Computação Concorrente (DCC/UFRJ)

Aula 2: Desenvolvimento de aplicações concorrentes

Prof. Silvana Rossetto

13 de março de 2012

Problema:

- Suponha uma empresa que usa um software de computação científica cujo processamento central consiste na **multiplicação de pares de matrizes de grande dimensão**
- Para melhorar o desempenho do software (i.e., diminuir o tempo de processamento), o dono da empresa decidiu investir em uma máquina com 4 núcleos de processamento, e agora ele quer saber qual o ganho de desempenho que poderá obter

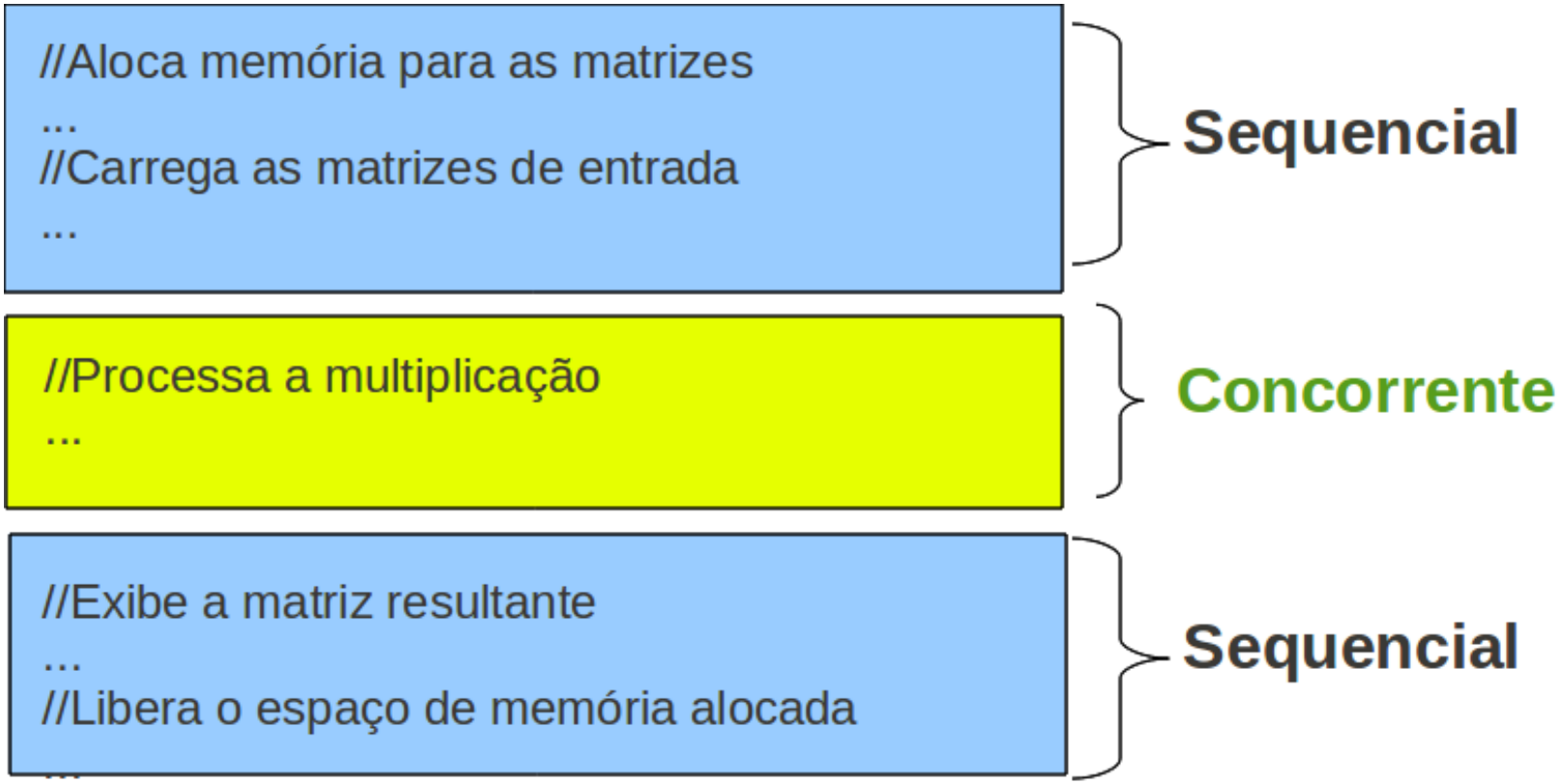
Você foi contratado para acessorar a empresa nessa transição: **quais passos seguir e como responder a pergunta do dono da empresa?**

