Aula 02

Engenharia Reversa e Análise de Malware

Ronaldo Pinheiro de Lima
crimescibernéticos.com@gmail.com

Aula 02

2. Estruturas Internas dos Softwares
2.1. Níveis de abstração
2.2. Engenharia Reversa
2.3. A Arquitetura x86
2.4. Memória Principal (RAM)
2.5. Instruções
2.6. Opcodes e Endianness (Ordenação)
2.7. Operandos
2.8. Registradores
2.9. A pilha (Stack)
2.10. Chamadas de Funções (Function Calls)
2.11. Layout da Pilha (stack layout)

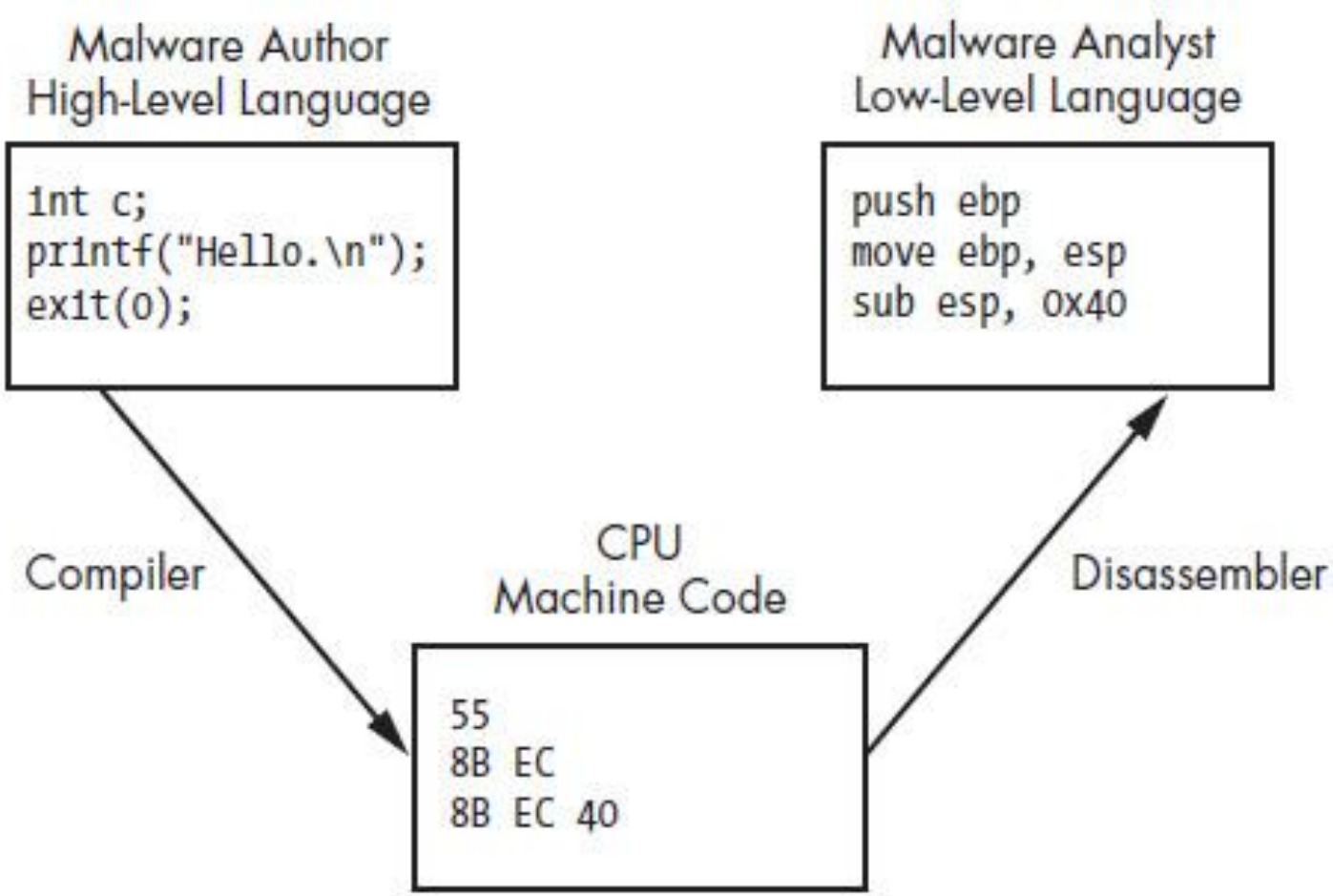
Estruturas Internas dos Softwares

- complexidade
- camadas de abstração
- programação menos baixo-nível

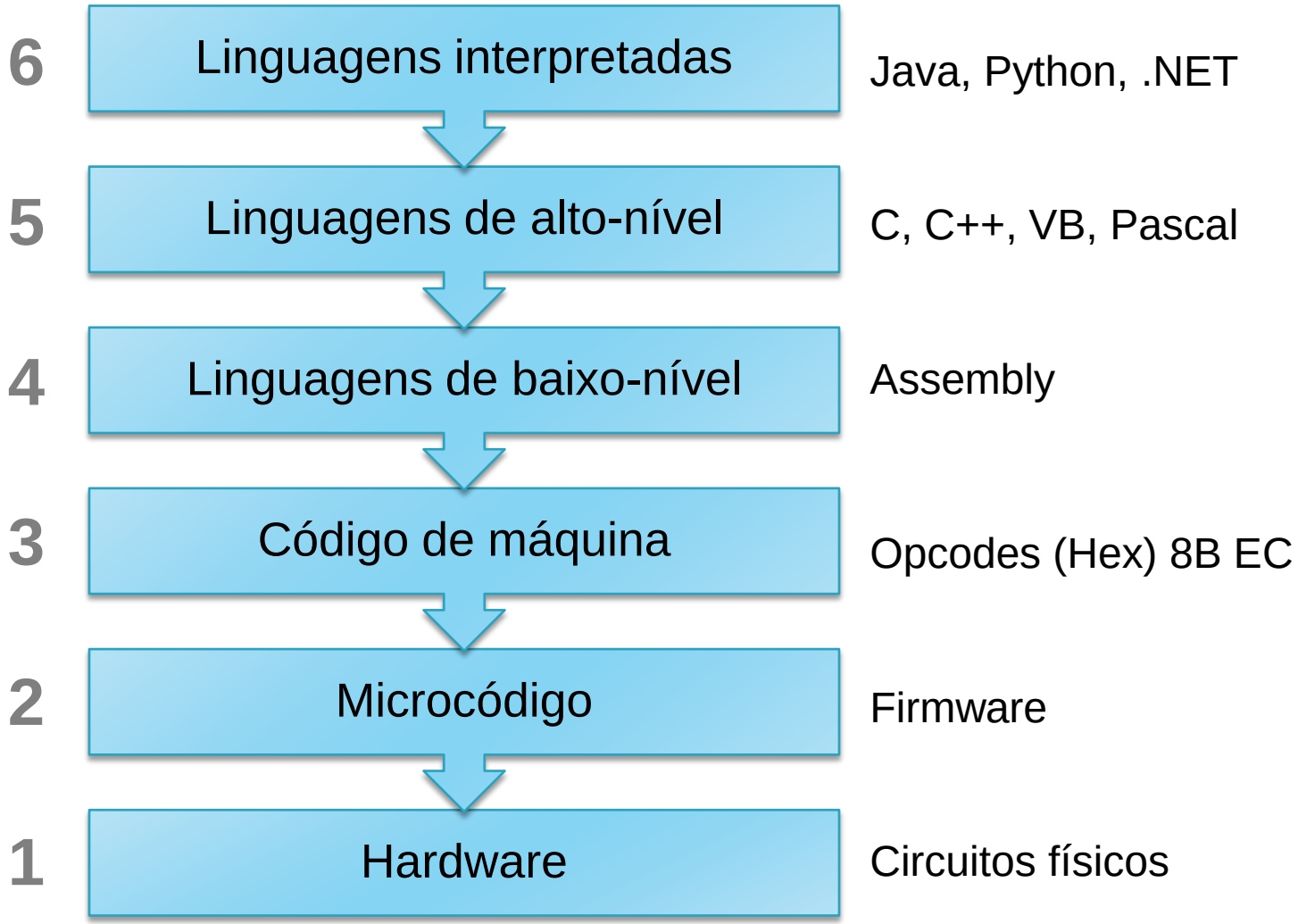
Níveis de abstração

- esconder detalhes para a camada acima
- compilação: alto-nível para baixo-nível

Níveis de abstração



Níveis de abstração



Engenharia Reversa

- **compilação:**

  alto-nível (5) p/ código de máquina (3)

