Chapter : 1

Computer Data Representation

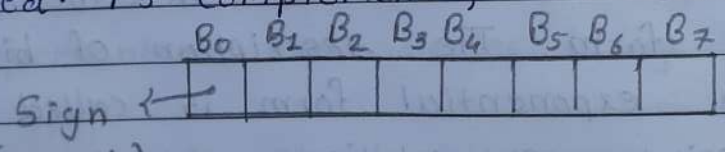
(1) Explain fixed point Representation.

-> In Integer number as radix point is fixed and assume to be to right of the right most digit. As radix point is fixed, the number system referred as fixed point number system.

-> Integer numbers are represented in two forms: signed Integer and Unsigned Integer.

-> Unsigned integer number represent positive numbers.

-> Computer doesn't have provision to represent negative sign so there are various technique used. 1's complement, 2's Complement.



-> The left most bit in the binary number represent sign of the number.

-> The MSB represent sign of the number.
If MSB is 1 so sign is negative.
If MSB is 0 so sign is positive

Ex. +6 -> 00110
-14 -> 11110
+24 -> 000100100

(2) Floating point Representation.

-> To accommodate very large integer and very small fractional, a computer must be able to represent numbers and operate on them in such way that the position of binary point is automatically adjusted as computatio procceds.
It is said to be floating point numbers.

-> Binary number can also be describe in exponential form. The description of binary number in exponential form is called floating point representation.

-> floating point representation have three field : Sign, Significant digits, exponent.

-> Let's consider the number 111101.1000110 to be represented in floating point format.

-> Firstly binary point is shifted to right of the first bit and the number is multiplied by correct scalling factor to get same value. It is said to be normalized form.

$$111101.1000110 \rightarrow \underbrace{1.111011000110}_{\text{significant digits}} \times \underbrace{2^{5}}_{\text{Scalling factor}}$$

Exponent

-> It is important to note that base in the scalling factor is fixed 0.

-> The string in significt digit is known as mantissa.

-> In above example we can say that,

Sign = 0

Mantissa = 111011000110

Exponent = 5

-> The standards for representing floating point numbers in 32-bit and 64-bit is represented and developed by IEEE.

-> The 32-bit standard representation also called as single precision representation.

S (1-bit)	E' (8-bit)	M (23-bit)

(Field 1) Sign ← 1-bit

(Field 2) Exponent ← 8-bit

(Field 3) Mantissa ← 23-bit

E' = E + 127. [Single precision bias is 127]

→ The 64-bit standard representation is also known as double-precision representation.

S	E'	M
1-bit	11-bit	52-bit

(Field 1) Sign ← 1-bit
(Field 2) Exponent ← 11-bit
(Field 3) Mantissa ← 52-bit

$E' = E + 1023$ [Double precision bias is 1023]

Ex Represent $(1259.125)_{10}$ in Single and double precision.

→ Step 1 : Convert number into binary

2	1259	
2	629	— 1
2	314	— 1
2	157	— 0
2	78	— 1
2	39	— 0
2	19	— 1
2	9	— 1
2	4	— 1
2	2	— 0
	1	— 0
		— 1

$0.125 \times 2 = 0.25 \rightarrow 0$
$0.25 \times 2 = 0.50 \rightarrow 0$
$0.50 \times 2 = 1.00 \rightarrow 1$

$(1259.125)_{10} = (10011101011.001)_2$

→ **Step 2: Normalize number**

$10011101011.001 = 1.001110101 1001 \times 2^{10}$

→ **Step 3: Single precision representation**

- Bias for single presion = 124

$E' = E + 127 = 10 + 127 = (137)_{10}$
$= (10001001)_2$

0	10001001	00111010 11001 ... 0
(S)	(E')	(M)

→ **Step 4: Double precision representation**

- Bias for double precision = 1023

$E' = E + 1023 = (1033)_{10}$
$= (10000001001)_2$

0	10000001001	00111010 11001 ... 0
(S)	(E')	(M)

(3) Explain Error Detection Code.

→ When digital information in binary form is transmitted from one circuit or system to another circuit or system an error may occur. It mean the signal is change 0 to 1 or vice versa due to presence of noise.

→ To maintain data integrity between transmitter and reciever extra bit or more than one bit are added in the data. This extra bit allows the detection and some time correction of error in data. Code which allow only error detection are called error detecting codes.
Which code allow error detection and correction it is called error detecting and correcting code.

→ A parity bit is used for the purpose of detecting error during transmission of binary information.

→ A parity bit is an extra bit included with binary message to make number of 1s either odd or even.

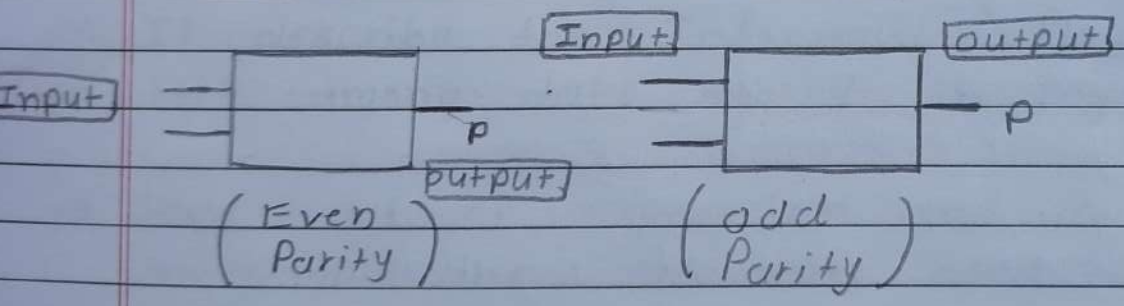
→ An error is detected if checked parity doesn't correspond with the one transmitted.

→ The circuit that genrates parity bit is called parity genrator.

→ The circuit that checks parity bit is called parity checker.

→ Truth table for odd - even parity genrator

x	y	z	P (Odd)	P (Even)
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1



→ If input in even parity genrator is odd then output is 1 and input is even then output is 0.

→ If input in odd parity genrator is even then output is 1 and input is odd then output is 0.

Chapter : 2

Register Transfer and micro-operation

(1) Explain Register Transfer language.

-> It is used to describe the micro-operation trasfer among register.

-> The term register transfer can perform micro operation and transfer the result operation to the same or other register.

-> It uses register as a primitive component in digital system instead of flip-flop and gates.

-> It describe the Information flow and processing task among data stored in register.

-> Uses set of expression and statment which reassemble the statments used in programming languages.

-> The presentation of digital function in register transfer logic is very user friendly.

→ **Micro-Operation** : It executed on the data store in register are called micro-operation. They are detailed low-level instruction used in some design to implement complex machine instructions.

→ **Register Transfer** : Information transformed from one register to another register is represent in symbolic form by replacement operator is called.

→ **Replacement Operator** : In the statment, $R_2 \leftarrow R_1$ act as a replacement operator. This statment defines the transfer of conten from R_1 to R_2.

→ **Register Transfer Operations** : The operation performed on the data stored in the register reffered to as register transfer operations.

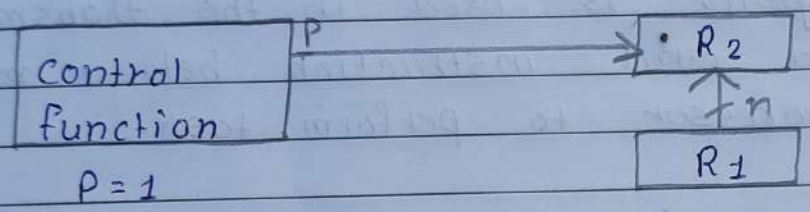
() → Denotes part of register.
← → Denotes transfer of Information.
, → Specify two micro operations
: → Denotes conditional operator
(:=) → Denotes another name for register.

(i) Simple Transfer : $[R_2 \leftarrow R_1]$

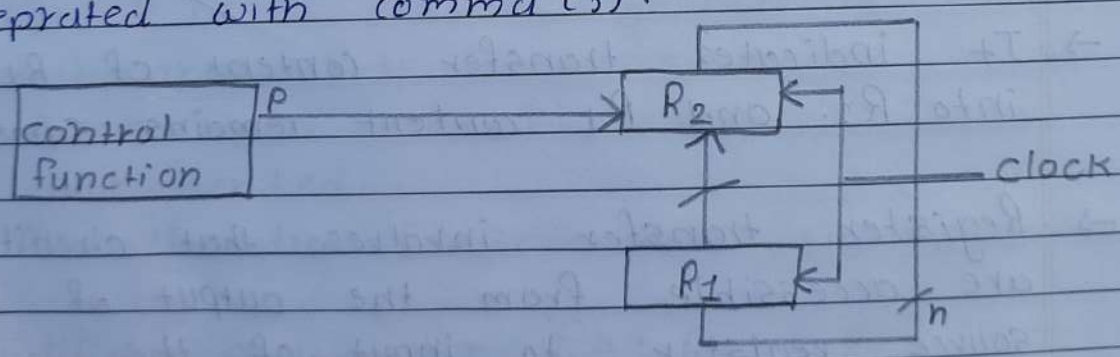
The content of R_1 are copied into R_2 without affecting content of R_1. It is an unconditional type transfer operation.

(ii) Conditional Type Transfer : $[p : R_2 \leftarrow R_1]$

It indicates if $p = 1$ then content of R_1 is transffered to R_2.



(iii) Simultaneous Operations : If 2 or more operations are occure simultaneously then they are seprated with comma (,).



$P : R_2 \leftarrow R_1 , R_1 \leftarrow R_2$

(2) Explain Register Transfer.

→ Register define the storage area that store the data and instructions.

→ It can send data and instructions from one register to another register, memory to register or vice versa the register transfer approach is used.

→ This register is used in the transmission of data and instruction between memory and processor to perform task.

→ The data transfer from one to another register is represented by symbolic notation of replacement operator.

$R_2 \leftarrow R_1$

→ It indicates transfer content of R_1 into R_2. and R_1 content remains same.

→ Register transfer involves that circuits are accessible from the output of source register to input of the destination register.

→ Genral way of representating a register is by the name of the register enclosed in rectangular box.

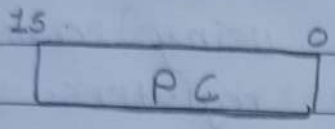
$\boxed{R_1}$

(a)

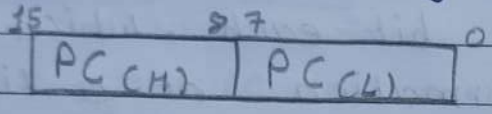
→ register is numbered in sequence of 0 to (n-1).

0	1	1	1	0	1	1	0

→ The numbering of bits in a register transfer can be marked on top.



→ A 16-bit register PC divided into 2 parts (0 to 7) are assign with lower byte and (8 to 15) are assigned as higher byte

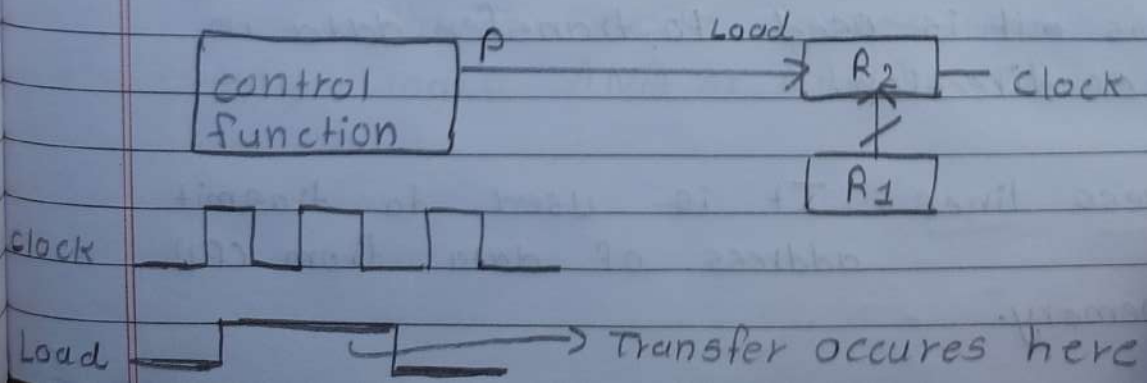


→ We need transfer to appear only under a fixed control condition using if-then statment.

if (P = 1) then ($R_2 \leftarrow R_1$)

→ Where P is a control signal created in the control area. P is either 0 or 1.
if P = 0 then transfer operation is not implemented.
if P = 1 then transfer operation is implemented.

→ Condition statment : [P : $R_2 \leftarrow R_1$]

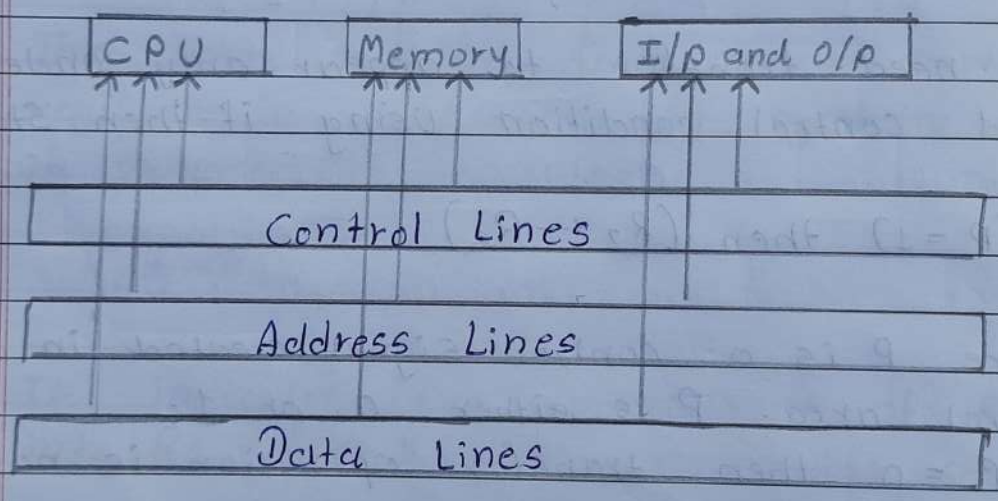


(3) What is Bus Transfer.

→ A bus transfer is most effective method to send data by using common bus system.

→ It is constructed using common bus register in multiple registers.

→ The mechanism of bus includes collection of lines. These collection of lines are register of one bit each, which share only one information at a time.



→ Data lines : Data lines provide path for moving data between system module. It is bidirectional which means it is used to transfer data in bothe direction.

→ Address lines : It is used to trasmit address of data from CPU to memory.

→ Control lines : It is used to transmit and receives control signal between the microprocessor and various devices attached to it.

→ The two method that can be used in bus transfer :
- Using multiplexer
- Using three states bus buffer

* Three-States Buffer

→ It can genrate a common bus. The buffer is an area of the memory which is added in between the other devices to block serval interaction and to connect the support. It is eshtablished on the three state 1, 0 and open circuit.

- The logic 0 and 1 are two signals similar to the ones in conventional gate.

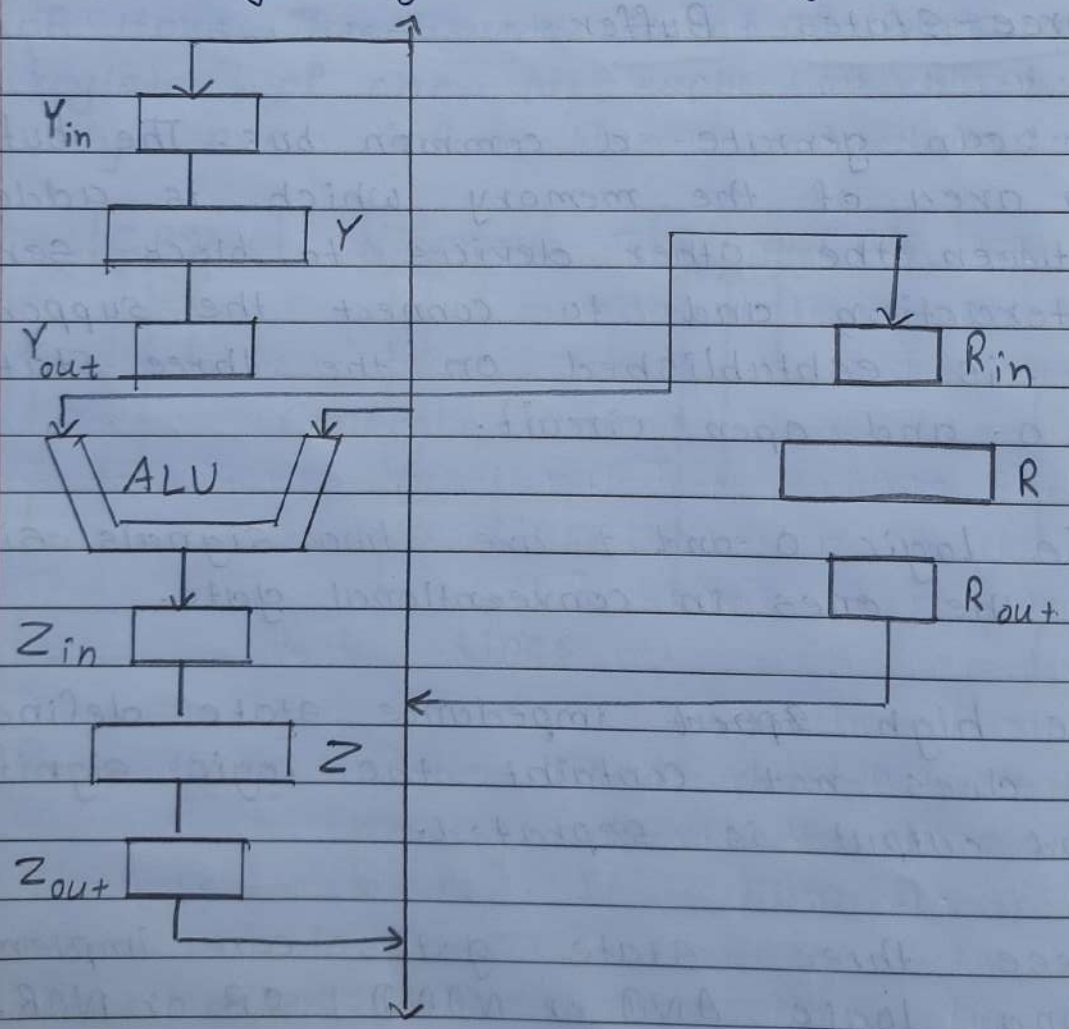
- The high ~~Spread~~ impedance state defines that it does not containt the logic significance and output is seprated.

