

Linguagem de Montagem

Registradores e Instrução MOV Aula 03

Edmar André Bellorini

Ano letivo 2016

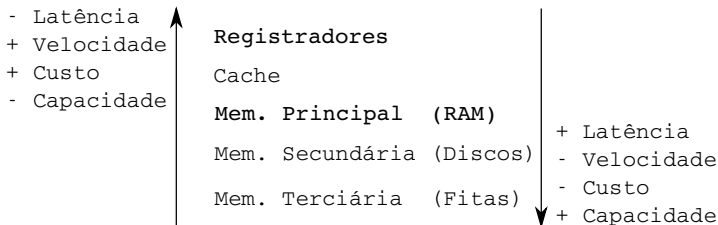
Dados Inicializados (.data) - Exemplos da Aula 02

```
v1: db      0x55                ; byte 0x55
v2: db      0x55,0x56,0x57      ; 3 bytes em sucessao
v3: db      'a',0x55            ; caracteres com aspas
v4: db      'hello',13,10,'$'  ; strings tambem
v5: dw      0x1234              ; 0x34 0x12
v6: dw      'a'                 ; 0x61 0x00
v7: dw      'ab'                ; 0x61 0x62
v8: dw      'abc'               ; 0x61 0x62 0x63 0x00
v9: dd      0x12345678          ; 0x78 0x56 0x34 0x12
```

Dados em memória

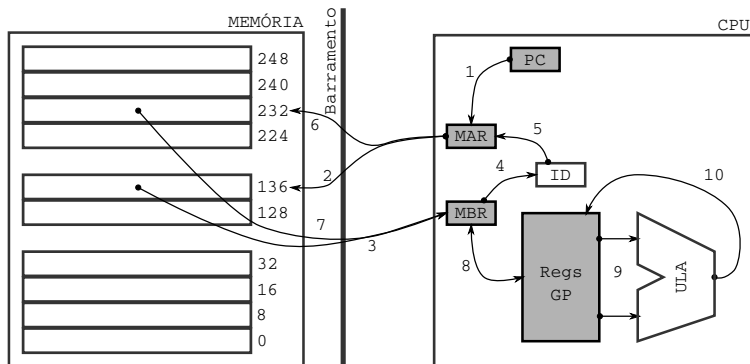
► Hierarquia das Memórias

- Agrupar memórias de grande capacidade, baixo custo, porém lentas com memórias rápidas, porém de alto custo e baixa capacidade
- Melhor estudado na disciplina de Organização e Arquitetura de Computadores (3ª Série)



Registradores

- ▶ É o tipo de memória mais rápida encontrada nos sistemas computacionais
- ▶ Faz parte do núcleo dos processadores



Classificação dos Registradores

- ▶ Visíveis ao usuário (programador)
 - ▶ São usados pelos programadores (ou montadores) para reduzir o acesso à memória e otimizar códigos
 - ▶ Ex.: Registradores de Propósito Geral (Reg GP)
- ▶ Não Visíveis
 - ▶ Controla o fluxo de operações internas e podem ser acessados somente por alguns programas privilegiados do S.O.
 - ▶ Ex.: PC (*Program-Counter*), MAR (*Memory Address Register*) e MBR (*Memory Buffer Register*)

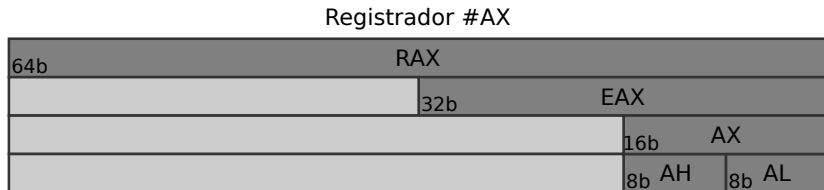
Registradores Visíveis

- ▶ de Propósito Geral
 - ▶ Armazenam dados e endereços usados nas operações lógicas e aritméticas
 - ▶ Usados diretamente pelo usuário
- ▶ de Controle
 - ▶ Flags (condicionais) e ponteiros para segmentos/pilha
 - ▶ Normalmente não são usados diretamente pelo usuário

