Microprocessor Lab Report

Abhiroop Mukherjee
Enrl. No: 510519109

Department of Computer Science and Technology
Indian Institute of Engineering Science and Technology, Shibpur

Contents

1 Assignment 1

1.1 Objective

Find out the sum of the first 30 natural numbers.

1.2 Tool/Experimental setup considered

- Jubin's 8085 Simulator

1.3 Procedure

We know that

$$1 + 2 + 3 + ... + 29 + 30 = \frac{30 \times 29}{2} = 435 = 01D1H$$

This result is not possible to store in a single register, so we need to use register pair to store the result.

1.4 Program

```
;end result is 465, more than 255 hence we need to do extended additions
       MVI D,1E; setup D, the counter as 30
       MVI C,01; setup BC as 1

L1:    DAD B; Double add BC to HL
       INX B; extended increment BC
       DCR D; decrement D
       JNZ L1; if D becomes 0, Z flag becomes 0 and we break
       HLT
; Ans will be in HL
```

Listing 1: assembly program to find sum of the first 30 natural numbers

1.5 Experimentation

Register	Value	7	6	5	4	3	2	1	0
Accumulator	00	0	0	0	0	0	0	0	0
Register B	00	0	0	0	0	0	0	0	0
Register C	1F	0	0	0	1	1	1	1	1
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	01	0	0	0	0	0	0	0	1
Register L	D1	1	1	0	1	0	0	0	1
Memory(M)	00	0	0	0	0	0	0	0	0

Figure 1: Register configuration after execution (observe HL)

1.6 Conclusion

We see that after the execution of program, the data stored in HL register pair is 01D1H, which is the hexadecimal value of 435.
Hence the Program is working as expected.

2 Assignment 2

2.1 Objective

From an array of 10-byte size integers (unsigned) find out the maximum and minimum.

2.2 Tool/Experimental setup considered

- Jubin's 8085 Simulator

2.3 Procedure

The idea is to linearly iterate through all the values of the arr, and update the register for minimum(C) and maximum(B) values.

2.4 Program

```
;Actual Program
# ORG 5000H
# ARR: DB 5,2,3,4,F,C,7,A,B,1
# ORG 0000
       LXI H,ARR
       MOV B,M  ;B is maximum val
       MOV C,M  ;C is minimum val
       MVI D,0A
;CMP R does (A - R) in background
;If A - R > 0 then Cy = 0, Z = 0
;If A - R = 0 then Cy = 0, Z = 1
;If A - R < 0 then Cy = 1, Z = 1

LP:    MOV A,M
       CMP B
       JC MIN   ;will Jump when Cy = 1, A - B < 0, A < B
       MOV B,A  ;will only happen if A > B

MIN:   CMP C
       JNC SKIP ;will Jump when Cy = 0, A - C > 0, A > C
       MOV C,A  ;will only happen if A < C

SKIP:  INX H
       DCR D
       JNZ LP
       HLT
```

Listing 2: assembly program to find minimum and maximum number in 10-byte unsigned array

2.5 Experimentation

Register	Value	7	6	5	4	3	2	1	0
Accumulator	01	0	0	0	0	0	0	0	1
Register B	0F	0	0	0	0	1	1	1	1
Register C	01	0	0	0	0	0	0	0	1
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	50	0	1	0	1	0	0	0	0
Register L	0A	0	0	0	0	1	0	1	0
Memory(M)	00	0	0	0	0	0	0	0	0

(a) result for {5,2,3,4,F,C,7,A,B,1}

Register	Value	7	6	5	4	3	2	1	0
Accumulator	06	0	0	0	0	0	1	1	0
Register B	0F	0	0	0	0	1	1	1	1
Register C	06	0	0	0	0	0	1	1	0
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	50	0	1	0	1	0	0	0	0
Register L	0A	0	0	0	0	1	0	1	0
Memory(M)	00	0	0	0	0	0	0	0	0

(b) result for {F,E,D,C,B,A,9,8,7,6}

Figure 2: Result for different inputs

2.6 Conclusion

We see that that after program execution, B has the maximum value of array, and C has the minimum value of the array.
Hence the Program is working as expected.

3 Assignment 3

3.1 Objective

Write a routine that produces a delay. The delay value must be passed to register pair DE.

3.2 Tool/Experimental setup considered

- Jubin's 8085 Simulator

3.3 Procedure

Idea is to assign DE a very big value (say FFFF), and decrement it in a loop till DE becomes 0 to produce delay in execution.

