Insper

Sistemas Hardware-Software

Aula 15 – Exceções, Processos e Sinais

2019 – Engenharia

Igor Montagner, Fábio Ayres <igorsm1@insper.edu.br>

Até agora

Um programa tem acesso total aos recursos da máquina:

- Pode ocupar toda RAM (acessar todos os 2^{64} endereços de memória)
- Tem uso exclusivo de todos os registradores
- Tem uso exclusivo do tempo da CPU
- Tem acesso instantâneo ao disco e à rede

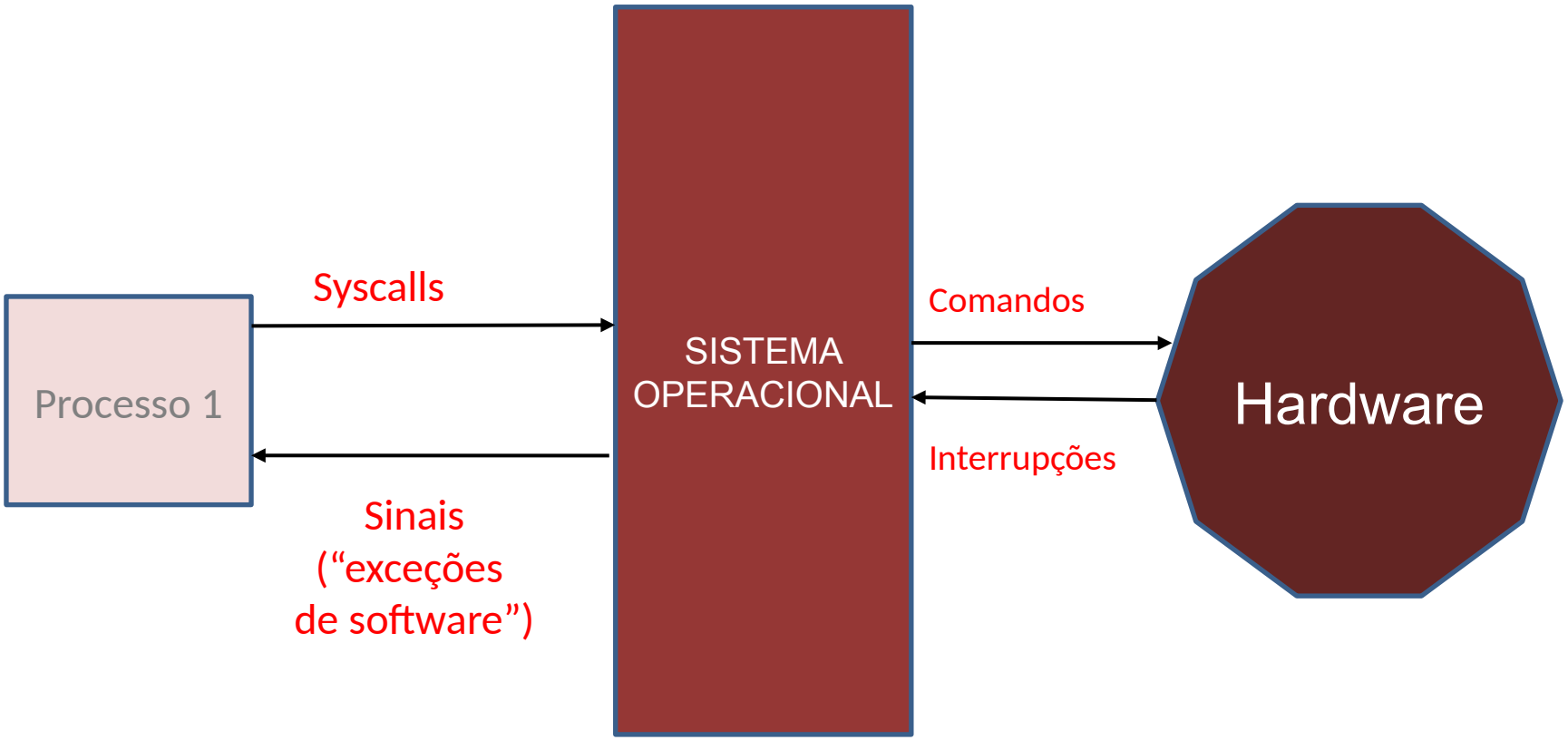
Até agora...

Um programa tem acesso total aos recursos da máquina:

- ~~Pode ocupar toda RAM (acessar todos os 2^{64} endereços de memória)~~
- ~~Tem uso exclusivo de todos os registradores~~
- ~~Tem uso exclusivo do tempo da CPU~~
- ~~Tem acesso instantâneo ao disco e à rede~~

Como o SO faz isso?

