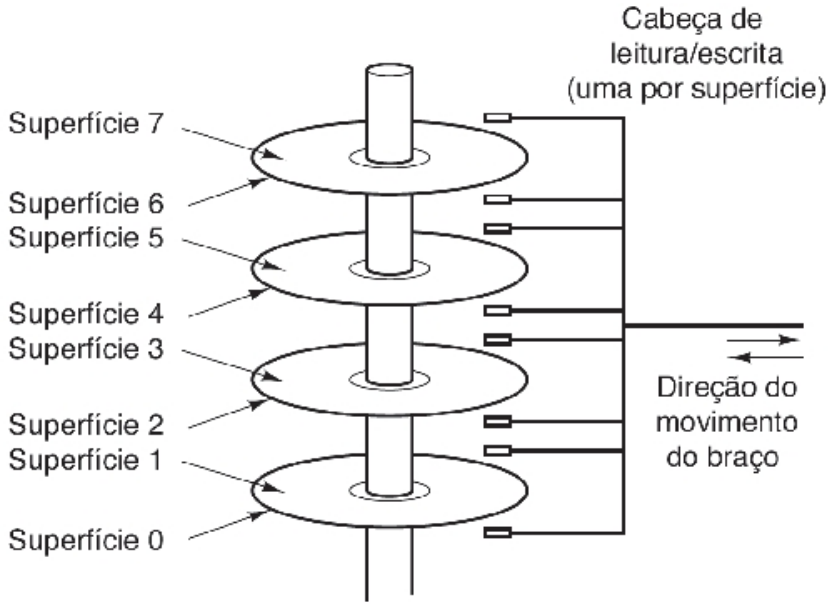
# Aula 19 – Hardware de Disco e Relógio

Norton Trevisan Roman
Clodoaldo Aparecido de Moraes Lima

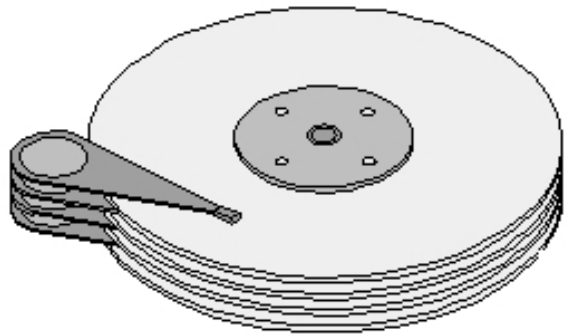
25 de novembro de 2014

# Hardware de Disco

- Consiste de:
  - Um ou mais pratos metálicos
    - Rodando a 5400, 7200 ou 10800 rpm
  - Um braço mecânico
    - Em cuja extremidade há cabeças para leitura/escrita de dados

