Symmetric Multiprocessing
Simultaneous Multithreading
Paralelismo ao nível dos dados

Luís Nogueira

luis@dei.isep.ipp.pt

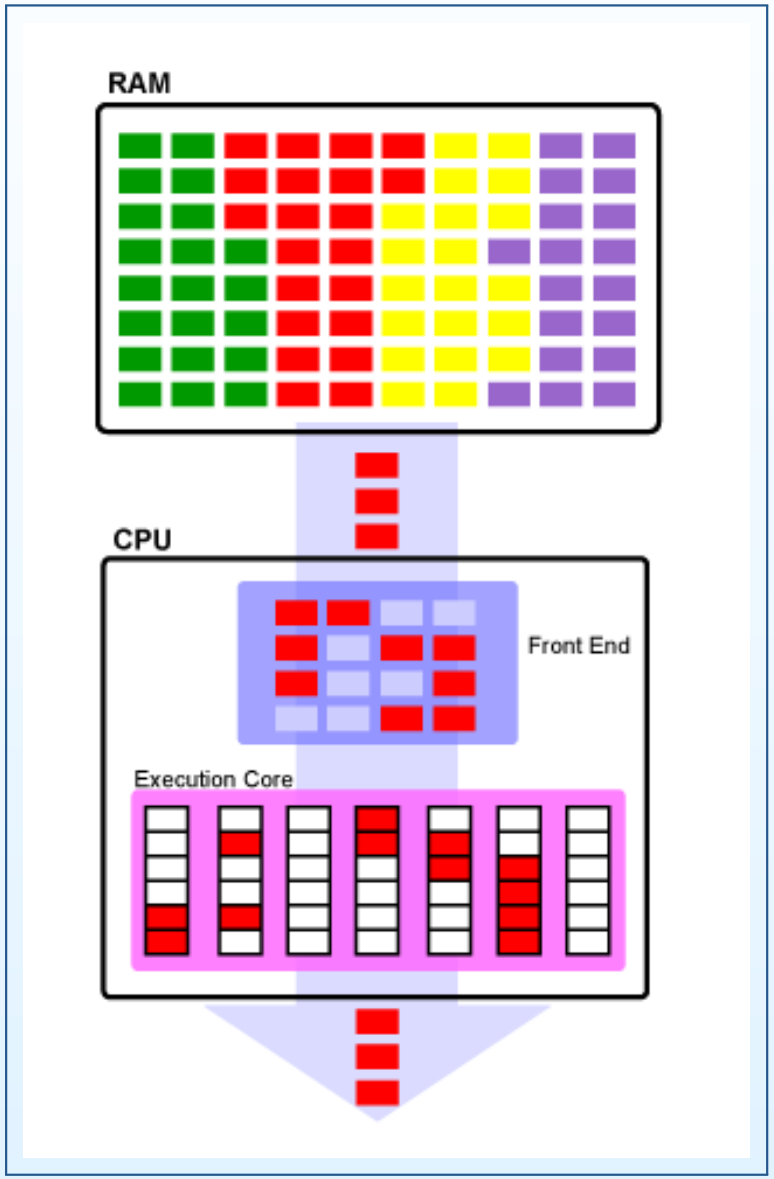
Departamento Engenharia Informática

Instituto Superior de Engenharia do Porto

Fraco paralelismo dos programas

- Exploração do paralelismo através de execução super-escalar é limitada
  - Maioria dos programas não possuem paralelismo suficiente
- Distribuição dinâmica e optimização do compilador em conjunto
  - Em média apenas 2 instruções por ciclo em programas “normais”
- Resultado
  - Unidades funcionais sub-aproveitadas
  - Performance máxima do CPU não é atingida

Single Threaded CPU



Maximizar utilização do hardware

- Programas são representados pelo S.O. por processos
- Associado a cada processo existe um contexto
  - Descreve estado actual da execução do processo (registos, PC, etc)
- Cada processo possui pelo menos uma thread
  - Fluxo de execução com o seu contexto local
- Ideia: Executar mais do que um processo/thread simultaneamente
- Objectivo: Aproveitar unidades funcionais inactivas

