Introdução à Software Básico: Ligador

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

Sumário

Ligadores

Módulos

- No caso de montadores simples, os programas são formados por um único módulo.
- Neste caso, os montadores geram arquivos executáveis diretamente. Que podem ser passados para o carregador.
- Montadores mais sofisticados permitem construir programas formados por vários **módulos**, que são compilados separadamente (programação modular).
- Nesse casso, o montador vai gerar um arquivo objeto para cada módulo

Módulos

- O conjunto de arquivos objeto vai ser tratado pelo ligador, que vai gerar um arquivo de saída no formato executável

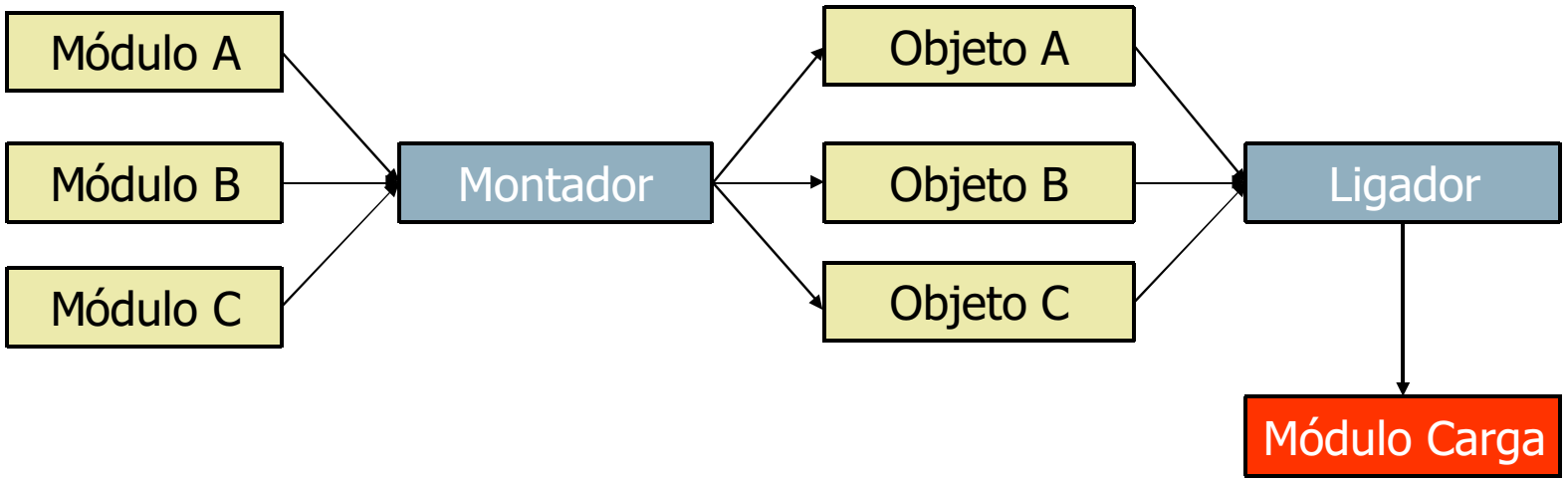


Figura: Programa Montado com vários módulos

Módulos

- Pode-se dizer que:
  - O formato de um arquivo objeto (saída do montador) é definido pelo projetista do ligador, pois é o ligador que vai ler esse tipo de arquivo
  - Já o formato do arquivo executável (saída do ligador) é definido pelo projetista do sistema operacional, pois o carregador faz parte do sistema operacional.
- Os ligadores realizam duas tarefas básicas:
  - juntam os diversos arquivos objeto em um único arquivo executável
  - resolvem as referências cruzadas.
- Uma referência cruzada surge quando o programador usa, em um módulo, um símbolo que é definido em outro módulo.

