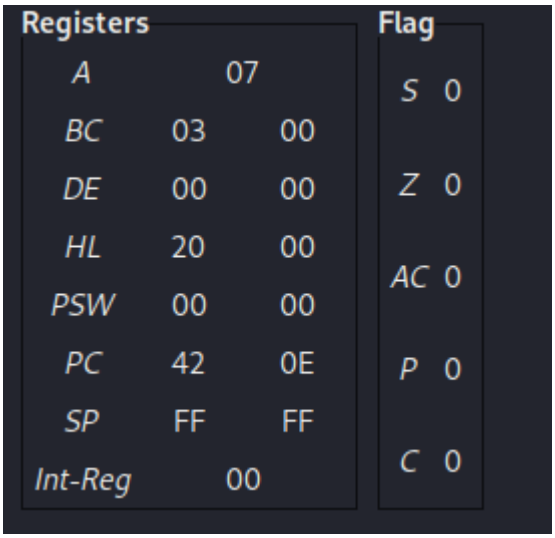
Tavaheed Tariq

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

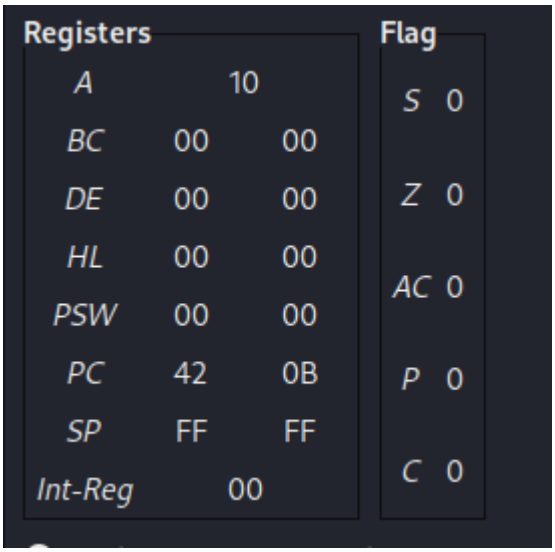
```
;Add Two 8 bit numbers using add immediate

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
```

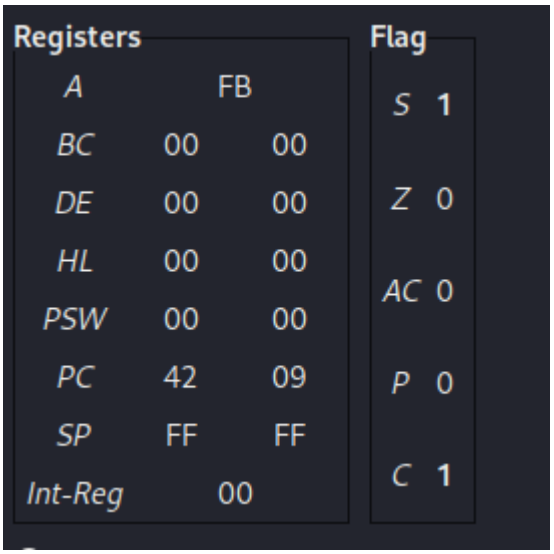
Subtraction of two numbers using immediate values

```
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)

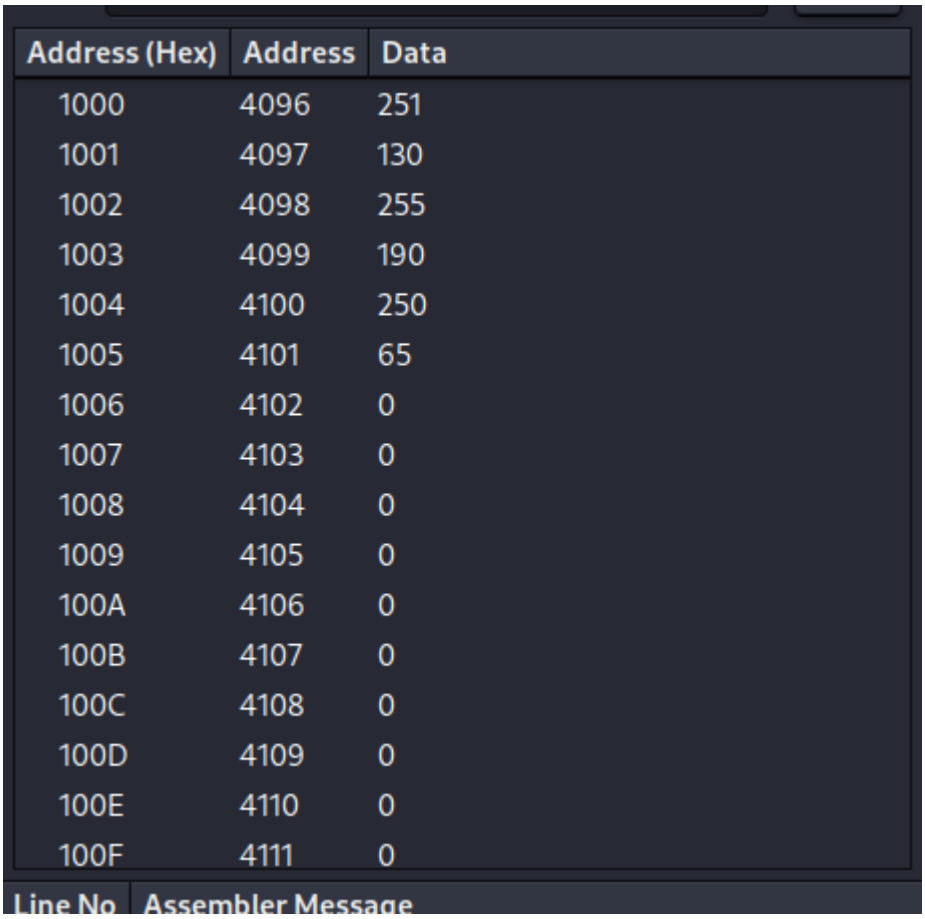
```
jmp start
;code
start:  nop
        MVI A, 05H    ; Load 05H into accumulator (first number)
        SUI 0AH       ; Subtract 0AH from accumulator (second number)
        hlt
```



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
```



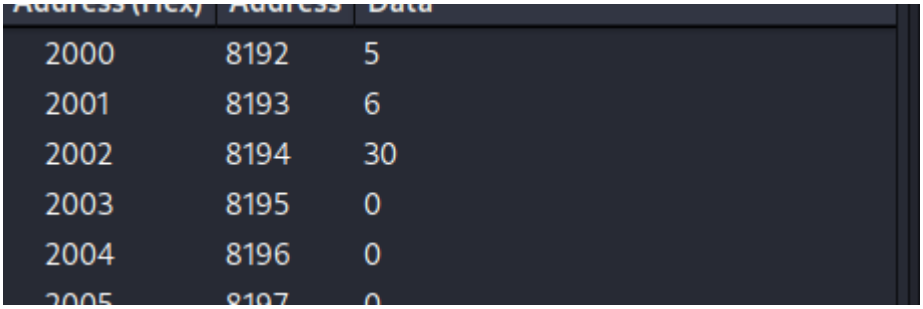
Address (Hex)	Address	Data
1000	4096	251
1001	4097	130
1002	4098	255
1003	4099	190
1004	4100	250
1005	4101	65
1006	4102	0
1007	4103	0
1008	4104	0
1009	4105	0
100A	4106	0
100B	4107	0
100C	4108	0
100D	4109	0
100E	4110	0
100F	4111	0

