



**VIDYALANKAR INSTITUTE OF  
TECHNOLOGY DEPARTMENT OF  
COMPUTER ENGINEERING**

Lab Manual

Subject: MICROPROCESSOR

SEM- IV

**Suvarna Bhat**

**2021-2022**

|                                    |              |
|------------------------------------|--------------|
| <b>Subject</b>                     | MP           |
|                                    |              |
| <b>Semester</b>                    | IV           |
|                                    |              |
| <b>Academic Year</b>               | 2021-22      |
|                                    |              |
| <b>Software Requirements</b>       | TASM, DosBox |
|                                    |              |
| <b>Hardware Requirements</b>       | Desktops     |
|                                    |              |
| <b>Theory Faculty In-charge</b>    | Suvarna Bhat |
|                                    |              |
| <b>Practical Faculty In-charge</b> | Suvarna Bhat |
|                                    |              |
|                                    |              |

**Vidyalankar Institute of Technology**  
**DEPARTMENT OF COMPUTER**  
**ENGINEERING LAB CODE**

1. Students should report to the concerned labs as per the time table schedule.
2. Students who turn up late to the labs will in no case be permitted to perform the experiment scheduled for the day.
3. After completion of the experiment, certification of the concerned staff in-charge in the observation book is necessary.
4. Students should bring a note book of about 100 pages and should enter the readings/observations into the note book while performing the experiment.
5. The record of observations along with the detailed experimental procedure of the experiment performed in the immediate last session should be submitted and certified by the faculty member.
6. The group-wise division made in the beginning should be adhered to, and no mix up of student among different groups will be permitted later.
7. The components required pertaining to the experiment should be collected from the concerned Lab Assistants.
8. When the experiment is completed, students should disconnect the setup made by them, and should return all the components/instruments taken for the purpose. Any damage of the equipment or burn-out of components will be viewed seriously either by putting penalty or by dismissing the total group of students from the lab for the semester/year.
9. Students should be present in the labs for the total scheduled duration. Students are required to prepare thoroughly to perform the experiment coming to Laboratory. Procedure sheets/data sheets provided to the students groups should be maintained neatly and to be returned after the experiment.

**Course Outcome**

|   |
|---|
| Use appropriate instructions to program microprocessor to perform various task. |
| Develop the program in assembly/ mixed language for Intel 8086 processor.       |
| Demonstrate the execution and debugging of assembly/ mixed language program.    |

### List of Experiments

| <b>Sr. No.</b> | <b>Name of Experiment</b>   | <b>CO</b> |
|----------------|---|-----------|
| 1              | Study of tools used for programming of microprocessors (TASM)   | CO2       |
| 2              | Develop Assembly language program using 8086 microprocessor for addition, subtraction of 16/32 bit numbers                              | CO2       |
| 3              | Develop Assembly language program using 8086 microprocessor for 16bit multiplication, 16 bit division                                   | CO2       |
| 4              | Develop Assembly language program using 8086 microprocessor for finding largest/smallest number from block of 16 bit numbers            | CO2       |
| 5              | Develop a 8086 Assembly Language Program for Conversions– (a) Packed to Unpacked BCD (b) Unpacked to Packed BCD (c) Packed BCD to ASCII | CO2       |
| 6              | Develop Assembly language program using 8086 microprocessor for block copy, block exchange with and without string instructions         | CO2       |
| 7              | Develop assembly language program using 8086 microprocessor for I/O using INT N   | CO2       |
| 8              | 8086 Assembly Language Program to Count Odd and Even Numbers from the given block of Numbers  | CO2       |
| 9              | 8086 Assembly Language Program to Arrange the Numbers in Ascending Order  | CO2       |
| 10             | 8086 Assembly Language Program to check if user entered string is palindrome or not   | CO2       |

| <b>EXPERIMENT NO. 1</b> |   |
|-------------------------|---|
| <b>Title</b>            | Study of tools used for programming of microprocessors (TASM) |
| <b>Outcome</b>          | CO2   |
| <b>Theory</b>           | Study Experiment  |
| <b>Conclusion</b>       |   |

| EXPERIMENT NO. 2              |   |           |                               |  |  |        |           |          |       |     |           |      |     |         |
|-------------------------------|---|-----------|-------------------------------|--|--|--------|-----------|----------|-------|-----|-----------|------|-----|---------|
| Title                         | Develop Assembly language program using 8086 microprocessor for addition, subtraction of 16/32 bit numbers.   |           |                               |  |  |        |           |          |       |     |           |      |     |         |
| Outcome                       | CO2   |           |                               |  |  |        |           |          |       |     |           |      |     |         |
| Algorithm                     | <p>Algorithm:</p> <p><b>Addition of 2, 16-bit numbers</b></p> <ol style="list-style-type: none"><li>1) Start</li><li>2) Allocate some space for the result and operands in data segment</li><li>3) In code segment, store accumulator with 1<sup>st</sup> operand</li><li>4) Store B register with 2<sup>nd</sup> operand</li><li>5) Initialise C register with 0</li><li>6) Add the content of register B with accumulator and store the result in accumulator</li><li>7) If operation doesn't result in carry, goto 9</li><li>8) Increment content of C register</li><li>9) The result is stored in the required memory location</li><li>10) Stop</li></ol> <p><b>Subtraction of 2, 16-bit numbers</b></p> <ol style="list-style-type: none"><li>1) Start</li><li>2) Allocate some space for the result and operands in data segment</li><li>3) In code segment, store accumulator with 1<sup>st</sup> operand</li><li>4) Store B register with 2<sup>nd</sup> operand</li><li>5) Initialise C register with 0</li><li>6) Subtract the content of register B from accumulator and store the result in accumulator</li><li>7) If operation doesn't result in carry, goto 9</li><li>8) Increment content of C register</li><li>9) The result is stored in the required memory location</li><li>10) Stop</li></ol> |           |                               |  |  |        |           |          |       |     |           |      |     |         |
| Code                          | <table><tr><th colspan="3">Addition of 2, 16-bit numbers</th></tr><tr><th>Labels</th><th>Mnemonics</th><th>Operands</th></tr><tr><td>start</td><td>MOV</td><td>AX, @data</td></tr><tr><td>down</td><td>MOV</td><td>SUM, AX</td></tr></table> <p>Program:</p> <p>.model small</p>  |           | Addition of 2, 16-bit numbers |  |  | Labels | Mnemonics | Operands | start | MOV | AX, @data | down | MOV | SUM, AX |
| Addition of 2, 16-bit numbers |   |           |                               |  |  |        |           |          |       |     |           |      |     |         |
| Labels                        | Mnemonics   | Operands  |                               |  |  |        |           |          |       |     |           |      |     |         |
| start                         | MOV   | AX, @data |                               |  |  |        |           |          |       |     |           |      |     |         |
| down                          | MOV   | SUM, AX   |                               |  |  |        |           |          |       |     |           |      |     |         |

```

.stack 100H
.data
    n1 DW 1234H
    n2 DW 0F321H
    SUM DW ?
ends
.code
    start:  MOV AX, @data
            MOV DS, AX
            MOV AX, n1
            MOV BX, n2
            MOV CX, 00H
            ADD AX, BX
            JNC DOWN
            INC CX
    DOWN:  MOV SUM, AX
            MOV SUM+2, DX
            MOV AH, 4CH
            INT 21H
    end     start
end

```

### Subtraction of 2, 16-bit numbers

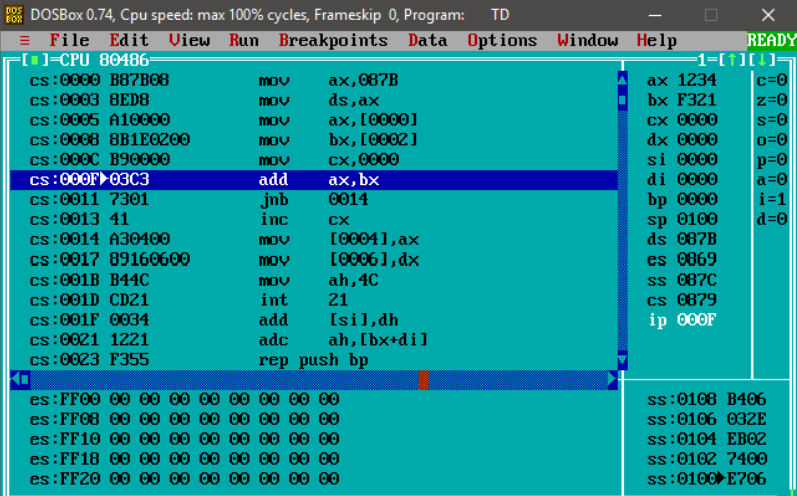
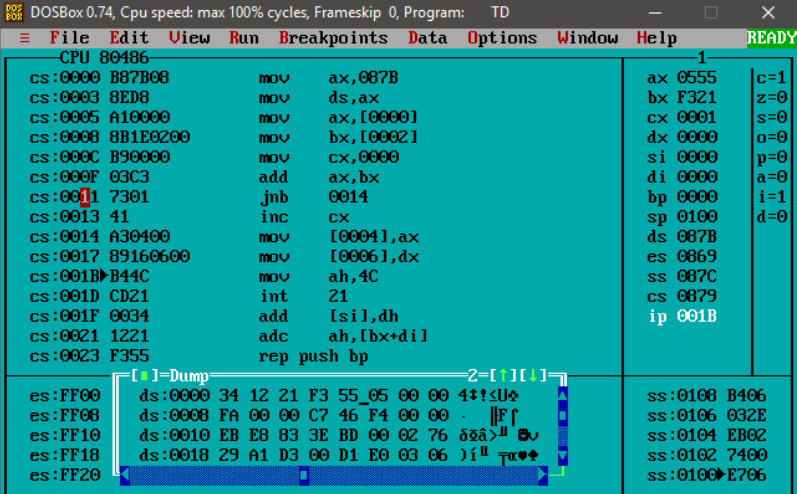
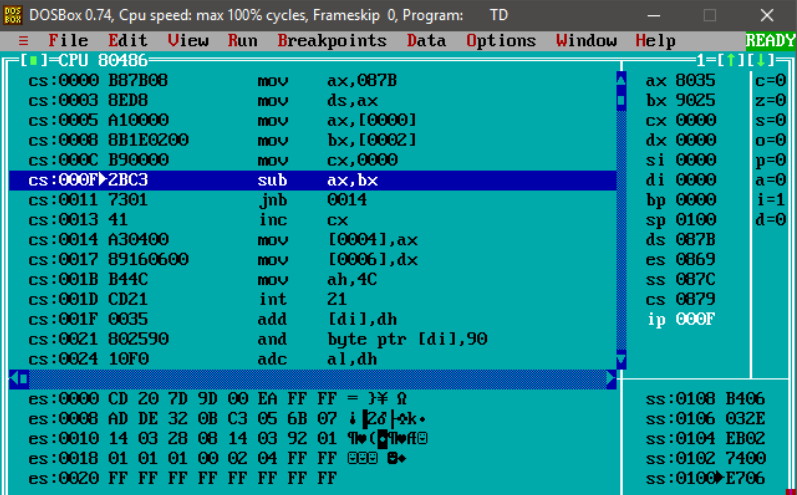
| Labels | Mnemonics | Operands  |
|--------|-----------|-----------|
| start  | MOV       | AX, @data |
| down   | MOV       | DIFF, AX  |

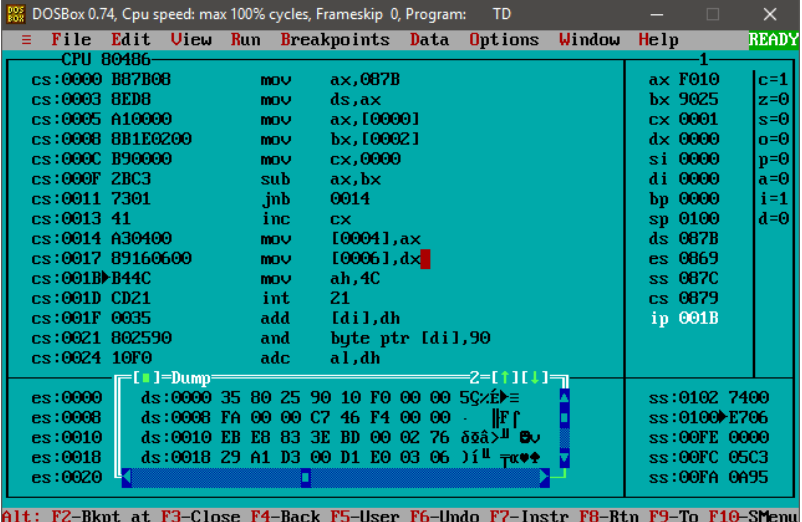
Program:

```

.model small
.stack 100H
.data
    n1 DW 0F321H
    n2 DW 1234H
    DIFF DW ?
ends
.code
    start:  MOV AX, @data
            MOV DS, AX
            MOV AX, n1
            MOV BX, n2
            MOV CX, 00H
            SUB AX, BX
            JNC DOWN
            INC CX
    DOWN:  MOV DIFF, AX
            MOV DIFF+2, DX
            MOV AH, 4CH
            INT 21H

```

|        |  |
|--------|--|
|        | <div>end      start</div> <div>end</div>   |
| Output | <div><div>Addition of 2, 16-bit numbers</div><div></div><div></div><div><div>Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu</div><div>Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu</div></div></div> <div><div>Subtraction of 2, 16-bit numbers</div><div></div><div><div>Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu</div><div>Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu</div></div></div> |

|                          |  |
|--------------------------|--|
|                          |  <p>The screenshot shows the DOSBox 0.74 interface with the CPU 80486 window open. The assembly code list on the left includes instructions like <code>mov ax,087B</code>, <code>mov ds,ax</code>, <code>mov ax,[0000]</code>, <code>mov bx,[0002]</code>, <code>mov cx,0000</code>, <code>sub ax,bx</code>, <code>jnb 0014</code>, <code>inc cx</code>, <code>mov [0004],ax</code>, <code>mov [0006],dx</code>, <code>mov ah,4C</code>, <code>int 21</code>, <code>add [di],dh</code>, <code>and byte ptr [di],90</code>, and <code>adc al,dh</code>. The register window on the right shows <code>ax: F010</code>, <code>bx: 9025</code>, <code>cx: 0001</code>, <code>dx: 0000</code>, <code>si: 0000</code>, <code>di: 0000</code>, <code>bp: 0000</code>, <code>sp: 0100</code>, <code>ds: 087B</code>, <code>es: 0869</code>, <code>ss: 087C</code>, <code>cs: 0879</code>, and <code>ip: 001B</code>. The status bar at the bottom indicates <code>Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu</code>.</p> |
| <p><b>Conclusion</b></p> | <ol style="list-style-type: none"> <li>1) For storing 16-bit data in 8086, we use the data type word DW.</li> <li>2) During both the operations, whenever the operation results in a carry, we increment the contents of C register to store the extra bit. Hence the whole result in case of addition is represented as CAX and during subtraction if C register is incremented, it implies the result is negative which can be confirmed observing the sign flag which would also be 1 (set).</li> <li>3) Jump instruction can be used to create a 'if' condition</li> <li>2) 4C is the service of int 21 that needs to be loaded in AH register for normal termination of program.</li> </ol>   |



| EXPERIMENT NO. 3 |   |          |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
|------------------|---|----------|--------|-----------|----------|-------|-----|----|------|-----|----|--|-----|----|--|------|----|--|-----|----|--|-----------|----|--|--|-----|--|--|----|
| <b>Title</b>     | Develop Assembly language program using 8086 microprocessor for 16bit multiplication, 16 bit division.  |          |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
| <b>Outcome</b>   | <b>CO2</b>  |          |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
| <b>Algorithm</b> | <p>Algorithm:</p> <p><b>Multiplication of 2, 16-bit numbers</b></p> <ol style="list-style-type: none"> <li>1) Start</li> <li>2) Allocate some space for the result and operands in data segment</li> <li>3) In code segment, store accumulator with 1<sup>st</sup> operand</li> <li>4) Store B register with 2<sup>nd</sup> operand</li> <li>5) Multiply the content of register B with accumulator. Higher order bits are stored in DX and lower order bits in AX.</li> <li>6) The result is stored in the required memory location</li> <li>7) Stop</li> </ol> <p><b>Division of 2, 16-bit numbers</b></p> <ol style="list-style-type: none"> <li>1) Start</li> <li>2) Allocate some space for the result and operands in data segment</li> <li>3) In code segment, store accumulator with 1<sup>st</sup> operand</li> <li>4) Store B register with 2<sup>nd</sup> operand</li> <li>5) Subtract the content of register pair DXAX with register BX. Quotient is stored in AX and Remainder is stored in DX.</li> <li>6) The result is stored in the required memory location</li> </ol> <p>Stop</p> |          |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
| <b>Code</b>      | <p><b>Multiplication of 2, 16-bit numbers</b></p> <table border="1"> <thead> <tr> <th>Labels</th><th>Mnemonics</th><th>Operands</th></tr> </thead> <tbody> <tr> <td>Start</td><td>MOV</td><td>AX</td></tr> <tr> <td>Down</td><td>MUL</td><td>DS</td></tr> <tr> <td></td><td>INT</td><td>BX</td></tr> <tr> <td></td><td>ENDS</td><td>DX</td></tr> <tr> <td></td><td>END</td><td>N1</td></tr> <tr> <td></td><td>END START</td><td>N2</td></tr> <tr> <td></td><td></td><td>PRO</td></tr> <tr> <td></td><td></td><td>AH</td></tr> </tbody> </table> <p>Program:</p> <pre> .model small .stack 100H .data     n1 dw 1234H     n2 dw 000FH     pro dw ? ends .code     START:    mov AX,@data </pre>  |          | Labels | Mnemonics | Operands | Start | MOV | AX | Down | MUL | DS |  | INT | BX |  | ENDS | DX |  | END | N1 |  | END START | N2 |  |  | PRO |  |  | AH |
| Labels           | Mnemonics   | Operands |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
| Start            | MOV   | AX       |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
| Down             | MUL   | DS       |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
|                  | INT   | BX       |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
|                  | ENDS  | DX       |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
|                  | END   | N1       |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
|                  | END START   | N2       |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
|                  |   | PRO      |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |
|                  |   | AH       |        |           |          |       |     |    |      |     |    |  |     |    |  |      |    |  |     |    |  |           |    |  |  |     |  |  |    |

```

                                mov DS,AX
                                mov AX,n1
                                mov BX,n2
                                mov CX,00H
                                mul BX
DOWN:                          mov pro,AX
                                mov pro+2,DX
                                mov AH,4CH
                                int 21H
                                end start
end

```

#### Division of 2, 16-bit numbers

| Labels | Mnemonics | Operands |
|--------|-----------|----------|
| Start  | MOV       | AX       |
| Down   | DIV       | DS       |
|        | INT       | BX       |
|        | ENDS      | CX       |
|        | END       | N1       |
|        | END START | N2       |
|        |           | Q        |
|        |           | R        |
|        |           | AH       |

Program:

```

.model small
.stack 100H
.data
    n1 dw 000AH
    n2 dw 0003H
    r dw ?
    q dw ?
ends
.code
    START:mov AX,@data
           mov DS,AX
           mov AX,n1
           mov BX,n2
           div bx
           mov q,AX
           mov r,DX
           mov AH,4CH
           int 21H
           end start

```

end

## Output

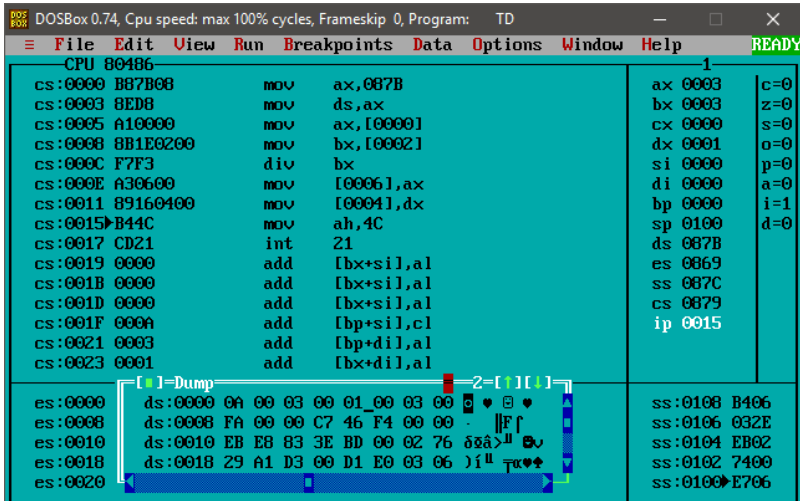
## Multiplication of 2, 16-bit numbers

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD
File Edit View Run Breakpoints Data Options Window Help READY
[CPU 80486]
cs:0000 B87B08 mov ax,0B7B ax 1234 c=0
cs:0003 8ED8 mov ds,ax bx 000F z=0
cs:0005 A10000 mov ax,[0000] cx 0000 s=0
cs:0008 8B1E0200 mov bx,[0002] dx 0000 o=0
cs:000C B90000 mov cx,0000 si 0000 p=0
cs:000F F7E3 mul bx di 0000 a=0
cs:0011 A30400 mov [0004],ax bp 0000 i=1
cs:0014 89160600 mov [0006],dx sp 0100 d=0
cs:0018 B44C mov ah,4C ds 0B7B
cs:001A CD21 int 21 es 0B69
cs:001C 0000 add [bx+si],al ss 0B7C
cs:001E 0000 add [bx+si],al cs 0B79
cs:0020 3412 xor al,12 ip 000F
cs:0022 0F000C str [si]
cs:0025 1101 adc [bx+di],ax
es:0000 CD 20 7D 9D 00 EA FF FF = }¥ ¤
es:0008 AD DE 32 0B C3 05 6B 07 i 2d -ak.
es:0010 14 03 28 08 14 03 92 01 70 (7)wff
es:0018 01 01 01 00 02 04 FF FF 88 80
es:0020 FF FF FF FF FF FF FF FF
ss:0108 B406
ss:0106 032E
ss:0104 EB02
ss:0102 7400
ss:0100 E706
Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu
```

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD
File Edit View Run Breakpoints Data Options Window Help READY
[CPU 80486]
cs:0000 B87B08 mov ax,0B7B ax 110C c=1
cs:0003 8ED8 mov ds,ax bx 000F z=0
cs:0005 A10000 mov ax,[0000] cx 0000 s=0
cs:0008 8B1E0200 mov bx,[0002] dx 0001 o=1
cs:000C B90000 mov cx,0000 si 0000 p=0
cs:000F F7E3 mul bx di 0000 a=0
cs:0011 A30400 mov [0004],ax bp 0000 i=1
cs:0014 89160600 mov [0006],dx sp 0100 d=0
cs:0018 B44C mov ah,4C ds 0B7B
cs:001A CD21 int 21 es 0B69
cs:001C 0000 add [bx+si],al ss 0B7C
cs:001E 0000 add [bx+si],al cs 0B79
cs:0020 3412 xor al,12 ip 0018
cs:0022 0F000C str [si]
cs:0025 1101 adc [bx+di],ax
[Dump]
ds:0000 34 12 0F 00 0C 11 01 00 47* 94
ds:0008 F8 00 00 C7 46 F4 00 00 . ||Ff
ds:0010 EB E8 B3 3E BD 00 02 76 5aâ>u 0u
ds:0018 29 A1 D3 00 D1 E0 03 06 iu Taw+
ss:0102 7400
ss:0100 E706
ss:00FE 0000
ss:00FC 05C3
ss:00FA 0A95
Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu
```

