CASE CONVERSION

Exp No: 08 Name: Srinithyee S K

Date: 16-10-20 Register Number: 185001166

AIM:

To write assembly language programs to perform alphabetical case conversion on the fly from standard input to standard output.

ALGORITHM:

1. Begin.
2. Declare the data segment.
3. Initialize data segment with the count (number of input characters)
4. Close the data segment.
5. Declare the code segment.
6. Load the data segment content into AX register.
7. Transfer the contents of AX register to DS register.
8. Move the count into CX register.
9. Loop through count C1:
    I. Move 01h to AH, to input a character.
    II. Interrupt to get input.
    III. If input > 60
        i. Subtract the ASCII value by 20h.
    IV. Else
        i. Add the ASCII value by 20h.
    V. Print the output through DOS's standard output by moving it into DL register.
10. Introduce an interrupt for safe exit. (INT 21h)
11. Close the code segment.
12. End.

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
count equ 10h	Number of input characters to be taken.
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 cx, count	Loads the value in count to CX register.
L1: mov ah, 1	To input a character.
int 21h	
	ASCII (hex): A-Z= 41-5A, a-z= 61-7A.
cmp al, 60h	If AL > 60, then jump to ‘upper’.
jnc upper	
add al, 20h	To convert the character to lowercase.
jmp skip	
upper: sub al, 20h	To convert the character to uppercase.
skip: mov ah, 2	To output a character.
mov dl,al	Transfers the contents in AL to DL to support printing.
int 21h	
loop L1	Loops till CX = 0.
mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
   Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG CASECNV.EXE
-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
-
```

SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG CASECNV.EXE
-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
-g
aABbcCDdEefFGghHIiJjkKlLMmnNOopP
Program terminated normally
-
```

RESULT:

The assembly level program was written to perform the above specified case conversion and the output was verified.

FLOATING POINT OPERATIONS

Exp No: 09 Name: Srinithyee S K

Date: 16-10-20 Register Number: 185001166

AIM:

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

1. Floating point Addition.
2. Floating point Subtract

PROGRAM 1: FLOATING POINT ADDITION

ALGORITHM:

1. Begin.
2. Declare the data segment.
3. Initialize data segment with the 2 floating point numbers and a variable for storing their sum.
4. Close the data segment.
5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Initialize Floating point operation using FINIT.
10. Move the contents of the two numbers into the stack ST.
11. Add them and store the value in top of the stack.
12. Move the content in top of the stack to variable 'sum'.
13. Introduce an interrupt for safe exit. (INT 21h)
14. Close the code segment.
15. End.

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	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG FLTADD.EXE
-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
-_
```

SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
076D:001C CD21          INT     21
076D:001E F8            CLC
076D:001F B700          MOV     BH,00
-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.
-g

Program terminated normally
-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 80 23 42 00 00 00 00-00 00 00 00 00 00 00 00   ..#B............
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.
-
```

PROGRAM – 2: FLOATING POINT SUBTRACTION:

ALGORITHM:

1. Begin.
2. Declare the data segment.
3. Initialize data segment with the 2 floating point numbers and variables for storing their difference diff.
4. Close the data segment.
5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Initialize Floating point operation using FINIT.
10. Move the contents of the two numbers into the stack ST.
11. Subtract them and store the value in top of the stack.
12. Move the content in top of the stack to variable 'diff'.
13. Introduce an interrupt for safe exit. (INT 21h)
14. Close the code segment.
15. End.

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	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG FLTSUB.EXE
-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 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
-
```

SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
076D:001C CD21          INT     21
076D:001E F8            CLC
076D:001F B700          MOV     BH,00
-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 DB 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.
-g

Program terminated normally
-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 DB 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.
-
```

RESULT:

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

DISPLAY A STRING

Exp No: 10 Name: Srinithyee S K

Date: 16-10-20 Register Number: 185001166

AIM:

To write an assembly language program to display a string through the standard output.

PROGRAM – 1: DISPLAY A STRING

ALGORITHM:

1. Begin.
2. Declare the data segment.
3. Initialize data segment with a variable for a string “Hello World!”
4. Close the data segment.
5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Load 09h into AH register (DOS function to write to standard output)
9. Store the offset of the string in DX register.
10. Introduce an interrupt for safe exit. (INT 21h)
11. Close the code segment.
12. End.

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
message db “Hello World!$”	Variable message has “Hello World!” as a string.
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	Move contents of AX to DS.
mov ah, 9	AH = 09h for DOS function to write to STDOUT.
mov dx, offset message	Load offset address of message to DX.
mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
Q:\>LINK DISPSTR.OBJ;

   Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG DISPSTR.EXE
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 B409          MOV     AH,09
076B:0007 BA0000        MOV     DX,0000
076B:000A CD21          INT     21
076B:000C B44C          MOV     AH,4C
076B:000E CD21          INT     21
076B:0010 89C3          MOV     BX,AX
076B:0012 80BFB82C00    CMP     BYTE PTR [BX+2CB8],00
076B:0017 7505          JNZ     001E
076B:0019 8846F8        MOV     [BP-08],AL
076B:001C EB1E          JMP     003C
076B:001E 8A5EF9        MOV     BL,[BP-07]
-
```

SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
076B:0003 8ED8          MOV     DS,AX
076B:0005 B409          MOV     AH,09
076B:0007 BA0000        MOV     DX,0000
076B:000A CD21          INT     21
076B:000C B44C          MOV     AH,4C
076B:000E CD21          INT     21
076B:0010 89C3          MOV     BX,AX
076B:0012 80BFB82C00    CMP     BYTE PTR [BX+2CB8],00
076B:0017 7505          JNZ     001E
076B:0019 8846F8        MOV     [BP-08],AL
076B:001C EB1E          JMP     003C
076B:001E 8A5EF9        MOV     BL,[BP-07]
-d 076A:0000
076A:0000  48 65 6C 6C 6F 20 57 6F-72 6C 64 21 24 00 00 00   Hello World!$...
076A:0010  B8 6A 07 8E D8 B4 09 BA-00 00 CD 21 B4 4C CD 21   .j.........!.L.!
076A:0020  89 C3 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.
-g
Hello World!
Program terminated normally
-
```

RESULT:

The assembly level program was written to display a string and the output was verified.

DISPLAY A SYSTEM DATE AND TIME

Exp No: 11 Name: Srinithyee S K

Date: 16-10-20 Register Number: 185001166

AIM:

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

1. Display System Date
2. Display System Time

PROGRAM -1 : SYSTEM DATE

ALGORITHM:

1. Begin.
2. Declare the data segment.
3. Initialize data segment with variables to store day, month and year.
4. Close the data segment.
5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Load 2Ah to AH register. (DOS function to obtain system date)
10. Call interrupt 21h to service the DOS function.
11. Load the offset address of variable ‘day’ to SI.
12. Transfer contents of DL register through SI to variable ‘day’.
13. Load the offset address of variable ‘month’ to SI.
14. Transfer contents of DH register through SI to variable ‘month’.
15. Load the offset address of variable ‘year’ to SI.
16. Transfer contents of CX register through SI to variable ‘year’.
17. Introduce an interrupt for safe exit. (INT 21h)
18. Close the code segment.
19. End.

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	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:

```
DOSBox 0.74-3, Cpu speed:    3000 cycles, Frameskip  0, Progra...

  Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG SYSDATE.EXE
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 B42A          MOV     AH,2A
076B:0107 CD21          INT     21
076B:0109 BE0000        MOV     SI,0000
076B:010C 8814          MOV     [SI],DL
076B:010E BE0100        MOV     SI,0001
076B:0111 8834          MOV     [SI],DH
076B:0113 BE0200        MOV     SI,0002
076B:0116 890C          MOV     [SI],CX
076B:0118 B44C          MOV     AH,4C
076B:011A CD21          INT     21
076B:011C FF7701        PUSH    [BX+01]
076B:011F 40            INC     AX
-
```

SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed:    3000 cycles, Frameskip  0, Progra...
076B:011A CD21          INT     21
076B:011C FF7701        PUSH    [BX+01]
076B:011F 40            INC     AX
-d 076A:0000
076A:0000  00 00 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   ................
-g

Program terminated normally
-d 076A:0000
076A:0000  0E 0A E4 07 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   ................
-
```

PROGRAM -2 : SYSTEM TIME

ALGORITHM:

1. Begin.
2. Declare the data segment.
3. Initialize data segment with variables to store hour, minute and second.
4. Close the data segment.
5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Load 2Ch to AH register. (DOS function to obtain system time)
10. Call interrupt 21h to service the DOS function.
11. Load the offset address of variable ‘hour’ to SI.
12. Transfer contents of CH register through SI to variable ‘hour’.
13. Load the offset address of variable ‘minute’ to SI.
14. Transfer contents of CL register through SI to variable ‘minute’.
15. Load the offset address of variable ‘second’ to SI.
16. Transfer contents of DH register through SI to variable ‘second’.
17. Introduce an interrupt for safe exit. (INT 21h)
18. Close the code segment.
19. End.

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	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...

   Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG SYSTIME.EXE
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 B42C          MOV     AH,2C
076B:0107 CD21          INT     21
076B:0109 BE0000        MOV     SI,0000
076B:010C 882C          MOV     [SI],CH
076B:010E BE0100        MOV     SI,0001
076B:0111 880C          MOV     [SI],CL
076B:0113 BE0200        MOV     SI,0002
076B:0116 8834          MOV     [SI],DH
076B:0118 B44C          MOV     AH,4C
076B:011A CD21          INT     21
076B:011C FF7701        PUSH    [BX+01]
076B:011F 40            INC     AX
-
```

SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
076B:011A CD21          INT     21
076B:011C FF7701        PUSH    [BX+01]
076B:011F 40            INC     AX
-d 076A:0000
076A:0000  00 00 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   ................
-g

Program terminated normally
-d 076A:0000
076A:0000  12 26 07 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   ................
-
```

RESULT:

The assembly level programs were written to perform the system operations: system date and system time and the output was verified.