Projeto 2019

Aula 08 - Aula de Projeto

Infraestrutura

Apresentação

- Esta é uma aula de projeto, em que se discutem alguns detalhes sobre o desenvolvimento do motor de eventos
- Este programa é uma infraestrutura a ser utilizada em outras fases do projeto, em que programas do sistema de programação serão desenvolvidos.
- Recomenda-se **começar a trabalhar o mais rápido possível,** para evitar dificuldades futuras devidas à superposição com outras disciplinas, provas etc.

- Criar um motor de eventos para simulações determinísticas de sistemas reativos definidos como conjunto de regras
- A primeira aplicação desse motor de eventos é a simulação de um subconjunto simples de um processador ou máquina virtual disponível
  - Qual hospedeiro usar (processador, máquina virtual)?
  - Qual é o subconjunto escolhido para simular?
- As entradas do simulador (eventos) correspondem às instruções dos programas cujo código se encontra armazenado na memória do processador simulado
- A ordem de ocorrência desses eventos pode variar de uma simulação para outra, conforme os dados processados pelo programa alvo da simulação, o conteúdo inicial dos seus elementos de armazenamento, e a primeira instrução executada

Especificações iniciais

1. Qual é o processador hospedeiro?
2. Especificar o hardware a simular
3. Qual é o tamanho da memória?
4. Listar as instruções a serem simuladas
5. Detalhar as instruções listadas em função das instruções da máquina hospedeira
6. Quais são os formatos das diversas instruções?

Sugestão para a documentação do projeto

Processador hospedeiro

- É a máquina (física, virtual ou abstrata) em que vai ser executado o simulador. Qual?_____
- Vantagens desta escolha? _____
- Desvantagens desta escolha?_____

Sugestão para a documentação do projeto

Levantamento de componentes

- Memória:
  - Número de bits da palavra _____
  - Número de bits de endereçamento _____
  - Número máximo de palavras _____
- Registradores:
  - Ponteiro de pilha: _____
  - Program counter: _____
  - Aritméticos/lógicos: _____
  - Outros: _____
- Equipamento de entrada/saída/armazenamento:
  - Entrada: _____
  - Saída: _____
  - Armazenamento: _____
- Blocos funcionais:
  - Unidade aritmética e lógica - Operações:_____
  - Unidade de controle - Instruções: _____
  - Sistema de Interrupções – Tipos: _____

Conjunto de instruções a simular

- Deseja-se especificar um conjunto simples (fácil de entender, usar, simular) de instruções a ser implementado contendo, por exemplo:
  - **Sequências** de comandos
  - Comandos de **Loop**
  - Comandos de **Decisão**
  - Comandos imperativos, compreendendo:
    - **Aritméticos/lógicos** (incluindo atribuições e chamadas de funções)
    - De **controle** (incluindo desvios e chamadas de procedimentos)
    - **Entrada/saída**
- São muito convenientes os conjuntos Turing-completos pois permitem codificar diretamente programas representados em diagramas de blocos planares, usados em programação imperativa estruturada (também chamados diagramas de Nassi-Schneidermann), pois atendem as condições do Teorema de Boehm-Jacopini

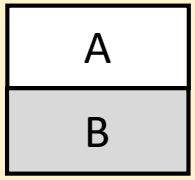
Para esclarecer

- **Máquina de Turing** = abstração que apresenta o máximo poder computacional entre os dispositivos estudados na teoria da computação
- **Turing-equivalente** = abstração que tem o mesmo poder computacional da Máquina de Turing
- **Turing-computável** = operação que pode ser processada por algum dispositivo Turing-equivalente
- **Turing-completa** = linguagem capaz de codificar qualquer operação Turing-computável

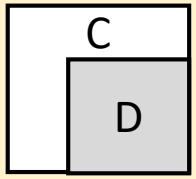
Diagramas de Nassi-Schneidermann

- São diagramas retangulares, formados exclusivamente pela justaposição de outros diagramas retangulares menores
- Os diagramas de Nassi-Schneidermann são compostos pela justaposição de blocos retangulares primitivos de três tipos:

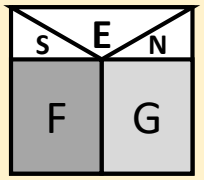
(1) sequência;



(2) bloco iterativo;



(3) bloco de decisão;



- Um retângulo vazio corresponde a uma operação vazia (nenhuma ação). Um retângulo com algum conteúdo interno A simboliza um comando imperativo que promove a execução de A ou da ação que A representa.
- A justaposição de um bloco A acima de um bloco B representa uma sequência formada pela execução da operação A, seguida pela da operação B. O retângulo resultante pode ser usado na composição de outros retângulos.
- Os blocos iterativos têm duas partes: C representa uma condição e D, o comando a executar repetidamente enquanto C for verdadeira.
- Os blocos de decisão têm três partes: E representa uma condição. F é o comando a executar se E for verdadeira, e G, o comando associado ao fato de E ser falsa.

