Lab Report

COEN311- LAB 2

Submitted to:
Amirreza Mousavi

Date performed: Feb 25th, 2021
Lab section: YL-X

By:

Bayan Alsalem
40105034

Objectives

- To learn about the different types of addressing mode
- Get more familiar with NASM and GDB

Theory

The different ways of determining the address of the operands are called addressing modes. The addressing modes describe a way that a machine instruction can specify how to retrieve a value from various locations. In direct addressing mode, the address field contains the address of the operand. In immediate addressing mode, the operand is a part of the instruction and no memory reference is required to fetch data. [1]

Conclusion

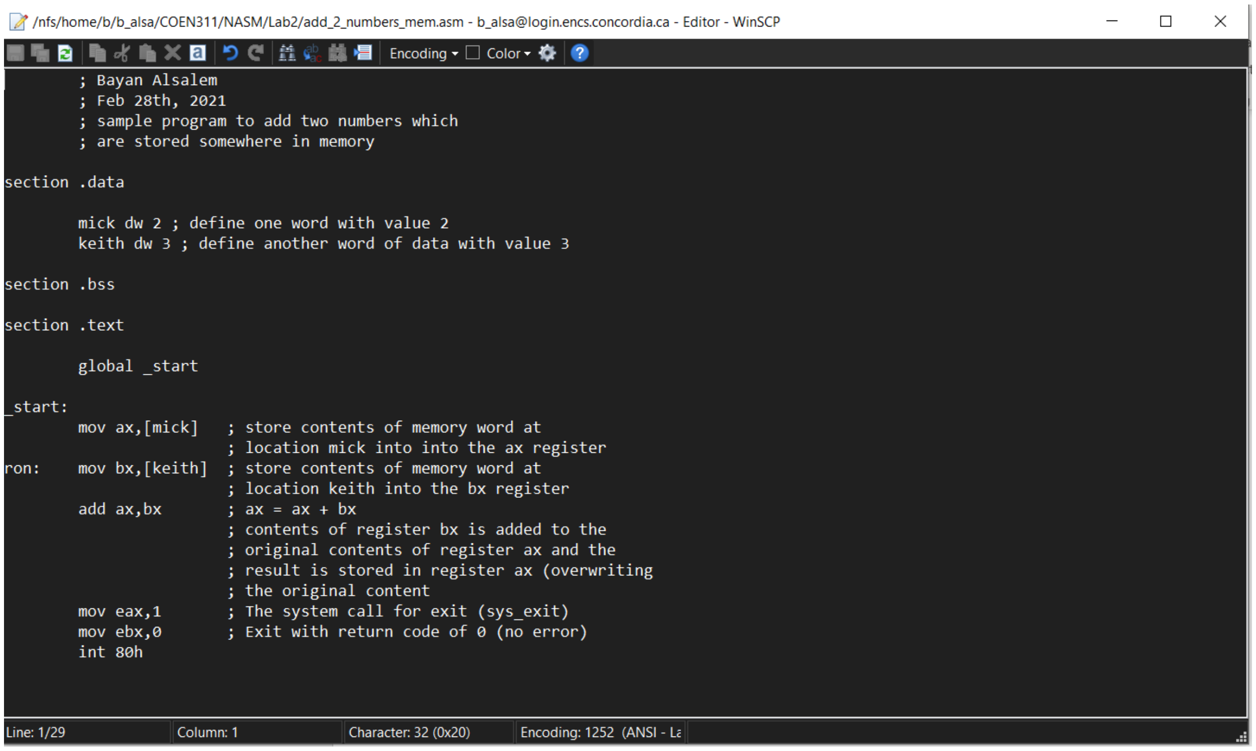
The addressing modes were introduced and explained. We learned about the difference between immediate and direct addressing modes. The command x (for “examine”) in the GDB debugger was successfully used to examine memory by specifying how much memory to display and in what unit size (b-> Bytes, h-> Halfwords (2 bytes)).

