




|   |   |
|---|---|
| <br><br><br>PCS 2302/2024<br>Laboratório de<br>Fundamentos da<br>Eng.de Computação | <p><b>PCS-2302 / PCS-2024</b></p> <p><b>Lab. de Fundamentos de Eng. de Computação</b></p>   |
|   | <p><b>Aula 08</b></p> <p><b>Construção de um Loader<br/>para o simulador MVN</b></p> <p><b>Professores:</b><br/>Marcos A. Simplício Junior<br/>Paulo Sergio Muniz Silva</p> |




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

1

|   |   |
|---|---|
| <br><br><br>PCS 2302/2024<br>Laboratório de<br>Fundamentos da<br>Eng.de Computação | <p><b>Roteiro</b></p>   |
|   | <ol style="list-style-type: none"><li>1. <i>Loader</i> binário</li><li>2. Projeto de um <i>Loader</i> para a MVN</li><li>3. Parte Experimental<ul style="list-style-type: none"><li>• Implementação de um <i>Loader</i> para a MVN, usando a linguagem simbólica do montador relocável.</li></ul></li></ol> |

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

2

USP



DEPARTAMENTO DE ENGENHARIA DE

COMPUTAÇÃO E SISTEMAS DIGITAIS

PCS

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

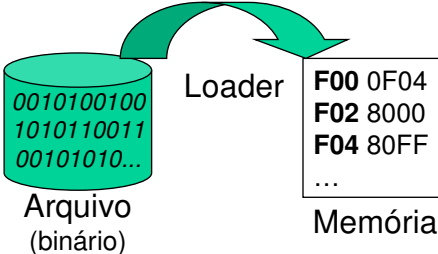
v.1.0 ago. 2012

3


Loader Binário (1)

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

•**Loader:** destinado a restaurar de um arquivo o conteúdo da memória principal da MVN;



USP



DEPARTAMENTO DE ENGENHARIA DE

COMPUTAÇÃO E SISTEMAS DIGITAIS

PCS

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

4

Loader Binário (2)

O formato do arquivo binário é o mesmo do *Dumper* da aula anterior. Recordando, ele é 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;

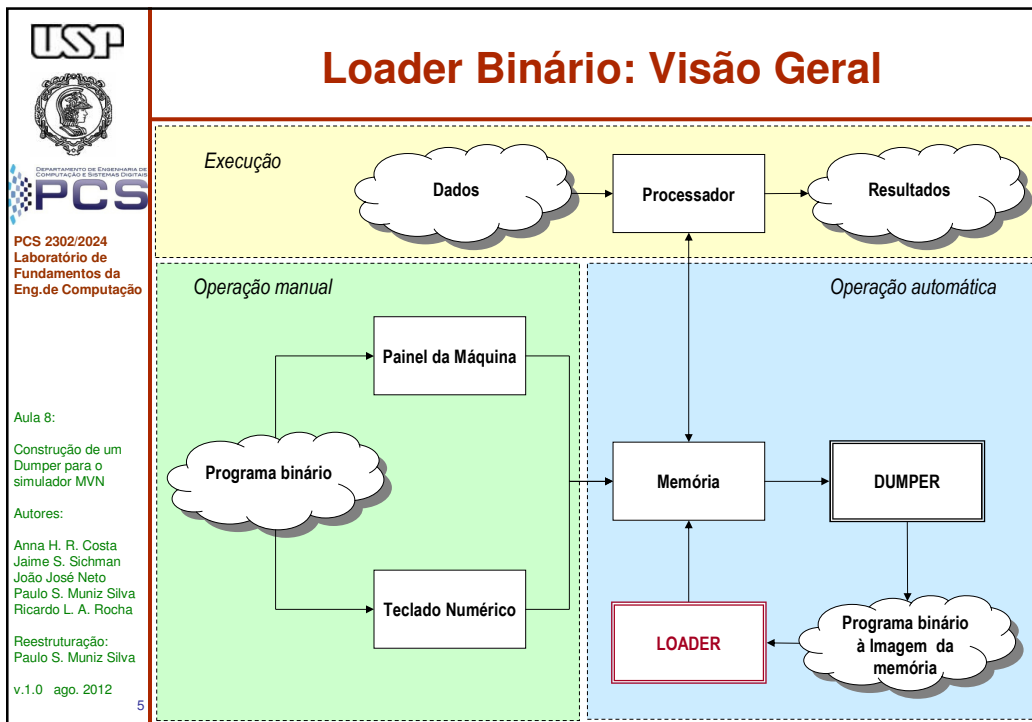
-**endereço inicial** – o endereço a partir do qual a imagem da memória foi copiada para o arquivo;

-**comprimento** – o tamanho da imagem da memória 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 a 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").



