Interrupções e E/S

26 de novembro de 2018

Roteiro

- Armazenamento em disco
- Interrupções e trocas de contexto
- Entrada e saída (E/S)
- Processo de inicialização
- Parte prática

Armazenamento em disco

Armazenamento em disco

Características de dispositivos de armazenamento:

- Confiabilidade: os dados continuam armazenados?
- Capacidade e possibilidade de expansão
- Taxa de transferência e tempo de resposta
- Custo, peso, etc.

Armazenamento em disco

Disco rígido (HD, HDD, hard drive):

Figura: Disco rígido por Evan-Amos

- Dispositivo **mecânico** de armazenamento
- Armazena o sistema de arquivos (arquivos e diretórios)
- Dispositivo barato p/ armazenamento permanente
- Muito lento quando comparado à RAM

Armazenamento em disco

L1 cache reference	0.5	ns			
L2 cache reference	7	ns			14x L1 cache
Main memory reference	100	ns			20x L2 cache, 200x L1 cache
Read 4K randomly from SSD	150,000	ns	150 us		~1GB/sec SSD
Read 1 MB sequentially from memory	250,000	ns	250 us		
Read 1 MB sequentially from SSD	1,000,000	ns	1,000 us	1 ms	~1GB/sec SSD, 4X memory
Disk seek	10,000,000	ns	10,000 us	10 ms	20x datacenter roundtrip
Read 1 MB sequentially from disk	20,000,000	ns	20,000 us	20 ms	80x memory, 20X SSD

Fonte: Latency Numbers Every Programmer Should Know - https://gist.github.com/jboner/2841832

Anatomia do disco

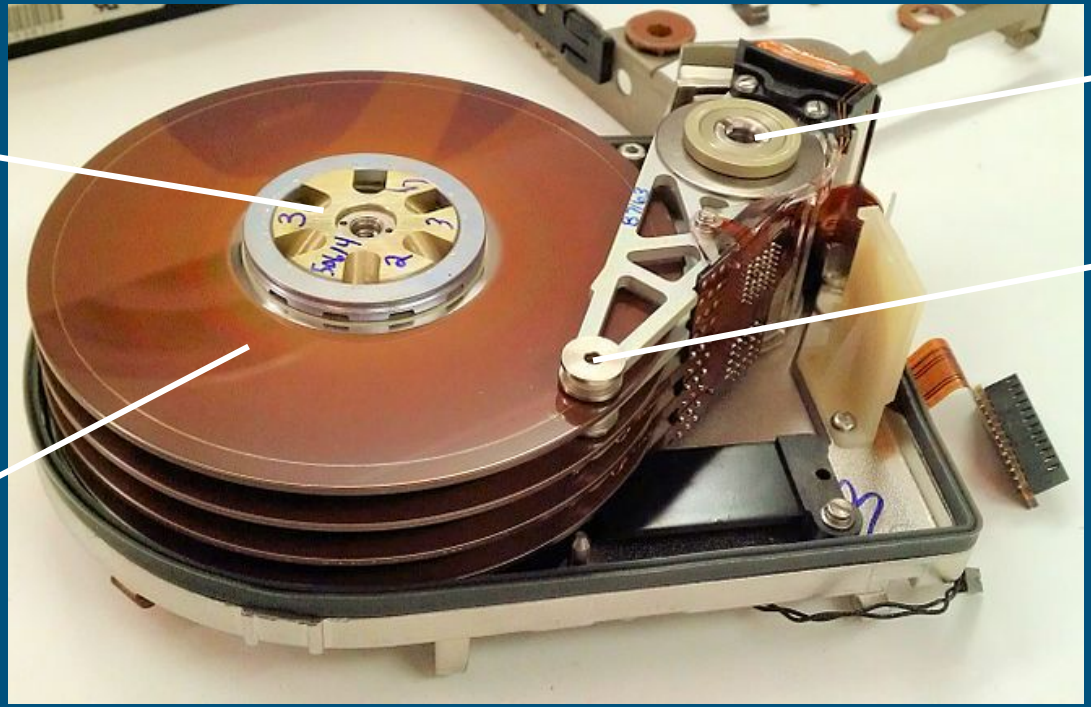
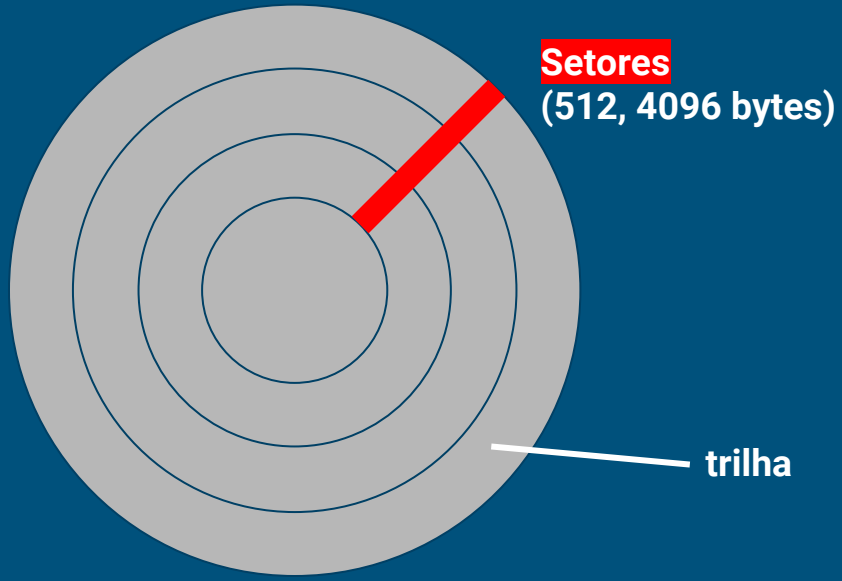
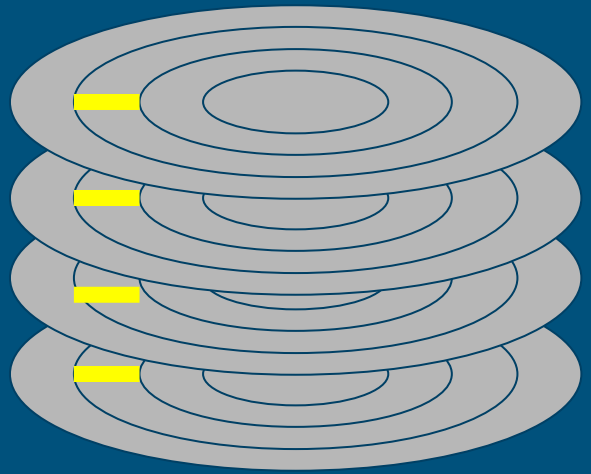


