**UNIT-V**

**INPUT-OUTPUT ORGANIZATION**

**CONTENTS:**

* + Peripheral devices
  + Input/Output Interface
  + Asynchronous Data Transfer
  + Modes Of Transfer
  + Priority Interrupt
  + DMA
  + Input-Output Processor (IOP)

**Peripheral Devices**

* Input or output devices that are connected to computer are called **peripheral devices**.
* For example: *Keyboards*, *display units* and *printers* are common peripheral devices.

There are three types of peripherals:

* **Input peripherals** : Allows user input, from the outside world to the computer. Example: Keyboard, Mouse etc.
* **Output peripherals**: Allows information output, from the computer to the outside world. Example: Printer, Monitor etc
* **Input-Output peripherals**: Allows both input(from outisde world to computer) as well as, output(from computer to the outside world). Example: Touch screen etc.

**Input - Output Interface**

* Input Output Interface provides a method for transferring information between internalstorage and external I/O devices.
* Peripherals connected to a computer need special communication links for interfacing themwith the central processing unit.
* The purpose of communication link is to resolve the differences that exist between thecentral computer and each peripheral.

The Major Differences are:-

1. Peripherals are electromechnical and electromagnetic devices and CPU and

memory are electronic devices. Therefore, a conversion of signal values may be

needed.

2. The data transfer rate of peripherals is usually slower than the transfer rate of CPU

and consequently, a synchronization mechanism may be needed.

3. Data codes and formats in the peripherals differ from the word format in the CPU and

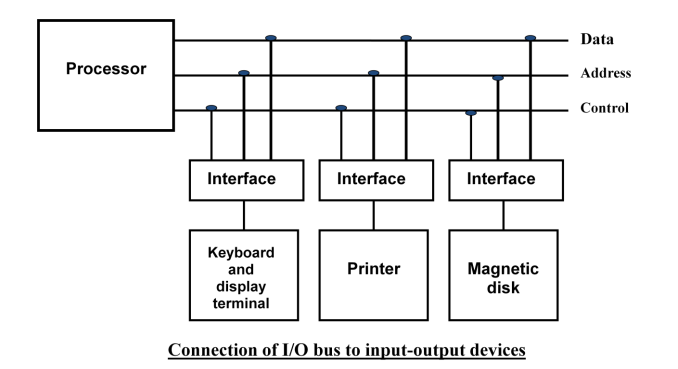
Memory

* The I/O Bus consists of data lines, address lines and control lines.
* The I/O bus from the processor is attached to all peripherals interface.
* To communicate with a particular device, the processor places a device address on address lines.
* Each peripheral has its own controller.
* For example, the printer controller controls the paper motion, the print timing
* The control lines are referred as I/O command. The commands are as following:
* Control command- A control command is issued to activate the peripheral and to inform itwhat to do.
* Status command- A status command is used to test various status conditions in the interfaceand the peripheral.
* Data Output command- A data output command causes the interface to respond by

transferring data from the bus into one of its registers.

* Data Input command- The data input command is the opposite of the data output.

In this case the interface receives on item of data from the peripheral and places it in itsbuffer register



**I/O Versus Memory Bus**

To communicate with I/O, the processor must communicate with the memory unit. Like the

I/O bus, the memory bus contains data, address and read/write control lines. There are 3 ways

that computer buses can be used to communicate with memory and I/O:

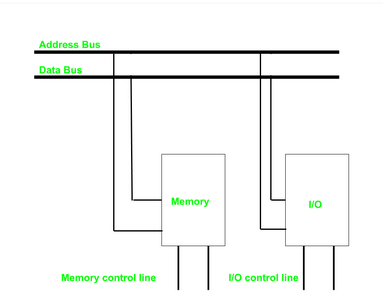
i. Use two Separate buses , one for memory and other for I/O.

ii. Use one common bus for both memory and I/O but separate control lines for each.

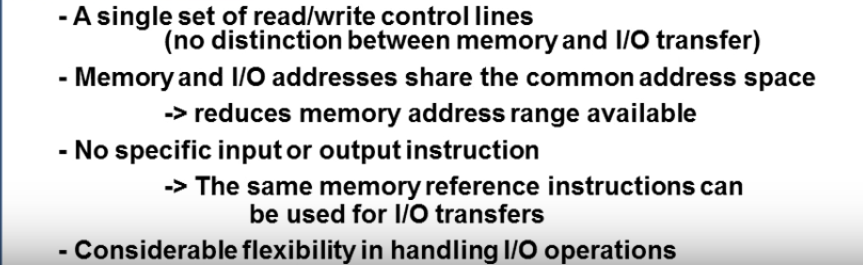
iii. Use one common bus for memory and I/O with common control lines.

