第六章 数据级并行技术

.

※本章主要内容

(1) 向量处理机　　(教材第4章，△)

向量处理方式，向量机结构，向量处理性能的优化

(2) 阵列处理机　　(教材第13章，◇)

操作模型，阵列机结构，并行算法；多媒体SIMD技术

(3) GPU(图形处理器)　　(量化分析第4.4节，△)

应用概述，基本结构，编程模型

※总体要求

掌握并行计算机相关概念，理解多媒体SIMD技术

第1节　向量处理机

※主要内容：向量处理方式，处理机结构，提高性能的技术

一、向量处理机的向量处理方式　（流水方式处理各分量）

示例：D＝A×(B＋C)，向量长度为N，分量操作以流水方式实现

***横向处理方式：**

方法—每次处理完一个目标分量　如$\{e_i=b_i+c_i，d_i=a_i\times e_i\}$

特点—有N次RAW冒险、2N次功能切换，不适于向量处理机

***纵向处理方式：**

向量应放在MEM中(向量长度不受限制)

方法—优先处理完一个子操作　如E＝B＋C、D＝A×E

特点—有0次RAW冒险、1次功能变换，适于向量处理机

***纵横处理方式：**

向量可放在REG中(处理速度快)

方法—对向量分组(N/n个组)，组内纵向处理、组间横向处理

特点—有0次RAW冒险、2N/n次功能变换，适于向量处理机

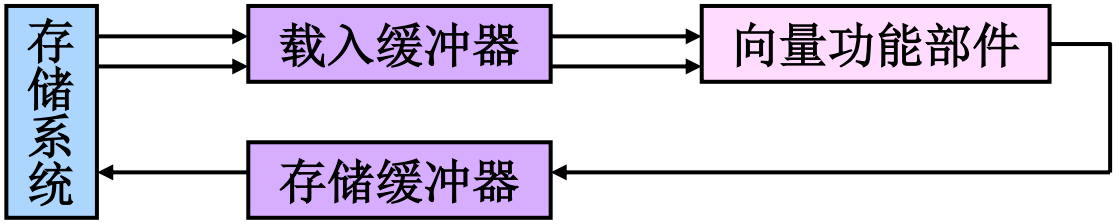
二、向量处理机结构

向量指令功能—OPD类型为向量，可处理向量的所有分量

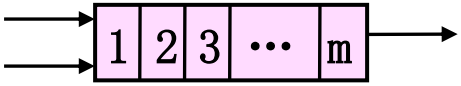
1、存储器-存储器结构

*目标：向量指令中向量长度不受限(向量放在MEM中[使用纵向处理方式])

*基本结构：　(标量部件未画出)



向量功能部件—流水化的标量部件(每拍流出1个结果)



功能部件(操作流水线)

向量存取部件—流水方式的MEM控制部件(多个数据/拍)

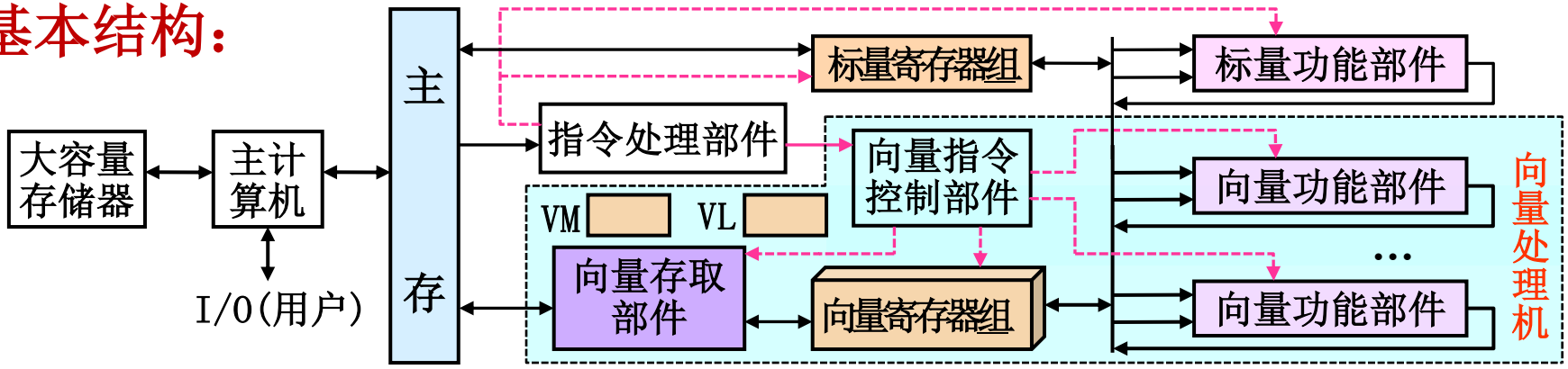
*特点：对MEM带宽要求较高，性能不理想(受限于MEM)

