Top Level View of Computer Function and Interconnection

Week 2

Tujuan

- instruction cycle and the role of interrupts.
- Describe the concept of interconnection within a computer system.
- Assess the relative advantages of point-to-point interconnection compared to bus interconnection.
- Present an overview of QPI.
- Present an overview of PCIe

- Komponen computer pada top level?
- Komponen saling terhubung dengan *interconnection* untuk melakukakn fungsi dasar computer, mengekskusi program
- At top level, we can characterize a computer system by describing
  - the external behavior of each component, that is, the data and control signals that it exchanges with other components,
  - the interconnection structure and the controls required to manage the use of the interconnection structure.

Computer component

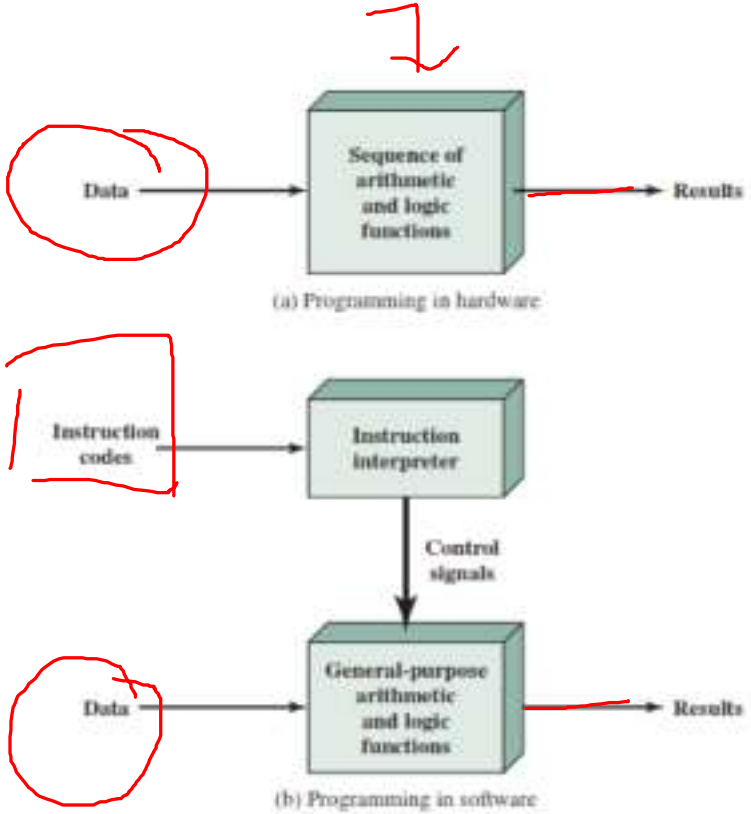
- von Neumann architecture:
  - Data and instructions are stored in a single read–write memory.
  - The contents of this memory are addressable by location, without regard to the type of data contained there.
  - Execution occurs in a sequential fashion (unless explicitly modified) from one instruction to the next

Hardware programming

- There is a small set of basic logic components that can be combined in various ways to store binary data and perform arithmetic and logical operations on that data
- Process of connecting the various components in the desired configuration as a form of **programming**.
- The resulting “program” is in the form of hardware and is termed a hardwired program.

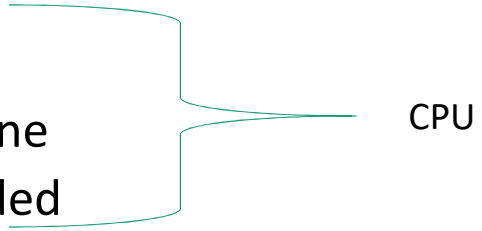
Program Concept

- 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



Program?

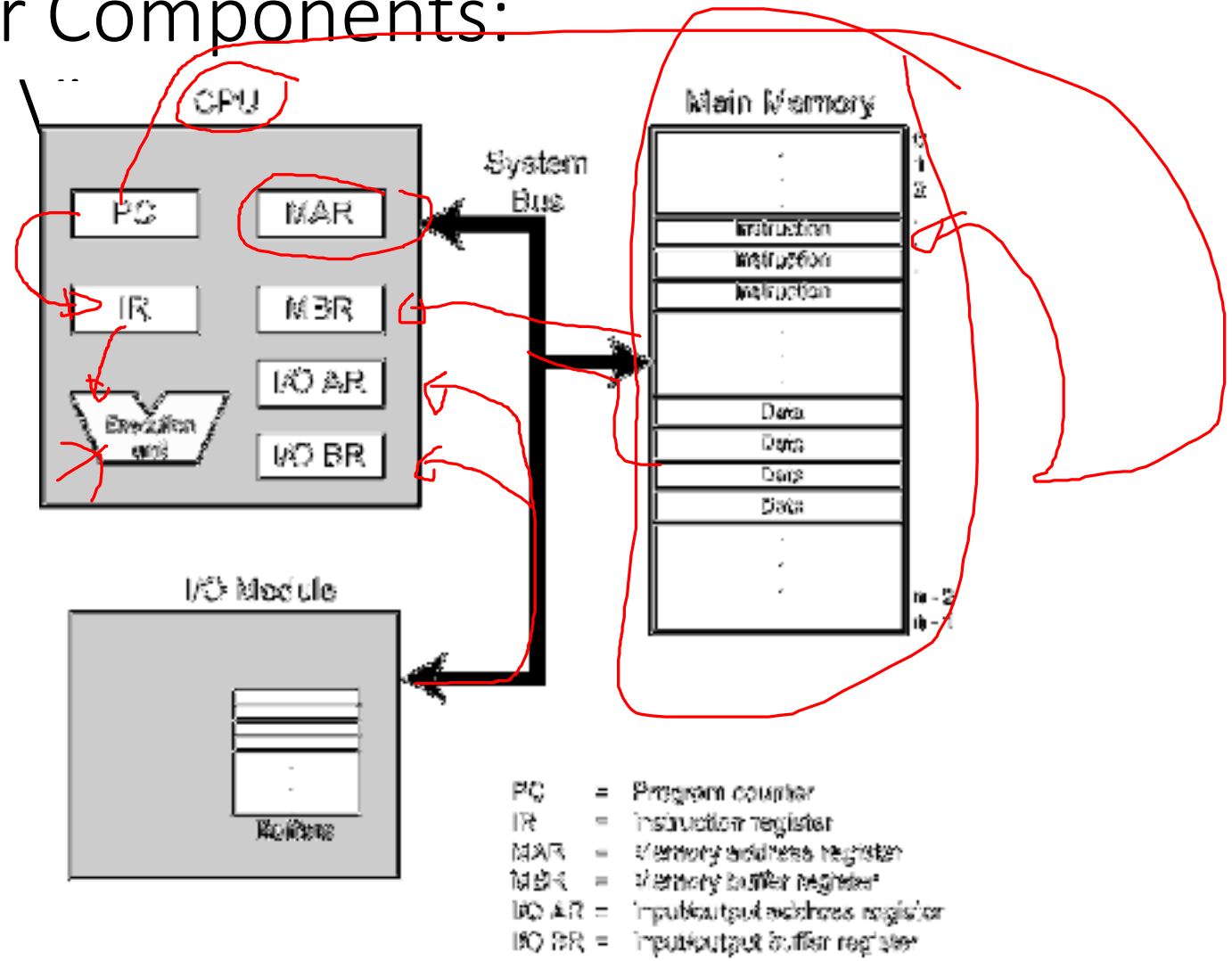
- A sequence of steps, or codes
- Each codes => instruction
  - Each instruction is interpreted
  - For each code, an arithmetic or logical operation is done
  - For each code ,a different set of control signals is needed



Program

- Kebutuhan lain:
  - Input output
  - Memory

Computer Components: Top Level V

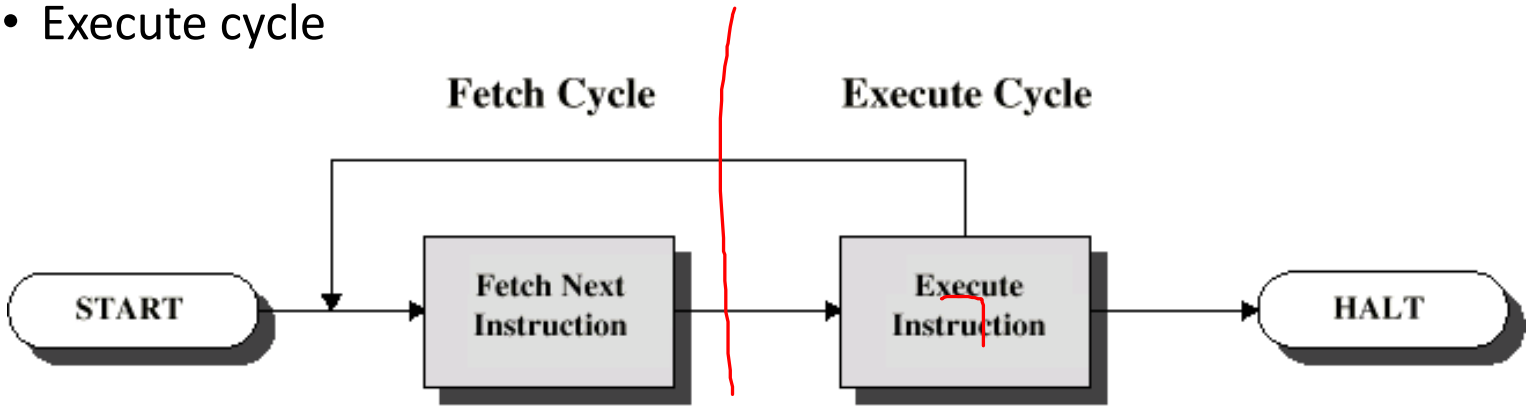


Fungsi?

