

## 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
1  .model small
2  .stack 64
3  .data
4  .code
5  start : MOV AX, @data
6          MOV DS, AX
7          MOV SI, 3000h
8          MOV DI, 3500h
9          MOV CH, 00h
10         MOV AX, 0000h
11         MOV CL, [SI]
12         INC SI
13 eve:     MOV BL, [SI]
14         TEST BL, 01
15         JNZ odd
16         ADD AL, BL
17         INC SI
18         LOOP eve
19         CMP CL, 00h
20         JNC terminate
21 odd:     ADD AH, BL
22         INC SI
23         LOOP eve
24         CMP CL, 00h
25         JNC terminate
26 terminate : MOV [DI], AL
27             INC DI
28             MOV [DI], AH
29             HLT
30         end
```

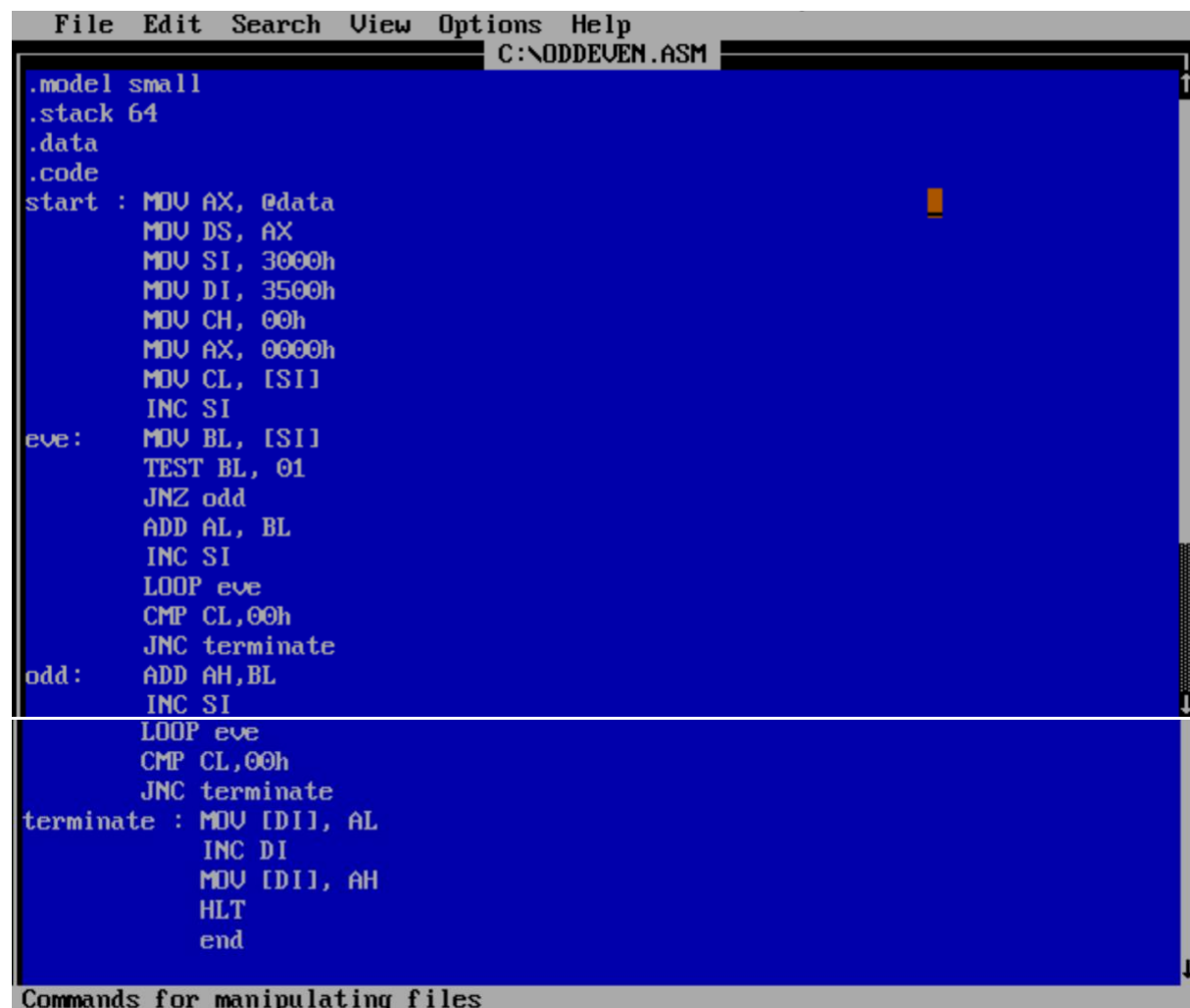
## Mounting the drive

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

Z:\>c:
```

