COMPUTER ARCHITECTURE

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

LECTURE 1

BY MR. NDUMISO E. KHUMALO

INTRODUCTION TO COMPUTER ARCHITECTURE

Objectives

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Definition of Computer Architecture, and Organization.
- History and generation of Computers.
- Designing for performance.
- Components of the computer system.
- Bus structure and interconnection structures.
- PCI.
- CPU Organization.
- Register Organization.
- Instruction Cycle and Instruction Pipeline

Computer Architecture vs Computer Organization

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Computer Architecture (aka
  - Study of system attributes that have direct impact on the logical execution of a program.
  - Relates to functionality i.e. what a computer does as seen by programmers
  - It defines instruction formats, instruction opcodes, registers, instruction and data memory; the effect of executed instructions on the registers and memory; and an algorithm for controlling instruction execution.
- Computer Organization
  - Computer operational units and their interconnections to realize or implement the architectural specifications
  - Relates to structure and behaviour i.e. how the computer does works or completes it work and how the components interact.

Computer Architecture vs Computer Organization

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Examples of architectural attributes include :
  - the instruction set
  - # of bits used to represent various data types
  - I/O mechanisms
  - techniques for addressing memory.

- Examples of organizational attributes include those hardware details transparent to the programmer, such as:
  - control signals;
  - interfaces between the computer and peripherals; and
  - the memory technology used.

Computer Architecture vs Computer Organization

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

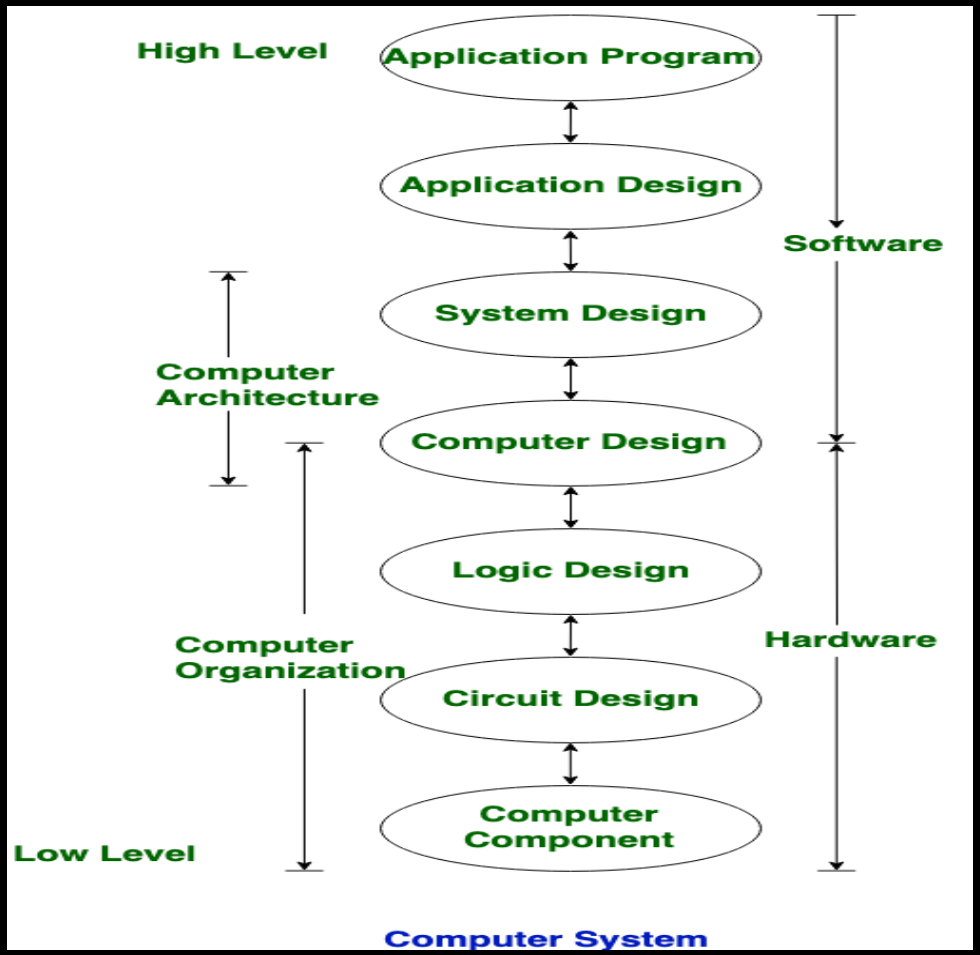
FOR EXAMPLE

- It is an architectural design issue whether a computer will have a multiply instruction.

- It is an organizational issue whether that instruction will be implemented by a special multiply unit or by a mechanism that makes repeated use of the add unit of the system.

Computer Architecture vs Computer Organization

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY



History of Computers

- Originally Used by the Governments
  - E.g. Military Applications

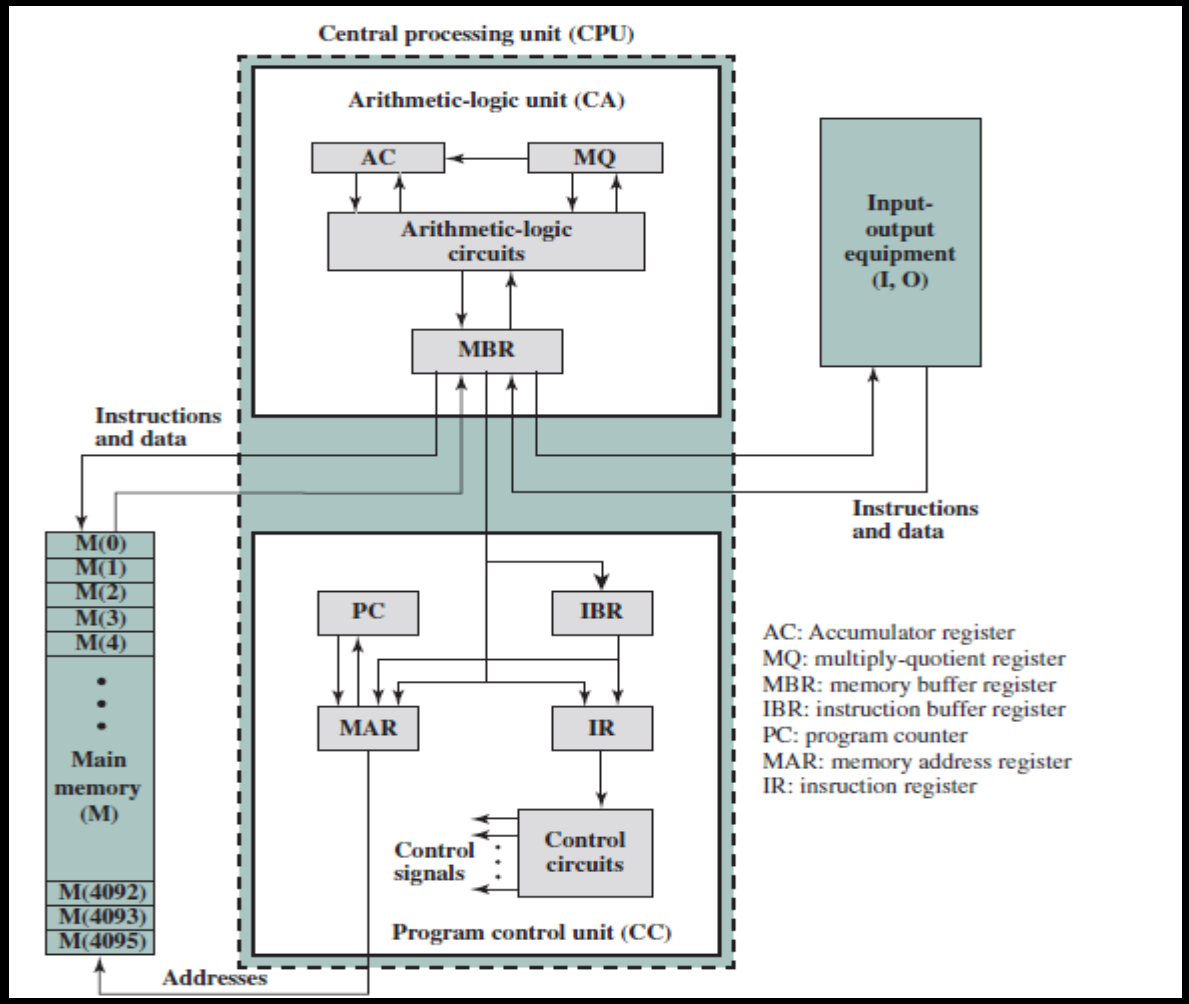
First Generation : Vacuum Tubes

- Used vacuum tubes for digital logic elements and memory
- An example is the IAS computer which implemented the *stored-program concept* developed by John von Neuman and Alan Turing

- IAS stands for Institute for Advanced Study

First Generation : Vacuum Tubes

- The structure of an IAS computer



First Generation : Vacuum Tubes

- An IAS computer consisted of :
  - 1. A main memory, which stores both data & instructions in 4096 storage locations (called *words*) of 40 bits each
    - Each number is represented by a sign bit and a 39-bit value.
    - A word may alternatively contain two 20-bit instructions, with each instruction consisting of an 8-bit operation code (opcode) specifying the operation to be performed and a 12-bit address designating one of the words in memory.

