

## LABORATORY 5

### TITLE OF THE LABORATORY EXERCISE: SEARCHING AN ELEMENT IN AN ARRAY

#### 1. INTRODUCTION AND PURPOSE OF EXPERIMENT

STUDENTS WILL BE ABLE TO PERFORM SEARCH OPERATIONS IN AN ARRAY OF INTEGERS OR CHARACTERS

#### 2. AIM AND OBJECTIVES

##### AIM

TO DEVELOP ASSEMBLY LANGUAGE PROGRAM TO PERFORM SEARCH OPERATIONS IN AN ARRAY

##### OBJECTIVES

AT THE END OF THIS LAB, THE STUDENT WILL BE ABLE TO

- IDENTIFY INSTRUCTIONS TO BE USED IN ASSEMBLY LANGUAGE
- PERFORM SEARCH OPERATIONS IN ASSEMBLY LANGUAGE

#### 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. SEARCHING AN ELEMENT IN AN ARRAY OF 'N' NUMBERS
2. READ A SENTENCE WITH AT LEAST ONE SPECIAL CHARACTER AND SEARCH FOR THE SPECIAL CHARACTER AND PRINT IT. E.G., CONSIDER THE INPUT {YOUREMAILID@MSRUAS.AC.IN }OUTPUT: @, .
3. DEVELOP AN ASSEMBLY LANGUAGE PROGRAM TO COMPUTE THE PARITY OF A HEXADECIMAL NUMBER STORED IN THE REGISTER1. IF REGISTER1 HAS ODD

NUMBER OF ONES, UPDATE REGISTER2 WITH 0X01. IF REGISTER1 HAS EVEN NUMBER OF ONES, UPDATE REGISTER2 WITH 0X00.

NOTE: REGISTER1 AND REGISTER2 CAN BE ANY GENERAL PURPOSE REGISTERS.

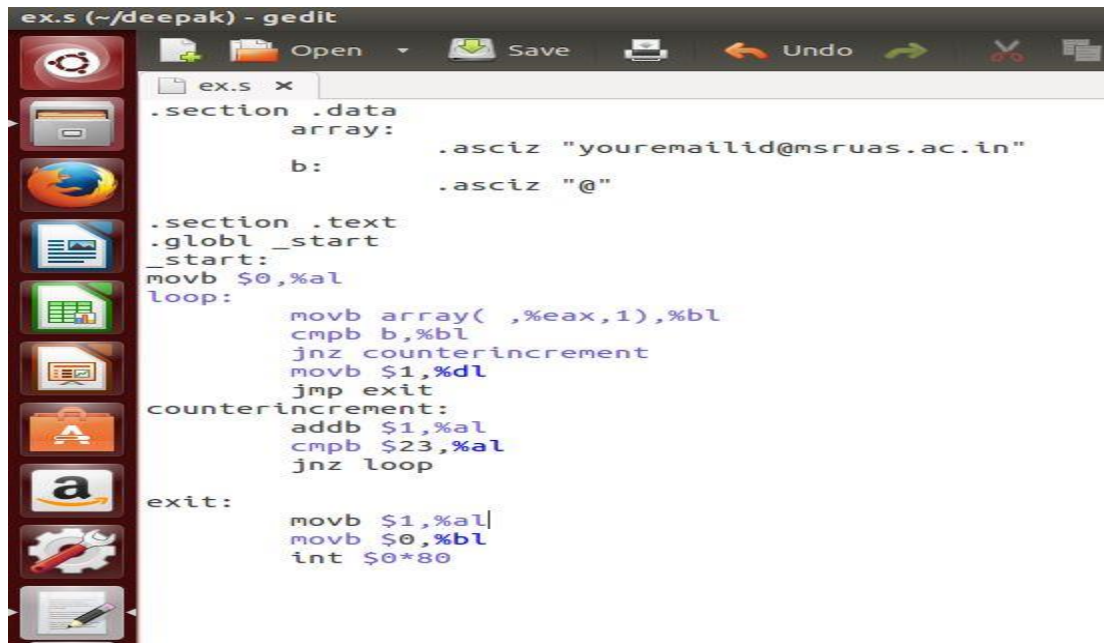
#### 5. CALCULATIONS/COMPUTATIONS/ALGORITHMS



```
ex.s (~/deepak) - gedit
.section .data
    array: .int 1,2,3,4,5,6
    b: .int 5

.section .text
.globl _start
_start:
    movl $0,%eax
loop:
    movl array(,%eax,4),%ebx
    cmpl b,%ebx
    jnz counterincrement
    movl $1,%edx
    jmp exit
counterincrement:
    addl $1,%eax
    cmpl $5,%eax
    jnz loop
exit:
    movl $1,%eax
    movl $0,%ebx
    int $0*80
```

