Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

Exercise 3 – String Manipulations
3A – Moving a string of bytes

Aim:

To move a string from source to location using string manipulation instructions.

Procedure for executing MASM:

1. Mount the local folder in the DOS-BOX using a temp disk name: `mount <disk-name> <folder-location>`
2. Change directory into the mounted disk: `<disk-name>: `
3. Assemble the instructions: `masm <file-name>.asm`
4. Link the object file(s) to produce an executable file(.exe): `link <file-name>.obj; ` Note that removal of semi-colon will make linking process interactive.
5. Debug the executable file to read the memory map and execute the program: `debug <file-name>.exe`. After entering debug mode,
    a. `d <segment:offset> ` - dump(read) memory map from the given location
    b. `e <segment:offset> ` - edit memory values from the given location. Use 'White space' to continue editing and 'new line' to exit editing.
    c. `u ` - unassemble code (with or without <segment:offset>)
    d. `g ` - execute the program
    e. `? ` - display command list
    f. `q` - quit the debugger

Algorithm:

1. Initialise data and extra segment using their respective registers.
2. Load the base address of data segment(ds) and extra segment(es) using an intermediate accumulator register(ax) as direct memory transfer is not allowed in 8086.
3. Load the words, count of bytes in the string to cx.
4. Load the source(si) and destination index(di) register with the offset values of src_string and dest_string from their respective segments, i.e., base address of the two strings respectively.
5. Clear the direction flag, so that SI and DI can auto-increment.
6. Copy byte from DS:SI to ES:DI. Check if CX != 0, then increment SI and DI and decrement CX. Repeat this till CX = 0.
7. Terminate the program.

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

Program:

Program	Comment
`; Program to move strings`	Comment after ';'
`assume cs:code, ds:data, es:extra`	Map CS to code segment, DS to data segment and ES to extra segment

`data segment`
`  words dw 0005H`
`  src_string db 41H, 70H, 70H, 6CH, 65H`
`data ends`
`extra segment`
`  dest_string db 00H, 00H, 00H, 00H, 00H`
`extra ends`
	Initialise data segment and extra segment
	db = define a byte, dw = define a byte
	Initialise words(word), src_string(string), dest_string(string)

`code segment`
`start:  mov ax, data`
`  mov ds, ax`
`  mov ax, extra`
`  mov es, ax`
	Initialise code segment
	Move the starting address of data segment in ax, then move ax to ds; starting address of extra segment in ax, then move ax to es.
	Since in 8086, only code segment register is loaded automatically, the remaining segment register can be assigned using general purpose registers.

`  mov cx, words`
`  mov si, offset src_string`
`  mov di, offset dest_string`
	Load cx register with words
	Load si and di index registers with the address of the src and dest strings.

`  cld`	Clear the direction flag.
`  rep movsb`	Copy byte from DS:SI to ES:DI. Check if CX != 0, then increment SI and DI and decrement CX. Repeat this till CX = 0.

`  mov ah, 4cH`
`  int 21H`
`code ends`
`end start`
	Set ah = 4cH
	Call interrupt routine 21H for DOS, which terminates if ah = 4cH

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

Unassembled code:

```
D:\>debug movstr.exe
-u
076C:0000 B86A07        MOV     AX,076A
076C:0003 8ED8          MOV     DS,AX
076C:0005 B86B07        MOV     AX,076B
076C:0008 8EC0          MOV     ES,AX
076C:000A 8B0E0000      MOV     CX,[0000]
076C:000E BE0200        MOV     SI,0002
076C:0011 BF0000        MOV     DI,0000
076C:0014 FC            CLD
076C:0015 F3            REPZ
076C:0016 A4            MOVSB
076C:0017 B44C          MOV     AH,4C
076C:0019 CD21          INT     21
```

Snapshot of sample input and output:

Before execution:

```
-d 076a:0000
076A:0000  05 00 41 70 70 6C 65 00-00 00 00 00 00 00 00 00   ..Apple.........
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  B8 6A 07 8E D8 B8 6B 07-8E C0 8B 0E 00 00 BE 02   .j....k.........
076A:0030  00 BF 00 00 FC F3 A4 B4-4C CD 21 FE 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.
-d 076b:0000
076B:0000  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076B:0010  B8 6A 07 8E D8 B8 6B 07-8E C0 8B 0E 00 00 BE 02   .j....k.........
076B:0020  00 BF 00 00 FC F3 A4 B4-4C CD 21 FE 77 09 89 46   ........L.!.w..F
076B:0030  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076B:0040  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076B:0050  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076B:0060  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
076B:0070  B4 00 B1 05 D3 E0 03 06-B4 2C 89 46 FC 8A 1E B6   .........,.F....
```

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

After execution:

```
-g

Program terminated normally
-d 076b:0000
076B:0000  41 70 70 6C 65 00 00 00-00 00 00 00 00 00 00 00   Apple...........
076B:0010  B8 6A 07 8E D8 B8 6B 07-8E C0 8B 0E 00 00 BE 02   .j....k.........
076B:0020  00 BF 00 00 FC F3 A4 B4-4C CD 21 FE 77 09 89 46   ........L.!.w..F
076B:0030  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076B:0040  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076B:0050  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076B:0060  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
076B:0070  B4 00 B1 05 D3 E0 03 06-B4 2C 89 46 FC 8A 1E B6   .........,.F....
```

Result:

Program to move a string from source to destination using string instruction is assembled, executed and verified.

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

3B – Comparing two strings of bytes

Aim:

To compare two strings of bytes.

Algorithm:

1. Initialise data and extra segment using their respective registers.
2. Load the base address of data segment(ds) and extra segment(es) using an intermediate accumulator register(ax) as direct memory transfer is not allowed in 8086.
3. Load the bytes, count of bytes [length of string + 1: to differentiate between the mismatch at last index case and perfect match case] in the string to cx.
4. Load the source(si) and destination index(di) register with the offset values of src_string and dest_string from their respective segments, i.e., base address of the two strings respectively.
5. Clear the direction flag, so that SI and DI can auto-increment.
6. Compare byte from DS:SI and ES:DI. Check if CX != 0, then compare bytes by subtracting, if zero flag is set, then increment SI and DI and decrement CX. Repeat this till CX = 0 or ZF = 0.
7. Load status with the value of CX, this will contain the index of first mismatch counted from back, if not, it will contain 0000H.
8. Terminate the program.

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

Program:

Program	Comment
`; Program to compare two strings`	Comment after ';'
`assume cs:code, ds:data, es:extra`	Map CS to code segment, DS to data segment and ES to extra segment

`data segment`
`  bytes dw 0004H`
`  src_string db 62H, 65H, 67H`
`  org 0020H`
`  status dw 0000H`
`data ends`
`extra segment`
`  dest_string db 62H, 65H, 74H`
`extra ends`
	Initialise data segment and extra segment
	db = define a byte, dw = define a byte
	Initialise bytes(word), status(word), src_string(string), dest_string(string)

`code segment`
`start:  mov ax, data`
`  mov ds, ax`
`  mov ax, extra`
`  mov es, ax`
	Initialise code segment
	Move the starting address of data segment in ax, then move ax to ds; starting address of extra segment in ax, then move ax to es.
	Since in 8086, only code segment register is loaded automatically, the remaining segment register can be assigned using general purpose registers.

`  mov cx, bytes`
`  mov si, offset src_string`
`  mov di, offset dest_string`
	Load cx register with bytes
	Load si and di index registers with the address of the src and dest strings.