First Generation : Vacuum Tubes

- An IAS computer also consisted of :
  - 2. An arithmetic and logic unit (ALU) capable of operating on binary data
  - 3. A control unit, which interprets the instructions in memory and causes them to be executed by fetching instructions from memory and executing them one at a time
  - 4. Input–output (I/O) equipment operated by the control unit

First Generation : Vacuum Tubes

- Both the Control Unit and ALU contain storage locations, called registers, defined as follows:
  - 1. Memory buffer register (MBR): Contains a word to be stored in memory or sent to the I/O unit, or is used to receive a word from memory or from the I/O unit.
  - 2. Memory address register (MAR): Specifies the address in memory of the word to be written from or read into the MBR.
  - 3. Instruction register (IR): Contains the 8-bit opcode instruction being executed.

First Generation : Vacuum Tubes

- Both the Control Unit and ALU contain storage locations, called registers, defined as follows:
  - 4. Instruction buffer register (IBR): Employed to hold temporarily the right-hand instruction from a word in memory.
  - 5. Program counter (PC): Contains the address of the next instruction pair to be fetched from memory.
  - 6. Accumulator (AC) and multiplier quotient (MQ): Employed to hold temporarily operands and results of ALU operations.

First Generation : Vacuum Tubes

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- The IAS operates by repetitively performing an instruction cycle with each instruction cycle consisting of two sub cycles.
  - During the fetch cycle, the opcode of the next instruction is loaded into the IR & the address portion is loaded into the MAR.
  - This instruction may be taken from the IBR, or it can be obtained from memory by loading a word into the MBR, and then down to the IBR, IR, and MAR.
  - Once opcode is in the IR, the execute cycle is performed

The Second Generation: Transistors

- The transistor, which is smaller, cheaper & generates less heat replaced the vacuum tube but used the same way to construct computers

- Unlike the vacuum tube, which requires wires, metal plates, a glass capsule, and a vacuum, the transistor is a solid-state device, made from silicon.

The Second Generation: Transistors

- The second generation saw the introduction of more complex arithmetic and logic units and control units, the use of high- level programming languages, and the provision of system software with the computer

- The size of main memory, in multiples of 2^{10} 36-bit words, grew from 2k (1k = 2^{10}) to 32k words, while the time to access one word of memory, the Memory Cycle Time, fell from 30 microseconds to 1.4 microseconds.

The Second Generation: Transistors

- The number of opcodes grew from a modest 24 to 185.

- Also, over the lifetime of this series of computers, the relative speed of the CPU increased by a factor of 50

The Third Generation: Integrated Circuits

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- A single, self-contained transistor is called a discrete component.
- In the1950s and early 1960s, electronic equipment was composed largely of discrete components - transistors, resistors, capacitors, and so on.
- These components were manufactured separately, packaged in their own containers, and soldered or wired together onto Masonite-like circuit boards, which were then installed in computers.

The Third Generation: Integrated Circuits

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Whenever an electronic device called for a transistor, a little tube of metal containing a pinhead-sized piece of silicon had to be soldered to a circuit board.

- The entire manufacturing process, from transistor to circuit board, was expensive and cumbersome because some computers would consist of about 10000 transistors

3rd Generation: Integrated Circuits

- the invention of the integrated circuit in micro electronics saw the beginning of the 3rd gen. era.
- Microelectronics emphasized on the reduction of the size of digital electronic circuits.
- The basic elements of a digital computer must perform data storage, movement, processing, & control functions.
- Only 2 fundamental types of components are required: gates and memory cells.

3rd Generation: Integrated Circuits

- Here, a computer consists of gates, memory cells, and interconnections among these elements.
- The gates and memory cells are, in turn, constructed of simple electronic components, such as transistors and capacitors
- A gate is a device that implements a simple Boolean or logical function

3rd Generation: Integrated Circuits

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- The integrated circuit exploits the fact that transistors, resistors, & conductors can be fabricated from a semiconductor such as silicon.
- This led to the fabrication an entire circuit in a tiny piece of silicon rather than assemble discrete components made from separate pieces of silicon into the same circuit.
- Many transistors can be produced at the same time on a single wafer of silicon

3rd Generation: Integrated Circuits

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Equally important, transistors can be connected with a process of metallization to form circuits.
- A thin wafer of silicon is divided into a matrix of small areas, each a few millimeters square.
- The identical circuit pattern is fabricated in each area, and the wafer is broken up into chips.
- Each chip consists of many gates and/or memory cells plus a number of input and output attachment points

3rd Generation: Integrated Circuits

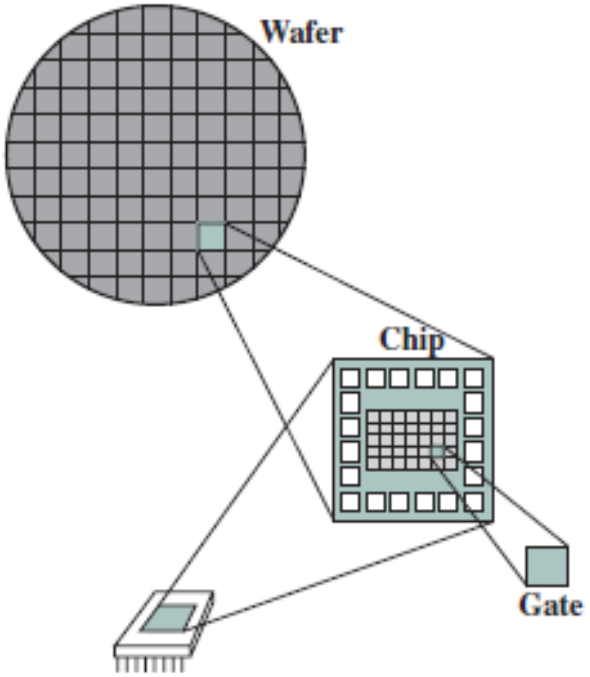


Figure 1.11 Relationship among Wafer, Chip, and Gate

Designing for performance

- the cost of computer systems continues to drop dramatically, while the performance and capacity of those systems continue to rise equally dramatically
  - Today's laptops have the computing power of an IBM mainframe from 10 or 15 years ago
  - Processors are so inexpensive that we now have microprocessors we throw away.

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

Designing for performance

- Workstation systems now support highly sophisticated engineering & scientific applications & have the capacity to support image & video applications
- In addition, businesses are relying on increasingly powerful servers to handle transaction and database processing and to support massive client/server networks that have replaced the huge mainframe computer centers of yesteryear.

