# LIST OF PROGRAMS

## EXPERIMENT 1:-ADDITION OF TWO 8 BIT DATA

LDA 8500

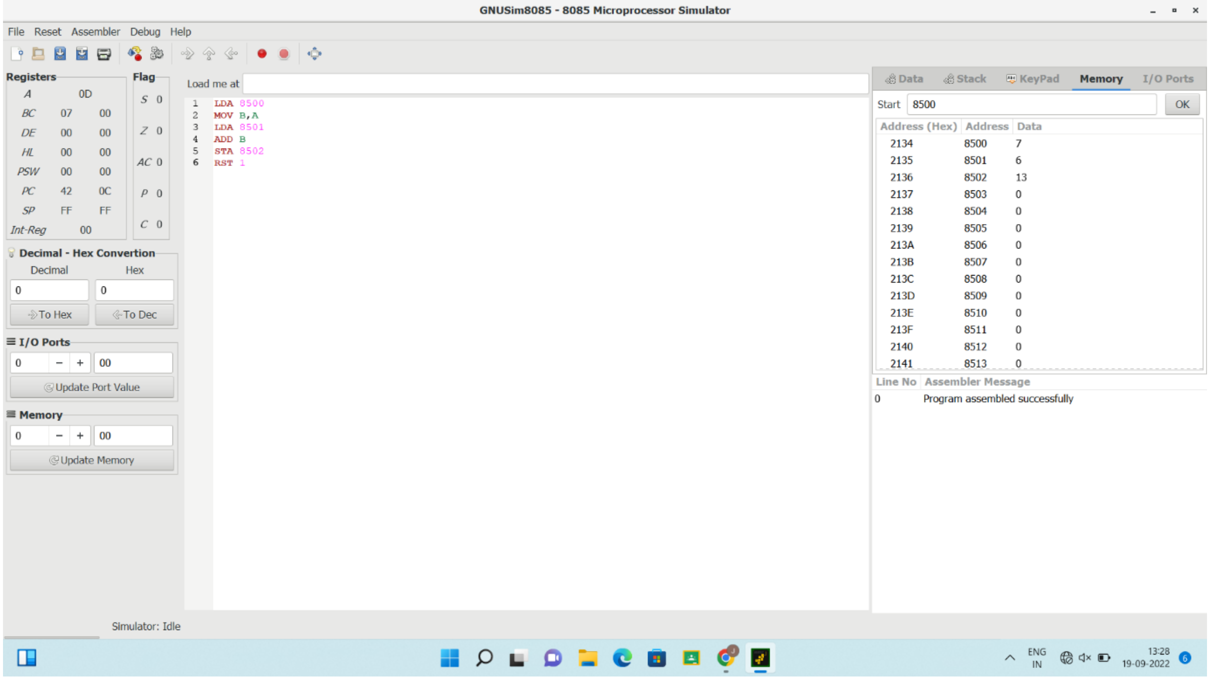
MOV B,A

LDA 8501

ADD B

STA 8502

RST1



```
LDA 8500
MOV B,A
LDA 8501
ADD B
STA 8502
RST 1
```

| Registers | | |
|---|---|---|
| A | 0D | |
| BC | 07 | 00 |
| DE | 00 | 00 |
| HL | 00 | 00 |
| PSW | 00 | 00 |
| PC | 42 | 0C |
| SP | FF | FF |
| Int-Reg | 00 | |

| Flag | |
|---|---|
| S | 0 |
| Z | 0 |
| AC | 0 |
| P | 0 |
| C | 0 |

Decimal: 0 Hex: 0

I/O Ports: 0 00

Memory: 0 00

Start: 8500

| Address (Hex) | Address | Data |
|---|---|---|
| 2134 | 8500 | 7 |
| 2135 | 8501 | 6 |
| 2136 | 8502 | 13 |
| 2137 | 8503 | 0 |
| 2138 | 8504 | 0 |
| 2139 | 8505 | 0 |
| 213A | 8506 | 0 |
| 213B | 8507 | 0 |
| 213C | 8508 | 0 |
| 213D | 8509 | 0 |
| 213E | 8510 | 0 |
| 213F | 8511 | 0 |
| 2140 | 8512 | 0 |
| 2141 | 8513 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

## EXPERIMENT 2: SUBTRACTION OF TWO 8 BIT DATA

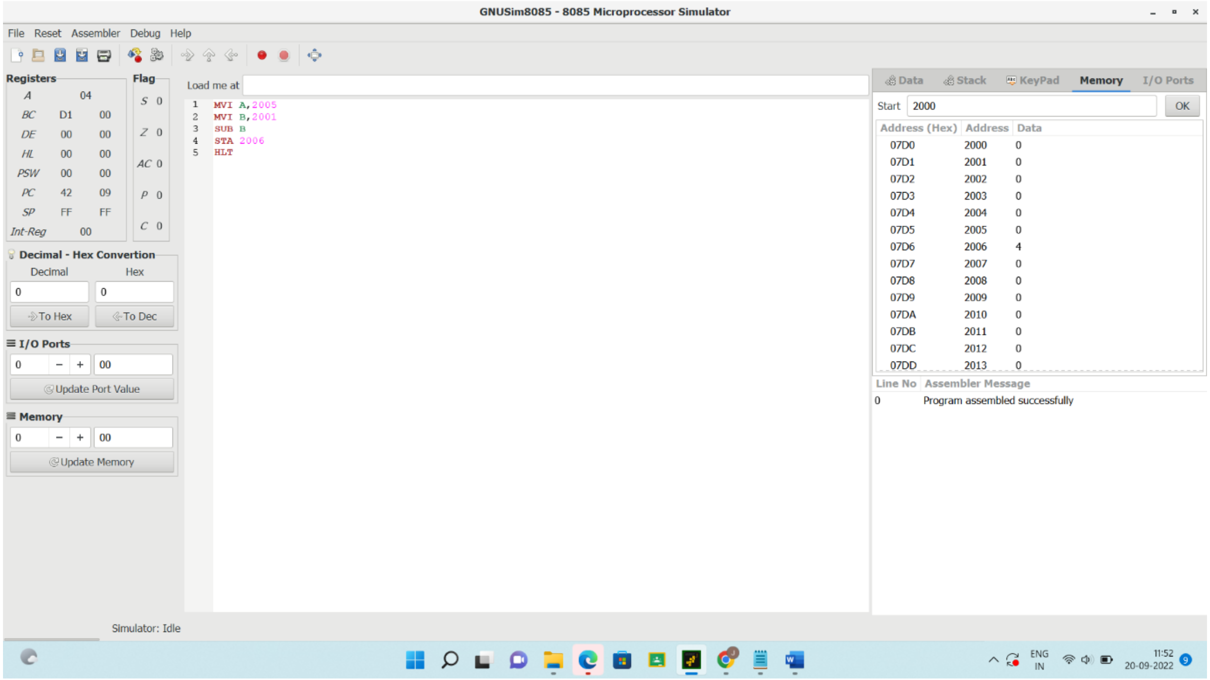
MVI A,2005

MVI B,2001

SUB B

STA 2006

HLT



Load me at

```
MVI A,2005
MVI B,2001
SUB B
STA 2006
HLT
```

| Registers | | |
|---|---|---|
| A | 04 | |
| BC | D1 | 00 |
| DE | 00 | 00 |
| HL | 00 | 00 |
| PSW | 00 | 00 |
| PC | 42 | 09 |
| SP | FF | FF |
| Int-Reg | 00 | |

| Flag | |
|---|---|
| S | 0 |
| Z | 0 |
| AC | 0 |
| P | 0 |
| C | 0 |

Start 2000

| Address (Hex) | Address | Data |
|---|---|---|
| 07D0 | 2000 | 0 |
| 07D1 | 2001 | 0 |
| 07D2 | 2002 | 0 |
| 07D3 | 2003 | 0 |
| 07D4 | 2004 | 0 |
| 07D5 | 2005 | 0 |
| 07D6 | 2006 | 4 |
| 07D7 | 2007 | 0 |
| 07D8 | 2008 | 0 |
| 07D9 | 2009 | 0 |
| 07DA | 2010 | 0 |
| 07DB | 2011 | 0 |
| 07DC | 2012 | 0 |
| 07DD | 2013 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

EXPERIMENT 3: ADDITION OF TWO 16 BIT DATA

LDA 2050

MOV B,A

LDA 2052

ADD B

STA 2062

LDA 2051

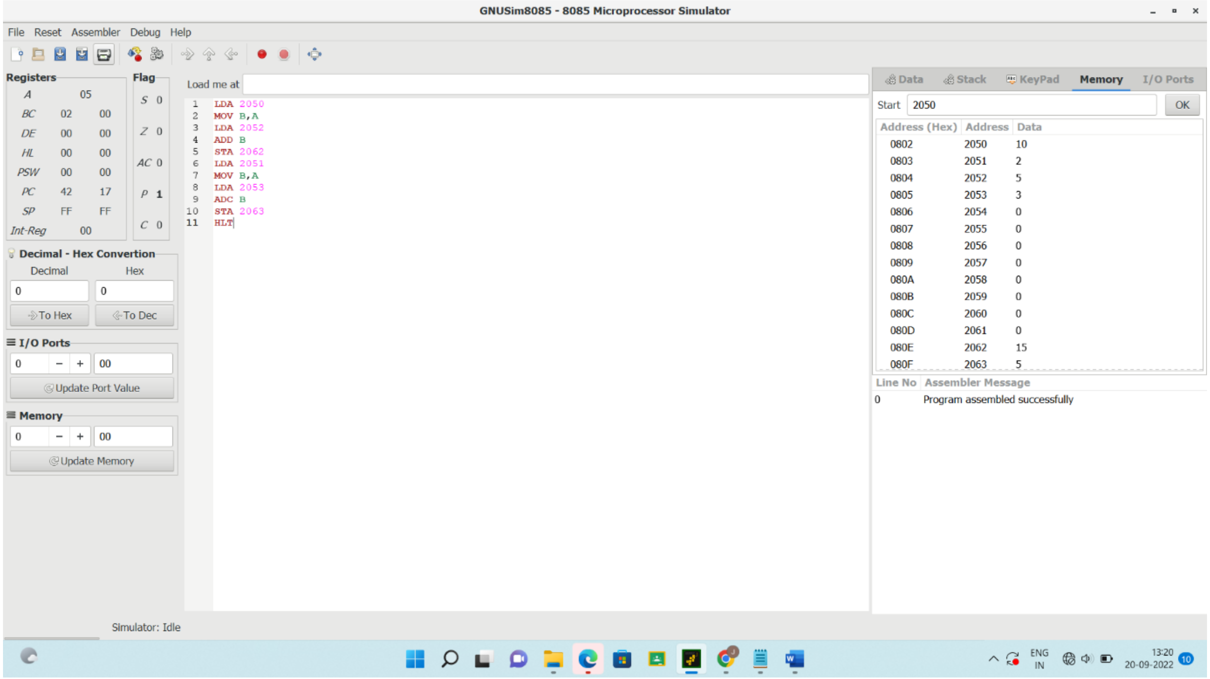
MOV B,A

LDA 2053

ADC B

STA 2063

HLT



```
LDA 2050
MOV B,A
LDA 2052
ADD B
STA 2062
LDA 2051
MOV B,A
LDA 2053
ADC B
STA 2063
HLT
```

| Registers | | |
|---|---|---|
| A | 05 | |
| BC | 02 | 00 |
| DE | 00 | 00 |
| HL | 00 | 00 |
| PSW | 00 | 00 |
| PC | 42 | 17 |
| SP | FF | FF |
| Int-Reg | 00 | |

| Flag | |
|---|---|
| S | 0 |
| Z | 0 |
| AC | 0 |
| P | 1 |
| C | 0 |

Start 2050

| Address (Hex) | Address | Data |
|---|---|---|
| 0802 | 2050 | 10 |
| 0803 | 2051 | 2 |
| 0804 | 2052 | 5 |
| 0805 | 2053 | 3 |
| 0806 | 2054 | 0 |
| 0807 | 2055 | 0 |
| 0808 | 2056 | 0 |
| 0809 | 2057 | 0 |
| 080A | 2058 | 0 |
| 080B | 2059 | 0 |
| 080C | 2060 | 0 |
| 080D | 2061 | 0 |
| 080E | 2062 | 15 |
| 080F | 2063 | 5 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

