

# Análise de Dependência

## **Tópicos:**

- Restrições à Vetorização
- Detecção de Dependência
- Teste de Dependência

# Restrições à Vetorização

- Caso Trivial de Vetorização: (supondo  $N \leq VL_{\max}$ )

```
DO i=1,N  
  A(i) = B(i) + C(i)  
ENDDO
```

## Código Vetorizado:

```
VL ← N  
Vx ← mem(B)           (load vector)  
Vy ← mem(C)           (load vector)  
Vz ← Vx + Vy          (add vector)  
mem(A) ← Vz           (store vector)
```

# Restrições à Vetorização (cont.)

- Problema: Dependências de dados

```
DO i=2,N
```

```
  A(i) = 2.0 * A(i-1)           ( Recorrência ! )
```

```
ENDDO
```

## Tentativa de Vetorização:

```
VL ← N-1
```

```
Vx ← mem(A)           (load vector)
```

```
Vy ← 2 * Vx           (mult scalar * vector)
```

```
mem(A2) ← Vy          (store vector)
```

Se houver vetorização, o resultado do cálculo é errado!

➡ Causa: valores intermediários **não** são usados

# Dependência de Dados

- **Definição:** Há dependência (de dados) entre dois comandos se uma mesma posição de memória é acessada por ambos os comandos.
- **Exemplo:**
  - (1)  $\underline{a} = b + c$
  - (2)  $x = \underline{a} + y$

Neste par de comandos, deve ser observado que:

  - (1) deve ser executado *antes* que (2)
  - o valor de “a” computado em (1) deve ser usado em (2)

# Dependência de Dados (cont.)

- **Situações de Possível Dependência:**

- Sequências de comandos escalares:

$$\begin{aligned}\underline{a} &= b + c \\ x &= \underline{a} + y\end{aligned}$$

- Comandos na mesma iteração de um loop:

$$\begin{aligned}\text{do } i=1, N \\ \quad \underline{a(i)} &= b(i) + c(i) \\ \quad x(i) &= \underline{a(i)} + y(i) \\ \text{enddo}\end{aligned}$$

- Comandos em iterações distintas de um loop:

$$\begin{aligned}\text{do } i=2, N \\ \quad \underline{a(i)} &= \underline{a(i-1)} + 3 \\ \text{enddo}\end{aligned}$$

(Diz-se que a dependência é *carregada* pelo loop)

# Dependência de Dados (cont.)

- **Tipos de Dependências de Dados:**
  - RAW (read after write): dependência de fluxo
  - WAR (write after read): anti-dependência
  - WAW (write after write): dependência de saída
  - RAR (read after read): não têm interesse prático

- **Exemplos:**

Dep.Fluxo:  
(RAW)

$$\underline{A} = B + C$$

$$X = \underline{A} + Y$$

Anti-Dep:  
(WAR)

$$A = B + \underline{C}$$

$$\underline{C} = X + Y$$

Dep.Saída:  
(WAW)

$$\underline{A} = B + C$$

$$\underline{A} = X + Y$$

# Dependência de Dados (cont.)

- **Mais Exemplos (com loops):**

- Dependência de Fluxo (*raw*) :

```
do i=2,N
  a(i) = a(i-1) + 3
enddo
```

- Anti-Dependência (*war*):

```
do i=1,N-1
  a(i) = a(i+1) + 3
enddo
```

- Dependência de Saída (*waw*):

```
do i=1,N-1
  s = s + random(i)
enddo
```

# Dependência de Dados (cont.)

- **Observações:**

- Apenas dependências de fluxo são *verdadeiras*
- Antidependências e dependências de saída ocorrem devido à reutilização de variáveis, e podem **sempre** ser eliminadas!

- **Exemplos:**

Dep.Fluxo:

$$\underline{A} = B + C$$

$$X = \underline{A} + Y$$

Anti-Dep:

$$A = B + \underline{C}$$

~~$$\underline{C} = X + Y$$~~

$$P = X + Y$$

Dep.Saída:

$$\underline{A} = B + C$$

~~$$\underline{A} = X + Y$$~~

$$P = X + Y$$



# Detecção de Dependência

- **Exemplo-1:**

```
do i=2,N  
  a(i) = a(i-1) + 3  
enddo
```

Há claramente dependência, pois  $a(2)$  é *escrito* na iteração  $i=2$  e *lido* na iteração  $i=3$ .

