PCS 3216
Sistemas de Programação

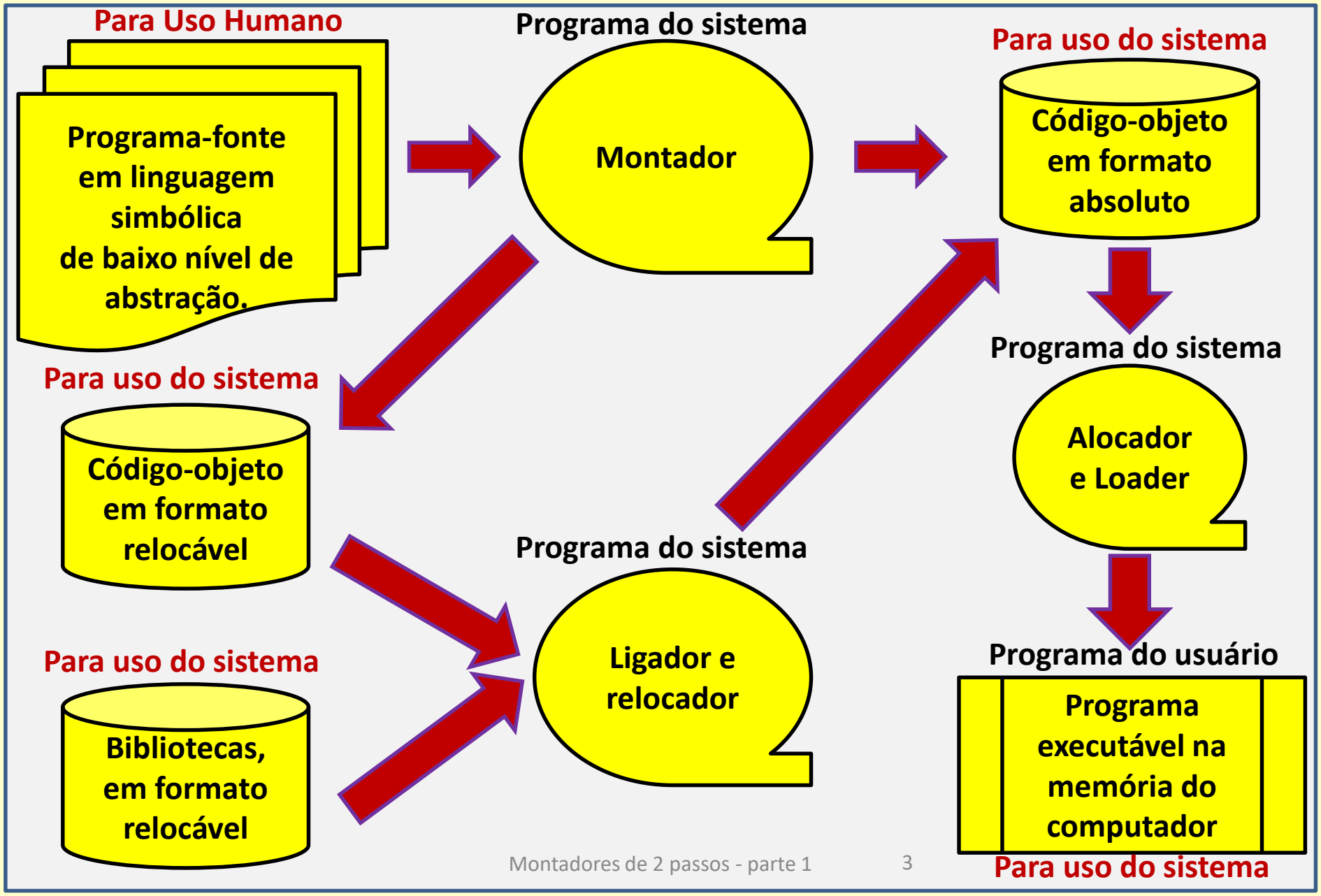
João José Neto

Aula 10 – Montadores de dois passos – Parte 1

Introdução

- Nesta aula:
  - Contextualiza-se de forma mais precisa o montador no âmbito dos sistemas de programação
  - Apresentam-se os principais termos técnicos da área
  - Comparam-se conceitual e estruturalmente os montadores de 2 e de 1 passo
  - Levanta-se a especificação básica de requisitos de um montador para a máquina virtual estudada na disciplina
  - Inicia-se a elaboração de um anteprojeto de montador de dois passos para a máquina virtual.

