19/6/17

COMPUTER ARCHITECTURE AND ORGANIZATION

UNIT-1 BASIC STRUCTURE OF COMPUTERS

→ **Computer:**

Computer is an electronic device which performs arithmetic and logical operations.

Types of computers:

1) **Personal computer:**

→ They are used in houses, schools, offices etc.

2) **Notebook computer:**

→ It is the compact version of PC.

3) **Work stations:**

→ They are used for graphics (high computation power than PC)

4) **Enterprise systems:**

→ They are used for processing business data (high computation power than work

stations)

5) Seavers:

→ They contain data base that can be accessed by other systems.

6) Super computers:

→ They are used for large scale numerical calculations. They are used for weather forecasting as well as in aircraft design and simulation.

Computer Organisation:

Using Design of input unit, output unit, memory unit and processor

→ Memory unit is used to store information.

→ Processor performs operation.

→ Computer architecture deals with the instruction set and hardware unit which is used to implement the instruction.

Functional Units:

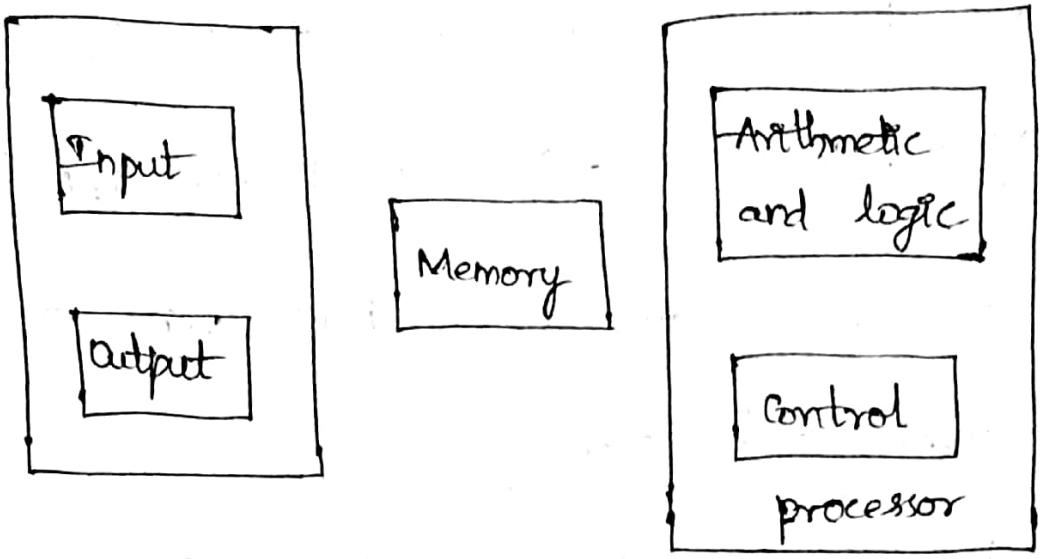


Fig. Basic functional units of a computer

→ Computer accepts information through input unit

→ Results are sent through output unit

→ Memory is used to store program and data.

Types of storages:

1) Primary storage (RAM) → volatile memory
2) Secondary storage.

Primary storage:

RAM has 3 levels depending on size and speed.

(When a file is saved, it is stored in hard disk and when it is opened again, it is stored in primary (RAM))

→ The fast and small RAM unit is called cache memory.

→ The large and slow RAM unit is called main memory.

To access memory, allocation of address is required.

→ Amount of information stored in each location is called word (Indicates amount of data)

→ We can perform operations in terms of words or multiple words or part of word.

Length of word → 16 bits to 64 bits

→ Primary storage is fast and expensive compared to secondary storage.

→ Secondary Storage

Ex: magnetic disk and optical disk.

→ Primary storage can be directly accessed by CPU [Main memory]

→ Secondary storage cannot be directly accessed by CPU [Hard disk]

Arithmetic and Logic Unit:

To perform operation between two operands, they should be brought to processor.

-ALU includes arithmetic operations → +, −, ×, /

Logical operations → AND, OR, NAND, NOR

Control Unit:

ALU + Control Unit → Processor.

It generates control signals that are required to in order to perform some operations.

Ex: Memory Read

Memory write

I/O Read

I/O Write

If input and output units are in single unit, it is called I/O unit

→ For graphic display, input and output units are in same unit called as I/O unit.

21/6/17

BASIC OPERATIONAL CONCEPTS

→ Add LOCA, R0

LOCA is an operand address of an present in memory

R0 is a register R0

Above instruction indicates addition of two operands where one is in memory and the other is in register

and stores the result in R0

→ Load LOCA, R1

↓

address of some memory location

The above instruction indicates loading of operand present in LOCA into register R1

→ Add R1, R0

Adds operands in R1 and R0 and stores the result in R0 register

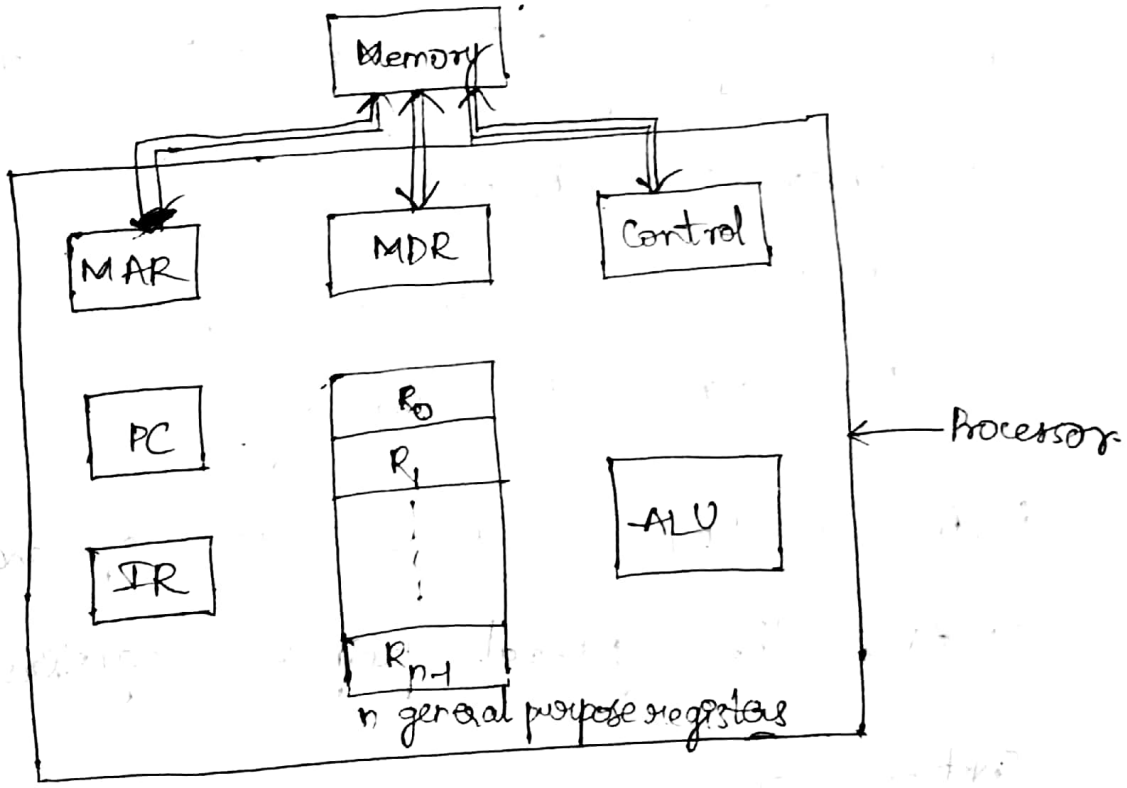


fig. connections between the processor and the memory.

MAR → Memory address Register

It contains adress of memory location which is to be accessed

MDR → Memory Data Register

It contains the data that is to be written or read by the memory

PC → Program counter

It holds the address of next instruction that is to be executed

IR → Instruction Register

It holds the instruction that is to be executed currently.

$R_0, R_1 \ldots R_{n-1}$ holds the operands called the general purpose registers.

Control unit:

It is used to generate control signals that are required to perform

some action.

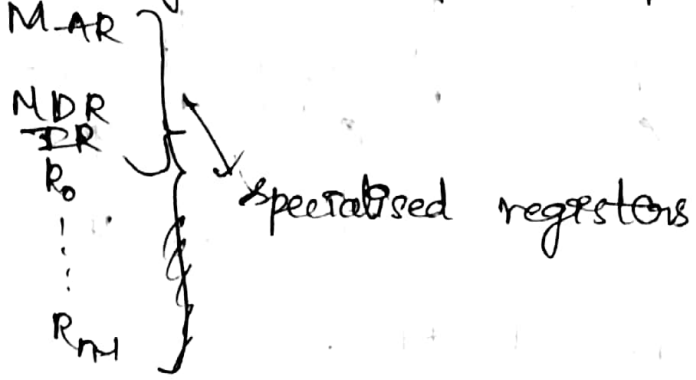
ALU - to poform Arithmetic and logical operation.