- These three state gates can implement using logic AND or NAND, OR or NOR.

* Using Multiplexer

→ A common bus can be genrated using a multiplexer. It facilitates in choosing the source register to place the binary data on the bus.

→ The bus register has input and output gatting controlled by control signal.



-> R_i is the Register and Rin and Rout are the input - output gating signal of R.

-> Y_i is the Register and Yin and Yout are the input - output gating signal of Y.

-> Z is the Register and Rin and Zout are the input - output gating signal of Z.

-> The switches are controlled by control signal. Rin and Rout are are the input and output signal of Register R_i.
When signal is ON R_i is set to 1 and when signal is OFF R_i set to 0.

-> When the input gating Rin set to 1, the data is loaded into the register bus R_i accessible on common bus.
When Rout is set to 1 the content of register R_i places on data bus.

(4) Explain Arithmetic Micro-Operations.

→ Micro-operation is an elemantry operation performed with the data stored in register.

→ Register Transfer Micro-Operation : It transfer information from one register to another register.

→ Arithmetic Micro-Operation : That perform arithmetic operation on numeric data stored in register.

→ Logic Micro-Operation : That perform bit manipulation operation on non-numeric data stored in register.

→ Shift Micro-Operation : That perform shift operation on data stored in register.

→ Micro-operation transfer information from one register to another but it does not change the information content at the time of transfer.

→ On the other hand other three types of micro-operation change the information content.

→ The basic Arithmetic Operations are.
Add, Substract, Increment, Decrement,

- Add ⇒ $R_3 \leftarrow R_1 + R_2$ = Content of R_1 and R_2 are added and transfer to R_3

- Substract ⇒ $R_3 \leftarrow R_1 - R_2$ = Content of R_2 are substract from R_1 and transfer to R_3.

- 1's Complement ⇒ $\overline{R_1}$ = Complement the content of R_1.

- 2's Complement ⇒ $\overline{R_1} + 1$ = Complement the content of R_1 and add 1 in it.

- Increment ⇒ $R_1 \leftarrow R_1 + 1$

- Decrement ⇒ $R_1 \leftarrow R_1 - 1$

(5) Explain binary adder and substractor.

→ The substraction of binary numbers can be done most conviniently by means of complements.

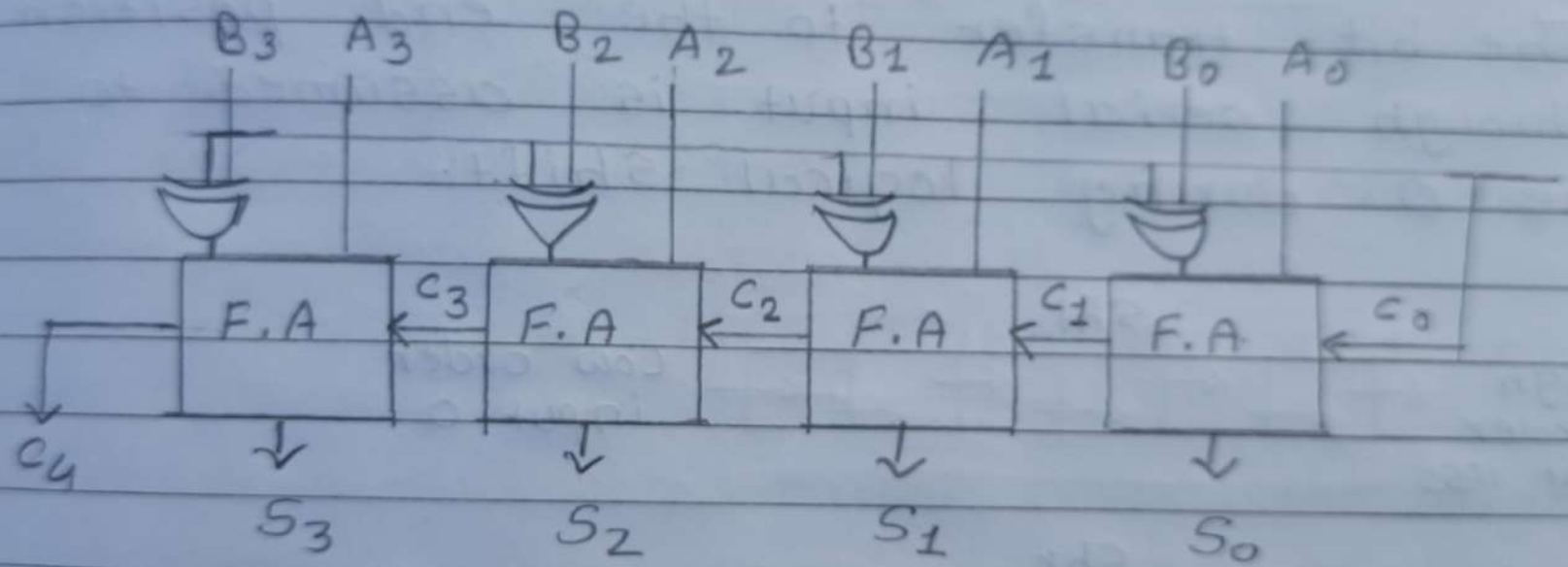
→ Remember substraction A-B can be done by taking 2's complement of B and adding it to A.

→ The 2's complement can be obtain by taking 1's complement and adding one. The 1's complement can be implemented by inverters and one can be added to sum throughout the input carry.

→ The addition and substraction are combined into one common circuit including an exclusive-OR gate with each full adder.

→ The Mode input [M] control the operation.
When, M = 0 → Adder
M = 1 → Substractor

→ Each exclusive-OR gate recives input M and one of the input B.
When, M = 0
$C_0 = 0$
$B \oplus 0 = B$
When, M = 1
$C_0 = 1$
$B \oplus 1 = B'$



→ Unsigned number

If, $A >= B$ then $A - B$

If, $A < B$ then $B - A$.

Signed number

$A - B$ provided there is no overflow.

(6) Explain Shift Micro-Operation and draw 4-bit combinational circuit.

* Logical - Shift

-> A logical shift one that transfer 0 through the serial input. we will adopt the symbol [shl] for logical shift-left and [shr] logical shift-right micro-operation.

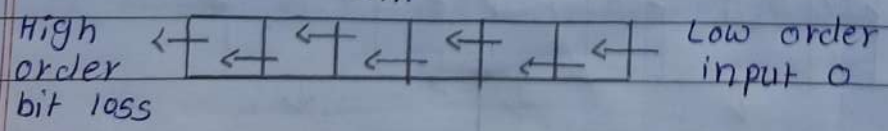
$R_1 \leftarrow shl\ R_1$
$R_2 \leftarrow shr\ R_2$

-> Here, two micro-operation that specify a 1-bit shift to the left of the content of register R_1 and a 1-bit shift to the right of the content of register.

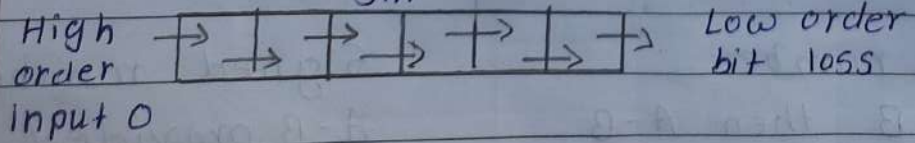
-> The register symbol must be same for both side of the arrow.

-> The bit transfer to the end position through serial input is assumed to be 0 during logical shift.

shl

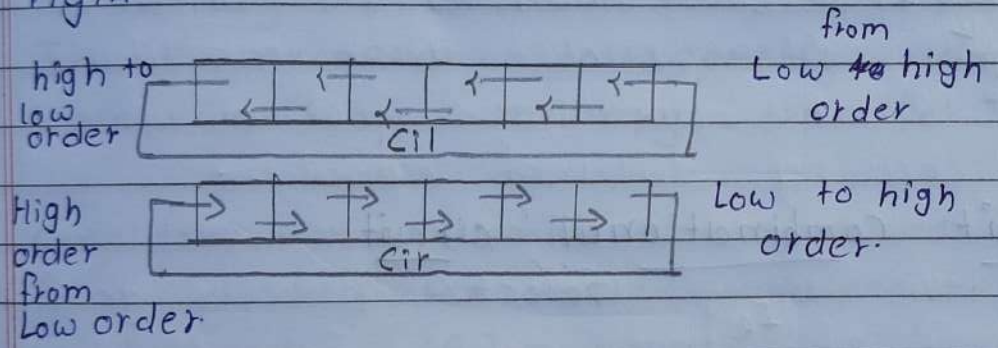


shr



* Circular - Shift

-> It is also known as rotate operation.
-> It circulates bits of register around two end. without loss information.
-> This is implemented by connecting the serial output of shift register to its serial input.
-> We will use symbol [Cil] for circular shift left and [Cir] for circular shift right.

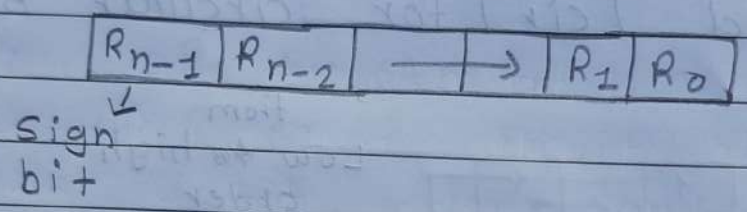


* Arithmetic - Shift

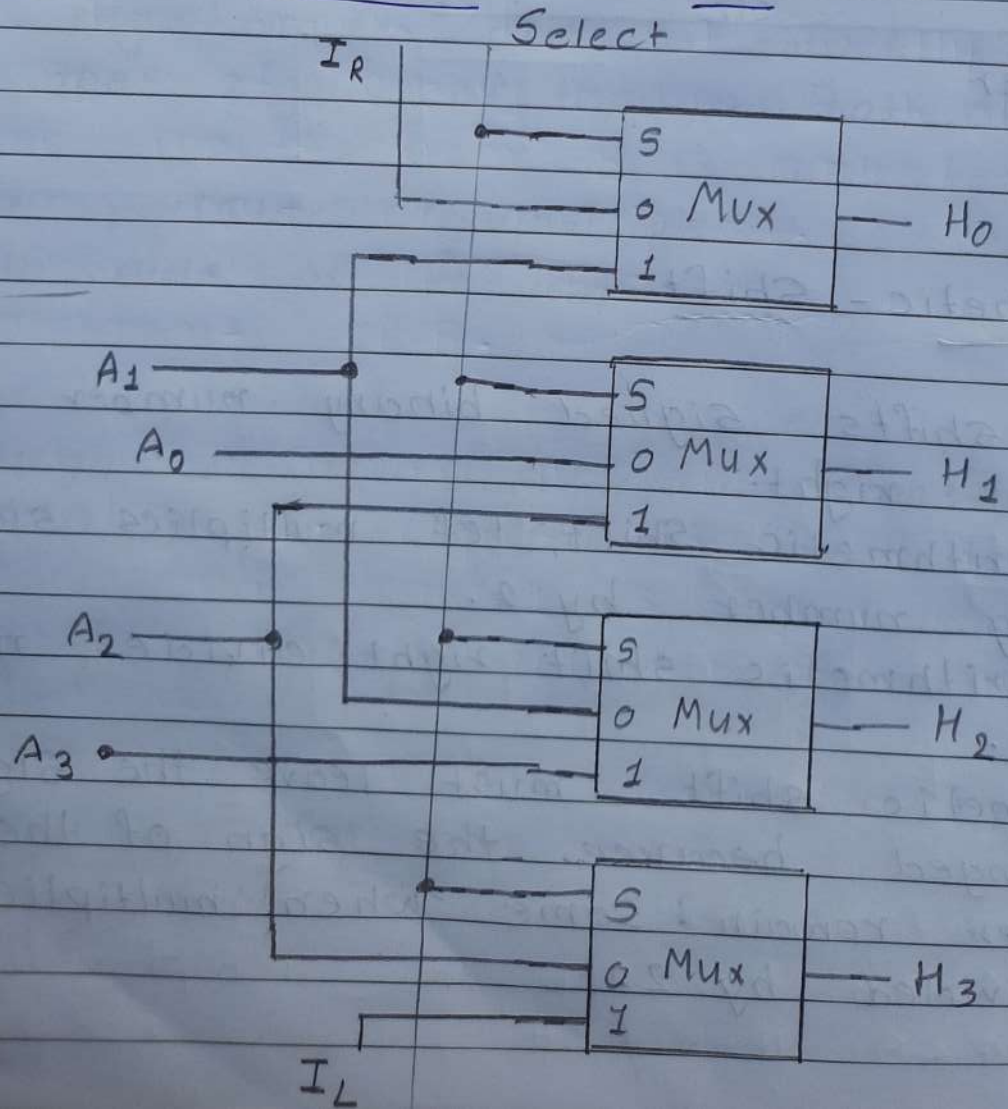
-> It shifts signed binary number to the left or right.
-> An arithmetic shift lef multiplies signed binary number by 2.
-> An arithmetic shift right divides number by 2.
-> Arithmetic shift must leave the sign bit unchanged because the sign of the number remain same when multiplied or divided by 2.

→ The left most bit in register hold sign bit unchanged because sign of the number remains remains same and other bit holds number.

→ The sign bit 0 for positive and 1 for negative.



* 4-bit Combinational circuit



(7) Explain memory transfer.

→ The transfer of information from a memory word to the outside enviroment is called read operation.

→ The transfer of new information to be stored in the memory called write operation.

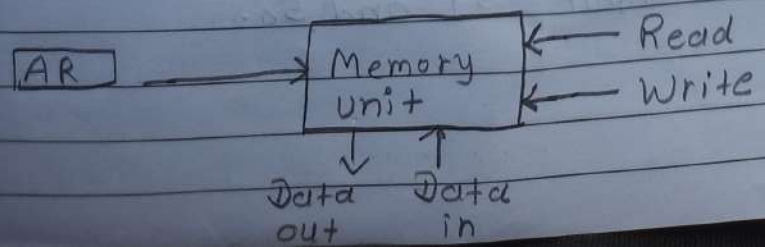
→ A memory word will be symbolic by [M].
→ It is necessary to specify address M when writing memory transfer operation.

→ Consider memory unit that recives address from a register called address register, Symbolic by [AR].

→ The data transfer to another register called data register, Symbolic by [DR].

→ Read : DR ← M [AR]
Transfer information into DR from the memory word M selected by address in AR.

→ Write : M [AR] ← R_1
Transfer the information from R_1 into memory word M selected by address AR.



(8) Explain 4-bit Arithmetic circuit.

→ Arithmetic micro operation can be implemen in one composite arithmetic circuit.

→ The basic component of arithmetic circuit is parallel adder.

→ By controlling data inputs to adder it is possible to obtain different micro operation.

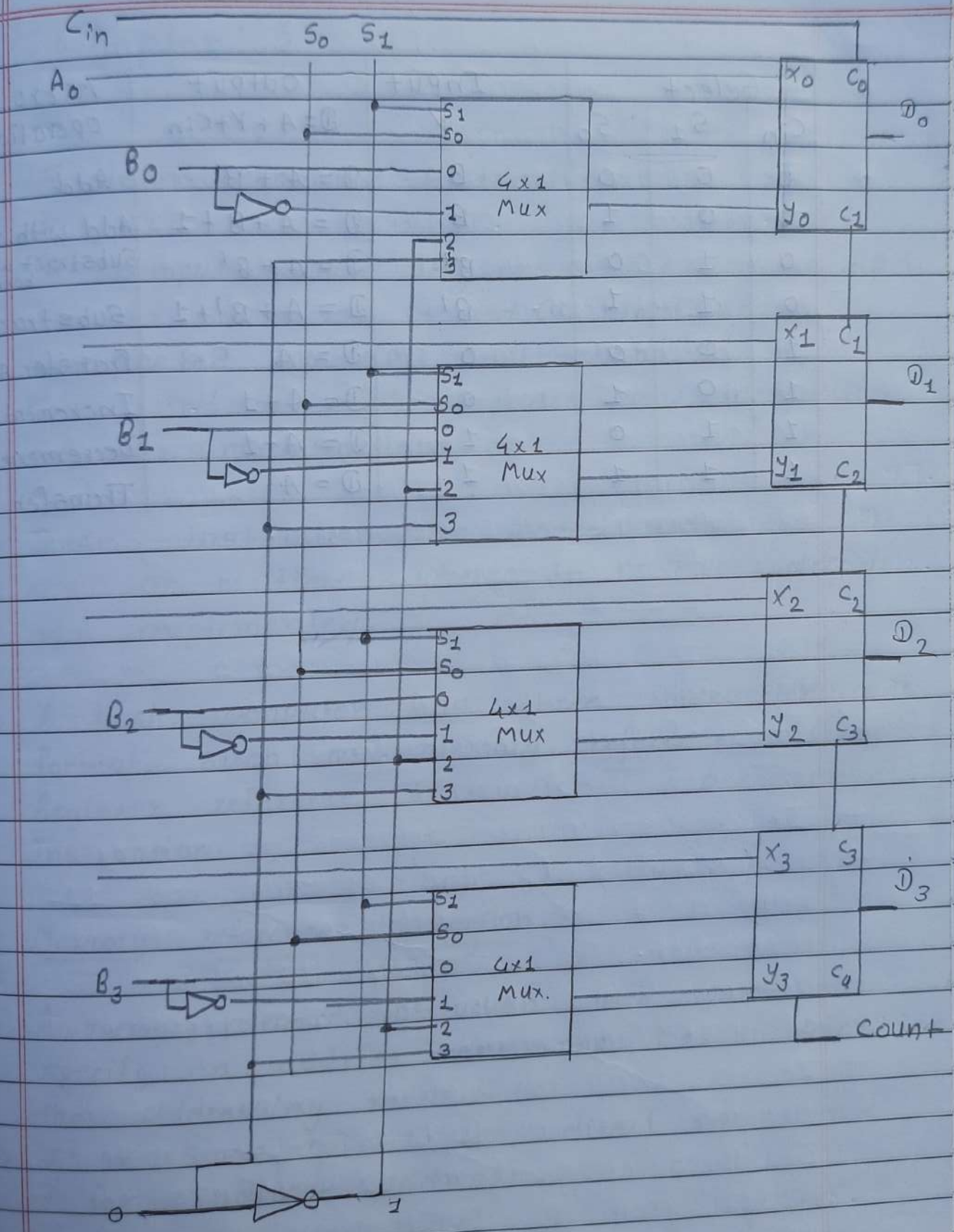
→ Hardware implementation consist of:

- 4-full adder circuit that constitute the 4-bit adder and four multiplexer for choosing different operation.

