

Practical-1

❖ Transfer operation for 8085

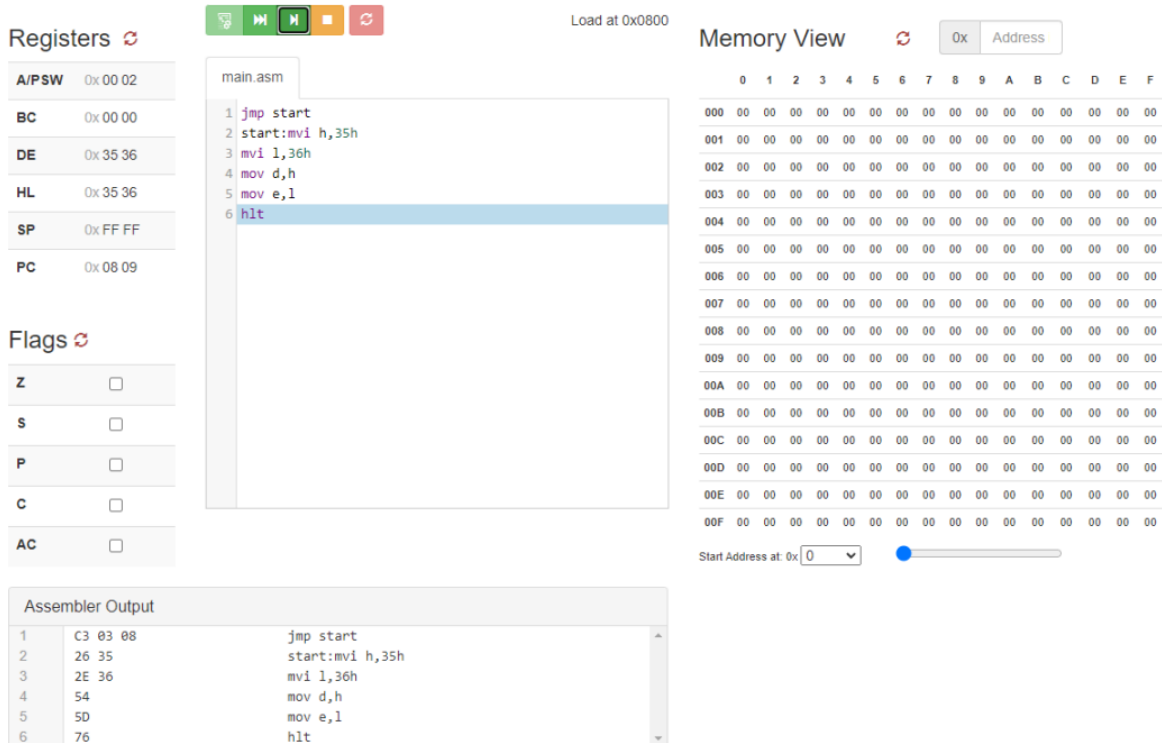
1. Write a program to load values 35H and 36H in register H and L then copy in register D and E.

INPUT:

```

jmp start
Start:mvi h,35h
Mvi l,36h
Mov d,h
Mov e,l
hlt
    
```

OUTPUT:



The screenshot displays an 8085 assembler simulator interface. On the left, the **Registers** window shows the state of the 8085 registers: A/PSW (0x0002), BC (0x0000), DE (0x3536), HL (0x3536), SP (0xFFFF), and PC (0x0809). Below this, the **Flags** window shows Z, S, P, C, and AC flags, all of which are currently cleared. The central area shows the assembly code for `main.asm`:

```

1 jmp start
2 start:mvi h,35h
3 mvi l,36h
4 mov d,h
5 mov e,l
6 hlt
    
```

On the right, the **Memory View** shows the memory contents starting from address 0x0800. The memory is filled with zeros. At the bottom, the **Assembler Output** window shows the assembly code with its corresponding machine code:

Line	Machine Code	Assembly Code
1	C3 03 08	jmp start
2	26 35	start:mvi h,35h
3	2E 36	mvi l,36h
4	54	mov d,h
5	5D	mov e,l
6	hlt	

2. write a program to copy register with value 2FH then copy register A to B and B to E.

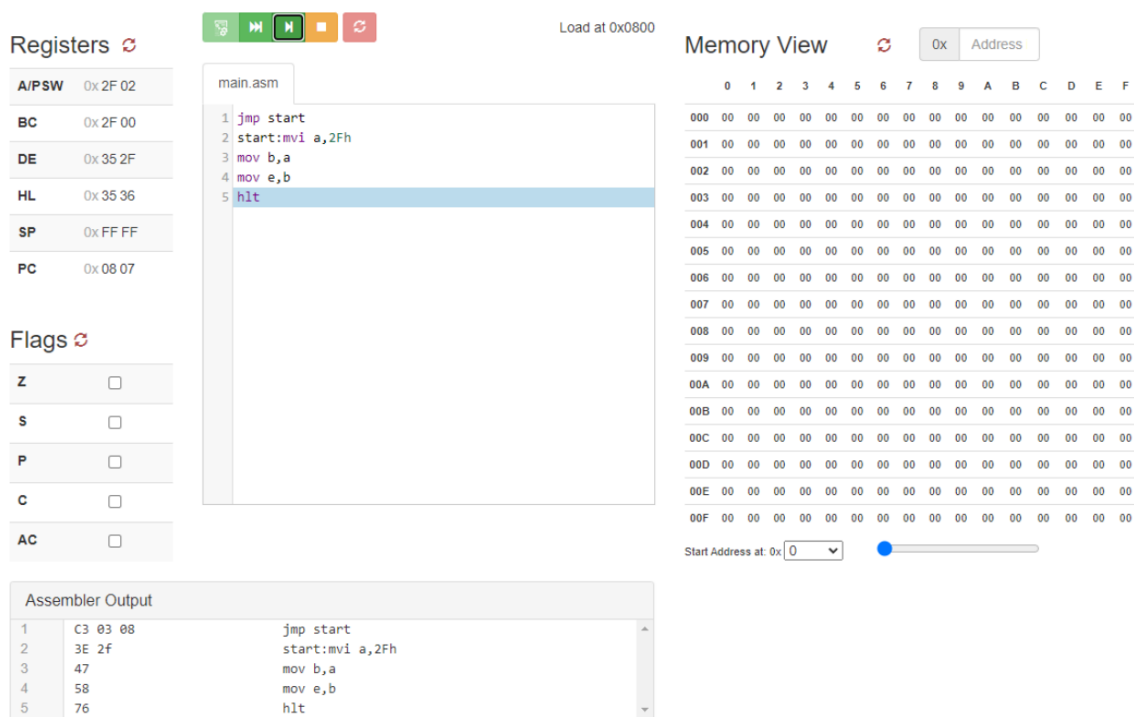
INPUT:

```

jmp start
Start:mvi a,2Fh
Mov b,a
Mov e,b
Hlt

```

OUTPUT:



Registers

A/PSW	0x2F02
BC	0x2F00
DE	0x352F
HL	0x3536
SP	0xFFFF
PC	0x0807

Flags

Z	<input type="checkbox"/>
S	<input type="checkbox"/>
P	<input type="checkbox"/>
C	<input type="checkbox"/>
AC	<input type="checkbox"/>

main.asm

```

1 jmp start
2 start:mvi a,2Fh
3 mov b,a
4 mov e,b
5 hlt

```

Memory View

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
001	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
002	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
005	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
006	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
007	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
008	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
009	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Start Address at: 0x0

Assembler Output

1	C3 03 08	jmp start
2	3E 2f	start:mvi a,2Fh
3	47	mov b,a
4	58	mov e,b
5	76	hlt

Practical-2

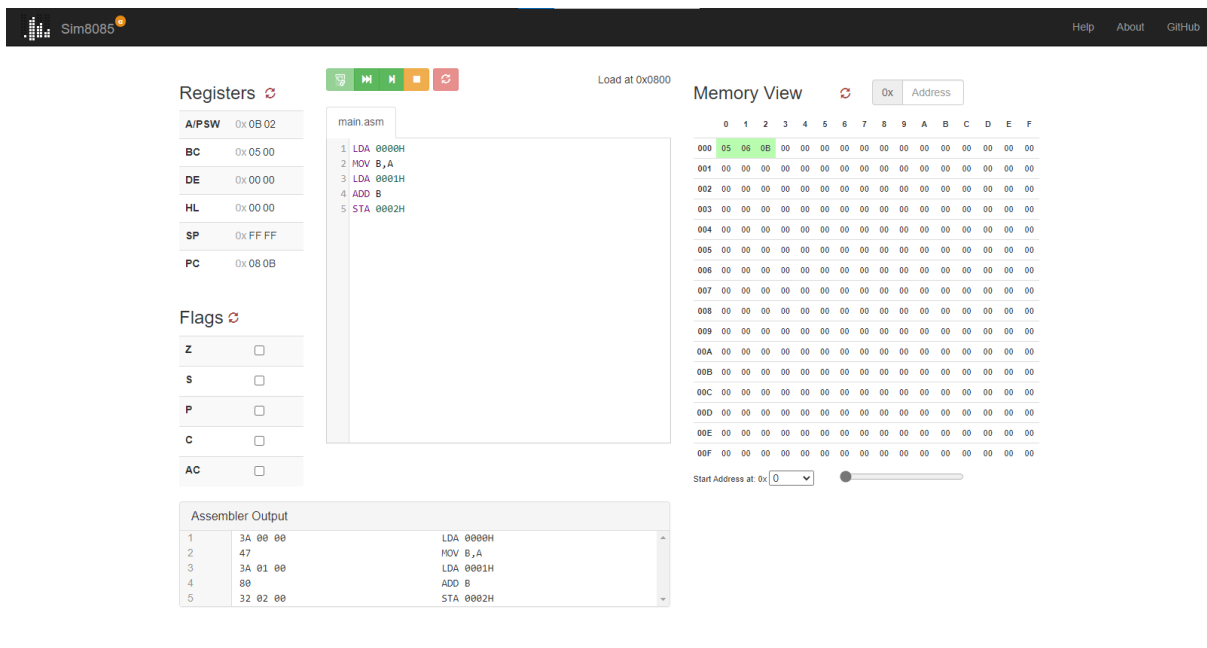
1. Objective : Addition of two 8-bit numbers.

