Lab Assignment-3

Aim

To write an assembly language programming using 8086 instruction set to find the sum of odd and even numbers in the array.

Algorithm

Input – number of array elements (count).
Enter the elements to be summed in the array.

Process → Compare the odd and even numbers.
Conclusion → By the above process, we can find the sum of odd and even numbers in the array.

Team Members

Mothishwaran.C (19MID0017)
Prashanth.S (19MID0020)
Saksham Verma (19MID0023)
Tharun.S (19MID0031)
Sabarinathan R.V(19MID0057)

Program

ODDEVEN.ASM

```
.model small
.stack 64
.data
.code
start : MOV AX, @data
        MOV DS, AX
        MOV SI, 3000h
        MOV DI, 3500h
        MOV CH, 00h
        MOV AX, 0000h
        MOV CL, [SI]
        INC SI
eve:    MOV BL, [SI]
        TEST BL, 01
        JNZ odd
        ADD AL, BL
        INC SI
        LOOP eve
        CMP CL,00h
        JNC terminate
odd:    ADD AH,BL
        INC SI
        LOOP eve
        CMP CL,00h
        JNC terminate
terminate : MOV [DI], AL
            INC DI
            MOV [DI], AH
            HLT
            end
```

Mounting the drive

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

Z:\>c:
```

Writing the code

```
C:\>edit oddeven.asm_
```

```
File  Edit  Search  View  Options  Help
                    C:\ODDEVEN.ASM
.model small
.stack 64
.data
.code
start : MOV AX, @data
        MOV DS, AX
        MOV SI, 3000h
        MOV DI, 3500h
        MOV CH, 00h
        MOV AX, 0000h
        MOV CL, [SI]
        INC SI
eve:    MOV BL, [SI]
        TEST BL, 01
        JNZ odd
        ADD AL, BL
        INC SI
        LOOP eve
        CMP CL,00h
        JNC terminate
odd:    ADD AH,BL
        INC SI
        LOOP eve
        CMP CL,00h
        JNC terminate
terminate : MOV [DI], AL
            INC DI
            MOV [DI], AH
            HLT
            end
Commands for manipulating files
```

Compiling the Code

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

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

  51662 + 464882 Bytes symbol space free

       0 Warning Errors
       0 Severe  Errors
```

Linking the object Code

```
C:\>link oddeven.obj

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

Run File [ODDEVEN.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
```

Debugging the .exe file

```
C:\>debug oddeven.exe
-t
```

Execution of Start segment

```
AX=076D  BX=0000  CX=0034  DX=0000  SP=0040  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=076E  CS=076A  IP=0003   NV UP EI PL NZ NA PO NC
076A:0003 8ED8          MOV     DS,AX
-t

AX=076D  BX=0000  CX=0034  DX=0000  SP=0040  BP=0000  SI=0000  DI=0000
DS=076D  ES=075A  SS=076E  CS=076A  IP=0005   NV UP EI PL NZ NA PO NC
076A:0005 BE0030        MOV     SI,3000   Source Index Register (SI)
-t

AX=076D  BX=0000  CX=0034  DX=0000  SP=0040  BP=0000  SI=3000  DI=0000
DS=076D  ES=075A  SS=076E  CS=076A  IP=0008   NV UP EI PL NZ NA PO NC
076A:0008 BF0035        MOV     DI,3500   Destination Index Register (DI)
-t

AX=076D  BX=0000  CX=0034  DX=0000  SP=0040  BP=0000  SI=3000  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=000B   NV UP EI PL NZ NA PO NC
076A:000B B500          MOV     CH,00     CX counter's higher nibble (CH)
-t

AX=076D  BX=0000  CX=0034  DX=0000  SP=0040  BP=0000  SI=3000  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=000D   NV UP EI PL NZ NA PO NC
076A:000D B80000        MOV     AX,0000   Accumulator Register (AX)
```

```
AX=0000  BX=0000  CX=0034  DX=0000  SP=0040  BP=0000  SI=3000  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0010   NV UP EI PL NZ NA PO NC
076A:0010 8A0C          MOV     CL,[SI]   CX's Counter lower Nibble  DS:3000=FF
-t                                                                   [CL]
```

