Universidade de São Paulo - São Carlos,SP
Instituto de Ciências Matemáticas e de Computação

**USP - ICMC - SSC**
**SSC 0510 - Informática - 2o. Semestre 2009**

# Disciplina de Arquitetura de Computadores

**Prof. Fernando Santos Osório**
**Email: fosorio [at] { icmc. usp. br , gmail. com }**
**Página Pessoal: http://www.icmc.usp.br/~fosorio/**

**Estagiário PAE Maurício Dias - Email: acdias29 [at] yahoo.com.br**

**Material on-line: COTEIA - http://coteia.icmc.usp.br**

*Aula 10*





## Aula 10 – Arquiteturas Especiais

**Conteúdos Abordados:**

1. **Arquiteturas Paralelas**
2. **Arquiteturas Avançadas:**
   **Processador Super-Escalares**
   **Processador VLIW**
   **Processador Super-Pipeline**
   **Processadores Vetoriais**
   **Processadores Multi-Core**

>> **Discussão: Trabalho 2**

## Arquiteturas Paralelas

### Classificação de Flynn (1972)

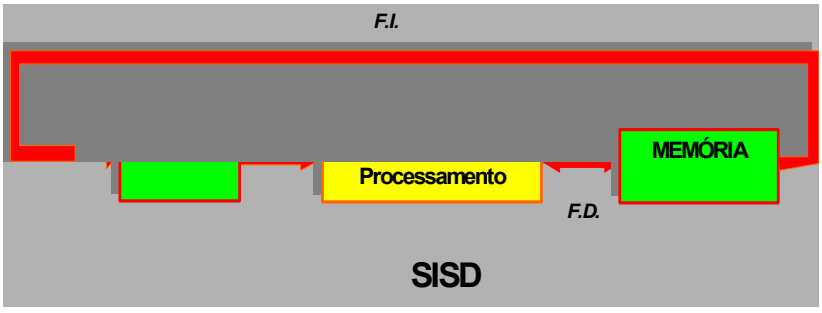
| Fluxo de instruções \ Fluxo de dados | Único | Múltiplo |
|---|---|---|
| Único | **SISD** *Single Instruction Single Data* | **SIMD** *Single Instruction Multiple Data* |
| Múltiplo | **MISD** *Multiple Instruction Single Data* | **MIMD** *Multiple Instruction Multiple Data* |





## Arquiteturas Paralelas

**SISD:** o computador consiste de uma unidade de processamento que recebe um fluxo simples de instruções e opera sobre um simples fluxo de dados;

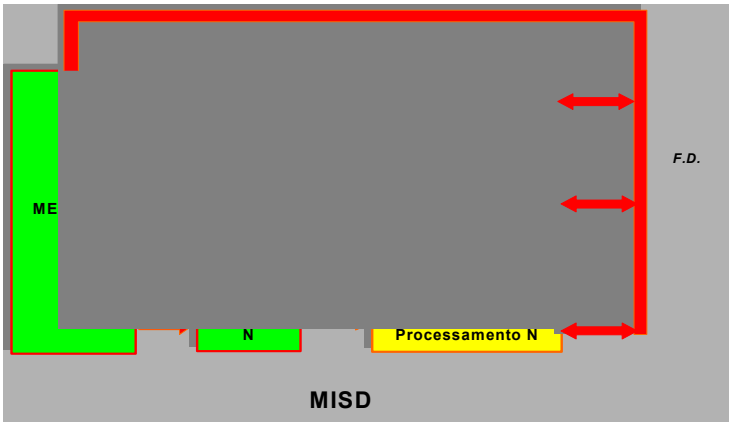



Ex.: computadores von Neumann (o usual)


Contribuição/Slides: Prof. Navarro / USP

## Arquiteturas Paralelas

**MISD:** N processadores, cada um com sua unidade de controle e unidade de processamento, dividem uma mesma memória e executam diferentes instruções sobre o mesmo dado



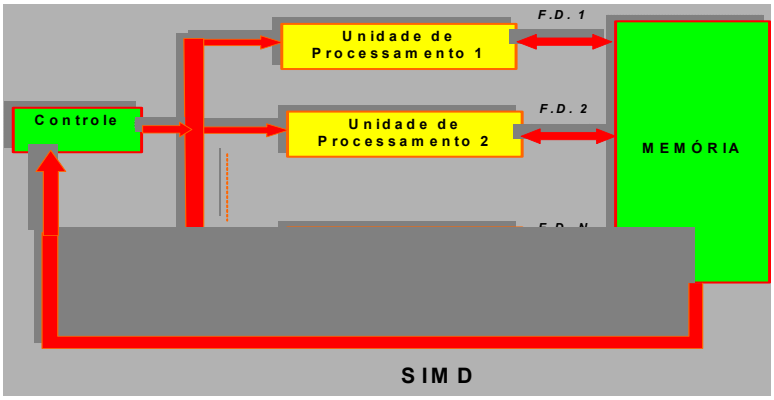

Ex.: arquitetura difícil de ser encontrada. Poderia ser utilizada para aplicar diferentes algorítmos em um mesmo dado.