INPUT:

```

LDA 0000H
MOV B,A
LDA 0001H
ADD B
STA 0002H
  
```

OUTPUT:



The screenshot displays the Sim8085 microprocessor simulator interface. The main window is divided into several sections:

- Registers:** Shows the current state of the registers. A/PSW is 0x0B02, BC is 0x0500, DE is 0x0000, HL is 0x0000, SP is 0xFFFF, and PC is 0x080B.
- Flags:** Shows the status of the flags. Z, S, P, C, and AC are all set to 0.
- Assembler Output:** Displays the assembly code being executed:


```

1  3A 00 00  LDA 0000H
2  47        MOV B,A
3  3A 01 00  LDA 0001H
4  80        ADD B
5  32 02 00  STA 0002H
      
```
- Memory View:** Shows the memory contents. The address 0000 is highlighted, and the value 05 is shown in the B register. The memory address 0002 is also highlighted, showing the result of the addition (02).

Conclusion:

→ In this practical, we learn about the LDA,MOV,ADD and STA instruction. Moreover how to add two numbers and store it in a different memory location.

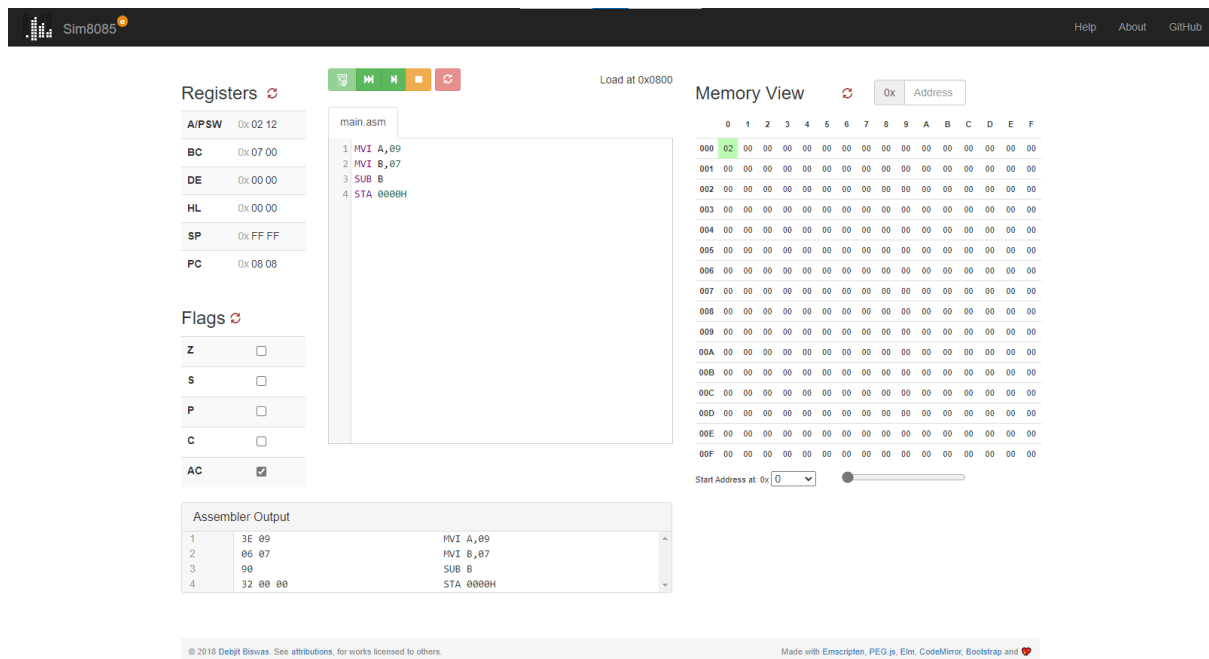
2. Objective: Subtraction of two 8-bit numbers.

INPUT:

```

MVI A,09
MVI B,07
SUB B
STA 0000H
    
```

OUTPUT:



Registers

A/PSW	0x 02 12
BC	0x 07 00
DE	0x 00 00
HL	0x 00 00
SP	0x FF FF
PC	0x 08 08

Flags

Z	<input type="checkbox"/>
S	<input type="checkbox"/>
P	<input type="checkbox"/>
C	<input type="checkbox"/>
AC	<input checked="" type="checkbox"/>

main.asm

```

1 MVI A,09
2 MVI B,07
3 SUB B
4 STA 0000H
    
```

Memory View

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
001	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
002	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
005	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
006	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
007	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
008	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
009	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Assembler Output

1	3E 09	MVI A,09
2	06 07	MVI B,07
3	90	SUB B
4	32 00 00	STA 0000H

Conclusion:

→ In this practical, we learn how to subtract two numbers stored in memory location. We also learn how to store the result at a new memory location.

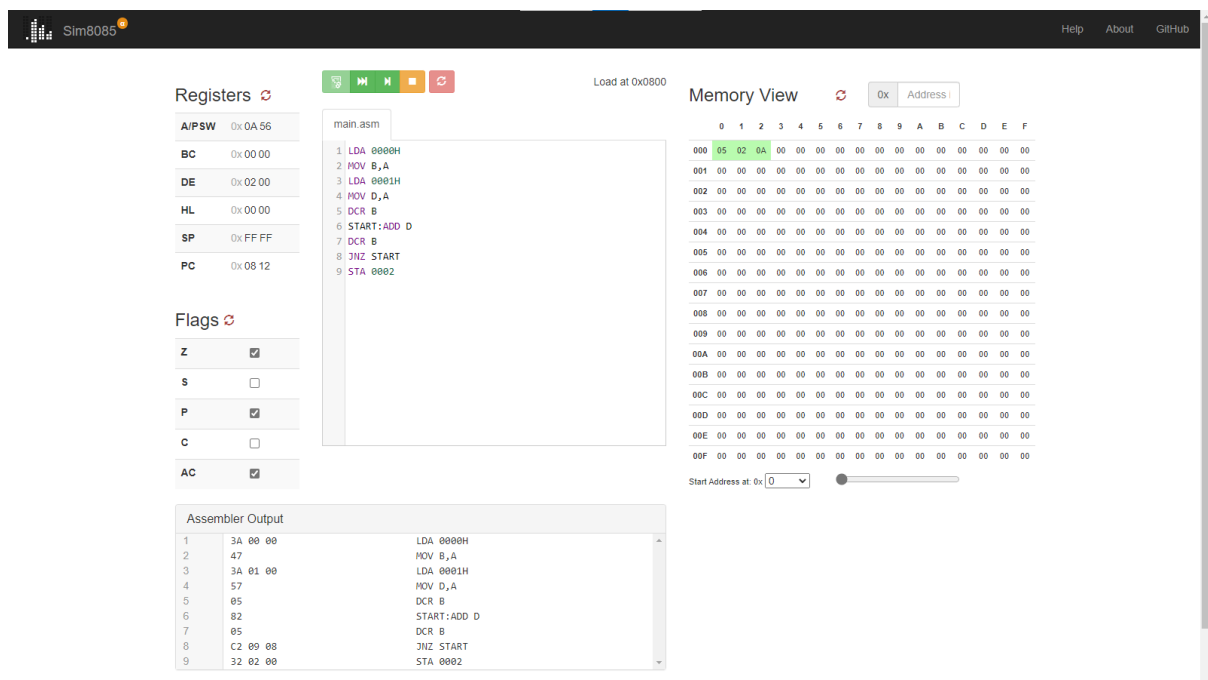
3. Objective: Multiplication of two 8- bit nos. using repeated Addition.

INPUT:

```

LDA 0000H
MOV B,A
LDA 0001H
MOV D,A
DCR B
START:ADD D
DCR B
JNZ START
STA 0002
  
```

OUTPUT:



The screenshot displays the 8085 simulator interface. The Registers window shows the following values: A/PSW: 0x0A56, BC: 0x0000, DE: 0x0200, HL: 0x0000, SP: 0xFFFF, PC: 0x0812. The Flags window shows Z (checked), S (unchecked), P (checked), C (unchecked), and AC (checked). The Memory View shows the program code starting at address 0000. The Assembler Output window shows the assembly code being executed:

Line	Address	Code	Instruction
1	3A 00 00	LDA 0000H	
2	47	MOV B,A	
3	3A 01 00	LDA 0001H	
4	57	MOV D,A	
5	05	DCR B	
6	82	START:ADD D	
7	05	DCR B	
8	C2 09 08	JNZ START	
9	32 02 00	STA 0002	

Conclusion:

In this practical we learn multiplication of two 8-bit numbers using LDA and MOV instruction.

