Experiment - 1a

Aim: Write a program to add two 16-bit numbers with/ without carry using 8086.

CODE

```
MOV CX, 0000H
MOV AX, [3000H]
MOV BX, [3002H]
ADD AX,BX
JNC ; A
INC CX
A : MOV [3004H], AX
MOV [3006H],CL
HLT
```

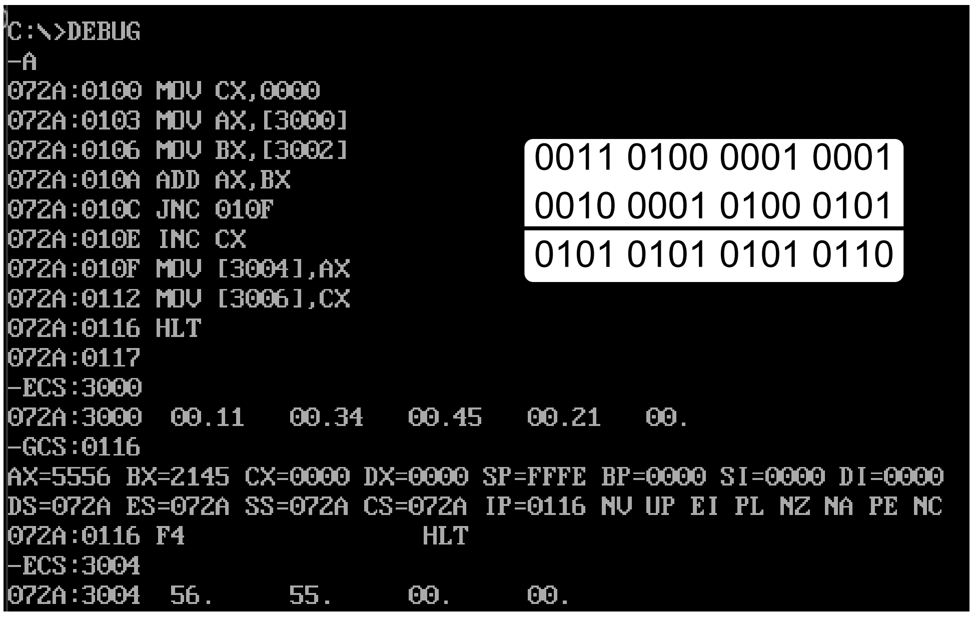
DATA

3000 : 11	3001 : 34	3002 : 45	3003 : 21

AX, 3411H 0011 0100 0001 0001
BX, 2145H 0010 0001 0100 0101
ADD AX,BX;

RESULT

AX: 5556H	CX: 0000H		
3004 : 55	3005 : 55	3006 : 00	3007 : 00



```
C:\>DEBUG
-A
072A:0100 MOV CX,0000
072A:0103 MOV AX,[3000]
072A:0106 MOV BX,[3002]
072A:010A ADD AX,BX
072A:010C JNC 010F
072A:010E INC CX
072A:010F MOV [3004],AX
072A:0112 MOV [3006],CX
072A:0116 HLT
072A:0117
-ECS:3000
072A:3000  00.11   00.34   00.45   00.21   00.
-GCS:0116
AX=5556 BX=2145 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0116 NV UP EI PL NZ NA PE NC
072A:0116 F4            HLT
-ECS:3004
072A:3004  56.     55.     00.     00.
```

```
0011 0100 0001 0001
0010 0001 0100 0101
0101 0101 0101 0110
```

Experiment - 1b

Aim: Write a program to subtract two 16-bit numbers with/ without carry using 8086.

CODE

```
MOV CX, 0000H
MOV AX, [3000H]
MOV BX, [3002H]
SUB AX,BX ; Jump if no borrow
JNC ; A
INC CX
A : MOV [3004H], AX
MOV [3006H],CL
HLT
```

DATA

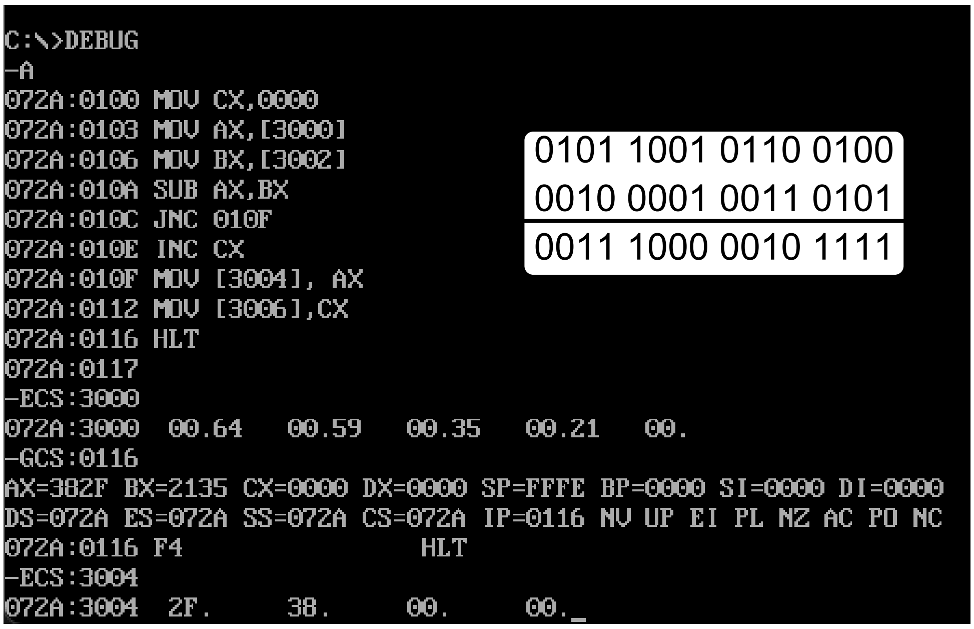
3000 : 64	3001 : 59	3002 : 35	3003 : 21

```
AX, 5964H  0101 1001 0110 0100
BX, 2135H  0010 0001 0011 0101
SUB AX,BX;
```

