Advanced Computer Architecture

COMP 5123

Fall 2016

Computer Science Department

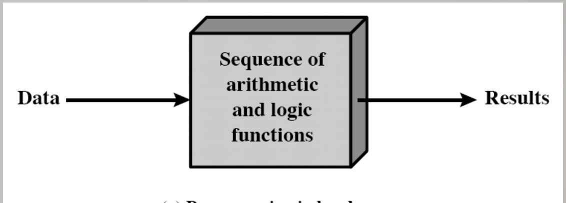
Prairie View A&M University

Mary Heejin Kim (aka Heejin Lim), Ph.D.

Chapter 3. A Top-Level View of Computer Function and Interconnection

How to control a computer - I

- For particular computation, a configuration of logic components can be constructed – **Hardwired system**
  - Inflexible



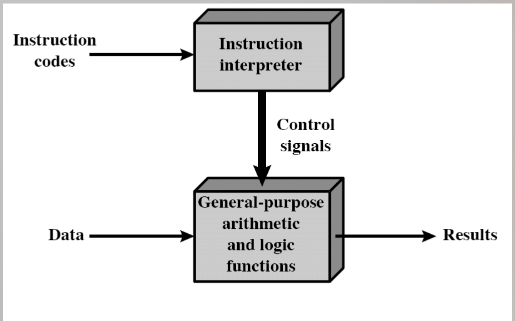
(a) Programming in hardware

  - *c.f.*, ASIC, FPGA (VHDL, Verilog)

3

How to control a computer - II

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



(b) Programming in software

4

What is a program?

- How do you supply **control signals**?
  - Write 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
- This new method of programming is called ***software*** (sequence of codes or instructions)
  - For each operation, a unique code is provided
    - e.g. ADD, MOVE
  - A hardware segment accepts the code and issues the control signals (hence, **Control Unit**)

5

Computer Components

- Contemporary computer designs are based on concepts developed by John von Neumann at the Institute for Advanced Studies, Princeton
- Referred to as the *von Neumann architecture* and is based on three key concepts:
  - 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

6

Software

- A sequence of codes or instructions
- Part of the hardware interprets each instruction and generates control signals
- Provide a new sequence of codes for each new program instead of rewiring the hardware

Major components:

- CPU
  - Instruction interpreter
  - Module of general-purpose arithmetic and logic functions
- I/O Components
  - Input module
    - Contains basic components for accepting data and instructions and converting them into an internal form of signals usable by the system
  - Output module
    - Means of reporting results

7

MEMORY

Memory address register (MAR)
- Specifies the address in memory for the next read or write

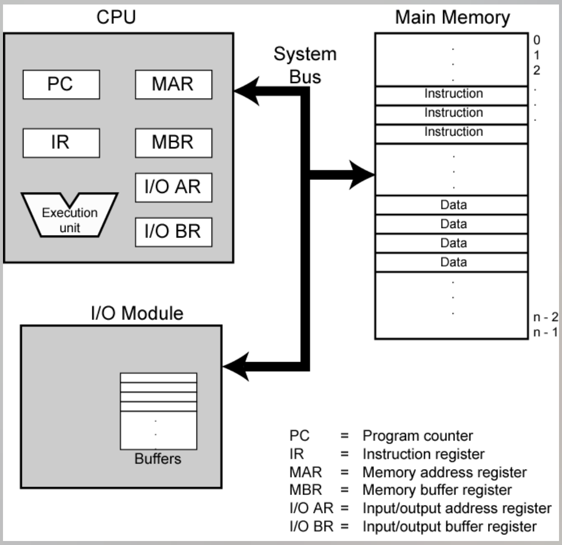
Memory buffer register (MBR)
- Contains the data to be written into memory or receives the data read from memory

I/O address register (I/OAR)
- Specifies a particular I/O device

I/O buffer register (I/OBR)
- Used for the exchange of data between an I/O module and the CPU

8

Computer Components:
Top Level View



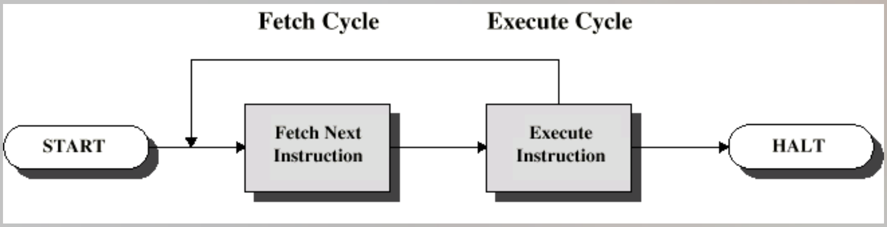
PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