Figura: IBM 0665-30 HDD 1 por Vanderdecken

Organização lógica

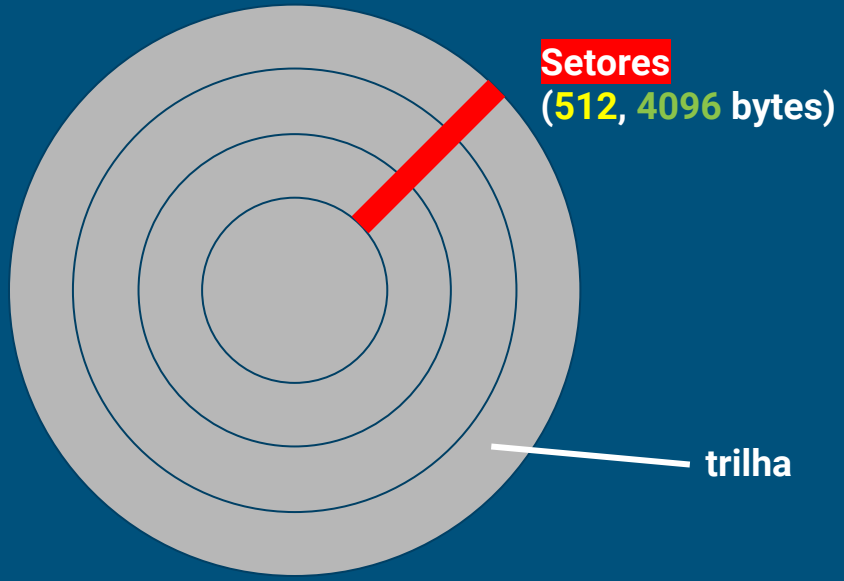


Cilindro (setores de todas as faces)



Por que é relevante?

Capacidade de armazenamento



Qual é a capacidade do HD?

- **H:** nº de faces (heads)
- **T:** qtd de trilhas/face
- **S:** qtd de setores/trilha

Cap (bytes) = **H** x **T** x **S** x **512**

4096

Modos de endereçamento

Como acessar um dado no disco?

- Método CHS
  - Modo de endereçamento usado antigamente
  - Dados são endereçados pela tripla (Cylinder, Head, Sector)
  - Ex: (0, 0, 1), (31, 7, 63), etc.

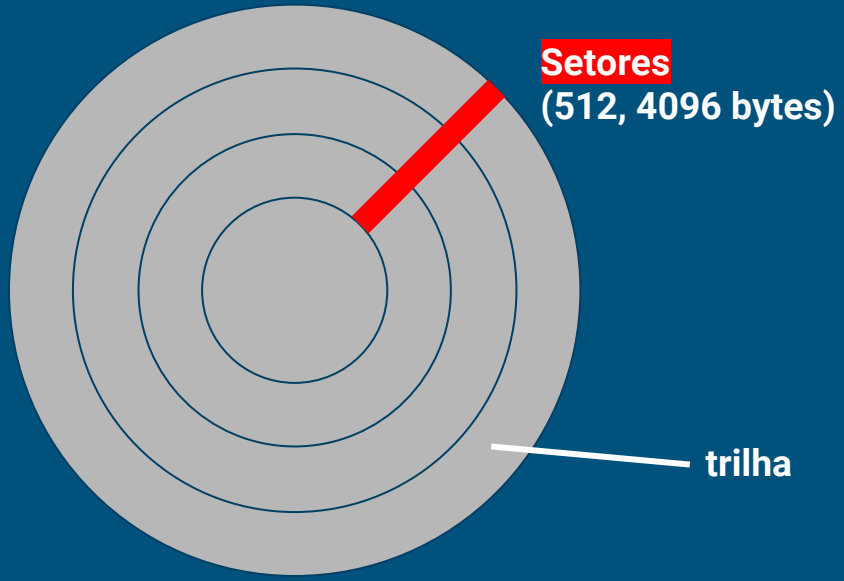
- Método LBA
  - Linear Block Addressing: usado atualmente
  - Disco dividido em blocos lógicos numerados em sequência
  - Ex: 0, 1, 2, 192, 32768, 33554432, etc.

