Exp No: 8 **Date:** 19/10/2020

Name: Swetha Saseendran

Reg No: 185001183

CASE CONVERSION

Aim:

To program and execute case conversions using 8086 microprocessor using DOSBOX.

Algorithm:

- Count carries the value od number of characters.
- In a loop input characters. Compare input value with 60h.
- If AL>60h, move to upper.
- Else add 20h.then move to skip.
- In Upper subtract 20h from AL.
- In skip output character using mov functions.
- End the program.

PROGRAM	*COMMENT*
; Program to convert case	
ASSUME *CS: CODE, DS: data*	
data SEGMENT	DEFINE DATA SEGMENT
COUNT equ 10h	

```
data ends

CODE SEGMENT

START:  MOV AX, data
        MOV DS, AX
        MOV CX, COUNT        LOOP COUNTER
        L1:MOV AH,1,         INPUT CHARACTER
        INT 21H

        CMP AL,60H           IF AL IS greater THAN 60
        JNC UPPER
        ADD AL,20H
        JMP SKIP             CONVERT TO UPPER CASE
UPPER:  SUB AL,20H           CHARACTER OUTPUT FUNCTION
SKIP:   MOV AH,2             CHARACTER MUST BE IN DL
        MOV DL, AL           DISPLAY THE CHARACTER
        INT 21H              REPEAT LOOP
        LOOP L1
        MOV AH,4CH           TERMINATE THE PROGRAM
        INT 21H

CODE ENDS

end start
```

Unassembled Code:

```
-u
076A:0000 B86A07        MOV     AX,076A
076A:0003 8ED8          MOV     DS,AX
076A:0005 B91000        MOV     CX,0010
076A:0008 B401          MOV     AH,01
076A:000A CD21          INT     21
076A:000C 3C60          CMP     AL,60
076A:000E 7304          JNB     0014
076A:0010 0420          ADD     AL,20
076A:0012 EB02          JMP     0016
076A:0014 2C20          SUB     AL,20
076A:0016 B402          MOV     AH,02
076A:0018 8AD0          MOV     DL,AL
076A:001A CD21          INT     21
076A:001C E2EA          LOOP    0008
076A:001E B44C          MOV     AH,4C
-
```

Snapshot of sample input and output:

INPUT:

```
-d 076a:0000
076A:0000  B8 6A 07 8E D8 B9 10 00-B4 01 CD 21 3C 60 73 04   .j.........!<`s.
076A:0010  04 20 EB 02 2C 20 B4 02-8A D0 CD 21 E2 EA B4 4C   . .., .....!...L
076A:0020  CD 21 80 BF B8 2C 00 75-05 88 46 F8 EB 1E 8A 5E   .!...,.u..F....^
076A:0030  F9 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   .........;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
aAAabBBbcCCcdDDdeEEefFFfgGGghHHh
Program terminated normally
-
-d 076a:0000
076A:0000  B8 6A 07 8E D8 B9 10 00-B4 01 CD 21 3C 60 73 04   .j.........!<`s.
076A:0010  04 20 EB 02 2C 20 B4 02-8A D0 CD 21 E2 EA B4 4C   . .., .....!...L
076A:0020  CD 21 80 BF B8 2C 00 75-05 88 46 F8 EB 1E 8A 5E   .!...,.u..F....^
076A:0030  F9 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   .........;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:

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

Exp No: 9 **Date:** 19/10/2020

FLOATING POINT OPERATIONS

Name: Swetha Saseendran

Reg No: 185001183

Aim:

To write assembly language programs to perform the following floating-point arithmetic:

1. Floating point Addition.
2. Floating point Subtraction

Programs:

(i) FLOATING POINT ADDITION

Algorithm:

- Declare the data segment.
- Initialize data segment with the 2 floating point numbers and a variable for storing their sum.
- Close the data segment.

- Declare the code segment.
- Set a preferred offset (preferably 100h)
- Load the data segment content into AX register.
- Transfer the contents of AX register to DS register.
- Initialize Floating point operation using FINIT.
- Move the contents of the two numbers into the stack ST.
- Add them and store the value in top of the stack.
- Move the content in top of the stack to variable ‘sum’.
- Introduce an interrupt for safe exit. (INT 21h)
- Close the code segment.

PROGRAM	COMMENTS
assume *cs:code, ds:data*	Declare code and data segment.
data segment	Initialize data segment with values.
org 00h	Directive to assign an offset address for a variable.
x dd 20.4375	Stores the first number.
org 10h	
y dd 20.4375	Stores the second number.
org 20h	
sum dd ?	Variable to store the value of the sum.
data ends	End of data segment.
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.
finit	Initialize 8087's stack.
fld x	Load 'x' into ST(0).
fld y	Load 'y' into ST(0).
fadd ST(0), ST(1)	ST(0) = ST(0) + ST(1)
fst sum	Store the value of sum in the variable 'sum'.
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:

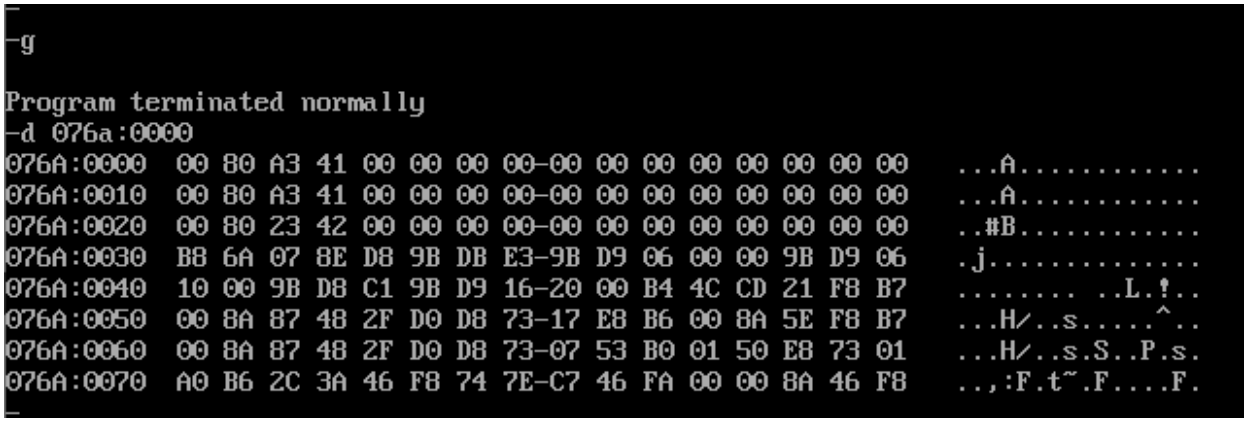
```
-u
076D:0000 B86A07        MOV     AX,076A
076D:0003 8ED8          MOV     DS,AX
076D:0005 9B            WAIT
076D:0006 DBE3                  FINIT
076D:0008 9B            WAIT
076D:0009 D9060000              FLD     DWORD PTR [0000]
076D:000D 9B            WAIT
076D:000E D9061000              FLD     DWORD PTR [0010]
076D:0012 9B            WAIT
076D:0013 D8C1                  FADD    ST,ST(1)
076D:0015 9B            WAIT
076D:0016 D9162000              FST     DWORD PTR [0020]
076D:001A B44C          MOV     AH,4C
076D:001C CD21          INT     21
076D:001E F8            CLC
076D:001F B700          MOV     BH,00
-
```

Snapshot of sample input and output:

INPUT:

```
-d 076a:0000
076A:0000  00 80 A3 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0010  00 80 A3 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  B8 6A 07 8E D8 9B DB E3-9B D9 06 00 00 9B D9 06   .j..............
076A:0040  10 00 9B D8 C1 9B D9 16-20 00 B4 4C CD 21 F8 B7   ........ ..L.!..
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:



(ii) FLOATING POINT SUBTRACTION:

Algorithm:

- Declare the data segment.
- Initialize data segment with the 2 floating point numbers and variables for storing their difference diff.
- Close the data segment.
- Declare the code segment.
- Set a preferred offset (preferably 100h)
- Load the data segment content into AX register.
- Transfer the contents of AX register to DS register.
- Initialize Floating point operation using FINIT.
- Move the contents of the two numbers into the stack ST.
- Subtract them and store the value in top of the stack.
- Move the content in top of the stack to variable ‘diff’.
- Introduce an interrupt for safe exit. (INT 21h)
- Close the code segment.

