Exp No: 3 **Date:** 14/09/2020

String Manipulations

Name: Swetha Saseendran

Register Number: 185001183

Aim:

To write and execute 8086 programs for String Manipulation operations like Move, Compare and Search.

Programs:

(i) Moving a string of bytes

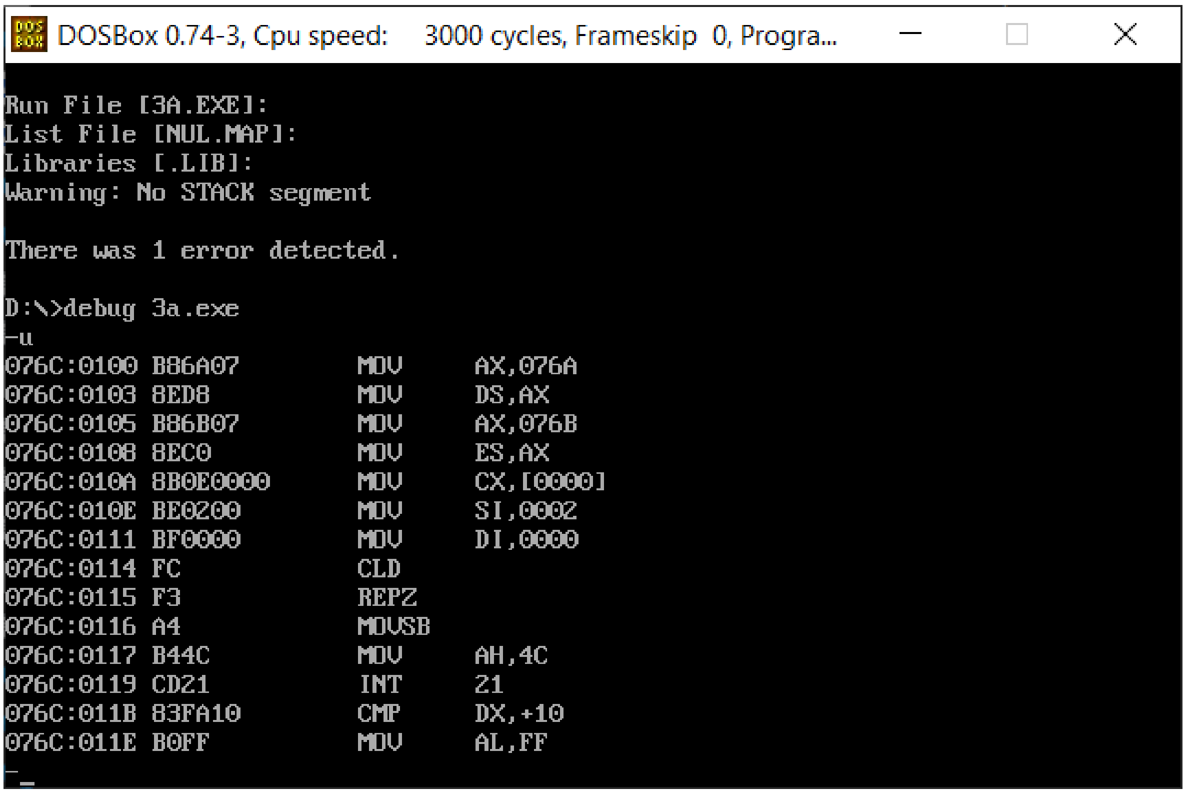
Algorithm:

- Move the data segment to DS register through AX register.
- Move the extra segment to ES register through AX register.
- Assign length of the string to CX register.
- Move offset of source string to SI register.
- Move offset of destination string DI register.
- Clear direction and move string byte by byte.