Simulator: Idle

# EXPERIMENT 4: SUBTRACTION OF TWO 16 BIT DATA

LDA 8500

MOV B,A

LDA 8501

MOV C,A

LDA 8502

MOV D,A

LDA 8503

SUB B

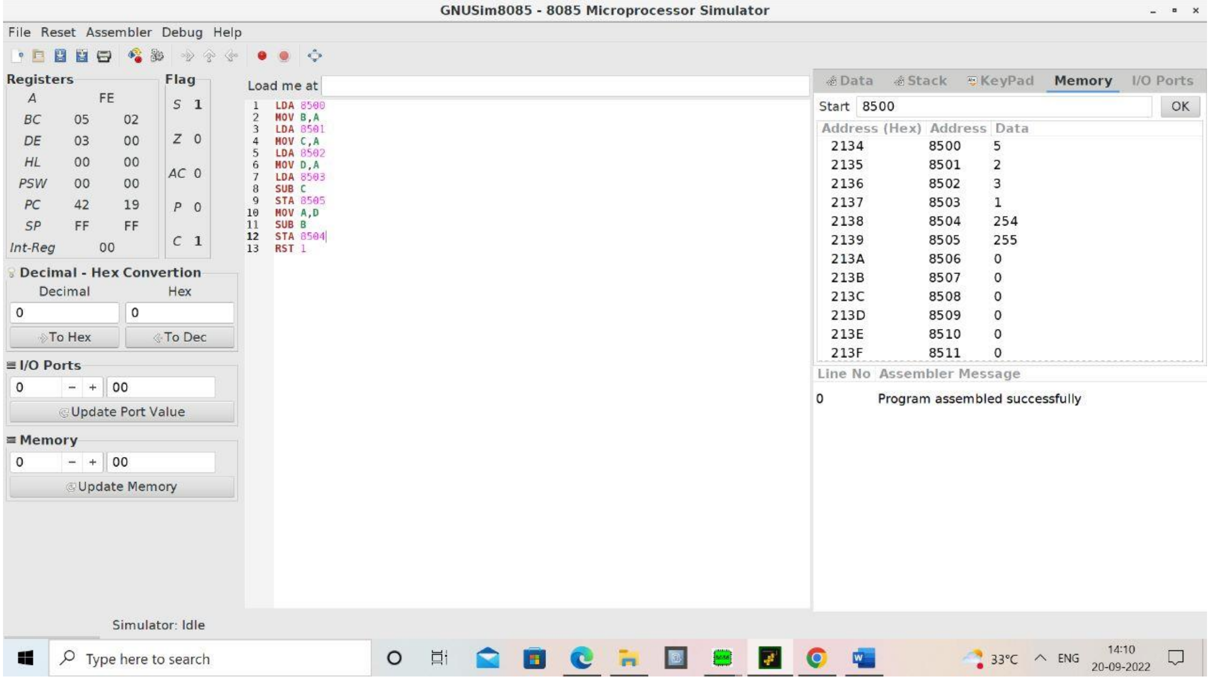
STA 8505

MOV A,D

SUB B

STA 8504

RST 1



```
LDA 8500
MOV B,A
LDA 8501
MOV C,A
LDA 8502
MOV D,A
LDA 8503
SUB C
STA 8505
MOV A,D
SUB B
STA 8504
RST 1
```

| Registers | | |
|---|---|---|
| A | FE | |
| BC | 05 | 02 |
| DE | 03 | 00 |
| HL | 00 | 00 |
| PSW | 00 | 00 |
| PC | 42 | 19 |
| SP | FF | FF |
| Int-Reg | 00 | |

| Flag | |
|---|---|
| S | 1 |
| Z | 0 |
| AC | 0 |
| P | 0 |
| C | 1 |

Decimal 0 Hex 0

I/O Ports 0 00

Memory 0 00

Start 8500

| Address (Hex) | Address | Data |
|---|---|---|
| 2134 | 8500 | 5 |
| 2135 | 8501 | 2 |
| 2136 | 8502 | 3 |
| 2137 | 8503 | 1 |
| 2138 | 8504 | 254 |
| 2139 | 8505 | 255 |
| 213A | 8506 | 0 |
| 213B | 8507 | 0 |
| 213C | 8508 | 0 |
| 213D | 8509 | 0 |
| 213E | 8510 | 0 |
| 213F | 8511 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

# EXPERIMENT 5: MULTIPLICATION OF TWO 8 BIT DATA

LDA 8500

MOV B,A

LDA 8501

MOV C,A

CPI 00

JZ LOOP1

XRA A

LOOP1: ADD B

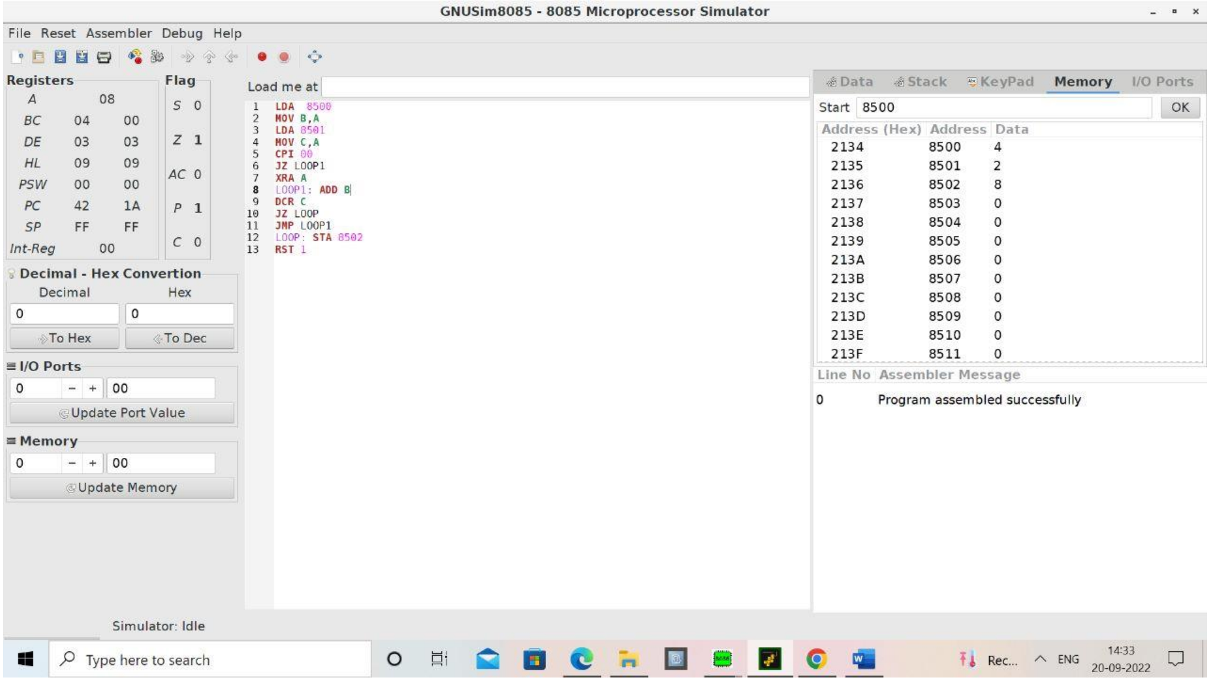
DCR C

JZ LOOP

JMP LOOP1

LOOP: STA 8502

RST 1



| Registers | | |
|---|---|---|
| A | 08 | |
| BC | 04 | 00 |
| DE | 03 | 03 |
| HL | 09 | 09 |
| PSW | 00 | 00 |
| PC | 42 | 1A |
| SP | FF | FF |
| Int-Reg | 00 | |

| Flag | |
|---|---|
| S | 0 |
| Z | 1 |
| AC | 0 |
| P | 1 |
| C | 0 |

Decimal: 0 Hex: 0

I/O Ports: 0 00

Memory: 0 00

Load me at

```
LDA  8500
MOV B,A
LDA 8501
MOV C,A
CPI 00
JZ LOOP1
XRA A
LOOP1: ADD B
DCR C
JZ LOOP
JMP LOOP1
LOOP: STA 8502
RST 1
```

Start 8500 OK

| Address (Hex) | Address | Data |
|---|---|---|
| 2134 | 8500 | 4 |
| 2135 | 8501 | 2 |
| 2136 | 8502 | 8 |
| 2137 | 8503 | 0 |
| 2138 | 8504 | 0 |
| 2139 | 8505 | 0 |
| 213A | 8506 | 0 |
| 213B | 8507 | 0 |
| 213C | 8508 | 0 |
| 213D | 8509 | 0 |
| 213E | 8510 | 0 |
| 213F | 8511 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

# EXPERIMENT 6:DIVISION OF TWO 8 BIT DATA

LDA 8501

MOV B,A

LDA 8500

MVI C,00

LOOP:  CMP B

JC LOOP1

SUB B

INR C

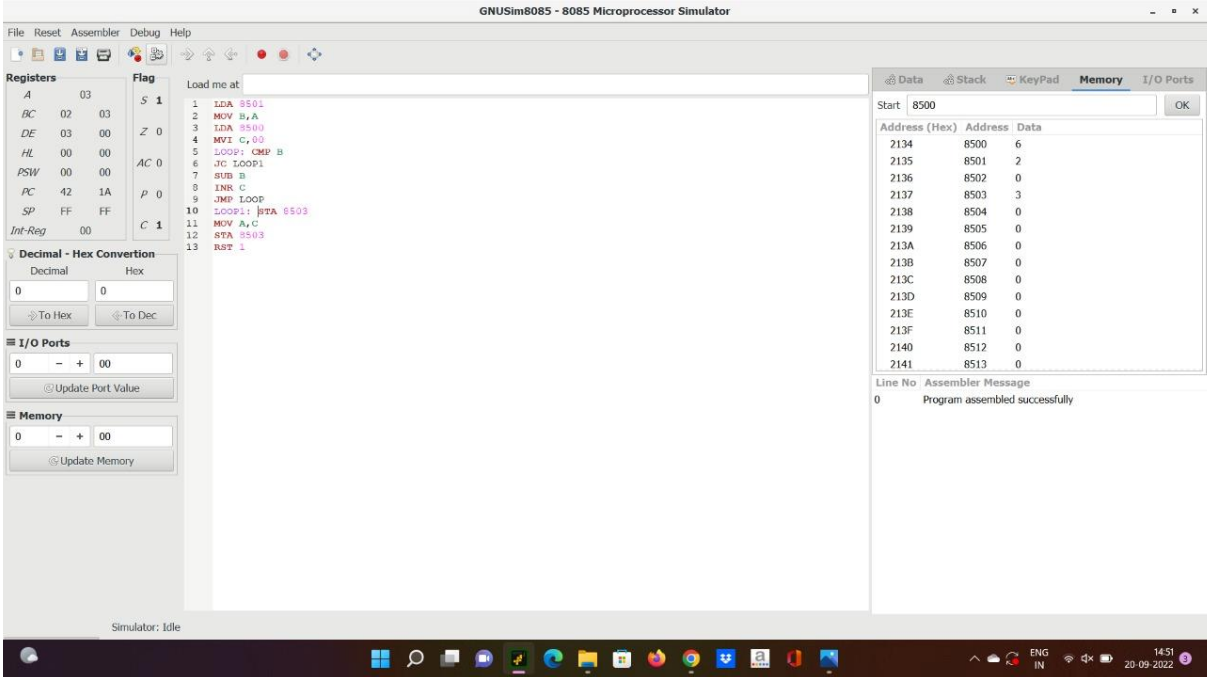
JMP LOOP

LOOP1: STA 8503

MOV A,C

STA 8503

RST 1



| Registers | | | Flag | |
|---|---|---|---|---|
| A | 03 | | S | 1 |
| BC | 02 | 03 | Z | 0 |
| DE | 03 | 00 | AC | 0 |
| HL | 00 | 00 | P | 0 |
| PSW | 00 | 00 | C | 1 |
| PC | 42 | 1A | | |
| SP | FF | FF | | |
| Int-Reg | 00 | | | |

```
LDA 8501
MOV B,A
LDA 8500
MVI C,00
LOOP: CMP B
JC LOOP1
SUB B
INR C
JMP LOOP
LOOP1: STA 8503
MOV A,C
STA 8503
RST 1
```

| Address (Hex) | Address | Data |
|---|---|---|
| 2134 | 8500 | 6 |
| 2135 | 8501 | 2 |
| 2136 | 8502 | 0 |
| 2137 | 8503 | 3 |
| 2138 | 8504 | 0 |
| 2139 | 8505 | 0 |
| 213A | 8506 | 0 |
| 213B | 8507 | 0 |
| 213C | 8508 | 0 |
| 213D | 8509 | 0 |
| 213E | 8510 | 0 |
| 213F | 8511 | 0 |
| 2140 | 8512 | 0 |
| 2141 | 8513 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