MAR → unidirectional i.e. address is transfered from MAR to memory

MDR → bidirectional

Control unit → u

All together is called processor



Interrupt service routine:

It is the program run by the processor when there is an interrupt from the control unit or device.

It is used to control temperature of boiler.

When there is an interrupt, the processor stops the execution of

normal program (normal execution) and saves the information in general registers and control unit and executes interrupt service routine.

Bus structures:

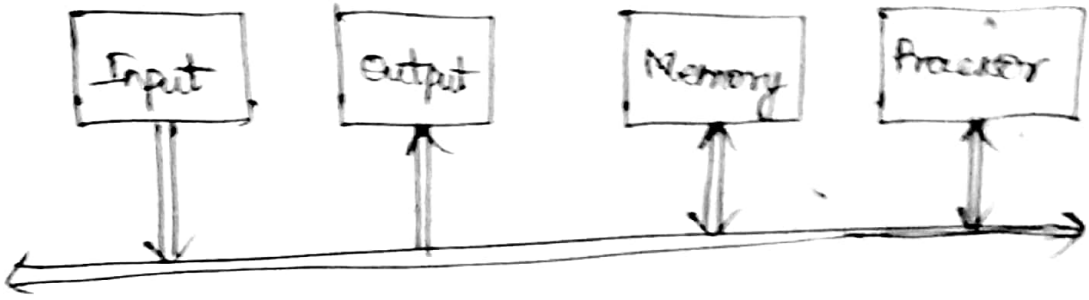


Fig. Single bus structure.

Bus is required to transfer data, address and control signals from one device to another device.

Bus contains set of lines or wires.

- Address lines
- data "
- control "

} system bus

Address lines are used to transfer address
data " " " data
control " " " control

signals.

Input → Unidirectional i.e. given to memory

Output → " i.e. taken from "

Memory → Bidirectional ie. read and written into memory

Processor → " i.e. data is loaded and taken out of memory.

Multiple buses increases speed of operation and cost.

Multiple buses → multiple transfer (more information)

→ Processor is fast in operation

Keyboard and printer are slow in operation.

Buffer:

It is an I/O device used to compensate the speed of operation b/w processor and printer and keyboard.

→ Buffer in printer receives the data transferred by processor character by character and and prints the data thereby ~~cons~~ compensating the diff. in speed of operation b/w printer and processor

22/6/17

Software:

→ Set of instructions is software.

→ Compiler is a software which converts high level language to machine language.

→ Text editor is also a software which is used for entering and editing the application program.

~~It is also used~~

→ Operating system is a type of software

which is used in transfering the inform^{n} between memory and disk.

It is also used in allocating space in memory and disk for programs and data and handling I/O operations.

Performance:

→ Performance is a quantity used to measure the speed of execution of a program.

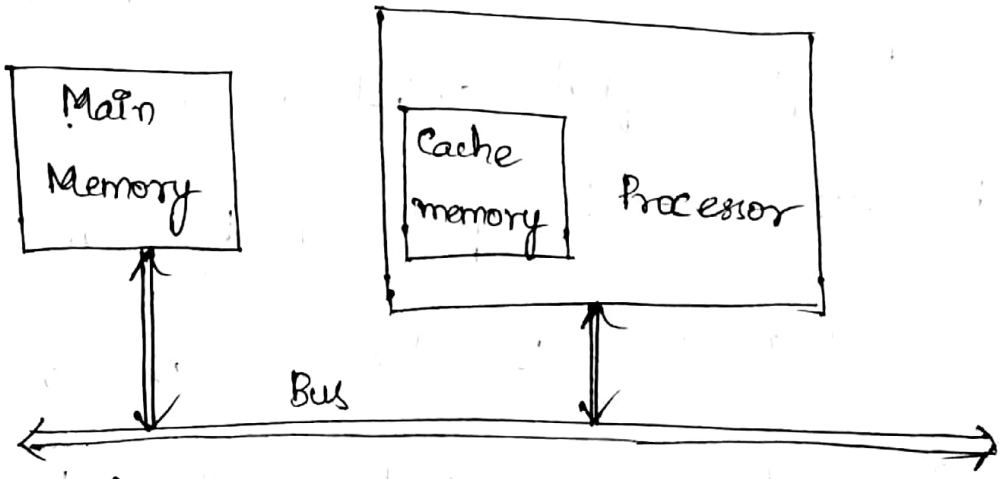


fig. Processor ~~cab~~ cache

→ Information is fetched from memory into processor.

→ When the information is fetched from the main memory by processor, (a copy) it is saved or placed in cache memory so that it can executed execute quickly when repeated

(Time taken for fetching the information from cache memory < Main memory)

Processor clock:

The processor circuits are controlled by a timing signal called as clock.

The action to be performed by executing some instructions is divided into no. of basic steps such that each basic step is completed within one clock cycle.

→ Clock frequency is of the order

500 MHz - 1.25 GHz.

Basic Performance Equation:

If N - no. of machine language instructions that are to be executed

S - Average no. of basic steps needed to execute an instruction.

R - clock rate (clock frequency)

Then,

T = program execution time = $\frac{N \times S}{R}$

To reduce T (fast), N↓, S↓ and R↑

Pipelining and superscalar operation:

Pipelining:

It is the process of fetching information from memory while the present instruction is being executed.

Advantage: Speed

Super scalar processor:

The processor which contains multiple functional units is called as super scalar processor.

Eg: Add R1, R2, R3

Operands in R1 and R2 are added and stored in R3 (overwritten) and next instruction is fetched (pipelining) (i.e. R3 is added and stored in R3)

Inorder to increase clock rate (faster), IC technology is used (which is faster i.e. ECL etc)

clock period is also decreased (clock rate ↑) when the processing done by basic step is reduced.

Instruction set:

CISC and RISC

CISC- Complex Instruction Set Computer

RISC - Reduced Instruction Set Computer.

CISC uses complex instructions

RISC uses simple instructions.

(which increases no. of instructions, N↑)

RISC : N ↑ S ↓

CISC : N ↓ S ↑

When CISC is combined with pipelining it gives best performance.

RISC is combined with pipelining for easier implementation.

Compiler can compile ~~into~~ into a fewer instructions and rearrange to improve the performance.

23/6/17

Performance Measurement:

SPEC → system Performance evaluation corporation

SPEC rating = Running time on the reference computer / Running time on the computer under test

It is the comparison of reference computer and computer under test.
(speed of execution)

Overall SPEC rating = $\left(\prod_{i=1}^{n} SPEC_i\right)^{1/n}$

n → no. of programs executed.
to test the performance

Multiprocessors and Multicomputer

→ If the system contains multiple computers, then it is called

multicomputer.

In multiprocessor system, either they run seperate task parallely or a large task is subdivided into subtasks.

Multicomputer is interconnection of computers.

Data Representation:

1) Fixed point representation;

Integer Representation

-14 → 1 000 111 0 (Sign-magnitude)

-14 → 1111 0001 (1's complement)

-14 → 1111 0010 (2's ")

Addition addition

Arithmetic addition.

+6 → 0000 0110

+13 → 0000 1101

+19 0001 0011 =

$-6 \longrightarrow$ ~~00110~~

+13

$-6 \rightarrow$ 0000 0110

1's complement of 6 $\longrightarrow$ 1111 1001

2's " " $\longrightarrow$ 1111 1010

13 $\longrightarrow$ 0000 1101

1111

0000 0111

Ignore

$= 7$

$+6 \longrightarrow$ 0000 0110

$-13 \longrightarrow$ ~~+10~~

$13 \rightarrow$ 0000 1101

1's complement $\rightarrow$ 1111 0010

2's " $\rightarrow$ 1111 0011

$6 \rightarrow$ 0000 0110

1111 1001

$= \pm(0000\ 0110 + 1)$

$= -7$

1st comp → carry generated
⇒ Add to LSB

MSB = 0 ⇒ +ve

MSB = 1 ⇒ -ve

Result = −(1's comp)

2's comp → carry generated
→ ignore

MSB = 0 ⇒ +ve

MSB = 1 ⇒ -ve

result = −(2's comp)

$= -7$

$-6 \rightarrow$ 1111 1010

$-13 \rightarrow$ 1111 0011

1111 1101

$= -(0000\ 0010 + 1) = -19$

Arithmetic subtraction:

$(\pm A) - (+B) = (\pm A) + (-B)$

$(\pm A) - (-B) = (\pm A) + (+B)$

Above eqns represents subtraction in terms of addition.

$-6 - (-13) = -6 + 13$

$13 \rightarrow$ 0000 1101 (or $-6 - (-13)$

$-6 \rightarrow$ 1111 1010 $= (1111\ 1010)$

0000 0111 $-(1111\ 0011)$

$= 00000111$

Overflow:-

It occurs when two n-digits no's are added and the result has n+1 digits.