## Division of 2, 16-bit numbers

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD
File Edit View Run Breakpoints Data Options Window Help READY
[CPU 80486]
cs:0000 B87B08 mov ax,0B7B ax 000A c=0
cs:0003 8ED8 mov ds,ax bx 0003 z=0
cs:0005 A10000 mov ax,[0000] cx 0000 s=0
cs:0008 8B1E0200 mov bx,[0002] dx 0000 o=0
cs:000C F7F3 div bx si 0000 p=0
cs:000E A30600 mov [0006],ax di 0000 a=0
cs:0011 89160400 mov [0004],dx bp 0000 i=1
cs:0015 B44C mov ah,4C ds 0B7B
cs:0017 CD17 int 21 es 0B69
cs:0019 0000 add [bx+si],al ss 0B7C
cs:001B 0000 add [bx+si],al cs 0B79
cs:001D 0000 add [bx+si],al
cs:001F 000A add [bp+si],cl ip 000C
cs:0021 0003 add [bp+di],al
cs:0023 0001 add [bx+di],al
es:0000 CD 20 7D 9D 00 EA FF FF = }¥ ¤
es:0008 AD DE 32 0B C3 05 6B 07 i 2d -ak.
es:0010 14 03 28 08 14 03 92 01 70 (7)wff
es:0018 01 01 01 00 02 04 FF FF 88 80
es:0020 FF FF FF FF FF FF FF FF
ss:0108 B406
ss:0106 032E
ss:0104 EB02
ss:0102 7400
ss:0100 E706
Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu
```

|            |  <p>DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD</p> <p>File Edit View Run Breakpoints Data Options Window Help</p> <p>CPU 8086</p> <table><tr><th>Address</th><th>Instruction</th><th>Register/Value</th><th>Register/Value</th></tr><tr><td>cs:0000</td><td>BB7B08</td><td>mov ax,087B</td><td>ax 0003</td></tr><tr><td>cs:0003</td><td>8ED8</td><td>mov ds,ax</td><td>bx 0003</td></tr><tr><td>cs:0005</td><td>A10000</td><td>mov ax,[0000]</td><td>cx 0000</td></tr><tr><td>cs:0008</td><td>8B1E0200</td><td>mov bx,[0002]</td><td>dx 0001</td></tr><tr><td>cs:000C</td><td>F7F3</td><td>div bx</td><td>si 0000</td></tr><tr><td>cs:000E</td><td>A30600</td><td>mov [0006],ax</td><td>di 0000</td></tr><tr><td>cs:0011</td><td>89160400</td><td>mov [0004],dx</td><td>bp 0000</td></tr><tr><td>cs:0015</td><td>B44C</td><td>mov ah,4C</td><td>sp 0100</td></tr><tr><td>cs:0017</td><td>CD21</td><td>int 21</td><td>ds 087B</td></tr><tr><td>cs:0019</td><td>0000</td><td>add [bx+si],al</td><td>es 0869</td></tr><tr><td>cs:001B</td><td>0000</td><td>add [bx+si],al</td><td>ss 087C</td></tr><tr><td>cs:001D</td><td>0000</td><td>add [bx+si],al</td><td>cs 0879</td></tr><tr><td>cs:001F</td><td>0000</td><td>add [bp+si],cl</td><td>ip 0015</td></tr><tr><td>cs:0021</td><td>0003</td><td>add [bp+di],al</td><td></td></tr><tr><td>cs:0023</td><td>0001</td><td>add [bx+di],al</td><td></td></tr></table> <p>es:0000 ds:0000 0A 00 03 00 01_00 03 00 00 00 00 00 00 00 00 00</p> <p>es:0008 ds:0008 FA 00 00 C7 46 F4 00 00 00 00 00 00 00 00</p> <p>es:0010 ds:0010 EB E8 83 3E BD 00 02 76 88 8A 00 00 00 00</p> <p>es:001B ds:001B 29 A1 D3 00 D1 E0 03 06 00 00 00 00 00</p> <p>es:0020 ds:0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00</p> <p>ss:0108 B406</p> <p>ss:0106 032E</p> <p>ss:0104 EB02</p> <p>ss:0102 7400</p> <p>ss:0100 E706</p> <p>Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu</p> | Address        | Instruction    | Register/Value | Register/Value | cs:0000 | BB7B08 | mov ax,087B | ax 0003 | cs:0003 | 8ED8 | mov ds,ax | bx 0003 | cs:0005 | A10000 | mov ax,[0000] | cx 0000 | cs:0008 | 8B1E0200 | mov bx,[0002] | dx 0001 | cs:000C | F7F3 | div bx | si 0000 | cs:000E | A30600 | mov [0006],ax | di 0000 | cs:0011 | 89160400 | mov [0004],dx | bp 0000 | cs:0015 | B44C | mov ah,4C | sp 0100 | cs:0017 | CD21 | int 21 | ds 087B | cs:0019 | 0000 | add [bx+si],al | es 0869 | cs:001B | 0000 | add [bx+si],al | ss 087C | cs:001D | 0000 | add [bx+si],al | cs 0879 | cs:001F | 0000 | add [bp+si],cl | ip 0015 | cs:0021 | 0003 | add [bp+di],al |  | cs:0023 | 0001 | add [bx+di],al |  |
|------------|---|----------------|----------------|----------------|----------------|---------|--------|-------------|---------|---------|------|-----------|---------|---------|--------|---------------|---------|---------|----------|---------------|---------|---------|------|--------|---------|---------|--------|---------------|---------|---------|----------|---------------|---------|---------|------|-----------|---------|---------|------|--------|---------|---------|------|----------------|---------|---------|------|----------------|---------|---------|------|----------------|---------|---------|------|----------------|---------|---------|------|----------------|--|---------|------|----------------|--|
| Address    | Instruction   | Register/Value | Register/Value |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:0000    | BB7B08  | mov ax,087B    | ax 0003        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:0003    | 8ED8  | mov ds,ax      | bx 0003        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:0005    | A10000  | mov ax,[0000]  | cx 0000        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:0008    | 8B1E0200  | mov bx,[0002]  | dx 0001        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:000C    | F7F3  | div bx         | si 0000        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:000E    | A30600  | mov [0006],ax  | di 0000        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:0011    | 89160400  | mov [0004],dx  | bp 0000        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:0015    | B44C  | mov ah,4C      | sp 0100        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:0017    | CD21  | int 21         | ds 087B        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:0019    | 0000  | add [bx+si],al | es 0869        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:001B    | 0000  | add [bx+si],al | ss 087C        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:001D    | 0000  | add [bx+si],al | cs 0879        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:001F    | 0000  | add [bp+si],cl | ip 0015        |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:0021    | 0003  | add [bp+di],al |                |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| cs:0023    | 0001  | add [bx+di],al |                |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |
| Conclusion | <p>1) For storing 16-bit data in 8086, we use the data type word DW.</p> <p>2) During multiplication, two 16-bit numbers when multiplied may produce a result larger than 16-bit. This is resolved internally by the microprocessor by storing the higher 16 bits in DX register and lower 16 bits in AX register.</p> <p>3) During division, there are 2 results generated i.e. quotient and remainder. The quotient is stored in AX while the remainder is stored in DX.</p> <p>2) 4C is the service of int 21 that needs to be loaded in AH register for normal termination of program.</p>  |                |                |                |                |         |        |             |         |         |      |           |         |         |        |               |         |         |          |               |         |         |      |        |         |         |        |               |         |         |          |               |         |         |      |           |         |         |      |        |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |         |         |      |                |  |         |      |                |  |

| EXPERIMENT NO. 4 |  |          |  |        |           |          |       |     |    |   |     |       |   |     |    |
|------------------|--|----------|--|--------|-----------|----------|-------|-----|----|---|-----|-------|---|-----|----|
| Title            | Develop Assembly language program using 8086 microprocessor for finding largest/smallest number from block of 16 bit numbers.  |          |  |        |           |          |       |     |    |   |     |       |   |     |    |
| Outcome          | CO2  |          |  |        |           |          |       |     |    |   |     |       |   |     |    |
| Algorithm        | <p>Algorithm:</p> <p><b>Finding the largest number from a block of 16-bit numbers</b></p> <ol style="list-style-type: none"><li>1) Start</li><li>2) Allocate some space for the result, length of array and array in data segment</li><li>3) In code segment load the effective address of the start of array in SI register</li><li>4) Initialize CX with length of the array(number of elements)</li><li>5) Assume the first element as largest and move it to AL register and a memory location 'large'</li><li>6) Compare the contents of memory location small with AL register.</li><li>7) If large &gt; AL, goto step 9</li><li>8) Move contents of AL in memory location 'large'</li><li>9) Increment the effective address in SI register such that it points to the next 16-bit number in array</li><li>10) Decrement CX register by 1</li><li>11) Repeat steps 6-10 until CX=0</li><li>12) Stop</li></ol> <p><b>Finding the smallest number from a block of 16-bit numbers</b></p> <ol style="list-style-type: none"><li>1) Start</li><li>2) Allocate some space for the result, length of array and array in data segment</li><li>3) In code segment load the effective address of the start of array in SI register</li><li>4) Initialize CX with length of the array(number of elements)</li><li>5) Assume the first element as largest and move it to AL register and a memory location 'small'</li><li>6) Compare the contents of memory location small with AL register.</li><li>7) If small &lt; AL, goto step 9</li><li>8) Move contents of AL in memory location 'small'</li><li>9) Increment the effective address in SI register such that it points to the next 16-bit number in array</li><li>10) Decrement CX register by 1</li><li>11) Repeat steps 6-10 until CX=0</li><li>12) Stop</li></ol> |          |  |        |           |          |       |     |    |   |     |       |   |     |    |
| Code             | <p><b>Largest number from a block of 16-bit numbers</b></p> <table><tr><th>Labels</th><th>Mnemonics</th><th>Operands</th></tr><tr><td>Start</td><td>MOV</td><td>AX</td></tr><tr><td>Y</td><td>LEA</td><td>@data</td></tr><tr><td>X</td><td>CMP</td><td>CX</td></tr></table>  |          |  | Labels | Mnemonics | Operands | Start | MOV | AX | Y | LEA | @data | X | CMP | CX |
| Labels           | Mnemonics  | Operands |  |        |           |          |       |     |    |   |     |       |   |     |    |