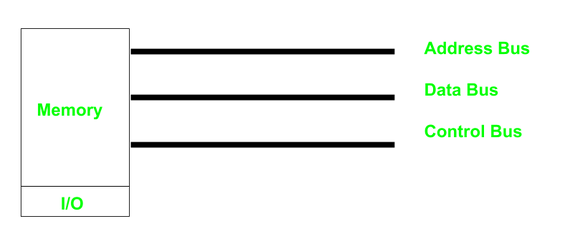
**Isolated I/O –**

* In Isolated I/O,the CPU has distinct input and output instructions and each of these instruction is associated with address of an interface register.
* Have common bus(data and address) for I/O and memory but separate read and write control lines for I/O.
* when CPU decode instruction then if data is for I/O then it places the address on the address line and set I/O read or write control line on due to which data transfer occurs between CPU and I/O.

****

**MEMORY MAPPED I/O**

****

****

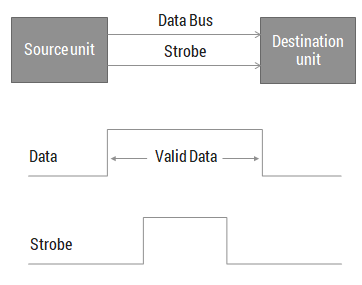
**Asynchronous Data Transfer**

If the registers in the interface share a common clock with cpuregisters,the transfer between the two is said to be synchronous

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

**Strobe Method**

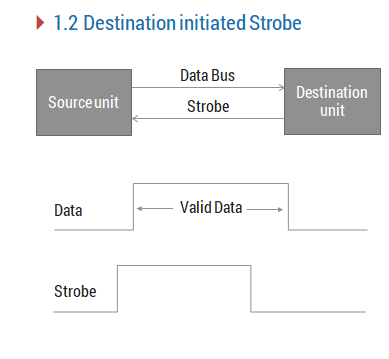
* **Source initiated Strobe**

****

When source initiates the process of data transfer. Strobe is just a signal.

(i) First, source puts data on the data bus and ON the strobe signal.  
(ii) Destination on seeing the ON signal of strobe, read data from the data bus.  
(iii) After reading data from the data bus by destination, strobe gets OFF.

**Destination initiated strobe**

****

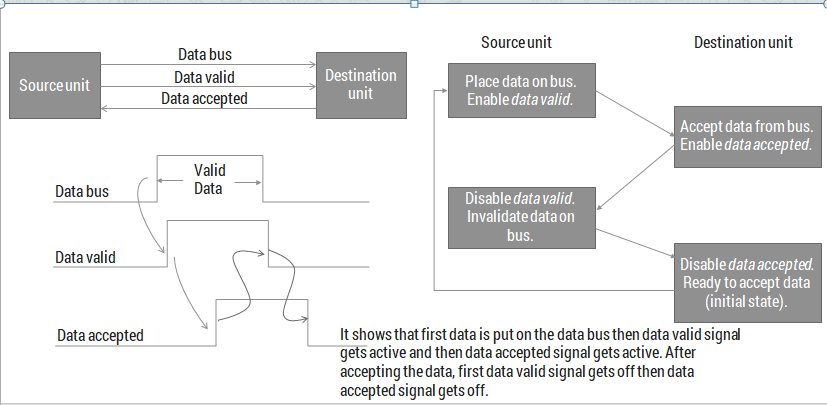
 i)First, the destination ON the strobe signal to ensure the source to put the fresh data on the data bus.  
(ii) Source on seeing the ON signal puts fresh data on the data bus.  
(iii) Destination reads the data from the data bus and strobe gets OFF signal.

**Disadvantage:**

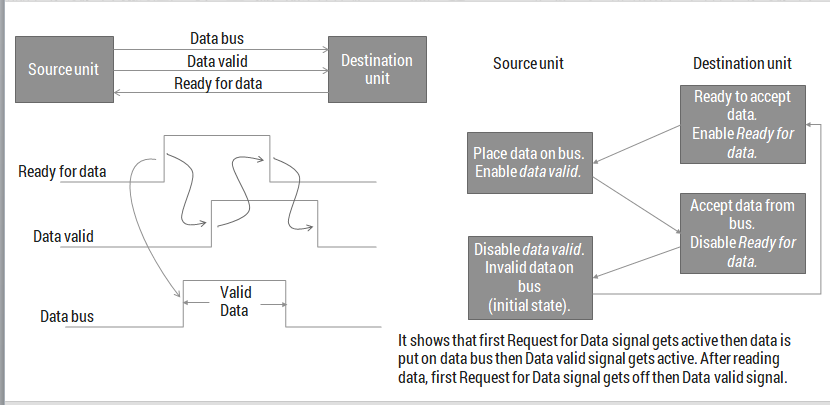
* In Source initiated Strobe, it is assumed that destination has read the data from the data bus but their is no surety.
* In Destination initiated Strobe, it is assumed that source has put the data on the data bus but their is no surety.