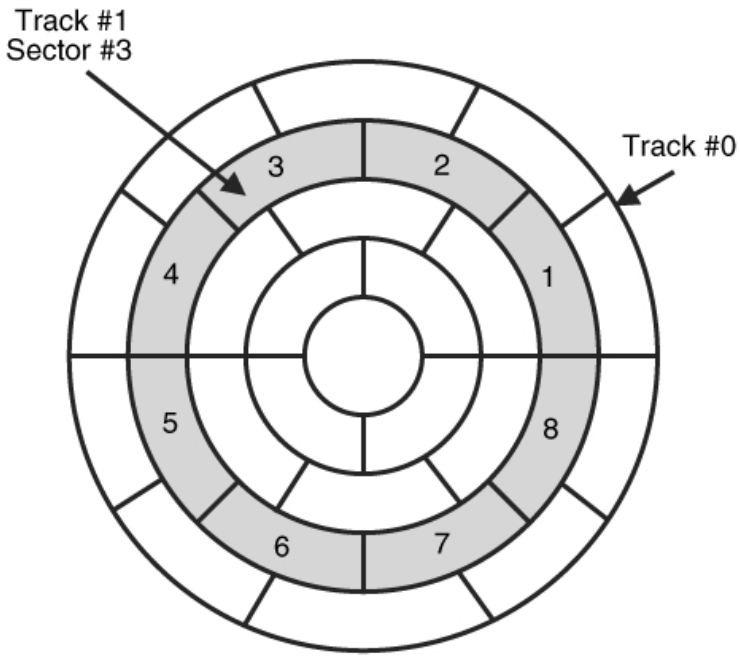


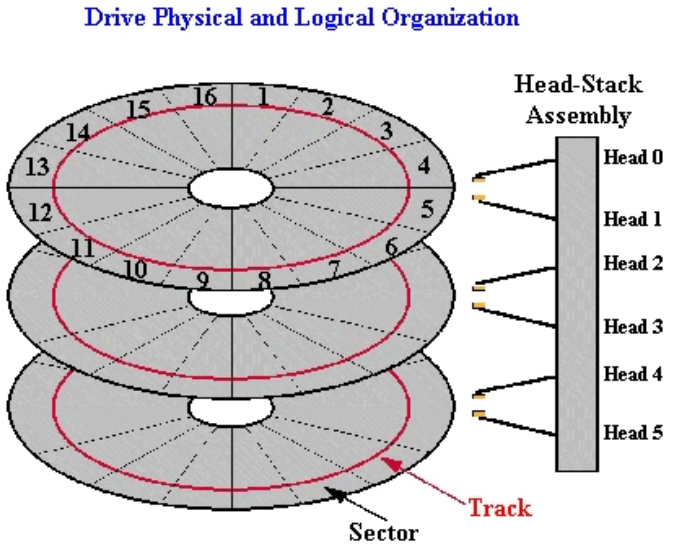

multi-platter hard disk



# Hardware de Disco

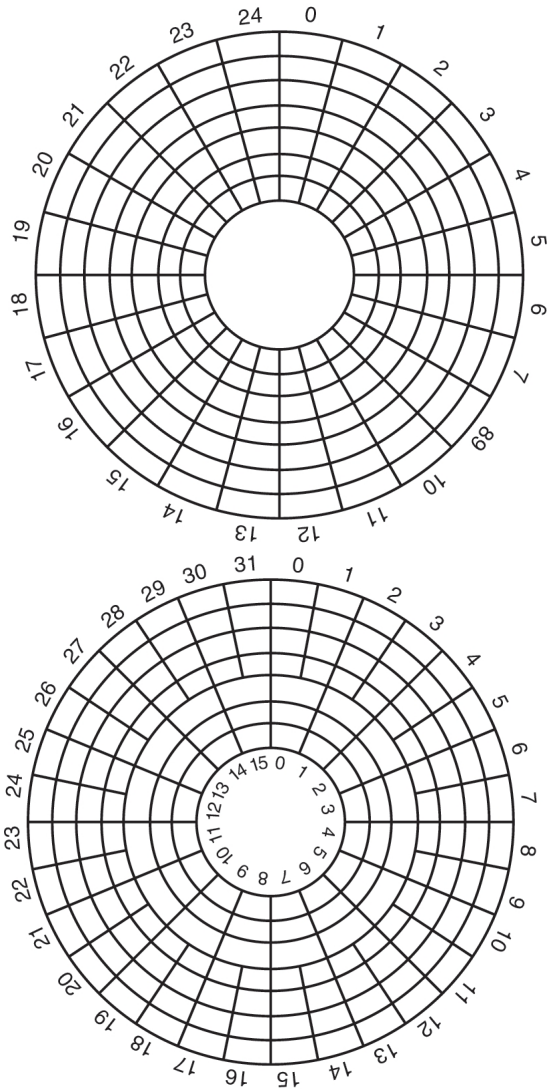
- Cada superfície de cada prato é dividida em trilhas
  - Cada trilha é dividida em setores ou blocos (512 bytes a 32K)
  - Um conjunto de trilhas (com a mesma distância do eixo central) formam um cilindro (versão 3D da trilha)



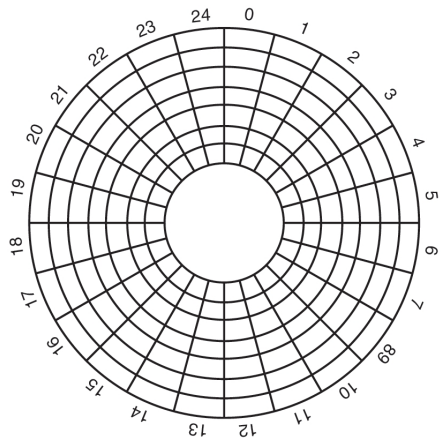
# Hardware de Disco

- Tamanho do disco:
  - nº cabeças (faces) × nº cilindros (trilhas) × nº setores × tamanho_setor
- Geometria:
  - A geometria especificada (usada pelo driver) pode diferir da real
  - Em discos antigos, o número de setores por trilha era o mesmo para todos os cilindros
  - Discos modernos são divididos em zonas
    - Mais setores nas externas que nas internas



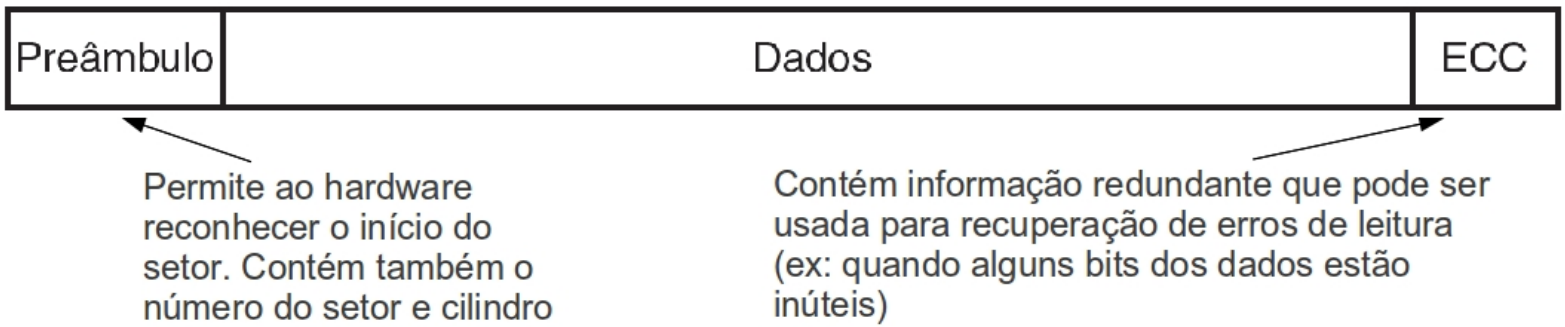
# Hardware de Disco – Geometria

- Apresentam uma geometria virtual ao SO
  - Escondem os detalhes de quantos setores há em cada trilha
  - O software age como se houvesse x cilindros, y cabeças e z setores por trilha
  - A controladora do dispositivo mapeia um pedido de (x,y,z) para o cilindro, cabeça e setor reais
  - Discos modernos possuem endereçamento lógico de bloco
    - Setores são numerados consecutivamente, iniciando no 0, sem considerar a geometria do disco



# Hardware de Disco – Formatação

- Antes que possa ser usado, recebe uma formatação de baixo nível feita por software
  - Cria série de trilhas concêntricas, contendo um certo número de setores
    - Com um pequeno intervalo entre cada setor
  - O setor é formatado da seguinte maneira:
    - Sendo o tamanho da parte de dados determinado pelo programa de formatação de baixo nível

