1. a.Calculate the SPEC rating for the program suite under test. Running times of the program suite for reference PC and PC under test are given below:

| Programs | Runtime on reference computer | Runtime in new computer |
|---|---|---|
| 1 | 50 Minutes | 5 Minutes |
| 2 | 75 Minutes | 4 Minutes |
| 3 | 60 Minutes | 6 Minutes |
| 4 | 30 Minutes | 3 Minutes |

**Solution:**

$$SPEC\ rating = \frac{\text{Running time on the reference computer}}{\text{Running time on the computer under test}}$$

$$SPEC\ rating = \left(\prod_{i=1}^{n} SPEC_i\right)^{\frac{1}{n}}$$ Where n is the number of programs in the suite.

**50/5*75/4*60/6*30/3]**

**= 10*18.75*10*10**

**=10**

**b.** Write ALP program to copy 'N' numbers from array 'A' to array 'B' using indirect addresses. (Assume A and B are the starting memory location of an array).

**Ans:**

**Program:**

```
Move N, R1              ; initialize counter R1=N
Move #A, R2             ; R2 is used as a pointer to the numbers in the list A
Move #B, R3             ; R3 is used as a pointer to the numbers in the list B
Clear R0                ; set sum R0=0
LOOP Move (R2), R0      ; move data pointed by R2 into R0 register
Move R0, (R3)           ; move R0 content to memory pointed by R3
Decrement R1            ; decrement counter R1
Branch > 0 LOOP         ; repeat until counter R1 becomes 0
```

2.a. What is a stack frame? Explain a commonly used layout for information in a subroutine stack frame.

- Stack Frame refers to locations that constitute a private work-space for the subroutines
- The work space is
  - Created at the time the subroutine is entered and
  - Freed up when the subroutine returns control to the calling program

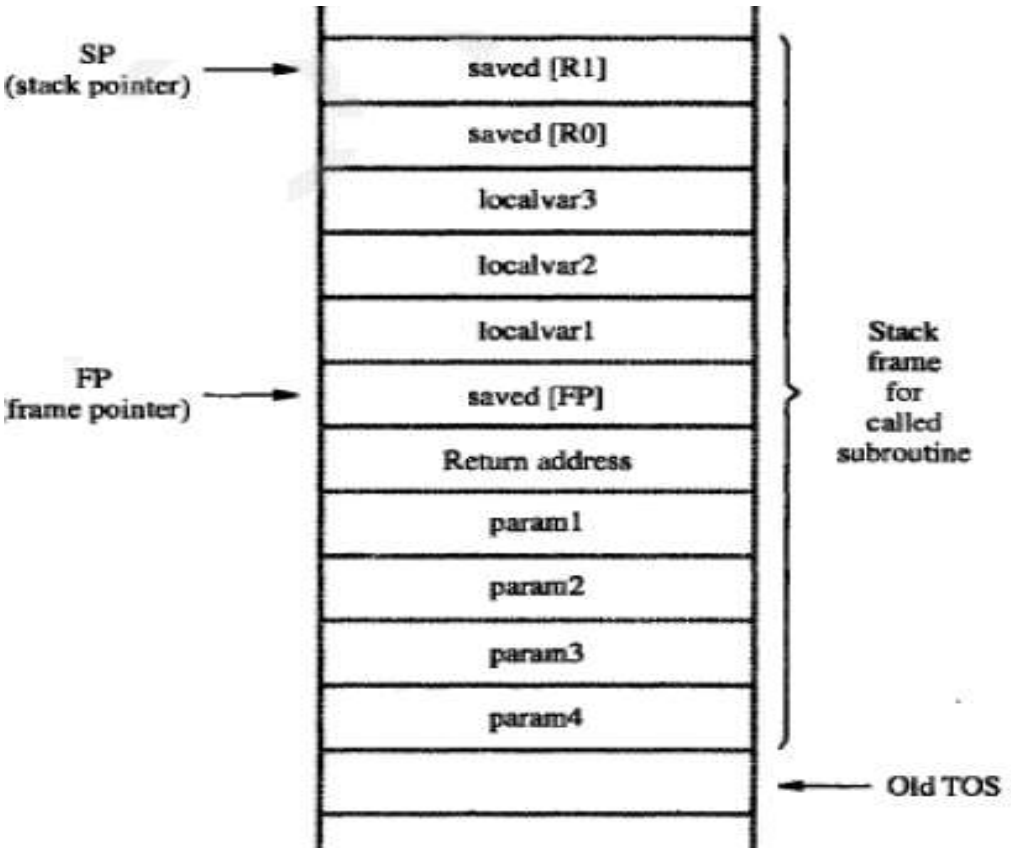



Figure. Subroutine stack frame example

2.b.

- There are many applications that require the bits of an operand to be shifted right or left some specified number of bit positions.
- The details of how the shifts are performed depend on whether the operand is assigned number or some more general binary-coded in formation.
- For general operands, we use a logical shift.

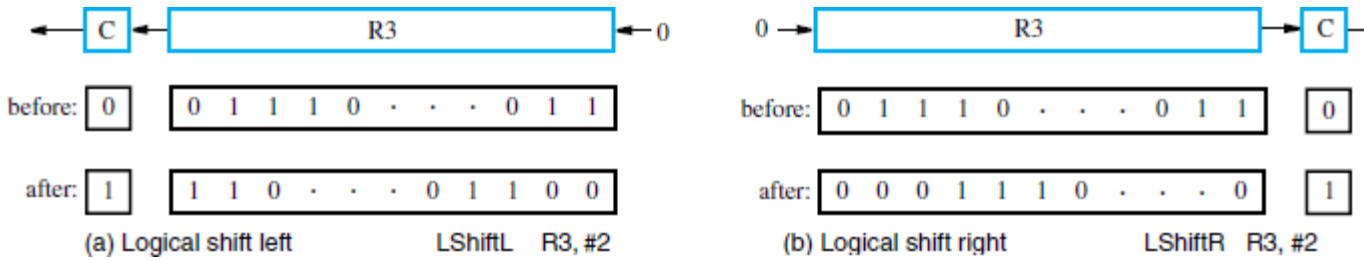
For a number ,we use an arithmetic shift ,which preserves the sign of the number.

## LOGICAL SHIFTS

- Two logical shift instructions are

    1) Shifting left (LShiftL) &
    2) Shifting right (LShiftR).

- These instructions shift an operand over a number of bit positions specified in a count operand contained in the instruction.



