Preetham v k

MicroProcessor Lab

4NI18IS062

'B' section

PART – A

(8086 Assembly Language Programming)

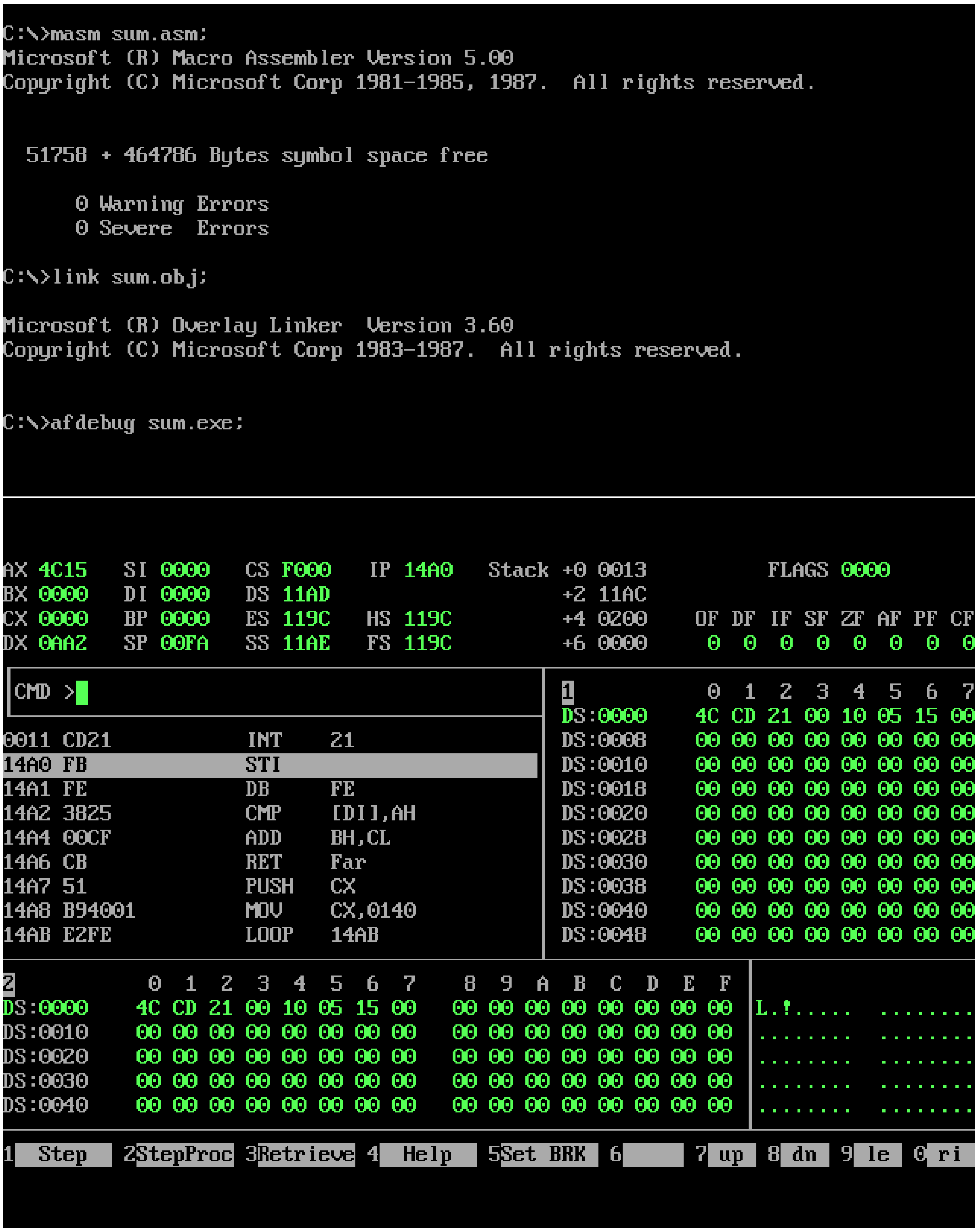
1. Write separate ALPs to add, to subtract and to find an average of two numbers.

Program:

Program to add 2 numbers

```
.MODEL SMALL
.STACK 100H
.DATA
   NUM1 DB 10H
   NUM2 DB 05H
   AD DB ?

.CODE
   MOV AX, @DATA
   MOV DS, AX

   MOV AL, NUM1
   ADD AL, NUM2
   MOV AD, AL

   MOV AH, 4CH
   INT 21H
END
```

Compilation and Outputs:



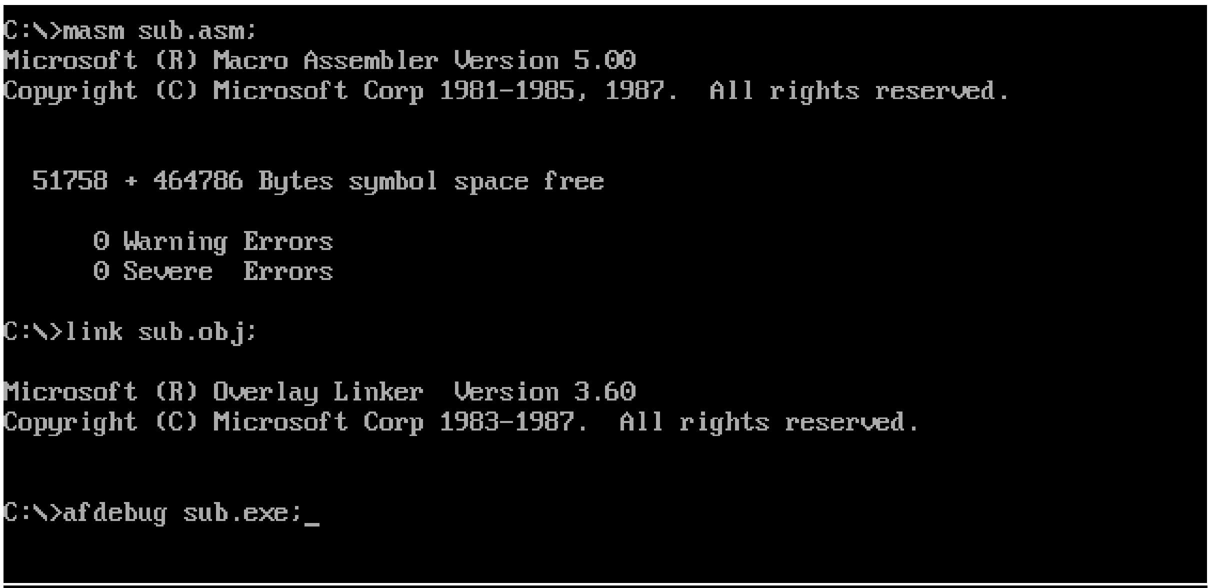
```
C:\>masm sum.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51758 + 464786 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link sum.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>afdebug sum.exe;
```

```
AX 4C15   SI 0000   CS F000   IP 14A0   Stack +0 0013        FLAGS 0000
BX 0000   DI 0000   DS 11AD                   +2 11AC
CX 0000   BP 0000   ES 119C   HS 119C         +4 0200    OF DF IF SF ZF AF PF CF
DX 0AA2   SP 00FA   SS 11AE   FS 119C         +6 0000     0  0  0  0  0  0  0  0

 CMD >                                     1          0  1  2  3  4  5  6  7
                                           DS:0000    4C CD 21 00 10 05 15 00
0011 CD21             INT     21           DS:0008    00 00 00 00 00 00 00 00
14A0 FB               STI                  DS:0010    00 00 00 00 00 00 00 00
14A1 FE               DB      FE           DS:0018    00 00 00 00 00 00 00 00
14A2 3825             CMP     [DI],AH      DS:0020    00 00 00 00 00 00 00 00
14A4 00CF             ADD     BH,CL        DS:0028    00 00 00 00 00 00 00 00
14A6 CB               RET     Far          DS:0030    00 00 00 00 00 00 00 00
14A7 51               PUSH    CX           DS:0038    00 00 00 00 00 00 00 00
14A8 B94001           MOV     CX,0140      DS:0040    00 00 00 00 00 00 00 00
14AB E2FE             LOOP    14AB         DS:0048    00 00 00 00 00 00 00 00

2           0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F
DS:0000    4C CD 21 00 10 05 15 00   00 00 00 00 00 00 00 00   L.!.....  ........
DS:0010    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0020    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0030    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0040    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........

1 Step  2StepProc 3Retrieve 4 Help  5Set BRK 6      7 up  8 dn  9 le  0 ri
```