**Handshake**

**Source-initiated**

****

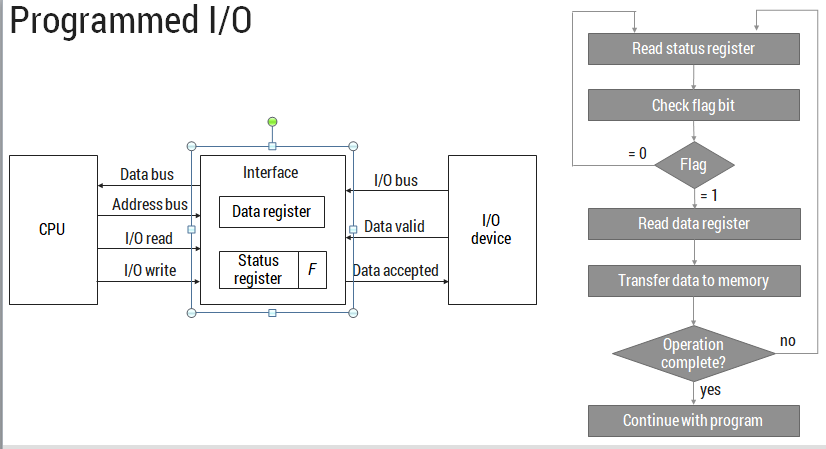
**Destination initiated handshake**

****

**Modes Of Transfer**

* Data transfer between the central computer and I/O devices may be handled in a variety of modes.
* Some modes use the CPU as an intermediate path; others transfer the data directly to and from the memory unit.
* Data transfer to and from peripherals may be handled in one of three possible modes:
  1. Programmed I/O
  2. Interrupt-initiated I/O
  3. Direct memory access (DMA)

**Programmed i/o**

****

* It is due to the result of the I/O instructions that are written in the computer program.
* Each data item transfer is initiated by an instruction in the program.
* In this case it requires constant monitoring by the CPU of the peripheral devices.
* Example
* In this case, the I/O device does not have direct access to the memory unit.
* A transfer from I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from device to the CPU and store instruction to transfer the data from CPU to memory.
* In programmed I/O, the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer.
* This is a time consuming process since it needlessly keeps the CPU busy. This situation can be avoided by using an interrupt facility**.**

**Interrupt-initiated I/O**

* An alternative to the CPU constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data.
* While the CPU is running a program, it does not check the flag.
* However, when the flag is set, the computer is momentarily interrupted from proceeding with current program and is informed of the fact that the flag has been set.
* The CPU deviates from what it is doing to take care of the input or output transfer.
* After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.

**Priority Interrupt**

* what if multiple devices generate interrupts simultaneously. In that case, we have to have a way to decide which interrupt is to be serviced first. In other words, we have to set a priority among all the devices for systemic interrupt servicing.

**Priority Interrupt by Software(Polling)**

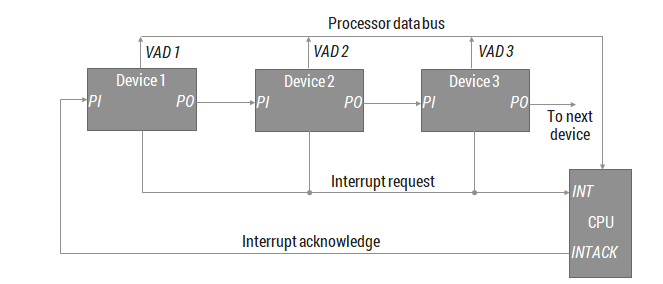
Polling is the software method of establishing priority of simultaneous interrupt.

* Establishes priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously.
* •Highest priority source is tested first and if its interrupt signal is on, control branches to a service routine for this source
* Flexible since it is established by software
* - Low cost since it needs a very little hardware
* - Very slow

**Priority Interrupt by Hardware**

**Priority Interrupt (Daisy-Chaining Technique)**

* Device with highest priority is placed first.
* Device that wants the attention send the interrupt request to the CPU.
* CPU then sends the INTACK signal which is applied to PI(priority in) of the first
* device.
* If it had requested the attention, it place its VAD(vector address) on the bus. And it
* block the signal by placing 0 in PO(priority out)
* If not it pass the signal to next device through PO(priority out) by placing 1.
* This process is continued until appropriate device is found.
* The device whose PI is 1 and PO is 0 is the device that send the interrupt request.



**Parallel Priority Interrupt** :

It consist of interrupt register whose bits are set separately by the interrupting devices.

Priority is established according to the position of the bits in the register.