Program	Comments
assume *cs:code,ds:data,es:extra*	Using assume directive to declare data,extra and code segment
data segment src db 01h,02h,03h,04h srcLen dw 0004h **data ends**	Declaring and initialising variables in data segment
extra segment dst db ? **extra ends**	Declaring and initialising variables in extra segment
code segment	
org 0100h	Set location for code segment at 0100h
start:	
mov ax,data	Move the content of Data segment to AX register
mov ds,ax	Move the content of AX register to DS register

mov ax,extra	Move the content of extra segment to AX register
mov es,ax	Move the content of AX register segment to Extra segment.
mov si,offset src	Assign the offset of source to SI register.
mov di,offset dst	Assign the offset of source to SI register.
mov cx,srcLen	Assign value scrLen to CX register(Length of the string)
rep movsb	Repeat move string operation for all bytes.
mov ah,4ch	Moves the hexadecimal value 4c to ah.
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates
code ends	Ending the code segment
end start	

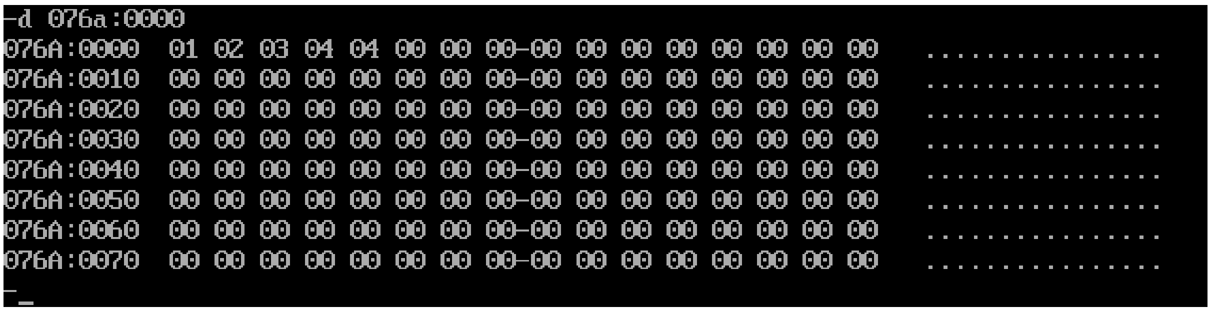
Unassembled Code:


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

```
Run File [3A.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
Warning: No STACK segment

There was 1 error detected.

D:\>debug 3a.exe
-u
076C:0100 B86A07        MOV     AX,076A
076C:0103 8ED8          MOV     DS,AX
076C:0105 B86B07        MOV     AX,076B
076C:0108 8EC0          MOV     ES,AX
076C:010A 8B0E0000      MOV     CX,[0000]
076C:010E BE0200        MOV     SI,0002
076C:0111 BF0000        MOV     DI,0000
076C:0114 FC            CLD
076C:0115 F3            REPZ
076C:0116 A4            MOVSB
076C:0117 B44C          MOV     AH,4C
076C:0119 CD21          INT     21
076C:011B 83FA10        CMP     DX,+10
076C:011E B0FF          MOV     AL,FF
-_
```

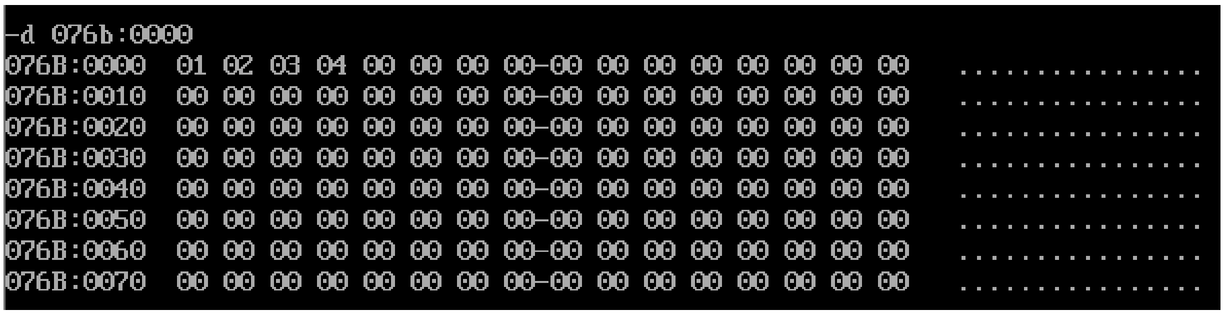
Snapshot of sample input and output:

INPUT:



```
-d 076a:0000
076A:0000  01 02 03 04 04 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:



```
-d 076b:0000
076B:0000  01 02 03 04 00 00 00 00-00 00 00 00 00 00 00 00   ................
076B:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076B:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076B:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076B:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076B:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076B:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076B:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

