

decimal

```
  4567
*  366
-----
 27402
 27402
13701
-----
1671522
```

hex

```
  11D7
*  16E
-----
  F9C2
 680A
 11D7
-----
198162
```

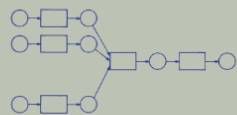
binary

```
  1000111010111
*  0000101101110
-----
  1000111010111
  1000111010111
  1000111010111
  1000111010111
  1000111010111
  1000111010111
  1000111010111
  1000111010111
  1000111010111
  1000111010111
-----
1000111010111
110011000000101100010
```

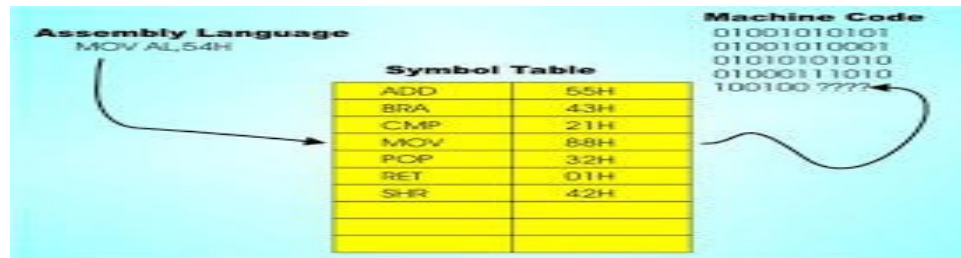
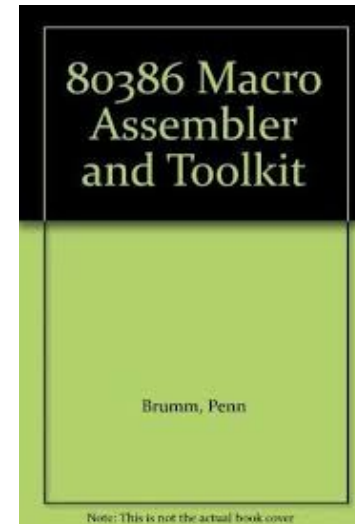
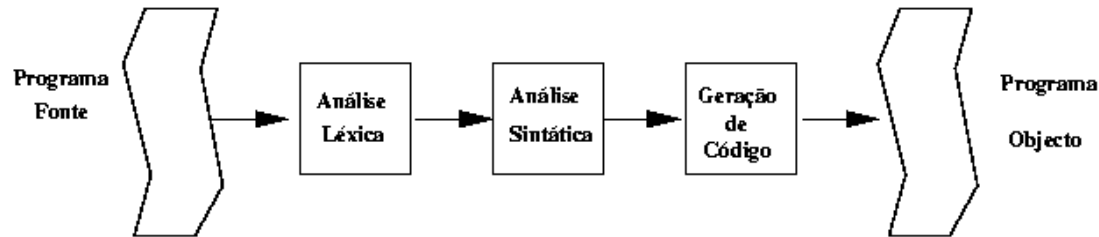
Multiplication examples

ASSEMBLERS, COMPILERS, AND PROGRAM TRANSLATION

PETER CALINGAERT



COMPUTER SCIENCE PRESS



Professor: Me. Anderson Priebe Ferrugem

Computação UFPEL

ferrugem@inf.ufpel.edu.br

Programação de Sistemas Processamento de Macros

Processamento de macros



**Prof.: Anderson Priebe
Ferrugem**

**Cursos:
Ciência da Computação
Engenharia de Computação**

**Centro de Desenvolvimento
Tecnológico
Universidade Federal de
Pelotas**

Sumário

Introdução

Processamento/Expansão Macros

Sintaxe das Macros

Implementações

Duas Passagens

Uma Passagem

Definições Aninhadas

Chamadas Aninhadas

Introdução

Modelos de Sub-rotinas

Sub-rotina Fechada (*closed subroutine*)

Sub-rotina Aberta (*open subroutine*)

Sub-rotina Fechada

Caracterizada pela incorporação de uma única simples cópia do seu texto no programa completo

Proporciona uma redução do tamanho do programa

Reduz também o esforço de escrita (um comando de chamada é uma abreviação da sub-rotina completa)

Espaço e esforço X Tempo

Introdução

Sub-rotina Aberta

"In line code"

Caracterizada pela transcrição de uma cópia da sub-rotina no lugar de cada chamada de uma sub-rotina fechada no programa

Múltiplas cópias de uma parte do programa são incorporadas no programa, ao invés de uma única simples cópia

