PCS 3216
Sistemas de Programação

João José Neto

Aulas 11 a 14 – Montadores de um passo e
Alguns aspectos de sua implementação

MONTADORES DE UM SÓ PASSO

Montadores de dois passos

- Os montadores de dois passos, como foi visto, dividem as tarefas de montagem dos programas simbólicos entre os dois passos em que são estruturados,
  - Especializando cada um deles
  - Propiciando desta maneira a obtenção de programas potentes e eficientes.

Inconveniência

- Muitas vezes, deseja-se construir montadores para máquinas ou situações em que a dupla leitura física do texto simbólico é indesejável
  - Quando os periféricos de leitura/gravação são muito lentos
  - Quando a operação de montagem é uma atividade muito solicitada no sistema
  - Quando o sistema desempenha muitas atividades e está sobrecarregado.
  - Quando se deseja que a operação de montagem seja muito rápida, por exemplo, para atendimento muito frequente de atividades discentes.

Dois passos

- Contorna-se o problema em montadores de dois passos:
  - Memorizando (em disco, por exemplo) enquanto leem, no primeiro passo, uma cópia do texto simbólico
  - Posteriormente, para completar a operação de montagem, a imagem do programa-fonte construída no primeiro passo é lida, a partir do disco, pelo segundo passo do montador (só a leitura a partir de um arquivo em disco e não a partir de periféricos lentos já colabora para reduzir substancialmente o tempo gasto pelo montador na atividade de releitura do programa fonte).

Sem disco

- Há casos em que se usa, alternativamente, a opção de fazer a operação de montagem em um único passo:
  - Por imposição das decisões de projeto
  - Quando não há disponibilidade de um dispositivo de memória de massa para esta tarefa
  - Quando não se dispõe de periféricos de entrada/saída rápida para ler/gravar o programa fonte/objeto.
  - Quando as exigências da aplicação inviabilizam o uso de outras soluções

Montadores de um só passo

- Para isto, são construídos os chamados *montadores de passo único*, que leem o programa fonte uma só vez.
- Há dois tipos principais de montadores de um só passo:
  - Montadores *load and go* ou *assemble and go* (em que o código-objeto é gerado diretamente na memória e fica pronto para ser executado imediatamente)
  - Montadores *que geram código-objeto carregável*, a ser posteriormente introduzido na memória para execução. Nesta modalidade, o código-objeto assume a forma de um programa de preenchimento, que deposita na memória o código-objeto na exata sequência em que as referências à frente na memória vão sendo conhecidas pelo montador, durante o processo de montagem do programa.

Funcionalidade idêntica

- Estes programas devem executar funcionalmente todas as tarefas que foram descritas para os montadores de dois passos.
- Deverão fazê-lo, entretanto, através de uma única leitura do texto simbólico.

Referências à frente

- Assim, será necessário resolver novamente alguns problemas já solucionados pelo esquema em dois passos, os quais voltam à tona pela imposição de uma única leitura do programa simbólico.
- O problema mais sério refere-se à dificuldade de obtenção do código de máquina para as instruções de referência à memória que têm como operandos endereços simbólicos ainda não definidos (referências à frente).

Solução

- Em montadores de um só passo, soluciona-se esta dificuldade através da memorização em uma tabela de códigos pendentes.
- Os elementos desta tabela referem-se a
  - Todas as instruções que apresentem operandos com endereços ainda não resolvidos
  - Incorporam as informações conhecidas acerca dos respectivos operandos
- Destinam-se a propiciar o preenchimento dessas informações assim que se tornarem disponíveis, durante a montagem do programa:
  - Na ocasião da definição do símbolo do qual dependem
  - O código completo pode ser finalmente montado e gerado no programa-objeto ou então depositado na posição correta de memória.

Geração do código-objeto

- Do ponto de vista de geração do código-objeto
  - As instruções com operandos não resolvidos geram, por exemplo, códigos incompletos.
  - Sua utilidade é a de reservar área de memória para o código definitivo, cujo endereço e formato são conhecidos, mas seu valor ainda não foi determinado.
- Na ocasião da definição do símbolo do qual dependem
  - As instruções que se apresentam ainda incompletas são preenchidas com as informações faltantes, obtendo-se dessa forma o código final associado à instrução.

Lista de Pendências

- A Lista de Pendências costuma ser organizada como lista ligada, cujos elementos, além do ponteiro para o próximo elemento da lista, contêm no seu campo de informação triplas (EI, OPI, DI), nas quais referenciam, respectivamente:
  - o endereço de memória da instrução correspondente
  - o código de operação associado à instrução
  - o deslocamento (translação) a ser sofrido pelo endereço de definição do símbolo referenciado para formar corretamente o operando da instrução.

Estruturas de dados

- Possível implementação da lista de pendências
  - Listas ligadas implementam muito adequadamente as listas de pendências, que representam o conjunto dos códigos incompletos presentes no programa-objeto em construção.
  - A cada símbolo, associa-se uma lista de referências à frente ainda não resolvidas, cada qual acompanhada de eventuais informações complementares sobre o operando associado.
  - Os elementos da lista de pendências são a ela incorporados toda vez que for encontrada uma referência a um símbolo indefinido, e são dela removidos sempre que o montador determinar o endereço a que se refira um desses símbolos.
  - Em algumas implementações, as listas de pendências são fisicamente armazenadas na mesma estrutura de dados da tabela de símbolos, funcionando assim como sua extensão.