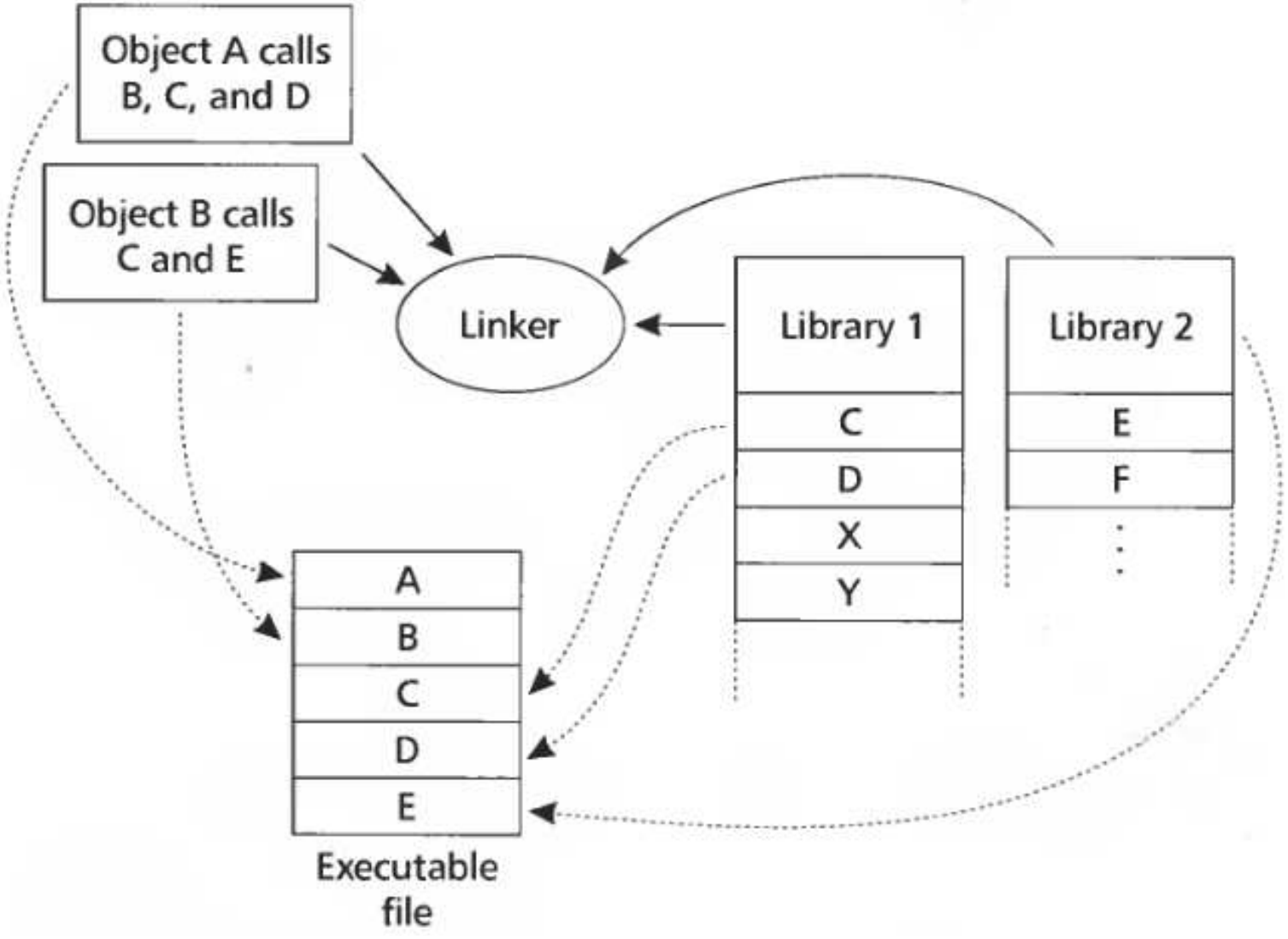


Figura: Referência cruzada

Diretivas EXTERN e PUBLIC

- Em cada módulo, o programador deve informar quais são os símbolos externos usados no módulo (símbolos importados).
  - Isto é feito através da diretiva EXTERN.
- Também deve informar os símbolos definidos no módulo que poderão ser usados por outros módulos (símbolos exportados).
  - Isto é feito através da diretiva PUBLIC.

Tabelas

- O montador informa ao ligador (no arquivo objeto de cada módulo) os símbolos externos que são usados e os símbolos internos que são exportados, através de duas tabelas:
  - **Tabela de Uso** - indica os símbolos externos utilizados no módulo.
  - **Tabela de Definições** - indica os símbolos públicos e seus atributos (é uma fração da TS).

Alterações no Montador

Modificações na primeira passagem:

- Quando a diretiva EXTERN é encontrada, insere o respectivo rótulo na TS com valor **"zero absoluto"** e a indicação de símbolo externo;
- Quando a diretiva PUBLIC é encontrada, insere o respectivo operando na Tabela de Definições, **sem incluir qualquer atributo além do nome**;
- Ao final da primeira passagem, o montador percorre todas as entradas da Tabela de Definições, copiando os respectivos atributos da TS para a Tabela de Definições.

Alterações no Montador

Modificação na segunda passagem:

- Quando um símbolo externo é usado como operando, anota na Tabela de Uso o nome do símbolo e a posição do código objeto em que ele é usado

Alterações no Montador

Exemplo

```
;Código fonte do módulo A:
MOD_A:  BEGIN
Y:      EXTERN
MOD_B:  EXTERN
        PUBLIC  VAL
        PUBLIC  L1
        INPUT   Y
        LOAD    VAL
        ADD     Y
        STORE   Y + 2
        JMPP    MOD_B
L1:     STOP
VAL:    CONST   5
        END
```

```
;Código fonte do módulo B:
MOD_B:  BEGIN
VAL:    EXTERN
L1:     EXTERN
        PUBLIC  Y
        PUBLIC  MOD_B
        OUTPUT  Y
        OUTPUT  VAL
        OUTPUT  Y + 2
        JMP     L1
Y:      SPACE   3
        END
```