## Arquiteturas Paralelas

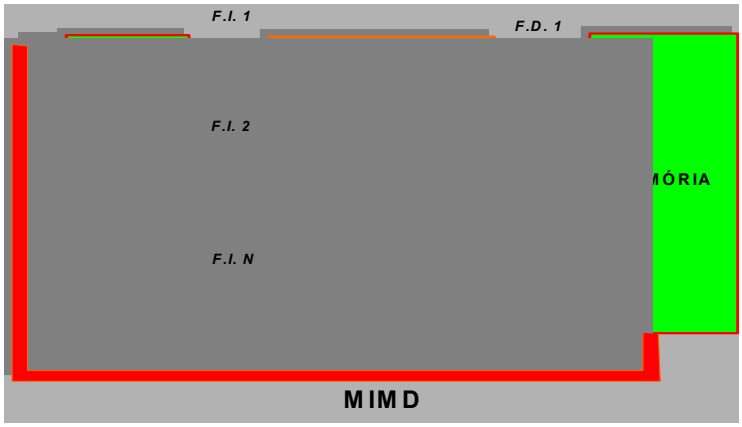
**SIMD:** o computador consiste de uma unidade de controle e N unidades de processamento. Todo o processamento esta sobre o controle de um único fluxo de instruções mas opera sobre N fluxos de dados;




Ex.: processadores vetoriais.

## Arquiteturas Paralelas

**MIMD:** consiste de N processadores distintos, controlados por N fluxos de instruções e operando sobre N fluxos de dados.




Ex.: multiprocessadores e multicomputadores.



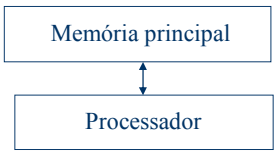


## Arquiteturas Paralelas

### Modelos de acesso à memória

Um computador convencional consiste de um processador executando um programa armazenado na memória:

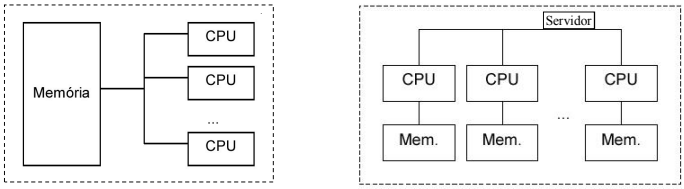
Instruções para o processador
Dados para ou do processador



## Arquiteturas Paralelas

### Modelos de acesso à memória

- Cada lugar da memória possui um endereço que inicia em 0 e vai até $2^n$ - 1, onde *n* é o número de bits do endereço
- Em computação paralela, pode-se ter:
  - memória compartilhada (multiprocessadores)
  - memória distribuída (multicomputadores)



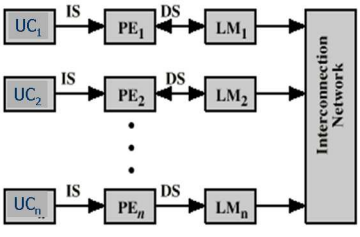




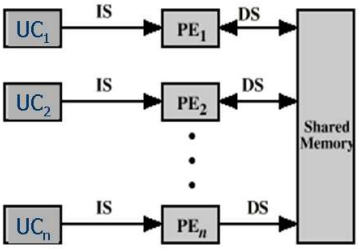

## Arquiteturas Paralelas

### MIMD

**Memória Distribuída**




**Memória Compartilhada**



## Arquiteturas Paralelas

### MIMD com Memória Compartilhada

- Programação:
  - Linguagens de programação paralela
    - Construções e instruções paralelas permitem declarações de variáveis compartilhadas e seções paralelas de código
    - Compilador responsável pela geração do código final executável
    - Usuário é responsável pela sincronização
  - Threads
    - Seqüências de código escritas em alto nível para processadores individuais que podem acessar localidades compartilhadas

Exemplos: SMP (Symetric MultiProcessors)
NUMA (NonUniform Memory Access)





