PCS3616

Programação de Sistemas

(Sistemas de Programação)

Semana 9, Aula 15

Editores de Ligação e Relocadores

Programação em linguagem de montagem

Escola Politécnica da Universidade de São Paulo

Roteiro

1. Ligador
   - Exemplo de operação
2. Relocador
   - Exemplo de operação
3. Retrospecto da disciplina

Exemplo de Operação do Relocador (1)

- Usando o exemplo do ligador, com três módulos de programa, e admitindo-se que a base de relocação utilizada é 100, os programas ligados anteriormente passam a ter seus endereços absolutos recalculados.

- Além disso, todas as linhas de código cujo valor admite um operando relocável devem ser alteradas e seus valores passam a conter apenas operandos absolutos, recalculados a partir da base de relocação.

Exemplo de Operação do Relocador (2)

Módulos Ligados: (850 bytes)

100 A (considera-se base de alocação 0)

200 B (considera-se base de alocação 0)

620 C (considera-se base de alocação 500 = tamanho do módulo 1)

750 D (considera-se base de alocação 700 = soma dos tamanhos dos módulos 1 e 2)

Relocador gera:

Módulos Ligados Absolutos: (850 bytes)

200 A (base de relocação 100)

300 B (base de relocação 100)

720 C (base de relocação 100)

850 D (base de relocação 100)

Exemplo

- Para testar a implementação do ligador, utiliza-se os dois módulos *somador.asm* e *principal.asm* da Aula 13:
    1. Fornecem-se ao montador relocável os arquivos fonte *somador.asm* e *principal.asm),* para gerar os códigos-objeto com as informações de endereços pendentes (*somador.mvn* e *principal.mvn)*.
    2. Fornecem-se estes últimos arquivos como entrada ao ligador, para gerar um único arquivo relocável (*exem.lig*), ainda com endereços relativos.
    3. Finalmente, fornece-se este último arquivo como entrada ao relocador, para gerar um único arquivo executável (*exem.mvn*).
    - Os conteúdos dos arquivos mencionados são apresentados em seguida. O conteúdo de *exem.lig* corresponde à ordem de ligação: *somador.mvn* e *principal.mvn*.

Exemplo

- somador.asm

```
; Somador
; *******
; Somador que recebe duas entradas, nas posições
; ENTRADA1 e ENTRADA2, e coloca o resultado da
; soma na posição SAIDA (externa).

SOMADOR >
ENTRADA1 >
ENTRADA2 >
SAIDA <

& /0000                    ; Origem relocável
; Entradas do programa.
ENTRADA1 K /0000
ENTRADA2 K /0000

; Programa
SOMADOR  JP /000           ; Ponto de entrada da subrotina
         LD ENTRADA1
         +  ENTRADA2
         MM SAIDA          ; Colocando na saída
         RS SOMADOR        ; Retornando
```

Exemplo

- principal.asm

```
; Principal
; *********
; Programa principal que chama o somador.

SOMADOR <
ENTRADA1 <
ENTRADA2 <
SAIDA >

@ /0000
        JP INICIO
VALOR1  K =50           ; valor 1 a somar
VALOR2  K #101101       ; valor 2 a somar
SAIDA   K /0000

INICIO  LD VALOR1       ; passando as variáveis
        MM ENTRADA1
        LD VALOR2
        MM ENTRADA2
        SC SOMADOR      ; chamando o somador
        HM /00
```

Exemplo

somador.mvn

```
2004 0000 ; "SOMADOR>"
2000 0000 ; "ENTRADA1>"
2002 0000 ; "ENTRADA2>"
4000 0000 ; "SAIDA<"
8000 0000
8002 0000
8004 0000
a006 8000
a008 4002
d00a 9000
a00c b004
```

principal.mvn

```
4000 0000 ; "SOMADOR<"
4001 0000 ; "ENTRADA1<"
4002 0000 ; "ENTRADA2<"
0006 0000 ; "SAIDA>"
0000 0008
0002 0032
0004 002d
0006 0000
0008 8002
500a 9001
000c 8004
500e 9002
5010 a000
0012 c000
```

Códigos-objeto produzidos pelo montador relocável e submetidos ao ligador

exem.lig

```
2004 0000 ; "SOMADOR>"
2000 0000 ; "ENTRADA1>"
2002 0000 ; "ENTRADA2>"
8000 0000
8002 0000
8004 0000
a006 8000
a008 4002
800a 9006
a00c b004
0006 0000 ; "SAIDA>"
0000 0008
0002 0032
0004 002d
0006 0000
0008 8002
200a 9000
000c 8004
200e 9002
2010 a004
0012 c000
```

