Computer Organization and Architecture

UNIT 2- System Buses

Dr.Sumita Nainan, MPSTME, NMIMS University

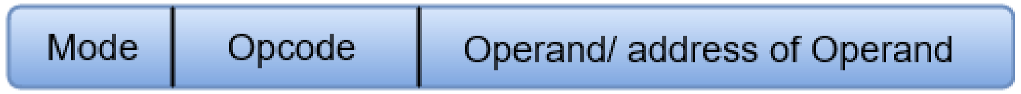
Syllabus-Unit 2-System Buses

- Overview of basic instruction cycle
- Interrupts
- Bus interconnection
- Elements of bus design
- Read and write timings diagram
- Bus hierarchy
- Bus arbitration techniques.

Computer Instructions

- Computer instructions are a set of machine language instructions that a particular processor understands and executes. A computer performs tasks on the basis of the instruction provided.

- An instruction comprises of groups called fields. These fields include:



- The Mode field which specifies how the operand will be located.

- The Operation code (Opcode) field which specifies the operation to be performed.

- The Address field which contains the location of the operand, i.e., register or memory location.
- e.g. MOV A, B

A basic computer has three instruction code formats which are:

1.Memory - reference instruction
2.Register - reference instruction
3.Input-Output instruction

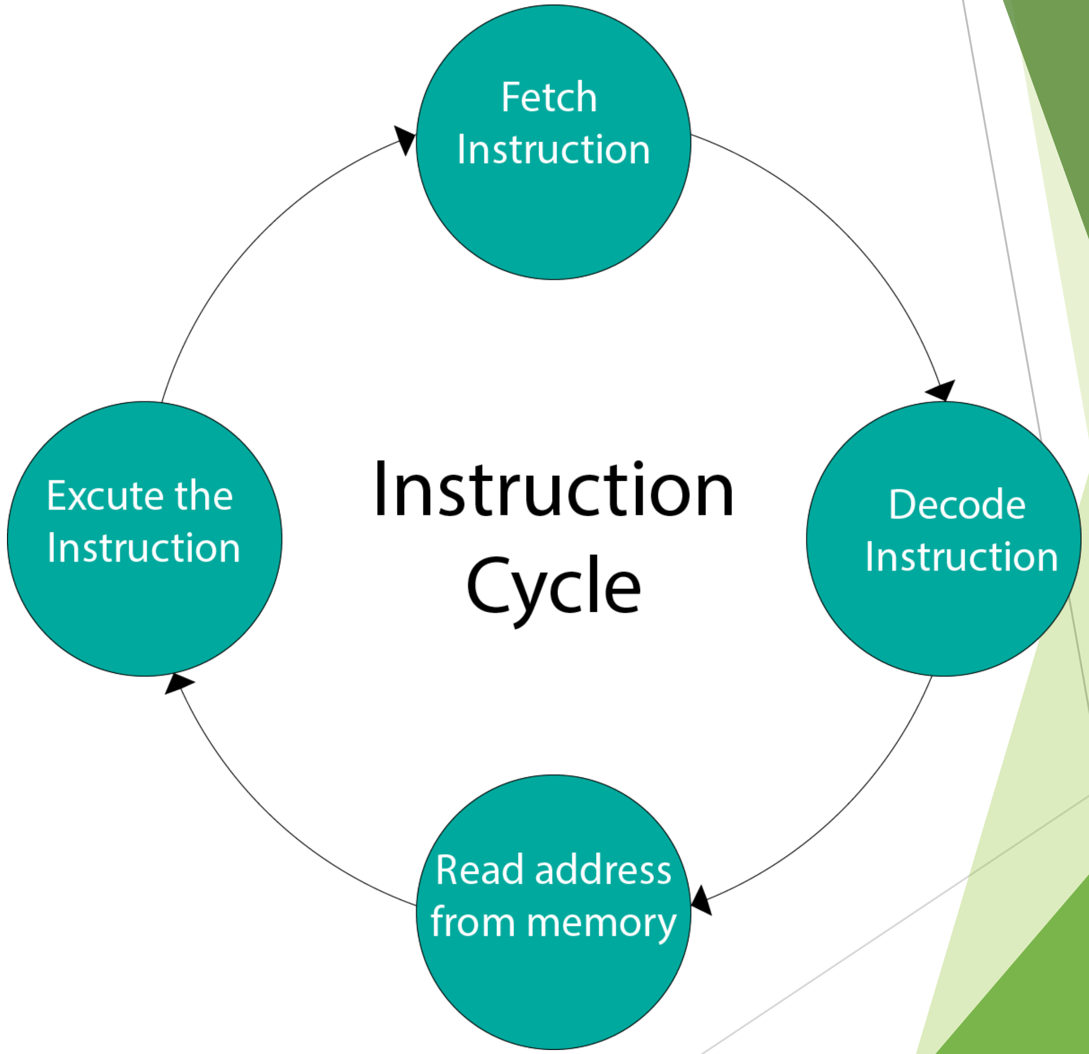
Overview of basic instruction cycle

A program residing in the memory unit of a computer consists of a sequence of instructions.

These instructions are executed by the processor by going through a cycle for each instruction.

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

1.Fetch instruction from memory.
2.Decode the instruction.
3.Read the effective address from memory.
4.Execute the instruction.



Overview of basic instruction cycle

Registers Involved In Each Instruction Cycle:

1. **Memory address registers(MAR)** : It is connected to System Bus address lines. It specifies the address of a read or write operation in memory.
2. **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.
3. **Program Counter(PC)** : Holds the address of the next instruction to be fetched.
4. **Instruction Register(IR)** : Holds the last instruction fetched.

Overview of basic instruction cycle

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

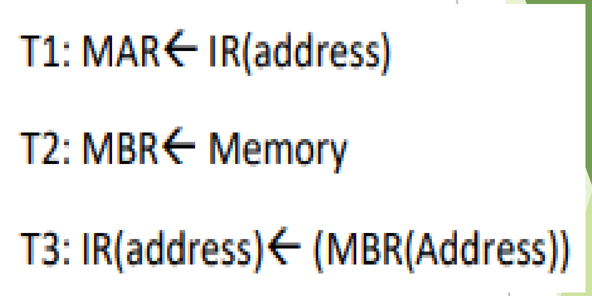
PC← PC + stepsize or length of instruction

T3: IR← MBR

Overview of basic instruction cycle

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.
- 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.



T1: MAR← IR(address)

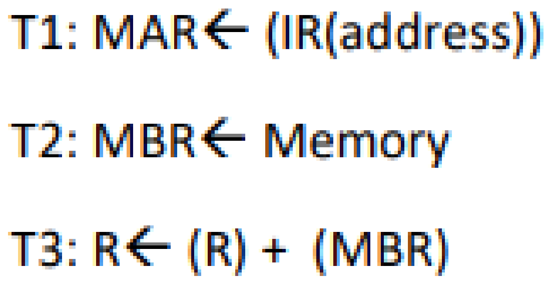
T2: MBR← Memory

T3: IR(address)← (MBR(Address))

Overview of basic instruction cycle

Execute instruction Cycle

- The initial cycles 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.
- 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.



