**Experiment No. : 3**

|  |  |
| --- | --- |
| **Name:** Sanskar Tewatia | **Date:** 06/02/2021 |
| **Lab partner Name:** Sayantika Mandal | **Course Code:** EED-308 |

# Objective:

The objective of this experiment is to familiarize yourself with the 8051 microcontroller, its simulator software EdSim51DI, and test simple programs on this simulator.

**Part 1** - Write a C program to display the binary patterns from 0 to 255 (and back to 0) on the LEDs interfaced with port 1.

**C Code** –

#include<reg51.h>

void delay ( unsigned char count)

{

int i;

while(count)

{

for(i=0;i<100;i++)

{

}

count--;

}

}

void main()

{

P1=0x00;

while(1)

{

while(P1<0xFF)

{

P1=P1+0x01;

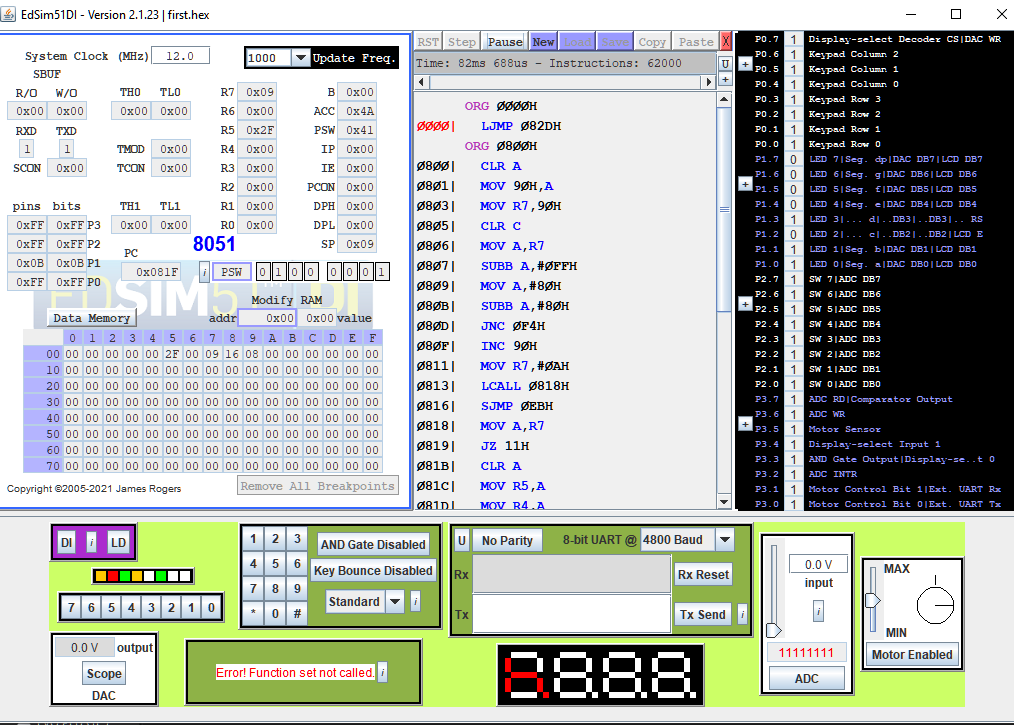
delay(10);

