Strathmore University

Bachelor of Science in Informatics and Computer Science (BICS)

ICS 3.1 A

# Lab 1: Introductory Lab to Microprocessors – Intel 8085

Jonyo, Janny – 166885

Mugeni, Amy Zawadi – 167181

Mudibo, Sean Kadida – 168329

Ogutu, Cindy Atieno – 158842

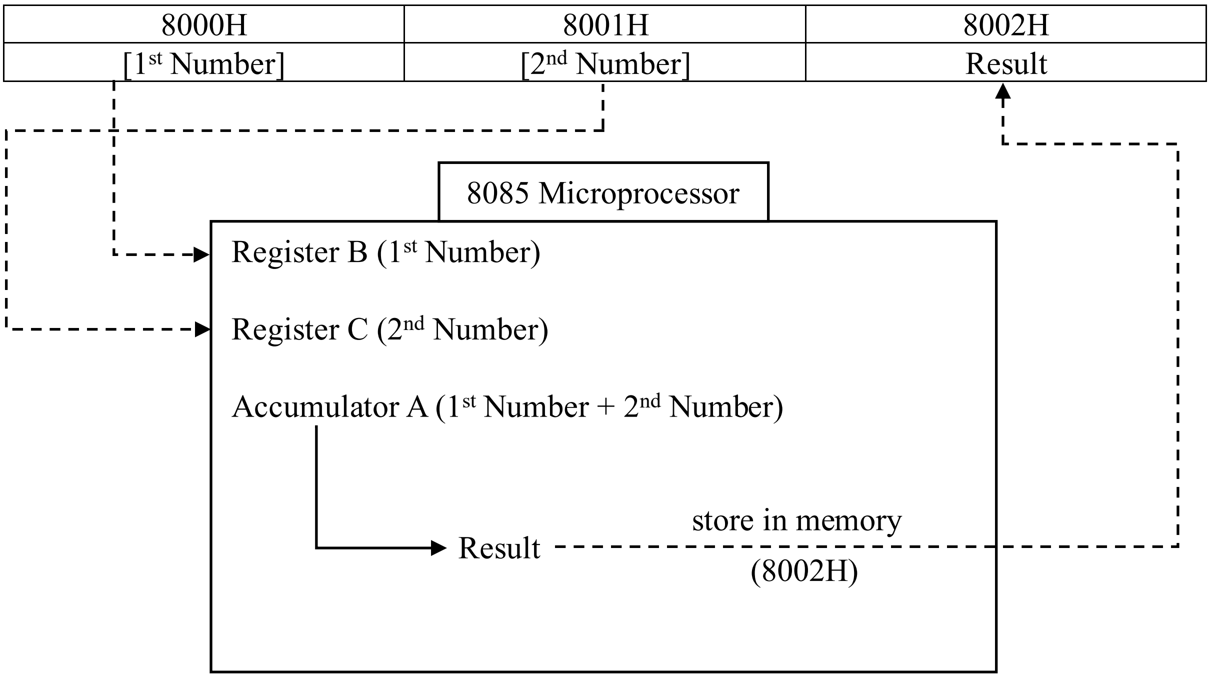
Kamau, Cyprian Njuguna – 155226

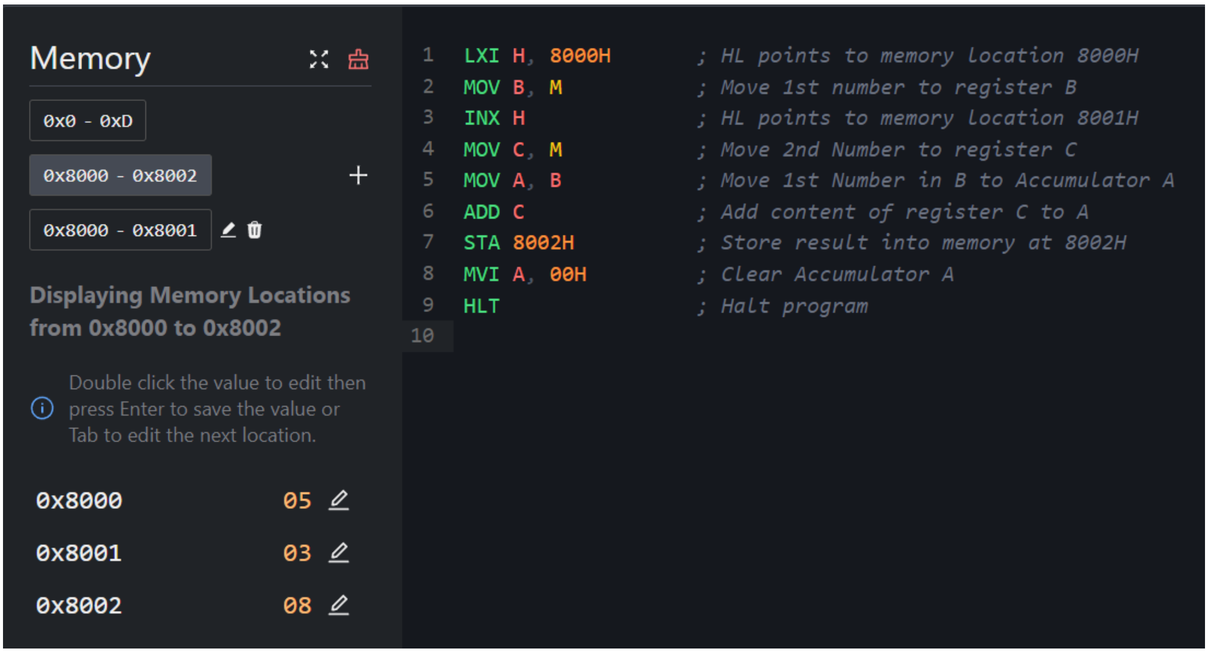
## (a) Arithmetic Operations – Addition and Subtraction

### Addition of two 8-bit numbers

```
LXI H, 8000H          ; HL points to memory location 8000H
MOV B, M              ; Move 1st number to register B
INX H                 ; HL points to memory location 8001H
MOV C, M              ; Move 2nd Number to register C
MOV A, B              ; Move 1st Number in B to Accumulator A
ADD C                 ; Add content of register C to A
STA 8002H             ; Store result into memory at 8002H
MVI A, 00H            ; Clear Accumulator A
