

UCS1512 – Microprocessors Lab

Matrix Operations

Exp no : 5

Name: Sreedhar V

Date : 09-10-2020

Reg no: 185001161

AIM:

To program and execute the matrix addition and matrix subtraction in 8086 using an emulator.

Matrix addition:

Algorithm:

- Program is set to run from any specified memory position.
- Move the address of data segment to register DS .
- Compare the dimensions of the two matrices (row1 equals rows 2 and col1 equals col2).
- If the dimensions doesn't match exit the program using jump instructions.
- Calculate the number of elements in the resultant matrix by multiplying number row and col value of either of the matrices
- Move the multiplied value to CX register as a counter.
- Move the offset of the mat1 and mat2 to SI and DI respectively.
- Move the offset of the result to BX register
- Repetitively add the contents of [SI] and [DI] and store the result in [BX] using an explicit loop until CX value becomes 0 and increment SI ,DI and BX registers each time inside the loop.
- Terminate the program.

Program:

CODE	COMMENT
<p>Program for Matrix Addition:</p> <p>assume code: cs,ds:data data segment row1 db 03h row2 db 03h col1 db 02h col2 db 02h org 0010h mat1 db 03h,01h,01h,01h,03h,04h org 0020h mat2 db 07h,04h,03h,01h,08h,06h org 0030h result db ? data ends</p> <p>code segment org 0100h start : mov ax,data mov ds,ax mov cl,row1 mov dl,row2 cmp cl,dl jne over mov cl,col1 mov dl,col2 cmp cl,dl jne over mov al,row2 mul cl mov cx,ax mov si, offset mat1 mov di, offset mat2 mov bx, offset result here: mov ah,00h mov al, [si] add al, [di] jnc here1 inc ah here1: mov [bx], al inc si inc di inc bx loop here over: mov ah,4ch int 21h code ends end start</p>	<p>Data segment is initialized row1,row2 is initialized to 03h and 03h col1,col2 is initialized to 02h and 02h</p> <p>mat1 is declared and initialized and address is set to 0010h</p> <p>mat2 is declared and initialized and address is set to 0020h</p> <p>result is declared and address is set to 0030h</p> <p>Code segment begins Originating address is set to 0100h Address of the data is transferred to AX , from AX transferred to DS. Move row1,row2 to CL and DL</p> <p>Compare CL and DL if not equal jump to label over</p> <p>Move col1,col2 to CL and DL</p> <p>Compare CL and DL if not equal jump to label over</p> <p>Move row2 to AL register Multiply CL with AL Move the result stored in AX to CX Offset of the mat1,mat2,result are transferred to SI, DI and BX respectively</p> <p>AH is initialized to 00h</p> <p>Add [SI] and [DI] , if there is carry increment AH register else jump to here1</p> <p>Move AL to [BX](result) Increment the SI,DI,BX registers</p> <p>Repeat till CX becomes 0</p> <p>Store the result Program terminates</p>

Unassembled code:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
-u
076E:0100 B86A07      MOV     AX,076A
076E:0103 8ED8        MOV     DS,AX
076E:0105 8A0E0000     MOV     CL,[0000]
076E:0109 8A160100     MOV     DL,[0001]
076E:010D 3BD1        CMP     CL,DL
076E:010F 752D        JNZ     013E
076E:0111 8A0E0200     MOV     CL,[0002]
076E:0115 8A160300     MOV     DL,[0003]
076E:0119 3BD1        CMP     CL,DL
076E:011B 7521        JNZ     013E
076E:011D A00100     MOV     AL,[0001]
-
```

Execution:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
There was 1 error detected.

D:\>debug 5a.exe
-d 076a:0000
076A:0000 03 03 02 02 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0010 03 01 01 01 03 04 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 07 04 03 01 08 06 00 00-00 00 00 00 00 00 00 00 .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
-g

Program terminated normally
-d 076a:0000
076A:0000 03 03 02 02 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0010 03 01 01 01 03 04 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 07 04 03 01 08 06 00 00-00 00 00 00 00 00 00 00 .....
076A:0030 0A 05 04 02 0B 0A 00 00-00 00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
-
```

