Open-Ended Asssignment 1

Aditya Gupta

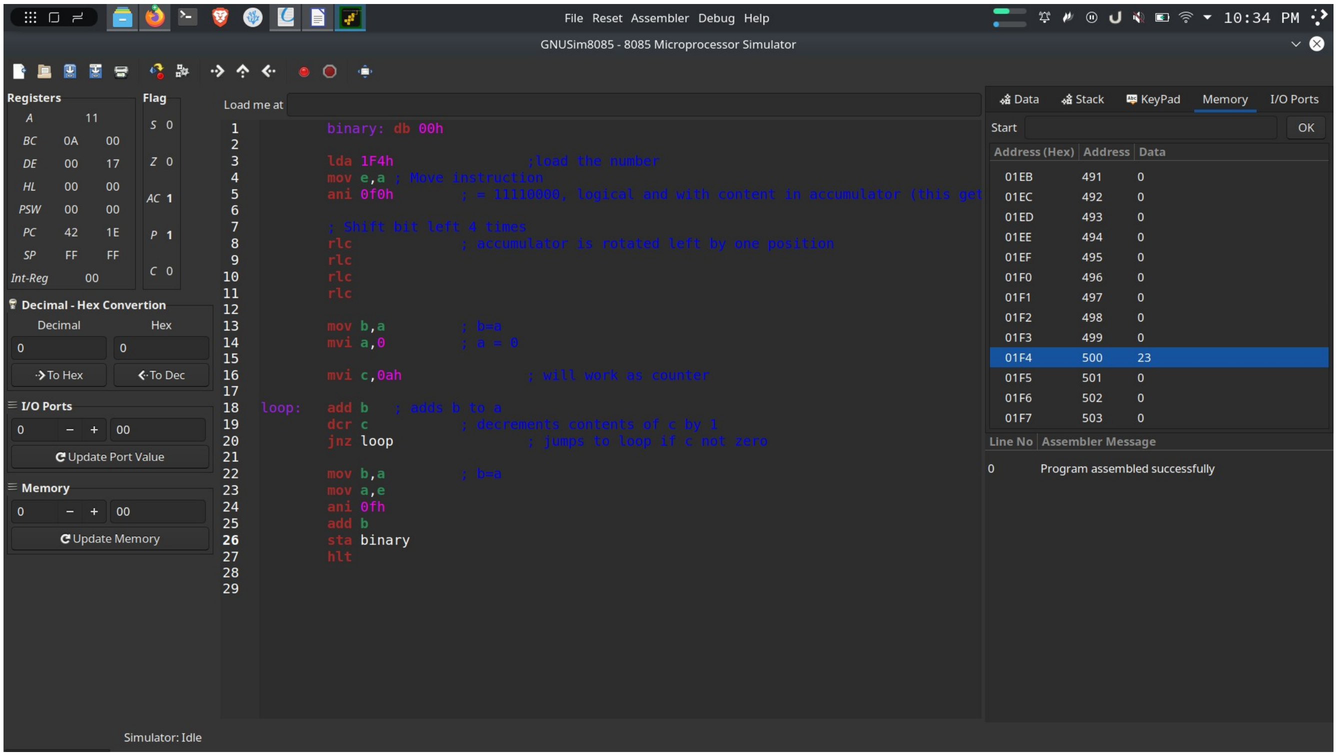
Note: In some programs, it may assume some data already present at some memory location, in GNUsim, will be in comment above program or in screenshot Memory section of gnusim8085

P1. Binary Packed Decimal (BCD) to Binary

Steps:

Break the BCD number into nibbles (4-bit)
Convert each nibble into decimal digit
Combine the digits to form the decimal number
Convert the decimal number into binary

```
     ; bcd is at 1F4H (500 base 10), for eg. I put 23

     binary:   db 00h

     LDA 1F4h       ;load the number into accumulator
     MOV e,a   ; Move instruction
     ANI 0f0h  ; = 11110000, logical and with content in accumulator
(this gets the ‘nibble’)

     ; Shift bit left 4 times
     RLC       ; accumulator is rotated left by one
position
     RLC
     RLC
     RLC

     MOV b,a        ; b=a
     MVI a,0        ; a = 0

     MVI c,0ah      ; will work as counter

loop:     add b     ; adds b to a
     DCR c          ; decrements contents of c by 1
     JNZ loop       ; jumps to loop if c not zero

     MOV b,a        ; b=a
     MOV a,e
     ANI 0fh
     ADD b
     STA binary
     HLT
```



Registers		
A	11	
BC	0A	00
DE	00	17
HL	00	00
PSW	00	00
PC	42	1E
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	0
AC	1
P	1
C	0

```
         binary: db 00h

         lda 1F4h              ;load the number
         mov e,a ; Move instruction
         ani 0f0h         ; = 11110000, logical and with content in accumulator (this get

         ; Shift bit left 4 times
         rlc              ; accumulator is rotated left by one position
         rlc
         rlc
         rlc

         mov b,a          ; b=a
         mvi a,0          ; a = 0

         mvi c,0ah             ; will work as counter

loop:    add b    ; adds b to a
         dcr c            ; decrements contents of c by 1
         jnz loop              ; jumps to loop if c not zero

         mov b,a          ; b=a
         mov a,e
         ani 0fh
         add b
         sta binary
         hlt
```

Address (Hex)	Address	Data
01EB	491	0
01EC	492	0
01ED	493	0
01EE	494	0
01EF	495	0
01F0	496	0
01F1	497	0
01F2	498	0
01F3	499	0
01F4	500	23
01F5	501	0
01F6	502	0
01F7	503	0

