UNIT IV: CPU Organization

FACULTY

Dr. K. ARTHI MCA, M.Phil., Ph.D.,
Assistant Professor,
Postgraduate Department of Computer Applications,
Government Arts College (Autonomous),
Coimbatore-641018.

General Register Organization:

The number of registers in a processor unit may vary from just one processor register to as many as 64 registers or more.

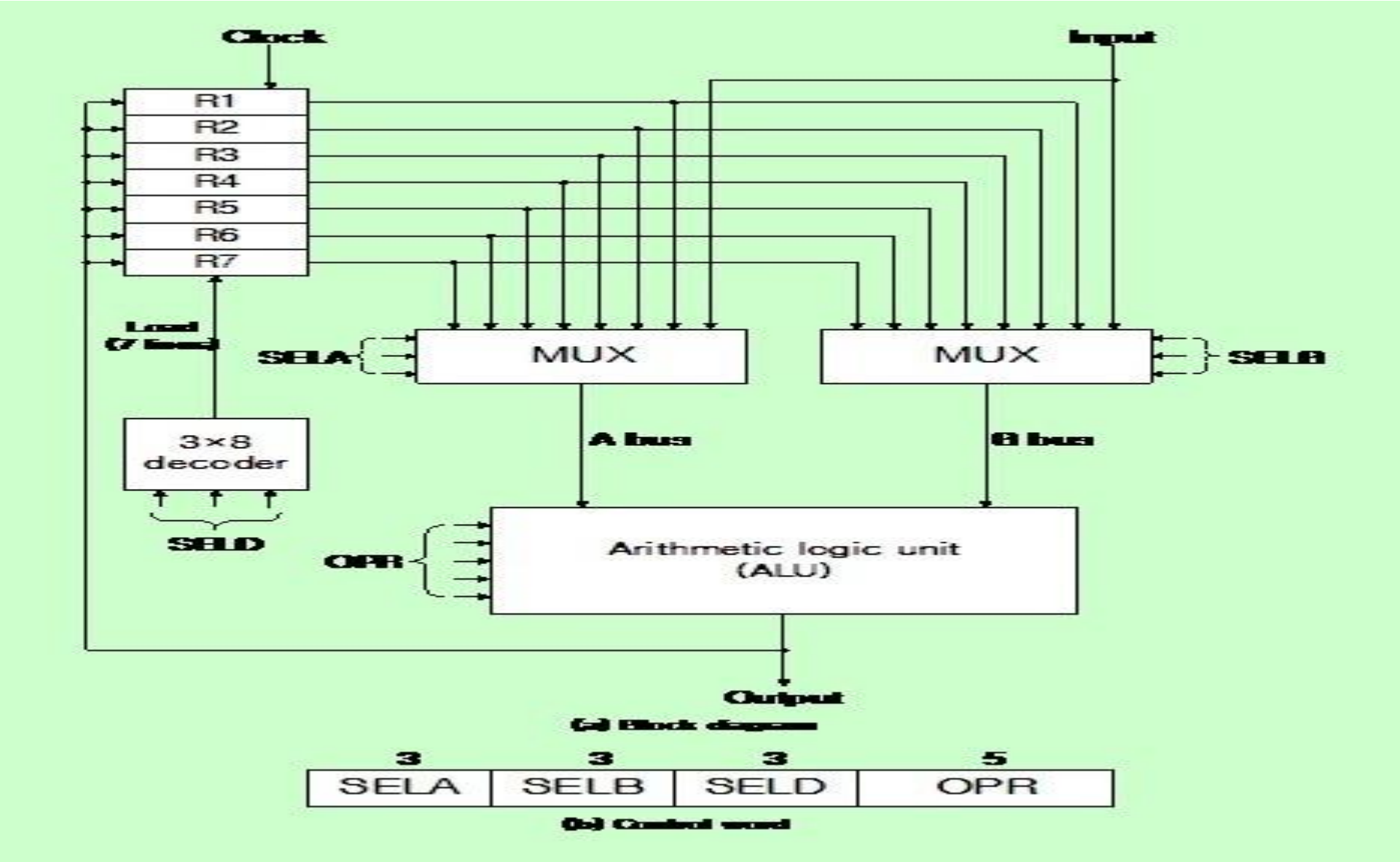
1.One of the CPU registers is called as an accumulator AC or 'A' register. It is the main operand register of the ALU.

2.The data register (DR) acts as a buffer between the CPU and main memory. It is used as an input operand register with the accumulator.

3.The instruction register (IR) holds the opcode of the current instruction.

4.The address register (AR) holds the address of the memory in which the operand resides. The program counter (PC) holds the address of the next instruction to be fetched for execution.
Additional addressable registers can be provided for storing operands and address. This can be viewed as replacing the single accumulator by a set of registers. If the registers are used for many purpose, the resulting computer is said to have general register organization. In the case of processor registers, a registers is selected by the multiplexers that form the buses.
When a large number of registers are included in the CPU, it is most efficient to connect them through a common bus system. The registers communicate with each other not only for direct data transfers, but also while performing various micro-operations. Hence it is necessary to provide a common unit that can perform all the arithmetic, logic and shift

A Bus organization for seven CPU registers



The output of each register is connected to true multiplexer (mux) to form the two buses A & B. The selection lines in each multiplexer select one register or the input data for the particular bus. The A and B buses forms the input to a common ALU. The operation selected in the ALU determines the arithmetic or logic micro-operation that is to be performed. The result of the micro-operation is available for output and also goes into the inputs of the registers. The register that receives the information from the output bus is selected by a decoder. The decoder activates one of the register load inputs, thus providing a transfer both between the data in the output bus and the inputs of the selected destination register.

The control unit that operates the CPU bus system directs the information flow through the registers and ALU by selecting the various components in the systems.

R1 ® R2 + R3

(1) MUX A selection (SEC A): to place the content of R2 into bus A

(2) MUX B selection (sec B): to place the content of R3 into bus B

(3) ALU operation selection (OPR): to provide the arithmetic addition (A + B)

(4) Decoder destination selection (SEC D): to transfer the content of the output bus into R1

These form the control selection variables are generated in the control unit and must be available at the beginning of a clock cycle. The data from the two source registers propagate through the gates in the multiplexer and the ALU, to the output bus, and into the into of the destination registers, all during the clock cycle intervals.

Computer Architecture: Interrupts

Data transfer between the CPU and the peripherals is initiated by the CPU. But the CPU cannot start the transfer unless the peripheral is ready to communicate with the CPU. When a device is ready to communicate with the CPU, it generates an interrupt signal. A number of input-output devices are attached to the computer and each device is able to generate an interrupt request.
The main job of the interrupt system is to identify the source of the interrupt. There is also a possibility that several devices will request simultaneously for CPU communication. Then, the interrupt system has to decide which device is to be serviced first.

Priority Interrupt:

A priority interrupt is a system which decides the priority at which various devices, which generates the interrupt signal at the same time, will be serviced by the CPU. The system has authority to decide which conditions are allowed to interrupt the CPU, while some other interrupt is being serviced. Generally, devices with high speed transfer such as *magnetic disks* are given high priority and slow devices such as *keyboards* are given low priority.
When two or more devices interrupt the computer simultaneously, the computer services the device with the higher priority first.

Types of Interrupts:

There different types of interrupts

1. Hardware Interrupts:
2. Software Interrupts

Hardware Interrupts:

When the signal for the processor is from an external device or hardware then this interrupts is known as **hardware interrupt**.
Let us consider an example: when we press any key on our keyboard to do some action, then this pressing of the key will generate an interrupt signal for the processor to perform certain action. Such an interrupt can be of two types:

- **Maskable Interrupt**

The hardware interrupts which can be delayed when a much high priority interrupt has occurred at the same time.

- **Non Maskable Interrupt**

The hardware interrupts which cannot be delayed and should be processed by the processor immediately.

Software Interrupts:

The interrupt that is caused by any internal system of the computer system is known as a **software interrupt**. It can also be of two types:

- **Normal Interrupt**

The interrupts that are caused by software instructions are called **normal software interrupts**.

