

COEN 311 Lab 3

By

Mamadou Diao Kaba

27070179

Performed on

March 18<sup>th</sup> 2021

## **Objectives**

The objectives of this lab are first to explore addressing modes. Secondly, it was to learn more about Linux utilities by using using the nasm assembler and the gdb command line debugger.

## **Theory**

The addressing modes of the Intel x86 role is to access the data operands of the assembly language instruction. The Linux nasm assembler role is to assemble the instructions of the Intel x86 into the machine. The gdb debugger role is to allow for the register and memory contents to be read by single-stepping through the program. Finally, the role of the ld loader program is to convert the machine code into an executable program and to load it into the memory.

## **Question**

NONE

## **Conclusion**

We further familiarised with the Linux nasm assembler by showing it assemble the instructions given into the machine (assembly language source code file), with the ld loader program role to convert the machine code into an executable program and to load it into the memory (listing files produced by nasm, object file). And finally, we explored addressing modes by analysing the .lst and .lis files of our program and by using the gdb debugger to allow for the register and memory contents to be read by single-stepping through the program (mainly Mick and Keith).

## APPENDIX

### ASM FILE

```
login.encs.concordia.ca - PuTTY
GNU nano 2.3.1 File: lab2.asm

;Mamadou Kaba
;February 25,2021

section .data
mick dw 2; define one word (2 bytes) of data
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
        mov eax,1;      The system call for exit(sys_exit)
        mov ebx,0;      Exit with return code of 0(no error)
        int 80h
```

### LST FILE

```
1          ;Mamadou Kaba
2          ;February 25,2021
3
4          section .data
5
6 00000000 0200          mick dw 2; define one word (2 bytes) of data
7 00000002 0300          keith dw 3; define another word of data with value 3
8
9          section .bss
10
11         section .text
12             global _start
13
14         _start:
15 00000000 66A1[00000000]          mov ax, [mick]; store contents of memory word at location mick into into the ax register
16
17 00000006 668B1D[02000000]      ron:    mov bx,[keith]; store contents of memory word at location keith into the bx register
18 0000000D 6601D8          add ax,bx;      ax = ax + bx contents of register bx is added to the original content
19 00000010 B801000000          mov eax,1;      The system call for exit(sys_exit)
20 00000015 BB00000000          mov ebx,0;      Exit with return code of 0(no error)
21 0000001A CD80          int 80h
```

## LIS FILE

/nfs/home/m/ma\_kaba/COEN311/NASM/lab2/lab2.lis - ma\_kaba@login.encs.concordia.ca - Editor - WinSCP

```
1 ;Mamadou Kaba
2 ;February 25,2021
3
4 section .data
5
6 00000000 0200 mick dw 2; define one word (2 bytes) of data
7 00000002 0300 keith dw 3; define another word of data with value 3
8
9 section .bss
10
11 section .text
12 global _start
13
14 _start:
15 00000000 66A1[00000000] mov ax, [mick]; store contents of memory word at location mick into into the ax register
16
17 00000006 66B1D[02000000] ron: mov bx,[keith]; store contents of memory word at location keith into the bx register
18 0000000D 6601D8 add ax,bx; ax = ax + bx contents of register bx is added to the original content
19 00000010 B801000000 mov eax,1; The system call for exit(sys_exit)
20 00000015 BB00000000 mov ebx,0; Exit with return code of 0(no error)
21 0000001A CD80 int 80h
```

## GDB COMMAND

```
[poise] [/home/m/ma_kaba/COEN311/NASM/lab2] > gdb lab2
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 lab2...(no debugging symbols found)...done.
(gdb) break _start
Breakpoint 1 at 0x8048080
(gdb) run
Starting program: /nfs/home/m/ma_kaba/COEN311/NASM/lab2/lab2

Breakpoint 1, 0x08048080 in _start ()
(gdb) disassemble
Dump of assembler code for function _start:
=> 0x08048080 <+0>: mov 0x804909c,%ax
End of assembler dump.
(gdb) ni
0x08048086 in ron ()
(gdb) disassemble
Dump of assembler code for function ron:
=> 0x08048086 <+0>: mov 0x804909e,%bx
0x0804808d <+7>: add %bx,%ax
0x08048090 <+10>: mov $0x1,%eax
0x08048095 <+15>: mov $0x0,%ebx
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)
```

Mov eax,4

Mov ebx,1

Mov ecx, message

Mov edx,22

Int 80h

Based index addressing

Mov bl, message

Mov eax, message

Mov bl, [eax] (bl =106)

Cmp bl, 0

Jz display