Preparação de código em linguagem de baixo nível



TIPOS MAIS FREQUENTES DE MONTADORES

Alguns Conceitos

- **Passo de montagem** – corresponde ao processamento que envolve uma leitura completa do programa fonte
- Montadores ***load and go*** – geram diretamente na memória o código objeto, para execução imediata
- ***Backtracking*** – técnica de gerar código objeto incompleto, cujas lacunas vão sendo preenchidas oportunamente
- Montadores de **dois passos** – são os que precisam ler o programa fonte duas vezes para gerar o código objeto
- Montadores de **um passo** – são aqueles que, para gerar o código objeto, precisam ler apenas uma vez o programa fonte
- Montadores que constroem em meio externo o **programa-objeto absoluto** devem gerá-lo no formato de um código binário que possa ser carregado pelo programa ***loader***

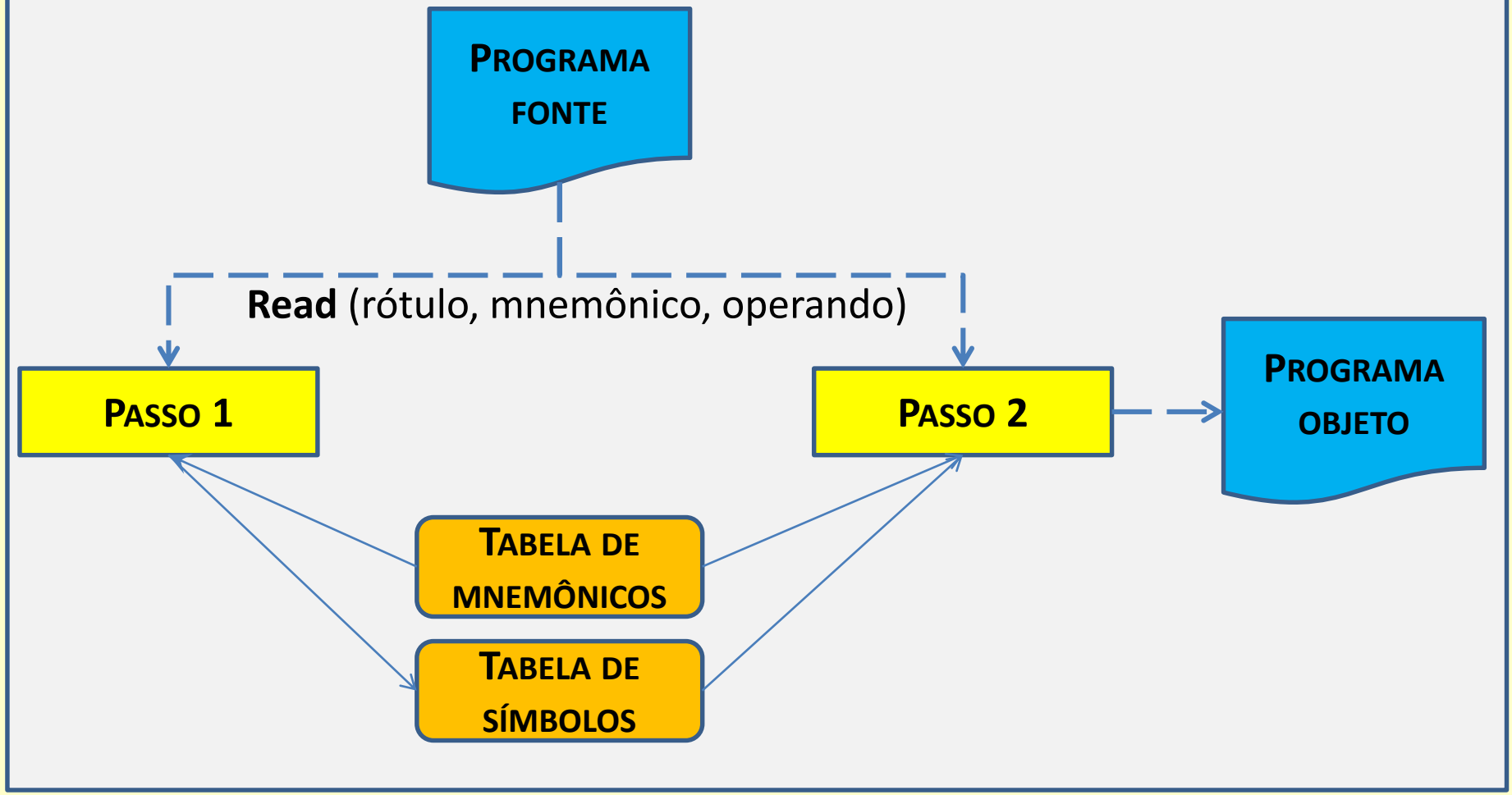
Resolução de endereços simbólicos

- Há duas técnicas clássicas para o problema da ***resolução*** (= conversão para valores numéricos) dos nomes que representam endereços simbólicos:
  - **Em dois passos** – a primeira técnica consiste em efetuar inicialmente a **coleta dos símbolos e o cálculo dos endereços a eles associados**, sem a preocupação em montar o código de máquina das instruções, pois isso é feito em uma segunda etapa.
  - **Em um passo** – a segunda consiste em, sempre que possível, efetuar simultaneamente essas duas tarefas, postergando somente a montagem de instruções que contenham ***referências à frente***, de modo que a montagem venha a ocorrer apenas quando se tornar possível a determinação exata do endereço associado ao símbolo em questão.

Montagem em dois e em um passo

- Aos montadores que funcionam usando a técnica da primeira solução dá-se o nome de ***montadores de dois passos***, porque, para completar a montagem de um programa, o montador necessita efetuar **duas leituras** completas do mesmo:
  - uma para montar a **tabela de símbolos e de atributos**, e
  - outra, para efetuar a **construção do programa traduzido**, em linguagem de máquina, a partir do texto fonte que foi lido e das tabelas construídas no primeiro passo.
- Os montadores que seguem o segundo esquema são denominados ***montadores de um passo***, e realizam **apenas uma leitura** do programa fonte, mas exigem para isso a manutenção de uma ***lista de pendências***;