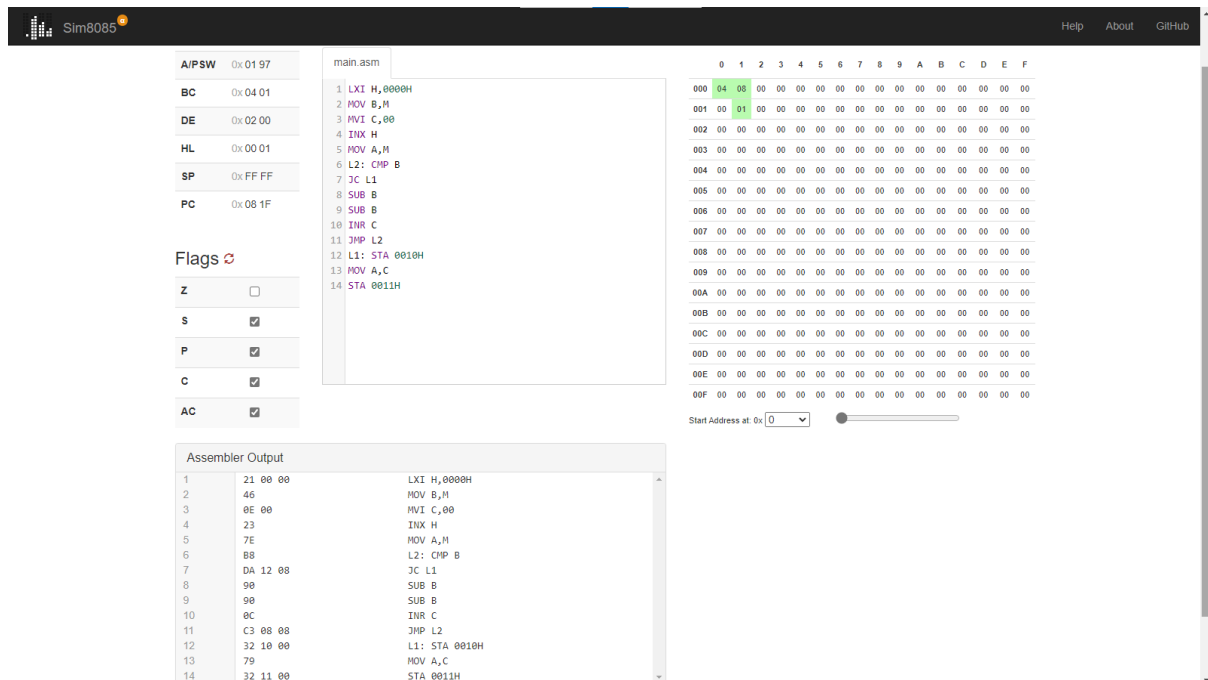
4. Objective: Division of two 8- bit nos. using repeated Subtraction

INPUT:

```

LXI H,0000H
MOV B,M
MVI C,00
INX H
MOV A,M
L2: CMP B
JC L1
SUB B
SUB B
INR C
JMP L2
L1: STA 0010H
MOV A,C
STA 0011H
  
```

OUTPUT:



The screenshot displays the Sim8085 microprocessor simulator interface. On the left, the register window shows the status of A/PSW, BC, DE, HL, SP, and PC. The main.asm code editor contains the assembly program. The memory dump on the right shows the contents of memory locations from 0000 to 00FF. The assembler output window at the bottom shows the translated machine code for each instruction.

Register	Value
A/PSW	0x0197
BC	0x0401
DE	0x0200
HL	0x0001
SP	0xFF FF
PC	0x081F

```

main.asm
1 LXI H,0000H
2 MOV B,M
3 MVI C,00
4 INX H
5 MOV A,M
6 L2: CMP B
7 JC L1
8 SUB B
9 SUB B
10 INR C
11 JMP L2
12 L1: STA 0010H
13 MOV A,C
14 STA 0011H
  
```

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
001	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
002	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
005	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
006	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
007	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
008	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
009	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Line	Address	Machine Code	Instruction
1	21 00 00	LXI H,0000H	
2	46	MOV B,M	
3	0E 00	MVI C,00	
4	23	INX H	
5	7E	MOV A,M	
6	B8	L2: CMP B	
7	DA 12 08	JC L1	
8	9B	SUB B	
9	9B	SUB B	
10	0C	INR C	
11	C3 08 08	JMP L2	
12	32 10 00	L1: STA 0010H	
13	79	MOV A,C	
14	32 11 00	STA 0011H	

Conclusion:

→ In this practical, we learn how to do the division operation using multiple subtraction.

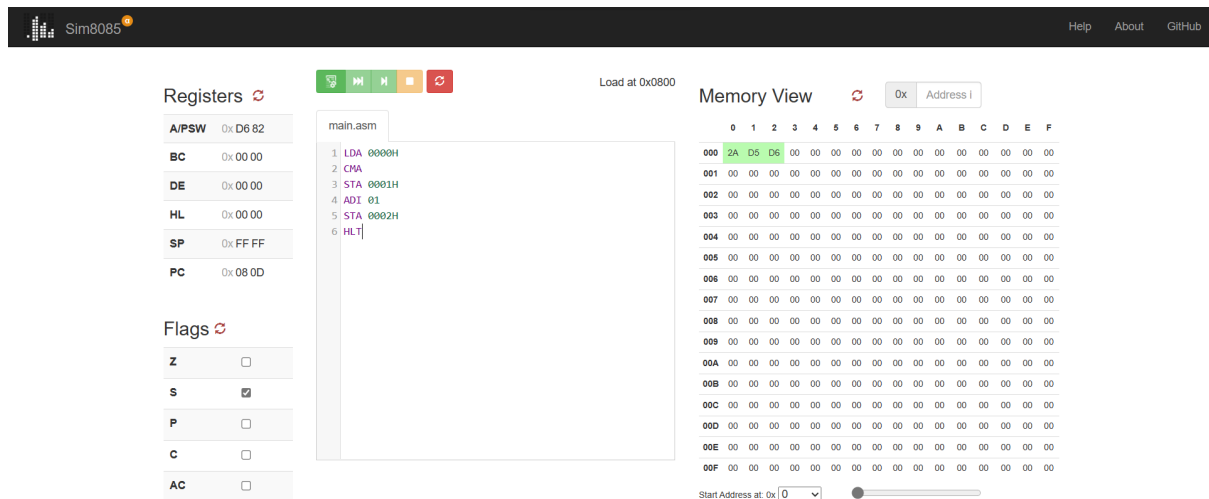
5. Objective: Find 1's & 2's complement of an 8 bit number.

INPUT:

```

LDA 0000H
CMA
STA 0001H
ADI 01
STA 0002H
HLT
    
```

OUTPUT:



The screenshot displays the Sim8085 microprocessor simulator interface. The top bar shows the simulator name 'Sim8085' and navigation links. The main window is divided into several sections:

- Registers:** A table showing the current values of the 8085 registers.

Register	Value
A/PSW	0x D6 82
BC	0x 00 00
DE	0x 00 00
HL	0x 00 00
SP	0x FFFF
PC	0x 08 0D
- Flags:** A section showing the status of the Z, S, P, C, and AC flags.

Flag	Status
Z	<input type="checkbox"/>
S	<input checked="" type="checkbox"/>
P	<input type="checkbox"/>
C	<input type="checkbox"/>
AC	<input type="checkbox"/>
- Assembly Code:** A window showing the loaded assembly code 'main.asm'.


```

1 LDA 0000H
2 CMA
3 STA 0001H
4 ADI 01
5 STA 0002H
6 HLT
            
```
- Memory View:** A window showing the memory contents. The address 0x0000 is highlighted, showing the value 0x2A. The memory view is organized in a grid with columns for hexadecimal addresses (0-15) and rows for memory locations (000-00F).

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	2A	D5	D6	00	00	00	00	00	00	00	00	00	00	00	00	00
001	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
002	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
005	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
006	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
007	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
008	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
009	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

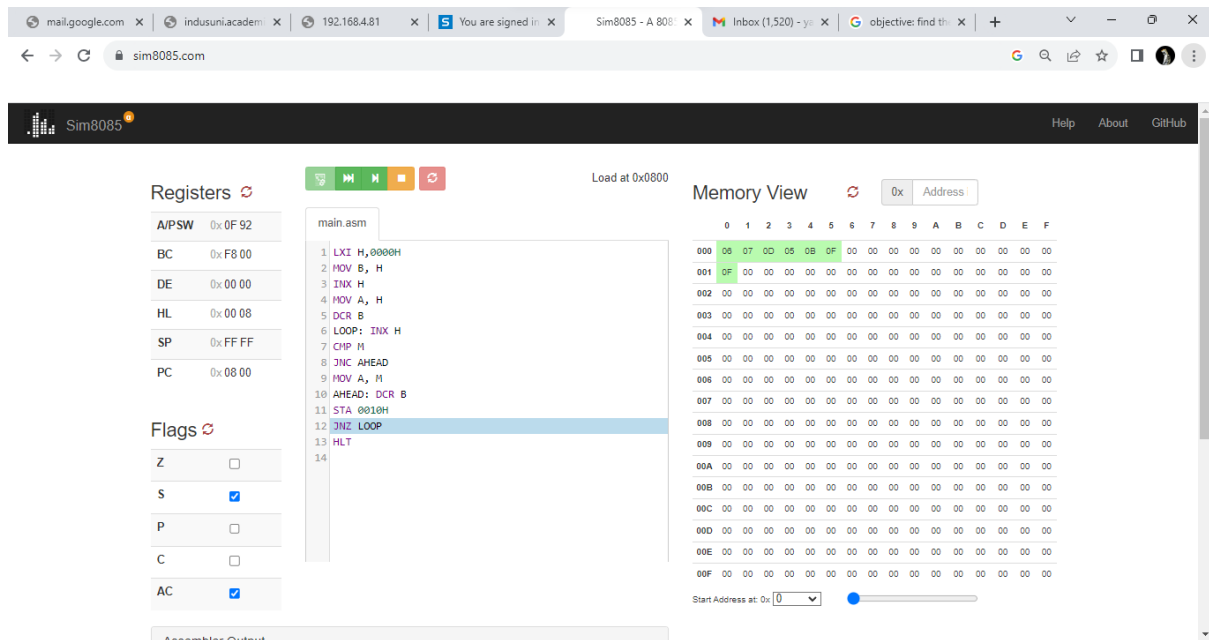
6. Objective: Find largest Number From an array.

