NAME:- KAUSHAL VASHISTH ROLL NO. 18ETCS002147

Laboratory 7

Title of the Laboratory Exercise: String manipulation

1. Introduction and Purpose of Experiment

   Students will be able to perform all string manipulations in assembly language

2. Aim and Objectives

   Aim

   To develop assembly language program to perform all string operations like inserting a byte, deleting a byte and copying a string as a sub-string

   Objectives

   At the end of this lab, the student will be able to

   – Identify instructions for performing string manipulation
   – Use indexed addressing mode
   – Apply looping instructions in assembly language
   – Use data segment to represent arrays

3. Experimental Procedure

   1. Write algorithm to solve the given problem
   2. Translate the algorithm to assembly language code
   3. Run the assembly code in GNU assembler
   4. Create a laboratory report documenting the work

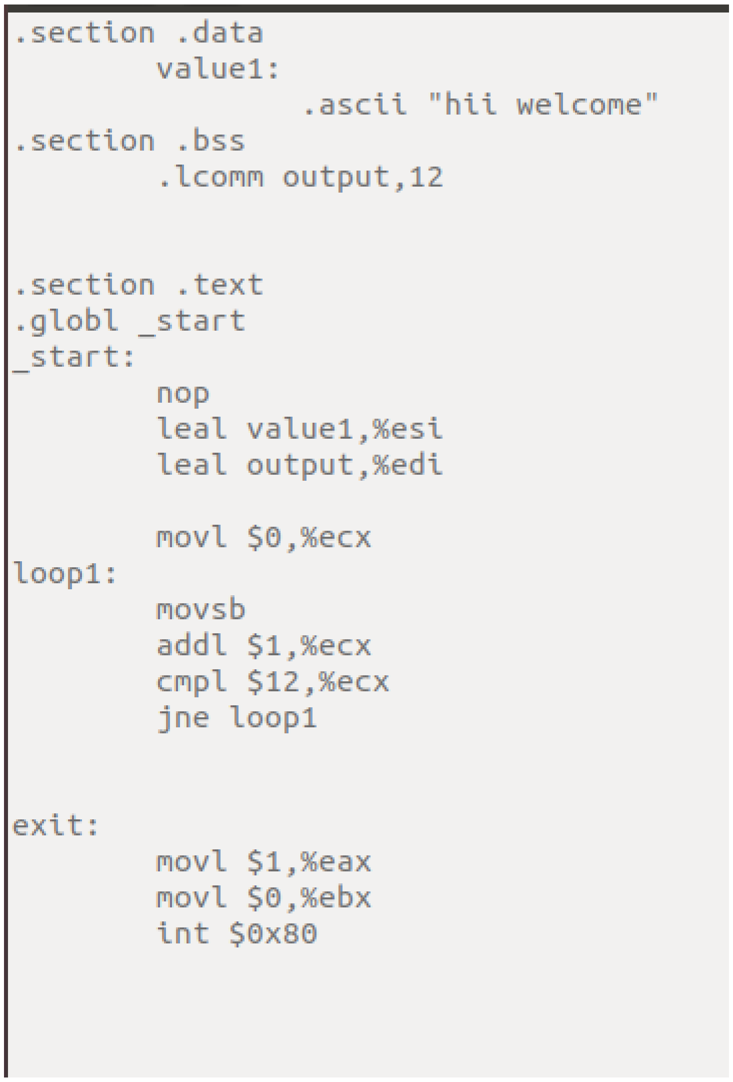
4. Questions

   Develop an assembly language program to perform the following

   1. Copy the contents of MSG1 to MSG2
   2. Copy the contents of MSG1 to MSG3 in reverse order

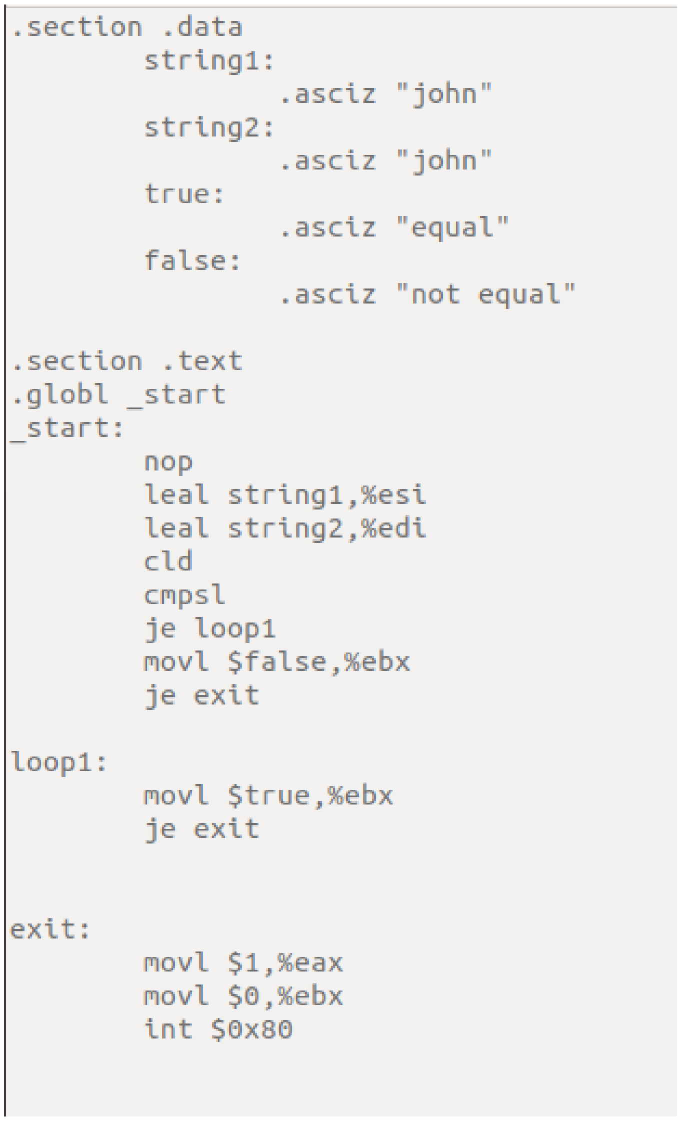
3. Develop an assembly language program to compare two strings and print a message “Equal” if they are equal, “Not Equal” if they are not equal.

5. Calculations/Computations/Algorithms



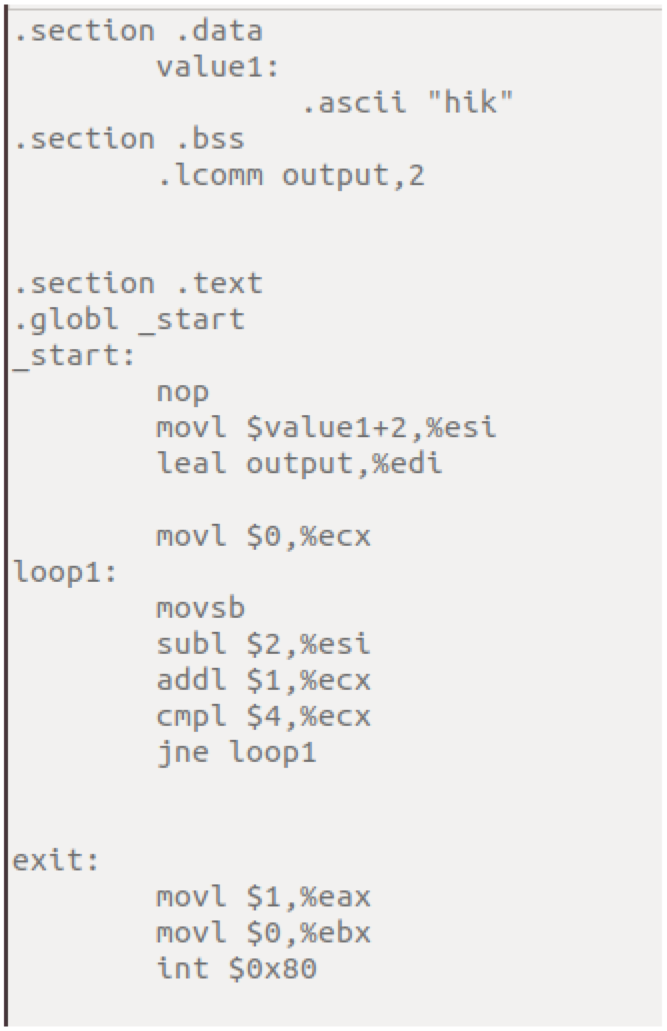
```
.section .data
        value1:
                .ascii "hii welcome"
.section .bss
        .lcomm output,12


.section .text
.globl _start
_start:
        nop
        leal value1,%esi
        leal output,%edi

        movl $0,%ecx
loop1:
        movsb
        addl $1,%ecx
        cmpl $12,%ecx
        jne loop1


exit:
        movl $1,%eax
        movl $0,%ebx
        int $0x80
```

