# UCS1512 – Microprocessors Lab

# Case Conversion

Exp no : 8 Name: Sreedhar V

Date : 25-10-2020 Reg no: 185001161

# AIM:

To program and execute the program for inverting the case of the letter on the fly in 8086 using an emulator.

# Case Conversion:

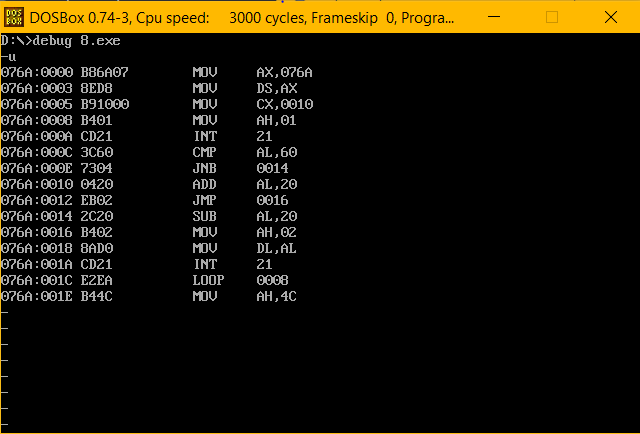
## Algorithm:

* + Program is set to run from any specified memory position.
  + Move the address of data segment to register DS .
  + Set the value of the count to be 10 using equ directive(for 10 counts).
  + Transfer the value of count to CX register.
  + Read the input by setting AH to 1 and executing INT 21h
  + Check whether the read character is lowercase or uppercase.
  + If it’s in uppercase, convert to lowercase by adding 20h
  + If it’s in lowercase, convert to uppercase by subtracting 20h
  + Repeat these steps till CX becomes 0
  + Terminate the program.

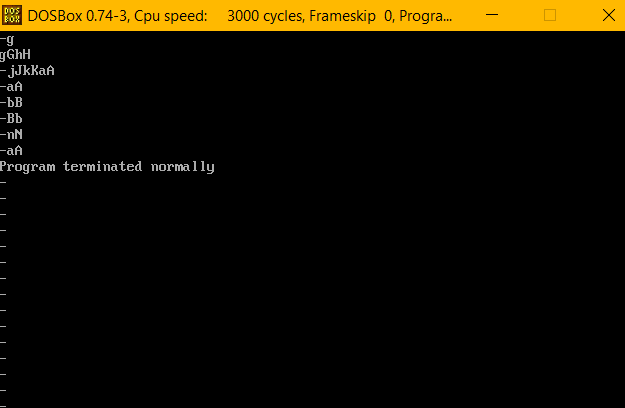
## Program:

|  |  |
| --- | --- |
| CODE | COMMENT |
| Program for Case conversion:  assume cs:code,ds:data  data segment  count equ 10h  data ends  code segment  start : mov ax,data  mov ds,ax  mov cx,count  l1 : mov ah,1  int 21h  cmp al,60h  jnc upper  add al,20h  jmp skip  upper : sub al,20h  skip : mov ah,2h  mov dl,al  int 21h  loop l1  mov ah,4ch  int 21h  code ends  end start | Data segment is initialized  count is initialized to 10h  Code segment begins  Address of the data is transferred to AX , from AX transferred to DS.  Move count to CX register  Move 1 to AH , when AH = 1, a character is read from standard input and echoed back to the standard output.  Compare AL with 96(60h) to decide whether it is lower case or upper case  Jump to “Upper” if no carry is generated  Add AL by 20h will convert to lower case  Jump to Skip  Sub AL by 20h will convert to upper case  AH = 2 will give the output  Move AL to DL , loop from l1 till CX becomes 0  Store the result  Program terminates |

Unassembled code:



Execution:



Result:

Case conversion is executed and verified using an emulator.

