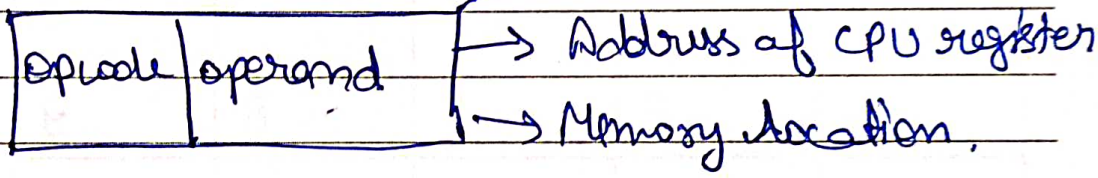
JULY 2012

Note

Addressing Mode :-

Introduction :-

A term addressing mode refers to the mechanism for specifying operand's. An operand may be specified as the part of the instruction or reference of the memory location's can be given. An operand could also be an address of the CPU registers.



Def:

The operand of an instruction is stored in main memory or CPU registers. Different methods are used for specify the operand in an instruction are called Addressing modes. A computer may use one or more addressing mode in its organization.

These different addressing modes are helpful in writing small, efficient & fast programs. These modes are handling complex data like indexing of an array, looping, program relocation & pointers to memory.

AUGUST 2012

August 01 Wednesday Register addressing is very efficient as no memory fetch is required to bring the data. In addition, very few bits are required to address a register. Thus, we save both in terms of time and space.

Types of Addressing Modes:-

1. Immediate Addressing Mode.
2. Implied " "
3. Direct " "
4. Indirect " "
5. Register " "

August 02 Thursday

6. Register Indirect " "
7. Auto Increment or Auto Decrement
8. Relative Addressing Mode.
9. Index Addressing Mode.
10. Base Register Addressing Mode.

(1) Immediate Addressing:-

Operand is mentioned in the instruction. The instruction is having operand field rather than address field. Hence no fetch operation is required. This mode provides data to its opcode immediately. Thats why this mode is called Immediate Addressing Mode.

AUGUST 2012

August 03 Friday

Syntax: | opcode | operand |

Disadvantage:- Value of data is limited by the length of the operand field in the instruction.

Advantage:- (1) No additional memory fetch cycles are required for fetching the operand.

(2) It is fast.

August 04 Saturday

e.g.:-

(1) Add R1, 15

R1 ← R1+15

(2) Move R0, 300

R0 ← 300.

(2) Implied or Implicit Addressing Mode:-

In this mode, the operands are specified implicitly in the definition of the instruction. This means operand are not present in the instruction.

August 05 Sunday

Syntax: | opcode |

August 06 Monday

Examples:

Top of Stack.

(i) POP → It will fetch out the contents from TOS

(ii) STC → Set Carry flag

(iii) HLT (Stop processing)

(iv) CMA (Complement AC)

(v) CIR (Circular Shift right)

(vi) CIL (Circular Shift left)

(vii) INC (Increment AC)

All these instruction do not have any specified operand field. This type of addressing mode is limited to specified no. of instruction.

August 07 Tuesday

3) Direct Addressing Mode:-

In this mode, the operands are residing in the memory. The address of the memory location where operands are residing is specified in the instruction itself as memory address. That means the address of the operand is given by the address field of the instruction. Only one memory access is required for operand fetch. That is why it is termed as direct addressing mode.

(10)

August 08 Wednesday

Syntax:

Opcode	Memory address

Effective address = Address field.

