

Arquitetura e Organização de Computadores

PROFESSOR ESP. GIULIANO ROBLEDO ZUCOLOTO MOREIRA

ENGENHARIA DA COMPUTAÇÃO

4º SEMESTRE

2018/2

Ambiente Virtual de Aprendizagem

- ▶ O AVA será utilizado como repositório para o envio de materiais, links, listas de exercícios, dentre outras atividades.

Avaliações

▶ LE

- ▶ Listas de exercícios, aplicadas conforme demanda detectada pelo professor. À nota LE são distribuídos 10 (dez) pontos. O valor de cada lista de exercícios é atribuído ao final do semestre, onde divide-se o valor da pontuação atribuída pelo número de listas aplicadas. Cada questão nas listas terá seu valor a julgamento do Professor. Dos valores das listas computarão nota para o aluno apenas as questões cuja resposta estiver correta.

▶ Trabalho

- ▶ **TPE:** Parte escrita valendo 10 (dez) pontos, distribuídos da seguinte maneira:
 - ▶ Adequação às normas científicas estabelecidas: 3,0 (três) pontos.
 - ▶ Conteúdo do trabalho: 7,0 (sete) pontos.
- ▶ **TA:** Apresentação pública na sala de projeções valendo 10 (dez) pontos , distribuídos da seguinte maneira:
 - ▶ Adequação às normas/normas científicas estabelecidas: 3,0 (três) pontos.
 - ▶ Segurança e postura na apresentação: 2,0 (dois) pontos.
 - ▶ Domínio do conteúdo apresentado: 5,0 (cinco) pontos.

Avaliações

- ▶ LE: Listas de exercícios valendo 10 (dez) pontos.
- ▶ Trabalho fragmentado em duas notas
 - ▶ TPE: Parte escrita valendo 10 (dez) pontos.
 - ▶ TA: Apresentação pública na sala de projeções valendo 10 (dez) pontos.
- ▶ A **nota final NF** é calculada da seguinte maneira:

$$NF = \frac{LE + TPE + TA}{3}$$

Ementa:

- ▶ Linguagem de Montagem.
- ▶ Formato de instruções.
- ▶ Ligadores e Carregadores.
- ▶ Memórias: tipos, organização e endereçamento.
- ▶ CPU's: barramento e unidade de controle.
- ▶ Acesso direto à memória (DMA).
- ▶ Interrupções e sua implementação.
- ▶ Arquitetura de computadores típicos.

Linguagem de Montagem

- ▶ Linguagem de Montagem.
- ▶ Formato de instruções.
- ▶ Ligadores e Carregadores.
- ▶ Memórias: tipos, organização e endereçamento.
- ▶ CPU's: barramento e unidade de controle.
- ▶ Acesso direto à memória (DMA).
- ▶ Interrupções e sua implementação.
- ▶ Arquitetura de computadores típicos.

Referências

- ▶ [1] STALLINGS: 350-351.
- ▶ [2] TANENBAUM, AUSTIN: 407-435.

Definição

A linguagem de montagem é uma linguagem que está a um passo de distância da linguagem de máquina. [1]

Uma linguagem de montagem pura é uma linguagem na qual cada declaração produz exatamente uma instrução de máquina [2].

Uma instrução de máquina é um conjunto organizado de bits que quando inserido na Unidade de Controle produz uma ação.

De bits à montagem

*Binário
(opcode)*

Hexadecimal

*Simbólico
(mnemônico)*

*Assembly
(montagem)*

Programa binário

$$N = I + J + K$$

$$I = 2$$

$$J = 3$$

$$K = 4$$

Endereço	Conteúdo			
PROGRAMA				
101	0010	0010	101	2201
102	0001	0010	102	1202
103	0001	0010	103	1203
104	0011	0010	104	3204
DADOS				
201	0000	0000	201	0002
202	0000	0000	202	0003
203	0000	0000	203	0004
204	0000	0000	204	0000

Programa binário

$$N = I + J + K$$

$$I = 2$$

$$J = 3$$

$$K = 4$$

Endereço	Conteúdo			
PROGRAMA				
101	0010	0010	101	2201
102	0001	0010	102	1202
103	0001	0010	103	1203
104	0011	0010	104	3204
DADOS				
201	0000	0000	201	0002 I
202	0000	0000	202	0003 J
203	0000	0000	203	0004 K
204	0000	0000	204	0000 N

Programa hexadecimal

$$N = I + J + K$$

$$I = 2$$

$$J = 3$$

$$K = 4$$

Endereço	Conteúdo
PROGRAMA	
101	2201
102	1202
103	1203
104	3204
DADOS	
201	0002
202	0003
203	0004
204	0000

