Roteiro para Apresentação do Relatório de Linguagens e Compiladores – PCS2056

Definição da Linguagem de Alto Nível:

1. Defina os conceitos fundamentais usados em linguagens de programação imperativas que você vai utilizar na sua linguagem. Procure, através de exemplos, determinar as diversas formas que cada um dos comandos pode assumir. Estabeleça, ainda que informalmente, uma forma geral para cada comando, de modo que as suas diversas variantes sejam todas casos particulares dessa forma geral.
2. Complete a descrição informal da linguagem acima, definindo, da maneira que achar mais conveniente, as formas sintáticas que julgar estarem omitidas.
3. Determine se a linguagem definida é suficiente para que com ela seja possível a construção de programas segundo o teorema de Böhm-Jacopini? Justifique. Caso a resposta seja negativa, complete-a para que tal limitação seja revertida.
4. Escreva (mostre no relatório) pequenos programas usando esta linguagem. Muita atenção à pontuação.
5. Explicite a linguagem de montagem que será utilizada como código-objeto.
6. Considerando a linguagem de montagem que 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.
7. Defina formalmente a linguagem assim especificada, usando cada uma das três notações metalingüísticas apresentadas anteriormente: BNF, notação de Wirth e Diagramas de Sintaxe. Muita atenção aos detalhes da notação: presença de <> em tomo do nome dos não-terminais em BNF, aspas envolvendo os terminais na notação de Wirth, o ponto final depois de cada regra na notação de Wirth, a ausência de notação para fechamentos em BNF, etc.
8. Crie, a partir da gramática, um conjunto de arquivos contendo pequenos programas corretos, escritos na nova linguagem, os quais deverão ser utilizados para as atividades de teste do futuro reconhecedor sintático da linguagem.

Análise Léxica:

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

Análise Sintática:

11. Aplique o método de construção do analisador sintático baseado no autômato de pilha estruturado sobre a gramática desenvolvida para a sua linguagem de programação. Implemente o analisador ou o reconhecedor sintático no computador.
12. Prepare em um arquivo um texto de entrada contendo um programa escrito na linguagem de programação definida pela gramática da segunda parte do projeto. Esse programa deve utilizar pelo menos uma vez cada uma das formas sintáticas permitidas pela gramática. Use os programas desenvolvidos no item 8.
13. Teste o programa alimentando-o com as saídas do analisador léxico desenvolvido na primeira parte deste projeto, quando aplicado ao texto de entrada contido no arquivo preparado no item anterior.

Prof. Ricardo Luis A. Rocha

Definição do Ambiente de Execução:

14. Defina a estrutura geral do ambiente de execução da sua linguagem. Explique como deve ser efetuada a seqüência de chamada e retorno de uma sub-rotina, definindo o registro de ativação utilizado e seu tamanho, como são acessados os parâmetros e as variáveis, como e quando são criados os registros de ativação.
15. Mostre a fórmula geral para calcular, em relação ao endereço do apontador de registro de ativação, os endereços virtuais dos parâmetros conforme o tipo do elemento. Idem para as variáveis locais.
16. Para o caso de sub-rotinas aninhadas, explique como acessar variáveis locais do bloco aninhador dentro do bloco aninhado.
17. Escreva na linguagem de montagem, para a máquina objeto definida, rotinas para tratamento de estrutura de pilha (push/pop/reset).
18. Escreva na linguagem de montagem, para a máquina objeto definida, rotinas para tratamento de entrada e saída (para/de uma região de memória).
19. Explique a estrutura dos programas montador e carregador. Como serão executados os seus programas na máquina virtual?