Chamadas de sistema



Insper

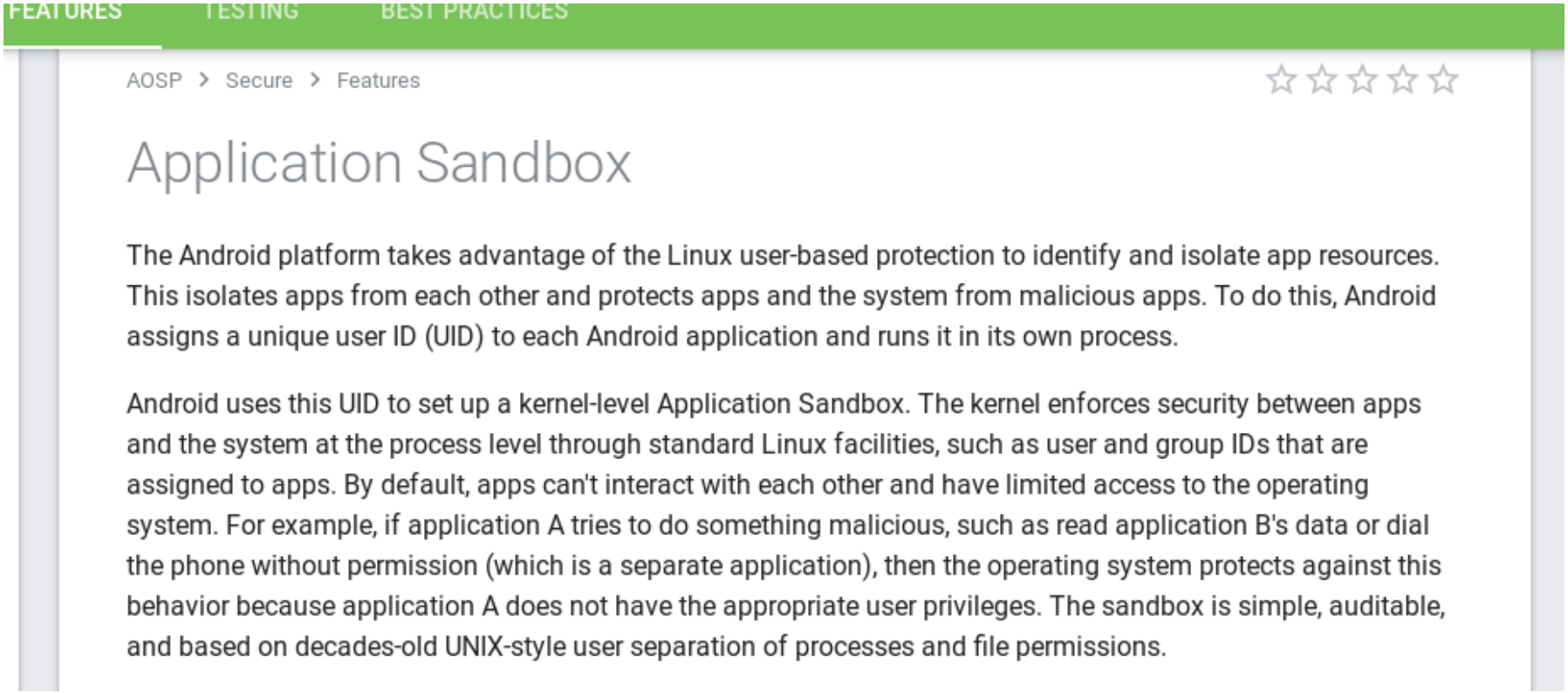
POSIX - syscalls

- Gerenciamento de usuários e grupos
- Manipulação de arquivos (incluindo permissões) e diretórios
- Criação de processos e carregamento de programas
- Comunicação entre processos
- Interação direta com hardware (via drivers)

POSIX - Arquivos

- `open, close, read, write`
- Todo arquivo pertence a um usuário e a grupo.
- Permissões: **dono do arquivo**, **membros do grupo dono** e **resto**.
- Codificação das permissões:
    - 4 para leitura
    - 2 para escrita
    - 1 para execução

Permissões na prática: Android



https://source.android.com/security/app-sandbox

Insper

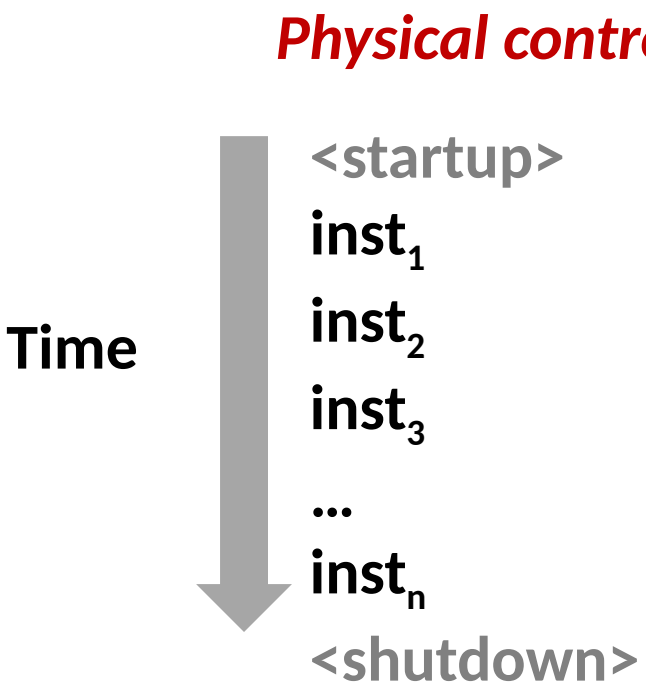
Parte 2 – aula passada

Hoje

- Definir o que é um processo e seu contexto
- Entender quais mecanismos de hardware são usados para alternar entre processos
- Analisar o ciclo de vida de um processo

Fluxo de controle

- Desde o início até o seu desligamento, a CPU apenas lê e executa uma sequência de instruções, uma por vez
- Esta sequência é o fluxo de controle da CPU



Insper

Alterando o fluxo de controle

Até o momento, temos dois mecanismos para alterar o fluxo de controle:

- Saltos (*jumps*: **`jmp`**) e desvios (*branches*: **`je`**, **`jl`**, **`jge`**, etc)

- Chamadas (calls) e retornos

Permitem alterar o fluxo de controle em função de mudanças no estado do programa

Alterando o fluxo de controle

Mas isto não basta: como reagir a mudanças no estado do sistema?

- Dados lidos do disco ou da rede
- Programa executa uma instrução ilegal ou em condições inválidas (como divisão por zero)
- Usuário digita Ctrl-C no teclado
- Timer de sistema notifica o programa

Precisamos de mecanismos para reagir a estes eventos "excepcionais"

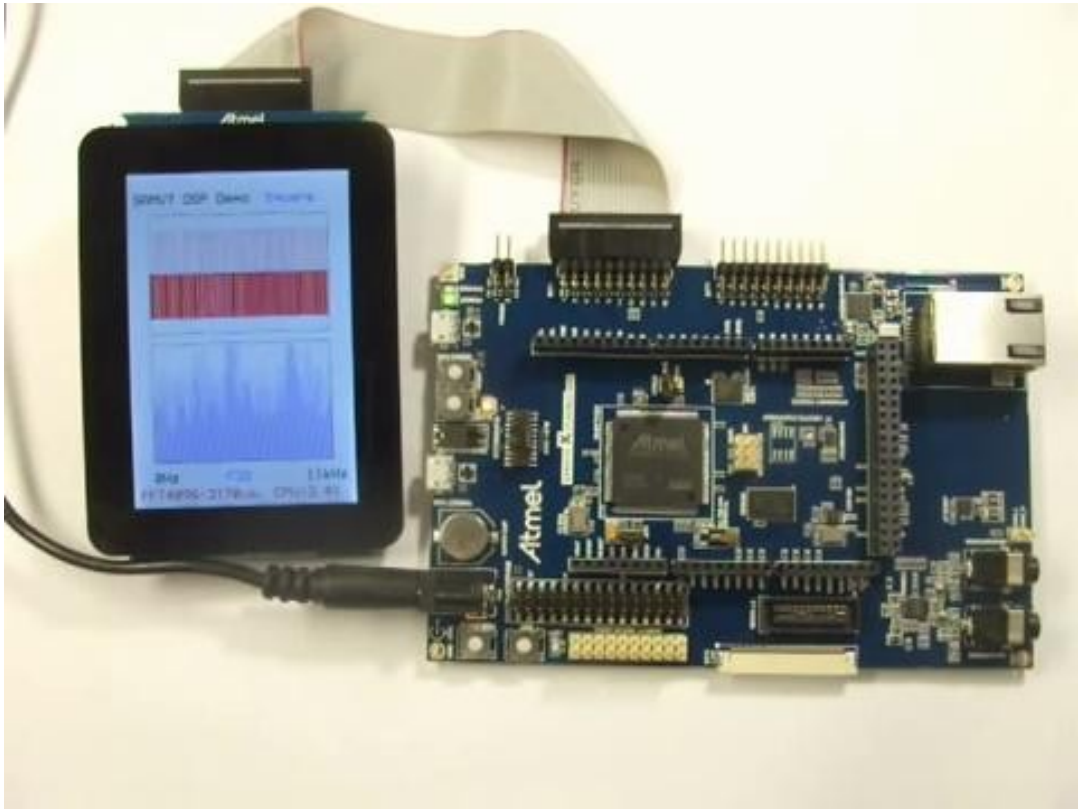
Alterando o fluxo de controle

Mas isto não basta: como reagir a mudanças no estado do sistema?