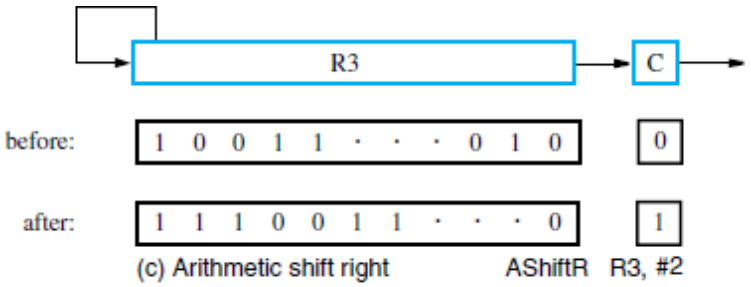


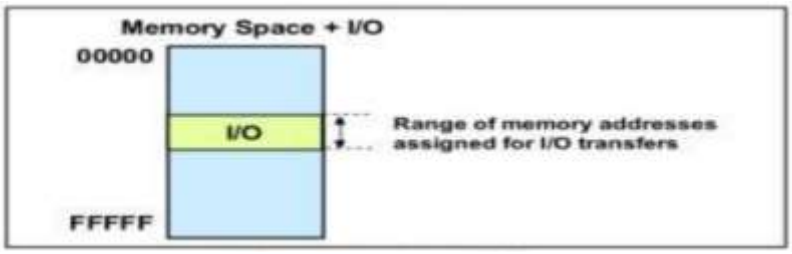

**Figure 2.23** Logical and arithmetic shift instructions.

3. a Define memory-mapped I/O and I/O mapped I/O with examples

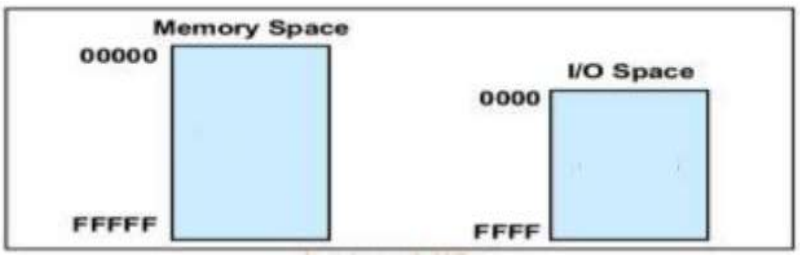
Two ways to assign addresses to I/O devices:

1. Memory-mapped I/O
   - Devices & memory share same address space.
   - I/O looks just like memory read/write.
   - No special instructions for I/O ➔ "load", "store", ... etc.
   - Bus has one Read & one Write control line.
   - **Pros**: Large selection of memory access instructions.
   - **Cons**: Valuable memory address space is used up!
2. Isolated I/O
   - Separate address space for devices.
   - Special instructions for I/O ➔ "in", "out", "test", ... etc.
   - Need two Rd & two Wr control lines.
   - **Pros**: efficient use of memory address space.
   - **Cons**: Not so many I/O instructions.

**Memory mapped I/O**

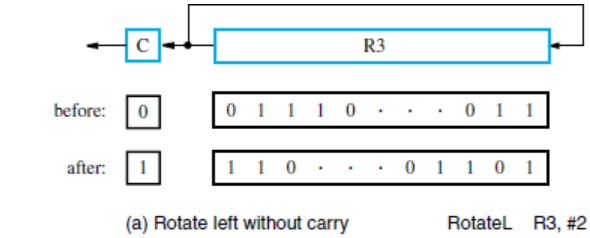



## 2. I/O (or Isolated) mapped I/O

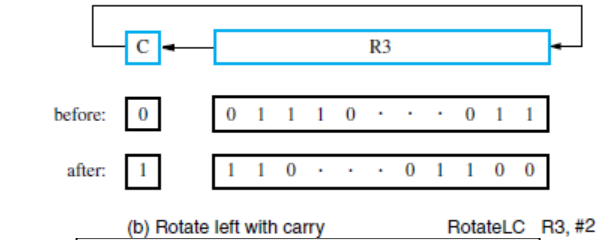


## ROTATE OPERATIONS

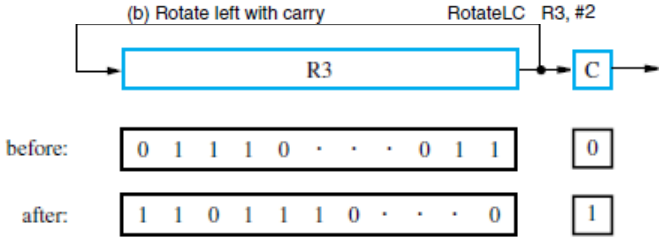
- In shift operations , the bits shifted out of the operand are lost, except for the last bit shifted out which is retained in the Carry-flag C.
- To preserve all bits a set of rotate instructions can be used.
- They move the bits that are shifted out of one end of the operand back into the other end.
- Two versions of both the left and right rotate instructions are usually provided .In one version, the bits of the operand is simply rotated.
- In the other version, the rotation includes the C flag.



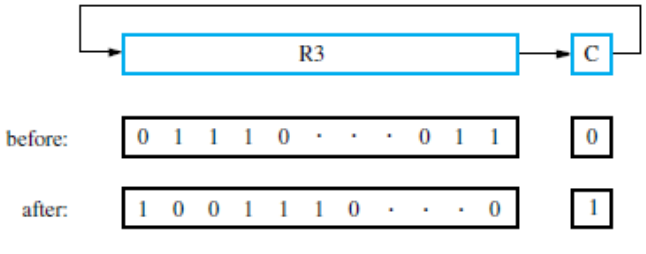

(a) Rotate left without carry RotateL R3, #2




(b) Rotate left with carry RotateLC R3, #2




(c) Rotate right without carry RotateR R3, #2



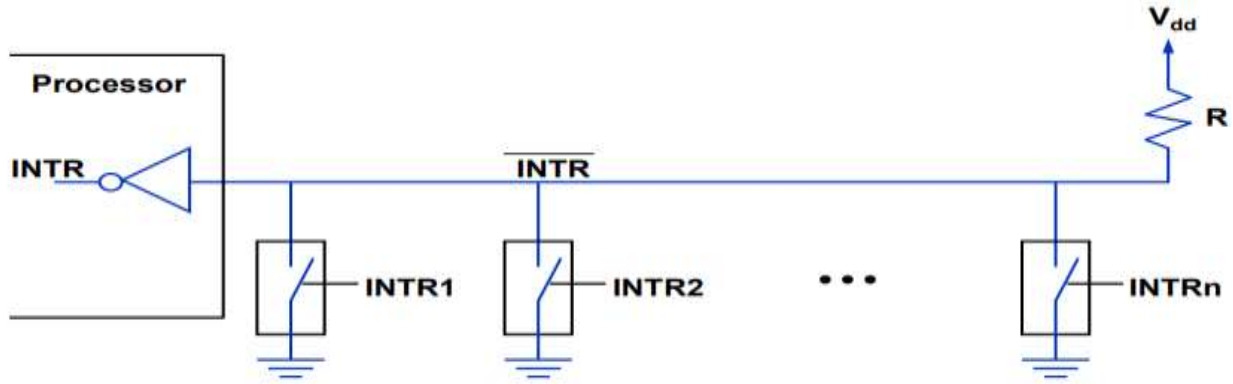

(d) Rotate right with carry RotateRC R3, #2

**Figure 2.25** Rotate instructions.

3.b Drawback of a Program controlled I/O: **Wait-loop**

- Instead of using wait-loop, let I/O device alert the processor when it is ready.
- Hardware sends an **interrupt-request signal** to the processor at the appropriate time, much like a **phone call**.
- Interrupt is a mechanism used by most processors to handle asynchronous type of events.
- Essentially, the interrupts allow devices to request that the processor stops what it is currently doing and executes software (called Interrupt Service Routine) to process the device's request, much like a Subroutine call that is initiated by the external device rather than by the program running on the processor.

Interrupts are also used when the processor needs to perform a long-running operation on some I/O device and wants to be able to do other work while waiting for the operation to complete




INTR=INTR1+INTR2+…+INTRn

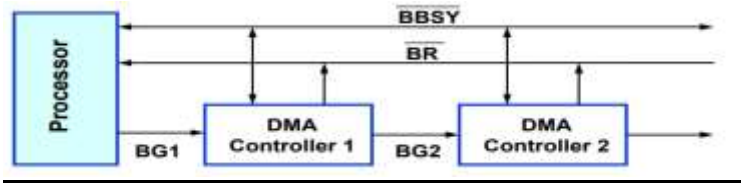
4.a & 4.b What are the different methods of DMA? Explain them in brief.

