MC011 - Laboratório de Construção de Compiladores

1o. Semestre de 2012

Turmas A e B

Aulas | Atendimento | Avaliação | Projetos | Links | Referências | Notas

Avisos

26/06
As notas Finais estão disponíveis aqui.

01/06
As notas do Projeto 2 estão disponíveis aqui.

1/06
As instruções de submissão foram incluídas no Projeto 3.

7/05
A especificação do Projeto 3 foi liberada abaixo. Faremos a apresentação deste projeto no lab as 14h desta quarta, 9/05.

7/05
As instruções para submissão do projeto 2 foram atualizadas abaixo.

2/05
Como foi discutido em aula hoje, a entrega e avaliação da Etapa 2 foi adiada para dia 16/5. O enunciado da Etapa 3 será colocado no ar semana que vem e sua entrega e avaliação estão mantidas para dia 20/06.

25/04

Como o conjunto de instruções do x86 é muito grande, segue uma lista com sugestões de instruções a serem implementadas na Etapa 2.

20/04
As notas do Projeto 1 estão disponíveis aqui.

13/03
A especificação do projeto 2 já está no ar. Veja na seção Projetos. Na próxima aula faremos uma explicação detalhada do projeto.

13/03
Disponibilizamos alguns casos de testes para o Projeto 1. Você pode encontra-los aqui. Consulte este link regularmente pois novos testes podem ser adicionados a qualquer momento.

7/03
Você pode consultar os grupos já cadastrados aqui

7/03
Atenção,todos os grupos deve ser cadastrados antes da entrega do Projeto 1. Cadastre seu grupo aqui

7/03
· **Atenção, por motivo de viagem do professor foi feita uma pequena alteração no calendário. A avaliação presencial do Projeto 1 será dia 4/4. A data de entrega permanece inalterada.**

13/02
· Página da disciplina no ar. Confira critérios de avaliação e calendário.

Aulas

Turmas A e B
· Qua: 13-16h, salas IC 351, 302 e 303

Professores

Contato
· Professor: Sandro Rigo (sandro AT ic dot unicamp dot br)
· Monitor: Alex Bredariol Grilo (abgrilo AT gmail dot com)
· OBS.: Quando enviar um e-mail favor colocar no subject [MC011], caso contrário você corre sério risco de seu

email ser filtrado como spam.

Avaliação

Avaliação

Teremos três projetos a serem elaborados (descrição, data de entrega e pesos abaixo) ao longo do semestre. Cada entrega será composta de uma avaliação presencial e da entrega do pacote com o código desenvolvido. A nota de cada etapa será composta por Pi = 0,3*(AP)+0,7*(I). Onde AP é a avaliação presencial e I é a implementação. A avaliação presencial é obrigatória a todos os alunos. A não participação nesta avaliação implica em nota zero na etapa correspondente. A média do desempenho será calculada por:
M = 0,3 * P1 + 0,35*P2 + 0,35*P3
Se M >= 5,0 o aluno aprovou-se. Caso contrário o aluno estará reprovado.
A entrega de todos os projetos é obrigatória. Nota menor do que 2,5 em qualquer um dos projetos acarreta automaticamente a reprovação com média final MF = Min (4,0; M).

Fraudes

Qualquer tentativa de fraude implicará em nota ZERO no projeto correspondente, reprovando automaticamente todos os envolvidos.

Calendário

- 28/03: Entrega do Projeto 1
- 04/04: Avaliação presencial do Projeto 1
- 16/05: Entrega e avaliação presencial do Projeto 2 <== NOVA DATA !!
- 20/06: Entrega e avaliação presencial do Projeto 3

Projetos

Projeto 1

Na primeira etapa será implementado um parser que converterá um subconjunto de latex para html.

O cabeçalho do arquivo latex que poderá conter:

- Definição de pacotes (\usepackage{nomepacote}): deverá ser ignorado

- Definição de do título (\title{titulo}): ver sobre o comando \maketitle adiante
- Definição do autor(\author{nomeautor}): deverá ser ignorado

O corpo do documento em latex poderá ter textos, texto sob o modo matemático ou comandos.

O modo matemático será demarcado somente pelo sinal cifrão (cuidado que \$ poderá ser utilizado para mostrar o caracter $ ao invés de entrar no modo matemático). O seu conteúdo deve ser renderizado como o modo matemático do Latex (existem bibliotecas javascript que podem ajudar nesta parte).

Os comandos latex que poderão estar incluídos no arquivo serão:

- \maketitle: irá mostrar o título descrito pelo \title no cabeçalho (você pode supor que sempre que um \maketitle estiver presente, existe um \title também).
- negrito: \textbf{texto1} deverá ter como resultado texto em negrito
- itálico : \textit{texto1} deverá ter como resultado texto em itálico
- listas não numeradas: \begin{itemize } \item texto1 \item texto2 \end{ itemize } deverá gerar uma lista com os itens texto1 e texto2 (detalhe as listas podem estar encadeadas)
- imagens: \includegraphics{arquivo} deverá mostrar a imagem arquivo

O projeto será implementado utilizando qualquer ferramenta, porém recomenda-se a utilização da ferramenta ANTLR. Note que a especificação do projeto está consideravalmente livre, então use sua criatividade para implementar o proposto.

Veja alguns casos de teste aqui.

