Name:- KAUSHAL VASHISTH ROLL NO:- 18ETCS002147

Laboratory 4

Title of the Laboratory Exercise: Controlling execution flow using conditional instructions

1. Introduction and Purpose of Experiment

   Students will be able to perform control flow operations using conditional instructions

2. Aim and Objectives

   Aim

   To develop assembly language program to perform control flow operations using conditional instructions.

   Objectives

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

   - Identify the appropriate assembly language instruction for the given conditional operations
   - Perform all conditional operations using assembly language instructions
   - Get familiar with assembly language program by developing simple programs

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. Print all even numbers in ‘n’ natural numbers
   2. Print all odd numbers in ‘n’ natural numbers
   3. Compute GCD for the given two natural numbers
   4. Compute LCM for the given two natural numbers

5. Develop an assembly language program to generate the first n numbers in Fibonacci series.

5. Calculations/Computations/Algorithms:-

Q1:- to print even numbers:-

Step1:- define array of size=12 with all of its elements as zero in section data

Step2:- start

Step3:-move value '0' to eax and to ecx general purpose registers using "movl" command.

// here register eax will be used as increment of even numbers and ecx wil be used as Index.

Step4:- run a loop:-

Step5:- move value of eax to ecxth position (i.e 1st position) of array.

Step6:- use command "addl $2,%eax " to increment eax value by 2. // to get even numbers

Step7:- use command "addl $1,%ecx " to increment ecx value by 1. // to get index

Step8:- terminate the loop when value of ecx=12.

// all even numbers will be stored in array and we can print it in terminal by array@12. Where 12 is length of array.

Step9:- stop

Q2:- to print odd numbers:-

Step1:- define array of size=12 with all of its elements as zero in section data

Step2:- start

Step3:-move value '3' to eax and '0'to ecx general purpose registers using "movl" command.

// here register eax will be used as increment of odd numbers and ecx wil be used as Index.

Step4:- run a loop:-

Step5:- move value of eax to ecxth position (i.e 1st position) of array.

Step6:- use command "addl $2,%eax " to increment eax value by 2. // to get odd numbers

Step7:- use command "addl $1,%ecx " to increment ecx value by 1. // to get index

Step8:- terminate the loop when value of ecx=12.

// all odd numbers will be stored in array and we can print it in terminal by

array@12. Where 12 is length of array.

Step9:- stop

Q3&4:- to find hcf and lcm of two numbers:-

Step1:- define a=40 and b=18 in section data

Step2:- start

Step3:-move value 'a' to eax and 'b'to ecx general purpose registers using "movl" command.

Step4:- run a loop:-

Step6:- move value '0' to edx register

Step7:- use command "divl %ebx " to divide the value of eax by ebx general register and quotient will be stored in eax and remainder in edx.

Step8:- move value of ebx to eax and edx to ebx general purpose registers using "movl" command.

Step9:- terminate the loop if value of edx is zero.

Step10:- move value of eax to variable 'multi' and value of a to eax general purpose registers using "movl" command.

Step11:- use command "mull b " to multiply the value of eax by b general register and answer will be stored in eax.

Step12:- use command "divl multi " to divide the value of eax by variable multi general register and quotient will be stored in eax and remainder in edx.

Step13:- move value of "multi" to ecx register.

// therefore the value of gcd will be stored in ecx register and value of lcm is stored in eax register.

Step14:- stop

Q5:- to print Fibonacci series:-

Step1:- define array of size=12 with all of its elements as zero except the second element as 1 in section data.

Step2:- start

Step3:-move value ‘0’ to eax , ‘1’to ebx and ‘1’to ecx general purpose registers using “movl” command.

// here register eax will be used as index1 , ecx wil be used as index 2 and ebx will give Sum.

Step4:- run a loop:-

Step5:- use command “addl array(,%eax,4),%ebx ” to store addition of 0,1 in ebx.

Step6:- increment index1 (ecx) by 1 using addl function

Step7:- move value of ebx(sum) to ecxth position (i.e index 1 position) of array.