INPUT:

```

LXI H,0000H
MOV B, H
INX H
MOV A, H
DCR B
LOOP: INX H
CMP M
JNC AHEAD
MOV A, M
AHEAD: DCR B
STA 0010H
JNZ LOOP
HLT
  
```

OUTPUT:



The screenshot displays the Sim8085 emulator interface. On the left, the 'Registers' window shows the following values: A/PSW: 0x0F92, BC: 0xF800, DE: 0x0000, HL: 0x0008, SP: 0xFFFF, and PC: 0x0800. Below the registers, the 'Flags' window shows the status of various flags: Z (Zero) is unchecked, S (Sign) is checked, P (Parity) is unchecked, C (Carry) is unchecked, and AC (Auxiliary Carry) is checked. The central area shows the assembly code being executed, with the current instruction 'JNZ LOOP' highlighted at address 0012. The 'Memory View' on the right shows the data stored in memory, with the array data at addresses 0000 to 000F. The instruction list on the left shows the code being executed, with the current instruction being 'JNZ LOOP' at address 0012.

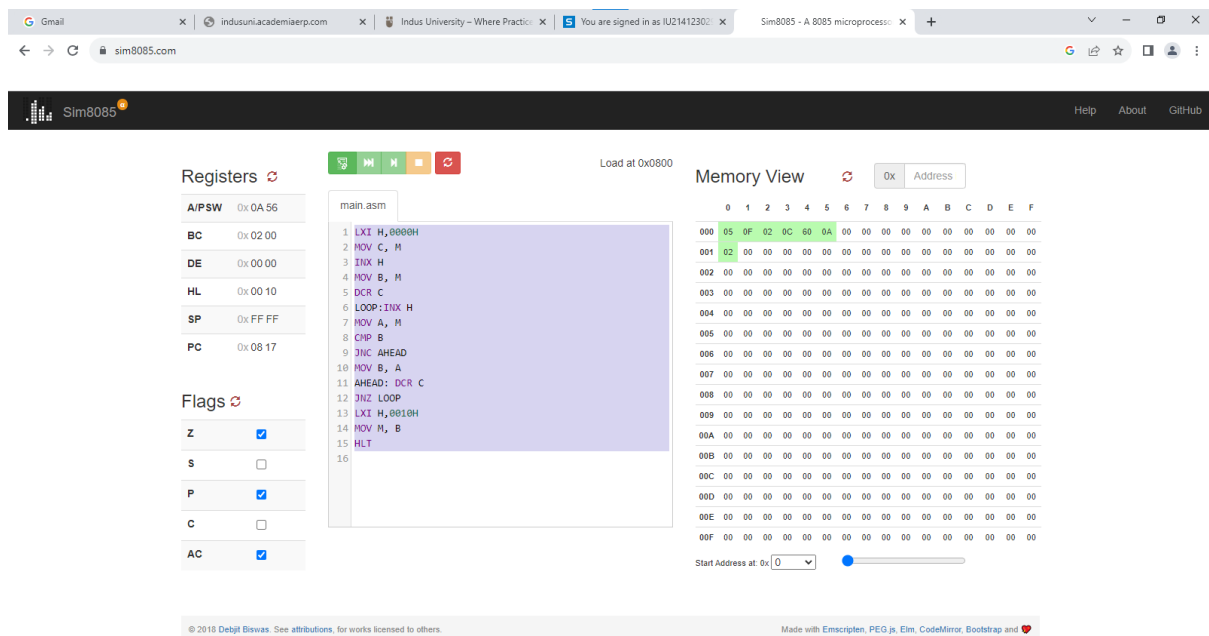
7. Objective: Find smallest No. from an array.

INPUT:

```

LXI H,0000H
MOV C, M
INX H
MOV B, M
DCR C
LOOP:INX H
MOV A, M
CMP B
JNC AHEAD
MOV B, A
AHEAD: DCR C
JNZ LOOP
LXI H,0010H
MOV M, B
HLT
    
```

OUTPUT:



The screenshot displays the Sim8085 microprocessor simulator interface. The top bar shows the browser tabs and the simulator title "Sim8085 - A 8085 microprocessor". The main interface is divided into several sections:

- Registers:** A table showing the current values of the 8085 registers.

Register	Value
A/PSW	0x0A56
BC	0x0200
DE	0x0000
HL	0x0010
SP	0xFFFF
PC	0x0817
- Flags:** A section showing the status of the Z, S, P, C, and AC flags.

Flag	Status
Z	<input checked="" type="checkbox"/>
S	<input type="checkbox"/>
P	<input checked="" type="checkbox"/>
C	<input type="checkbox"/>
AC	<input checked="" type="checkbox"/>
- Memory View:** A table showing the memory contents at various addresses. The address 0x0005 is highlighted in green.

Address	Value
000	05 0F 02 0C 60 0A
001	02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
002	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
003	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
006	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
007	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
008	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
009	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00A	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00D	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00E	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00F	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

The bottom of the interface shows the copyright notice: "© 2019 Debit Browser. See attributions, for works licensed to others." and the text "Made with Emscripten, PEG.js, Elm, CodeMirror, Bootstrap and ❤️".

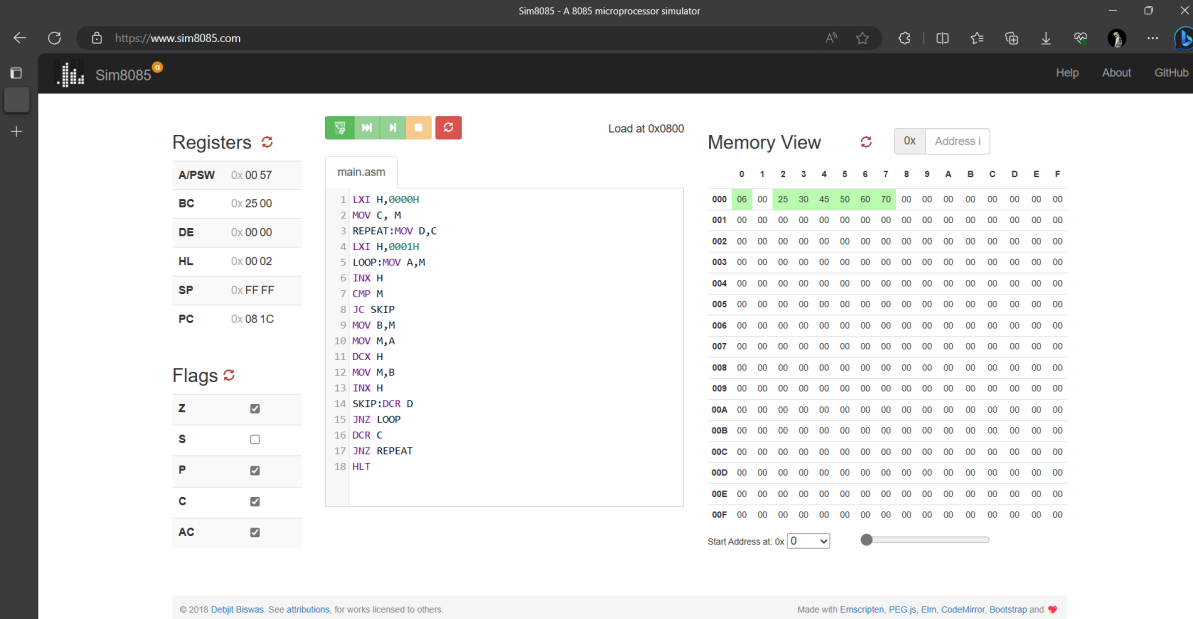
8. Objective: Arrange data bytes in ascending order.

INPUT:

```

LXI H,0000H
MOV C, M
REPEAT:MOV D,C
LXI H,0001H
LOOP:MOV A,M
INX H
CMP M
JC SKIP
MOV B,M
MOV M,A
DCX H
MOV M,B
INX H
SKIP:DCR D
JNZ LOOP
DCR C
JNZ REPEAT
HLT
  
```

OUTPUT:



The screenshot displays the Sim8085 microprocessor simulator interface. The main window is divided into several sections:

- Registers:** Shows the current state of the 8085 registers. A/PSW is 0x0057, BC is 0x2500, DE is 0x0000, HL is 0x0002, SP is 0xFFFF, and PC is 0x081C.
- Flags:** Shows the status of the Z, S, P, C, and AC flags. Z is checked, S is unchecked, P is checked, C is checked, and AC is checked.
- Memory View:** Displays the memory contents starting from address 0x0000. The first few bytes are 06, 25, 30, 45, 50, 60, 70, and the rest are 00.
- Assembly Code:** Shows the assembly code being executed, with line numbers 1 through 18. The code is:


```

1 LXI H,0000H
2 MOV C, M
3 REPEAT:MOV D,C
4 LXI H,0001H
5 LOOP:MOV A,M
6 INX H
7 CMP M
8 JC SKIP
9 MOV B,M
10 MOV M,A
11 DCX H
12 MOV M,B
13 INX H
14 SKIP:DCR D
15 JNZ LOOP
16 DCR C
17 JNZ REPEAT
18 HLT
      
```