(ii) Comparing 2 strings of bytes

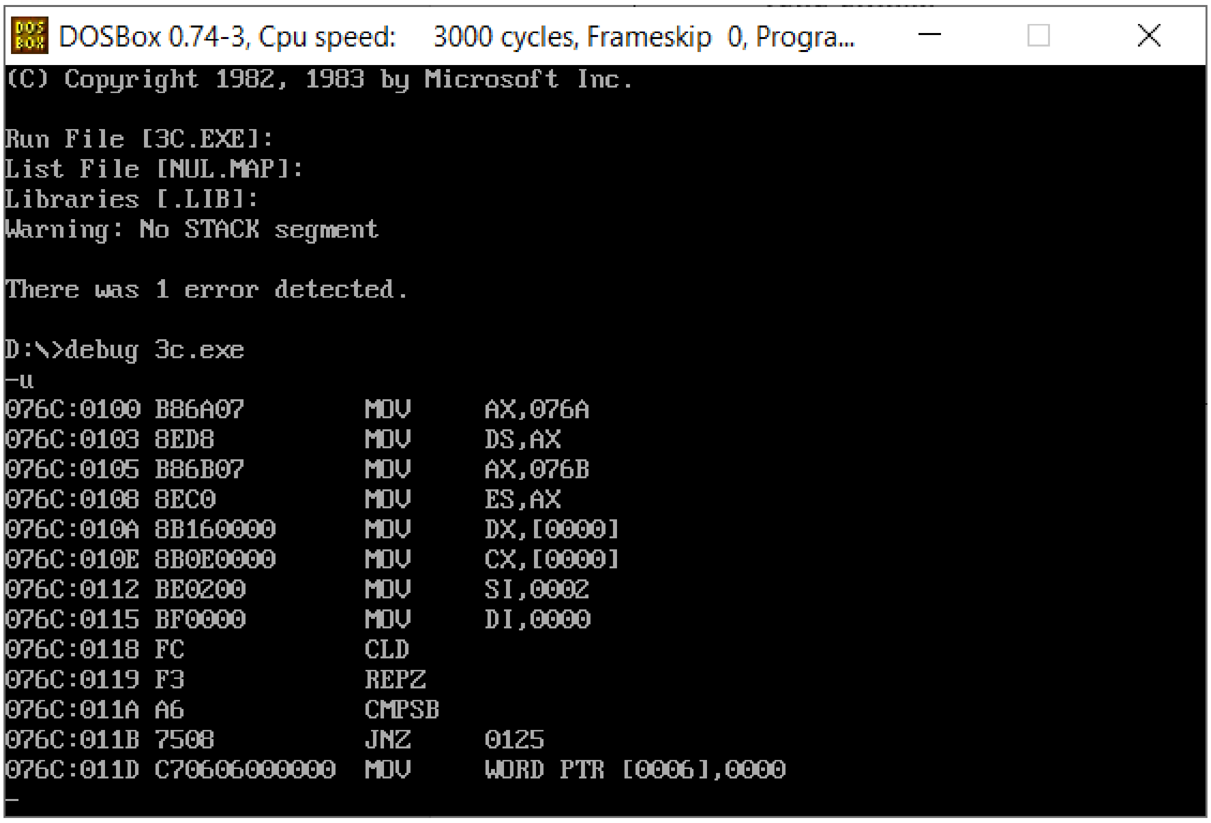
Algorithm:

- Move the data segment to DS register through AX register.
- Move the extra segment to ES register through AX register.
- Assign length of the string to CX register.
- Move offset of source string to SI register.
- Move offset of destination string DI register.
- Clear direction and compare string byte by byte.
- If all bytes are equal, assign zero to status. Else Assign CX to Status.

Program	Comments
assume cs:code,ds:data,es:extra	Using assume directive to declare data,extra and code segment
data segment	Declaring and initialising variables in data segment

src db 01h,02h,03h,04h	
srcLen dw 0004h	
flag dw 0004h	
data ends	
extra segment	Declaring and initialising variables in extra segment
str2 db 02h,01h,03h,04h	
extra ends	
code segment	Set location for code segment at 0100h
org 0100h	
start:	
mov ax,data	Move the content of Data segment to AX register
mov ds,ax	Move the content of AX register to DS register
mov ax,extra	Move the content of extra segment to AX register
mov es,ax	Move the content of AX register segment to Extra segment.
mov si,offset src	Assign the offset of source to SI register.
mov di,offset dst	Assign the offset of source to SI register.
mov cx,srcLen	Assign value scrLen to CX register(Length of the string)
rep cmpsb	Repeat comparison of strings byte by byte till they are equal.
jne mismatch	Jump to mismatch if bytes are not equal
mov flag,cx	Move the content of CX to flag
mov ah,4ch	Moves the hexadecimal value 4c to ah.
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates
mismatch:	
sub flag,cx	Subtract CX from flag to get index
mov ah,4ch	Moves the hexadecimal value 4c to ah.
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates
code ends	Ending the code segment
end start	