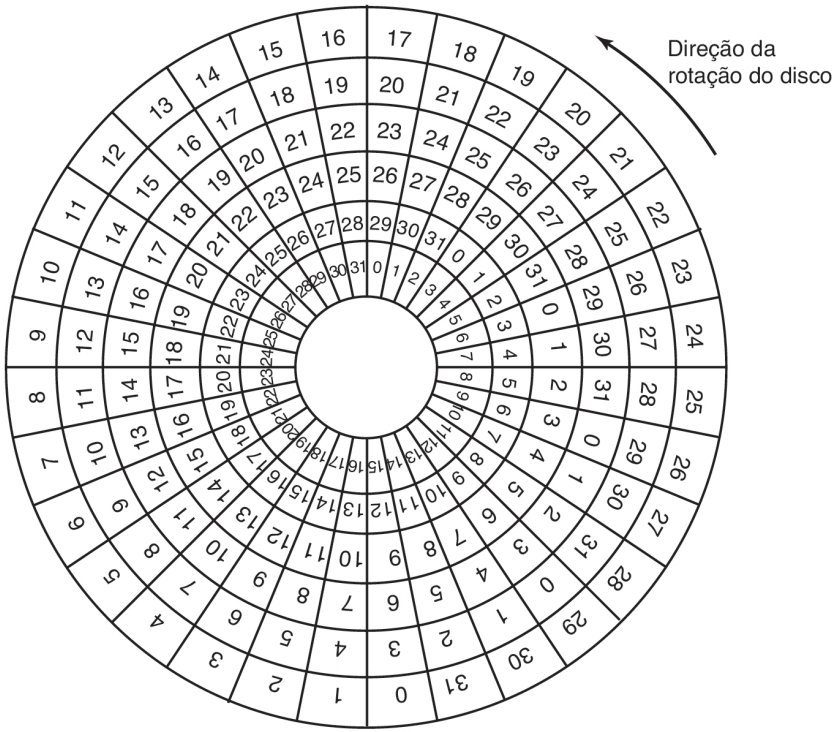


# Formatação de Baixo Nível

- Como precaução adicional, HDs têm um número de setores sobressalentes alocados
  - Usados na substituição de setores com defeito de fabricação
- Como resultado dessa formatação, a capacidade do HD é reduzida
  - Dependendo do tamanho do prâmbulo, do intervalo entre os setores e do ECC (error correction code, ou checksum), bem como do número de setores sobressalentes reservados
  - A redução pode chegar a 20%
    - Muitos fabricantes anunciam com a capacidade pré-formatação

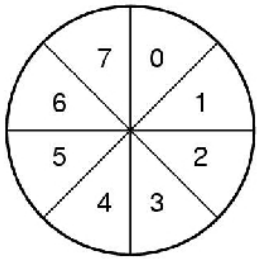
# Formatação de Baixo Nível

- A posição do setor 0 de cada trilha é deslocada em relação à trilha anterior
  - Torção cilíndrica (cylinder skew)
  - Aumenta o desempenho
    - Se o que deve ser lido for além do limite da trilha, não é preciso fazer nova busca para o setor 0 da trilha seguinte. Basta mover o braço e manter o disco rodando – quando a cabeça chegar na trilha seguinte, o setor 0 estará sob ela

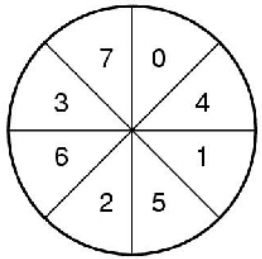


# Formatação de Baixo Nível

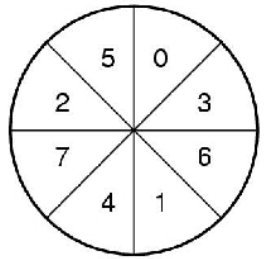
- Ao ser lido, o conteúdo do disco é transferido a um buffer na controladora
  - Quando fica cheio, o buffer é transferido à memória → toma tempo
  - Entre duas transferências do buffer à memória, pode-se passar da posição do dado no disco → deve-se esperar nova rotação
  - Solução: numerar os setores de modo entrelaçado durante a formatação
    - Dá algum tempo para que o buffer seja transferido



Sem entrelaçamento



Entrelaçamento simples



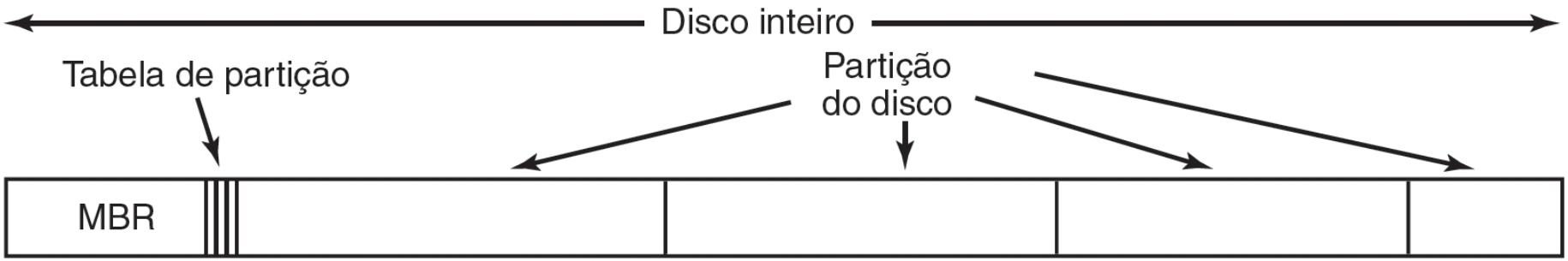
Entrelaçamento duplo

# Hardware de Disco – Particionamento

- Executado após a formatação de baixo nível
  - Do ponto de vista lógico, cada partição é tratada como um disco separado
- Setor 0 do disco contém o master boot record (MBR)
- Componentes do MBR:
  - Código (programa) de boot – primary boot loader
  - Tabela de partições (ao final do MBR), com o setor de início e o tamanho de cada partição

# Hardware de Disco – Particionamento

- Tabela de partições:
  - Normalmente, com espaço para 4 partições
  - Uma delas é marcada como ativa na tabela (para que se possa iniciar o computador a partir do HD)
  - Certos bootloaders, como o GRUB, substituem o MBR padrão com seu próprio código



# Hardware de Disco – Formatação

- Formatação de alto nível
  - Último passo, feito em cada partição separadamente
  - Define
    - Bloco de boot
    - Lista ou bitmap de blocos livres no disco
    - Diretório raiz (localização)
    - Sistema de arquivos vazio (veremos mais adiante)
  - Altera a tabela de partições
    - Dizendo o sistema de arquivos (Ext-3, NTFS etc) que é usado na partição
    - Partições podem ter sistemas de arquivos independentes