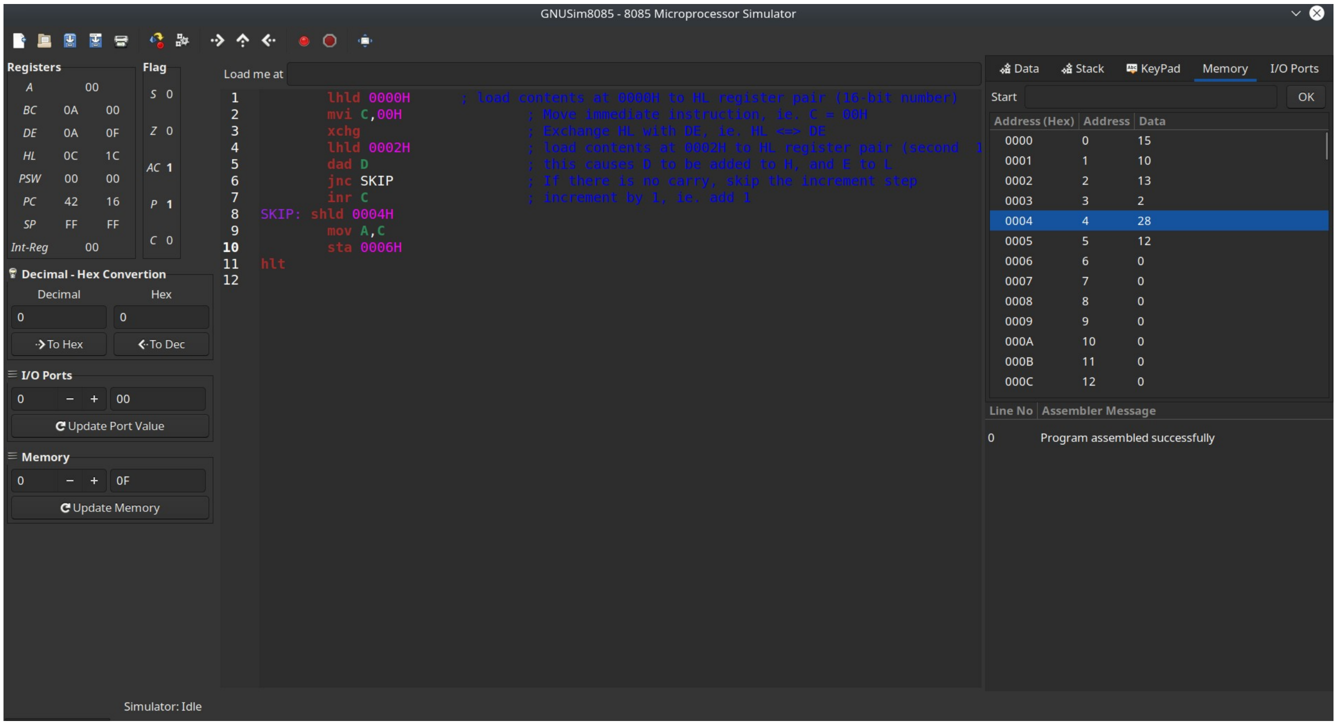
Line No	Assembler Message
0	Program assembled successfully

P2. Add two 16-bit numbers

Steps: Since registers are 8-bit, store each number in register pairs, then add them

```
     LHLD 0000H     ; load contents at 0000H to HL register pair (16-
bit number)
     MVI C,00H      ; Move immediate instruction, ie. C = 00H
     XCHG           ; Exchange HL with DE, ie. HL <=> DE
     LHLD 0002H          ; load contents at 0002H to HL register pair
(second  16-bit number)
     DAD D               ; this causes D to be added to H, and E to L
     JNC SKIP       ; If there is no carry, skip the increment step
     INR C               ; increment by 1, ie. add 1
SKIP: SHLD 0004H    ; store HL at 0004H
     MOV A,C        ; a=c
```

```
    STA 0006H     ; store accumulator content at 0006H
hlt
```



```
        lhld 0000H      ; load contents at 0000H to HL register pair (16-bit number)
        mvi C,00H               ; Move immediate instruction, ie. C = 00H
        xchg                    ; Exchange HL with DE, ie. HL <=> DE
        lhld 0002H              ; load contents at 0002H to HL register pair (second  1
        dad D                   ; this causes D to be added to H, and E to L
        jnc SKIP                ; If there is no carry, skip the increment step
        inr C                   ; increment by 1, ie. add 1
SKIP: shld 0004H
        mov A,C
        sta 0006H
hlt
```

