

Universidade de São Paulo

**Escola de Artes, Ciências e
Humanidades**

**Disciplina: ACH2034 - Organização de
Computadores Digitais**

Exercício Programa 2

Louise Karen de Moura Martins - 8921284

Vitor Augusto de Souza Corrêa - 7611501

1. Introdução

Para executar o EP, a assinatura é: java EP2OCD <numero de bytes da memória>.

Pesquisamos algumas referências de implementações e decidimos optar para que cada instrução tenha: 1 byte do operador e 1 byte para cada operando. Portanto, a instrução MOV A, [100] teria o equivalente à três bytes. Na mesma linha de raciocínio, nossa flag Overflow irá alertar quando um valor estiver fora do intervalo -127 à 128 para valores (supomos que instruções sempre caberão).

Temos duas classes: a classe EP2OCD que controla o fluxo do programa e interface visual e a classe CPU que tem implementado as instruções e flags.

Uma instância da CPU possui como atributos as flags, os registradores e a memória.

Usamos um array de Strings chamado de memory (o atributo de “memória” citado acima), que tem o tamanho passado pelo argumento na chamada do EP. Nesse array são armazenadas as instruções (as flags e outros tem a sua própria “memória”) e é onde podemos manipular a memória diretamente.

Temos dois modos de execução do EP:

- O modos de “Ler arquivo” permite que o usuário escreva todo seu algoritmo em um .txt e o EP irá consumí-lo passo a passo;
- O modos “Freestyle” permite que o usuário escreva uma operação por vez.

Ambos os modos de execução escrevem, à cada instrução, a situação dos registradores e das flags.

2. Sintaxe

A sintaxe das instruções do EP é:

<Operação> <operador1>, <operador2>

(A operação é separada do operador1 por um espaço em branco. O operador1 é separado do operador2 por uma vírgula e um espaço em branco).

Nossos operadores tem três tipos:

1. Registradores: É possível utilizar os registradores A, B, C e D da seguinte forma: MOV A, B;
2. Endereço da memória: Utilizamos a seguinte notação para o endereço de memória: MOV [100], [30] – onde esses valores estão entre 0 e o valor inserido na chamada do EP. A notação [B] indica o endereço de memória do registrador B;
3. Constantes: Basta colocar o número para usarmos constantes: MOV A, 2.

3. Operações

- MOV <destino, origem> – Executa destino = origem

Essa operação aceita:

MOV reg, reg

MOV reg, endereço

MOV reg, constante

MOV endereço, reg

MOV endereço, constante

- CMP <a, b> – Executa ZF = (a == b)

Essa operação aceita:

CMP reg, reg

CMP reg, endereço

CMP reg, constante

Operações Matemáticas

- ADD <destino, origem> – Executa destino = destino + origem

Essa operação aceita:

ADD reg, reg

ADD reg, endereço

ADD reg, constante

- SUB <destino, origem> – Executa destino = destino - origem

Essa operação aceita:

SUB reg, reg

SUB reg, endereço

SUB reg, constante

- INC <destino> – Executa destino = destino + 1

Essa operação aceita:

INC reg

- DEC <destino> – Executa $\text{destino} = \text{destino} - 1$
Essa operação aceita:
DEC reg
- MUL <multiplicador> –
Executa $\text{Registrador A} = \text{Registrador A} * \text{multiplicador}$
Essa operação aceita:
MUL reg
MUL endereço
MUL constante
- DIV <divisor> –
Executa $\text{Registrador A} = \text{Registrador A} / \text{divisor}$
Essa operação aceita:
DIV reg
DIV endereço
DIV constante

Operações Lógicas

- AND <destino, origem> –
Executa $\text{destino} = \text{destino AND origem}$
Essa operação aceita:
AND reg, reg
AND reg, endereço
AND reg, constante
- OR <destino, origem> –
Executa $\text{destino} = \text{destino OR origem}$
Essa operação aceita:
OR reg, reg
OR reg, endereço
OR reg, constante

Operações de Pulo

- Todas as operações de pulo tem a mesma assinatura:

JMP endereço

- O tipo de pulo ocorrerá se a condição para ele for satisfeita (JE ter o ZF igual a zero e assim por diante). Se o pulo for feito para um espaço vazio de memória, o programa é encerrado.