Setores do disco

Cada setor armazena:

- ID do setor, sync field (p/ controlador)
- Dados do usuário (512 bytes, 4096 bytes)
- Código de detecção de erros
- *Gap*

Desempenho



Medidas de desempenho:

- **Seek time:** tempo p/ posicionar o cabeçote na trilha
- **Latência rotacional:** quanto demora p/ atingir o início do setor
- **Tempo de transferência:** tempo p/ leitura/escrita dos dados
- **Taxa de transferência:** medida em MB/s

Desempenho

Requisições aos dados do HD são colocadas em um *buffer* do controlador:

- Políticas de escalonamento
  - Aumentar taxa de transferência
  - Reduzir tempo médio de resposta

- Exemplos
  - FIFO: First in, first out
  - SSTF: Shortest Seek Time First
  - Método do elevador: movimento das cabeças usando direção preferencial

Solid State Drive (SSD)

- Mais rápido que o disco
  - Baseado em circuitos digitais
  - Não é um dispositivo mecânico
- Mais robusto que o disco
  - Menos partes móveis, menos chance de quebrar
  - **Gasta menos energia**, é menor
- Custa mais caro por GB
  - HD 500 GB: R$ 120 (atualmente)
  - SSD 480 GB: R$ 420 (x 3.5)

Interrupções

Interrupções

O que é uma **interrupção**?

- Uma *interrupção* na atividade normal do processador
- Um sinal emitido ao processador comunicando que algo precisa de atenção
- Processador deve parar o que está fazendo e tratar a interrupção
- Causam um desvio no fluxo normal da CPU (necessita *trocar de contexto*)

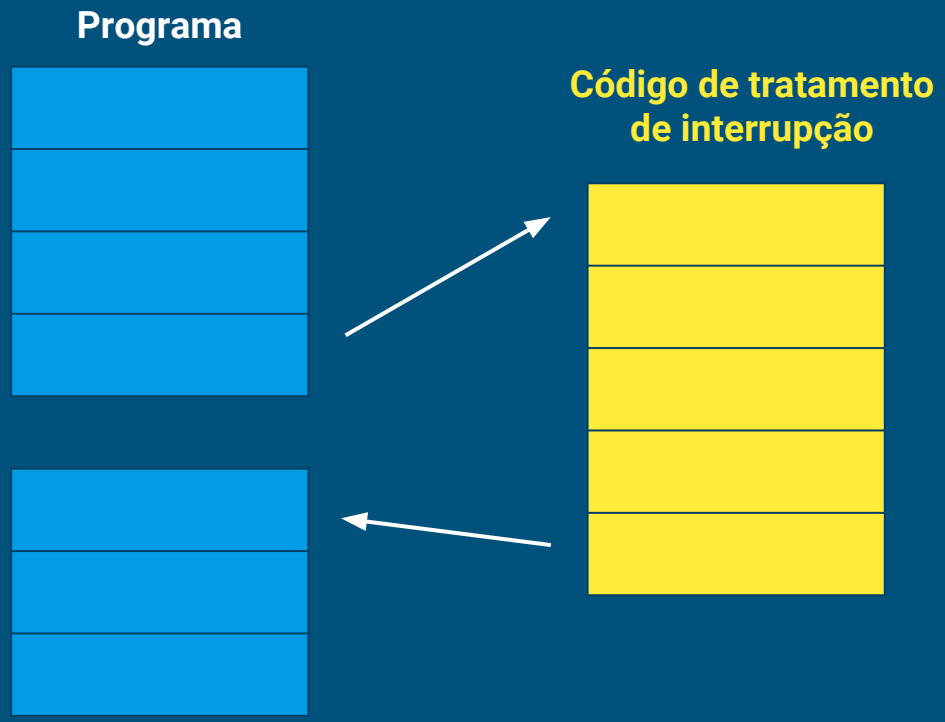
Interrupções

- Interrupções de software
  - Ex: divisão por zero, system calls p/ requisições E/S, etc.
- Interrupções de hardware
  - Ex: disco, teclado, placa de rede, clock, etc.
  - *Drivers* dos fabricantes - tratamento de interrupções
- Ex: CPU recebe interrupção do mouse
  - CPU deve parar o que está fazendo: **troca de contexto**
  - CPU deve tratar a interrupção: ler posição e armazenar na memória
  - CPU deve retomar o que fazia antes: retornar ao programa

Interrupções

Troca de contexto:

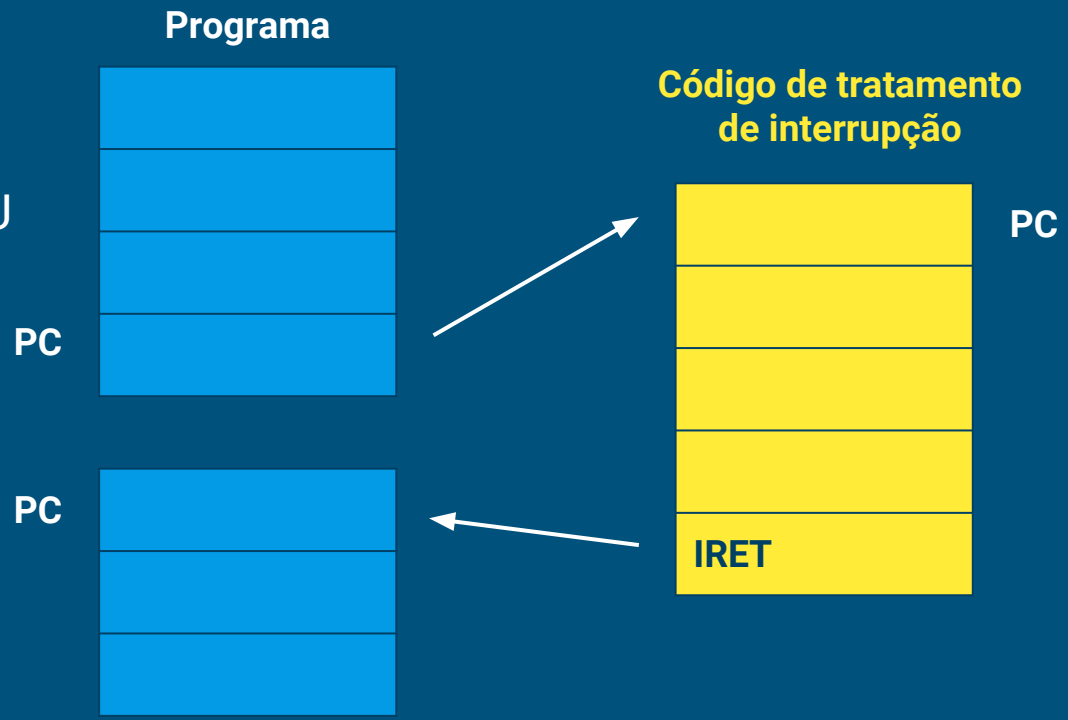
- O processador pausa a execução do **programa**
- **Interrupção** é tratada
- A execução do **programa** é resumida



Interrupções

Troca de contexto:

1. Salva o estado atual da CPU
2. Identifica o tipo de interrupção e localiza o código
3. **Trata a interrupção**
4. Restaura o estado anterior da CPU
5. Retorna ao programa



Interrupções

Onde ficam as rotinas de tratamento das interrupções?

Vetor de interrupções: associa um código de interrupção a um endereço.

Exemplo de implementação:

1. **Jump table:** cada entrada do vetor é uma instrução *jump* para a rotina de tratamento da interrupção.

2. Para tratar a interrupção de código k (k = 0, 1, ..., 255), salva-se o estado da CPU e, em seguida, pula-se para a posição k da jump table.

Interrupções

Suponha:

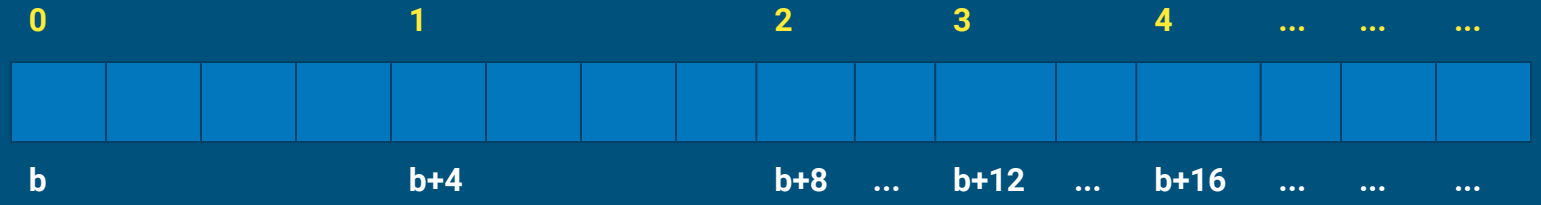
- Endereço base do vetor de interrupções na RAM: *b*
- Cada entrada do vetor de interrupções ocupa: *s* bytes
- Vetor de interrupções tem comprimento 256 (0, 1, ..., 255)

Então:

- Modifico o Program Counter (PC) p/ o endereço da posição k do vetor: $addr_k$ = ???

