String Manipulations

Expt No: 2 **Name:** Mahesh Bharadwaj K
Date : 10/09/2020 **Reg No:** 185001089

Aim:

To write and execute 8086 programs for string manipulation operations.

Procedure:

- Mount masm folder to a drive on DOSBOX.
- Navigate to mounted drive using 'dir' .
- Save 8086 program with the extension **'.asm'** in the same folder using the command **'edit'**.
- Assemble the **.asm** file using the command **'masm filename.asm'**.
- Link the assmebled **.obj** file using the command **'link filename.obj'**.
- Debug the executable file **.exe** with the **'debug filename.exe'** command.

  i. **U:** To view the un-assembled code.
  ii. **D:** Used as 'D segment:offset' to see the content of memory locations starting from segment:offset address.
  iii. **E:** To change the values in memory.
  iv. **G:** Execute the program using command.
  v. **Q** exits from the debug session.

Algorithm:

1. Moving string of bytes

∗ Move the data segment address to the AX register and then MOVe it to the DS register.

∗ Move the extra segment address to the AX register and then MOVe it to the ES register.

∗ Initially set the CX to length of array using MOV CX, count

∗ Initialise SI using LEA SI, str1

∗ Initialise DI using LEA DI, str2

∗ Move bytes one by one using REPE MOVSB

2. Comparing 2 Strings of bytes

* Move the data segment address to the AX register and then MOVe it to the DS register.
* Move the extra segment address to the AX register and then MOVe it to the ES register.
* Initially set the CX to length of array using MOV CX, count
* Increment CX using INC CX
* Clear Direction Flag using CLD
* Initialise SI using LEA SI, str1
* Initialise DI using LEA DI, str2
* Compare strings bytes usings REPE CMPSB
* Move value of CX into result.

3. Searching a byte in a string

* Move the data segment address to the AX register and then MOVe it to the DS register.
* Move the extra segment address to the AX register and then MOVe it to the ES register.
* Initially set the CX to length of array using MOV CX, count
* Increment CX using INC CX
* Clear Direction Flag using CLD
* Initialise DI using LEA SI, seq
* Move value to search into AL
* Compare bytes in string with AL using REPNE SCASB
* Move CX into result.

4. Moving a string without using string instructions

- ∗ Move the data segment address to the AX register and then MOVe it to the DS register.
- ∗ Move the extra segment address to the AX register and then MOVe it to the ES register.
- ∗ Initially set the CX to length of array using MOV CX, count
- ∗ Initialise SI using LEA SI, str1
- ∗ Initialise DI using LEA DI, str2
- ∗ Move value at SI into BL
- ∗ Move value at BL into DI
- ∗ increment SI using INC SI
- ∗ increment DI using INC DI
- ∗ Loop till CX becomes 0

