PCS 3216
Sistemas de Programação

Aula 13

Preparação de programas relocáveis

DETALHAMENTO DOS VÁRIOS TIPOS DE ENDEREÇAMENTO

Resolução de endereços em programas simbólicos relocáveis

- Na sequência, detalha-se um pouco mais o tratamento que deve ser dado pelo montador para a **resolução** de cada uma das formas de **endereçamento** tipicamente encontradas nos programas relocáveis: **absoluto, relocável, relativo e simbólico**.

Endereçamento absoluto

- O primeiro tipo de endereçamento que pode encontrado em programas-objeto relocáveis é o já conhecido endereçamento ***absoluto***.
- Mesmo em programas não absolutos, este tipo de endereçamento é, muitas vezes, necessário, já que alguns programas necessitam referenciar posições físicas específicas e invariáveis de memória, independente da região de memória em que o programa deverá ser executado.

Usos do endereçamento absoluto

- Isto ocorre nos casos em que o programa deve **acessar regiões de comunicação** com o sistema operacional, ou, em casos mais simples, posições de memória através das quais o **hardware interage com o software**, como é o caso das posições associadas a eventos de **interrupção**.

- **Endereços absolutos** são endereços já resolvidos, portanto **dispensam tratamento de resolução**.

Endereçamento relativo

- O segundo tipo de endereçamento é o ***relativo***.
- Através de ***deslocamentos***, as instruções de referência à memória fazem referências a endereços pertencentes ao programa que se está desenvolvendo.
- Esses **deslocamentos** correspondem a **distâncias** entre algum **ponto de referência** (interno ao programa) e a posição relativa de memória assim referenciada.

Referenciais para endereçamento relativo

- Os deslocamentos são portanto **endereços relativos**, e cada Sistema de Programação geralmente convenciona alguns **pontos de referência** (por exemplo: a primeira instrução executável do programa; o início da área de *common*; o início da área de dados, o início da área de pilha, etc.), em relação aos quais são expressos, internamente ao módulo relocável, os demais endereços, internos ao programa.

Endereçamento relocável

- Os endereços relocáveis referem-se a locais cuja **posição física** permanece indefinida, e portanto, **flutuante,** até que seja escolhido o endereço de memória a partir do qual deverá residir o programa que a contém.
- Através de operações de **relocação**, tais referências relocáveis devem ser, então, convertidas em referências absolutas, para que só então o programa possa ser executado.

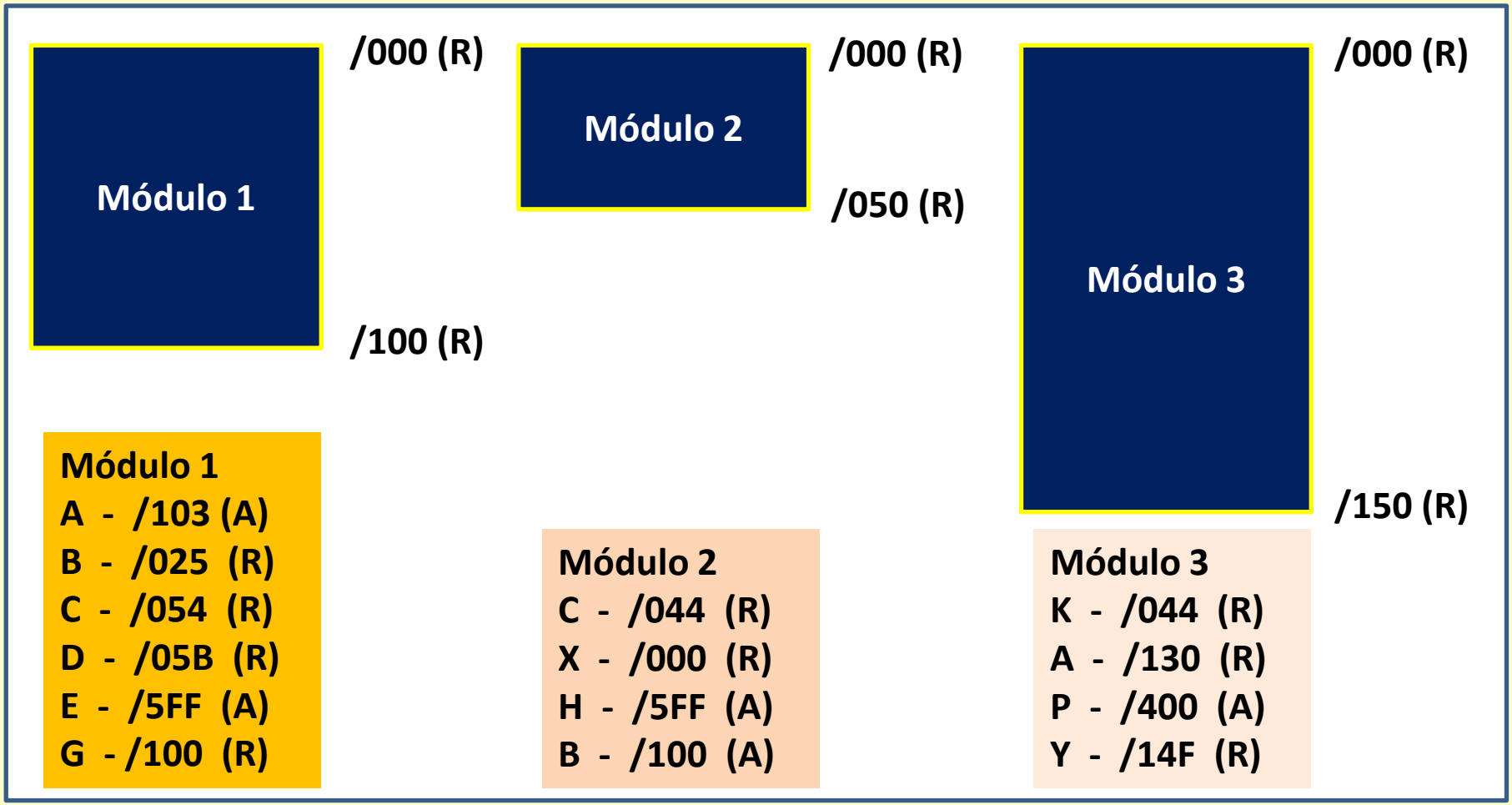
Resolução de endereços relativos

- Para ser possível executar o programa, todos os endereços relativos devem ser convertidos para endereços absolutos (operação de **resolução de endereços relativos**).
- Isto é feito através da **adição**, aos endereços relativos, de um valor, correspondente ao **endereço absoluto** associado ao **ponto de referência** a que se referem os endereços relativos em questão (**base de relocação**).

