Linguagens de Montagem

Modos de Endereçamento - por Pilha
Passagem de Parâmetros via S.O.

Aula 09

Edmar André Bellorini

Ano letivo 2021

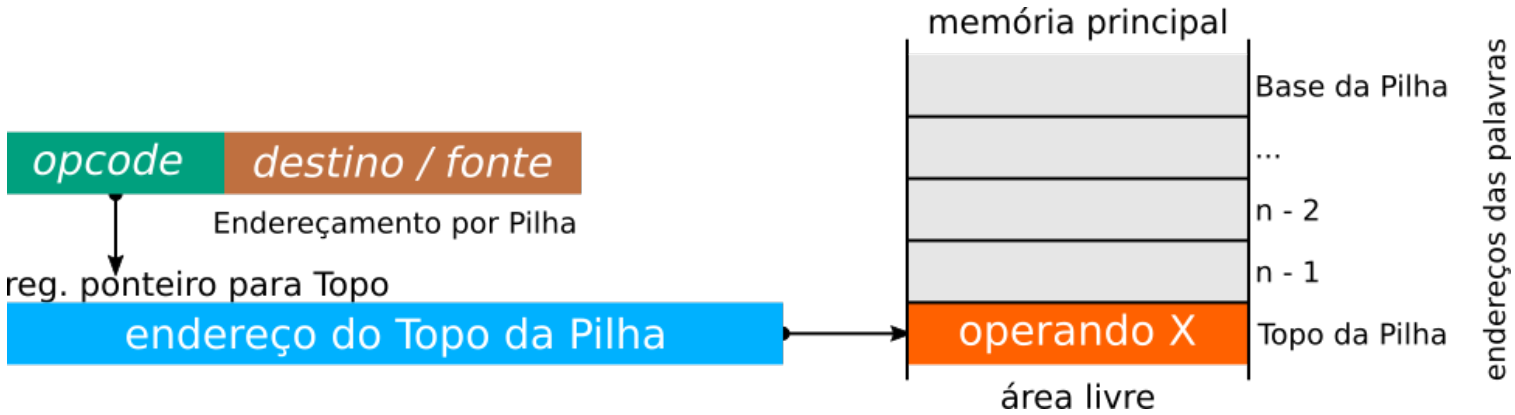
Modos de Endereçamento

Modos de Endereçamento:

- Imediato
- Direto
- Indireto
- por Registrador
- Indireto por Registrador
- por Deslocamento
- por Pilha (~~aula 09~~ nesta aula)

Endereçamento por Pilha

- Operando está na posição de memória indicado no registrador de topo da pilha



- 2 instruções:

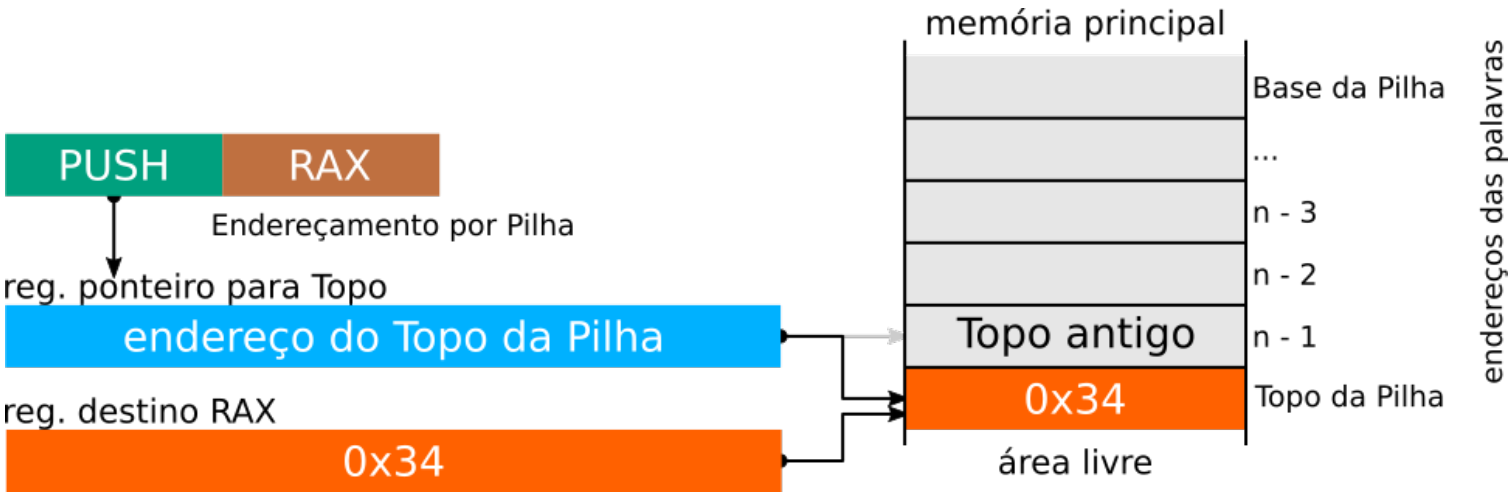
```
PUSH operando64bits
POP  operando64bits
```