Registers

A	00	
BC	0A	00
DE	0A	0F
HL	0C	1C
PSW	00	00
PC	42	16
SP	FF	FF
Int-Reg	00	

Flag

S	0
Z	0
AC	1
P	1
C	0

Decimal: 0 Hex: 0

I/O Ports: 0 − + 00

Memory: 0 − + 0F

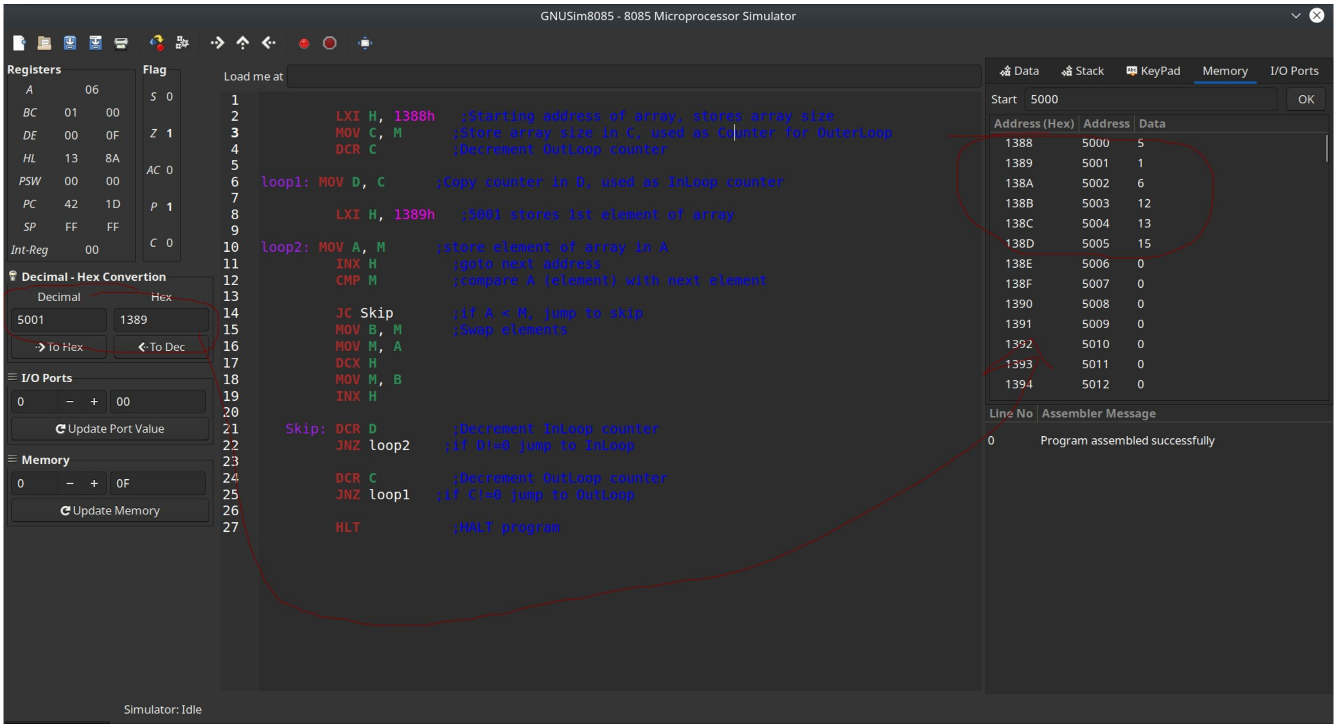
Address (Hex)	Address	Data
0000	0	15
0001	1	10
0002	2	13
0003	3	2
0004	4	28
0005	5	12
0006	6	0
0007	7	0
0008	8	0
0009	9	0
000A	10	0
000B	11	0
000C	12	0

Line No	Assembler Message
0	Program assembled successfully

P3. Sort array (ascending order)

```
; 1388H -> Number of elements (5 in my case)
; 1339H -> arr[0]
; 1340H -> arr[1]
; and so on

         LXI H, 1388h   ;Starting address of array, stores array size
         MOV C, M      ;Store array size in C, used as Counter for
OuterLoop
         DCR C         ;Decrement OutLoop counter

loop1: MOV D, C      ;Copy counter in D, used as InLoop counter

         LXI H, 1389h   ; 1389h stores 1st element of array

loop2: MOV A, M      ;store element of array in A
         INX H         ;goto next address
         CMP M         ;compare A (element) with next element

         JC Skip       ;if A < M, jump to skip
```

```
        MOV B, M       ;Swap elements
        MOV M, A
        DCX H
        MOV M, B
        INX H

  Skip: DCR D          ;Decrement loop2 counter
        JNZ loop2     ;if D!=0 jump to InLoop

        DCR C          ;Decrement loop1 counter
        JNZ loop1    ;if C!=0 jump to  loop1

        HLT            ;HALT program
```



GNUSim8085 - 8085 Microprocessor Simulator

Registers		
A	06	
BC	01	00
DE	00	0F
HL	13	8A
PSW	00	00
PC	42	1D
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	1
AC	0
P	1
C	0

Decimal - Hex Convertion

Decimal: 5001 Hex: 1389

I/O Ports: 0 00

Memory: 0 0F

```
        LXI H, 1388h   ;Starting address of array, stores array size
        MOV C, M      ;Store array size in C, used as Counter for OuterLoop
        DCR C         ;Decrement OutLoop counter

loop1: MOV D, C      ;Copy counter in D, used as InLoop counter

        LXI H, 1389h   ;5001 stores 1st element of array

loop2: MOV A, M      ;store element of array in A
        INX H         ;goto next address
        CMP M         ;compare A (element) with next element

        JC Skip       ;if A < M, jump to skip
        MOV B, M      ;Swap elements
        MOV M, A
        DCX H
        MOV M, B
        INX H

   Skip: DCR D         ;Decrement InLoop counter
        JNZ loop2    ;if D!=0 jump to InLoop

        DCR C         ;Decrement OutLoop counter
        JNZ loop1   ;if C!=0 jump to OutLoop

        HLT           ;HALT program
```

Start 5000

Address (Hex)	Address	Data
1388	5000	5
1389	5001	1
138A	5002	6
138B	5003	12
138C	5004	13
138D	5005	15
138E	5006	0
138F	5007	0
1390	5008	0
1391	5009	0
1392	5010	0
1393	5011	0
1394	5012	0

