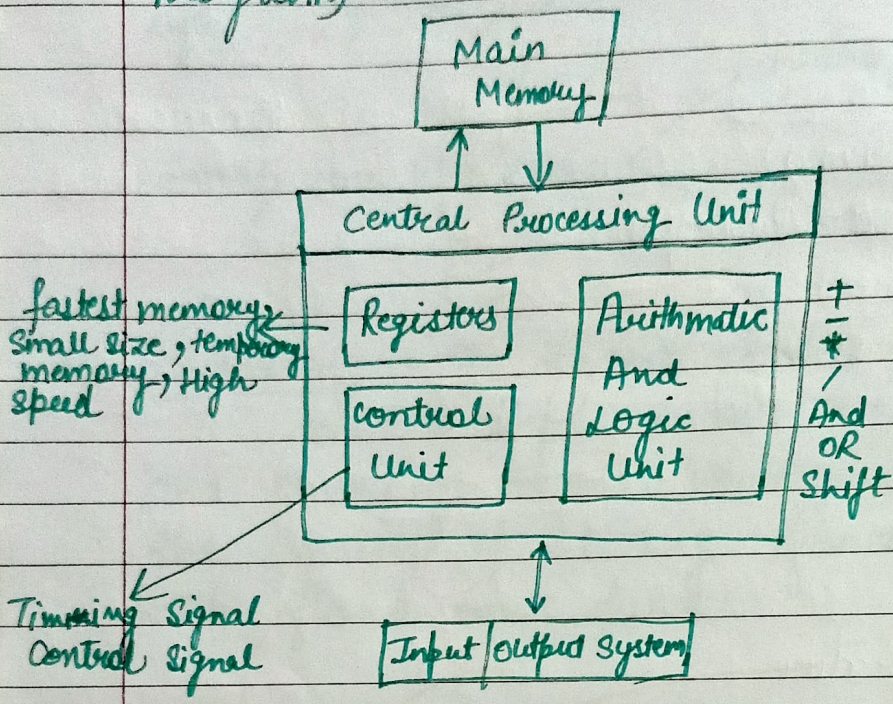
DOMS | Page No.
Date / /

"Von Neumann Architecture (Stored Memory Program)



Microprocessor: It is a small sized electronic component inside a computer that carries out various tasks involved in data processing as well as arithmetic & logical operations.
It is build over an integrated circuit comprising million of small components like resistors, transistors & diodes.

Microcontroller: It is a small computing device which has a CPU a fixed amount of RAM, ROM & other peripherals all embedded on a single chip.
keyboard, Mouse, Washing Machine, Digital Camera, Pen drive, Remote Controller, Microwave are few examples of microcontrollers.

