4.6 EXERCÍCIOS PROPOSTOS

1. Uma memória bem dimensionada e com largura de banda adequada é fundamental para o bom funcionamento do computador. Com relação aos tipos de memórias utilizadas de um computador, responda:
   a) Qual a diferenciação entre memórias de acesso sequencial e aleatório?
   Resposta: Memórias de acesso sequencial são aquelas em que os dados são lidos ou gravados sequencialmente, em uma ordem específica, exigindo que a leitura/gravação seja feita em uma posição após a outra, como as memórias FIFO. Já as memórias de acesso aleatório permitem o acesso direto a qualquer posição de memória, independentemente da ordem, com uso de um endereço, como a memória principal do computador.
   b) Qual a diferenciação entre memórias semicondutoras voláteis e não voláteis?
   Resposta: Memórias semicondutoras voláteis são aquelas que perdem seus dados quando a energia é desligada, ou seja, os dados armazenados são temporários. A memória principal do computador é um exemplo típico de memória volátil. Já as memórias semicondutoras não voláteis são capazes de manter os dados mesmo quando a energia é desligada, tornando-as ideais para armazenamento permanente. Exemplos incluem memórias ROM (Read-Only Memory), memórias Flash e memórias EEPROM (Electrically Erasable Programmable Read-Only Memory), utilizadas, por exemplo, para armazenar a BIOS/UEFI.
   c) Relacione os principais tipos de memórias semicondutoras não voláteis.
   Resposta: Os principais tipos de memórias semicondutoras não voláteis são:
      - ROM (Read-Only Memory): Contém informações pré-gravadas de fábrica e não pode ser alterada pelo usuário.
      - PROM (Programmable Read-Only Memory): Pode ser programada pelo usuário uma única vez com o uso de um dispositivo de programação conectado a um computador.
      - EPROM (Erasable Programmable Read-Only Memory): Pode ser apagada por exposição à luz ultravioleta e então reprogramada em dispositivo de programação conectado a um computador.
      - EEPROM (Electrically Erasable Programmable Read-Only Memory): Pode ser apagada eletricamente e reprogramada múltiplas vezes sem necessidade de retirada da placa de circuito.
      - Flash Memory: Tipo especial de EEPROM, com a capacidade de apagar e reprogramar blocos de memória em vez de bytes individuais, sem necessidade de retirada do circuito, que pode ser lida e escrita com alta velocidade, em comparação aos demais tipos.
   d) Quais as principais diferenças entre as memórias estáticas e dinâmicas?
   Resposta: As principais diferenças entre as memórias estáticas (SRAM - Static Random Access Memory) e dinâmicas (DRAM - Dynamic Random Access Memory) estão relacionadas ao seu funcionamento, características elétricas e desempenho, tais como:
      i. Tecnologia de armazenamento: A memória estática é construída utilizando seis transistores em uma configuração de flip-flop, formando uma célula de memória. Essa configuração mantém o estado dos dados sem a necessidade de atualização contí-

nua. A memória dinâmica utiliza um transistor e um capacitor para armazenar um bit na célula de memória. Como os capacitores tendem a perder carga com o tempo, os dados precisam ser periodicamente atualizados para evitar a perda de informação.

ii. Volatilidade: A memória estática é volátil, ou seja, os dados são mantidos enquanto houver energia elétrica, mas são perdidos quando o computador é desligado. A memória dinâmica também é volátil, porém, devido ao seu funcionamento baseado em capacitores, requer atualizações periódicas (*refresh*) para manter os dados armazenados. Sem essas atualizações, os dados são perdidos.

iii. Acesso e velocidade: A memória estática possui um acesso muito mais rápido do que a memória dinâmica, pois não requer o processo de atualização, o que simplifica seus circuitos. Adicionalmente, seus transistores são maiores, fornecem mais corrente e podem ser lidos sem perda de conteúdo. A memória dinâmica tem um acesso mais lento, entre outros fatores devido ao processo de atualização necessário para manter os dados. Ainda têm a necessidade de reescrita dos dados durante os ciclos de leitura, pois ela é destrutiva, ou seja, drena o seu capacitor durante essa operação.

iv. Densidade de armazenamento: A memória estática requer mais transistores por célula de memória, o que a torna menos densa em relação à DRAM. Isso faz com que a SRAM seja mais cara e ocupe mais espaço em um *chip*. A memória dinâmica é mais densa, pois requer menos componentes para armazenar cada bit de dados. Isso a torna mais econômica em termos de espaço e custo.

v. Consumo de energia: A memória estática consome mais energia do que a memória dinâmica devido à sua configuração de flip-flop que requer mais e maiores transistores em cada célula. Por conta de sua configuração mais simples, a memória dinâmica consome menos energia em comparação com a SRAM.

e) Por que as memórias dinâmicas necessitam de atualização periódica do seu conteúdo?
Resposta: As memórias dinâmicas (DRAM) são baseadas em capacitores para armazenar os dados. Devido às perdas naturais do capacitor, as informações armazenadas nele começam a se dissipar ao longo do tempo. Portanto, para manter a integridade dos dados, é necessário atualizar periodicamente (*refresh*) a carga elétrica nos capacitores da DRAM.

f) Por que as leituras nas memórias dinâmicas são destrutivas? Qual a solução adotada para este problema?
Resposta: As leituras nas memórias dinâmicas (DRAM) são destrutivas porque o processo de leitura envolve a descarga do capacitor para determinar o valor do bit. Essa descarga esvazia o capacitor, fazendo com que a informação armazenada seja perdida. Para resolver esse problema, as memórias DRAM utilizam um circuito adicional, que reescreve o valor lido no capacitor.

g) Qual a classificação completa do tipo de memória utilizada como memória principal nos modernos computadores?
Resposta: A memória principal é volátil, o que significa que os dados armazenados nela são temporários e são perdidos quando o computador é desligado. A memória principal é um tipo de memória de acesso aleatório, o que significa que o processador pode ler e escrever dados em qualquer posição de memória diretamente. Cada célula de memória

possui um endereço exclusivo, e o processador usa esse endereço, formado por bits, para acessar ou armazenar dados na memória. A memória principal é composta por chips de memória semicondutora dinâmica devido à sua alta densidade e menor custo em comparação com as memórias estáticas. Modernamente, a memória principal é composta por memória semicondutoras síncronas, ou seja, que fazem as transferências de dados em rajada, sincronizadas pelo relógio do barramento do processador.