Eg: When 8-bit register is used, the range of the no's that can be stored is from -128 to 127

In general, for n-bit register, the range of the no's that can be stored is

-2^{n-1} to $+(2^{n-1}-1)$.

$$
\begin{array}{rl}
+70 - & \overset{0}{0}\ \overset{1}{1}\ 0\ 0\ 0\ 1\ 1\ 0 \\
+80 - & 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0 \\
\hline
+150 & 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0
\end{array}
$$

150 cannot be represented using 8-bits. Over flow occurs carry into the sign bit and carry out of the sign bit are not equal

-70	1 1 0	10111010
-80		10110000
-150		01101010

9's comp → carry generated
⇒ +ve
Obt → -ve
result = -(9's comp)

10's comp → carry → +ve
no carry → -ve
result = -(10's comp)
>9 ⇒ +6 (0110)

Decimal Fixed-point Representation:

Representation of decimal no's in BCD is called decimal fixed-point representation.

(+375) + (-240)

0375 → 0000 0011 0111 0101

9760 → 1001 0111 0110 0000

+

1001 1010 1101 0101

1 0110 0110

9 10 13 5

1010 0001 0011 0101

0110

110 144

XII 0000 0001 0011 0101

0 1 3 5

Floating-Point Representation:-

It contains 2 parts

1) Mantissa
2) Exponent

Consider a decimal number

+6132.789

Mantissa exponent | Fraction | Exponent

+0.6132789 | +04

Fractional part is mantissa

$(0.6132789 \times 10^4 = 6132.789)$

$m \times r^e$

m - mantissa

r - radix or base

e - exponent.

Consider a binary number

+1001.11

Representing it in 8-bit fraction and 6-bit exponent

Fraction	Exponent
01001110	000100

↓

+ve no.

For decimal, point is represented

For binary, " " not " .

For binary, $m \times 2^e = (0.1001110)_2 \times 2^4$

$r = 2$ in case of binary

29/6/17

Floating point number is said to be normalised when the most significant digit of the mantissa is non-zero.

There are 2 standard forms of floating point no. given by

1) ANSI (American National Standard Institute)

2) IEEE (Institute of Electrical and Electronic Engineers)

ANSI - 32 bit floating point no's in byte format is as given below.

Byte 1	Byte 2	Byte 3	Byte 4
SEEEEEEE	MMMMMMMM	MMMMMMMM	MMMMMMMM

S → Sign of mantissa

E = exponent bits in 2's complement

M = Mantissa bits.

Eg: 13

1) Represent 13 in 32 bit floating point ANSI form.

~~32 = 1101~~

① $13 = 1101 = 0.1101 \times 2^4$

Mantissa, Exponent

= 00000100 11010000 00000000 00000000

~~sign~~ exponent (4)

② $-17 = -10001 = -0.10001 \times 2^5$

~~ANSI format~~

= 10000101 10001000 00000000 00000000

③ -0.125 in ANSI format

-2 → 00000010

11111101

+1

11111110

$-0.125 = -0.125 \times = -0.001$

$= -0.1 \times 2^{-2}$

= 11111110 10000000 00000000 00000000

In −0.001, MSD of mantissa is not 1.
So, it is normalised to -0.1×2^{-2}. (represented in normalised form).

Error detection Codes:

When binary information is transmitted through channel, noise is added and 0 changes to 1 and 1 changes to 0.

Error detection indicates checking whether there is error or not

Error correction is correcting the error in transmission.

Even parity contains even no. of 1's

Odd " " odd " "

Error detection codes

Parity Bit generation

Message xyz	P(odd)	P(even)
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1

Source

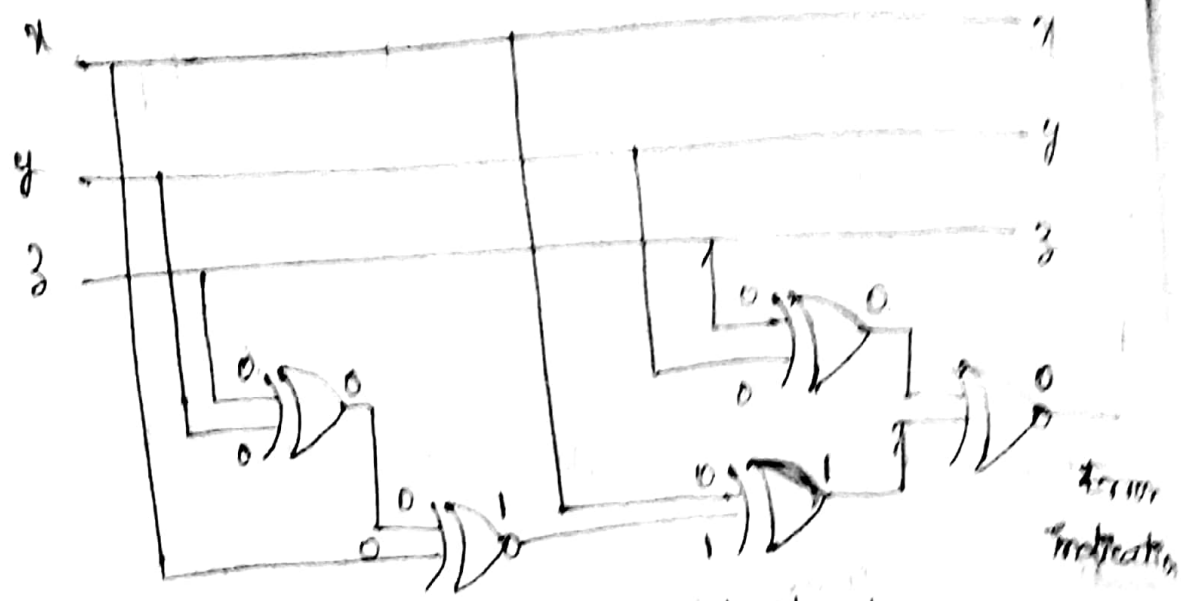


Fig. Error detection with odd parity bit

Output = 0 indicates no error and i/p contains odd no. of 1's

Output = 1 indicates error and i/p contains even no. of 1's

30/6/17

Number systems:

Convert the following binary number to decimal

1) $(101110)_2$

Sol: $2 + 4 + 8 + 32 = 46$

② Convert the following no's with the indicated bases to decimals.

a) $(2121)_3$

$= 1+6+9+54+81$

$= (151)_{10}$

b) $(4310)_5$

$= 5+75+500$

$= (580)_{10}$

c)

③ Convert the following decimal no. to binary

$(673)_{10}$

Sol:

2	673
2	336–1
2	168–0
2	84–0
2	42–0
2	21–0
2	10–1
2	5–0
2	2–1
	1–0

$= (1010100001)_2$

(or)

16	673
16	42–1
	2–10

$= (2A1)_{16}$

$= (0010\ 1010\ 0001)_2$

④ Convert the following decimal no's to the bases indicated.

a) $(7562)_{10} = (\quad)_8$

Sol:

$(7562)_{10} = (111\ 101\ 110\ 010)_8$

8	7562
8	945 – 2
8	118 – 1
8	14 – 6
	1 – 6

$= (16812)_8$

b) $(1938)_{10} = (\quad)_{16}$

16	1938
16	121 – 2
	7 – 9

$= (792)_{16}$

⑤ convert the hexadecimal number $(F3A7C2)_{16}$ to binary and octal.

Sol: $(F3A7C2)_{16} = (1111\ 0011\ 1010\ 0111\ 1100\ 0010)_2$

$= (74723702)_8$

(6) Show the value of all bits of a 12-bit register that holds the no. equivalent to decimal (215) in

a) binary

b) binary coded octal

c) binary coded hexdecimal.

d) binary coded decimal.

Sol: a) $(215)_{10}$

2	215
2	107 — 1
2	53 — 1
2	26 — 1
2	13 — 0
2	6 — 1
2	3 — 0
	1 — 1

$= (11010111)_2 = (0000\ 1101\ 0111)_2$

b) $(215)_{10} = (\quad)_8$

$= (0000\ 1101\ 0111)_2 = (0327)_8$

~~$= (010\ 001\ 111)_8$~~

~~$= (000\ 010\ 001\ 111)_8$~~

c) $(215)_{10} = (\quad)_{16}$

~~$= (0010\ 0001\ 0111)$~~

$= (0000\ 1101\ 0111)$

$= (0D7)_{16}$

d) $(215)_{10} = (0010\ 0001\ 0111)$.

(7) Obtain the 9's complement of the following 8-digit decimal number

a) 1234986

9's compliment of

Sol: $(1234986) = (87650123)$

(8) Obtain the 10's complement of the following 6-digit decimal number

123900

Sol: 9's complement of (123900) = (876099)
+1

10's complement = (876100)

3/8/17

Microoperations:

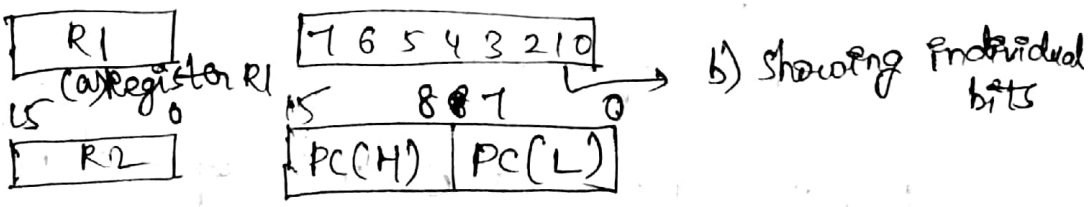
→ They are defined as the operations performed on data stored in registers.

