Differentiate between computer organization and computer architecture.

Computer Architecture	Computer Organization
Computer Architecture is concerned with the way hardware components are connected together to form a computer system.	Computer Organization is concerned with the structure and behaviour of a computer system as seen by the user.
It acts as the interface between hardware and software.	It deals with the components of a connection in a system.
Computer Architecture helps us to understand the functionalities of a system.	Computer Organization tells us how exactly all the units in the system are arranged and interconnected.
A programmer can view architecture in terms of instructions, addressing modes and registers.	Whereas Organization expresses the realization of architecture.
While designing a computer system architecture is considered first.	An organization is done on the basis of architecture.
Computer Architecture deals with high-level design issues.	Computer Organization deals with low-level design issues.
Architecture involves Logic (Instruction sets, Addressing modes, Data types, Cache optimization)	Organization involves Physical Components (Circuit design, Adders, Signals, Peripherals)

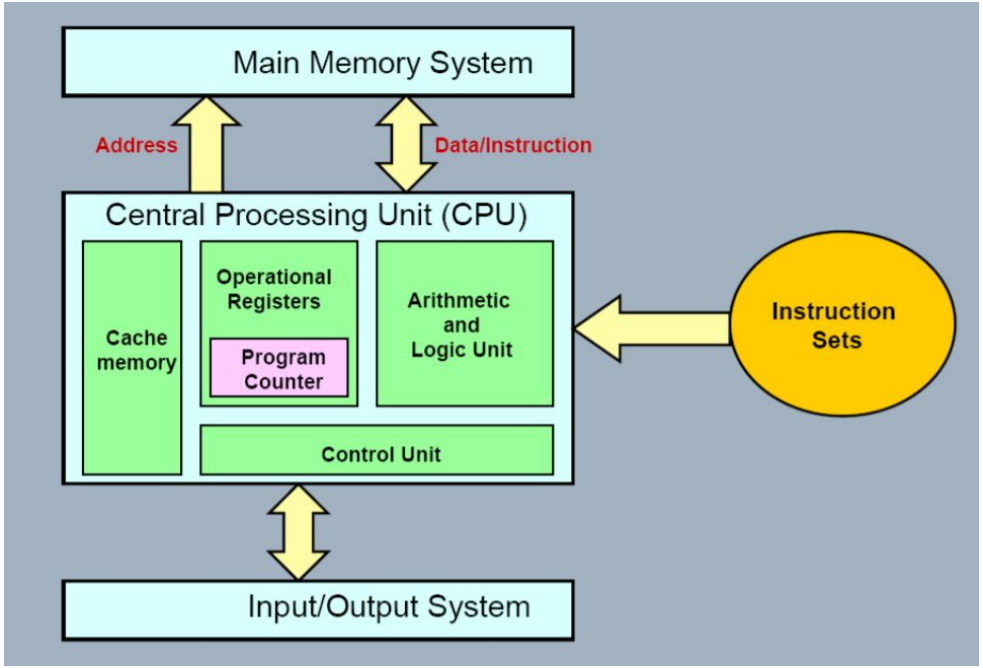
With a neat diagram, explain the internal architecture of the CPU.

OR

With a neat block diagram explain the functional units in the basic structure of computer

OR

Explain the basic organization of a digital computer with description of each unit.



A computer consist of mainly five independent functional parts

- **Input Unit**

  This unit accepts information from human operators with the help of electomechanical devices such as keyboard. Whenever a key is pressed the corresponding letter or digit is automatically translated into its corresponding binary code and transmitted over a cable to either the memory or processor. The received information can be stored in computers memory for later references otherwise it can be immediately used by the ALU circuitry to perform the desired operations.

- **Memory Unit**

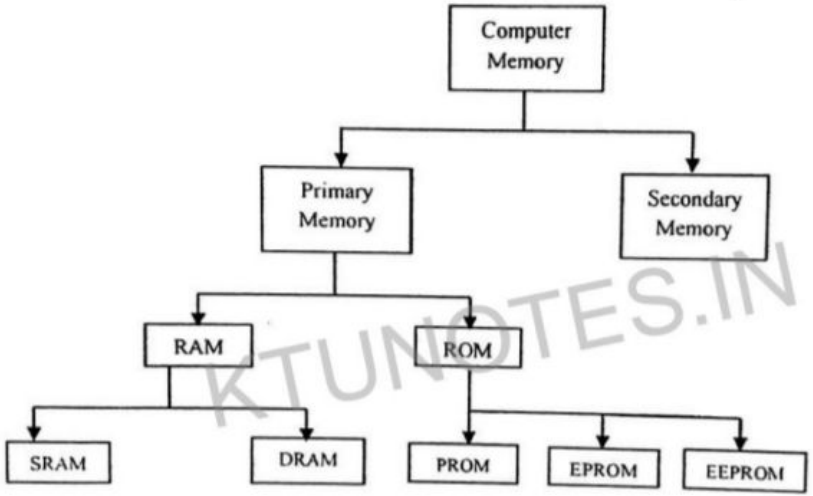


Fig 1.1: Classification of Memories

It is the storage unit of the computer system. The input information is processed based on the stored instructions called programs. these programs are stored in memory unit. The memory contains a large number of semiconductor storage cells where each cell can store one bit of information. The cells are processed in groups of fixed sizes called words, with a distinct address for each word. Addresses are simply numbers that can identify successive locations. The number of bits in each word determines word length of the computer (16 to 64 bits). Memory units are mainly classified into two types:

- ✓ **Primary Memory**

  It is a fast memory operating at electronic speeds. Programs are stored in this memory during their execution period. Primary memory is divided into two types:

  - o **RAM(Random Access Memory)**

    It is a volatile memory that means the contents will lost when the power is switched off. RAM can again be divided into two types.

    - ▪ **Static Memory(SRAM)**

      Memories that consist of circuits that are capable of retaining their states as long as the power is applied are known as static memories. These memories are designed by using transistors and inverters.

    - ▪ **Dynamic Memory(DRAM)**

      These are less expensive RAMs. The cost of static RAMs are high because of the usage of the several transistors. Dynamic RAMs are implemented with the help of a capacitor and a single transistor. Such cells don't retain their state indefinitely hence they are called dynamic RAMs.

- **ROM(Read Only Memory)**

  This is one of the major types of memory using personnel computers. ROM is a type of memory that normally can only be read as opposed to RAM which can be both read and written. It is a non- volatile memory that is contents will retain even when the power is switched off. ROM can again be divided into three types:

PROM (Programmable) is read-only memory that can be modified only once by a user. The user buys a blank PROM and enters the desired contents using a PROM program. Inside the PROM chip, there are small fuses which are burnt open during programming. It can be programmed only once and is not erasable.

