PCS 2302/2024
Laboratório de Fundamentos da Eng.de Computação

PCS-2302 / PCS-2024
Lab. de Fundamentos de Eng. de Computação

Aula 07

Construção de um Dumper para o simulador MVN

Professores:
Marcos A. Simplício Junior
Anna Helena Reali Costa

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Roteiro

1. *Dumper* binário
2. Projeto de um *Dumper* para a MVN
3. Parte Experimental
   - Implementação de um *Dumper* para a MVN, usando a linguagem simbólica do montador relocável.

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

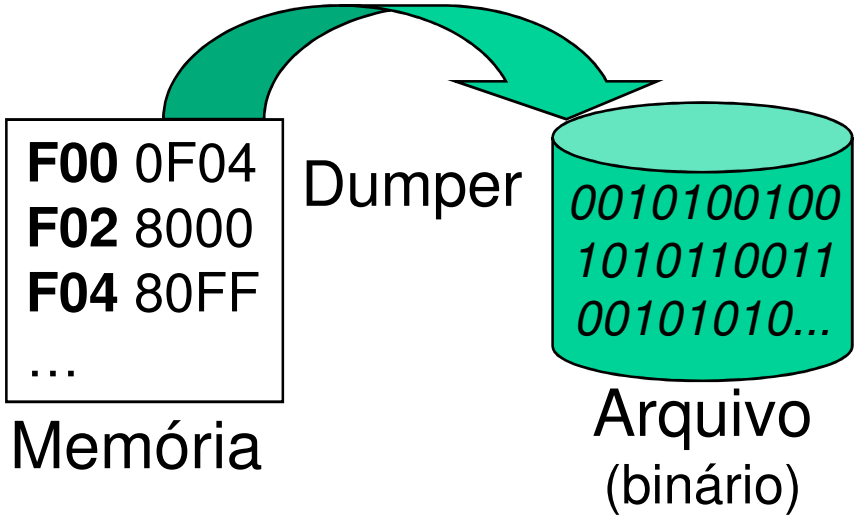
Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Dumper Binário (1)

Pretende-se implementar o seguinte programa que será incorporado à biblioteca elementar da MVN:

- **Dumper:** destinado a armazenar em arquivo uma imagem binária do conteúdo da memória principal da MVN.



Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Dumper Binário (2)