9

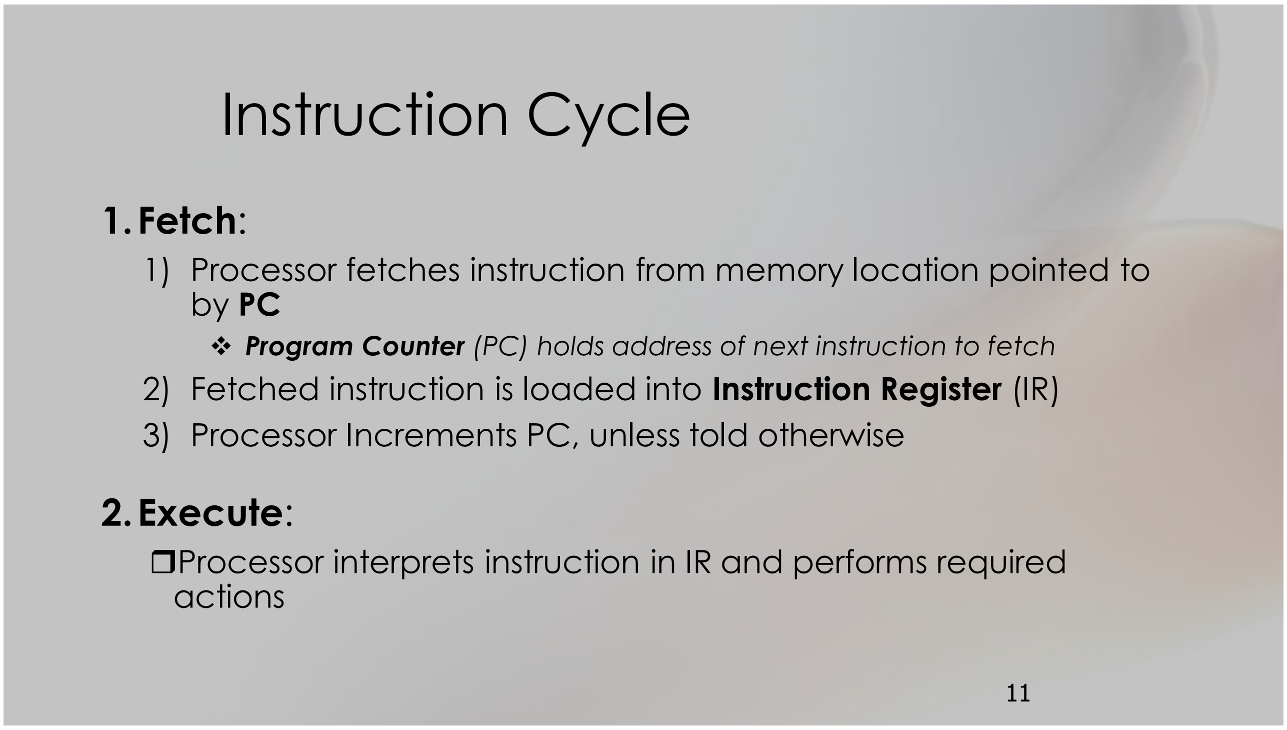
Computer Functions - Instruction Cycle

- Execution of a program in two steps:
  1.Fetch
  2.Execute

- Halts: when turned off, unrecoverable error, or by halt instruction



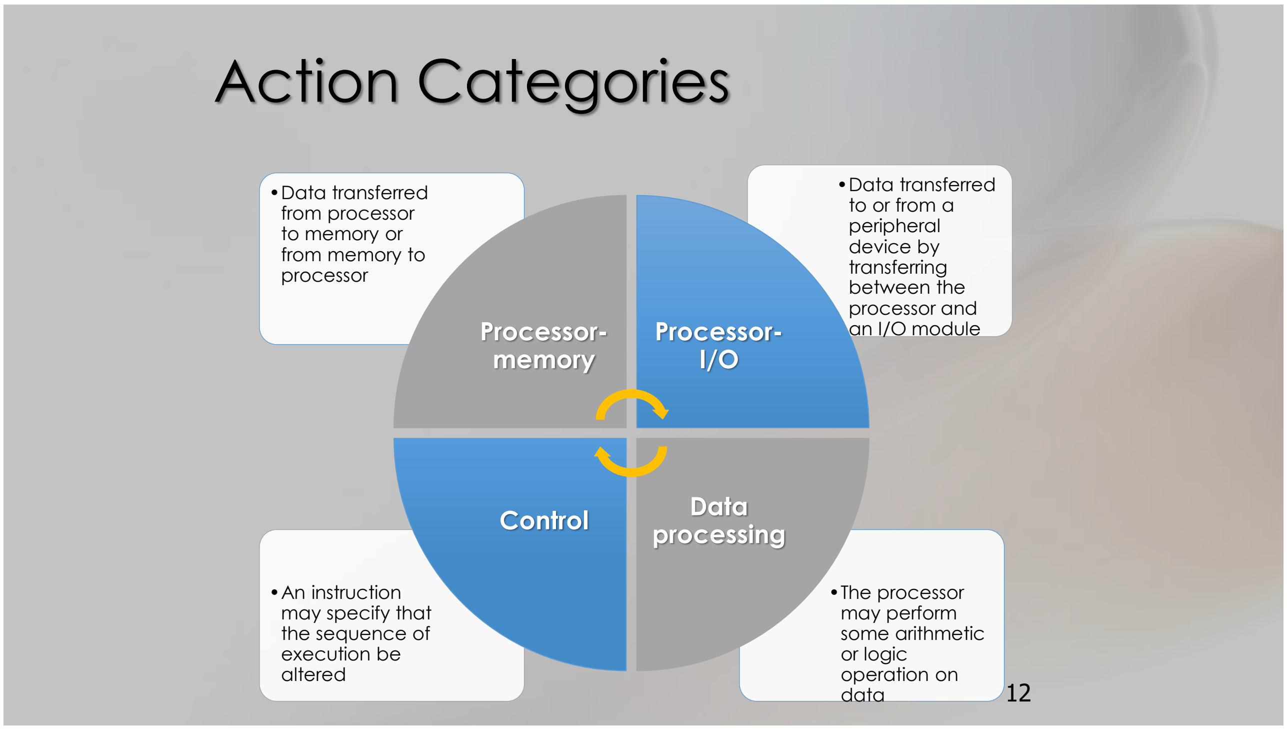
10



Instruction Cycle

1. **Fetch**:
   1) Processor fetches instruction from memory location pointed to by **PC**
      - ***Program Counter*** *(PC) holds address of next instruction to fetch*
   2) Fetched instruction is loaded into **Instruction Register** (IR)
   3) Processor Increments PC, unless told otherwise
2. **Execute**:
   - Processor interprets instruction in IR and performs required actions

11

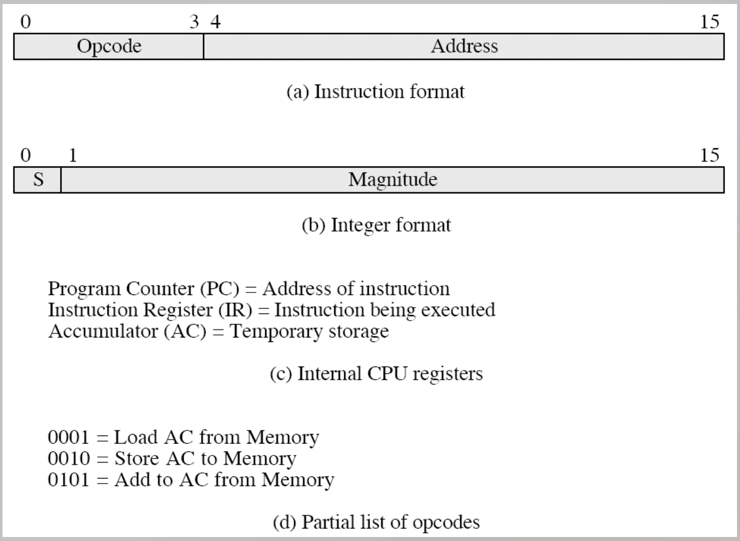


Action Categories

- Data transferred from processor to memory or from memory to processor
- Data transferred to or from a peripheral device by transferring between the processor and an I/O module
- An instruction may specify that the sequence of execution be altered
- The processor may perform some arithmetic or logic operation on data

12

Example – Hypothetical Computer



(a) Instruction format

(b) Integer format

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

(c) Internal CPU registers

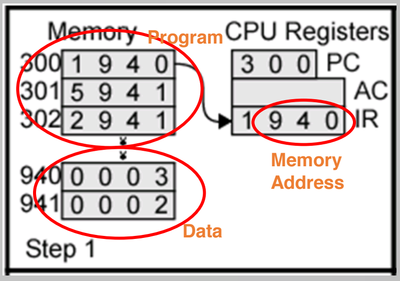
0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

(d) Partial list of opcodes

- Instruction size = data size = **16 bit**
  - So, memory = 16-bit word
- Opcode = 4 bit
  - 16 different opcodes (2^4)
- Address = 12 bits
  - 4096 words address (2^{12})
- 0010 0001 0010 1101 (= 0x212D)
  - **Store** AC content to memory address **0x12D**
  - In assembly Language,
  - e.g., : STR 12D

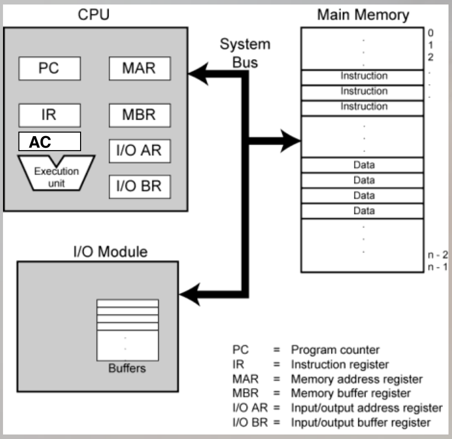
13

Example of Program Execution



Opcode:

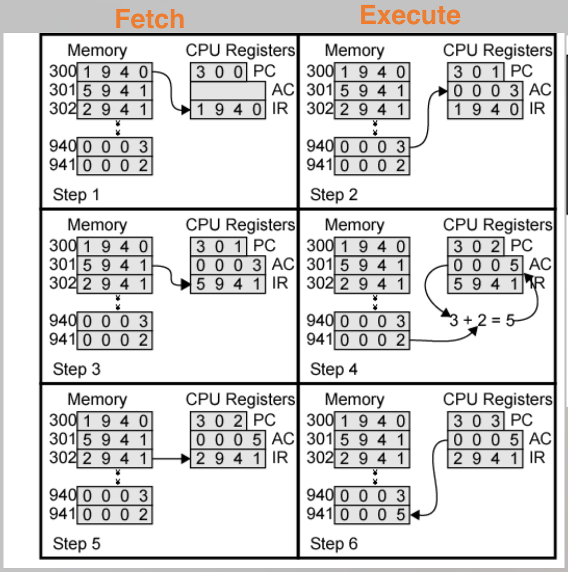
1: Load (mem → AC)
2: Store (AC → mem)
5: AC + mem → AC

