Register is made up of F/F

UNIT-1

(Register - for holding temporary data)
(F/F store 1 bit)

REGISTER TRNSFER LANGUAGE (RTL) -

* MICRO INSTRUCTION-OPERATIONS -
(ex - shift, load, count, Increment, decrement, etc.)

The operation executed on data stored in registers are called micro operations.

The set of micro operations is known as micro instructions. & the set of micro instructions is known as a micro program. ex -

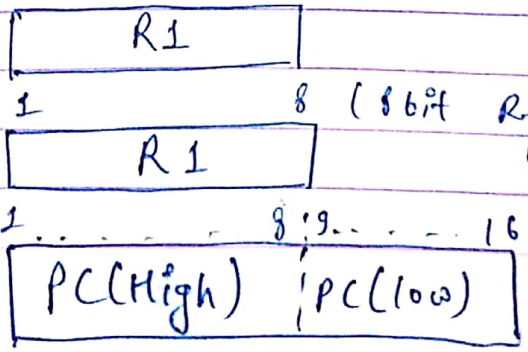
The Symbolic notation used to describe the micro operation transfer with airthmetic or logic operation among registers is called a register transfer language (RTL).

* REGISTER TRANSFER -

(i) General Purpose Register → R1, R2

(ii) Special Purpose Register - (MAR) Memory address register.
(MDR) Memory Data register.
IR Instruction Register
SP (Stack Pointer), PC (Programme Count), A or AC (Accumulator).

* Block Diagram



Scanned by CamScanner

If System have n bit Data bus then a Register of n bit

micro operations

(i) $R_1 \leftarrow R_2$ (Single Clock pulse)
(RTL Statement)

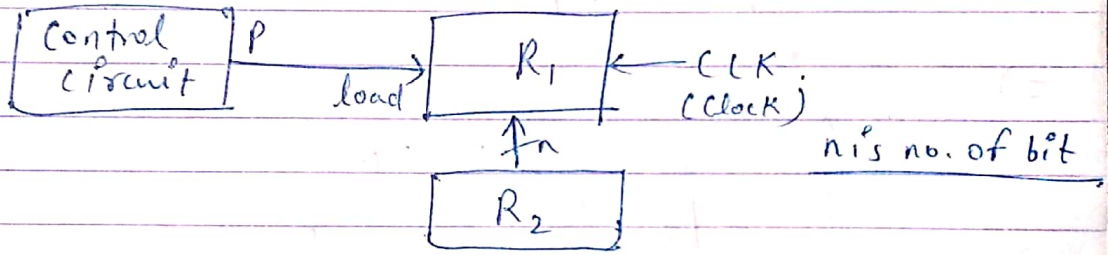
(ii) $R_1 \leftarrow R_2$, $R_3 \leftarrow R_2$. (Same clock pulse)

(iii) $P : R_1 \leftarrow R_2$ } Conditional Statement or control function.

P ↓ condition variable

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

Block Diagram of $(P : R_1 \leftarrow R_2)$.

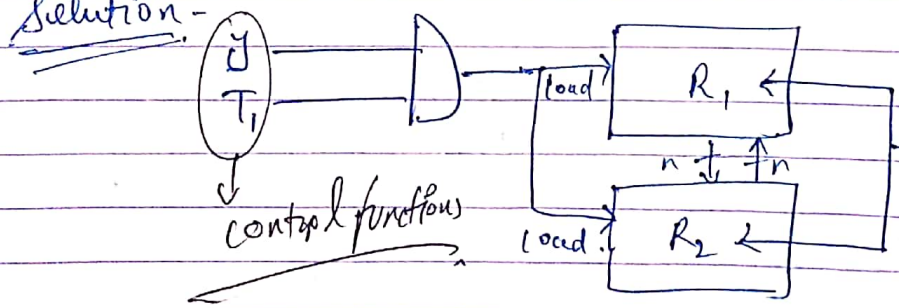


EXERCISE - 1

Q Show the block Diagram of the hardware that emplements the following register transfer statement.

$T_1 : R_2 \leftarrow R_1$, $R_1 \leftarrow R_2$.

Solution -



Scanned by CamScanner

1+1 = 0

Q Represent the following conditional control statement by two register transfer statements with control functions.

If P=1 then $(R_1 \leftarrow R_2)$ else if (Q=1) then $(R_1 \leftarrow R_3)$.

Solution:-

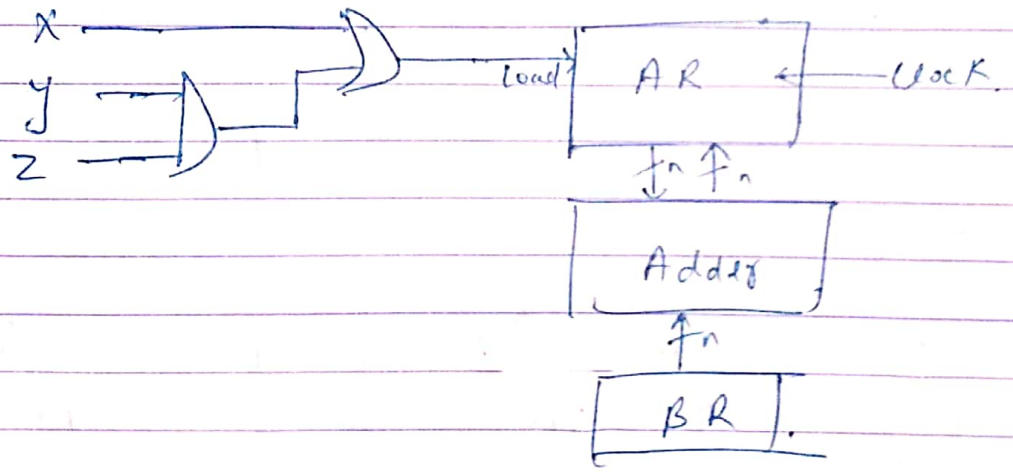
$P : (R_1 \leftarrow R_2)$

$\bar{P}Q : (R_1 \leftarrow R_3)$.

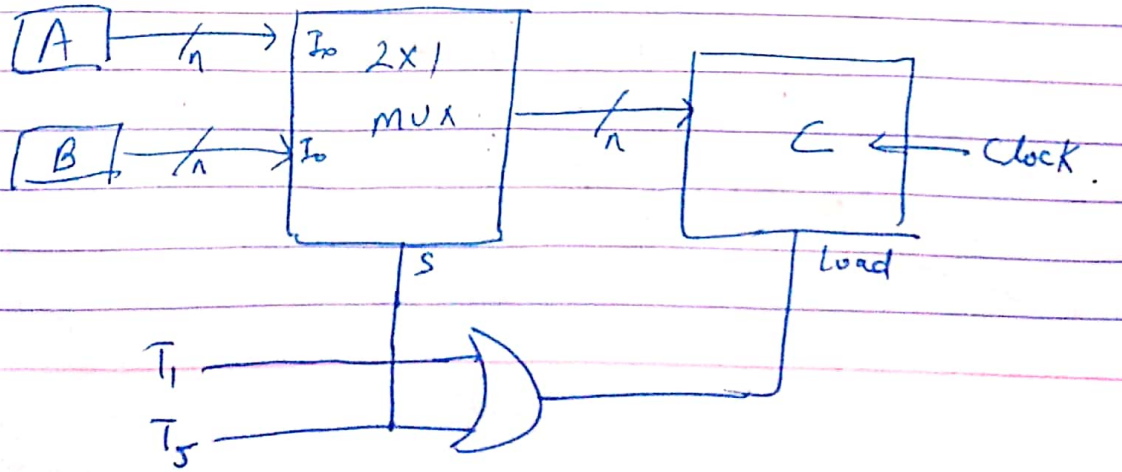
Q Draw the block diagram for the hardware that implements the following statements-