Estruturas de Dados e Algoritmos do Compilador:

20. Projete uma rotina responsável pela montagem e pela busca de informação em uma **tabela de símbolos (= tabela de nomes)**, a ser usada para guardar os nomes das variáveis, e de uma **tabela de atributos**, na qual são informados o seu tipo, uma indicação sobre o fato de já ter sido declarada ou não, referenciada ou não, e um indicador da posição de memória onde as mesmas foram alocadas.
21. Projete um algoritmo para a **tradução das declarações de variáveis simples: inteira, booleana, real**. Este algoritmo deve promover a alocação de espaço para as variáveis, e estabelecer a correspondência entre as variáveis indicadas na declaração e sua respectiva área alocada na memória, contabilizando o uso da memória e registrando as variáveis e seus atributos nas tabelas de símbolos e de atributos.
22. Estenda o algoritmo anterior de forma que traduza também **declarações de agregados** homogêneos (vetores e matrizes). Esse procedimento deverá alocar espaço para conter na memória os elementos do agregado, e também construir e depositar na memória, para cada matriz ou vetor declarado, o correspondente descritor, registrando tudo isso nas tabelas de símbolos e de atributos.
23. Projete um algoritmo que, dados os índices e o descritor de um agregado homogêneo, **localize** na memória o elemento referenciado, **extraindo** daí o valor corrente ou então nele **depositando** um novo valor fornecido.
24. Idem para **agregados heterogêneos**.

Semântica Dinâmica:

25. Escolha as convenções de representação das variáveis e constantes aritméticas e booleanas: formatos internos para números inteiros e reais, e a convenção para os valores true e false.
26. Projete um esquema de tradução de expressões aritméticas simples, sem parênteses, envolvendo variáveis simples e constantes aritméticas e as cinco operações aritméticas binárias usuais: soma, subtração, multiplicação, divisão, potenciação. Completar o projeto de expressões aritméticas ampliando as expressões da questão anterior para incorporarem operadores + e – unários, parênteses e chamadas de funções. **Defina as estruturas de dados e os algoritmos necessários**.
27. Projete um esquema de tradução para a comparação entre expressões aritméticas usando operadores de comparação >, <, =, ≠, ≥, ≤. Projete a compilação de expressões booleanas simples, envolvendo variáveis e constantes booleanas, e os operadores booleanos usuais and, or,

not. Ampliar as expressões booleanas incorporando as comparações entre expressões aritméticas, chamadas de funções booleanas e parênteses. Complete as expressões booleanas permitindo a comparação entre expressões booleanas através dos operadores de comparação = e ≠.

28. Defina detalhadamente o mapeamento de cada um dos comandos da linguagem de alto nível que você está implementando para a forma de código-objeto de nível mais baixo, a ser produzido cada vez que tais comandos forem encontrados no programa-fonte pelo compilador. Isso completa a especificação do código-objeto que deve ser gerado pelo compilador que está sendo construído.
29. Defina, para cada um dos comandos da linguagem de alto nível a ser compilada, as rotinas do ambiente de execução que forem necessárias para o correto funcionamento do código gerado em tempo de execução, completando dessa forma a especificação do conjunto das rotinas que compõem o ambiente de execução da linguagem implementada.
30. Verifique as especificações acima desenvolvidas em conjunto as especificações elaboradas nas aulas anteriores, testando sua consistência e compatibilidade mútua. Isso é importante para que nas próximas atividades as rotinas semânticas possam ser construídas e integradas com o reconhecedor sintático para formar o compilador desejado.

Integração das rotinas semânticas:

31. Insira as rotinas semânticas do analisador semântico no analisador sintático baseado em autômatos de pilha estruturados já desenvolvido para a linguagem de programação.
32. Identifique na gramática desenvolvida quais são e os pontos de inserção das rotinas semânticas. Insira rotinas para verificação de erros sintáticos dependentes de contexto (tipos de expressões, declaração de variáveis antes de seu uso, etc.).
33. Implemente e finalize o analisador semântico. Complementar o analisador sintático.
34. Usar o arquivo um texto de entrada contendo um programa escrito na linguagem de programação definida pela gramática da segunda parte do projeto como teste.

Lembre-se de que o relatório não deve conter programas escritos na linguagem que você usou como meio para implementar o compilador, deve conter apenas estruturas de dados (explicadas por meio de figuras e não de comandos em linguagem de alto nível), e algoritmos descritos em pseudo-código ou outra notação conveniente.

Toda definição, descrição, ou lista de objetos de caráter eminentemente técnico deve ser acompanhada de explicação em língua portuguesa.