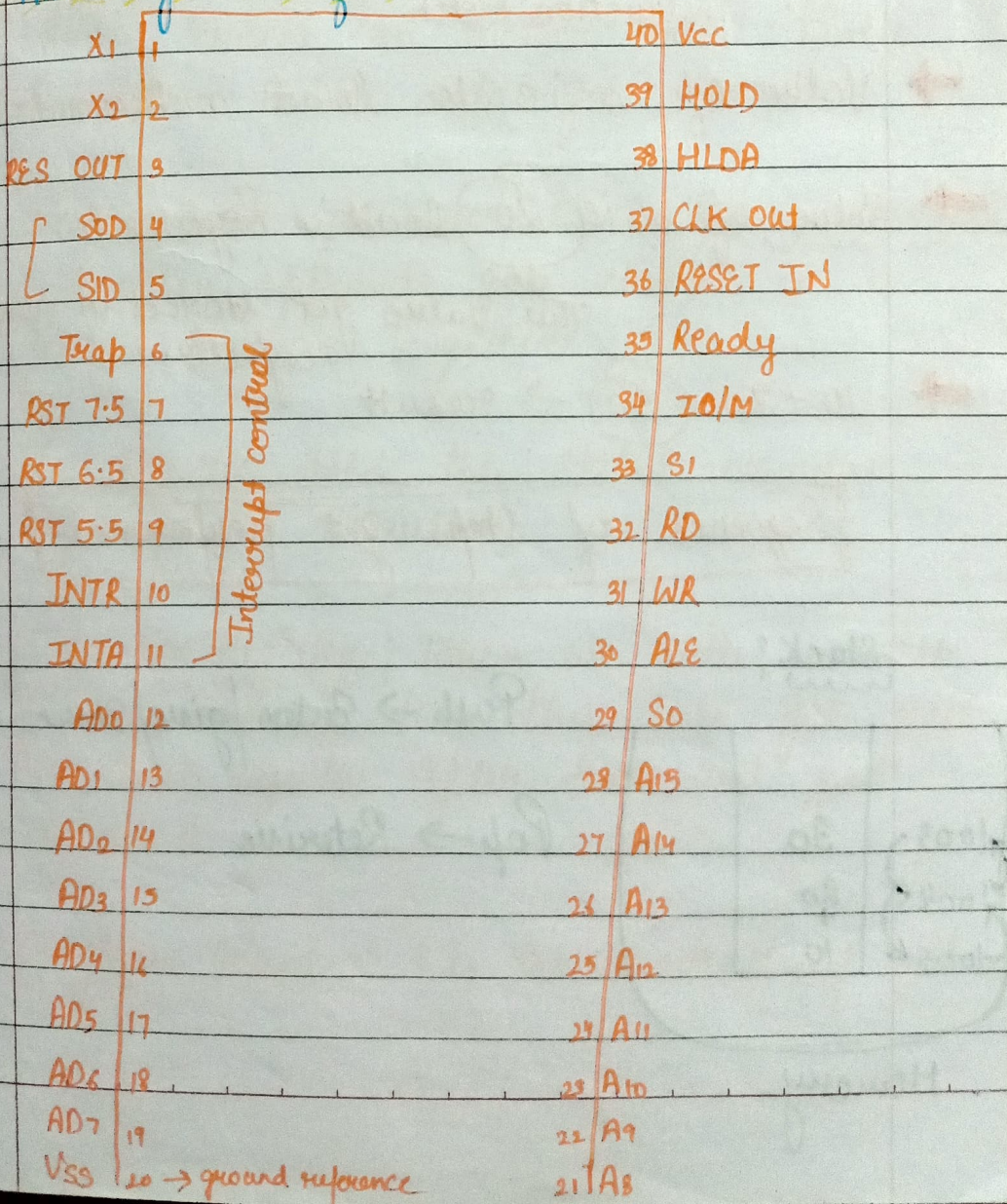
8085 Microprocessor

- Intel 8085 is an 8-bit microprocessor produced by Intel in 1976.
- It is modified version of 8080 with two extra instructions (RIM & SIM).
- It is 40 pin integrated circuit.
- Its operating frequencies are 3, 5 & 6 MHz.
- 8085 uses a single 5 volt (V) power supply.
- It has 16 bit address bus.
- Data bus width 8-bit.

$2^{16} \rightarrow 2^6 \times 2^{10}$

$64 \times 1KB = 64KB$

* Pin Diagram of 8085

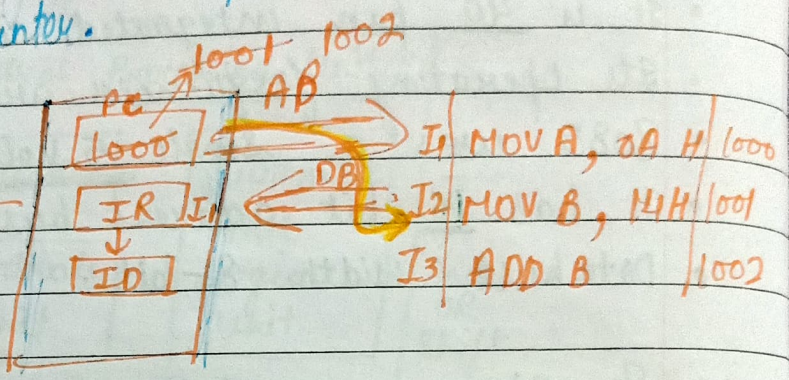


AD - Address Bus
DB - Data Bus
IR → Instruction Register
ID → Inst. Decode

DOMS | Page No. | Date

Program Counter : It holds the address. In this the address get incremented after fetching data. Operating System upload the address in program counter.

eg
```
int a, b, c
a = 10;
b = 20;
c = a+b
```



1. Instruction Fetch

→ Value of a i.e 10 load in Accumulator (A)

→ Value of b ie (20) load in Register B.

This value get stored in temporary register.

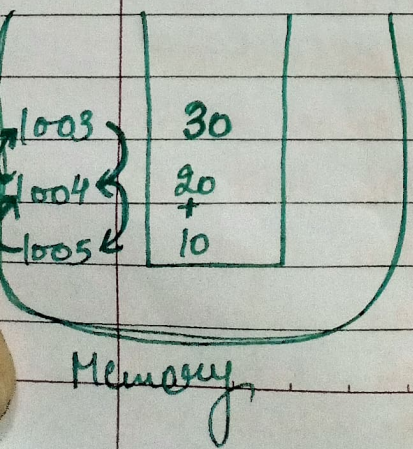
→ 10 + 20 = (30) → result

Sequence of Steps get performed.

Stack :

Push → Enter / give / Store data.

Pop → Retrive



W & Z are personal Registors of microprocessor. Users can't use it.

DOMS | Page No. | Date

Types of Registers In 8085

- General Purpose
  - B(8) | C(8)
  - D(8) | E(8)
  - H(8) | L(8)
- Special Registers
  - Accumulator (8)
  - Flag Register (8)
  - Instruction Register (8)
  - Program Counter (16)
  - Stack Pointer (16)
  - Temporary Register

Registers : Storing Unit, Fastest Storing Unit, Embed on mother board. Only 8 bit or 16 bit data get stored.

* One Operand get stored in Accumulator (Ac) and other operands get stored in general purpose registers.

* Flag Register tells the status or the sign of the result

* Instruction Register : When we call the inst from memory then this inst get rest in Inst register after that inst get decoded.

* Program Counter has the address of inst.

* Stack Pointer : Use to Push & Pop the data.

* Temporary Register used to store the temporary data

Flag Register : 8 bit, 5 Flags

S	Z	-	AC	-	P	-	CY
Sign	Zero	X	Aux carry	X	Parity	X	carry

eg A = 10 H
B = 20 H
ADD B AC ← AC+B

Carry bit:

*
```
  00010000
+ 00100000
  00110000
```
no carry

If Carry is 1 then CY = 1
If Carry is 0 then CY gets to 0

even

* Parity Bit :

If the total no. of 1's in result is 'even' then P is 1

If the total no. of 1's in result is 'odd' then P is 0

DOMS | Page No.
Date / /

* Auxiliary Carry: If D_3 produce carry then AC get set to 1. If not then AC set to 0.

* Zero: If the result is full zero then Z set to 1. If not then it set to 0.

* Sign: D_7 MSB bit tells the sign.
  $D7 = 0 = +ve$
  $D7 = 1 = -ve$

Instructions in 8085

An inst is a command to the microprocessor to perform a given task on a particular data at a particular time.