Direct memory access (DMA) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, without the continuous intervention by the processor. To transfer large blocks of data at high speed, DMA approach will be used.

**Centralized Bus Arbitration**

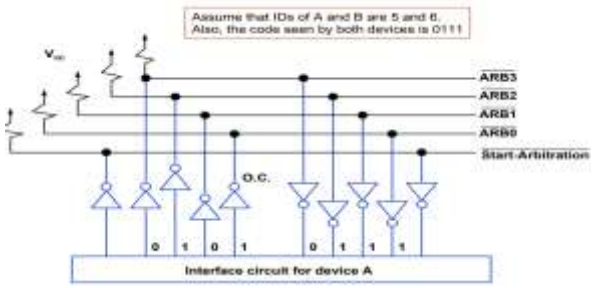
Explain: i) Interrupt enabling; ii) Interrupt disabling; iii) Edge triggering, with respect to Interrupts.

- A single bus-arbiter performs the required arbitration
- Normally, processor is the bus-master.
- Processor may grant bus-mastership to one of the DMA controllers.
- A DMA controller indicates that it needs to become bus-master by activating BR line.
- The signal on the BR line is the logical OR of bus-requests from all devices connected to it.
- Then, processor activates BG1 signal indicating to DMA controllers to use bus when it becomes free.
- BG1 signal is connected to all DMA controllers using a daisy-chain arrangement.
- If DMA controller-1 is requesting the bus,
  Then, DMA controller-1 blocks propagation of grant-signal to other devices.
  Otherwise, DMA controller-1 passes the grant downstream by asserting BG2.
- Current bus-master indicates to all devices that it is using bus by activating BBSY line.
- The bus-arbiter is used to coordinate the activities of all devices requesting memory transfers.
- Arbiter ensures that only 1 request is granted at any given time according to a priority scheme.
  (BR → Bus-Request, BG → Bus-Grant, BBSY → Bus Busy).




Distributed Bus Arbitration

- No central arbiter used
- Each device on bus is assigned a 4-bit identification number.
- When one or more devices request the bus, they assert the Start-Arbitration signal and place their 4-bit ID number on ARB[3..0].
- The request that has the highest ID number ends up having the highest priority.
- Advantages – offers higher reliability (operation of the bus is not dependent on any one device).
- Assume that two devices, A and B, having ID numbers 5 and 6, respectively, are requesting the use of the bus.
- Device A transmits the pattern 0101, and device B transmits the pattern 0110.
- The code seen by both devices is 0111.
- Each device compares the pattern on the arbitration lines to its own ID, starting from the most significant bit.
- If it detects a difference at any bit position, it disables its drivers at that bit position and for all lower-order bits. It does so by placing a 0 at the input of these drivers.
- In the case of our example, device A detects a difference on line ARB 1. Hence, it disables its drivers on lines ARB1 and ARB0.
- This causes the pattern on the arbitration lines to change to 0110, which means that B has won the contention.

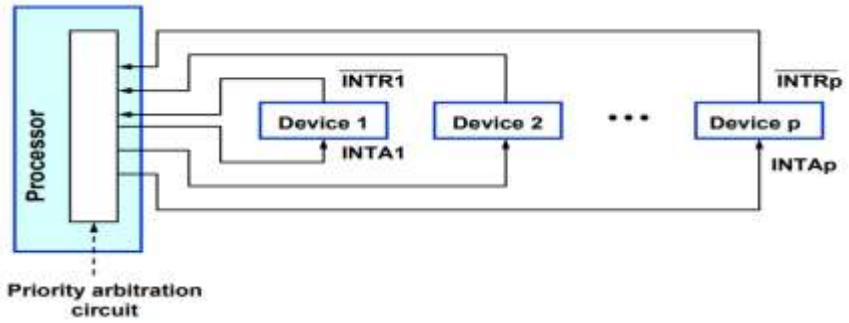



5.a Explain: i) Interrupt enabling; ii) Interrupt disabling; iii) Edge triggering, with respect to Interrupts.

5.b Explain in detail, the situations where a number of devices capable of initiating interrupts are connected to the processor? How to resolve the problems?

- when multiple devices can send interrupt requests to the processor.
    - During the execution of an interrupt service routine of device, the processor does not accept interrupt requests from any other device.
    - Since the interrupt service routines are usually short, the delay that this causes is generally acceptable.
- However, for certain devices this delay may not be acceptable.
    - Which devices can be allowed to interrupt a processor when it is executing an interrupt service routine of another device?
- Interrupt Nesting: A mechanism has to be developed such that even though a processor is executing an ISR, another interrupt request should be accepted and serviced
- I/O devices are organized in a priority structure:
    - An interrupt request from a high-priority device is accepted while the processor is executing the interrupt service routine of a low priority device.
- A priority level is assigned to a processor that can be changed under program control.
    - Priority level of a processor is the priority of the program that is currently being executed.
    - When the processor starts executing the interrupt service routine of a device, its priority is raised to that of the device.

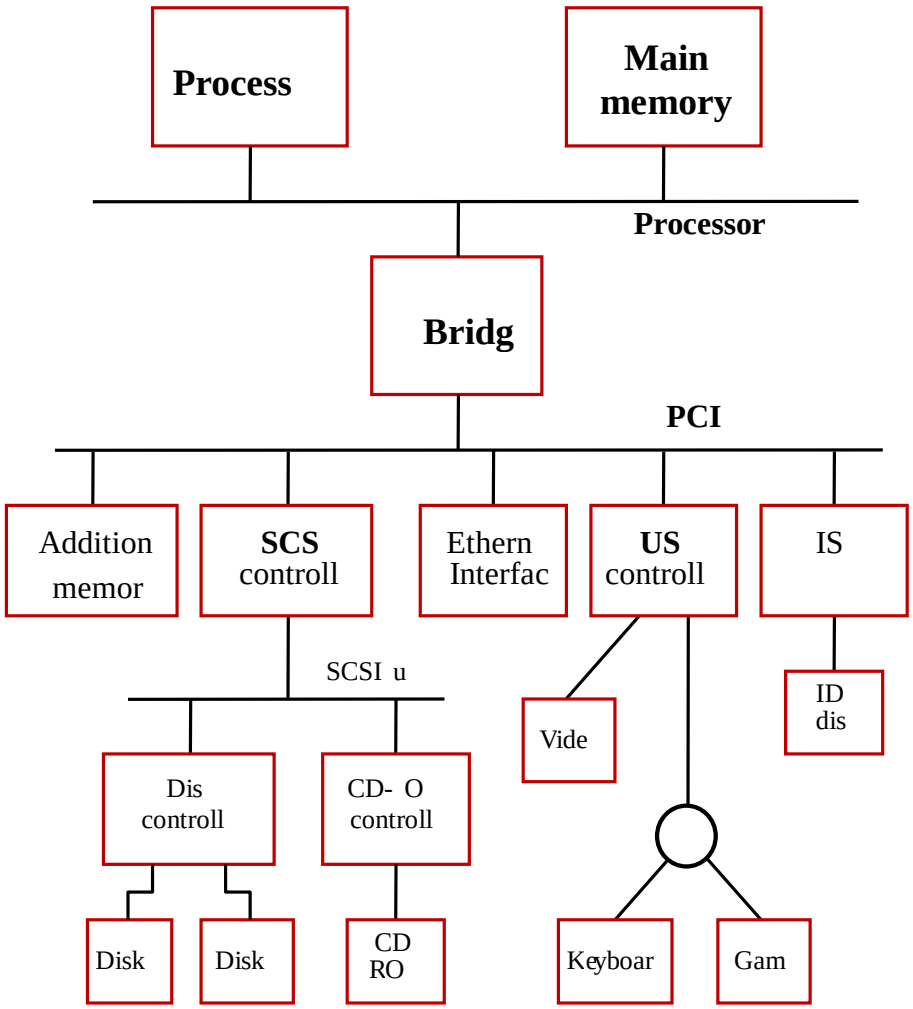
      If the device sending an interrupt request has a higher priority than the processor, the processor accepts the interrupt request

