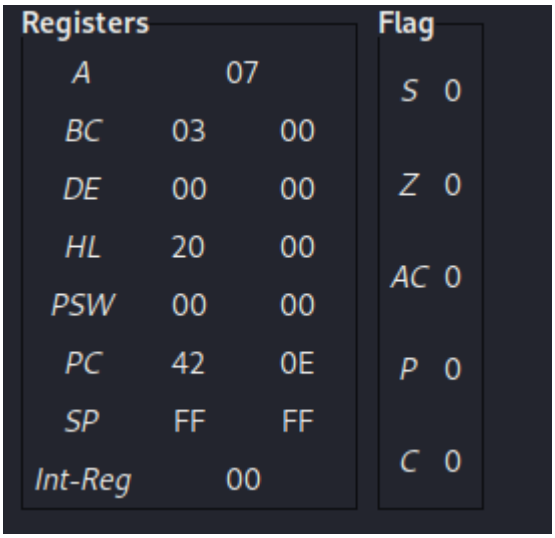
Tavaheed Tariq

2022BITE008

Lab File for Microprocessor 8085

Addition of two numbers using registers

```
;Add Two 8 bit numbers using registers

jmp start

;code
start: nop
MVI A, 04H
MVI B, 03H
ADD B
LXI H, 2000H
MOV M, A
hlt
```



Addition of two numbers using immediate values

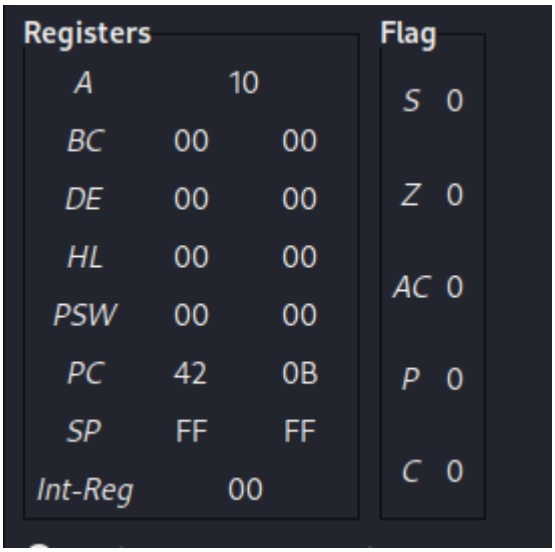
```
;Add Two 8 bit numbers using add immediate

jmp start

;code
start: nop
SUB A
ADI 05H       ;add 05H to accumulator
ADI 04H       ;add 04H to accumulator
hlt
```

Subtraction of two numbers using registers

```
jmp start
;code
start: nop
MVI A, 07H            ; Load 07H into accumulator
MVI B, 03H            ; Load 03H into register B
SUB B                 ; Subtract B from A
LXI H, 2000H          ; Load address 2000H into H-L pair
MOV M, A              ; Store result at memory location 2000H
hlt
```

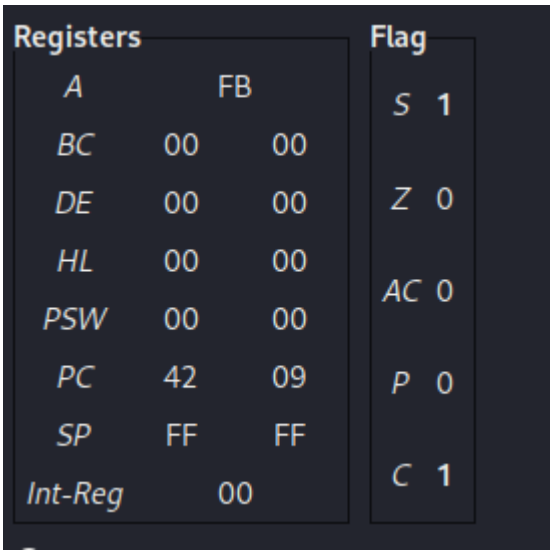
Subtraction of two numbers using immediate values

```
jmp start
;code
start: nop
MVI A, 25H    ; Load 25H into accumulator
SUI 10H       ; Subtract 10H from accumulator
SUI 05H       ; Subtract 05H from accumulator
hlt
```



Subtraction of two numbers using immediate values (producing a negative result)

