Introdução à Software Básico: Montadores - parte 2

Departamento de Ciência da Computação
Instituto de Ciências Exatas
Universidade de Brasília

Sumário

Montadores

1. Montador de duas passagem: algoritmo da segunda passagem
2. Montador de passagem única

O Algoritmo de duas passagens

- Fluxo de informações para a geração da tabela de símbolos em um montador de duas passagens

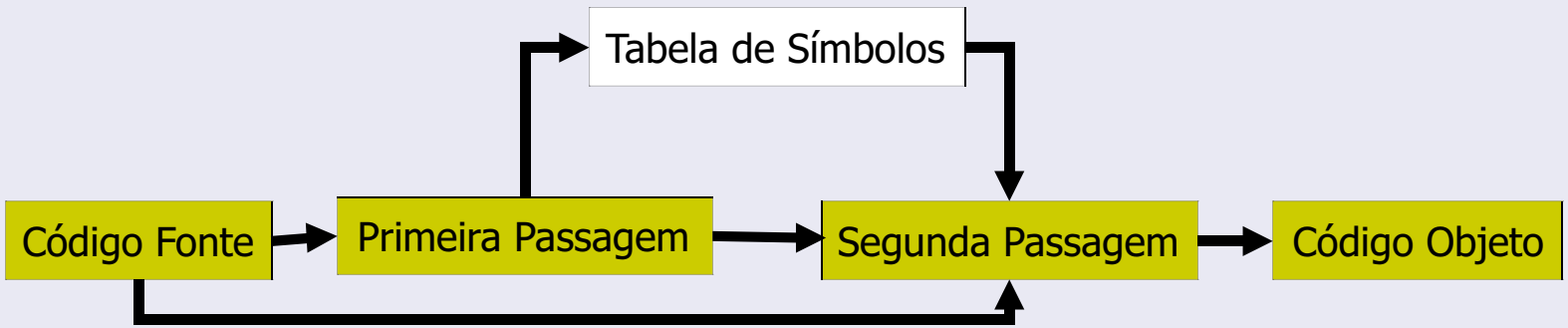


Figura: Fluxo do montador de duas passagens

O Algoritmo de duas passagens

- O processo executado pelo montador em cada passagem é descrito como segue:
- **Primeira Passagem**
  - Na primeira passagem, o montador coleta informações de definições de rótulos, símbolos, etc, e os armazena na tabela símbolos:
    - Símbolo, valor (endereço)
- **Segunda Passagem**
  - Na segunda passagem, os valores (endereços) dos símbolos já são conhecidos e as declarações podem então ser "montadas"

Algoritmo da segunda passagem:

```
Contador_posição = 0
Contador_linha = 1
Enquanto arquivo fonte não chegou ao fim,  faça:
   Obtém uma linha do fonte
   Separa os elementos da linha: rótulo,operação,operandos,comentários
   Ignora o rótulo e os comentários

   Para cada operando que é símbolo
   Procura operando na TS
        Se não achou: Erro, símbolo indefinido
   Procura operação na tabela de instruções
   Se achou:
        contador_posição = contador_posição + tamanho da instrução
        Se número e tipo dos operandos está correto então
                gera código objeto conforme formato da instrução
        Senão: Erro, operando inválido
   Senão:
        Procura operação na tabela de diretivas
        Se achou:
                Chama subrotina que executa a diretiva
                Contador_posição = valor retornado pela subrotina
        Senão: Erro, operação não identificada
   Contador_linha = contador_linha + 1
```

