**Exercício 1**

* **Registrador de Instrução (IR)** de **38 bits**
* **Dois operandos** de **16 bits cada**

**Cálculo**

* **Dois operandos** de 16 bits cada → **16 + 16 = 32 bits**
* O restante dos **38 bits do IR** será para o **código de operação (opcode)**

**Quantidade de bits para o opcode:**  
Tamanho total da instrução - Tamanho dos operandos = **38 - 32 = 6 bits**

**Influência do campo de operação (opcode)**

O campo de operação (opcode) define a quantidade de diferentes instruções que o processador pode reconhecer.  
Com **6 bits**, podemos representar **2⁶ = 64 instruções diferentes**.  
Quanto maior o campo de operação, mais instruções distintas o processador pode suportar.

**Exercício 2**

**Vantagem do pipeline sobre processadores sem pipeline**

Em um processador sem pipeline, cada instrução é executada de forma sequencial, sem sobreposição.  
Já em um processador com pipeline, várias instruções são processadas simultaneamente, em diferentes estágios, melhorando a eficiência.

**Cálculo do tempo total para 12 instruções**

* Cada estágio do pipeline demora 5 ns
* Um pipeline precisa de N + (k - 1) ciclos para concluir N instruções, onde k é o número de estágios.

Supondo um pipeline de 5 estágios, o tempo total para executar 12 instruções será:

T=(12+5−1)×5 ns=16×5=80 ns

**Resposta final:** **80 ns**

Em um processador sem pipeline, o tempo seria muito maior, pois cada instrução levaria 5 × 5 = 25 ns. Para 12 instruções:

12×25=300 ns

Portanto, o pipeline reduz o tempo em 300 ns → 80 ns, aumentando o desempenho.

**Exercício 3**

**(a) Tempo para uma instrução sem desvios**

Se cada estágio demora **5 ns**, uma instrução completa seu ciclo em **5 × 5 = 25 ns**.

**(b) Impacto do desvio na quarta instrução**

* Sem desvios, 10 instruções levariam: (10+5−1)×5=14×5=70 ns
* O desvio interrompe o pipeline e força a reinicialização, gerando um atraso de 5 ciclos.
* Com esse atraso, o tempo total aumenta para 70 ns + (5 × 5 ns) = 95 ns.

**Conclusão:** O desvio reduz a eficiência do pipeline, pois causa bolhas (stalls) e aumenta o tempo de execução.

**Exercício 4**

Instrução: LOAD R1, (R2)

A instrução carrega um valor de memória no registrador R1, sendo que o endereço da memória está armazenado em R2.

**Modo de endereçamento usado**

Esse é o modo de endereçamento indireto por registrador.

Como funciona?

* O registrador **R2** contém um endereço de memória.
* O processador acessa esse endereço e carrega o valor contido nele para **R1**.

Vantagem: Permite acessar posições variáveis na memória, facilitando estruturas como arrays e ponteiros.

**Exercício 5**

**(a) Por que o endereçamento por registrador é eficiente?**

* Operações dentro dos registradores são **mais rápidas** do que acessar a memória.
* Reduz a necessidade de acessos à memória RAM, minimizando gargalos.
* Ideal para instruções frequentes, como somas e comparações.

**(b) Comparação entre o modo registrador e o modo direto**

| **Modo** | **Vantagens** | **Desvantagens** |
| --- | --- | --- |
| **Registrador** | Muito rápido (acesso interno), reduz gargalos. | Limitado ao número de registradores disponíveis. |
| **Direto** | Simples de entender e usar. | Mais lento, pois exige acesso à memória principal. |

**Conclusão:**

* O modo por registrador é melhor para cálculos rápidos.
* O modo direto é útil para acessar dados fixos da memória.