# UCS1512 – Microprocessors Lab

# Floating point operations

Exp no : 9 Name: Sreedhar V

Date : 25-10-2020 Reg no: 185001161

# AIM:

To program and execute the code for floating point operations like addition and subtraction in 8086 using an emulator.

# Floating point addition:

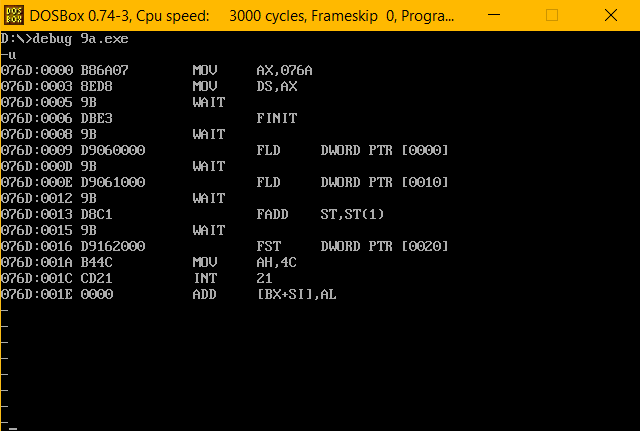
## Algorithm:

* + Program is set to run from any specified memory position.
  + Move the address of data segment to register DS .
  + Use define double word(dd) directive to declare the variable -x,y and sum and initialize them.
  + Initialize the floating point unit of 8087 ‘s stack using FINIT
  + Load the contents x,y to stack using FLD
  + Add the contents of the stack using FADD
  + Transfer the result from the stack to the variable sum.
  + Terminate the program.

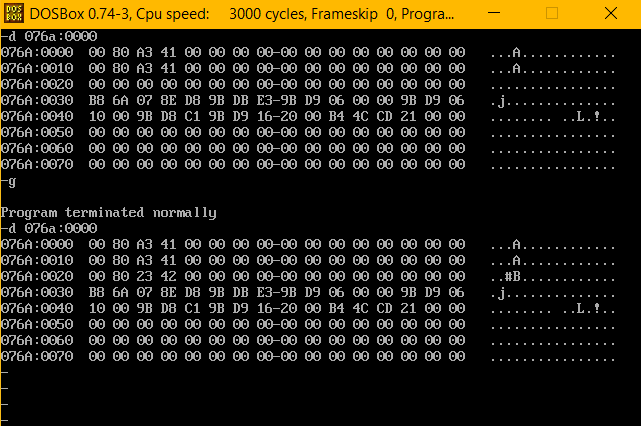
## Program:

|  |  |
| --- | --- |
| CODE | COMMENT |
| Program Floating point addition:  assume cs:code,ds:data  data segment  org 00h  x dd 20.4375  org 10h  y dd 20.4375  org 20h  sum dd ?  data ends  code segment  start : mov ax,data  mov ds,ax  finit  fld x  fld y  fadd st(0),st(1)  fst sum  mov ah,4ch  int 21h  code ends  end start | Data segment is initialized  x and y are declared and initialized with values 20.4375 and 20.4375 respectively  sum is declared  Code segment begins  Address of the data is transferred to AX , from AX transferred to DS.  Initialize the floating point unit stack of  8087.  loading x to ST(0) //stack  loading y to ST(0)    Adding the stack contents  Storing ST(0) to sum.  Program terminates |

Unassembled code:



Execution:



Result:

Floating point addition is executed and verified using an emulator.

