

Processamento da Informação

Fabício Olivetti de França

18 de Fevereiro de 2020

Atividade 01: Computador IntCode

Computador IntCode

Nessa atividade trabalharemos com um computador chamado IntCode. Esse computador é composto por uma memória contendo valores inteiros que ou representam uma determinada operação ou um dado a ser utilizado para o processamento. Além da memória, o computador também possui uma variável `pos` que indica a posição do comando atual a ser executado, e uma variável `status` que indica se ainda existem operações a serem realizadas (`IDLE`), se está aguardando alguma ação do usuário (`WAIT`) ou se o programa terminou (`HALT`).

Para criar uma nova máquina IntCode, dados os valores de `memoria`, `pos`, `status` fazemos:

```
m = Maquina(memoria, pos, status)
```

Também podemos criar uma máquina a partir de uma lista de inteiros representando a memória como:

```
m = criaMaquina(lista)
```

essa máquina terá como valores padrões `pos = 0` e `status = IDLE`.

Uma máquina IntCode `m` permite o acesso aos seus três campos através da sintaxe:

```
m.memoria[i] # acessa a i-ésima posição da memória
m.pos        # a posição que deve ser lida nesse instante
m.status     # status atual da máquina
```

Essa sintaxe permite a leitura mas não a escrita de novos valores para essas variáveis. Portanto, para alterarmos o estado atual de uma máquina devemos primeiro gravar os valores desses campos em variáveis e, depois, criar uma nova máquina:

```

memoria, pos, status = m

# altera o valor pos+3 da memória para 2 e avança pos em 4 posições
memoria[pos + 3] = 2
pos                = pos + 4

m = Maquina(memoria, pos, status) # nova máquina, novo estado

```

IntCode que apenas sabe terminar

Atualmente nosso computador IntCode possui apenas uma instrução representada pelo valor 99, os valores que representam uma instrução são chamados de *opcodes*. O *opcode* 99 indica que o programa encerrou e nada mais deve ser feito, isso altera o status do IntCode para **HALT**:

```

def terminar(m : Maquina) -> Maquina:
    memoria, pos, status = m
    return Maquina(memoria, pos, HALT)

```

A execução da máquina é feita lendo cada valor da memória, passo a passo, e executando a instrução correspondente. Para nossa máquina atual o único programa possível é [99] que simplesmente lê o valor 99 e termina.

Exercício

Vamos implementar outras duas operações: *opcodes* 1 (adição) e 2 (multiplicação). Para isso complete as funções `somar` e `multiplicar` no código *IntCode.py* de tal forma que:

- Para somar, leia os valores nas posições `pos+1`, `pos+2`, `pos+3` da memória e armazene nas variáveis `x`, `y`, `z`.
- Na posição de memória `z`, armazene a soma dos valores nas posições `x` e `y` da memória. Avance `pos` em 4 posições, pois já consumimos quatro valores do nosso programa (*opcode* e 3 variáveis).
- Retorne a nova máquina.

Para multiplicar o procedimento é análogo, apenas trocando a operação matemática.

Considere o programa exemplo [1,0,0,0,99]. A máquina começa sua leitura em `pos=0` e executa os seguintes passos:

```

[1,0,0,0,99], pos=0, status=IDLE

memoria[pos] == opcode(1) == somar

x = 0

```

```

y = 0
z = 0

memoria[0] = memoria[0] + memoria[0] = 1 + 1 = 2
pos = 4

# Nova máquina:
[2,0,0,0,99] , pos=4, status=IDLE

memoria[pos] == opcode(99) == terminar

[2,0,0,0,99], pos=4, status=HALT
Um outro exemplo para reforçar:
[2,4,4,5,99,0], pos=0, status=IDLE

memoria[pos] == opcode(2) == multiplicar

x = 4
y = 4
z = 5

memoria[5] = memoria[4] * memoria[4] = 99*99 = 9801

[2,4,4,5,99,9801], pos=4, status=IDLE

memoria[pos] == opcode(99) == terminar

[2,4,4,5,99,9801], pos=4, status=HALT

```