Inclusão de pendências

- Na ocasião da inclusão de mais um elemento na lista de referências, pode-se verificar se o símbolo referenciado já está definido ou não.
- Se isso não ocorrer, cria-se e inclui-se na lista de pendências correspondente ao símbolo uma tripla (EI, OPI, DI), onde
  - EI é o valor do contador de instruções
  - OPI é o código de operação da instrução que executou a referência
  - DI é o deslocamento imposto pelo seu operando

Resolução de pendências

- Posteriormente, na ocasião da definição do símbolo, a lista de pendências a ele associado deverá ter cada um dos seus elementos processado e eliminado.
- Para isso constrói-se, para cada um dos elementos da lista, um bloco de código-objeto contendo as informações registradas na lista de pendências, complementadas com a informação do endereço de memória associado ao símbolo recém-definido.
- Após a geração de tal código, os elementos da lista de pendências referentes ao símbolo em questão podem ser descartados.

Geração descontínua de código

- Do ponto de vista da geração de código, se esta for efetuada diretamente na memória, não haverá qualquer necessidade de processamento adicional.
- Se, entretanto, a geração for feita em meio externo, como por exemplo em fita ou arquivo, certamente haverá uma descontinuidade na sequência de bytes gerados, em termos dos endereço R de memória ocupados.

Esvaziamento do bloco de código

- Assim sendo, deverá haver obrigatoriamente um esvaziamento forçado do último bloco de código, então incompletamente preenchido.
- Isso deve ser feito para liberar espaço no bloco de código, cedendo área para a geração das informações de preenchimento das lacunas anteriormente criadas pelo montador, correspondentes a referências à frente.

Backtracking

- Esta técnica pode ser classificada na classe dos algoritmos de *backtracking*, anteriormente mencionados.
- O conteúdo do programa-objeto assim construído reflete a sequência como a memória é preenchida pelo montador à medida que este vai conhecendo os endereços associados aos símbolos.
- Preserva, portanto, a ordem de preenchimento da memória, na sequência exata em que as informações vão sendo coletadas ou construídas ao longo da montagem.

Interpretação

- Assim, a fita gerada pelo montador de um passo não representa uma imagem do conteúdo da memória.
- Trata-se, na realidade, de um código ou programa de preenchimento da memória.
- Naturalmente, a execução deste programa é efetuada pelo carregador absoluto, o qual executa a sua interpretação.
- Ao final dessa interpretação, os bytes do código-objeto representado nesse programa estará devidamente carregado nos endereços corretos.
- **Exercício**: Analise o carregador absoluto que você construiu (*loader*) e identifique os vários elementos citados na discussão acima.

Listagens

- Um outro problema advindo da supressão de um passo na lógica do montador, manifesta-se nas operações de listagem do programa simbólico, ao lado do correspondente programa numérico.
- Optando-se por efetuar a listagem à medida que se monta o programa, surge um problema: havendo falta de informação, os códigos numéricos correspondentes às pendências não poderão ser listados na ocasião.

Listagem de códigos pendentes

- Alguns montadores de um passo imprimem os códigos gerados exatamente na mesma ordem em que os mesmos forem sendo gerados na fita
- Em outras palavras, os códigos pendentes são impressos logo após os símbolos dos quais dependem serem definidos (ocorrerem como rótulos)
- Outros montadores imprimem códigos incompletos, com indicações de que estão pendentes
- Ao final da montagem, imprimem o conteúdo da tabela de símbolos, com cujo auxílio o usuário pode compor manualmente os endereços omitidos.

Listagens melhores

- As duas opções mencionadas fornecem listagens que se mostram incômodas para o usuário.
- Alguns montadores de um passo, mais sofisticados, geram em arquivos em disco imagens da listagem do programa e do código associado.
- Nesses arquivos, esses montadores alteram o conteúdo dessa imagem da listagem na ocasião da resolução das referências à frente.
- Com esta última solução, é possível obter, com montadores de um passo, listagens idênticas às produzidas pelos de dois passos.

Desvantagem

- Como desvantagem principal do uso de montadores em um só passo, em geral é necessário ter à disposição uma considerável área de memória de massa.

- Incorporando as soluções apresentadas às ideias já utilizadas na lógica do montador de dois passos, é possível obter um montador capaz de efetuar em um único passo a montagem dos seus programas-fonte.

Conclusão

- Há naturalmente vantagens e desvantagens de se utilizar, na confecção de um montador, uma lógica de um ou de dois passos.
- Normalmente, deve-se levar em conta a finalidade a que o montador se destina, a frequência com que será utilizado, o tempo de retorno desejado, o tamanho médio dos programas que serão traduzidos, e o tamanho máximo dos mesmos, que irá servir para definir a dimensão de suas tabelas e áreas de dados, e mesmo, no caso particular, o tipo mais adequado de montador.