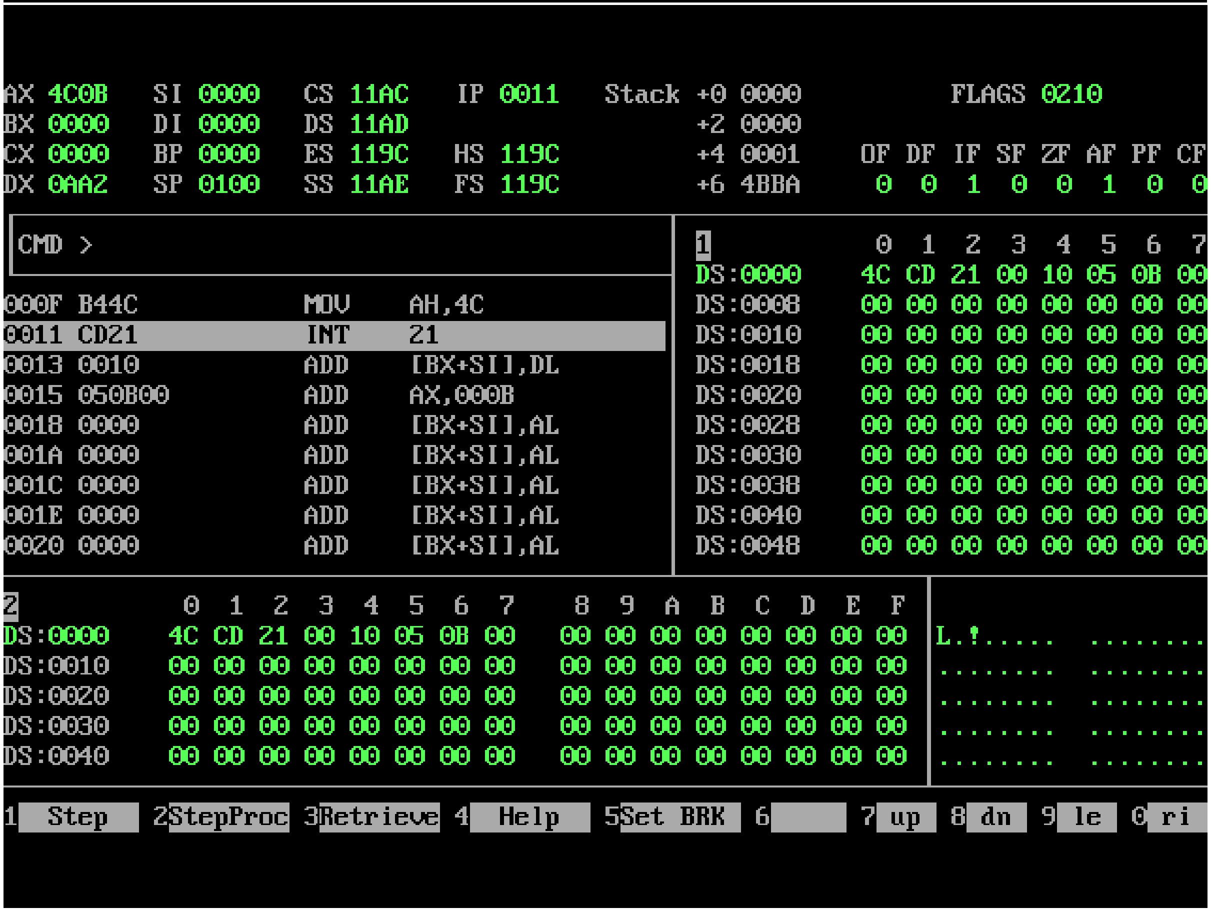
Program to find diff b/w 2 numbers

```
.MODEL SMALL
.STACK 100H
.DATA
   NUM1 DB 10H
   NUM2 DB 5H
   SUB DB ?

.CODE
   MOV AX, @DATA
   MOV DS, AX

   MOV AL, NUM1
   SUB AL, NUM2
   MOV SUB, AL

   MOV AH, 4CH
   INT 21H
END
```

Compilation and Outputs:



```
C:\>masm sub.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51758 + 464786 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link sub.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>afdebug sub.exe;_
```

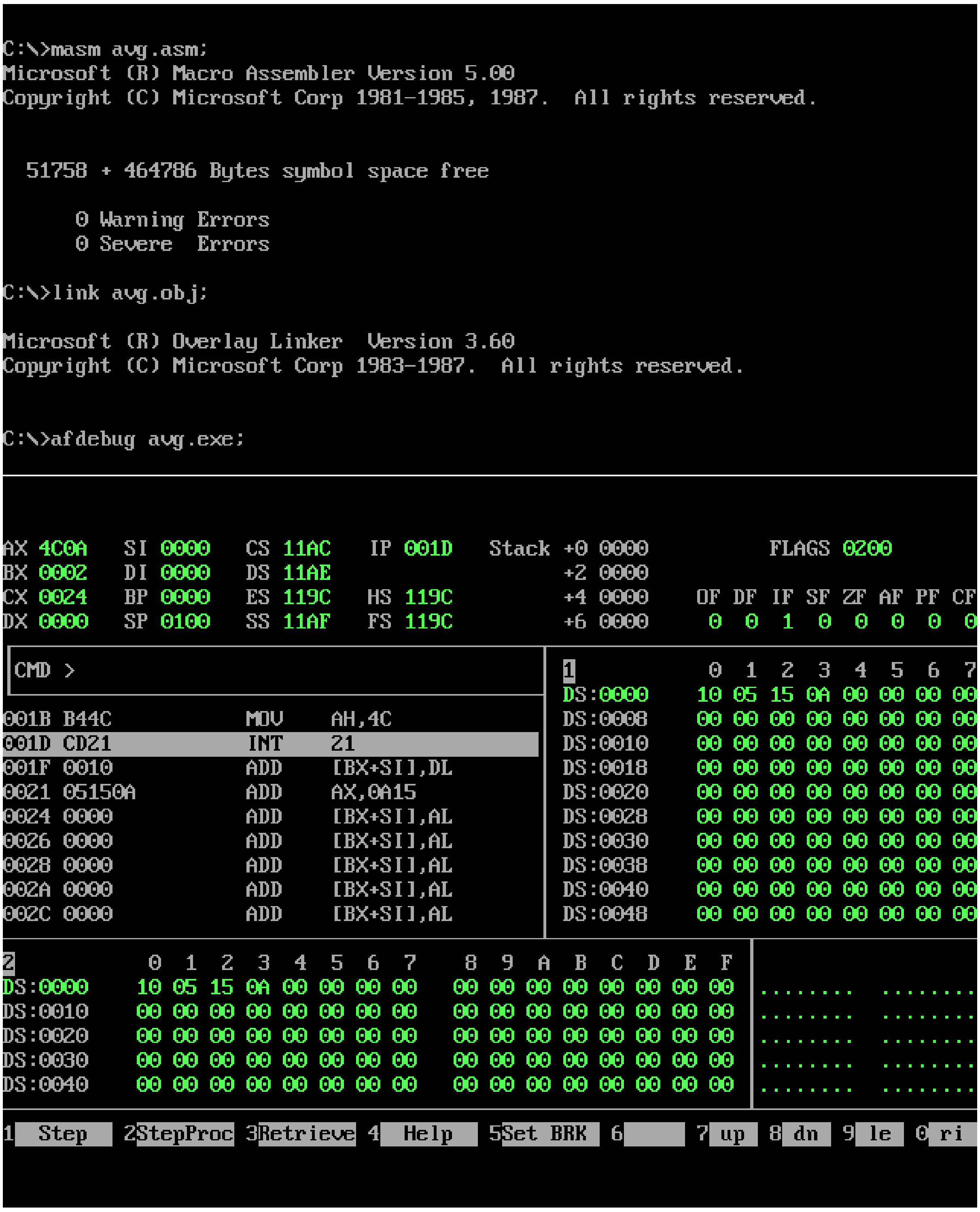


```
AX 4C0B   SI 0000   CS 11AC   IP 0011   Stack +0 0000         FLAGS 0210
BX 0000   DI 0000   DS 11AD                   +2 0000
CX 0000   BP 0000   ES 119C   HS 119C         +4 0001    OF DF IF SF ZF AF PF CF
DX 0AA2   SP 0100   SS 11AE   FS 119C         +6 4BBA     0  0  1  0  0  1  0  0

 CMD >                                      1          0  1  2  3  4  5  6  7
                                            DS:0000    4C CD 21 00 10 05 0B 00
000F B44C         MOV    AH,4C              DS:0008    00 00 00 00 00 00 00 00
0011 CD21         INT    21                 DS:0010    00 00 00 00 00 00 00 00
0013 0010         ADD    [BX+SI],DL         DS:0018    00 00 00 00 00 00 00 00
0015 050B00       ADD    AX,000B            DS:0020    00 00 00 00 00 00 00 00
0018 0000         ADD    [BX+SI],AL         DS:0028    00 00 00 00 00 00 00 00
001A 0000         ADD    [BX+SI],AL         DS:0030    00 00 00 00 00 00 00 00
001C 0000         ADD    [BX+SI],AL         DS:0038    00 00 00 00 00 00 00 00
001E 0000         ADD    [BX+SI],AL         DS:0040    00 00 00 00 00 00 00 00
0020 0000         ADD    [BX+SI],AL         DS:0048    00 00 00 00 00 00 00 00

2           0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F
DS:0000    4C CD 21 00 10 05 0B 00   00 00 00 00 00 00 00 00   L.!.....  ........
DS:0010    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0020    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0030    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0040    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........

1  Step  2StepProc 3Retrieve 4  Help  5Set BRK 6       7 up  8 dn  9 le  0 ri
```

Program to find average of 2 numbers

```
.MODEL SMALL
.STACK 100H
.DATA
   NUM1 DB 10H
   NUM2 DB 5H
   AD DB ?
   AV DB ?

.CODE
   MOV AX, @DATA
   MOV DS, AX

   MOV AL, NUM1
   ADD AL, NUM2
   MOV AD, AL
   MOV AH, 00H
   MOV AL, AD
   MOV BL,02H
   DIV BL
   MOV AV, AL

   MOV AH, 4CH
   INT 21H
END
```

Compilation and Outputs:



```
C:\>masm avg.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51758 + 464786 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link avg.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>afdebug avg.exe;
```