EPROM (Erasable Programmable) can be erased by exposing it to ultraviolet light for a duration of up to 40 minutes.

EEPROM (Electrically Erasable Programmable) is programmed and erased electrically. It can be erased and reprogrammed about ten thousand times.

- ✓ **Secondary Memory**

  This memory is used when large amount of data and programs have to be stored. Compared to the primary memory these are cheaper but performs at low speed. Examples are CD ROMS, Magnetic disk, USB drives etc.

- **Arithmetic and Logic unit**

  This is the main part of the processing unit of a computer. The desired operations are performed by this unit. Arithmetic and logic operations can be separated with the help of a mode selector. When the mode selector bit is zero it performs arithmetic operations and performs logical operations when the bit is one. To perform the operation the required operands have to be bringing into the processor, where they are stored in registers.
- **Control unit**

  This is one of another core part of a processing unit of a computer. It is known as the nerve centre of a computer system. It coordinates the operation of all other units in computer system. It sends the control signals to other units and senses their states. Example of control signals are read, write etc.
- **Output unit**

  The processed results are sending to the outside world through output device. Example: Printer.

With the help of a block schematic explain the basic operational concepts of a digital computer.

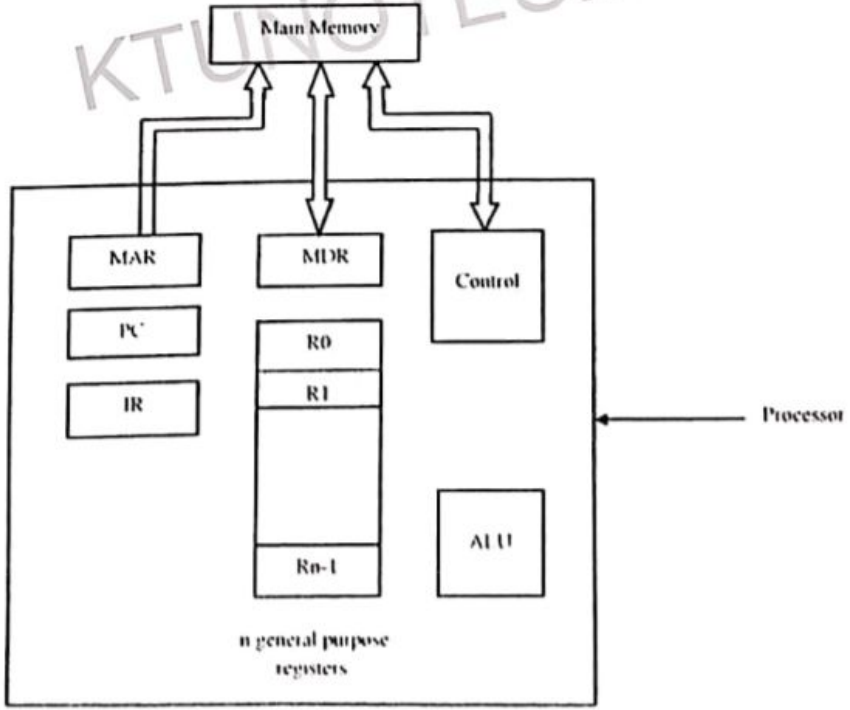


Fig 1.3: Connection between the processor and main memory

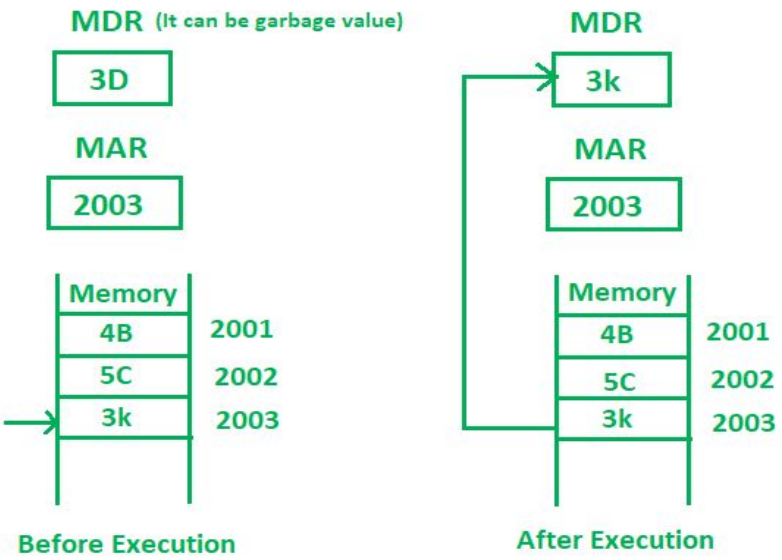
Memory Address Register (MAR) and Memory Data Register (MDR) are the two registers which are facilitating the communication between the processor and memory. In order to read an information from memory the address of the memory location in which the information is residing have to be put in MAR and a Read control signal will be sent to memory unit. When the memory unit sees the address in address line of the bus and read signal in control line of the bus memory will start the read operation from the concerned address and the result will be sent to the MDR. From there the information can be transferred to any other registers inside the processor through internal processor bus. Similarly for Write operation initially the data to be written into the memory has to be placed in MDR and the address in which the desired information have to be kept in memory will be placed in MAR and a Write control signal will be sent to the memory unit. When the memory unit sees the address, data and write signal in the external memory bus it will start the corresponding write operation.

Other than MAR and MDR few registers like PC(Program Counter), IR(Instruction Register) and some general purpose registers R_0, R_1,....R_{n-1} are there within the processor unit. PC is a register which holds the address of the next instruction to be fetched. Initially PC will be assigned by the starting program address and after fetching of each instruction it will be incremented by a size of word byte. IR is a register which holds the decoded instruction to be executed.

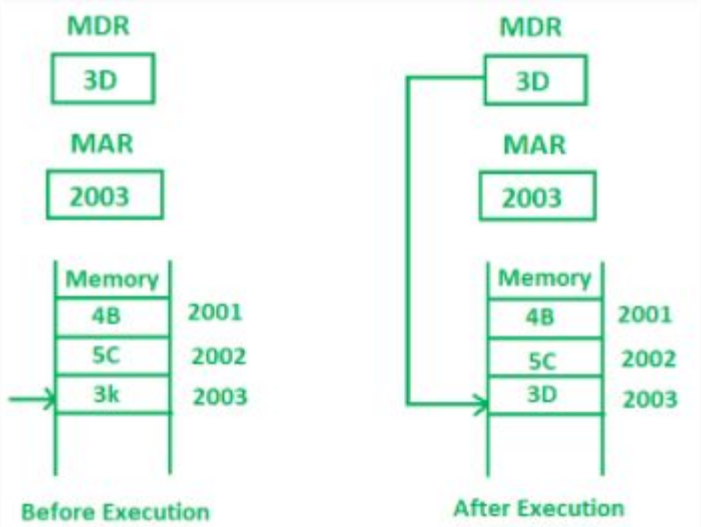
Normal execution of programs may be pre-empted if some device requires urgent servicing. In order to deal with the situation immediately the normal execution of the current program must be interrupted. To do this the device raises an interrupt signal. An interrupt is nothing but it is a request from an I/O device for service by the processor. The processor executes an interrupt service routine (ISR) to service the same. Before servicing an interrupt the current state of the processor must be saved in memory locations. After ISR is completed the state of the processor is restored so that the interrupted program may continue.