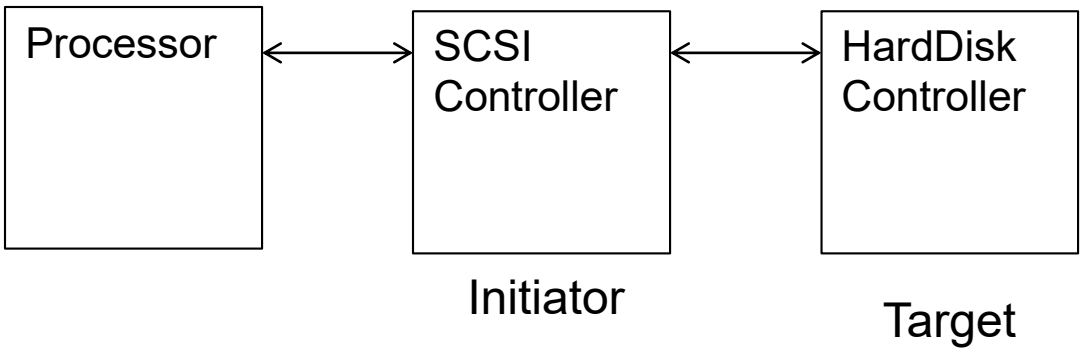



- Each device has a separate interrupt-request and interrupt-acknowledge line.
- Each interrupt-request line is assigned a different priority level.
- Interrupt requests received over these lines are sent to a priority arbitration circuit in the processor.
- If the interrupt request has a higher priority level than the priority of the processor, then the request is accepted.

| | | |
|---|---|---|
| 7 | a | Briefly discuss the main phases involved in the operation of SCSI bus. |
| | b | List the SCSI bus signals with their functionalities. |
| | c | List the functions of an I/O interface. |

**SCSI (Small Computer System Interface):**






Steps for Read operation using SCSI controller

1) The SCSI controller contends for control of the bus (initiator).
2) When the initiator wins the arbitration-process, the initiator
   → selects the target controller and
   → hands over control of the bus to it.
3) The target starts an output operation. The initiator sends a command specifying the required read-operation.
4) The target
   → sends a message to initiator indicating that it will temporarily suspend connection b/w them.
   → then releases the bus.
5) The target controller sends a command to the disk drive to move the read head to the first sector involved in the requested read-operation.
6. The target
   → transfers the contents of the data buffer to the initiator and
   → then suspends the connection again.
7) The target controller sends a command to the disk drive to perform another seek operation.
8) As the initiator controller receives the data, it stores them into the main-memory using the DMA approach.
9) The SCSI controller sends an interrupt to the processor indicating that the data are now available.

| Category | Name | Function |
|---|---|---|
| Data | – DB(0) to – DB(7) | Data lines: Carry one byte of information during the information transfer phase and iden tify device during arbitration, selection and reselection phases |
| | – DB(P) | Parity bit for the data bus |
| Phase | – BSY | Busy: Asserted when the bus is not free |
| | –SEL | Selection: Asserted during selection and reselection |
| Information type | – C/D | Cortrol/Data: Asserted during transfer of control information (command, status or message) |
| | – MSG | Message: indicates that the information being transferred is a message |

c.

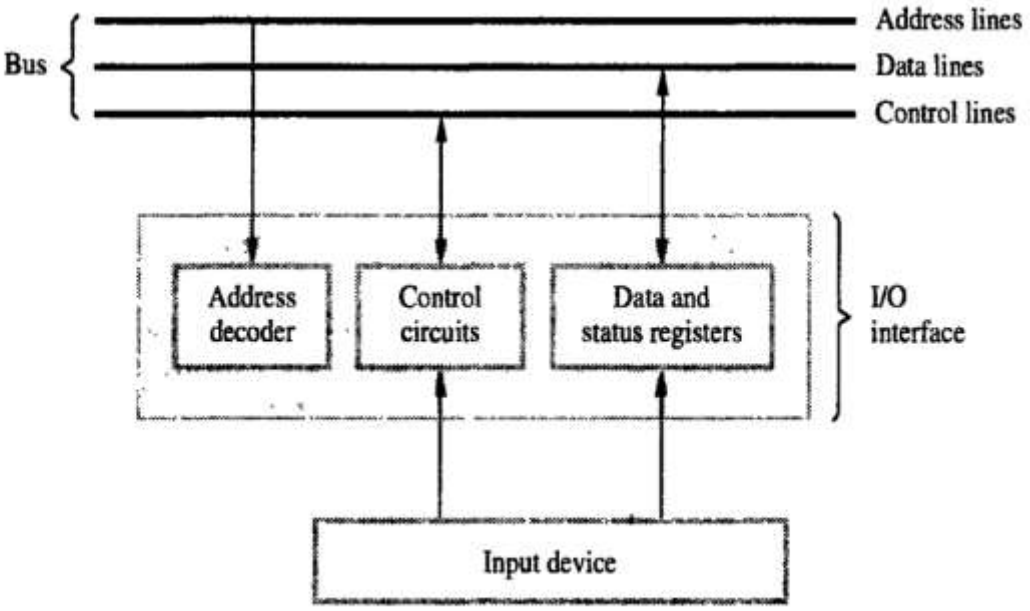
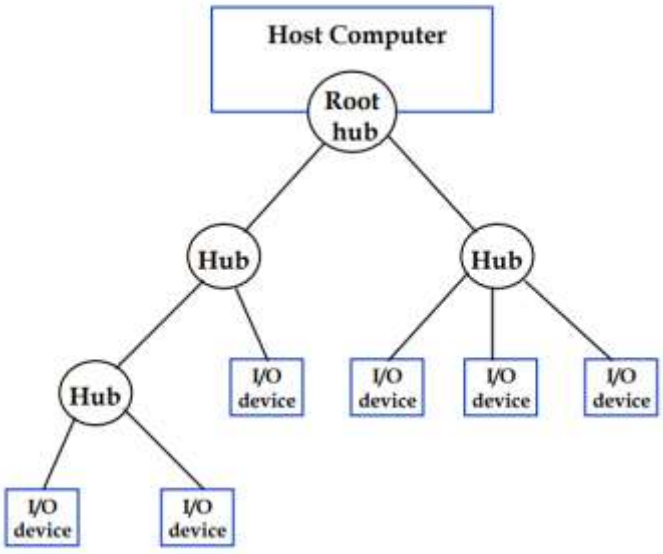



**Figure 4.2** I/O interface for an input device.

8.a Explain the following with respect to USB:
i) USB Architecture
ii) USB Addressing
iii) USB Protocols.



