Linguagens de montagem
Capítulo 3 – Introdução a linguagens de montagem

Ricardo Anido
Instituto de Computação
Unicamp

Linguagens de montagem

- Uma linguagem de montagem é basicamente uma linguagem de programação muito simples.
- Compilador de uma linguagem de montagem é chamado de montador (em inglês, *assembler*).
- Como em um programa em linguagem de alto nível, programa em linguagem de montagem é uma sequência de comandos.
- Comandos são muito simples!

Introdução a Linguagem de Montagem

- No LEG um *símbolo* é uma sequência de letras, dígitos ou o caractere '_', que se inicia com uma letra.
- Comandos da linguagem de montagem são identificados por *símbolos reservados* do montador (têm um significado fixo para o montador e não podem ter o seu significado redefinido pelo programador).
- Cada comando em linguagem de montagem é traduzido pelo montador em uma instrução de máquina.
- Operandos devem aparecer à direita do comando, e, se mais de um operando é necessário, devem ser separados por vírgulas.
- No LEG usaremos a convenção de que o operando que será modificado pela instrução aparece mais à esquerda.

Exemplo de comando no LEG

Um exemplo de comando em linguagem de montagem, para uma instrução do LEG que já vimos, é

```
set  r5,0x7000
```

Quando processado pelo montador, este comando em linguagem de montagem é traduzido na sequência de palavras binárias

```
0x02000500
0x00007000
```

Exemplo de comando no LEG

Outro exemplo:

```
add  r5,r6
```

Quando processado pelo montador, este comando em linguagem de montagem é traduzido na sequência de palavras binárias

```
0x11000506
```

Comentários

- Na linguagem de montagem do LEG, um comentário é iniciado pelo caractere ‘’ e se estende até o final da linha.
- Comentários e linhas em branco são desconsiderados pelo montador.

Formato geral de um comando

Cada linha de um programa em linguagem de montagem do LEG pode ter o seguinte formato, em que [*comp*] indica que a presença do componente *comp* é opcional (a ordem de cada um dos componentes na linha é fixa):

[*rótulo:*] [*comando*] [*lista_de_operandos*] [@ *comentário*]

Formato geral de um comando

- *rótulo* é um símbolo, definido pelo programador, associado a um endereço de memória específico.
- Rótulos são usados para definir pontos importantes em um programa (início de um procedimento, por exemplo), para definir variáveis, ou simplesmente para documentação.

Exemplo de programa em LEG

```
@ um pequeno programa para somar 0x7000 com 0x400 e colocar
@ o resultado no registrador r5
inicio:
    set    r5,0x7000    @ carrega primeiro termo da soma
    set    r6,0x0400    @ carrega segundo termo
    add    r5,r6        @ e soma os dois termos
```

Quando processado pelo montador, este comando em linguagem de montagem é traduzido na sequência de palavras binárias do programa montado à mão, usado como exemplo anteriormente:

```
0x02000500
0x00007000
0x02000600
0x00000400
0x11000506
```

Diretivas

- Além de comandos, que representam as instruções do processador, uma linguagem de montagem inclui também *diretivas*.
- Diretivas, ao contrário dos comandos, não são traduzidas em código de máquina.
- Servem para transmitir informações adicionais ao montador, como por exemplo definir uma constante que será utilizada em vários pontos do programa.

Diretiva de definição de constantes

Constantes podem ser definidas em qualquer parte do programa em linguagem de montagem, mas usualmente são definidas no início do programa.Uma constante é introduzida pela diretiva .EQU (abreviatura de *equate*, em inglês, 'é igual a'), cuja sintaxe é

nome `.equ` *valor*

Exemplo de definição de constantes

```
@ definição de constantes

VERDADEIRO      .equ    0xff            @ um valor hexadecimal
FALSO           .equ    0               @ um valor decimal
MAXVAL          .equ    1000            @ um outro valor decimal
MINVAL          .equ    MAXVAL/2        @ uma expressão inteira
```

Diretiva para reservar de espaço em memória

- Em linguagens de montagem, para definir variáveis simplesmente associamos um símbolo (o nome da variável) a um endereço ou conjunto de endereços da memória.
- Diferentemente de linguagens de alto nível, não há informação sobre o tipo do dado que pode ser armazenado no endereço de memória associado ao nome.
- o endereço de memória associado pode conter um inteiro, um valor real, ou qualquer outro tipo de dados.