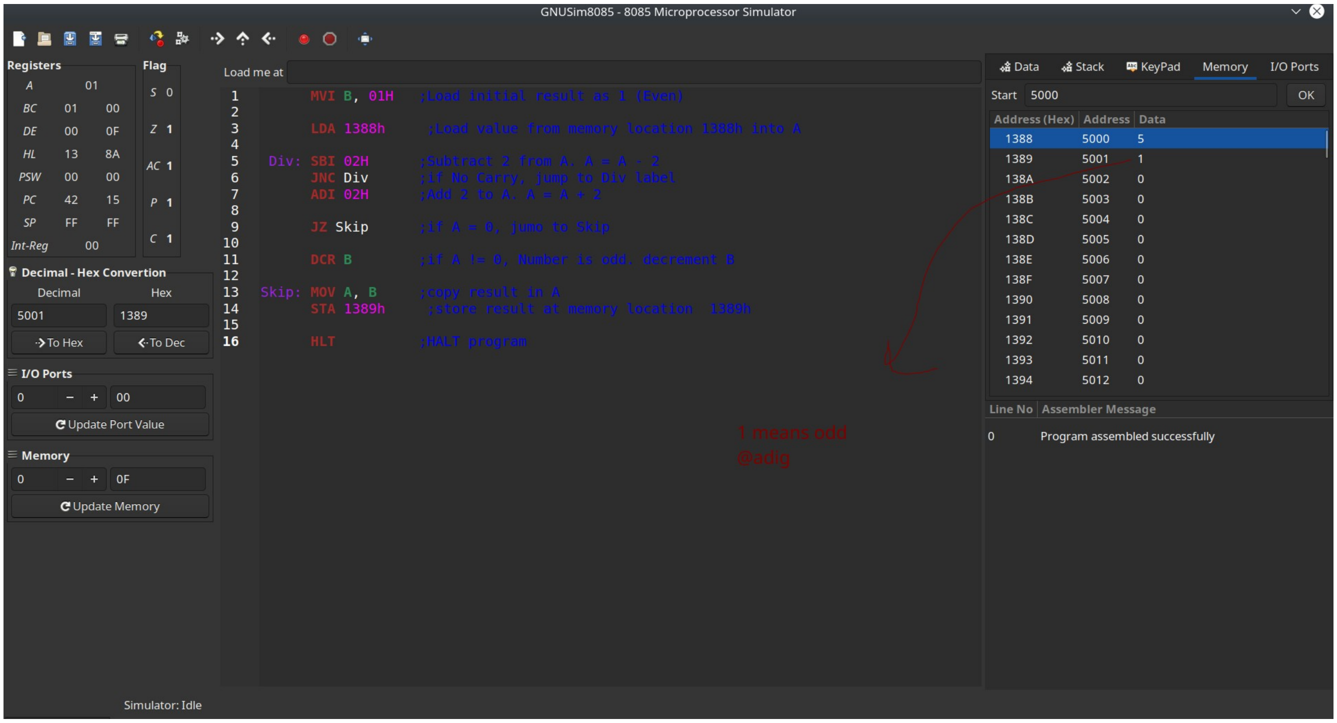
Line No	Assembler Message
0	Program assembled successfully

Simulator: Idle

P4. Check if number even or odd

```
      MVI B, 01H   ;Load initial result as 1 (Even)

      LDA 1388h     ;Load value from memory location 1388h into A

 Div: SBI 02H      ;Subtract 2 from A. A = A - 2
      JNC Div      ;if No Carry, jump to Div label
      ADI 02H      ;Add 2 to A. A = A + 2
```

```
        JZ Skip      ;if A = 0, jumo to Skip

        DCR B        ;if A != 0, Number is odd. decrement B

Skip:   MOV A, B     ;copy result in A
        STA 1389h     ;store result at memory location  1389h

        HLT          ;HALT program
```



Registers		
A	01	
BC	01	00
DE	00	0F
HL	13	8A
PSW	00	00
PC	42	15
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	1
AC	1
P	1
C	1

Decimal: 5001 Hex: 1389

I/O Ports: 0 − + 00

Memory: 0 − + 0F

```
        MVI B, 01H   ;Load initial result as 1 (Even)

        LDA 1388h     ;Load value from memory location 1388h into A

 Div:   SBI 02H      ;Subtract 2 from A. A = A - 2
        JNC Div      ;if No Carry, jump to Div label
        ADI 02H      ;Add 2 to A. A = A + 2

        JZ Skip      ;if A = 0, jumo to Skip

        DCR B        ;if A != 0, Number is odd. decrement B

Skip:   MOV A, B     ;copy result in A
        STA 1389h     ;store result at memory location  1389h

        HLT          ;HALT program
```

Start 5000

Address (Hex)	Address	Data
1388	5000	5
1389	5001	1
138A	5002	0
138B	5003	0
138C	5004	0
138D	5005	0
138E	5006	0
138F	5007	0
1390	5008	0
1391	5009	0
1392	5010	0
1393	5011	0
1394	5012	0

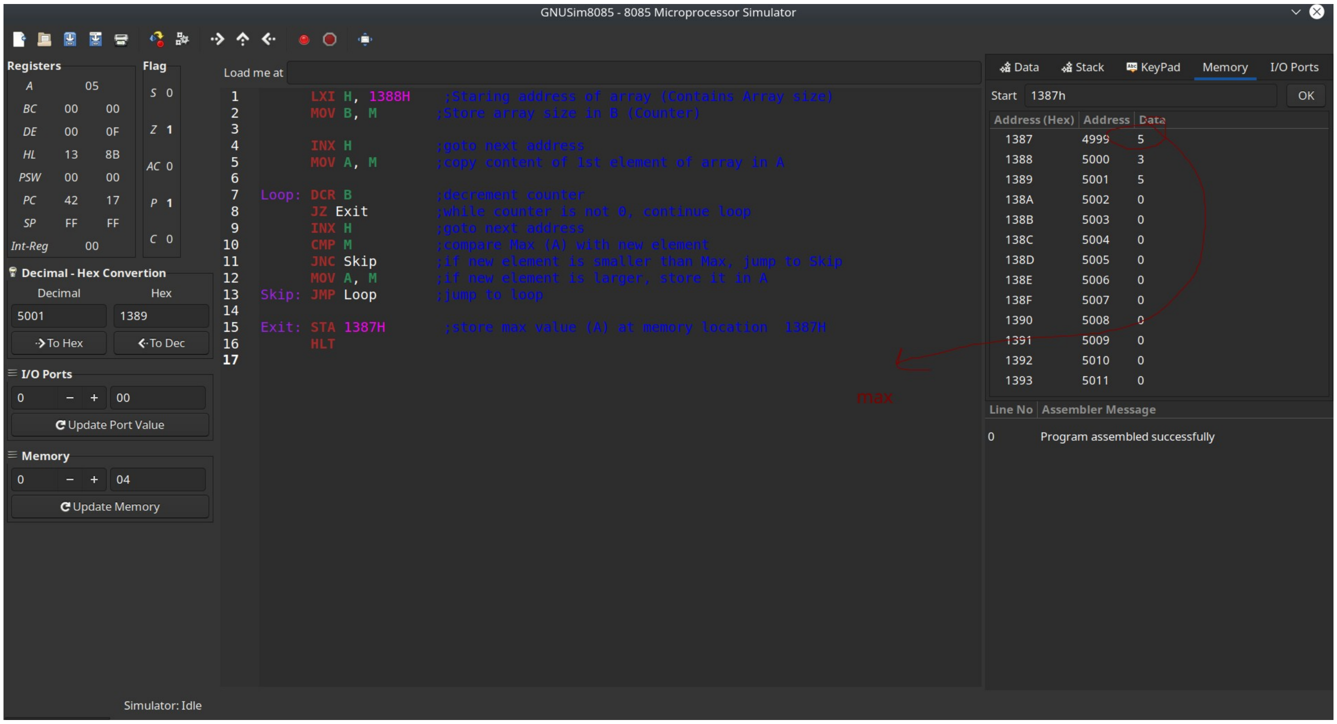
Line No	Assembler Message
0	Program assembled successfully

