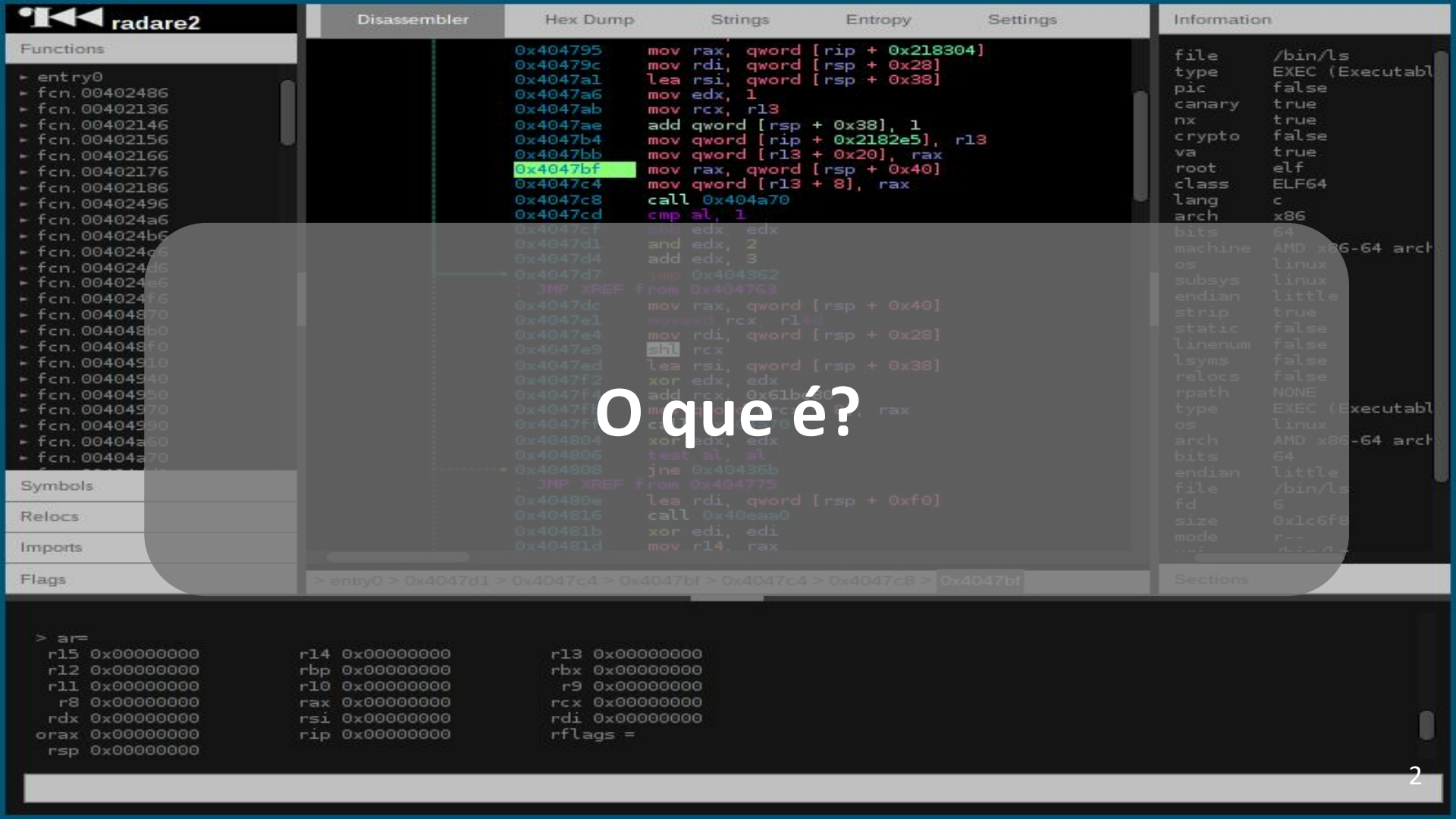
ICMC USP
SÃO CARLOS

GANESH

Engenharia Reversa

asrever ed etnerf


radare2
Disassembler
Hex Dump
Strings
Entropy
Settings
Information
Functions
Symbols
Relocs
Imports
Flags
Sections
O que é?

Baby Steps

- file
- strings

```
$ file hello
Hello: ELF 64-bit LSB shared obj…

$ strings hello
/lib64/ld-linux-x86-64.so.2…
```

Como fazer?

- O que um computador entende?
- Como ele entende?
- Quando ele entende?

Como fazer?