- **Exception**

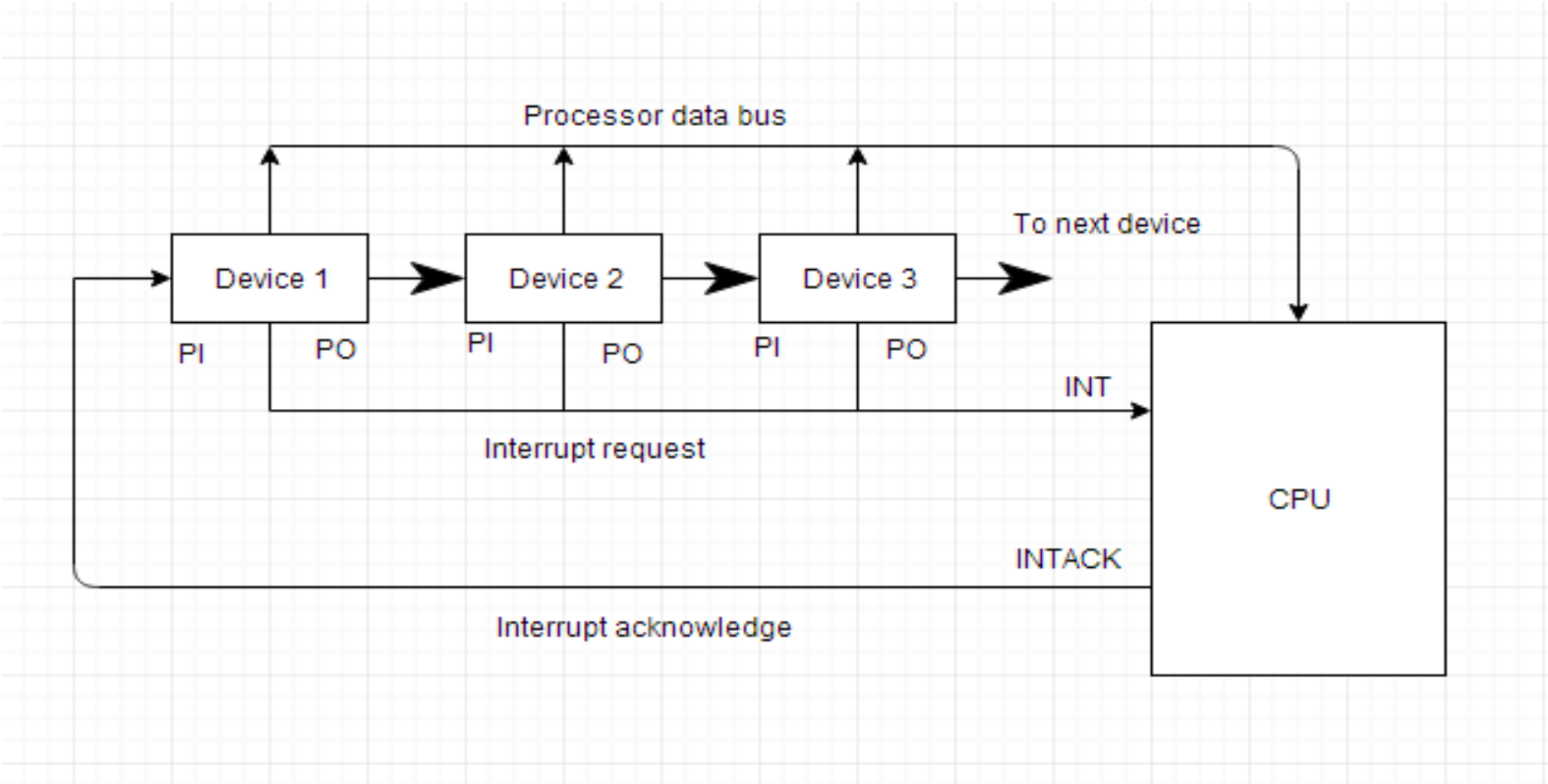
Unplanned interrupts which are produced during the execution of some program are called **exceptions**, such as division by zero.

Daisy Chaining Priority:

This way of deciding the interrupt priority consists of serial connection of all the devices which generates an interrupt signal. The device with the highest priority is placed at the first position followed by lower priority devices and the device which has lowest priority among all is placed at the last in the chain.

In daisy chaining system all the devices are connected in a serial form. The interrupt line request is common to all devices. If any device has interrupt signal in low level state then interrupt line goes to low level state and enables the interrupt input in the CPU. When there is no interrupt the interrupt line stays in high level state. The CPU respond to the interrupt by enabling the interrupt acknowledge line. This signal is received by the device 1 at its PI input. The acknowledge signal passes to next device through PO output only if device 1 is not requesting an interrupt.

The following figure shows the block diagram for daisy chaining priority system.

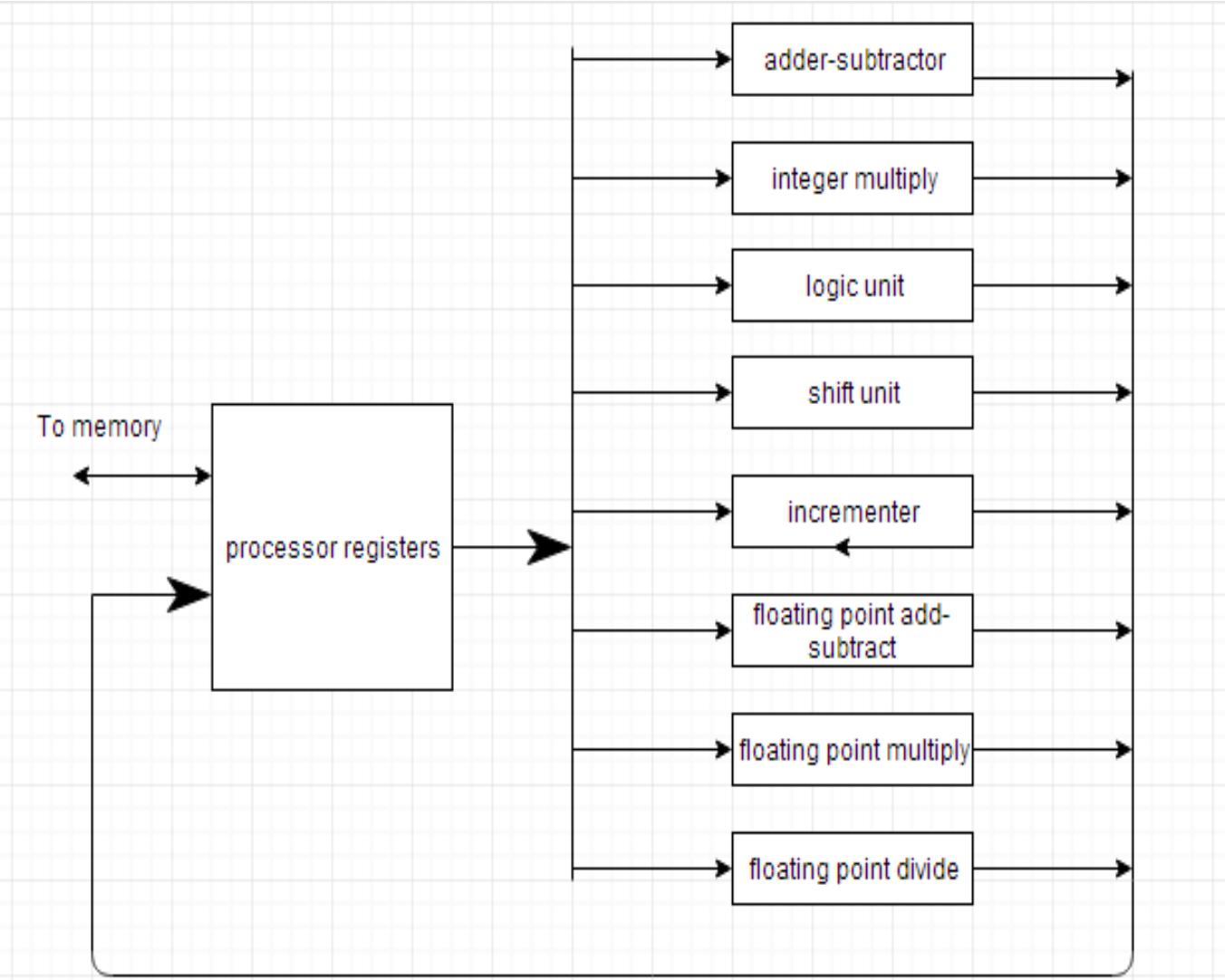


Parallel Processing and Data Transfer Modes in a Computer System:

Instead of processing each instruction sequentially, a **parallel processing** system provides concurrent data processing to increase the execution time.
In this the system may have two or more ALU's and should be able to execute two or more instructions at the same time. The purpose of parallel processing is to speed up the computer processing capability and increase its throughput.

NOTE: Throughput is the number of instructions that can be executed in a unit of time.

Parallel processing can be viewed from various levels of complexity. At the lowest level, we distinguish between parallel and serial operations by the type of registers used. At the higher level of complexity, parallel processing can be achieved by using multiple functional units that perform many operations simultaneously.



PROCESSOR WITH MULTIPLE FUNCTIONAL UNITS

Data Transfer Modes of a Computer System:

According to the data transfer mode, computer can be divided into 4 major groups:

1.SISD
2.SIMD
3.MISD
4.MIMD

SISD (Single Instruction Stream, Single Data Stream)

It represents the organization of a single computer containing a control unit, processor unit and a memory unit. Instructions are executed sequentially. It can be achieved by pipelining or multiple functional units.

SIMD (Single Instruction Stream, Multiple Data Stream)

It represents an organization that includes multiple processing units under the control of a common control unit. All processors receive the same instruction from control unit but operate on different parts of the data.

They are highly specialized computers. They are basically used for numerical problems that are expressed in the form of vector or matrix. But they are not suitable for other types of computations

MISD (Multiple Instruction Stream, Single Data Stream)

It consists of a single computer containing multiple processors connected with multiple control units and a common memory unit. It is capable of processing several instructions over single data stream simultaneously. MISD structure is only of theoretical interest since no practical system has been constructed using this organization.