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

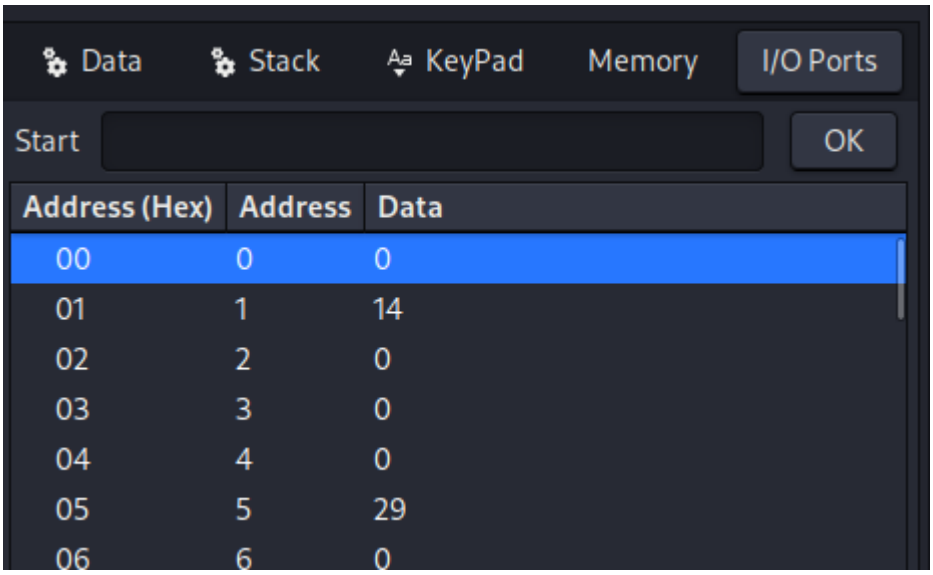
Division of two 8-bit numbers

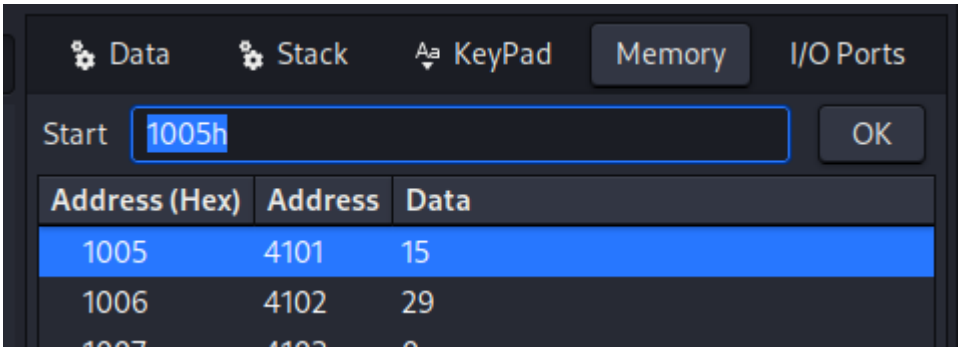
```
jmp start
; code
start: nop
       LXI H, 2000H    ; Load address 2000H into HL
       MOV B, M        ; Load first number (dividend) into B
       INX H           ; Point to 2001H
       MOV C, M        ; Load second number (divisor) into C
       MVI D, 00H      ; Clear D (for quotient)

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 B, M	46	3003
IN 01H	DB 01	3004 3005
SUB B	90	3006
LXI H, 2000H	21 00 20	3007 3008 3009
MOV M, A	77	3010
HLT	76	3011

result

Address (Hex)	Address	Data
2000	8192	254
2001	8193	0

inputs

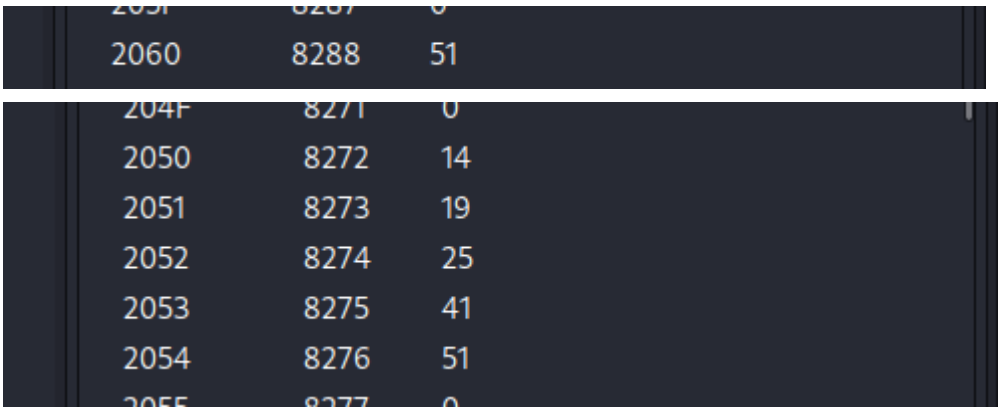
Address (Hex)	Address	Data
00	0	0
01	1	13

1006	4102	15
1007	4103	0

Find the largest in an array

Mnemonics	Hex-codes	Address
LXI H, 2050H	21 50 20	3000 3001 3002
MOV A, M	7E	3003
STA 2060H	32 60 20	3004 3005 3006
LDA 2040H	3A 40 20	3007 3008 3009
