Introdução à Programação de Processadores Vetoriais

Tópicos:

- Conceitos Básicos
- Encadeamento de Operações
- Desempenho

Ref.: Tutorial sobre Sistemas Vetoriais
(disponível na home-page)

Conceitos Básicos

- **Compilador Vetorizador**: transforma código de alto nível em código objeto para um Sist. Vetorial
- **Notação Vetorial**:

```
DO 20 I=1,N
  X(I)= 2.0 * Y(I)
ENDDO
```

```
DO 30 J=1,N,2
  X(J)=3.0 * Z(J)
enddo
```

são equivalentes a

```
X(1:N)= 2.0* Y(1:N)
```

```
X(1:N:2)= 3.0* Z(1:N:2)
```

onde: `X = X(LimInf:LimSup:Stride)`

Obs: quando omitido, `Stride` = 1

Conceitos Básicos (cont.)

- **Com vetores multidimensionais:** `A(N,M)`

  `A(2,1:M)` : segunda linha de `A`

  `A(2,:)` : segunda linha de `A`
- **Vantagens da Notação Vetorial:**
  - Possibilidade de vetorização pelo compilador torna-se explícita
  - Maior clareza do código
- **Observações:**
  - Disponível em algumas linguagens (Ex: Fortran 90)
  - Operações apenas entre vetores *compatíveis*

Operação Vetorial Típica

- Código de Alto Nível:

```
do i=1,N
      Z(i) = s * X(i) + Y(i)
enddo
```

- Em notação vetorial: **Z** = s * **X** + **Y**

  (**X**,**Y**,**Z** vetores, s escalar)

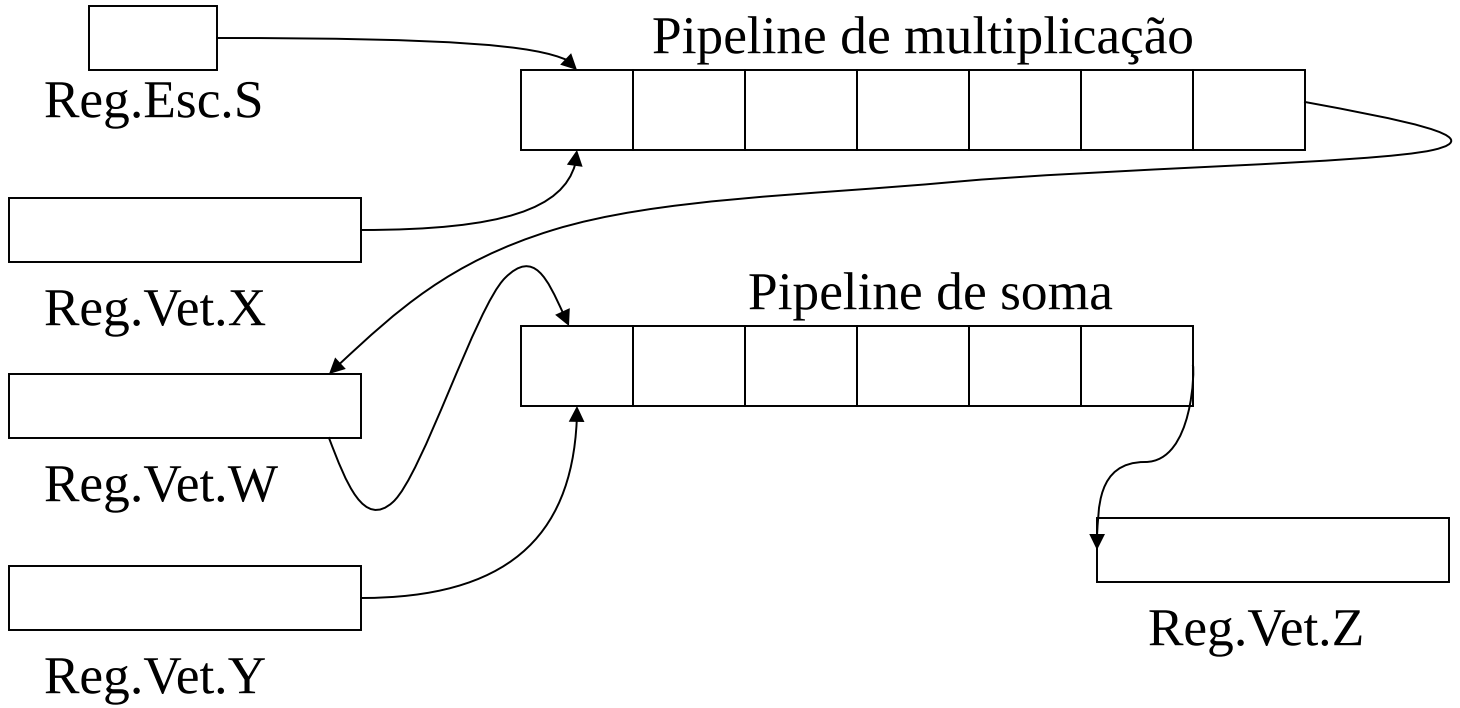
- Típico Código Objeto Vetorizado:
  - load R1 ← mem(s)
  - vload V1 ← mem(X)
  - vload V2 ← mem(Y)
  - svmult V3 ← R1 × V1
  - vadd V4 ← V3 + V2
  - vstore mem(Z) ← V4