2. Quantos bits de endereço são necessários para endereçar um conjunto de memória de 2 GiB?
   Resposta: Para endereçar um conjunto de memória de 2 GiB, precisamos calcular quantos bits são necessários para representar cada endereço possível. Um Gibibyte (GiB) é igual a 2^{30} bytes, ou seja, 1.073.741.824 bytes. Para determinar a quantidade de bits de endereço necessários, podemos utilizar a fórmula: Bits de endereço = log_2(Tamanho da memória em bytes) No caso, a memória tem 2^{31} bytes, então: Bits de endereço = $log_2(2^{31})$ = 31 bits Portanto, são necessários 31 bits de endereço para endereçar um conjunto de memória de 2 GiB.
3. Quantas pastilhas são necessárias para montar um módulo de memória de 2 GiB com 64 bits de largura, e qual a capacidade de cada pastilha em bytes, se cada pastilha tem um barramento de dados de 8 bits?
   Resposta: Para montar um módulo de memória de 2 GiB com 64 bits de largura, precisamos calcular quantas pastilhas de memória são necessárias e qual a capacidade de cada pastilha. Primeiro, vamos converter 2 GiB para bytes:
   2 GiB = 2 * 1024 * 1024 * 1024 bytes = 2.147.483.648 bytes
   Agora, precisamos determinar quantas pastilhas de memória são necessárias para atingir a largura de 64 bits. Cada pastilha tem um barramento de dados de 8 bits, então:
   Número de pastilhas = Largura do barramento de dados / Largura do módulo de memória
   Número de pastilhas = 64 bits / 8 bits = 8 pastilhas
   Portanto, são necessárias 8 pastilhas de memória para montar um módulo de memória de 2 GiB com 64 bits de largura. Agora, para determinar a capacidade de cada pastilha, basta dividir a capacidade total do módulo pelo número de pastilhas:
   Capacidade de cada pastilha = Capacidade total do módulo / Número de pastilhas
   Capacidade de cada pastilha = 2.147.483.648 bytes / 8 pastilhas = 268.435.456 bytes
   Cada pastilha tem uma capacidade de 268.435.456 bytes ou 256 MiB.
4. Quantos bits são necessários para um código de correção de erro com correção de um bit errado e detecção de até dois bits errados em uma palavra de memória com largura de dados de 64 bits? Justifique.
   Resposta: Um código de Hamming é um tipo de código de correção de erros que pode detectar e corrigir erros na transmissão de dados. Para um código de Hamming que pode corrigir erros de 1 bit em uma palavra de dados de 64 bits, precisamos calcular o número de bits de paridade necessários. Digamos que o número de bits de paridade seja ‘p‘. De acordo com a fórmula do código de Hamming, temos $2^p \geq p + 64 + 1$. Resolvendo esta desigualdade, obtemos ‘p = 7‘. Portanto, precisamos de pelo menos 7 bits de paridade para implementar um código de Hamming que possa corrigir erros de 1 bit. Para a deteção de 2 bits errados, precisamos adicionar um bit a mais de paridade simples, totalizando então 8 bits para correção de erro com correção de um bit errado e detecção de até dois bits errados em uma palavra de memória com largura

de dados de 64 bits.

5. As memórias síncronas são utilizadas como memória principal nos modernos computadores. Com relação às memórias síncronas, responda:
   a) Por que as memórias síncronas passaram ser utilizadas pelos modernos processadores?
   Resposta: As memórias síncronas passaram a ser utilizadas pelos modernos processadores devido à necessidade de acompanhar a crescente velocidade e eficiência dos processadores, que transferem a informação em blocos entre a memória principal e a sua memória cache interna. Assim, após o endereço inicial, várias posições sequenciais de memória são transferidas sucessivamente a cada ciclo de relógio.
   b) Quais as principais diferenças entre as memórias síncronas do tipo SDRAM, DDR e DDR2?

   Resposta: As principais diferenças são:
   - Transferência de dados: As SDRAM (Single Data Rate Synchronous DRAM) são a primeira geração de memórias síncronas, capazes de transferir dados apenas uma vez a cada ciclo de relógio. As DDR e DDR2 podem transferir os dados duas vezes a cada ciclo de relógio, tanto na transição positiva como negativa do relógio.
   - Bancos de matriz de células de memória: Uma das principais diferenças entre a memória DDR e a memória DDR2 é a forma como elas organizam seus bancos de matriz de células de memória. A memória DDR tem dois bancos de memória, já a DDR2 tem quatro bancos de memória, sendo que o barramento externo trabalha no dobro da frequência da matriz de células de memória.
   - Velocidade do relógio: A SDRAM opera com velocidades de relógio mais baixas em comparação com a DDR e DDR2. As memórias DDR e DDR2 possuem frequências de relógio mais altas, o que permite uma maior taxa de transferência de dados.
   - Tensão de Operação: A DDR e DDR2 operam com uma tensão de alimentação mais baixa do que a SDRAM. A SDRAM geralmente opera com 3,3V, enquanto as memórias DDR operam com 2,5V e as DDR2 com 1,8V. Isso ajuda a economizar energia e reduzir o consumo do sistema.
   - Capacidade de Armazenamento: As memórias DDR e DDR2 têm maior densidade de armazenamento em comparação com a SDRAM. Isso significa que é possível ter módulos de memória DDR e DDR2 com capacidades maiores, permitindo que os computadores tenham mais memória disponível para executar tarefas complexas e exigentes.
   - Compatibilidade: Devido às diferenças de tensão, design e pinagem, os módulos de memória DDR e DDR2 não são compatíveis com os slots de memória da SDRAM, e vice-versa. Isso significa que um computador projetado para utilizar SDRAM não pode ser atualizado para DDR ou DDR2 sem substituir os módulos de memória e, da mesma forma, um computador projetado para utilizar DDR ou DDR2 não pode usar módulos de memória SDRAM.

