向量处理机

- 基本概念
- 基本结构
- 设计目标
- 关键技术
- 协处理器
- 性能评价

基本概念

- 向量处理机
- 什么是向量处理
- 向量处理方式

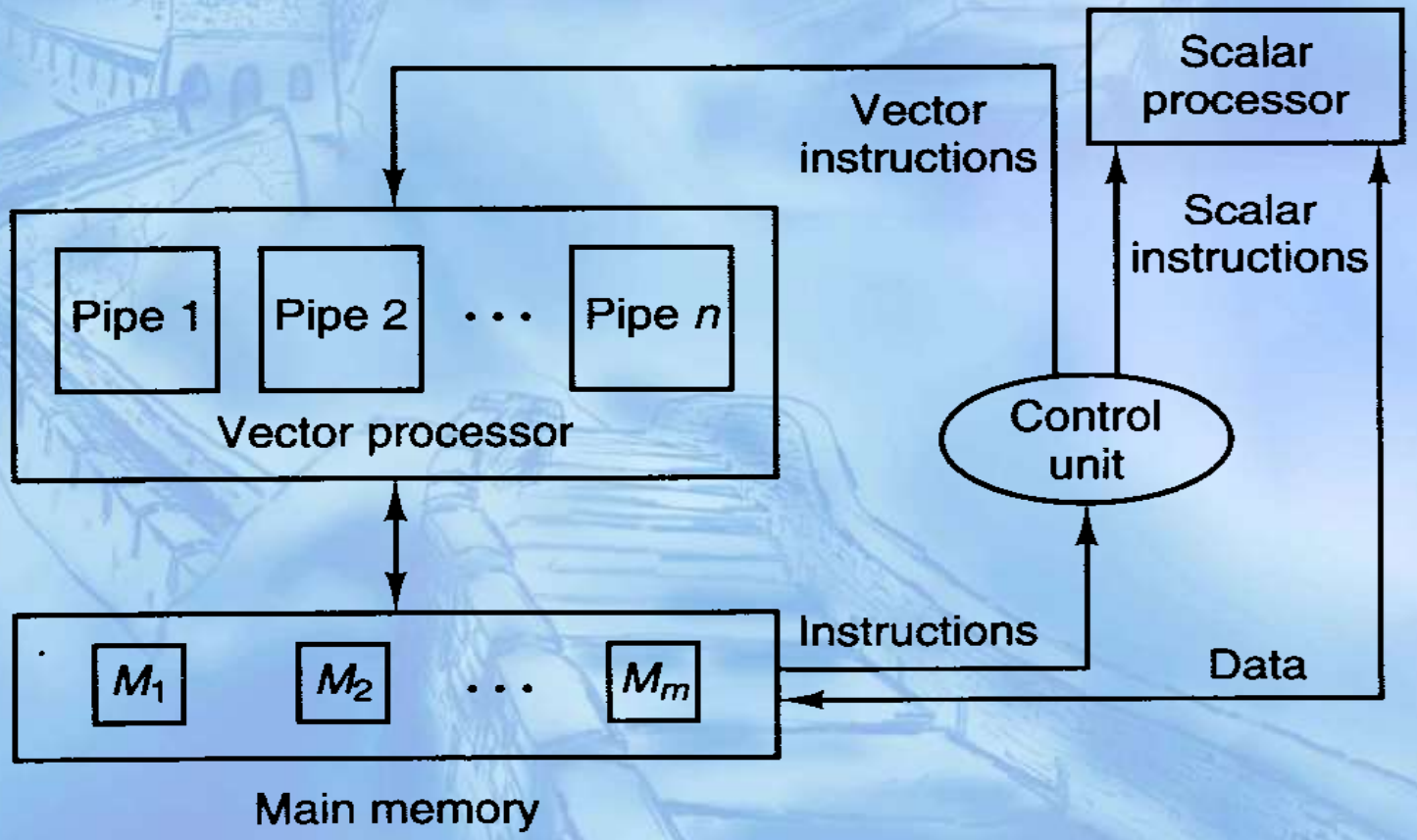
向量处理机

具有向量数据表示和向量指令系统的处理机，是解决数值计算问题的一种高性能计算机结构。有两个主要优点：效率高和适用性广，一般都采用流水线结构，有多条流水线并行工作。

向量处理机通常属大型或巨型机，也可以用微机加一台向量协处理器组成。一般向量计算机中包括有一台高性能标量处理机。

必须把要解决的问题转化为向量运算，向量处理机才能充分发挥作用

向量处理机


Scalar processor
Vector instructions
Scalar instructions
Pipe 1
Pipe 2
Pipe n
Vector processor
Control unit
Instructions
Data
M_1
M_2
M_m
Main memory

什么是向量处理

— 例子

用Fortran语言编写的一个简单程序：

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

什么是向量处理

— 标量处理

```
        INITIALIZE  I=1
10      READ        B(I)                    ;读数指令
        READ        C(I)
        ADD         B(I)+C(I)               ;运算指令
        STORE       A(I)←B(I)+C(I)          ;存数指令
        READ        A(I+1)
        MULTIPLY    2*A(I+1)                ;运算指令
        STORE       B(I)←2*A(I+1)           ;存数指令
        INCREMENT I←I+1                     ;运算指令
        IF I≤N GOTO 10                      ;条件转移指令
        STOP
```

什么是向量处理

— 向量处理

```
A(1:N)=B(1:N)+C(1:N) ;并行运算指令
TEMP(1:N)=A(2:N+1)   ;并行取数指令
B(1:N)=2*TEMP(1:N)   ;并行运算指令
```

一条向量指令处理N个操作数或N对操作数

向量处理方式

- 横向处理方式
- 纵向处理方式
- 纵横处理方式

- C语言程序

```
for (i=1;i<=N;i++)
   y[i]=a[i]×(b[i]+c[i]);
```

采用同一例子说明

横向处理方式

- **处理方法**

  又称为**水平处理方式、横向加工方式**等。向量计算是按行的方式从左至右横向地进行。

- **举例**

  逐个分量进行处理：假设中间结果为T(I)