Mask register is used to provide facility for the higher priority devices to interrupt

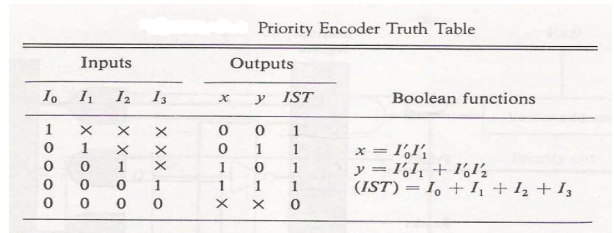
when lower priority device is being serviced or disable all lower priority devices

when higher is being serviced.

Corresponding interrupt bit and mask bit are ANDed and applied to priority encoder.

Priority encoder generates two bits of vector address.

Another output from it sets IST(interrupt status flip flop).



**Direct Memory Access (DMA):**

In the Direct Memory Access (DMA) the interface transfer the data into and out of thememory unit through the memory bus.

The transfer of data between a fast storage device suchas magnetic disk and memory is often limited by the speed of the CPU.

Removing the CPUfrom the path and letting the peripheral device manage the memory buses directly wouldimprove the speed of transfer. This transfer technique is called Direct Memory Access(DMA).

During the DMA transfer, the CPU is idle and has no control of the memory buses. A DMA

Controller takes over the buses to manage the transfer directly between the I/O device and

memory.

The CPU may be placed in an idle state in a variety of ways. One common method

extensively used in microprocessor is to disable the buses through special control signals

such as:

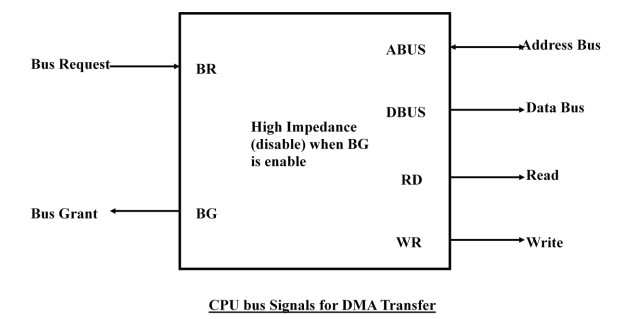
 Bus Request (BR)

 Bus Grant (BG)

These two control signals in the CPU that facilitates the DMA transfer. The Bus Request

(BR) input is used by the DMA controller to request the CPU.

When this input is active, theCPU terminates the execution of the current instruction and places the address bus, data busand read write lines into a high Impedance state. High Impedance state means that the output is disconnected.



The CPU activates the Bus Grant (BG) output to inform the external DMA that the Bus

Request (BR) can now take control of the buses to conduct memory transfer without

processor.

When the DMA terminates the transfer, it disables the Bus Request (BR) line. The CPU

disables the Bus Grant (BG), takes control of the buses and return to its normal operation.

The transfer can be made in several ways that are:

i. DMA Burst

ii. Cycle Stealing

i) DMA Burst :- In DMA Burst transfer, a block sequence consisting of a number of

memory words is transferred in continuous burst while the DMA controller is master

of the memory buses.

ii) Cycle Stealing :- Cycle stealing allows the DMA controller to transfer one data word

at a time, after which it must returns control of the buses to the CPU.

**DMA Controller:**

The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device.

The DMA controller has three registers:

i. Address Register

ii. Word Count Register

iii. Control Register

i. Address Register :- Address Register contains an address to specify the desiredlocation in memory.

ii. Word Count Register :- WC holds the number of words to be transferred. Theregister is incre/decre by one after each word transfer and internally tested for zero.

i. Control Register :- Control Register specifies the mode of transferThe unit communicates with the CPU via the data bus and control lines.

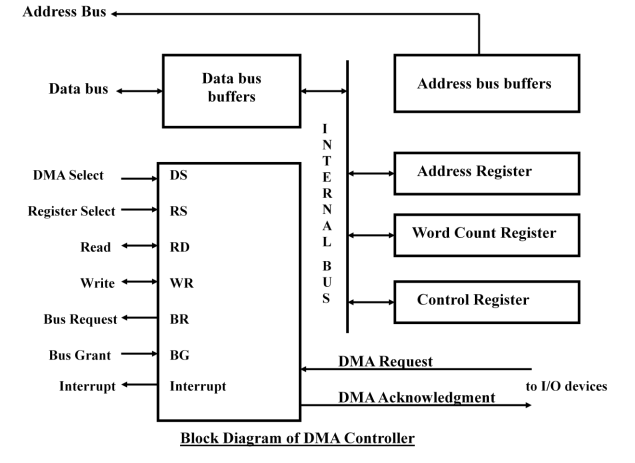
Theregisters in the DMA are selected by the CPU through the address bus by enabling the

DS (DMA select) and RS (Register select) inputs. The RD (read) and WR (write)

inputs are bidirectional.