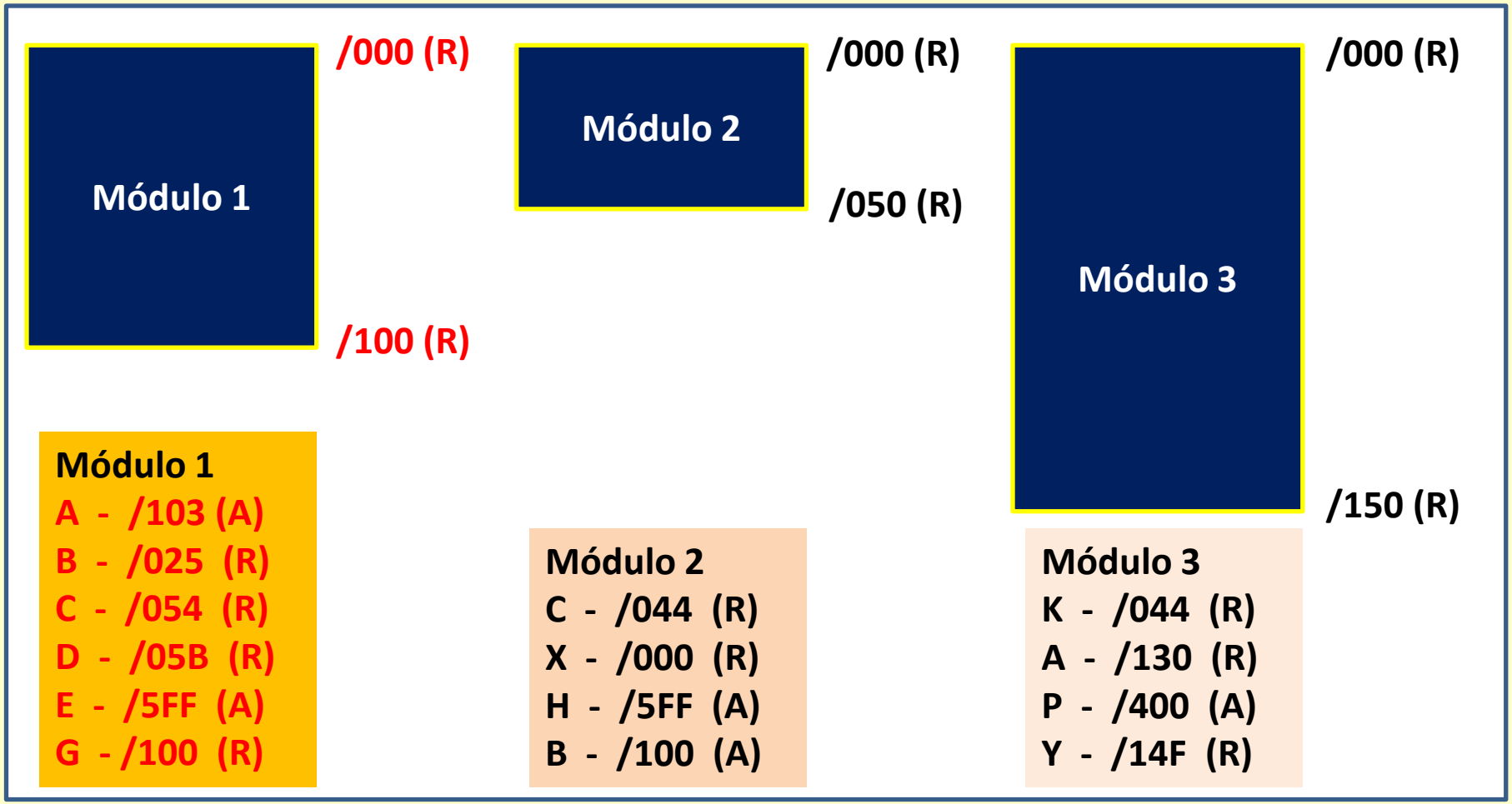
Relocações estáticas intermediárias

- Isto acarretaria um excessivo aumento no tempo de execução do programa, o qual pode ser facilmente reduzido se for efetuada a priori uma **relocação (estática)** das rotinas que compõem o programa, mediante a **alteração física dos endereços relativos** internos a cada módulo.
- Tal operação simplifica o código objeto, produzindo **um único módulo** composto, em que figuram apenas referências relocáveis na forma de endereços **relativos a uma única base**.
- Convém, em cada caso particular, levar em conta se esta operação traz reais vantagens, dado que se trata de um procedimento não obrigatório, o qual pode tornar-se oneroso, se sistematicamente utilizado.

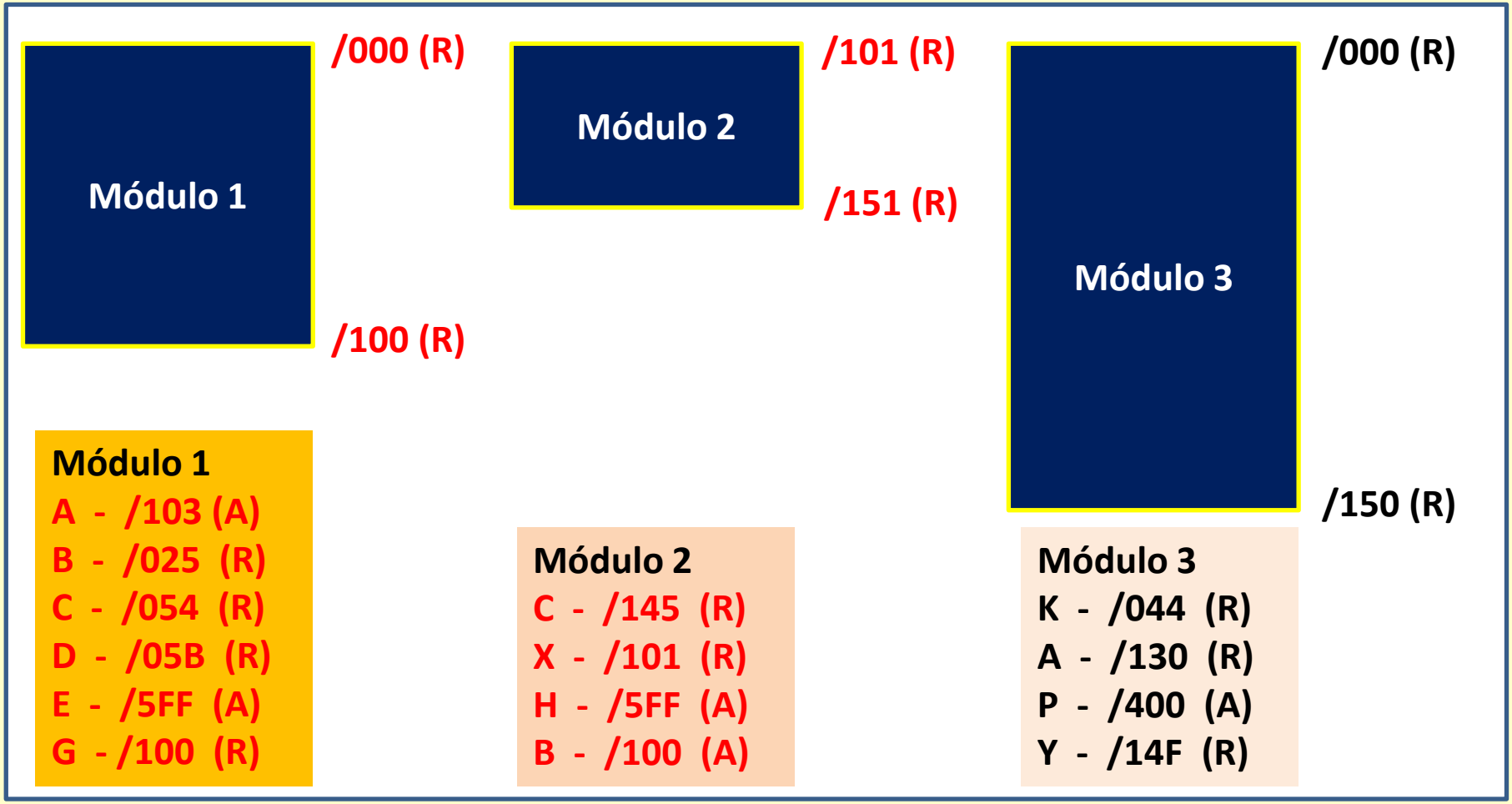
Relocação estática intermediária (situação inicial)



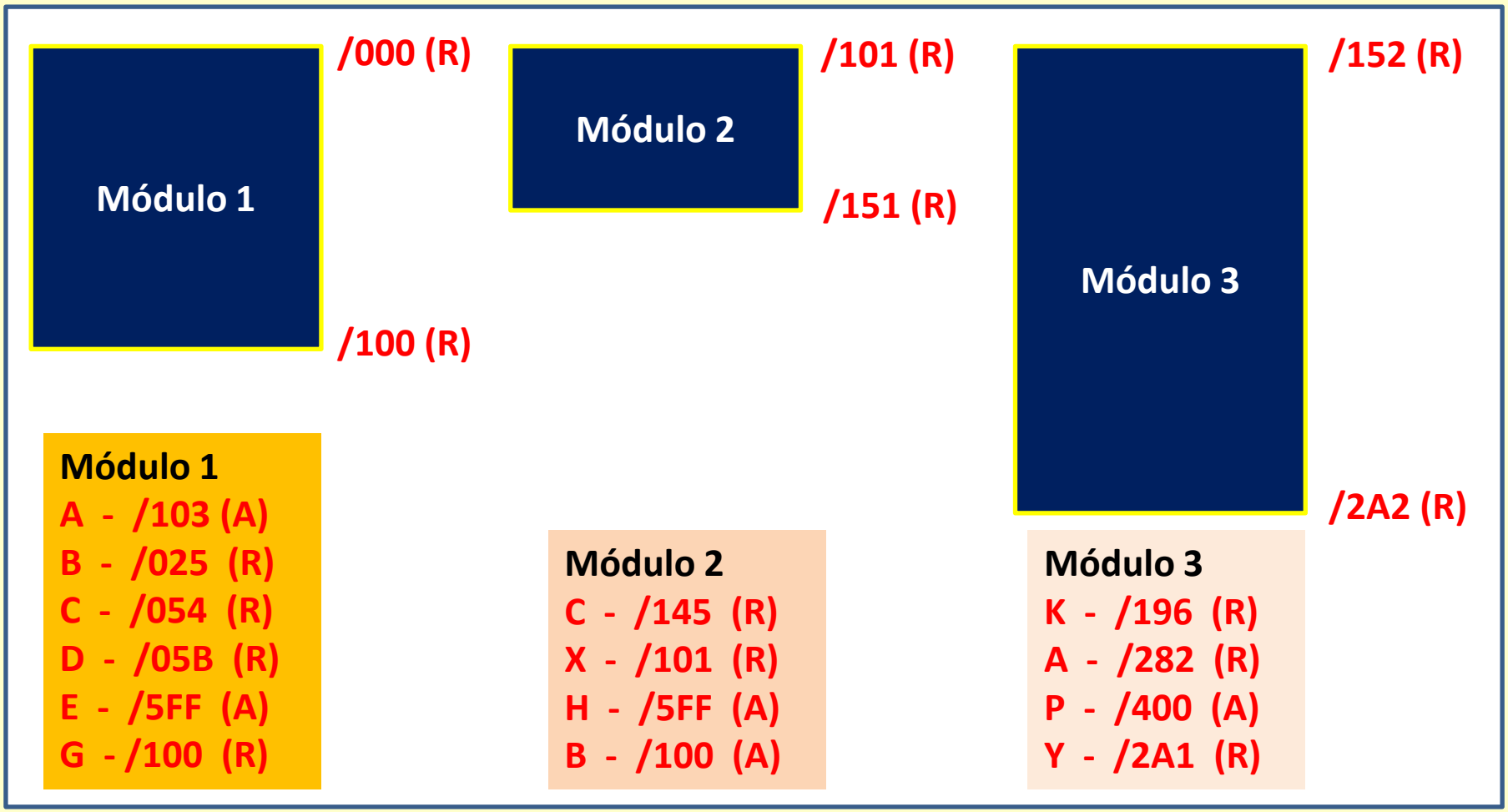
Alocação com relocação,módulo a módulo (Módulo 1: alocação em /000 (R))