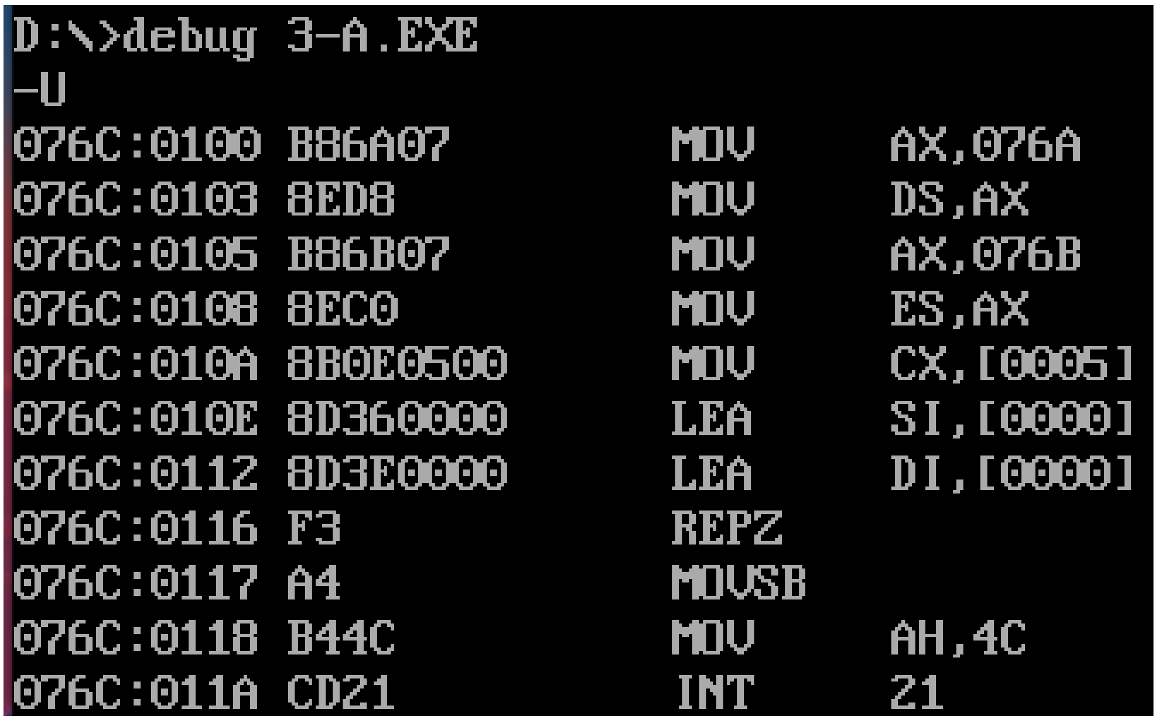
---

1. Moving string of bytes

Program:

Program	**Comments**
start: MOV AX,data	Move data segment address contents to AX register
MOV ds,AX	Move data in AX register to DS register
MOV AX, extra	Move extra segment address to AX register
MOV es, AX	Move data in AX register to ES register
MOV CX,count	Move length of array into CX register
LEA str1, SI	Load effective address of str1 into SI
LEA str1, DI	Load effective address of str2 into DI
CLD	To clear direction flag to increment SI & DI
REPE MOVSB	Moves bytes from string at SI to DI till CX becomes 0
MOV ah,4ch	
int 21h	Request interrupt routine

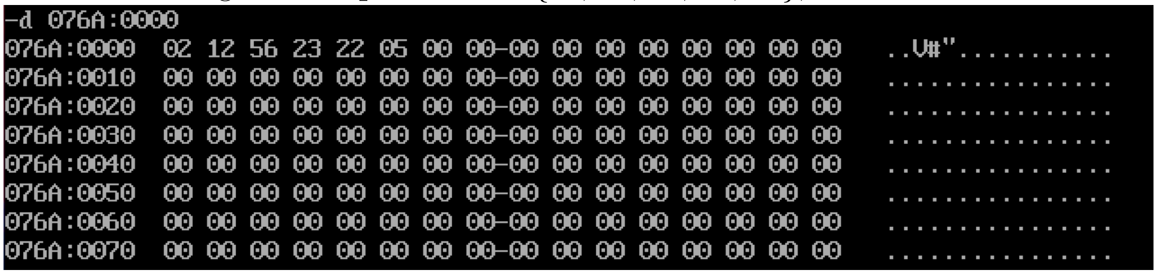
Unassembled Code:



```
D:\>debug 3-A.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 8B0E0500      MOV     CX,[0005]
076C:010E 8D360000      LEA     SI,[0000]
076C:0112 8D3E0000      LEA     DI,[0000]
076C:0116 F3            REPZ
076C:0117 A4            MOVSB
076C:0118 B44C          MOV     AH,4C
076C:011A CD21          INT     21
```

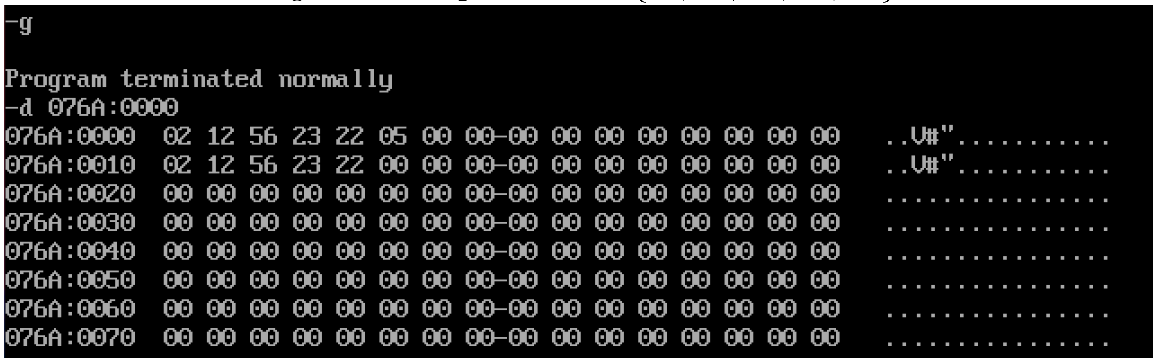
Input and Output:

Figure 1: **Input:** str1 = {02, 12, 56, 23, 22}, count = 05



```
-d 076A:0000
076A:0000  02 12 56 23 22 05 00 00-00 00 00 00 00 00 00 00   ..V#"...........
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: **Output:** str2 = {02, 12, 56, 23, 22}



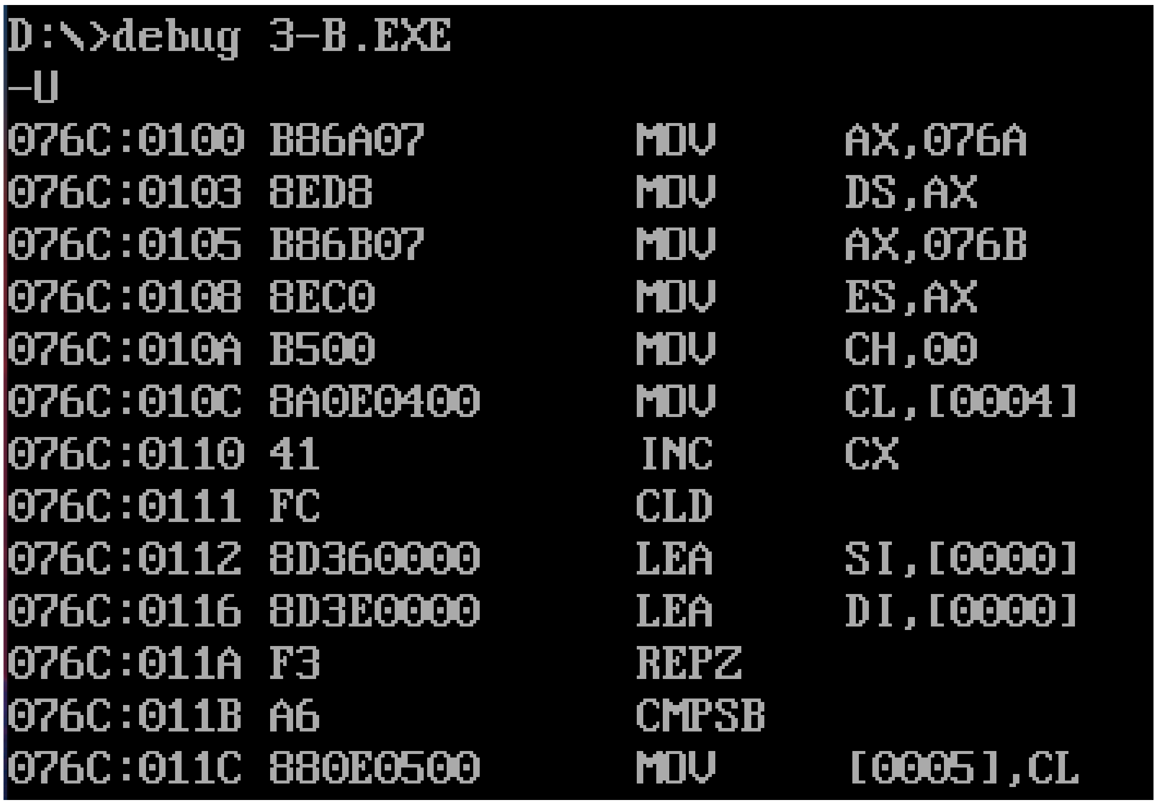
```
-g

Program terminated normally
-d 076A:0000
076A:0000  02 12 56 23 22 05 00 00-00 00 00 00 00 00 00 00   ..V#"...........
076A:0010  02 12 56 23 22 00 00 00-00 00 00 00 00 00 00 00   ..V#"...........
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   ................
```

Comparing 2 string of bytes

Program:

Program	**Comments**
start: MOV AX,data	Move data segment address contents to AX register
MOV ds,AX	Move data in AX register to DS register
MOV AX, extra	Move extra segment address to AX register
MOV es, AX	Move data in AX register to ES register
MOV CX,count	Move length of array into CX register
INC CX	Increment count value
LEA str1, SI	Load effective address of str1 into SI
LEA str1, DI	Load effective address of str2 into DI
CLD	Clear Direction Flag to Increment SI & DI
REPE CMPSB	Compares bytes of array pointed by SI & DI till different values obtained or CX becomes 0
MOV result,CL	Move CL value into result
MOV ah,4ch	
int 21h	Request interrupt routine

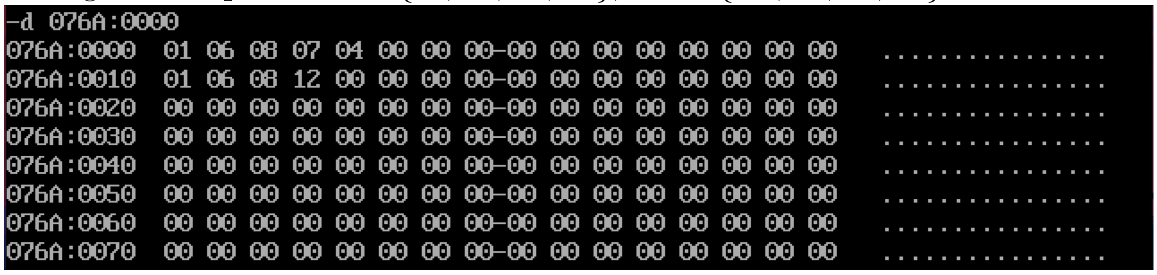
Unassembled Code:



```
D:\>debug 3-B.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 B500          MOV     CH,00
076C:010C 8A0E0400      MOV     CL,[0004]
076C:0110 41            INC     CX
076C:0111 FC            CLD
076C:0112 8D360000      LEA     SI,[0000]
076C:0116 8D3E0000      LEA     DI,[0000]
076C:011A F3            REPZ
076C:011B A6            CMPSB
076C:011C 880E0500      MOV     [0005],CL
```

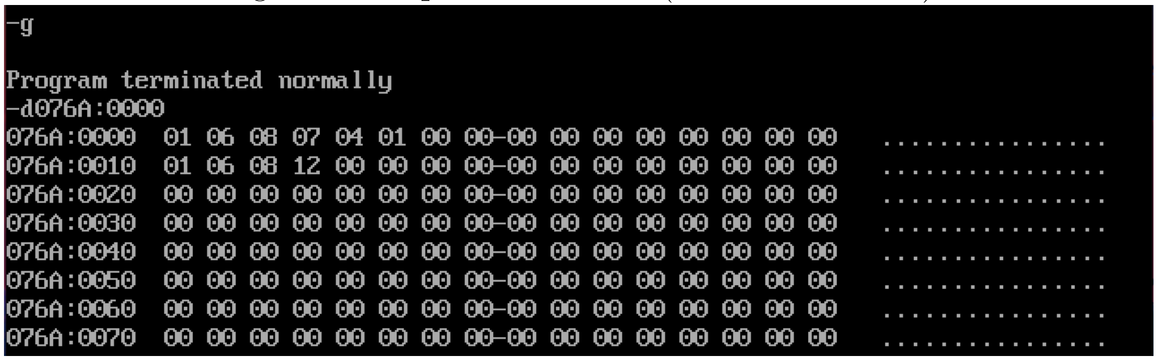
Input and Output:

Figure 3: **Input:** str1 = {01, 06, 08, 07}, str2= { 01, 06, 08, 12} count = 04



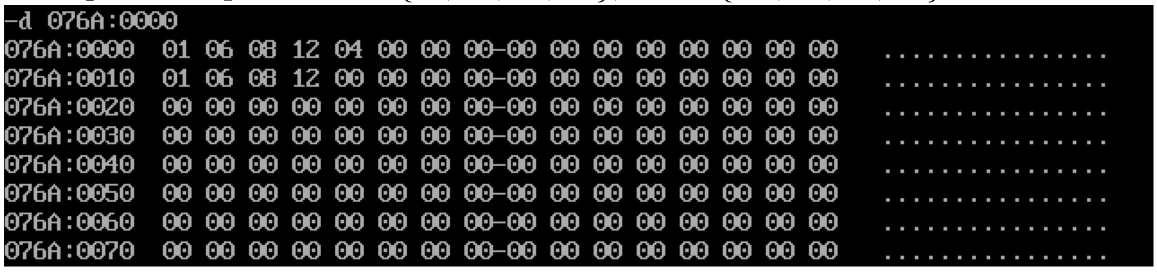
```
-d 076A:0000
076A:0000  01 06 08 07 04 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  01 06 08 12 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 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 4: **Output:** result = 01 (index of difference)



