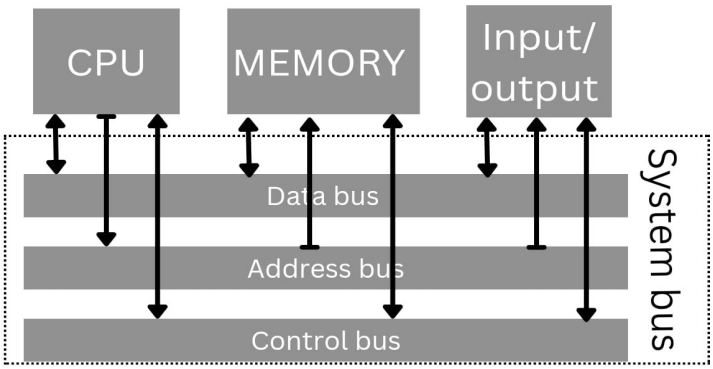
Author : Abhay Kumar Singh

COMPUTER ORGANIZATION AND ARCHITECTURE

UNIT - 1

System Bus



- System Bus Facilitates data and control signal transfer among computer components, serving as a communication highway for information exchange.
- The system bus consists of three main types of buses:

1. Address Bus

- **Purpose**: Carries memory addresses, with each signal combination representing a unique memory location.
- **Size**: Defines the maximum addressable memory; e.g., a 32-bit address bus can access 2^32 (4 gigabytes) of memory.
- **Direction**: Unidirectional, from CPU to memory or I/O devices.

2. Data Bus

- **Purpose**: Transmits data between the CPU, memory, and peripherals, carrying both instructions and processed data.
- **Size**: Specifies the parallel data transmission capacity; common sizes include 8-bit, 16-bit, 32-bit, or 64-bit data buses.
- **Direction**: Bidirectional, allowing data transfer in both directions.

3. Control Bus

- **Purpose**: Transmits control signals for coordinating and managing activities among connected components, encompassing read/write signals, interrupt requests, and clock signals.
- **Signals**: Includes common signals like Read (RD), Write (WR), Clock (CLK), Interrupt Request (IRQ), and others.
- **Direction**: Bidirectional, though some signals may be unidirectional.

Single Shared Bus

- All components (CPU, memory, peripherals) share a single communication pathway (bus).
- **Advantage**: Simplicity in design and lower cost.
- **Disadvantage**: Potential for congestion and slower performance as all data must use the same pathway.

Multiple Shared Buses

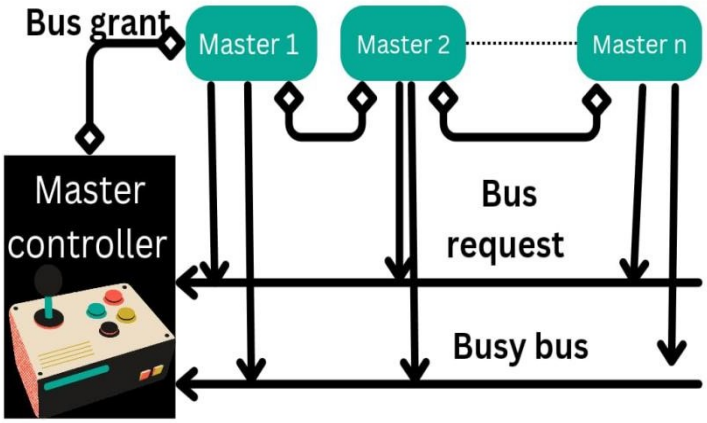
- Different components use separate communication pathways (buses).
- **Advantage**: Reduced congestion, faster communication between specific components.
- **Disadvantage**: Increased complexity in design and higher cost due to multiple buses.

Bus arbitration

- Bus arbitration is a mechanism which decides the selection of current master to access bus.
- Multiple master or slave units connected to a shared bus may concurrently request access to the bus.
- In such situation, bus access is given to the master having highest priority.
- Three different mechanisms are commonly used for this:

1. Daisy chaining method

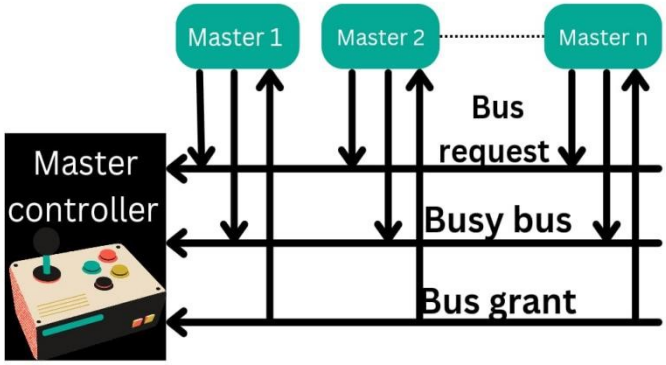
- all masters make use of the same line for bus request.
- The bus grant signal serially propagates through each master until it encounters the first one that is requesting access to the bus.
- This master blocks the propagation of the bus grant signal, activates the busy line and gains control of the bus.
- Therefore any other requesting module will not receive the grant signal and hence cannot get the bus access.
- During any bus cycle, the bus master may be any device – the processor or any DMA controller unit, connected to the bus.



