## Q22 (33 livro)

A seguir algumas questões para praticar conversões de unidades:

**(a)** Quantos segundos há em um nano ano?

**(b)** Micrômetros são muitas vezes chamados de mícrons. Qual o comprimento de um mega mícron?

**(c)** Quantos bytes existem em uma memória de 1 PB?

**(d)** A massa da Terra é 6.000 yotta gramas. Quanto é isso em quilogramas?

### **Resposta:**

1. Um nano ano é × 10-9 x 365 x 24 x 3600 = 31. 536 ms.
2. 1 metro
3. Há 250 bytes, o que é 1.099.511.627.776 bytes.
4. É 6 x 1024 kg ou 6 ´ 1027 g.

**Resumo:**

1. Para calcular quantos segundos tem em um nano ano precisamos multiplicar ano, hora, minutos e segundo e dividir por 1 bilhao 10 ^9
2. Um mícron é muito pequeno 1×10 −6 metros. "Mega" significa 1×1061×10 6.
3. O PB armazena dados equivalente a 1024 TB por isso que 1 pb tem 250 bytes
4. 6.000 yottagramas é igual a 6×10276×10 27 quilogramas. Por isso que

É 6 x 1024 kg ou 6 ´ 1027 g.

## Q01 (01 livro)

**1.** Na Figura 2.2, são mostrados três estados de processos. Na teoria, com três estados, poderia haver seis transições, duas para cada. No entanto, apenas quatro transições são mostradas. Existe alguma circunstância na qual uma delas ou ambas as transições perdidas possam ocorrer?

**Resposta:**

A transição de bloqueado para executado é concebível. Suponha que um processo seja bloqueado na E/S e a E/S seja concluída. Se a CPU estiver ociosa, o processo pode ir diretamente de bloqueado para execução. A outra transição que falta, de pronta para bloqueada, é impossível. Um processo pronto não pode fazer E/S ou qualquer outra coisa que possa bloqueá-lo. Somente um processo em execução pode bloquear.

**Resumo.** O processo tem entrada e saída após ser concluído ela vai passar por uma fase bloquear e executado se a CPU está disponível, no entanto um processo também tem o estado pronto que está esperando pra ser executado ele não pode ser bloqueado.

## Q02 (02 livro)

**2.** Suponha que você fosse projetar uma arquitetura de computador avançada que realizasse chaveamento de processos em hardware, em vez de interrupções. De qual informação a CPU precisaria? Descreva como o processo de chaveamento por hardware poderia funcionar.

### **Resposta:**

Você pode ter um registro contendo um ponteiro para a entrada da tabela de processo atual. Quando a E/S for concluída, a CPU armazenará o estado atual da máquina na entrada atual da tabela de processos. Em seguida, ele iria para o vetor de interrupção para o dispositivo de interrupção e buscaria um ponteiro para outra entrada de tabela de processo (o procedimento ser-vice). Esse processo seria então iniciado.

.

**Resumo.** Para chaveamento a CPU precisaria saber do estado dos processos como por exemplo registradores e PC Quando um processo termina ou precisa ser trocado, a CPU salva o estado atual, carrega o estado do próximo processo e continua a execução a partir desse ponto. Esse ciclo se repete para todos os processos.

## Q03 (03 livro)

**3.** Em todos os computadores atuais, pelo menos parte dos tratadores de interrupções é escrita em linguagem de montagem. Por quê?

**Resumo.** Os tratadores de interrupções em computadores são frequentemente escritos em linguagem de montagem por razões de eficiência, controle de hardware, recursos limitados, necessidade de lidar com interrupções de baixo nível e para oferecer flexibilidade e personalização ao adaptar o código às necessidades específicas do sistema e do hardware.

### **Resposta**:

Geralmente, as linguagens de alto nível não permitem o tipo de acesso ao hardware da CPU que é necessário. Por exemplo, um manipulador de interrupção pode ser necessário para habilitar e desabilitar a manutenção de interrupção de um dispositivo específico ou para manipular dados dentro da área de pilha de um processo. Além disso, as rotinas de serviço de interrupção devem ser executadas o mais rápido possível.

## Q04 (04 livro)

**4.** Quando uma interrupção ou uma chamada de sistema transfere controle para o sistema operacional, geralmente uma área da pilha do núcleo separada da pilha do processo interrompido é usada. Por quê?

### **Resposta:**

Existem várias razões para usar uma pilha separada para o kernel. Dois deles são os seguintes. Primeiro, você não quer que o sistema operacional trave porque um programa de usuário mal escrito não permite espaço suficiente na pilha. Em segundo lugar, se o kernel deixar dados de pilha no espaço de memória de um programa de usuário ao retornar de uma chamada do sistema, um usuário mal-intencionado poderá usar esses dados para descobrir informações sobre outros processos.

**Resumo.** Ao lidar com interrupções ou chamadas de sistema, o sistema operacional usa uma pilha separada para manter suas informações organizadas e seguras, evitando misturar dados com os programas dos usuários. Isso ajuda na eficiência e na segurança das operações do sistema operacional.

## Q05 (05 livro)

**5.** Um sistema computacional tem espaço suficiente para conter cinco programas em sua memória principal. Esses programas estão ociosos esperando por E/S metade do tempo. Qual fração do tempo da CPU é desperdiçada?

### **Resposta**:

A chance de que todos os cinco processos estejam ociosos é de 1/32, portanto, o tempo ocioso da CPU é de 1/32.

**Resumo.** Metade do tempo da CPU é desperdiçada porque os programas estão ociosos esperando por operações de entrada/saída (E/S). Portanto, 50% do tempo da CPU não está sendo usado para realizar tarefas ativas.

## Q06 (06 livro)

**6.** Um computador tem 4 GB de RAM da qual o sistema operacional ocupa 512 MB. Os processos ocupam 256 MB cada (para simplificar) e têm as mesmas características. Se a meta é a utilização de 99% da CPU, qual é a espera de E/S máxima que pode ser tolerada?

Resposta:

Há espaço suficiente para 14 processos na memória. Se um processo tem uma E/S de p, então a probabilidade de que todos eles estejam aguardando E/S é p14. Igualando-se a 0,01, obtém-se a equação p14 = 0. 01. Resolvendo isso, obtemos p=0.72 , para que possamos tolerar processos com até 72% de espera de E/S.

**Resumo.** Para atingir 99% de utilização da CPU em um computador com 4 GB de RAM e 512 MB para o sistema operacional, os processos podem ocupar cerca de 2,27 GB de memória. Se cada processo ocupa 256 MB, cinco desses processos podem ser mantidos na memória. Portanto, a espera de E/S máxima tolerada é de aproximadamente 2,27 GB.