# Experiment No 8: Case Conversion

**Date: 1-11-2020** **NAME: A Susmithaa Raam**
**REG.NO: 185001181**

## 1 AIM:

To write and execute 8086 programs to convert case characters dynamically (on the fly)

## 2 Algorithm & Program

**INITIALIZATION:**

- Declare and initialize the operands and the code and data segments.

### 3.1 Case conversion:

To convert case characters on the fly.

1. Initialize data and code segment and variables
2. Move the starting address of data segment to DS
3. Transfer the contents of COUNT to CX register
4. **JUMP** L1 block
   - Move 1 to AH and compare value in AL register with 60
   - If no carry is generated on comparison jump to upper block else ADD 20H to AL and jump to skip
5. **JUMP** Upper block
   - Convert lower case to upper case by subtracting 20H from AL
6. **JUMP** Skip block
   - Move contents of AL to DL register and display the character
   - Loop L1 until termination
7. Terminate the program

#### 3.1.1 Case conversion

| Code | Comment |
|---|---|
| ASSUME CS:CODE,DS:data | |
| data SEGMENT<br>COUNT equ10h<br>data ends | Initialize data segment<br>and variables<br>End data segment |
| CODE SEGMENT<br>START:<br>MOV AX,data<br>MOV DS,AX<br>MOV CX, COUNT ; | Initialize code segment<br><br>Transfer address of data segment to<br>DS and COUNT to CX register<br>Loop Counter |

| | |
|---|---|
| L1: | L1 block |
| MOV AH,1 ; | Input character, |
| INT 21H ; | AL = character, ASCII(hex) : |
| CMP AL,60H | A-Z=41-5A, a-z=61-7A |
| JNC UPPER | Compare AL and 60 to check for upper case |
| ADD AL,20H | If no carry jump to Upper |
| JMP SKIP | |
| UPPER: | Upper block: |
| SUB AL,20H; | Convert to upper case |
| SKIP: | Skip Block |
| MOV AH, 2 ; | Character output function |
| MOV DL,AL ; | Transfer the contents to AL to DL register |
| INT 21H ; | Display the character |
| LOOP L1 ; | Repeat Loop |
| MOV Ah ,4CH | |
| INT 21H | Termination of execution |
| CODE ENDS | End of the code segment |
| end start | Terminate program |

