PCS3616

Programação de Sistemas

(Sistemas de Programação)

Semana 7, Aula 12

Introdução

Programação em linguagem de máquina E/S

Escola Politécnica da Universidade de São Paulo

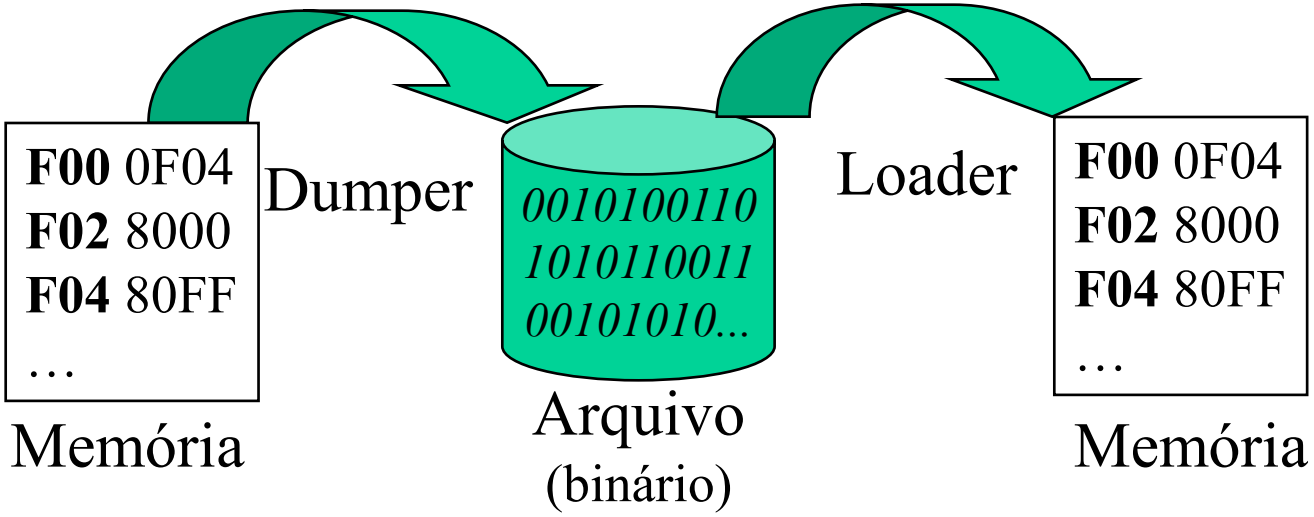
Roteiro

1. *Dumper* e *Loader* binários.
2. Projeto de um ***dumper*** e de um ***loader*** em linguagem de máquina binária para a MVN.

Dumper/Loader Binários (1)

Pretende-se implementar (**na linguagem da máquina de von Neumann**) dois programas:

- **Dumper**: destinado a memorizar em arquivo uma imagem do conteúdo da memória principal da Máquina de von Neumann;
- **Loader**: destinado a restaurar de um arquivo um conteúdo da memória principal da Máquina de von Neumann;



Dumper/Loader Binários (2)

