UNIVERSIDADE ESTADUAL DE SANTA CRUZ - UESC

LUIZ AUGUSTO BELLO MARQUES DOS ANJOS

RELATORIO PARA TRABALHO PROJ1B PARA A DISCIPLINA CET087 – CONCEITOS DE LINGUAGEM DE PROGRAMAÇÃO

ILHÉUS – BAHIA

2024

SUMÁRIO:

1. INTRODUÇÃO:

Neste relatório, compartilho os detalhes da minha implementação de uma máquina de p-código em linguagem C. A máquina de p-código é uma construção fundamental em ciência da computação, e esta implementação foi inspirada nas ideias de Niklaus Wirth, conforme apresentadas em seu influente livro "Algorithms + Data Structures = Programs", publicado em 1976.

O objetivo principal deste projeto foi desenvolver um simulador da máquina de p-código, capaz de interpretar um conjunto limitado de instruções e executá-las de forma eficiente. Esta implementação envolveu a criação de um ambiente simulado que incluiu registradores específicos e uma pilha de dados, conforme especificado por Wirth.

Ao longo deste relatório, descrevo o processo de desenvolvimento, desde a concepção até a implementação final do código em C. Exploro em detalhes as instruções suportadas, e estrutura do código.

Em resumo, o código implementa um simulador de máquina de p-código em C, capaz de ler um conjunto de instruções, interpretá-las e executá-las, registrando em um arquivo de saída o estado interno da máquina em cada passo da execução. Isso permite uma análise detalhada do comportamento do programa durante a execução.

2.a INTERPRETADOR P-CODE EM C:

Este código implementa uma máquina de p-código em linguagem C, inspirada na máquina de p-código original proposta por Niklaus Wirth em Pascal. A máquina de p-código é uma representação simplificada de um ambiente de execução de programas, composta por um conjunto limitado de instruções e registradores.

Modelo de Entrada:

O programa lê as instruções que são declaradas em um vetor no próprio código, no formato “op arg1 arg2”, onde “op” é o código da operação, "argl" é um parâmetro para o nível lexical , e "arg2" é outro parâmetro que varia dependendo da operação, podendo ser um valor inteiro, um endereço do programa, a identidade de um operador etc.

Modelo de Saída:

A saída do programa é registrada no arquivo "resposta.txt". Cada linha deste arquivo representa o estado interno da máquina após a execução de uma instrução, exibindo o apontador de instrução "p", a instrução "i" a ser executada e, depois da execução da instrução "i", os valores dos apontadores "p", "b", "t" e as posições não vazias da pilha de execução

Funcionamento do Código:

1. **Definições e Estruturas de Dados:**
    i. O código define algumas constantes, como o tamanho máximo da pilha (stacksize), o tamanho máximo da tabela de código (cxmax), e os códigos de operação (fct).
    ii. Define também uma estrutura de dados “instrucao”, que representa uma instrução com seu código de operação (f), nível léxico (l), e argumento (a).
2. **Função parse_instrucao:**
    i. Esta função analisa uma linha do arquivo de entrada e extrai os componentes da instrução, atribuindo-os à estrutura "instrucao”.
3. **Função interpret:**
    i. Esta é a principal função do programa, responsável por interpretar e executar as instruções lidas do arquivo.
    ii. Utiliza um loop para percorrer todas as instruções lidas do arquivo.
    iii. A cada iteração, a função verifica o código de operação da instrução e executa a operação correspondente.
    iv. Mantém registros de estado (ponteiro de instrução p, base b, topo da pilha t) e manipula a pilha de dados de acordo com as instruções.
    v. A saída dessa função é direcionada para um arquivo de texto chamado "resposta.txt", registrando o estado interno da máquina após cada instrução.

4. **Função main:**
    i. Esta função coordena a leitura das instruções, chamando parse_instrucao para processá-las e armazená-las em um array de instrucao.
    ii. Em seguida, chama a função interpret para executar as instruções.
    iii. Por fim, fecha os arquivos abertos e encerra o programa.

2.b CÓDIGO DO INTERPRETADOR EM C:

2.c Código para compilação e execução no terminal:

- gcc -o Interpretador_Questao1 Interpretador_Questao1.c
- ./ Interpretador_Questao1.
- gcc -o Interpretador_Questao2 Interpretador_Questao2.c
- ./ Interpretador_Questao2.c
- gcc -o Interpretador_Questao3 Interpretador_Questao3.c
- ./ Interpretador_Questao3.c

3. QUESTÕES RESOLVIDAS:

3.1 Soma dos números naturais de 1 até 10:

lit 0 0
lit 0 10
sto 0 0
lit 0 1
sto 0 1
lit 0 0
sto 0 2
lod 0 1

lod 0 0

opr 0 2

sto 0 0

lit 0 1

lod 0 1

opr 0 2

sto 0 1

lit 0 10

lod 0 1

opr 0 11

jpc 0 23

jmp 0 8

opr 0 0

Algoritmo Questão 1:

O algoritmo funciona calculando a soma dos números naturais de 1 a 10 usando um loop simples de iteração e um acumulador para armazenar o resultado. Ele segue uma abordagem sistemática, atualizando a variável de iteração a cada passo e verificando se o limite de iteração foi alcançado para determinar quando parar o cálculo.