T1: MAR$\leftarrow$ (IR(address))

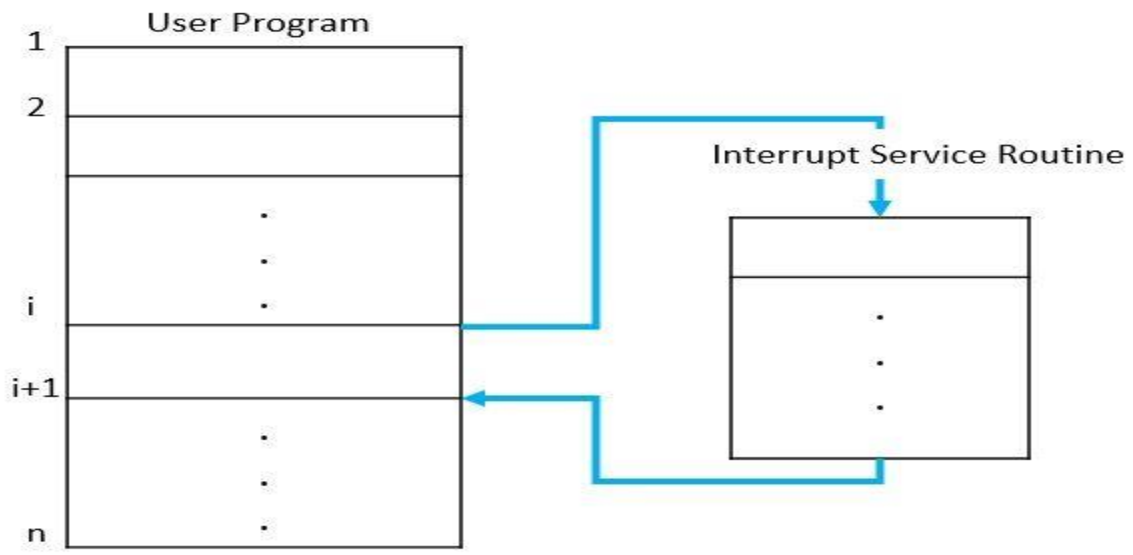
T2: MBR$\leftarrow$ Memory

T3: R$\leftarrow$ (R) + (MBR)

Interrupts

Execute instruction Cycle

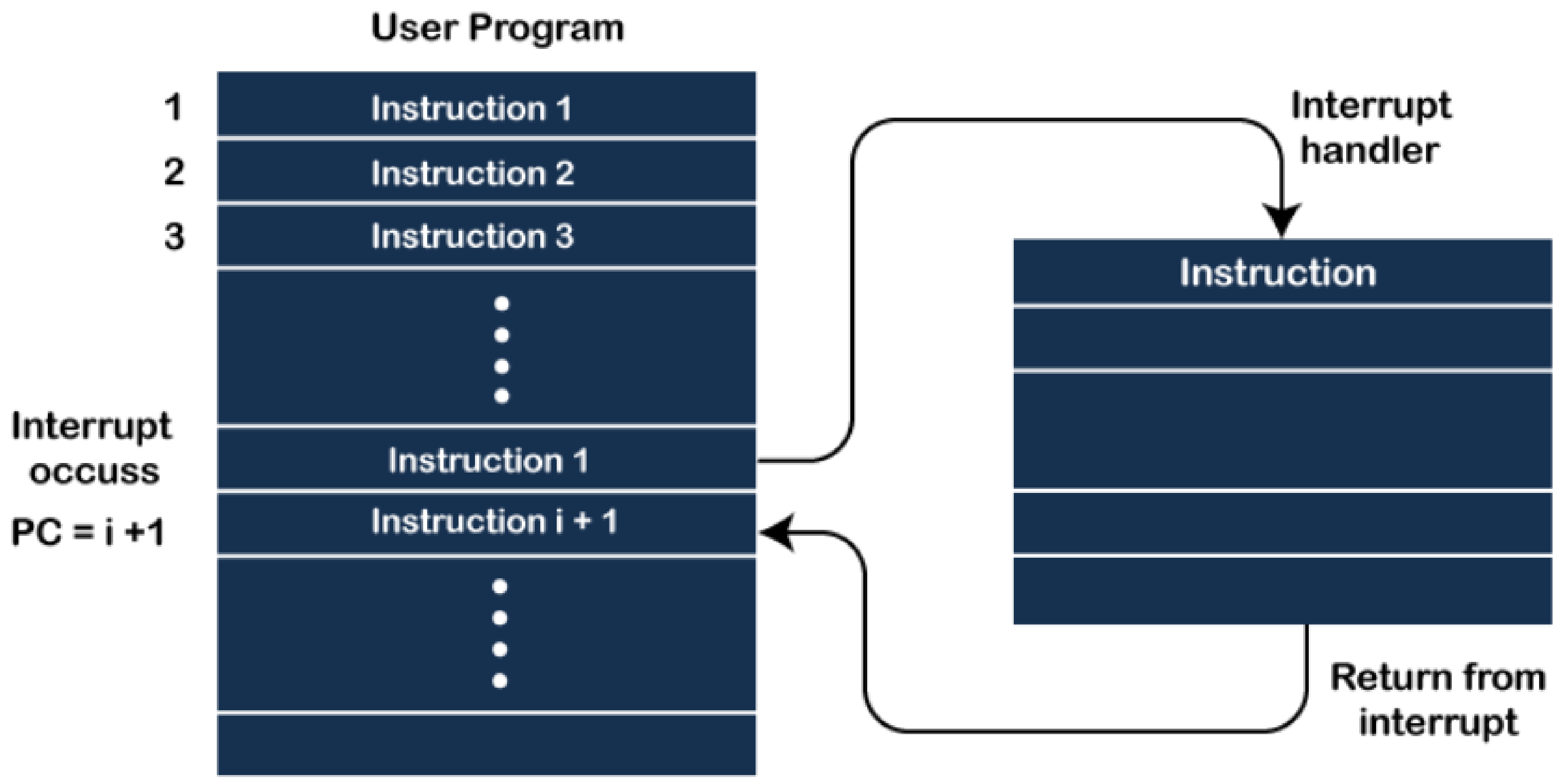
An interrupt in computer architecture is a signal that requests the processor to suspend its current execution and service the occurred interrupt. To service the interrupt the processor executes the corresponding interrupt service routine (ISR). After the execution of the interrupt service routine, the processor resumes the execution of the suspended program. Interrupts can be of two types of hardware interrupts and software interrupts.



Transfer Control Via Interrupts

Interrupts

- An interrupt is **a signal from a device attached to a computer or from a program within the computer that requires the operating system to stop and figure out what to do next**.



Interrupts…

- Interrupts can be **classified** as
- 1. **Internal Interrupts** OR
- 2. **External Interrupts**
- Internal Interrupts are initiated by the CPU or by an Instruction while Ext. Int. are caused by a signal being sent to the CPU from somewhere in the system.
- e.g. Divide by zero(Internal Int.)
- From an I/O device for data Transfer(ext. int.)

Interrupts…

- The action that executes one of the special routines whenever certain conditions arise from within the program or from the computer system – is called an **INTERRUPT.**
- The program executed thereafter(Routine) is called the ISR- Interrupt Service Routine
- **Types** Of Interrupts
- 1. Hardware Interrupt.
- 2. Software Interrupt

Procedure followed….

- POLLING
- INTERRUPTS
- INTERRUPT SERVICE ROUTINR(ISR)