RESULT

AX: 382FH   CX: 0000H

3004 : 2F	3005 : 38	3006 : 00	3007 : 00



```
C:\>DEBUG
-A
072A:0100 MOV CX,0000
072A:0103 MOV AX,[3000]
072A:0106 MOV BX,[3002]
072A:010A SUB AX,BX
072A:010C JNC 010F
072A:010E INC CX
072A:010F MOV [3004], AX
072A:0112 MOV [3006],CX
072A:0116 HLT
072A:0117
-ECS:3000
072A:3000  00.64   00.59   00.35   00.21   00.
-GCS:0116
AX=382F BX=2135 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0116 NV UP EI PL NZ AC PO NC
072A:0116 F4            HLT
-ECS:3004
072A:3004  2F.    38.    00.    00._
```

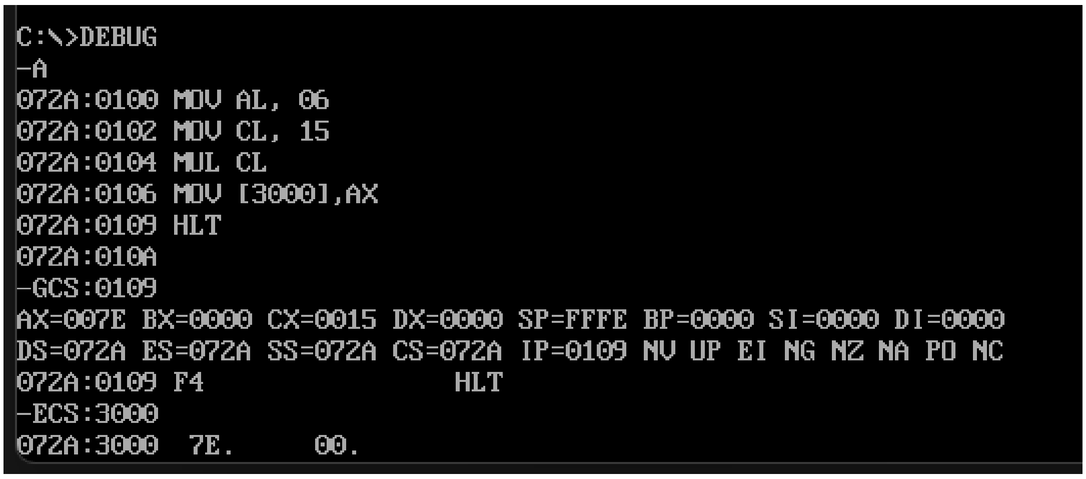
```
0101 1001 0110 0100
0010 0001 0011 0101
0011 1000 0010 1111
```

Experiment - 2a

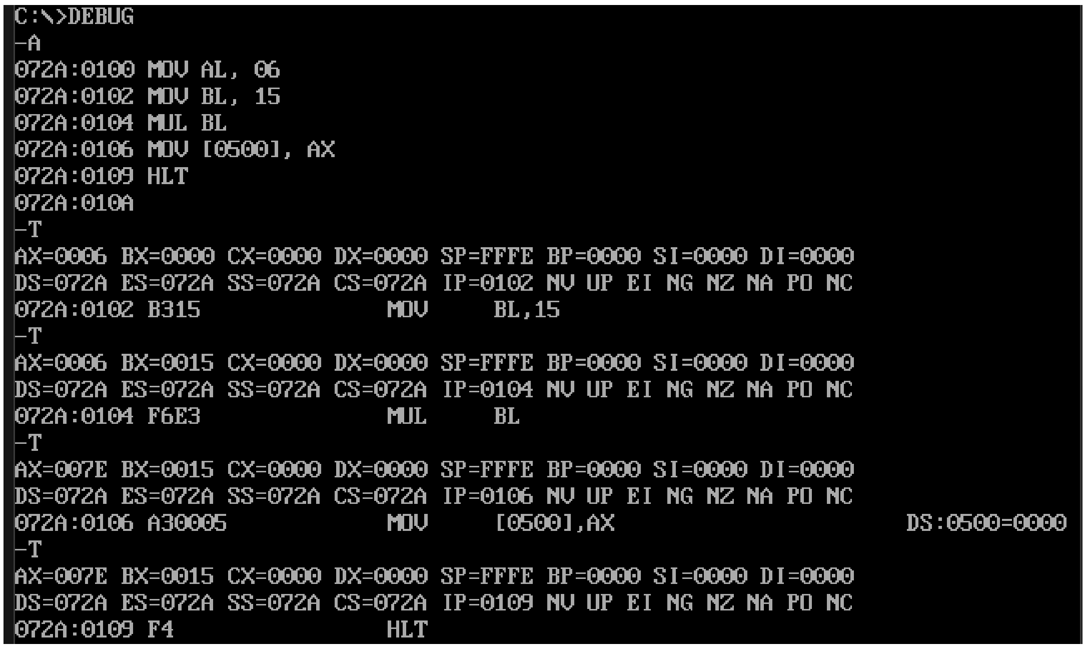
Aim: Write a program to multiply two 8 bit numbers using 8086.

CODE

```
MOV AL, 06H
MOV BL, 15H
MUL BL ; AX=AL*BL
MOV [3000H], AX
HLT
```



```
C:\>DEBUG
-A
072A:0100 MOV AL, 06
072A:0102 MOV CL, 15
072A:0104 MUL CL
072A:0106 MOV [3000],AX
072A:0109 HLT
072A:010A
-GCS:0109
AX=007E BX=0000 CX=0015 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0109 NV UP EI NG NZ NA PO NC
072A:0109 F4                  HLT
-ECS:3000
072A:3000  7E.     00.
```