They are: Arithmetic, logical and data transfer operations.

Register transfer language:

The symbolic notation used to describe the microoperations is called register transfer language.

Register transfer:



c) Numbering of bits

d) Divided into two parts

fig. Block diagram of register

Register is a set of flip-flops

R2 → 16 bit register (0-15)

PC → Program counter

d) → represents that 16 bit register is divided into two parts i.e. higher order bit (15-8) and lower order bit (0-7)

Eg: R2 ← R1

Symbolic notation of transferring content of R1 to R2

- Condition

Ex: If (P=1) then (R2 ← R1)

P is called as control signal.

Above eqn can also be written as

P: R2 ← R1

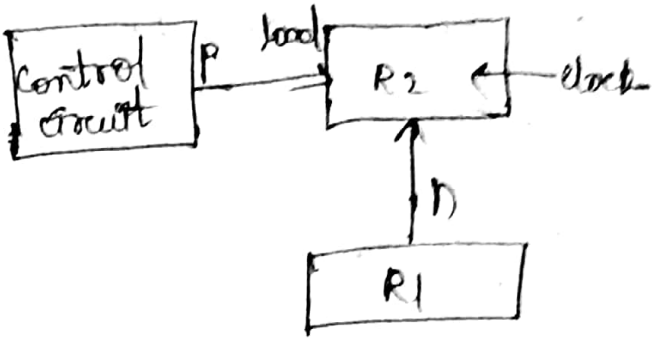
micro operation

P: R2 ← R1 → control function.

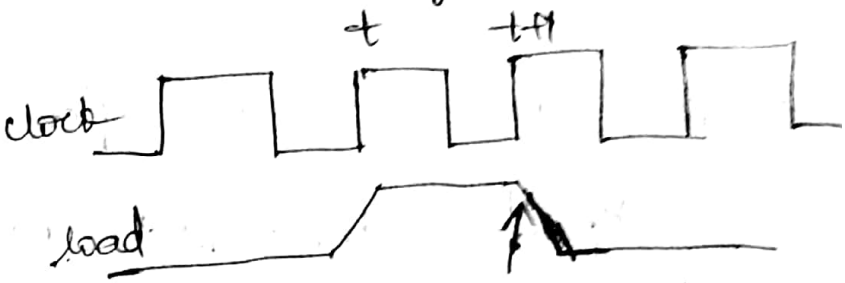
The above control function can also represented using block diagram.

Here n – no. of bits transferred from R1 to R2

→ the control circuit enables load and transfer of information



a) Block diagram



transfer occurs here

b) Timing diagram

takes place at the rise half of the clock signal

If P=1 load signal is enabled at 't' and transfer takes place immediately after t but not at 't')

Ex: T: R2 ← R1, R1 ← R2

This control function consists of 2 microoperations

Bus and Memory transfers

Bus is required to transfer data from one register to other register.

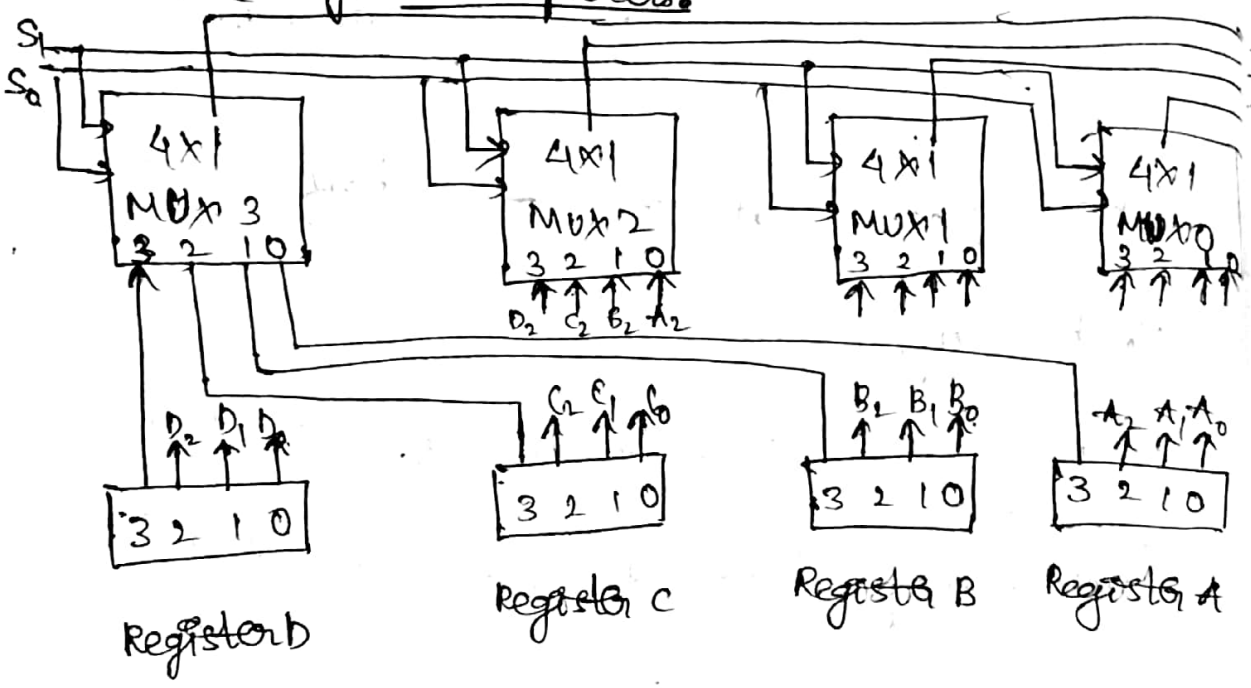
(i.e. o/p of one register is connected to i/p of another)

Output of source register " "

i/p of destination " using common bus.

→ These buses can be implemented using multiplexers, three state buffers.

Bus using Multiplexers:



4 line common bus → transfers 4 bits

4 bit registers → 4 bits (i.e. 4 flip flops)

~~o/p of regr common bus~~

o/p of multiplexer is connected to

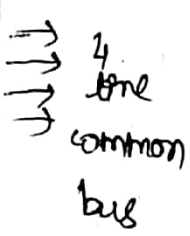
common bus lines.

If $S_1 = 0$, $S_0 = 0$, Register A is selected

$S_1 = 0$, $S_0 = 1$ " B "

$S_1 = 1$, $S_0 = 0$ " C "

$S_1 = 1$, $S_0 = 1$ " D "



5/4/17

Bus using three state buffers:

To transfer information from register C to register R1, information is transferred from C to bus and it is from bus to register R1

BUS ← C, R1 ← BUS (or)

⇒ R1 ← C

Three state Bus Buffers:

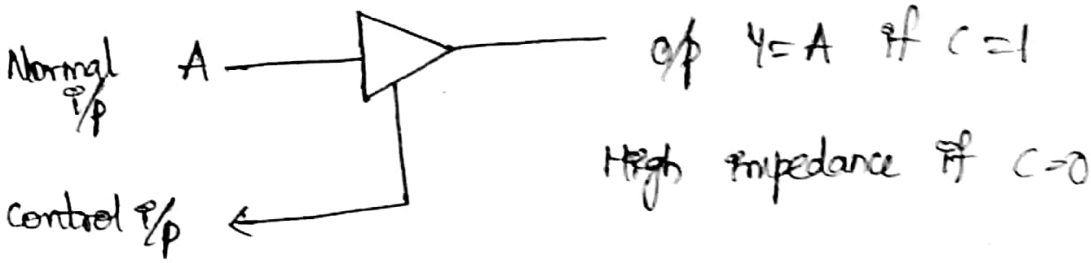


Fig. Graphic symbol for three state buffer

(i) When control i/p is 1, i/p will be transferred to o/p a) i/p=0, b) i/p=1

(ii) When control i/p is 0, it is in high impedance state and o/p is disconnected