Figure 1:Copy the contents of MSG1 to MSG2



```
.section .data
        string1:
                .asciz "john"
        string2:
                .asciz "john"
        true:
                .asciz "equal"
        false:
                .asciz "not equal"

.section .text
.globl _start
_start:
        nop
        leal string1,%esi
        leal string2,%edi
        cld
        cmpsl
        je loop1
        movl $false,%ebx
        je exit

loop1:
        movl $true,%ebx
        je exit


exit:
        movl $1,%eax
        movl $0,%ebx
        int $0x80
```

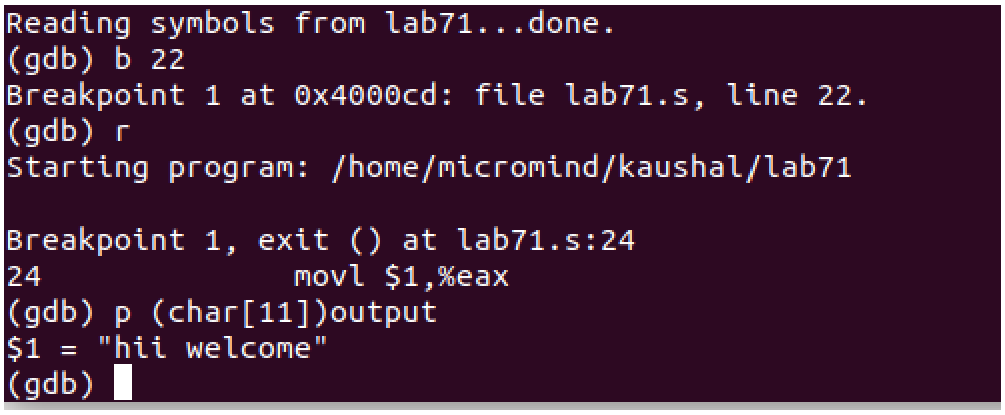
Figure 2: program to compare two strings



```
.section .data
        value1:
                .ascii "hik"
.section .bss
        .lcomm output,2


.section .text
.globl _start
_start:
        nop
        movl $value1+2,%esi
        leal output,%edi

        movl $0,%ecx
loop1:
        movsb
        subl $2,%esi
        addl $1,%ecx
        cmpl $4,%ecx
        jne loop1


exit:
        movl $1,%eax
        movl $0,%ebx
        int $0x80
```

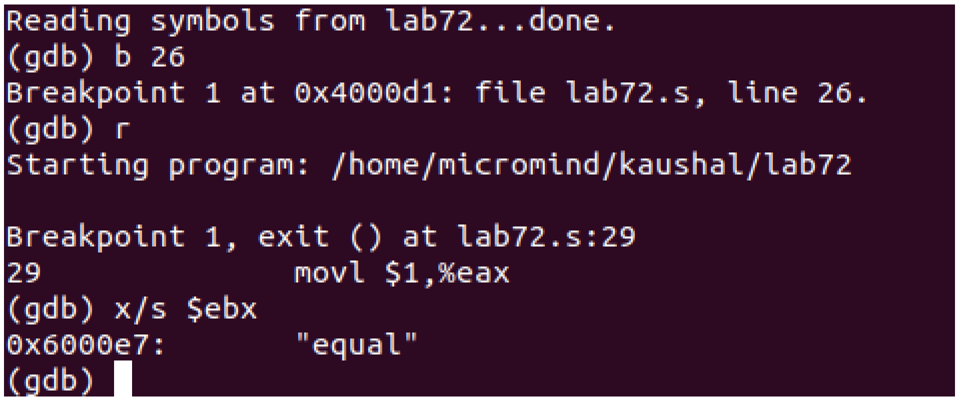
Figure 3:Copy the contents of MSG1 to MSG3 in reverse order

