Exp No: 5

Date: 08/10/2020

Matrix operations

Name: Swetha Saseendran

Reg No: 185001183

Aim:

To write and execute 8086 programs for Matrix operations (Addition & Subtraction).

Programs:

(i) Matrix Addition

Algorithm:

- Declare the data segment.
- Initialize data segment with matrices 1 and 2, with their dimensions and resultant matrix.
- Close the data segment.

- Declare the code segment.
- Set a preferred offset (preferably 100)
- Load the data segment content into AX register.
- Transfer the contents of AX register to DS register.
- Compare row1 and row2, if not equal then exit the program.
- Compare col1 and col2, if not equal then exit the program.
- Position SI at matrix1, and DI at matrix2.
- Multiply row1 and col1 to find length len of the matrix.
- Move the len to CL register.
- Till CL goes to zero:
    - Add values at SI and DI and push it into the stack.
    - Increment SI and DI.
    - Decrement CL.
- Move SI to end of resultant matrix.
- Till CL goes to zero:
    - Pop the value from top of the stack and put it at SI.
    - Decrement SI.

Program	Comments
assume *cs:code, ds:data*	Using assume directive to declare data, extra and code segment
data segment	Using assume directive to declare data, extra and code segment
mat1 db 23h,24h,55h,11h	
mat2 db 21h,44h,57h,22h	
row1 db 02h	
col1 db 02h	
row2 db 02h	
col2 db 02h	
len db 00h	
resi dw ?	
data ends	
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: *mov ax, data*	Transfer data from "data" to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov al, row1	Move row1 to AL
mov bl, row2	Move row2 to BL
cmp al, bl	Comparing row count of both matrices.
jne break	Exiting if not same.
mov al, col1	Move col1 to AL
mov bl, col2	Move col2 to BL
cmp al, bl	Comparing col count of both matrices.
jne break	Exiting if not same.
mov si, offset mat1	Set SI to point to Matrix 1's starting index.
mov di, offset mat2	Set DI to point to Matrix 2's starting index.
mov al, row1	Move row1 to AL
mov bl, col1	Move row2 to BL
mul bl	AL has the value of row1 * col1.
mov len, al	Move len to AL
mov cl, len	Finding no. of elements in the matrix.
mov ch, 00h	Clear CH.
mov ax, 0000h	Clear AX.
looper: *mov al, [si]*	Pushing each element-wise sum into stack
mov ah, 00h	AH <- 00H
mov bl, [di]	
mov bh, 00h	BH <- 00H
add ax, bx	Add the 2 elements from each matrix.

push ax	
inc si	Move to next element in matrix 1.
inc di	Move to next element in matrix 2.
dec cx	Decrement counter by 1.
jz prewrk	If addition is over, jump to prewrk
jmp looper	Repeat addition for all elements.
prewrk: *mov si, offset resi + 0001h*	Set the SI to store values in result matrix "resi" properly.
mov cl, len	Set counter to length of the matrix.
mov ch, 00h	Clear CH.
add si, cx	Set SI to point to the last location of the matrix.
retloop: *pop ax*	Popping each element from stack into resultant matrix.
mov [si], al	Move AL to [SI]
dec si	Decrement SI.
mov [si], ah	Move AL to [SI]
dec si	Decrement SI
dec cx	Decrement counter by 1.
jz break	Stop popping if all elements are popped (CX = 0)
jmp retloop	Pop the next element and put it in the matrix.
break: *mov ah, 4ch*	Moves the hexadecimal value 4c to ah.
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).
code ends	
end start	

Unassembled Code:

```
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A00800        MOV     AL,[0008]
076B:0108 8A1E0A00      MOV     BL,[000A]
076B:010C 38D8          CMP     AL,BL
076B:010E 7551          JNZ     0161
076B:0110 A00900        MOV     AL,[0009]
076B:0113 8A1E0B00      MOV     BL,[000B]
076B:0117 38D8          CMP     AL,BL
076B:0119 7546          JNZ     0161
076B:011B BE0000        MOV     SI,0000
076B:011E BF0400        MOV     DI,0004
```

Snapshot of sample input and output:

INPUT:

```
-d 076A:0000
076A:0000  23 24 55 11 21 44 57 22-02 02 02 02 00 00 00 00   #$U.!DW".......
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   ................
```

OUTPUT:

```
-g

Program terminated normally
-d 076A:0000
076A:0000  23 24 55 11 21 44 57 22-02 02 02 02 04 00 00 00   #$U.!DW".......
076A:0010  44 00 68 00 AC 00 33 00-00 00 00 00 00 00 00 00   D.h...3.........
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   ................
-
```

(ii) Matrix Subtraction

Algorithm:

- Declare the data segment.
- Initialize data segment with matrices 1 and 2, with their dimensions and resultant matrix.
- Close the data segment.