- **Avantages**
  - It is a simple and cheaper method.
  - It requires the least number of lines and this number is independent of the number of masters in the system.
- **Disadvantages**
  - Propagation delay arises in this method.
  - If one device fails then the entire system will stop working.

2. Polling method

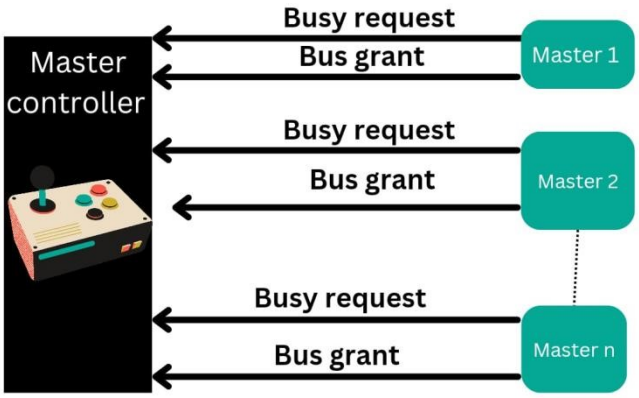
- The polling method involves sequentially checking each master to determine which one is ready to access the bus.



- **Advantages :**
  - If the one module fails entire system does not fail.
  - The priority can be changed by altering the polling sequence stored in the controller.
- **Disadvantages :**
  - It requires more bus request and grant signals (2 x n signals for n modules).
  - Polling overhead can consume a lot of CPU time.

3. Independent Request or Fixed Priority Method -

- A unique pair of bus requests and bus grant lines are provided to each master, and each pair is given a priority.
- The controller's built-in priority decoder chooses the utmost priority request and then asserts the matching bus grant signal.
- **Advantage**:
  - This technique produces a quick response.
- **Disadvantage**:
  - A significant number of control lines are needed, which raises the cost of the hardware.



Memory transfer

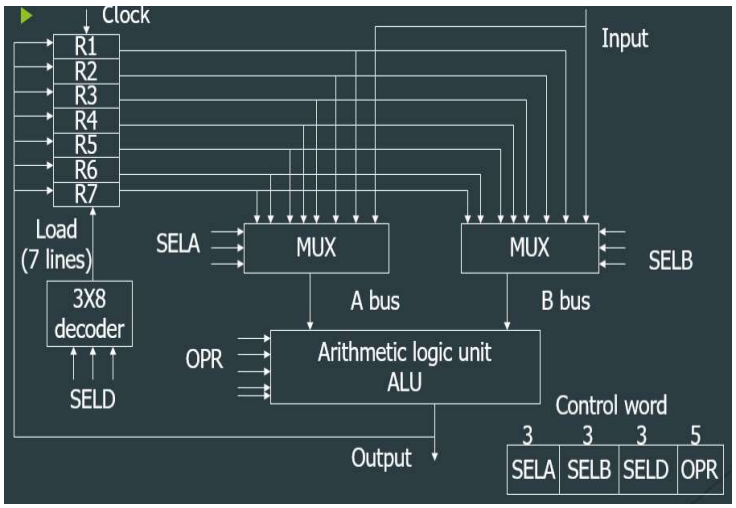
- Memory transfer involves two main operations: reading (fetching) and writing (storing) data.
- Reading transfers a copy of data from a memory location to the CPU.
- Writing transfers information from the CPU to a specific memory location, replacing the previous content.
- Two important registers are used for memory transfer:
- the Address Register (AR) and the Data Register (DR).
- T**he Address Register (AR)** holds the memory location address from which data needs to be read or where data needs to be written.
- **The Data Register (DR)** is used to temporarily store the data being transferred between memory and the CPU.
- **Read Operation:**To perform a read operation:

  DR ← M[AR]
- **Write Operation**: To perform a write operation:

  M[AR] ← DR

General Register-Based CPU Organization:

- In a general register-based CPU organization:
- Multiple general-purpose registers are used instead of a single accumulator register.
- Two or three address fields are present in the instruction format.
- Each address field specifies a general register or a memory word.
- The availability of many CPU registers enables efficient storage of heavily used variables and intermediate results.
- This organization minimizes memory references, leading to increased program execution speed and reduced program size



- **Example Instruction:**
  - Example assembly language instruction: MULT R1, R2, R3
  - MULT: Mnemonic for multiplication.
  - R1, R2, R3: Register operands.
  - R1 <-- R2 * R3: Indicates that the result of the multiplication is stored in register R1.
- **Advantages**:
  - Supports efficient data manipulation.
  - Reduces the need for frequent memory accesses.
  - Common in modern processor architectures, especially Reduced Instruction Set Computing (RISC) architectures.

Register Stack

- An ordered set of elements with a variable length.
- Accessible one element at a time from the top.
- Also known as a pushdown list or Last-In-First-Out (LIFO) list.

