COMPUTER ARCHITECTURE

8-bit addition

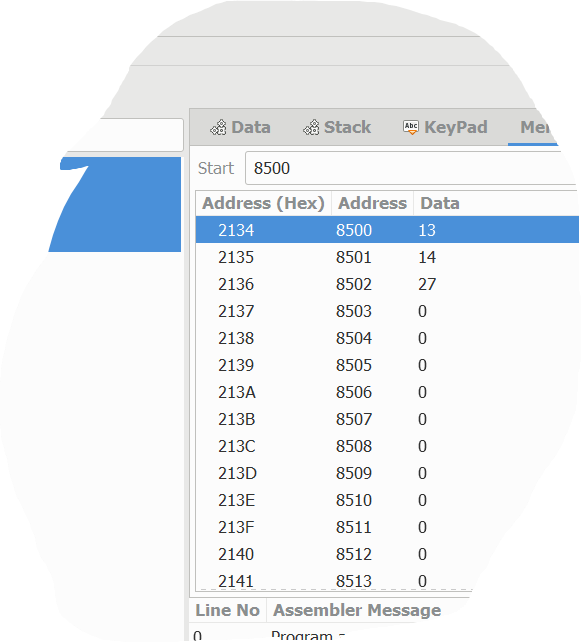
LDA 8500

MOV B,A

LDA 8501

ADD B

STA 8502

RST 1 

8-bit sub

LXI ,8500

MOV B,A

INX

MOV A,B

RST 1

INPUT:

8500 45H

8501 13H

OUTPUT:

8502 32H

8-BIT MULTI

LDA 8500

MOV B,A

LDA 8501

MOV C,A

CPI 00

JZ LOOP

XRA A

ADD B

DCR C

JZ LOOP

JMP LOOP1

STA 8502

RST 1

INPUT:

8500 45H

8501 13H

OUTPUT:

8502 32H

8-BIT MULTI

LDA 8500

MOV B,A

LDA 8501

MOV C,A

CPI 00

JZ LOOP+

XRA A

ADD B

DCR C

JZ LOOP

JMP LOOP1

STA 8502

RST1

INPUT:

8500 06H

8501 02H

OUTPUT:

8502 0CH

8-BIT DIVISION

LDA 8501

MOV B,A

LDA 8500

MVI C,00

|  |
| --- |
| CMP B |
| JC LOOP |
| SUB B |
| INR C |
| JMP LOOP |
| STA 8503 |
| MOV A,C |
| STA 8502 |
| RST 1 |

**INPUT:**

8500 06H

8501 02H

**OUTPUT:**

8502 03H(Quotient)

8503 00H(Reminder)

**16-BIT ADDITION**

|  |
| --- |
| MOV SI,1200H |
| LODSW |
| MOV BX,AX |
| LODSW |
| ADD BX,AX |
| MOV DI,1300H |
| MOV [DI],BX |
| HLT |

**INPUT:**

1200 13H 1201 13H

1202 14H 1203 14H

**OUTPUT:**

1300 27H

1301 27H

**16-BIT SUBTRACTION**

|  |
| --- |
| MOV SI,1200H |
| LODSW |
| MOV BX,AX |
| LODSW |
| SUB BX,AX |
| MOV DI,1300H |
| MOV [DI],BX |
| HLT |

**INPUT:**

1200 08H 1201 08H

1202 04H 1203 04H

**OUTPUT:**

1300 04H

1301 04H

Swap Two 8 bits numbers

MOV AL, [0600]

MOV BL, AL

MOV AL, [0601]

MOV [0600], AL

MOV [0601], BL

HLT

INPUT:

0600:05

0601:04

OUTPUT:

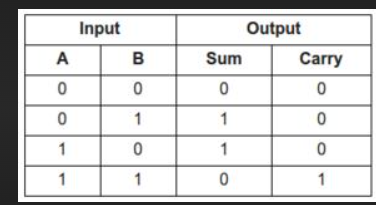
0600:04

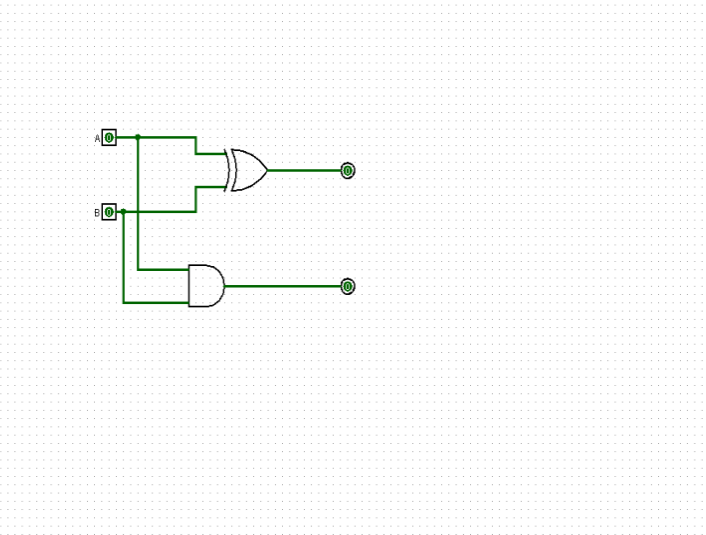
0601:05

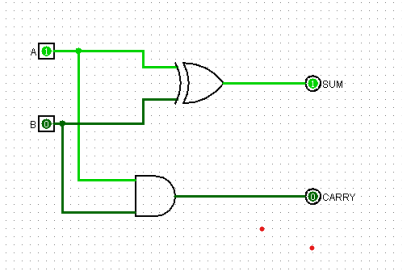
LOGISIM SIMULATOR

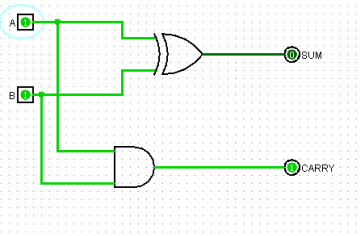
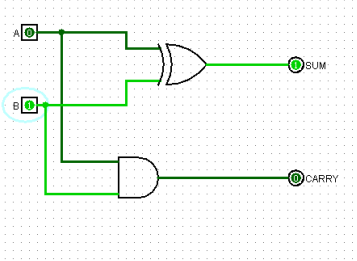
Half adder

Truth table:





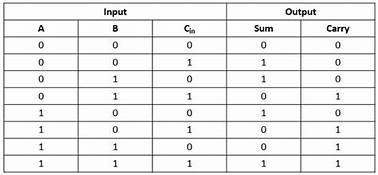


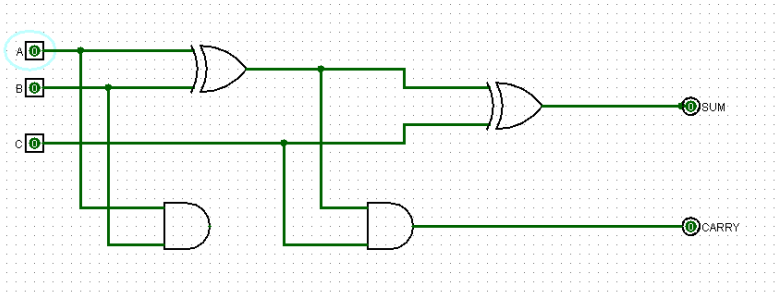


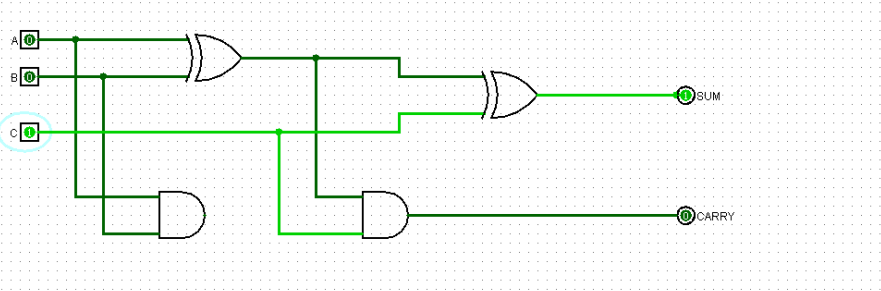
32-BIT FULL ADDER USING LOGISIM SIMULATOR

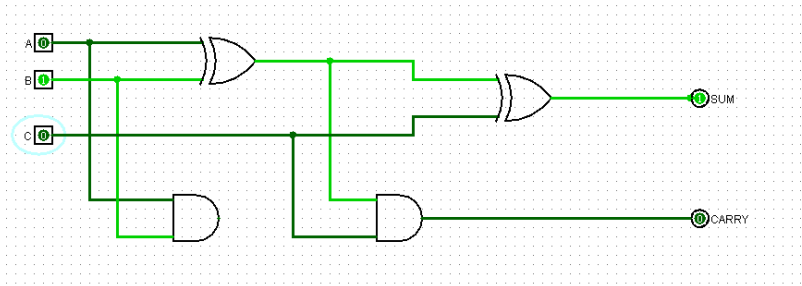
Full adder

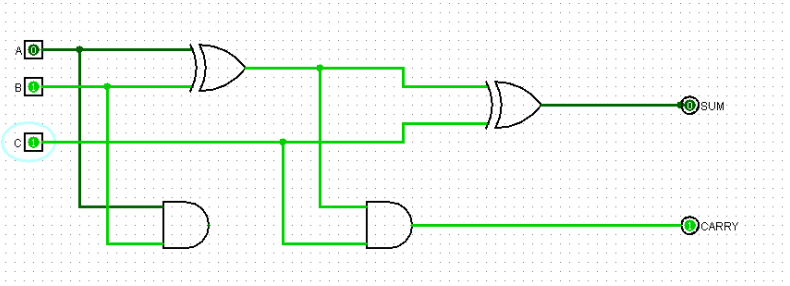
Truth table:

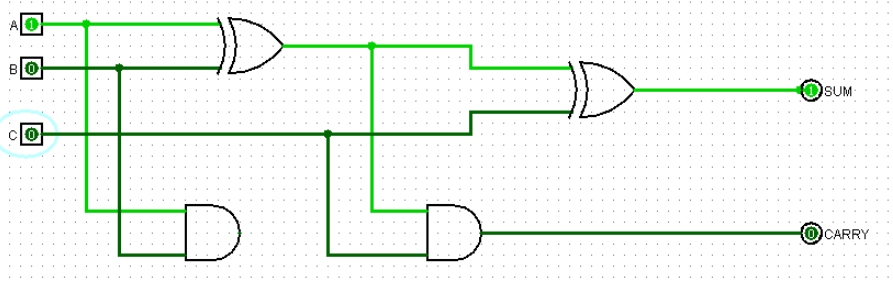


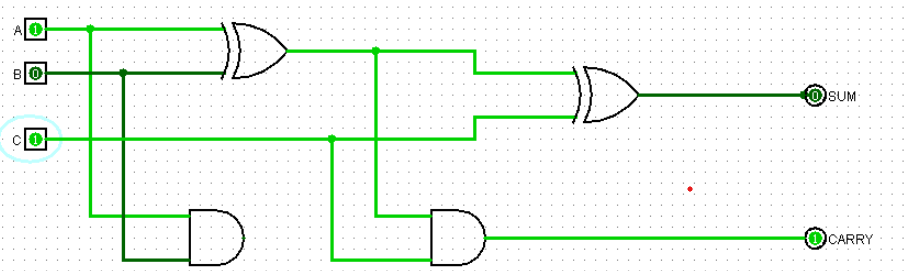


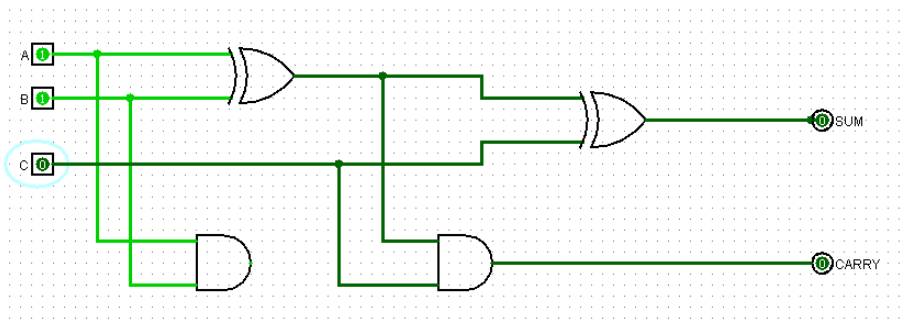


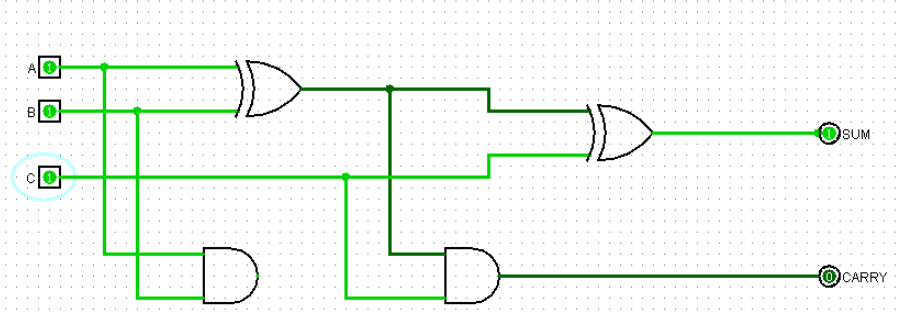












DESIGN OF 2 STAGE PIPELINE

Program:

import java.util.Scanner;

public class Main{

public static void main(String[]arges);{

int a[] = new int [3];

public static void main(String[]arges);{

int a[] = new int [3];

for(int i=0;i<3;i++){

a[i]=s.nextInt();

}

int instructions=0;

int clock =1;

int add=0;

int div =a[0];

int sub = a[0];

int stage =0;

int mul =1;

for(int i=0;ii<3;i++){

if(a[i]==0){

break;

}

else{

instructions =instructions+1;

clock = clock+1;

add = add+a[i];

mul = mul\*a[i];

sub = sub-a[i];

div =div/a[i];

}

} System.out.println("The number of clock cycles required are:\n"+clock);

System.out.println("addition:\n"+add); System.out.println("multiplication:\n"+mul);

System.out.println("highthroughput is:\n"); float throughput; throughput =instructions/clock;

System.out.println("The throughput requires for execution of the instruction:\n"+throughput+"%"); }

}