# Lab Report

COEN311- LAB 1

Submitted to:
Amirreza Mousavi

Date performed: Feb 11th, 2021
Lab section: YL-X

By:

Bayan Alsalem
40105034

## Objectives

- To learn to write assembly language programs on the intel x86 architecture.
- To get familiar with the assembler **NASM** and **nano** the text editor
- To understand the structure of a **NASM** program
- To learn how to debug using the **GDB** debugger

## Theory

Netwide Assembler (NASM) is an assembler and dissembler for the Intel x86 architecture. Compiling an assembly program using NASM can be done on Linux or Windows.[1] Nano is a terminal-based text editor. The loader program (ld) loads the code and data of the executable object file into memory and then runs the program by jumping to the first instruction.[2] The GDB debugger is used to trace the program and check the contents of the registers and the memory.

## Questions

### Explain the differences between an assembly language source code file, the listing file produced by nasm, the object file (.o) and the final executable program.

- Source Code File
  Is the file that conatins the the assembly code/program.

- Listing File
  This file contains both the statements of the source file (the assembly code) on the right and the Hexadecimal machine code generated for each statement on the left. It also shows the names and values of all labels and variables and the number of bytes of each statement (memory locations).

- Object file
  The object file is generated after the assembler converts the assembly language statements into machine code. This file contains object code.

- Executable program
  It is the complete program after linking the object file with other object files in order to run the program. The executable file should be loaded into the memory to execute the program.

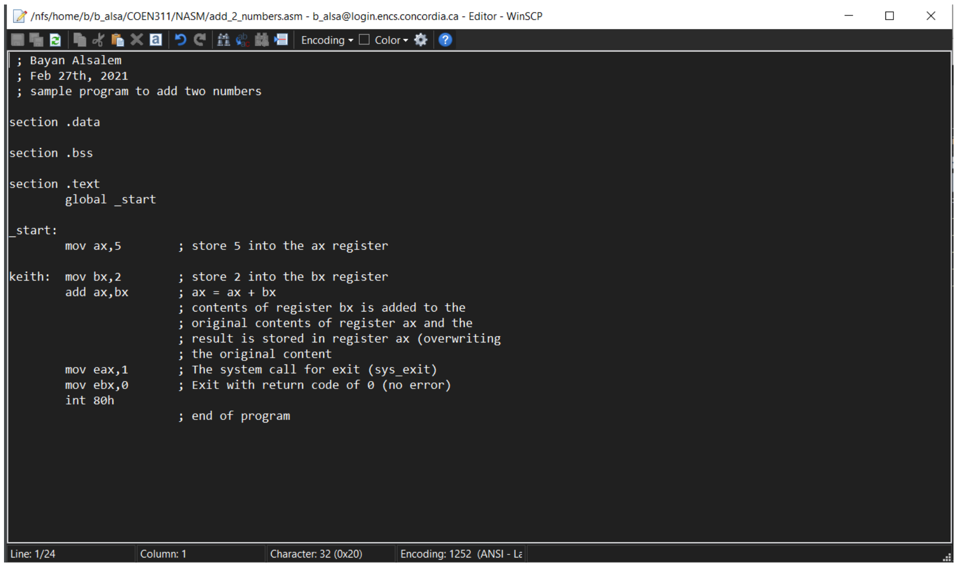
## Conclusion

The source file was ready with the addition of 2 numbers assembly program using the text editor **nano**. Then, the NASM was used to assemble the source file using thr command line:
**nasm -f elf add_2_numbers.asm -l add_2_numbers.lis**
An object file was generated after assembling the code, with that we used the linker/loader (ld) to produce an executable file. GDB debugger was successfully used to check the contents of the registers and memory instruction by instruction.
All the required files were succssufly generated: the source file, the object file, and the excutable file.

## Appendix

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

```
; Bayan Alsalem
 ; Feb 27th, 2021
 ; sample program to add two numbers

section .data
section .bss
section .text
        global _start
_start:
        mov ax,5        ; store 5 into the ax register

keith:  mov bx,2        ; store 2 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
        ; end of program
```



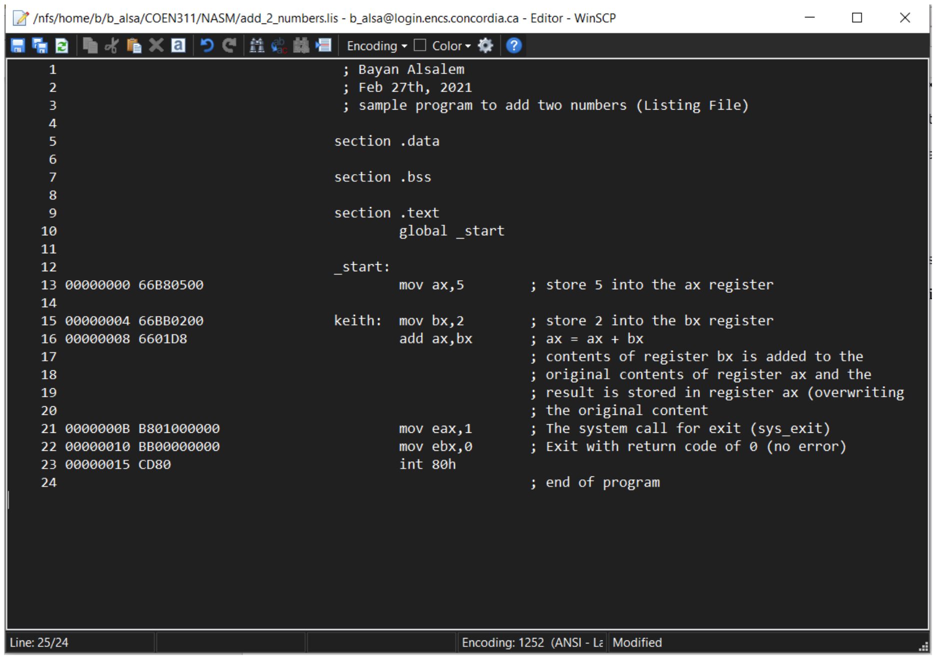
/nfs/home/b/b_alsa/COEN311/NASM/add_2_numbers.asm - b_alsa@login.encs.concordia.ca - Editor - WinSCP

```
 ; Bayan Alsalem
 ; Feb 27th, 2021
 ; sample program to add two numbers

section .data

section .bss

section .text
        global _start

_start:
        mov ax,5        ; store 5 into the ax register

keith:  mov bx,2        ; store 2 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
                        ; end of program
```

Line: 1/24 Column: 1 Character: 32 (0x20) Encoding: 1252 (ANSI - La

**The corresponding listing file**

```
     1                                  ; Bayan Alsalem
     2                                  ; Feb 27th, 2021
     3                                  ; sample program to add two numbers
     4
     5                                 section .data
     6
     7                                 section .bss
     8
     9                                 section .text
    10                                         global _start
    11
    12                          _start:
    13 00000000 66B80500            mov ax,5   ; store 5 into the ax register
    14
    15 00000004 66BB0200          keith: mov bx,2 ; store 2 into the bx
register
    16 00000008 6601D8                 add ax,bx ; ax = ax + bx
    17                              ; contents of register bx is added to the
    18                             ; original contents of register ax and the
    19                            ; result is stored in register ax
(overwriting
    20                                ; the original content
    21 0000000B B801000000             mov eax,1       ; The system call for
exit (sys_exit)
    22 00000010 BB00000000             mov ebx,0       ; Exit with return
code of 0 (no error)
    23 00000015 CD80                   int 80h
    24                                         ; end of program
```

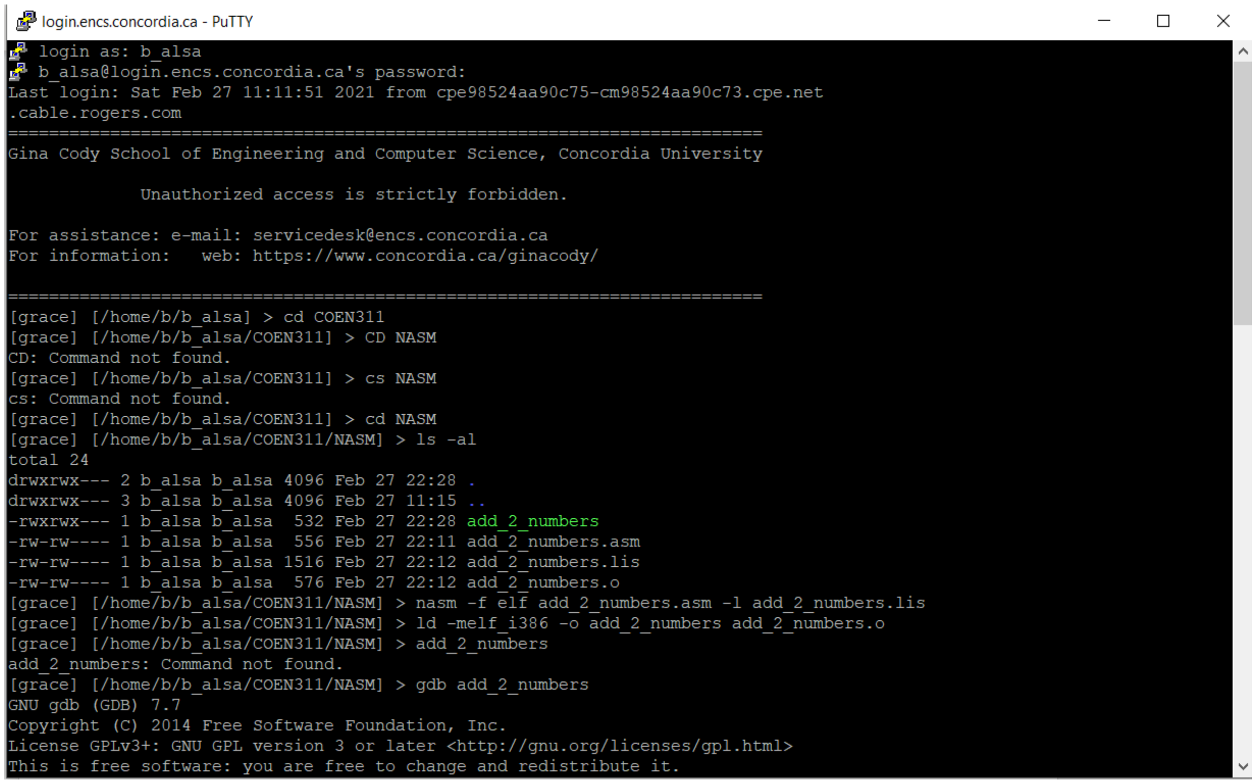


/nfs/home/b/b_alsa/COEN311/NASM/add_2_numbers.lis - b_alsa@login.encs.concordia.ca - Editor - WinSCP

```
     1                                  ; Bayan Alsalem
     2                                  ; Feb 27th, 2021
     3                                  ; sample program to add two numbers (Listing File)
     4
     5                                 section .data
     6
     7                                 section .bss
     8
     9                                 section .text
    10                                         global _start
    11
    12                                 _start:
    13 00000000 66B80500                       mov ax,5        ; store 5 into the ax register
    14
    15 00000004 66BB0200               keith:  mov bx,2        ; store 2 into the bx register
    16 00000008 6601D8                         add ax,bx       ; ax = ax + bx
    17                                                         ; contents of register bx is added to the
    18                                                         ; original contents of register ax and the
    19                                                         ; result is stored in register ax (overwriting
    20                                                         ; the original content
    21 0000000B B801000000                     mov eax,1       ; The system call for exit (sys_exit)
    22 00000010 BB00000000                     mov ebx,0       ; Exit with return code of 0 (no error)
    23 00000015 CD80                           int 80h
    24                                                         ; end of program
```