Avaliação do ganho de desempenho previsto

Algoritmo básico

```
//Aloca memória para as matrizes
...
//Carrega as matrizes de entrada
...
//Processa a multiplicação
...
//Exibe a matriz resultante
...
//Libera o espaço de memória alocada
...
```

PASSO 1: análise do algoritmo básico



PASSO 2: cálculo do ganho previsto

Lei de Amdahl

- Estima o ganho de velocidade de execução de um programa usando vários processadores
- **O ganho de velocidade da execução é dado por:** $T(sequencial)/T(concorrente)$

Lei de Amdahl para estimar ganho da concorrência

Tempo sequencial

- t_s: tempo da parte sequencial do programa (que deve executar em um único processador)
- $t_p(1)$: tempo da parte concorrente do programa usando um processador
- $T(1)$: tempo total do programa usando um processador $(t_s + t_p(1))$

Tempo concorrente

- $t_p(P)$: tempo da parte paralela do programa usando P processadores $(t_p(1)/P)$
- $T(P)$: tempo total do programa usando P processadores $(t_s + t_p(P))$.

Lei de Amdahl para estimar ganho da concorrência

O ganho de velocidade da execução é dado por:

$T(1)/T(P)$

Retomando o problema inicial...

Considere que o programa que implementa a multiplicação de matrizes gasta (com 1 processador) $t_s = 8seg$ de processamento para a parte sequencial e $t_p = 16seg$ de processamento para a parte concorrente, qual será o ganho de velocidade executando a aplicação em uma máquina com 4 núcleos de processamento?

Exercício

Considere um programa com 10 atividades (tempo similar), das quais 8 podem executar em paralelo (ao mesmo tempo). Qual será seu ganho de execução em uma máquina com 4 processadores?

Exercício

Considere um programa com 10 atividades (tempo similar), das quais 8 podem executar em paralelo (ao mesmo tempo). Qual será seu ganho de execução em uma máquina com 4 processadores?

Resposta

- seja **p** a fração da tarefa que pode ser executada em paralelo
- assumindo que o tempo para um processador completar a aplicação seja de 1 unidade de tempo, com n-processadores a parte paralela gastará p/n e a parte sequencial $1 - p$, somando $(1 - p + p/n)$, então, $S = \frac{1}{(1-p+p/n)}$
- $p = 8/10 = 4/5$, então $1 - p = 1 - 4/5 = 1/5$
- daí, $S = \frac{1}{(1/5+(4/5)/4)} = \frac{1}{(2/5)} =$**2,5**