Exemplo

- Geração do código Módulo A

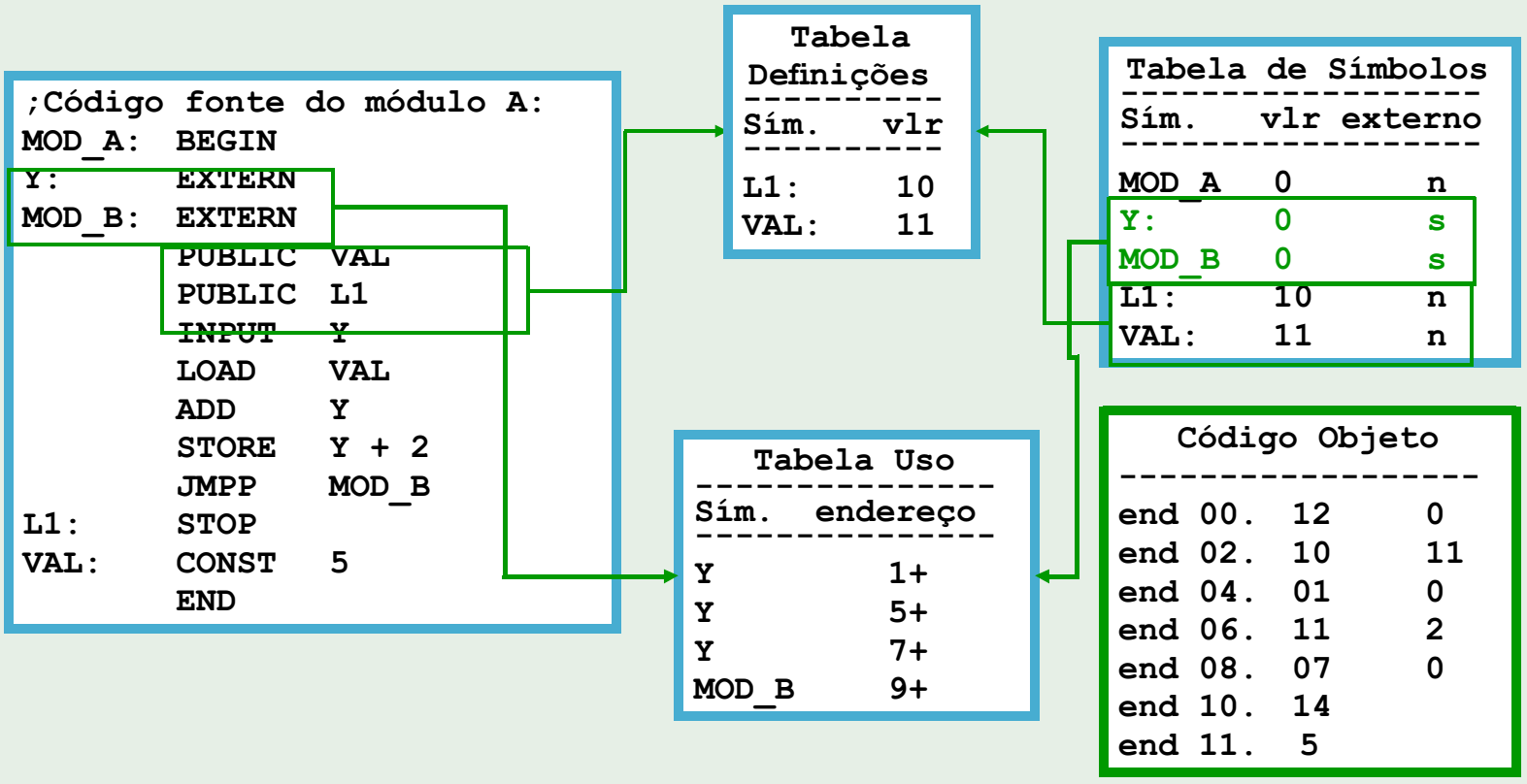
```
;Código fonte do módulo A:
MOD_A:  BEGIN
Y:      EXTERN
MOD_B:  EXTERN
        PUBLIC  VAL
        PUBLIC  L1
        INPUT   Y
        LOAD    VAL
        ADD     Y
        STORE   Y + 2
        JMPP    MOD_B
L1:     STOP
VAL:    CONST   5
        END
```

```
Tabela de Símbolos
------------------
Sím.    vlr externo
------------------
MOD_A   0       n
Y:      0       s
MOD_B   0       s
L1:     10      n
VAL:    11      n
```

```
   Código Objeto
-----------------
end 00.  12     0
end 02.  10     11
end 04.  01     0
end 06.  11     2
end 08.  07     0
end 10.  14
end 11.   5
```

Exemplo

- Geração do código Módulo A



Exemplo

- Geração do código Módulo B

```
;Código fonte do módulo B:
MOD_B:  BEGIN
VAL:    EXTERN
L1:     EXTERN
        PUBLIC  Y
        PUBLIC  MOD_B
        OUTPUT  Y
        OUTPUT  VAL
        OUTPUT  Y + 2
        JMP     L1
Y:      SPACE   3
        END
```

Tabela Definições

Sím.	vlr
MOD_B	0
Y	8

Tabela de Símbolos

Sím.	vlr	externo
MOD_B	0	n
L1	0	s
VAL	0	s
Y	8	n

Tabela Uso

Sím.	endereço
VAL	3+
L1	7+

Código Objeto

```
end 00. 13      8
end 02. 13      0
end 04. 13      10
end 06. 05      0
end 08. xx
end 09. xx
end 10. xx
```