   c) Quais as principais diferenças entre as memórias síncronas do tipo DDR3 e DDR4?
   Resposta: A matriz de memória dos módulos DDR3 estão organizadas internamente em oito bancos que trabalham a um quarto da frequência do barramento externo, o requer

um buffer capaz de armazenar 8 bits em comparação com os 4 bits da memória DDR2. As memórias DDR3 são alimentadas com 1,5 volts, possuem latência típica de 5 a 17 ciclos de relógio e velocidades de relógio do barramento externo de até 1600 MHz. As matrizes de memória dos módulos DDR4, introduzidas em 2014, trabalham a um oitavo da frequência do barramento externo com 16 bancos de memória internos, sendo alimentadas com 1,2 volts e possuem latência inicial típica de 10 a 19 ciclos de relógio. As memórias DDR3 tem uma interface de 240 pinos, enquanto as memórias DDR4 tem uma interface de 288 pinos e são mecanicamente diferentes. A velocidade de relógio da DDR3 chega até 1600 MHz, enquanto a velocidade de relógio da DDR4 vai até 2400 MHz.

6. Com relação às memórias Flash, responda:
   a) O que são memórias Flash e qual a sua utilização nos modernos computadores?
   Resposta: Memórias Flash são um tipo de memória não volátil utilizada nos computadores modernos para armazenamento de dados. Elas retêm as informações mesmo quando a energia é desligada. As memórias Flash são amplamente utilizadas em dispositivos de armazenamento como unidades de estado sólido (SSD), cartões de memória, pen drives e outros dispositivos de armazenamento portáteis. Também são utilizadas em firmware de dispositivos eletrônicos, como BIOS/UEFI de computadores e firmware de smartphones e tablets.
   b) Quais as diferenças entre as memórias Flash do tipo NOR e do tipo NAND?
   Resposta: escritas e isso pode levar até alguns segundos. Contudo, a leitura e escrita de um byte por vez é muito rápida. As memórias Flash do tipo NOR permitem cerca de 100.000 a 1.000.000 ciclos de apagamentos por bloco antes de se esgotarem, dependendo da tecnologia de fabricação. As leituras e escritas nas memórias Flash do tipo NAND são feitas em blocos de 512 bytes (igual ao setor de um disco rígido), de um modo mais rápido que nas memórias NOR. Contudo, tipicamente, páginas com até 4 KiB (oito blocos) são lidas e escritas de uma vez. Antes da escrita, as células são apagadas em blocos de 16 a 512 KiB. Menos cara que a Flash do tipo NOR, a Flash NAND pode ser reescrita cerca de 100.000 de vezes, para as tecnologias mais comuns, como as células de apenas um nível (SLC), diminuindo conforme aumenta o número de bits armazenados por célula. Como resultado, a célula NAND de nível quádruplo (QLC), com quatro bits por célula, é geralmente usada para aplicações de leitura intensiva, e os tipos de NAND com menos bits são usados para cargas de trabalho de escrita intensiva. A memória Flash NOR é geralmente usada em aplicativos que exigem tempos de acesso rápidos para leitura byte a byte, como firmware de dispositivos eletrônicos. Já a memória Flash NAND é usada em dispositivos de armazenamento, como SSDs e cartões de memória, onde a velocidade de gravação é mais importante.
   c) Por que as memórias Flash tem uma vida útil limitada e o que pode ser feito para prolongar sua durabilidade?
   Resposta: As memórias Flash têm uma vida útil limitada devido ao processo de gravação e apagamento que ocorre nas células de memória. Cada célula pode suportar apenas um número limitado de ciclos de gravação/apagamento antes de começar a se degradar. Para prolongar a durabilidade das memórias são implementados algoritmos de gerenciamento

de desgaste que distribuem uniformemente as operações de escrita e apagamento em todas as células da memória, evitando o desgaste excessivo de algumas áreas específicas.

d) As operações de re-escrita ou apagamento podem ser feitas byte a byte? Explique.

Resposta: As operações de reescrita ou apagamento em memórias Flash são geralmente feitas em nível de página (em memórias NAND) ou em nível de bloco (em memórias NOR e NAND). As páginas e blocos são unidades de escrita/apagamento que consistem em várias células de memória. Não é possível realizar operações de reescrita ou apagamento byte a byte em memórias Flash, pois essas operações são realizadas em nível de página ou bloco, o que requer o acesso e modificação de múltiplas células simultaneamente. Antes da escrita, as células NAND são apagadas em blocos de 16 a 512 KiB. As células NOR devem ser apagadas em blocos de 64, 128 ou 256 KiB antes de serem escritas e isso pode levar até alguns segundos.

7. A utilização da hierarquia de memória visa oferecer um sistema de memória com tempo de acesso e capacidade de armazenamento adequadas, mas dentro de parâmetros de custo razoáveis. Com relação à hierarquia de memória de um computador, responda:

a) Quais são os principais componentes da hierarquia de memória do computador?