3.4 Program

```
       LXI D,FFFF
       CALL DELAY
       HLT
;delay: this subroutine produces delay
;in: value in DE pair
;out: none
;destroys: A

DELAY: DCX D    ;doesn't affect any flags, that's why doing OR
       MOV A,E
       ORA D    ;will give 0 only when both D and E 00
       JNZ DELAY
       RET
```

Listing 3: assembly program to produce delay

3.5 Conclusion

We see that the code runs for sometime, then completes it's execution, signifying that the delay function worked and delayed execution of CPU for some time.
Hence the Program is working as expected.

4 Assignment 4

4.1 Objective

Write a subroutine to move a block of bytes from location X to location Y.
Note that the caller would specify

- X, the source address
- Y, the destination address
- Z, the block size

Note that X, Y and Z are 16-bit quantities.

4.2 Tool/Experimental setup considered

- Jubin's 8085 Simulator

4.3 Procedure

Start reading numbers from location X and save them to location Y, after each iteration, update address of X and Y to next byte. Do this Z times and the whole block is copied.

4.4 Program

```
;setup of data
#ORG 2500
#ARR: DB 4,2,6,7,8

# DESTLOC EQU 4500
#ORG 0000
;let BC = 5, X = 2500, Y = 4500
;hence add data from 2500 to 2504
       LXI B,0005
       LXI D, ARR
       LXI H,DESTLOC
       CALL MOVE
       HLT
;MOVE: move Z number of bytes from loc (X to X+Z) to loc (Y to Y+Z)
;Z store in BC
;X store in DE
;Y store in HL

MOVE:  LDAX D   ;A = Mem[DE]
       MOV M,A  ;Mem[HL] = A
       INX H    ;HL++
       INX D    ;DE++
       DCX B    ;BC--
;DCX doesn't set flags so do manual check by OR
       MOV A,B
       ORA C
       JNZ MOVE
       RET
```

Listing 4: assembly program to move block

4.5 Experimentation

Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
0000	01
0001	05
0003	11
0005	25
0006	21
0008	45
0009	CD
000A	0D
000C	76
000D	1A
000E	77
000F	23
0010	13
0011	0B
0012	78
0013	B1
0014	C2
0015	0D
0017	C9
2500	04
2501	02
2502	06
2503	07
2504	08
4500	04
4501	02
4502	06
4503	07
4504	08
FFFE	0C

(a) result for {4,2,6,7,8}

Memory Editor

Memory Range: 0000 ---- FFFF

Memory Address	Value
0000	01
0001	0A
0003	11
0005	25
0006	21
0008	45
0009	CD
000A	0D
000C	76
000D	1A
000E	77
000F	23
0010	13
0011	0B
0012	78
0013	B1
0014	C2
0015	0D
0017	C9
2500	04
2501	02
2502	06
2503	07
2504	08
2505	01
2506	02
2507	04
2508	05
2509	06
4500	04
4501	02
4502	06
4503	07
4504	08
4505	01
4506	02
4507	04
4508	05
4509	06
FFFE	0C

(b) result for {4,2,6,7,8,1,2,4,5,6}

Figure 3: Result for different inputs

4.6 Conclusion

We see that all the data from location 2500(X) to (2500 + Z) has been copied to location 4500(Y) to (4500 + Z) [Z is 5 in 3a and 10 in 3b].
Hence the Program is working as expected.

5 Assignment 5

5.1 Objective

Write a function isODD(unsigned n) in assembly that takes an unsigned integer (a byte) and determines if it is odd (returns 1) or 0 if it is even.

5.2 Tool/Experimental setup considered

- Jubin's 8085 Simulator

5.3 Procedure

Odd numbers will always be in the form of $2x + 1$, which means that they will have 1 as their LSB. So we just check if $number \wedge 01$ is 1 or not. If the result is 1, then the number is odd, else it is even.

5.4 Program

```
# NUM EQU 49 ;73 in hex
      MVI B,NUM
      CALL ISODD
      HLT
;isODD(n): function which tell if n is odd or even
;in: n = B
;out: ans in C, if n even C = 0,else 1
;destroys: A
;idea: X AND 01 = 0 if X is even, 1 is X is Odd

ISODD: MOV A,B
      ANI 01   ;A = A and 01
      MOV C,A  ;store result in C
      RET
```