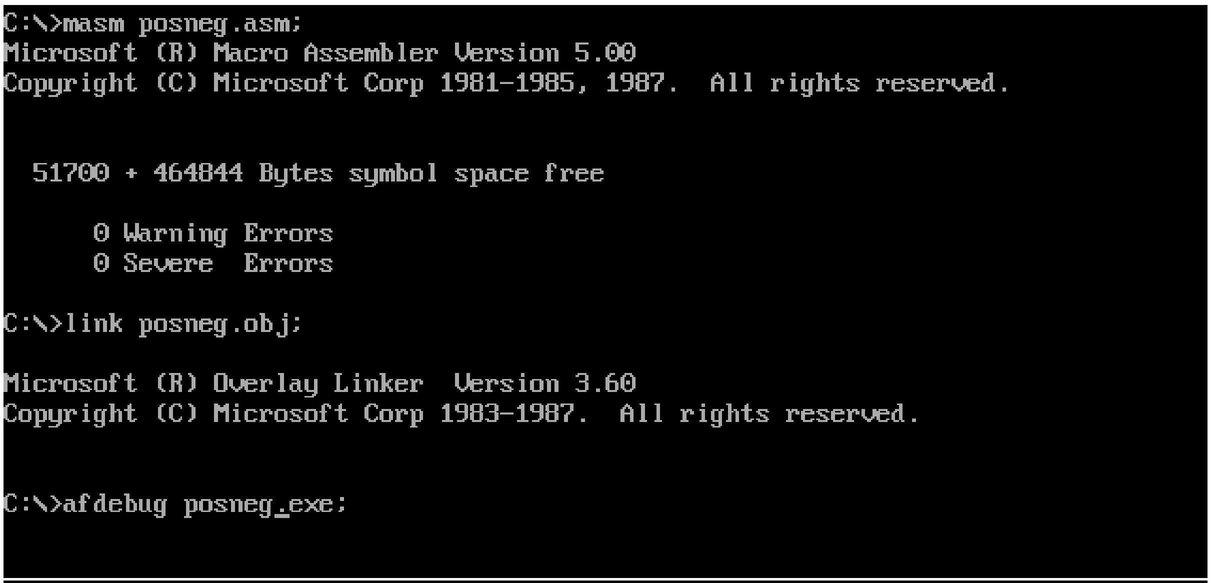
```
AX 4C0A   SI 0000   CS 11AC   IP 001D   Stack +0 0000        FLAGS 0200
BX 0002   DI 0000   DS 11AE                   +2 0000
CX 0024   BP 0000   ES 119C   HS 119C         +4 0000     OF DF IF SF ZF AF PF CF
DX 0000   SP 0100   SS 11AF   FS 119C         +6 0000      0  0  1  0  0  0  0  0

 CMD >                                       1           0  1  2  3  4  5  6  7
                                             DS:0000    10 05 15 0A 00 00 00 00
001B B44C          MOV    AH,4C              DS:0008    00 00 00 00 00 00 00 00
001D CD21          INT    21                 DS:0010    00 00 00 00 00 00 00 00
001F 0010          ADD    [BX+SI],DL         DS:0018    00 00 00 00 00 00 00 00
0021 05150A        ADD    AX,0A15            DS:0020    00 00 00 00 00 00 00 00
0024 0000          ADD    [BX+SI],AL         DS:0028    00 00 00 00 00 00 00 00
0026 0000          ADD    [BX+SI],AL         DS:0030    00 00 00 00 00 00 00 00
0028 0000          ADD    [BX+SI],AL         DS:0038    00 00 00 00 00 00 00 00
002A 0000          ADD    [BX+SI],AL         DS:0040    00 00 00 00 00 00 00 00
002C 0000          ADD    [BX+SI],AL         DS:0048    00 00 00 00 00 00 00 00

2          0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F
DS:0000   10 05 15 0A 00 00 00 00   00 00 00 00 00 00 00 00   ........ ........
DS:0010   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........ ........
DS:0020   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........ ........
DS:0030   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........ ........
DS:0040   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........ ........

1 Step  2StepProc 3Retrieve 4 Help  5Set BRK 6      7 up 8 dn 9 le 0 ri
```

2. Write an ALP to check given number is positive or negative.

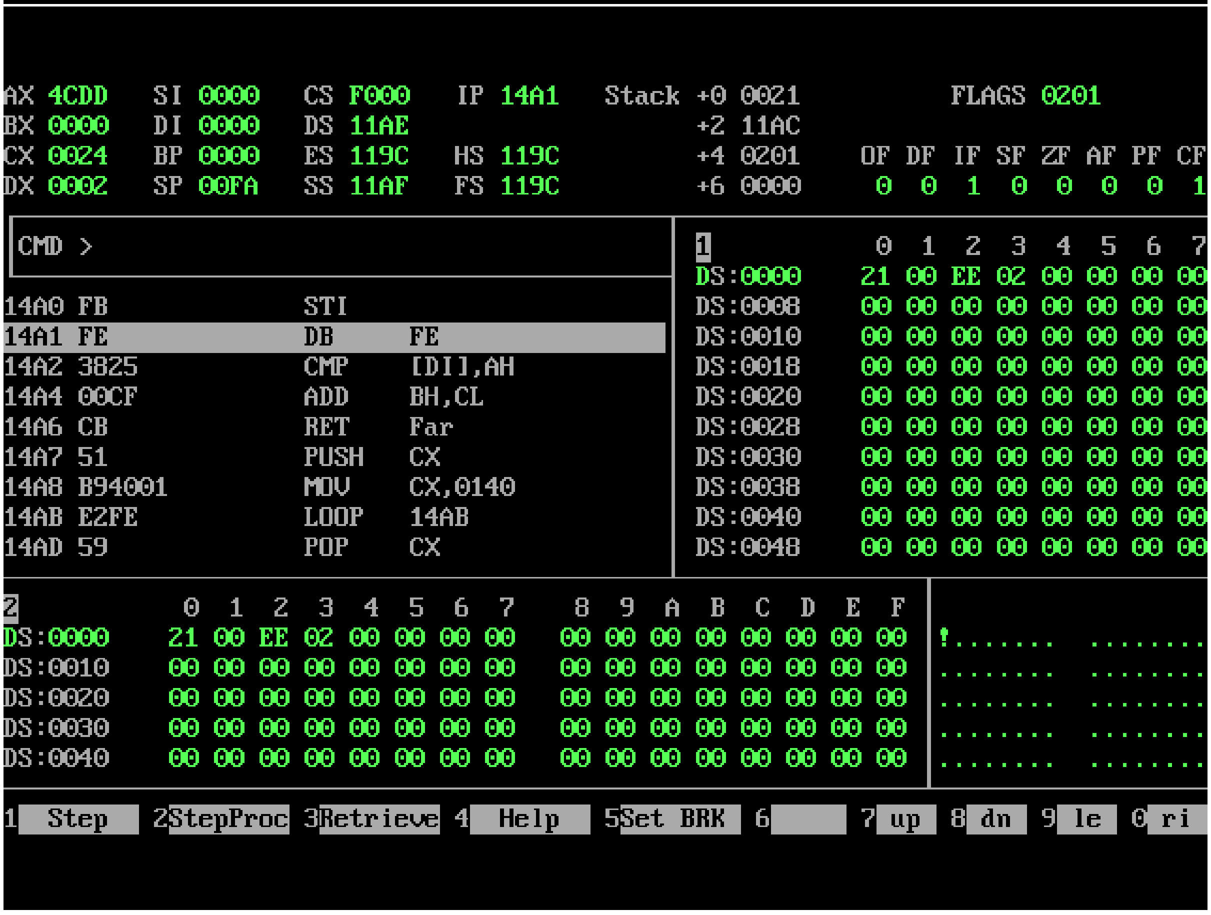
Program:

```
; Program to check number is positive or not
.MODEL SMALL
.STACK 100H
.DATA
   NUM DB -12H
   RES DB ?

.CODE
    MOV AX , @DATA    ; Initializing Data Segment
    MOV DS , AX

    MOV AL , NUM      ; LOAD NUMBER
    ROL AL , 01       ; ROTATE BY 01

    JC  DN
     MOV RES , 1      ; POSITIVE
     JMP EXIT
  DN:
     MOV RES , 2       ; NEGATIVE

EXIT:
     MOV DL , RES

    MOV AH , 4CH      ; Service routine for exit
    INT 21H
END
```

Compilation and Outputs:



```
C:\>masm posneg.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51700 + 464844 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link posneg.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>afdebug posneg.exe;
```



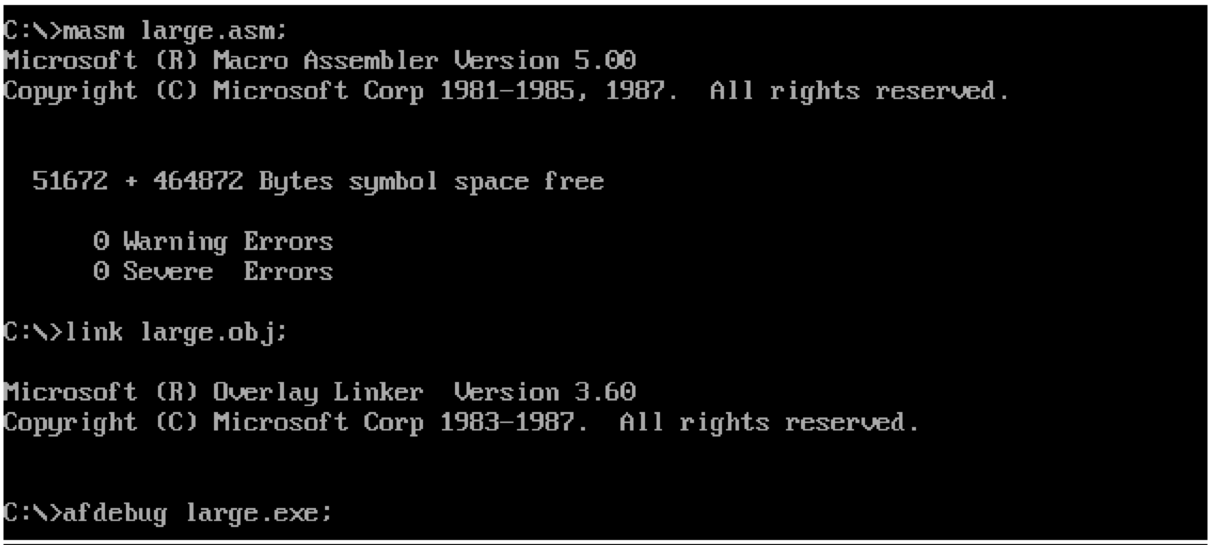
```
AX 4CDD   SI 0000   CS F000   IP 14A1   Stack +0 0021        FLAGS 0201
BX 0000   DI 0000   DS 11AE                   +2 11AC
CX 0024   BP 0000   ES 119C   HS 119C         +4 0201    OF DF IF SF ZF AF PF CF
DX 0002   SP 00FA   SS 11AF   FS 119C         +6 0000     0  0  1  0  0  0  0  1

 CMD >                                      1          0  1  2  3  4  5  6  7
                                            DS:0000   21 00 EE 02 00 00 00 00
14A0 FB                STI                  DS:0008   00 00 00 00 00 00 00 00
14A1 FE                DB      FE           DS:0010   00 00 00 00 00 00 00 00
14A2 3825              CMP     [DI],AH      DS:0018   00 00 00 00 00 00 00 00
14A4 00CF              ADD     BH,CL        DS:0020   00 00 00 00 00 00 00 00
14A6 CB                RET     Far          DS:0028   00 00 00 00 00 00 00 00
14A7 51                PUSH    CX           DS:0030   00 00 00 00 00 00 00 00
14A8 B94001            MOV     CX,0140      DS:0038   00 00 00 00 00 00 00 00
14AB E2FE              LOOP    14AB         DS:0040   00 00 00 00 00 00 00 00
14AD 59                POP     CX           DS:0048   00 00 00 00 00 00 00 00

2           0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F
DS:0000    21 00 EE 02 00 00 00 00   00 00 00 00 00 00 00 00   !.......  ........
DS:0010    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0020    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0030    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0040    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........

1 Step   2StepProc 3Retrieve 4 Help   5Set BRK  6       7 up  8 dn  9 le  0 ri
```