Resposta: Os principais componentes da hierarquia de memória do computador são:

i. Registradores: São os níveis mais rápidos e mais próximos do processador. São usados para armazenar dados e instruções temporariamente enquanto estão sendo processados. Os registradores têm a menor capacidade de armazenamento, mas o tempo de acesso é extremamente rápido.

ii. Cache: A cache é uma memória de acesso rápido e pequena que fica entre o processador e a memória principal. Ela armazena cópias de dados e instruções frequentemente acessados pelo processador, reduzindo o tempo de acesso à memória principal.

iii. Memória Principal: A memória principal é a principal memória de trabalho do computador, onde os programas e dados são armazenados enquanto estão em uso. Ela é mais lenta que a memória cache, mas tem uma capacidade maior.

iv. Memória Secundária (Disco Rígido, SSD, etc.): A memória secundária é uma memória de armazenamento não volátil que é usada para armazenar dados permanentemente, mesmo quando o computador está desligado. Ela tem uma capacidade muito maior que a memória principal, mas é mais lenta.

b) Qual a quantidade mínima de informação que é movimentada entre cada um dos seus níveis?

Resposta: Os registradores movimentam quantidades de memória iguais ao seu tamanho. Para os demais níveis, contudo, as informações são movimentadas em blocos, se aproveitando da localidade espacial. Entre a memória cache e a memória principal são blocos ou linhas com 32 a 128 bytes, por exemplo. Entre a memória principal e a memória secundária são movimentadas páginas, cujo tamanho pode ter entre 1 e 4 Kibytes.

c) Qual a relação entre custo, tempo de acesso e capacidade de armazenamento dos seus diversos níveis?

Resposta: A relação entre custo, tempo de acesso e capacidade de armazenamento nos diversos níveis da hierarquia de memória é inversamente proporcional. Ou seja, quanto

mais rápido e mais próximo do processador, menor será a capacidade de armazenamento e maior será o custo. Por outro lado, quanto mais distante do processador, maior será a capacidade de armazenamento e menor será o custo, mas o tempo de acesso será mais lento.

d) Quem é o responsável pelo controle da movimentação de informação entre cada um dos seus níveis?

Resposta: O compilador é quem determina quando uma informação vai ser transferida de/-para os registradores, com uso de instruções de load e store, por exemplo. Entre a memória cache e a memória principal a movimentação dos blocos/linhas é feita automaticamente pelo hardware, no caso o controlador de cache, que utiliza uma máquina de estados para isso. Entre a memória principal e a memória principal o controle da movimentação das páginas é feita software, ou seja, o gerenciador de memória virtual do sistema operacional.

e) Explique os conceitos de localidade temporal e espacial e sua importância do funcionamento da hierarquia de memória.

Resposta: Os conceitos de localidade temporal e espacial são fundamentais no funcionamento da hierarquia de memória e têm um papel crucial em melhorar o desempenho dos sistemas de computação.

- Localidade Temporal: A localidade temporal refere-se à propriedade de que dados ou instruções recentemente acessados têm uma alta probabilidade de serem acessados novamente em um futuro próximo. Em outras palavras, quando um item é referenciado na memória, é provável que ele seja acessado novamente várias vezes em um curto espaço de tempo. Isso acontece devido ao fluxo de execução dos programas e à repetição de padrões de acesso aos dados.

  Por exemplo, em um laço em um programa, os dados dentro do laço são acessados repetidamente a cada iteração, demonstrando a localidade temporal. Essa propriedade é explorada pelas memórias cache, que mantêm os dados recentemente acessados em níveis mais rápidos e próximos ao processador para reduzir o tempo de acesso.
- Localidade Espacial: A localidade espacial refere-se à propriedade de que, quando um item é referenciado na memória, é provável que os itens vizinhos também sejam acessados em um futuro próximo. Em outras palavras, os dados próximos a um item recentemente acessado têm alta probabilidade de serem necessários em seguida. Isso ocorre devido ao princípio de que programas tendem a acessar dados próximos ou contíguos em seus endereços.

  Por exemplo, quando um bloco de dados é buscado na memória, é provável que os dados ao redor desse bloco também sejam necessários, como em um acesso sequencial a um vetor. Essa propriedade é explorada pelas memórias cache, que armazenam blocos de dados inteiros em vez de apenas itens individuais para aproveitar a localidade espacial e reduzir o tempo de acesso.

8. Os modernos processadores não podem prescindir do uso da memória cache para manterem padrões de desempenho adequados. Com relação à memória cache responda:

   a) Qual o principal motivo para utilização de memória cache nos modernos computadores?

   Resposta: O principal motivo para a utilização de memória cache nos modernos computa-

dores é reduzir o tempo de acesso à memória principal (RAM) pelo processador. A memória cache é uma memória de acesso mais rápido e de menor capacidade que fica entre o processador e a memória principal. Ela armazena cópias dos dados e instruções frequentemente acessados pelo processador, de forma a fornecer acesso mais rápido a esses dados, sem a necessidade de acessar a memória principal toda vez que o processador precisar de um dado.

b) Qual o tipo de memória utilizada nas memórias caches? Por quê?
Resposta: O tipo de memória mais comumente utilizada nas memórias caches é a memória estática de acesso aleatório (SRAM - Static Random Access Memory). Isso ocorre porque a SRAM é mais rápida e consome menos energia do que a memória dinâmica de acesso aleatório (DRAM - Dynamic Random Access Memory) utilizada na memória principal. Embora a SRAM seja mais cara e tenha menor capacidade de armazenamento do que a DRAM, essas características são compensadas pelo ganho significativo de desempenho proporcionado pela memória cache.

c) Explique como o conceito de localidade espacial é explorado na organização da memória cache.
Resposta: O conceito de localidade espacial é explorado na organização da memória cache através do uso de blocos de dados (linhas de cache) que contêm não apenas os dados atualmente solicitados pelo processador, mas também os dados próximos a eles. Quando um dado é solicitado pelo processador, é muito provável que dados próximos a ele também sejam necessários em breve, devido à localidade espacial. Assim, a memória cache armazena esses blocos de dados adjacentes, antecipando futuras solicitações e reduzindo a necessidade de buscar dados adicionais na memória principal.
Ao explorar a localidade espacial, a memória cache maximiza as chances de acertar em uma solicitação de dado (hit), proporcionando um acesso rápido ao dado diretamente da cache, sem a necessidade de acessar a memória principal (miss). Essa organização eficiente melhora significativamente o desempenho geral do sistema, uma vez que o processador pode acessar dados frequentemente sem atrasos causados por acessos à memória principal mais lentos.