Interrupt Method

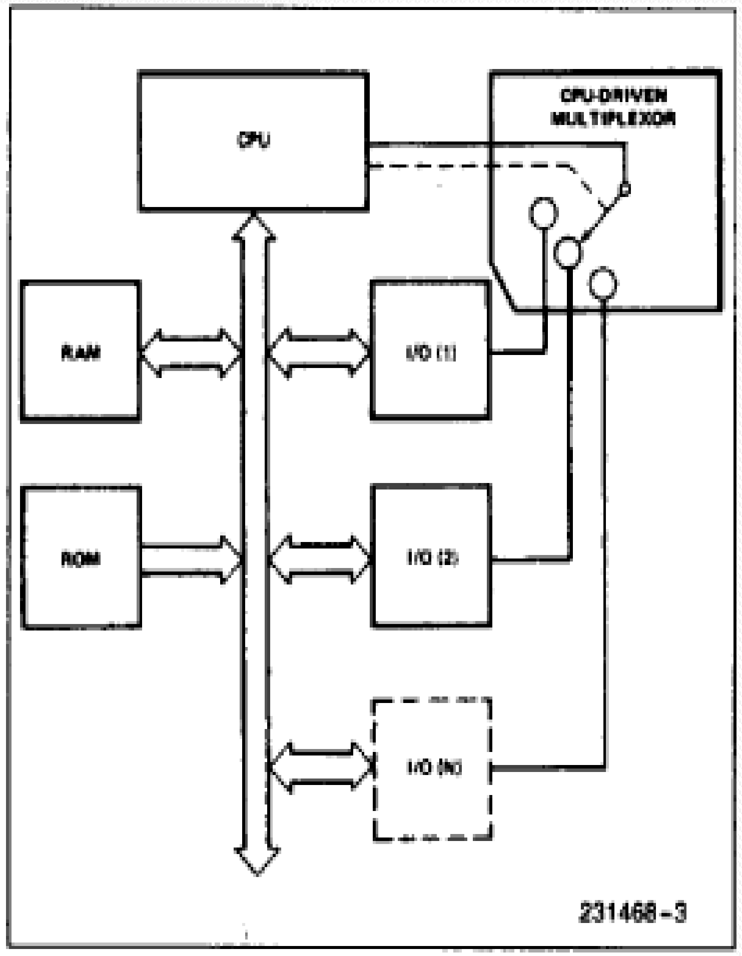


Figure 3a. Polled Method

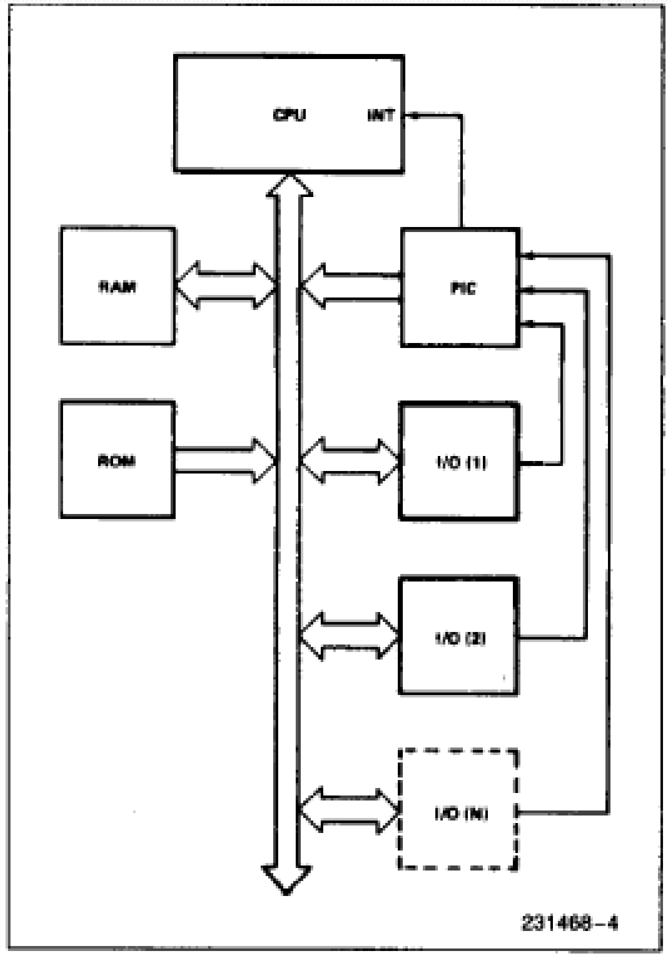
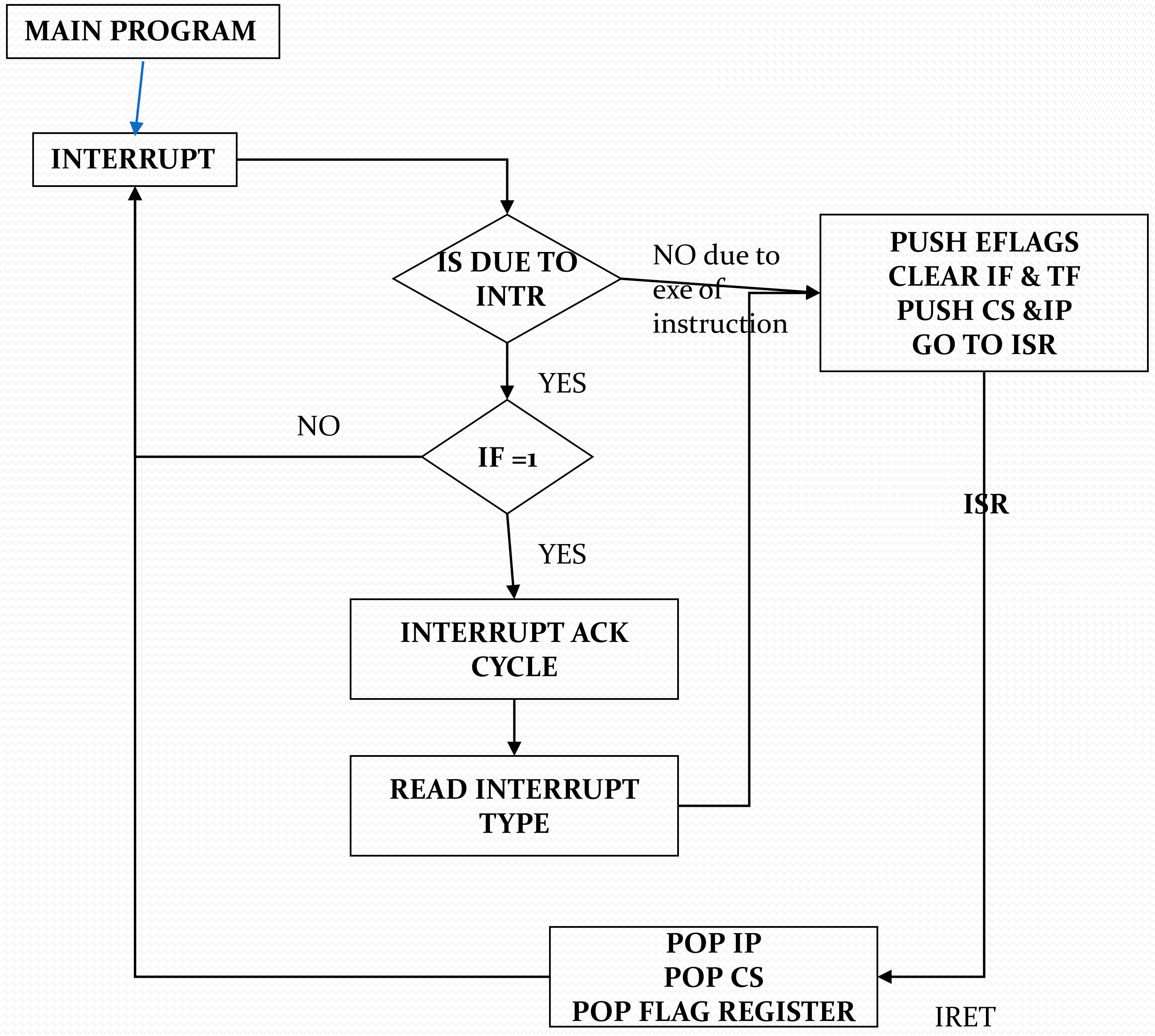


Figure 3b. Interrupt Method

FLOW CHART


MAIN PROGRAM
INTERRUPT
IS DUE TO INTR
NO due to exe of instruction
PUSH EFLAGS
CLEAR IF & TF
PUSH CS &IP
GO TO ISR
YES
NO
IF =1
YES
INTERRUPT ACK CYCLE
READ INTERRUPT TYPE
ISR
POP IP
POP CS
POP FLAG REGISTER
IRET

Bus Architecture- Interconnection

- **What is a Bus?**
- Computers comprises of many internal components and in order for these components to communicate with each other, a ‘bus’ is required for that purpose.
- *A bus is a common pathway through which information flows from one component to another.* This pathway is used for communication purpose and can be established between two or more computer components.
- It is a method of transmitting data from one part of the computer to another part of the computer.
- Buses are the basic transportation lines for moving data, instructions, addresses, and other information inside a computer.
- Generally the computer bus will connect all devices to the computer CPU and main memory.The word bus comes from the Latin adjective omnibus, meaning all or everyone.
- A bus is a set (group) of parallel lines on which information (data, addresses, instructions, and other information) travels on inside a computer.
- Information travels on buses as a series of electrical pulses, each pulse representing a one bit or a zero bit (there are trinary, or three-state, buses, but they are rare).
- Buses are the links between components of a computer system, and they are often the performance bottleneck in the system

