

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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto


Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

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

**Aula 08**  
**Linguagem Simbólica**  
**Montador Absoluto**

**Professores:**  
Jaime Simão Sichman (PCS 2302)  
Ricardo Luis de Azevedo da Rocha (PCS 2024)

1



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007


Aula 8:  
Linguagem Simbólica  
Montador Absoluto

Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

**Roteiro**

- Construção de Programas em Linguagem de Máquina
- Linguagem Simbólica
- Montador
  - Esquema geral
  - Estruturas de Dados
  - Algoritmos
- Parte Experimental
  - Implementação de um montador absoluto para o simulador MVN

2



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007


Aula 8:  
Linguagem Simbólica  
Montador Absoluto

Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

**Construção de Programas em Linguagem de Máquina (1)**

- Escrever um programa usando diretamente codificação binária não é uma tarefa simples, e tampouco agradável.
- Entretanto, foi este tipo de codificação que permitiu a construção dos primeiros programas do curso.
- Naturalmente, se um programa é muito grande ou se lida com diversas estruturas complexas (listas, etc.), a sua codificação se torna ainda mais difícil e complexa.

3



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007


Aula 8:  
Linguagem Simbólica  
Montador Absoluto

Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

**Construção de Programas em Linguagem de Máquina (2)**

- Por conta disso, torna-se imprescindível construir alguma **abstração** que facilite a programação e a verificação dos programas.
- A primeira idéia, mais natural, é utilizar o modelo de máquina existente e, a partir dele, definir nomes (mnemônicos) para cada instrução da máquina.
- Posteriormente, verifica-se que somente isso não basta, pois é necessário lidar com os endereços dentro de um programa (rótulos, operandos, sub-rotinas), com a reserva de espaço para tabelas, com valores constantes.
- Enfim, é necessário definir uma **linguagem simbólica**.

4



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007


Aula 8:  
Linguagem Simbólica  
Montador Absoluto

Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

**Linguagem Simbólica**

- Uma instrução de máquina tem usualmente o aspecto seguinte em sua **imagem mnemônica**:  
**0012 JZ 042 ; 1042 0012=rótulo JZ=mnemônico 042=operando numérico**
- A mesma instrução, em linguagem simbólica, pode ser escrita com ou sem um rótulo simbólico, e pode também referenciar um operando através de um rótulo simbólico ou numérico:  
**Q JZ R ; Q=rótulo JZ=mnemônico R=operando simbólico**  
**JZ R ; rótulo omitido JZ=mnemônico R=operando simbólico**  
**Q JZ 042 ; Q=rótulo JZ=mnemônico 042=operando numérico**  
**JZ 042 ; rótulo omitido JZ=mnemônico 042=operando numérico**
- Convenciona-se que sempre o primeiro elemento da linha é um rótulo; caso o rótulo for omitido deverá haver uma instrução
- Entre os elementos de uma linha deve haver ao menos um espaço
- Cada linha deve conter uma instrução/pseudo-instrução completa
- À direita de um ponto-e-vírgula, todo texto é ignorado (=comentário)
- Mnemônicos e significado das pseudo-instruções:
  - @ (Operando numérico: define endereço da instrução seguinte)
  - \$ (Reserva de área de dados)
  - # (Final físico do texto-fonte. Operando=endereço de execução)
  - K (Constante. Operando numérico = valor da constante, em hexadecimal)

5



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto

Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

**Exemplo de programa em linguagem simbólica**

O programa abaixo, que foi dado como exemplo na aula 2:

```

0100      8F00  Obtém o endereço para onde se deseja mover o dado
0102      4F02  Compõe o endereço com o código de operação Move
0104      9106  Guarda instrução montada para executar em seguida
0106      9000  Executa a instrução recém-montada
0108      ....  Provavelmente, o código seguinte altera o conteúdo de 0F00
....
015C      0100  Volta a repetir o procedimento, para outro endereço.
....
0F00      034C  Endereço (34C) para onde se deseja mover o dado
0F02      9000  Código de operação Move, com operando 000