1 means odd
@adig

P6. Find largest number in array

```
; 1387H -> Will store max
; 1388H -> Number of elements (5 in my case)
; 1339H -> arr[0]
; 1340H -> arr[1]
; and so on

        LXI H, 1388H    ;Staring address of array (Contains Array size)
        MOV B, M       ;Store array size in B (Counter)

        INX H          ;goto next address
        MOV A, M       ;copy content of 1st element of array in A
```

```
Loop: DCR B          ;decrement counter
      JZ Exit        ;while counter is not 0, continue loop
      INX H          ;goto next address
      CMP M          ;compare Max (A) with new element
      JNC Skip       ;if new element is smaller than Max, jump to
Skip
      MOV A, M       ;if new element is larger, store it in A
Skip: JMP Loop       ;jump to loop

Exit: STA 1387H       ;store max value (A) at memory location  1387H
      HLT
```



```
      LXI H, 1388H    ;Staring address of array (Contains Array size)
      MOV B, M        ;Store array size in B (Counter)

      INX H           ;goto next address
      MOV A, M        ;copy content of 1st element of array in A

Loop: DCR B           ;decrement counter
      JZ Exit         ;while counter is not 0, continue loop
      INX H           ;goto next address
      CMP M           ;compare Max (A) with new element
      JNC Skip        ;if new element is smaller than Max, jump to Skip
      MOV A, M        ;if new element is larger, store it in A
Skip: JMP Loop        ;jump to loop

Exit: STA 1387H        ;store max value (A) at memory location  1387H
      HLT
```

Registers		
A	05	
BC	00	00
DE	00	0F
HL	13	8B
PSW	00	00
PC	42	17
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	1
AC	0
P	1
C	0

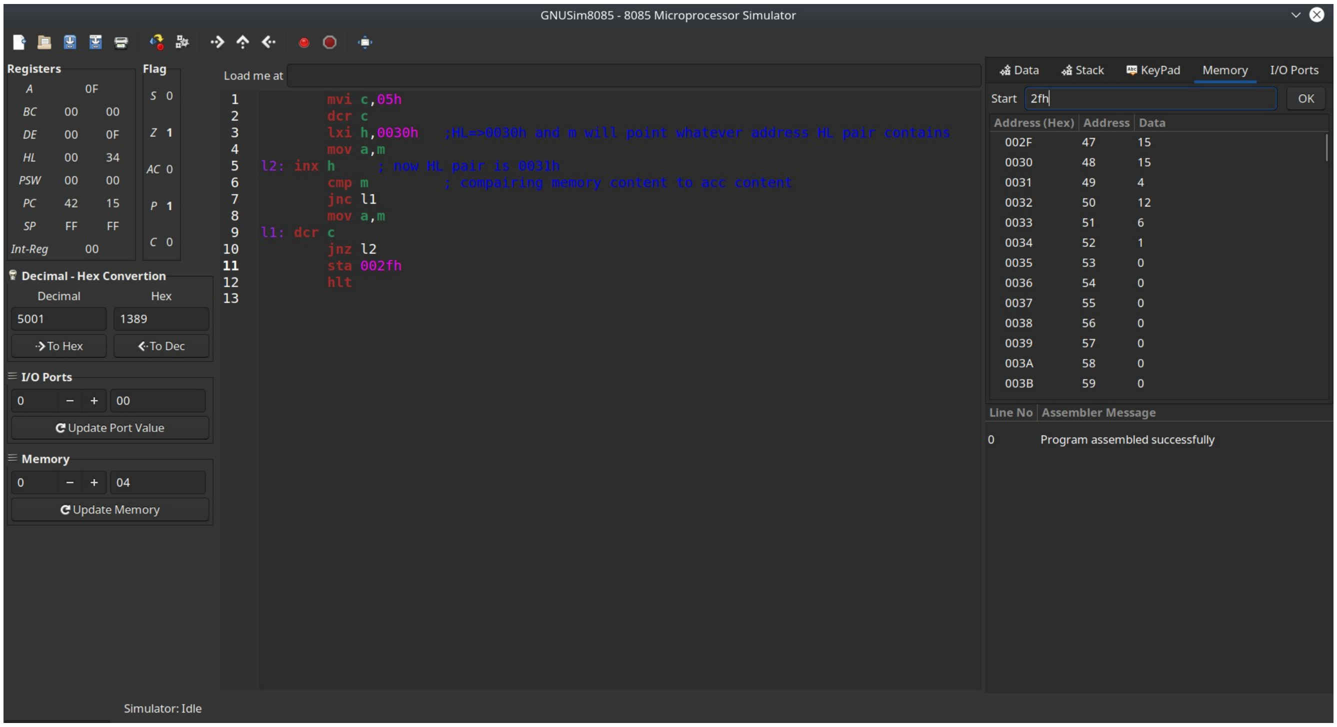
Address (Hex)	Address	Data
1387	4999	5
1388	5000	3
1389	5001	5
138A	5002	0
138B	5003	0
138C	5004	0
138D	5005	0
138E	5006	0
138F	5007	0
1390	5008	0
1391	5009	0
1392	5010	0
1393	5011	0