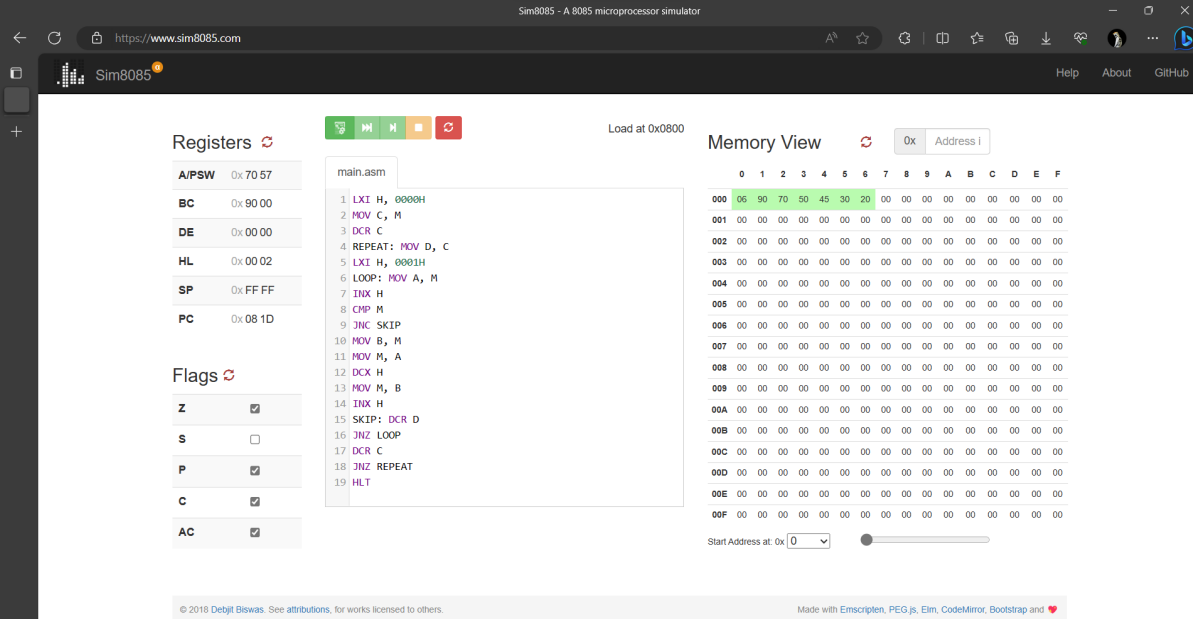
9. Objective: Arrange data bytes in descending order.

INPUT:

```

LXI H,0000H
MOV C, M
REPEAT:MOV D,C
LXI H,0001H
LOOP:MOV A,M
INX H
CMP M
JC SKIP
MOV B,M
MOV M,A
DCX H
MOV M,B
INX H
SKIP:DCR D
JNZ LOOP
DCR C
JNZ REPEAT
HLT
  
```

OUTPUT:



The screenshot displays the Sim8085 microprocessor simulator interface. The main window is divided into several sections:

- Registers:** Shows the current state of registers. A/PSW is 0x7057, BC is 0x9000, DE is 0x0000, HL is 0x0002, SP is 0xFFFF, and PC is 0x081D.
- Flags:** Shows the status of flags. Z (Zero) is checked, S (Sign) is unchecked, P (Parity) is checked, C (Carry) is checked, and AC (Auxiliary Carry) is checked.
- Memory View:** Displays a table of memory addresses and their corresponding data. The data is arranged in descending order: 06, 90, 70, 50, 45, 30, 20, followed by zeros.
- Assembly Code:** Shows the assembly code being executed, with line numbers 1 through 19. The code includes instructions like LXI, MOV, DCR, INX, CMP, JC, and HLT.

❖ Introduction to MASM:

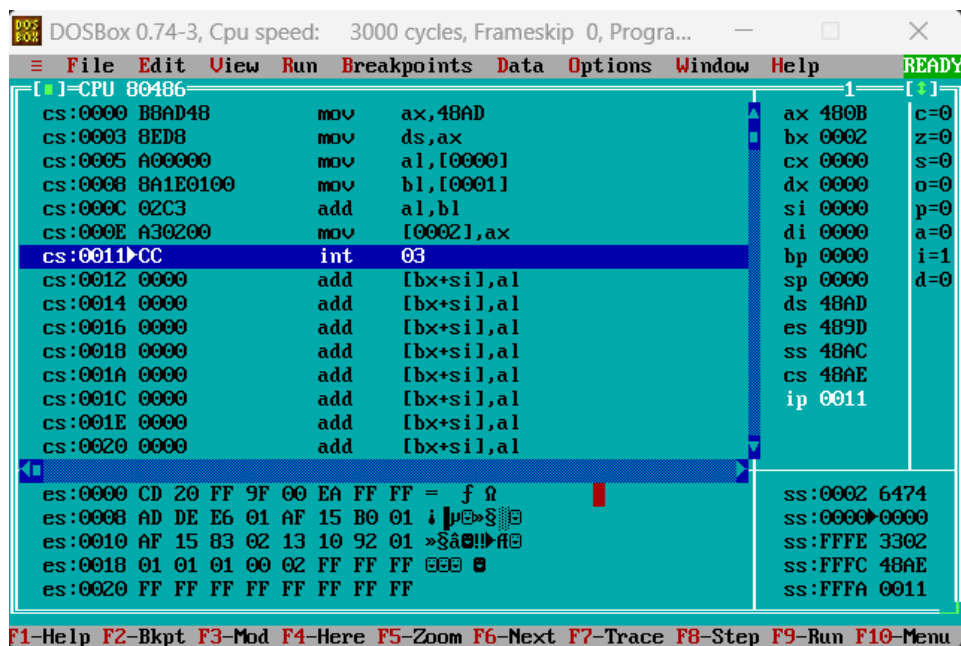
1. A) TO PERFORM ADDITION OPERATION ON 8-BIT DATA

INPUT:

```
data segment
a db 09h
b db 02h
c dw ?
data ends
```

```
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov al,a
mov bl,b
add al,bl
mov c,ax
int 3
code ends
end start
```

OUTPUT:



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

Address	Instruction	Comment
cs:0000	B8AD48	mov ax,48AD
cs:0003	8ED8	mov ds,ax
cs:0005	A00000	mov al,[0000]
cs:0008	8A1E0100	mov bl,[0001]
cs:000C	02C3	add al,bl
cs:000E	A30200	mov [0002],ax
cs:0011	CC	int 03
cs:0012	0000	add [bx+si],al
cs:0014	0000	add [bx+si],al
cs:0016	0000	add [bx+si],al
cs:0018	0000	add [bx+si],al
cs:001A	0000	add [bx+si],al
cs:001C	0000	add [bx+si],al
cs:001E	0000	add [bx+si],al
cs:0020	0000	add [bx+si],al

es:0000 CD 20 FF 9F 00 EA FF FF = f 0
 es:0008 AD DE E6 01 AF 15 B0 01 i p 0 0 0 0
 es:0010 AF 15 83 02 13 10 92 01 » 8 A 0 1 1 1 1 1
 es:0018 01 01 01 00 02 FF FF FF 0 0 0 0
 es:0020 FF FF FF FF FF FF FF FF

ax 480B c=0
 bx 0002 z=0
 cx 0000 s=0
 dx 0000 o=0
 si 0000 p=0
 di 0000 a=0
 bp 0000 i=1
 sp 0000 d=0
 ds 48AD
 es 489D
 ss 48AC
 cs 48AE
 ip 0011

ss:0002 6474
 ss:0000 0000
 ss:FFFE 3302
 ss:FFFC 48AE
 ss:FFFA 0011

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

1. B) TO PERFORM SUBTRACTION OPERATION ON 8-BIT DATA.

INPUT:

```

data segment
a db 2Ah
b db 13h
c dw ?
data ends

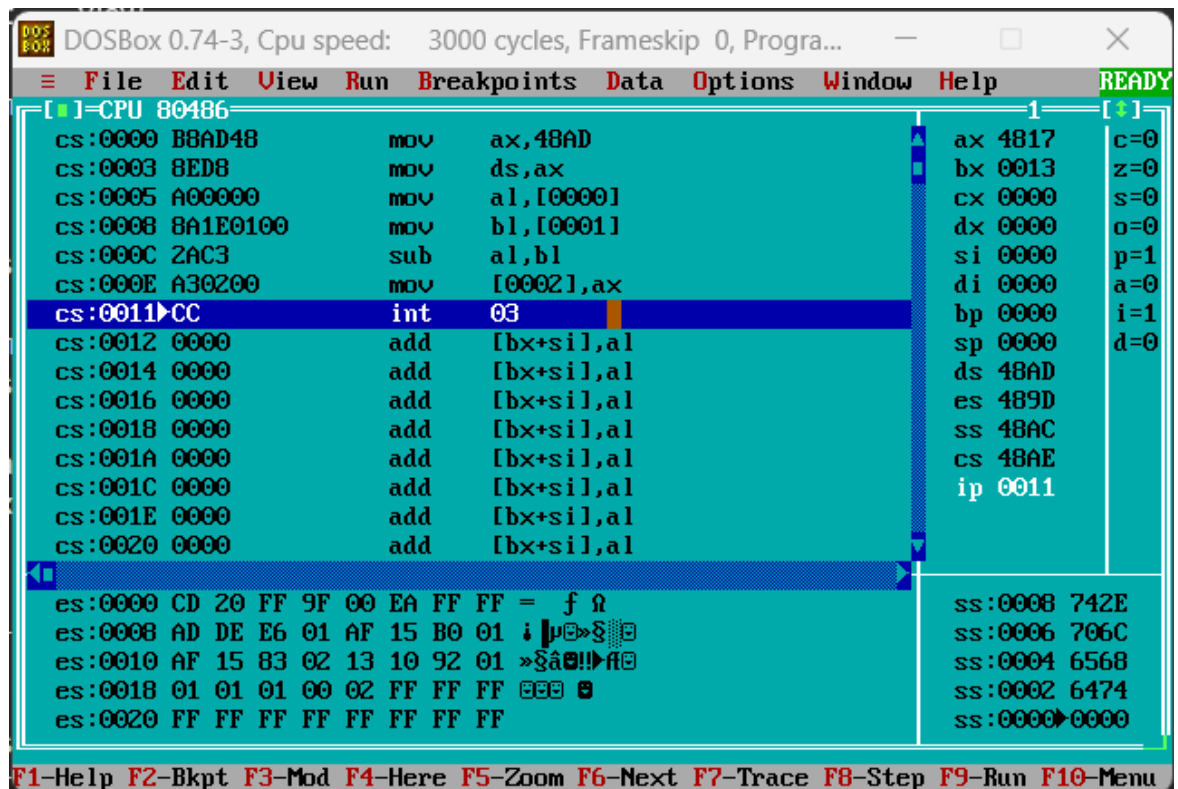
```

```

code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov al,a
mov bl,b
sub al,bl
mov c,ax
int 3
code ends
end start