Illustrate the steps involved in a typical memory read and write cycle with example?

Memory Address Register (MAR) and Memory Data Register (MDR) are the two registers which are facilitating the communication between the processor and memory. In order to read an information from memory the address of the memory location in which the information is residing have to be put in MAR and a Read control signal will be sent to memory unit. When the memory unit sees the address in address line of the bus and read signal in control line of the bus memory will start the read operation from the concerned address and the result will be sent to the MDR. From there the information can be transferred to any other registers inside the processor through internal processor bus. Similarly for Write operation initially the data to be written into the memory has to be placed in MDR and the address in which the desired information have to be kept in memory will be placed in MAR and a Write control signal will be sent to the memory unit. When the memory unit sees the address, data and write signal in the external memory bus it will start the corresponding write operation.



Memory Read Operation



Memory Write Operation

Explain load and store operations.

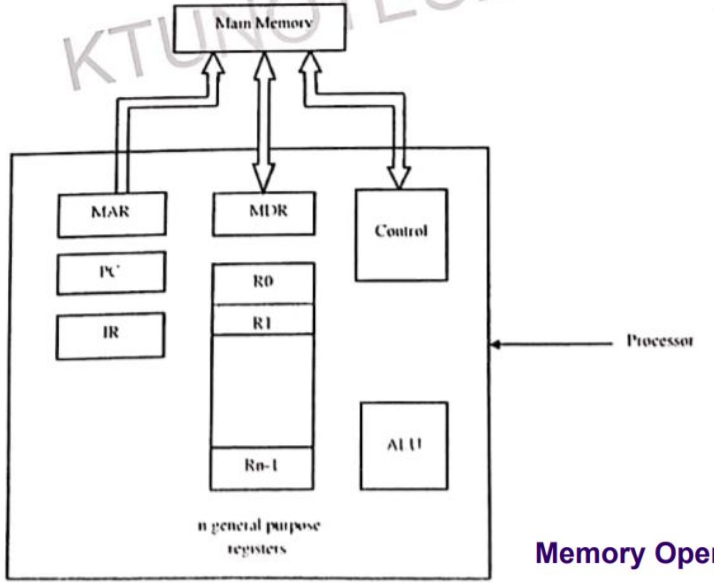


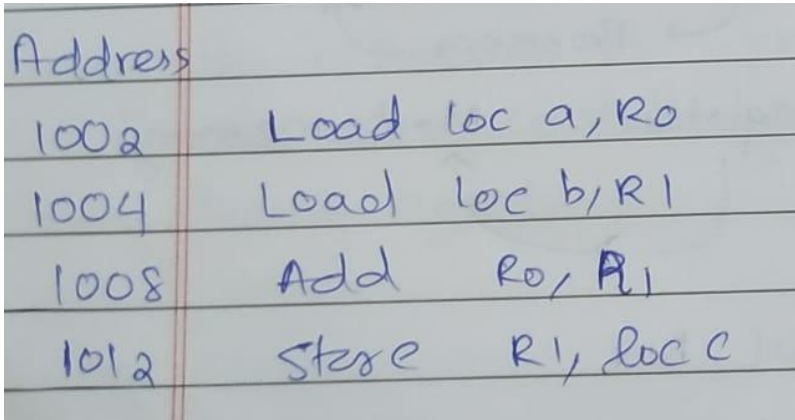
Fig 1.3: Connection between the processor and main memory

To execute an instruction, the words containing the instruction have to be brought out to the processor from memory. Operands and results also have to be moved in between of main memory and processor. Main operations are:

- **Load(Read or Fetch)**
  Transfer copy of the contents of a specific memory location to the processor. The memory contents remain unchanged.
- **Store(Write)**
  It transfers information from processor to memory. It will destroy the earlier content of the memory location. Information can be transferred between processor and memory in terms of bytes or words. One byte or one word can be transferred in a single operation.

Memory Operations

- Load (or Read or Fetch)
  - Copy the content
  - Memory content doesn't change
  - Address → MAR, read frm mem, content → MDR
  - Registers can be used.
- Store (or Write)
  - Overwrite the content in memory
  - Address → MAR and Data → MDR, write to mem
  - Registers can be used



- Programs reside in the memory.
- Inputs are entered through input devices.
- PC (Program counter) has address of the first instruction or next instruction to be executed.
- The contents of PC are transferred to MAR (Memory Address Register) and send over system bus to memory.
- A Read signal is sent to the memory by control unit.
- The instruction is fetched and **loaded** into MDR (Memory Data Register) after duration of memory access time.
- The contents of MDR are transferred to IR (Instruction Register).
- Now the instruction is ready to be decoded and executed.
- In order to perform the operation, ALU gets operands from General-purpose register (R0, … Rn-1) or MDR.
- After performing the operation in ALU, the result is **stored** back to General-purpose register (R0, … Rn-1).
- In order to write the data in the memory, the data is **stored** in the MDR. Then the write signal is produced by control unit (via control bus) and the data from MDR is written into memory (via the data bus) at the location specified in the MAR (transferred via address bus).
- During the execution, PC is incremented to point to the next instruction.

Write notes on bus structures. **List out and explain the 3 types of system buses.**

Bus Structures

Different functional units of a computer can be connected using a structure called bus system structure. There are external bus structures and internal bus structures. Bus interface between memory and processor is called external memory bus and bus structure inside the processor is called internal bus structures. Bus consists of three lines of carries, one for holding address, one for holding data and one for carrying control information.