Line No	Assembler Message
0	Program assembled successfully

P7. Given 5 numbers, find max (stored in 2fH)

```
;Numbers are stored in 0030h to 0034h and result will be store in
002fh

     mvi c,05h
     dcr c
     lxi h,0030h   ;HL=>0030h and m will point whatever address HL
pair contains
```

```
     mov a,m
l2: inx h      ; now HL pair is 0031h
     cmp m          ; compairing memory content to acc content
     jnc l1
     mov a,m
l1: dcr c
     jnz l2
     sta 002fh
     hlt
```



```
        mvi c,05h
        dcr c
        lxi h,0030h   ;HL=>0030h and m will point whatever address HL pair contains
        mov a,m
l2: inx h      ; now HL pair is 0031h
        cmp m          ; compairing memory content to acc content
        jnc l1
        mov a,m
l1: dcr c
        jnz l2
        sta 002fh
        hlt
```

Registers		
A	0F	
BC	00	00
DE	00	0F
HL	00	34
PSW	00	00
PC	42	15
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	1
AC	0
P	1
C	0

Decimal: 5001 Hex: 1389

I/O Ports: 0 − + 00

Memory: 0 − + 04

Start: 2fh

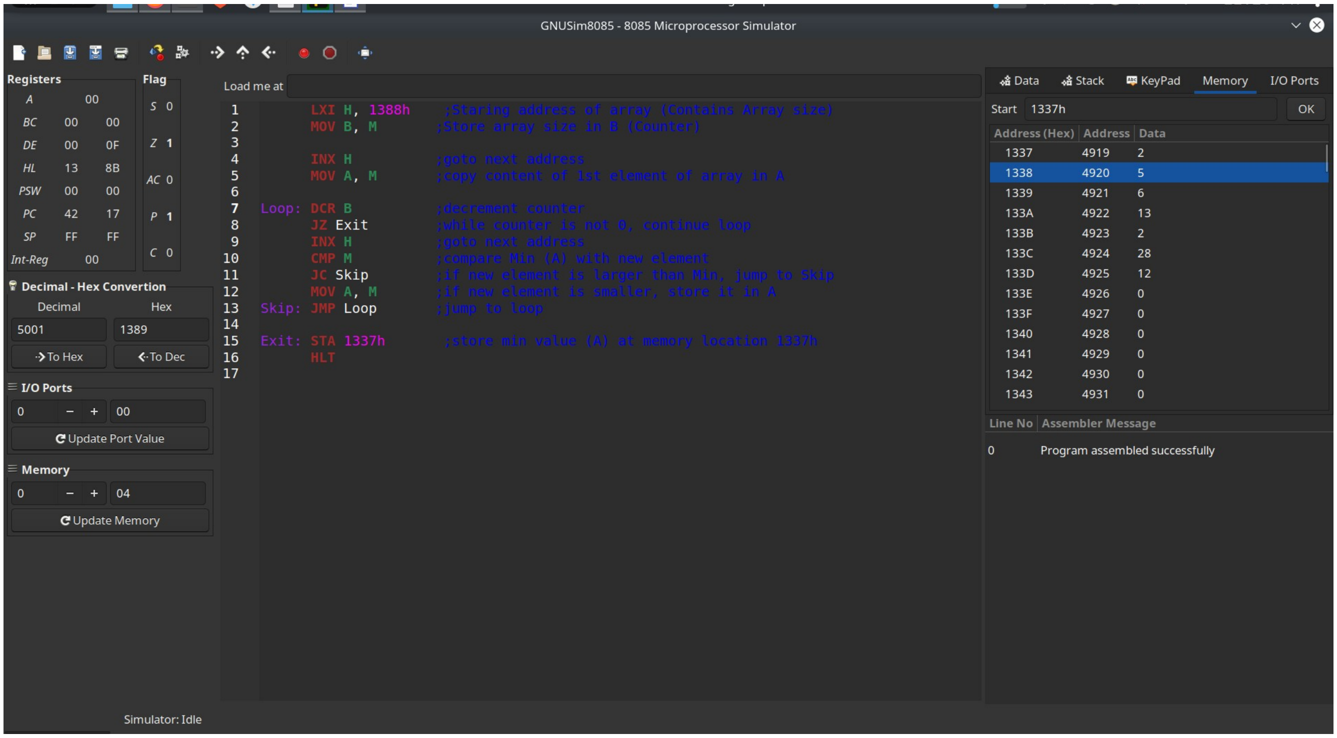
Address (Hex)	Address	Data
002F	47	15
0030	48	15
0031	49	4
0032	50	12
0033	51	6
0034	52	1
0035	53	0
0036	54	0
0037	55	0
0038	56	0
0039	57	0
003A	58	0
003B	59	0

Line No	Assembler Message
0	Program assembled successfully

P8. Minimum in array

```
; 1337H -> Will store min
; 1338H -> Number of elements (5 in my case)
; 1339H -> arr[0]
; and so on

      LXI H, 1338h    ;Staring address of array (Contains Array size)
      MOV B, M       ;Store array size in B (Counter)

      INX H          ;goto next address
      MOV A, M       ;copy content of 1st element of array in A
```

```
Loop: DCR B          ;decrement counter
      JZ Exit        ;while counter is not 0, continue loop
      INX H          ;goto next address
      CMP M          ;compare Min (A) with new element
      JC Skip        ;if new element is larger than Min, jump to Skip
      MOV A, M       ;if new element is smaller, store it in A
Skip: JMP Loop       ;jump to loop

Exit: STA 1337h       ;store min value (A) at memory location 1337h
      HLT
```



Registers		
A	00	
BC	00	00
DE	00	0F
HL	13	8B
PSW	00	00
PC	42	17
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	1
AC	0
P	1
C	0