Submissão

A submissão dos arquivos do projeto deverão ser feitos através do Susy . As Senhas foram enviadas por email, se você não recebeu a sua, entre em contato com o monitor da disciplina.

No susy deverá ser submetido um arquivo .tar.gz que deverá conter a seguinte estrutura na sua raiz:

- Um diretório chamado raXXXXXX_raYYYYYY, se você está fazendo o projeto em duplas ou raXXXXXX se está fazendo o projeto individualmente. Dentro deste diretório, devem aparecer os

seguintes arquivos:

- Um diretório chamado src, onde você deverá colocar todo o conteúdo de código do seu projeto (arquivo com gramática, código fonte, arquivos de bibliotecas, etc.). **Atenção:** não colocar binários de qualquer tipo (inclusive .class) do seu projeto.
- Um arquivo Makefile que deverá responder aos seguintes comandos:
  - 'make need_install'. Este comando deverá imprimir '1' caso seja necessário instalar algum pacote para compilar ou rodar seu programa. Caso não seja necessário, esse comando não deverá fazer nada
  - 'make compile'. Este comando deverá preparar seu projeto para ser rodado (se precisar compilar ou configurar alguma coisa, coloque isso dentro desse comando). Após a execução desse comando, o seu projeto deverá estar pronto para ser executado.
  - 'make run INPUT=<arquivo de entrada> OUTPUT=<arquivo de saída>'. Esse comando deverá rodar seu projeto, lendo o arquivo tex de entrada e escrevendo a saída no arquivo html passado.
- Opcionalmente, escreva um arquivo LEIAME.txt. Use esse arquivo para informar qualquer coisa que achar pertinente sobre seu laboratório (fez coisas a mais, a menos, precisa instalar alguma coisa, etc.). **Atenção:** caso seja necessária a instalação de algum pacote, este arquivo é obrigatório e deve conter informações detalhadas sobre os pacotes que devem ser instalados (links para o projeto, versão instalada, como instalar, etc).
- Opcionalmente, você pode colocar qualquer coisa que achar pertinente dentro de um diretório chamado 'misc'. Neste caso, escreva no arquivo LEIAME.txt o que há dentro desse diretório.

Atenção: O não cumprimento de qualquer requisito de submissão influenciará na nota do grupo!.

Projeto 2

Este segundo projeto abordará o tópico de geração de código.

Teoria

A parte teórica da necessária para a implementação do laboratório foi vista no curso de MC910, e pode ser revista nos slides da disciplina ou capítulo 9 do Appel (2a edição). É importante ressaltar que conhecimentos sobre Árvores de representações intermediária, também vistos na disciplina MC910 e contidos no capítulo 7 do Appel (2a edição), são necessários.

Descrição

Você deverá implementar uma classe na linguagem Java que recebe uma lista de árvores e sobre elas executa um algoritmo de seleção de instruções (maximal munch, por exemplo) e devolve uma lista com as instruções referentes a essas árvores. Para mais detalhes sobre os nós que podem estar nessas árvores, você pode verificar o livro do Appel (2a edição) na seção 7.1 ou na documentação das classes . As instruções deverão ser instruções assembly para a arquitetura x86, e você deverá utilizar as classes auxiliares para essas instruções . Note que a arquitetura é diferente da arquitetura utilizada pelo livro do Appel, portanto a parte da implementação do livro (seção 9.3) não necessariamente diz respeito ao que deve ser implementado no projeto (porém pode ajudar a entender melhor o que fazer).

Uma estrutura inicial do projeto pode ser encontrada aqui.

Submissão

A submissão dos arquivos do projeto deverão ser feitos através do Susy . Deverá ser enviado um arquio .tar.gz e seu conteúdo deve seguir a estrutura passada (obviamente, você deve alterar o seu RA e de sua dupla ou remover o segundo RA). Se você decidiu separar a implementação em várias classes, altere o Makefile para que ele compile e rode corretamente. Novamente, o não cumprimento de qualquer requisito de submissão influenciará na nota do grupo!

Projeto 3

Neste terceiro projeto você deverá implementar a otimização de Constant Propagation para o compilador.

Teoria

Para a implementação dessa otimização, serão necessários os conceitos de Reaching definitions slides da disciplina ou capítulo 17 do Appel (2a edição). A descrição de Constant propagation se encontra no capítulo 17 do Appel (2a edição).

Descrição

Para essa otimização, primeiramente você terá que implementar a análise de fluxo de dados no compilador de Reaching Definitions. A partir daí, você deverá implementar a otimização de Constant Propagation usando essa análise. O Constant Propagation deverá alterar o assembly das instruções afetadas.

Uma estrutura inicial do projeto pode ser encontrada aqui.

Submissão

A submissão dos arquivos do projeto deverão ser feitos através do Susy . Deverá ser enviado um arquio .tar.gz e seu conteúdo deve seguir a estrutura passada (obviamente, você deve alterar o seu RA e de sua dupla ou remover o segundo RA). Se você decidiu separar a implementação em várias classes, altere o Makefile para que ele compile e rode corretamente. Novamente, o não cumprimento de qualquer requisito de submissão influenciará na nota do grupo!

Links

Referências Principais

Notas de Aula

Modern Compiler Implementation in Java
Andrew Appel, 2a Edicão

Compiladores : Princípios, Técnicas e Ferramentas
Aho, Sethi & Ullman

Implementação de Linguagens de Programação
Kowaltowski, Editora Guanabara Dois, 1983.