- The four inputs from A directly go to the X input of binary adder. Each of the four inputs from B is connected to data input of multiplexer. multiplexer's data input also recieve complement of B.

- Other two inputs are connected to logic-0 and logic-1. Logic-0 is fixed voltage value and logic-1 is signal can be genrated through invertor whose input is 0.

- Four Multiplexer are controlled by two selection input S_1 and S_0.


Cin
S0 S1
A0
B0
B1
B2
B3
4x1 Mux
4x1 Mux
4x1 Mux
4x1 Mux.
X0 C0
Y0 C1
X1 C1
Y1 C2
X2 C2
Y2 C3
X3 C3
Y3 C4
D0
D1
D2
D3
Count
0
1

Select			Input	output	Micro
Cin	S_1	S_0	Y	D = A + Y + Cin	operation
0	0	0	B	D = A + B	Add
0	0	1	B	D = A + B + 1	Add with carry
0	1	0	B'	D = A + B'	Substract with borrow
0	1	1	B'	D = A + B' + 1	Substract
1	0	0	0	D = A	Transfer A
1	0	1	0	D = A + 1	Increment A
1	1	0	1	D = A - 1	Decrement A
1	1	1	1	D = A	Transfer A

Chapter : 3

(1) Explain computer instruction.

-> A computer has programs store in its RAM in the form of 1s and 0s that are interpreted by CPU instruction.
One word of RAM includes one instruction in the machine language.

-> These instruction are loaded into the CPU one at a time, where it recieves decode and implemented.

-> A basic computer has three instruction code format such as memory reference instruction, Register reference instruction, I-O reference instruction.

* Memory reference instruction

-> A memory-reference instruction uses 12-bit to specify an address and one bit to determine the addressing mode. I.
-> I is same as 0 for direct adress and 1 for Indirect address.

15	14 ... 12	11 ... 0
1	Opcode	Address

opcode = 000 through 110

* Register reference Instruction

→ The register reference instruction are Identified by operation code 111 with a 0 in the left most bit of the instruction.
It determines an operation On or test of the AC register.
An operand from memory not required because additional 12 bits are used to determine the operation.

15 — 12	11 — 0
0 1 1 1	Register operation

Opcode = 111, I = 0

* Input - Output Instruction

→ It does not require reference to memory and is identified by the operation code 111 with a 1 in the left most bit of the instruction. The remaining 12 bit can be determine the type of I-O operation or test implemented.

→ It is identified by the computer control from the four bits in 12 to 15 of the instruction. If three opcode from 12 to 14 are not similar to 111 then the instruction is memory type reference instruction. and bit 15 is taken as the addressing mode I.

15 — 12	11 — 0
1 1 1 1	I/o Operation

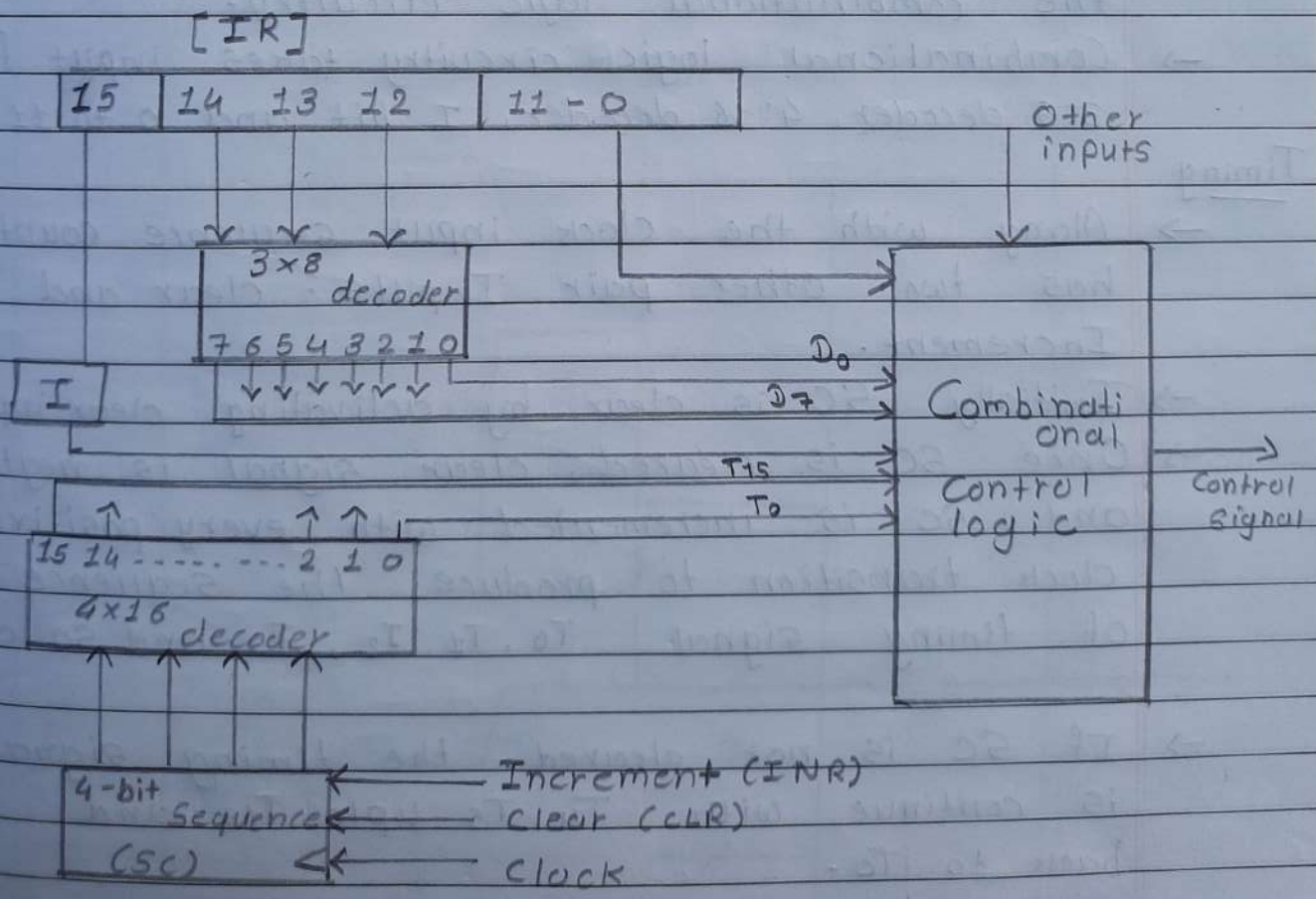
opcode = 111, I = 0

(2) Explain Timing and Control Unit.

→ Components of control units are
(i) two decoder (3 x 8, 4x16)
(ii) IR
(iii) Sequence counter.
(iv) Control logic gate.

→ An instruction read from memory placed in the IR. In control unit the IR is divided into three parts : I bit (15), operation code (12-14) bit, bits (0 to 11).

→ The operation code in bits 12 to 14 are decoded with 3x8 decoder.



-> Instruction Register [IR] used to genrate control signals. The Instruction in IR is divided into three part :
I code, operation bit and bit 0 to 11.

-> The 3:8 decoder decodes the Operation code bits and output of decoder are used as an input of combinational control logic.

-> The four bit sequence counter count from 0000 to 1111. These four bits are decoded by 4:16 decoder to genrate timing signals from 0 through 15.

-> The I bit from IR is also connected to the combinational logic circuitry.
-> Combinational logic circuitry takes input from 3:8 decoder, 4:16 decoder, I bit and 0 to 11 bits.

Timing

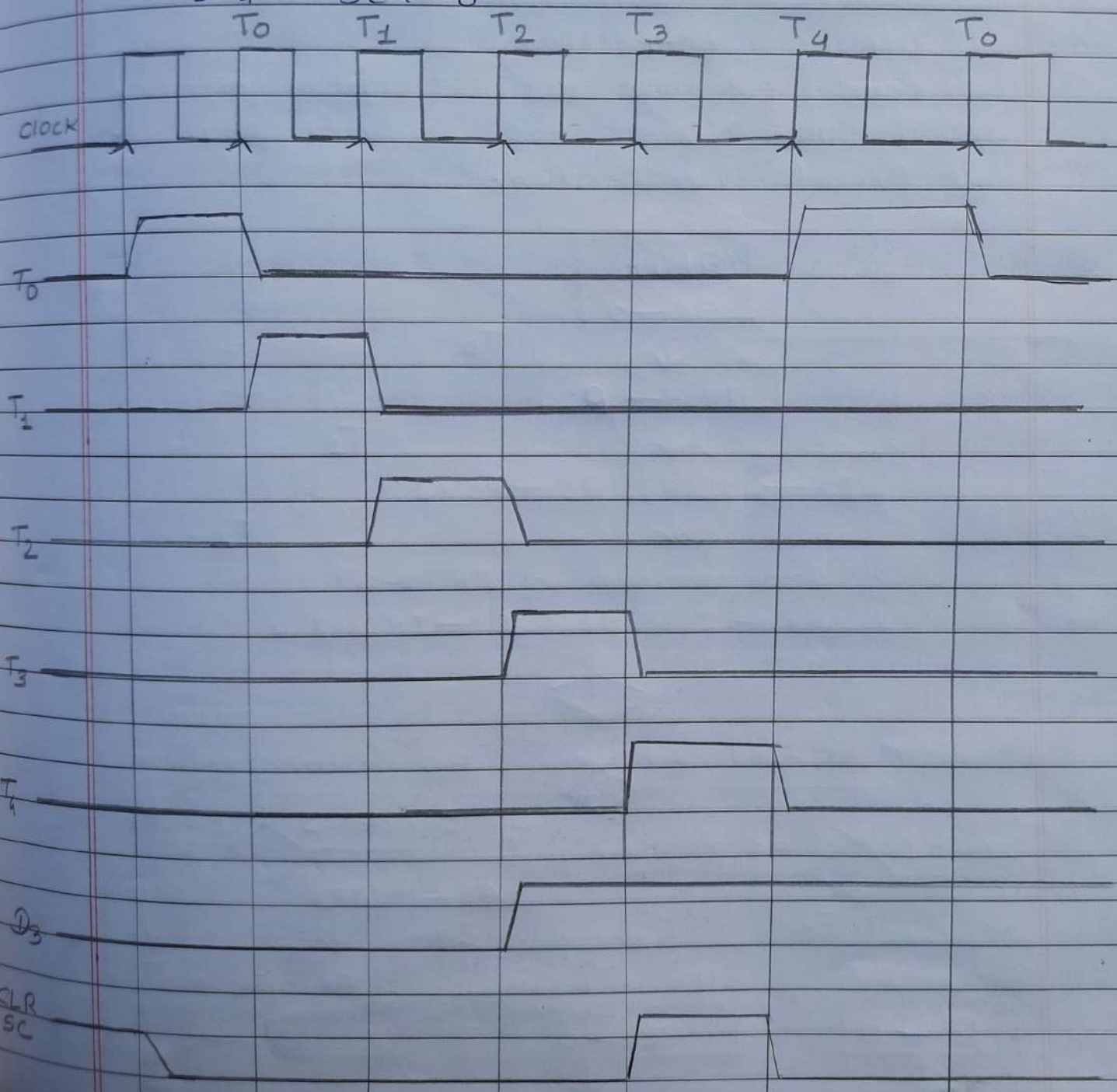
-> Along with the clock input sequence counter has two other pair Inputs : clear and Increment.
-> Initially SC is clear by activating clear input.
-> Once SC is cleared, clear signal is negted and SC is incremented with every positive clock transition to produce the sequence of timing signal T_0, T_1, T_2, T_3 and so on.

-> If SC is not cleared the timing signal is continue with T_4, T_5 upto T_{15} and back to T_0.

→ However, If SC cleared in between a new sequence will start from T_0, T_1 and so on.

→ As shown in figure SC is cleared 0 in T_4 if decoder output D_2 is active (logic : 1) This can be expressed symbolic as

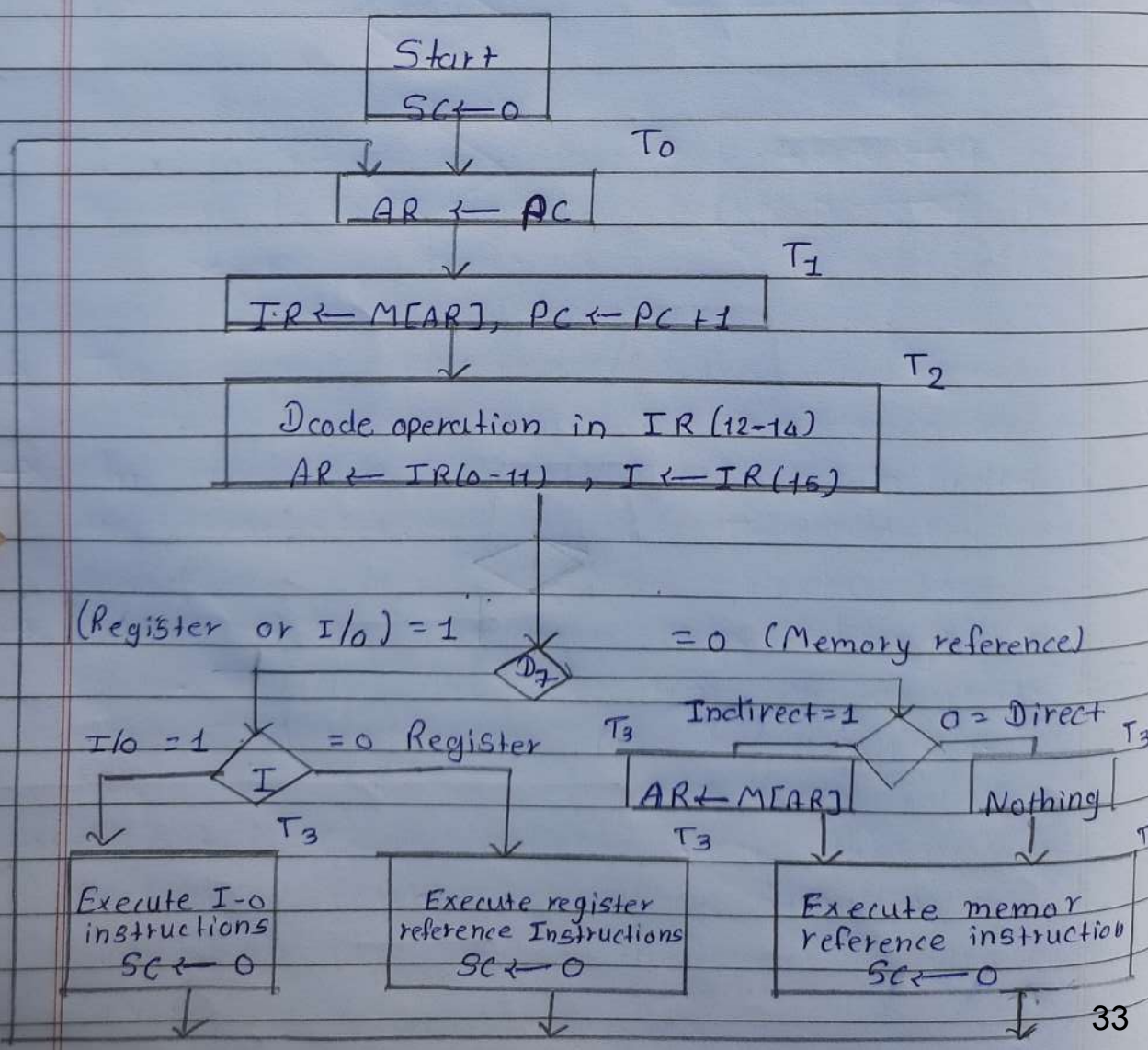
$$D_2 T_4 : SC \leftarrow 0$$



(3) Explain Flowchart for Instruction Cycle.

→ A program residing in the memory unit of the computer consist of a sequence of instruction. In instruction cycle include phases:

(i) fetch an instruction from memory
(ii) Decode instruction
(iii) Read effective address from memory if instructions has indirect address.
(iv) Execute instruction.



-> The flowchart present an initial configuration for the instruction cycle and show how the control determines the instruction type after the decoding.