计算第1个分量：
$$T(1) = B(1) + C(1)$$
$$Y(1) = A(1) \times T(1)$$

计算第2个分量：
$$T(2) = B(2) + C(2)$$
$$Y(2) = A(2) \times T(2)$$

……

计算最后一个分量：
$$T(N) = B(N) + C(N)$$
$$Y(N) = A(N) \times T(N)$$

横向处理方式

- **分析**

存在两个问题：在计算向量的每个分量时，都发生写读数据相关，流水线效率低；如果采用多功能流水线，还必须频繁进行流水线切换。所以横向处理方式对向量处理机不适合，即使在标量处理机中，也经常通过编译器进行指令流调度。

纵向处理方式

- **处理方法**

  也称为**垂直处理方式**、**纵向加工方式**等。向量计算是按列的方式自上而下纵向地进行。

- **举例**

$$T(1) = B(1) + C(1)$$
$$T(2) = B(2) + C(2)$$
$$\cdots\cdots$$
$$T(N) = B(N) + C(N)$$
$$Y(1) = A(1) \times T(1)$$
$$Y(2) = A(2) \times T(2)$$
$$\cdots\cdots$$
$$Y(N) = A(N) \times T(N)$$

纵向处理方式

- **分析**

因为数据相关不影响流水线连续工作，不同的运算操作只需要切换1次，所以这种处理方式适用于向量处理机。

结果的存储直接面向存储器，N的大小可以不受限制，但速度受到存储器吞吐量的限制。

采用向量指令只需要2条：

```
VADD    B, C, T
VMUL    A, T, Y
```

纵横处理方式

- **处理方法**

又称为分组处理方式、纵横向加工方式等。横向处理和纵向处理相结合的方式。即：将长度为N的向量分成若干组，每组长度为n，组内采用纵向处理方式，组间采用横向处理方式。

纵横处理方式

- **举例**

第1组: $T(1,n) = B(1,n) + C(1,n)$

$Y(1,n) = A(1,n) \times T(1,n)$

第2组: $T(n+1,2n) = B(n+1,2n) + C(n+1,2n)$

$Y(n+1,2n) = A(n+1,2n) \times T(n+1,2n)$

……

最后第k+1组: $T(kn+1,N) = B(kn+1,N) + C(kn+1,N)$

$Y(kn+1,N) = A(kn+1,N) + T(kn+1,N)$

纵横处理方式

- **分析**

减少了访问主存储器的次数，降低对存储器信息流量的要求，也减少访问存储器发生冲突引起的等待时间，因而提高了处理速度。

适合用于寄存器-寄存器结构的向量处理机中，因为向量寄存器的长度是有限的，例如，每个向量寄存器有64个寄存器。当向量长度N大于向量寄存器长度n时，需要分组处理。

基本结构

向量处理机的最关键问题是**存储器系统能够满足运算部件带宽的要求**。主要采用两种方法：

- 存储器 - 存储器结构

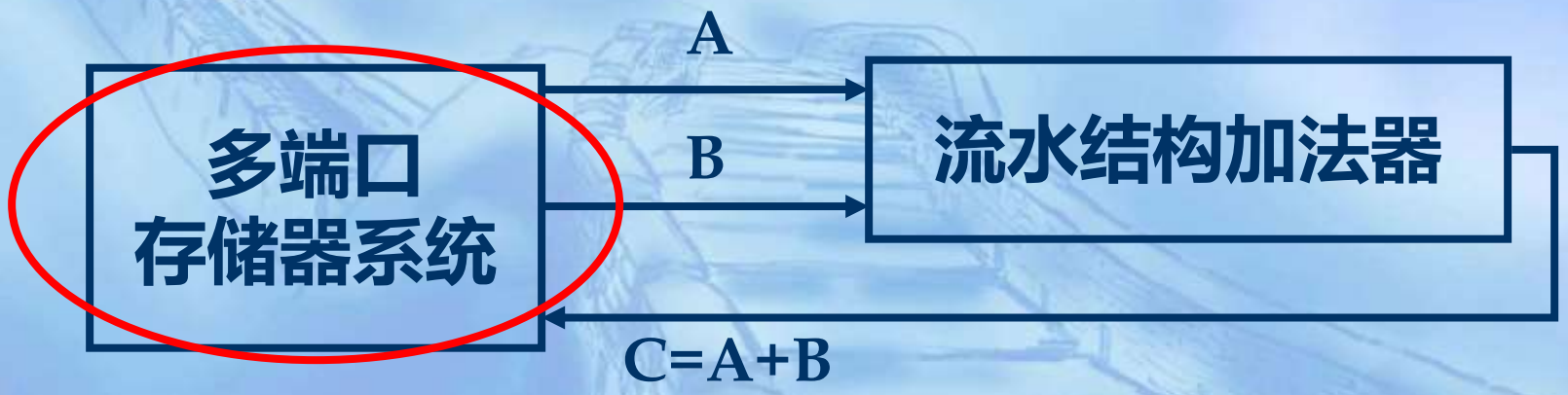
  多个独立的存储器模块并行工作。处理机结构简单，对存储系统的访问速度要求很高。