MIMD (Multiple Instruction Stream, Multiple Data Stream

It represents the organization which is capable of processing several programs at same time. It is the organization of a single computer containing multiple processors connected with multiple control units and a shared memory unit. The shared memory unit contains multiple modules to communicate with all processors simultaneously. Multiprocessors and multicomputer are the examples of MIMD. It fulfills the demand of large scale computations

Pipelining:

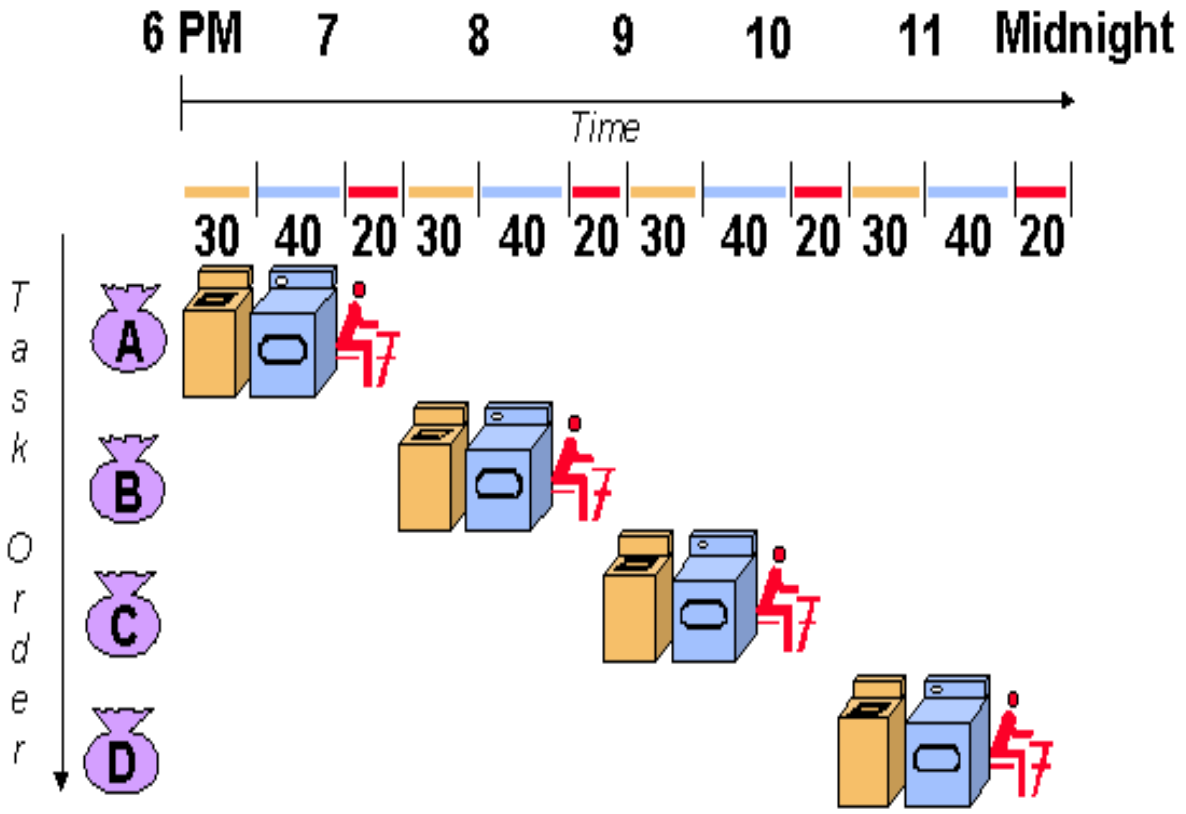
*Pipelining is the process of accumulating instruction from the processor through a pipeline. It allows storing and executing instructions in an orderly process. It is also known as **pipeline processing**

*Pipelining is a series of stages, where some work is done at each stage in parallel

*Pipelining is a technique where multiple instructions are overlapped during execution. Pipeline is divided into stages and these stages are connected with one another to form a pipe like structure. Instructions enter from one end and exit from another end

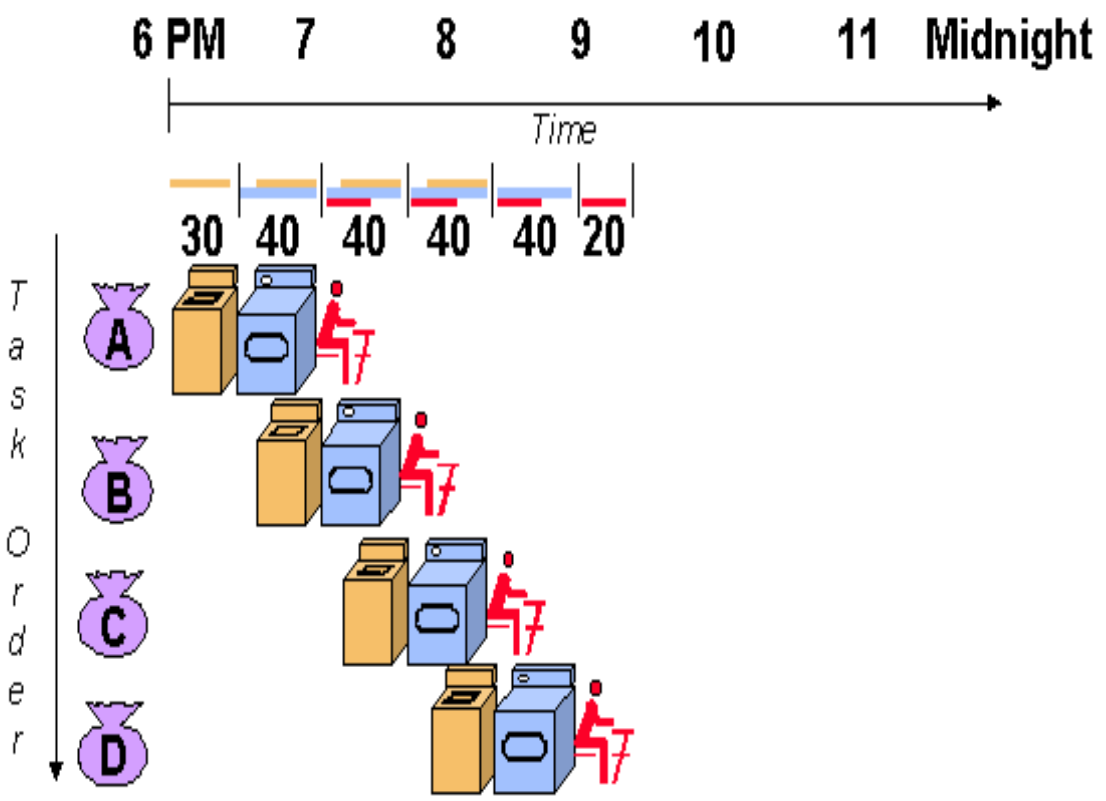
- 4 loads of laundry that need to washed, dried, and folded.
  - 30 minutes to wash, 40 min. to dry, and 20 min. to fold.
  - We have 1 washer, 1 dryer, and 1 folding station.
- What's the most efficient way to get the 4 loads of laundry done? Let's see

Non Pipelined Laundry:



- Takes a total of 6 hours; nothing is done in parallel

Pipelined Laundry:



- Using this method, the laundry would be done at 9:30.

Types of Pipeline:

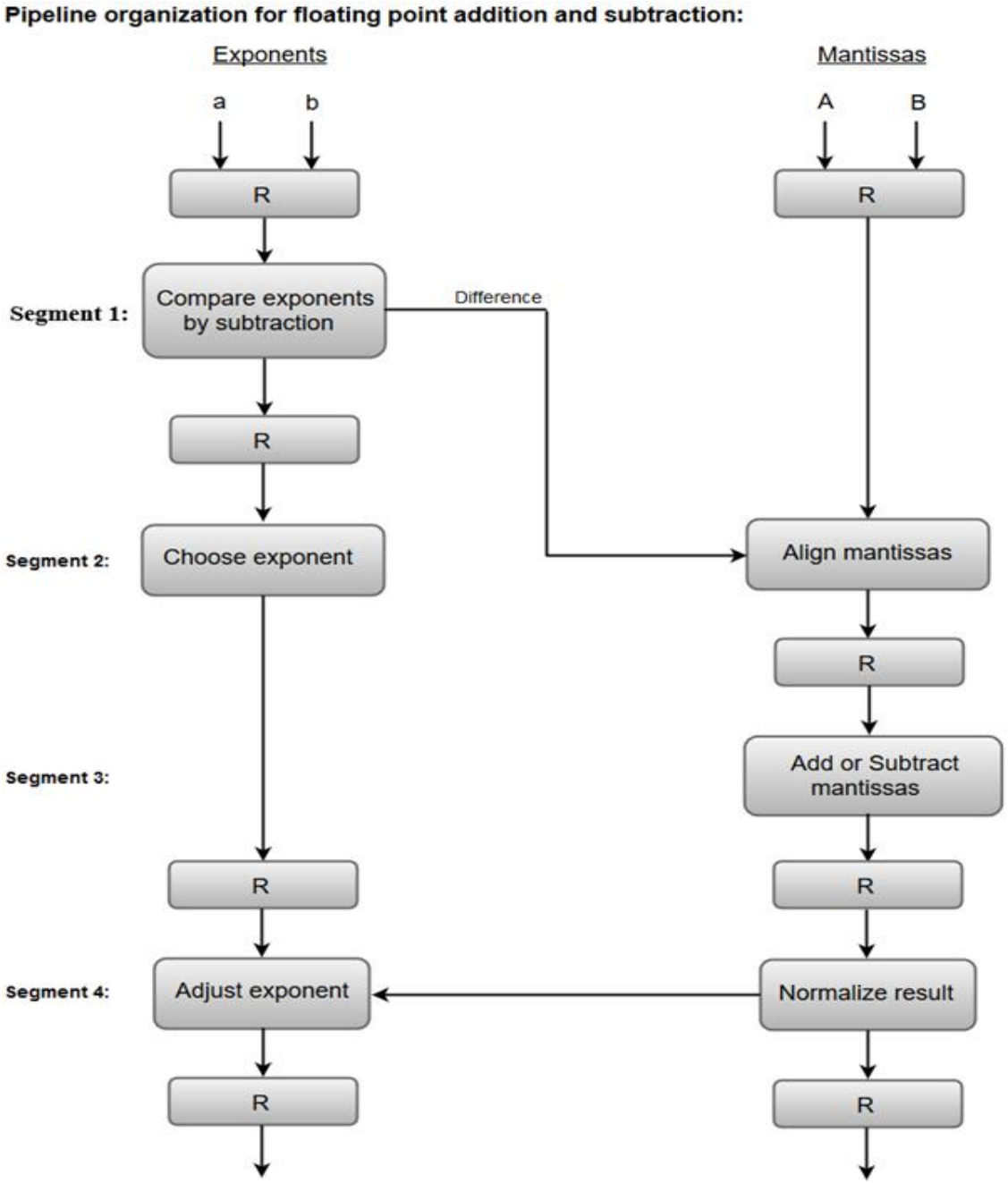
It is divided into 2 categories:

1.Arithmetic Pipeline
2.Instruction Pipeline

An arithmetic pipeline divides an arithmetic problem into various sub problems for execution in various pipeline segments. It is used for floating point operations, multiplication and various other computations. The process or flowchart arithmetic pipeline for floating point addition is shown in the diagram.