PROGRAM	COMMENTS
assume *cs:code, ds:data*	Declare code and data segment.
data segment	Initialize data segment with values.
org 00h	Directive to assign an offset address for a variable.
x dd 20.4375	Stores the first number.
org 10h	
y dd 20.4375	Stores the second number.
org 20h	
diff dd ?	Variable to store the value of the difference.
data ends	End of data segment.
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.
finit	Initialize 8087's stack.
fld x	Load 'x' into ST(0).
fld y	Load 'y' into ST(0).
fsub ST(0), ST(1)	ST(0) = ST(0) - ST(1)
fst diff	Store the value of sum in the variable 'diff'.
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:

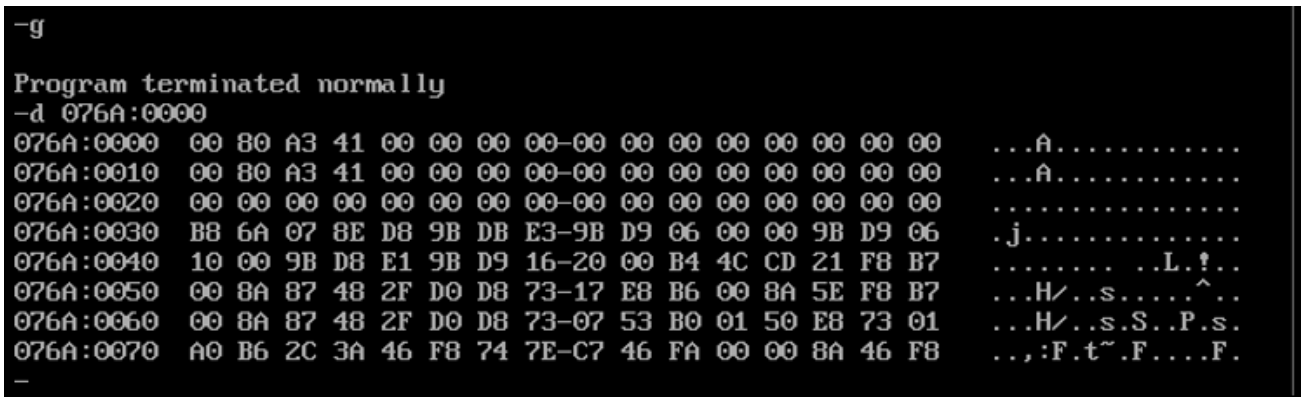
```
-u
076D:0000 B86A07        MOV     AX,076A
076D:0003 8ED8          MOV     DS,AX
076D:0005 9B            WAIT
076D:0006 DBE3                  FINIT
076D:0008 9B            WAIT
076D:0009 D9061000              FLD     DWORD PTR [0010]
076D:000D 9B            WAIT
076D:000E D9060000              FLD     DWORD PTR [0000]
076D:0012 9B            WAIT
076D:0013 D8E1                  FSUB    ST,ST(1)
076D:0015 9B            WAIT
076D:0016 D9162000              FST     DWORD PTR [0020]
076D:001A B44C          MOV     AH,4C
076D:001C CD21          INT     21
076D:001E F8            CLC
076D:001F B700          MOV     BH,00
-
```

Snapshot of sample input and output:

INPUT:

```
-d 076A:0000
076A:0000  00 80 A3 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0010  00 80 A3 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  B8 6A 07 8E D8 9B DB E3-9B D9 06 00 00 9B D9 06   .j..............
076A:0040  10 00 9B D8 E1 9B D9 16-20 00 B4 4C CD 21 F8 B7   ........ ..L.!..
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:



Result:

The assembly level programs were written to perform the above specified floating-point arithmetic operations and their output was verified.

Exp No: 10 **Date:** 19/10/2020

Name: Swetha Saseendran

Reg No: 185001183

DISPLAY A STRING

Aim:

To program and execute case conversions using 8086 microprocessor using DOSBOX.

Programs:

Algorithm:

- Initialise message in data segment.
- Use DOS function and offset of message to output the string.
- Terminate the program.

PROGRAM	*COMMENT*
DATA SEGMENT	INITIALISE THE DATA SEGMENT
MESSAGE DB "message$"	
DATA ENDS	
CODE SEGMENT	
ASSUME *CS:CODE,DS:DATA*	
START: *MOV AX,DATA*	
MOV DS,AX	DOS FUNCTION #9
MOV AH,9	
MOV DX,OFFSET MESSAGE	OFFSET OF THE STRING
INT 21H ;	DISPLAY IT
MOV AH,4CH	TERMINATE THE PROGRAM
INT 21H	
CODE ENDS	
END START	

Unassembled Code:

```
-u
076C:0000 B86A07        MOV     AX,076A
076C:0003 8ED8          MOV     DS,AX
076C:0005 B409          MOV     AH,09
076C:0007 BA0000        MOV     DX,0000
076C:000A CD21          INT     21
076C:000C B44C          MOV     AH,4C
076C:000E CD21          INT     21
076C:0010 F9            STC
076C:0011 B700          MOV     BH,00
076C:0013 D1E3          SHL     BX,1
076C:0015 8B87AE16      MOV     AX,[BX+16AE]
076C:0019 3B46FE        CMP     AX,[BP-02]
076C:001C 7709          JA      0027
076C:001E 8946FE        MOV     [BP-02],AX
-
```

Snapshot of sample input and output:

INPUT:

```
-d 076a:0000
076A:0000  54 48 49 53 20 49 53 20-54 48 45 20 53 54 52 49   THIS IS THE STRI
076A:0010  4E 47 24 00 00 00 00 00-00 00 00 00 00 00 00 00   NG$.............
076A:0020  B8 6A 07 8E D8 B4 09 BA-00 00 CD 21 B4 4C CD 21   .j.........!.L.!
076A:0030  F9 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   .........;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
THIS IS THE STRING
Program terminated normally
-d 076a:0000
076A:0000  54 48 49 53 20 49 53 20-54 48 45 20 53 54 52 49   THIS IS THE STRI
076A:0010  4E 47 24 00 00 00 00 00-00 00 00 00 00 00 00 00   NG$.............
076A:0020  B8 6A 07 8E D8 B4 09 BA-00 00 CD 21 B4 4C CD 21   .j.........!.L.!
076A:0030  F9 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   .........;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:

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

Exp No: 11 **Date:** 19/10/2020

Name: Swetha Saseendran

Reg No: 185001183

DISPLAY SYSTEM DATE AND TIME

Aim:

To write assembly language programs to perform the following system operations:

1. Display System Date
2. Display System Time

Programs:

(i) SYSTEM DATE

Algorithm:

- Declare the data segment.
- Initialize data segment with variables to store day, month and year.
- Close the data segment.