Programa hexadecimal

$$N = I + J + K$$

$$I = 2$$

$$J = 3$$

$$K = 4$$

Endereço	Conteúdo
PROGRAMA	
101	2 201
102	1 202
103	1 203
104	3 204
DADOS	
201	0002 I
202	0003 J
203	0004 K
204	0000 N

Programa simbólico

(mnemônico)

$$N = I + J + K$$

$$I = 2$$

$$J = 3$$

$$K = 4$$

Endereço	Instrução	
PROGRAMA		
101	LDA	201
102	ADD	202
103	ADD	203
104	STA	204
DADOS		
201	DAT	0002
202	DAT	0003
203	DAT	0004
204	DAT	0000

Programa simbólico

(mnemônico)

$$N = I + J + K$$

$$I = 2$$

$$J = 3$$

$$K = 4$$

Endereço	Instrução	
PROGRAMA		
101	LDA	201
102	ADD	202
103	ADD	203
104	STA	204
DADOS		
201	DAT	0002 I
202	DAT	0003 J
203	DAT	0004 K
204	DAT	0000 N

Endereços absolutos

Comparando as técnicas de programação apresentadas é perceptível que todas fizeram uso de endereços absolutos.

- ▶ **O que significa na prática usar endereços absolutos?**
 - ▶ ... significa que o programa e os dados podem ser carregados em apenas uma posição da memória e nós precisamos saber esta posição antecipadamente. Pior ainda, suponha que queiramos mudar o programa um dia adicionando ou excluindo uma linha. Isso irá alterar os endereços de todas as palavras subsequentes. (STALLINGS, 2010, p.351).

Programa *assembly*

$N = I + J + K$

$I = 2$

$J = 3$

$K = 4$

Rótulo	Operação	Operando
PROGRAMA		
FORMUL	LDA	I
	ADD	J
	ADD	K
	STA	N
DADOS		
I	DATA	2
J	DATA	3
K	DATA	4
N	DATA	0

Elementos da linguagem de montagem

Rótulo

Mnemônico

Operando(s)

Comentário
(opcional)

Endereços simbólicos

Observando o programa assembly percebe-se que os endereços numéricos foram substituídos por rótulos, flexibilizando a organização do programa em memória.

▶ Como funciona o sistema de endereços simbólicos?

- ▶ Cada linha ainda consiste de três campos. O primeiro campo ainda é para endereço, porém um símbolo é usado no lugar de um endereço numérico absoluto. Algumas linhas não possuem endereço, o que implica que o endereço dessa linha é um a mais do que o endereço da linha anterior. Para instruções que referenciam à memória, o terceiro campo também contém um endereço simbólico. (STALLINGS, 2010, p.351).

▶ Melhorias?

“Com estas últimas melhorias nós temos uma linguagem de montagem (*linguagem assembly*).” (STALLINGS, 2010, p.351).

Endereços simbólicos

A organização do programa simbólico em memória é feita por um ***assembler*** (montador).

▶ Como é gerado um programa com endereços simbólicos?

- ▶ Programas escritos em linguagem de montagem (programas *assembly*) são traduzidos para a linguagem de máquina por um *assembler* (montador). Esse programa precisa fazer não apenas a tradução simbólica discutida anteriormente, como deve também atribuir endereços de memória para endereços simbólicos. (STALLINGS, 2010, p.351).

▶ Resultados

O desenvolvimento da linguagem de montagem foi um grande marco na evolução da tecnologia de computadores. Foi o primeiro passo para linguagens de alto nível usadas atualmente. Embora poucos programadores usem linguagem de montagem, teoricamente todas as máquinas fornecem uma. Elas são usadas, quando usadas, para programas de sistema como compiladores e rotinas de E/S. (STALLINGS, 2010, p.351).

Destques

Momento de reflexão
para expansão do
raciocínio a respeito de
linguagem de montagem.

▶ Da abordagem do assunto destaca-se:

- ▶ Uma linguagem de montagem (*assembly*) é uma representação simbólica da linguagem de máquinas de um processador específico, acrescida de tipos de instruções adicionais que facilitam a escrita do programa e que fornecem as instruções para o montador (*assembler*).
- ▶ Um montador é um programa que traduz a linguagem de montagem em código de máquinas.
- ▶ O primeiro passo na criação de um processo ativo é carregar um programa na memória principal e criar uma imagem do processo.
- ▶ Um *linker* (ligador) é usado para resolver quaisquer referências entre os módulos carregados.
- ▶ (STALLINGS, 2010, p.581).