| Start            | MOV  | AX       |  |        |           |          |       |     |    |   |     |       |   |     |    |
| Y                | LEA  | @data    |  |        |           |          |       |     |    |   |     |       |   |     |    |
| X                | CMP  | CX       |  |        |           |          |       |     |    |   |     |       |   |     |    |

|  |           |      |
|--|-----------|------|
|  | ENDS      | larg |
|  | END       | arr  |
|  | END START | len  |
|  | JNG       | si   |
|  | INC       | DS   |
|  | LOOP      |      |
|  | INT       |      |

model small

.stack 100H

.data

arr db 2h,4h,5h,11h,10h

len dw \$-arr

larg db ?

ends

.code

START: mov ax,@data

mov ds,AX

mov cx,len

lea si,arr

mov al,[si]

mov larg,al

Y: mov al,[si]

cmp al,larg

jng X

mov larg,al

X: inc si

loop Y

mov AH,4CH

int 21H

end start

end

#### Smallest number from a block of 16-bit numbers

| Labels | Mnemonics | Operands |
|--------|-----------|----------|
| Start  | MOV       | AX       |
| Y      | LEA       | @data    |
| X      | CMP       | CX       |
|        | ENDS      | smal     |
|        | END       | arr      |
|        | END START | len      |
|        | JG        | si       |
|        | INC       | DS       |
|        | LOOP      |          |
|        | INT       |          |

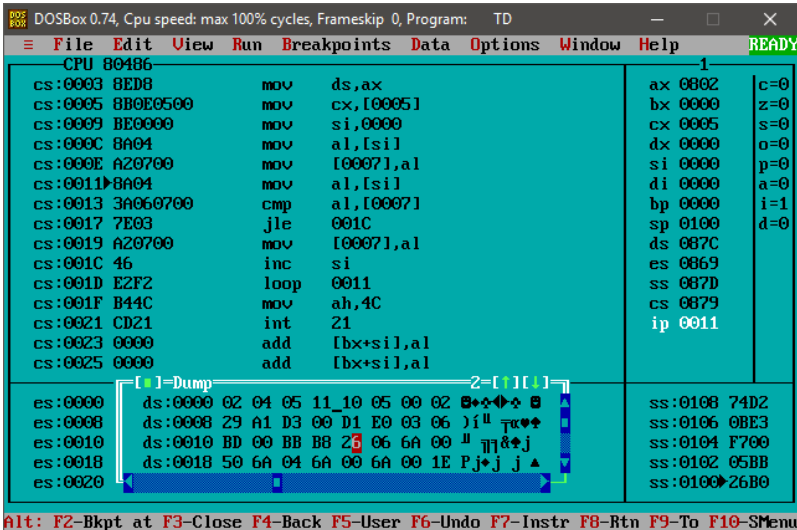
model small

.stack 100H

.data

arr db 2h,4h,5h,11h,10h

len dw \$-arr

|          | <pre>        smal db ? ends  .code  START:   mov ax,@data           mov ds,AX           mov cx,len           lea si,arr           mov al,[si]           mov smal,al  Y:        mov al,[si]           cmp al,smal           jg X           mov smal,al  X:        inc si           loop Y           mov AH,4CH           int 21H          end    start end</pre>  |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
|----------|--|---------------|-----|-------|---------|-------------------------|------------|---------|-------------------------|---------------|---------|-------------------------|-------------|---------|-------------------------|-------------|---------|--|--|----------|-------|----|------|----|------|----|------|----|------|----|------|----|------|----|------|----|------|----|------|----|------|----|------|----|------|----|------|
| Output   | <p><b>Largest number from a block of 16-bit numbers</b></p>  <p>The screenshot shows the DOSBox 0.74 interface. The CPU is 80486. The assembly code is as follows:</p> <pre>cs:0003 8ED8      mov     ds,ax cs:0005 8B0E0500   mov     cx,[0005] cs:0009 BE0000   mov     si,0000 cs:000C 8A04      mov     al,[si] cs:000E A20700   mov     [0007],al cs:0011 8A04      mov     al,[si] cs:0013 3A060700  cmp     al,[0007] cs:0017 7E03      jle     001C cs:0019 A20700   mov     [0007],al cs:001C 46        inc     si cs:001D E2F2      loop    0011 cs:001F B44C      mov     ah,4C cs:0021 CD21      int     21 cs:0023 0000      add     [bx+si],al cs:0025 0000      add     [bx+si],al</pre> <p>The memory dump shows the following data:</p> <table><tr><th>Address</th><th>Hex</th><th>ASCII</th></tr><tr><td>ds:0000</td><td>02 04 05 11 10 05 00 02</td><td>B+ + + + +</td></tr><tr><td>ds:0008</td><td>29 A1 D3 00 D1 E0 03 06</td><td>&gt; f u T a + +</td></tr><tr><td>ds:0010</td><td>BD 00 BB B8 26 06 6A 00</td><td>" j j &amp; + j</td></tr><tr><td>ds:0018</td><td>50 6A 04 6A 00 6A 00 1E</td><td>P j + j j +</td></tr><tr><td>ds:0020</td><td></td><td></td></tr></table> <p>The registers show the following values:</p> <table><tr><th>Register</th><th>Value</th></tr><tr><td>ax</td><td>0002</td></tr><tr><td>bx</td><td>0000</td></tr><tr><td>cx</td><td>0005</td></tr><tr><td>dx</td><td>0000</td></tr><tr><td>si</td><td>0000</td></tr><tr><td>di</td><td>0000</td></tr><tr><td>bp</td><td>0000</td></tr><tr><td>sp</td><td>0100</td></tr><tr><td>ds</td><td>007C</td></tr><tr><td>es</td><td>0069</td></tr><tr><td>ss</td><td>007D</td></tr><tr><td>cs</td><td>0079</td></tr><tr><td>ip</td><td>0011</td></tr></table> <p>Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu</p> | Address       | Hex | ASCII | ds:0000 | 02 04 05 11 10 05 00 02 | B+ + + + + | ds:0008 | 29 A1 D3 00 D1 E0 03 06 | > f u T a + + | ds:0010 | BD 00 BB B8 26 06 6A 00 | " j j & + j | ds:0018 | 50 6A 04 6A 00 6A 00 1E | P j + j j + | ds:0020 |  |  | Register | Value | ax | 0002 | bx | 0000 | cx | 0005 | dx | 0000 | si | 0000 | di | 0000 | bp | 0000 | sp | 0100 | ds | 007C | es | 0069 | ss | 007D | cs | 0079 | ip | 0011 |
| Address  | Hex  | ASCII         |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| ds:0000  | 02 04 05 11 10 05 00 02  | B+ + + + +    |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| ds:0008  | 29 A1 D3 00 D1 E0 03 06  | > f u T a + + |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| ds:0010  | BD 00 BB B8 26 06 6A 00  | " j j & + j   |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| ds:0018  | 50 6A 04 6A 00 6A 00 1E  | P j + j j +   |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| ds:0020  |  |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| Register | Value  |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| ax       | 0002   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| bx       | 0000   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| cx       | 0005   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| dx       | 0000   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| si       | 0000   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| di       | 0000   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| bp       | 0000   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| sp       | 0100   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| ds       | 007C   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| es       | 0069   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| ss       | 007D   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| cs       | 0079   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |
| ip       | 0011   |               |     |       |         |                         |            |         |                         |               |         |                         |             |         |                         |             |         |  |  |          |       |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |    |      |

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

CPU 80486

|                  |      |            |         |     |
|------------------|------|------------|---------|-----|
| cs:0003 8ED8     | mov  | ds,ax      | ax 0810 | c=1 |
| cs:0005 8B0E0500 | mov  | cx,[0005]  | bx 0000 | z=0 |
| cs:0009 BE0000   | mov  | si,0000    | cx 0000 | s=0 |
| cs:000C 8A04     | mov  | al,[si]    | dx 0000 | o=0 |
| cs:000E A20700   | mov  | [0007],al  | si 0005 | p=1 |
| cs:0011 8A04     | mov  | al,[si]    | di 0000 | a=0 |
| cs:0013 3A060700 | cmp  | al,[0007]  | bp 0000 | i=1 |
| cs:0017 7E03     | jle  | 001C       | sp 0100 | d=0 |
| cs:0019 A20700   | mov  | [0007],al  | ds 087C |     |
| cs:001C 46       | inc  | si         | es 0869 |     |
| cs:001D E2F2     | loop | 0011       | ss 087D |     |
| cs:001F B44C     | mov  | ah,4C      | cs 0879 |     |
| cs:0021 CD21     | int  | 21         | ip 001F |     |
| cs:0023 0000     | add  | [bx+si],al |         |     |
| cs:0025 0000     | add  | [bx+si],al |         |     |

[J]=Dump 2-[1][1]

|         |                                 |              |
|---------|---------------------------------|--------------|
| es:0000 | ds:0000 02 04 05 11 10 05 00 11 | ss:0108 74D2 |
| es:0008 | ds:0008 29 A1 D3 00 D1 E0 03 06 | ss:0106 0BE3 |
| es:0010 | ds:0010 BD 00 BB B8 26 06 6A 00 | ss:0104 F700 |
| es:0018 | ds:0018 50 6A 04 6A 00 6A 00 1E | ss:0102 05BB |
| es:0020 |                                 | ss:0100 26B0 |

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

## Smallest number from a block of 16-bit numbers

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

CPU 80486

|                  |      |            |         |     |
|------------------|------|------------|---------|-----|
| cs:0000 B87C08   | mov  | ax,087C    | ax 0802 | c=0 |
| cs:0003 8ED8     | mov  | ds,ax      | bx 0000 | z=0 |
| cs:0005 8B0E0500 | mov  | cx,[0005]  | cx 0005 | s=0 |
| cs:0009 BE0000   | mov  | si,0000    | dx 0000 | o=0 |
| cs:000C 8A04     | mov  | al,[si]    | si 0000 | p=0 |
| cs:000E A20700   | mov  | [0007],al  | di 0000 | a=0 |
| cs:0011 8A04     | mov  | al,[si]    | bp 0000 | i=1 |
| cs:0013 3A060700 | cmp  | al,[0007]  | sp 0100 | d=0 |
| cs:0017 7F03     | jg   | 001C       | ds 087C |     |
| cs:0019 A20700   | mov  | [0007],al  | es 0869 |     |
| cs:001C 46       | inc  | si         | ss 087D |     |
| cs:001D E2F2     | loop | 0011       | cs 0879 |     |
| cs:001F B44C     | mov  | ah,4C      | ip 0011 |     |
| cs:0021 CD21     | int  | 21         |         |     |
| cs:0023 0000     | add  | [bx+si],al |         |     |

[J]=Dump 2-[1][1]

|         |                                 |              |
|---------|---------------------------------|--------------|
| es:0000 | ds:0000 02 04 05 11_10 05 00 02 | ss:0108 74D2 |
| es:0008 | ds:0008 29 A1 D3 00 D1 E0 03 06 | ss:0106 0BE3 |
| es:0010 | ds:0010 BD 00 BB B8 26 06 6A 00 | ss:0104 F700 |
| es:0018 | ds:0018 50 6A 04 6A 00 6A 00 1E | ss:0102 05BB |
| es:0020 |                                 | ss:0100 26B0 |

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

CPU 80486

|                  |      |            |         |     |
|------------------|------|------------|---------|-----|
| cs:0000 B87C08   | mov  | ax,087C    | ax 0810 | c=0 |
| cs:0003 8ED8     | mov  | ds,ax      | bx 0000 | z=0 |
| cs:0005 8B0E0500 | mov  | cx,[0005]  | cx 0000 | s=0 |
| cs:0009 BE0000   | mov  | si,0000    | dx 0000 | o=0 |
| cs:000C 8A04     | mov  | al,[si]    | si 0005 | p=1 |
| cs:000E A20700   | mov  | [0007],al  | di 0000 | a=0 |
| cs:0011 8A04     | mov  | al,[si]    | bp 0000 | i=1 |
| cs:0013 3A060700 | cmp  | al,[0007]  | sp 0100 | d=0 |
| cs:0017 7F03     | jg   | 001C       | ds 087C |     |
| cs:0019 A20700   | mov  | [0007],al  | es 0869 |     |
