Linguagens de Montagem

Sub-Rotinas
Funções e Procedimentos x86
ANEXO - Aula 10

Edmar André Bellorini

Ano letivo 2021

Montar e Ligar aplicações x32 em máquinas x64

- Dependências:
  - gcc-multilib e libc6-i386

```
sudo apt-get install gcc-multilib libc6-i386
```

- Montar:

```
nasm -f elf32 nome.asm
```

- Ligar/linkar:

```
ld nome.o -m elf_i386 -o nome.x
```

Protocolo x86 para chamadas de sub-programas

- Convenção de chamadas
  - **Chamador** (*caller*)
    - **Antes** da instrução CALL

    1. Salvar *caller-saved registers* (se necessário)
       EAX, ECX e EDX
    2. Passagem de parâmetros para sub-programa via PILHA
    3. **Sempre** usar instrução CALL

    - **Após** instrução CALL

    4. Remover parâmetros da PILHA
    5. Recuperar os valores dos *caller-saved registers*

Protocolo x86 para chamadas de sub-programas

- Convenção de chamadas
  - **Chamado** (*callee*)
    - **Antes** de executar corpo de sub-programa

    1. Criar *stack-frame*
    2. Alocar espaço na PILHA para variáveis locais
    3. *calle-saved registers* (se necessário)
       EBX, EDI, ESI

    - **Depois** da execução do corpo do sub-programa

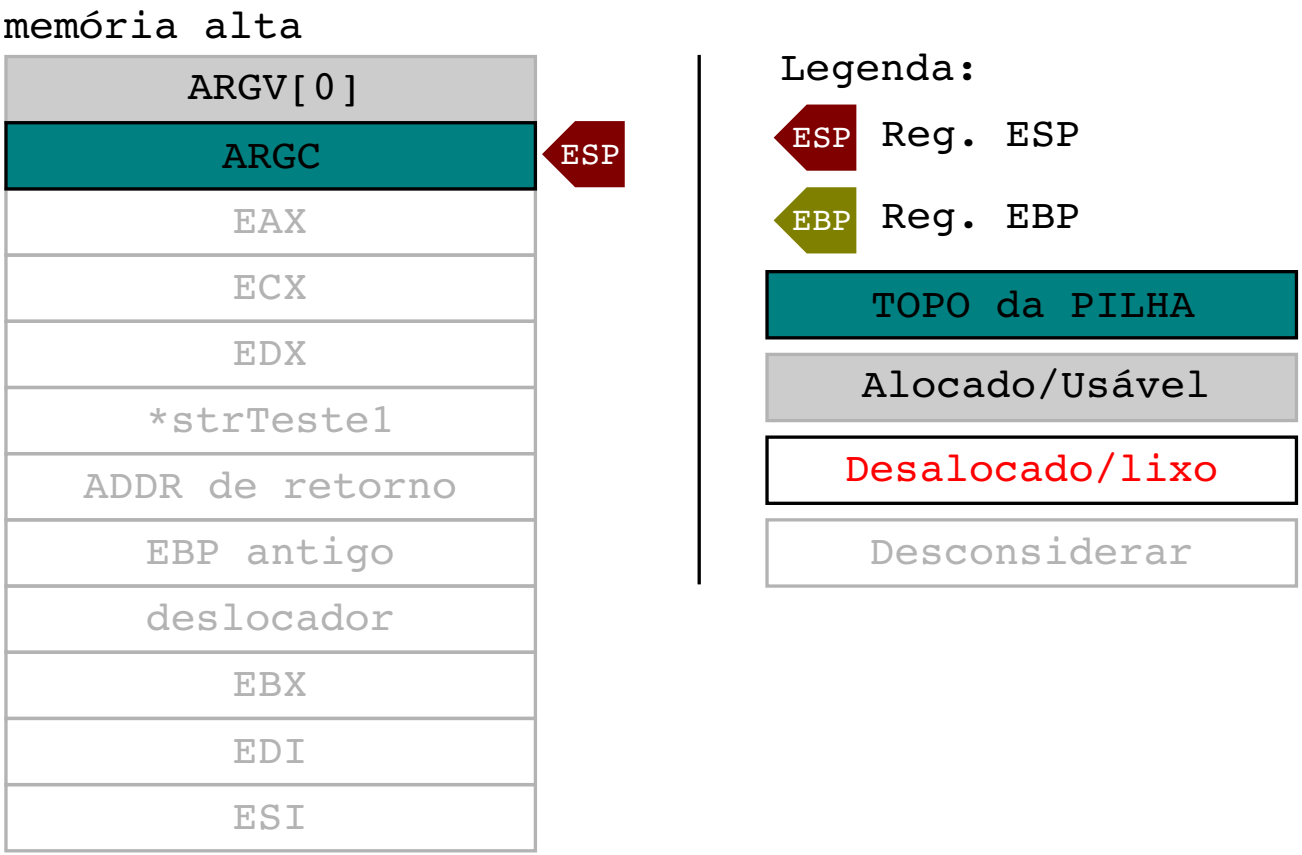
    4. Deixar resultado/retorno do sub-programa em EAX
    5. Recuperar os *calle-saved registers*
    6. Desalocar todas as variáveis locais
    7. Garantir endereço de retorno no topo da PILHA
    8. **Sempre** retornar com instrução RET