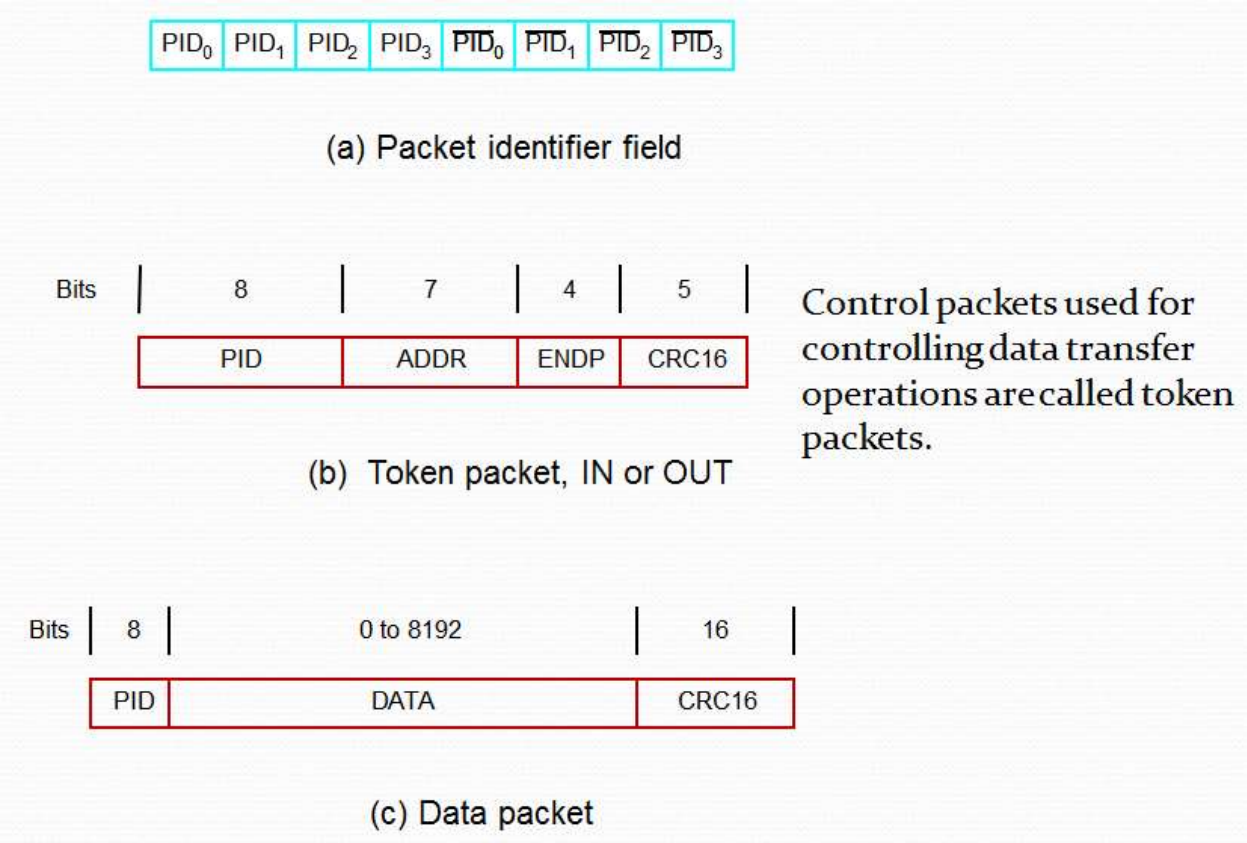
**Addressing:**

- Each device may be a hub or an I/O device.
- Each device on the USB is assigned a 7-bit address.
- This address
  - → is local to the USB tree and
  - → is not related in any way to the addresses used on the processor-bus.
- A hub may have any number of devices or other hubs connected to it, and addresses are assigned arbitrarily.
- When a device is first connected to a hub, or when it is powered-on, it has the address 0.
- The hardware of the hub detects the device that has been connected, and it records this fact as part of its own status information.
- Periodically, the host polls each hub to
  - → collect status information and
  - → learn about new devices that may have been added or disconnected.
- When the host is informed that a new device has been connected, it uses sequence of commands to
  - → send a reset signal on the corresponding hub port.
  - → read information from the device about its capabilities.
  - → send configuration information to the device, and
  - → assign the device a unique USB address.
- Once this sequence is completed, the device
  - → begins normal operation and
  - → responds only to the new address.

**USB Protocols**

- a packet consists of one or more bytes of information. There are many types of packets that perform a variety of control functions.
- The information transferred on the USB can be divided into two broad categories: control and data.
  - Control packets perform such tasks as addressing a device to initiate data transfer, acknowledging that data have been received correctly, or indicating an error.
  - Data packets carry information that is delivered to a device.
- A packet consists of one or more fields containing different kinds of information. The first field of any packet is called the packet identifier, PID, which identifies the type of that packet.
- They are transmitted twice. The first time they are sent with their true values, and the second time with each bit complemented
- The four PID bits identify one of 16 different packet types. Some control packets, such as ACK (Acknowledge), consist only of the PID byte.

**USB Packet Format**




. Discuss in detail about the following bus types with timing diagram.

i) Synchronous Bus
ii) Asynchronous Bus

**Solution:**

- A bus protocol is the set of rules that govern the behavior of various devices connected to the bus as to when to place information on the bus, assert control signals, and so on.

8.b Explain Synchronous Bus and Asynchronous Bus with neat Timing diagrams.

8

## Buses

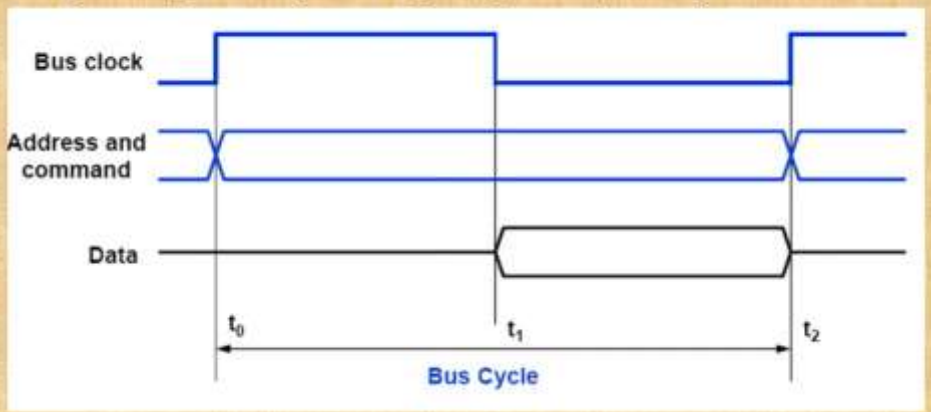
- ✓ The **bus** lines used for **transferring information** is grouped into 3 types. They are,
  - Address line
  - Data line
  - Control line.
- ✓ Control signals: Specifies read / write operation has to perform.
- ✓ Also carries *timing info.*: req. for ***synchronization***.
- ✓ During data transfer: one device plays the role of a Master
- ✓ ***Master*** device initiates the data transfer: Initiator.
- ✓ The device addressed by the master is called as ***Slave / Target***.