Unassembled Code:

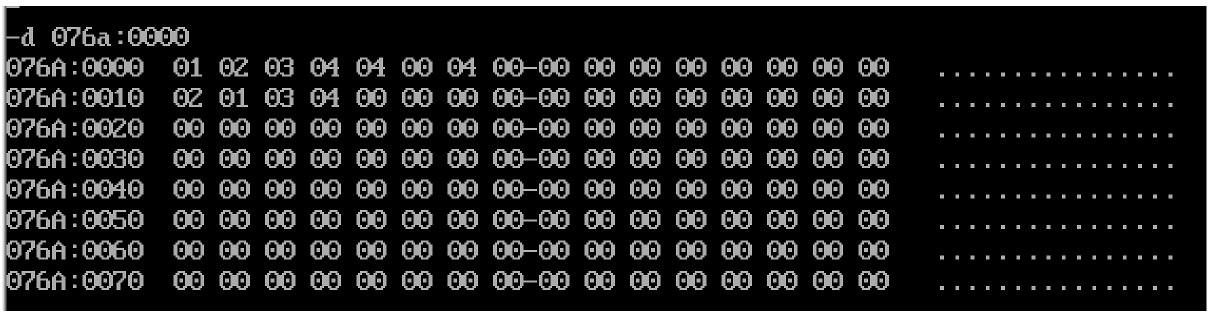


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
(C) Copyright 1982, 1983 by Microsoft Inc.

Run File [3C.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
Warning: No STACK segment

There was 1 error detected.

D:\>debug 3c.exe
-u
076C:0100 B86A07        MOV     AX,076A
076C:0103 8ED8          MOV     DS,AX
076C:0105 B86B07        MOV     AX,076B
076C:0108 8EC0          MOV     ES,AX
076C:010A 8B160000      MOV     DX,[0000]
076C:010E 8B0E0000      MOV     CX,[0000]
076C:0112 BE0200        MOV     SI,0002
076C:0115 BF0000        MOV     DI,0000
076C:0118 FC            CLD
076C:0119 F3            REPZ
076C:011A A6            CMPSB
076C:011B 7508          JNZ     0125
076C:011D C70606000000  MOV     WORD PTR [0006],0000
-
```

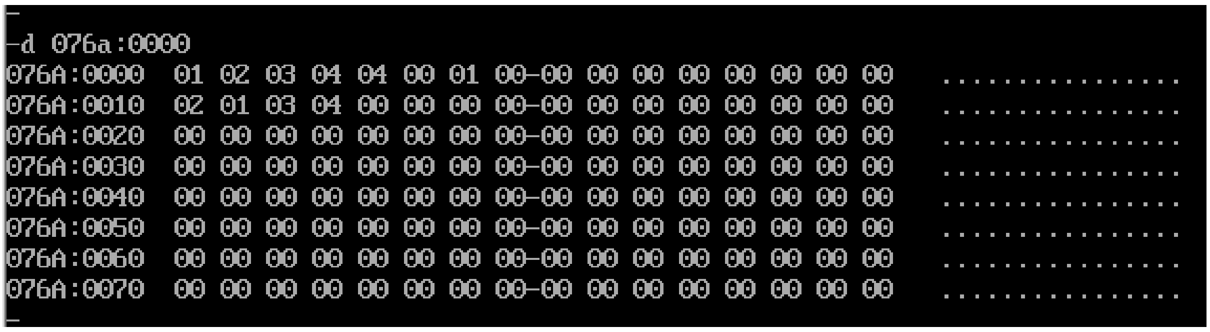
Snapshot of sample input and output:

INPUT:



```
-d 076a:0000
076A:0000  01 02 03 04 04 00 04 00-00 00 00 00 00 00 00 00   ................
076A:0010  02 01 03 04 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:



```
-
-d 076a:0000
076A:0000  01 02 03 04 04 00 01 00-00 00 00 00 00 00 00 00   ................
076A:0010  02 01 03 04 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   ................
-
```

(iii) Searching a byte in a string

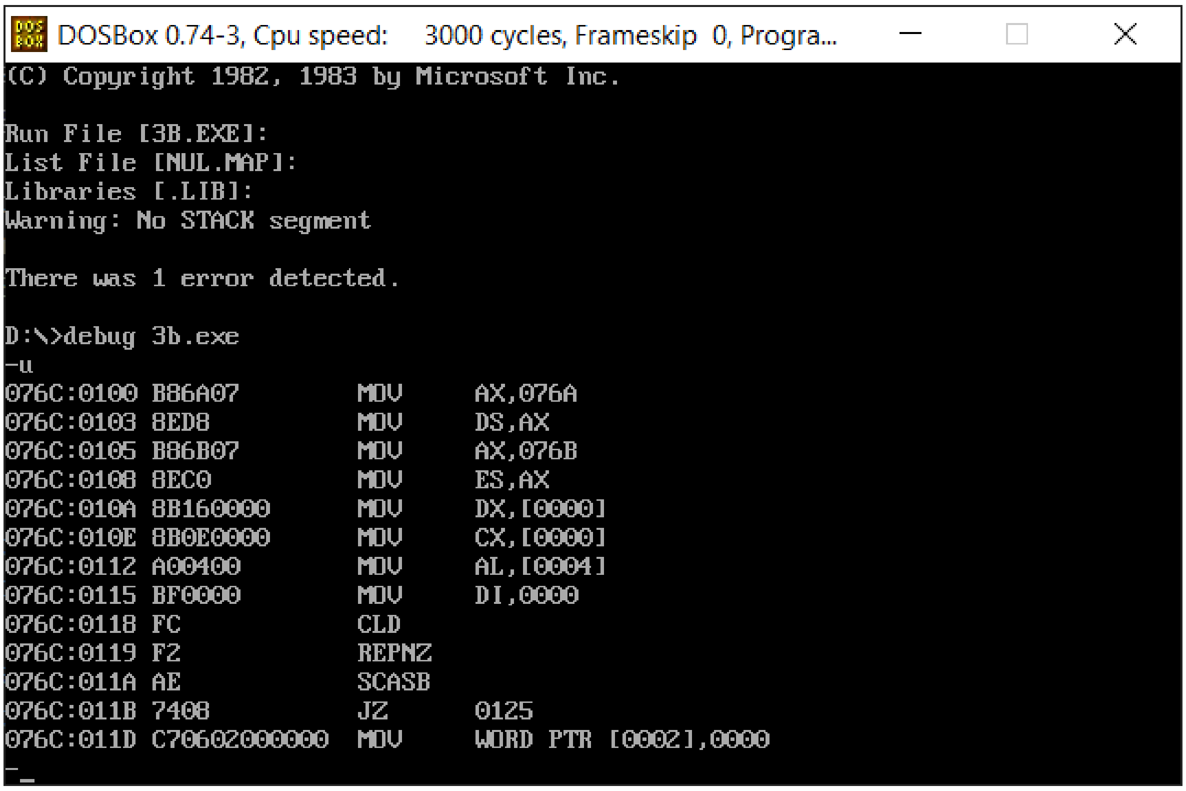
Algorithm:

- Move the data segment to DS register through AX register.
- Move the data segment to ES register through AX register.
- Assign length of the string to CX register.
- Move strSub to AL register.
- Move offset of str to DI register.
- Clear direction and search the string byte by byte.
- If all bytes are equal, assign CX to status. Else Assign zero to flag

Program	Comments
assume cs:code,ds:data,es:extra	Using assume directive to declare data,extra and code segment
data segment strSub db 03h strLen dw 04h flag dw 0004h **data ends**	Declaring and initialising variables in data segment
extra segment str2 db 02h,01h,03h,04h **extra ends**	Declaring and initialising variables in extra segment
code segment *org 0100h*	Set location for code segment at 0100h
start:	
mov ax,data	Move the content of Data segment to AX register
mov ds,ax	Move the content of AX register to DS register
mov ax,extra	Move the content of extra segment to AX register
mov es,ax	Move the content of AX register segment to Extra segment.
mov di,offset str	Assign the offset of str to SI register.
mov di,offset dst	Assign the offset of source to SI register.
mov cx,srcLen	Assign value strLen to CX register(Length of the string)
mov al,strSub	Assign value strSub to AL register
cld	Clear direction
repne scasb	Repeat searching strings byte by byte till they are not equal.
jz match	Jump to match if bytes are equal
mov flag,0000h	Assign 0 to flag
mov ah,4ch	Moves the hexadecimal value 4c to ah.