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

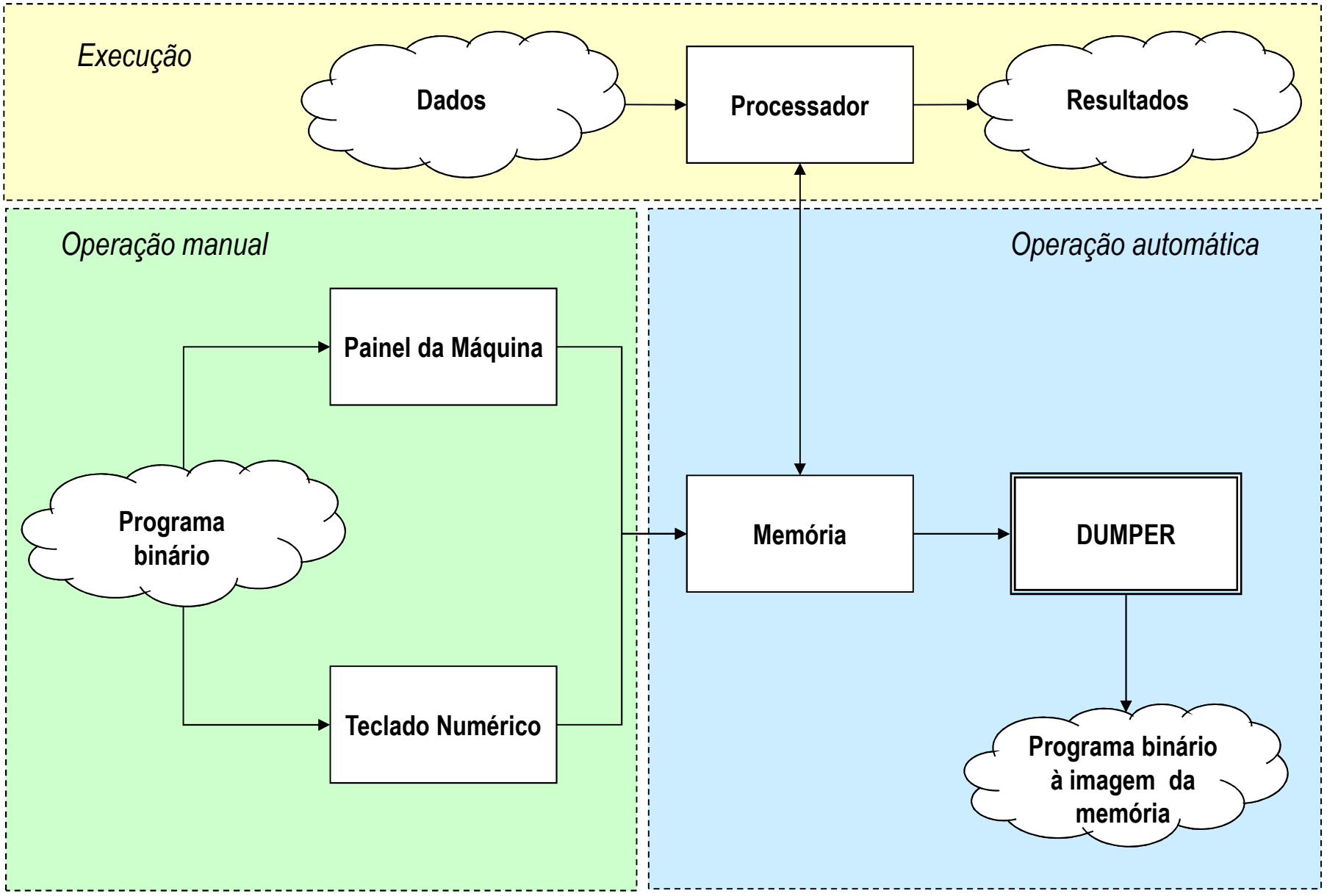
- **imagem da memória** – uma cópia do conteúdo de todas as posições de memória em que estamos interessados;
- **endereço inicial** – o endereço a partir do qual a imagem da memória foi copiada para o arquivo;
- **comprimento** – o tamanho da imagem da memória compreendido no bloco, a partir do endereço 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, utiliza-se apenas um ou dois bytes, obtidos pela simples soma de todos os bytes do bloco. Neste caso denomina-se “Checksum”.
  - Nos casos de maior responsabilidade, aplica-se a essas informações um polinômio, guardando-se o resultado em diversos bytes. Neste caso, é muitas vezes denominado CRC (“Cyclic Redundancy Check”).

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

Dumper Binário: Visão Geral



PCS 2302/2024
Laboratório de Fundamentos da Eng.de Computação

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Dumper Binário – Observações

- *Dumper*: normalmente utilizado para fins de depuração. Exemplo: "*core dump*" permite verificação do estado da memória em certo ponto da execução de um programa.

- Estratégia semelhante usada por memória virtual: dados não sendo usados são enviados para o disco temporariamente, permitindo que outras aplicações usem a memória física (swap de memória)

- Na disciplina, o *dumper* será essencialmente utilizado para gerar um arquivo com a imagem binária de uma região especificada da memória, para ser utilizado por outros programas de sistema.

Aula 8:

Construção de um Dumper para o simulador MVN

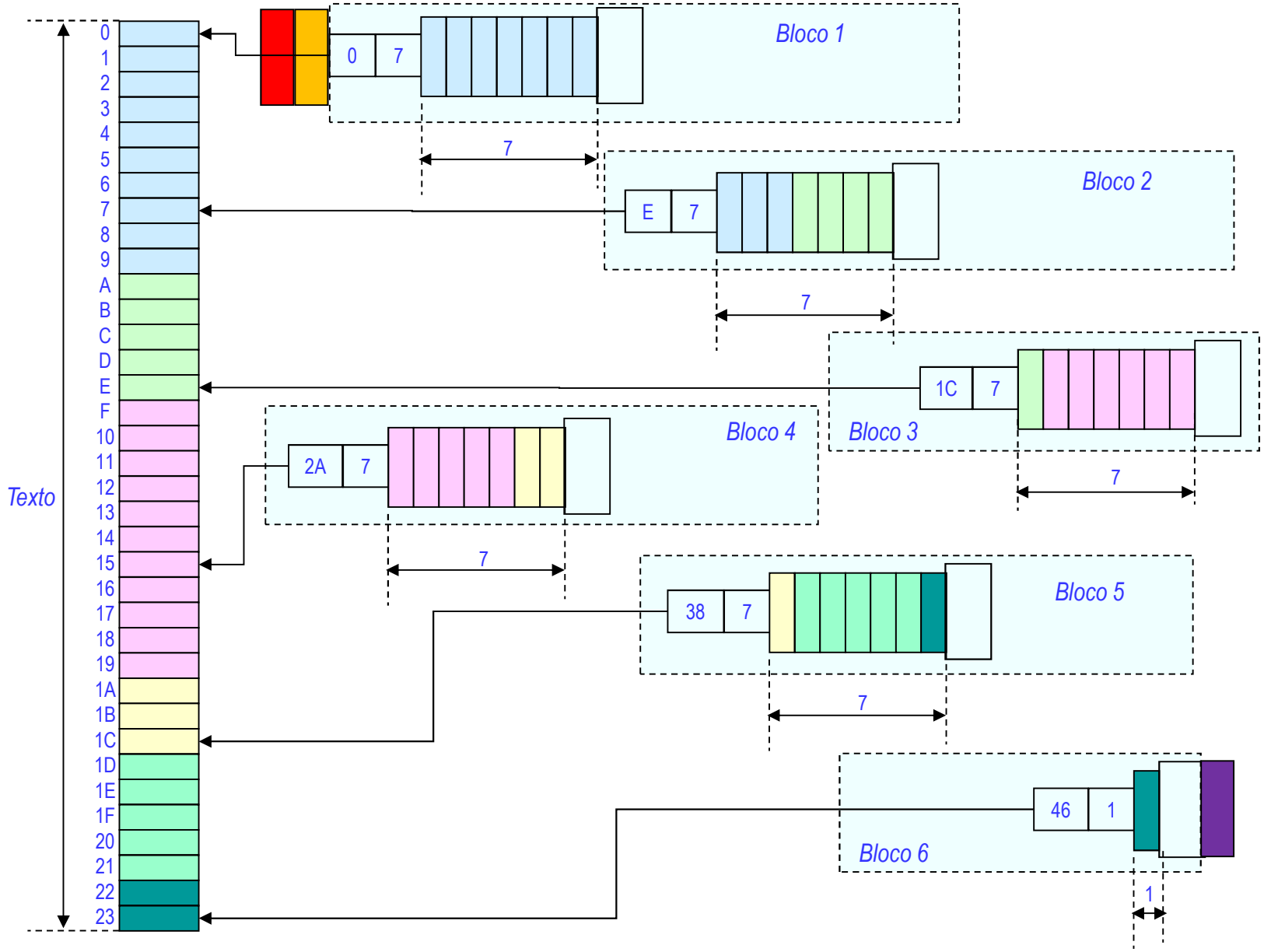
Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Dumper Binário: Formato (1)
Imagem da Memória



PCS 2302/2024
Laboratório de Fundamentos da Eng.de Computação

Aula 8:

Construção de um Dumper para o simulador MVN

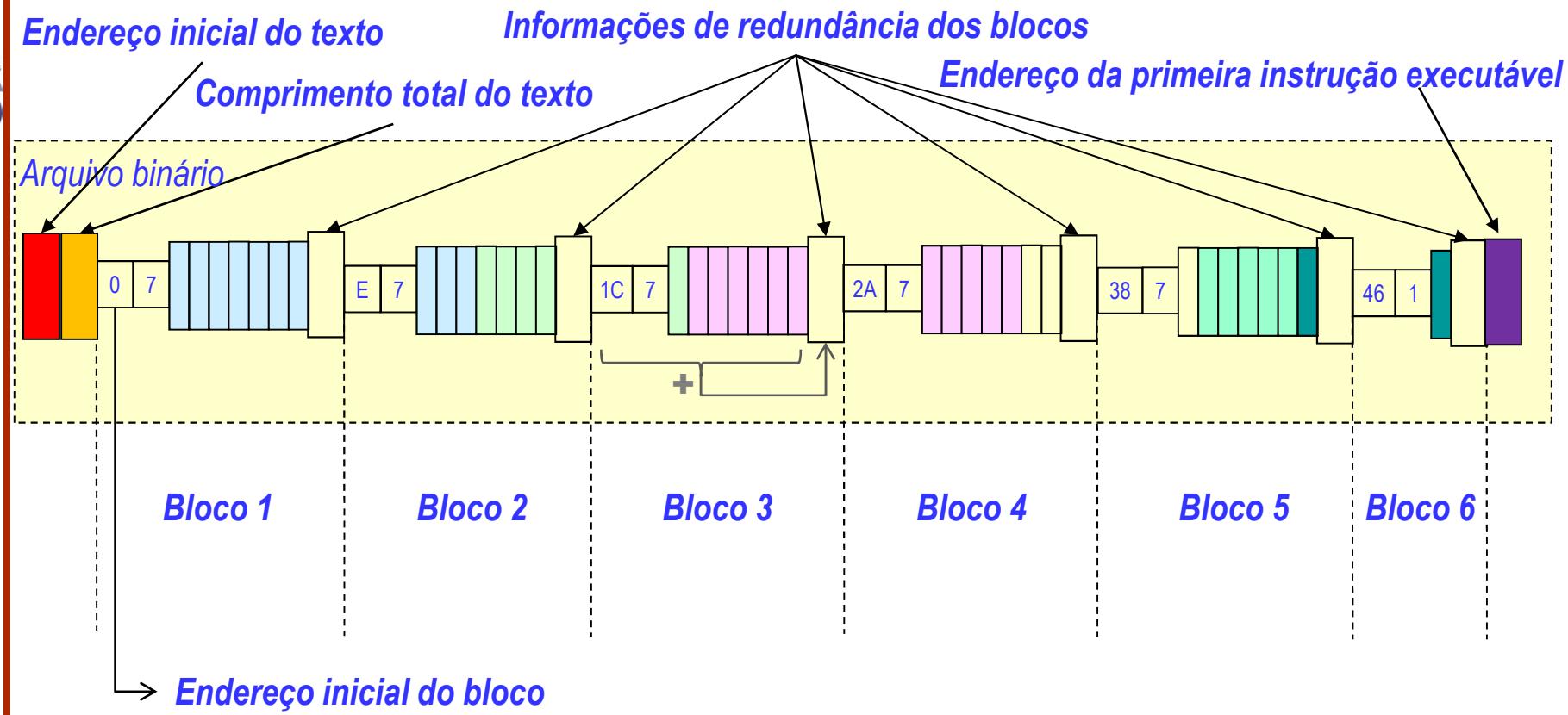
Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Dumper Binário: Formato (2)
Formato Completo



Na disciplina, o formato do arquivo gerado pelo Dumper define a última palavra (word) do arquivo como responsável por conter o "endereço da primeira instrução executável" do programa, cuja imagem foi gravada. Caso a imagem não seja a de um programa executável, convenciona-se que o valor nessa posição seja 0xFFFF.

A informação de redundância é calculada a partir dos dados que compõem o bloco, incluindo o endereço inicial, comprimento e conteúdo da memória.

PCS 2302/2024
Laboratório de Fundamentos da Eng.de Computação

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Dumper Binário para a MVN

- No início do arquivo
  - O endereço inicial do texto a ser carregado e o comprimento total do texto devem ter 2 bytes cada (uma **word**);
- Em cada bloco:
  - O endereço inicial do bloco e o comprimento do bloco devem ter 2 bytes cada (uma **word**);
  - Por simplicidade, sugere-se utilizar o checksum como informação de redundância dos blocos, utilizando 2 bytes. Ignora-se aqui o caso em que a soma ultrapassa o valor máximo válido permitido para uma **word**, ou seja, a **word** conterá os 16 bits menos significativos do checksum;
  - A imagem da memória deve ser representada em **words** contíguas (2 bytes).