- Declare the code segment.
- Set a preferred offset (preferably 100)
- Load the data segment content into AX register.
- Transfer the contents of AX register to DS register.
- Compare row1 and row2, if not equal then exit the program
- Compare col1 and col2, if not equal then exit the program
- Position SI at matrix1, and DI at matrix2.
- Multiply row1 and col1 to find length len of the matrix.
- Move the len to CL register.
- Till CL goes to zero:
    - Subtract values at SI and DI and push it into the stack.
    - Increment SI and DI.
    - Decrement CL.
- Move SI to end of resultant matrix.
- Till CL goes to zero:
    - Pop the value from top of the stack and put it at SI.
    - Decrement SI.

Program	*Comments*
assume *cs:code, ds:data*	Using assume directive to declare data, extra and code segment
data segment	Using assume directive to declare data, extra and code segment
mat1 db 23h,24h,55h,11h	
mat2 db 21h,44h,57h,22h	
row1 db 02h	

col1 db 02h	
row2 db 02h	
col2 db 02h	
len db 00H	
resi dw ?	
data ends	
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: *mov ax, data*	Transfer data from "data" to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov al, row1	Move row1 to AL
mov bl, row2	Move row2 to BL
cmp al, bl	Comparing row count of both matrices.
jne break	Exiting if not same.
mov al, col1	Move col1 to AL
mov bl, col2	Move col2 to BL
cmp al, bl	Comparing col count of both matrices.
jne break	Exiting if not same.
mov si, offset mat1	Set SI to point to Matrix 1's starting index.
mov di, offset mat2	Set DI to point to Matrix 2's starting index.
mov al, row1	Move row1 to AL
mov bl, col1	Move row2 to BL
mul bl	AL has the value of row1 * col1.
mov len, al	Move len to AL
mov cl, len	Finding no. of elements in the matrix.
mov ch, 00h	Clear CH.
mov ax, 0000h	Clear AX.
looper: *mov al, [si]*	Pushing each element-wise sum into stack
mov ah, 00h	AH <- 00H
mov bl, [di]	
mov bh, 00h	BH <- 00H
sub ax, bx	Subtract the 2 elements from each matrix.
push ax	
inc si	Move to next element in matrix 2.
inc di	Move to next element in matrix 1.
dec cx	Decrement counter by 1.
jz prewrk	If addition is over, jump to "prewrk"
jmp looper	Repeat addition for all elements.

prewrk:	*mov si, offset resi + 0001h*	Set the SI to store values in result matrix “resi” properly.
	mov cl, len	Set counter to length of the matrix.
	mov ch, 00h	Clear CH.
	add si, cx	Set SI to point to the last location of the matrix.
	add si, cx	
retloop:	*pop ax*	Popping each element from stack into resultant matrix.
	mov [si], al	Move AL to [SI]
	dec si	Decrement SI.
	mov [si], ah	Move AL to [SI]
	dec si	Decrement SI
	dec cx	Decrement counter by 1.
	jz break	Stop popping if all elements are popped (CX = 0)
	jmp retloop	Pop the next element and put it in the matrix.
break:	*mov ah, 4ch*	Moves the hexadecimal value 4c to ah.
	int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).
	code ends	
	end start	

