1(a) What is the *memory data register*?

- It is a register where the word to be stored into the memory location is first loaded by the CPU.
- The MDR is used exclusively by the CPU and is not directly accessible to programmers.

(b) Describe the main constituents of a typical *control unit*.

A typical CPU has three major components: (1) register set, (2) arithmetic logic unit (ALU), and (3) control unit (CU).

The register set differs from one computer architecture to another.

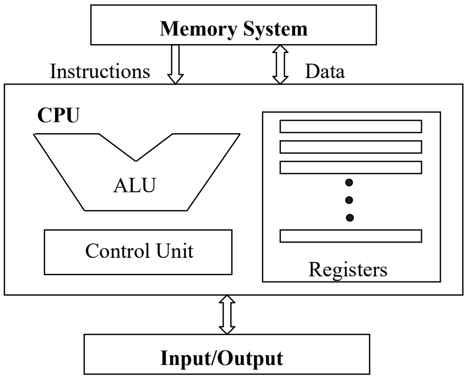
- It is usually a combination of general-purpose and special purpose registers.
- General-purpose registers are used for any purpose, hence the name general purpose.
- Special-purpose registers have specific functions within the CPU.
- For example, the program counter (PC) and the instruction register (IR).

The ALU provides the circuitry needed to perform the arithmetic, logical and shift operations demanded of the instruction set.

- Arithmetic operations are addition, subtraction, multiplication and division
- Logical operations are AND, OR, NOT
- Shift and rotate operations such as LEFT and RIGHT rotate operations

The control unit is the entity responsible for fetching the instruction to be executed from the main memory and decoding and then executing it.

- The CPU fetches instructions from memory,
- reads and writes data from and to memory, and
- transfers data from and to input/output devices.



Control unit and its constituents

(c) Compare and contrast *microprogrammed* and *hardwired control units*.

(i) **Similarity**

- Both of them are center of coordination of all the activities that occur in their computer systems.

- **Differences**

S/No	Microprogrammed Control unit	Hardwired Control unit
1	The control signals associated with	Fixed logic circuits that correspond

	operations are stored in special memory units inaccessible by the programmer as control words.	directly to the Boolean expressions are used to generate the control signals
2	A sequence of microinstructions is called a microprogram, which is stored in a ROM or RAM called a control memory CM	It does not have any control memory
3	It is slower	It is faster than microprogrammed control.
4	It is cheaper than Hardwired Control unit	Hardwired control could be very expensive and complicated for complex systems
5	microprogrammed control could adapt easily to changes in the system design.	Hardwired control will require a redesign of the entire systems in the case of any change
6	MCP control unit is more expensive for small control unit	Hardwired control is more economical for small control units.

(d) Suppose that the instruction set of a machine has three instructions: *Inst-1, Inst-2* and *Inst-3*; and A, B, C, D, E, F, G and H are the control lines. The following table shows the control lines that should be activated for the three instructions at the three steps T0, T1 and T2.

Step	**Inst-1**	**Inst-2**	**Inst-3**
T0	**A, B, G**	**F, H, G**	**F, C**
T1	**G, A**	**A**	**B, C**
T2	**B, C**	**H, D**	

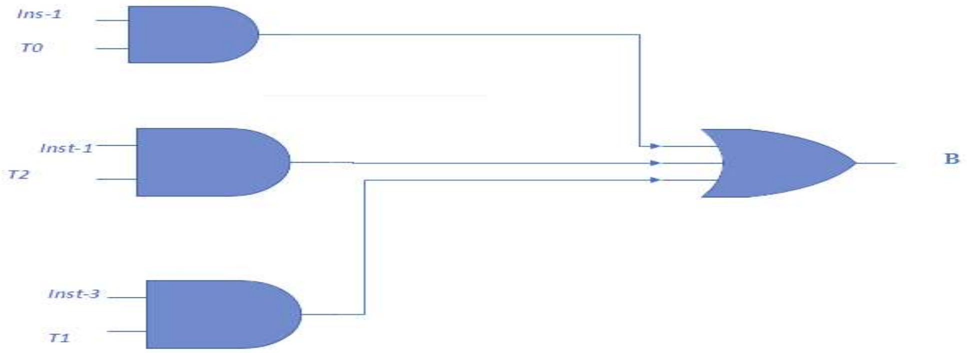
Using Hardwired approach:

(e) Write Boolean expressions for all the control lines B, C, D, G.