- 寄存器 - 寄存器结构

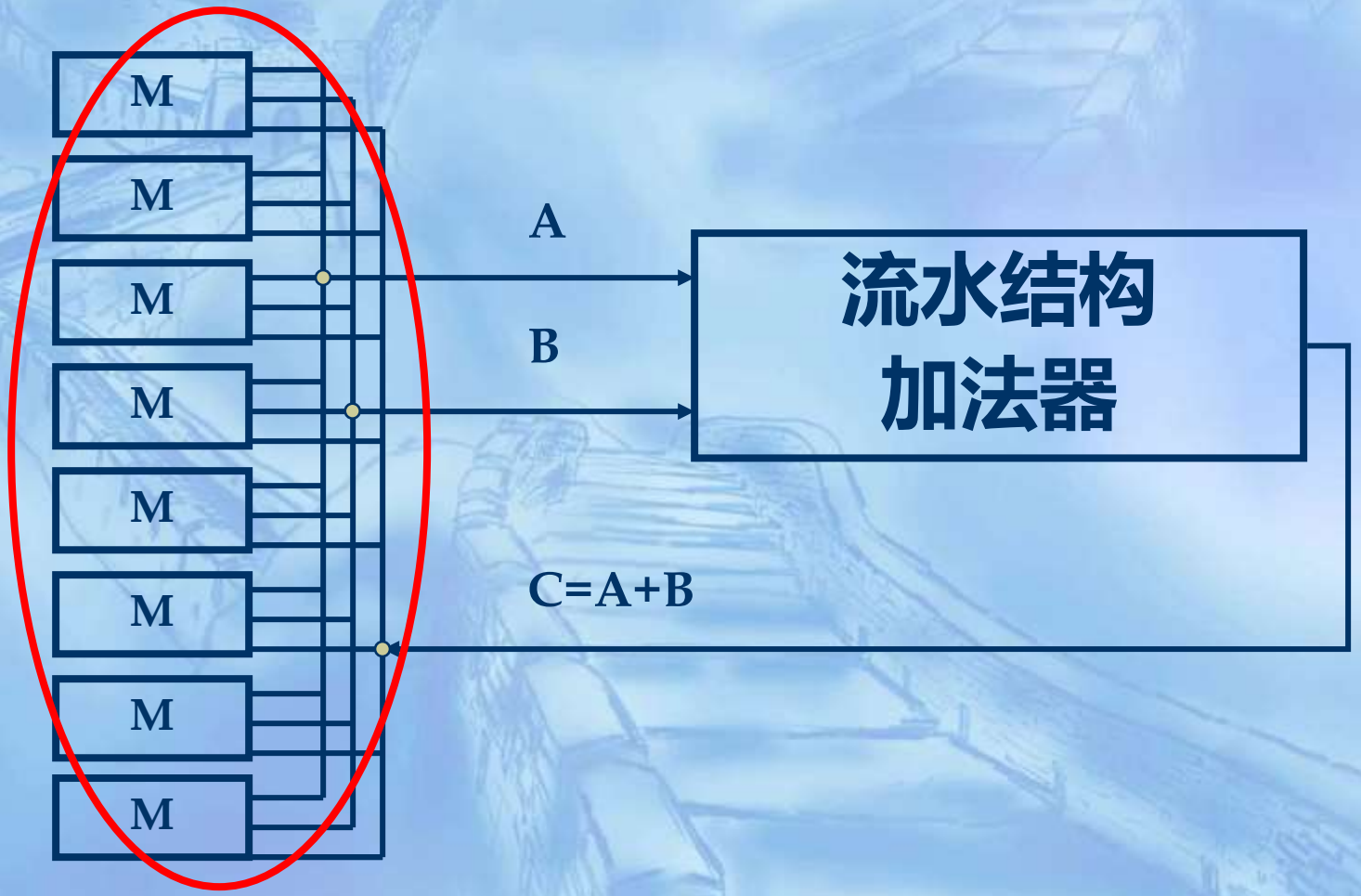
  运算通过向量寄存器进行。需要大量高速寄存器，对存储系统访问速度的要求降低，而且利用高速寄存器可完成对矩阵元素的特殊运算。

存储器 — 存储器结构

假设A、B、C都是有8个元素的向量，现向量处理机需完成如下运算：C=A+B。



存储器 — 存储器结构



存储器—存储器结构

采用多个存储体交叉和并行访问来提高存储器速度，但应该注意解决存储器访问冲突。下面分情况进行介绍（**假设一个存储周期占两个处理机周期**）：

- 理想情况
- 实际情况

数据存储

模块0	A[0]		B[6]		C[4]		…
模块1	A[1]		B[7]		C[5]		…
模块2	A[2]	B[0]			C[6]		…
模块3	A[3]	B[1]			C[7]		…
模块4	A[4]	B[2]		C[0]			…
模块5	A[5]	B[3]		C[1]			…
模块6	A[6]	B[4]		C[2]			…
模块7	A[7]	B[5]		C[3]			…

处理时序图

流水段4						0	1	2	3	4	5	6	7
流水段3					0	1	2	3	4	5	6	7	
流水段2				0	1	2	3	4	5	6	7		
流水段1			0	1	2	3	4	5	6	7			
存储体7						RB5	RB5	RA7	RA7	W3	W3		
存储体6					RB4	RB4	RA6	RA6	W2	W2			
存储体5				RB3	RB3	RA5	RA5	W1	W1				
存储体4			RB2	RB2	RA4	RA4	W0	W0					
存储体3		RB1	RB1	RA3	RA3								
存储体2	RB0	RB0	RA2	RA2									W6
存储体1		RA1	RA1					RB7	RB7			W5	W5
存储体0	RA0	RA0					RB6	RB6			W4	W4	

