COA Preparation

Tags	CSTE Year 2 Term 2
Created time	@February 14, 2024 3:34 PM
Last edited time	@February 17, 2024 1:28 PM
Created by	Ⓑ Borhan

- **Computer Architecture and Organization**

Computer Architecture	Computer Organization
Computer architecture explains **what a computer should do**	Computer organization explains **how a computer works.**
Computer architecture provides a **functional behavior** of computer system.	Computer organization provides **structural relationship** between parts of computer system
Computer architecture deals with **high level design**	Computer Organization delas with **low level design (**often called microarchitecture **)**
Computer architecture assists in understanding the **functionality of the computer.**	Computer organization helps to understand the **exact management of component of a computer**
Actor in computer parts are **hardware** parts	Actor in computer organization is **performance**
Computer architecture **design first**	It is started **after computer architecture**
Programmer view	Transparent from programmer
It involves with **logical** attributes, like instruction sets, data types, addressing modes, etc.	It involves the relationship among **physical parts** of the system, like circuit etc.

- **IBM 370**

Aspect	IBM System 370
Architecture	32-bit mainframe computer architecture
Instruction set	a wide range of instructions for arithmetic, logical, data manipulation operations supporting both fixed-point and floating-point arithmetic
Addressing modes	provides various addressing modes including direct, indirect, indexed, base/displacement
Operating system	IBM’s OS/360 and its successor third party operating system like UNIX
Peripheral support	support for peripherals such as printers, disk drives, tape drives and communication device
Virtualization	introduced virtualization capabilities
Performance	high performance, reliability, suitable for large scale computing task in business, government and academia

- **Differences between Structure and function**

Structure	Function
Structure refers to the organization and **interconnection of hardware components** within a computer system	Function refers to **the operation and tasks performed by the hardware components** within computer system to execute instructions, process data, and interact peripherals.
The structure includes components such as **CPU, memory, buses, input/output devices**.	Functions encompass operations such as **instruction fetching, decoding, execution, memory access, arithmetic/logic operations, i/o operation, system control** etc.
It involves the arrangement and interconnection of **hardware components to facilitate data flow, control signals and communication within computer system.**	Functions are organized to **execute instructions in a sequential or parallel manner, utilizing the capabilities** of different hardware components.
Structure optimization involves **designing and configuring hardware components to minimize bottlenecks, reduce latency, increase throughput and enhance overall performance.**	Function optimization focuses on i**mproving the efficiency of individual operations and algorithm implemented by the hardware components, utilizing techniques** such as pipelining, parallel processing, caching etc.

- **Computer**

System

- **CPU : Control Processing Unit**

  Potions of the computer that fetches and executes instructions
  - **CU**
    - Control Unit
    - Controls the operation of the CPU
  - **ALU**
    - Arithmetic and Logic Unit
    - Perform Data processing function
  - **Resistors**
    - Provides storage internal to the CPU
  - **CPU Interconnection**
    - Some mechanism that provides for communication among the CU, ALU, registers
- **Main Memory : Stores data**
- **I/O:** moves data between the computer and its external environment

Functions

- **Data Processing**

  The data may take a wide variety of forms, and the range of processing requirements is broad.
- **Data Storage**

  Even if the computer is processing data on the fly, the computer must temporarily store at least those piece of data that are being worked on at any given moment. Thus, there is at least **short-term memory** data storage function. Equally important, the computer performs a **long-term data storage** function.
- **Data Movement**

  The computer ‘s operating environment consists of device that serve as either source or destination of data.
  - **I/O :** when data are received from or delivered to a device, that is directly connected to the computer

- **System Interconnection:** Some mechanism that provides for **communication** among CPU, main memory and I/O. A common example of system interconnection is by means of a **system bus**.
    - **Data communications:** when data are moved over longer distances to or from a remote device
- **Control:** A control unit manages the **computer's resources and orchestrates the performance of its functional parts** in response to instructions.

- **A top-down view of a computer**

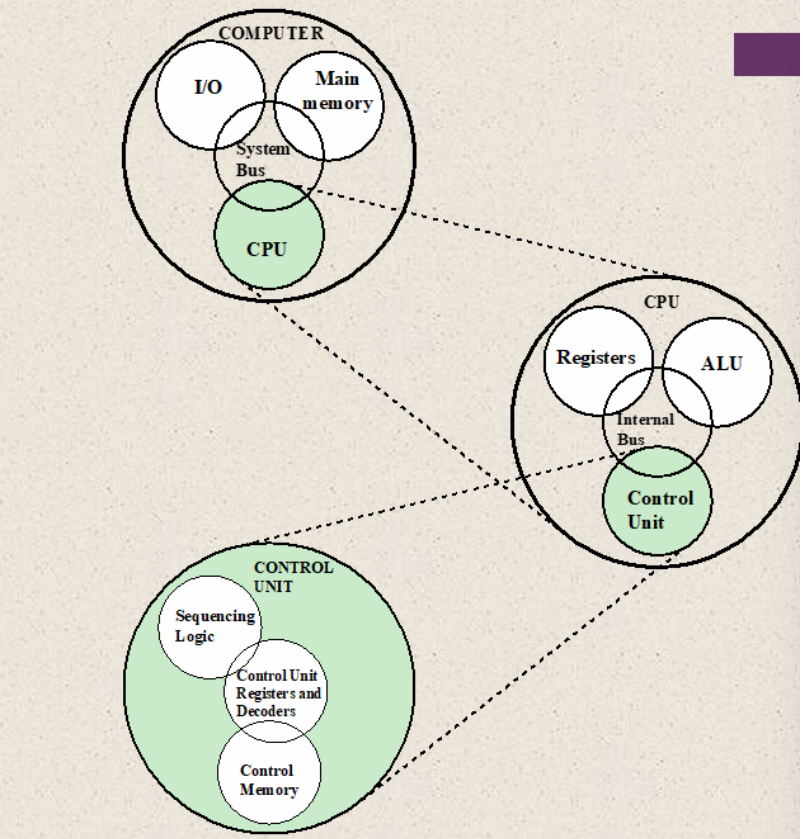
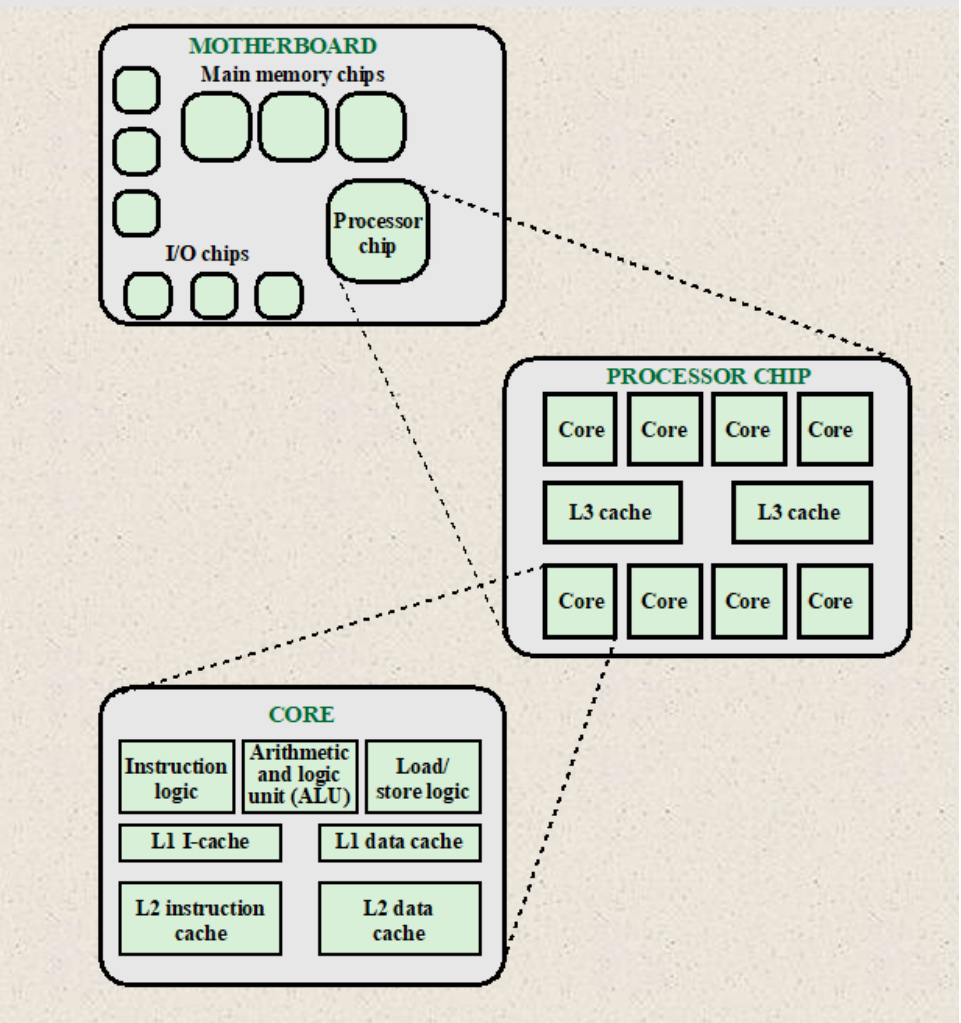


Figure 1.1 A Top-Down View of a Computer

- **Simplified view of Major elements of a multicore computer**



- **Computer Generations differences**

Aspect	First Generation	Second Generation	Third Generation	Fourth Generation	Fifth Generation
Time Period	1940s to early 1950s	Late 1950s to mid-1960s	Late 1960s to mid-1970s	Mid-1970s to mid-1990s	Mid 1990s to Present
Technology	Vacuum tubes	Transistors	Integrated Circuits	Microprocessor	AI, Quantum Computing, Nanotechnology
Size and Power	Large, bulky, high power consumption	Smaller, less power consumption	Further miniaturization, lower power usage	Smaller, more efficient , increasingly powerful	Higher compact efficient and powerful
Programming	Machine language	Assembly language	High-level language (FORTRAN, COBOL)	High-level languages (C, C++, Java)	Focus on AI, ML. NLP, Robotics
Main usage	Scientific calculations,	Business data processing,	Business applications,	Personal computers,	AI research, ML, Robotics

