Projeto de Compilador
E7 de **Otimização**

Prof. Lucas Mello Schnorr
schnorr@inf.ufrgs.br

1 Introdução

A sétima e última etapa do trabalho de implementação de um compilador para a Linguagem consiste em tópicos relacionados às técnicas de otimização de código. O grupo tem a liberdade de escolher **uma entre duas opções** seguintes: opção (A) Implementar alguma técnica de otimização; ou opção (B) Gerar grafo de controle de fluxo para uma entrada

Atenção: Escolha uma opção e informe o professor no arquivo `README.txt` na raiz da solução desta etapa.

2 Opção (A) – Implementar Técnica de Otimização

2.1 Otimização de código

Nesta opção você deve selecionar ao menos uma técnica de otimização de código, que pode ser sobre código intermediário ou sobre o código assembly, e demonstrar seu funcionamento. Caso a técnica de otimização seja aplicada no código intermediário, deve ser possível de observar um efeito no código assembly. A escolha da(s) técnica(s) é livre. Para atingir a nota máxima nesta etapa, pede-se que você consiga mostrar que o código gerado ficou diferente (melhorado, por exemplo, mais curto). Para isso, altere a função principal `main` de maneira que ela aceite um parâmetro `-O` que, se informado, ativa a otimização implementada, e caso ausente, gera código normal como na etapa anterior, sem otimização.

2.2 Exemplos

O grupo deve fornecer dois exemplos de códigos a serem compilados, que incluem avaliação de expressões aritméticas e comandos de fluxo de controle, que levem a uma otimização quando o parâmetro `-O` é informado. O grupo é encorajado a demonstrar, também através de exemplo, que isso gerou otimização no tempo de execução do programa.

2.3 Documentação

Inclua no arquivo de texto puro `README.txt` na raiz do projeto uma mini-documentação que ilustre os exemplos e usos das funcionalidades implementadas nesta etapa.

2.4 Dicas Básicas

2.4.1 Entrada e Saída Padrão

Organize a sua solução para que o compilador leia da entrada padrão o programa em nossa linguagem e gere o programa em assembly na saída padrão. Dessa forma, pode-se realizar o seguinte comando (`etapa7` é o binário do compilador) quando deseja-se gerar código sem otimizações:

```
./etapa7 < entrada > saida.s
```

A geração de código com otimizações deve ser da seguinte forma (perceba o parâmetro `-O` para o programa, que deve ser tratado através dos parâmetros `argc` e `argv` da função principal):

```
./etapa7 -O < entrada > saida.s
```

O código assembly sem e com otimização deverá ser capaz de ser reconhecido e montado para um programa executável através do seguinte comando (onde `programa` é um programa executável):
`gcc saida.s -o programa`

3 Opção (B) – Gerar grafo de controle de fluxo

3.1 Grafo de controle de fluxo

Nesta opção o grupo deve criar um grafo de controle de fluxo de blocos básicos, considerando a definição de instruções líderes vistas em aula. O grupo pode gerar a partir de código intermediário ou assembly, desde que demonstre o funcionamento efetivo da geração do grafo de controle de fluxo. Faça com que o grafo seja gerado no formato DOT, visualiável com uma ferramenta estilo `xdot` (conforme usamos na E3).

3.2 Exemplos

O grupo deve fornecer dois exemplos de códigos a serem compilados, que incluem avaliação de expressões aritméticas e comandos de fluxo de controle, que levem a grafos de controle de fluxo de blocos básicos.

3.3 Documentação

Inclua no arquivo de texto puro `README.txt` na raiz do projeto uma mini-documentação que ilustre os exemplos e usos das funcionalidades implementadas nesta opção.

3.4 Dicas Básicas

3.4.1 Entrada e Saída Padrão

Organize a sua solução para que o compilador leia da entrada padrão o programa em nossa linguagem e gere

o grafo de fluxo de controle de blocos básicos no formato DOT na saída padrão. Dessa forma, pode-se realizar o seguinte comando (`etapa7` é o binário do compilador) quando deseja-se gerar o grafo de fluxo de controle.

```
./etapa7 < entrada > CFD.dot
```

A Arquivo `main.c`

Utilize a função principal no arquivo `main.c` semelhante aquela já implementada na etapa anterior. O grupo deve modificá-la para implementar as funcionalidades necessárias da etapa corrente. Não esqueça de liberar a memória corretamente, como uma boa prática de programação.