UNIVERSIDADE DE ÉVORA

Ano letivo 2017/2018
Linguagens de Programação

Relatório do 2º trabalho prático:

Docentes: Teresa Gonçalves , Nuno Miranda

Discentes: Henrique Raposo nº 33101 e José Medeiros nº 31174

Introdução

O objectivo desta primeira parte do trabalho é implementar uma máquina abstrata CIMS (Conjunto de Instruções Máquina Simples)

Descrição

Neste trabalho são utilizados o Jlex e o CUP do Java fornecidos pelo docente nas aulas práticas , deste ficheiros foi alterado o CIMS.cup principalmente, de forma a funcionar com o CIMS.lex e introduzir instruções na maquina.

As instruções são objectos com hierarquia , sendo a classe Instrução a classe principal , os diferentes tipos de instruções extendidos por esta , e as diferentes instruções idem aspas.

Estas Instruções são guardadas na ArrayList mem que contém todas as instruções do programa lido , por ordem de leitura.

Os Objectos Etiquetas estão guardados numa Hashtable dentro da class CIMS.java onde o nome da função é a chave, e as etiquetas o valor.

Descrição da máquina:

Hashtable<String,Etiquetas> etiquetas - hash table com objetos etiquetas , esta hash table , tem como chave o nome da etiqueta (função da programa) e o objecto etiqueta contém a posição onde esta se encontra.

ArrayList<Instrucao> mem – array list utilizada para guardar todas as instruções lidas , estas estão organizadas por ordem de leitura para a facil a utilização

Stack<Integer> pilhaAval – pilha de inteiros , utilizada pra fazer os calculos da máquina , e operações.

int pc; - program counter , indica a instrução onde a maquina se encontra.

Stack<RegistoAtivacao> blocos – stack que contém os registos de ativação de modo a organizar os blocos do programa.

ArrayList<RegistoAtivacao> memexec; //a memória de execucao, onde se encontram os registos de ativaçao dos blocos do programa cuja execucao ainda nao terminou.

RegistoAtivacao AA; apontador ambiente , aponta para o registo de ativaçao corrente

Stack<Integer> args; - stack utilizada para colocar os valores da função coloca_args de modo a serem utilizados na próxima função

int flag; - variável utilizada para continuar ou parar o ciclo de execução do programa (caso a stack de blocos fique vazia na intrução regressa, esta variável passa para zero terminando assim o ciclo de execução.

int pcTemp; - variável temporaria apenas para guardar o program counter em algumas operações.

Para além de um ciclo de execução e do contrutor este objeto também possui um método para encontrar a main , de modo a que esta seja a primeira função a ser executada.

Descrição do registo de ativação:

RegistoAtivacao ControlLink; - Apontador para a função que chamou e criou este registo.

int EnderecoRetorno; - program counter da posição onde é suposto o programa continuar após a conclusão desta função.

int[] vars; - array de variaveis deste registo de ativação , capacidade dependente do numero passado pelo construtor.

int[] args; - semelhante ao anterior mas relativamente aos argumentos.

int nargs; - número de argumentos que a função vai receber.

int nvars; -número de variaveis que a função vai receber.

Instruções

Soma

soma:

valor2 = desempilha()

valor1 = desempilha()

empilha(valor2 – valor1)

pc = pc + 1

Sub

sub:

valor2 = desempilha()

valor1 = desempilha()

empilha(valor2 – valor1)

pc = pc + 1

Div

div:

valor2 = desempilha()

valor1 = desempilha()

empilha(valor2 / valor1)

pc = pc + 1

Mod

mod:

valor2 = desempilha()

valor1 = desempilha()

empilha(valor2 % valor1)

pc = pc + 1

Mult

mult:

valor2 = desempilha()

valor1 = desempilha()

empilha(valor2 * valor1)

pc = pc + 1

Salta

salta(string etiqueta):

pc = etiquetas.get(etiqueta).posicao

pc = pc + 1

Sigual

sigual(string etiqueta):

valor2 = desempilha()

valor1 = desempilha()

```
        if ( valor2 == valor1)
                pc = etiquetas.get(etiqueta).posicao
                pc = pc + 1
        else if ( valor1 != valor2)
                pc = pc + 1
```

Smenor

```
sigual(string etiqueta):
        valor2 = desempilha()
        valor1 = desempilha()
        if ( valor1 > valor2)
                pc = etiquetas.get(etiqueta).posicao
                pc = pc + 1
        else
                pc = pc + 1
```

Empilha

```
empilha(int a):
        empilha(a)
        pc = pc + 1
```

Empilha_var

```
empilha_var(int a, int b):
        RA temp = AA
        while(a>0)
                temp = temp.ControlLink
        empilha(temp.vars[b-1])
        pc = pc + 1
```

Empilha_arg

```
empilha_arg(int a, int b):
        RA temp = AA
```

```
        while(a>0)
            if(temp.ControlLink != NULL)
                temp = temp.ControlLink
                a = a - 1
        empilha(temp.args[b-1])
        pc = pc + 1
```

Atribui_var

```
atribui_var(int a, int b):
        RA temp = AA
        while(a>0)
            temp = temp.ControlLink
        temp.vars[b-1] = desempilha()
        pc = pc + 1
```

Atribui_arg

```
atribui_arg(int a, int b):
        RA temp = AA
        while(a>0)
            temp = temp.ControlLink
        temp.args[b-1] = desempilha()
        pc = pc + 1
```

Coloca_arg

```
coloca_arg(int a):
        empilha(desempilha())
        pc = pc + 1
```

Chama

```
chama(int a, string etiqueta):
        pc = pc + 1
```

```
        pcTemp = pc + 1
        proxEtiqueta = etiquetas.get(etiqueta)
        pc = proxEtiqueta.posicao
        pc = pc + 1
```

Locais

```
locais(int arg, int var):
        RA reg = new RA(arg, var, pcTemp, AA)
        blocos.empilha(reg)
        AA = reg
        int j = arg
        int i = 0
        while(j>0)
                AA.args[i] = desempilha()
                i = i + 1
                j = j – 1
        pc = pc + 1
```

Regressa

```
regressa:
```

```
        blocos.desempilha()
        if(blocos.empty() == false)
                pc = AA.EnderecoRetorno
                AA = AA.ControlLink
        else
                flag = 0
```

Escreve_int

```
escreve_int:
        int valor = pilhaAval.desempilha()
        print(“ “ + valor + “ “)
        pc = pc + 1
```

Escreve_str

```
escreve_ str(string str):
        print(“ “ + str + “ “)
        pc = pc + 1
```

Muda_linha

```
muda_linha:
        print()
        pc = pc + 1
```