Exercício 1:

- **d) RAM é volátil e permite leitura e escrita, enquanto ROM é apenas leitura.**

Resposta correta: A RAM é volátil, permitindo leitura e escrita. A ROM é não-volátil, sendo somente leitura.

Exercício 2:

a) O tamanho da memória em posições (endereços).

- Uma largura de endereço de 32 bits significa que o processador pode acessar 2^{32} posições de memória.
- Portanto, o tamanho da memória será:

  2^{32} = 4.294.967.296 posições

b) A capacidade máxima de armazenamento, considerando que cada posição armazena 8 bits.

- Como cada posição armazena **8 bits** (1 byte), a capacidade total será:

  4.294.967.296 posições x 1 byte = 4.294.967.296 bytes = 4GB

Exercício 3:

- **c) Gerar os sinais de controle necessários para operações de leitura e escrita na memória.**

Resposta correta: O **controlador de memória** é responsável por **gerar os sinais** para **ler e escrever** na memória.

Exercício 4:

A hierarquia de memória é uma organização das diferentes camadas de memória com base no desempenho (velocidade) e no custo de armazenamento. O objetivo é equilibrar a velocidade e o custo do acesso à memória, com dados frequentemente acessados armazenados em camadas mais rápidas e caras.

1. **Registradores**:
   - Mais rápidos e menor capacidade.
   - Armazenam dados temporários usados diretamente pelo processador.
2. **Cache**:
   - Muito rápida, mas de capacidade limitada.
   - Armazena dados frequentemente usados, reduzindo a necessidade de acessar a memória principal.
3. **Memória Principal (RAM)**:

    - Mais lenta que a cache, mas maior capacidade.
    - Armazena dados temporários e programas em execução.
4. **Memória Secundária (HD/SSD)**:
    - Mais lenta e com capacidade muito maior.
    - Armazena dados permanentemente, como sistemas operacionais, arquivos e programas.

Exercício 5:

1. **Localidade Temporal**:
    - Refere-se ao princípio de que, se um dado ou instrução foi acessado recentemente, há uma boa chance de que será acessado novamente em breve.
    - Exemplo: Em loops de programas, onde uma variável é acessada várias vezes.
2. **Localidade Espacial**:
    - Refere-se ao princípio de que, se um dado ou instrução foi acessado, é provável que os dados adjacentes também sejam acessados em breve.
    - Exemplo: Ao acessar um array ou estrutura de dados sequencialmente, o próximo elemento ou próximo endereço de memória será acessado logo em seguida.

Impacto no desempenho da cache:

- A localidade temporal permite que dados usados recentemente fiquem na cache, acelerando o acesso.
- A localidade espacial permite que blocos de dados próximos sejam carregados na cache, melhorando a eficiência do sistema.

Exemplo prático: Em um programa de ordenação de array, ao acessar elementos adjacentes, a localidade espacial ajuda a manter esses dados na cache, enquanto a localidade temporal garante que, após várias iterações, os dados mais acessados ainda estarão disponíveis rapidamente.