SSN College of Engineering
Department of Computer Science and Engineering
UCS1512 – Microprocessors Lab
STRING MANIPULATIONS

Name: Prasanna Kumaran D
Registration Number: 185001110 15 September 2020
Semester: V

1 AIM:

To write and execute 8086 programs using strings like search, move and compare.

2 PROCEDURE:

- Write the program in a text editor and save it as a ***.asm*** file under the MASM directory.
- Launch DOSBOX application and mount the MASM folder using the command prompt.
- Use the following syntax for mounting : ' **mount [LOCAL DRIVE] FILEPATH** '. Enter into the local drive('LOCAL DRIVE:').
- The code file can be edited using the command **edit FILENAME.asm**. Save the changes and exit.
- Assemble the code using the command '**masm FILENAME.asm**' to generate the object file. The object file is in the format 'FILENAME.obj'
- Add dynamic libraries using the syntax **'link FILENAME.obj'** to generate the executeable(**.exe**) file.
- Enter the debug mode using debug FILENAME.exe to execute and analuse the memory contents. The various commands used in debug mode are as follows:-
  - U :- Displays unassembled code.
  - D :- Refers to the offset from which contents in the memory are displayed.
  - E :- Change the value in memory.
  - G :- execute the code.
  - Q :- Quit debug mode.

3 Algorithm & Program

INITIALIZATION:

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

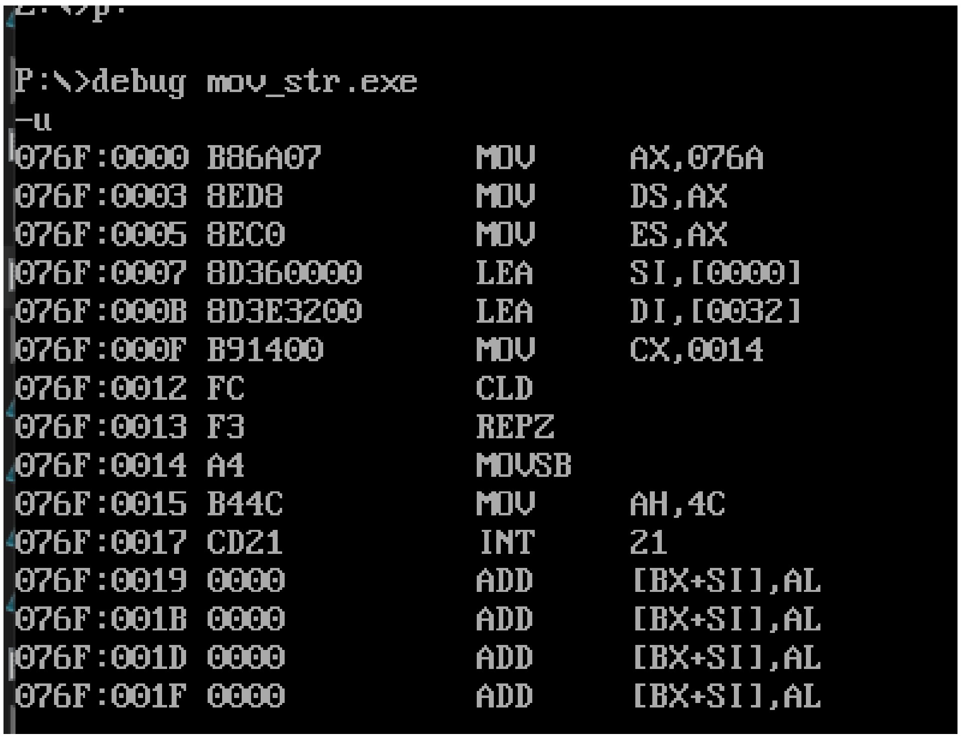
3.1 Moving a string of bytes:

To move bytes in a string

- Load the contents of the operands into their respective registers.
- Load effective address of the string to move into the SI register.
- Load effective address of the destination into the DI register.
- Clear the direction flag as string must be copied from lower to higher index value
- Repeat movsb until count becomes 0.
- Terminate the program

3.1.1 Moving a string of bytes: Program

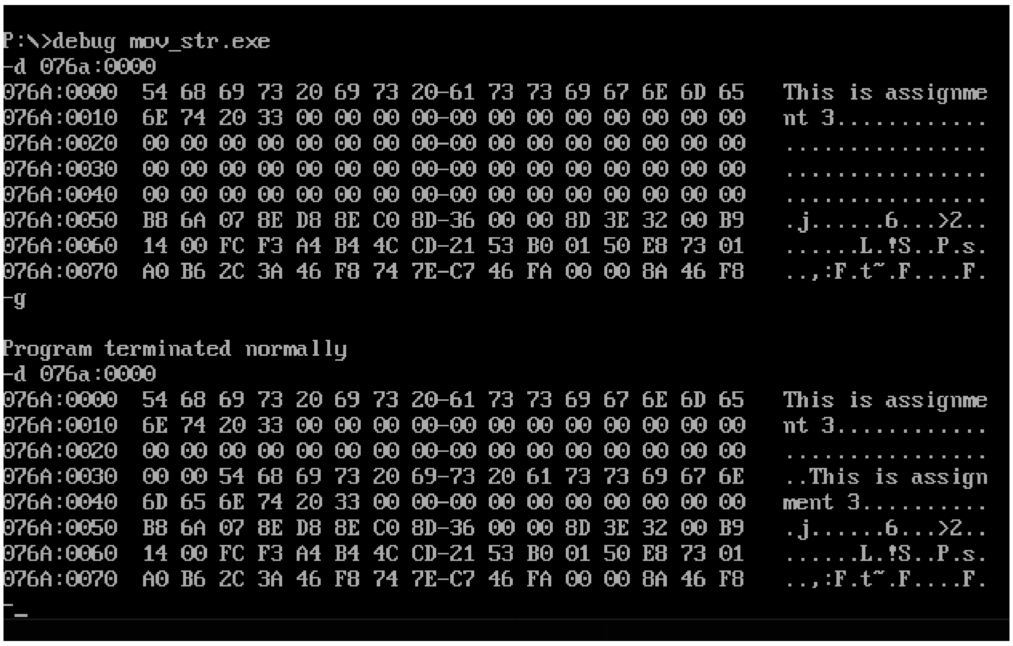
Program	Comments
mov ds, ax	Transfers contents of AX register to DS register.
mov es, ax	Transfers contents of AX register to ES register
lea si, mystring	Load effective address of mystring variable to SI register.
lea di, final	Load effective address of target destination variable to DI register.
mov cx, 20	Transfers contents 20(0s) to CX register.
cld	Clears directional flag.
REP movsb	REP represents repeat string instruction until count becomes 0.
mov ah, 4ch	Move the hexadecimal value 4c to ah
int 21h	When software interrupt 21 is called with AH=45,process is terminated



```
Z:\>p:

P:\>debug mov_str.exe
-u
076F:0000 B86A07        MOV     AX,076A
076F:0003 8ED8          MOV     DS,AX
076F:0005 8EC0          MOV     ES,AX
076F:0007 8D360000      LEA     SI,[0000]
076F:000B 8D3E3200      LEA     DI,[0032]
076F:000F B91400        MOV     CX,0014
076F:0012 FC            CLD
076F:0013 F3            REPZ
076F:0014 A4            MOVSB
076F:0015 B44C          MOV     AH,4C
076F:0017 CD21          INT     21
076F:0019 0000          ADD     [BX+SI],AL
076F:001B 0000          ADD     [BX+SI],AL
076F:001D 0000          ADD     [BX+SI],AL
076F:001F 0000          ADD     [BX+SI],AL
```