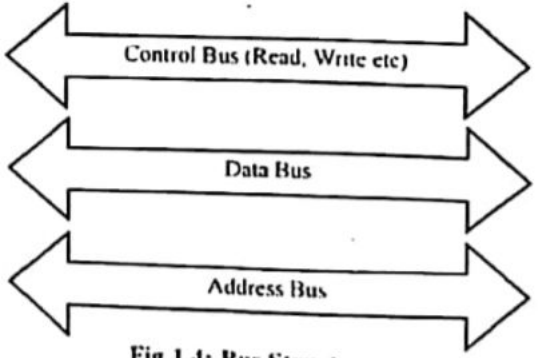


Fig 1.4: Bus Structure

Address Bus is a group of conducting wires which carries address only. It is unidirectional because address flow in one direction, from processor to memory. The Length of the address bus determines the amount of memory a system can address. For example: system with a 32-bit address bus can address 2^32 memory locations.

Data bus is a group of conducting wires which carries Data only. It is bidirectional because data flow in both directions, from microprocessor to memory or Input/Output devices and from memory or Input/Output devices to microprocessor. The width of the data bus is directly related to the largest number that the bus can carry, such as an 8 bit bus can represent 2^8 unique values, this equates to the number 0 to 255. A 16 bit bus can carry 0 to 65535.

Control bus is a group of conducting wires, which is used to generate timing and control signals to control all the associated peripherals, microprocessor uses control bus to process data, that is what to do with selected memory location. Some control signals are: Memory read, Memory write, I/O read, I/O Write, Opcode fetch.

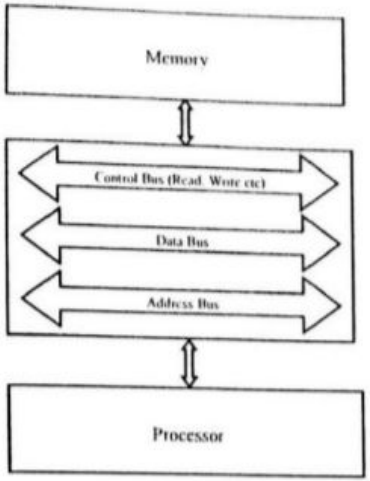


Fig 1.5: External processor bus

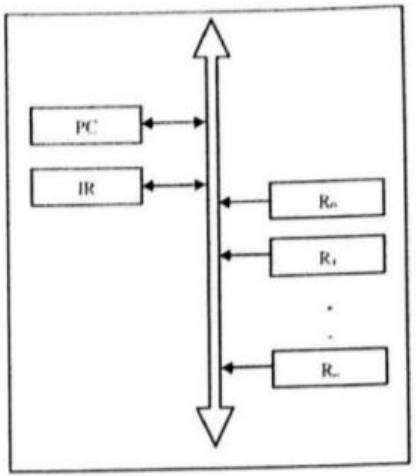


Fig 1.6: Internal processor bus

Bus Structures can be classified into two:

- **Single bus structure**
  Only two units can actively participate at a time, that is only one transfer takes place at a time.
  Advantages
  - Low cost
  - Flexibility for attaching peripheral devices

  Disadvantages
  - Only one transfer at a time
- **Multiple bus structure**
  It contains multiple buses, so that more than one transfer can takes place.
  Advantages
  - More concurrency in operations.
  - Better performance

  Disadvantages
  - Increased cost

What is meant by Byte addressable memory? Discuss Big- endian and Little endian assignments.

Basic information quantities are, bit, byte and word. Byte is of size 8 bit (always this is constant).Word length may from 16 to 64 bits. So normally distinct addresses will be assigned to each byte location. This is known as byte addressability.

Big-Endian and Little-Endian Assignments

Byte addresses can be assigned across words in two ways:

- **Big – Endian**
  Lower byte addresses are used for most significant bytes of the word.
- **Little – Endian**
  Lower byte addresses are used for least significant bytes of the word.

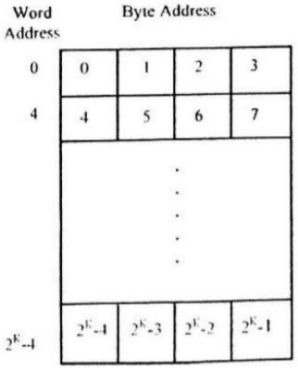


Fig 1.10(a): Big- Endian Assignment

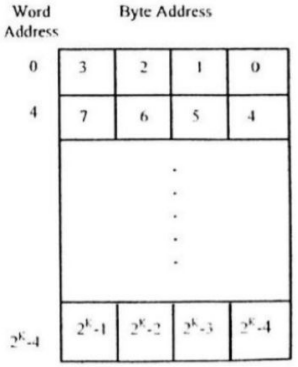
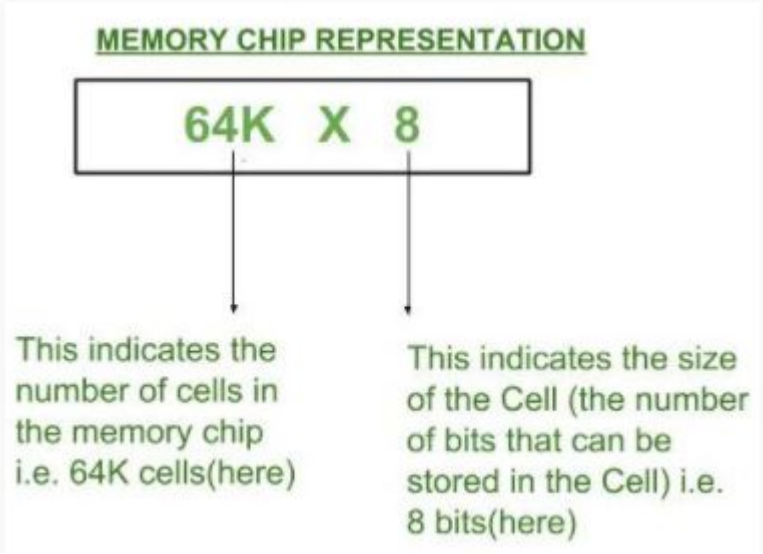


Fig 1.10(b): Little- Endian Assignment



The following information can be obtained from the memory chip representation shown above:

1. **Data Space in the Chip =** 64K X 8
2. **Data Space in the Cell =** 8 bits
3. **Address Space in the Chip =** $\log_2(64K)$=16 bits

Now we can clearly state the difference between Byte Addressable Memory & Word Addressable Memory.

Byte Addressable Memory	Word Addressable Memory
When the ***data space in the cell = 8 bits*** then the corresponding ***address space*** is called as **Byte Address**.	When the ***data space in the cell = word length of CPU*** then the corresponding ***address space*** is called as **Word Address**.
Based on this data storage i.e. *Bytewise storage*, the memory chip configuration is named as **Byte Addressable Memory**.	Based on this data storage i.e. *Wordwise storage*, the memory chip configuration is named as **Word Addressable Memory**.
For eg. : **64K X 8** chip has 16 bit Address and **cell size = 8 bits (1 Byte)** which means that in this chip, data is stored **byte by byte**.	For eg. : For a 16-bit CPU, **64K X 16** chip has 16 bit Address & **cell size = 16 bits (Word Length of CPU)** which means that in this chip, data is stored **word by word**.