`  cld`	Clear the direction flag.
`  repe cmpsb`	Compare byte from DS:SI and ES:DI. Check if CX != 0, then compare bytes by subtracting, if zero flag is set, then increment SI and DI and decrement CX. Repeat this till CX = 0 or ZF = 0.
`  mov status, cx`	Move cx to status. Contains index of first mismatch or 0000H

`  mov ah, 4cH`
`  int 21H`
`code ends`
`end start`
	Set ah = 4cH
	Call interrupt routine 21H for DOS, which terminates if ah = 4cH

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

Unassembled code:

```
D:\>debug cmpstr.exe
-u
076E:0000 B86A07        MOV     AX,076A
076E:0003 8ED8          MOV     DS,AX
076E:0005 B86D07        MOV     AX,076D
076E:0008 8EC0          MOV     ES,AX
076E:000A 8B0E0000      MOV     CX,[0000]
076E:000E BE0200        MOV     SI,0002
076E:0011 BF0000        MOV     DI,0000
076E:0014 FC            CLD
076E:0015 F3            REPZ
076E:0016 A6            CMPSB
076E:0017 890E2000      MOV     [0020],CX
076E:001B B44C          MOV     AH,4C
076E:001D CD21          INT     21
```

Snapshot of sample input and output:

Case i: Unequal Strings
Before Execution:

```
-d 076a:0000
076A:0000  04 00 62 65 67 00 00 00-00 00 00 00 00 00 00 00   ..beg...........
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  62 65 74 00 00 00 00 00-00 00 00 00 00 00 00 00   bet.............
076A:0040  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BE 02   .j....m.........
076A:0050  00 BF 00 00 FC F3 A6 89-0E 20 00 B4 4C CD 21 B7   ......... ..L.!.
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.
-d 076d:0000
076D:0000  62 65 74 00 00 00 00 00-00 00 00 00 00 00 00 00   bet.............
076D:0010  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BE 02   .j....m.........
076D:0020  00 BF 00 00 FC F3 A6 89-0E 20 00 B4 4C CD 21 B7   ......... ..L.!.
076D:0030  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076D:0040  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
076D:0050  B4 00 B1 05 D3 E0 03 06-B4 2C 89 46 FC 8A 1E B6   .........,.F....
076D:0060  2C B7 00 D1 E3 FF B7 FA-15 9A 00 00 A0 04 89 DE   ,...............
076D:0070  06 C4 7E FA B9 00 01 5A-1E 8E DA FC F2 A5 1F 8A   ..~....Z........
```

After Execution: 076a:0020 = 01H -> Strings are **different** at index 1 from back, i.e., 3 from the front

```
-g

Program terminated normally
-d 076a:0000
076A:0000  04 00 62 65 67 00 00 00-00 00 00 00 00 00 00 00   ..beg...........
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  62 65 74 00 00 00 00 00-00 00 00 00 00 00 00 00   bet.............
076A:0040  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BE 02   .j....m.........
076A:0050  00 BF 00 00 FC F3 A6 89-0E 20 00 B4 4C CD 21 B7   ......... ..L.!.
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.
```

Case ii: Equal Strings
Before Execution:

```
-e 076d:0000
076D:0000  62.     65.     74.67

-d 076a:0000
076A:0000  04 00 62 65 67 00 00 00-00 00 00 00 00 00 00 00   ..beg...........
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  62 65 67 00 00 00 00 00-00 00 00 00 00 00 00 00   beg.............
076A:0040  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BE 02   .j....m.........
076A:0050  00 BF 00 00 FC F3 A6 89-0E 20 00 B4 4C CD 21 B7   ......... ..L.!.
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.
-d 076d:0000
076D:0000  62 65 67 00 00 00 00 00-00 00 00 00 00 00 00 00   beg.............
076D:0010  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BE 02   .j....m.........
076D:0020  00 BF 00 00 FC F3 A6 89-0E 20 00 B4 4C CD 21 B7   ......... ..L.!.
076D:0030  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076D:0040  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
076D:0050  B4 00 B1 05 D3 E0 03 06-B4 2C 89 46 FC 8A 1E B6   .........,.F....
076D:0060  2C B7 00 D1 E3 FF B7 FA-15 9A 00 00 A0 04 89 DE   ,...............
076D:0070  06 C4 7E FA B9 00 01 5A-1E 8E DA FC F2 A5 1F 8A   ..~....Z........
```

