



第13章 阵列处理机

- 13. 1 [阵列处理机的操作模型和特点](#)
- 13. 2 [阵列处理机的基本结构](#)
- 13. 3 [阵列处理机实例](#)
- 13. 4 [阵列处理机的并行算法举例](#)

阵列处理机

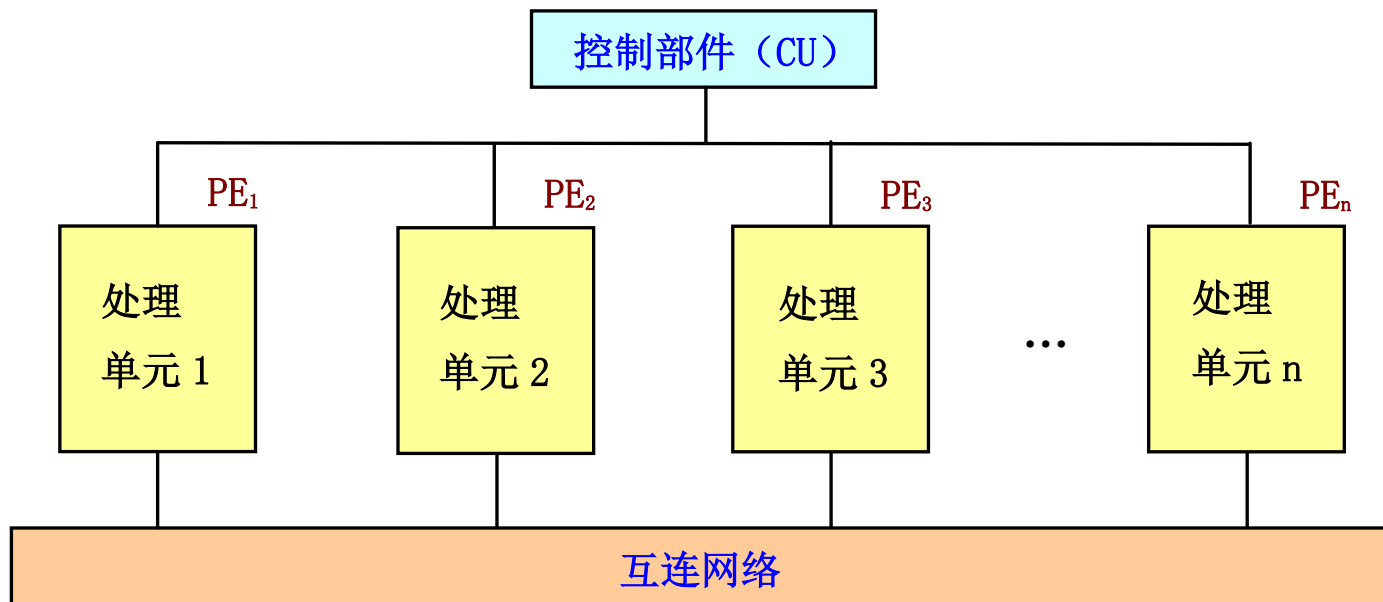
- **核心**：一个由多个处理单元构成的阵列
- 采用资源重复的方法，设置较多的处理单元来提高并行性。
- 用单一的控制部件来控制多个处理单元对各自的数据进行相同的运算和操作。
 - 又称为**SIMD**计算机。
- 有时还被称为**并行处理机**。

13.1 阵列处理机的操作模型和特点

1. 阵列处理机的操作模型

- 用一个控制部件CU同时管理多个处理单元PE。
- CU对指令进行译码，并把指令播送到各处理单元。
- 所有处理单元均被动地接收并执行从控制部件广播来的同一条指令，但它们所操作的对象却是不同的数据。

13.1 阵列处理机的操作模型和特点



阵列处理机的操作模型

13.1 阵列处理机的操作模型和特点

2. 阵列处理机的操作模型可用五元组表示

阵列处理机 = (N, C, I, M, R)

其中：

- N：机器的处理单元（PE）数。

例如：Illiac IV计算机有64个PE

MP-1计算机有16384个PE

- C：控制部件CU直接执行的指令集，包括标量指令和程序流控制指令。
- I：由CU广播至所有PE进行并行执行的指令集。
 - 包括算术运算、逻辑运算、数据寻径、屏蔽以及其他由每个PE对它的数据所执行的局部操作。

13.1 阵列处理机的操作模型和特点

➤ M：屏蔽方案集

- 每种屏蔽将所有PE划分成允许操作和禁止操作两种工作模式。

➤ R：数据寻径功能集

- 说明互连网络中PE间通信所需要的各种设置模式。

例如：MasPar MP-1计算机的操作特性如下：

- (1) MP-1是一种SIMD机器，其PE数 $N=1024\sim 16384$ 。
- (2) CU执行标量指令，将译码后的向量指令广播到PE阵列，并控制PE间通信。
- (3) 每个PE都是RISC处理机，能执行不同数据的整数运算和标准浮点运算。PE从CU接收指令。

13.1 阵列处理机的操作模型和特点

(4) 屏蔽方案设在每个PE中，并由CU连续监控，它能在运行时动态地使每个PE处于工作或禁止状态。

(5) MP-1有一个X-Net网格网络和一个全局多级交叉开关寻径器，以实现CU-PE之间、X-Net的8个近邻和全局寻径器的通信。

2. 阵列处理机的特点

- 以单指令流多数据流方式工作。
- 通过设置多个相同的处理单元来开发并行性。
 - 利用并行性中的同时性，而不是并发性。所有处理单元必须同时进行相同的操作。
- 以某一类算法为背景的专用计算机。



13.1 阵列处理机的操作模型和特点

- 阵列机的研究必须与并行算法的研究密切结合，以便能充分发挥它的处理能力。
- 阵列机的控制器实质上是一台标量处理机，而为了完成I/O操作以及操作系统的管理，尚需一个前端机。