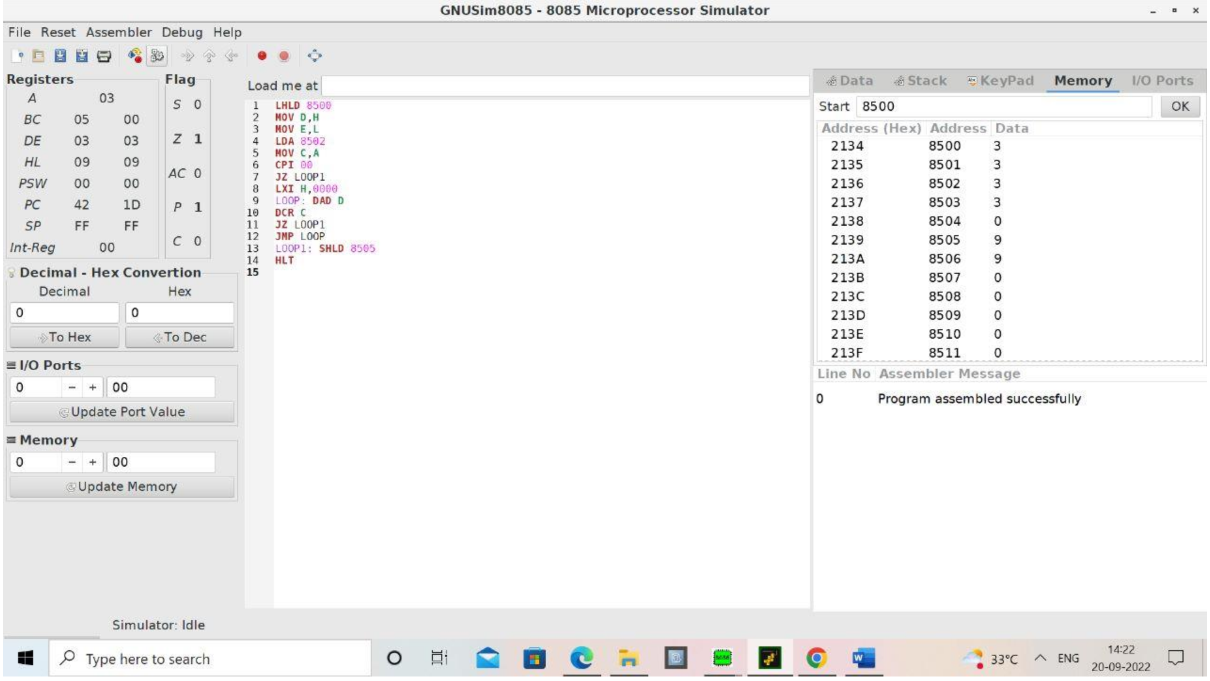
# EXPERIMENT 7:MULTIPLICATION OF TWO 16 BIT DATA

LHLD 8500

MOV D,H

MOV E,L

```
LDA 8502

MOV C,A

CPI 00

JZ LOOP1

LXI H,0000

LOOP: DAD D

DCR C

JZ LOOP1

JMP LOOP

LOOP1: SHLD 8505

HLT
```



Registers

| | | |
|---|---|---|
| A | 03 | |
| BC | 05 | 00 |
| DE | 03 | 03 |
| HL | 09 | 09 |
| PSW | 00 | 00 |
| PC | 42 | 1D |
| SP | FF | FF |
| Int-Reg | 00 | |

Flag

| | |
|---|---|
| S | 0 |
| Z | 1 |
| AC | 0 |
| P | 1 |
| C | 0 |

Load me at

```
LHLD 8500
MOV D,H
MOV E,L
LDA 8502
MOV C,A
CPI 00
JZ LOOP1
LXI H,0000
LOOP: DAD D
DCR C
JZ LOOP1
JMP LOOP
LOOP1: SHLD 8505
HLT
```

Start 8500

| Address (Hex) | Address | Data |
|---|---|---|
| 2134 | 8500 | 3 |
| 2135 | 8501 | 3 |
| 2136 | 8502 | 3 |
| 2137 | 8503 | 3 |
| 2138 | 8504 | 0 |
| 2139 | 8505 | 9 |
| 213A | 8506 | 9 |
| 213B | 8507 | 0 |
| 213C | 8508 | 0 |
| 213D | 8509 | 0 |
| 213E | 8510 | 0 |
| 213F | 8511 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

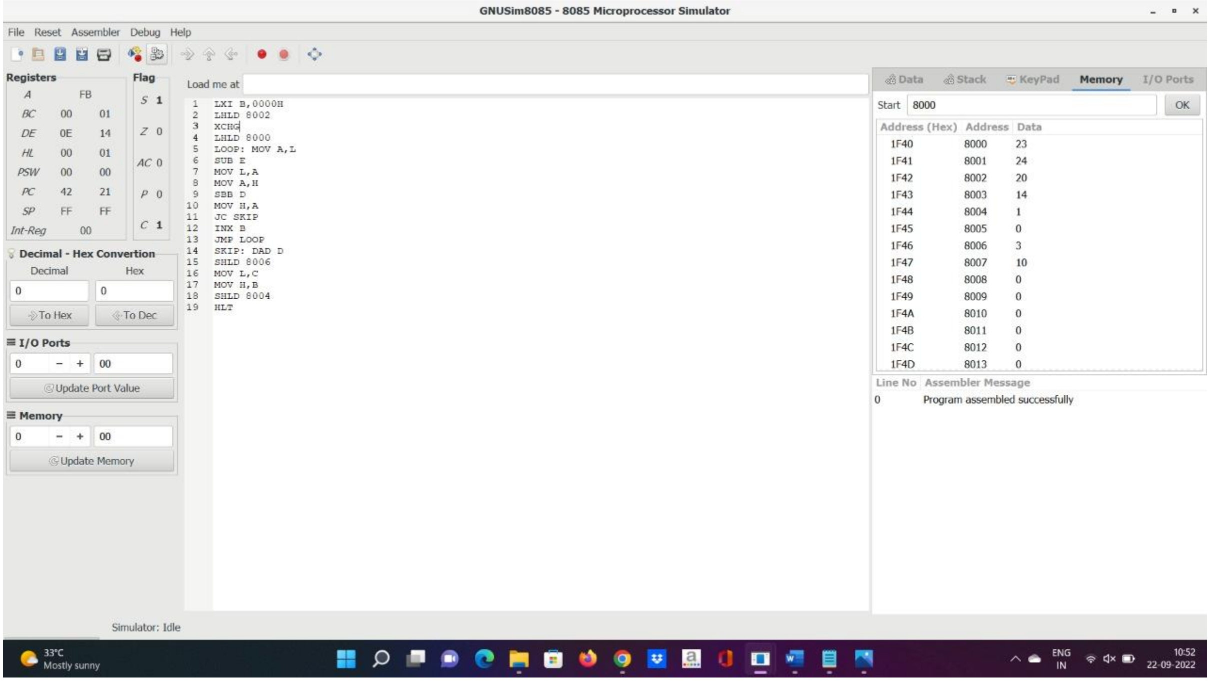
| Decimal | Hex |
|---|---|
| 0 | 0 |

I/O Ports: 0 – + 00

Memory: 0 – + 00

## EXPERIMENT: 8 DIVISION OF TWO 16 BIT DATA

```
LXI B,0000H
LHLD 8002
XCHG
LHLD 8000
LOOP: MOV A,L
SUB E
MOV L,A
MOV A,H
SBB D
MOV H,A
JC SKIP
INX B
JMP LOOP
SKIP: DAD D
SHLD 8006
MOV L,C
MOV H,B
SHLD 8004
HLT
```



| Registers | | |
|---|---|---|
| A | FB | |
| BC | 00 | 01 |
| DE | 0E | 14 |
| HL | 00 | 01 |
| PSW | 00 | 00 |
| PC | 42 | 21 |
| SP | FF | FF |
| Int-Reg | 00 | |

| Flag | |
|---|---|
| S | 1 |
| Z | 0 |
| AC | 0 |
| P | 0 |
| C | 1 |

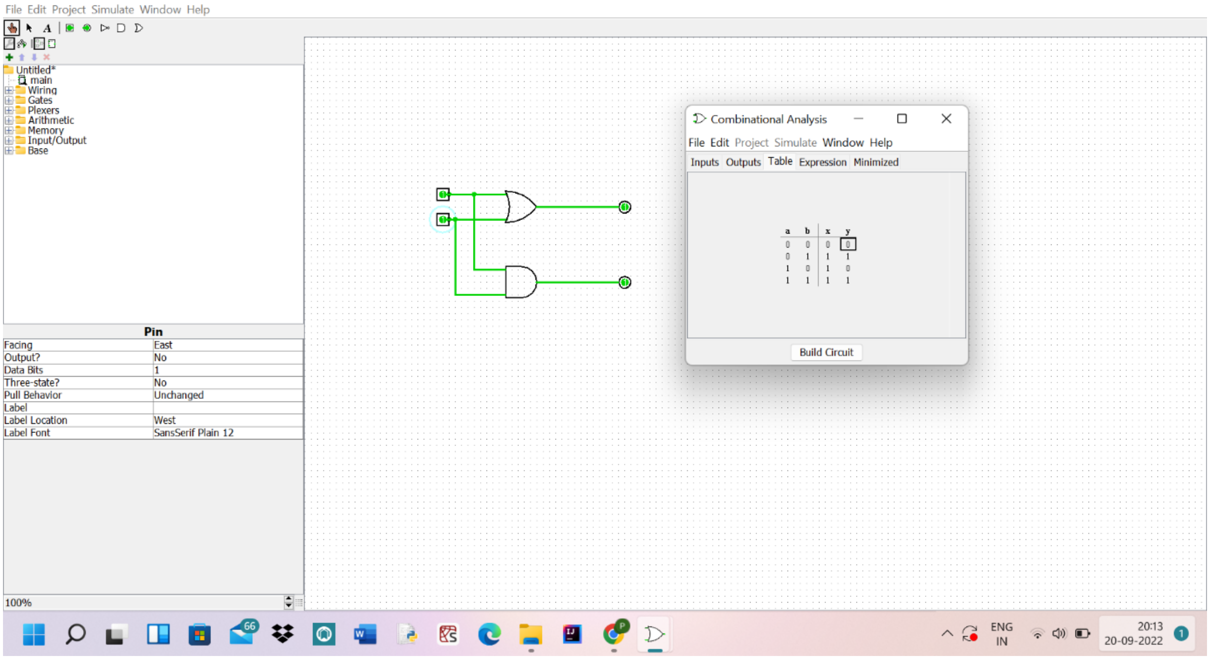
```
LXI B,0000H
LHLD 8002
XCHG
LHLD 8000
LOOP: MOV A,L
SUB E
MOV L,A
MOV A,H
SBB D
MOV H,A
JC SKIP
INX B
JMP LOOP
SKIP: DAD D
SHLD 8006
MOV L,C
MOV H,B
SHLD 8004
HLT
```

Start 8000

| Address (Hex) | Address | Data |
|---|---|---|
| 1F40 | 8000 | 23 |
| 1F41 | 8001 | 24 |
| 1F42 | 8002 | 20 |
| 1F43 | 8003 | 14 |
| 1F44 | 8004 | 1 |
| 1F45 | 8005 | 0 |
| 1F46 | 8006 | 3 |
| 1F47 | 8007 | 10 |
| 1F48 | 8008 | 0 |
| 1F49 | 8009 | 0 |
| 1F4A | 8010 | 0 |
| 1F4B | 8011 | 0 |
| 1F4C | 8012 | 0 |
| 1F4D | 8013 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

