SSC0611
Arquitetura de Computadores

19ª Aula – Arquiteturas Paralelas

Arquiteturas SIMD

Profa. Sarita Mazzini Bruschi

sarita@icmc.usp.br

Arquiteturas SIMD

- Um único fluxo de dados, vários fluxos de intruções
- Tipos de arquiteturas
  - Mais antigas
    - Processadores Vetorais
    - Processadores Matriciais
  - Mais recentes
    - Extensões de conjunto de instruções SIMD para multimídia
    - Unidades de Processamento Gráfico

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 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)
```

Arquiteturas SIMD
Processadores Vetoriais

- Exemplo (continuação)
  - 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

- Exemplo (continuação)
  - Programa utilizando instruções vetoriais

    TEMP(1:N) = A(2:N + 1)

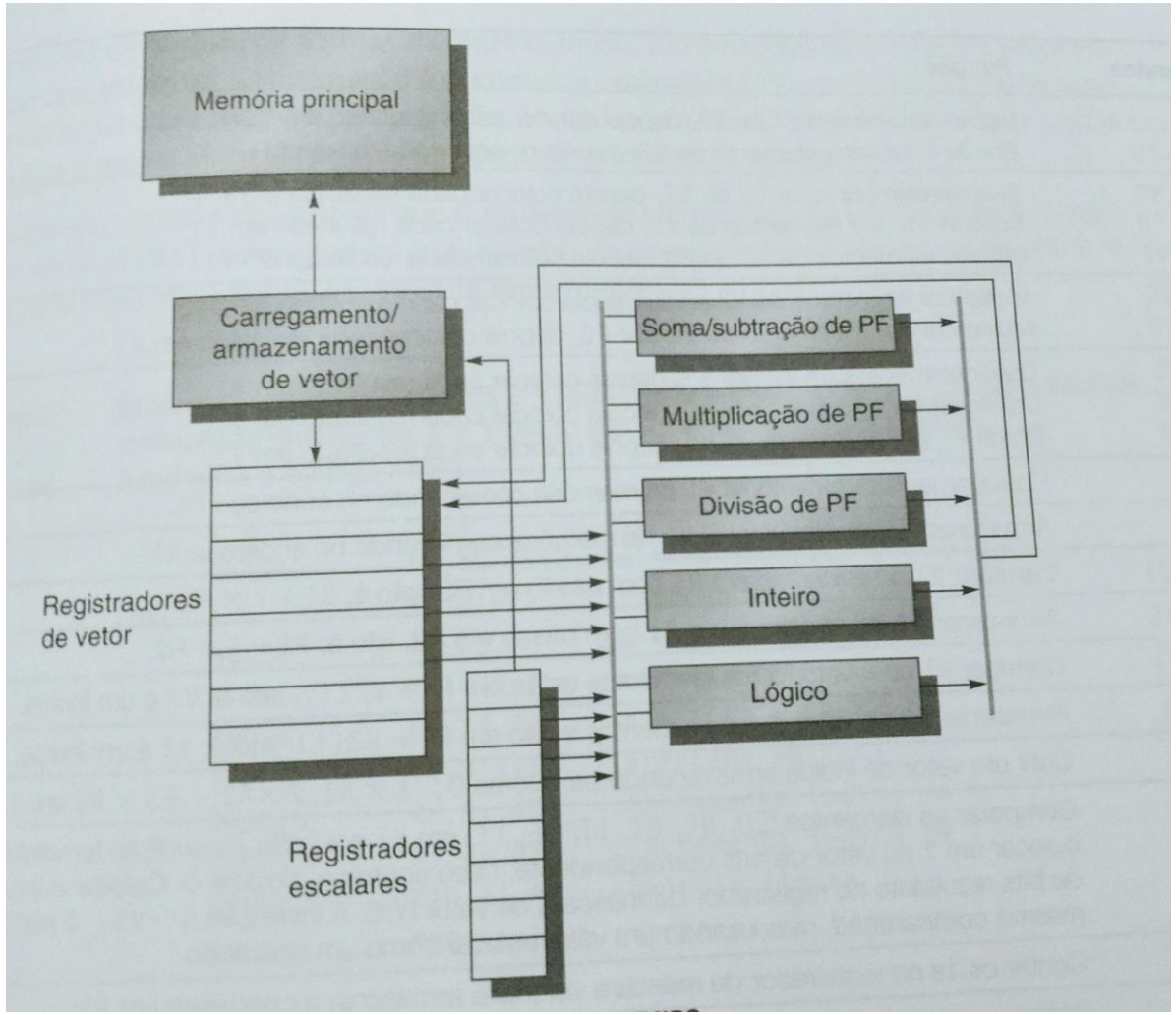
    A(1:N) = B(1:N) + C(1:N)

    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

- Arquitetura VMIPS



Arquiteturas SIMD
Processadores Vetoriais

- Arquitetura VMIPS
  - Operação: Y = a * X + Y

Escalar

```
      l.d     $f0,a($sp)      :load scalar a
      addiu   $t0,$s0,#512    :upper bound of what to load
loop: l.d     $f2,0($s0)      :load x(i)
      mul.d   $f2,$f2,$f0     :a x x(i)
      l.d     $f4,0($s1)      :load y(i)
      add.d   $f4,$f4,$f2     :a x x(i) + y(i)