3. Write an ALP to find the largest of N numbers.

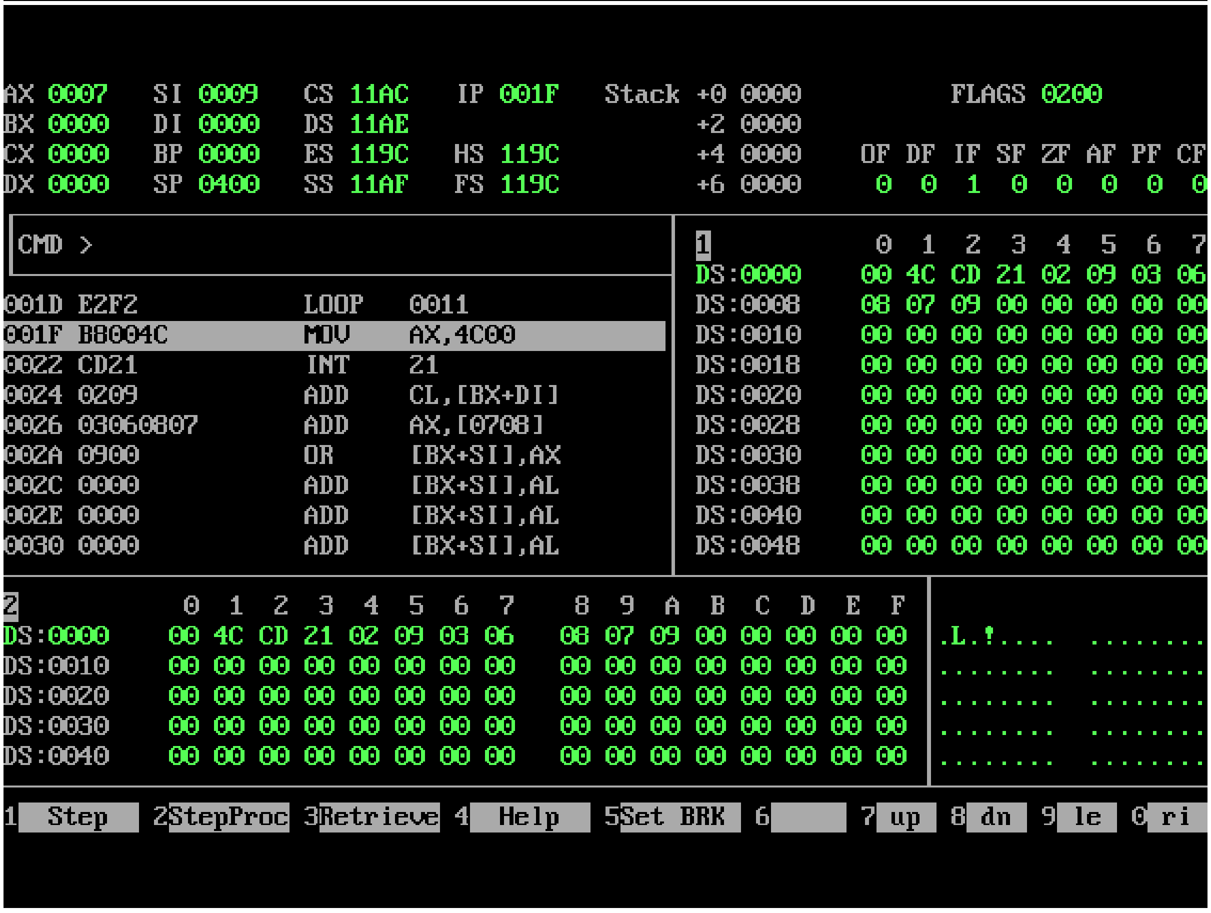
Program:

```
.MODEL SMALL
.STACK
.DATA
        LIST DB 02h, 09h, 03h, 06h, 08h, 07h
        large db ?
.CODE
        MOV AX, @DATA
        MOV DS, AX
        MOV SI, OFFSET LIST
        MOV CL, 05h
        XOR AX, AX
        MOV AL, [SI]
        MOV large, AL
 UP:    INC SI
        MOV AL, [SI]
        CMP large, AL
        JNB GO
        mov large, AL
 GO:    LOOP UP
        mov ax,4c00h
        INT 21h
END
```

Compilation and Outputs:



```
C:\>masm large.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51672 + 464872 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link large.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>afdebug large.exe;
```



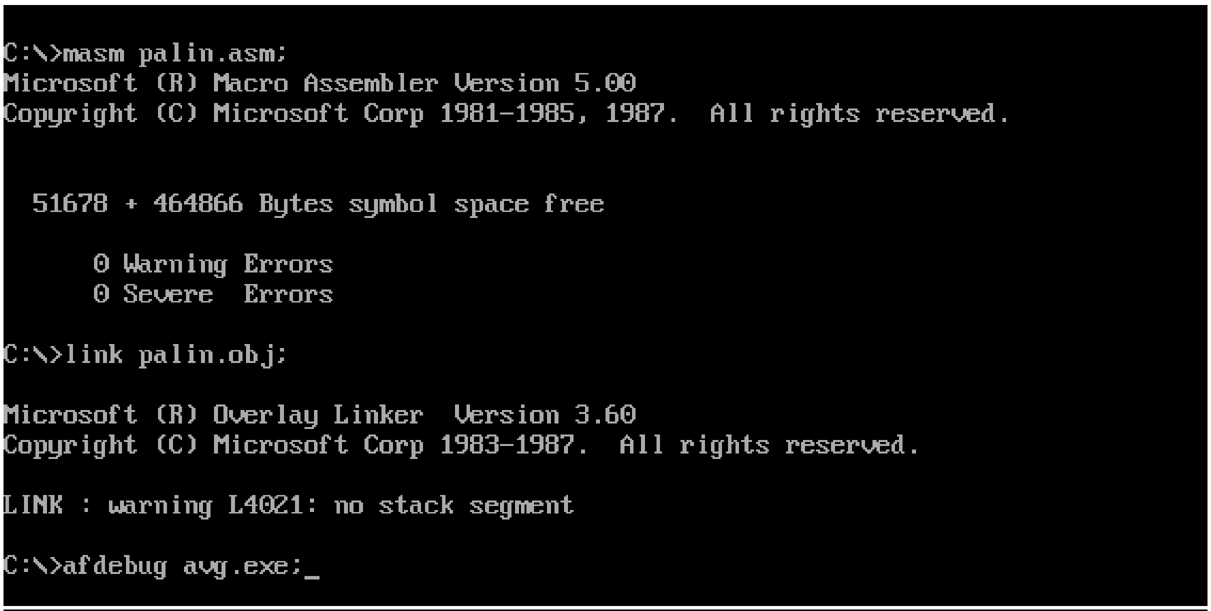
```
AX 0007   SI 0009   CS 11AC   IP 001F   Stack +0 0000       FLAGS 0200
BX 0000   DI 0000   DS 11AE                   +2 0000
CX 0000   BP 0000   ES 119C   HS 119C         +4 0000     OF DF IF SF ZF AF PF CF
DX 0000   SP 0400   SS 11AF   FS 119C         +6 0000      0  0  1  0  0  0  0  0

 CMD >                                      1          0  1  2  3  4  5  6  7
                                            DS:0000    00 4C CD 21 02 09 03 06
001D E2F2           LOOP    0011            DS:0008    08 07 09 00 00 00 00 00
001F B8004C         MOV     AX,4C00         DS:0010    00 00 00 00 00 00 00 00
0022 CD21           INT     21              DS:0018    00 00 00 00 00 00 00 00
0024 0209           ADD     CL,[BX+DI]      DS:0020    00 00 00 00 00 00 00 00
0026 03060807       ADD     AX,[0708]       DS:0028    00 00 00 00 00 00 00 00
002A 0900           OR      [BX+SI],AX      DS:0030    00 00 00 00 00 00 00 00
002C 0000           ADD     [BX+SI],AL      DS:0038    00 00 00 00 00 00 00 00
002E 0000           ADD     [BX+SI],AL      DS:0040    00 00 00 00 00 00 00 00
0030 0000           ADD     [BX+SI],AL      DS:0048    00 00 00 00 00 00 00 00

2          0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F
DS:0000    00 4C CD 21 02 09 03 06   08 07 09 00 00 00 00 00   .L.!....  ........
DS:0010    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0020    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0030    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........
DS:0040    00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........

1  Step   2StepProc 3Retrieve 4  Help   5Set BRK  6      7 up  8 dn  9 le  0 ri
```