# Hardware de Disco – Boot

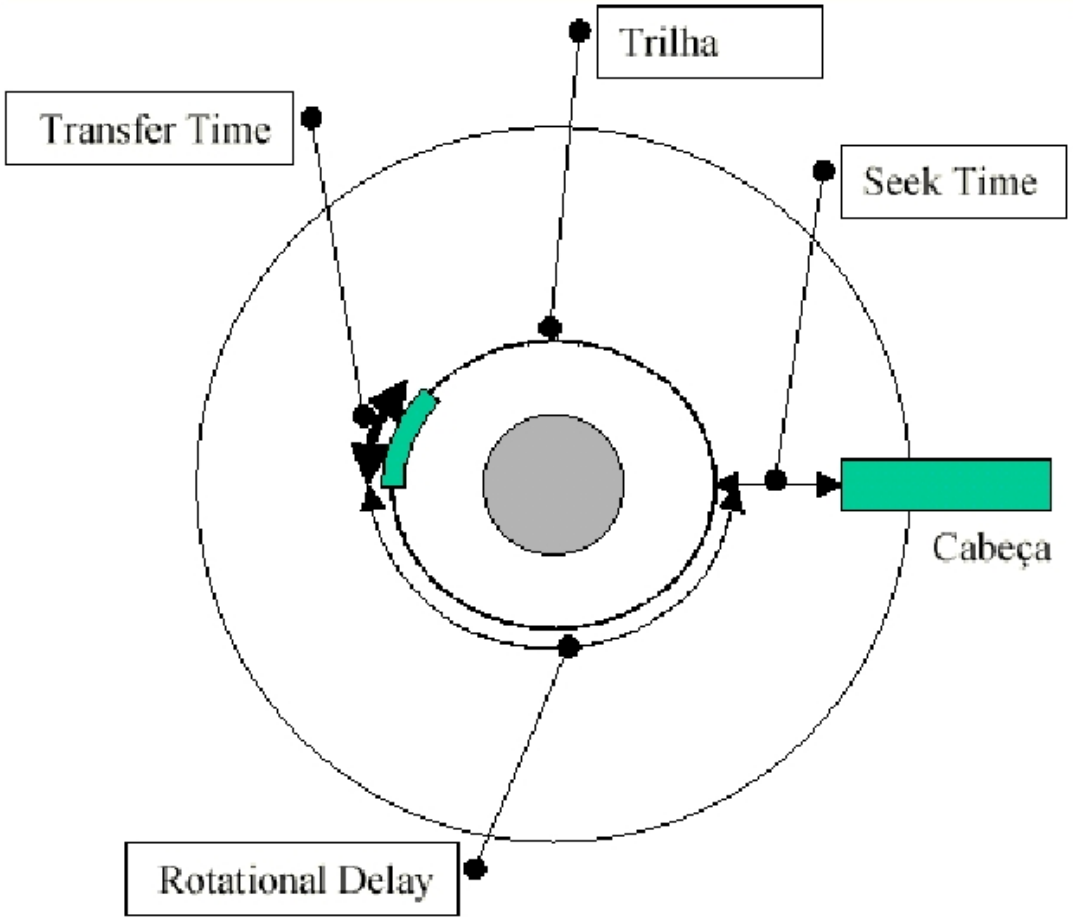
- Ao ser ligado o computador, a bios (programa) lê e executa o MBR
  - O programa no MBR localiza a partição ativa, olhando a tabela de partições
    - Aqui cabe escolha, no caso de múltiplos SO
  - Ele lê então seu primeiro bloco (bloco de boot), executando-o
    - Este bloco contém um programa – bootstrap loader (algumas vezes, o próprio kernel)
    - O bootstrap loader busca no sistema de arquivos o kernel do SO, carregando-o e executando-o
    - Por uniformidade, toda partição terá um bloco de boot em seu início, mesmo não contendo um SO que possa ser inicializado

# Hardware de Disco – Drivers de Disco

- Fatores que influenciam tempo para leitura/escrita no disco:
  - Tempo de posicionamento (seek) $\rightarrow$ tempo para mover o braço para o cilindro correto
  - Atraso rotacional (latência) $\rightarrow$ tempo necessário para rotar o setor correto sob o cabeçote
  - Tempo de transferência real dos dados
  - $T_{acesso} = T_{posicionamento} + T_{latência} + T_{transferência}$

# Hardware de Disco – Drivers de Disco

$$t_{access} = t_{seek} + t_{rotationaldelay} + t_{transfer}$$



# Hardware de Disco – Drivers de Disco

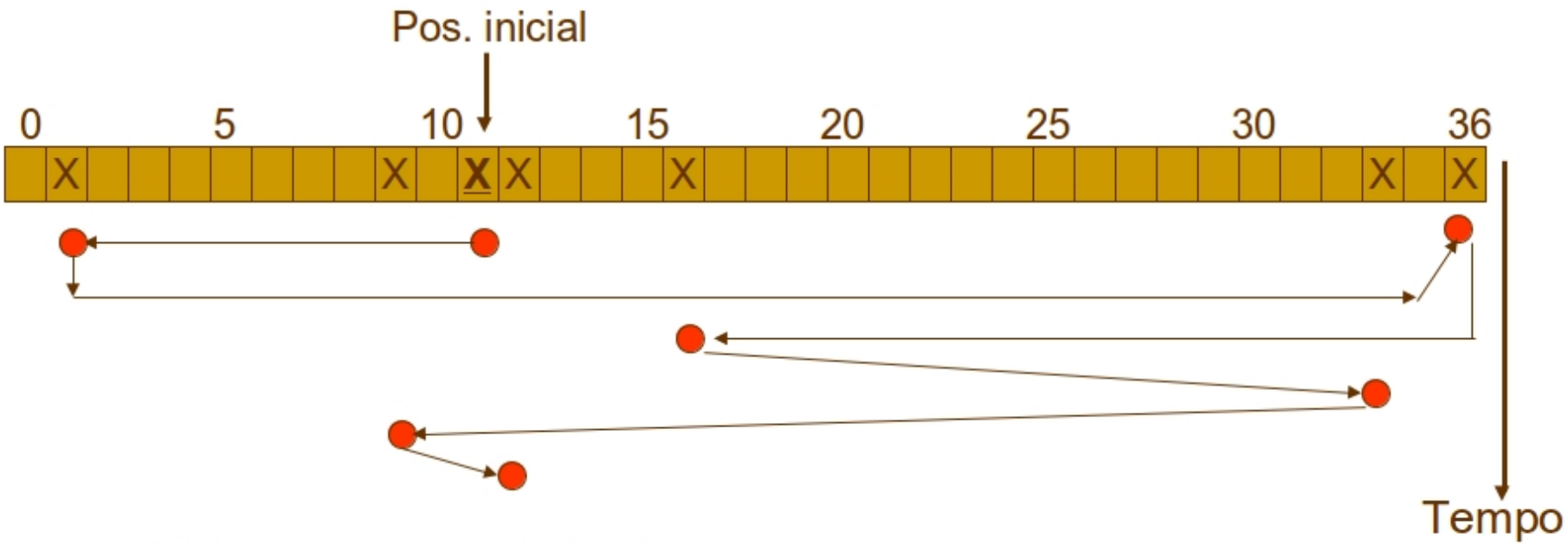
- Para muitos discos, o tempo de posicionamento domina
  - Bom lugar para reduções
- Quando o disco está muito carregado, é provável que, durante uma busca, outras requisições sejam geradas por outros processos
  - O driver mantém uma tabela de requisições pendentes, indexada pelo número do cilindro
    - Com todas as requisições pendentes em uma lista ligada
    - Cada entrada da tabela tem a lista de requisições para seu cilindro correspondente

