Assembly (1A)

Young Won Lim
5/26/16

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Young Won Lim
5/26/16

ELF

```
/* tiny.c */
int main(void){
  return 42;
}
```

```
$ gcc -Wall -s -O3 tiny.c
$ ./a.out ; echo $?
42
```

```
; tiny.asm
BITS 32
GLOBAL main
SECTION .text
main:
        mov     eax, 42
        ret
```

```
$ nasm -f elf tiny.asm
$ gcc -Wall -s tiny.o
$ ./a.out ; echo $?
42
```

```
; tiny.asm
BITS 32
GLOBAL _start
SECTION .text
_start:
        mov     eax, 42
        ret
```

ELF

```
; tiny.asm
BITS 32
GLOBAL _start
SECTION .text
_start:
            mov     eax, 1
            mov     ebx, 42
            int     0x80
```

```
$ nasm -f elf tiny.asm
$ gcc -Wall -s -nostdlib tiny.o
$ ./a.out ; echo $?
42
```

```
$ nasm -f elf tiny.asm
$ ld -s tiny.o
$ ./a.out ; echo $?
42
$ wc -c a.out
   368 a.out
```

```
$ objdump -x a.out | less
```

ELF

$ objdump -x a.out | less

```
$ nasm -f elf tiny.asm
$ ld -s tiny.o
$ ./a.out ; echo $?
42
$ wc -c a.out
   368 a.out
```

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000007  08048080  08048080  00000080  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .comment      0000001c  00000000  00000000  00000087  2**0
                  CONTENTS, READONLY
```

gcc -S func.c

```
int func() {
   int pow = 1;
   int x = 0;

   while (pow != 128) {
        pow = pow * 2;
        x = x + 1;
   }

   return (x);
}
```

gcc -c -S func.c

```
     .file     "func.c"
     .text
     .globl    func
     .type     func, @function
func:
.LFB0:
     .cfi_startproc
     pushq     %rbp
     .cfi_def_cfa_offset 16
     .cfi_offset 6, -16
     movq %rsp, %rbp
     .cfi_def_cfa_register 6
     movl $1, -8(%rbp)
     movl $0, -4(%rbp)
     jmp  .L2
.L3:
     sall -8(%rbp)
     addl $1, -4(%rbp)
.L2:
     cmpl $128, -8(%rbp)
     jne  .L3
     movl -4(%rbp), %eax
     popq %rbp
     .cfi_def_cfa 7, 8
     ret
     .cfi_endproc
.LFE0:
     .size     func, .-func
     .ident    "GCC: (Ubuntu 5.3.1-14ubuntu2) 5.3.1 20160413"
     .section  .note.GNU-stack,"",@progbits
```

Disassembly

```
gcc -c -Wall -S func.c
```

```
gcc -c -g -Wall func.c

objdump -S -l func.o
```

```
-S, --source          Intermix source code with disassembly
-l, --line-numbers    Include line numbers and filenames in output
```

objdump -S example (1)

```
func.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <func>:
int func() {
   0:  55                    push   %rbp
   1:  48 89 e5                  mov    %rsp,%rbp
   int pow = 1;
   4:  c7 45 fc 01 00 00 00      movl   $0x1,-0x4(%rbp)
   int x = 0;
   b:  c7 45 f8 00 00 00 00      movl   $0x0,-0x8(%rbp)
```

objdump -S example (2)

```
   while (pow != 128) {
  12: eb 07                   jmp    1b <func+0x1b>
      pow = pow * 2;
  14: d1 65 fc                shll   -0x4(%rbp)
      x = x + 1;
  17: 83 45 f8 01             addl   $0x1,-0x8(%rbp)
int func() {
   int pow = 1;
   int x = 0;

   while (pow != 128) {
  1b: 81 7d fc 80 00 00 00       cmpl   $0x80,-0x4(%rbp)
  22: 75 f0                   jne    14 <func+0x14>
      pow = pow * 2;
      x = x + 1;
   }

   return (x);
  24: 8b 45 f8                mov    -0x8(%rbp),%eax
}
  27: 5d                 pop    %rbp
  28: c3                 retq
```

objdump -S -l example (1)

```
func.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <func>:
func():
/home/young/func.c:1
int func() {
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
/home/young/func.c:2
   int pow = 1;
   4:   c7 45 fc 01 00 00 00    movl   $0x1,-0x4(%rbp)
/home/young/func.c:3
   int x = 0;
   b:   c7 45 f8 00 00 00 00    movl   $0x0,-0x8(%rbp)
```

objdump -S -l example (2)

```
/home/young/func.c:5
   while (pow != 128) {
  12:   eb 07                   jmp    1b <func+0x1b>
/home/young/func.c:6
    pow = pow * 2;
  14:   d1 65 fc                shll   -0x4(%rbp)
/home/young/func.c:7
    x = x + 1;
  17:   83 45 f8 01             addl   $0x1,-0x8(%rbp)
/home/young/func.c:5
int func() {
   int pow = 1;
   int x = 0;

   while (pow != 128) {
  1b:   81 7d fc 80 00 00 00    cmpl   $0x80,-0x4(%rbp)
  22:   75 f0                   jne    14 <func+0x14>
/home/young/func.c:10
    pow = pow * 2;
    x = x + 1;
   }

   return (x);
  24:   8b 45 f8                mov    -0x8(%rbp),%eax
/home/young/func.c:11
}
  27:   5d                      pop    %rbp
  28:   c3                      retq
```

References

[1] Essential C, Nick Parlante
[2] Efficient C Programming, Mark A. Weiss
[3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
[4] C Language Express, I. K. Chun
[5] “A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux“
http://cseweb.ucsd.edu/~ricko/CSE131/teensyELF.htm