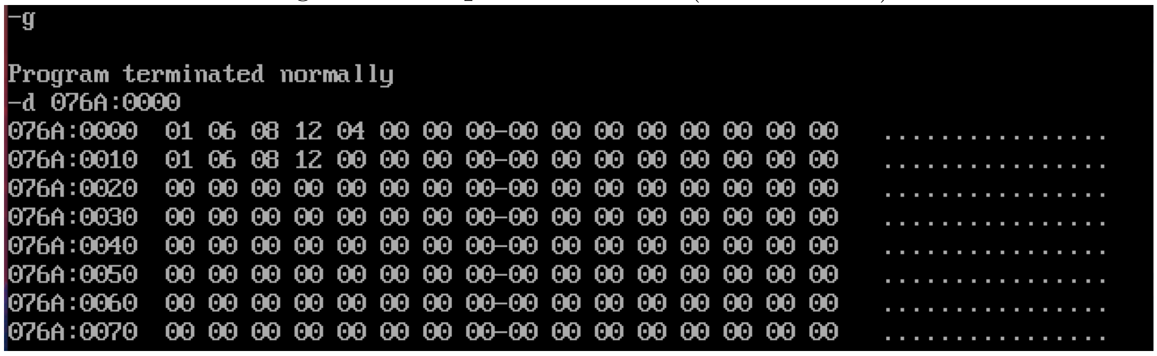
```
-g

Program terminated normally
-d076A:0000
076A:0000  01 06 08 07 04 01 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  01 06 08 12 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 5: **Input:** str1 = {01, 06, 08, 12}, str2= { 01, 06, 08, 12} count = 04



```
-d 076A:0000
076A:0000  01 06 08 12 04 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  01 06 08 12 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 6: **Output:** result = 0 (No difference)



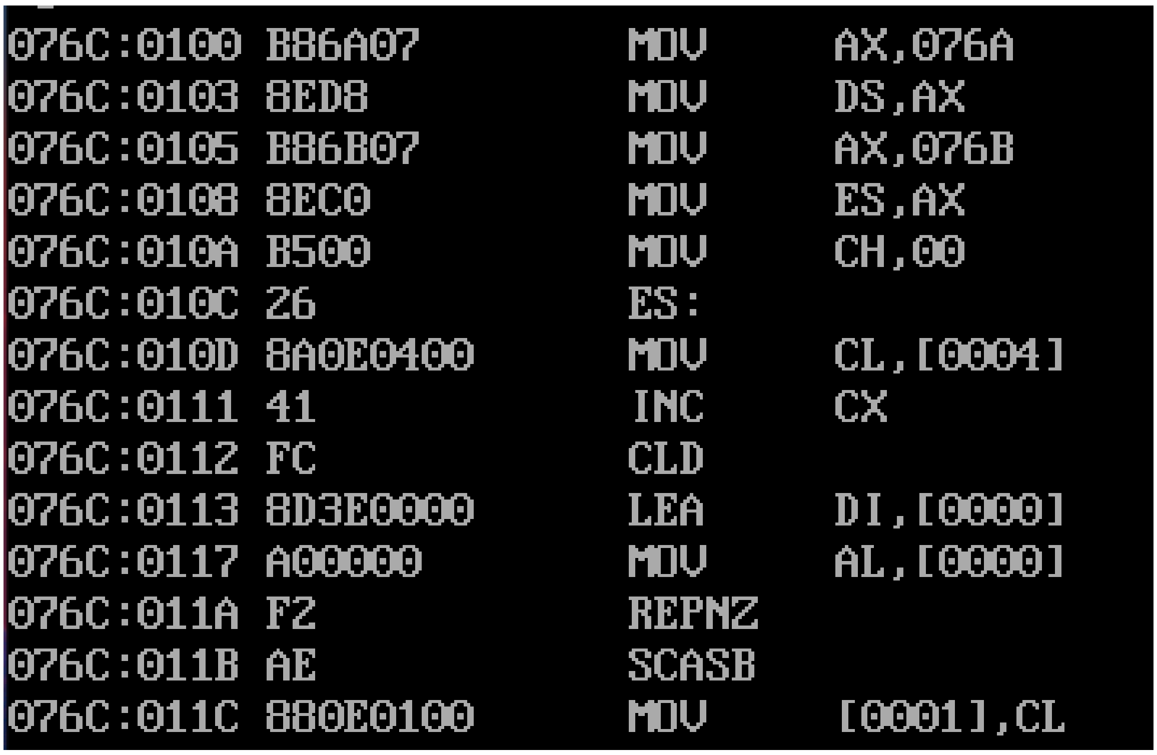
```
-g

Program terminated normally
-d 076A:0000
076A:0000  01 06 08 12 04 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  01 06 08 12 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   ................
```

Searching for a byte in a string

Program:

Program	Comments
start: MOV AX,data	Move data segment address contents to AX register
MOV ds,AX	Move data in AX register to DS register
MOV AX, extra	Move extra segment address to AX register
MOV es, AX	Move data in AX register to ES register
MOV CX,count	Move length of array into CX register
INC CX	Increment count value
CLD	Clear Direction Flag to Increment SI & DI
LEA seq, DI	Load effective address of seq into DI
MOV AL, val	Move value to search for into AL register
REPNE SCASB	Compares bytes of array pointed by DI with AL register till equal value obtained or CX becomes 0
MOV result,CL	Move CL value into result
MOV ah,4ch	
int 21h	Request interrupt routine

