PCS 3216
Sistemas de Programação

Aula 15

Programação Simbólica Relocável

INTRODUÇÃO

Motivação

- Programas absolutos carecem de modularidade pois estão fortemente vinculados a endereços fixos.
- Esta característica pode ser acrescentada à programação em linguagem simbólica introduzindo-se o conceito de relocabilidade.
- A programação absoluta dificulta também a criação de bibliotecas, e isto pode ser atenuado criando-se módulos, montados separadamente.

Relocação automática

- Há hardwares que podem atenuar esse quadro disponibilizando ao usuário um conjunto de instruções que possibilite a referência a endereços com **relocação automática.**
- Contudo, sua utilização sempre acarreta um ***overhead*** em tempo de execução, ao ser feito o chaveamento do controle entre um módulo e outro.

Referências simbólicas entre módulos

- Os módulos assim concebidos podem **referenciar-se simbolicamente** uns aos outros.
- Operações de **ligação entre módulos** que se referenciam permitem compor **módulos compostos** a partir dos existentes, culminando na obtenção de um único módulo contendo o programa completo.
- Uma **operação de relocação** (correção dos endereços relativos internos aos módulos) permite **transformar esses programas (relocáveis)** em **programas executáveis (absolutos)**.

Linguagens simbólicas e módulos relocáveis

- Linguagens simbólicas que viabilizam a modularização de programas são chamadas ***linguagens simbólicas relocáveis.***
- **Módulos relocáveis** costumam formar programas incompletos, que precisam ligar e **resolver suas referências simbólicas** externas, antes da execução.
- A resolução dessas referências ocorre só quando for conhecida a posição de memória a elas associada.

Ligação e relocação

- O programa só fica pronto para a execução quando todas as suas referências à memória forem convertidas em endereços absolutos. Isto é realizado na época da **relocação dos endereços** relativos presentes no programa.
- A época da resolução das referências simbólicas entre módulos é chamada **época de ligação**, ou de resolução de referências simbólicas (*linking, binding*).

Tratamento de endereços relativos

- Em um **programa relocável:**
  - Os **endereçamentos internos** devem ser **relativos.**
  - Todo módulo relocável costuma ser montado **a partir do endereço relativo zero.**
  - Uma vez decidido alocar um módulo relocável a partir de um endereço absoluto X, todos os endereços internos devem ser **relocados para X posições adiante, na memória.**

Pseudos e código-objeto

- Na linguagem simbólica, o programa absoluto difere do relocável quanto às **pseudo-instruções** disponíveis, que nos programas relocáveis se destinam a tratar conceitos ausentes nos programas absolutos.
- O código objeto **relocável** também difere em estrutura do absoluto, pois contém **meta-dados** diferentes, sendo particularmente significativos os relativos às **referências simbólicas** que participam das **conexões entre módulos**.

Montador e bibliotecas

- Para a elaboração de programas usando a opção relocável, a principal ferramenta é o **montador**, com o qual há a possibilidade de montagem de programas-objeto relocáveis, cujas coleções podem formar ***bibliotecas relocáveis***.
- Muitos montadores relocáveis dão ao usuário a opção de montar programas absolutos, já que os programas absolutos constituem um caso muito particular de formato, dentre os muitos que podem ser tratados pelo montador no caso geral.

Características dos programas relocáveis

- ***Programas relocáveis*** não ficam vinculados a posições fixas de memória.
- A conveniência da reutilização de códigos já validados sugere o uso de ***bibliotecas*** relocáveis.
- ***Programas-objeto relocáveis*** apresentam os mesmos elementos dos programas absolutos, exceto quanto a ***meta-dados*** adicionais e ao uso de ***endereços relativos e simbólicos***.