实际的阵列机系统是由3部分构成的一个异构型多处理机系统。

13.2 阵列处理机的基本结构

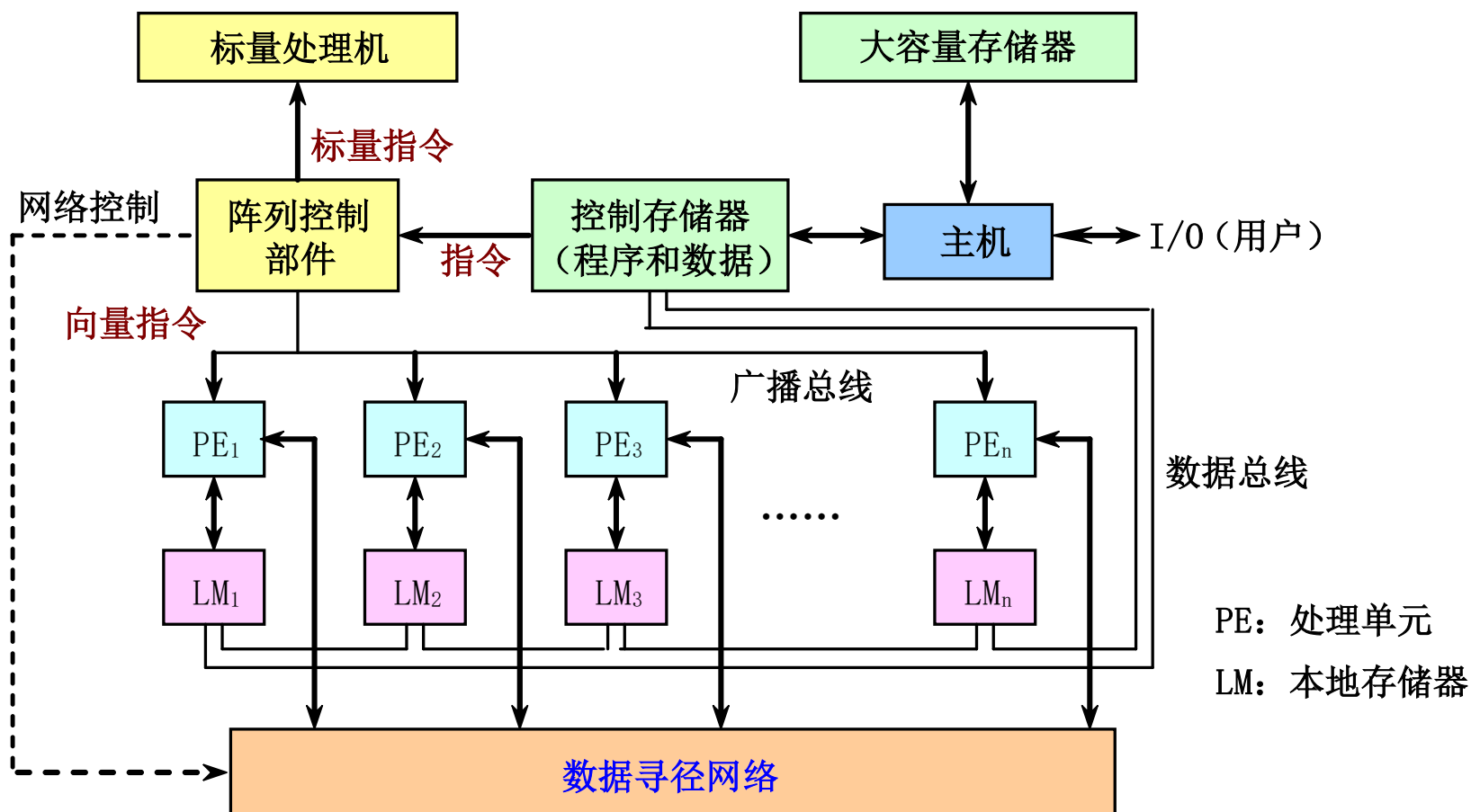
13.2.1 分布式存储器的阵列机

1. 分布式存储器的阵列机结构

- 含有多个相同的处理单元PE，每个PE有各自的本地存储器LM。
- PE之间通过数据寻径网络以一定方式互相连接。它们在阵列控制部件的统一指挥下，实现并行操作。
- 指令的执行顺序基本上是串行进行的。
- 程序和数据是通过主机装入控制存储器。



13.2 阵列处理机的基本结构



分布式存储器的阵列处理机结构

13.2 阵列处理机的基本结构

2. 指令送到控制部件进行译码。

- **标量指令**：直接由标量处理机执行。
- **向量指令**：阵列控制部件通过广播总线将它广播到所有PE中去并行地执行。

3. 执行程序所需的数据集经划分后通过数据总线分布存放到各PE的本地存储器LM。

4. 各PE之间通过数据寻径网络互连，实现PE间的通信，控制部件通过执行程序来控制数据寻径网络。

5. PE的同步是在控制部件的控制下由硬件实现。

- 可以让所有PE在同一个周期执行同一条指令



13.2 阵列处理机的基本结构

- 也可以通过采用屏蔽逻辑来控制某些PE在指定的指令周期是否参与执行

6. 各种阵列处理机的主要差别

在于数据寻径网络的不同。

- Illiac IV： 4-邻连接网络结构
(在过去是最常用的一种)
- CM-2： 嵌在网格中的超立方体
- MasPar MP-1： X-Net加多级交叉开关寻径器