- (Hardware) Dados lidos do disco ou da rede
- (Hardware) Programa executa uma instrução ilegal ou em condições inválidas (como divisão por zero)
- (Software) Usuário digita Ctrl-C no teclado
- (Hardware ou Software) Timer de sistema notifica o programa

Exceções/Interrupções

Muito usadas em Embarcados



Insper

Interrupções (diferenças)

Embarcados:

- Somente um programa rodando, mas com várias tarefas concorrentes
- Tarefas compartilham espaço de memória

Desktop/Celular:

- Vários programas (não confiáveis) rodando
- Programas começam e terminam a qualquer momento
- Isolamento de memória e recursos

Exceções síncronas

Resultam da execução de uma instrução

- ***Traps***: intencionais (*system calls)*
  - Controle retorna para a próxima instrução

- ***Faults***: não-intencionais, talvez recuperáveis
  - Exemplos: page faults (recuperável), falha de proteção (não-recuperável)
  - Ou re-executa a instrução falha, ou termina o programa

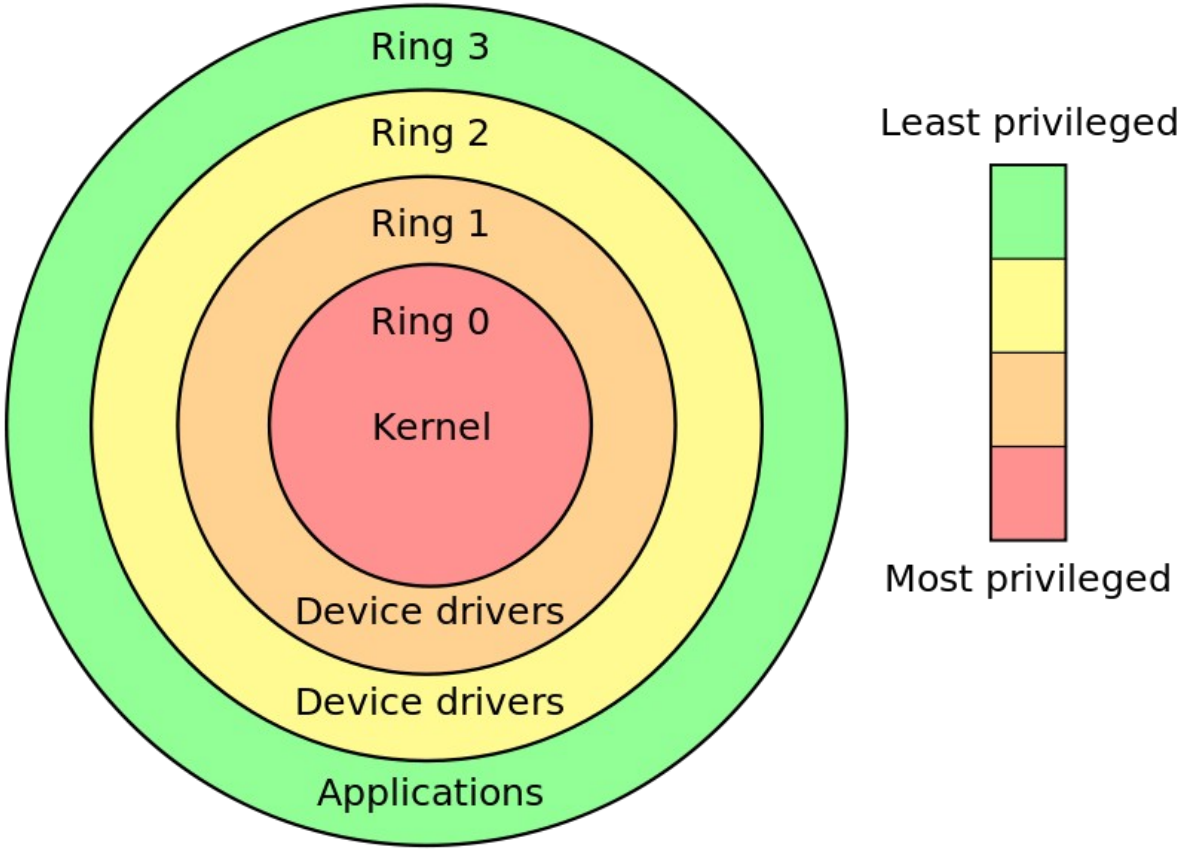
- ***Aborts***: não-intencionais, não-recuperáveis (instrução ilegal)
  - Termina o programa

Exceções assíncronas (interrupções)

- Causadas por eventos externos ao processador
  - Alguém ligou o pino de interrupção do processador
  - *Handler* retorna para a próxima instrução

- Exemplos:
  - *Timer: u*m chip de *timer* externo ao processador dispara uma interrupção periodicamente, com intervalo de alguns milissegundos.
  - I/O: disco, rede, teclado, etc.

Níveis de proteção em x86



https://en.wikipedia.org/wiki/Protection_ring

Insper

Interrupções em Sistemas Operacionais

Kernel: software do sistema que gerencia

- Programas
- Memória
- Recursos do hardware

Roda com privilégios totais no hardware.

Toda interrupção roda neste modo de proteção (ring0 ou *kernel land*).

Interrupções em Sistemas Operacionais

Processo de usuário: qualquer programa sendo executado no computador. **A falha de um processo não afeta os outros.**

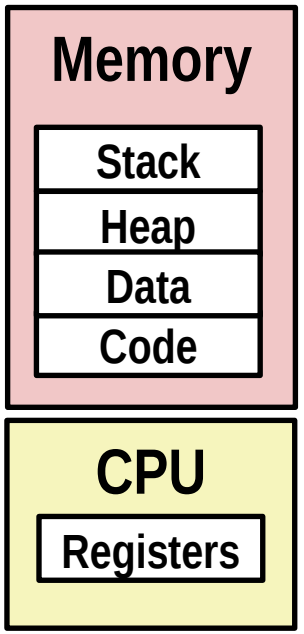
Roda com privilégios limitados (ring 3 ou *user land*). Interaje com o hardware por meio de chamadas ao kernel.

- Memória
- Acesso ao disco e outros periféricos
- Comunicar com outros processos