Figura: Algoritmo da segunda passagem

O Algoritmo de duas passagens

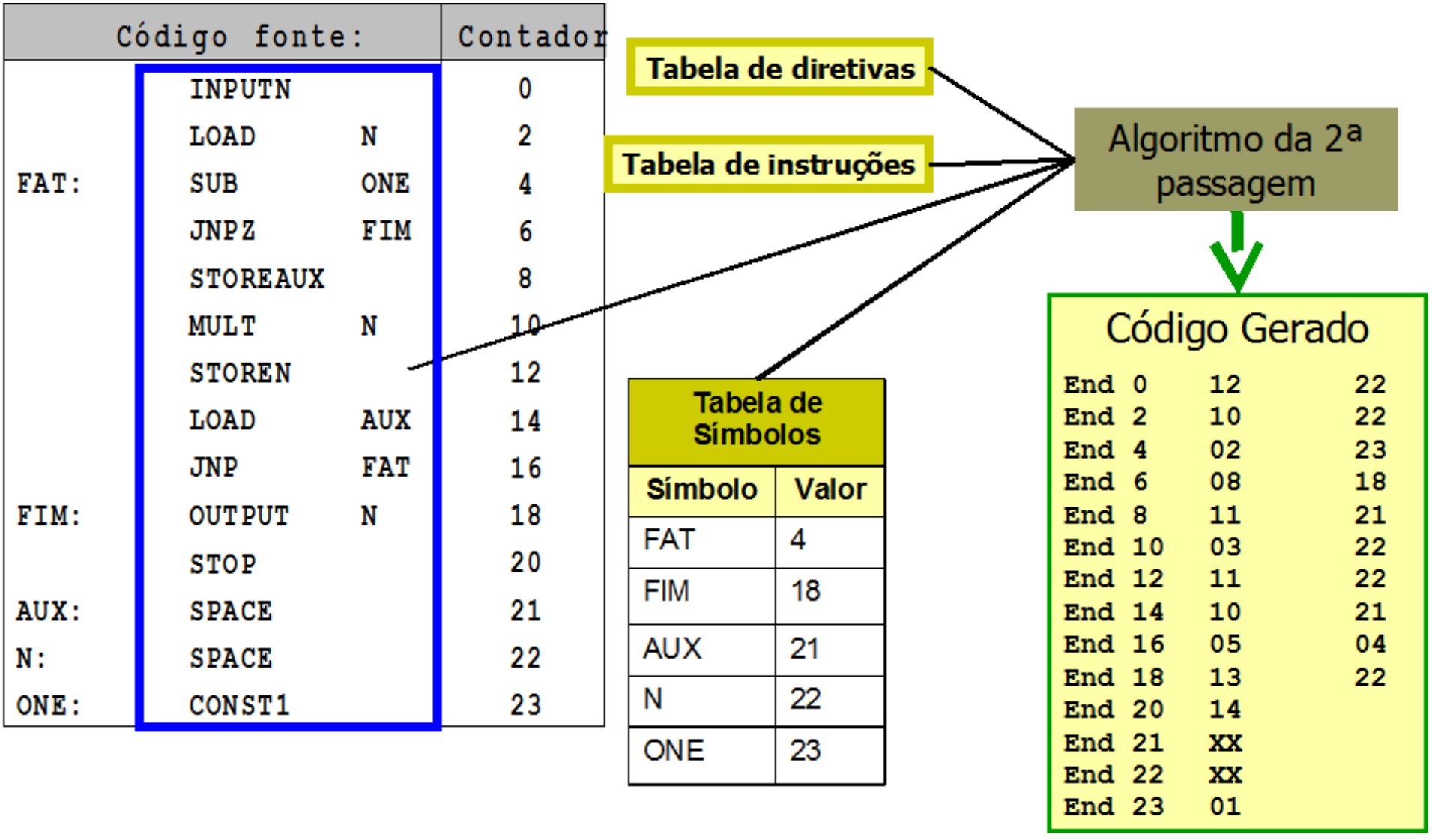


Figura: Algoritmo da segunda passagem

Exemplo

- Gere o código objeto do exercício anterior – série de Fibonacci

```
          COPY      ZERO,     OLDER
          COPY      ONE,      OLD
          INPUT     LIMIT
          OUTPUT    OLD
FRONT:    LOAD      OLDER
          ADD       OLD
          STORE     NEW
          SUB       LIMIT
          JMPP      FINAL
          OUTPUT    NEW
          COPY      OLD,      OLDER
          COPY      NEW,      OLD
          JMP       FRONT
FINAL:    OUTPUT    LIMIT
          STOP
ZERO:     CONST     0
ONE:      CONST     1
OLDER:    SPACE
OLD:      SPACE
NEW:      SPACE
LIMIT:    SPACE
```