Organização de um montador simples, em dois passos



Implementação de montadores de dois passos

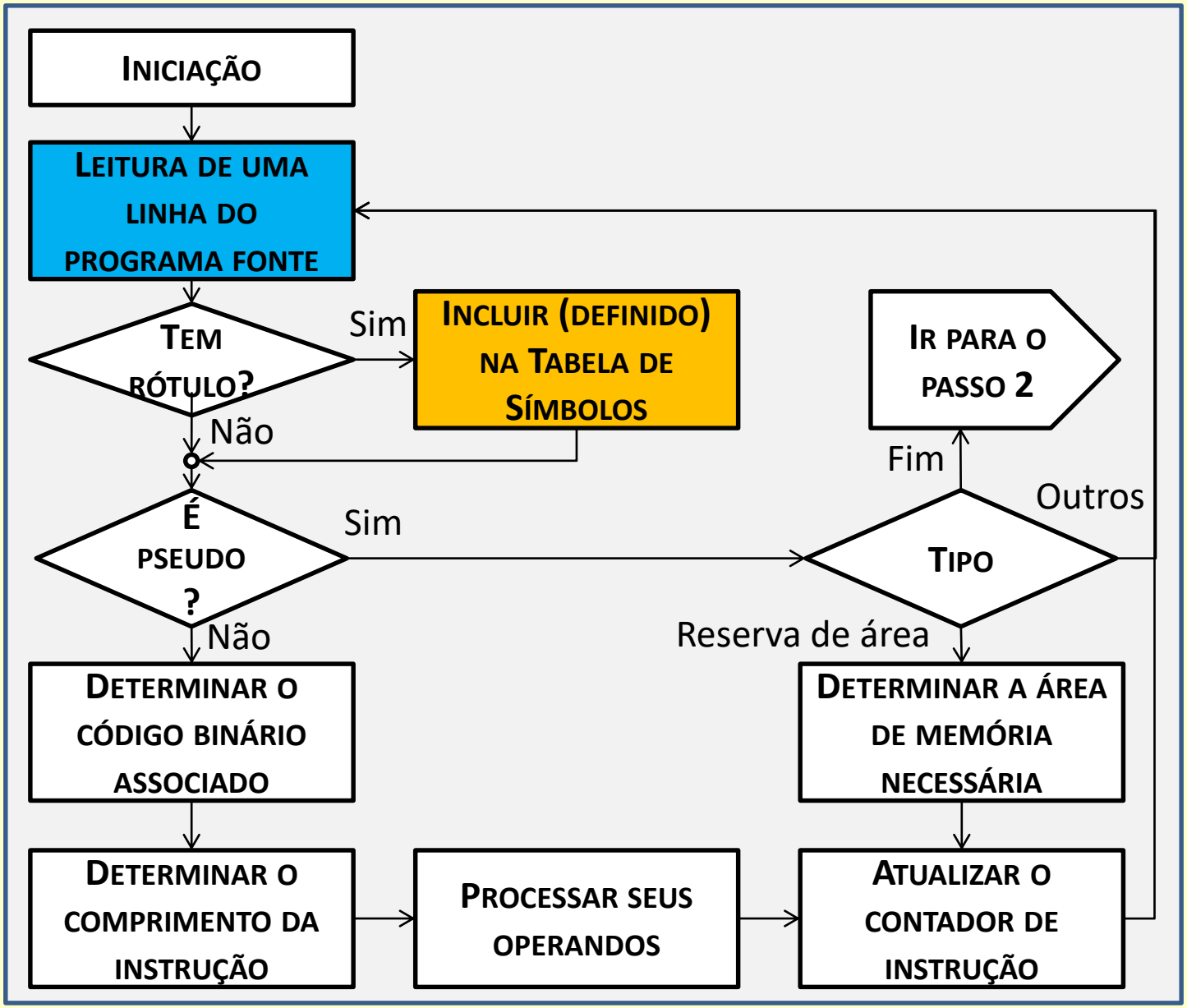
- É a arquitetura de montador mais difundida
- De um lado, tem necessidade de uma ***área menor de armazenamento*** (em cada passo)
- Porém, exige a ***releitura*** completa do programa fonte
- Estruturalmente ***mais simples***, propicia uma implementação menos artificiosa
  - **Primeiro passo**:
    - Leitura do texto-fonte para a **montagem da tabela de símbolos**
    - Análise da tabela de símbolos em **busca de inconsistências**
  - **Segundo passo**:
    - Releitura do programa fonte para a **montagem do código-objeto**
    - **Geração** do programa objeto **em formato carregável**

PASSO 1 do Montador

Atividades principais:

- Construção da tabela de símbolos
- Consulta à tabela de mnemônicos
- Construção da tabela de equivalências
- Cálculo do endereço de cada instrução
- Teste de consistência da tabela de símbolos
- Geração de tabelas de referências cruzadas