HLT                   ; Halt program
```

### Diagram showing data/instruction movement in the microprocessor





Memory

0x0 - 0xD

0x8000 - 0x8002

0x8000 - 0x8001

Displaying Memory Locations from 0x8000 to 0x8002

Double click the value to edit then press Enter to save the value or Tab to edit the next location.

| | |
|---|---|
| 0x8000 | 05 |
| 0x8001 | 03 |
| 0x8002 | 08 |

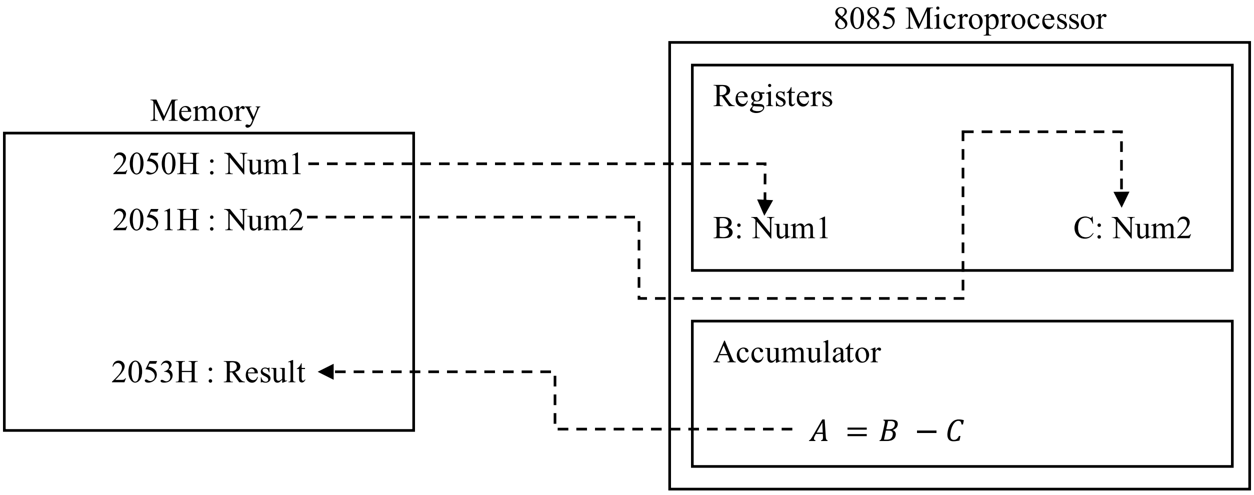
```
LXI H, 8000H        ; HL points to memory location 8000H
MOV B, M            ; Move 1st number to register B
INX H               ; HL points to memory location 8001H
MOV C, M            ; Move 2nd Number to register C
MOV A, B            ; Move 1st Number in B to Accumulator A
ADD C               ; Add content of register C to A
STA 8002H           ; Store result into memory at 8002H
MVI A, 00H          ; Clear Accumulator A
HLT                 ; Halt program
```

