PCS3616

Programação de Sistemas

(Sistemas de Programação)

Semana 5, Aula 9

Introdução

Programação em linguagem de máquina E/S

Trechos de Código Java da MVN

Escola Politécnica da Universidade de São Paulo

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**

Exercício

Desenvolva um programa que, dada uma sequência de 5 dados, verifica se ela está escrita em ordem decrescente. No caso da sequência estar ordenada, o valor na variável de saída deve ser 0001, em caso contrário, o valor deve ser 0000. Use auto-modificação do código para ler a sequência de dados e desenvolva sub-rotina **Menor** (dados x e y verifica se x<y).

Nos comentários do código documente as condições, premissas e limitações consideradas.

Endereço de início do programa principal: 0000
Endereço da variável de saída: 0002 (resultado)
Endereços da lista de dados: 0004 a 000C

E/S na MVN

- As instruções “D” e “E” (ou GD e PD em “assembly”) permitem o acesso a dados externos à MVN. Por enquanto o acesso tem sido usado por meio de teclado e monitor, ou seja, da seguinte forma:
  - **Entrada: <END> D000**
  - **Saída: <END> E100**
- No primeiro byte da instrução apresenta-se o código (D, ou E) e o dispositivo de acesso, 0 para teclado e 1 para monitor. No segundo byte apresenta-se a unidade lógica que mapeia algum elemento externo. Na MVN só há um teclado e um monitor, logo as unidades são representadas por 00.

Operações de E/S na MVN

OP	Tipo	Dispositivo

OP — D (entrada) ou E (saída)

Tipo — Tipos de dispositivo:

- 0 = Teclado
- 1 = Monitor
- 2 = Impressora
- 3 = Disco

Dispositivo — Identificação do dispositivo. Pode-se ter vários tipos de dispositivo, ou unidades lógicas (LU). No caso do disco, um arquivo é considerado uma unidade lógica.

Pode-se ter, portanto, até 16 tipos de dispositivos e, cada um, pode ter até 256 unidades lógicas.

Uso de E/S

```
; ========================================================================
; Exemplo de entrada e saída (leitura do teclado/escrita no monitor).
;
; O programa imprime "X: " na tela e espera o usuário digitar um valor para
; X, que deve ser uma sequência de dois caracteres quaisquer. Depois, soma 1
; ao valor de cada um dos bytes lidos e imprime o resultado.
;
; Exemplo 1: se o usuário digitar "ab", o programa irá imprimir "bc".
; Exemplo 2: se o usuário digitar "Zz", o programa irá imprimir "[{".
; ========================================================================
;
0000 0100  ; Pula para o início das instruções;
;
; ========================================================================
; Área de dados
; ========================================================================
;
00F6 0001  ; Constante: 0x001
00F8 0100  ; Constante: 0x100
00FA 0000  ; Variável: Byte 1
00FC 0000  ; Variável: Byte 2
00FE 0000  ; Variável usada pelo programa principal para guardar o dado lido
;
; ========================================================================
; Programa
; Obs: o programa não está otimizado para desempenho, e sim para
; compreensão da sua lógica. A ordem de algumas operações poderia ser
; trocada para economizar algumas instruções.
; ========================================================================
;
```

```
; Escrita no monitor
;
0100 3058  ; Carrega 'X' no acumulador
0102 E100  ; Imprime 'X'
0104 303A  ; Carrega ':' no acumulador
0106 E100  ; Imprime ':'
0108 3020  ; Carrega ' ' no acumulador
010A E100  ; Imprime ' '
;
; Leitura do teclado.
; IMPORTANTE: a leitura sempre é realizada dois bytes por vez!
;
010C D000  ; Lê dois bytes do teclado
010E 90FE  ; Guarda os bytes na memória
;
; Separação dos bytes lidos
;
0110 60F8  ; Multiplica o valor lido por 0x100
0112 70F8  ; Divide o valor lido por 0x100
0114 90FC  ; Armazena o byte na variável "Byte 2"
0116 80FE  ; Carrega no acumulador o valor lido novamente
0118 70F8  ; Divide por 0x100
011A 90FA  ; Armazena na variável "Byte 1"
;
; Cálculo e impressão do resultado
;
011C 80FA  ; Carrega o 1o. byte
011E 40F6  ; Soma 1
0120 E100  ; Imprime
0122 80FC  ; Carrega o 2o. byte
0124 40F6  ; Soma 1
0126 E100  ; Imprime
0128 0200  ; Pula para a instrução de parada.
;
0200 C200  ; The End.
```