```

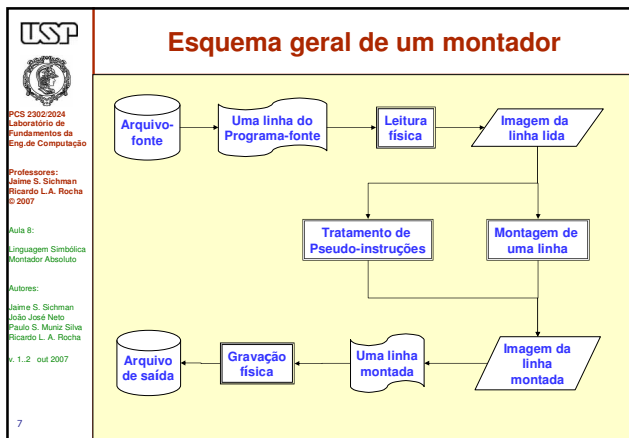
codificado em linguagem simbólica, fica com o seguinte aspecto:

```

# /0100 ; @origem do código 0100=posição de memória (em hexadecimal)
P LD E ; P=rótulo LD=load R=endereço simbólico da constante 034C
+ M ; +add M=rótulo de onde está uma instrução Move 0000
MM X ; MM=move X=endereço da instrução seguinte
X MM 0 ; reservado para guardar a instrução recém-montada
...
JP P ; JP=jump (desvio) P=rótulo da primeira instrução deste programa
...
E X 034C ; E=rótulo X=constante 034C=operando numérico, em hexadecimal
M MM /0000 ; M=rótulo MM=move 0000=operando zero
# P ; #final físico P=rótulo de primeira instrução a ser executada

```

6



**USP**

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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto

Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

### Construção de um Montador

- A construção de um montador pressupõe que sejam resolvidos os seguintes problemas:
  - definição dos rótulos:** determinar qual será o endereço efetivo de um nome encontrado;
  - definição das instruções:** determinar os mnemônicos;
  - definição das pseudo-instruções:** determinar os mnemônicos e o que fazem.
- Para cumprir esta tarefa é necessário completar, em primeiro lugar, as definições dos mnemônicos (instruções e pseudo-instruções), para se pensar posteriormente, nos algoritmos.

8

**USP**

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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto

Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

### Pseudo-Instruções

- Nas aulas anteriores foram determinados os mnemônicos das instruções, nesta aula serão definidas aqueles relativos às pseudo-instruções.
- As pseudo-instruções utilizadas no montador desta aula (montador absoluto) são as seguintes:
  - @** : Recebe um operando numérico, define o endereço da instrução seguinte;
  - K** : Constante, o operando numérico tem o valor da constante (em hexadecimal);
  - \$** : Reserva de área de dados, o operando numérico define o tamanho da área a ser reservada;
  - #** : Final físico do texto fonte.

9

**USP**

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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto

Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

### Formas de Construção de um Montador

- Há mais de uma forma de se tratar o problema de construção de um Montador. Pelo menos duas são imediatas:
  - Montador de um passo:** O montador somente lê o texto fonte uma vez, armazenando os rótulos não definidos em uma lista de pendências enquanto gera o código para cada linha de entrada completamente definida, para, ao final, listar os rótulos da lista e completar as linhas de código que ainda não haviam sido completamente definidas;
  - Montador de dois passos:** O montador lê o texto fonte da primeira vez apenas para acertar as definições dos rótulos e pseudo-instruções. Terminada esta fase (passo1), o montador lê novamente o texto fonte, agora com todos os endereços resolvidos para gerar o código correspondente ao programa

10

**USP**

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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto

Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

### Estruturas de Dados do Montador (1)

- O montador precisará de um conjunto de estruturas de dados que o permitirão conduzir a tarefa. Dentro deste conjunto, há as seguintes estruturas de dados:
  - locationCounter** : define a localização atual (endereço corrente) de execução.
  - Tabela de instruções**: define as instruções válidas (símbolo e valor).
  - Tabela de pseudo-instruções**: define as pseudo-instruções válidas (símbolo e valor).
  - Tabela de símbolos**: permite armazenar e recuperar os rótulos (símbolo e endereço real).

11

**USP**

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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007


Aula 8:  
Linguagem Simbólica  
Montador Absoluto

Autores:  
Jaime S. Sichman  
João José Neto  
Paulo S. Muniz Silva  
Ricardo L. A. Rocha  
v. 1.2 out 2007

### Estruturas de Dados do Montador (2)

- Além destas estruturas, o montador utiliza um conjunto de arquivos (um de entrada e pelo menos dois de saída). Pode ser necessário gerar o texto objeto em algum formato específico, para que um programa *loader* possa carregá-lo na memória.
- Pode-se, ainda, armazenar o conteúdo do texto fonte durante o passo 1 para facilitar a execução do passo 2.

