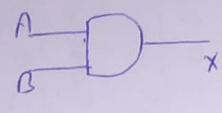
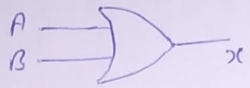
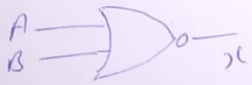
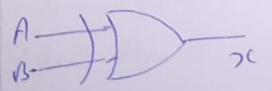
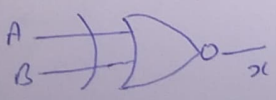
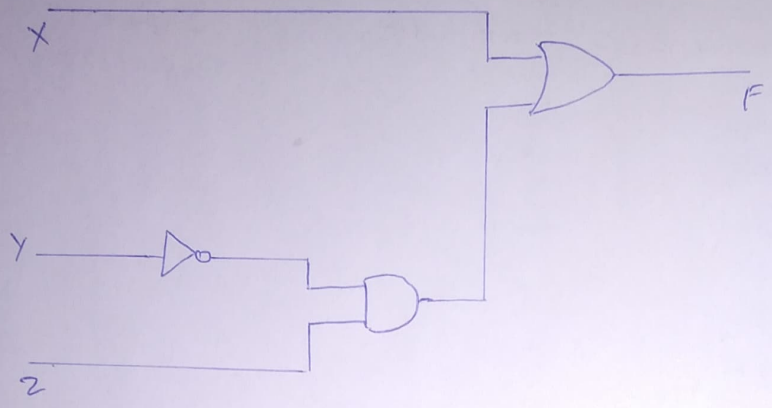
Name - Sumit Kumar
Class - MCA I st
Date -

13.10

NAME	Graphic Symbol	Algebric function	Truth table	Input sensitivity
AND		$x = A \cdot B$ or $x = AB$	A B x 0 0 0 0 1 0 1 0 0 1 1 1	0
OR		$x = A + B$	A B x 0 0 0 0 1 1 1 0 1 1 1 1	1
Inverter		$x = A'$	A x 0 1 1 0	Not Applicable
Buffer		$x = A$	A x 0 0 1 1	Not Applicable
NAND		$x = (AB)'$	A B x 0 0 1 0 1 1 1 0 1 1 1 0	0
NOR		$x = (A+B)'$	A B x 0 0 1 0 1 0 1 0 0 1 1 0	1
Exclusive-OR (XOR)		$x = A \oplus B$ or $x = A'B + AB'$	A B x 0 0 0 0 1 1 1 0 1 1 1 0	Not Applicable
Exclusive NOR		$x = (A \oplus B)'$ or $x = A'B + AB'$	A B x 0 0 1 0 1 0 1 0 0 1 1 1	Not Applicable

1) $F = X + \bar{Y}Z$

To solve the above equation we use the three gates namely NOT, AND

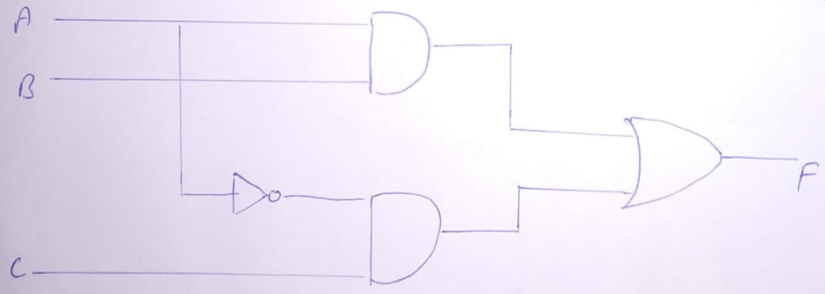


X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	ϕ	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Truth table

2) $F = AB + \bar{A}C$

To solve the above eqn we use 4 logic gates 2 AND NOT OR



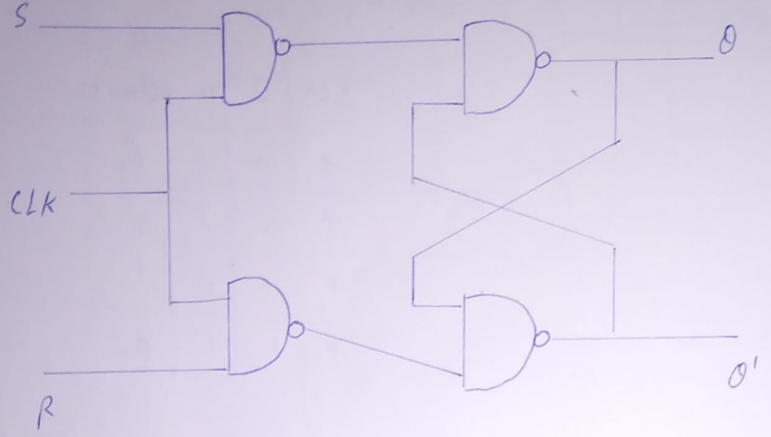
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Truth Table

ame - Sumit Kumar
lass - MCA I^st
Date -

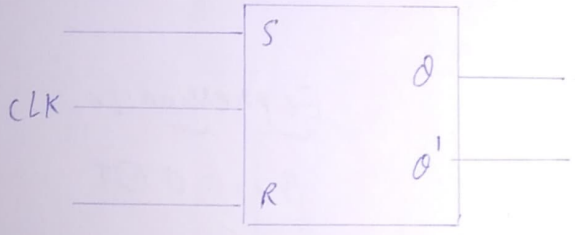
Implementing the flip-flop

(i) S-R Flip Flop



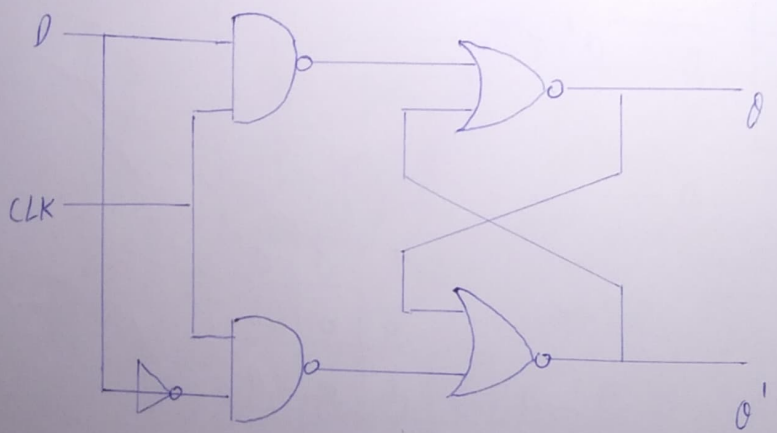
Expression :-

$Q_{n+1} = S + RQ_n$



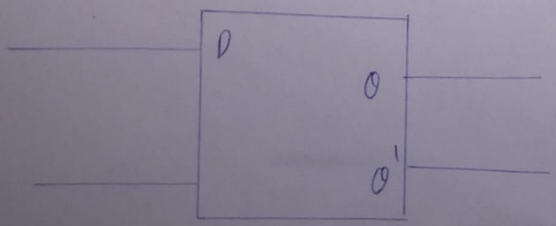
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Invalid

(ii) D-Flip Flop



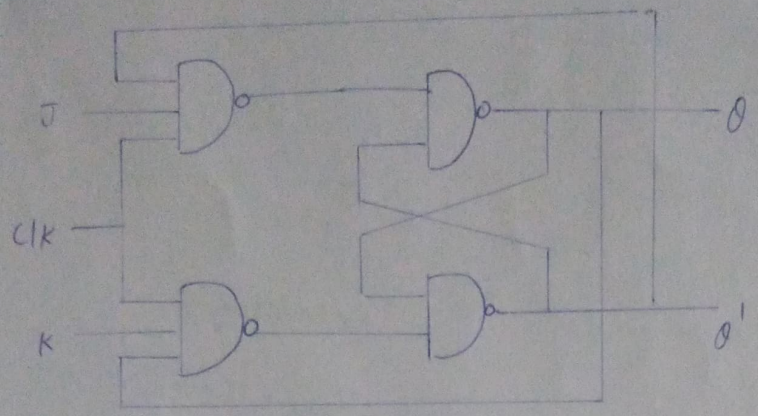
Expression :-

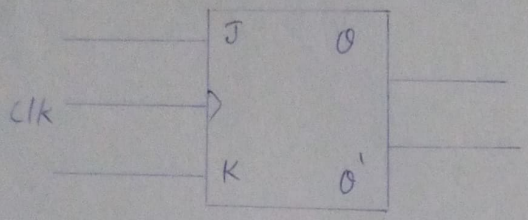
$Q(t+1) = D$



Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

(iii) J-K Flip Flop :-



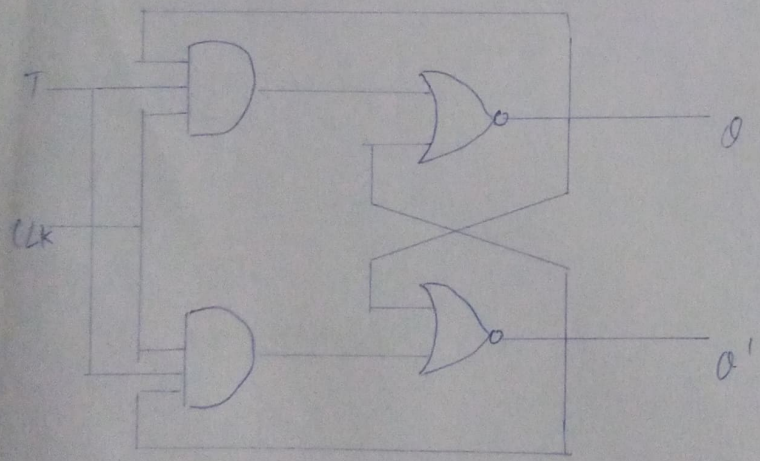


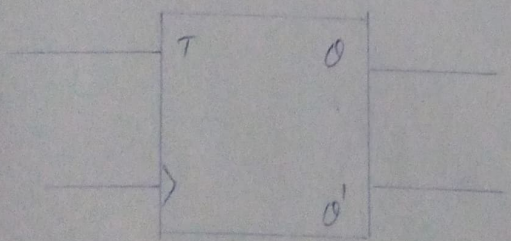
Expression :-

$Q_{n+1} = JQ_n' + K'Q_n$

Clk	J	K	Q_{n+1}
↑	0	0	Q_n
↑	0	1	0
↑	1	0	1
↑	1	1	Q_n'