Figure 1: Move string bytes - unassembled



```
P:\>debug mov_str.exe
-d 076a:0000
076A:0000  54 68 69 73 20 69 73 20-61 73 73 69 67 6E 6D 65   This is assignme
076A:0010  6E 74 20 33 00 00 00 00-00 00 00 00 00 00 00 00   nt 3............
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  B8 6A 07 8E D8 8E C0 8D-36 00 00 8D 3E 32 00 B9   .j......6...>2..
076A:0060  14 00 FC F3 A4 B4 4C CD-21 53 B0 01 50 E8 73 01   ......L.!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  54 68 69 73 20 69 73 20-61 73 73 69 67 6E 6D 65   This is assignme
076A:0010  6E 74 20 33 00 00 00 00-00 00 00 00 00 00 00 00   nt 3............
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 54 68 69 73 20 69-73 20 61 73 73 69 67 6E   ..This is assign
076A:0040  6D 65 6E 74 20 33 00 00-00 00 00 00 00 00 00 00   ment 3..........
076A:0050  B8 6A 07 8E D8 8E C0 8D-36 00 00 8D 3E 32 00 B9   .j......6...>2..
076A:0060  14 00 FC F3 A4 B4 4C CD-21 53 B0 01 50 E8 73 01   ......L.!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: Moving a string

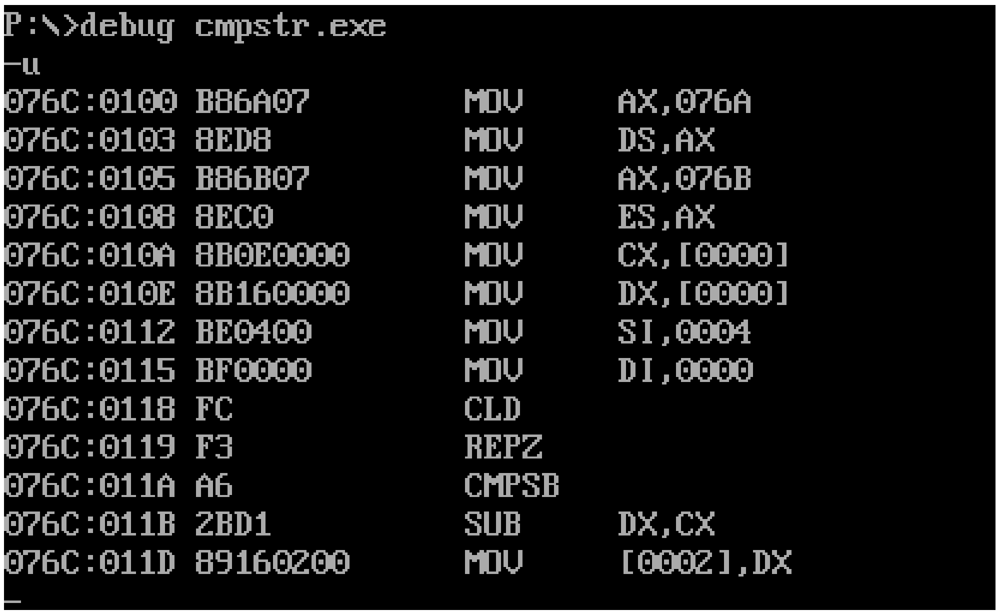
3.2 Comparing 2 strings of bytes:

To Compare bytes of two strings and return zero if equal else return the index of first occurrence.

- Initialize data and code segments.
- Load the contents of operands into their respective registers.
- Initialize count with number of bytes to compare