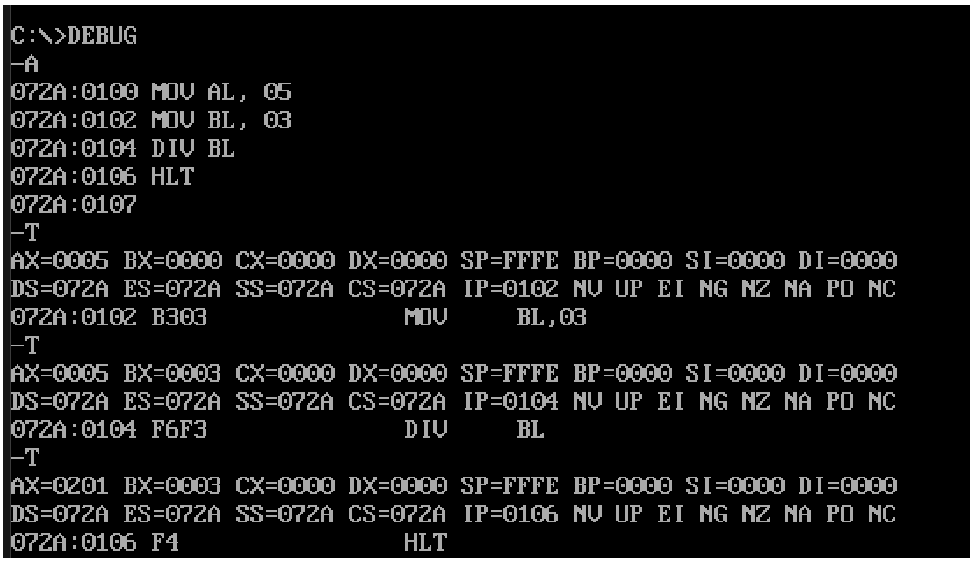
```
C:\>DEBUG
-A
072A:0100 MOV AL, 06
072A:0102 MOV BL, 15
072A:0104 MUL BL
072A:0106 MOV [0500], AX
072A:0109 HLT
072A:010A
-T
AX=0006 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0102 NV UP EI NG NZ NA PO NC
072A:0102 B315              MOV     BL,15
-T
AX=0006 BX=0015 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0104 NV UP EI NG NZ NA PO NC
072A:0104 F6E3              MUL     BL
-T
AX=007E BX=0015 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0106 NV UP EI NG NZ NA PO NC
072A:0106 A30005            MOV     [0500],AX                      DS:0500=0000
-T
AX=007E BX=0015 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0109 NV UP EI NG NZ NA PO NC
072A:0109 F4                HLT
```

Experiment - 2b

Aim: a) Write a program to divide two 8 bit numbers using 8086.

CODE

MOV AL, 05H
MOV BL, 03H
DIV BL ; After division, the instruction stores quotient in AL and the remainder in AH register.
MOV [0500H], AX
HLT

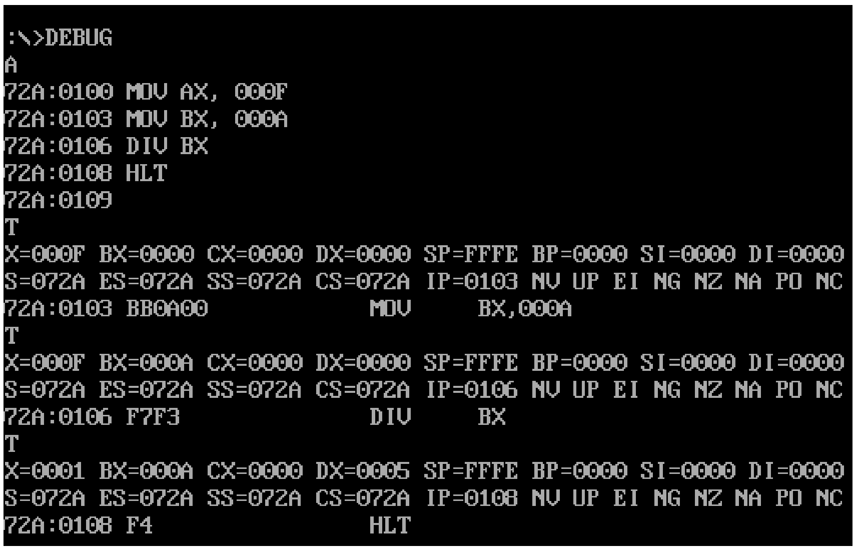


```
C:\>DEBUG
-A
072A:0100 MOV AL, 05
072A:0102 MOV BL, 03
072A:0104 DIV BL
072A:0106 HLT
072A:0107
-T
AX=0005 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0102 NV UP EI NG NZ NA PO NC
072A:0102 B303              MOV     BL,03
-T
AX=0005 BX=0003 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0104 NV UP EI NG NZ NA PO NC
072A:0104 F6F3              DIV     BL
-T
AX=0201 BX=0003 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0106 NV UP EI NG NZ NA PO NC
072A:0106 F4                HLT
```

Aim: b) Write a program to divide two 16 bit numbers using 8086.

CODE

```
MOV AX,000FH; In this case, the AX register holds the numerator.
MOV BX,000AH
DIV BX;   After division, the quotient is stored in the AX register and
the remainder goes to the DX register.
MOV [0500H], AX
HLT
```