Alternativa para custo de espaço e esforço

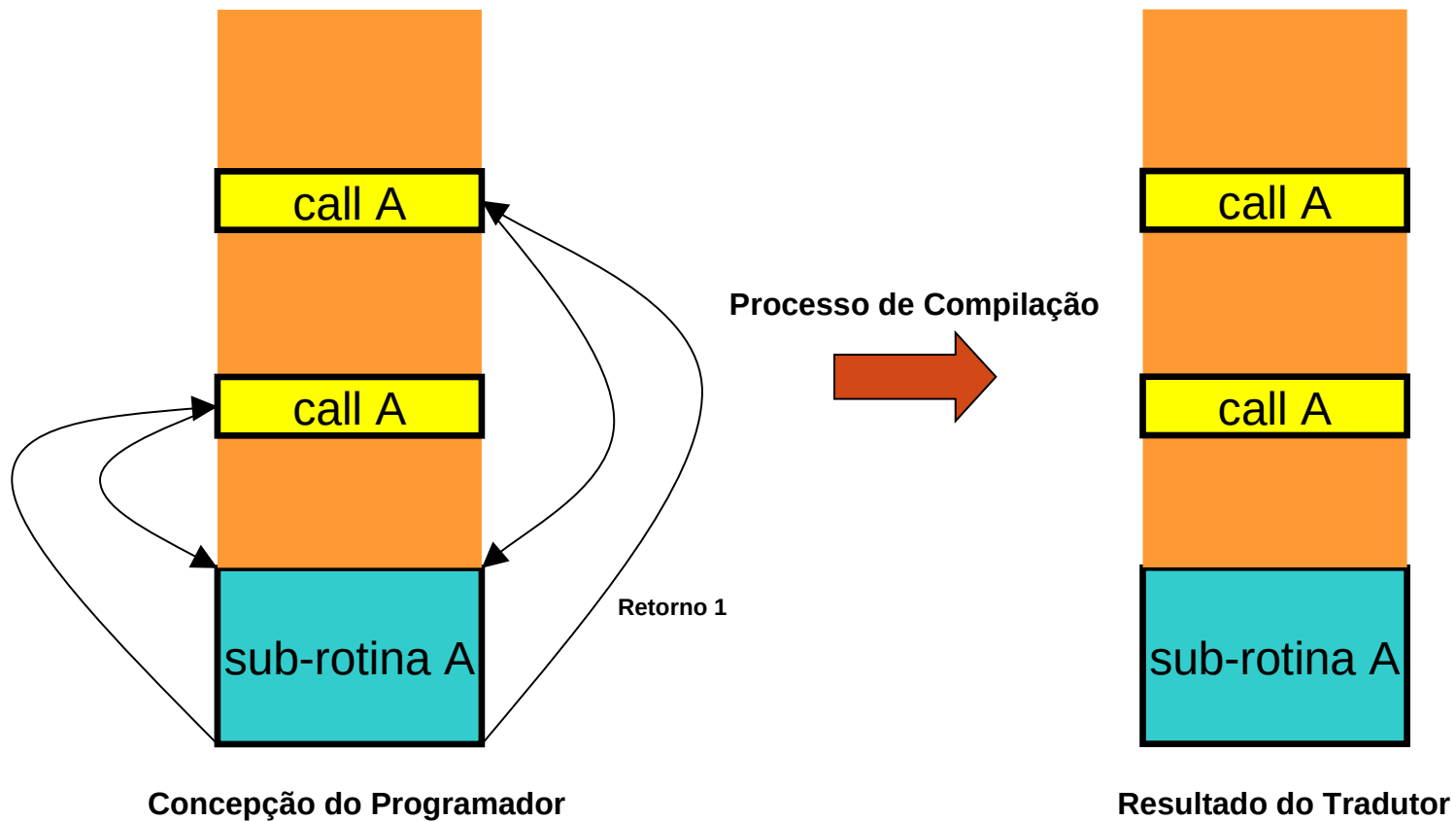
Chamada de **MACRO**

Sub-rotina Fechada x Aberta

Chamada x Cópia da sub-rotina

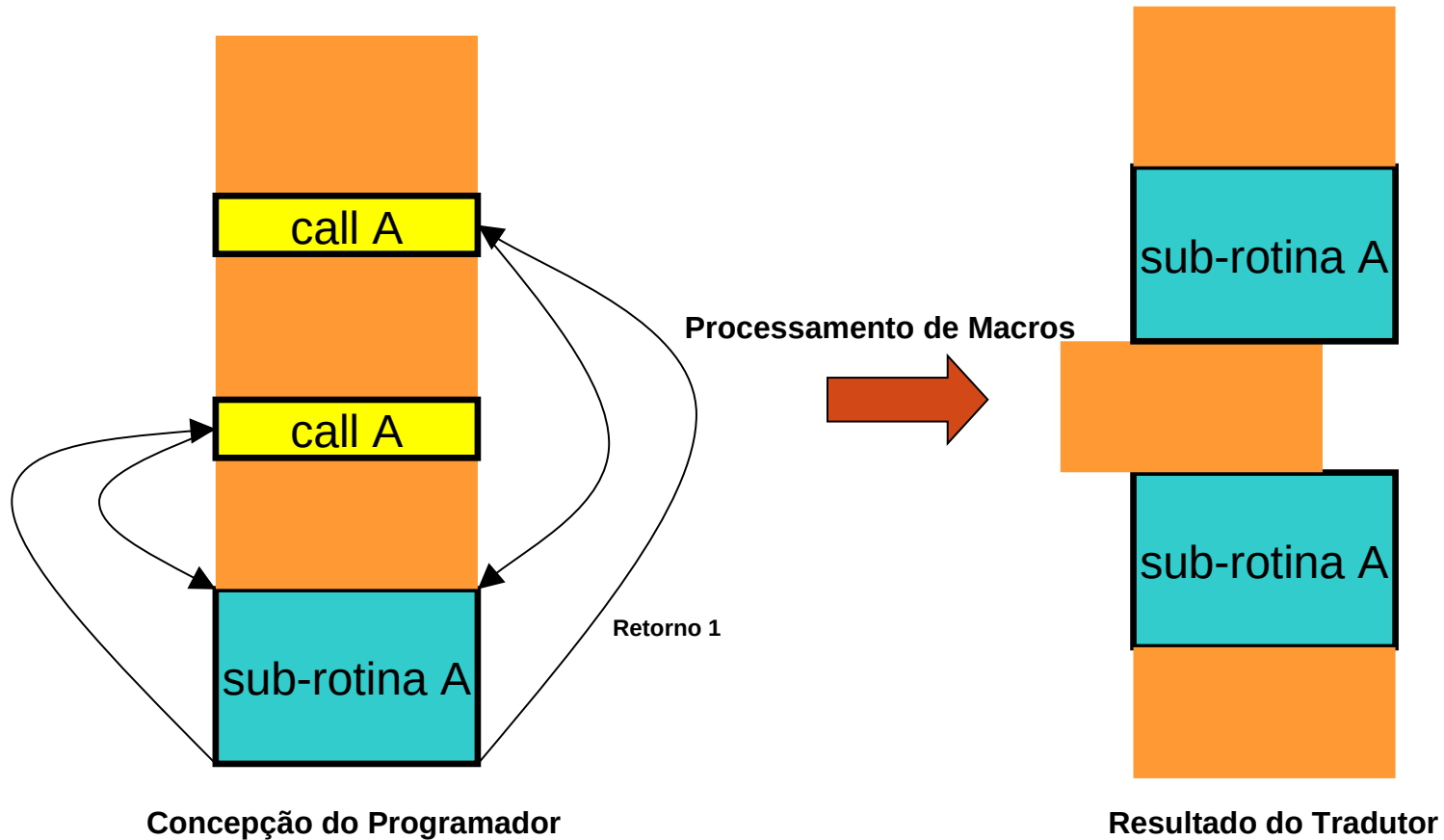
Introdução

Utilizando Sub-rotina Fechada



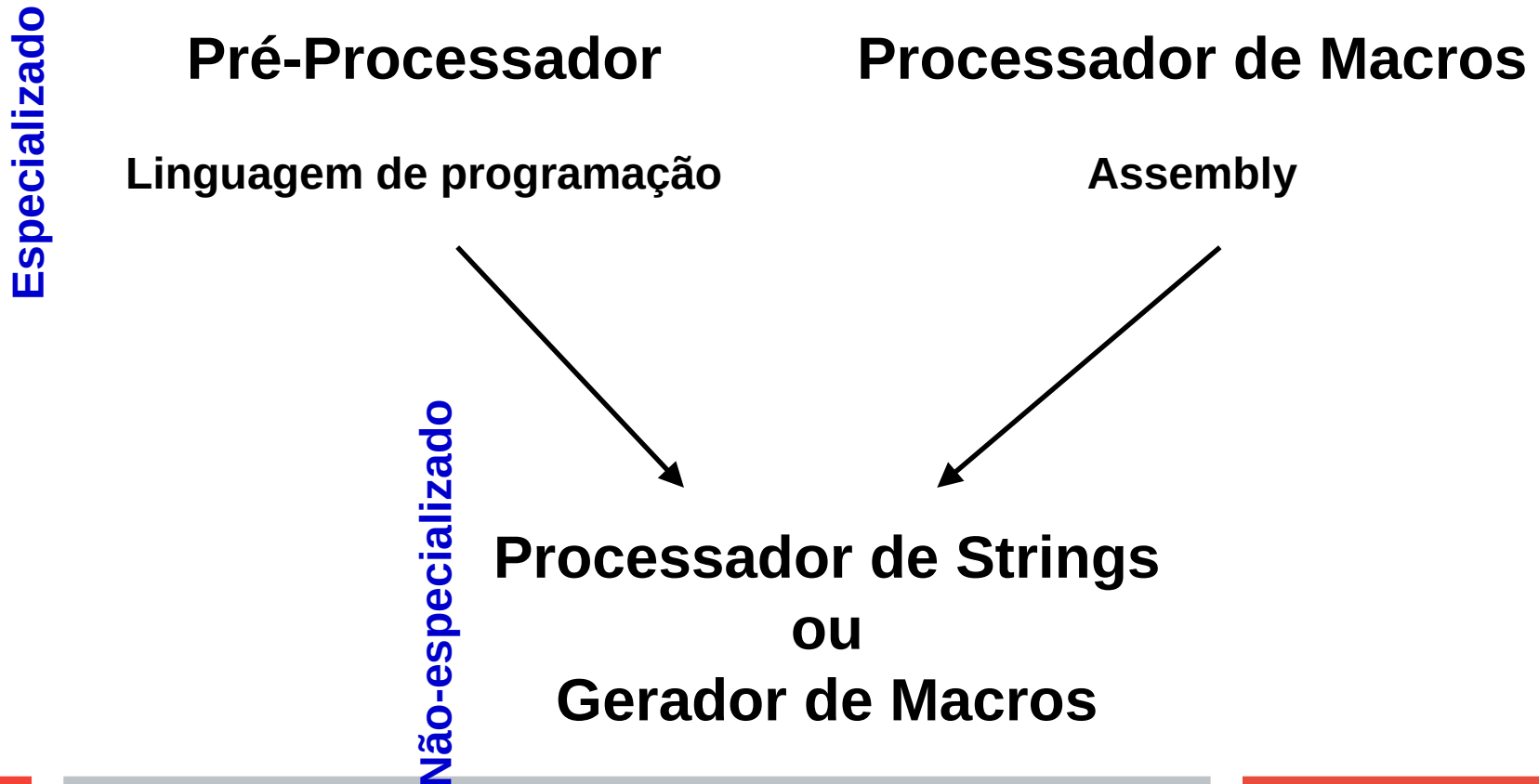
Introdução

Utilizando Sub-rotina Aberta



Introdução

Tratamento de Macros pelos Tradutores



Processamento de Macros

Macros

Grande repetição de grupos de instruções em assembly
Uso de uma única instrução (instrução de macro) que corresponde a uma seqüência de instruções

MACRO = sub-rotina aberta

CHAMADA = instrução de macro

TRADUTOR = processador de macros

Processamento de Macros

Expansão de Macros

Substituição de instruções de macro pelo esqueleto de uma macro com a substituição dos parâmetros pelos valores atuais