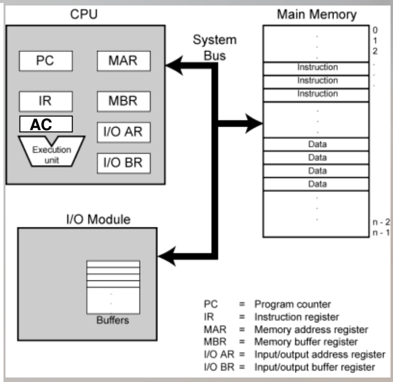


PC	=	Program counter
IR	=	Instruction register
MAR	=	Memory address register
MBR	=	Memory buffer register
I/O AR	=	Input/output address register
I/O BR	=	Input/output buffer register

14

Example of Program Execution





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

Opcode:

1: Load (mem → AC)
2: Store (AC → mem)
5: AC + mem → AC

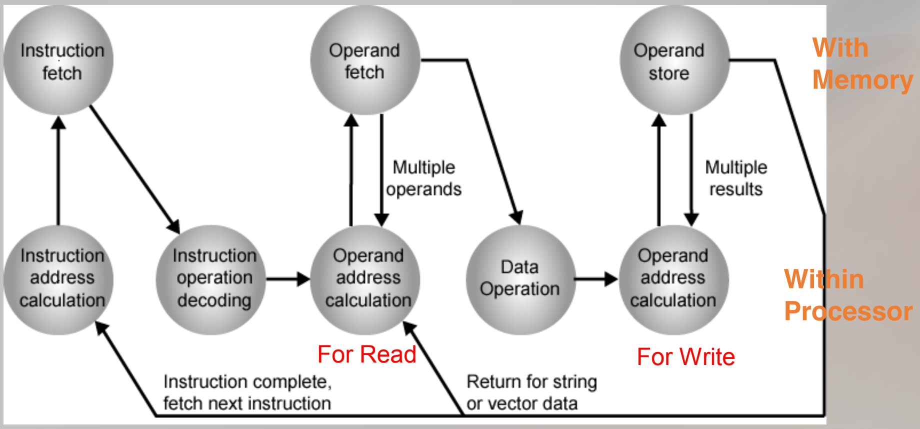
15

Execution states

- What kind of operations does the processor perform?
  - Instruction address calculation
  - Instruction fetch
  - Instruction operation decode
  - Operand address calculation (memory or I/O dev)
  - Operand fetch
  - Operand store
  - Data operation

16

Instruction Cycle State Diagram


Instruction fetch
Operand fetch
Operand store
With Memory
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Within Processor
For Read
For Write
Instruction complete, fetch next instruction
Return for string or vector data

17

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

18

Instruction Cycle with Interrupts

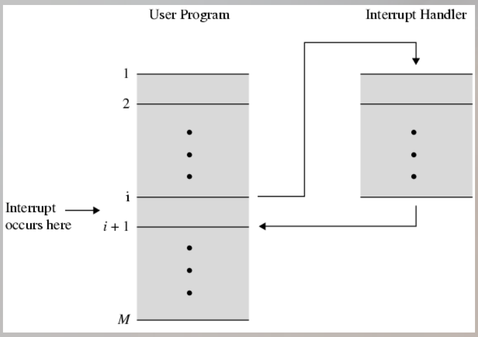
1. Processor checks for interrupt
   - Indicated by an interrupt signal
   - If no interrupt, fetch next instruction
2. If interrupt pending:
   - Process interrupt

- *Yes, there is some overhead with interrupt. But the gain is much greater.*

19

Interrupt Processing in Detail

1. **Suspend** execution of current program
2. **Save context** (the address of next instruction and any other data)
3. **Set PC to starting addr.** of interrupt handler routine
4. **Process interrupt** (i.e., execute int. handler routine)
5. **Restore context** and continue interrupted program at the point of interruption



20

Table 3.1 Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure such as power failure or memory parity error.

Other types of Interrupts

- The interrupts in the textbook (Table 3.1) are **hardware interrupts**
- A **software interrupt** is an interrupt generated within a processor by executing an instruction. E.g., system calls.
  - It causes a context switch to the interrupt handler similarly to a hardware interrupt
  - In some machines, it is called exception.
  - Difference between the two is just *who triggers the interrupt*.
- Processors typically have an internal *interrupt mask* which allows software to ignore all external hardware interrupts while it is set.
  - A ***maskable interrupt*** is essentially a hardware interrupt which may be ignored by setting a bit in an interrupt mask register.
  - A ***non-maskable interrupt*** is a hardware interrupt which typically does not have a bit-mask associated with it.

Non-Maskable Interrupt

- A special type of interrupt that **can not be ignored** by standard interrupt masking techniques
  - To signal attention for non-recoverable hardware errors
- Often used when **response time** is critical, and when an interrupt should never be disabled in the normal operation of the system.
  - internal system chipset errors
  - corruption in system memory
  - data corruption detected on system and peripheral buses

23

Interrupt Handler Routine

- Interrupt Handler routine
  - a subroutine in an Operating System or device driver whose execution is triggered by the reception of an interrupt
  - a.k.a., an *Interrupt Service Routine*
- Interrupt Vector
  - The memory address of an interrupt handler, or an index into an array called an *interrupt vector table* or *dispatch table*.
    - Interrupt vector tables contain the **memory addresses of interrupt handlers.**
  - When an interrupt is generated, the processor saves its execution state via a context switch, and begins execution of the interrupt handler at the interrupt vector

24

PC Interrupts

- Interrupt Number Assignment
  - Interrupt number 00h to 0Fh are assigned for **hardware interrupts**.
    - Computers newer than AT, have additional hardware interrupt numbers assigned from 0A0h to 0A7h. assigned
  - The rest, i.e. 10h to 0FFh are for **software interrupts**. (the maximum number of interrupts is only 0FFh or 255)
    - Trigger software interrupt by assembly inst. INT, e.g., "INT 21h"
- Example Interrupt numbers

```
INT 00h CPU: Division By Zero
INT 01h CPU: Single Step for debugging
INT 02h CPU: NMI, used i.g. by POST for memory errors
INT 03h CPU: Breakpoint for debugging
INT 04h CPU: Numeric Overflow
INT 05h CPU: Print Screen
INT 08h hardware interrupt request IRQ0:
INT 9h IRQ1: called by keyboard
INT Bh IRQ3: called by 2nd serial port COM2
INT Ch IRQ4: called by 1st serial port COM1
…..
```

25

Program Flow Control

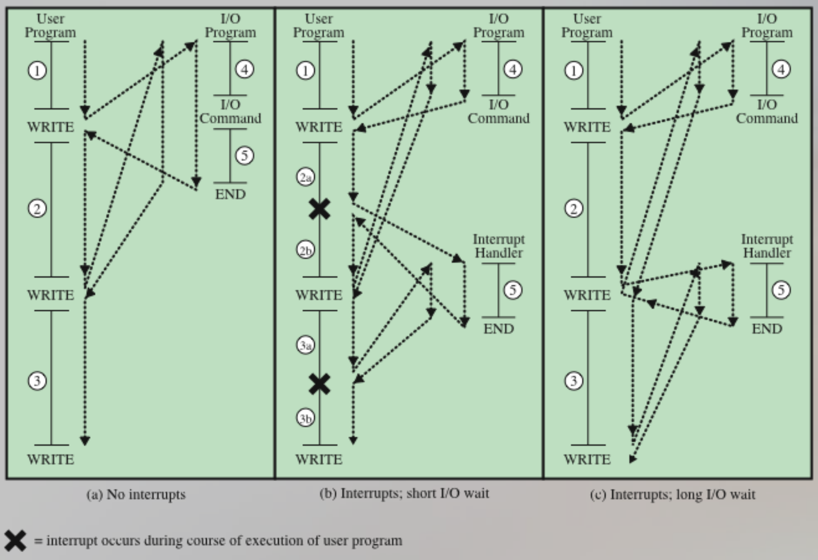
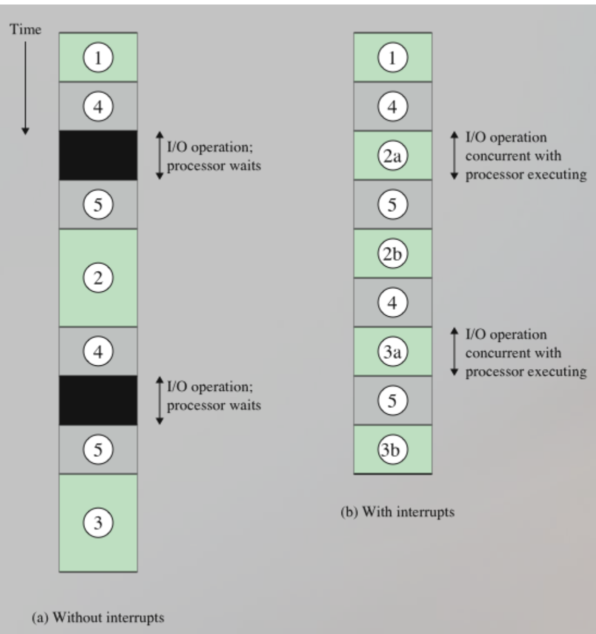