      s.d     $f4,0($s1)      :store into y(i)
      addiu   $s0,$s0,#8      :increment index to x
      addiu   $s1,$s1,#8      :increment index to y
      subu    $t1,$t0,$s0     :compute bound
      bne     $t1,$zero,loop  :check if done
```

Vetorial

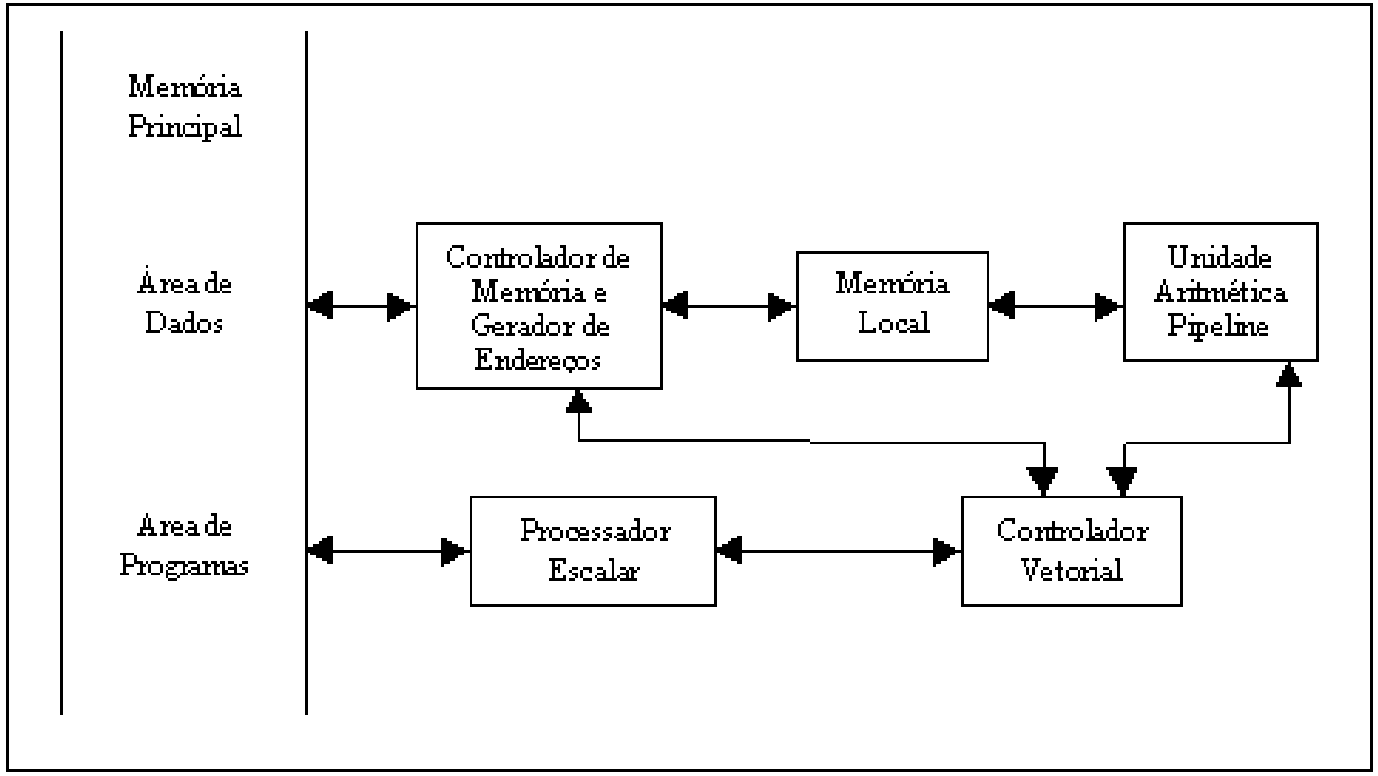
```
l.d     $f0,a($sp)      :load scalar a
lv      $v1,0($s0)      :load vector x
mulvs.d $v2,$v1,$f0     :vector-scalar multiply
lv      $v3,0($s1)      :load vector y
addv.d  $v4,$v2,$v3     :add y to product
sv      $v4,0($s1)      :store the result
```

Arquiteturas SIMD
Processadores Vetoriais

- Os processadores vetorias podem ser de dois tipos:
  - Registrador-Registrador
    - Operações vetorais sobre registradores
    - Somente as operações de *load vector* e *store vector* acessam a memória
  - Memória-Memória
    - Operandos buscados diretamente da memória

Arquiteturas SIMD
Processadores Vetoriais

- Arquitetura de um processador vetorial genérico



Arquiteturas SIMD
Processadores Vetoriais

- Arquitetura de um processador vetorial genérico
  - Banco de registradores vetoriais
  - Gerador de endereço de vetor:
    - Responsável por gerar o endereço de um vetor a ser buscado na memória.
  - Memória
  - Unidades aritméticas *pipeline (Functional units):*
    - Efetuam as operações sobre os vetores.