13.2 阵列处理机的基本结构

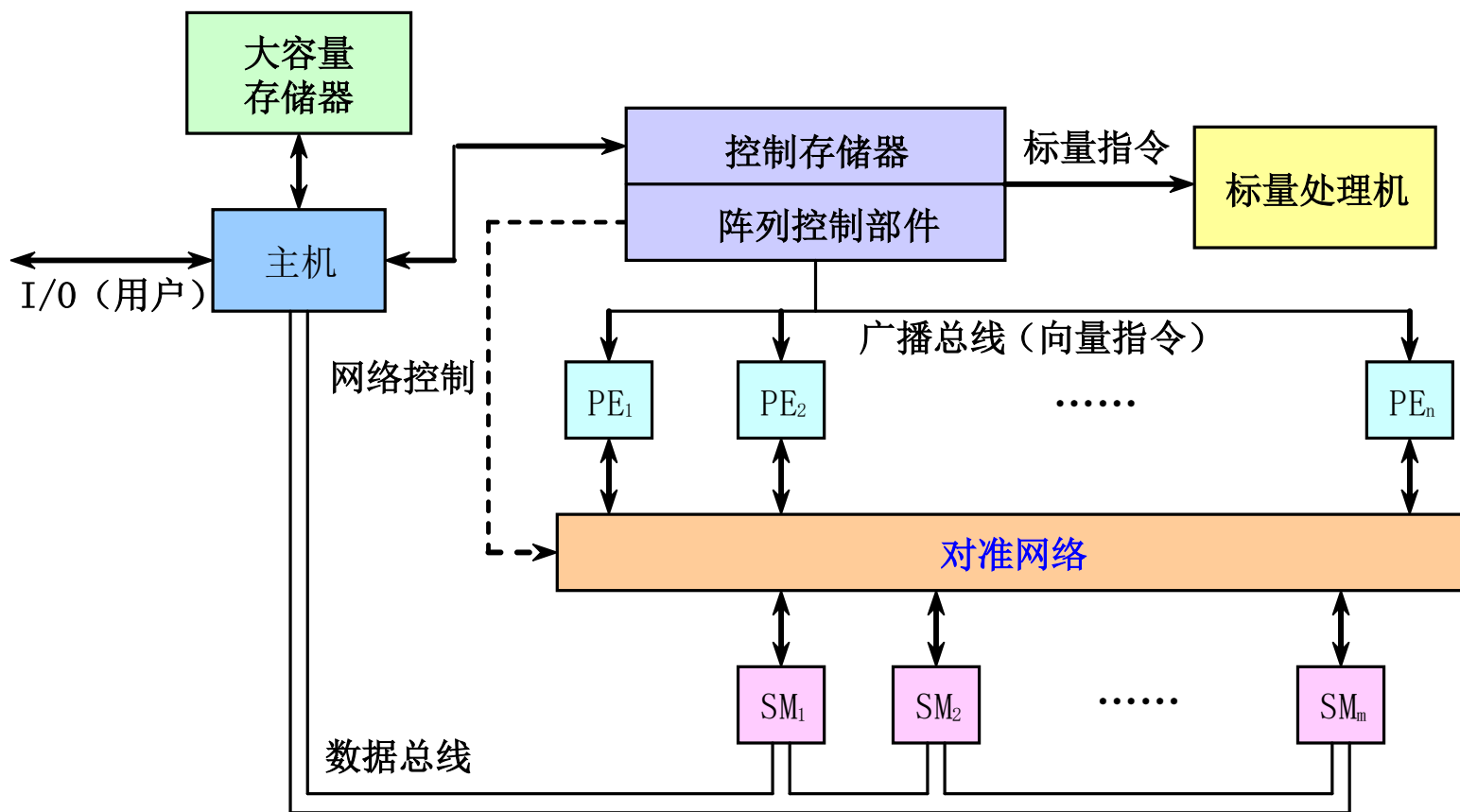
13.2.2 共享存储器的阵列机

共享存储器的阵列处理机结构

- 集中设置存储器
 - 共享的多体并行存储器SM通过对准网络与各处理单元PE相连。
 - 存储模块的数目等于或略大于处理单元的数目。
- 必须减少存储器访问冲突
(将数据合理地分配到各存储器模块中)
- 在处理单元数目不太多的情况下是很理想的
- 所有阵列指令都必须使用长度为n的向量操作数
(n为PE的个数)



13.2 阵列处理机的基本结构



共享存储器的阵列处理机结构

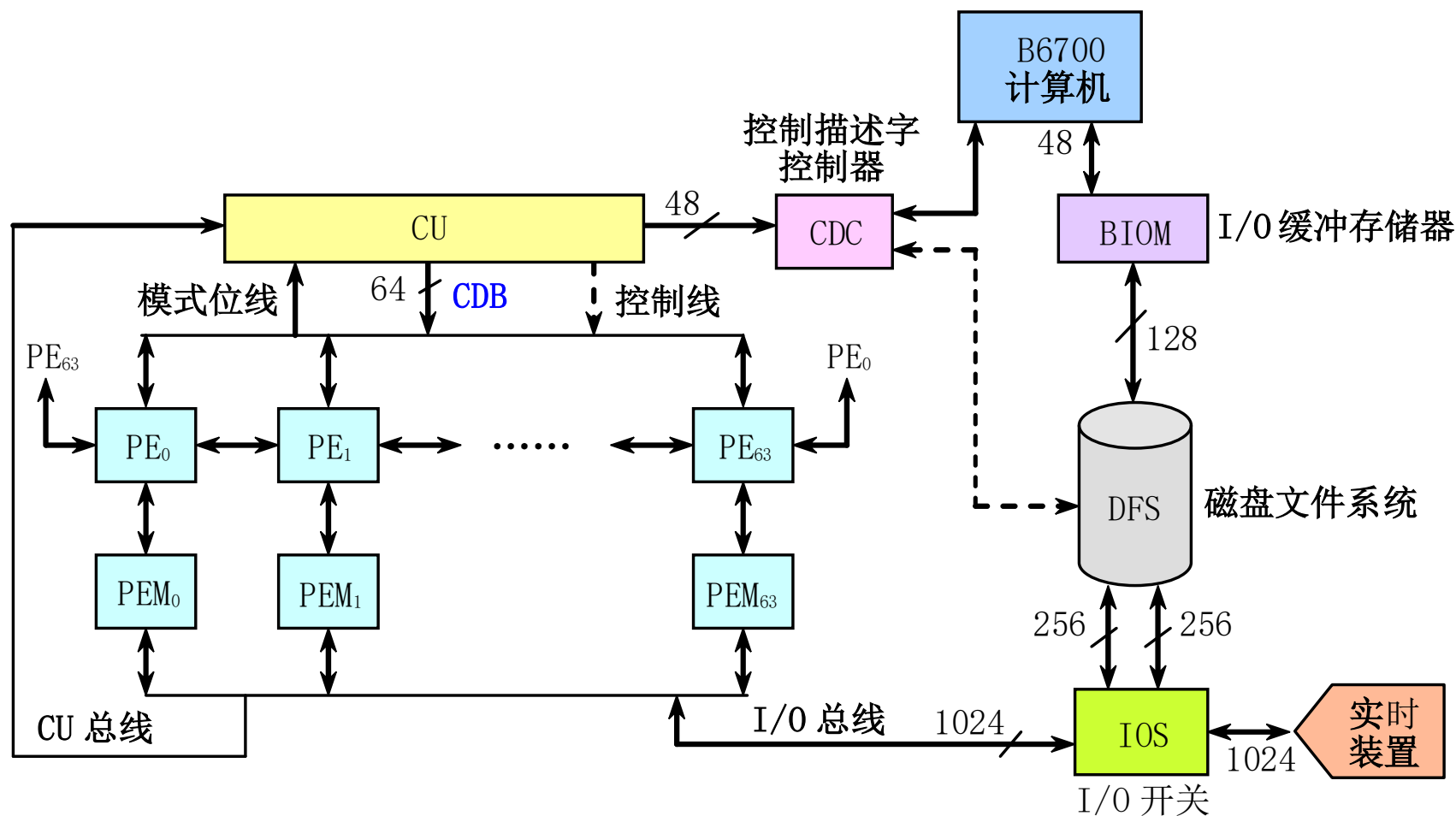
互连网络是共享存储器SM和处理单元PE之间的必由之路。

13.3 阵列处理机实例

13.3.1 实例1：Illiac IV阵列处理机

- 美国宝来公司和伊利诺大学合作研制 1972年
- 最早的阵列处理机
- 一个由3种类型处理机联合组成的多机系统
 - 处理单元阵列：专门用于数组运算
 - 阵列控制器（CU）：既是处理单元阵列的控制部分，又可以看作是一台相对独立的小型标量处理机。
 - 一台标准的B6700计算机：担负Illiac IV输入输出系统和操作系统管理功能