| cs:001C 46       | inc  | si         | ss 087D |     |
| cs:001D E2F2     | loop | 0011       | cs 0879 |     |
| cs:001F B44C     | mov  | ah,4C      | ip 001F |     |
| cs:0021 CD21     | int  | 21         |         |     |
| cs:0023 0000     | add  | [bx+si],al |         |     |

[J]=Dump 2-[1][1]

|         |                                 |              |
|---------|---------------------------------|--------------|
| es:0000 | ds:0000 02 04 05 11 10 05 00 02 | ss:0108 74D2 |
| es:0008 | ds:0008 29 A1 D3 00 D1 E0 03 06 | ss:0106 0BE3 |
| es:0010 | ds:0010 BD 00 BB B8 26 06 6A 00 | ss:0104 F700 |
| es:0018 | ds:0018 50 6A 04 6A 00 6A 00 1E | ss:0102 05BB |
| es:0020 |                                 | ss:0100 26B0 |

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

## Conclusion

- 1) For storing 16-bit data in 8086, we use the data type word DW.
- 2) LEA instruction is used to retrieve the base address of the array.



|  |   |
|--|---|
|  | <p>3) CMP instruction performs comparison of 2 numbers by performing subtraction and interpreting its result. The result is not stored anywhere, only the flag register is affected accordingly.</p> <p>4) LOOP instruction is used to simulate a loop. It does 2 operations: Decrement CX and check if CX=0, if not it Loops to a given label.</p> <p>5) Jump instruction can be used to create a 'if' condition</p> <p>6) 4C is the service of int 21 that needs to be loaded in AH register for normal termination of program.</p> |
|--|---|

| EXPERIMENT NO. 5 |   |
|------------------|---|
| <b>Title</b>     | Develop a 8086 Assembly Language Program for Conversions– (a) Packed to Unpacked BCD (b) Unpacked to Packed BCD (c) Packed BCD to ASCII   |
| <b>Name</b>      | <b>Sanika Chavan</b>  |
| <b>Roll No</b>   | <b>19101A0053</b>   |
| <b>Outcome</b>   | <b>CO2</b>  |
| <b>Algorithm</b> |   |
| <b>Code</b>      | <pre> (a)Packed to Unpacked BCD  .model small .stack 100H .data cinf     a db 35h ; store 1<sup>st</sup> no at location a     ab1 ?    ; define ub1 location location to save unpacked version of a     ab2 ?    ; define ub1 location location to save unpacked version of a ends  .code start: mov ax,@data       mov ds,ax       mov ax,000h ;clear content of accumulator       mov bl, 00fh ;move 0fh into reg bl       mov bh, 0f0h ;move f0h in reg bh       mov al, a    ; move the first ASCII in al       AND al,bl    ; AND the content of al and bl       mov ub1,al   ; move the result of ANDing at location ub1       mov al,a     ;move the first ASCII no in al       AND al,bh    ; AND the content of al and bh       mov cl,04h   ; move 04h in reg cl       ror al,cl    ; shift the content of ch reg to left by 4 bits       mov ub2,al   ; move the result at location ub1       mov ah, 4ch       int 21h       end start end  (b)Unpacked to Packed BCD .model small .stack 100H .data     a db 05h ; store 1<sup>st</sup> no at location a     b db 03h ; store 2<sup>nd</sup> no at location b     pb1 db ? ; define pb1 locatio to save packed version of a and b  ends  .code start: mov ax,@data       mov ds,ax       mov ax,000h ;clear content of accumulator       mov al,b    ; move the second unpacked in al       mov cl,04h  ; move 04h in reg cl       ror al,cl   ; shift the content of ch reg to left by 4 bits       mov ah,a    ; move the first number into register ah       OR al,ah    ; or the content of al with ah       mov pb1,al       mov ah, 4ch </pre> |

|            |  |
|------------|--|
|            | <div>int 21h<br/>end start<br/>end</div>   |
| Output     | <div><div><div><div><div>CPU 80486</div><div>cs:0000 B8AE48      mov    ax,48AE      ax 4C35      c=0</div><div>cs:0003 8ED8      mov    ds,ax      bx 0000      z=0</div><div>cs:0005 B80000      mov    ax,0000      cx 0004      s=0</div><div>cs:0008 A00D00      mov    al,[000D]      dx 0000      o=0</div><div>cs:000B B104      mov    cl,04      si 0000      p=1</div><div>cs:000D D2C8      ror    al,cl      di 0000      a=0</div><div>cs:000F 8A260C00      mov    ah,[000C]      bp 0000      i=1</div><div>cs:0013 0AC4      or     al,ah      sp 0100      d=0</div><div>cs:0015 A20E00      [ ]=Dump      2-[↑↓↑↓]</div><div>cs:0018 B44C      ds:0000 26 0C 00 0A C4 A2 0E 00 8F 0-6f</div><div>cs:001A CD21      ds:0008 B4 4C CD 21 05 03 35 00 L=154</div><div>cs:001C 050335      ds:0010 00 00 00 00 00 00 00 00</div><div>cs:001F 0000      ds:0018 00 00 00 00 00 00 00 00</div><div>es:0000 CD 20 FF 9F 00 EA FF FF = f 9</div><div>es:0008 AD DE E0 01 C5 15 AA 01 i x S</div><div>es:0010 C5 15 89 02 20 10 92 01 L Se A</div><div>es:0018 01 03 01 00 02 FF FF FF B B</div><div>ss:0102 0000</div><div>ss:0100 0000</div></div><div>F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu</div><div><div><div><div>CPU 80486</div><div>cs:001D A20700      mov    [0007],al      ax 4C03      c=0</div><div>cs:0020 B44C      mov    ah,4C      bx F00F      z=0</div><div>cs:0022 CD21      int    21      cx 0004      s=0</div><div>cs:0024 353405      xor    ax,0534      dx 0000      o=0</div><div>cs:0027 0300      add    ax,[bx+sil]      si 0000      p=1</div><div>cs:0029 0000      add    [bx+sil],al      di 0000      a=0</div><div>cs:002B 0000      add    [bx+sil],al      bp 0000      i=1</div><div>cs:002D 0000      add    [bx+sil],al      sp 0100      d=0</div><div>cs:002F 0000      [ ]=Dump      2-[↑↓↑↓]</div><div>cs:0031 0000      ds:0000 B4 4C CD 21 35 34 05 03 L=154</div><div>cs:0033 0000      ds:0008 00 00 00 00 00 00 00 00</div><div>cs:0035 0000      ds:0010 00 00 00 00 00 00 00 00</div><div>cs:0037 0000      ds:0018 00 00 00 00 00 00 00 00</div><div>es:0000 CD 20 FF 9F 00 EA FF FF = f 9</div><div>es:0008 AD DE E0 01 C5 15 AA 01 i x S</div><div>es:0010 C5 15 89 02 20 10 92 01 L Se A</div><div>es:0018 01 03 01 00 02 FF FF FF B B</div><div>ss:0102 0000</div><div>ss:0100 0000</div></div><div>F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu</div></div></div></div></div></div> |
| Conclusion | <div><p>The ASCII number can be unpacked or converted into HEX form by masking its upper nibble. This masking can be done by two methods. One method is we can AND the ASCII number by 0FH or we can perform SAL/SHL operation 4 times. This SHL instruction shifts the content of the source to left and appends 0s in LSB.</p><p>The unpacked number can be packed by using OR operation.</p><p>For e.g., 01H and 02H are two numbers to be packed, we can shift one of the number 4 times towards left by using SAL/SHL instruction and OR shifted number with the another number i.e. if 01H is shifted left four times, it becomes 10H. Then OR it with 02H. The result will be 12H which is packed version of those two numbers.</p></div>   |

| EXPERIMENT NO. 6 |   |
|------------------|---|
| <b>Title</b>     | Develop Assembly language program using 8086 microprocessor for block copy, block exchange with and without string instructions.  |
| <b>Outcome</b>   | <b>CO2</b>  |
| <b>Algorithm</b> | <p><b>Copying Byte array from one location to another without using string instructions</b></p> <ol style="list-style-type: none"> <li>1) Start.</li> <li>2) Allocate some space for the input array, output array and length of array in data segment.</li> <li>3) Initialize CX with the length of the array.</li> <li>4) Load the effective address of input string in SI and that of output string in DI.</li> <li>5) Move the contents from the location whose offset is currently pointed by SI in Data segment to the location whose offset is currently pointed by DI in Data segment.</li> <li>6) Increment SI,DI.</li> <li>7) Decrement CX.</li> <li>8) Repeat steps 5-7 until CX = 0.</li> <li>9) Stop.</li> </ol> <p><b>Copying Byte array from one location to another using string instructions</b></p> <ol style="list-style-type: none"> <li>1) Start.</li> <li>2) Allocate some space for the input array, output array and length of array in data segment.</li> <li>3) Initialize CX with the length of the array.</li> <li>4) Load the effective address of input string in SI and that of output string in DI.</li> <li>5) Overlap the Data Segment and Extra Segment in memory.</li> <li>6) Using the REPNZ prefix for MOVSB instruction, the array is copied into destination location.</li> <li>7) Stop.</li> </ol> <p><b>Copying Word array from one location to another without using string instructions</b></p> <ol style="list-style-type: none"> <li>1) Start.</li> <li>2) Allocate some space for the input array, output array and length of array in data segment.</li> <li>3) Initialize CX with the length of the array.</li> <li>4) Load the effective address of input string in SI and that of output string in DI.</li> <li>5) Move the contents from the location whose offset is currently pointed by SI in Data segment to the location whose offset is currently pointed by DI in Data segment.</li> <li>6) Increment SI and DI by 2.</li> <li>7) Decrement CX by 2.</li> <li>8) Repeat steps 5 &amp; 6 until CX = 0.</li> <li>9) Stop.</li> </ol> <p><b>Copying Word array from one location to another using string instructions</b></p> |

|             |   |
|-------------|---|
|             | 1) Start.<br>2) Allocate some space for the input array, output array and length of array in data segment.<br>3) Initialize CX with the length of the array.<br>4) Load the effective address of input string in SI and that of output string in DI.<br>5) Overlap the Data Segment and Extra Segment in memory.<br>6) Using the REPZ prefix for MOVSB instruction, the array is copied into destination location.<br>7) Stop.  |
| <b>Code</b> | 1. Copying Byte array from one location to another without using string instructions<br>Labels Mnemonics Operands<br>START MOV AX,@DATA<br>REPEAT LEA DS<br>INC CX<br>LOOP LEN<br>INT ARR1<br>ARR2<br>SI<br>DI<br>21H,4CH<br>BL<br><br>.model small<br>.stack 100h<br>.data<br>arr1 db 02h,09h,06h,10h,07h<br>arr2 db ?<br>len dw \$-arr1<br>ends<br>.code<br>start: mov ax,@data<br>mov ds,ax<br>mov cx,len<br>lea si,arr1<br>lea di,arr2<br>repeat: mov bl,[si]<br>mov ds:[di],bl<br>inc si<br>inc di<br>loop repeat<br>mov ah,4ch<br>int 21h<br>ends<br>end start<br>2. Copying Byte array from one location to another using string instructions<br>Labels Mnemonics Operands<br>START MOV AX,@DATA<br>LEA DS |