Autores:

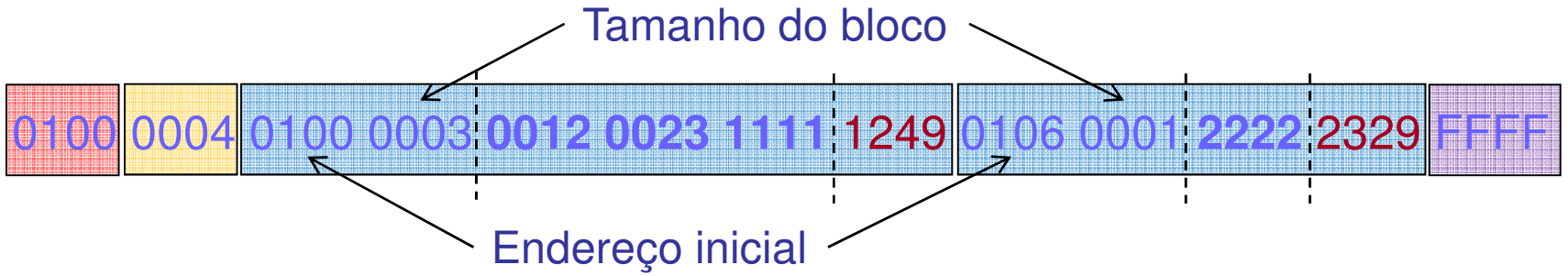
Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Dumper Binário para a MVN

- Ao final do arquivo:
  - O campo de 2 bytes no final do arquivo ("endereço de sua primeira instrução executável") deve conter um valor passado como parâmetro pelo programa principal.
- Exemplo:
  - Dump de 4 words a partir da posição 0100, com blocos de 3 words, primeira instrução executável = FFFF, e supondo conteúdo da memória: [0100] **0012 0023 1111 2222**



Checksum: 0100+0003+0012+0023+1111 = 1249

0106+0001+2222 = 2329

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

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 **words** em cada bloco;
4. No início do arquivo: gravar o endereço inicial do texto e o comprimento total que ele ocupa na memória;
5. Para cada bloco a ser gravado:
   - 5.1. Determinar os limites do bloco;
   - 5.2. Gravar o endereço inicial do bloco;
   - 5.3. Gravar a quantidade de **words** do bloco;
   - 5.4. Ler na memória os dados a copiar e gravá-los no bloco;
   - 5.5. Calcular a redundância (checksum) do bloco e incluir no mesmo.
6. Ao final do arquivo: caso seja a imagem de um programa, gravar o endereço da primeira instrução executável; caso contrário, gravar o valor 0xFFFF.

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

Exercício 0 (sem entrega)

- Familiarizando-se com o comando de escrita da MVN:
  - Crie uma unidade de disco com identificador 0 na MVN, (1) usando o comando “s” ou (2) editando o arquivo “disp.lst” fornecido, que deve estar na mesma pasta que a MVN
  - Monte e execute o seguinte código:

```
        @     /0000  ; endereço absoluto
        JP    INI    ; vai para início do programa
VAL     K     /1234  ; valor a ser escrito no disco
INI     LD    VAL    ; carrega valor no acumulador
        PD    /300   ; escreve valor do acumulador no disco
                     ; cujo ID é 00 (operação de “append”)
END     HM    END    ; fim
        #     INI
```

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Exercícios 1 a 3 (Obrigatórios)

Cada grupo deverá projetar, implementar e testar um ***Dumper*** binário, na linguagem de montagem da MVN, de modo incremental, como descrito mais adiante.

- O *Dumper* deve seguir a estrutura de sub-rotina;
- Você pode usar o arquivo "TYGXXA07E03_main.mvn" como seu main para testes, adaptando **os valores** dos parâmetros conforme necessário
  - **Atenção**: não faça alterações na região indicada
- Parâmetros de entrada da sub-rotina:
  - Endereço inicial da memória: DUMP_INI
  - Comprimento total da imagem (quantidade de **words** no dump): DUMP_TAM
  - Comprimento do bloco (quant. máx. de **words** no bloco): DUMP_BL
  - Endereço da primeira instrução executável: DUMP_EXE
  - Número da Unidade Lógica (LU) do tipo **Disco** (0x3): DUMP_UL

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

Exercício 1

1. Desenvolva, inicialmente, uma sub-rotina *dumper* rudimentar para gravar em arquivo a imagem binária de toda a memória, sem incluir o endereço inicial, o comprimento do *dump* nem o *checksum*. Verifique o conteúdo do arquivo com um programa aplicativo que permita visualizar código binário na representação hexadecimal (ex.: Sublime, ou Neo Hex Editor). **(Obrigatório)**