Ex:-

(1) LDAM[x]

AC ← M[x]

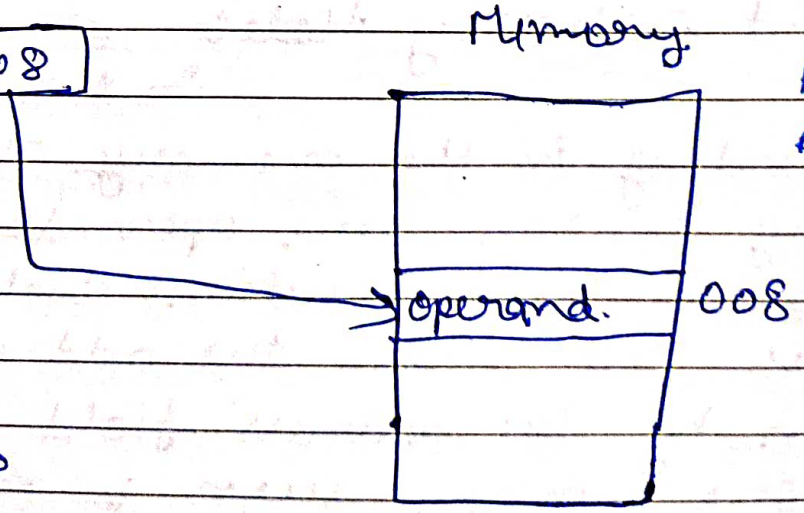
This instruction will load the contents of memory location having address x to AC.

August 09 Thursday

(2) ADD 008

The data value found at memory address 008 is added into AC.

ADD	008



AC = 500

M[008] = 100

After Execution: AC ← AC + M[008]

AC ← 500 + 100

AC ← 600

AUGUST 2012

August 10 Friday (4) Indirect Addressing Mode's

Syntax

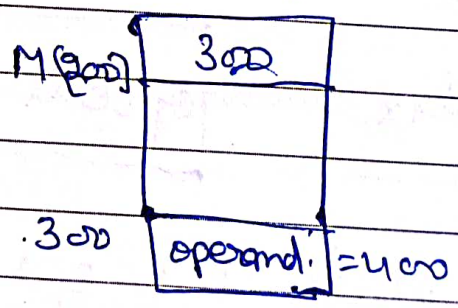
opcode	Memory Address

Effective Address = Memory [AF]

The memory address field of the instruction specifies the memory address where the operands are residing. That means the instruction specifies the memory address which further contains the memory address where the operand is residing. This addressing mode is helpful to implement

August 11 Saturday the concept of pointers.

Ex:- (1) Load R1, M[M[200]]