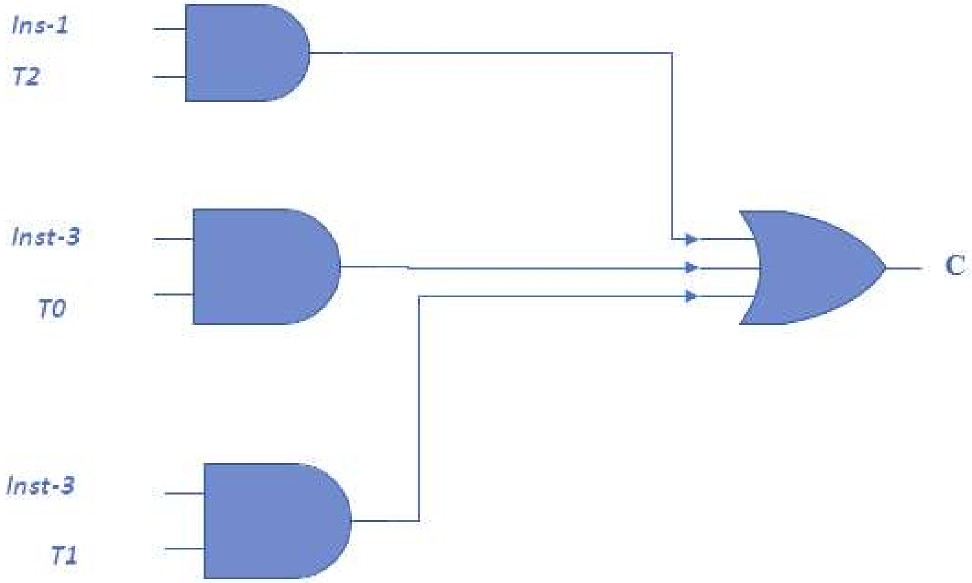
The Boolean expressions for control lines B, C, D, G are as follows:
B= *Inst-1.*T0 + *Inst-1.*T2 + *Inst-3.T1*
C= *Inst-1.T2* + *Inst-3.T0* + *Inst-3.T1*
D= *Inst-2.T2*
G= *Inst-1.T0* + *Inst-1.T1* + *Inst-2.T0*

(f) Draw the logic circuit for each control line.

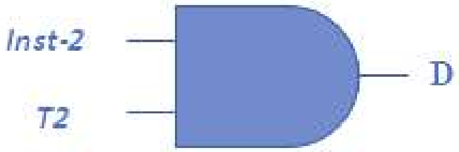
B= *Inst-1*.T0 + *Inst-1*.T2 + *Inst-3.T1*



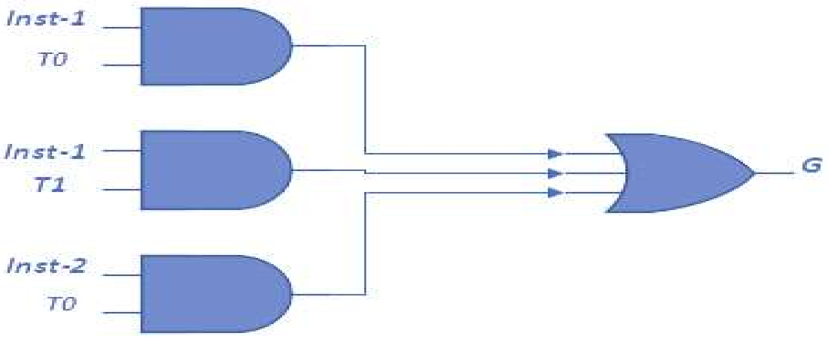
C= *Inst-1.T2* + *Inst-3.T0* + *Inst-3.T1*



D= *Inst-2.T2*



G= ***Inst-1.T0 + Inst-1.T1 + Inst-2.T0***



(g) What is semantic gap as elucidated in *RISC/CISC* evolution cycle?