}

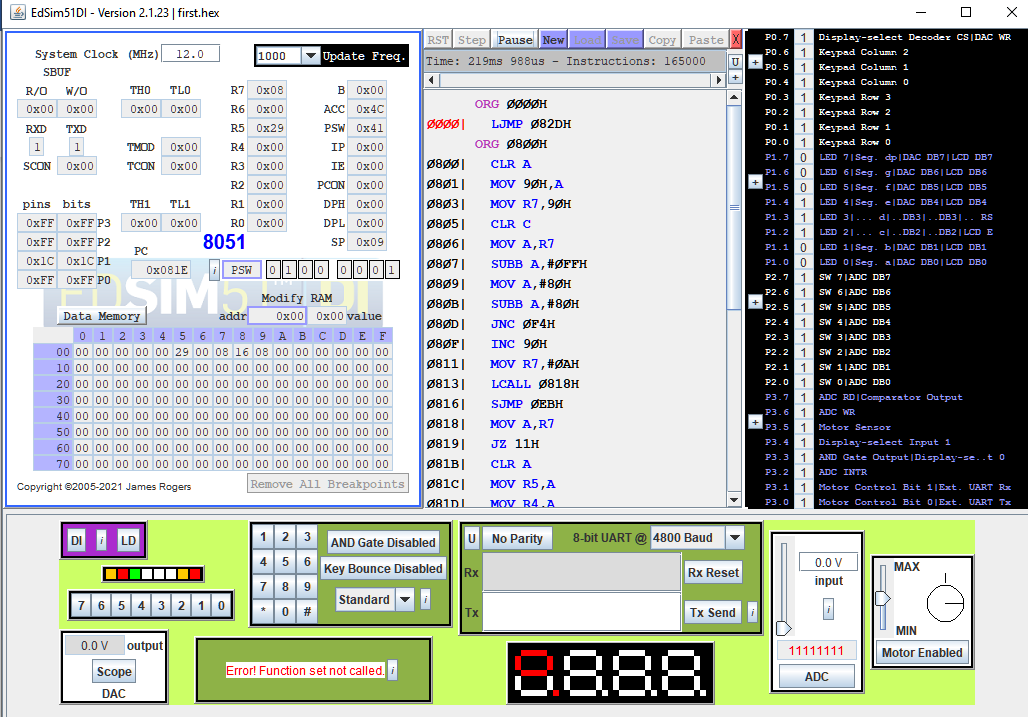
}

}

# Observations: the counting from 0-255 is displayed on the bottom left portion where white color denotes that bit as high. In this C code, a delay function has been made and the actual code inside void main has a variable 0x00 which is equivalent to 0000000000 in binary, a while loop is run to make this run from 0 -255



**Figure 1 - Count at 1011**



**Figure 2 - Count at 11100**

PART 2 - Write a C program to multiplexes the number 1234 on the four 7-segment displays.

**C Code** -

#include<reg51.h>

sbit a0=P3^3;

sbit a1=P3^4;

void delay ( unsigned char count)

{

int i;

while(count)

{

for(i=0;i<100;i++)

{

}

count--;

}

}

void main()

{

while(1)

{

a1=1;

a0=1;

P1=0x67;

delay(10);

a1=1;

a0=0;

P1=0xA4;

delay(10);

a1=0;

a0=1;

P1=0xB0;

delay(10);

a1=0;

a0=0;

P1=0x99;

delay(10);

}

#include<reg51.h>

sbit a=P3^3;

sbit b=P3^4;

void delay ( unsigned char count)

{

int i;

while(count)

{

for(i=0;i<100;i++)

{

}

count--;

}

}

void main()

{

while(1)

{

a=1;

b=1;

P1=0x67;

delay(1000);

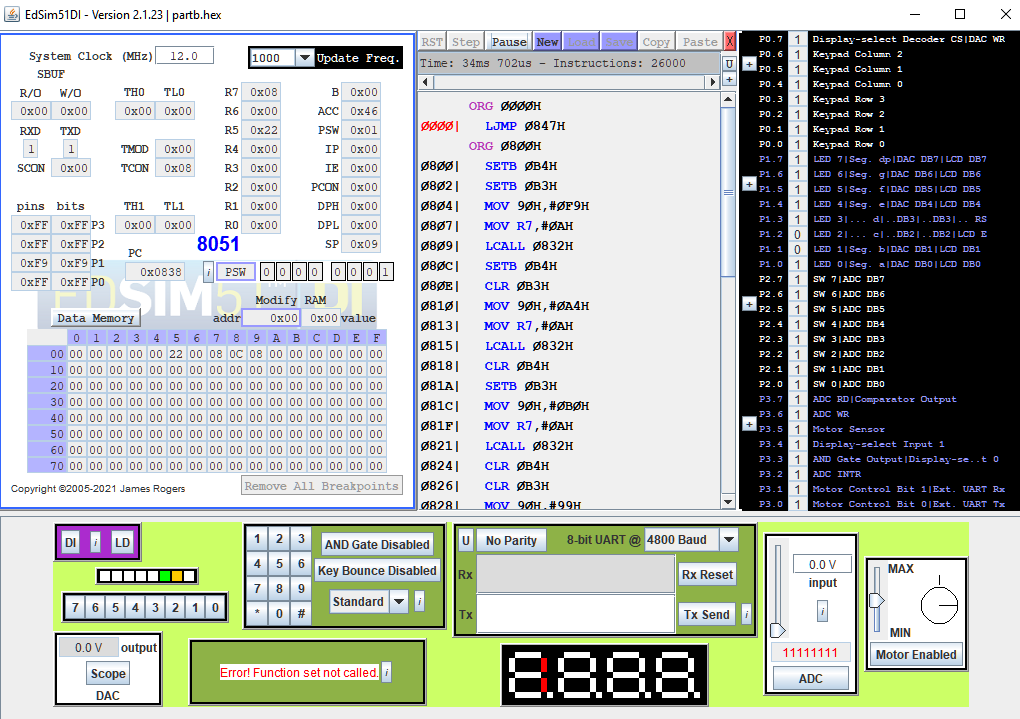
P1=0x3f;