12



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto


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

v. 1.2 out 2007

13

## Construção do Montador

- Nesta disciplina foi escolhido realizar um montador de dois passos. Esta escolha nos conduz à definição das ações a serem realizadas em cada um dos dois passos do montador. Assim temos:
  - Passo1:** O objetivo é definir os símbolos encontrados, sejam eles rótulos encontrados antes das instruções, ou ainda rótulos de destino de alguma instrução. Para isso deve:
    - Manter atualizado o endereço de execução corrente, chamado de **locationCounter**.
    - Armazenar os valores dos símbolos (rótulos) na Tabela de Símbolos (TS) para uso posterior no passo 2.
    - Processar as pseudo-instruções.
  - Passo2:** O objetivo é gerar o código objeto e possivelmente um arquivo de listagem contendo além do código objeto, o texto fonte à direita do código objeto. Para isso, este passo deve:
    - Recuperar os valores dos símbolos (da TS).
    - Gerar as instruções.
    - Processar as pseudo-instruções.



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

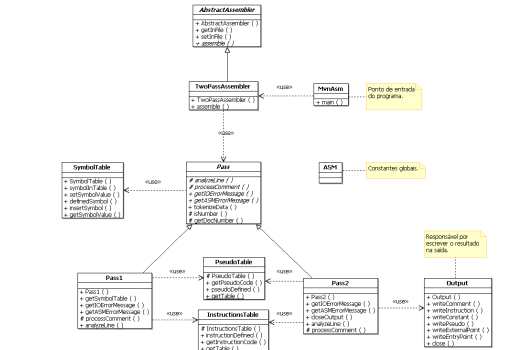
Aula 8:  
Linguagem Simbólica  
Montador Absoluto

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

v. 1.2 out 2007

14


## Diagrama de Classes do Montador Java



```

classDiagram
    class AbstractAssembler {
        +getLine()
        +getTS()
        +assemble()
    }
    class TwoPassAssembler {
        +getLine()
        +getTS()
        +assemble()
    }
    class SymbolTable {
        +SymbolTable()
        +insertSymbol()
        +getSymbol()
        +getSymbolValue()
    }
    class PseudoTable {
        +PseudoTable()
        +insertPseudo()
        +getPseudo()
    }
    class Pass1 {
        +Pass1()
        +getLine()
        +getPseudoTable()
        +getSymbolTable()
    }
    class Pass2 {
        +Pass2()
        +getLine()
        +getPseudoTable()
        +getSymbolTable()
    }
    class InstructionTable {
        +InstructionTable()
        +insertInstruction()
        +getInstruction()
    }
    class ASM {
        +ASM()
    }
    class Output {
        +Output()
        +writeComment()
        +writePseudo()
        +writeSymbol()
        +writeInstruction()
    }
    class Main {
        +Main()
    }
    class GlobalConstants {
    }
    class Response {
    }

    AbstractAssembler <|-- TwoPassAssembler
    TwoPassAssembler <|-- Pass1
    TwoPassAssembler <|-- Pass2
    SymbolTable <|-- PseudoTable
    SymbolTable <|-- InstructionTable
    PseudoTable <|-- InstructionTable
    InstructionTable <|-- ASM
    InstructionTable <|-- Output
    Main --> TwoPassAssembler
    Main --> GlobalConstants
    Main --> Response
    
```



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto


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

v. 1.2 out 2007

15

## Classes do Montador Absoluto

- O montador é definido a partir de uma classe abstrata (*AbstractAssembler*) porque o projeto deve prever a possibilidade de existência de um montador de um passo, ou de dois passos. A partir da decisão de projeto, o montador construído para esta disciplina é de dois passos (*TwoPassAssembler*). As demais classes são:
  - Tabela de instruções** (*InstructionsTable*): define as instruções válidas (símbolo e valor).
  - Tabela de pseudo-instruções** (*PseudoTable*): define as pseudo-instruções válidas (símbolo e valor).
  - Tabela de símbolos** (*SymbolTable*): permite armazenar e recuperar os rótulos (símbolo e endereço real).
  - Passo** (*Pass*): define a estrutura dos passos, que são derivados desta classe (*Pass1* e *Pass2*).
  - Saída** (*Output*): responsável por toda saída de dados para os arquivos.
  - Constantes** (*ASM*): define as constantes utilizadas no montador.
  - Ponto de entrada** (*Main*): contém o programa principal que inicia o montador.