TABELA DE SÍMBOLOS

Simbolo	Valor
FRONT	10
FINAL	30
ZERO	33
ONE	34
OLDER	35
OLD	36
NEW	37
LIMIT	38

O Algoritmo de duas passagens

Solução

```
     CÓDIGO GERADO
end.  0: 09 33 35
end.  3: 09 34 36
end.  6: 12 38
end.  8: 13 36
end. 10: 10 35
end. 12: 01 36
end. 14: 11 37
end. 16: 02 38
end. 18: 07 30
end. 20: 13 37
end. 22: 09 36 35
end. 25: 09 37 36
end. 28: 05 10
end. 30: 13 38
end. 32: 14
end. 33: 0
end. 34: 1
end. 35: xx
end. 36: xx
end. 37: xx
end. 38: xx
```

OBS: O valor "xx" representa um valor qualquer. Normalmente zero é usado nestes casos

Observações sobre os endereços das instruções

- O código foi gerado para o endereço zero de memória. Caso fosse necessário gerar o código para outro endereço bastaria alterar o valor inicial do contador de posições.
- Uma solução mais geral é gerar o código sempre para o endereço zero, mas informar também as posições (palavras) do código que contém endereços.
- A indicação das posições que contém endereços é conhecida como **informação de relocação**.
- Nesse caso, a tarefa de acertar os endereços em função do ponto de carga (isto é, a relocação do programa) fica para o **carregador**.

O Algoritmo de uma passagem

- O algoritmo de duas passagens soluciona o problema de referências posteriores de forma bastante simples
- Se todos os símbolos referidos na linha lida (instrução simbólica) já estão definidos, então a instrução de máquina é gerada diretamente, de forma completa
- Obviamente, um montador de um único passo nem sempre poderá determinar o endereço de um símbolo assim que encontrar o mesmo.
- **Como criar então um montador de uma única passagem?**

O Algoritmo de uma passagem

Como criar então um montador de uma única passagem?

- Um símbolo ainda não definido é inserido na tabela de símbolos como havíamos feito até agora
- A instrução de máquina também é gerada.
- Porém, o campo correspondente ao símbolo indefinido fica para ser preenchido **mais tarde**.

Análise das referências posteriores

- Quando uma referência é feita a um rótulo, o montador irá procurar o símbolo da tabela de símbolo da mesma forma que fizemos antes. Aqui, temos várias possibilidades.
  - Se todos os símbolos referidos na linha lida (instrução simbólica) já estão definidos, então a instrução de máquina é gerada diretamente, de forma completa. Neste caso o símbolo estará marcado como "definido=true" na Tabela de Símbolos (TS) e o valor (endereço) já é conhecido.
  - Se o símbolo não estiver na TS, então inserimos o mesmo e marcamos "definido = false". Um campo é criado na TS para apontar para uma lista de símbolos indefinidos.

O Algoritmo de uma passagem

Análise das referências posteriores

- Se o símbolo encontrado já estiver na TS mas ainda não estiver definido, então ele é inserido na lista especifica do símbolo e "ligado" com os itens que já se encontram na lista.
- Quando, finalmente, o símbolo aparece como rótulo, o seu valor (que é dado pelo valor do location counter) é usado para preencher os campos especificados na lista.
- Por último, esse valor é colocado na TS, com a indicação de "símbolo definido". Daqui para a frente, novas ocorrências do símbolo serão substituídas diretamente pelo valor que está na TS

O Algoritmo de uma passagem

Tabela de Símbolos			
Símbolo	Valor	Def	Lista
N	22	T	&
FAT	4	T	-1
ONE	23	T	&
FIM	18	T	&
AUX	21	T	&