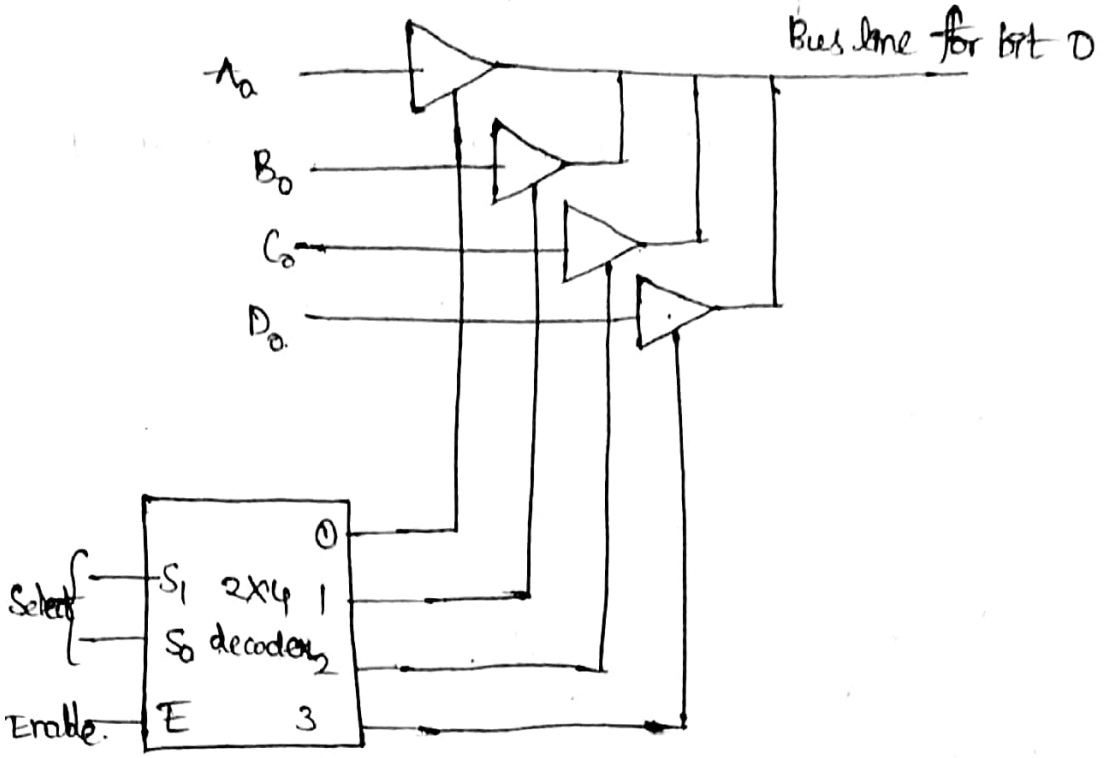
∴ It has 3 states.

(i) C=1, i/p=0, o/p=0

(ii) C=1, i/p=1, o/p=1

(iii) C=0, i/p=X, o/p=0 or disconnected

Only one register can use the bus at a given point of time.



O/p's of decoder acts as control i/p's to buffer

When EN=0, all the o/p's of decoder are disabled

When EN=1, $S_1=S_0=0$, buffer corresponding to A_0 will be activated and others are disabled.

Similarly, there are buffers for A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2 and A_3, B_3, C_3, D_3 for 4-bit register.

When EN=1, $S_0=0$, $S_1=1$, buffers corresponding to B_0 will be activated

$S_0=1$, $S_1=0$, " " C_0 will be activated

$S_0=1$, $S_1=1$, buffer corresponding

b1 b2 will be activated

→ For n bit register, n buffers are required

→ For four no. of registers, we 2 selection lines are required. i.e 2×4 decoder

→ For 2^n registers, $n \times 2^n$ decoder are required

→ No. of bus lines depends on the no. of bits.

→ No. of selection lines i.e decoder depends on the no. of registers

Memory Transfer

→ Read : DR ← M[AR] —①

Address Register contains the address of the memory location to be accessed. Data Register contains the data that is to be read from the memory or to be written in to the memory

Scanned by CamScanner

(1) indicates that the data is read from the memory into data register.

M[AR] → Memory word selected by the address in AR ~~which~~ is transmitted to DR

(1) indicates memory read Operation.

→ M[AR] ← R_1 : writing the data from the register in to the memory is Memory write Operation.

It indicates the data transfer from register R_1 to memory which is write operation

6/7/17

Arithmetic Microoperations

Microoperations: operations performed on data stored in registers.

Basic Arithmetic Operations

→ * $R3 \leftarrow R_1 + R_2$

content of R_1 and R_2 is transfered to R_3.

→ * $R3 \leftarrow R_1 - R_2$

content of $R_1 - R_2$ is transfered to R_3.

* $R_2 \leftarrow \overline{R}_2$ (1's complement)

$\overline{R}_2$ indicates complimentary content of R_2

* $R_3 \leftarrow \overline{R_2} + 1$

$\overline{R_2} + 1 \rightarrow$ 2's complement of content of R_2 is transfered to R_2.

2's complement = negation operation $(-R_2)$

Ex: $R_1 + \overline{R_2} + 1$ indicates $R_1 - R_2$

* $R_3 \leftarrow R_1 + \overline{R_2} + 1$

$R_1 + \underbrace{\overline{R_2} + 1}$ $\quad = R_1 - R_2$

2's complement of R_2

* $R_1 \leftarrow R_1 + 1$

Incrementing the content of R_1 by 1

* $R_1 \leftarrow R_1 - 1$

decrement the content of R_1 by 1.

Binary Adder:

FA - Full Adder

C_0 - input carry

$C_4 \rightarrow$ o/p carry

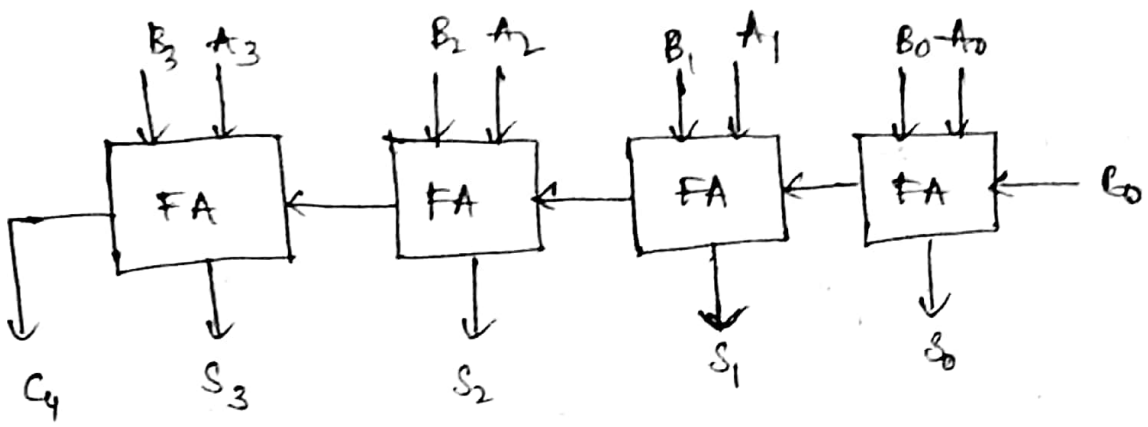


Fig. 4-bit binary adder.

It is used to add two 4-bit numbers

(Full Adder - 3 bits along with carry

Half adder - 2 bits only)

Binary Adder - Subtractor:

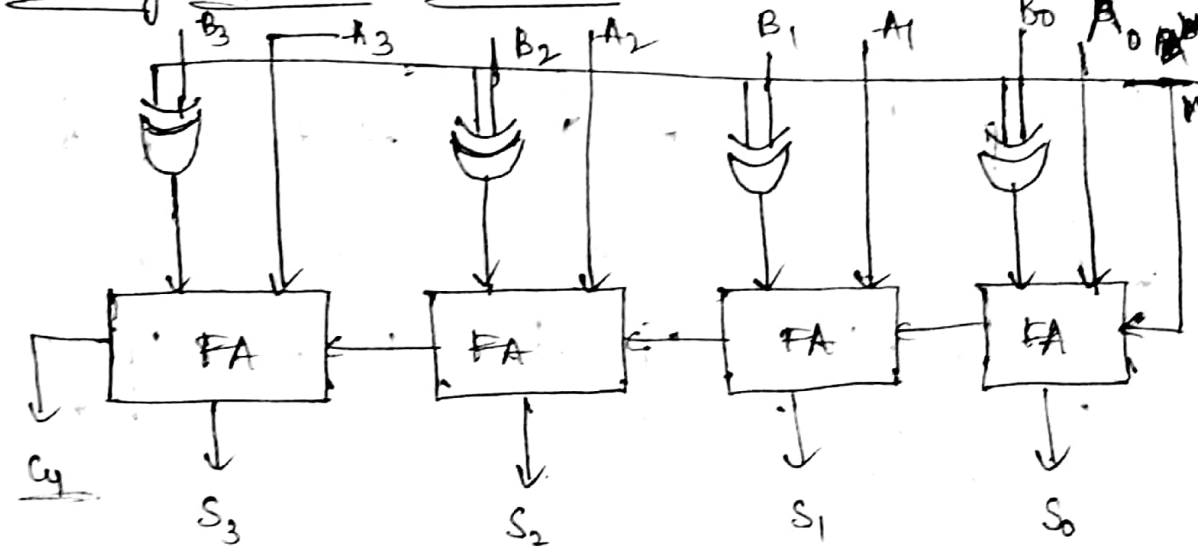


Fig. 4 bit adder-subtractor.

