Segundo EP
Máquina Virtual

3 de novembro de 2009

Neste exercício programa você vai implementar um processador virtual simples, capaz de executar programas em "linguagem de máquina".

Um programa deste processador consiste em uma sequência de números (*bytes*) e uma coleção de 256 registradores capazes de armazenar um valor *double* cada um.

Além disso, o processador possui dois registradores especiais: o ponteiro de instruções (**PI**) e um armazenador de estado (**E**), discutidos abaixo.

Os registradores são representados simplesmente por um vetor de *doubles*:

```
double reg[256] = {0.,};
```

e são referenciados por seu índice no vetor, veja os exemplos na seção 2.

De modo similar, as instruções ficam em um outro vetor de *bytes*:

```
signed char prog[MAXMEM] = {0,};
```

`MAXMEM` é uma constante com o tamanho máximo do programa, use pelo menos 1024.

O registrador **PI** indica a próxima instrução a ser executada. Cada instrução corresponde a um código de operação (somar, ler, imprimir, desvio, etc) e pode conter alguns operandos.

A entrada é a lista das instruções, terminadas pelo código especial 1000, seguida por uma segunda lista, de pares de valores:

1. Um inteiro indicando o índice da variável
2. Um *double* indicando o valor correspondente no início do programa

A segunda lista termina com o valor -1.

1 Instruções

Os códigos de instruções estão listados a seguir, separados por classes.

Aritméticas As instruções aritmeticas começam em 10 e possuem 3 argumentos: os índices dos dois operandos e onde será guardado o resultado.

Estes são os códigos:

10 soma
11 subtração
12 multiplicação
13 divisão

Por exemplo, a sequência:

12 5 1 42

significa que o conteúdo do registrador 5 será multiplicado pelo conteúdo do registrador 1 e o resultado será armazenado no registrador 42. Em **C** fica assim:

```
reg[42] = reg[5] * reg[1];
```

Fácil, não?

Lógicos As operações lógicas (comparação, negação, etc, seguem uma forma similar, mas o resultado é armazenado no resgistrador especial **E**. O valor 0 (zero) corresponde a falso, qualquer outro significa verdadeiro, como em **C**.

Os códigos começam em 20:

20 maior
21 menor
22 maior ou igual
23 menor ou igual
24 igual
25 diferente
26 negação (apenas um operando)
27 compara com 0 (apenas um operando)

Movimentação Começam em 30 e possuem os seguintes significados:

30 *orig dest*	copia o valor da *orig* para *dest*
31 r_1 r_2	troca o valor de r_1 com o de r_2
32 *reg*	copia o valor de **E** em *reg*
33 *reg*	copia o valor de *reg* em **E**

Desvios Os desvios são especiais, eles alteram especificamente o valor do registrador **PI**, somando o argumento. São usados para *if* e *laços*.

Começam em 40:

40 *delta*	soma *delta* ao valor de **PI** (`PI += prog[PI];`)
41 *delta*	soma *delta* ao valor de **PI**, se $\mathbf{E} \neq 0$
42 *delta*	soma *delta* ao valor de **PI**, se $\mathbf{E} = 0$

Miscelânea Em 50 começam algumas instruções especiais:

50 *dest*	imprime na saída o conteúdo da variável em *dest*
51 *dest*	lê um valor[1] da entrada e coloca na variável *dest*
52 r	incrementa a variável em r
53 r	decrementa a variável em r
-1	término do programa

Matemáticos A partir da posição 100 ficam as funções matemáticas:

100 *orig dest*	*dest* recebe a raiz quadrada de *orig*
101 *orig dest*	*dest* recebe seno de *orig*
102	idem, cosseno
103	tangente
etc	

2 Exemplo

Um exemplo de entrada. Siga os códigos e tente entender.

2.1 Fibonacci

Lê um número n e imprime os n primeiros elementos da série de Fibonacci, a partir do terceiro. Os dois primeiros estão inicialmente colocados nas variáveis 1 e 2.

```
51 0 27 0 42 18 10 2 3 1 30 2 3 30 1 2 50 1 53 0 40 -18 -1 1000
1 1
2 1
-1
```

Explicando melhor:

0	51 0	lê n em $reg[0]$
2	27 0	$n = 0$?
4	42 18	se for, vai para 22
6	10 2 3 1	$reg[1] = reg[2] + reg[3]$
10	30 2 3	$reg[3] = reg[2]$
13	30 1 2	$reg[2] = reg[1]$
16	50 1	imprime $reg[1]$
18	53 0	$reg[0] - -$
20	40 -18	vai para 2
22	-1	fim do programa

3 Bônus especial

Ponto adicional para quem fizer a impressão do programa em código de máquina formatada, com uma instrução por linha e trocando o código da instrução por um nome adequado. Por exemplo, o programa acima ficaria:

```
 0: Leia      0
 2: Zero?     0
 4: Sim->    22
 6: Soma      2   3 ->  1
10: Copia     2   3
13: Copia     1   2
16: Imprime   1
18: Dec       0
20: Desvia    2
22: Fim
```

Boa sorte!!!