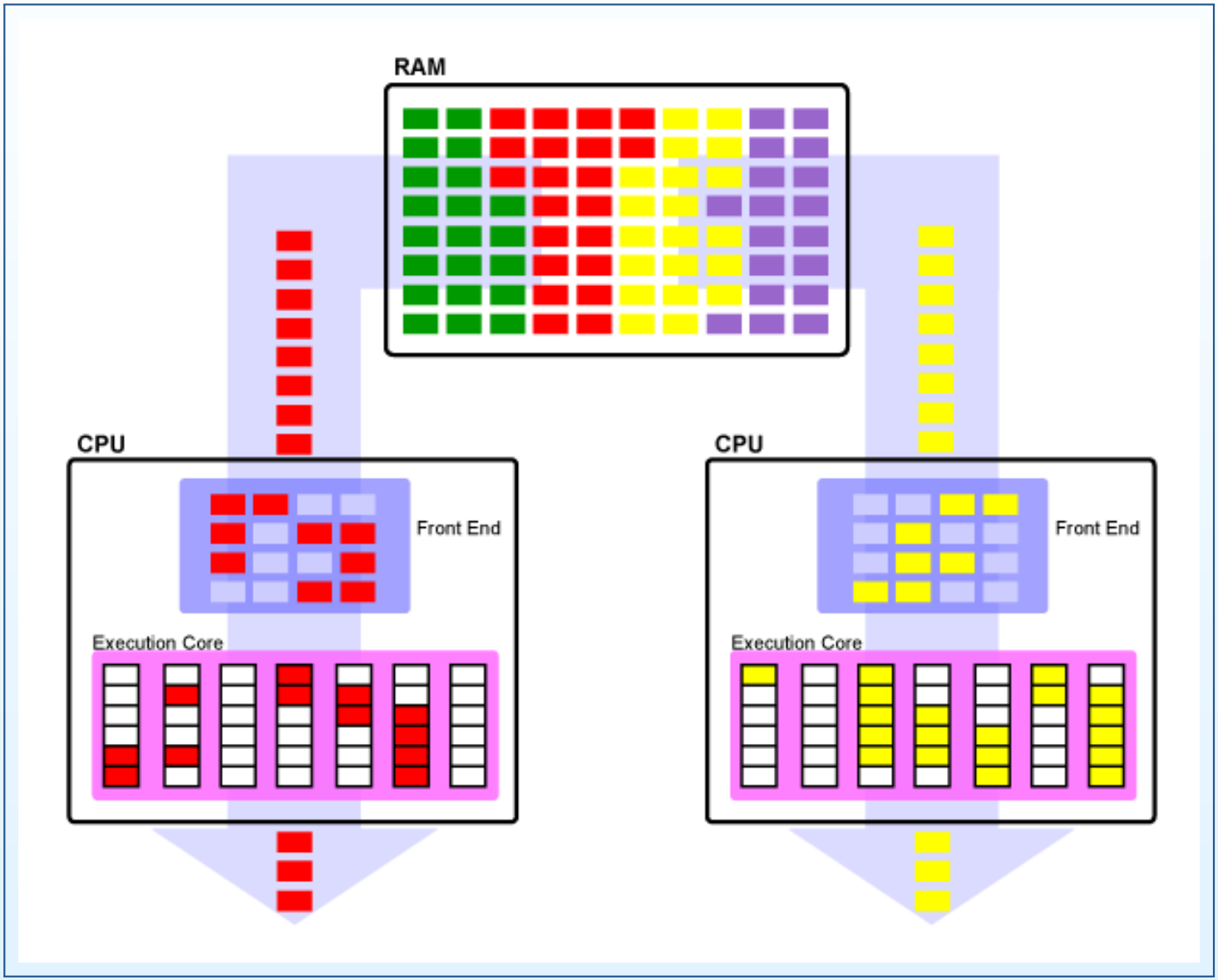
Symmetric Multiprocessing (SMP)

- Executar cada processo/thread num CPU diferente
  - Foco: processadores multi-core
  - 2 ou mais CPUs idênticos ligados a uma memória partilhada
- Suporte para SMP tem de ser dado pelo S.O.
  - Caso contrário CPUs adicionais permanecem inactivos
- Escalonamento do S.O. divide processos pelos CPUs
  - Maior tempo de execução disponível para os processos
  - Menor tempo de espera pelo time slice

Symmetric Multiprocessing (SMP)

- Continuam a existir recursos comuns
  - Acessos resolvidos por exclusão mútua
- SMP não optimiza forma como os programas usam o hardware de cada CPU
- Multiplica tempo de execução para os processos
  - Assim como o desperdício de hardware por ciclo de relógio!