FIG 1 PROGRAM TO SEARCHING AN ELEMENT IN AN ARRAY OF 'N' NUMBERS



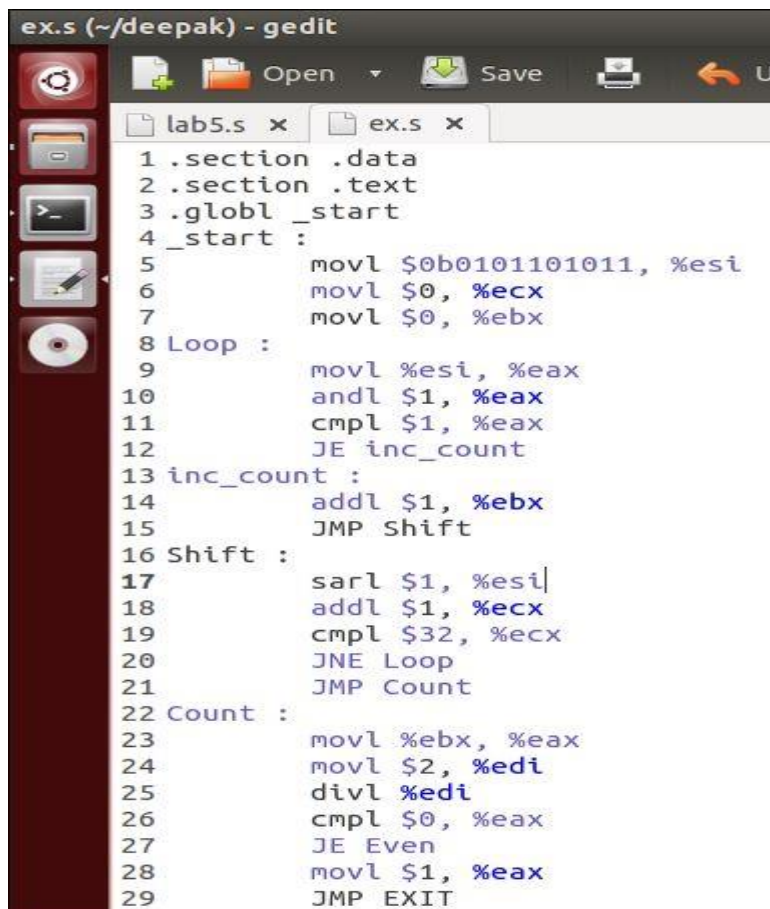
```

ex.s (~/.deepak) - gedit
.section .data
array:
    b:      .asciz "youremailid@msruas.ac.in"
    .asciz "@"

.section .text
.globl _start
_start:
movb $0,%al
loop:
    movb array(,%eax,1),%bl
    cmpb b,%bl
    jnz counterincrement
    movb $1,%dl
    jmp exit
counterincrement:
    addb $1,%al
    cmpb $23,%al
    jnz loop
exit:
    movb $1,%al
    movb $0,%bl
    int $0*80

```

FIG 2 PROGRAM TO READ A SENTENCE WITH AT LEAST ONE SPECIAL CHARACTER AND SEARCH FOR THE SPECIAL CHARACTER AND PRINT IT.



```

ex.s (~/.deepak) - gedit
lab5.s x ex.s x
1 .section .data
2 .section .text
3 .globl _start
4 _start :
5     movl $0b0101101011, %esi
6     movl $0, %ecx
7     movl $0, %ebx
8 Loop :
9     movl %esi, %eax
10    andl $1, %eax
11    cmpl $1, %eax
12    JE inc_count
13 inc_count :
14    addl $1, %ebx
15    JMP Shift
16 Shift :
17    sarl $1, %esi
18    addl $1, %ecx
19    cmpl $32, %ecx
20    JNE Loop
21    JMP Count
22 Count :
23    movl %ebx, %eax
24    movl $2, %edi
25    divl %edi
26    cmpl $0, %eax
27    JE Even
28    movl $1, %eax
29    JMP EXIT