When the BG (Bus Grant) input is 0, the CPU can communicatewith the DMA registers through the data bus to read from or write to the DMAregisters.

When BG =1, the DMA can communicate directly with the memory byspecifying an address in the address bus and activating the RD or WR control.



**DMA Transfer**:

The CPU communicates with the DMA through the address and data buses as with

any interface unit. The DMA has its own address, which activates the DS and RS

lines. The CPU initializes the DMA through the data bus. Once the DMA receives the

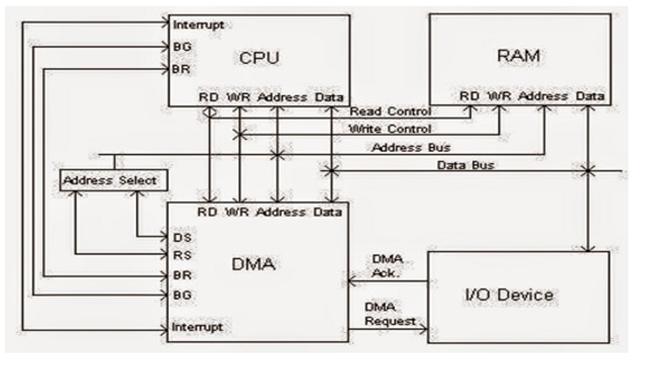
start control command, it can transfer between the peripheral and the memory.

When BG = 0 the RD and WR are input lines allowing the CPU to

communicate with the internal DMA registers. When BG=1, the RD and WR are

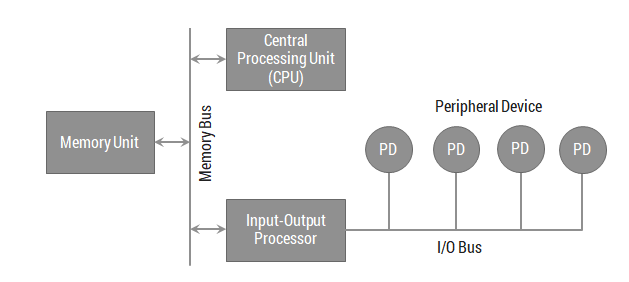
output lines from the DMA controller to the random access memory to specify the

read or write operation of data.



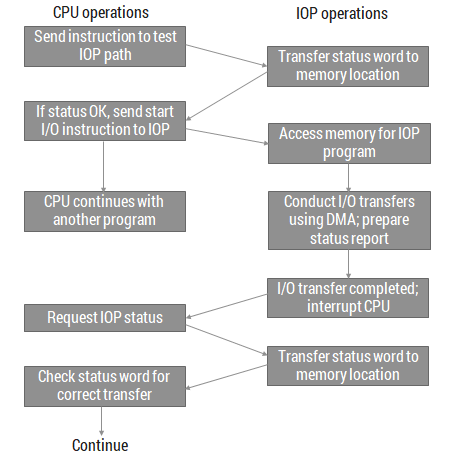
**Input-Output Processor (IOP)**

* An input-output processor (IOP) is a processor with direct memory access capability
* The IOP is similar to CPU except that it is designed to handle only I/O processing
* The IOP fetches and executes I/O instructions to facilitate I/O transfer
* Both CPU and IOP exists in the system however CPU is master,while IOP is slave
* The CPU only initiates the I/O program after that IOP operates independent of CPU
* The I/O processor is capable of performing actions without interruption or intervention from the CPU. The CPU only needs to initiate the I/O processor by telling it what activity to perform. Once the necessary actions are performed, the I/O processor then provides the results to the CPU.



* Memory occupies the central position and can communicate with each processor by DMA. 
* CPU is responsible for processing data. 
* IOP provides the path for transfer of data between various peripheral devices and memory. 
* Data formats of peripherals differ from CPU and memory. IOP maintain such problems. 
* Data are transfer from IOP to memory by stealing one memory cycle.
* Instructions that are read from memory by IOP are called commands to distinguish them from instructions that are read by the CPU

**CPU – IOP Communication**



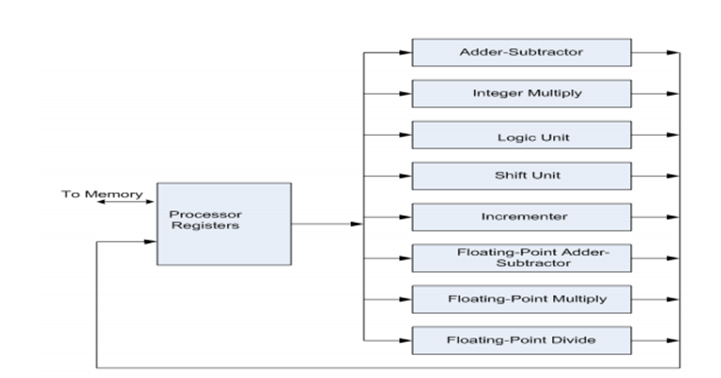
**PIPELINING**

**CONTENTS:**

* + Parallel Processing
  + Pipelining
  + Arithmetic Pipeline
  + Instruction Pipeline