```

MOVSB CX
REPNZ LEN
INT     ARR1
        ARR2
        SI
        DI
        21H,4CH
        ES

```

```

.model small
.stack 100h
.data
    arr1 db 02h,09h,06h,10h,07h
    arr2 db ?
    len dw $-arr1
ends

.code
start:  mov ax,@data
        mov ds,ax
        mov es,ax
        mov cx,len
        lea si,arr1
        lea di,arr2
        repnz movsb
        mov ah,4ch
        int 21h

    ends
end start

```

3. Copying Word array from one location to another without using string instructions

| Labels | Mnemonics | Operands   |
|--------|-----------|------------|
| START  | MOV       | AX,@DATA   |
| REPEAT | LEA       | DS         |
|        | ADD       | CX         |
|        | SUB       | LEN        |
|        | JNZ       | ARR1       |
|        | INT       | ARR2       |
|        |           | SI         |
|        |           | DI         |
|        |           | 21H,4CH,2H |
|        |           | BX         |

```

.model small
.stack 100h
.data
    arr1 dw 0212h,0933h,0621h,1023h,0798h
    arr2 dw ?
    len dw $-arr1
ends

```

|        |   |
|--------|---|
|        | <pre> .code start:  mov ax,@data         mov ds,ax         mov cx,len         lea si,arr1         lea di,arr2 repeat: mov bx,[si]         mov ds:[di],bx         add si,2h         add di,2h         sub cx,2h         jnz repeat         mov ah,4ch         int 21h          ends end start  4.      Copying Word array from one location to another using string instructions Labels Mnemonics   Operands START MOV    AX,@DATA       LEA     DS       MOVSW   CX       REPNZ  LEN       INT     ARR1            ARR2            SI            DI            21H,4CH            ES  .model small .stack 100h .data arr1 dw 0212h,0933h,0621h,1023h,0798h arr2 dw ? len dw \$-arr1 ends  .code start:  mov ax,@data         mov ds,ax         mov es,ax         mov cx,len         lea si,arr1         lea di,arr2         repnz movsw         mov ah,4ch         int 21h          ends end start </pre> |
| Output | Copying Byte array from one location to another without using string  |

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help READY

CPU 80486

|                  |      |             |         |     |
|------------------|------|-------------|---------|-----|
| cs:0000 B87B08   | mov  | ax,087B     | ax 087B | c=0 |
| cs:0003 8ED8     | mov  | ds,ax       | bx 0000 | z=0 |
| cs:0005 8B0E0600 | mov  | cx,[0006]   | cx 0000 | s=0 |
| cs:0009 BE0000   | mov  | si,0000     | dx 0000 | o=0 |
| cs:000C BF0500   | mov  | di,0005     | si 0000 | p=0 |
| cs:000F 8A1C     | mov  | bl,[sil     | di 0000 | a=0 |
| cs:0011 881D     | mov  | ldil,bl     | bp 0000 | i=1 |
| cs:0013 46       | inc  | si          | sp 0100 | d=0 |
| cs:0014 47       | inc  | di          | ds 087B |     |
| cs:0015 E2F8     | loop | 000F        | es 0869 |     |
| cs:0017 B44C     | mov  | ah,4C       | ss 087C |     |
| cs:0019 CD21     | int  | 21          | cs 0879 |     |
| cs:001B 0000     | add  | [bx+sil],al | ip 0005 |     |
| cs:001D 0000     | add  | [bx+sil],al |         |     |
| cs:001F 0002     | add  | [bp+sil],al |         |     |

[ ]=Dump 2=[ ]

|               |                                 |              |
|---------------|---------------------------------|--------------|
| es:0000 CD 20 | ds:0000 02 09 06 10 07 00 06 00 | ss:0108 B406 |
| es:0008 AD DE | ds:0008 FA 00 00 C7 46 F4 00 00 | ss:0106 032E |
| es:0010 14 03 | ds:0010 EB E8 83 3E BD 00 02 76 | ss:0104 EB02 |
| es:0018 01 01 | ds:0018 29 A1 D3 00 D1 E0 03 06 | ss:0102 7400 |
| es:0020 FF FF |                                 | ss:0100 E706 |

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help READY

CPU 80486

|                  |      |             |         |     |
|------------------|------|-------------|---------|-----|
| cs:0000 B87B08   | mov  | ax,087B     | ax 087B | c=0 |
| cs:0003 8ED8     | mov  | ds,ax       | bx 0002 | z=0 |
| cs:0005 8B0E0600 | mov  | cx,[0006]   | cx 0000 | s=0 |
| cs:0009 BE0000   | mov  | si,0000     | dx 0000 | o=0 |
| cs:000C BF0500   | mov  | di,0005     | si 0006 | p=0 |
| cs:000F 8A1C     | mov  | bl,[sil     | di 000B | a=0 |
| cs:0011 881D     | mov  | ldil,bl     | bp 0000 | i=1 |
| cs:0013 46       | inc  | si          | sp 0100 | d=0 |
| cs:0014 47       | inc  | di          | ds 087B |     |
| cs:0015 E2F8     | loop | 000F        | es 0869 |     |
| cs:0017 B44C     | mov  | ah,4C       | ss 087C |     |
| cs:0019 CD21     | int  | 21          | cs 0879 |     |
| cs:001B 0000     | add  | [bx+sil],al | ip 0017 |     |
| cs:001D 0000     | add  | [bx+sil],al |         |     |
| cs:001F 0002     | add  | [bp+sil],al |         |     |

[ ]=Dump 2=[ ]

|               |                                 |              |
|---------------|---------------------------------|--------------|
| es:0000 CD 20 | ds:0000 02 09 06 10 07_02 09 06 | ss:0108 B406 |
| es:0008 AD DE | ds:0008 10 07 02 C7 46 F4 00 00 | ss:0106 032E |
| es:0010 14 03 | ds:0010 EB E8 83 3E BD 00 02 76 | ss:0104 EB02 |
| es:0018 01 01 | ds:0018 29 A1 D3 00 D1 E0 03 06 | ss:0102 7400 |
| es:0020 FF FF |                                 | ss:0100 E706 |

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu

Copying Byte array from one location to another using string



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help READY

CPU 80486

|                  |       |             |         |     |
|------------------|-------|-------------|---------|-----|
| cs:0000 B87B08   | mov   | ax,087B     | ax 087B | c=0 |
| cs:0003 8ED8     | mov   | ds,ax       | bx 0000 | z=0 |
| cs:0005 8EC0     | mov   | es,ax       | cx 0000 | s=0 |
| cs:0007 8B0E0600 | mov   | cx,[0006]   | dx 0000 | o=0 |
| cs:000B BE0000   | mov   | si,0000     | si 0000 | p=0 |
| cs:000E BF0500   | mov   | di,0005     | di 0000 | a=0 |
| cs:0011 F2A4     | repnz | movsb       | bp 0000 | i=1 |
| cs:0013 B44C     | mov   | ah,4C       | sp 0100 | d=0 |
| cs:0015 CD21     | int   | 21          | ds 087B |     |
| cs:0017 0000     | add   | [bx+sil],al | es 087B |     |
| cs:0019 0000     | add   | [bx+sil],al | ss 087C |     |
| cs:001B 0000     | add   | [bx+sil],al | cs 0879 |     |
| cs:001D 0000     | add   | [bx+sil],al | ip 0007 |     |
| cs:001F 0002     | add   | [bp+sil],al |         |     |
| cs:0021 09061007 | or    | [0710],ax   |         |     |

[ ]=Dump 2=[ ]

|                 |                                 |              |
|-----------------|---------------------------------|--------------|
| 0869:0000 CD 20 | ds:0000 02 09 06 10 07 00 06 00 | ss:0108 B406 |
| 0869:0008 AD DE | ds:0008 FA 00 00 C7 46 F4 00 00 | ss:0106 032E |
| 0869:0010 14 03 | ds:0010 EB E8 83 3E BD 00 02 76 | ss:0104 EB02 |
| 0869:0018 01 01 | ds:0018 29 A1 D3 00 D1 E0 03 06 | ss:0102 7400 |
| 0869:0020 FF FF |                                 | ss:0100 E706 |

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help READY

CPU 80486

|                  |       |             |         |     |
|------------------|-------|-------------|---------|-----|
| cs:0000 B87B08   | mov   | ax,087B     | ax 4C7B | c=0 |
| cs:0003 8ED8     | mov   | ds,ax       | bx 0000 | z=0 |
| cs:0005 8EC0     | mov   | es,ax       | cx 0000 | s=0 |
| cs:0007 8B0E0600 | mov   | cx,[0006]   | dx 0000 | o=0 |
| cs:000B BE0000   | mov   | si,0000     | si 0006 | p=0 |
| cs:000E BF0500   | mov   | di,0005     | di 000B | a=0 |
| cs:0011 F2A4     | repnz | movsb       | bp 0000 | i=1 |
| cs:0013 B44C     | mov   | ah,4C       | sp 0100 | d=0 |
| cs:0015 CD21     | int   | 21          | ds 087B |     |
| cs:0017 0000     | add   | [bx+sil],al | es 087B |     |
| cs:0019 0000     | add   | [bx+sil],al | ss 087C |     |
| cs:001B 0000     | add   | [bx+sil],al | cs 0879 |     |
| cs:001D 0000     | add   | [bx+sil],al | ip 0015 |     |
| cs:001F 0002     | add   | [bp+sil],al |         |     |
| cs:0021 09061007 | or    | [0710],ax   |         |     |

[ ]=Dump 2=[ ]

|                 |                                 |              |
|-----------------|---------------------------------|--------------|
| 0869:0000 CD 20 | ds:0000 02 09 06 10 07 02 09 06 | ss:0108 B406 |
| 0869:0008 AD DE | ds:0008 10 07 02 C7 46 F4 00 00 | ss:0106 032E |
| 0869:0010 14 03 | ds:0010 EB E8 83 3E BD 00 02 76 | ss:0104 EB02 |
| 0869:0018 01 01 | ds:0018 29 A1 D3 00 D1 E0 03 06 | ss:0102 7400 |
| 0869:0020 FF FF |                                 | ss:0100 E706 |

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

Copying Word array from one location to another without using string

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help READY

CPU 80486

|                  |     |            |         |     |
|------------------|-----|------------|---------|-----|
| cs:0000 B87C08   | mov | ax,087C    | ax 087C | c=0 |
| cs:0003 8ED8     | mov | ds,ax      | bx 0000 | z=0 |
| cs:0005 8B0E0C00 | mov | cx,[000C]  | cx 0000 | s=0 |
| cs:0009 BE0000   | mov | si,0000    | dx 0000 | o=0 |
| cs:000C BF0A00   | mov | di,000A    | si 0000 | p=0 |
| cs:000F 8B1C     | mov | bx,[sil    | di 0000 | a=0 |
| cs:0011 891D     | mov | ldil,bx    | bp 0000 | i=1 |
| cs:0013 83C602   | add | si,0002    | sp 0100 | d=0 |
| cs:0016 83C702   | add | di,0002    | ds 087C |     |
| cs:0019 83E902   | sub | cx,0002    | es 0869 |     |
| cs:001C 75F1     | jne | 000F       | ss 087D |     |
| cs:001E B44C     | mov | ah,4C      | cs 0879 |     |
| cs:0020 CD21     | int | 21         | ip 0005 |     |
| cs:0022 0000     | add | [bx+sil,al |         |     |
| cs:0024 0000     | add | [bx+sil,al |         |     |

[ ]=Dump 2=[ ]

|               |                                 |              |
|---------------|---------------------------------|--------------|
| es:0000 CD 20 | ds:0000 12 02 33 09 21 06 23 10 | ss:0108 74D2 |
| es:0008 AD DE | ds:0008 98 07 00 00 0C 00 03 06 | ss:0106 0BE3 |
| es:0010 14 03 | ds:0010 BD 00 BB B8 26 06 6A 00 | ss:0104 F700 |