Aspect	First Generation	Second Generation	Third Generation	Fourth Generation	Fifth Generation
	codebreaking	scientific research	database management	networking, internet	
Memory	Magnetic drum, limited capacity	Magnetic core, improved capacity	Semiconductor RAM	Magnetic and solid-state storage, RAM	Advance memory technology
User Interface	Punch cards, no interactive interface	batch processing limited interactive access	Time-sharing system, limited GUI	GUI, OS Support	NLP, AI agents
Performance	Slow, limited capabilities	Improved speed and capabilities	Further speed improvement	Rapid advancement in speed and capabilities	Focus on optimization and parallelism
Example	ENIAC	IBM 7000	IBM 360	IBM PC, Apple Macintosh	IBM Watson, Deep Blue, Google DeepMind

- **IAS Structure**

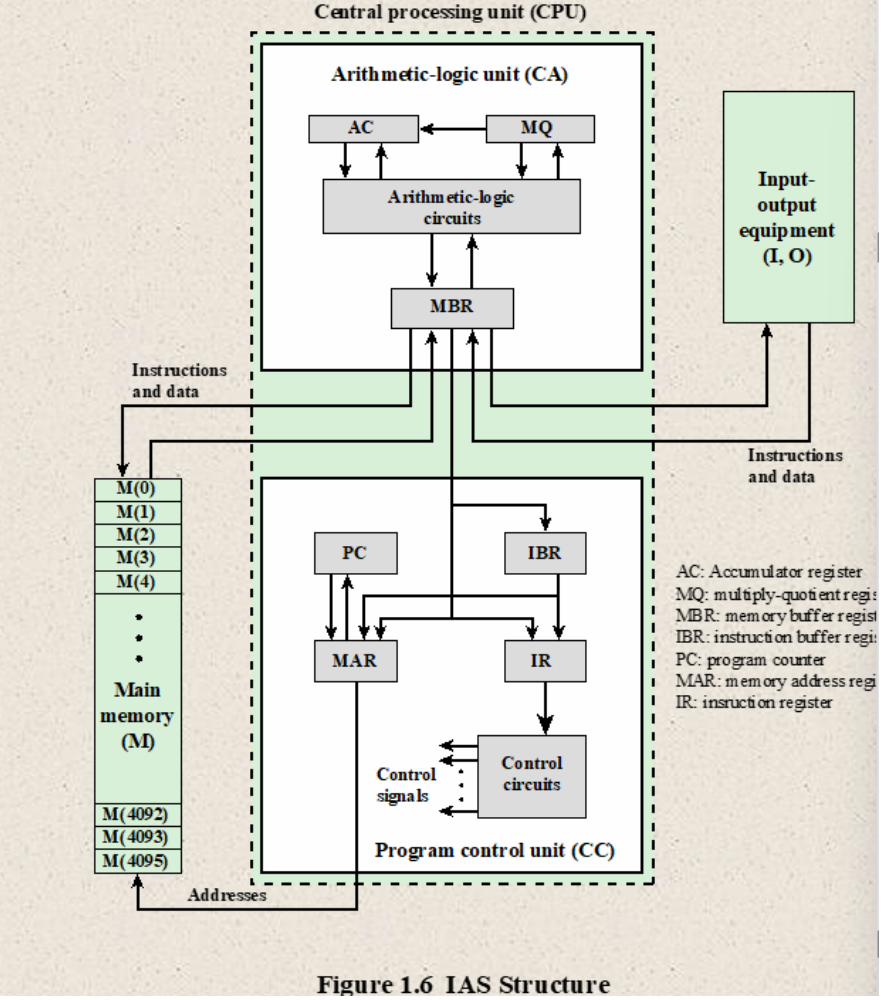


Figure 1.6 IAS Structure

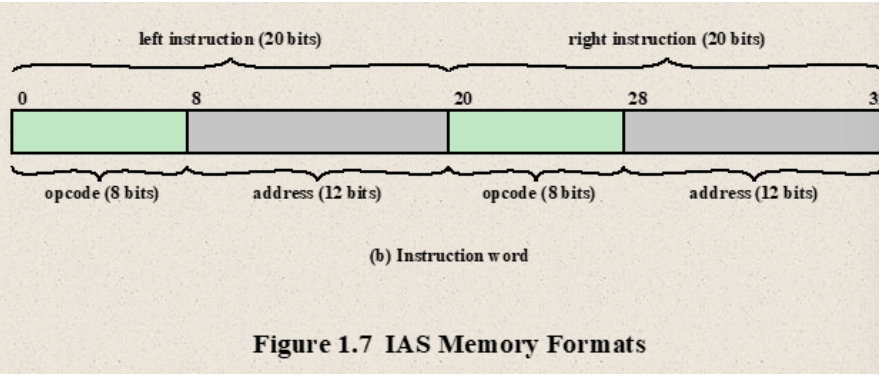


Figure 1.7 IAS Memory Formats

- **Registers**
  - **MBR : Memory Buffer Register**
    - **Contains a word** to be stored/received in/from memory or sent to/from the I/O input.
  - **MAR : Memory Address Register**
    - Specifies the **address in memory of the word to be written** from or read into the MBR
  - **IR : Instruction Register**

        - Contains the **8 bit opcode instruction b**eing executed
    - **IBR : Instruction Buffer Register**
        - Employed to temporarily **hold the right hand instruction** from a word in memory
    - **PC : Program Counter**
        - Contains **the address of the next instruction pair to be fetched** from memory
    - **Accumulator (AC) and Multiplier Quotient (MQ):**
        - Employed to temporarily hold **operands and results of ALU operations**

- **IBM 7094 Configuration**

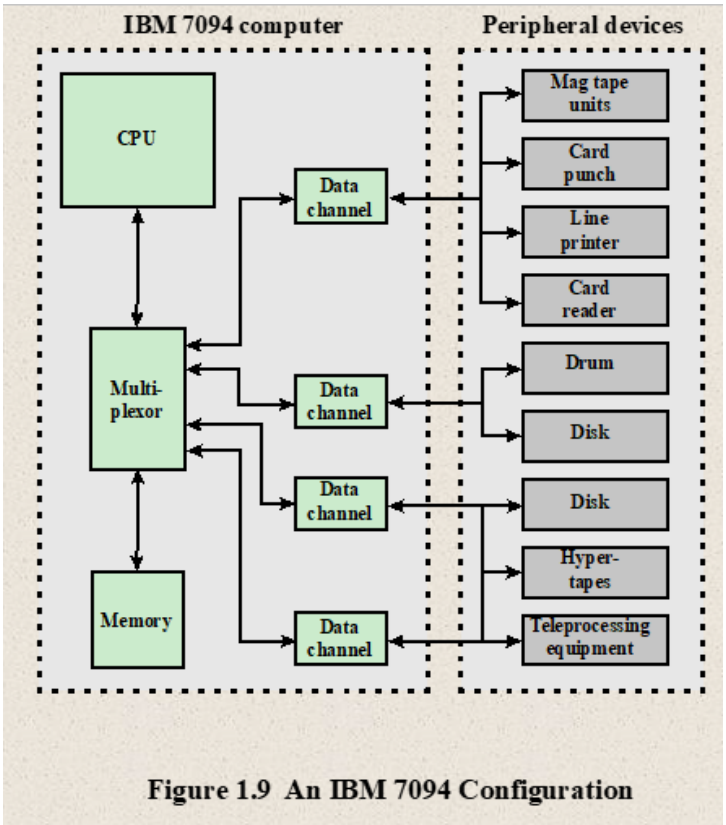


Figure 1.9 An IBM 7094 Configuration

- **Moore's Law and it's consequence**

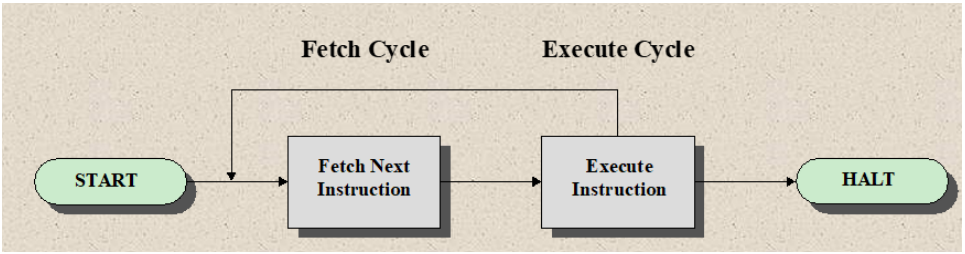
Moore's Law: Number of transistors that could be put on a single chip was doubling every year.

Consequences:

- The cost of computer logic and memory circuitry has fallen at a dramatic rate
- The electrical path length is shortened increasing operating speed.
- Computer becomes smaller and it more convenient to use in a variety of environments.
- Reduction in power and cooling requirements.

- Fewer interchip connection.

- **Basic Instruction Cycle**



- **Fetch Cycle**
  - The processor fetches an instruction from memory
  - The PC holds the address of the instruction to be fetched next
  - The processor increment the PC after each instruction fetch
  - The fetched instruction is loaded into the IR
  - The processor interprets the instruction and perform the required action.