# Escalonamento do Braço – Algoritmos

- Executado pelo Driver
- First-Come, First-Served (FCFS)
  - O driver aceita uma requisição por vez, e as executa nessa ordem
  - Pouco pode ser feito para otimização
  - Ex:
    - Disco com 37 cilindros
    - Atualmente lendo bloco no cilindro 11
    - Surgem requisições para os cilindros 1,36,16,34,9,12, nesta ordem

# Escalonamento do Braço – Algoritmos

Disco com 37 cilindros;
Lendo bloco no cilindro 11;
Requisições: 1,36,16,34,9,12, nesta ordem

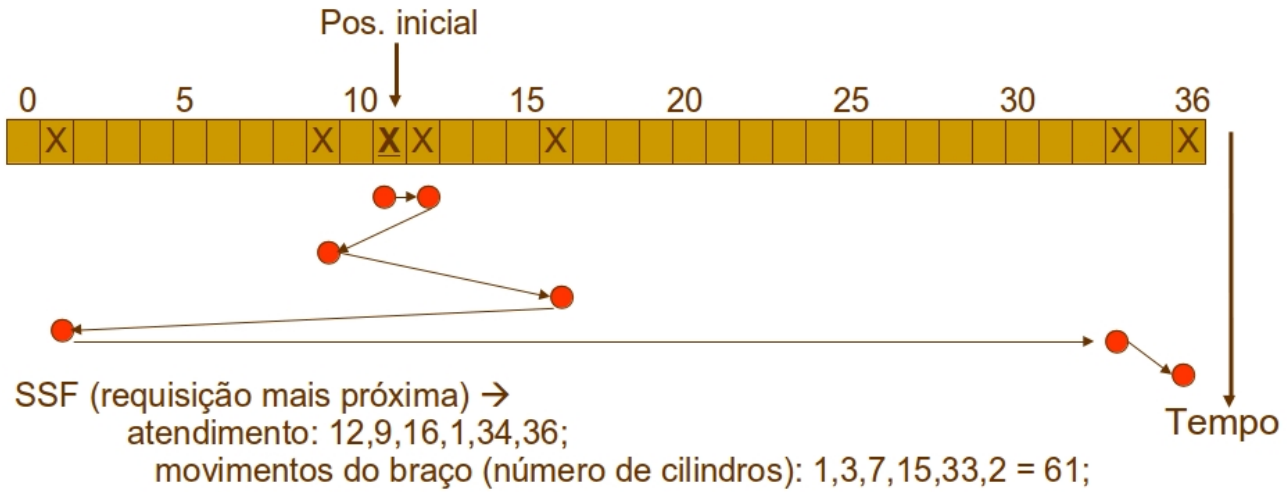



FCFS → atendimento: 1,36,16,34,9,12;
movimentos do braço (número de cilindros): 10,35,20,18,25,3 = 111;

# Escalonamento do Braço – Algoritmos

- Shortest Seek First (SSF)
  - Usando a tabela do driver, sempre atenda a requisição mais próxima da posição atual da cabeça de leitura/gravação
  - Minimiza o tempo de posicionamento

Disco com 37 cilindros;
Lendo bloco no cilindro 11;
Requisições: 1,36,16,34,9,12, nesta ordem




SSF (requisição mais próxima) →
atendimento: 12,9,16,1,34,36;
movimentos do braço (número de cilindros): 1,3,7,15,33,2 = 61;

# Escalonamento do Braço – Algoritmos

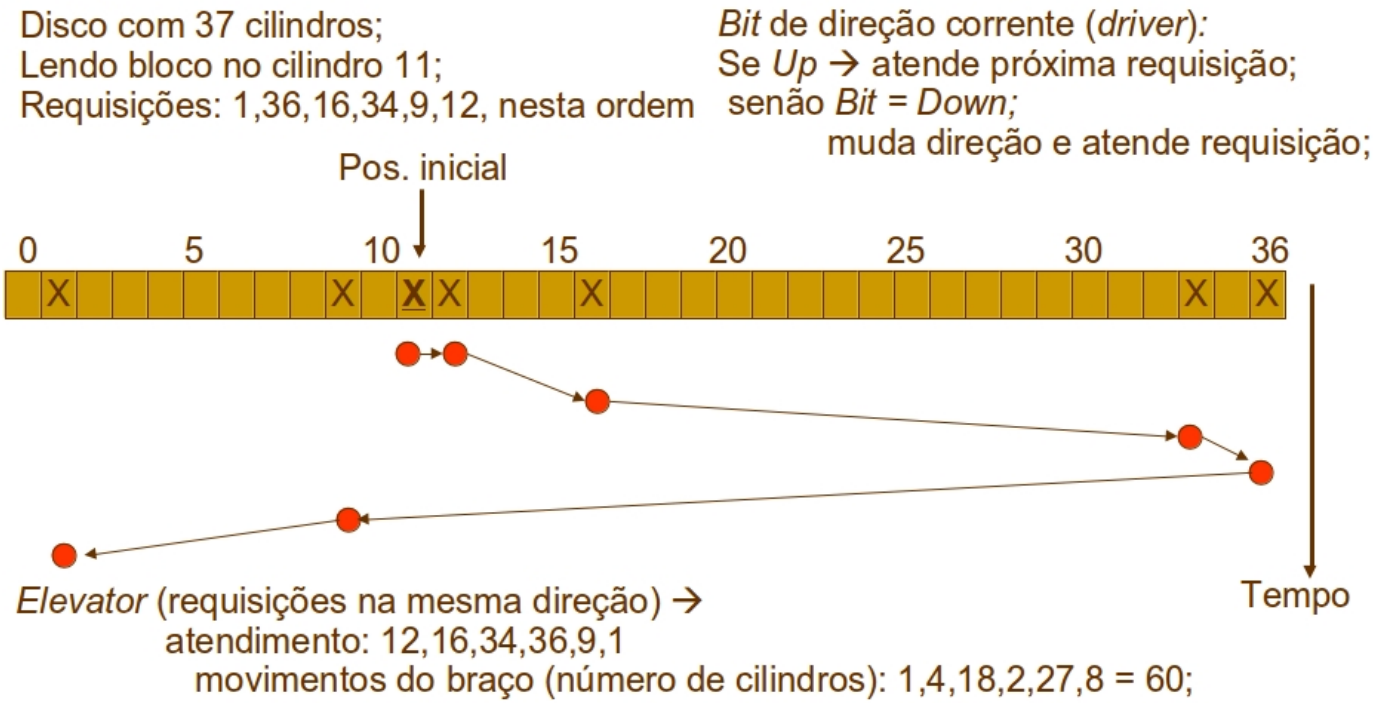
- Shortest Seek First (SSF) – Problemas
  - Se mais requisições forem chegando, a cabeça tenderá a não se mover muito de sua posição original
  - Se o disco estiver carregado, tenderá a ficar no meio a maior parte do tempo
  - Requisições nos extremos do disco demorarão a ser atendidas