- In particular, *it calls for going back to basics rather than providing extra hardware support for high-level languages*.
- This paradigm shift relates to what is known as the semantic gap,
  - a measure of the difference between the operations provided in the high-level languages (HLLs) and those provided in computer architectures.
- It is recognized that the wider the semantic gap, the larger the number of undesirable consequences.
- These include (a) execution inefficiency,
  (b) excessive machine program size, and;
  (c) increased compiler complexity.

(h) Highlight the common characteristics of *RISC*-based machines.

Today, RISC-based machines are reality and they are characterized by a number of common features such as:

- simple and reduced instruction set,
- *fixed instruction format,*
- one instruction per machine cycle,
- *pipeline instruction fetch/execute units*,
- ample number of general-purpose registers (or alternatively optimized compiler code generation),
- *Load/Store memory operations, and*
- hardwired control unit design.

(i) Explain the two design alternatives experienced in early design architecture of computer systems.

- A computer with the minimum number of instructions has the disadvantage that a large number of instructions will have to be executed in realizing even a simple function.
- This will result in a speed disadvantage.
- *On the other hand*, a computer with an inflated number of instructions has the disadvantage of complex decoding and hence a *speed disadvantage*.
- It is then natural to believe that a computer with a carefully selected reduced set of instructions should strike a balance between the above two design alternatives.

(j) What constitutes a carefully selected reduced set of instructions in *RISC* based machines?

These aspects should include:

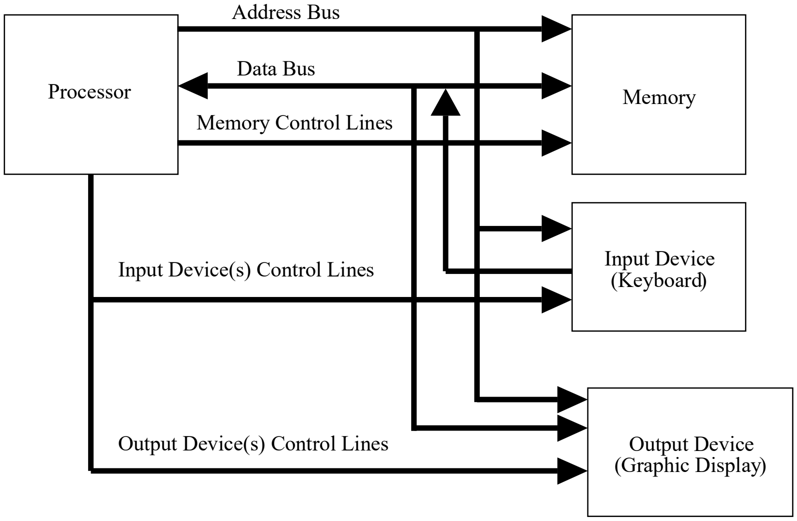
a) operations that are most frequently performed during execution of typical (benchmark) programs,
b) operations that are most time consuming, and;
c) the type of operands that are most frequently used.

(k) What is the distinguishing factor among input devices.

- The distinguishing factor among input devices (and also among output devices) is their data processing rate,
- It is defined as the average number of characters that can be processed by a device per second.

(l) Describe, in detail, the concept of shared I/O arrangement.

