1.

Explique as principais vantagens e desvantagens de E/S programada, E/S usando IRQs e E/S usando DMA. OBS: não é necessário apresentar a definição de cada técnica, mas sim explicar as vantagens e desvantagens de cada técnica em relação às outras duas.

E/S programada

Vantagens:

- É simples (em comparação a E/S usando IRQs e a E/S usando DMA)

Desvantagens:

- Segura a CPU até que a E/S não for concluída
- Tempo ocioso da CPU em dispositivos com baixa vazão de processamento.
- Espera ocupada (Busy wait)

E/S usando IRQs

Vantagens:

- O dispositivo avisa quando está pronto, isso evita o tempo de espera da CPU enquanto o dispositivo não estiver pronto, como ocorre na E/S programada
- Não possui espera ocupada, como ocorre na E/S programada

Desvantagens:

- Reduz bastante a contenção do sistema (em comparação a E/S programada)
- Ocorrência de uma interrupção para cada caractere/evento (não ocorre na E/S programada)
- O tratamento de interrupções possui um custo computacional significativo em muitos casos, pois tal tratamento aumenta o tempo de execução do processo (o tratamento das interrupções é mais eficaz na E/S usando DMA)

E/S usando DMA

Vantagens:

- A frequência de interrupções e o tratamento de tais interrupções é mais eficaz do que na E/S usando IRQs
- Ao usar o acesso direto à memória, ou seja, ao fazer o controlador de DMA alimentar uma impressora, caractere por caractere, tem-se a CPU livre para realizar outras atividades
- O DMA executa uma E/S programada sem a participação do CPU
- Reduz o número de interrupções da CPU de uma por caractere para uma por buffer impresso

Desvantagens:

- O processador tem que parar enquanto ocorre uma transferência por DMA.

2.

Liste os principais eventos (estudados na disciplina) que ocorrem, tanto em nível de HW quanto em nível de SO, ao ser gerada uma interrupção (IRQ). OBS: não é necessário apresentar a definição de IRQ, ou demais elementos que compõem a resposta. Objetivamente liste os eventos na ordem que eles ocorrem no sistema.

1. Realiza o salvamento dos registradores que não foram salvos pelo HW de interrupção.
2. Realiza a configuração do novo contexto para a rotina de tratamento da interrupção (ISR - ISR – Interrupt Service Routine).
3. Realiza a configuração da pilha para a ISR.
4. Realiza o envio do sinal para o controlador (APIC).
5. Realiza a cópia dos registradores para a PCB do processo.
6. Realiza a execução o código do tratador de interrupção (ISR).
7. Realiza a escolha de qual será o próximo processo a executar.
8. Realiza a configuração do novo contexto.
9. Realiza o carregamento dos registradores com o contexto de HW do próximo processo, incluindo o PC (program counter).
10. Realiza a execução do próximo processo.

3.

Explique o que são os níveis top half e bottom half no projeto e programação de interrupt handlers. OBS: Além de apresentar a definição de top/bottom half, objetivamente, explique para que servem cada um desses dois níveis no projeto de interrupt handlers.

Top Half (nível mais crítico, ou seja, precisa ser executado imediatamente)

> Definição: É executado pelo kernel imediatamente após a execução de HW.
> Utilidade: Realiza as tarefas que não podem esperar.

Bottom Half(nível menos crítico, ou seja, não precisa ser executado imediatamente)

> Definição: É programado pelo Top Half para ser executado assim que for possível.
> Utilidade: Realiza as tarefas que podem aguardar para serem executadas.

4.

Explique como device drivers (DD) e interrupt handlers (IH) se relacionam dentro do kernel do SO. OBS: não é necessário apresentar a definição de DD e IH. Objetivamente explique como esses dois elementos se relacionam.

DDs e interrupt handlers são programados seguindo uma abordagem orientada a eventos. Os DDs acionam o Hardware para que ele realize a operação de I/O, o Hardware aciona o interrupt handler e o interrupt handler aciona o Device Driver quando a operação de I/O for concluída.

5.
É desejável que Interrupt handlers (IH) suportem a propriedade de reentrância. Explique do que se trata essa propriedade e porque ela é desejável na implementação de IH.

Reentrância:

Essa é a propriedade de uma rotina ter suas múltiplas invocações (chamadas) executando concorrentemente de forma segura e em um mesmo contexto de execução (thread).

A reentrância é desejável na implementação de IH, pois tal propriedade permite que a segunda execução não interfira na execução da primeira execução.