Nomes dos Arquivos: TYGXXA07E01_main.asm
TYGXXA07E01_dumper.asm

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Exercício 2

2. Estender o programa desenvolvido anteriormente, incluindo no *dumper* os seus parâmetros de entrada:
   - **Limites** de *dump*: endereço inicial (DUMP_INI) e tamanho total da imagem (DUMP_TAM)
   - Número da **unidade lógica** (DUMP_UL)
- Nesta versão, ainda gere o arquivo desconsiderando o *checksum*. **(Obrigatório)**

Nomes dos Arquivos: TYGXXA07E02_main.asm
TYGXXA07E02_dumper.asm

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Exercício 3

3. Finalmente, gere o arquivo no formato definitivo, em blocos. Ele deve receber todos os parâmetros descritos:
   - Endereço inicial (DUMP_INI), tamanho total da imagem (DUMP_TAM) e número da unidade lógica (DUMP_UL)
   - Comprimento do bloco (DUMP_BL) e endereço da primeira instrução executável (DUMP_EXE)
- Escreva no disco os dados da memória no formato descrito em aula, considerando o **checksum de cada bloco**. **(Obrigatório)**

Nomes dos Arquivos: TYGXXA07E03_main.asm
TYGXXA07E03_dumper.asm

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

Exercício 3 (formato)

Formato do arquivo:

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

- **endereço inicial** – dois bytes (uma **word**), representando o endereço a partir do qual a imagem da memória deve ser (ou foi) copiada para o arquivo;
- **comprimento** – número de **words** 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 words.
- **imagem da memória** – uma cópia dos conteúdos de todas as posições de memória em que estamos interessados (lembrar que os endereços estão alinhados em **words**).
- **redundância** – uma **word** contendo os 16 bits menos significativos do checksum de **todos os dados do bloco** (endereço, comprimento e imagem da memória).

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Lista de Comandos

- **Para a execução do montador**
  - **java -cp MLR.jar montador.MvnAsm [<arquivo asm>]**
  - **Exemplo: java -cp MLR.jar montador.MvnAsm test.asm**
- **Para a execução do linker**
  - **java -cp MLR.jar linker.MvnLinker <arquivo-objeto1> <arquivo-objeto2> ... <arquivo-objetoN> -s <arquivo-saida>**
  - **Exemplo: java -cp MLR.jar linker.MvnLinker prog1.mvn prog2.mvn –s test.mvn**
  - Obs.: coloque a função main como primeiro argumento (isso facilita a execução, pois a primeira instrução do programa ligado será do main)
- **Para a execução do relocador**
  - **java -cp MLR.jar relocator.MvnRelocator <arquivo-objeto> <arquivo-saida> <base-relocação> <endereço-inicio-execução>**
  - **Exemplo: java -cp MLR.jar relocator.MvnRelocator test.mvn final.mvn 0000 000**
- **Para a execução da MVN**
  - **java -jar mvn.jar**
  - Obs.: Se houver problemas com caracteres especiais, use:
    - **java -Dfile.encoding=cp850 -jar mvn.jar**

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

Tabela de mnemônicos para as instruções da MVN (de 2 caracteres)

Operação **0** **Jump** Mnemônico **JP**	Operação **1** **J**ump if **Z**ero Mnemônico **JZ**	Operação **2** **J**ump if **N**egative Mnemônico **JN**	Operação **3** **L**oad **V**alue 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** **S**ubroutine **C**all Mnemônico **SC**	Operação **B** **R**eturn from **S**ub. Mnemônico **RS**
Operação **C** **H**alt **M**achine Mnemônico **HM**	Operação **D** **G**et **D**ata Mnemônico **GD**	Operação **E** **P**ut **D**ata Mnemônico **PD**	Operação **F** **O**perating **S**ystem Mnemônico **OS**

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

PCS 2302/2024
Laboratório de Fundamentos da Eng.de Computação

Aula 8:

Construção de um Dumper para o simulador MVN

Autores:

Anna H. R. Costa
Jaime S. Sichman
João José Neto
Paulo S. Muniz Silva
Ricardo L. A. Rocha

Reestruturação:
Paulo S. Muniz Silva

v.1.0 ago. 2012

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