# Escalonamento do Braço – Algoritmos

- Elevador
  - O problema de escalonar os andares de um elevador, em um edifício alto, é semelhante ao braço do disco
    - Requisições chegam continuamente e aleatoriamente
  - Muitos elevadores tentam conciliar eficiência e justiça
    - Continuam se movendo na mesma direção até não haver mais requisições pendentes naquela direção
    - Então trocam de direção
  - No caso do disco, o driver deve manter 1 bit para a direção (up ou down)
    - Quando uma requisição termina, o driver verifica o bit
    - Se for up, o braço é movido à próxima requisição mais alta

# Escalonamento do Braço – Algoritmos

- Elevador
  - Se não houver requisições pendentes nessa direção, o bit é feito down, e o braço se move à próxima requisição mais baixa

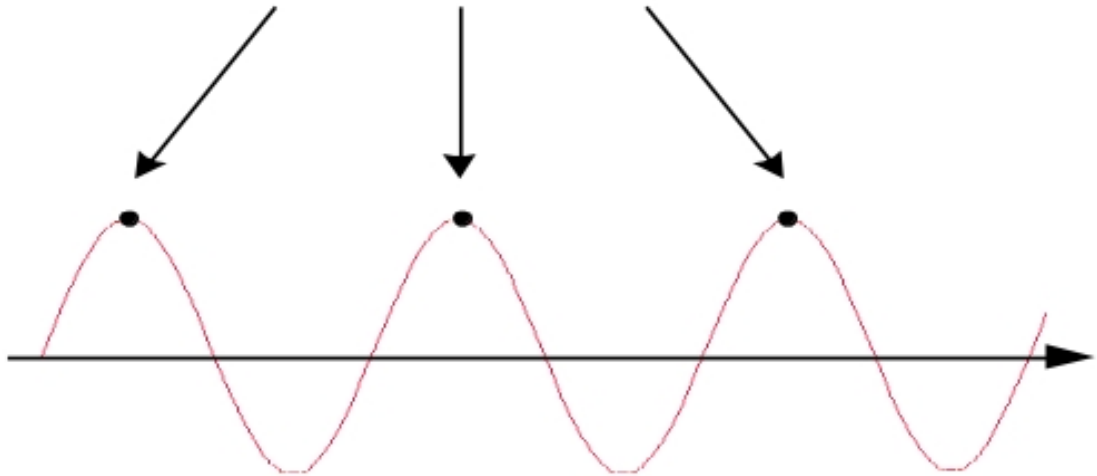


# Clocks (Timers)

- Componentes do relógio:
  - Hardware (clock hardware) e software (clock driver)
- Hardware:
  - Dispositivo que gera pulsos síncronos
  - Localizados na CPU ou na placa-mãe
  - Sinal utilizado para a execução de instruções
  - Presente em qualquer sistema multiprogramado
    - Fundamental para ambientes TimeSharing
    - Responsável pela sincronização dos vários circuitos do computador

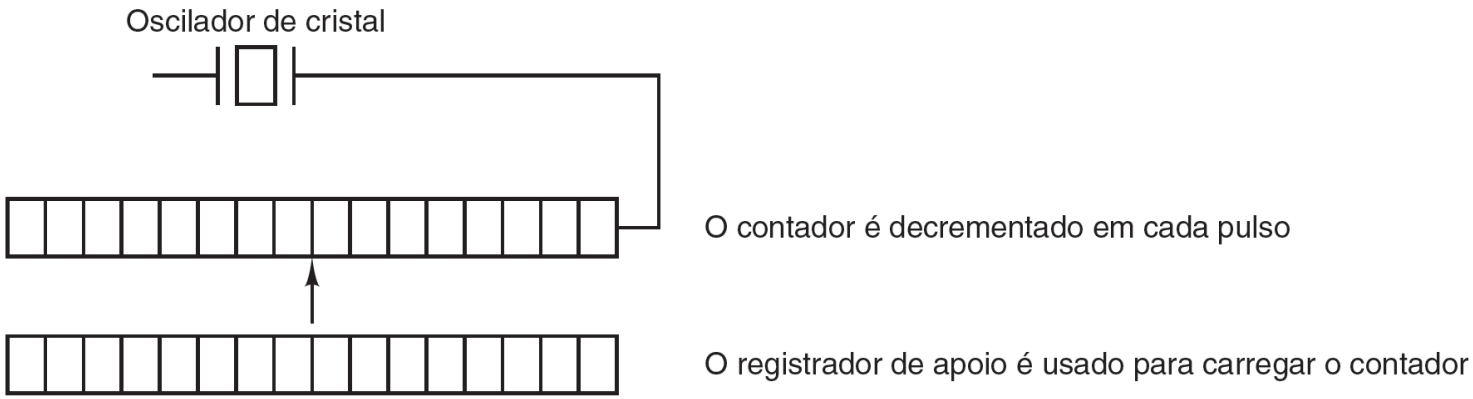
# Clocks (Timers) – Hardware

- Frequência de clock
  - Número de vezes que o pulso se repete por segundo (Hz)
- Dois tipos:
  - Básico: usa o sinal da rede elétrica (110/220 V) para fazer contagem (50/60 Hz) $\rightarrow$ cada oscilação da rede é uma interrupção
    - Raros hoje