MOV B, A	47	3010
DCR B	05	3011
INX H	23	3012
MOV A, M	7E	3013
LDA 2060H	3A 60 20	3014 3015 3016
CMP M	97	3017
JNC SKIP	D2 12 30	3018 3019 3020

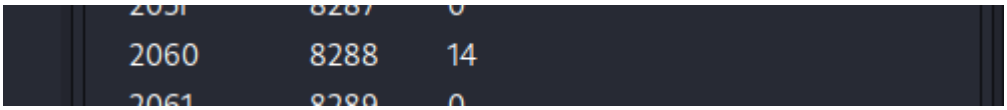
Mnemonics	Hex-codes	Address
MOV A, M	7E	3021
STA 2060H	32 60 20	3022 3023 3024
SKIP: DCR B	05	3025
JNZ LOOP	C2 12 30	3026 3027 3028
HLT	76	3029

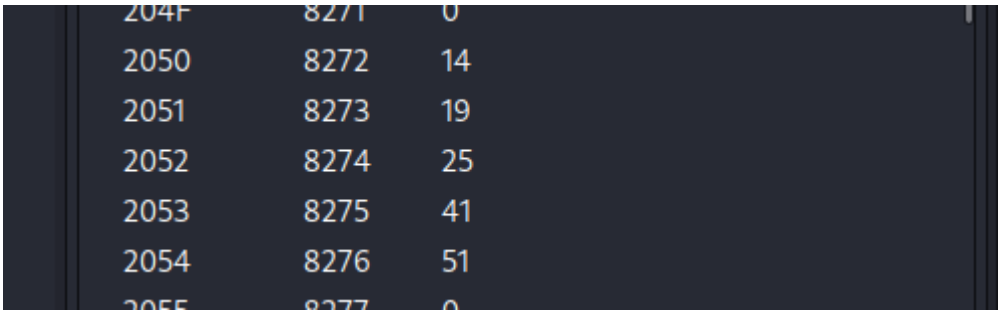


Find the Smallest Number in an Array

Mnemonics	Hex-codes	Address
LXI H, 2050H	21 50 20	3000 3001 3002
MOV A, M	7E	3003
STA 2060H	32 60 20	3004 3005 3006
LDA 2040H	3A 40 20	3007 3008 3009
MOV B, A	47	3010
DCR B	05	3011
INX H	23	3012
MOV A, M	7E	3013

Mnemonics	Hex-codes	Address
LDA 2060H	3A 60 20	3014 3015 3016
CMP M	97	3017
JC SKIP	DA 12 30	3018 3019 3020
MOV A, M	7E	3021
STA 2060H	32 60 20	3022 3023 3024
SKIP: DCR B	05	3025
JNZ LOOP	C2 12 30	3026 3027 3028
HLT	76	3029

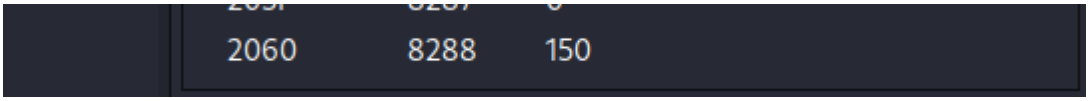


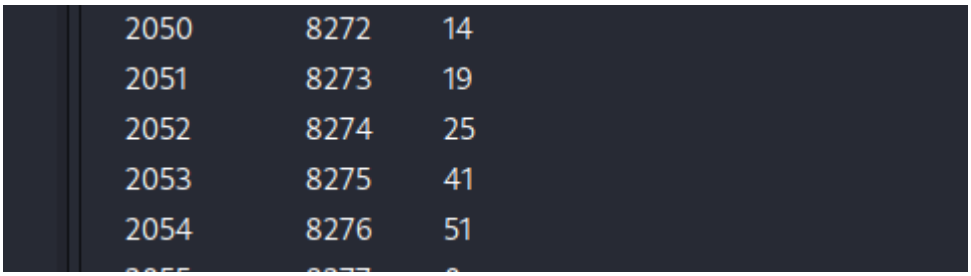


Program to Calculate the Sum of Numbers in an Array and Handle Carry