- **malware no disco**:

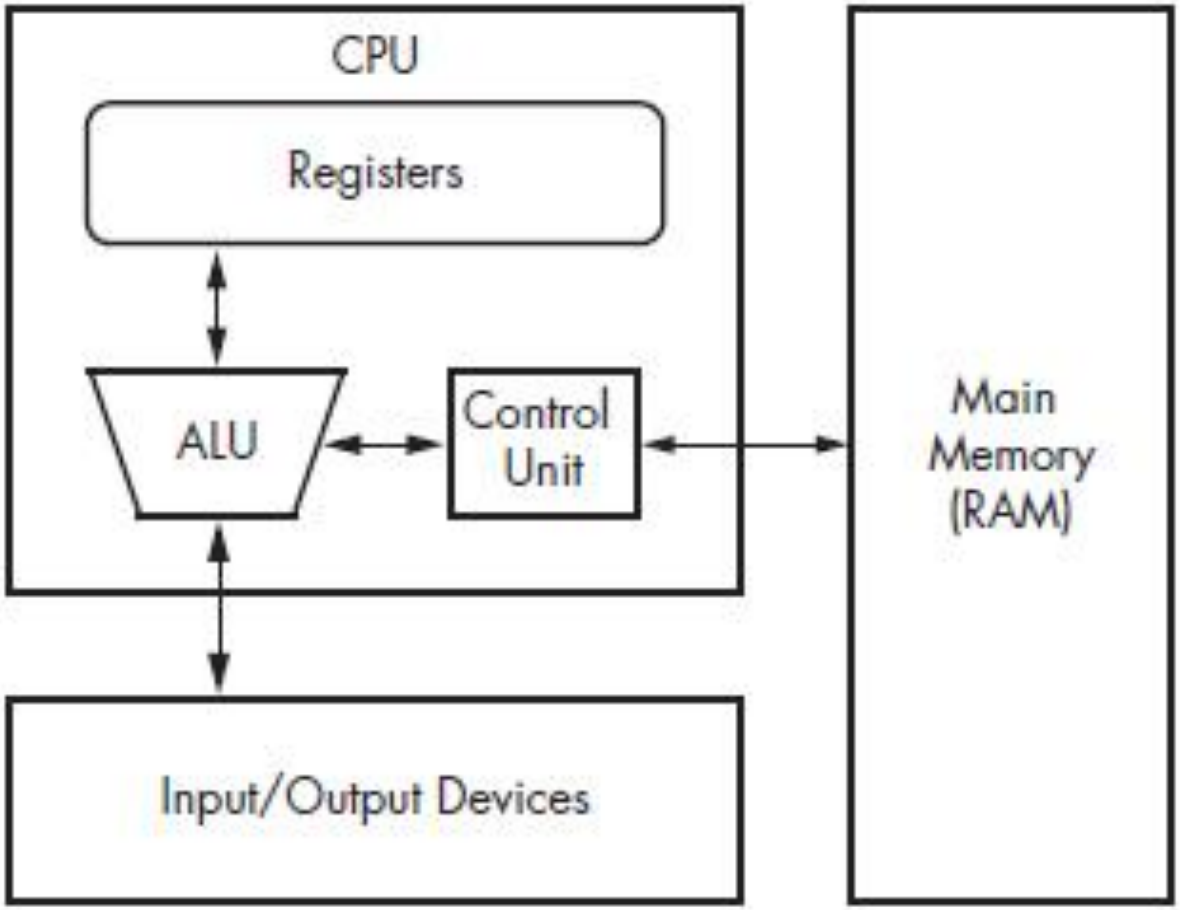
  código de máquina (3)

- **disassembly**:

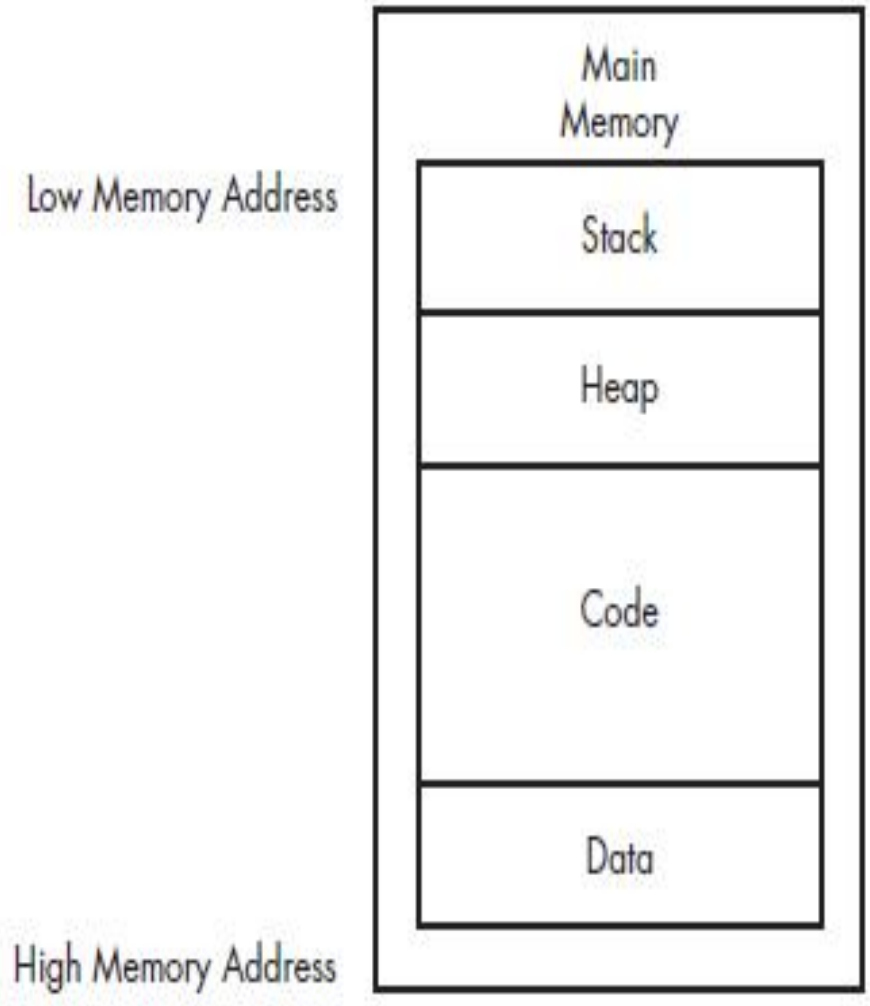
  código de máquina (3) p/ ling. baixo-nível (4)

- assembly: classe de linguagens (processador)
- malwares utilizam mais Intel x86 32-bit

Arquitetura x86



Memória Principal (RAM)



Stack (pilha): variáveis locais e parâmetros de funções, controle do fluxo do programa.

Heap: memória dinâmica do programa.

Código: instruções executadas pela CPU.

Dados: dados globais, estáticos, do programa.

Formato das Instruções

Mnemônico	Operando destino	Operando origem
MOV	ECX,	0x42
LEA	EDX,	[EBP-28]
SUB	ESP,	0x0C
PUSH	EBP	
XOR	EAX,	EAX
JMP	0X41414141	

Opcodes

Instrução	MOV ECX,	0x42
Opcodes	B9	42 00 00 00

Endianness (ordenação dos bytes)

- **little-endian**
  - arquitetura Intel x86, AMD64
  - byte menos significativo primeiro
  - ex.: 127.0.0.1 – 0x0100007F
- **big-endian**
  - TCP/IP, RISC, Motorola
  - byte mais significativo primeiro
  - ex.: 127.0.0.1 – 0x7F000001

Operandos

Imediatos	0x42	0x00401828	0x0C
Registradores	EAX	ECX	EBP
Endereços memória	[0x0012F8D4]	[ECX]	[EBP+0x08]