Código-objeto produzido pelo ligador

exem.mvn

```
0100 0000
0102 0000
0104 0000
0106 8100
0108 4102
010a 9006
010c b104
0000 0008
0002 0032
0004 002d
0006 0000
0008 8002
000a 9100
000c 8004
000e 9102
0010 a104
0012 c000
```

Código-objeto produzido pelo relocador (base =100)

Retrospecto da Disciplina

- Esta seção da aula tem como objetivo promover uma recapitulação da disciplina até o momento, recordando o encadeamento dos assuntos apresentados e ressaltando seu inter-relacionamento em um sistema básico.

- Para fechar o panorama dos programas de sistemas, ainda é necessário estudar dois programas importantes: o compilador e o sistema operacional, que não serão vistos aqui por serem ministrados em outras disciplinas.

- Nesta disciplina, a tônica é o estudo da funcionalidade dos diversos programas de um sistema básico, a estrutura e organização de sua lógica e de seus dados.

Máquinas de Turing e de Von Neumann

- Nas primeiras aulas mostrou-se, em termos pragmáticos, um modelo da **máquina de Turing** recordando que este é o modelo conceitual de computação estudado na teoria.
  - Máquinas de Turing constituem versões formais de algoritmos (Tese de Church-Turing).
  - É possível construir uma **máquina de Turing Universal**, capaz de computar cada uma das funções definidas por qualquer outra máquina de Turing.
- Constatou-se que a resolução de problemas usando máquinas de Turing é pouco eficiente. Neste momento, foi introduzido o **modelo de von Neumann**.
  - Embora do ponto de vista teórico não haja vantagem do uso deste (ou de qualquer outro) modelo em relação à máquina de Turing, em termos práticos a máquina de Von Neumann em geral permite realizar muito mais eficientemente os algoritmos, graças ao acesso aleatório às posições de memória, impossível na máquina de Turing, que usa fitas com acesso seqüencial.

Programação da MVN

- A partir desse ponto, a disciplina concentrou-se na implementação de programas para serem executados em uma máquina de von Neumann particular, a MVN.
  - A Máquina de Von Neumann adotada na disciplina é uma versão muito simplificada e didática de computadores reais.
  - Computadores eletrônicos têm a mesma estrutura e princípio de funcionamento, mas oferecem um conjunto de instruções maior e mais poderoso, o que facilita a escrita de programas (a disciplina *Organização de Computadores* ilustra este aspecto).

Programação de Sistemas

- A construção de programas diretamente em formato binário não se mostrou conveniente nem simples.
- Foi vantajoso construir uma abstração simbólica sobre a linguagem de máquina – a linguagem de montagem.
  - Cada instrução passou a ser representada por um **mnemônico**
  - Algumas pseudo-instruções foram introduzidas
  - Rótulos simbólicos substituíram endereços numéricos
- Do mesmo modo que o simulador MVN, construiu-se um montador na linguagem Java, um programa capaz de receber um programa fonte em linguagem de montagem e convertê-lo automaticamente para código de máquina.
- O primeiro montador desenvolvido era absoluto, i.e., o código de máquina gerado era carregado e executado em endereços absolutos definidos em relação a uma origem especificada no programa fonte.

Programação de Sistemas (continuação)

- Nessa versão do montador, não era possível reutilizar diretamente partes de programas já construídos, devido a conflitos de endereços entre a parte reutilizada e o novo código em desenvolvimento.
- Para resolver tais problemas foi introduzido o conceito de **relocabilidade**, que permite ao montador gerar um código que possa ser carregado e executado em diferentes regiões de memória, permitindo o uso de módulos externos.
  - Entretanto é necessário introduzir duas novas funcionalidades, a **ligação** e a **relocação** (podem ser implementadas como programas separados ou como um único módulo).
  - A função do **ligador** é editar e unir (ligar) os módulos de programa montados separadamente, resolvendo referências simbólicas entre eles.
  - A função do relocador é transformar as referências relocáveis à memória em referências absolutas corrigindo os endereços de acordo com a região de memória onde o módulo for carregado

Programação de Sistemas (continuação)

- Os temas tratados, tanto no nível conceitual como no nível experimental, permitem visualizar o processo de desenvolvimento de programas como uma evolução natural dos experimentos.
- É possível melhorar ainda mais, para o programador, a produtividade de escrita de programas se ele dispuser de linguagens com o nível mais alto possível de abstração, e que sejam mais independentes possível da máquina hospedeira utilizada.