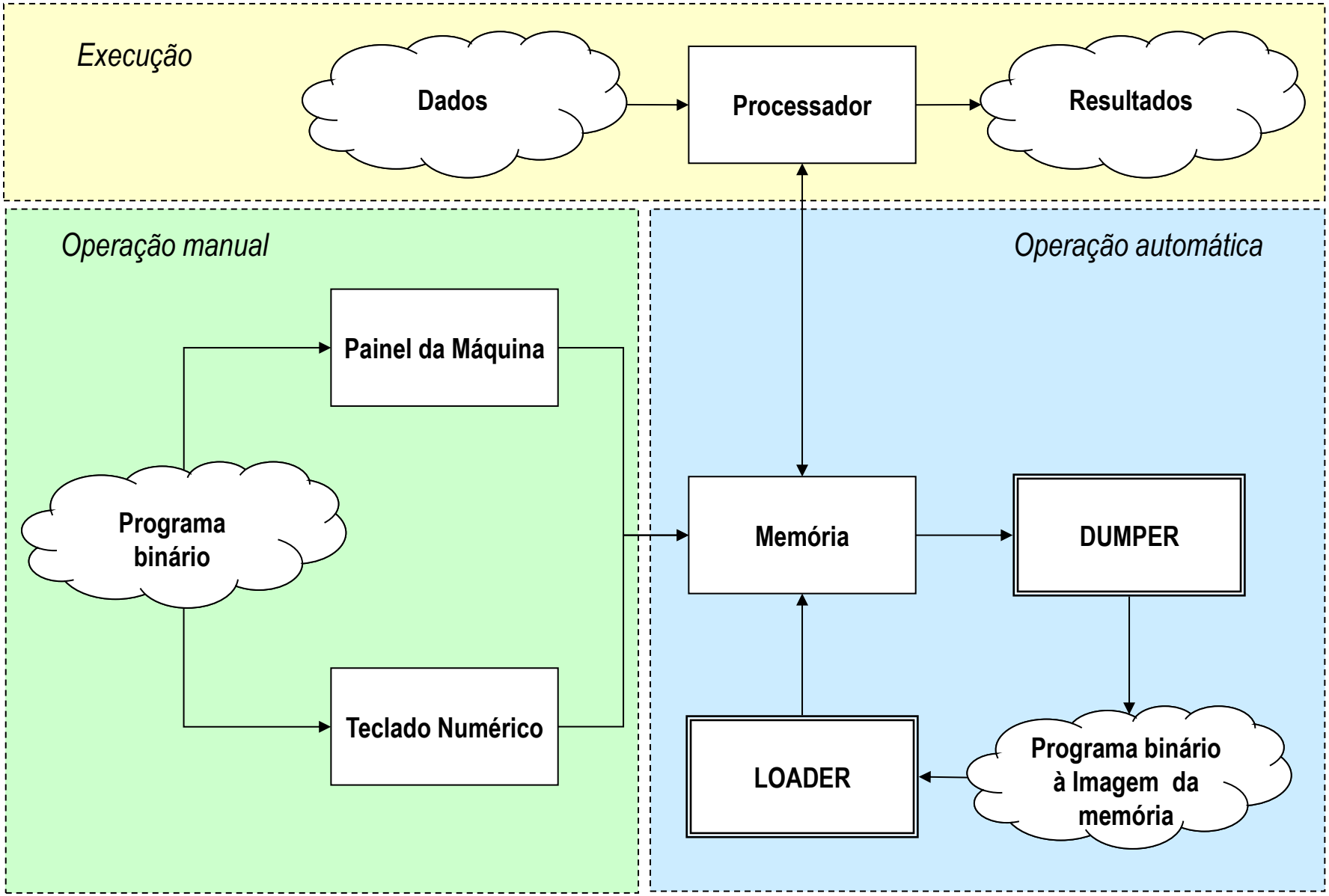
O arquivo binário deverá apresentar, em seu formato final, uma **sequência de blocos**, cada qual contendo os seguintes elementos (em ordem de importância):

- **imagem da memória** – uma cópia dos conteúdos de todas as posições de memória em que estamos interessados;
- **endereço inicial –** o endereço a partir do qual a imagem da memória foi copiada para o arquivo;
- **comprimento** – o tamanho da imagem da memória compreendidos no bloco, a partir do endereço inicial estipulado;
- **redundância** – dois ou mais bytes resultantes de uma função aplicada ao conjunto dos bytes contidos no bloco. O objetivo desses bytes é propiciar uma futura verificação de consistência.
  - Em versões menos sofisticadas, constam de um byte apenas, obtido pela simples soma de todos os bytes do bloco. Neste caso denomina-se “Checksum”.
  - Nos casos de maior responsabilidade, são obtidos aplicando-se a essas informações um polinômio, e guardando-se o resultado em diversos bytes. Neste caso, é muitas vezes denominado CRC (“Cyclic Redundancy Check”).