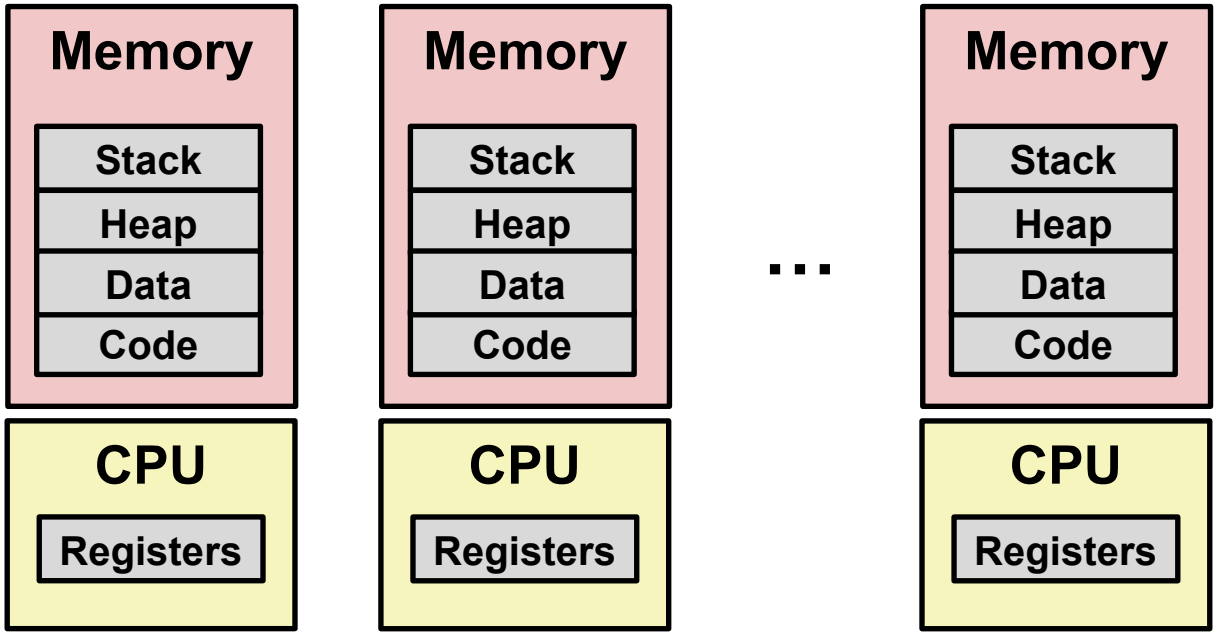
Processos

- Fluxo de controle lógico
  - Cada programa parece ter uso exclusivo da CPU
  - Provido pelo mecanismo de *chaveamento de contexto*

- Espaço de endereçamento privado
  - Cada programa parece ter uso exclusivo da memória principal
  - Provido pelo mecanismo de *memória virtual*