13.3 阵列处理机实例



Illiac IV系统总框图

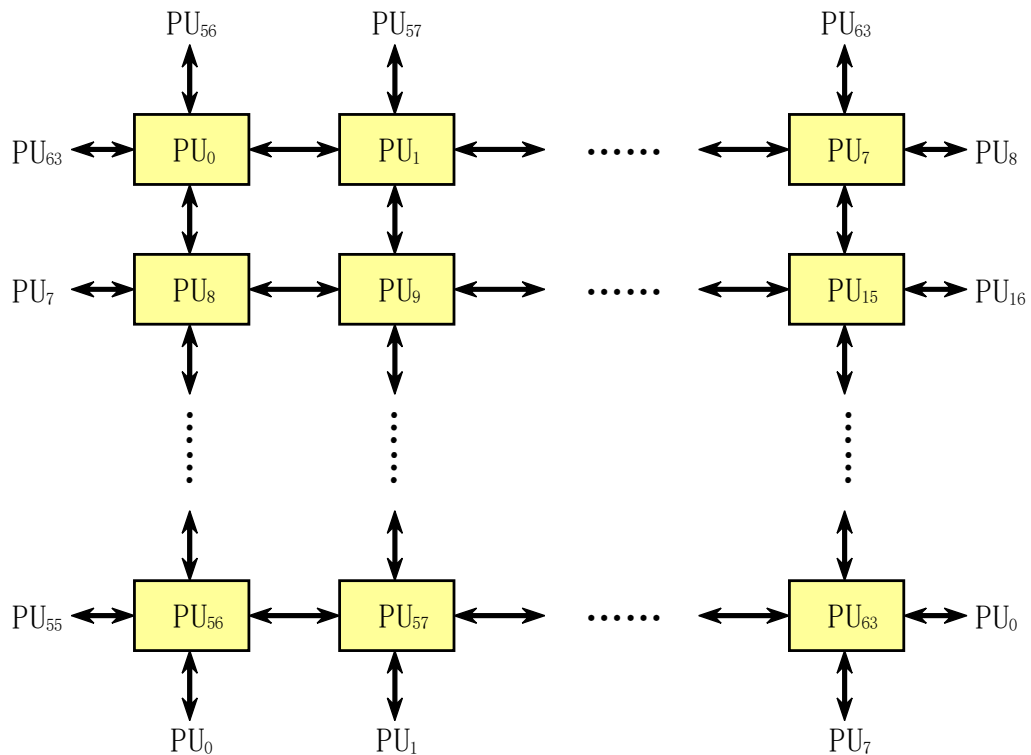
13.3 阵列处理机实例

1. Illiac IV阵列

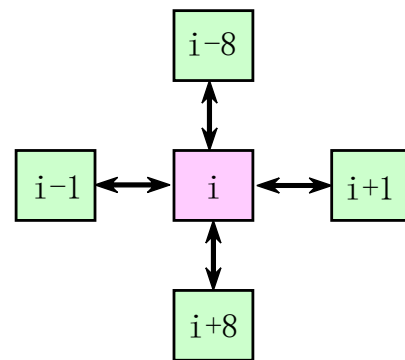
- 由64个处理单元（PE）、64个本地存储器（PEM）和存储器逻辑部件（MLU）组成；
- 把每个PE和PEM对看成是一个处理部件PU；
 - 64个处理部件 $PU_0 \sim PU_{63}$ 排列成一个 8×8 方阵
- Illiac IV的阵列结构又称为闭合螺线阵列；
- 既便于一维长向量（多至64个元素）的处理，又便于二维数组运算，以缩短处理单元之间的路径距离。
 - 步距不等于 ± 1 或 ± 8 的任意处理单元间通信可用软件方法寻找最短路径，其最短距离都不会超过7步。



Illiac IV处理部件的连接



(a)



(b)

例如：从 PU_{10} 到 PU_{46} 的距离以下列路径为最短

$$\mathbf{PU}_{10} \rightarrow \mathbf{PU}_9 \rightarrow \mathbf{PU}_8 \rightarrow \mathbf{PU}_0 \rightarrow \mathbf{PU}_{63} \rightarrow \mathbf{PU}_{62} \rightarrow \mathbf{PU}_{54} \rightarrow \mathbf{PU}_{46}$$

13.3 阵列处理机实例

- 一般情况， $n \times n$ 个单元组成的阵列中，任意两个处理单元之间的最短距离不会超过 $(n-1)$ 步。
- 每个处理单元有6个可编程序寄存器
 - 64位字长的累加器RGA
 - 64位字长的操作数寄存器RGB
 - 64位字长的数据路由寄存器RGR
 - 64位字长的通用寄存器RGS
(可被程序用来暂存中间结果)
 - 16位的变址寄存器
 - 8位的模式寄存器
(存放PE屏蔽信息以及状态位)



13.3 阵列处理机实例

➤ 运算部件

- 加/乘算术单元
- 逻辑单元
- 移位单元
- 地址加法器等

➤ 操作数来源

- PE本身的寄存器
- PEM
- CU的公共数据总线
- PE的4个近邻



13.3 阵列处理机实例

- 并行的加法速度

每秒1010次8位定点加法或 150×106 次64位浮点加法

- 每一个处理单元有一个自己的本地存储器PEM

- PE和PEM之间经过存储器逻辑部件MLU相连

2. 阵列控制器CU

- 一台小型计算机

- 对阵列的处理单元进行控制
- 利用本身的内部资源执行一整套指令，用以完成标量操作。

- 功能



13.3 阵列处理机实例

- 对指令流进行控制和译码，包括执行一整套标量指令；
 - 向各处理单元发出执行数组操作指令所需的控制信号；
 - 产生并向所有处理单元广播公共的地址部分；
 - 产生并向所有处理单元广播公共的数据；
 - 接收和处理由各PE计算出错、系统I/O操作以及B6700所产生的陷阱中断信号。
- 阵列控制器CU与处理单元之间有4条信息通路
- CU总线
 - 公共数据总线CDB
 - 模式位线
 - 指令控制线（大约有200根）