- One word or 1 byte inst (8-bit)
- Two " " 2 " " (16-bit)
- Three " " 3 " " (24-bit)

Branch Inst in 8085:

Unconditional Branching

- JMP (Jump)
- CALL (Call Subroutine)
- RET (Return)

Conditional Branching (3 Byte)

- JC (Jump if Carry)
- JNC (Jump if no carry)
- JZ (Jump if Zero)
- JNZ (Jump if not Zero)
- JPE (Jump if Parity even)
- JPO (Jump if Parity Odd)
- JM (Jump if Minus)
- JP (Jump if Plus)

Muskan

DOMS | Page No.
Date / /

* Unconditional Branching: In this no condition get checked.
[Immediate addressing mode]
- JMP = 3 Byte Opcode → C3 [JMP 2021]
- CALL = 3 Byte [Call 2021]
- RET = 1 Byte Opcode → C9

* Conditional: If the condition is true then we jump to the particular address otherwise we will continue our execution.

→ Types of Buses:

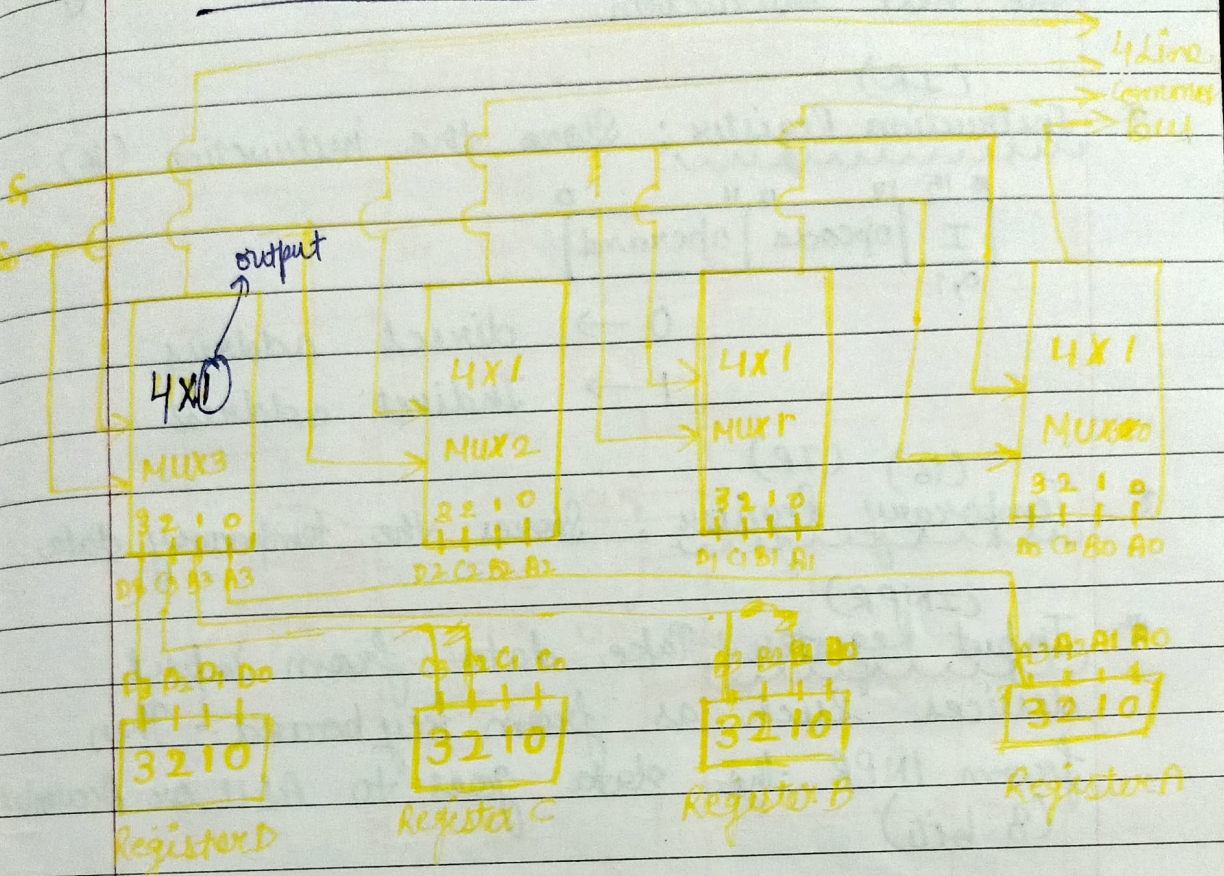
1. Address Bus: 16 bits, Unidirectional, Carries address.

2. Data bus: Bidirectional, Carry data, It is also used between registers to travel data, Length depend on Word Size, 16 bits.

3. Control Bus: Use to send control signals or timing signals

DOMS | Page No.
Date / /

Common Bus System (Using Multiplexer)



→ No of MUX = No. of bits in registers

→ No. of Input = No. of registers (4)

Various types of Registers :

* Address Register (AR) : To fetch the data from the particular address (12)

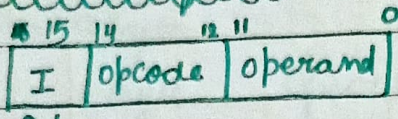
* Data Register (DR) : Where we store data (16)

* Accumulator : Store intermediate result (16)

DOMS | Page No.
Date / /

(PC)
* Program Counter: It stores the address of the next instruction.

(IR)
* Instruction Register: Stores the instruction (16)



0 → direct address
1 → Indirect address

(16) (TR)
* Temporary Register: Stores the temporary data

(INPR)
* Input Register: Take data from input devices such as from keyboard. Then from INPR the data goes to ALU or Accumlator (8 bit)

(OUTR)
* Output Register: 8 bit, It ~~dat~~ take data that the ALU produce to give data to output devices.

