Trabalho Prático de Compiladores I
Etapa 4 e final: Compilador Integrado

Felipe Buzatti e Letícia Lana Cherchiglia
{buzatti,letslc}@dcc.ufmg.br

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

29 de Junho de 2011

1 Introdução

O objetivo desse trabalho é apresentar um compilador integrado para a mini-linguagem MiniL e gerar código para a máaquina virtual (MEPA), de forma a ser possível executar programas em MinL. A máquina virtual MEPA foi definida anteriormente e seus códigos não foram alterados.

Desta forma, neste trabalho temos a junção das etapas anteriores (Tabela de Símbolos, Análise Léxica e Análise Sintática), com a adição de mais uma etapa (Análise Semântica) e dos códigos referentes à maquina virtual MEPA e ainda a realização de testes para verificar a corretude de nosso compilador.

2 Descrição

Em relação às partes integrantes de nosso sistema, temos que somente a análise sintática e a geração de código foram realizadas com o auxílio de ferramentas (Bison). Todas as outras partes foram implementadas por nós, o que gerou um aprendizado maior no final do trabalho, apesar do esforço ter sido também maior.

Neste relatório iremos focar apenas na análise semântica e no modo de execução do compilador como um todo, visto que as outras partes já foram descritas com detalhes nos trabalhos anteriores e não vemos necessidade de repetí-las neste documento.

2.1 Análise Semântica

Infelizmente neste trabalho não foi possível realizar tudo o que foi proposto na especificação, mas temos um compilador que funciona corretamente para a maioria dos comandos e expressões, exceto para alguns casos, como por exemplo:

- Procedimentos passados como parâmetros
- Arranjos
- Gotos

Também não está sendo realizada a verificação de tipo dos parâmetros em um procedimento e utilizar um return no meio deste não funciona, o retorno só ocorre no final deste.

Em relação a detalhes de nossa implementação, decidimos fazer a verificação de tipos adicionando um campo de tipo na nossa estrutura lista de comandos, com os valores 0 para integer, 1 para real, 2 para booleam e 3 para character. Desta forma, ao realizarmos uma atribuição ou uma soma, por exemplo, era somente verificar se os tipos das listas envolvidas na operação eram os mesmos. Também foi utilizado o campo tipo da tabela de símbolos para a tipagem de variáveis, que já existia anteriormente e agora passou a fazer sentido.

Além disso, nosso compilador já gera o código final (MEPA) diretamente após a análise semântica, o que consideramos mais efetivo, visto que nesta etapa do trabalho teríamos que realizar todo o processo de compilação como um todo.

3 Formato dos dados

3.1 Entrada

A entrada do programa pode ser feita em qualquer arquivo de texto puro. Obviamente a execução correta pressupõe que esse arquivo de texto contenha programas válidos da linguagem minL, (respeitados os casos nos quais o nosso compilador não funciona - vide acima), mas o sistema integrado compilador+MEPA é capaz de capturar erros para situações inesperadas, abortando a análise e imprimindo uma mensagem de erro explicativa.

3.2 Saída

Caso o arquivo de entrada esteja correto, haverá a saida de um arquivo de texto contendo o código MEPA e também a execução deste programa, visto que a execução do sistema nada mais é do que a entrada de um arquivo MinL seguida da saída de seu código correspondente em MEPA e a interpretação deste código na MEPA.

4 Organização do trabalho

Dentro da pasta tpfinal, temos os seguintes arquivos:

- **Pasta MEPA:** contém os códigos da MEPA, sem alteração, e um makefile próprio
- **Pasta scr:** contém os códigos de nosso compilador e um makefile próprio
- **Pasta testes:** contém os testes realizados, assim como o default (exemplo.txt)
- **Arquivo instrucoes.txt:** contém instruções para a execução do teste default (exemplo.txt)
- **Arquivo makefile:** makefile geral, que atua chamando os outros dois makefiles. Está executando com o teste default (exemplo.txt)

5 Compilação e Execução

Foram criados 3 makefiles neste trabalho, um para a MEPA, outro para o compilador e outro que chama os dois primeiros.

O makefile principal se encontra na pasta tpfinal. Com o comando 'make', teremos a compilação tanto da MEPA como do nosso compilador. Com o comando 'make run', teremos a mesma compilação descrita anteriormente aliada à execução do arquivo de teste default, exemplo.txt, que se encontra na pasta Testes.

Para executar somente o compilador passa-se como primeiro argumento um arquivo que contém o programa a ser compilado. E como segundo argumento, opcional, um arquivo para o qual o código MEPA será escrito. Caso esse segundo argumento não se faça presente o código MEPA é direcionado para a saída padrão. Assim, temos como exemplos válidos:

- Usa a entrada e a saída padrão: $< nome\_do\_programa >$
- Usa apenas a saída padrão: $< nome\_do\_programa >< arquivo\_entrada >$
- Lê diretamente dos arquivos: $< nome\_do\_programa >< arquivo\_entrada >$ $< arquivo\_saida >$

Para a realização dos testes através do sistema integrado (compilador+MEPA), recomendamos o seguinte: utilizar o makefile principal, alterando nas linhas seguinte deste o nome do arquivo que se deseja testar, lembrando que este deve estar na pasta testes:
$ENTRADA = testes/'arquivo\_entrada.txt'$
$SAIDA =' arquivo\_saida.txt'$

6 Testes

Para testar a corretude do analisador foram feitos os testes que se encontram dentro da pasta de mesmo nome, além de listados em anexo. Também se encontram em anexo os respectivos resultados destes testes.

7 Conclusão

Através deste trabalho pudemos compreender melhor o funcionamente de um compilador e conhecer a máquina virtual MEPA, assim como a linguagem MinL. Entretanto, alguns itens do compilador proposto por nós não puderam ser corretamente implementados, o que nos leva a crer que o produto final de nosso trabalho poderia ser melhorado.

Desta forma, acreditamos que o trabalho foi um exercício válido dentro da matéria, mas que poderíamos ter todas suas funcionalidades implementadas de modo correto caso nos fosse dado mais tempo, pois a implementação dos itens faltantes não é trivial.

8 Bibliografia

1 http://www.gnu.org/software/bison/ Bison - GNU parser generator.

2 Monica S. Lam e Ravi Sethi Alfred V. Aho. Compiladores - Princípios, Técnicas E Ferramentas.