Diretiva para reservar de espaço em memória

Podemos reservar espaço na memória no LEG de duas maneiras:

- inicializando o espaço com um valor conhecido
- deixando o espaço não inicializado (e portanto contendo um valor desconhecido)

Diretiva de reserva de espaço em memória sem inicialização

Para reservar espaço em memória, sem inicialização, usamos a diretiva .SKIP, que tem o formato geral

[*rótulo:*] `.skip` *expressão_inteira* [@ *comentário*]

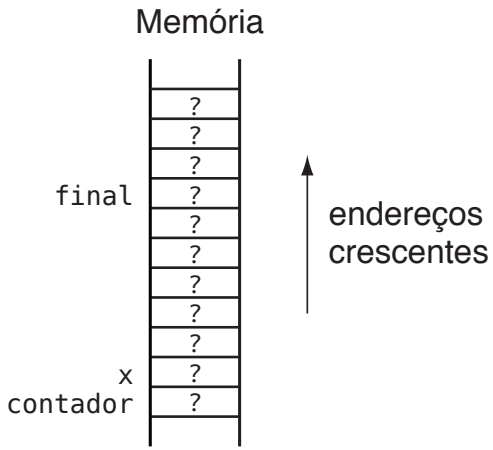
Esta diretiva reserva *expressão_inteira* bytes de memória, e associa o primeiro endereço a *rótulo*.

Exemplo de reserva de espaço com .SKIP

```
@ exemplo de reserva de espaço na memória para variáveis
@ vamos primeiro definir uma constante

TAMANHO         .equ    3

@ agora reservamos espaço para algumas variáveis
contador:       .skip   1
x:              .skip   TAMANHO*2
final:          .skip   4
```



Reserva e inicialização de bytes

A diretiva .BYTE reserva e inicializa bytes na memória:

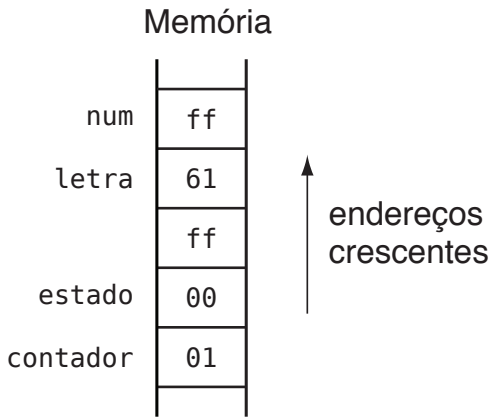
[*rótulo:*] `.byte` [*lista_de_valores*] [@ *comentário*]

A *lista_de_valores* é uma lista, separada por vírgulas, em que cada elemento pode ser uma expressão inteira, um caractere entre aspas simples, ou uma sequência de caracteres entre aspas simples.

Exemplo de reserva e inicialização com .BYTE

```
@ exemplo de reserva de espaço na memória e inicialização de variáveis
@ vamos primeiro definir uma constante
MAXVAL          .equ    256

@ agora reservamos espaço e inicializamos algumas variáveis
contador:       .byte   0x01            @ um valor hexadecimal
estado:         .byte   0, MAXVAL-1     @ uma lista de valores
letra:          .byte   'a'             @ um caractere entre aspas simples
num:            .byte   -1              @ um valor decimal
```



Reserva e inicialização de palavras

A diretiva .WORD reserva e inicializa palavras na memória:

[*rótulo:*] `.word` [*lista_de_valores*] [@ *comentário*]