Endereçamentos

- Códigos-objeto relocáveis apresentam referências a vários tipos de endereços:
  - **Absolutos** – para referenciar posições fixas de memória: endereços do núcleo do Sistema Operacional, nas memórias disponíveis, etc.
  - **Relativos** – usados em referências internas aos programas relocáveis: usam deslocamentos em relação a alguma base de relocação.
  - **Simbólicos** – usados para permitir referências mútuas através de rótulos simbólicos, importados ou exportados entre módulos distintos.

Relocação

- Para executar um programa, todos os endereços relativos devem ser ***relocados***, ou seja, convertidos em endereços absolutos.
- Isto pode ser feito mediante a adição da base de relocação apropriada ao endereço relativo que eles representam.
- Algumas arquiteturas possibilitam a ***relocação dinâmica***, em que essa adição é feita no momento da execução, por hardware, automaticamente.

Relocação dinâmica

- Quando em um módulo todos os endereços são relativos a uma mesma base de referência, diz-se que é ***intrinsecamente relocável***.
- A base de relocação deve ser mantida em um registrador especial (***registrador de base***).
- É preciso **atualizar** o conteúdo do **registrador de base** cada vez que se migra de um módulo para outro, durante a execução do programa.
- Para reduzir o **impacto da relocação dinâmica**, faz-se o possível para que as operações adicionais exigidas sejam **executadas em paralelo** com as demais operações da máquina.

Relocação estática e dinâmica

- Se forem **intrinsecamente relocáveis**, os módulos podem ser executados em **qualquer posição física** da memória **sem alteração**.
- Porém, **nem sempre o hardware** dispõe de recursos que **permitam tal prática**.
- ***Overheads*** podem ser significativos quando houver **frequentes mudanças de contexto**.
- Reduz-se esse *overhead* efetuando uma ***relocação estática*** de grupos de módulos na ocasião da sua ligação, dessa forma **reduzindo o total de módulos**, e portanto a necessidade de mudanças de contexto durante a execução.

Endereçamento simbólico

- Outro tipo de endereçamento tratado pelo montador relocável é o ***endereçamento simbólico***.
- Neste caso, os módulos podem referenciar-se mutuamente ***importando*** e ***exportando*** os nomes de alguns dos seus rótulos (***externals/entry-points***).
- Para tratar a ocorrência de endereços deste tipo, é necessário que eles sejam **convertidos em endereços absolutos ou relativos** antes que possam ser processados da forma anteriormente apresentada para os demais rótulos.
- O módulo do sistema de programação que efetua este tratamento é o ***ligador*** (*linker, binder*).

LIGAÇÃO E RELOCAÇÃO

- Os programas de sistema anteriormente descritos como apoios à programação absoluta oferecem ao usuário ferramentas de boa qualidade, com as quais se pode trabalhar comodamente no desenvolvimento de programas, sem a necessidade de recursos adicionais.
- Apesar disso, entretanto, não permitem a aplicação de vários conceitos modernos ao processo de desenvolvimento de programas, entre os quais um de grande importância é o de apoiar a modularidade.

- Programas escritos em linguagem simbólica absoluta normalmente perdem muito de sua potencial flexibilidade de utilização, uma vez que as linguagens absolutas vinculam o código-objeto a posições fixas de memória, estabelecidas a priori pelo programador.
- Desta maneira, programas escritos em linguagens absolutas só funcionam perfeitamente se seu código-objeto for carregado exatamente nos endereços de memória para os quais tenham sido projetados.

- Isto ocorre, como foi estudado anteriormente, em decorrência dos endereçamentos absolutos existentes nos operandos das instruções de referência à memória.
- Existem processadores cujo hardware oferece recursos de relocação automática das referências à memória, através de registradores especiais da arquitetura, que corrigem dinamicamente os endereçamentos, em tempo de execução das instruções.

- Mesmo nestes casos, existe a necessidade de acertar o conteúdo do registrador toda vez que houver transferência da execução do programa entre duas partes que tenham sido independentemente desenvolvidas.
- De qualquer modo, o uso da programação simbólica absoluta, independentemente do hardware existente, vincula o uso dos programas desenvolvidos, tornando-os muitos dependentes das posições de memória inicialmente idealizadas para seu funcionamento.