9. As memórias caches apresentam variações de projeto visando atender diversos requisitos específicos para cada situação. Com relação à isso responda:

a) Quais as vantagens e desvantagens do mapeamento completamente associativo?
Resposta: Vantagens: Qualquer bloco pode ser armazenado em qualquer linha da cache, permitindo um uso mais eficiente do espaço disponível na memória cache. Como consequência, a taxa de acerto é mais alta.
Desvantagens: O *hardware* necessário para implementar o mapeamento completamente associativo pode ser mais complexo e caro, devido à quantidade de comparações necessárias para determinar se houve acerto ou não. Como um bloco pode ser colocado em qualquer linha da cache, a busca do bloco na cache pode ser mais lenta.

b) Quais as vantagens e desvantagens do mapeamento direto?
Resposta: Vantagens: O mapeamento direto é a forma mais simples de mapeamento e requer menos *hardware*, sendo mais rápido para determinar se houve um acerto ou não.

Desvantagens: A ocorrência de conflito de blocos pode levar a uma baixa taxa de acerto em determinadas situações, especialmente quando a cache é muito pequena.

c) Quais as vantagens e desvantagens do mapeamento associativo por conjunto?

Resposta: Vantagens: O mapeamento associativo por conjunto combina características do mapeamento associativo e do mapeamento direto, buscando obter benefícios de ambos. Ao permitir que vários blocos compartilhem o mesmo conjunto, há uma redução no número de conflitos e, portanto, uma melhoria potencial na taxa de acerto. O *hardware* para realizar as comparações é relativamente simples e rápido.
Desvantagens: O mapeamento associativo por conjunto pode ser mais complexo do que o mapeamento direto, mas menos complexo do que o mapeamento completamente associativo. O *hardware* pode ser mais complexo se algoritmos de substituição mais sofisticados forem empregados.

d) Quais são e como se comparam, em termos de complexidade e desempenho, as políticas de substituição de blocos para as caches associativas?

Resposta: As políticas de substituição de blocos para caches associativas são utilizadas para determinar qual bloco será removido da cache quando uma nova linha precisa ser trazida e todos os conjuntos estão ocupados. Algumas das políticas mais comuns são:

- Aleatório: É a política mais simples de implementar, um bloco é escolhido aleatoriamente para ser substituído no conjunto. É uma política de fácil implementação, mas tem problemas de desempenho, pois diminui a taxa de acerto, em esquemas totalmente associativos e nos casos de acesso com alta localidade espacial e temporal.
- Firs In First Out (FIFO): O bloco que está há mais tempo no conjunto é removido. É uma política menos simples para se implementar e pode diminuir a taxa de acerto quando o bloco mais antigo for ainda muito utilizado.
- Least Recently Used (LRU): O bloco a ser substituído no conjunto é aquele que não é referenciado (lido ou escrito) há mais tempo. É considerada a política mais eficiente para melhorar o desempenho da cache, pois tenta manter os blocos mais relevantes na cache. No entanto, sua implementação requer *hardware* adicional para manter o histórico de acesso, tornando-a mais complexa em termos de *hardware*.
- Least Frequently Used (LFU) - Menos Frequente Utilizado: A política LFU substitui o bloco que foi acessado menos vezes. Isso requer a manutenção de um contador de frequência de acesso para cada bloco na cache. No entanto, sua implementação pode ser mais simples que a LRU com taxas de acerto bem próximas.

e) Quais as dificuldades para implementação de uma política de substituição LRU verdadeira? Quais os algoritmos aproximativos normalmente utilizados?
Resposta: A dificuldade na implementação de uma política de substituição LRU verdadeira está principalmente relacionada à necessidade de manter um registro detalhado do histórico de acesso a cada bloco na cache. Isso requer uma estrutura de dados complexa e uma quantidade significativa de *hardware* dedicado ao rastreamento do tempo de acesso de cada bloco. Em caches com tamanhos maiores, a complexidade e o custo podem se tor-

nar proibitivos. Devido às dificuldades de implementar o LRU verdadeiro, muitos sistemas utilizam algoritmos aproximativos, que são mais simples e requerem menos recursos de *hardware*. Alguns dos algoritmos aproximativos comumente utilizados incluem:

- Pseudo-LRU: O algoritmo Pseudo-LRU é uma abordagem aproximada do LRU que usa bits para rastrear o histórico de acesso. Cada conjunto tem um contador ou conjunto de bits que é atualizado para indicar o status dos blocos (usado ou não usado) dentro do conjunto. Essa abordagem é mais simples do que o LRU verdadeiro.
- Least Frequently Used (LFU): Embora o LFU não seja um algoritmo LRU, ele é frequentemente usado como uma abordagem aproximativa para substituição de blocos em caches. Ele substitui o bloco menos frequentemente utilizado, o que pode ser uma boa aproximação do LRU em certos casos.

f) Quais as políticas possíveis a serem utilizadas para as operações de escrita com acerto na cache?

Resposta

- Write-through: A escrita é realizada tanto na cache quanto na memória principal simultaneamente. Isso garante a consistência dos dados entre a cache e a memória principal.
- Write-back: A escrita é feita apenas na cache e os dados são posteriormente copiados para a memória principal quando o bloco é substituído na cache. Isso reduz o tráfego na memória principal, mas requer uma lógica adicional para rastrear os blocos sujos (com alterações não refletidas na memória principal).

g) Quais as políticas possíveis a serem utilizadas para as operações de escrita com falha na cache?