- **Computer Components : Top-Level View**

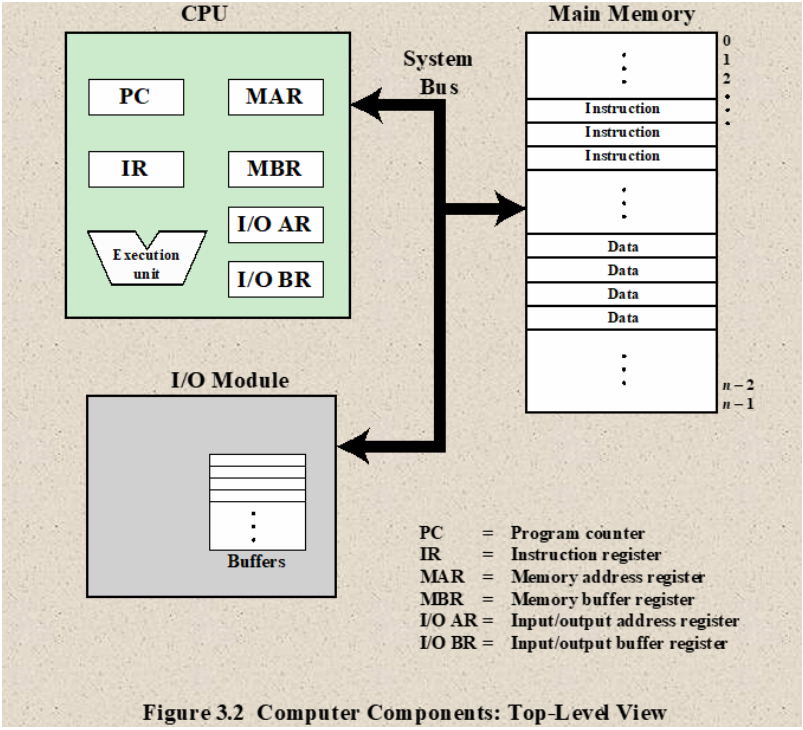


Figure 3.2 Computer Components: Top-Level View

- **Action categories**
  - **Processor memory :** Data may be transferred from processor to memory or from memory to processor.
  - **Processor I/O:** Data maybe transferred to or from a peripheral device by transferring between the processor and I/O module.
  - **Data Processing:** The processor may perform some arithmetic or logic operation on data.
  - **Control:** An instruction may specify that the sequence be altered. For example, the processor may fetch an instruction from location 149, which specifies that he next instruction be from location 182. The processor will remember this fact by setting the program counter to 182 rather than 150.

- **Characteristics of a Hypothetical machine**

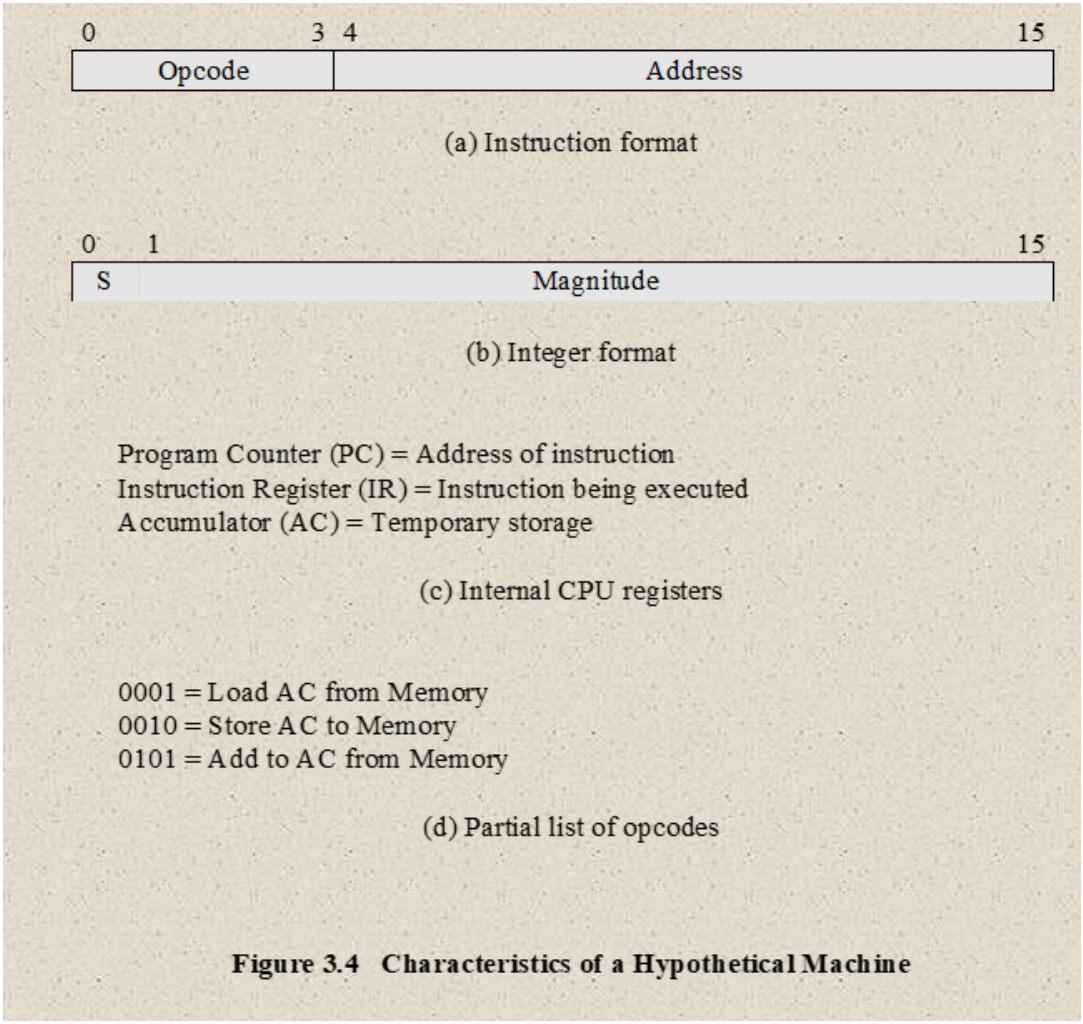


Figure 3.4 Characteristics of a Hypothetical Machine

- **Program Execution**

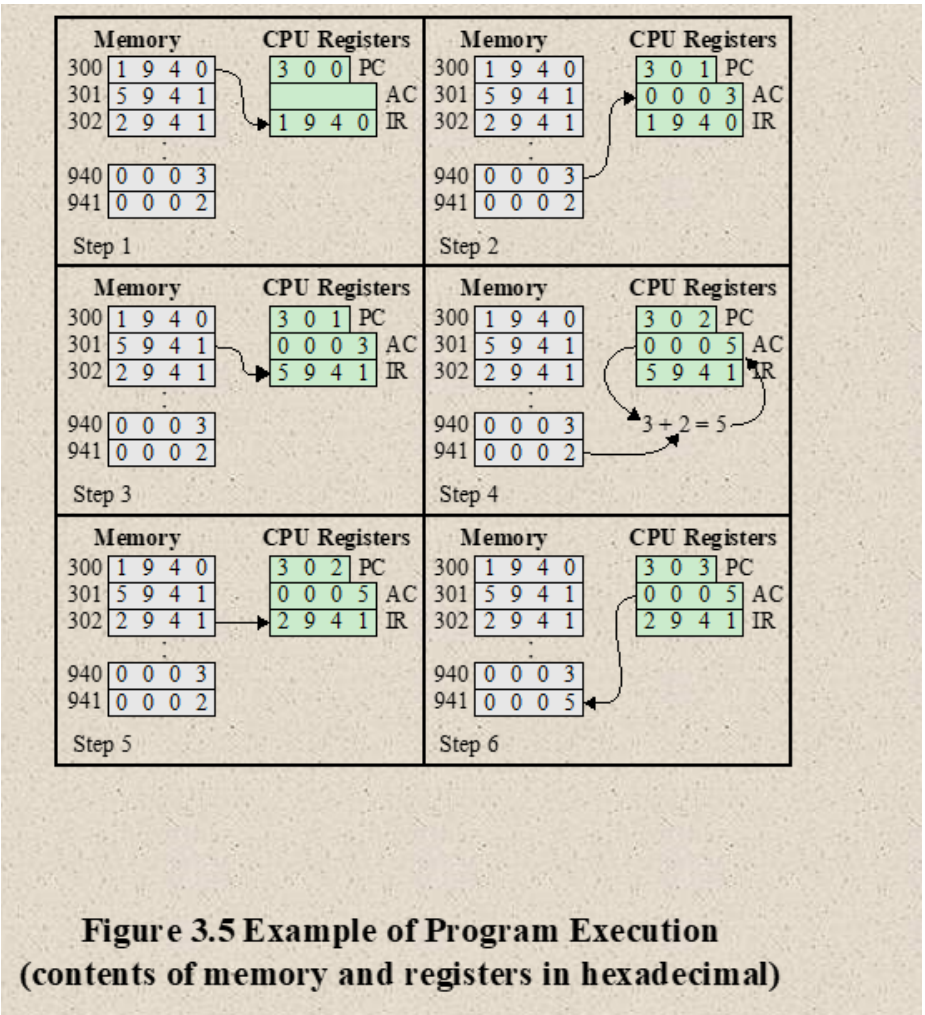
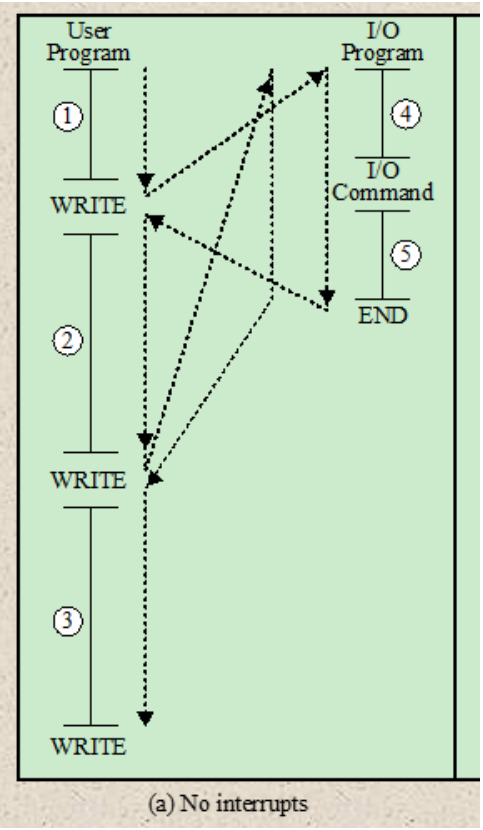


Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)

- **Classes of interrupts**
  - **Program:** Generated by some condition occurs as a result of an instruction execution, such as arithmetic overflow, division by zero.
  - **Timer:** Generated by a timer within processor.
  - **I/O:** Generated by an I/O controller, to signal normal completion of an operation.
  - **Hardware Failure:** Generated by a failure such as power failure or memory parity error

- **No Interrupt Program flow**



- **Instruction Cycle with Interrupts**

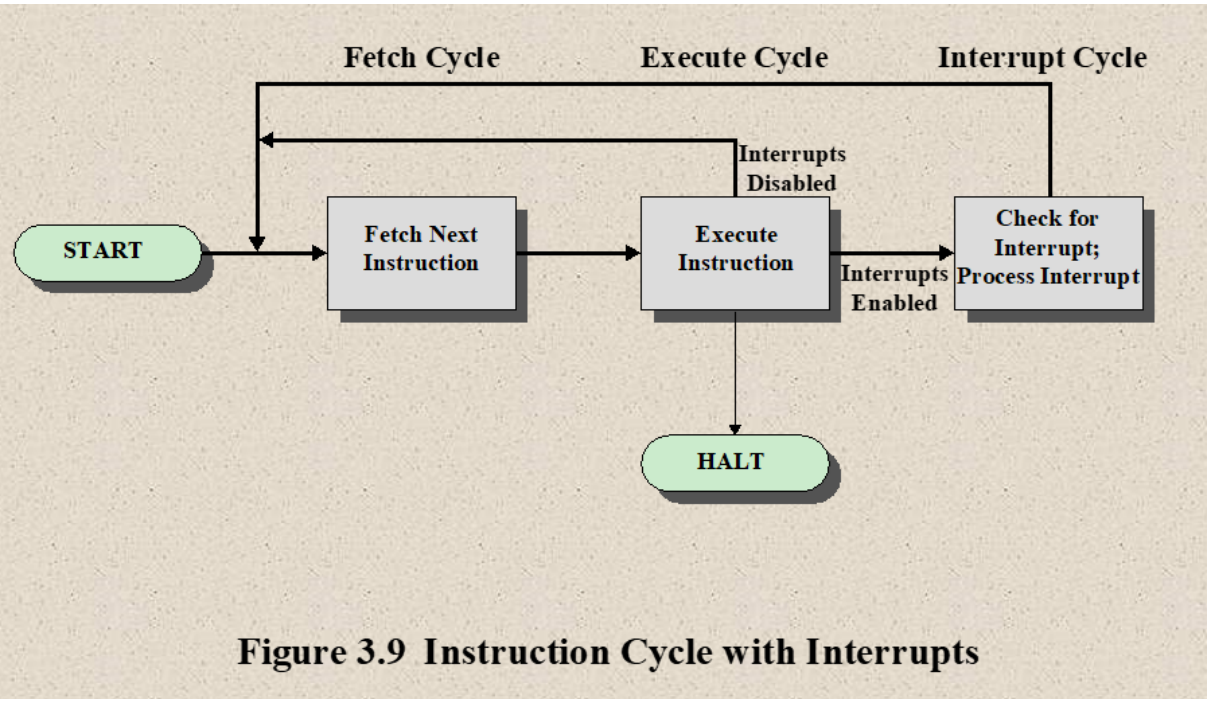


Figure 3.9 Instruction Cycle with Interrupts

- **Transfer of Control via Interrupts**

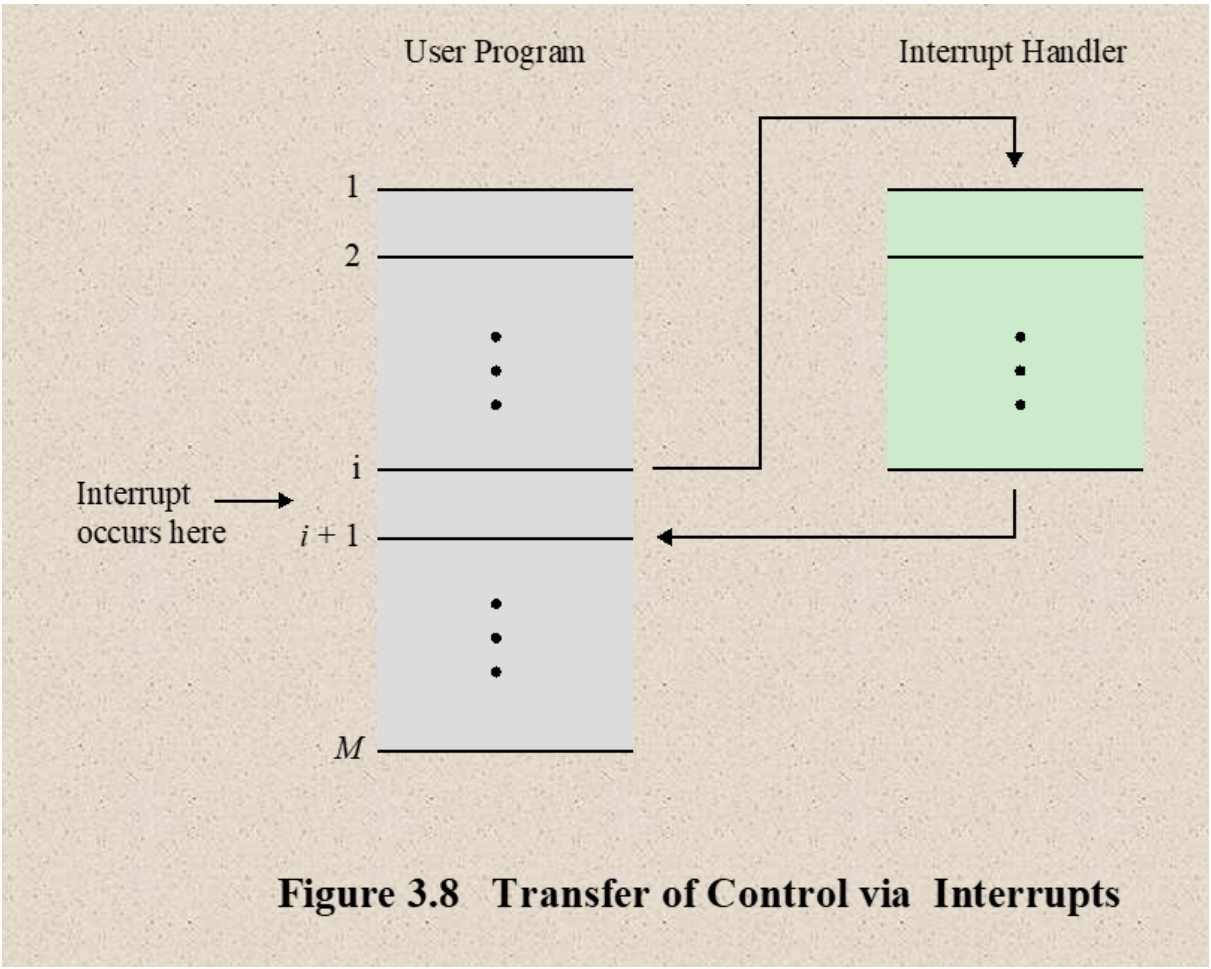


Figure 3.8 Transfer of Control via Interrupts

- **Instruction Cycle State Diagram with Interrupts**