Registradores de Propósito Geral

- ▶ Máquinas de 32 bits (x86)
 - ▶ Gerais: EAX, EBX, ECX, EDX
 - ▶ Segmentos: ESI, EDI, EBP, **ESP**
- ▶ Máquinas 64 bits (x86_x64)
 - ▶ Gerais: RAX, RBX, RCX, RDX, R8 até R15
 - ▶ Segmentos: RSI, RDI, RBP, **RSP**

Exemplo de Registrador: RAX



- ▶ O acesso ao registrador pode ser realizado de diversas formas
 - ▶ RAX: acessa os 64 bits
 - ▶ EAX: 32 bits menos significativos de RAX (0 até 31)
 - ▶ AX: 16 bits menos significativos de RAX (0 até 15)
 - ▶ AL: 8 bits menos significativos de RAX (0 até 7)
 - ▶ AH: 8 bits mais significativos de AX (8 até 15)
- ▶ Essa forma de acesso também vale para RBX, RCX e RDX

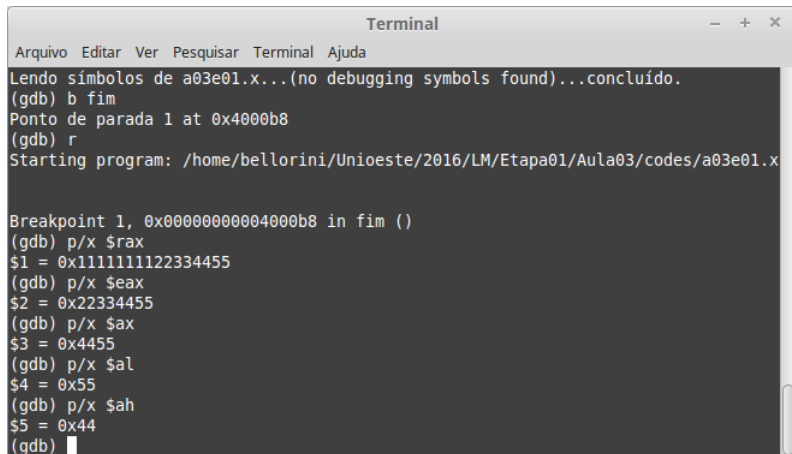
Exemplo de acesso ao registrador #AX

► Programa de 64 bits

```
1  section .data
2      num: dq 0x1111111122334455
3
4  section .text
5      global _start
6
7  _start:
8      mov rax, [num]
9
10 fim:
11     mov rax, 1
12     mov rbx, 0
13     int 0x80
```

Debugger

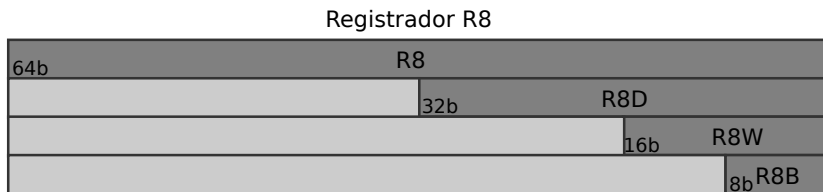
- ▶ Referenciar conteúdo de um registrador: “\$nomeReg”



```
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
Lendo símbolos de a03e01.x...(no debugging symbols found)...concluído.
(gdb) b fim
Ponto de parada 1 at 0x4000b8
(gdb) r
Starting program: /home/bellorini/Unioeste/2016/LM/Etapa01/Aula03/codes/a03e01.x

Breakpoint 1, 0x00000000004000b8 in fim ()
(gdb) p/x $rax
$1 = 0x11111111122334455
(gdb) p/x $eax
$2 = 0x22334455
(gdb) p/x $ax
$3 = 0x4455
(gdb) p/x $al
$4 = 0x55
(gdb) p/x $ah
$5 = 0x44
(gdb) █
```

