- Assignment 2
 - Question 1
 - Objective
 - Tool / Experimental setup considered
 - Procedure
 - Program
 - Experimentation
 - Conclusion
 - Question 2
 - Objective
 - Tool / Experimental setup considered
 - Procedure
 - Program
 - Experimentation
 - Conclusion
 - Question 3
 - Objective
 - Tool / Experimental setup considered
 - Procedure
 - Program
 - Experimentation
 - Conclusion

Assignment 2

Question 1

Objective

Write a subroutine to move a block of bytes from location X to location Y. Note that the caller would specify X and Y; the source and destination along with the block size, say, Z. X, Y and Z are 16-bit quantities.

Tool / Experimental setup considered

• Used Jubin's 8085 Simulator.

Procedure

For this program, I am

- Storing Z in B-C
- Storing X in D-E
- Storing Y in H-L

Now, we will start reading from X and writing to Y, Z times and the entire block will be copied.

Program

```
; Storing Z in B-C
; Storing X in D-E
; Storing Y in H-L
# ORG 3000H
# ARR: DB 1, 2, 3, 4
```

```
# ORG 0000H
LXI B, 0004; count
LXI D, 3000 ; source
LXI H, 4000 ; destination
CALL MOVE
HLT
; Moving 4 bytes from 3000 -> 4000
MOVE:
                                       LDAX D % \left( A\right) =\left( A\right) +\left( 
                                       MOV M, A ; M -> HL
                                       INX H
                                       INX D
                                       DCX B
                                   MOV A, B
                                       ORA C
                                       JNZ MOVE
                                       RET
```

Experimentation

0017	C9
3000	01
3001	02
3002	03
3003	04
4000	01
4001	02
4002	03
4003	04
FFFE	0C

Conclusion

We can see that values from 4000 to 4004 is copied from 3000 to 3004. Here Z is 4, X is 3000, and Y is 4000. Hence, the block of bytes is copied.

Question 2

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.

Tool / Experimental setup considered

• Used Jubin's 8085 Simulator.

Procedure

Odd numbers always have their 0-th bit set. So, we can use the AND instruction to check if the 0-th bit is set.

Program

```
# NUM EQU 7
MVI A, NUM
CALL ISODD
HLT

// Will store 1 in register B if odd, else 0.
ISODD:
   ANI 01  ; AND Immediate with the data and store the result in Acc
```

Experimentation

When the NUM is 7 (Odd).

Register	Value	7	6	5	4	3	2	1	0
Accumulator	01	0	0	0	0	0	0	0	1
Register B	01	0	0	0	0	0	0	0	1
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)	3E	0	0	1	1	1	1	1	0

When the NUM is 8 (Even).

Register	Value	7	6	5	4	3	2	1	0
Accumulator	00	0	0	0	0	0	0	0	0
Register B	0.0	0	0	0	0	0	0	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	0.0	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	0.0	0	0	0	0	0	0	0	0
Register L	00	0	0	0	0	0	0	0	0
Memory(M)	3E	0	0	1	1	1	1	1	0

Conclusion

In case the NUM was odd, the register C was set. And, when the NUM was even, the register C was not set. Hence, the function returns 1 if the NUM was odd and 0 if the NUM was even.

Question 3

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.

Tool / Experimental setup considered

• Used Jubin's 8085 Simulator.

Procedure

Looped through the no. of bytes to be added. Each time we add the content of X and Y and store the result in X. For that we use the ADC which adds the content along with the carry. Finally, we check if there is a carry and if yes, we add 1 to the result.

Note: Here we are assuming the numbers are written in Little-Endian format i.e the most significant bit is stored in the largest address.

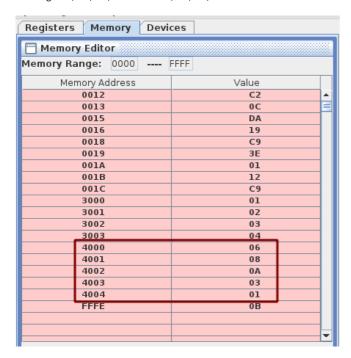
Program

```
# ORG 3000h
# ARR: DB 01,02,03,04
# ORG 4000h
# ARR: DB 05,06,07,FF
# LEN EQU 04
# ORG 0000h
LXI H,3000H; Y
```

```
LXI D,4000H ; X
MVI B, LEN ; Z
CALL SUM
HLT
SUM:
 LDAX D ; copies the contents of that memory location D-E into the accumulator
 ADC M ; ADd with Carry
 STAX D ; contents of the A are copied into the memory location
 INX D
 INX H
 DCR B
  JNZ SUM
  JC CARRY ; If there is some left out carry
CARRY:
 MVI A, 01h
 STAX D
  RET
```

Experimentation

Adding 01,02,03,04 and 05,06,07,FF



Conclusion

Result of adding 01,02,03,04 and 05,06,07,FF is 06,08,0A,03,01 and that is exactly what we got. Hence, the function returns the correct result.