Trabalho Prático 5
Segunda, Julho 2, 2018

1 Introduction

Neste trabalho você irá implementar o gerador de código para Cool. O gerador de código irá receber como entrada a AST anotada que foi saída do TP4, e irá gerar o código correspondente em assembly MIPS.

Seu gerador será considerado correto se o código gerado funcionar corretamente, de acordo com a semântica operacional da linguagem, que está descrita no *Cool Reference Manual*.

Também é importante que leiam a seção *The Runtime System* do *A Tour of the Cool Support Code*. Nela estão listadas diversas informações importantes sobre a interface entre o código gerado e o runtime system.

2 Arquivos e diretórios

Os arquivos relacionados a este TP podem ser encontrados na pasta *assignments/PA5*.

De forma similar aos outros TPs, *make* irá criar a estrutura associada ao *mycoolc* e *make cgen* irá compilar o código relativo ao gerador de código.

Os arquivos que vocês poderão alterar para este TP são:

- cgen.cc
  Este arquivo irá conter quase todo seu código deste TP. O ponto de entrada para seu gerador de código é o método **program class::cgen(ostream&)**, que é chamado na raiz da AST. Além de algumas constantes, nós provemos funções para emitir instruções MIPS, um esqueleto para codificação de strings, inteiros, booleans e um esqueleto de uma tabela de classes (**CgenClassTable**). Você é livre para modificar o código.

- cgen.h
  Cabeçalho de cgen.cc. Você pode adicionar o que quiser neste arquivo. Ele provê classes para implementar o grafo de herança.

- emit.h
  Este arquivo contém várias macros de geração de código usadas na emissão de instruções MIPS, dentre outras coisas. Você pode modificar este arquivo.

- cool-tree.h
  Contém as declarações das classes dos nós da AST. Você pode adicionar declarações de campos ou métodos nas classes deste arquivo. A implementação dos métodos que você declarar deverão ser feitas no arquivo cgen.cc.

- cgen_supp.cc
  Contém código de suporte para o gerador de código. Contém várias funções que podem ser úteis. Você pode adicionar código neste arquivo, mas não modifique o que já está aqui.

3 Design

Em alto nível, o seu gerador de código deverá:

1. Determinar e emitir código para constantes globais, tais como protótipos de objetos.
2. Determinar e emitir código para tabelas globais, como a **class_nameTab**, a **class_objTab**, e as tabelas de dispatch.
3. Determinar e emitir código para o método de inicialização de cada classe.
4. Determinar e emitir código para cada definição de método.

Existem várias maneiras de se escrever o gerador de código. Uma maneira seria realizá-lo em 2 passos. No primeiro passo, decida como será o layout dos objetos de cada classe, particularmente o offset em que cada atributo se localiza. Usando esta informação no segundo passo, recursivamente visite cada feature e gere código de máquina correspondente para cada expressão.

Existem diversos aspectos que você deve levar em consideração quando for projetar o gerador de código:

- Ele deve funcionar corretamente com o runtime system do Cool, que é explicado no *A Tour of the Cool Support Code*.
- Você deve ter uma visão bem clara da semântica de Cool. Ela é descrita no *The Cool Reference Manual*.
- Você deve entender o conjunto de instruções do MIPS. Uma visão geral pode ser encontrada no arquivo *spim.pdf*, que pode ser encontrado no Moodle.
- Você deve decidir quais invariantes o código gerado irá obedecer e esperar (por exemplo, quais registradores irão ser salvos, quais podem ser sobreescritos etc).

O código que você irá gerar não precisa ser igual ao do coolc. Coolc inclue um alocador de registradores simple e algumas outras pequenas mundaças que não são necessárias para este TP. No entanto, o código gerado deve executar corretamente com o runtime system, obedecendo a semântica da linguagem.

3.1 Verificação de erros

No final do manual do Cool, são listados seis erros de runtime. Você deve lidar com os três primeiros - dispatch em void, case em void, case sem match - e imprimir uma mensagem de erro apropriada antes de abortar o programa. Você pode deixar o próprio SPIM lidar com divisões por zero. Os 2 últimos erros - substring com range inválido e heap overflow - são de responsabilidade do runtime system em trap.handler. Na tabela *Labels defined in the runtime system*, em *A Tour of the Cool Support Code*, são listadas algumas funções que exibem mensagens de erros para você.

3.2 Garbage Collection

Seu gerador de código deve trabalhar corretamente com o generational garbage collector do Cool runtime system. A função **code_select_gc** gera código que seta opções do GC a partir de flags passadas pela linha de comando. Existem 3 flags que afetam o garbage collector: -g, -t e -T. Por padrão, o garbage

collector é desabilitado. A flag -g o habilita. Enquanto habilitado, não só ele recupera memória, mas também verifica que "-1" separa todos os objetos do heap, verificando, portanto, se o programa (ou o próprio coletor) não sobreescreveram acidentalmente o fim de um objeto. A flag -t faz com que o coletor execute em toda alocação. A flag -T causa código extra a ser gerado, que realiza mais testes de validação durante a execução do coletor.

O jeito mais simples de começar a implementação é não usar o coletor. Quando você decidir usar o coletor, verifique a interface do Garbage Collector que é descrita na seção *The Garbage Collector* do *A Tour of the Cool Support Code*. Não é trivial garantir que seu gerador de código funcione corretamente com o garbage collector em todas as circunstâncias.

4 Testes

Você irá executar seu gerador de código usando mycoolc, um shell script que junta o gerador com as outras fases do compilador. mycoolc pode receber uma flag, -c, para depurar o gerador de código. Usar esta flag simplesmente faz com que a variável global `cgen_debug` seja setada. Isso pode ser útil para você depurar seu código.

4.1 Spim

SPIM é um interpretador de MIPS e não um assembler de fato. Uma dificuldade que advém disso é que se seu código fizer referência a rótulos indefinidos, o erro só irá acontecer quando o código correspondente for executado de fato. Além disso, o erro só é reportado para rótulos indefinidos que aparecem na seção de código do seu programa. Se você tiver definições de dados constantes que referem a um rótulo indefinido, o SPIM não só não lhe avisará, como irá assumir o valor 0 para esses rótulos.

5 Submissão

Os seguintes arquivos deverão ser enviados em um zip: cool-tree.h, cgen.h, cgen.cc, cgen_supp.cc e emit.h.