- Initialize status which displays the result
- Move starting address of data segment into DS and that of extra segment into es.
- Move the string values into SI and DI registers.
- Clear the direction flag as string must be copied from lower to higher index value.
- Repeat cmpsb (compare string byte) until count becomes 0.
- Terminate the program.

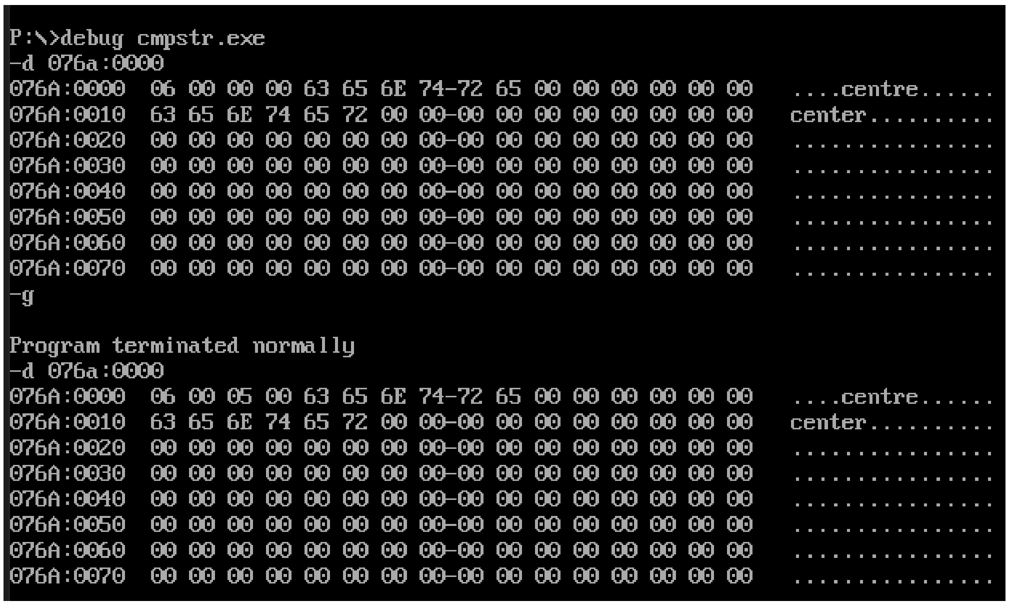
3.2.1 Comparing 2 strings of bytes: Program

Program	**Comments**
assume cs:code,ds:data,es:extra	
data segment count dw 0006h status dw 0000h str1 db 'centre' data ends	Initialize data segment and variables End data segment
extra segment str2 db 'center' extra ends	Initialise extra segment End extra segment
code segment	Initialise code segment
start: movcontents ax,data mov dscontents of ,ax mov ax,extra mov es,ax mov cx,count mov dx,count mov si, offset str1 mov di, offset str2 cld repe cmpsb sub dx,cx mov status,dx mov ah,4ch int 21h code ends end start	Transferring address of data segment to ds Transferring address of extra segment to ax and then to es Transfer count to cx and dx Transfer string 1 contents to SI register Transfer contents of string 2 to DI register Clear the directional flag Repeat comparing the string until cx becomes 0 Subtract dx and cx to find number of mismatch. Transfer contents of dx register to status register. Termination of execution End code segment Terminate