  - Controlador vetorial *(Vector Data Switch):*
    - Responsável por direcionar um vetor para a unidade

Arquiteturas SIMD
Processadores Vetoriais

- Processador Escalar:
  - Realiza operações sequenciais, como por exemplo, testes de dependências, gerenciamento de E/S e memória, etc
  - Quando identifica uma operação vetorial, encaminha para o Controlador Vetorial
- Endereços são encaminhados ao Controlador de Memória e Gerador de Endereços, onde os operandos são buscados
- A Unidade Aritmética Pipeline executa as instruções vetorais

Arquiteturas SIMD
Processadores Vetoriais

- A memória utiliza bancos entrelaçados (*interleaving*)
  - Associação do endereço com o banco de memória de maneira cíclica
  - Exemplo: endereço 0 no banco 0, endereço 1 no banco 1, ..., endereço N no banco 0, endereço N+1 no banco 1, etc.
  - Diversos bancos podem ser ativados ao mesmo tempo

Arquiteturas SIMD
Processadores Vetoriais

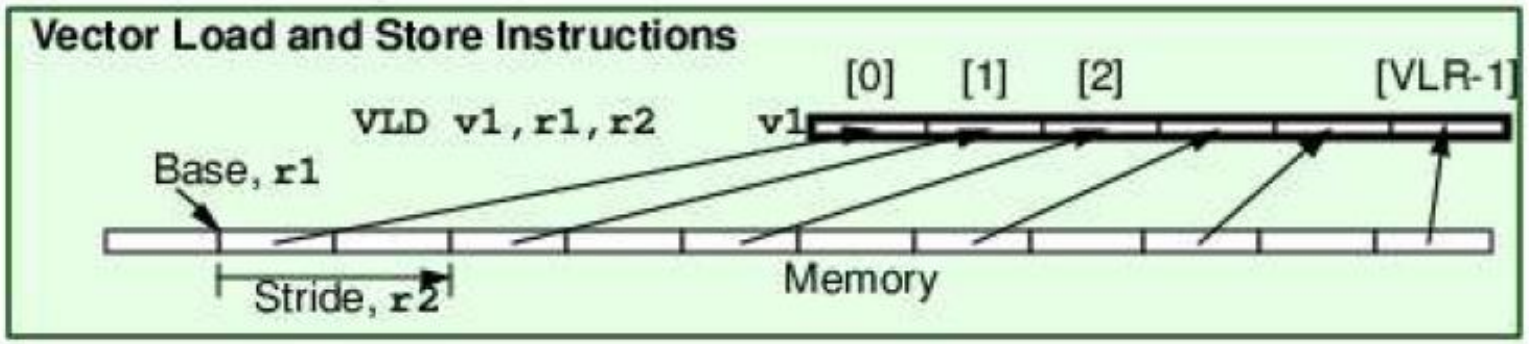
Beat	VAG	ADDR BUS	BANK 0	BANK 1	BANK 2	BANK 3	DATA BUS	DATA SWITCH	VRF
1	X[0]								
2	X[1]	X[0]							
3	X[2]	X[1]	RAS X[0]						
4	X[3]	X[2]	CAS X[0]	RAS X[1]					
5	X[4]	X[3]	DATA X[0]	CAS X[1]	RAS X[2]				
6	X[5]	X[4]	CYCLE	DATA X[1]	CAS X[2]	RAS X[3]	X[0]		
7	X[6]	X[5]	RAS X[4]	CYCLE	DATA X[2]	CAS X[3]	X[1]	X[0]	
8	X[7]	X[6]	CAS X[4]	RAS X[5]	CYCLE	DATA X[3]	X[2]	X[1]	X[0]
9		X[7]	DATA X[4]	CAS X[5]	RAS X[6]	CYCLE	X[3]	X[2]	X[1]
10			CYCLE	DATA X[5]	CAS X[6]	RAS X[7]	X[4]	X[3]	X[2]
11				CYCLE	DATA X[6]	CAS X[7]	X[5]	X[4]	X[3]
12					CYCLE	DATA X[7]	X[6]	X[5]	X[4]
13						CYCLE	X[7]	X[6]	X[5]
14								X[7]	X[6]
15									X[7]

Arquiteturas SIMD
Processadores Vetoriais

- Endereçamento de vetores
  - Três maneiras:
    - Acesso contíguo
      - Os elementos estão contíguos na memória
    - Acesso não contíguo, mas regular
      - Os elementos estão separados por intervalos de tamanho k
    - Acesso esparso ou indexado
      - Os elementos estão em posições arbitrárias na memória e o endereço de cada posição é armazenado na posição correspondente de um vetor auxiliar

Arquiteturas SIMD
Processadores Vetoriais

- Exemplo de uma instrução de carregamento de um vetor



Arquiteturas SIMD
Processadores Vetoriais

- Pipeline Vetorial
  - O tempo de execução de uma instrução vetorial não é o mesmo tempo de uma instrução escalar, ou seja, ADDV não leva a mesma quantidade de ciclos que uma instrução ADD
  - Se isso acontecesse, seriam necessárias *n* ULAs, uma para cada elemento do vetor, o que é impraticável

Arquiteturas SIMD
Processadores Vetoriais

- Pipeline Vetorial
  - Operações necessárias para se realizar uma soma de dois números em ponto flutuante:
    A. Os expoentes dos dois operandos são comparados de modo que o número de maior magnitude é usado como base.
    B. O significando do número de menor magnitude sofre um *shift* de modo que os expoentes dos dois operandos estejam de acordo.