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

E/S na MVN

- Para se transformar caracteres ASCII que representam números em valores binários/ hexadecimais é necessário convertê-los a partir de seu valor em ASCII.
  - Lembre-se que na representação dos valores em ASCII a ordem é preservada sempre, tanto para os números quanto para as letras (ordem lexicográfica usual).
- Dessa maneira ao se receber como entrada dois bytes contendo os caracteres “78”, tem-se na verdade 0x3738, os caracteres codificados em ASCII.
- Portanto para se transformar esses caracteres nos números correspondentes é necessário construir uma rotina de conversão para cada um dos bytes.
  - Basicamente basta subtrair 0x30 de cada byte para obtê-los.
  - Depois multiplica-se o byte 1 por 0x100 (16^2) e adiciona-se ao byte 2.

Dicas

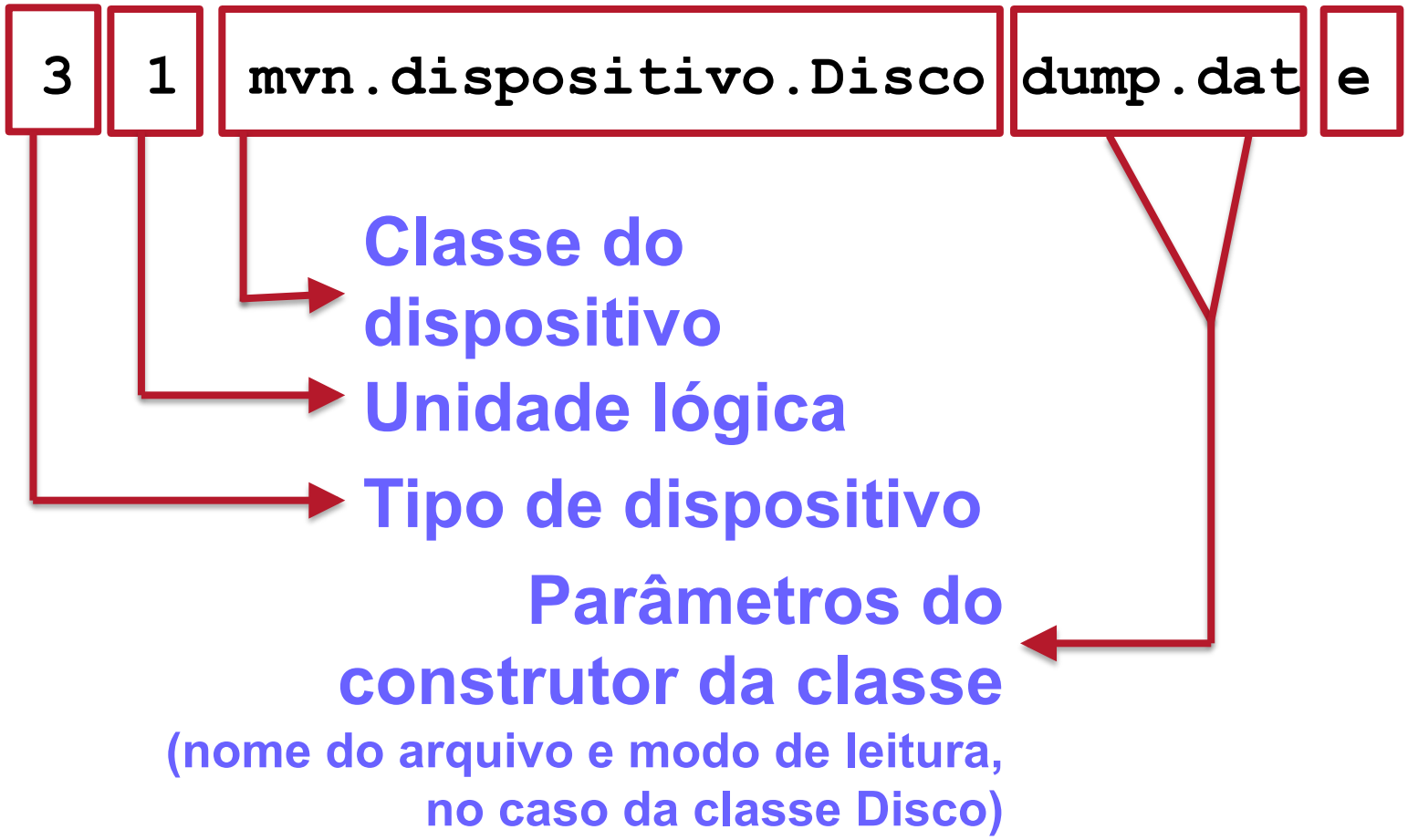
- Para facilitar o desenvolvimento, é possível inicializar os dispositivos da MVN através de arquivo.
- Para tanto, crie um arquivo "**disp.lst**" na raiz do projeto NetBeans ou na pasta do arquivo *.JAR.
- Cada linha deste arquivo indica um dispositivo a ser adicionado no gerenciador.