Inicialização: O algoritmo começa definindo algumas variáveis necessárias para a execução do cálculo. Isso inclui um acumulador para armazenar a soma dos números naturais, uma variável para iterar sobre os números de 1 a 10 e uma variável temporária para armazenar valores intermediários, se necessário.

Iteração: O algoritmo usa um loop para iterar sobre os números de 1 a 10. Durante cada iteração, ele adiciona o número atual ao acumulador para calcular a soma total.

Atualização da variável de iteração: Após cada iteração, a variável de iteração é incrementada para prosseguir para o próximo número na sequência.

Condição de parada: O algoritmo verifica se a variável de iteração atingiu o valor de 10. Se sim, ele concluiu a soma e pode encerrar o loop. Caso contrário, ele continua o loop para a próxima iteração.

3.2 Soma dos quadrados dos números naturais de 1 até 100.

lit 0 0

lit 0 100

sto 0 0

lit 0 1

sto 0 1

lit 0 0

sto 0 2

lod 0 1

lod 0 1

opr 0 4

lod 0 0

opr 0 2

sto 0 0

lit 0 1

lod 0 1

opr 0 2

sto 0 1

lit 0 100

lod 0 1

opr 0 11

jpc 0 24

jmp 0 8

opr 0 0

Algoritmo Questão 2:

O algoritmo calcula a soma do quadrado dos números naturais de 1 a 100, usando um loop de iteração, um acumulador para armazenar o resultado e operações de multiplicação e adição para calcular os quadrados e somá-los ao acumulador.

Inicialização: Inicialmente, o acumulador é configurado com o valor zero para armazenar a soma dos quadrados dos números naturais. Além disso, uma variável é inicializada com o valor 100 para representar o limite superior da sequência.

Configuração da variável de iteração: Uma variável de iteração é inicializada com o valor 1, pois começaremos a calcular a soma dos quadrados a partir do número 1.

Configuração da variável temporária: Uma variável temporária é inicializada com o valor zero. Esta variável temporária será usada para armazenar temporariamente os quadrados dos números.

Iteração sobre os números: O algoritmo começa um loop para iterar sobre os números de 1 a 100. Durante cada iteração:

- O número atual (variável de iteração) é duplicado (multiplicado por si mesmo) usando a operação opr 0 4, que representa a multiplicação.
- O quadrado do número atual é adicionado ao acumulador, usando a operação opr 0 2, que representa a adição.

Atualização da variável de iteração: Após cada iteração, a variável de iteração é incrementada em 1, movendo-se para o próximo número na sequência.

Verificação da condição de parada: O algoritmo verifica se a variável de iteração atingiu o valor 100, que é o limite superior da sequência. Se sim, o loop é encerrado e o cálculo é concluído.

3.3 Soma dos cubos dos números naturais de 1 até 1000.

lit 0 0

lit 0 1000

sto 0 0

lit 0 1

sto 0 1

lit 0 0

sto 0 2

lod 0 1

lod 0 1

lod 0 1

opr 0 4

opr 0 4

opr 0 4

opr 0 2

lod 0 0

opr 0 2

sto 0 0

lit 0 1

lod 0 1

opr 0 2

sto 0 1

lit 0 1000

lod 0 1

opr 0 11

jpc 0 24

jmp 0 8

opr 0 0

Algoritmo Questão 3:

O algoritmo calcula a soma do cubo dos números naturais de 1 a 1000, usando um loop de iteração, um acumulador para armazenar o resultado e operações de multiplicação e adição para calcular os cubos e somá-los ao acumulador.

Inicialização: Inicialmente, o acumulador é configurado com o valor zero para armazenar a soma dos cubos dos números naturais. Uma variável é inicializada com o valor 1000 para representar o limite superior da sequência.

Configuração da variável de iteração: Uma variável de iteração é inicializada com o valor 1, pois começaremos a calcular a soma dos cubos a partir do número 1.

Configuração da variável temporária: Uma variável temporária é inicializada com o valor zero. Esta variável temporária será usada para armazenar temporariamente os cubos dos números.

Iteração sobre os números: O algoritmo começa um loop para iterar sobre os números de 1 a 1000. Durante cada iteração:

- O número atual (variável de iteração) é triplicado (elevado ao cubo) três vezes consecutivas usando a operação opr 0 4, que representa a multiplicação.
- O cubo do número atual é adicionado ao acumulador, usando a operação opr 0 2, que representa a adição.

Atualização da variável de iteração: Após cada iteração, a variável de iteração é incrementada em 1, movendo-se para o próximo número na sequência.

Verificação da condição de parada: O algoritmo verifica se a variável de iteração atingiu o valor 1000, que é o limite superior da sequência. Se sim, o loop é encerrado e o cálculo é concluído.

4. LINKS PARA DOWNLOAD

- link para o GitHub: https://github.com/UESC/tree/main/Proj1b

5. REFERÊNCIAS:

- https://en.wikipedia.org/wiki/P-code_machine
- https://prograd.uesc.br/MaterialApoio/Aula/VirtualMachine.pdf