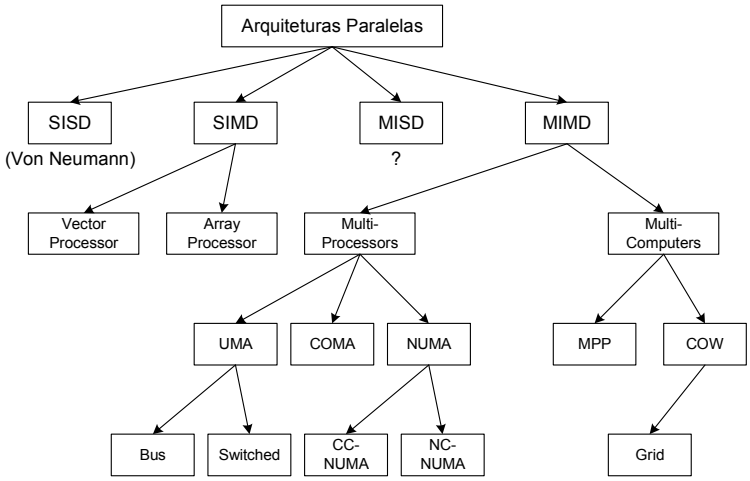
## Arquiteturas Paralelas

### MIMD com Memória Distribuída

- Programação:
  - Bibliotecas com rotinas para passagem de mensagens que são ligadas a programas seqüenciais convencionais são bastante utilizadas;
  - Problema dividido em um número de tarefas que se comunicam;
  - MPI é um protocolo independente de linguagem usado para programar computadores paralelos.

"Message Passing Interface (MPI) is a specification for an API that allows many computers to communicate with one another.
It is used in computer clusters and supercomputers."

## Arquiteturas Paralelas






Fonte: A. Tanenbaum
Structured Computer Organization

UMA: Uniform memory access / NUMA: Nouniform memory access
COMA: Cache-Only Memory Architectures / CC - Cache Coherent



## Arquiteturas Avançadas

**Arquiteturas Avançadas:**

- **Processador Super-Escalar, VLIW, Super-Pipeline**
- **Processador Vetorial**
- **Processador Multi-Core**

**Paralelismo:** ***No nível de instrução***

- **Processadores Escalares**
- **Processadores Super-escalares**
- **Processadores Pipelined**
- **Processadores Super-pipelined**
- **Processadores VLIW (*Very Large Instruction Word*)**



Fonte: Sarita – USP/ICMC

## Processadores Super-escalares

- Processadores Escalares
  - Uma instrução por ciclo
  - Uma instrução terminada por ciclo
- Processadores Super-escalares
  - Múltiplas instruções consideradas em um único ciclo
  - Múltiplas instruções podem ser terminadas em um ciclo
  - Desenvolvidos como uma alternativa a processadores vetoriais

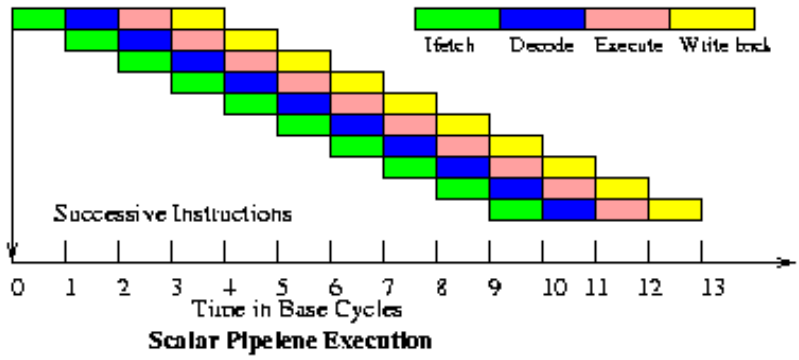






## Processadores Super-escalares

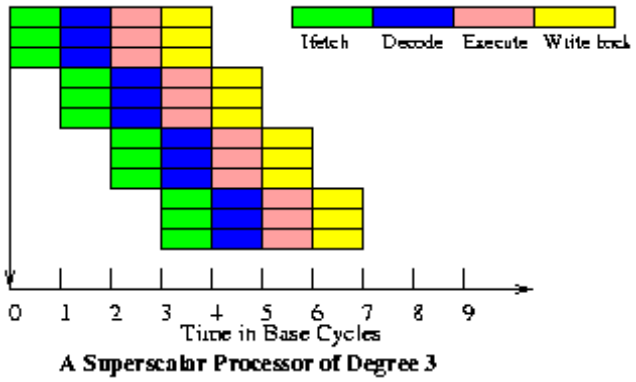
- Processador pipeline de 4 estágios (proc. escalar)