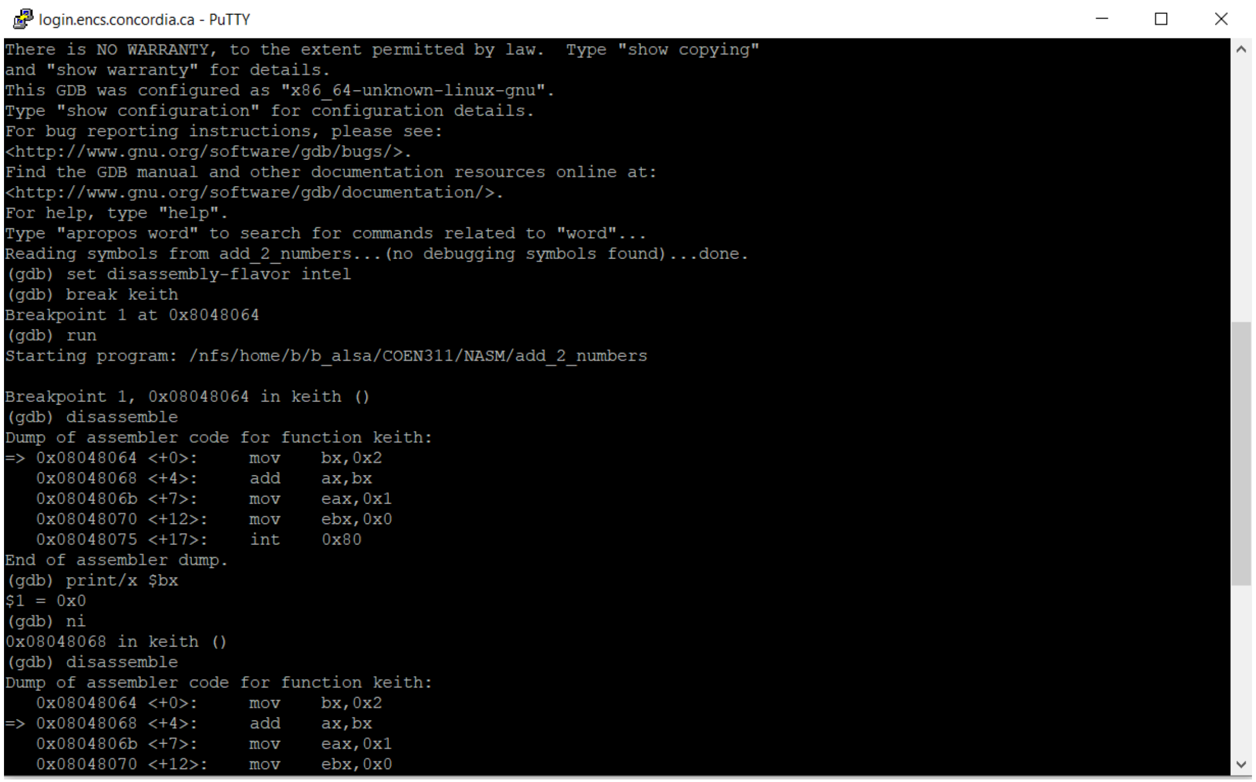
Line: 25/24 | Encoding: 1252 (ANSI - La | Modified

