Lista 9 - Microprogramação

1. Qual é a diferença entre uma implementação por hardware e uma implementação microprogramada?

R: A diferença principal entre as duas abordagens é que a implementação por hardware é feita diretamente através de circuitos físicos, enquanto a implementação microprogramada usa uma camada de microcódigo para traduzir instruções de máquina em operações a nível de circuito.

2. Qual o nome da classe de máquinas (arquitetura) que não são microprogramadas? E as que são microprogramadas?

R: A classe de máquinas que não são microprogramadas é conhecida como RISC (Reduced Instruction Set Computer), que se refere a um tipo de design de arquitetura de computador que utiliza um conjunto pequeno e altamente otimizado de instruções.

Por outro lado, a classe de máquinas que são microprogramadas é conhecida como CISC (Complex Instruction Set Computer). Essas máquinas utilizam uma camada de microcódigo para traduzir instruções de máquina em operações a nível de circuito.

3. Qual a função do nível da microprogramação?

4. Em arquiteturas microprogramadas há o que chamamos de **instruções** e o que chamamos de **microinstruções**. Qual a diferença entre elas? Faça uma relação com a arquitetura RISC-V estudada anteriormente.

5. Quais as vantagens da microprogramação?

R:

* Sistematização do controle via programação
* Há a possibilidade de ganhar desempenho (ex.: executando algumas microinstruções em paralelo)
* Compatibilidade entre conjuntos de microinstruções em máquinas de uma mesma série (ex.: Intel 286 e 386)
* Emulação: interpretação de instruções de uma máquina em outra máquina.

6. Após alguma operação na ALU, a mesma emite dois sinais de controle denominados **N** e **Z**. O que eles representam?

Os sinais de controle N e Z são emitidos pela Unidade Lógica e Aritmética (ALU) após a realização de uma operação. Eles são usados para indicar o resultado da operação:

N: Este sinal indica que o valor de saída da ALU é negativo.

Z: Este sinal indica que o valor de saída da ALU é zero.

Esses sinais são úteis para tomar decisões com base no resultado da operação realizada pela ALU, como em operações de comparação ou instruções de salto condicional.

7. Uma das operações realizadas pela ALU é simplesmente a repetição da entrada para a saída, sem qualquer alteração nos bits dessa entrada. Qual o objetivo dessa operação?

O objetivo dessa operação é permitir o movimento e o uso direto de dados dentro do processador. Por exemplo, pode ser usada quando os dados precisam ser movidos de um registrador para outro, ou de um registrador para a memória, ou vice-versa. Também pode ser usada quando os dados de entrada precisam ser usados diretamente como resultado de uma instrução, sem qualquer processamento adicional.

8. Considere a microinstrução de número 3 abaixo. Explique por que ela realiza a soma de IR com IR e após, realiza ainda um shift-left sobre o resultado dessa soma.

|  |  |
| --- | --- |
| 3 | TIR:=lshift(IR+IR); if *n* goto 19; |

A soma de IR com IR, seguida por um deslocamento à esquerda (lshift) no resultado equivale a uma operação de multiplicação por 2, daria 2 shifts.

9. Considere a microinstrução de número 22 abaixo. Explique para que ela serve utilizando a instrução de mais alto nível (assembly) **JZER** como exemplo e considerando que **JZER** se encontra em IR.

|  |  |
| --- | --- |
| 22 | PC:=band (IR, AMASK); goto 0; |

A microinstrução está atualizando o valor do PC. Ela está pegando o valor atual no IR, que contém a instrução JZER (Jump if Zero), e realizando uma operação de “bitwise AND” com AMASK, que é uma máscara de endereço usada para extrair o campo de endereço da instrução. O resultado dessa operação é então usado para atualizar o PC. Isso extrai o endereço de desvio da instrução JZER e atualiza o PC com esse endereço.

10. Uma instrução de mais alto nível (assembly) possui as seguintes microinstruções que fazem parte de sua execução. Que instrução é esta?

|  |  |
| --- | --- |
|  | **Comando** |
| 0 | MAR:=PC; RD; |
| 1 | PC:=PC+1; RD; |
| 2 | IR:=MBR; if N *goto 28;* |
| 28 | TIR:=lshift(IR+IR); if N goto 40; |
| 40 | Tir:=lshift(TIR); if N goto 46; |
| 46 | Tir:=lshift(tir); if n goto 50; |
| 50 | Tir:= lshift(tir) if n goto 65; |
| 51 | Tir:=lshift(tir) if n goto 59; |
| 59 | ALU:=TIR; if N goto 62 |
| 62 | MAR:=SP; SP:=SP+1; RD; |
| 63 | Rd; |
| 64 | AC:=MBR; goto 0; |

A sequência de microinstruções parece implementar uma instrução de assembly que envolve um deslocamento à esquerda e uma operação de pilha.

As primeiras três microinstruções (0, 1 e 2) são típicas de um ciclo de busca de instrução. Elas buscam a próxima instrução a ser executada da memória e a colocam no Registrador de Instrução (IR).

As microinstruções 28, 40, 46, 50 e 51 realizam uma série de deslocamentos à esquerda no valor em IR. Isso é equivalente a multiplicar o valor por 2 para cada deslocamento.

A microinstrução 59 move o resultado dos deslocamentos para a ALU.

As microinstruções 62, 63 e 64 parecem implementar uma operação de pilha. Elas incrementam o Ponteiro de Pilha (SP), lêem um valor da memória no local apontado por SP e o movem para o Acumulador (AC).

Com base nessa análise, a instrução de assembly correspondente poderia ser algo como SHL (Shift Left) seguido por POP (Pop da pilha).