Registers Used in Stack Organization:

- **Stack Pointer Register (SP):**

Contains a 6-bit binary value representing the address of the top of the stack.

Limited to values from 000000 to 111111 (0 to 63).

- **FULL Register:** 1-bit register.

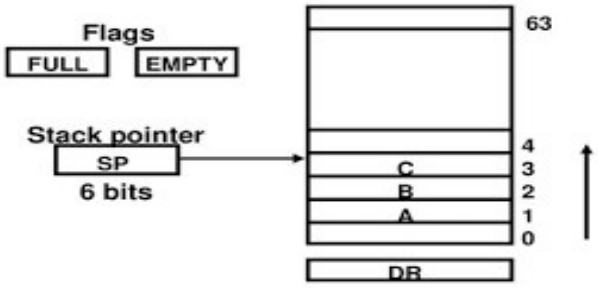
Set to 1 when the stack is full.

- **EMPTY Register:** 1-bit register.

Set to 1 when the stack is empty.

- **Data Register (DR):**

Holds data to be written into or read from the stack.



Working of POP and PUSH:

- **PUSH ():**

```
if FULL == 0:
    SP = SP + 1        // Increment stack pointer
    M[SP] = DR   // Write item Data Register to top of  stack
    if SP == 0:
        FULL = 1       // Mark the stack as full
    EMPTY = 0          // Mark the stack as not empty
```

- **POP ():**

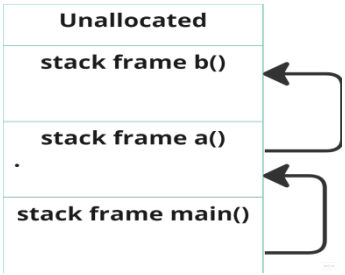
```
if EMPTY == 0:
    DR = M[SP]  //Read item from top of stack to Data Register
    SP = SP - 1        // Decrement stack pointer
    if SP == 0:
        EMPTY = 1      // Mark the stack as empty
    FULL = 0           // Mark the stack as not full
```

Memory stack

- Memory stack is a dynamic data structure used in processor processes.
- Maintains automatic process-state data, aiding program execution.
- Manages the flow of control in subroutines.
- Stores return addresses during subroutine calls and pops them upon returning.
- Reflects the execution state of the process.
- Stores subroutine arguments and local variables.
- Information pushed onto the stack as a result of a function call is called a "frame."
- The address of the current frame is stored in the frame pointer register.
- Dynamic structure supports any level of nesting within memory constraints.

- **Example of memory stack :**

```
1   b(){  .....   }
2   a(){
3       b();
4   }
5   main(){
6       a();
7   }
```



Register stack VS MEMORY stack

Register Stack	Memory Stack:
: Generally within the CPU	Typically in RAM.
Momentary spaces for internal processes in the CPU.	Series of memory spaces used in processor processes.
Provides fast access to data.	Data is temporarily stored in registers.
Limited in size.	Larger in size.

Addressing mode

- A technique used in computer architecture to specify how operands are chosen for instructions
- Determining how the processor references memory to fetch or store data.

1. Immediate Addressing Mode:

- Operand is specified explicitly in the instruction.
- Example: MOV AX, #5 (moves the immediate value 5 into register AX).

2. Register Addressing Mode:

- Operand is the content of a register.
- Example: ADD BX, CX (adds the contents of registers BX and CX).

3. Direct Addressing Mode:

- Operand is the content of the memory location directly specified by the instruction.
- Example: MOV AX, [1000] (moves the content of memory location 1000 into register AX).

4. Indirect Addressing Mode:

- Operand is the content of the memory location whose address is specified by a register or another memory location.
- Example: MOV AX, [BX] (moves the content of the memory location whose address is in register BX into register AX).

5. Indexed Addressing Mode:

- Operand is found at the sum of a base register and an index register, possibly scaled by a factor.
- Example: MOV AX, [SI + 100] (moves the content of the memory location whose address is the sum of the base register SI and the immediate value 100 into register AX).

6. Relative Addressing Mode:

- Operand is at a location whose address is given by the sum of the address in a register and a constant offset.
- **Example**:
  - Jump+3 if accumulator == 2
  - If accumulator != 2 then Jump +5
  - This is called a conditional jump and it is making use of relative addressing .

7. Stack Addressing Mode:

- Operand is implicitly the top of the stack.
- Example: PUSH AX (pushes the content of register AX onto the stack).

8. Displacement Addressing Mode:

- The effective address of the operand is calculated as the sum of the content of register BX and the displacement value (10), resulting in the memory address from which data is to be loaded into the AX register.
- Operand Address = Base Register + Displacement
- MOV AX, [BX + 10]
- In this instruction: BX is the base register, 10 is the displacement value.

9. Implied Addressing Mode:

- Operands are specified implicitly in the definition of the instruction .
- Zero address instruction is a stack organized computer are implied mode instruction since the operands are implied to be on the top of the stack .