PASSO 3: projeto e implementação da solução concorrente

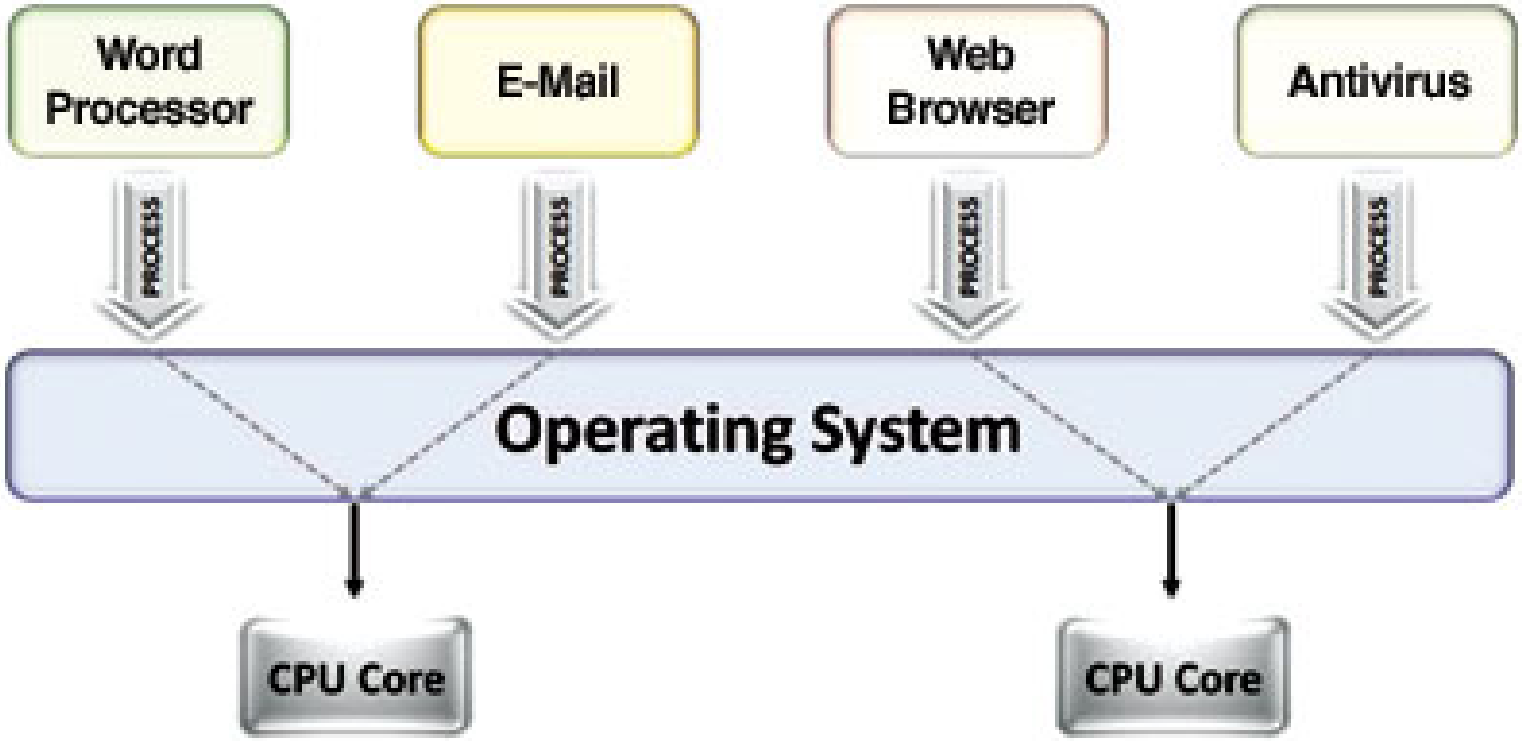
Para explorar o hardware com vários núcleos de processamente é preciso implementar a versão concorrente da aplicação, como fazer isso?

Conceitos básicos

Processo: programa em execução

1. **Coleção de recursos**: espaço de memória virtual para armazenar a imagem do processo (código, dados, pilha), dispositivos de E/S alocados, descritores de arquivos abertos, etc.
2. **Escalonamento/execução**: um processo tem um estado de execução e é a entidade que é escalonada pelo Sistema Operacional para execução pelo processador