Mnemonics	Hex-codes	Address
LXI H, 2050H	21 50 20	3000 3001 3002
MVI A, 00H	3E 00	3003 3004

Mnemonics	Hex-codes	Address
MVI C, 00H	0E 00	3005 3006
STA 2060H	32 60 20	3007 3008 3009
STA 2061H	32 61 20	3010 3011 3012
LDA 2040H	3A 40 20	3013 3014 3015
MOV B, A	47	3016
LOOP: LDA 2060H	3A 60 20	3017 3018 3019
ADD M	86	3020
JNC SKIP	D2 12 30	3021 3022 3023
INR C	0C	3024
SKIP: STA 2060H	32 60 20	3025 3026 3027
MOV A, C	4F	3028
STA 2061H	32 61 20	3029 3030 3031
INX H	23	3032
DCR B	05	3033
JNZ LOOP	C2 12 30	3034 3035 3036
HLT	76	3037



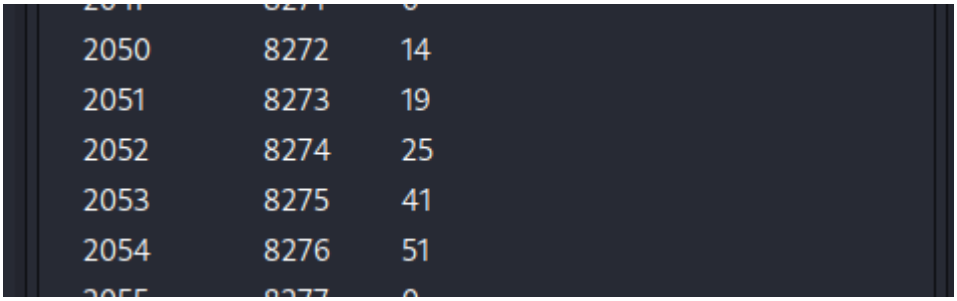


2050	8272	14
2051	8273	19
2052	8274	25
2053	8275	41
2054	8276	51

Program to Sort an Array in Ascending Order

Mnemonics	Hex-codes	Address
LDA 2040H	3A 40 20	3000 3001 3002
MOV C, A	4F	3003
DCR C	0D	3004
OUTER: LXI H, 2050H	21 50 20	3005 3006 3007
MOV B, C	47	3008
INNER: MOV A, M	7E	3009
INX H	23	3010
CMP M	94	3011
JC SKIP	DA 12 30	3012 3013 3014
MOV D, M	57	3015
MOV M, A	7E	3016
DCX H	0D	3017