Scalar Pipelene Execution

## Processadores Super-escalares

- Processador Super-escalar
  com grau = 3 e com pipeline de 4 estágios




A Superscalar Processor of Degree 3

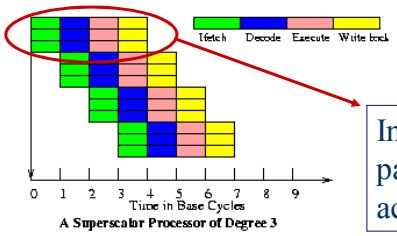






## Processadores Super-escalares

- Um processador super-escalar com grau *m* ***pode*** finalizar até *m* instruções por ciclo.
  - Depende da dependência de dados, conflito por recurso e dos desvios




A Superscalar Processor of Degree 3

Instruções executadas em paralelo: evitar conflito de acesso e escrita de operandos!

## Processadores Super-escalares

### O que significa Super-escalar?

- Instruções comuns
  - Aritméticas, *load/store* e desvios condicionais podem ser iniciados e executados independentemente
- Aplicável igualmente a RISC e CISC
  - Na prática é implementado usualmente em máquinas RISC







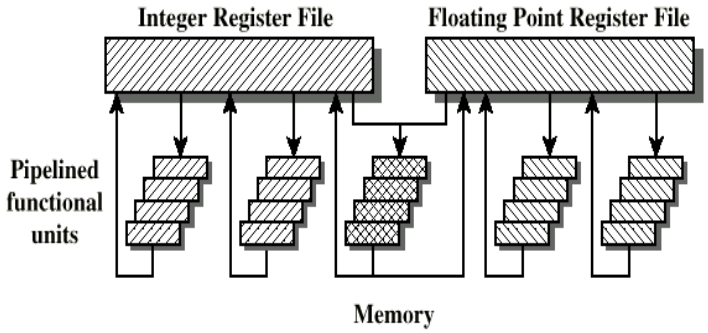
## Processadores Super-escalares

### O que significa Super-escalar?

- Maioria das operações realizadas em valores escalares.
  Valores escalares são aqueles que podem ser representados por um único número.
- Melhora desempenho dessas operações para melhorar o desempenho total.
- Proposta foi aceita rapidamente
  - Máquinas super-escalares comerciais surgiram apenas +/- 2 anos após o surgimento do termo super-escalar
  - Máquinas RISCs comerciais levaram +/- 8 anos

## Organização super-escalar genérica

- Várias unidades funcionais organizadas em *pipeline*
- Ex.: duas operações com inteiros
  duas operações com ponto flutuante
  uma operação de carga/armazenamento







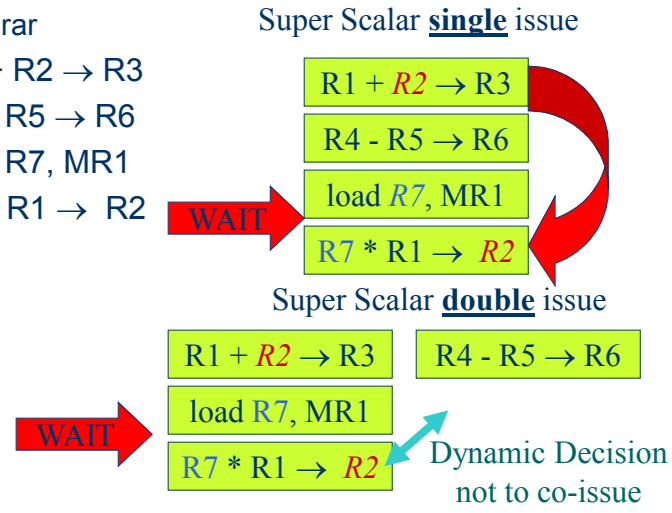



## Processadores Super-escalares: Limitações

- Paralelismo no nível de instrução
  - Nível que as instruções podem ser executadas em paralelo
- Otimizações baseadas:
  - No compilador e em técnicas de hardware
- Limitações intrínsecas:
  - Dependência de dados verdadeira (escrita-leitura)
  - Dependências de desvio
  - Conflitos no uso de recursos
  - Dependência de saída (escrita-escrita)
  - Antidependência (leitura-escrita)