- Este tipo de esquema dificulta, inclusive, a criação de bibliotecas de programas de uso geral já testados, e, consequentemente, o intercâmbio de programas desenvolvidos independentemente por vários grupos e em ocasiões diversas.
- Este problema é resolvido pela aplicação do conceito de programação modular, através da qual o programa é dividido em segmentos lógicos, a critério do programador, e cada segmento lógico é desenvolvido como se fosse um todo.

- Implementado desta maneira, um programa constará de um conjunto de segmentos lógicos (módulos), cada qual pode referenciar qualquer dos outros módulos, podendo também ser por eles referenciado.
- Para a obtenção do programa executável a partir dos módulos componentes, e necessário ligá-los diretamente, e acertar convenientemente as referências absolutas a todas as instruções de referência à memória.

- Isto é feito através da aplicação dos conceitos de ***ligação entre módulos*** e de ***relocação*** dos endereços relativos presentes no código.
- Sistemas que operam desta maneira oferecem ao usuário o recurso da ***relocabilidade*** para ser aplicado aos seus programas.

- Programas escritos pelo usuário, visando explorar este recurso, são denominados ***programas relocáveis***.
- Uma linguagem simbólica que facilita ao usuário ferramentas para o desenvolvimento de programas modulares e relocáveis é denominada ***linguagem simbólica relocável.***

- Ao contrário dos programas absolutos, programas desenvolvidos em linguagens simbólicas relocáveis apresentam-se incompletos, logo não podem ser executados diretamente.
- Por ser possível efetuar referências entre módulos independentemente desenvolvidos, os endereços referenciados entre um módulo e outro apresentam-se "***não ligados***", ou seja, não são associados a qualquer posição específica de memória, absoluta ou relativa.

- A vinculação dos endereços não-ligados, contidos em programas relocáveis na forma de "***referências externas***" simbólicas a posições de memória, não é efetuada na ocasião da tradução do programa simbólico para o formato objeto.
- Ao contrário, é postergada até que se conheça a posição de memória, relativa ou absoluta, a ser ocupada por todos esses endereços, externamente referenciados.

- A época em que os símbolos correspondentes às referências externas são associados aos respectivos endereços físicos (absolutos ou relativos) é denominada "***fase de resolução de referências externas***", ou "***fase de ligação entre módulos***" ("***linking***", "***binding***").
- Para que um programa-objeto relocável possa ser executado, ele não pode depender das posições físicas de memória em que será carregado, devendo por isso ser libertado de referências internas absolutas, que o vinculam a posições fixas de memória.

- Assim sendo, o endereçamento interno a um módulo relocável deverá ser sempre relativo.
- Usualmente, convenciona-se que todo módulo tem início em uma "***posição zero***" própria do módulo. à qual todos os endereços internos do módulo são relativos.

- Dessa forma, uma vez decidido, por qualquer método, que o módulo deverá ser carregado em uma posição física X de memória, o módulo poderá ser convertido em um programa executável através de um procedimento denominado "***relocação***", que consiste em "***resolver as referências relativas***", adicionando-se a todas elas a "***base de relocação***" X.

- Do ponto de vista da linguagem de programação simbólica, a diferença fundamental entre suas formas absoluta e relocável reside no repertório das peseudo-instruções disponíveis, já que nos programas relocáveis são identificados diversos novos conceitos, que devem ser representados de algum modo nos programas.

- Outra grande diferença é notada no código-objeto gerado, cujo formato é, usualmente, incompatível com o formato absoluto, por apresentar, em relação a este, um grande número de informações (meta-dados) adicionais, antes dispensáveis ou não justificáveis.