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- As well, cloud service providers use massive high-performance banks of servers to satisfy high-volume, high-transaction-rate applications for a broad spectrum of clients.
- the basic building blocks for today's computer miracles are virtually the same as those of the IAS computer from over 50 years ago
  - the techniques for squeezing the maximum performance out of the materials at hand have become increasingly sophisticated

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- the driving factors behind the need to design for performance are:
  - 1. Microprocessor Speed
    - There is a relentless pursuit of speed by processor chip manufacturers
    - chipmakers can unleash a new generation of chips every three years—with four times as many transistors due to Moore's Law
    - Moore observed that the number of transistors that could be put on a single chip was doubling every year, and correctly predicted that this pace would continue into the near future

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- while the chipmakers have been busy learning how to fabricate chips of greater and greater density, the processor designers must come up with ever more elaborate techniques for feeding the monster
- These techniques include:
  - Pipelining:
    - Pipelining enables a processor to work simultaneously on multiple instructions by performing a different phase for each of the multiple instructions at the same time

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- These techniques include:
  - Branch prediction :
    - The processor looks ahead in the instruction code fetched from memory and predicts which branches, or groups of instructions, are likely to be processed next.
    - If the processor guesses right most of the time, it can prefetch the correct instructions and buffer them so that the processor is kept busy. Especially if it predicts multiple branches
    - Thus, branch prediction potentially increases the amount of work available for the processor to execute

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- These techniques include:
  - Superscalar execution
    - This is the ability to issue more than one instruction in every processor clock cycle. In effect, multiple parallel pipelines are used.
  - Data flow analysis:
    - The processor analyzes which instructions are dependent on each other's results, or data, to create an optimized schedule of instructions.
    - In fact, instructions are scheduled to be executed when ready, independent of the original program order

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- These techniques include:
  - Speculative execution
    - Using branch prediction and data flow analysis, some processors speculatively execute instructions ahead of their actual appearance in the program execution, holding the results in temporary locations.
    - This enables the processor to keep its execution engines as busy as possible by executing instructions that are likely to be needed.

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- the driving factors behind the need to design for performance are:
  - 2. Performance Balance
    - While processor power has raced ahead at breakneck speed, other critical components of the computer have not kept up
    - There is a need to look for performance balance: an adjustment/tuning of the organization and architecture to compensate for the mismatch among the capabilities of the various components
    - The problem created by such mismatches is particularly critical at the interface between processor and main memory.

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Performance Balance
  - While processor speed has grown rapidly, the speed with which data can be transferred between main memory and the processor has lagged badly.
  - If memory or the pathway fails to keep pace with the processor's insistent demands, the processor stalls in a wait state, and valuable processing time is lost

- There are ways to tackle this problem:
  - Increase the number of bits that are retrieved at one time by making DRAMs "wider" rather than "deeper" and by using wide bus data paths.

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- There are ways to tackle this problem:
  - Change the DRAM interface to make it more efficient by including a cache or other buffering scheme on the DRAM chip.
  - Reduce the frequency of memory access by incorporating increasingly complex and efficient cache structures between the processor and main memory.
    - This includes the incorporation of one or more caches on the processor chip as well as on an off-chip cache close to the processor chip.

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- There are ways to tackle this problem:
  - Increase the interconnect bandwidth between processors and memory by using higher-speed buses and a hierarchy of buses to buffer and structure data flow
- the driving factors behind the need to design for performance are:
  - As designers wrestle with the challenge of balancing processor performance with that of main memory and other computer components, the need to increase processor speed remains

Designing for performance

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- There are three approaches to achieving increased processor
  - Increase the hardware speed of the processor : shrinking the size of the logic gates on the processor chip, so that more gates can be packed together more tightly and to increasing the clock rate
  - Increase the size and speed of caches that are interposed between the processor and main memory.
  - Make changes to the processor organization and architecture that increase the effective speed of instruction execution. Typically, this involves using parallelism in one form or another

Components of the computer system

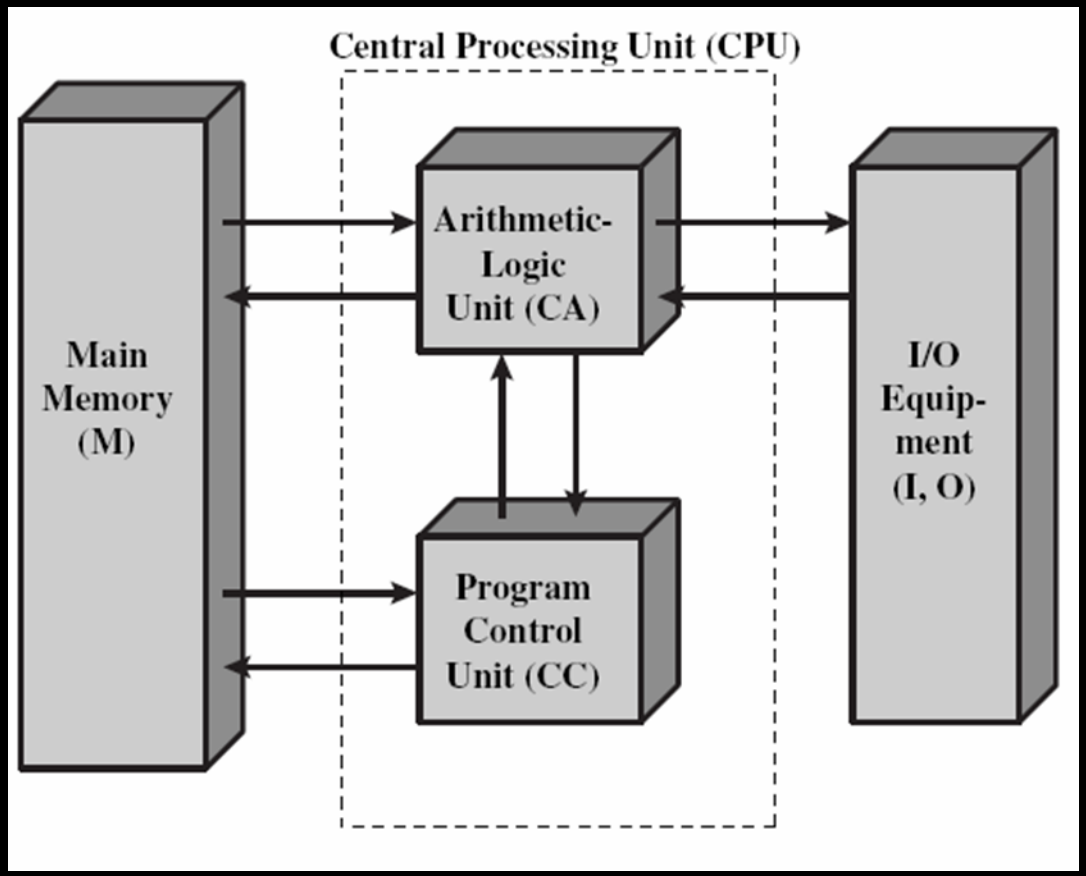
LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Virtually all contemporary computer designs are based on concepts developed by John von Neumann
- Such a design is referred to as the von Neumann architecture and is based on 3 key concepts:
  - Data & 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.

Components of the computer system

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- the von Neumann architecture



Components of the computer system

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- The functions of the different computer components are
  - ALU - performs arithmetic and logic operations
    - e.g., adders, multipliers, shifters
  - memory - holds data and instructions
    - e.g., cache, main memory, disk
  - input - sends data to the computer
    - e.g., keyboard, mouse

Components of the computer system

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- The functions of the different computer components are
  - output - gets data from the computer
    - e.g., screen, sound card
  - Control unit - gives directions to the other components
    - e.g., bus controller, memory interface unit

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- A computer consists of a set of components or modules of 3 basic types (processor, memory, I/O) that communicate with each other
- In effect, a computer is a network of basic modules.
  - Thus, there must be paths for connecting the modules.
- The collection of paths connecting the various modules is called the interconnection structure
  - The design of this structure will depend on the exchanges that must be made among modules

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Over the years, a number of interconnection structures have been tried.

- By far the most common are
  - (1) the bus and various multiple-bus structures, and
  - (2) point-to-point interconnection structures with packetized data transfer.

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Bus Structure
  - The bus was the dominant means of computer system component interconnection for decades.
  - For general-purpose computers, it has gradually given way to various point-to-point interconnection structures, which now dominate computer system design.
  - However, bus structures are still commonly used for embedded systems, particularly microcontrollers..