Alocação das unidades de processamento

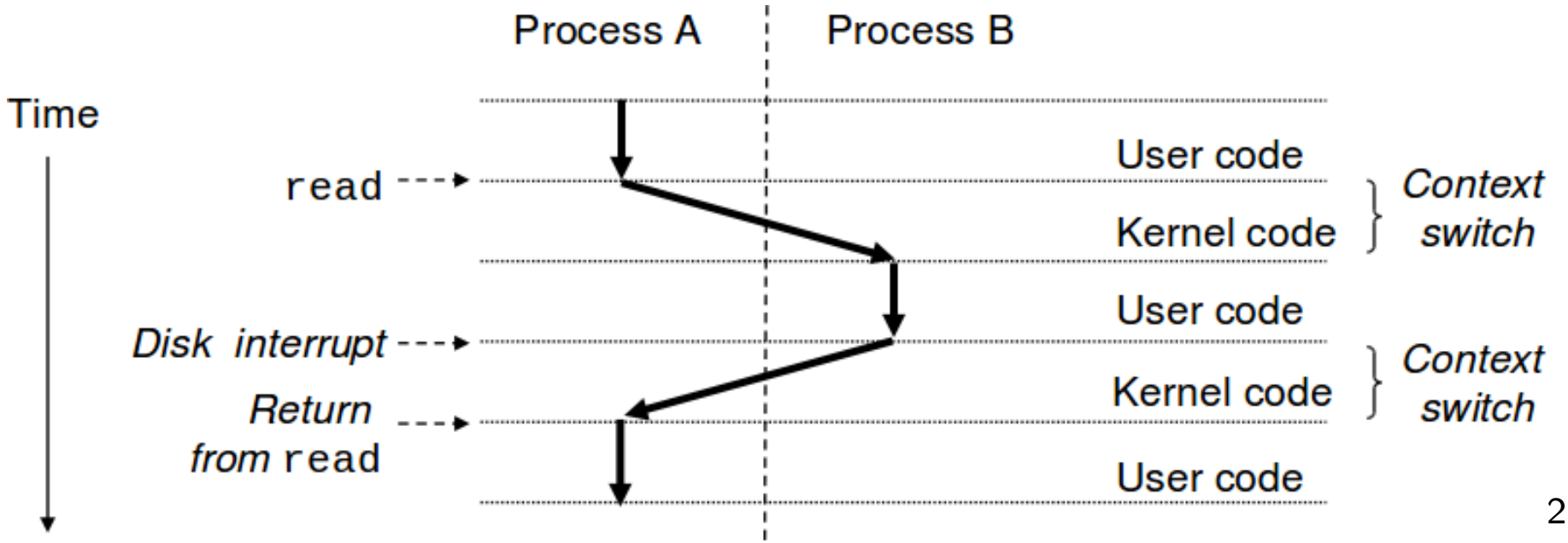


[1] *Fonte: msdn.microsoft.com*

Troca de contexto

- Quando o SO transfere o controle do processo atual para algum novo processo, ele executa uma **troca de contexto**:
  1. salva o contexto do processo atual
  2. restaura o contexto do novo processo
  3. passa o controle para o novo processo