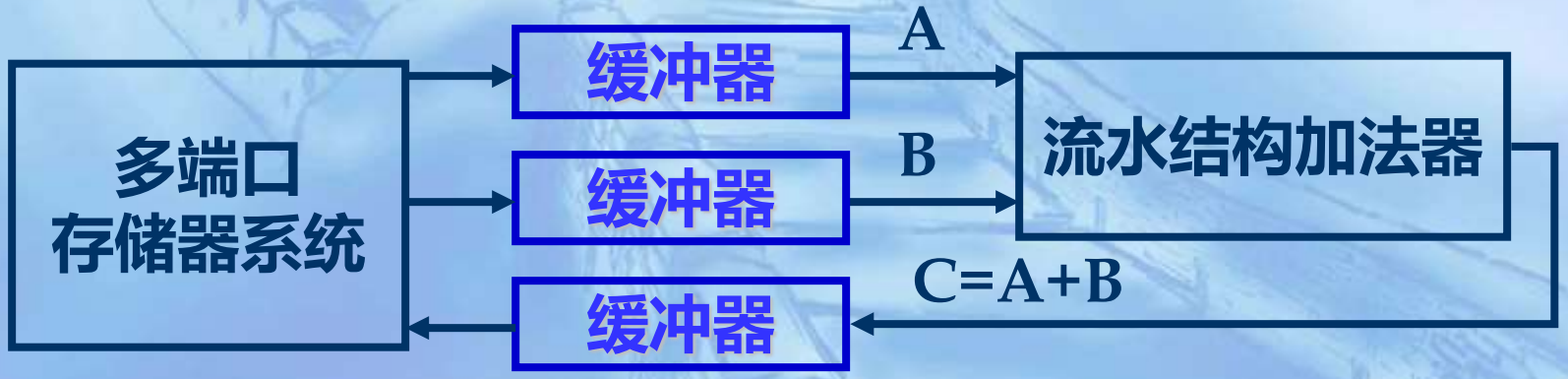
问题及解决

- **问题**

实际情况与理想情况并非一样，例如：向量的元素有时不能存放在我们希望的存储体。

- **解决**

可以在流水线的输入端和输出端增加缓冲器来消除争用存储器。



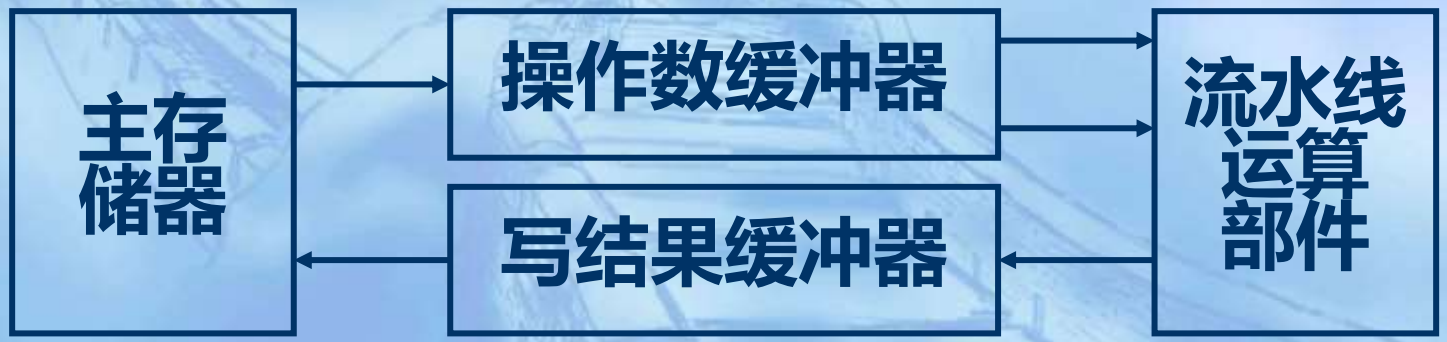
处理时序图

(所有向量都从模块0开始存放)

流水段4								0	1	2	3	4	5
流水段3							0	1	2	3	4	5	6
流水段2						0	1	2	3	4	5	6	7
流水段1					0	1	2	3	4	5	6	7	
存储体7								RA7	RA7	RB7	RB7		
存储体6							RA6	RA6	RB6	RB6			
存储体5	A延迟2					RA5	RA5	RB5	RB5				
存储体4					RA4	RA4	RB4	RB4					W4
存储体3				RA3	RA3	RB3	RB3					W3	W3
存储体2			RA2	RA2	RB2	RB2					W2	W2	
存储体1		RA1	RA1	RB1	RB1					W1	W1		
存储体0	RA0	RA0	RB0	RB0					W0	W0			

总 结

操作数缓冲器和写结果缓冲器主要用于**解决访问存储器冲突**。主要优缺点：硬件结构简单，造价低；但速度相对较低。



寄存器－寄存器结构

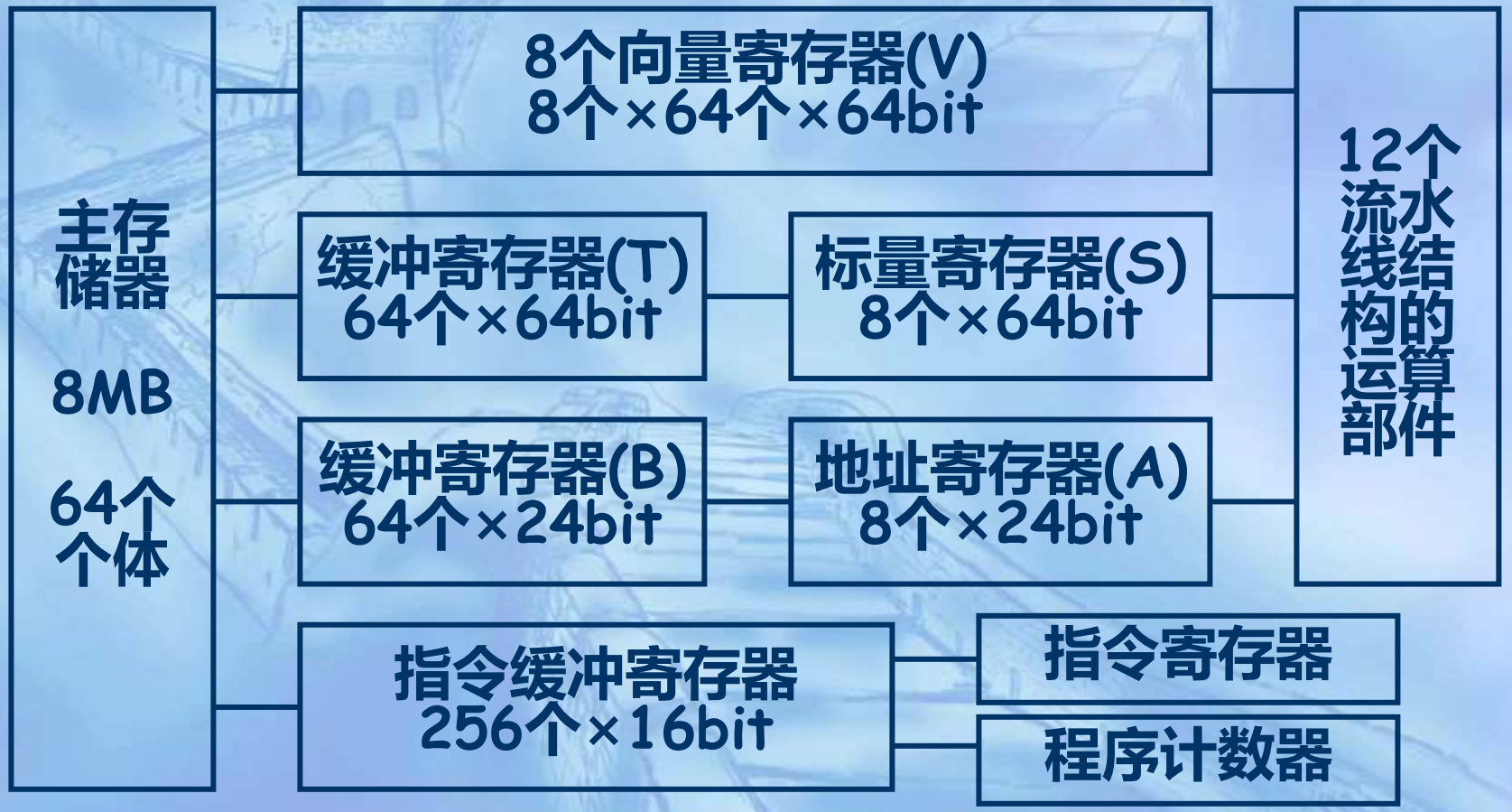
把存储器-存储器结构中的缓冲器改为**向量寄存器**，运算部件需要的操作数从向量寄存器中读取，运算的中间结果也写到向量寄存器中。