**USP**



**PCS**

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

Aula 8:  
Construção de um  
Dumper para o  
simulador MVN

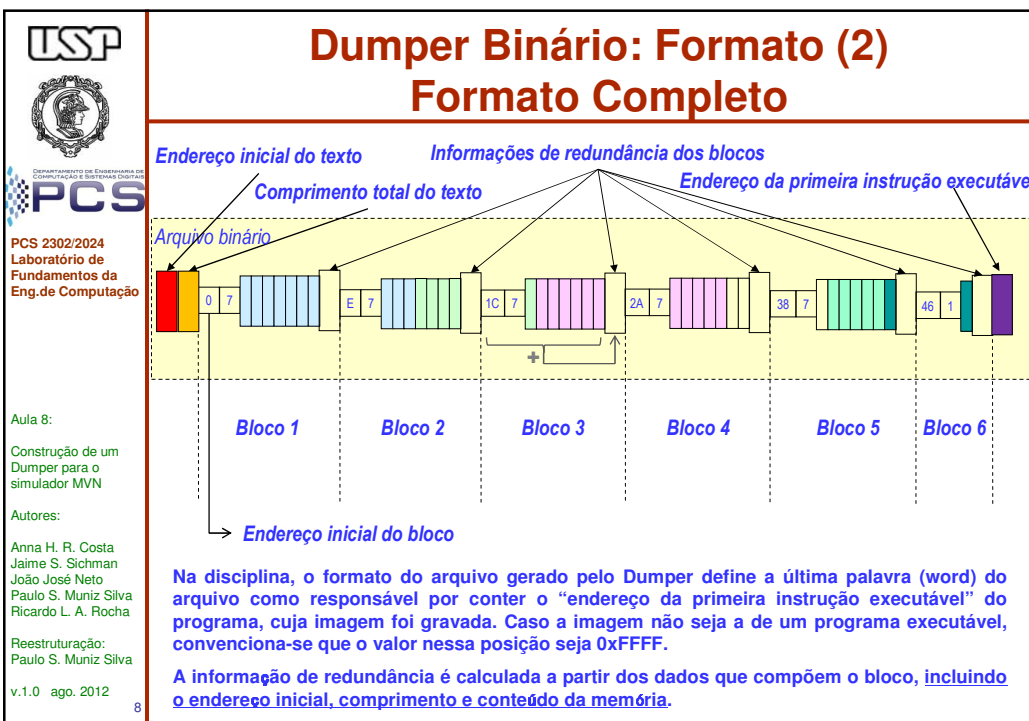
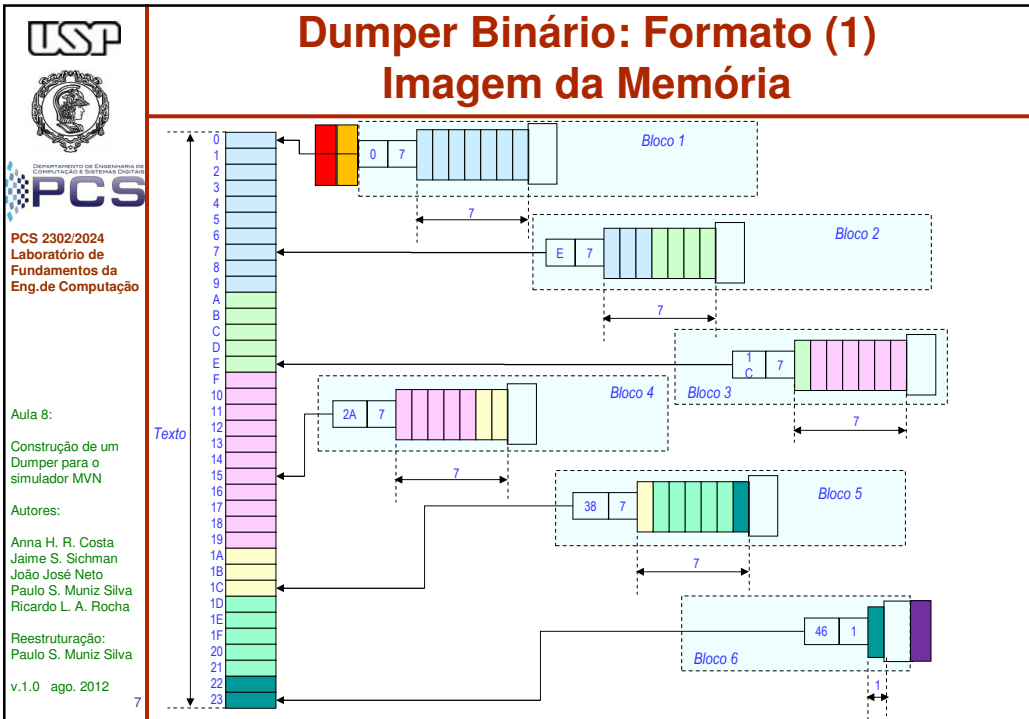
Autores:  
Anna H. R. Costa  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha



Reestruturação:  
Paulo S. Muniz Silva

v.1.0 ago. 2012

## Loader Binário – Observações

- O *loader* normalmente é utilizado para carregar programas executáveis. Para que o programa carregado possa ser executado, o *loader* deve ter a informação da primeira instrução executável do programa.
- Ele também pode ser utilizado para carregar uma imagem da memória que não seja um programa executável, por exemplo, dados, constantes, etc. utilizados por um ou mais programas.



PCS

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

## Loader Binário para a MVN

- Em suma, no arquivo a ser carregado:
  - 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 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).

PCS

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

Aula 8:  
Construção de um  
Dumper para o  
simulador MVN

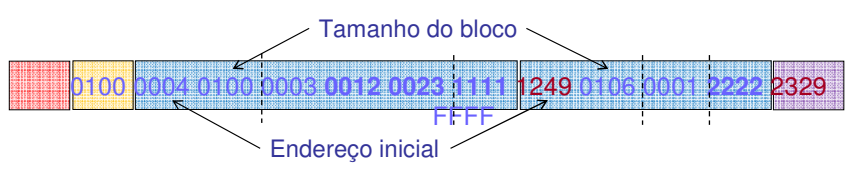
Autores:  
Anna H. R. Costa  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha

Reestruturação:  
Paulo S. Muniz Silva

v.1.0 ago. 2012



## Loader Binário para a MVN

- Ao final do arquivo:
  - O campo de 2 bytes no final do arquivo ("endereço de sua primeira instrução executável") deve ser carregado no acumulador.
- Exemplo:
  - Dump da 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~~

PCS

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

## Loader Binário para MVN: Operação Básica

1. Ler, no início do arquivo, o endereço inicial e o comprimento total da imagem do texto. Verificar se a imagem cabe na memória, emitindo uma mensagem de erro (**FFFE** no acumulador) e parando se não for o caso.
2. Para cada bloco do arquivo binário lido:
  - 2.1. Ler o endereço inicial do bloco;
  - 2.2. Ler o número de words do bloco;
  - 2.3. Ler no arquivo todos os dados do bloco e gravá-los na memória;
  - 2.4. Aplicar a função para calcular o checksum a partir dos dados transferidos, do endereço inicial e do número de words;
  - 2.5. Comparar o checksum calculado com o checksum lido do arquivo;
  - 2.6. Emitir mensagem de erro (**FFFD** no acumulador) em caso de discrepância e parar.
3. Ler, ao final do arquivo, o valor do campo de endereço da primeira instrução executável e **armazená-la no acumulador**.

11

PCS

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

## Exercício 1 (não deve ser entregue)

- Familiarizando-se com o comando de leitura 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
  - Coloque os caracteres ‘abc’ no arquivo que representa o disco
  - Monte e execute o seguinte código:
 

|     |    |       |                                      |
|-----|----|-------|--------------------------------------|
|     | @  | /0000 | ; endereço absoluto                  |
|     | JP | INI   | ; vai para início do programa        |
| INI | GD | /300  | ; lê três words do disco cujo ID     |
|     | GD | /300  | ; é 00 e carrega o valor lido no     |
|     | GD | /300  | ; acumulador. Fim de arquivo: /FFFF. |
| END | HM | END   | ; fim                                |
|     | #  | INI   |                                      |

12

PCS

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



## Exercícios 2 e 3 (Obrigatórios)

---

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

- O *Loader* deve seguir a estrutura de sub-rotina;
- Você pode usar o arquivo “TYGXXA08E03\_main.mvn” como seu main para testes, adaptando **os valores** dos parâmetros conforme necessidade
- Parâmetro de entrada da sub-rotina:
  - Número da Unidade Lógica (UL) do tipo **Disco** (0x3) , do arquivo a ser carregado na memória: **LOADER\_UL**
- Valor de retorno (acumulador)
  - Endereço da primeira instrução executável, lida do final do arquivo