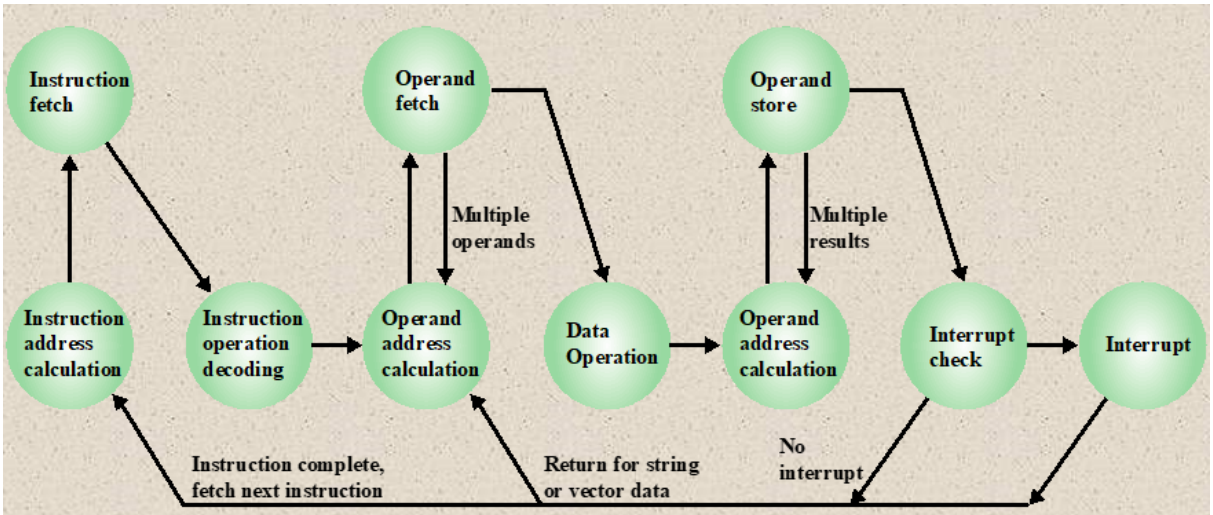


Figure 3.12 Instruction Cycle State Diagram, With Interrupts

- **Transfer of Control with Multiple Interrupts**

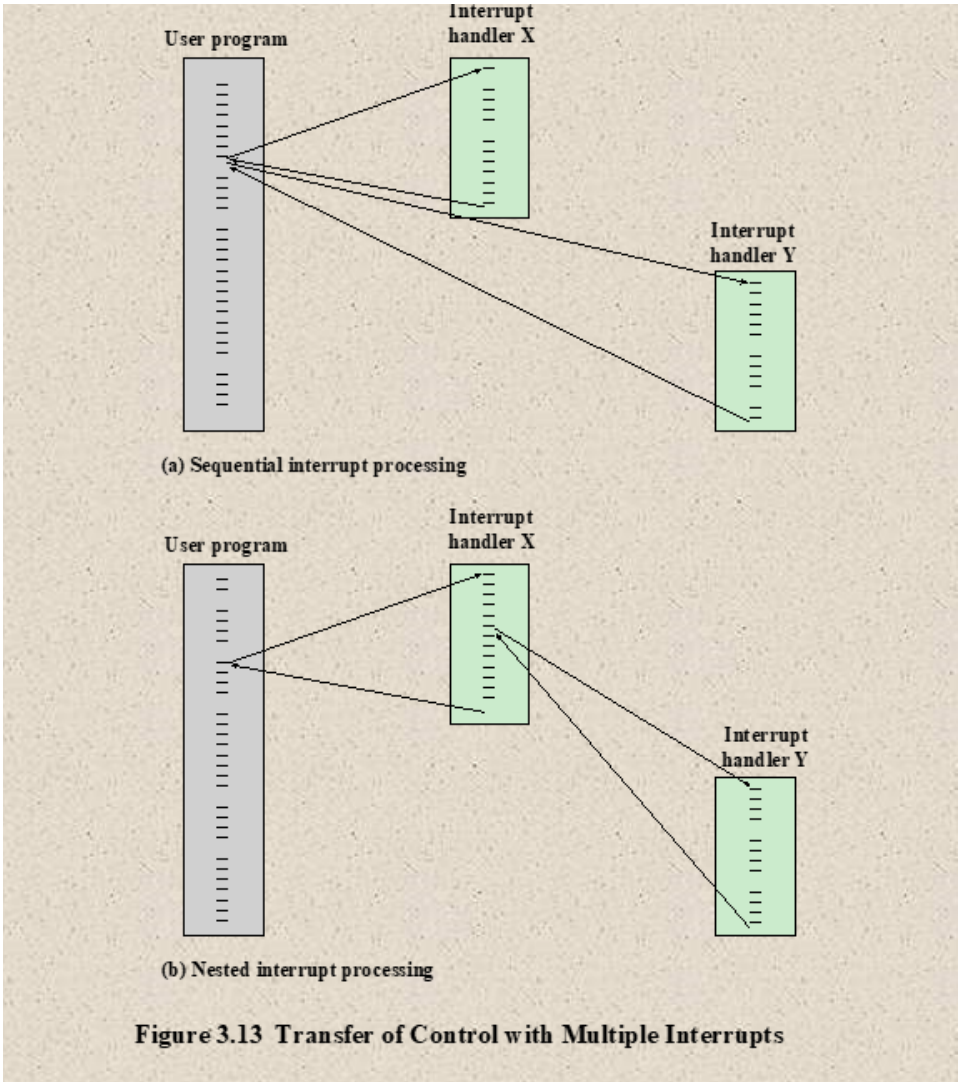


Figure 3.13 Transfer of Control with Multiple Interrupts

- **Time Sequence of Multiple Interrupts**

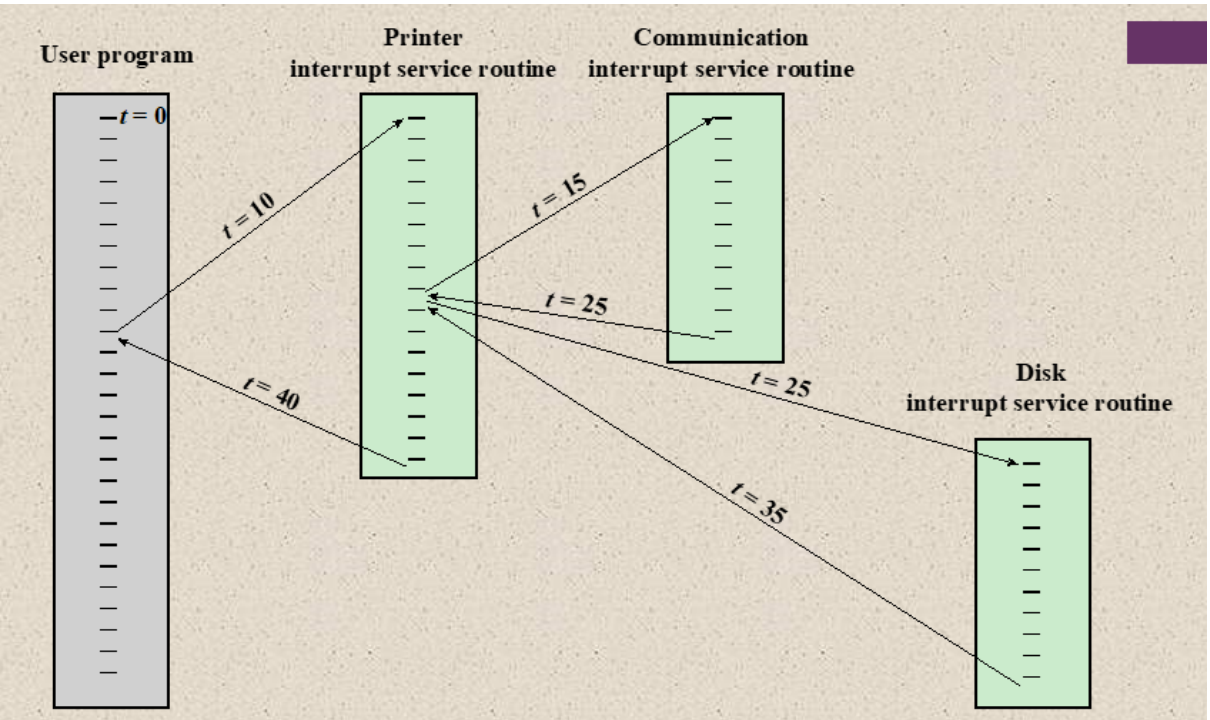


Figure 3.14 Example Time Sequence of Multiple Interrupts

- **I/O Function**
  - I/O Module can exchange data directly with the processor
  - Processor can read data from or write data to an I/O module
  - In some cases it is desirable to allow I/O exchanges to occur directly with memory (known as Direct Memory Access DMA)

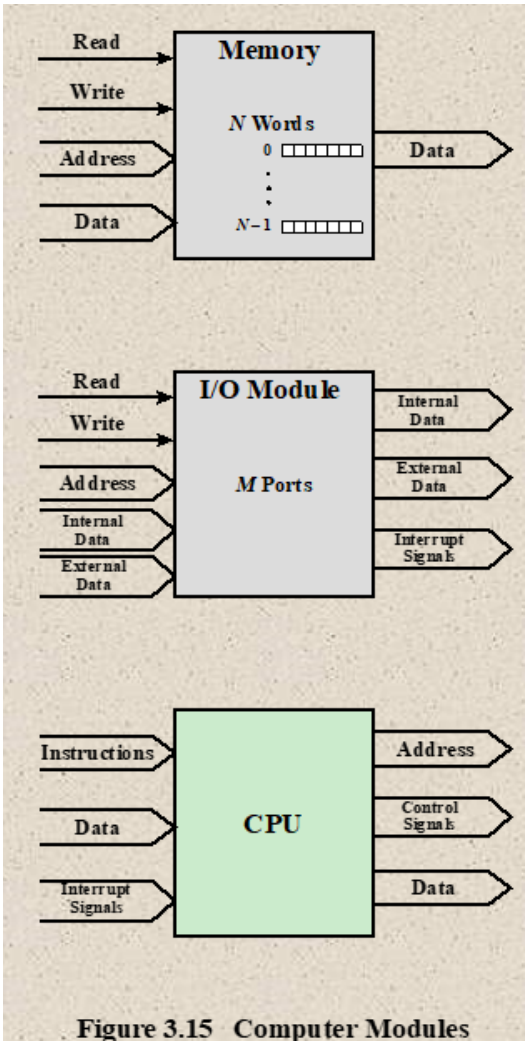
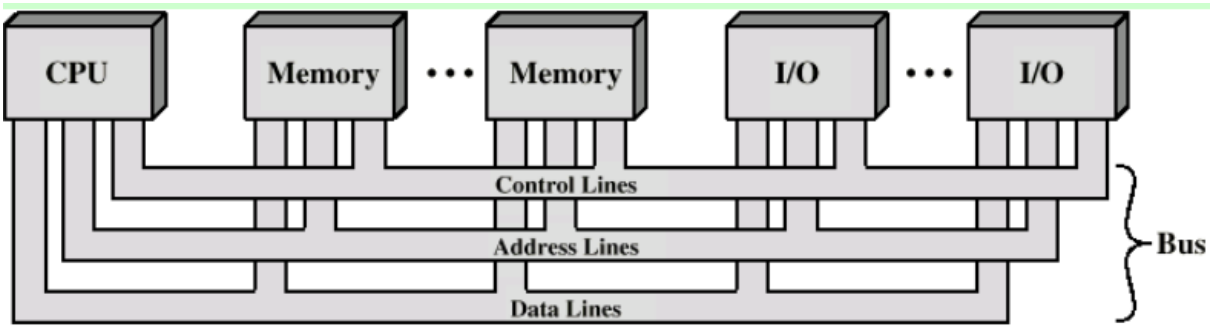


Figure 3.15 Computer Modules

- **Bus**

A communication pathway connecting two or more devices.

