Professor Eduardo Kugler Viegas
Performance em Sistemas Ciberfísicos

Atividade Implementação CPU e ULA

Considere que você é responsável pela implementação de uma CPU com uma ***Unidade de Controle,*** uma ***ULA*** e uma ***Memória Cache*** em nível de software através da linguagem de programação Python.

A sua CPU possui cinco registradores sendo eles:

Tabela 1. Registradores

Registrador	ID	Descrição
CP	0x01	Registrador de ponteiro de programa
AX	0x02	Registrador de propósito geral
BX	0x03	Registrador de propósito geral
CX	0x04	Registrador de propósito geral
DX	0x05	Registrador de propósito geral

Além disso sua CPU também possui suporte a Flag Zero (ZF), e fornece suporte ao seguinte conjunto de instruções (*Instruction Set Architecture,* ISA):

Tabela 2. Instruction Set Architecture

#	Instrução	OPCode	Descrição
0	ADD Reg, Byte	0x00 IDReg1 Byte	Soma valor de Byte ao registrador IDReg1
1	ADD Reg, Reg	0x01 IDReg1 IDReg2	Soma valor do registrador IDReg2 ao registrador IDReg1
2	INC Reg	0x10 IDReg1	Incrementa valor de IDReg1
3	DEC Reg	0x20 IDReg1	Decrementa valor de IDReg1
4	SUB Reg, Byte	0x30 IDReg1 Byte	Subtrai valor de Byte do registrador IDReg1
5	SUB Reg, Reg	0x31 IDReg1 IDReg2	Subtrai valor do registrador IDReg2 do registrador IDReg1
6	MOV Reg, Byte	0x40 IDReg1 Byte	Soma valor de Byte ao registrador IDReg1
7	MOV Reg, Reg	0x41 IDReg1 IDReg2	Soma valor do registrador IDReg2 ao registrador IDReg1
8	JMP Byte	0x50 Byte	Salta execução da CPU para o endereço Byte
9	CMP Reg, Byte	0x60 IDReg1 Byte	Compara valor do registrador IDReg1 com Byte. Flag Zero ZF é definida como 1 caso os valores sejam iguais
10	CMP Reg, Reg	0x61 IDReg1, IDReg2	Compara valor do registrador IDReg1 com registrador IDReg2. Flag Zero ZF é definida como 1 caso os valores sejam iguais
11	JZ Byte	0x79 Byte	Salta execução da CPU para endereço Byte caso a flag ZF seja 1

Baseado na especificação, os componentes da sua CPU comunicam entre si através da seguinte arquitetura:

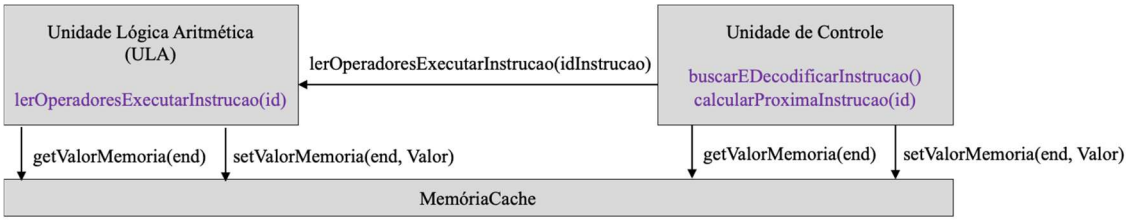

Figura 1. **Arquitetura da CPU**

Considerando a arquitetura acima descrita, você deverá implementar as seguintes funções a serem adicionadas a CPU:

1. Buscar Instrução e Decodificar Instrução

Implemente a função *buscarEDecodificarInstrucao*. A função deve retornar o ID da instrução a ser executada (campo # da tabela 2).

A função também deve exibir no terminal uma string referente a instrução que será executada caso CPU_DEBUG seja verdadeiro. Exemplo de saída:

```
buscarEDecodificarInstrucao: instrução ADD Registrador e Byte
buscarEDecodificarInstrucao: instrução MOV Registrador e Byte
buscarEDecodificarInstrucao: instrução JMP Byte
buscarEDecodificarInstrucao: instrução JZ Byte
```

2. Calcular Próximo CP

Implemente a função *calcularProximaInstrucao*. A função deve retornar atualizar o valor do registrador de contador de programa com base na instrução corrente.

A função também deve exibir no terminal uma string referente ao próximo endereço de memoria de CP caso CPU_DEBUG seja verdadeiro. Exemplo de saída:

```
buscarEDecodificarInstrucao: instrução ADD Registrador e Byte
calcularProximaInstrucao: mudando CP para 4
buscarEDecodificarInstrucao: instrução MOV Registrador e Byte
calcularProximaInstrucao: mudando CP para 7
buscarEDecodificarInstrucao: instrução JMP Byte
calcularProximaInstrucao: mudando CP para 10
buscarEDecodificarInstrucao: instrução JZ Byte
calcularProximaInstrucao: mudando CP para 12
```

Se certifique que o seu código executa corretamente comentando as linhas 14 a 18 para alterar o programa a ser executado

3. Ler Operadores Executar Instrução

Implemente a função *lerOperadoresExecutarInstrucao*. A função deve executar a instrução recebida pelo parâmetro *instrução*.

A função também deve exibir no terminal uma string referente a instrução que será executada caso CPU_DEBUG seja verdadeiro. Exemplo de saída:

```
buscarEDecodificarInstrucao: instrução ADD Registrador e Byte
lerOperadoresExecutarInstrucao: somando 0x05 em BX
calcularProximaInstrucao: mudando CP para 4
buscarEDecodificarInstrucao: instrução MOV Registrador e Byte
lerOperadoresExecutarInstrucao: atribuindo 0x01 em AX
calcularProximaInstrucao: mudando CP para 7
buscarEDecodificarInstrucao: instrução JMP Byte
calcularProximaInstrucao: mudando CP para 10
buscarEDecodificarInstrucao: instrução JZ Byte
calcularProximaInstrucao: mudando CP para 12
```

Se certifique que o seu código executa corretamente comentando as linhas 14 a 18 para alterar o programa a ser executado. Faça o teste de mesa e se certifique que os valores dos registradores estão conforme esperados