- O novo processo retoma a sua execução exatamente do ponto onde ele parou anteriormente

Troca de contexto



[2]

[2] *Fonte: http://csapp.cs.cmu.edu*

Threads

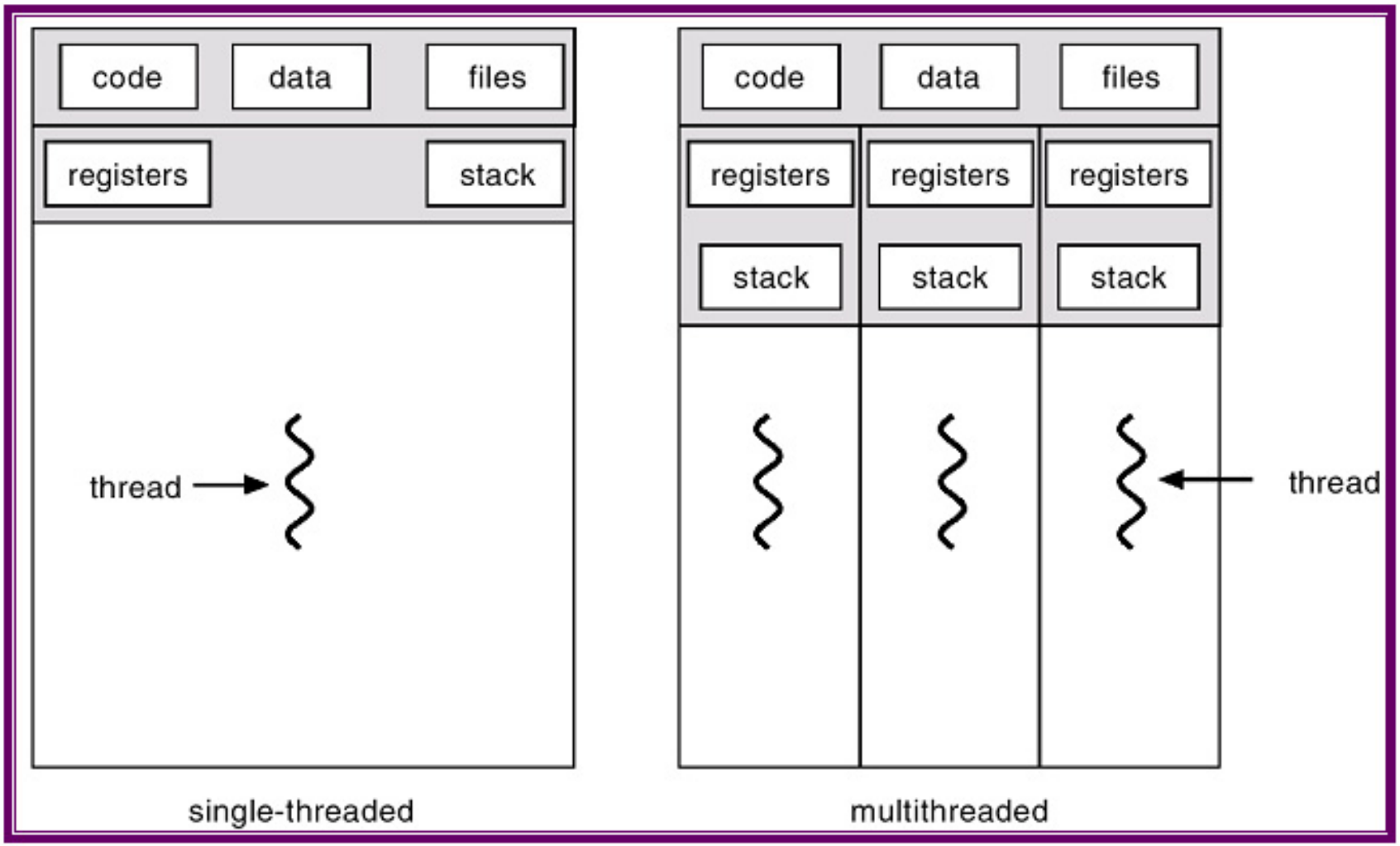
- Um processo pode consistir de **várias unidades de execução** chamadas *threads*
- Cada uma executa dentro do contexto do processo e **compartilha o mesmo código e dados globais com as outras threads**