13

PCS

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



## Exercício 2

---

1. Desenvolva, uma sub-rotina *loader* que carregue na memória a imagem gravada no arquivo gerado no segundo passo da implementação do *dumper* da aula passada (arquivo TYGXXA07E02\_dumper.asm). Esse arquivo deve conter o endereço inicial de carga na memória e o tamanho da imagem a ser carregada, não contendo os demais elementos do formato definido para o *dumper*. A implementação deve incluir o tratamento de erro anteriormente indicado, caso a imagem não caiba na memória. Desenvolva um programa principal de teste. **(Obrigatório)**

**Nomes dos Arquivos:** TYGXXA08E02\_main.asm  
TYGXXA08E02\_dumper.asm  
TYGXXA08E02\_loader.asm

14

PCS

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

## Exercício 3

2. Finalmente, desenvolva o restante do *loader*, permitindo que ele carregue na memória o arquivo gerado no último passo da implementação do *dumper* da aula passada (arquivo TYGXXA07E03\_dumper.asm), contendo a imagem no formato completo. A implementação deve incluir o tratamento de erro anteriormente indicado, quando houver uma discrepância de checksum. **(Obrigatório)**

Nomes dos Arquivos: TYGXXA08E03\_main.asm  
TYGXXA08E03\_dumper.asm  
TYGXXA08E03\_loader.asm

15

PCS

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


## 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 relocador.MvnRelocator <arquivo-objeto> <arquivo-saida> <base-relocação> <endereço-inicio-execução>`
  - **Exemplo:** `java -cp MLR.jar relocador.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`

16



|  <p> <b>PCS</b><br/>           PCS 2302/2024<br/>           Laboratório de<br/>           Fundamentos da<br/>           Eng.de Computação         </p> <p>           Aula 8:<br/>           Construção de um<br/>           Dumper para o<br/>           simulador MVN         </p> <p>           Autores:<br/>           Anna H. R. Costa<br/>           Jaime S. Sichman<br/>           João José Neto<br/>           Paulo S. Muniz Silva<br/>           Ricardo L. A. Rocha         </p> <p>           Reestruturação:<br/>           Paulo S. Muniz Silva<br/>           v.1.0 ago. 2012         </p> | Tabela de mnemônicos para as instruções da MVN (de 2 caracteres) |  |  |  |
|---|--|--|--|--|
|   | Operação 0<br><b>Jump</b><br>Mnemônico <b>JP</b>                 | Operação 1<br><b>Jump if Zero</b><br>Mnemônico <b>JZ</b>   | Operação 2<br><b>Jump if Negative</b><br>Mnemônico <b>JN</b> | Operação 3<br><b>Load Value</b><br>Mnemônico <b>LV</b>       |
|   | Operação 4<br><b>Add</b><br>Mnemônico <b>+</b>                   | Operação 5<br><b>Subtract</b><br>Mnemônico <b>–</b>        | Operação 6<br><b>Multiply</b><br>Mnemônico <b>*</b>          | Operação 7<br><b>Divide</b><br>Mnemônico <b>/</b>            |
|   | Operação 8<br><b>Load</b><br>Mnemônico <b>LD</b>                 | Operação 9<br><b>Move to Memory</b><br>Mnemônico <b>MM</b> | Operação A<br><b>Subroutine Call</b><br>Mnemônico <b>SC</b>  | Operação B<br><b>Return from Sub.</b><br>Mnemônico <b>RS</b> |
|   | Operação C<br><b>Halt Machine</b><br>Mnemônico <b>HM</b>         | Operação D<br><b>Get Data</b><br>Mnemônico <b>GD</b>       | Operação E<br><b>Put Data</b><br>Mnemônico <b>PD</b>         | Operação F<br><b>Operating System</b><br>Mnemônico <b>OS</b> |

|  <p> <b>PCS</b><br/>           PCS 2302/2024<br/>           Laboratório de<br/>           Fundamentos da<br/>           Eng.de Computação         </p> <p>           Aula 8:<br/>           Construção de um<br/>           Dumper para o<br/>           simulador MVN         </p> <p>           Autores:<br/>           Anna H. R. Costa<br/>           Jaime S. Sichman<br/>           João José Neto<br/>           Paulo S. Muniz Silva<br/>           Ricardo L. A. Rocha         </p> <p>           Reestruturação:<br/>           Paulo S. Muniz Silva<br/>           v.1.0 ago. 2012         </p> | 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 |