Smallest x86 ELF Hello World

(That I could achieve)

Final size: 142 bytes

Intro

This page is a combination tutorial/documentary about my attempts at creating the smallest x86 ELF binary that would execute saying Hello World on Ubuntu Linux. My first attempts started with C then progressed to x86 assembly and finally to a hexeditor. I ended up compromising and switching to a "Hi World" app instead in order to fit the string data into the elf magic number. The final result is a completely corrupted x86 ELF Binary that still runs.

From start to finish.

- The first thing you need to do is get an a proper environment setup.
  - Install Ubuntu (or a distro of your choice)
  - run: **sudo apt-get install g++ gcc nasm**
  - System versions

```
user@computer:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 8.04.1
Release:        8.04
Codename:       hardy
user@computer:~$ uname -a
Linux ryanh-desktop 2.6.24-19-generic #1 SMP Wed Jun 18 14:43:41 UTC 2008 i686 GNU/Linux
user@computer:~$ gcc --version
gcc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu7)
Copyright (C) 2007 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
user@computer:~$ nasm -version
NASM version 0.99.06-20071101 compiled on Nov 15 2007
```

- My first attempts started with C, the following is what I used for chello.c

  Code: **chello.c**

```c
#include <stdio.h>
int main() {
  printf ("Hi World\n");
  return 0;
}
```

- Command: gcc

```
user@computer:~$ gcc -o chello chello.c
user@computer:~$ ./chello
Hi World
```

- My initial executable was **6363 bytes**.
- You can use readelf to dump the ELF header from the executable.
- Command: readelf

```
user@computer:~$ readelf -h chello
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x80482f0
  Start of program headers:          52 (bytes into file)
  Start of section headers:          3220 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         7
  Size of section headers:           40 (bytes)
  Number of section headers:         36
  Section header string table index: 33
```

- ldd is useful for showing all the dynamic libraries an executable is linked to.
- Command: ldd

```
user@computer:~$ ldd chello
        linux-gate.so.1 =>  (0xb7f77000)
        libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7e18000)
        /lib/ld-linux.so.2 (0xb7f78000)
```

- file will give you a description of what a file is.
- Command: file

```
user@computer:~$ file chello
chello: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.6.8, dynamically linked (uses shared libs), not
```

- The "not stripped" returned from the file command means that the debugging symbols haven't been stripped from the excutable.
- Command: strip

```
user@computer:~$ strip -s chello
```

- After stripping the executable was now **2984 bytes**, still unacceptable! Time to take drastic measures...
- I scratched the C attempt and dropped using printf, instead opting for nasm x86 assembly.

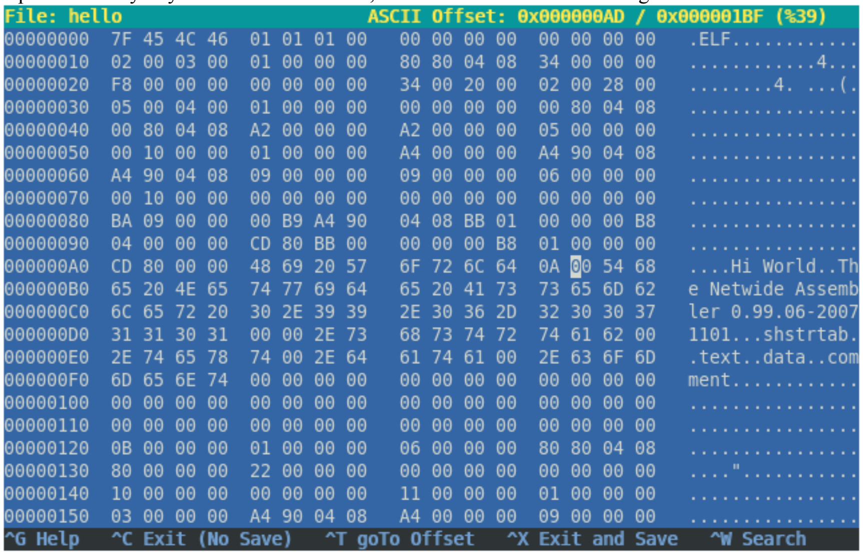
file: hello.asm

```
        SECTION .data
msg:    db "Hi World",10
len:    equ $-msg

        SECTION .text

        global main
main:
        mov     edx,len
        mov     ecx,msg
        mov     ebx,1
        mov     eax,4

        int     0x80
        mov     ebx,0
        mov     eax,1
        int     0x80
```