Bus Architecture- Interconnection

Functions of Buses in Computers

1. **Data sharing –**

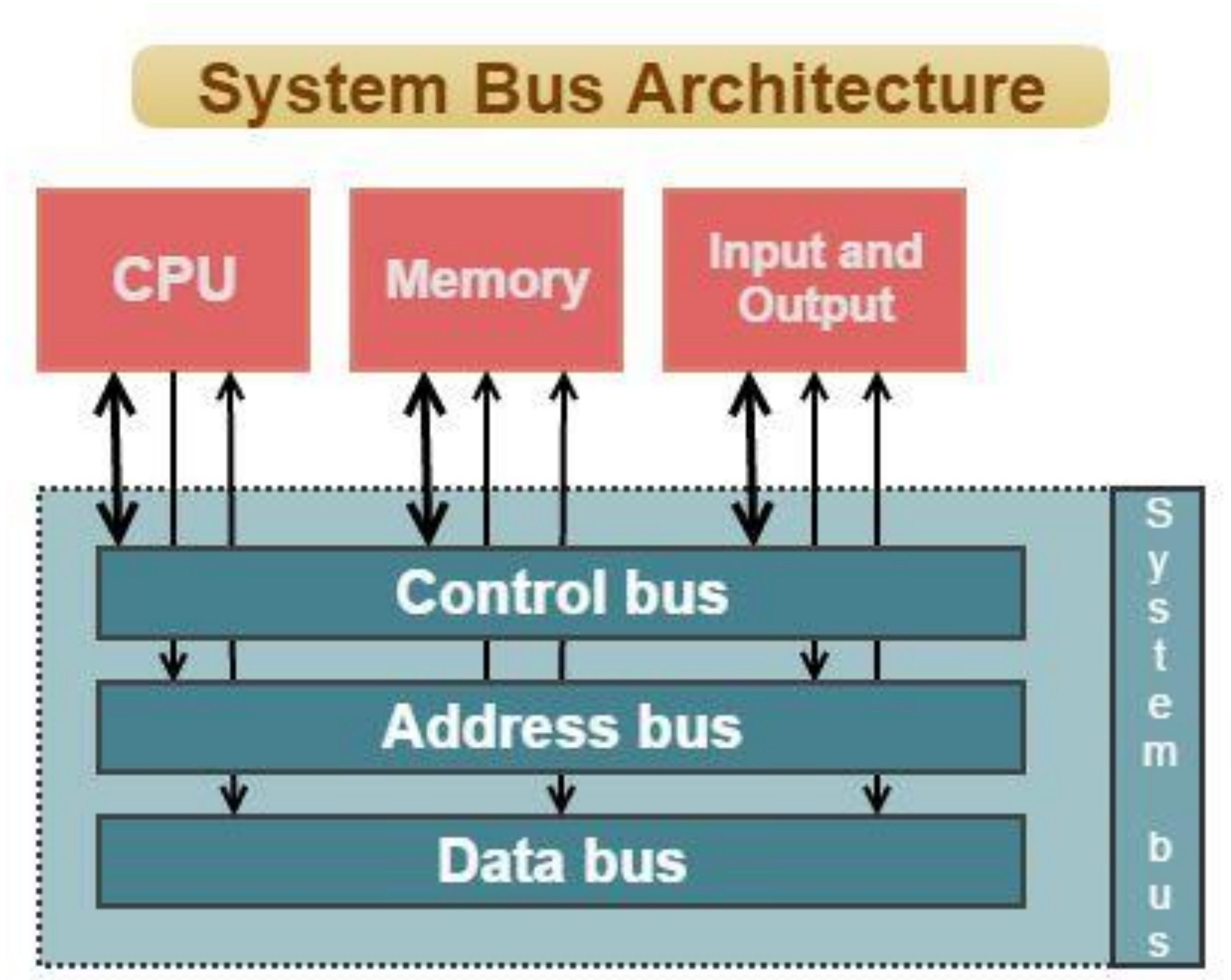
- All types of buses found on a computer must be able to transfer data between the computer peripherals connected to it.
- The data is transferred in in either serial or parallel, which allows the exchange of 1, 2, 4 or even 8 bytes of data at a time. (A byte is a group of 8 bits).
- Buses are classified depending on how many bits they can move at the same time, which means that we have 8-bit, 16-bit, 32-bit or even 64-bit buses.

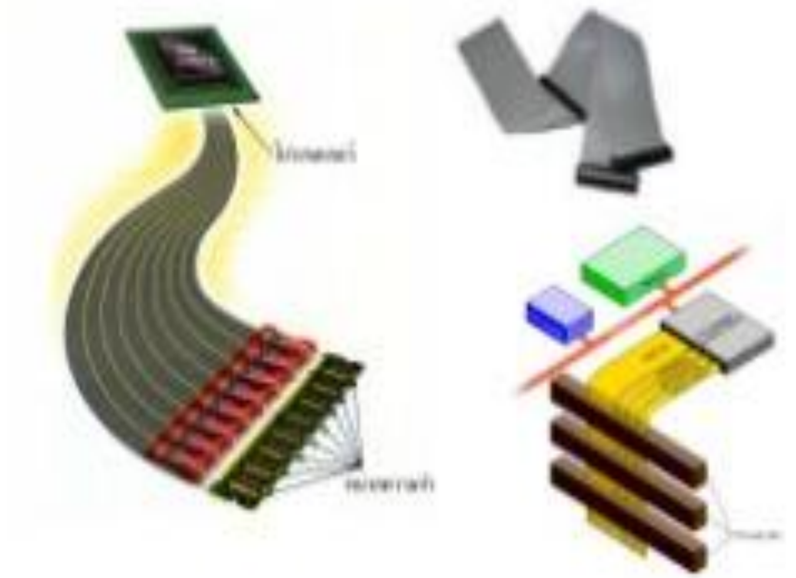
- 2. **Addressing –**
- A bus has address lines, which match those of the processor.
- This allows data to be sent to or from specific memory locations.

- 3. **Power –**
- A bus supplies power to various peripherals that are connected to it.

- 4. **Timing –**
- The bus provides a system clock signal to synchronize the peripherals attached to it with the rest of the system.
- The expansion bus facilitates the easy connection of additional components and devices on a computer for example the addition of a TV card or sound card.

Bus Architecture- Interconnection



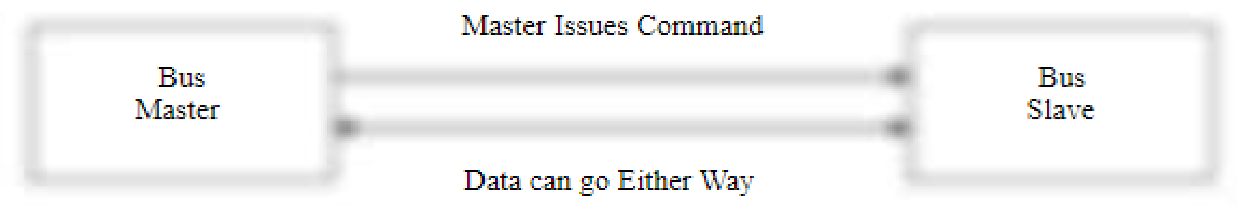


The General Organization of a Bus

- **Control Lines**
- Signal requests and acknowledgements.
- Indicate what type of information is on the data lines.