- In the first arrangement, I/O devices are assigned particular addresses, isolated from the address space assigned to the memory.
- The execution of an input instruction at an input device address will cause the character stored in the input register of that device to be transferred to a specific register in the CPU.
- Similarly, the execution of an output instruction at an output device address will cause the character stored in a specific register in the CPU to be transferred to the output register of that output device.
- This arrangement, called shared I/O



Shared I/O arrangement

(m) Highlight the distinguishing characteristics between Daisy Chain Bus Arbitration and Independent Source Bus Arbitration

S/No	Daisy Chain Bus Arbitration (DCBA)	Independent Source Bus Arbitration (ISBA)
1	In DCBA, I/O devices present their interrupt requests to the interrupt request line INR (similar to the polling arrangement). Upon recognizing the arrival of an interrupt request, the processor, through a daisy chained grant line (GL), sends its grant to the requesting device to start communication with the processor.	In ISBA, each I/O device has its own interrupt request line, through which it can send its interrupt request, independent of the other devices.
2	In the case of multiple requests, the DCBA arrangement gives highest priority to the device physically nearer to the processor.	I/O device priority in the ISBA does not depend on the device location. A priority arbitration circuitry is needed in order to deal

		with simultaneous interrupt requests.

(n) Discuss the concept of context switching.

- To process an interrupt, the context of the current process must be saved and the interrupt handling routine must be invoked. This process is called context switching
- A process context has two parts: processor context and memory context.
- The processor context is the state of the CPU’s registers including program counter (PC), program status words (PSWs), and other registers.
- The memory context is the state of the program’s memory including the program and data.

(o) With the aid of a suitable diagram, explain the operational process of a 2-bus organization.