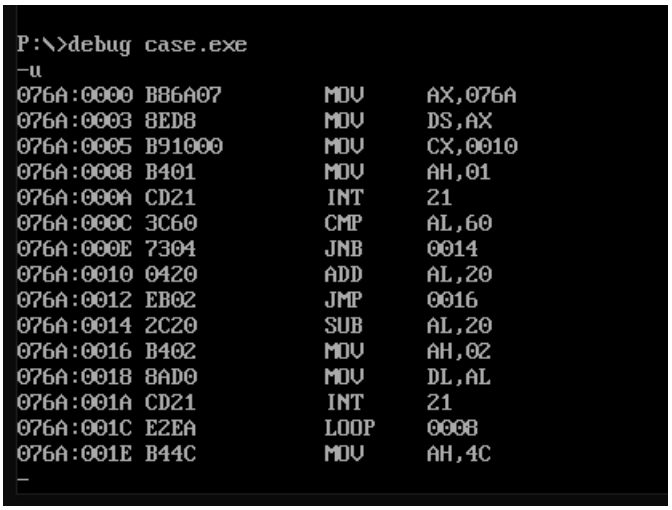


Figure 1: Case conversion - unassembled

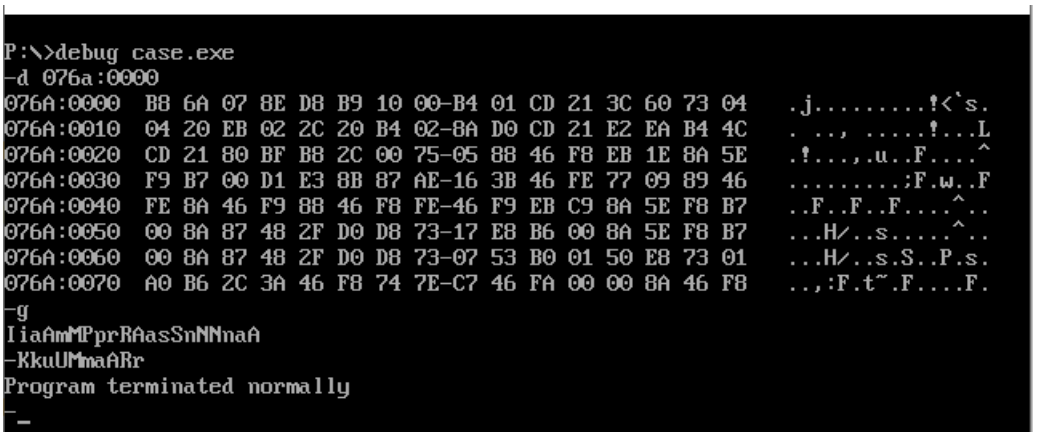


Figure 2: Case conversion - Output

## RESULT:

Thus, 8086 programs to convert case characters was implemented.

# Experiment No 9:
## Floating point operations

---

# 1 AIM:

To write and execute 8086 programs to perform arithmetic operations on floating point numbers

# 2 Algorithm & Program

**INITIALIZATION:**

- Declare and initialize the operands and the code and data segments.

## 3.1 Floating point Addition:

To perform addition on floating point numbers

1. Initialize data and code segment and operands
2. Move the starting address of data segment to DS
3. Initialize the 8087 stack using **FINIT** command
4. Using **FLD** load X to ST(0) and Y into ST(1)
5. Add the floating point numbers using **FADD** (ST(0) = X + Y)
6. Store ST(0) to SUM variable using **FST** command
7. Terminate the program

### 3.1.1 Floating point addition

| Code | Comment |
|---|---|
| ASSUME CS:CODESEG, DS:DATASEG | |
| DATASEG SEGMENT ;<br>ORG 00H<br>X DD 20.4375<br>ORG 10H<br>Y DD 20.4375<br>ORG 20H<br>SUM DD?<br>DATASEG ENDS ; | Initialize data segment and variables<br><br><br><br><br><br><br>End data segment |
| CODESEG SEGMENT<br>START:<br>MOV AX,DATASEG<br>MOV DS,AX ; assign value to DS<br>FINIT ;<br>FLD X ;<br>FLD Y ;<br>FADD ST(0),ST(1) ;<br>FST SUM ;<br>MOV AH,4CH;<br>INT 21H;<br>CODESEG ENDS ;<br>END START | Initialize code segment<br><br>Transfer address of data segment to<br>DS register<br>Initialize 8087 stack<br>load X into ST(0)<br>load Y into ST(1)<br>ST(0) = X + Y<br>Store ST(0) to sum<br><br>Termination of execution<br>End of the code segment<br>Terminate program |

```
P:\>debug floatadd.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
-
```

Figure 1: Floating point addition - unassembled

```
-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.
-
```

Figure 2: Floating point addition - Output

## 3.2 Floating point Subtraction:

To perform subtraction on floating point numbers

1. Initialize data and code segment and operands
2. Move the starting address of data segment to DS
3. Initialize the 8087 stack using **FINIT** command
4. Using **FLD** load X to ST(0) and Y into ST(1)
5. Subtract the floating point numbers using **FSUB** (ST(0) = X - Y)
6. Store ST(0) to SUM variable using **FST** command
7. Terminate the program

### 3.2.1 Floating point subtraction

| Code | Comment |
|---|---|
| ASSUME CS:CODESEG, DS:DATASEG | |
| DATASEG SEGMENT ;<br>ORG 00H<br>X DD 20.4375<br>ORG 10H<br>Y DD 0.125<br>ORG 20H<br>SUM DD?<br>DATASEG ENDS ; | Initialize data segment and variables<br><br><br><br><br><br><br>End data segment |
| CODESEG SEGMENT<br>START:<br>MOV AX,DATASEG<br>MOV DS,AX<br>FINIT ;<br>FLD X ;<br>FLD Y ;<br>FSUB ST(0),ST(1) ;<br>FST SUM ;<br>MOV AH,4CH;<br>INT 21H;<br>CODESEG ENDS ;<br>END START | Initialize code segment<br><br>Transfer address of data segment to<br>DS register<br>Initialize 8087 stack<br>load X into ST(0)<br>load Y into ST(1)<br>ST(0) = X - Y<br>Store ST(0) to sum<br><br>Termination of execution<br>End of the code segment<br>Terminate program |

```
P:\>debug floatsub.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 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
-
```

