TP4
Software Básico

Pedro Araujo Pires 03/07/2011

Introdução

Neste trabalho foi desenvolvido um editor de ligação para módulos de programas feitos para a máquina virtual desenvolvida na trabalho prático 1. O editor de ligação recebe vários módulos, gerados pelo montador, e combina todos eles em um executável para a máquina virtual.

Implementação

Para implementar o editor de ligação, foi utilizada uma estratégia similar à descrita no livro texto da disciplina: ele primeiro junta os arquivos em um único espaço de endereçamento, calcula as novas posições de memória que os símbolos de cada módulo referenciam, e escreve um um arquivo o programa executável, com as referências corretas à memória.

Para que seja possível executar esses passos, o montador implementado no trabalho prático 2 teve de ser modificado. Quando a tradução é feita, o módulo gerado contém informações adicionais a serem usadas pelo editor de ligação. Esse informações são o total de linhas de código no módulo, e a tabela de símbolos do módulo. Além disso, quando há uma referência a um símbolo desconhecido, esse símbolo é mantido no código.

Quando o editor de ligação é executado, primeiramente é montado um *array* com os nomes dos módulos a serem *linkados*. Esse *array* determina a ordem em que os módulos serão traduzidos para o programa executável. O módulo com o código do início do programa sempre ocupa a primeira posição do *array*. Os outros módulos ocupam as posições de acordo com a ordem em que eles foram informados na chamada do editor de ligação. Essa parte corresponde à fase de alocação, descrita no livro.

Os arquivos são lidos na ordem do *array*, e é contruída uma tabela de símbolos geral, que contém todos os símbolos declarados em todos os módulos. A tabela de símbolos e o tamanho de cada módulo são usados para isso. Se todos os módulos fossem combinados em um arquivo único, essa seria a tabela de símbolos gerada pelo montador. Essa parte do processo corresponde à fase de ligação.

A última parte do processo consiste em criar o arquivo com o programa executável. Para fazer isso, os módulos são lidos em sequência (ignorando as informações adicionais geradas pelo montador), e as instruções lidas são escritas no arquivo de saída. Quando uma referência externa é encontrada, a tabela de símbolos é consultada, e a posição correta de memória é escrita no arquivo. Para fazer isso, é usada uma estratégia similar à do montador: é mantido um contador de instruções (ILC), e esse contador é usado para o cálculo da posição correta de memória. Essa parte corresponde à fase de relocação, descrita no livro.

Execução

Para compilar o programa, deve-se usar o comando `make`, dentro da pasta com o código fonte. Ele irá gerar um executável, que recebe pelo menos dois argumentos: `-m` e `-o`, que indicam o módulo que contém o início do programa, e o nome do arquivo executável a ser gerado. Além dessas opções, também devem ser informados os nomes de todos os módulos a serem processados peloo editor de

ligação. Apesar de ser especificado que os nomes dos módulos viriam antes das opções `-m` e `-o`, a implementação permite que eles sejam informados em qualquer ordem. Abaixo um exemplo de chamada do editor de ligação:

```
./ligador modulo1.o modulo2.o modulo3.o -m main.o -o output.maq
```

Testes

Para testar o editor de ligação, foi feito um programa que simula uma calculadora. Ele possui vários módulos, e funciona da seguinte forma: ele lê três inteiros, A, B e C, e executa uma operação entre os números A e C, de acordo com o código B. Quando B é 1, a operação feita é A + C. Para B igual a 2, a operação é A – C. Para B igual a 3, a operação é A * C. Para B igual a 4 a operação é A / C, e para B igual a 5, a operação é A ^ B. Quando a operação de divisão é feita, é impresso o quociente e o resto da divisão.

Esse programa está dividido em 6 módulos, de forma que há um módulo para cada operação, e um módulo com o programa principal. Cada módulo é chamado através da instrução CALL, e os parâmetros (A e C) são passados para os procedimentos através da pilha de execução. Ao final da execução, o resultado de cada operação também é passado através da pilha. Esse comportamento permitiu que o módulo da multiplicação fosse usado em dois lugares: no programa principal, quando a operação é de multiplicação, e no módulo de potência, que necessita da mltiplicação para calcular o resultado.