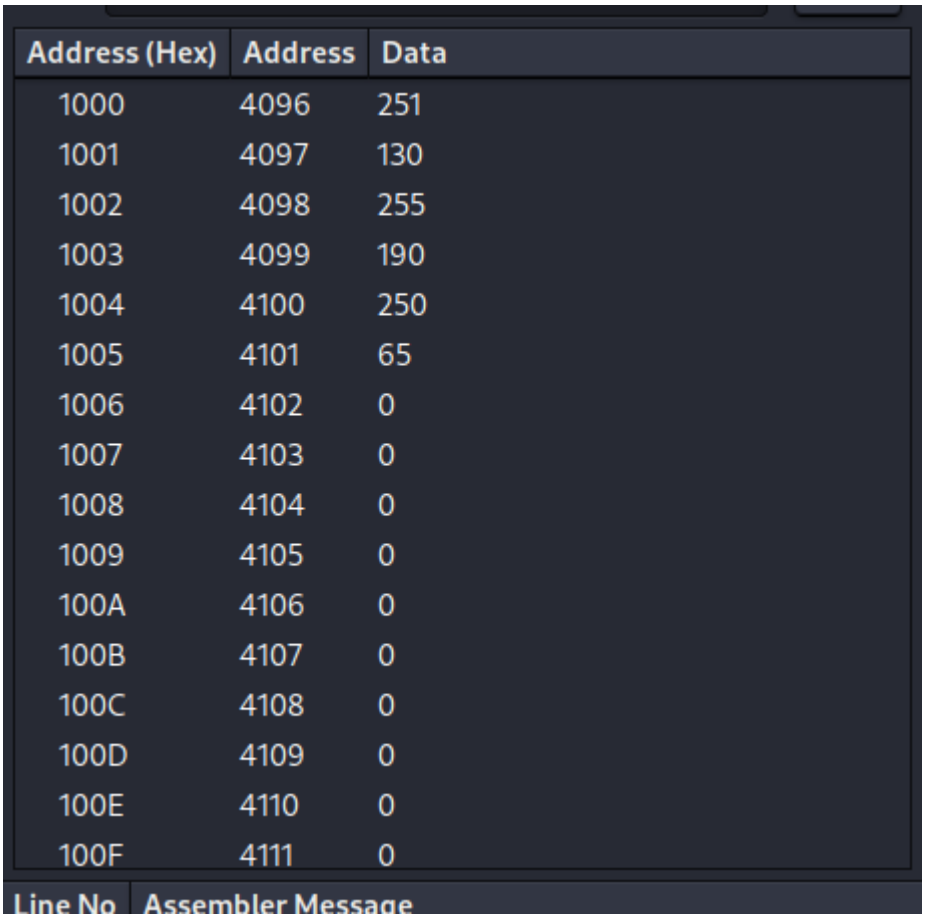
```
jmp start
;code
start:  nop
        MVI A, 05H    ; Load 05H into accumulator (first number)
        SUI 0AH       ; Subtract 0AH from accumulator (second number)
        hlt
```



Addition of two 16 bit numbers

```
; Add Two 16-bit numbers using registers
jmp start


start: nop
;load 16 bit numbers from memory
;as 8085 uses little endian hence it will store first 8 bits of lsb then 8
bits of msb
LXI H, 1000H
MOV C, M ; lsb 8 bits of 1st number
INX H  ;1001H
MOV B, M  ;msb 8 bits of 1st number
INX H
MOV E, M  ;lsb 8 bits of number 2
INX H
MOV D, M  ;msb 8 bits of number 2

; Add lower bytes
MOV A, C
ADD E
MOV C, A ;lower byte of added number is stored in C register

; Add upper bytes with carry
MOV A, B
ADC D
MOV B, A ;upper byte of added number is stored in B register

; Store the result in little endian format
INX H
```

```
MOV M, C  ; store lower byte
INX H
MOV M, B  ; sotre upper byte
HLT
```



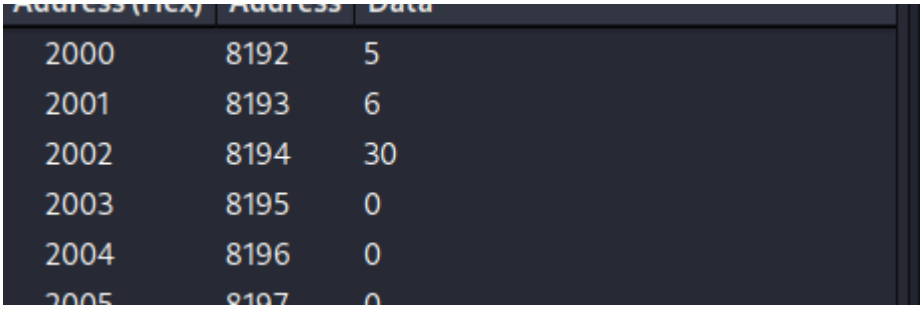
Address (Hex)	Address	Data
1000	4096	251
1001	4097	130
1002	4098	255
1003	4099	190
1004	4100	250
1005	4101	65
1006	4102	0
1007	4103	0
1008	4104	0
1009	4105	0
100A	4106	0
100B	4107	0
100C	4108	0
100D	4109	0
100E	4110	0
100F	4111	0