  - Contém 1 referência à memória
  - Operando *fonte* é **implícito**

Endereçamento por Pilha - Exemplo

- Armazenar valor na pilha
  1. Identifica tamanho |p| do operando
  2. Decrementa |p| em reg. ponteiro
  3. Aloca operando à partir do reg. ponteiro

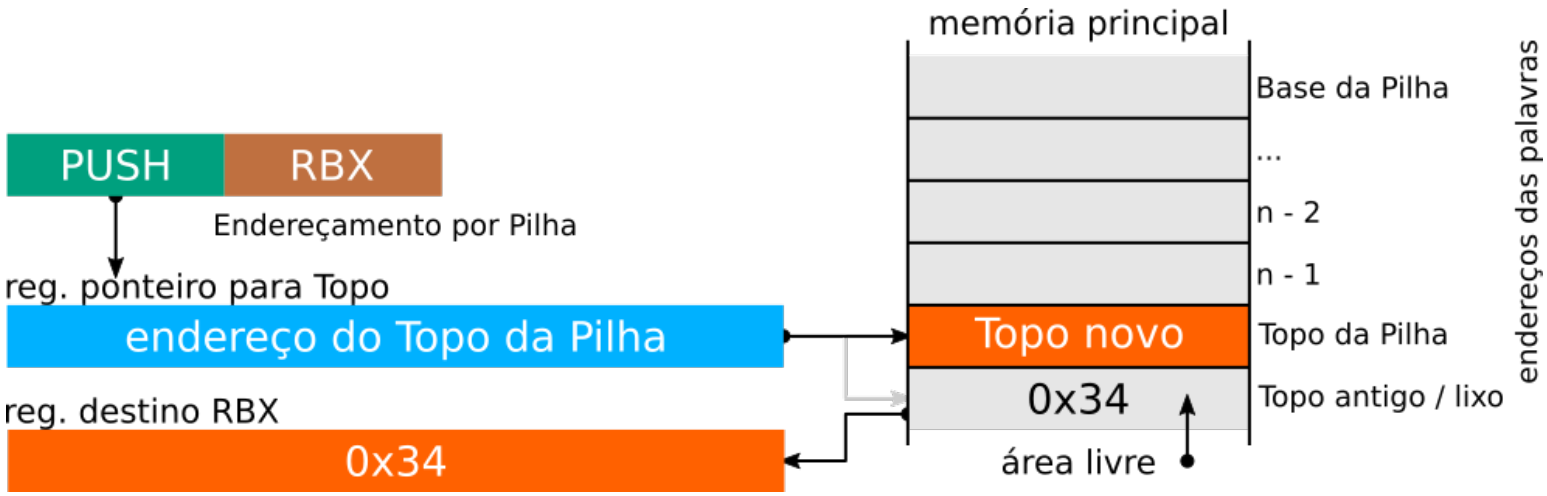
```
PUSH rax
```



Endereçamento por Pilha - Exemplo

- Ler valor da pilha
  1. Identifica tamanho |p| do operando
  2. Copia palavra de tamanho |p| para *destino*
  3. Incrementa |p| em reg. ponteiro

```
POP rbx
```



Endeçamento por Pilha - Exemplos a09e01a.asm e a09e01b.asm

```
40      ...
41      ; write
42      mov rax, 1
43      mov rdi, 1
44      mov rsi, strLida
45  l3:
46      ; retorna no. de chars lidos
47      pop rdx
48  l4:
49      syscall
50      ...
```

Pilha

- Estrutura FILO / LIFO
  - *First-In, Last-Out* / *Last-In, First-Out*
- Utilizada para:
  - Comunicação do SO com o programa
    - Passagem de parâmetros via linha de comando
    - Aula atual
  - Armazenamento temporário de variáveis locais
    - Internas à subprogramas
    - Aula 10
  - apenas x86: Passagem de parâmetros
    - Parâmetros para subprogramas
    - Endereço de retorno
    - Não abordado mais na disciplina

