PCS2056 – Linguagens e Compiladores

Assunto: Linguagens de paradigma imperativo e sua compilação

Palavras-chave:
Construções sintáticas básicas;
Máquina de execução subjacente;
Constantes;
Variáveis;
Expressões;
Atribuições de valor;
Seqüências;
Comandos condicionais;
Comandos iterativos;
Teorema de Böhm-Jacopini;
Estruturas de dados homogêneas;
Estruturas de dados heterogêneas;
Ponteiros e estruturas dinâmicas;
Procedimentos e Funções;
Parâmetros e Argumentos;
Programa principal;

Atividade prática (para desenvolver em aula e trazer pronto na aula de 11.09.2008 – próxima aula):

1. Identifique, em alguma linguagem de programação imperativa que você conheça bem, cada um dos conceitos mencionados acima.
2. Procure, através de exemplos, determinar as diversas formas que cada um dos comandos pode assumir. Se possível estabeleça, ainda que informalmente, uma forma geral para cada comando, de modo que as suas diversas variantes sejam todas casos particulares dessa forma geral.
3. Considerando que uma linguagem de montagem será usada como código-objeto, estabeleça correspondências entre os diversos comandos da linguagem de alto nível e a correspondente seqüência de comandos em linguagem de montagem que lhe seja equivalente.

PCS2056 – Linguagens e Compiladores

Assunto: Análise léxica;
Enunciado da **parte 1** do trabalho: Construção de um **analisador léxico**
Data de entrega: **25 de setembro de 2008**

Palavras-chave:

Análise léxica
texto-fonte
caracteres ASCII
identificadores
inteiros sem sinal
caracteres especiais
espaçadores
palavras reservadas
comentários
token ou átomo = (classe, valor)
expressões regulares
autômatos reconhecedores
transdutores sintáticos
listagem do texto-fonte
expansão de macros

Questões:

1. Quais são as funções do analisador léxico nos compiladores/interpretadores?

2. Quais as vantagens e desvantagens da implementação do analisador léxico como uma fase separada do processamento da linguagem de programação em relação à sua implementação como sub-rotina que vai extraindo um átomo a cada chamada?

3. Defina formalmente, através de expressões regulares sobre o conjunto de caracteres ASCII, a sintaxe de cada um dos tipos de átomos a serem extraídos do texto-fonte pelo analisador léxico, bem como de cada um dos espaçadores e comentários.

4. Converta cada uma das expressões regulares, assim obtidas, em autômatos finitos equivalentes que reconheçam as correspondentes linguagens por elas definidas.

5. Crie um autômato único que aceite todas essas linguagens a partir de um mesmo estado inicial, mas que apresente um estado final diferenciado para cada uma delas.

6. Transforme o autômato assim obtido em um transdutor, que emita como saída o átomo encontrado ao abandonar cada um dos estados finais para iniciar o reconhecimento de mais um átomo do texto.

7. Converta o transdutor assim obtido em uma sub-rotina, escrita na linguagem de programação de sua preferência. Não se esqueça que o final de cada átomo é determinado ao ser encontrado o primeiro símbolo do átomo ou do espaçador seguinte. Esse símbolo não pode ser perdido, devendo-se, portanto, tomar os cuidados de programação que forem necessários para reprocessá-los, apesar de já terem sido lidos pelo autômato.

8. Crie um programa principal que chame repetidamente a sub-rotina assim construída, e a aplique sobre um arquivo do tipo texto contendo o texto-fonte a ser analisado. Após cada chamada, esse programa principal deve imprimir as duas componentes do átomo extraído (o tipo e o valor do átomo encontrado). Faça o programa parar quando o programa principal receber do analisador léxico um átomo especial indicativo da ausência de novos átomos no texto de entrada.

9. Relate detalhadamente o funcionamento do analisador léxico assim construído, incluindo no relatório: descrição teórica do programa; descrição da sua estrutura; descrição de seu funcionamento; descrição dos testes realizados e das saídas obtidas.

10.Explique como enriquecer esse analisador léxico com um expansor de macros do tipo #DEFINE, não paramétrico nem recursivo, mas que permita a qualquer macro chamar outras macros, de forma não cíclica. (O expansor de macros não precisa ser implementado).