Código fonte:			Contador
	INPUT	N	0
	LOAD	N	2
FAT:	SUB	ONE	4
	JNPZ	FIM	6
	STORE	AUX	8
	MULT	N	10
	STORE	N	12
	LOAD	AUX	14
	JNP	FAT	16
FIM:	OUTPUT	N	18
	STOP		20
AUX:	SPACE		21
N:	SPACE		22
ONE:	CONST	1	23

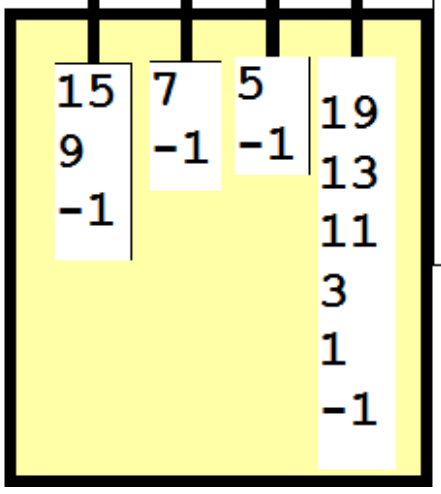


Figura: montador de passagem única

O Algoritmo de uma passagem

Tabela de Símbolos			
Símbolo	Valor	Def	Lista
N	22	T	&
FAT	4	T	-1
ONE	23	T	&
FIM	18	T	&
AUX	21	T	&

Código fonte:			Contador
	INPUT	N	0
	LOAD	N	2
FAT:	SUB	ONE	4
	JNPZ	FIM	6
	STORE	AUX	8
	MULT	N	10
	STORE	N	12
	LOAD	AUX	14
	JNP	FAT	16
FIM:	OUTPUT	N	18
	STOP		20
AUX:	SPACE		21
N:	SPACE		22
ONE:	CONST	1	23

15
9
-1

7
-1

5
-1

19
13
11
3
1
-1

Código Gerado
Em memória

End	0	12	22
End	2	10	22
End	4	02	-
End	6	08	-
End	8	11	-
End	10	03	22
End	12	11	22
End	14	10	22
End	16	05	4
End	18	13	-
End	20	14	22
End	21	XX	
End	22	XX	
End	23	01	

Figura: montador de passagem única

O Algoritmo de uma passagem

Exemplo

- Utilize o algoritmo de uma passagem para gerar a tabela de símbolos para o código objeto do programa mostrado abaixo (calcula area triângulo)

```
INPUT     B
INPUT     H
LOAD      B
MULT      H
DIV       DOIS
STORE     R
OUTPUT    R
STOP
B:        SPACE
H:        SPACE
R:        SPACE
DOIS:     CONST   2
```

O Algoritmo de uma passagem

Solução

```
INPUT    B       0
INPUT    H       2
LOAD     B       4
MULT     H       6
DIV      DOIS    8
STORE    R       10
OUTPUT   R       12
STOP             14
B:       SPACE   15
H:       SPACE   16
R:       SPACE   17
DOIS:    CONST 2 18
```

Tabela de Símbolos

Símbolo	Valor	Def	Lista
B	15	T	→5→1
H	16	T	→7→3
R	17	T	→13→11
DOIS	18	T	→9

O Algoritmo de uma passagem

Observação

- O uso de uma lista encadeada para resolver o problema de referências posteriores pode ser feita de forma mais "econômica" em termos de memória.
- Podemos armazenar a lista dentro do próprio código a ser gerado

O Algoritmo de uma passagem

Exemplo

- Vamos utilizar o mesmo código para gerar a lista no exemplo abaixo.