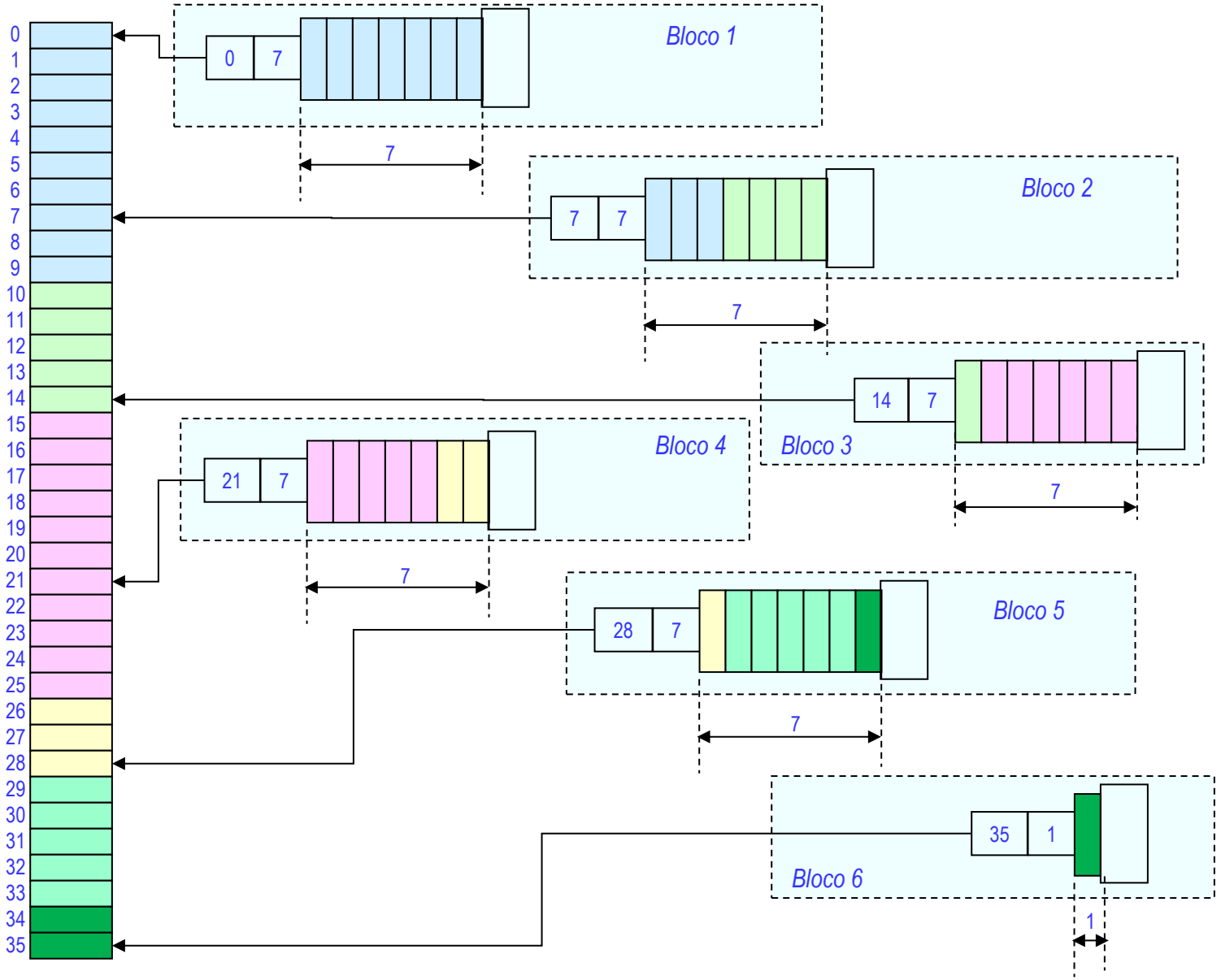
Dumper/Loader Binários (3)

- A memória do computador não retém para sempre todos os programas nela executados;
- A quantidade de tais programas é muito grande;
- É inviável inseri-los manualmente a cada execução;
- Uma vez depurados, convém guardar os programas em algum meio externo, para uso futuro;
- Assim, desejando-se executar um programa, basta copiá-lo para a memória e utilizá-lo;
- Isso pode ser feito através de outro programa.