Line No | Assembler Message

Multiplication of two 8-bit numbers

```
jmp start
;code
start: nop
        LXI H, 2000H    ; Load address 2000H into HL
        MOV B, M        ; Load first number into B
        INX H           ; Point to 2001H
        MOV C, M        ; Load second number into C
        MVI A, 00H      ; Clear A (for result)
        MVI D, 00H      ; Clear D (for carry)

MULT:   ADD B           ; Add B to A
        JNC NOCARRY     ; Jump if no carry
        INR D           ; Increment D if there was a carry
NOCARRY: DCR C           ; Decrement C
        JNZ MULT        ; If C not zero, continue multiplication
```

```
        INX H           ; Point to 2002H
        MOV M, A        ; Store lower byte of result
        INX H           ; Point to 2003H
        MOV M, D        ; Store upper byte of result
        HLT             ; Halt program
```



Address (Hex)	Address	Data
2000	8192	5
2001	8193	6
2002	8194	30
2003	8195	0
2004	8196	0
2005	8197	0

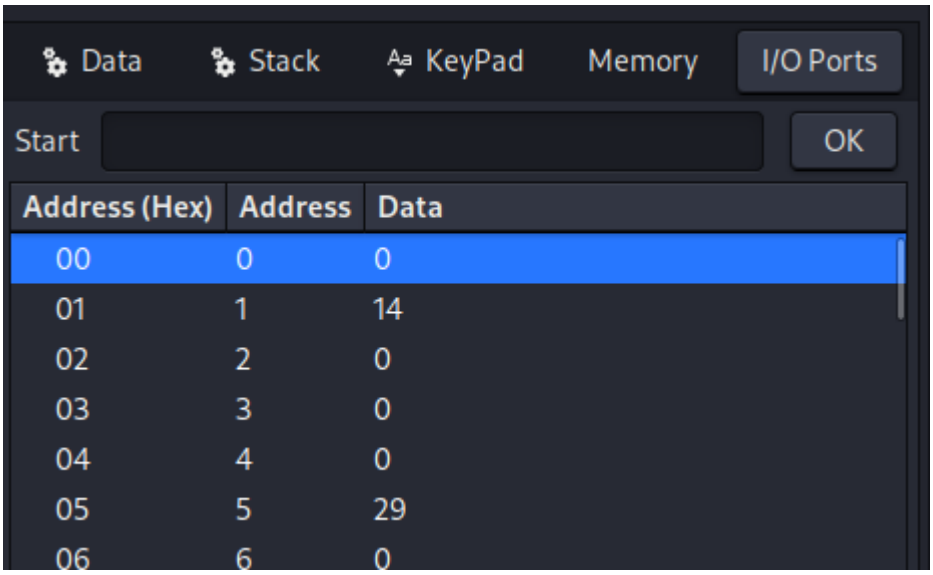
Division of two 8-bit numbers

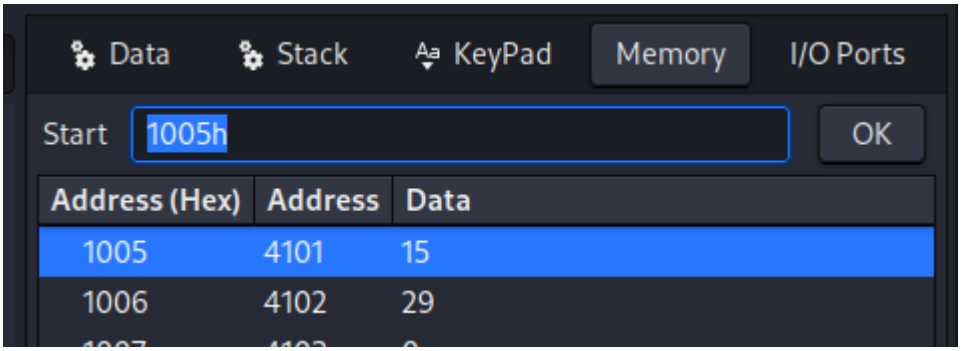
```
jmp start
; code
start: nop
       LXI H, 2000H    ; Load address 2000H into HL
       MOV B, M        ; Load first number (dividend) into B
       INX H           ; Point to 2001H
       MOV C, M        ; Load second number (divisor) into C
       MVI D, 00H      ; Clear D (for quotient)

DIVIDE: MOV A, B        ; Load dividend into A
        SUB C           ; Subtract divisor from A
        JC FINISH       ; Jump to FINISH if result is negative
        INR D           ; Increment quotient
        MOV B, A        ; Store result back in B
        JMP DIVIDE      ; Continue division

FINISH: MOV A, B        ; Load final remainder into A
        INX H           ; Point to 2002H
        MOV M, A        ; Store remainder
        INX H           ; Point to 2003H
        MOV M, D        ; Store quotient
        HLT             ; Halt program
```