## Writing the code

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

The screenshot shows the MASM editor window with the title bar 'File Edit Search View Options Help' and the file name 'C:\NODDEVEN.ASM'. The code is written in assembly language and includes comments. The code defines a small model, sets up a stack of 64 bytes, and defines data and code segments. The code starts at a label 'start' and moves the address of the data segment into the AX register, then into the DS register. It then moves the address of the data segment into the SI register, and the address of the code segment into the DI register. It then moves the value 0000h into the AX register, and the value 00h into the CL register. It then increments the SI register. The code then enters a loop labeled 'eve' where it moves the value at the address pointed to by SI into the BL register, tests the BL register for the 01h bit, and jumps to the 'odd' label if not zero. It then increments the AL register by the value in BL, increments the SI register, and loops back to the 'eve' label. The code then compares the CL register with 00h and jumps to the 'terminate' label if not carry. The 'odd' label increments the AH register by the value in BL and increments the SI register. The 'terminate' label moves the value in the AL register into the DI register, increments the DI register, moves the value in the AH register into the DI register, and then halts the program. The code ends with the 'end' instruction.

```
.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  NU UP EI PL NZ NA PO NC  
076A:0003 8EDB      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  NU 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  NU 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  NU 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  NU UP EI PL NZ NA PO NC  
076A:000D B80000    MOV     AX,0000 Accumulator Register (AX)  
-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  NU UP EI PL NZ NA PO NC  
076A:0010 8A0C      SI SI MOV     CL,[SI] CX's Counter lower Nibble DS:3000=FF  
-t [FF] [CL]  
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  NU UP EI PL NZ NA PO NC  
076A:0012 46        INC     SI  
-t  
  
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  NU 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 NU UP EI PL NZ NA PO NC
076A:0010 8A0C          MOV     CL,[SI]          DS:3000=04
-e ds:3000 4 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 8B 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 NU UP EI PL NZ NA PO NC
```

## Executing the remaining start segment

```
076A:0012 46          INC     SI
-t
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 NU UP EI PL NZ NA PO NC
```

*Now SI is pointing to the 1st element in the array*

## 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 NU UP EI PL NZ NA PO NC
076A:0013 8A1C          MOV     BL,[SI] BL => 1st element in the array 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 NU UP EI PL NZ NA PO NC
076A:0015 F6C301      TEST     BL,01 AND 0001
-t 0001
0001
0001 ← add
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 NU UP EI PL NZ NA PO NC
076A:0018 750A          JNZ     0024 [Jumping to the odd segment]
-t
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 NU UP EI PL NZ NA PO NC
```

```

076A:0024 02E3      ADD     AH,BL      0000      AH → 0001
-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  NU 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  NU UP EI PL NZ NA PO NC
076A:0027 E2EA      LOOP    0013

```

Pointing to the 2<sup>nd</sup> 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  NU UP EI PL NZ NA PO NC
076A:0013 8A1C      MOV     BL,[SI]      DS:3002=02
-t

```

BL → 2<sup>nd</sup> array element [i.e 2]

```

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  NU UP EI PL NZ NA PO NC
076A:0015 F6C301    TEST    BL,01      0001
-t

```

0001  
0000 ⇒ even

```

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  NU UP EI PL ZR NA PE NC
076A:0018 750A      JNZ     0024
-t

```

Condition fails

```

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  NU UP EI PL ZR NA PE NC
076A:001A 02C3      ADD     AL,BL      0000
-t

```

0000  
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  NU UP EI PL NZ NA PO NC
076A:001C 46        INC     SI
-t

```

Points to 3<sup>rd</sup> array element [i.e 3]

```

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  NU UP EI PL NZ NA PE NC
076A:001D E2F4      LOOP    0013
-t

```

calling the eve loop

```

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  NU UP EI PL NZ NA PE NC
076A:0013 8A1C      MOV     BL,[SI]      DS:3003=03
-t

```

BL contains 0011

```

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  NU UP EI PL NZ NA PE NC
076A:0015 F6C301    TEST    BL,01      0011
-t

```

0011  
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  NU UP EI PL NZ NA PO NC
076A:0018 750A          JNZ     0024  jumping to odd
-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=0024  NU UP EI PL NZ NA PO NC
076A:0024 02E3          ADD     AH,BL  0001
                                     0011  AH → 4
                                     0100 ⇒ 4
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=0026  NU UP EI PL NZ NA PO NC
076A:0026 46          INC     SI  Points to the 4th array element
-t
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  NU 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  NU UP EI PL NZ NA PO NC
076A:0013 8A1C          MOV     BL,[SI]  BL → 4th array element [BL → 4]
-t
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  NU UP EI PL NZ NA PO NC
076A:0015 F6C301       TEST    BL,01  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  NU UP EI PL ZR NA PE NC
076A:0018 750A          JNZ     0024  condition fails
-t
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  NU 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  NU 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  NU UP EI PL NZ NA PE NC
076A:001D E2F4          LOOP    0013  calling the eve loop
But the counter ends the loop
-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=001F  NU UP EI PL NZ NA PE NC
076A:001F 80F900       CMP     CL,00  true condition
-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=0022  NU 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 NU UP EI PL ZR NA PE NC
076A:002E 8805          MOV     DI,AL    AL → 6 [even number sum]
-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 NU 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 ..l....&"...+..
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 .....;.....

```

```

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 NU UP EI PL NZ NA PO NC
076A:0031 8825          MOV     DI,AH    AH → 4 [odd number sum]
-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 NU UP EI PL NZ NA PO NC
076A:0033 F4          HLT     (end)
-d ds:3500
076D:3500 06 04 5D C3 E8 C9 0C A3-26 22 05 03 00 2B D2 01 ..l....&"...+..
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 .....;.....

```

[16 bits] AX (Accumulator Register)

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

**Video Link**

[MPU LA 3 - YouTube](#)