**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.**



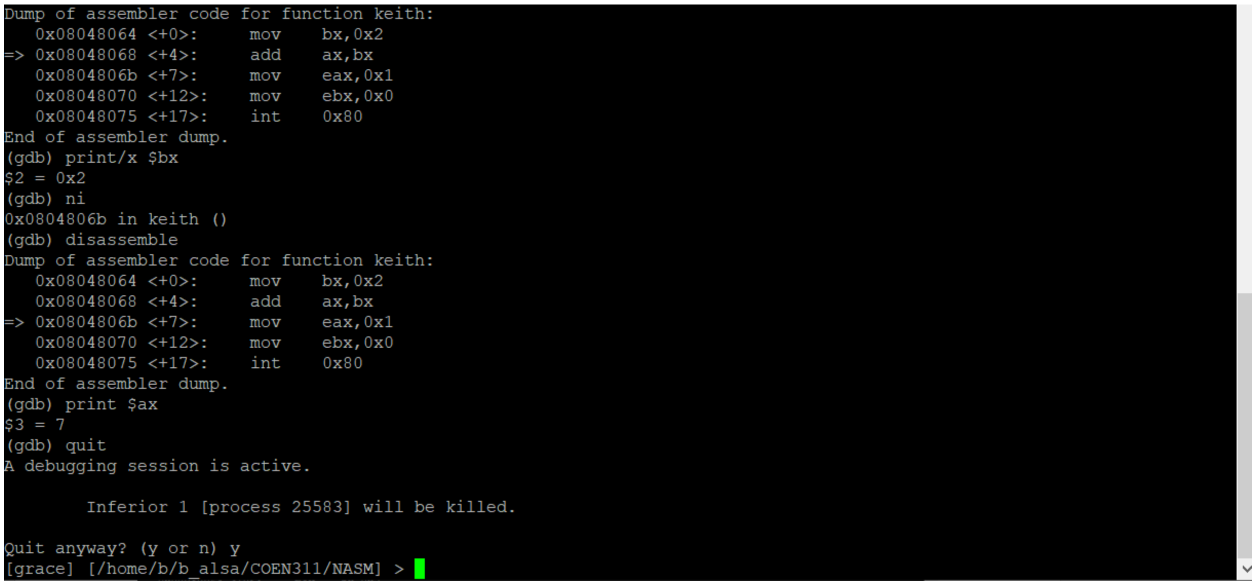
login.encs.concordia.ca - PuTTY

```
 login as: b_alsa
 b_alsa@login.encs.concordia.ca's password:
Last login: Sat Feb 27 11:11:51 2021 from cpe98524aa90c75-cm98524aa90c73.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/

==========================================================================
[grace] [/home/b/b_alsa] > cd COEN311
[grace] [/home/b/b_alsa/COEN311] > CD NASM
CD: Command not found.
[grace] [/home/b/b_alsa/COEN311] > cs NASM
cs: Command not found.
[grace] [/home/b/b_alsa/COEN311] > cd NASM
[grace] [/home/b/b_alsa/COEN311/NASM] > ls -al
total 24
drwxrwx--- 2 b_alsa b_alsa 4096 Feb 27 22:28 .
drwxrwx--- 3 b_alsa b_alsa 4096 Feb 27 11:15 ..
-rwxrwx--- 1 b_alsa b_alsa  532 Feb 27 22:28 add_2_numbers
-rw-rw---- 1 b_alsa b_alsa  556 Feb 27 22:11 add_2_numbers.asm
-rw-rw---- 1 b_alsa b_alsa 1516 Feb 27 22:12 add_2_numbers.lis
-rw-rw---- 1 b_alsa b_alsa  576 Feb 27 22:12 add_2_numbers.o
[grace] [/home/b/b_alsa/COEN311/NASM] > nasm -f elf add_2_numbers.asm -l add_2_numbers.lis
[grace] [/home/b/b_alsa/COEN311/NASM] > ld -melf_i386 -o add_2_numbers add_2_numbers.o
[grace] [/home/b/b_alsa/COEN311/NASM] > add_2_numbers
add_2_numbers: Command not found.
[grace] [/home/b/b_alsa/COEN311/NASM] > gdb add_2_numbers
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.
```



login.encs.concordia.ca - PuTTY

```
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...(no debugging symbols found)...done.
(gdb) set disassembly-flavor intel
(gdb) break keith
Breakpoint 1 at 0x8048064
(gdb) run
Starting program: /nfs/home/b/b_alsa/COEN311/NASM/add_2_numbers

Breakpoint 1, 0x08048064 in keith ()
(gdb) disassemble
Dump of assembler code for function keith:
=> 0x08048064 <+0>:     mov    bx,0x2
   0x08048068 <+4>:     add    ax,bx
   0x0804806b <+7>:     mov    eax,0x1
   0x08048070 <+12>:    mov    ebx,0x0
   0x08048075 <+17>:    int    0x80
End of assembler dump.
(gdb) print/x $bx
$1 = 0x0
(gdb) ni
0x08048068 in keith ()
(gdb) disassemble
Dump of assembler code for function keith:
   0x08048064 <+0>:     mov    bx,0x2
=> 0x08048068 <+4>:     add    ax,bx
   0x0804806b <+7>:     mov    eax,0x1
   0x08048070 <+12>:    mov    ebx,0x0
```



```
Dump of assembler code for function keith:
   0x08048064 <+0>:     mov    bx,0x2
=> 0x08048068 <+4>:     add    ax,bx
   0x0804806b <+7>:     mov    eax,0x1
   0x08048070 <+12>:    mov    ebx,0x0
   0x08048075 <+17>:    int    0x80
End of assembler dump.
(gdb) print/x $bx
$2 = 0x2
(gdb) ni
0x0804806b in keith ()
(gdb) disassemble
Dump of assembler code for function keith:
   0x08048064 <+0>:     mov    bx,0x2
   0x08048068 <+4>:     add    ax,bx
=> 0x0804806b <+7>:     mov    eax,0x1
   0x08048070 <+12>:    mov    ebx,0x0
   0x08048075 <+17>:    int    0x80
End of assembler dump.
(gdb) print $ax
$3 = 7
(gdb) quit
A debugging session is active.

        Inferior 1 [process 25583] will be killed.

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