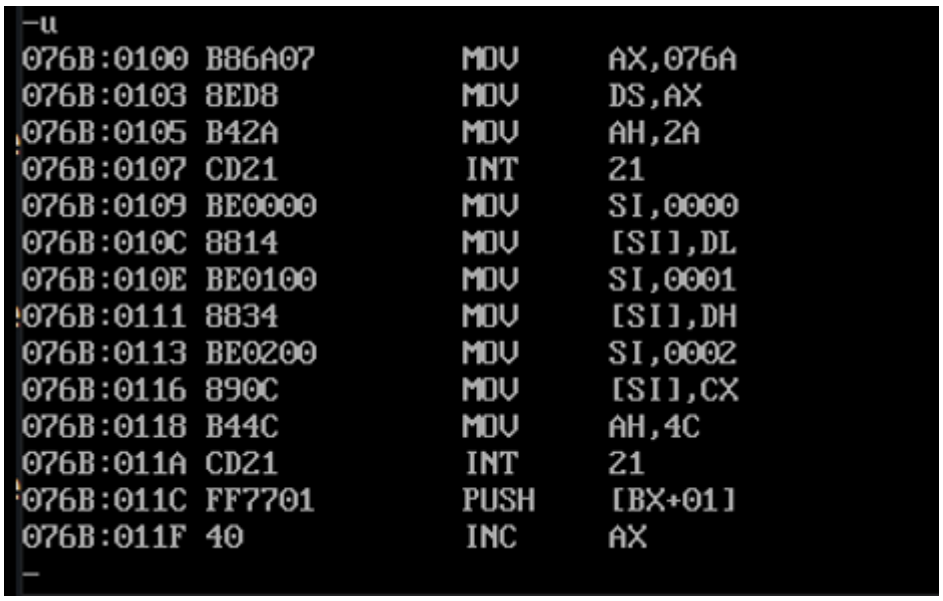
- Declare the code segment.
- Set a preferred offset (preferably 100h)
- Load the data segment content into AX register.
- Transfer the contents of AX register to DS register.
- Load 2Ah to AH register. (DOS function to obtain system date)
- Call interrupt 21h to service the DOS function.
- Load the offset address of variable ‘day’ to SI.
- Transfer contents of DL register through SI to variable ‘day’.
- Load the offset address of variable ‘month’ to SI.

- Transfer contents of DH register through SI to variable ‘month’.
- Load the offset address of variable ‘year’ to SI.
- Transfer contents of CX register through SI to variable ‘year’.
- Introduce an interrupt for safe exit. (INT 21h)
- Close the code segment.

PROGRAM	*COMMENTS*
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
day db 01 dup(?)	Variable to store day.
month db 01 dup(?)	Variable to store month.
year db 02 dup(?)	Variable to store year.
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 ah, 2Ah	Load 2Ah to AH (DOS code for system date function)
int 21h	Interrupt DOS with 21h to get the system date.
mov si, offset day	Load offset of variable ‘day’ to SI.
mov [si], dl	Copy to ‘day’ the value of DL through SI.
mov si, offset month	Load offset of variable ‘month’ to SI.
mov [si], dh	Copy to ‘month’ the value of DH through SI.
mov si, offset year	Load offset of variable ‘year’ to SI.
mov [si], cx	Copy to ‘year’ the value of CX through SI.
mov ah, 4ch	Moves the hexadecimal value 4c to ah.

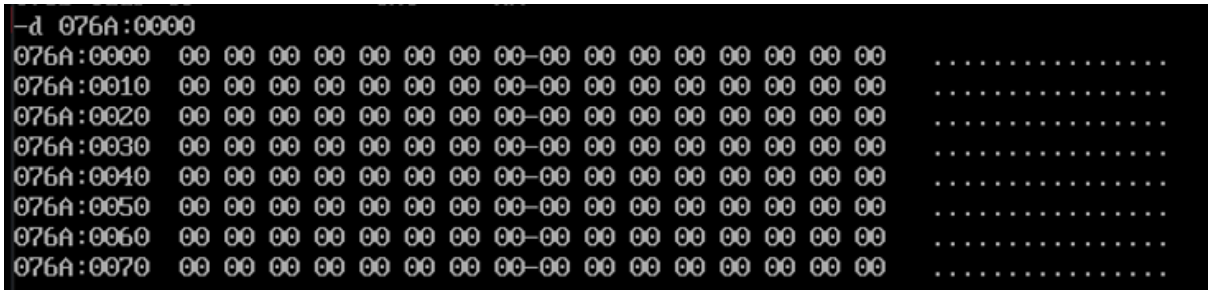
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).
code ends *end start*	