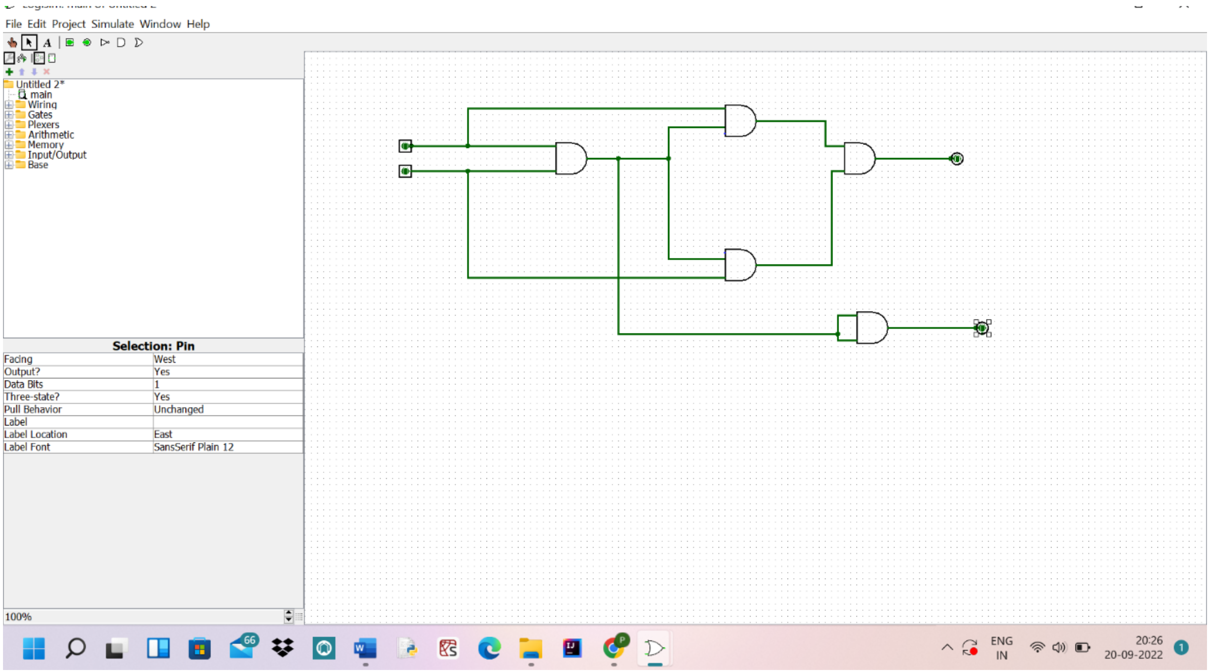
# EXPERIMENT 9: 2-BIT HALF ADDER



| Pin | |
|---|---|
| Facing | East |
| Output? | No |
| Data Bits | 1 |
| Three-state? | No |
| Pull Behavior | Unchanged |
| Label | |
| Label Location | West |
| Label Font | SansSerif Plain 12 |

| a | b | x | y |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 |

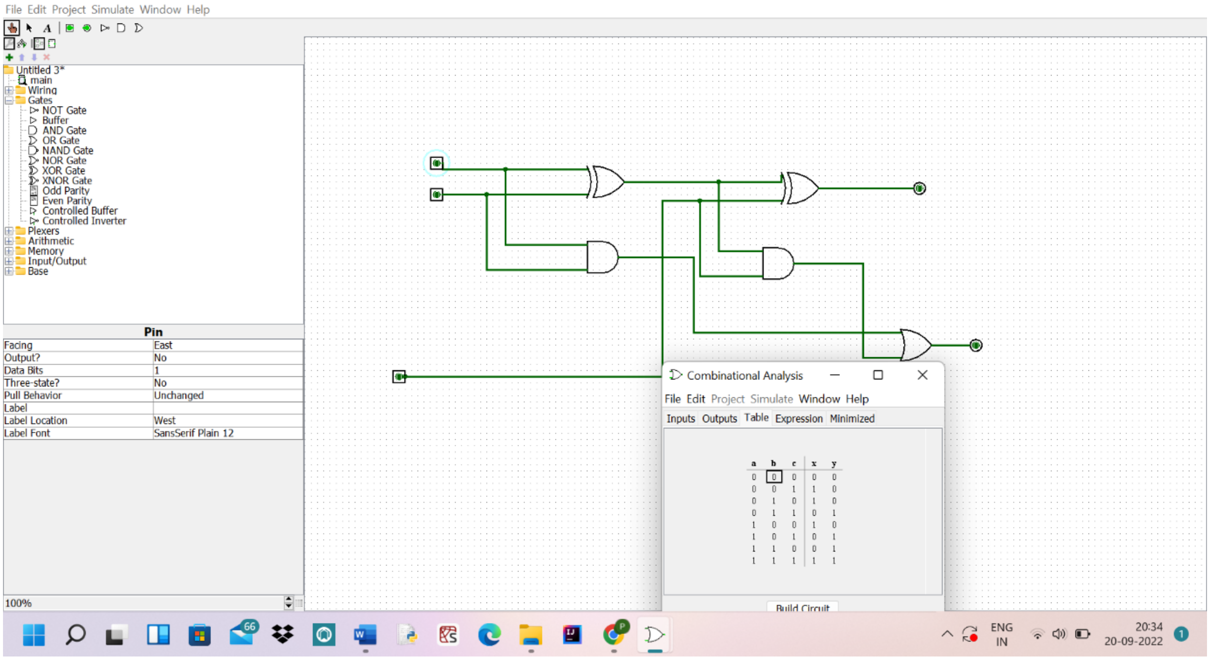
# EXPERIMENT 10: 3-BIT FULL ADDER



**Selection: Pin**

| | |
|---|---|
| Facing | West |
| Output? | Yes |
| Data Bits | 1 |
| Three-state? | Yes |
| Pull Behavior | Unchanged |
| Label | |
| Label Location | East |
| Label Font | SansSerif Plain 12 |

# EXPERIMENT 11: 2-BIT HALF ADDER WITH NAND



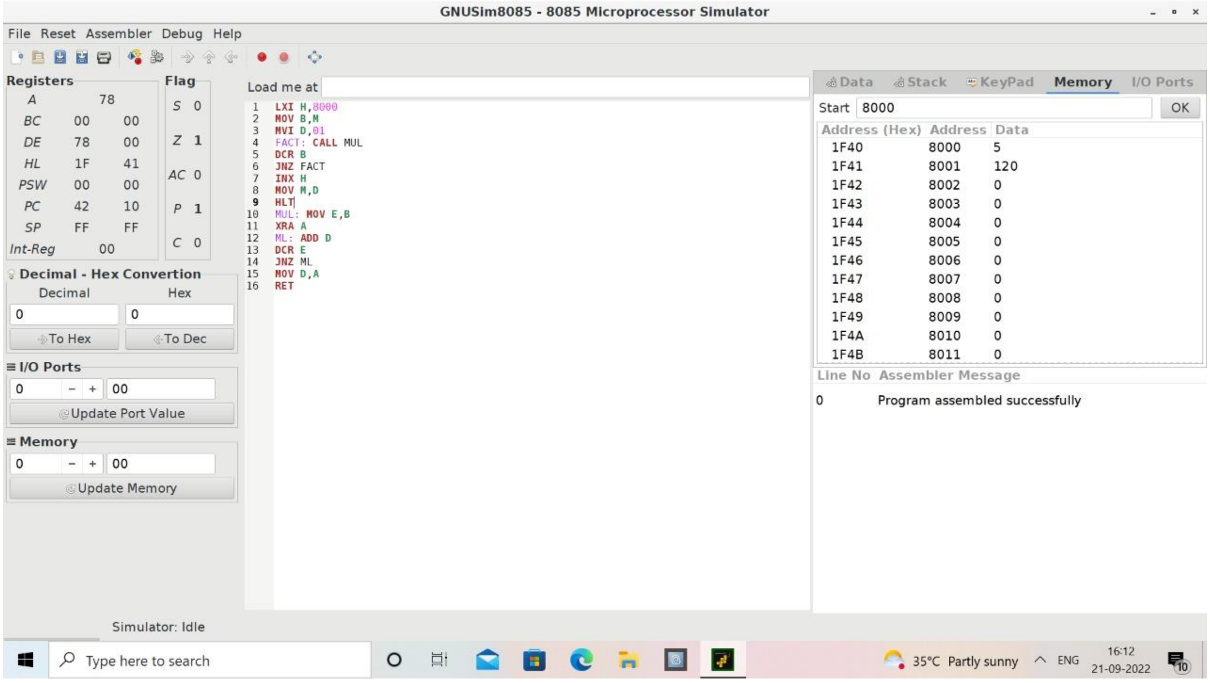
**Pin**

| | |
|---|---|
| Facing | East |
| Output? | No |
| Data Bits | 1 |
| Three-state? | No |
| Pull Behavior | Unchanged |
| Label | |
| Label Location | West |
| Label Font | SansSerif Plain 12 |

| 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 |

# EXPERIMENT 12: FACTORIAL OF N IN GIVEN NUMBER

```
LXI H,8000
MOV B,M
MVI D,01
FACT:  CALL MUL
DCR B
JNZ FACT
INX H
MOV M,D
HLT
MUL: MOV E,  B
XRA A
ML: ADD D
DCR E
JNZ ML
MOV D,A
RET
```



```
LXI H,8000
MOV B,M
MVI D,01
FACT: CALL MUL
DCR B
JNZ FACT
INX H
MOV M,D
HLT
MUL: MOV E,B
XRA A
ML: ADD D
DCR E
JNZ ML
MOV D,A
RET
```

Registers

|  |  |  |
|---|---|---|
| A | 78 |  |
| BC | 00 | 00 |
| DE | 78 | 00 |
| HL | 1F | 41 |
| PSW | 00 | 00 |
| PC | 42 | 10 |
| SP | FF | FF |
| Int-Reg | 00 |  |

Flag

|  |  |
|---|---|
| S | 0 |
| Z | 1 |
| AC | 0 |
| P | 1 |
| C | 0 |

Decimal: 0 Hex: 0

I/O Ports: 0 – + 00

Memory: 0 – + 00

Load me at

Start 8000

| Address (Hex) | Address | Data |
|---|---|---|
| 1F40 | 8000 | 5 |
| 1F41 | 8001 | 120 |
| 1F42 | 8002 | 0 |
| 1F43 | 8003 | 0 |
| 1F44 | 8004 | 0 |
| 1F45 | 8005 | 0 |
| 1F46 | 8006 | 0 |
| 1F47 | 8007 | 0 |
| 1F48 | 8008 | 0 |
| 1F49 | 8009 | 0 |
| 1F4A | 8010 | 0 |
| 1F4B | 8011 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