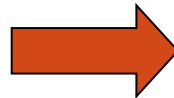
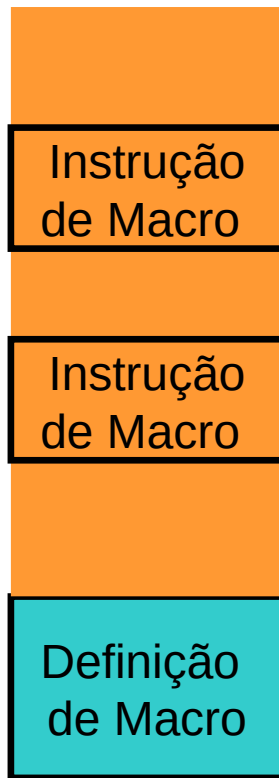
Exemplos

Chamadas para rotinas com serviços de sistema como entrada e saída, sincronização de processos, ...

Macros de sistema (disponíveis em biblioteca)

Expansão de Macros

**Programa com as
definições de macro**



Programa Expandido

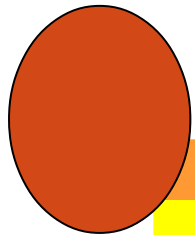


**Utilizado como fonte
em outro tradutor**



Sintaxe das Macros

Definição da Macro e chamada (versão 1)



INCR	MCDEFN	L, A
L	LOAD	A
	ADD	@1
	STORE	A
	MCEND	

← Delimitador & Protótipo

← Esqueleto

← Delimitador

INCR MARK, COUNT ← Chamada

Sintaxe das Macros

Definição da Macro e chamada (versão 2)

MCDEFN			← Delimitador
L	INCR	A	← Protótipo
L	LOAD	A	← Esqueleto
	ADD	@1	
	STORE	A	
MCEND			← Delimitador