```

```

30 Even :
31      movl $0, %eax
32 EXIT :
33      movl $1, %eax
34      movl $0, %ebx
35      int $0x80

```

FIG 3 PROGRAM AN ASSEMBLY LANGUAGE PROGRAM TO COMPUTE THE PARITY OF A BINARY NUMBER

#### 6. PRESENTATION OF RESULTS

```

mplab@msruas-cse-vbox-ubt: ~/deepak
Breakpoint 1 at 0x8048097: file ex.s, line 23.
(gdb) run
Starting program: /home/mplab/deepak/ex

Breakpoint 1, exit () at ex.s:23
23      movl $1,%eax
(gdb) info registers
eax      0x4      4
ecx      0x0      0
edx      0x1      1
ebx      0x5      5
esp      0xbffff050      0xbffff050
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8048097      0x8048097 <exit>
eflags   0x246     [ PF ZF IF ]
cs       0x73     115
ss       0x7b     123
ds       0x7b     123
es       0x7b     123
fs       0x0      0
gs       0x0      0
(gdb)

```

FIG 4 RESULT OF PROGRAM TO SEARCHING AN ELEMENT IN AN ARRAY OF 'N' NUMBERS

```

(gdb) info registers
eax            0xb      11
ecx            0x0      0
edx            0x1      1
ebx            0x40     64
esp            0xbffff050  0xbffff050
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x804808f  0x804808f <exit>
eflags        0x246    [ PF ZF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
(gdb) print /c b
$3 = 64 '@'
(gdb)

```

FIG 5 RESULT OF PROGRAM TO READ A SENTENCE WITH AT LEAST ONE SPECIAL CHARACTER AND SEARCH FOR THE SPECIAL CHARACTER AND PRINT IT.

```

Starting program: /home/exam/deepak/ex
Breakpoint 1, EXIT () at ex.s:33
33      movl $1, %eax
(gdb) info registers
eax            0x1      1
ecx            0x20     32
edx            0x1      1
ebx            0x5       5
esp            0xbffff050  0xbffff050
ebp            0x0      0x0
esi            0x0      0
edi            0x2       2
eip            0x8048098  0x8048098 <EXIT>
eflags        0x202    [ IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
(gdb)

```

FIG 6 RESULT OF PROGRAM AN ASSEMBLY LANGUAGE PROGRAM TO COMPUTE THE PARITY OF A BINARY NUMBER

## 1. CONCLUSIONS

EXECUTION FLOW CAN BE CONTROLLED BY USING CONDITIONAL INSTRUCTIONS, WHICH INCLUDES A CMP INSTRUCTION FOLLOWED BY A JUMP INSTRUCTION, A CMP INSTRUCTION COMPARES THE TWO OPERANDS AND UPDATES THE FLAG REGISTER, THIS IS THEN USED WITH JUMP INSTRUCTION TO GO TO SOME OTHER PART OF THE PROGRAM, USING THIS WE CAN FORM LOOPING STRUCTURES TO DO STUFF LIKE SEARCHING ELEMENT IN ARRAY, SEARCHING SPECIAL CHARACTER IN ARRAY, AND PROGRAM AN ASSEMBLY LANGUAGE PROGRAM TO COMPUTE THE PARITY OF A BINARY NUMBER

## COMMENTS

### 1. LIMITATIONS OF EXPERIMENTS

ALTHOUGH LOOPING STRUCTURES CAN BE FORMED USING THE CMP, JCC INSTRUCTIONS BUT RECURSIVE STRUCTURES ARE COMPLEX TO FORM USING JUST THESE INSTRUCTIONS.

### 2. LIMITATIONS OF RESULTS

NONE

### 3. LEARNING HAPPENED

WE LEARNT THE USE OF COMPARE, UNCONDITIONAL JUMP AND CONDITIONAL JUMP INSTRUCTIONS TO FORM LOOPING STRUCTURES AND CONDITIONAL STATEMENTS.

### 4. RECOMMENDATIONS

SINCE A PROGRAM CAN CONTAIN NUMEROUS LOOP LABELS, EACH LABEL SHOULD BE CAREFULLY NAMED, AND THE PROGRAMMER MUST KEEP TRACK OF WHICH PARTS OF THE PROGRAM JUMP TO WHERE, ELSE THERE MIGHT BE CHANCES OF FORMING INFINITE LOOPS.

SIGNATURE AND DATE