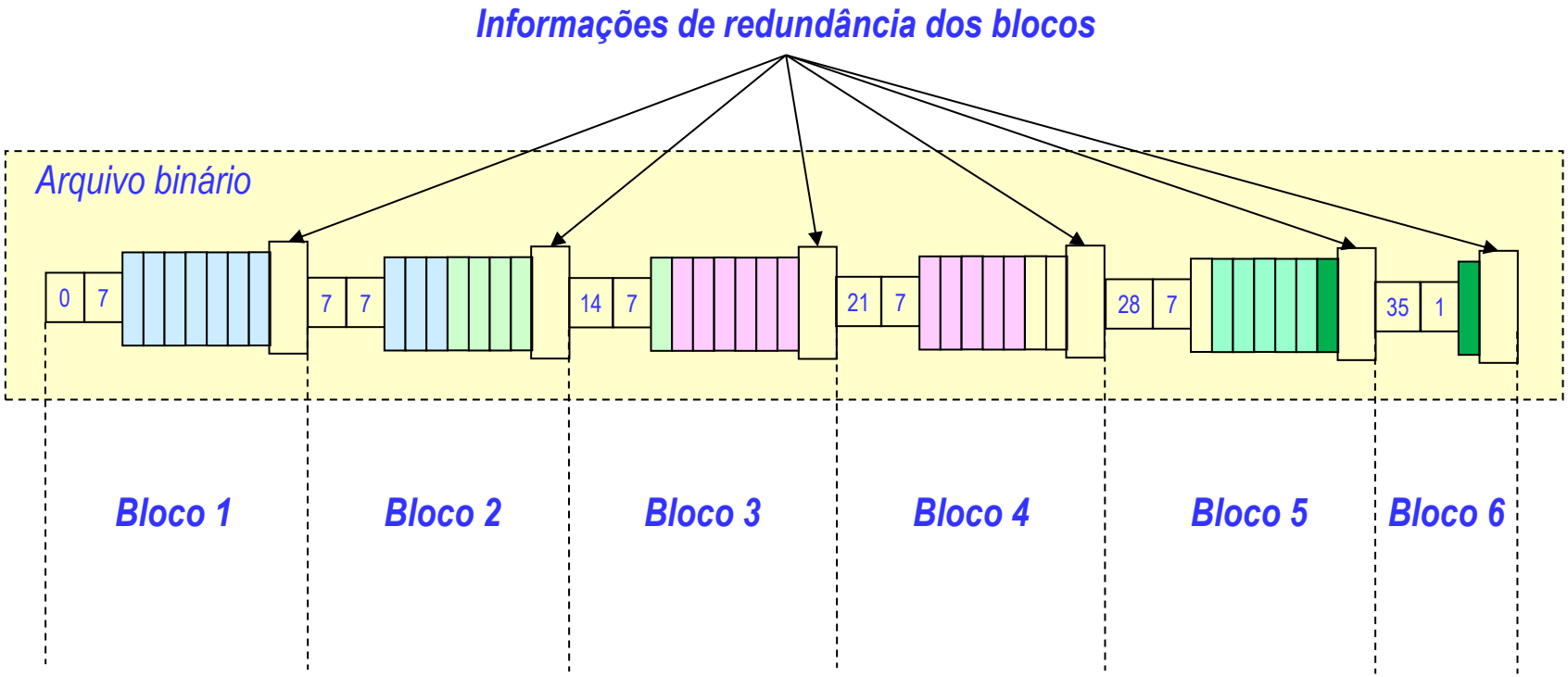
Dumper/Loader Binários: Visão Geral



Dumper/Loader Binários: Formato (1)



Dumper/Loader Binários: Formato (2)



As informações de redundância se calculam a partir dos dados que compõem o bloco, incluindo o endereço inicial, comprimento e conteúdo da memória.

Dumper/Loader Binários para a MVN

- Em cada bloco:
  - O endereço inicial e o comprimento devem ter 2 bytes cada (uma palavra);
  - Por simplicidade, sugere-se o utilizar o checksum como informação de redundância dos blocos, utilizando 2 bytes. Portanto, deve-se ter um tratamento adequado para o caso de a soma ultrapassar o valor máximo válido permitido;
  - A imagem da memória deve ser representada em palavras contíguas (2 bytes).