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

After Execution: 076a:0020 = 00H -> Strings are **same**

```
-g

Program terminated normally
-d 076a:0000
076A:0000  04 00 62 65 67 00 00 00-00 00 00 00 00 00 00 00   ..beg...........
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  62 65 67 00 00 00 00 00-00 00 00 00 00 00 00 00   beg.............
076A:0040  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BE 02   .j....m.........
076A:0050  00 BF 00 00 FC F3 A6 89-0E 20 00 B4 4C CD 21 B7   ......... ..L.!.
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.
```

Result:

Program to compare two strings of bytes is assembled, executed and verified.

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

3C – Searching a byte in a string

Aim:

To search for a byte in the given string

Algorithm:

1. Initialise data and extra segment using their respective registers.
2. Load the base address of data segment(ds) and extra segment(es) using an intermediate accumulator register(ax) as direct memory transfer is not allowed in 8086.
3. Load the bytes, count of bytes [length of string + 1: to differentiate between the key occurrence at last index case and no occurrence case] in the string to cx.
4. Load the destination index(di) register with the offset values of search_string, i.e., base address of the string.
5. Load al with key to be searched in search_string.
6. Clear the direction flag, so that SI and DI can auto-increment.
7. Compare the byte from AL with ES:DI. Check if CX != 0, then compare bytes by subtracting, if zero flag is not set, then increment SI and DI and decrement CX. Repeat this till CX = 0 or ZF != 0.
8. Load status with the value of CX, this will contain the index of first occurrence counted from back, if no occurrence, it will contain 00H.
9. Terminate the program.

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

Program:

Program	Comment
`; Program to search for byte`	Comment after ';'
`; in a string`	
`assume cs:code, ds:data, es:extra`	Map CS to code segment, DS to data segment and ES to extra segment
`data segment`	Initialise data segment and extra segment
`bytes dw 0004H`	db = define a byte, dw = define a byte
`key db 01H`	Initialise bytes(word), status(word), src_string(string), dest_string(string)
`org 0020H`	
`status dw 0000H`	
`data ends`	
`extra segment`	
`search_string db 01H, 02H, 03H`	
`extra ends`	
`code segment`	Initialise code segment
`start: mov ax, data`	Move the starting address of data segment in ax, then move ax to ds; starting address of extra segment in ax, then move ax to es. Since in 8086, only code segment register is loaded automatically, the remaining segment register can be assigned using general purpose registers.
`mov ds, ax`	
`mov ax, extra`	
`mov es, ax`	
`mov cx, bytes`	Load cx register with bytes
`mov di, offset search_string`	Load di index register with the address of the search_string.
`mov al, key`	Load al with the byte to search, key
`cld`	Clear the direction flag.
`repne scasb`	Compare the byte from AL with ES:DI. Check if CX != 0, then compare bytes by subtracting, if ZF = 0, then increment SI and DI and decrement CX. Repeat this till CX = 0 or ZF != 0.
`mov status, cx`	Move cx to status. Contains index of first mismatch or 0000H
`mov ah, 4cH`	Set ah = 4cH
`int 21H`	Call interrupt routine 21H for DOS, which terminates if ah = 4cH
`code ends`	
`end start`	

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

Unassembled code:

```
D:\>debug SCANSTR.EXE
-u
076E:0000 B86A07        MOV     AX,076A
076E:0003 8ED8          MOV     DS,AX
076E:0005 B86D07        MOV     AX,076D
076E:0008 8EC0          MOV     ES,AX
076E:000A 8B0E0000      MOV     CX,[0000]
076E:000E BF0000        MOV     DI,0000
076E:0011 A00200        MOV     AL,[0002]
076E:0014 FC            CLD
076E:0015 F2            REPNZ
076E:0016 AE            SCASB
076E:0017 890E2000      MOV     [0020],CX
076E:001B B44C          MOV     AH,4C
076E:001D CD21          INT     21
```

Snapshot of sample input and output:

Case i: String containing key
Before execution:

```
-d 076a:0000
076A:0000  04 00 01 00 00 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  01 02 03 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BF 00   .j....m.........
076A:0050  00 A0 02 00 FC F2 AE 89-0E 20 00 B4 4C CD 21 B7   ........ ..L.!.
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.
-d 076d:0000
076D:0000  01 02 03 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076D:0010  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BF 00   .j....m.........
076D:0020  00 A0 02 00 FC F2 AE 89-0E 20 00 B4 4C CD 21 B7   ........ ..L.!.
076D:0030  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076D:0040  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
076D:0050  B4 00 B1 05 D3 E0 03 06-B4 2C 89 46 FC 8A 1E B6   .........,.F....
076D:0060  2C B7 00 D1 E3 FF B7 FA-15 9A 00 00 A0 04 89 DE   ,...............
076D:0070  06 C4 7E FA B9 00 01 5A-1E 8E DA FC F2 A5 1F 8A   ..~....Z........
```

After execution: 076a:0020 = 03H -> Key **found** at index 3 from back, i.e., 1 from the front

```
-g

Program terminated normally
-d 076a:0000
076A:0000  04 00 01 00 00 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  03 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  01 02 03 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BF 00   .j....m.........
076A:0050  00 A0 02 00 FC F2 AE 89-0E 20 00 B4 4C CD 21 B7   ......... ..L.!.
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.
```

Case ii: String not containing key
Before execution:

```
-e 076a:0000
076A:0000  04.     00.     01.ff

-d 076a:0000
076A:0000  04 00 FF 00 00 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  01 02 03 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BF 00   .j....m.........
076A:0050  00 A0 02 00 FC F2 AE 89-0E 20 00 B4 4C CD 21 B7   ......... ..L.!.
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.
-d 076d:0000
076D:0000  01 02 03 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076D:0010  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BF 00   .j....m.........
076D:0020  00 A0 02 00 FC F2 AE 89-0E 20 00 B4 4C CD 21 B7   ......... ..L.!.
076D:0030  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076D:0040  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
076D:0050  B4 00 B1 05 D3 E0 03 06-B4 2C 89 46 FC 8A 1E B6   .........,.F....
076D:0060  2C B7 00 D1 E3 FF B7 FA-15 9A 00 00 A0 04 89 DE   ,...............
076D:0070  06 C4 7E FA B9 00 01 5A-1E 8E DA FC F2 A5 1F 8A   ..~....Z........
```

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

After execution: 076a:0020 = 00H -> Key **not found**

```
-g

Program terminated normally
-d 076a:0000
076A:0000  04 00 FF 00 00 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  01 02 03 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  B8 6A 07 8E D8 B8 6D 07-8E C0 8B 0E 00 00 BF 00   .j....m.........
076A:0050  00 A0 02 00 FC F2 AE 89-0E 20 00 B4 4C CD 21 B7   ........ ..L.!.
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.
```

Result:

Program to search a byte in the given string is assembled, executed and verified.

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

3D – moving a string without using string instructions

Aim:

To move a string from source to location without using string manipulation instructions.

Algorithm:

1. Initialise data and extra segment using their respective registers.
2. Load the base address of data segment(ds) and extra segment(es) using an intermediate accumulator register(ax) as direct memory transfer is not allowed in 8086.
3. Load the bytes, count of bytes in the string to cx.
4. Load the source(si) and destination index(di) register with the offset values of src and dest from their respective segments, i.e., base address of the two strings respectively.
5. Label ‘here’ for this stub. Load al with the value at si using [si]. Move the value from al to es:di using es:[di].
6. Increment SI and DI.
7. Loop to ‘here’ label till CX = 0.
8. Terminate the program.