[1] https://www.geeksforgeeks.org/difference-between-direct-and-immediate-addressing-modes/

Appendix

- **The .asm text file containing the Intel x86 assembly language program which adds the contents of two registers.**

```
section .data

mick dw 2 ; define one word with value 2
keith dw 3 ; define another word of data with value 3

section .bss

section .text

        global _start

_start:

        mov ax,[mick]   ; store contents of memory word at

                        ; location mick into into the ax register


ron:    mov bx,[keith]          ; store contents of memory word at

                        ; location keith into the bx register

        add ax,bx       ; ax = ax + bx

                        ; contents of register bx is added to the

                        ; original contents of register ax and the

                        ; result is stored in register ax (overwriting

                        ; the original content

        mov eax,1       ; The system call for exit (sys_exit)

        mov ebx,0       ; Exit with return code of 0 (no error)

        int 80h
```



/nfs/home/b/b_alsa/COEN311/NASM/Lab2/add_2_numbers_mem.asm - b_alsa@login.encs.concordia.ca - Editor - WinSCP

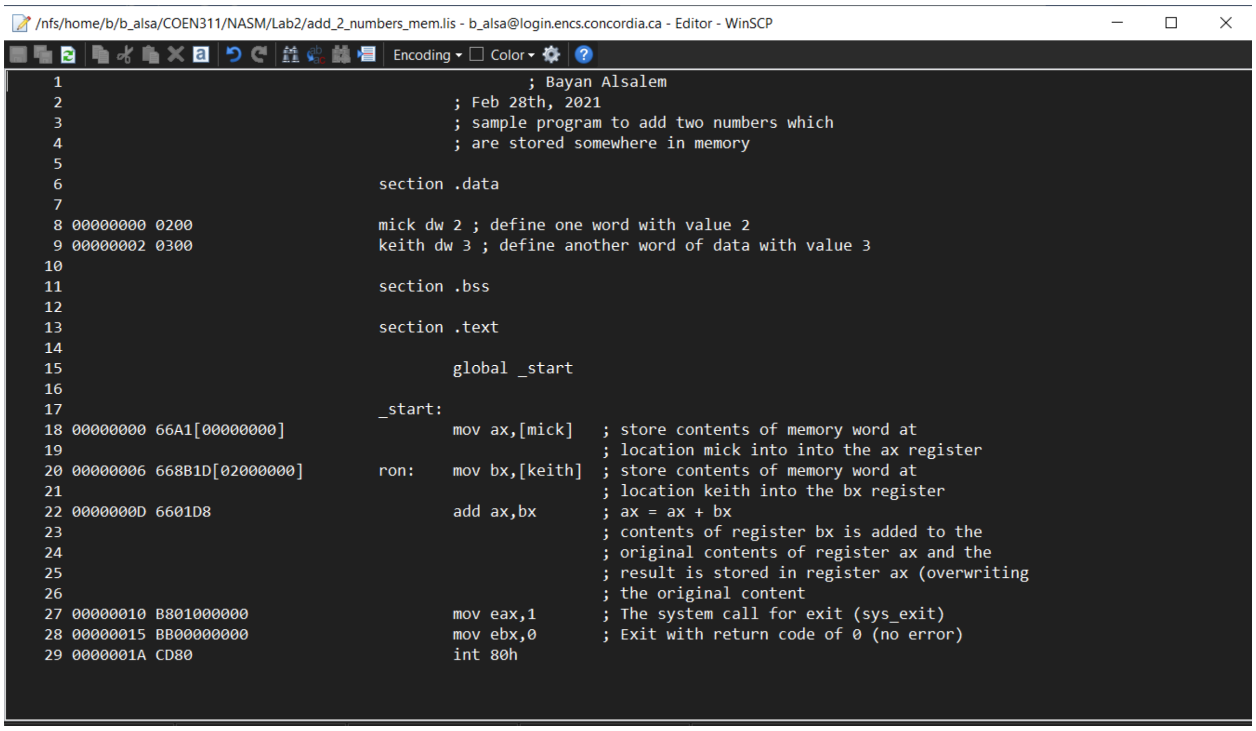
```
        ; Bayan Alsalem
        ; Feb 28th, 2021
        ; sample program to add two numbers which
        ; are stored somewhere in memory

section .data

        mick dw 2 ; define one word with value 2
        keith dw 3 ; define another word of data with value 3

section .bss

section .text

        global _start

_start:
        mov ax,[mick]   ; store contents of memory word at
                        ; location mick into into the ax register
ron:    mov bx,[keith]  ; store contents of memory word at
                        ; location keith into the bx register
        add ax,bx       ; ax = ax + bx
                        ; contents of register bx is added to the
                        ; original contents of register ax and the
                        ; result is stored in register ax (overwriting
                        ; the original content
        mov eax,1       ; The system call for exit (sys_exit)
        mov ebx,0       ; Exit with return code of 0 (no error)
        int 80h
```