Dumper Binário para MVN: Operação Básica

1. Escolher os limites de memória do dump desejado;
2. Determinar a quantidade de dados da memória a copiar para o arquivo;
3. Escolher o número máximo de palavras em cada bloco;
4. Para cada bloco a ser gravado:
   4.1. Determinar os limites do bloco;
   4.2. Gravar o endereço inicial do bloco;
   4.3. Gravar a quantidade de palavras no bloco;
   4.4. Ler na memória os dados a copiar e gravá-los no bloco;
   4.5. Calcular os bytes de redundância (checksum) do bloco e incluir no mesmo.

Loader Binário para MVN: Operação Básica

Para cada bloco do arquivo binário lido:

- Ler o endereço inicial do bloco;
- Ler o número de palavras do bloco;
- Ler no arquivo todos os dados do bloco, e gravá-los na memória;
- Aplicar a função para calcular os bytes de redundância a partir dos dados transferidos;
- Comparar o checksum calculado com o checksum lido do arquivo;
- Emitir mensagem de erro em caso de discrepância e parar.

Dumper/Loader Binários para MVN: Variáveis

- Os dois programas têm lógica similar, e podem utilizar as mesmas variáveis em sua operação. Algumas delas podem ser:
  - INICIAL – endereço inicial do load/dump (alinhado na palavra)
  - FINAL – endereço final do load/dump (alinhado na palavra)
  - NWORDS – número de palavras do load/dump
  - COMPRIMENTO – número de palavras da imagem da memória compreendidos no bloco.
  - FALTAM – número de palavras restantes
  - ...

Implementação

Formato do arquivo:

O arquivo binário à imagem da memória deverá apresentar, em seu formato final, uma **sequência de blocos**, cada qual contendo a seguinte sequência:

- **endereço inicial** – dois bytes (uma palavra), representando o endereço a partir do qual a imagem da memória deve ser (ou foi) copiada para o arquivo;
- **comprimento** – número de palavras compreendidas no bloco, a partir do endereço inicial estipulado, inclusive. Como seguiremos uma certa tradição de estabelecer o tamanho do bloco em 128 bytes, o comprimento deverá ser inferior ou igual a 128 bytes = 64 palavras (calculem o valor adequado).
- **imagem da memória** – uma cópia dos conteúdos de todas as posições de memória em que estamos interessados (lembrar que os endereços estão alinhados em palavras).
- **redundância** – uma palavra contendo os 16 bits menos significativos do checksum de **todos os dados do bloco** (endereço, comprimento e imagem da memória).

Algumas dicas para o Dumper

1. Comece desenvolvendo um *dumper* rudimentar para gravar em arquivo a imagem de toda a memória, sem incluir endereço inicial, número de bytes nem *checksum*. Verifique o conteúdo do arquivo com um programa aplicativo que permita visualizar código binário na representação hexadecimal;
2. Em seguida, inclua no *dumper* a entrada dos limites de *dump*, e gere o arquivo desconsiderando o *checksum*. Os limites podem estar definidos no programa principal do *dumper*;
3. Finalmente, gere o arquivo no formato definitivo.

Obs.: O código do **Dumper** deve utilizar apenas a memória que está dentro do intervalo estabelecido, isto é, de 0x700 a 0xAFF.

Algumas dicas para o Loader

1. Comece desenvolvendo um *loader* rudimentar para recuperar na memória a imagem de um único bloco. Verifique o conteúdo do arquivo com um programa aplicativo que permita visualizar código binário na representação hexadecimal;
2. Em seguida, inclua no *loader* o cálculo do valor do checksum;
3. Implemente o tratamento de erro quando houver uma discrepância (emitir mensagem de erro);
4. Finalmente, recupere a partir do arquivo toda a região de memória pertinente.

Obs.: O código do **Loader** deve utilizar apenas a memória que está dentro do intervalo estabelecido, isto é, de 0xB00 a 0xFFF.

Outras dicas

- Para facilitar o desenvolvimento, é possível inicializar os dispositivos da MVN através de arquivo.
- Para tanto, crie um arquivo "**disp.lst**" na raiz do projeto NetBeans ou na pasta do arquivo *.JAR.
- Cada linha deste arquivo indica um dispositivo a ser adicionado no gerenciador.