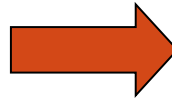
MARK INCR COUNT ← Chamada

Sintaxe das Macros

Exemplo de Expansão de Macro

Chamada

MARK INCR COUNT



Macro Expandida

MARK LOAD COUNT
 ADD @1
 STORE COUNT

Definição da Macro

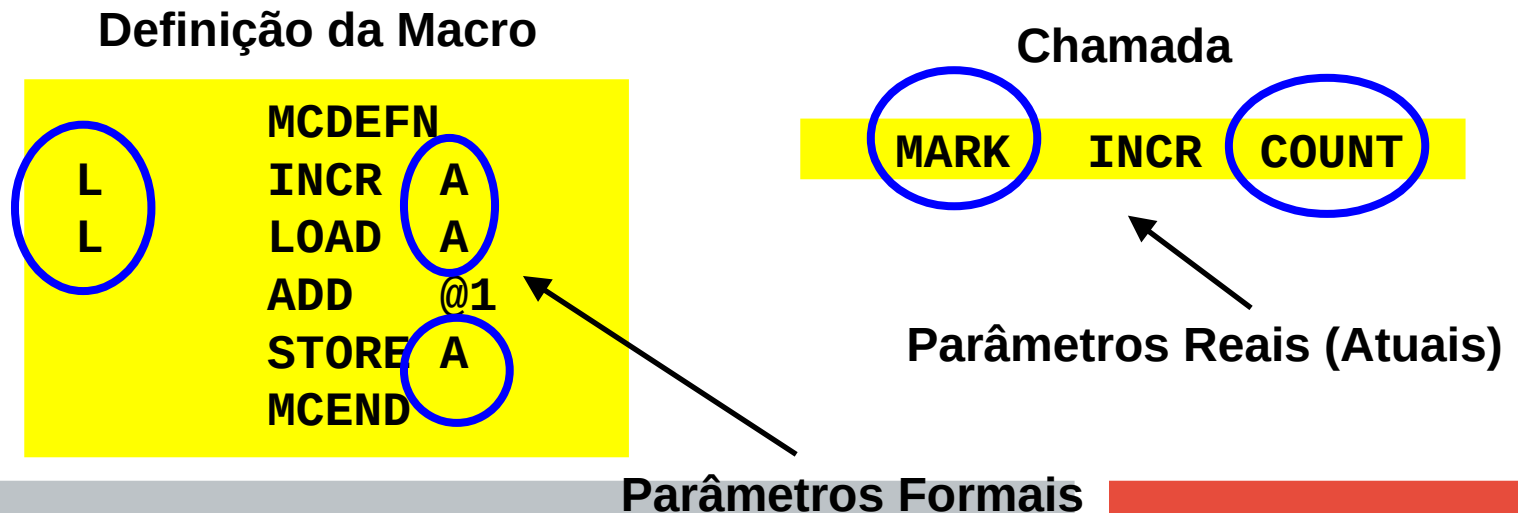
MCDEFN
L INCR A
L LOAD A
 ADD @1
 STORE A
MCEND

Sintaxe das Macros

Sintaxe dos Parâmetros

Parâmetros Formais: parâmetros declarados na definição (protótipo) e usados no esqueleto

Parâmetros Reais (atuais): contidos na chamada da macro



Sintaxe das Macros

Sintaxe dos Parâmetros

Durante a expansão: os parâmetros reais substituem os parâmetros formais

A substituição dos parâmetros formais pelos valores atuais depende da chamada e é realizada separadamente para cada chamada

Duas abordagens sintáticas para os parâmetros:

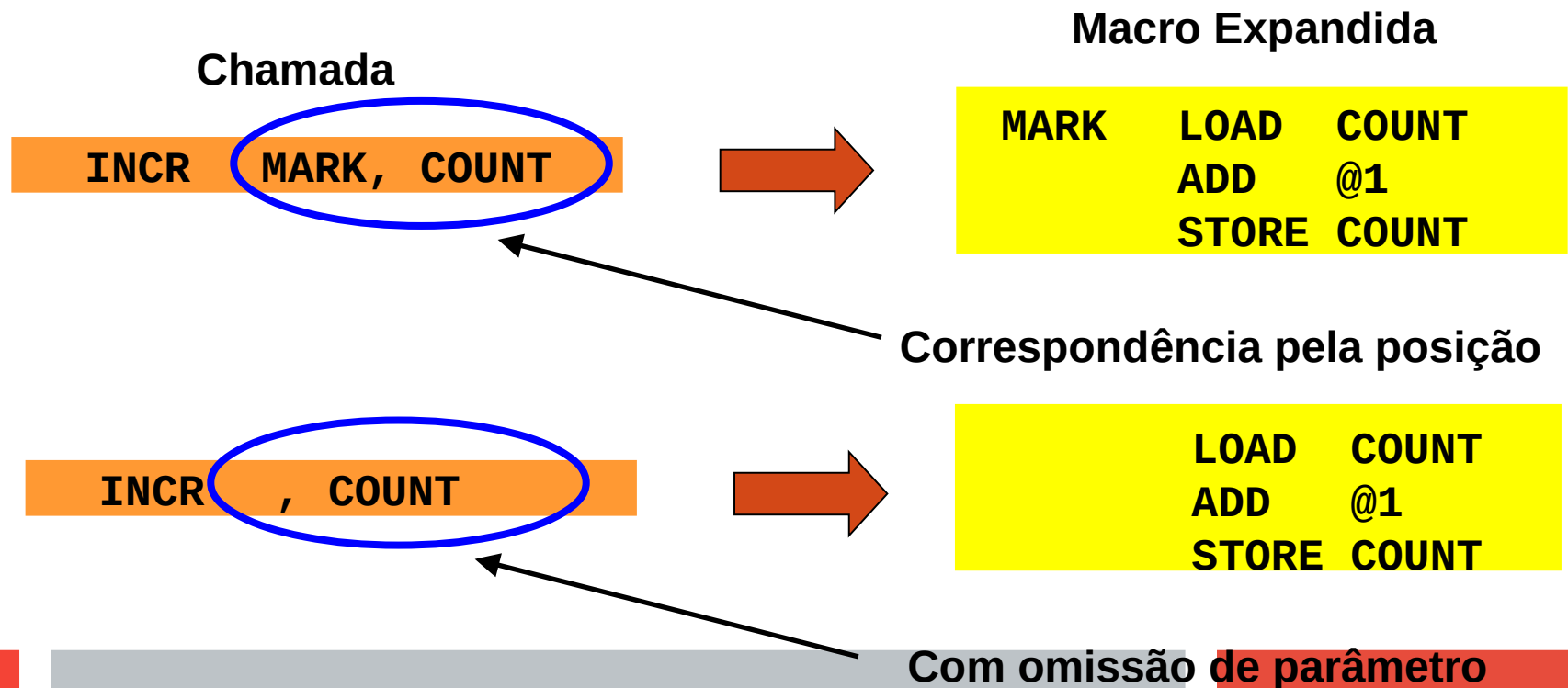
Posicional

Palavra-chave

Sintaxe das Macros +++

Sintaxe dos Parâmetros

Posicional: os parâmetros são distinguidos pela posição que ocupam na lista de parâmetros



Sintaxe das Macros +++

Sintaxe dos Parâmetros: Posicional

Exemplo: Assembler do IBM 360-370

Alternativa no IBM 360-370

Correspondência posicional sem o uso de protótipo

Forma sintática: &SYSLIST(*i*), onde *i* é o *i*-ésimo operando da chamada

Sintaxe das Macros +++

Sintaxe dos Parâmetros

Palavra-chave: os parâmetros reais devem possuir correspondência com os parâmetros formais por identificação