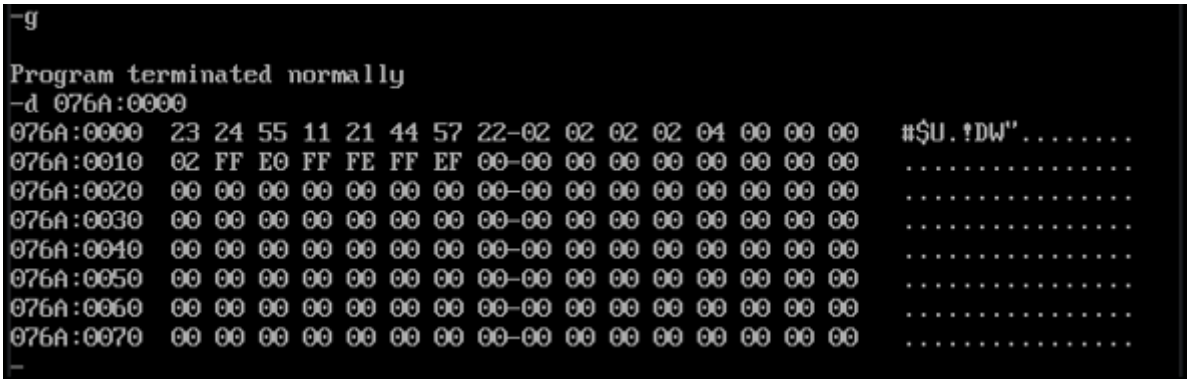
Unassembled Code:

```
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A00800        MOV     AL,[0008]
076B:0108 8A1E0A00      MOV     BL,[000A]
076B:010C 38D8          CMP     AL,BL
076B:010E 7551          JNZ     0161
076B:0110 A00900        MOV     AL,[0009]
076B:0113 8A1E0B00      MOV     BL,[000B]
076B:0117 38D8          CMP     AL,BL
076B:0119 7546          JNZ     0161
076B:011B BE0000        MOV     SI,0000
076B:011E BF0400        MOV     DI,0004
```

Snapshot of sample input and output:

INPUT:

OUTPUT:



Result:

The assembly level programs were written to perform the above specified matrix operations and the result was verified.

Exp No: 6 `Date:08 /10/2020

Sorting

Name: Swetha Saseendran

Reg No: 185001183

Aim:

To program and execute the sorting of 8 bit N values in ascending and descending order in 8086 microprocessor using DOSBOX.

Programs:

(i) SORTING IN ASCENDING ORDER

Algorithm:

- Program is set to run from any specified memory position.
- Load data from arr to register AX.
- Compare the digits in arr move the smaller to front and larger to back.
- Use the instruction XCHG to mov of e between the digits
- Move the digits until zero flag becomes zero and length of arr becomes zero
- Terminate the program.

Program	Comments
;To sort a set of numbers in an arr in ascending order	
DATA SEGMENT	
arr DB 05H, 04H, 03H, 02H, 01H	Array with 05, 04, 03, 02, 01 as input
arrlen DB 04H	Array length as 04
DATA ENDS	
ASSUME CS:CODE,DS:DATA	
Code SEGMENT	
START: MOV AX,DATA	Address of data segment moved to ax

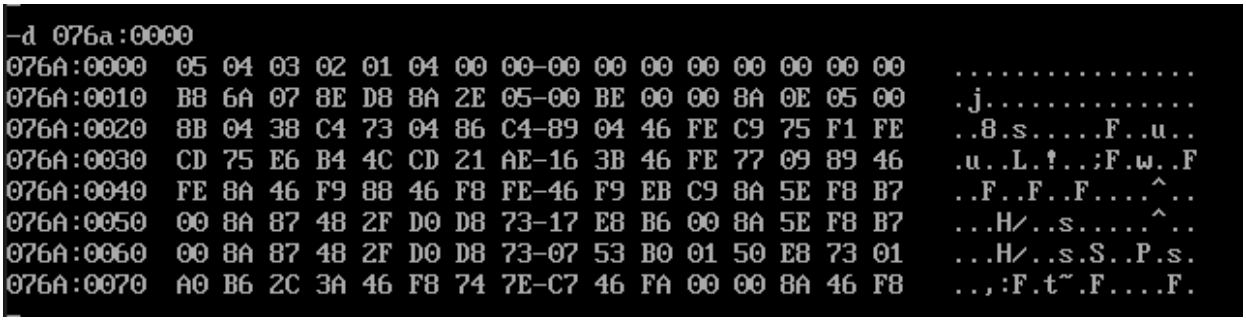
MOV DS,AX	Address of ax moved to ax
MOV CH, arrlen ;outer loop	Value of arrlen moved to ch
iteration	Starting pointer of arr
OUTER: MOV SI, offset(arr)	Inner loop iteration (reinitialize)
MOV CL, arrlen ;	
INNER: MOV AX, [SI]	
CMP AH, AL	Jump if no carry to SKIP
JNC SKIP	
XCHG AL, AH	AH AL is stored together
MOV [SI], AX	
SKIP: INC SI	Decrease inner loop
DEC CL	
JNZ INNER	Decease outer loop
DEC CH	
JNZ OUTER	Terminate the program
MOV AH,4CH	
INT 21H	
Code ENDS	
END START	
END	

Unassembled Code:

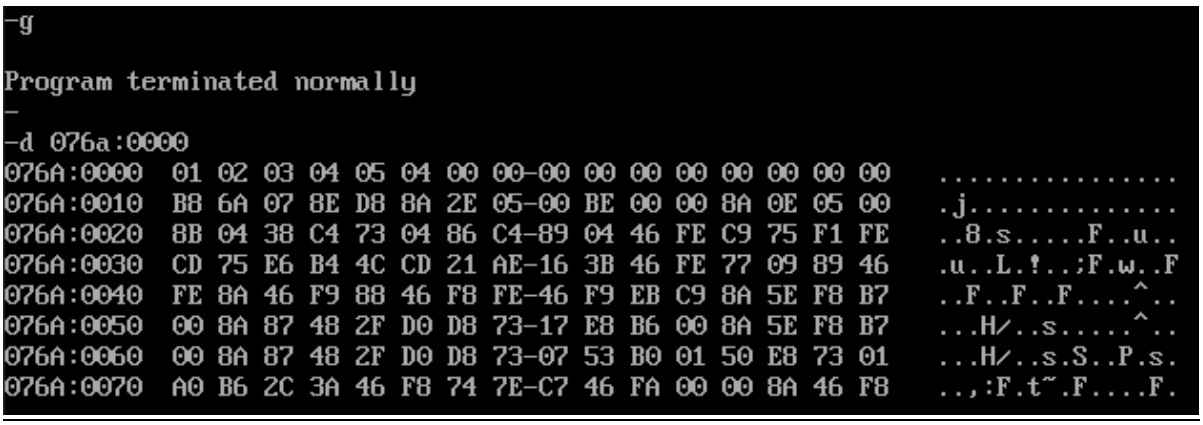
```
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 8A2E0500      MOV     CH,[0005]
076B:0009 BE0000        MOV     SI,0000
076B:000C 8A0E0500      MOV     CL,[0005]
076B:0010 8B04          MOV     AX,[SI]
076B:0012 38C4          CMP     AH,AL
076B:0014 7304          JNB     001A
076B:0016 86C4          XCHG    AL,AH
076B:0018 8904          MOV     [SI],AX
076B:001A 46            INC     SI
076B:001B FEC9          DEC     CL
076B:001D 75F1          JNZ     0010
076B:001F FECD          DEC     CH
-
```

Snapshot of sample input and output:

INPUT:



OUTPUT:



(ii) SORTING IN DESCENDING ORDER

Algorithm:

- Program is set to run from any specified memory position.
- Load data from arr to register AX.
- Compare the digits in arr move the larger digit to front and smaller digit to back of arr.
- Use the instruction XCHG to move between the digits
- Move the digits until zero flag becomes zero and length of arr becomes zero
- Terminate the program.

Program	Comments
;To sort a set of numbers in an arr in descending order	
DATA SEGMENT	
arr DB 01H, 02H, 03H, 04H, 05H	Array with 01, 02, 03, 04, 05 as input
arrlen DB 04H	Array length as 04
DATA ENDS	
ASSUME CS:CODE,DS:DATA	
Code SEGMENT	
START: MOV AX,DATA	Address of data segment moved to ax
MOV DS,AX	Address of ax moved to ax
MOV CH, arrlen	Value of arrlen moved to ch
OUTER: MOV SI, offset(arr)	Starting pointer of arr
MOV CL, arrlen	Inner loop iteration (reinitialize)
INNER: MOV AX, [SI]	
CMP AH, AL	
JC SKIP	Jump if carry to SKIP
XCHG AL, AH	
MOV [SI], AX	AH AL is stored together
SKIP: INC SI	
DEC CL	Decrease inner loop
JNZ INNER	
DEC CH	Decease outer loop
JNZ OUTER	
MOV AH,4CH	
INT 21H	
Code ENDS	Terminate the program
END START	
END	

Unassembled Code:

```
-
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 8A2E0500      MOV     CH,[0005]
076B:0009 BE0000        MOV     SI,0000
076B:000C 8A0E0500      MOV     CL,[0005]
076B:0010 8B04          MOV     AX,[SI]
076B:0012 38C4          CMP     AH,AL
076B:0014 7204          JB      001A
076B:0016 86C4          XCHG    AL,AH
076B:0018 8904          MOV     [SI],AX
076B:001A 46            INC     SI
076B:001B FEC9          DEC     CL
076B:001D 75F1          JNZ     0010
076B:001F FECD          DEC     CH
-_
```

Snapshot of sample input and output:

INPUT:

```
-
-d 076a:0000
076A:0000  01 02 03 04 05 04 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 8A 2E 05-00 BE 00 00 8A 0E 05 00   .j..............
076A:0020  8B 04 38 C4 72 04 86 C4-89 04 46 FE C9 75 F1 FE   ..8.r.....F..u..
076A:0030  CD 75 E6 B4 4C CD 21 AE-16 3B 46 FE 77 09 89 46   .u..L.!..;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-
```

OUTPUT:

```
-g

Program terminated normally
-d 076a:0000
076A:0000  05 04 03 02 01 04 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 8A 2E 05-00 BE 00 00 8A 0E 05 00   .j..............
076A:0020  8B 04 38 C4 72 04 86 C4-89 04 46 FE C9 75 F1 FE   ..8.r.....F..u..
076A:0030  CD 75 E6 B4 4C CD 21 AE-16 3B 46 FE 77 09 89 46   .u..L.!..;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-_
```

Result:

Therefore, the ascending and descending sorting are performed and verified using MASM.

Exp No: 7 **Date:** 08/10/2020

BCD ADDITION AND SUBTRACTION **Name:** Swetha Saseendran

Reg No: 185001183

Aim:

To program and execute 8 bit BCD Addition and Subtraction using DOS-BOX.

(i) BCD Addition

Programs:

Algorithm:

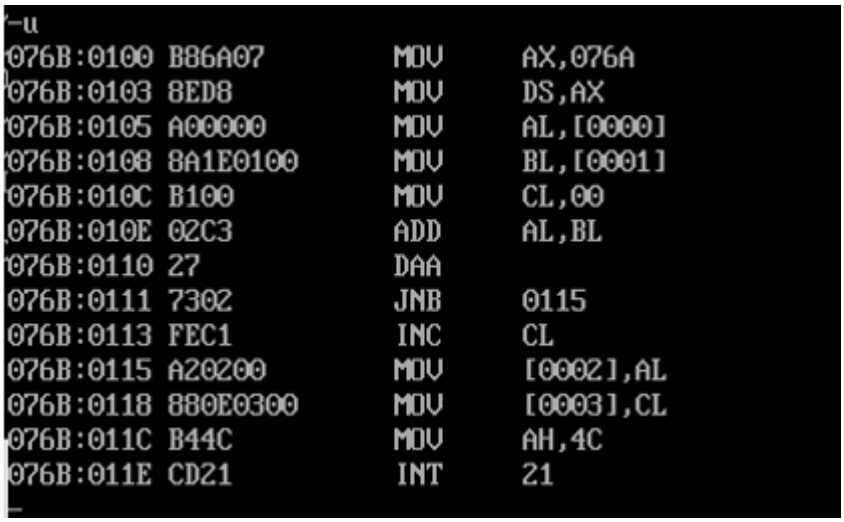
- Program is set to run from any specified memory position.
- Load data from opr1 to register AL (first operand)
- Load data from opr2 to register BL (second operand)
- Initialize carry to 0.
- Add these two numbers (contents of register AL and register BL)
- Decimal adjust after addition
- Jump to here label if there is no carry.
- Increment carry. Store additional values to result.
- Store additional values to result.
- Terminate the program.

Program	*Comments*
assume *cs:code, ds:data*	Using assume directive to declare data, extra and code segment
data segment	Declaring and initialising variables in data segment
opr1 db 11h	
opr2 db 99h	

result db 00h	
carry db 00h	
data ends	Data segment ends
code segment	Start of code segment
org 0100h	Originating address ←0100
start:	
mov ax, data	AX ←data
mov ds, ax	DS←ax
mov al, opr1	Move opr1 to AL register
mov bl, opr2	Move opr2 to BL register
mov cl,00h	CL←00h
add al, bl	AL=AL+BL
daa	Decimal adjust after addition
jnc here	Jump if no carry to here
inc cl	Increment CL
here:	
mov result, al	Result ←AL
mov carry, cl	Move opr1 to AL register
Mov ah, 4ch	AH←4Ch
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).