## Processadores Super-escalares: Limitações

- Considerar
  - R1 + R2 → R3
  - R4 - R5 → R6
  - load R7, MR1
  - R7 * R1 → R2





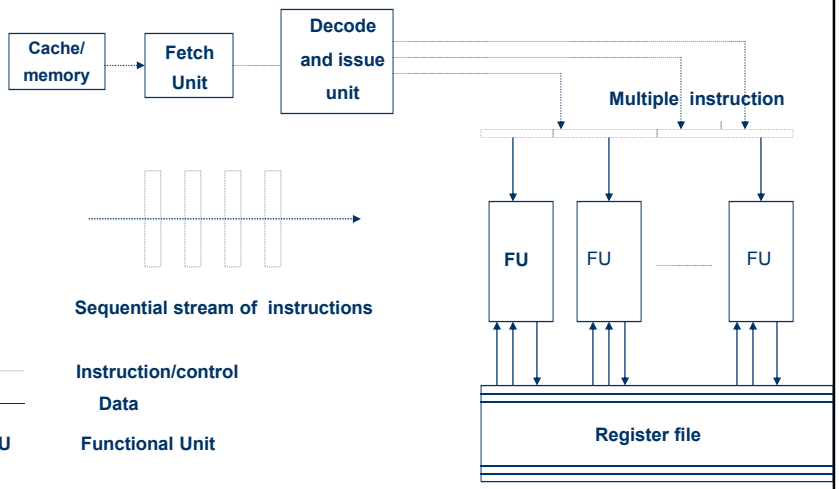





## Processadores VLIW

- VLIW – *Very Large Instruction Word*
- Explora paralelismo em nível de instrução
- Instruções de 128-1024 bits
- Cada instrução consiste de múltiplas instruções independentes
- Diversas unidades funcionais interligadas por um único registrador compartilhado

## Comparação: Super-escalares



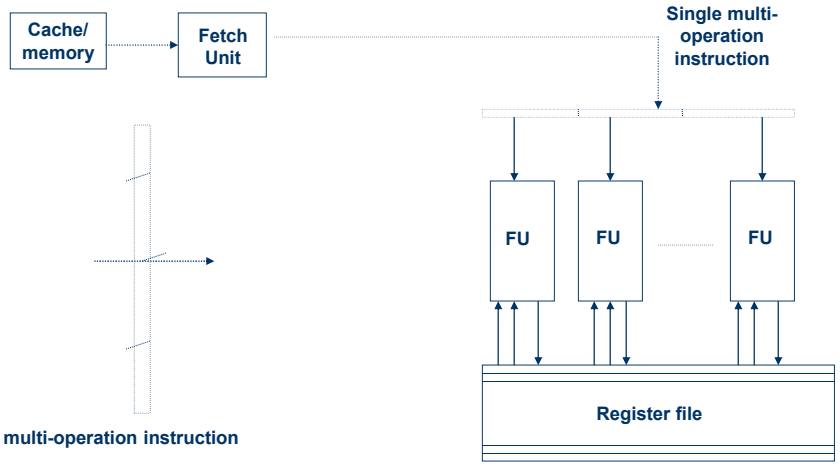



[Ref : Sima et al]





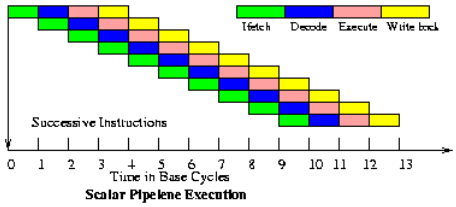
## Comparação: VLIW



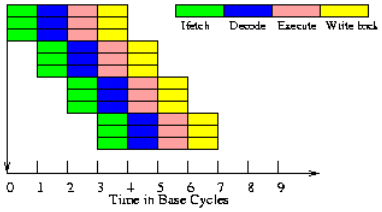



[Ref : Sima et al]

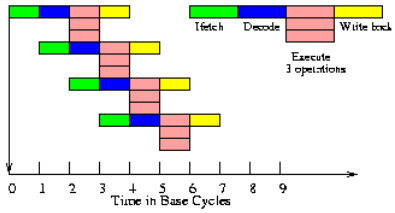
## Comparações: Pipeline, Super-escalar e VLIW




Scalar Pipelene Execution




A Superscalar Processor of Degree 3




VLIW Execution with Degree 3



[Ref : Hwang et al]