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Bus Structure
  - A bus is 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
  - If two devices transmit during the same time period, their signals will overlap and become garbled. Thus, only one device at a time can successfully transmit

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

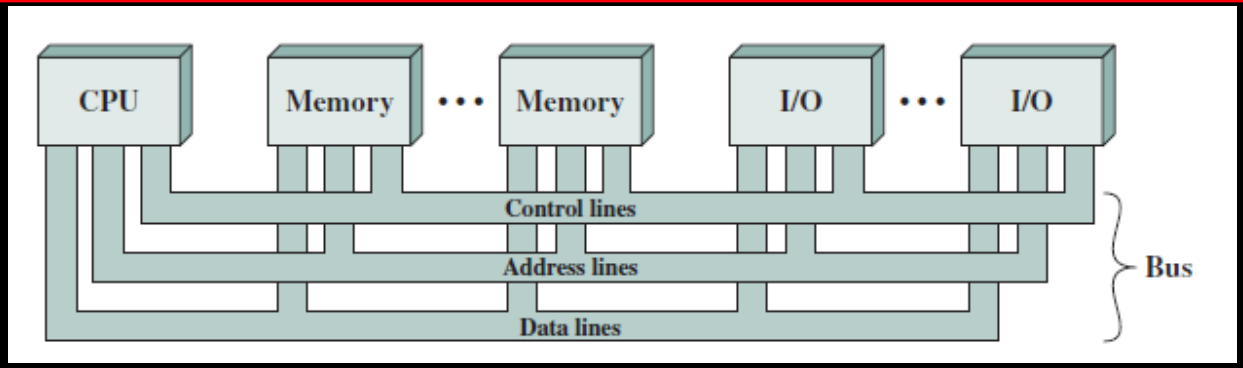
- Bus Structure
  - Typically, a bus consists of multiple communication pathways, or lines with each line capable of transmitting signals representing binary 1 and binary 0.
  - Over time, a sequence of binary digits can be transmitted across a single line
  - Taken together, several lines of a bus can be used to transmit binary digits simultaneously in parallel
    - E.g. an 8-bit unit of data can be transmitted over eight bus lines.

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Bus Structure
  - Computer systems contain a number of different buses that provide pathways between components at various levels of the computer system hierarchy.
  - A bus that connects major computer components (processor, memory, I/O) is called a ***system bus***.
  - on any bus the lines can be classified into three functional groups: data, address, and control lines

Bus structure and interconnection structures,



– The data lines provide a path for moving data among system modules. These lines, collectively, are called the data bus

- The data bus may consist of 32, 64, 128, or even more separate lines, the number of lines being referred to as the width of he data bus
- Because each line can carry only one bit at a time, the number of lines determines how many bits can be transferred at a time

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- The address lines are used to designate the source or destination of the data on the data bus

  - For example, if the processor wishes to read a word (8, 16, or 32 bits) of data from memory, it puts the address of the desired word on the address lines.
  - Clearly, the width of the address bus determines the maximum possible memory capacity of the system
  - Furthermore, the address lines are generally also used to address I/O ports

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

– The control lines are used to control the access to 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

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

– Typical control commands include:

- **Memory write:** causes data on the bus to be written into the addressed location.
- **Memory read:** causes data from the addressed location to be placed on the bus.
- **I/O write:** causes data on the bus to be output to the addressed I/O port.
- **I/O read:** causes data from the addressed I/O port to be placed on the bus.
- **Transfer ACK:** indicates that data have been accepted from or placed on the bus.
- **Bus request:** indicates that a module needs to gain control of the bus.
- **Bus grant:** indicates that a requesting module has been granted control of the bus.
- **Interrupt request:** indicates that an interrupt is pending.
- **Interrupt ACK:** acknowledges that the pending interrupt has been recognized.
- **Clock:** is used to synchronize operations.
- **Reset:** initializes all modules.

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

The operation of the bus is as follows.

- If one module wishes to send data to another, it must do two things:
  - (1) obtain the use of the bus, and
  - (2) transfer data via the bus.
- If one module wishes to request data from another module, it must
  - (1) obtain the use of the bus, and
  - (2) transfer a request to the other module over the appropriate control and address lines. It must then wait for that second module to send the data.

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Point-to-point interconnect
  - New systems no longer use the shared bus structure but the point-to-point interconnection
  - 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 buses
    - At higher and higher data rates, it becomes increasingly difficult to perform the synchronization and arbitration functions in a timely fashion

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Point-to-point interconnect
  - Compared to the shared bus, the point-to-point interconnect has lower latency, higher data rate, & better scalability.
  - One example of the point-to-point interconnect approach: Intel's QuickPath Interconnect (QPI).

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

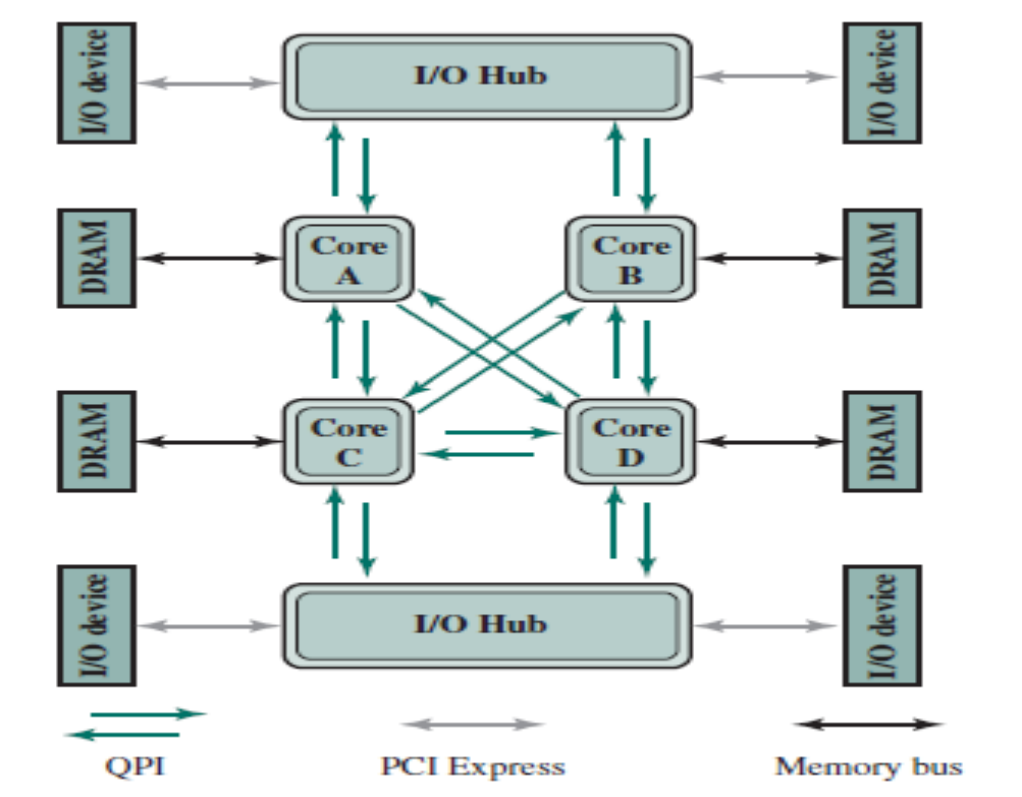
- Main characteristics of QPI and other point-to-point

- **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.

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- Main characteristics of QPI and other point-to-point



Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

- The QPI links (indicated by the green arrow pairs in the figure) form a switching fabric that enables data to move throughout the network
- Direct QPI connections can be established between each pair of core processors
- If core A needs to access the memory controller in core B, it sends its request through either cores D or C, which must in turn forward that request on to the memory controller in core B.

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

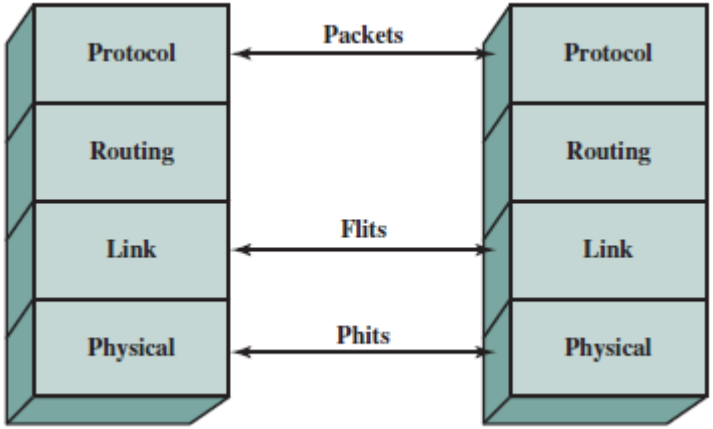
- In addition, QPI is used to connect to an I/O module, called an I/O hub (IOH).
  - The IOH acts as a switch directing traffic to and from I/O deviceS
- In newer systems, the link from the IOH to the I/O device controller uses an interconnect technology called PCI Express (PCIe),
- QPI is defined as a four-layer protocol architecture and has the following layers

Bus structure and interconnection structures,

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

– 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.