Figure 3: Floating point subtraction - unassembled

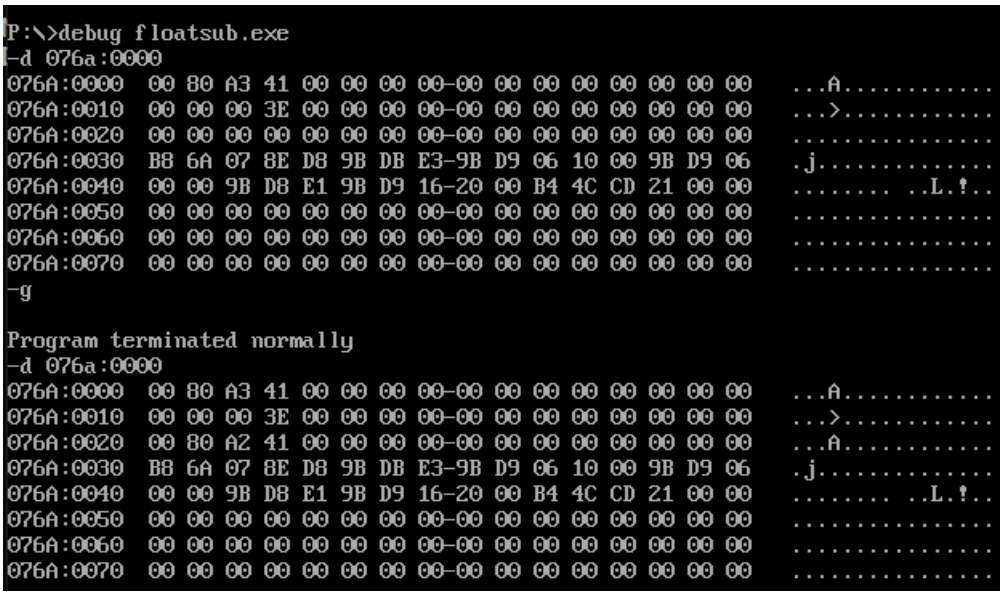


Figure 4: Floating point subtraction - Output

## RESULT:

Thus, 8086 programs to perform arithmetic operation on floating point numbers was implemented.

# Experiment No 9:
## Display a string

## 1 AIM:

To write and execute 8086 programs to display a string in standard output device

## 2 Algorithm & Program

**INITIALIZATION:**

- Declare and initialize the operands and the code and data segments.

### 3.1 Display a string:

To display a string in standard output device

1. Initialize data and code segment and the message variable
2. Move the starting address of data segment to DS register
3. Initialize AH register to 9 (DOS function 9)
4. Move the offset of the message variable to the DX register
5. Display the message
6. Terminate the program

#### 3.1.1 Display a string

| Code | Comment |
|---|---|
| DATA SEGMENT<br>MESSAGE DB ”THIS IS THE STRING$”<br><br>DATA ENDS | Initialize data segment and store the message in a variable<br><br>End data segment |
| CODE SEGMENT<br>ASSUME CS:CODE,DS:DATA<br>START:<br>MOV AX,DATA<br>MOV DS,AX<br>MOV AH,9<br>MOV DX,OFFSET MESSAGE<br>INT 21H<br>MOV AH, 4CH<br>INT 21H<br>CODE ENDS<br>END START | Initialize code segment<br><br>Transfer address of data segment to DS register<br>Initialize AH to 9 (Dos function #9)<br><br>Move the offset of the message variable to the DX register<br>Display the message<br><br>End of the code segment<br>Terminate program |

```
P:\>debug displa.exe
-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
-
```

Figure 1: Display a string - unassembled

```
-d 076a:000
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.
-g
THIS IS THE STRING
Program terminated normally
-d 076a:0000
```

Figure 2: Display a string - Output

## RESULT:

Thus, 8086 programs to display a string on the standard output device was implemented.

