***Q1.****Faça um estudo comparativo entre barramento síncrono e barramento assíncrono.*

Barramentos síncronos necessitam pouca lógica, podendo ser facilmente implementados com FSM, mas é preciso que todos dispositivos operem na mesma frequência de clock, além disso, barramentos não podem ser tão longo se a taxa de transferência for alta.

Um barramento assíncrono não é guiado pelo clock. Dessa forma, ele pode acomodar uma diferente gama de dispositivos com diferentes velocidades, e o barramento pode ser ampliado sem o risco de ficar “fora de sincronia”. Para coordenar a transferência de dados, um barramento assíncrono utiliza um **handshaking protocol**, ou seja, são usados sinais de “requerimento” e de “pronto”. A transmissão do dado ocorre somente quando todas as partes envolvidas estiverem prontas para enviar/receber.

***Q2.****Um problema em se usar barramento é que vários dispositivos podem requerer o acesso ao mesmo num dado instante. Como o acesso ao barramento pode ser controlado?*

Não é inteligente designar essa tarefa à CPU, muitas interrupções diminuem a eficiência do processador. Podemos ter um árbitro centralizado, que decidirá que dispositivo usará o bus, inclusive um árbitro híbrido, centralizado e distribuído, onde existe prioridades. Nessa abordagem, os dispositivos são ligados em série, os mais próximos do árbitro possuem maior prioridade.

Ainda temos a arbitragem descentralizada, como no NuBus e Ethernet, no primeiro, múltiplas requisições sao feitas e os dispositivos determinam quem terá o barramento, no segundo, as requisições são independentes e colisões são detectadas e resolvidas.

***Q3.****As operações de E/S são realizadas via Sistema Operacional para garantir a integridade do sistema protegendo o mesmo de acessos indevidos. Considerando as diferentes formas da arquitetura suportar operações de E/S explique como a proteção de acessos pode ser garantida.*

O OS precisa gerir as permissões de cada programa-usuário, um programa qualquer não pode escrever no disco sem permissão concedida. Técnicas como "Read Write Execute" podem ser utilizadas para garantir que programas só leiam, escrevam e no que lhe for permitido.

O OS faz isso impedindo que o programa do usuário se comunique com os dispositivos de E/S diretamente. Existem formas de realizar esse intermédio:

* Dar comandos aos dispositivos de E/S (read, write, seek, etc)
* O dispositivo de E/S notificar o SO quando completar uma operação ou encontrar um erro.
* Os dados só poderão ser transferidos da memória para o dispositivo de E/S. Nunca se pode ler ou escrever diretamente do disco, por exemplo.

***Q4.****Compare as três maneiras de se implementar uma operação de E/S: polling, Interrupt Driven e via DMA. Em que situações cada uma destas maneiras devem ser utilizadas?   
Exemplifique.*

**Polling:** É o jeito mais simples de um dispositivo de E/S se comunicar com o processador. O dispositivo coloca a informação num registrador de status, que o processador está periodicamente checando para ver se há alguma alteração. A desvantagem disso é que o processador é muito mais rápido: ele pode checar o registrador várias vezes, sendo que o dispositivo pode não ter feito alteração alguma (como por exemplo, o mouse não ter se mexido nas últimas 1000 checagens).

O uso ideal para polling é quando as taxas de alteração dos dispositivos de E/S são **predeterminadas.** Assim, o overhead é mais previsível. É geralmente usado em sistemas de tempo real.

**Interrupt Driven:** É um esquema que faz uso de interrupções para dizer quando o dispositivo de E/S precisa da atenção do processador. Mas isso não quer dizer que ele seja síncrono, dependente de instrução, ou que pode evitar que uma instrução seja executada. A checagem se existe algum dispositivo de E/S pendente de ser tratado é feito antes de iniciar uma nova instrução. Feita a comunicação, geralmente o restante do processamento é gerenciado pelo sistema operacional. (Como por exemplo, uma tela pressionada do teclado)

Interrupt Driven Communication é usada por quase todos os sistemas em pelo menos alguns de seus dispositivos de E/S.

**DMA:** Significa Direct Memory Access, e é implementado com um controlador especial. Esse controlador assume controle do barramento (bus master), o que significa que ele pode iniciar requisições. O DMA é usado porque o overhead usando um sistema Interrupt Driven para transferir dados do disco rígido é absurdo. Por isso, o DMA é usado como um assistente - ele livra o processador de ficar constantemente checando se o dado foi transferido, realiza a transferência de forma autônoma, sem “incomodar” o processador para que ele possa continuar realizando outras tarefas, e só realiza uma interrupção novamente quando a transferência for concluída, para que o processador ou ‘pergunte’ ao DMA ou cheque na memória se a transferência foi bem sucedida.