4. Write an ALP to find whether the given string is palindrome or not.

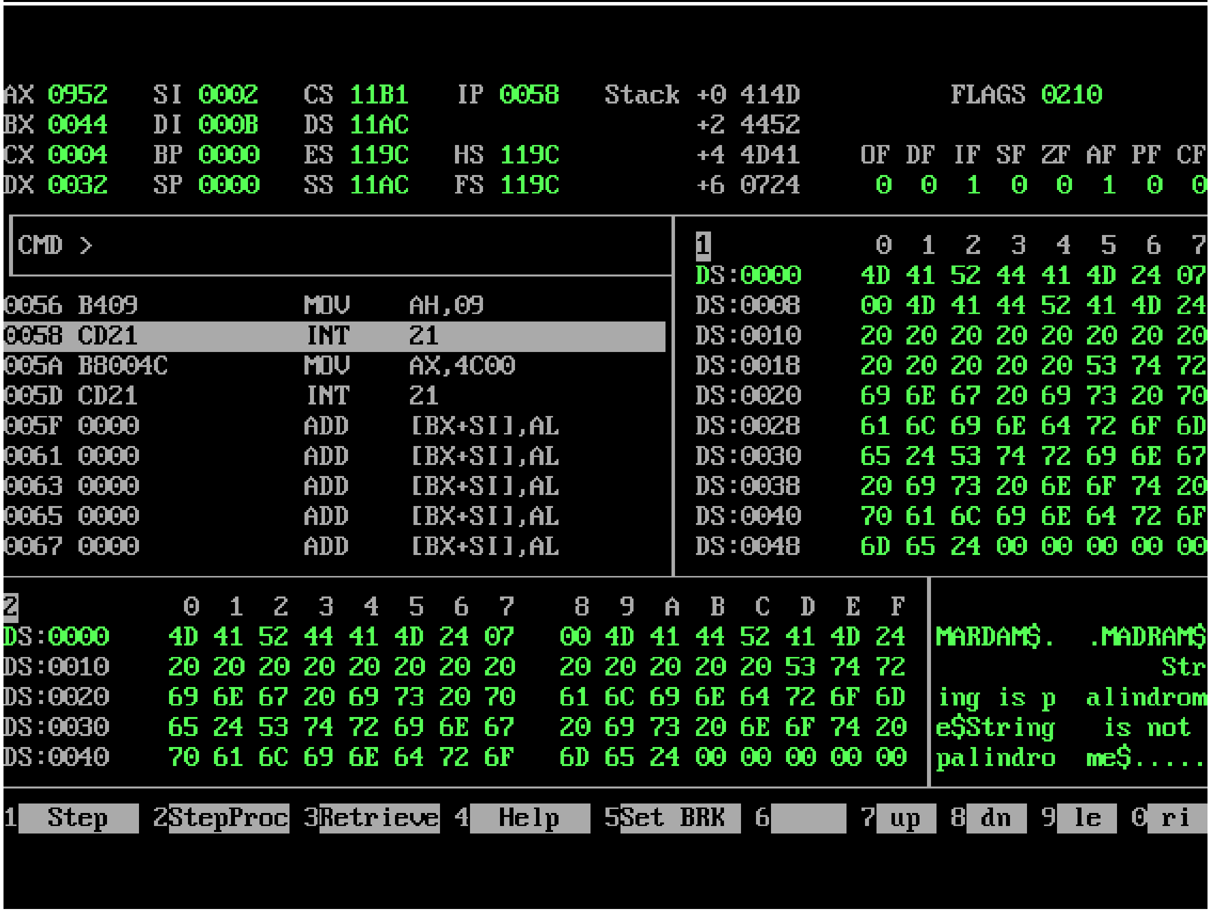
Program:

```
.MODEL SMALL
.STACK
.DATA
        str1 db 'MARDAM','$'
        strlen1 dw $-str1
        strrev db 20 dup(' ')
        str_palin db 'String is palindrome','$'
        str_not_palin db 'String is not palindrome','$'

.CODE

                mov ax,@DATA
                mov ds,ax
                mov cx,strlen1
                add cx,-2

                lea si,str1
                lea di,strrev

                add si,strlen1
                add si,-2
                L1:
                        mov al,[si]
                        mov [di],al
                        dec si
                        inc di
                        loop L1
                        mov al,[si]
                        mov [di], al
                        inc di
                        mov dl, '$'
                        mov [di], dl
                        mov cx, strlen1
                        add cx,-1
                        lea si, str1
                        lea di, strrev

                Palin_Check:
                        mov al, [si]
                        mov bl, [di]
                        cmp al,bl
                        JNE Not_Palin
                        inc si
                        inc di
                        loop Palin_Check

                Palin:
                        lea dx, str_palin
                        mov ah, 09h
                        int 21h
                        jmp Exit

                Not_Palin:
                        lea dx, str_not_palin
                        mov ah, 09h
                        int 21h



                Exit:
                        mov ax, 4c00h
                        int 21h
End
```

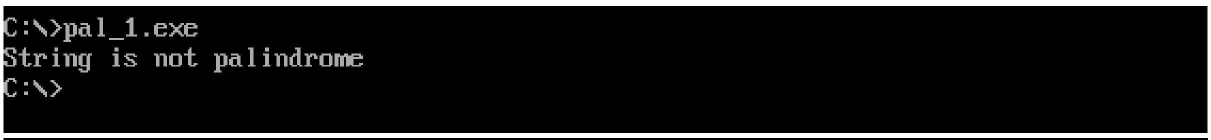
Compilation and Outputs:



```
C:\>masm palin.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


 51678 + 464866 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link palin.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

LINK : warning L4021: no stack segment

C:\>afdebug avg.exe;_
```



```
AX 0952   SI 0002   CS 11B1   IP 0058   Stack +0 414D        FLAGS 0210
BX 0044   DI 000B   DS 11AC                   +2 4452
CX 0004   BP 0000   ES 119C   HS 119C         +4 4D41    OF DF IF SF ZF AF PF CF
DX 0032   SP 0000   SS 11AC   FS 119C         +6 0724     0  0  1  0  0  1  0  0

 CMD >                                       1        0  1  2  3  4  5  6  7
                                             DS:0000  4D 41 52 44 41 4D 24 07
0056 B409          MOV    AH,09              DS:0008  00 4D 41 44 52 41 4D 24
0058 CD21          INT    21                 DS:0010  20 20 20 20 20 20 20 20
005A B8004C        MOV    AX,4C00            DS:0018  20 20 20 20 20 53 74 72
005D CD21          INT    21                 DS:0020  69 6E 67 20 69 73 20 70
005F 0000          ADD    [BX+SI],AL         DS:0028  61 6C 69 6E 64 72 6F 6D
0061 0000          ADD    [BX+SI],AL         DS:0030  65 24 53 74 72 69 6E 67
0063 0000          ADD    [BX+SI],AL         DS:0038  20 69 73 20 6E 6F 74 20
0065 0000          ADD    [BX+SI],AL         DS:0040  70 61 6C 69 6E 64 72 6F
0067 0000          ADD    [BX+SI],AL         DS:0048  6D 65 24 00 00 00 00 00

2        0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
DS:0000  4D 41 52 44 41 4D 24 07  00 4D 41 44 52 41 4D 24  MARDAM$.  .MADRAM$
DS:0010  20 20 20 20 20 20 20 20  20 20 20 20 20 53 74 72               Str
DS:0020  69 6E 67 20 69 73 20 70  61 6C 69 6E 64 72 6F 6D  ing is p alindrom
DS:0030  65 24 53 74 72 69 6E 67  20 69 73 20 6E 6F 74 20  e$String  is not
DS:0040  70 61 6C 69 6E 64 72 6F  6D 65 24 00 00 00 00 00  palindro me$.....

1 Step  2StepProc 3Retrieve 4 Help  5Set BRK 6      7 up  8 dn  9 le  0 ri
```



```
C:\>pal_1.exe
String is not palindrome
C:\>
```

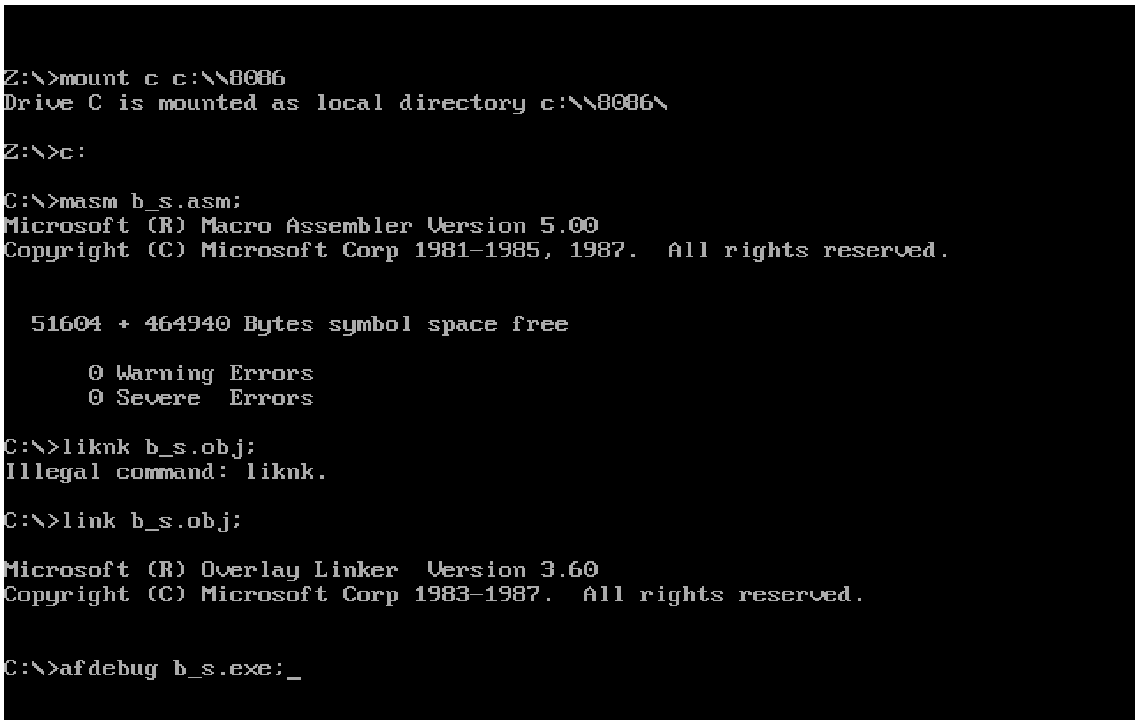
5. Write an ALP to perform binary search and display the output on the monitor.