```

OUTPUT:



```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
File Edit View Run Breakpoints Data Options Window Help
[ ] CPU 80486
cs:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 A00000 mov al,[0000]
cs:0008 8A1E0100 mov bl,[0001]
cs:000C 2AC3 sub al,bl
cs:000E A30200 mov [0002],ax
cs:0011 CC int 03
cs:0012 0000 add [bx+si],al
cs:0014 0000 add [bx+si],al
cs:0016 0000 add [bx+si],al
cs:0018 0000 add [bx+si],al
cs:001A 0000 add [bx+si],al
cs:001C 0000 add [bx+si],al
cs:001E 0000 add [bx+si],al
cs:0020 0000 add [bx+si],al
ax 4817 c=0
bx 0013 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=1
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
ds 48AD
es 489D
ss 48AC
cs 48AE
ip 0011
es:0000 CD 20 FF 9F 00 EA FF FF = f 0
es:0008 AD DE E6 01 AF 15 B0 01 : 01 02 03 04
es:0010 AF 15 83 02 13 10 92 01 » 05 06 07 08
es:0018 01 01 01 00 02 FF FF FF 09 0A 0B 0C
es:0020 FF FF FF FF FF FF FF FF 0D 0E 0F 10
ss:0008 742E
ss:0006 706C
ss:0004 6568
ss:0002 6474
ss:0000 0000
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

1. C) TO PERFORM MULTIPLICATION OPERATION ON 8-BIT DATA.

INPUT:

```

data segment
a db 09h
b db 02h
c dw ?
data ends

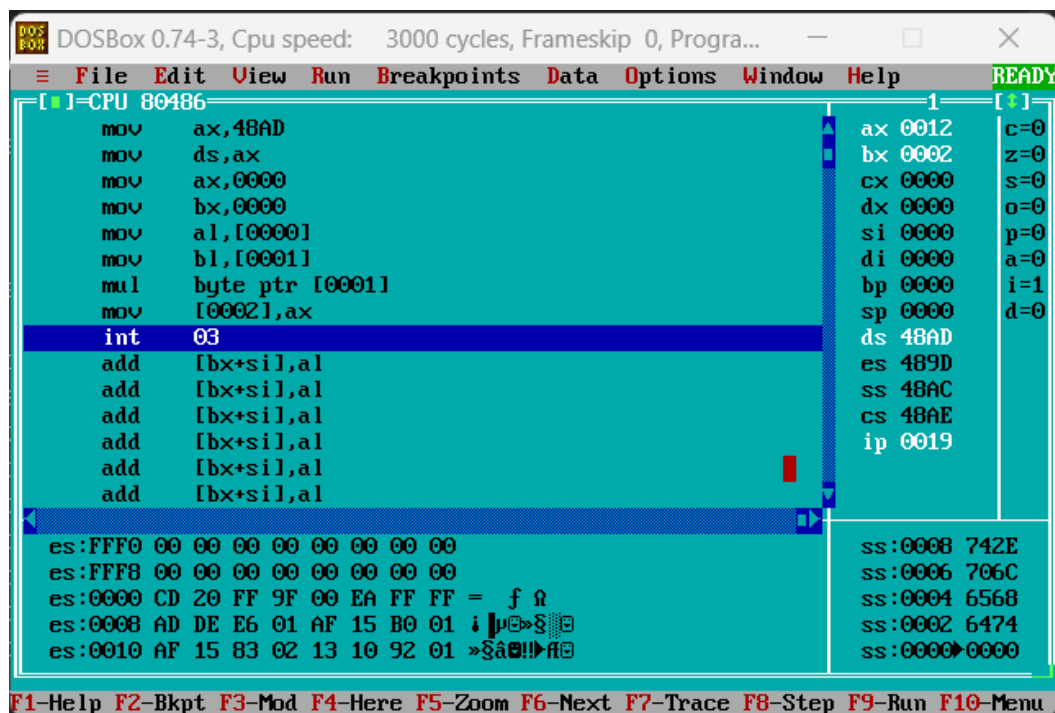
```

```

code segment
assume cs:code, ds:data
start:
mov ax,data
mov ds,ax
mov ax,0000h
mov bx,0000h
mov al,a
mov bl,b
mul b
mov c,ax
int 3
code ends
end start

```

OUTPUT:



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

File Edit View Run Breakpoints Data Options Window Help

[]-CPU 80486

```

mov ax,48AD
mov ds,ax
mov ax,0000
mov bx,0000
mov al,[0000]
mov bl,[0001]
mul byte ptr [0001]
mov [0002],ax
int 03
add [bx+si],al
add [bx+si],al
add [bx+si],al
add [bx+si],al
add [bx+si],al
add [bx+si],al

```

ax 0012 c=0
bx 0002 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
ds 48AD
es 489D
ss 48AC
cs 48AE
ip 0019

es:FFF0 00 00 00 00 00 00 00 00
es:FFF8 00 00 00 00 00 00 00 00
es:0000 CD 20 FF 9F 00 EA FF FF = f Ω
es:0008 AD DE E6 01 AF 15 B0 01 i |µ|>§|
es:0010 AF 15 83 02 13 10 92 01 »§â|!|ff

ss:0008 742E
ss:0006 706C
ss:0004 6568
ss:0002 6474
ss:0000 0000

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

1. D) TO PERFORM DIVISION OPERATION ON 8-BIT DATA.

INPUT:

```

data segment
a db 28h
b db 02h
c dw ?
data ends

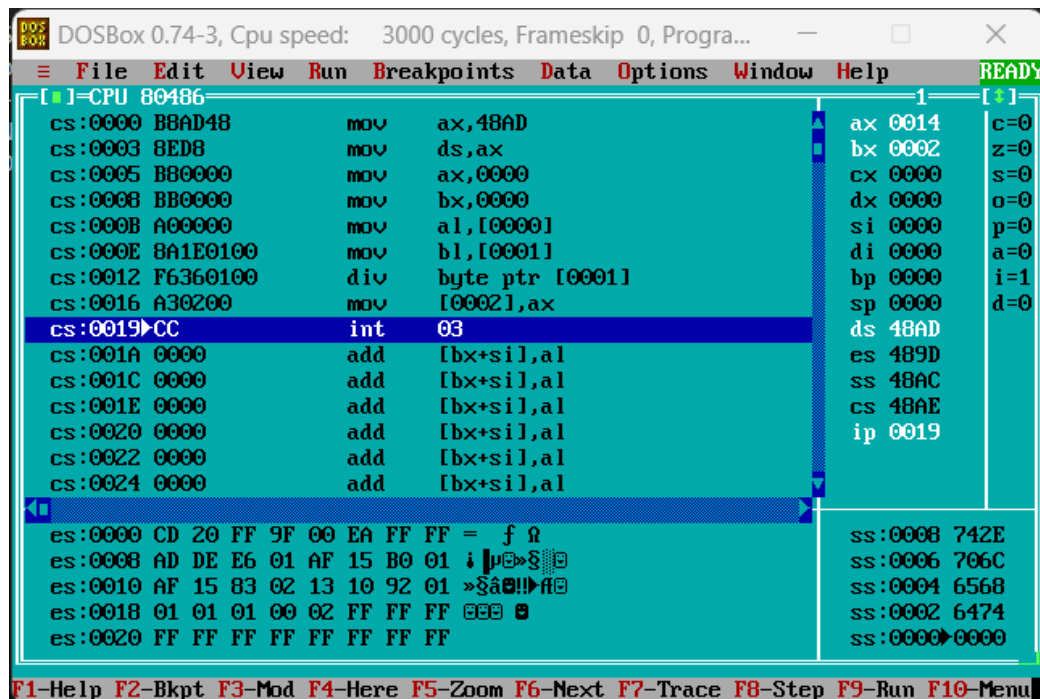
```

```

code segment
assume cs:code, ds:data
start:
mov ax,data
mov ds,ax
mov ax,0000h
mov bx,0000h
mov al,a
mov bl,b
div b
mov c,ax
int 3
code ends
end start

```

OUTPUT:



```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
File Edit View Run Breakpoints Data Options Window Help
[1]-CPU 80486
cs:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 B80000 mov ax,0000
cs:0008 BB0000 mov bx,0000
cs:000B A00000 mov al,[0000]
cs:000E 8A1E0100 mov bl,[0001]
cs:0012 F6360100 div byte ptr [0001]
cs:0016 A30200 mov [0002],ax
cs:0019 CC int 03
cs:001A 0000 add [bx+si],al
cs:001C 0000 add [bx+si],al
cs:001E 0000 add [bx+si],al
cs:0020 0000 add [bx+si],al
cs:0022 0000 add [bx+si],al
cs:0024 0000 add [bx+si],al
es:0000 CD 20 FF 9F 00 EA FF FF = f
es:0008 AD DE E6 01 AF 15 B0 01 i
es:0010 AF 15 83 02 13 10 92 01 »
es:0018 01 01 01 00 02 FF FF FF
es:0020 FF FF FF FF FF FF FF FF
ax 0014 c=0
bx 0002 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
ds 48AD
es 489D
ss 48AC
cs 48AE
ip 0019
ss:0008 742E
ss:0006 706C
ss:0004 6568
ss:0002 6474
ss:0000 0000
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