Registradores de 64 bits R8 até R15



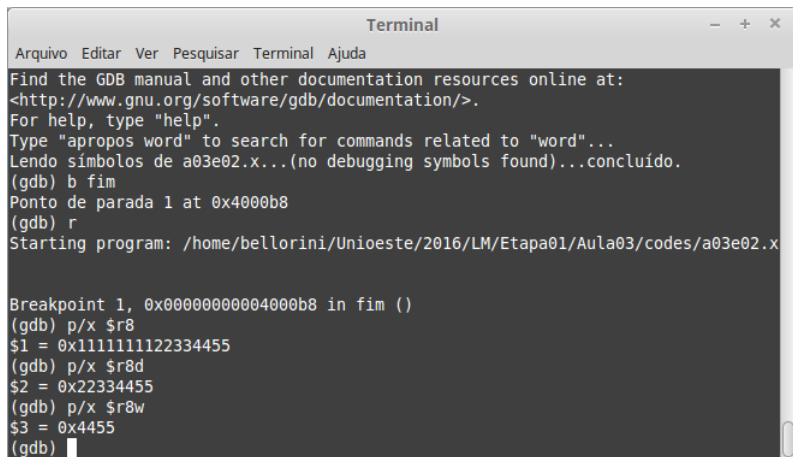
- ▶ O acesso ao registrador pode ser realizado de diversas formas
 - ▶ R8: acessa 64 bits
 - ▶ R8D: 32 bits menos significativos de R8 (0 até 31)
 - ▶ R8W: 16 bits menos significativos de R8 (0 até 15)
 - ▶ R8B: 8 bits menos significativos de R8 (0 até 7)
- ▶ Esta forma de acesso também vale para R9 até R15

Exemplo para registradores de 64bits R8 até R15

► Programa de 64 bits

```
1  section .data
2  num: dq 0x1111111122334455
3
4  section .text
5  global _start
6
7  _start:
8  mov R8, [num]
9
10 fim:
11 mov rax, 1
12 mov rbx, 0
13 int 0x80
```

Debugger



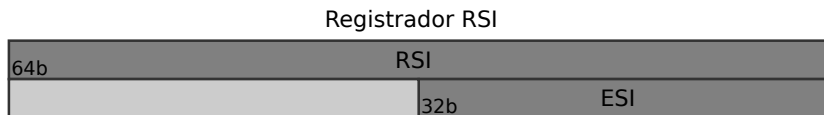
```
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Lendo símbolos de a03e02.x...(no debugging symbols found)...concluído.
(gdb) b fim
Ponto de parada 1 at 0x4000b8
(gdb) r
Starting program: /home/bellorini/Unioeste/2016/LM/Etapa01/Aula03/codes/a03e02.x

Breakpoint 1, 0x00000000004000b8 in fim ()
(gdb) p/x $r8
$1 = 0x11111111122334455
(gdb) p/x $r8d
$2 = 0x22334455
(gdb) p/x $r8w
$3 = 0x4455
(gdb) |
```

Registradores de segmentos

- ▶ RSI e RDI
 - ▶ Registradores de fonte e destino para algumas instruções
 - ▶ A princípio, podem ser usados como propósito geral para instruções básicas
- ▶ RBP e RSP
 - ▶ São registradores especiais (ponteiros) para estrutura do programa
 - ▶ RBP pode ser usado para propósito geral, porém é altamente não recomendado
 - ▶ RSP pode ser usado para propósito geral, porém normalmente é fatal para a aplicação

Exemplo para registradores de segmento: RSI



- ▶ O acesso ao registrador pode ser realizado de duas forms
 - ▶ RSI: acessa 64 bits
 - ▶ ESI: 32 bits menos significativos (0 até 31)
- ▶ Vale também para RDI, RBP e RSP
- ▶ A função destes registradores será melhor estudada em aulas futuras