Interrupções

Qual o endereço $addr_k$ da posição k do vetor de interrupções? Sup. s = 4 bytes



$addr_0 = b$

$addr_1 = b+4$

$addr_2 = b+8$

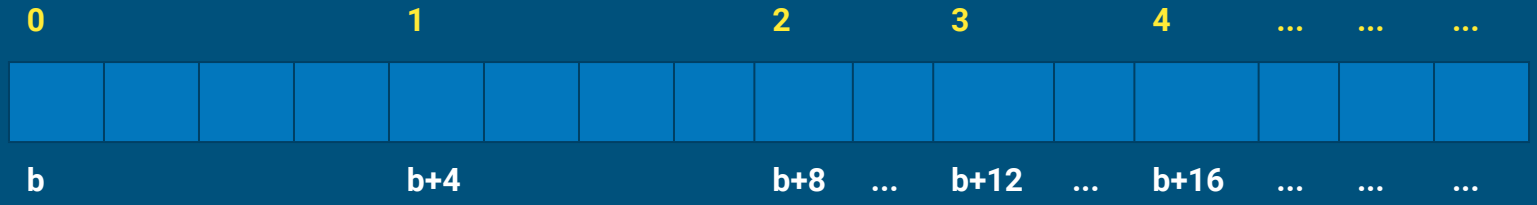
$addr_3 = b+12$

$addr_4 = b+16$

$addr_k = ???$

Interrupções

Qual o endereço $addr_k$ da posição k do vetor de interrupções? Sup. s = 4 bytes



$addr_0 = b$

$addr_1 = b+4$

$addr_2 = b+8$

$addr_3 = b+12$

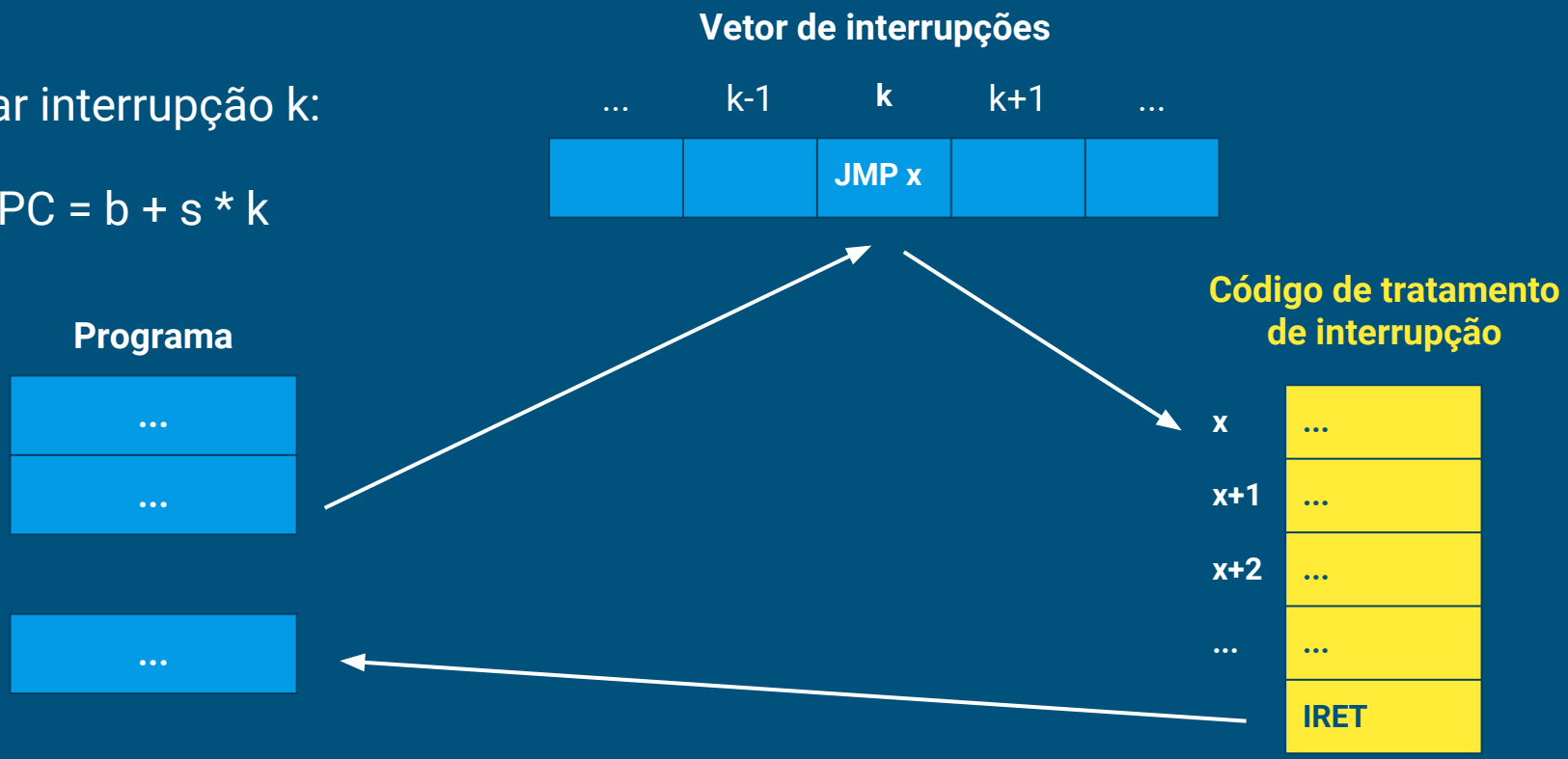
$addr_4 = b+16$

$addr_k = b + s * k$

Interrupções

Tratar interrupção k:

PC = b + s * k



Interrupções

Como salvar o estado da CPU?

- Salvar o *contexto de execução* (registradores*) em uma **pilha**
- Pode-se restaurar o contexto posteriormente
- Permite aninhamento de interrupções