# Experiment No 10:
## Display system date and time

## 1 AIM:

To write and execute 8086 programs to display system date and time in standard output device

## 2 Algorithm & Program

**INITIALIZATION:**

- Declare and initialize the operands and the code and data segments.

### 3.1 Display system date:

To display system date in standard output device

1. Initialize data and code segment and the day, month and year variable
2. Move the starting address of data segment to **DS** register
3. Move 2AH to AH to get system date
4. Move the offset of day to SI and copy the contents of **DL**(day value) to top of the SI stack
5. Move the offset of month to SI and copy the contents of **DH**(month value) to top of the SI stack
6. Move the offset of year to SI and copy the contents of **CX**(year value) to top of the SI stack
7. Display the date
8. Terminate the program

#### 3.1.1 Display system date

| Code | Comment |
|---|---|
| assume cs :code,ds :data | |
| data segment<br>day db 01 dup(?)<br>month db 01 dup(?)<br>year db 02 dup(?)<br>data ends | Initialize data segment and<br>declare day, month and year array<br><br><br><br>End data segment |

| | |
|---|---|
| code segment<br>org 0100h<br>start:<br>mov ax,data<br>mov ds,ax<br>mov ah,2ah<br>int 21h<br>mov si,offset day<br>mov [si],dl<br>mov si,offset month<br>mov [si],dh<br>mov si,offset year<br>mov [si],cx<br>mov ah,4ch<br>int 21h<br>code ends<br>end start | Initialize code segment<br><br>Transfer address of data segment to DS register<br><br>INT 21h /AH=2Ah - get system date;<br>Store the offset of day to SI register and<br>DL stores the day value<br>Move the offset of the month variable to the top of the current stack<br>and DH holds the value of the month<br>Move the offset of year to the top of the current stack<br>and CX holds the value of the year<br><br>Termination of execution<br>End of the code segment<br>Terminate program |

```
P:\>debug date.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
```

Figure 1: Display system date - unassembled

```
-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  10 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   ................
```

Figure 2: Display system date - Output

## 3.2 Display system time:

To display system time on the standard output device

1. Initialize data and code segment and the hour, minute and seconds variable
2. Move the starting address of data segment to **DS** register
3. Move 2CH to AH to get system time
4. Move the offset of day to the current SI location and copy the contents of **CH**(hour value) to top of the SI stack
5. Move the offset of month to SI and copy the contents of **CL**(minute value) to top of the SI stack
6. Move the offset of year to SI and copy the contents of **DH**(seconds value) to top of the SI stack
7. Display the time
8. Terminate the program

### 3.2.1 Display system time

| Code | Comment |
|---|---|
| assume cs :code,ds :data | |
| data segment<br>hour db 01 dup(?)<br>minute db 01 dup(?)<br>second db 02 dup(?)<br>data ends | Initialize data segment and<br>declare hour, minute and second array<br><br><br><br>End data segment |
| code segment<br>org 0100h<br>start:<br>mov ax,data<br>mov ds,ax<br>mov ah,2ch<br>int 21h<br>mov si,offset hour<br>mov [si],ch<br>mov si,offset minute<br>mov [si],cl<br>mov si,offset second<br>mov [si],dh<br>mov ah,4ch<br>int 21h<br>code ends<br>end start | Initialize code segment<br><br><br><br>Transfer address of data segment to DS register<br><br>INT 21h/AH=2Ch- get system time<br>Note : (CH= hour. CL= minute. DH= second)<br>Move the offset value of hour to the current address of SI register<br>and move the contents of CH(Hour value) to the top of the stack<br>Move the offset value of minute to the current address of SI register<br>and move the contents of CL(minute value) to the top of the stack<br>Move the offset value of second to the current address of SI register<br>and move the contents of DH(second value) to the top of the stack<br><br>Termination of execution<br>End of the code segment<br>Terminate program |