-> First step we fetch instruction from memory and set $SC \leftarrow 0$ then decode the instruction.

-> If $D_7 = 1$ then instruction must be register reference or input-output reference.

-> If $D_7 = 0$ then instruction must be memory reference.

-> If $D_7 = 0$ and $I = 1$ then we have memory reference instruction with Indirect address. It is necessary to read effective address from memory. set $SC \leftarrow 0$

-> If $D_7 = 0$, and $I = 0$ then we have not to do anything since effective adress is already in AR. set $SC \leftarrow 0$.

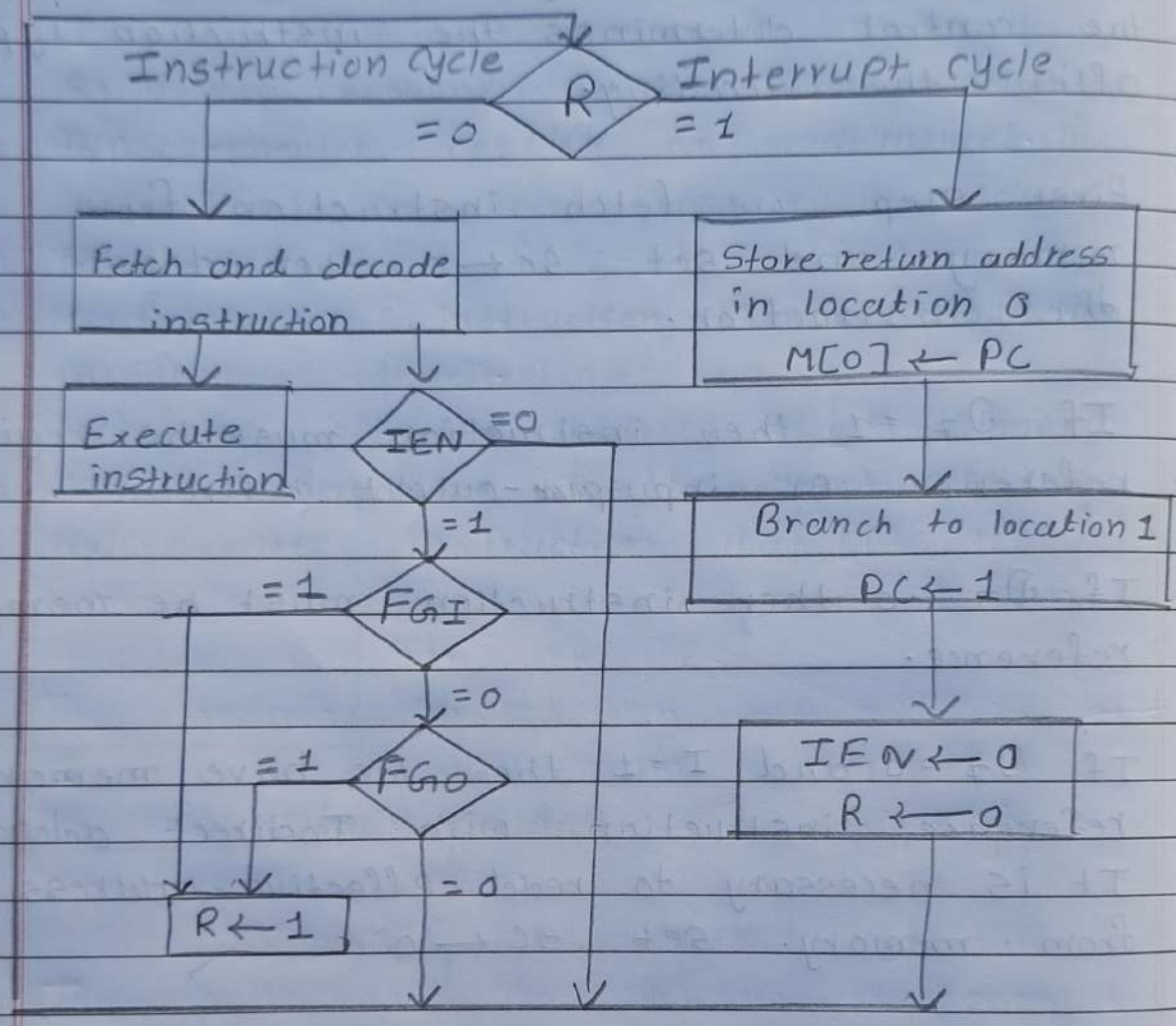
-> If $D_7 = 1$ and $I = 1$ then it must be I-O instruction we have to execute I-O instruction and set $SC \leftarrow 0$.

-> If $D_7 = 1$ and $I = 0$ then it must be register reference instruction we have to execute instruction and set $SC \leftarrow 0$.

-> After Step-4 control goes back to Step 1 to fetch, decode and execute next instruction.

(4) Explain Interrupt Cycle.

→



→ An interrupt flip-flop R is included in the computer.

→ When R = 0 computer goes through instruction cycle. During the execution phase instruction cycle IEN checked by the controller.

→ If it is 0, it indicates that the program does not want to use interrupt so control continues with the next instruction cycle.

-> If IEN = 1 control checks the flag bits.

-> If both flags are 0, it indicates neither input nor the output register are ready for transfer information.

-> If either flag is set to 1 while IEN = 1 flip-flop R is set to 1.

-> At the end execute phase control checks the value of R and if it is equal to 1 then it goes to an interrupt cycle instead of instruction cycle.

* Interrupt cycle.

-> The interrupt cycle is a hardware implementation of a branch and save return address operation.

-> Here we choose memory location at address 0 as the place for storing the return address.

-> Control then insert address 1 into PC and clear IEN and R so no more interuption can occure untill the interruption request from the flag has been serviced.

-> Suppose that an interrupt occurs and R = 1 while the control is executing instruction at address 255. At this time return address is 256 in PC.

-> The programmer has previously place an I-O service programs in memory starting from address 1120 and BUN 1120 instruction at address 1.

-> The content of PC(256) is stored in memory location 0, PC is set to 1, and R is clear to 0.

-> At the begining of the next instruction cycle. the instruction that is read from memory is in address 1 since this is content of PC.
The branch instruction at address ~~one~~ 1 causes the program to transfer to the Input-output service program at address 1120.

-> The instruction that returns the computer to the original place in the main program is a branch indirect instruction with an address part of 0. This instruction is placed at the end of the Ilo service program.

-> The execution of Indirect BUN Instruction result in placing into PC the return address from location 0.

Chapter : 4

Assembly Language

(1) Explain assembly language and Rules.

→ The symbolic program which contains letter, numerals or special character is known as an assembly language program.

→ The basic unit of assembly program logic is line of code.

→ The specific language is defined by set of rules that specify that the symbol can be used and they may be combined to form ~~of~~ a line of code.

* Rules

→ Each line of assembly language program is arranged in three field.
(i) Lable
(ii) Instruction
(iii) Comment

(i) The Lable field may be empty or it may specify symbolic address.
- A symbolic address does not contain more than three alphnumeric character.
- The first character must be letter, the next two may letter or numerals.

- A symbolic address in the lable field is terminated by comma so that it will be recognized as a lable by assembler.

(ii) The instruction field specifies a machine instruction or pseudo instruction.
The instruction field in assembly language program may specify one of the following iteams.
(1) Memory-reference Instruction [MRI]
(2) Register-reference or Input-Output Instruction [Non-MRI]
(3) Psedo Instruction with or without operand.

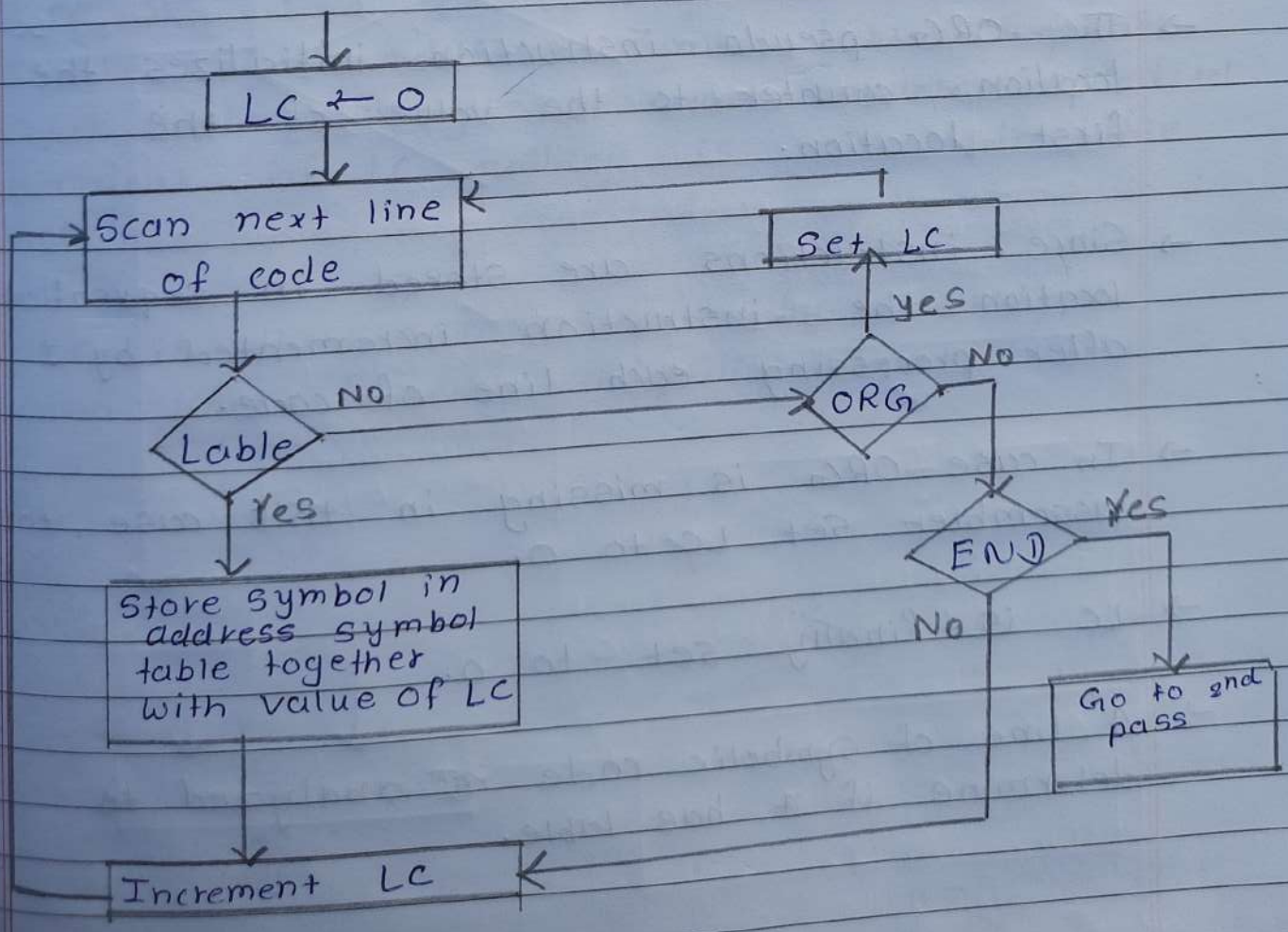
(iii) The comment field may be empty or include comment.
- A line of code may or may not have comment, but if has it must be started by slash for the assembler to recognize the comment field.
- Comments are useful for explaning a program and understanding step by step procedure taken by program.
- Comments are neglected during binary translation process.

(2) Define Assembler and Explain first pass.

→ An assembler is a program that accept a symbolic language program and produce it's binary machine language equivalent.

→ The input symbolic program is called the source program and resulting binary program is called objective program.

→ The assembler is a program that operates on character string and produce an equivalent binary interpretation.



-> During the first pass it genrates table that correlates all the user-defined adress symbol. with their binary value.

-> The binary translation done during second pass.

-> To keep track all of the location of instruction the assembler use memory word called a Location counter (LC).

-> The content of LC store the value of the memory location that assigned to the instruction.

-> The ORG pseudo instruction initializes the location counter to the value of the first location.

-> Since instructions are stored in sequential location the instruction incremented by 1 after processing each line of code.

-> In case ORG is missing in this case, the assembler set LC to 0.

-> LC is finally set to 0.

-> A line of symbolic code is analyzed to determine if it has lable.

-> If the line of code has no lable, the assembler checks the symbol in the instruction field.

-> If it contains an ORG pseduo instruction the assembler set LC to the number that follows ORG and goes back to process the next line.

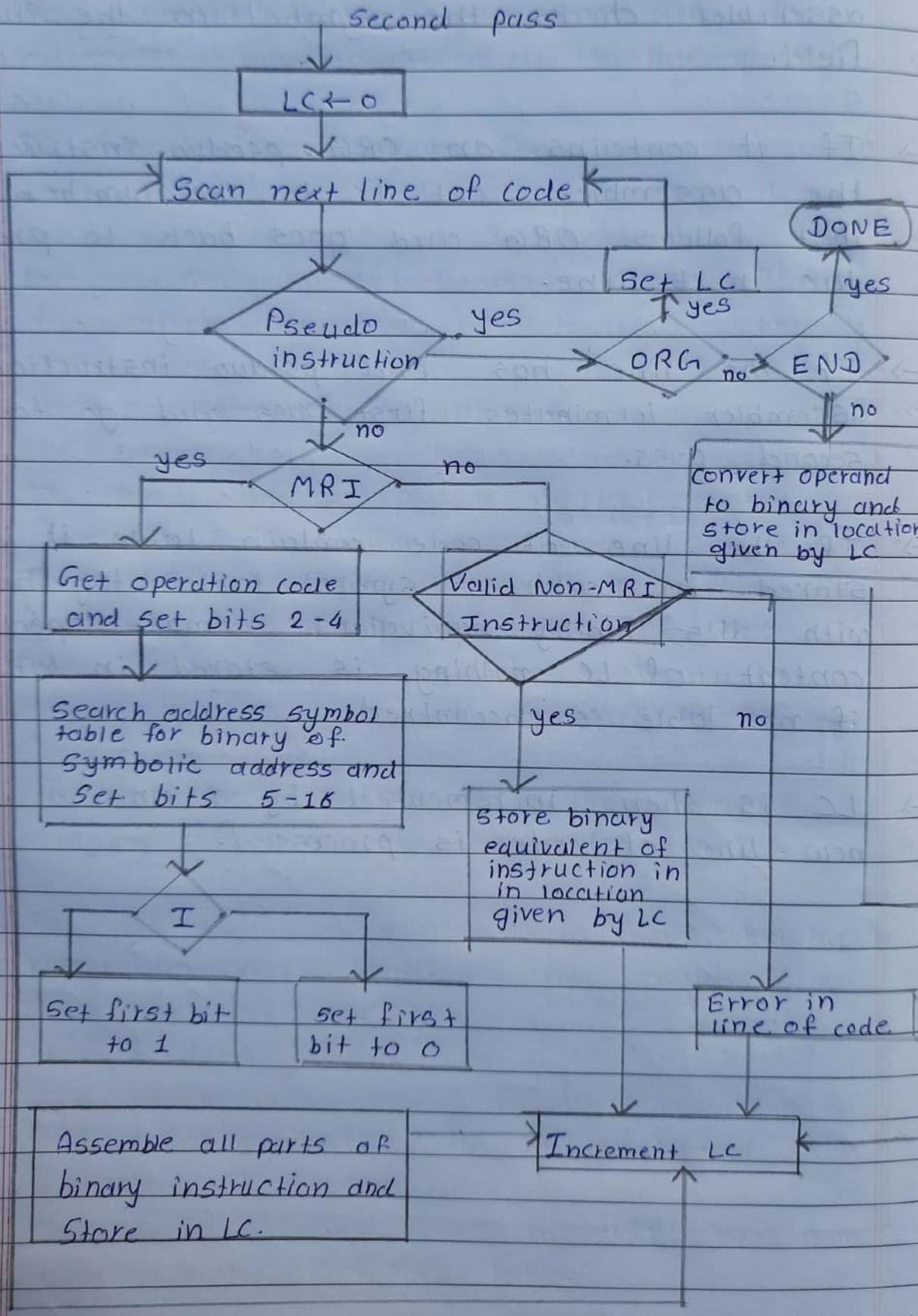
-> If the line has end pseduo instruction, the assembler terminates first pass and go to second pass.

-> If the line of code contain lable it is stored in address symbol table together with it's binary equivalant number specified content of LC nothing is stored in table if no lable is encountred.

-> LC is then incremented by 1 and a new line of code is processed.

(3) Explain Second pass.

→



-> Machine instruction are translated during second pass by means of table look-up procedures.

-> The assembler uses four table. Any symbol that is encountered in program must be avaleible as entry in one of these tables; Otherwise symbol can not be interpreted.

(a) Pseudo instruction table
(b) MRI table
(c) Non-MRI table
(d) Address symbol table.

-> The entries of the pseudo instruction table are the four symbol ORG, END, DEC and HEX.

-> Each entry refers to assembler to a subroutine that processes the pseudo instruction when encountered in program.

-> The MRI table contains the seven symbol of the memory reference instruction and their 3-bit operation code equivalent.

-> The non-MRI table contains the symbol of 18 register reference and Input-output instruction and their 16-bit binary code equivalent.

-> The address symbol table genratesd during first pass of assembly process.

-> The assembler searches these table to find the symbol that it is currently processing in order to determine its binary value.

-> LC is finally set to 0.
Lines of code that analyze one at a time.

-> Lables are neglacted during the 2nd pass so the assembler goes immediatley to the instruction field and proceeds to check the first symbol encountered.

-> It first checks Pseudo instruction table
A match with ORG sends assembler to the sub routine that sets LC to its initial value.
A match with END terminates the translation process. An operand in Pseudo instruction causes a conversion of operand to binary.