delay(1000);

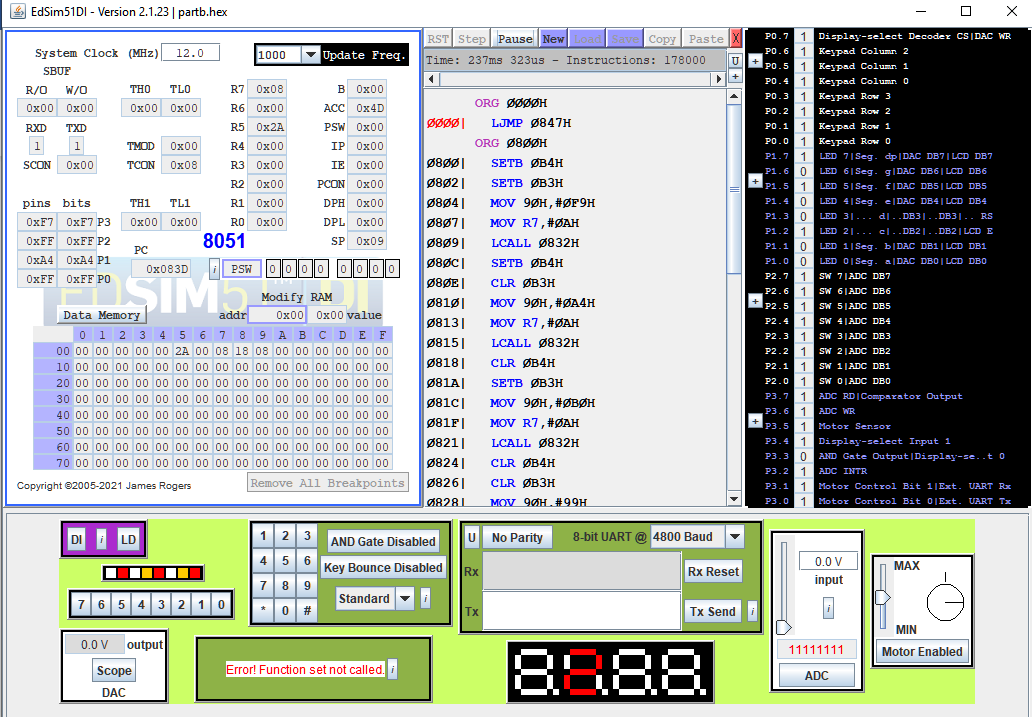
}

}

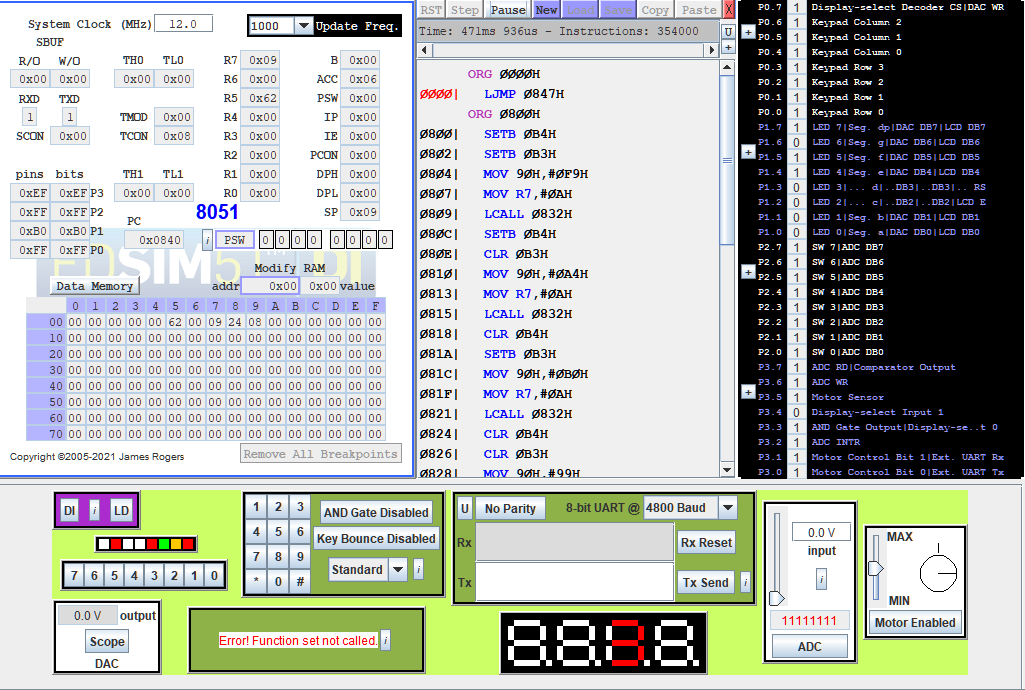
# Observations: the 7 segment displays at the center-bottom part of the screen show the numbers 1,2,3,4 one after the other. In the C Code, there is again a Delay loop, and in the void main code, the variable P1 is set to the values to be displayed(1,2,3,4) and 2 variables a0 and a1 are toggle/select that particular 7 segment display digit with the value set in P1 variable.



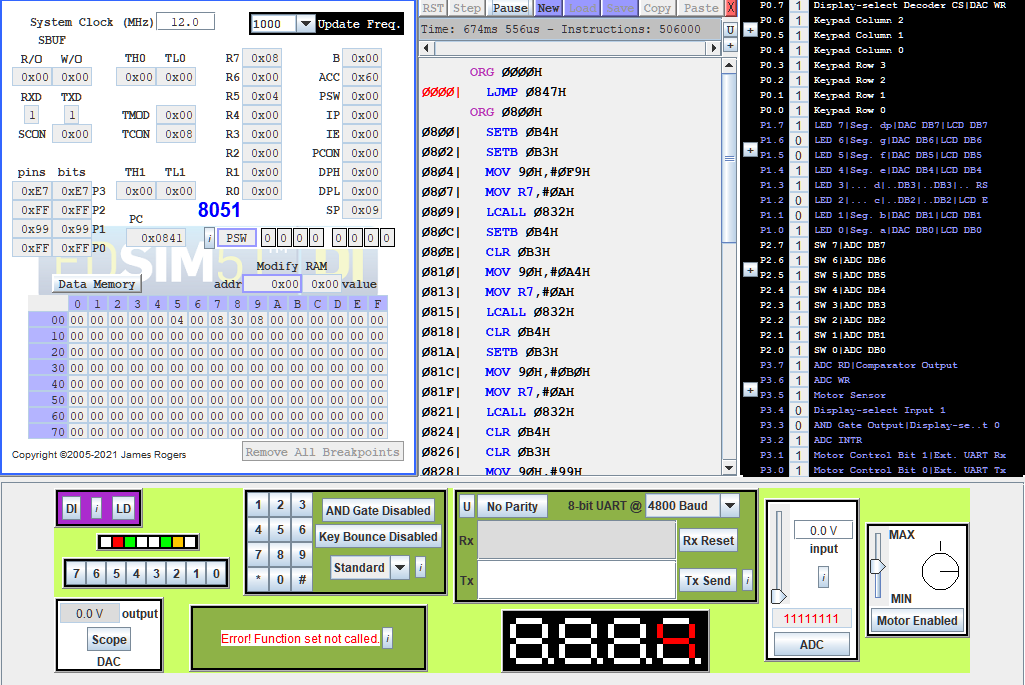
**Figure 3 - Displaying 1**



**Figure 4 - Displaying 2**



**Figure 5 - Displaying 3**



**Figure 6 - Displaying 4**

**PART C** - Write a C program to interface the LCD module and display some messages on that.

**C Code** –

#include<reg51.h>

sbit DB7 = P1^7;

sbit DB6 = P1^6;

sbit DB5 = P1^5;

sbit DB4 = P1^4;

sbit RS = P1^3;

sbit E = P1^2;

sbit clear = P2^4;

sbit ret = P2^5;

sbit left = P2^6;

sbit right = P2^7;

void returnHome(void);

void entryModeSet(bit id, bit s);

void displayOnOffControl(bit display, bit cursor, bit blinking);

void cursorOrDisplayShift(bit sc, bit rl);

void functionSet(void);

void setDdRamAddress(char address);

void sendChar(char c);

void sendString(char\* str);

bit getBit(char c, char bitNumber);

void delay(void);

void main(void) {

functionSet();

entryModeSet(1, 0); // increment and no shift

displayOnOffControl(1, 1, 1); // display on, cursor on and blinking on

sendString("Embedded good");

setDdRamAddress(0x40); // set address to start of second line

sendString("lab over");