PCI

- PCI Express
  - 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).

PCI

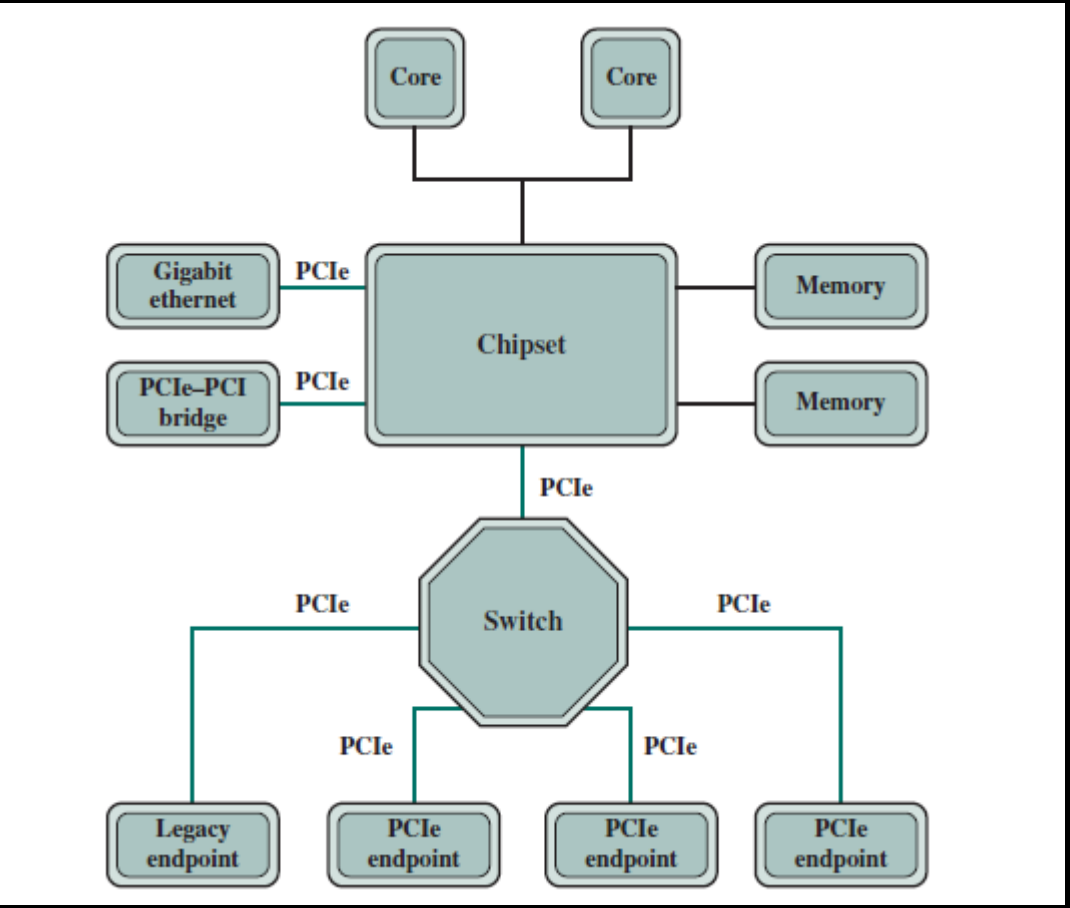
- PCI Express
  - As with the system bus discussed in the preceding sections, the bus-based PCI scheme has not been able to keep pace with the data rate demands of attached devices.
  - A new version, known as PCI Express (PCIe) has been developed.
  - PCIe, as with QPI, is a point-to-point interconnect scheme intended to replace bus-based schemes such as PCI.

PCI

- PCI Express
  - 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

PCI

- Typical Configuration Using PCIe



PCI

- Typical Configuration Using PCIe
  - A root complex device, also referred to as 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
  - The root complex acts as a buffering device, to deal with difference in data rates between I/O controllers & memory and processor components.

PCI

- Typical Configuration Using PCIe
  - The root complex also translates between PCIe transaction formats and the processor and memory signal and control requirements
  - The chipset will typically support multiple PCIe ports, some of which attach directly to a PCIe device, and one or more that attach to a switch that manages multiple PCIe streams.

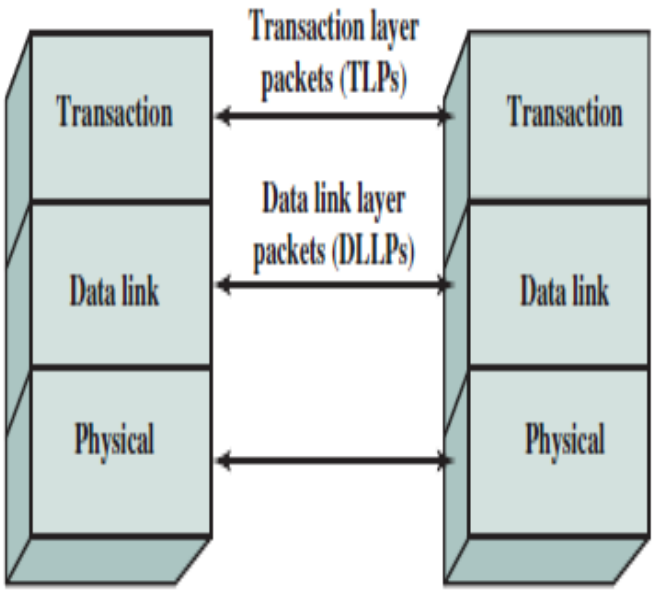
PCI

- Typical Configuration Using PCIe
  - PCIe links from the chipset may attach to the following kinds of devices that implement 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: intended for existing designs that have been migrated to PCI Express, & it allows legacy behaviors such as use of I/O space & locked transactions
    - PCIe/PCI bridge: Allows older PCI devices to be connected to PCIe-based systems.

PCI

- As with QPI, PCIe interactions are defined using a protocol architecture.
- The PCIe protocol architecture encompasses the following 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).



Register Organization

- a computer system employs a memory hierarchy.
- At higher levels of the hierarchy, memory is faster, smaller, and more expensive (per bit)
- Within the processor, there is a set of registers that function as a level of memory above main memory and cache in the hierarchy

Register Organization

- The registers in the processor perform two roles:
  - User-visible registers: Enable the machine-or assembly language programmer to minimize main memory references by optimizing use of registers.
  - Control and status registers: Used by the control unit to control the operation of the processor and by privileged operating system programs to control the execution of programs.

Register Organization

- User-Visible Registers
  - A user- visible register is one that may be referenced by means of the machine language that the processor executes.
  - We can characterize these in the following categories:
    - General-purpose registers can be assigned to a variety of functions by the programmer. Sometimes their use within the instruction set is orthogonal to the operation. That is, any general- purpose register can contain the operand for any opcode

Register Organization

- Data registers may be used only to hold data and cannot be employed in the calculation of an operand address

- Address registers may themselves be somewhat general purpose, or they may be devoted to a particular addressing mode.

- Condition codes (also referred to as flags), , which are at least partially visible to the user are bits set by the processor hardware as the result of operations.

Register Organization

- Control and Status Registers
  - There are a variety of processor registers that are employed to control the operation of the processor.
  - Most of these, on most machines, are not visible to the user.
  - Some of them may be visible to machine instructions executed in a control or operating system mode.
  - different machines will have different register organizations and use different terminology

Register Organization

- Control and Status Registers
  - Four registers are essential to instruction execution:
    - Program counter (PC): Contains the address of an instruction to be fetched.
    - Instruction register (IR): Contains the instruction most recently fetched.
    - Memory address register (MAR): Contains the address of a location in memory.
    - Memory buffer register (MBR): Contains a word of data to be written to memory or the word most recently read.

Register Organization

- Control and Status Registers
  - Many processor designs include a register or set of registers, often known as the program status word (PSW), that contain status information.
  - The PSW typically contains condition codes plus other status information
  - Common fields or flags include the following:
    - Sign: Contains the sign bit of the result of the last arithmetic operation.
    - Zero: Set when the result is 0.