x+yz : AR ← AR+BR

⇒



Q T1 : C ← A Draw the block Diagram.
T5 : C ← B

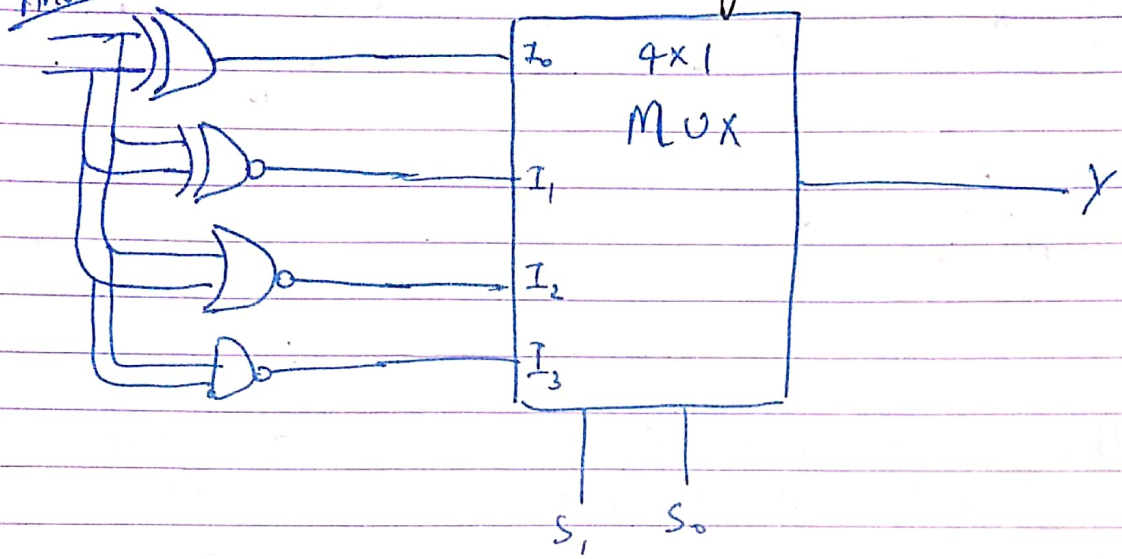


Scanned by CamScanner

Exercise :-

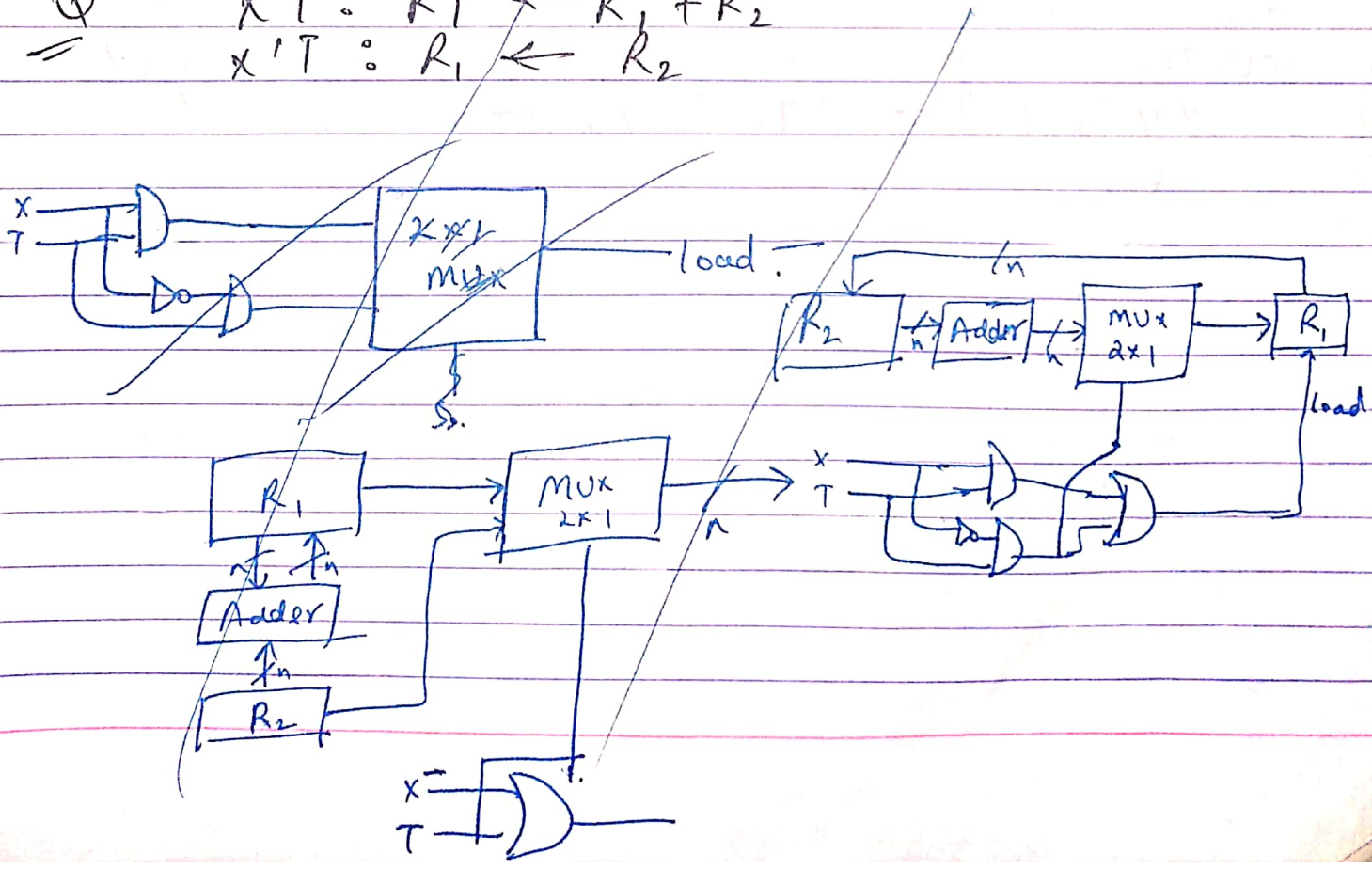
Q Design a digital circuit that performs the four logic operations of XOR, XNOR, NOR & NAND using two selection variables. Show the block diagram.