- **Bus Interconnection Scheme**



- **Interconnection Structure**

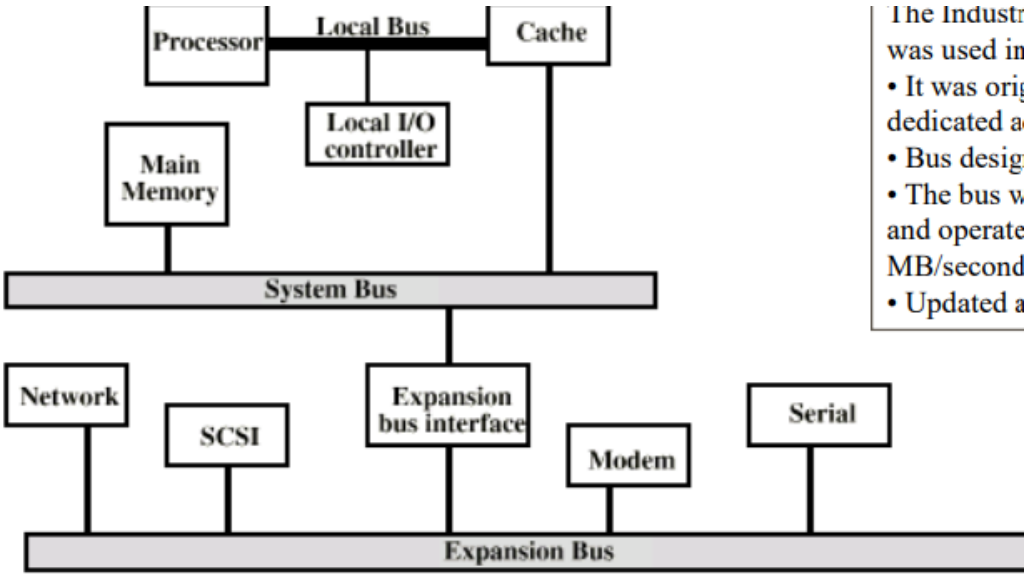
- **Memory to Processor :** the processor reads an instruction from memory
- **Processor to memory:** the processor writes a unit of data to memory
- **I/O to processor:** the processor reads data from I/O device
- **Processor to I/O :** The processor sends data to the I/O
- **I/O to or from memory:** For those two cases, an I/O module is allowed to exchange data directly with memory, without going through the processor

- **Bus Type**
  - **Data Bus:** carries data, Width is a key determinant of performance (16, 32 bit)
  - **Address bus:** Identify the source or destination of data
  - **Control Bus:** Control and timing information, memory signal, Interrupt request, Clock signals

  2

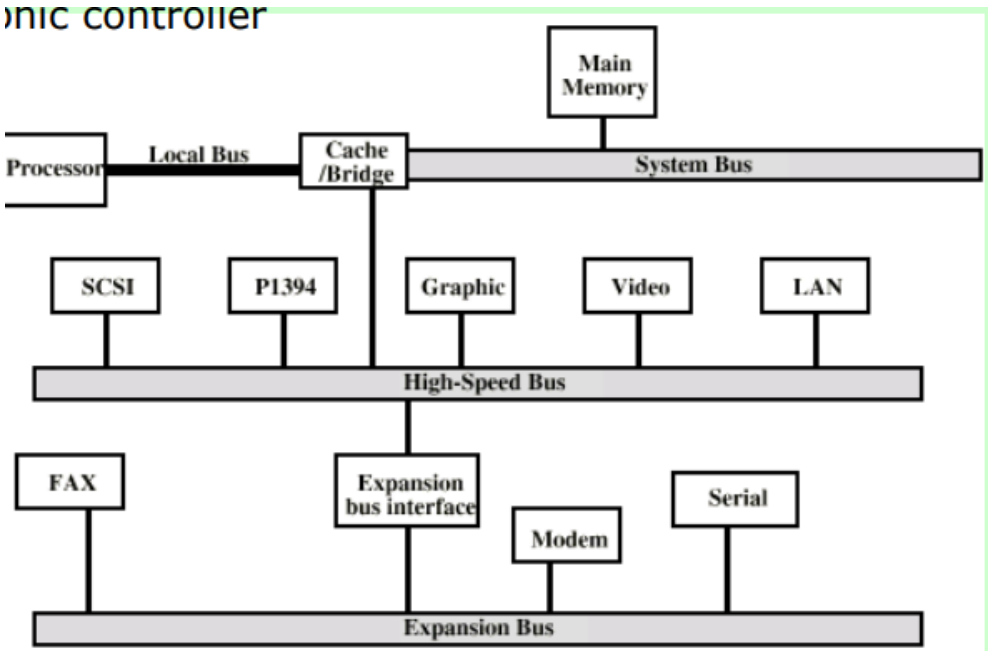
  - **Dedicated:** Permanently assigned to one function or to a physical subset of computer components
  - **Multiplexed:** same line used for multiple purpose
- **Bus Width**
  - The width of a bus is the number of lines. The more data lines, the more data that can be transferred simultaneously.
  - “32 bit bus” → 32 data lines
  - the more address lines → larger the maximum amount of memory that can be accessed
  - the greater the width → the more hardware required

- ISA bus

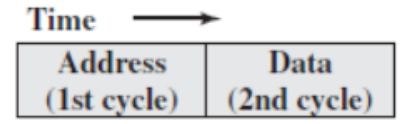


The Industr
was used in
• It was orig
dedicated a
• Bus desig
• The bus w
and operate
MB/second
• Updated a

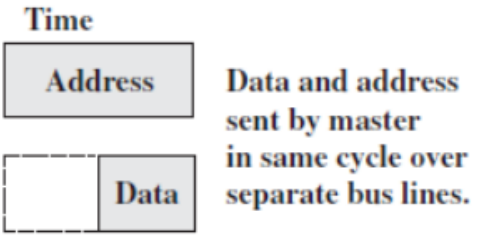
- High Performance Bus



- **Multiplexed Bus Operation**

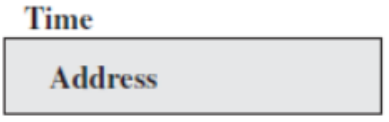


Write (multiplexed) operation



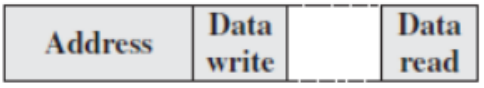
Write (non-multiplexed) operation

Read (multiplexed) operation



Read (non-multiplexed) operation

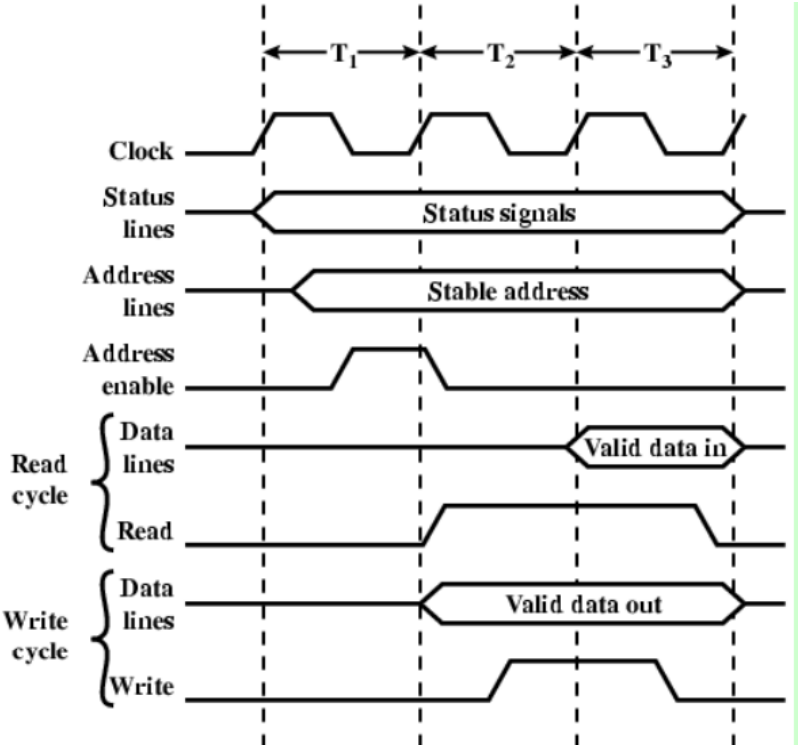
Read-modify-write operation



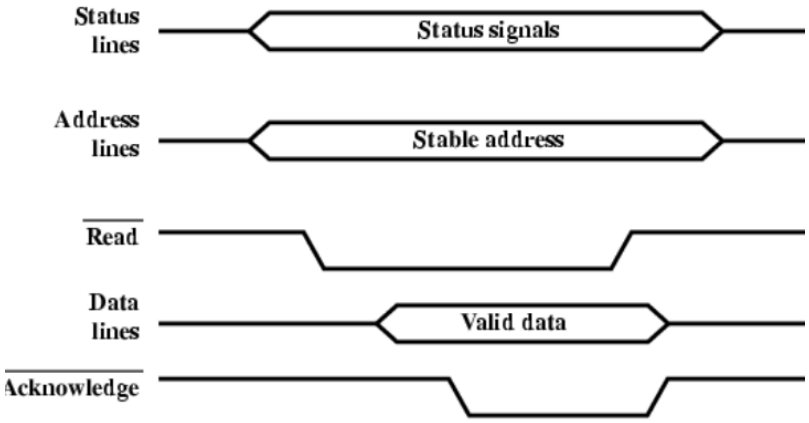
Read-after-write operation

Block data transfer

- **Synchronous timing diagram**



- **Asynchronous write diagram**



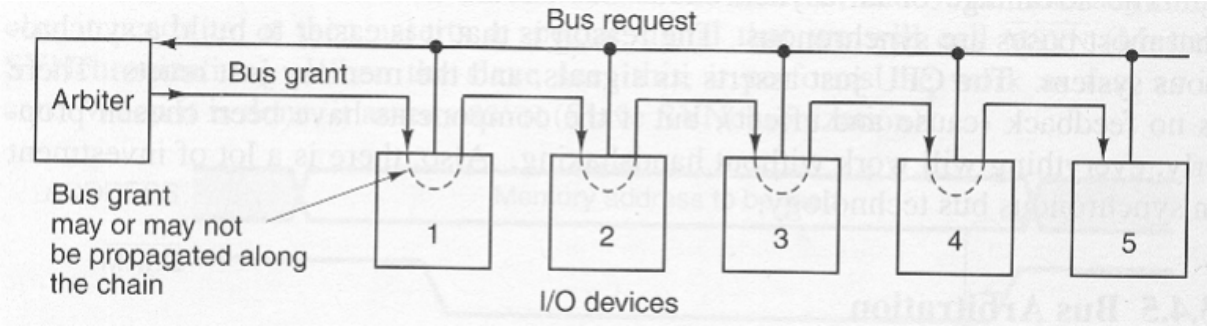
- **Bus Arbitration**

  Bus arbitration is a process by which another device is selected to become a bus mater.