- Exemplo de arquivo:

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

Dicas

- Sintaxe:



MVN em Java

```
package mvn;
import java.util.BitSet;

public class Bits8 extends BitSet implements Comparable<Bits8>{
        public static final int BYTE_SIZE= 8;
        public static final int NIBBLE_SIZE= 4;
        public static final int HEXBYTE_SIZE= BYTE_SIZE / NIBBLE_SIZE;
        public Bits8(){
                super(BYTE_SIZE);
                }
        public Bits8(byte initialValue){
                this();
                setValue(initialValue);
                }
        private void setValue(byte value){
                for(int i = 0; i < BYTE_SIZE; i++){
                        this.set(i, (1 & (value >> i)) != 0);}
                        }
```

MVN – Métodos estáticos

```
package mvn;

public class MvnControle{
private static final int MIN_ADDRESS= 0x0000;
private static final int MAX_ADDRESS= 0x0FFF;
private static final String DEV_NAME_TECLADO= "Teclado";
// Este nome deve ser igual ao da classe
private static final String DEV_NAME_MONITOR= "Monitor";
private static final String DEV_NAME_IMPRESSORA= "Impressora";
private static final String DEV_NAME_DISCO= "Disco";

public MvnControle(){
          this.memoria = new Memoria(MIN_ADDRESS, MAX_ADDRESS);
          this.io = new GerenciadorDispositivos();
          this.cpu = new UnidadeControle(io, memoria);}
...
public static String availableDevices(){
          StringBuilder out = new StringBuilder(MSG_HEADER_TIPOSDISPOSITIVOSDISPONIVEIS);
          out.append(System.getProperty("line.separator"));
          for(int i = 0; i < DEVICES.length; i++){
                    out.append(String.format("   %-10s -> %d", DEVICES[i][0], i));
                    out.append(System.getProperty("line.separator"));}
          return out.toString();
          }
```

MVN – Métodos estáticos

```
package mvn.controle;
...
public class PainelControle {...
 public PainelControle(MvnControle mvn, boolean debug) {
     this.mvn = mvn;
     this.terminal = new TerminalPadrao(debug);
     initialize();
         }
...
  public void dispositivos() throws MVNException {
    terminal.exibeLinha(mvn.listDispositivos());
    terminal.pulaLinha();
    char acao = terminal.obtem(MSG_PROMPT_ALTERARDISPOSITIVO, " ").charAt(0);
        if (acao != ADICIONAR && acao != REMOVER) {return;}
        terminal.exibe(MvnControle.availableDevices());
        String strTipo = terminal.obtem(MSG_PROMPT_TIPODISPOSITIVO, "");
        if (strTipo.isEmpty()) {return;}
…
```