# EXPERIMMENT 13: LARGEST NUMBER IN ARRAY

LXI H,8000

MOV C,M

INX H

MOV B,M

DCR C

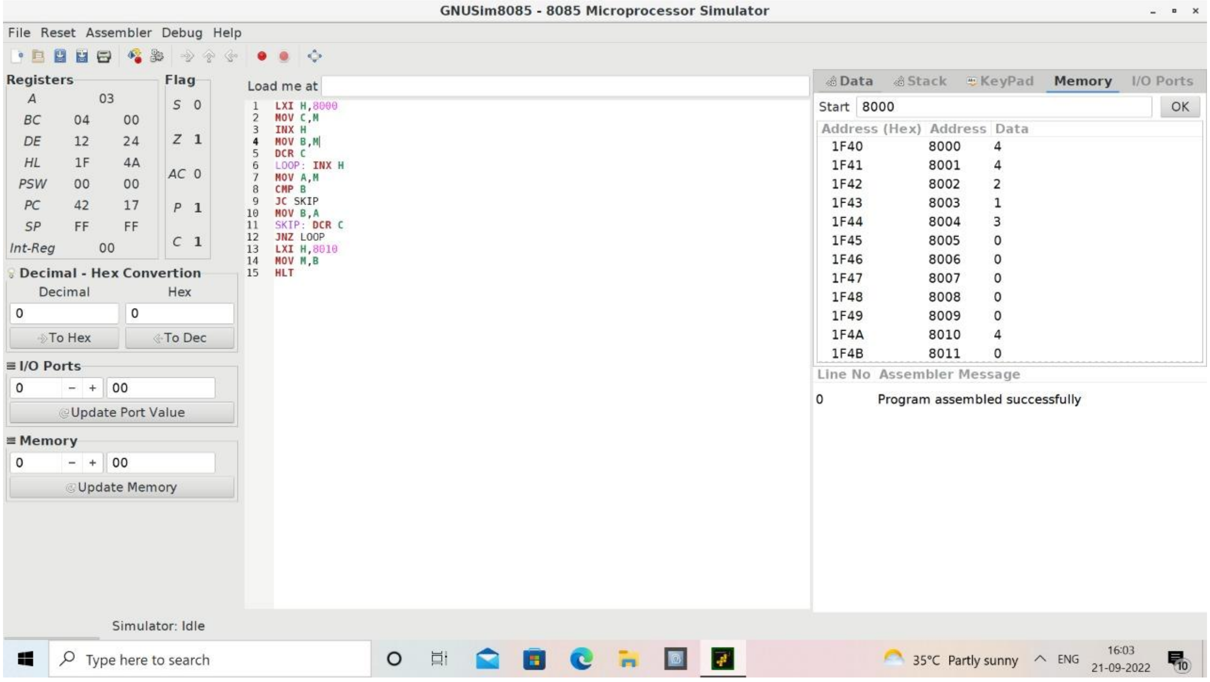
LOOP: INX H

MOV A,M

CMP B

JC SKIP

```
MOV B,A

SKIP: DCR C

JNZ LOOP

LXI H,8010

MOV M,B

HLT
```



| Registers | | |
|---|---|---|
| A | 03 | |
| BC | 04 | 00 |
| DE | 12 | 24 |
| HL | 1F | 4A |
| PSW | 00 | 00 |
| PC | 42 | 17 |
| SP | FF | FF |
| Int-Reg | 00 | |

| Flag | |
|---|---|
| S | 0 |
| Z | 1 |
| AC | 0 |
| P | 1 |
| C | 1 |

```
LXI H,8000
MOV C,M
INX H
MOV B,M
DCR C
LOOP: INX H
MOV A,M
CMP B
JC SKIP
MOV B,A
SKIP: DCR C
JNZ LOOP
LXI H,8010
MOV M,B
HLT
```

Start 8000 OK

| Address (Hex) | Address | Data |
|---|---|---|
| 1F40 | 8000 | 4 |
| 1F41 | 8001 | 4 |
| 1F42 | 8002 | 2 |
| 1F43 | 8003 | 1 |
| 1F44 | 8004 | 3 |
| 1F45 | 8005 | 0 |
| 1F46 | 8006 | 0 |
| 1F47 | 8007 | 0 |
| 1F48 | 8008 | 0 |
| 1F49 | 8009 | 0 |
| 1F4A | 8010 | 4 |
| 1F4B | 8011 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

## EXPERIMENT 14: 2 STAGE PIPELINE FOR ADDITION AND SUBTRACTION OF TWO NUMBERS

```
#include<stdio.h>

int main()

{

int counter =1,a,b,choice,res,ins;
```

```
printf("Enter number 1:");
scanf("%d",&a);
counter = counter+1;
printf("Enter number 2:");
scanf("%d",&b);
counter = counter +1;
printf("1-Addition:\n2-Subtraction:\n3-Multiplication:\n4-
Division:");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("Performing addition\n");
res = a+b;
counter = counter+1;
break;
case 2: printf("Performing subtraction\n");
res = a-b;
counter = counter+1;
break;
case 3: printf("Performing Multiplication\n");
res = a*b;
counter = counter+1;
break;
case 4: printf("Performing Division\n");
res = a/b;
counter = counter+1;
break;
default: printf("Wrong input");
break;
}
printf("The cycle value is:%d\n",counter);
printf("Enter the number of instructions:");
scanf("%d",&ins);
int performance_measure = ins/counter;
printf("The performance measure
is:%d\n",performance_measure);
return 0;
}
```

## EXPERIMENT 15:  3 STAGE PIPELINE FOR AND,OR,NAND OF TWO NUMBERS.

```
#include<stdio.h>
int main( )
{
float a,b,counter=1,res,INS;
float performance_measure;
printf("Enter the number 1: ");
scanf("%f",&a);
printf("Enter the number 2: ");
scanf("%f",&b);
counter =counter+1;
res=a || b;
counter=counter+2;
printf("enter no.of instruction:");
scanf("%f",&INS);
performance_measure=INS/counter;
printf("performance_measure:%f ",performance_measure);
return 0;
}
#include<stdio.h>
int main( )
{
float a,b,counter=1,res,INS;
float performance_measure;
printf("Enter the number 1: ");
scanf("%f",&a);
printf("Enter the number 2: ");
```

```
scanf("%f",&b);
counter =counter+1;
res=a&&b;
counter=counter+2;
printf("enter no.of instruction:");
scanf("%f",&INS);
performance_measure=INS/counter;
printf("performance_measure:%f ",performance_measure);
return 0;
}
```

## EXPERIMENT 16: 4 STAGE PIPELINE FOR MULTIPLICATION AND DIVISION OF TWO NUMBERS.

```
#include<stdio.h>
void main(){
int counter=0;
int input;
int num1,num2;
int op;
int res;
int ins;
int performance_measure=0;
printf("\n Enter 1st value: ");
scanf("%d",&num1);
counter+=1;
printf("\n Enter the 2nd value: ");
scanf("%d",&num2);
counter+=1;
printf("\n Enter the option:
\n1)Addition\n2)Subraction\n3)Multiplication\n4)Division");
scanf("%d",&op);
switch(op){
case 1:
printf("Performing addition operation");
printf("Performing addition operation");
res=num1+num2;
counter+=1;
break;
case 2:
printf("Performing subraction operation");
res=num1-num2;
counter+=1;
break;
case 3:
printf("Performing multiplication operation");
```

```
res=num1*num2;
counter+=1;
break;
case 4:
if(num2==0){
printf("\n Denominator can't be zero");
}
else{
printf("Performing division operation");
res=num1/num2;
counter+=1;
break;
}
default:
printf("Invalid case...");
counter+=3;
break;
}
printf("\n CYCLE VALUE IS : %d",counter);
printf("Enter the no.instruction");
scanf("%d",&ins);
performance_measure=ins/counter;
```

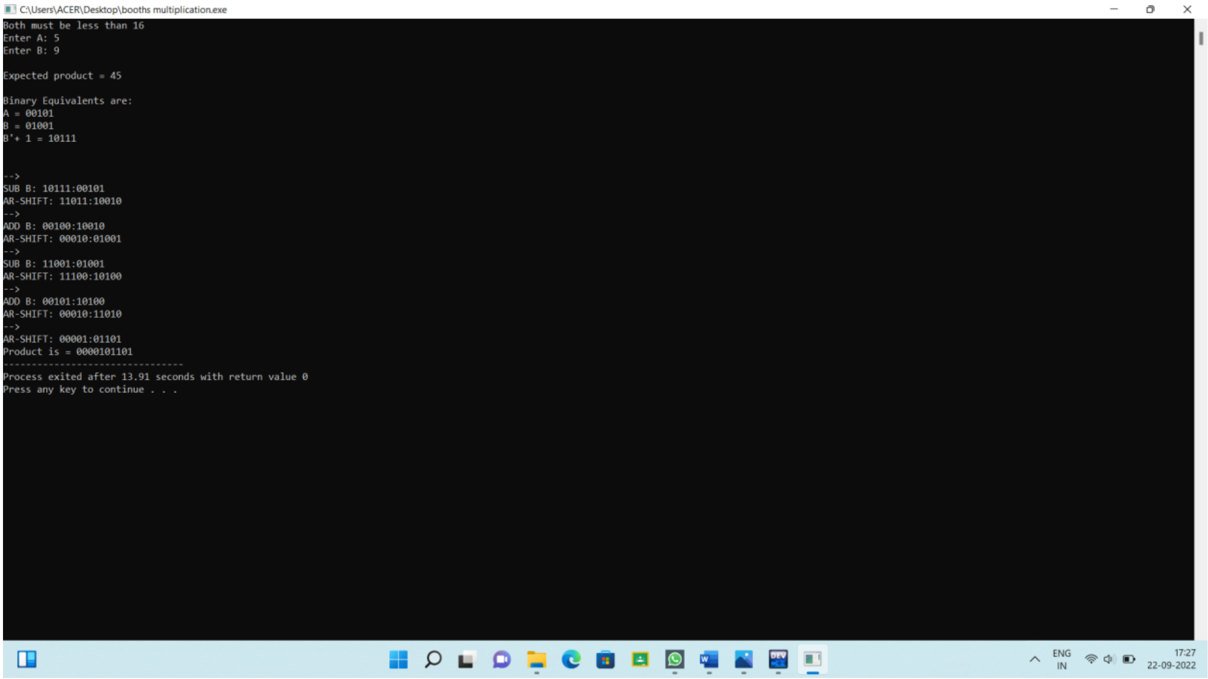
```
printf("\n Performance Measure is:
%d",performance_measure);
}
```

## EXPERIMENT 17: BOOTH’S MULTIPLICATION OF TWO SIGNED NUMBERS.

```
#include <stdio.h>
#include <math.h>
int a = 0,b = 0, c = 0, a1 = 0, b1 = 0, com[5] = { 1, 0, 0, 0, 0};
int anum[5] = {0}, anumcp[5] = {0}, bnum[5] = {0};
int acomp[5] = {0}, bcomp[5] = {0}, pro[5] = {0}, res[5] = {0};
void binary(){
    a1 = fabs(a);
    b1 = fabs(b);
    int r, r2, i, temp;
    for (i = 0; i < 5; i++){
        r = a1 % 2;
        a1 = a1 / 2;
        r2 = b1 % 2;
        b1 = b1 / 2;
        anum[i] = r;
        anumcp[i] = r;
        bnum[i] = r2;
        if(r2 == 0){
            bcomp[i] = 1;
        }
        if(r == 0){
            acomp[i] =1;
        }
  }
c = 0;
for ( i = 0; i < 5; i++){
        res[i] = com[i]+ bcomp[i] + c;
        if(res[i] >= 2){
            c = 1;
        }
        else
            c = 0;
        res[i] = res[i] % 2;
  }
for (i = 4; i >= 0; i--){
  bcomp[i] = res[i];
}
if (a < 0){
   c = 0;
  for (i = 4; i >= 0; i--){
        res[i] = 0;
  }
  for ( i = 0; i < 5; i++){
        res[i] = com[i] + acomp[i] + c;
        if (res[i] >= 2){
             c = 1;
        }
        else
             c = 0;
        res[i] = res[i]%2;
  }
  for (i = 4; i >= 0; i--){
        anum[i] = res[i];
        anumcp[i] = res[i];
  }
}
if(b < 0){
  for (i = 0; i < 5; i++){
        temp = bnum[i];
        bnum[i] = bcomp[i];
        bcomp[i] = temp;
```

```
     }
   }
}
void add(int num[]){
    int i;
    c = 0;
    for ( i = 0; i < 5; i++){
           res[i] = pro[i] + num[i] + c;
           if (res[i] >= 2){
                c = 1;
           }
           else{
                c = 0;
           }
           res[i] = res[i]%2;
     }
     for (i = 4; i >= 0; i--){
         pro[i] = res[i];
         printf("%d",pro[i]);
     }
   printf(":");
   for (i = 4; i >= 0; i--){
           printf("%d", anumcp[i]);
     }
}
void arshift(){//for arithmetic shift right
    int temp = pro[4], temp2 = pro[0], i;
    for (i = 1; i < 5  ; i++){//shift the MSB of product
       pro[i-1] = pro[i];
    }
    pro[4] = temp;
    for (i = 1; i < 5  ; i++){//shift the LSB of product
       anumcp[i-1] = anumcp[i];
    }
    anumcp[4] = temp2;
    printf("\nAR-SHIFT: ");//display together
    for (i = 4; i >= 0; i--){
       printf("%d",pro[i]);
    }
    printf(":");
    for(i = 4; i >= 0; i--){
       printf("%d", anumcp[i]);
    }
}
```

```
int main(){
   int i, q = 0;
   printf("\t\tBOOTH'S MULTIPLICATION ALGORITHM");
   printf("\nEnter two numbers to multiply: ");
   printf("\nBoth must be less than 16");
   //simulating for two numbers each below 16
   do{
        printf("\nEnter A: ");
        scanf("%d",&a);
        printf("Enter B: ");
        scanf("%d", &b);
     }while(a >=16 || b >=16);

    printf("\nExpected product = %d", a * b);
    binary();
    printf("\n\nBinary Equivalents are: ");
    printf("\nA = ");
    for (i = 4; i >= 0; i--){
        printf("%d", anum[i]);
    }
    printf("\nB = ");
    for (i = 4; i >= 0; i--){
        printf("%d", bnum[i]);
    }
    printf("\nB'+ 1 = ");
    for (i = 4; i >= 0; i--){
        printf("%d", bcomp[i]);
    }
    printf("\n\n");
    for (i = 0;i < 5; i++){
           if (anum[i] == q){//just shift for 00 or 11
               printf("\n-->");
               arshift();
               q = anum[i];
           }
           else if(anum[i] == 1 && q == 0){//subtract and shift for
10
              printf("\n-->");
              printf("\nSUB B: ");
              add(bcomp);//add two's complement to implement
subtraction
              arshift();
              q = anum[i];
        }
        else{//add ans shift for 01
          printf("\n-->");
          printf("\nADD B: ");
          add(bnum);
          arshift();
          q = anum[i];
        }
    }

    printf("\nProduct is = ");
    for (i = 4; i >= 0; i--){
        printf("%d", pro[i]);
    }
    for (i = 4; i >= 0; i--){
        printf("%d", anumcp[i]);
    }
}
```



```
Both must be less than 16
Enter A: 5
Enter B: 9

Expected product = 45

Binary Equivalents are:
A = 00101
B = 01001
B'+ 1 = 10111



-->
SUB B: 10111:00101
AR-SHIFT: 11011:10010
-->
ADD B: 00100:10010
AR-SHIFT: 00010:01001
-->
SUB B: 11001:01001
AR-SHIFT: 11100:10100
-->
ADD B: 00101:10100
AR-SHIFT: 00010:11010
-->
AR-SHIFT: 00001:01101
Product is = 0000101101
--------------------------------
Process exited after 13.91 seconds with return value 0
Press any key to continue . . .
```