- PC
- IR
- MAR
- MBR

Instruction Cycle

- The processing required for a single instruction is called an instruction cycle
- Two steps:
  - Fetch cycle
  - Execute cycle



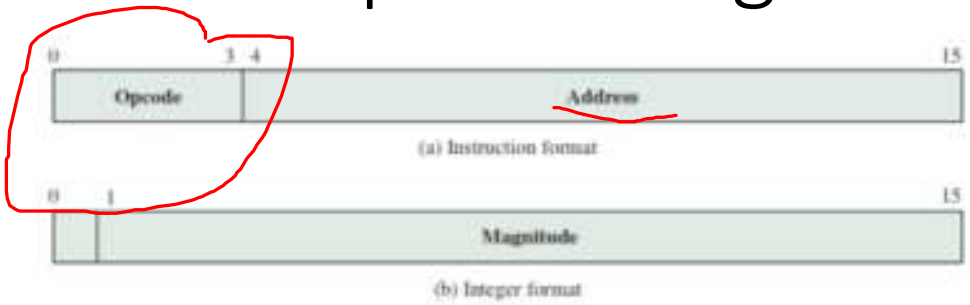
Fetch Cycle

- Program Counter (PC) holds address of next instruction to fetch
- Processor fetches instruction from memory location pointed to by PC
- Increment PC after each instruction fetch so that it will fetch the next instruction in sequence
  - Unless told otherwise
- Instruction loaded into Instruction Register (IR)
- Processor interprets instruction and performs required actions

Execute Cycle

- actions fall into four categories:
  - 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

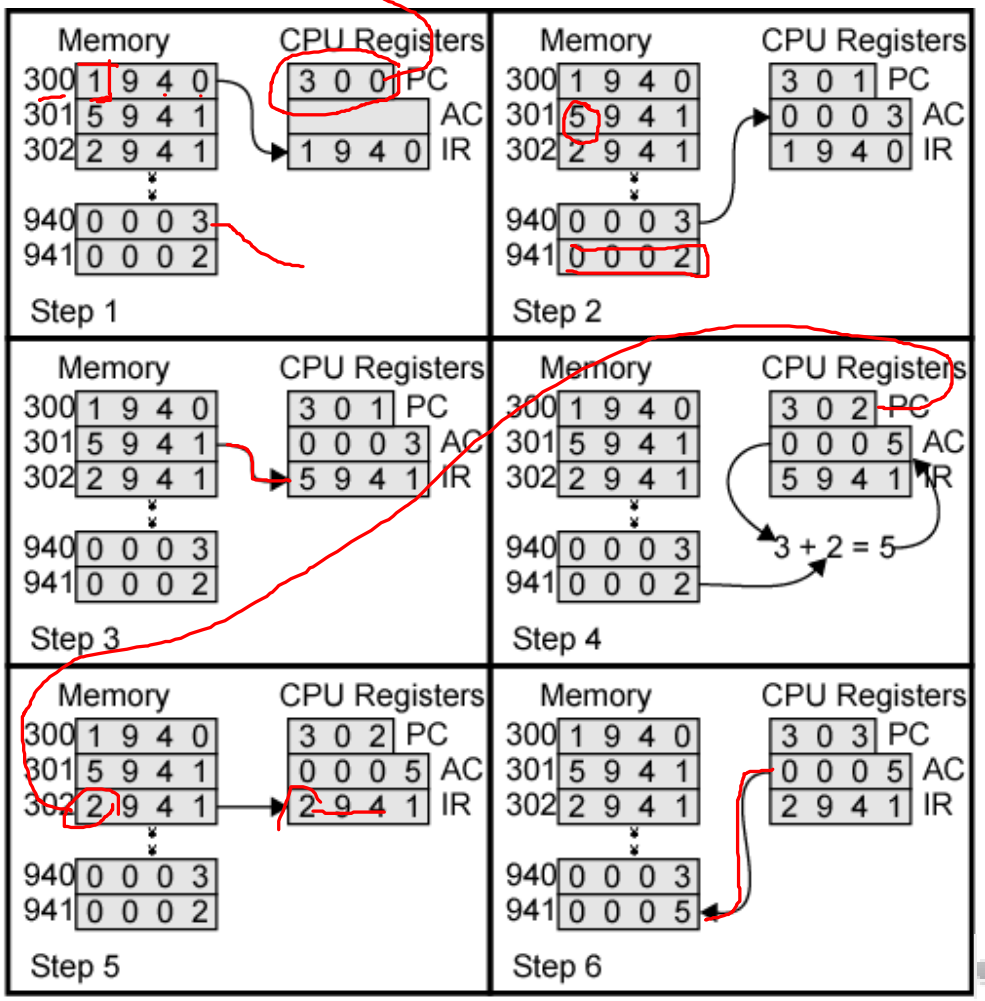


Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

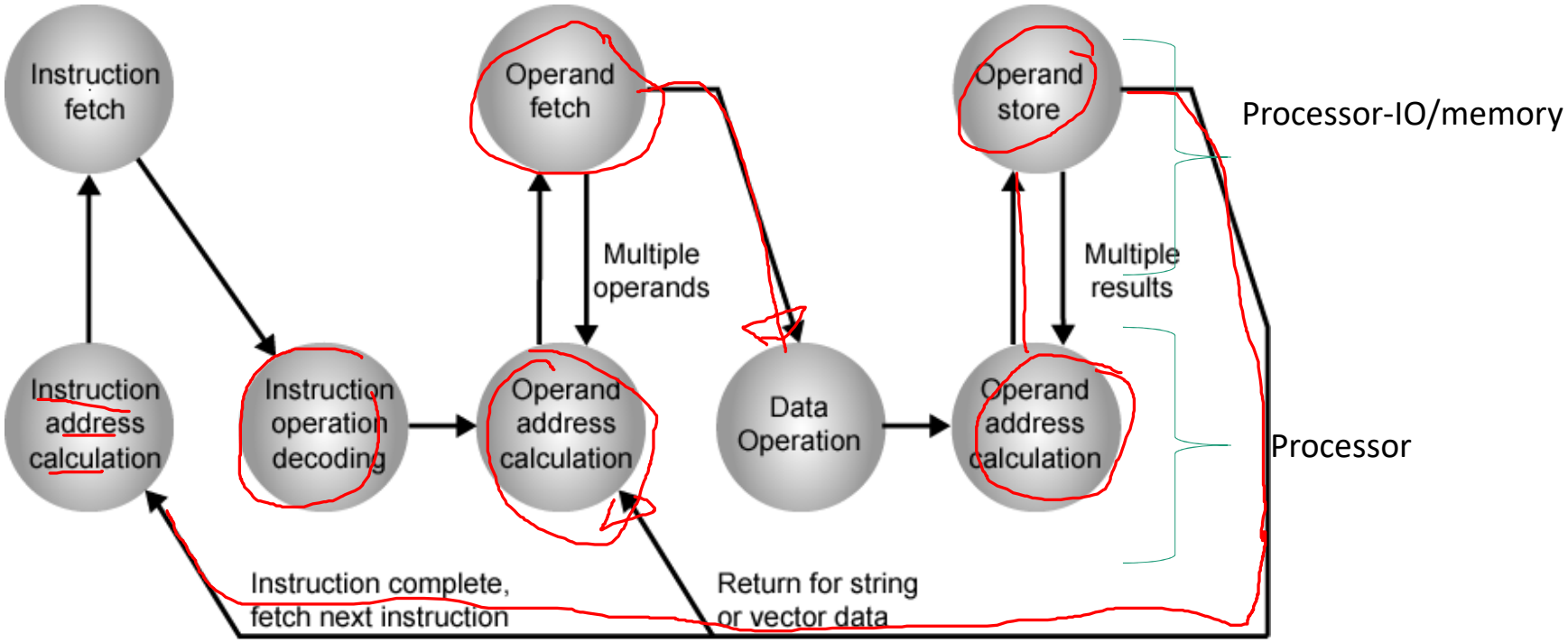
(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes



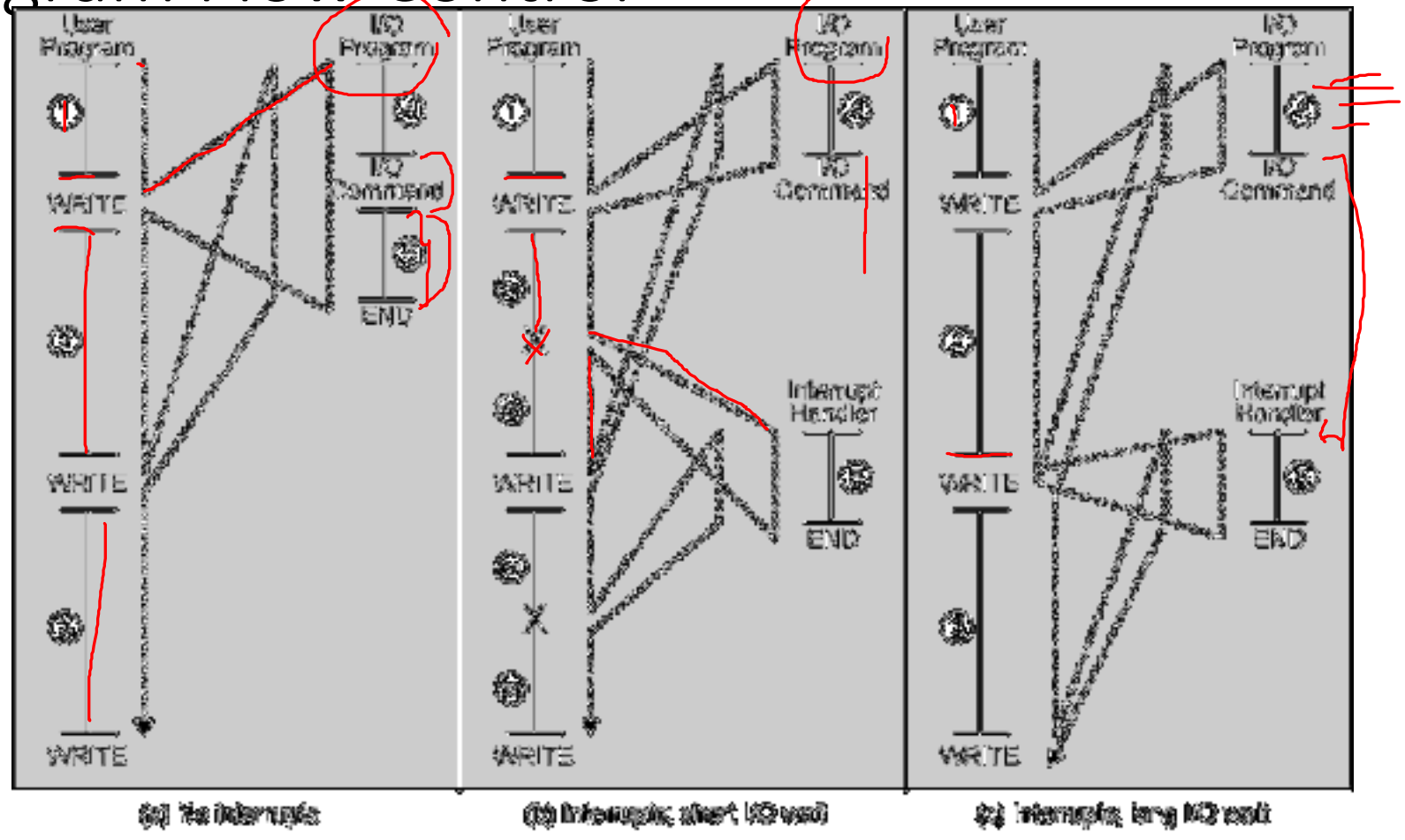
Instruction Cycle State Diagram



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

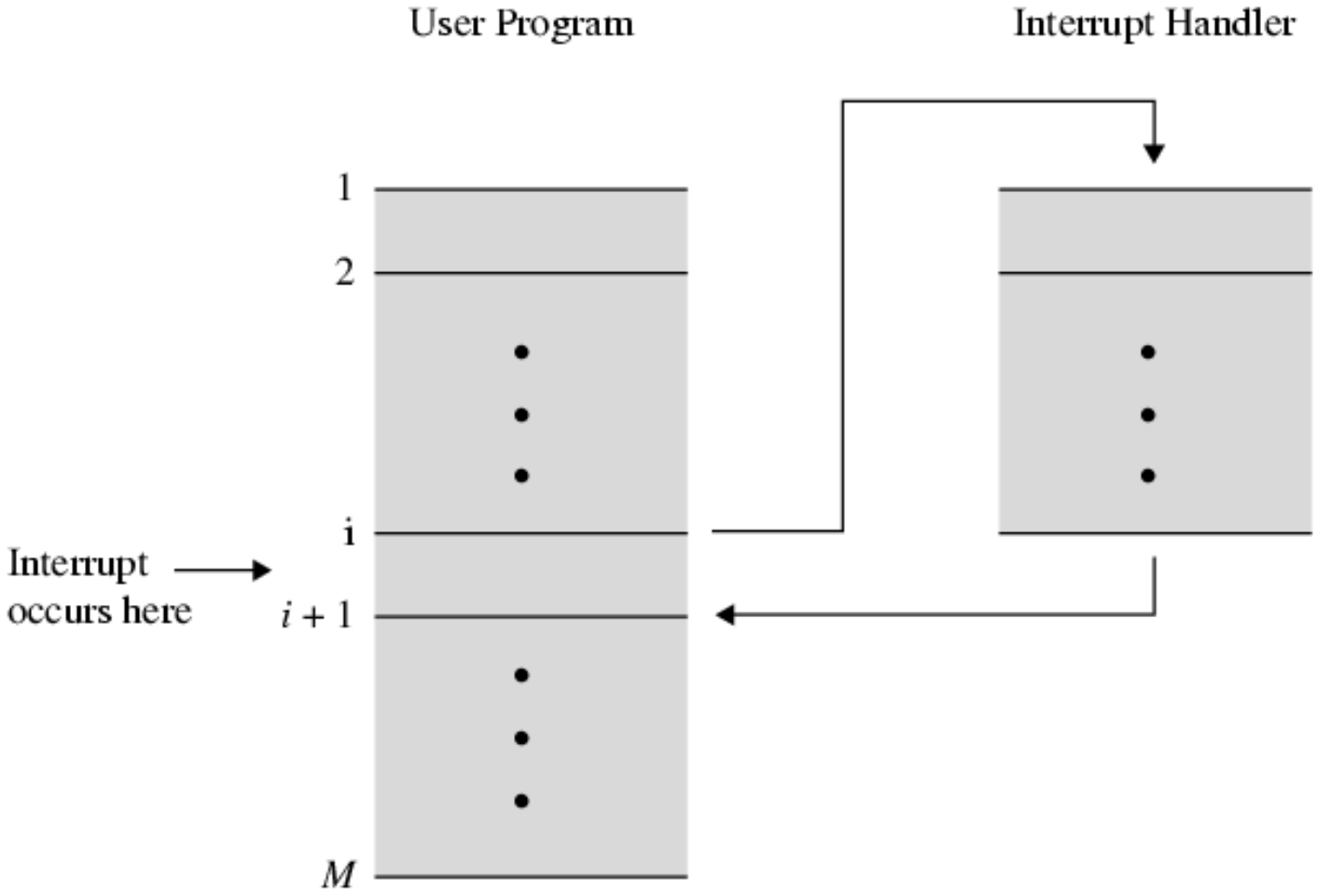
Program Flow Control



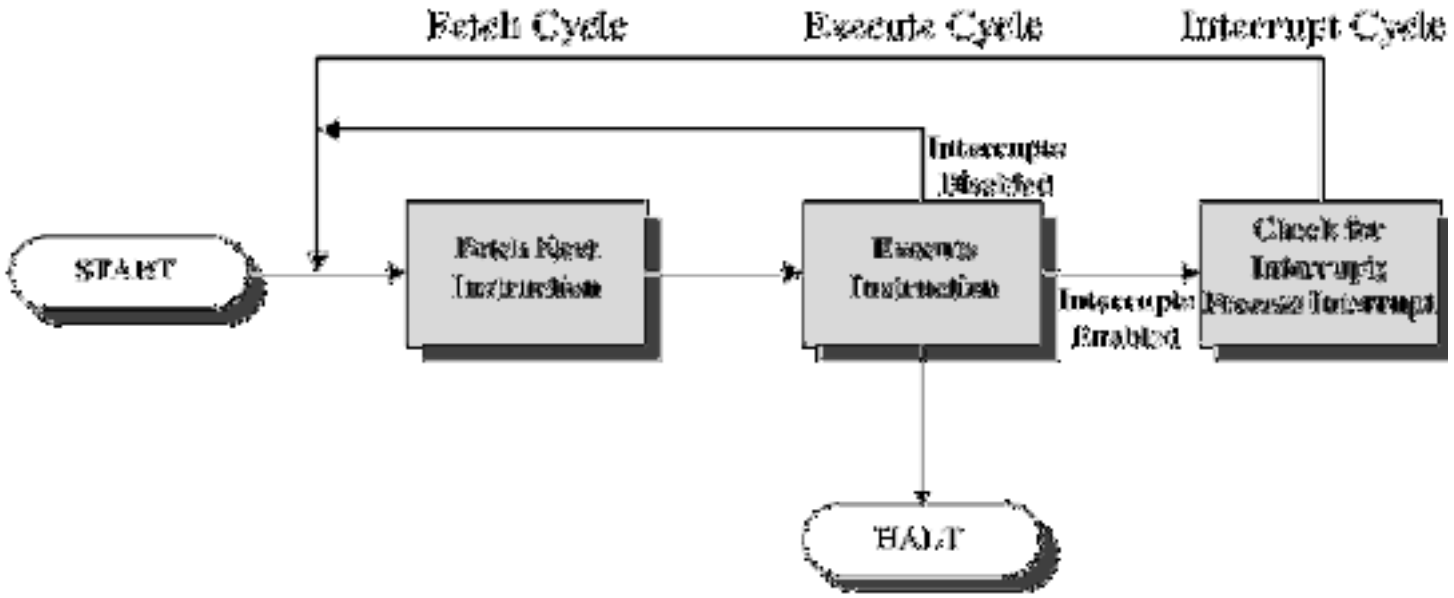
Interrupt Cycle

- 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: This means saving the address of the next instruction to be executed (current contents of the program counter) and any other data relevant to the
  - processor’s current activit
  - Set PC to start address of interrupt handler routine
  - Process interrupt
  - Restore context and continue interrupted program

