Atividade 6: Assembler/parte 2

Módulo *SymbolTable*

[**Feito**] Funções a serem implementadas pelo aluno (pg. 115, livro Noam/Shimon):

- Constructor
- addEntry
- contains
- getAddress

Aplicação *Assembler*

Q1 Usando o Assembler criado em aula, use-o para gerar as instruções de máquina do seguinte código:

```
// Teste assembler.py vs 2
// Escrever na memória 0 o número 31 e na memória 1 o número 11
    @R0
    D = M      // carrega em D o conteúdo de M['R0']
    @R1
    D = D + M  // soma D com M['R1'] e carrega em D
    @ sum
    M = D      // escreve em M['sum'] o valor da soma
// verifica se a soma é 42
    @ 42
    D = A  // carrega D - 42 em D
    @sum
    D = D-M
    @ INCORRETO
    D; JNE     // se D não for 0, pula para #14
(CORRETO)
    @ verifica
    M = 1     // escreve em M['verifica'] o valor 1
    @ END
    0; JMP    // pula para final6
(INCORRETO)
```

```
    @ verifica
    M = -1    // escreve em M['verifica'] o valor -1
// fim do programa
(END)
    @ END
    0; JMP    // laço infinito
```

Esse código está disponível no material de download como o arquivo **teste2.asm**. Nossa aplicação assembler deverá gerar o arquivo **teste2.hack**. A ser carregado no emulador **CPUEmulator.*{sh|bat}***.

Ao testar as instruções de máquina, explique como finaliza a memória quando incializamos a mesma com:

- **a)** RAM_0 = 11; RAM_1 = 13
- **b)** RAM_0 = 31; RAM_1 = 11

Q2 Explique em suas palavras o funcionamento do módulo *SymbolTable* e a necessidade de percorrermos as instruções em assembly por duas vezes na aplicação *Assembler*.

Uma versão da aplicação **`assembler.py`** está disponível. Para rodar no Linux:

`python3 -m assembler.py`

Ou

python3 -m assembler.py teste1

Ou

(única vez)
`chmod +x assembler.py`
(sempre)
./assembler.py ou ./assembler.py teste1

Referências para o trabalho

O site http://www.nand2tetris.org contém o simulador de CPU a ser usado no trabalho. Encontra-se em https://www.nand2tetris.org/software no link indicado aqui:

Download the Nand2tetris Software Suite Version 2.6 (about 730K).

Usaremos o simulador **`CPUEmulator.sh`** (linux/mac) ou o **`CPUEmulator.bat`** (windows) para rodar os códigos de máquina.

Na dúvida, o livro ***The Elements of Computing Systems: Building a Modern Computer from First Principles*** de Noam Nisan e Shimon Schocken será nosso guia para essa atividade, em especial o capítulo 6 (*Assembler*). Os padrões de implementação são os que eles propõem para chegarmos na programação do *Hack computer system*.

Entrega

Os alunos deverão dar upload na área de *assignments* do Teams de seus arquivos ".hack" gerados assim como o ".pdf" contendo a descrição dos códigos, compactados em um único arquivo (.zip, .7z, .tar.xz, …), até a **data definida no *assignment***.