Answer :-



Q $xT : R_1 \leftarrow R_1 + R_2$

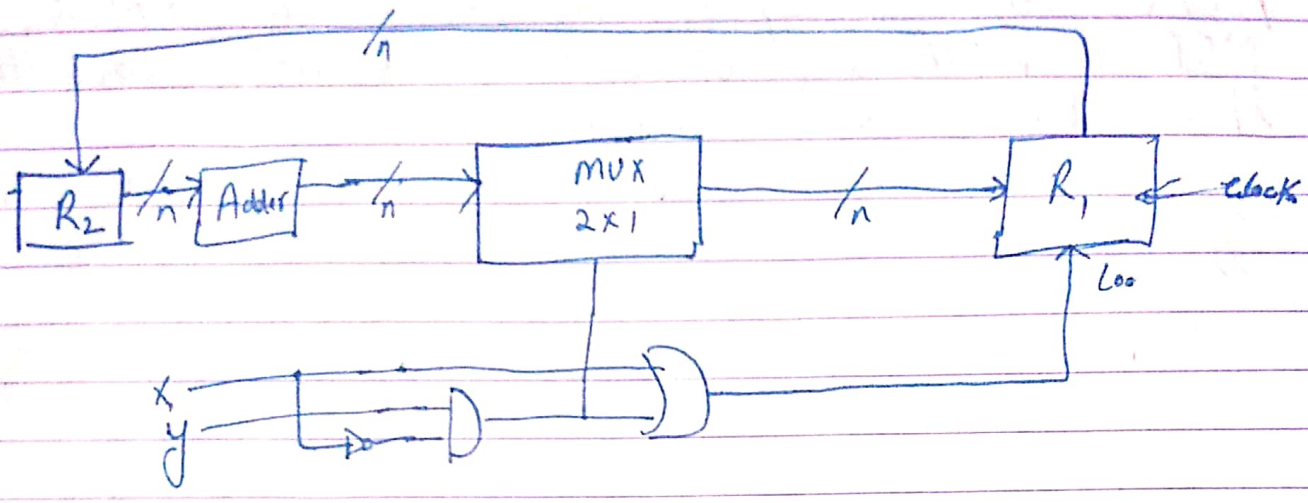
$x'T : R_1 \leftarrow R_2$



Scanned by CamScanner

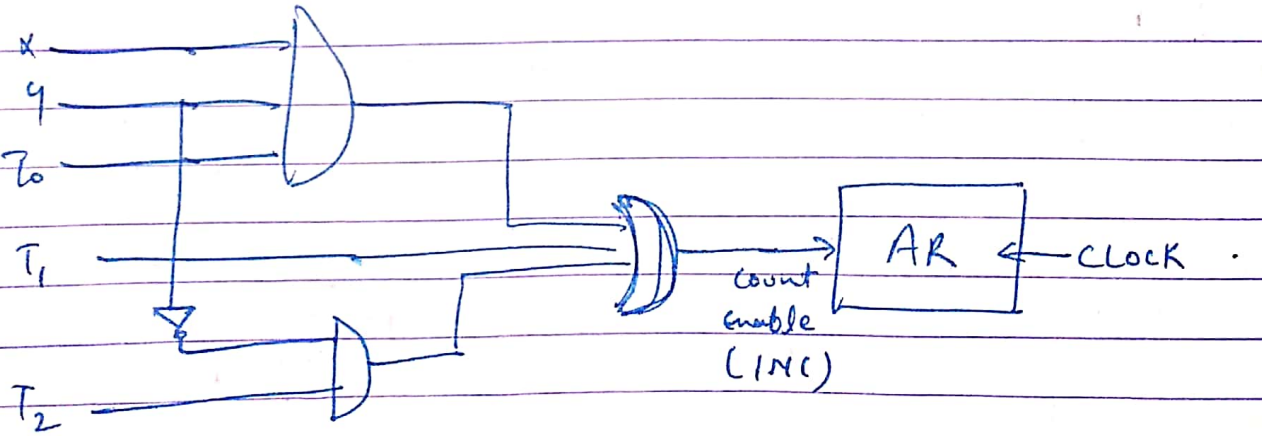
Q $x : R_1 \leftarrow R_1 + R_2$.

$x'y : R_1 \leftarrow R_1 + 1$



Q Show the hardware that increments the following RTL statement. Include the logic gates for the control function & draw the block diagram for the binary counter with a count enable input.

$xyT_0 + T_1 + y'T_2 : AR \leftarrow AR + 1$.



Scanned by CamScanner

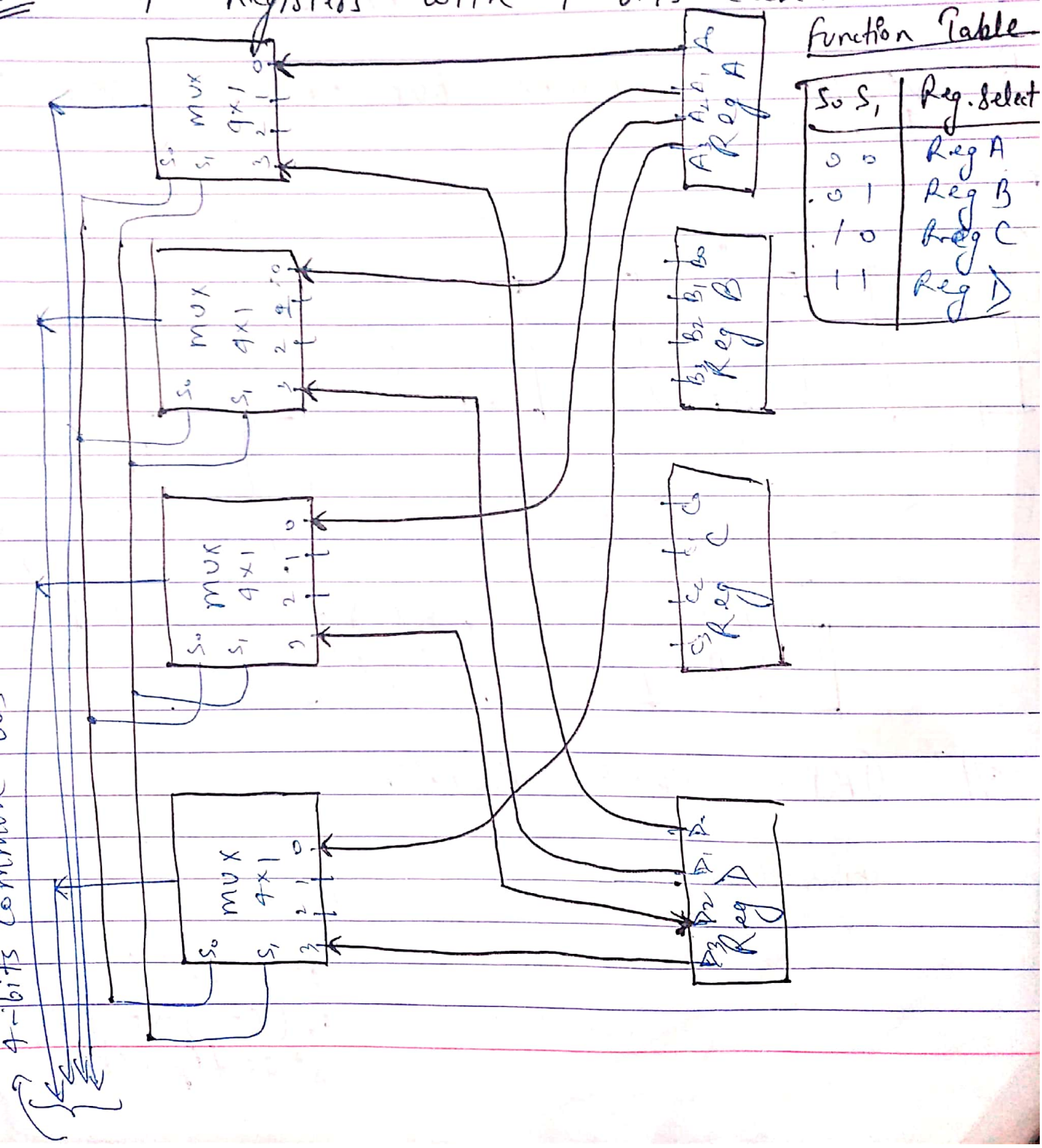
No. of MUX = No. of bit of Register

BUS TRANSFER -

(R1 ← C) = BUS → C, R1 ← BUS

* COMMON BUS SYSTEM (using Multiplexer)

ex- 4- Registers with 4 bits each.