Program:

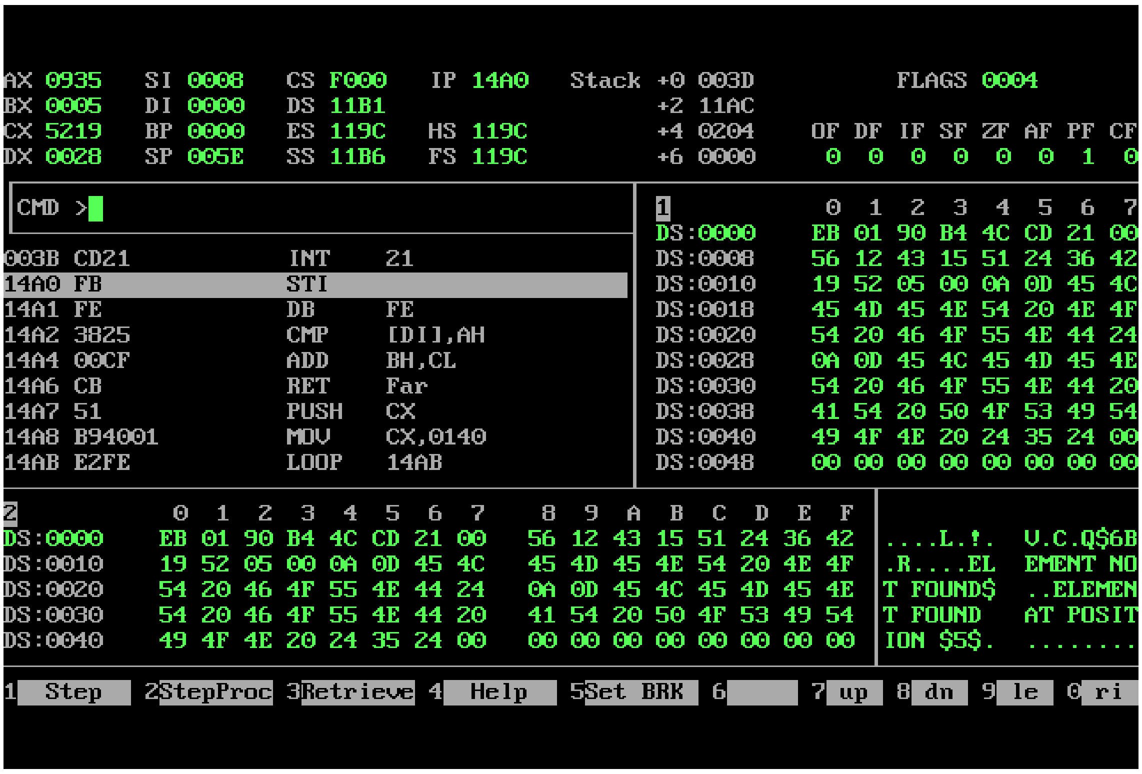
```
.MODEL SMALL
.STACK 100
.DATA
        ARR DW 1256H,1543H,2451H,4236H, 5219H
        LEN DW ($-ARR)/2
        KEY EQU 5219H
        MSG1 DB 10,13,"ELEMENT NOT FOUND$"
        MSG2 DB 10,13,"ELEMENT FOUND AT POSITION $"
        RES DB ?,"$"
.CODE
        MOV AX,@DATA
        MOV DS,AX
        MOV BX,01H
        MOV DX,LEN
        MOV CX,KEY
        RPT:    CMP BX,DX
                JA FAIL
                MOV AX,BX
                ADD AX,DX
                SHR AX,01       ;divide by 2
                MOV SI,AX
                DEC SI
                ADD SI,SI      ;stored as word so occupies 2 bytes
                CMP CX,ARR[SI]
                JAE SEC
                DEC AX
                MOV DX,AX
                JMP RPT

        SEC:    JE SUCCESS
```

```
            INC AX
            MOV BX,AX
            JMP RPT

        SUCCESS:    ADD AL,30H
                    MOV RES,AL
                    LEA DX,MSG2
                    MOV AH,09H
                    INT 21H
                    LEA DX,RES
                    MOV AH,09H
                    INT 21H
                    JMP EXIT

        FAIL:   LEA DX,MSG1
                MOV AH,09H
                INT 21H
                JMP EXIT

        EXIT:   MOV AH,4CH
                INT 21H
END
```

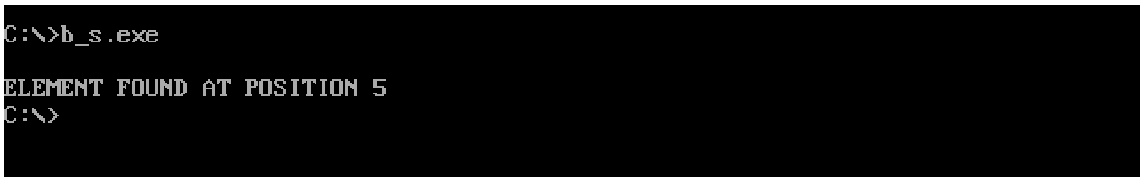
Compilation and Outputs:



```
Z:\>mount c c:\\8086
Drive C is mounted as local directory c:\\8086\

Z:\>c:

C:\>masm b_s.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  51604 + 464940 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>liknk b_s.obj;
Illegal command: liknk.

C:\>link b_s.obj;

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.


C:\>afdebug b_s.exe;_
```



```
AX 0935   SI 0008   CS F000   IP 14A0   Stack +0 003D        FLAGS 0004
BX 0005   DI 0000   DS 11B1                   +2 11AC
CX 5219   BP 0000   ES 119C   HS 119C         +4 0204     OF DF IF SF ZF AF PF CF
DX 0028   SP 005E   SS 11B6   FS 119C         +6 0000      0  0  0  0  0  0  1  0

 CMD >                                      1          0  1  2  3  4  5  6  7
                                            DS:0000    EB 01 90 B4 4C CD 21 00
003B CD21            INT    21              DS:0008    56 12 43 15 51 24 36 42
14A0 FB              STI                    DS:0010    19 52 05 00 0A 0D 45 4C
14A1 FE              DB     FE              DS:0018    45 4D 45 4E 54 20 4E 4F
14A2 3825            CMP    [DI],AH         DS:0020    54 20 46 4F 55 4E 44 24
14A4 00CF            ADD    BH,CL           DS:0028    0A 0D 45 4C 45 4D 45 4E
14A6 CB              RET    Far             DS:0030    54 20 46 4F 55 4E 44 20
14A7 51              PUSH   CX              DS:0038    41 54 20 50 4F 53 49 54
14A8 B94001          MOV    CX,0140         DS:0040    49 4F 4E 20 24 35 24 00
14AB E2FE            LOOP   14AB            DS:0048    00 00 00 00 00 00 00 00

2        0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
DS:0000  EB 01 90 B4 4C CD 21 00  56 12 43 15 51 24 36 42  ....L.!.  V.C.Q$6B
DS:0010  19 52 05 00 0A 0D 45 4C  45 4D 45 4E 54 20 4E 4F  .R....EL  EMENT NO
DS:0020  54 20 46 4F 55 4E 44 24  0A 0D 45 4C 45 4D 45 4E  T FOUND$  ..ELEMEN
DS:0030  54 20 46 4F 55 4E 44 20  41 54 20 50 4F 53 49 54  T FOUND   AT POSIT
DS:0040  49 4F 4E 20 24 35 24 00  00 00 00 00 00 00 00 00  ION $5$.  ........

1 Step  2StepProc 3Retrieve 4 Help  5Set BRK 6      7 up  8 dn  9 le  0 ri
```



```
C:\>b_s.exe

ELEMENT FOUND AT POSITION 5
C:\>
```

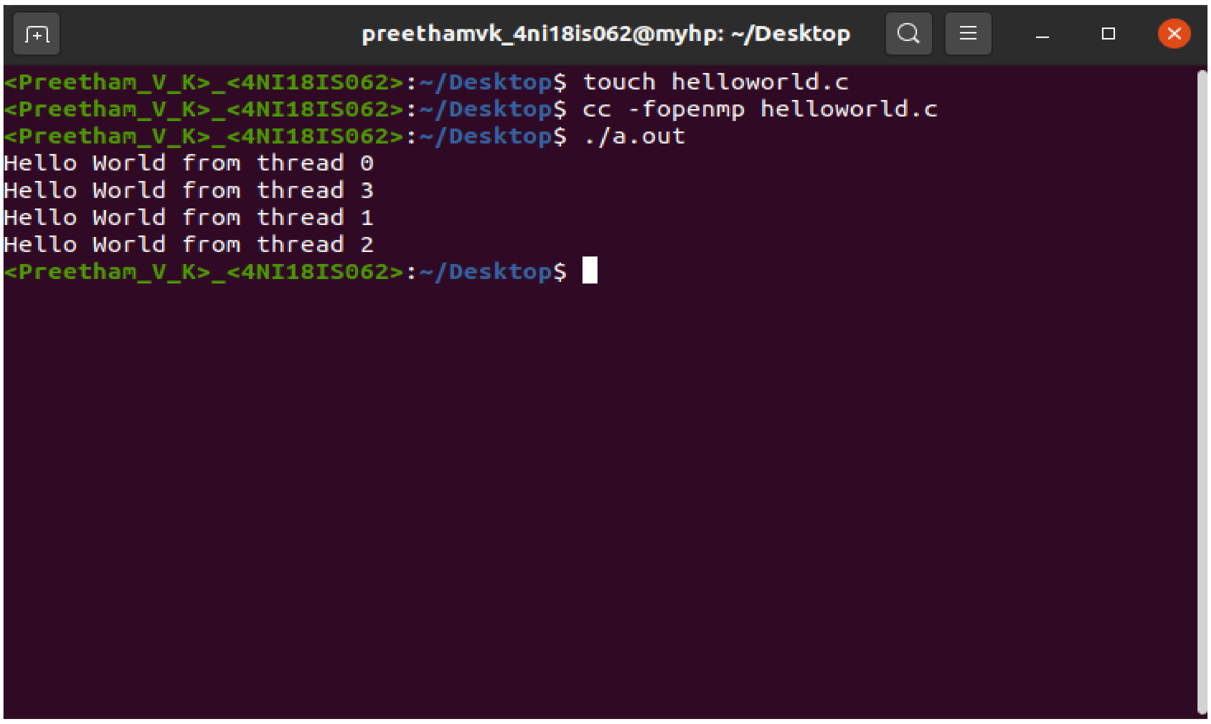
(OpenMP Programming)