# EXPERIMENT 18: RESTORING DIVISION OF TWO NUMBERS

```
#include<stdlib.h>
#include<stdio.h>
int acum[100]={0}   ;
void add(int acum[],int b[],int n);
int q[100],b[100];
int main()
{
int x,y;
printf("Enter the Number :");
scanf("%d%d",&x,&y);
int i=0;
```

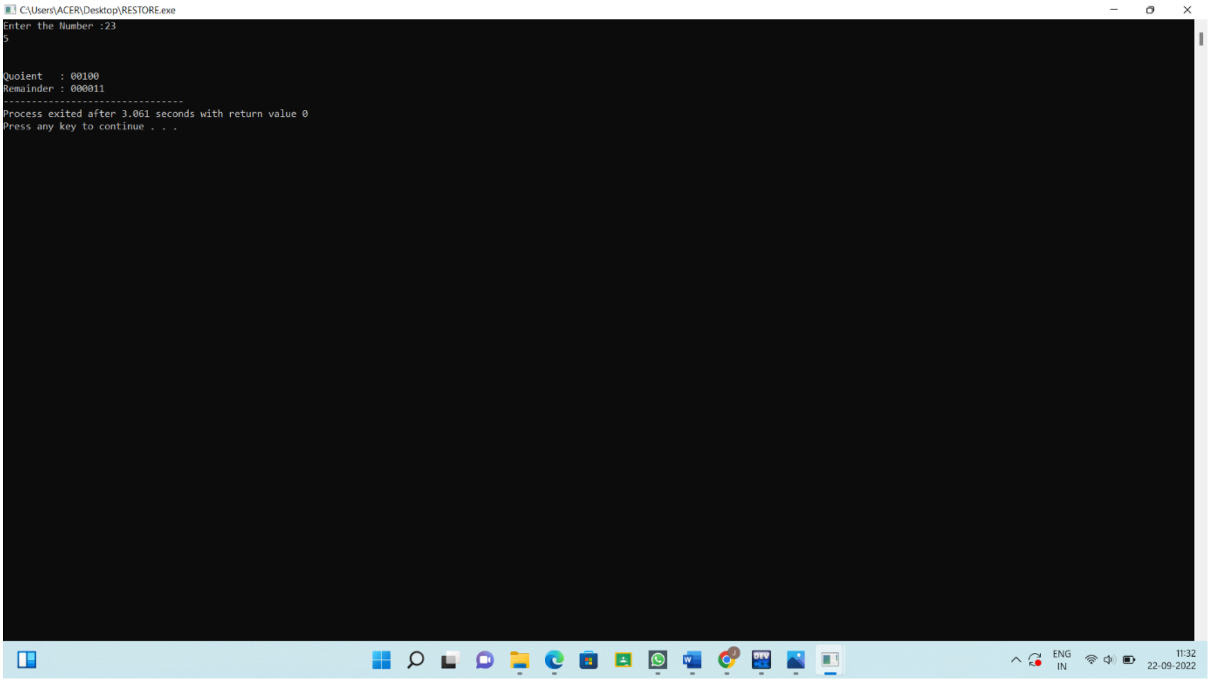
```
while(x>0||y>0)
{
if(x>0)
{
q[i]=x%2;
x=x/2;
}
else
{
q[i]=0;
}
if(y>0)
{
b[i]=y%2;
y=y/2;
}
else
{
b[i]=0;
}
i++;
}
```

```
int n=i;
int bc[50];
printf("\n");
for(i=0;i<n;i++)
{
if(b[i]==0)
{
bc[i]=1;
}
else
{
bc[i]=0;
}
}
bc[n]=1;
for(i=0;i<=n;i++)
{
if(bc[i]==0)
{
bc[i]=1;
i=n+2;
```

```
}
else
{
bc[i]=0;
}
}
int l;
 b[n]=0;
int k=n;
int n1=n+n-1;
int j,mi=n-1;
for(i=n;i!=0;i--)
{
for(j=n;j>0;j--)
{
acum[j]=acum[j-1];

}
acum[0]=q[n-1];
for(j=n-1;j>0;j--)
{
q[j]=q[j-1];
```

```
}


add(acum,bc,n+1);
if(acum[n]==1)
{
q[0]=0;
add(acum,b,n+1);
}
else
{
q[0]=1;
}
}
printf("\nQuoient   : ");

for(  l=n-1;l>=0;l--)
{
printf("%d",q[l]);


}
printf("\nRemainder : ");
for( l=n;l>=0;l--)
```

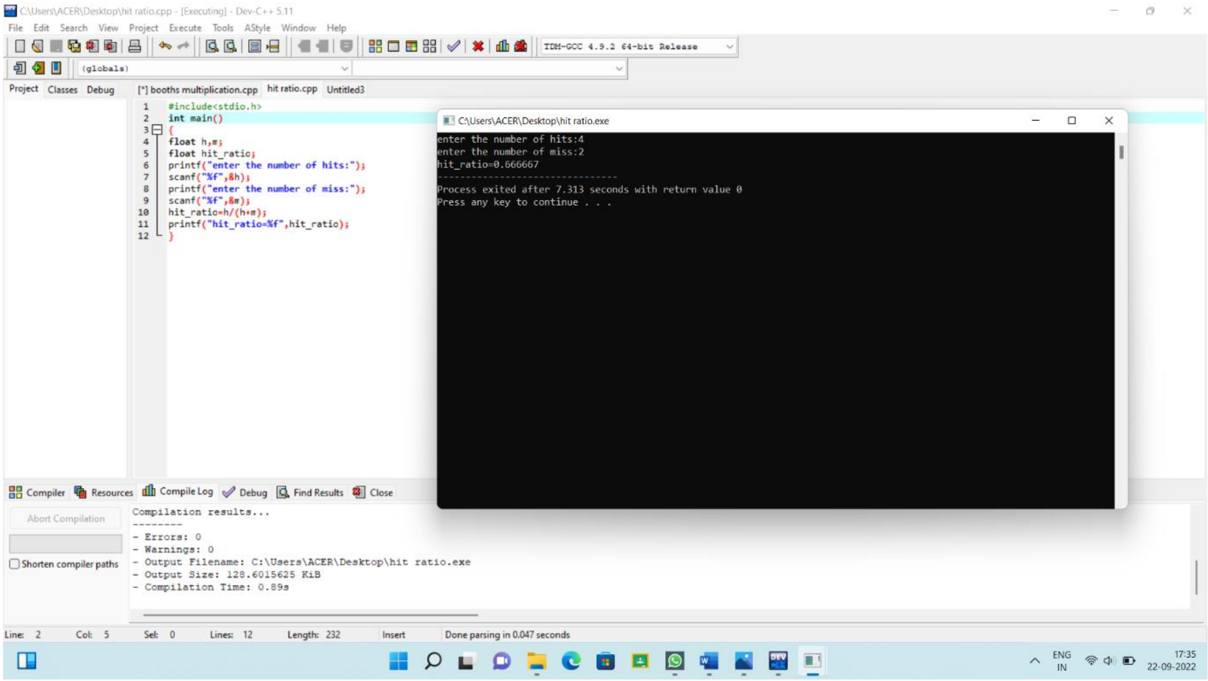
```
{
printf("%d",acum[l]);
}
return 0;
}
void add(int acum[],int bo[],int n)
{
int i=0,temp=0,sum=0;
for(i=0;i<n;i++)
{
sum=0;
sum=acum[i]+bo[i]+temp;
if(sum==0)
{
acum[i]=0;
temp=0;
}
else if (sum==2)
{
acum[i]=0;
temp=1;
}
else if(sum==1)
{
acum[i]=1;
temp=0;
}
else if(sum==3)
{
acum[i]=1;
temp=1;
}
}
}
```



```
Enter the Number :23
5


Quoient   : 00100
Remainder : 000011
--------------------------------
Process exited after 3.061 seconds with return value 0
Press any key to continue . . .
```

EXPERIMENT 19: FIND THE HIT RATIO OF THE GIVEN NUMBER.

```
#include<stdio.h>

int main()

{

float h,m;

float hit_ratio;

printf("enter the number of hits:");

scanf("%f",&h);

printf("enter the number of miss:");

scanf("%f",&m);

hit_ratio=h/(h+m);

printf("hit_ratio=%f",hit_ratio);

}
```



C:\Users\ACER\Desktop\hit ratio.cpp - [Executing] - Dev-C++ 5.11

```
#include<stdio.h>
int main()
{
float h,m;
float hit_ratio;
printf("enter the number of hits:");
scanf("%f",&h);
printf("enter the number of miss:");
scanf("%f",&m);
hit_ratio=h/(h+m);
printf("hit_ratio=%f",hit_ratio);
}
```