- To retrieve information from memory, either for one word or one byte (8-bit), addresses for each location are needed.
- A *k*-bit address memory has 2^k memory locations, namely $0 - 2^k-1$, called memory space.
- 24-bit addr memory: 2^{24} = 16,777,216 = 16M ($1M=2^{20}$)
- 32-bit addr memory: 2^{32} = 4G ($1G=2^{30}$)
- 1K(kilo)=2^{10}
- 1T(tera)=2^{40}

KB
MB
GB
TB

Problems

1. A memory has 32-bit address and byte-addressable, what is the size of the memory (in bytes)?
2. A memory has 24-bit address and word-addressable with a word length of 32 bits, what is the size of the memory (in bytes)?
3. A memory has 16-bit address and byte addressable. Word length is 32 bits. How many words can we store in such a memory?

Note: A memory with k-bit address

a) byte-addressable → 2^k bytes

b) word-addressable → 2^k words

- Size of memory = 2^address size in bytes if its byte addressable and in words if its word addressable.
- Size of memory in bytes = No.of words x bytes/word
- Size of memory in words = No. of bytes x bytes/word

Answers

1. 32-bit address, byte addressable memory

   No of bytes = 2^{32} = $2^{30} \times 2^2$

   = 4G bytes ($1G = 2^{30}$)

2. 24-bit address, word addressable, 1Word =32 bits (4 bytes)

   No of words = 2^{24} = $2^{20} \times 2^4$

   No of bytes = No of words x (bytes/word)

   = $2^{20} \times 2^4 \times 2^2$ (1 word = 4 bytes)

   = 64M bytes ($1M = 2^{20}$)

3. 16-bit address, byte addressable, 1Word =32 bits (4 bytes)

   No of words = No of bytes / (bytes/word)

   = $2^{16} / 2^2$ = $2^{10} \times 2^6 / 2^2$ (1 word = 4 bytes)

   = 16 K words ($1K = 2^{10}$)

Outline various types of basic instructions. **OR** Explain about instruction type and instruction format.

Write notes on three address, two address and one address instructions, giving example for each.

Zero Address Instructions

Instructions can be used with zero operands in which the locations of all operands are specified implicitly. (It is possible by storing the operands in a structure called pushdown stack).

- **Three Address Instructions**

Syntax

Operation source1, source2, destination

Example

Add A, B, C

Where, A and B are source operands and C is destination operand. Add is the operation to be performed on operands. Suppose that K bits are needed to specify the memory address of each operand, and then totally 3K bits are needed totally to specify the memory address of all operands in the above sample case. In addition, some bits are needed to denote Add instruction also. Three address instruction is too large to fit in one word for most cases. An alternative approach is to use two address instructions.

Two Address Instructions

Syntax

Operation source, destination

Example

Add A, B

Which performs the operation B ←[A] + [B]. Square bracket indicates the contents of the location specified. That is, it adds the contents of A and B and the result is stored back to B. Here the value of location B will be overwritten. To preserve the value of B, we can go for another instruction Move B, C. It moves the content of B to C, leaving the contents of location B unchanged. We couldn't find an alternative approach by using a single two address instruction.

Add A, B → Move B, C
Add A, C
(Leave B unchanged)

(Single two address instruction)

Here, in all instructions we are adopting a scheme such that source operand is specified first then destination operand. This may not be the case with all architectures. In some of the machine instructions, it will follow a scheme of destination first, and then source. There is no uniform scheme for specifying operands. Even two address instructions also, may not be fit into one word for usual word length. Another possibility is to have machine instructions with single operand. These are called One Address Instructions.

One Address Instructions

Here one of the register called Accumulator is implicit in all cases.

Example

Add A

Add the content of memory location A to the content of the accumulator register and places the sum back to accumulator.

Question

Represent C←[A] + [B] in terms of one address instruction

Ans

Load A
Add B
Store C

Load instruction copies the contents of memory location A into accumulator. Add B, adds the content of B to Accumulator and Store C, and stores the content of accumulator to memory location C.

Question

Write the instruction sequence for C=A+B (suppose that arithmetic operations are allowed only on register operands)

Answer

```
Move A, Ri
Move B, Rj
Add  Ri , Rj
Move Rj, C
```

Question

Write the instruction sequence for C=A+B(suppose that one operand in memory, other in register)

Answer

```
Move A, Ri
Add B, Ri
Move Ri, C
```

What are condition codes? List the different condition codes.

- **Condition Codes**

  Whenever a conditional branch instruction is executed, it may require the results of various operations (to check the condition). Processors keeping this information in individual bits called condition code flags. These flags are grouped together in a special processor register called condition code Register or Status register.

  Four commonly used flags are:

  - N (Negative) → Set to 1 if result is negative, otherwise zero.
  - Z (Zero) → Set to 1 if result is zero, otherwise cleared to zero.
  - V (Overflow) → Set to 1 if arithmetic overflow occur, otherwise cleared to zero.
  - C (Carry) → Set to 1 if a carry after the operation, otherwise cleared to zero.

  In the above example, Branch>0 tests the condition code flags N and Z .That is ,the branch is taken only if register R either contain a negative or zero value.

With the help of examples, explain the different addressing modes.

OR

Describe any 4 addressing modes with examples.

OR

Explain Direct and Indirect addressing with examples.

OR

What is meant by addressing mode? Explain absolute and indirect addressing modes with suitable examples.

The different ways in which the location of an operand is specified in an instruction is called addressing mode.

Different types of addressing modes are:

- Immediate mode
- Register mode and Absolute mode
- Indirect mode
- Index mode - Base with Index, Base with Index and offset
- Relative mode
- Auto Increment and decrement mode

There are two modes for accessing modes to access the variables. They are

- **Register Mode**
  The operand is the contents of the processor register. The name (address) of the register is given in the instruction.
- **Absolute Mode (Direct Mode)**
  The operand is in new location. The address of this location is given explicitly in the instruction.

Example

MOVE LOC, R2

The above instruction uses the register and absolute mode. The processor register is the temporary storage where the data in the register are accessed using register mode. The absolute mode can represent global variables in the program.

Mode Assembler	Syntax	Addressing Function
Register mode	Ri	EA=Ri
Absolute mode	LOC	EA=LOC

Where, EA is Effective Address

Address and data constants can be represented in assembly language using Immediate Mode.