## Buses

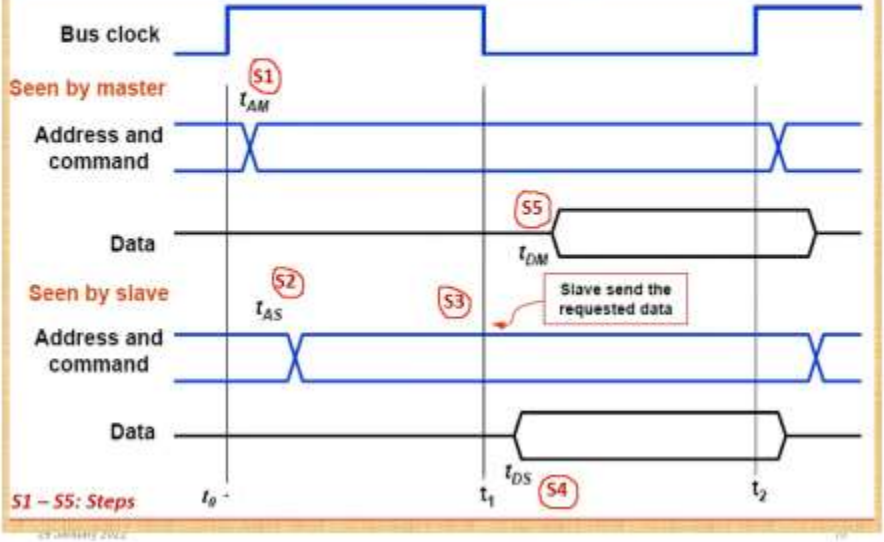
- ✓ A ***bus protocol***: governs the ***behavior*** of various devices connected to the bus.
- ✓ **2 types of buses (also called as Bus Protocols).**
  1. **Synchronous Bus**
  2. **Asynchronous Bus.**

# Synchronous Buses

- ✓ All devices derive timing information from a **common clock line**.
- ✓ Equally spaced pulses on clock line define **equal time**.
- ✓ During a "bus cycle", one data transfer take place: *Theoretically*
- ✓ Fig: Timing of an ***input transfer (operation)*** of a synchronous bus.

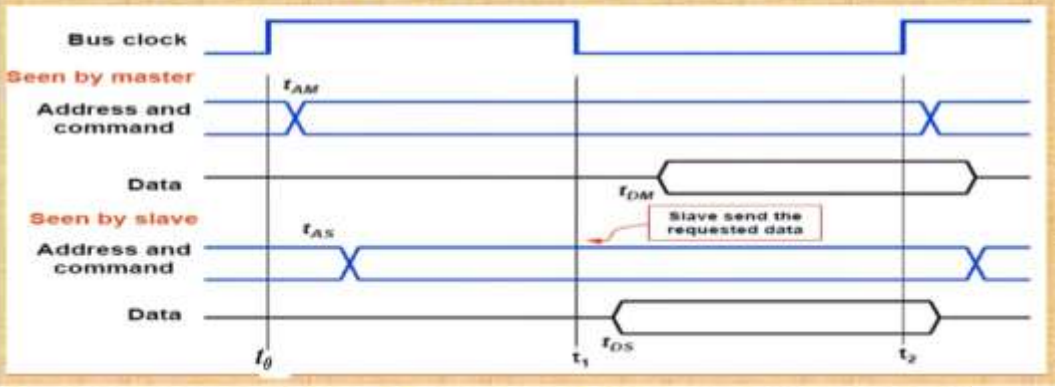



## *Detailed timing diagram for the input transfer*: SB



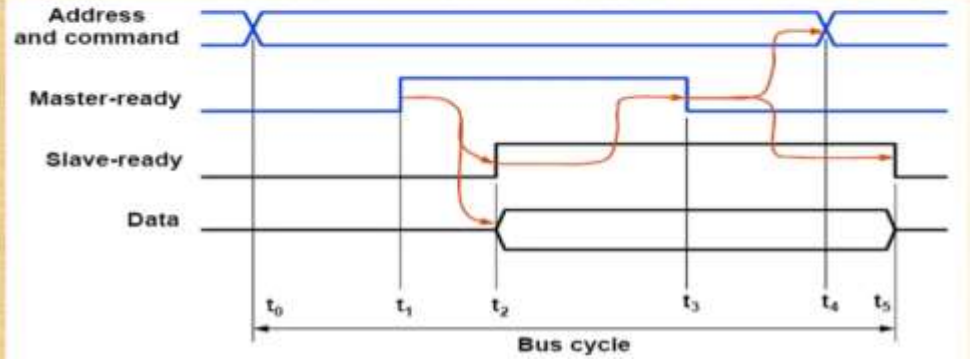
## Detailed timing diagram for the input transfer: SB

- ✓ TD: two views of the signal except the clock.: Master & Salve.
- ✓ The master sends the address & cmd. signals: at $t_0$.
  - But actually appear on the bus at $t_{AM}$.
- ✓ At $t_{AS}$ the signals reach the ***slave***.
- ✓ Slave *decodes* address at $t_1$ & *sends* the requested *data* at $t_{DS}$.
- ✓ Master sees data: $t_{DM}$ & *strobes* the data into its i/p buffer at $t_2$.

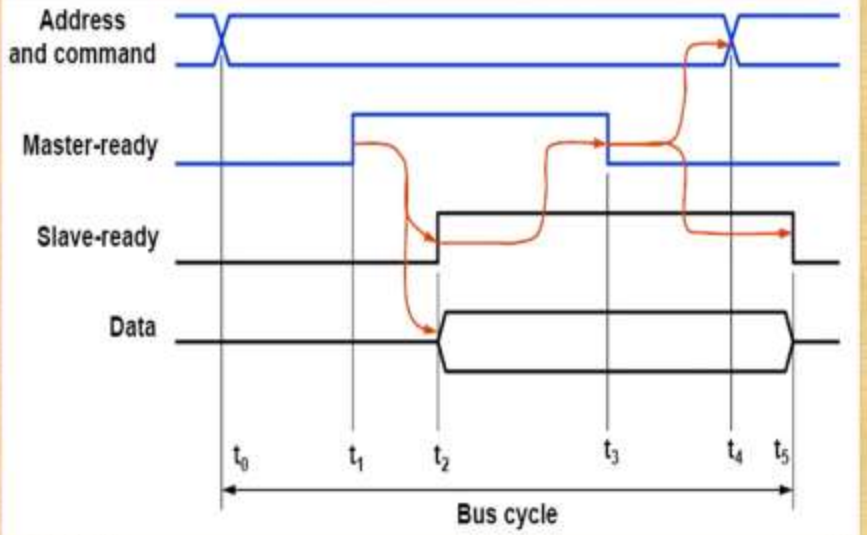



## Asynchronous Buses

- ✓ Alternate scheme for **controlling data transfer on the bus**.
- ✓ The bus uses of "**handshake**" between **Master** & the **Slave**.
- ✓ The common clock is replaced by two timing control lines.
  - – Master–ready
  - – Slave-ready.
- Fig: Handshake control of data transfer during an **input operation**