    C. Os significandos são somados.
    D. É feito o arredondamento do resultado.
    E. Verifica-se se ocorreu alguma exceção durante a adição (por exemplo, *overflow*).
    F. O resultado da adição é normalizado.
- Solução: implementação de um pipeline para execução das operações sobre ponto flutuante

Arquiteturas SIMD
Processadores Vetoriais

Step	Time → τ	2τ	3τ	4τ	5τ	6τ	7τ	8τ
A	$x_1 + y_1$						$x_2 + y_2$	
B		$x_1 + y_1$						$x_2 + y_2$
C			$x_1 + y_1$					
D				$x_1 + y_1$				
E					$x_1 + y_1$			
F						$x_1 + y_1$		

Figure 2: Scalar Floating-point Addition of Vector Elements.

Illustration 2Adição FP escalar de cada elemento do vetor.

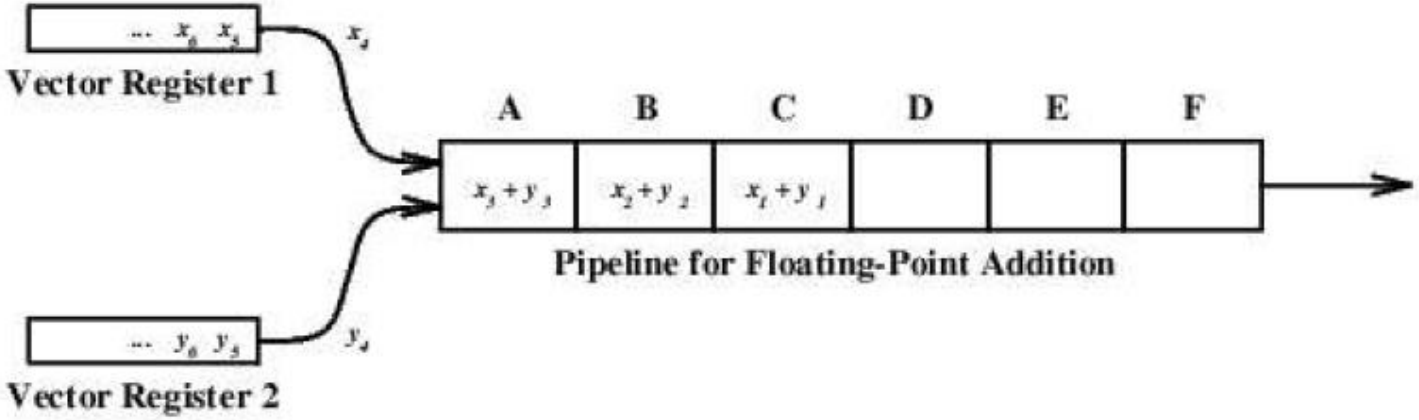


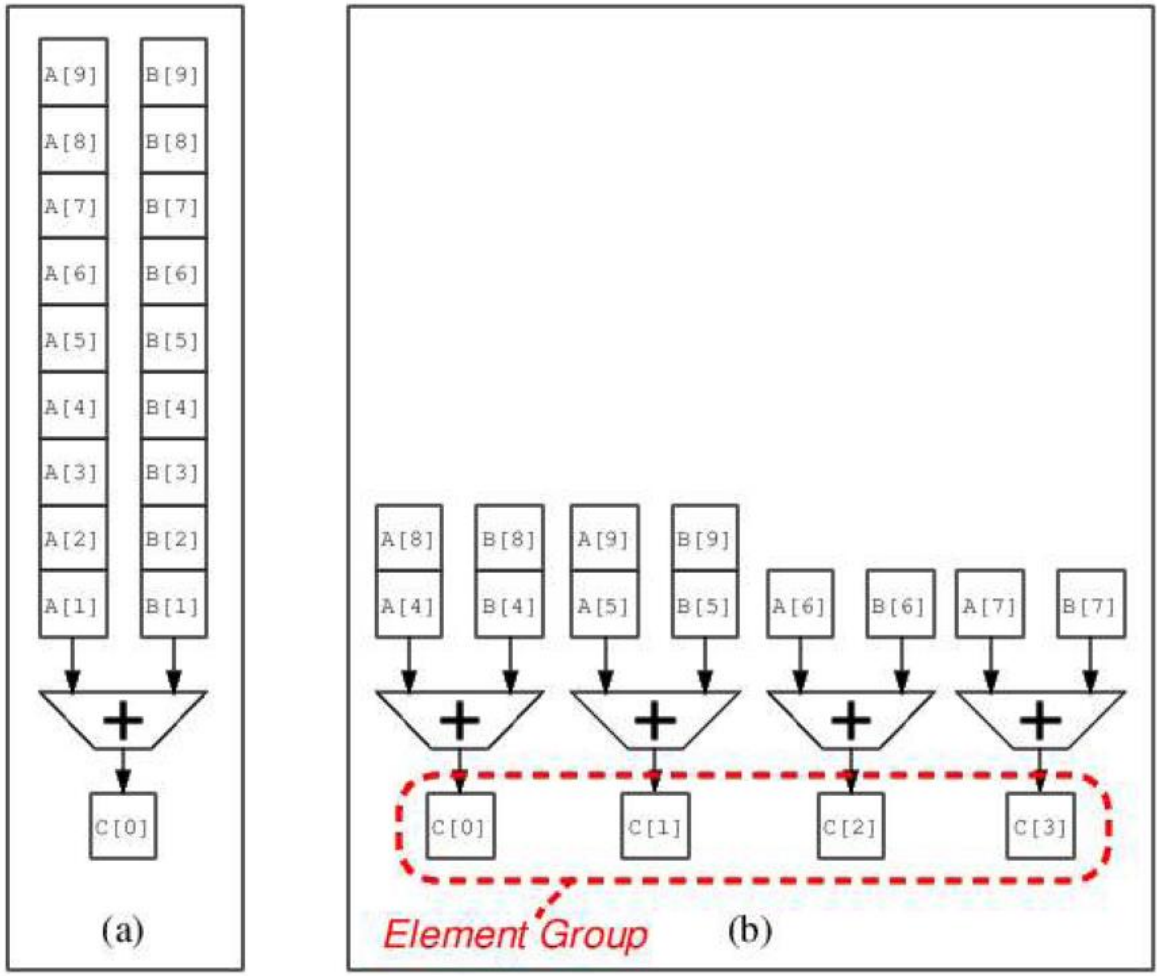
Figure 3: Pipeline for Floating-point Addition of Vector Elements.