- **Exemplo-2:**

```
do i=1,N  
  a(24*i-4) = a(5*i+15) + 3  
enddo
```

Pergunta: Há dependência ???

# Detecção de Dependência (cont.)

- **Resposta:**

Se  $a(24*j - 4)$  e  $a(5*k + 15)$  representarem a mesma posição de memória (mesmo elemento), para algum par  $(j,k)$  tal que  $1 \leq j, k \leq N$ , então **há** dependência!

**Problema Prático:** Chegar a esta decisão de forma *automática* (através do compilador).

**Em termos formais:** Existem  $j$ ,  $k$ , tais que

$$1 \leq j \leq N, 1 \leq k \leq N, \text{ e } 24*j - 4 = 5*k + 15 \quad ?$$

# Detecção de Dependência (cont.)

- **Testes de Dependência:**
  - Aplicados automaticamente pelos compiladores
  - Inúmeros testes disponíveis na literatura
  - Em geral, há diretivas que o programador pode usar para dizer que **não** há dependências
- **Hipótese Simplificadora:**
  - Índices dos arrays de interesse são *combinação linear* das variáveis do loop;  
Ex: 

```
do i=1,N  
    A(4*i-3) = A(3*i+2) + ...  
enddo
```

# Detecção de Dependência (cont.)

- **Complicações Possíveis:**

- Limites do loop podem não ser conhecidos em tempo de compilação; Ex: do  $i=1, \underline{N}$
- Pode haver subscritos de subscritos  
Ex:  $A(K(i)) = A(\dots)$
- *Aliasing* de variáveis, tal como em:
  - Fortran: `common A, B`  
 $A(\dots) = B(\dots) + \dots$
  - C (ponteiros):  $*p = *q + 3$ , onde  
 $p = \&A[3*i - 2]$ ,  $q = \&A[2*i + 5]$

# Teste de Dependência

- Veremos apenas um teste *simples*: Teste do MDC (consultar literatura para testes mais poderosos)
- Equações *Diophantine*:  $a_1x_1 + a_2x_2 + \dots + a_nx_n = c$ 
  - Dados  $\{a_1, a_2, \dots, a_n, c\}$  encontrar valores inteiros para  $\{x_1, x_2, \dots, x_n\}$  que satisfaçam a equação
- **Teste:** Há solução inteira *se e somente se*  $g$  divide  $c$   
onde  $g = \text{MDC}(a_1, a_2, \dots, a_n)$ 
  - Se não houver solução  $\Rightarrow$  Não há dependência
  - Se houver solução  $\Rightarrow$  Pode haver dependência (depende da solução estar dentro dos limites do loop)

# Teste de Dependência (cont.)

- **Exemplo-1:**

```
do i=1,N  
  a(2*i) = a(2*i+1) + ...  
enddo
```

Equação:  $2 * X_1 = 2 * X_2 + 1 \rightarrow 2 * X_1 - 2 * X_2 = 1$

MDC(2, -2) = 2 mas 2 **não** divide 1 !

Logo, não há soluções, e assim **não** existe dependência;  
as iterações do loop são independentes  
→ Loop pode ser vetorizado

# Teste de Dependência (cont.)

- **Exemplo-2:**

```
do i=1,N
  a(19*i+3) = ...
  ... = a(2*i+21) + ...
enddo
```

Equação:  $19 * X_1 + 3 = 2 * X_2 + 21 \rightarrow 19 * X_1 - 2 * X_2 = 18$

$\text{MDC}(19, -2) = 1$  e 1 **divide** 18 !

Neste caso,  $X_1=2$  e  $X_2=10$  são soluções, logo:

- Não há dependência se  $N \leq 9$  (loop **pode** ser vetorizado)
- Há dependência se  $N > 9$  (loop **não** pode ser vetorizado)

# Teste de Dependência (cont.)

- Teste do MDC:
  - Ainda é muito usado hoje, devido à sua simplicidade
  - Caso  $MDC=1$  ocorre com frequência na prática, tornando o teste pouco conclusivo.
  - Pode ser estendido para mais que uma dimensão, mas a *forma* do espaço de iteração não é considerada
- Testes de Dependência em Geral:
  - Continuam a ser *tema de pesquisa* corrente
  - São importantes tanto para *vetorização* como para *paralelização*