- Compiling the asm

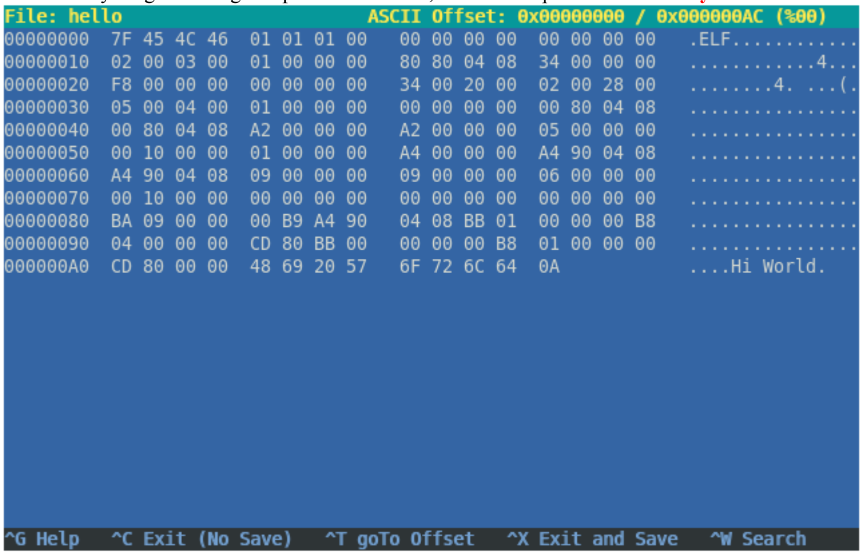
```
user@computer:~$ nasm -f elf hello.asm
user@computer:~$ gcc -o hello hello.o -nostartfiles -nostdlib -nodefaultlibs
user@computer:~$ strip -s hello
user@computer:~$ ./hello
Hi World
```

- Before stripping the file was **770 bytes** after stripping **448 bytes**. However there is still useless headers and sections to destroy.
- Open the binary in your favorite hex editor, I use the curses hexeditor and ghex2.

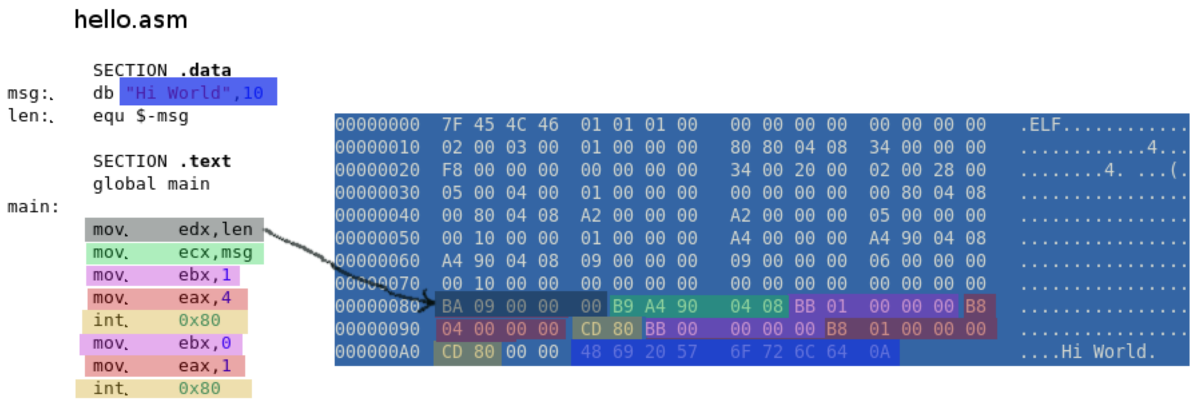


```
File: hello                              ASCII Offset: 0x000000AD / 0x000001BF (%39)
00000000  7F 45 4C 46  01 01 01 00   00 00 00 00  00 00 00 00   .ELF............
00000010  02 00 03 00  01 00 00 00   80 80 04 08  34 00 00 00   ............4...
00000020  F8 00 00 00  00 00 00 00   34 00 20 00  02 00 28 00   ........4. ...(.
00000030  05 00 04 00  01 00 00 00   00 00 00 00  00 80 04 08   ................
00000040  00 80 04 08  A2 00 00 00   A2 00 00 00  05 00 00 00   ................
00000050  00 10 00 00  01 00 00 00   A4 00 00 00  A4 90 04 08   ................
00000060  A4 90 04 08  09 00 00 00   09 00 00 00  06 00 00 00   ................
00000070  00 10 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000080  BA 09 00 00  00 B9 A4 90   04 08 BB 01  00 00 00 B8   ................
00000090  04 00 00 00  CD 80 BB 00   00 00 00 B8  01 00 00 00   ................
000000A0  CD 80 00 00  48 69 20 57   6F 72 6C 64  0A 00 54 68   ....Hi World..Th
000000B0  65 20 4E 65  74 77 69 64   65 20 41 73  73 65 6D 62   e Netwide Assemb
000000C0  6C 65 72 20  30 2E 39 39   2E 30 36 2D  32 30 30 37   ler 0.99.06-2007
000000D0  31 31 30 31  00 00 2E 73   68 73 74 72  74 61 62 00   1101...shstrtab.
000000E0  2E 74 65 78  74 00 2E 64   61 74 61 00  2E 63 6F 6D   .text..data..com
000000F0  6D 65 6E 74  00 00 00 00   00 00 00 00  00 00 00 00   ment............
00000100  00 00 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000110  00 00 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000120  0B 00 00 00  01 00 00 00   06 00 00 00  80 80 04 08   ................
00000130  80 00 00 00  22 00 00 00   00 00 00 00  00 00 00 00   ...."...........
00000140  10 00 00 00  00 00 00 00   11 00 00 00  01 00 00 00   ................
00000150  03 00 00 00  A4 90 04 08   A4 00 00 00  09 00 00 00   ................
^G Help   ^C Exit (No Save)   ^T goTo Offset   ^X Exit and Save   ^W Search
```

- Delete everything including and past offset 0xAD, this will drop it down to **173 bytes**



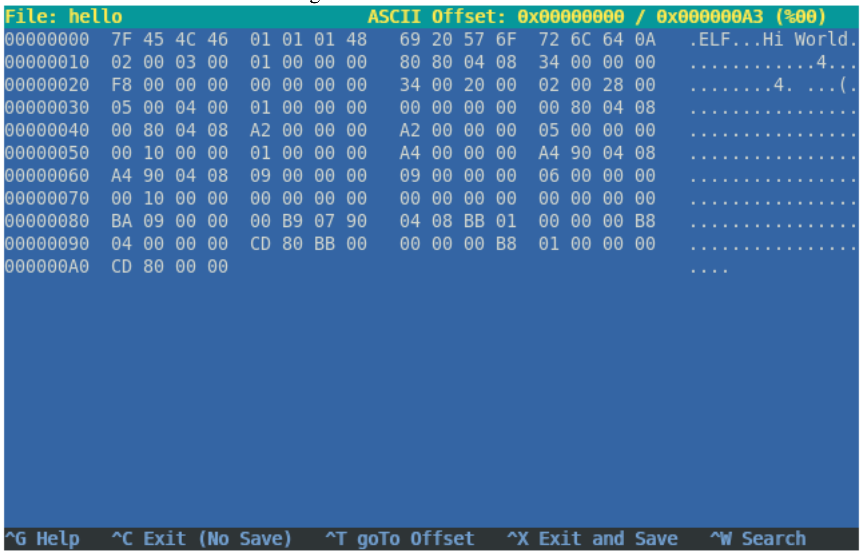
```
File: hello                              ASCII Offset: 0x00000000 / 0x000000AC (%00)
00000000  7F 45 4C 46  01 01 01 00   00 00 00 00  00 00 00 00   .ELF............
00000010  02 00 03 00  01 00 00 00   80 80 04 08  34 00 00 00   ............4...
00000020  F8 00 00 00  00 00 00 00   34 00 20 00  02 00 28 00   ........4. ...(.
00000030  05 00 04 00  01 00 00 00   00 00 00 00  00 80 04 08   ................
00000040  00 80 04 08  A2 00 00 00   A2 00 00 00  05 00 00 00   ................
00000050  00 10 00 00  01 00 00 00   A4 00 00 00  A4 90 04 08   ................
00000060  A4 90 04 08  09 00 00 00   09 00 00 00  06 00 00 00   ................
00000070  00 10 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000080  BA 09 00 00  00 B9 A4 90   04 08 BB 01  00 00 00 B8   ................
00000090  04 00 00 00  CD 80 BB 00   00 00 00 B8  01 00 00 00   ................
000000A0  CD 80 00 00  48 69 20 57   6F 72 6C 64  0A            ....Hi World.
^G Help   ^C Exit (No Save)   ^T goTo Offset   ^X Exit and Save   ^W Search
```



```
hello.asm

        SECTION .data
msg:    db "Hi World",10
len:    equ $-msg

        SECTION .text
        global main
main:
        mov     edx,len
        mov     ecx,msg
        mov     ebx,1
        mov     eax,4
        int     0x80
        mov     ebx,0
        mov     eax,1
        int     0x80
```

```
00000000  7F 45 4C 46  01 01 01 00   00 00 00 00  00 00 00 00   .ELF............
00000010  02 00 03 00  01 00 00 00   80 80 04 08  34 00 00 00   ............4...
00000020  F8 00 00 00  00 00 00 00   34 00 20 00  02 00 28 00   ........4. ...(.
00000030  05 00 04 00  01 00 00 00   00 00 00 00  00 80 04 08   ................
00000040  00 80 04 08  A2 00 00 00   A2 00 00 00  05 00 00 00   ................
00000050  00 10 00 00  01 00 00 00   A4 00 00 00  A4 90 04 08   ................
00000060  A4 90 04 08  09 00 00 00   09 00 00 00  06 00 00 00   ................
00000070  00 10 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000080  BA 09 00 00  00 B9 A4 90   04 08 BB 01  00 00 00 B8   ................
00000090  04 00 00 00  CD 80 BB 00   00 00 00 B8  01 00 00 00   ................
000000A0  CD 80 00 00  48 69 20 57   6F 72 6C 64  0A            ....Hi World.
```

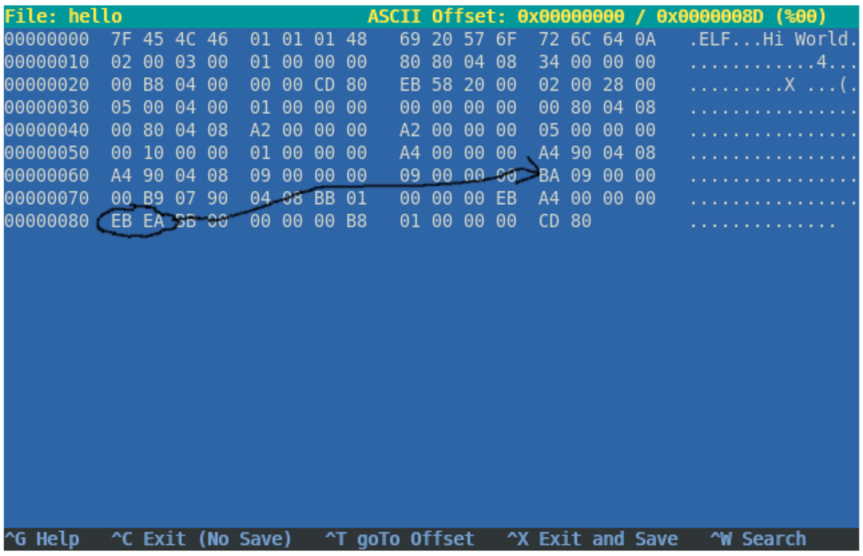
-
- Move 0xA4-0xAC to 0x7 and Change offset 0x86 from 0xA4 to its new location 0x07. Delete 0xA2 and 0xA3



```
File: hello                        ASCII Offset: 0x00000000 / 0x000000A3 (%00)
00000000  7F 45 4C 46  01 01 01 48   69 20 57 6F  72 6C 64 0A   .ELF...Hi World.
00000010  02 00 03 00  01 00 00 00   80 80 04 08  34 00 00 00   ............4...
00000020  F8 00 00 00  00 00 00 00   34 00 20 00  02 00 28 00   ........4. ...(.
00000030  05 00 04 00  01 00 00 00   00 00 00 00  00 80 04 08   ................
00000040  00 80 04 08  A2 00 00 00   A2 00 00 00  05 00 00 00   ................
00000050  00 10 00 00  01 00 00 00   A4 00 00 00  A4 90 04 08   ................
00000060  A4 90 04 08  09 00 00 00   09 00 00 00  06 00 00 00   ................
00000070  00 10 00 00  00 00 00 00   00 00 00 00  00 00 00 00   ................
00000080  BA 09 00 00  00 B9 07 90   04 08 BB 01  00 00 00 B8   ................
00000090  04 00 00 00  CD 80 BB 00   00 00 00 B8  01 00 00 00   ................
000000A0  CD 80 00 00                                           ....

^G Help   ^C Exit (No Save)   ^T goTo Offset   ^X Exit and Save   ^W Search
```

-
- The file should be **164 bytes** and now its time to enter the twilight zone... The rest is a lot to explain, basically I attempted to find what I could change in the elf head with out having it segfault on me.I added some jmps and completely corrupted the executable, however it still runs :). Here is some useful information: In x86 0xD9D0 is nop or no operation, useful for just filling space if you need to. 0xEB followed by a single signed byte is a relative jmp. Really you should read the intel docs on x86 instructions A-M N-Z .

-
  ```
  typedef struct {
          unsigned char   e_ident[EI_NIDENT];
          Elf32_Half      e_type;
          Elf32_Half      e_machine;
          Elf32_Word      e_version;
          Elf32_Addr      e_entry;
          Elf32_Off       e_phoff;
          Elf32_Off       e_shoff;
          Elf32_Word      e_flags;
          Elf32_Half      e_ehsize;
          Elf32_Half      e_phentsize;
          Elf32_Half      e_phnum;
          Elf32_Half      e_shentsize;
          Elf32_Half      e_shnum;
          Elf32_Half      e_shtrndx;
  } Elf32_Ehdr;
  ```

- 

```
File: hello                        ASCII Offset: 0x00000000 / 0x0000008D (%00)
00000000  7F 45 4C 46  01 01 01 48   69 20 57 6F  72 6C 64 0A    .ELF...Hi World.
00000010  02 00 03 00  01 00 00 00   80 80 04 08  34 00 00 00    ............4...
00000020  00 B8 04 00  00 00 CD 80   EB 58 20 00  02 00 28 00    .........X ...(.
00000030  05 00 04 00  01 00 00 00   00 00 00 00  00 80 04 08    ................
00000040  00 80 04 08  A2 00 00 00   A2 00 00 00  05 00 00 00    ................
00000050  00 10 00 00  01 00 00 00   A4 00 00 00  A4 90 04 08    ................
00000060  A4 90 04 08  09 00 00 00   09 00 00 00  BA 09 00 00    ................
00000070  00 B9 07 90  04 08 BB 01   00 00 00 EB  A4 00 00 00    ................
00000080  EB EA BB 00  00 00 00 B8   01 00 00 00  CD 80          ..............

^G Help   ^C Exit (No Save)   ^T goTo Offset   ^X Exit and Save   ^W Search
```

Conclusion.

Final size: 142 bytes

helloworld.tar.gz

I am certain that there are ways to get it even smaller. There may also be more things that can be removed from the header to increase size, but I didn't spend the enough time fully researching the ELF header format. Another option might be to use the a.out format instead of ELF may allow you to get even smaller.

Comments, suggestions, and critical criticism accepted: henszey@gmail.com

Home