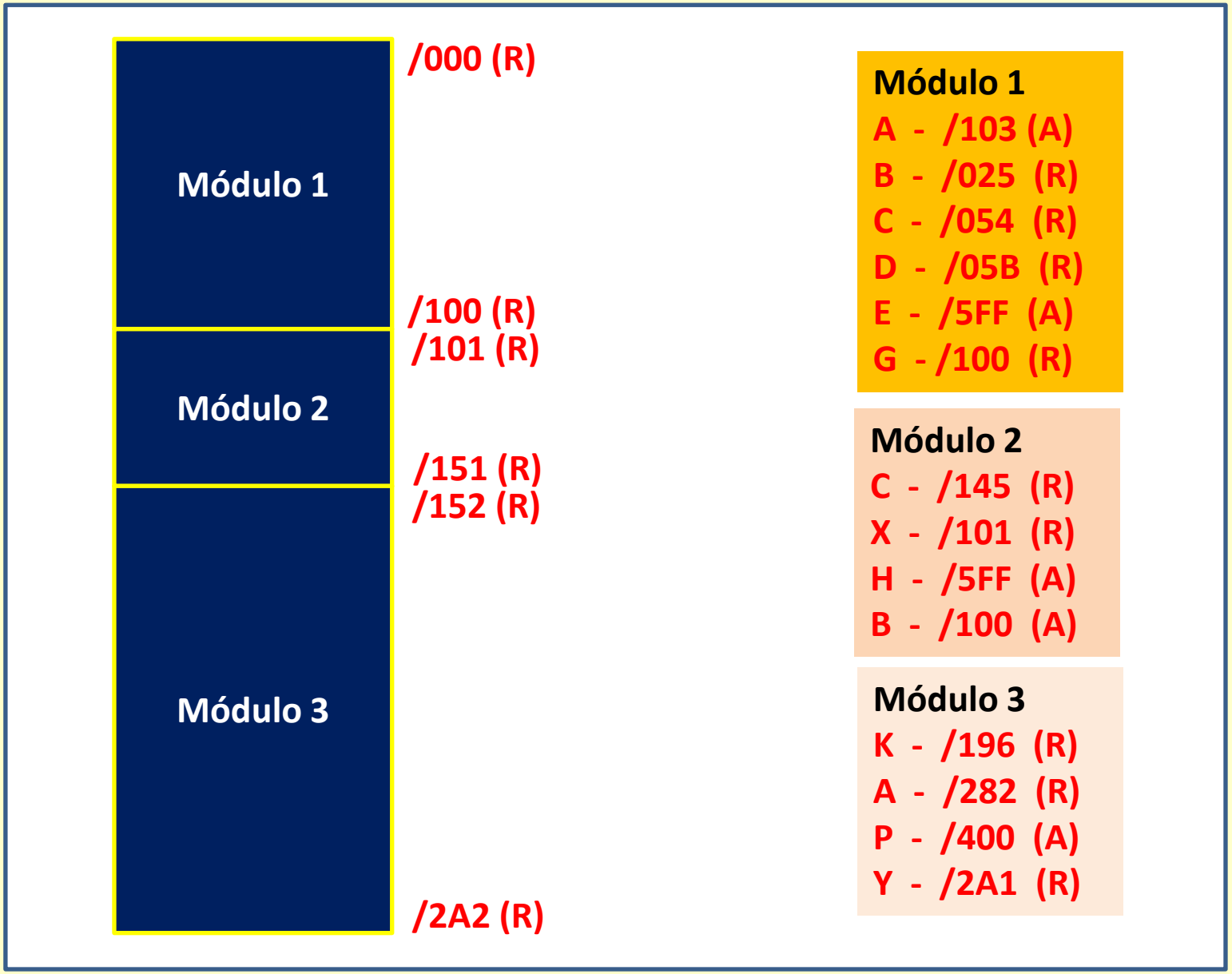
Alocação com relocação,módulo a módulo (Módulo 2: alocação em /101 (R))



Alocação com relocação,módulo a módulo (Módulo 3: alocação em /152 (R))



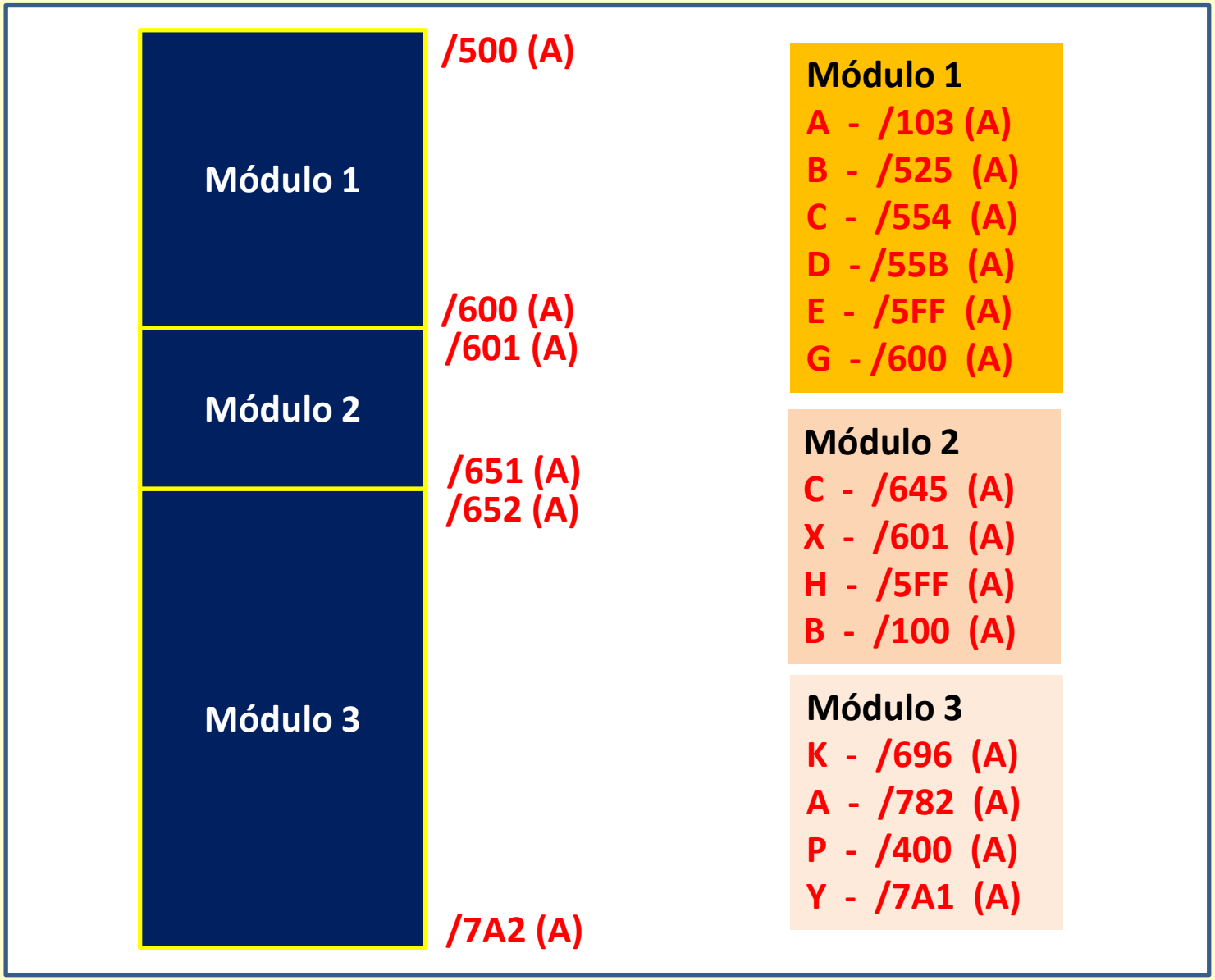
Result.final da reloc.estática intermediária