Classes abstratas na MVN

```
package mvn.controle;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.PrintStream;

public abstract class Terminal{
        private static final String PROMPT_STRING= "> ";
        private static final int TERMINAL_WIDTH= 80;
        private static final String ERR_NAO_ESPECIFICADO= "Erro não
                        especificado.";
        private static final String ERR_ENTRADA_TECLADO= "Erro na
                        entrada do terminal";
        private BufferedReader in;
        private PrintStream out;
        private PrintStream err;
        private boolean ativado;
        private boolean debug;
        ...
```

Classes abstratas na MVN

```
package mvn.controle;
...
public abstract class Terminal{
          ...
          public Terminal(boolean debug){
                    this.debug = debug;
                    this.ativado = true;
                    in = setIn();
                    out = setOut();
                    err = setErr();
                    }
          public Terminal(){
                    this(false);
          }
          public void desativa(){
                    ativado = false;}
          public void ativa(){
                    ativado = true;}
...
protected abstract BufferedReader setIn();
protected abstract PrintStream setOut();
protected abstract PrintStream setErr();  ...
```

Interfaces na MVN

```
package mvn;
import mvn.controle.MVNException;
public interface Dispositivo{
public static final String ERR_WRITEONLYDEVICE= "Dispositivo
       \"%s\" disponí-vel somente para escrita.";
public static final String ERR_READONLYDEVICE= "Dispositivo
       \"%s\" disponí-vel somente para leitura.";

       public void escrever(Bits8 in) throws MVNException;
       public Bits8 ler() throws MVNException;
       public boolean podeLer();
       public boolean podeEscrever();
       public void reset() throws MVNException;
       public Bits8 skip(Bits8 val) throws MVNException;
       public Bits8 position() throws MVNException;
       public Bits8 size() throws MVNException;
```

Interfaces na MVN

```
package mvn.dispositivo;

public class Disco implements Dispositivo {

public Disco(String arquivo, char modoOperacao) throws MVNException {
        switch (modoOperacao) {
            case MODO_LEITURA:
                this.modoOperacao = LEITURA;
                break;
            case MODO_ESCRITA:
                this.modoOperacao = ESCRITA;
                break;
            case MODO_LEITURAESCRITA:
                this.modoOperacao = LEITURAESCRITA;
                break;
            default:
                this.modoOperacao = INVALIDO; }
        this.arquivo = new File(arquivo);
        outFile = null;
        inFile = null;
        initializeDevice();
    }
```

Interfaces na MVN – continuação

```
package mvn.dispositivo;

public class Disco implements Dispositivo {
…
          public void escrever(Bits8 in) throws MVNException{
     //código de escrever} ;
          public Bits8 ler() throws MVNException{
          //código de ler} ;
          public boolean podeLer(){
                    return modoOperacao == LEITURA || modoOperacao == LEITURAESCRITA;}
          ...
```

Tratando Exceções

```
public void reset() throws MVNException {
        if (podeLer()) {
            try {
                inFile.close();
                inFile = new FileInputStream(arquivo);
            } catch (IOException ex) {
                throw new MVNException(ERR_IOERROR,
                                       arquivo.getName());
            }
        } else {
            // modo de operacao inadequado
            throw new MVNException(ERR_WRITEONLYDEVICE,
                                   this);
        }
    }
```