| es:0018 01 01 | ds:0018 50 6A 04 6A 00 6A 00 1E | ss:0102 05BB |
| es:0020 FF FF |                                 | ss:0100 26B0 |

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help READY

CPU 80486

|                  |     |            |         |     |
|------------------|-----|------------|---------|-----|
| cs:0000 B87C08   | mov | ax,087C    | ax 4C7C | c=0 |
| cs:0003 8ED8     | mov | ds,ax      | bx 0212 | z=1 |
| cs:0005 8B0E0C00 | mov | cx,[000C]  | cx 0000 | s=0 |
| cs:0009 BE0000   | mov | si,0000    | dx 0000 | o=0 |
| cs:000C BF0A00   | mov | di,000A    | si 000C | p=1 |
| cs:000F 8B1C     | mov | bx,[sil    | di 0016 | a=0 |
| cs:0011 891D     | mov | ldil,bx    | bp 0000 | i=1 |
| cs:0013 83C602   | add | si,0002    | sp 0100 | d=0 |
| cs:0016 83C702   | add | di,0002    | ds 087C |     |
| cs:0019 83E902   | sub | cx,0002    | es 0869 |     |
| cs:001C 75F1     | jne | 000F       | ss 087D |     |
| cs:001E B44C     | mov | ah,4C      | cs 0879 |     |
| cs:0020 CD21     | int | 21         | ip 0020 |     |
| cs:0022 0000     | add | [bx+sil,al |         |     |
| cs:0024 0000     | add | [bx+sil,al |         |     |

[ ]=Dump 2=[ ]

|               |                                 |              |
|---------------|---------------------------------|--------------|
| es:0000 CD 20 | ds:0000 12 02 33 09 21 06 23 10 | ss:0108 74D2 |
| es:0008 AD DE | ds:0008 98 07 12 02 33 09 21 06 | ss:0106 0BE3 |
| es:0010 14 03 | ds:0010 23 10 98 07 12 02 6A 00 | ss:0104 F700 |
| es:0018 01 01 | ds:0018 50 6A 04 6A 00 6A 00 1E | ss:0102 05BB |
| es:0020 FF FF |                                 | ss:0100 26B0 |

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

Copying Word array from one location to another using string

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

CPU 80486

|                  |                |         |     |
|------------------|----------------|---------|-----|
| cs:0000 B87B08   | mov ax,087B    | ax 087B | c=0 |
| cs:0003 8ED8     | mov ds,ax      | bx 0000 | z=0 |
| cs:0005 8EC0     | mov es,ax      | cx 0000 | s=0 |
| cs:0007 8B0E0C00 | mov cx,[000C]  | dx 0000 | o=0 |
| cs:000B BE0000   | mov si,0000    | si 0000 | p=0 |
| cs:000E BF0A00   | mov di,000A    | di 0000 | a=0 |
| cs:0011 F2A5     | repnz movsw    | bp 0000 | i=1 |
| cs:0013 B44C     | mov ah,4C      | sp 0100 | d=0 |
| cs:0015 CD21     | int 21         | ds 087B |     |
| cs:0017 0000     | add [bx+si],al | es 0869 |     |
| cs:0019 0000     | add [bx+si],al | ss 087C |     |
| cs:001B 0000     | add [bx+si],al | cs 0879 |     |
| cs:001D 0000     | add [bx+si],al | ip 0005 |     |
| cs:001F 0012     | add [bp+si],dl |         |     |
| cs:0021 0233     | add dh,[bp+di] |         |     |

[ ]=Dump 2=[ ]

|               |                                 |              |
|---------------|---------------------------------|--------------|
| es:0000 CD 20 | ds:0000 12 02 33 09 21 06 23 10 | ss:0108 B406 |
| es:0008 AD DE | ds:0008 98 07 00 00 0C 00 00 00 | ss:0106 032E |
| es:0010 14 03 | ds:0010 EB E8 83 3E BD 00 02 76 | ss:0104 EB02 |
| es:0018 01 01 | ds:0018 29 A1 D3 00 D1 E0 03 06 | ss:0102 7400 |
| es:0020 FF FF |                                 | ss:0100 E706 |

Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

CPU 80486

|                  |                |         |     |
|------------------|----------------|---------|-----|
| cs:0000 B87B08   | mov ax,087B    | ax 4C7B | c=0 |
| cs:0003 8ED8     | mov ds,ax      | bx 0000 | z=0 |
| cs:0005 8EC0     | mov es,ax      | cx 0000 | s=0 |
| cs:0007 8B0E0C00 | mov cx,[000C]  | dx 0000 | o=0 |
| cs:000B BE0000   | mov si,0000    | si 0018 | p=0 |
| cs:000E BF0A00   | mov di,000A    | di 0022 | a=0 |
| cs:0011 F2A5     | repnz movsw    | bp 0000 | i=1 |
| cs:0013 B44C     | mov ah,4C      | sp 0100 | d=0 |
| cs:0015 CD21     | int 21         | ds 087B |     |
| cs:0017 0000     | add [bx+si],al | es 087B |     |
| cs:0019 0000     | add [bx+si],al | ss 087C |     |
| cs:001B 0000     | add [bx+si],al | cs 0879 |     |
| cs:001D 0000     | add [bx+si],al | ip 0015 |     |
| cs:001F 0012     | add [bp+si],dl |         |     |
| cs:0021 0233     | add dh,[bp+di] |         |     |

[ ]=Dump 2=[ ]

|                 |                                 |              |
|-----------------|---------------------------------|--------------|
| 0869:0000 CD 20 | ds:0000 12 02 33 09 21 06 23 10 | ss:0108 B406 |
| 0869:0008 AD DE | ds:0008 98 07 12 02 33 09 21 06 | ss:0106 032E |
| 0869:0010 14 03 | ds:0010 23 10 98 07 12 02 33 09 | ss:0104 EB02 |
| 0869:0018 01 01 | ds:0018 21 06 23 10 98 07 12 02 | ss:0102 7400 |
| 0869:0020 FF FF |                                 | ss:0100 E706 |

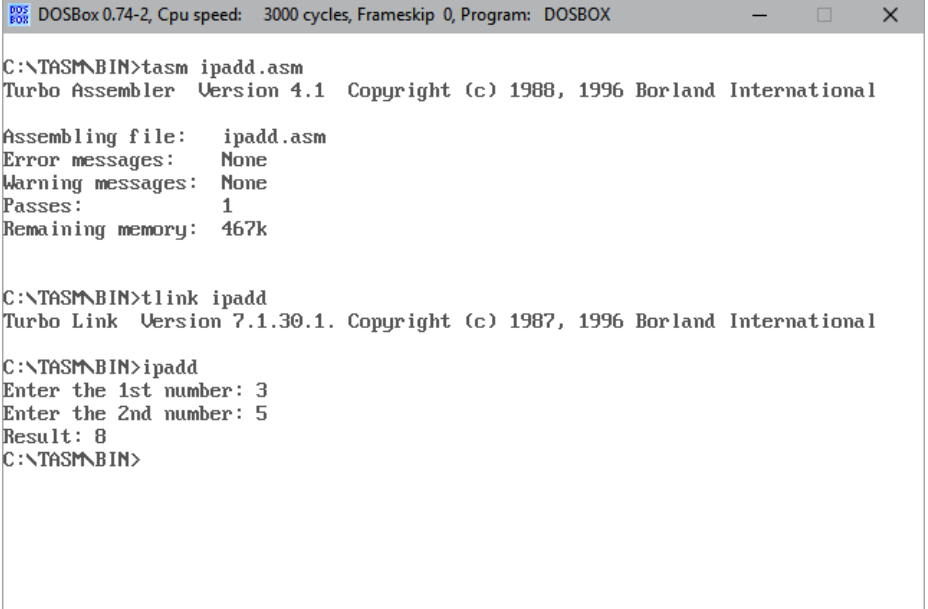
Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-SMenu

## Conclusion

- 1) Segment override prefix is used to change the default segment when physical address is calculated internally by the microprocessor.
- 2) String instructions by default use the Data segment for input and Extra segment for output.
- 3) The instruction prefix REP NZ can only be used with string instructions and makes the processor execute the following string instruction till the whole input string is processed.

| EXPERIMENT NO. 7 |  |                        |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|------------------|--|------------------------|--------|-----------|----------|-------|-----|----------|--|-----|----|--|-----|----|--|-----|----------------|--|-----|----|--|-----|----|--|-----|----|--|--|----|--|--|----|--|--|---------|--|--|------------------------|
| Title            | Develop assembly language program using 8086 microprocessor for I/O using INT N.   |                        |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
| Outcome          | CO2  |                        |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
| Algorithm        | <p>Algorithm</p> <p><b>Perform addition of 2 numbers using I/O operations</b></p> <p>1) Start</p> <p>2) Allocate some space for the messages to be displayed in the data segment.</p> <p>3) Display message to accept 1<sup>st</sup> input.</p> <p>4) Accept the input from user and store it in some register.</p> <p>5) Display message to accept 2<sup>nd</sup> input.</p> <p>6) Accept the 2<sup>nd</sup> input from user and store in in some register.</p> <p>7) Perform addition and store the result in DL register.</p> <p>8) Stop.</p>   |                        |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
| Code             | <p><b>Perform addition of 2 numbers using I/O operations</b></p> <table><tr><th>Labels</th><th>Mnemonics</th><th>Operands</th></tr><tr><td>START</td><td>MOV</td><td>AX,@DATA</td></tr><tr><td></td><td>LEA</td><td>DS</td></tr><tr><td></td><td>INC</td><td>AH</td></tr><tr><td></td><td>INT</td><td>msg1,msg2,msg3</td></tr><tr><td></td><td>SUB</td><td>CL</td></tr><tr><td></td><td>ADD</td><td>BL</td></tr><tr><td></td><td>AAA</td><td>DL</td></tr><tr><td></td><td></td><td>DX</td></tr><tr><td></td><td></td><td>AL</td></tr><tr><td></td><td></td><td>21H,4CH</td></tr><tr><td></td><td></td><td>09H,01H,30H,13H,10,02H</td></tr></table> <p>.model small<br/>.stack 100h<br/>.data<br/>    msg1 db 'Enter the 1st number: \$'<br/>    msg2 db 'Enter the 2nd number: \$'<br/>    msg3 db 'Result: \$'<br/>    ends<br/>.code<br/>    start:        mov ax,@data<br/>                  mov ds,ax<br/>                  mov ah,09h<br/>                  lea dx,msg1<br/>                  int 21h<br/>                  mov ah,01h<br/>                  int 21h<br/>                  sub al,30h<br/>                  mov bl,al<br/><br/>                  mov DL, 10      ;printing new line<br/>                  mov AH, 02h</p> |                        | Labels | Mnemonics | Operands | START | MOV | AX,@DATA |  | LEA | DS |  | INC | AH |  | INT | msg1,msg2,msg3 |  | SUB | CL |  | ADD | BL |  | AAA | DL |  |  | DX |  |  | AL |  |  | 21H,4CH |  |  | 09H,01H,30H,13H,10,02H |
| Labels           | Mnemonics  | Operands               |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
| START            | MOV  | AX,@DATA               |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|                  | LEA  | DS                     |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|                  | INC  | AH                     |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|                  | INT  | msg1,msg2,msg3         |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|                  | SUB  | CL                     |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|                  | ADD  | BL                     |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|                  | AAA  | DL                     |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|                  |  | DX                     |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|                  |  | AL                     |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|                  |  | 21H,4CH                |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |
|                  |  | 09H,01H,30H,13H,10,02H |        |           |          |       |     |          |  |     |    |  |     |    |  |     |                |  |     |    |  |     |    |  |     |    |  |  |    |  |  |    |  |  |         |  |  |                        |

|     |     |  |
|-----|-----|--|
|     |     | <pre> int 21h mov DL, 13 mov AH, 02h int 21h  mov ah,09h lea dx,msg2 int 21h mov ah,01h int 21h sub al,30h mov cl,al  mov DL, 10    ;printing new line mov AH, 02h int 21h mov DL, 13 mov AH, 02h int 21h  mov ah,09h lea dx,msg3 int 21h  add cl,bl mov al,cl aaa add al,30h mov dl,al  mov ah,02h int 21h  mov ah,4ch int 21h start </pre> |
|     | end |  |
| end |     |  |

|                   |   |
|-------------------|---|