Relocação final

- A relocação estática intermediária ainda não produz, como pode ser observado, um código executável, pois muitas referências permanecem relocáveis após sua aplicação.
- Por outro lado, o seu resultado final é um **código objeto relocável sem referências simbólicas**, pronto para ser relocado a partir de uma base única, a ser arbitrada.
- Para obter portanto um **código absoluto**, pronto para ser carregado na memória para execução, basta efetuar a **relocação final** a partir de uma origem (base de relocação), geralmente estabelecida pelo usuário ou então pelo sistema operacional.
- Se, ao resultado da relocação estática intermediária do exemplo anterior, aplicarmos o endereço absoluto **/500 como base de relocação** para construir um código pronto para a execução, com todos os endereços resolvidos, obtemos o resultado que pode ser visualizado a seguir.

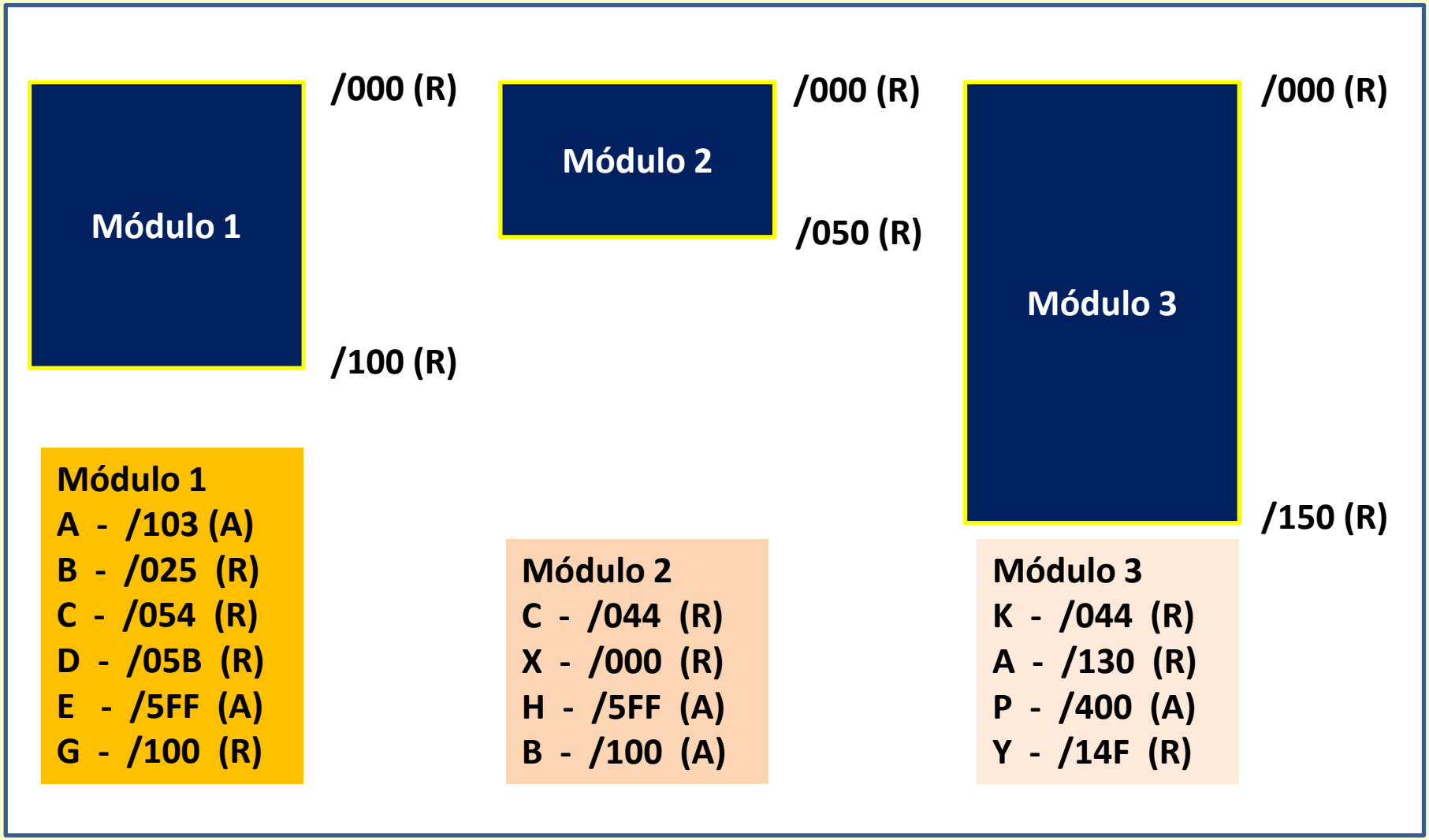
Código final, relocado p/exec. em /500 (A)



Fase de relocação

- Qualquer que seja o caso, portanto, é de grande importância, sempre que necessário, o ajuste *a priori* de todos os endereços relativos contidos no código do programa, antes da sua execução.
- A operação de ***relocação estática*** constitui uma tarefa fundamental na preparação de programas para a execução em um Sistema de Programação.
- A atividade de um Sistema de Programação para resolver completamente os endereços do programa, (convertê-los em programas absolutos) constitui a ***fase de relocação*** dos módulos que compõem esses programas.

Situação inicial: relocação módulo a módulo



Alocação com relocação, módulo a módulo (Módulo 1: alocação em /500 (A))

/500 (A)

/600 (A)

/000 (R)

/050 (R)

/000 (R)

/150 (R)

Módulo 1
A - /103 (A)
B - /525 (R)
C - /554 (R)
D - /55B (R)
E - /5FF (A)
G - /600 (R)

Módulo 2
C - /044 (R)
X - /000 (R)
H - /5FF (A)
B - /100 (A)

Módulo 3
K - /044 (R)
A - /130 (R)
P - /400 (A)
Y - /14F (R)

Alocação com relocação, módulo a módulo (Módulo 2: alocação em /601 (A))

/500 (A)

/600 (A)

/601 (A)

/651 (A)

/000 (R)

/150 (R)

Módulo 1
A - 103 (A)
B - 525 (A)
C - 554 (A)
D - 55B (A)
E - 5FF (A)
G - 600 (A)

Módulo 2
C - /645 (A)
X - /601 (A)
H - /5FF (A)
B - /100 (A)

Módulo 3
K - /044 (R)
A - /130 (R)
P - /400 (A)
Y - /14F (R)

Alocação com relocação, módulo a módulo (Módulo 3: alocação em /652 (A))

/500 (A)

/600 (A)

/601 (A)

/651 (A)

/652 (A)

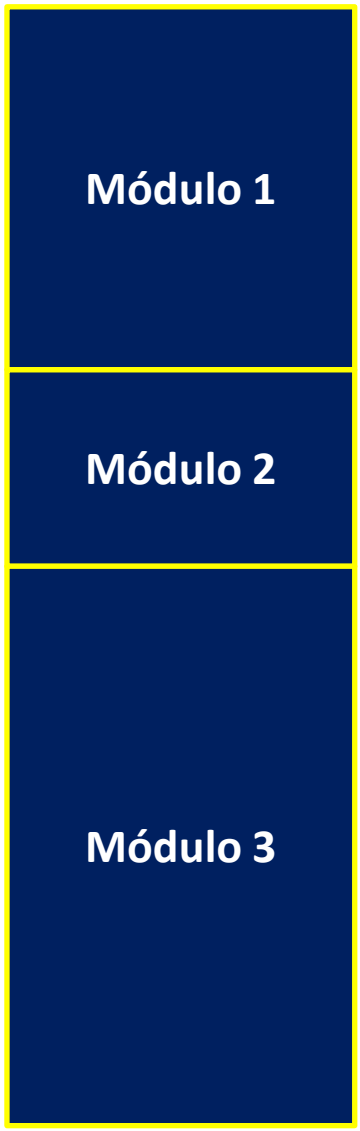
/7A2 (A)

Módulo 1
A - 103 (A)
B - 525 (A)
C - 554 (A)
D - 55B (A)
E - 5FF (A)
G - 600 (A)

Módulo 2
C - /645 (A)
X - /601 (A)
H - /5FF (A)
B - /100 (A)

Módulo 3
K - /696 (A)
A - /782 (A)
P - /400 (A)
Y - /7A1 (A)

Código final resultante da relocação



/500 (A)

/600 (A)
/601 (A)

/651 (A)
/652 (A)

/7A2 (A)

Módulo 1
A - /103 (A)
B - /525 (A)
C - /554 (A)
D - /55B (A)
E - /5FF (A)
G - /600 (A)

Módulo 2
C - /645 (A)
X - /601 (A)
H - /5FF (A)
B - /100 (A)

Módulo 3
K - /696 (A)
A - /782 (A)
P - /400 (A)
Y - /7A1 (A)

Notas sobre a relocação apresentada

- Inicialmente havia **3 módulos relocáveis**, contendo apenas referências internas, a endereços absolutos e relocáveis.
- Após a relocação separada de cada módulo, as referências internas **absolutas se mantêm** e as **relocáveis se transformam em absolutas** mediante a **soma da base** de relocação específica de cada módulo.
- A base de relocação vai sendo atualizada de acordo com a ocupação da memória pelo código absoluto em construção.
- A **política de alocação** adotada posiciona **contiguamente** na memória os módulos à medida que vão sendo relocados.
- Ao final, todos os endereços se tornam **absolutos** e os três módulos se transformam em um código único, ocupando **posições adjacentes de memória**.