Register Organization

- Carry: Set if an operation resulted in a carry (addition) into or borrow (subtraction) out of a high- order bit. Used for multiword arithmetic operations.
- Equal: Set if a logical compare result is equality.
- Overflow: Used to indicate arithmetic overflow.
- Interrupt Enable/Disable: Used to enable or disable interrupts.

- Supervisor: Indicates whether the processor is executing in supervisor or user mode. Certain privileged instructions can be executed only in supervisor mode, and certain areas of memory can be accessed only in supervisor mode.

Register Organization

– Example Microprocessor Register Organizations

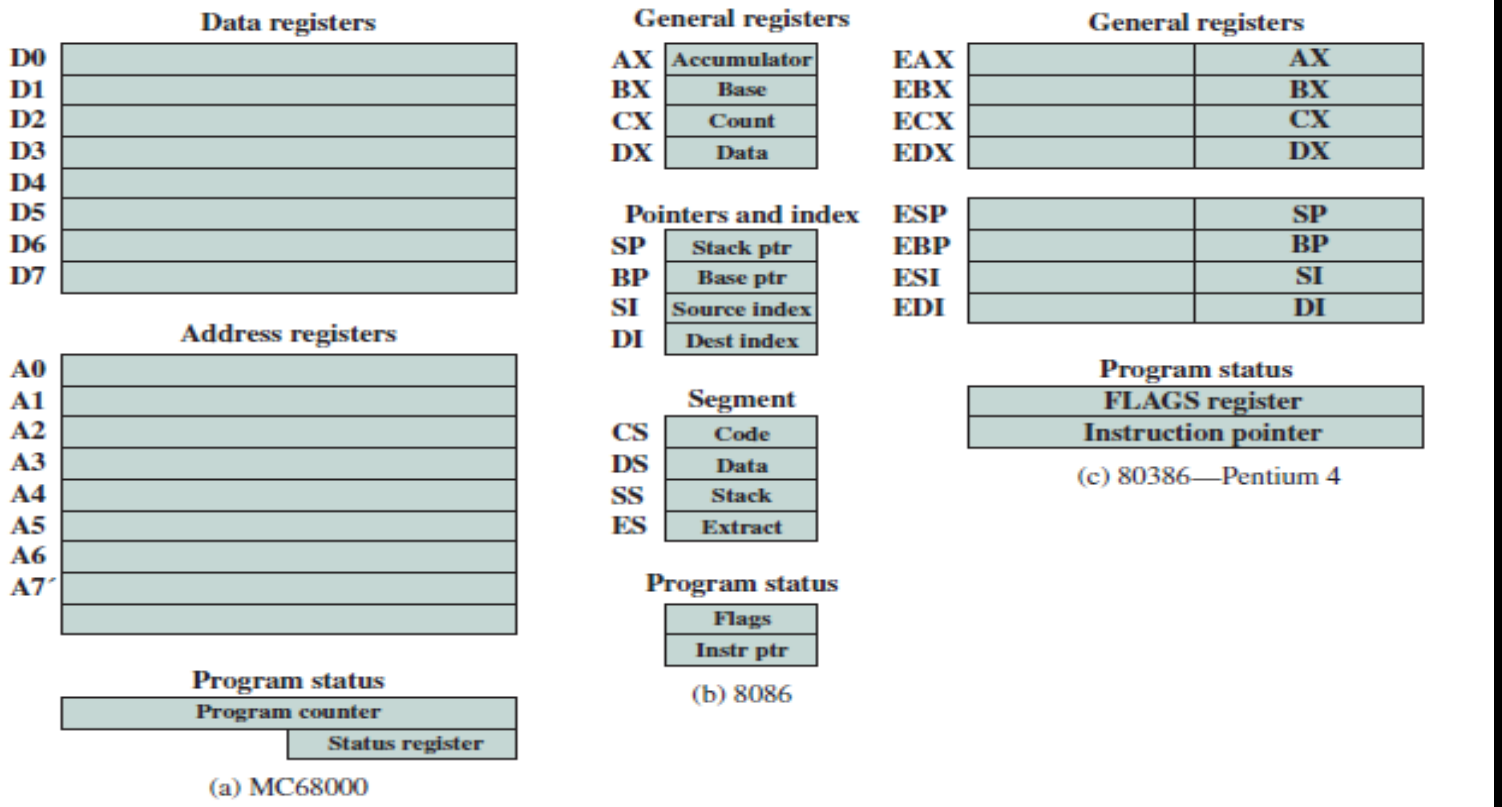


Figure 14.3 Example Microprocessor Register Organizations

Instruction Cycle

- an instruction cycle includes the following stages:
  - Fetch: Read the next instruction from memory into the processor.
  - Execute: Interpret the opcode and perform the indicated operation.
  - Interrupt: If interrupts are enabled and an interrupt has occurred, save the current process state and service the interrupt.

Instruction Cycle

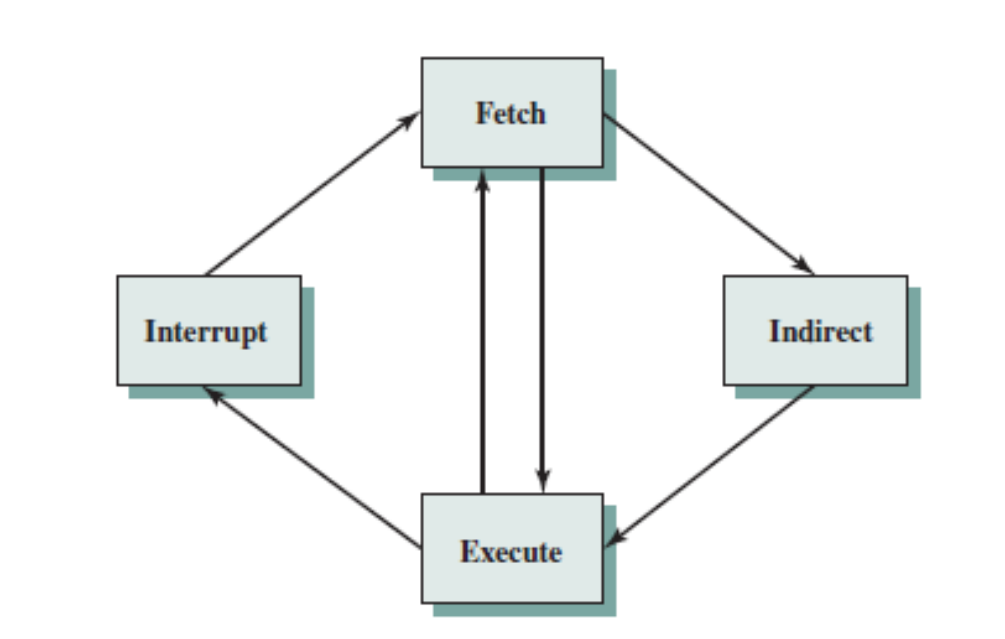
- There is one additional stage, known as the indirect cycle
  - the execution of an instruction may involve one or more operands in memory, each of which requires a memory access
  - Further, if indirect addressing is used, then additional memory accesses are required.
  - We can think of the fetching of indirect addresses as one more instruction stage

Instruction Cycle

- The main line of activity consists of alternating instruction fetch and instruction execution activities.
- After an instruction is fetched, it is examined to determine if any indirect addressing is involved
- If so, the required operands are fetched using indirect addressing.
- Following execution, an interrupt may be processed before the next instruction fetch.