Result:

Matrix addition is executed and verified using an emulator

Matrix subtraction:

Algorithm:

- Program is set to run from any specified memory position.
- Move the address of data segment to register DS .
- Compare the dimensions of the two matrices (row1 equals rows 2 and col1 equals col2).
- If the dimensions doesn't match exit the program using jump instructions.
- Calculate the number of elements in the resultant matrix by multiplying number row and col value of either of the matrices
- Move the multiplied value to CX register as a counter.
- Move the offset of the mat1 and mat2 to SI and DI respectively.
- Move the offset of the result to BX register
- Repetitively subtract the contents of [SI] and [DI] and store the result in [BX] using an explicit loop until CX value becomes 0 and increment SI ,DI and BX registers each time inside the loop.
- Terminate the program.

Program:

CODE	COMMENT
Program for Matrix Addition: assume code: cs,ds:data data segment row1 db 03h row2 db 03h col1 db 02h col2 db 02h org 0010h mat1 db 03h,01h,01h,01h,03h,04h org 0020h mat2 db 07h,04h,03h,01h,08h,06h org 0030h result db ? data ends code segment org 0100h start : mov ax,data mov ds,ax mov cl,row1 mov dl,row2 cmp cl,dl jne over mov cl,col1 mov dl,col2 cmp cl,dl jne over mov al,row2 mul cl mov cx,ax mov si, offset mat1 mov di, offset mat2 mov bx, offset result here: mov ah,00h mov al, [si] sub al, [di] jnc here1 inc ah here1: mov [bx], al inc si inc di inc bx loop here over: mov ah,4ch int 21h code ends end start	 Data segment is initialized row1,row2 is initialized to 03h and 03h col1,col2 is initialized to 02h and 02h mat1 is declared and initialized and address is set to 0010h mat2 is declared and initialized and address is set to 0020h result is declared and address is set to 0030h Code segment begins Originating address is set to 0100h Address of the data is transferred to AX , from AX transferred to DS. Move row1,row2 to CL and DL Compare CL and DL if not equal jump to label over Move col1,col2 to CL and DL Compare CL and DL if not equal jump to label over Move row2 to AL register Multiply CL with AL Move the result stored in AX to CX Offset of the mat1,mat2,result are transferred to SI, DI and BX respectively AH is initialized to 00h Subtract [SI] and [DI] , if there is carry increment AH register else jump to here1 Move AL to [BX](result) Increment the SI,DI,BX registers Repeat till CX becomes 0 Store the result Program terminates

Unassembled code:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
D:\>debug 5b.exe
-u
076E:0100 B86A07      MOV     AX,076A
076E:0103 8ED8        MOV     DS,AX
076E:0105 8A0E0000      MOV     CL,[0000]
076E:0109 8A160100      MOV     DL,[0001]
076E:010D 38D1        CMP     CL,DL
076E:010F 752D        JNZ     013E
076E:0111 8A0E0200      MOV     CL,[0002]
076E:0115 8A160300      MOV     DL,[0003]
076E:0119 38D1        CMP     CL,DL
076E:011B 7521        JNZ     013E
076E:011D A00100      MOV     AL,[0001]
```

Execution:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
-d 076a:0000
076A:0000 03 03 02 02 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0010 03 06 05 01 03 04 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 07 04 03 01 08 06 00 00-00 00 00 00 00 00 00 00 .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
-g
Program terminated normally
-d 076a:0000
076A:0000 03 03 02 02 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0010 03 06 05 01 03 04 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 07 04 03 01 08 06 00 00-00 00 00 00 00 00 00 00 .....
076A:0030 FC 02 02 00 FB FE 00 00-00 00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
```

Result:

Matrix subtraction is executed and verified using an emulator.

UCS1512 – Microprocessors Lab

Sorting

Exp no : 6

Name: Sreedhar V

Date : 09-10-2020

Reg no: 185001161

AIM:

To program and execute the code for sorting an array of N-bit values in ascending and descending order using bubble sort in 8086 using an emulator.