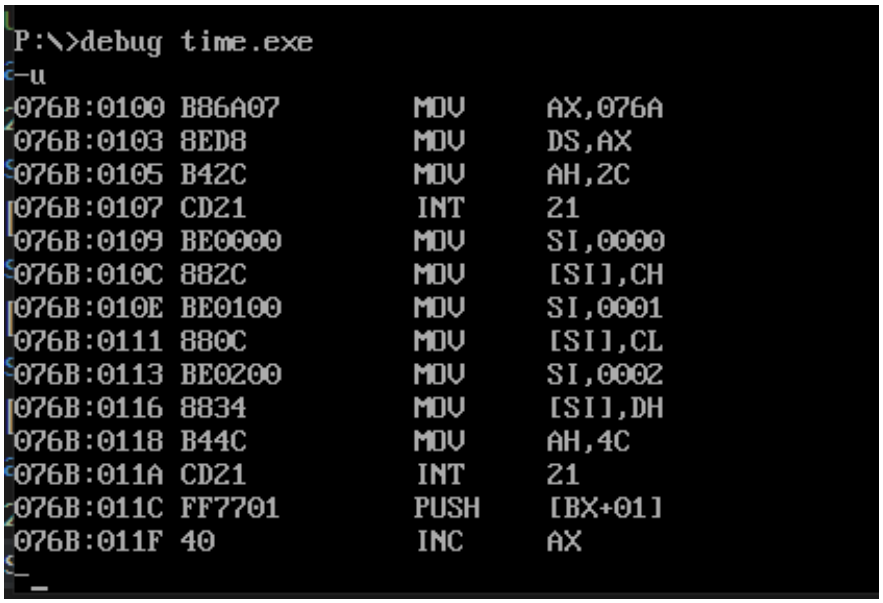


Figure 3: Display system time - unassembled

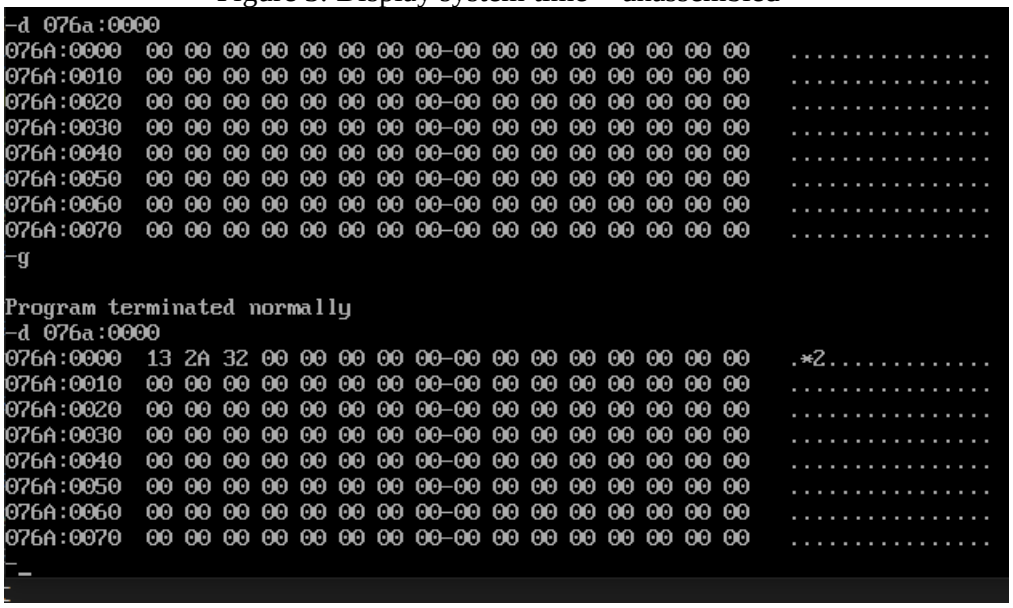


Figure 4: Display system time - Output

## RESULT:

Thus, 8086 programs to display the system date and time on the standard output device was implemented.