**Parallel Processing**

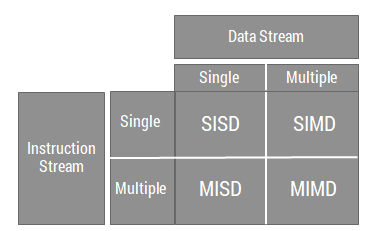
* Parallel processing is a term used to denote a large class of techniques that are used to provide simultaneous data-processing tasks for the purpose of increasing the computational speed of a computer system.
* The system may have two or more ALUs to be able to execute two or more instruction at the same time. 
* The system may have two or more processors operating concurrently. 
* It can be achieved by having multiple functional units that perform same or different operation simultaneously.
* Purpose of parallel processing is to speed up the computer processing capability and increase its throughput.
* Throughput: The amount of processing that can be accomplished during a given interval of time.

****

**CLASSIFICATION**

* There are variety of ways in which the parallel processing can be classified
* Internal Organization of Processor ¡
* Interconnection structure between processors
* Flow of information through system
* . One classification introduced by M. J. Flynn considers the organization of a computer system by the number of instructions and data items that are manipulated simultaneously.
* The sequence of instructions read from memory constitutes an instruction stream .
* The operations performed on the data in the processor constitutes a data stream **.**

**Flynn's taxonomy**

****

**Single Instruction Single Data (SISD)**

* SISD represents the organization of a single computer containing a control unit, a processor unit, and a memory unit.
* Instructions are executed sequentially and the system may or may not have internal parallel processing capabilities.

**Single Instruction Multiple Data (SIMD)**

* SIMD represents an organization that includes many processing units under the supervision of a common control unit.
* All processors receive the same instruction from the control unit but operate on different items of data.

**Multiple Instruction Single Data (MISD)**

* There is no computer at present that can be classified as MISD.
* MISD structure is only of theoretical interest since no practical system has been constructed using this organization.

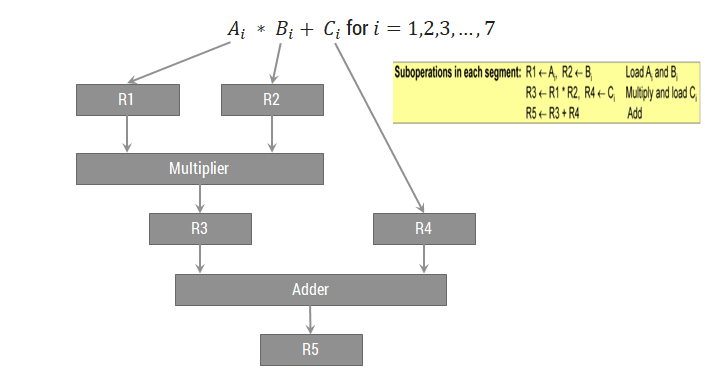
**Multiple Instruction Multiple Data (MIMD)**

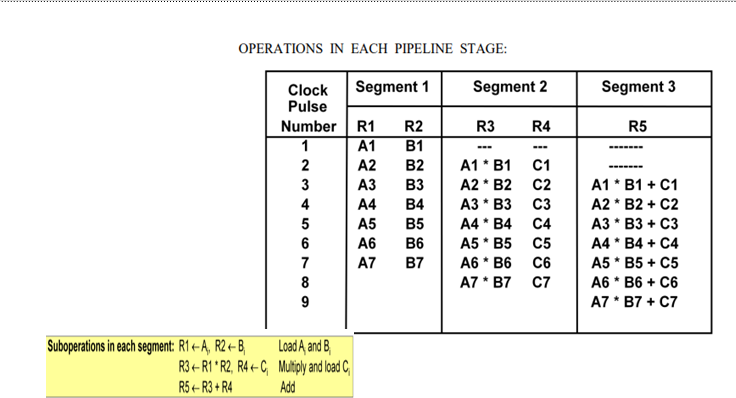
* MIMD organization refers to a computer system capable of processing several programs at the same time.
* Most multiprocessor and multicomputer systems can be classified in this category.
* Contains multiple processing units.
* Execution of multiple instructions on multiple data.