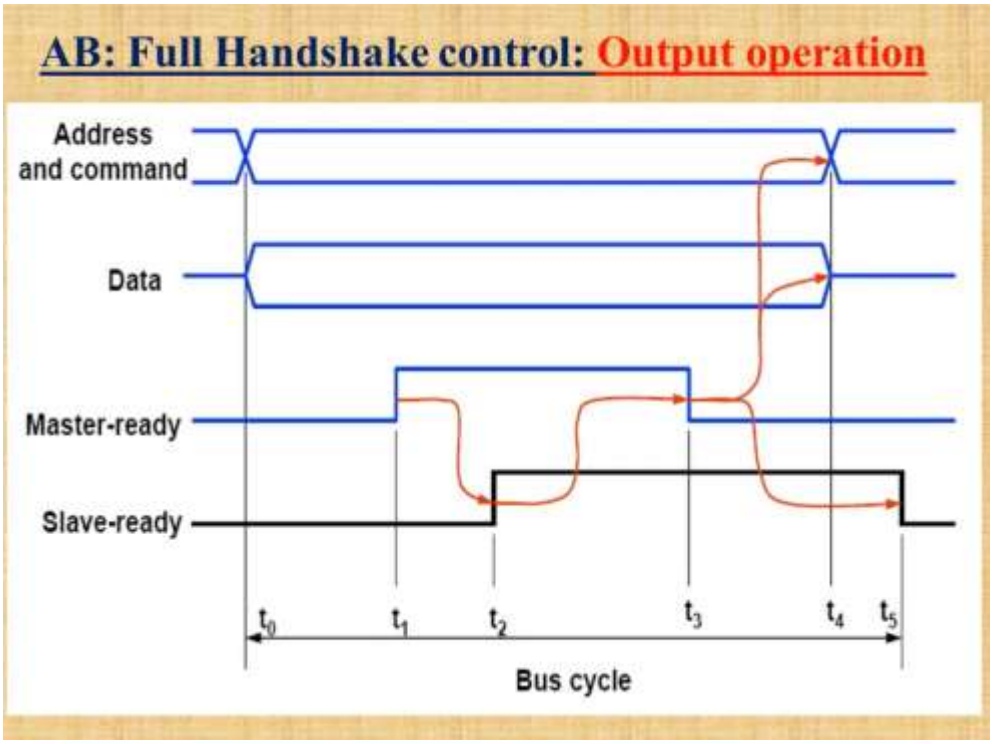
## AB: Full Handshake control: Input operation






## AB: Full Handshake control: Input operation

- ✓ t0: The master places the address and command information on the bus.
- ✓ t1: The master sets the Master-ready line=1: inform the I/O devices.
  - The delay t1 – t0 is intended to allow for any *skew* on the bus.
- ✓ t2: Slave puts the data on the data lines & sets Slave-ready=1.
- ✓ t3: Slave-ready signal arrives at the master: it strobes the data & remove the Master–ready signal.
- ✓ t4: The master removes the address and command information on the bus.
- ✓ t5: It removes the data and the Slave–ready signal from the bus.
- ✓ This completes the i/p transfer.




## AB: Full Handshake control: Output operation

- ✓ At t0 the master: o/p data (DL)+slave address (AL)+ctrl info (CL)
- ✓ At t1 the master raises the Master-ready signal.
- ✓ Then the selected slave strobes the data to its i/p buffer and it indicates this by setting the Slave – ready signal to 1.
- ✓ This completes the o/p transfer

- ✓ A change of state is one signal is followed by a change is the other signal.
- ✓ Hence this scheme is called as **Full Handshake**.
- ✓ It provides the higher degree of flexibility and reliability.

9.A

| | |
|---|---|
| a | With a neat sketch, explain the organization of a microprogrammed control unit. |

| | |
|---|---|
| b | With a neat block diagram, explain hardwired control unit. Show the generation Zin and end control signals. |

a.

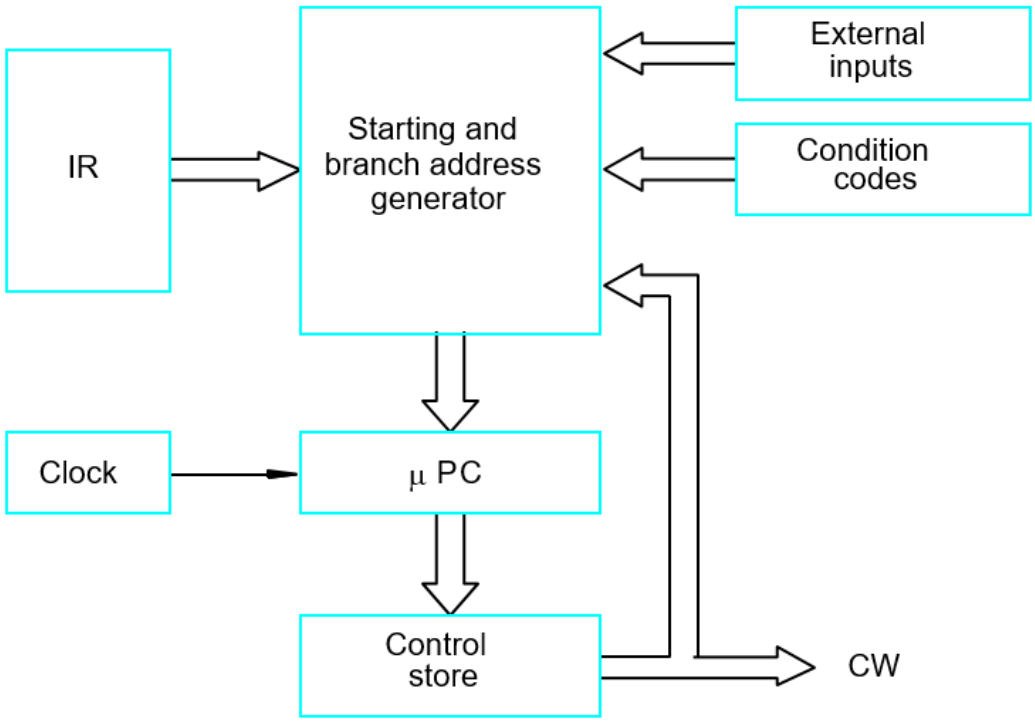
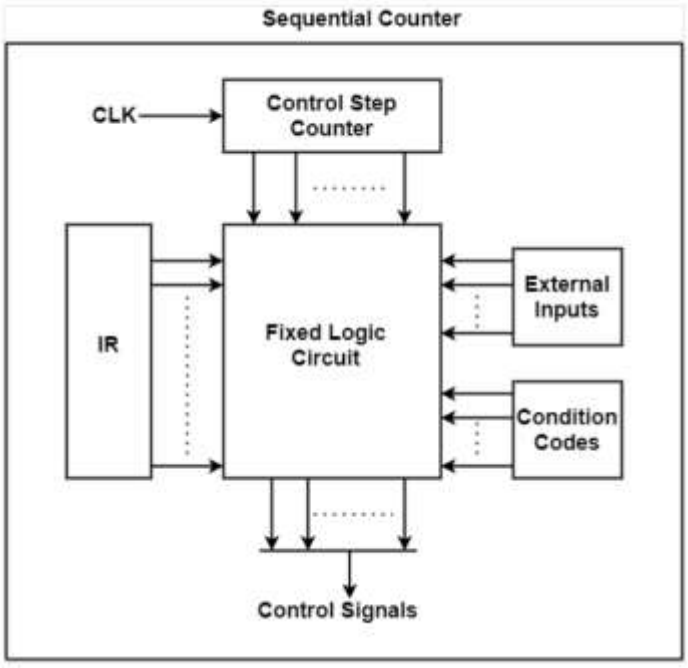