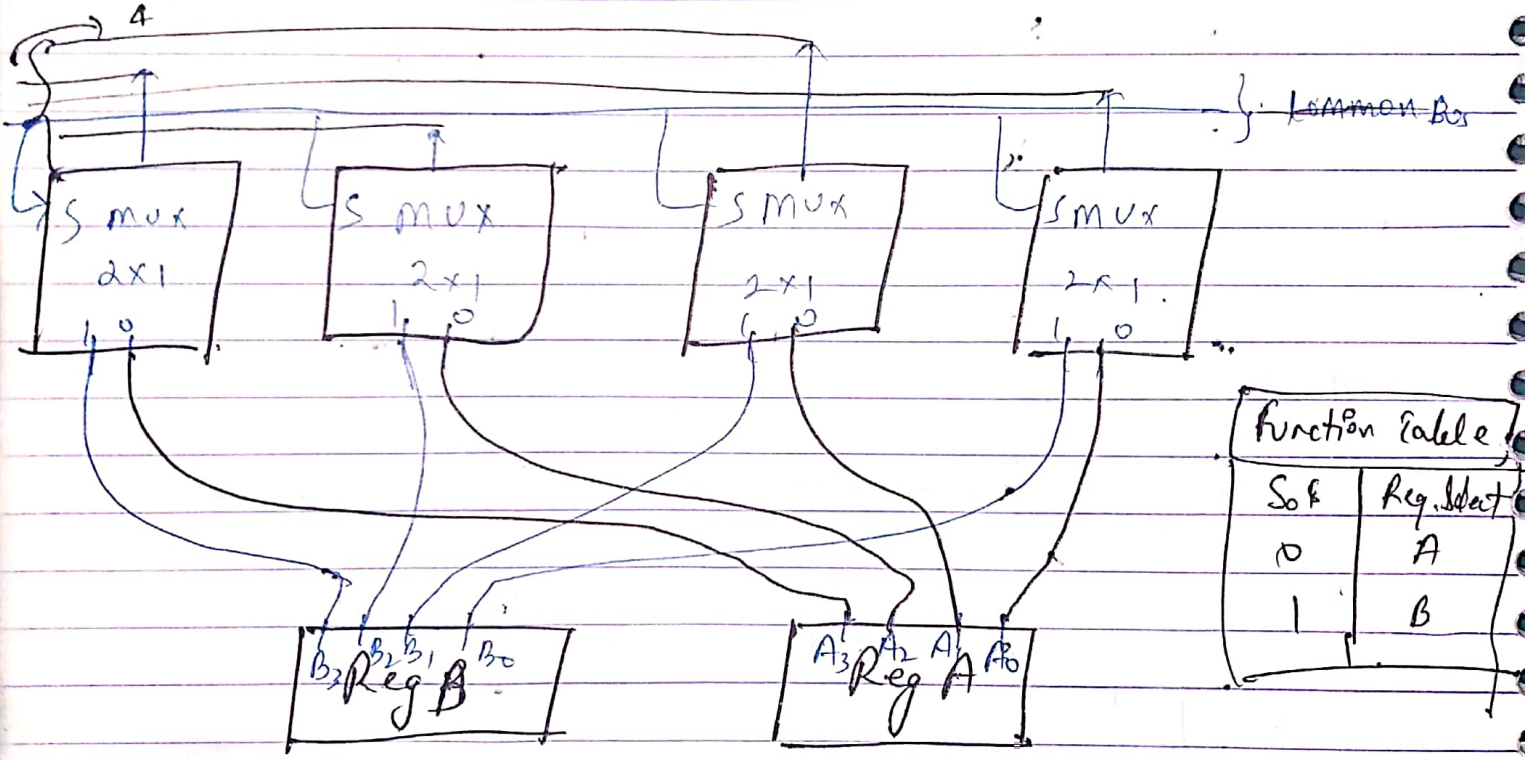
Function Table

S_0 S_1	Reg. Select
0 0	Reg A
0 1	Reg B
1 0	Reg C
1 1	Reg D

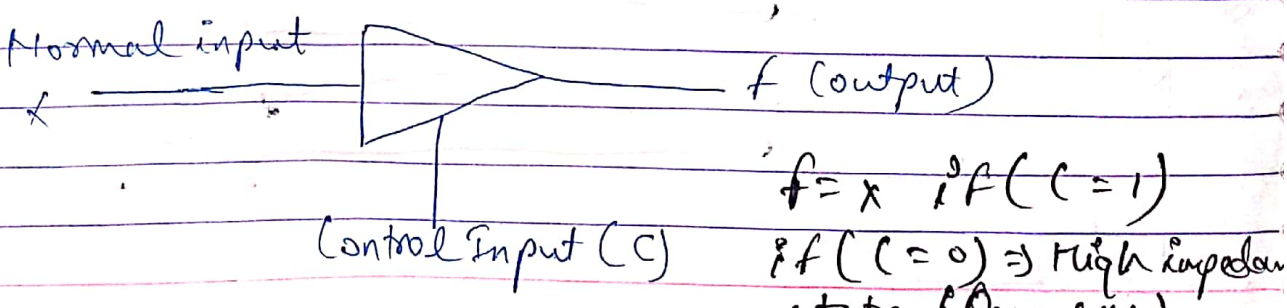
Scanned by CamScanner

NOTE - In general, a bus system will multiplex k registers of n-bits each to produce n - n-bits common bus. The no. of MUX needed to construct the bus is equal to n-bits. The size of each MUX must be kx1.

Q Design a common bus of 2 Register with = 4-bits.

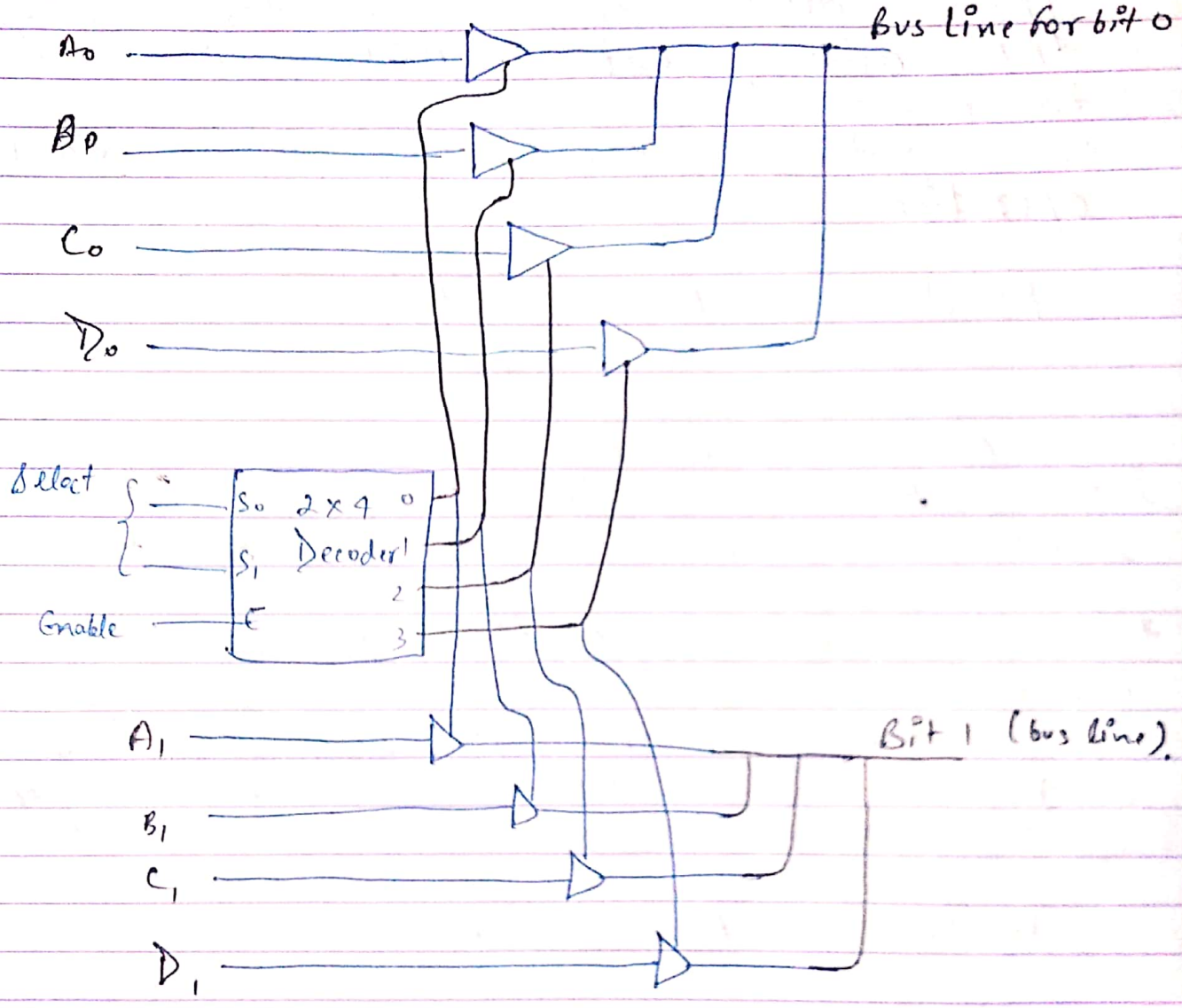


TRI - STATE BUFFER -



Scanned by CamScanner

Common Bus System using Tri-state Buffer -



Advantage -

(only single decoder is used for whole bus system)

decoder output enable's & disables tri-buffer gate.

Scanned by CamScanner

MEMORY TRANSFER -

* Consider a memory unit that receives the address from a register, called the address register (AR). The data are transferred to another register, called the data register (DR). The read operation can be stated as follows -

Read : DR ← M[AR]

M - Memory word

This causes are transfer of information into DR from the memory word M selected by the address - in AR.