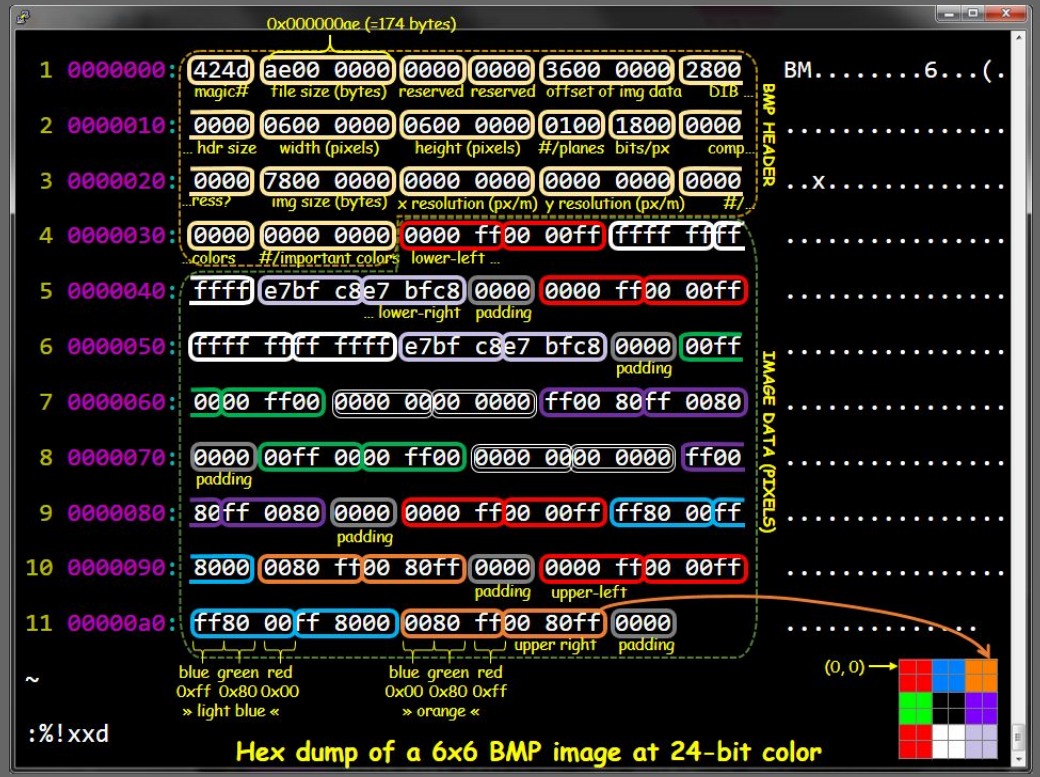
- O que um computador entende?
- Como ele entende?

Exemplos

- Printf
  - convencao de C
- Saidas
  - 0 - stdin
  - 1 - stdout
  - 2 - stderr

Tirando Informações de Arquivos

- “Magic Numbers”
- Pontos de entradas: Memória Real x Virtual
- Mapeamento de memória



Binários

- O que um compilador faz?
- O que é um arquivo binário?
- Como uma CPU executa um arquivo binário?

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

Comandos úteis

- objdump
- readelf
- Hexdump (xxd)

```
$ xxd hello | less
00000000: 7f45 4c46 0201 … ELF

$ objdump -Mintel -D ./hello | grep “main>:” -A 8
400546:      55          push rbp

$ readelf -h ./hello | grep Entry
  Entry point address:           0x400450
```

Assembly, disassemblers e debuggers

- Assembly é uma família de linguagens!
  - Focaremos no Intel x86
- Disassemblers
  - Transformam o arquivo binário em algo “entendível”
- Debuggers
  - Te permitem ver o que acontece “por dentro” enquanto um programa executa

Registradores - Controle de Armazenamento

Propósito geral

- eax: operações aritméticas e acumulador
- ebx: registrador base para stack
- ecx: contador em loops
- edx: endereços de dados

Hello World!

```
;; Program Hello W0rld
section .text
global _start

_start:
        mov     edx,len                         ;message length
        mov     ecx,msg                         ;message to write
        mov     ebx,1                           ;file descriptor (stdout)
        mov     eax,4                           ;system call number (sys_write)
        int     0x80                            ;call kernel

        mov     eax,1                           ;system call number (sys_exit)
        int     0x80                            ;call kernel

section     .data
        msg     db  'Hello, world!',0xa         ;our dear string
        len     equ $ - msg                     ;length of our dear string


          ;db - defined bytes: bytes definidos
          ;0xa - \n
```

Instruções Assembly

- Intel vs AT&T (Intel é melhor)
  - Intel: <inst> <dst>,<src>
- instruções aritmeticas:
  - mov
  - add
  - sub
  - xor
  - and

```
$ objdump -Mintel -D ./hello | grep “main>:” -A 8
400546:   55                    push   rbp
400547:   48 89 e5              mov    rbp,rsp
40054a:   bf e4 05 40 00        mov    edi,0x4005e4
40054f:   e8 dc fe ff ff        call   400430
<puts@plt>
400554:   bf 00 00 00 00        mov    edi,0x0
400559:   e8 e2 fe ff ff        call   400440
<exit@plt>
```