C:\Users\ACER\Desktop\hit ratio.exe

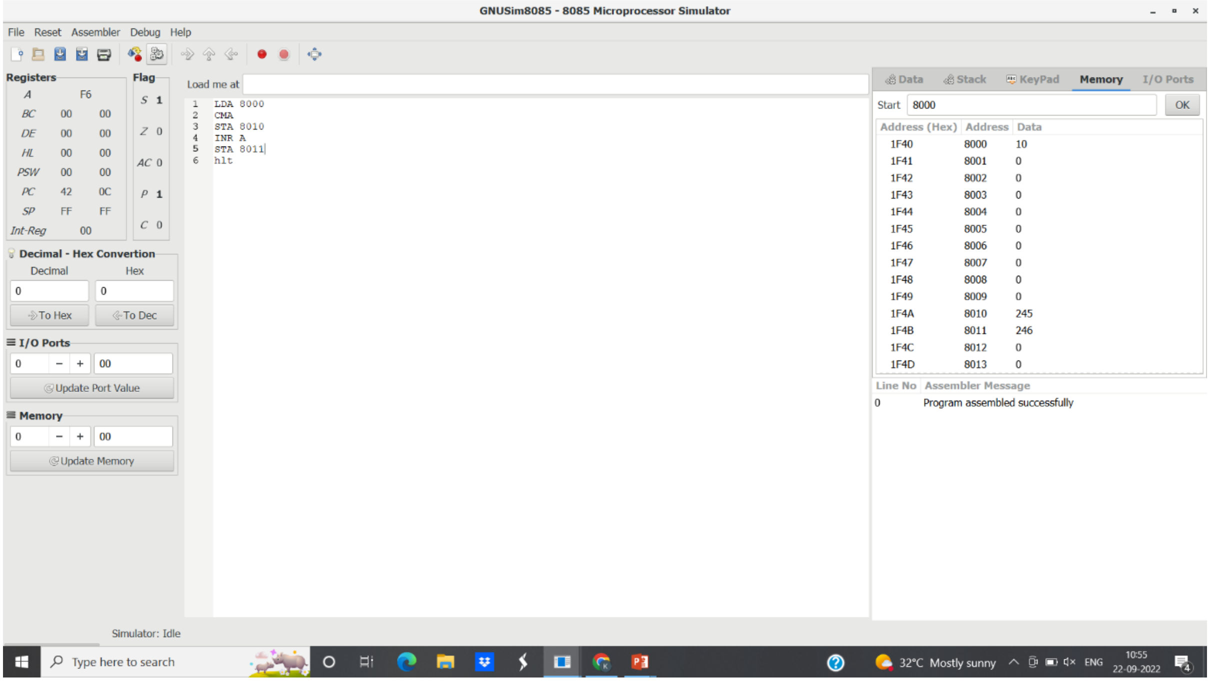
```
enter the number of hits:4
enter the number of miss:2
hit_ratio=0.666667
--------------------------------
Process exited after 7.313 seconds with return value 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\ACER\Desktop\hit ratio.exe
- Output Size: 128.6015625 KiB
- Compilation Time: 0.89s
```

Line: 2 Col: 5 Sel: 0 Lines: 12 Length: 232 Insert Done parsing in 0.047 seconds

## EXPERIMENT 20: 1’S AND 2’S COMPLIMENT OF 8 BIT NUMBER

```
LDA 8000
CMA
STA 8010
INR A
STA 8011
HLT
```



| Registers | | |
|---|---|---|
| A | F6 | |
| BC | 00 | 00 |
| DE | 00 | 00 |
| HL | 00 | 00 |
| PSW | 00 | 00 |
| PC | 42 | 0C |
| SP | FF | FF |
| Int-Reg | 00 | |

| Flag | |
|---|---|
| S | 1 |
| Z | 0 |
| AC | 0 |
| P | 1 |
| C | 0 |

```
LDA 8000
CMA
STA 8010
INR A
STA 8011
hlt
```

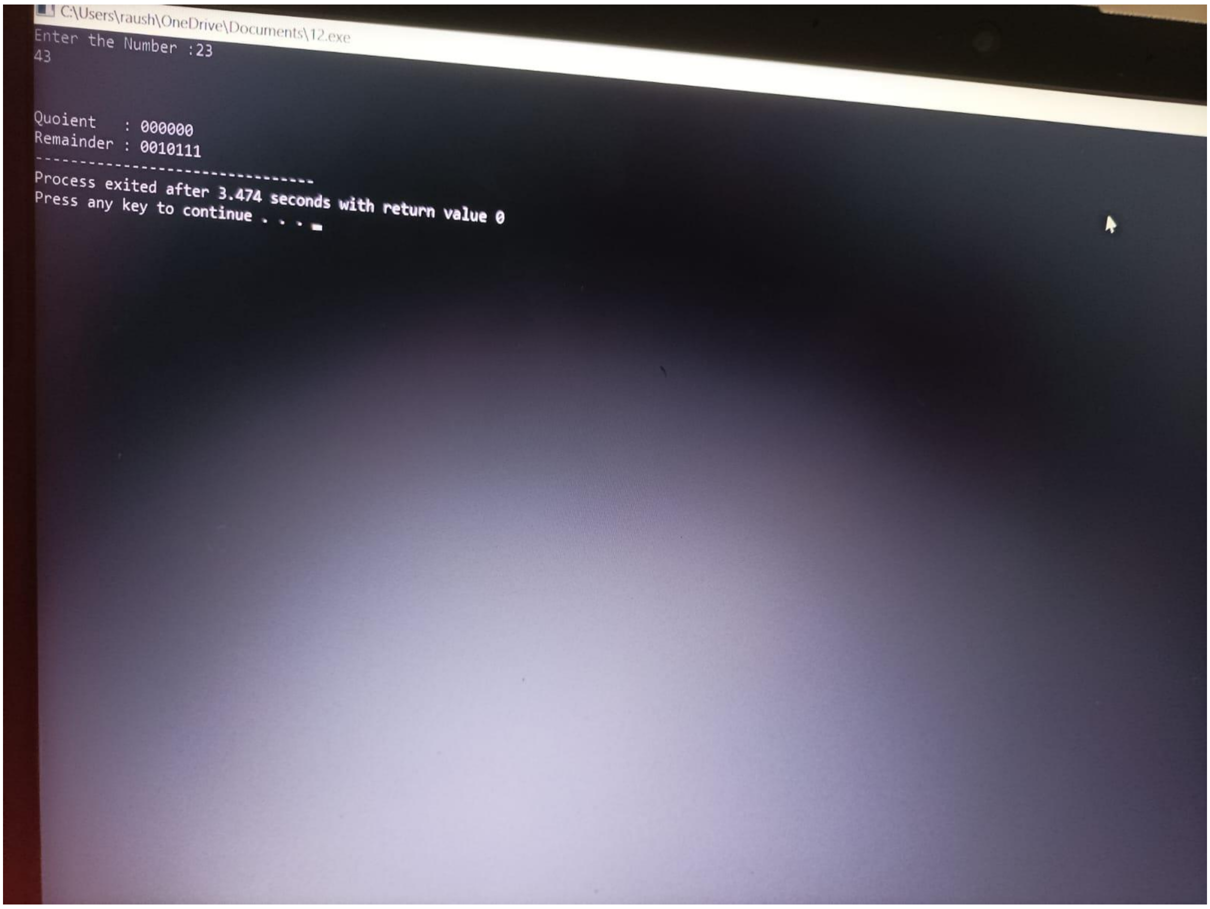
Start 8000

| Address (Hex) | Address | Data |
|---|---|---|
| 1F40 | 8000 | 10 |
| 1F41 | 8001 | 0 |
| 1F42 | 8002 | 0 |
| 1F43 | 8003 | 0 |
| 1F44 | 8004 | 0 |
| 1F45 | 8005 | 0 |
| 1F46 | 8006 | 0 |
| 1F47 | 8007 | 0 |
| 1F48 | 8008 | 0 |
| 1F49 | 8009 | 0 |
| 1F4A | 8010 | 245 |
| 1F4B | 8011 | 246 |
| 1F4C | 8012 | 0 |
| 1F4D | 8013 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |

# EXPERIMENT:21-DECIMAL TO BINARY

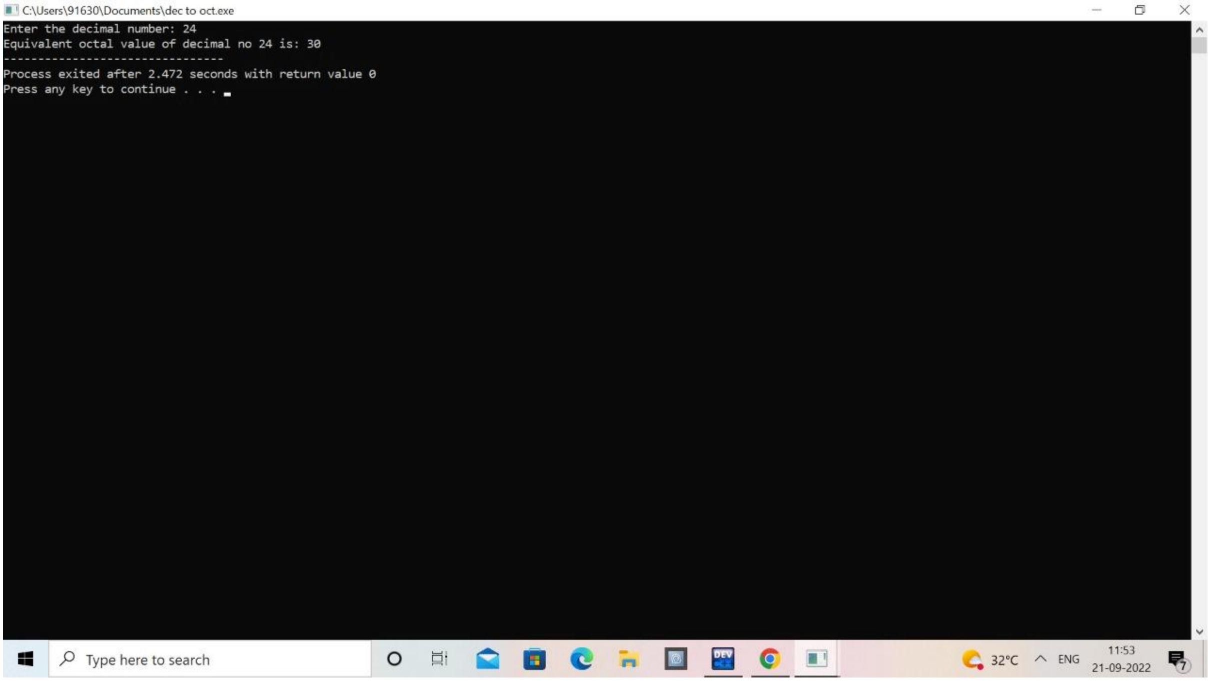
```
#include<stdio.h>
 #include<stdlib.h>
 int main(){
 int a[10],n,i;
 system ("cls");
 printf("Enter the number to convert: ");
 scanf("%d",&n);
 for(i=0;n>0;i++)
 {
 a[i]=n%2;
 n=n/2;
 }
 printf("\nBinary of Given Number is=");
 for(i=i-1;i>=0;i--)
 {
printf("%d",a[i]);
 }
 return 0;
 }
```



```
C:\Users\raush\OneDrive\Documents\12.exe
Enter the Number :23
43


Quoient   : 000000
Remainder : 0010111
--------------------------------
Process exited after 3.474 seconds with return value 0
Press any key to continue . . .
```

EXPERIMENT 22: DECIMAL TO OCTAL NUMBER.

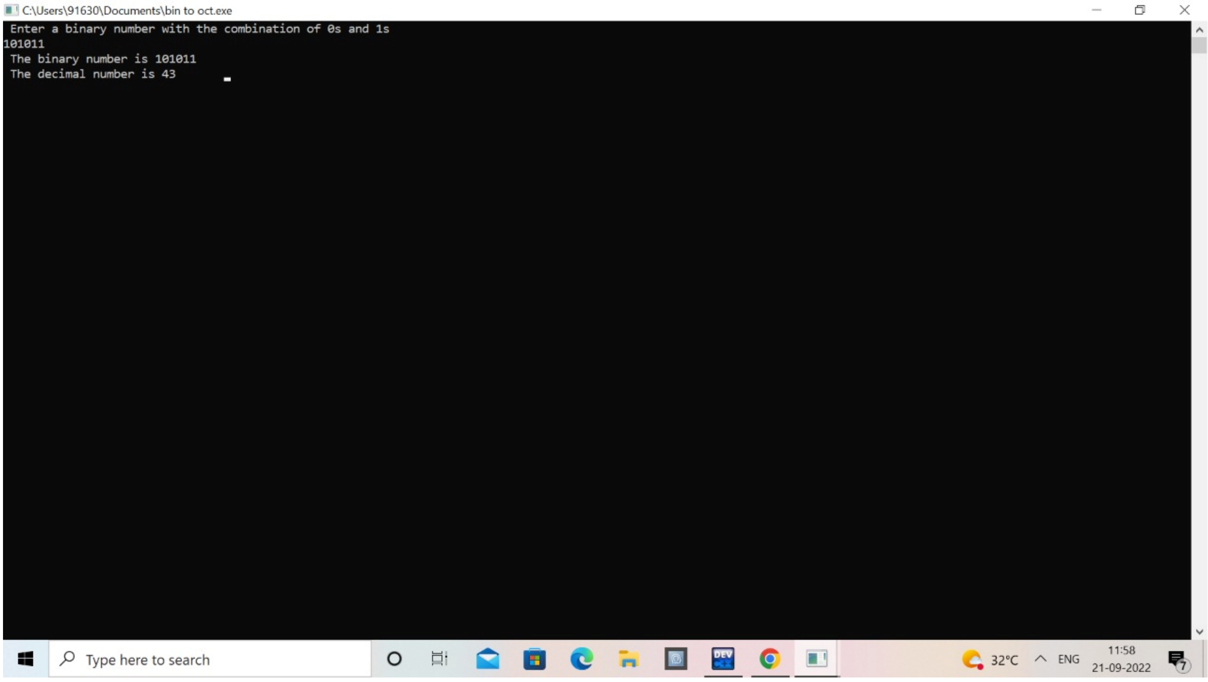
```
#include <stdio.h>
int main()
{
 long decimalnum, remainder, quotient,octalnum=0;
 int octalNumber[100], i = 1, j;
 printf("Enter the decimal number: ");
 scanf("%ld", &decimalnum);
 quotient = decimalnum;
 while (quotient != 0)
 {
 octalNumber[i++] = quotient % 8;
 quotient = quotient / 8;
 }

 for (j = i - 1; j > 0; j--)
 octalnum = octalnum*10 + octalNumber[j];
 printf("Equivalent octal value of decimal no %d is: %d ",
decimalnum,octalnum);
 return 0;
}
```



```
Enter the decimal number: 24
Equivalent octal value of decimal no 24 is: 30
--------------------------------
Process exited after 2.472 seconds with return value 0
Press any key to continue . . .
```