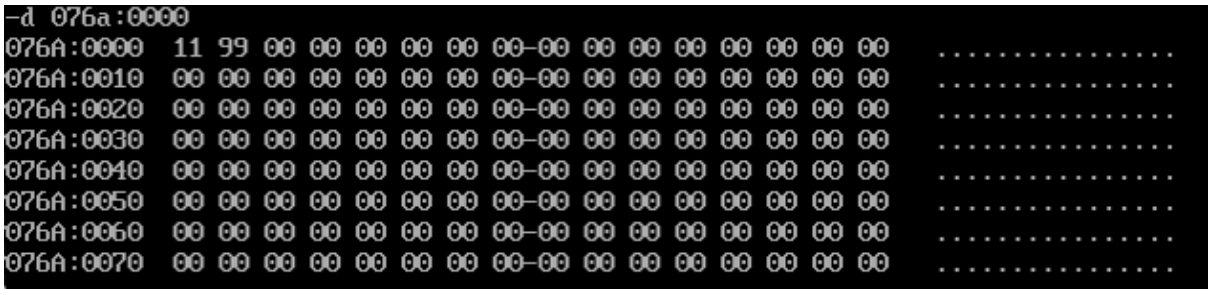
code ends	Code segment ends
end start	End of start label

Unassembled Code:

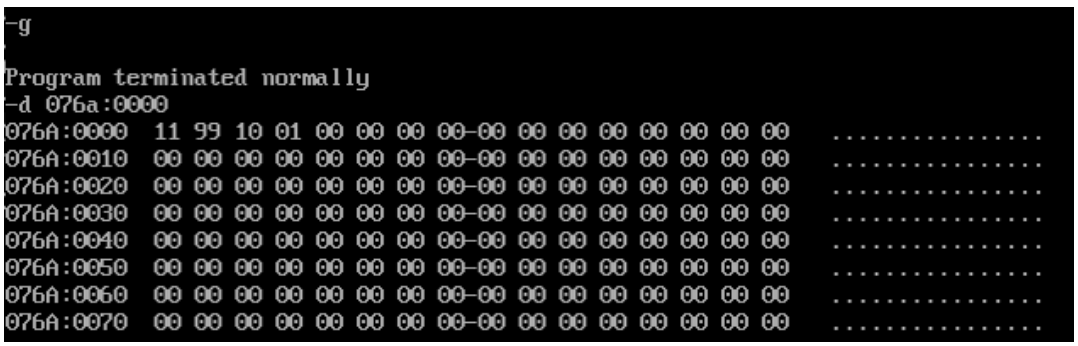


Snapshot of sample input and output:

INPUT:



OUTPUT:



(ii) BCD Subtraction

Algorithm:

- Program is set to run from any specified memory position.
- Load data from opr1 to register AL (first operand)
- Load data from opr2 to register BL (second operand)
- Initialize carry to 0.
- Subtract these two numbers (contents of register AL and register BL)
- Decimal adjust after subtraction
- Jump to here label if there is no carry.
- Increment carry. Store additional values to result.
- Store additional values to result.
- Terminate the program

Program	*Comments*
assume *cs:code,ds:data*	Using assume directive to declare data, extra and code segment
data segment	Declaring and initialising variables in data segment
opr1 db 11h	
opr2 db 99h	
result db 00h	
carry db 00h	
data ends	Data segment ends
code segment	Start of code segment
org 0100h	Originating address ←0100
start:	
mov ax,data	AX ←data

mov ds,ax	DS←ax
mov al, opr1	Move opr1 to AL register
mov bl, opr2	Move opr2 to BL register
mov cl, 00h	CL←00h
sub al, bl	AL=AL-BL
das	Decimal adjust after subtraction
jnc here	Jump if no carry to here
inc cl	Increment CL
mov dl, 99h	DL←99h
sub dl, al	DL=DL-AL
add dl, 01h	DL=DL+01h
mov al, dl	Move value in DL to AL
daa	
here:	
mov result, al	Result ←AL
mov carry, cl	carry←CL
mov ah, 4ch	AH←4Ch
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).
code ends	Code segment ends
end start	End of start label

Unassembled Code:

```
-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 B100          MOV     CL,00
076B:010E 2AC3          SUB     AL,BL
076B:0110 2F            DAS
076B:0111 730C          JNB     011F
076B:0113 FEC1          INC     CL
076B:0115 B299          MOV     DL,99
076B:0117 2AD0          SUB     DL,AL
076B:0119 80C201        ADD     DL,01
076B:011C 8AC2          MOV     AL,DL
076B:011E 27            DAA
076B:011F A20200        MOV     [0002],AL
```

Snapshot of sample input and output:

INPUT:

```
-d 076a:0000
076A:0000  11 99 00 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 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   ................
```

OUTPUT:

```
-g

Program terminated normally
-d 076a:0000
076A:0000  11 99 88 01 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 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 Addition and Subtraction have been programmed and executed in 8086 microprocessor using DOS-BOX.