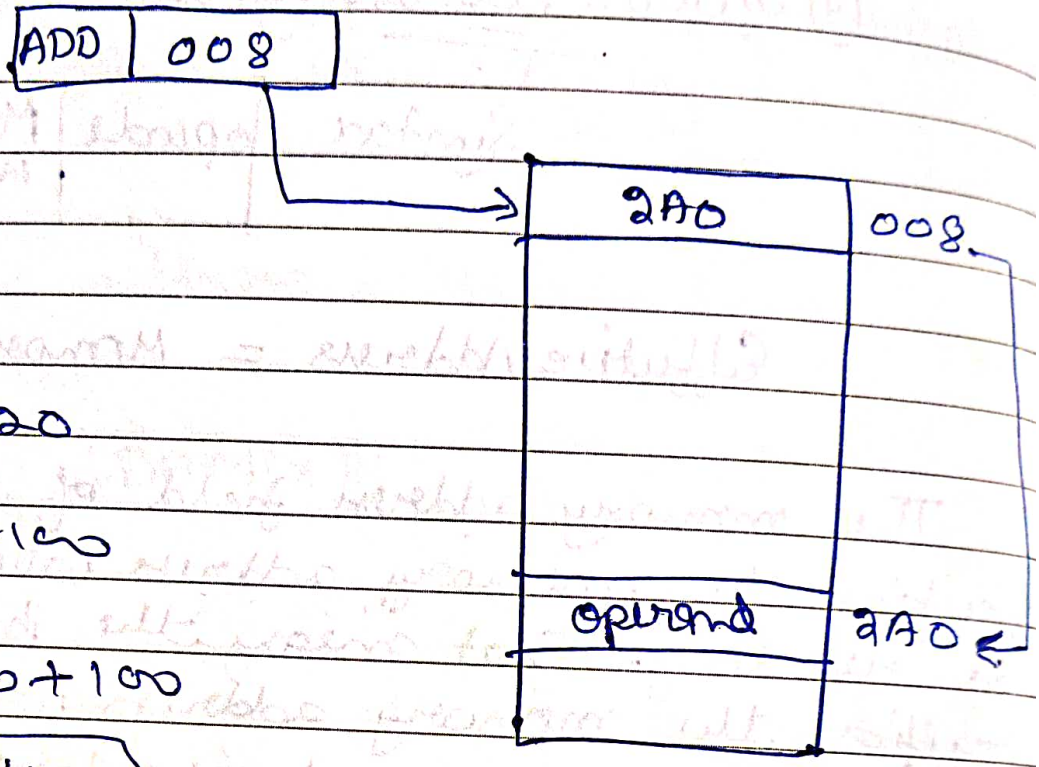


R1 ← 4000.

August 12 Sunday (5) Add 008

(12)

August 13 Monday



AC = 320

operand = 100

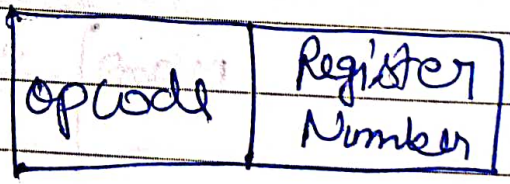
AC ← 320 + 100

AC ← 420

August 14 Tuesday

(5) Register Addressing Mode:

Syntax



In this Mode, the operands are in the one of the CPU registers. The register address is given in the instruction. This addressing mode provides fast access of operan without any memory access. It is even faster than direct addressing mode.

August 15 Wednesday This mode is also called register direct addressing mode.

Ex :-

(i) Add R_1, R_2

$R_1 \leftarrow R_1 + R_2$.

(ii) Add R_1

$AC \leftarrow AC + R_1$

$AC = 100$

$R1 = 200$

August 16 Thursday

$AC \leftarrow 100 + 200$

$\boxed{AC \leftarrow 300}$

(3) Register Indirect Addressing Mode :-

Syntax	Opcode	Register Number

Effective Address = M[R]

In this mode, register contains the memory address of operand rather than the operand itself. That means register contains the address of operand than that of

August 17 Friday operand itself. This mode uses register pair to contain 16 bit address of operand.

Ex: LDAX B AC ← M[BC]

LDAX B means load the AC with the contents of memory whose address is residing in the register pair BC.

This mode is helpful for quick access of memory location in array.

(2) Auto Increment or Auto decrement mode:

August 18 Saturday When the address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register after every access to a table. This can be achieved using this mode.

Syntax:

Opcode	Register Number

In auto increment the register pair is incremented after the execution of the instruct

August 19 Sunday In auto decrement the register pair value is decremented before the execution of the instruction. This is required in sequential execution of the instruction in the program.

1000

(15)

AUGUST 2012

igust 20 Monday Ex:- LDAX B AC ← M[BC].

In auto increment the value of register pair BC is incremented after the execution of instruction.

In aut decrement the value of register pair BC is decremented before the execution of the instruction.

(a) MOVE (R2), +R0 ; It copies the contents of R0 to R2 then the value is incremented by 1.

(b) MOVE R1, −(R0) ; Initially contents R0 is dec. then copied to R1.

igust 21 Tuesday

Add AC, (R1)+

⇒ ADD AC ← R1

R1 ← R1 + 1

After accessing the operand, the contents of this register is incremented to point the next item in the list.

LDAX B

AC ← M[BC]

1000

1001

(16)

1000
PC 1001

August 22 Wednesday

(8) Relative Addressing Mode:-

Syntax:- | opcode | offset |

EA = Address field of inst.
+
Contents of PC

In this mode, the address field part (offset) of instruction is added to program counter to get the effective address. At this EA, the operands are residing in memory.

