

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
<i>assume cs:code, ds:data</i>	Declare code and data segment.
<i>data segment</i>	Initialize data segment with values.
<i>day db 01</i>	Variable to store day.
<i>dup(?)</i>	
<i>month db 01</i>	Variable to store month.
<i>dup(?)</i>	
<i>year db 02</i>	Variable to store year.
<i>dup(?)</i>	
<i>data ends</i>	
<i>code segment</i>	Start the code segment.
<i>org 0100h</i>	Initialize an offset address.
<i>start: mov ax, data</i>	Transfer data from "data" to AX.
<i>mov ds, ax</i>	Transfer data from memory location AX to DS.
<i>mov ah, 2Ah</i>	Load 2Ah to AH (DOS code for system date function)
<i>int 21h</i>	Interrupt DOS with 21h to get the system date.
<i>mov si, offset day</i>	Load offset of variable 'day' to SI.
<i>mov [si], dl</i>	Copy to 'day' the value of DL through SI.
<i>mov si, offset month</i>	Load offset of variable 'month' to SI.
<i>mov [si], dh</i>	Copy to 'month' the value of DH through SI.
<i>mov si, offset year</i>	Load offset of variable 'year' to SI.
<i>mov [si], cx</i>	Copy to 'year' the value of CX through SI.
<i>mov ah, 4ch</i>	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 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
-
```

### Snapshot of sample input and output:

INPUT:

```
-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  .....
```

OUTPUT:

```
-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 .....
```

(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
<i>assume cs:code, ds:data</i>	Declare code and data segment.
<i>data segment</i>	Initialize data segment with values.
<i>hour db 01 dup(?)</i>	Variable to store hour.
<i>minute db 01 dup(?)</i>	Variable to store minute.

<i>second</i>	<i>db</i>	<i>02</i>	Variable to store second.
<i>dup(?)</i>			
<b>data ends</b>			
<b>code segment</b>			Start the code segment.
<i>org</i>	<i>0100h</i>		Initialize an offset address.
<b>start:</b>	<i>mov</i>	<i>ax, data</i>	Transfer data from “data” to AX.
	<i>mov</i>	<i>ds, ax</i>	Transfer data from memory location AX to DS.
	<i>mov</i>	<i>ah, 2Ch</i>	Load 2Ch to AH (DOS code for system time function)
	<i>int</i>	<i>21h</i>	Interrupt DOS with 21h to get the system time.
	<i>mov</i>	<i>si, offset hour</i>	Load offset of variable ‘hour’ to SI.
	<i>mov</i>	<i>[si], ch</i>	Copy to ‘hour’ the value of CH through SI.
	<i>mov</i>	<i>si, offset minute</i>	Load offset of variable ‘minute’ to SI.
	<i>mov</i>	<i>[si], cl</i>	Copy to ‘minute’ the value of CL through SI.
	<i>mov</i>	<i>si, offset second</i>	Load offset of variable ‘second’ to SI.
	<i>mov</i>	<i>[si], dh</i>	Copy to ‘second’ the value of DH through SI.
	<i>mov</i>	<i>ah, 4ch</i>	Moves the hexadecimal value 4c to ah.
	<i>int</i>	<i>21h</i>	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).
<b>code ends</b>			
<b>end start</b>			

## Unassembled Code:

```

-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
-

```

Snapshot of sample input and output:

INPUT:

```
-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 .....
```

OUTPUT:

```
-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 above specified system operations, namely, system date and system time and the output was verified.