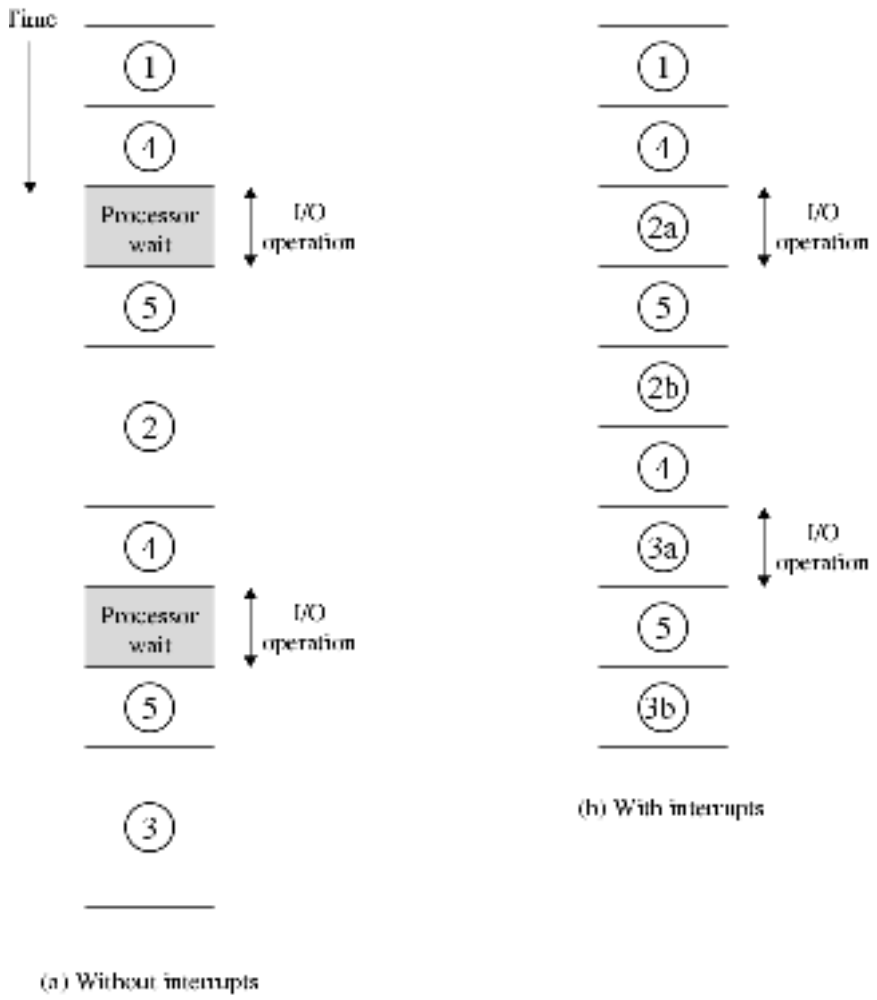
Transfer of Control via Interrupts



Instruction Cycle with Interrupts



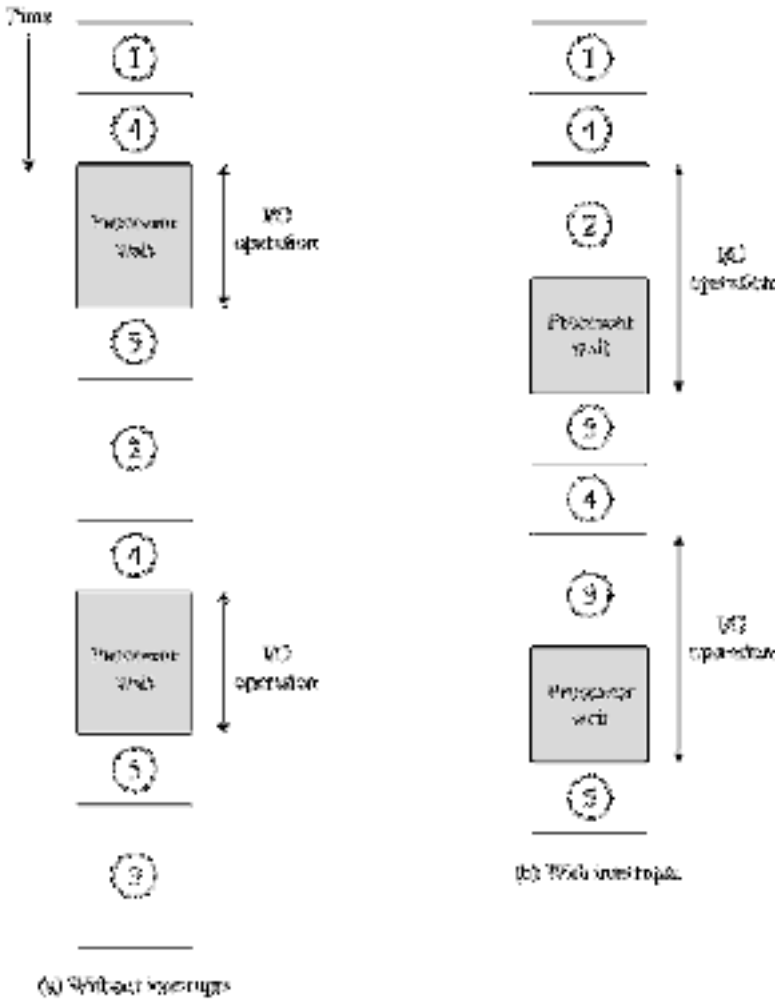
Program Timing Short I/O Wait



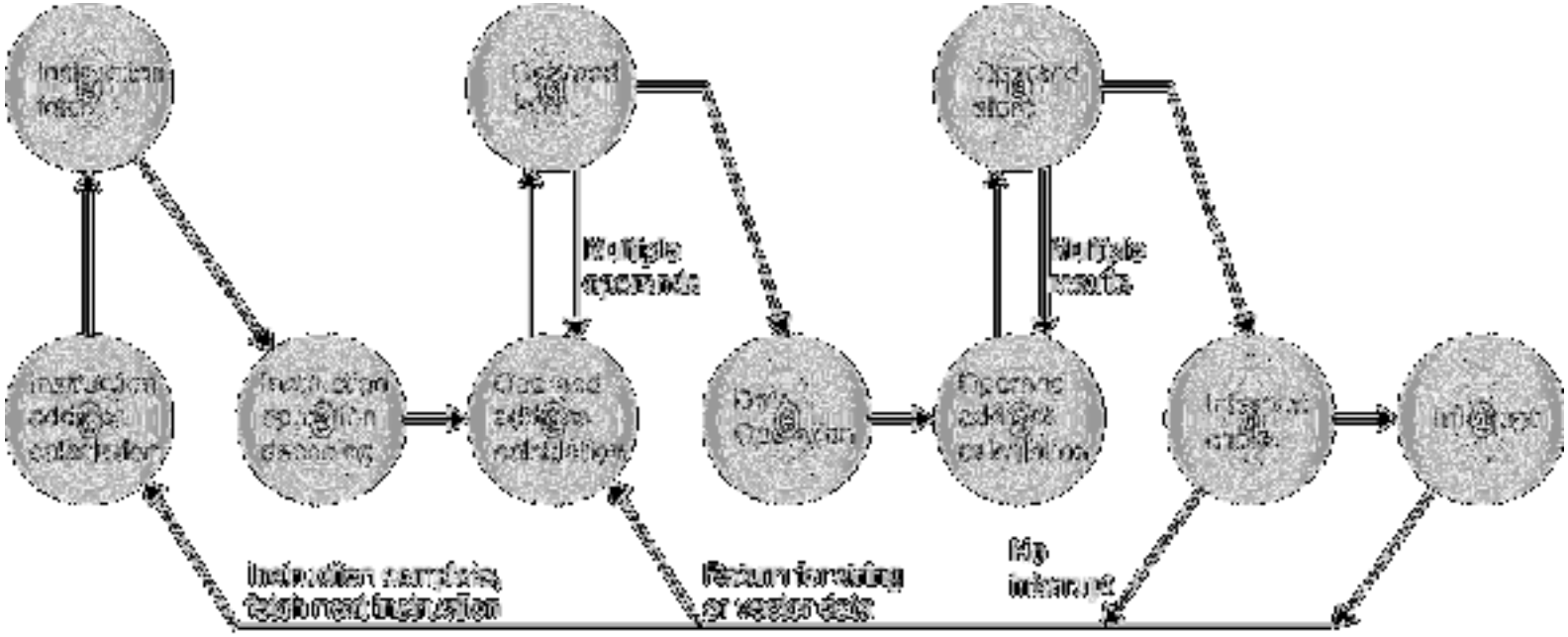
(a) Without interrupts

(b) With interrupts

Program Timing
Long I/O Wait



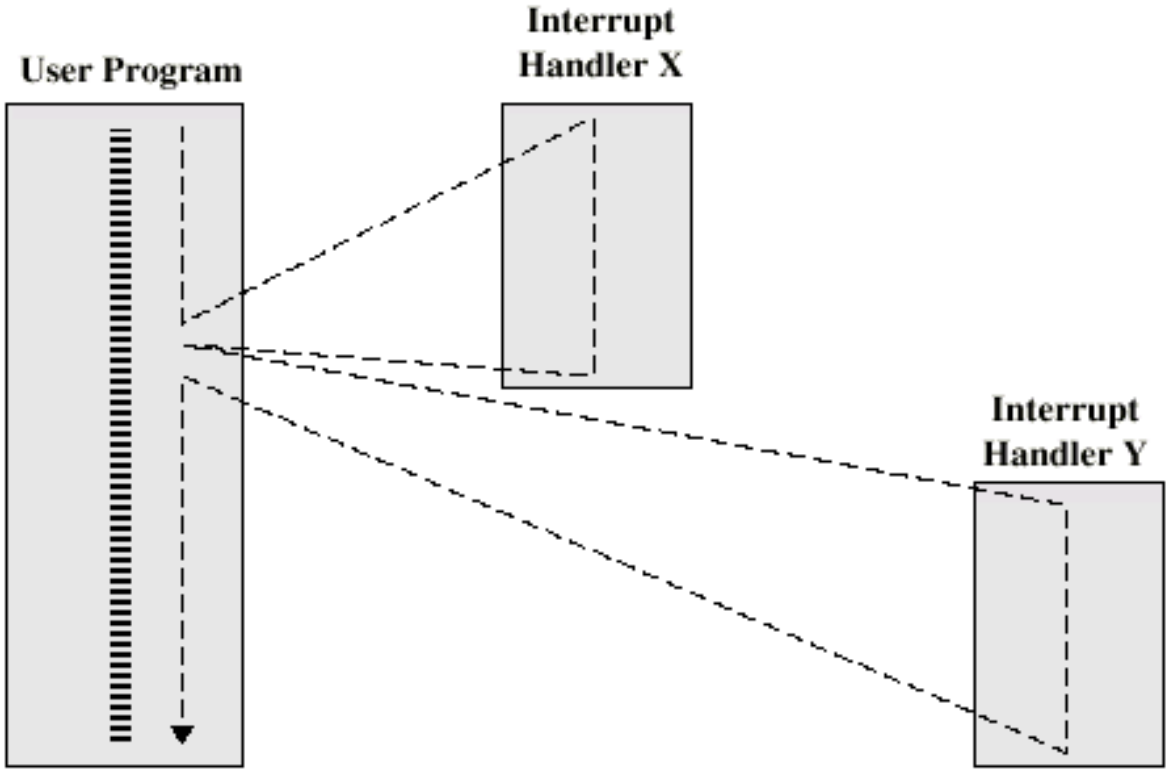
Instruction Cycle (with Interrupts) - State Diagram