Caixinhas importantes

Assembly
(linguagem de
montagem)

Assembler
(montador)

Linker (ligador)

Loader
(carregador)


Montadores

Existem duas abordagens para os montadores: montador de dois passos e montador de um passo.

O montador é um software utilitário que recebe como entrada um programa em linguagem de máquina (*assembly*) e produz o código objeto como saída. O código objeto é um arquivo binário. O montador enxerga este arquivo como um bloco de memória iniciando na posição relativa 0. (STALLINGS, 2010, p.588).

Montador de dois passos

Existem duas abordagens para os montadores: montador de dois passos e montador de um passo.



O montador faz duas passagens pelo código fonte [...]. No primeiro passo o montador se preocupa apenas com definições dos rótulos. O primeiro passo é usado para construir uma tabela de símbolos que contém uma lista de todos os rótulos e seus valores de contador de posição (LC, do inglês Location Counter). (STALLINGS, 2010, p.588).

O segundo passo lê o programa novamente desde o começo. Cada instrução é traduzida no código binário de máquina apropriado. (STALLINGS, 2010, p.589).

Tradução para código binário

Existem duas abordagens para os montadores: montador de dois passos e montador de um passo.

A tradução envolve as seguintes operações:

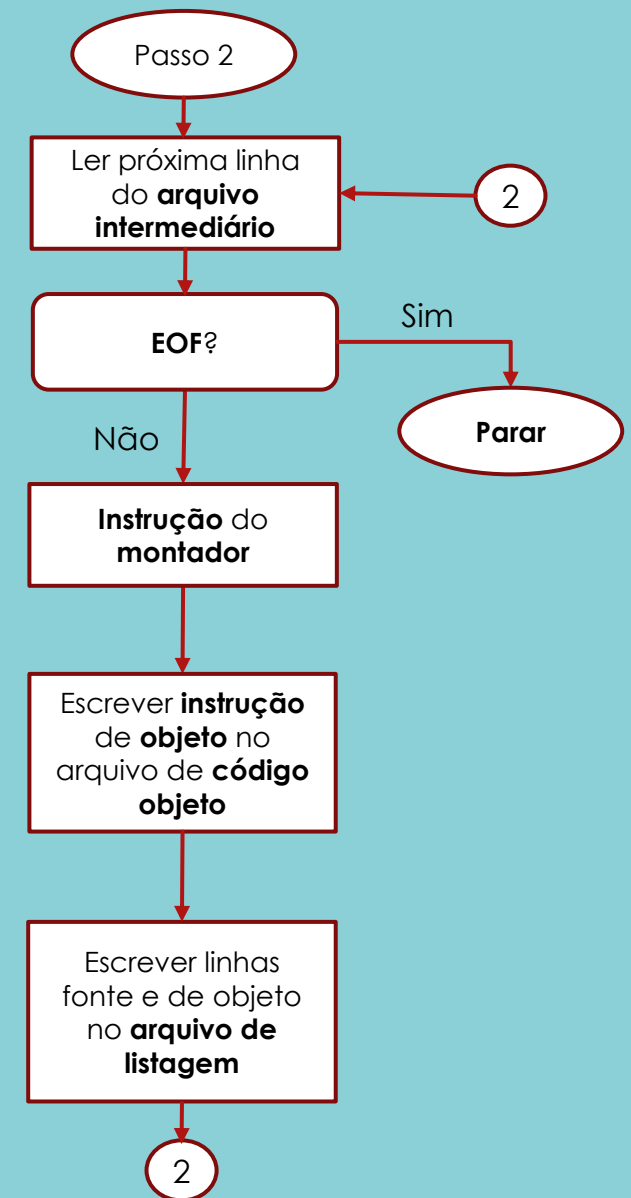
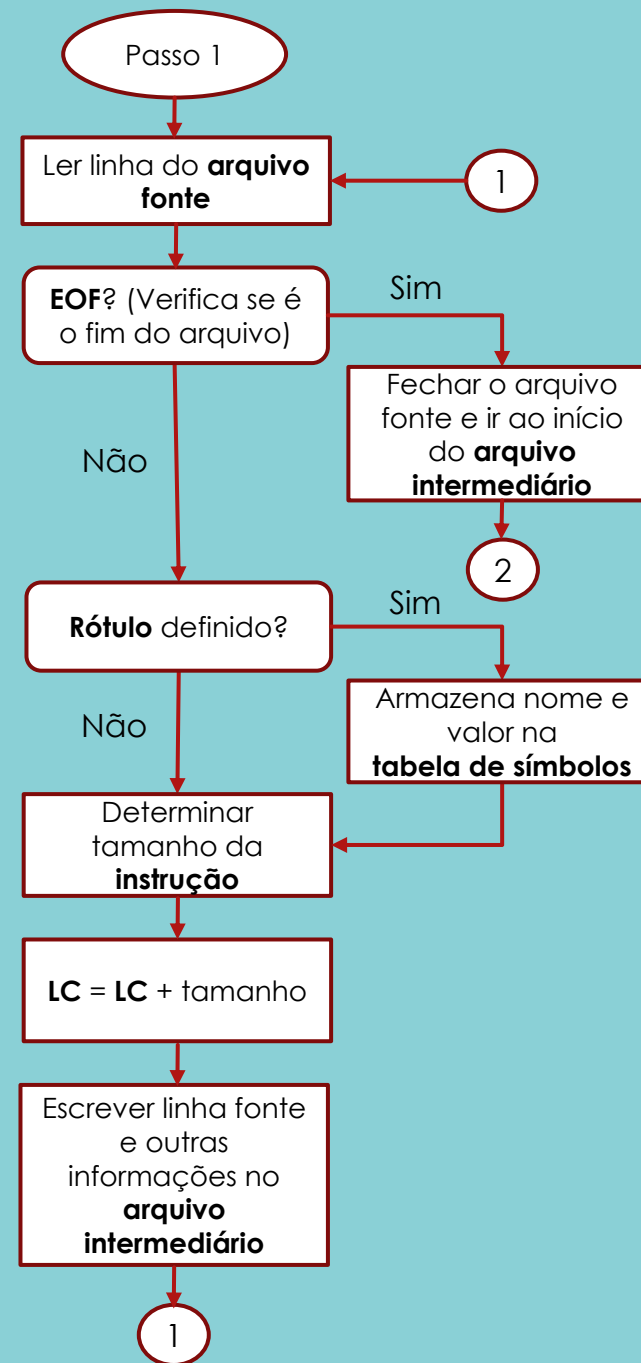
1. Traduzir mnemônico em *opcode* binário.
2. Usar o *opcode* para determinar o formato da instrução, a posição e o tamanho de vários campos na instrução.
3. Traduzir o nome de cada operando para o registrador ou código de memória apropriado.
4. Traduzir cada valor imediato em uma cadeia binária.
5. Traduzir quaisquer referências a rótulos em valores de LC apropriado usando a tabela de símbolos.
6. Definir quaisquer outros bits necessários dentro da instrução, incluindo indicadores do modo de endereçamento, bits de código condicionais e assim por diante.