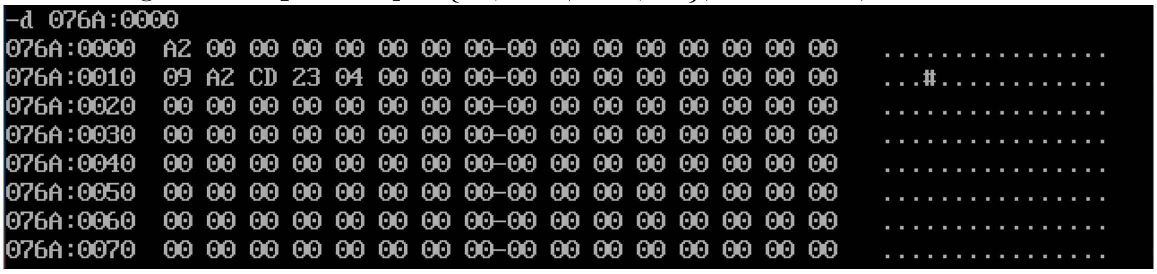
Unassembled Code:



```
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 B500          MOV     CH,00
076C:010C 26            ES:
076C:010D 8A0E0400      MOV     CL,[0004]
076C:0111 41            INC     CX
076C:0112 FC            CLD
076C:0113 8D3E0000      LEA     DI,[0000]
076C:0117 A00000        MOV     AL,[0000]
076C:011A F2            REPNZ
076C:011B AE            SCASB
076C:011C 880E0100      MOV     [0001],CL
```

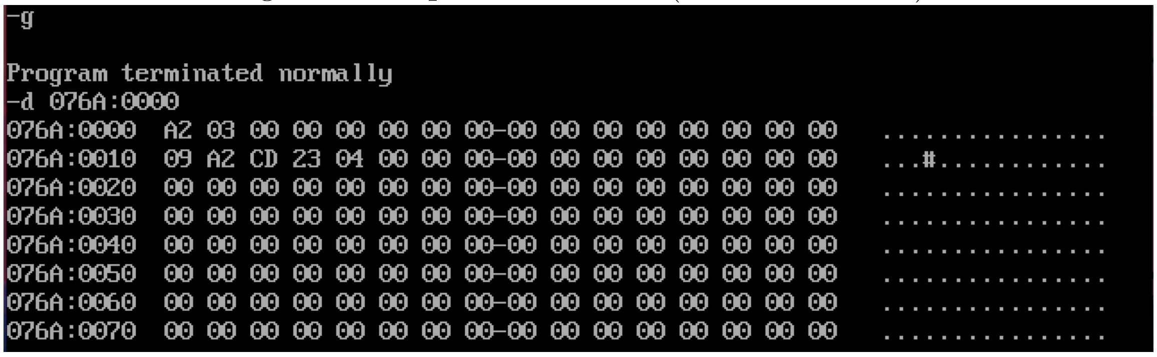
Input and Output:

Figure 7: **Input:** seq = {09, 0A2,0CD, 23}, count = 4, value = 0A2



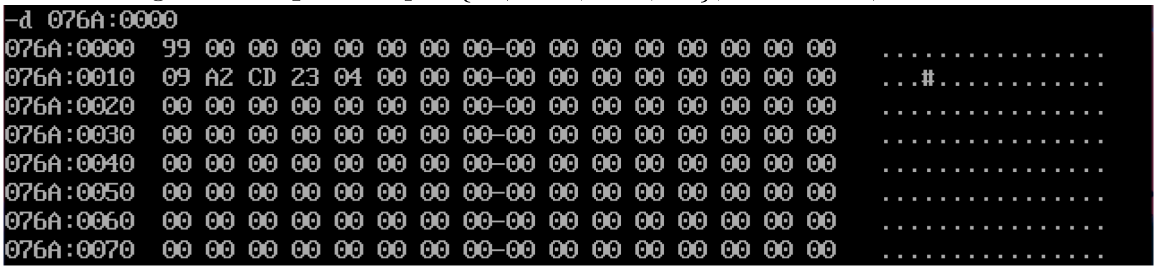
```
-d 076A:0000
076A:0000  A2 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  09 A2 CD 23 04 00 00 00-00 00 00 00 00 00 00 00   ...#............
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 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 8: **Output:** result = 03 (Found at index 3)



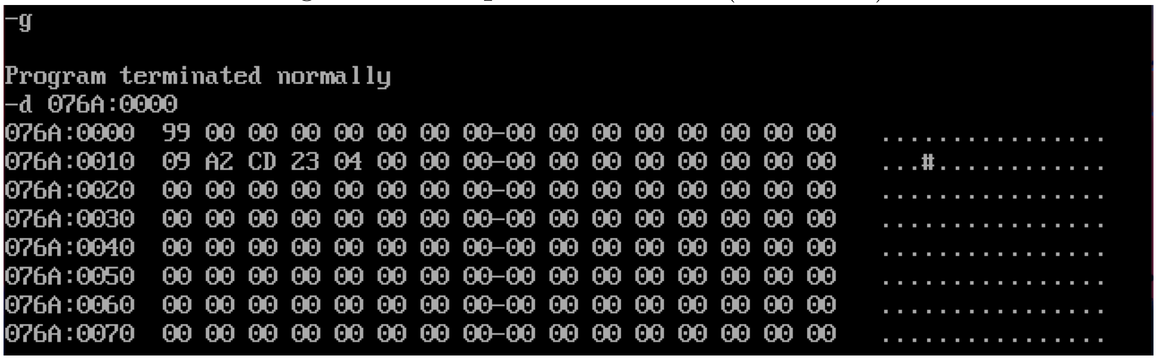
```
-g

Program terminated normally
-d 076A:0000
076A:0000  A2 03 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  09 A2 CD 23 04 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 9: **Input:** seq = {09, 0A2,0CD, 23}, count = 4, value = 99



```
-d 076A:0000
076A:0000  99 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  09 A2 CD 23 04 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 10: **Output:** result = 00 (Not found)



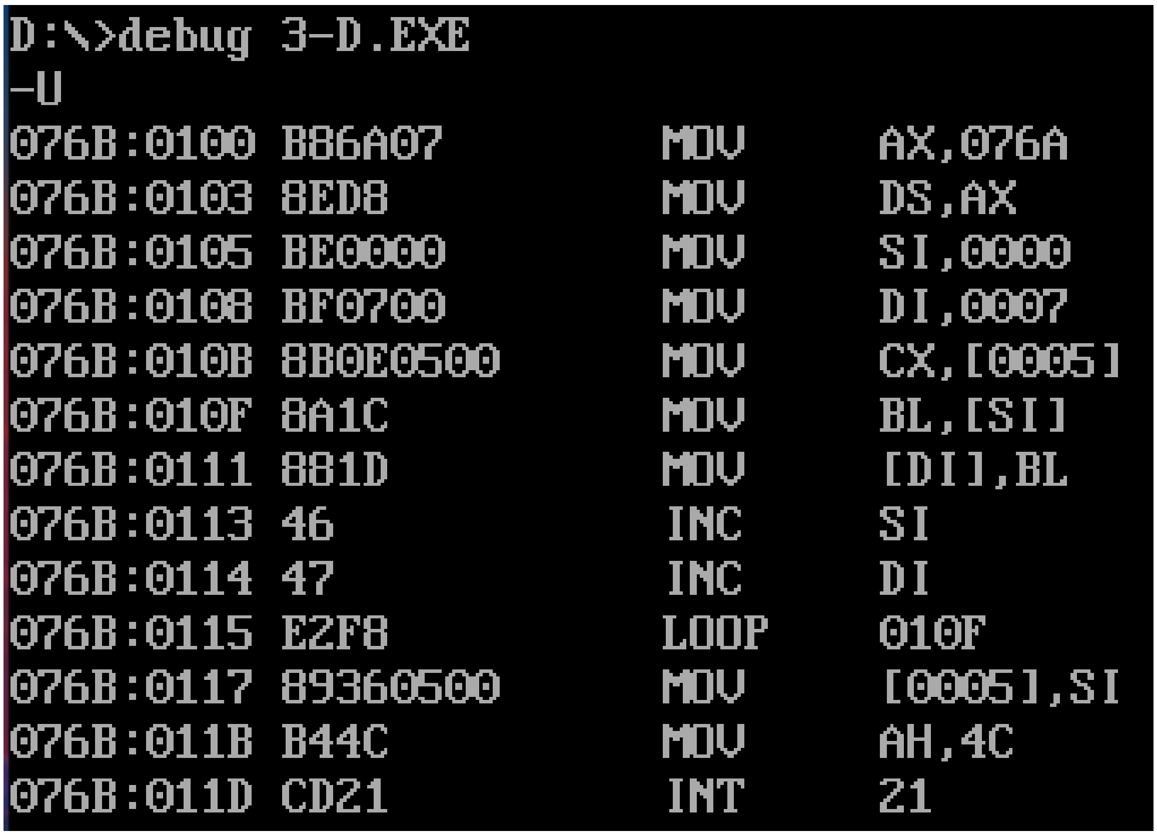
```
-g