```
P:\>debug cmpstr.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 8B160000      MOV     DX,[0000]
076C:0112 BE0400        MOV     SI,0004
076C:0115 BF0000        MOV     DI,0000
076C:0118 FC            CLD
076C:0119 F3            REPZ
076C:011A A6            CMPSB
076C:011B 2BD1          SUB     DX,CX
076C:011D 89160200      MOV     [0002],DX
-
```

Figure 3: String compare - unassembled



```
P:\>debug cmpstr.exe
-d 076a:0000
076A:0000  06 00 00 00 63 65 6E 74-72 65 00 00 00 00 00 00   ....centre......
076A:0010  63 65 6E 74 65 72 00 00-00 00 00 00 00 00 00 00   center..........
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  06 00 05 00 63 65 6E 74-72 65 00 00 00 00 00 00   ....centre......
076A:0010  63 65 6E 74 65 72 00 00-00 00 00 00 00 00 00 00   center..........
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 4: String compare - Output

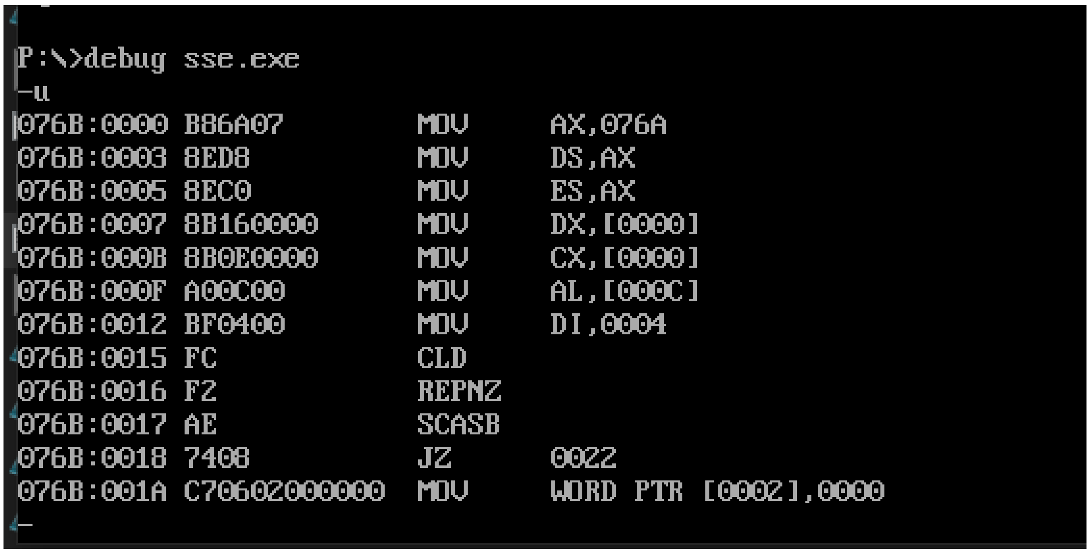
3.3 Searching a byte in a string:

To search a byte in a string and return zero if equal else return index of first occurrence.

- Initialize the data and code segments
- Load the contents of operands into their respective registers
- Initialize count with number of bytes to compare.
- Initialize status that will display the result
- Move starting address of data segment to DS and that of extra segment into ES
- Move the string values to AL and DI register
- Clear the directional flag as string must be copied from lower to higher index value
- Repeat scasb until count becomes 0
  - if equal to zero subtract DX and CX and move the result to status
- Terminate program

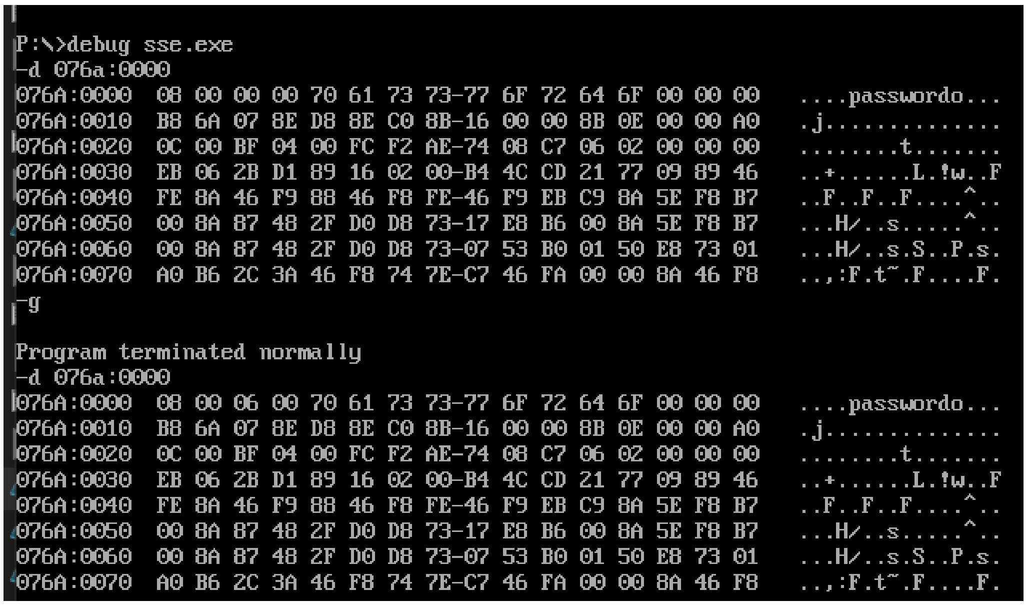
3.3.1 Searching a byte in a string: Program