Sintaxe: *formal = actual*

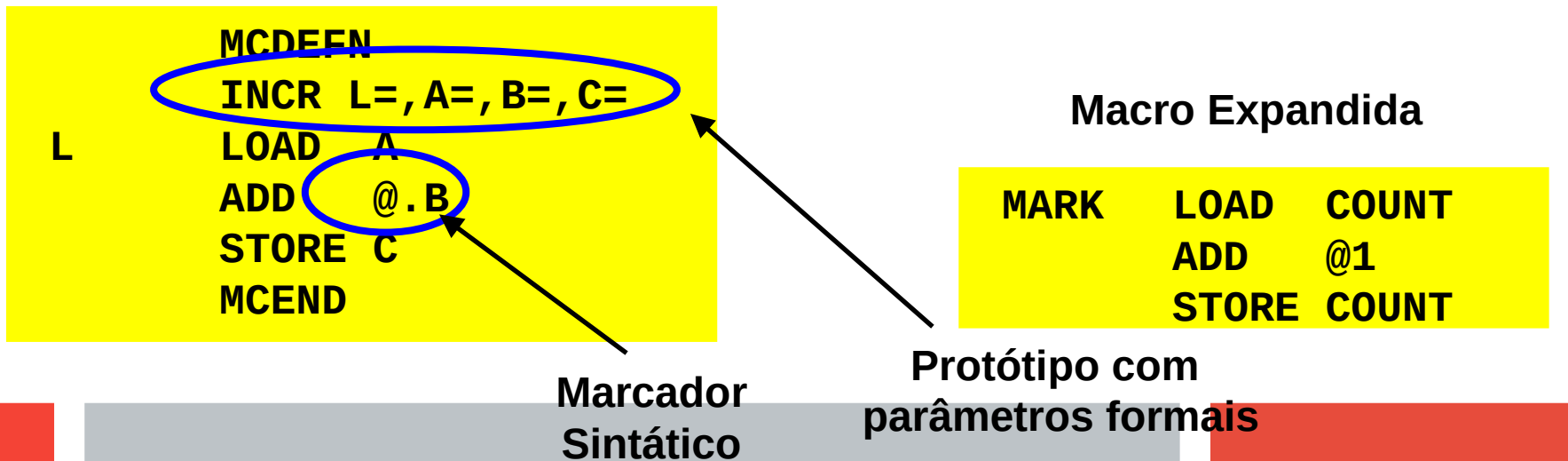
Chamada

INCR L=MARK, A=COUNT, B=1, C=COUNT

Definição da Macro

L

```
MCDEFN
  INCR L=, A=, B=, C=
  LOAD A
  ADD @.B
  STORE C
MCEND
```



Macro Expandida

```
MARK    LOAD    COUNT
         ADD     @1
         STORE   COUNT
```

Marcador
Sintático

Protótipo com
parâmetros formais


Sintaxe das Macros

Sintaxe dos Parâmetros

Palavra-chave: pode ser estabelecido apenas pelo protótipo ou no próprio esqueleto (por exemplo utilizando o marcador sintático '=')

Definição da Macro

```
MCDEFN
INCR L=, A=, B=, C=
L=    LOAD  A=
      ADD   @. B=
      STORE C=
MCEND
```



Sintaxe das Macros +++ mostrar expansão

Sintaxe dos Parâmetros

Palavra-chave: omissão de parâmetros e uso de valores *default* (por falta)

Definição da Macro com Valores Default

```
MCDEFN
  INCR L=, A=ACCUM, B=1, C=ACCUM
L      LOAD  A
      ADD   @.B
      STORE C
      MCEND
```

Sintaxe das Macros +++

Suporte a rótulos (*labels*)

Caso ocorra um label não definido como parâmetro, as diferentes expansões provocarão múltiplas ocorrências de um símbolo

Solução: cada chamada de macro deverá gerar um ocorrência distinta do rótulo

Solução 1

definidos como parâmetros

+++

Solução 2.1

Contador com número de vezes que a macro é expandida
O contador é concatenado com os rótulos gerados na forma de string (a concatenação pode ser realizada de forma automática)

Sintaxe das Macros +++

Solução 2.2

Utilizar um parâmetro formal definido pelo sistema (**SER**)

Exemplo:

MASS.SER é um label na macro REVISE

Primeira expansão: MASS0023

Segunda expansão: MASS0036

Considerando que são a 23ª e 26ª expansão do processador de macros

Contador Global x Contador Local +++

Implementações

Implementação de duas Passagens

Algoritmo definido em três etapas (modos)

Modo Normal (Modo de Cópia)

Modo de Definição

Modo de Expansão

Implementação de uma Passagem

Algoritmo também definido em três etapas (modos)

Modo Normal (Modo de Cópia)

Modo de Definição

Modo de Expansão

Implementação de duas passagens

Implementação de duas passagens

É possível se cada macro é definida uma única vez

Hipótese proposta:

Sem definição de macro dentro de macro

Sem chamada de macro dentro de macro

Definida uma única vez

Proposta de duas passagens

Primeira passagem: coleta as definições

Segunda passagem: expande as chamadas

Implementação de duas passagens

Estruturas de Dados

Tabela de Definição de Macros

Tabela que mantém as definições das macros

Texto de Entrada

```
MCDEFN
SPOT  ABSDIF A, B, C
SPOT  LOAD  A
      SUB  B
      BRPOS ST.SER
      LOAD B
      SUB  A
ST.SER STORE C
MCEND
```

Tabela de Definição de Macros

```
SPOT  ABSDIF A, B, C
#0    LOAD  #1
      SUB  #2
      BRPOS ST.SER
      LOAD #2
      SUB  #1
ST.SER STORE #3
MCEND
```

Implementação de duas passagens

Estruturas de Dados

Tabela de Nomes de Macro (opcional)

Utilizado para diferenciar chamadas de macro de outro texto na segunda passagem

Lista de Parâmetros Reais (atuais)