(*) If M = 0, Input carry $C_0 = 0$ and the

o/p of first full adder = $A_0 + B_0$

Similarly o/p from all full adder = $A + B$

∴ It acts like adder

(ii) When $M = 1$, $C_0 = 1$

the output is $A + \bar{B} + 1 = A - B$

(i) $A - B$ for unsigned no's when $A \geq B$

(ii) and 2's complement of $B - A$ if $A < B$

(∵ -ve no = 2's complement of +ve)

b) for signed numbers it is $A - B$ when there is no overflow

Binary Incrementer:

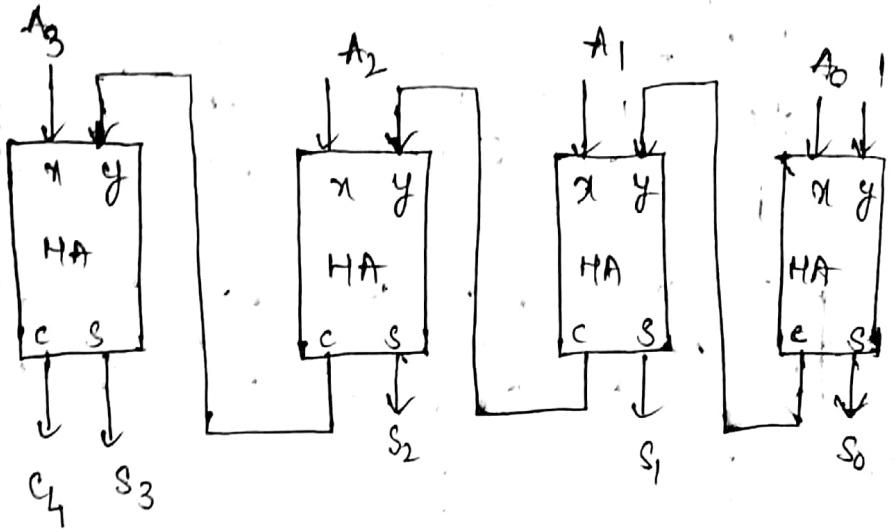


fig. 4-bit binary incrementer

$O/P = A + 1$

Half adder is used

As only 2 bits are added (∵ A_0 & 1...),

Half adder is used.

7/7/19

① Perform the subtraction with the following unsigned decimal no's by taking the 10's complement of the subtrahend.

a) 5250 − 1321

9's

1321 → 8678

10's " → 8679

$$\begin{array}{r} {}^{1}5^{1}2^{1}5^{1}0 \\ +\ 8679 \\ \hline 13929 \end{array}$$

= 3929

b) 1753 − 8640

8640

9's complement → 1359

+1

10's complement → 1360

$$\begin{array}{r} {}^{1}1^{1}1^{1}7^{1}5^{1}3 \\ 1360 \\ \hline 3113 \end{array}$$

10's complement of 3113 = 6887

c) 20-100

020 → 80

100 → 9's
899
+1

10's → 900
+ 20
920

= -(0:

= -(079
+1)

= -80

d) 1200-250

250 → 9's
749
+1

10's → 750
1200
950

= -(80

= 950

② Perform the subtraction of the following unsigned no's by taking the 1's complement of the subtrahend.

a) 11010 − 10000

Sol: 11010

1's complement of 10000 =

"

11010
10000
01010

= 01010

b) 100 − 110000

1's complement of 110000 =

2's " " " =

01000
100
01100

= −(10100)

(c) ~~42 and 8~~

(3) Perform the arithmetic operations in binary using 2's complement representation using 7-bit

a) (+42) + (-13)

Sol:

42 → 0101010

13 → 0001101

1's complement of 13 → 1110010

2's " " " → 1110011

0101010

0011101

= 0011101

(4) ~~Rep~~ Perform the following arithmetic operations using signed 10's complement representation for -ve no's

a) (-638) + (+785)

Sol: (-638)+185

8.

9's complement of 638 = 361

10's " " = 362

185

147

= 147

b) (-638)-(+185)

9's complement of 0638 = 9361

10's " " " = 9362

~~9's " " 185 = 814~~

~~10's " "~~

9's " " 185 = 9814

10's " " 185 = 9815

+9362

9177

9's = -0822

10's = -823

5) Represent the no. $(46.5)_{10}$ in floating point binary no. with 24 bits.

The normalized fraction mantissa has 16 bits and the exponent has 8 bits.

Sol: $(46.5)_{10}$

$(46)_{10} = 101110$

$(0.5)_{10} = 0.5 \times 2 = 0.1$

$46.5 = 101110.1$

$= 0.1011101 \times 2^{6}$

0101110100000000 00000110

16 bit mantissa

8 bit exponent (+6)

Sign of mantissa

7/7/17

Arithmetic circuit:

→ A single circuit that performs several arithmetic operations on registers.

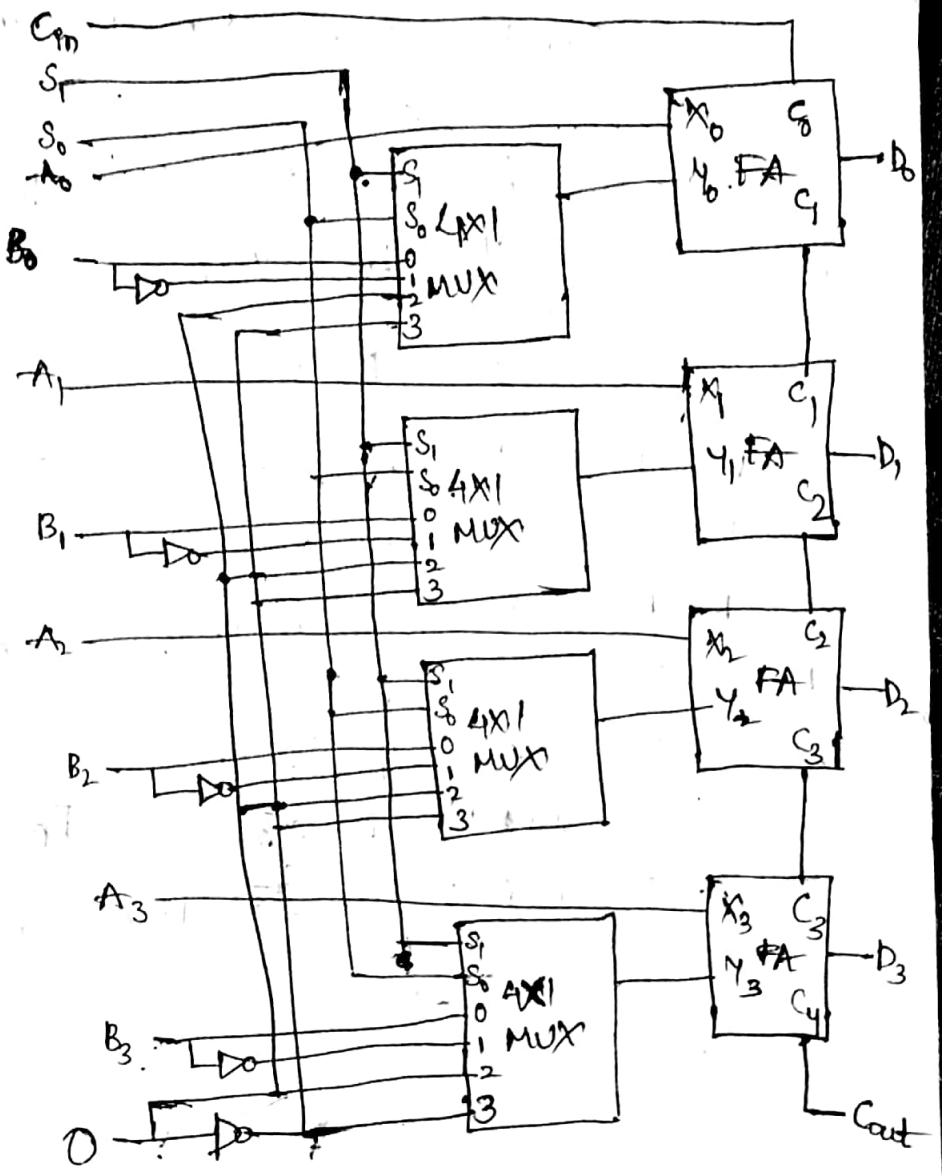


Fig. 4-bit arithmetic circuit

Arithmetic Circuit Function Table

Select		I/p	O/p	Microoperation
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with c
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract w/ borro
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer
1	0	1	0	$D = A + 1$	Increment A
1	1	0	All 1's	$D = A + 1 + 0$	Decrement A
1	1	1	All 1's	$D = A + 1 + 1 = A$	Transfer A

0001
1110
1
1111

Logic Microoperations:

→ $P: R_1 \leftarrow R_1 \oplus R_2$

P → control signal

$R_1 \oplus R_2$ are transfered to R_1 if P=1

$Q: R1 \leftarrow R_2 + R_3 \;,\; R_4 \leftarrow R5 \vee R_6$

$+$: Addition

$\vee$: logical OR

'+' on control function indicates 'OR'

'+' in microoperation in addition.