Unassembled Code:

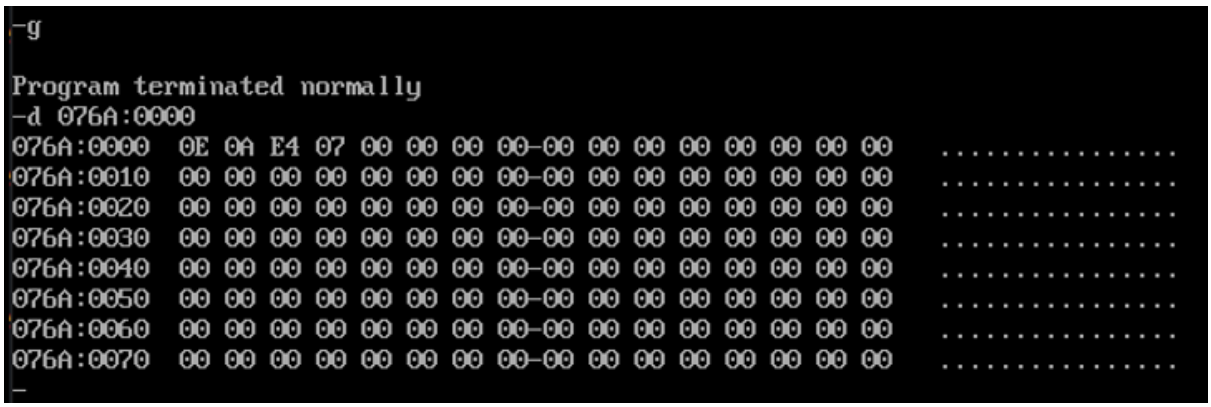


Snapshot of sample input and output:

INPUT:



OUTPUT:



(ii) SYSTEM TIME

Algorithm:

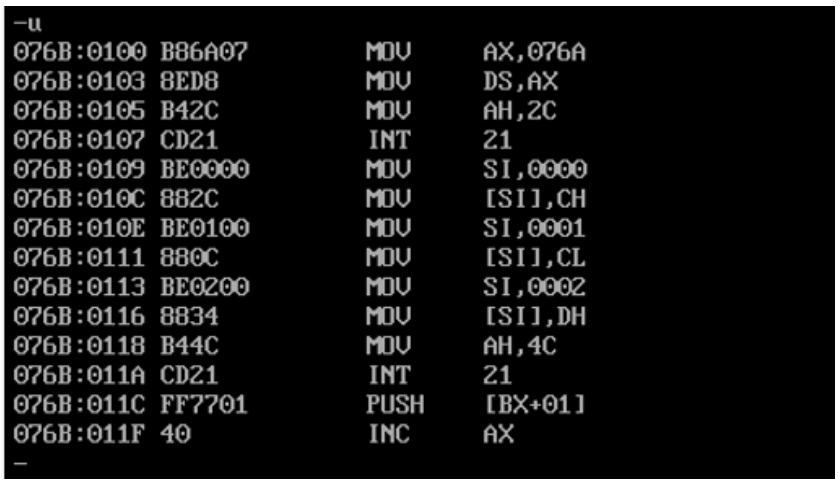
- Declare the data segment.
- Initialize data segment with variables to store hour, minute and second.
- Close the data segment.

- Declare the code segment.
- Set a preferred offset (preferably 100h)
- Load the data segment content into AX register.
- Transfer the contents of AX register to DS register.
- Load 2Ch to AH register. (DOS function to obtain system time)
- Call interrupt 21h to service the DOS function.
- Load the offset address of variable 'hour' to SI.
- Transfer contents of CH register through SI to variable 'hour'.
- Load the offset address of variable 'minute' to SI.
- Transfer contents of CL register through SI to variable 'minute'.
- Load the offset address of variable 'second' to SI.
- Transfer contents of DH register through SI to variable 'second'.
- Introduce an interrupt for safe exit. (INT 21h)
- Close the code segment.