Listing 5: assembly program to check if given number isOdd or not

5.5 Experimentation

Register	Value	7	6	5	4	3	2	1	0
Accumulator	01	0	0	0	0	0	0	0	1
Register B	49	0	1	0	0	1	0	0	1
Register C	01	0	0	0	0	0	0	0	1
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	00	0	0	0	0	0	0	0	0
Register L	00	0	0	0	0	0	0	0	0
Memory(M)	06	0	0	0	0	0	1	1	0

(a) result for odd number(73)

Register	Value	7	6	5	4	3	2	1	0
Accumulator	00	0	0	0	0	0	0	0	0
Register B	4A	0	1	0	0	1	0	1	0
Register C	00	0	0	0	0	0	0	0	0
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	00	0	0	0	0	0	0	0	0
Register L	00	0	0	0	0	0	0	0	0
Memory(M)	06	0	0	0	0	0	1	1	0

(b) result for even number(73)

Figure 4: Result for both even and odd numbers

5.6 Conclusion

We see that in case of odd number, C is 1 after execution and in case of even number, C is 0 after execution.
Hence the Program is working as expected.

6 Assignment 6

6.1 Objective

Write a function to add two multi-byte numbers stored in location X and Y. The result is stored in X. Pass a parameter Z indicating the no. of bytes to be added.

6.2 Tool/Experimental setup considered

- Jubin's 8085 Simulator

6.3 Procedure

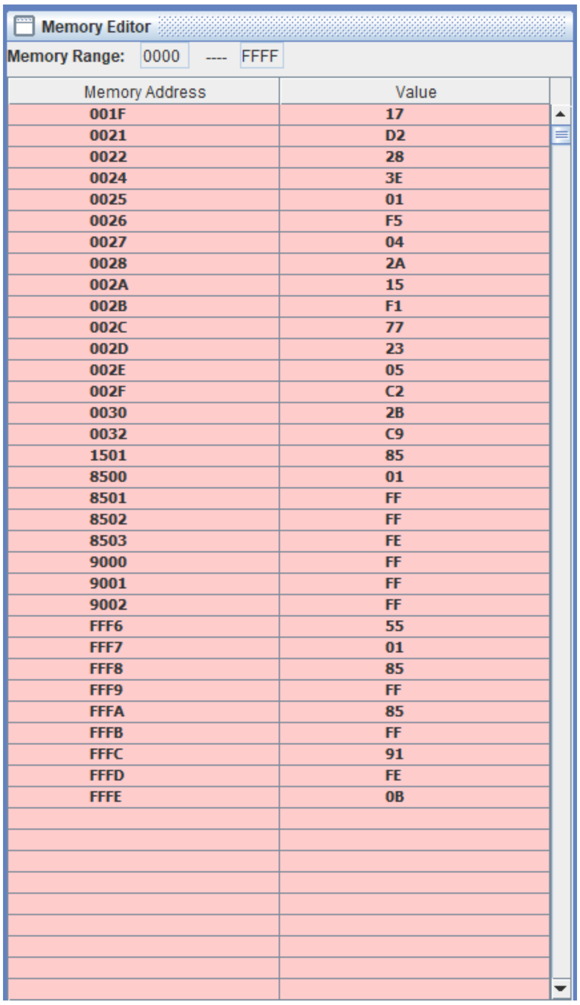
We simulate the default way of adding numbers, we go from right to left, adding (with carry) the numbers and adding it to stack, then we keep popping the elements and save it in X.