6. Write a program to print Hello World from multiple threads using OpenMP.

Program:

```
#include <stdio.h>

#include <omp.h>

#include<stdlib.h>


int main(int argc,char *argv[])

{

   #pragma omp parallel

   {

      printf("Hello World from thread %d\n",omp_get_thread_num());

   }

  return 0;

}
```

Compilation and Outputs:



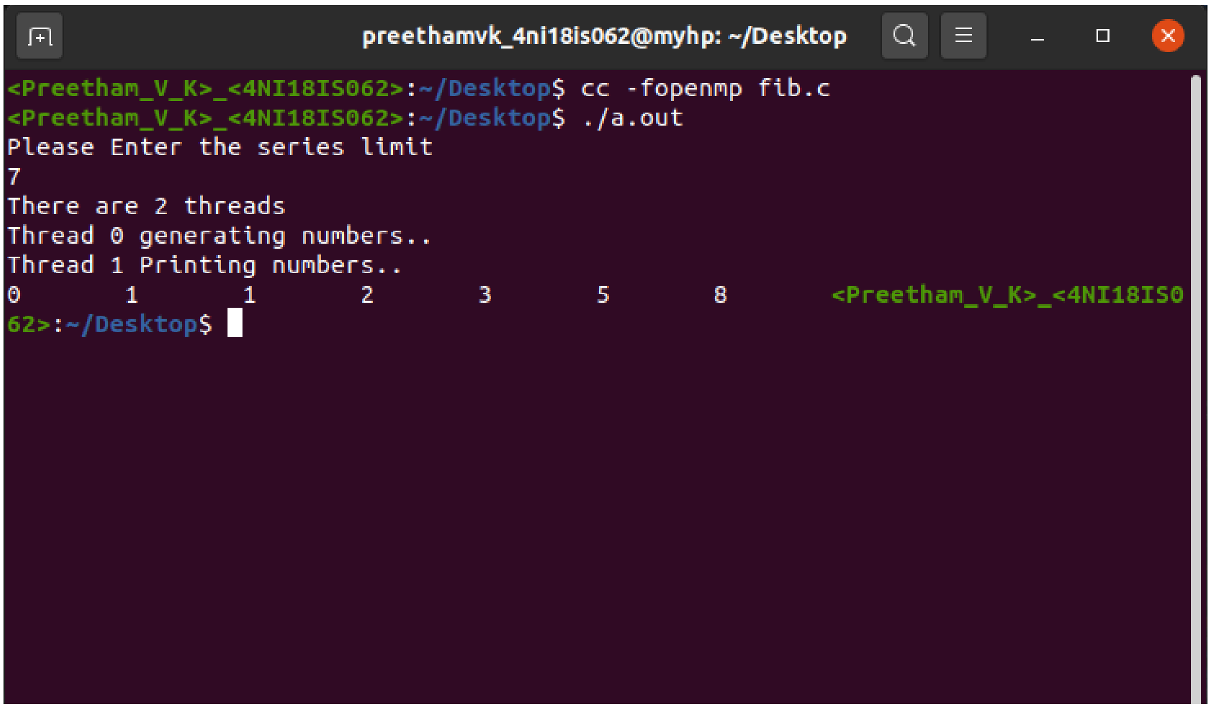
```
<Preetham_V_K>_<4NI18IS062>:~/Desktop$ touch helloworld.c
<Preetham_V_K>_<4NI18IS062>:~/Desktop$ cc -fopenmp helloworld.c
<Preetham_V_K>_<4NI18IS062>:~/Desktop$ ./a.out
Hello World from thread 0
Hello World from thread 3
Hello World from thread 1
Hello World from thread 2
<Preetham_V_K>_<4NI18IS062>:~/Desktop$ 
```

7. Write a program to generate Fibonacci series using OpenMP.

Program:

```
#include<stdio.h>

#include<omp.h>


int fib(int n)

{

if(n<2) return n;

else return fib(n-1)+fib(n-2);

}


int main()

{

int fibnumber[100],i,j,n;

printf("Please Enter the series limit\n");

scanf("%d",&n);

#pragma omp parallel num_threads(2)

{

#pragma omp critical

if(omp_get_thread_num()==0)

{

printf("There are %d threads\n", omp_get_num_threads());

printf("Thread %d generating numbers..\n", omp_get_thread_num());

for(i=0;i<n;i++)

fibnumber[i]=fib(i);

}

else

{

printf("Thread %d Printing numbers..\n", omp_get_thread_num());

for(j=0;j<n;j++)

printf("%d\t", fibnumber[j]);
}


}
 return 0;
}
```

Compilation and Outputs:



```
<Preetham_V_K>_<4NI18IS062>:~/Desktop$ cc -fopenmp fib.c
<Preetham_V_K>_<4NI18IS062>:~/Desktop$ ./a.out
Please Enter the series limit
7
There are 2 threads
Thread 0 generating numbers..
Thread 1 Printing numbers..
0       1       1       2       3       5       8       <Preetham_V_K>_<4NI18IS0
62>:~/Desktop$ 
```

8. Write a program for Matrix multiplication using OPENMP.

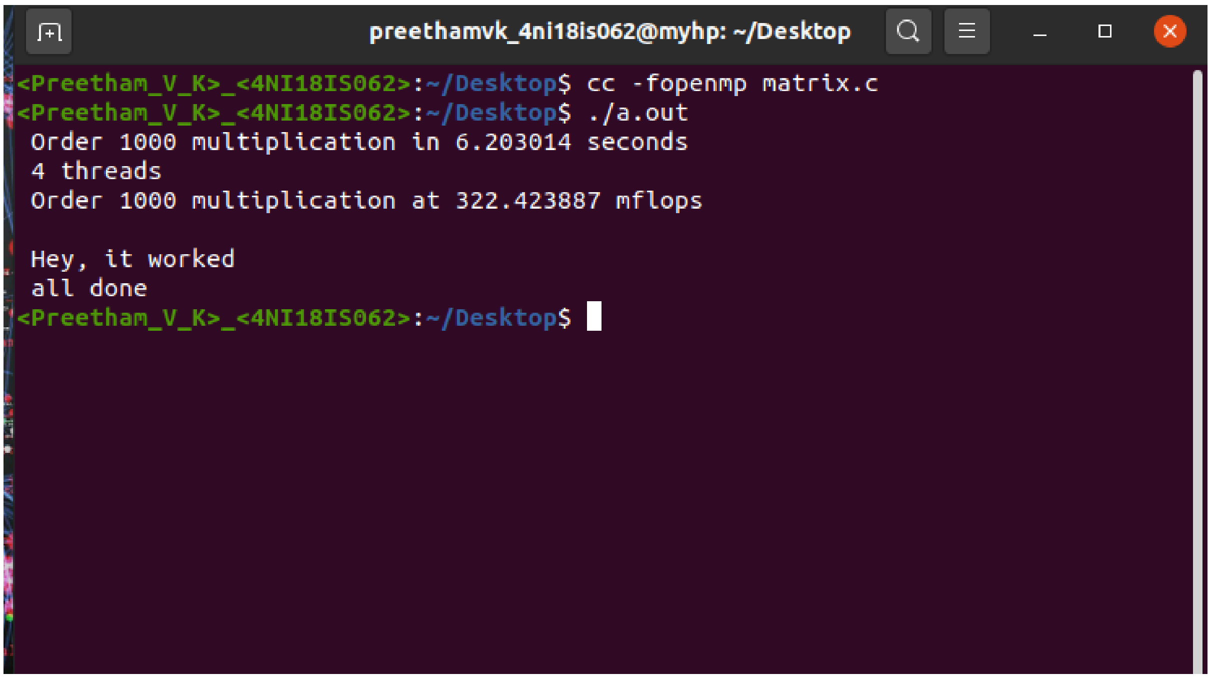
Program:

```
#include <malloc.h>
#include <stdio.h>
#include <omp.h>
#define ORDER 1000
#define AVAL 3.0
#define BVAL 5.0
#define TOL 0.001
int main(int argc, char *argv[])
{
int Ndim, Pdim, Mdim; /* A[N][P], B[P][M], C[N][M] */
int i,j,k;
double *A, *B, *C, cval, tmp, err, errsq;
 double dN, mflops;
double start_time, run_time;
Ndim = ORDER;
Pdim = ORDER;
Mdim = ORDER;
A = (double *)malloc(Ndim*Pdim*sizeof(double));
 B = (double *)malloc(Pdim*Mdim*sizeof(double));
 C = (double *)malloc(Ndim*Mdim*sizeof(double));
/* Initialize matrices */
for (i=0; i<Ndim; i++)
for (j=0; j<Pdim; j++)
*(A+(i*Ndim+j)) = AVAL;
for (i=0; i<Pdim; i++)
for (j=0; j<Mdim; j++)
*(B+(i*Pdim+j)) = BVAL;
for (i=0; i<Ndim; i++)
for (j=0; j<Mdim; j++)
```

```
*(C+(i*Ndim+j)) = 0.0;
start_time = omp_get_wtime();
/* Do the matrix product */
#pragma omp parallel for private(tmp, i, j, k)
for (i=0; i<Ndim; i++){
for (j=0; j<Mdim; j++){
 tmp = 0.0;
for(k=0;k<Pdim;k++){
/* C(i,j) = sum(over k) A(i,k) * B(k,j) */
tmp += *(A+(i*Ndim+k)) * *(B+(k*Pdim+j));
}
*(C+(i*Ndim+j)) = tmp;
}
}
/* Check the answer */
run_time = omp_get_wtime() - start_time;
printf(" Order %d multiplication in %f seconds \n", ORDER,
run_time);
 printf(" %d threads\n",omp_get_max_threads());
 dN = (double)ORDER;
 mflops = 2.0 * dN * dN * dN/(1000000.0* run_time);
 printf(" Order %d multiplication at %f mflops\n", ORDER,mflops);
cval = Pdim * AVAL * BVAL;
errsq = 0.0;
for (i=0; i<Ndim; i++){
for (j=0; j<Mdim; j++){
err = *(C+i*Ndim+j) - cval;
 errsq += err * err;
}
}
if (errsq > TOL)
```

```
printf("\n Errors in multiplication: %f",errsq);

else

printf("\n Hey, it worked");

printf("\n all done \n");

}
```

Compilation and Outputs:



```
<Preetham_V_K>_<4NI18IS062>:~/Desktop$ cc -fopenmp matrix.c
<Preetham_V_K>_<4NI18IS062>:~/Desktop$ ./a.out
 Order 1000 multiplication in 6.203014 seconds
 4 threads
 Order 1000 multiplication at 322.423887 mflops

 Hey, it worked
 all done
<Preetham_V_K>_<4NI18IS062>:~/Desktop$
```

PART – B

1. **Read status of eight input bits from the Logic Controller Interface and display FF if it is even parity bits otherwise displays 00. Also display number of 1,s in the input data.**

   Program:

```
INITDS MACRO

MOV AX,@DATA

MOV DS,AX

ENDM

INIT8255 MACRO

MOV AL,CW

MOV DX,CR

OUT DX,AL

ENDM

INPB MACRO

MOV DX,PB

IN AL,DX

ENDM

INPC MACRO

MOV DX,PC

IN AL,DX

ENDM

OUTPA MACRO

MOV DX,PA

OUT DX,AL

ENDM

DISP_MSG MACRO

MOV AH,9

LEA DX,MSG

INT 21H

ENDM
```

```
EXIT MACRO
MOV AH,4CH
INT 21H
ENDM

.MODEL SMALL
.STACK
.DATA
PA EQU 0DC50H
PB EQU 0DC51H
PC EQU 0DC52H
CR EQU 0DC53H
CW DB 82H
.CODE
INITDS
INIT8255
INPB
MOV BL,AL
MOV BH,0
MOV CX,8
NEXTBIT:ROR AL,1
JNC NEXT
INC BH
NEXT:LOOP NEXTBIT
MOV AL,BL
OR AL,AL
JPE DISPFF
MOV AL,00H
DISP:OUTPA
CALL DELAY
MOV AL,BH
```

```
OUTPA

EXIT

DISPFF:MOV AL,0FFH

JMP DISP

DELAY PROC

MOV AX,0FFFH

B2:MOV CX,0FFFFH

B1:LOOP B1

DEC AX

JNZ B2

RET

DELAY ENDP

END
```

2. Perform the following functions using the Logic Controller Interface.

a. BCD up-down counter

```
.MODEL SMALL

.STACK

.DATA

PA EQU 0DC50H

CW DB 80H

CWR EQU 0DC53H

.CODE

MOV AX,@DATA

MOV DS,AX

MOV AL,CW

MOV DX,CWR

OUT DX,AL

MOV AL,0

UP:MOV DX,PA

OUT DX,AL

CALL DELAY

ADD AL,1

DAA

CMP AL,99H

JNE UP

DOWN:MOV DX,PA

OUT DX,AL

CALL DELAY

ADD AL,99H

DAA

CMP AL,99H

JNE DOWN

MOV AH,4CH

INT 21H
```

```
DELAY PROC

MOV BX,0FFFH

B2:MOV CX,0FFFFH

B1:LOOP B1

DEC BX

JNZ B2

RET

DELAY ENDP

END
```

b. Ring counter

```
.MODEL SMALL
.DATA
PA EQU 0DC50H
PB EQU 0DC51H
PC EQU 0DC52H
CWR DW 0DC53H
.CODE
MOV AX,@DATA
MOV DS,AX
MOV AL,80H
MOV DX,CWR
OUT DX,AL
MOV AL,01H
MOV CX,16
UP:MOV DX,PA
OUT DX,AL
CALL DELAY
ROL AL,01
LOOP UP
QUIT:MOV AH,4CH
INT 21H
DELAY PROC
PUSH CX
PUSH BX
MOV CX,0FFFFH
UP2:MOV BX,0FFFH
UP1:NOP
DEC BX
JNZ UP1
LOOP UP2
```

```
POP BX

POP CX

RET

DELAY ENDP

END
```

c. Jonson counter

```
.MODEL SMALL
.DATA
PA EQU 0DC50H
PB EQU 0DC51H
PC EQU 0DC52H
CWR DW 0DC53H
.CODE
MOV AX,@DATA
MOV DS,AX
MOV AL,80H
MOV DX,CWR
OUT DX,AL
MOV AL,00H
MOV CX,16
UP:MOV DX,PA
OUT DX,AL
CALL DELAY
ROL AL,01
XOR AL,01H
LOOP UP
QUIT:MOV AH,4CH
INT 21H
DELAY PROC
PUSH CX
PUSH BX
MOV CX,0FFFFH
UP2:MOV BX,0FFFH
UP1:NOP
DEC BX
JNZ UP1
```

```
LOOP UP2

POP BX

POP CX

RET

DELAY ENDP

END
```

3. Display message FIRE and HELP alternately with flickering effects on a seven segment display interface for a suitable period of time.

Program:

```
.DATA

FIR DB 86H,88H,0F9H,8EH

HEL DB 8CH,0C7H,86H,89H


.CODE

START:  MOV AX,@DATA

        MOV DS,AX

        MOV DX,303H

        MOV AL,80H

        OUT DX,AL

        MOV AH,0AH

LP:     MOV BX,00H

        LEA SI,FIR

LP1:    MOV CX,07H

LP2:    MOV DX,301H

        MOV AL,SI[BX]

        ROR AL,CL

        OUT DX,AL

        MOV DX,302H

        MOV AL,0FFH

        OUT DX,AL

        MOV AL,00H

        OUT DX,AL

        DEC CX

        JNS LP2

        INC BX

        CMP BX,04H

        JB LP1
```

```
          CALL DELAY1
          MOV BX,00H
          LEA SI,HEL
LP3:      MOV CX,07H
LP4:      MOV DX,301H
          MOV AL,SI[BX]
          ROR AL,CL
          OUT DX,AL
          MOV DX,302H
          MOV AL,0FFH
          OUT DX,AL
          MOV AL,00H
          OUT DX,AL
          DEC CX
          JNS LP4
          INC BX
          CMP BX,04H
          JB LP3
          CALL DELAY1
          DEC AH
          JNS LP
         MOV AH,4CH
        INT 21H
        DELAY1 PROC
        PUSH CX
        PUSH BX
        MOV BX,0AAAH
LP5:    LOOP LP5
        DEC BX
        JNZ LP5
        POP BX
        POP CX

        RET

        DELAY1 ENDP

END START
```

4. Scan 3X8 Keypad for key closure and to store the code of the key pressed in a memory location or display it on the screen. Also display row and column of the key pressed.

Program:

```
.MODEL SMALL
.DATA
MSG DB "0123456789ABCDEFGHIJ"
RD DB 13,10,"READ CHARACTER IS = $"
RW DB 13,10,"ROW NUMBER IS = "
ROW DB ?
CL1 DB 13,10,"COLUMN NUMBER IS = "
COL DB ?,'$'
EN DB 13,10,"ENTER 20 CHARACTERS FROM KEYPAD.$"

.CODE
START:  MOV AX,@DATA
        MOV DS,AX
        MOV DX,303H
        MOV AL,90H
        OUT DX,AL
        LEA DX,EN
        MOV AH,09H
        INT 21H
        MOV CX,14H
LP:     MOV DX,302H
        MOV AL,07H
        OUT DX,AL
        MOV DX,300H
LP1:    IN AL,DX
        CMP AL,00H
        JE LP1
        CALL CVT
        MOV BX,0403H
LP2:    MOV AL,BH
        MOV DX,302H
        OUT DX,AL
        MOV DX,300H
        IN AL,DX
        ROR BH,01H
        DEC BL
        CMP AL,00H
        JE LP2
        ADD BL,'1'
        MOV COL,BL
        CALL DISP
        LOOP LP
        MOV AH,4CH
        INT 21H

        CVT PROC
        PUSH CX
        MOV CX,08H
LP3:      ROL AL,01H
        JC LP4
        LOOP LP3
LP4:      ADD CL,'0'
        MOV ROW,CL
        POP CX
        RET
        CVT ENDP

        DISP PROC
        MOV AL,COL
        SUB AL,'1'
        MOV BL,08H
        MOV AH,00H
        MUL BL
        MOV BL,ROW
        SUB BL,'1'
        ADD AL,BL
        MOV BX,AX
        LEA DX,RD
        MOV AH,09H
        INT 21H
        LEA SI,MSG
        MOV DL,SI[BX]
        MOV AH,02H
        INT 21H
        LEA DX,RW
        MOV AH,09H
        INT 21H
        PUSH CX
        PUSH BX
        MOV BX,011H
LP5:      LOOP LP5
        DEC BX
        JNZ LP5
        POP BX
        POP CX
        RET
        DISP ENDP
END START
```