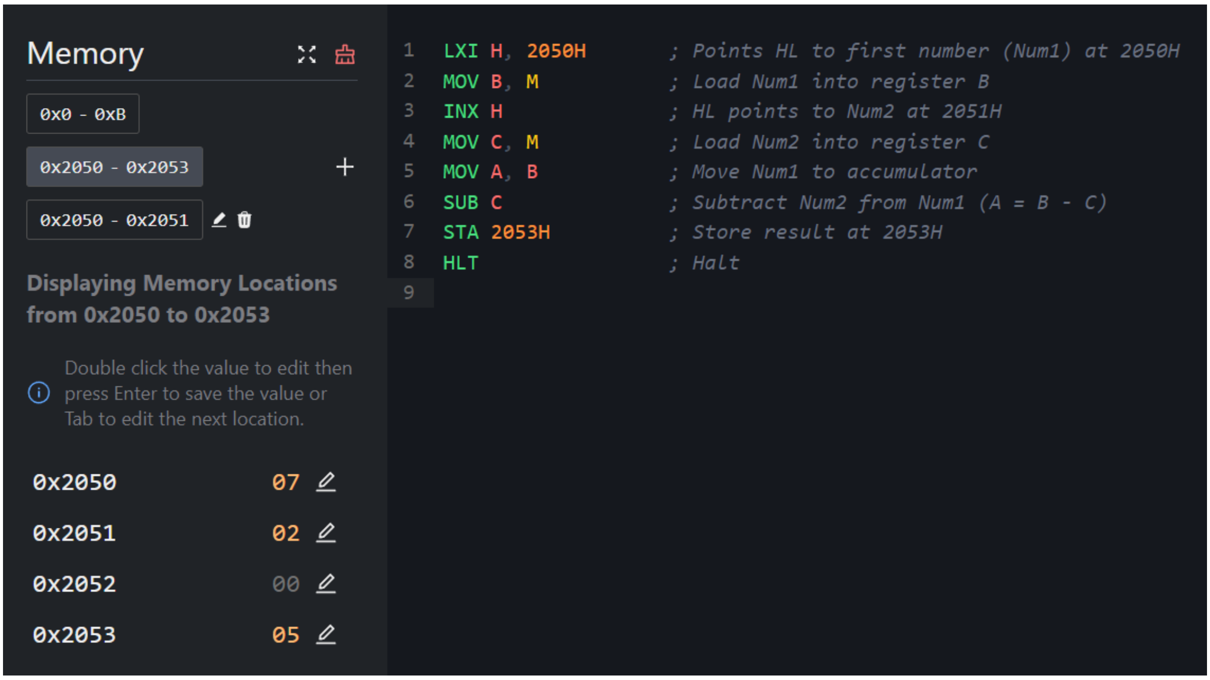
Figure 1: Sim8085 Demonstration of Addition of Two 8-bit Numbers (5 + 3 = 8)

### Subtraction of two 8-bit numbers

```
LXI H, 2050H        ; Points HL to first number (Num1) → 2050H
MOV B, M            ; Load Num1 to register B
INX H               ; Points HL to Num2 → (2051H)
MOV C, M            ; Load Num2 into register C
MOV A, B            ; Move Num1 to Accumulator
SUB C               ; Subtract Num2 from Num1 (A = B − C)
STA 2053H           ; Store result at 2053H
HLT                 ; Halt program
```

### Diagram showing data/instruction movement in the microprocessor





Displaying Memory Locations from 0x2050 to 0x2053

Double click the value to edit then press Enter to save the value or Tab to edit the next location.

