Relatório Exercício Programa I - Batalha de Robôs

Fellipe Souto Sampaio *
Gervásio Protásio dos Santos Neto †
Vinícius Jorge Vendramini ‡

16 de setembro de 2013

MAC 0242 Laboratório de Programação II
Prof. Marco Dimas Gubitoso

Instituto de Matemática e Estatística - IME USP
Rua do Matão 1010
05311-970 Cidade Universitária, São Paulo - SP

*Número USP: 7990422 e-mail: fellipe.sampaio@usp.com
†Número USP: 7990996 e-mail: gervasio.neto@usp.br
‡Número USP: 7991103 e-mail: vinicius.vendramini@usp.br

1 Introdução

Este relatório pretende explicar a implementação do exercício programa I, detalhando a implementação de seu funcionamento, e do projeto de suas classes e objetos.

Detalharemos a forma de implementação e manuseio da pilha e sua relação com o módulo operação, e o uso das expressões regulares para o processamento da entrada de dados.

2 Expressões Regulares

Para processamento do arquivo-fonte contendo o prorama a ser interpretado foram utilizadas expressões regulares do Perl.

Usou-se o operador diamante (<>) dentro de um loop while para ler a entrada até o final e a cada linha da entrada viu-se se ela se conformava com o esperado de uma linha de código.

Mais especificamente, a seguinte expressões regulares aparecem em nossa implementação do processador de texto:

```
/^#.*[\n\f]*/ #verifica se se trata de um comentario

/(\b[a-zA-Z]*\b:\s*)[\n\f#]*$/ #identifica uma linha
                               # so com label

/(\b[a-zA-Z]*\b:\s*)?(\b[a-zA-Z]{2,4}\b[^:]?)
(\w*)\s*[\n\f#]*/ #identifica uma linha executavel
```

Durante o processamento do textos, dependendo do que foi capturado pela expressão regular, realiza-se um comando:

- Comentário: Linhas de comentário e comentários em linhas válidas são ignorados.
- Label: caso a label seja uma string válida (não é um undef), ela é inserida em um hash na stack (uma instância do módulo stack, detalhado mais a frente). A label é usada como key do hash, o valor guardado é a posição do programa marcada pelo label.
- Comandos: Ao identificar um par ordenado (comando,argumento) é criada uma referência anônima para esse par que é então inserido no vetor de instruções.

3 Vetor de Instruções

O vetor de instruções é uma estrura de dados que contém a sequência dos comandos que devem ser executados, acompanhados de seus respectivos argumentos.

Sempre que é encontrada uma instrução válida, ela e seu argumentos são inseridos em um vetor anônimo, e este então é inserido no vetor de instruções.

Durante a execução do programa, o vetor é percorrido e seu conteúdo (os comandos) são interpretardos por funções do módulo stack e do módulo operations.

Com execessão de quando é executado um pulo (JMP), a leitura do vetor é linear, como a sua posição atual (o Program Counter) sendo um atributo da pilha.

4 Pilha de Dados

Na implementação de uma máquina virtual o elemento principal é a pilha de dados. Em nosso programa foi criado a classe pilha, um objeto munido de operações usuáis, como empilhar, desempilhar, devolver o topo, duplicar entre outros. Todas estas operações são aplicadas diretamente na pilha, independente do tipo de dado empilhado.

O método principal de operação da pilha é chamado *makeOperation*, no qual a instrução a ser executada é recebida, testando se a instrução está presente no hashing de instruções válidas, em caso positivo o conteudo é uma referência para qual médoto deve ser aplicado (saltos, empilhamento, comparação lógica do conteúdo, descarte, impressão ...). No segundo caso, negativo, é verificado se a instrução é chave do hashing do objeto operation.

5 Operações Lógicas e Aritméticas

Neste módulo está a classe operation, responsável pelas operações lógicas e aritméticas entre elementos da pilha. Caso a operação requisitada exista esta é aplicada aos elementos que foram desempilhados da pilha. O valor final da operação binária é devolvido para que seja empilhado. Semelhante a implementação da classe stack,o hashing das operações tem como conteúdo referências para as funções que devem ser aplicadas.

6 Integração

Na execução do projeto procurou-se ao máximo atender a um projeto de orientação a obejto.

Os módulos stack e operations regimentam a execução. Em stack, há uma referência para um objeto do tipo operations.

É no móduo pilha que são chamadas (por meio de referências anônimas) as funcções responsáveis pela execução de operações. Primeiro checa-se se a operação a ser realizada é uma operação de pilha; se for, é realizada. Cado contrário, invoca-se a funcção pertinente do operations por meio do objeto previamente instanciando.

Por fim o readSource executa um loop (um ciclo Fetch, Decode, Execute - FDX), chamando as funções da pilha que processam comandos por meio de um objeto to tipo stack. O loop roda até chegar-se ao fim do vetor de instruções ou encontrar-se o comando END.

Anexo 1 - Códigos Testados

Sequencia de Fibonacci

```
PUSH 1
PUSH 0
STO 0
STO 1
PUSH 20
STO 2
LOOP:
RCL 0
RCL 1
DUP
STO 0
ADD
DUP
STO 1
PRN
RCL 2
PUSH 1
SUB
DUP
STO 2
PUSH 0
EQ
JIF LOOP
END
```

Potências de 2

```
# inicializa
PUSH 1
STO 0
PUSH 10000000
STO 1

LOOP: POP
RCL 0
DUP
ADD
STO 0
RCL 0
PRN
RCL 0
RCL 1
LT
JIF       LOOP

END
```

A resposta para a pergunta fundamental da vida, universo e tudo mais

```
PUSH 10
PUSH 4
ADD
PUSH 3
MUL
PRN
END
```