- **Data Lines**
- Carry information between the source and the destination:
- 1.Data and addresses 2.Complex commands

The General Organization of a Bus- Master Slave Configuration



Master Versus Slave Representation

A bus transaction includes two parts:
1. Issuing the command (and address) - request
2. Transferring the data - action

Master is the one who starts the bus transaction by:

- Issuing the command (and address)
- Slave is the one who responds to the address by:
- Sending data to the master if the master ask for data
- Receiving data from the master if the master wants to send data

Elements of bus design

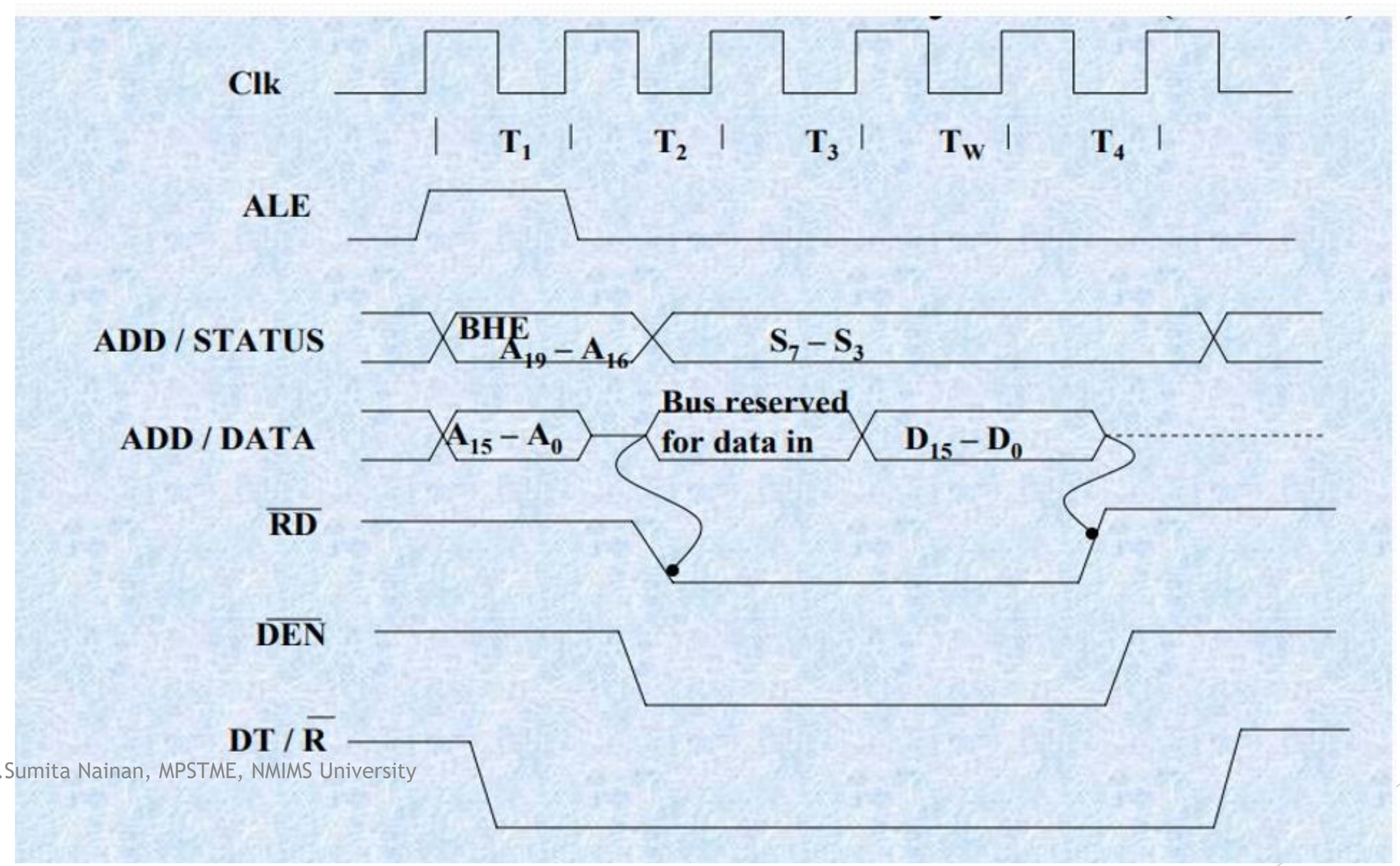
- **Advantages of Buses**
- New devices can be added easily.
- Peripherals can be moved between computer systems that use the same bus standard.
- A single set of wires is shared in multiple ways.

- **Disadvantages of Buses**
- Bus creates a communication bottleneck because the bandwidth of the bus can limit the maximum I/O throughput.
- The maximum bus speed is largely limited by
- 1.The length of the bus
- 2.The number of devices on the bus
- 3.The need to support a range of devices with widely varying latencies

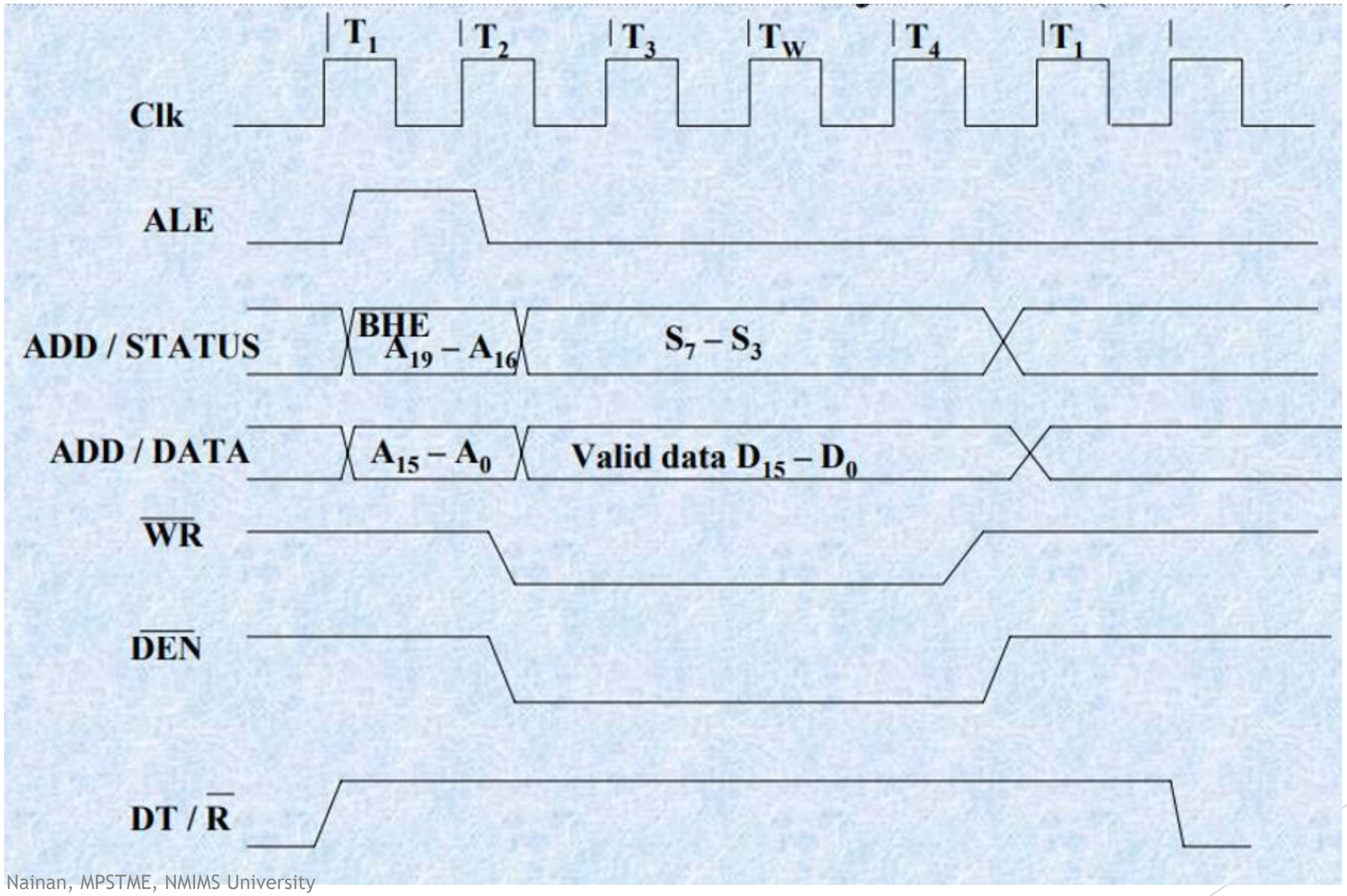
- Widely varying data transfer rates

Timings diagram-Read

Timing Diagram is the **graphical representation of process in steps with respect to time**. The timing diagram represents the clock cycle and duration, delay, content of address bus and data bus, type of operation ie. Read/write/status signals



Timings diagram -Write



BUS InterConnection

1. **Address Bus:**

- Address bus carry the memory address while reading from or writing into the memory.
- Address bus carry I/O post address or device address from I/O port.
- In uni-directional address bus only the CPU could send address and other units could not address the microprocessor.
- Now a days computers have bi-directional address bus.

