# UCS1512 – Microprocessors Lab

# 16 BIT ARITHMETIC OPERATIONS

Exp no : 3 Name: Sreedhar V

Date : 11-09-2020 Reg no: 185001161

# AIM:

To program and execute the string manipulation like moving a string of bytes,comparing 2 strings of bytes,searching a byte in a string and moving a string without using string instructions in 8086 using an emulator.

# Moving a string of bytes:

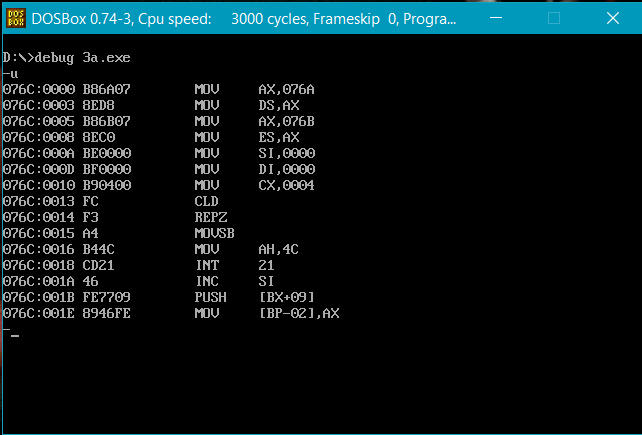
## Algorithm:

* + Program is set to run from any specified memory position.
  + Move the address of data segment to register DS .
  + Move the address of extra segment to register ES .
  + Move the offset of the source and dest to SI and DI respectively .
  + Initialize register CX to size of the source (No. Of bytes).
  + Clear the direction flag using CLD instruction.
  + Repetitively move the each byte of source to destination using rep movsb.
  + Terminate the program.

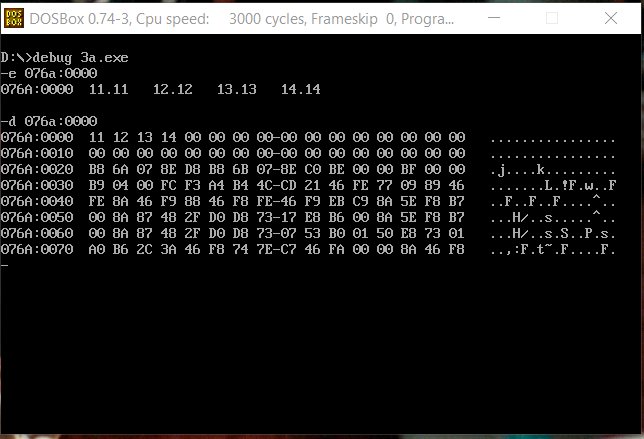
## Program:

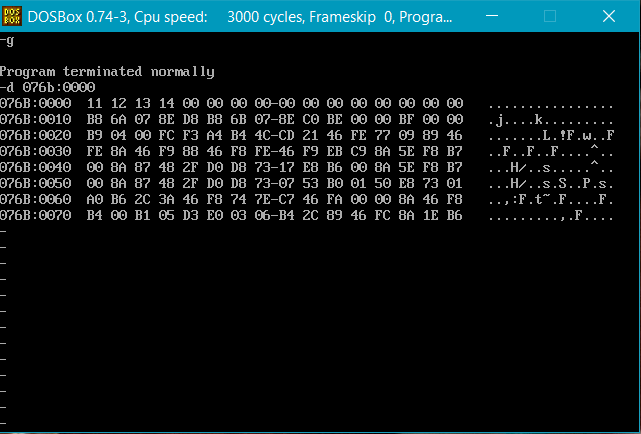
|  |  |
| --- | --- |
| CODE | COMMENT |
| Program for moving a string of bytes:  assume code: cs,ds:data,es:extra  data segment  source db 11h,12h,13h,14h  data ends  extra segment  dest db ?  extra ends  code segment  org 0100h  start : mov ax,data  mov ds,ax  mov ax,data  mov es,ax  mov si,offset source  mov di,offset dest  mov cx,0004h  cld  rep movsb  mov ah,4ch  int 21h  code ends  ends start | Data segment is initialized  source is initialized and set to 11h,12h,13h,14h  Extra segment is initialized  dest is initialized  Code segment begins  Originating address is set to 0100h  Address of the data and extra are transferred to AX , from AX transferred to  DS and ES respectively  Offset of the source and dest are transferred to SI and DI respectively  Initialize CX to 0004h,since string of 4 bytes is used  Clear the direction flag  Move each byte of string from SI to DI  till CX becomes 0  Program terminates |

Unassembled code:



Sample input and output:





Result:

Moving a string of bytes is executed and verified using an emulator.