Endereçamento simbólico

- Outro tipo ainda de endereçamento, encontrado em programas objeto relocáveis, é denominado **endereçamento simbólico**, e corresponde a **referências nominais** a posições de memória cujos endereços são desconhecidos no instante em que o programa é escrito, presumindo-se a sua existência em um dos demais módulos de que se compõe o programa.
- Para que esse tipo de referência possa ser utilizado de forma prática, o módulo **portador da declaração do símbolo** deve informar o fato, da mesma forma que o módulo que **deseja fazer referência** a um desses símbolos deve acusar essa intenção ao montador.
- Esse protocolo costuma ser estabelecido através do uso de **duas pseudo-instruções**, uma para informar os símbolos que o módulo **disponibiliza** aos demais, e outra, para indicar os símbolos que o módulo **deseja acessar**, mas que pertençam a outros módulos.

Referências externas

- Seria inútil a possibilidade de se efetuar programação modular se fossem completamente estanques os módulos de que o programa se compõe.
- Assim, é conveniente que cada módulo possa utilizar-se das funções executadas pelos demais, o que se faz mediante referência mútua, com o auxílio de instruções de referência à memória com operandos especiais.
- Para isso, usa-se o conceito de ***referências externas,*** que são operandos que fazem referência a símbolos externos aos módulos, ou seja, endereços simbólicos declarados em outro módulo do programa, diferente daquele no qual são referenciados.

Ponto de acesso (*export*)

- Globalmente, a referência mútua entre os módulos é viável apenas para determinados pontos de cada módulo, indicados pelo programador para serem visíveis aos demais módulos.
- Estes símbolos são declarados, em cada módulo, como sendo ***exports, entry points*** ou ***pontos de acesso*** de entrada ao módulo.
- Cada módulo pode, usualmente, apresentar um ou mais pontos de acesso, cada qual designado por um nome simbólico.

Endereço externo (*import*)

- Estes nomes simbólicos funcionam como se fossem **endereços simbólicos globais**, acessíveis a qualquer dos módulos que deles necessite.
- Nestes, os endereços simbólicos globais a serem referenciados devem ser declarados como sendo ***imports*** ou **endereços externos** (***external addresses***).

Referências entre módulos

- Com esse mecanismo, cria-se um protocolo de comunicação entre módulos desenvolvidos em linguagem simbólica, através da **referência mútua entre os módulos**, efetuada por meio do endereçamento simbólico.
- Tais endereçamentos são **resolvidos fora do montador**, pelos mecanismos do Sistema de Programação responsáveis pela operação de **ligação** (***linking***, ***binding***)

Resolução de referências entre módulos

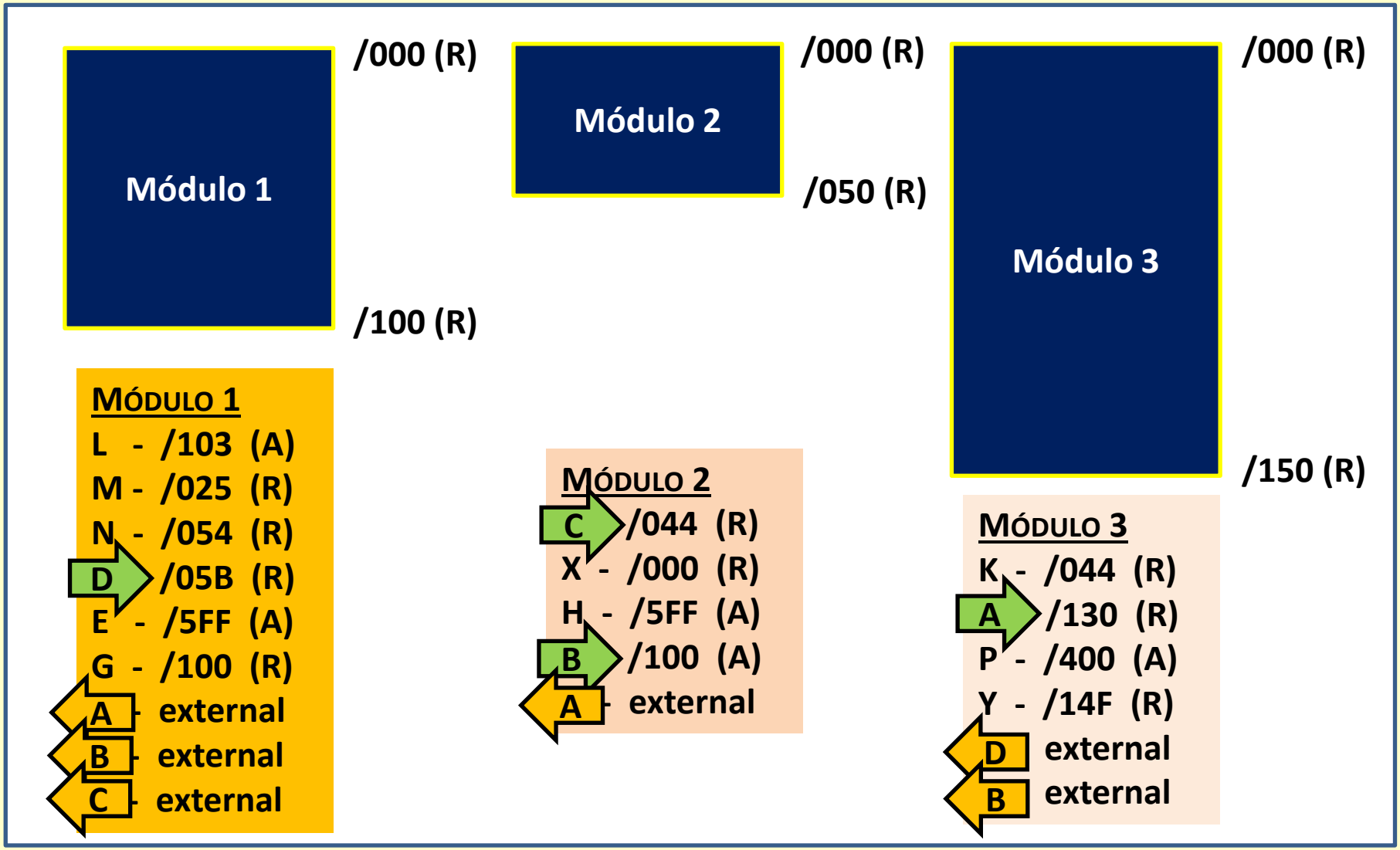
- Para que seja possível a execução de um programa que apresente endereçamento simbólico para referências entre módulos, é necessário converter tais endereços simbólicos em endereços relativos ou absolutos, recaindo então nos casos anteriores.
- Um programa nessas condições se torna **apto para a execução** quando **cada uma das suas referências** do tipo ***import*** tiver sido **satisfeita** pela presença de declarações únicas das respectivas referências, na forma de símbolos do tipo ***export***, além das ações de alocação e de relocação.
- A estratégia, portanto, para recair nas situações já estudadas, consiste apenas em **efetuar em primeiro lugar uma relocação estática intermediária**, com a finalidade de remover as referências simbólicas entre os módulos.
- Para finalizar, basta apenas efetuar as operações de alocação e de relocação, da forma como foi visto anteriormente.