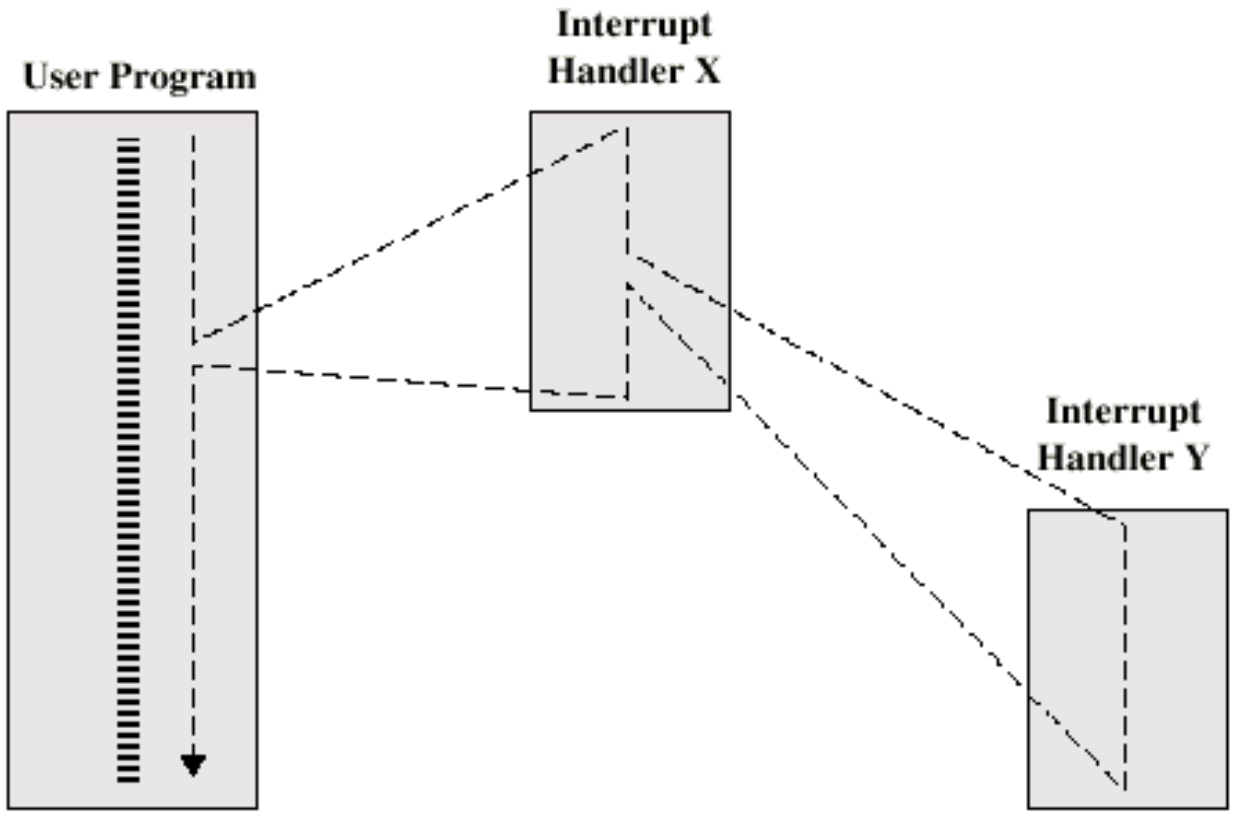
Multiple Interrupts

- 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

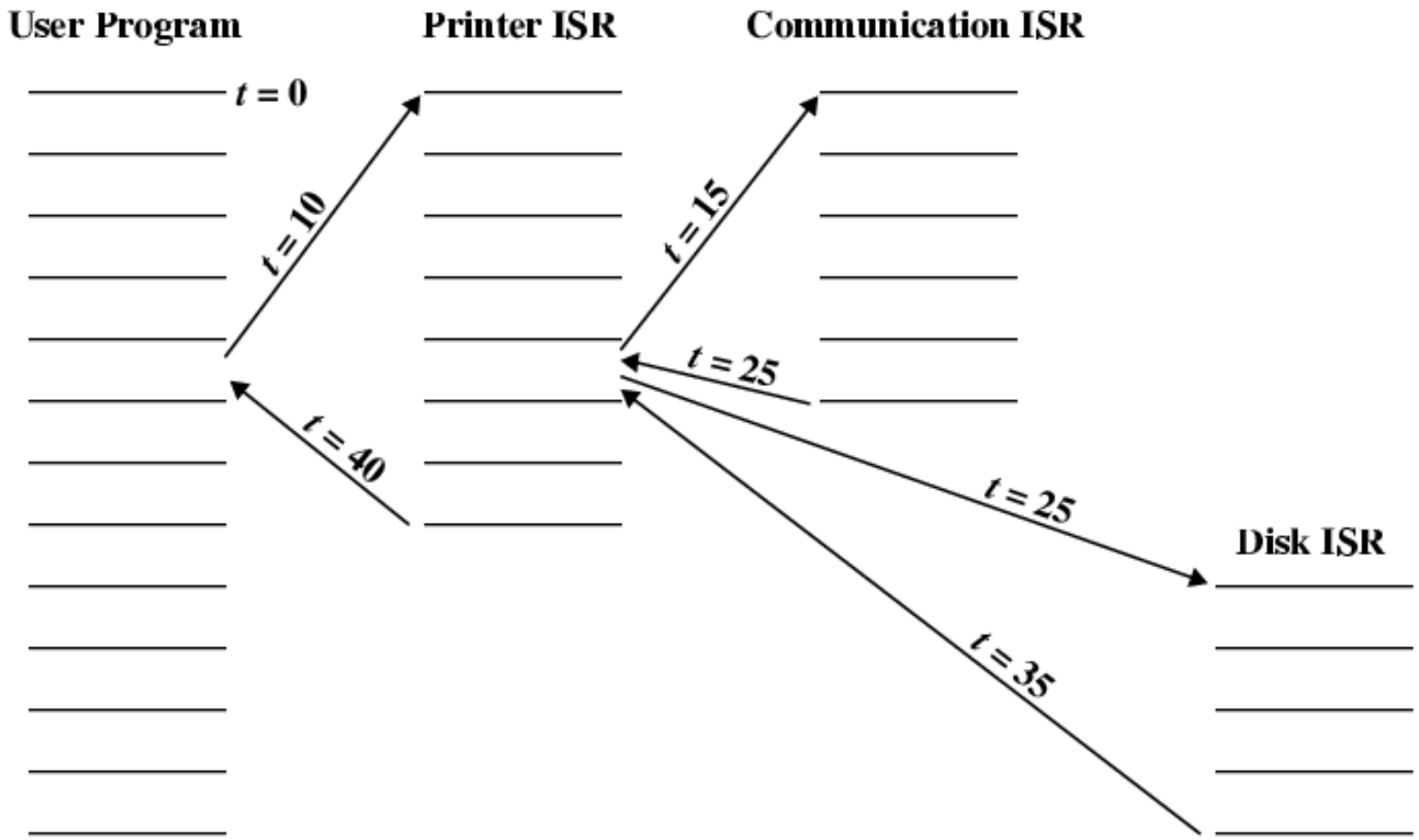
Multiple Interrupts - Sequential



Multiple Interrupts – Nested

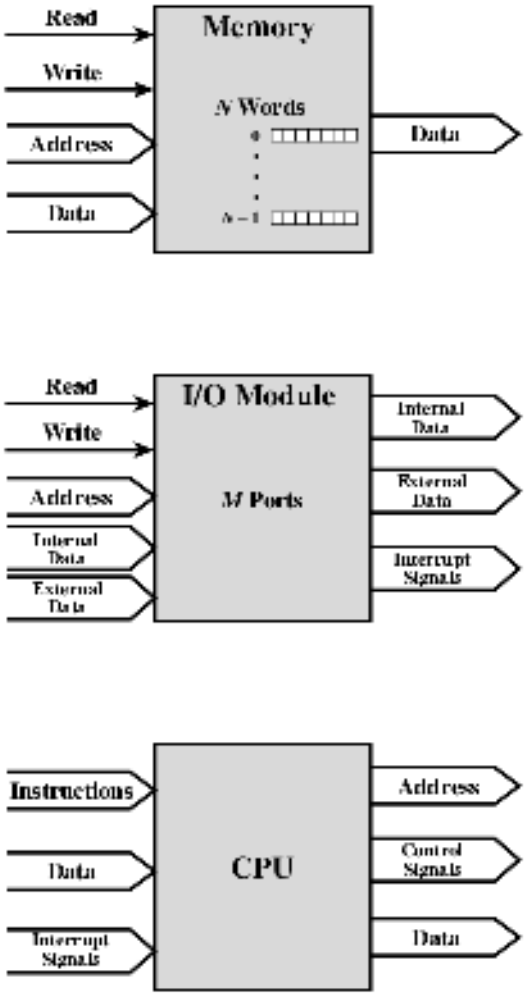


Time Sequence of Multiple Interrupts



Connection

- 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

Interconnection structure

(1) the bus and various multiple-bus structures

(2) pointto-point interconnection structures with packetized data transfer

Buses

- There are a number of possible interconnection systems
- A bus is a communication pathway connecting two or more devices
- 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
- A key characteristic of a bus is that it is a shared transmission medium
- Multiple devices connect to the bus, and a signal transmitted by any one device is available for reception by all other devices attached to the bus
- 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