2. A) TO PERFORM AN ADDITION OPERATION AN ASSEMBLY LANGUAGE ON 16-BIT NUMBER.

INPUT:

```
Code segment
Assume CS: code
Start :
Mov ax,0000h
Mov bx, ax
Mov dx, ax
Mov si,3000h
Mov ax ,[si]
Inc si
Inc si
Mov bx, [si]
Inc si
Inc si
Add ax,bx
Mov [si], ax
Jc l1
Inc si
Inc si
Mov [si], dx
Int 3
L1: inc dx
Inc si
Inc si
Mov [si], dx
Int 3
Code ends
End start
```


OUTPUT:

```

CPU 80486
cs:000D 46      inc    si
cs:000E 8B1C    mov    bx,[si]
cs:0010 46      inc    si
cs:0011 46      inc    si
cs:0012 03C3    add    ax,bx
cs:0014 8904    mov    [si],ax
cs:0016 7205    jb     001D
cs:0018 46      inc    si
cs:0019 46      inc    si
cs:001A 8914    mov    [si],dx
cs:001C CC      int    03
cs:001D 42      inc    dx
cs:001E 46      inc    si

cs:0000 B8 00 00 8B D8 8B D0 BE 7  i+ i
cs:0008 00 30 8B 04 46 46 8B 1C 0i+FFi
cs:0010 46 46 03 C3 89 04 72 05 FF e+r
cs:0018 46 46 89 14 CC 42 46 46 FF e BFF

ax 0000    c=0
bx 0000    z=0
cx 0000    s=0
dx 001A    o=0
si 30B8    p=1
di 0000    a=0
bp 0000    i=1
sp 0000    d=0
ds 489D
es 489D
ss 48AC
cs 48AD
ip 0010

ss:0002 6474
ss:0000 0000

```

2. B) TO PERFORM A SUBTRACTION OPERATION ASSEMBLY LANGUAGE ON 16-BIT NUMBER.**INPUT:**

```
Code segment
Assume CS: code
Start :
Mov ax, 0000h
Mov bx, ax
Mov dx, ax
Mov si, 3000h
Mov ax, [si]
Inc si
Inc si
Mov bx, [si]
Inc si
Inc si
Sub ax, bx
Mov [si], ax
Jc l1
Inc si
Inc si
Mov [si], dx
Int 3
L1: inc dx
Inc si
Inc si
Mov [si], dx
Int 3
Code ends
End start
```

OUTPUT:

```

CPU 80486
cs:0451 9AA7017108 call 0871:01A7
cs:0456 A07558 mov al,[5875]
cs:0459 98 cbw
cs:045A 0BC0 or ax,ax
cs:045C 750C jne 046A
cs:045E 803E308200 cmp byte ptr [823]
cs:0463 7405 je 046A
cs:0465 9A3900511B call 1B51:0039
cs:046A 5E pop si
cs:046B 8BE5 mov sp,bp
cs:046D 5D pop bp
cs:046E CB retf
cs:046F 55 push bp

ax 0020 c=0
bx 0000 z=0
cx 0000 s=0
dx 2110 o=0
si 7075 p=0
di 0000 a=0
bp 0000 i=1
sp FFF0 d=0
ds 2110
es 489D
ss 48AC
cs 15C5
ip 04AB

48AD:0000 B8 00 00 8B D0 8B D8 BE 71 70 8B 04 46 46 8B 1C qp i+FF i+
48AD:0008 71 70 8B 04 46 46 8B 1C qp i+FF i+
48AD:0010 46 46 F7 F3 89 04 46 46 FF ≈ ≤ e+FF
48AD:0018 89 14 CC 01 88 04 46 E2 e 7 || 0 e+FF

ss:FFF2 489D
ss:FFF0 0000

```

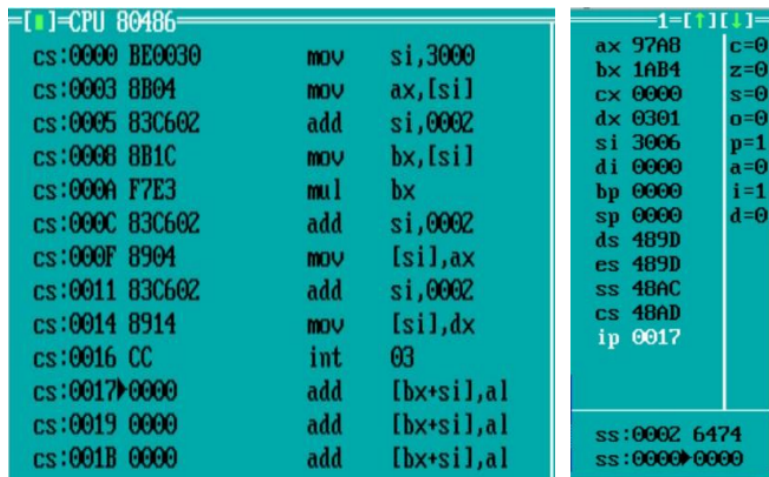
2. C) TO PERFORM A MULTIPLICATION OPERATION ASSEMBLY LANGUAGE ON 16-BIT NUMBER.

INPUT:

```

Code segment
Assume CS:code
Start :
Mov ah, 0000h
Mov si, 3000h
Mov ax, [si]
Add si, 02h
Mov bx, [si]
Mul bx
Add si, 02h
Mov [si], ax
Add si, 02h
Add [si], dx
Int 3
Code ends
End start
    
```

OUTPUT:

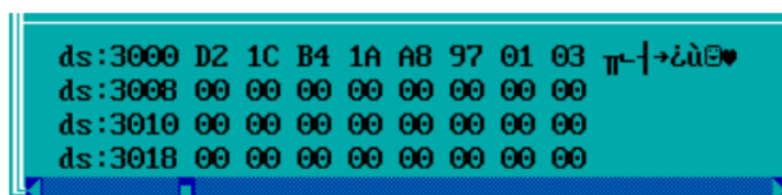


```

CPU 80486
cs:0000 BE0030      mov     si,3000
cs:0003 8B04        mov     ax,[si]
cs:0005 83C602      add     si,0002
cs:0008 8B1C        mov     bx,[si]
cs:000A F7E3        mul     bx
cs:000C 83C602      add     si,0002
cs:000F 8904        mov     [si],ax
cs:0011 83C602      add     si,0002
cs:0014 8914        mov     [si],dx
cs:0016 CC          int     03
cs:0017 0000        add     [bx+si],al
cs:0019 0000        add     [bx+si],al
cs:001B 0000        add     [bx+si],al

ax 97A8  c=0
bx 1AB4  z=0
cx 0000  s=0
dx 0301  o=0
si 3006  p=1
di 0000  a=0
bp 0000  i=1
sp 0000  d=0
ds 489D
es 489D
ss 48AC
cs 48AD
ip 0017

ss:0002 6474
ss:0000 0000
    
```



```

ds:3000 D2 1C B4 1A A8 97 01 03
ds:3008 00 00 00 00 00 00 00 00
ds:3010 00 00 00 00 00 00 00 00
ds:3018 00 00 00 00 00 00 00 00
    
```

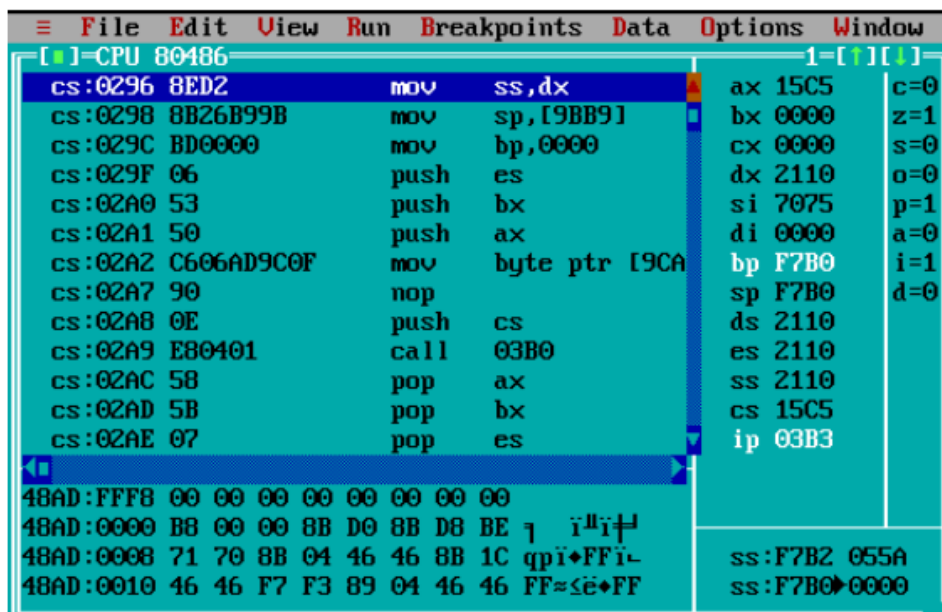
2. D) TO PERFORM A DIVISION OPERATION ASSEMBLY LANGUAGE ON 16-BIT NUMBER.