(iv) T-Flip Flop :-





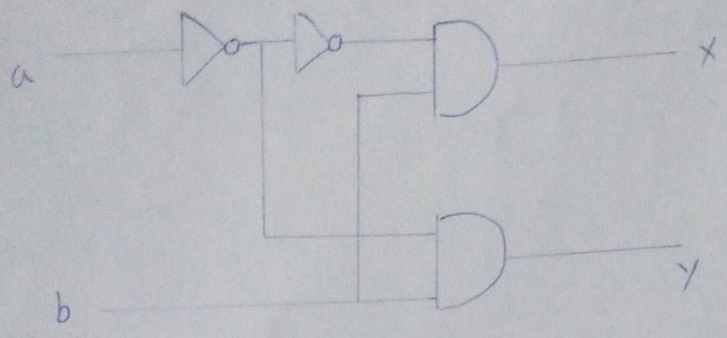
Expression :-

$Q_{+1} = Q \oplus T$

T	Q	Q'
0	0	0
1	0	1
0	1	0
1	1	0

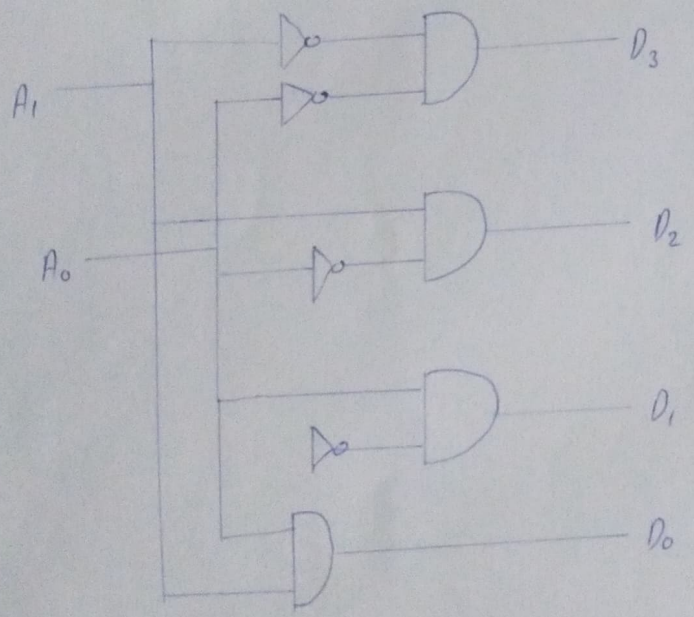
ne - Sumit Kumar
ss - MCA Ist
ate -

1. Implementing 1 to 2 Decoder



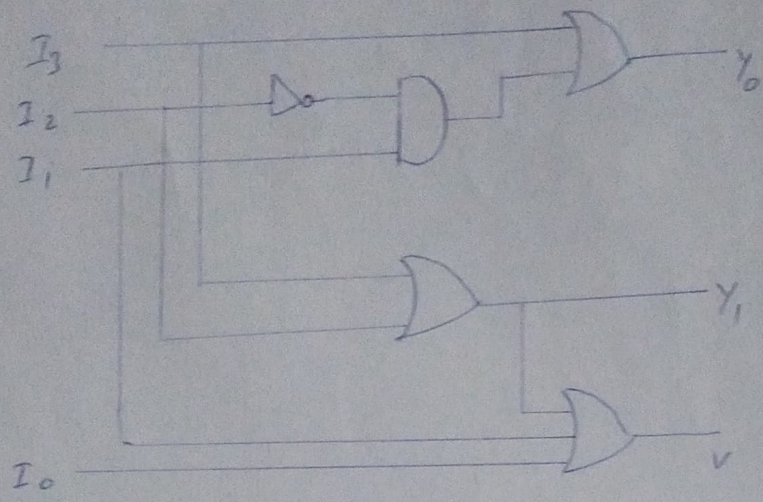
a	b	x	y
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

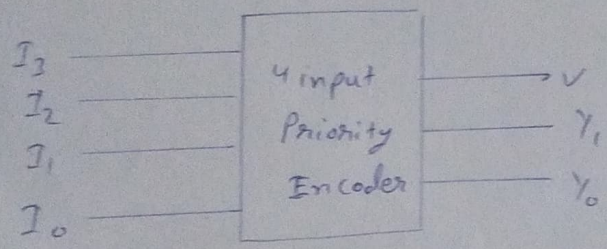
2. Implementing 2 to 4 Decoder



A_1	A_0	D_3	D_2	D_1	D_0
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

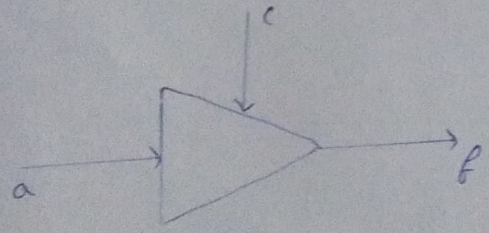
3. Implementing 4 input Priority Encoder