Sorting in ascending order:

Algorithm:

- Program is set to run from any specified memory position.
- Move the address of data segment to register DS .
- Move the length of the array to CH register(outer loop).
- Make a label loop1 for outer loop.
- Move the offset of the array to the SI register.
- Move the length of the array to CL register(inner loop).
- Make a label loop2 for inner loop.
- Move content of SI to AL and SI + 1 to AH.
- Compare AH and AL , if there is no carry produced jump to label skip ,else swap the two values using XCHG instruction.
- Move the content of AL to [SI] and content of AH to [SI+1].
- Make a label skip to skip the swapping process if the elements in correct order.
- Increment SI and decrement CL ,till CL becomes zero jump to label loop2.
- Decrement CH ,till CH becomes zeros jump to label loop1.
- Terminate the program.

Program:

CODE	COMMENT
Program Sorting ascending order:	
assume cs:code,ds:data	
data segment ;assuming N to be 10 array db 05h,03h,02h,07h,06h,01h,00h,09h,08h,04h data ends	Data segment is initialized decimal is declared and initialized.
code segment org 0100h	Code segment begins Originating address is set to 0100h Address of the data is transferred to AX , from AX transferred to DS.
start : mov ax,data mov ds,ax mov ch , 09h	Move 09h to CH (outer loop count)
loop1 : mov si ,offset array mov cl , 09h	label loop1 Move the offset of array to SI Move 09h to CL(inner loop count)
loop2 : mov al , [si] mov ah , [si+1] cmp ah , al jnc skip xchg al,ah mov [si] ,al mov [si+1],ah	label loop2 Move [SI] to AL and [SI+1] to AH registers Compare AH and AL If there is no carry(i.e they are correct order) jump to skip else swap AL and AH Move AL to [SI] and AH to [SI+1] again
skip: inc si dec cl jnz loop2 dec ch jnz loop1 mov ah,4ch int 21h	label skip increment SI decrement CL if CL is not zero jump to loop2. decrement CH if CH is not zero jump to loop1
code ends end start	Program terminates

Unassembled code:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
D:\>debug 6a.exe
-u
076B:0100 B86A07      MOV     AX,076A
076B:0103 8ED8        MOV     DS,AX
076B:0105 B509        MOV     CH,09
076B:0107 BE0000     MOV     SI,0000
076B:010A B109        MOV     CL,09
076B:010C 8A04        MOV     AL,[SI]
076B:010E 8A6401     MOV     AH,[SI+01]
076B:0111 38C4        CMP     AH,AL
076B:0113 7307        JNB     011C
076B:0115 86C4        XCHG    AL,AH
076B:0117 8804        MOV     [SI],AL
076B:0119 886401     MOV     [SI+01],AH
076B:011C 46          INC     SI
076B:011D FEC9        DEC     CL
076B:011F 75EB        JNZ     010C
```

Execution:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
-d 076a:0000
076A:0000 05 03 02 07 06 01 00 09-08 04 00 00 00 00 00 00 .....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
-g
Program terminated normally
-d 076a:0000
076A:0000 00 01 02 03 04 05 06 07-08 09 00 00 00 00 00 00 .....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
```

Result:

Sorting in ascending order is executed and verified using an emulator.

Sorting in descending order:

Algorithm:

- Program is set to run from any specified memory position.
- Move the address of data segment to register DS .
- Move the length of the array to CH register(outer loop).
- Make a label loop1 for outer loop.
- Move the offset of the array to the SI register.
- Move the length of the array to CL register(inner loop).
- Make a label loop2 for inner loop.
- Move content of SI to AL and SI + 1 to AH.
- Compare AH and AL , if there is a carry produced jump to label skip else swap the two values using XCHG instruction.
- Move the content of AL to [SI] and content of AH to [SI+1].
- Make a label skip to skip the swapping process if the elements in correct order.
- Increment SI and decrement CL ,till CL becomes zero jump to label loop2.
- Decrement CH ,till CH becomes zeros jump to label loop1.
- Terminate the program.