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

Program:

Program	Comment

`; Move a string`
`; without string operations`
 	
`assume cs: code, ds: data, es: extra`
 	
`data segment`
`  count dw 0003H`
`  src db 62H, 65H, 67H`
`data ends`
`extra segment`
`  dest db 00H, 00H, 00H`
`extra ends`
 	
`code segment`
`start:  mov ax, data`
`  mov ds, ax`
`  mov ax, extra`
`  mov es, ax`
 	
`  mov cx, count`
`  mov si, offset src`
`  mov di, offset dest`
 	
`here:`
`  mov al, [si]`
`  mov es:[di], al`
`  inc si`
`  inc di`
 	
`  loop here`
 	
`  mov ah, 4cH`
`  int 21H`
`code ends`
`end start`
	Comment after ‘;’
	
	Map CS to code segment, DS to data segment and ES to extra segment
	
	Initialise data segment and extra segment
	db = define a byte, dw = define a byte
	Initialise count(word), src(string), dest(string)
	
	Initialise code segment
	Move the starting address of data segment in ax, then move ax to ds; starting address of extra segment in ax, then move ax to es.
	Since in 8086, only code segment register is loaded automatically, the remaining segment register can be assigned using general purpose registers.
	
	Load cx register with count
	Load si and di index registers with the address of the src and dest strings.
	
	Move [si] to al
	Move al to es:[di]
	Increment si
	Increment di
	
	Decrement cx. If cx != 0, jump to ‘here’.
	
	Set ah = 4cH
	Call interrupt routine 21H for DOS, which terminates if ah = 4cH

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

Unassembled code:

```
D:\>debug movbes.exe
-u
076C:0000 B86A07        MOV     AX,076A
076C:0003 8ED8          MOV     DS,AX
076C:0005 B86B07        MOV     AX,076B
076C:0008 8EC0          MOV     ES,AX
076C:000A 8B0E0000      MOV     CX,[0000]
076C:000E BE0200        MOV     SI,0002
076C:0011 BF0000        MOV     DI,0000
076C:0014 8A04          MOV     AL,[SI]
076C:0016 26            ES:
076C:0017 8805          MOV     [DI],AL
076C:0019 46            INC     SI
076C:001A 47            INC     DI
076C:001B E2F7          LOOP    0014
076C:001D B44C          MOV     AH,4C
076C:001F CD21          INT     21
```

Snapshot of sample input and output:

Before execution:

```
-d 076a:000
076A:0000  03 00 62 65 67 00 00 00-00 00 00 00 00 00 00 00   ..beg...........
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  B8 6A 07 8E D8 B8 6B 07-8E C0 8B 0E 00 00 BE 02   .j....k.........
076A:0030  00 BF 00 00 8A 04 26 88-05 46 47 E2 F7 B4 4C CD   ......&..FG...L.
076A:0040  21 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.
-d 076b:0000
076B:0000  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076B:0010  B8 6A 07 8E D8 B8 6B 07-8E C0 8B 0E 00 00 BE 02   .j....k.........
076B:0020  00 BF 00 00 8A 04 26 88-05 46 47 E2 F7 B4 4C CD   ......&..FG...L.
076B:0030  21 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   !.F..F..F....^..
076B:0040  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076B:0050  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076B:0060  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
076B:0070  B4 00 B1 05 D3 E0 03 06-B4 2C 89 46 FC 8A 1E B6   .........,.F....
```

Aravind Kannan Rathinasabapathi
195001022

Date: 30-07-2021
Ex. No. 3

After execution:

```
-g

Program terminated normally
-d 076b:0000
076B:0000  62 65 67 00 00 00 00 00-00 00 00 00 00 00 00 00   beg.............
076B:0010  B8 6A 07 8E D8 B8 6B 07-8E C0 8B 0E 00 00 BE 02   .j....k.........
076B:0020  00 BF 00 00 8A 04 26 88-05 46 47 E2 F7 B4 4C CD   ......&..FG...L.
076B:0030  21 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   !.F..F..F....^..
076B:0040  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076B:0050  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076B:0060  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
076B:0070  B4 00 B1 05 D3 E0 03 06-B4 2C 89 46 FC 8A 1E B6   .........,.F....
```

Result:

Program to move a string from source to destination without using string instruction is assembled, executed and verified.