Decimal: 5001 Hex: 1389

I/O Ports: 0 − + 00

Memory: 0 − + 04

```
        LXI H, 1388h     ;Staring address of array (Contains Array size)
        MOV B, M        ;Store array size in B (Counter)

        INX H           ;goto next address
        MOV A, M        ;copy content of 1st element of array in A

Loop:   DCR B           ;decrement counter
        JZ Exit         ;while counter is not 0, continue loop
        INX H           ;goto next address
        CMP M           ;compare Min (A) with new element
        JC Skip         ;if new element is larger than Min, jump to Skip
        MOV A, M        ;if new element is smaller, store it in A
Skip:   JMP Loop        ;jump to loop

Exit:   STA 1337h        ;store min value (A) at memory location 1337h
        HLT
```

Start 1337h

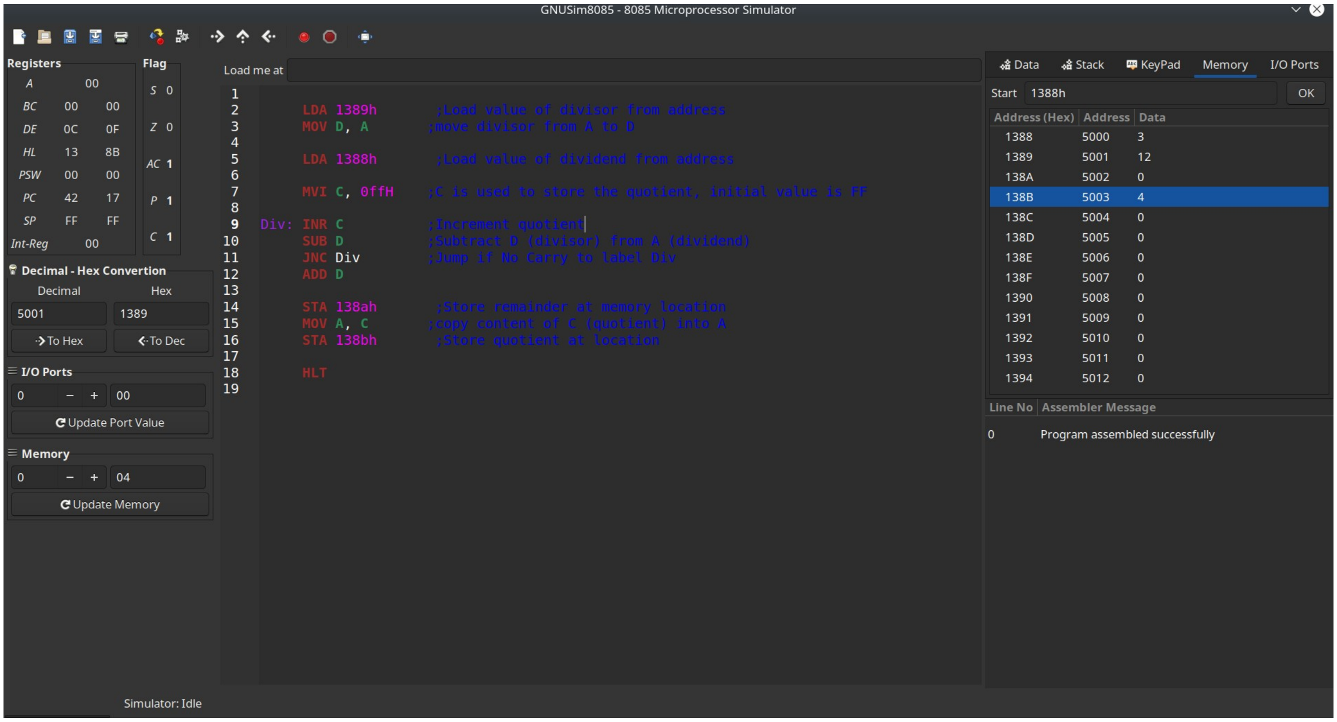
Address (Hex)	Address	Data
1337	4919	2
1338	4920	5
1339	4921	6
133A	4922	13
133B	4923	2
133C	4924	28
133D	4925	12
133E	4926	0
133F	4927	0
1340	4928	0
1341	4929	0
1342	4930	0
1343	4931	0

Line No	Assembler Message
0	Program assembled successfully

P9. Divide 2 byte-length numbers

```
     LDA 1389h       ;Load value of divisor from address 1389h
     MOV D, A       ;move divisor from A to D

     LDA 1388h       ;Load value of dividend from address

     MVI C, 0ffH    ;C is used to store the quotient, initial value
is FF

Div: INR C          ;Increment quotient
```

```
    SUB D           ;Subtract D (divisor) from A (dividend)
    JNC Div         ;Jump if No Carry to label Div
    ADD D

    STA 138ah        ;Store remainder at memory location

    MOV A, C        ;copy content of C (quotient) into A
    STA 138bh        ;Store quotient at location 138b

    HLT
```



```

      LDA 1389h       ;Load value of divisor from address
      MOV D, A       ;move divisor from A to D

      LDA 1388h       ;Load value of dividend from address

      MVI C, 0ffH    ;C is used to store the quotient, initial value is FF

Div:  INR C          ;Increment quotient
      SUB D          ;Subtract D (divisor) from A (dividend)
      JNC Div        ;Jump if No Carry to label Div
      ADD D

      STA 138ah       ;Store remainder at memory location
      MOV A, C       ;copy content of C (quotient) into A
      STA 138bh       ;Store quotient at location

      HLT

```

Registers

A	00	
BC	00	00
DE	0C	0F
HL	13	8B
PSW	00	00
PC	42	17
SP	FF	FF
Int-Reg	00	

Flag

S	0
Z	0
AC	1
P	1
C	1

Decimal	Hex
5001	1389

I/O Ports: 0 – + 00

Memory: 0 – + 04

Start 1388h

Address (Hex)	Address	Data
1388	5000	3
1389	5001	12
138A	5002	0
138B	5003	4
138C	5004	0
138D	5005	0
138E	5006	0
138F	5007	0
1390	5008	0
1391	5009	0
1392	5010	0
1393	5011	0
1394	5012	0