The following sub operations are performed in this case:
1.Compare the exponents.
2.Align the mantissas.
3.Add or subtract the mantissas.
4.Normalize the result

First of all the two exponents are compared and the larger of two exponents is chosen as the result exponent. The difference in the exponents then decides how many times we must shift the smaller exponent to the right. Then after shifting of exponent, both the mantissas get aligned. Finally the addition of both numbers take place followed by normalization of the result in the last segment.



1. Compare exponents by subtraction:

The exponents are compared by subtracting them to determine their difference. The larger exponent is chosen as the exponent of the result . The difference of the exponents, i.e., **3 -2 = 1** determines how many times the mantissa associated with the smaller exponent must be shifted to the right.

2. Align the mantissas:

The mantissa associated with the smaller exponent is shifted according to the difference of exponents determined in segment one.

$X = 0.9504 * 10^3$ $Y = 0.08200 * 10^3$

3. Add mantissas:

The two mantissas are added in segment three.

$Z = X + Y = 1.0324 * 10^3$

4. Normalize the result:

After normalization, the result is written as:

$Z = 0.1324 * 10^4$

Instruction Pipeline :

• Instruction execution process lends itself naturally to
Pipelining

– overlap the subtasks of instruction fetch, decode and execute

- Fetch instruction (FI)
- Decode instruction (DI)
- Calculate operands(CO)
- Fetch operands (FO)
- Execute instructions (EI)
- Write result (WR)

Instructions Fetch:

• The IF stage is responsible for obtaining the requested instruction from memory. The instruction and the program counter are stored in the register as temporary storage.

Decode Instruction:

• The DI stage is responsible for decoding the instruction and sending out the various control lines to the other parts of the processor.

Calculate Operands:

• The CO stage is where any calculations are performed.

The main component in this stage is the ALU. The ALU is made up of arithmetic, logic and capabilities

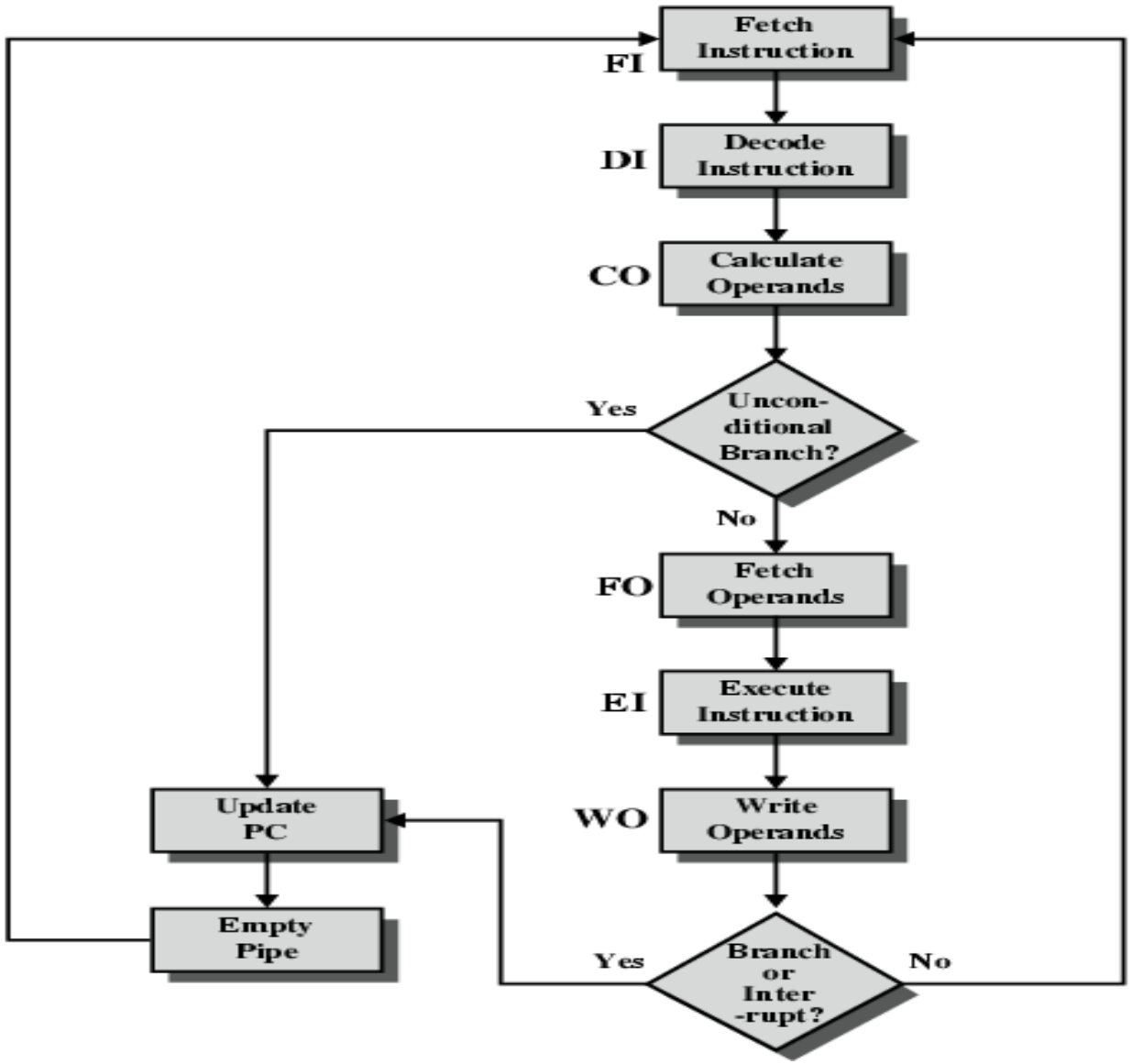
Fetch Operands and Execute Instruction:

• The FO and EI stages are responsible for storing and loading values to and from memory. They also responsible for input and output from the processor respectively.

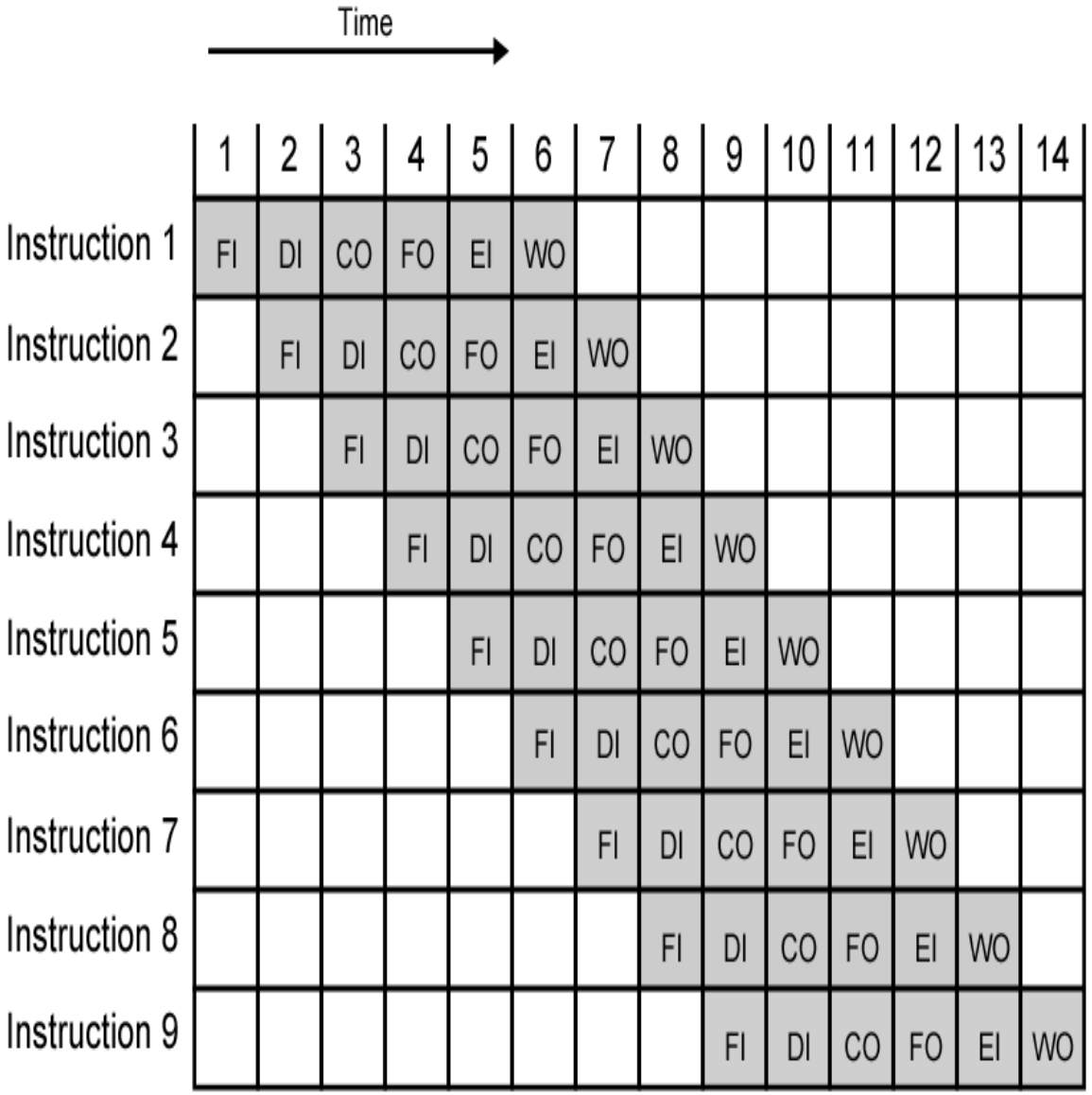
Write Operands:

• The WO stage is responsible for writing the result of a calculation, memory access or input into the register file.

Six Stage Instruction Pipeline :



Timing Diagram for Instruction Pipeline Operation :



Advantages:

- Pipelining makes efficient use of resources.
- Quicker time of execution of large number of instructions
- The parallelism is invisible to the programmer.

Can Pipelining **Get Us Into Trouble?**

Yes: Pipeline Hazards

structural hazards: attempt to use the same resource by two different instructions at the same time.

data hazards: attempt to use data before it is ready. An instruction's source operand(s) are produced by a prior instruction still in the pipeline.

control hazards: attempt to make a decision about program control flow before the condition has been evaluated and the new PC target address calculated

Vector(Array) Processor and its Types:

Array processors are also known as multiprocessors or vector processors. They perform computations on large arrays of data. Thus, they are used to improve the performance of the computer.

Types of Array Processors:

There are basically two types of array processors:

1.Attached Array Processors
2.SIMD Array Processor

To improve the performance of the host computer in numerical computational tasks auxiliary processor is attached to it.

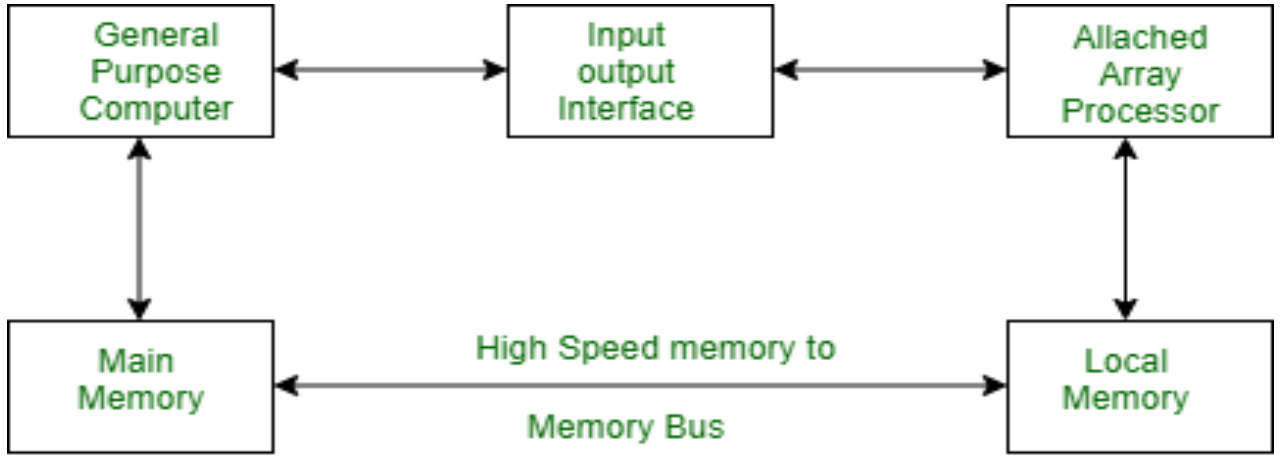


Figure - Interconnection of an attached array Processor to a host computer

Attached array processor has two interfaces:

1.Input output interface to a common processor.
2.Interface with a local memory.

Here local memory interconnects main memory. Host computer is general purpose computer. Attached processor is back end machine driven by the host computer.

The array processor is connected through an I/O controller to the computer & the computer treats it as an external interface.

SIMD is the organization of a single computer containing multiple processors operating in parallel. The processing units are made to operate under the control of a common control unit, thus providing a single instruction stream and multiple data streams.

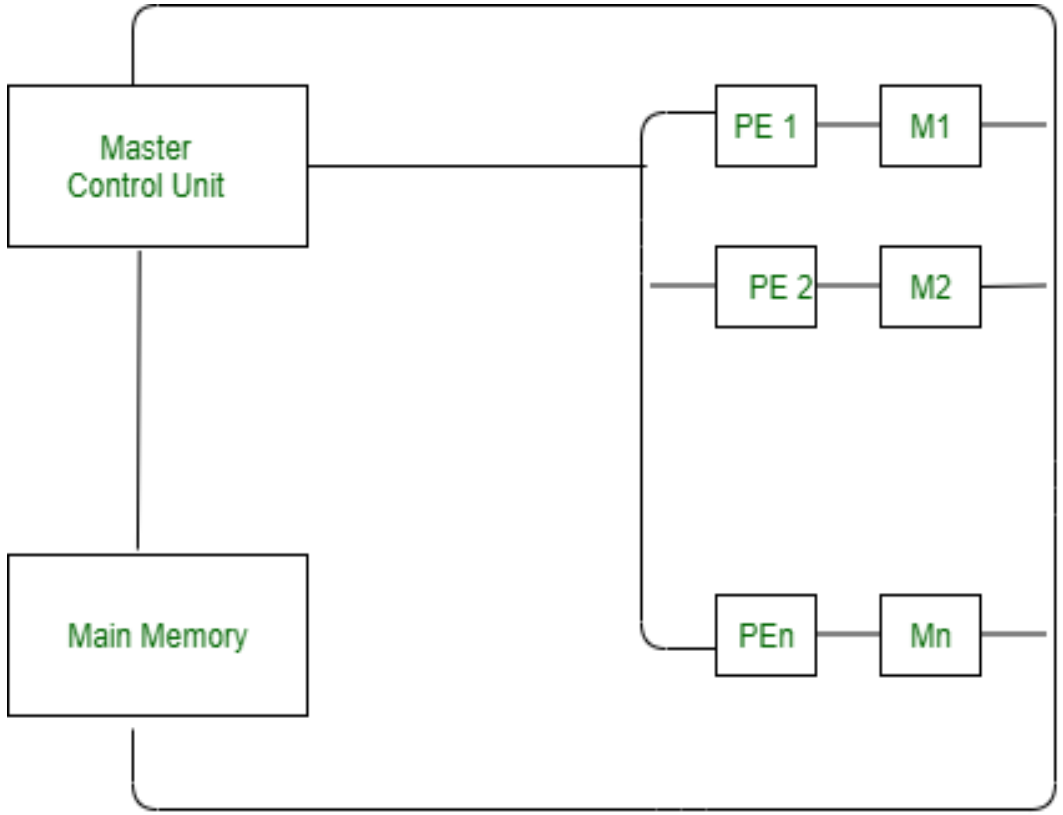


Figure - SIMD Array Processor Organization

A general block diagram of an array processor is shown below. It contains a set of identical processing elements (PE's), each of which is having a local memory M. Each processor element includes an **ALU** and **registers**. The master control unit controls all the operations of the processor elements. It also decodes the instructions and determines how the instruction is to be executed.
The main memory is used for storing the program. The control unit is responsible for fetching the instructions. Vector instructions are send to all PE's simultaneously and results are returned to the memory.
The best known SIMD array processor is the **ILLIAC IV** computer developed by the **Burroughs corps**. SIMD processors are highly specialized computers. They are only suitable for numerical problems that can be expressed in vector or matrix form and they are not suitable for other types of computations.