| | |
|---|---|
| 0x2050 | 07 |
| 0x2051 | 02 |
| 0x2052 | 00 |
| 0x2053 | 05 |

```
LXI H, 2050H        ; Points HL to first number (Num1) at 2050H
MOV B, M            ; Load Num1 into register B
INX H               ; HL points to Num2 at 2051H
MOV C, M            ; Load Num2 into register C
MOV A, B            ; Move Num1 to accumulator
SUB C               ; Subtract Num2 from Num1 (A = B - C)
STA 2053H           ; Store result at 2053H
HLT                 ; Halt
```

Figure 2: Sim8085 Demonstration of Subtraction of Two 8-bit Numbers (7 − 2 = 5)

### (b) Finding the Largest Number

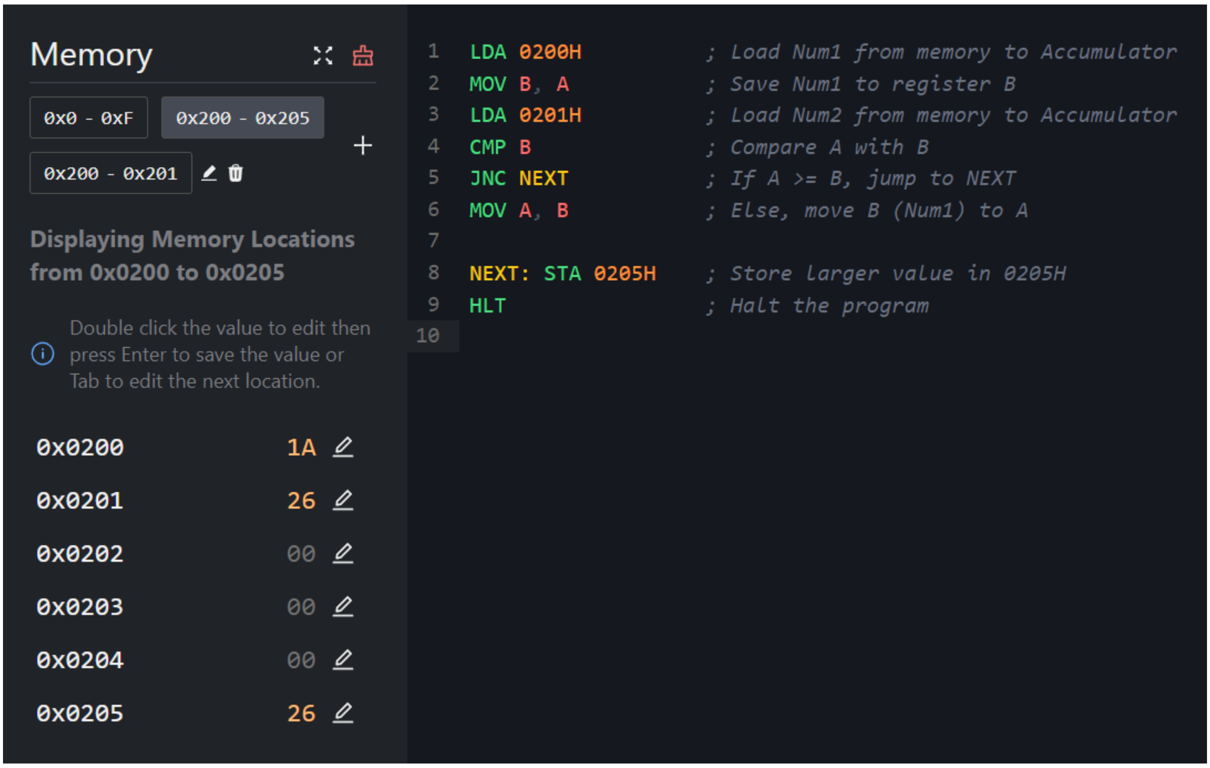
```
LDA 0200H           ; Load Num1 from memory to Accumulator
MOV B, A            ; Save Num1 to register B
LDA 0201H           ; Load Num2 from memory to Accumulator
CMP B               ; Compare A with B
JNC NEXT            ; If A >= B, jump to NEXT
MOV A, B            ; Else, move B (Num1) to A
NEXT: STA 0205H     ; Store Accumulator value in 0205H
HLT                 ; Halt the program
```

### Diagram showing data/instruction movement in the microprocessor

| | | |
|---|---|---|
| Input Data | 26 | 38 |
| Memory Address | <u>200H</u> | <u>201H</u> |