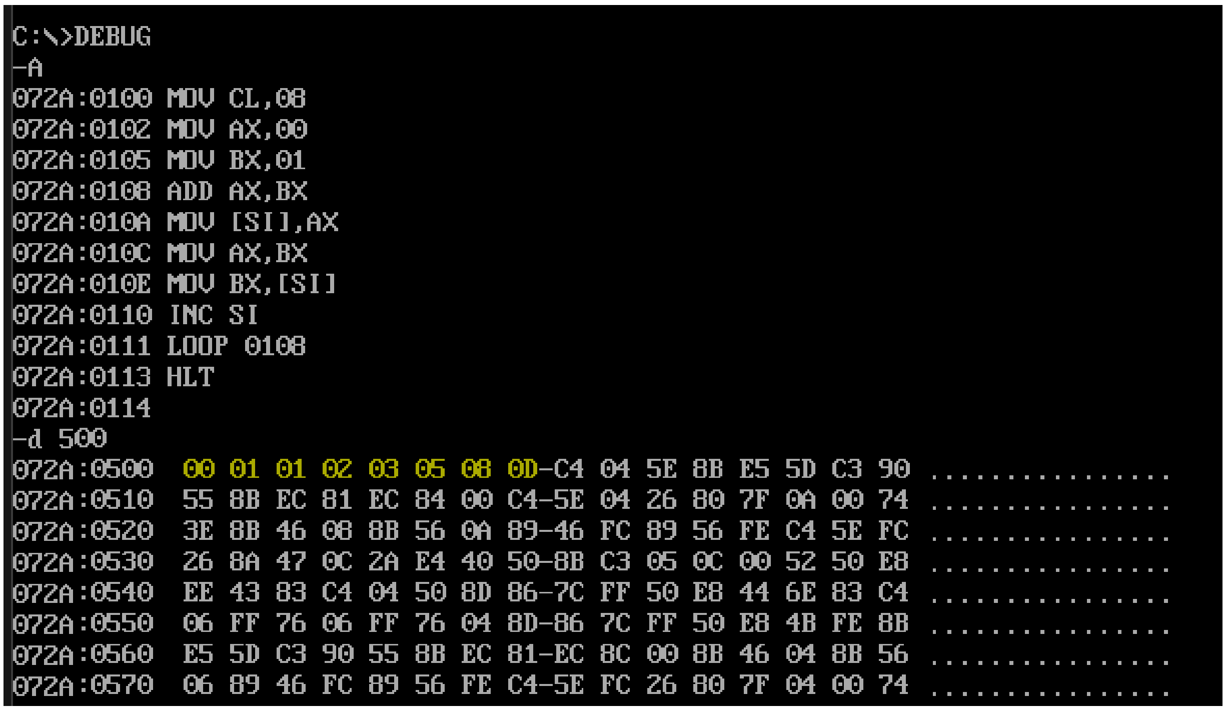
```
:\>DEBUG
A
72A:0100 MOV AX, 000F
72A:0103 MOV BX, 000A
72A:0106 DIV BX
72A:0108 HLT
72A:0109
T
X=000F BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
S=072A ES=072A SS=072A CS=072A IP=0103 NV UP EI NG NZ NA PO NC
72A:0103 BB0A00            MOV     BX,000A
T
X=000F BX=000A CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
S=072A ES=072A SS=072A CS=072A IP=0106 NV UP EI NG NZ NA PO NC
72A:0106 F7F3              DIV     BX
T
X=0001 BX=000A CX=0000 DX=0005 SP=FFFE BP=0000 SI=0000 DI=0000
S=072A ES=072A SS=072A CS=072A IP=0108 NV UP EI NG NZ NA PO NC
72A:0108 F4                HLT
```

Experiment - 3

Aim: Write a Program to generate Fibonacci series.

CODE

```
MOV CL,08H; Load the count value for CL for looping
MOV AX,00H; Default No
MOV BX,01H; Default No
L1: ADD AX,BX
      MOV [SI],AX
      MOV AX,BX
      MOV BX,[SI]
      INC SI
      LOOP L1
HLT
```



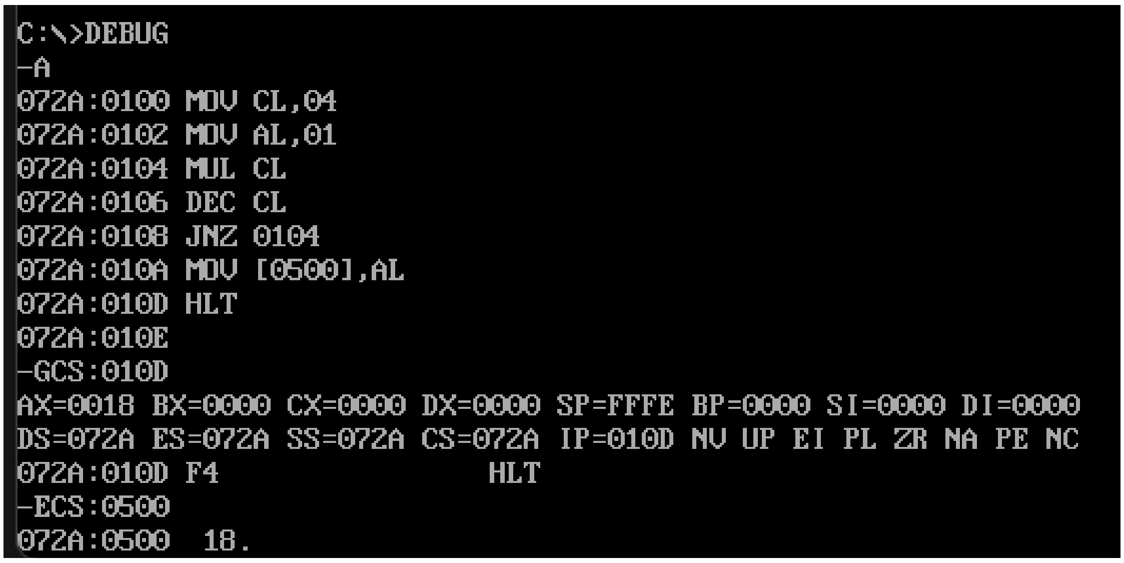
```
C:\>DEBUG
-A
072A:0100 MOV CL,08
072A:0102 MOV AX,00
072A:0105 MOV BX,01
072A:0108 ADD AX,BX
072A:010A MOV [SI],AX
072A:010C MOV AX,BX
072A:010E MOV BX,[SI]
072A:0110 INC SI
072A:0111 LOOP 0108
072A:0113 HLT
072A:0114
-d 500
072A:0500  00 01 01 02 03 05 08 0D-C4 04 5E 8B E5 5D C3 90  ................
072A:0510  55 8B EC 81 EC 84 00 C4-5E 04 26 80 7F 0A 00 74  ................
072A:0520  3E 8B 46 08 8B 56 0A 89-46 FC 89 56 FE C4 5E FC  ................
072A:0530  26 8A 47 0C 2A E4 40 50-8B C3 05 0C 00 52 50 E8  ................
072A:0540  EE 43 83 C4 04 50 8D 86-7C FF 50 E8 44 6E 83 C4  ................
072A:0550  06 FF 76 06 FF 76 04 8D-86 7C FF 50 E8 4B FE 8B  ................
072A:0560  E5 5D C3 90 55 8B EC 81-EC 8C 00 8B 46 04 8B 56  ................
072A:0570  06 89 46 FC 89 56 FE C4-5E FC 26 80 7F 04 00 74  ................
```

Experiment - 4a

Aim: Write a Program to generate Factorial of a number.