Program	Comments
assume cs:code, ds:data	
data segment	Initialize data segment and variables
count dw 0008h	
status dw 0000h	
str1 db ’password’	
str2 db ’o’	
data ends	End data segment
code segment	Initialize code segment
org 0000h	
start:	
mov ax, data	
mov ds, ax	Transfer address of data segment to DS
mov es, ax	Transfer address of data segment to ES
mov dx, count	Transfer contents of count to DX
mov cx, count	Transfer contents of count to CX
mov al, [str2]	Transfer string 2 value to AL register
mov di, offset str1	Transfer string 1 value to DI register
cld	Clear directional flag
repne scasb	Repeat scanning the string until CX becomes 0.
je s1	Jump to S1
mov status, 0000h	Initialize status to 0
jmp s2	Jump to S2
s1:	
sub dx, cx	Subtract contents in DX and CX register
mov status, dx	Transfer contents of DX into status register
s2:	
mov ah, 4ch	Termination of execution.
int 21h	
code ends	End of the code segment
end start	Terminate program



```
P:\>debug sse.exe
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 8EC0          MOV     ES,AX
076B:0007 8B160000      MOV     DX,[0000]
076B:000B 8B0E0000      MOV     CX,[0000]
076B:000F A00C00        MOV     AL,[000C]
076B:0012 BF0400        MOV     DI,0004
076B:0015 FC            CLD
076B:0016 F2            REPNZ
076B:0017 AE            SCASB
076B:0018 7408          JZ      0022
076B:001A C70602000000  MOV     WORD PTR [0002],0000
-
```

Figure 5: String search - unassembled



```
P:\>debug sse.exe
-d 076a:0000
076A:0000  08 00 00 00 70 61 73 73-77 6F 72 64 6F 00 00 00   ....passwordo...
076A:0010  B8 6A 07 8E D8 8E C0 8B-16 00 00 8B 0E 00 00 A0   .j..............
076A:0020  0C 00 BF 04 00 FC F2 AE-74 08 C7 06 02 00 00 00   ........t.......
076A:0030  EB 06 2B D1 89 16 02 00-B4 4C CD 21 77 09 89 46   ..+......L.!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

Program terminated normally
-d 076a:0000
076A:0000  08 00 06 00 70 61 73 73-77 6F 72 64 6F 00 00 00   ....passwordo...
076A:0010  B8 6A 07 8E D8 8E C0 8B-16 00 00 8B 0E 00 00 A0   .j..............
076A:0020  0C 00 BF 04 00 FC F2 AE-74 08 C7 06 02 00 00 00   ........t.......
076A:0030  EB 06 2B D1 89 16 02 00-B4 4C CD 21 77 09 89 46   ..+......L.!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.
```