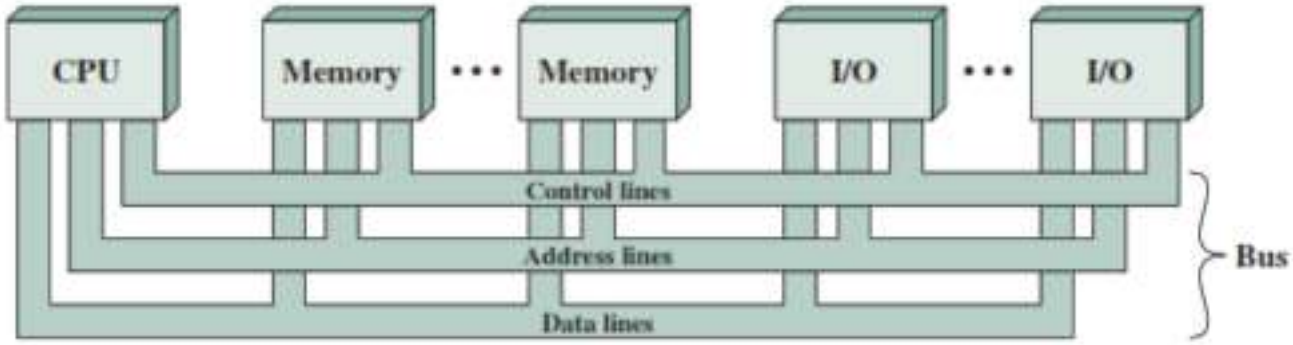


Figure 3.16 Bus Interconnection Scheme

- 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

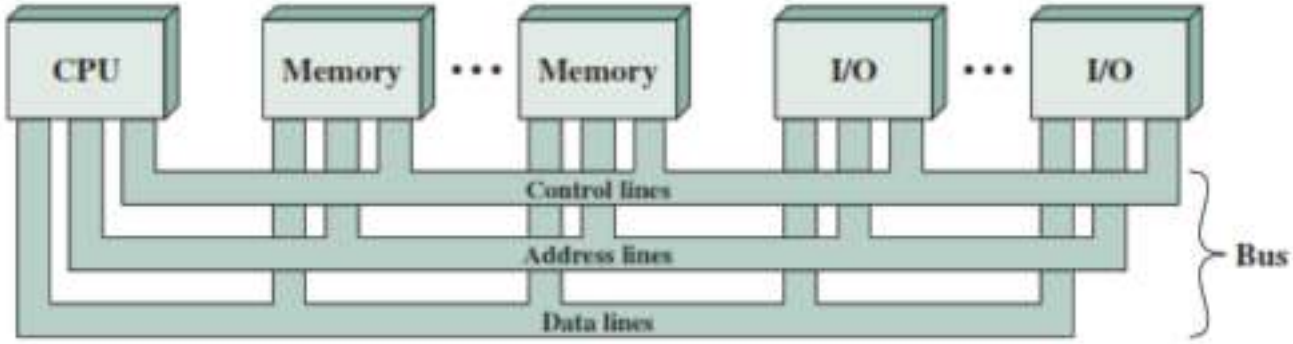


Figure 3.16 Bus Interconnection Scheme

- 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

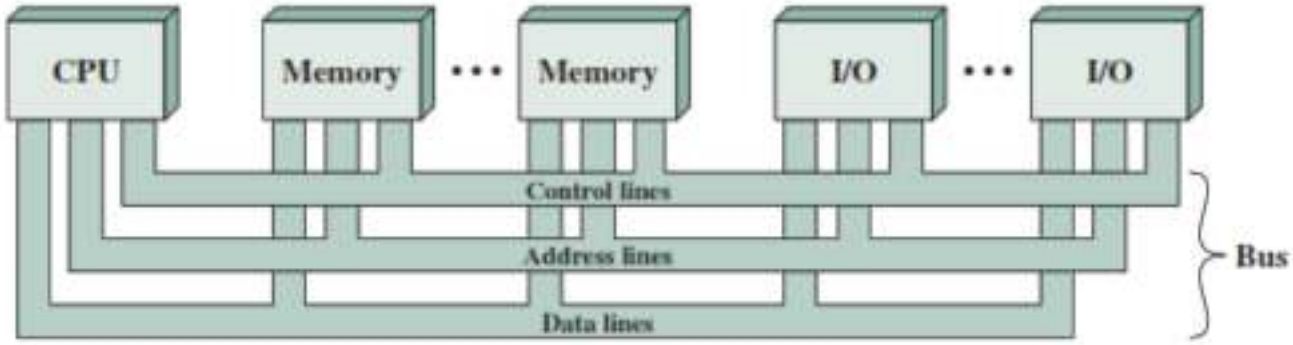
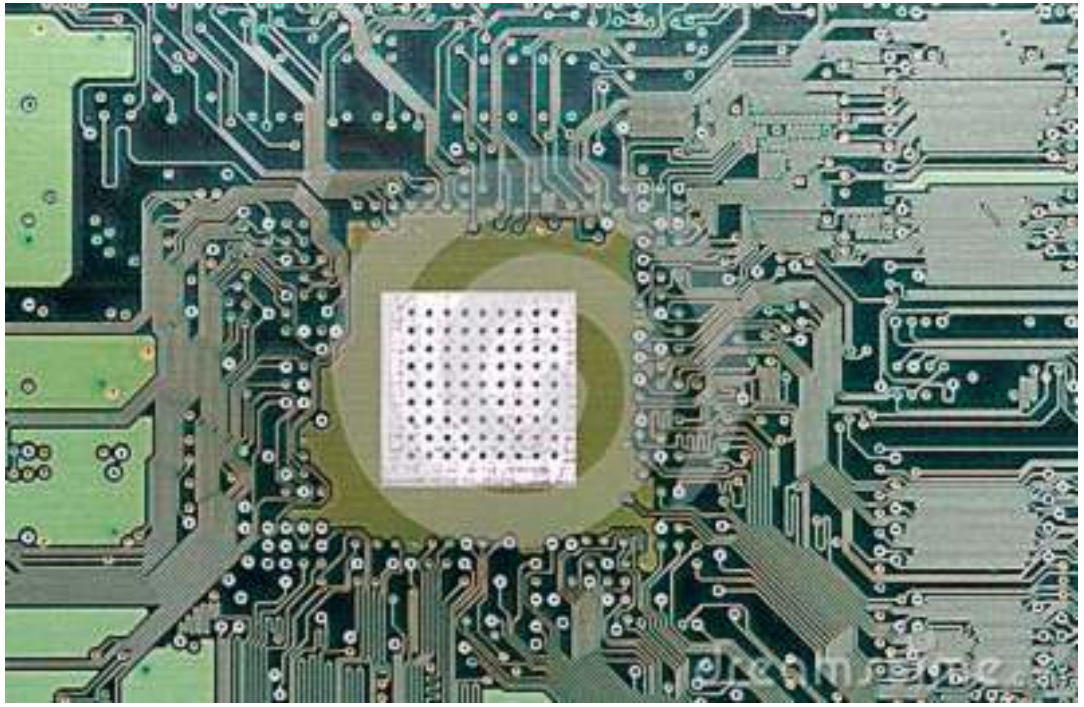


Figure 3.16 Bus Interconnection Scheme

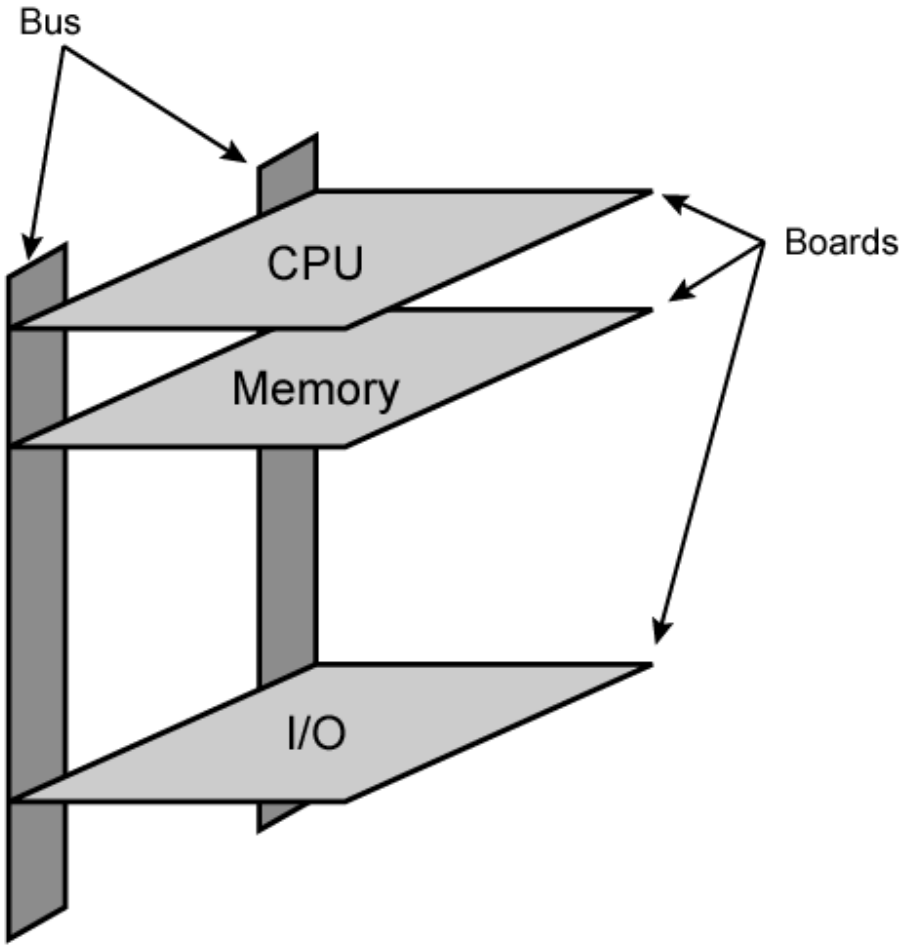
- Control and timing information
  - Memory read/write signal
  - Interrupt request
  - Clock signals

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



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

point-to-point interConneCt

- The principal reason driving the change from bus to point-to-point interconnect was the electrical constraints encountered with increasing the frequency of wide synchronous bus
- Compared to the shared bus, the point-to point interconnect has lower latency, higher data rate, and better scalability

QuickPath Interconnect (QPI)

- Multiple direct connections: Multiple components within the system enjoy direct pairwise connections to other components. This eliminates the need for arbitration found in shared transmission systems.
- Layered protocol architecture: As found in network environments, such as TCP/IP-based data networks, these processor-level interconnects use a layered protocol architecture, rather than the simple use of control signals found in shared bus arrangements.