# Comparing 2 strings of bytes:

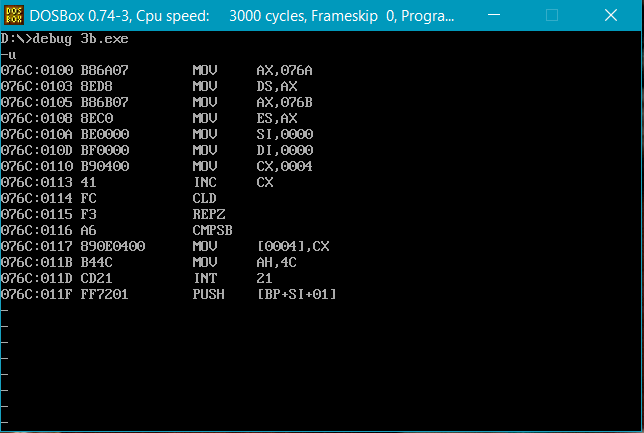
## Algorithm:

* + Program is set to run from any specified memory position.
  + Move the address of data segment to register DS .
  + Move the address of extra segment to register ES .
  + Move the offset of the source and dest to SI and DI respectively .
  + Initialize register CX to size of the source (No. Of bytes)
  + Clear the direction flag using CLD instruction.
  + Increment the CX register.
  + Repeat until the bytes of source and dest strings are equal.
  + Store the value of CX to status variable (Indicating the index of mismatch).
  + Terminate the program.

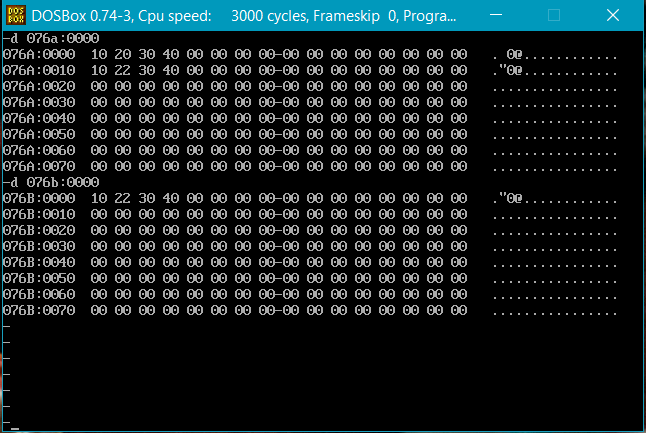
Program:

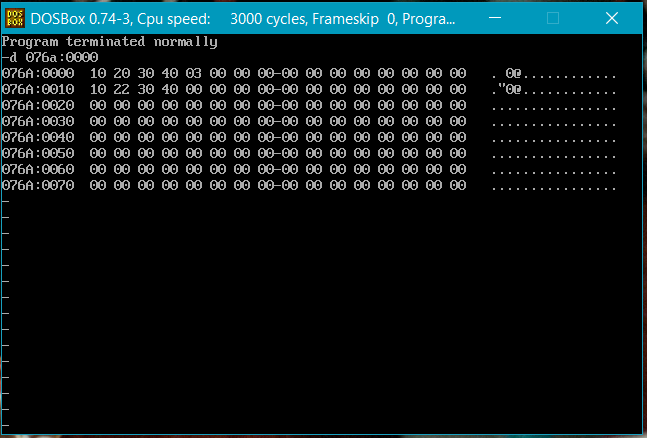
|  |  |
| --- | --- |
| CODE | COMMENT |
| Program for Comparing 2 strings of bytes:  assume code: cs,ds:data,es:extra  data segment  source db 10h,20h,30h,40h  status dw 0000h  data ends  extra segment  dest db 10h,22h,30h,40h  extra ends  code segment  org 0100h  start : mov ax,data  mov ds,ax  mov ax,data  mov es,ax  mov si,offset source  mov di,offset dest  mov cx,0004h  inc cx  cld  repe cmpsb  mov status,cx  mov ah,4ch  int 21h  code ends  ends start | Data segment is initialized  source is initialized and set to 10h,20h,30h,40h  status is initialized and set to 0000h  Extra segment is initialized  dest is initialized and set to 10h,22h,30h,40h  Code segment begins  Originating address is set to 0100h  Address of the data and extra are transferred to AX , from AX transferred to  DS and ES respectively  Offset of the source and dest are transferred to SI and DI respectively  Initialize CX to 0004h,since string of 4 bytes is used  Increment CX  Clear the direction flag  Compare each byte of string from SI and DI  till CX becomes 0  Transfer data from CX to status  Program terminates |

Unassembled code:



Execution:





Result:

Comparing 2 strings of bytes is executed and verified using an emulator.