PERIPHERAL DEVICES

Input Devices

- Keyboard
- Optical input devices
  - Card Reader
  - Paper Tape Reader
  - Bar code reader
  - Digitizer
  - Optical Mark Reader
- Magnetic Input Devices
  - Magnetic Stripe Reader
- Screen Input Devices
  - Touch Screen
  - Light Pen
  - Mouse
- Analog Input Devices

Output Devices

- Card Puncher, Paper Tape Puncher
- CRT
- Printer (Impact, Ink Jet, Laser, Dot Matrix)
- Plotter
- Analog
- Voice

INPUT/OUTPUT INTERFACES

* Provides a method for transferring information between internal storage (such as memory and CPU registers) and external I/O devices

* Resolves the *differences* between the computer and peripheral devices

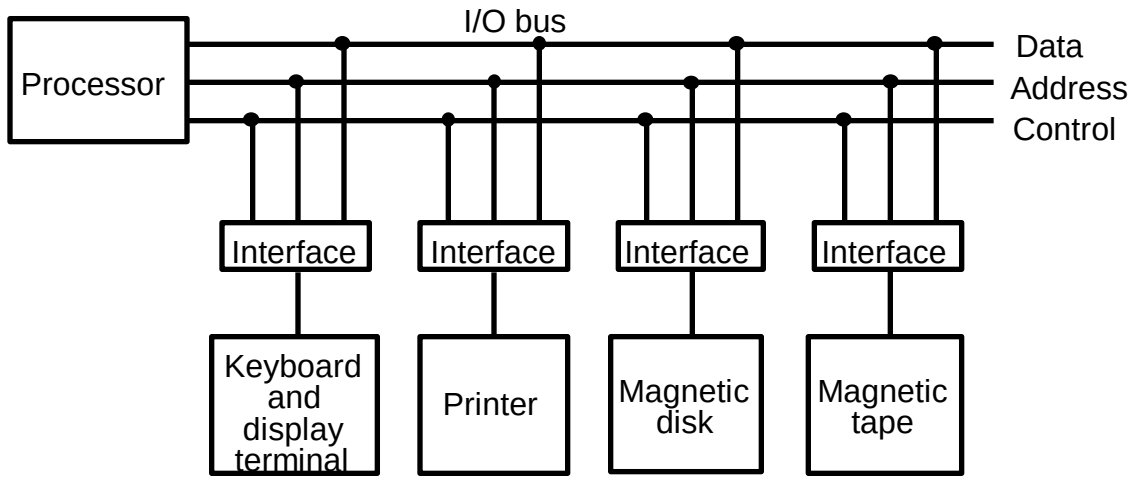
- Peripherals - Electromechanical Devices
CPU or Memory - Electronic Device

- Data Transfer Rate
Peripherals - Usually slower
CPU or Memory - Usually faster than peripherals
Some kinds of Synchronization mechanism may be needed

- Unit of Information
Peripherals - Byte
CPU or Memory - Word

- Operating Modes
Peripherals - Autonomous, Asynchronous
CPU or Memory - Synchronous

I/O BUS AND INTERFACE MODULES

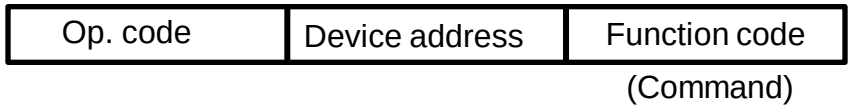


Each peripheral has an interface module associated with it

Interface

- Decodes the device address (device code)
- Decodes the commands (operation)
- Provides signals for the peripheral controller
- Synchronizes the data flow and supervises the transfer rate between peripheral and CPU or Memory

Typical I/O instruction



I/O BUS AND MEMORY BUS

Functions of Buses

* *MEMORY BUS* is for information transfers between CPU and the MM

* *I/O BUS* is for information transfers between CPU and I/O devices through their I/O interface

Physical Organizations

* Many computers use a common single bus system for both memory and I/O interface units

  - Use one common bus but separate control lines for each function
  - Use one common bus with common control lines for both functions

* Some computer systems use two separate buses, one to communicate with memory and the other with I/O interfaces

I/O Bus

- Communication between CPU and all interface units is via a common I/O Bus
- An interface connected to a peripheral device may have a number of *data registers* , a *control register*, and a *status register*
- A command is passed to the peripheral by sending to the appropriate interface register
- Function code and sense lines are not needed (Transfer of data, control, and status information is always via the common I/O Bus)

Synchronous and Asynchronous Operations

Synchronous - All devices derive the timing information from common clock line

Asynchronous - No common clock

Asynchronous Data Transfer

Asynchronous data transfer between two independent units requires that *control signals* be transmitted between the communicating units *to indicate the time at which data is being transmitted*

Two Asynchronous Data Transfer Methods

Strobe pulse

- A strobe pulse is supplied by one unit to indicate the other unit when the transfer has to occur

Handshaking

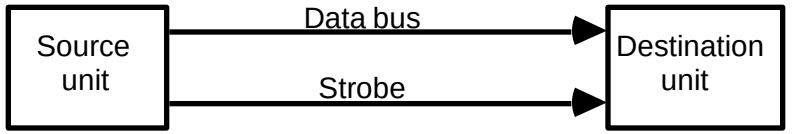
- A control signal is accompanied with each data being transmitted to indicate the presence of data
- The receiving unit responds with another control signal to acknowledge receipt of the data

STROBE CONTROL

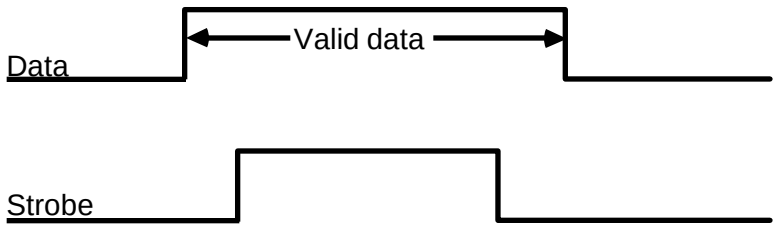
* Employs a single control line to time each transfer
* The strobe may be activated by either the source or the destination unit

Source-Initiated Strobe for Data Transfer

Block Diagram

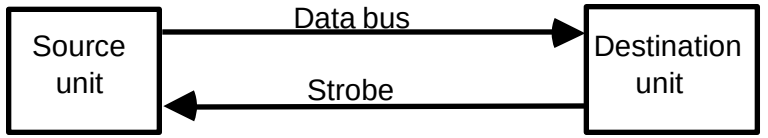


Timing Diagram

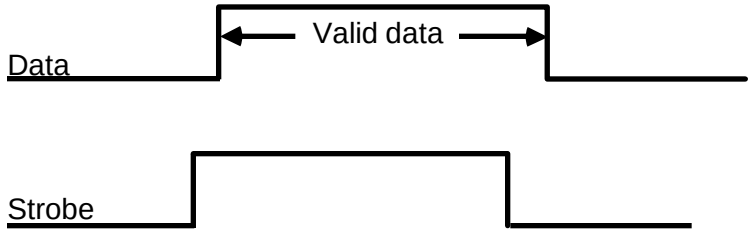


Destination-Initiated Strobe for Data Transfer

Block Diagram



Timing Diagram



HANDSHAKING

Strobe Methods

Source-Initiated

The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received data

Destination-Initiated

The destination unit that initiates the transfer no way of knowing whether the source has actually placed the data on the bus

To solve this problem, the *HANDSHAKE* method introduces a second control signal to provide a *Reply* to the unit that initiates the transfer

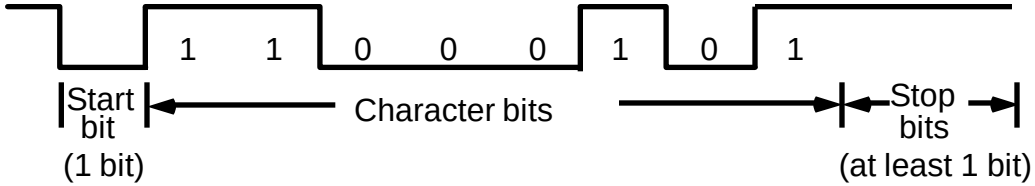
ASYNCHRONOUS SERIAL TRANSFER

Four Different Types of Transfer

Asynchronous serial transfer
Synchronous serial transfer
Asynchronous parallel transfer
Synchronous parallel transfer

Asynchronous Serial Transfer

- Employs special bits which are inserted at both ends of the character code
- Each character consists of three parts; Start bit; Data bits; Stop bits.



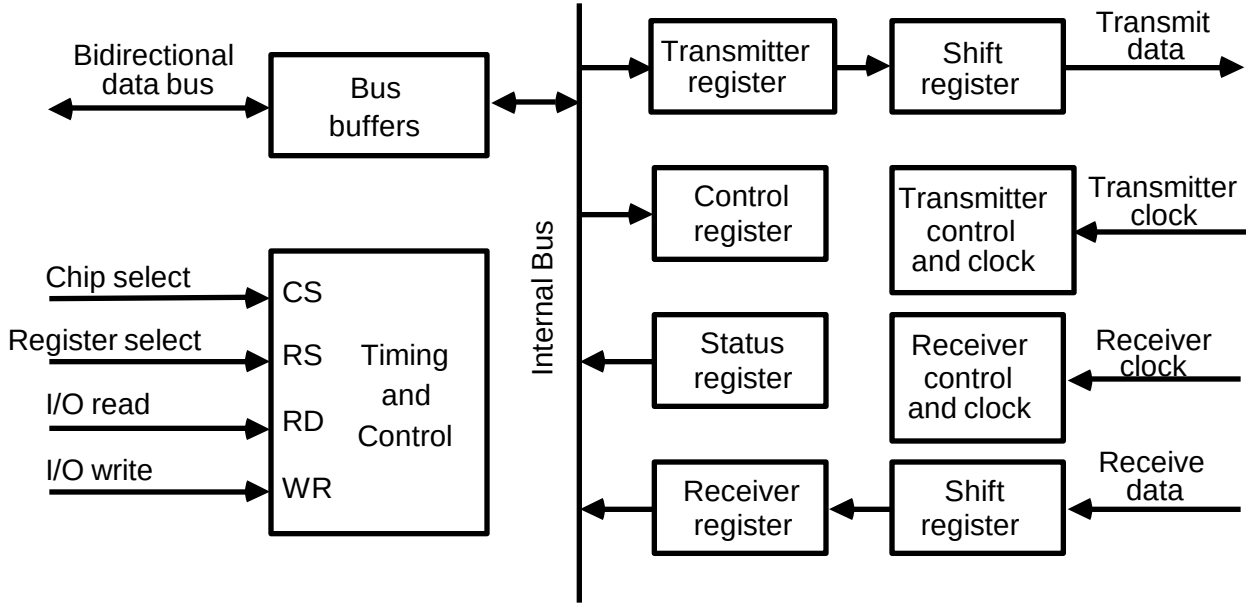
A character can be detected by the receiver from the knowledge of 4 rules;

