UNIVERSIDADE TUIUTI DO PARANÁ

FACULDADE: FACET - Faculdade de Ciências Exatas e de Tecnologia		
CURSO: Superior de Tecnologia em Análise e Desenvolvimento de Sistemas		
ANO LETIVO: 2023-2	REGIME: Semestral	PERÍODO: 1°/2º
COMPONENTE CURRICULAR: Arquitetura e Organização de Computadores		
PROFESSOR: Igor Pereira dos Santos		

Manual de instruções

Simulador Assembly – Comandos básicos

Endereço do simulador: https://schweigi.github.io/assembler-simulator/
Documentação: https://schweigi.github.io/assembler-simulator/instruction-set.html

1. Comentários podem ser escritos utilizando o **;** , seguido do comentário.
2. Para declarar variáveis deve ser utilizado o comando **DB**, o nome da variável é seguido de :, conforme o exemplo abaixo que cria uma variável e atribui um hello world como string.

   hello: **DB** "Hello World!" ; Variable

   DB 0 ; String terminator

   num1: **DB** 10 ; Variable

   Variavel precisa ser declarada após o jumper do start

```
        JMP start
num1: DB 10 ; Variable
num2: DB 10 ; Variable
```

3. Todo código deve possuir a label de inicio e o JMP para esta função de inicialização:

```
        JMP start
start:
```

4. A operação MOV destino, origem realiza a cópia/transferência dos dados de uma constante, endereço ou de outro registrador para o registro de destino.

```
start:
        MOV C, hello    ; Point to var
```

5. A operação CALL é uma chamada de uma função dentro do simulador, no caso em questão da função print nesta linha do código ela é chamada e executada.

```
CALL print
```

6. A operação PUSH envia o valor para a pilha do registrador SP e ela pode ser incrementada e decrementada para que aponte para a posição atual da memória.

```
PUSH A
PUSH B
```

Utilizações possíveis

```
PUSH reg
PUSH address
PUSH constant
```

7. A operação ADD destino, origem realiza uma operação de soma entre os valores e armazena no registrador de origem. A operação SUB funciona do mesmo formato, mas realiza a operação de subtração.

```
ADD A, B
```

Utilizações possíveis

```
ADD reg, reg
ADD reg, address
ADD reg, constant
SUB reg, reg
SUB reg, address
SUB reg, constant
```

8. A operação INC reg, DEC reg incrementa ou decrementa o registrador por um, neste caso o incremento e decremento faz apontar para o próximo ou anterior endereço de memória.

```
INC C
INC D
```

9. A operação CMP compara dois valores e retorna para a flag true se forem iguais.

```
CMP B, [C]
```

10. A operação JNZ é um jump condicional, neste caso ele vai pular para a chamada da função se o valor não for zero.

```
.loop:
        MOV A, [C]
        MOV [D], A
        INC C
        INC D

        CMP B, [C]

        JNZ .loop
```

11. A operação POP remove da pilha o valor do registrador.

```
POP B
POP A
```

12. A operação HLT para a execução do processador.
13. A operação RET sai da função de sub-rotina.