User Program
I/O Program
1
4
WRITE
I/O Command
5
END
2
WRITE
3
WRITE
2a
2b
3a
3b
Interrupt Handler
(a) No interrupts
(b) Interrupts; short I/O wait
(c) Interrupts; long I/O wait
✖ = interrupt occurs during course of execution of user program

Figure 3.7 Program Flow of Control Without and With Interrupts

26

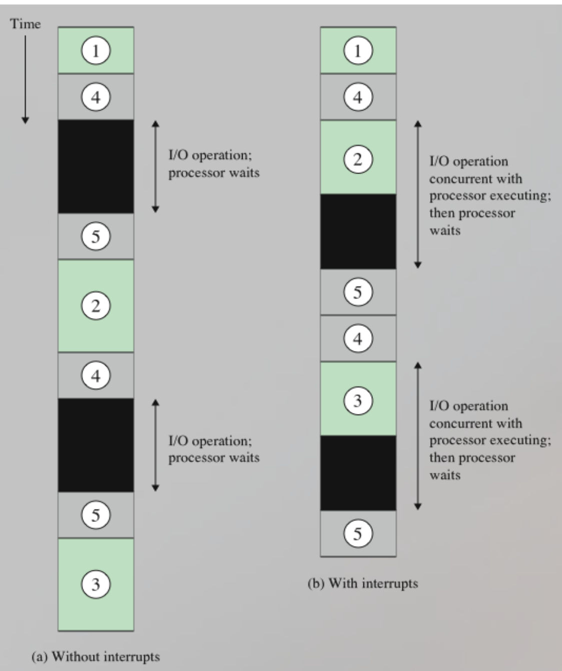


Time
1
4
I/O operation; processor waits
5
2
4
I/O operation; processor waits
5
3
(a) Without interrupts
1
4
2a
I/O operation concurrent with processor executing
5
2b
4
3a
I/O operation concurrent with processor executing
5
3b
(b) With interrupts

Figure 3.10 Program Timing: Short I/O Wait

Program Timing: Short I/O Wait

27



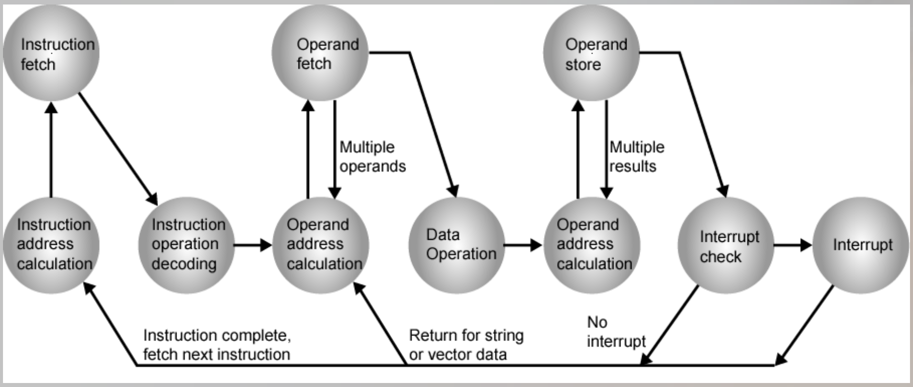
Time
1
4
I/O operation; processor waits
5
2
4
I/O operation; processor waits
5
3
(a) Without interrupts
1
4
2
I/O operation concurrent with processor executing; then processor waits
5
4
3
I/O operation concurrent with processor executing; then processor waits
5
(b) With interrupts

Figure 3.11 Program Timing: Long I/O Wait

Program Timing: Long I/O Wait

28

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

29

Instruction Cycle with Interrupts

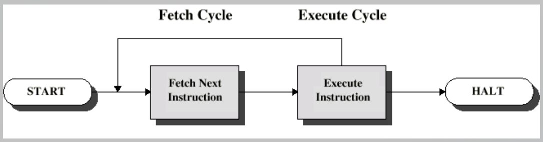
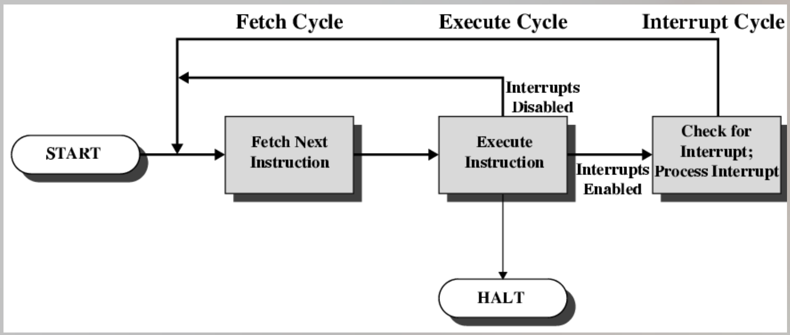


Fig 3.3. Basic Instruction Cycle



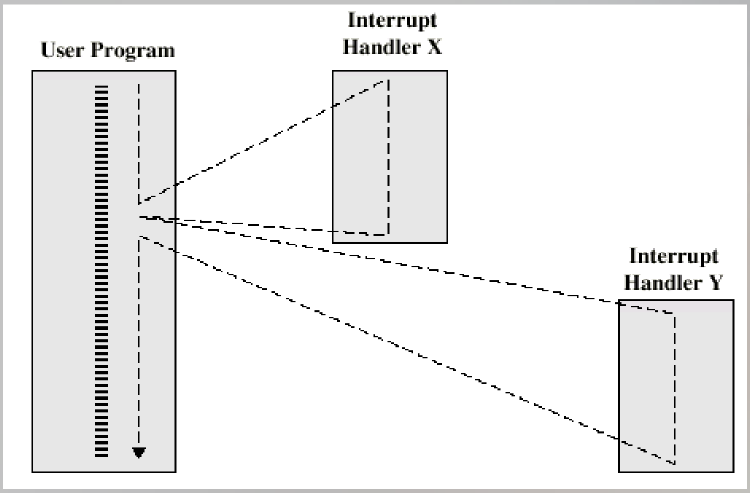
30

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

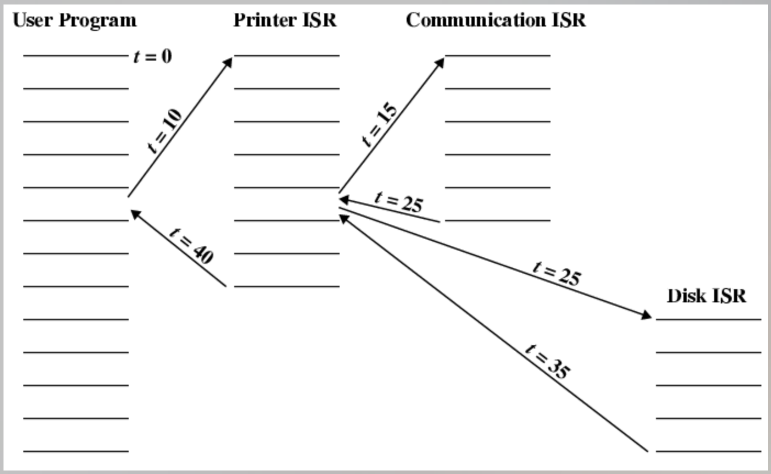
31

Multiple Interrupts - Sequential



32

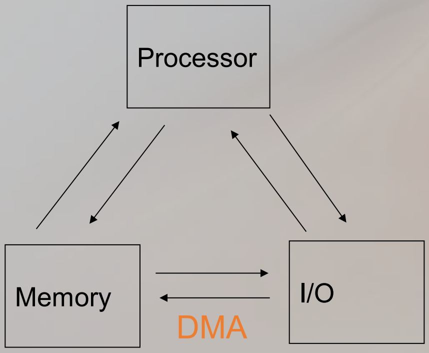
Time Sequence of Multiple Interrupts



33

Connecting

- All the units must be connected
  - The collection of paths = **interconnection structure**
- Different type of connection for different type of unit
  - Memory
  - Input/Output
  - CPU