**Pipelining**

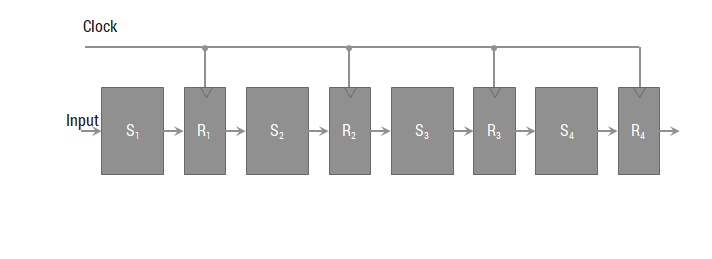
* Pipeline is a technique of decomposing a sequential process into sub operations, with each sub process being executed in a special dedicated segment that operates concurrently with all other segments.
* The result obtained from the computation in each segment is transferred to the next segment in the pipeline.
* The registers provide isolation between each segment.
* The technique is efficient for those applications that need to repeat the same task many times with different sets of data.

**Pipelining example**

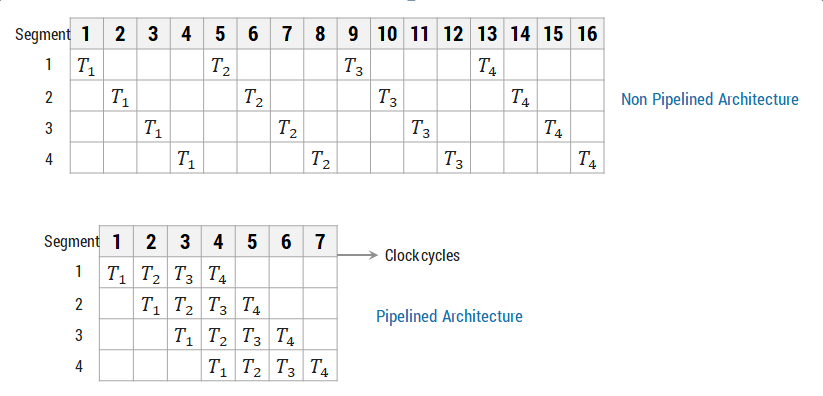
****

****

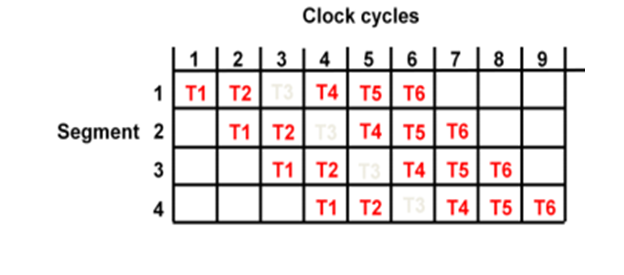
**General structure of four segment pipeline**

****

**Space-time Diagram**

****

**Pipeline Speedup**

****

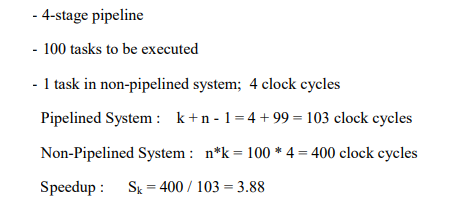
* Consider the case where a k-segment pipeline used to execute n tasks.
* N=6 k=4

Pipelined Machine (k stages, n tasks)

* The first task t1 requires k clock cycles to complete its operation since there are k segments
* The remaining n-1 tasks require n-1 clock cycles
* The n tasks clock cycles = k+(n-1) (9 in previous example)

Conventional Machine (Non-Pipelined)

* Cycles to complete each task in nonpipeline = k
* For n tasks, n cycles required =n\*k
* Speedup ratio = nk/(k+n-1)

****

**Arithmetic Pipeline**

* An arithmetic pipeline divides an arithmetic problem into various sub problems for execution in various pipeline segments.
* Usually found in high speed computers.
* Used to implement floating point operations, multiplication of fixed point numbers and similar operations.
* Example:
* Consider an example of floating point addition and subtraction.

𝑋=𝐴 ×〖10〗^𝑎

𝑌=𝐵 ×〖10〗^𝑏

* A and B are two fractions that represent the mantissas and a and b are the exponents.