A troca de contexto entre elas é menos custosa do que a troca de contexto entre processos

Threads

- Uma **thread** é uma unidade básica de uso da CPU (escalonada pelo processador) e compreende:
  - **um identificador da thread, um conjunto de registradores e uma pilha**
- Compartilha com outras threads do mesmo processo:
  - **seção de código e de dados, arquivos abertos, conexões de rede e sinais**

Threads dentro de processos



[3] *Fonte: www.csc.villanova.edu*

Processos versus Threads

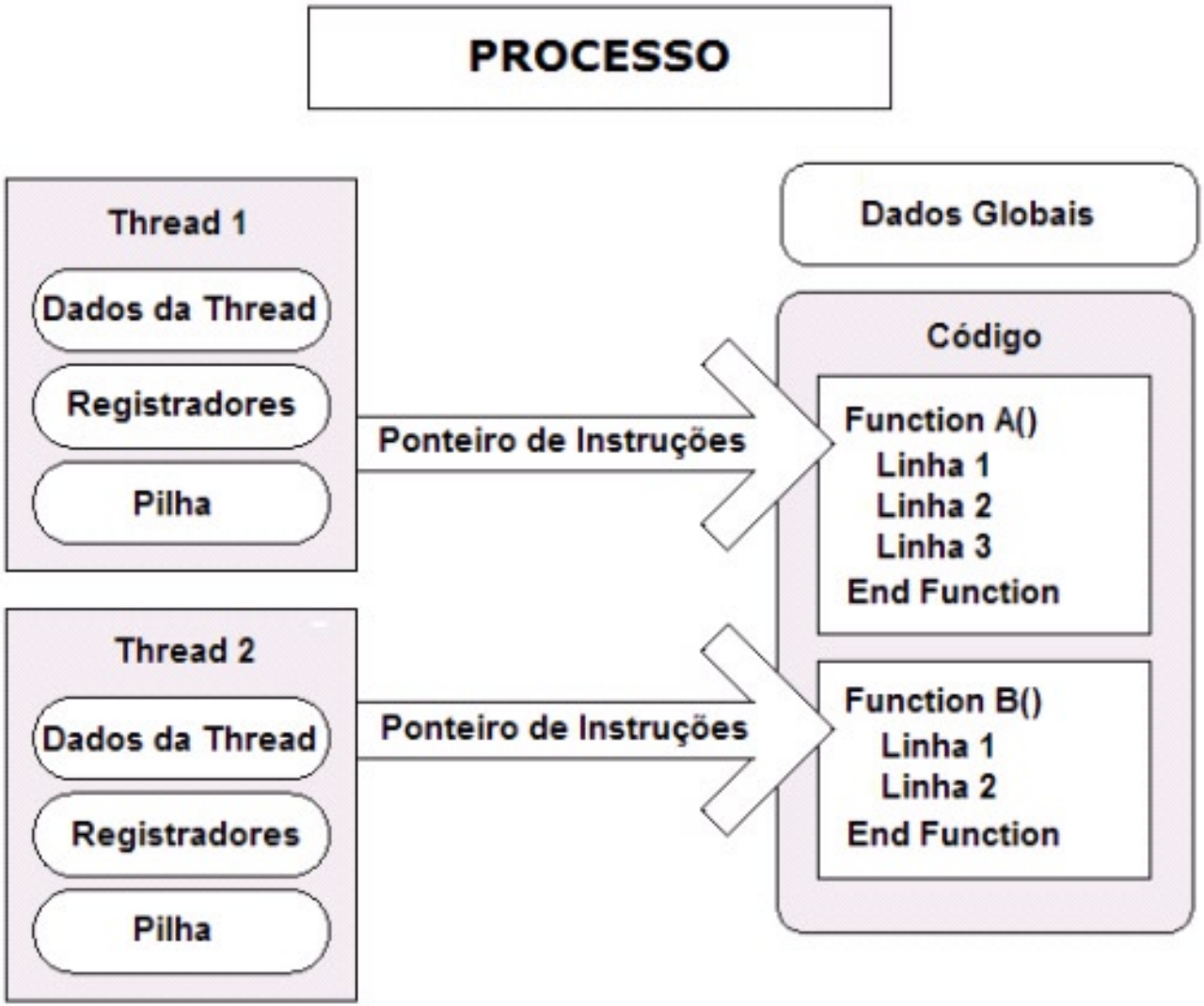
Processos

Com base na abstração de **processo**, podemos ter sistemas onde vários programas executam ao mesmo tempo

Threads

Com base no conceito de **threads**, podemos ter **vários fluxos de controle executando dentro de um único processo**