Figure 6: Byte search

3.4 Moving a string without using string instructions

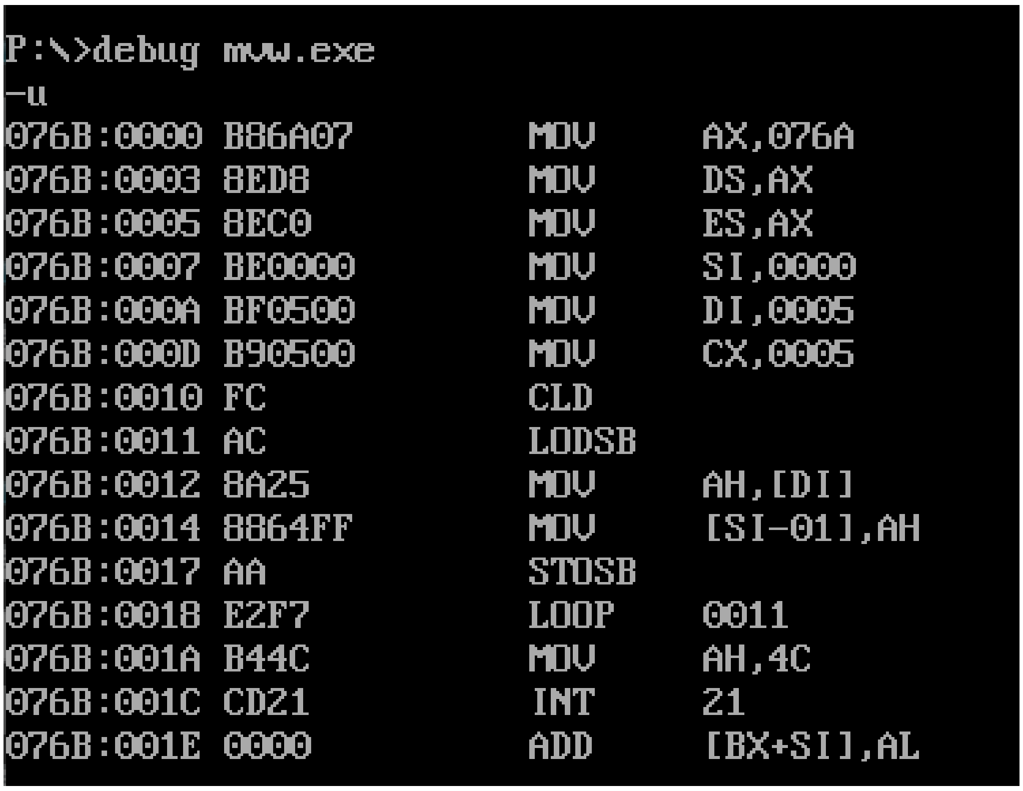
To move a string from a source location to destination without using string operations

- Initialise the data segment
- Move data segment address to ds
- Initialise the extra segment
- Initialise the source index register with the address of the string and the destination index register with the location
- Initialise the counter register CX
- Clear the direction flag that automatically increments the index registers
- Looping segment
  - Load the data move
  - Load data pointed by DI into AH
  - Load AH to location pointed by SI-1
  - Store the data in the location
- Terminate the program