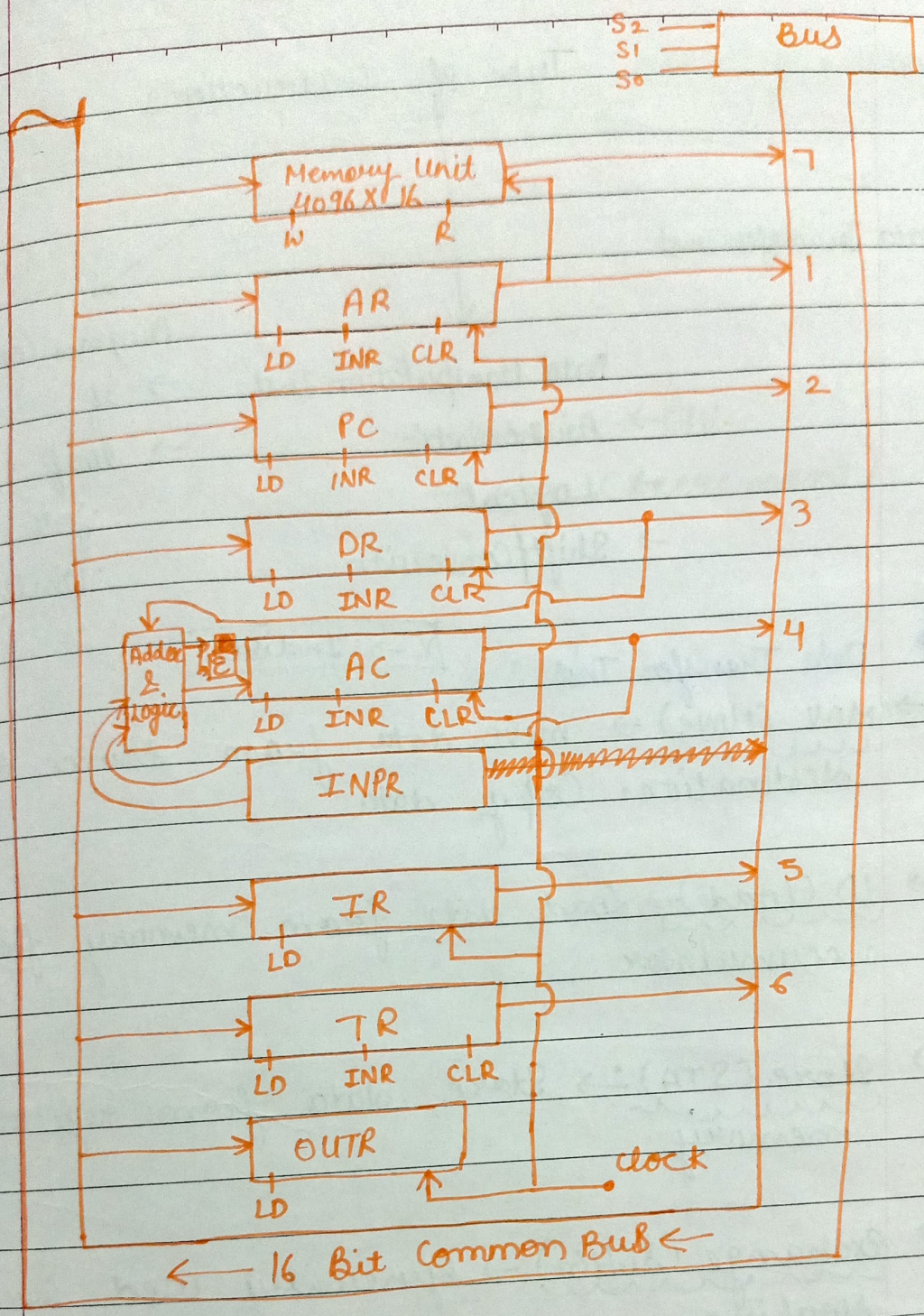
Common Bus System

- W → Write
- R → Read
- LD → Load
- INR → Increement
- CLR → Clear

eg:
One address Inst: X = (A+B)*(C+D)

Load A	AC ← M(A)
Add B	AC ← AC + M(B)
Store T	TR ← AC
Load C	AC ← M(C)

Common Bus System



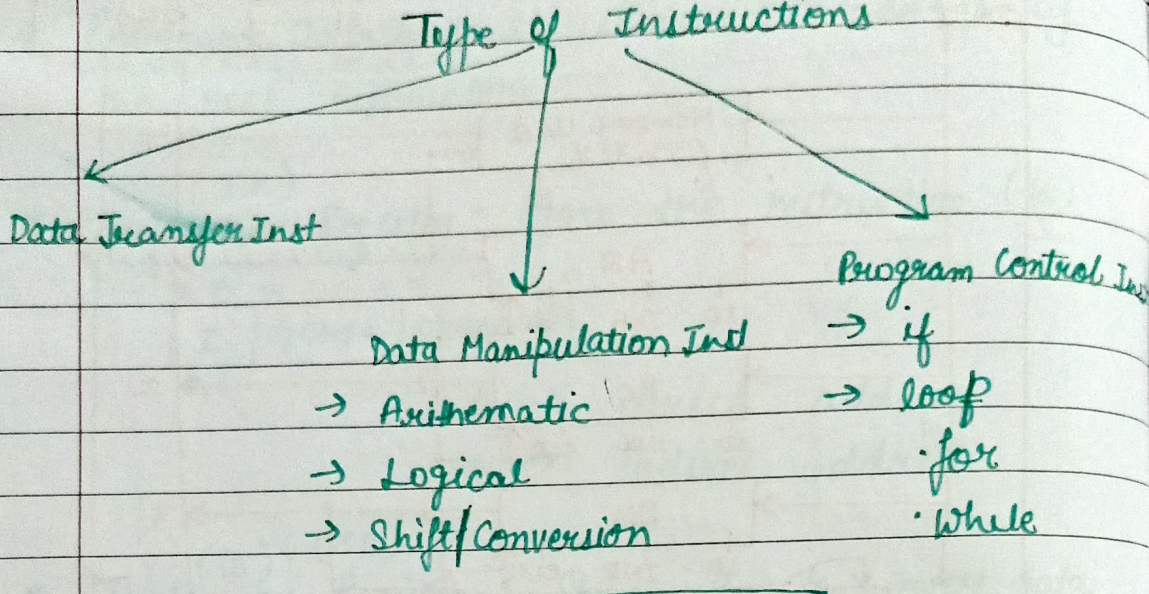
ADD D AC ← AC + M(D)