Lógica resumida do passo 1 do montador

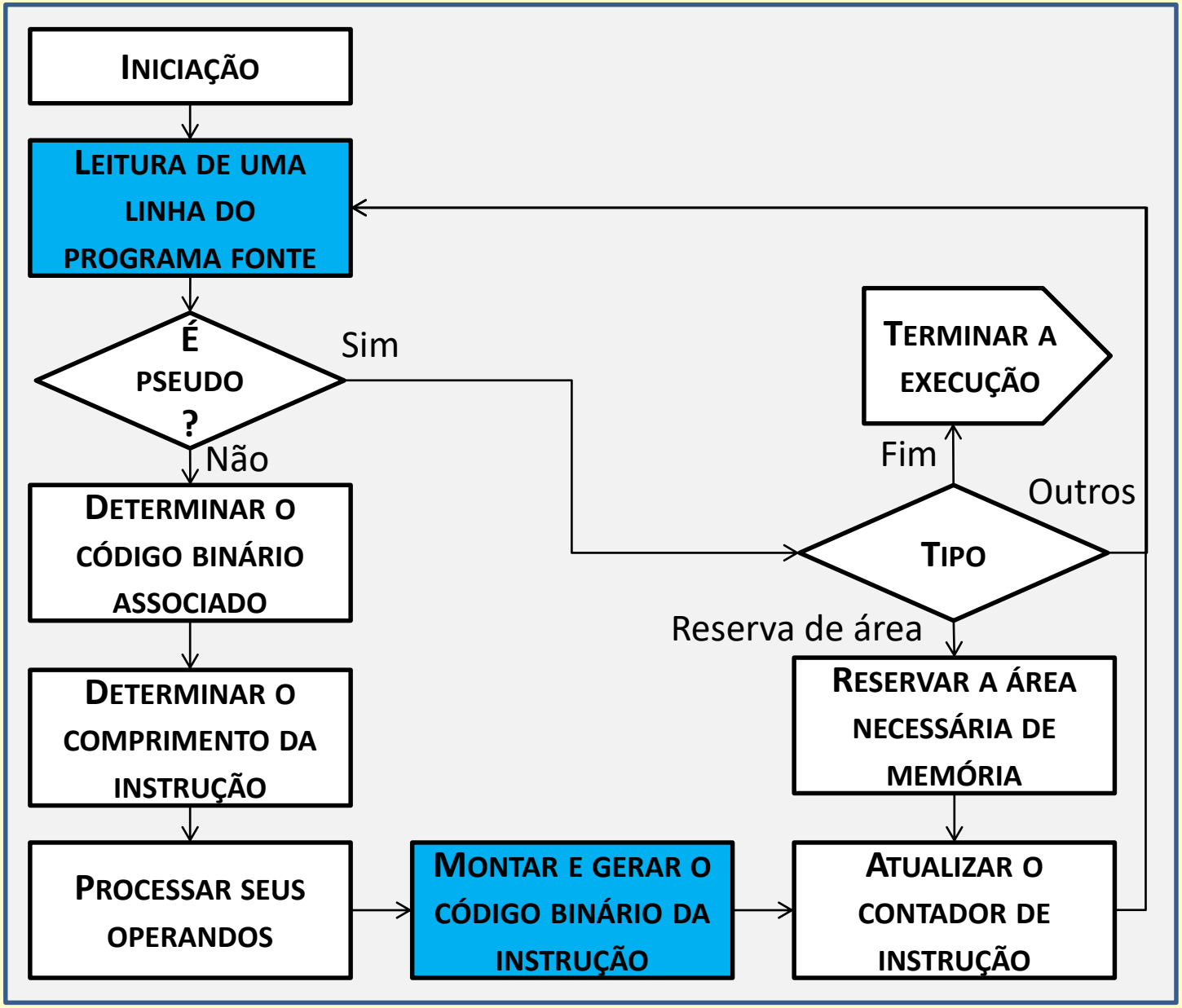


PASSO 2

Atividades principais

- Consulta à tabela de códigos
- Montagem do código objeto
- Avaliação das expressões dos operandos
- Geração da listagem formatada
- Geração do programa objeto

Lógica resumida do passo 2 do montador



Principais Estruturas de Dados

- **Tabela de símbolos** (símbolo - endereço - definido - referenciado)
- Extensão da tabela de símbolo para geração de **referências cruzadas**
  - Linha de definição
  - Link para ordem alfabética
  - Ponteiro para lista de referências
- **Lista de referências** (para referências cruzadas)
  - Link
  - Número da Linha
- **Tabela de mnemônicos e códigos**
  - Mnemônico
  - Código
  - Classe
- **Tabela de equivalências**
  - Símbolo
  - Link
- **Área de saída**
  - Bloco de código objeto gerado