34

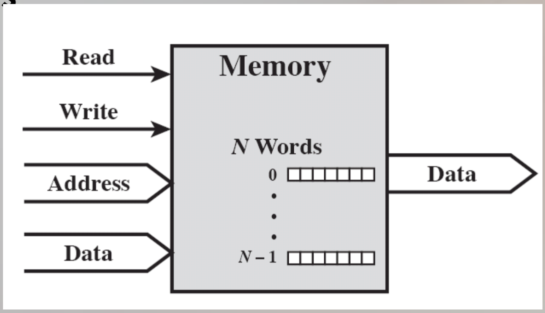
A note on I/O operation (Ch. 7)

- Processor is the king
  - Processor can read/write to I/O devices as well as to memory
- However,
  - Sometimes the I/O devices needs to read/write to memory
  - Can I/O devices do it without processor?
- DMA (Direct Memory Access)
  - The processor grants to an I/O module the authority to read from or write to memory so that the I/O memory transfer can occur without tying up the processor
  - The I/O module issues read or write commands to memory relieving the processor of responsibility for the exchange
  - This operation is known as direct memory access (DMA)

35

Memory Connection

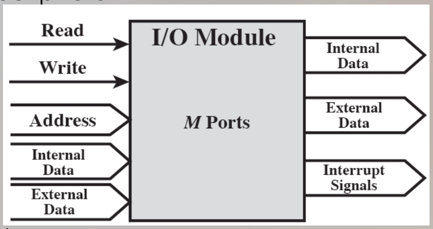
- Receives and sends **data**
- Receives **addresses** (of locations)
- Receives **control signals**
  - Read
  - Write
  - Timing



36

Input/Output Connection

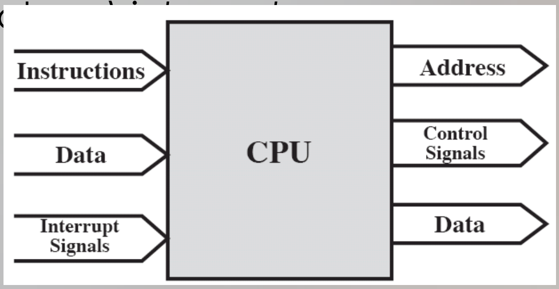
- Similar to memory from processor's viewpoint
- Controls more than 1 external device → port
- **Input**
  - Receive **control signals** from processor
  - Receive **addresses** from processor
    - e.g. port number to identify peripheral
  - Receive **data** from peripheral
  - Receive **data** from processor
- **Output**
  - Send **data** to computer
  - Send **data** to peripheral
  - Send **interrupt** signals (control)
  - Send **control signals** to peripherals
    - e.g. spin disk



37

CPU Connection

- Reads **instruction** and **data**
- Writes out **data** (after processing)
- Sends **control signals** to other units
- Receives (& ac



38

How do we connect them?

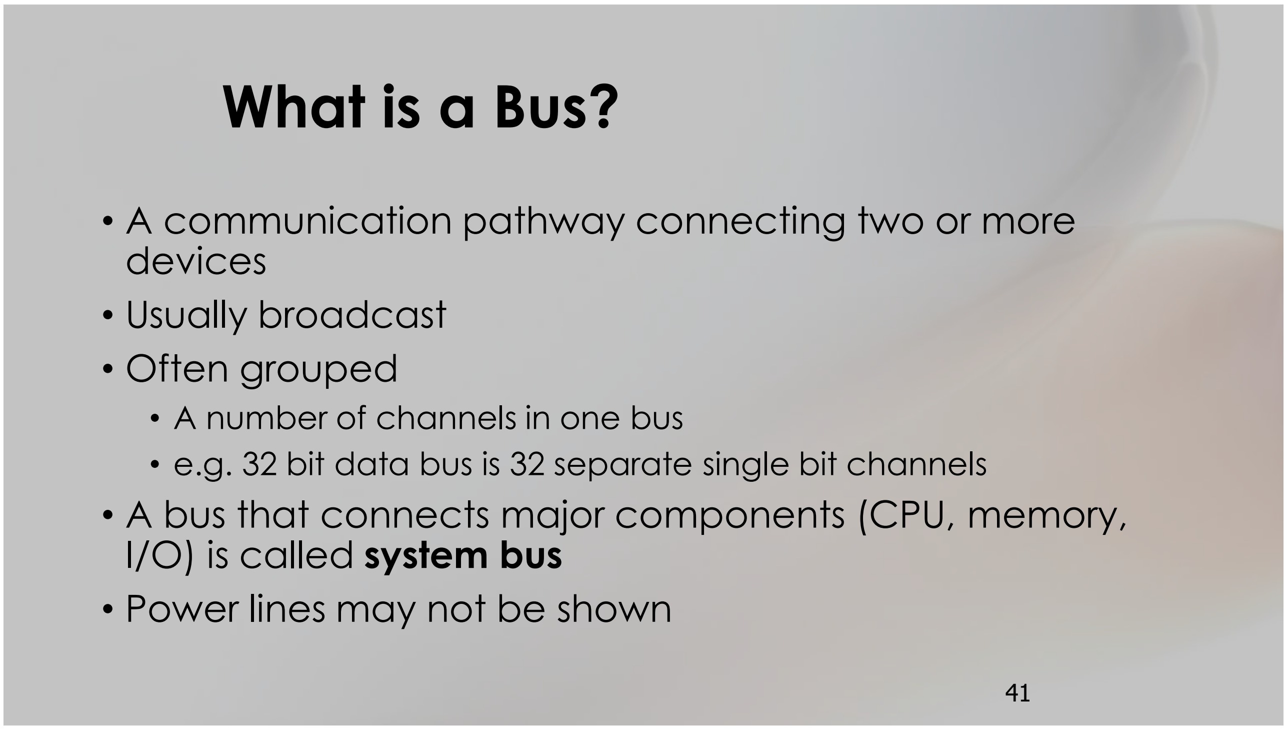
- There are a number of possible interconnection systems
- Single and multiple **bus structures** are most common
  - e.g. Unibus (DEC-PDP)
  - PCI
  - AGP
  - Front-Side Bus
- Bus: A communication pathway connecting two or more devices

39

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)