MULT AC ← AC * TR

Store X

Inst Set : Set of Commands

DOMS | Page No.
Date / /



X → Indirect.

➡ Data Transfer Inst :

→ MOV (Move) → move data from source to destination. Copy data

→ LD (Load) → Load data from memory for accumulator

→ Store (STA) → Store data from register to memory

→ Exchange (XCHG) → generally used for swaping

→ Input (IN) → Take data from phenipheal devices to memory.

→ Output ( ) → To give output or give data to phenipheal devices.

DOMS | Page No.
Date / /

→ Push → Enter Word in Memory, Used memory as Stack (LIFO)

→ Pop → Retrieve data

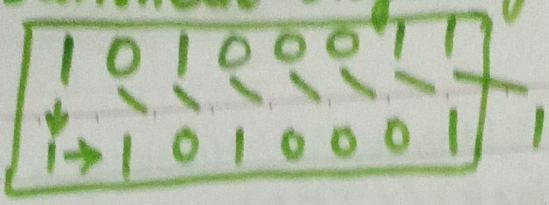
➡ Arithmetic Inst :
→ Add → Sub → MUL → DIV
→ INC (Increment) → DEC (Decrement)
→ Add with Carry (ADC)
→ Sub with borrow
→ Negate [-ve show krna]

➡ Logical Inst
→ Complement (COM or NOT)
→ Clear (CLR) → Logical AND (AND)
→ Logical OR (OR) → XOR (EX-OR)
→ Clear Carry (CLRC)
→ Set Carry (STC)
→ Complement Carry (CMC)
→ Enabled Intrupt (EI)
→ Disabled Intrupt (DI)

➡ Shift Inst : ×2 ↓
→ Logical Shift Left | 0 0 0 0 1 0 1 0 | → 20
| 0 0 0 1 0 1 0 0 |

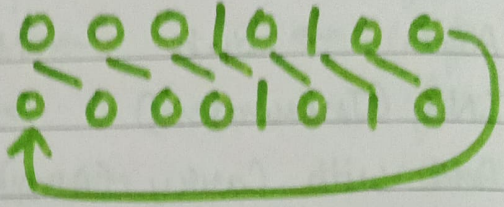
→ Logical Shift ÷2 Right | 0 0 0 0 1 0 1 0 | → 5
| 0 0 0 0 0 1 0 1 | 0

→ Arithmetic Shift Right :

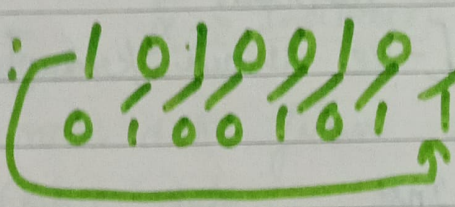


→ Arithmetic Shift Left : [Same as Logical Left]

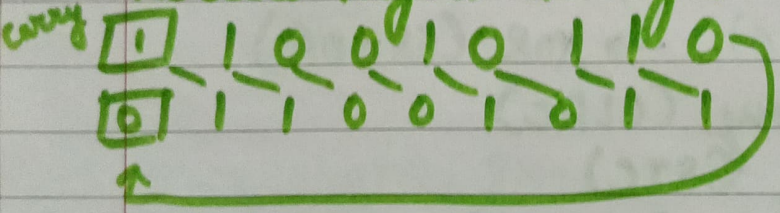
→ Rotate Right :



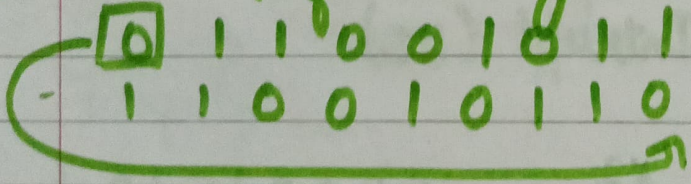
→ Rotate Left :



→ Rotate Right through Carry :



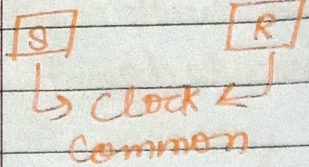
→ Rotate Left through Carry :



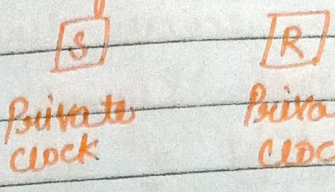
Concept of Asynchronous data transfer

DOMS | Page No.
Date / /

Synchronous



Asynchronous



→ Two units share a common clock.

→ Two units are independent & each unit has its own private clock.

→ Both devices have to coordinate for data transfer.

- Asynchronous data transfer :

→ It is used when speed of I/o devices do not match with processor & Timing characteristics of I/o devices is not predictable

Define Asynchronous data transfer between two independent units requires control signals to be transmitted between communating units to indicate the time at which data is being transmitted.

In asynchronous data tranfer 2 two methods are used :

1. Strobe Control :

→ A strobe pulse is supplied by one of the unit to indicate the other unit when the transfer has to occur.