Áreas de Dados usadas pelo montador

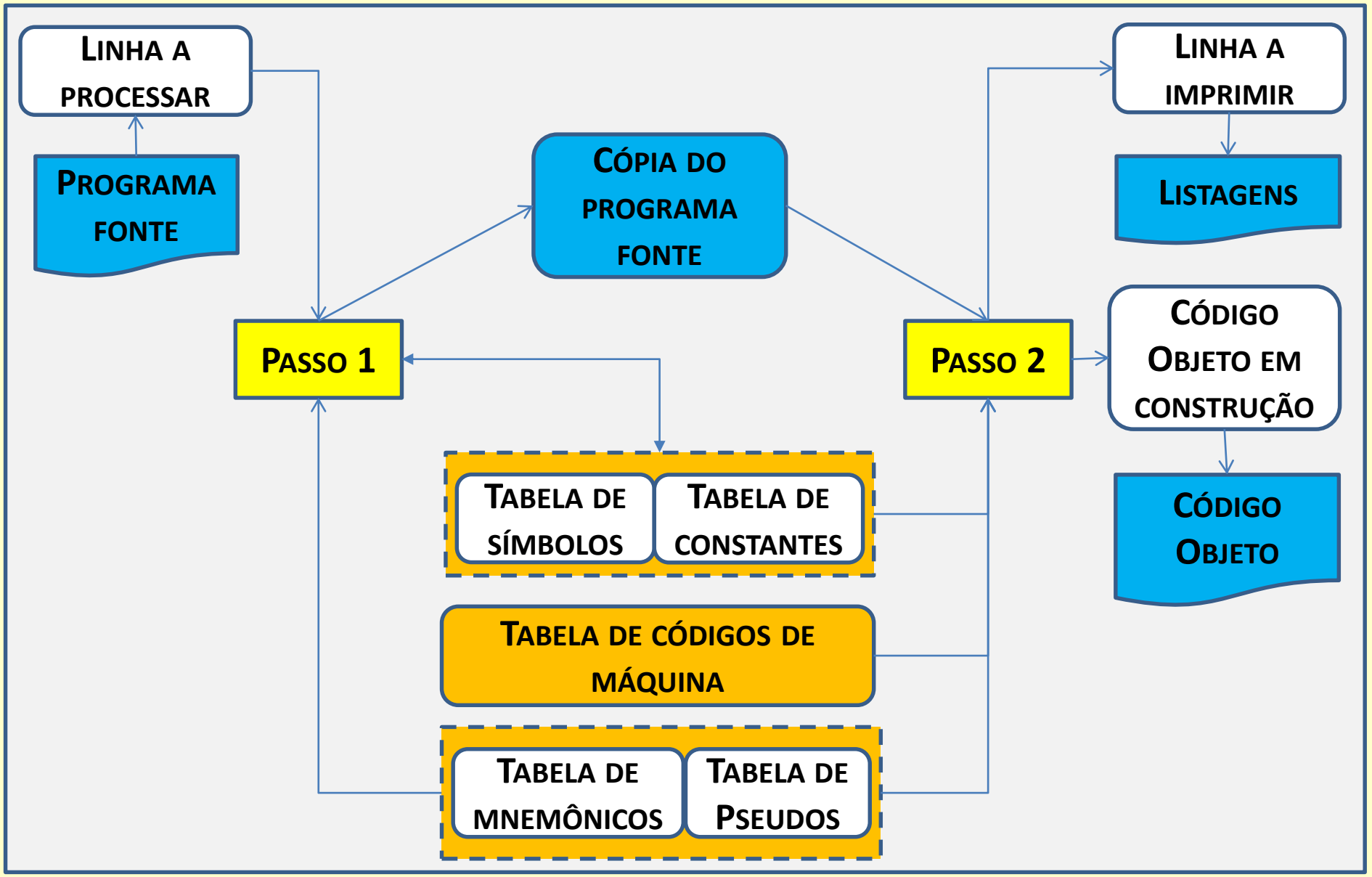


Tabela de Símbolos

- O montador constrói a tabela de símbolos para que o programador possa referenciar por **nome** as **posições de memória** em seus programas em linguagem simbólica
- O montador se incumbe de associar cada **nome simbólico** ao correspondente **endereço físico** na memória
- Quando a localização física das posições ocupadas pelo código é conhecido, **endereços absolutos** são registrados na tabela de símbolos, associados ao endereço simbólico.
- Em montadores para programas com endereçamento **relativo**, os endereços associados aos diversos rótulos (endereços simbólicos) designam localizações (**distâncias**) relativas à posição ocupada na memória pela primeira posição do programa, e permanecem relativos até o momento em que o endereço de alocação física na memória se tornar conhecido.

Coleta de Informação sobre os Símbolos

- As tabelas de símbolos são **criadas** durante a execução do **primeiro passo** do montador, e guardam, sobre os rótulos referenciados no programa, informações a serem usadas no segundo passo, tais como:
  - **Nome** do símbolo
  - **Endereço** ou **valor** numérico associado ao símbolo
  - Informação sobre o **tipo de relocação** necessário no caso de alteração do endereço do programa
  - Informação sobre a **acessibilidade ao símbolo** fora do módulo em que ele foi definido
- Há **muitas** formas **alternativas de organização física** para a tabela de símbolos memorizar a coleção de pares conceituais do tipo ***(símbolo-atributos)***: vetores de registros, tabelas bidimensionais, listas ligadas etc.