# Clocks (Timers) – Hardware

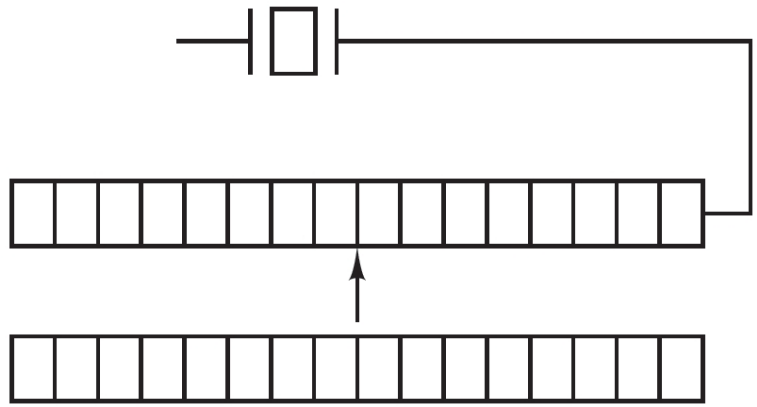
- Dois tipos:
  - Com 3 componentes (programável, de alta precisão):
    - Oscilador de cristal (Piezzoelétrico): cristal propriamente cortado e montado sob tensão elétrica – gera sinal periódico
    - Contador
    - Registrador de apoio




Quando o contador chegar a zero, uma interrupção é gerada

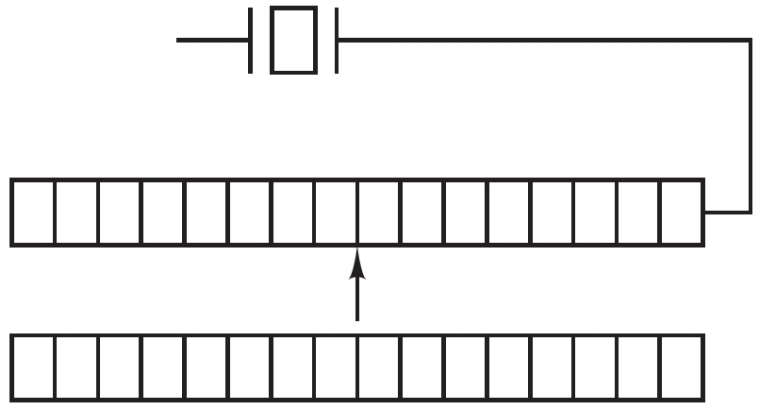
# Hardware – Clocks Programáveis

- Podem operar de dois modos básicos:
  - One-shot mode (disparo único)
  - Square-wave mode (onda quadrada)
- One-shot mode:
  - Quando o clock é iniciado, ele copia o valor do registrador de apoio no contador
  - A cada pulso do cristal, o contador é decrementado



# Hardware – Clocks Programáveis

- One-shot mode:
  - Quando o contador zera, ele gera uma interrupção
    - O clock pára até que seja explicitamente reiniciado pelo software



- Square-wave mode:
  - Após chegar a zero e causar a interrupção, o registrador de apoio é automaticamente copiado para o contador
  - O processo todo é repetido novamente

# Hardware – Clocks Programáveis

- Essas interrupções periódicas são chamadas de pulsos (ou tiques) de relógio (clock ticks)
  - Chips de clocks programáveis possuem, em geral, 2 ou mais relógios independentes
- Vantagem:
  - A freqüência das interrupções pode ser controlada por software (via registrador)
- Muitos computadores possuem clock de segurança, com bateria (lido assim que o computador é ligado)
  - Evita a perda do horário atual quando o computador é desligado

# Clocks (Timers) – Software

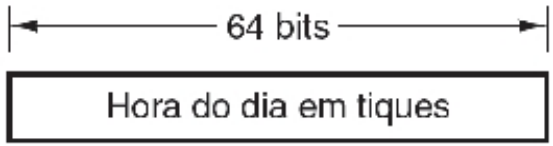
- O hardware apenas gera interrupções em intervalos conhecidos
  - Todo o resto depende do driver do clock
- Funções do clock driver:
  - Manter a hora do dia
  - Evitar que processos executem por mais tempo que o permitido
  - Contabilizar o uso da CPU
  - Tratar a chamada de sistema alarm (feita pelos processos do usuário)
  - Fornecer temporizadores "guardiões" para o sistema
  - Fazer monitoramento e coletar estatísticas

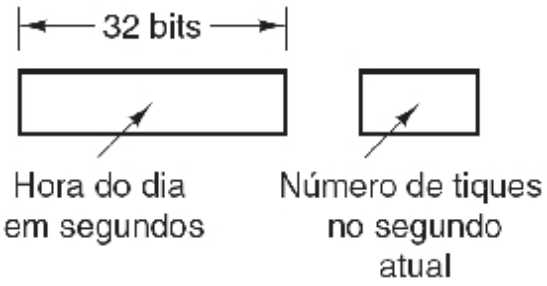
# Clocks (Timers) – Software

- Manter a hora do dia:
  - Também chamada de tempo real
  - Hora e data correntes:
    - Checa a CMOS – usa baterias para não perder as informações
    - Pergunta ao usuário
    - Checa pela rede em algum host remoto
  - Traduzida para o número de clock ticks:
    - Desde as 12 horas de 1º de janeiro de 1970 (UTC) no UNIX
    - Desde 1º de janeiro de 1980 no Windows
    - De fato, contam segundos, não tiques (mais adiante...)

