TP3
Software Básico

Pedro Araujo Pires 12/06/2011

Introdução

Neste trabalho foi desenvolvido um expansor de macros para programas feitos para a máquina virtual desenvolvida na trabalho prático 1. O expansor gera um arquivo que serve de entrada para o montador. O expansor reconhece duas pseudo-instruções, BEGINMACRO e ENDMACRO, que indicam, respectivamente, o início e o final de uma definição de macro. Além disso, cada macro pode ter no máximo 1 argumento.

Implementação

Para implementar o montador, foi utilizada uma estratégia similar à do montador de dois passos: o arquivo com o programa é lido uma vez, e todas as definições de macros são armazenadas. Em seguida, o arquivo é lido novamente, e onde há uma chamada de macro, é feita a expansão da macro. As instruções que não fazem parte de uma definição de macro, e não são chamadas de macro permanecem inalteradas. As chamadas de macro são trocadas pelo corpo da macro, e as definições de macro são retiradas.

Para armazenar as definições de macros, foi utilizada uma *struct* (`Macro`) que armazena o nome da macro, o nome do argumento (se houver), um *array* de *strings*, que são as instruções, e o total de instruções presentes na macro.

Para criar essas estruturas, sempre que é lida uma pseudo-instrução BEGINMACRO, é criada uma nova macro. Todas as instruções subsequentes são adicionadas à estrutura, até que se leia a pseudo-instrução ENDMACRO. Se for lida uma instrução que não faz parte de uma definição de macro, ela é ignorada. Essa nova macro é então adicionada a um *array* de macros. Ao final de leitura do arquivo, é adicionada mais uma posição no *array*, que aponta para NULL. A única finalidade dessa posição é indicar o fim do *array*.

Depois que todas as macros foram reconhecidas, é feita uma segunda leitura do arquivo. Nessa leitura, a cada instrução lida, é feita uma verificação se a instrução é uma chamada de macro. Se não for uma chamada de macro, a instrução é escrita no arquivo de saída sem nenhuma alteração (inclusive com comentários). Caso seja uma chamada de macro, a macro é expandida no arquivo de saída. Em qualquer momento, se for lida a psudo-instrução BEGINMACRO, todas as instruções subsequentes são ignoradas, até que se leia a pseudo-instrução ENDMACRO.

Quando está acontecendo a expansão da macro, primeiramente e verificado se a macro possui argumento ou não. Caso ela não possua, todas as instruções são escritas no arquivo de saída em ordem. Se a macro possuir um argumento, a cada instrução processada é verificada se o operando da instrução é o argumento da macro. Caso seja o argumento, é feita a substituição.

A especificação do trabalho diz que não há definição de macros dentro de outras definições de macros, mas não fala nada sobre chamada de macros dentro de macros. Então, no processo de expansão de uma macro, é verificada se uma instrução é uma chamada de macro. Caso seja, a função que expande uma macro é chamada de novo (recursivamente). Depois de expandida, a expansão da macro inicial continua.

Execução

Para compilar o programa, deve-se usar o comando `make`, dentro da pasta com o código fonte. Ele irá gerar um executável, que deve ser executado da seguinte forma:

```
./expansor <entrada> <saida>
```

O primeiro argumento é o nome do arquivo de entrada, e o segundo argumento é o nome do arquivo que será gerado.

Testes

Para testar o expansor foram implementados dois programas, que obrigatoriamente usam macros na sua implementação. O primeiro deles, `potencia.macro`, é um programa que lê dois números, a e b, e calcula o valor de a^b. Este progrma foi utilizado para testar a chamada de macros dentro de outras macros. Foram definidas 3 macros, de forma que na primeira macro existe uma chamada para a segunda, e na segunda uma chamada para a terceira.

O segundo programa, `mediana.macro`, é um programa que lê 7 números, e imprime a mediana deles. O problema deste programa está em ordenar os números. Para isso foi utilizado o algoritmo de inserção para fazer esta ordenação, devido à sua simplicidade de implementação. Foi feita uma macro que acha o menor elemento, uma que acha o segundo menor, uma para o terceiro e uma para o quarto. O quarto elemento depois de ordenado é a mediana dos números.

Todos os testes foram bem sucedidos, de forma que depois que as macros foram expandidas, o programa resultante foi passado pelo montador, e a saída do montador foi executada pela máquina virtual.