Laboratory 2

Title of the Laboratory Exercise: Arithmetic Operations

1. Introduction and Purpose of Experiment

   Students will be able to perform all arithmetic operations using assembly instructions

2. Aim and Objectives

   Aim

   To develop assembly language program to perform all arithmetic operations.

   Objectives

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

   - Identify the appropriate assembly language instruction for the given arithmetic operations
   - Perform all arithmetic operations using assembly language instructions
   - Understand different data types and memory used
   - 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

- Consider the following source code fragment

```
Int a,b,c,d;
a= (b + c)-d + (b*c) / d;
```

  - Assume that *b, c, d* are in registers. Develop an assembly language program to perform this assignment statements.

- Assume that *b is* in registers and *c*, *d* in memory. Develop an assembly language program to perform this assignment statements.

  Value of b= 7654321

  Value of c= 3110000

  Value of d=2344

5. Calculations/Computations/Algorithms

   Consider the following source code fragment

   Int a,b,c,d;

   *a= (b + c)-d + (b*c) / d;*

- Assume that *b, c, d* are in registers. Develop an assembly language program to perform this assignment statements.

ex.s

```
.section .data
        b:
                .int 10
        c:
                .int 20
        d:
                .int 2

        a:
                .int 0
.section .text
.globl _start
_start:

movl $10,%eax
movl $20,%ebx
movl $2,%ecx

mull %ebx
divl %ecx
addl b,%ebx
subl %ecx,%ebx
addl %ebx,%eax
movl %eax,a

movl $1,%eax
movl $0,%ebx
int $0*80
```

Figure 2.1

Figure 2.1 represents the algorithm for the given source code fragment for all three general purpose registers

In this algorithm as the question suggests, it uses all the inputs as general purpose registers. By using “mov” command the assigned values of b, c, d are moved to the registers eax, ebx, ecx , respectively. As per the given expression a= (b + c)-d + (b*c) / d; , according to “Bodmas rule” by using “mul” command ebx register is multiplied to eax and stored in eax by default. By using “div” command eax register is devided by ecx and stored in register eax, now the expression for eax register becomes eax = (b*c) / d; . then by “add” command b is added to register ebx and stored in ebx, by using “sub” command ecx register is subtracted by ebx and stored in ebx, then the register ebx becomes ebx = (b + c)-d; , again by using “add” command ebx is added to eax and stored in eax, then eax becomes eax = (b + c)-d + (b*c)/ d; the required answer, then by “mov” command answer is moved to the memory region a and displayed.

- Assume that *b is* in registers and *c, d* in memory. Develop an assembly language program to perform this assignment statements.

  Value of b= 7654321

  Value of c= 3110000

  Value of d=2344

ex.s

```
.section .data
        c:
                .int 3110000
        d:
                .int 2344

        a:
                .int 0
.section .text
.globl _start
_start:

movl $7654321,%eax

divl d
mull c
movl $7654321,%ebx
addl c,%ebx
subl d,%ebx
addl %ebx,%eax
movl %eax,a

movl $1,%eax
movl $0,%ebx
int $0*80
```

Figure 2.2

Figure 2.2 represents the algorithm for the given source code fragment for b as general purpose register and c and d are located in memory region.

Here b is stored in a general purpose register and c and d are stored in memory region, assign the given values 3110000 and 2344 to c and d respectively. and using “mov” command move the given integer value 7654321 to the general purpose register eax , and then by using “div” command divide the eax by d and stored in eax, then using “mul’ command multiply the c to eax and stored in eax, then eax becomes, eax = (b*c)/d; then again by using the “mov” command mov the integer value 7654321 to the register ebx, by using “add” command add the c to the ebx and stored in ebx ,then by “sub” command subtract the d by ebx and stored in ebx, then ebx becomes ebx = (b+c)-d; , then by “add” command add the obtained ebx to the eax and stored in eax then eax becomes, eax = (b+c)-d+(b*c)/d; , then by “mov” command move the obtained answer to a and display a.