INPUT:

```

Code segment
assume CS: code
start: mov ax,0000h
      mov dx, ax
      mov bx, ax
      mov si, 7071h
      mov ax, [si]
      inc si
      inc si
      mov bx, [si]
      inc si
      inc si
      div bx
      mov [si], ax
      inc si
      inc si
      mov [si], dx
      int 3
      code ends
      end start
  
```

OUTPUT:



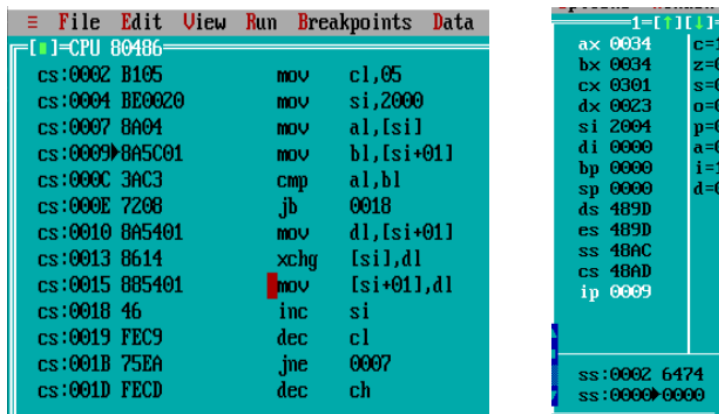
3. WRITE A PROGRAM TO ARRANGE GIVEN NUMBERS IN ASCENDING ORDER.

INPUT:

```

Code segment
Assume CS:code
Start: mov ch, 05h
L1: mov cl, 05h
Mov si, 2000h
L2: mov al, [si]
Mov bl, [si + 1]
Cmp al, bl
Jc l3
Mov dl, [si + 1]
Xchg [si], dl
L3: inc si
Dec cl
Jnz l2
Dec ch
Jnz l
Int 3
Code ends
End start
    
```

OUTPUT:



The screenshot displays a debugger interface with three main panels. The top panel shows the assembly code being executed, with instructions like 'mov cl, 05', 'mov si, 2000', 'mov al, [si]', 'mov bl, [si+01]', 'cmp al, bl', 'jb 0018', 'mov dl, [si+01]', 'xchg [si], dl', 'mov [si+01], dl', 'inc si', 'dec cl', 'jne 0007', and 'dec ch'. The middle panel shows the state of the registers, including ax, bx, cx, dx, si, di, bp, sp, ds, es, ss, cs, and ip. The bottom panel shows the memory contents at addresses ds:2000, ds:2008, ds:2010, and ds:2018.

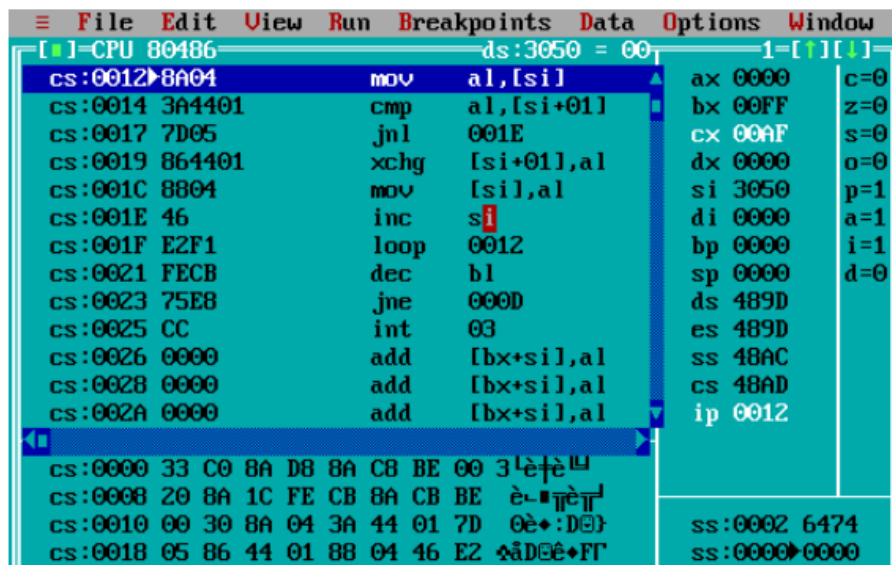
4. WRITE A PROGRAM TO ARRANGE GIVEN NUMBERS IN DESCENDING ORDER.

INPUT:

```

Code segment
Assume CS: code
Start: xor ax , ax
Mov bl, al
Mov cl, al
Mov si, 2000h
Mov bl, [si]
Dec bl
L3: mov cl, bl
Mov si, 3000h
L2: mov al, [si]
Cmp al, [si +1]
Jge l1
Xchg al, [si +1]
Mov [si], al
L1: inc si
Loop L2
Dec bl
Jnz L3
Int 3
Code ends
End start
    
```

OUTPUT:



The screenshot shows an assembly editor window with the following content:

Address	Code	Register/Value
cs:0012	mov al,[si]	ax 0000
cs:0014	cmp al,[si+01]	bx 00FF
cs:0017	jnl 001E	cx 00AF
cs:0019	xchg [si+01],al	dx 0000
cs:001C	mov [si],al	si 3050
cs:001E	inc si	di 0000
cs:001F	loop 0012	bp 0000
cs:0021	dec bl	sp 0000
cs:0023	jne 000D	ds 489D
cs:0025	int 03	es 489D
cs:0026	add [bx+si],al	ss 48AC
cs:0028	add [bx+si],al	cs 48AD
cs:002A	add [bx+si],al	ip 0012

At the bottom, the memory dump shows:

```

cs:0000 33 C0 8A D8 8A CB BE 00 31 2E 14
cs:0008 20 8A 1C FE CB 8A CB BE 00 1E 14
cs:0010 00 30 8A 04 3A 44 01 7D 0E 0D 00
cs:0018 05 86 44 01 88 04 46 E2 0A 0E 0F
    
```

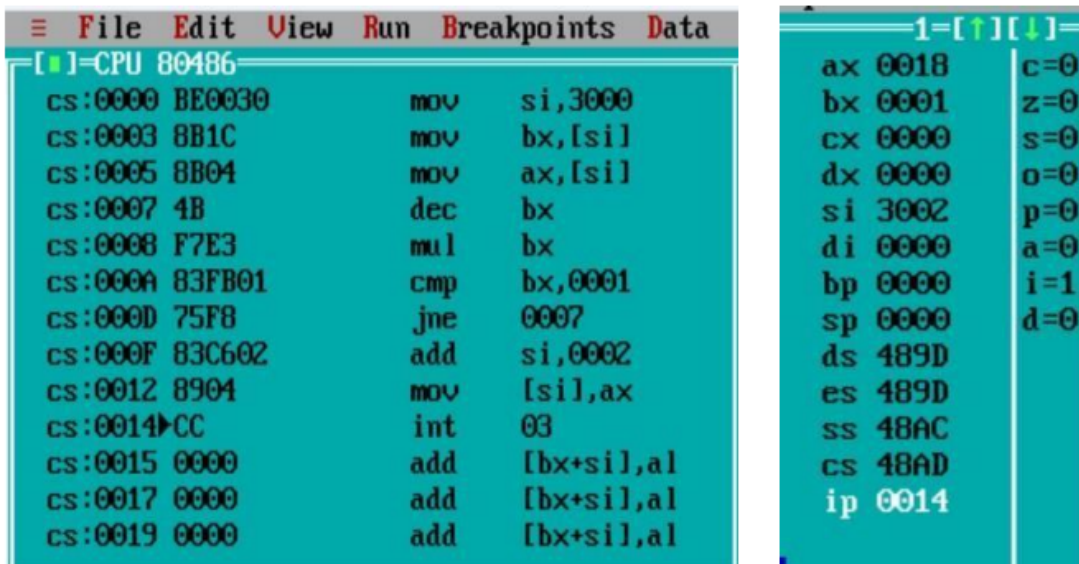
5. WRITE AN ASSEMBLY LANGUAGE PROGRAM TO FIND THE FACTORIAL OF GIVEN NUMBER.

INPUT:

```

Code segment
Assume CS: code
Start:
Mov si, 3000h
Mov bx, [si]
Mov ax, [si]
L1: dec bx
Mul bx
Cmp bx, 01h
Jnz l1
Add si, ax
Int 3
Code ends
End start
  
```

OUTPUT:

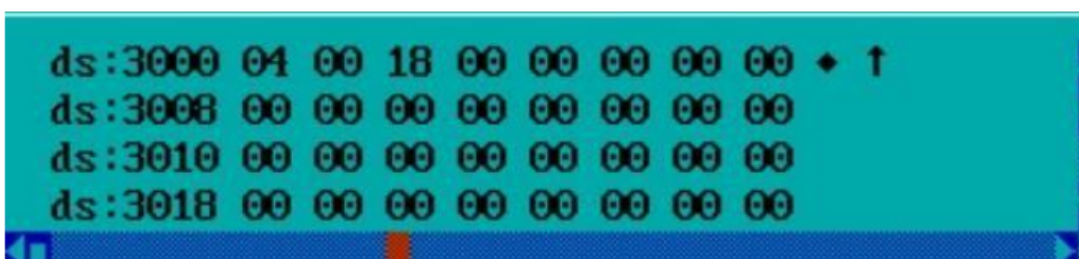


The screenshot shows the 8086 assembly editor with the following code loaded:

Address	Code
cs:0000	BE0030 mov si,3000
cs:0003	8B1C mov bx,[si]
cs:0005	8B04 mov ax,[si]
cs:0007	4B dec bx
cs:0008	F7E3 mul bx
cs:000A	83FB01 cmp bx,0001
cs:000D	75F8 jne 0007
cs:000F	83C602 add si,0002
cs:0012	8904 mov [si],ax
cs:0014	CC int 03
cs:0015	0000 add [bx+si],al
cs:0017	0000 add [bx+si],al
cs:0019	0000 add [bx+si],al

The register window on the right shows the following values:

Register	Value
ax	0018
bx	0001
cx	0000
dx	0000
si	3002
di	0000
bp	0000
sp	0000
ds	489D
es	489D
ss	48AC
cs	48AD
ip	0014



The screenshot shows the 8086 memory window with the following data loaded:

Address	Data
ds:3000	04 00 18 00 00 00 00 00
ds:3008	00 00 00 00 00 00 00 00
ds:3010	00 00 00 00 00 00 00 00
ds:3018	00 00 00 00 00 00 00 00