2. **Data Bus:**

- Data bus carry the data.
- Data bus is a bidirectional bus.
- Data bus fetch the instructions from memory.
- Data bus used to store the result of an instruction into memory.
- Data bus carry commands to an I/O device controller or port.
- Data bus carry data from a device controller or port.
- Data bus issue data to a device controller or port.

BUS InterConnection

3. Control Bus:

- Different types of control signals are used in a bus:
- **Memory Read:** This signal, is issued by the CPU or DMA controller when performing a read operation with the memory.
- **Memory Write:** This signal is issued by the CPU or DMA controller when performing a write operation with the memory.
- I/O Read: This signal is issued by the CPU when it is reading from an input port.
- I/O Write: This signal is issued by the CPU when writing into an output port.
- Ready: The ready is an input signal to the CPU generated in order to synchronize the show memory or I/O ports with the fast CPU.
- A **system bus** is a single computer bus that connects the major components of a computer system, combining the functions of a data bus to carry information, an address bus to determine where it should be sent, and a control bus to determine its operation.

BUS InterConnection

3. Control Bus:

- The **control lines** are used to control the access to and the use of the data and address lines.
- Control signals transmit both command and timing information among system modules.
- Timing signals indicate the validity of data and address information. Command signals specify operations to be performed.
- Typical control lines include
- **Memory write**: Causes data on the bus to be written into the addressed location
- **Memory read**: Causes data from the addressed location to be placed on the bus
- **I/O write**: Causes data on the bus to be output to the addressed I/O port
- **I/O read**: Causes data from the addressed I/O port to be placed on the bus

BUS InterConnection

3. Control Bus:

- **Transfer ACK:** Indicates that data have been accepted from or placed on the bus
- **Bus request**: Indicates that a module needs to gain control of the bus
- **Bus grant**: Indicates that a requesting module has been granted control of the bus
- **Interrupt request**: Indicates that an interrupt is pending
- **Interrupt ACK**: Acknowledges that the pending interrupt has been recognized
- **Clock**: Is used to synchronize operations
- **Reset**: Initializes all modules

The operation of the bus is as follows. If one module wishes to send data to another, it must do two things: (1) obtain the use of the bus, and (2) transfer data via the bus. If one module wishes to request data from another module, it must (1) obtain the use of the bus, and (2) transfer a request to the other module over the appropriate control and address lines. It must then wait for that second module to send the data.

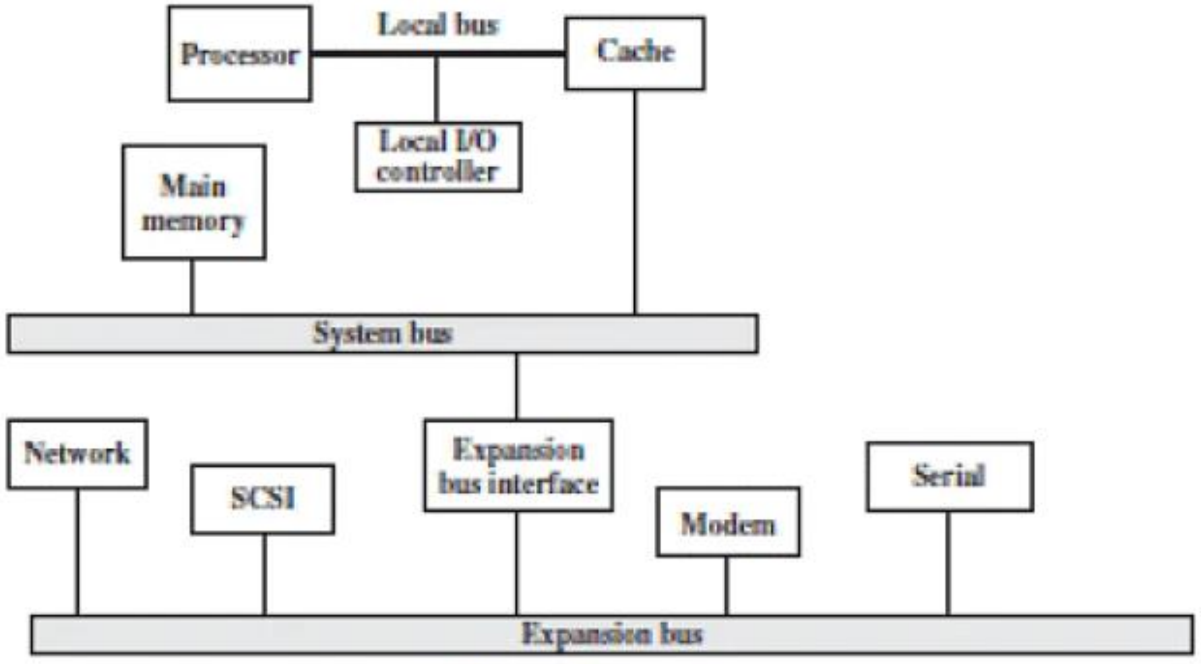
BUS InterConnection

Multiple-Bus Hierarchies

- If a great number of devices are connected to the bus, performance will suffer.
- There are two main causes:

1. In general, the more devices attached to the bus, the greater the bus length and hence the greater the propagation delay.
2. The bus may become a bottleneck as the aggregate data transfer demand approaches the capacity of the bus.

- Most computer systems use multiple buses, generally laid out in a hierarchy.
- A typical traditional structure is shown in Figure.