Código fonte:			Contador
	INPUT	N	0
	LOAD	N	2
FAT:	SUB	ONE	4
	JNPZ	FIM	6
	STORE	AUX	8
	MULT	N	10
	STORE	N	12
	LOAD	AUX	14
	JNP	FAT	16
FIM:	OUTPUT	N	18
	STOP		20
AUX:	SPACE		21
N:	SPACE		22
ONE:	CONST	1	23

```
End  0    12  -
End  2    10  -
End  4    02  -
End  6    08  -
End  8    11  -
End 10    03  -
End 12    11  -
End 14    10  -
End 16    05  4
End 18    13  -
End 20    14
End 21    XX
End 22    XX
End 23    01
```

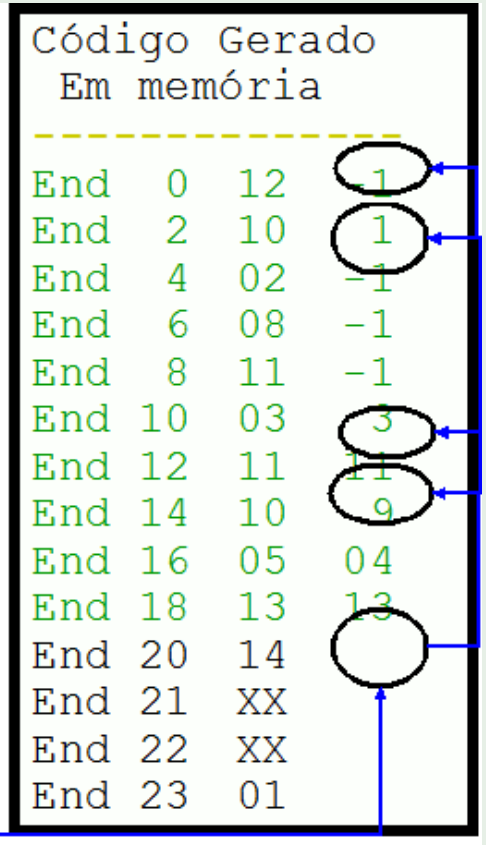
O Algoritmo de uma passagem

Exemplo

Código fonte:			Contador
	INPUT	N	0
	LOAD	N	2
FAT:	SUB	ONE	4
	JNPZ	FIM	6
	STORE	AUX	8
	MULT	N	10
	STORE	N	12
	LOAD	AUX	14
	JNP	FAT	16
FIM:	OUTPUT	N	18
	STOP		20
AUX:	SPACE		21
N:	SPACE		22
ONE:	CONST	1	23