- Exemplo de arquivo:

```
0 0 mvn.dispositivo.Teclado
1 0 mvn.dispositivo.Monitor
3 1 mvn.dispositivo.Disco dump.dat e
```

Outras dicas

- Sintaxe:

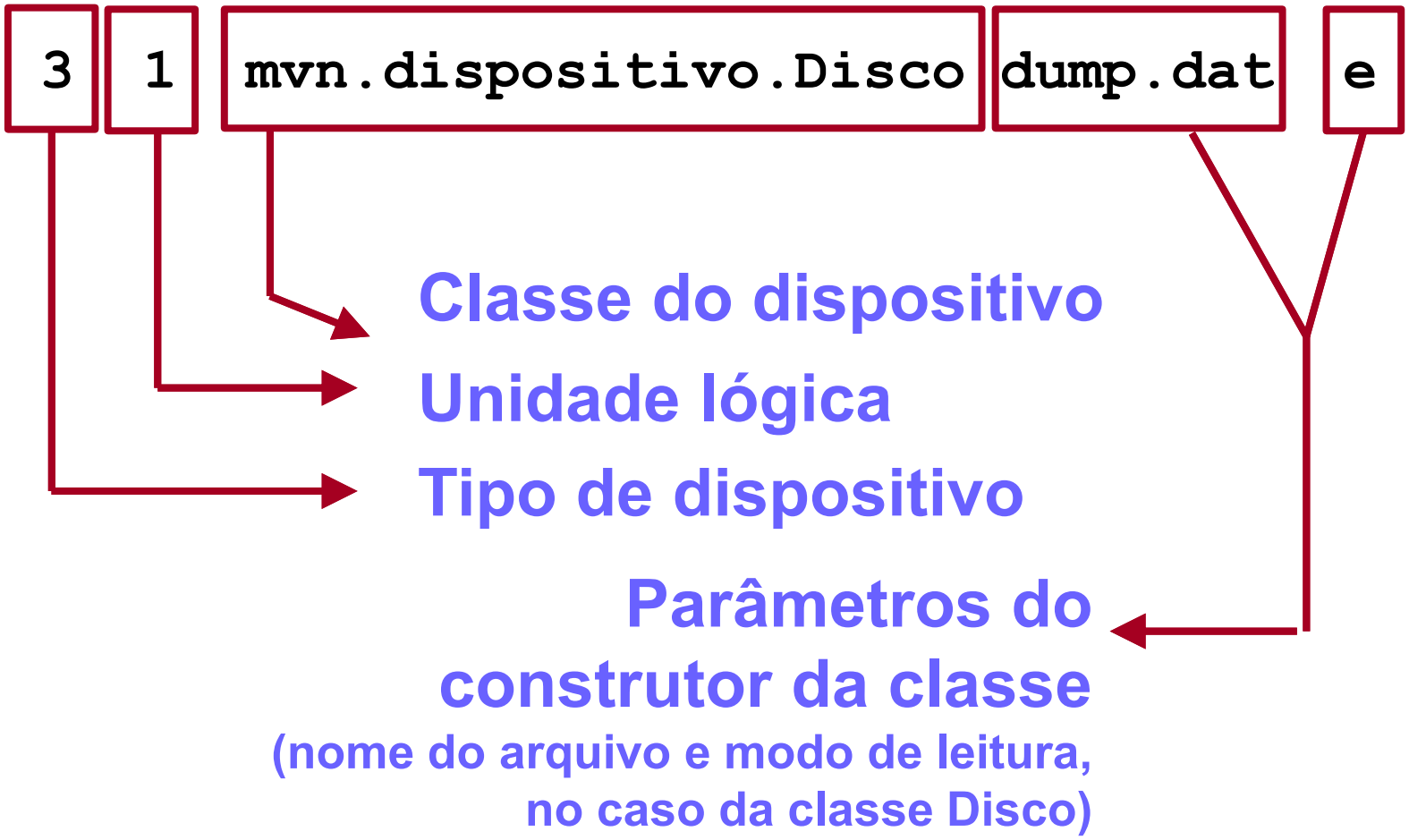


Tabela de mnemônicos para a MVN (de 2 caracteres)

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.