Resposta:

- Write-allocate: Quando ocorre uma falha na cache durante uma operação de escrita, o bloco é trazido para a cache e, em seguida, a escrita é realizada na cache. Isso permite que os benefícios da cache sejam aproveitados, mas também aumenta a latência para escritas.

- No-write-allocate: Quando ocorre uma falha na cache durante uma operação de escrita, o bloco não é trazido para a cache e a escrita é realizada diretamente na memória principal. Isso evita a latência adicional da escrita na cache, mas pode resultar em mais acessos à memória principal.

h) Quais as associações mais comuns entre as políticas de escrita?

Resposta: A política de escrita *write-through* é frequentemente usada em conjunto com a política de *no-write-allocate*. A política de escrita *write-back* é frequentemente usada em conjunto com a política *write-allocate*.

i) Em um sistema com cache virtual, por que, quando há troca de contexto entre processos, é necessário fazer-se uma operação de *flush* na memória cache?

Resposta: Quando ocorre uma troca de contexto entre processos em um sistema com cache virtual, é necessário fazer uma operação de flush na memória cache para garantir que o novo processo que será executado não tenha acesso a dados residuais do processo anterior,

já que os endereços virtuais podem sofrer de ”aliasing”e os dados do processo anterior serem acessados incorretamente. Para evitar comportamentos indesejados e garantir a correta execução do novo processo, é necessário limpar a cache antes de iniciar a execução do próximo processo.

j) Quais soluções você acredita que poderiam ser adotadas para evitar a necessidade desta operação de *flush*?
Resposta: Uma solução possível para evitar a necessidade de uma operação de flush completa na cache durante a troca de contexto é armazenar o PID de cada processo junto com o endereço virtual no rótulo da memória cache.

k) Defina o que são os três Cs e qual a sua influência na taxa de acerto da memória cache.
Resposta:
- Falhas compulsórias: São falhas no acesso à cache, causadas pelo primeiro acesso a um bloco que nunca esteve na cache.
- Falhas devido à capacidade: São falhas que ocorrem porque a cache não pode armazenar todos os blocos necessários à execução de um programa.
- Falhas por conflitos ou colisão: São falhas que ocorrem no acesso à cache quando diversos blocos competem pelo mesmo conjunto. Não ocorrem em caches totalmente associativas.

Os três Cs influenciam a taxa de acerto da memória cache, e projetar uma cache eficiente envolve equilibrar esses princípios para atender aos requisitos específicos de cada situação e aplicação.

10. No processador Intel i7 são utilizadas caches multinível, sendo as caches L1 e L2 privativas de cada núcleo e a cache L3 é compartilhada entre todos os núcleos. Em relação a essa organização, responda:

a) Porque são utilizadas caches separadas para dados e instruções no primeiro nível (L1) de cache? Quais as vantagens e desvantagens dessa separação?
Resposta: As caches separadas para dados e instruções no primeiro nível (L1) de cache são utilizadas para melhorar o desempenho do processador. Essa separação é comum em processadores modernos, como o Intel i7. As vantagens são que o processador pode buscar dados e instruções simultaneamente, o que é necessário para manter o *pipeline* do processador funcionando sem paradas frequentes. Também, ao separar os caches, a podemos ter políticas de escrita aplicadas só a cache de dados, assim como capacidades, tamanho de bloco e associatividade diferentes para cada uma das caches. Como desvantagens, o uso de caches separadas para dados e instruções requer mais *hardware* para a implementação deste controle, o que aumenta ligeiramente a complexidade e o custo de fabricação.

b) Qual a finalidade do uso de uma cache de nível 2 privada para cada núcleo?
Resposta: A cache de nível 2 (L2) privada para cada núcleo tem como finalidade fornecer uma camada intermediária de armazenamento entre a cache L1 privada do núcleo e a cache L3 compartilhada, tem menor latência de acesso, diminuindo o tempo médio de acesso. Como cada núcleo tem sua própria cache L2 privada, há menos interferência entre os núcleos durante o acesso aos dados.

c) Quais as finalidades para uso de uma cache de nível 3 compartilhada entre os diversos

núcleos?

Resposta: Uma cache compartilhada de nível 3 permite a redução do tempo de acesso às variáveis compartilhadas entre as diversas *threads* ou processos executando em cada núcleo, evitando a necessidade de ida à memória. Ao aumentar a capacidade da cache de nível 3, há uma maior probabilidade de os dados estarem presentes na cache compartilhada, reduzindo a necessidade de acessar a memória principal. Isso resulta em menor latência de acesso aos dados, melhorando o desempenho geral do sistema executando aplicações paralelas.

11. Resolva os seguintes problemas relativos à organização da memória cache:

a) Considere processador A com uma cache com linhas de 32 bytes e um outro processador B com uma cache com linhas de 128 bytes, mas ambas com mapeamento direto e capacidade igual a 256 Kibytes. A taxa de acerto observada da cache do processador A é de 96% e do processador B é 98%. Esse aumento na taxa de acerto se deve, por hipótese, ao fato de a cache do processador B ter um bloco maior que a cache do processador A. Mas observamos também que o tempo de acerto é de 2 ns em ambos os casos, mas que o tempo de falha da cache A é de 40 ns e da cache B é 100 ns, em função do maior tempo necessário para a busca de um bloco maior na memória. Qual das caches é a mais eficiente?

