

Linguagem de Montagem

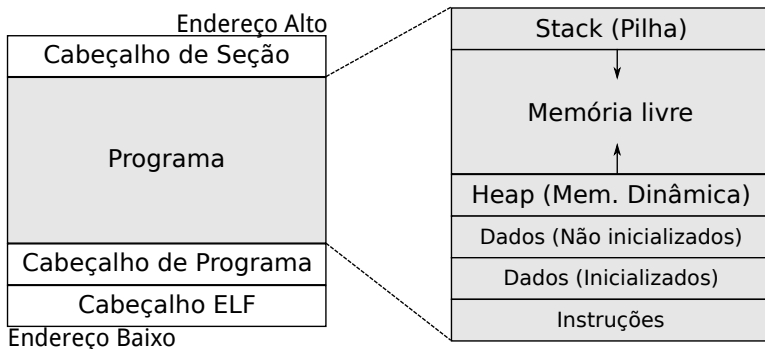
Variáveis Não Inicializadas e Estrutura de Programas Aula 04

Edmar André Bellorini

Ano letivo 2016

Introdução

► Estrutura de um programa ELF (Aula 02)



Dados Inicializados (.data)

- ▶ Variáveis com valores pré-definidos

```
section .data  
    simbolo: tamanho valor
```

- ▶ Especificação de tamanho por pseudo-instruções

```
db: byte  
dw: word (2B)  
dd: dualword (4B)  
dq: quadword (8B)
```

Dados Não Inicializados (.bss)

- ▶ Variáveis **sem** valores pré-definidos

```
section .bss
```

```
    simbolo: tamanho quantidade
```

- ▶ Uma linha é uma variável não inicializada
- ▶ Uma variável é definida por um símbolo (*label*)

```
    char c;
```

- ▶ c é o nome da variável, ou seja, o símbolo para referência
- ▶ Uma variável tem um tamanho em memória reservado por uma pseudo-instrução

```
    resb: byte
```

```
    resw: word (2B)
```

```
    resd: dword (4B)
```

```
    resq: quadword (8B)
```

Dados Não Inicializados - Exemplos

```
n1: resb 1 ; 1 byte
n2: resb 2 ; 2 bytes
n3: resb 3 ; 3 bytes
n4: resb 8 ; 8 bytes
n5: resw 1 ; 2 bytes
n6: resw 2 ; 4 bytes
n8: resw 2 ; 4 bytes
n9: resd 1 ; 4 bytes
na: resd 2 ; 8 bytes
```

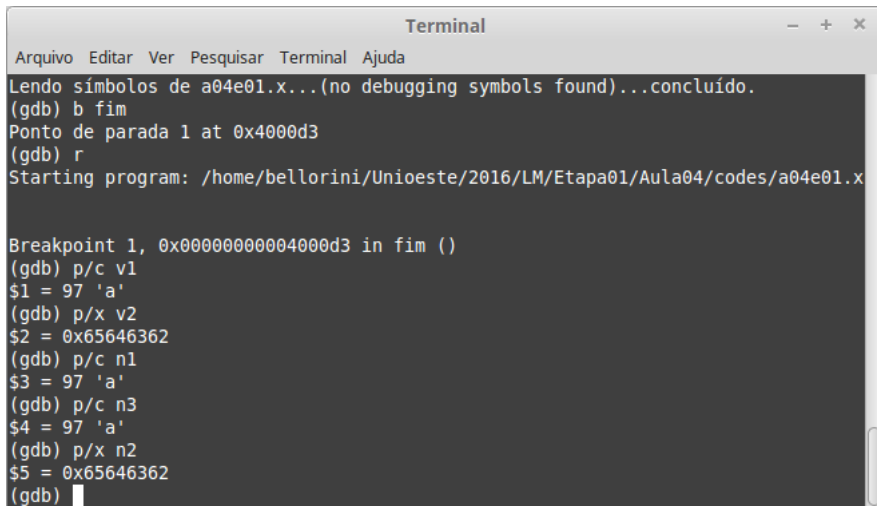
Instrução de movimentação de dados

- ▶ MOV: movimento (cópia) de dados da fonte para destino
`MOV destino, fonte`
 - ▶ Exatamente a mesma instrução para movimentação de dados inicializados

Exemplo a04e01.asm

```
1  section .data
2      v1: db 0x61
3      v2: dd 0x65646362
4
5  section .bss
6      n1: resb 1
7      n2: resd 1
8      n3: resb 1
9
10 section .text
11     global _start
12     _start:
13         mov al, [v1]
14         mov [n1], al
15         mov [n3], al
16         mov ebx, [v2]
17         mov [n2], ebx
18
19     fim:
20         mov eax, 1
21         mov ebx, 0
22         int 0x80
```