EXPERIMENT 23: BINARY TO DECIMAL NUMBER

```
#include <stdio.h>
int main()
{
 long decimalnum, remainder, quotient,octalnum=0;
 int octalNumber[100], i = 1, j;
 printf("Enter the decimal number: ");
 scanf("%ld", &decimalnum);
 quotient = decimalnum;
 while (quotient != 0)
 {
 octalNumber[i++] = quotient % 8;
 quotient = quotient / 8;
 }

 for (j = i - 1; j > 0; j--)
 octalnum = octalnum*10 + octalNumber[j];
 printf("Equivalent octal value of decimal no %d is: %d ",
decimalnum,octalnum);
 return 0;
}
int convert(long long n) {
 int dec = 0, i = 0, rem;
 while (n!=0) {
 rem = n % 10;
 n /= 10;
 dec += rem * pow(2, i);
 ++i;
 }
 return dec;
}
```

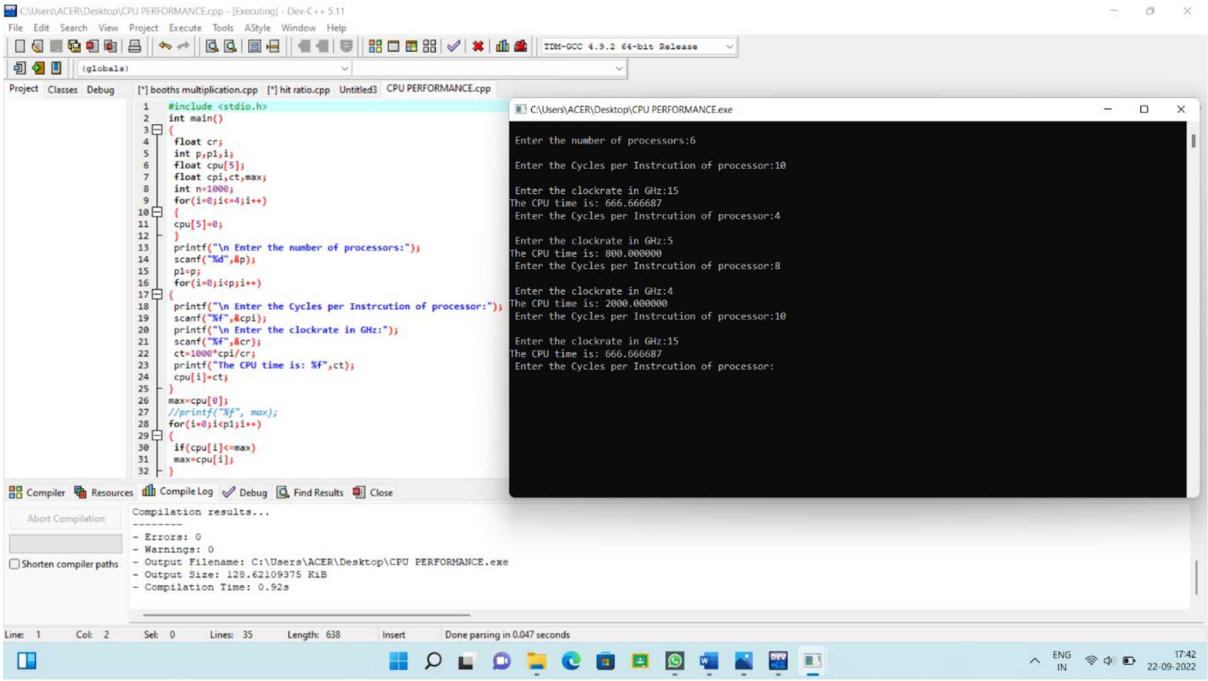


```
 Enter a binary number with the combination of 0s and 1s
101011
 The binary number is 101011
 The decimal number is 43
```

## EXPERIMENT 24: CPU PERFORMANCE

```
#include <stdio.h>

int main()

{

 float cr;

 int p,p1,i;

 float cpu[5];

 float cpi,ct,max;

 int n=1000;

 for(i=0;i<=4;i++)

 {

 cpu[5]=0;

 }

 printf("\n Enter the number of processors:");
 scanf("%d",&p);
 p1=p;
 for(i=0;i<p;i++)
{
 printf("\n Enter the Cycles per Instrcution of processor:");
 scanf("%f",&cpi);
 printf("\n Enter the clockrate in GHz:");
 scanf("%f",&cr);
 ct=1000*cpi/cr;
 printf("The CPU time is: %f",ct);
 cpu[i]=ct;
}
max=cpu[0];
//printf("%f", max);
for(i=0;i<p1;i++)
{
 if(cpu[i]<=max)
 max=cpu[i];
}
printf("\n The processor has lowest Execution time is: %f ",
max);
 return 0;
```

}



C:\Users\ACER\Desktop\CPU PERFORMANCE.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
[*] booths multiplication.cpp
[*] hit ratio.cpp
Untitled3
CPU PERFORMANCE.cpp

```c
#include <stdio.h>
int main()
{
 float cr;
 int p,p1,i;
 float cpu[5];
 float cpi,ct,max;
 int n=1000;
 for(i=0;i<=4;i++)
 {
 cpu[5]=0;
 }
 printf("\n Enter the number of processors:");
 scanf("%d",&p);
 p1=p;
 for(i=0;i<p;i++)
{
 printf("\n Enter the Cycles per Instrcution of processor:");
 scanf("%f",&cpi);
 printf("\n Enter the clockrate in GHz:");
 scanf("%f",&cr);
 ct=1000*cpi/cr;
 printf("The CPU time is: %f",ct);
 cpu[i]=ct;
}
max=cpu[0];
//printf("%f", max);
for(i=0;i<p1;i++)
{
 if(cpu[i]<=max)
 max=cpu[i];
}
```

C:\Users\ACER\Desktop\CPU PERFORMANCE.exe

```
 Enter the number of processors:6

 Enter the Cycles per Instrcution of processor:10

 Enter the clockrate in GHz:15
The CPU time is: 666.666687
 Enter the Cycles per Instrcution of processor:4

 Enter the clockrate in GHz:5
The CPU time is: 800.000000
 Enter the Cycles per Instrcution of processor:8

 Enter the clockrate in GHz:4
The CPU time is: 2000.000000
 Enter the Cycles per Instrcution of processor:10

 Enter the clockrate in GHz:15
The CPU time is: 666.666687
 Enter the Cycles per Instrcution of processor:
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\ACER\Desktop\CPU PERFORMANCE.exe
- Output Size: 128.62109375 KiB
- Compilation Time: 0.92s
```

```
Line: 1   Col: 2   Sel: 0   Lines: 35   Length: 638   Insert   Done parsing in 0.047 seconds
```

# EXPERIMENT 25: SWAPPING TO 8 BIT DATA

LDA 0000H

MOV B,A

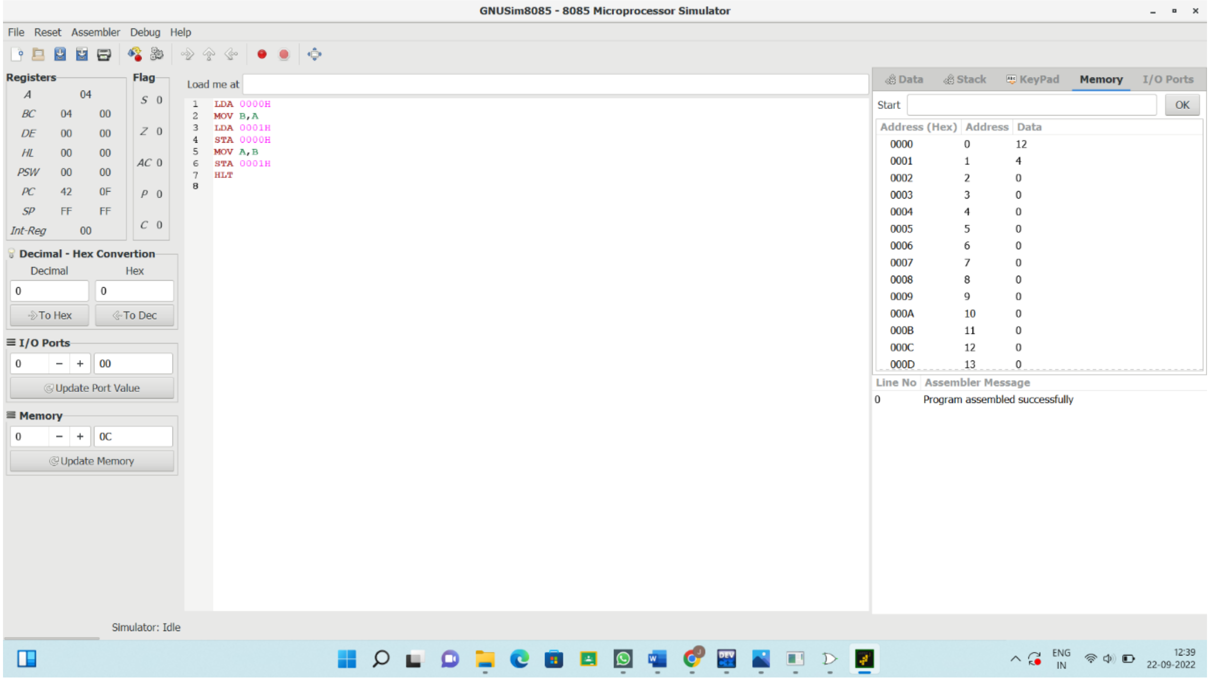
LDA 0001H

STA 0000H

MOV A,B

STA 0001H

HLT



```
LDA 0000H
MOV B,A
LDA 0001H
STA 0000H
MOV A,B
STA 0001H
HLT
```

| Registers | | |
|---|---|---|
| A | 04 | |
| BC | 04 | 00 |
| DE | 00 | 00 |
| HL | 00 | 00 |
| PSW | 00 | 00 |
| PC | 42 | 0F |
| SP | FF | FF |
| Int-Reg | 00 | |

| Flag | |
|---|---|
| S | 0 |
| Z | 0 |
| AC | 0 |
| P | 0 |
| C | 0 |

Decimal: 0 Hex: 0

I/O Ports: 0 − + 00

Memory: 0 − + 0C

| Address (Hex) | Address | Data |
|---|---|---|
| 0000 | 0 | 12 |
| 0001 | 1 | 4 |
| 0002 | 2 | 0 |
| 0003 | 3 | 0 |
| 0004 | 4 | 0 |
| 0005 | 5 | 0 |
| 0006 | 6 | 0 |
| 0007 | 7 | 0 |
| 0008 | 8 | 0 |
| 0009 | 9 | 0 |
| 000A | 10 | 0 |
| 000B | 11 | 0 |
| 000C | 12 | 0 |
| 000D | 13 | 0 |

| Line No | Assembler Message |
|---|---|
| 0 | Program assembled successfully |