Handshaking Method:

→ A control signal is accompanied with each data item being transmitted to indicate the presence of data

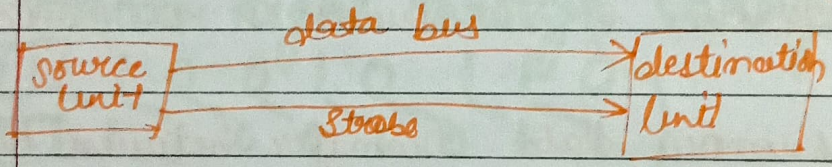
→ The receive unit responds with another control signal to acknowledge the receipt of data.

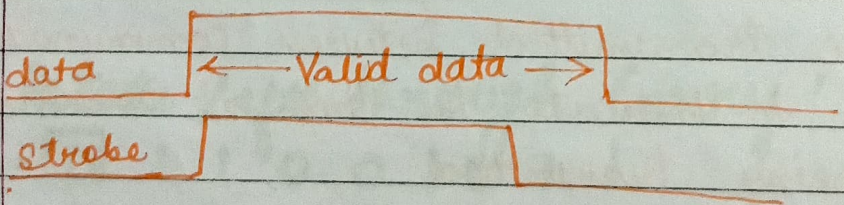
Strobe Control :-

→ It employes a single control line to time each transfer.

→ The strobe may be activated by either source or destination unit.

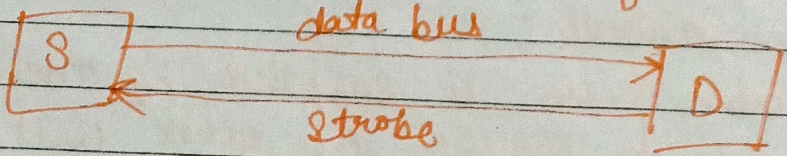
Source initiated strobe for data transfer.

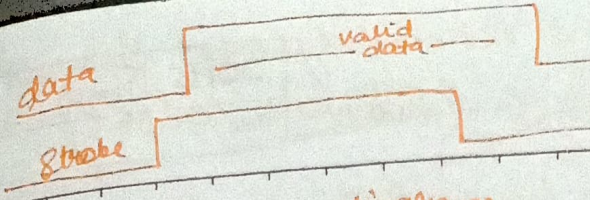




Timing diagram

Destination initiated strobe for data transfer

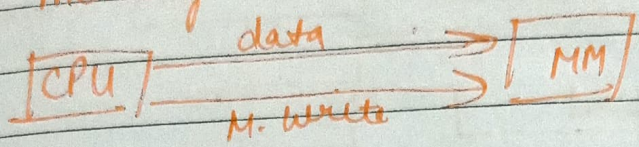




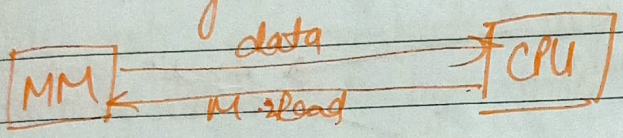
DOMS | Page No.
Date / /

Timing diagram

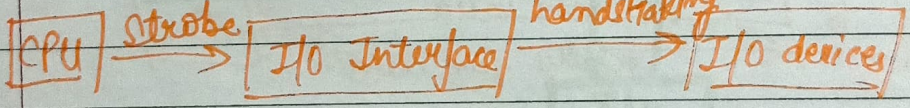
Source initiated eg:
memory write control signal from CPU to memory unit.



Destination initiated ed.
Memory read control signal from CPU to memory unit.



disadvantage:
→ In this there is no way of knowing whether the destination unit has actually received the data item that was placed in the bus



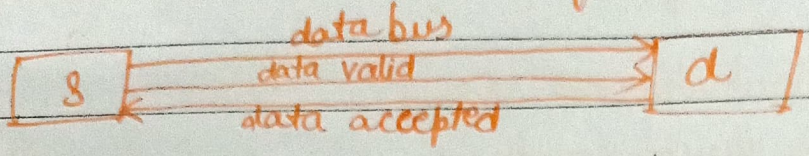
Handshaking Method :

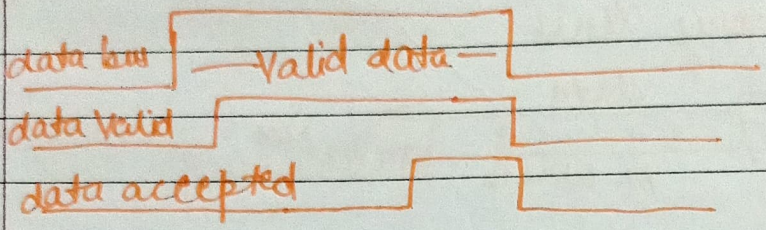
→ Solves the problem of Strobe method by introducing a second control signal that provides a reply to unit that initiates the transfer.