Registradores

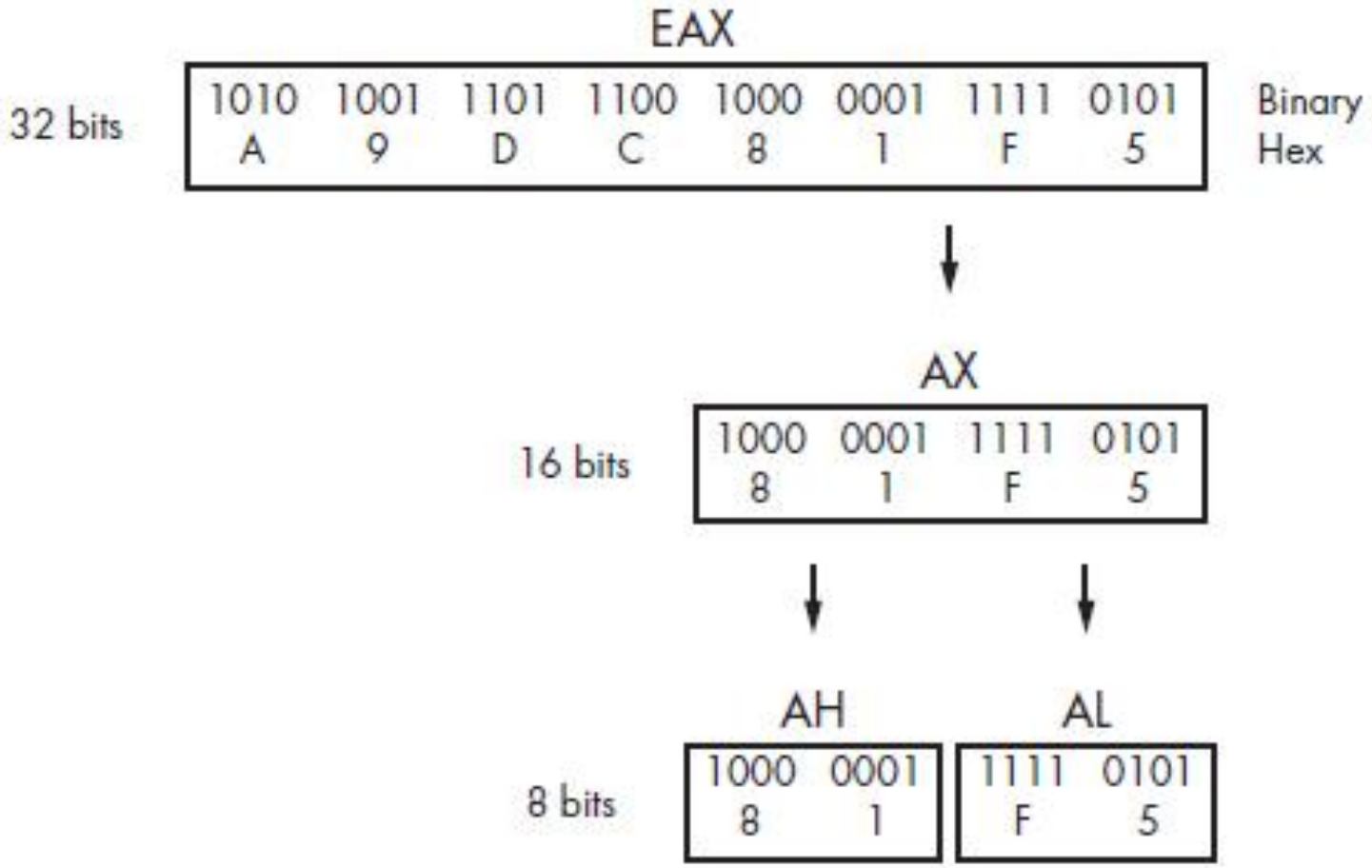
- **uso geral:** execução do programa
- **segmentados:** mapear seções da memória
- **flags:** tomadas de decisão
- **instruction pointer:** próxima instrução a ser executada

Registradores x86

Registradores de uso geral	Registradores segmentados	Flags de status	Instruction pointers
EAX (AX, AH, AL)	CS	EFLAGS	EIP
EBX (BX, BH, BL)	SS		
ECX (CX, CH, CL)	DS		
EDX (DX, DH, DL)	ES		
EBP (BP)	FS		
ESP (SP)	GS		
ESI (SI)			

- **uso geral:** apesar do nome seguem alguns padrões
- conveções dos compiladores: retorno de funções no EAX, divisão com EAX e EDX, controle da pilha ESP e EBP, etc.

Registradores x86



Flags: EFLAGS – registrador de status

- 32 bits, cada bit é uma flag
- controlar a execução e operações da CPU
- ligada: 1, desligada: 0

- **ZF:** 1 quando operação igual a 0, outros resultados é 0
- **CF:** 1 quando resultado muito grande ou pequeno p/ operando, outros resultados é 0
- **SF:** 1 quando resultado é negativo e 0 quando positivo
- **TF:** debugging, se for 1 a CPU executa debugging

EIP, Instruction Pointer

- contador do programa
- endereço da próxima instrução a ser executada

Em um ataque, controle o EIP e seja feliz!