I_3	I_2	I_1	I_0	Y_1	Y_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

4. Tri-State Buffer



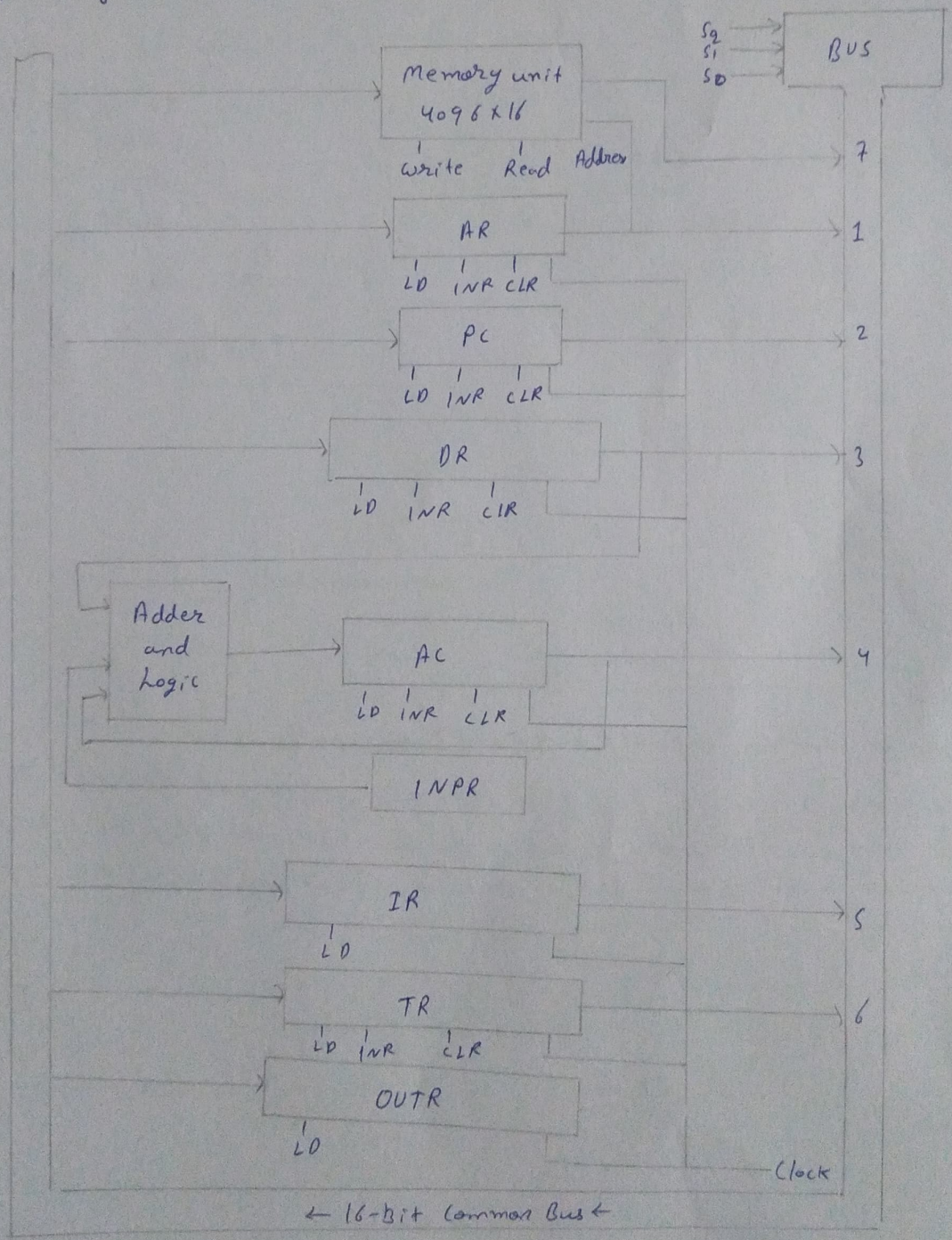
c	a	f
0	0	Z
0	1	Z
1	0	0
1	1	1

Name - Sumit Kumar
Class - MCA 1st year

2/11

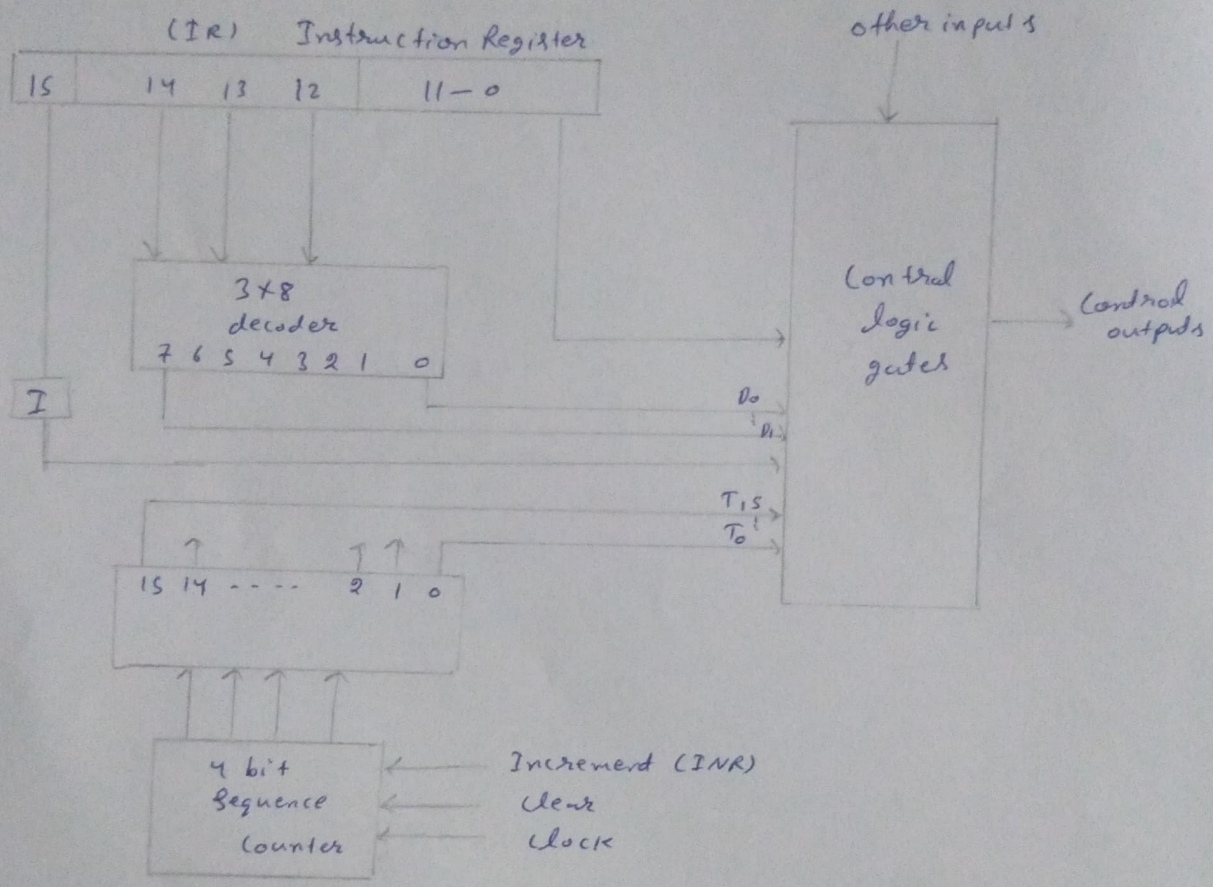
Basic Computer Registers Connected to a Common Bus

A Basic computer has eight registers, a memory unit, and a control unit (to be presented) Paths must be provided to transfer information from one register to another and between memory and register. the no of wires will be excessive if connections are made between the outputs of each register and the input of the other registers.



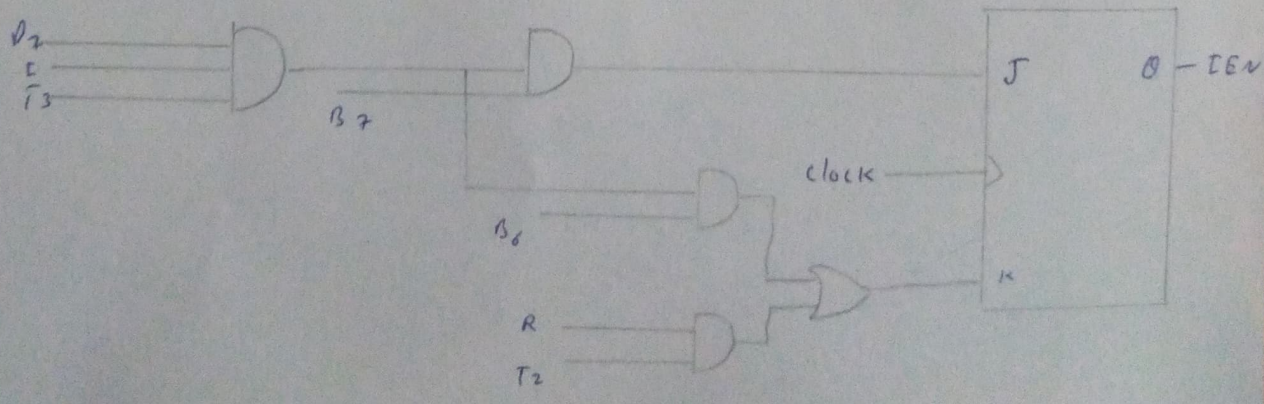
Design of Control Unit of Basic Computer

It consists of 2 decoder, a sequence counter and a number of control logic gates An instruction read from memory is placed in the instruction register (IR). The operation code in bits 12 through 14 are decoded with 3×8 decoder. the eight outputs of the decoder are the symbols D_0 through D_7. The subscripted decimal no is equivalent to the binary value of corresponding operation code.



Design a control input for IEN (Interrupt Enable)

the interrupt enable flip-flop IEN can be set and cleared with two inst when IEN is cleared to 0 (with the IOF inst) the flag can't interput th computer. when IEN is set to 1 (with the ION instruction) the computer can be interrupt

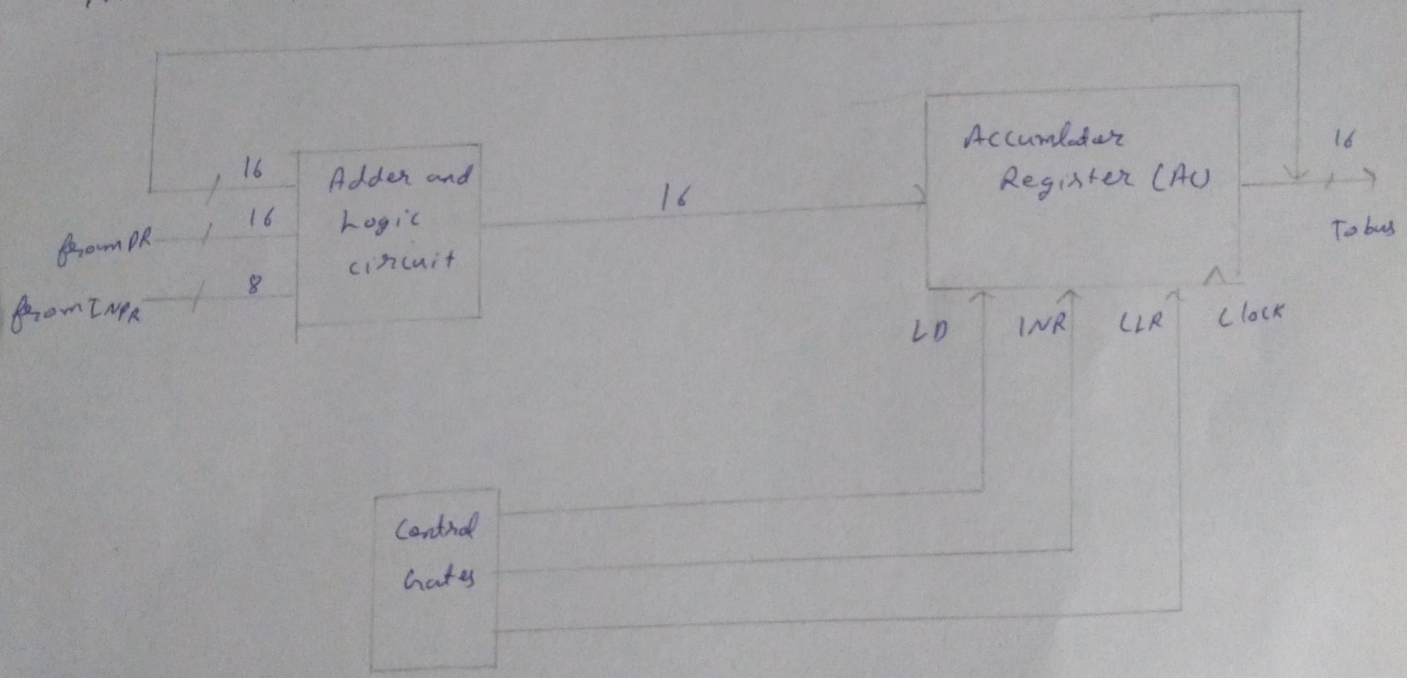


me - Sumit Kumar

lass - MCA Ist year

Design a circuit Associated with AC

the adder and logic circuit has 3 sets of inputs one set of 16 inputs comes from the outputs of AC. Another set of 16 inputs comes from the data register, DR A third set of 8 inputs comes from the input register INPR. the outputs of the adder and logic circuit provide the data inputs for the register. In addition is necessary to include logic gates for controlling LD, INR and CLR in the register as well as in operations of the logic circuits.