August 23 Thursday

Since, offset part can be -ve or +ve.

Ex:- (i) JUMP +8 (PC)

100
8
108

EA = PC + 8

PC = 300 (offset is +8)

∴ EA = 308

(ii) JUMP -8 (PC)

EA = PC - 8

∴ EA = 292

ie at 292, operands are residing.

August 24 Friday ⇒ This mode is used for branch address
⇒ This mode provides quick address calculation with memory access.

(9) Index Address Mode :-

Syntax | Opcode | Address |

In this mode, the address part of instruction is added to the contents of index register to get the effective address. Index register is one of the registers of CPU containing index value. That index value is termed as offset or
August 25 Saturday displacement.

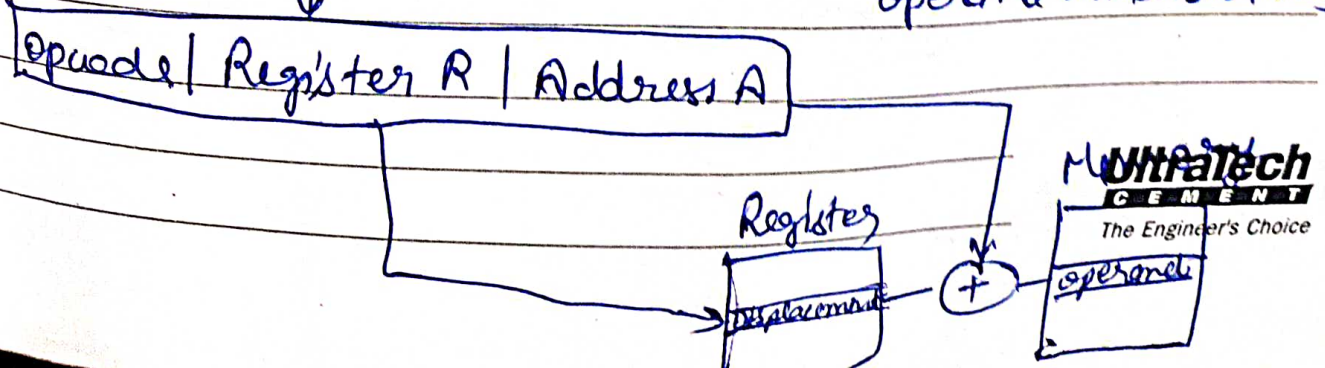
EA = Address + Contents
field of Index Register.

Ex :- Add 002

R1 ← Index Register & holds the value 1

EA = 002 + 1 = 003

August 26 Sunday



(18)

August 27 Monday (10) Base Register Addressing Mode:-

Syntax | opcode | offset |

EA = Address field + Contents of Base Register

In this mode, the offset part of instruction is added to the contents of base register to get the EA. At this EA the operand is residing.

Usage:- (i) It is quite useful for accessing array element as well as character's in string.

August 28 Tuesday (ii) Relocation of program in memory. This is generally required in multiprograming environment.

2 मई

AUGUST 2012

Chapter 11

August 29 Wednesday

I/O Organization

I/O Interface :-

I/O interface provides a method for transferring information b/w internal storage and external I/O devices. The purpose of the communication link is to resolve the differences that exist b/w the central computer and each peripheral. The major differences are :

① peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. ∴ a conversion of signal values

August 30 Thursday may be required.

② The data transfer rate of peripherals is slower than the transfer rate of the CPU, synchronization mechanism may be needed.

③ Data code and formats in peripherals differ from the word format in the CPU and memory.

④ The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

August 31 Friday To resolve these differences, computer systems include special hardware components b/w the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface b/w the processor bus and peripheral devices.

Two main types of interface are CPU interface that corresponds to the system bus and I/O interface that depends on the nature of input-output device.

The main function of input-output interface circuit are data conversion, synchronization and device selection.