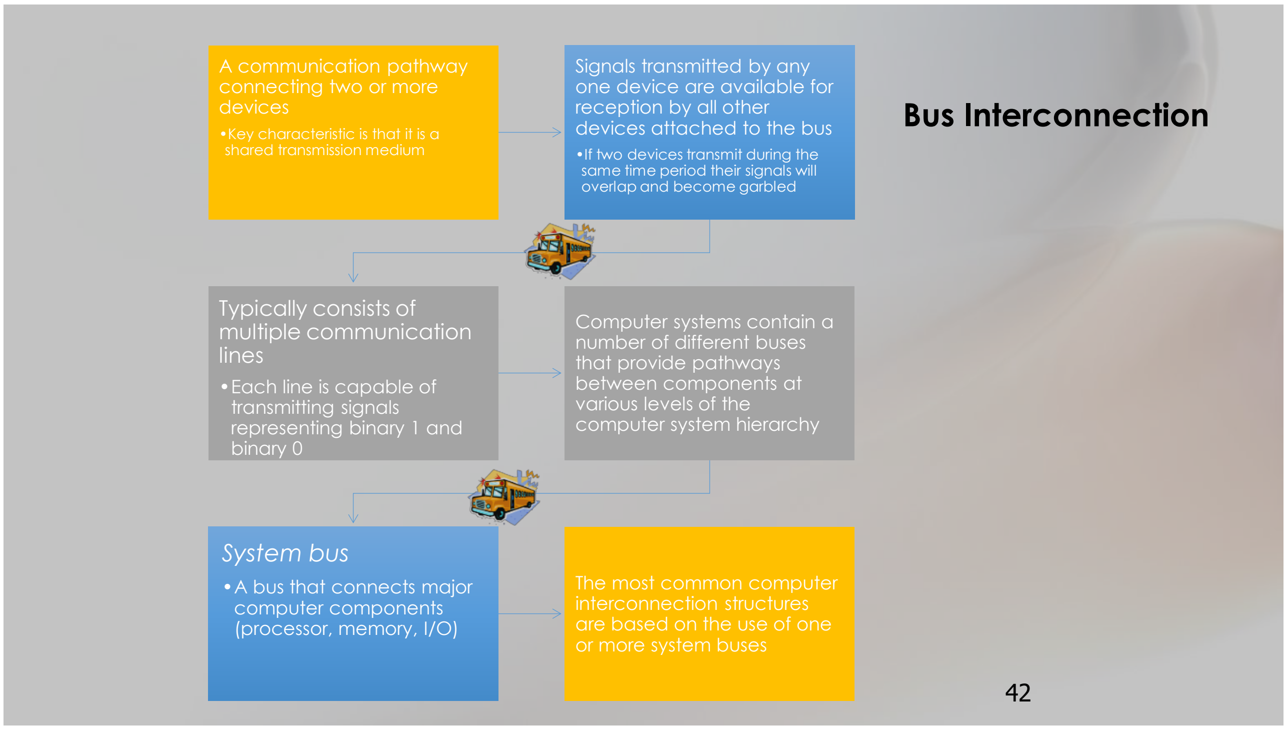
40



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
- A bus that connects major components (CPU, memory, I/O) is called **system bus**
- Power lines may not be shown

41



Bus Interconnection

A communication pathway connecting two or more devices

- Key characteristic is that it is a shared transmission medium

Signals transmitted by any one device are available for reception by all other devices attached to the bus

- If two devices transmit during the same time period their signals will overlap and become garbled

Typically consists of multiple communication lines

- Each line is capable of transmitting signals representing binary 1 and binary 0

Computer systems contain a number of different buses that provide pathways between components at various levels of the computer system hierarchy

System bus

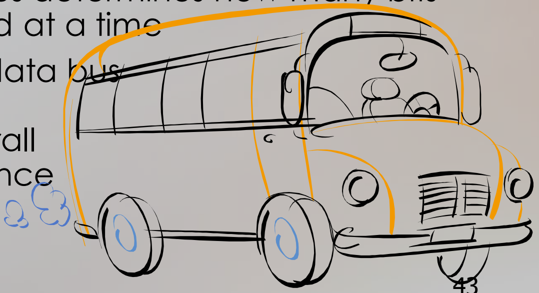
- A bus that connects major computer components (processor, memory, I/O)

The most common computer interconnection structures are based on the use of one or more system buses

42

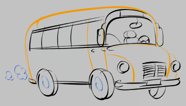
Data Bus

- Data lines that provide a path for moving data among system modules
- May consist of 32, 64, 128, or more separate lines
- The number of lines is referred to as the *width* of the data bus
- The number of lines determines how many bits can be transferred at a time
- The width of the data bus is a key factor in determining overall system performance



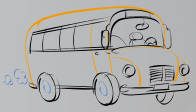
43

Address Bus



- Used to designate the source or destination of the data on the data bus
- Width determines the maximum possible memory capacity of the system
- Also used to address I/O ports
  - The higher order bits are used to select a particular module on the bus and the lower order bits select a memory location or I/O port within the module

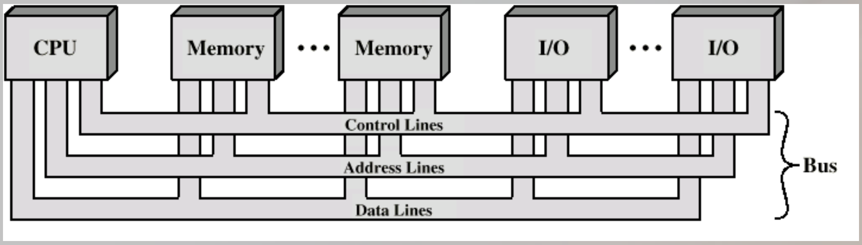
Control Bus



- Used to control the access and the use of the data and address lines
- Because the data and address lines are shared by all components there must be a means of controlling their use
- Control signals transmit both command and timing information among system modules
- Timing signals indicate the validity of data and address information
- Command signals specify operations to be performed

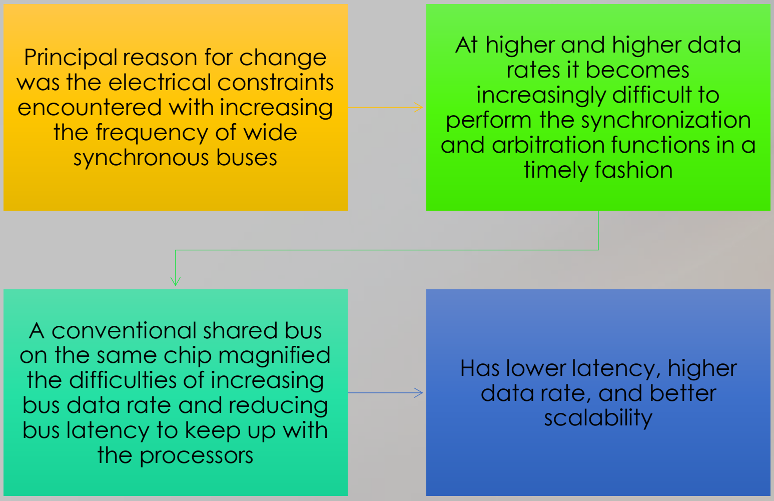
44

Bus Interconnection Scheme



45

Point-to-Point Interconnect



Principal reason for change was the electrical constraints encountered with increasing the frequency of wide synchronous buses

At higher and higher data rates it becomes increasingly difficult to perform the synchronization and arbitration functions in a timely fashion

A conventional shared bus on the same chip magnified the difficulties of increasing bus data rate and reducing bus latency to keep up with the processors

Has lower latency, higher data rate, and better scalability

46

+

Quick Path Interconnect

QPI

- Introduced in 2008
- Multiple direct connections
  - Direct pairwise connections to other components eliminating the need for arbitration found in shared transmission systems
- Layered protocol architecture
  - 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 sent as a sequence of packets each of which includes control headers and error control codes