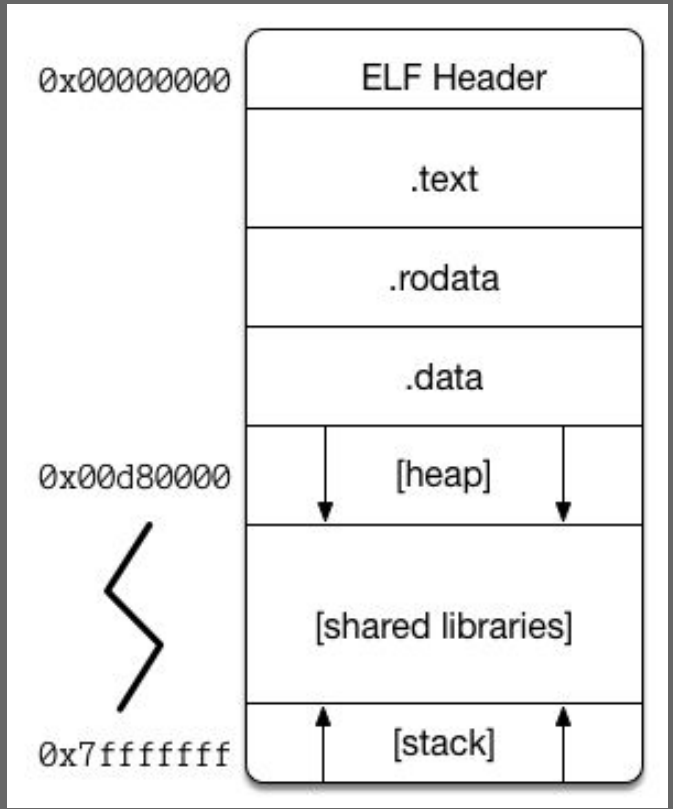
Hello Challenge!

```
;;first ASM challenge
section .text
global _start

_start:
     mov     ecx,4
     mov     eax,16
     add     ebx,4
     add     ecx,eax
     sub     ebx,eax
     xor     ebx,ebx
     mov     eax,1
     int     0x80
```

Memória

- .text (0x400000)
  - Seção com código executável
- .(ro) data
  - Read only data - variáveis inicializadas
- heap
  - Alocação dinâmica (malloc)
- shared libraries
  - Bibliotecas do C
- stack (0x7fffffff)
  - Utilizada para a execução

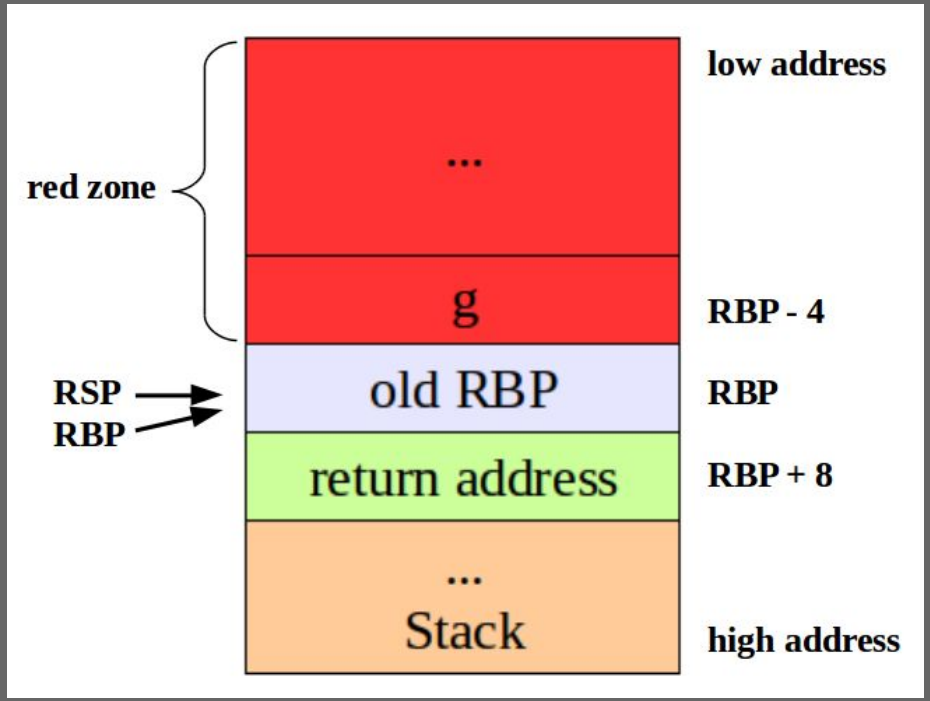


Hello Challenge!

```
;;segundo desafio ASM
section .text
global _start

_start:
     mov eax,-1
     mov ebx,3
     xor eax,ebx
     add ebx,ebx
     xor eax,ebx
     mov ebx,10
     and eax,ebx
     xor ebx,ebx
     cmp eax,ebx
     jz isZero
notZero:
     mov ebx,eax
     jmp exit
isZero:
     mov ebx,-1
exit:
     mov eax,1
     int 0x80
```

Funcionamento da pilha



- Pilha
  - rsp: stack pointer (topo)
  - rbp: base pointer (base)

Links úteis e leituras complementares

- bit.ly/revganesh2019
- https://shellterlabs.com/en/training/get-started/art-reverse-engineering/
- https://beginners.re/RE4B-EN.pdf
- http://www.sig9.com/articles/att-syntax

GANESH

Grupo de Segurança da Informação
ICMC / USP - São Carlos, SP
http://ganesh.icmc.usp.br/
ganesh@icmc.usp.br