Tabela de Símbolos

Símbolo	Valor	Def	Lista
N	22	T	19
FAT	4	T	-1
ONE	23	T	5
FIM	18	T	7
AUX	21	T	15

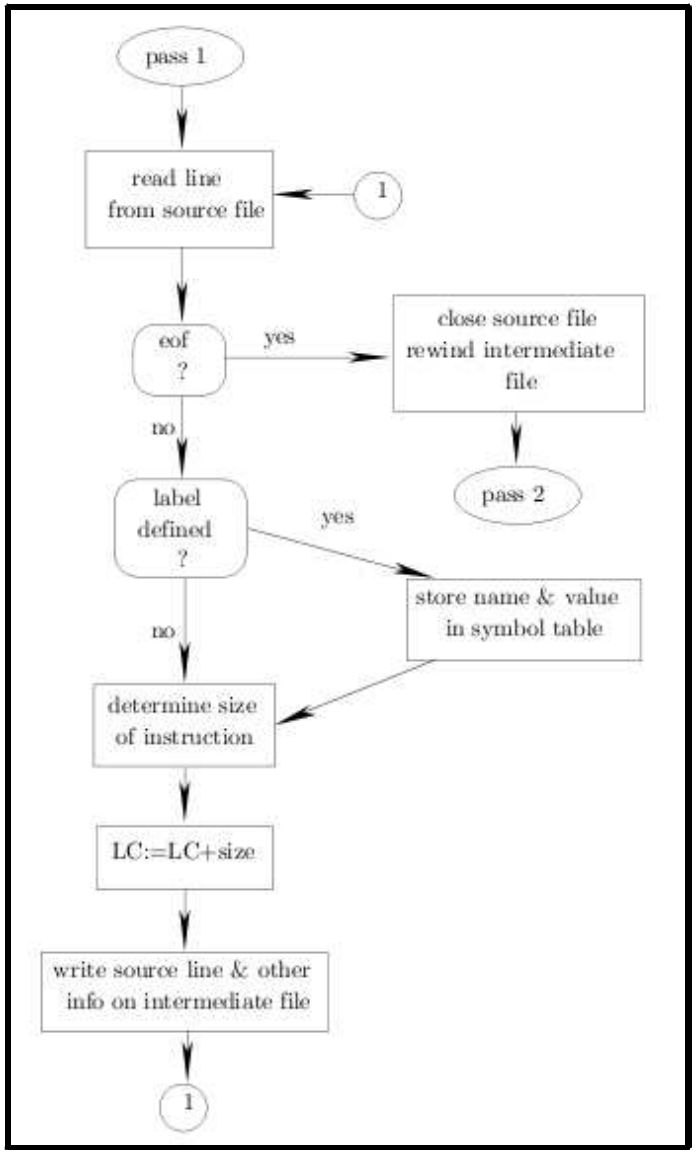


O Algoritmo de uma passagem

Vantagens/Desvantagens

- O algoritmo de duas passagems é mais simples de implementar e requer menos memória do algoritmo de passagem única.
- O algoritmo de passagem única economiza uma leitura completa do arquivo fonte.