// The program can be controlled via some of the switches on port 2.

// If switch 5 is closed the cursor returns home (address 0).

// Otherwise, switches 6 and 7 are read - if both switches are open or both switches

// are closed, the display does not shift.

// If switch 7 is closed, continuously shift left.

// If switch 6 is closed, continuously shift right.

while (1) {

if (ret == 0) {

returnHome();

}

else {

if (left == 0 && right == 1) {

cursorOrDisplayShift(1, 0); // shift display left

}

else if (left == 1 && right == 0) {

cursorOrDisplayShift(1, 1); // shift display right

}

}

}

}

// LCD Module instructions -------------------------------------------

// To understand why the pins are being set to the particular values in the functions

// below, see the instruction set.

// A full explanation of the LCD Module: HD44780.pdf

void returnHome(void) {

RS = 0;

DB7 = 0;

DB6 = 0;

DB5 = 0;

DB4 = 0;

E = 1;

E = 0;

DB5 = 1;

E = 1;

E = 0;

delay();

}

void entryModeSet(bit id, bit s) {

RS = 0;

DB7 = 0;

DB6 = 0;

DB5 = 0;

DB4 = 0;

E = 1;

E = 0;

DB6 = 1;

DB5 = id;

DB4 = s;

E = 1;

E = 0;

delay();

}

void displayOnOffControl(bit display, bit cursor, bit blinking) {

DB7 = 0;

DB6 = 0;

DB5 = 0;

DB4 = 0;

E = 1;

E = 0;

DB7 = 1;

DB6 = display;

DB5 = cursor;

DB4 = blinking;

E = 1;

E = 0;

delay();

}

void cursorOrDisplayShift(bit sc, bit rl) {

RS = 0;

DB7 = 0;

DB6 = 0;

DB5 = 0;

DB4 = 1;

E = 1;

E = 0;

DB7 = sc;

DB6 = rl;

E = 1;

E = 0;

delay();

}

void functionSet(void) {

// The high nibble for the function set is actually sent twice. Why? See 4-bit operation

// on pages 39 and 42 of HD44780.pdf.

DB7 = 0;

DB6 = 0;

DB5 = 1;

DB4 = 0;

RS = 0;

E = 1;

E = 0;

delay();

E = 1;

E = 0;

DB7 = 1;

E = 1;

E = 0;

delay();

}

void setDdRamAddress(char address) {

RS = 0;

DB7 = 1;

DB6 = getBit(address, 6);

DB5 = getBit(address, 5);

DB4 = getBit(address, 4);

E = 1;

E = 0;

DB7 = getBit(address, 3);

DB6 = getBit(address, 2);

DB5 = getBit(address, 1);

DB4 = getBit(address, 0);

E = 1;

E = 0;

delay();

}

void sendChar(char c) {

DB7 = getBit(c, 7);

DB6 = getBit(c, 6);

DB5 = getBit(c, 5);

DB4 = getBit(c, 4);

RS = 1;

E = 1;

E = 0;

DB7 = getBit(c, 3);

DB6 = getBit(c, 2);

DB5 = getBit(c, 1);

DB4 = getBit(c, 0);

E = 1;

E = 0;

delay();

}

// -- End of LCD Module instructions

// --------------------------------------------------------------------

void sendString(char\* str) {

int index = 0;

while (str[index] != 0) {

sendChar(str[index]);

index++;

}

}

bit getBit(char c, char bitNumber) {

return (c >> bitNumber) & 1;