Huge example - a11e02.asm

- Será visto passo-a-passo

```
43      ; passo 2 - Antes da chamada CALL
44      ; empilhar parametros da direita para esquerda
45      push strTeste1
46
47      ; passo 3 - chamada CALL
48      call strLength
49
50      ; retorno do sub-programa em EAX
51      mov [str1L], eax
```

Antes do passo-a-passo - estado inicial da PILHA



Passo-a-passo com exemplo a11e02.asm - parte 01

- Convenção de chamadas
  - **Chamador** (caller)
    - **Antes** da instrução CALL

    1. Salvar *caller-saved registers* (se necessário) EAX, ECX e EDX

```
36      ...
37      ; passo 1 - Antes da chamada CALL
38      ; salvar registradores EAX, ECX e EDX
39      push eax
40      push ecx
41      push edx
42      ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA

Passo-a-passo com exemplo a11e02.asm - parte 02

- Convenção de chamadas
  - **Chamador** (caller)
    - **Antes** da instrução CALL
    2. Passagem de parâmetros para sub-programa via PILHA

```
42      ...
43      ; passo 2 - Antes da chamada CALL
44      ; empilhar parametros da direita para esquerda
45      push strTeste1
46      ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA

Passo-a-passo com exemplo a11e02.asm - parte 03

- Convenção de chamadas
  - **Chamador** (caller)
    - **Antes** da instrução CALL
    3. **Sempre** usar instrução CALL

```
46      ...
47      ; passo 3 - chamada CALL
48      call strLength
49      ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA

Passo-a-passo com exemplo a11e02.asm - parte 04

- Convenção de chamadas
  - **Chamado** (*callee*)
    - **Antes** de executar corpo de sub-programa

    1. Criar *stack-frame*

```
70      ...
71      ; passo 1 - Antes do corpo do sub-programa
72      ; criar stack-frame
73      push ebp
74      mov ebp, esp
75      ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA

Passo-a-passo com exemplo a11e02.asm - parte 05

- Convenção de chamadas
  - **Chamado** (*callee*)
    - **Antes** de executar corpo de sub-programa
    2. Alocar espaço na PILHA para variáveis locais

```
75      ...
76      ; passo 2 - Antes do corpo do sub-programa
77      ; criar espaco para variaveis locais
78      sub esp, 4  ; 4 bytes para variavel inteira local
79      ; [esp-4] eh o deslocador ate encontrar '0'
80      ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA

Passo-a-passo com exemplo a11e02.asm - parte 06

- Convenção de chamadas
  - **Chamado** (*callee*)
    - **Antes** de executar corpo de sub-programa
    2. *calle-saved registers* (se necessário)
       EBX, EDI, ESI