Arquiteturas SIMD
Processadores Vetoriais

- Qual seria o desempenho se uma operação de soma de dois vetores de 1000 posições fosse realizada utilizando uma unidade de ponto flutuante escalar e com uma unidade de ponto flutuante com pipeline?
- Suponha um clock de 200 MHz para cada estágio do pipeline

Arquiteturas SIMD
Processadores Vetoriais

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



Arquiteturas SIMD
Processadores Vetoriais

- Problemas para a vetorização
  - Quando o tamanho do vetor é maior do que o dos registradores vetoriais, o compilador deve quebrar em várias operações vetoriais
  - Stride
    - Quantidade de dados que devem ser “saltadas” na leitura/escrita
    - Utilizado principalmente quando se trabalha com matrizes, pois estas não são armazenadas sequencialmente na memória
    - Normalmente as instruções de *load* e *store* possuem um valor para se especificar o stride

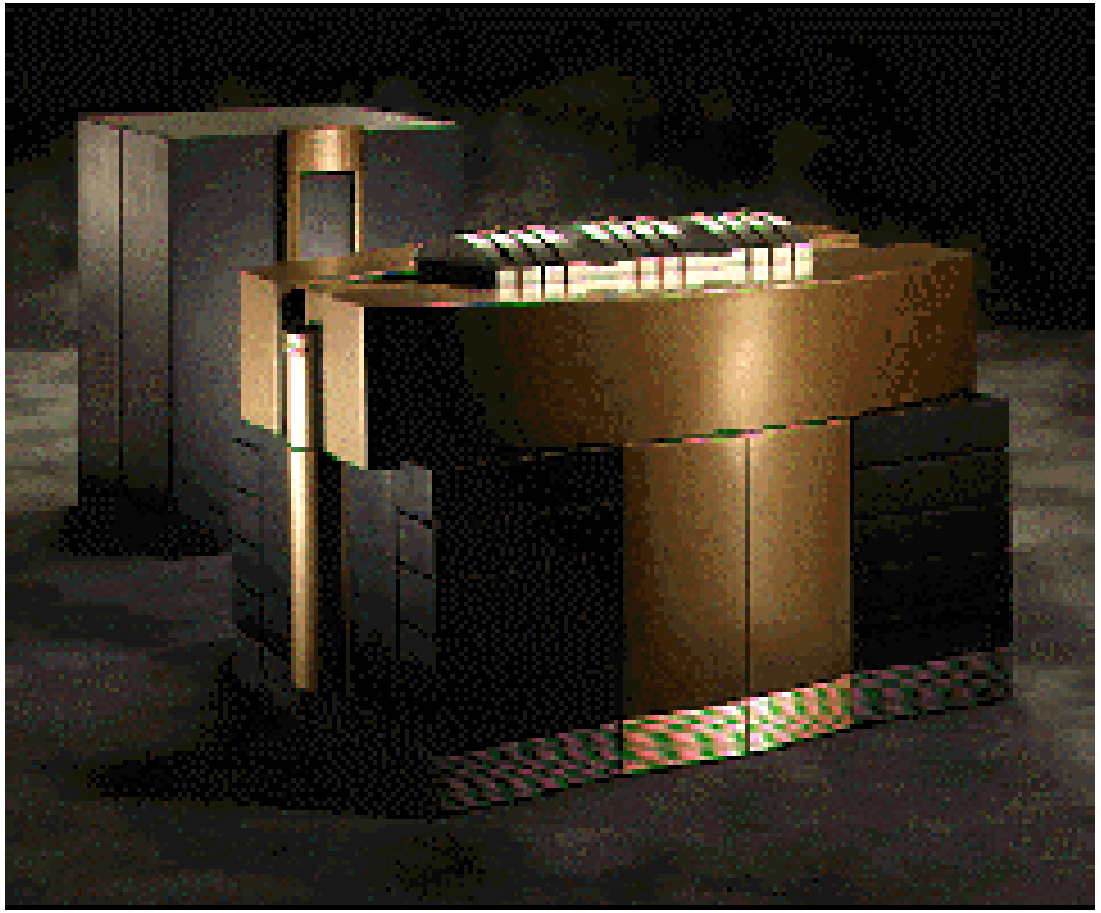
Arquiteturas SIMD
Processadores Vetoriais

- Problemas para a vetorização
  - Existência de desvios condicionais nos *loops*

```
for (i=1; i<256; i++)
    if (a[i] < 0)
            a[i] = a[i] + b[i];
```

Arquiteturas SIMD
Processadores Vetoriais

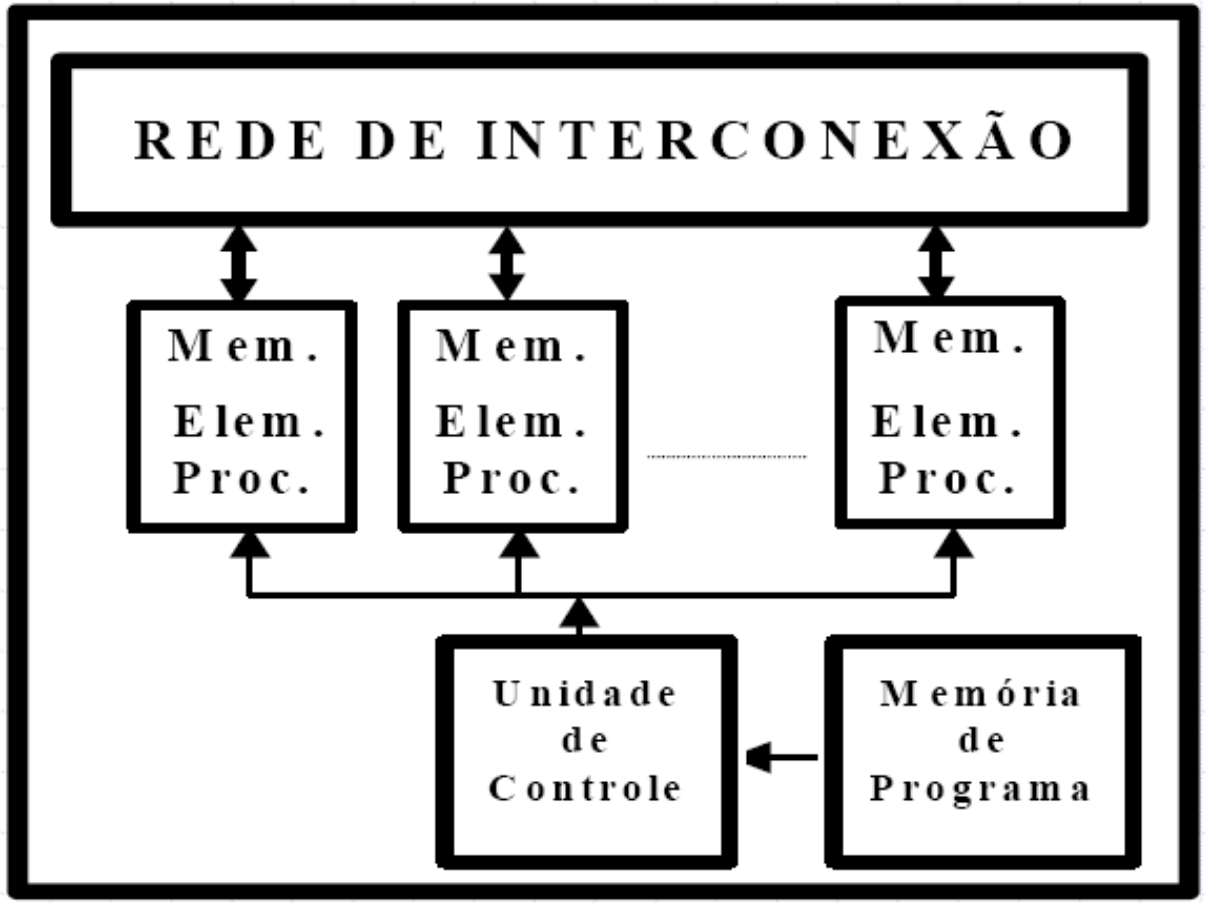
- Supercomputador Vetorial Cray T90



Arquiteturas SIMD
Arquiteturas Matriciais

- Processadores executam sincronizadamente a mesma instrução sobre dados diferentes.
- Utiliza vários processadores especiais muito mais simples, organizados em geral de forma matricial.
- Muito eficiente em aplicações onde cada processador pode ser associado a uma sub-matriz independente de dados (processamento de imagens, algoritmos matemáticos, etc.)