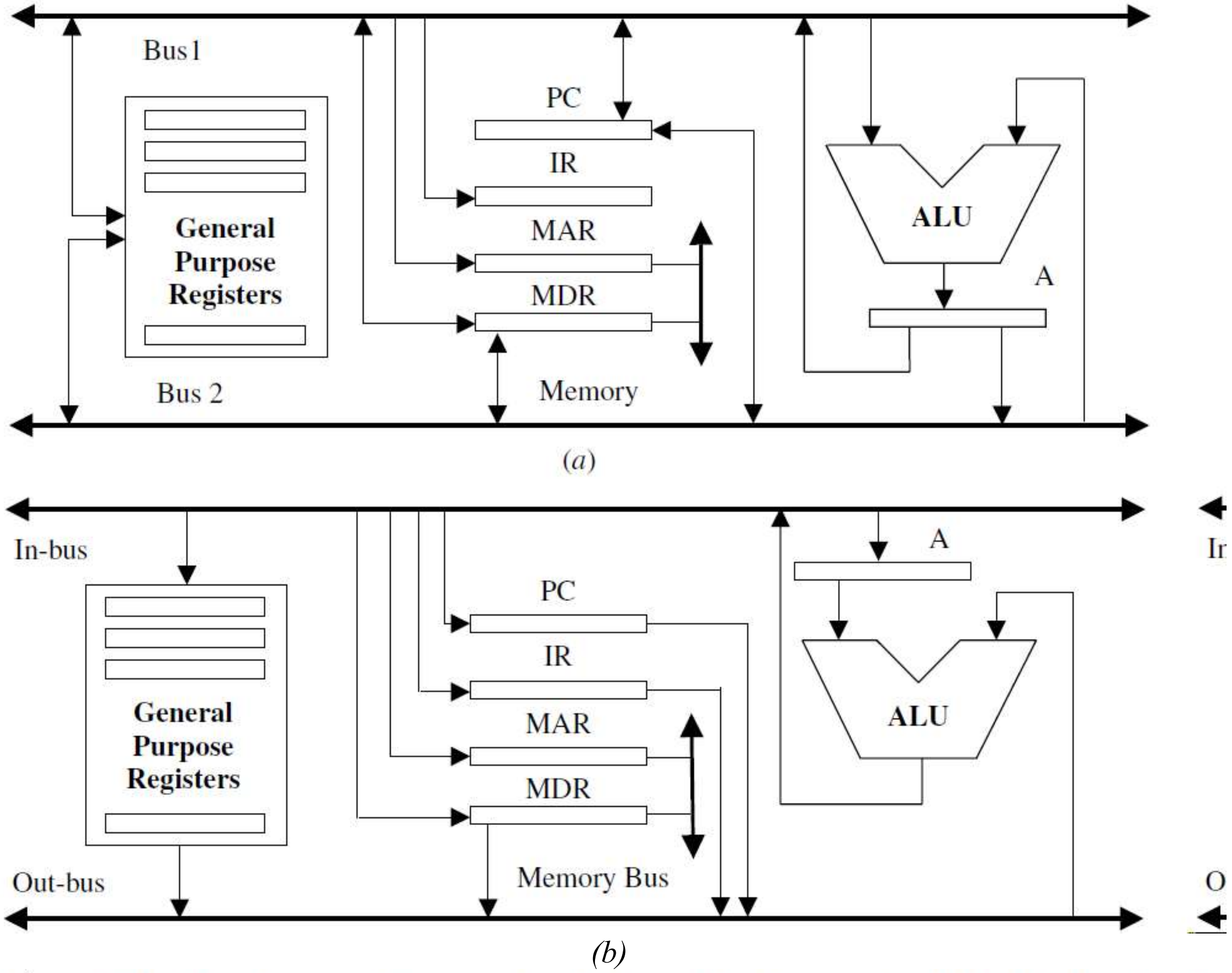


Figure 5.4 Two-bus organizations. (*a*) An Example of Two-Bus Datapath. (*b*) Another Example of Two-Bus Datapath with in-bus and out-bus

The operational process of a 2-bus organization are as follows:

- Using two buses is a faster solution than the one-bus organization.
- In this case, general-purpose registers are connected to both buses.
- Data can be transferred from two different registers to the input point of the ALU at the same time. Therefore, a two-operand operation can fetch both operands in the same clock cycle.
- An additional buffer register may be needed to hold the output of the ALU when the two buses are busy carrying the two operands.
- *Figure a* shows a two-bus organization.
- In some cases, one of the buses may be dedicated for moving data into registers (in-bus), while the other is dedicated for transferring data out of the registers (out-bus).
- In this case, the additional buffer register may be used, as one of the ALU inputs, to hold one of the operands.

- The ALU output can be connected directly to the in-bus, which will transfer the result into one of the registers.
- Figure b shows a two-bus organization with in-bus and out-bus.

(p) With the aid of an illustration, state the procedure involved in realizing a typical and simple execution cycle of an instruction in the CPU.

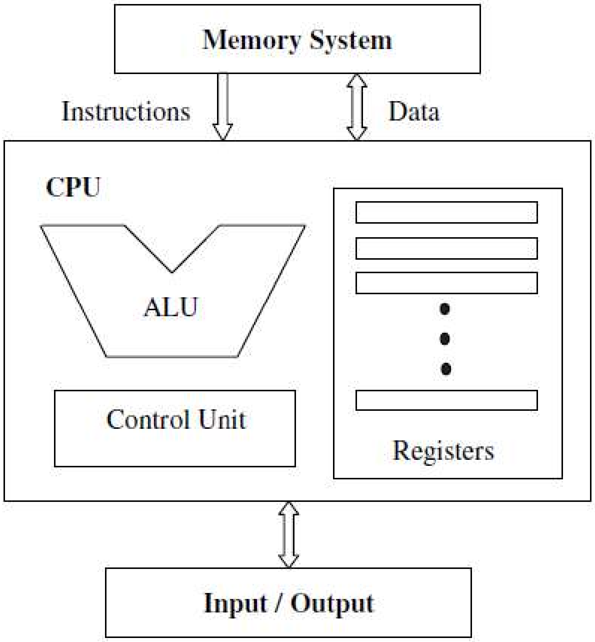


Figure 5.1 Central processing unit main components and interactions with the memory and I/O