**Example of Arithmetic Pipeline**

* Consider the two normalized floating-point numbers:

X = 0.9504 x 103 Y = 0.8200 x 102

* Segment-1: The larger exponent is chosen as the exponent of result.
* Segment-2: Aligning the mantissa numbers

X = 0.9504 x 103 Y = 0.0820 x 103

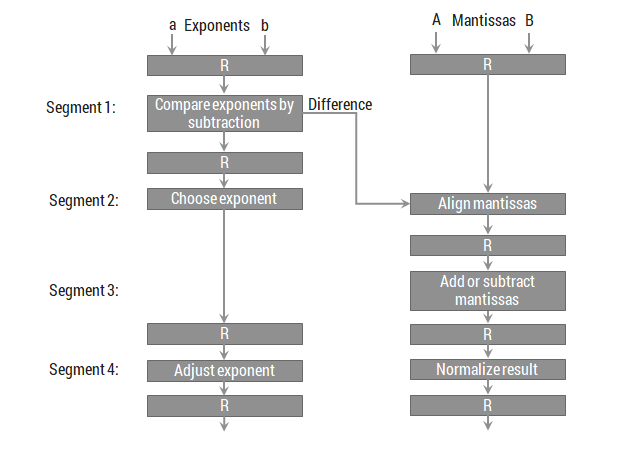
* Segment-3: Addition of the two mantissas produces the sum

Z = 1.0324 x 103

* Segment-4: Normalize the result

Z = 0.10324 x 104

* The sub-operations that are performed in the four segments are:
  1. Compare the exponents
  2. Align the mantissas
  3. Add or subtract the mantissas
  4. Normalize the result



**Instruction Pipeline**

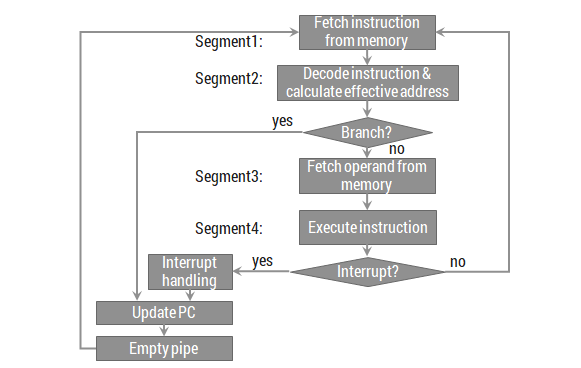
* An instruction pipeline reads instruction from the memory while previous instructions are being executed in other segments of the pipeline. Thus we can execute multiple instructions simultaneously.
* In the most general case, the computer needs to process each instruction with the following sequence of steps

1. Fetch the instruction from memory.
2. Decode the instruction.
3. Calculate the effective address.
4. Fetch the operands from memory.
5. Execute the instruction.
6. Store the result in the proper place.

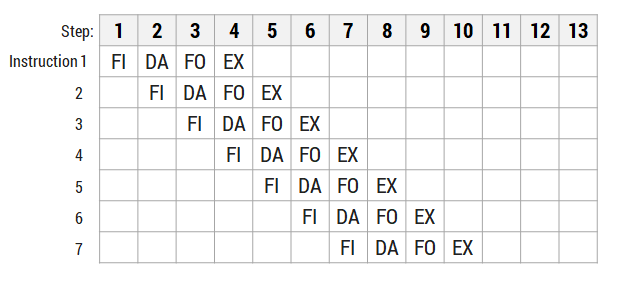
* Different segments may take different times to operate on the incoming information.
* The design of an instruction pipeline will be most efficient if the instruction cycle is divided into segments of equal duration.
* Assume that the decoding of the instruction can be combined with the calculation of the effective address into one segment.
* Assume further that most of the instructions place the result into a processor registers so that the instruction execution and storing of the result can be combined into one segment.
* This reduces the instruction pipeline into four segments.

1. FI: Fetch an instruction from memory
2. DA: Decode the instruction and calculate the effective address of the operand
3. FO: Fetch the operand
4. EX: Execute the operation

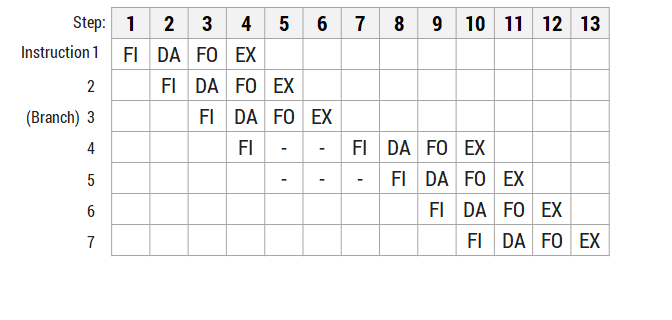
**Four segment CPU pipeline**



**Space-time Diagram**



**Space-time Diagram(BRANCHING)**



**Pipeline Conflict**

* There are three major difficulties that cause the instruction pipeline conflicts.
  1. Resource conflicts caused by access to memory by two segments at the same time. Most of these conflicts can be resolved by using separate instruction and data memories.
  2. Data dependency conflicts arise when an instruction depends on the result of a previous instruction, but this result is not yet available.
  3. Branch difficulties arise from branch and other instructions that change the value of PC.