# Floating point subtraction:

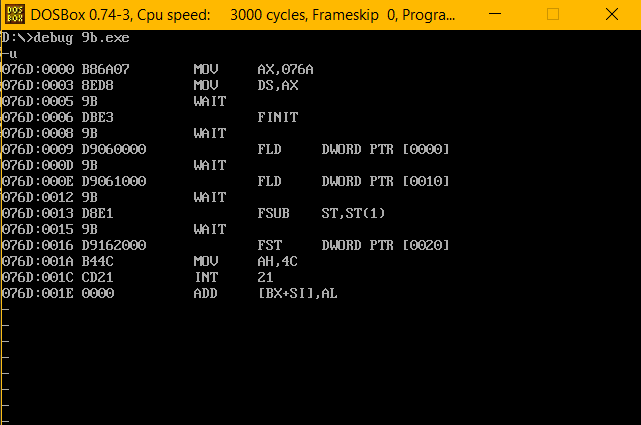
## Algorithm:

* + Program is set to run from any specified memory position.
  + Move the address of data segment to register DS .
  + Use define double word(dd) directive to declare the variable -x,y and sum and initialize them.
  + Initialize the floating point unit of 8087 ‘s stack using FINIT
  + Load the contents x,y to stack using FLD
  + Add the contents of the stack using FSUB
  + Transfer the result from the stack to the variable diff.
  + Terminate the program.

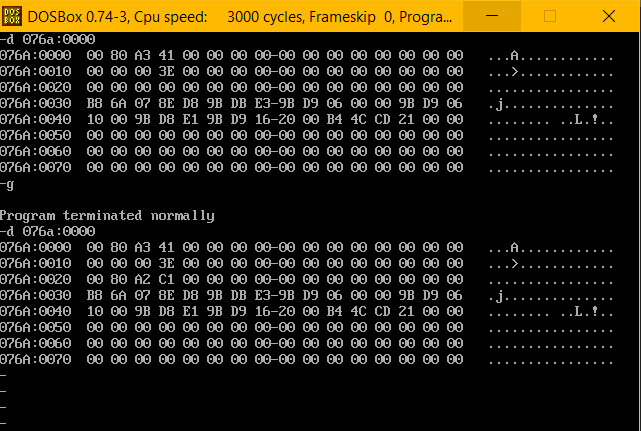
## Program:

|  |  |
| --- | --- |
| CODE | COMMENT |
| Program Floating point subtraction:  assume cs:code,ds:data  data segment  org 00h  x dd 20.4375  org 10h  y dd 0.125  org 20h  diff dd ?  data ends  code segment  start : mov ax,data  mov ds,ax  finit  fld x  fld y  fsub st(0),st(1)  fst diff  mov ah,4ch  int 21h  code ends  end start | Data and code segment is initialized  x and y are declared and initialized with values 20.4375 and 20.4375 respectively  diff is declared  Code segment begins  Address of the data is transferred to AX , from AX transferred to DS.  Initialize the floating point unit stack of  8087.  loading x to ST(0) //stack  loading y to ST(0)    Subtracting the stack contents  Storing ST(0) to diff.  Program terminates |

Unassembled code:



Execution:



Result:

Floating point subtraction is executed and verified using an emulator.

# UCS1512 – Microprocessors Lab

# Display a String

Exp no : 10 Name: Sreedhar V

Date : 25-10-2020 Reg no: 185001161

# AIM:

To program and execute the code for displaying a string in the standard output in 8086 using an emulator.

# Display a string:

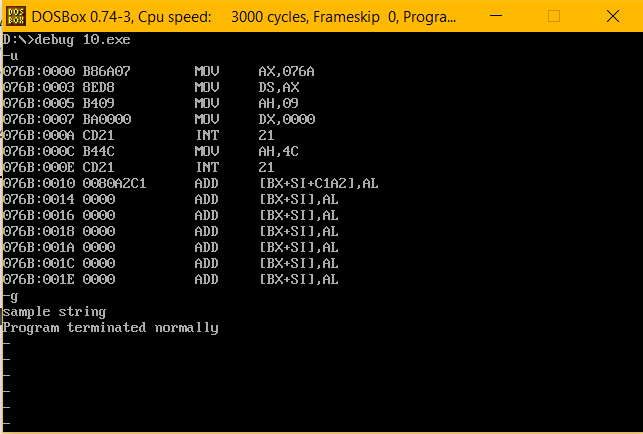
## Algorithm:

* + Program is set to run from any specified memory position.
  + Move 09h into the AH register and move the offset address of the variable which stores the string into the DX register
  + Use Interrupt 21h to display the string into standard output stream
  + Terminate the program

## **Program**:

|  |  |
| --- | --- |
| CODE | COMMENT |
| ;Program for displaying a string.  assume cs:code,ds:data  data segment  message db "sample string$"  data ends  code segment  start : mov ax,data  mov ds,ax  mov ah,9h  mov dx,offset message  int 21h  mov ah,4ch  int 21h  code ends  end start | Data and code segment initialized  message is declared and initialized to “sample string”    Code segment begins  Address of data segment moved to AX , from AX transferred to DS.  Move 9H to AH and offset value of the message to  the DX register  Call int 21h to display the message  Program terminates |

Unassembled code and Execution:



Result:

Displaying a string is executed and verified using an emulator.

# UCS1512 – Microprocessors Lab

# Display system date and time

Exp no : 11 Name: Sreedhar V

Date : 25-10-2020 Reg no: 185001161

# AIM:

To program and execute the code for displaying the system date and time in 8086 using an emulator.

# Display system date:

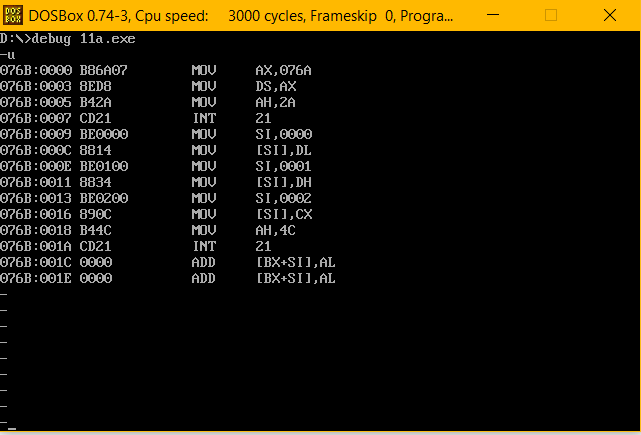
## Algorithm:

* + Program is set to run from any specified memory position.
  + Declare variables for day, month and year.
  + Move 2Ah into AH register and execute int 21h to get the system date.
  + After executing this function,move the value of day available in DL register to day, value of month available in DH to month and the value of year available in CX to year.
  + Terminate the program.

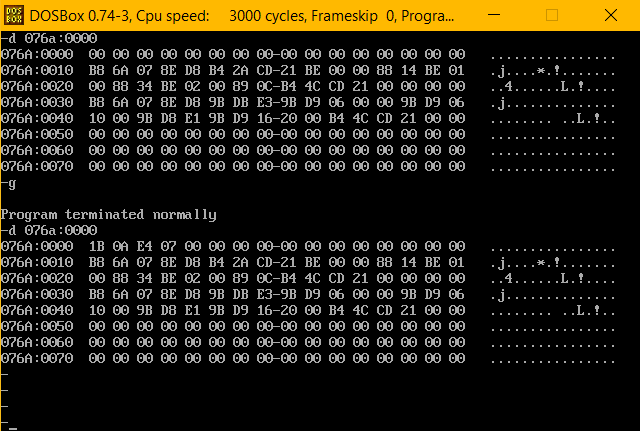
## Program:

|  |  |
| --- | --- |
| CODE | COMMENT |
| Program Displaying the date:  assume cs:code,ds:data  data segment  day db 01 dup(?)  month db 01 dup(?)  year db 02 dup(?)  data ends  code segment  start: mov ax,data  mov ds,ax  mov ah,2ah  int 21h  mov si,offset day  mov [si],dl  mov si,offset month  mov [si],dh  mov si,offset year  mov [si],cx  mov ah,4ch  int 21h  code ends  end start | Data and Code segment is initialized  Declare 1 byte to day  Declare 1 byte to month  Declare 2 byte to year //Uninitialized  Code segment begins  Address of the data is transferred to AX , from AX transferred to DS.  Move 2ah to AH and execute int 21 to get the system date    Value of the day , month and year are stored in DL ,DH and CX registers respectively    Move the contents of DL,DH,CX registers  to the offset value of variables declared to  store them.(DL to day , DH to month ,CX to year).    Program terminates |