* Data conversion refers to conversions b/w digital and analog signals and conversions b/w serial and parallel data formats.
* Synchronation refers to matching of operating speeds of CPU and other peripherals
* Device Selection refers to the selection of I/O device by CPU in a queue manner.

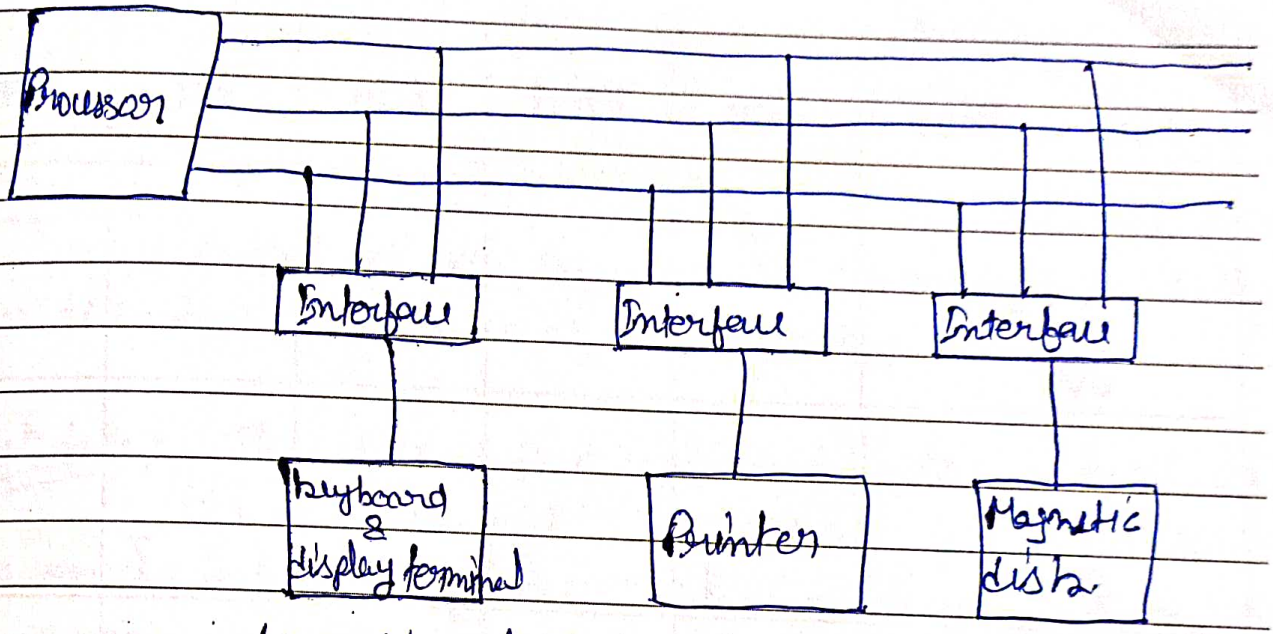
Note

I/O Bus and Interface Modules:

A typical communication links b/w the processor and several peripherals is shown in figure. Each peripheral device has associated with it an interface unit.

Each interface decodes the address and control received from the I/O Bus, interprets them for peripheral and provides signals for

Note peripheral controller. It also synchronizes the data flow and supervises the transfer b/w peripheral and processor.



Connection of I/O bus to input-output devices.

The I/O bus from the processor is attached to all peripheral interfaces. To communicate with a particular device, the processor places a device address on the address lines. Each interface attached to the I/O bus contains an address decoder that monitors the address lines. When the interface detects its own address, it activates the path b/w the bus lines and the device that controls. All peripherals whose address does not correspond to the address in the bus are disabled by their interface.

Control Command :- A control command is issued to activate the peripheral and to inform it what to do.

SEPTEMBER 2012

September 01 Saturday Status :- A status command is used to test various conditions in the interface and peripheral. e.g- that may check the status of peripheral before a transfer is initiated. During the transfer, one or more errors may occurs which are detected by the interface.