Registrador Ponteiro para Topo: RSP

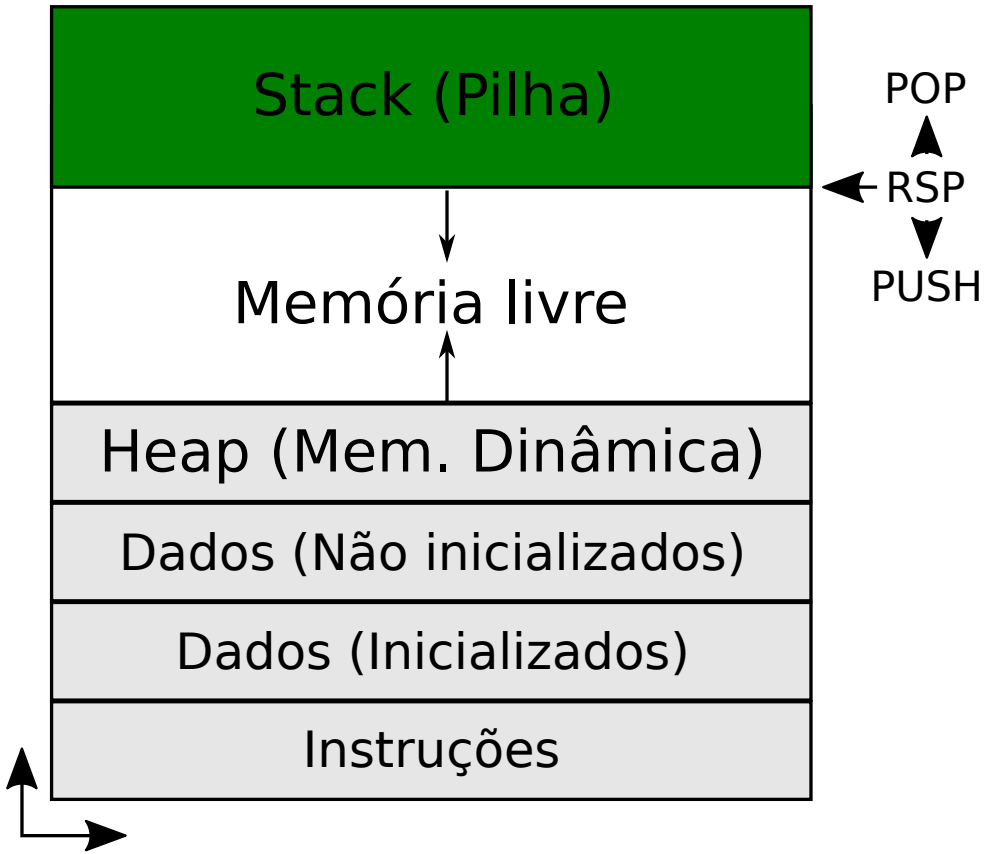
- Registrador de propósito geral de segmento
  - RSP (64bits)
  - ESP (32bits)
- Contém endereço para TOPO da Pilha
  - Subtrai |p| de RSP

```
PUSH operando
```

  - Adiciona |p| ao RSP

```
POP operando
```

Registrador RSP



Exemplo a09e01a.asm (dnovo!)

```
Breakpoint 1, 0x0000000000401036 in l1 ()
(gdb) p /x $rsp
$1 = 0x7fffffffdf80
(gdb) c
Continuing.

Breakpoint 2, 0x0000000000401037 in l2 ()
(gdb) p /x $rsp
$2 = 0x7fffffffdf78
(gdb) c
Continuing.
Vc digitou           :
Breakpoint 3, 0x0000000000401066 in l3 ()
(gdb) p /x $rsp
$3 = 0x7fffffffdf78
(gdb) c
Continuing.

Breakpoint 4, 0x0000000000401067 in l4 ()
(gdb) p /x $rsp
$4 = 0x7fffffffdf80
(gdb)
```

Exemplos a09e02.asm e a09e02.asm

a09e02.asm para 32 bits

```
15      ...
16    ; push byte  [v1] ; np
17      push word  [v2]
18      push dword [v3]
19    ; push qword [v4] ; np
20      ...
```

- Não permitido:
  - push byte (8 bits)
  - push qword (64 bits)
- Se *push* não permite
  - *pop* também não

a09e02.asm para 64 bits

```
15      ...
16    ; push byte  [v1] ; np
17      push word  [v2]
18    ; push dword [v3] ; np
19      push qword [v4]
20      ...
```

- Não permitido:
  - push byte (8 bits)
  - push dword (32 bits)
- Se *push* não permite
  - *pop* também não

Ver último slide (anexo)

Passagem de parâmetros via S.O.