A Pilha (Stack)

- memória para funções: variáveis locais, parâmetros
- controle do fluxo do programa
- PUSH (empilha), POP (desempilha)
- PUSHA e POPA (16-bit), PUSHAD e POPAD (32-bit) - todos
- LIFO, last in, first out
- ESP (stack pointer) e EBP (base pointer)
- alocação de endereços top-down
- [EBP-valor] = variável local
- [EBP+valor] = parâmetro

Chamadas de funções

- execução do programa é transferida para a função
- forma da pilha depende do compilador
- mais comum: cdecl

prólogo:

```
PUSH EBP
MOV EBP, ESP
SUB ESP, valor
```

epílogo:

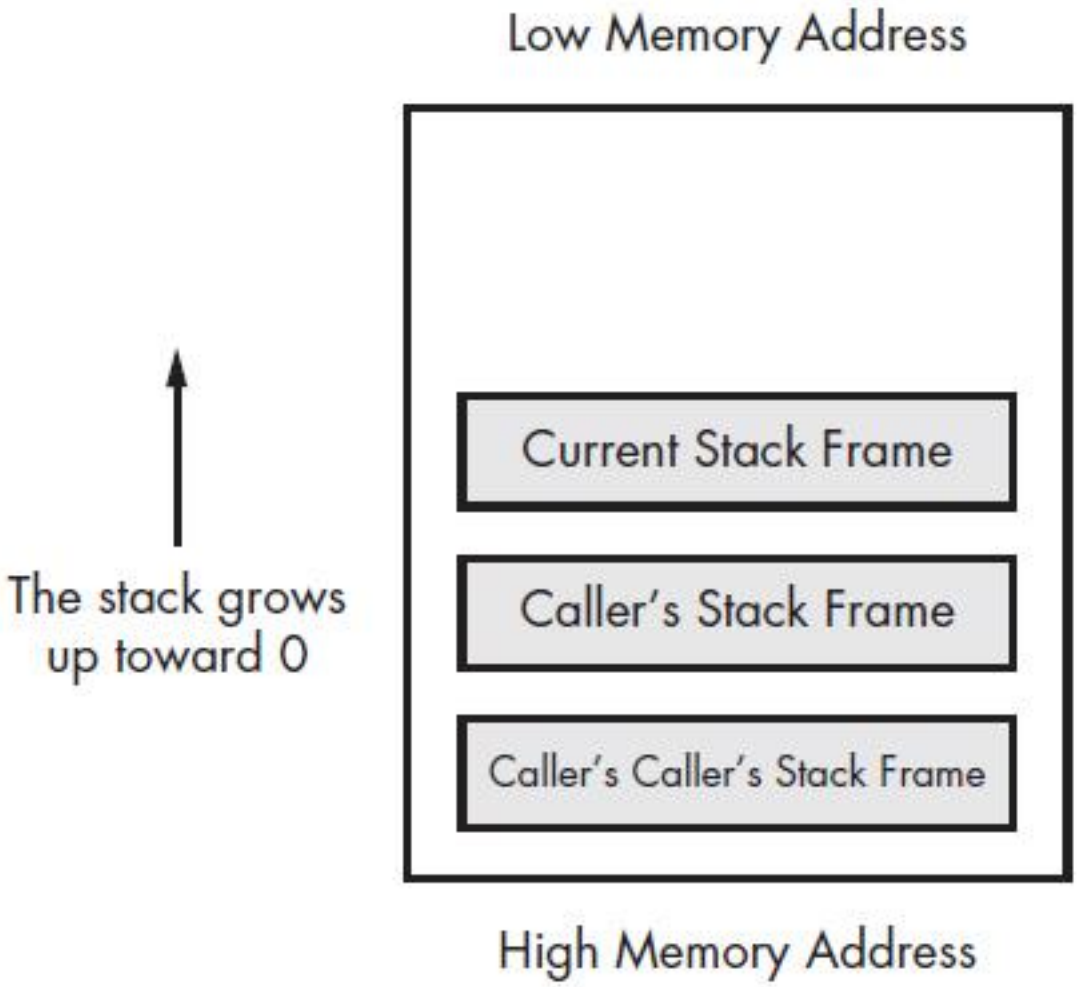
```