6. Presentation of Results
    - Assume that *b, c, d* are in registers. Develop an assembly language program to perform this assignment statements.

```
exam@msruas-cse-vbox-ubt:~/deepak$ as -gstabs ex.s -o ex.o
exam@msruas-cse-vbox-ubt:~/deepak$ ld ex.o -o ex
exam@msruas-cse-vbox-ubt:~/deepak$ gdb ex
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ex...done.
(gdb) break 27
Breakpoint 1 at 0x80480a8: file ex.s, line 27.
(gdb) run
Starting program: /home/exam/deepak/ex
```

Figure 2.3

Figure 2.3 represent the command to open directory and break the program at certain break points

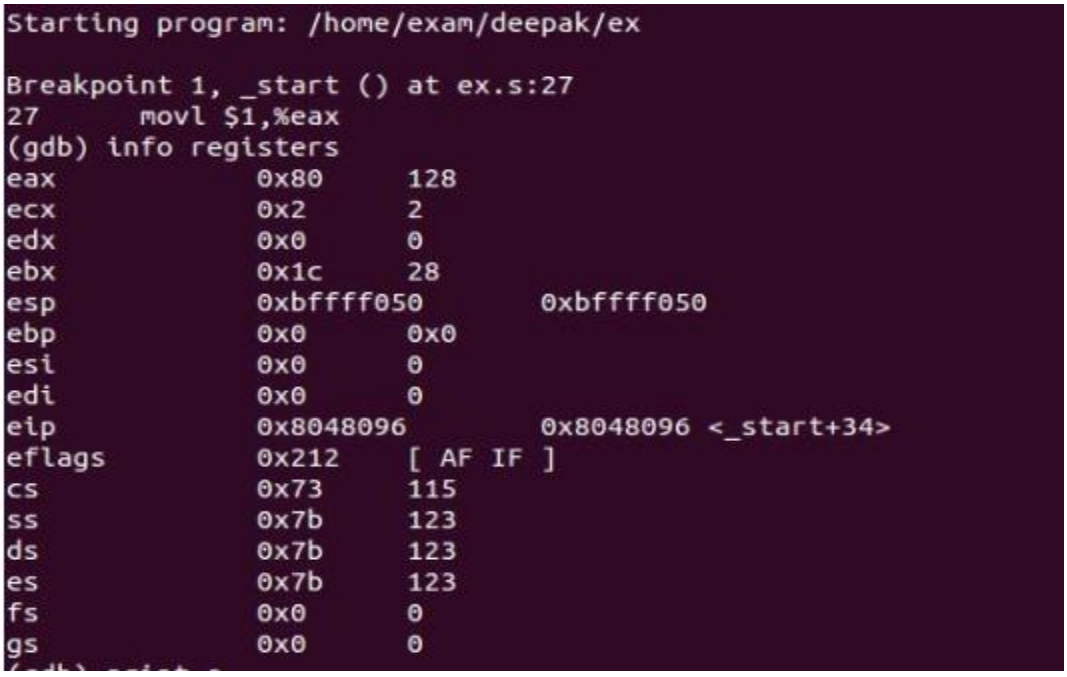


Figure 2.4

Figure 2.4 represents the break point 1 where all the data transfer units are assigned to default values