13.3 阵列处理机实例

3. 输入输出系统

由磁盘文件系统DFS、I/O分系统和B6700管理计算机组成。

➤ 磁盘文件系统DFS

- 两套大容量并行读写磁盘系统及其相应的控制器；
- 每套有13台磁盘机，总容量为109位；
- 每台磁盘机有128道，每道一个磁头，并行读写，数据宽度为256位，最大传输率为 $502 \times 106\text{b/s}$ ；平均等待时间为19.6ms；
- 如果两个通道同时发送或接收数据，则数据宽度为512位，最大传输率为109b/s。

13.3 阵列处理机实例

➤ I/O系统

包括3部分：

- 输入/输出开关IOS
 - 作为一个开关，把DFS或可能连上的实时装置转接到阵列存储器，进行大批数据的I/O传送；
 - 作为DFS和PEM之间的缓冲，以平衡两边不同的数据宽度。
- 控制描述字控制器CDC
 - 对阵列控制器CU的I/O请求进行管理
- BIOM
 - 在DFS和B6700之间，是为了取得二者之间传送带宽上的匹配。



13.3 阵列处理机实例

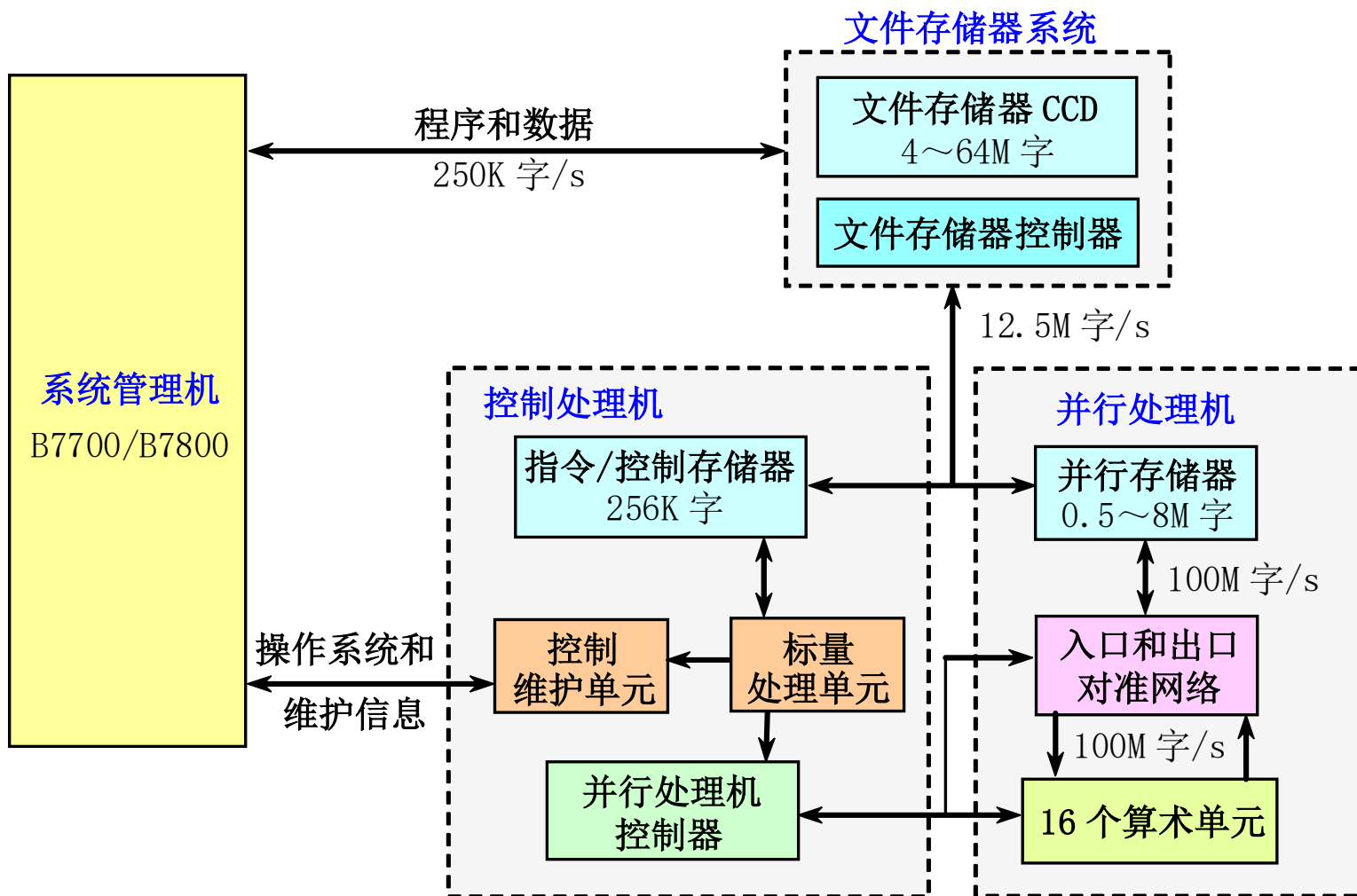
➤ B6700管理计算机

- 管理全部系统资源，完成用户程序的编译或汇编，
- 为Illiac IV进行作业调度、存储分配、产生I/O控制描述字送至CDC、处理中断、提供操作系统所具备的其他服务等。