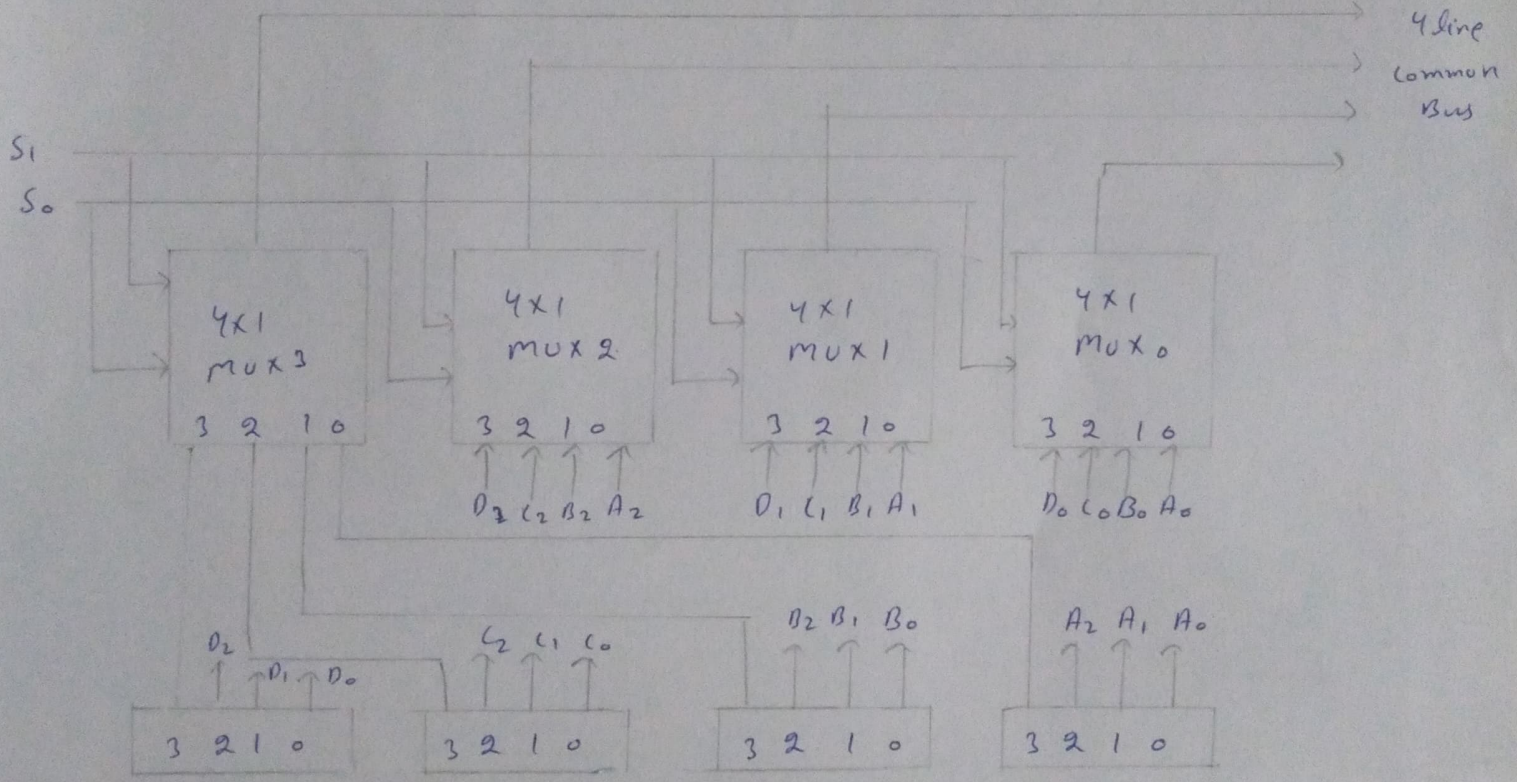
Name - Sumit Kumar

Class - MCA I St year

2/11

Four Register Bus System

An efficient scheme for transferring information between registers in a multiple-register conf is a common bus system. the transfer of info from a bus into one of many destination registers can be accomplished by connecting the Bus lines to the inputs of all destination registers and activating the load control of the particular destination registers selected.



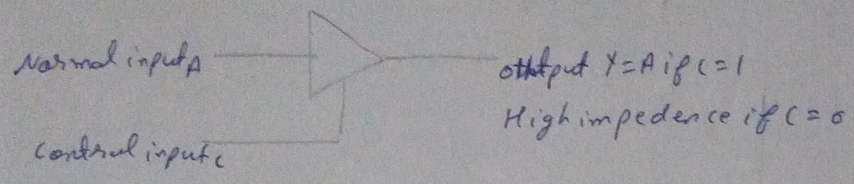
Bus system for 4 Register

S_1	S_0	Register Selected
0	0	A
0	1	B
1	0	C
1	1	D

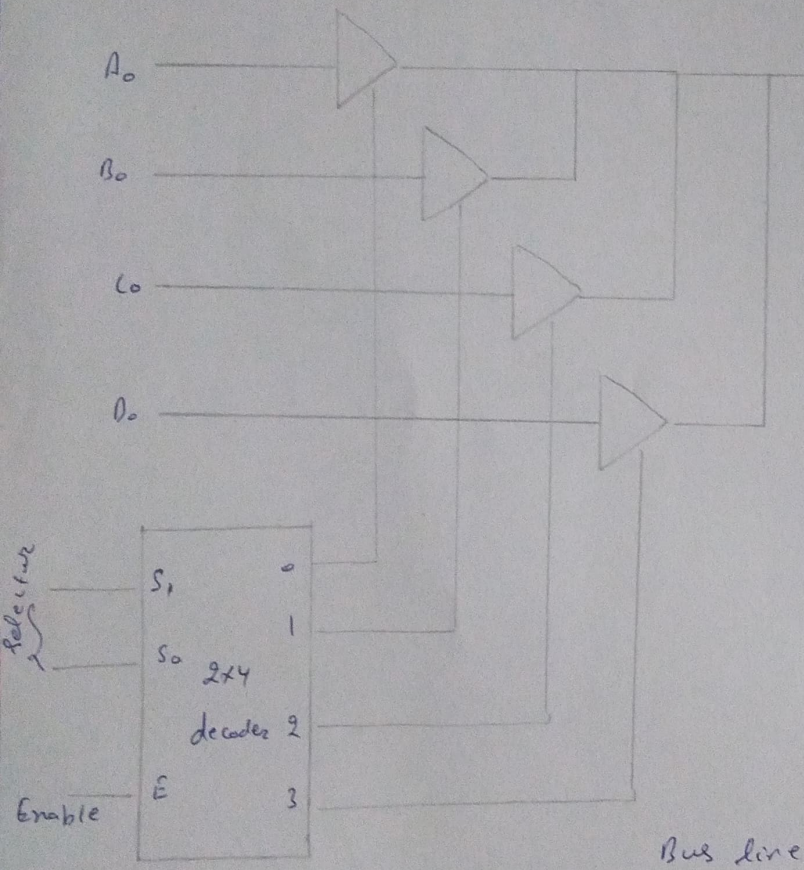
Three state Bus Buffers

A three state gate is a digital circuit that exhibits 3 states. Two of the states are signals equivalent to logic 1 & 0 as a conventional gate. The third state is a high impedence state. The high-impedence state

behaves like an open circuit. which means that the output is disco... and does not have logic significance. 3 state gate may perform any conventional logic, such as AND or NAND. However the one commonly used in design of a bus system is the buffer gate



Enable	I N	OUT
0	0	Hi-z
0	1	Hi-z
1	0	0
1	1	1

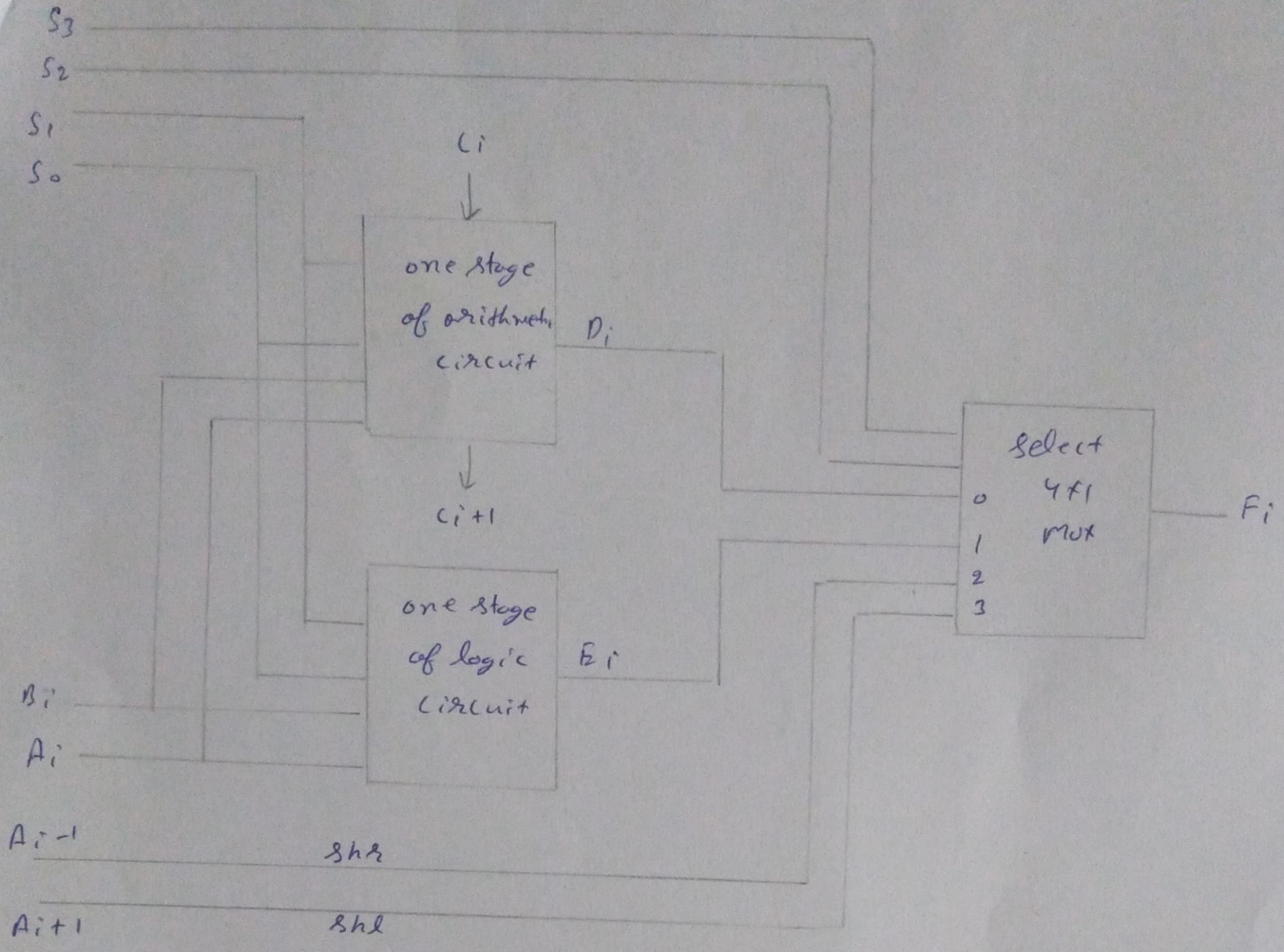


Bus line with 3 state Buffer

One stage of arithmetic logic shift unit

To perform a microoperation the contents of specified registers are placed in the inputs of the common ALU. the ALU performs an operation and the result of the operation is then transferred to a designation register. The ALU is a combinational circuit so that the entire register transfer operation from the source registers through the ALU and into the destination register can be performed during one clock pulse period. The shift microoperations are often performed in a seperate unit, but sometime the shift unit is made part of ALU