- When data are not being sent, the line is kept in the 1-state (idle state)
- The initiation of a character transmission is detected by a *Start Bit* , which is always a 0
- The character bits always follow the *Start Bit*
- After the last character , a *Stop Bit* is detected when the line returns to the 1-state for at least 1 bit time

The receiver knows in advance the transfer rate of the bits and the number of information bits to expect

UNIVERSAL ASYNCHRONOUS RECEIVER-TRANSMITTER - UART -

A typical asynchronous communication interface available as an IC



CS	RS	Oper.	Register selected
0	x	x	None
1	0	WR	Transmitter register
1	1	WR	Control register
1	0	RD	Receiver register
1	1	RD	Status register

Transmitter Register
- Accepts a data byte(from CPU) through the data bus
- Transferred to a shift register for serial transmission

Receiver
- Receives serial information into another shift register
- Complete data byte is sent to the receiver register

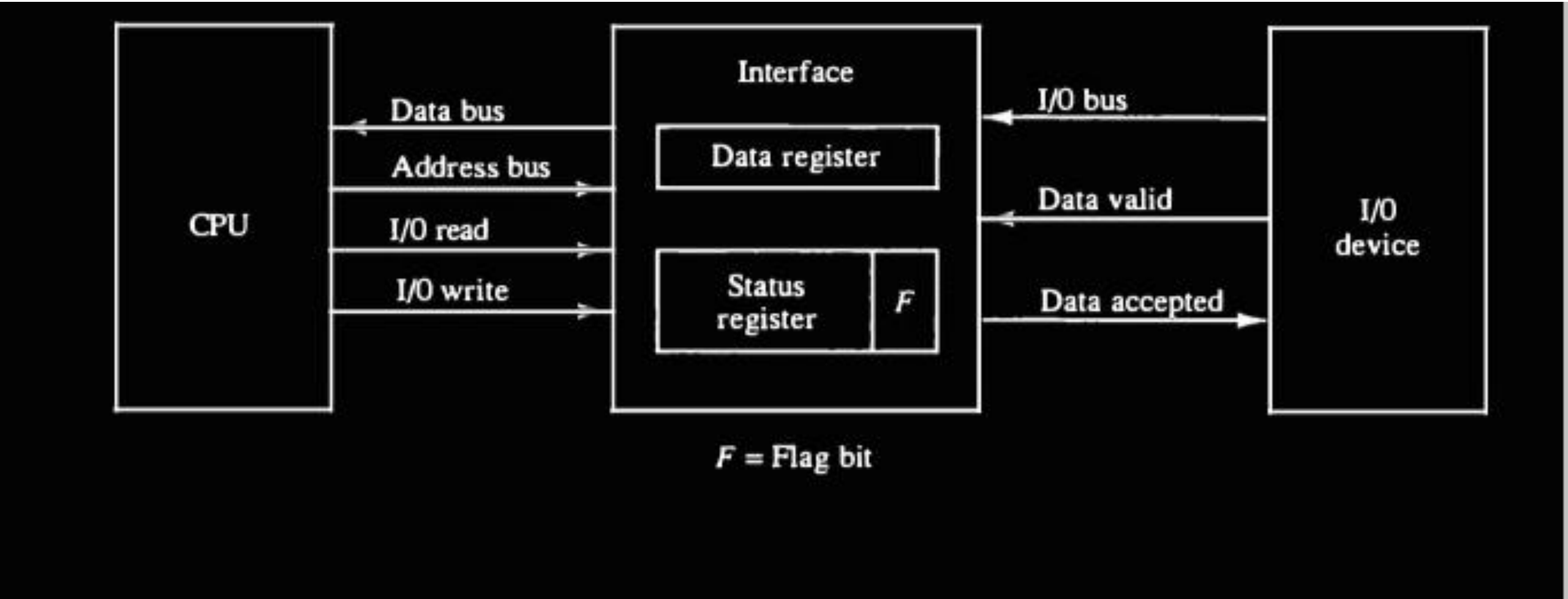
Status Register Bits
- Used for I/O flags and for recording errors

Control Register Bits
- Define baud rate, no. of bits in each character, whether to generate and check parity, and no. of stop bits

Programmed i/o

- These operation are a results of i/o instruction written in the computer program. Data transfer in initiated by an instruction in the program.
- Usually the data transfer data between CPU register and peripheral device. Other instruction are used to transfer data between CPU and memory.
- CPU peripheral has to be constantly.

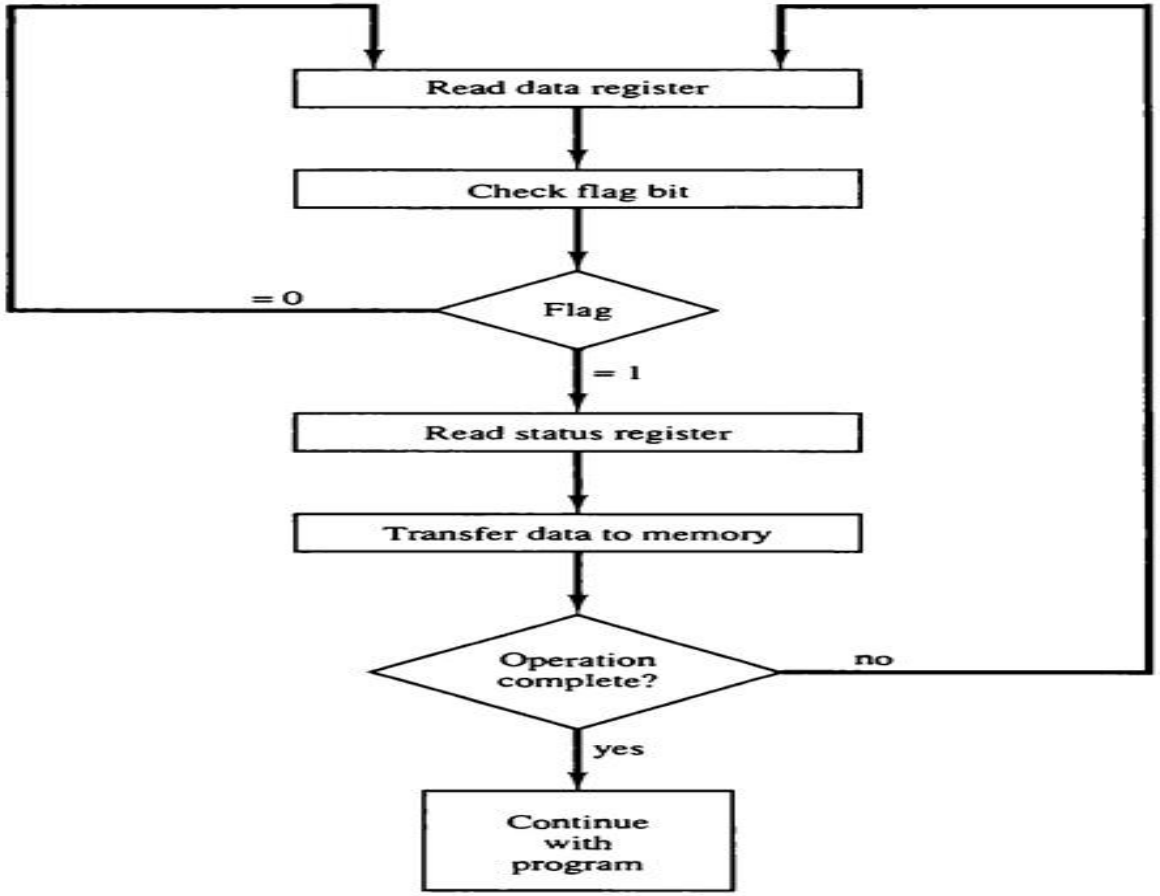
Data transfer from i/o device to CPU



Direct memory access (DMA)

- The interface transfer data into out of the memory unit through the memory bus.
- The CPU initiates the transfer of supplying the interface with the starting address and the number of words needed to be transferred and then proceed to execute other tasks.
- When the request is granted by memory controller, dma transfer data directly into memory.

CPU program to input data



Priority interrupt

- Data transfer between the CPU and an i/o device is initiated is initiated by the CPU. However, the CPU cannot start the transfer unless the device is ready to communication with the CPU.
- Higher priority interrupts can make requests while servicing a lower priority interrupts

Daisy chaining priority

- The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt.
- The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain.
- The interrupt request line is common to all devices and forms a wired logic connection.

Parallel priority interrupt

- The parallel priority interrupt method uses a register whose bits are set separately by the interrupt signal from each device.
- Priority is established according to the position of the bits in the register.
- The mask register can be programmed to disable lower-priority interrupts while a higher-priority device is being serviced.

Software routines

- A priority interrupt system is a combination of hardware and software techniques.
- So far we have discussed the hardware aspects of a priority interrupt system.
- The computer must also have software routines for servicing the interrupt requests and for controlling the interrupt hardware registers.