- Argumentos passados por linha de comando

```
./nomedoPrograma.x arg1 arg2 arg3 ...
```

- Código C:

```
...
int main(int argc, char *argv[]){
...
```

  - argc → no. de argumentos
  - argv → vetor de ponteiros para argumentos string

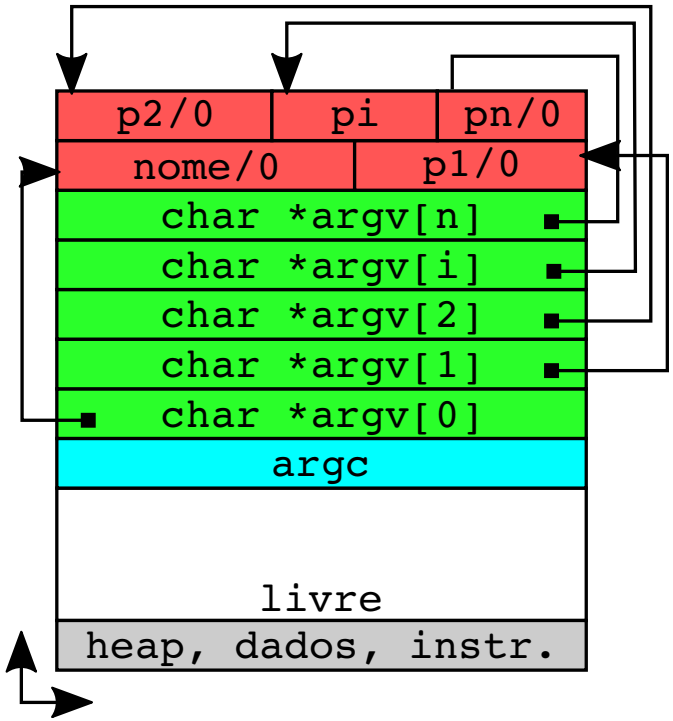
Exemplo a09e03c.c

- Código em C

```
6     ...
7     printf("\nNome do programa: %s\n", argv[0]);
8     argc --;
9
10    while(argc > 0){
11       printf("param           : %s\n", argv[k]);
12       k++;
13       argc--;
14    }
15    ...
```

Passagem de parâmetros via S.O.

- Utilização da Pilha



Passagem de parâmetros via S.O. em LM

- Utilização da Pilha
  - Topo da Pilha contém número de argumentos
    - Sempre é $>= 1$
  - O primeiro argumento é ponteiro para nome do programa
  - Demais parâmetros
    - Sempre no formato *NULL-terminated-string*

```
section .data
    str1 : db 'a', 'b', 'c', 0
    str2 : db 'abc', 0
```

Exemplo a09e03.asm

- Lista argumentos passados como parâmetros por S.O.
  - Mesma funcionalidade do exemplo a09e03c.c

```
34   ...
35 printSaida:
36       pop r15                ; parametro *string
37       xor r9d,r9d
38 laco:
39       mov r8b, [r15+r9]      ; caractere a ser impresso
40       cmp r8b, 0             ; null-char
41       je testaParam          ; terminou a string?
42   ...
```

Atividade

- a09at01: Comando **mv** simplificado.
  - O comando **mv** move arquivos de um diretório para outro, porém também é usado para renomear um arquivo.
    Por exemplo:
    ```
    $: mv nomeVelho.txt nomeNovo.txt
    ```
    renomeia o arquivo nomeVelho.txt para nomeNovo.txt
  - Deve ser criado o comando **lmrename** utilizando a chamada de sistema **sys_rename()** (syscall: 0x52)
    - comando **rename** existe, evite confusões
    - comando recebe 2 argumentos: **nomeVelho** e **nomeNovo**
    - **copie** as strings da pilha para variáveis **não inicializadas**
      Dica: S.O. já atribuí `null-char` como terminador de Strings
    - execução:
      ```
      $: .\a09at01.x nomeVelho.txt nomeNovo.txt
      ```

Fim do Documento

Dúvidas?

Aula 10:

- Subprogramas!
  - Também conhecidos como Procedimentos e Funções