Strobe Control + Acknowledgement Signal
[Two wire control]

DOMS Page No. Date /

Source - Initiated handshaking





Source

Destination

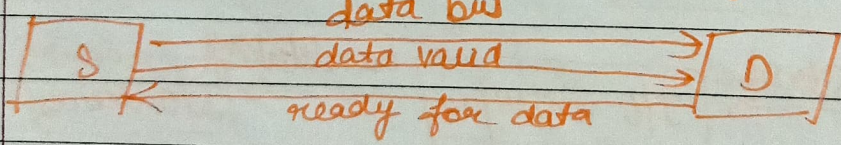
Place data on bus
Enable data valid

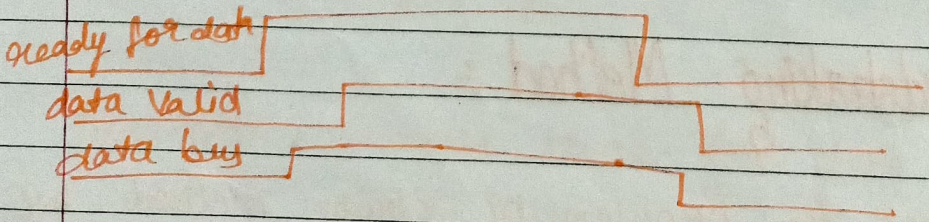
Accept data from bus
Enable data accepted

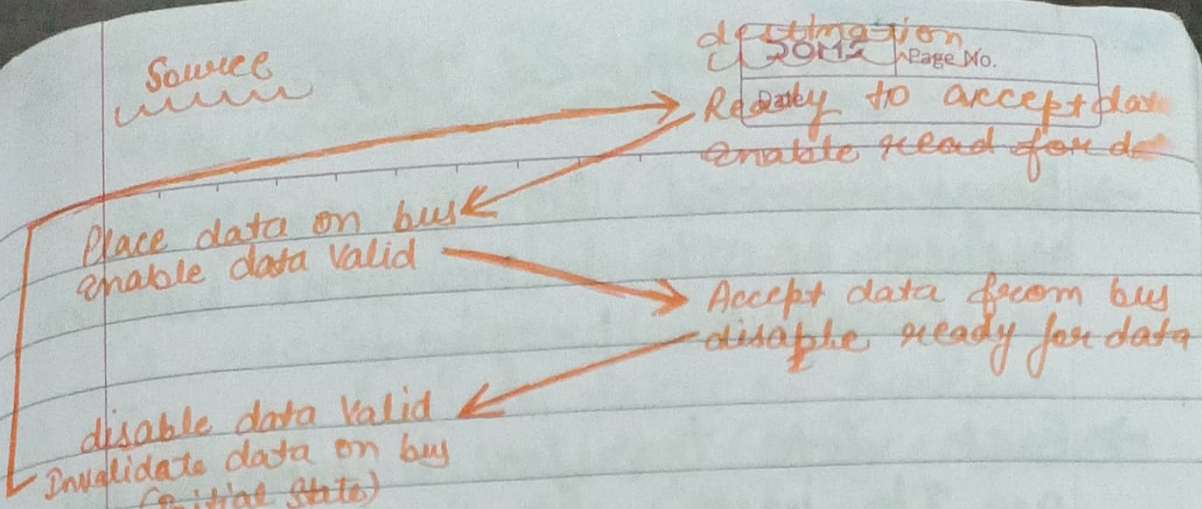
Disable data valid
Invalidate data on bus

Disable data accepted
ready to accept data
(Initial State)

Destination Initiated handshaking :





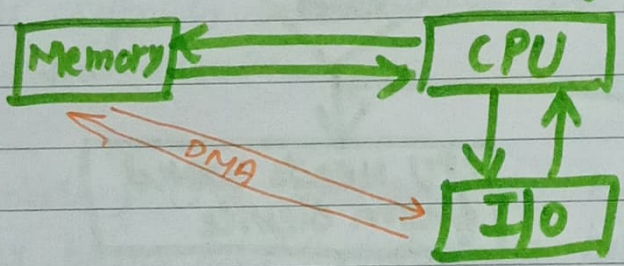


Advantage:

→ It provides a high degree of flexibility & reliability.

→ If one unit is faulty, the data transfer will not be completed such an error can be detected by time out mechanism which produces an alarm if the data transfer is not completed within a predetermined time.

Modes Of Transfer :-



Data transfer between the central computer & I/O devices may be handled in variety of modes.

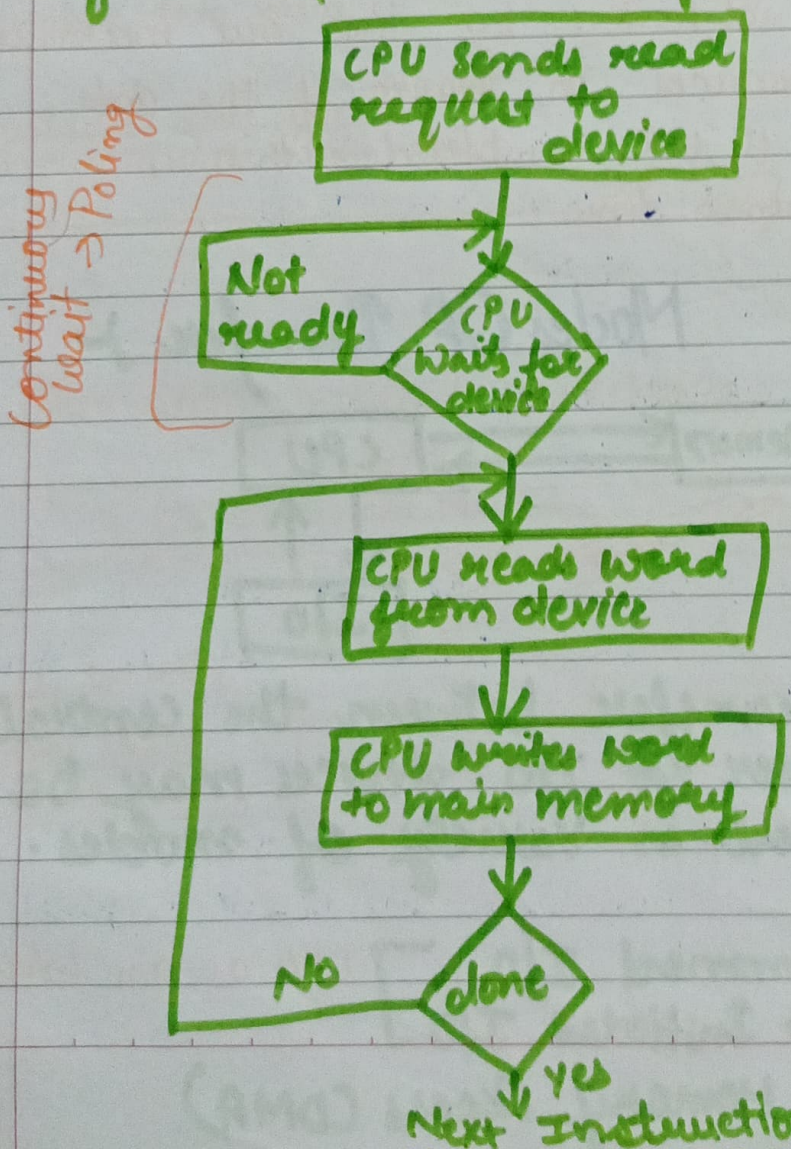
1. Programmed I/O] CPU as an intermediate
2. Interrupt Initiated I/O]
3. Direct Memory Access (DMA)