le - Sumit Kumar
alls - MCA 1st year



one stage of Arithemic Logic shift unit

S_3	S_2	S_1	S_0	C_{in}	operation	function
0	0	0	0	0	F = A	Transfer A
0	0	0	0	1	F = A+1	inc A
0	0	0	1	0	F = A+B	Addition
0	0	0	1	1	F = A+B+1	Add with carry
0	0	1	0	0	F = A+$\bar{B}$	Subtract with Borrow
0	0	1	0	1	F = A+$\bar{B}$+1	Sub
0	0	1	1	0	F = A-1	Dec A
0	0	1	1	1	F = A	Transfer A
0	1	0	0	x	F = A∧B	AND
0	1	0	1	x	F = A∨B	OR
0	1	1	0	x	F = A⊕B	XOR
0	1	1	1	x	F = $\bar{A}$	Complemet A
1	0	x	x	x	F = shrA	Shift right A into F
1	1	x	x	x	F = shlA	Shift left A into F

Name - Sumit Kumar
Class - MCA 1st year
Sub - CSA

Ans 1

Five Example of External Interrupt

i) Button Press
ii) Sensor Input
iii) Communication Ports
iv) Time overflow
v) External Hardware Signal

Five Example of Internal Interrupt

i) Timer Compare Match
ii) Watchdog Timer
iii) ADC Conversion Complete.
iv) Serial Communication Receive Buffer
v) Software Interrupts

Difference b/w Software Interrupt & Subroutine call

Software Interrupt	Subroutine call
1. It is triggred by software Interactions often used for system call	1. It is a programming technique for reusing code.
2. Software Interrupt handle specific tasks initiated by the program	2. It invoke separate code modules for organized and reusable functionality.

Ans 2

a) To clear the first eight bits you would perform a bitwise AND operation with a mask that has 0s in the first eight positions and 1s elsewhere. the mask would be 1111111100000000 in binary

b) To set the last eight bits to 1, you would perform a bitwise OR operation with a mask that has 1s in the last eight position and 0s elsewhere. The mask would be 0000000011111111 in binary.

c) To complement the middle eight bits you would perform a bitwise XOR operation with a mask that has 1s in the middle eight positions and 0s elsewhere. the mask would be 0000111100001111 in binary.

Ans 3

Interrupt Service Routine:

```
ISR:
If special_occasion_mod = 0xFFFF:
Service_input_device()
if location_mod == 0x0000:
Service_output_device()
Clear_interrupt_flag()
```

special_occasion_mod → represents the special occasion module

location_mod → represents the location module

- service_input_device() and service_output_device() → are functions to handle the input and output devices respectively.
- the condition special_occasion_mod == 0xFFFF and location_mod == 0x0000 check if all bits are set to 1 and 0, resp.

Ans 4

a) A control word for a microoperation typically consists of various fields that control different aspects of the operation, such as the operation to be performed the source and destination registers and any additional control signals

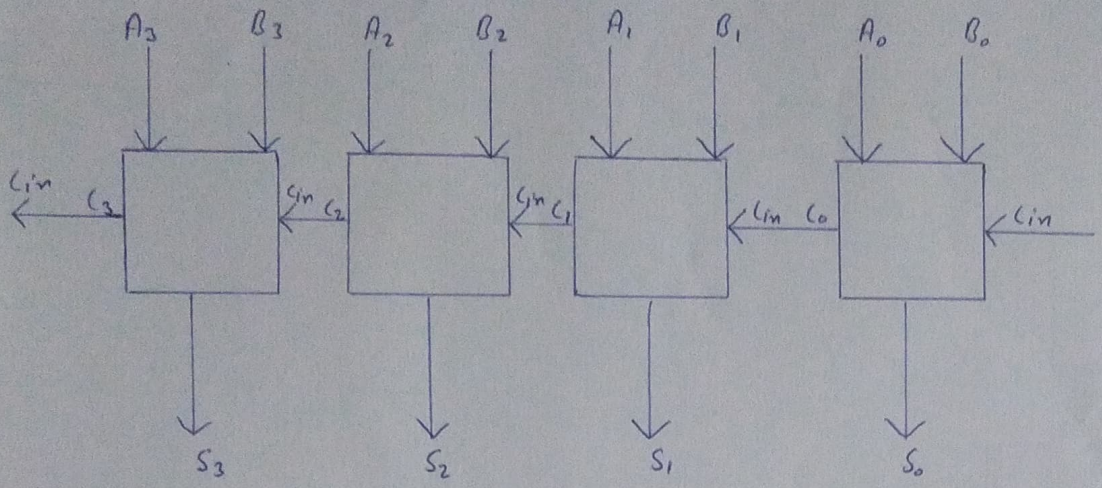
b) the number of bits in each field of the control word would depend on the specific requirements of the computer architecture However, a general encoding schema might include fields such as:

- Operation Code (opcode): To specify the ALU or shifter operation.
- Source Register Address: To identify the source register.
- Destination Register Address: To identify the destination register
- ALU operation controls: Additional bits to control specific ALU operations.
- Shifter operation Control: Additional bits to control specific shifter operation
- Control Signals Bits for various control signals that might be needed.

Name - Sumit Kumar
Class - MCA I^st year

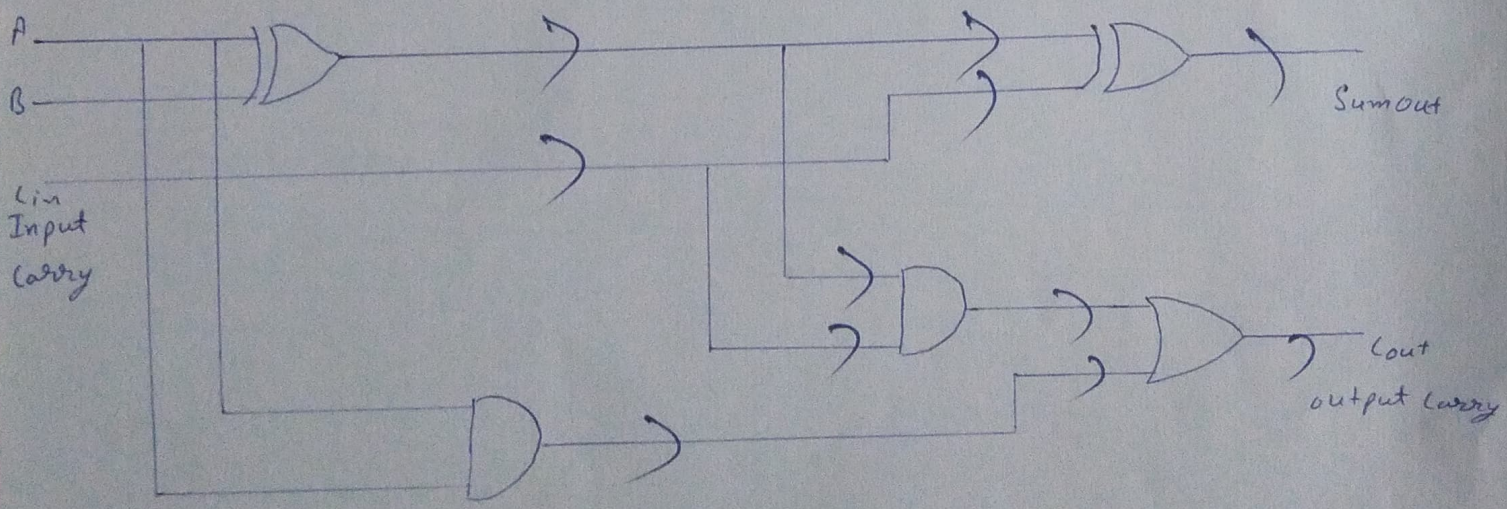
Binary Adder

30.10



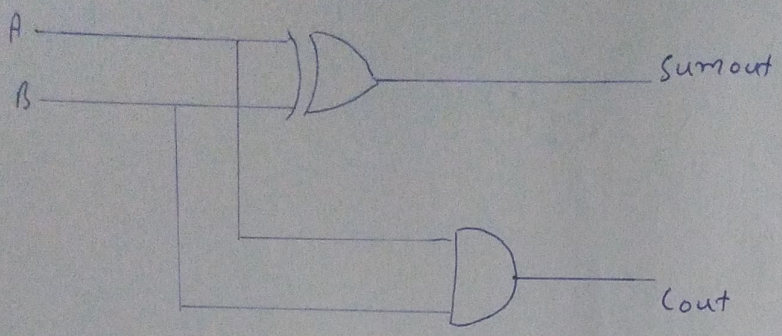
- A, B are augend and addend bits. lower-order bit is defined by subscript 0.
- C0, C1, C2, C3 are the carry inputs. C4 is the carry output produced by the last full-Adder.
- Cout of the first Adder is connected as the Cin of the next full Adder
- S0, S1, S2, S3 are sum outputs that produce the sum of augend and addend bits.