* registradores variam de acordo com a arquitetura

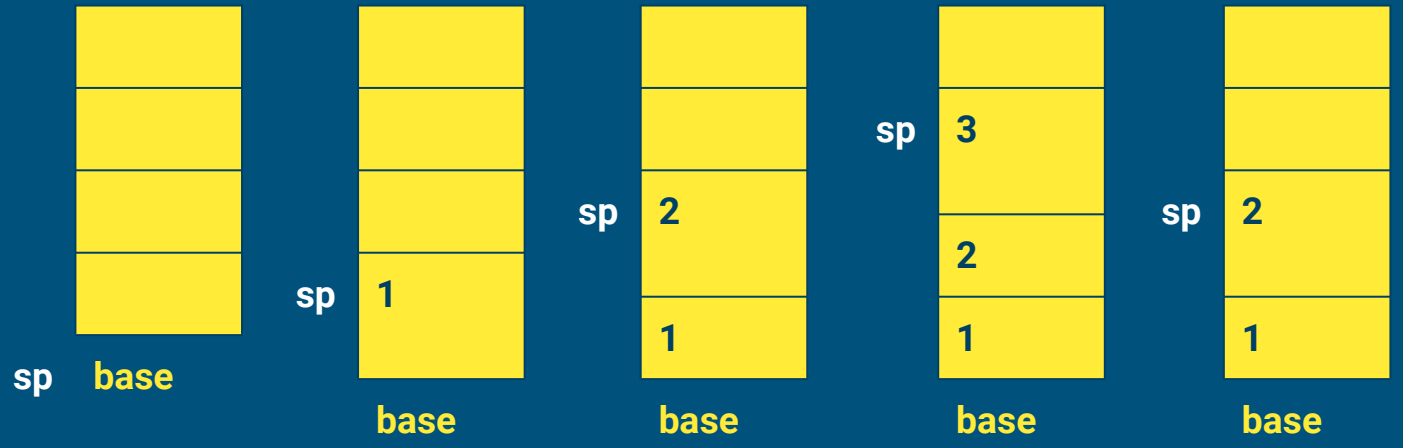
Interrupções

Pilha: estrutura de dados LIFO (Last-In First-Out)

- Operações básicas: *empilha*, *desempilha*
- Mantém-se um vetor de dados e um contador: *stack pointer* (sp)

Exemplo:

- Empilha 1
- Empilha 2
- Empilha 3
- Desempilha 3



Interrupções

Pilha: estrutura de dados LIFO (Last-In First-Out)

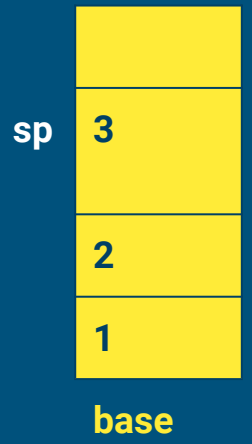
```
int pilha[PILHA_MAX];
int base = PILHA_MAX;
int sp = PILHA_MAX;

int pilha_cheia() {
    return (sp == 0);
}

int pilha_vazia() {
    return (sp == base);
}
```

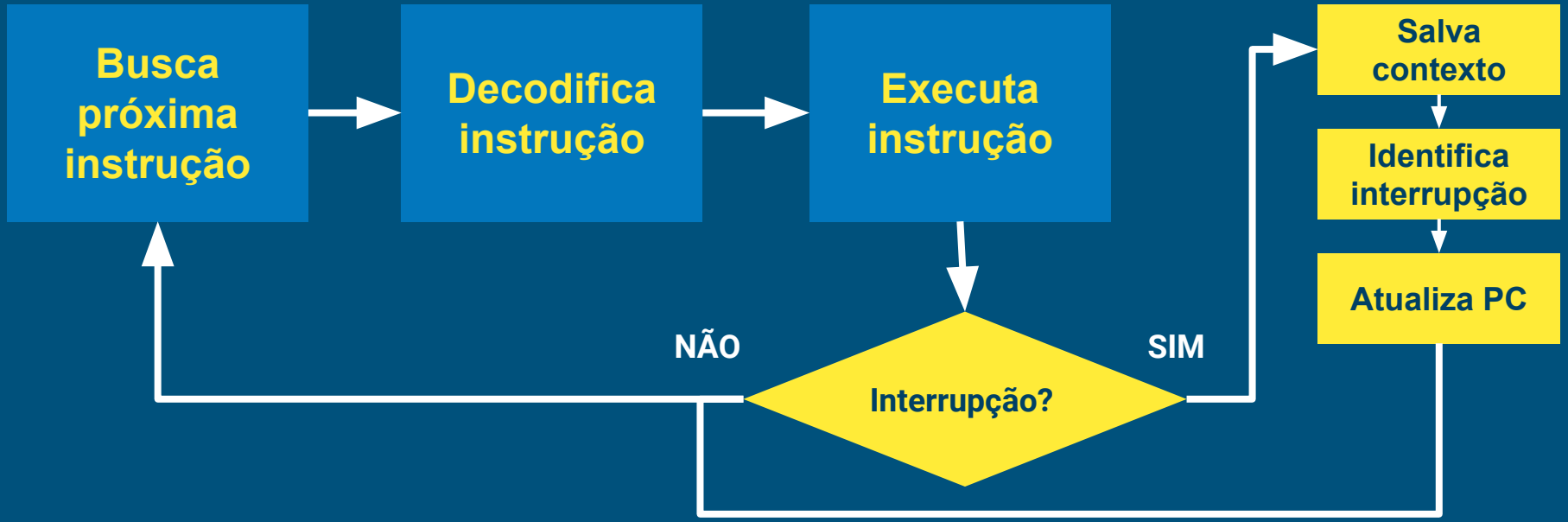
```
void empilha(int x) {
    if(!pilha_cheia())
        pilha[--sp] = x;
}

int desempilha() {
    if(!pilha_vazia())
        return pilha[sp++];
    else return 0;
}
```



Lembra do ciclo de instrução?