Instruction Cycle

- The Instruction Cycle



Instruction Cycle

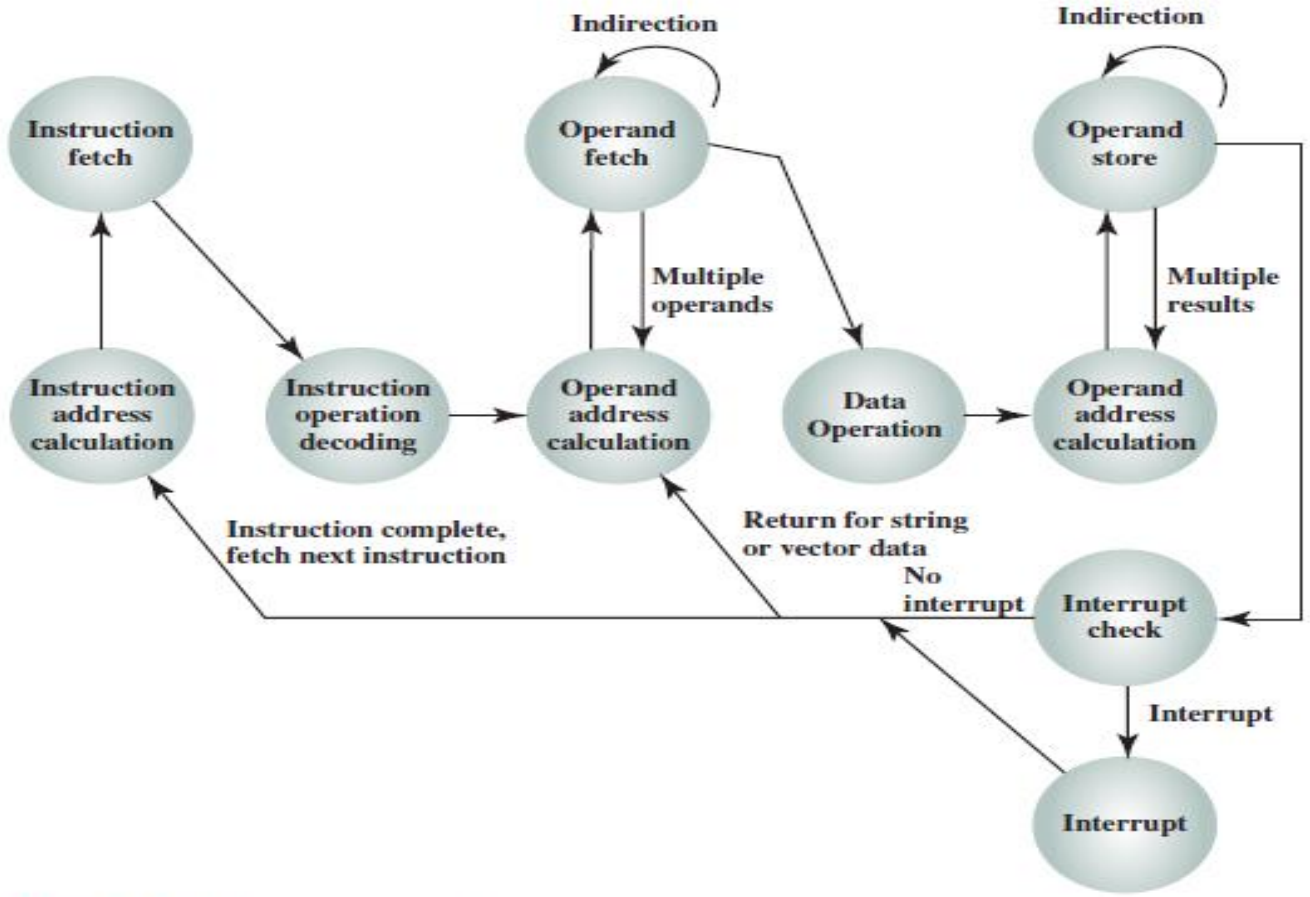


Figure 14.5 Instruction Cycle State Diagram

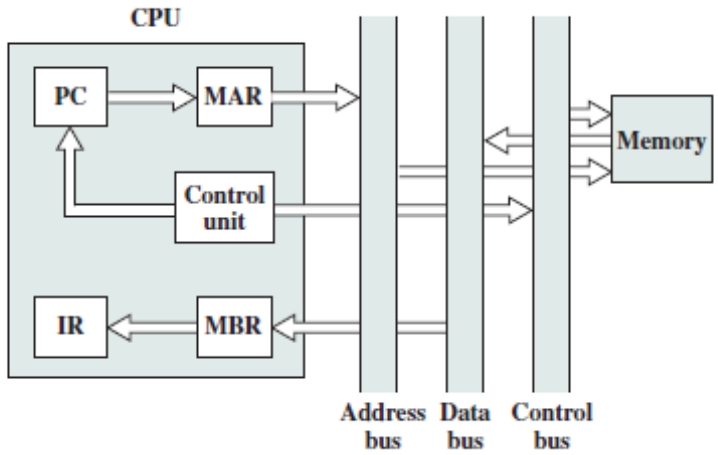
Instruction Cycle

- The exact sequence of events during an instruction cycle depends on the design of the processor
- We can, however, indicate in general terms what must happen.
  - Let us assume that a processor that employs a memory address register (MAR), a memory buffer register (MBR), a program counter (PC), and an instruction register (IR).

Instruction Cycle

- During the fetch cycle, an instruction is read from memory
- The PC contains the address of the next instruction to be fetched. This address is moved to the MAR and placed on the address bus.
- The control unit requests a memory read, and the result is placed on the data bus and copied into the MBR and then moved to the IR.
- Meanwhile, the PC is incremented by 1, preparatory for the next fetch.

Instruction Cycle



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

Figure 14.6 Data Flow, Fetch Cycle

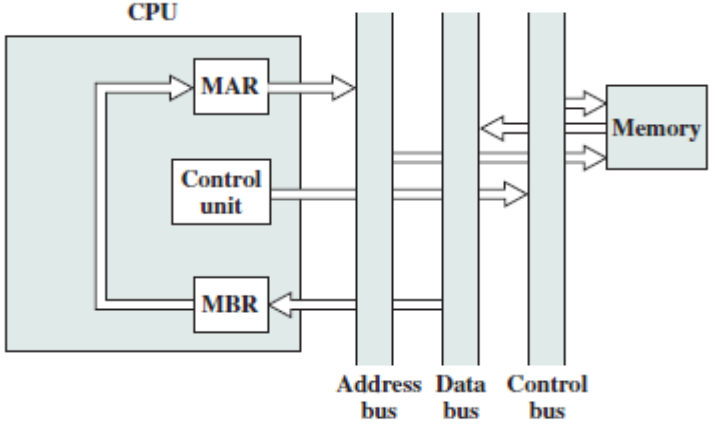


Figure 14.7 Data Flow, Indirect Cycle

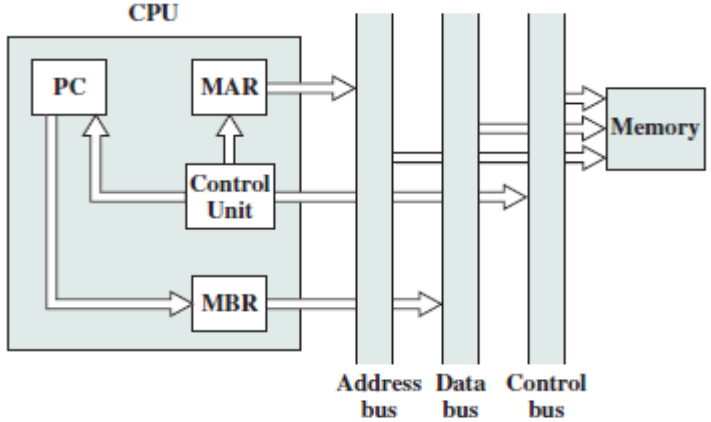


Figure 14.8 Data Flow, Interrupt Cycle

Instruction Cycle

- Once the fetch cycle is over, the control unit examines the contents of the IR to determine if it contains an operand specifier using indirect addressing

- If so, an indirect cycle is performed
  - The right- most N bits of the MBR, which contain the address reference, are transferred to the MAR.
  - Then the control unit requests a memory read, to get the desired address of the operand into the MBR.

Instruction Cycle

- The execute cycle takes many forms; the form depends on which of the various machine instructions is in the IR.
  - This cycle may involve transferring data among registers, read or write from memory or I/O, and/or the invocation of the ALU.
- Like the fetch and indirect cycles, the interrupt cycle is simple and predictable
  - The current contents of the PC must be saved so that the processor can resume normal activity after the interrupt.
  - Thus, the contents of the PC are transferred to the MBR to be written into memory

Instruction Cycle

- The special memory location reserved for this purpose is loaded into the MAR from the control unit
- The PC is loaded with the address of the interrupt routine. As a result, the next instruction cycle will begin by fetching the appropriate instruction.

Instruction Pipelining

- As computer systems evolve, greater performance can be achieved by taking advantage of improvements in technology, such as faster circuitry
- In addition, organizational enhancements to the processor can improve performance
  - An example is the use of multiple registers rather than a single accumulator, and the use of a cache memory
- Another organizational approach, which is quite common, is instruction pipelining.