Instrução de movimentação de dados: MOV

- ▶ MOV: movimento (cópia) de dados da fonte para destino
`MOV destino, fonte`
- ▶ Para os exemplos, considere:

```
section .data
```

```
    v1: dq 0x1111111122334455
```

```
    v2: dq 0x0000000000000000
```

```
    v3: dq 0x0000000000000000
```


Sintaxe MOV - pt1

► Sintaxe

- Cópia de dados da memória para registrador

```
MOV reg8 , r/m8
```

```
MOV reg16, r/m16
```

```
MOV reg32, r/m32
```

```
MOV reg64, r/m64
```

- Exemplo

```
MOV al , [v1] ; 8 bits de conteudo de v1 para al
```

```
MOV ebx, [v1] ; 32 bits de conteudo de v1 para EBX
```

```
MOV rcx, [v1] ; 64 bits de conteudo de v1 para RCX
```

Sintaxe MOV - pt2

► Sintaxe

- Cópia de dados de registrador para memória

`MOV r/m8 , reg8`

`MOV r/m16, reg16`

`MOV r/m32, reg32`

`MOV r/m64, reg64`

- Exemplo

`MOV [v2], al ; 8 bits de al para conteúdo de v2`

`MOV [v2], ebx ; 32 bits de EBX para conteúdo de v2`

`MOV [v2], rcx ; 64 bits de RCX para conteúdo de v2`

Sintaxe MOV - pt3

► Sintaxe

- Cópia de dados de imediato para registrador

```
MOV reg8 , imm8
```

```
MOV reg16, imm16
```

```
MOV reg32, imm32
```

```
MOV reg64, imm64
```

- Exemplo

```
MOV al , 0x10 ; 8 bits para al
```

```
MOV ebx, 0x20202020 ; 32 bits para EBX
```

```
MOV rcx, 0x3030303030303030 ; 64 bits para RCX
```

Sintaxe MOV - pt4

► Sintaxe

- Cópia de dados de imediato para memória

```
MOV r/m8 , imm8
```

```
MOV r/m16, imm16
```

```
MOV r/m32, imm32
```

- Exemplo

```
MOV byte [v3], 0x10
```

; 8 bits para conteudo de v3

```
MOV word [v3], 0x1515
```

; 16 bits para conteudo de v3

```
MOV dword [v3], 0x20202020
```

; 32 bits para conteudo de v3

- Identificador de tamanho de palavra é requerido
- NASM não suporta movimentação de imediato de 64 bits para memória

Exercício de Fixação

- ▶ Escreva um código funcional (montável e linkável) que realize todas as 15 formas de movimentação de dados.
 - ▶ Considere a seguinte seção para seu código

```
1  section .data
2      pt1r8   : db 0x10                ; parte 1
3      pt1r16  : dw 0x2020
4      pt1r32  : dd 0x30303030
5      pt1r64  : dq 0x4040404040404040
6
7      pt2m8   : db 0x00                ; parte 2
8      pt2m16  : dw 0x0000
9      pt2m32  : dd 0x00000000
10     pt2m64  : dq 0x0000000000000000
11
12     ; parte 3 nao contem variaveis
13
14     pt4m8   : db 0x00                ; parte 4
15     pt4m16  : dw 0x0000
16     pt4m32  : dd 0x00000000
```

Exercício de Fixação

- ▶ Se, e somente se, Máquina de 32 bits
 - ▶ Desconsiderar instruções de 64 bits
 - ▶ Total de 12 instruções para serem testadas
- ▶ É necessário debuggar para confirmar os valores em memória e registradores
- ▶ Use breakPoints
- ▶ Deve ser entregue apenas o relatório.

Relatório

- ▶ Somente relatório
 - ▶ O modelo de relatório para a disciplina de LM está disponível em anexo da Aula 01
Arquivo **modeloRelatorioLM.odt**
 - ▶ Somente serão aceitos os relatórios em formato .pdf com nome do arquivo seguindo o padrão **PXX.nome.sobrenome.pdf**
Onde PXX é o número da prática, neste caso: P03