```
80      ...
81      ; passo 3 - Antes do corpo do sub-programa
82      ; salvar registradores EBX, EDI e ESI
83      push ebx
84      push edi
85      push esi
86      ...
```

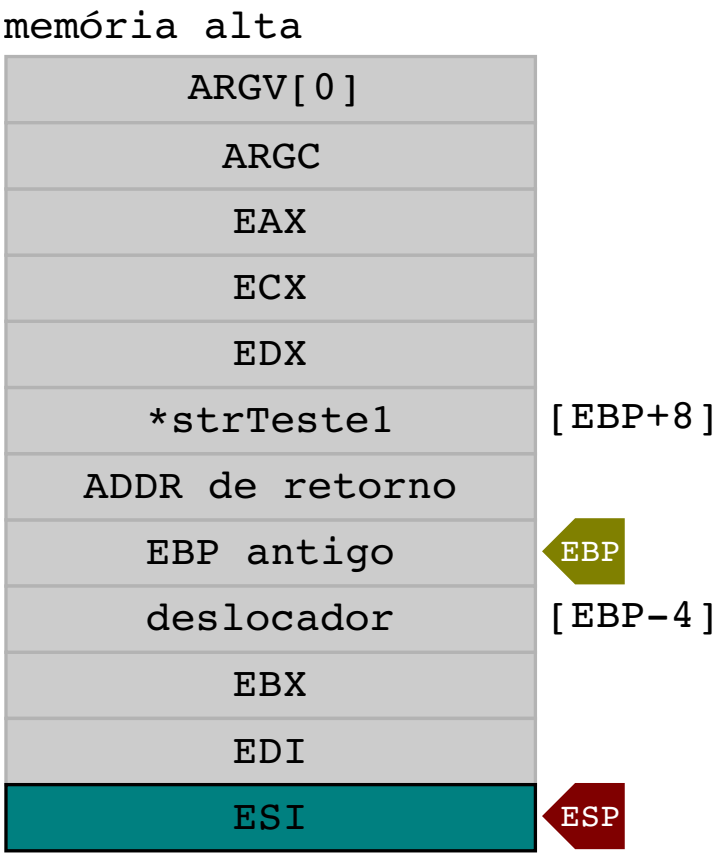
Passo-a-passo com exemplo a11e02.asm - PILHA

Passo-a-passo com exemplo a11e02.asm - parte 07

- Convenção de chamadas
  - **Chamado** (*callee*)
    - sub-programa
    - parâmetro *char[ ] → [ebp+8]
    - variável local → [ebp-4]

```
87      ...
88      mov ecx, [ebp+8] ; char *c[ ] - parametro 1
89      mov dword [ebp-4], 0
90      laco:
91         mov edx, [ebp-4]  ; deslocador
92         mov al, [ecx+edx] ; char[deslocador]
93         ; inc edx           ; para contar '\0'
94         cmp al, 0         ; char eh zero?
95         je finaliza
96         inc edx           ; desloc++
97         mov [ebp-4], edx  ; guarda deslocador
98         jmp laco
99      ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA



Passo-a-passo com exemplo a11e02.asm - parte 08

- Convenção de chamadas
  - **Chamado** (*callee*)
    - **Antes** de executar corpo de sub-programa
    4. Deixar resultado/retorno do sub-programa em EAX

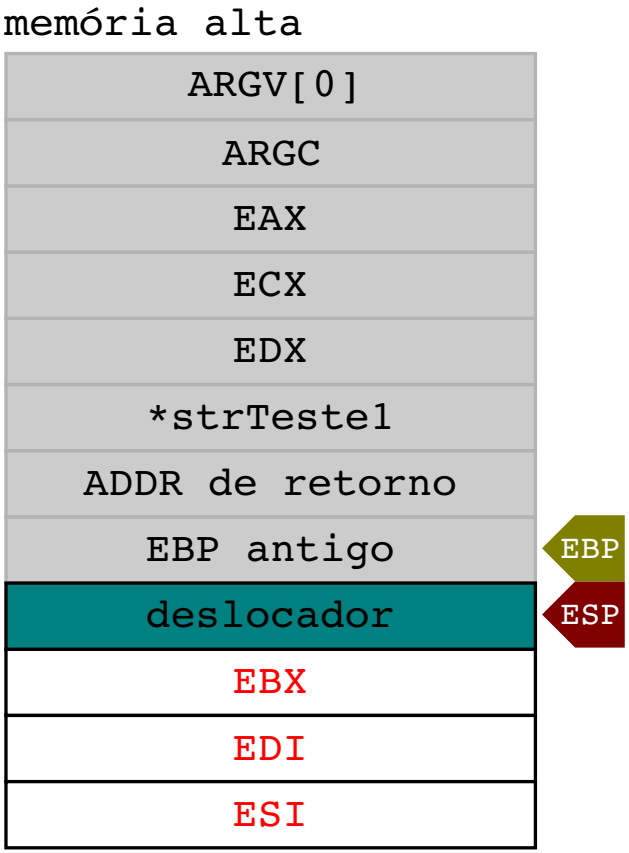
```
101     ...
102     ; passo 4 - depois do corpo do sub-programa
103     ; copiar resultado para EAX
104     mov eax, [ebp-4]
105     ...
```