13.3.2 实例2：BSP计算机

- 美国宝来公司和伊利诺依大学 1979年
- 共享存储器结构的SIMD计算机的典型代表
- 最高处理性能：每秒5千万次浮点运算
- 依靠并行性来提高性能





BSP计算机系统的框图

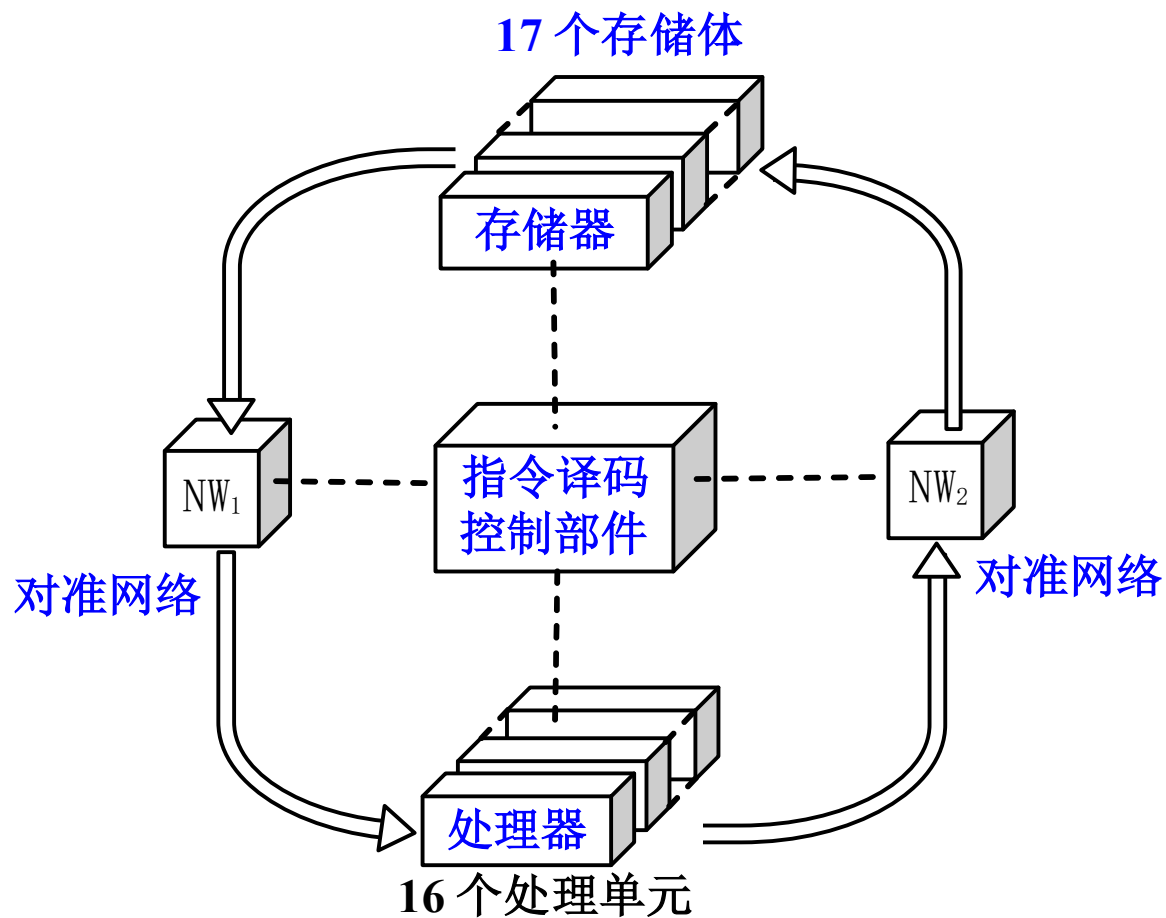
13.3 阵列处理机实例

- BSP处理机由3部分构成：控制处理机，并行处理机，文件存储器。

1. BSP处理机

- 并行处理机
 - 包含16个算术单元AE、由17个存储体组成的一个无冲突访问的并行存储器和两套对准网络（分别为入口和出口对准网络）
 - 一条5级的数据流水线
 - 从17个存储器输出端口并行读出16个操作数；
 - 经对准网络NW₁将16个操作数重新排列，形成16个算术单元所需要的顺序；

13.3 阵列处理机实例



BSP的5级数据流水线结构示意图

13.3 阵列处理机实例

- 将排列好的16个操作数送到16个算术单元进行处理；
 - 所得的16个结果经对准网络 NW_2 重新排列成在17个存储体中存储所需要的次序；
 - 写入并行存储器。
- 两套对准网络的作用：在读或写并行存储器时，使并行存储器中为保证无冲突访问而错开存放的操作数顺序能够与算术单元并行处理所要求的正常顺序协调一致。
- 这种流水线对提高系统处理效率有很大作用。
- 有效地实现了处理单元、存储器和互连网络在时间上重叠工作，在理想情况下能取得带宽的完全匹配。



13.3 阵列处理机实例

- 可把大于16的任意长度的向量按16个分量的标准长度分为若干段，依次在时间上重叠起来进行处理。
- 实现不同向量指令的重叠执行。
- 数据保存在由17个存储体组成的并行存储器中，每个存储体的容量可达512K字，存储周期为160ns。
(一个无冲突访问存储器)

➤ 控制处理机

控制并行处理机，提供与系统管理机相连的接口。

- 标量处理单元：处理存储在指令/控制存储器中的全部操作系统和用户程序的指令。

13.3 阵列处理机实例

- 全部的向量指令以及某些成组运算的标量指令被送给并行处理机控制器。在经过合格性检查之后，并行处理机控制器将指令转换为微操作序列去控制16个AE操作。
- 指令/控制存储器的容量为256K字，存储周期为160ns，字长为56位，其中8位是校验位，提供单错校正和双错检测的能力。
- 控制维护单元：系统管理机与控制处理机的接口，用来对控制处理机进行初始化以及监控命令的通信和维护。

13.3 阵列处理机实例

➤ 文件存储器

- ❑ **BSP**直接控制下的唯一外围设备。
- ❑ **BSP**程序执行过程中所产生的暂存文件和输出文件都是先存放在文件存储器中，然后才被送给系统管理机，输出给用户。
- ❑ 文件存储器的数据传输率较高，大大缓解了**I/O**受限问题。

2. **BSP**并行存储器

- 由**17**个存储体组成
- 可以实现**无冲突访问**



13.3 阵列处理机实例

实现无冲突访问的硬件支持：

- 质数个存储器端口（存储体数是质数17）
- 存储端口和AE之间的交叉开关（对准网络）
- 特殊的存储器地址生成机构

讨论一台含N个AE和M个存储体的类BSP机的情况。

➤ 地址映像规则

- 先将二维数组按列优先或者按行优先的顺序变换为一维数组，以形成一个一维线性地址空间，地址用A表示。
- 然后将地址A变换成并行存储器地址 (i, j) 。
其中：j 是存储体体号， $j = A \pmod{M}$

13.3 阵列处理机实例

- **i**: 在相应存储体内的地址, $i = \left\lfloor \frac{A}{N} \right\rfloor$ 。
- 存储体的个数**M**是一个质数。

➤ 一个比较简单的例子

设并行存储器的体数**M=7**（质数），运算单元数**N=6**。

考虑下述**4×5**的数组：

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$$

13.3 阵列处理机实例

按上述地址映像规则，将这个 4×5 二维数组在 $M=7$ 、 $N=6$ 的并行存储器中存储的情况：

数组元素	a_{00}	a_{10}	a_{20}	a_{30}	a_{01}	a_{11}	a_{21}	a_{31}	a_{02}	a_{12}	a_{22}	a_{32}	a_{03}	a_{13}	a_{23}	a_{33}	a_{04}	a_{14}	a_{24}	a_{34}
线性地址 A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
体号 j	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0	1	2	3	4	5
体内地址 i	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3