CODE

```
MOV CL,04H
MOV AL,01H
A: MUL CL
DEC CL
JNZ A
MOV [0500H], AL
HLT
```



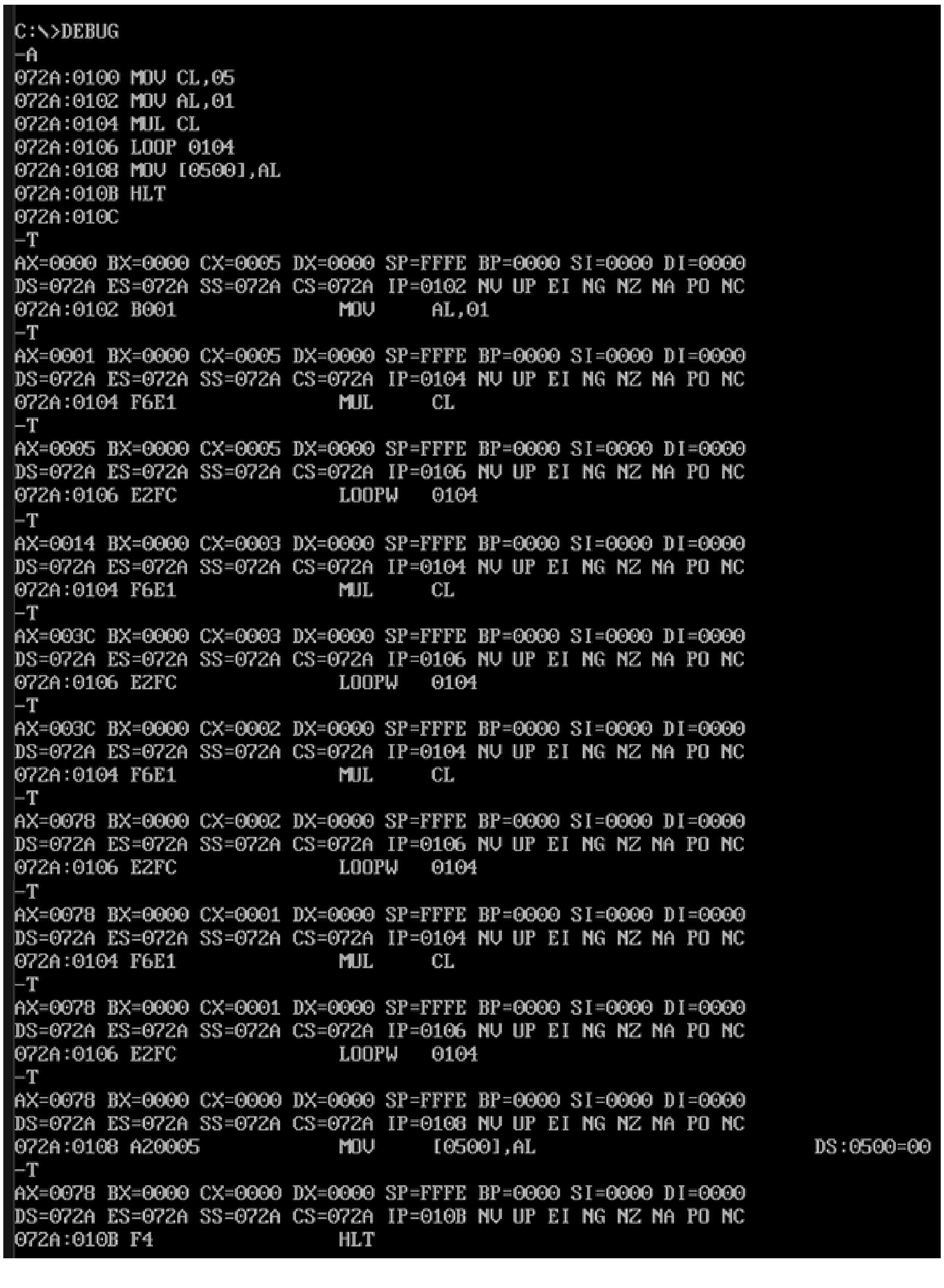
```
C:\>DEBUG
-A
072A:0100 MOV CL,04
072A:0102 MOV AL,01
072A:0104 MUL CL
072A:0106 DEC CL
072A:0108 JNZ 0104
072A:010A MOV [0500],AL
072A:010D HLT
072A:010E
-GCS:010D
AX=0018 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=010D NV UP EI PL ZR NA PE NC
072A:010D F4            HLT
-ECS:0500
072A:0500  18.
```

Experiment - 4b

Aim: Write a Program to generate Factorial of a number.