Passo-a-passo com exemplo a11e02.asm - parte 09

- Convenção de chamadas
  - **Chamado** (*callee*)
    - Recuperar os *calle-saved registers*
    5. Deixar resultado/retorno do sub-programa em EAX

```
107     ...
108     ; passo 5 - depois do corpo do sub-programa
109     ; recuperar registradores EBX, EDI e ESI
110     pop esi ; ultimo empilhado
111     pop edi
112     pop ebx ; primeiro empilhado
113     ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA

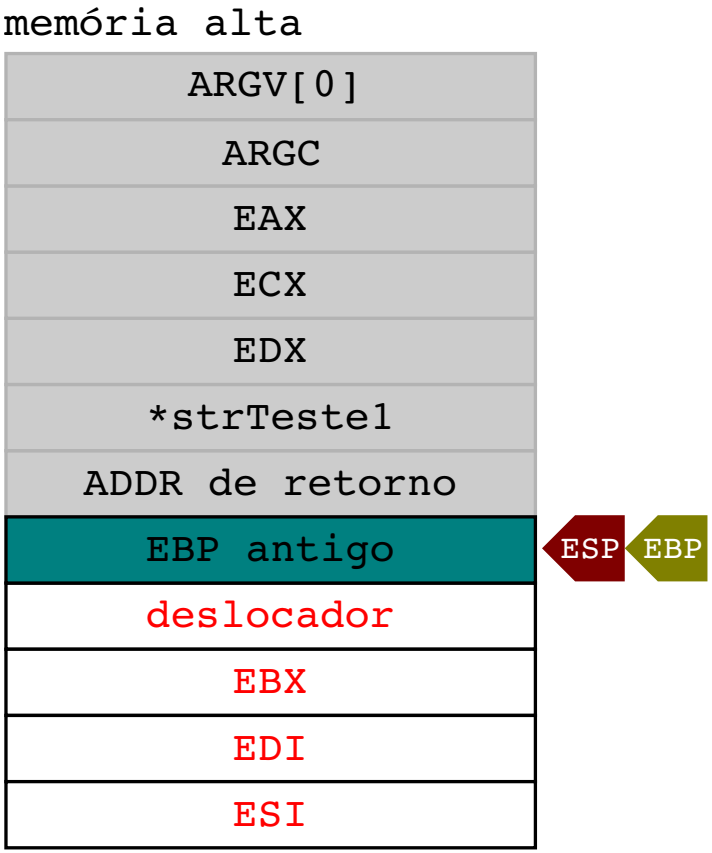


Passo-a-passo com exemplo a11e02.asm - parte 10

- Convenção de chamadas
  - **Chamado** (*callee*)
    - Recuperar os *calle-saved registers*
    6. Desalocar todas as variáveis locais

```
113     ...
114     ; passo 6 - depois do corpo do sub-programa
115     ; desalocar variaveis locais
116     mov esp, ebp
117     ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA

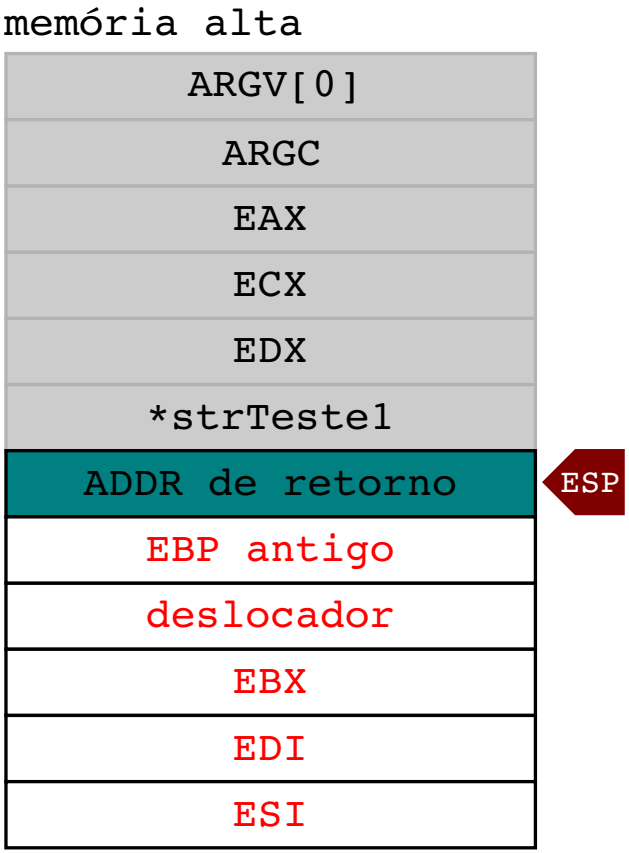


Passo-a-passo com exemplo a11e02.asm - parte 11

- Convenção de chamadas
  - **Chamado** (*callee*)
    - Recuperar os *calle-saved registers*
    7. Garantir endereço de retorno no topo da PILHA

```
117     ...
118     ; passo 7 - depois do corpo do sub-programa
119     ; recuperar ebp antigo, garantir endereco de retorno
120     pop ebp
121     ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA



Passo-a-passo com exemplo a11e02.asm - parte 12

- Convenção de chamadas
  - **Chamado** (*callee*)
    - Recuperar os *calle-saved registers*
    - 8 **Sempre** retornar com instrução RET

```
121     ...
122     ; passo 8 - depois do corpo do sub-programa
123     ; sempre, sempre, sempre retorne com RET
124     ret
125     ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA

Passo-a-passo com exemplo a11e02.asm - parte 13

- Convenção de chamadas
  - **Chamador** (caller)
    - **Antes** da instrução CALL
    4. Remover parâmetros da PILHA

```
52      ...
53      ; passo 4
54      ; remover parametros da PILHA
55      add esp, 4 ; +4 para cada parametro empilhado
56      ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA

Passo-a-passo com exemplo a11e02.asm - parte 14

- Convenção de chamadas
  - **Chamador** (caller)
    - **Antes** da instrução CALL

    5. recuperar os valores dos *caller-saved registers*

```
56      ...
57      ; passo 5
58      ; recuperar registradores EAX, ECX e EDX
59      pop edx  ; ultimo empilhado
60      pop ecx
61      pop eax  ; primeiro empilhado
62      ...
```

Passo-a-passo com exemplo a11e02.asm - PILHA

Finalizando o passo-a-passo com exemplo a11e02.asm

- Material de apoio para seção *x*86

▸ x86 Assembly Guide

Seção *Calling Convention*

  - Acessado em 05/07/2021
  - Obs.: O guia utiliza modelo de memória FLAT do 486
    Não abordado na disciplina
    Algumas coisas podem parecer estranhas

Chamada de funções externas (C)

- Utilização de funções externas
  - Declaração

```
    extern nomeDaFuncao
```

  - Chamada utilizando CALL

```
     CALL nomeDaFuncao
```

  - Passagem de parâmetro utilizando protocolo da arquitetura

Chamada de funções externas (C)

- Alteração no código .asm

```
    section .text
        global _start
```

é alterado para:

```
    section .text
        global main
```

  - Agora temos a função main
    - que requer a criação do stack-frame

Exemplo a11e04.asm - printf em x86

Seção mantida apenas por curiosidade!

- Chamada para *printf* em arquitetura x86

```
24      ...
25      push umaStr ; endereco para string 'umaStr'
26      push dword [umInt] ; conteudo do inteiro 'umInt'
27      push strCtrl ; string de controle para printf
28      call printf
29      ...
```

- Montar:

```
nasm -f elf32 a11e04.asm
```

- Linkar:

```
gcc -m32 a11e04.o -o a110e04.x
```