Compare

| | |
|---|---|
| Output data | 38 |
| Memory Address | <u>205H</u> |



Displaying Memory Locations from 0x0200 to 0x0205

Double click the value to edit then press Enter to save the value or Tab to edit the next location.

```
0x0200    1A
0x0201    26
0x0202    00
0x0203    00
0x0204    00
0x0205    26
```

```
LDA 0200H              ; Load Num1 from memory to Accumulator
MOV B, A               ; Save Num1 to register B
LDA 0201H              ; Load Num2 from memory to Accumulator
CMP B                  ; Compare A with B
JNC NEXT               ; If A >= B, jump to NEXT
MOV A, B               ; Else, move B (Num1) to A

NEXT: STA 0205H        ; Store larger value in 0205H
HLT                    ; Halt the program
```

Figure 3: Sim8085 Demonstration of Finding the Largest Number Between Two 8-bit Numbers (26 and 38)

### (c) Bonus Task: Multiplication and Division

Describe how multiplication and division is done in an 8085 microprocessor.

**Multiplication of two 8-bit numbers**

```
; Multiply 5 x 4 using successive addition method
; Result will be in HL and also stored at 8020H (low) and 8021H (high)

        MVI B, 05H          ; Multiplicand = 5
        MVI C, 04H          ; Multiplier = 4
        MVI H, 00H          ; Clear high byte
        MVI L, 00H          ; Clear low byte

LOOP:
        MOV A, C            ; Copy multiplier to A
        CPI 00H             ; Compare with 0
        JZ DONE             ; If zero, end loop
        ; Add multiplicand to result
        MOV A, L            ; Get low byte of result
        ADD B               ; Add multiplicand
        MOV L, A            ; Store back to low byte
        JNC SKIP_INR        ; If no carry, skip high byte increment
        INR H               ; Increment high byte if carry occurred
SKI_INR:
        DCR C               ; Decrement multiplier
        JMP LOOP            ; Repeat
DONE:
        SHLD 8020H          ; Store result in memory at 8020H
        HLT                 ; Stop execution
```

**Multiplication Example**

$5 \times 4$

Multiplicand = 5 (00000101)

Multiplier = 4 (00000100)

Result = 0

Iteration 1:

- Multiplier LSB = 0, don’t add
- Shift multiplicand left : 00001010 (10)
- Shift multiplier right : 00000010 (2)

Iteration 2:

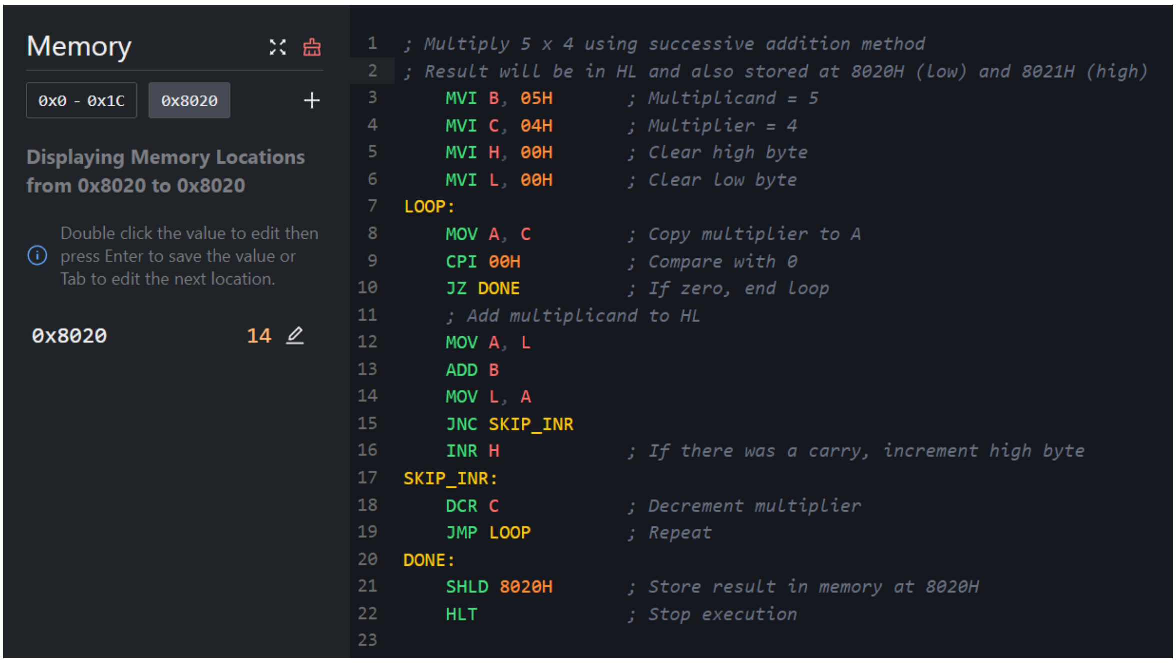
- Multiplier LSB = 0, don’t add
- Shift multiplicand left : 00010100 (20)
- Shift multiplier right: 00000001 (1)