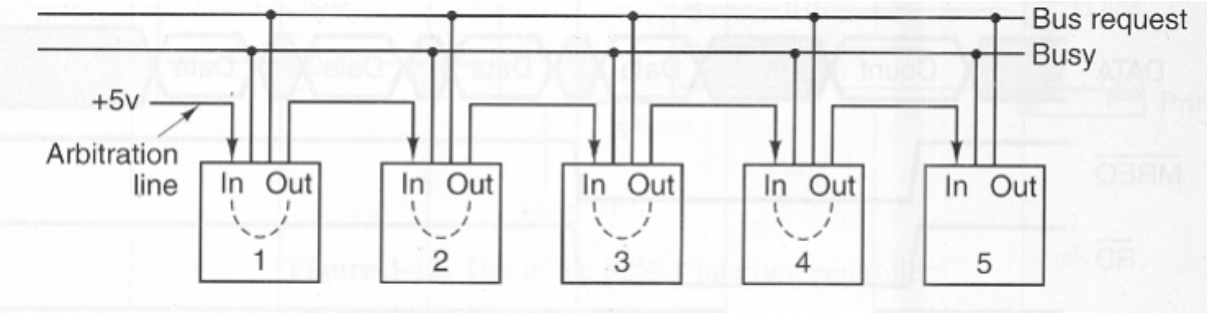
- **Differences between Bus Arbitration**

Centralized	Distributed	Decentralized
Central authority	Shared among all nodes or devices on the bus	Distributed among multiple independent nodes

Centralized	Distributed	Decentralized
Controlled by a single arbiter	May involve negotiation among requesting devices	Decisions made autonomously by individual nodes
Granted sequentially	Negotiated among requesting devices	Nodes independently decide when to access the bus
Algorithm: fixed-priority or round robin	Token Passing, CSMA	Voting, Consensus
simple implementation	complex due to negotiation	may require sophisticated algorithm and coordination mechanism
performance : Efficient in smaller systems	Performance may vary depending on algorithm	Performance influenced by the arbitration method chosen
Example :PCI, ISA	Ethernet	Blockchain-based protocols



Centralized



Decentralized

Previous:

- **What is addressing mode? Differentiate between a direct and an indirect addressing mode.**

Addressing mode: Addressing mode is a basically a technique **used for determining the operand** that associates with any given instruction.

Parameters	Direct Addressing Mode	Indirect Addressing Mode
Address Field	Address field contains the effective address of operand.	Address field contains reference of effective address.
Memory References	Requires only one memory reference	Requires two memory references
Processing Speed	This addressing mode has fast addressing compared to indirect addressing mode	It is slower than direct addressing mode.
Classification	No further classification	Further classified into two categories- Memory Indirect and Register Indirect Addressing Mode.
Calculation	No further calculation is required to perform the operation.	Requires further calculation to find the effective address.
Address Space	It occupies a smaller amount of space than the indirect mode.	It occupies a large amount of space than the direct mode.
Overhead	No additional overhead is involved while searching for operand.	Additional overhead is involved while searching for operand.
Advantage	Easy as no intermediary is involved.	Availability of large address space.
Disadvantage	Address space is restricted.	Requires more number of memory references.
Application	It aids in accessing static data and using variables.	It assists in passing arrays as parameters and implementing pointers.

Addressing method	Explicit: address of the operand is directly specified in the instruction	Explicit: address of the operand is stored in a memory location specified by the instruction
Flexibility	Less flexible: only allows for fixed memory addressing	More flexible: allows for more dynamic memory addressing
Code size	Requires less code: operand address is directly specified in the instruction	Requires more code: additional instructions are needed to load the operand address from memory
Execution speed	Faster: operand address is immediately available	Slower: additional memory accesses are required to obtain the operand address
Complexity	Less complex: requires fewer instructions and is generally easier to debug	More complex: requires additional instructions and memory accesses, and can be more difficult to debug

- **Distinguish between RISC and CISC Processor**

RISC	CISC
Reduced Instruction Set Computer	Complex Instruction Set Computer
Simple Instruction	complex
Simple processor circuitry c	complex
Requires more register	less
Large Program	small
More RAM usage	less

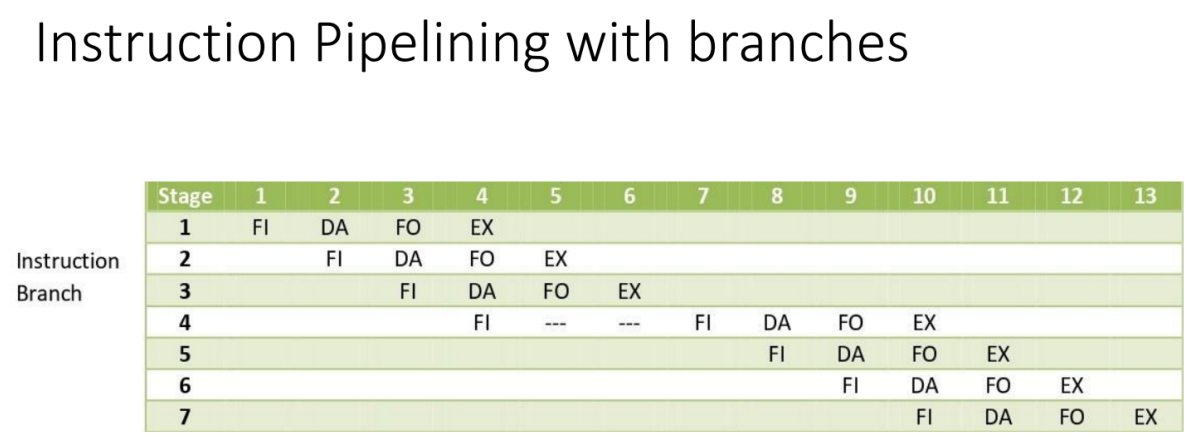
RISC	CISC
Fixed length instruction	variable
Fixed clock cycle	variable
Focus on Software	Hardware
used for high-end application, like video processing, telecommunication	low-end-application like home automation device, security system

- **Show the instruction pipelining with branch operations. Can Pipelining be a trouble? why? How can we resolve it ?**

Instruction Pipelining is a technique used to **improve instruction throughout and overall performance** by allowing multiple instructions to be processed concurrently.

Handling Branches in instruction pipelining introduces challenges because the next instruction to be fetched may depend on the outcome of a branch instruction.

For Branch Operation



FI → Fetch the instruction from memory

DA → Decode the instruction

FO → Fetch the operands from memory

Ex → Execute the instruction.

Yes, Pipelining be a trouble for several issues and complications. These are:

- **Pipeline Hazards**
  - Data
  - Control
  - Structural
- **Pipeline flush**

Incorrect branch prediction can lead to these type of Hazards

- **Clock Speed Limitations**

  Deeper Pipelines may face this type of trouble.

The way of resolving:

- We can always resolve hazards by waiting
- Pipeline control must detect hazards and take action to resolve hazards.

- **Difference Between Half Adder and Full Adder**

Parameter	Half Adder	Full Adder
Basics	The Half Adder is a type of combinational logic circuit that adds two of the 1-bit binary digits. It generates carry and sum of both the inputs.	The Full Adder is also a type of combinational logic that adds three of the 1-bit binary digits for performing an addition operation. It generates a sum of all three inputs along with a carry value.
Adding the Previous Carry	The Half Adder does not add the carry obtained from the previous addition to the next one.	The Full Adder, along with its current inputs A and B, also adds the previous carry.
Hardware Architecture	A Half Adder consists of only one AND gate and EX-OR gate.	A Full Adder consists of one OR gate and two EX-OR and AND gates.
Total Inputs	There are two inputs in a Half Adder- A and B.	There are a total of three inputs in a Full Adder- A. B. C-in.
Usage	The Half Adder is good for digital measuring devices, computers, calculators, and many more.	The Full Adder comes into play in various digital processors, the addition of multiple bits, and many more.
Logical Expression	Here is the logical expression of Half Adder: C = A * B S = A $\oplus$ B	Here is the logical expression of Full Adder: Cout = (AB) + CinA $\oplus$ CinB S =A $\oplus$ B $\oplus$ Cin

- **Distinguish between SRAM and DRAM**

SRAM	DRAM
Static RAM	**Dynamic** RAM
Structure is flip-flops made of **transistors**	**Capacitors**
faster	slower
complex but less dense	less complex but more **dense**
expensive	cheap

SRAM	DRAM
used in **cache memory**	**main memory**

- **4 Bit Ripple carry adder. What is the disadvantage of it ?**

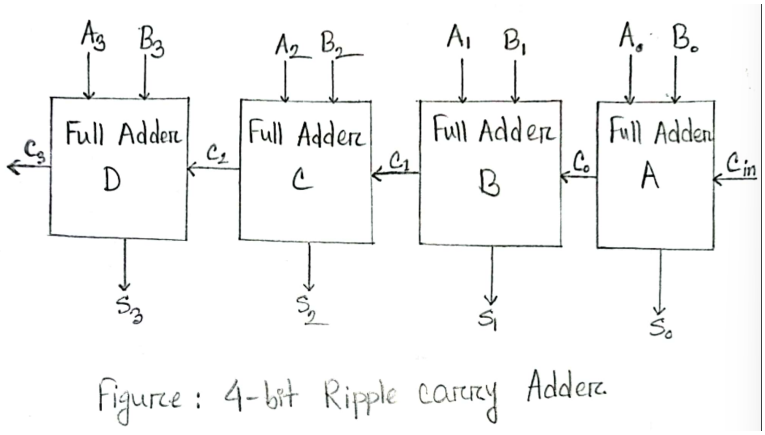


Figure : 4-bit Ripple carry Adder.