MOV M, D	56	3018
INX H	23	3019
SKIP: DCR B	05	3020
JNZ INNER	C2 12 30	3021 3022 3023
DCR C	0D	3024
JNZ OUTER	C2 05	3025 3026

Mnemonics	Hex-codes	Address
	30	3027
HLT	76	3028



Program to Implement Bubble Sort in Descending Order

Mnemonics	Hex-codes	Address
LDA 2040H	3A 40 20	0000 0001 0002
MOV C, A	79	0003
DCR C	0D	0004
OUTER: LXI H, 2050H	21 50 20	0005 0006 0007
MOV B, C	41	0008
INNER: MOV A, M	7E	0009
INX H	23	000A
CMP M	CP	000B
JNC SKIP	D2 08 00	000C 000D 000E
MOV D, M	5A	000F
MOV M, A	77	0010
DCX H	0A	0011
MOV M, D	77	0012
INX H	23	0013

Mnemonics	Hex-codes	Address
SKIP: DCR B	05	0014
JNZ INNER	C2 08 00	0015 0016 0017
DCR C	0D	0018
JNZ OUTER	C2 05 00	0019 0020 0021
HLT	76	0022

2050	8272	51
2051	8273	41
2052	8274	25
2053	8275	19
2054	8276	14

Program to Find the Factorial of a Number

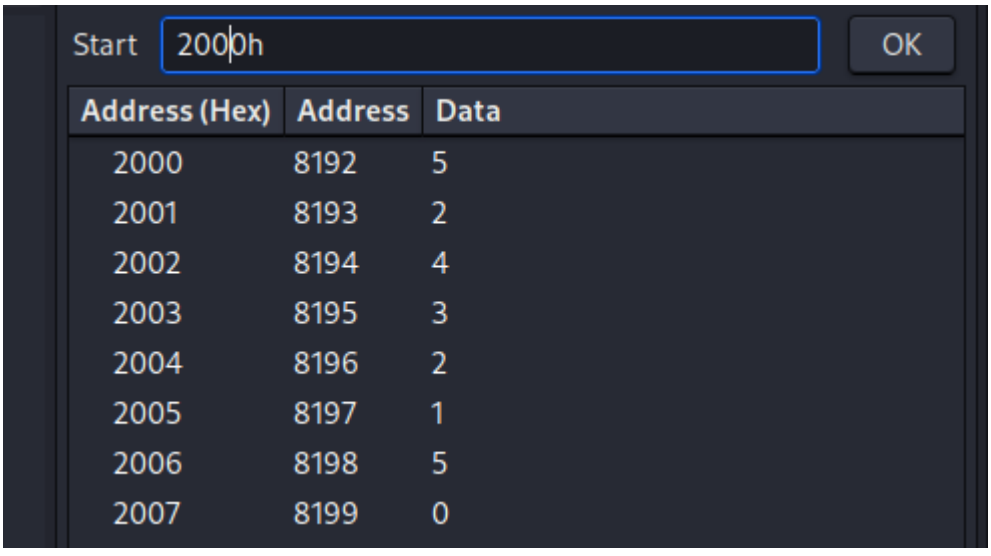
Mnemonics	Hex-codes	Address
LDA 2050H	3A 50 20	3000 3001 3002
MOV B, A	47	3003
MVI A, 01H	3E 01	3004 3005
STA 2060H	32 60 20	3006 3007 3008
