University of Central Punjab
(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)
FACULTY OF INFORMATION TECHNOLOGY

Computer Organization and Assembly Language

Lab 12	
Topic	1. String operations

PART 1

String Instructions

Instruction	Functionality actually performed	
movsb	1. Mov [ES:DI],[DS:SI] 2. Inc si 3. Inc di	Invalid instruction (memory to memory)
movsw	1. Mov [ES:DI],[DS:SI] 2. Add si,2 3. Add di,2	Invalid instruction (memory to memory)
scasb	1. Cmp al,[ES:DI];ZF=1 if same 2. Inc DI	
scasw	1. Cmp ax,[ES:DI];ZF=1 if same 2. Add DI,2	
cmpsb	1. Cmp [DS:SI],[ES:DI];ZF=1 if same 2. Inc SI 3. Inc DI	Invalid instruction (memory to memory)
cmpsw	1. Cmp [DS:SI],[ES:DI];ZF=1 if same 2. Add si,2 3. Add di,2	Invalid instruction (memory to memory)
lodsb	1. Mov al,[DS:SI] 2. Inc si	
lodsw	1. Mov ax,[DS:SI] 2. Add si,2	
stosb	1. Mov [ES:DI],al 2. Inc di	
stosw	1. Mov [ES:DI],ax 2. Add di,2	
Rep	It repeats the instruction cx times.	
Repe	It executes the instruction cx times or until zf remains 1.	
Repne	It executes the instruction cx times or exit when zf becomes 1.	

Note: All yellow highlighted instructions will depend upon direction flag(cld, std) see second last example.

String Examples

Simple String(Example)

```
[org 0x100]
jmp start

data1 db ‘Abcd,edfg,ijkl,mnopqr’,0

data2: times 30 db 0

start:

mov si, data1

mov di, data2

mov cx, 21

l1:

mov al, [si]

mov [di], al

inc si

inc di

loop l1

mov ax,0x4c00

int 21h
```

movsb(Example)

```
[org 0x100]
jmp start
data1 db ‘Abcd,edfg,ijkl,mnopqr’,0
data2: times 30 db 0
start:
mov si, data1
mov di, data2
mov cx, 21

l1:

movsb

loop l1

mov ax,0x4c00
int 21h
```

Using loop instruction(Example)

```
[org 0x100]
jmp start
data1 db ‘Abcd,edfg,ijkl,mnopqr’,0;
data2: times 100 db 0

start:
mov si, data1
mov di, data2
mov cx, 21
l1:
movsb
loop l1

mov ax,0x4c00

int 21h
```

Using REP instruction(Example)

```
[org 0x100]
jmp start
data1 db ‘Ali,Irfan,Aslam,Imran’,0;
data2: times 100 db 0

start:
mov si, data1
mov di, data2
mov cx, 21
REP MOVSB
mov ax,0x4c00

int 21h
```

Using SCAS instruction(Example)

```
[org 0x100]
jmp start
STR1 db 'CCComuter',0
start:
mov di, STR1;
MOV AL, 'C';
MOV CX, 8;
REPE SCASB
;this code runs till zf remain 1.
;keep in mind the functionality of rep and
repe is different
```

Using CMPS instruction(Example)

```
[org 0x100]
jmp start
STR1 db 'comiputer',0
STR2 db 'computer',0
start:
mov di, STR1;
mov si, STR2;
MOV CX, 7;
REPE CMPSB
;this code runs till comparison between two
strings is giving zf=1.
;keep in mind the functionality of rep and
repe is different
```

Using LODSB instruction(Example)

```
[org 0x100]
jmp start
STR1 db 'AbCd123',0
STR2 db 'AbCD123',0
count db 0
start:
mov di, STR1;
Mov si, STR2;
MOV CX, 7;
L1:
LODSB
SCASB
je L2
jne L3
L2:
inc byte [count]
L3:
loop L1
mov ax,0x4c00
int 21h
;calculating how many characters same.
```

Using STOSB instruction(Example)

```
[org 0x100]
jmp start
STR1 db 'Assembly',0
STR2 times 8 db 0
start:
Mov si, STR1;
Mov di, STR2;
MOV CX, 7;
L1:
LODSB
STOSB

loop L1
;making copy of a string.
```

Traversing array from left to right

```
[org 0x100]
mov si,array1
mov cx,17
cld    ;reset the direction flag
        ;increments the si and di in string
operations
rep lodsb

mov ax,0x4c00
int 21h
array1 db '14 January, 2019.'
```

Traversing array from right to left

```
[org 0x100]
mov si,array1
mov cx,17

add si,16        ;to get the address of last
character in the string.
std    ;set direction flag decrements the si
and di in string operations

rep lodsb


mov ax,0x4c00
int 21h
array1 db '14 January, 2019.'
```

Using string operations with video memory.

```
[org 0x100]
jmp start
data1 db ‘Hello world’;
data2: times 11 db 0
start:
mov si, data1
mov di, data2
mov cx, 11

l1:

movsb

loop l1

mov cx,21
mov ax,0xb800
mov es,ax
mov si,data1
mov di,0
mov ah,0x3f
label1:
        lodsb
        stosw
        loop label1
mov ax,0x4c00
int 21h
```

Display a Number on screen.

```
[org 0x100]
jmp start
printnum:
push bp
mov bp, sp
mov ax,[bp+4]
mov cx,0
mov bx, 10
Divide:
 ; use base 10 for division
mov dx,0
div bx              ; remainder goes to DX and quotient goes to AX
push dx

inc cx
cmp ax,0
jnz Divide
Display:
 mov ax, 0xb800;
 Mov es, ax;
 mov di, 0;
l1:
pop ax
add al,0x30
mov ah, 0x07;                          ; Attribute byte, use any number
```

```
Mov [es:di],ax;                    ; printing message on the screen;
Add di,2;
loop l1

pop bp
ret 2

start:
mov ax, 1234   ;it is one thousand 2 hundred and thirty four (decimal).
push ax ; place number on stack
call printnum                      ; call the printnum subroutine


mov ax,0x4c00
int 21h
```