SI
FF
3000

```
AX=0000  BX=0000  CX=00FF  DX=0000  SP=0040  BP=0000  SI=3000  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0012   NV UP EI PL NZ NA PO NC
076A:0012 46            INC     SI
-t
```

Increment

```
AX=0000  BX=0000  CX=00FF  DX=0000  SP=0040  BP=0000  SI=3001  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0013   NV UP EI PL NZ NA PO NC
```

Entering and Verifying the entered data

```
-t

AX=0000  BX=0000  CX=0034  DX=0000  SP=0040  BP=0000  SI=3000  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0010   NV UP EI PL NZ NA PO NC
076A:0010 8A0C          MOV     CL,[SI]                          DS:3000=04
-e ds:3000 ← elements
076D:3000  04.4   [83.1    C4.2    06.3    C4.4]  Elements in the array

-d ds:3000
076D:3000  04 [01 02 03 04]5E F8 26-89 47 06 8B 36 22 21 D1   .....^.&.G..6"!.
076D:3010  E6 8B 1E 38 21 C7 00 00-00 8B 1E 08 27 C7 00 00   ...8!.......'...
076D:3020  00 8B 1E 22 21 D1 E3 D1-E3 8B 3E EC 26 2B C0 89   ..."!.....>.&+..
076D:3030  41 02 89 01 8B 1E 22 21-8B 3E FE 26 C6 01 00 EB   A....."!.>.&....
076D:3040  45 90 8B 1E 22 21 D1 E3-D1 E3 8B 3E EC 26 A1 3A   E..."!.....>.&.:
076D:3050  21 8B 16 3C 21 89 01 89-51 02 C4 5E FC 26 8A 47   !..<!...Q..^.&.G
076D:3060  05 8B 1E 22 21 8B 3E FE-26 88 01 8B F3 D1 E6 8B   ..."!.>.&.......
076D:3070  1E 38 21 8B 7E FC 26 8B-45 08 89 00 8B 1E 08 27   .8!.~.&.E......'
-t
            Loop counter

AX=0000  BX=0000  CX=0004  DX=0000  SP=0040  BP=0000  SI=3000  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0012   NV UP EI PL NZ NA PO NC
```

Executing the remaining start segment

```
076A:0012 46            INC     SI
-t                                   Now SI is pointing to the
                                     1st element in the array
AX=0000  BX=0000  CX=0004  DX=0000  SP=0040  BP=0000  SI=3001  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0013   NV UP EI PL NZ NA PO NC
```

Execution of eve segment

```
AX=0000  BX=0000  CX=0004  DX=0000  SP=0040  BP=0000  SI=3001  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0013   NV UP EI PL NZ NA PO NC
076A:0013 8A1C          MOV     BL,[SI]   BL => 1st element in the  DS:3001=01
-t                                        [i.e 1]  array

AX=0000  BX=0001  CX=0004  DX=0000  SP=0040  BP=0000  SI=3001  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0015   NV UP EI PL NZ NA PO NC
076A:0015 F6C301        TEST    BL,01  AND  0001
-t                                          0001
                                            ----
                                            0001 ← odd

AX=0000  BX=0001  CX=0004  DX=0000  SP=0040  BP=0000  SI=3001  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0018   NV UP EI PL NZ NA PO NC
076A:0018 750A          JNZ     0024
-t                                  [Jumping to the odd segment]

AX=0000  BX=0001  CX=0004  DX=0000  SP=0040  BP=0000  SI=3001  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0024   NV UP EI PL NZ NA PO NC
```

```
076A:0024 02E3          ADD     AH,BL
-t

AX=0100  BX=0001  CX=0004  DX=0000  SP=0040  BP=0000  SI=3001  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0026   NV UP EI PL NZ NA PO NC
076A:0026 46            INC     SI
-t

AX=0100  BX=0001  CX=0004  DX=0000  SP=0040  BP=0000  SI=3002  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0027   NV UP EI PL NZ NA PO NC
076A:0027 E2EA          LOOP    0013
```