# Clocks (Timers) – Software

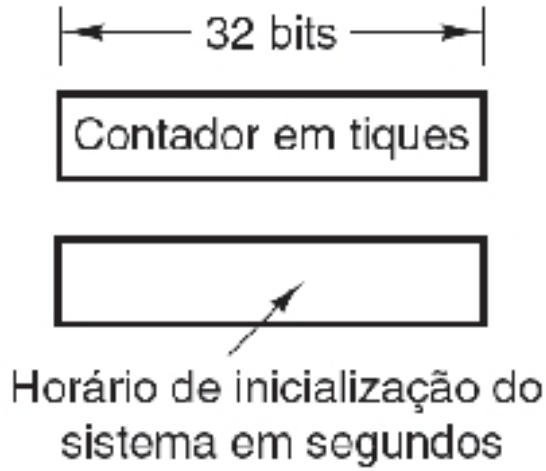
- Manter a hora do dia:
  - Basta incrementar o contador a cada tick
    - Problema: a 60Hz, um contador de 32 bits chegará em seu limite em pouco mais de 2 anos
  - Solução: três abordagens:
  - *Contador com 64 bits*
    - Maior custo de incrementar
  - *Contar o tempo em segundos*, em vez de tiques
    - Usar contador auxiliar para contar os tiques até dar 1s
    - $2^{32}$ s $>$ 136 anos



# Clocks (Timers) – Software

- Manter a hora do dia:
  - *Contar tiques relativos à hora em que o sistema foi iniciado*
    - Armazena o horário de inicialização, a partir do relógio de segurança (backup clock), na memória
    - Toda vez que o sistema precisar, usa esse valor + o conteúdo do contador para calcular a hora



# Clocks (Timers) – Software

- Controlar a duração da execução de processos
  - Quando um processo inicia, o escalonador inicializa um contador com o valor do quantum em tiques de clock
    - A cada interrupção do clock, o driver do clock decrementa esse contador em 1
    - Quando chega a 0, o driver do clock chama o escalonador (para decidir se troca ou não o processo)
- Contabilizar o uso da CPU
  - Quanto tempo o processo já foi executado?
    - Processo inicia $\rightarrow$ inicia um segundo clock
    - Processo é parado $\rightarrow$ esse clock é lido (diz o quanto rodou)
    - Durante interrupções, o valor desse segundo clock é salvo e restaurado depois

# Clocks (Timers) – Software

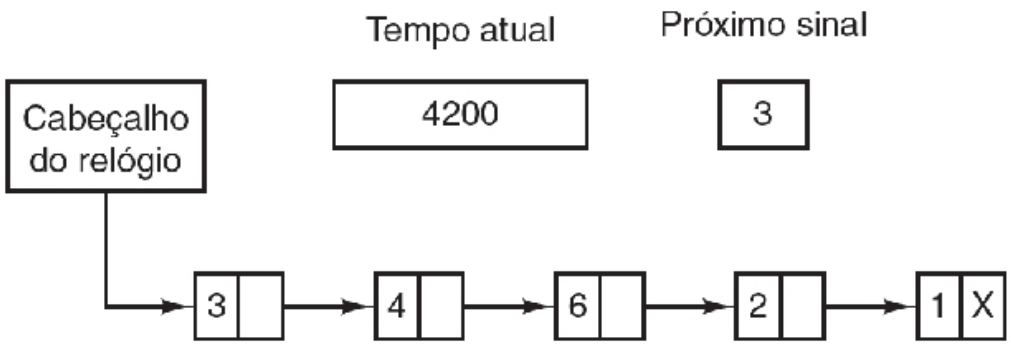
- Contabilizar o uso da CPU
  - Quanto tempo o processo já foi executado?
    - Alternativamente, pode-se manter um ponteiro para a entrada na tabela de processos do processo em execução
    - A cada tique do relógio, um campo nessa entrada é incrementado
    - Abordagem menos precisa: se muitas interrupções ocorrerem durante a execução de um processo, ainda assim será contado um tique completo
- Alarmes
  - Em alguns sistemas, processos podem solicitar "avisos" após um certo intervalo
    - Ex: rede, caso em que pacotes não confirmados podem ter que ser reenviados

# Clocks (Timers) – Software

- Alarmes
  - Avisos podem ser: um sinal, uma interrupção ou uma mensagem
  - Se o driver gerenciar clocks suficientes, basta determinar um clock separado para cada requisição
  - Se não tiver, terá que simular clocks virtuais múltiplos com um único clock físico:
    - Manter uma lista encadeada com os tempos dos alarmes pendentes, ordenada pelo tempo
    - Cada item na lista diz quantos tiques do clock, após o tique anterior, deve-se esperar antes de se enviar o sinal

# Clocks (Timers) – Software

- Alarmes
  - Ex: Sinais esperados em 4203, 4207, 4213, 4215 e 4216




A cada tique, "Próximo sinal" é decrementado (assim como sua entrada na lista).

Quando chega a 0, o sinal correspondendo ao primeiro item da lista é emitido.

Este item é então removido da lista

"Próximo sinal" recebe o valor do começo da lista $\rightarrow$ 4

# Clocks (Timers) – Software

- Temporizadores guardiões (watchdog timer):
  - Partes do SO também precisam de temporizadores
  - Ex: acionador de disco: somente quando o disco está em rotação na velocidade ideal é que as operações de E/S podem ser iniciadas
    - Ao receber uma requisição, o driver do dispositivo inicia o motor, e então define um temporizador guardião para causar uma interrupção após um certo intervalo
  - O mecanismo usado pelo driver de relógio para tratar desse tipo de temporizador é o mesmo usado em alarmes
    - Quando um temporizador dispara, contudo, em vez de causar um sinal, o driver chama um procedimento fornecido pelo requisitante

# Clocks (Timers) – Software

- Tarefas básicas do driver de relógio (clock driver) durante uma interrupção de relógio:
  - Incrementar o tempo real
  - Decrementar o quantum e comparar com 0 (zero)
  - Contabilizar o uso da CPU
  - Decrementar o contador do alarme
  - Gerenciar o tempo de acionamento de dispositivos de E/S (via temporizadores guardiões)

# Referências Adicionais

- `http://www.dedoimedo.com/computers/grub.html`