# Searching a byte in a string:

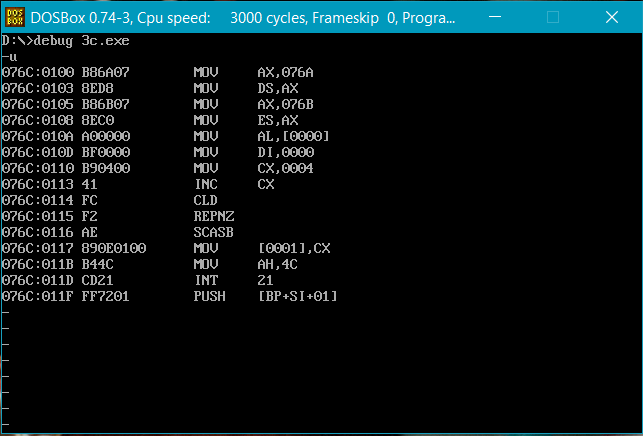
## Algorithm:

* + Program is set to run from any specified memory position.
  + Move the address of data segment to register DS .
  + Move the address of extra segment to register ES .
  + Move the source to Accumulator Register AL.
  + Move the offset of the dest to DI .
  + Increment CX register.
  + Clear the direction flag using CLD instruction.
  + Initialize register CX to size of the source (No. Of bytes).
  + Repeat until the source byte and DI are not equal.
  + Store the value of CX to status variable (Indicating the index of match).
  + Terminate the program.

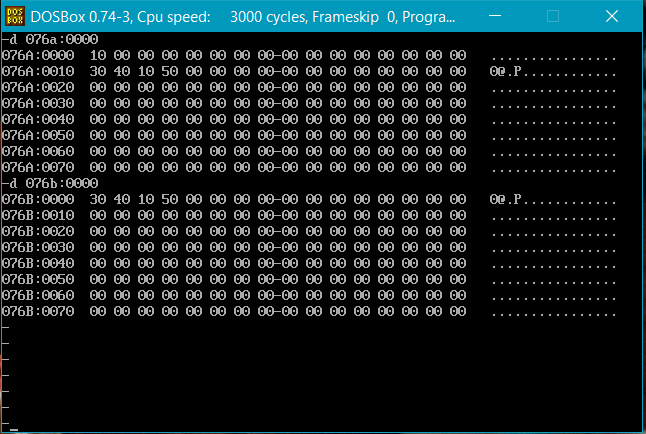
## Program:

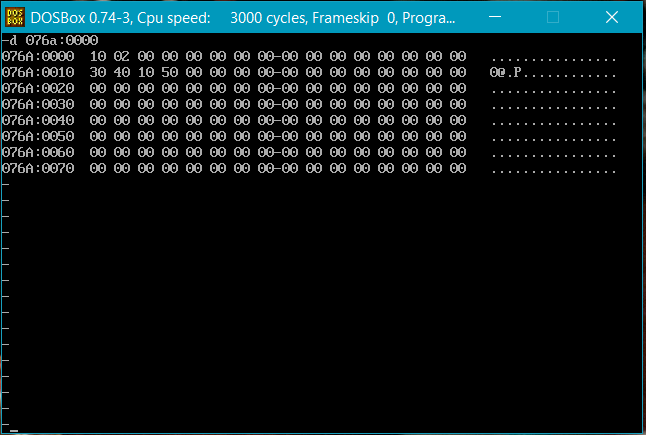
|  |  |
| --- | --- |
| CODE | COMMENT |
| Program for Searching a byte in a string:  assume code: cs,ds:data,es:extra  data segment  source db 10h  status dw 0000h  data ends  extra segment  dest db 30h,40h,10h,50h  extra ends  code segment  org 0100h  start : mov ax,data  mov ds,ax  mov ax,data  mov es,ax  mov al,source  mov di,offset dest  mov cx,0004h  inc cx  cld  repne scasb  mov status,cx  mov ah,4ch  int 21h  code ends  ends start | Data segment is initialized  source is initialized and set to 10h  status is initialized and set to 0000h  Extra segment is initialized  dest is initialized and set to 30h,40h,10h,50h  Code segment begins  Originating address is set to 0100h  Address of the data and extra are transferred to AX , from AX transferred to  DS and ES respectively  Value of source is transferred to AL  Offset of the dest is transferred to DI  Move data from source to AL register  Initialize CX to 0004h,since string of 4 bytes is used  Increment CX  Clear the direction flag  Compare the byte of string from AL and content in DI until CX becomes 0  Transfer data from CX to status  Program terminates |