Exemplo

Módulo A

```
;Código fonte do
;módulo A:
MOD_A:  BEGIN
Y:      EXTERN
MOD_B:  EXTERN
        PUBLIC  VAL
        PUBLIC  L1
        INPUT   Y
        LOAD    VAL
        ADD     Y
        STORE   Y + 2
        JMPP    MOD_B
L1:     STOP
VAL:    CONST   5
        END
```

Tabela Definições

Sím.	vlr
L1	10
VAL	5

Código Objeto

```
end 00. 12      0
end 02. 10      11
end 04. 01      0
end 06. 11      2
end 08. 07      0
end 10. 14
end 11.  5
```

Tabela Uso

Sím.	end
Y	1+
Y	5+
Y	7+
MOD_B	9+

Módulo B

```
;Código fonte do
;módulo B:
MOD_B:  BEGIN
VAL:    EXTERN
LI:     EXTERN
        PUBLIC  Y
        PUBLIC  MOD_B
        OUTPUT  Y
        OUTPUT  VAL
        OUTPUT  Y + 2
        JMP     L1
Y:      SPACE   3
        END
```

Tabela Definições

Sím.	vlr
MOD_B	0
Y	8

Código Objeto

```
end 00. 13      8
end 02. 13      0
end 04. 13      10
end 06. 05      0
end 08. xx
end 09. xx
end 10. xx
```

Tabela Uso

Sím.	end
VAL	3+
L1	7+

Processo de Ligação

- Como os endereços de cada módulo foram gerados para o mesmo endereço inicial (zero), o ligador tem que relocar os módulos em um único espaço de endereços.
- Para isso, o ligador executa as seguintes tarefas:
  - Constrói uma tabela contendo o tamanho de cada arquivo objeto
    - Essa informação é chamada de **fator de correção**
  - Colhe informações das diversas tabelas de definições e cria a tabela **global de definições**

Processo de Ligação

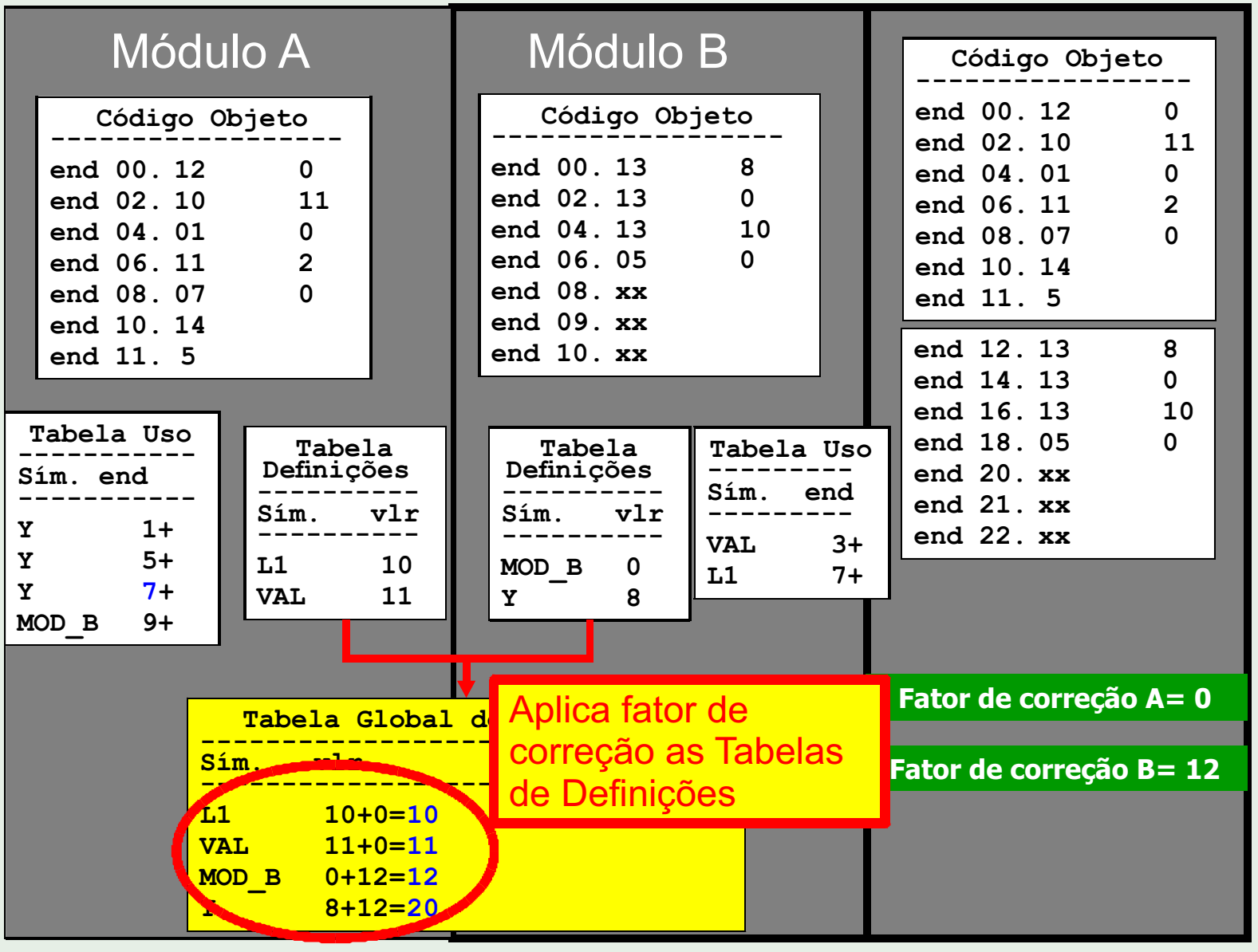
Para cada código objeto faz:

- Ler as entradas na **tabela de uso**, encontrar o valor do símbolo na **tabela global de definições** e corrigir os endereços
- Somar o fator de correção as instruções que contém um endereço de memória ainda não corrigidas
- Corrigir os endereços **relativos** de acordo com o fator de correção
- Gerar o código executável

Algoritmo de Ligação

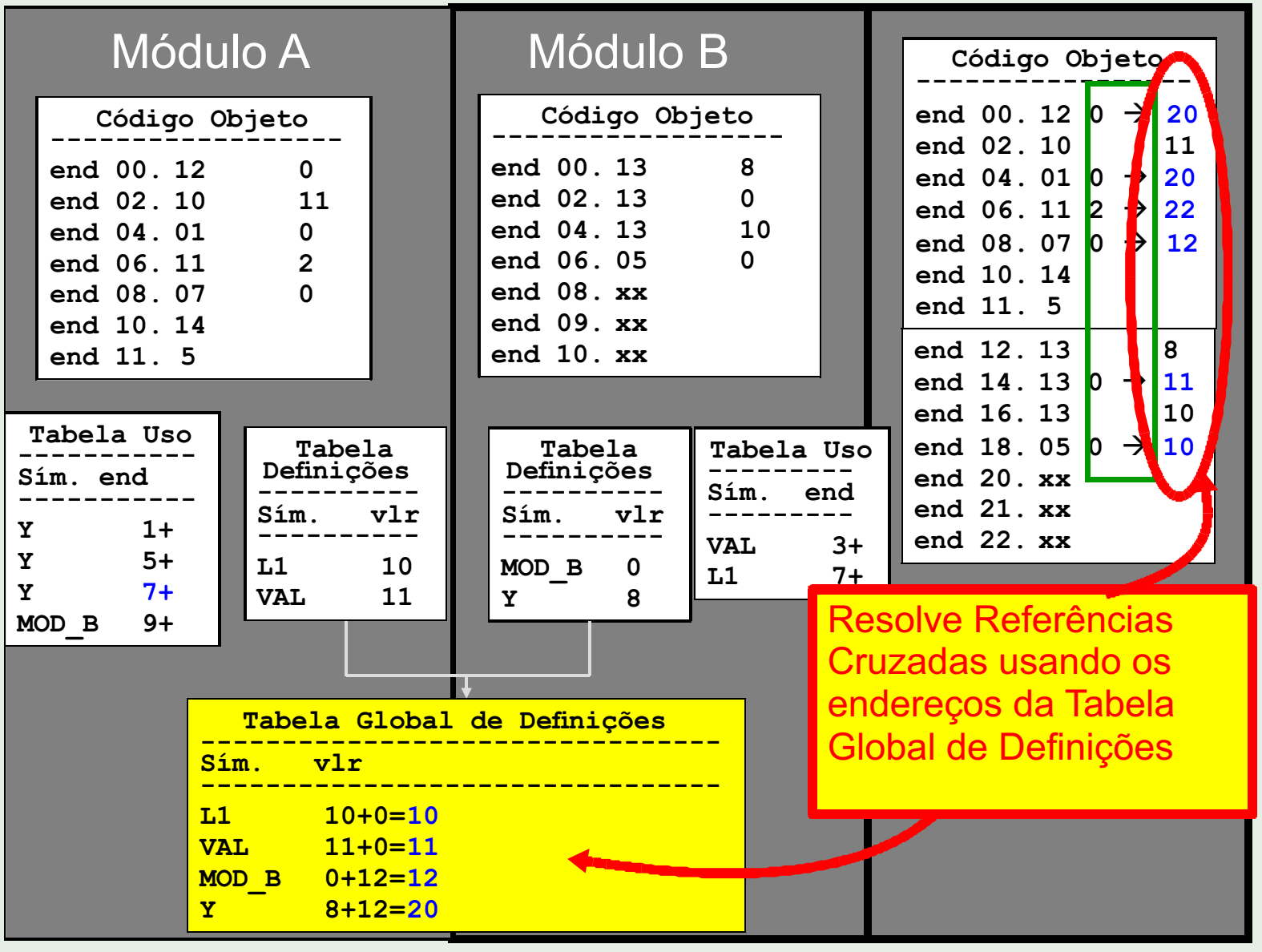
- 1.– Alinha os códigos objeto a serem ligados
- 2.– Constrói tabela com fatores de correção
- 3.– Constrói tabela global de definições, utilizando os fatores de correção.
- 4.– Para cada objeto:
  - 4.1.– Corrigir os endereços das entradas da tabela de uso, utilizando a tabela global de definições
  - 4.2.– Corrigir os endereços do código usando os fatores de correção
  - 4.3.– Corrigir os endereços relativos usando os fatores de correção
- 5.– Gerar código executável e salvar em arquivo