* The write operation transfer the information or content of a data register in a memory word M selectected by the address stored in AR. The write operation stated as follows -

Write : M[AR] ← DR

This causes the transfer of information from DR into the memory word M.

Scanned by CamScanner

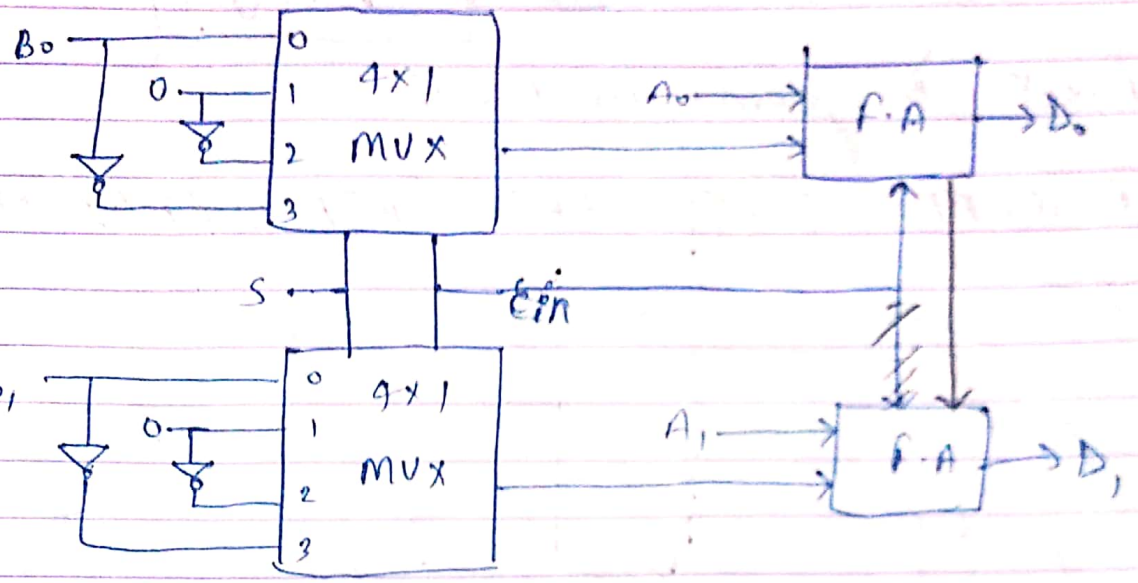
S	C_{in}	
0	0	D = A+B
0	1	D = A+1
1	0	D = A-1
1	1	D = A+$\bar{B}$+1

A.L.U

ex- Q) Draw the logic diagram for the first two stages.

S	C_{in} = 0	C_{in} = 1	
0	D = A+B (add)	D = A+1	(increment)
1	D = A-1 (decrement)	D = A+$\bar{B}$+1	(subtraction)
		(A-B)	

Solution-

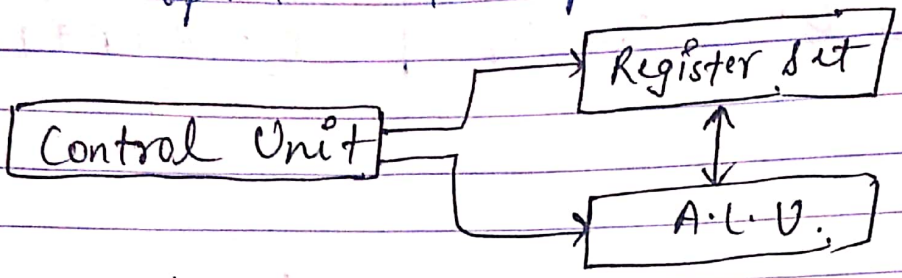


CENTRAL PROCESSING UNIT (C.P.U.)

* The part of the computer that performs the bulk of data processing operations is called the central processing unit (C.P.U.). The C.P.U. is made up of three major parts - A.L.U., C.U., Register Set. The Register Set store intermediate data used during the execution of the instructions. The A.L.U., performs the

Scanned by CamScanner

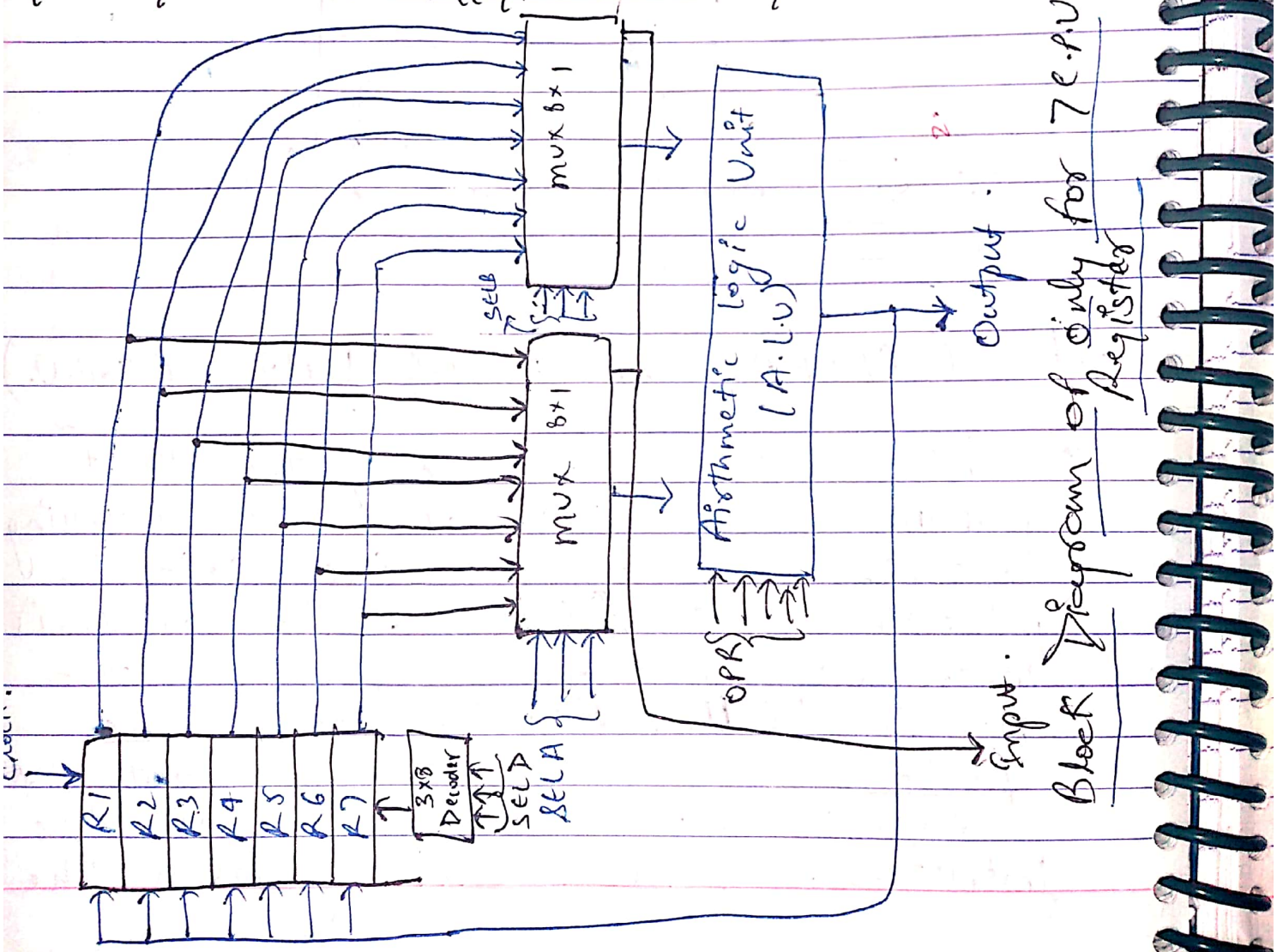
required micro operations for executing the instruction. The Control Unit supervises the transfer of information among the registers & instructs the A.L.U. as to which operation to perform.