Unassembled code:



Execution:





Result:

Searching a byte in a string is executed and verified using an emulator.

# Moving a string without using string instruction:

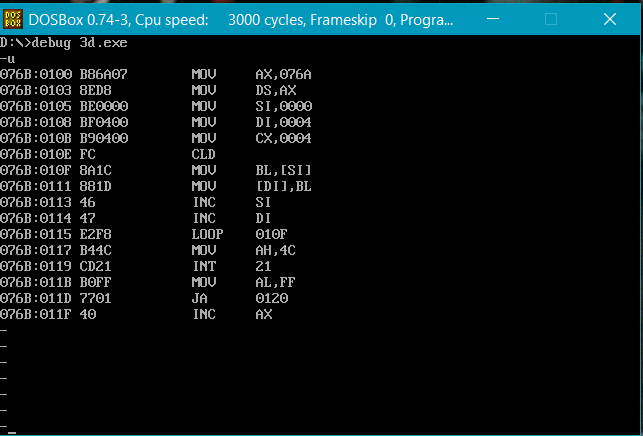
## Algorithm:

* + Load data from opr1 to register AX (first number).
  + Move the address of data segment to register DS .
  + Move the offset of the source and dest to SI and DI respectively .
  + Initialize register CX to size of the source (No. Of bytes).
  + Clear the direction flag using CLD instruction.
  + Repetitively move the each byte of source to destination using an explicit loop until CX becomes 0 and increment SI and DI index registers each time inside the loop.
  + Terminate the program.

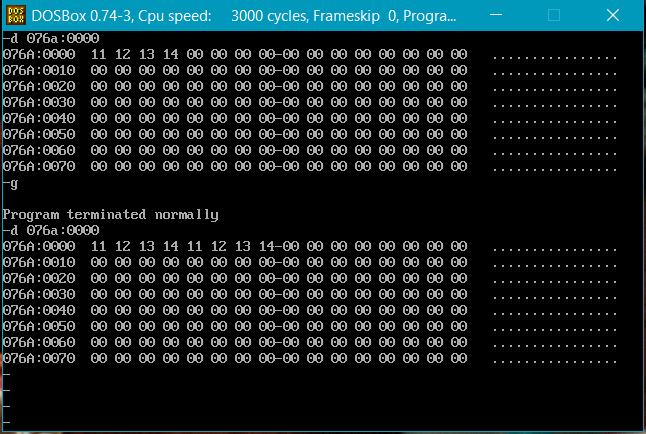
## Program:

|  |  |
| --- | --- |
| CODE | COMMENT |
| Program for moving a string without using string instructions:  assume code: cs,ds:data  data segment  source db 11h,12h,13h,14h  dest db ?  data ends  code segment  org 0100h  start : mov ax,data  mov ds,ax  mov si,offset source  mov di,offset dest  mov cx,0004h  cld  here : mov bl,[si]  mov [di],bl  inc si  inc di  loop here  mov ah,4ch  int 21h  code ends  ends start | Data segment is initialized  source is initialized and set to 11h,12h,13h,14h  dest is initialized  Code segment begins  Originating address is set to 0100h  Address of the data are transferred to AX , from AX transferred to DS.  Offset of the source and dest are transferred to SI and DI respectively  Initialize CX to 0004h,since string of 4 bytes is used  Clear the direction flag  Move each byte of string from SI to DI  till CX becomes 0 by transferring the value in SI address to BL and from BL to DI.  Increment SI and DI  Loop label is here.  Program terminates |

Unassembled code:



Execution:



Result:

Moving a string without using string instruction is executed and verified using an emulator.