

Linguagem de Montagem

Modos de Endereçamento Endereçamento por Pilha Aula 09

Edmar André Bellorini

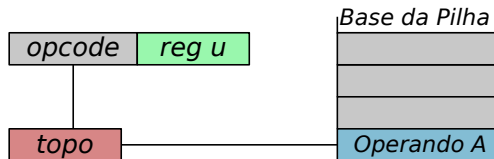
Ano letivo 2016

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
```

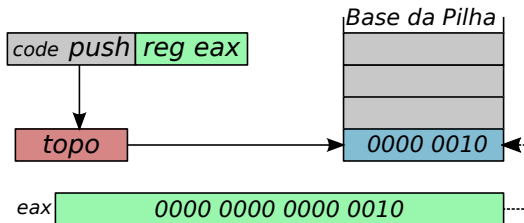
```
POP operando64bits
```

- ▶ Contém 1 referência à memória
- ▶ Implícito

Endereçamento por Pilha - Exemplo

- ▶ Armazenar valor na pilha

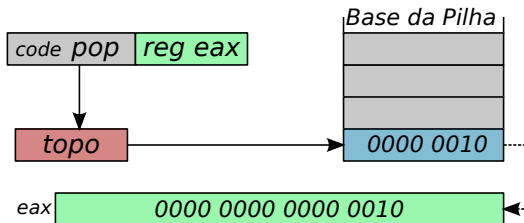
1 **PUSH** *eax*



Endereçamento por Pilha - Exemplo

- Ler valor da pilha

1 `POP eax`



Endeçamento por Pilha - Exemplo A09e01.asm

```
40     ...  
41     ; write  
42     mov eax, 4  
43     mov ebx, 1  
44     mov ecx, strLida  
45 13:  
46     ; retorna no. de chars lidos  
47     pop rdx  
48 14:  
49     int 0x80  
50     ...
```

Pilha

- ▶ Estrutura FILO
 - ▶ *First-In, Last-Out*
- ▶ Utilizada para:
 - ▶ Armazenamento temporário de variáveis locais
 - ▶ Internas à subprogramas
 - ▶ Comunicação do SO com o programa
 - ▶ Passagem de parâmetros via linha de comando
 - ▶ Passagem de parâmetros (x86)
 - ▶ Parâmetros para subprogramas
 - ▶ Endereço de retorno

Registrador RSP

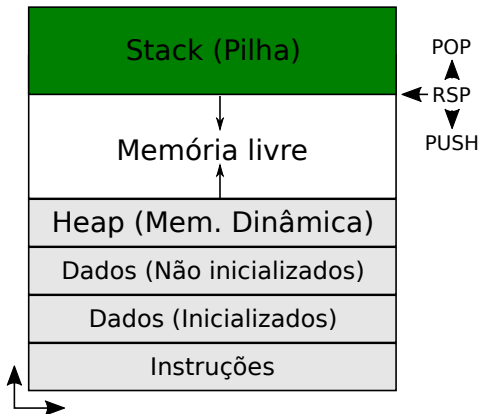
- ▶ Registrador de propósito geral de segmento
 - ▶ RSP (64bits)
 - ▶ ESP (32bits)
- ▶ Contém endereço para TOPO da Pilha
 - ▶ Subtraí operando-Bytes de RSP

PUSH operando

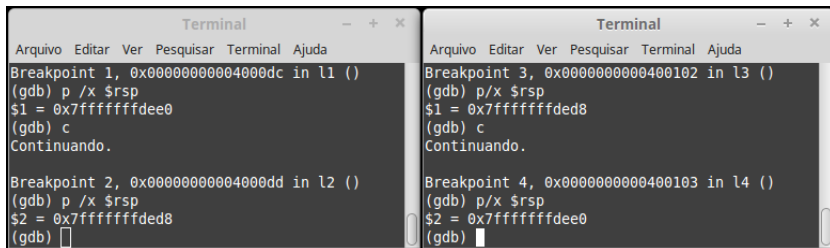
- ▶ Adiciona operando-Bytes de RSP

POP operando

Registrador RSP



Exemplo a09e01.asm (dnovo!)



```
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
Breakpoint 1, 0x0000000004000dc in l1 ()
(gdb) p /x $rsp
$1 = 0x7fffffffdee0
(gdb) c
Continuando.

Breakpoint 2, 0x0000000004000dd in l2 ()
(gdb) p /x $rsp
$2 = 0x7fffffffded8
(gdb) █

Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
Breakpoint 3, 0x000000000400102 in l3 ()
(gdb) p/x $rsp
$1 = 0x7fffffffdded8
(gdb) c
Continuando.

Breakpoint 4, 0x000000000400103 in l4 ()
(gdb) p/x $rsp
$2 = 0x7fffffffdee0
(gdb) █
```

Exemplos a09e02_x32.asm e a09e02_x64.asm

a09e02_x32.asm

```
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

Ver último slide (anexo)

a09e02_x64.asm

```
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

Passagem de parâmetros via S.O.

- ▶ Argumentos passados por linha de comando
./nomedoPrograma.x arg1 arg2 arg3 ...

- ▶ Código C:

```
2  ...  
3  int main(int argc, char *argv[]){  
4  ...
```

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

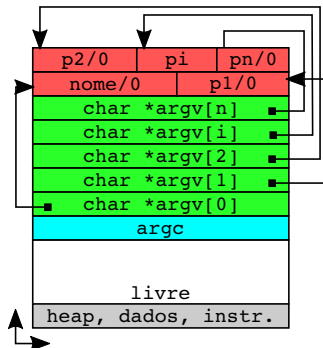
Exemplo a09e03c.c

► 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

```
27  ...
28  printSaida:
29      pop r15    ; parametro *string
30
31  laco:
32      mov r9d, [desl]
33      mov r8b, [r15+r9]
34      cmp r8b, 0
35      je testaParam
36  ...
```


Exercício de Fixação

► EF0901: Comando **mv** simplificado.

- O comando **mv** move arquivos de um diretório para outro, porém também é usado para renomear um arquivo.

Por exemplo:

```
$: mv nomeOld.txt nomeNew.txt
```

renomeia o arquivo nomeOld.txt para nomeNew.txt

- Deve ser criado o comando **lmrename** utilizando a chamada de sistema `rename()`
 - o comando **rename** existe, evite confusões
 - o comando deve receber 2 argumentos: **nomeOld** e **nomeNew**
 - copie as strings da pilha para variáveis não inicializadas
 - execução:

```
$: lmrename oldname.txt newname.txt
```

Relatório

- ▶ Somente relatório
 - ▶ O modelo de relatório para a disciplina de LM está disponível em anexo da Aula 01
 - ▶ Arquivo **modeloRelatorioLM.odt**
 - ▶ A data do relatório é a data de entrega (ver moodle)
 - ▶ Somente serão aceitos os relatórios em formato .pdf com nome do arquivo seguindo o padrão:
TY.PXX.nome.sobrenome.pdf
 - ▶ TY é o número da turma prática (1, 2, 3 ou 4)
 - ▶ PXX é o número da prática, neste caso: P09
 - ▶ Ex.: aluno Warren Robinett da turma prática 9 (de 1979):
 - ▶ T9.P09.Warren.Robinett.pdf

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
```