Resposta: Para determinar qual das caches é mais eficiente, precisamos levar em consideração tanto a taxa de acerto quanto o tempo médio de acesso (tempo de falha) das caches. Vamos calcular o tempo médio de acesso (TMA) para cada cache para cada cache usando a fórmula TMA = (Taxa de acerto * Tempo de acerto) + (Taxa de falha * Tempo de falha):

- Para o processador A: Tempo de acerto = 2 ns.Tempo de falha = 40 ns.
  TMA = (0,96 * 2 ns) + (0,04 * 40 ns) = 1,92 ns + 1,6 ns = 3,52 ns
- Para o processador B: Tempo de acerto = 2 ns. Tempo de falha = 100 ns.
  TMA = (0,98 * 2 ns) + (0,02 * 100 ns) = 1,96 ns + 2 ns = 3,96 ns

Portanto, o tempo médio de acesso da cache do processador A é de 3,52 ns, enquanto o tempo médio de acesso da cache do processador B é de 3,96 ns. Podemos concluir que a cache do processador A é mais eficiente em comparação com a cache do processador B.

b) Considere um sistema constituído de um processador, memória cache de dois níveis e memória principal, no qual o tempo de acesso à memória cache L1 é de 4 ns, a cache nível 2 é 24 ns e à memória principal é de 100 ns. Os tempos já incluem o tempo de busca do bloco ha hierarquia inferior no caso de falha. Constata-se que as taxas de acerto da memória cache são respectivamente 95% e 90%. Calcule o tempo médio de acesso à memória do sistema.

Resposta: Para calcular o tempo médio de acesso à memória do sistema, precisamos levar em consideração as taxas de acerto e os tempos de acesso de cada nível de cache e da memória principal. Vamos calcular o tempo médio de acesso (TMA) para o sistema usando a fórmula:

TMA = (Taxa de acerto * Tempo de acesso) + (Taxa de falha * Tempo de falha)

Vamos calcular o TMA para cada nível de cache e a memória principal:

- Para a memória cache L1:TMA_L1 = (0,95 * 4 ns) + (0,05 * TMA_L2)
- Para a memória cache L2: TMA_L2 = (0,90 * 24 ns) + (0,10 * 100 ns) = 21,6 ns + 10 ns

= 31,6 ns

Agora, substituindo-se o valor obtido na primeira equação temos:

TMA_L1 = (0,95 * 4 ns) + (0,05 * 31,6 ns) = 3,8 ns + 1,58 ns = 5,38 ns

Portanto, o tempo médio de acesso à memória do sistema é de aproximadamente 5,38 ns.

c) Em um sistema de memória com cache operando com o esquema *write-through*, o tempo de falha (memória principal) é de 100 ns e o tempo de acerto (memória cache) é de 5 ns. 80% dos acessos do processador ao sistema de memória são de leitura e os 20% restantes são de escrita. A taxa de acerto na memória cache é de 98%. Qual o tempo de acesso médio a este sistema de memória?

Resposta: Em uma cache write-through, nas operações de escrita, o tempo de acerto é igual ao tempo de falha, pois os dados são escritos tanto na cache quanto na memória principal simultaneamente. Dado que o tempo de acerto (tempo de escrita) é igual ao tempo de falha (100 ns), podemos recalcular o tempo médio de acesso (TMA) ao sistema de memória considerando os acessos de leitura e escrita. Considerando que a taxa de acerto na memória cache é de 98% e que os acessos de leitura correspondem a 80%, temos:

TMA_total = 0,8 ((0,98 * 5 ns) + (0,02 * 100 ns)) + (0,20 * 100 ns) = 0,8 * (4,9 ns + 2 ns) + 20 ns =

TMA_total = 5,52 + 20 = 25,52 ns

Portanto, o tempo médio de acesso a este sistema de memória é de aproximadamente 25,52 ns.

d) Uma memória cache com mapeamento direto com capacidade igual a 4 Mbytes possui 64 bytes em cada linha da cache. Se o endereço do sistema possui 36 bits, quantos bits são gastos para o rótulo, índice e para o *offset*?

Resposta: Para uma memória cache com mapeamento direto, o endereço de memória é dividido em três partes: o rótulo (tag), o índice e o offset. O tamanho do offset é determinado pelo número de bytes em cada linha da cache. Nesse caso, são 64 bytes. Como 64 é igual a 2^6 (2 elevado à potência de 6), são necessários 6 bits para representar o offset.

O tamanho do índice é determinado pelo número de linhas na cache. Nesse caso, a capacidade da cache é de 4 Mbytes, e cada linha possui 64 bytes. Portanto, o número total de linhas é de 4 Mbytes / 64 bytes = 2^{16} (2 elevado à potência de 16). Assim, são necessários 16 bits para representar o índice. Para determinar o tamanho do rótulo (tag), devemos subtrair o número de bits do índice e do offset do tamanho total do endereço.

- Tamanho do endereço = 36 bits (dado no enunciado)
- Tamanho do índice = 16 bits
- Tamanho do offset = 6 bits
- Tamanho do rótulo (tag) = Tamanho do endereço - Tamanho do índice - Tamanho do offset = 36 bits - 16 bits - 6 bits = 14 bits

e) Uma memória cache com mapeamento completamente associativo por possui 2048 linhas de 128 bytes, sendo que o endereço do sistema possui 32 bits. Quantos bits são gastos para o rótulo e para o *offset*?

Resposta: Em uma memória cache com mapeamento completamente associativo, o endereço de memória é dividido em duas partes: o rótulo (tag) e o offset. O tamanho do offset

é determinado pelo número de bytes em cada linha da cache. Nesse caso, são 128 bytes. Como 128 é igual a 2^7 (2 elevado à potência de 7), são necessários 7 bits para representar o offset.

Para determinar o tamanho do rótulo (tag), devemos subtrair o número de bits do offset do tamanho total do endereço.

- Tamanho do endereço = 32 bits (dado no enunciado)
- Tamanho do offset = 7 bits (determinado pelo número de bytes em cada linha da cache)
- Tamanho do rótulo (tag) = Tamanho do endereço - Tamanho do offset = 32 bits - 7 bits = 25 bits

Portanto, na memória cache com mapeamento completamente associativo, são gastos 25 bits para o rótulo e 7 bits para o offset.