Output data :- A data output command causes the interface to respond by transferring data from the bus into one of its registers.

Input data :- The data input command is the opposite of the data output. The interface places the data on the data lines, where they are accepted by the

September 02 Sunday processor.

Asynchronous Data Transfer :-

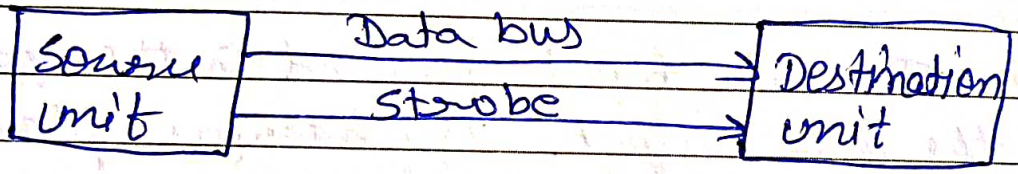
Asynchronous data transfer b/w two independent units requires that control signals be transmitted b/w communicating units to indicate the time at which data is being transmitted.

* One way of achieving this is by means of a strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occure.

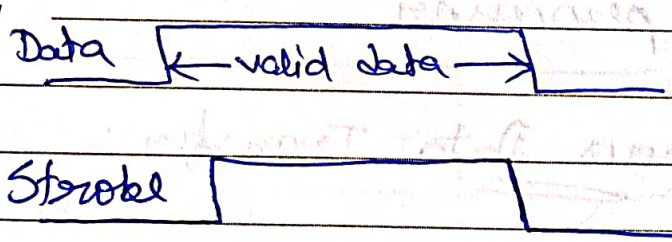
SEPTEMBER 2012

Strobe is a single line that informs the destination unit when a valid data word is available in the bus.

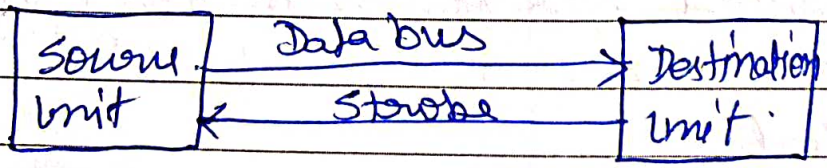
September 03 Monday Another method commonly used is to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement b/w two independent units is referred to as handshaking

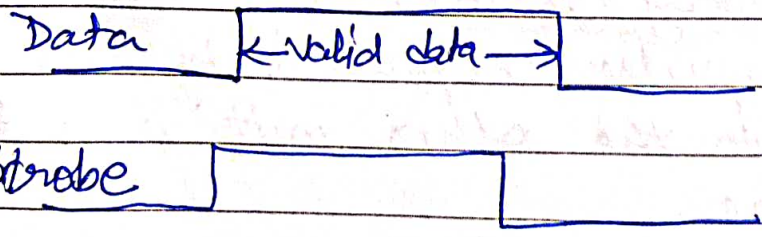


September 04 Tuesday



Source-initiated strobe for data T/f.