EXEMPLO COMPLETO

Um exemplo detalhado completo

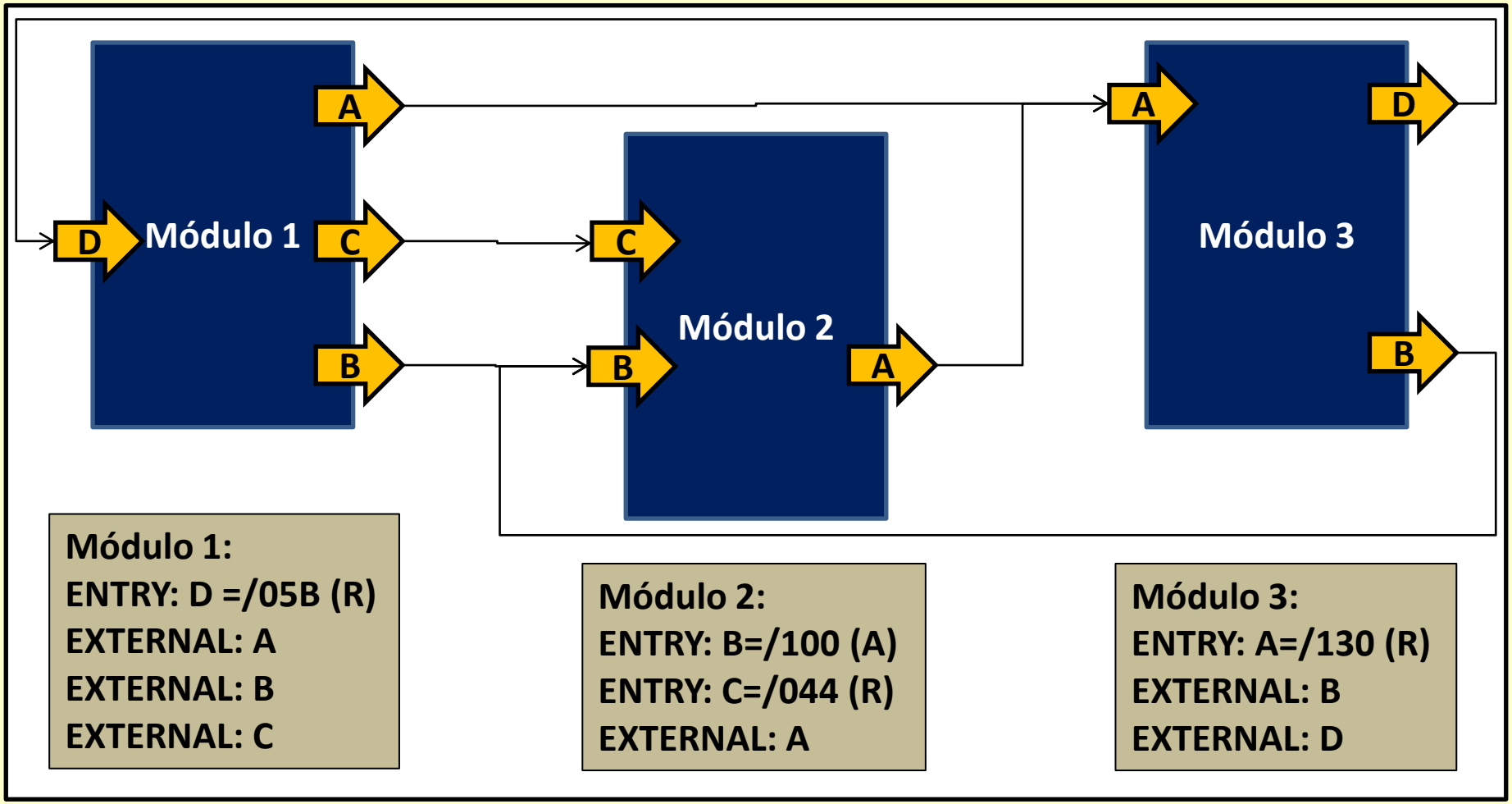
- Neste exemplo, são usados **os mesmos três módulos** dos casos ilustrativos anteriores
- Foram apenas **incorporadas algumas referências simbólicas** entre módulos:
  - O módulo 1 tem D como *entry point,* e como *externals*, A, B e C
  - O módulo 2 tem B e C como *entry points*, e como *external*, A
  - O módulo 3 tem A como *entry point*, e como *externals*, B e D
- Agregando essas informações à situação inicial já apresentada, temos a **situação inicial** deste exemplo.
- Assim, tem-se também, além das referências internas absolutas e relocáveis, as **quatro referências simbólicas** A, B, C e D introduzidas como entry points e externals, e que precisam ser resolvidas.
- A primeira operação a realizar é a **relocação estática intermediária**, com a finalidade de eliminar as referências simbólicas entre módulos através da construção de um só módulo relocável a partir dos três inicialmente apresentados, e em seguida, alocar e relocar o módulo relocável assim obtido.
- ***Observação:*** adotada aqui apenas por razão didática, **a realização em separado** da relocação estática intermediária **poderia ser evitada na prática** construindo-se de uma só vez o código absoluto desejado, para isso **juntando** as operações de **relocação estática intermediária, de alocação e de relocação** final a partir de uma base absoluta de relocação.