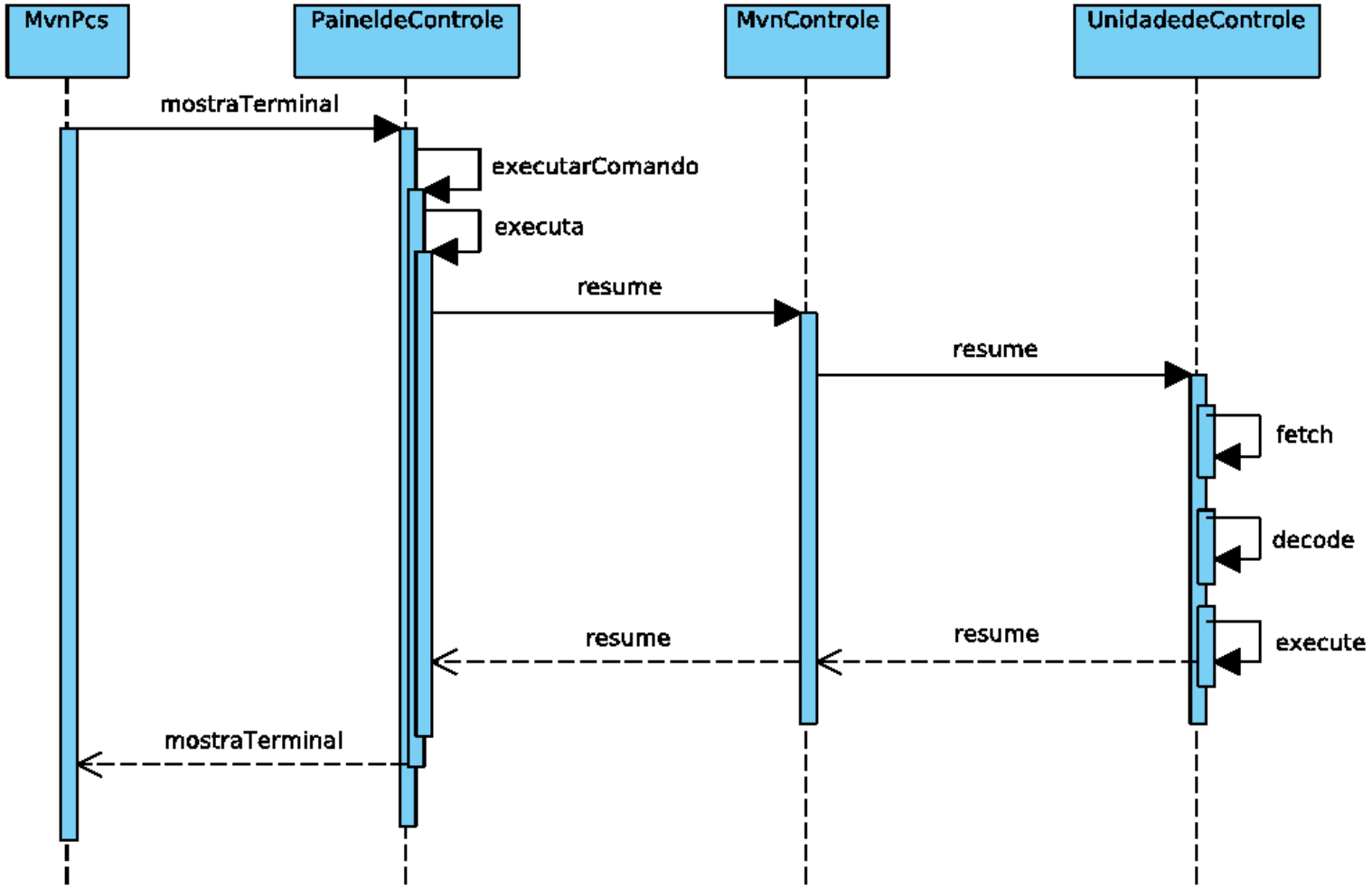
Trata IOException

Encapsula IOException como MVNException

Inicializando a MVN

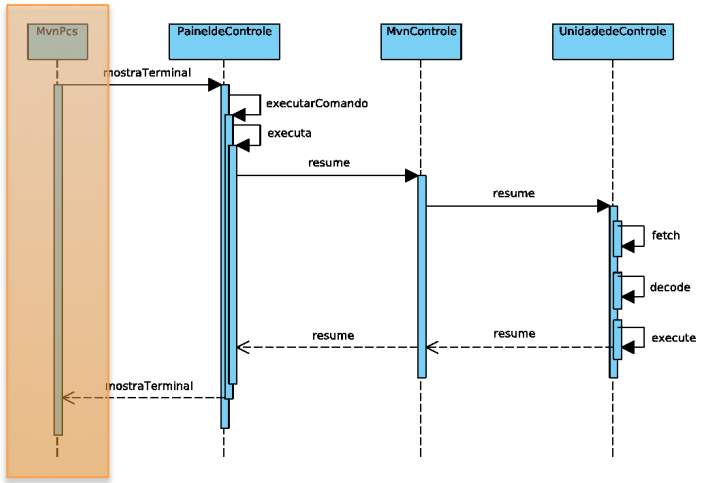
```
package mvn;
import mvn.controle.PainelControle;

/**
 * Classe que inicia a MVN.
*/
public class MvnPcs{
/*** Controlador da MVN */
private static MvnControlemvnPcs;
/*** Painel de Controle da MVN */
private static PainelControle painel;

public static void main(String args[]){
        mvnPcs = new MvnControle();
        painel = new PainelControle(mvnPcs, false);
        painel.mostrarTerminal();
        }
} // Fim da Classe MvnPcs
```