Block Diagram of C.P.U

M. Imp

GENERAL REGISTERS ORGANIZATION -



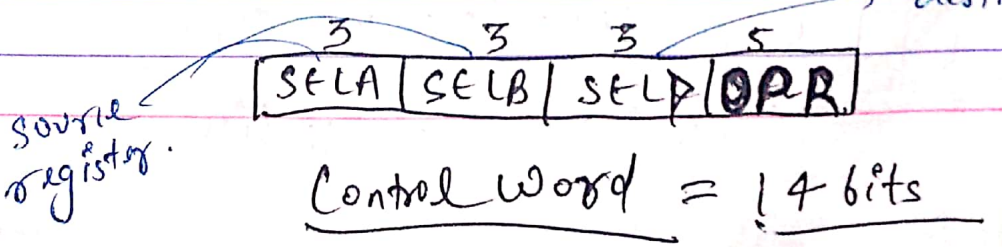
Block Diagram of only for 7 C.P.U Registers

Scanned by CamScanner

Since, Memory locations are needed for storing pointers, counters, return addresses, temporary result, & partial products during multiplication. Since, Memory Access is the most time consuming operation in a computer, it is more convenient & more efficient to store these intermediate value in the processor registers. The register communicate with each other & not only for direct data transfer, but also while performing various micro operations. Hence, it is necessary to provide a common unit that can perform all the arithmetic, logic & shift micro operations in the processor.

CONTROL WORD

The control unit initiates a series of steps of micro operations. At any certain point of time some micro operations are to be initiated, while other remain idle. A string of individual bits that represent the various control signals known as a control word. Control word can be programme to perform various micro operations on the components of the system. Each control word is a complete unique combination of bits.



Control Word = 14 bits

Scanned by CamScanner

ex- $R1 \leftarrow R2 + R3$.

R_1	R_2	R_3	OPR
010	011	001	00100

Let + => 00100.

Exercise -

Q A bus organized C.P.U. has 16 register with 32 bit each, an A.L.U. & a destination decoder.

(a) How many multiplexers are their in the A.B Bus & what is the size of each multiplexer?

(b) How many selection inputs are needed for MUX A & MUX B?

(c) How many inputs & output are their in the decoder??

(d) How many inputs & outputs are Their in the A.L.U. for data, including input & output ~~carries~~ ?. carries?

(e) formulate a control word for the system assuming that the A.L.U has 35 operation.

(a) Solution - (a) 32, 16 x 1, A.

(b) 4 Selection lines

(c) 4 x 16 decoder

(d) ~~16~~ inputs & 32+1 ~~output~~ (input) $32+32+1 = 65$ | output = 32 H = 35

(e)

SELA	SELB	SELD	OPR
4	4	4	6

Control Word = 18 bits

(Size of A Reg. 32 + Size of B Reg. 32 + Operation bits 6.)

Scanned by CamScanner

Solution- $R1 \leftarrow R5 - R6$

0101	0110	000	111100

let -> 111100.

Q Specify the control word that must be applied to the processor to implement the following micro-operations.

(a) $R_1 \leftarrow R2 + R3$. 010011000100100

(b) $R_4 \leftarrow R_5$ (Transfer) 101xxx1000000000

(c) $R_5 \leftarrow R_5 - 1$ (decrement) 101xxx101 01110

(d) $R_6 \leftarrow$ shR1 001xxx110 01111

(e) $R_7 \leftarrow$ input . 000xxx111 00000

Solution Given- input = 000 & output = 000.

R1 → 001

R2 → 010

⋮

R_7 → 111

3	3	3	5
SELA	SELB	SELD	OPR

← 14 →

Scanned by CamScanner

STACK ORGANIZATION -

* REGISTER STACK -

A stack can be placed in a portion of a large memory or it can be organized as a collection of a finite set of memory words or registers. The stack pointer register (SP) consist a binary number whose value is equal to the address of the word is currently on top of the stack.

The one bit register (FULL) is said to be one when the stack is full. & the one bit register (EMPTY) is said to one when the stack is empty. Initially, Stack pointer is cleared to 0, & full is cleared to 0. & empty is said to 1.

* PUSH OPERATION -

(SP ← SP+1) Increment stack pointer.
(M[SP] ← DR) Write item on top of the stack.
(If (SP = 0) then (FULL ← 1)) check if stack is full
(EMPTY ← 0) Mark the stack not empty

* POP OPERATION -

(DR ← M[SP]) Read item from the top of the stack.

Scanned by CamScanner

(SP ← SP - 1) Decrement the stack pointer.
(If (SP = 0) then (EMPTY ← 1)) check if stack is empty
(FULL ← 0) Mark the stack not full.

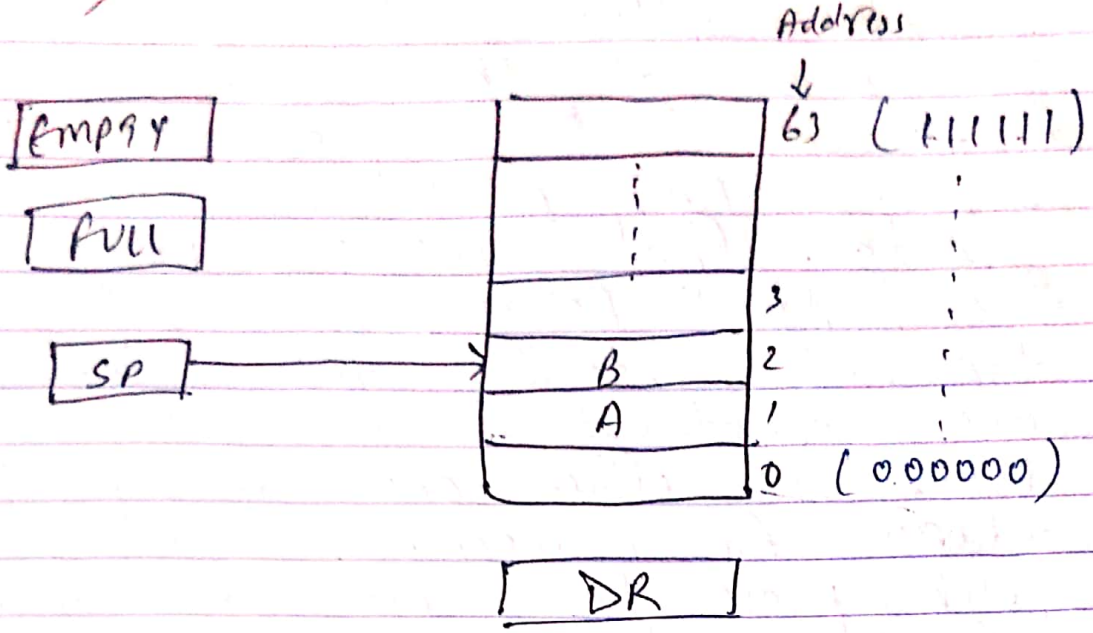


Fig:- Block Diagram of a 64(2^6) words stack.

:-Q If SP = 000000 in the stack. How many items are their in the stack if
(a) FULL = 1 & EMPTY = 0.
(b) EMPTY = 1 & FULL = 0.

Answer: (a) 64
(b) 0.