向量寄存器与标量寄存器的主要差别是：**一个向量寄存器能够保存一个向量**，例如：64个64位寄存器，用以实现连续访问一个向量的各个分量。

需要有标量寄存器和地址寄存器等共同工作。

举 例

— CRAY-1向量处理机结构



提 示

主要向量处理机都采用寄存器－寄存器结构，包括Cray处理机（Cray-1、Cray-2、 X-MP、Y-MP、C90、 T90和 SV1)、日本的超级计算机（NEC SX/2 ～ SX/5、Fujitsu VP200 ～VPP5000、Hitachi S820 和S-8300）和小型超级计算机（Convex C-1 ～C-4）。第一台向量处理机（CDC）采用存储器－存储器结构。

从现在开始，我们集中讨论寄存器－寄存器结构。

设计目标

- 较好地维持向量/标量性能平衡
- 可扩展性随处理机数目的增加而提高
- 增加存储器系统的容量和性能
- 提供高性能的I/O和易访问的网络

较好地维持向量/标量性能平衡

- 实际应用中通常既有向量计算又有标量计算，而且两类计算有一定的比例。关键问题是：**希望向量硬件和标量硬件都能够充分利用，不要空闲**。
- **向量平衡点**(vector balance point)：为使向量/标量硬件的利用率相等，一个程序中向量代码所占的百分比。
  - 例如：一个系统的向量运算速度为90Mflops，标量运算速度为10Mflops。如果程序的90%是向量运算，10%是标量运算，硬件利用率最高；则向量平衡点为0.9。
- 向量处理机的向量平衡点必须与用户程序的向量化程度相匹配。
  - 例如：IBM向量计算机维持较低的向量/标量比例（3～5）。这种做法能够适应通用应用问题对标量和向量处理要求。

可扩展性随处理机数目的增加而提高

可扩展性是指在确定的应用背景下，向量处理机系统要随处理机数目的增加而线性地提高。

- **规模可扩展性**：其资源部件（包括处理单元、存储器等）个数从小到大可扩展，需考虑价格、效率
- **换代可扩展性**：硬件、软件和算法的换代
- **问题可扩展性**：是指数据集规模，一台问题可扩展的计算机应在问题规模增大时仍能很好地工作，而且问题规模扩展到相同大时计算机仍能高效地运行

增加存储器系统的容量和性能

- **增加存储器系统的容量和性能**
  - 大规模存储器系统必须为标量处理提供低时延，为向量处理提供高频率，为解决大型复杂问题提供大容量和高吞吐率的性能
  - 为此存储器必须采用高效的层次结构