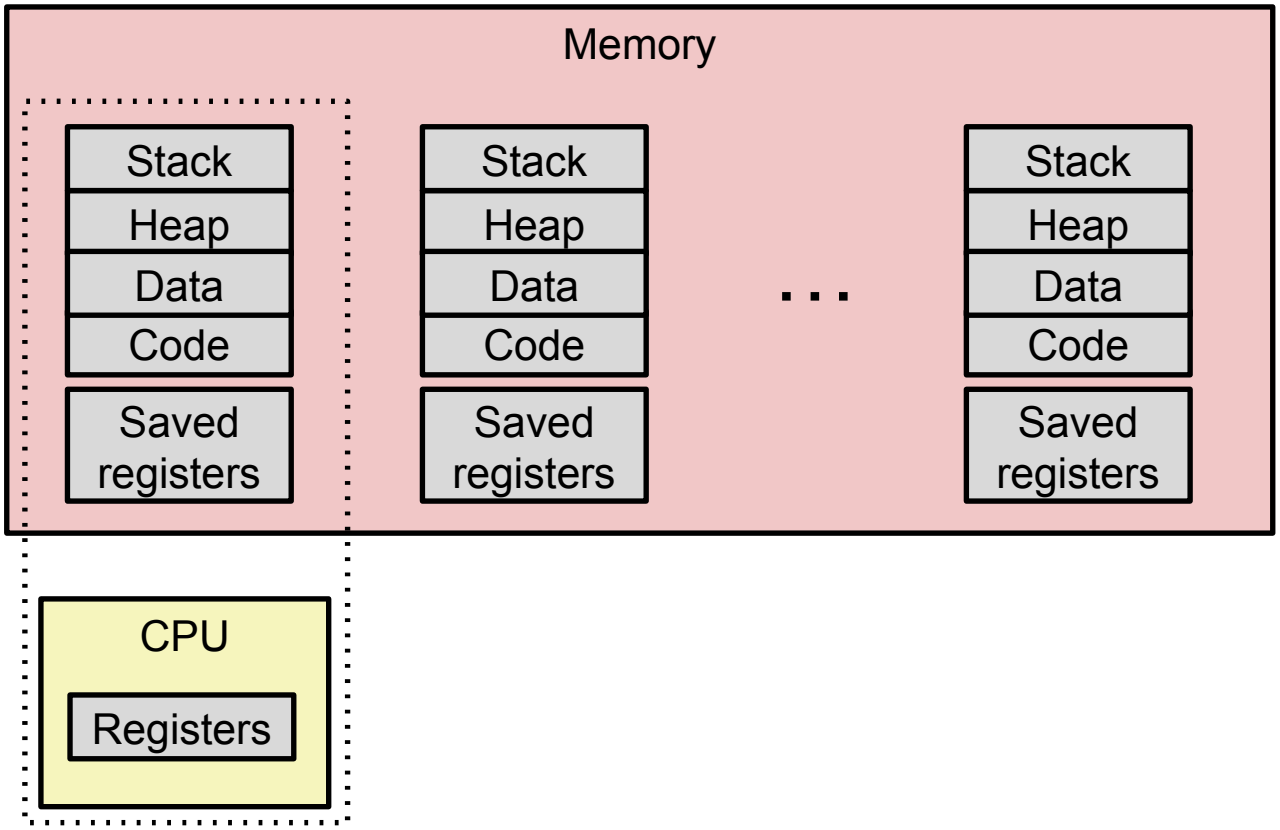


A ilusão do multiprocessamento



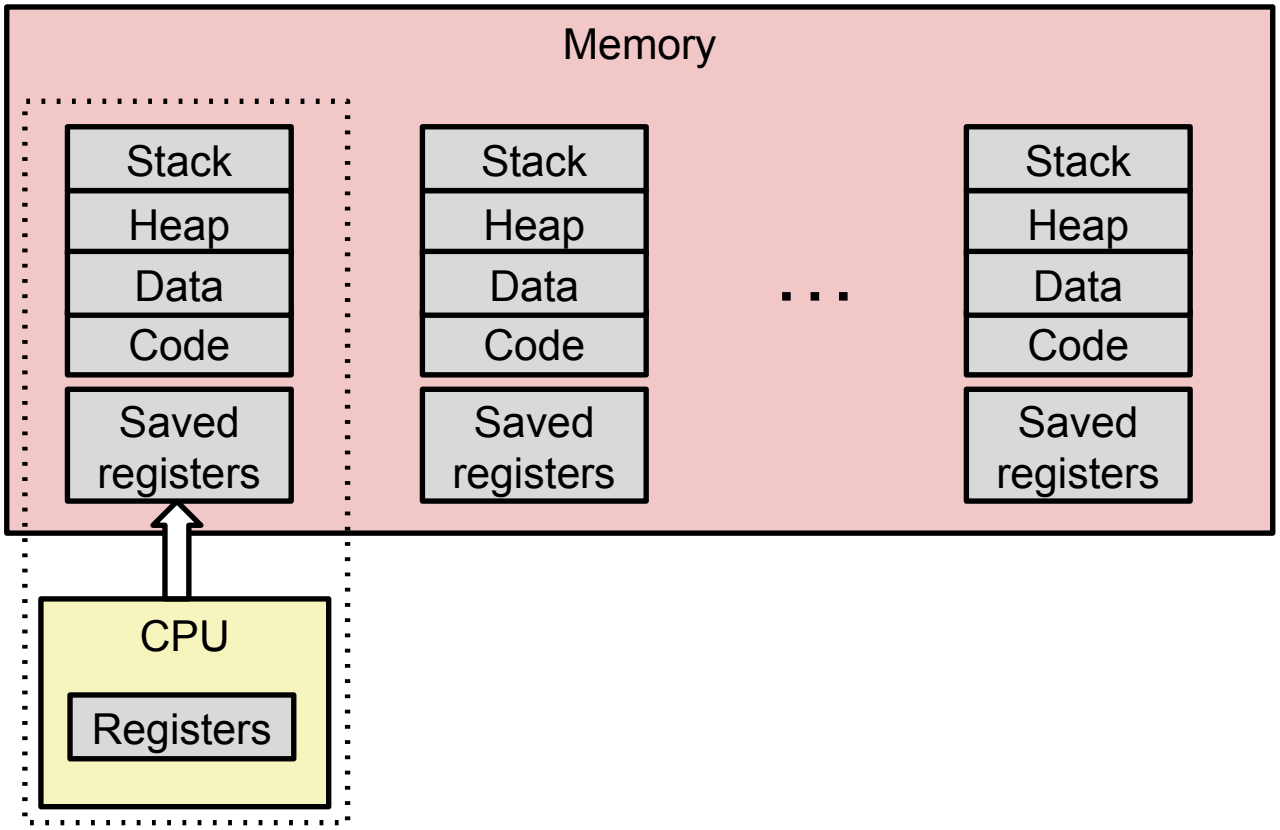
Insper

A realidade do multiprocessamento



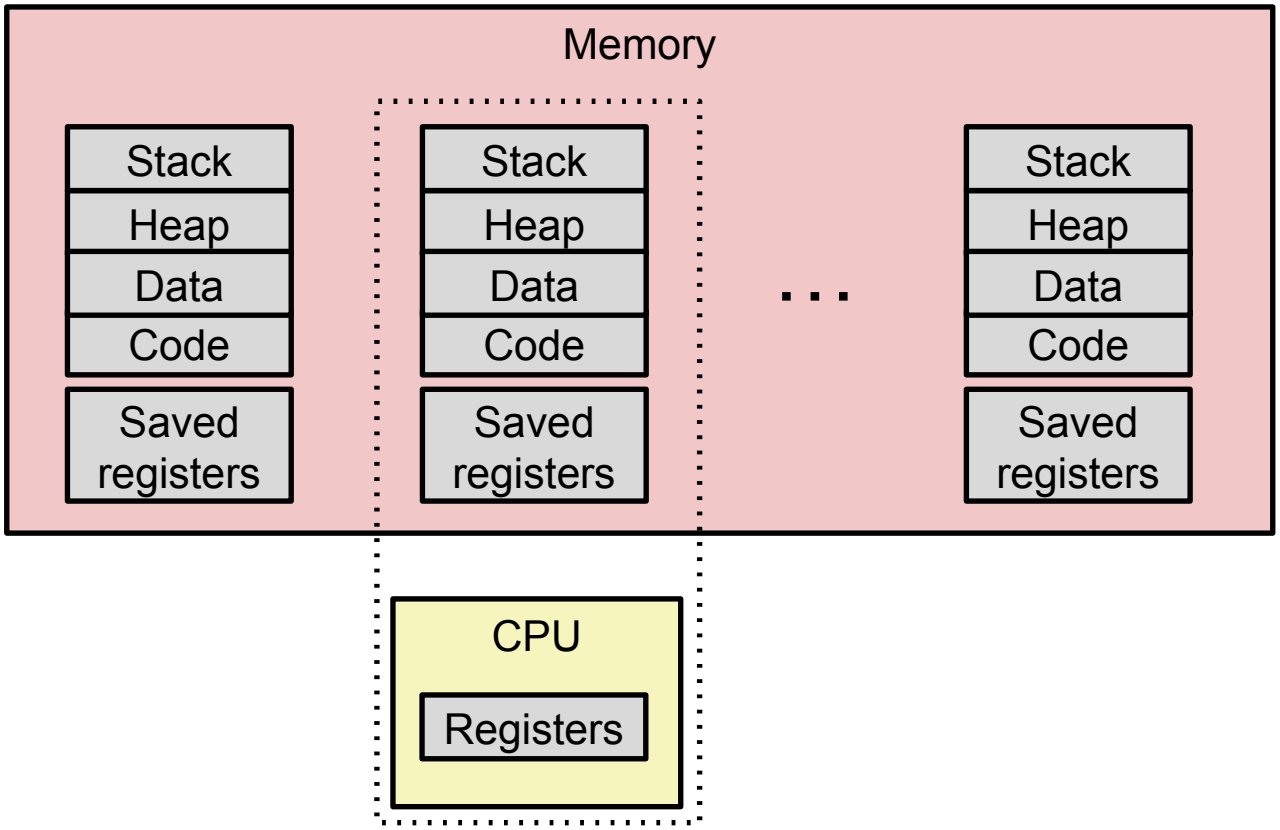
- Execução de processos intercalada
- Espaços de endereçamento gerenciados pelo sistema de memória virtual
- Valores de registradores para processos em espera são gravados em memória