Symmetric Multiprocessing (SMP)



Simultaneous Multithreading (SMT)

- Explorar paralelismo dentro dos processos
  - Maximizar utilização do CPU
- Apenas um CPU físico
  - Fornece dois ou mais processadores lógicos ao S.O.
- Exige duplicação apenas dos componentes que armazenam contexto dos processos
  - Pequena parte do hardware do CPU
  - Unidades funcionais partilhadas pelas threads

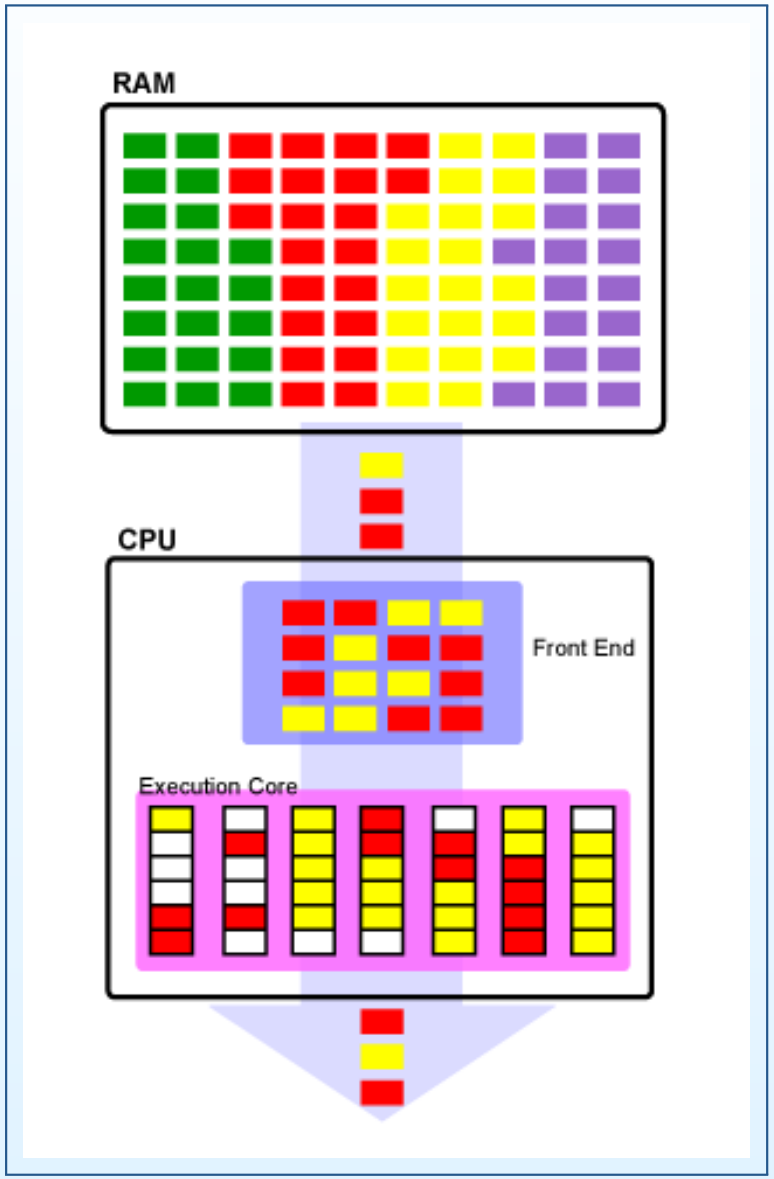
Simultaneous Multithreading (SMT)

- Processador executa várias threads simultaneamente
  - Idealmente resultando na maximização da utilização das unidades funcionais
  - Aumentando IPC $\rightarrow$ aumenta performance
- Assume que existem vários programas em execução
  - Ou que os programas possuem diversas threads
- Performance depende das aplicações
  - Melhora significativamente rendering 3D e bases de dados
  - Pode ser pior em aplicações sensíveis a dados na cache

Simultaneous Multithreading (SMT)

- Técnica proposta em 1972 por Leonard Shar
  - Mais de 30 anos para evolução dos semicondutores tornar técnica comercialmente viável
- Pentium 4 foi o primeiro processador comercial a usar SMT
  - Intel denomina técnica como hyper-threading
- IBM POWER5
  - Processador dual-core com SMT
  - Permite atribuir prioridades às threads
  - SMT ligado e desligado dinamicamente para lidar com situações em que SMT prejudica performance

Simultaneous Multithreading (SMT)