-> This operands is placed in the memory location specified by the content of LC.

-> The location counter is then incremented by 1 then assembler continues the analyze the next line of code.

-> If symbol encountered is not pseudo instruction then assembler refers to MRI table.

-> If symbol is not found in MRI table then it refers to Non-MRI table.

-> A symbol found in non-MRI table corresponds to a register reference or Input-Output instruction.

-> The assembler store 16-bit instruction into memory word specified by LC.

-> The Location counter is incremented and new line is analyzed.

-> When symbol finds in MRI table, then assembler extracts its equivalent 3-bit code.

-> A memory refrence instruction specified by two or three symbol. The symbolic address is converted to binary by searching address symbol table.

-> The three parts of the binary instruction codes are assembled and store in memory location specified by LC. then LC incremented an continue process.

-> One important task of an assembler is to check for possible errors in the symbolic program. This is called error dignostics.

-> One such error may be an invalid machine code symbol which is detected by its being absent in MRI and non-MRI table.

-> The assembler can't translate such symbol because it does not know its binary value. In such case the assembler print error message to inform the programmer that program has an error at specific line of code.

Chapter: 5

Microprogrammed Control Organization.

(1) Explain Address Sequencing.

-> Micro instructions are stored in control memory in groups with each group specifying a routine.

-> To appreciate the address sequencing in a micro-program control unit Let us specify the steps that control must undergo during execution.

* Step: 1

-> An initial address is loaded into the control address register when power is turned on in the computer.

-> These address is usually address of first microinstruction that activates the instruction fetch routine.

-> The fetch routine may be sequenced by incrementing the control address register through the rest of its microinstruction.

-> At the end of the fetch routine, the instruction is in the instruction register of the computer.

* Step : 2

-> The control memory next must go through the routine that determinates the effective address of operand.

-> A machine instruction may have bits that specify various addressing mode such as indirect addressing and index register.

-> The effective address computation routine in control memory can be reached through branch micro-instructions.

-> When the effective address computation routine is completed the address of the operand is avalaible in memory address.

* Step : 3

-> The next step to genrate micro-operation that execute the instruction fetched from memory.

-> The microoperations step to be genrated in processor register depends on the operation code part of the instruction.

-> Each instruction has its own micro program routine stored in given location of counter memory.

→ The transformation from the instruction code bits to an address in control memory where the routine is located reffer to as mapping process.

→ A mapping procedure is a rule that transforms the instructions code into control memory address.

* Step : 4

→ Once the required routine is reached the micro instruction that executes the instructions may be sequenced by incrementing the control address register.

→ Micro program that employes sub routine will require an external register for storing return address.

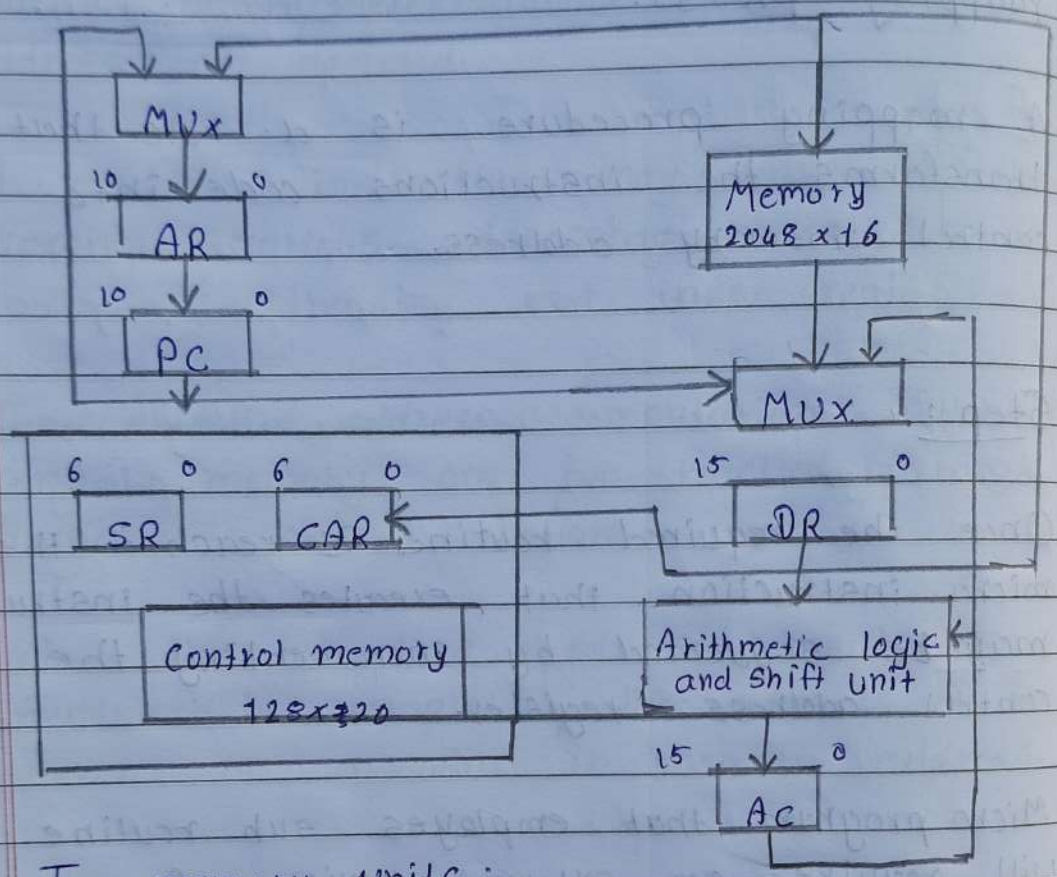
→ Return address can't be stored in ROM because unit has no writing capability.

→ When the execution of instruction is completed, control must return to fetch routine.

→ This is accomplished by executing an unconditional branch micro instruction to the first address of the fetch routine.

(2) Draw and explain Computer Hardware configuration.

→



→ Two memory units :
Main memory → for storing instruction and data
Control memory → for storing the micro program

→ Six Registers :
Processor unit Register : AC , PC , AR , DR
Control unit Register : CAR , SBR

→ Multiplexer : The transfer of information among the registers in the processor is done through multiplexer.

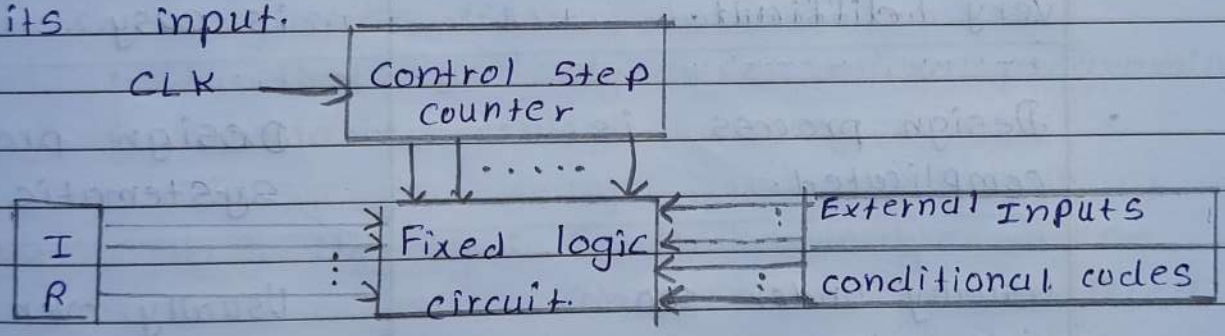
→ ALU : The arithmetic logic and shift unit performs micro operations with data from AC and DR and places the result in AC.

(3) Explain Hardwired Control.

→ In the hardwired control the control unit uses fixed logic circuit to interpreted instructions and genrate control signals from them.

→ The fixed logic circuit uses content of the control step counter, content of the instruction register, content of condition code flag and external input signal such that MFC and interrupt requests to genrate control signals.

→ Here the fixed logic circuit block includes combinational circuit that genrates the required control output depending on state of all its input.



* Advantage : It is fast because control signals are genrated by combinational circuit.
  - It has great chip area efficiency.

* Disadvantage : When more control signals required by CPU, more complex will be design of control unit
  - Modification in control signals are difficult.
  - Delay in genration of control signals upon number of gates.

(4) Comparision between Hardwired and Microprogram control.

Hardwired	Microprogram
• Implemented in hardware.	Implemented in software.
• Speed is fast.	Speed is slow.
• Not flexible	More flexible
• Ability to handle is difficult	Ability to handle is easy.
• Ability to Support is very difficult.	Ability to Support is easy
• Design process is complicated.	Design process is Systematic
• Usually under 100 instruction	Usually over 100 instruction.
• Uses less chip area.	Uses more chip area

(5) Define following terms

→ Hardwired control Unit : When the control signals are genrated by hardware using conventional logic design technique the control unit is said to be hardwired.

→ Microprogramed control unit : A control unit whose binary control variable are stored in memory is called micro program control unit.

→ Dynamic Microprograming : A more advanced devlopment known as microprograming permits a micro program to be loaded initially from an auxilary memory such as magnetic disk.
Control unit that use dynamic micro programing employ a writable control memory.

→ Writable control Memory : Control storage whose content can be modified allow the change is microprogram and instruction set can be changed or modified is known as writable control memory.

→ Control Memory : It is storage in the micro-programmed control unit to store microprogram.

→ Control word : The control variables at any given time can be represented by control word string of 1's and 0's called control word.

→ **Micro Operations :** In computer CPU, ~~each~~ micro operation are detailed low-level instruction used in some design to implement complex machine instructions.

→ **Micro Instruction :** A symbolic micro program can be translated into binary by means of assembler.

- Each line of the assembly language microprogram defines symbolic micro instruction.
- Each micro instruction divided into five field : lable, micro operation, CD, BR, AD

→ **Micro Program :** A sequence of microprogram instructions constitute microprogram.

- If the control unit is in operation the control memory can be ROM.
- ROM words are made permanant during the hardware production unit.
- The content of word ROM at a given address specifies a micro Instruction.

→ **Micro Code :** Micro instruction can be saved by employing subroutines that use common section of micro code.

- The sequence of micro operation needed to genrate the effective address of operand for instruction is common in all memory reference instruction.

Chapter: 6

Central Processing Unit

(1) What is Stack Organization? Explain Register and Memory Stack.

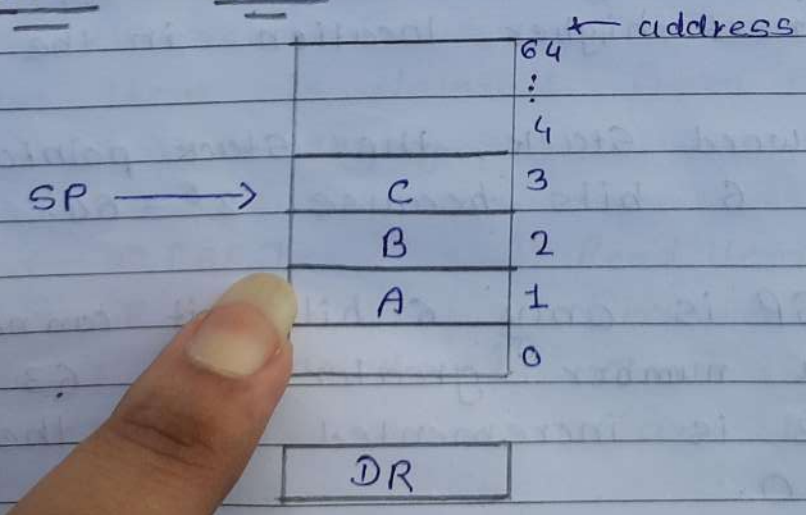
→ A stack is a storage device that stores information in such manner that the item stored last ~~in~~ and item retrieved first.

→ Stack in digital computers is essentially a memory unit with an address register that can count only. The register that holds the address for stack is called a stack pointor.

→ The value of stack pointor (SP) is always points at the top item in the stack.

→ The physical register of stack are always avalaible for reading or writing.

* Register Stack.



-> A stack can be placed in a portion of a large memory or it can be organize as a collection of a finite number of memory word or register.

-> The SP contains a binary number whose value is equal to the address of the word that is currently on top of the stack.

-> Three item placed in stack : A, B and C in that order. Item C is top of the stack so that the content of SP is now 3.

-> To remove top of the item the stack is popped by reading the memory word at address 3 and decrementing the content of SP. Item B is now on top of the stack since SP hold address 2.

-> To insert a new item the stack is pushed by incrementing SP and writing word in the next-higher location in the stack.

-> In 64-word stack, the stack pointor contains 6 bits because $2^6 = 64$.

-> Since SP is only 6 bits, it can not exceed a number greater than 63 When 63 is incremented by 1 the resut is 0.

-> Similarly when 000000 is deremented by 1 the result is 111111. The one bit register is FULL is set to 1 when stack is full. and register EMTY set to 1 when stack is empty of items.

-> DR is data register that holds binary value to be written into or read out of the stack.

=> **PUSH:**

- If Stack is not full (FULL = 0) a new item is inserted with a push operation. PUSH operation consist of following micro operation.

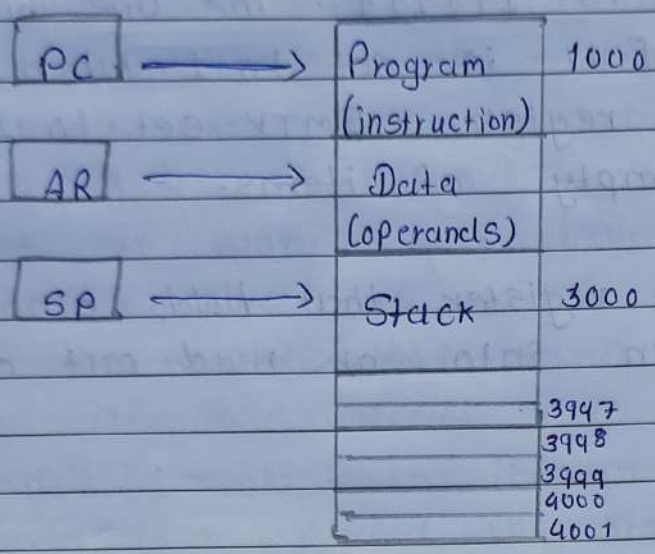
SP ← SP + 1	Increment stack pointor.
M[SP] ← DR	Write item on top of the stack.
IF(SP = 0) then (FULL ← 1)	Check stack is full
EMTY ← 0	Mark the stack not empty

=> **POP:**

- A new item is deleted from the stack if the stack is not empty.

DR ← M[SP]	Read item on top of stack
SP ← SP - 1	Decrement stack pointor
IF(SP = 0) then (EMTY ← 1)	Check if stack is empty
FULL ← 0	Mark the stack not full

* Memory Stack



-> The implementation of stack in the CPU is done by assigning portion of memory to a stack operation and using a processor register as SP.

-> Portion of memory divided into three parts: program, data and stack.

-> The program counter PC points to address of next instruction in the program.

-> The stack pointor SP points to top of the stack which is used for push or pop item from stack.

-> Three register are connected to a common bus either one can provide an address for memory.

-> We assume that the item in the stack communicate with data register DR.

⇒ **PUSH :**

- A new item is inserted with push operation.

$$SP \leftarrow SP - 1$$
$$M[SP] \leftarrow DR$$

- The stack pointor decremented so that it points at the address of next word.
- A memory write operation inserts the word from DR into top of the stack.

⇒ **POP :**

$$DR \leftarrow M[SP]$$
$$SP \leftarrow SP + 1$$

- The top item is read from the stack into DR.
- The stack pointor is then incremented to point at the next item in the stack.
- We need two micro instruction for POP operation.
  (1) an access to memory through SP
  (2) updating SP
- The advantage of memory stack is that the CPU can refer to it without having to specify an address, since the address is always avalaible and automatically updated in the SP.

(2) Write a note on Addressing Modes.

-> The term addressing mode refers to the way which the operand of an instruction specified. The addressing mode specifies rule for interpreting or modifying address field of the instruction before operand is executed.

(a) Impiled mode : In this mode operands are specified implicity in the definition of the instructions.

- The instruction "Complement accumulator" is an impiled mode instruction because the operand in the accumulator register is impiled in the definition of instruction.

(b) Immediate Mode : In this mode operand is specified instruction itself.

- In other words an immediate mode instruction has an operand field rather than instruction field.

- The operand field contains the actual operand to be used in conjuction with the operation specified in instruction.

(c) Register Mode : In this mode operands are in register that within the CPU.

- The particular register is selected from a register field in instruction.
- A k-bit field can specify any one 2^k registers.

(d) Direct Mode : In this mode effective address is equal to the address part of the instructions.

- The operand reside in memory and it's address is given directly by the address field of the instruction.

(e) Indirect Address Mode : In this mode address field instruction gives the address where the effective address is stored in memory.

- Control fetches the instruction from memory and uses its address part to acess memory again to read the effective address.
- The effective address in this mode obtain from the following:

Effective Address = address part of instruction + content of CPU register

(3) Explain Characteristics of RISC and CISC.

→ Characteristics of RISC

- Relatively few instruction.
- Relatively few addressing modes.
- Memory access limited to load and store instruction
- All operation done within register of CPU.
- Fixed-length easily decode instruction.
- Single cycle instruction execution.
- ~~Hardwired~~
- Hardwired rather than microprogramed control.
- Relatively large number of register in process unit.
- Efficient instruction pipline.

→ Characteristics of CISC

- A large number of instruction - typically 100 to 250.
- Some instructions that perform specialized tasks and are used infrequently.
- large variety of addressing mode - typically 5 to 20 different modes.
- Variable length instruction format.
- Instruction that manupilates operands in memory.

(4) Explain Instruction Format.