Processador vai verificar se há interrupção após executar instrução:



Entrada e saída (E/S)

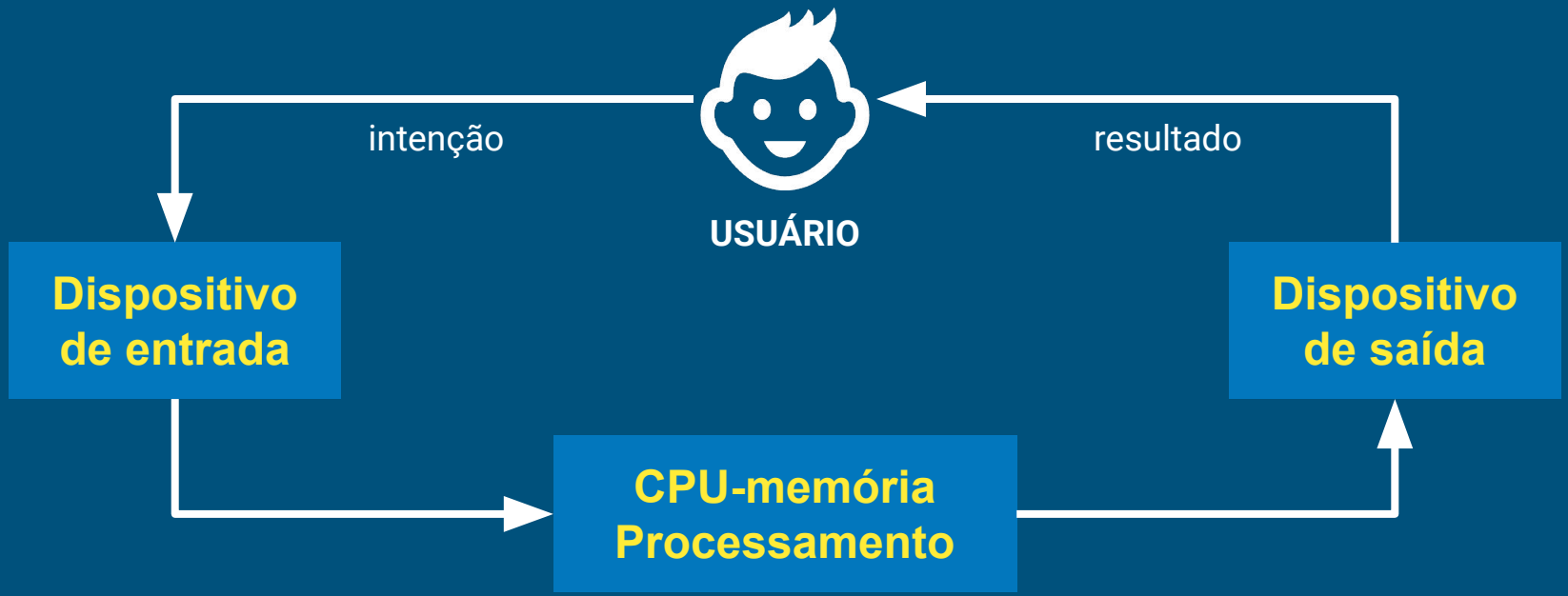
Entrada e saída (E/S)

É preciso que o computador se comunique com o usuário:

- Não basta a CPU processar os dados
- Deve haver a participação de um humano no ciclo de interação
- Usa-se os **dispositivos de entrada e saída** (E/S ou I/O)
  - Localizados fora do núcleo CPU-memória
  - Grande variedade de dispositivos: mouse, teclado, scanner, joystick, tela multitouch...
- Dispositivos precisam se comunicar com o computador

Entrada e saída (E/S)

Ciclo de interação:



Entrada e saída (E/S)

Um sistema de E/S deve ser capaz de:

- Receber/enviar dados ao mundo externo
- Converter os dados de entrada p/ um formato inteligível para a máquina
- Converter os dados de saída p/ um formato compreensível ao usuário
- Considera-se E/S qualquer transferência de dados de/para CPU-memória

Entrada e saída (E/S)

Devido às diferenças de dispositivos, a CPU não trabalha diretamente com eles.

- CPU se conecta a dispositivos que fazem o elo entre as especificidades de cada dispositivo e a CPU: interfaces de E/S (**controladoras**)

Entrada e saída (E/S)

Programação de dispositivos é específica e complexa

- SO provê camada de abstração p/ programas (open, write, etc.)
- BIOS services: https://en.wikipedia.org/wiki/BIOS_interrupt_call

Entrada e saída (E/S)

Formas de transmissão de dados:

- Serial: dados são transmitidos bit-a-bit, sequencialmente
- Paralela: dados são transmitidos em conjuntos de bits, em cabos paralelos

Qual das formas é mais rápida?

- Velocidade do barramento

Data rates

SATA revision 3.0	600 MB/s[72]
SATA revision 2.0	300 MB/s
SATA revision 1.0	150 MB/s[73]
PATA (IDE) 133	133.3 MB/s[f]

Fonte: https://en.wikipedia.org/wiki/Serial_ATA

Entrada e saída (E/S)

Diversidade de dispositivos

- Dispositivos de E/S são normalmente muito mais lentos que a CPU
  - Ex: teclado, disco...
- Utiliza-se canais específicos p/ a comunicação
  - PCIe, SATA, USB, etc.
- Grande variedade de dispositivos de E/S e de taxa de transferência de dados requer subsistema de controle e sincronização
  - Ponte sul (*south bridge*)

Entrada e saída (E/S)

Como ocorre a comunicação CPU-dispositivo?