Exemplo

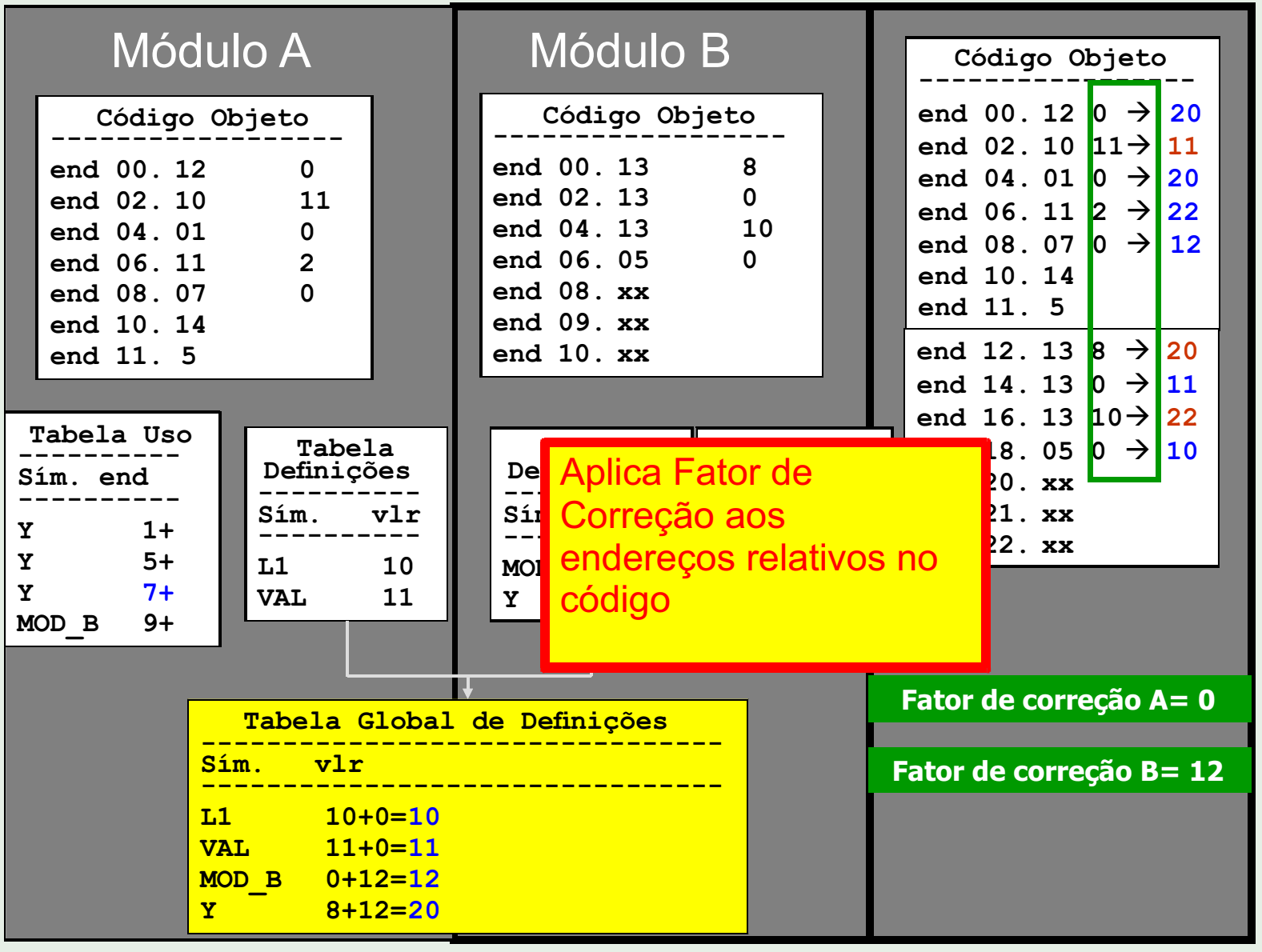

Módulo A
Código Objeto
end 00. 12 0
end 02. 10 11
end 04. 01 0
end 06. 11 2
end 08. 07 0
end 10. 14
end 11. 5
Tabela Uso
Sím. end
Y 1+
Y 5+
Y 7+
MOD_B 9+
Tabela Definições
Sím. vlr
L1 10
VAL 11
Módulo B
Código Objeto
end 00. 13 8
end 02. 13 0
end 04. 13 10
end 06. 05 0
end 08. xx
end 09. xx
end 10. xx
Tabela Definições
Sím. vlr
MOD_B 0
Y 8
Tabela Uso
Sím. end
VAL 3+
L1 7+
Código Objeto
end 00. 12 0
end 02. 10 11
end 04. 01 0
end 06. 11 2
end 08. 07 0
end 10. 14
end 11. 5
end 12. 13 8
end 14. 13 0
end 16. 13 10
end 18. 05 0
end 20. xx
end 21. xx
end 22. xx
Tabela Global d
Sím.
L1 10+0=10
VAL 11+0=11
MOD_B 0+12=12
8+12=20
Aplica fator de correção as Tabelas de Definições
Fator de correção A= 0
Fator de correção B= 12