Iteration 3:

- Multiplier LSB = 1, add multiplicand to result : $0 + 20 = 20$
- Shift multiplicand left : 00101000 (40)
- Shift multiplier right : 00000000 (0)

Multiplier is now 0, so we stop.

Result = $20\ (5\ \times 4 = 20)$



Displaying Memory Locations from 0x8020 to 0x8020

Double click the value to edit then press Enter to save the value or Tab to edit the next location.

0x8020 14

```
; Multiply 5 x 4 using successive addition method
; Result will be in HL and also stored at 8020H (low) and 8021H (high)
    MVI B, 05H        ; Multiplicand = 5
    MVI C, 04H        ; Multiplier = 4
    MVI H, 00H        ; Clear high byte
    MVI L, 00H        ; Clear low byte
LOOP:
    MOV A, C          ; Copy multiplier to A
    CPI 00H           ; Compare with 0
    JZ DONE           ; If zero, end loop
    ; Add multiplicand to HL
    MOV A, L
    ADD B
    MOV L, A
    JNC SKIP_INR
    INR H             ; If there was a carry, increment high byte
SKIP_INR:
    DCR C             ; Decrement multiplier
    JMP LOOP          ; Repeat
DONE:
    SHLD 8020H        ; Store result in memory at 8020H
    HLT               ; Stop execution
```

Figure 4: Sim8085 Demonstration of Multiplication of Two 8-bit Numbers

### Division of two 8-bit numbers

The 8085 microprocessors did not have a division operation because there were not enough transistors available on the chip. Nonetheless, division can be achieved using repetitive subtraction.

**Example:** Write an assembly language program to divide two 8-bit numbers and store the result at locations **8020H** and **8021H.** 8020H will store the quotient while 8021H will store the remainder. The data is being saved at location 8000H and 8001H while the result is being stored at location 8050H and 8051H.

### Input

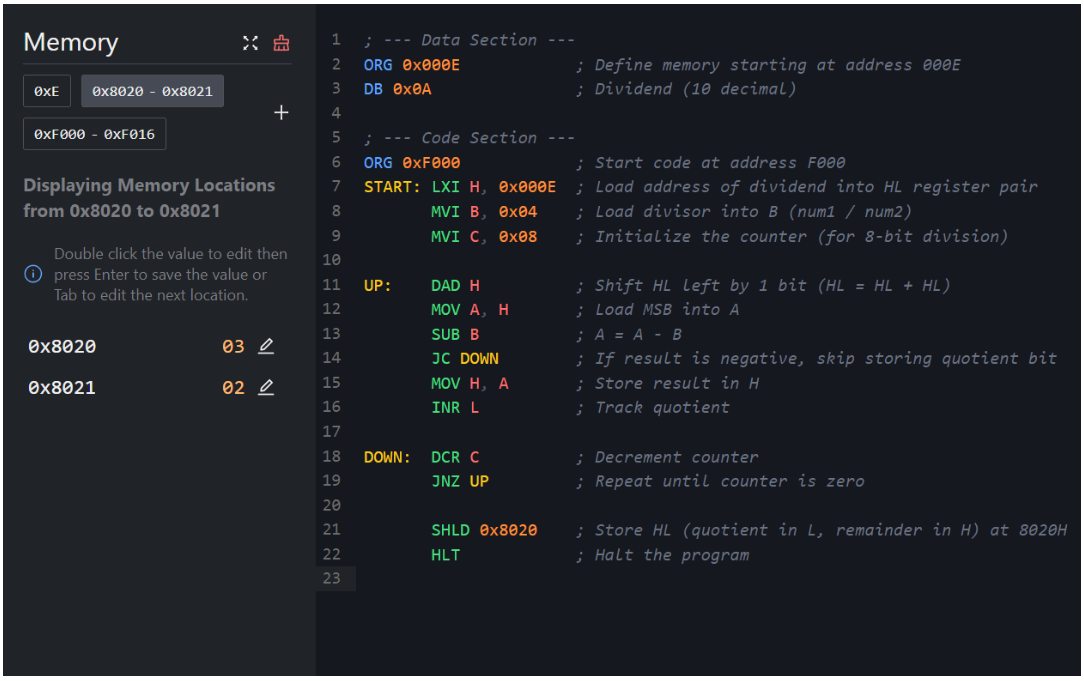
- Dividend: 0EH
- Divisor: 04H
- Quotient will be 3, remainder will be 2