Utilizado para preparar a substituição dos parâmetros atuais

Chamada

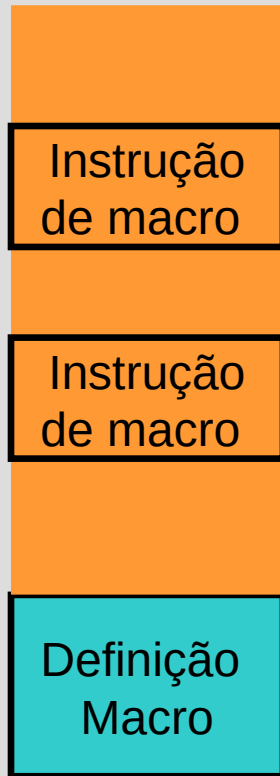
ABSDIF INPRES, OUTPRES, PRESSURE

Lista de Parâmetros Reais

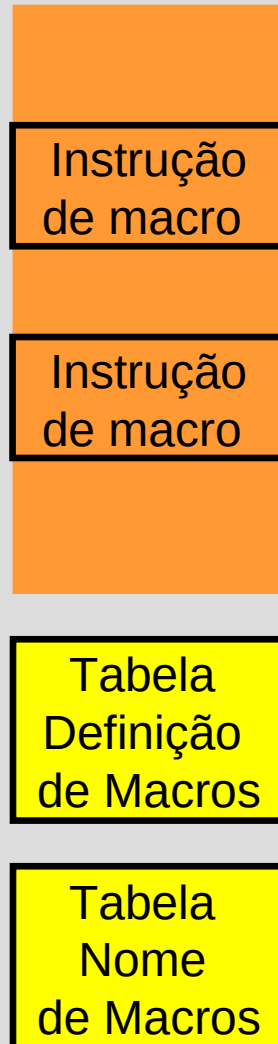
#0	'	'
#1	' INPRES	'
#2	' OUTPRES	'
#3	' PRESSURE'	

Implementação de duas passagens

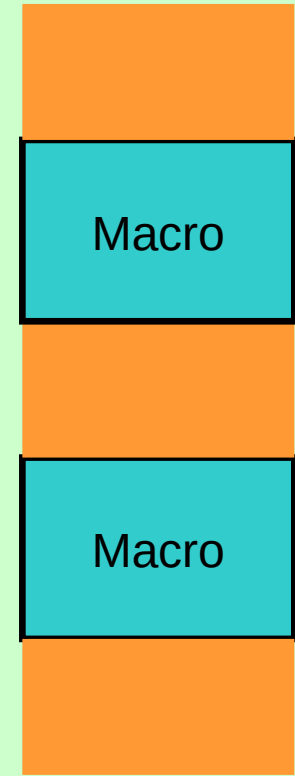
Texto de Entrada



Texto Intermediário



Texto Saída



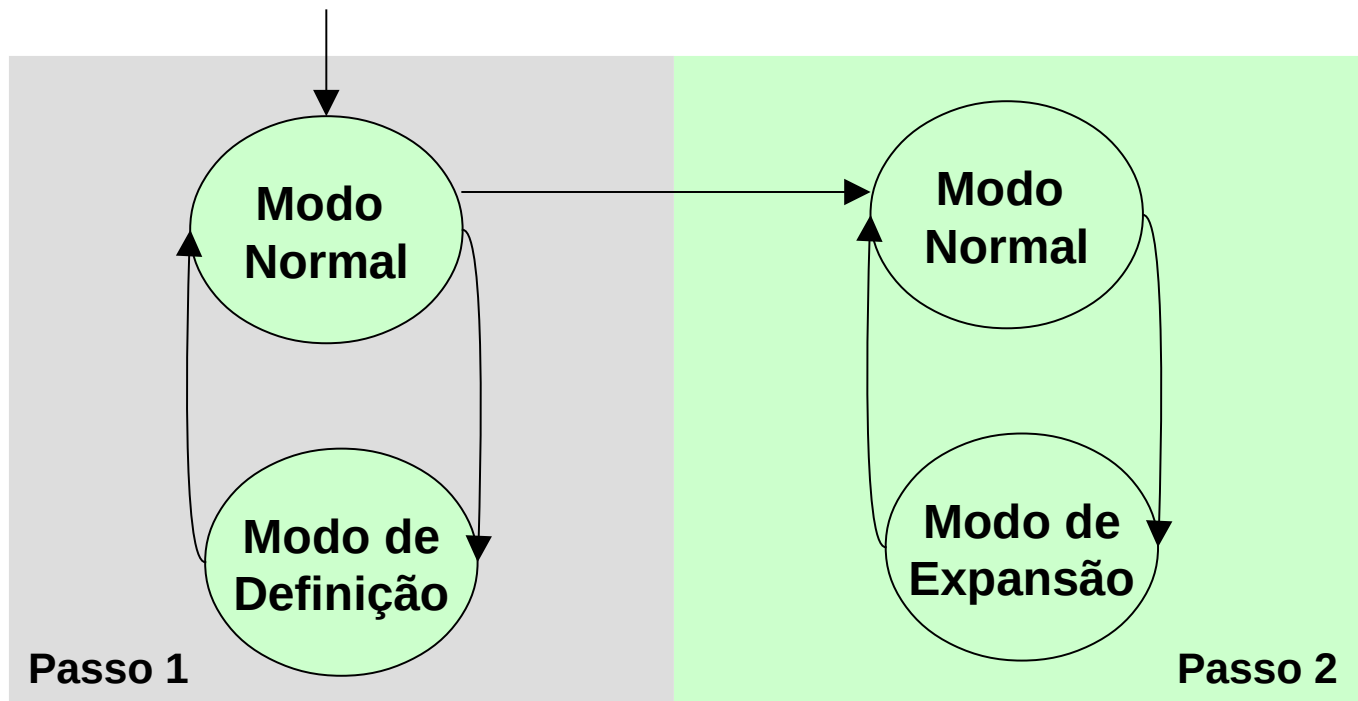
Passo 1

Passo 2

Implementação de duas passagens

Algoritmos

Diagrama de Estados



Implementação de duas passagens

Algoritmo

Passo 1:

Modo Normal: o processador de macros simplesmente copia as instruções do texto de entrada para o texto intermediário até encontrar um delimitador de macro

Modo de Definição: entra quando encontra um "MCDEFN". O processador copia para apenas o texto da definição para a tabela de definição de macros.

Com pré-tratamento dos parâmetros formais (expansão será mais simples). Troca do parâmetro formal pelo número da posição do protótipo)

Sem pré-tratamento (expansão necessitará de mais tarefas)

Quando é encontrado um protótipo o nome da macro poderá ser armazenado na Tabela de Nome de Macros (para uso no Passo 2)

Implementação de duas passagens

Algoritmo

Passo 2:

Utiliza o código intermediário sem definições (gerado pelo Passo 1) mas com chamadas

Modo Normal: o processador de macros simplesmente copia as instruções do texto de intermediário para o texto de saída até encontrar o nome da macro (que pode estar na Tabela de Nomes de Macros) no campo do código da operação e troca para o Modo de Expansão

Modo de Expansão: o processador de macros prepara os parâmetros reais para serem substituídos. O processador de macros copia as instruções do esqueleto para o texto de saída, substituindo cada ocorrência da lista de parâmetros reais e cada ocorrência do .SER pelo número serial da expansão. Quando o processador encontra o delimitador retorna para o modo normal.