Address (Hex)	Address	Data
2000	8192	30
2001	8193	4
2002	8194	2
2003	8195	7

Write a program to add the first number from memory location 1006H and second number from input port 01H, store the result at 1008H, the carry (if any) at 1009H , and display result at output port 06H

```
jmp start

start:  NOP
LXI H, 1006H
MOV B, M
IN 01H    ;stores input of port 01 to accumulator
ADD B
OUT 06H    ;stores value present in accumulator to 06 port
MOV C, A
JNC store_sum    ; store the result if no carry is generated
MVI A, 01H      ; If carry is generated, move 1 into accumulator A
STA 1009H       ; Store the carry at memory location 1007H

store_sum: DCR H
MOV M, C
HLT
```





Program to Subtract a Number from Input Port 01H from the Number at Memory Location 1006H

Mnemonics	Hex-codes	Address
LXI H, 1006H	21 06 10	3000 3001 3002
MOV A, M	7E	3003
IN 01H	DB 01	3004 3005
SUB A	97	3006
JNC NoBorrow	D2 0C 30	3007 3008 3009
STC	37	300A
NoBorrow: LXI H, 1008H	21 08	300B 300C

Mnemonics	Hex-codes	Address
	10	300D
MOV M, A	77	300E
OUT 06H	D3 06	300F 3010
HLT	76	3011

Find the largest in an array

Mnemonics	Hex-codes	Address
LXI H	21 00 20	3000 3001 3002
MOV C, M	4E	3003
INX H	23	3004
MOV A, M	7E	3005
DCR C	0D	3006
Loop: INX H	23	3007
MOV B, M	46	3008
CMP B	B8	3009
JNC Next	D2 11 20	300A 300B 300C
MOV A, B	78	300D
Next: DCR C	0D	2011
JNZ Loop	C2 07 30	2012 2013 2014
STA 2050H	32 50 20	2100 2101 2102
HLT	76	2103

Find the Smallest Number in an Array

Mnemonics	Hex-codes	Address
LXI H	21 00 20	3000 3001 3002
MOV C, M	4E	3003
INX H	23	3004
MOV A, M	7E	3005
DCR C	0D	3006
Loop: INX H	23	3007
MOV B, M	46	3008
CMP B	B8	3009
JC Next	DA 11 20	300A 300B 300C
MOV A, B	78	300D
Next: DCR C	0D	2011
JNZ Loop	C2 07 30	2012 2013 2014
STA 2050H	32 50 20	2100 2101 2102
HLT	76	2103

Program to Calculate the Sum of Numbers in an Array and Handle Carry

Mnemonics	Hex-codes	Address
LXI H	21 00 20	3000 3001 3002
MOV C, M	4E	3003
INX H	23	3004
MVI D, 00	16 00	3005 3006

Mnemonics	Hex-codes	Address
MVI E, 00	1E 00	3007 3008
Loop: MOV A, M	7E	3009
ADD L	85	300A
JNC Next	D2 0E 30	300B 300C 300D
INX D	13	300E
Next: MOV E, A	5F	300F
INX H	23	3010
DCR C	0D	3011
JNZ Loop	C2 09 30	3012 3013 3014
MOV A, D	7A	3015
STA 2050H	32 50 20	3016 3017 3018
MOV A, E	7B	3019
STA 2051H	32 51 20	301A 301B 301C
HLT	76	301D

Program to Sort an Array in Ascending Order

Mnemonics	Hex-codes	Address
LXI H	21 00 20	3000 3001 3002
MOV C, M	4E	3003
DCR C	0D	3004
MOV B, C	41	3005
INX H	23	3006