Half Adder



- Cin is the carry bit from the previous stage of binary addition
- Cout is obtained using AND and OR operations
- The sum output of the full adder is obtained by XOR the bits A, B and Cin

Half Adder

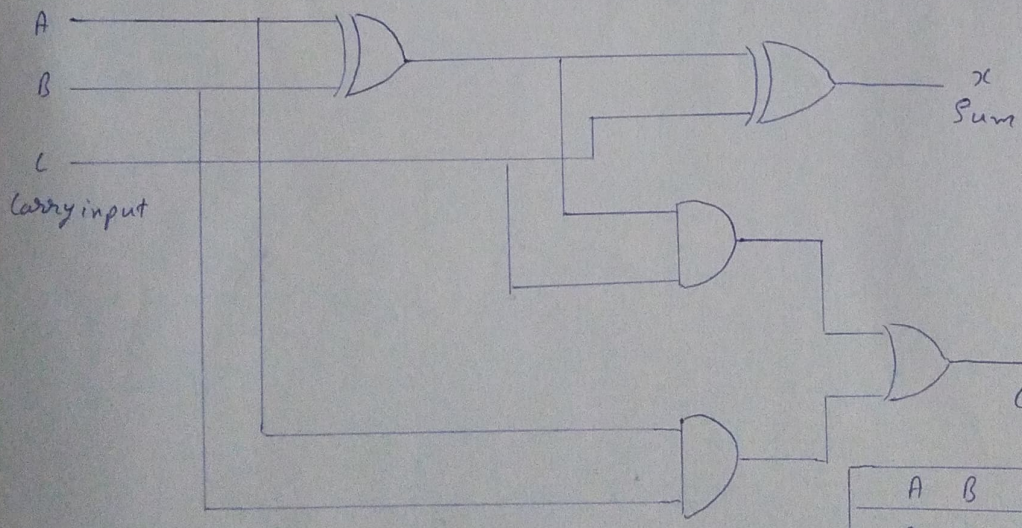


Input		output	
A	B	outc	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Input			ou	
A	B	Cin	Sum	
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- SUM output is the least significant bit (LSB)
- Carry output is the most significant bit (MSB)

Binary Ripple Carry Adder

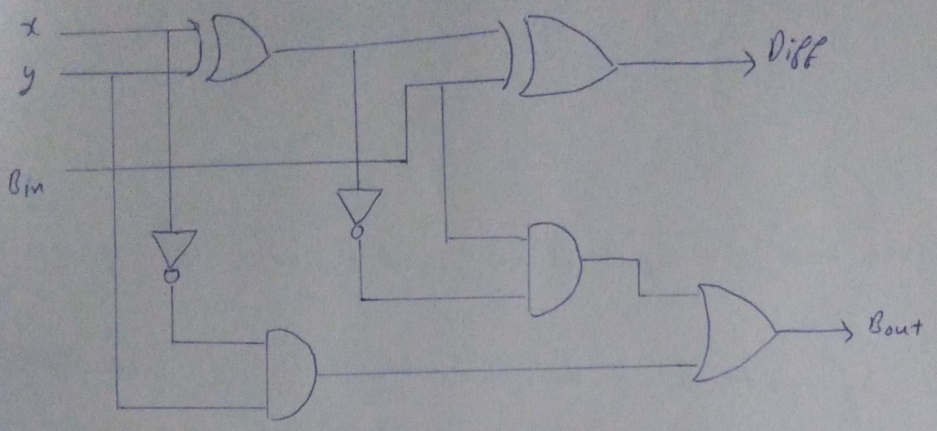


A	B	C	x	y
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- Sumit Kumar

- MCA I st year

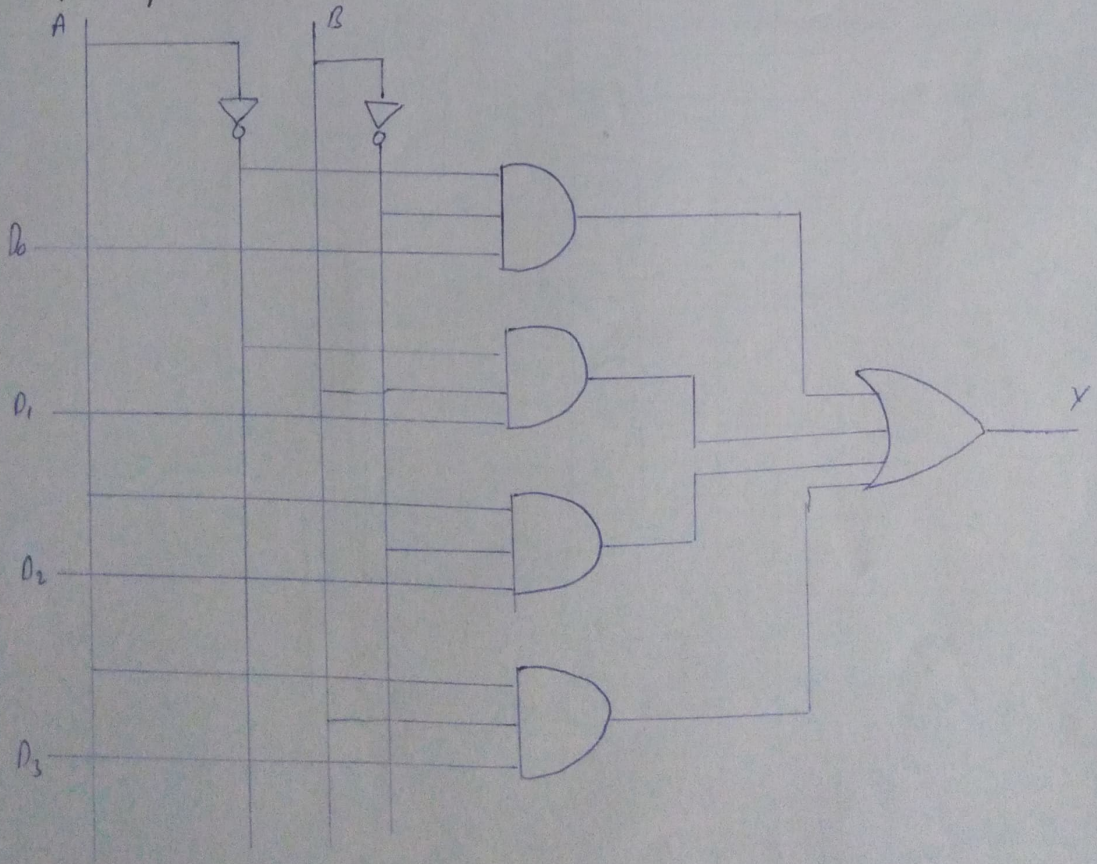
Subtractor (Binary Subtractor)



Bin	x	Y	Diff	Bout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

- Bin → Borrow input from previous circuit
- Bout → Borrow out input for next circuit

4 to 1 Multiplexer Circuit

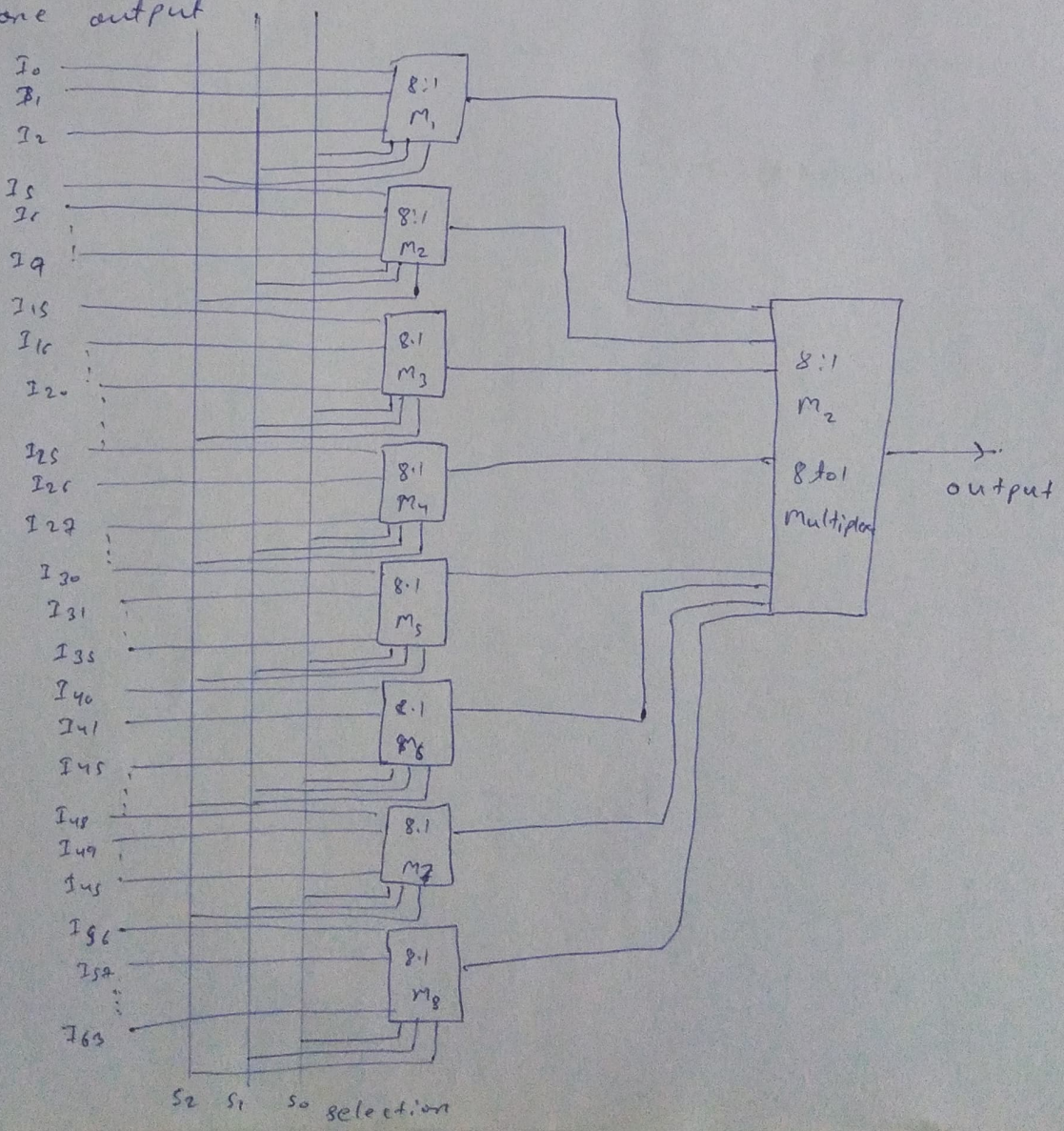


Select		output
A	B	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

64 to 1 Multiplexer

The 64 to 1 multiplexer has 64 input lines and 6 selection lines and one output.