Fluxogramas – Montador de 2 passagens



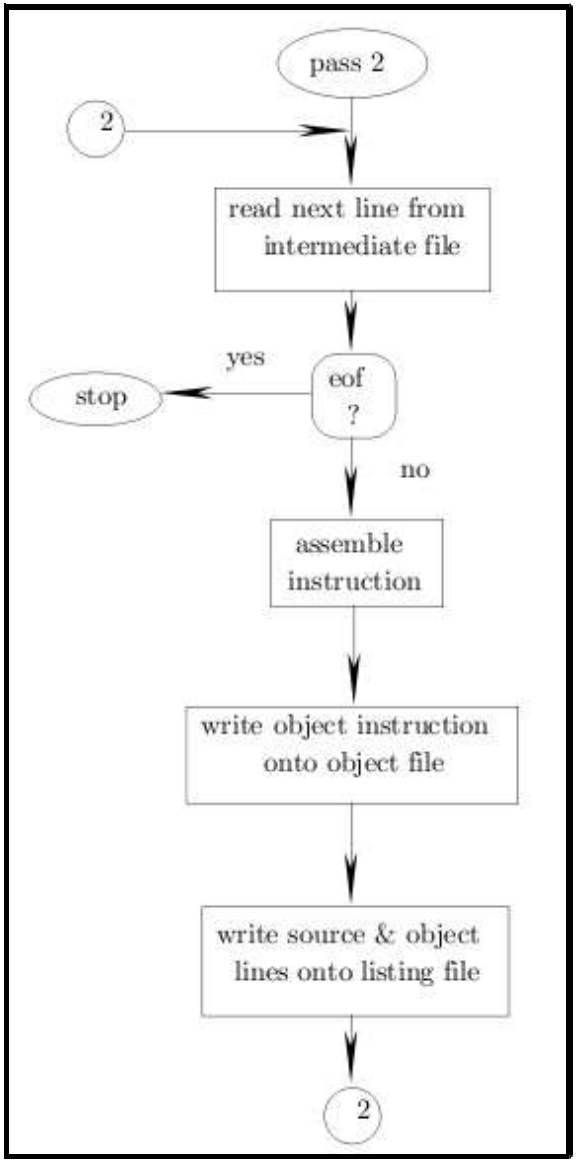


Figura: Fluxograma montador de duas passagens

Fluxogramas – Montador de passagem única

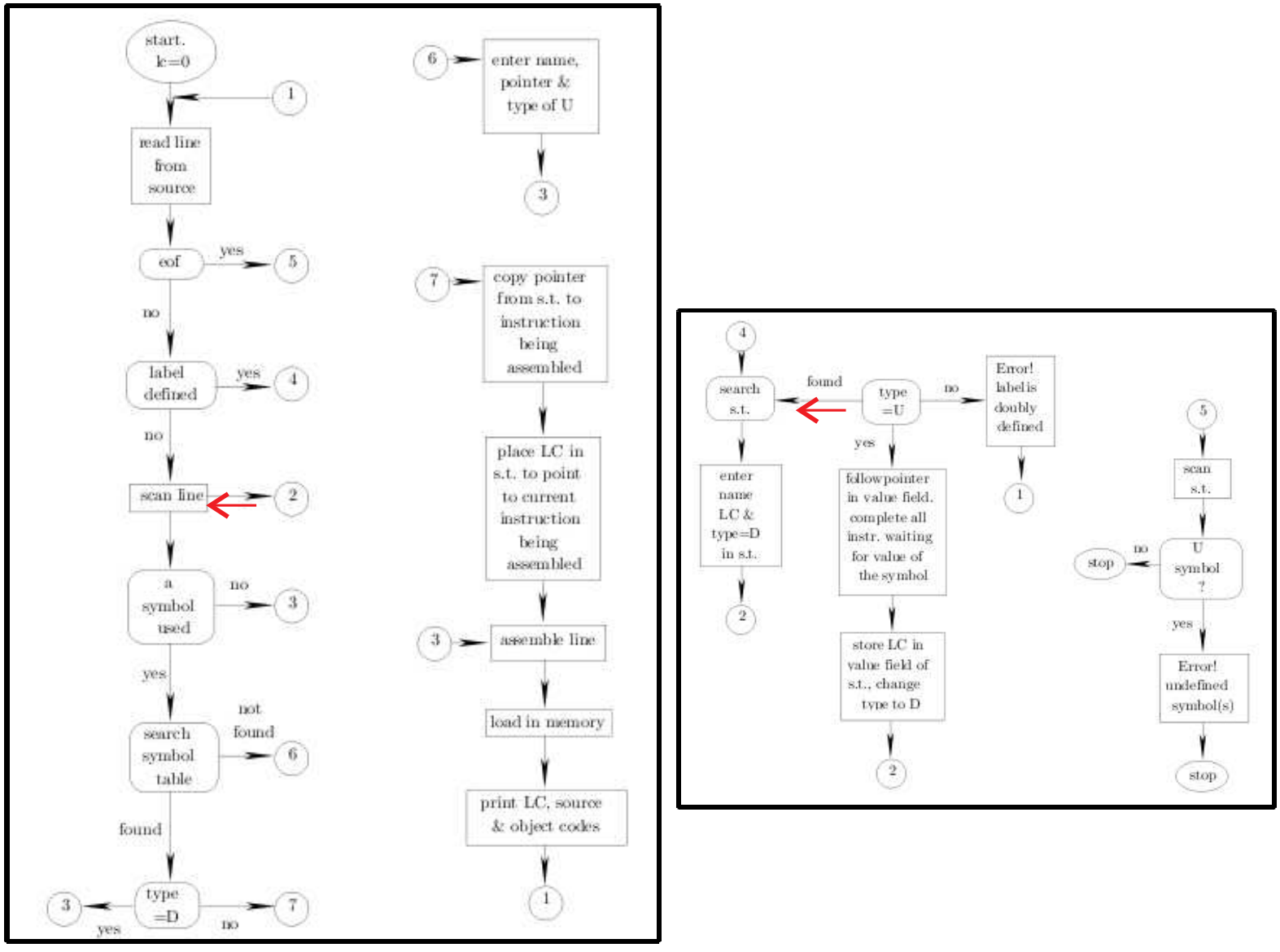


Figura: Fluxograma montador de duas passagens

Próxima Aula

Montador avançado