存储体号 j

	0	1	2	3	4	5	6
0	0 a_{00}	1 a_{10}	2 a_{20}	3 a_{30}	4 a_{01}	5 a_{11}	
1	7 a_{31}	8 a_{02}	9 a_{12}	10 a_{22}	11 a_{32}		6 a_{21}
2	14 a_{23}	15 a_{33}	16 a_{04}	17 a_{14}		12 a_{03}	13 a_{13}
3	21	22	23		18 a_{24}	19 a_{34}	20
4	28	29		24	25	26	27

体内地址 i

13.4 阵列处理机的并行算法举例

以Illiac IV为例，讨论阵列处理机的算法。

1. 有限差分问题

- 把一个有规则的网格覆盖在整个场域上，用网格点上的变量值写出差分方程组以代替场方程来进行计算。
- 描述平面场的拉普拉斯方程

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0$$

13.4 阵列处理机的并行算法举例

- 将二阶偏导数表示为差分形式

$$\frac{\partial^2 U}{\partial x^2} = \frac{U(x+h, y) - 2U(x, y) + U(x-h, y)}{h^2}$$

$$\frac{\partial^2 U}{\partial y^2} = \frac{U(x, y+h) - 2U(x, y) + U(x, y-h)}{h^2}$$

- 代入原方程，则可得有限差分计算公式

$$U(x, y) = \frac{U(x+h, y) + U(x, y+h) + U(x-h, y) + U(x, y-h)}{4}$$

(**x**, **y**) : 平面网格点坐标 **h**: 网格间距



13.4 阵列处理机的并行算法举例

- 差分法求解的精度与网格间距有直接的关系，网格越小，精度越高，但求解所花费的时空开销越大。
- Illiac IV在计算时，是把内部网格点分配给各个处理单元的。因此，上述计算过程可以并行地完成，从而大幅度地提高处理速度。

2. 矩阵加

考虑两个 8×8 的矩阵A和B的相加，所得结果矩阵C也是一个 8×8 的矩阵。

13.4 阵列处理机的并行算法举例

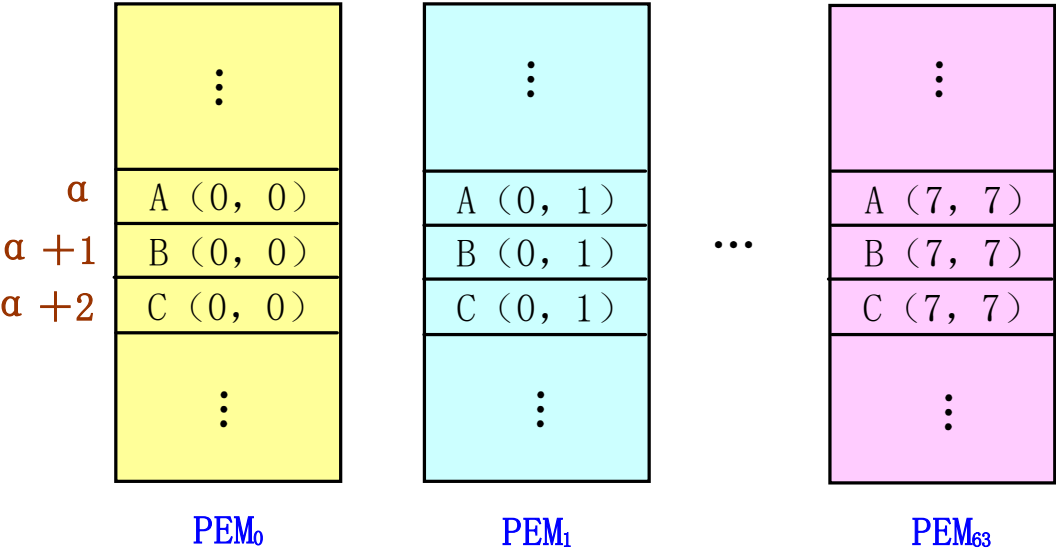
- 把A、B、C中位于相应位置的分量存放在同一PEM内。

假设：

- A的分量在全部64个PEM中存放的单元地址都是 α ；
 - B的全部分量的地址都是 $\alpha+1$ ；
 - C的全部分量的地址都是 $\alpha+2$ 。
- 用3条Illiac IV的汇编指令就可以实现矩阵相加。

13.4 阵列处理机的并行算法举例

LDA ALPHA ; 全部A的分量由PEM_i送PE_i的累加器RGA_i
ADRN ALPHA + 1 ; 全部B的分量与 (RGA_i) 进行浮点加,
 结果送RGA_i
STA ALPHA + 2 ; 全部 (RGA_i) 由PE_i送PEM_i的 $\alpha + 2$ 单元



矩阵相加存储器分配举例



13.4 阵列处理机的并行算法举例

3. 矩阵乘

设A、B和C为3个 8×8 的二维矩阵。若给定A和B，则 $C=A*B$ 的64个分量可利用下列公式计算。

$$c_{ij} = \sum_{k=0}^7 a_{ik} b_{kj} \quad 0 \leq i, j \leq 7$$



13.4 阵列处理机的并行算法举例

- 在SISD计算机上求解，执行下列FORTRAN程序：

```
DO 10 I=0, 7
```

```
DO 10 J=0, 7
```

```
C (I, J) =0
```

```
DO 10 K=0, 7
```

```
10 C(I, J) =C(I, J)+A(I, K)*B(K, J)
```

