

EXPERIMENT NO- 5

AIM: WAP based on string instruction to transfer a block and to find string length.

Resource Required: P-IV and above RAM 128MB, Dot Matrix Printer, Emu 8086, MASM 611/ TASM, Turbo C/C++, Printer, Printout Stationary.

THEORY:

String Instructions:

String is a group of bytes/words and their memory is always allocated in a sequential order.

Following is the list of instructions under this group –

1. REP – Used to repeat the given instruction till $CX \neq 0$.
2. MOVS/MOVSb/MOVSW – Used to move the byte/word from one string to another.
3. COMS/COMPSb/COMPSW – Used to compare two string bytes/words.
4. SCAS/SCASb/SCASW – Used to scan a string and compare its byte with a byte in AL or string word with a word in AX.
5. LODS/LODSb/LODSW – Used to store the string byte into AL or string word into AX.

ALGORITHM:

a) To transfer a block of data (non-overlapping block)

Step I : Initialize the data in the source and destination memory

Step II : Initialize SI and DI with source and destination address

Step III : Initialize CX with count

Step IV : Initialize DF flag to zero

Step V : Transfer the data byte from source to destination

Step VI : Repeat above step till $CX \neq 0$

Step VII : Display the result.

Step VIII : Stop.

b) To find the length of String

Step I : Initialize the data segment

Step II : Initialize the len variable to 0

Step III : Point the starting element of the list by SI register

Step IV : Compare [si] with '\$'

Step V : If equal go to VIII step else step V

Step VI : Increment len

Step VII : Increment SI

Step VIII : Display the length

Step IX End

CONCLUSION: We have successfully a)Transfer data from one block to another

b)Calculated length of a string

in Assembly Language using EMU8086.

Code and Output:

a)Transferring block of data:

The screenshot displays the EMU8086 assembly environment with the following components:

- Assembly Code (Left Panel):**

```

01 data segment
02     str1 db 02h,07h,99h,55h
03 data ends
04
05 extra segment
06     str2 db 04 dup(?)
07 extra ends
08
09
10 code segment
11     assume cs: code ds: data es: extra
12     Start:
13         mov ax,data
14         mov ds,ax
15         mov ax,extra
16         mov es,ax
17         lea si,str1
18         lea di,str2
19
20         mov cx,04h
21         cld
22         rep movsb
23
24         int 03h
25
26
27 code ends
28     End Start
29

```
- Registers (Bottom Left):**

Register	H	L
AX	07	11
BX	00	00
CX	00	00
DX	00	00
CS	F400	
IP	0104	
SS	0710	
SP	FFFA	
BP	0000	
SI	0004	
DI	0004	
DS	0710	
ES	0711	
- Memory (Bottom Center):**

Address	Value	Comment
F4100	FF 255	RES
F4101	FF 255	RES
F4102	CD 205	=
F4103	FF 255	RES
F4104	CF 207	±
F4105	00 000	NULL
F4106	00 000	NULL
F4107	00 000	NULL
F4108	00 000	NULL
F4109	00 000	NULL
F410A	00 000	NULL
F410B	00 000	NULL
F410C	00 000	NULL
F410D	00 000	NULL
F410E	00 000	NULL
F410F	00 000	NULL
F4110	00 000	NULL
F4111	00 000	NULL
F4112	00 000	NULL
F4113	00 000	NULL
F4114	00 000	NULL
F4115	00 000	NULL
F4116	00 000	NULL
F4117	00 000	NULL
F4118	00 000	NULL
F4119	00 000	NULL
F411A	00 000	NULL
F411B	00 000	NULL
F411C	00 000	NULL
F411D	00 000	NULL
- Variables (Top Right):**

Variable	Value
STR1	02h, 07h, 99h, 55h
STR2	02h, 07h, 99h, 55h
- Flags (Middle Right):**

CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	0
DF	0
- Source Code (Bottom Right):**

```

07 extra ends
08
09
10 code segment
11     assume cs: code ds: data
12     Start:
13         mov ax,data
14         mov ds,ax
15         mov ax,extra
16         mov es,ax
17         lea si,str1
18         lea di,str2
19
20         mov cx,04h
21         cld
22         rep movsb
23
24         int 03h
25
26
27
28
29

```

b) String Length: