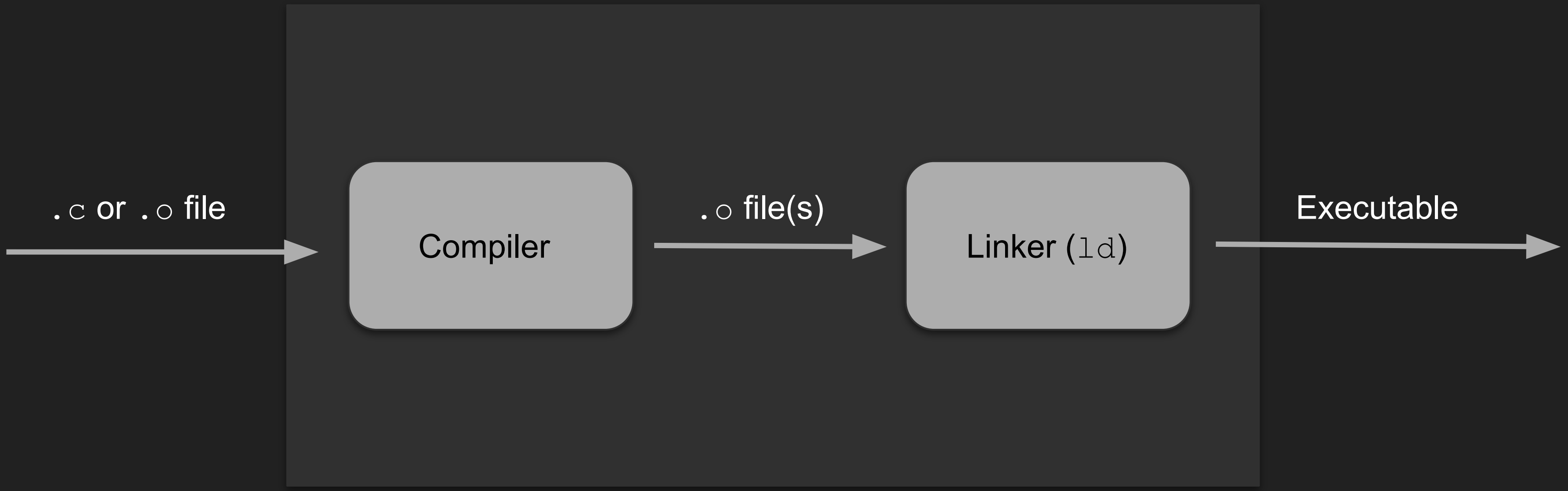
Code golfing with ELF executables

What is GCC?

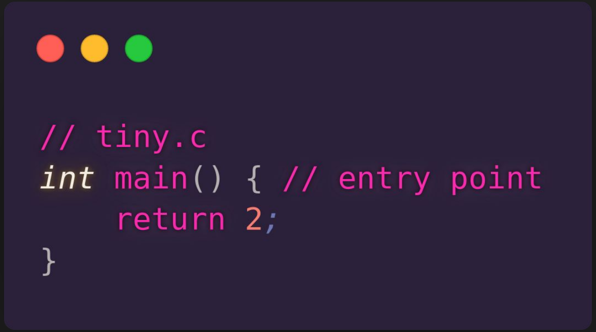
- Compiles C, C++, and a few others
- Uses `libc`, which contains system functions
- Acts as a frontend for the GNU linker
- Links object `.o` files into a binary executable (ELF)

GCC

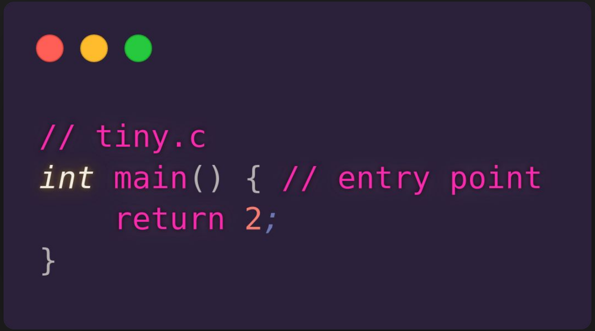


What is the simplest program?

- Programs returning 0 and 1 are already taken in *NIX (`true` and `false`)
- What about returning 2?



```c
// tiny.c
int main() { // entry point
    return 2;
}
```



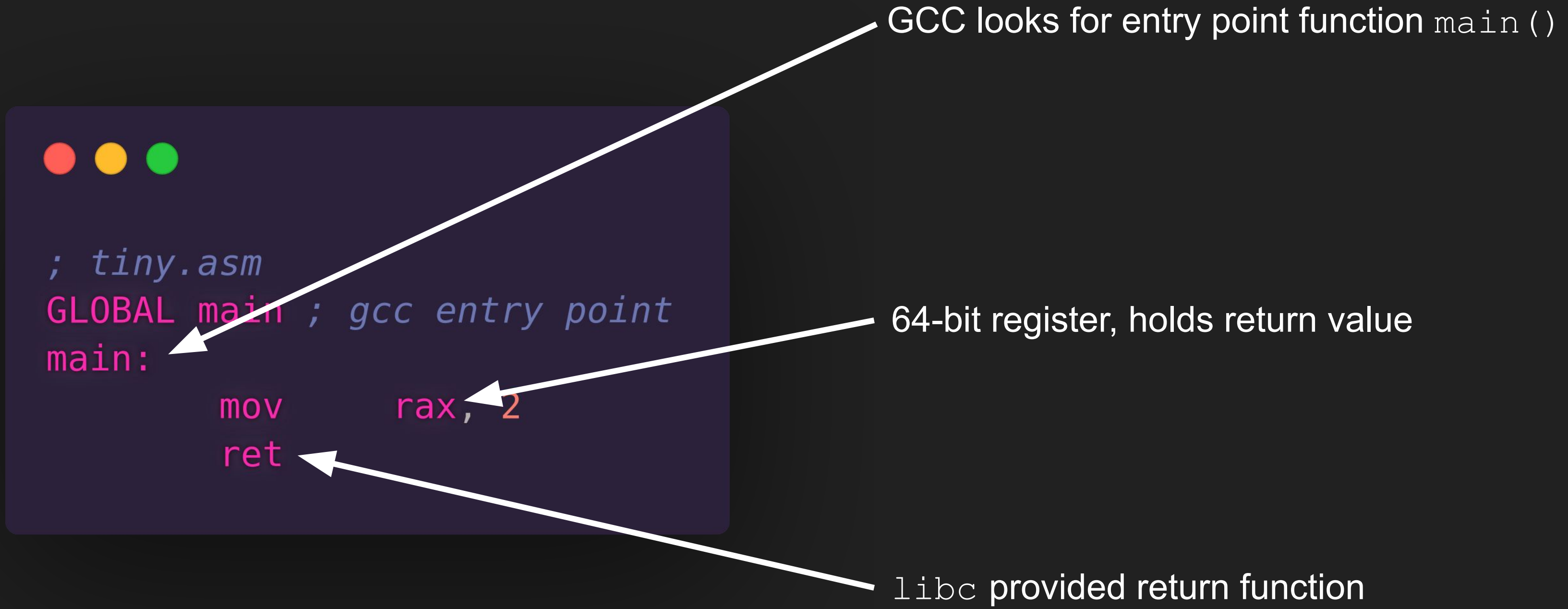
```c
// tiny.c
int main() { // entry point
    return 2;
}
```

Optimising GCC

- Strip executable `-s`
- Optimise executable `-O3`

Optimising GCC

- Strip executable `-s`
- Optimise executable `-O3`
- Stop using C

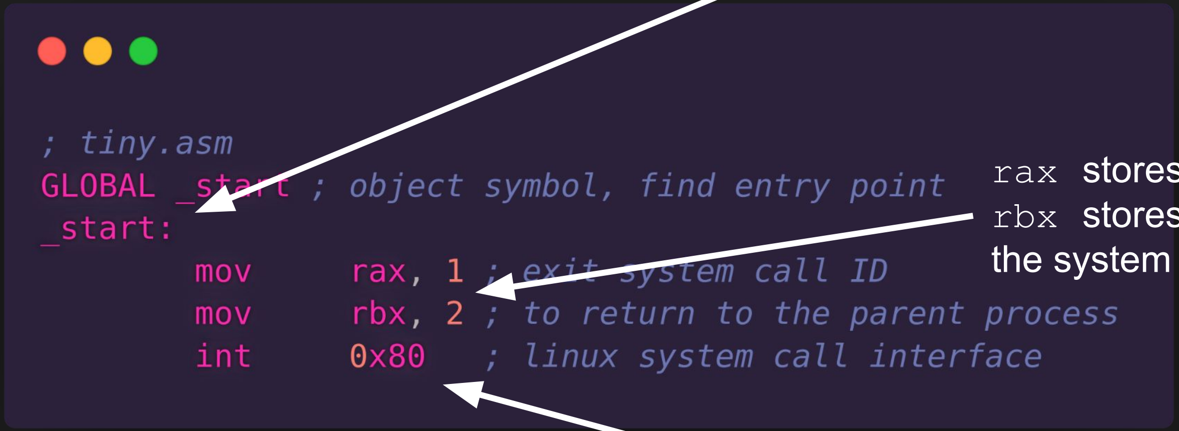


```
; tiny.asm
GLOBAL main ; gcc entry point
main:
        mov     rax, 2
        ret
```

GCC looks for entry point function main()
64-bit register, holds return value
libc provided return function

Optimising GCC

- Strip executable `-s`
- Optimise executable `-O3`
- Stop using C
- Stop using `libc`



```
; tiny.asm
GLOBAL _start ; object symbol, find entry point
_start:
        mov     rax, 1 ; exit system call ID
        mov     rbx, 2 ; to return to the parent process
        int     0x80   ; linux system call interface
```

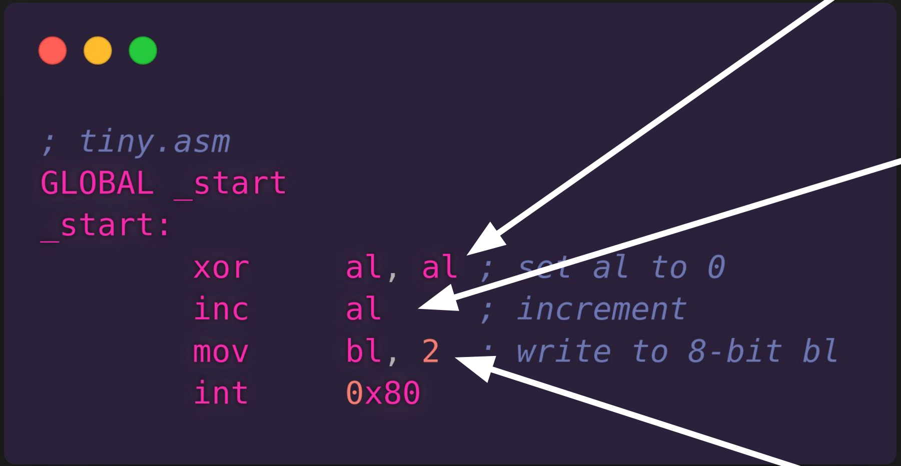
Name of `ld`'s default entry point

`rax` stores the system call ID for exit,
`rbx` stores the first parameter passed to the system call

Starts system call - passes values stored in `rax` and `rbx`

Optimising GCC

- Strip executable `-s`
- Optimise executable `-O3`
- Stop using C
- Stop using `libc`
- Stop using 64-bit registers



```
; tiny.asm
GLOBAL _start
_start:
        xor     al, al ; set al to 0
        inc     al     ; increment
        mov     bl, 2  ; write to 8-bit bl
        int     0x80
```

Using 8-bit `al` instead of 64-bit `rax`

Set `al` to 1 by `xor`ing to 0, then incrementing - smaller operations

Write to 8-bit `bl` instead of `rbx`

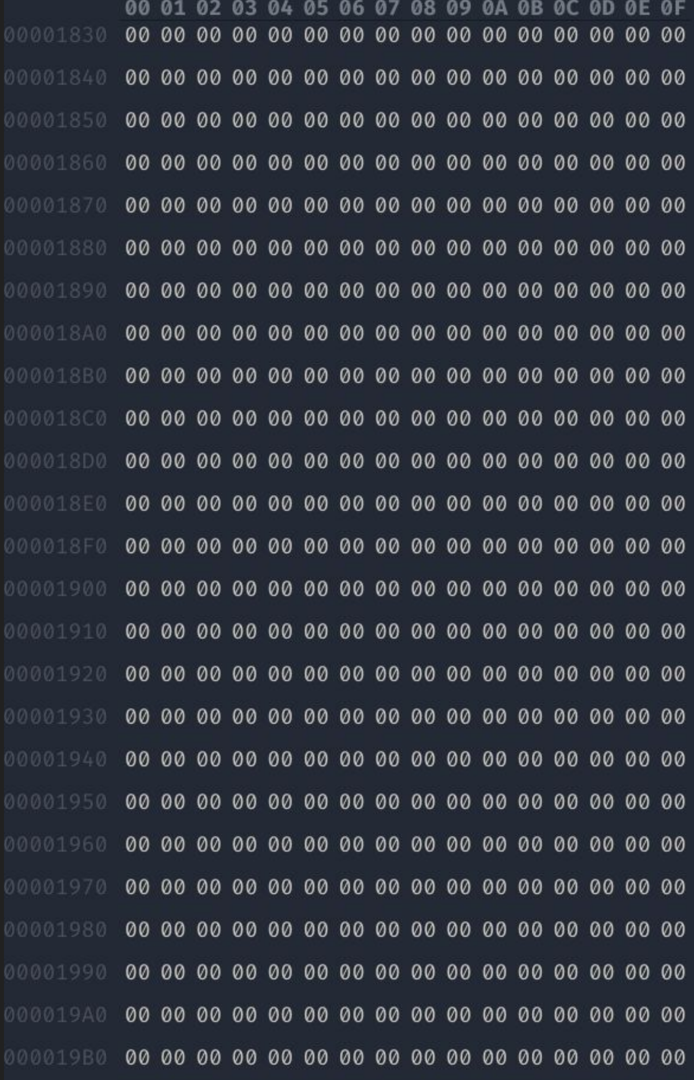
Note: 8-bit registers represent the 8 LSBs of their 64-bit counterparts

Optimising GCC

- Strip executable `-s`
- Optimise executable `-O3`
- Stop using C
- Stop using `libc`
- Stop using 64-bit registers
- Stop using GCC

86.4%

Zeroed padding space



```
         00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00001830 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001880 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001890 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000018A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000018B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000018C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000018D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000018E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000018F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001930 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001940 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001950 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001960 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001970 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001980 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001990 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000019B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Optimising GCC

- Strip executable `-s`
- Optimise executable `-O3`
- Stop using C
- Stop using `libc`
- Stop using 64-bit registers
- Stop using GCC
- Remove section header table

ELF Structure

- Must include ELF header containing information eg. 32/64 bit, endianness, type of ELF, table locations, etc
- We are generating Loadable (executable) ELFs
- Section header table only used by compiler and linker - we can remove it

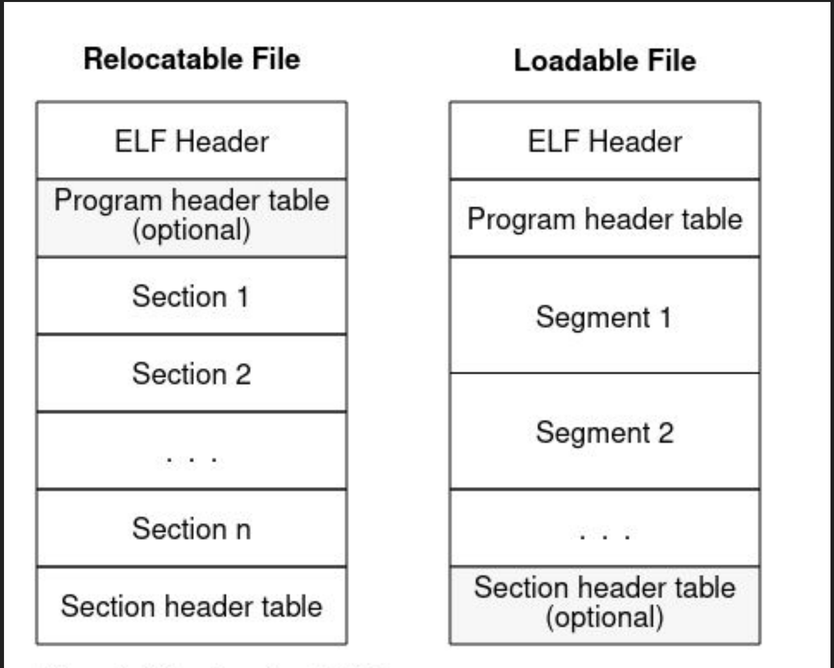
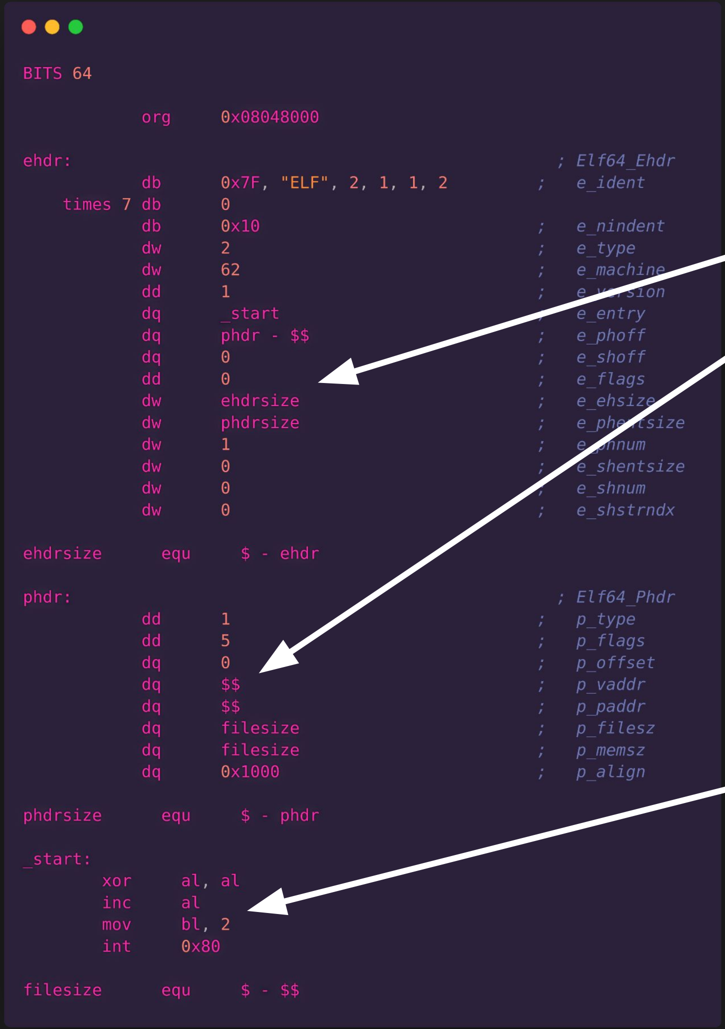


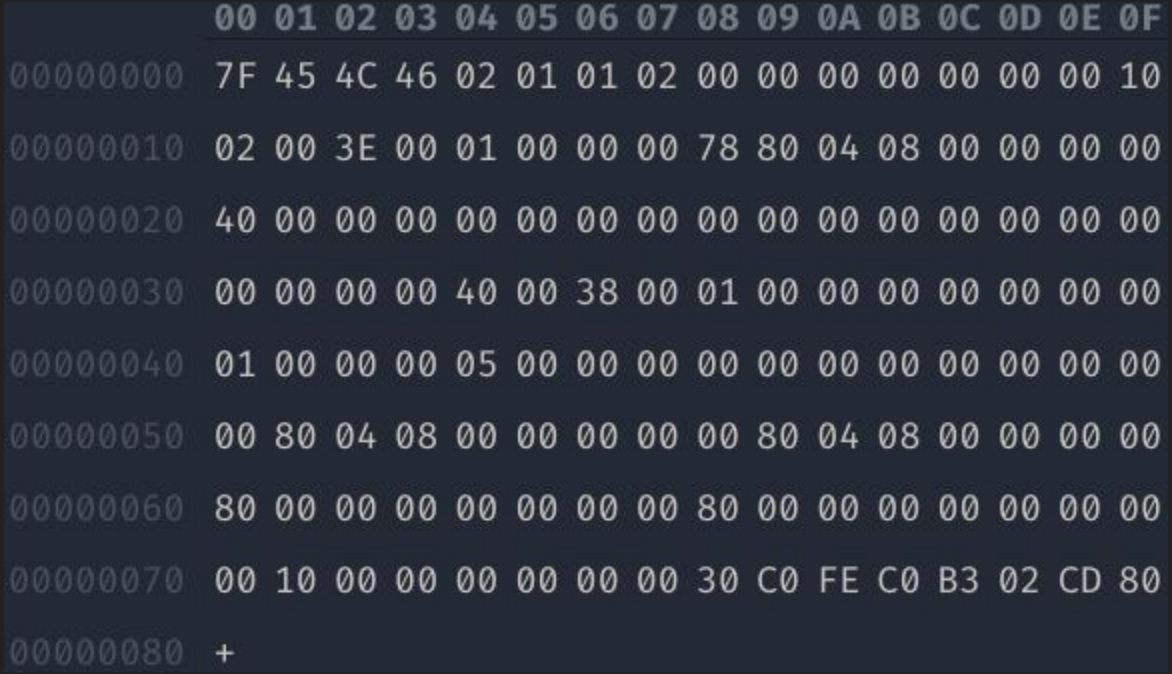
Figure 1. Structure of an ELF File



```
BITS 64

            org     0x08048000

ehdr:                                                   ; Elf64_Ehdr
            db      0x7F, "ELF", 2, 1, 1, 2           ;   e_ident
    times 7 db      0
            db      0x10                              ;   e_nindent
            dw      2                                 ;   e_type
            dw      62                                ;   e_machine
            dd      1                                 ;   e_version
            dq      _start                            ;   e_entry
            dq      phdr - $$                         ;   e_phoff
            dq      0                                 ;   e_shoff
            dd      0                                 ;   e_flags
            dw      ehdrsize                          ;   e_ehsize
            dw      phdrsize                          ;   e_phentsize
            dw      1                                 ;   e_phnum
            dw      0                                 ;   e_shentsize
            dw      0                                 ;   e_shnum
            dw      0                                 ;   e_shstrndx

ehdrsize      equ     $ - ehdr

phdr:                                                   ; Elf64_Phdr
            dd      1                                 ;   p_type
            dd      5                                 ;   p_flags
            dq      0                                 ;   p_offset
            dq      $$                                ;   p_vaddr
            dq      $$                                ;   p_paddr
            dq      filesize                          ;   p_filesz
            dq      filesize                          ;   p_memsz
            dq      0x1000                            ;   p_align

phdrsize      equ     $ - phdr

_start:
        xor     al, al
        inc     al
        mov     bl, 2
        int     0x80

filesize      equ     $ - $$
```

Instantiating header values

Our assembly code to run



```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  7F 45 4C 46 02 01 01 02 00 00 00 00 00 00 00 10
00000010  02 00 3E 00 01 00 00 00 78 80 04 08 00 00 00 00
00000020  40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030  00 00 00 00 40 00 38 00 01 00 00 00 00 00 00 00
00000040  01 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00
00000050  00 80 04 08 00 00 00 00 00 80 04 08 00 00 00 00
00000060  80 00 00 00 00 00 00 00 80 00 00 00 00 00 00 00
00000070  00 10 00 00 00 00 00 00 30 C0 FE C0 B3 02 CD 80
00000080  +
```

Felix B.

https://fbcf.xyz/

efbicief#3835

Thank you!