关键技术

- 链接技术
- 向量循环/分段开采技术
- 向量递归技术
- 稀疏矩阵的处理技术

链接技术

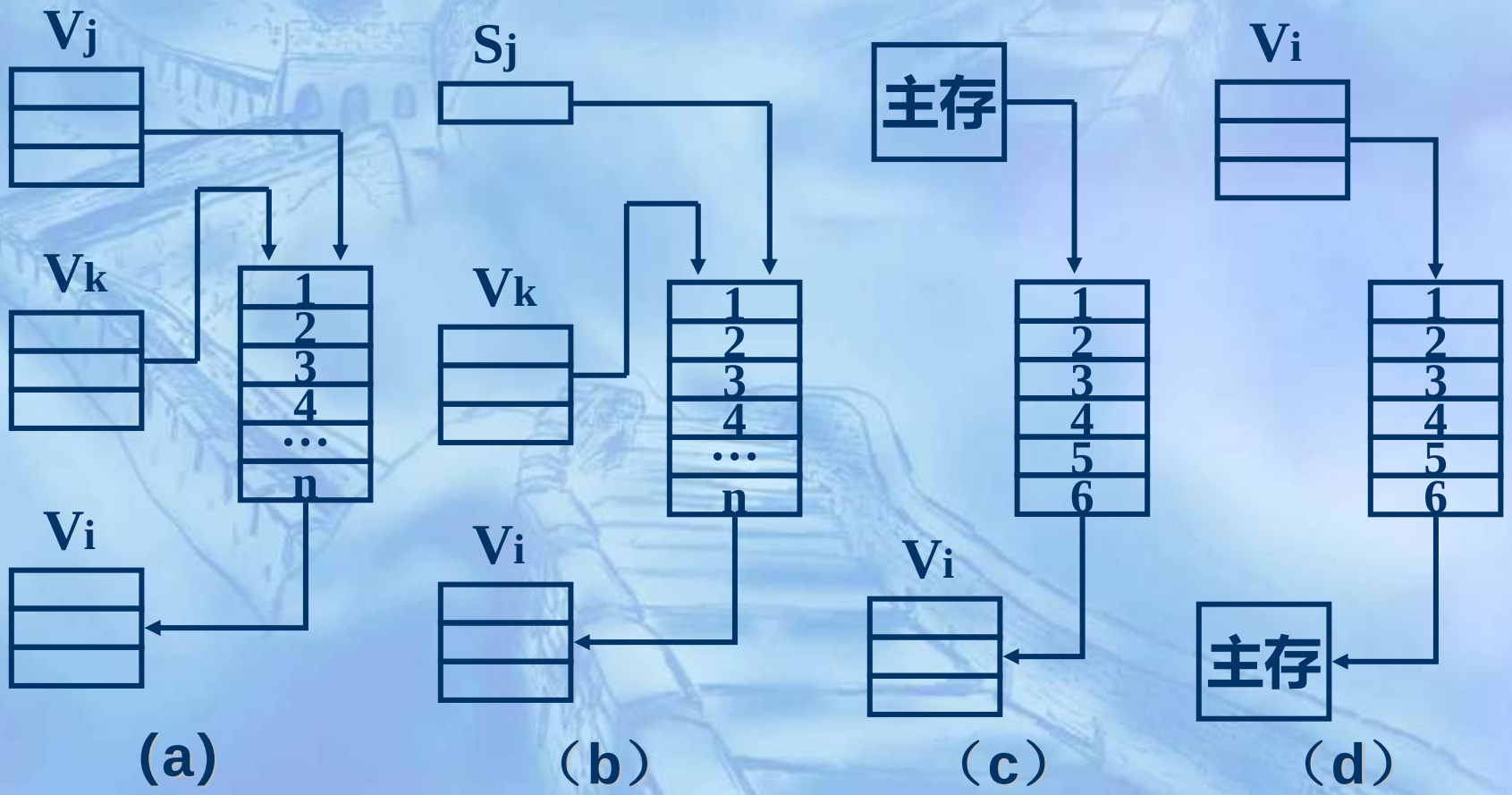
- 向量指令的类型
- 向量运算中的相关和冲突
- 向量链接技术

向量指令的类型

以CRAY-1向量处理机为例，有四类指令：

- **向量与向量操作**：Vi←Vj op Vk
- **向量与标量操作**：Vi←Sj op Vk
- **向量取**：Vi←存储器
- **向量存**：存储器←Vi

向量指令的类型



浮点加6拍、浮点乘7拍

向量运算中的相关和冲突

V0←V1 + V2
V3←V4×V5
(a) 不相关的指令

V0←V1 + V2
V3←V0×V4
(b) 写读数据相关

V0←V1 + V2
V3←V4 + V5
(c) 功能部件冲突

V0←V1 + V2
V3←V1×V4
(d) 读读数据相关

提示：采用顺序发射顺序完成方式。

向量链接技术

—— 基本思想

对于有写读数据相关的向量指令，可以采用“相关专用通道”：**从一个流水线部件得到的结果直接送入另一个流水线部件的操作数寄存器**，这样多条向量指令可以并行执行，这种技术称为流水线的链接技术。

向量链接技术

—— 链接要求

- 没有向量寄存器冲突和运算部件冲突；
- 只有当前一条指令的第一个结果分量送入结果向量寄存器的那一个时钟周期方可链接，否则只能串行执行；
- 若一条向量指令的两个源操作数分别是两条先行指令的结果时，要求：
  - 先行的两条指令产生结果的时间必须相等；
  - 先行的两条指令的向量长度必须相等。
- 要进行链接执行的向量指令的向量长度必须相等，否则无法进行链接。

向量链接技术

—— 举例(要求)

若要进行向量运算：D=A×(B＋C)，假设向量长度≤64，且B和C已由存储器取至V0和V1，则下面3条向量指令即可完成上述运算。

V3←A

V2←V0+V1

V4←V2*V3

向量链接技术

—— 举例(调度一)

- **三条向量指令全部串行执行**

所需时间为：

$$[(1+6+1)+N-1]+[(1+6+1)+N-1]$$
$$+[(1+7+1)+N-1]$$
$$=3N+22\ (拍)$$

注意：CRAY-1启动访存、将元素送往功能部件和将结果存入Vi都需要有1拍的传送延迟。

向量链接技术

— 举例(调度二)

- **前两条并行执行，第三条串行执行**

所需时间为：

$$[(1+6+1)+N-1]+[(1+7+1)+N-1]$$

$$=2N+15\ (拍)$$

注意：CRAY-1启动访存、将元素送往功能部件和将结果存入Vi都需要有1拍的传送延迟。

向量链接技术

—— 举例(调度三)