CODE

```
MOV CL,05
MOV AL,01
A: MUL CL
LOOP A
MOV [0500H], AL
HLT
```



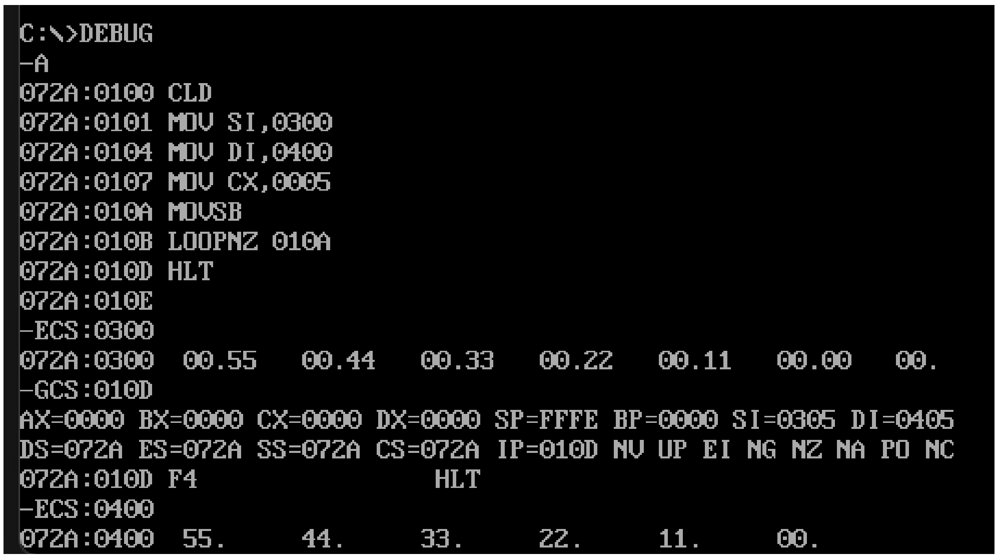
```
C:\>DEBUG
-A
072A:0100 MOV CL,05
072A:0102 MOV AL,01
072A:0104 MUL CL
072A:0106 LOOP 0104
072A:0108 MOV [0500],AL
072A:010B HLT
072A:010C
-T
AX=0000 BX=0000 CX=0005 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0102 NV UP EI NG NZ NA PO NC
072A:0102 B001          MOV     AL,01
-T
AX=0001 BX=0000 CX=0005 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0104 NV UP EI NG NZ NA PO NC
072A:0104 F6E1          MUL     CL
-T
AX=0005 BX=0000 CX=0005 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0106 NV UP EI NG NZ NA PO NC
072A:0106 E2FC          LOOPW   0104
-T
AX=0014 BX=0000 CX=0003 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0104 NV UP EI NG NZ NA PO NC
072A:0104 F6E1          MUL     CL
-T
AX=003C BX=0000 CX=0003 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0106 NV UP EI NG NZ NA PO NC
072A:0106 E2FC          LOOPW   0104
-T
AX=003C BX=0000 CX=0002 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0104 NV UP EI NG NZ NA PO NC
072A:0104 F6E1          MUL     CL
-T
AX=0078 BX=0000 CX=0002 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0106 NV UP EI NG NZ NA PO NC
072A:0106 E2FC          LOOPW   0104
-T
AX=0078 BX=0000 CX=0001 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0104 NV UP EI NG NZ NA PO NC
072A:0104 F6E1          MUL     CL
-T
AX=0078 BX=0000 CX=0001 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0106 NV UP EI NG NZ NA PO NC
072A:0106 E2FC          LOOPW   0104
-T
AX=0078 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=0108 NV UP EI NG NZ NA PO NC
072A:0108 A20005        MOV     [0500],AL                      DS:0500=00
-T
AX=0078 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=072A ES=072A SS=072A CS=072A IP=010B NV UP EI NG NZ NA PO NC
072A:010B F4            HLT
```

Experiment - 5

Aim: Write a program to transfer a block of 5 bytes, starting address is 0300 and transfer the block at address 0400 by using string instructions.

CODE

```
CLD              ;CLD clear the directional flag, auto increments SI & DI
                   register
MOV SI,0300      ;MOV SI, 300 assigns 300 to SI
MOV DI,0400      ;MOV DI, 400 assigns 400 to DI
MOV CX,0005      ;MOV CX, 0005 assign 0000 to CX register
A: MOVSB         ;MOVSB
LOOPNZ A         ; *
HLT              HLT stops the execution of the program.
```



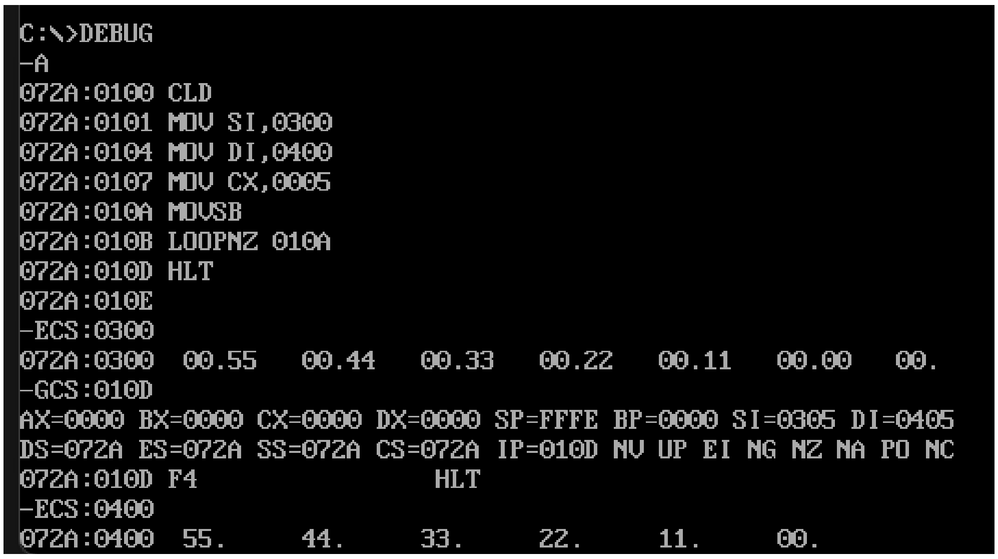
```
C:\>DEBUG
-A
072A:0100 CLD
072A:0101 MOV SI,0300
072A:0104 MOV DI,0400
072A:0107 MOV CX,0005
072A:010A MOVSB
072A:010B LOOPNZ 010A
072A:010D HLT
072A:010E
-ECS:0300
072A:0300  00.55   00.44   00.33   00.22   00.11   00.00   00.
-GCS:010D
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0305 DI=0405
DS=072A ES=072A SS=072A CS=072A IP=010D NV UP EI NG NZ NA PO NC
072A:010D F4            HLT
-ECS:0400
072A:0400  55.     44.     33.     22.     11.     00.
```

Experiment - 5

Aim: Write a program to transfer a block of 5 bytes, starting address is 0300 and transfer the block at address 0400 by using string instructions.