Aspecto típico de Tabelas de Símbolos

IDENTIFICADOR SIMBÓLICO	VALOR	ENDEREÇO INICIAL	TAMANHO EM BYTES	INFORMAÇÃO DE RELOCAÇÃO
ABCD	-	/0030	1	ABSOLUTO
XYZ	-	/0123	50	ABSOLUTO
A1	-	INDEFINIDO	100	RELOCÁVEL
B	-	/0000	20	ABSOLUTO
-	150	/05B2	1	ABSOLUTO
-	/0123	/05B3	2	ABSOLUTO

Tabela de Mnemônicos

- Esta tabela é essencial para a montagem do código-objeto.
- Cada mnemônico do código simbólico tem associada uma linha desta tabela, contendo:
  - O **mnemônico** simbólico
  - Indicação dos **operandos exigidos** pela instrução
  - Valor numérico **binário** associado a seu **código**
  - **Número de bytes ocupados** pela instrução
  - Classe da instrução – **número e tipo de operandos**

Aspecto típico de Tabelas de Mnemônicos

ESTRUTURA DO CÓDIGO DE MÁQUINA	TIPO	MNEMÔNICO	NOME	VALOR	OPERANDO
0000xxxxxxxxxxx	inst.ref.mem.	JMP	JUMP	-	ENDEREÇO
0001xxxxxxxxxxx	inst.ref.mem.	LDA	LOAD	-	ENDEREÇO
0010xxxxxxxxxxx	inst.ref.mem.	STA	STORE	-	ENDEREÇO
	...	...			
xxxxxxxx	constante-8	K	BYTE	OPERANDO	CONST. BYTE
xxxxxxxxxxxxxxxx	Endereço	ADDR	POINTER	OPERANDO	CONST. ADDR
	...	...			

Pseudo Instruções

- **Pseudo-instruções** são linhas, na linguagem simbólica do programa fonte, que têm aparência muito semelhante à das linhas que representam instruções de máquina, porém não especificam instruções mas fornecem informações de orientação ao montador.
- M**ontadores absolutos** costumam permitir ao programador fornecer ao montador informações através de **pseudo-instruções** tais como as seguintes (naturalmente, os mnemônicos costumam ser escolhidos arbitrariamente):
  - **ORG** – (define nova origem para o código a ser montado em seguida)
  - **BLOC** – (reserva na memória um vetor a ser usado como área de trabalho)
  - **DB, DW, DA** – (preenche uma ou mais posições da área de código ou de dados com um valor fornecido)
  - **EQU** – (define um novo símbolo como sinônimo de outro)
  - **END** – (demarca o final físico do programa fonte)
- **Montadores relocáveis** em geral oferecem, além destas, um repertório mais variado de pseudo-instruções, direcionadas especificamente para os aspectos da programação simbólica relocável.

Tabelas para pseudo instruções

- Normalmente não se costuma ter uma tabela exclusiva de **pseudo instruções (ou *pseudos*)**, pois, devido à similaridade física e de tratamento por parte do montador, os mnemônicos das pseudo instruções **compartilham a tabela de mnemônicos** já apresentada, que contém as representações simbólicas das instruções de máquina.
- No caso particular das **pseudos**, é conveniente que haja na tabela, em alguma das colunas de **atributos** associados ao mnemônico, uma informação de que se trata de uma pseudo instrução, e outra, que indique a qual das pseudo-instruções do repertório esta se refere.

Constantes

- Analogamente, **constantes** que são utilizadas pelo programa que está sendo montado necessitam ser repetidamente processadas pelo montador, ou então, tabeladas para uma consulta mais eficiente durante o trabalho de montagem.
- Assim, as **constantes** podem também **partilhar**, com os demais nomes simbólicos utilizados no programa, **espaço na tabela de símbolos**, em cujas colunas devem neste caso ser registrados os **atributos associados** à constante em questão: tipo, valor, endereço na memória, etc.