Programação de Sistemas – Visão Geral

- Neste ponto de evolução, podem ser visualizadas a programação de sistemas e as interfaces com outros programas básicos de sistema, como é o caso dos compiladores e sistemas operacionais.
- Uma **linguagem de alto nível** qualquer (como, por exemplo, Java) permite que o programador tenha preocupação apenas com o problema e com as abstrações que melhor resolvam o problema.
  - Um compilador que traduz um texto escrito em linguagem de alto nível para alguma forma intermediária
  - Alternativamente, um interpretador pode analisar e executar diretamente as operações especificadas no programa-fonte
  - Em ambos os casos, no momento da execução do programa, um ambiente de execução provê, na máquina hospedeira, um conjunto de funcionalidades que completam o programa-objeto gerado pelo compilador, ou então são diretamente acionadas pelo interpretador.

Processo de Execução de Linguagens de Alto Nível

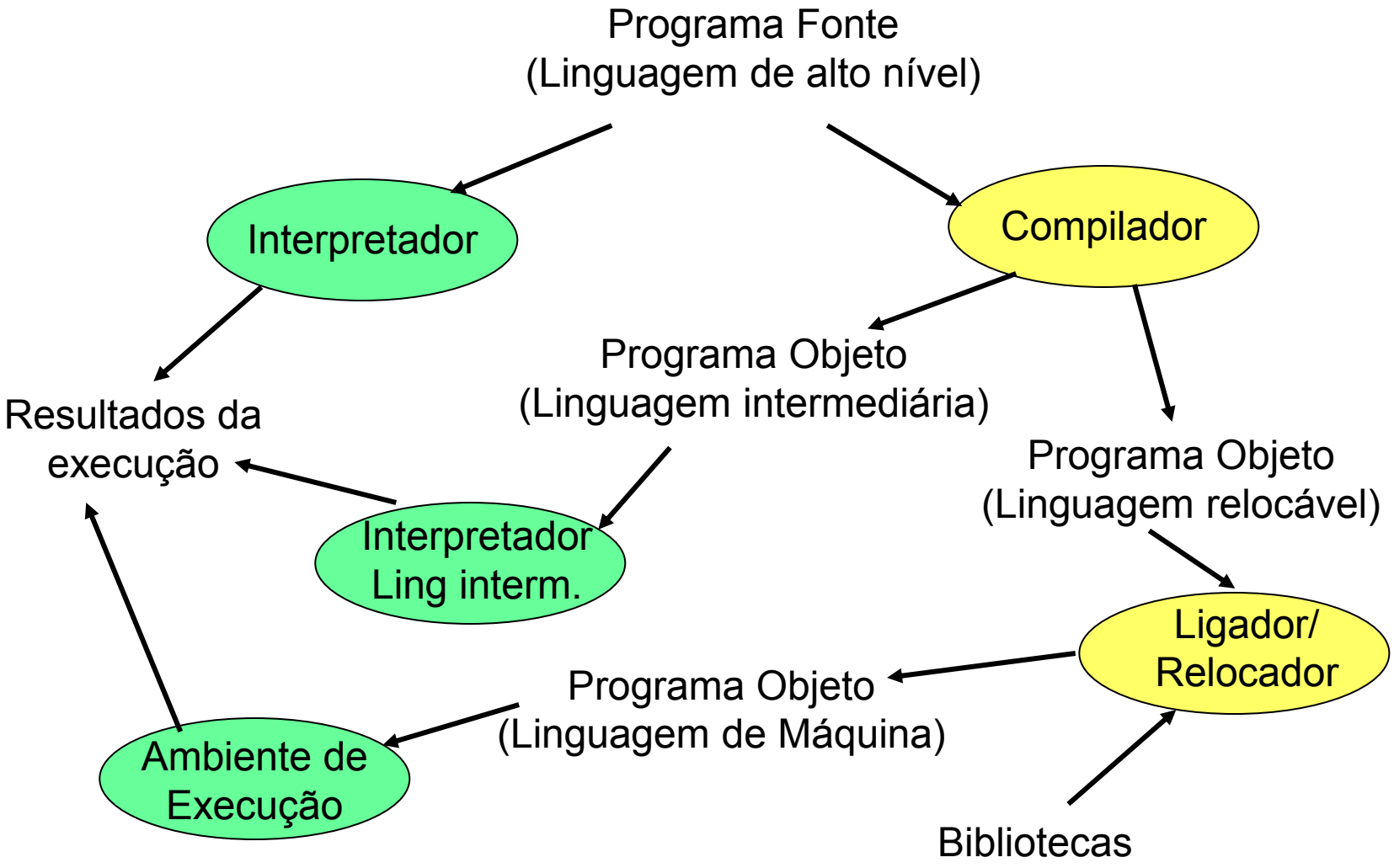

Diagrama do Sistema Desenvolvido na Disciplina

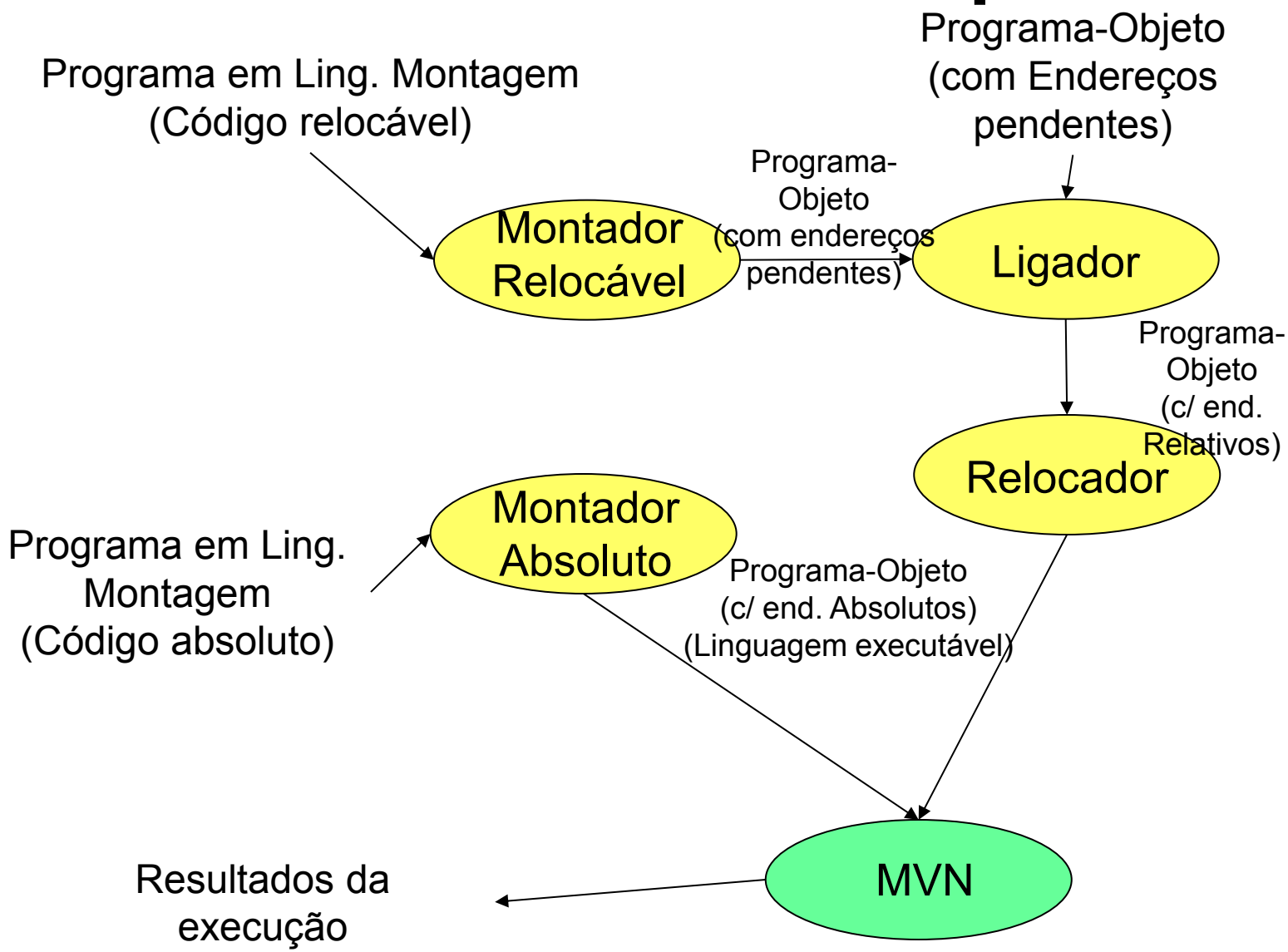

Tabela de mnemônicos para a MVN (de 2 caracteres)