SMP vs SMT

- Comparação dos esquemas SMP e SMT revela
  - Teoricamente a mesma quantidade de trabalho pode ser produzida por ambas as técnicas
  - Mas com menor quantidade de hardware em SMT
- Threads em SMT partilham apenas um processador
  - Acesso partilhado pode atrasar execução de determinadas threads
  - Restrições na complexidade dos CPUs impõe um limite no nº de threads em simultâneo

Arquitecturas Vectoriais

- Explorar paralelismo ao nível dos dados
  - Executar instrução sobre conjunto de dados
- Primeiros protótipos em supercomputadores
  - Cray, CDC, Convex
- Actualmente qualquer PC efectua cálculo numérico intensivo
  - Exemplo: processamento multimédia

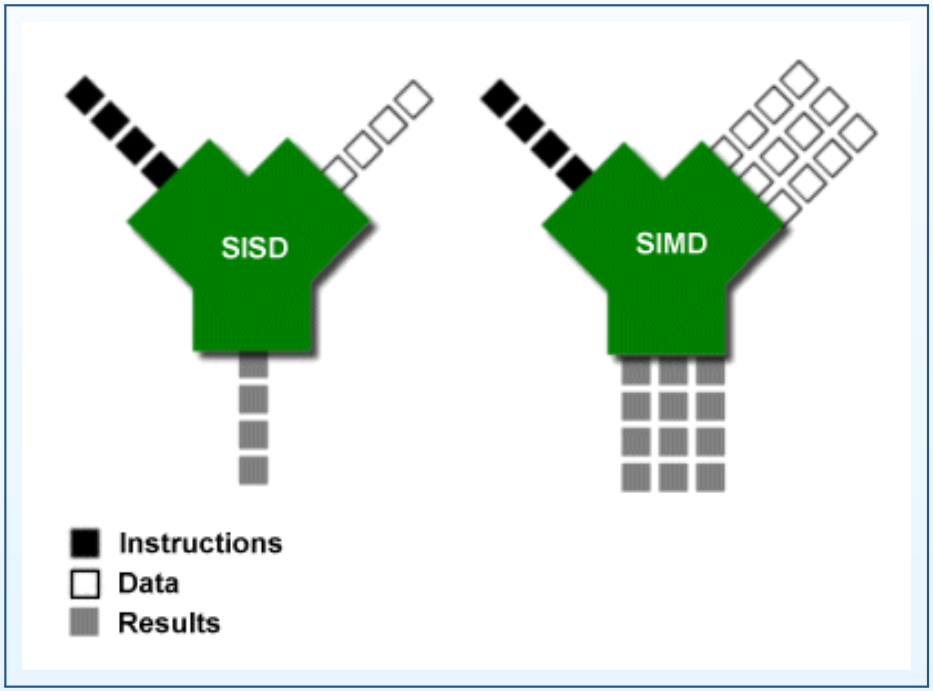
Single Instruction Multiple Data (SIMD)

C	Assembly
`double x[64],y[64];`	`add z[0-15],x[0-15],y[0-15]`
`double z[64];`	`add z[16-31],x[16-31],y[16-31]`
`for(i=0;i<64,i++)`	`add z[32-47],x[32-47],y[32-47]`
`z[i]=x[i]+y[i];`	`add z[48-63],x[48-63],y[48-63]`

- Processador vectorial $\rightarrow$ 4 iterações
  - `add` aplicada a blocos de 16 valores
  - Explora paralelismo ao nível dos dados (SIMD)
- Processador escalar $\rightarrow$ 64 iterações
  - Single Instruction Single Data (SISD)
  - Explora paralelismo ao nível das instruções

Single Instruction Multiple Data (SIMD)

SISD vs SIMD



Single Instruction Multiple Data (SIMD)

- Registos e unidades vectoriais especializadas
  - Vectores mantidos em registos especiais no CPU
  - Instruções aplicadas sobre registos vectoriais
- Vector como unidade básica de computação
  - Em oposição a valores individuais nas arquitecturas escalares
- Aumenta a performance
  - Aplicações onde paralelismo de dados é fácil de obter

Single Instruction Multiple Data (SIMD)

- Processador SIMD puro não é suficientemente flexível
  - Difícil executar código generalista
- Actualmente SIMD usado em ISAs especializadas
  - MMX/SSE, 3DNow!, AltiVec
- Exige elevada largura de banda para a memória
- Muito difícil para um compilador usar SIMD em código “normal”
  - Programas tiram proveito se escritos em ISA específica