- **三条向量指令采用链接技术**

所需时间为：

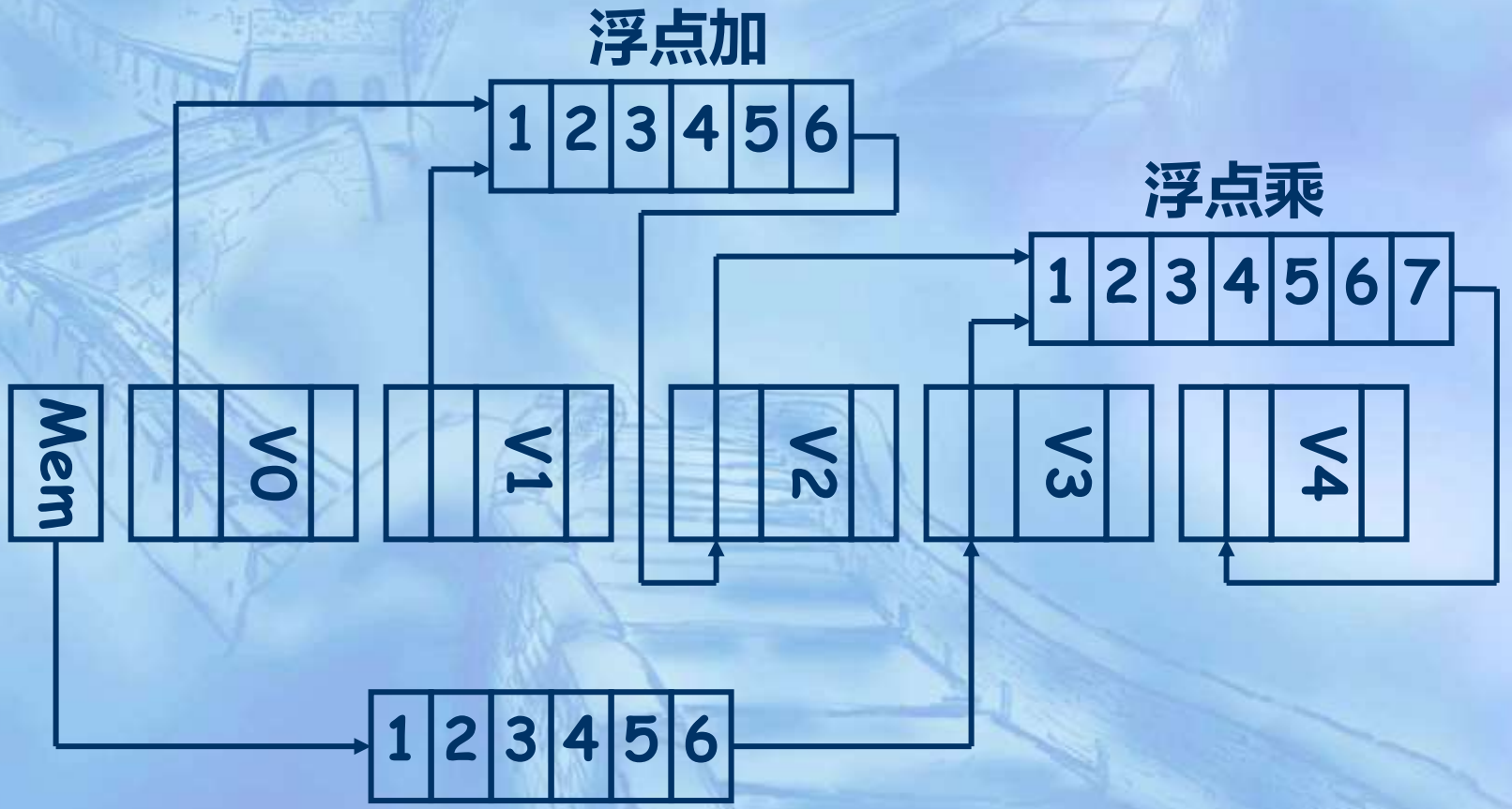
$$(1+6+1)+(1+7+1)+N-1$$

$$=N+16\text{（拍）}$$

注意：CRAY-1启动访存、将元素送往功能部件和将结果存入Vi都需要有1拍的传送延迟。

向量链接技术

— 举例(调度三)



向量循环/分段开采技术

当向量的长度大于向量寄存器的长度时，必须把长向量分成长度固定的段，采用循环结构处理这个长向量，这种技术称为向量循环开采技术，也称为向量分段开采技术。

这种分段和循环由系统硬件和软件控制完成，对于程序员是透明的。

向量循环/分段开采技术

例如： A和B为长度N的向量。

for (i=1; i<N; i++) A[i]=5*B[i]+C;

- **当N为64或更小时，产生A数组的7条指令序列是：**

1：S1←5.0 ；在标量寄存器内设置常数
2：S2←C ；将常数C装入标量寄存器
3：VL←N ；在VL寄存器内设置向量长度
4：V0←B ；将B向量读入向量寄存器
5：V1←S1×V0 ；B数组的每个分量和常数相乘
6：V2←S2＋V1 ；C和5 × B相加
7：A←V2 ；将结果向量存入A数组

向量循环/分段开采技术

例如： A和B为长度N的向量。

for (i=1; i<N; i++) A[i]=5*B[i]+C;

- **当N超过64时，就需要采用向量循环：**

在进入循环以前，把N除以64，以确定循环次数。如果有余数，则在第一次循环中首先计算余数个分量。

循环由第4条到第7条指令组成。

4：V0←B ；将B向量读入向量寄存器

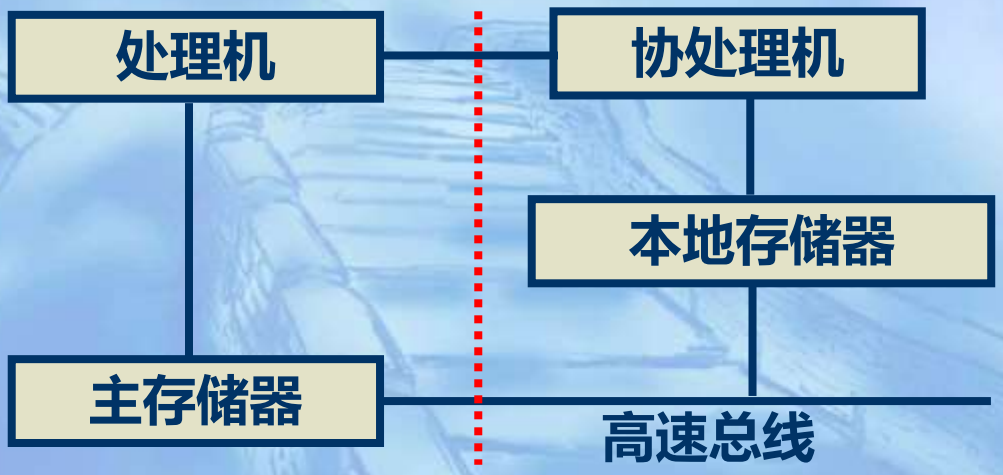
5：V1←S1×V0 ；B数组的每个分量和常数相乘

6：V2←S2＋V1 ；C和5 × B相加

7：A←V2 ；将结果向量存入A数组

向量协处理器

向量协处理器是为解决科学计算所要求的大量向量处理而设计的一种装置。它一般和中、小型计算机组合起来，作为主计算机的外围设备，承担处理向量的任务。这样，就可以得到较高的吞吐率和精度，其价格又可以为一般中、小用户所接受。



带向量协处理器的计算机结构框图

性能评价

- 向量指令处理时间T_{vp}
- 最大性能R_{∞}
- 半性能向量长度$n_{1/2}$
- 向量长度临界值n_v

向量指令处理时间T_{vp}

- 一条向量指令的处理时间
- 一批向量指令的处理时间

一条向量指令的处理时间

$$T_{vp} = T_s + T_{vf} + (n - 1)\,T_c$$
$$= [s + e + (n - 1)]\,T_c$$
$$= [T_{start} + n]\,T_c$$

- T_{vp}：一条向量指令的处理时间；
- T_s：向量流水线的建立时间：s个时钟周期；
- T_{vf}：向量流水线的流过时间：e个时钟周期；
- T_c：流水线的时钟周期时间；
- n：向量长度；
- T_{start}：该向量指令的启动时间：s+e-1个时钟周期，此后每个时钟周期流出一个结果，共有n个结果。

一批向量指令的处理时间

一组向量操作的执行时间主要取决于：**向量的长度、向量操作之间是否存在流水功能部件的冲突**和**数据的相关性**。

把几条能在一个时钟周期内同时开始执行的向量指令称为一个**编队**；同一个编队中的指令一定不存在功能部件冲突和数据相关。

将编队数记作T_{chime}。

一批向量指令的处理时间

- 向量长度≤向量寄存器长度时
- 向量长度 > 向量寄存器长度时

向量长度≤向量寄存器长度时

$$T_{all} = \sum_{i=1}^{m} T_{vp}^{(i)} = \sum_{i=1}^{m} (T_{start}^{(i)} + n)T_c = (\sum_{i=1}^{m} T_{start}^{(i)} + mn)T_c = (T_{start} + mn)T_c$$