→ Three address Instructions

- It can be represented symbolic as,
  ADD A, B, C
- Where A, B, C are the variables. These variables name are assigned to location in the memory.
- In this Instructions operands A and B called source operands and C called destination operands.
- ADD is the operation to be performed on the operand.
- Genral instruction format.
  Operation Source 1, Source 2, Destination
  (a) Bits required to specify the three memory addresses of three operands. If n-bits are required to specify one memory address 3n bits are required to specify.
  (b) bits required to specify operation.

→ Two address instruction

- It can be represented symbolic as,
  ADD A, B.

- This instruction adds the content of variables A and B stores the sum in variable B destroying its previous content.

- Here operand A is source operand and B is source and destination both operand.

- Genral instruction format.

  Operation Source, Destination.

- To represent this instruction,
  (a) Bits required to specify the two memory addresses of two operands.
  (b) Bits required to specify operation.

→ <u>One Address Instruction</u>.

- It can be representd symbdic as,
  ADD B

- This instruction add the content of variable A into processor register called accumulator and stores the sum back into accumulator destroying the previous content.

- In this instruction second operand assumed in a unique location accumulator.

- Genral instruction

  Operation Source

(5) Explain Reverse polish Notation (RPN).

-> The postfix notation is reffered as Reverse polish notation places the operator after Operand.

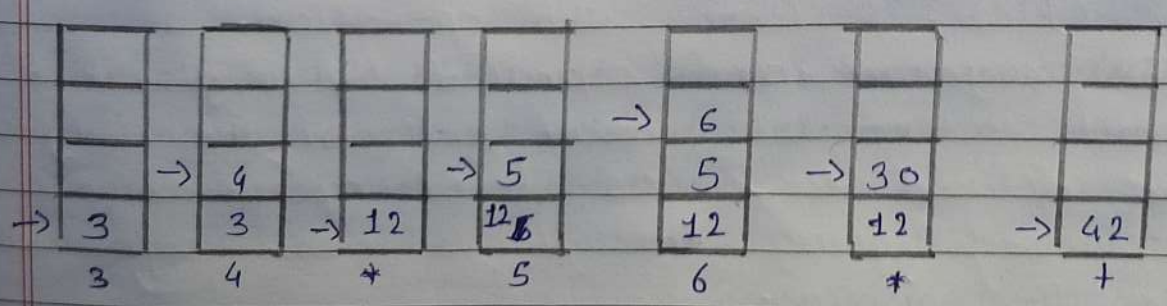
-> A + B -> Infix Notation
+AB -> Prefix or polish notation
AB + -> Postfix or Reverse polish notation.

-> A * B + C * D is written in RPN as
AB * CD * +

-> The conversion from infix notation to RPN must take into consideration the operational hierarchy adopted for infix notation.

-> This heirarchy indicates we first perform all arithmetic inside inner parantheses, inside outer parantheses and perform multiplication and division before addition and substraction.

Ex. (3 * 4) + (5 * 6) => 34 * 56 * +



Chapter : 7

Pipeline and Vector Processing

(1) Explain Pipeline technique. OR Explain Structure of four segmant pipeline.

-> Pipeline is a technique of decomposing a sequential process into sub operation with each sub process being executed in a special dedicated segmant.

-> A pipeline can be visualized as a collection of processing segmant through which binary information flows.

-> Each segmant performs partial processing dedicated by the way the task is partitioned.

-> It is characteristics of pipeline that serval computation can be in progress in distincts segmant at same time.

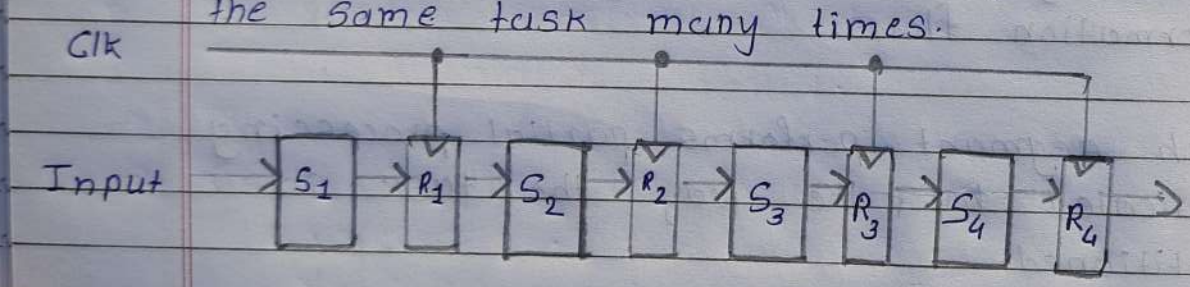
-> The result obtained from the computation in each segmant is transfered to the next segmant in the pipeline.

-> The overlaping of computation is made possible by associating a register with each segmant.

-> The register provides isolation between each segmant so that each can operate on data simultaneously.

-> Any operation that can be decomposed into a sequence of sub operation of about the same time complexity can be implemented by pipeline processor.

-> This technique is efficient for those application that needs to repeat the same task many times.



-> The operands pass through all four segmant in fixed sequence.

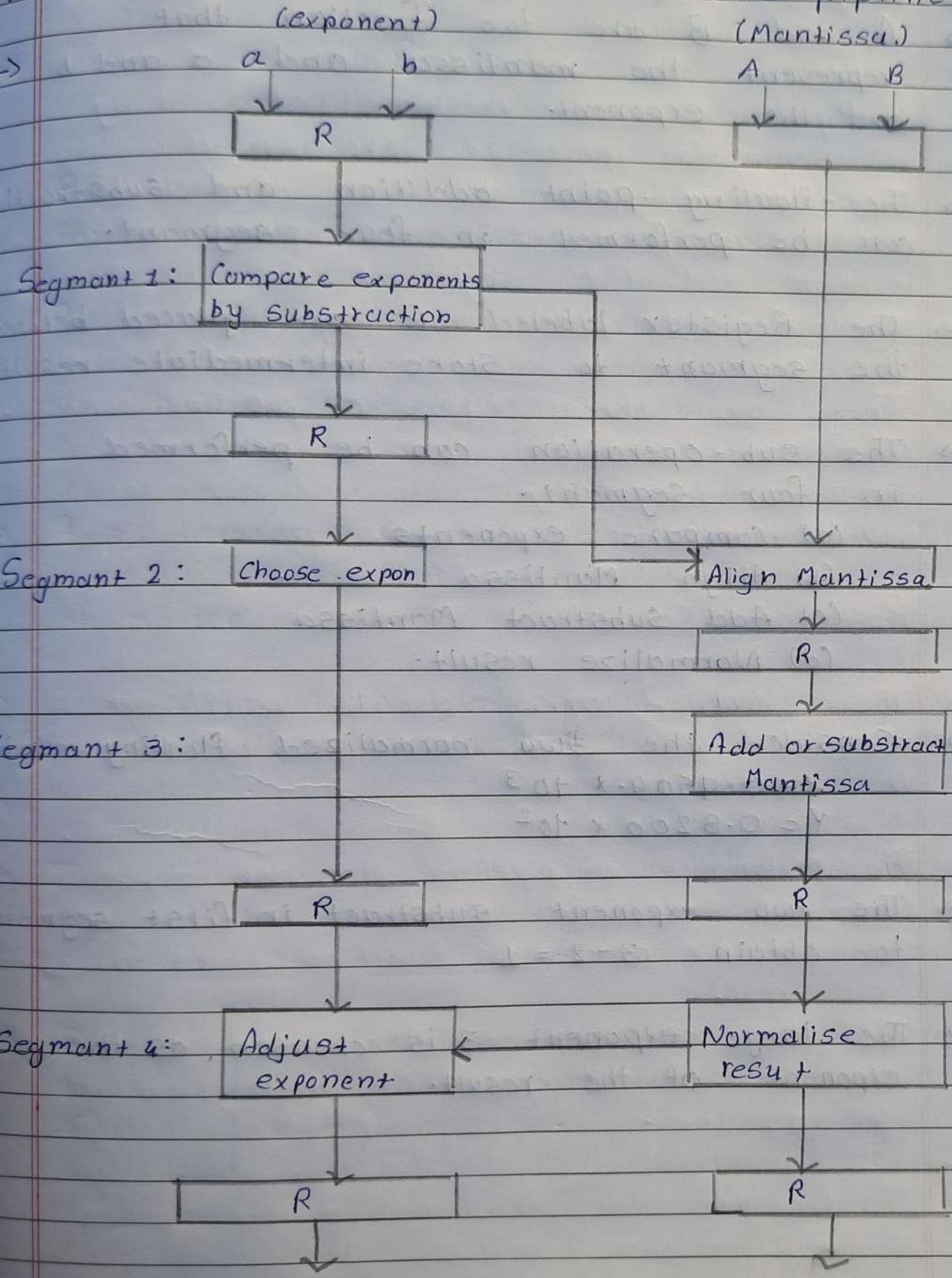
-> Each segmant consist of combinational circuit S, which perform sub operation.

-> The segmant are seprated by Register R, which intermidate result between the stages.

-> We defined task as total operation performed going through all the segmants in the pipeline.

(2) Draw and explain Arithmetic pipline.

→



→ $X = A \times 2^a$
$Y = B \times 2^b$

-> A and B are two fraction that represent the mantissa and a and b are the exponent.

-> The floating point addition and substraction can be performed in four segmant.

-> The Register labeled R are placed between the segmant to store intermediate result.

-> The sub-operation can be performed in four segmant.

(a) Compare exponents
(b) Align Mantissa
(c) Add substract Mantissa
(d) Normalize result.

-> Consider the two normalized floating number

$x = 0.9504 \times 10^3$

$Y = 0.8200 \times 10^2$

-> The two exponent substract in first segmant to obtain $3 - 2 = 1$

-> The large exponent 3 is choosen as the exponent of the result.

-> The next segmant shift Mantissa Y to the right to obtain.

$$X = 0.9504 \times 10^3$$
$$Y = 0.0820 \times 10^3$$

-> This aligns Mantissa under the same exponent. The addition of two mantissa in segmant 3 produce the sum

$$Z = 1.0324 \times 10^3$$

-> Now we have to normalize this sum in segmant 4.

$$Z = 0.10324 \times 10^4$$

(3) Explain Pipeline Conflict.

→ ~~Th~~ Pipeline is a technique of breaking a sequential process into small sub-operation. The execution of each of these sub procedure takes place in a certain dedicated segmant that function together with all other segmant.

→ There are three types of pipeline conflict.
(i) Resource conflict
(ii) Data dependency
(iii) Branch dificulties

* Resource Conflict

→ These conflict arise due to insuffcient resources when it is not possible to overlap the operations.

→ The perfomance of pipeline processor depends on either of the two condition.
- Whether or not the function units are pipelined?
- Do the multiple execution units allow all avaleible mixture of instruction in the pipeline?

→ If for particalar combination the pipeline is avoid resource conflicts then there is a structural hazard.

-> structural hazards happen if two instruction need the help of a given hardware resources at an equal time. It is occured when resources request for memory.

-> Consider a situation where one instruction need to access memory for storage of the resut while other instruction is being fetched. If both instruction reside in same catch unit then only one instruction can proceed and other instruction is delayed. To avoide these issue many processor use seprate catches for instruction and data.

* Data Dependency

-> These conflicts arise when instruction is in the pipeline depends on the result of the previous instruction and these instruction are still in pipeline and not executed yet.

-> When either the source and the destination operand of instruction is not avalaible at expected time in pipeline. It is called data hazard of data conflict.

* Branch Difficulties

-> These difficuty is faced when the branch and other instruction change the content of the program counter.

-> There are two types of branches such as conditional and unconditional.

-> Conditional branches may or may not cause branching but unconditional branch always causing branch~~es~~ing.
It is called control hazard.

-> The following critical action can be followed during pipeline process which helps handling control hazard.

- Timely detection of branch instruction.
- Early calculation of branch address.
- Early testing of branch condition.

(4) Explain Memory Interleaving.

-> Instead of using two memory buses for simultaneous acess, the memor can be partitioned into a number of module connected to a common memory address and data buses

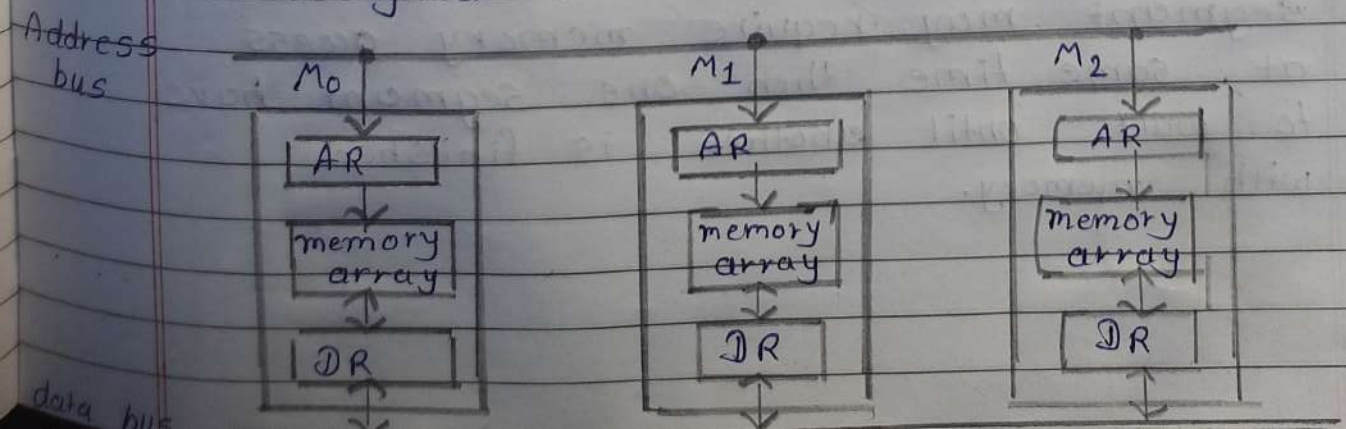
-> A memory module is a memory array together with its own address Register AR and data register DR.

-> Each memory has own AR and DR.

-> The address register recieve information from a common address bus and data register communicate with bidirectional data bus.

-> The modular system permits one module to initiate a memory access while other module are in in the process of reading or writting.

-> The advantage is that it allow interleaving technique.
-> In interleaved memory different sets of address are assigned different Memory module.



(5) Explain Instruction Pipeline.

→ Pipeline processing can occure not only in data stream but in the instruction stream as well as.

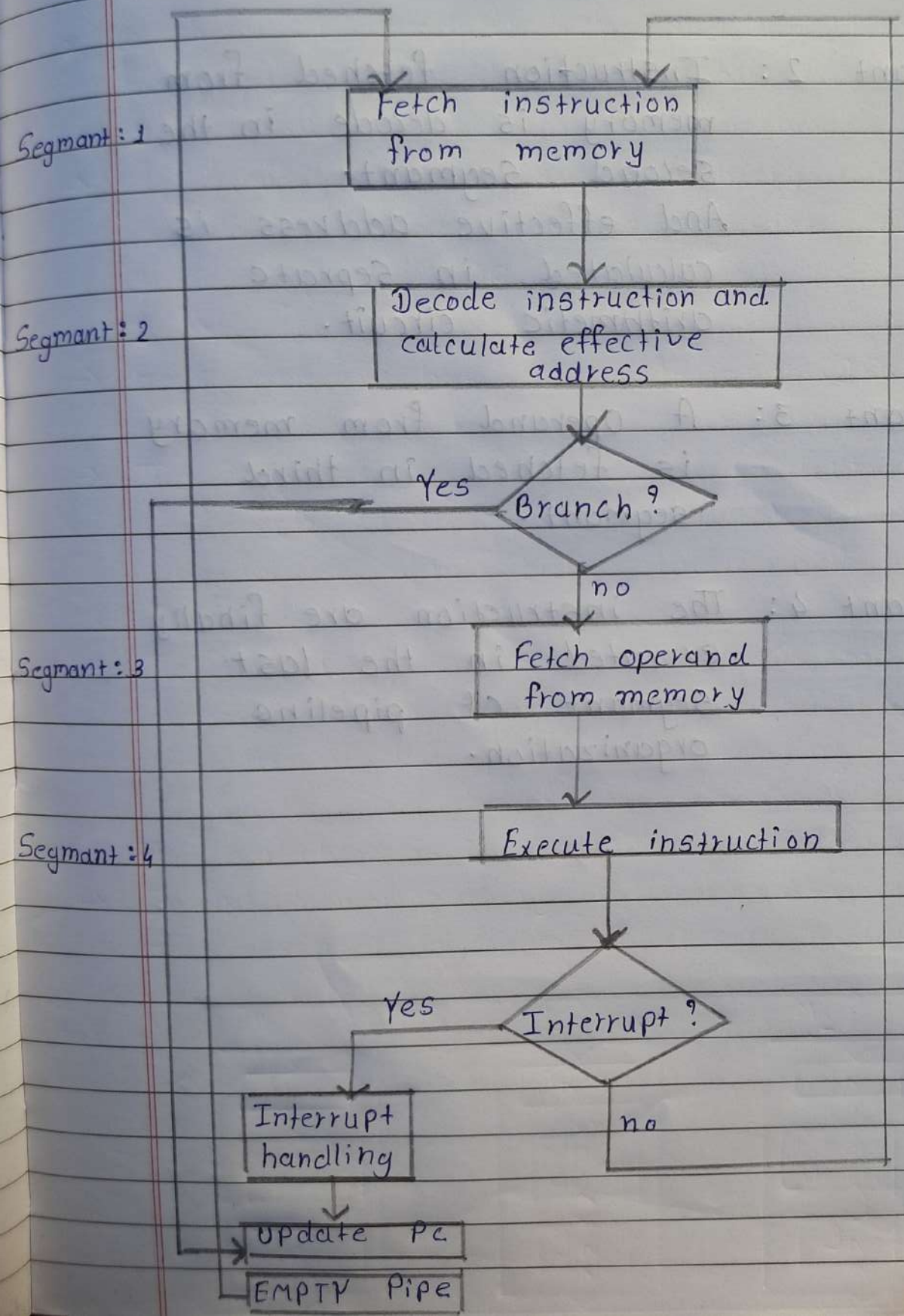
→ Most of the digital computer with complex instruction are required instruction pipeline to carry out operation like fetch, decode and execute instruction.