6.4 Program

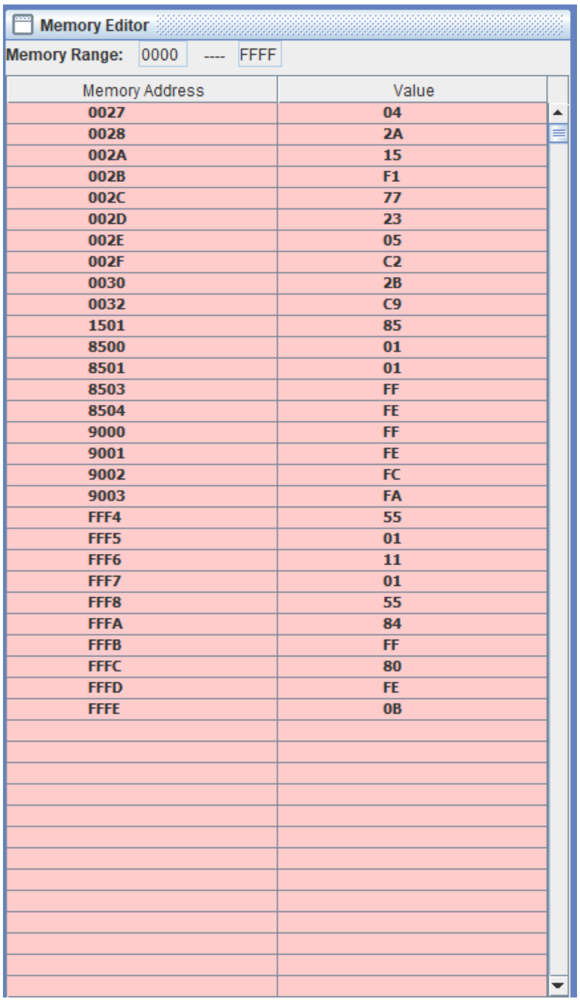
```
# ORG 8500
# NUM1: DB FF,FF,FF
# ORG 9000
# NUM2: DB FF,FF,FF
# NUMBYTE EQU 3
# ORG 0000
       MVI C,NUMBYTE    ;C <- 03
       LXI H,NUM1       ;HL <- 8500
       LXI D,NUM2       ;DE <- 9000
       CALL ADDB
       HLT
;add Z bytes, first number starts from X, other number starts from Y
;Z->C
;X->HL
;Y->DE
;destroys -> B
# SAVELOC EQU 1500
ADDB:  SHLD SAVELOC ;save start of X to 1500 addr to be read later
       MVI B,00     ;set B as 0, will act as counter afterward
;now we need to put HL and DE pair to back of array
       XCHG         ;swap HL and DE
       DAD B        ;HL = HL + BC [with B = 0] [HL now is DE]
       DCX H
       XCHG         ;swap HL and DE again
       DAD B        ;HL = HL + BC
       DCX H
;now both DE and HL are in the end of their array
LOOP:  LDAX D   ;A -> Mem[DE]
       ADC M    ;A -> A+ Mem[HL] + carry
       PUSH PSW ;push AF data to SP, first A, then F
       INR B
       DCX H
       DCX D
       DCR C
       JNZ LOOP ;jump to Loop till C!=0
;now addition done and saved to Stack, need to check if carry exist
       JNC SKIP
;these will execute only when there is a carry
       MVI A,01
       PUSH PSW
       INR B

SKIP:  LHLD SAVELOC   ;read saved LH data to go to start of X

L1:    POP PSW  ;pop SP to AF, first F then A
       MOV M,A
       INX H
       DCR B
       JNZ L1
       RET
```

Listing 6: assembly program to add multi-byte numbers

6.5 Experimentation



(a) result for {FF,FF,FF} + {FF,FF,FF}



(b) result for {01,02,03,04} + {FF,FE,FC,FA}

Figure 5: Result of multi-byte addition (start looking from address 8500)

6.6 Conclusion

We see that the result of multi-byte addition is correct.
Hence the Program is working as expected.

7 Assignment 7

7.1 Objective

Write a fast sub-routine to multiply 9 by 15.

7.2 Tool/Experimental setup considered

- Jubin's 8085 Simulator

7.3 Procedure

We use the Shift-and-Add Multiplication to fast multiply 15 and 9, as register size is 8 bits, we can do this multiplication by using a loop which runs 8 times, i.e O(1)
This method is faster than default loop method, which run in O(min(m, n)), where m and n are the numbers that will be multiplied.

7.4 Program

```
       LXI D,000F   ;15 in hex
       MVI A,09
       LXI H,0000   ;result will be in HL
       MVI B,08     ;8 bit data, 8 rotations to iterate through all the bits of A

LP:    DAD H    ;HL = HL + HL (multiply by 2)(assume left shift)
       RAL      ;rotate A left(<-), leftmost value in C flag
       JNC SKIP
       DAD D    ;if 1 in A's bit, we add D also
SKIP:  DCR B
       JNZ LP
       SHLD 2500
       HLT
;ans in HL register
```

Listing 7: assembly program to fast multiply 15 times 9

7.5 Experimentation

Register	Value	7	6	5	4	3	2	1	0
Accumulator	04	0	0	0	0	0	1	0	0
Register B	00	0	0	0	0	0	0	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	00	0	0	0	0	0	0	0	0
Register E	0F	0	0	0	0	1	1	1	1
Register H	00	0	0	0	0	0	0	0	0
Register L	87	1	0	0	0	0	1	1	1
Memory(M)	00	0	0	0	0	0	0	0	0

Figure 6: Register configuration after execution (look at HL)

7.6 Conclusion

$9 \times 15 = 135 = 0087H$
We see that the result in HL register is same as what we expected.
Hence the Program is working as expected.