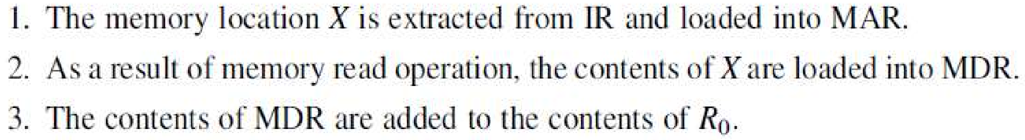
The procedure involved in realizing a typical and simple execution cycle of an instruction in the CPU is as follows:

1. The next instruction to be executed, whose address is obtained from the PC, is fetched from the memory and stored in the IR.
2. The instruction is decoded.
3. Operands are fetched from the memory and stored in CPU registers, if needed.
4. The instruction is executed.
5. Results are transferred from CPU registers to the memory, if needed.

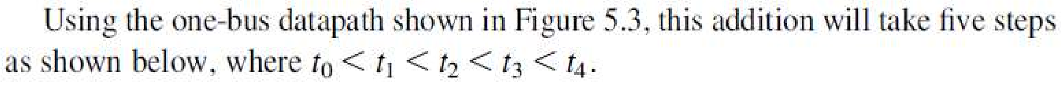
(q) In a simple CPU, an instruction adds the contents of memory location X to register R0 and stores the result in R0.

(i) Highlight the sequence of steps involved in executing the arithmetic operation;

The sequence of steps involved in executing the arithmetic operation are as follows:

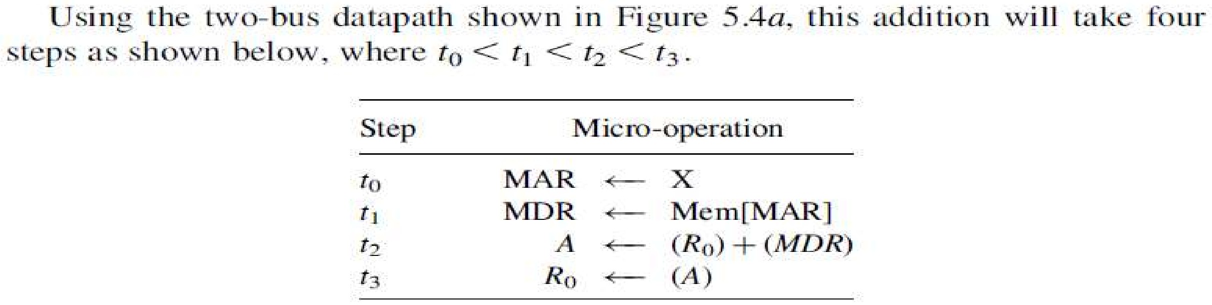


1. The memory location X is extracted from IR and loaded into MAR.
2. As a result of memory read operation, the contents of X are loaded into MDR.
3. The contents of MDR are added to the contents of R_0.



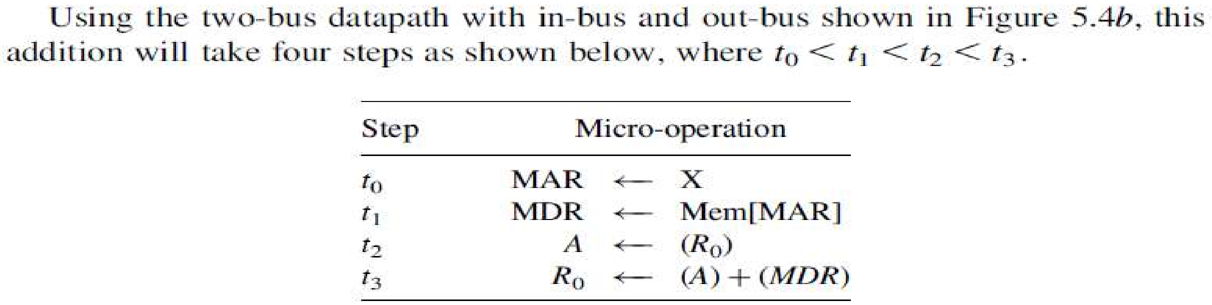
Using the one-bus datapath shown in Figure 5.3, this addition will take five steps as shown below, where $t_0 < t_1 < t_2 < t_3 < t_4$.