- **Immediate mode**
  The operand is given explicitly in the instruction.
  Example

Move 200 immediate, R0

It places the value 200 in the register R0.The immediate mode used to specify the value of source operand. In assembly language, the immediate subscript is not appropriate so # symbol is used. It can be re-written as:

Move #200, R0

Assembly Syntax	Addressing Function
Immediate #value	Operand =value

What do you mean by effective address of an operand?

Instruction does not give the operand or its address explicitly. Instead it provides information from which the new address of the operand can be determined. This address is called effective Address (EA) of the operand.

➢ **Indirect Mode**

The effective address of the operand is the contents of a register. We denote the indirection by the name of the register or new address given in the instruction.

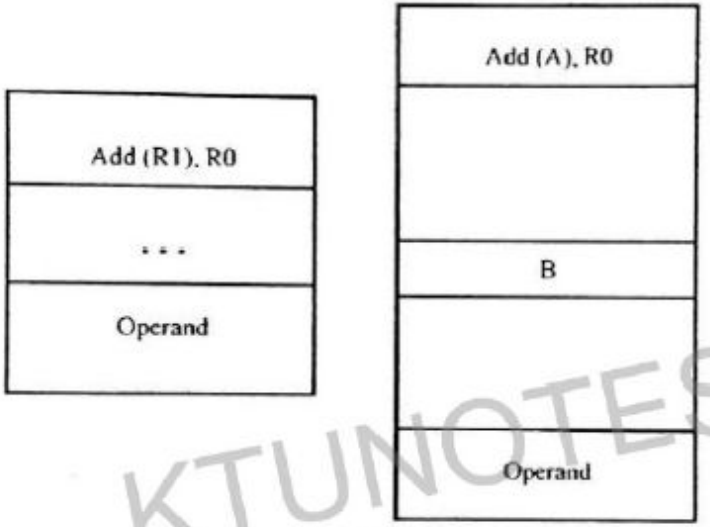


Fig 1.14: Indirect mode

Address of an operand (B) is stored into R1 register. If we want this operand, we can get it through register R1 (indirection). The register or new location that contains the address of an operand is called the pointer.

Mode	Assembler Syntax	Addressing Function
Indirect	Ri, LOC	EA=[Ri] or EA=[LOC]

- **Index Mode**

The effective address of an operand is generated by adding a constant value to the contents of a register. The constant value uses either special purpose or general purpose register. We indicate the index mode symbolically as,

X(Ri)

Where,

X – denotes the constant value contained in the instruction

Ri – It is the name of the register involved

The Effective Address of the operand is,

EA=X + [Ri]

The index register R1 contains the address of a new location and the value of X defines an offset (also called a displacement).

To find operand,

1. First go to Reg R1 (using address)-read the content from R1-1000
2. Add the content 1000 with offset 20 get the result.

   1000+20=1020
3. Here the constant X refers to the new address and the contents of index register define the offset to the operand.
4. The sum of two values is given explicitly in the instruction and the other is stored in register.

Example

Add 20(R1) , R2 (or) EA=>1000+20=1020

Index Mode	Assembler Syntax	Addressing Function
Index	X(Ri)	EA=[Ri]+X
Base with Index	(Ri,Rj)	EA=[Ri]+[Rj]
Base with Index and offset	X(Ri,Rj)	EA=[Ri]+[Rj]+X

➢ **Relative Addressing**

It is same as index mode. The difference is, instead of general purpose register, here we can use program counter (PC).

- **Relative Mode**

  The Effective Address is determined by the Index mode using the PC in place of the general purpose register (gpr). This mode can be used to access the data operand. But its most common use is to specify the target address in branch instruction.

  Example

  Branch>0 Loop

  It causes the program execution to goto the branch target location. It is identified by the name loop if the branch condition is satisfied.

Mode	Assembler Syntax	Addressing Function
Relative	X(PC)	EA=[PC]+X

➢ Additional Modes

There are two additional modes. They are

- **Auto-increment mode**

  The Effective Address of the operand is the contents of a register in the instruction. After accessing the operand, the contents of this register is automatically incremented to point to the next item in the list.

Mode	Assembler Syntax	Addressing Function
Auto-increment	(Ri)+	EA=[Ri]; Increment Ri

- **Auto-decrement mode**

  The Effective Address of the operand is the contents of a register in the instruction. After accessing the operand, the contents of this register is automatically decremented to point to the next item in the list.

Mode	Assembler Syntax	Addressing Function
Auto-decrement	-(Ri)	EA=[Ri]; Decrement Ri

Write notes on different instruction sequencing techniques.