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto

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

v. 1.2 out 2007


16

## Lógica Geral do Montador

- O algoritmo utilizado é o seguinte:
 

```

begin
  Marque o endereço inicial de geração de código como 0 /* locationCounter */;
  abra o arquivo com o programa;
  while não encontra fim de arquivo do
    Passo1: leia uma linha preenchendo a Tabela de Símbolos (TS);
  end
  reinicie o arquivo;
  while não encontra fim de arquivo do
    Passo2: begin
      leia uma linha;
      monte e gere o código objeto correspondente usando a TS;
      escreva o código gerado nos arquivos de saída (em arquivo de saída carregável);
    end
  end
  fecha os arquivos;
end
            
```



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto

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


v. 1.2 out 2007

17

## Leitura e Tratamento das Linhas

```

begin
  Leia a linha até o final <EOL>;
  Separe os tokens da linha (ou seja, as palavras);
  while há tokens do
    if não é comentário then
      | armazena temporariamente o token;
    end
  end
  if há mais de um token armazenado then
    | Analise a linha;
  else
    | Processe o comentário;
  end
  incremente o contador de linhas;
  pegue a próxima linha;
end
            
```



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

Professores:  
Jaime S. Sichman  
Ricardo L.A. Rocha  
© 2007

Aula 8:  
Linguagem Simbólica  
Montador Absoluto

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


v. 1.2 out 2007

18

## Análise de uma Linha

```

begin
  if há 3 tokens then
    // há um rótulo e deve ser tratado;
    if rótulo não está definido na TS then
      | define o valor do rótulo na TS como sendo o locationCounter;
      | incremente o locationCounter de 2 (porque a abstração da MVN é Word);
    else
      | erro! o rótulo já estava definido;
    end
  end
  Teste a validade do restante da linha (operador e operando);
end
            
```



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

Professores:  
 Jaime S. Sichman  
 Ricardo L.A. Rocha  
 © 2007

Aula 8:  
 Linguagem Simbólica  
 Montador Absoluto

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

v. 1.2 out 2007


19

## Teste de Operador e de Operando

```

begin
  verifique se o operador é válido (se é instrução ou pseudo);
  verifique se o operando é válido (se é número ou rótulo);
end

```



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

Professores:  
 Jaime S. Sichman  
 Ricardo L.A. Rocha  
 © 2007

Aula 8:  
 Linguagem Simbólica  
 Montador Absoluto

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

v. 1.2 out 2007


20

## Montagem e Geração de Código

```

begin
  if argumento é instrução then
    pegue e monte o código correspondente a partir da Tabela de instruções;
  else
    if é pseudo then
      trate a pseudo corretamente;
    end
  end
end

```



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

Professores:  
 Jaime S. Sichman  
 Ricardo L.A. Rocha  
 © 2007

Aula 8:  
 Linguagem Simbólica  
 Montador Absoluto


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

v. 1.2 out 2007

21

## Exercícios (1)

- A principal tarefa desta aula é implementar o montador absoluto completo em Java, usando a lógica ilustrada nas transparências, e completando a especificação. Deve-se manter o padrão utilizado até o momento para o código e a documentação.
- O diagrama de classes do montador é aquele apresentado.
- Será fornecido um arquivo fonte com o montador absoluto, faltando implementar a classe referente à **Tabela de Símbolos**.



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

Professores:  
 Jaime S. Sichman  
 Ricardo L.A. Rocha  
 © 2007

Aula 8:  
 Linguagem Simbólica  
 Montador Absoluto


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

v. 1.2 out 2007

22

## Exercícios (2)

- Além desta classe, todo o tratamento referente às pseudo-instruções precisa ser realizado. Deve-se verificar no diagrama de classes e nos arquivos recebidos outras eventuais necessidades para completar o montador.
- Testar o montador com a geração de código do programa fornecido no arquivo **tmontabs.asm**
- O montador deve gerar como saída os arquivos **tmontabs.mvn** e **tmontabs.lst** correspondentes



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

Professores:  
 Jaime S. Sichman  
 Ricardo L.A. Rocha  
 © 2007

Aula 8:  
 Linguagem Simbólica  
 Montador Absoluto

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

v. 1.2 out 2007

23

## Tabela de mnemônicos para a MVN (de 2 caracteres)

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