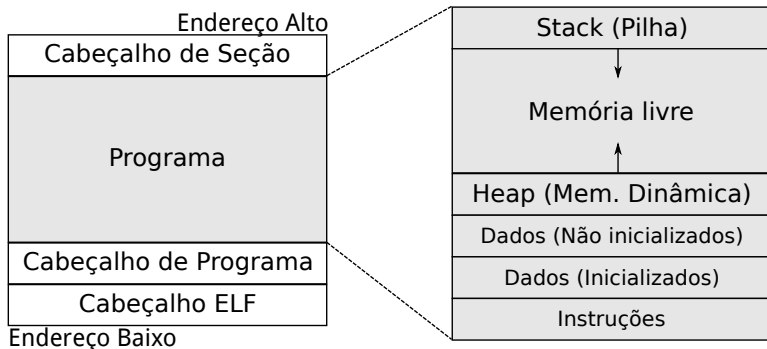
Debugger



```
Terminal
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
Lendo símbolos de a04e01.x...(no debugging symbols found)...concluído.
(gdb) b fim
Ponto de parada 1 at 0x4000d3
(gdb) r
Starting program: /home/bellorini/Unioeste/2016/LM/Etapa01/Aula04/codes/a04e01.x

Breakpoint 1, 0x00000000004000d3 in fim ()
(gdb) p/c v1
$1 = 97 'a'
(gdb) p/x v2
$2 = 0x65646362
(gdb) p/c n1
$3 = 97 'a'
(gdb) p/c n3
$4 = 97 'a'
(gdb) p/x n2
$5 = 0x65646362
(gdb) 
```


Esta imagem de novo?



- ▶ **Endereço das variáveis**
 - ▶ Dados Inicializados é mais “baixo”
 - ▶ Dados Não Inicializados é mais “alto”

Endereços das Variáveis

- ▶ Cada símbolo para uma variável é uma referência

```
section. data
```

```
    v1: db 0x61
```

- ▶ v1 é uma referência para o conteúdo 0x61
- ▶ Cada referência é um endereço de memória
 - ▶ v1 é na verdade um endereço de memória
 - ▶ É válido tanto para .data quanto para .bss
- ▶ Como descobrir os endereços em tempo de execução?
 - ▶ Instrução MOV

```
    mov rax, v1
```

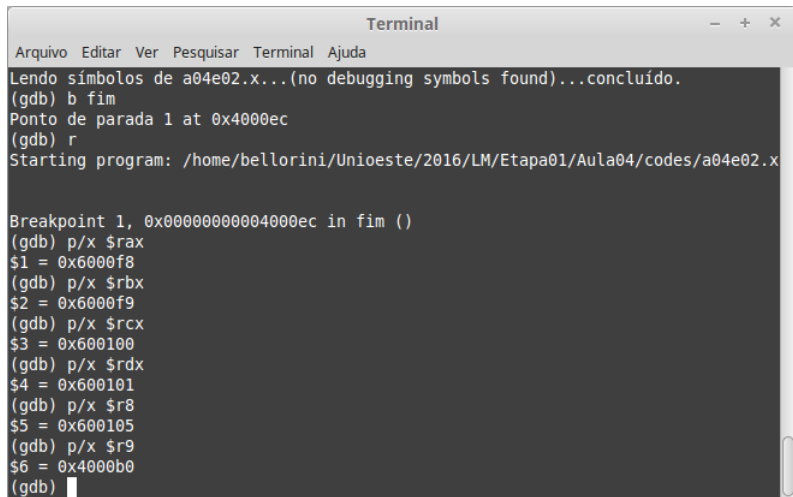
 - ▶ Registrador RAX contém o endereço de v1

Exemplo a04e02.asm

```
1  section .data
2      v1: db 0x61
3      v2: dd 0x65646362
4
5  section .bss
6      n1: resb 1
7      n2: resd 1
8      n3: resb 1
9
10 section .text
11     global _start
```

```
12  _start:
13      mov rax, v1
14      mov rbx, v2
15      mov rcx, n1
16      mov rdx, n2
17      mov r8 , n3
18      mov r9 , _start
19
20  fim:
21      mov eax, 1
22      mov ebx, 0
23      int 0x80
```