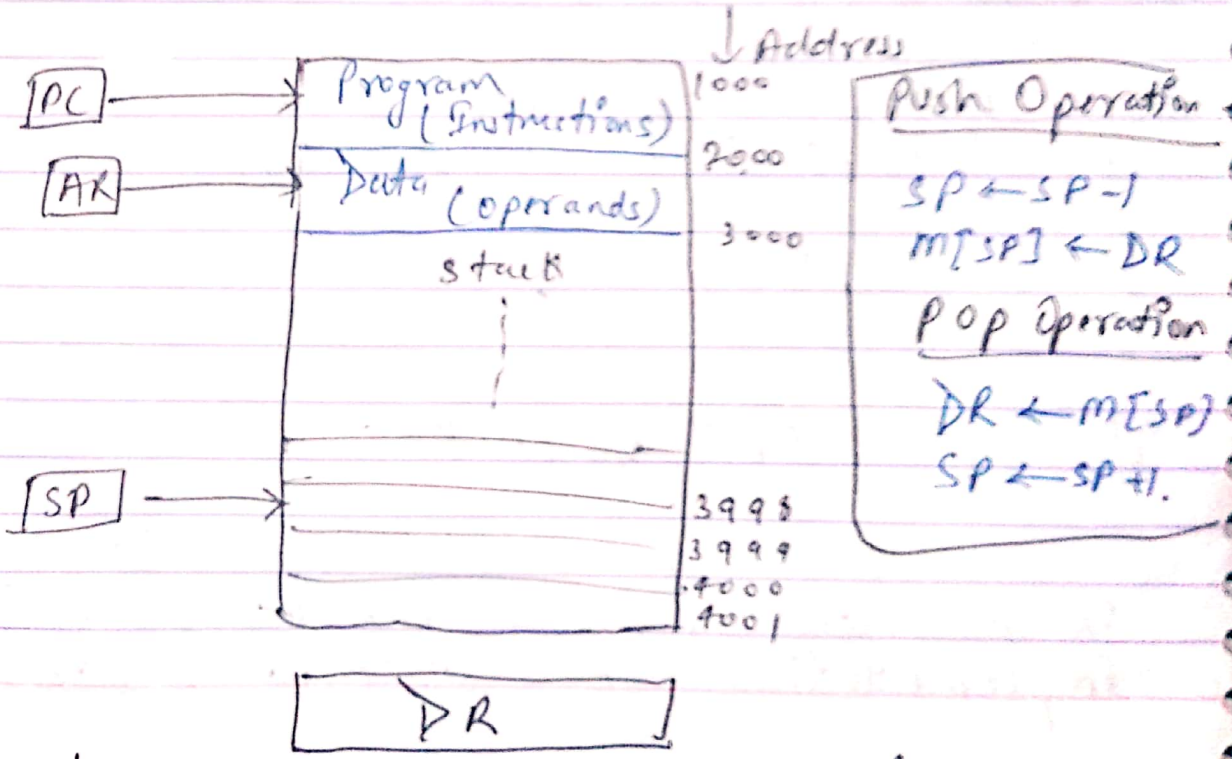
MEMORY STACK-

A ~~register~~ stack can be implemented in a Random Access Memory (RAM). The implementation of stack in the C.P.U.

Scanned by CamScanner

Instruction → { fetch phase, decoder ", execute " }

is done by assigning a portion of memory to a stack operation & using a processor register as a stack Pointer. The programe counter (PC) points at the address of the next instruction in the programe. The address register (AR) points at an array of data. That means, hold base addrss The Stack Pointer (SP) points at a top of stack. The three register are connected to a common address bus, an either one can provide an address for memory. PC is used in fetch phase to read an instruction. AR is used during the execute phase to read an operand (data). SP is used to push or pop items into or from the stack.



Eg:- Compute Memory with data, code & stack segments

Scanned by CamScanner

Infix to Postfix. → DS.

(Imp) Exercise -

Q The content of the top of a memory stack is 5320. The content of the stack pointer SP is 3560. A two word called sub-routine (function) instruction is located in memory at address 1120 followed by the address field of 6720 at location 1121. What are the content of programme counter (PC), SP & the top of the stack?

(a) Before the call instruction is fetched from memory.

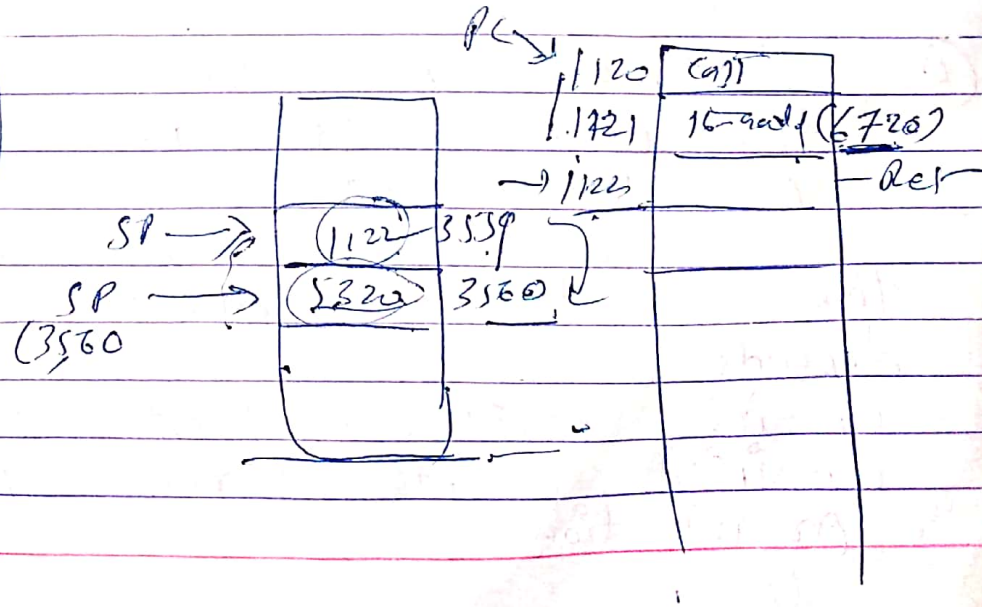
(b) After the call instruction is executed.

(c) After the return from sub-routine.

Hint - CALL 16-bit address
RET

SP ← SP - 1
M[SP] ← PC
PC ← effective-address.

	PC	SP	Top
		3560	5320
(i)	1120	3559	1122
(ii)	6720	3559	5320
(iii)	1122	3560	5320



Scanned by CamScanner

INSTRUCTION FORMAT -

Operand	Operation	mode

In instruction format the bits of the instruction are divided into groups called fields. The most common fields found in instruction format are -

(a) An operation code field that specifies the operation to be performed

(b) An address field that designates a memory address or a processor register.

(c) The mode field that specifies the way the operand or the effective address is determined.

The number of address field in the instruction format of a computer depends on the internal organization of its register. Most computers are classified into three types of C.P.U. organization -

Scanned by CamScanner

(i) Single Accumulator Organization -

All operations are performed with an implide accumulator register. The instruction format in this type of computer uses <u>one address field</u>.

ex- ADD x, where x is the address of the operand or effective address.

ADD x ⇒ A ← A + M[x].
(or AC)

ex- SUB x ⇒ AC ← AC + M[x].

(ii) General Register Organization -

The instruction format in this type of computer needs <u>three</u> or two <u>register address field</u>.

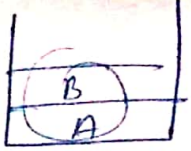
ex- ADD R1, R2, R3 ⇒ R1 ← R2 + R3
ADD R1, R2 ⇒ R1 ← R1 + R2.

address field

(iii) Stack Organization -

Computers with stack organization would have PUSH & POP instructions which requires an address field. The Stack Pointer is updated automatically. Operation Type instructions do not need an address field in stack orgonized

Scanned by CamScanner



computers. This type of instructions are zero address field instructions.

ex- (i) PUSH x.
(ii) ADD. , MUL, SUB, XOR.

This operation has the effect of popping the two top numbers from the stack, adding the numbers & pushing the sum into the stack.

Exercise-

Q WAP to evaluate the arithmetic statement $X = (A+B) * (C+D)$

(a) Using a general register computer with 3- address instruction.
(b) With 2- address instruction.
(c) Using a accumulator type computer with one address instruction.
(d) Using a stack organized computer with zero - address operation instruction.