Unassembled code:



Execution:



Result:

Displaying system date is executed and verified using an emulator.

# Display system time:

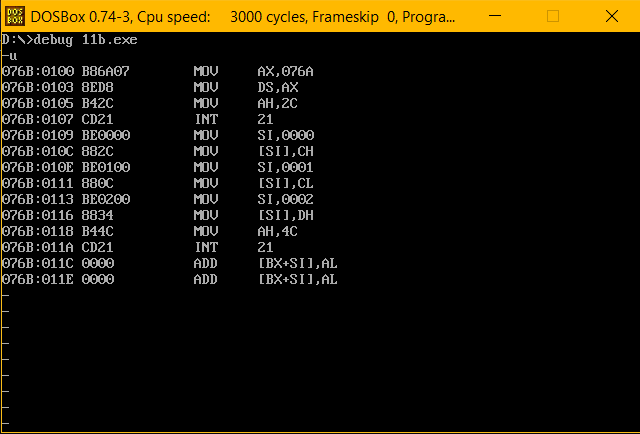
## Algorithm:

* + Program is set to run from any specified memory position.
  + Declare variables for day, month and year.
  + Move 2Ch into AH register and execute int 21h to get the system time.
  + After executing this function,move the value of hour available in CH register to day, value of minute available in CL register to month and the value of second available in DH register to year.
  + Terminate the program.

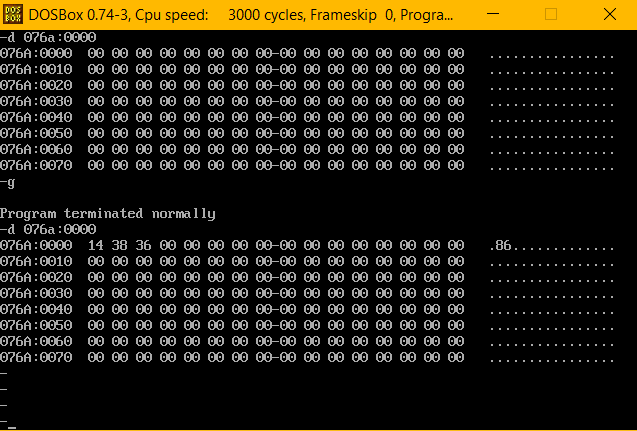
## Program:

|  |  |
| --- | --- |
| CODE | COMMENT |
| Program Displaying the time:  assume cs:code,ds:data  data segment  hour db ?  minute db ?  second db ?  data ends  code segment  org 0100h  start: mov ax,data  mov ds,ax  mov ah,2ch  int 21h  mov si,offset hour  mov [si],ch  mov si,offset minute  mov [si],cl  mov si,offset second  mov [si],dh  mov ah,4ch  int 21h  code ends  end start | Data and Code segment is initialized  Declare 1 byte to hour  Declare 1 byte to minute  Declare 1 byte to second //Uninitialized  Code segment begins  Address of the data is transferred to AX , from AX transferred to DS.  Move 2ch to AH and execute int 21 to get the system time    Value of the hour , minute and second are stored in CH ,CL and DH registers respectively    Move the contents of CH,CL and DH registers to the offset value of variables declared to store them.(CH to hour,  CL to minute , DH to second).    Program terminates |

Unassembled code:



Execution:



Result:

Displaying system time is executed and verified using an emulator.