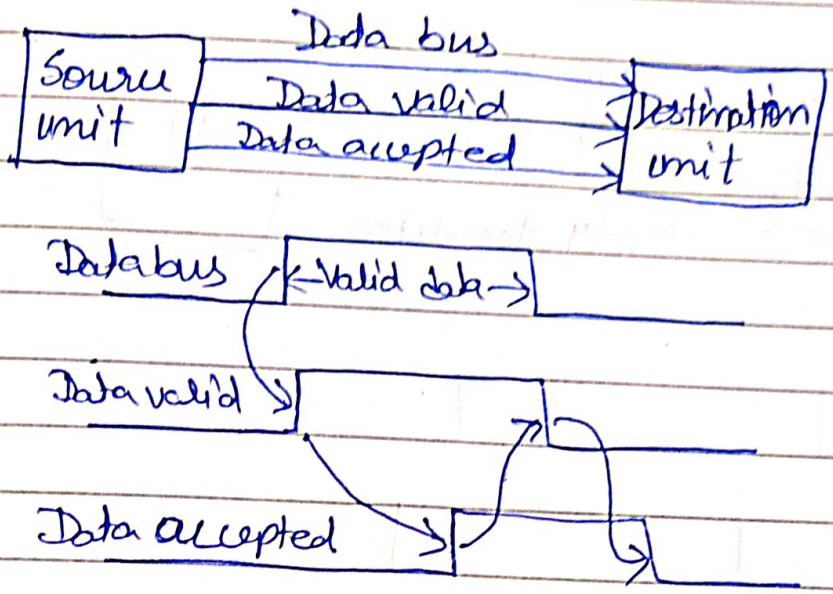




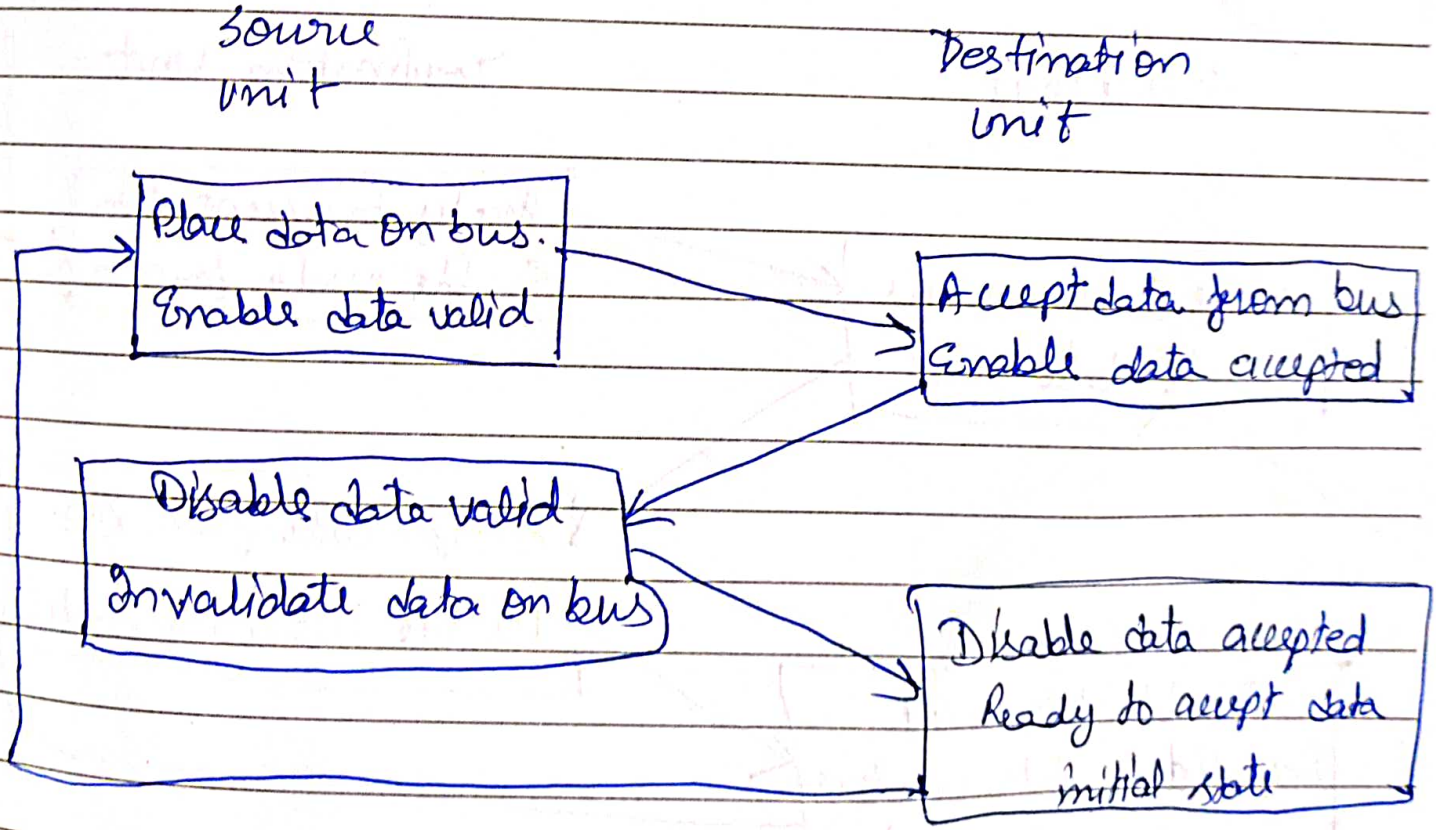
Destination-initiated strobe for data Transfer.