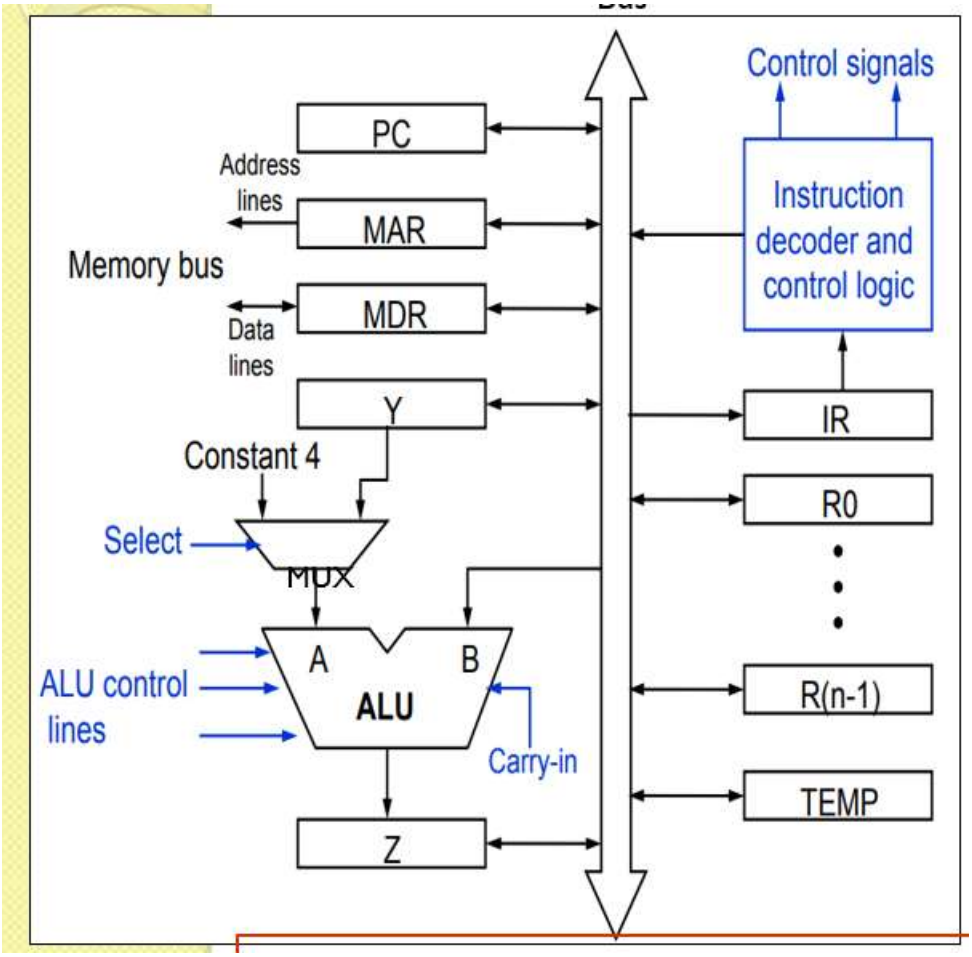

Figure 7.18. Organization of the control unit to allow conditional branching in the microprogram.

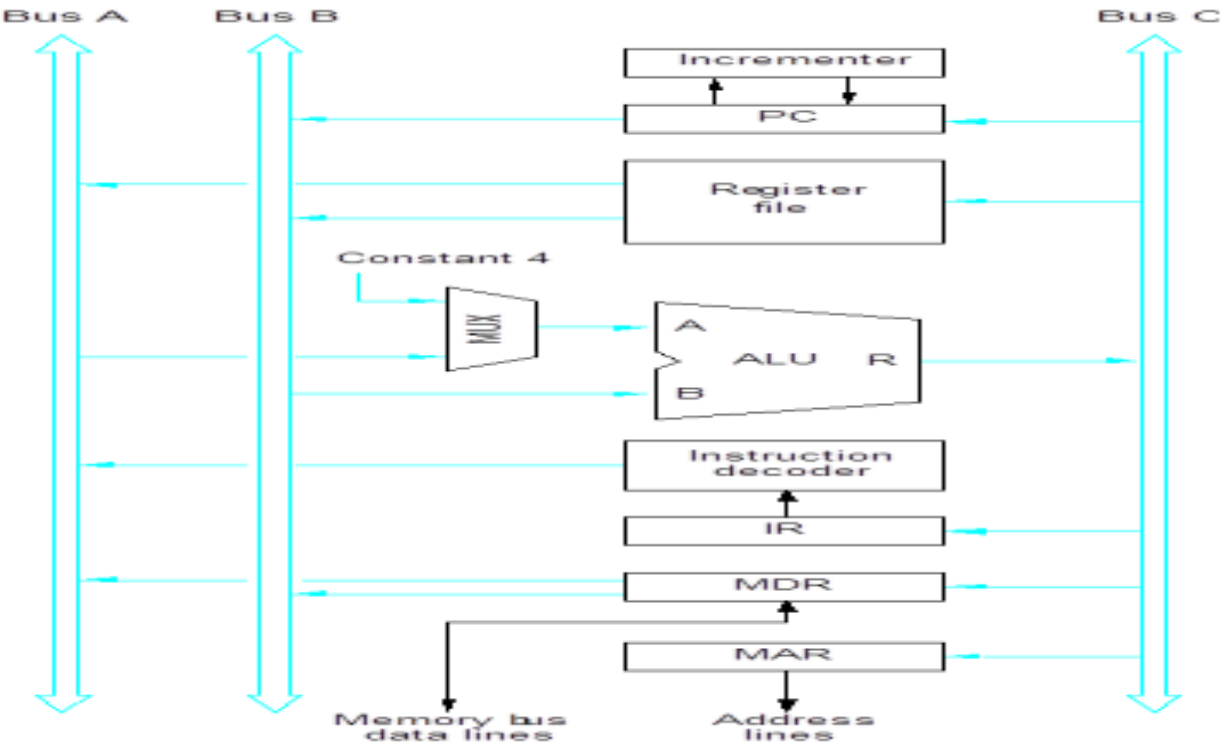
b.A hardwired control is a mechanism of producing control signals using Finite State Machines (FSM) appropriately. It is designed as a sequential logic circuit. The final circuit is constructed by physically connecting the components such as gates, flip flops, and drums. Hence, it is named a hardwired controller.

The figure shows a 2-bit sequence counter, which is used to develop control signals. The output obtained from these signals is decoded to generate the required signals in sequential order.


Sequential Counter
CLK
Control Step Counter
IR
Fixed Logic Circuit
External Inputs
Condition Codes
Control Signals

10.a


Control signals
PC
Address lines
MAR
Memory bus
MDR
Data lines
Y
Constant 4
Select
MUX
A
B
ALU
ALU control lines
Carry-in
Z
Instruction decoder and control logic
IR
R0
R(n-1)
TEMP


Bus A
Bus B
Bus C
Incrementer
PC
Register file
Constant 4
MUX
A
ALU
R
B
Instruction decoder
IR
MDR
MAR
Memory bus data lines
Address lines

**10.b Execution of complete instruction.**

Add (R3), R1

Adds the content of memory location pointed to by R3 to registers R1.

Executing this instruction requires following actions:

1. Fetch the instruction
2. Fetch the first operand (the contents of the memory location pointed to by R3)
3. Perform the addition
4. Load the result into R1

Add (R3), R1

Instruction fetch

| Step | Action |
|---|---|
| 1 | $PC_{out}$, $MAR_{in}$, Read, Select4 Add, $Z_{in}$ |
| 2 | $Z_{out}$, $PC_{in}$, $Y_{in}$, WMFC |
| 3 | $MDR_{out}$, $IR_{in}$ |
| 4 | $R3_{out}$, $MAR_{in}$, Read |
| 5 | $R1_{out}$, $Y_{in}$, WMFC |
| 6 | $MDR_{out}$, SelectY, Add, $Z_{in}$ |
| 7 | $Z_{out}$, $R1_{in}$, End |