(r) How can the arithmetic operation be realized in *two-bus* organization by using micro-operation?



Using the two-bus datapath shown in Figure 5.4*a*, this addition will take four steps as shown below, where $t_0 < t_1 < t_2 < t_3$.

Step	Micro-operation
t_0	MAR $\leftarrow$ X
t_1	MDR $\leftarrow$ Mem[MAR]
t_2	$A \leftarrow (R_0) + (MDR)$
t_3	$R_0 \leftarrow (A)$

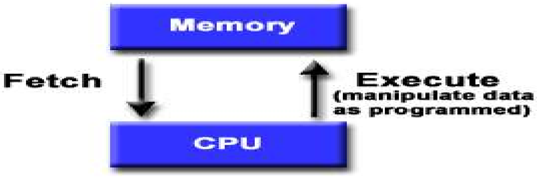


Using the two-bus datapath with in-bus and out-bus shown in Figure 5.4*b*, this addition will take four steps as shown below, where $t_0 < t_1 < t_2 < t_3$.

Step	Micro-operation
t_0	MAR $\leftarrow$ X
t_1	MDR $\leftarrow$ Mem[MAR]
t_2	$A \leftarrow (R_0)$
t_3	$R_0 \leftarrow (A) + (MDR)$

(s) What is Von Neumann Architecture?

- *A Von Neumann computer uses the stored-program concept.*
- The *CPU executes a stored program that specifies a sequence of both read and write operations on the memory*.



Von Neumann Computer Architecture

- Its basic design includes the followings: *memory is used to store both program and data instructions.*

(t) Discuss the characteristics of shared memory architecture?

The following depict the general characteristics of shared memory parallel computers architecture:

- it varies widely, but generally have in common the ability for all processors to access all memory as global address space;
- multiple processors can operate independently but share the same memory resources;
- changes in a memory location effected by one processor are visible to all other processors.

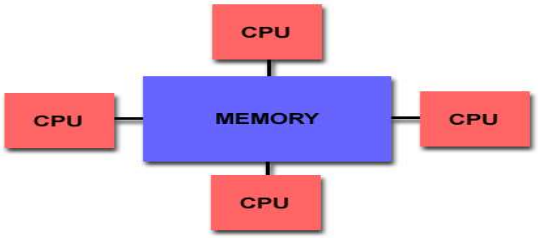


Chart of Shared Memory

- Shared memory machines can be divided into two main classes based upon memory access times: **Uniform Memory Access (UMA)** and **Non-Uniform Memory Access (NUMA)**.

- **Uniform Memory Access (UMA):** Most commonly represented today by Symmetric Multiprocessor (SMP) machines. Identical processors and equal access and access times to memory.
- **Non-Uniform Memory Access (NUMA):** Often made by physically linking two or more SMPs. One SMP can directly access memory of another SMP. Not all processors have equal access time to all memories

(u) Differentiate between *Single Instruction Stream, Single Data Stream* and *Multiple Instruction Stream, Multiple Data Stream*.

(i) Single Instruction, Multiple Data (SIMD):

- This pertains to a *type of parallel computer in which* ***all processing units execute the same instruction at any given clock cycle*** *and* ***each processing unit can operate on different data elements****.*
- This type of machine typically has an instruction dispatcher, a very high-bandwidth internal network, and a very large array of very small-capacity instruction units.
- It displays a *synchronous (lockstep) and deterministic execution*.
- Two varieties: Processor Arrays and Vector Pipelines

(ii) Multiple Instructions, Multiple Data (MIMD):

- Currently, this is the most common type of parallel computer.
- Most modern computers fall into this category in which every processor may be executing a different instruction stream and every processor may be working with different data streams.
- Execution can be *synchronous or asynchronous, deterministic or non-deterministic*
- Examples are most current supercomputers, networked parallel computer "grids" and multi-processor SMP computers - including some types of PCs.