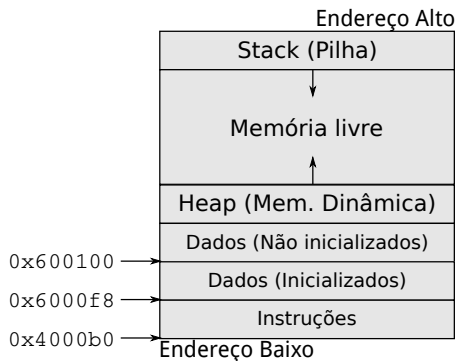
Debugger



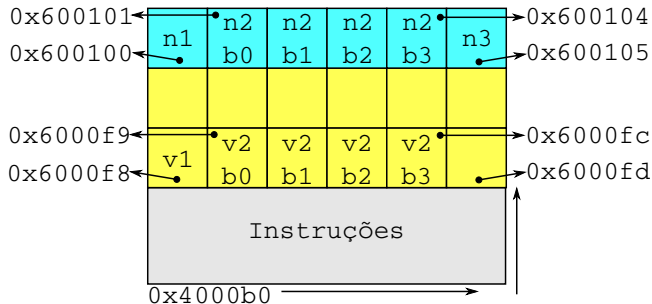
The image shows a terminal window titled "Terminal" with a menu bar containing "Arquivo", "Editar", "Ver", "Pesquisar", "Terminal", and "Ajuda". The terminal output shows the following sequence of events:

```
Lendo símbolos de a04e02.x...(no debugging symbols found)...concluído.  
(gdb) b fim  
Ponto de parada 1 at 0x4000ec  
(gdb) r  
Starting program: /home/bellorini/Unioeste/2016/LM/Etapa01/Aula04/codes/a04e02.x  
  
Breakpoint 1, 0x0000000004000ec in fim ()  
(gdb) p/x $rax  
$1 = 0x6000f8  
(gdb) p/x $rbx  
$2 = 0x6000f9  
(gdb) p/x $rcx  
$3 = 0x600100  
(gdb) p/x $rdx  
$4 = 0x600101  
(gdb) p/x $r8  
$5 = 0x600105  
(gdb) p/x $r9  
$6 = 0x4000b0  
(gdb) 
```

Estrutura do exemplo (Visão Abstrata)



Estrutura do exemplo (Visão Detalhada)



- Obs.: Os valores podem ser diferentes de acordo com a máquina/montador para um mesmo código

Comportamento

- ▶ Aula 02
 - ▶ Não existe “tipo de dados” em memória
 - ▶ É a instrução que determina o comportamento
- ▶ É possível indicar um possível tipo de dado
 - ▶ Já fazemos isso quando definimos uma variável inicializada

```
section .data
    v1: db 0x61
```

 - ▶ v1 é uma variável de 1 byte
 - ▶ o valor 0x61 é armazenado como um número hexadecimal
 - ▶ é possível usar outras bases numéricas

Números

- ▶ Bases numéricas aceitas

- ▶ Hexadecimal

- v1a: dq 0x61 ; *0x ou h*

- v1b: dq 61h

- ▶ Decimal

- v2a: dq 97d ; *d ou sem definicao*

- v2b: dq 97 ; *nasm assume base decimal*

- ▶ Octal

- v3: dq 141o

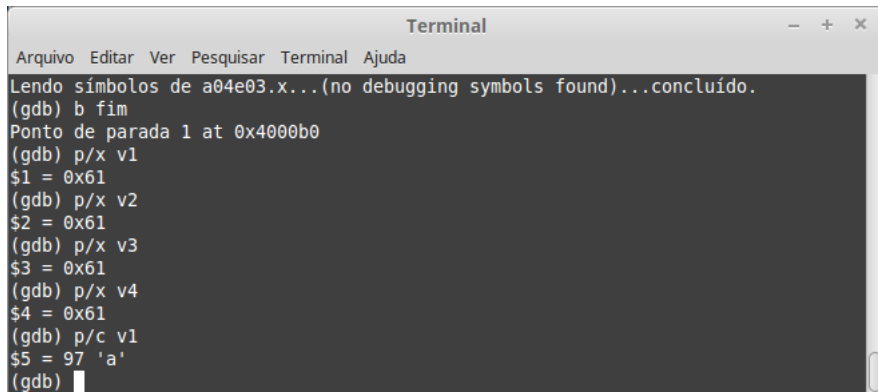
- ▶ Binário

- v4: dq 1100001b

Exemplo a04e03.asm

```
1  section .data
2      v1: dq 0x61
3      v2: dq 97d
4      v3: dq 141o
5      v4: dq 1100001b
6
7  section .text
8      global _start
9
10     _start:
11     fim:
12         mov rax, 1
13         mov rbx, 0
14         int 80h
```

Debugger



```
Terminal
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
Lendo símbolos de a04e03.x...(no debugging symbols found)...concluído.
(gdb) b fim
Ponto de parada 1 at 0x4000b0
(gdb) p/x v1
$1 = 0x61
(gdb) p/x v2
$2 = 0x61
(gdb) p/x v3
$3 = 0x61
(gdb) p/x v4
$4 = 0x61
(gdb) p/c v1
$5 = 97 'a'
(gdb) 
```

- ▶ $v1 = v2 = v3 = v4$
- ▶ 0x00000061
- ▶ 'a'