47

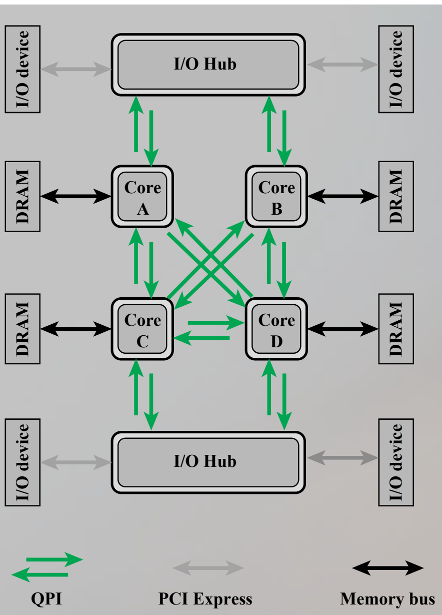


Figure 3.17 Multicore Configuration Using QPI

48

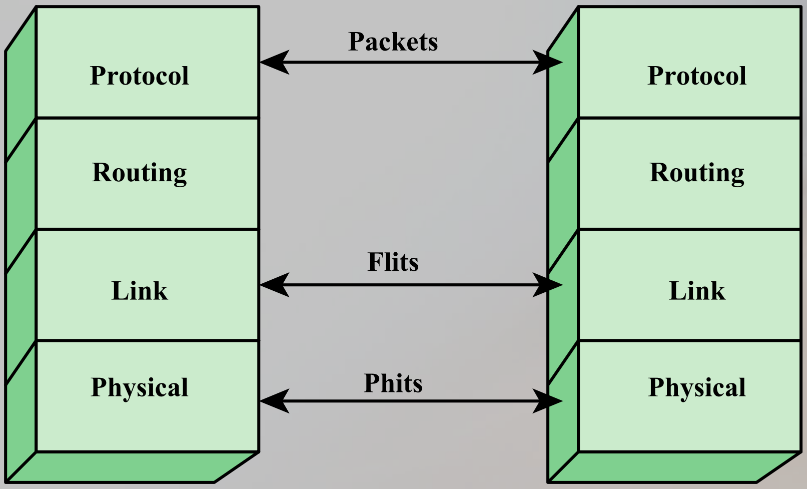


Figure 3.18 QPI Layers

49

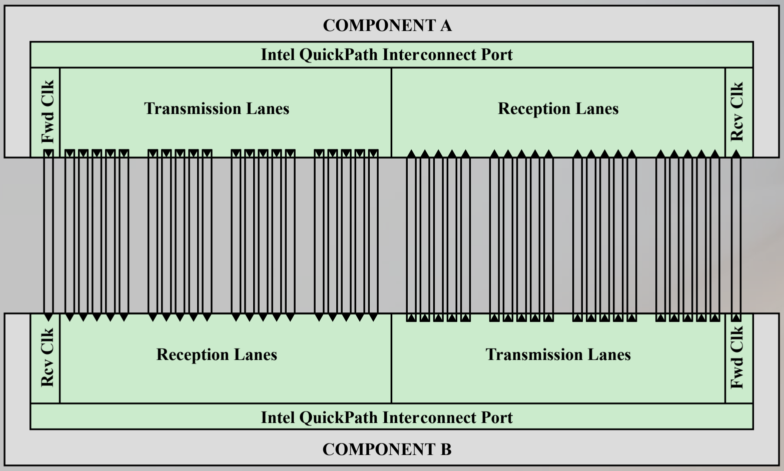


Figure 3.19 Physical Interface of the Intel QPI Interconnect

50

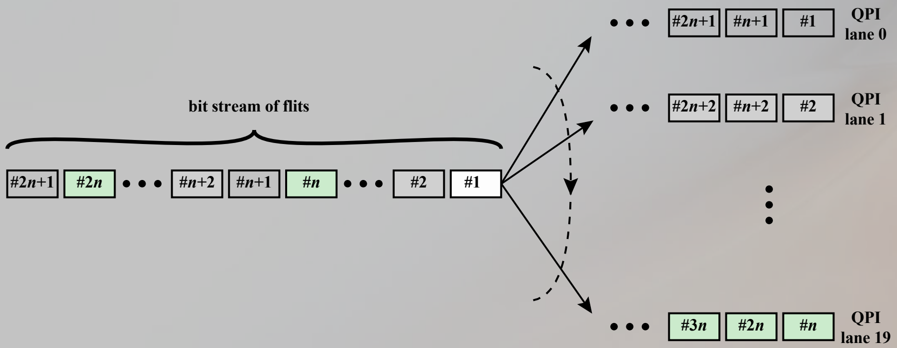


Figure 3.20 QPI Multilane Distribution

51

QPI Link Layer

- Performs two key functions: *flow control* and *error control*
  - Operate on the level of the flit (flow control unit)
  - Each flit consists of a 72-bit message payload and an 8-bit error control code called a *cyclic redundancy check* (CRC)

- Flow control function
  - 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 function
  - Detects and recovers from bit errors, and so isolates higher layers from experiencing bit errors

52

QPI Routing and Protocol Layers

Routing Layer

- Used to determine the course that a packet will traverse across the available system interconnects
- Defined by firmware and describe the possible paths that a packet can follow

Protocol Layer

- Packet is defined as the unit of transfer
- One key function performed at this level is a cache coherency protocol which deals with making sure that main memory values held in multiple caches are consistent
- A typical data packet payload is a block of data being sent to or from a cache

53

Peripheral Component Interconnect (PCI)

- A popular high bandwidth, processor independent bus that can function as a mezzanine or peripheral bus
- Delivers better system performance for high speed I/O subsystems
- PCI Special Interest Group (SIG)
    - Created to develop further and maintain the compatibility of the PCI specifications
- PCI Express (PCIe)
    - Point-to-point interconnect scheme intended to replace bus-based schemes such as PCI
    - Key requirement 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

54

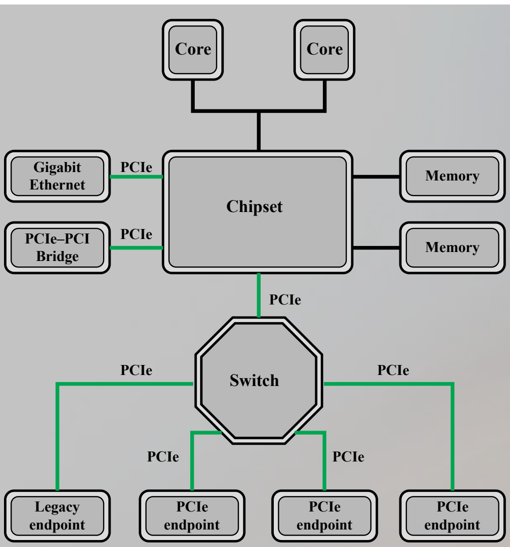


Figure 3.21 Typical Configuration Using PCIe

55

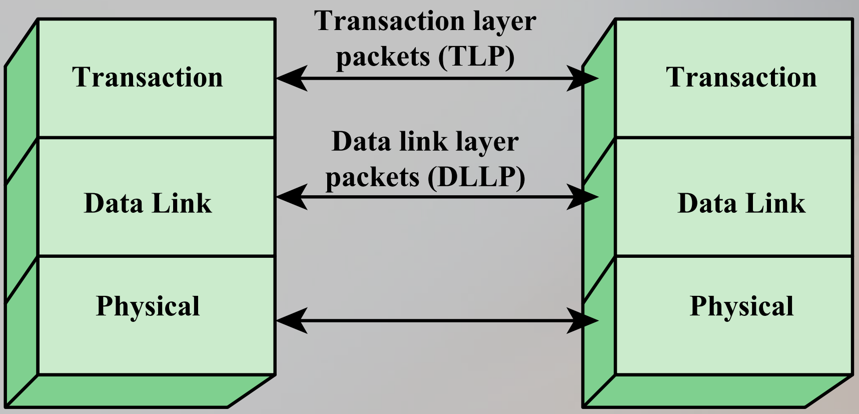

Transaction layer packets (TLP)
Data link layer packets (DLLP)
Transaction
Data Link
Physical
Transaction
Data Link
Physical