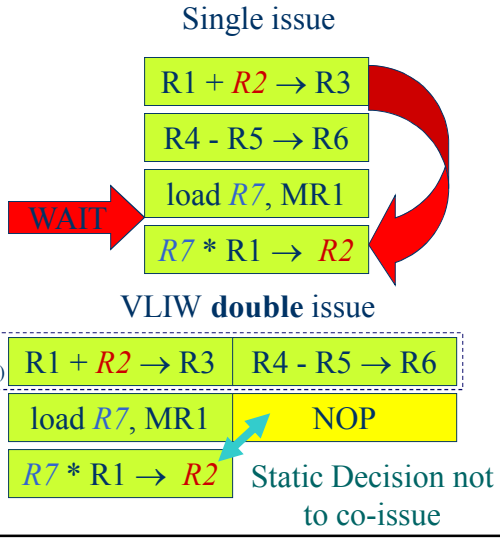


## Processadores VLIW x Super-escalar

- Porque VLIW é menos popular que super-escalar?
  - Compatibilidade entre códigos binários
  - Mesmo compilador pode ser utilizado para escalar e super-escalar
  - Exemplo de Super-escalar: IBM RS/6000
  - Exemplo de VLIW: Crusoe TM 5400

## Processadores VLIW: Limitações

- Considerar
  - R1 + R2 → R3
  - R4 - R5 → R6
  - load R7, MR1
  - R7 * R1 → R2








## Processadores VLIW: Histórico e Processadores

- O termo é criado por J. A. Fisher (Yale) em 1983
  - ELI S12 (protótipo)
  - Trace (Comercial)
- Origem a partir da otimização de micro-código horizontal
- Outro trabalho pioneiro de B. Ramakrishna Rau em 1982
  - Poly cyclic (Protótipo)
  - Cydra-5 (Comercial)
- Desenvolvimentos recentes
  - Trimedia – Philips
  - TMS320C6X – Texas Instruments

## Arquiteturas Super-pipeline

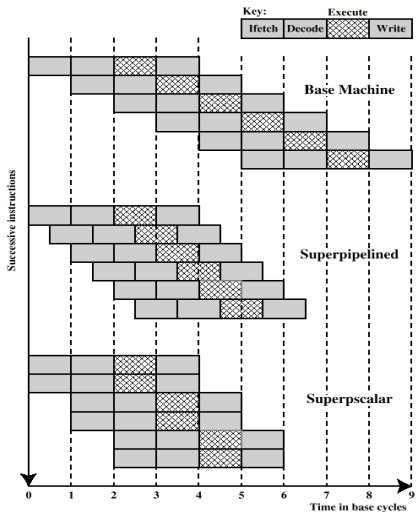
- É um *pipeline* com muitos estágios
- Estágios necessitam tempos de ciclo menores
  - Normalmente menos que a metade
- Velocidade interna de *clock* duplicada
  - Executa duas "atividades“ por ciclo de *clock* externo
- Super-escalar permite executar a busca em paralelo







## Arquiteturas Super-pipeline x Super-escalar



## Arquiteturas Avançadas

### Pontos Importantes:

- Processador super-escalar:
  - Emprega vários *pipelines* de instrução independentes
  - Cada pipeline com seus estágios, executando instruções diferentes simultaneamente
  - Novo nível de paralelismo: diversos fluxos de instrução cada vez
- Processador precisa buscar várias instruções:
  - Instruções próximas que sejam independentes e possam ser executadas em paralelo (ao mesmo tempo)
  - Problemas com a dependência de dados
  - Identificadas as dependências, execução pode ser feita fora de ordem



USP - SSC-510 - Informática T1 / 2009-2
Arquitetura de Computadores
Prof. Fernando Osório

## Arquiteturas Avançadas

### Arquiteturas paralelas avançadas: Arquiteturas SIMD

- Um único fluxo de instruções , vários fluxos de dados
- Tipos de arquiteturas paralelas SIMD
  - Processadores **Vetoriais**
  - Processadores **Matriciais**

## Arquiteturas SIMD
## Processadores Vetoriais

- Processadores Vetoriais provêm instruções de alto nível sobre vetores de dados, tais como multiplicar, subtrair, somar
- Em máquinas escalares, essas operações são realizadas através de um *loop*
- Em máquinas vetoriais, essas operações (podem) são realizadas em uma única instrução vetorial







## Arquiteturas SIMD
## Processadores Vetoriais

- Vantagens:
  - Redução da quantidade de *fetch* e *decode* de instruções.
  - Não há necessidade de verificação de conflitos de dados, pois as operações entre elementos dos vetores que estão na mesma operação são independentes.
  - Operações vetoriais são atômicas e eliminam a sobrecarga gerada pelos saltos condicionais e comparações necessárias ao controle de repetições.
  - As operações entre elementos podem ser paralelizadas ou executadas em pipeline.
  - Como há uma quantidade menor de instruções por programa, há uma quantidade menor de falha no *cache* de instruções.