(a) Traditional bus architecture

BUS InterConnection

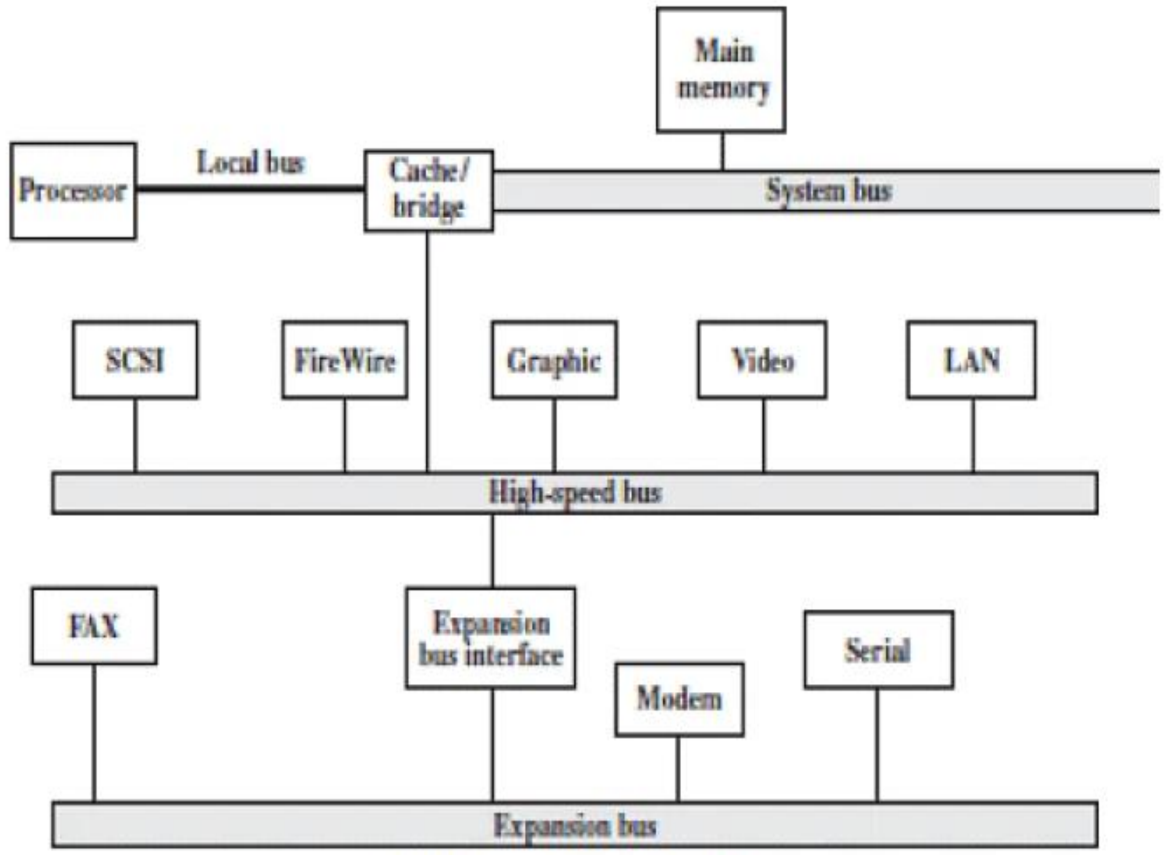
Multiple-Bus Hierarchies

- There is a local bus that connects the processor to a cache memory and that may support one or more local devices
- The cache memory is connected to a system bus to which the main memory modules are attached.
- It is possible to connect I/O controllers directly onto the system bus.
- A more efficient solution is to make use of one or more expansion buses for this purpose.
- This arrangement allows the system to support a wide variety of I/O devices and at the same time insulate memory-to-processor traffic from I/O traffic.

BUS Inter Connection

Multiple-Bus Hierarchies

- Network connections include local area networks (LANs), wide area networks (WANs), SCSI (small computer system interface), serial port.
- This traditional bus architecture is reasonably efficient but begins to break down as higher and higher performance is seen in the I/O devices.
- In response to these growing demands, a common approach taken by industry is to build a high-speed bus that is closely integrated with the rest of the system, requiring only a bridge between the processor's bus and the high-speed bus.
- This arrangement is sometimes known as a mezzanine architecture.
- Figure shows a typical realization of this approach

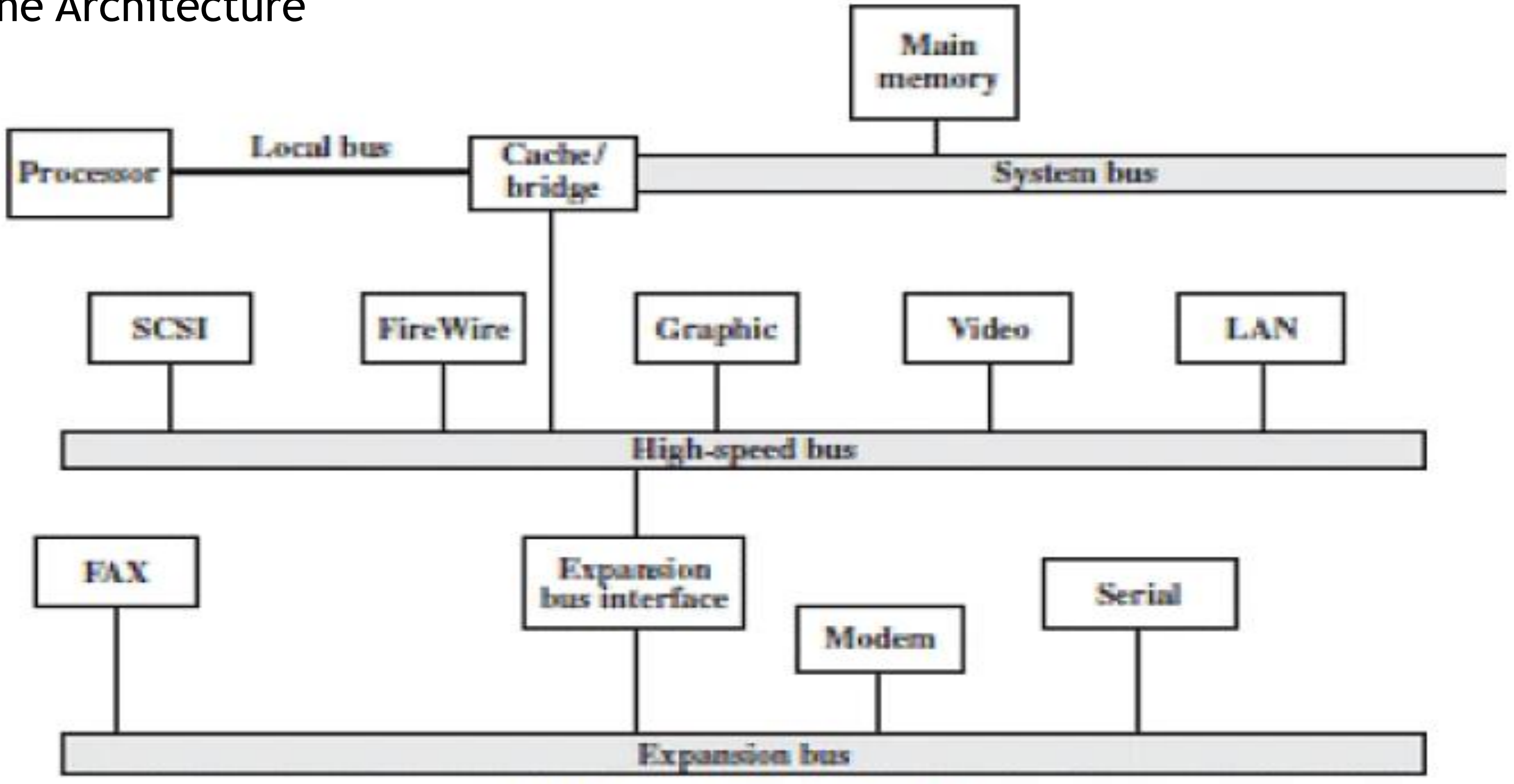


(b) High-performance architecture

https://www.embedded.com/high-speed-bus-architectures-bring-new-challenges/

BUS Inter-Connection

Mezzanine Architecture



(b) High-performance architecture

BUS Inter-Connection

Multiple-Bus Hierarchies

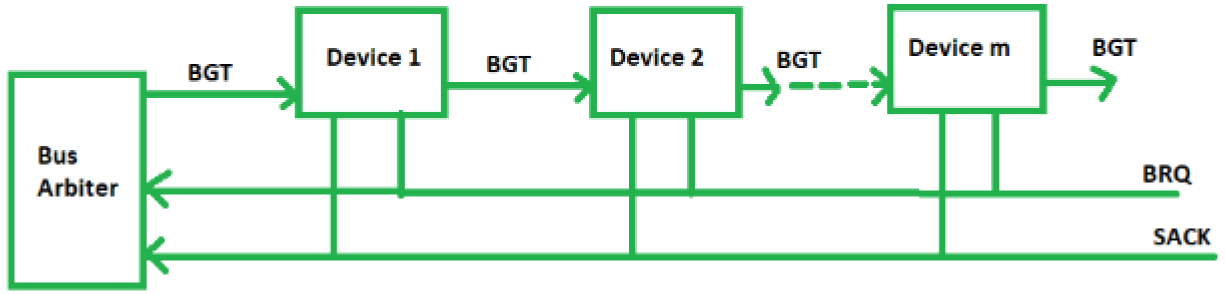
- Again, there is a local bus that connects the processor to a cache controller, which is in turn connected to a system bus that supports main memory.
- The cache controller is integrated into a bridge, or buffering device, that connects to the high-speed bus.
- This bus supports connections to high-speed LANs, video and graphics workstation controllers, SCSI and Fire Wire Lower-speed devices are still supported off an expansion bus, with an interface buffering traffic between the expansion bus and the high-speed bus.
- The advantage of this arrangement is that the high-speed bus brings high-demand devices into closer integration with the processor and at the same time is independent of the processor.