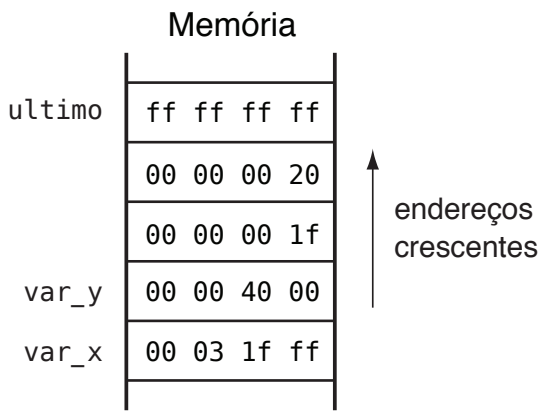
onde *lista_de_valores* é uma lista, separada por vírgulas, em que cada elemento pode ser uma expressão numérica, um caractere entre aspas simples, ou uma sequência de caracteres entre aspas simples.

Exemplo de reserva e inicialização com .WORD

```
@ exemplo de uso da diretiva .word

@ primeiro definimos algumas constantes
ALTO            .equ    0x32000
BAIXO           .equ    0x2000

var_x:          .word   ALTO-1
var_y:          .word   BAIXO*2
                .word   31,32
ultimo:         .word   -1
```



Diretiva para indicar endereço de montagem

- O montador mantém um *apontador de montagem*
- Apontador de montagem é incrementado do número de palavras da instrução a cada instrução montada.
- Apontador de montagem também é incrementado quando espaço é reservado com diretivas.

Diretiva para indicar endereço de montagem

Para alterar o valor do apontador de montagem, utilizamos a diretiva `.ORG`, que tem o formato

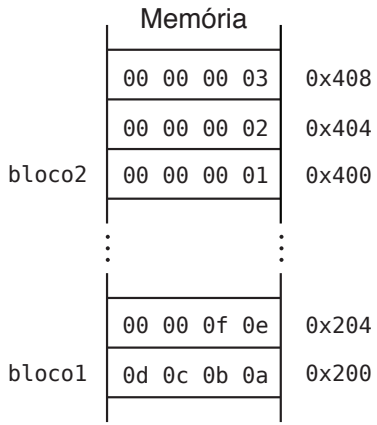
`.org` *expressão_inteira* [`@` *comentário*]

Diretiva para indicar endereço de montagem

```
@ exemplo de uso de diretiva de ponto de montagem
MAXVAL          .equ    256
TAM_BLOCO       .equ    16
NUM_BLOCO       .equ    64

@ aqui definimos o endereço inicial de montagem de um bloco
                .org    0x200
bloco1:         .byte   10,11,12,13,14,15

@ este outro bloco deve começar em outro endereço
                .org    TAM_BLOCO*NUM_BLOCO   @ 16*64 = 0x400
bloco2:         .word   0,1,2
```



Como funciona um montador

```
TAM_BLOCO .equ  256

@ reserva de espaço e inicialização de uma variável inteira
var1:     .word -1

@ suponha que o programa propriamente deva se iniciar no
@ endereço 0x1000. Assim, alteramos o endereço de montagem
          .org  0x1000
inicio:
          set   r5,var2        ; alguns exemplos de instruções
          set   r6,TAM_BLOCO   ; sem nenhum significado específico,
          add   r5,r6          ; usadas apenas para ilustrar
          set   r7,var1        ; o funcionamento do montador
@ o programa continua com outros comandos não mostrados
          ...
@ o endereço de montagem é alterado novamente
          .org  0x2000
@ vamos reservar espaço e inicializar uma outra variável inteira
var2:     .word TAM_BLOCO*2
          .end
```

Como funciona um montador

- Vamos descrever o funcionamento de um montador simples, de dois passos.
- A cada passo, lê o programa fonte do início ao fim.
- No primeiro passo o montador determina os valores de todos os rótulos e constantes definidos no programa.
- No segundo passo o montador realmente produz o arquivo binário executável.

Como funciona um montador

O montador mantém:

- uma variável *PontoDeMontagem*, que armazena o endereço corrente de montagem, inicialmente zero (primeiro endereço de memória);
- uma *Tabela de Símbolos*, que associa cada símbolo do programa fonte a um *valor* e a um *tipo* (que pode ser Comando, Constante, Registrador, Rótulo ou Indefinido).