- Memory-mapped I/O
  - Operações de entrada e saída baseadas em operações de memória
  - Registradores dos controladores são mapeados para endereços reservados da memória
  - Circuitos mais simples, usa-se instruções de acesso à memória

- Port-mapped I/O
  - Usa portas especiais para fazer entrada e saída de dados
  - Utiliza-se espaço de endereçamento separado da RAM
  - Necessita de instruções especiais (in / out)

Entrada e saída (E/S)

Game Boy Advance: memory-mapped I/O

area	start	end	length	port-size
IO RAM	0400:0000h	0400:03FFh	1kb	16 bit
VRAM	0600:0000h	0601:7FFFh	96kb	16 bit

Fonte: GBA Hardware
https://www.coranac.com/tonc/text/hardware.htm#sec-memory

Figura: Game Boy Advance por Evan-Amos

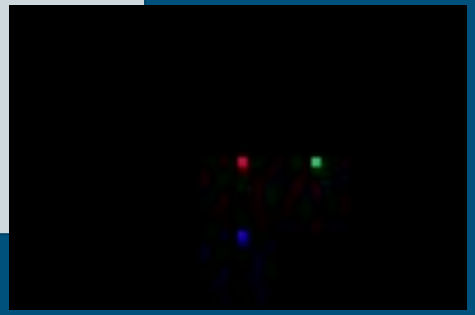
Entrada e saída (E/S)

```
int main()
{
    *(unsigned int*)0x04000000 = 0x0403;

    ((unsigned short*)0x06000000)[120+80*240] = 0x001F;
    ((unsigned short*)0x06000000)[136+80*240] = 0x03E0;
    ((unsigned short*)0x06000000)[120+96*240] = 0x7C00;

    while(1);

    return 0;
}
```



Fonte: My first GBA demo
https://www.coranac.com/tonc/text/first.htm

Técnicas p/ entrada e saída (E/S)

Três técnicas: ***Polling***, E/S via interrupção, E/S via DMA

1. Toda a comunicação é feita pela CPU
2. O programa deve sempre verificar o estado do dispositivo
3. Se o dispositivo tiver dados para enviar...
    a. A CPU deve ler os dados e escrevê-los na RAM
4. Ineficiente, devido às diferenças de tempo de funcionamento
    a. CPU perde tempo; poderia desempenhar outras tarefas

Técnicas p/ entrada e saída (E/S)

Três técnicas: *Polling*, **E/S via interrupção**, E/S via DMA

1. CPU solicita algo a um dispositivo (*comando*)

2. Controlador desempenha tarefa (ex: disk read), enquanto a CPU fica livre
    a. Há participação menor da CPU em relação ao *polling*

3. Controlador dispara interrupção
    a. *Polling* é como checar seu celular a cada minuto para ver se chegou mensagem
    b. E/S via interrupção é como ser notificado de que uma mensagem chegou :)

4. CPU faz a transferência dos dados (do controlador p/ memória)

Técnicas p/ entrada e saída (E/S)

Três técnicas: *Polling*, E/S via interrupção, **E/S via DMA**

1. Técnica de E/S que utiliza *Direct Memory Access* (DMA)
    a. Libera a CPU do trabalho de transferir os dados p/ a memória

2. Ocorre interrupção quando operação termina, não a cada evento
    a. Mais eficiente!

3. Requer Controladora DMA
    a. DMA faz a E/S fora da CPU
    b. Interessante para transferências de grandes volumes de dados

Técnicas p/ entrada e saída (E/S)

DMA: projetado p/ acelerar transferências de dados

Processo de inicialização

BIOS: Basic Input/Output System

BIOS é um programa utilizado para inicializar o hardware

- Armazenado em memória não-volátil
  - Incluído na placa-mãe (lida com especificidades do modelo)
- É o primeiro software a ser executado quando se liga a máquina
  - *Power-on self-test* (POST): testa e inicializa componentes de hardware
- Carrega o **boot loader** da memória secundária
- Provê uma *camada de serviços* (abstração p/ acesso ao hardware)
  - INT 13h; atualmente em desuso (SOs lidam com o hardware diretamente)

BIOS carrega o boot loader

O **boot sector** é um setor especial de um dispositivo de armazenamento (HD, pen-drive...)

- A BIOS identifica e carrega o boot sector na RAM
  - Se houver múltiplos dispositivos conectados, escolhe um (*boot order*)
- Boot sector contém o código do **boot loader**
  - Programa responsável por carregar outro programa (ex: sistema operacional) na RAM
- BIOS carrega e executa o boot loader

MBR: Master Boot Record

Primeiro setor de um disco particionado:

- Contém codigo de *bootstrapping*
  - Chain-loading
- Descrição das partições
- Assinatura 55AA

Endereço		Descrição		Tamanho em bytes
Hex	Dec			
0000	0	Código de arranque do SO		**440** (max. 446)
01B8	440	Assinatura opcional		**4**
01BC	444	normalmente nulo ; 0x0000		**2**
01BE	446	**Tabela de partições primarias** (Quatro entradas de 16 bytes (IBM Partition Table scheme))		**64**
01FE	510	55h	MBR signature; 0x55AA	**2**
01FF	511	AAh		
Tamanho total do MBR : 440 + 4 + 2 + 64 + 2 =				**512**

Fonte: Master boot record
https://en.wikipedia.org/wiki/Master_boot_record

Parte prática

Roteiro

1. Desenvolver controladora de disco
2. Introduzir mecanismo de interrupções na máquina virtual
3. Introduzir mecanismo de E/S na máquina virtual
4. Construir programa de inicialização: carregar bootloader do disco
5. Se for possível..... Carregar S.O. do disco :)

Dúvidas?