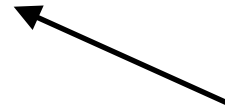
Implementação de uma passagem

Implementação de uma passagens

É possível com pequenas restrições

Hipótese proposta:

Com chamadas aninhadas
Com definições aninhadas
Sem referências avante
Com redefinição de macros



Sem é muito simples !

Modos (algoritmo)

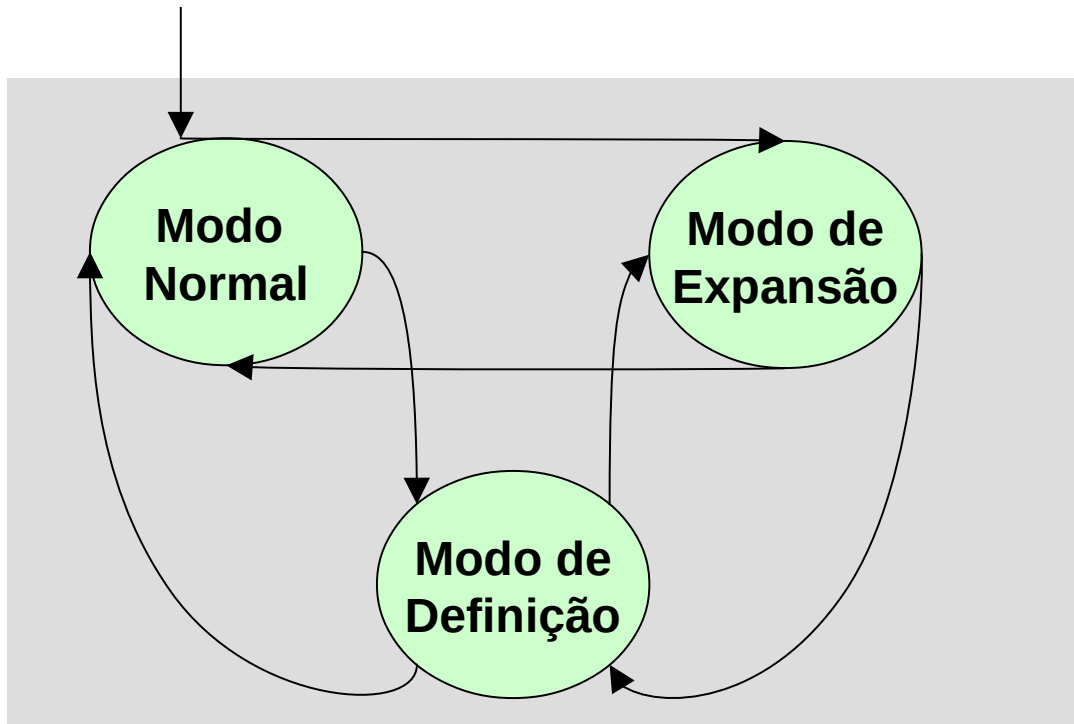
Definição
Expansão
Cópia

Algoritmo da página 82 (Figura 4.5)

Implementação de uma passagem

Algoritmos

Diagrama de Estados



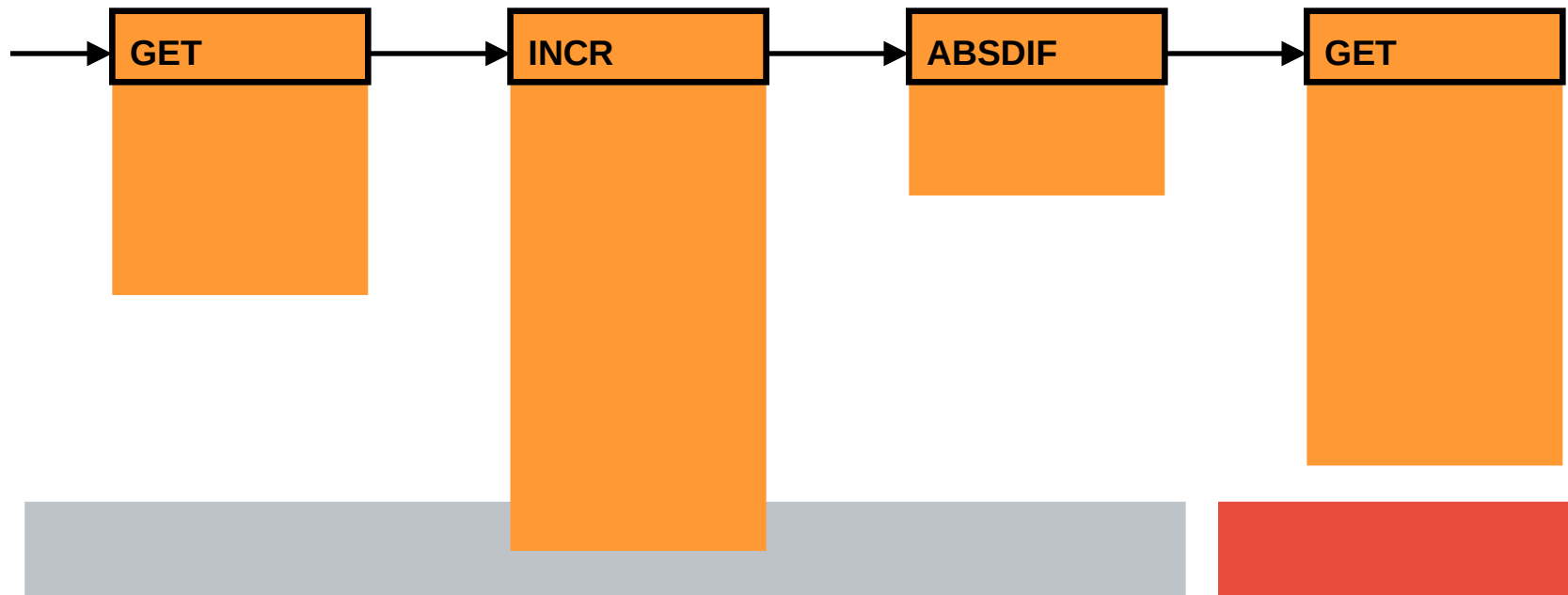
Implementação de uma passagem

Problema

Tabela de definição de macros

Não existe garantia de que uma macro redefinida possa ser maior que a anterior

Solução: definições em cadeia



Definições Aninhadas

Definições aninhadas

O esqueleto de uma macro pode conter a definição de outra macro
A definição da macro A apenas especifica que o texto da em que uma chamada da macro A for expandida inclui a definição da macro B

```
A      MCDEFN  
      - - - - -  
      - - - - -  
B      MCDEFN  
      - - - - -  
      - - - - -  
      MCEND  
      - - - - -  
      - - - - -  
      MCEND
```