## Program

| Address | HEX Codes | Labels | Mnemonics | Comments |
|---|---|---|---|---|
| F000 | 21, 0E, 00 | START | LXI H, 0CH | Load 8-bit dividend in HL register pair |
| F003 | 06, 04 | | MVI B, 04H | Load divisor in B to perform num1 / num2 |
| F005 | 0E, 08 | | MVI C, 08 | Initialize the counter |
| F007 | 29 | UP | DAD H | Shifting left by 1 bit HL = HL + HL |
| F008 | 7C | | MOV A, H | Load H in A |
| F009 | 90 | | SUB B | perform A = A – B |
| F00A | DA, 0F, F0 | | JC DOWN | If MSB < divisor then shift to left |
| F00D | 67 | | MOV H, A | If MSB > divisor, store the current value of A in H |
| F00E | 2C | | INR L | Tracking quotient |
| F00F | 0D | DOWN | DCR C | Decrement the counter |
| F010 | C2, 07, F0 | | JNZ UP | If not exhausted, then go again |
| F013 | 22, 20, 80 | | SHLD 8020 | Store the result at 8020 H |
| F016 | 76 | | HLT | Stop |

## Output

| Address | Data |
|---|---|
| .<br>. | .<br>. |
| 8020 | 03 |
| 8021 | 02 |
| .<br>. | .<br>. |



Memory

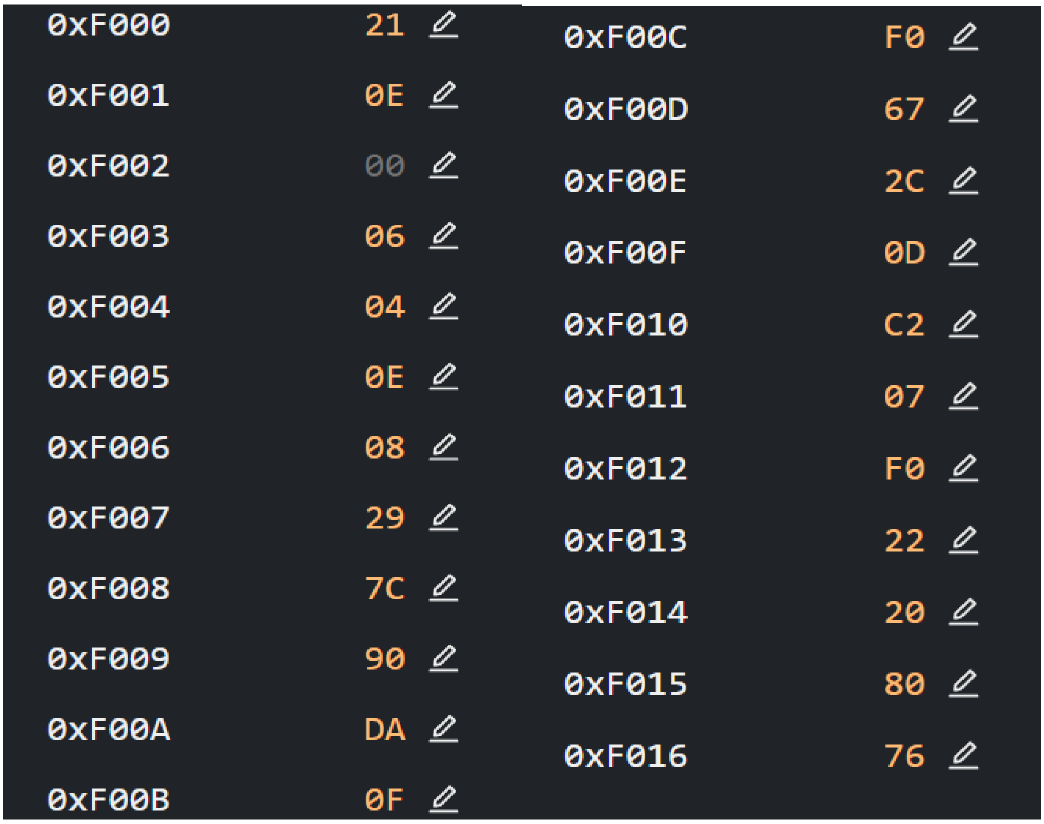
Displaying Memory Locations from 0x8020 to 0x8021