Mnemonics	Hex-codes	Address
Loop1: MOV D, C	4A	3007
LXI H, 2001H	21 01 20	3008 3009 300A
Loop2: MOV A, M	7E	300B
INX H	23	300C
CMP M	BE	300D
JC Skip	DA 13 30	300E 300F 3010
XCHG	EB	3011
MOV M, A	77	3012
Skip: DCR D	15	3013
JNZ Loop2	C2 0B 30	3014 3015 3016
DCR C	0D	3017
JNZ Loop1	C2 07 30	3018 3019 301A
HLT	76	301B

Program to Implement Bubble Sort in Descending Order

Mnemonics	Hex-codes	Address
LXI H	21 00 20	3000 3001 3002
MOV C, M	4E	3003
DCR C	0D	3004
MOV B, C	41	3005
INX H	23	3006
OuterLoop: MOV D, C	4A	3007

Mnemonics	Hex-codes	Address
LXI H, 2001H	21 01 20	3008 3009 300A
InnerLoop: MOV A, M	7E	300B
INX H	23	300C
CMP M	BE	300D
JNC Skip	D2 13 30	300E 300F 3010
MOV E, M	5E	3011
MOV M, A	77	3012
DCX H	2B	3013
MOV M, E	73	3014
Skip: DCR D	15	3015
JNZ InnerLoop	C2 0B 30	3016 3017 3018
DCR C	0D	3019
JNZ OuterLoop	C2 07 30	301A 301B 301C
HLT	76	301D

Program to Find the Factorial of a Number

Mnemonics	Hex-codes	Address
LXI H	21 00 20	3000 3001 3002
MOV B, M	46	3003
MVI C, 01	0E 01	3004 3005
MOV A, B	78	3006
DCR B	05	3007

Mnemonics	Hex-codes	Address
Loop: MOV D, B	42	3008
MOV E, C	4B	3009
CALL Multiply	CD 10 30	300A 300B 300C
MOV C, L	4D	300D
MOV B, D	43	300E
DCR B	05	300F
JNZ Loop	C2 08 30	3010 3011 3012
STA 2050H	32 50 20	3013 3014 3015
HLT	76	3016

Subroutine for Multiplication

Mnemonics	Hex-codes	Address
Multiply: MOV A, D	7A	3010
MVI H, 00	26 00	3011 3012
MOV L, E	6B	3013
CALL Add	CD 18 30	3014 3015 3016
DCR D	15	3017
JNZ Multiply	C2 10 30	3018 3019 301A
RET	C9	301B

Subroutine for Addition

Mnemonics	Hex-codes	Address
Add: MOV A, L	7E	301C
ADD H	84	301D
MOV L, A	6F	301E
RET	C9	301F

Program to Search for a Number in an Array

Mnemonics	Hex-codes	Address
LXI H	21 00 20	3000 3001 3002
MOV C, M	4E	3003
INX H	23	3004
MOV A, M	7E	3005
INX H	23	3006
Loop: CMP M	BE	3007
JZ Found	CA 0E 30	3008 3009 300A
INX H	23	300B
DCR C	0D	300C
JNZ Loop	C2 07 30	300D 300E 300F
MVI A, 00	3E 00	3010 3011
STA 2050H	32 50 20	3012 3013 3014
HLT	76	3015
Found: MVI A, 01	3E 01	3016 3017
STA 2050H	32 50	3018 3019

Mnemonics	Hex-codes	Address
	20	301A
HLT	76	301B

Program to Compare Two Hexadecimal Numbers

Mnemonics	Hex-codes	Address
MVI A, XX	3E XX	3000 3001
MOV B, A	47	3002
MVI A, YY	3E YY	3003 3004
CMP B	B8	3005
JZ Equal	CA 0B 30	3006 3007 3008
JC Less	DA 0E 30	3009 300A 300B
JMP Greater	C3 11 30	300C 300D 300E
Equal: MVI A, 00	3E 00	300F 3010
JMP End	C3 14 30	3011 3012 3013
Less: MVI A, -1	3E FF	3014 3015
JMP End	C3 14 30	3016 3017 3018
Greater: MVI A, 01	3E 01	3019 301A
End: STA 2050H	32 50 20	301B 301C 301D

Mnemonics	Hex-codes	Address
HLT	76	301E

Program to Count the Number of 1's in the Contents of Register B

Mnemonics	Hex-codes	Address
MVI C, 00	0E 00	3000 3001
Loop: RLC	07	3002
JNC Skip	D2 06 30	3003 3004 3005
INR C	0C	3006
Skip: DCR B	05	3007
JNZ Loop	C2 02 30	3008 3009 300A
MOV A, C	79	300B
STA 2050H	32 50 20	300C 300D 300E
HLT	76	300F