Dinâmica da Inicialização



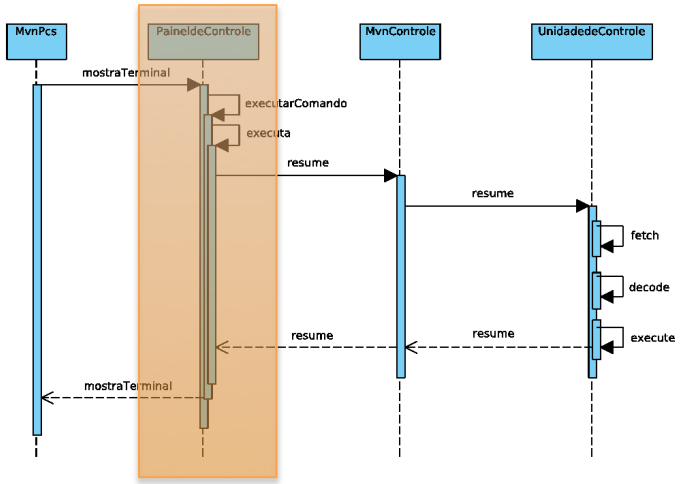
Código MVN do Diagrama de Sequência

```
public static void main(String args[]){
        mvnPcs = new MvnControle();
        painel = new PainelControle(mvnPcs, false);
        painel.mostrarTerminal();
}
```



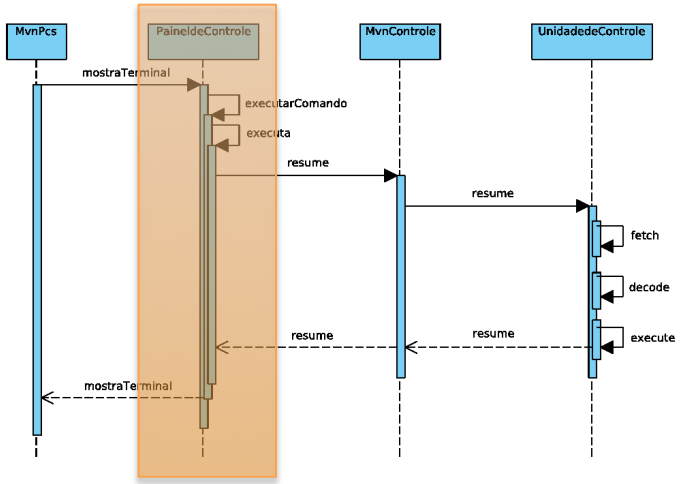
Código MVN do Diagrama de Sequência

```
public void mostrarTerminal(){
        …
        try{
                leTerminal = executaComando(comando,
        linhaComando.toString().trim());
        }catch(MVNException e){
                terminal.erro(e);
        }catch(Exception e){
                terminal.erro(ERR_EXECUCAO_TERMINAL, e);
        }
        …
}
```