int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates
match:	Subtract CX from flag to get index
sub flag,cx	Subtract CX from flag to get index
mov ah,4ch	Moves the hexadecimal value 4c to ah.
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates
code ends	Ending the code segment
end start	

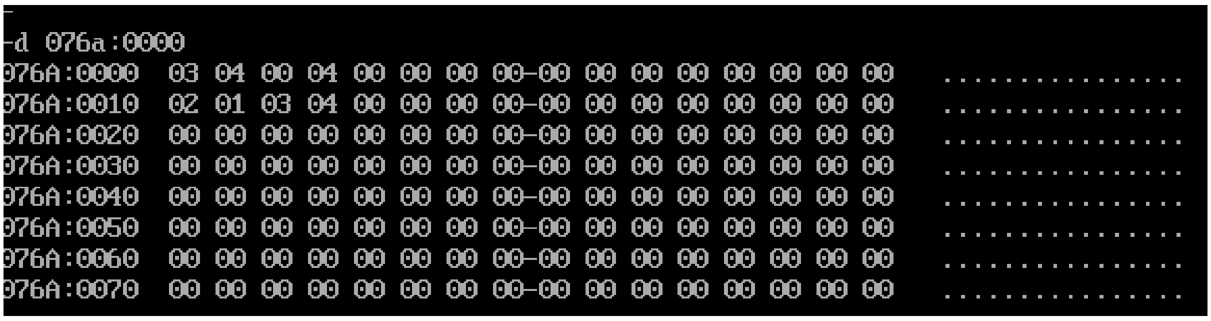
Unassembled Code:


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

```
(C) Copyright 1982, 1983 by Microsoft Inc.

Run File [3B.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
Warning: No STACK segment

There was 1 error detected.

D:\>debug 3b.exe
-u
076C:0100 B86A07        MOV     AX,076A
076C:0103 8ED8          MOV     DS,AX
076C:0105 B86B07        MOV     AX,076B
076C:0108 8EC0          MOV     ES,AX
076C:010A 8B160000      MOV     DX,[0000]
076C:010E 8B0E0000      MOV     CX,[0000]
076C:0112 A00400        MOV     AL,[0004]
076C:0115 BF0000        MOV     DI,0000
076C:0118 FC            CLD
076C:0119 F2            REPNZ
076C:011A AE            SCASB
076C:011B 7408          JZ      0125
076C:011D C70602000000  MOV     WORD PTR [0002],0000
-_
```

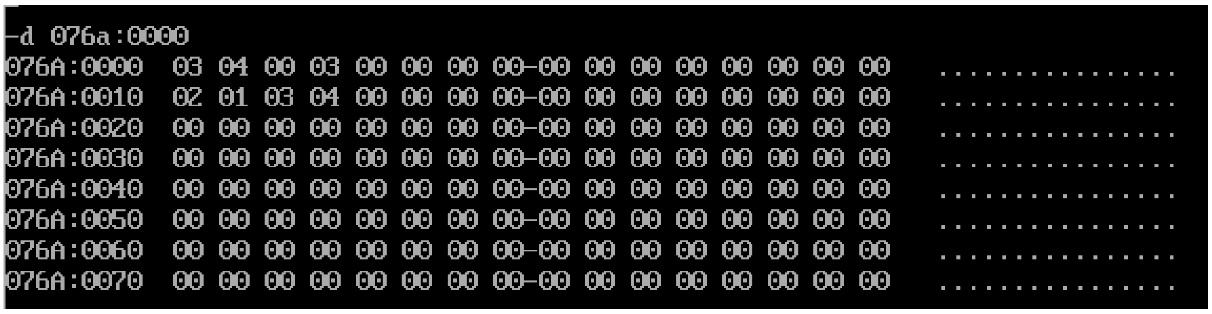
Snapshot of sample input and output:

INPUT:



```
-
-d 076a:0000
076A:0000  03 04 00 04 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  02 01 03 04 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:



```
-d 076a:0000
076A:0000  03 04 00 03 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  02 01 03 04 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   ................
```

(iv) Moving a string without using string instructions

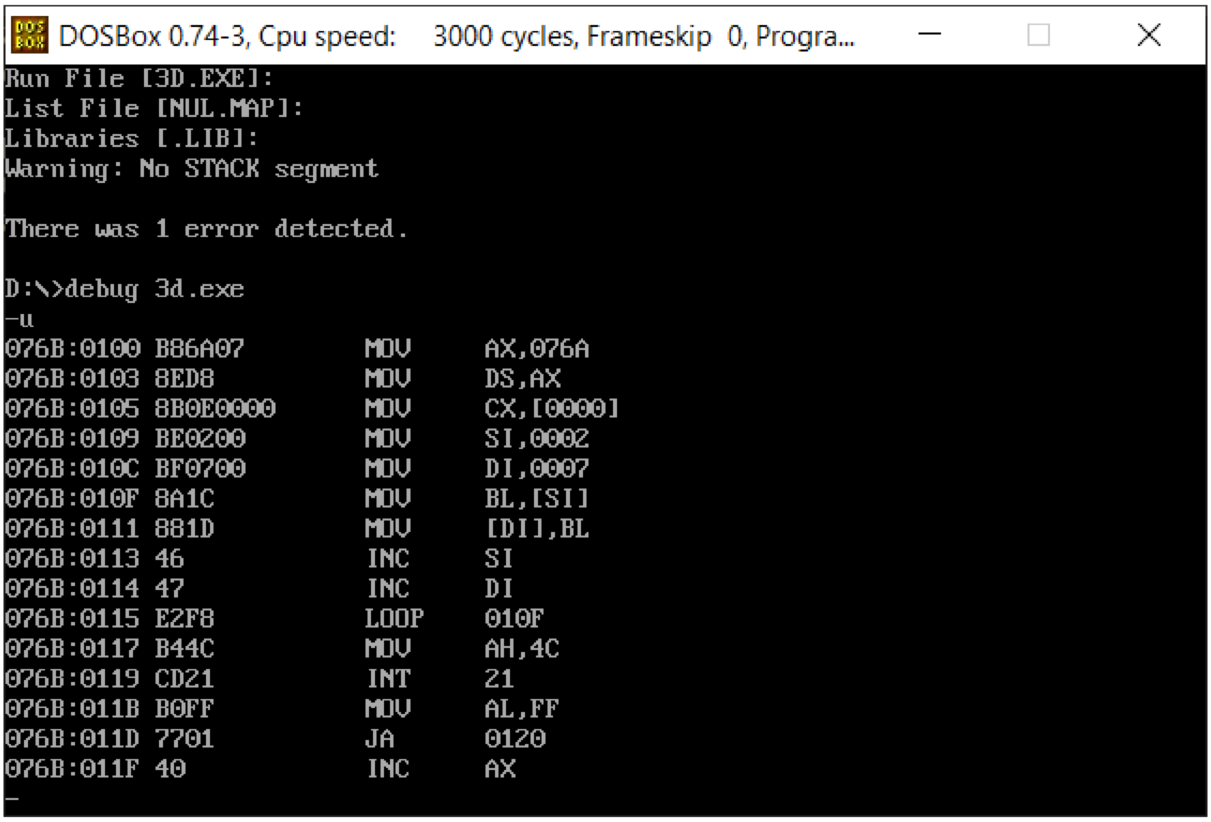
Algorithm:

- Move the data segment to DS register through AX register.
- Move the extra segment to ES register through AX register.
- Assign length of the string to CX register.
- Move offset of source string to SI register.
- Move offset of destination string DI register.
- Move [SI] to [DI] through AL register.
- Increment SI and DI.
- Repeat Step 6 and 7 for all the bytes in the source string.

Program	Comments
***assume** cs:code,ds:data*	Using assume directive to declare data and code segment
data segment src db 01h,02h,03h,04h srcLen dw 0004h dst db ? **data ends**	Declaring and initialising variables in data segment
code segment *org 0100h*	Set location for code segment at 0100h
start: *mov ax,data*	Move the content of Data segment to AX register
mov ds,ax	Move the content of AX register to DS register
mov si,offset src	Assign the offset of source to SI register.
mov di,offset dst	Assign the offset of source to SI register.
mov cx,srcLen	Assign value scrLen to CX register(Length of the string)
copy:	Repeat comparison of strings byte by byte till they are equal.

mov al,[si]	Transfer Address of SI to AL.
mov [di],al	Transfer Value of AL to Address of DI.
inc si	SI=SI+1
inc di	DI=DI+1
dec cx	Decrement counter register CX
jnz copy	Loop to here until all the bytes are parsed
mov ah,4ch	Moves the hexadecimal value 4c to ah.
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates
code ends	Ending the segment with the segment name
end start	

Unassembled Code:

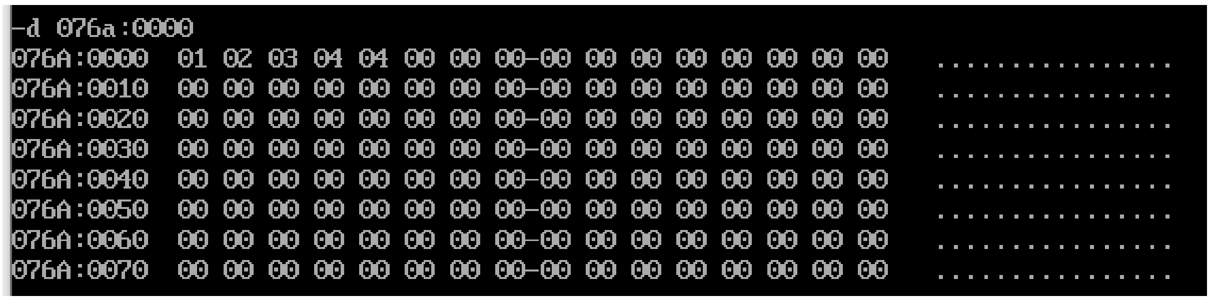


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
Run File [3D.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
Warning: No STACK segment

There was 1 error detected.

D:\>debug 3d.exe
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 8B0E0000      MOV     CX,[0000]
076B:0109 BE0200        MOV     SI,0002
076B:010C BF0700        MOV     DI,0007
076B:010F 8A1C          MOV     BL,[SI]
076B:0111 881D          MOV     [DI],BL
076B:0113 46            INC     SI
076B:0114 47            INC     DI
076B:0115 E2F8          LOOP    010F
076B:0117 B44C          MOV     AH,4C
076B:0119 CD21          INT     21
076B:011B B0FF          MOV     AL,FF
076B:011D 7701          JA      0120
076B:011F 40            INC     AX
-
```