It can be created by putting 8 to 1 multiplexer in parallel and then connecting to other 8 to 1 multiplexer in series so that overall there will be 64 input lines 6 selection lines and one output

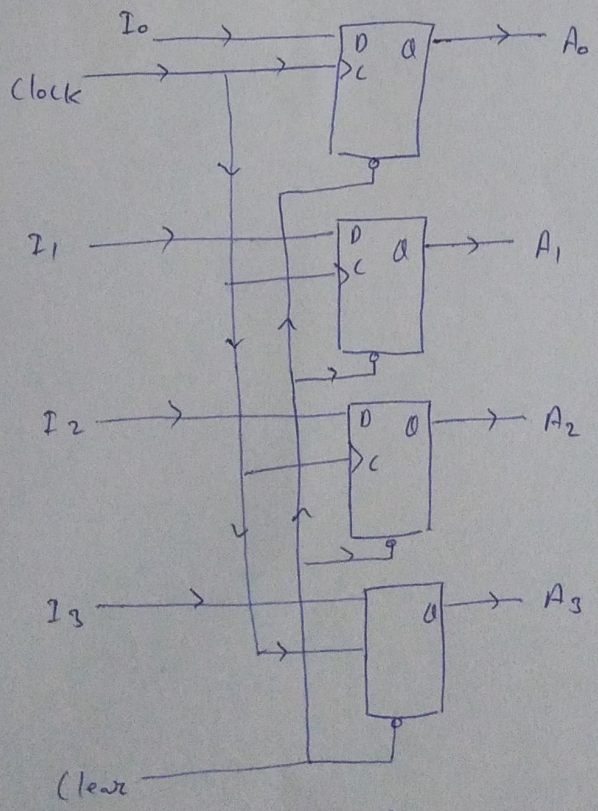


- Sumit Kumar
MCA 1st year

S_5	S_4	S_3	S_2	S_1	S_0	MUX selection
0	0	0	X	X	X	m_1
0	0	1	X	X	X	m_2
0	1	0	X	X	X	m_3
0	1	1	X	X	X	m_4
1	0	0	X	X	X	m_5
1	0	1	X	X	X	m_6
1	1	0	X	X	X	m_7
1	1	1	X	X	X	m_8

4 Bit Register

It is a group of 4 flip flops and is capable of strong 4 bit of information



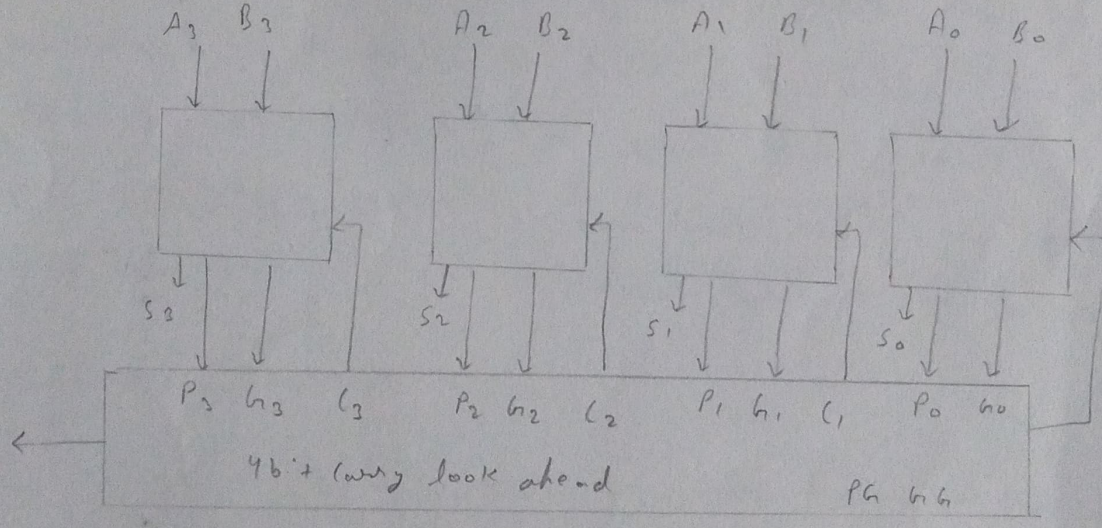
4 bit Register

Name - Sumit Kumar
Class - MCA I st year
Roll no - 58

7-12

1. Design of Carry lookahead Adders.

To reduce the computation time, there are faster ways to add two binary numbers by using carry lookahead adders. they work by creating two signals P and G known to be carry Propogator and carry Generator. the carry propogator is propogated to the next level whereas the carry generator is used to generate the output carry regardless of input carry the block diagram of a 4 bit carry lookahead adder is shown here below



The number of gate levels for the carry propogation can be found from the circuit of full adder.

$P_i = A_i \oplus B_i$

$G_i = A_i \cdot B_i$

$Sum_i = P_i \oplus C_i$

$C_{i+1} = G_i + (P_i \cdot C_i)$

$C_1 = G_0 + P_0 \cdot C_0$

$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot G_2 + P_1 \cdot P_0 \cdot C_0$

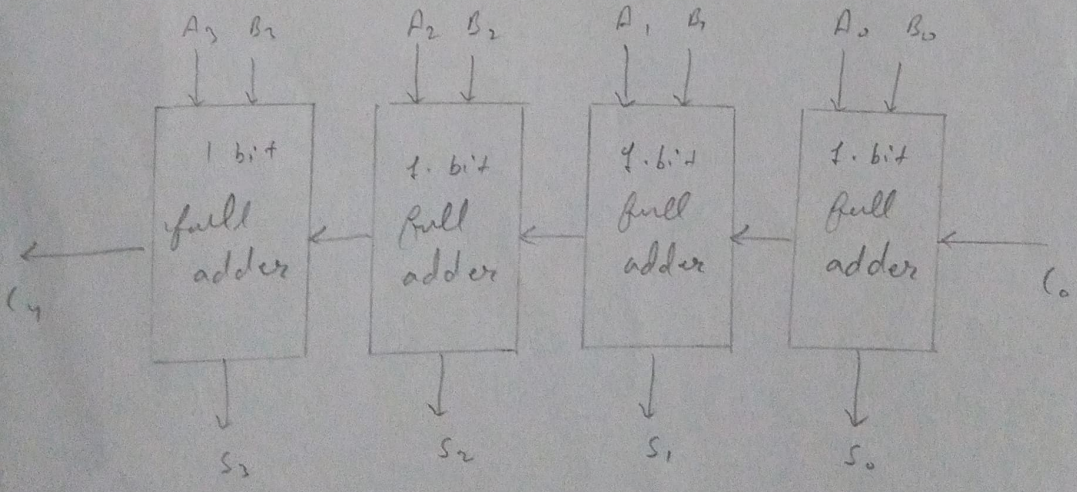
$C_3 = G_2 + P_2 \cdot C_2 = G_2 P_2 \cdot G_1 + P_2 \cdot P_1 \, G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$

$C_4 = G_3 + P_3 \cdot C_3 = G_3 P_3 \cdot G_2 P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \, G_0 + P_3 \cdot P_2 \cdot P_1 P_0 \cdot C_0$

2. Ripple carry Adder

Arithmetic operations like addition, sub, mul, divison are basic operations to be implemented in digital computers using basic gates like AND, OR, NOR, NAND etc.

Half Adder can be used to add two one bit binary numbers. It is also possible to create a logical circuit using multiple full adders to add N bit binary number. Each full adder inputs a Cin, which is the Cout of the previous adder. this kind of adder is a Ripple carry Adder. Since each carry bit "ripples" to the next full adder. the first (and only the first) full adder may be replaced by a half adder



various types of interrupt

There are two types of interrupts which are as follows

(1) Hardware interrupts

(2) Software interrupts

(1) Hardware interrupts:

the interrupt signal generated from external devices and input/output device are made interrupt to CPU when the instructions are ready.

Hardware interrupts are classified into two types.

(a) Maskable interrupt

(b) Non Maskable interrupt

(a) Maskable interrupt: the hardware interrupts that can be designed when a higher priority interrupt has occurred to the processor.

(b) Non Maskable Interrupt: the hardware that cannot be delayed and immediately be serviced by the processor.

(2) Software Interrupt

The interrupt signal generated from internal devices and software programs need to access any system call then software interrupts are present.