- $T^{(i)}_{vp}$：第 i 个编队的执行时间
- m：编队数 T_{chime}
- $T^{(i)}_{start}$：第 i 编队中各指令的启动时间的最大值
- T_{start}：该组指令总的启动时间，$T_{start} = \sum_{i=1}^{m} T_{start}^{(i)}$
- T_c：流水线“瓶颈”段的执行时间；
- n：向量长度；

举 例

—— 问题

在某台向量处理机上执行DAXPY代码(Y=a×X+Y)，代码如下：

```
LV      V1, Rx      ;取向量X
MULTSV  V2,F0,V1    ;向量和标量相乘
LV      V3,Ry       ;取向量Y
ADDV    V4,V2,V3    ;加法
SV      Ry,V4       ;存结果
```

问：这组向量操作能划分成几个编队？假设每种流水功能部件只有一个，且启动时间分别为：取数和存数部件为12个时钟周期、乘法部件为7个、加法部件为6个。请计算完成这一组向量操作所需的总时间为多少？

举 例

—— 解答

可分成4个编队：第1条指令LV为第1个编队，MULTSV指令和第2条LV指令为第2个编队，ADDV指令为第3个编队，SV指令为第4个编在此处键入公式。队。

编队	开始时间	启动时间	结果时间
LV	1	12	12+n
MULTSV、LV	13+n	12	24+2n
ADDV	25+2n	6	30+3n
SV	31+3n	12	42+4n

$$T_{\text{all}} = (12 + 12 + 6 + 12) + 4n = 42 + 4n$$

向量长度＞向量寄存器长度时

需进行分段开采，向量长度为n的一组向量操作的整个执行时间为：

$$T_n = \left\lceil \frac{n}{MVL} \right\rceil \times (T_{loop} + T_{start}) + n \times T_{chime}$$

其中：T_{loop}为执行标量代码的开销，T_{start}为所有编队的向量启动时间，T_{chime}为编队数，MVL是向量寄存器的长度。T_{loop}可以看作是一个常数，Cray 1机的T_{loop}约等于15。

举 例

—— 问题

在某台向量处理机上执行DAXPY代码(Y=a×X+Y)，代码如下：

```
1: LV      V1,Rx      ; 取向量X
2: MULTSV  V2,F0,V1   ; 向量和标量相乘
3: LV      V3,Ry      ; 取向量Y
4: ADDV    V4,V2,V3   ; 加法
5: SV      Ry,V4      ; 存结果
```

向量寄存器长度为64，向量长度为n ，各功能部件的启动时间与上例相同。求总的执行时间。

举 例

—— 解答

指令1、2，指令3、4和指令5分成三个编队，前两个编队中两条指令如采用链接技术执行，则：$T_{chime}=3$，$T_{loop}=15$， MVL=64 ， $T_{start}=12+7+12+6+12=49$。

$$T_n = \left\lceil \frac{n}{MVL} \right\rceil \times (T_{loop} + T_{start}) + n \times T_{chime}$$

$$= \left\lceil \frac{n}{64} \right\rceil \times (15+49) + 3n = \left\lceil \frac{n}{64} \right\rceil \times 64 + 3n$$

$$= \begin{cases} 4n & n \bmod 64 = 0 \\ 4n+64 & n \bmod 64 \neq 0 \end{cases}$$

最大性能R_∞

R_∞表示当向量长度为无穷大时的向量流水线的最大性能。常在评价峰值性能时使用，单位为MFLOPS。可表示为：

$$R_\infty = \lim_{n\to\infty} \frac{\text{浮点运算次数}\times\text{时钟频率}}{\text{循环所花费时钟周期数}} = \frac{\text{浮点运算次数}\times\text{时钟频率}}{\lim\limits_{n\to\infty}\left\lceil \frac{T_n}{n} \right\rceil}$$

其中：n为向量长度；T_n为一组向量操作的整个执行时间。

举 例

— 前例

假设时钟频率为500MHZ。因为每个循环有2个浮点运算操作，所以：

$$R_\infty = \frac{2 \times 500\text{MHZ}}{\lim\limits_{n\to\infty} \frac{4n + 64\text{或}4n}{n}} = \frac{2 \times 500\text{MHZ}}{4} = 250\text{MFLOPS}$$

半性能向量长度$n_{1/2}$

为达到一半R_∞值所需的向量长度称为半性能向量长度$n_{1/2}$，主要评价向量流水线建立时间对性能的影响。

通常希望向量流水线有较小的 $n_{1/2}$，例如：CRAY-1的$n_{1/2} = 10 \sim 20$，CYBER 205的$n_{1/2} = 100$。

举 例

— 前例

$$\frac{1}{2}R_\infty = \frac{\text{浮点运算次数}\times\text{时钟频率}}{\text{循环所花费时钟周期数}} = \frac{\text{浮点运算次数}\times\text{时钟频率}}{\dfrac{T_{n_{1/2}}}{n_{1/2}}}$$

因为：R_∞=250MFLOPS，因此有：

$$250 / 2 = 2\, n_{1/2} / Tn_{1/2} \times 500$$

假设：$n_{1/2} \le 64$，因此：$Tn_{1/2} = 64 + 3\, n_{1/2}$

代入上式解得：$n_{1/2} = 12.8$

所以：$n_{1/2} = 13$

向量长度临界值n_v

n_v表示向量流水方式的工作速度优于标量串行方式工作时所需得向量长度临界值。该参数既衡量建立时间，也衡量标量/向量速度比对性能的影响。

举 例

— 前例

对于前例，若按标量方式工作，则一个循环的执行时间为：10（建立循环的开销）+12（取数）+7（乘法）+ 12（取数）+6（加法）+12（存数）=59（个时钟周期），如按向量方式工作，则总执行时间为：$T_{n<64}=3n+64$。

所以：$59n_v = 3n_v + 64$，解得：

$$n_v = \left\lceil \frac{64}{56} \right\rceil = 2$$