→ The computer needs to process each instruction with following step.
(a) Fetch instruction from memory
(b) Decode the instruction
(c) Calculate effective address
(d) Fetch the operand from memory
(e) Store result in proper placed.

→ Each step is executed in a particular segmant and there are times when different segmant may take different times to operate on Incoming information.

→ There are times when two or more segment may require memory access at same time then one segmant have to wait until another is finished with memory.

→ The organization of an instruction pipeline will be more efficient if the instruction cycle is divided into segmant of equal duration.



-> Segmant 1 : The instruction fetch segmant Implemented using FIFO buffer.

-> Segmant 2 : Instruction fetched from memory is decode in the second segmant. And effective address is calculated in seprate arithmetic circuit.

-> Segmant 3 : A operand from memory is fetched in third segmant.

-> Seymant 4 : The instruction are finally executed in the last segmant of pipeline organization.

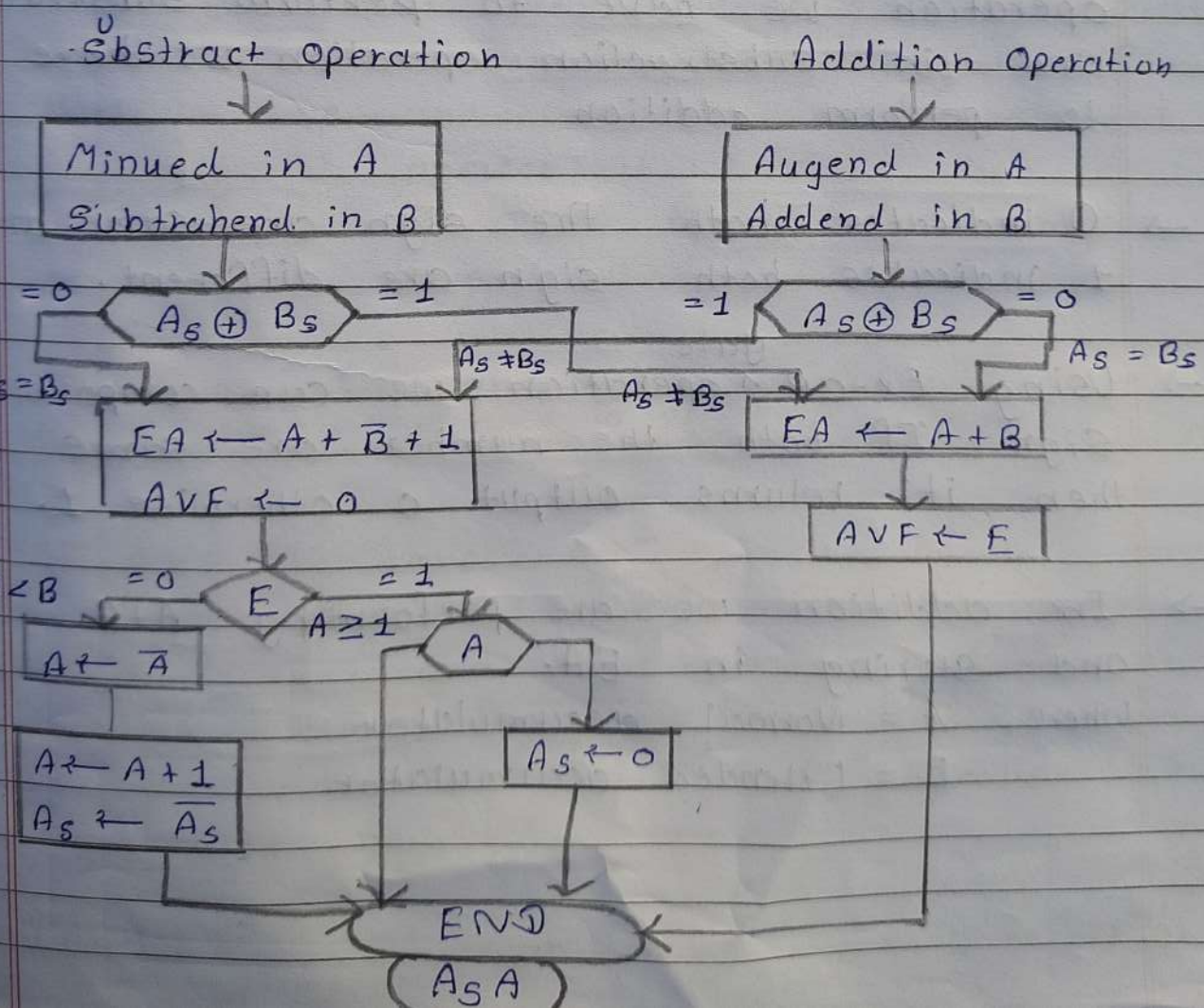
Chapter : 8

Computer Arithmetic

(1) Explain addition and substraction using signed Magnitude data.

→ In unsigned data we don't consider sign of Integer value. Wheather having (+5) or (-5) there is only (5).

→ In signed data we have to consider sign of Integer value.



-> Here in Substraction operation we having Minued in A and substrahend in B.

-> In addition we are having Augend in A and Addend in B.

-> We need to compare wheather both the sign are same or not If both the sign are same in Addition operation we have to perform addition and in substraction we have to perform substraction.

-> If signs are different then in Addition operation we have to perform substraction and in substraction operation we have to perform addition

-> 0 indicates both the sign are same and 1 indicates both sign are different.

-> Using Ex-OR gate operation we can compare sign. If both the number are same then it returns output 0 otherwise 1.

-> In addition we are performing A+B and storing in EA.
Where, A = Normal accumulator
E = Extended accumulator

-> If overflow genrated then overflow bit stores in E. After that we are storing overflow bit in AVF. AVF representing flag for overflow.

-> If we have to perform substraction we have to add 2's complement of substrahend into minued $(\bar{B}+1)$ is a 2's comp of 11. The resutt will store in EA.
In substraction AVF is always 0 because here no overflow bit in substraction.

-> After substraction we have two option wheather $A \geq B$ or $A < B$.
-> If $A \geq B$ then the answere is always positive.
~~If A < B then the answere is negative form that means answer is in 2's complement~~
for original answer if $A \geq B = 0$ then my answer is 0 if $A \geq B \neq 0$ then my answere is correct.

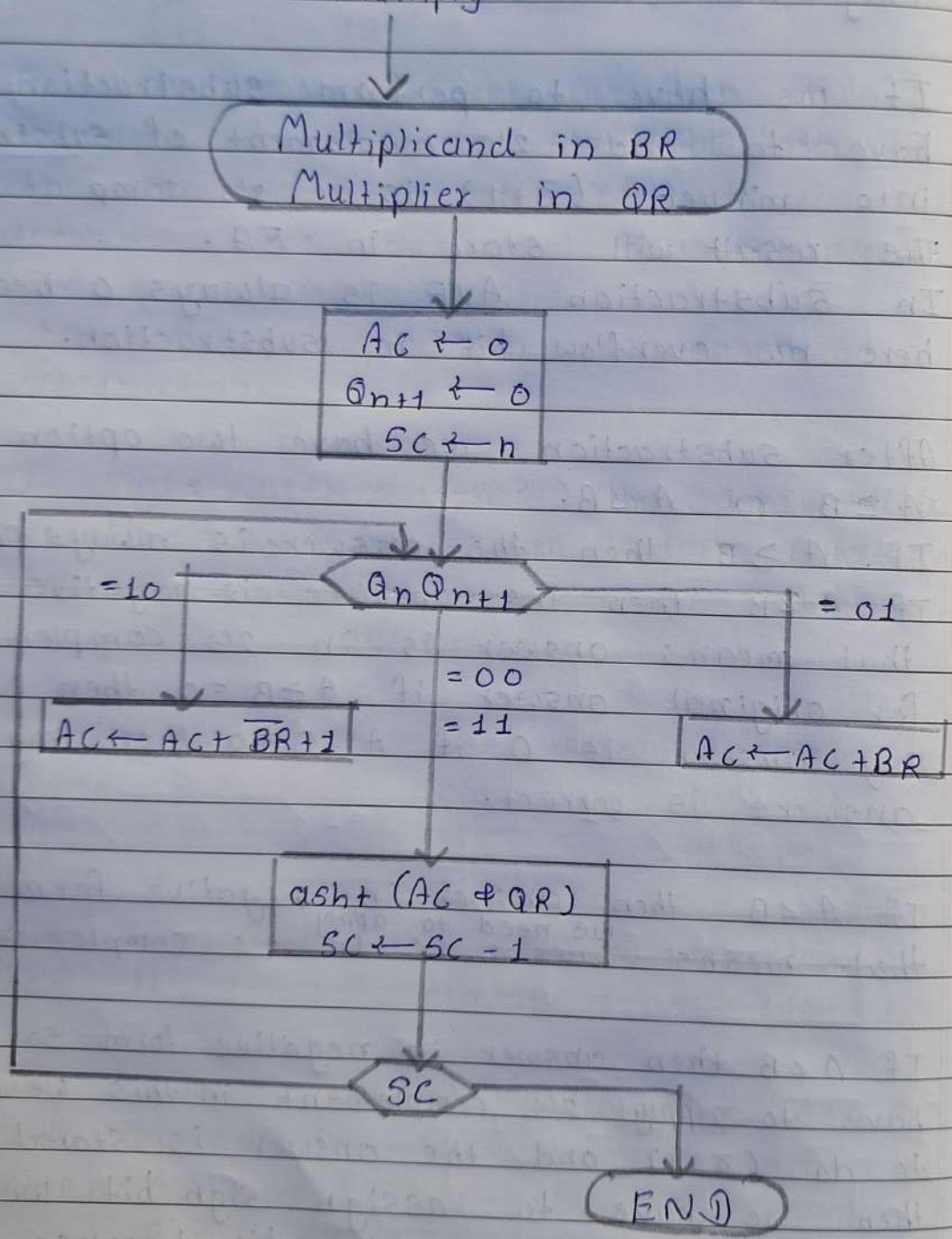
-> ~~If A < B then answer in negative form that means answer is in 2's complement form~~ we need to apply

-> If $A < B$ then answer in negative form so we have to apply 2's complement in this we have to do $(\bar{A}+1)$ and the answer is stored in A. then we have to assign sign bit we have to do complement of sign bit because we are getting negative sign.

-> [As A] result is stored in A and sign bit store in As.

(2) Explain Booth's Algorithm.

→



-> Booth algorithm gives a procedure for multiplying binary integer in signed 2's complement Representation.

-> In operates on the fact that string of 0's in the multiplier required no addition but just shifting and a string of 1's multiplier from bit weight 2^k to weight 2^m can be treated as $2^{k+1} - 2^m$.

-> We have to store multiplicand in BR that is register and Multiplier in QR it is also register.

-> Initially AC and Q_{n+1} bits are set to 0 and the SC is a sequence counter that represents the total bit set n, which is equal to number of bit in multiplier.

-> After that we encountered two bit of multiplier as Q_n and Q_{n+1} where Q_n represent last bit of QR and Q_{n+1} represented incremented bit of Q_n by 1.

-> Suppose two bit of multiplier is equal to 10 that means we have to substract multiplier from AC then perform the arithmetic shift operation.

-> If two of multiplier equal to 01 that means we need to perform addition of multiplicand and AC after that perform the arithmetic shift operation.

-> The arithmetic shift operation in Booth's algorithm to shift AC and QR bits to the right by one and remains sign bit in AC unchanged.

-> And Sequential counter is decrement by 1.

-> If SC = 0 then END the operation
If SC ≠ 0 then again we need to check $Q_n Q_{n+1}$.

Multiply (15) with (13) Using booth's algorithm.

15 : 01111
13 : 01101

AC	QR	Q_{n+1}	BR	SC	comment.
00000	01111	0	01101	5	Initial value
10011	01111	0	01101	4	sub: AC + BR'+1
11001	10111	1	01101		A+shr
11100	11011	1	01101	3	A+shr
11110	01101	1	01101	2	A+shr
11111	00110	1	01101	1	A+shr.
01100	00110	1	01101	0	add: AC+BR
00110	00011	1	01101		A+shr.

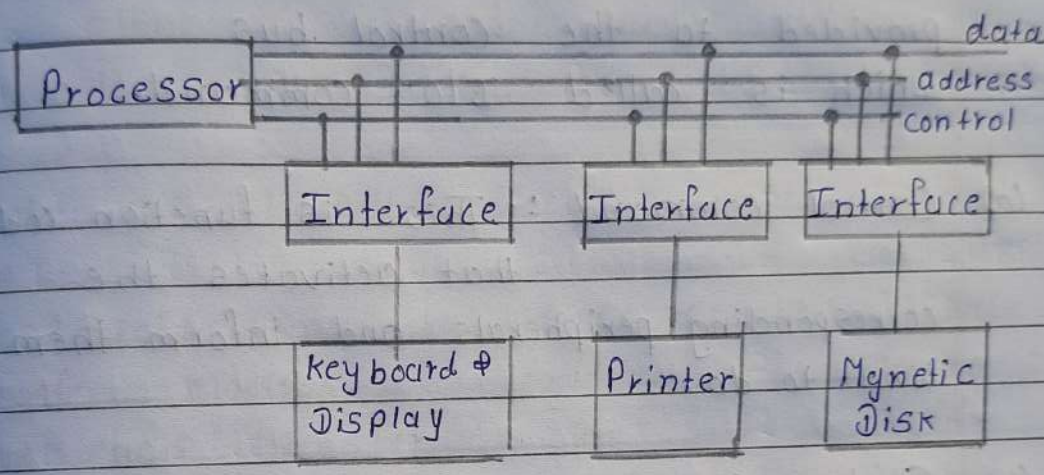
→ 15 x 13 = 195 = 0011000011

Chapter: 9

Input-Output Organization

(1) Explain I/O Bus and Interface Module.

→



→ The data bus, address bus and control bus that arise out of the processor and are intended to communicate with I/O devices are called I/O bus.

→ Input-Output device attached to computer is also called peripherals.

→ The I/O bus is connected to all peripherals interface. To communicate a particular device the processor places a device address on the address bus.

→ Each interface attached to I/O bus contains an address decoder that monitor the address line.

-> When interface detects an address to be it's own. it activates the path between the bus and the device that it controls. All other peripherals are disable.

-> At the same time a function code is provided to the control bus which is called Ilo command.

(a) Control command : This is function code that activates the corresponding peripherals and inform them what to do.

(b) Status Command : It is used to test various status condition in the interface and periphred devices like BUSY, ERROR data avalaible or not in buffer.

(c) Data output Command : It cause the interface to respond by transferring data from processor to the peripherals. The data is sent from CPU to the buffer of interface after this command is provided.

(d) Data Input Command : This command is sent by the CPU if the data is to be read from peripheral. After this command the data of peripheral are extracted into the buffer of the interface and are read by the CPU.

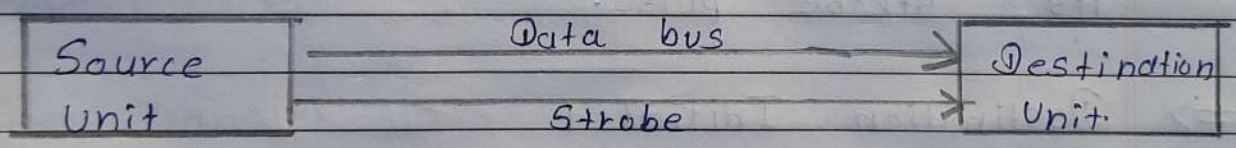
(2) Explain Asynchronous data transfer and Explain Strobe control in detail.

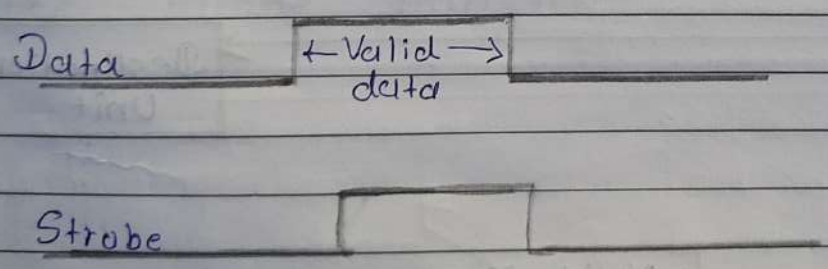
→ Data transfer between two independent unit where internal timing in each unit is independent from each other. Such two units are said to be asynchronous to each other. If transfer occurs between them then it is called asynchronous data transfer.

* Strobe control Method

→ The Strobe control method of asynchnous data transfer employs a single control line to time each transfer.

⇒ Source Initiated :





• The Strobe may be activated by either the Source or the destination unit.

- The databus carries binary information from source unit to destination unit
- The strobe is a single line that inform the destination unit when valid data word is avalaible in the bus.
- The source unit first places the data on the data bus.
- After delay to ensure that the data settle to a steady value the source activate the strobe pulse.
- The information on the data bus and the strobe signal remains in the active state for a sufficient time period to allow the destination unit to receive data.
- The source remove data from the bus a brief period after it disable its strobe pulse.

⇒ Destination Initiated

- Data transfer initiated by the destination unit. In this case the destination unit activates the strobe pulse, informing source to provide data.

- The source unit responds by placing the requested binary information on the data bus.

- The data must be valide and remain in the bus long enough for the destination unit to accept it.

- The falling edge of strobe pulse can be used again to trigger a destination register.