Programmed I/O

→ Programmed I/O operations are result of I/O instructions written in computer program.

→ Each data transfer is initiated by an I/O instruction in the program (to access registers or memory on a device).

→ Transferring data under program control requires constant monitoring of the I/O devices by the CPU.



→ In program I/O, CPU issues an I/O request & then CPU stays in program loop (Polling) until the I/O device indicates that it is ready for data transfer. [The I/O devices takes no further action to alert the CPU (i.e it does not interrupt the CPU]

Disadvantage
→ Time Consuming process Since it keep CPU busy needlessly.

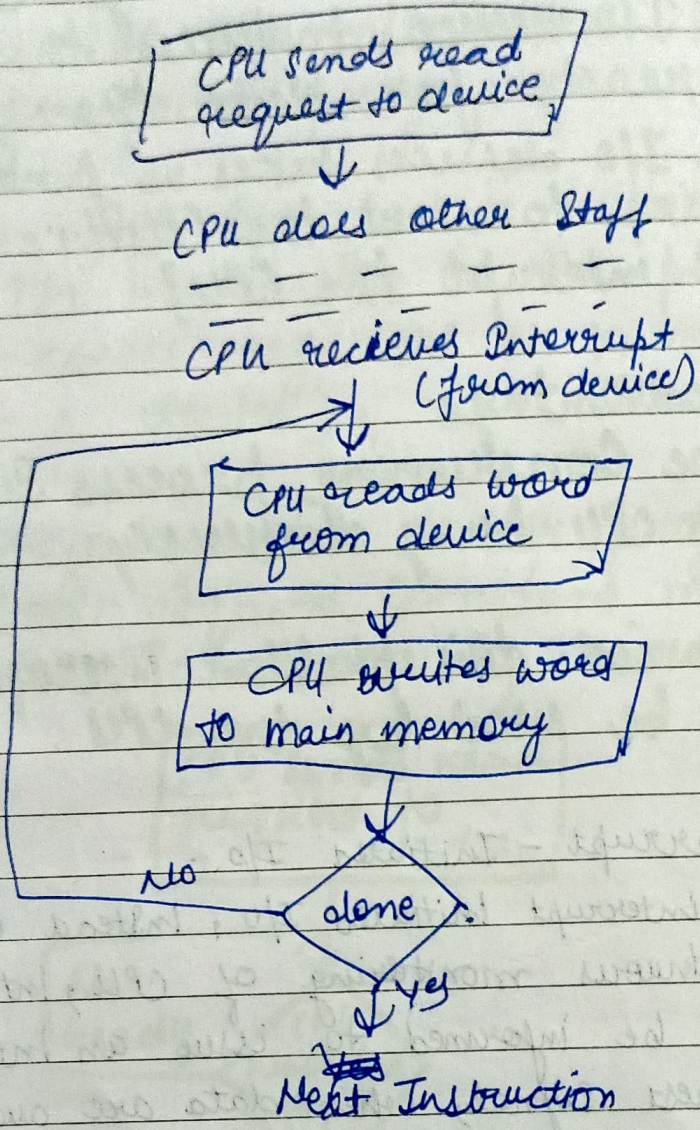
To avoid this problem interrupt facility can be used by the CPU.

* Interrupt - Initiated I/O.
→ In interrupt Initiated I/O, instead of continuous monitoring of CPU, interface will be informed to issue an interrupt request signal, when data are available from the device.

→ Meanwhile, CPU proceeds to execute another programs & interface keeps monitoring the device.

→ When device is ready for data-transfer it generates interrupt request.

→ Upon detecting the external interrupt signal, the CPU stops the task it is performing, process I/O data transfer & then resumes the original task it was performing



Advantages:

→ during interrupt the wait period of CPU is ideally eliminated.

* Direct Memory Access (DMA)

→ To transfer large blocks of data at high speed between external device & main memory DMA approach is used often.

→ DMA allows data transfer between I/O device & main memory with minimal intervention of CPU.

→ DMA means CPU grants I/O interface authority to read from or write to memory without its involvement.

→ DMA itself controls data transfer between main memory & I/O device.

→ CPU is only involved in beginning & end of the transfer & interrupted only after entire block has been transferred.

CPU sends read request to DMA unit

↓

CPU does other Staff

— — — — —

CPU receives DMA interrupt (from I/O device)

↓

Next Instruction

* CPU ask the DMA controller to transfer data between a device & main memory & then CPU proceeds to execute other task.
* The DMA Controller issues a requests to the right I/O device, waits & m data transfer between the device & main memory
* When data transfer is finished, the DMA Controller interrupt the CPU.