Situação inicial, c/referências entre módulos

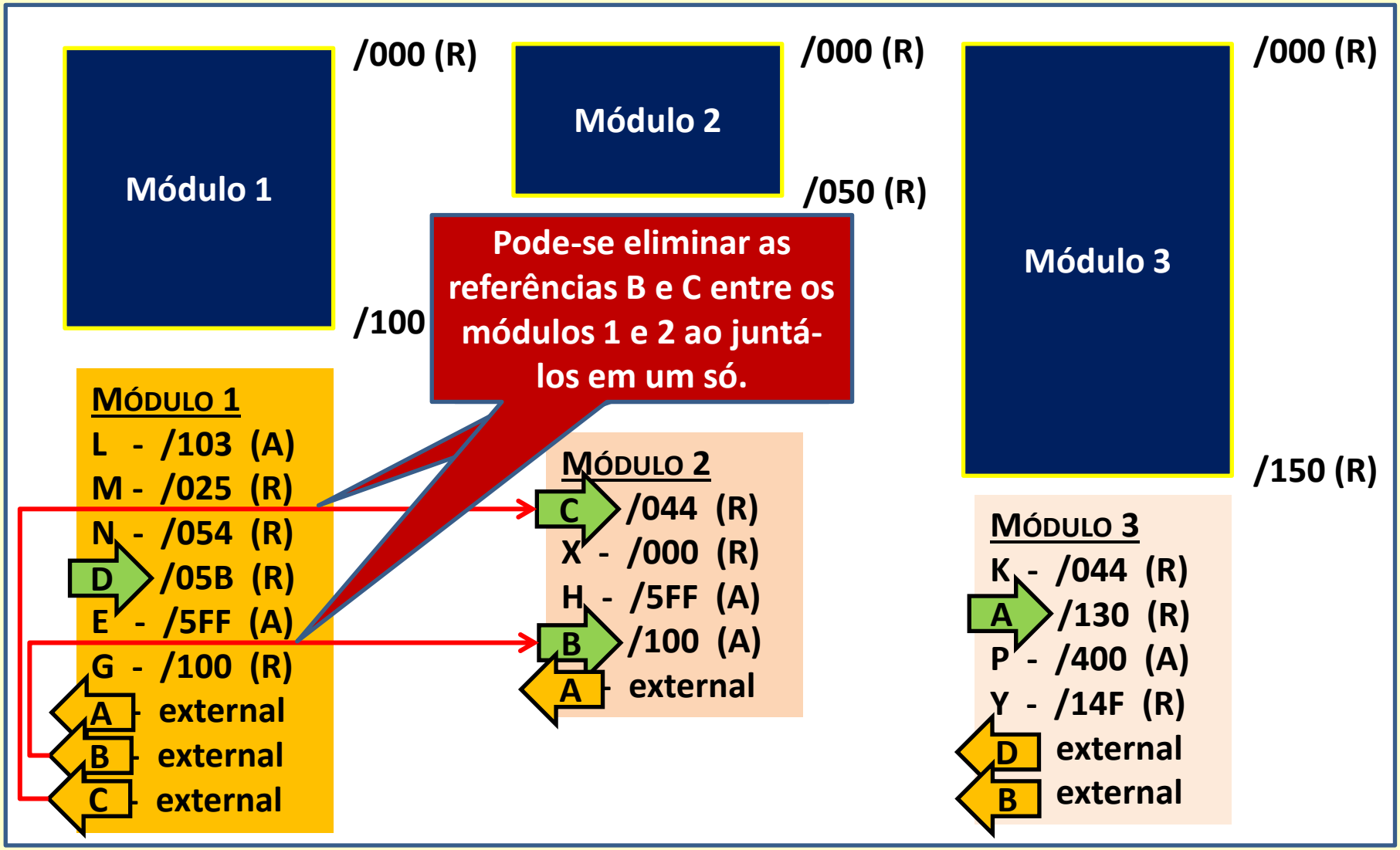


Esquema da ligação das referências simbólicas

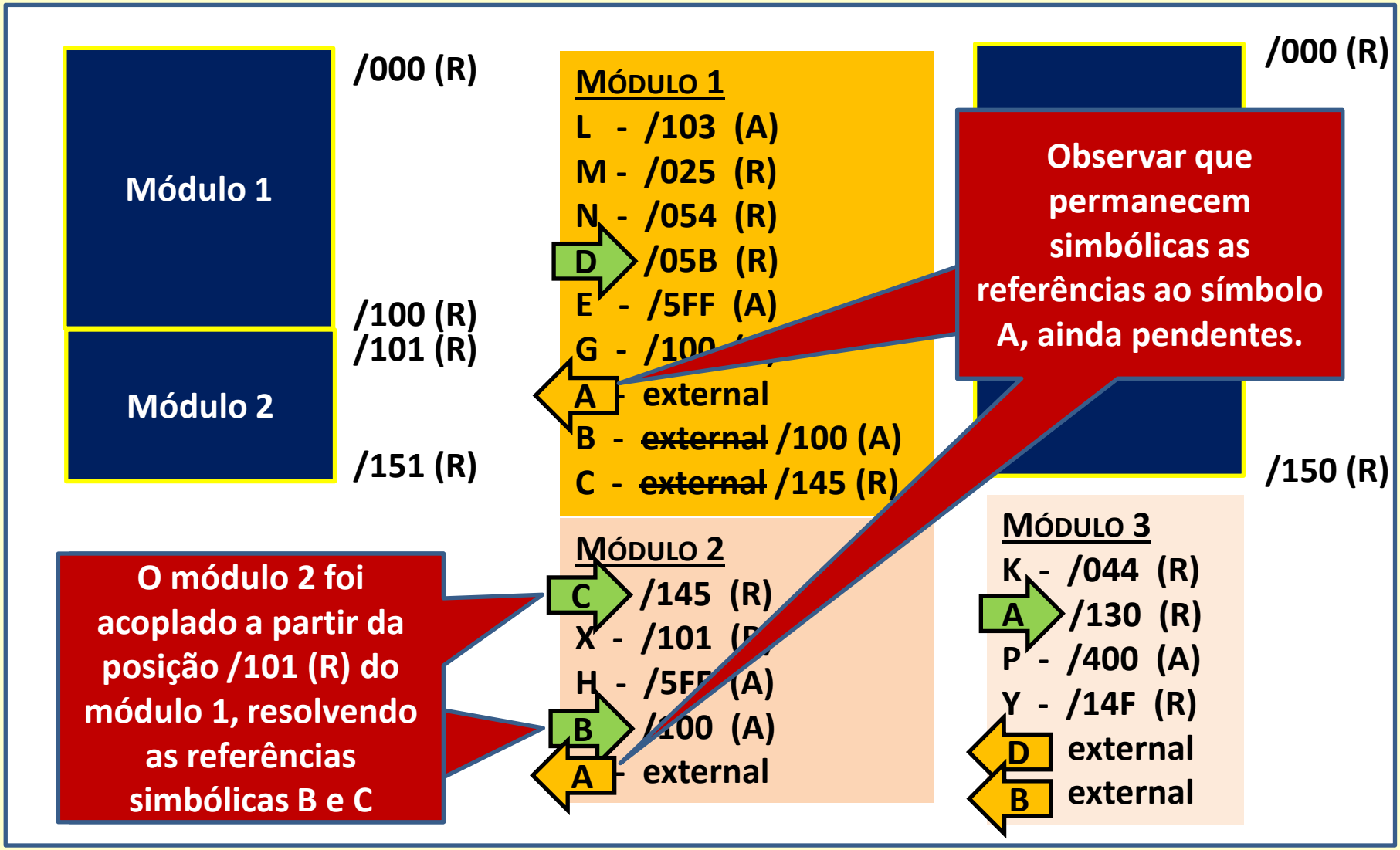
Este diagrama mostra apenas as conexões simbólicas existentes entre os três módulos iniciais, e os endereços associados a elas no módulo a que pertencem.



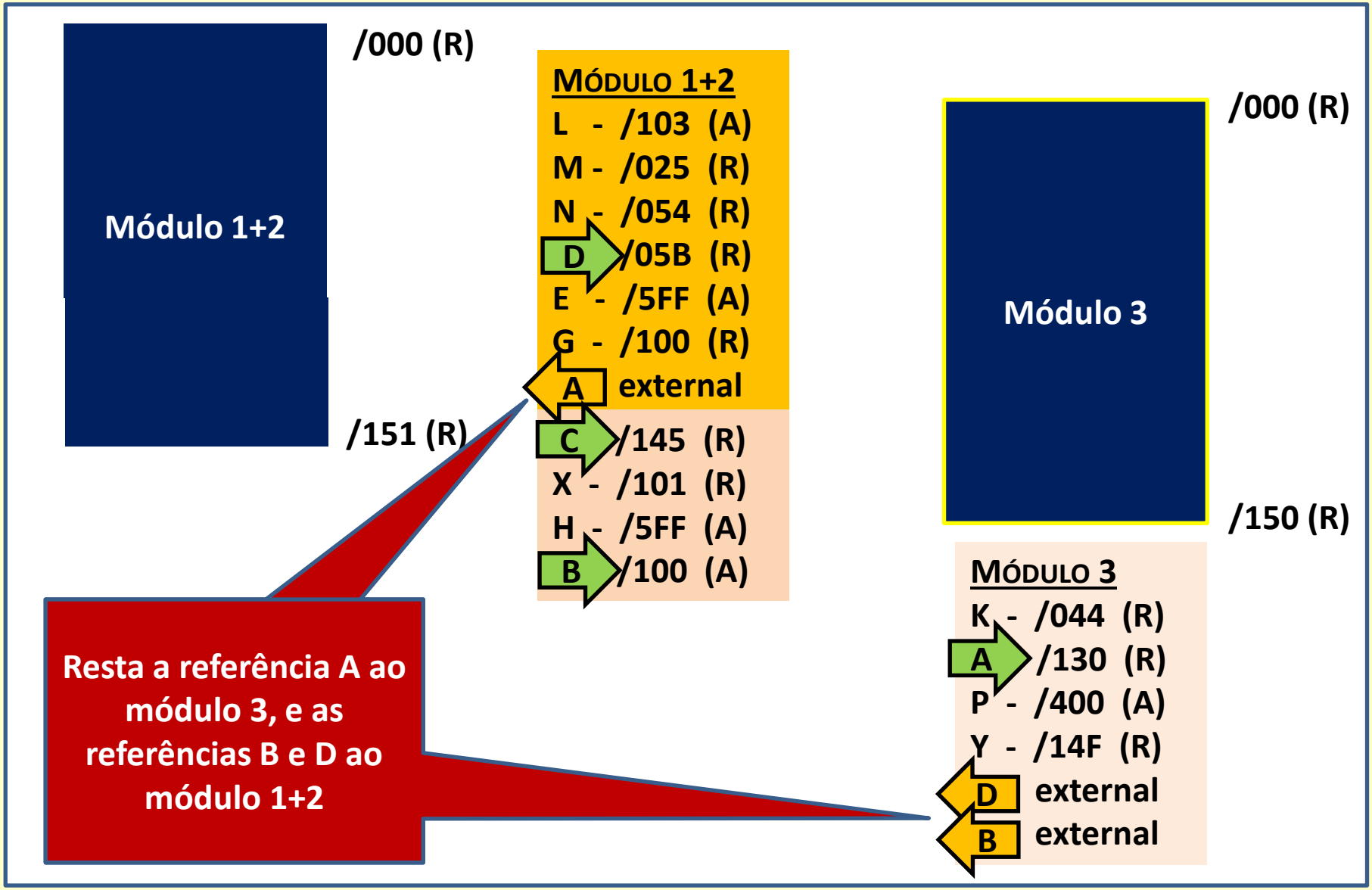
Resolução de referências simbólicas entre os módulos 1 e 2