Operação **0** **Jump** Mnemônico **JP**	Operação **1** Jump if **Zero** Mnemônico **JZ**	Operação **2** Jump if **Negative** Mnemônico **JN**	Operação **3** Load **Value** Mnemônico **LV**
Operação **4** **Add** Mnemônico **+**	Operação **5** **Subtract** Mnemônico **–**	Operação **6** **Multiply** Mnemônico *****	Operação **7** **Divide** Mnemônico **/**
Operação **8** **Load** Mnemônico **LD**	Operação **9** Move to **Memory** Mnemônico **MM**	Operação **A** **Subroutine** Call Mnemônico **SC**	Operação **B** **Return** from Sub. Mnemônico **RS**
Operação **C** **Halt** Machine Mnemônico **HM**	Operação **D** **Get** Data Mnemônico **GD**	Operação **E** **Put** Data Mnemônico **PD**	Operação **F** **Operating System** Mnemônico **OS**

Tabela de caracteres ASCII (7 bits. Ex: "K" = 4b)

	0	1	2	3	4	5	6	7
0	NUL		SP	0	@	P	`	p
1			!	1	A	Q	a	q
2			"	2	B	R	b	r
3			#	3	C	S	c	s
4			$	4	D	T	d	t
5			%	5	E	U	e	u
6			&	6	F	V	f	v
7	BEL		'	7	G	W	g	w
8			(	8	H	X	h	x
9			)	9	I	Y	i	y
a	LF		*	:	J	Z	j	z
b		ESC	+	;	K	[	k	{
c			,	<	L	\	l	|
d	CR		-	=	M	]	m	}
e			.	>	N	^	n	~
f			/	?	O	_	o	DEL

Bibliografia (Programação de Sistemas)

Relíquias Preciosas

Barron, D. W. ***Assemblers and Loaders*** (3rd. ed.) MacDonald/Elsevier, 1978

Beck, L. L. ***System Software - An Introduction to Systems Programming*** Addison-Wesley, 1996

Calingaert, P. ***Assemblers, Compilers and Program Translation*** Computer Science Press, 1979

Donovan, J. J. ***Systems Programming*** McGraw-Hill, 1972

Duncan, F.G. ***Microprocessor Programming and Software Development*** Prentice Hall, 1979.

Freeman, P. ***Software System Principles*** SRA, 1975

Gear, C. W. ***Computer Organization and Programming (3rd. ed.)*** McGraw-Hill, 1980

Graham, R. M. ***Principles of Systems Programming*** John Wiley & Sons, 1975

Gust, P. ***Introduction to Machine and Assembly Language Programming*** Prentice Hall, 1985

Maginnis, J. B. ***Elements of Compiler Construction*** Appleton-Century-Crofts, Meredith Co., 1972

Presser, L. and White, J. R. ***Linkers and Loaders*** ACM Comp. Surveys, vol. 4, n. 3, pp. 149-168, 1972

Rosen, S. (ed.) ***Programming Systems and Languages*** McGraw-Hill, 1967

Tseng, V. (ed.) ***Microprocessor Development and Development Systems*** McGraw-Hill, 1982

Ullman, J. D. ***Fundamental Concepts of Programming Systems*** Addison-Wesley, 1976

Wegner, P. ***Progr. Languages, Inf. Structures and Machine Organization*** McGraw-Hill, 1968.

Welsh, J. and McKeag, M. ***Structured System Programming*** Prentice-Hall,1980

Referências Bibliográficas

Bryant R. E. and O'Hallaron, D. R. ***Computer Systems: A Programmer's Perspective***, 2010.

DONOVAN, J. ***Systems Programming***, 1972.

Leitura complementar:

UM SIMULADOR-INTERPRETADOR PARA A LINGUAGEM DE MÁQUINA DO PATINHO FEIO.

(João José Neto, Aspectos do Projeto de Software de um Minicomputador, Dissertação de Mestrado, EPUSP, S. Paulo, 1975, cap.3)

Transparências extraídas e alteradas de:

José Neto, J., Sichman, J. S., Silva, P.S.M., Rocha, R.L.A. ***Material didático da disciplina PCS 2024 – Laboratório de Fundamentos da Engenharia de Computação***, PCS/EPUSP, São Paulo, SP. 2005-2015.