CODE

```
CLD              ;CLD clear the directional flag, auto increments SI & DI
                   register
MOV SI,0300      ;MOV SI, 300 assigns 300 to SI
MOV DI,0400      ;MOV DI, 400 assigns 400 to DI
MOV CX,0005      ;MOV CX, 0005 assign 0000 to CX register
A: MOVSB         ;MOVSB
LOOPNZ A         ; *
HLT              HLT stops the execution of the program.
```



```
C:\>DEBUG
-A
072A:0100 CLD
072A:0101 MOV SI,0300
072A:0104 MOV DI,0400
072A:0107 MOV CX,0005
072A:010A MOVSB
072A:010B LOOPNZ 010A
072A:010D HLT
072A:010E
-ECS:0300
072A:0300  00.55   00.44   00.33   00.22   00.11   00.00   00.
-GCS:010D
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0305 DI=0405
DS=072A ES=072A SS=072A CS=072A IP=010D NV UP EI NG NZ NA PO NC
072A:010D F4             HLT
-ECS:0400
072A:0400  55.     44.     33.     22.     11.     00.
```

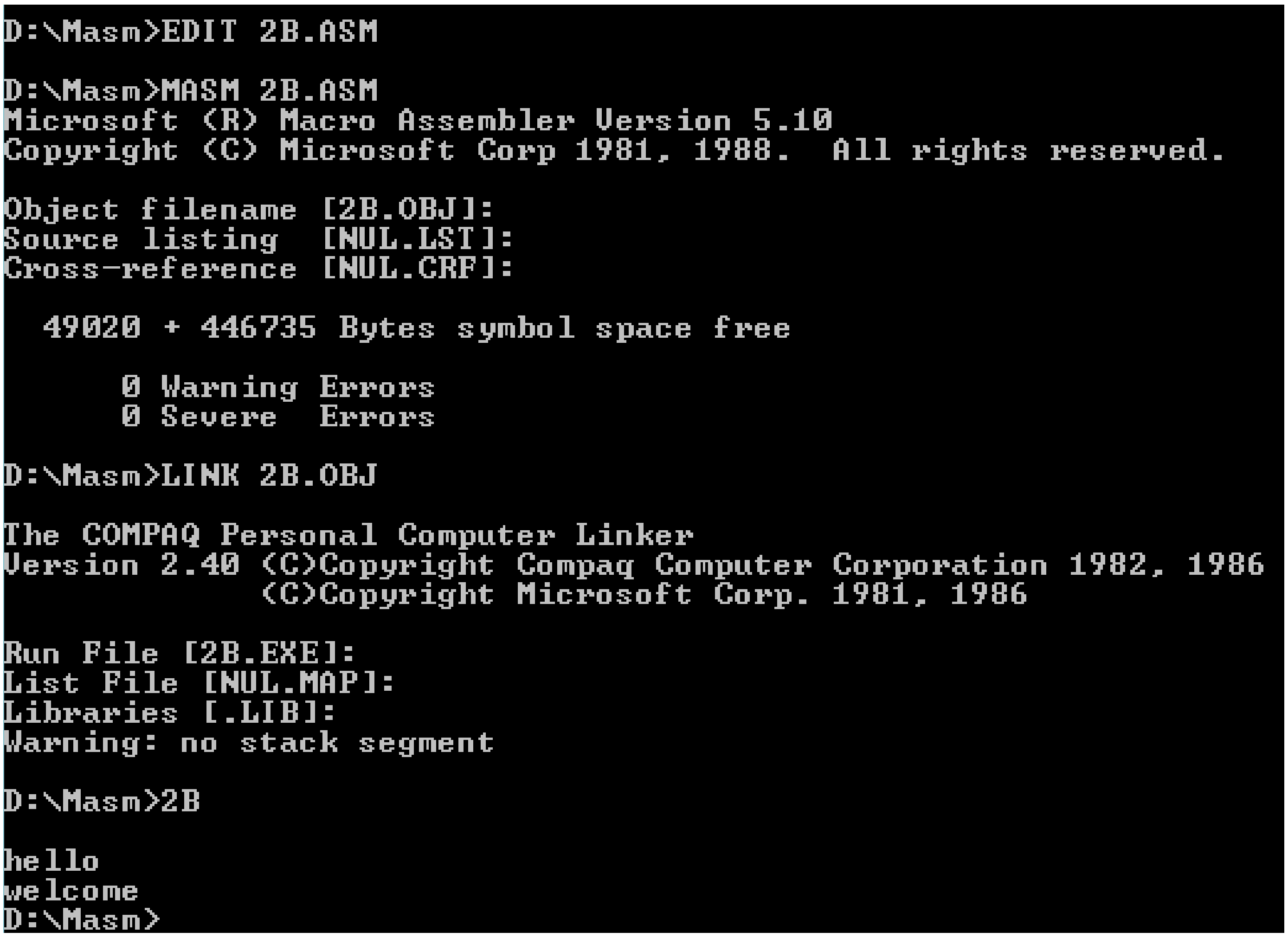
Experiment - 6

Aim: Write an assembly level program to print a given string.
a) Without Macro

CODE

```
DATA SEGMENT
MSG1 DB,0AH,0DH, ‘HELLO$’ ;The string to be printed
MSG2 DB,0AH,0DH, ‘WELCOME$’
DATA ENDS
CODE  SEGMENT
ASSUME CS:CODE, DS: DATA
START: MOV AX,DATA
MOV DS,AX  ; load address of the string
MOV AH,09H ; output the string1
LEA DX,MSG1 ; loaded in dx
INT 21H ; interrupt to exit
MOV AH,09H ; output the string2
LEA DX,MSG2 ; loaded in dx
INT 21H ; interrupt to exit
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