Line: 1/29 | Column: 1 | Character: 32 (0x20) | Encoding: 1252 (ANSI - La

- **<u>The corresponding listing file</u>**

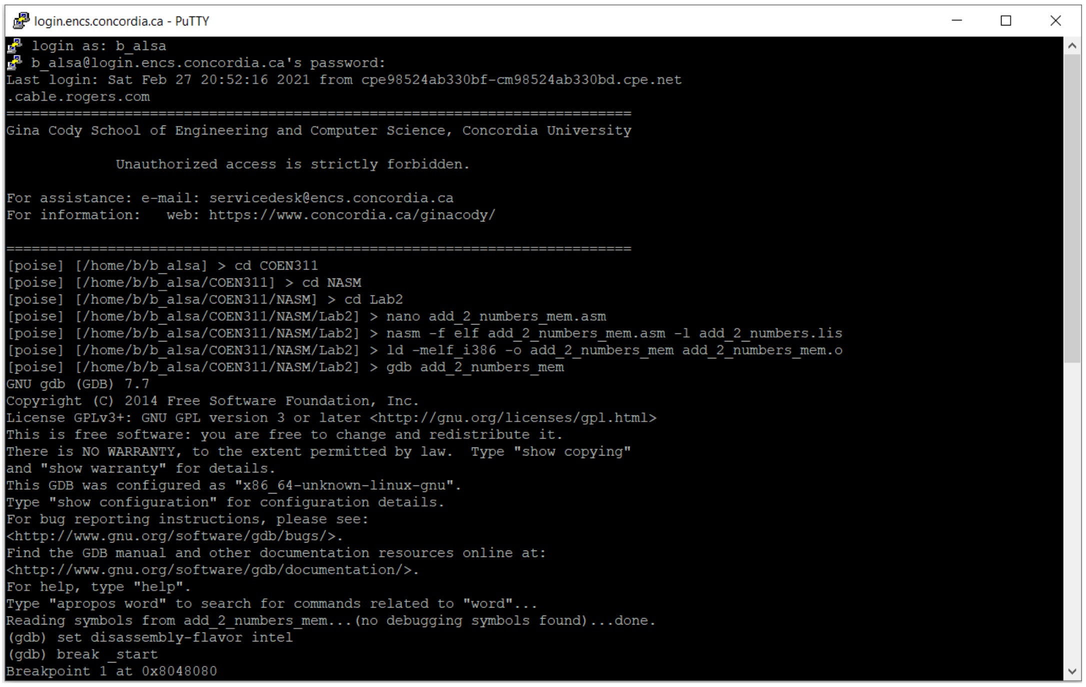
```
     2                                  ; Feb 28th, 2021
     3                                  ; sample program to add two numbers which
     4                                  ; are stored somewhere in memory
     5
     6                                 section .data
     7
     8 00000000 0200                   mick dw 2 ; define one word with value 2
     9 00000002 0300                   keith dw 3 ; define another word of data with value 3
    10
    11                                 section .bss
    12
    13                                 section .text
    14
    15                                   global _start
    16
    17                                 _start:
    18 00000000 66A1[00000000]           mov ax,[mick]   ; store contents of memory word at
```