## Arquiteturas SIMD
## Processadores Vetoriais

Exemplo:

Programa em Fortran

```
            DO 100 I = 1, N
                    A(I) = B(I) + C(I)
100                 B(I) = 2 * A(I+1)
```

> Soma vetores B(i) + C(i)  a atribui em A(i)
> B(i) é obtido multiplicando 2 * A(i+1)
> Laço repetido para os N elementos dos vetores A(i), B(i), C(i).







## Arquiteturas SIMD
## Processadores Vetoriais

```
            DO 100 I = 1, N
                    A(I) = B(I) + C(I)
100                 B(I) = 2 * A(I+1)
```

- Exemplo

  Programa utilizando instruções **escalares**

```
    INITIALIZE I = 1
10  READ B(I)
    READ C(I)
    ADD B(I) + C(I)
    STORE A(I)                  // B(I) + C(I)
    READ A(I + 1)
    MULTIPLY 2 * A (I + 1)
    STORE B(I)                  // 2* A(I + 1)
    INCREMENT I <I+ 1
    IF I <= N GO TO 10
    STOP
```

## Arquiteturas SIMD
## Processadores Vetoriais

```
      DO 100 I = 1, N
         A(I) = B(I) + C(I)
100      B(I) = 2 * A(I+1)
```

- Exemplo
  - Programa utilizando instruções **vetoriais**

    TEMP(1:N) = A(2:N + 1)
    A(1:N) = B(1:N) + C(1:N)
    B(1,N) = 2 * TEMP(1:N)

  - Onde A(1:N) é o vetor A
  - Necessidade do vetor TEMP para armazenamento do vetor A antes da atualização

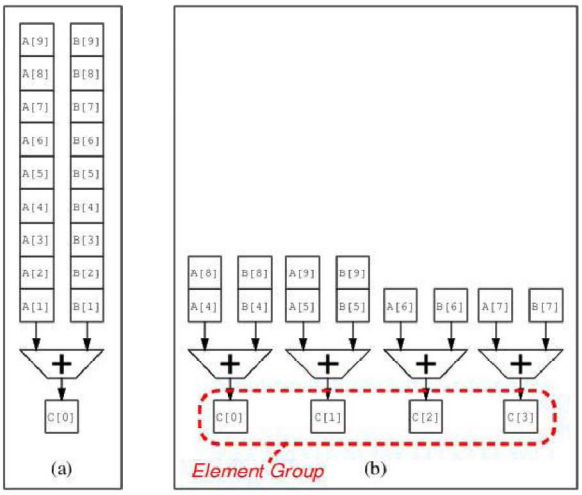






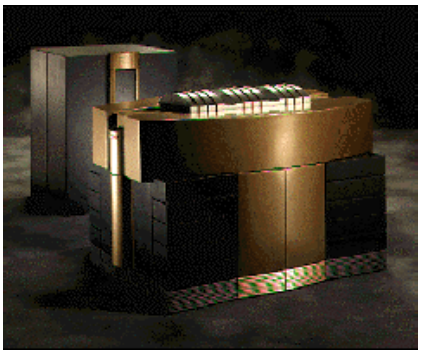
## Arquiteturas SIMD
## Processadores Vetoriais

Para melhorar ainda mais o desempenho, pode-se aumentar o número de ULAs



## Arquiteturas SIMD
## Processadores Vetoriais: Arquiteturas Comerciais

- Supercomputador Vetorial Cray T90









## Paralelismo no nível de tarefas

- Paralelismo a nível de *thread*
  - Considera aplicações com múltiplas *threads*
  - Várias opções:
    - *Multithreading (MT)*
    - *Superthreading*
    - *Simultaneous Multithreading (SMP)* ou *Hyperthreading*
    - Múltiplos Cores

## Multithread

- Técnica para diminuir as perdas associadas com processamento de uma única *thread*
- A técnica é chamada de time-slice multithreading ou superthreading
- O processador que utiliza essa técnica é chamado processador *multithreaded*
- Processador *multithreaded* é capaz de executar mais de uma *thread* em um instante