List of logic Microoperations:

Truth tables for 16 functions of two variables:

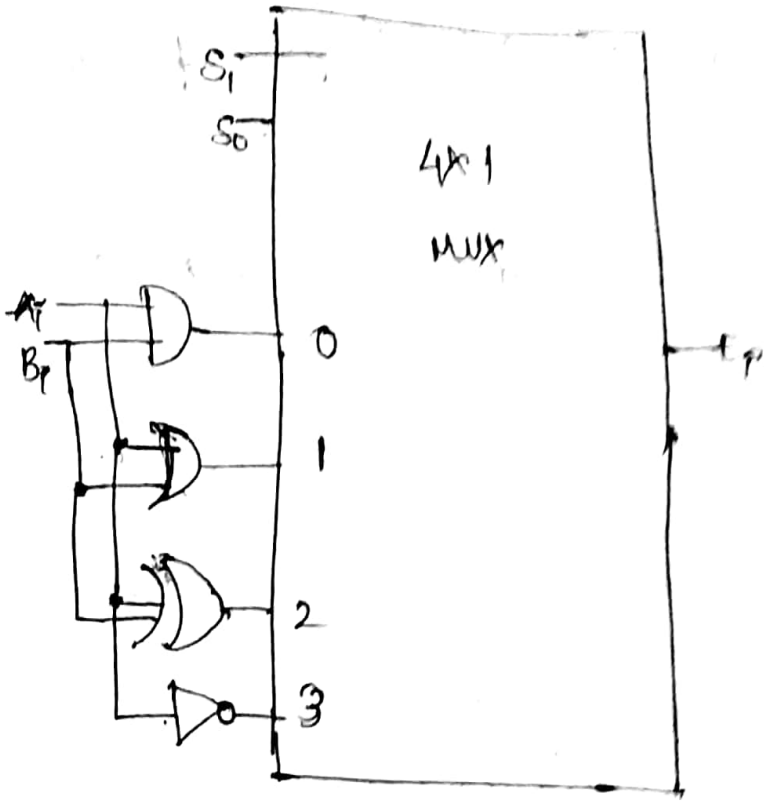
x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

~~Truth Table~~

Boolean Logic & Microoperations

Boolean function	Microoperation	Name
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \bar{B}$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F_3 \leftarrow \bar{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	Transfer B
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	Exclusive-OR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x+y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	Exclusive NOR
$F_{10} = y'$	$F \leftarrow \bar{B}$	Complement B
$F_{11} = x + y'$	$F \leftarrow A \vee \bar{B}$	
$F_{12} = x'$	$F \leftarrow \bar{A}$	Complement A
$F_{13} = x' + y$	$F \leftarrow \bar{A} \vee B$	
$F_{14} = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow$ all 1's	set to all 1's

Logic Microoperations



a) logic diagram

fig. one stage of logic diagram.

S_1	S_0	Output	operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	complement

It is the circuit used to perform logic operations. Other logic operations

can be derived from these 4 operation

- The above circuit is one stage of log diagram.

Similary, for 4-bit registers, 4 MUX's are required

Logical

Logic microoperations can be used to manipulate bits in register.

Selective-set: set the selective bits in a register. (should be made 1's)

Setting → making 1

clearing → making 0

Ex: 1010 A before

1100 B (logic operand)

To set the first 2 bits of register A, corresponding bits of register B should be 1 and OR operation should be performed

1010 A before

1100 B (logic operand)

OR 1110 A after.

Selective-complement:

complementing the selective bits of a register

Ex: 1010 A before

1100 B (logic operand)

EOR 0110 A after

EOR operation is used for selective complement

Selective-clear

Making the selective bits of a register.

Ex: 1010 A before

1100 B (logic operand)

0010 A after

$A \wedge \bar{B}$

Mask

It is similar to selective clear but the bits in B corresponding to the selective bits in register A should be '0' while selective clear.

Ex: 1010 A before
1100 B (logic operand)
AND 1000 A after masking

Insert:

~~It is used in~~

→ Insert operation is used insert some new bits into the register.

→ To insert bits, ~~first~~ ~~next~~ mask operation is performed on the bits and then OR operation is performed

Ex: 0110 1010 A before
0000 1111 B (mask)
0000 1010 A after masking

0000 1010 A before
1001 0000 B (insert)
OR 1001 1010 A after insertion

Clear:

→ Clear operation is used to check whether the content of the two registers are equal or not

Ex:

1010 A

1010 B

EX-OR 0000 $A \leftarrow A \oplus B$

EX-OR operation is used for clear operation.

Result = 0000 indicates the content of 2 registers are equal

Shift Microoperations:

Content of register can be shifted left or right in shift left operation or shift right operation.

→ The serial I/p in shift left operation will be transfering the bit into right most position.

→ The serial input in shift right operation will be transfering the bit into left most position.

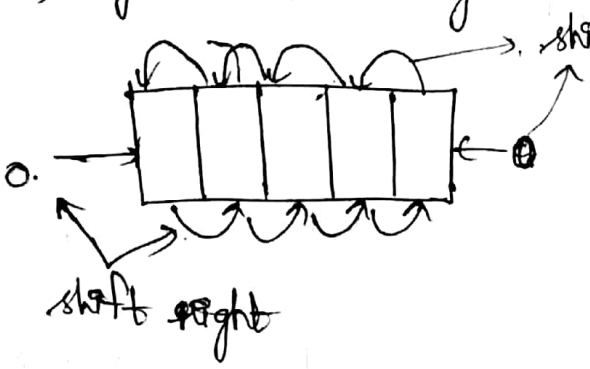
1) Logical shift
2) Circular shift
3) Arithmetic u

1) Logical shift:

It will be transfering '0' ~~to the serial~~ to the serial i/p

1) shift left - '0' will be inserted at LSB and then shifted
2) shift right - 0 " " " MSB and then shifted.

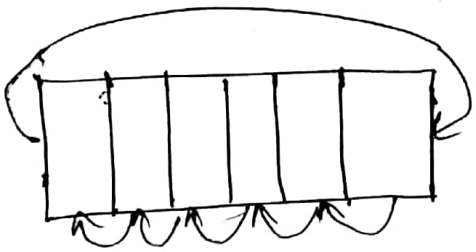
~~3) Logic shift & right~~



2) Circular shift:

The serial o/p of shift register is connected to the serial i/p of shift register.

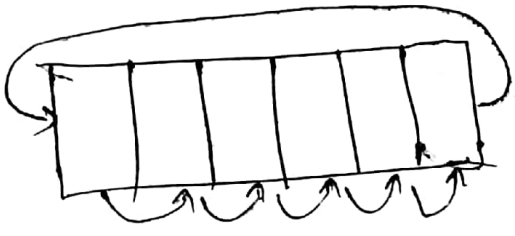
Ex: 1)



circular shift left

MSB bit is transfered to LSB

2)

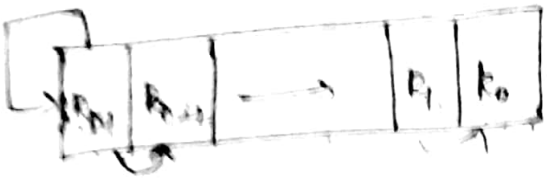


In circular shift operation LSB is transfered to MSB

Arithmetic shift:

Used to shift the signed binary numbers to left or right

Arithmetic shift right

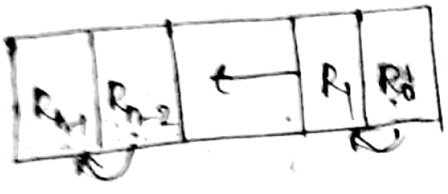


(R_{n-1} and R_{n-2} are same)

It transfers

shifts the signed binary numbers divided by 2

Arithmetic shift left:



After shift left operation, if the sign changes $\Rightarrow$ overflow has occured

$$V_s = R_{n-1} \oplus R_{n-2}$$

$V_s = 0 \Rightarrow$ if $R_{n-1} = R_{n-2}$

13/1/17 If $V_S = 0$, there is no overflow.

4-bit combinational circuit shifter:

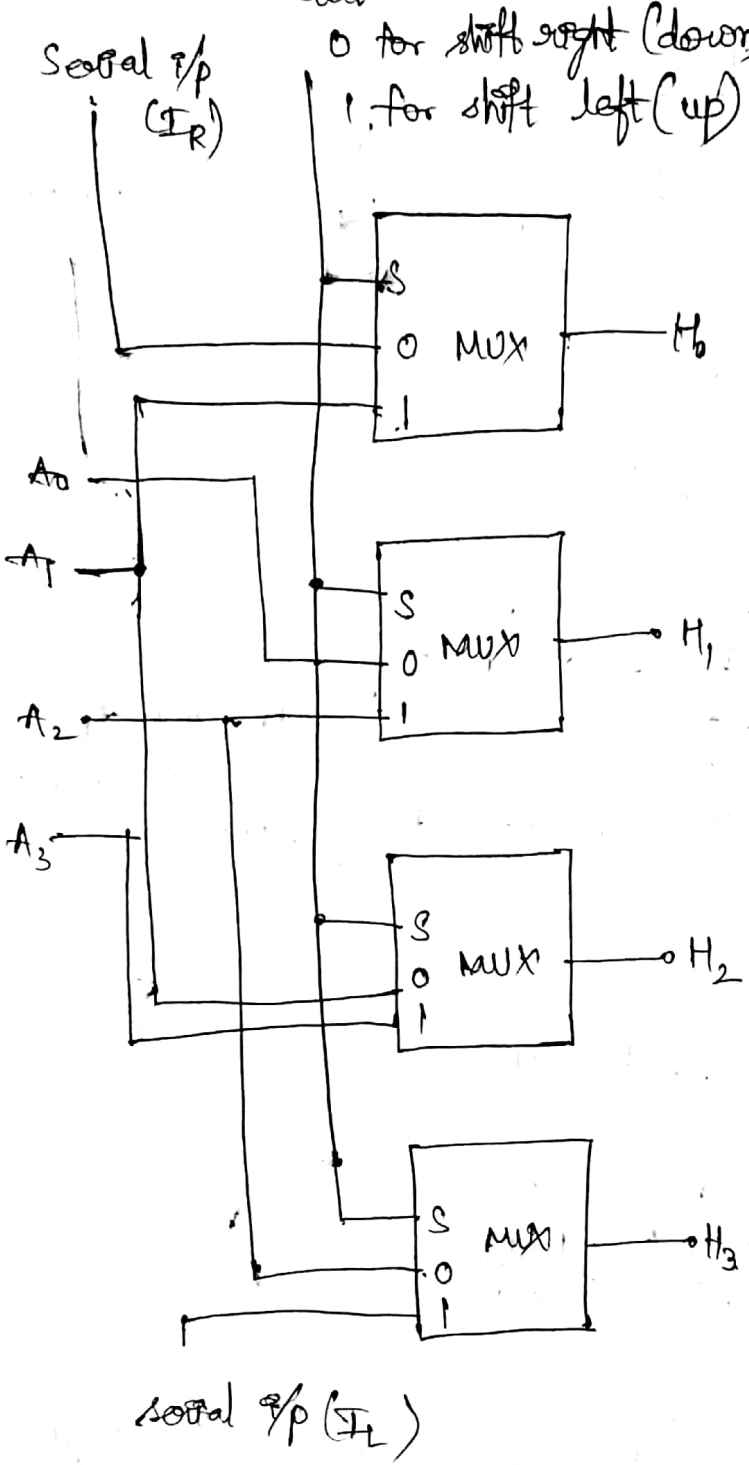


Fig. 4 bit ~~configuration~~ combinational circuit shifter. (for both logical and arithmetic shift)

I_R → serial i/p for shift right

I_L → " " " " left

Function table

Select	Output			
S	H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

The values of I_R depends on operation (logic or arithmetic)

Actual o/p is A_0 A_1 A_2 A_3. It is shifted to right/left depending on 'S'.

Symbolic designations of shift microoperations:

* $R \leftarrow shlR$

shl indicates shift left register (logical shift left)

* $R \leftarrow shrR$

shift right register R → indicates logical shift right " "

* $R \leftarrow cilR$

cil indicates circular shift left register R

* $R \leftarrow cirR$

cir indicates circular shift right register 'R'

* R ← ashl R

ashl → arithmetic shift left register R

* R ← ashr R

ashr → arithmetic shift right register R

Arithmetic Logic Shift Unit:

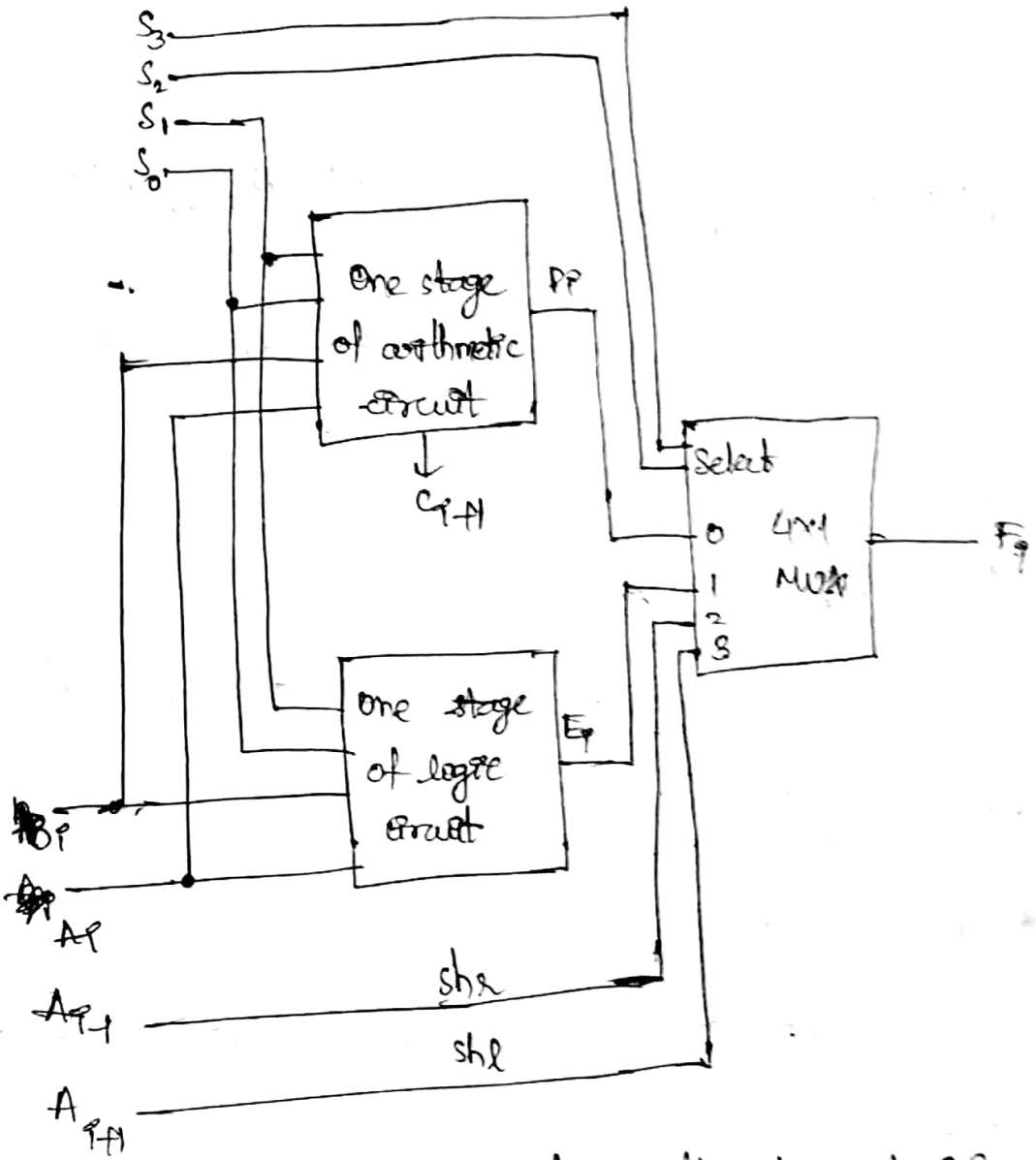


Fig. One stage of arithmetic logic shift circuit

14/7/17

It performs 14 operations

4 logical "
8 Arithmetic " (7 distinct arithmetic operations)
2 shift "

Function Table for Arithmetic 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	$F = A + 1$	Increment 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	$F = A + \overline{B}$	Subtract with borrow
0	0	1	0	1	$F = A + \overline{B} + 1$	Subtraction
0	0	1	1	0	$F = A - 1$	Decrement 1
0	0	1	1	1	$F = A$	Transfer A
0	1	0	0	X	$F = A \cap B$	AND
0	1	0	1	X	$F = A \vee B$	OR
0	1	1	0	X	$F = A \oplus B$	XOR
0	1	1	1	X	$F = \overline{A}$	Complement A
1	0	X	X	X	$F = shr A$	Shift right A into F
1	1	X	X	X	$F = shl A$	Shift left A into F

S_3, S_2 defines the ~~kind of operation~~ (logic or arithmetic or shift

S_1, S_0, C_{in} defines operation

① A 8-bit register contains the binary value 10011100. What is the register value after arithmetic shift right?

Starting from the initial number, 10011100, determine the register value after an arithmetic shift left. and state whether is an overflow.

Sol:

~~1000~~

1 0 0 1 1 1 0 0

~~Arithmetic~~ shift right

1 1 0 0 1 1 1 0

128 64 32 16 8 4 2

=206

100 111 00

00111000
3210 8

=56

Over flow (i.e. MSB changed) occurs

② Starting from an initial value of R = 11011101, determine the sequence of binary values in 'R' after a logical shift left followed by a circular shift right, followed by a logical shift right and a circular shift left.

Sol: R = 11011101

logical shift left = 10111010

circular shift right = 01011101

logical shift right = 00101110

circular shift left = 01011100

(3) Register A holds the 8-bit binary 11011001. Determine the B operand and the logic microoperation to be performed in order to change the value in A to a) 01101101 b) 11111101

Sol:

~~11011001~~

~~01111~~

11011001

01101101

~~11011001~~

~~10100101~~

11011001

01101101 (complemented)

A 11011001

B 10110101

01101100

* Error

11011001

11111101

Set operation or complement operation

~~(ex-or)~~ (OR) ~~(ex)~~ (EX-OR)

A 11011001

B 00100100

OR 11111101

or

A - 11011001

B - 00100100

EX-OR 11111101