Encadeamento de Operações

- Operação vetorial: **Z** = s * **X** + **Y**
  (**X**,**Y**,**Z** vetores, s escalar)
- Núcleo da Implementação Normal:
  Reg.Vet.W ← Reg.Esc.S * Reg.Vet.X
  Reg.Vet.Z ← Reg.Vet.W + Reg.Vet.Y
- Constatações:
  - Segunda operação só inicia após término da primeira
  - É necessário armazenar vetor de resultado temporário
- Otimização:
  - Iniciar segunda operação assim que possível
    (alimentar pipeline de soma com resultados da multiplicação)

Encadeamento (cont.)

- Exemplo: **Z** = s * **X** + **Y**



Encadeamento (cont.)

- Conseqüências do Encadeamento:
  - Unidades de soma e multiplic. operam simultaneamente
  - Duas operações aritméticas são completadas por ciclo
  - Comprimento efetivo do pipeline é igual a um mais a soma dos dois pipelines individuais
- Observações:
  - Operações vetoriais de leitura/escrita em memória também podem participar de encadeamento
  - Controle do encadeamento é feito por hardware
  - Há múltiplas leituras/escritas aos registros, por ciclo
  - Encadeamento é essencial para alto desempenho

Desempenho

- Supondo uma operação de k passos num vetor de comprimento N:

Execução **serial** (sem pipeline):

$$T_{serial} = (S_S + k\ N)\ T$$

onde:

S_S = overhead de inicialização

T = tempo de ciclo do relógio

e para $N \rightarrow \infty$

$$T_{serial} \rightarrow k\ N\ T$$

Desempenho (cont.)

Execução **com pipeline**:

$$T_{pipe} = (S_p + K)\,T + (N-1)\,T$$

onde:

S_p = overhead de início da operação vetorial

K = número de estágios do pipeline

e para $N \rightarrow \infty$

$$T_{pipe} \rightarrow N\,T$$

Logo, para $N \rightarrow \infty$, $T_{serial} = K\,T_{pipe}$

Desempenho (cont.)

- Figuras de Desempenho:
  - R_N : Taxa em Mflops para um vetor de tamanho N
  - $R\infty$: R_N para $N \rightarrow \infty$
  - $N_{1/2}$: valor de N tal que $R_N = R\infty / 2$
  - N_V : valor de N tal que $T_{pipe} < T_{serial}$
- Exemplos:

	$R\infty$	$N_{1/2}$	T(nsec)
Cray-1	22	18	12.5
CDC-Cyber 205	50	86	20.0
Cray-X-MP	70	53	9.5

Desempenho (cont.)

- Formas de Ganho de Desempenho:
  - Vetorização
    - Um processador vetorial (com pipelining)
  - Paralelização
    - Vários processadores convencionais
  - Vetorização e Paralelização
    - Vários processadores vetoriais (caso dos sistemas atuais)
    - Ex: Processamento de uma grade 2-D (X,Y)
      - Vetorização em X, Paralelização em Y