- Do ponto de vista de ferramenta, disponível, em geral, para auxiliar o programador no desenvolvimento de programas relocáveis em linguagem simbólica, pode-se afirmar que o principal programa de suporte é, certamente, o montador.
- Este tipo de esquema viabiliza a elaboração de bibliotecas de programas relocáveis, compostas de módulos independentemente desenvolvidos, e de uso livre para os programadores que tenham acesso às bibliotecas.

- Uma vez efetuada a ligação entre os módulos relocáveis que compõem um programa, é criado um programa-objeto, sem referências externas pendentes, e que ainda pode conter endereços não-resolvidos, por tratar-se de um programa relocável.

- Pela natureza dos programas relocáveis, entretanto, montadores não bastam, exigindo, para completá-los, o auxílio de programas ligadores e relocadores (ou combinação dos dois), alocadores de memória para a escolha da região mais adequada para carregar o programa, bibliotecas de rotinas relocáveis, e desmontadores de programas-objeto relocáveis.

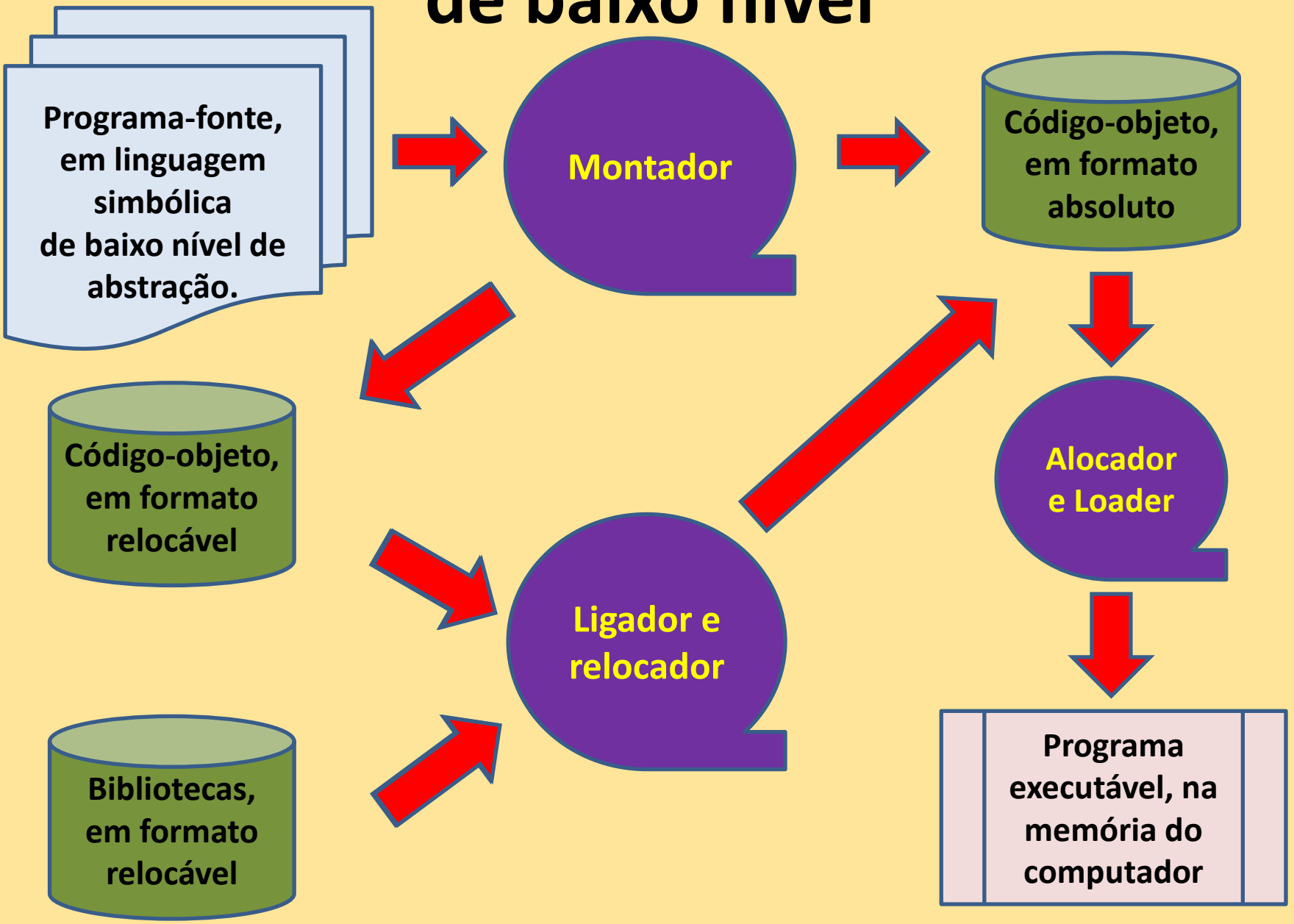
- Com esse conjunto de programas de sistema, o usuário fica em poder de um potencial muito forte de desenvolvimento de programas relocáveis, podendo criar e manter módulos independentes, que poderão servir como pontos de partida para a implementação dos seus programas.