Números negativos

- ▶ Números são representados sempre em complemento de dois
 - ▶ Exemplo a04e04.asm

```
1  section .data
2      v1: dq 100d
3      v2: dq -100d
4
5  section .text
6      global _start
7
8  _start:
9  fim:
10     mov rax, 1
11     mov rbx, 0
12     int 80h
```

— + ×

Movimentação de dados numéricos com extensão de sinal/zero

- ▶ Movimentar (copiar) n bytes para p bytes
 - ▶ $n < p$
- ▶ Instrução movsx (*move sign-extends*)
 - ▶ Copia fonte para destino e mantém magnitude
 - ▶ Valores sinaizados (*signed*)
 - ▶ Sintaxe

MOVSX reg16,r/m8 ; 1b para 2b

MOVSX reg32,r/m8 ; 1b para 4b

MOVSX reg32,r/m16 ; 2b para 4b

MOVSX reg64,r/m8 ; 1b para 8b

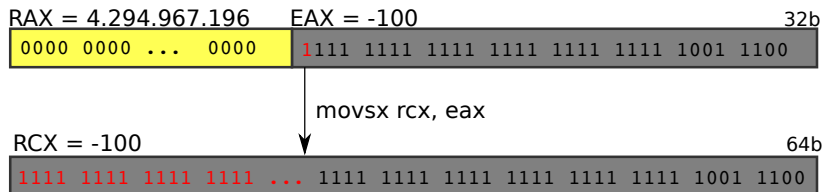
MOVSX reg64,r/m16 ; 2b para 8b

MOVSX reg64,r/m32 ; 4b para 8b

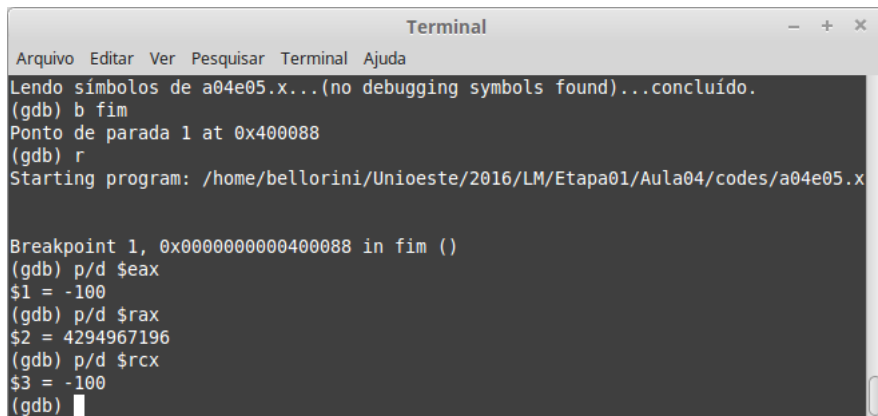
Exemplo a04e05.asm

```
1  section .text
2      global _start
3
4  _start:
5      mov     eax, -100
6      movsx   rcx, eax
7
8  fim:
9      mov     rax, 1
10     mov     rbx, 0
11     int     80h
```

Exemplo a04e05.asm



Debugger



The image shows a terminal window titled "Terminal" with a menu bar containing "Arquivo", "Editar", "Ver", "Pesquisar", "Terminal", and "Ajuda". The terminal output shows the following sequence of events:

```
Lendo símbolos de a04e05.x...(no debugging symbols found)...concluído.  
(gdb) b fim  
Ponto de parada 1 at 0x400088  
(gdb) r  
Starting program: /home/bellorini/Unioeste/2016/LM/Etapa01/Aula04/codes/a04e05.x  
  
Breakpoint 1, 0x0000000000400088 in fim ()  
(gdb) p/d $eax  
$1 = -100  
(gdb) p/d $rax  
$2 = 4294967196  
(gdb) p/d $rcx  
$3 = -100  
(gdb) 
```


Exercícios de Fixação

- ▶ EF0401: Escreva um código funcional (montável e linkável) que contenha a seguinte seção:

```
section .data
    v1: db 10d
    v2: dw -20d
    v3: dq -30d
```

- ▶ Crie a seção para variáveis não inicializadas que contenha as variáveis n1 (1 byte), n2 (2b) e n3 (4b)
- ▶ Realize as seguintes cópias no código principal
 - ▶ $n1 = v1$
 - ▶ $n2 = v2$
 - ▶ $n3 = v3$
 - ▶ Obs: mov não realiza movimentação de memória para memória (ver aula 03)

Exercícios de Fixação

- ▶ EF0402: Construa um código funcional que copie os dados da seção `.data` para a seção `.bss`

- ▶ Seção `.data`

```
section .data
    v1: db 10d
    v2: dw -20d
    v3: dw -30d
```

- ▶ Seção `.bss`

```
section .bss
    n1: resq 1
    n2: resq 1
    n3: resq 1
```

- ▶ Importante: respeitar o tamanho em bytes das variáveis.

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: P04