(STALLINGS, 2010, p.588).

Montador de dois passos: Fluxograma

O montador de dois passos faz uso de um arquivo intermediário.

Imagens: STALLINGS (2008, p. 590).



Montador de um passo

Existem duas abordagens para os montadores: montador de dois passos e montador de um passo.

É possível implementar um montador que faz uma única passagem pelo código fonte (sem contar o passo para processar macros). A principal dificuldade em tentar montar um programa em uma única passagem envolve referências futuras a rótulos. Os operandos das instruções podem ser símbolos que ainda não foram definidos no programa fonte. Portanto, o montador não sabe qual endereço relativo a inserir na instrução traduzida.

(STALLINGS, 2010, p.588).

Linker (Ligador)


Existem duas abordagens para os montadores: montador de dois passos e montador de um passo.

É possível implementar um montador que faz uma única passagem pelo código fonte (sem contar o passo para processar macros). A principal dificuldade em tentar montar um programa em uma única passagem envolve referências futuras a rótulos. Os operandos das instruções podem ser símbolos que ainda não foram definidos no programa fonte. Portanto, o montador não sabe qual endereço relativo a inserir na instrução traduzida.

(STALLINGS, 2010, p.588).

Loader (Carregador)

Existem duas abordagens para os montadores: montador de dois passos e montador de um passo.



É possível implementar um montador que faz uma única passagem pelo código fonte (sem contar o passo para processar macros). A principal dificuldade em tentar montar um programa em uma única passagem envolve referências futuras a rótulos. Os operandos das instruções podem ser símbolos que ainda não foram definidos no programa fonte. Portanto, o montador não sabe qual endereço relativo a inserir na instrução traduzida.

(STALLINGS, 2010, p.588).

Sentenças da linguagem de montagem

Instrução

Diretiva

Definição
de macro

Comentário

Linguagem de montagem do Microprocessador Intel 8085

Conhecendo linguagens de montagem.

Conteúdo fragmentado em arquivos PDF e em *links* no tópico **Linguagem de Montagem** disponível no AVA.