Step8:- increment index2 (eax) by 1 using addl function

Step9:- terminate the loop when value of ecx=12.

// all Fibonacci numbers will be stored in array and we can print it in terminal by array@12. Where 12 is length of array.

Step10:- stop

6. Presentation of Results

1. Print all even numbers in ‘n’ natural numbers

```
1   .section .data
2       array:
3           .int 0,0,0,0,0,0,0,0,0,0,0,0
4   .section .text
5   .globl _start
6   _start:
7       movl $0,%eax
8       movl $0,%ecx
9   loop:
10      movl %eax,array( ,%ecx,4)
11      addl $2,%eax
12      addl $1,%ecx
13      cmp $12,%ecx
14      jne loop
15
16
17
18  movl $0,%eax
19  movl $1,%ebx
20  int $0x80
```

```
(gdb) r
Starting program: /home/mplab/kkk/lab44

Program received signal SIGSEGV, Segmentation fault.
0x0804809c in ?? ()
(gdb) print array@12
$1 = {0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22}
(gdb)
```

2) Print all odd numbers in ‘n’ natural numbers

```
1   .section .data
2       array:
3           .int 0,0,0,0,0,0,0,0,0,0,0,0
4   .section .text
5   .globl _start
6   _start:
7       movl $3,%eax
8       movl $0,%ecx
9   loop:
10      movl %eax,array( ,%ecx,4)
11      addl $2,%eax
12      addl $1,%ecx
13      cmp $12,%ecx
14      jne loop
15
16
17
18  movl $0,%eax
19  movl $1,%ebx
20  int $0x80
```

```
Program received signal SIGSEGV, Segmentation fault.
0x0804809c in ?? ()
(gdb) print array@12
$1 = {3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25}
(gdb)
```

3&4) Gcd and lcm :-

```
1   .section .data
2       a:
3           .int 40
4       b:
5           .int 18
6       multi:
7           .int 1
8   .section .text
9   .globl _start
10  _start:
11      movl a,%eax
12      movl b,%ebx
13  loop:
14      movl $0,%edx
15      divl %ebx
16      movl %ebx,%eax
17      movl %edx,%ebx
18      cmpl $0,%edx
19      jnz loop
20
21      movl %eax,multi
22      movl a,%eax
23      mull b
24      divl multi
25      movl multi,%ecx
26
27  movl $0,%eax
28  movl $1,%ebx
29  int $0x80
```

```
Breakpoint 1 at 0x80480ab: file lcm.s, line 26.
(gdb) run
Starting program: /home/mplab/hcf/lcm

Breakpoint 1, loop () at lcm.s:27
27      movl $0,%eax
(gdb) info registers
eax            0x168    360
ecx            0x2      2
edx            0x0      0
ebx            0x0      0
esp            0xbffff050       0xbffff050
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x80480ab        0x80480ab <loop+44>
eflags         0x202    [ IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
(gdb)
```

5) Fibonacci

```
.section .data
    array:
        .int 0,1,0,0,0,0,0,0,0,0,0,0
.section .text
.globl _start
_start:
    movl $0,%eax
    movl $1,%ebx
    movl $1,%ecx

loop:
    addl array( ,%eax,4),%ebx
    addl $1,%ecx
    movl %ebx,array( ,%ecx,4)
    addl $1,%eax
    cmp $12,%ecx
    jne loop

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

Output

```
Program received signal SIGSEGV, Segmentation fault.
0x080480a8 in ?? ()
(gdb) print array@12
$1 = {0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89}
(gdb)
```

7. Analysis and Discussions:-
    - Even and odd program can be done with incrementing index by 1 and incrementing anyone of the registers as by 2 (this register value is stored in array which gives the even or odd series.)
    - Similarly, Fibonacci program can be done using two indices and register which stores sum of last 2 indices of array. These 2 indices will run with difference of 1.

- Gcd of 2 numbers can be found by logic of long division method and lcm can be found by gcd divided by a*b.

8. Conclusions:-

Control flow operations using conditional instructions in assembly language have been studied both practically and theoretically.

Signature and date Marks