- The destination unit then disable the strobe The source remove data from bus after time interval.

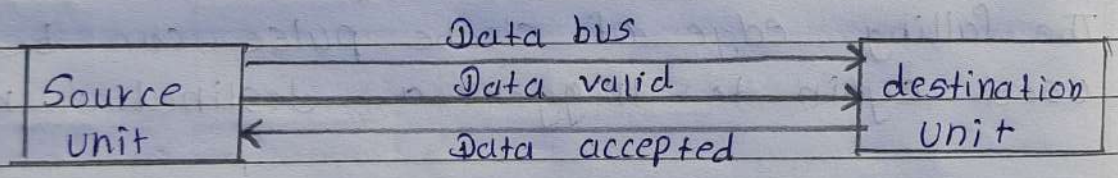
- The transfer of data between the CPU and an interface unit is similar to the strobe transfer just describe.

(3) Explain Asynchronous data transfer Using Hand Shaking Method.

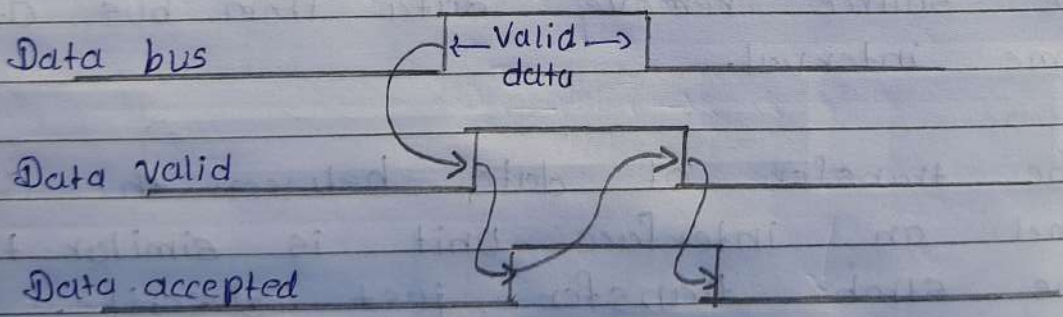
→ The handshake method solves the problem of strobe method by introducing a second control signal that provides reply to the unit that initiate the transfer.

⇒ Source Initiated

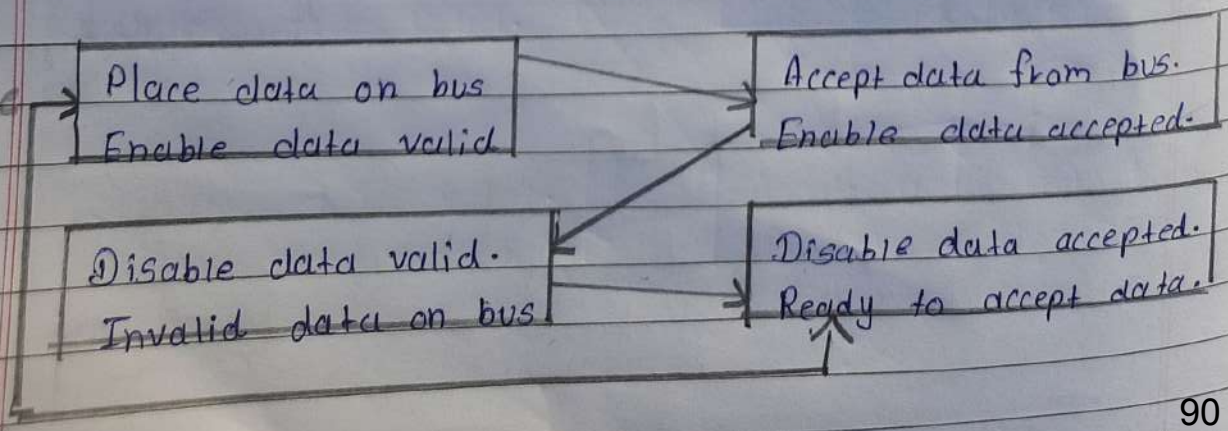
Block diagram



Timing diagram



Sequence of event



- One control line is in the same direction as the data flow in the bus from source to destination unit.

- It is used by source unit to inform destination unit wheather there is valid data in bus.

- The other control line is other direction from destination to source unit. It is used by destination unit to inform the source wheather it can accept data.

- The sequence of control during the transfer depends on the unit that initiates transfer.

- The two handshaking line the data valid, which is genrated by the source unit, and data accepted genrated by destination unit, the timing diagram shows exchange of signals between the two unit.

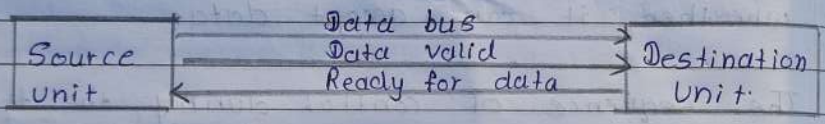
- The source unit initiate the transfer by placing data on the bus and enabling its data valid signal.

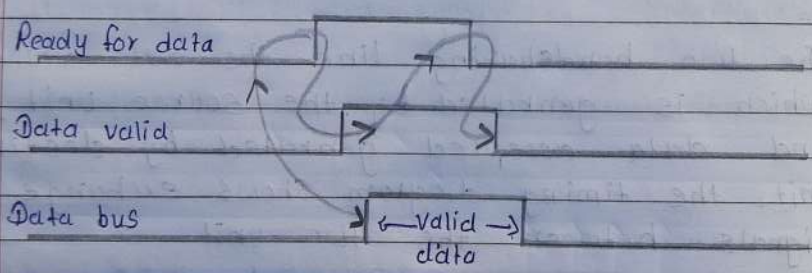
- The data accepted signal is activated by the destination unit after it accepts data from bus.

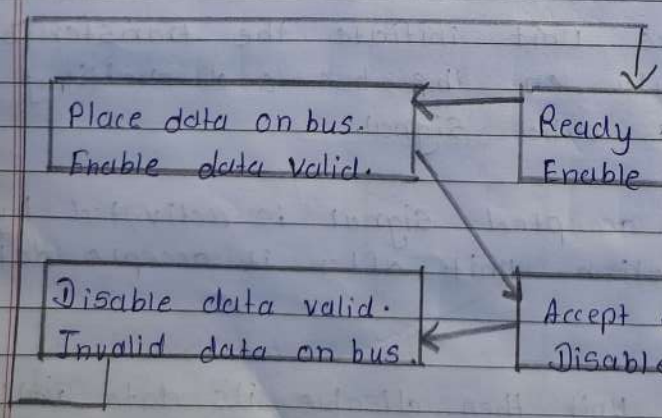
- The source unit then disable its data valid signal which invalidates the data on bus.

- The source unit does not send data until after the destination unit show its rediness to accept new data by disabling its data accepted signal.

⟹ Destination - Initiated







- The destination initiated transfer using handshaking line shown in figure.

- Note that name of the signal that genrated by the destination unit has been changed to ready for data to reflect its new meaning.

- The source unit in this case does not place data on the bus until after it recives ready for data signal from destination unit.

- The handshaking procedure follow same pattern as in source initiated.

- The sequence of event in both cases would be identical if we consider the ready for data signal as complement of data accepted

- The only difference between source and destination initiated transfer is in their choice of inital state.

(4) Explain DMA.

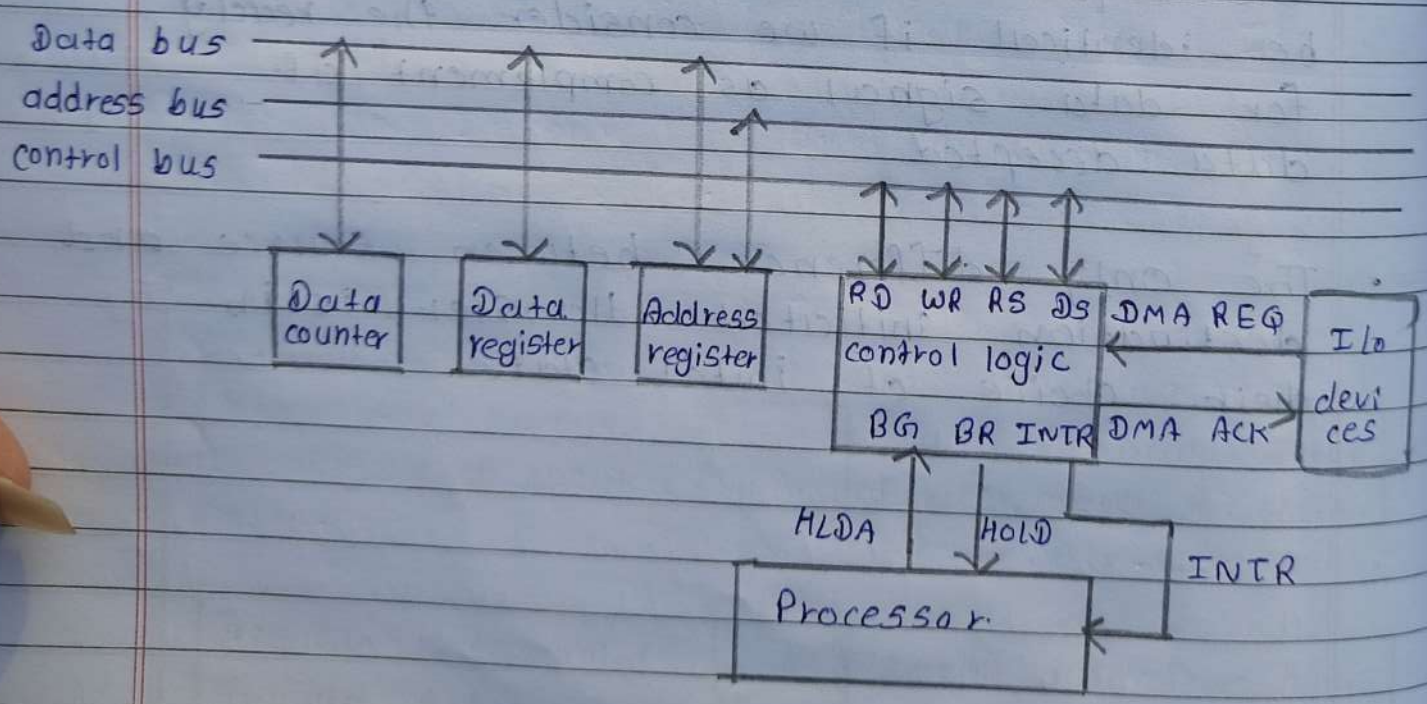
→ DMA stands for direct memory address.

→ Transfer data under programmed I/o between CPU and periphral.

→ In DMA, interface transfer data into and out of memory through the memory bus.

→ When transfer is made the DMA request memory cycle through memory bus.

→ When request is granted by memory controller DMA transfer data directly into memory.



- DMA controller communicates with the CPU via the data bus and control line.

- The register in DMA is selected by CPU through the address bus by enabling the DS and RS inputs.

- The RD and WR inputs are bidirectional.

- When the BG input is 0, the CPU can communicate with the DMA register through the data bus to read from or write the DMA register RD and WR signals are input signal for DMA.

→ When BG = 1, the CPU has the buses and DMA communicate directly with the memory by specifying an address in the address bus and activating the RD or WR signals. DMA consist of data count, data register, address register, control logic.

- Data counter register stores the number which gives the number data transfer to be done in one DMA cycle. It is automatically decremented after each word transfer.

- Data register acts as buffer whereas adress register initially holds the starting address of the device. Actually, it stores the address of next word to be transfered. It is automatically increamented and decremented after each word transfer.

- After each transfer, Data counter is tested for zero. When data count reaches zero the DMA transfer halts.

- The DMA controller normally provided with an interrupts capability, in which case it sends an interrupt to processor to signal the end of I/O data transfer.

(5) Differentiate Isolated I/O and Memory mapped I/O.

→

Isolated I/O	Memory Mapped I/O
• Memory and I/O have seperate address space.	Both have same seperate address space
• All address can be used by memory.	I/O addresseble memory become less for memory.
• I/O address are called ports.	Normal memory address are for both.
• More efficient due to seprate buses.	Less efficient.
• Larger in size due to more buses	Smaller in size.

(1) Explain catch coherence.

-> In a single CPU System two copies of same data one in catch memory and other in main memory may become different. This data inconsistensy is called catche coherence problem.

-> Catch updating systems eliminates data inconsistency in the main memory caused by catch write operations.

-> In multiprocessor system another bus master can take over control of the system bus. This bus master could write data into main memory.

-> The 80386 Supports four different approaches to prevent data inconsistency, that is to protect catche coherency.

(a) Bus watching (b) Hardware transparency
(c) Non-cachable memory (d) cache flushing

(a) Bus watching : In bus watching cache controller invalidates the cache entry if another master writes to a location in shared memory which also reside in the cache memory.

(b) Hardware transperency : In this accesses of all devices to the main memory are routed through the same cache or by copying all cache writes both to the main memory and to all other caches that share the same memory.

(c) Non-caheble Memory : The 80386 Dx partition its main memory into a cacheble and non-cacheble memory. By designing shared memory as non-cacheble memory cache coherency can be maintained since shared memory never copied into cache.

(d) Cache flushing : To avoid data inconsistency a catch flush write any altered data to the main memory and caches in the system are flushed before a device writes to shared memory.

(2) Explain Virtual Memory

-> In most modern computer physical main memory is not as large as the address space by an address issued by processor.

-> Here the virtual memory technique is used to extend the size of physical memory. It uses secondary storage such as disk to extend size of physical memory.

-> When program does not completley fit into the memory it is divided into segmant.

-> The segmant which are currently executed are kept in main memory and remaining segmant are stored in secondary storage devices.

-> If executing program need segmant which are not present in main memory then the required segmant copied from secondary storage.

-> If a new segmant of program is copied ~~from~~ into main memory it must be replace another segmant in the memory.

-> In modern computer the OS moves program and data automatically between the main memory and secondary storage.

-> Technique that automatically swap program into secondary storage device are called virtual memory.

-> The physical inaddress of main memory the set of such address is called memory space.

-> Memory space consist of actual main memory location directly addreseble for processing.

-> The address space is allowed to be larger than the memory space in computer with virtual memory.

-> If virtual address refers to a part of the program or data space that is in main memory then the content of appropriate location in the main memory are accessed imediateley.

-> If reffered address is not in the main memory its content must be brought into a suitable location in the main memory before they can be used.

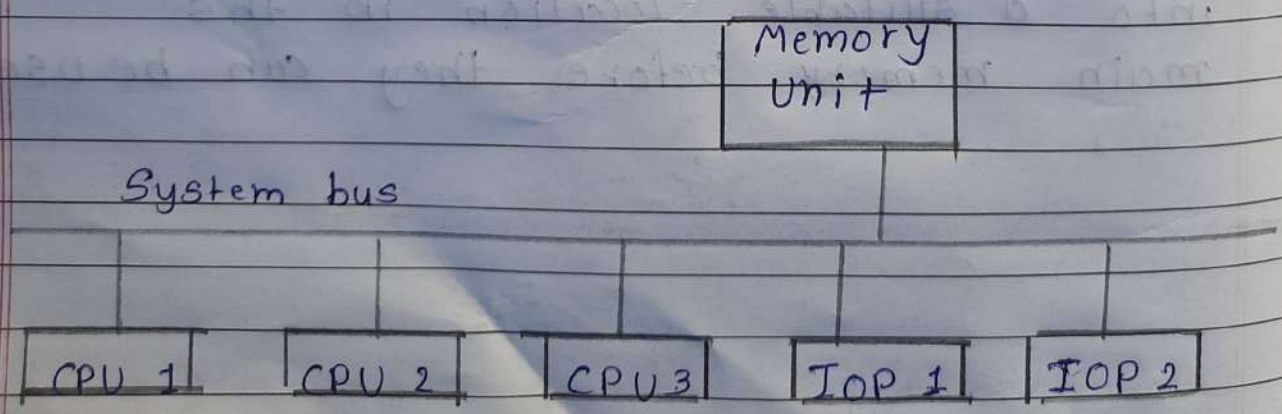
(3) Explain Time shared common bus interconnection Structure.

-> A common bus multiprocessor system consist of a number of processor connected through a common path to a memory unit.

-> Only one processor can communicate with the memory or another processor at any given time.

-> Transfer operations conducted by the processor that is in control of bus at the time.

-> Any other processor wishing to transfer must first determine avalability status of bus and only after the bus becomes avalaible.



-> A command is issued to inform the destination unit, what operation is to be performed.

-> The reciving unit recognizes its address in the bus and respons to the control signal from sender after which the transfer is initiate.

-> A single common bus system is restricted to one transfer at a time.

-> This means that when one processor is communicating with the memory all other processor are either busy or must be waiting for the bus.

-> As cosequence the total overall transfer rate within the system is limited by the speed of single path.

-> Here we have number of local bus each connected to its own local memory.

-> Each local bus may be connected to CPU, an IOP, or any combination of processor.

-> A system bus Controller links all local bus to common bus system.

-> The I/O devices connected to Local IOP, as well as the local memory, are avalaible to the local processor.

-> The memory connected to local bus common system bus is shared by all processor.

-> If IOP is connected to the system bus the I/O devices attached to it may be made avalaible to all processor.

-> Only one processor can communicate shared memory with shared memory and other common resources through the system bus at any given time.

-> The other processor are kept busy communicating with their local memory and I/O devices.

(4) Differentiate Tightly and Losely Coupled System.

→

Tightly coupled	Loosely coupled
• Task and processor communicate in a highly synchronized fashion	Task and processor do not communicate in synchronized fashion.
• Communicates through a common shared memory.	Communicate by message passing packet.
• Shared memory system	Distrubuted memor system
• Overhed for data exchange is lower comparitively.	Overhead for data exchang higher.