0000
0001
0001

AH ⇒ 0001

Pointing to the 2nd array element

Running the loop (going to eve)

```
AX=0100  BX=0001  CX=0003  DX=0000  SP=0040  BP=0000  SI=3002  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0013   NV UP EI PL NZ NA PO NC
076A:0013 8A1C          MOV     BL,[SI]                          DS:3002=02
-t

AX=0100  BX=0002  CX=0003  DX=0000  SP=0040  BP=0000  SI=3002  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0015   NV UP EI PL NZ NA PO NC
076A:0015 F6C301        TEST    BL,01
-t

AX=0100  BX=0002  CX=0003  DX=0000  SP=0040  BP=0000  SI=3002  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0018   NV UP EI PL ZR NA PE NC
076A:0018 750A          JNZ     0024
-t

AX=0100  BX=0002  CX=0003  DX=0000  SP=0040  BP=0000  SI=3002  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=001A   NV UP EI PL ZR NA PE NC
076A:001A 02C3          ADD     AL,BL
```

BL ⇒ 2nd array element [i.e 2]

0010
0001
0000 ⇒ even

condition fails

0000
0010
0010

AL ⇒ 0010

```
AX=0102  BX=0002  CX=0003  DX=0000  SP=0040  BP=0000  SI=3002  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=001C   NV UP EI PL NZ NA PO NC
076A:001C 46            INC     SI
-t

AX=0102  BX=0002  CX=0003  DX=0000  SP=0040  BP=0000  SI=3003  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=001D   NV UP EI PL NZ NA PE NC
076A:001D E2F4          LOOP    0013
-t

AX=0102  BX=0002  CX=0002  DX=0000  SP=0040  BP=0000  SI=3003  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0013   NV UP EI PL NZ NA PE NC
076A:0013 8A1C          MOV     BL,[SI]                          DS:3003=03
-t

AX=0102  BX=0003  CX=0002  DX=0000  SP=0040  BP=0000  SI=3003  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0015   NV UP EI PL NZ NA PE NC
076A:0015 F6C301        TEST    BL,01
```

Points to 3rd array element [i.e 3]

Calling the eve loop

BL contains 0011

0011
0001
0001 ⇒ odd

```
AX=0102  BX=0003  CX=0002  DX=0000  SP=0040  BP=0000  SI=3003  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0018   NV UP EI PL NZ NA PO NC
076A:0018 750A          JNZ     0024
-       t
```

jumping to odd

```
AX=0102  BX=0003  CX=0002  DX=0000  SP=0040  BP=0000  SI=3003  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0024   NV UP EI PL NZ NA PO NC
076A:0024 02E3          ADD     AH,BL
-t
```

0001
0011
0100 ⇒ 4

AH ⇒ 4

```
AX=0402  BX=0003  CX=0002  DX=0000  SP=0040  BP=0000  SI=3003  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0026   NV UP EI PL NZ NA PO NC
076A:0026 46            INC     SI
-t
```

Points to the 4th array element

```
AX=0402  BX=0003  CX=0002  DX=0000  SP=0040  BP=0000  SI=3004  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0027   NV UP EI PL NZ NA PO NC
076A:0027 E2EA          LOOP    0013
```

calling the eve

```
AX=0402  BX=0003  CX=0001  DX=0000  SP=0040  BP=0000  SI=3004  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0013   NV UP EI PL NZ NA PO NC
076A:0013 8A1C          MOV     BL,[SI]                         DS:3004=04
-t
```

BL ⇒ 4th array element

[BL ⇒ 4]

```
AX=0402  BX=0004  CX=0001  DX=0000  SP=0040  BP=0000  SI=3004  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0015   NV UP EI PL NZ NA PO NC
076A:0015 F6C301        TEST    BL,01
-t
```

0100
0001
0000 → even

```
AX=0402  BX=0004  CX=0001  DX=0000  SP=0040  BP=0000  SI=3004  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0018   NV UP EI PL ZR NA PE NC
076A:0018 750A          JNZ     0024
-t
```

condition fails

```
AX=0402  BX=0004  CX=0001  DX=0000  SP=0040  BP=0000  SI=3004  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=001A   NV UP EI PL ZR NA PE NC
076A:001A 02C3          ADD     AL,BL
```

