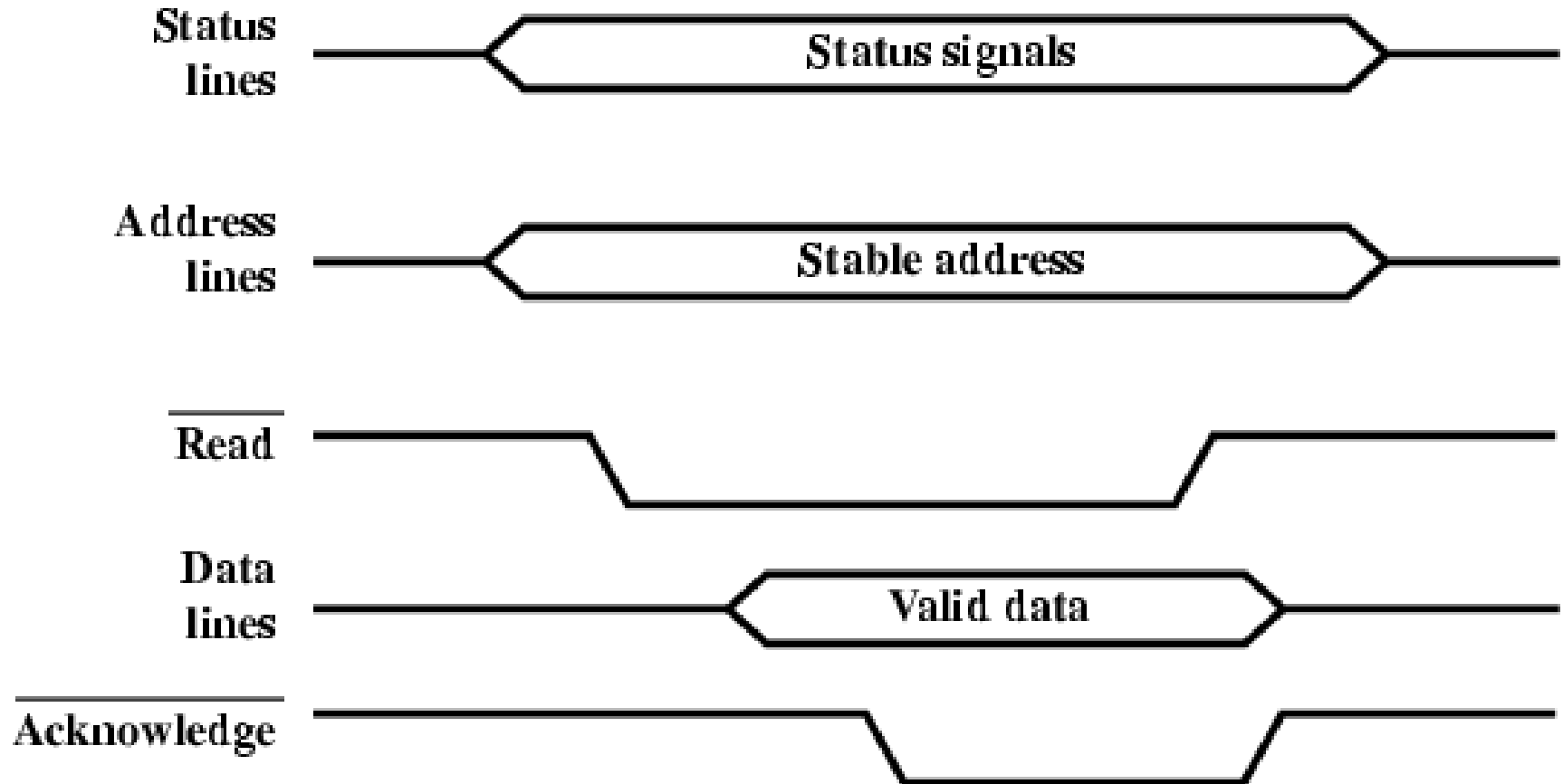


- Co-ordination of events on bus
- Synchronous
 - Events determined by clock signals
 - Control Bus includes clock line
 - A single 1-0 is a bus cycle
 - All devices can read clock line
 - Usually sync on leading edge
 - Usually a single cycle for an event

Synchronous Timing Diagram



Asynchronous Timing – Read Diagram



Asynchronous Timing – Write Diagram



Status
lines



Address
lines



Data
lines



Write



Acknowledge



NEXT LECTURE



Chapter Four: **Cache Memory**