| <b>Output</b>     |  <p>The screenshot shows a DOSBox window titled "DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX". The terminal output is as follows:</p> <pre> C:\TASM\BIN&gt;tasm ipadd.asm Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International  Assembling file:   ipadd.asm Error messages:   None Warning messages:  None Passes:           1 Remaining memory: 467k  C:\TASM\BIN&gt;tlink ipadd Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International  C:\TASM\BIN&gt;ipadd Enter the 1st number: 3 Enter the 2nd number: 5 Result: 8 C:\TASM\BIN&gt; </pre> |
| <b>Conclusion</b> | <ol style="list-style-type: none"> <li>1) DX is used to store the offset of the message which is to be displayed on the output screen.</li> <li>2) By default the input taken from the user is stored in AL.</li> <li>3) Combination of various data stored in AH and interrupts allow us to make use of different services like display message, display data in register, display new line etc.</li> </ol>  |

| EXPERIMENT NO. 8 |  |
|------------------|--|
| <b>Title</b>     | 8086 Assembly Language Program to Count Odd and Even Numbers from the given block of Numbers   |
| <b>Outcome</b>   | <b>CO2</b>   |
| <b>code</b>      | <pre> .model small .stack 100H .data     array db 02h,04h,05h,10h,11h oddcnt db ?     evencnt db ? ends .code start: mov ax,@data mov       ds,ax mov cx,05h       lea si,array next:  mov al,[si]       ror al,01h      ;rotaterightarrayelementpresentinalsoits lsb goes into CF       jnceven        ; if CF =0 goto lable even       inc dl          ; else due to CF=1 increment odd count jmp ahead ; goto       nextelement even:  inc dh          ;due to CF=0 increment even count in reg dh mov oddcnt,dl;copy       count in dl reg into variable oddcount       mov evencnt,dh ; copy count in dh reg into variable evencount       ahead: inc si      ;increment array pointer to address next element       loop next ;dec byte counter cx and if cx not=0 goto lable next       mov ah,4ch int       21h end start end </pre> |

## Output

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

CPU 80486

|         |       |     |            |    |      |     |
|---------|-------|-----|------------|----|------|-----|
| cs:0025 | CD 21 | int | 21         | ax | 4C88 | c=1 |
| cs:0027 | 00 02 | add | [bp+si],al | bx | 0000 | z=0 |
| cs:0029 | 04 05 | add | al,05      | cx | 0000 | s=0 |
| cs:002B | 10 11 | adc | [bx+di],dl | dx | 0302 | o=0 |
| cs:002D | 01 03 | add | [bp+di],ax | si | 0000 | p=0 |
| cs:002F | 00 00 | add | [bx+si],al | di | 0000 | a=0 |
| cs:0031 | 00 00 | add | [bx+si],al | bp | 0000 | i=1 |
| cs:0033 | 00 00 | add | [bx+si],al | sp | 0100 | d=0 |
| cs:0035 | 00 00 |     |            |    |      |     |
| cs:0037 | 00 00 |     |            |    |      |     |
| cs:0039 | 00 00 |     |            |    |      |     |
| cs:003B | 00 00 |     |            |    |      |     |
| cs:003D | 00 00 |     |            |    |      |     |

[ ]=Dump

|         |                         |          |
|---------|-------------------------|----------|
| ds:0000 | 46 E2 E8 B4 4C CD 21 00 | FF 3 L=? |
| ds:0008 | 02 04 05 10 11 01 03 00 |          |
| ds:0010 | 00 00 00 00 00 00 00 00 |          |
| ds:0018 | 00 00 00 00 00 00 00 00 |          |

|         |                         |                   |
|---------|-------------------------|-------------------|
| es:0000 | CD 20 FF 9F 00 EA FF FF | = f 0             |
| es:0008 | AD DE E0 01 C5 15 AA 01 | i 0 3 0           |
| es:0010 | C5 15 89 02 20 10 92 01 | + 0 0 0 0 0 0 0 0 |
| es:0018 | 01 03 01 00 02 FF FF FF | 0 0 0 0 0 0 0 0   |

ss:0102 0000  
ss:0100 0000

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

## Conclusion

To find the number of even and odd numbers in given string, we can make use of RCR instruction which rotates all the bits in a specified word or byte some number of bit positions to the right. The operation is circular because the LSB of the operand is rotated into the carry flag and the bit in the carry flag is rotated around into the MSB of the operand.

Thus by checking carry flag we can increment the content of location where number of even numbers is to be saved if carry flag is reset. Then we can also find the number of odd numbers in string by subtracting number of even numbers from the string length.



[illegible]

## Output

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

| CPU 80486 |        | 1   |            |    |      |     |
|-----------|--------|-----|------------|----|------|-----|
| cs:0022   | FECB   | dec | ch         | ax | 4C01 | c=1 |
| cs:0024   | 75E1   | jne | 0007       | bx | 0002 | z=1 |
| cs:0026   | B44C   | mov | ah,4C      | cx | 0000 | s=0 |
| cs:0028   | CD21   | int | 21         | dx | 0002 | o=0 |
| cs:002A   | 0102   | add | [bp+si],ax | si | 000B | p=1 |
| cs:002C   | 0304   | add | ax,[si]    | di | 0000 | a=0 |
| cs:002E   | 050607 | add | ax,0706    | bp | 0000 | i=1 |
| cs:0031   | 0808   | or  | [bx+si],cl | sp | 0100 | d=0 |
| cs:0033   | 0000   |     |            |    |      |     |
| cs:0035   | 0000   |     |            |    |      |     |
| cs:0037   | 0000   |     |            |    |      |     |
| cs:0039   | 0000   |     |            |    |      |     |
| cs:003B   | 0000   |     |            |    |      |     |

[J]=Dump 2=[I][I]

| ds:0000 |                         | 75 EA FE CD 75 E1 B4 4C |  | u8=L |  |
|---------|-------------------------|-------------------------|--|------|--|
| ds:0008 | CD 21 01 02 03 04 05 06 | =f8000000               |  |      |  |
| ds:0010 | 07 08 09 0A 0B 0C 0D 0E | =f8000000               |  |      |  |
| ds:0018 | 0F 10 11 12 13 14 15 16 | =f8000000               |  |      |  |

| es:0000 |                         | CD 20 FF 9F 00 EA FF FF |  | = f 0 |  |
|---------|-------------------------|-------------------------|--|-------|--|
| es:0008 | AD DE E0 01 C5 15 AA 01 | i x8-S-0                |  |       |  |
| es:0010 | C5 15 89 02 20 10 92 01 | S20 f80                 |  |       |  |
| es:0018 | 01 03 01 00 02 FF FF FF | 0000 0                  |  |       |  |

ss:0102 0000  
ss:0100 0000

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

## Conclusion

To find the smallest number in a given string, we can make use of the CMP instruction which will compare the number of one memory location copied in AX with the number of other memory location. If the result of comparison does not generate a carry, then the number stored in the second memory location is smaller than the number in the first memory location which is copied in AX. But, as our area of interest is the smallest number stored in the second memory location, we will copy it to AX and then further cycle of comparison process will be continued till the counter becomes zero. Then at the end, the number in AX is the smallest number among all the string elements.

**EXPERIMENT NO. 10**

|                |   |
|----------------|---|
| <b>Title</b>   | 8086 Assembly Language Program to check if user entered string is palindrome or not   |
| <b>Outcome</b> | <b>CO2</b>  |
| <b>code</b>    | <pre>.model small .stack 100h .data     Arr db 9 dup(00h)     msg db 'enter string\$'     msg1 db 'string is palindrome\$'     msg2 db 'string is not palindrome\$' ends .code start: mov ax,@data       mov ds,ax       mov es,ax       mov cx,0005h       mov bx,0000h to_read_str:lea dx,msg             mov ah,09h             int 21h nxt_chr:mov ah,01h          int 21h          mov [bx],al          inc bx          loop nxt_chr          dec bx          mov si,0000h          mov di,bx          mov cx,0002h          CLD nxt_cmp:cmpsb         jne ahead         dec di         dec di         loop nxt_cmp         lea dx,msg1         mov ah,09h         int 21h         mov ah,01h         int 21h         jmp stop ahead:  lea dx,msg2         mov ah,09h         int 21h stop:  mov ah,4ch         int 21h         ends end start</pre> |

# Output

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

[CPU 80486] 1=[↑][↓]

|                |             |         |     |
|----------------|-------------|---------|-----|
| cs:0027 A6     | cmpsb       | ax 010D | c=0 |
| cs:0028 7512   | jne 003C    | bx 0004 | z=0 |
| cs:002A 4F     | dec di      | cx 0000 | s=0 |
| cs:002B 4F     | dec di      | dx 001E | o=0 |
| cs:002C E2F9   | loop 0027   | si 0002 | p=0 |
| cs:002E BA1E00 | mov dx,001E | di 0002 | a=0 |
| cs:0031 B409   | mov ah,09   | bp 0000 | i=1 |
| cs:0033 CD21   | int 21      | sp 0100 | d=0 |
| cs:0035 B401   | mov ah,01   | ds 48B1 |     |
| cs:0037 CD21   | int 21      | es 48B1 |     |
| cs:0039 EB08   | jmp 0043 ↓  | ss 48B6 |     |
| cs:003B 90     | nop         | cs 48AD |     |
| cs:003C BA3300 | mov dx,0033 | ip 0039 |     |

489D:0000 CD 20 FF 9F 00 EA FF FF = f Ω

489D:0008 AD DE E0 01 C5 15 AA 01 i |x0|S-0

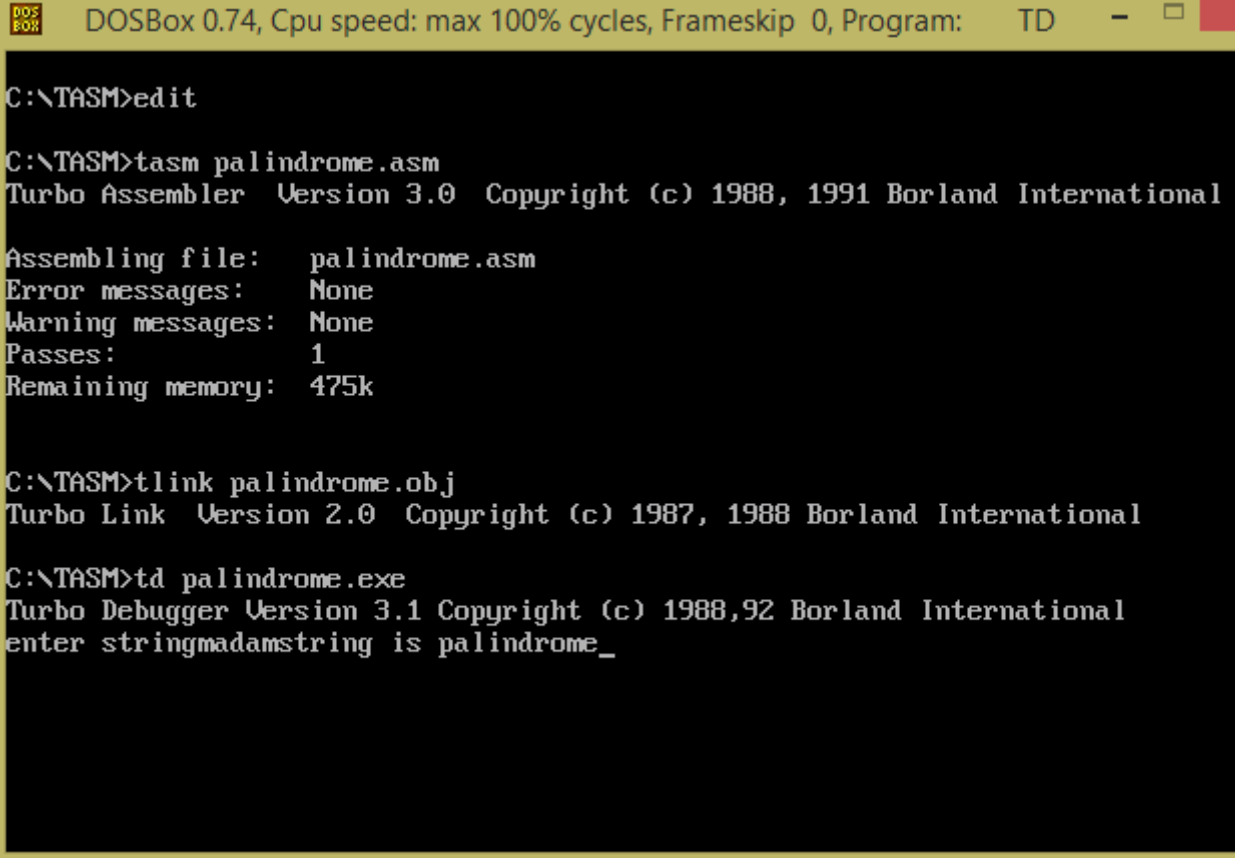
489D:0010 C5 15 89 02 20 10 92 01 |Se0 >ff0

489D:0018 01 03 01 00 02 FF FF FF 0♥0 0

ss:0102 48B1

ss:0100 0051

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

|                   |  |
|-------------------|--|
|                   |  <p>DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD</p> <pre> C:\TASM&gt;edit  C:\TASM&gt;tasm palindrome.asm Turbo Assembler Version 3.0 Copyright (c) 1988, 1991 Borland International  Assembling file:    palindrome.asm Error messages:    None Warning messages:  None Passes:            1 Remaining memory:  475k  C:\TASM&gt;tlink palindrome.obj Turbo Link Version 2.0 Copyright (c) 1987, 1988 Borland International  C:\TASM&gt;td palindrome.exe Turbo Debugger Version 3.1 Copyright (c) 1988,92 Borland International enter stringmadamstring is palindrome_ </pre> |
| <b>Conclusion</b> | <p>Palindrome check for a given string in 8086 Assembly language.</p> <p>The program prompts the user for a string and checks whether the given string is a palindrome displays the appropriate message.</p>   |