Definições Aninhadas

Pode ser implementado com uma passagem

Algoritmo da página 87 (Figura 4.11)

Algoritmo 4.5 com as seguintes modificações:

A saída do modo de definição não é causada por um delimitador

Definition-level counter (contador de nível de definição)

Controle de coincidência de nomes de parâmetros formais

Pilha de parâmetros formais

Substituição de parâmetros pelo par $\#(d, i)$, onde d é o contador de nível e i é a ordem do parâmetro

Definições Aninhadas

**Formal-
parameter
stack**

Input text

Macro-definition table entry

<i>top</i>	MCDEFN		
	X	A, B, C, D	X
G 3, 4		A, B, C, D	#(1, 1), #(1, 2), #(1, 3), #(1, 4)
E 3, 3	MCDEFN		MCDEFN
C 3, 2	Y	A, B, E, F	Y
A 3, 1		A, B, C, D	#(2, 1), #(2, 2), #(1, 3), #(1, 4)
F 2, 4 ←	MCDEFN		MCDEFN
E 2, 3	Z	A, C, E, G	Z
B 2, 2		A, B, C, D	#(3, 1), #(2, 2), #(3, 2), #(1, 4)
A 2, 1		E, F, G, H	#(3, 3), #(2, 4), #(3, 4), H
D 1, 4	MCEND		MCEND
C 1, 3		E, F, G, H	#(2, 3), #(2, 4), G, H
B 1, 2	MCEND		MCEND
A 1, 1		E, F, G, H	E, F, G, H
<i>bottom</i>	MCEND		MCEND

Figure 4.10 Formal Parameters in Nested Definitions

Definições Aninhadas

Exemplo

Line	Input
1	MCDEFN
2	A FORMAL 1
3	LOAD FORMAL 1
4	MCDEFN
5	B FORMAL 2
6	STORE FORMAL 2
7	MCEND
8	MCEND
9	A ACTUAL1
10	B ACTUAL2

```
d - 0 { definition-level  
counter}  
e - false {expansion-mode  
switch}
```

Definições Aninhadas

Line read	<i>d</i>	<i>e</i>	Line written	Macro definitions		Output	
1	0	false					
2	1	false	11	A	FORMAL1		
3	1	false	12	LOAD	#(1,1)		
4	1	false	13	MCDEFN			
5	2	false	14	B	FORMAL2		
6	2	false	15	STORE	#(2,1)		
7	2	false	16	MCEND			
8	1	false	17	MCEND			
9	0	false					
11	0	true					
12	0	true	18			LOAD	ACTUAL1
13	0	true					
14	1	true	19	B	FORMAL2		
15	1	true	20	STORE	#(1,1)		
16	1	true	21	MCEND			
17	0	true					
10	0	false					
19	0	true					
20	0	true	22			STORE	ACTUAL2
21	0	true					

(b) Successive actions

Figure 4.12 Macro Processing Trace

Definições Aninhadas

Exemplo

```
MCDEFN
SCALE RP
MCDEFN
MULTSC A, B, C
LOAD A
MULT B
SHIFTR RP
STORE C
MCEND
MCDEFN
DIVSC A, B, C
LOAD A
MULT B
SHIFTL RP
STORE C
MCEND
MCEND
```

Definições de macro geradas por SCALE 3:

```
MULTSC A, B, C
LOAD A
MULT B
SHIFTR 3
STORE C
MCEND
```

Seqüência do contador de nível:
0, 1, 2, 1, 2, 1, 0.

```
DIVSC A, B, C
LOAD A
MULT B
SHIFTL 3
STORE C
MCEND
```


Definições Aninhadas

Exemplo

```
MCDEFN
SCALE RP
MCDEFN
MULTSC A, B, C
LOAD A
MULT B
SHIFTR RP
STORE C
MCEND
MCDEFN
DIVSC A, B, C
LOAD A
MULT B
SHIFTL RP
STORE C
MCEND
MCEND
```

Definições de macro geradas por SCALE 3:

```
MULTSC A, B, C
LOAD A
MULT B
SHIFTR 3
STORE C
MCEND
```

Seqüência do contador de nível:
0, 1, 2, 1, 2, 1, 0.

```
DIVSC A, B, C
LOAD A
MULT B
SHIFTL 3
STORE C
MCEND
```

Chamadas Aninhadas

Chamadas Aninhadas

O texto da definição de uma macro contém um chamada de macro

Algoritmo da página 93 (Figura 4.18)

```
A      MCDEFN  
      - - - - -  
      - - - - -  
      B  
      - - - - -  
      - - - - -  
      MCEND
```

Chamadas Aninhadas

Exemplo

Com Aninhamento

```
MCDEFN
DISCR  A, B, C, D
MULTSC A, C, TEMP1
MULTSC TEMP1, @4, TEMP1
MULTSC B, B, TEMP2
SUB    TEMP1
STORE  D
MCEND
```

Sem Aninhamento

```
MCDEFN
DISCR  A, B, C, D
LOAD   A
MULT   C
SHIFTR 3
STORE  TEMP1
LOAD   TEMP1
MULT   @4
SHIFTR 3
STORE  TEMP1
LOAD   B
MULT   B
SHIFTR 3
STORE  TEMP2
SUB    TEMP1
STORE  D
MCEND
```

Chamadas Aninhadas

Exemplo

Chamada

DISCR P, Q, R, S

Primeiro nível de expansão

**MULTSC P, R, TEMP1
MULTSC TEMP1, @4, TEMP1
MULTSC Q, Q, TEMP2
SUB TEMP1
STORE S**

Chamadas Aninhadas

Exemplo

Primeiro nível de expansão

```
MULTSC P, R, TEMP1  
MULTSC TEMP1, @4, TEMP1  
MULTSC Q, Q, TEMP2  
SUB TEMP1  
STORE S
```

Segundo nível de expansão

```
LOAD P  
MULT R  
SHIFTR 3  
STORE TEMP1  
LOAD TEMP1  
MULT @4  
SHIFTR 3  
STORE TEMP1  
LOAD Q  
MULT Q  
SHIFTR 3  
STORE TEMP2  
SUB TEMP1  
STORE S
```

Onde aprofundar os estudos ?

CALINGAERT, Peter. Assemblers, Compilers, and Program Translation. Potomac: Computer Science Press, Inc, 1979.

KOLIVER, C. . Tradução de Programas: da Montagem à Carga. 1. ed. Caxias do Sul, RS: EDUCS, 1996. v. 1. 208p .