- 19 ; location mick into into the ax register
- 20 00000006 668B1D[02000000] ron: mov bx,[keith] ; store contents of memory word at
- 21 ; location keith into the bx register
- 22 0000000D 6601D8 add ax,bx ; ax = ax + bx
- 23 ; contents of register bx is added to the
- 24 ; original contents of register ax and the
- 25 ; result is stored in register ax (overwriting
- 26 ; the original content
- 27 00000010 B801000000 mov eax,1 ; The system call for exit (sys_exit)
- 28 00000015 BB00000000 mov ebx,0 ; Exit with return code of 0 (no error)
- 29 0000001A CD80 int 80h

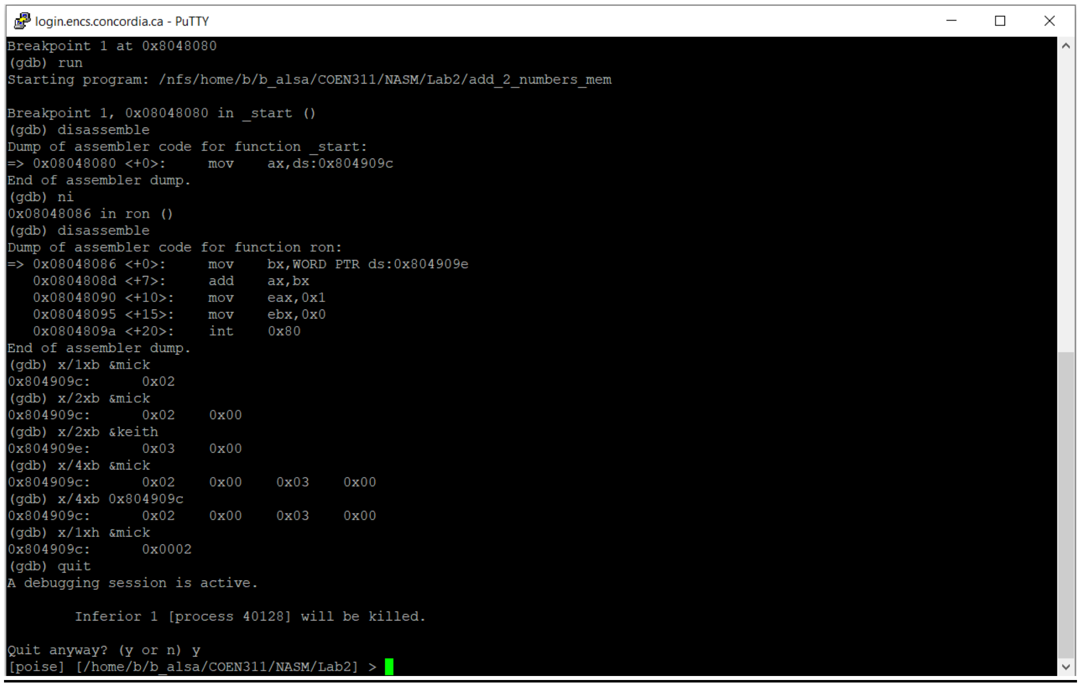

/nfs/home/b/b_alsa/COEN311/NASM/Lab2/add_2_numbers_mem.lis - b_alsa@login.encs.concordia.ca - Editor - WinSCP
Encoding
Color

```
 1                                             ; Bayan Alsalem
 2                                     ; Feb 28th, 2021
 3                                     ; sample program to add two numbers which
 4                                     ; are stored somewhere in memory
 5
 6                             section .data
 7
 8 00000000 0200               mick dw 2 ; define one word with value 2
 9 00000002 0300               keith dw 3 ; define another word of data with value 3
10
11                             section .bss
12
13                             section .text
14
15                                     global _start
16
17                             _start:
18 00000000 66A1[00000000]             mov ax,[mick]   ; store contents of memory word at
19                                                     ; location mick into into the ax register
20 00000006 668B1D[02000000]   ron:    mov bx,[keith]  ; store contents of memory word at
21                                                     ; location keith into the bx register
22 0000000D 6601D8                     add ax,bx       ; ax = ax + bx
23                                                     ; contents of register bx is added to the
24                                                     ; original contents of register ax and the
25                                                     ; result is stored in register ax (overwriting
26                                                     ; the original content
27 00000010 B801000000                 mov eax,1       ; The system call for exit (sys_exit)
28 00000015 BB00000000                 mov ebx,0       ; Exit with return code of 0 (no error)
29 0000001A CD80                       int 80h
```