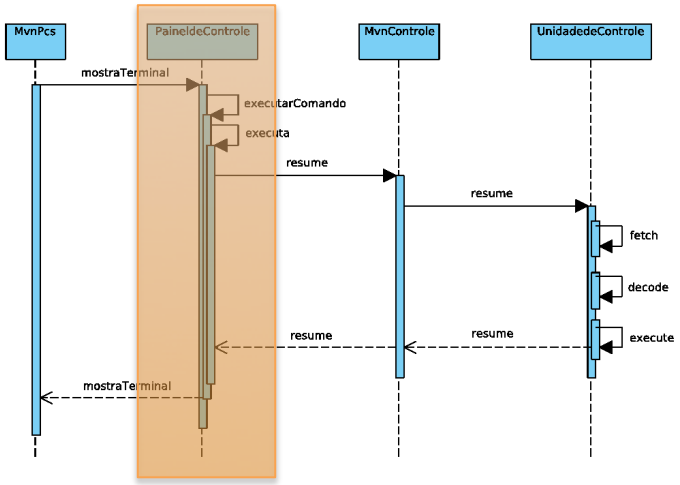
Código MVN do Diagrama de Sequência

```
private boolean executaComando(char comando, String
                                        parametros) throws MVNException {
        switch(comando){
                case CMD_INICIALIZA:
                        initialize();
                        break;
                case CMD_ASCII:
                        loadTextFiletoMemory(parametros);
                        break;
                case CMD_EXECUTA:
                        executa(ExtractArguments(parametros));
                        break;
                ...
        }
        return true;
}
```



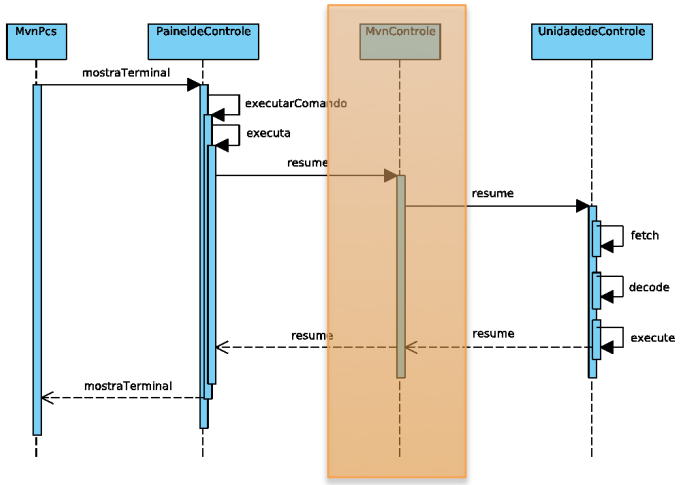
Código MVN do Diagrama de Sequência

```
private void executa(String[] args) throws MVNException{
        …
        boolean continueRunning;
        do{
                continueRunning = mvn.resume();

                if(outputBuffer.length() > 0){
                        terminal.exibe(outputBuffer.toString());
                        outputBuffer.delete(0, outputBuffer.length());
                }

                …
        }while(continueRunning);

}
```



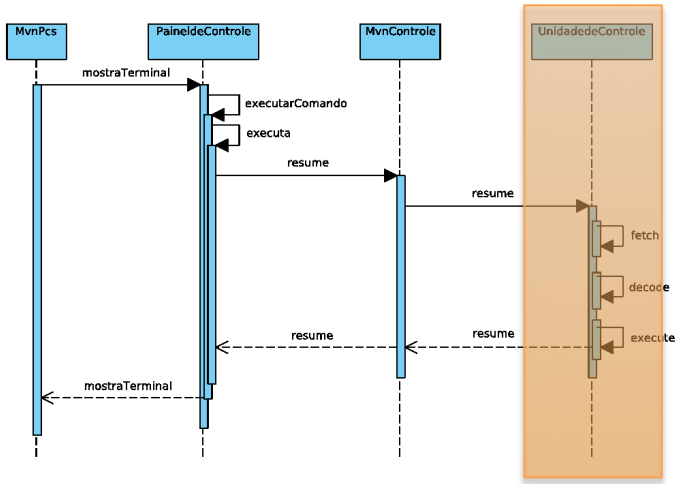
Código MVN do Diagrama de Sequência

```
public boolean resume() throws MVNException{
        boolean stillrunning = cpu.resume();

        if(showRegs){
                outputInfo(cpu.obterRegs().toString());
        }

        return stillrunning;
}
```



Código MVN do Diagrama de Sequência

```
public boolean resume() throws MVNException{
        fetch();
        decode();
        execute();

        return(regs.getRegister(OP).toInt() != HM);
}
```



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.