```
data segment
msg1 db 0ah,0dh, "HELLO$" ; The string to be printed
msg2 db 0ah,0dh, "WELCOME$" ; The string to be printed
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS: DATA
START: MOV AX,DATA
    MOV DS,AX  ; load address of the string
    MOV AH,09H ; output the string1
    LEA DX,MSG1 ; loaded in dx
    INT 21H ; interrupt to exit
    MOV AH,09H ; output the string2
    LEA DX,MSG2 ; loaded in dx
    INT 21H ; interrupt to exit
    MOV AH,4CH
    INT 21H
CODE ENDS
END START
```



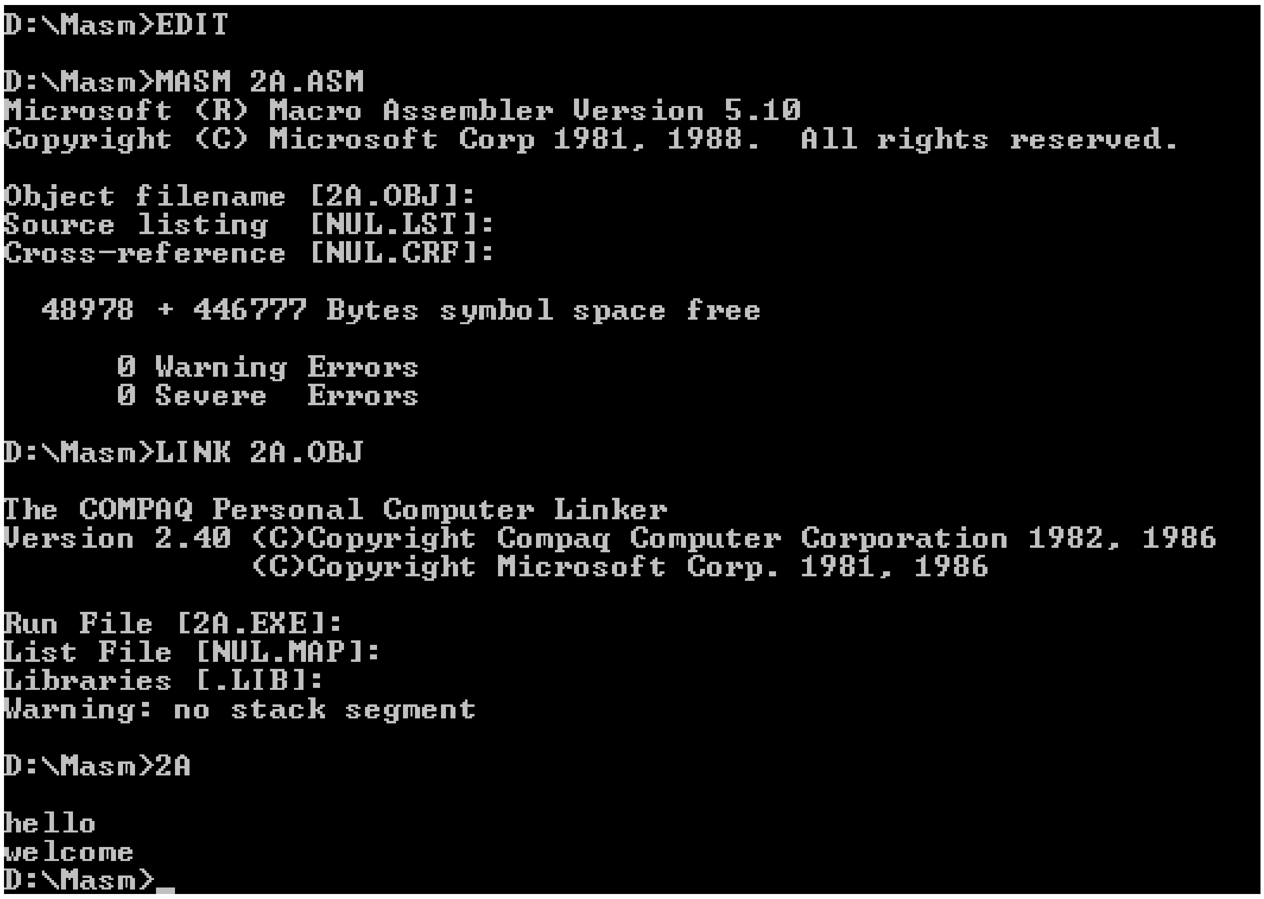
```
D:\Masm>EDIT 2B.ASM

D:\Masm>MASM 2B.ASM
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988.  All rights reserved.

Object filename [2B.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  49020 + 446735 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

D:\Masm>LINK 2B.OBJ

The COMPAQ Personal Computer Linker
Version 2.40 (C)Copyright Compaq Computer Corporation 1982, 1986
             (C)Copyright Microsoft Corp. 1981, 1986

Run File [2B.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
Warning: no stack segment

D:\Masm>2B

hello
welcome
D:\Masm>
```

b) Using Macro

CODE

```
PRINTSTRING MACRO MSG
MOV AH,09H ; output the string
LEA DX,STRING ; loaded in dx
INT 21H ; interrupt to exit
ENDM
DATA SEGMENT
MSG1 DB,0AH,0DH ‘HELLO$’ ; The string to be printed
MSG2 DB,0AH,0DH, ‘WELCOME$’
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS: DATA
START: MOV AX,DATA
MOV DS,AX ; load address of the string
PRINTSTRING MSG1
PRINTSTRING MSG2
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

```
PRINTSTRING MACRO MSG
MOV AH,09H ; output the string
LEA DX,STRING ; loaded in dx
INT 21H ; interrupt to exit
ENDM
DATA SEGMENT
msg1 db,0ah,0dh, "HELLO$"; The string to be printed
msg2 db,0ah,0dh, "WELCOME$"
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS: DATA
START: MOV AX, DATA
    MOV DS,AX ; load address of the string
    PRINTSTRING MACRO MSG1
    PRINTSTRING MACRO MSG2
    MOV AH,4CH
    INT 21H
    CODE ENDS
END START
```



```
D:\Masm>EDIT

D:\Masm>MASM 2A.ASM
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988.  All rights reserved.

Object filename [2A.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  48978 + 446777 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

D:\Masm>LINK 2A.OBJ

The COMPAQ Personal Computer Linker
Version 2.40 (C)Copyright Compaq Computer Corporation 1982, 1986
             (C)Copyright Microsoft Corp. 1981, 1986

Run File [2A.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
Warning: no stack segment

D:\Masm>2A

hello
welcome
D:\Masm>_
```