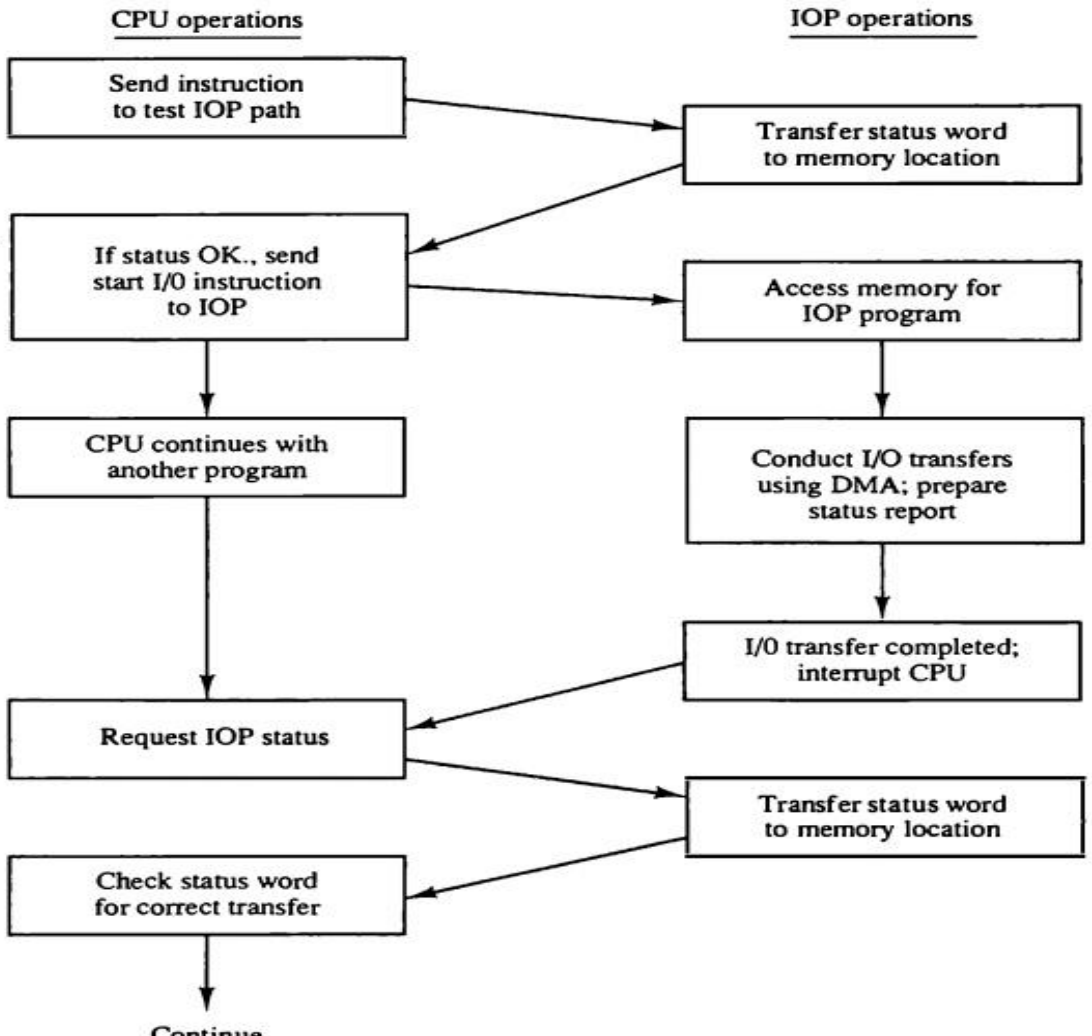
Initial and final operation

- Each interrupt service routine must have an initial and final set of operations for controlling the registers in the hardware interrupt system.
- Remember that the interrupt enable lEN is cleared at the end of an interrupt cycle.
- This flip-flop must be set again to enable higher-priority interrupt requests, but not before lower-priority interrupts are disabled.

i/o processor

- Many computer combines the interface logic with the requirements for direct memory access into one unit and call it an i/o processer.
- The iop can handle many peripherals through a dma and interrupt facility.
- The computer is divided into three separate modules in such a system.
  - memory unit
  - CPU
  - iop
- Cpu is the master while the iop is a slave processor. The cpu performs the tasks of initiating all operations

CPU- iop communication



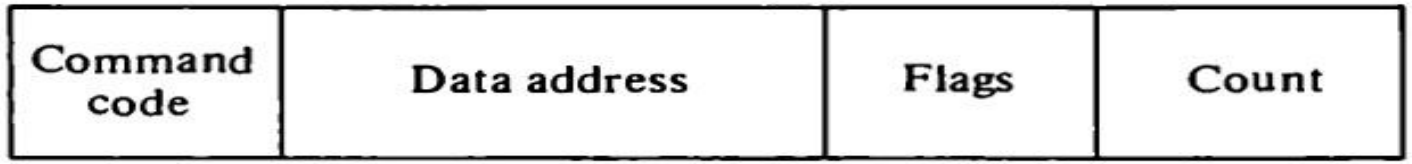
IBM 370 i/o channel

- The VO processor in the IBM 370 computer is called a channel.
- A typical computer system configuration includes a number of channels with each channel attached to one or more VO devices.
- There are three types of channels
  - multiplexer
  - Selector
  - Block-multiplexer

IBM 370 i/o related word formats

(a) I/O instruction format

(b) Channel status word format

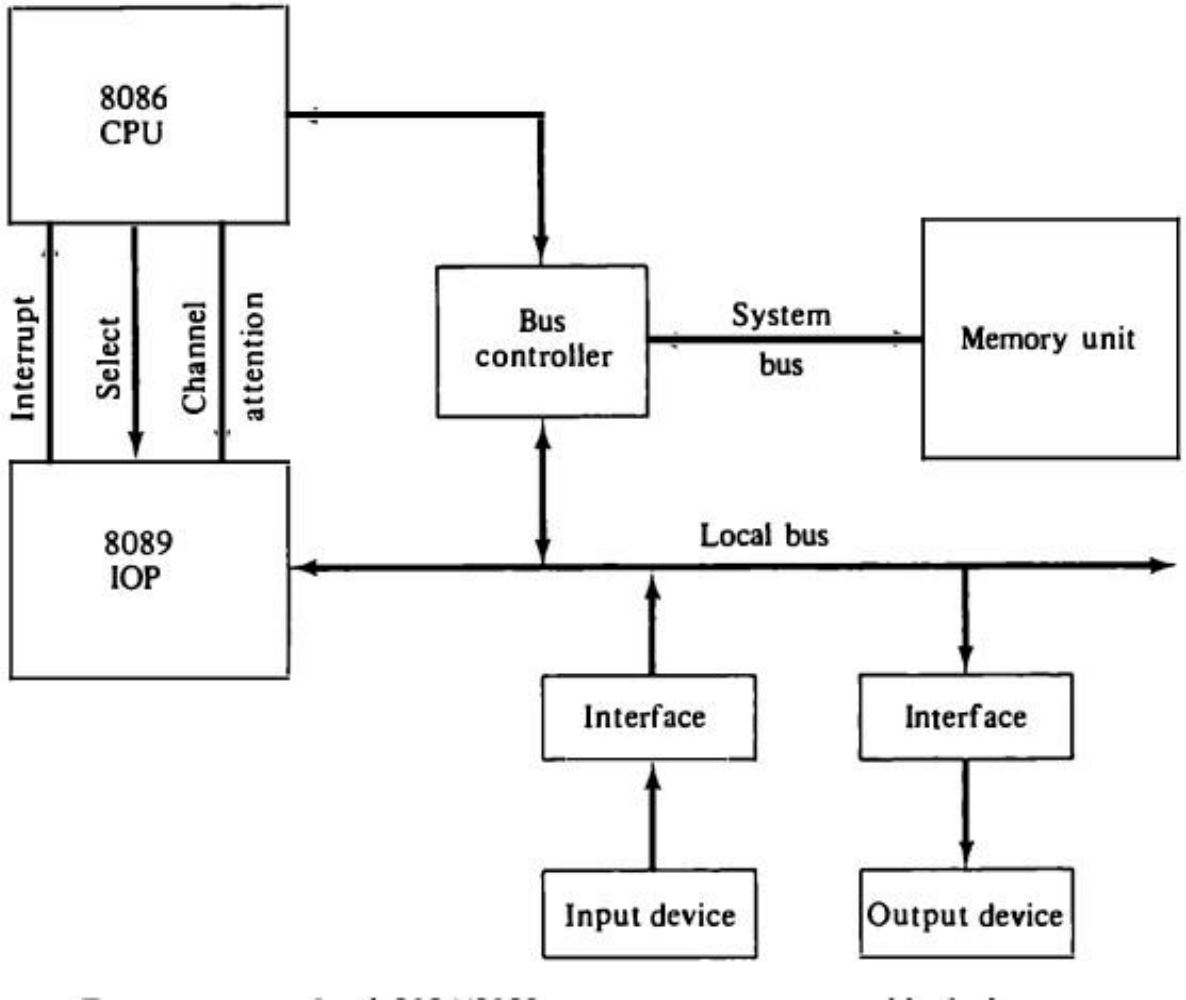


(c) Channel command word format

Intel 8089 iop

- The Intel8089 l/0 processor is contained in a 40-pin integrated circuit package. Within the 8089 are two independent units called channels.
- Each channel combines the general characteristics of a processor unit with those of a direct memory access controller.
- A microcomputer system using the Intel 8086/8089 pair of integrated circuits is shown in diagram.

Intel 8086/8089 micro computer system block diagram



THANK YOU

This content is taken from the text books and reference books prescribed in the syllabus.