Disadvantages:

- It doesn't allow to use all full adders simultaneously.
- Each full adder has to necessarily wait until the carry bit becomes available,
- This increase propagation time
- It is slow

- **What is Parallel processing ? Discuss about the levels parallel processing with proper examples.**

Parallel processing is a method in computing of **running two or more processor** to handle separate parts of an overall task.

Four levels:

- **Instruction level :** It refers the situation where different instruction of a program are executed by different processing elements. Most processor has several execution units and can execute several instruction at the same time.
- **Loop level:** At this level consecutive loop iterations are the candidates for parallel execution. There is a lot of scope for parallel execution at loop level.

  Example : in C language,

  ```
  for(i=0; i≤n; i++)
      A(i) = B(i)+C(i)
  ```

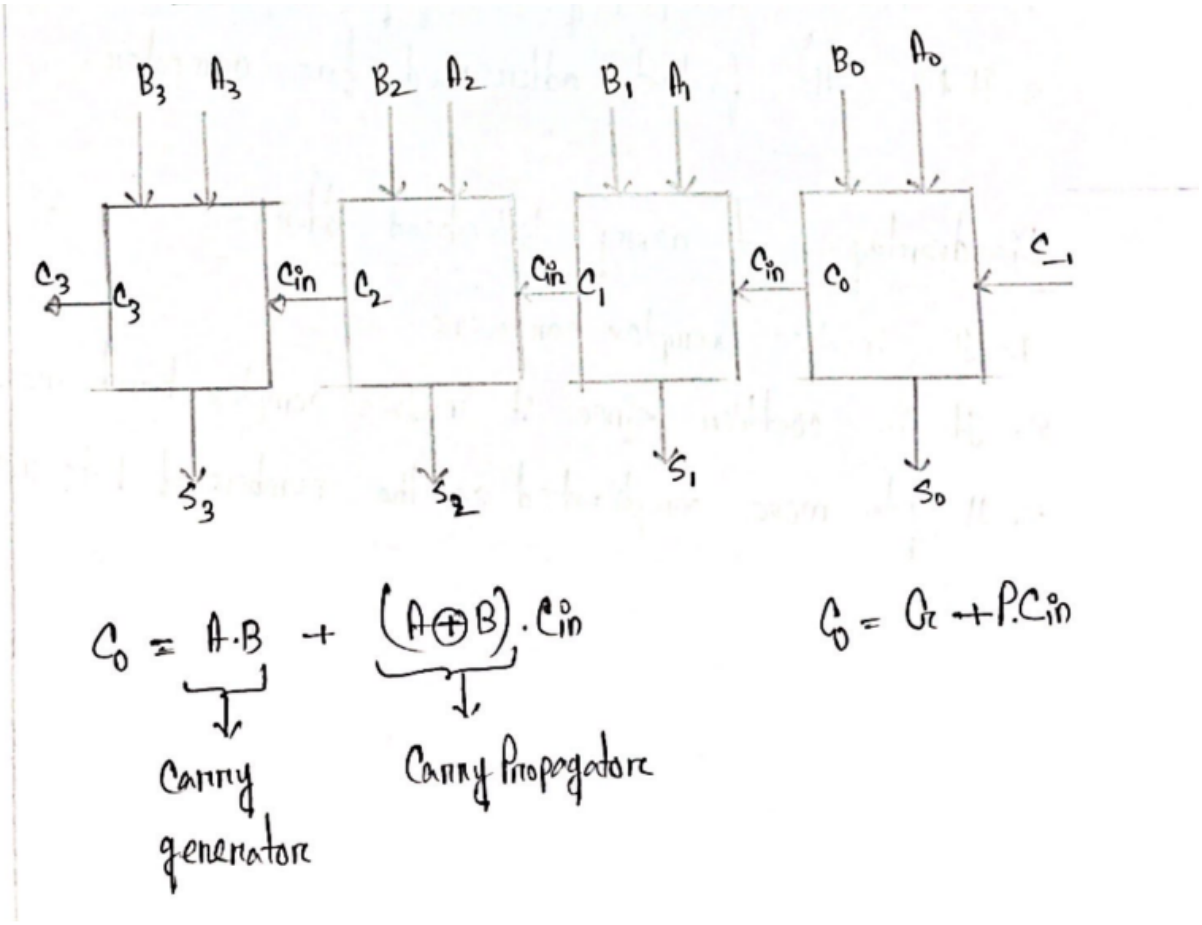
  Each of instruction A(i)=B(i)+C(i) can be executed by different processing element provided. There are at least n processing element.

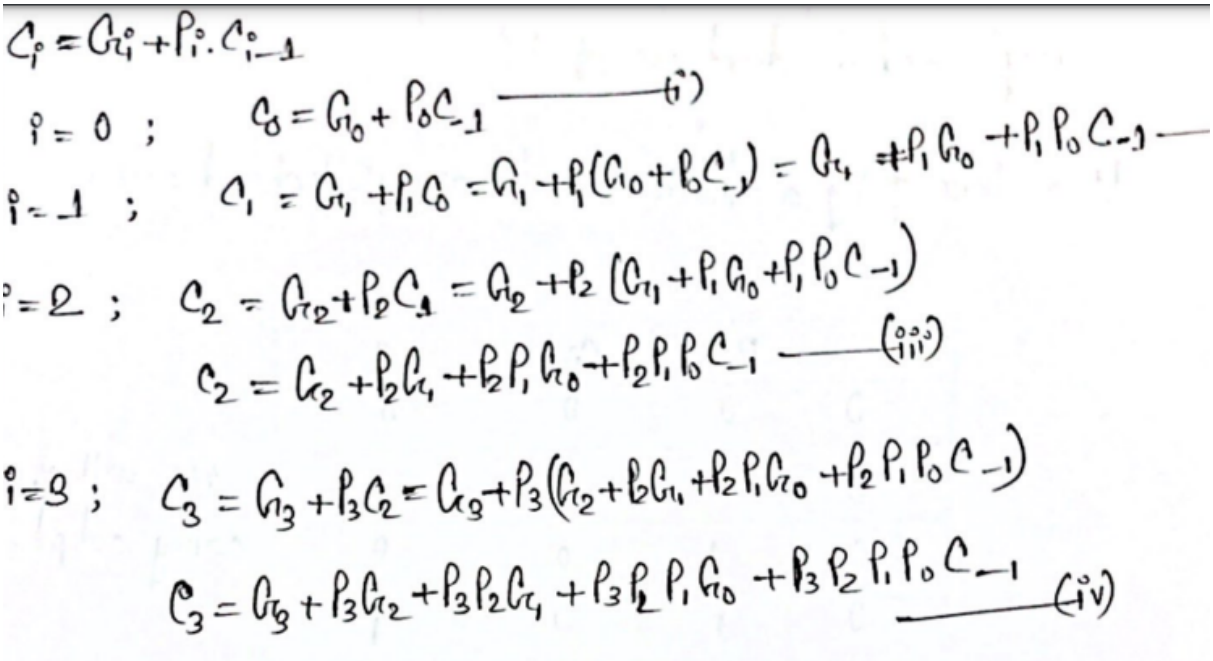
- **Procedure level:** here parallelism is available in the form of parallel executable procedures. In this process the design of the algorithm plays a major role. Parallelism at this level is performed using the operating system, but it needs a good programming that helped to perform and analyze this type of parallelism.
- **Program level :** This is usually the responsibility of the operating system which runs processes concurrently. Different programs are obviously independent of each other. So, parallelism can be extracted by operating of the system at this level.

- **Differences between SDRAM and DDR SDRAM**

SDRAM	DDR SDRAM
Synchronous DRAM	Double Data Rate SDRAM
168 pins and 2 notches	184 pins and 1 notch
released in 1997	released in 2000
3.3 volts requred	2.5 volts as standard, 1.8 volt as low voltage
Clock speed ranges between 100 to 166 MHz	130 to 200 MHz
prefetch timing 1 ns	2 ns
completion of previous read/write operation to continue other operation	does not wait for completion of previous read and write operation

- **Derive the equation of 4-bit carry look ahead adder. What are the advantage and disadvantage of it?**





Advantage

- It generates the carry-in for each fill adder simultaneously

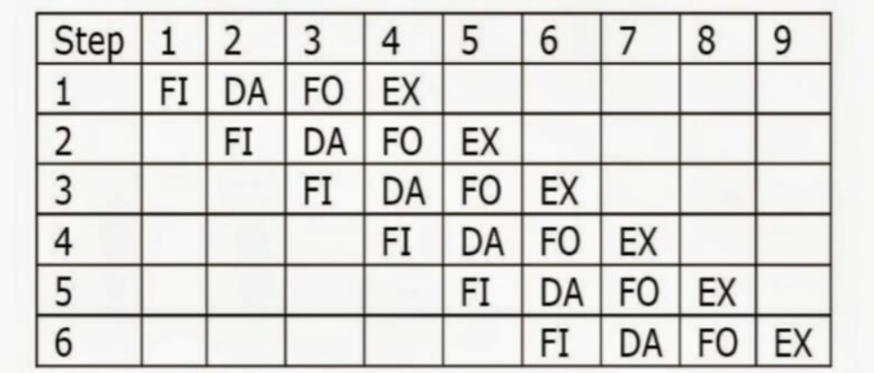
- It reduces the propagation delay
- It is fastest

Disadvantages

- Complex hardware
- Costlier
- It gets more complicated as the number of bits increase.

-

Instruction Pipelining without no branch operations



Step	1	2	3	4	5	6	7	8	9
1	FI	DA	FO	EX					
2		FI	DA	FO	EX				
3			FI	DA	FO	EX			
4				FI	DA	FO	EX		
5					FI	DA	FO	EX	
6						FI	DA	FO	EX