Algoritmo de Ligação

Exemplo

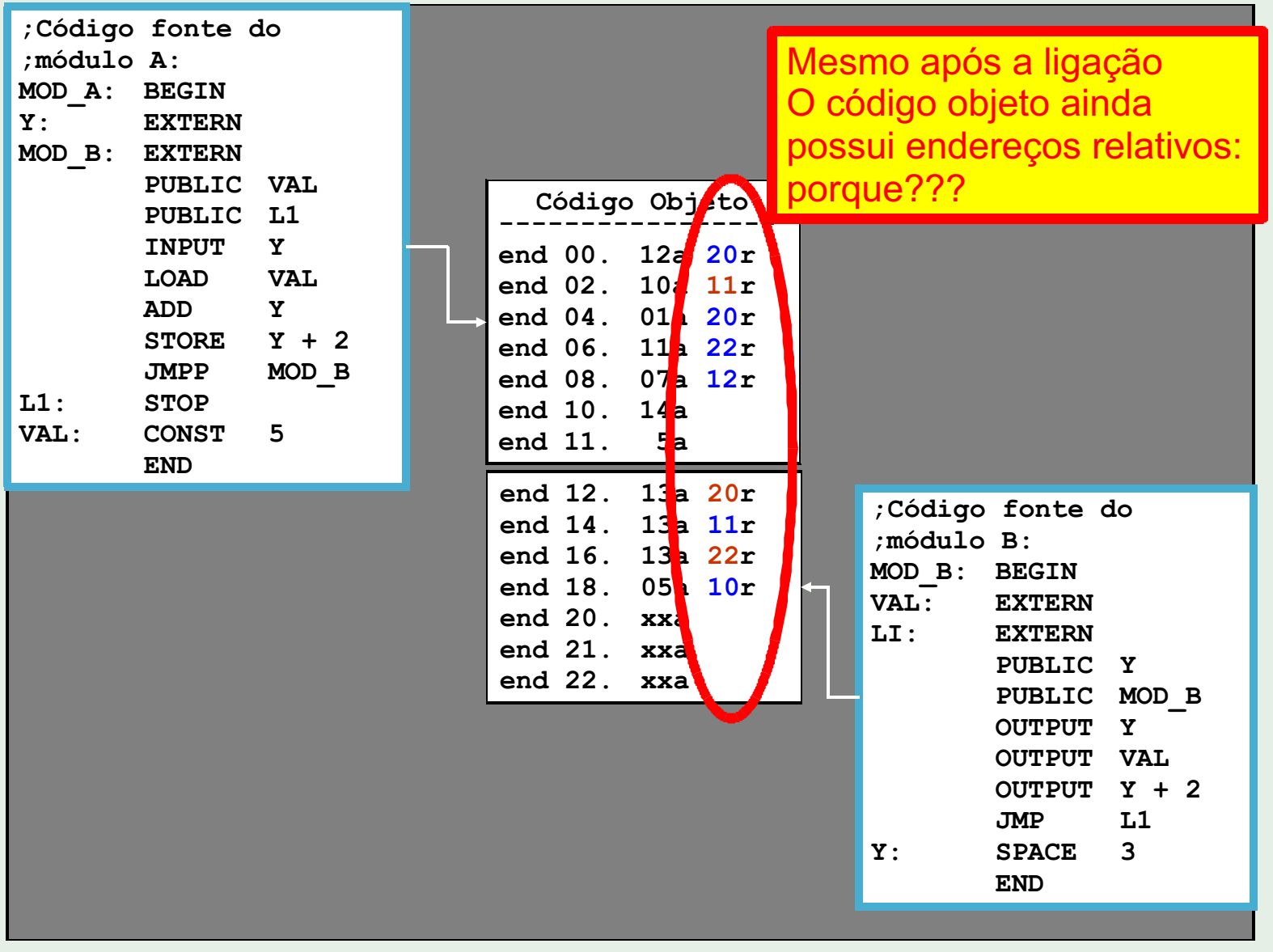


Exemplo


Módulo A
Código Objeto
end 00. 12 0
end 02. 10 11
end 04. 01 0
end 06. 11 2
end 08. 07 0
end 10. 14
end 11. 5
Módulo B
Código Objeto
end 00. 13 8
end 02. 13 0
end 04. 13 10
end 06. 05 0
end 08. xx
end 09. xx
end 10. xx
Código Objeto
end 00. 12 0 → 20
end 02. 10 11→ 11
end 04. 01 0 → 20
end 06. 11 2 → 22
end 08. 07 0 → 12
end 10. 14
end 11. 5
end 12. 13 8 → 20
end 14. 13 0 → 11
end 16. 13 10→ 22
18. 05 0 → 10
20. xx
21. xx
22. xx
Tabela Uso
Sím. end
Y 1+
Y 5+
Y 7+
MOD_B 9+
Tabela Definições
Sím. vlr
L1 10
VAL 11
Aplica Fator de Correção aos endereços relativos no código
Tabela Global de Definições
Sím. vlr
L1 10+0=10
VAL 11+0=11
MOD_B 0+12=12
Y 8+12=20
Fator de correção A= 0
Fator de correção B= 12