- Adicionalmente, os programas de sistema voltados para a programação relocável oferecem ao usuário, muitas vezes, recursos para a programação absoluta como uma das suas diversas opções de utilização.

- Em particular, os montadores apresentam-se, também, mais ricos em opções, oferecendo dessa maneira ao usuário uma liberdade muito maior de escolha entre os recursos que o sistema lhes disponibiliza.

- Em geral, muitos desses recursos são indevidamente considerados como opções exclusivas dos montadores da linguagem simbólica relocável, quando na verdade poderiam facilmente constar na lógica dos montadores absolutos, ou mesmo operar como programas independentes.

Elaboração de programas em linguagem de baixo nível



PRINCIPAIS CONCEITOS ENVOLVIDOS

- Os **programas absolutos** caracterizam-se. como foi visto, pela sua forte vinculação a **posições físicas de memória**, às vezes já determinadas na época em que o programa é escrito.
- Isto cria um esquema rígido, em que a cada programa ficam associadas regiões de memória preestabelecidas.

- Esta característica seria perfeitamente tolerável se os programas fossem sempre tão diferentes uns dos outros que cada novo programa exigisse um novo desenvolvimento completo.
- Felizmente, apesar de as aplicações dos computadores variarem significativamente, muitas partes dos mais diversos programas sistematicamente se repetem, como acontece com certas sub-rotinas, tais como, por exemplo, as responsáveis pela formatação de entrada e saída, de cálculo de funções transcendentais e matriciais , que são utilizadas em quase todos os programas científicos.

- Por outro lado, isto torna um tanto **complicado** o esquema de **programação absoluta** para ser utilizado no caso geral da programação simbólica, o que motiva a criação de mecanismos de representação de programas-objeto em uma forma tal que, mediante algum processamento prévio, seja possível criar versões absolutas do mesmo, vinculadas à posição de memória que se desejar.

- Nessas representações, os programas apresentam-se na forma denominada ***linguagem de máquina relocável*** ou ***linguagem-objeto relocável*** ou simplesmente ***relocável***.
- Esta forma de representação do programa contém todas as informações exibidas nos programas em forma absoluta, exceto no que tange à origem (posição física) associada ao programa.

Endereçamento absoluto

- Um programa-objeto relocável apresenta, internamente, vários tipos de endereçamento.
- O primeiro, já conhecido, é o próprio endereçamento ***absoluto***.
- Este tipo de endereçamento é, muitas vezes, necessário, uma vez que alguns programas necessitam referenciar determinadas posições físicas invariáveis de memória, a despeito da região de memória em que o programa deverá residir.

- 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 aos eventos de **interrupção**.

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.