0010
0100
0110 → 6

[AL ⇒ 6]

```
AX=0406  BX=0004  CX=0001  DX=0000  SP=0040  BP=0000  SI=3004  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=001C   NV UP EI PL NZ NA PE NC
076A:001C 46            INC     SI
-t

AX=0406  BX=0004  CX=0001  DX=0000  SP=0040  BP=0000  SI=3005  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=001D   NV UP EI PL NZ NA PE NC
076A:001D E2F4          LOOP    0013
-t
```

calling the eve loop

But the counter ends the loop

```
AX=0406  BX=0004  CX=0000  DX=0000  SP=0040  BP=0000  SI=3005  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=001F   NV UP EI PL NZ NA PE NC
076A:001F 80F900        CMP     CL,00
-t
```

true condition

```
AX=0406  BX=0004  CX=0000  DX=0000  SP=0040  BP=0000  SI=3005  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0022   NV UP EI PL ZR NA PE NC
076A:0022 730A          JNB     002E
```

jump to terminate

```
AX=0406  BX=0004  CX=0000  DX=0000  SP=0040  BP=0000  SI=3005  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=002E   NV UP EI PL ZR NA PE NC
076A:002E 8805          MOV     [DI],AL
-t

AX=0406  BX=0004  CX=0000  DX=0000  SP=0040  BP=0000  SI=3005  DI=3500
DS=076D  ES=075A  SS=076E  CS=076A  IP=0030   NV UP EI PL ZR NA PE NC
076A:0030 47            INC     DI
-d ds:3500
076D:3500  06 E5 5D C3 E8 C9 0C A3-26 22 05 03 00 2B D2 01   ..].....&"...+..
076D:3510  06 B6 25 11 16 B8 25 A1-10 19 24 FE 3D 9C 00 75   ..%...%...$.=..u
076D:3520  05 E8 F6 95 EB 3D 83 3E-AE 07 00 74 03 E8 38 FD   .....=.>...t..8.
076D:3530  A1 10 19 24 FE 3D 98 00-74 26 76 03 E9 99 00 3D   ...$.=..t&v....=
076D:3540  80 00 74 2C 3D 82 00 74-27 3D 8A 00 74 66 3D 8C   ..t,=..t'=..tf=.
076D:3550  00 74 29 3D 94 00 74 36-3D 96 00 74 2B E9 89 00   .t)=..t6=..t+...
076D:3560  E8 3D F6 83 3E 26 22 01-75 03 E9 A7 00 E9 63 FF   .=..>&".u.....c.
076D:3570  E8 7F E1 EB EE 90 E8 3B-F9 EB E8 90 E8 EF F9 EB   .......;........
```

AL ⇒ 6 [even number sum]

sum of even numbers

```
AX=0406  BX=0004  CX=0000  DX=0000  SP=0040  BP=0000  SI=3005  DI=3501
DS=076D  ES=075A  SS=076E  CS=076A  IP=0031   NV UP EI PL NZ NA PO NC
076A:0031 8825          MOV     [DI],AH
-t

AX=0406  BX=0004  CX=0000  DX=0000  SP=0040  BP=0000  SI=3005  DI=3501
DS=076D  ES=075A  SS=076E  CS=076A  IP=0033   NV UP EI PL NZ NA PO NC
076A:0033 F4            HLT
-d ds:3500
076D:3500  06 04 5D C3 E8 C9 0C A3-26 22 05 03 00 2B D2 01   ..].....&"...+..
076D:3510  06 B6 25 11 16 B8 25 A1-10 19 24 FE 3D 9C 00 75   ..%...%...$.=..u
076D:3520  05 E8 F6 95 EB 3D 83 3E-AE 07 00 74 03 E8 38 FD   .....=.>...t..8.
076D:3530  A1 10 19 24 FE 3D 98 00-74 26 76 03 E9 99 00 3D   ...$.=..t&v....=
076D:3540  80 00 74 2C 3D 82 00 74-27 3D 8A 00 74 66 3D 8C   ..t,=..t'=..tf=.
076D:3550  00 74 29 3D 94 00 74 36-3D 96 00 74 2B E9 89 00   .t)=..t6=..t+...
076D:3560  E8 3D F6 83 3E 26 22 01-75 03 E9 A7 00 E9 63 FF   .=..>&".u.....c.
076D:3570  E8 7F E1 EB EE 90 E8 3B-F9 EB E8 90 E8 EF F9 EB   .......;........
```

AH ⇒ 4 [Odd number sum]

(end)

[16 bits] AX (Accumulator Register)

AL [8 bits]	AH [8 bits]
Even number sum	Odd number sum

Video Link

[MPU LA 3 - YouTube]