Straight line sequencing

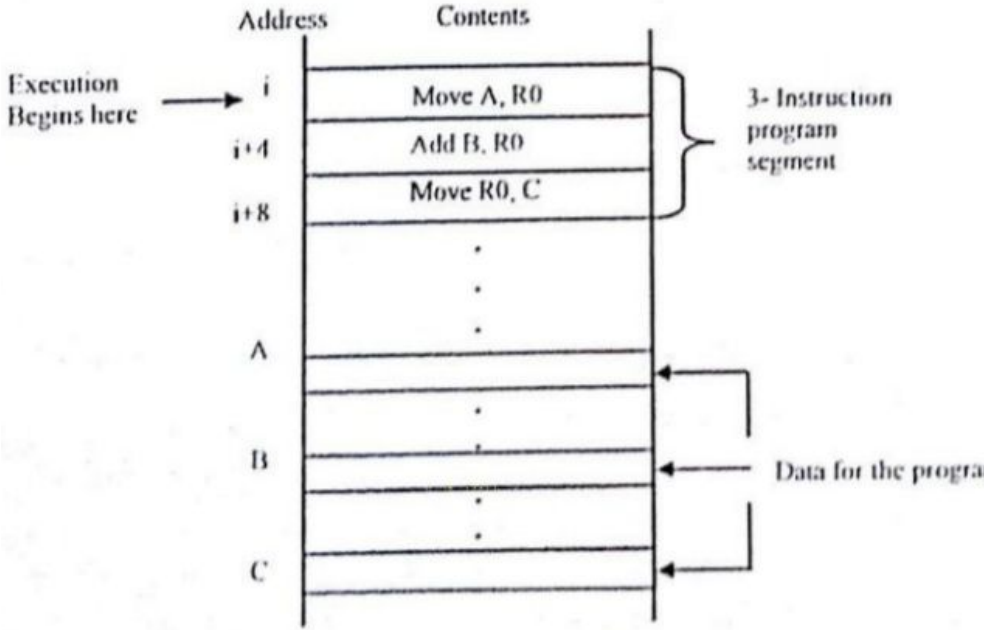


Fig 1.11: Program execution

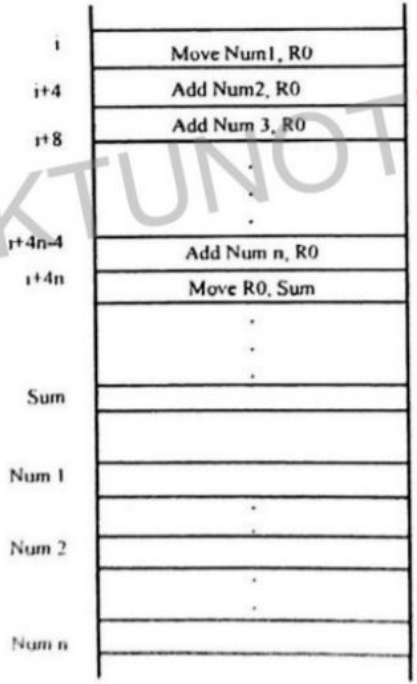


Fig 1.12: Straight Line Sequencing Program for adding 'n' numbers

Initially PC (Program Counter) contains the address of the next instruction to be executed. In this example initially address i is placed in PC. The processor control circuitry use the information in PC to fetch and execute instructions, one at a time, in the order of increasing addresses. This is called straight line sequencing .While the instruction is being executed, the value of PC will be updated by incrementing 4 bytes, (because here length of instruction is 4 bytes). Instruction execution consists of 2 phases:

- **Instruction Fetch**
  Instruction is fetched from memory location whose address is in PC. This instruction is placed in IR (Instruction Register).
- **Instruction Execute**
  Instruction in IR (Instruction Register) is analyzed to see which operation have to be performed by the processor. This may include several operations like fetching of operands from memory (or from processor registers), performing an ALU operation, storing of result into memory etc.

After the execution phase, PC will contain the address of the next instruction to be executed. Then a new instruction fetch phase will begin.

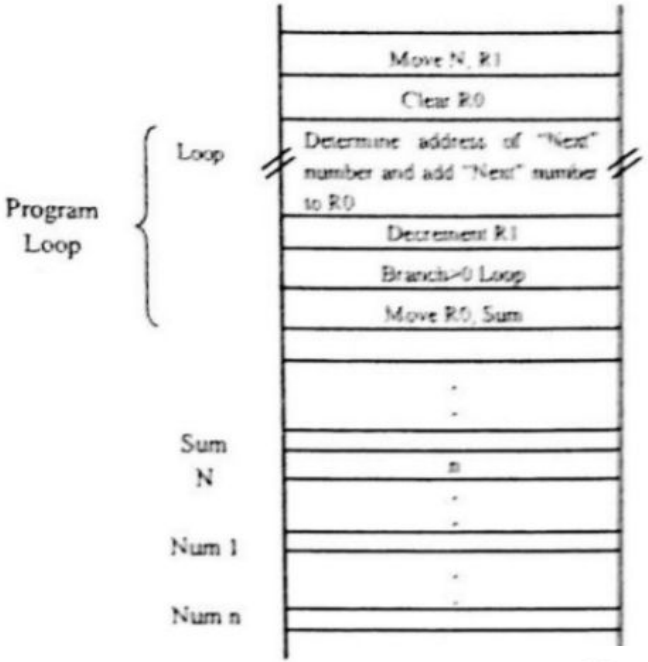


Fig 1.13: Using loop to add

Branching

No of entries in the list (n) is stored at memory location N. After each addition operation this value is decremented by one. This value is a number greater than zero that means again there are numbers to be added to the list.

Branch Instruction loads a new value into the program counter .The processor fetches and executes the instructions from these addresses (known as branch target) instead of loading the instruction from address of PC. A conditional branch instruction causes a branch only if the specified condition is satisfied. In other cases, PC is incremented in the normal way and the next instruction in sequential address order is fetched and executed.

What is meant by an instruction cycle? Describe its two phases?

Instruction Cycle

A program that exists inside a computer's memory unit consists of a series of instructions. The processor executes these instructions through a cycle for each instruction.

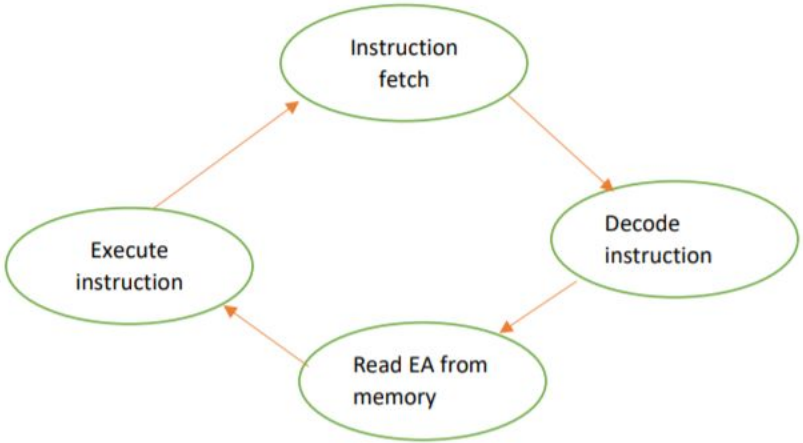
In a basic computer, each instruction cycle consists of the following phases:

Instruction fetch: fetch instruction from memory

Decode the instruction: what operation to be performed.

Read the effective address from memory

Execute the instruction



Registers Involved In Each Instruction Cycle:

Memory address registers(MAR) : It is connected to System Bus address lines. It specifies the address of a read or write operation in memory.

Memory Buffer Register(MBR) : It is connected to the data lines of the system bus. : It is connected to the system bus Data Lines. It holds the memory value to be stored, or the last value read from the memory.

Program Counter(PC) : Holds the address of the next instruction to be fetched.

Instruction Register(IR) : Holds the last instruction fetched.

Fetch cycle

The address of the next instruction to execute is in the Program Counter(PC) at the beginning of the fetch cycle.

Step 1: The address in the program counter is transferred to the Memory Address Register(MAR), as this is the only register that is connected to the system bus address lines.

Step 2: The address in MAR is put on the address bus, now a Read order is provided by the control unit on the control bus, and the result appears on the data bus and is then copied into the memory buffer register. Program counter is incremented by one, to get ready for the next instruction. These two acts can be carried out concurrently to save time.

Step 3: The content of the MBR is moved to the instruction register(IR).