A realidade do multiprocessamento



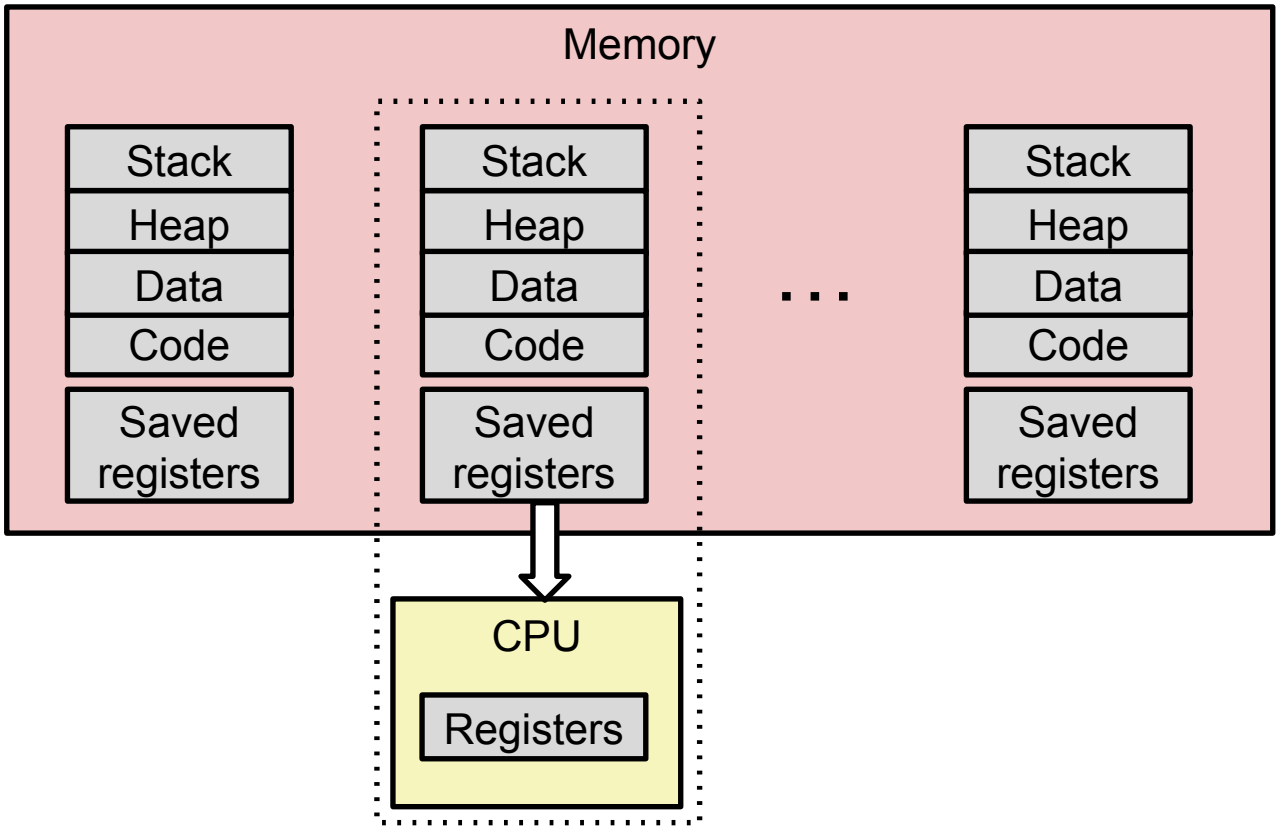
- Grava registradores na memória

A realidade do multiprocessamento



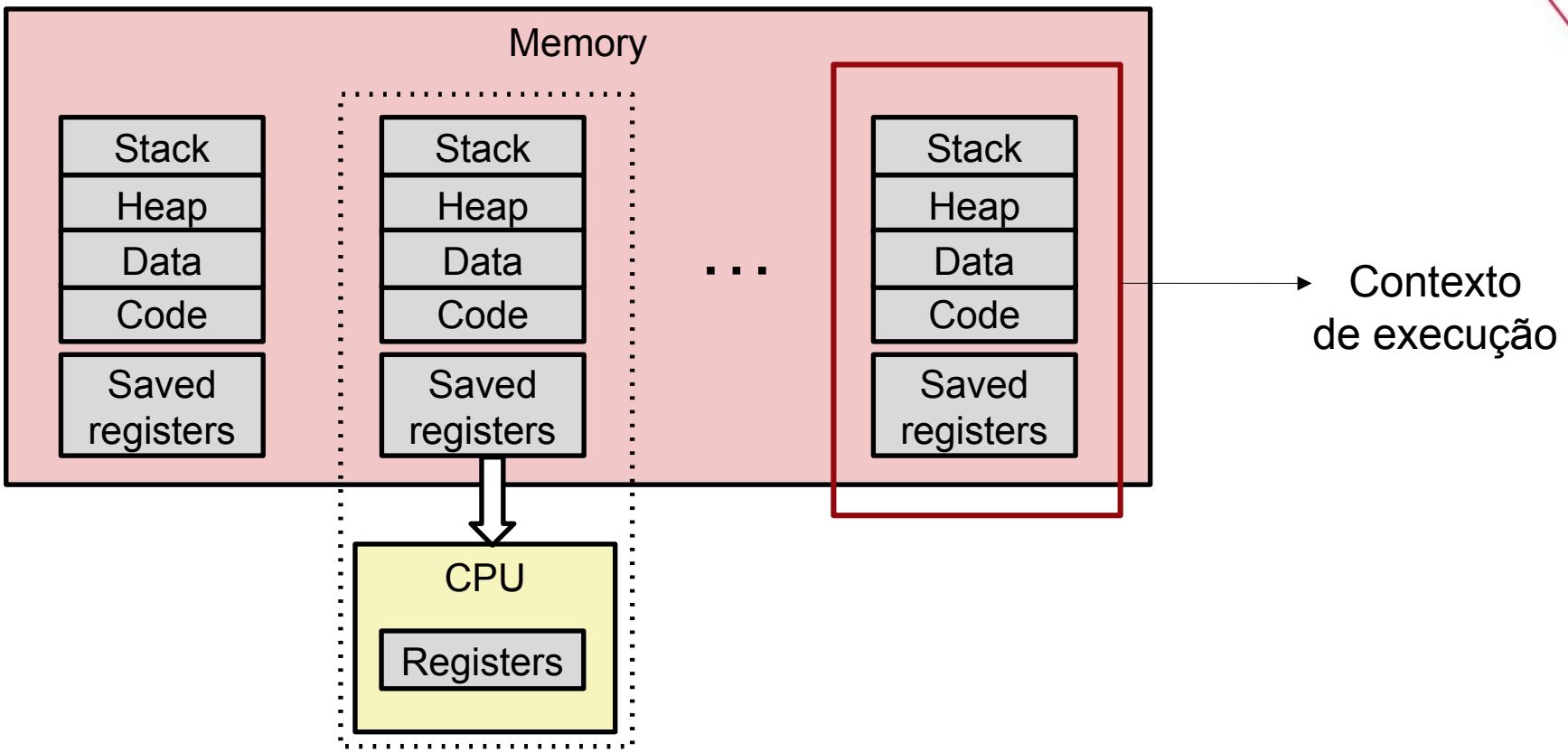
- Escolhe próximo processo a ser executado