3.4.1 Moving a string without using string instructions: Program

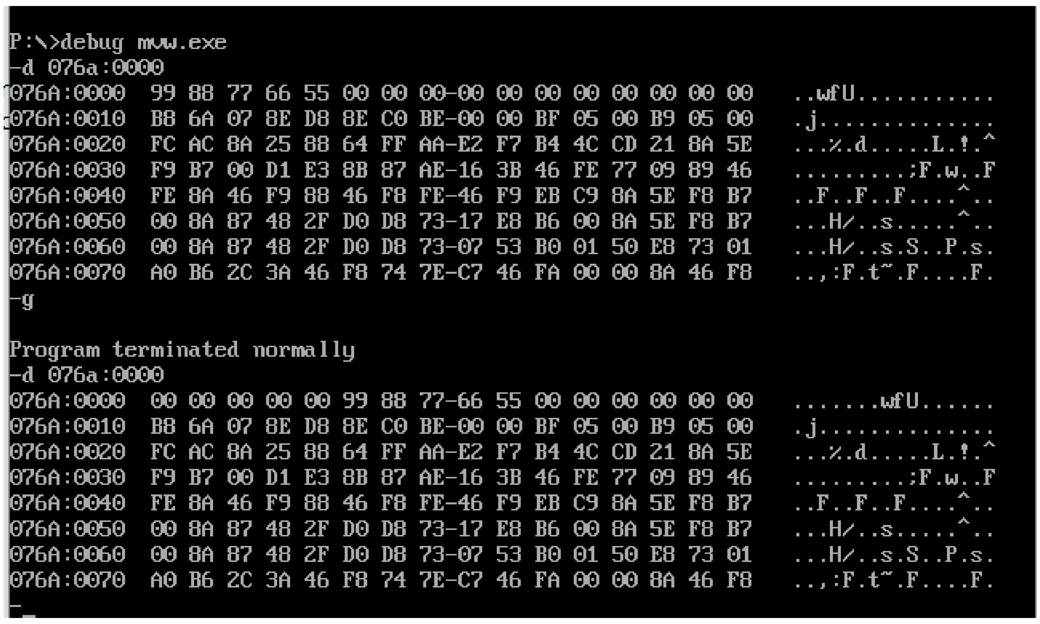
Program	Comments
assume cs:code,ds:data	
data segment string db 99h,88h,77h,66h,55h; loc db ?; data ends	Initialize data segment and variables End data segment

code segment
org 0000h;
start:
mov ax,data;
mov ds,ax;
mov es,ax;
mov si,offset string;
mov di,offset loc;
mov cx,05h;
cld;
looping:lodsb
mov ah,[di];
mov [si-1],ah;
stosb
loop looping
mov ah,4ch;
int 21h;
code ends
end start
	
	start code segment
	Transfer address of data segment to ds
	
	Initialize extra segment
	Initialize the source and destination index registers
	
	Initialize the counter
	
	Clear direction flag
	Data pointed by SI is loaded into AL and increment SI
	Data pointed by DI is loaded into AH
	Data is stored in location pointed by Sl-1
	Data is stored in location pointed by DI and increment DI
	Repeat until the entire string is transferred
	Termination of Execution
	End of code segment
	Terminate program



```
P:\>debug mvw.exe
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 8EC0          MOV     ES,AX
076B:0007 BE0000        MOV     SI,0000
076B:000A BF0500        MOV     DI,0005
076B:000D B90500        MOV     CX,0005
076B:0010 FC            CLD
076B:0011 AC            LODSB
076B:0012 8A25          MOV     AH,[DI]
076B:0014 8864FF        MOV     [SI-01],AH
076B:0017 AA            STOSB
076B:0018 E2F7          LOOP    0011
076B:001A B44C          MOV     AH,4C
076B:001C CD21          INT     21
076B:001E 0000          ADD     [BX+SI],AL
```

Figure 7: Move without string instructions - unassembled



```
P:\>debug mvw.exe
-d 076a:0000
076A:0000  99 88 77 66 55 00 00 00-00 00 00 00 00 00 00 00   ..wfU...........
076A:0010  B8 6A 07 8E D8 8E C0 BE-00 00 BF 05 00 B9 05 00   .j..............
076A:0020  FC AC 8A 25 88 64 FF AA-E2 F7 B4 4C CD 21 8A 5E   ...%.d.....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

Program terminated normally
-d 076a:0000
076A:0000  00 00 00 00 00 99 88 77-66 55 00 00 00 00 00 00   .......wfU......
076A:0010  B8 6A 07 8E D8 8E C0 BE-00 00 BF 05 00 B9 05 00   .j..............
076A:0020  FC AC 8A 25 88 64 FF AA-E2 F7 B4 4C CD 21 8A 5E   ...%.d.....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.
-_
```

Figure 8: Move without string instructions

4 RESULT:

Thus,8086 programs for arithmetic operations of strings like move, search and compare have been executed successfully using MS - DOSBox.