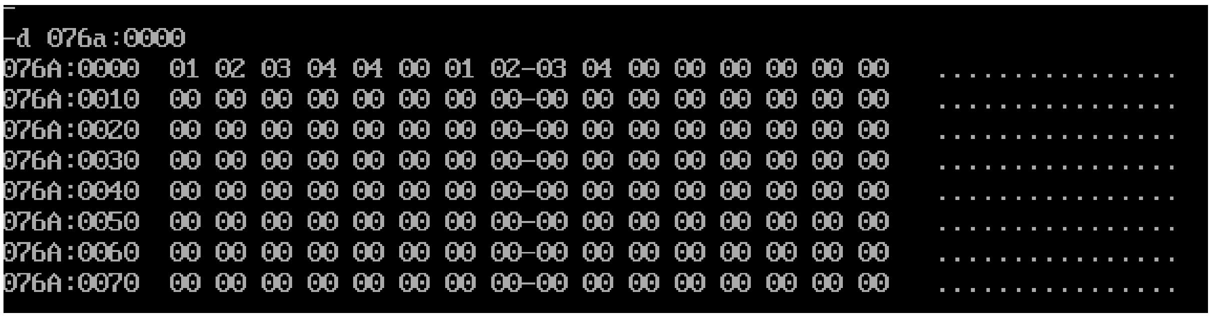
Snapshot of sample input and output:

INPUT:



```
-d 076a:0000
076A:0000  01 02 03 04 04 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:



```
-d 076a:0000
076A:0000  01 02 03 04 04 00 01 02-03 04 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 program to perform basic string manipulations using an 8086 microprocessor has been implemented.