Teorema de Boehm-Jacopini

- *Mediante transformações adequadas, com eventuais introduções de variáveis auxiliares e de duplicações de código,* ***qualquer lógica*** *representada em diagramas de blocos não estruturados* ***pode ser convertida em diagramas planares equivalentes*** *(tais como os da notação de Nassi-Schneidermann).*
- Isso representa uma potencial simplificação para a lógica dos programas, visto que diagramas planares são muito mais simples de conceber, ler e compreender, e portanto também mais simples de analisar, modificar e reparar
- Uma implicação direta desse teorema é que qualquer programa computacional pode ser reescrito na forma de diagramas estruturados.

Instruções a simular

- Listar e detalhar para uso no projeto:
  - Tipos de endereçamento de memória
  - Instruções de Referência à memória
  - Instruções de Referência a registradores
  - Instruções de Controle de fluxo
  - Instruções Aritméticas e lógicas
  - Instruções de Entrada e saída
  - Instruções de controle do estado da máquina

Formatos das Instruções a simular

- Para cada item, detalhar o formato das instruções (em nível dos bits do código de máquina):
  - Tipos de endereçamento de memória
  - Instruções de Referência à memória
  - Instruções de Referência a registradores
  - Instruções de Controle de fluxo
  - Instruções Aritméticas e lógicas
  - Instruções de Entrada e saída
  - Instruções de controle do estado da máquina

Funcionamento das Instruções a simular

- Para cada item, detalhar o funcionamento da instrução da máquina simulada em função das instruções da máquina hospedeira (com a intenção de criar as rotinas de tratamento associadas aos eventos correspondentes):
  - Tipos de endereçamento de memória
  - Instruções de Referência à memória
  - Instruções de Referência a registradores
  - Instruções de Controle de fluxo
  - Instruções Aritméticas e lógicas
  - Instruções de Entrada e saída
  - Instruções de controle do estado da máquina

Entradas do programa

- São os códigos dos programas a serem simulados, presentes na memória simulada.
- Como em um computador convencional, antes do início do processamento simulado:
  - O código do programa precisa ser carregado na memória simulada
  - Todas as áreas de dados de entrada do programa devem ser preenchidas com valores iniciais
  - Todos os elementos de hardware simulados, que representam áreas de armazenamento, devem estar preenchidos devidamente
  - É preciso indicar o endereço da primeira instrução para dar partida ao processo.

Obtenção dos eventos a simular

- Programas não são, nem é prático convertê-los em listas finitas de instruções
- O extrator de eventos pode extrair uma a uma as instruções a serem executadas, não sendo necessário pré-programá-las na lista de eventos.
- Portanto, a simulação de cada instrução:
  - Extrai uma instrução (a indicada pelo Program Counter)
  - Executa sua simulação
  - Relógio da simulação (soma tempo de processamento)
  - Simula outros eventos, presentes na lista de eventos
  - Aponta a próxima instrução (apontada pelo Program Counter)
  - Repete esse procedimento enquanto o programa está ativo

Alternativa de implementação

- Desejando-se uniformizar o tratamento dos eventos, mantendo a ideia de extrair uma instrução por vez para ser simulada, pode-se
  - fazer a busca da instrução, convertê-la em um evento com instante de ocorrência igual ao tempo corrente indicado pelo relógio da simulação, e inserindo-o na lista de eventos, com tipo “instrução a executar”
  - Inserir na lista outro evento para adiantar o relógio de simulação, para considerar o tempo de processamento da instrução que acaba de ser simulada
- Essa técnica costuma ser utilizada para simular o relógio de simulação: um evento “interrupção de relógio” é tratado, e programa a ocorrência do próximo evento do mesmo tipo.
- Economiza-se tempo de processamento programando apenas eventos de relógio que coincidam com os instantes para os quais foram programados outros eventos na lista.

Roteiro básico

- Montar um motor de eventos para interpretar programas contidos na memória simulada
- Criar o mecanismo básico de extração do próximo evento (ou seja, instrução) a simular
- Criar rotinas de tratamento dos eventos para simular as instruções da máquina simulada
- Escrever pequenos programas para efetuar testes
- Testar o simulador com esses programas.

Eventos artificiais auxiliares

- Para simplificar o desenvolvimento do trabalho sugere-se adicionar alguns eventos artificiais:
  - Execução de um loader absoluto (para preencher a memória simulada com o código a simular, a partir de um arquivo de tipo texto que contenha esse código)
  - Execução de um dumper absoluto (para mostrar na tela ou gravar em arquivo o conteúdo da memória simulada após a simulação do código)
  - Ligar trace (rastreamento, para imprimir passo a passo o conteúdo dos registradores da máquina simulada)
  - Desligar trace (inibe o rastreamento)
  - Início de simulação (indica valor inicial do program conter)
  - Fim de simulação (para impor momento do término da simulação e imprimir resultados finais)