A realidade do multiprocessamento



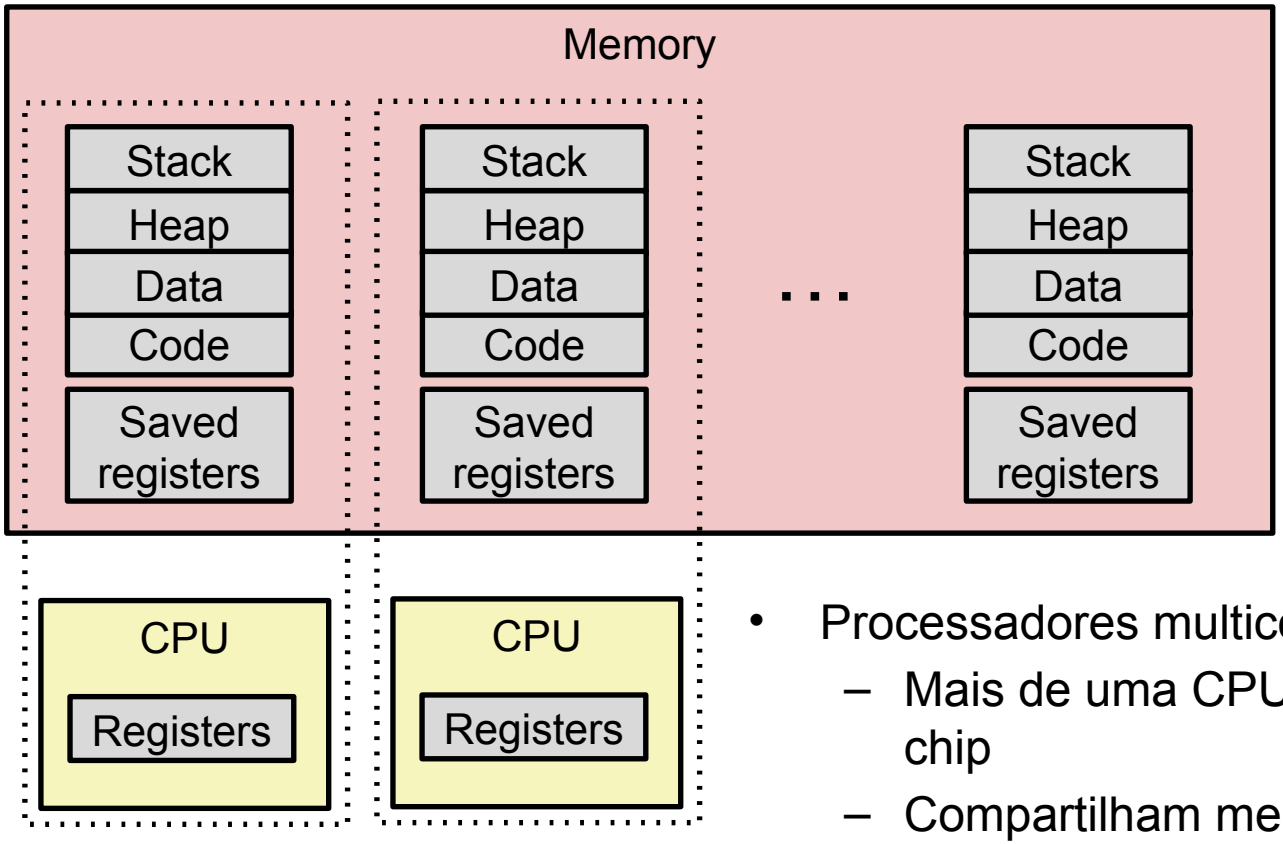
- Carrega registradores gravados e troca de espaço de endereçamento (*context switch* – chaveamento de contexto)

A realidade do multiprocessamento



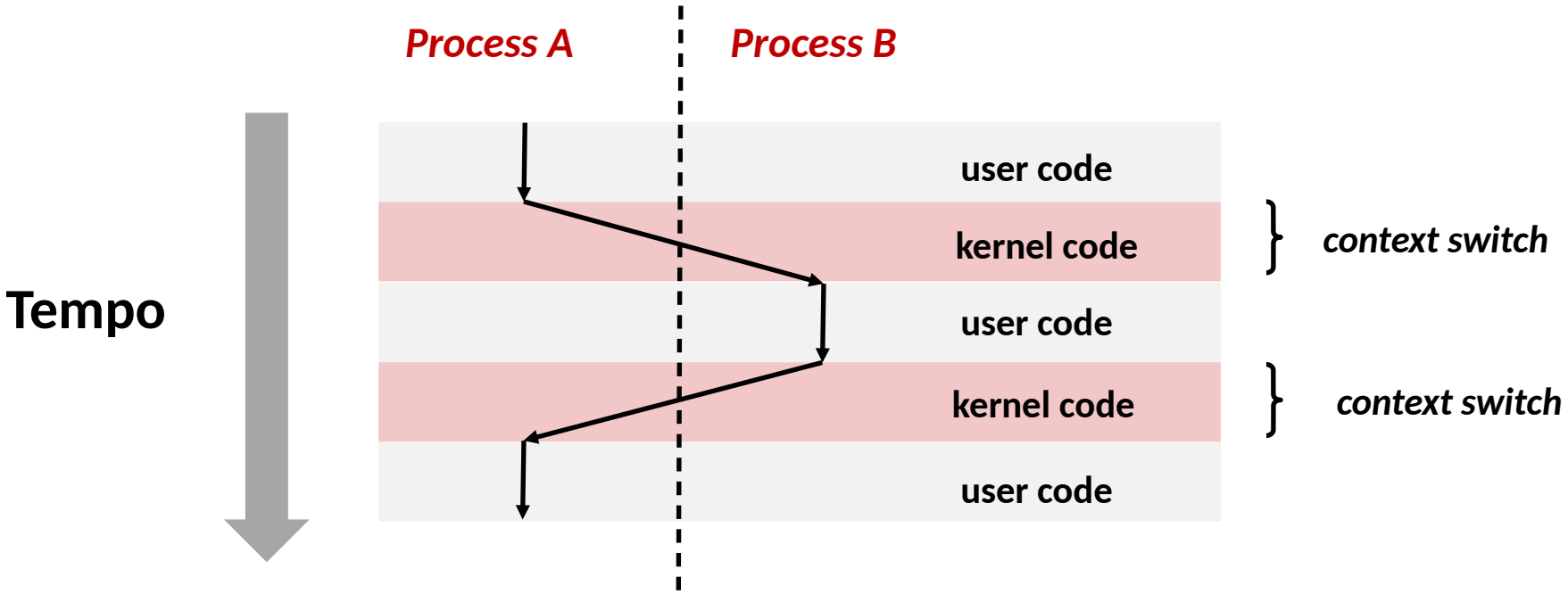
- Carrega registradores gravados e troca de espaço de endereçamento (*context switch* – chaveamento de contexto)