Instruction fetch cycle consist of four micro operation:

T1: MAR← PC

T2: MBR← memory

PC← PC + stepsize or length of instruction

T3: IR← MBR

Decode instruction cycle

The next move is to fetch source operands once an instruction is fetched. Indirect addressing (it can be obtained by any addressing mode, here it is done by indirect addressing) is obtained by Source Operand. You don't need to fetch register-based operands. If the opcode is executed, it will require a similar process to store the result in main memory. Micro-operations take place:

T1: MAR← IR(address)

T2: MBR← Memory

T3: IR(address)← (MBR(Address))

Step 1: The instruction address field is passed to the MAR. This is used to fetch the operand 's address.

Step 2: The address field of the IR is updated from the MBR.

Step 3: The IR is now in the state. Now IR is ready for the execute cycle.

Execute instruction Cycle

The initial three cycles (Fetch, Indirect, and Interrupt) are predictable and quick. Each requires simple , small, and fixed micro-operation sequences. The same micro-operation is repeated every time around in each event. Execute instruction cycle is different from them. Like, there is N different series of micro-operations that can occur for a computer with different N opcodes.

Example

ADD R , X

T1: MAR← (IR(address))

T2: MBR← Memory

T3: R← (R) + (MBR)

Step 1: The address portion of IR is loaded into the MAR.
Step 2: The address field of the IR is updated from the MBR, so the reference memory location is read.
Step 3: Now, the contents of R and MBR are added by the ALU.

Explain single bus organization with the help of a diagram. Specify with examples, how memory operations are done in the given organization.

Give the control sequence for execution of instruction Add[R2],R1 using a single bus organization.

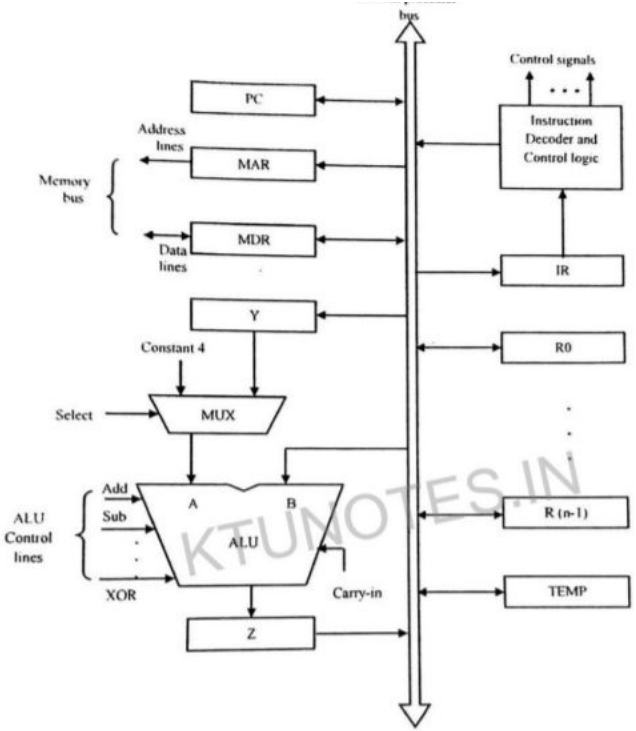
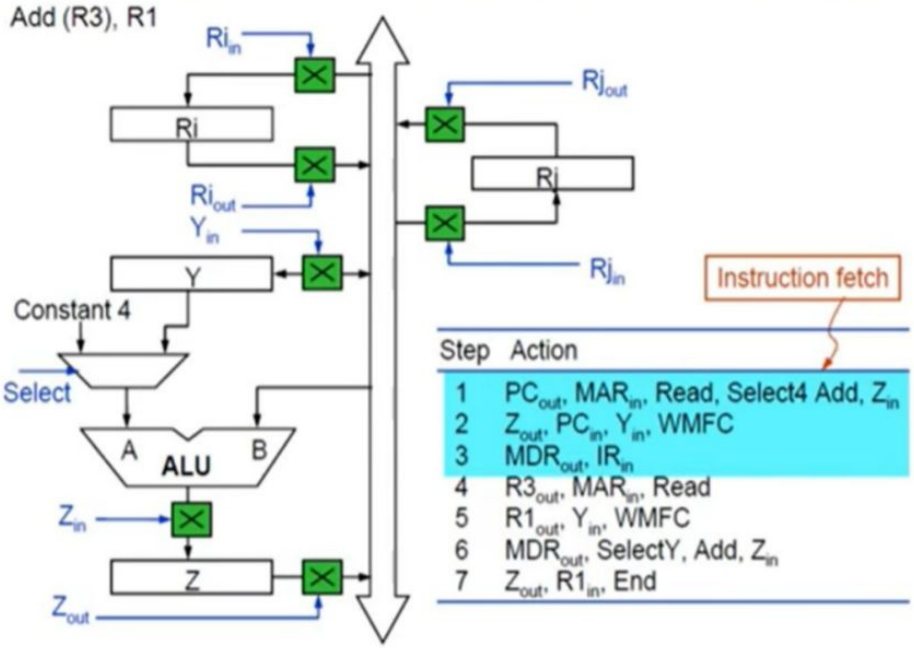


Fig 2.1: Single-bus organization of the datapath inside a processor

Control Sequence - Execution of a Complete Instruction



Explain the process of storing a word in memory using a single bus organization. Specify which all control signals will be activated.

Storing a word in memory

Writing a word into a memory location follows a similar procedure. The desired address will be loaded in MAR, the data to be written are loaded in MDR, and then a write command is issued.

Example MOVE R2, (R1)

$R1_{out}$, MAR_{in}
$R2_{out}$, MDR_{in}, WRITE
MDR_{outE}, WMFC

The action specified in one step can be completed in one clock cycle. Exemption may come for some steps like MFC depending on the speed of the addressed device.

Write down the sequence of actions needed to fetch and execute the instruction: Store R6, X(R8).

PCout , MARin, Read, Select 4 , Add , Zin
Zout , PCin , Yin , WFMFC
MDRout, IRin
R8out,MARin,Read,WMFC
MDRout, X, ADD, Zin
Zout,R6in,End

Store R6, X(R8)
stores the contents of register R6 into memory location X + [R8]. It can be implemented as follows:
1. Fetch the instruction and increment the program counter.
2. Decode the instruction and read registers R6 and R8.
3. Compute the effective address X + [R8].
4. Store the contents of register R6 into memory location X + [R8].
5. No action.

Enumerate the sequence of actions involved in executing an unconditional branch instruction

- Control sequence for an unconditional branch instruction is as follows:
    1) PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
    2) Z_{out}, PC_{in}, Y_{in}, WMFC
    3) MDR_{out}, IR_{in}
    4) Offset-field-of-IR_{out}, Add, Z_{in}
    5) Z_{out}, PC_{in}, End

Write notes on multiple bus organization.

OR

Explain the data path implementation for the instruction ADD (R3), R1 in a 3-bus processor unit.