12. A memória virtual é um conjunto de *hardware* e *software* que busca uma melhor utilização possível da memória principal do computador. Com relação à memória virtual responda:
    a) Enumere e descreva quais as principais funções que a memória virtual realiza.

    Resposta: O sistema de memória virtual é responsável pelo controle de qual informação vai ficar armazenada na memória principal, à disposição do processador, e qual vai ser movida para a memória secundária. O objetivo principal é oferecer o melhor uso possível para a memória principal e, como consequência, o melhor desempenho possível para os processos ativos. Assim, a memória é compartilhada entre os diversos processos em execução, procurando dar a cada um deles uma fatia de memória proporcional à sua demanda, sem que isso, contudo, comprometa a execução dos demais processos. A memória virtual é um conjunto de hardware e de rotinas do sistema operacional, que além do controle da hierarquia entre a memória principal e a memória secundária, realiza a proteção dos processos, evitando que um processo modifique informações que pertençam a algum outro processo. Finalmente, a memória virtual também faz a translação de endereços virtuais em endereços reais, já que os programas normalmente enxergam um espaço de endereçamento maior que a memória física disponível. Em resumo, a memória virtual realiza três funções principais:
        - Controle da hierarquia entre a memória principal e a memória secundária.
        - Proteção, evitando que um programa modifique informações que pertençam a algum outro.
        - Mapeamento dos endereços de um espaço de endereçamento virtual em endereços físicos.

    b) Qual a definição de página? Qual o tamanho típico de uma página?
    Resposta: Uma página é uma unidade de tamanho fixo de dados que é transferida entre a memória principal e a memória secundária. O tamanho típico de uma página varia dependendo do sistema operacional e da arquitetura do processador, mas geralmente é de 1 KiB e 16 KiB.

    c) O que acontece com um processo quando a página acessada não está na memória principal?

Resposta: Quando um processo acessa uma página que não está na memória principal, ocorre uma falha de página. Nesse caso, o sistema operacional interrompe a execução do processo, retirando-o da fila de prontos, e transfere a página solicitada da memória secundária para a memória principal. Isso pode envolver a substituição de uma página existente na memória principal para liberar espaço para a nova página. Depois que a página solicitada é transferida para a memória principal, o processo é retirado da fila de bloqueados e movido para a fila de prontos para que sua execução seja retomada.

d) Qual a função, tamanho típico e organização do Translation Lookaside Buffer (TLB)?
Resposta: O Translation Lookaside Buffer (TLB) é uma cache de hardware que armazena as traduções recentes de endereços virtuais para endereços físicos. O tamanho típico do TLB varia dependendo da arquitetura do processador, mas geralmente é de algumas centenas de entradas. O TLB é organizado como uma cache completamente associativa ou associativa por conjunto, onde cada entrada contém um endereço virtual, um endereço físico e bits de controle.

e) Quando uma página substituída na memória principal deve ser escrita na área de *swap*?
Resposta: Quando uma página substituída na memória principal deve ser escrita na área de swap depende se a página foi modificada enquanto estava na memória principal. Se a página foi modificada, ela deve ser escrita na área de swap antes de ser substituída para garantir que as alterações não sejam perdidas. Se a página não foi modificada, ela pode ser simplesmente descartada sem precisar ser escrita na área de swap.

f) Qual deve ser o tamanho da área de troca (*swap*) em disco em relação à memória principal?

Resposta: O tamanho ideal da área de troca (swap) em disco em relação à memória principal depende do uso específico do sistema e dos programas em execução.Há algum tempo atrás, a quantidade recomendada de espaço de troca aumentava linearmente com a quantidade de memória principal no sistema. No entanto, os sistemas modernos geralmente incluem dezenas a centenas de Gigabibytes de memória principal. Como consequência, grandes quantidades de memória swap não são efetivas para garantir o funcionamento do sistema, devido ao grande tempo que seria perdido enviando e trazendo quantidades enormes de páginas da memória secundária para a memória principal e vice-versa. Assim, à medida que aumenta a quantidade de memória principal, o tamanho do espaço de troca tende a ser proporcionalmente menor.

g) Qual é a alternativa mais moderna para a utilização da área de troca (*swap*) em disco? Como ela funciona?
Resposta: Uma alternativa mais moderna à utilização da área de troca (swap) em disco é o uso da compactação de memória. A compactação de memória funciona comprimindo os dados armazenados na memória principal para liberar espaço para novas páginas sem precisar transferi-las para a área de swap em disco. Isso pode melhorar o desempenho do sistema, pois a compactação e descompactação dos dados na memória principal é geralmente mais rápida do que transferi-los para o disco.

h) O algoritmo de substituição LRU (menos recentemente utilizado) tem excelente desempenho. Porque ele é tão difícil de implementar? Quais técnicas são utilizadas para aproxima-

ção do algoritmo LRU?

Resposta: O algoritmo de substituição LRU (Menos Recente Utilizado) é conhecido por ter excelente desempenho em teoria, pois, em termos ideais, ele substitui a página que não foi usada há mais tempo, minimizando assim a quantidade de erros de página e maximizando o aproveitamento da memória cache. No entanto, sua implementação prática pode ser complexa e difícil em certos contextos, especialmente em *hardware* com limitações de recursos ou em ambientes com múltiplos processadores. A dificuldade na implementação do LRU decorre de algumas razões, tais como para determinar qual página é a menos recentemente utilizada, o algoritmo LRU precisa manter um registro detalhado do histórico de acesso a todas as páginas na memória cache. Em ambientes com muitas páginas e acesso concorrente, manter esse rastreamento pode ser custoso em termos de recursos computacionais e memória, além de necessitar de hardware dedicado.

Devido a essas dificuldades, várias técnicas de aproximação do algoritmo LRU foram desenvolvidas para tornar mais viável sua implementação em certos contextos. Algumas dessas técnicas incluem algoritmo do relógio, LFU e Pseudo-LRU.