}

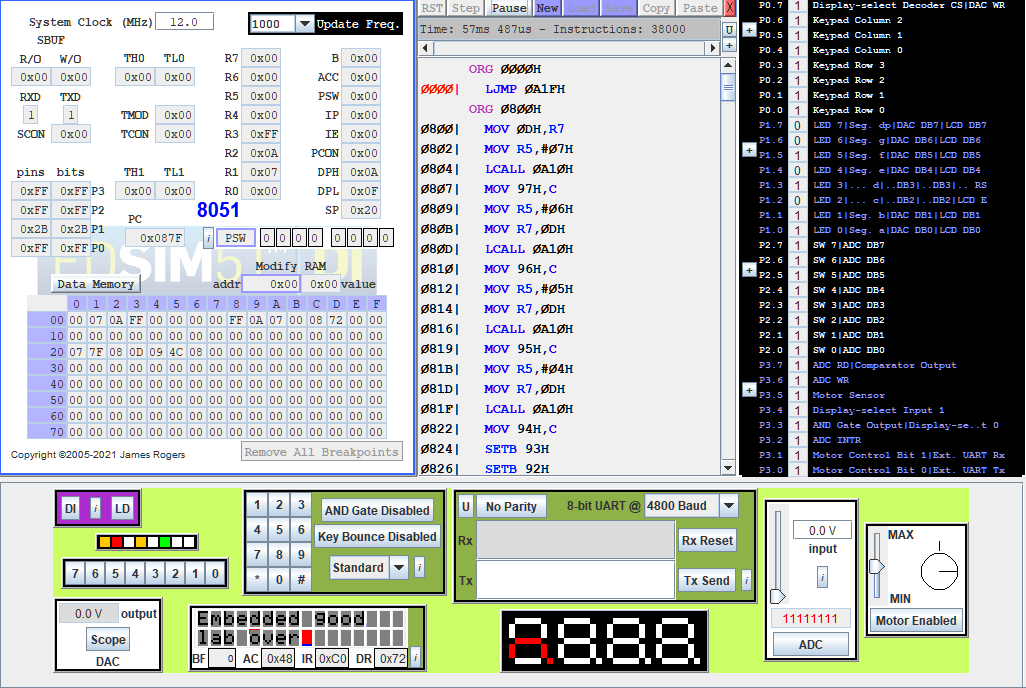
void delay(void) {

char c;

for (c = 0; c < 50; c++);

}

**Observations - From the documentation, the code to display on the LCD is written. It has multiple functions, each having its own specific task. The input string is given in the void main function.**

****

**Figure 7 - Input string is displayed on the LCD at bottom of the screen.**

Results – In this lab, we were asked to write 3 different C codes to interface the 8051 microcontroller with its simulation software EdSim51DI and be able to use 3 of its functions (LCD, 7 Segment and Counters) to display simple outputs understandable by us. The objective was achieved.

MVI R1 00 // position for X

MVI R2 00 // position for Y

MVI R3 80 // address 80 given in question paper

MVI R4 05

NOP

CUD 24 // First loop to reach AB

INC R2

MVS R2

SBA R4

JCD NZ 09

CUD 24 // second loop to reach BC

INC R1

MVS R1

SBA R4

JCD NZ 10

CUD 24 // third loop to reach CD

DCR R2

JCD NZ 17

CUD 24 // last loop to reach DA

DCR R1

JCD NZ 1C

JUD 00

NOP

NOP

MVS R3

STA R1

INC R0

STA R2

INC R0

MVD R3

RTU

MVI R1 00 //[R1]=x

MVI R2 00 //[R2]=y

MVI R3 80 //[R3]=address

MVI R4 04 //[R4]=a

NOP

CUD 20 //storing x,y for AB

INC R2

MVS R2

SBA R4

JCD NZ 09

CUD 20 //storing x,y for BC

INC R1

JCD NZ --

CUD 20 //storing x,y for CD

DCR R2

JCD NZ --

CUD 20 //storing x,y for DA

DCR R1

JCD NZ --

JUD 00

MVI R1 00 //position of x

MVI R2 00 //position of y

MVI R3 80 //address given in question.

MVI R4 03 // value of a

NOP

CUD 24 // loop for AB

INC R2

MVS R2

SBA R4

JCD NZ 09

CUD 24 // loop for BC

INC R1

MVS R1

SBA R4

JCD NZ 10

CUD 24 // loop for CD

DCR R2

JCD NZ 17

CUD 24 //loop for DA

DCR R1

JCD NZ 1C

JUD 00

NOP

NOP

MVS R3

STA R1

INC R0

STA R2

INC R0

MVD R3

RTU