Processadores DMA geralmente são processadores de propósito único, e são implementados para realizar essa interface entre o disco rígido e o processador, quando uma transferência se faz necessária.

***Q5.****Descreva quais os modelos de memória em processadores multi-cores citando vantagens e desvantagens de cada um.*

Num multi core, é utilizado um mesmo espaço de endereçamento de memória para todos os processadores. Eles se comunicam através de variáveis compartilhadas, e todos os processadores podem acessar qualquer lugar na memória por meio de loads e stores.

Os processadores multi-cores podem ser classificados em dois tipos, de acordo com os modelos de memória: **Memória Centralizada** (SMP ou Symmetric Multiprocessors, também chamado de UMA - Uniform Memory Access) e **Memória Distribuída** (NUMA - Non Uniform Memory Access).

Numa abordagem **UMA**, os acessos à memória principal levam o mesmo período, independente do processador que requisita o acesso e de qual word é requisitada.

Já na abordagem de **NUMA**, alguns acessos à memória são mais rápidos que outros, dependendo de que processador requisita que word.

Também existem os processadores que não fazem uso de memória compartilhada: eles possuem apenas seções de memória **privada**, e se comunicam através de troca de mensagens.

***Q6.****Quais os modelos de comunicação em processos paralelos e quais modelos de memória mais se adequam a cada modelo de comunicação?*

Temos o **message-passing,** onde os processadores trocam mensagens por meios de uma rede local. Para isso, é necessário que o sistema tenha rotinas para comunicação. O modelo de memória mais adequado nesse caso, é o de memória privada, muito utilizado em clusters.

Também existem os processadores baseados em **memória compartilhada**, que se comunicam por meio de variáveis compartilhadas usando loads e stores. Nesse caso, a abordagem mais utilizada seria a NUMA, que distribui fisicamente a memória entre os processadores. A abordagem UMA também é viável, mas pouco utilizada.

***Q7.****O que é coerência de cache?*

A cache é atualizada constantemente, é importante que a seus dados estejam coerentes com a memória principal, além disso, em sistemas multiprocessados, cada CPU pode ter uma cache própria, mas com dados compartilhados, é importantes que os dados sejam lidos e escritos corretamente. Ao atualizar seus dados, o processador deveria precisa se importar se o dado é compartilhado ou não causando várias cópias discrepantes de um mesmo dado.

Ou seja, a coerência de cache diz respeito a **consistência** **no valor** dos dados entre as versões nas caches de vários processadores.

***Q8.****Quais as técnicas para resolver o problema da coerência de cache considerando os diferentes modelos de memória?*

O problema de coerência de cache surge em caches compartilhadas por vários processadores. Uma cache privada não sofre deste problema pois somente um processador pode acessá-la.

Esse problema pode ser resolvido nas caches compartilhadas através da implementação de **protocolos** pelo controlador da cache ou pelo controlador de memória que irão cuidar de **rastrear o estado de compartilhamento dos dados.**

Dois dos protocolos mais comuns são: **snooping** ou **baseada em diretório**.

No protocolo de **snooping**, cada cache que tem uma cópia do dado também tem uma cópia do estado de compartilhamento. Todas as caches são acessíveis através de um mesmo barramento, e os controladores de cache ficam monitorando este barramento (snooping) para saber se alguma cópia que eles detém está sendo requisitada ou não por outras caches.

O protocolo de snooping usa de duas técnicas para manter a consistência dos dados: **write update** e **write invalidate.** No primeiro, após acontecer uma escrita em uma das caches, todas as caches que possuírem uma cópia desse dado atualizam seus valores. No caso do invalidate, sempre que ocorre uma escrita numa variável compartilhada, as caches que possuem uma cópia tratam de invalidá-las. Como o protocolo write update causa maior uso do barramento para manter as variáveis atualizadas, o mais usado é o write invalidate.

No protocolo **baseado em diretório**, o status de compartilhamento de um bloco da memória principal é armazenado em um local centralizado (diretório). Toda mudança feita deve ser comunicada ao diretório, que cuida de comunicar a quem for necessário (quem tiver cópias dos dados) as alterações feitas. Um único diretório mantém o estado de cada bloco na memória principal. As informações no diretório podem incluir quais caches possuem cópias do bloco, se ele é sujo etc.