Double click the value to edit then press Enter to save the value or Tab to edit the next location.

```
0x8020        03
0x8021        02
```

```
; --- Data Section ---
ORG 0x000E              ; Define memory starting at address 000E
DB 0x0A                 ; Dividend (10 decimal)

; --- Code Section ---
ORG 0xF000              ; Start code at address F000
START: LXI H, 0x000E    ; Load address of dividend into HL register pair
       MVI B, 0x04      ; Load divisor into B (num1 / num2)
       MVI C, 0x08      ; Initialize the counter (for 8-bit division)

UP:    DAD H            ; Shift HL left by 1 bit (HL = HL + HL)
       MOV A, H         ; Load MSB into A
       SUB B            ; A = A - B
       JC DOWN          ; If result is negative, skip storing quotient bit
       MOV H, A         ; Store result in H
       INR L            ; Track quotient

DOWN:  DCR C            ; Decrement counter
       JNZ UP           ; Repeat until counter is zero

       SHLD 0x8020      ; Store HL (quotient in L, remainder in H) at 8020H
       HLT              ; Halt the program

```

Figure 5: Sim8085 Demonstration of Division of Two 8-bit Numbers (14 ÷ 4 = 3 rem 2)



```
0xF000    21
0xF001    0E
0xF002    00
0xF003    06
0xF004    04
0xF005    0E
0xF006    08
0xF007    29
0xF008    7C
0xF009    90
0xF00A    DA
0xF00B    0F
0xF00C    F0
0xF00D    67
0xF00E    2C
0xF00F    0D
0xF010    C2
0xF011    07
0xF012    F0
0xF013    22
0xF014    20
0xF015    80
0xF016    76
```

Figure 6: Executed Instructions, Including Code for Operations like Loading the Dividend, Performing the Subtraction, and Looping through the Division Steps

## References

Scratch Learners. (2020, July 28). *Addition of two 8-bit numbers in 8085 | Assembly language program | Program for hexadecimal addition* [Video]. YouTube. https://www.youtube.com/watch?v=GLfEQGY70Ek

GeeksforGeeks. (2023, November). *Assembly language programming for beginners: 8-bit addition*. GeeksforGeeks. https://www.geeksforgeeks.org/assembly-language-programming-for-beginners-8-bit-addition/

GeeksforGeeks. (2025, March 27). *8085 program to subtract two 8-bit numbers with or without borrow*. GeeksforGeeks. https://www.geeksforgeeks.org/8085-program-subtract-two-8-bit-numbers-without-borrow/

Techno Tutorials (e-Learning). (2020, December 31). *Unit 2 L 13 | Program for two 8-bit Subtraction in 8085 Microprocessor | 8085 Programming* [Video]. YouTube. https://www.youtube.com/watch?v=Z1ZMJNoxy9O

GeeksforGeeks. (2025, March 27). *8085 program to find larger two 8-bit numbers.* https://www.geeksforgeeks.org/8085-program-find-larger-two-8-bit-numbers/

Intel Corporation. (1981). *MCS-80/85 Family User's Manual. Intel Corporation.* Pages 5-17 to 5-22 cover arithmetic operations and software implementations.

Gaonkar, R. S. (2013). *Microprocessor Architecture, Programming, and Applications with the 8085 (6th ed.).* Penram International Publishing. Chapter 11 includes detailed implementations of arithmetic operations.

https://physicsteacher.in/2022/01/31/multiplication-of-two-8-bit-numbers-using-8085-microprocessor/

Yadav, C. (2020, June). *8085 Program to Divide Two 8-Bit Numbers.* TutorialsPoint. https://www.tutorialspoint.com/8085-program-to-divide-two-8-bit-numbers