三重循环，每重循环执行8次，共需512次乘加的时间。

- 在SIMD阵列处理机上求解这个问题

13.4 阵列处理机的并行算法举例

执行下列FORTRAN程序：

```
DO 10 I=0, 7
```

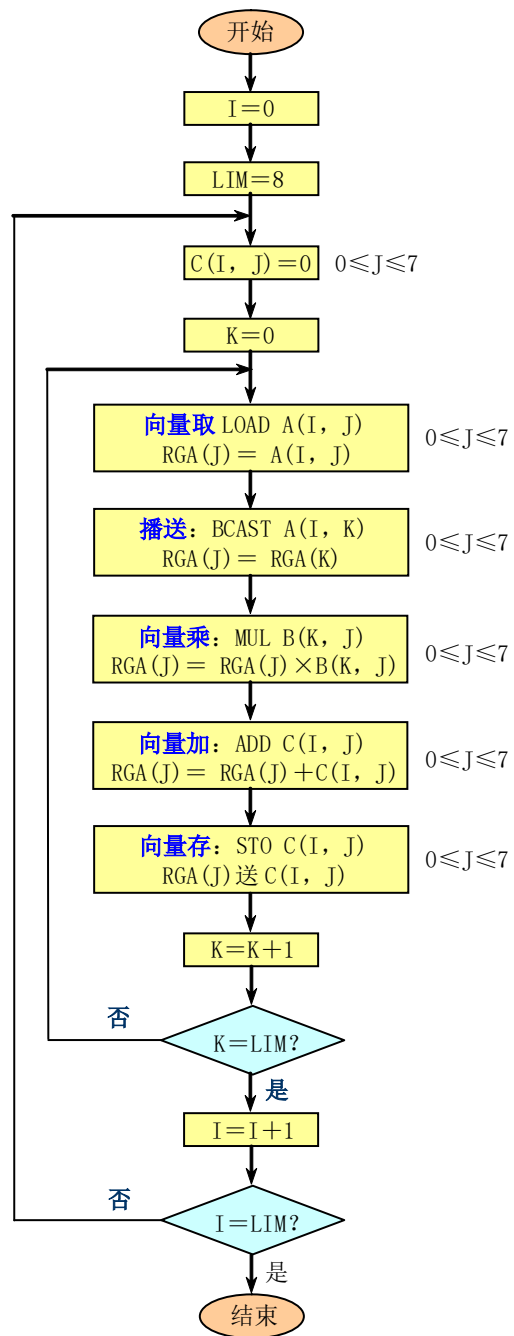
```
  C (I, J) =0
```

```
DO 10 K=0, 7
```

```
  10 C(I, J) =C(I, J)+A(I, K)*B(K, J)
```

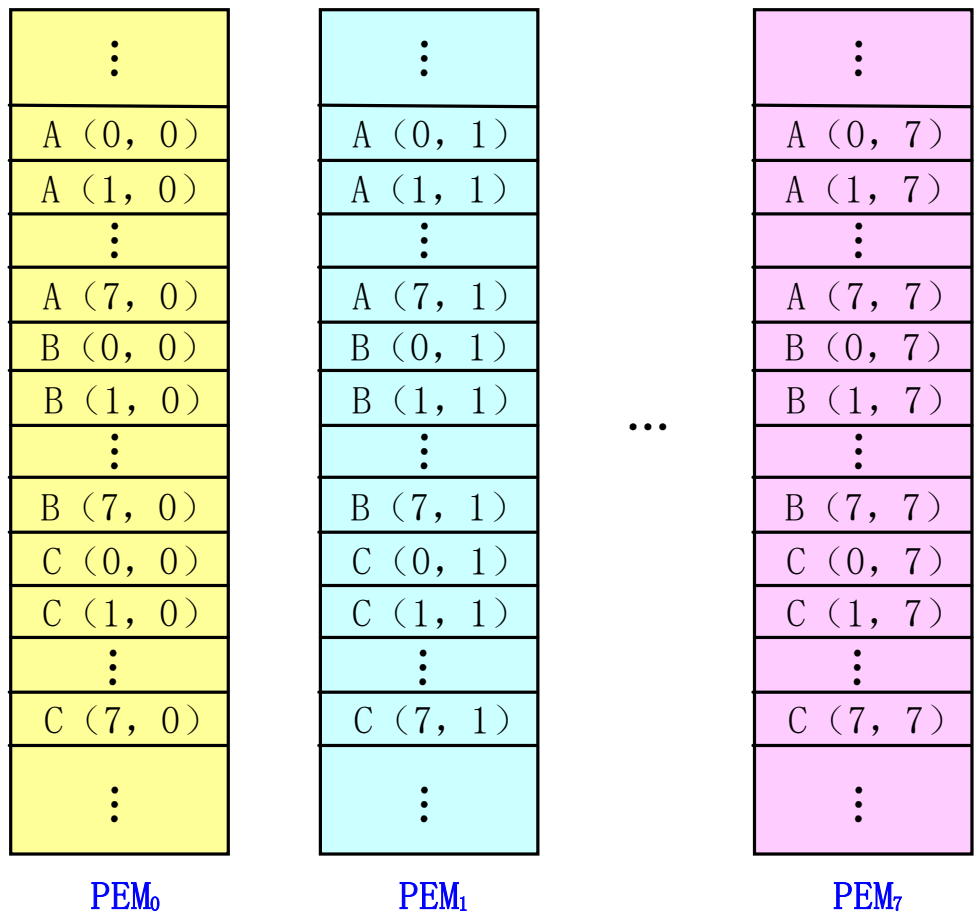
速度提高到原来的8倍，即每个处理单元的计算时间缩短为64次乘加时间。

程序流程图：



13.4 阵列处理机的并行算法举例

➤ A、B、C向量在处理部件存储器中的存放



13.4 阵列处理机的并行算法举例

4. 累加和

一个将N个数的顺序相加转变为并行相加的问题。

- 只有处于活动状态的处理单元才能执行相应的操作。
- 取 $N=8$ 。即有8个数 $A(I)$ 要顺序累加 ($0 \leq I \leq 7$)
- 在SIMD计算机上可写成下列FORTRAN程序：

```
C=0
```

```
DO 10 I=0, 7
```

```
10 C=C+A(I)
```

这是一个串行程序，共要进行8次加法。



13.4 阵列处理机的并行算法举例

- 在阵列处理机上采用成对递归相加的算法，则只需 $\log_2 8 = 3$ 次加法。

首先，把原始数据 $A(I)$ ， $0 \leq I \leq 7$ ，分别放到8个PEM的 α 单元中，

然后按照下面的步骤求累加和：

- 置全部 PE_i 为活动状态， $0 \leq i \leq 7$ ；
- 全部 $A(I)$ ， $0 \leq I \leq 7$ ，从 PEM_i 的 α 单元读到相应 PE_i 的累加寄存器 RGA_i 中， $0 \leq i \leq 7$ ；
- 令 $K=0$ ；
- 将全部 PE_i 的 (RGA_i) 传送到 RGR_i ， $0 \leq i \leq 7$ ；

13.4 阵列处理机的并行算法举例

- 全部 PE_i 的 (RGR_i) 经过互连网络向右传送 2^K 步距,
 $0 \leq i \leq 7$;
- $j = 2^K - 1$;
- 置 PE_0 至 PE_j 为不活动状态;
- 处于活动状态的所有 PE_i 执行;
$$(RGA_i) = (RGA_i) + (RGR_i) \quad j < i \leq 7$$
- $K = K + 1$;
- 若 $K < 3$, 则转回第四步, 否则继续往下执行;
- 置全部 PE_i 为活动状态, $0 \leq i \leq 7$;
- 全部 PE_i 的 (RGA_i) 存入相应的 PEM_i 的 $\alpha + 1$ 单元中,
 $0 \leq i \leq 7$ 。

13.4 阵列处理机的并行算法举例

计算过程示意图：