SEPTEMBER 2012

September 05 Wednesday **Handshaking**



September 06 Thursday

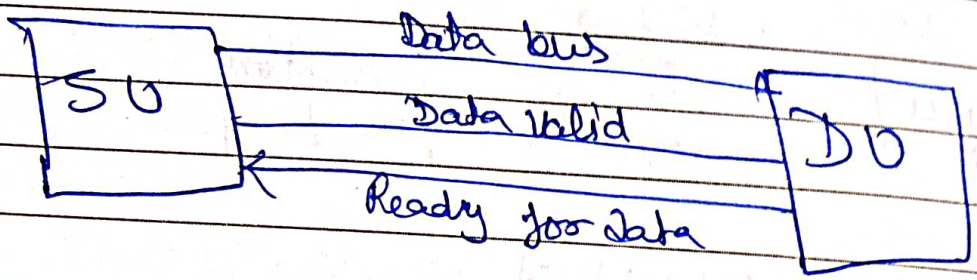


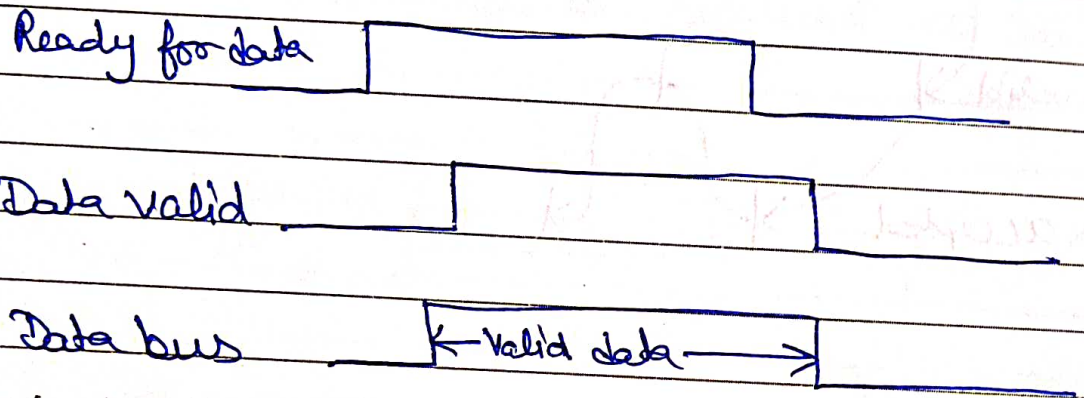
Source-initiated T/F using handshaking

SEPTEMBER 2012

September 07 Friday

Destination-initiated using handshaking





September 08 Saturday

Source unit

Destination unit

Place data on bus
Enable data valid

Ready to accept data
Enable ready for data

Accept data from bus
Disable ready for data

September 09 Sunday

Disable data valid
Invalidate data on bus
(initial state)

SEPTEMBER 2012

September 10 Monday

Synchronous :- When two units share same clock, then the transfer b/w two units is said to be synchronous.

Asynchronous :- When the two units uses its own clock, that transfer is the asynchronous transfer.

September 11 Tuesday