Program terminated normally
-d 076A:0000
076A:0000  99 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  09 A2 CD 23 04 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   ................
```

Moving a string without string instructions

Program:

Program	Comments
start: MOV AX,data	Move data segment address contents to AX register
MOV ds,AX	Move data in AX register to DS register
MOV AX, extra	Move extra segment address to AX register
MOV es, AX	Move data in AX register to ES register
MOV CX,count	Move length of array into CX register
LEA arr1, SI	Load effective address of arr1 into SI
LEA arr1, DI	Load effective address of arr2 into DI
here: MOV BL, [SI]	Move value at location pointed by SI Into BL register
MOV [DI], BL	Move value at BL register into location pointed by DI
INC SI	Increment SI register
INC DI	Increment DI register
LOOP here	Loop to *here* till CX becomes 0
MOV ah,4ch	
int 21h	Request interrupt routine

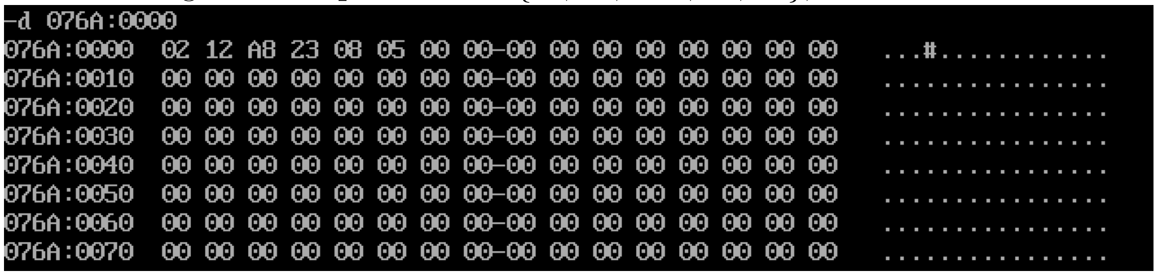
Unassembled Code:



```
D:\>debug 3-D.EXE
-U
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 BE0000        MOV     SI,0000
076B:0108 BF0700        MOV     DI,0007
076B:010B 8B0E0500      MOV     CX,[0005]
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 89360500      MOV     [0005],SI
076B:011B B44C          MOV     AH,4C
076B:011D CD21          INT     21
```

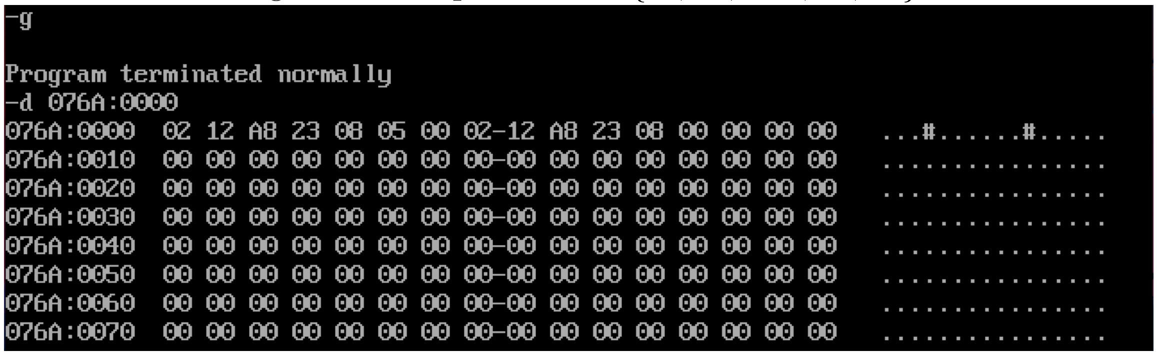
Input and Output:

Figure 11: **Input:** arr1 = {02, 12, 0A8, 23, 08}, count = 0005



```
-d 076A:0000
076A:0000  02 12 A8 23 08 05 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 12: **Output:** arr2 = {02, 12, 0A8, 23, 08}



```
-g

Program terminated normally
-d 076A:0000
076A:0000  02 12 A8 23 08 05 00 02-12 A8 23 08 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:

8086 ASL programs for string manipulations like moving a string of bytes, comparing two strings of bytes, searching a byte in a string using string instructions and also to move a string without using string instructions have been executed successfully using MS - DOSBox.