Program:

CODE	COMMENT
Program Sorting descending order: assume cs:code,ds:data data segment ;assuming N to be 10 array db 05h,03h,02h,07h,06h,01h,00h,09h,08h,04h data ends code segment org 0100h start : mov ax,data mov ds,ax mov ch , 09h loop1 : mov si ,offset array mov cl , 09h loop2 : mov al , [si] mov ah , [si+1] cmp ah , al jc skip xchg al,ah mov [si] ,al mov [si+1],ah skip: inc si dec cl jnz loop2 dec ch jnz loop1 mov ah,4ch int 21h code ends end start	 Data segment is initialized decimal is declared and initialized. Code segment begins Originating address is set to 0100h Address of the data is transferred to AX , from AX transferred to DS. Move 09h to CH (outer loop count) label loop1 Move the offset of array to SI Move 09h to CL(inner loop count) label loop2 Move [SI] to AL and [SI+1] to AH registers Compare AH and AL If there is a carry(i.e they are correct order) jump to skip else swap AL and AH Move AL to [SI] and AH to [SI+1] again label skip increment SI decrement CL if CL is not zero jump to loop2. decrement CH if CH is not zero jump to loop1 Program terminates

Unassembled code:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
D:\>debug 6b.exe
-u
076B:0100 B86A07      MOV     AX,076A
076B:0103 8ED8        MOV     DS,AX
076B:0105 B509        MOV     CH,09
076B:0107 BE0000      MOV     SI,0000
076B:010A B109        MOV     CL,09
076B:010C 8A04        MOV     AL,[SI]
076B:010E 8A6401      MOV     AH,[SI+01]
076B:0111 38C4        CMP     AH,AL
076B:0113 7207        JB      011C
076B:0115 86C4        XCHG    AL,AH
076B:0117 8804        MOV     [SI],AL
076B:0119 886401      MOV     [SI+01],AH
076B:011C 46          INC     SI
076B:011D FEC9        DEC     CL
076B:011F 75EB        JNZ     010C
-
```

Execution:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
-d 076a:0000
076A:0000 05 03 02 07 06 01 00 09-08 04 00 00 00 00 00 00 .....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
-g
Program terminated normally
-d 076a:0000
076A:0000 09 08 07 06 05 04 03 02-01 00 00 00 00 00 00 00 .....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
-
```

Result:

Sorting in descending order is executed and verified using an emulator.

UCS1512 – Microprocessors Lab

8 BIT ARITHMETIC OPERATIONS

Exp no : 7

Name: Sreedhar V

Date : 09-10-2020

Reg no: 185001161

AIM:

To program and execute the 8 bit BCD addition and subtraction in 8086 using an emulator.

8-Bit BCD Addition:

Algorithm:

- Program is set to run from any specified memory position.
- Load data from opr1 to register AL (first number).
- Load data from opr2 to register BL (second number).
- Add these two numbers (contents of register AL and register BL).
- Initialize carry to 0.
- Decimal adjust after addition using DAA instruction for converting the result to BCD form.
- Jump to final steps if there is no carry.
- Increment carry.
- Store additional values to result.
- Terminate the program

Program:

CODE	COMMENT
<pre>;Program for 8-bit BCD addition assume cs:code,ds:data data segment opr1 db 99h opr2 db 99h result db 00H carry db 00H data ends code segment org 0100h start: mov ax,data mov ds,ax mov al,opr1 mov bl,opr2 mov ch,00h add al,bl daa jnc here inc ch here: mov result,al mov carry,ch mov ah,4ch int 21h code ends end start</pre>	<p>Data segment initialized opr1 initialised and set to 99 opr2 initialised and set to 99 result initialised and set to 00 carry initialised and set to 00</p> <p>Code segment begins Originating address is set at 0100</p> <p>Address of data segment moved to ax From ax, transferred to ds Value of opr1 transferred to al Value of opr2 transferred to bl ch is initialised and set to 0 Addition takes place</p> <p>DAA is used to decimal adjust after addition.(i.e) if the lower nibble(AL) > 9 or AF =1 it adds 06h to AL and if the higher nibble(AL) > 9 or CF=1 it adds 60h to AH.</p> <p>Junction created</p> <ul style="list-style-type: none">• Jump if no carry• Else increment ch <p>data transferred from al to result data transferred from ch to carry</p> <p>Program terminates</p>