MOV ESP, EBP
POP EBP
RET
```

Sequência de chamada de função

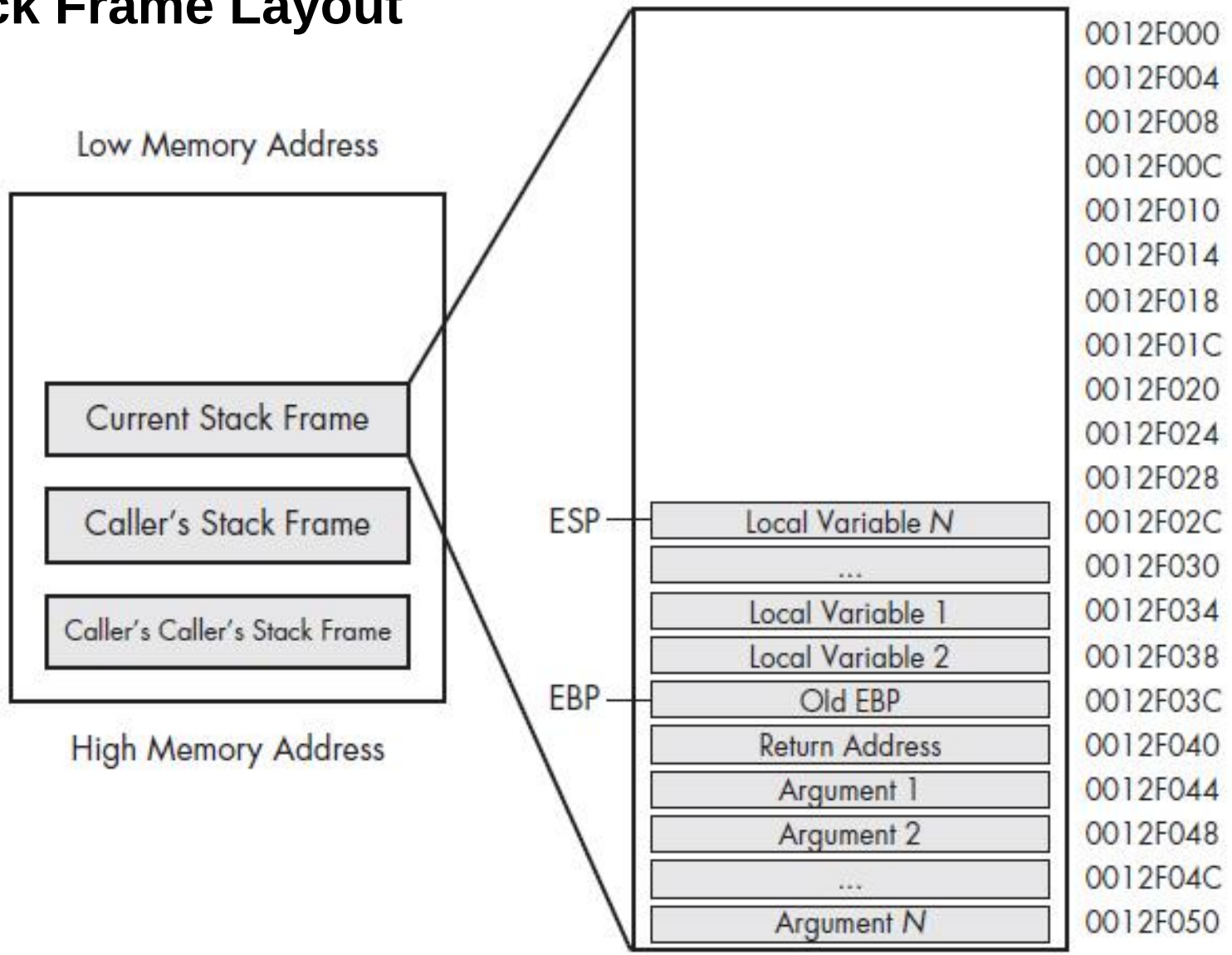
1. PUSH argumentos (parâmetros)
2. CALL function_address

   EIP é colocado na pilha e aponta p/ início da função
3. Prólogo da função
4. Função executa
5. Epílogo (= LEAVE)
6. RET (retira o endereço de retorno e coloca no EIP)
7. Programa continua a execução de onde parou

Layout da Pilha



Stack Frame Layout



Obrigado!

A explicação detalhada de todos os tópicos está na apostila.

Ronaldo Pinheiro de Lima
crimescibernetico.com@gmail.com
http://www.crimescibernetico.com
@crimescibernet