- **A screenshot (or ASCII copy and paste from your terminal window) of your gdb debugging session with the contents of the destination register after the add instruction has been executed.**



```
login as: b_alsa
b_alsa@login.encs.concordia.ca's password:
Last login: Sat Feb 27 20:52:16 2021 from cpe98524ab330bf-cm98524ab330bd.cpe.net
.cable.rogers.com
==========================================================================
Gina Cody School of Engineering and Computer Science, Concordia University

             Unauthorized access is strictly forbidden.

For assistance: e-mail: servicedesk@encs.concordia.ca
For information:   web: https://www.concordia.ca/ginacody/

==========================================================================
[poise] [/home/b/b_alsa] > cd COEN311
[poise] [/home/b/b_alsa/COEN311] > cd NASM
[poise] [/home/b/b_alsa/COEN311/NASM] > cd Lab2
[poise] [/home/b/b_alsa/COEN311/NASM/Lab2] > nano add_2_numbers_mem.asm
[poise] [/home/b/b_alsa/COEN311/NASM/Lab2] > nasm -f elf add_2_numbers_mem.asm -l add_2_numbers.lis
[poise] [/home/b/b_alsa/COEN311/NASM/Lab2] > ld -melf_i386 -o add_2_numbers_mem add_2_numbers_mem.o
[poise] [/home/b/b_alsa/COEN311/NASM/Lab2] > gdb add_2_numbers_mem
GNU gdb (GDB) 7.7
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 "x86_64-unknown-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 add_2_numbers_mem...(no debugging symbols found)...done.
(gdb) set disassembly-flavor intel
(gdb) break _start
Breakpoint 1 at 0x8048080
```



```
Breakpoint 1 at 0x8048080
(gdb) run
Starting program: /nfs/home/b/b_alsa/COEN311/NASM/Lab2/add_2_numbers_mem

Breakpoint 1, 0x08048080 in _start ()
(gdb) disassemble
Dump of assembler code for function _start:
=> 0x08048080 <+0>:     mov    ax,ds:0x804909c
End of assembler dump.
(gdb) ni
0x08048086 in ron ()
(gdb) disassemble
Dump of assembler code for function ron:
=> 0x08048086 <+0>:     mov    bx,WORD PTR ds:0x804909e
   0x0804808d <+7>:     add    ax,bx
   0x08048090 <+10>:    mov    eax,0x1
   0x08048095 <+15>:    mov    ebx,0x0
   0x0804809a <+20>:    int    0x80
End of assembler dump.
(gdb) x/1xb &mick
0x804909c:      0x02
(gdb) x/2xb &mick
0x804909c:      0x02    0x00
(gdb) x/2xb &keith
0x804909e:      0x03    0x00
(gdb) x/4xb &mick
0x804909c:      0x02    0x00    0x03    0x00
(gdb) x/4xb 0x804909c
0x804909c:      0x02    0x00    0x03    0x00
(gdb) x/1xh &mick
0x804909c:      0x0002
(gdb) quit
A debugging session is active.

        Inferior 1 [process 40128] will be killed.

Quit anyway? (y or n) y
[poise] [/home/b/b_alsa/COEN311/NASM/Lab2] >
```