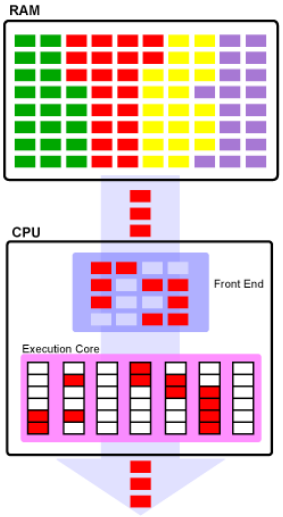


## Multithread

- *Threads* compartilham *suporte em hardware para troca entre threads em execução sem intervenção de software*.
- CPU possui informações sobre os estados associados com cada *thread* (contador de programa, registradores, etc.).
- O hardware também deve conter um mecanismo para o escalonamento de *thread*, fetch e decodificação de instruções.

## CPU com uma thread x Multithread

- RAM: 4 programas em execução
- Front End: busca até 4 instruções
- Sete pipelines de execução
- Programa vermelho em execução
- Quadrados brancos: estágios vazios
- Programas possuem uma ou mais *threads*



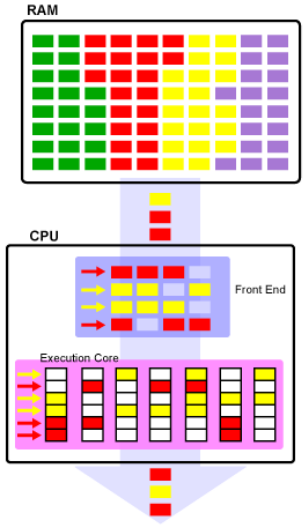







## CPU com uma thread x Multithread

- Número menor de estados perdidos
- Um estágio do pipeline só pode ter instrução de uma *thread*
- Front End – 4 instruções por clock
- Processador – 7 unidades funcionais



## Multi-Core

- Múltiplos núcleos de CPU em um processador
- Execuções simultâneas de tarefas
- Cada núcleo com o seu pipeline
- Cada núcleo com os recursos necessários para execução de seu programa





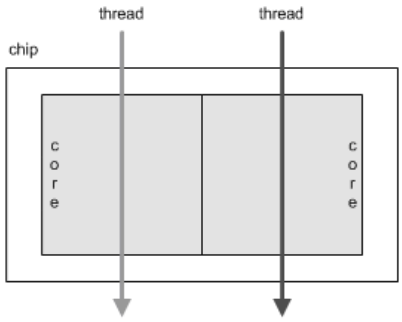


## Multi-Core

- Possibilidade de Múltiplos núcleos
- Intel Dual-Core: 2 núcleos
- Mainframes: diversos núcleos

| | |
|---|---|
| Produced | From 2006 to 2008 |
| Common manufacturer(s) | Intel |
| Max CPU clock | 1.06 GHz to 2.33 GHz |
| FSB speeds | 533 MT/s to 667 MT/s |
| Min feature size | 0.065 µm |
| Instruction set | x86-686 |
| Microarchitecture | P6 (Pentium M) derivative |
| Cores | 1 or 2 |
| Socket(s) | Socket M |
| Core name(s) | Yonah |






Multicore processors:
one chip, multiple cores,
multiple threads

| Intel Pentium Dual-Core processor family | | | | | | |
|---|---|---|---|---|---|---|
| Logo | Desktop | | | Laptop | | |
| | Code-named | Core | Date released | code-named | Core | Date released |
| intel Pentium Dual-Core inside | Allendale | dual (65nm) | Jun 2007 | Yonah | dual (65nm) | Jan 2007 |
| | Wolfdale | dual (45nm) | Aug 2008 | Merom | dual (65nm) | Nov 2007 |

List of Intel Pentium Dual-Core microprocessors



## INFORMAÇÕES SOBRE A DISCIPLINA

**USP - Universidade de São Paulo - São Carlos, SP**
**ICMC - Instituto de Ciências Matemáticas e de Computação**
**SSC - Departamento de Sistemas de Computação**

**Prof. Fernando Santos OSÓRIO**
**Web institucional: http://www.icmc.usp.br/ssc/**
**Página pessoal: http://www.icmc.usp.br/~fosorio/**
**E-mail: fosorio [at] icmc. usp. br ou fosorio [at] gmail. com**

**Disciplina de Arquitetura de Computadores / Informática**
**Estagiário PAE: Maurício A. Dias**
**Web disciplina: COTEIA - Http://coteia.icmc.usp.br**
**> Programa, Material de Aulas, Critérios de Avaliação,**
**> Lista de Exercícios, Trabalhos Práticos, Datas das Provas**