- Não incluem processamento escalar

Arquiteturas SIMD
Processadores Matriciais



Arquiteturas SIMD
Processadores Matriciais

- Conectividade
  - Vizinhança
  - Árvore
  - Pirâmide
  - Hipercubo

Arquiteturas SIMD
Processadores Matriciais

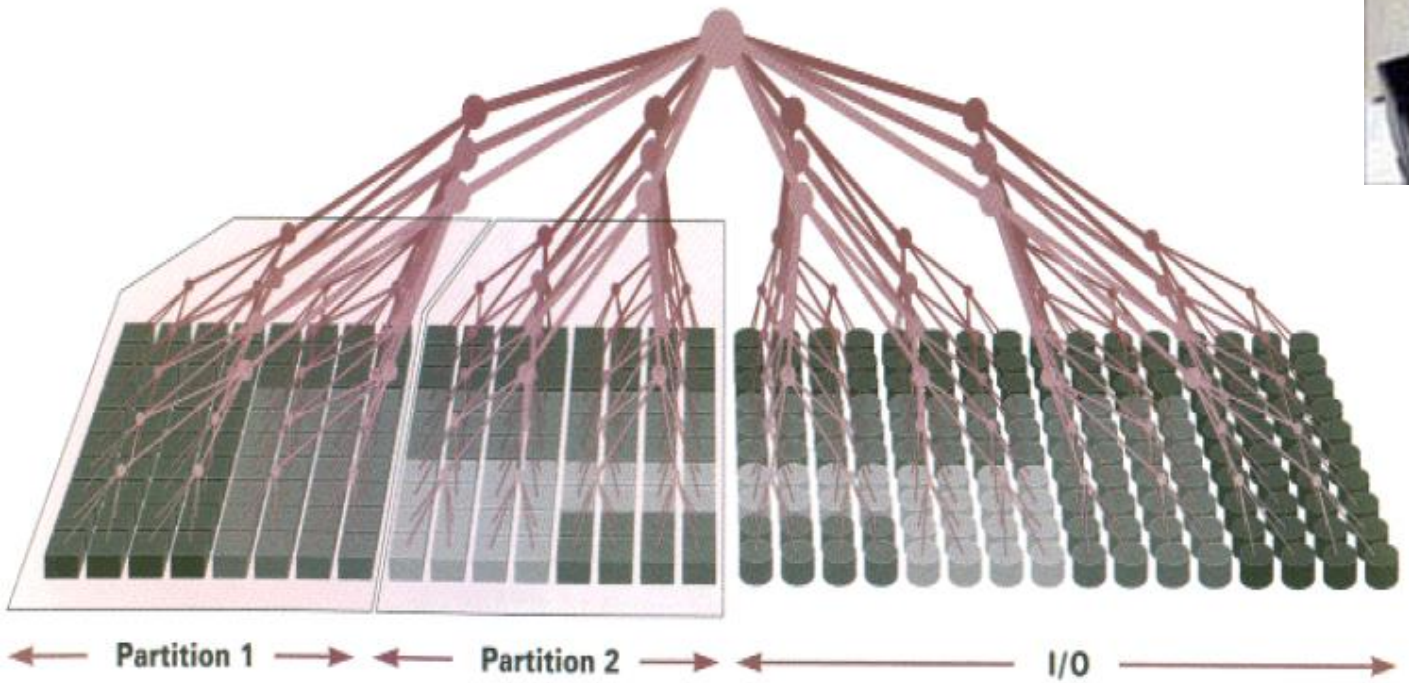
- Exemplos
  - Illiac IV
    - Matriz 8 x 8 (1968)
  - MPP
    - Matriz 128 x 128 (1983)
  - Connection Machine
    - Hipercubo (16K x 4) processadores (1985)
  - MasPar
    - Multi-level crossbar switch (1990)
    - 16K processadores

Arquiteturas SIMD
Processadores Matriciais

- CM-5 (Connection Machine 5)
  - FROSTBURG: primeiro computador massivamente paralelo comprado pela NSA (*US National Security Agency*), em 1991, com 256 nós

Arquiteturas SIMD
Processadores Matriciais

- CM-5
  - Conexão em árvore

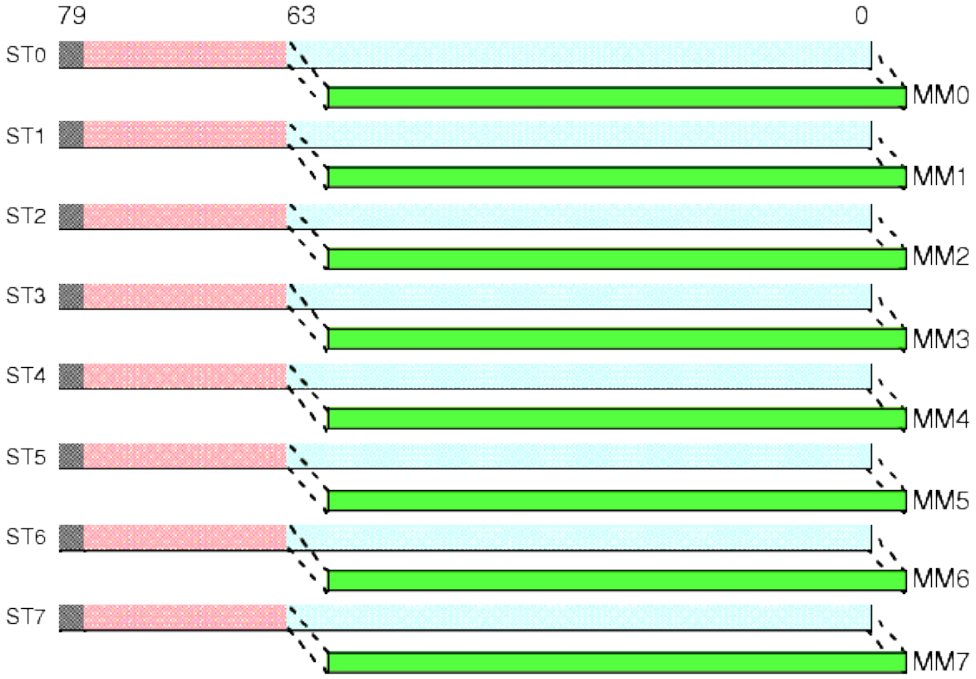


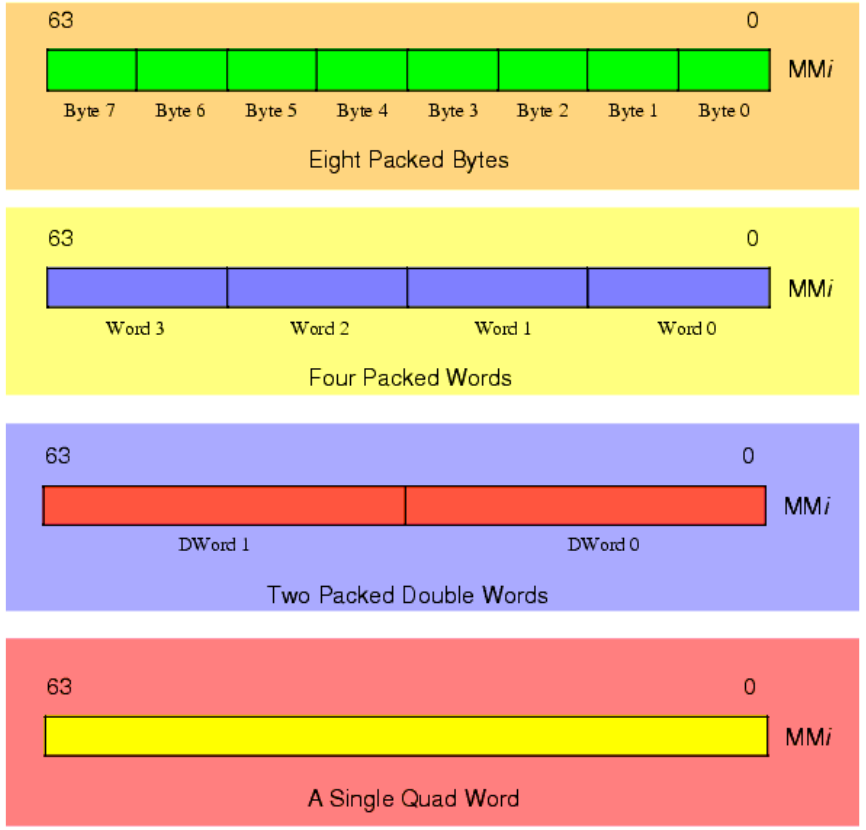
Arquiteturas SIMD
Extensões do conjunto de instruções para multimídia

- Extensões:
  - MMX (*Multimedia Extension* - 1996)
  - SSE (*Streaming SIMD Extensions* - 1999)
  - AVX (*Advanced Vector Extensions* - 2008)

Extensões do conjunto de instruções para multimídia MMX

- 8 registradores “vetoriais” (MM0 .. MM7) de 64 bits, mapeados nos registradores de ponto flutuante (ST0 .. ST7) e suporta 4 tipos de dados nos vetores





Extensões do conjunto de instruções para multimídia SSE

- 8 novos registradores de 128 bits (XMM0 .. XMM7) para, a princípio, armazenar somente 4 números ponto flutuante de precisão simples de 32 bits. Depois expandiu para armazenar também:
  - 2 números ponto flutuante de precisão dupla
  - 2 números inteiros de 64 bits
  - 4 números inteiros de 32 bits
  - 8 números short-integer de 16 bits
  - 16 bytes ou caracteres

Extensões do conjunto de instruções para multimídia SSE

- Exemplo de código escalar em operações de ponto flutuante:

```
vec_res.x = v1.x + v2.x;
vec_res.y = v1.y + v2.y;
vec_res.z = v1.z + v2.z;
vec_res.w = v1.w + v2.w;
```

- Exemplo de código utilizando as instruções SSE

```
movaps xmm0, [v1] ;        xmm0 = v1.w | v1.z | v1.y | v1.x
addps xmm0, [v2] ;xmm0 = v1.w+v2.w | v1.z+v2.z | v1.y+v2.y | v1.x+v2.x
movaps [vec_res], xmm0
```

- Link interessante: http://sci.tuomastonteri.fi/programming/sse

Extensões do conjunto de instruções para multimídia AVX

- 8 registradores do SSE aumentaram de 128 bits para 256 bits e foram renomeados para YMM0 .. YMM7. Mais 8 registradores foram acrescentados (YMM8 .. YMM16)
- Compatibilidade das operações do SSE pode ser obtida utilizando o prefixo VEX* para operar nos 128 bits menos significativos dos registradores YMM

* Prefixo VEX permite a definição de novas instruções e a extensão ou modificação de códigos de instruções previamente definidos

Extensões do conjunto de instruções para multimídia AVX

- Alguns links interessantes
  - https://software.intel.com/en-us/articles/using-avx-without-writing-avx-code
  - https://software.intel.com/en-us/articles/a-guide-to-auto-vectorization-with-intel-c-compilers/