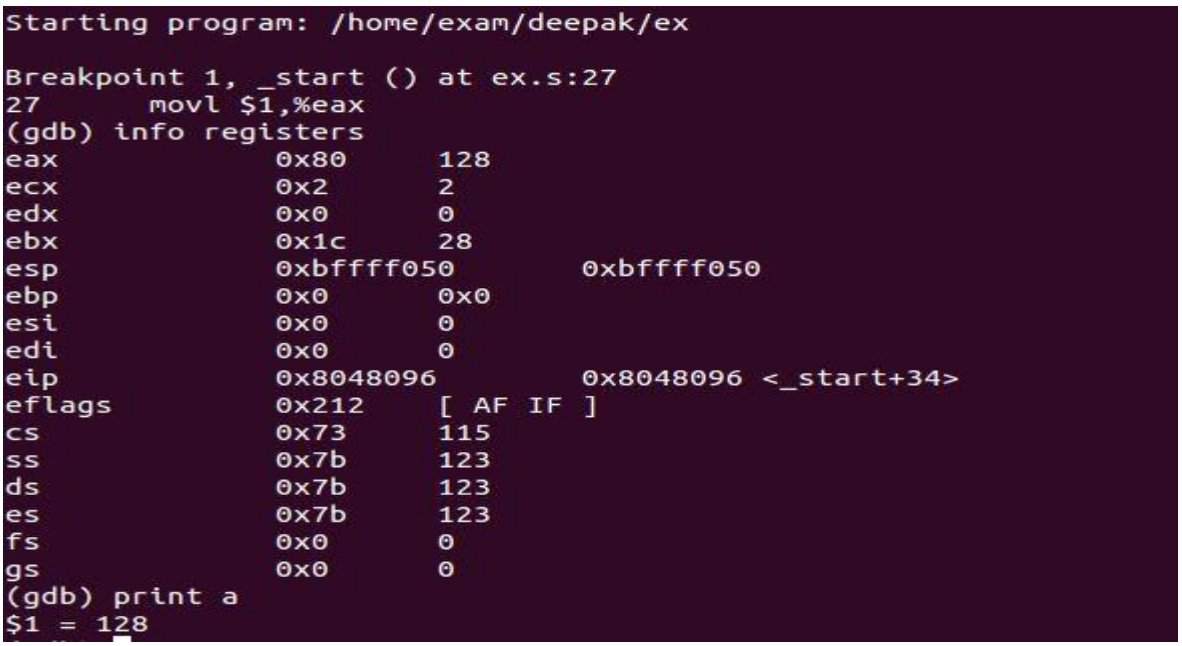


Figure 2.5

Figure 2.5 represents the break point 2 where answer is assigned to general purpose register eax

The answer to this question is 129 which is stored in memory region a , so as we print a we can see the answer 129

- Assume that *b is* in registers and *c*, *d* in memory. Develop an assembly language program to perform this assignment statements.

  Value of b= 7654321

  Value of c= 3110000

  Value of d=2344

```
Starting program: /home/exam/deepak/ex

Breakpoint 1, _start () at ex.s:23
23      movl %eax,a
(gdb) info registers
eax            0x5de03f69       1574977385
ecx            0x0      0
edx            0x2      2
ebx            0xa436f9 10761977
esp            0xbffff050       0xbffff050
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x8048098        0x8048098 <_start+36>
eflags         0x206    [ PF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
```

Figure 2.6

Figure 2.6 represents the break point 1 where answer is assigned to register edx:eax

Here the question is too complicated that b and c having 7 bits, so the answer is more than 8 bit, but the problem is we don't have the register to store that value , so we are using edx:eax register that is lower order edx and higher order eax , so in this problem hexadecimal number 5de03f69 is stored in eax but this is not an answer since 8 bits are completely filled and edx filled by number 2, so the result is stored in both lower order edx and higher order eax , so our answer is hexadecimal number 25de03f69 that is 10,16,49,11,977 in decimal.

7. Analysis and Discussions
   - Learn to Identify the appropriate assembly language instruction for the given arithmetic operation (i.e ADD,SUB,MUL,DIV)
   - Used to Perform all arithmetic operations using assembly language instructions
   - Understand different data types and memory used
   - Get familiar with assembly language program by developing simple programs
8. Conclusions

   From the given two programs we can conclude that all arithmetic operations can be used in assembly level programming, for more than eight bit answer we can't use general purpose register so we are using edx:eax register.

9. Comments

   1. Limitations of Experiments

      For answer more than eight bit, it can't display a proper answer.

   2. Limitations of Results

      Result can't be stored in general purpose register when answer is more than eight bit, so answer is stored in lower order edx and higher order eax. So printing an answer doesn't show a correct value.

   3. Learning happened

      Learned to use assembly language instruction with all arithmetic operation of a assembly level programming, and also learn the data types used in the programming and how the data can be stored in register etc..

   4. Recommendations

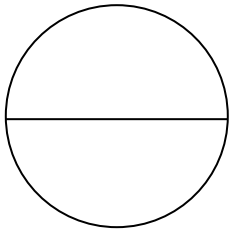
      We are supposed to carefully do operations by taking consideration of whether particular bit of answer can be stored in which general purpose register by giving thought to this we can overcome the limitations.

Name: DEEPAK R

Reg No: 18ETCS002041

Signature and date

Marks



****