6. Presentation of Results



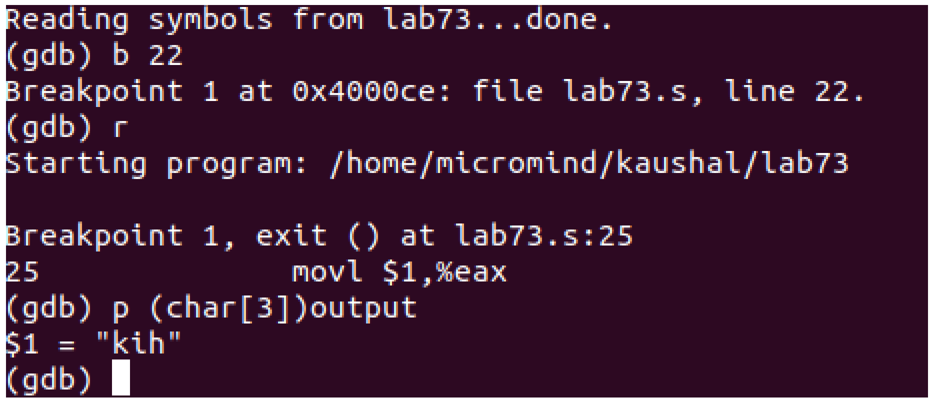
```
Reading symbols from lab71...done.
(gdb) b 22
Breakpoint 1 at 0x4000cd: file lab71.s, line 22.
(gdb) r
Starting program: /home/micromind/kaushal/lab71

Breakpoint 1, exit () at lab71.s:24
24              movl $1,%eax
(gdb) p (char[11])output
$1 = "hii welcome"
(gdb) 
```

Figure 4: results of fig 1



```
Reading symbols from lab72...done.
(gdb) b 26
Breakpoint 1 at 0x4000d1: file lab72.s, line 26.
(gdb) r
Starting program: /home/micromind/kaushal/lab72

Breakpoint 1, exit () at lab72.s:29
29              movl $1,%eax
(gdb) x/s $ebx
0x6000e7:       "equal"
(gdb) 
```

Figure 5: results for figure 2



```
Reading symbols from lab73...done.
(gdb) b 22
Breakpoint 1 at 0x4000ce: file lab73.s, line 22.
(gdb) r
Starting program: /home/micromind/kaushal/lab73

Breakpoint 1, exit () at lab73.s:25
25              movl $1,%eax
(gdb) p (char[3])output
$1 = "kih"
(gdb) 
```

Figure 6: output for figure 3

7. Analysis and Discussions:-

we have learned the following commands in string operation's lab.

Solaris Mnemonic	**Intel/AMD Mnemonic**	**Description**	**Notes**
`cmps{q}`	`CMPS`	compare string	`cmpsq` valid only under **`-xarch=amd64`**
`cmpsb`	`CMPSB`	compare byte string	
`cmpsl`	`CMPSD`	compare doubleword string	
`cmpsw`	`CMPSW`	compare word string	
`lods{q}`	`LODS`	load string	`lodsq` valid only under **`-xarch=amd64`**
`lodsb`	`LODSB`	load byte string	
`lodsl`	`LODSD`	load doubleword string	
`lodsw`	`LODSW`	load word string	
`movs{q}`	`MOVS`	move string	`movsq` valid only under **`-xarch=amd64`**
`movsb`	`MOVSB`	move byte string	`movsb` is not `movsb{wlq}`. See Table 3–1
`movsl`, `smovl`	`MOVSD`	move doubleword string	
`movsw`, `smovw`	`MOVSW`	move word string	`movsw` is not `movsw{lq}`. See Table 3–1
`rep`	`REP`	repeat while `%ecx` not zero	
`repnz`	`REPNE`	repeat while not equal	

Solaris Mnemonic	**Intel/AMD Mnemonic**	**Description**	**Notes**
`repnz`	`REPNZ`	repeat while not zero	
`repz`	`REPE`	repeat while equal	
`repz`	`REPZ`	repeat while zero	
`scas{q}`	`SCAS`	scan string	`scasq` valid only under –**`xarch=amd64`**
`scasb`	`SCASB`	scan byte string	
`scasl`	`SCASD`	scan doubleword string	
`scasw`	`SCASW`	scan word string	
`stos{q}`	`STOS`	store string	`stosq` valid only under –**`xarch=amd64`**
`stosb`	`STOSB`	store byte string	
`stosl`	`STOSD`	store doubleword string	
`stosw`	`STOSW`	store word string	

8. Conclusions :

   successfully developed assembly language programs to perform all string operations like inserting a byte, deleting a byte and copying a string as a sub-string

Signature and date

Marks