MOV A, B	78	3009
CPI 01H	F9 01	300A 300B
JC DONE	DA 0C 30	300C 300D 300E
JZ DONE	ZF 0C	300F 3010

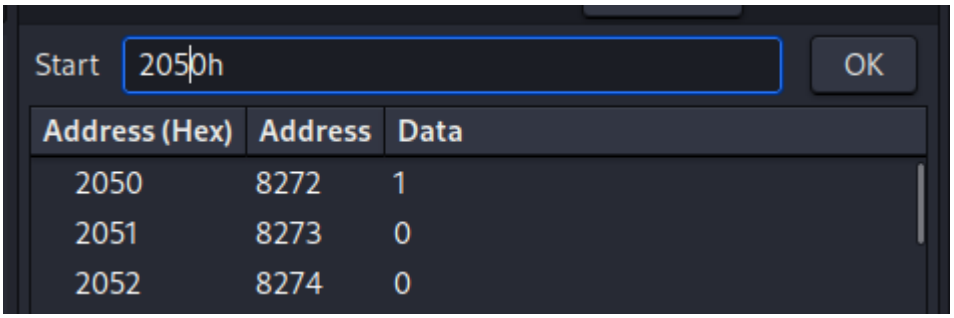
Mnemonics	Hex-codes	Address
	30	3011
LDA 2060H	3A 60 20	3012 3013 3014
MOV C, A	4F	3015
MVI D, 00H	06 00	3016 3017
ADD C	80	3018
INR D	14	3019
MOV E, A	5F	3020
MOV A, B	78	3021
CMP D	FE	3022
MOV A, E	7F	3023
JNZ LOOP	C2 14 30	3024 3025 3026
STA 2060H	32 60 20	3027 3028 3029
DCR B	10	3030
CPI 01H	F9 01	3031 3032
JNZ MULT	C2 06 30	3033 3034 3035
DONE: HLT	76	3036

Program to Search for a Number in an Array

Mnemonics	Hex-codes	Address
LXI H, 2000H	21 00 20	3000 3001 3002
MOV C, M	7E	3003
INX H	23	3004

Mnemonics	Hex-codes	Address
MOV A, M	7E	3005
INX H	23	3006
CMP M	96	3007
JZ FOUND	FZ 0C 30	3008 3009 3010
INX H	23	3011
DCR C	0D	3012
JNZ LOOP	C2 10 30	3013 3014 3015
MVI A, 00H	3E 00	3016 3017
STA 2050H	32 50 20	3018 3019 3020
HLT	76	3021
FOUND: MVI A, 01H	3E 01	3022 3023
STA 2050H	32 50 20	3024 3025 3026
HLT	76	3027

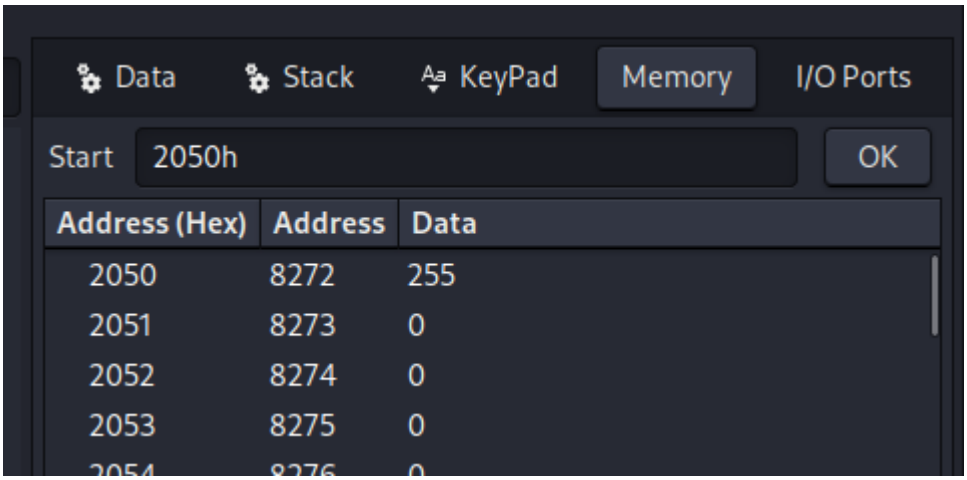




Program to Compare Two Hexadecimal Numbers

Mnemonics	Hex-codes	Address
MVI A, 05H	3E 05	3000 3001
MOV B, A	47	3002
MVI A, 01H	3E 01	3003 3004
CMP B	B8	3005
JZ Equal	FZ 0C 30	3006 3007 3008
JC Less	F2 0A 30	3009 3010 3011
JMP Greater	C3 14 30	3012 3013 3014
Equal: MVI A, 00H	3E 00	3015 3016
JMP END	C3 18 30	3017 3018 3019
Less: MVI A, 0FFH	3E FF	3020 3021
JMP END	C3 18 30	3022 3023 3024
Greater: MVI A, 01H	3E 01	3025 3026

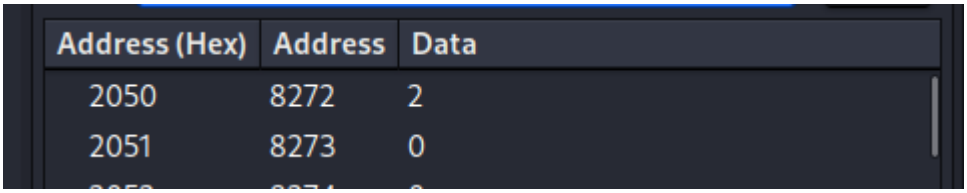
Mnemonics	Hex-codes	Address
END: STA 2050H	32 50 20	3027 3028 3029
HLT	76	3030



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

Mnemonics	Hex-codes	Address
MVI B, 10H	3E 10	3000 3001
ADD B	80	3002
MVI C, 00H	3E 00	3003 3004
Loop: RLC	07	3005
JNC Skip	D2 08 30	3006 3007 3008
INR C	04	3009
Skip: DCR B	0D	3010
JNZ Loop	C2 05 30	3011 3012 3013
MOV A, C	47	3014

Mnemonics	Hex-codes	Address
STA 2050H	32 50 20	3015 3016 3017
HLT	76	3018



Address (Hex)	Address	Data
2050	8272	2
2051	8273	0