Unassembled code:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
D:\>debug 7a.exe
-u
076B:0100 B86A07      MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A00000          MOV     AL,[0000]
076B:0108 8A1E0100       MOV     BL,[0001]
076B:010C B500          MOV     CH,00
076B:010E 02C3          ADD     AL,BL
076B:0110 27          DAA
076B:0111 7302          JNB     0115
076B:0113 FEC5          INC     CH
076B:0115 A20300       MOV     [0003],AL
076B:0118 8B2E0200       MOV     [0002],CH
076B:011C B44C          MOV     AH,4C
076B:011E CD21          INT     21
```

Execution:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
-d 076a:0000
076A:0000 99 99 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-g
Program terminated normally
-d 076a:0000
076A:0000 99 99 01 98 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

Result:

8- Bit BCD addition is executed and verified using an emulator.

8-Bit BCD Subtraction:

Algorithm:

- Program is set to run from any specified memory position.
- Load data from opr1 to register AL (first number).
- Load data from opr2 to register BL (second number).
- Add these two numbers (contents of register AL and register BL).
- Initialize carry to 0.
- Decimal adjust after subtraction using DAS instruction for converting the result to BCD form.
- Jump to final steps if there is no carry.
- If carry produced, increment carry.
- Convert the result to 10's complement by subtracting result with 99h and add the value with 01h .
- Again use DAS instruction for converting the result to BCD form.
- Store additional values to result.
- Terminate the program

Program:

CODE	COMMENT
<pre>;Program for 8-bit BCD subtraction assume cs:code,ds:data data segment opr1 db 99h opr2 db 99h result db 00H carry db 00H data ends code segment org 0100h start: mov ax,data mov ds,ax mov al,opr1 mov bl,opr2 mov ch,00h add al,bl das jnc here inc ch mov cl, 99h sub cl,al add cl,ch mov al,cl das here: mov result,ah mov carry,ch mov ah,4ch int 21h code ends end start</pre>	<p>Data segment initialized opr1 initialised and set to 99 opr2 initialised and set to 99 result initialised and set to 00 carry initialised and set to 00</p> <p>Code segment begins Originating address is set at 0100</p> <p>Address of data segment moved to ax From ax, transferred to ds Value of opr1 transferred to al Value of opr2 transferred to bl ch is initialised and set to 0 Addition takes place</p> <p>DAS is used to decimal adjust after subtraction.(i.e) if the lower nibble(AL) > 9 or AF =1 it subtracts 06h to AL and if the higher nibble(AL) > 9 or CF=1 it subtracts 60h to AH. Junction created</p> <ul style="list-style-type: none">• Jump if no carry• Else increment ch and take the 10's complement of the result• And adjust for BCD using DAS instruction. <p>data transferred from al to result data transferred from ch to carry</p> <p>Program terminates</p>

Unassembled code:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
D:\>debug 7b.exe
-u
076B:0100 B86A07      MOV     AX,076A
076B:0103 B8D8      MOV     DS,AX
076B:0105 A00000      MOV     AL,[0000]
076B:0108 8A1E0100      MOV     BL,[0001]
076B:010C B500      MOV     CH,00
076B:010E 2AC3      SUB     AL,BL
076B:0110 2F      DAS
076B:0111 730B      JNB     011E
076B:0113 FEC5      INC     CH
076B:0115 B199      MOV     CL,99
076B:0117 2AC8      SUB     CL,AL
076B:0119 02CD      ADD     CL,CH
076B:011B 8AC1      MOV     AL,CL
076B:011D 2F      DAS
076B:011E A20200      MOV     [0002],AL
```

Execution:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
-d 076a:0000
076A:0000 77 8B 00 00 00 00 00 00-00 00 00 00 00 00 00 00  w.....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
-g
Program terminated normally
-d 076a:0000
076A:0000 77 8B 11 01 00 00 00 00-00 00 00 00 00 00 00 00  w.....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
```

Result:

8- Bit BCD subtraction is executed and verified using an emulator.