- Packetized data transfer: Data are not sent as a raw bit stream. Rather, data are sent as a sequence of packets, each of which includes control headers and error control codes

QPI layers

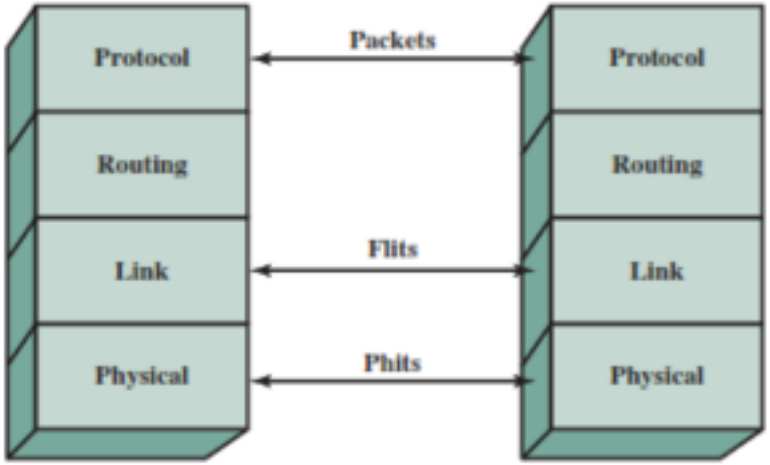


Figure 3.18 QPI Layers

- Physical: Consists of the actual wires carrying the signals, as well as circuitry and logic to support ancillary features required in the transmission and receipt of the 1s and 0s. The unit of transfer at the Physical layer is 20 bits, which is called a Phit (physical unit).
- Link: Responsible for reliable transmission and flow control. The Link layer's unit of transfer is an 80-bit Flit (flow control unit).
- Routing: Provides the framework for directing packets through the fabric.
- Protocol: The high-level set of rules for exchanging packets of data between devices. A packet is comprised of an integral number of Flits

Physical Layer

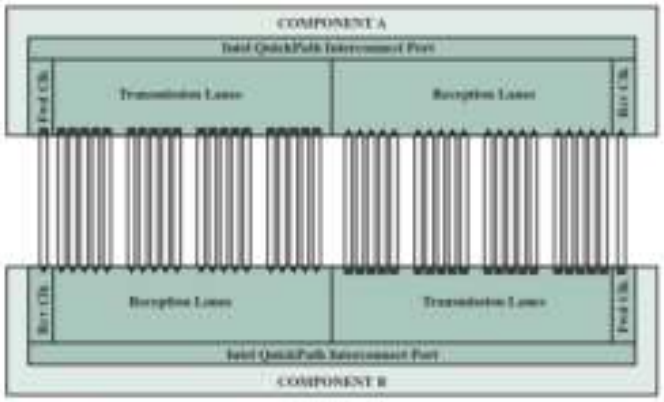


Figure 5.19 Physical Interface of the Intel QPI Interconnect

- Each data path consists of a pair of wires that transmits data one bit at a time; the pair is referred to as a lane.
- There are 20 data lanes in each direction (transmit and receive), plus a clock lane in each direction
- The lanes in each direction are grouped into four quadrants of 5 lanes each.
- Another function performed by the physical layer is that it manages the translation between 80-bit flits and 20-bit phits using a technique known as multilane distribution