Bus Arbitration

- **Bus Arbitration** refers to the process by which the current bus master accesses and then leaves the control of the bus and passes it to the another bus requesting processor unit.
- The controller that has access to a bus at an instance is known as **Bus master**.
- A conflict may arise if the number of DMA controllers or other controllers or processors try to access the common bus at the same time, but access can be given to only one of those.
- Only one processor or controller can be Bus master at the same point of time.
- To resolve these conflicts, Bus Arbitration procedure is implemented to coordinate the activities of all devices requesting memory transfers.
- The selection of the bus master must take into account the needs of various devices by establishing a priority system for gaining access to the bus.
- The **Bus Arbiter** decides who would become current bus master.

Bus Arbitration cont..

- There are two approaches to bus arbitration:
- **Centralized bus arbitration –** A single bus arbiter performs the required arbitration.
- **Distributed bus arbitration –** All devices participate in the selection of the next bus master.
- **Methods of BUS Arbitration –**
  There are **three** bus arbitration methods:
- **(i) Daisy Chaining method –**
  It is a centralized bus arbitration method. During any bus cycle, the bus master may be any device – the processor or any DMA controller unit, connected to the bus.



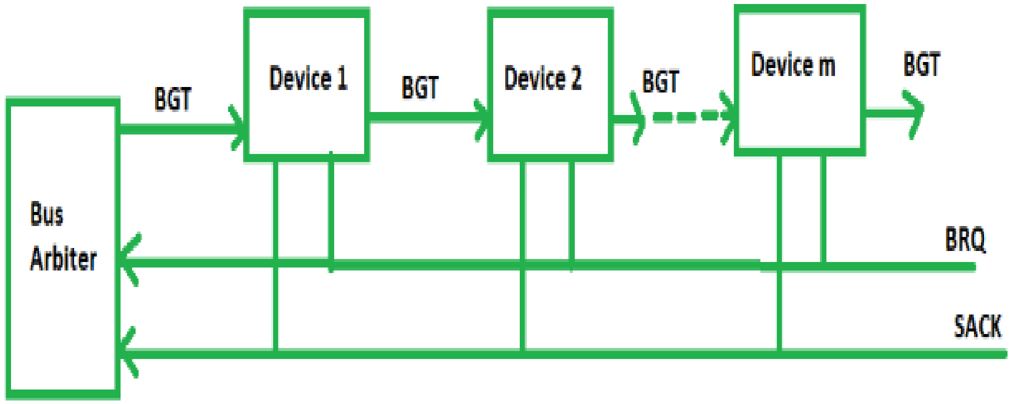
Daisy chained bus arbitration

Bus Arbitration cont..

1. **Daisy Chaining method**

Advantages –
•Simplicity and Scalability.
•The user can add more devices anywhere along the chain, up to a certain maximum value.

Disadvantages –
•The value of priority assigned to a device is dependent on the position of master bus.
•Propagation delay is arises in this method.
•If one device fails then entire system will stop working.

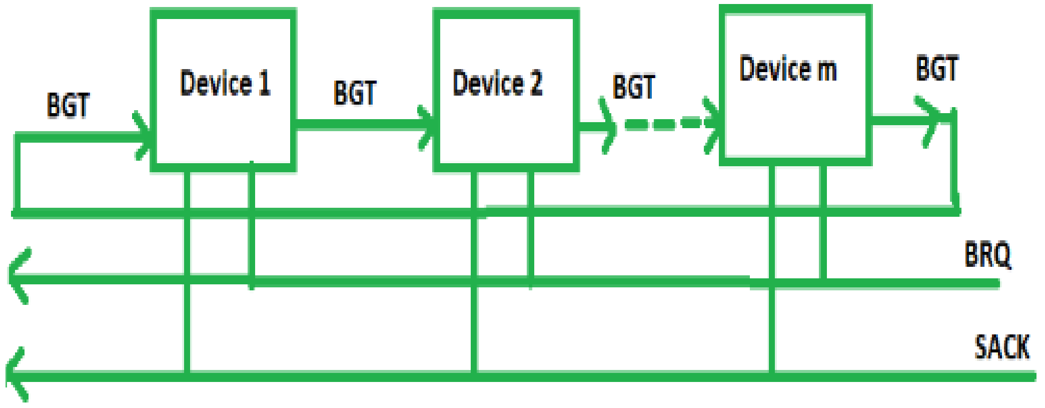


Daisy chained bus arbitration

Bus Arbitration cont..

(ii) Polling or Rotating Priority method –

In this method, the devices are assigned unique priorities and complete to access the bus, but the priorities are dynamically changed to give every device an opportunity to access the bus.



Rotating priority bus arbitration

Advantages –

•This method does not favor any particular device and processor.
•The method is also quite simple.
•If one device fails then entire system will not stop working.

Disadvantages –

•Adding bus masters is different as increases the number of address lines of the circuit

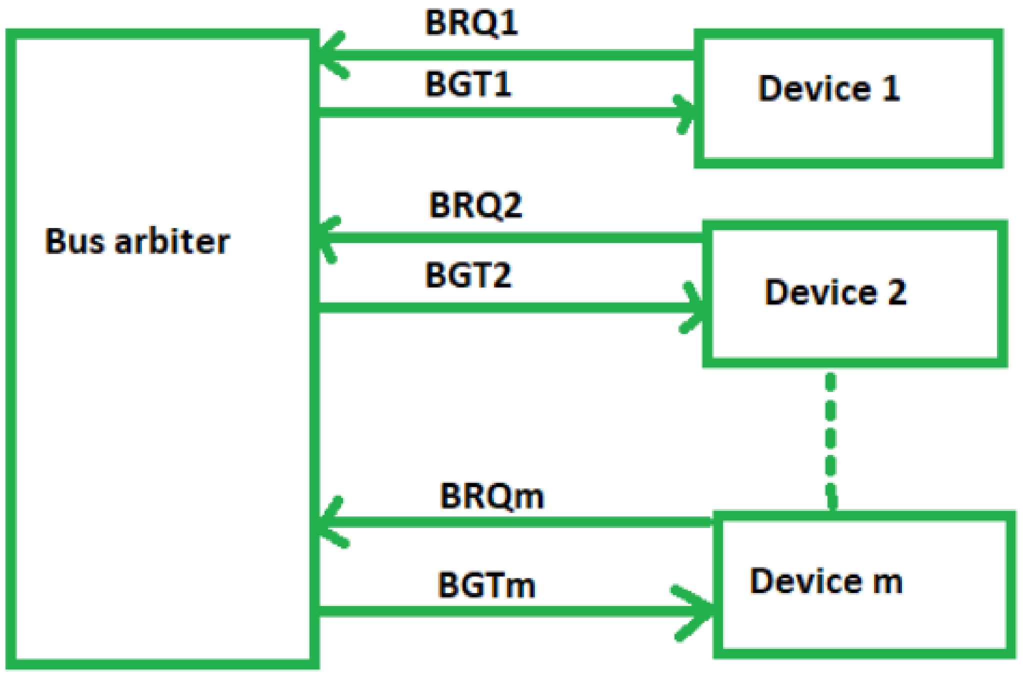
Bus Arbitration cont..

(iii) Fixed priority or Independent Request method –

In this method, the bus control passes from one device to another only through the centralized bus arbiter

Advantages –
This method generates fast response.

Disadvantages –
Hardware cost is high as large no. of control lines are required.



Fixed priority bus arbitration method