Software interrupt is divided into two types

(a) Normal interrupts

(b) Exception

(a) Normal interrupt: the interrupts that are caused by the software instructions are called software instructions

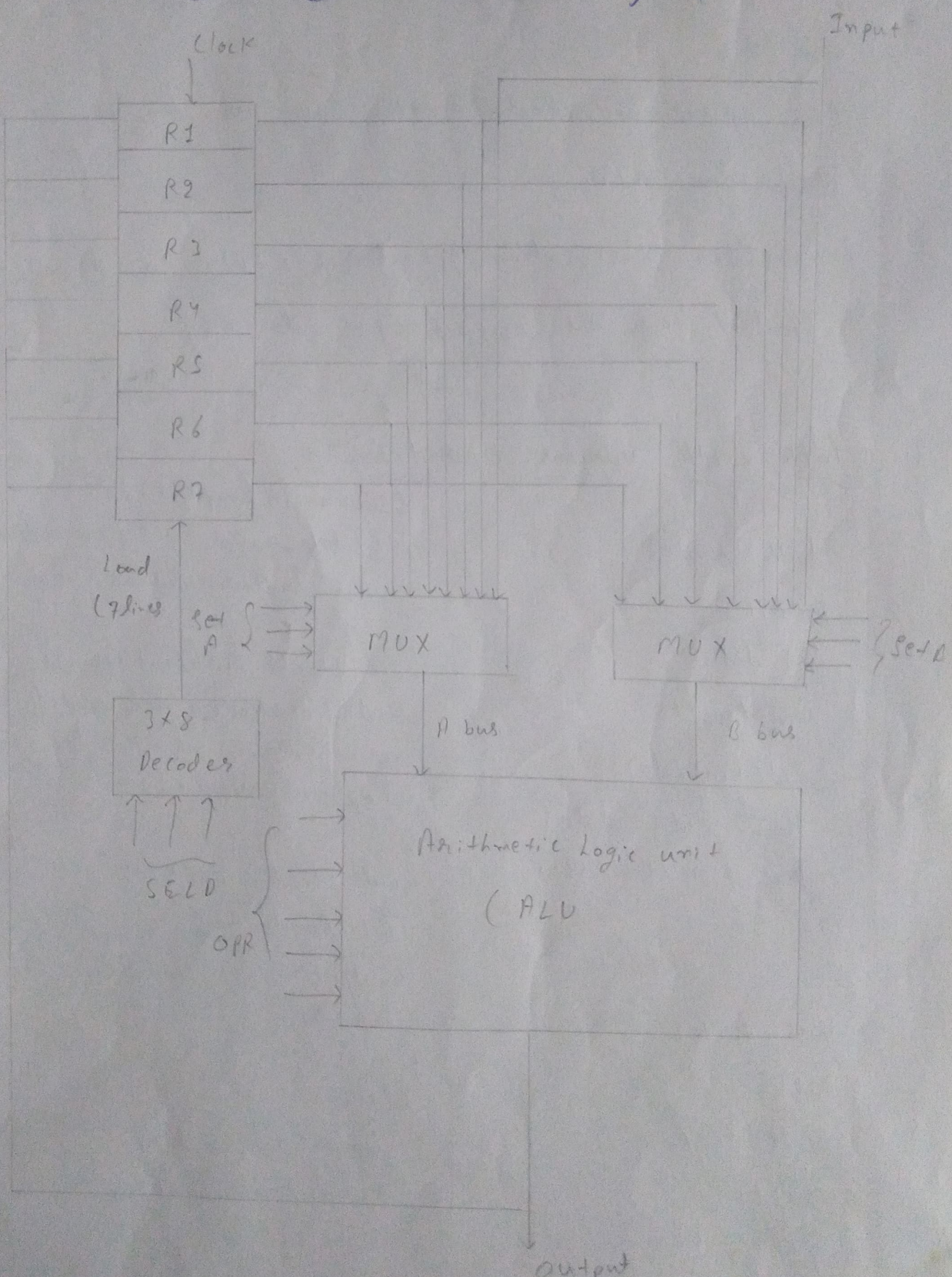
Name - Sumit Kumar

Class - MCA 1st year

58

30.11

1. Register set with common ALU



2. Perform R1 ← R2 + R3

1. MUX A selector (SELA): to place the content of R2 into
2. MUX B selector (SELB): to place the content of R3 into bus B
3. ALU operation selector (OPR): to provide the arithmetic addition A+B
4. Decoder destination selector (SELD): to transfer the content of the output bus into R1

3. Stack with various Operations

```
class Node{
public:
int data;
Node *next;
Node(int data){
this->data = data;
this->next = NULL;
}
};
class stack{
public:
Node *top;
stack():
top(NULL){}
bool isEmpty();
void push(int item);
void pop();
void display();
```

```
stack::isEmpty(){
if(top==NULL) return true;
else return false;
}
void stack::push(int item){
Node *newNode = new Node(item);
newNode->next = top;
top = newNode;
}
void stack::pop(){
if(isEmpty()){
cout<<"stack Underflow"<<endl;
return;
}
else{
Node *temp = top;
top = top->next;
delete temp;
}
}
```

```
{ stack:: size(){
int count = 0;
if (top == NULL){
cout<< "stack is empty!" <<endl;
return 0;
}
else {
Node *temp = top;
while(temp! = NULL){
count++;
temp= temp->next;
}
}
return count;
}
int main(){
stack stk;
stk.push(1);
stk.push(2);
stk.push(3);
stk.push(4);
stk.push(5);
stk.pop();
stk.pop();
cout<< "size of the stack is: " << stk.size()<< endl;
return 0;
}
```

Output stack is 3 → 2 → 1

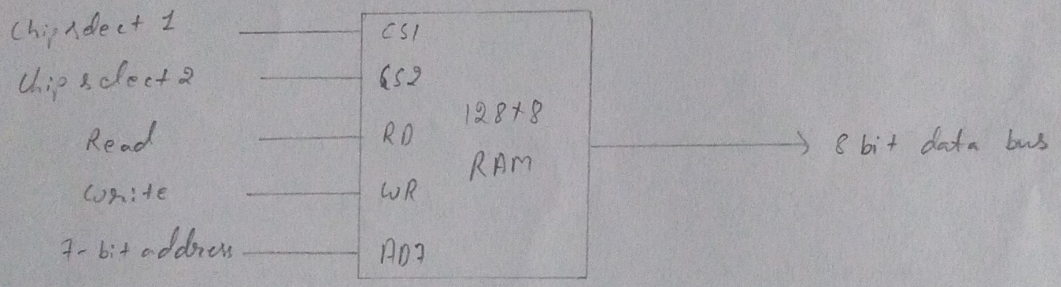
Name - Sumit Kumar
Roll No - 58
class - MCA Ist year

14.12

1. RAM and ROM Chips:

(i) RAM chips:

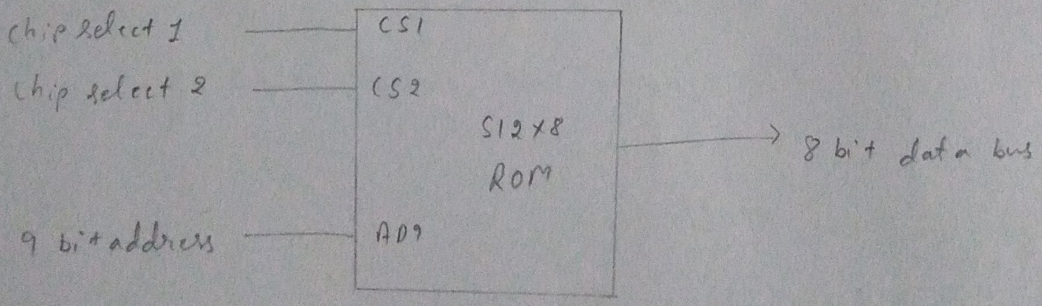
A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip only when needed. Another common feature is a bidirectional data bus that allows the transfer of data either from memory to CPU during a read operation or from CPU to memory during a write operation.



Typical RAM chip

(ii) ROM Chips:

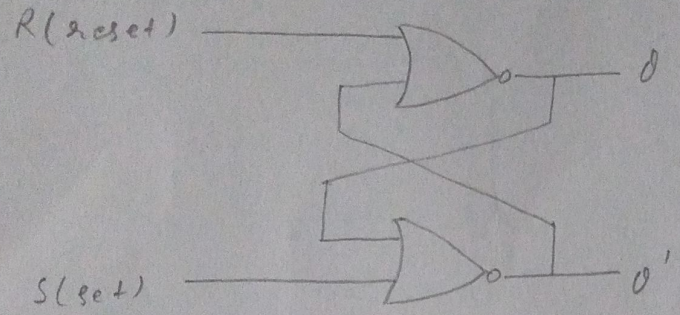
A ROM chip is organised externally in the similar manner. However, since a ROM can only read the data bus can only be in an output mode.



Typical ROM chip

2. Synthesis of flip-flops

A basic flip-flop circuit can be constructed using two cross coup NAND/NOR gates shown below. Each flip-flop has two output Q and and two inputs, set and reset. when the set input goes to 1 th Q output goes to 1 and the Q' goes to 1 when reset goes to 1



3. Registers and Counters:

(i) Register :- Register is a group of flip-flops. Its basic function is to hold info within a digital system so as to make it available to the logic units during the computing process. However, a register may also have additional capabilities associated with it.

(ii) Counter:- Counter is essentially a register that goes through a predetermined sequence of states. the gates in the Counter are connected in such a way as to produce the prescribed sequence of binary states.

4. Design of Combinational Multipliers

Combinational Multipliers do multiplication of two unsigned binary number. Each bit of the multiplied against the multiplicand the product is aligned according to the position of the bit within the multipliers and the resulting products are then summed to

the final result Main advantage of binary multiplication that the generation of intermediate products are simple if multiplier bit is as 1 the product is an oppropriately shifted copy of the multiplicand if the multiplier bit is a 0 the product is simply 0.

the design of a combinational multiplier to multiply two 4 bit binary number is illustrated below

$$
\begin{array}{cccccccc}
 & & & A_3 & A_2 & A_1 & A_0 & \\
 & & & B_3 & B_2 & B_1 & B_0 & \\
\hline
 & & & A_3 & A_2 & A_1 & A_0 & \\
 & & B_0 & B_0 & B_0 & B_0 & & \\
 & A_3 & A_2 & A_1 & A_0 & & & \\
 & B_1 & B_1 & B_1 & B_1 & & & \\
A_3 & A_2 & A_1 & A_0 & & & & \\
B_2 & B_2 & B_2 & B_2 & & & & \\
A_3 & A_2 & A_1 & A_0 & & & & \\
B_3 & B_3 & B_3 & B_3 & & & & \\
\hline
S_6 & S_5 & S_4 & S_3 & S_2 & S_1 & S_0 &
\end{array}
$$