Link Layer

- performs two key functions: flow control and error control
- Operate on the level of the flit (flow control unit)
- Flow control function is needed to ensure that a sending QPI entity does not overwhelm a receiving QPI entity by sending data faster than the receiver can process the data and clear buffers for more incoming data
- Error control: cyclic redundancy check => CRC (8 bit)

PCIexpress

- The peripheral component interconnect (PCI) is a popular high-bandwidth, processor-independent bus that can function as a mezzanine or peripheral bus.
- Compared with other common bus specifications, PCI delivers better system performance for high-speed I/O subsystems (e.g., graphic display adapters, network interface controllers, and disk controllers).
- The bus-based PCI scheme has not been able to keep pace with the data rate demands of attached devices. Accordingly, a new version, known as PCI Express (PCIe) has been developed

PCIe

- A key requirement for PCIe is high capacity to support the needs of higher data rate I/O devices, such as Gigabit Ethernet.
- Another requirement deals with the need to support time-dependent data streams
  - It is no longer acceptable to treat all data as equal—it is more important, for example, to process streaming data first since late real-time data is as useless as no data

PCIe

- Root complex device, or a chipset or a host bridge, connects the processor and memory subsystem to the PCI Express switch fabric comprising one or more PCIe and PCIe switch devices.
  - root complex acts as a buffering device, to deal with difference in data rates between I/O controllers and memory and processor components.
  - root complex also translates between PCIe transaction formats and the processor and memory signal and control requirements.

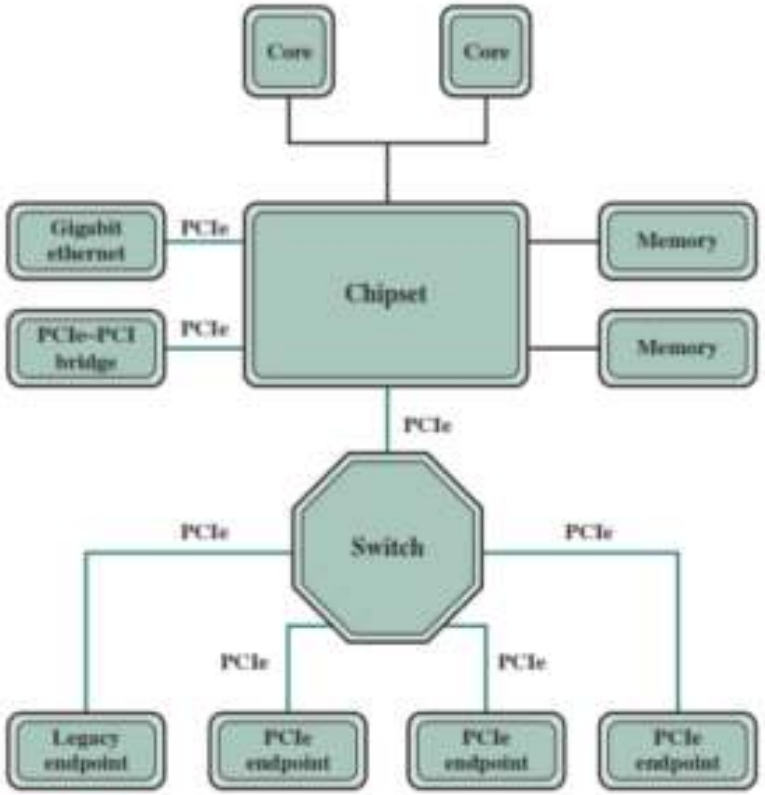


Figure 3.21 Typical Configuration Using PCIe

- Switch: The switch manages multiple PCIe streams.
- PCIe endpoint: An I/O device or controller that implements PCIe, such as a Gigabit ethernet switch, a graphics or video controller, disk interface, or a communications controller
- Legacy endpoint: Legacy endpoint category is intended for existing designs that have been migrated to PCI Express, and it allows legacy behaviors such as use of I/O space and locked transactions. PCI Express endpoints are not permitted to require the use of I/O space at runtime and must not use locked transactions. By distinguishing these categories, it is possible for a system designer to restrict or eliminate legacy behaviors that have negative impacts on system performance and robustness.
- PCIe/PCI bridge: Allows older PCI devices to be connected to PCIe-basedsystems.

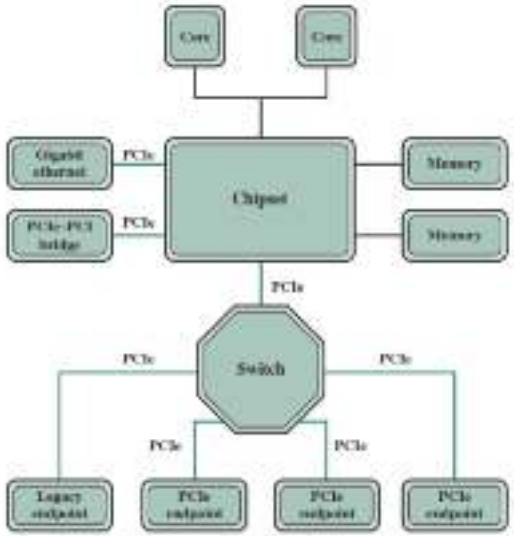


Figure 3.21 Typical Configuration Using PCIe

PCIe Layers

- Physical: Consists of the actual wires carrying the signals, as well as circuitry and logic to support ancillary features required in the transmission and receipt of the 1s and 0s.
- Data link: Is responsible for reliable transmission and flow control. Data packets generated and consumed by the DLL are called Data Link Layer Packets (DLLPs).
- Transaction: Generates and consumes data packets used to implement load/ store data transfer mechanisms and also manages the flow control of those packets between the two components on a link. Data packets generated and consumed by the TL are called Transaction Layer Packets (TLPs).

PCIe layers

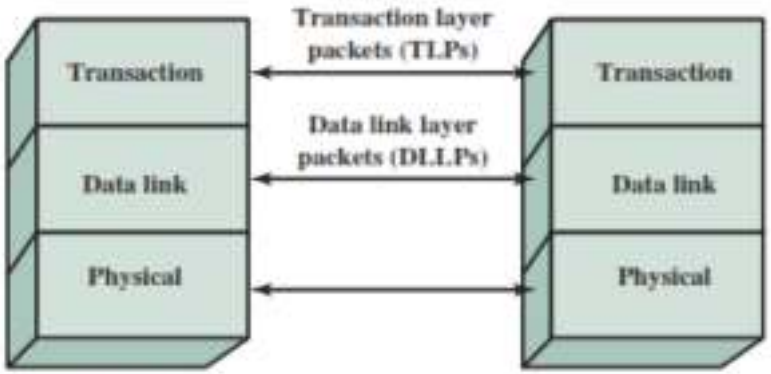
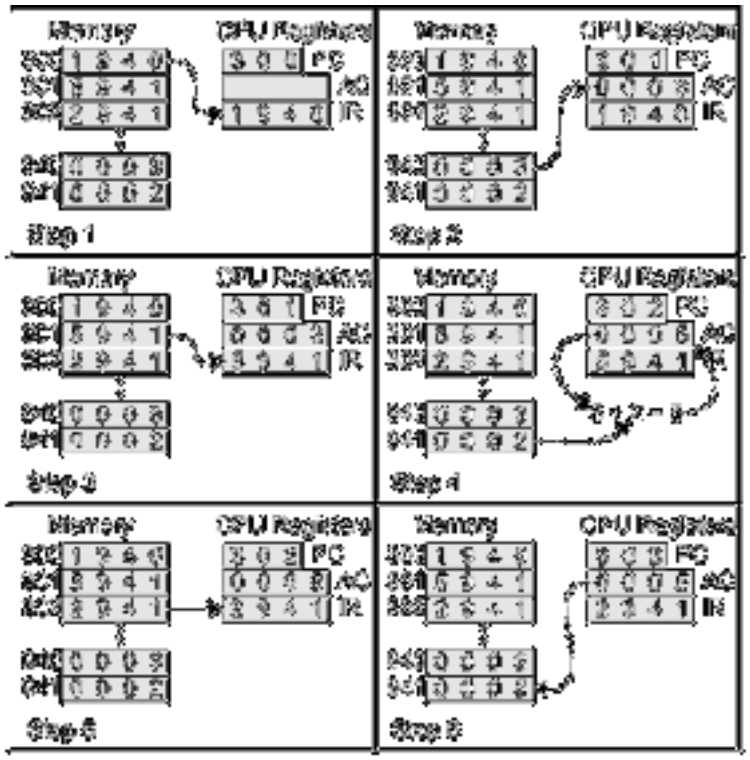


Figure 3.22 PCIe Protocol Layers



(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

3.1 The hypothetical machine of Figure 3.4 also has two I/O instructions:

0011 = Load AC from I/O

0111 = Store AC to I/O

In these cases, the 12-bit address identifies a particular I/O device. Show the program execution (using the format of Figure 3.5) for the following program:

1. Load AC from device 5.
2. Add contents of memory location 940.
3. Store AC to device 6.

Assume that the next value retrieved from device 5 is 3 and that location 940 contains a value of 2.

3005

5940

Dev 5 | 3