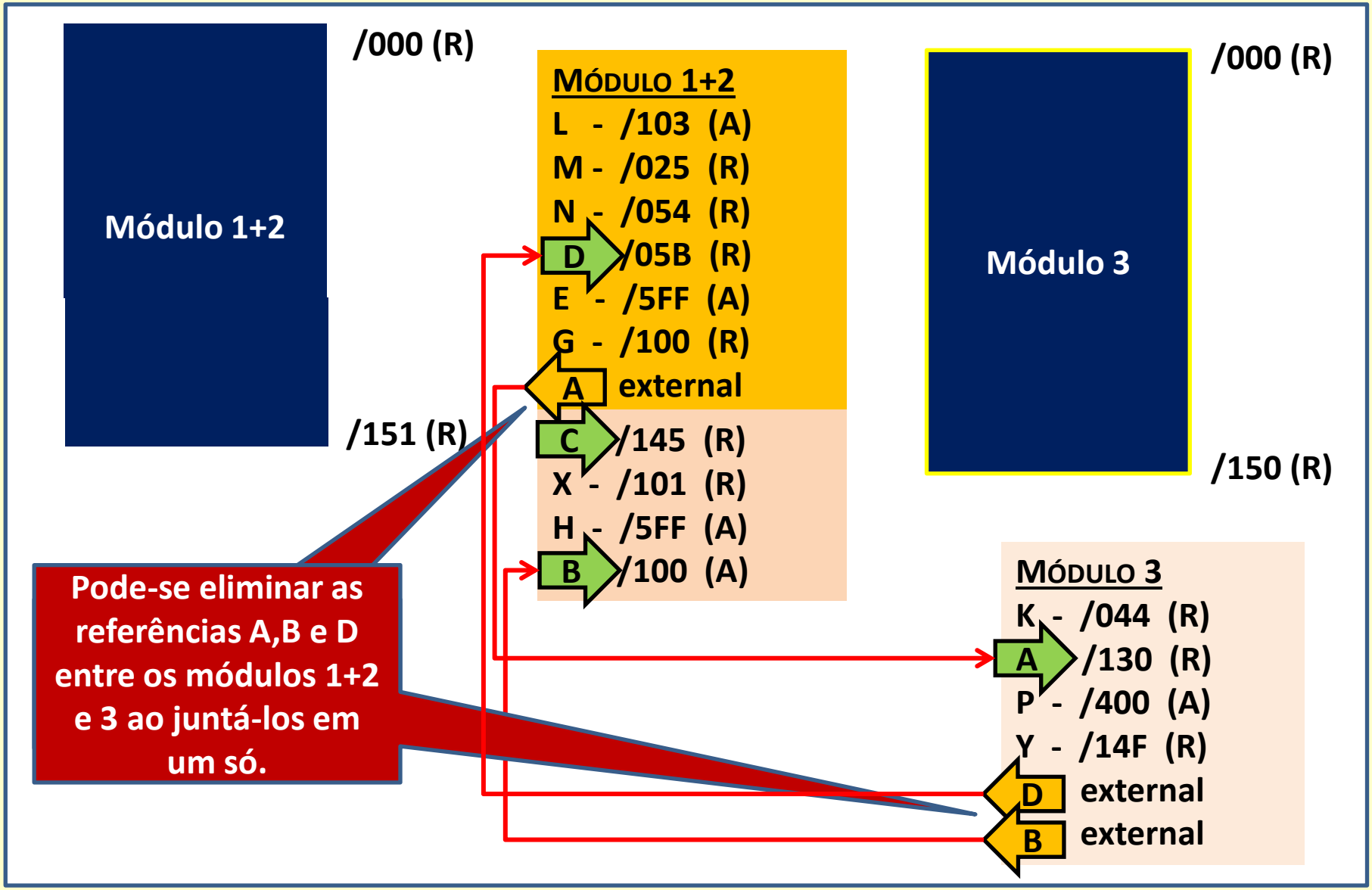
Incorporação do módulo 2 ao módulo 1



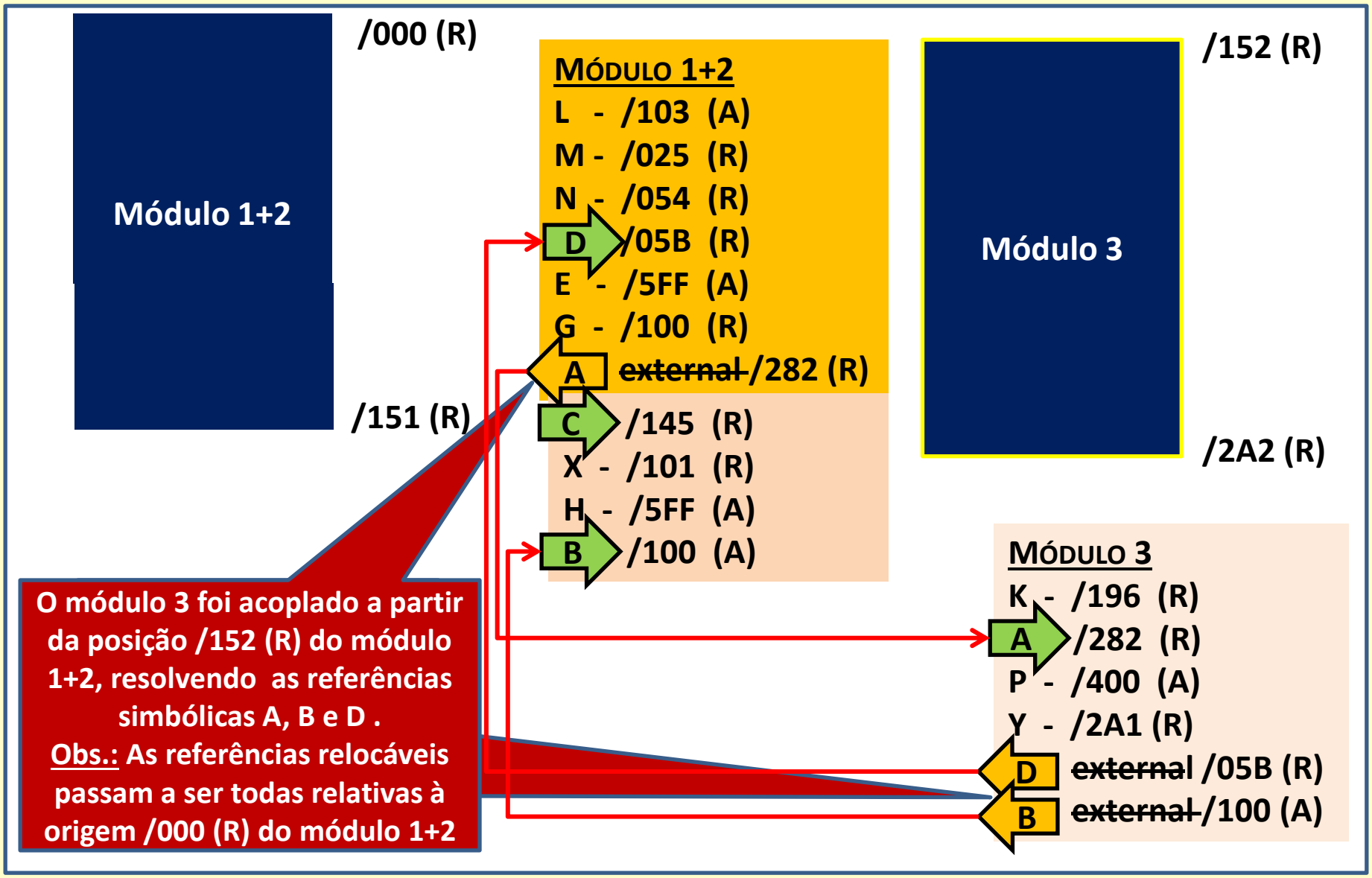
Resultado da fusão dos módulos 1 e 2



Resolução das referências entre 1+2 e 3



Resolução das referências entre 1+2 e 3



Módulo relocável único resultante

MÓDULO 1+2+3

L - /103 (A)
M - /025 (R)
N - /054 (R)
D - /05B (R)
E - /5FF (A)
G - /100 (R)
C - /145 (R)
X - /101 (R)
H - /5FF (A)
B - /100 (A)
K - /044 (R)
A - /130 (R)
P - /400 (A)
Y - /2A1 (R)

Alocação em /500 (A) e relocação final

ENDEREÇOS ABSOLUTOS FINAIS

L - /103 (A)
M - /525 (A)
N - /554 (A)
D - /55B (A)
E - /5FF (A)
G - /600 (A)

C - /645 (A)
X - /601 (A)
H - /5FF (A)
B - /100 (A)

K - /544 (A)
A - /630 (A)
P - /400 (A)
Y - /7A1 (A)

Comentários sobre o exemplo

- Estão envolvidos no processo **três módulos (1, 2 e 3)**
  - O módulo 1 tem D como *entry point* e A,B,C como *externals*
  - O módulo 2 tem como *entry points* B,C e como *external*, A
  - O módulo 3 tem como *entry point* A, e como *externals*, B e D
- Essas relações simplesmente **associam** os ***entry points*** aos correspondentes ***externals,*** em outro módulo.
- Notar que, feitas todas as associações, **nenhum *external*** fica **sem conectar-se ao *entry point*** correspondente.
- **Este mecanismo**, similar ao desempenhado no primeiro passo do montador para a construção da tabela de símbolos, **é executado por um** módulo do sistema de programação que desempenhe o papel de **ligador (*linker*).**

FIM