Algoritmo de Ligação

Exemplo



Exemplo 2

- Para o código abaixo (pseudo-asm) gere o código objeto utilizando o algoritmo de duas passagens
  - crie as tabelas de símbolo, TU, TD
  - gere o executável de acordo com o algoritmo de ligação

```
MOD_A:  BEGIN
R:      EXTERN
MOD_B:  EXTERN
        PUBLIC  A
        PUBLIC  B
        PUBLIC  L1
        INPUT   A
        INPUT   B
        JMP     MOD_B
L1:     OUTPUT  R+1
        STOP
A:      SPACE   1
B:      SPACE   1
        END
```

```
MOD_B:  BEGIN
A:      EXTERN
L1:     EXTERN
        PUBLIC  R
        PUBLIC MOD_B
        LOAD    A
        MUL     B
        STORE   R
        DIV     DOIS
        STORE   R+1
        JMP     L1
R:      SPACE   2
DOIS:   CONST   2
        END
```

Figura: Programa área de um triangulo

Solução - Geração de Tabelas (MONTADOR)

```
MOD_A:  BEGIN
R:      EXTERN
MOD_B:  EXTERN
        PUBLIC  A
        PUBLIC  B
        PUBLIC  L1
        INPUT   A
        INPUT   B
        JMP     MOD_B
L1:     OUTPUT  R+1
        STOP
A:      SPACE   1
B:      SPACE   1
        END
```

```
 0. 12  9
 2. 12 10
 4. 05  0
 6. 13  1
 8. 14
 9. xx
10. xx
```

```
 0. 10  0
 2. 03  0
 4. 11 12
 6. 04 14
 8. 11 13
10. 05  0
12. xx
13. xx
14.  2
```

```
MOD_B:  BEGIN
A:      EXTERN
B:      EXTERN
L1:     EXTERN
        PUBLIC  R
        PUBLIC MOD_B
        LOAD    A
        MUL     B
        STORE   R
        DIV     DOIS
        STORE   R+1
        JMP     L1
R:      SPACE   2
DOIS:   CONST   2
        END
```

TD	
MOD_B	0
R	12

TS	
MOD_A	0
R	0
MOD_B	0
L1	6
A	9
B	10

TU	
MOD_B	5+
R	7+

TD	
L1	6
A	9
B	10

TU	
A	1+
B	2+
L1	11+

TS	
MOD_B	0
A	0
B	0
L1	0
R	12
DOIS	14

Solução - Ligador

TS	
MOD_A	0
R	0
MOD_B	0
L1	6
A	9
B	10

TU	
MOD_B	5+
R	7+

```
 0. 12  9
 2. 12 10
 4. 05  0
 6. 13  1
 8. 14
 9. xx
10. xx
```

TD	
L1	6
A	9
B	10

TGD		
Sím.	vlr	
L1	6+ 0=	6
A	9+ 0=	9
B	10+ 0=	10
MOD_B	0+11 =	11
R	12+11=	23

```
 0. 12  9
 2. 12 10
 4. 05  0
 6. 13  1
 8. 14
 9. xx
10. xx
11. 10  0
13. 03  0
15. 11 12
17. 04 14
19. 11 13
21. 05  0
23. xx
24. xx
25.  2
```

TU	
A	1+
B	3+
L1	11+

TS	
MOD_B	0
A	0
B	0
L1	0
R	12
DOIS	14

TD	
MOD_B	0
R	12

```
 0. 10  0
 2. 03  0
 4. 11 12
 6. 04 14
 8. 11 13
10. 05  0
12. xx
13. xx
14.  2
```

Fator de correção A= 0

Fator de correção B= 11

Solução - Ligador

TS	
MOD_A	0
R	0
MOD_B	0
L1	6
A	9
B	10

TU	
MOD_B	5+
R	7+

0.	12	9
2.	12	10
4.	05	0
6.	13	1
8.	14	
9.	xx	
10.	xx	

TD	
L1	6
A	9
B	10

0.	12	9
2.	12	10
4.	05	11
6.	13	24
8.	14	
9.	xx	
10.	xx	
11.	10	9
13.	03	10
15.	11	23
17.	04	25
19.	11	24
21.	05	6
23.	xx	
24.	xx	
25.	2	

TU	
A	1+
B	3+
L1	11+

TS	
MOD_B	0
A	0
B	0
L1	0
R	12
DOIS	14

TD	
MOD_B	0
R	12

0.	10	0
2.	03	0
4.	11	12
6.	04	14
8.	11	13
10.	05	0
12.	xx	
13.	xx	
14.	2	

TGD

Sím.	vlr
L1	6+ 0=6
A	9+ 0=9
B	10+ 0=10
MOD_B	0+11 =11
R	12+11=23

Fator de correção A= 0

Fator de correção B= 11

Algoritmo de Ligação

Exemplo 2

- Re-escreva os programa FAT (fatorial) e Fibonacci para que sejam utilizados mais de um módulo:
  - Cada um dos programas deve ser montado e ligado.
- Considerando o algoritmo de ligação que acabamos de ver, indique como podem ser identificados os seguintes erros:
  - Símbolo foi definido em mais de um módulo
  - Símbolo não foi definido em nenhum módulo
  - Expressão resultou em um endereço inválido

Próxima Aula

Carregador