PROGRAM	*COMMENTS*
assume *cs:code, ds:data*	Declare code and data segment.
data segment	Initialize data segment with values.
hour db 01 dup(?)	Variable to store hour.
minute db 01 dup(?)	Variable to store minute.

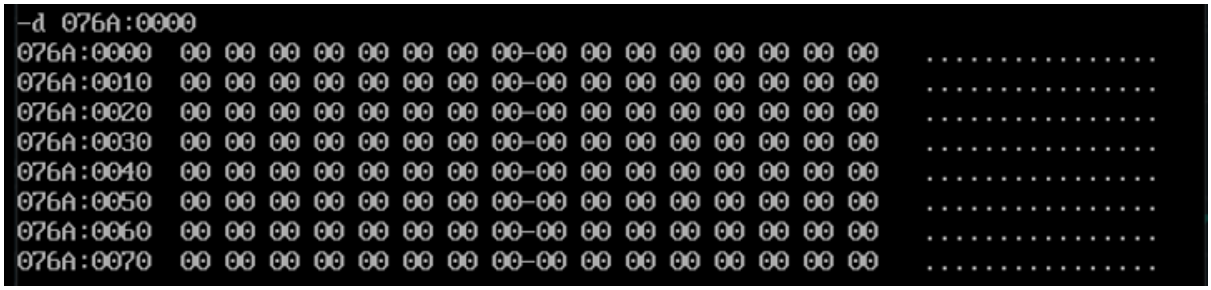
second db 02 dup(?)	Variable to store second.
data ends	
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: *mov ax, data*	Transfer data from “data” to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov ah, 2Ch	Load 2Ch to AH (DOS code for system time function)
int 21h	Interrupt DOS with 21h to get the system time.
mov si, offset hour	Load offset of variable ‘hour’ to SI.
mov [si], ch	Copy to ‘hour’ the value of CH through SI.
mov si, offset minute	Load offset of variable ‘minute’ to SI.
mov [si], cl	Copy to ‘minute’ the value of CL through SI.
mov si, offset second	Load offset of variable ‘second’ to SI.
mov [si], dh	Copy to ‘second’ the value of DH through SI.
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:

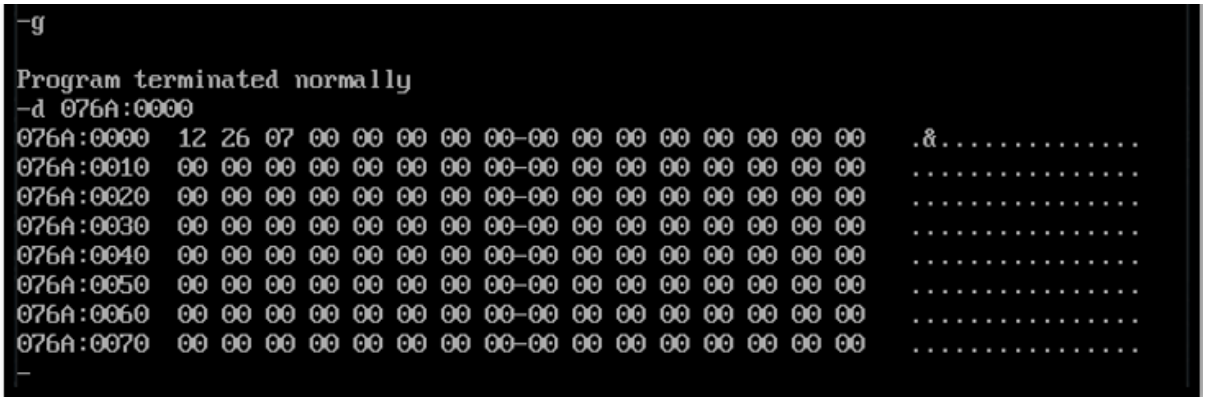


Snapshot of sample input and output:

INPUT:



OUTPUT:



Result:

The assembly level programs were written to perform the above specified system operations, namely, system date and system time and the output was verified.