- Os deslocamentos são portanto **endereços relativos**, e cada sistema, em geral, convenciona alguns **pontos de referência**, em relação aos quais são expressos, internamente ao módulo relocável, os endereços internos do programa (por exemplo: o início do programa; o início da área de *common*; o início da área de dados).

- Para ser possível executar o programa, todos os endereços relativos devem ser convertidos para endereços absolutos.
- 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.

Relocação

- A soma desse endereço absoluto (***base de relocação***) com o deslocamento (endereço relativo) gera um endereço absoluto, que pode ser finalmente utilizado pelo hardware na execução do programa.
- Esta operação de conversão de endereços relativos é denominada ***relocação***.

Relocação dinâmica

- Há arquiteturas que permitem que a base de relocação seja **dinamicamente adicionada** ao endereço relativo, apenas **no instante da execução** da instrução, postergando ao máximo a relocação, o que aumenta muito a flexibilidade dos programas assim construídos, pois evita a associação prematura de endereços físicos definitivos às instruções.

Registrador de base

- Isto é feito através do que se chama ***relocação dinâmica***.
- A base de relocação é mantida em um registrador especial, em hardware (***registrador de base***), cujo conteúdo deve ser atualizado ao início da execução de cada módulo do programa.
- Nessas máquinas, relocar um módulo consiste, portanto, simplesmente em carregar o registrador de base com o endereço inicial de memória a partir do qual o módulo estiver fisicamente carregado.

- Os operandos das instruções de referência à memória são corrigidos no instante da execução destas instruções por meio da adição do conteúdo do registrador de base, operação esta executada dinamicamente pelo hardware.
- Em geral, o **hardware** costuma efetuar essa operação **em paralelo** com outras atividades do processador, para que não haja gasto adicional significativo de tempo na execução das instruções, como efeito colateral da relocação dinâmica.

- Nessas máquinas, o **código-objeto** pode, portanto, ser feito **intrinsecamente relocável**, o que permite que os programas sejam executados em qualquer região de memória, sem a necessidade de alterações físicas neste código.
- Nem sempre, porém, o hardware está equipado com este tipo de recurso para a relocação automática em tempo de execução.

Mudanças de contexto

- Entretanto, de acordo com a maneira como o programa tiver sido implementado, mesmo que o hardware esteja provido desses recursos, vezes há em que o número de ***mudanças de contexto*** necessárias à execução de um programa pode tornar-se muito grande, forçando uma substituição demasiado frequente do conteúdo do registrador de base.

Relocação estática intermediária

- 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, de modo que seja obtido **um único módulo** composto, em que figurem apenas referências à memória na forma de endereços **relativos a uma só base**.

- Qualquer que seja o caso, portanto, a operação de ***relocação estática***, ou seja, uma correção a priori dos endereços relativos, é de grande importância, e sua aplicação aos programas constitui a denominada ***fase de relocação*** dos módulos que os compõem.

Endereçamento simbólico

- Um terceiro tipo de endereçamento, encontrado em textos-objeto relocáveis, é denominado ***endereçamento simbólico***, e corresponde a **referências a** posições de memória cujos **endereços** são **desconhecidos** no instante em que o programa é escrito, mas cuja existência se conhece, sabendo-se que **pertencem a outros módulos** do programa.

Referências externas

- Seria inútil a possibilidade de se efetuar uma programação modular se os módulos de que o programa se compõe fossem completamente estanques.
- Assim, é conveniente que cada módulo possa utilizar-se das funções executadas pelos demais, o que é feito mediante referência mútua, pelas instruções de referência à memória, através do uso do conceito de ***referências externas*** aos módulos.

Ponto de acess (*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*** ou ***pontos de acesso*** ao módulo.
- Cada módulo pode, usualmente, apresentar um ou mais pontos de acesso (***entry points***), 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***).

- 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.

- 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 se torna **apto para a execução** quando **todas as referências** do tipo ***import*** tiverem sido **satisfeitas** pela presença de declarações dos mesmos símbolos, como sendo do tipo ***export***.