Instruction Pipelining

- Instruction pipelining is similar to the use of an assembly line in a manufacturing plant.

- An assembly line takes advantage of the fact that a product goes through various stages of production

- By laying the production process out in an assembly line, products at various stages can be worked on simultaneously

Instruction Pipelining

- This process is also referred to as pipelining, because, as in a pipeline, new inputs are accepted at one end before previously accepted inputs appear as outputs at the other end.
- an instruction has a number of stages
- consider subdividing instruction processing into two stages: fetch instruction and execute instruction
- There are times during the execution of an instruction when main memory is not being accessed

Instruction Pipelining

- This time could be used to fetch the next instruction in parallel with the execution of the current one
- The pipeline below has two independent stages

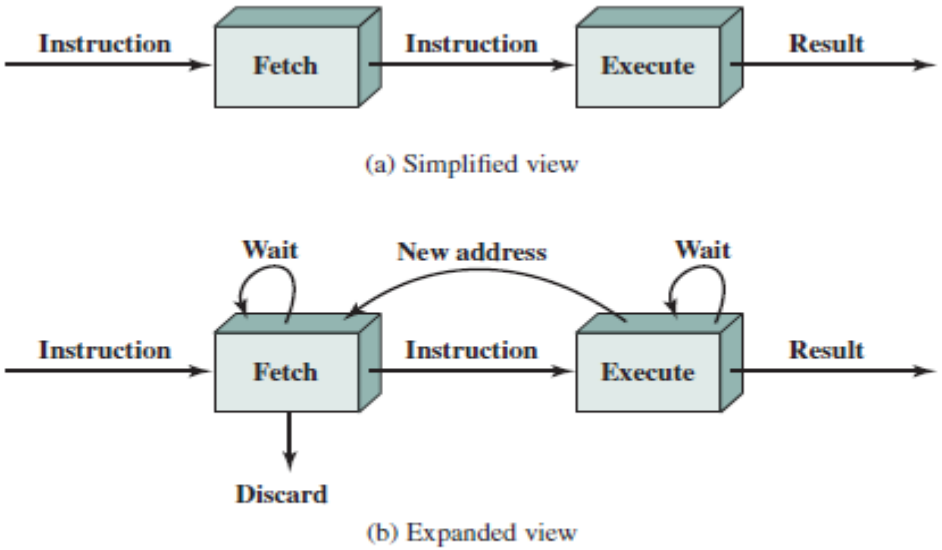


Figure 14.9 Two-Stage Instruction Pipeline

Instruction Pipelining

- The first stage fetches an instruction and buffers it
- When the second stage is free, the first stage passes it the buffered instruction.
- While the second stage is executing the instruction, the first stage takes advantage of any unused memory cycles to fetch and buffer the next instruction.
  - This is called instruction prefetch or fetch overlap
  - It requires more registers to store data between stages.

Instruction Pipelining

- this process will speed up instruction execution.
- If the fetch & execute stages were of equal duration, the instruction cycle time would be halved
- But this doubling of execution rate is unlikely for two reasons
  - The execution time will generally be longer than the fetch time. Execution will involve reading and storing operands and the performance of some operation. Thus, the fetch stage may have to wait for some time before it can empty its buffer

Instruction Pipelining

- A conditional branch instruction makes the address of the next instruction to be fetched unknown. Thus, the fetch stage must wait until it receives the next instruction address from the execute stage. The execute stage may then have to wait while the next instruction is fetched.

- While these factors reduce the potential effectiveness of the two-stage pipeline, some speedup occurs.

Instruction Pipelining

- To gain further speedup, the pipeline must have more stages processing.
- decomposition of the instruction processing

- **Fetch instruction (FI):** Read the next expected instruction into a buffer.
- **Decode instruction (DI):** Determine the opcode and the operand specifiers.
- **Calculate operands (CO):** Calculate the effective address of each source operand. This may involve displacement, register indirect, indirect, or other forms of address calculation.
- **Fetch operands (FO):** Fetch each operand from memory. Operands in registers need not be fetched.
- **Execute instruction (EI):** Perform the indicated operation and store the result, if any, in the specified destination operand location.
- **Write operand (WO):** Store the result in memory.

  – With this decomposition, the various stages will be of more nearly equal duration but we assume equal duration

Instruction Pipelining

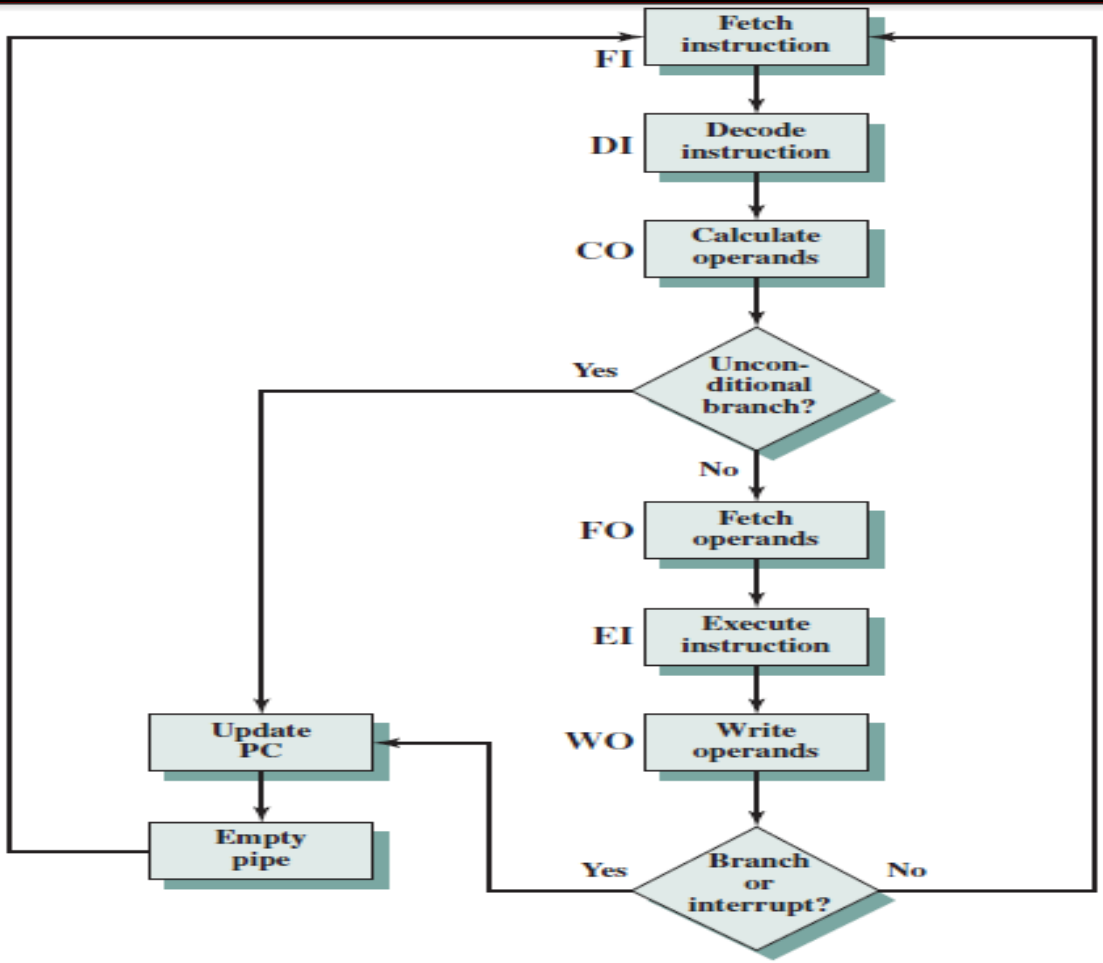
- 

Figure 14.12 Six-Stage CPU Instruction Pipeline

Instruction Pipelining

- Instruction pipelining is a powerful technique for enhancing performance but requires careful design to achieve optimum results with reasonable complexity.

LIMKOKWING UNIVERSITY COLLEGE OF CREATIVE TECHNOLOGY

THE END

- NEXT TOPIC : **CACHE MEMORY, INTERNAL MEMORY TECHNOLOGY, EXTERNAL MEMORY**