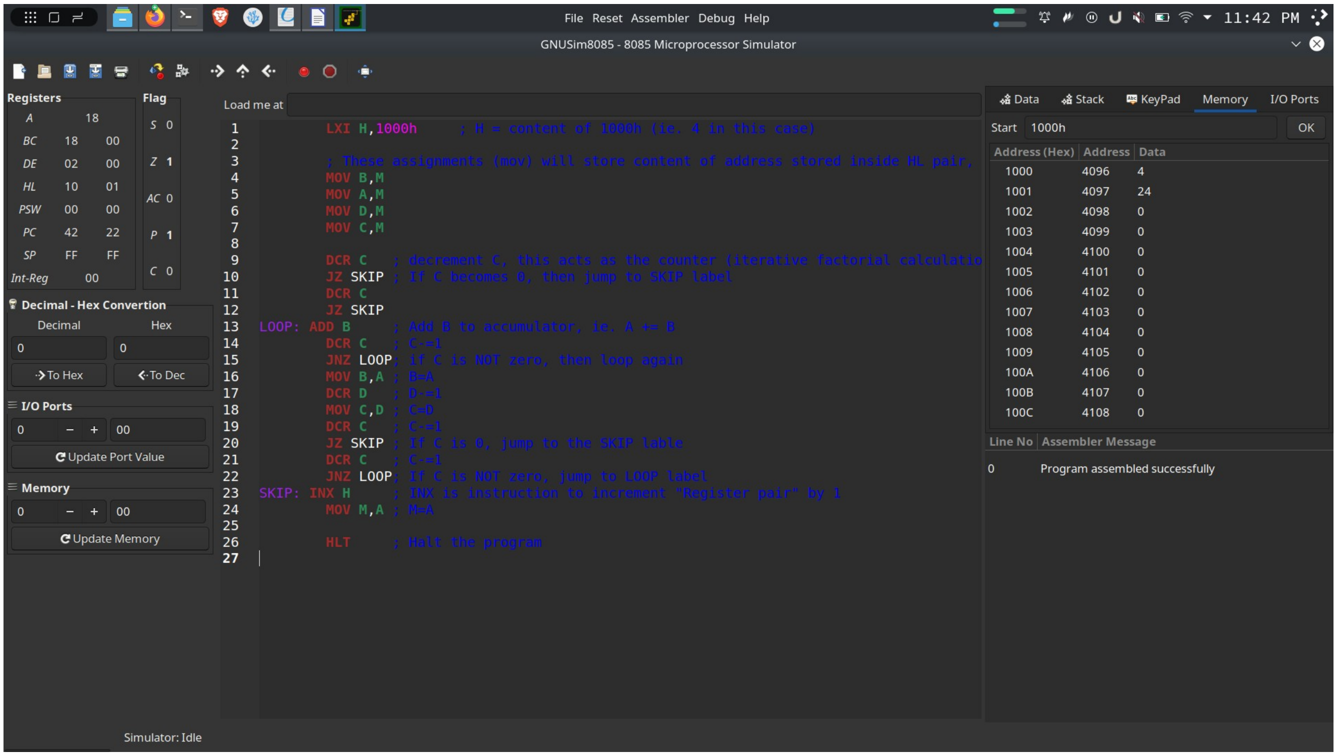
Line No	Assembler Message
0	Program assembled successfully

Simulator: Idle

P10. Factorial

```
    LXI H,1000h    ; H = content of 1000h (ie. 4 in this case)

    ; These assignments (mov) will store content of address stored
inside HL pair, ie. address at memory M
    MOV B,M
    MOV A,M
    MOV D,M
    MOV C,M
```

```
     DCR C     ; decrement C, this acts as the counter (iterative factorial calculation)
     JZ SKIP ; If C becomes 0, then jump to SKIP label
     DCR C
     JZ SKIP
LOOP: ADD B    ; Add B to accumulator, ie. A += B
     DCR C     ; C-=1
     JNZ LOOP; if C is NOT zero, then loop again
     MOV B,A   ; B=A
     DCR D     ; D-=1
     MOV C,D   ; C=D
     DCR C     ; C-=1
     JZ SKIP   ; If C is 0, jump to the SKIP lable
     DCR C     ; C-=1
     JNZ LOOP; If C is NOT zero, jump to LOOP label
SKIP: INX H    ; INX is instruction to increment "Register pair" by 1
     MOV M,A   ; M=A

     HLT  ; Halt the program
```



Registers		
A	18	
BC	18	00
DE	02	00
HL	10	01
PSW	00	00
PC	42	22
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	1
AC	0
P	1
C	0

Decimal: 0 Hex: 0

I/O Ports: 0 − + 00

Memory: 0 − + 00

```
        LXI H,1000h     ; H = content of 1000h (ie. 4 in this case)

        ; These assignments (mov) will store content of address stored inside HL pair,
        MOV B,M
        MOV A,M
        MOV D,M
        MOV C,M

        DCR C   ; decrement C, this acts as the counter (iterative factorial calculatio
        JZ SKIP ; If C becomes 0, then jump to SKIP label
        DCR C
        JZ SKIP
LOOP: ADD B     ; Add B to accumulator, ie. A += B
        DCR C   ; C-=1
        JNZ LOOP; if C is NOT zero, then loop again
        MOV B,A ; B=A
        DCR D   ; D-=1
        MOV C,D ; C=D
        DCR C   ; C-=1
        JZ SKIP ; If C is 0, jump to the SKIP lable
        DCR C   ; C-=1
        JNZ LOOP; If C is NOT zero, jump to LOOP label
SKIP: INX H     ; INX is instruction to increment "Register pair" by 1
        MOV M,A ; M=A

        HLT     ; Halt the program
```

Start 1000h

Address (Hex)	Address	Data
1000	4096	4
1001	4097	24
1002	4098	0
1003	4099	0
1004	4100	0
1005	4101	0
1006	4102	0
1007	4103	0
1008	4104	0
1009	4105	0
100A	4106	0
100B	4107	0
100C	4108	0

Line No	Assembler Message
0	Program assembled successfully