A realidade moderna do multiprocessamento

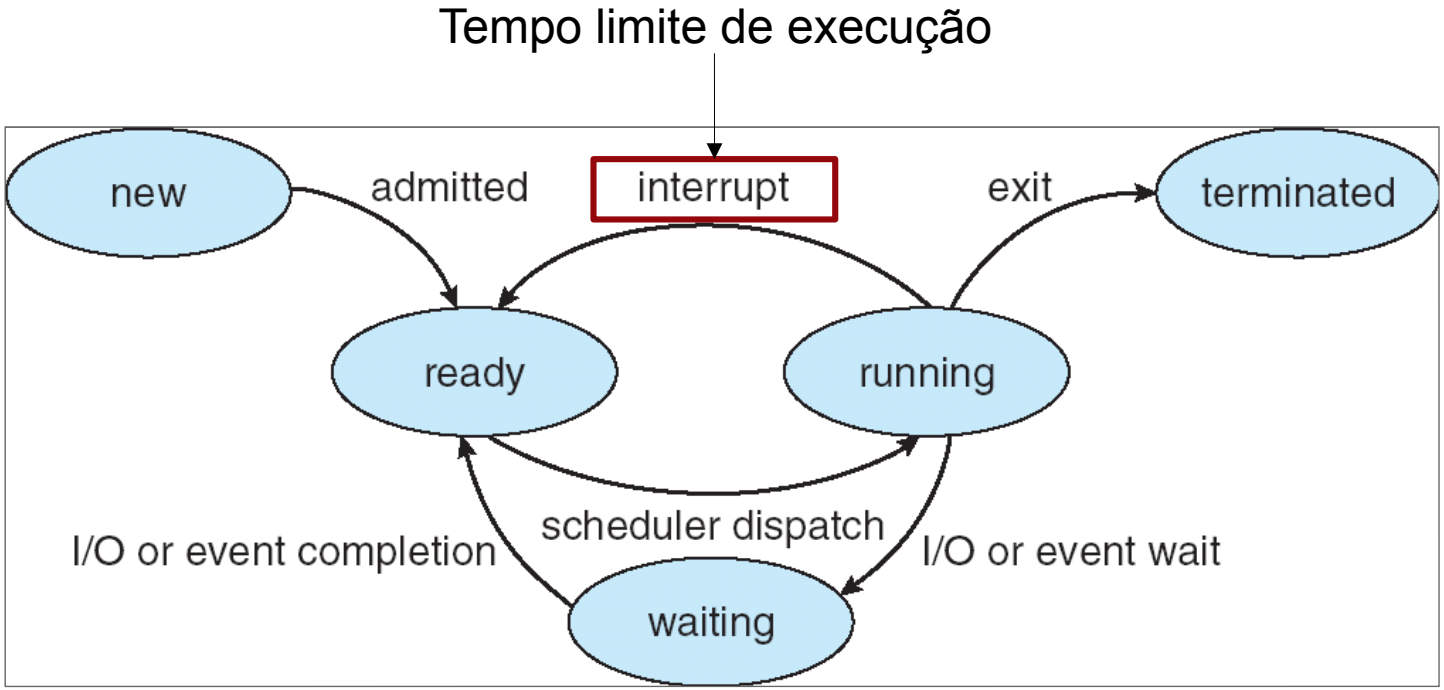


- Processadores multicore
  - Mais de uma CPU em um mesmo chip
  - Compartilham memória principal e parte do cache (cache L3)
  - Cada core pode executar um processo separado
    - Agendamento de processos em cores feito pelo kernel

Chaveamento de contexto

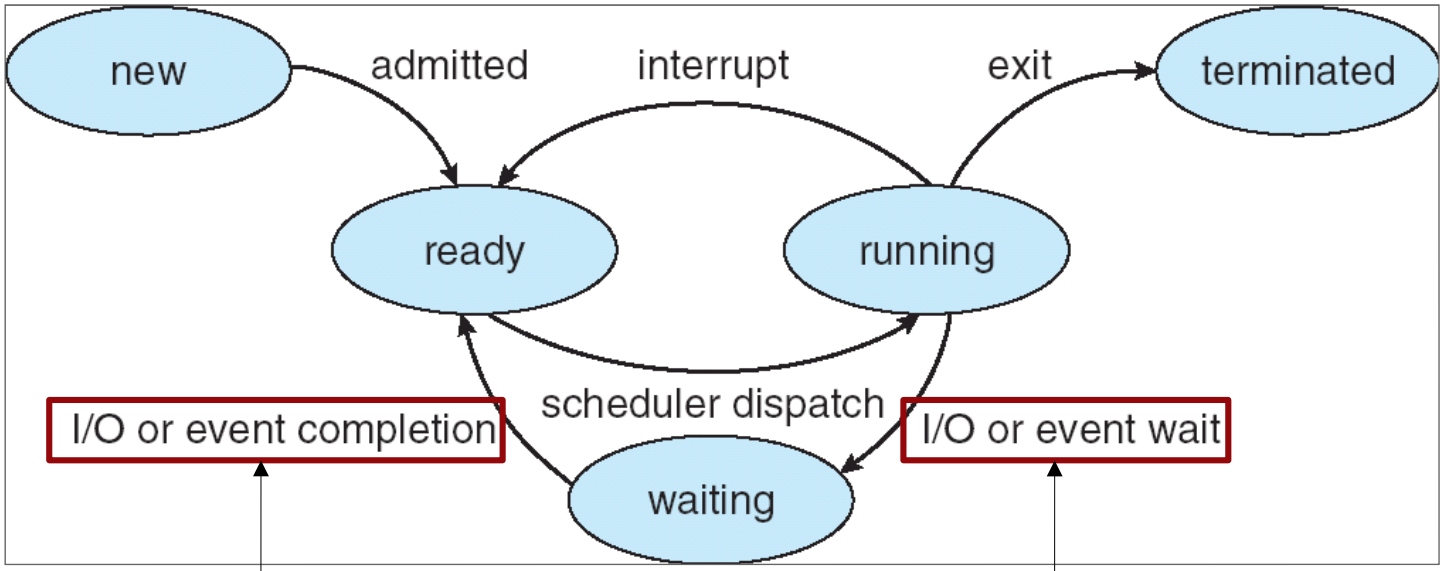


Ciclo de vida de processos



Insper

Ciclo de vida de processos



Interrupção ligada por algum hardware (disco, rede, usb, etc)

Chamada de sistema
Para acessar hardware

Insper

Criação de processos

Criamos processos usando a chamada de sistema *fork*

```
pid_t fork();
```

O fork cria um clone do processo atual e retorna duas vezes

No processo original (pai) `fork` retorna o pid do filho

O pid do pai é obtido chamando

```
pid_t getpid();
```

No processo filho `fork` retorna o valor 0. O pid do filho é obtido usando

```
pid_t getpid();
```

O pid do pai pode ser obtido usando a chamada

```
pid_t getppid();
```

Criação de processos (10 minutos)

```c
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>

int main() {
    pid_t pai, filho;
    int variavel = 5;

    filho = fork();
    if (filho == 0) {
        // processo filho aqui
        pai = getppid();
        filho = getpid();
        variavel *= 2;
        printf("eu sou o processo filho %d, meu pai é %d\nvariavel %d\n",
            filho, pai, variavel);
    } else {
        // processo pai aqui!
        pai = getpid();
        printf("eu sou o processo pai %d, meu filho é %d\nvariavel %d\n",
            pai, filho, variavel);
    }
    return 0;
}
```

Atividade

- Roteiro parte 1

- O programa `man` contém documentação sobre todas as funções usadas hoje. Use-o ;)

- O texto do manual por vezes é confuso ou muito técnico. Se você não entendeu ele me chame.

Valor de retorno

- Um processo pode esperar pelo fim de outro processo filho usando as funções

```
pid_t wait(int *wstatus);
pid_t waitpid(pid_t pid, int *wstatus, int options);
```

- A primeira espera qualquer um dos filhos, enquanto a segunda espera um filho (ou grupo de filhos) específico.

- Ambas bloqueiam até que um processo filho termine e retornam o pid do processo que acabou de terminar.

- O valor de retorno do processo é retornado via o ponteiro `wstatus.`

E se o processo filho deu ruim?

- É possível checar se um processo filho terminou corretamente usando o conteúdo de `wstatus` e as seguintes macros:
  - `WIFEXITED(wstatus)`: `true` se o filho acabou sem erros
  - `WEXITSTATUS(wstatus)`: valor retornado pelo main
  - `WIFSIGNALED(wstatus)`: `true` se o filho foi terminado de maneira abrupta (tanto por um ctrl+c quanto por um erro)
  - `WTERMSIG(wstatus)`: código numérico representando a razão do encerramento do filho

Atividade

- Parte 2

- O programa `man` contém documentação sobre todas as funções usadas hoje. Use-o ;)

- O texto do manual por vezes é confuso ou muito técnico. Se você não entendeu ele me chame.

Como executar novos programas?

- fork só permite a criação de clones de um processo!
- Família de funções `exec` permite o carregamento de um programa do disco
- É permitido setar as variáveis de ambiente do novo programa e seus argumentos.
- Funções da família `exec` nunca retornam: o programa atual é destruído durante o carregamento do novo programa

Insper

www.insper.edu.br