Programação multithreading

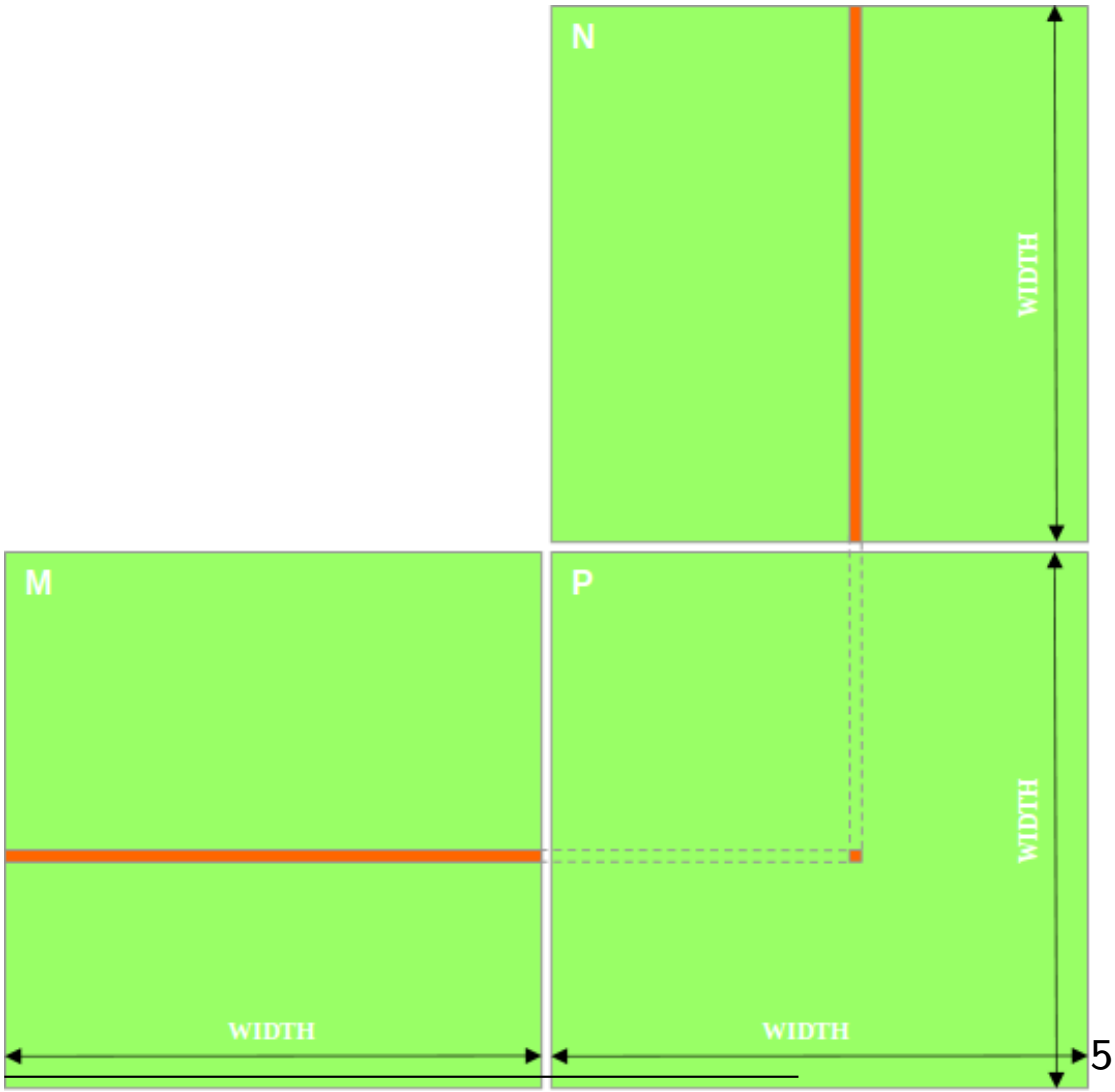


[4] *Fonte: msdn.microsoft.com*

Benefícios de threads

- Menos tempo para criar uma thread do que um processo filho
- Menos tempo para terminar uma thread do que um processo
- Menos tempo para trocar o contexto entre threads do mesmo processo
- Mais eficiência de comunicação através do uso de **memória compartilhada** dentro de um mesmo processo

PASSO 4: dividindo a aplicação entre threads...



[5] *Programming Massively Parallel Processors: A Hands-on Approach, 2010 David B. Kirk/NVIDIA Corporation and Wen-mei Hwu. Elsevier Inc.*

Exemplos de aplicações multithreading

- **Execução em background**: em aplicações com interface visual, uma thread pode ser responsável por exibir os menus e capturar os eventos de entrada e outra pode ser responsável por executar os comandos e atualizar a interface
- Normalmente **melhora a percepção de velocidade da aplicação**, permitindo que o programa apresente os próximos comandos enquanto o comando anterior ainda está sendo executado

Exemplos de aplicações multithreading

Processamento assíncrono: elementos assíncronos do programa em threads distintas, *ex., uma thread é responsável por periodicamente fazer um backup da aplicação enquanto outra thread é responsável pelo programa principal*

Exemplos de aplicações multithreading

Estrutura modular: Programas que envolvem uma variedade de atividades ou uma variedade de fontes e destinos de entrada e saída pode ser mais fácil de projetar e implementar usando threads

Exemplos de aplicações multithreading

Sobreposição de processamento e comunicação: um processo com várias threads pode computar um lote de dados enquanto lê o próximo lote de um dispositivo

Aplicações Web: o uso de várias linhas de execução garante que operações rápidas (ex., exibição de texto) não precisem esperar por operações mais lentas (ex., exibição de imagens)

Custos associados à programação concorrente

Parâmetros do sistema ou biblioteca de thread

1. qual é o custo de criar uma thread?
2. qual é o custo de manter uma thread bloqueada? (esperando pelo processador)
3. qual é o custo da troca de contexto entre threads?

A aplicação concorrente pode gastar mais tempo de execução do que a aplicação sequencial dependendo do projeto da solução e do ambiente disponível

Tarefa de casa :-)

...para o próximo lab...

Implementar uma versão sequencial e outra concorrente de multiplicação de matrizes em C

Referências bibliográficas

- *Computer Systems - A Programmer's Perspective* (**Cap.1**)