Figure 3.22 PCIe Protocol Layers

56

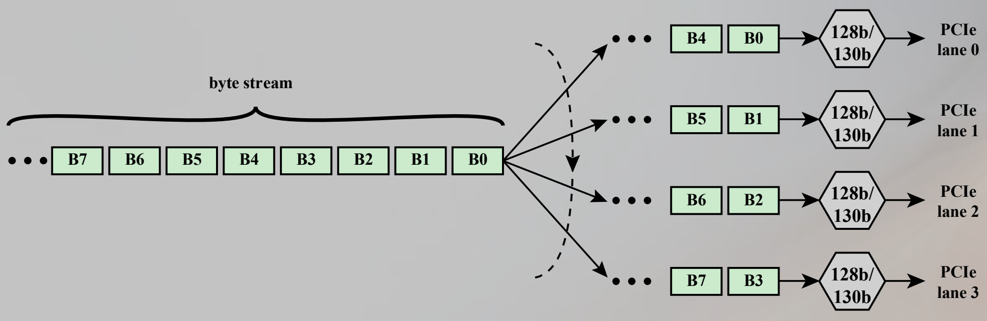


Figure 3.23 PCIe Multilane Distribution

57

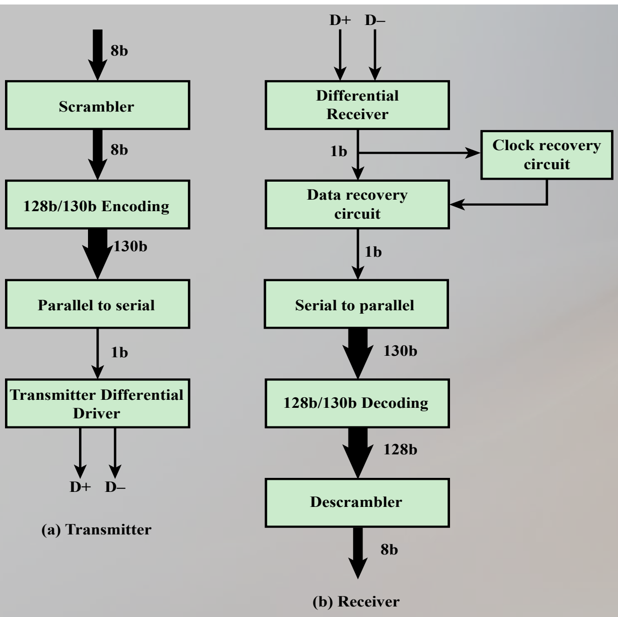


Figure 3.24 PCIe Transmit and Receive Block Diagrams

58

PCIe

Transaction Layer (TL)

- Receives read and write requests from the software above the TL and creates request packets for transmission to a destination via the link layer
- Most transactions use a *split transaction* technique
  - A request packet is sent out by a source PCIe device which then waits for a response called a *completion* packet
    - TL messages and some write transactions are posted transactions (meaning that no response is expected)
    - TL packet format supports 32-bit memory addressing and extended 64-bit memory addressing

The TL supports four address spaces:

- Memory
  - The memory space includes system main memory and PCIe I/O devices
  - Certain ranges of memory addresses map into I/O devices
- I/O
  - This address space is used for legacy PCI devices, with reserved address ranges used to address legacy I/O devices
- Configuration
  - This address space enables the TL to read/write configuration registers associated with I/O devices
- Message
  - This address space is for control signals related to interrupts, error handling, and power management

Table 3.2
PCIe TLP Transaction Types

Address Space	TLP Type	Purpose
Memory	Memory Read Request	Transfer data to or from a location in the system memory map.
	Memory Read Lock Request	
	Memory Write Request	
I/O	I/O Read Request	Transfer data to or from a location in the system memory map for legacy devices.
	I/O Write Request	
Configuration	Config Type 0 Read Request	Transfer data to or from a location in the configuration space of a PCIe device.
	Config Type 0 Write Request	
	Config Type 1 Read Request	
	Config Type 1 Write Request	
Message	Message Request	Provides in-band messaging and event reporting.
	Message Request with Data	
Memory, I/O, Configuration	Completion	Returned for certain requests.
	Completion with Data	
	Completion Locked	
	Completion Locked with Data	

61

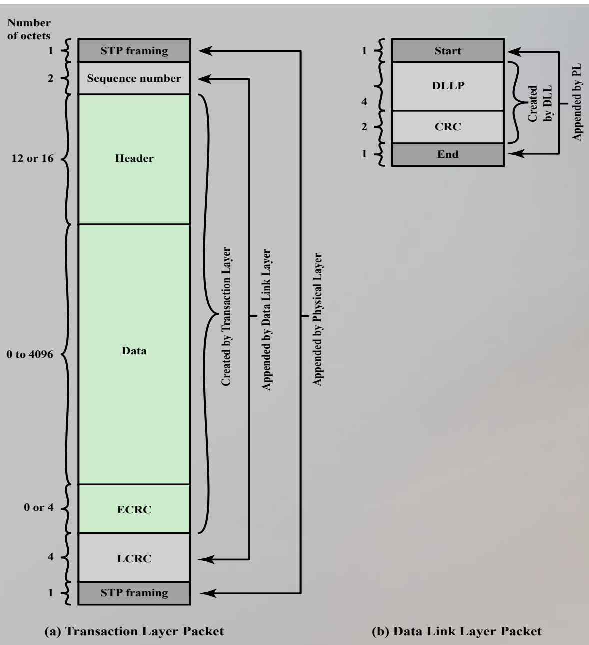
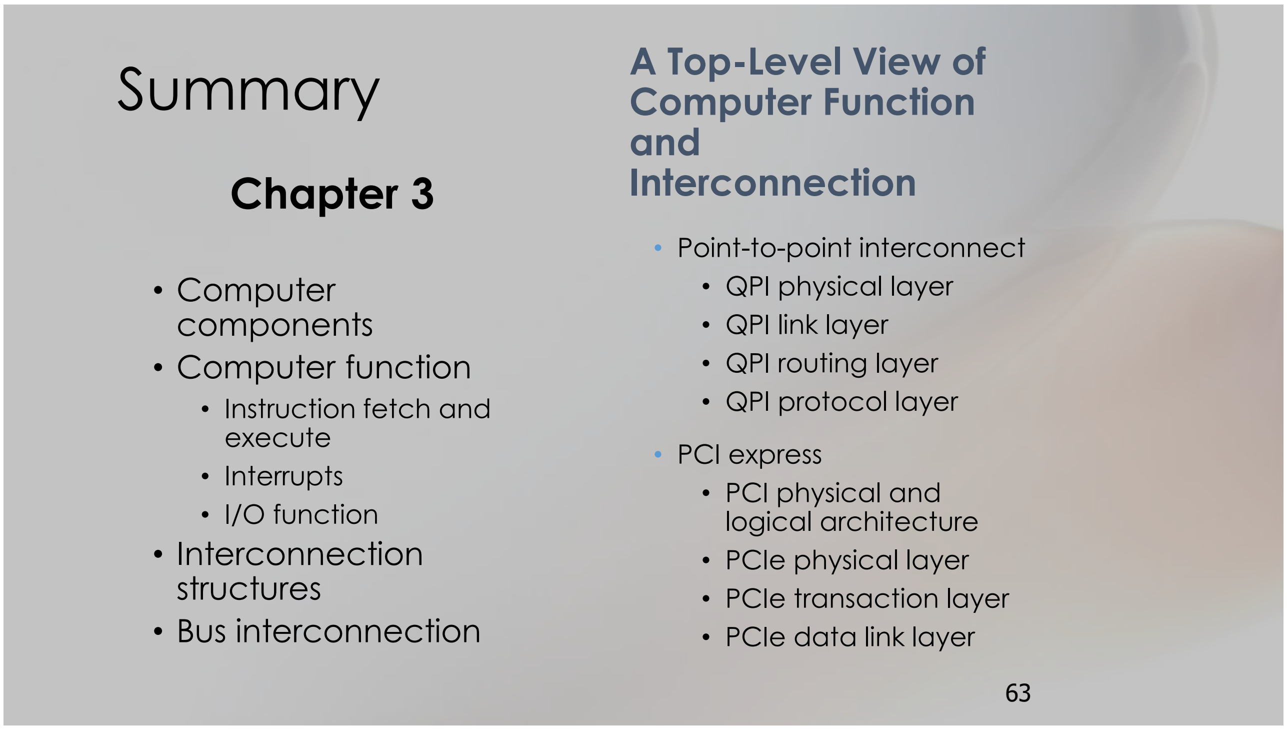


Figure 3.25 PCIe Protocol Data Unit Format

62



Summary

Chapter 3

- Computer components
- Computer function
  - Instruction fetch and execute
  - Interrupts
  - I/O function
- Interconnection structures
- Bus interconnection

A Top-Level View of Computer Function and Interconnection

- Point-to-point interconnect
  - QPI physical layer
  - QPI link layer
  - QPI routing layer
  - QPI protocol layer
- PCI express
  - PCI physical and logical architecture
  - PCIe physical layer
  - PCIe transaction layer
  - PCIe data link layer

63