Solution-

Answer (a)

memory word location

```
ADD  R1, A, B : R1 ← M[A] + M[B]
ADD  R2, C, D : R2 ← M[C] + M[D]
MUL  R3, R1, R2 : R3 ← R1[R1] + M[R2]
MUL  X, R1, R2 : M[X] ← R1 * R2.
```

Answer (b)

```
MOV  R1, A :  R1 ← M[A].
MOV  R2, B :  R2 ← M[B]
ADD  R1, R2 :  R1 ← R1 + R2.
```

Scanned by CamScanner

```
MOV   R3, C  :  R3 ← m[C]
MOV   R4, D  :  R4 ← m[D]
ADD   R3, R4 :  R3 ← R3 + R4
MUL   R1, R3 :  R1 ← R1 * R3.
MOV   X, R1  :  m[X] ← R1.
```

Answer - (c)

```
LOAD   A :  AC ← m[A].
ADD    B :  AC ← AC + m[B].
STORE  T :  m[T] ← AC
LOAD   C :  AC ← m[C].
ADD    D :  AC ← AC + m[D]
MUL    T :  AC ← AC * m[T].
STORE  X :  m[X] ← A
```

Answer - (d)

Postfix conversion of ((A+B) * (C+D)) is AB+CD+*.

```
PUSH   A :  TOP ← m[A].
PUSH   B :  TOP ← m[B].
ADD     :  TOP ← m[A] + m[B]
PUSH   C :  TOP ← m[C].
PUSH   D :  TOP ← m[D].
ADD     :  TOP ← m[C] + m[D].
MUL     :  TOP ← (A+B) * (C+D).
POP    X :  m[X] ← TOP.
```

Scanned by CamScanner

Q = $X = \left(\frac{(A - B + C * (D * E - F))}{(G + H * K)} \right)$.

(a)

```
MUL  R1, D, E : R1 ← M[D] * M[E].
SUB  R2, R1, F : R2 ← R1 - M[F].
MUL  R3, C, R2 : R3 ← M[C] * R2.
SUB  R4, A, B  : R4 ← M[A] - M[B].
ADD  R5, R3, R4 : R5 ← R3 + R4.
MUL  R6, H, K  : R6 ← M[H] * M[K].
ADD  R7, G, R6 : R7 ← M[G] + R6.
DIV  X, R5, R7 : M[X] ← R5 / R7.
```

(b)

```
MOV  R1, E  : R1 ← M[E].
MOV  R2, D  : R2 ← M[D]
MUL  R1, R2 : R1 ← R1 * R2.
MOV  R3, F  : R3 ← M[F]
SUB  R1, R3 : R1 ← R1 - R3.
MOV  R4, C  : R4 ← M[C].
MUL  R1, R4 : R1 ← R1 * R4.
MOV  R5, A  : R5 ← M[A]
MOV  R6, B  : R6 ← M[B].
SUB  R5, R6 : R5 ← R5 + R6.
ADD  R5, R1 : R5 ← R5 + R1.
MOV  R7, H  : R7 ← M[H].
MOV  R8, K  : R8 ← M[K]
MUL  R7, R8 : R7 ← R7 * R8.
MOV  R9, G  : R9 ← M[G].
ADD  R7, R9 : R7 ← R9 + R7
DIV  R5, R7 : R5 : R5/R7.
```

Scanned by CamScanner

$(((A-B)+(C*((D*E)-F)))/(G+(H*K)))$

(c)

LOAD	D	:	AC ← m[D].
MUL	E	:	AC ← AC * m[E]
SUB	F	:	AC ← AC - m[F]
MUL	C	:	AC ← m[C] * AC.
STORE	T	:	m[T] ← AC.
LOAD	A	:	AC ← m[A]
SUB	B	:	AC ← AC - m[B].
ADD	T	:	AC ← AC + m[T].
STORE	T1	:	m[T1]
LOAD	H	:	AC ← m[H]
MUL	K	:	AC ← AC * m[K].
ADD	G	:	AC ← AC + m[G].
STORE	T2	:	m[T2] ← AC
LOAD	T1	:	AC ← m[T1]
DIV	T2	:	~~DIV T2~~ AC ← AC/m[T2].
STORE	X	:	~~STORE~~ m[X] ← AC.

(d) AB - CDE * F - * + G HK * + /

PUSH A
PUSH B
SUB
PUSH C
PUSH D
PUSH E
MUL
PUSH F
SUB
MUL
ADD
PUSH G
PUSH H
PUSH K

~~PUSH~~ MUL
ADD
DIV

Scanned by CamScanner

ADDRESSING MODES -

The operation field of an instruction describes the operation to be performed. This operation must be executed on some data stored in C.P.U. registers or memory words. The way the operands are chosen during program execution is depends on the addressing modes of the instruction. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

* IMPLIED ADDRESSING MODE -

In this mode, the operands are specified implicitly in the instruction.

ex- CMA : complement the content of accumulator.

ADD } Zero address
SUB } stack - organization computer.

* IMEDIATE ADDRESSING MODE -

In this mode, the operand is specified in the instruction itself.

ex MVI B, 45 : B ← 45.
(45 → operand.)

Scanned by CamScanner

* REGISTER ADDRESSING MODE -

In this type of mode, the operands are in registers that resides within the C.P.U.

ex- ADD R1, R2 : R1 ← R1 + R2.

* REGISTER INDIRECT ADDRESSING MODE -

In this type of mode, the instruction specifies a register whose content gives the address of the operand in memory.

ex- ADD R1, R2 : R1 ← R1 + M[R2].

* DIRECT ADDRESSING MODE -

In this mode, the effective address (the address of the operand) is equal to the address part of the instruction.

ex- ADD R1, X : R1 ← R1 + M[X].

* INDIRECT ADDRESSING MODE -

In this mode the address field of the instruction gives the address where the effective address is stored in memory.

Scanned by CamScanner

Ex: ADD R1, X ; R1 ← R1 + M[M[X]].

* RELATIVE ADDRESSING MODE -

In this mode the content of the programe counter is added to the address part of the instruction to obtain the effective address.

effective address = content of PC + address field value.

* INDEXED ADDRESSING MODE -

In this mode the content of the indexed register is added to the address part of the instruction to obtain the effective address

effective address = content of indexed reg. + address field value

* BASE REGISTER ADDRESSING MODE -

In this mode the content of base register is added to the address part of the instruction to obtain the effective address.

effective address = content of base reg. + address field value.

Scanned by CamScanner

* AUTOINCREMENT & AUTODECREMENT ADDRESSING MODE.

This is similar to the register indirect addressing mode except that the register is incremented or decremented after or before its value is used to access memory.

ex- post ADD R1, R2 : R1 ← M[R2], R2 ← R2+1

pre → ADD R1, R2 : R2 ← R2-1, R1 ← R1+M[R2]

EXERCISE -

Q

PC = 200

R1 = 400

XR = 100

AC

Add	Memory
200	LOAD AC
201	500
202	NEXT INSTRUCTION
	⋮
399	450
400	700
	⋮
500	800
	⋮
600	900
	⋮
702	325
	⋮
800	300

LOAD AC, 500

Relative

PC = 200

if 0 or 1 addressing

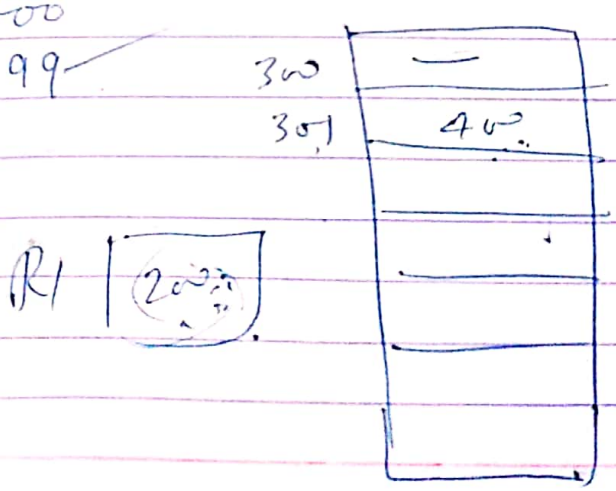
Scanned by CamScanner

	Effective Address	Content of AC.
Direct	500	800
Immediate	201	500
Indirect	800	300
Relative	202+500=702	325
Indexed	100+500=600	900
Register	—	400
Reg. Indirect	400	700
Auto Increment	400	700
Auto Decrement	399 (401)	450
Base Register	400+500=900	

In case of register there is no register

Q An instruction is stored at location 300 with the address field at location 301. The address field has the value 400. The processor register R1 contains the number 200. Evaluate the effective address of the following mode of the instruction is -

(a) direct — 400
(b) immediate — 301
(c) relative — 302+400 = 702
(d) register indirect — 200
(e) indexed with R1 as the index register — 600 (200+400)
(f) register — NULL
(g) Auto increment — 200
(h) Auto decrement — 199



Scanned by CamScanner