4. Consultas

<https://schweigi.github.io/assembler-simulator/instruction-set.html>

<https://stackoverflow.com/questions/9617877/assembly-jg-jnle-jl-jnge-after-cmp>

5. Testes

Vamos executar o seguinte teste, um somatório:

MOV A, 10

ADD B, 1

CMP 10, B

JNE [3]

```
vitor@bostonmonster: ~/Vitor Augusto/USP/OCD/EP2
vitor@bostonmonster:~/Vitor Augusto/USP/OCD/EP2 $ java EP2OCD 256
Executar em qual modo?
1 - Ler arquivo
2 - Freestyle
1
Digite o caminho do arquivo:
microprograma.txt
Instrução MOV A, 10 inserida na memória na posição 0 - parametros nas posições 1 e 2
Instrução ADD B, 1 inserida na memória na posição 3 - parametros nas posições 4 e 5
Instrução CMP 10, B inserida na memória na posição 6 - parametros nas posições 7 e 8
Instrução JNE [3] inserida na memória na posição 9 - parametro na posição 10

Instrução: MOV,A,10
Registradores:
AX: 10 - BX: 0 - CX: 0 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: ADD,B,1
Registradores:
AX: 10 - BX: 1 - CX: 0 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: CMP,10,B
Registradores:
AX: 10 - BX: 1 - CX: 0 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: JNE,[3]
Registradores:
AX: 10 - BX: 1 - CX: 0 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)
```

```
vitor@bostonmonster: ~/Vitor Augusto/USP/OCD/EP2
AX: 10 - BX: 9 - CX: 0 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: CMP,10,B
Registradores:
AX: 10 - BX: 9 - CX: 0 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: JNE,[3]
Registradores:
AX: 10 - BX: 9 - CX: 0 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: ADD,B,1
Registradores:
AX: 10 - BX: 10 - CX: 0 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: CMP,10,B
Registradores:
AX: 10 - BX: 10 - CX: 0 - DX: 0
Flags:
ZF: 1 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: JNE,[3]
Registradores:
AX: 10 - BX: 10 - CX: 0 - DX: 0
Flags:
ZF: 1 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)
vitor@bostonmonster:~/Vitor Augusto/USP/OCD/EP2 $
vitor@bostonmonster:~/Vitor Augusto/USP/OCD/EP2 $
```


Outro teste, que calcula B elevado à C e escreve em A o resultado:

```
MOV B, 2
MOV C, 3
MOV A, 1
CMP C, 0
JE [256]
MUL B
DEC C
JNE [9]
```

```
vitor@bostonmonster: ~/Vitor Augusto/USP/OCD/EP2
vitor@bostonmonster:~/Vitor Augusto/USP/OCD/EP2 $ java EP2OCD 256
Executar em qual modo?
1 - Ler arquivo
2 - Freestyle
1
Digite o caminho do arquivo:
microprograma.txt
Instrução MOV B, 2 inserida na memória na posição 0 - parametros nas posições 1 e 2
Instrução MOV C, 3 inserida na memória na posição 3 - parametros nas posições 4 e 5
Instrução MOV A, 1 inserida na memória na posição 6 - parametros nas posições 7 e 8
Instrução CMP C, 0 inserida na memória na posição 9 - parametros nas posições 10 e 11
Instrução JE [256] inserida na memória na posição 12 - parametro na posição 13
Instrução MUL B inserida na memória na posição 14 - parametro na posição 15
Instrução DEC C inserida na memória na posição 16 - parametro na posição 17
Instrução JNE [9] inserida na memória na posição 18 - parametro na posição 19

Instrução: MOV,B,2
Registradores:
AX: 0 - BX: 2 - CX: 0 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: MOV,C,3
Registradores:
AX: 0 - BX: 2 - CX: 3 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: MOV,A,1
Registradores:
AX: 1 - BX: 2 - CX: 3 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: CMP,C,0
Registradores:
```

```
vitor@bostonmonster: ~/Vitor Augusto/USP/OCD/EP2
Registradores:
AX: 4 - BX: 2 - CX: 1 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: CMP,C,0
Registradores:
AX: 4 - BX: 2 - CX: 1 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: JE,[256]
Registradores:
AX: 4 - BX: 2 - CX: 1 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: MUL,B
Registradores:
AX: 8 - BX: 2 - CX: 1 - DX: 0
Flags:
ZF: 0 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: DEC,C
Registradores:
AX: 8 - BX: 2 - CX: 0 - DX: 0
Flags:
ZF: 1 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)

Instrução: JNE,[9]
Registradores:
AX: 8 - BX: 2 - CX: 0 - DX: 0
Flags:
ZF: 1 - SF: 0 - OF: 0
(Pressione Enter para a próxima instrução)
vitor@bostonmonster:~/Vitor Augusto/USP/OCD/EP2 $
```