└←常采用多体交叉+多体并行结构

2、寄存器-寄存器结构

*目标：向量处理性能较高(向量放在REG中[使用纵横处理方式])

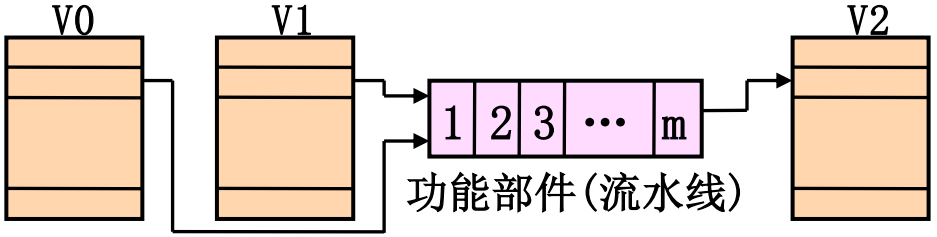
*基本结构：

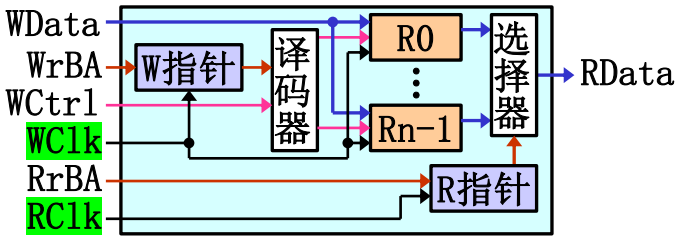


指令处理部件—实现取指、译码，直接控制标量指令

向量功能部件—流水化的标量部件(同MEM-MEM型向量机)

向量寄存器—标量REG数组，每拍可读和写1个分量(下标可不同)





向量长度/屏蔽寄存器—控制操作中分量的个数/是否参与

3、向量指令

***向量指令的功能：** 运算、规约、存取、压缩&还原(稀疏矩阵)等

└←≈标量指令的循环

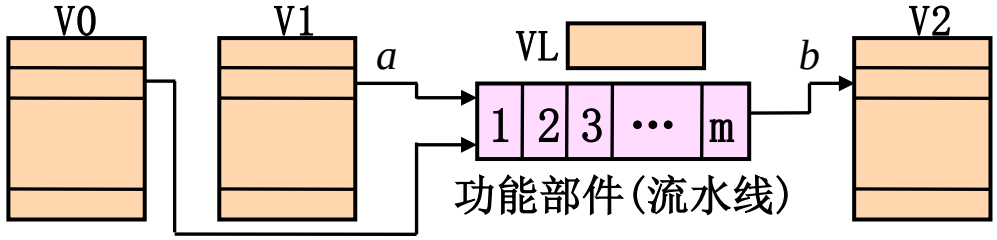
***M-M型向量机的向量指令：** 有M-M、M-S型，目的OPD可为标量

例：$M[B_D] \leftarrow M[B_{S1}]$ OP $M[B_{S2}]$

***R-R型向量机的向量指令：** 有V-V、V-S、V-M型，目的OPD同上

例：$V_D[B_D] \leftarrow V_{S1}[B_{S1}]$ OP $V_{S2}[B_{S2}]$

例1：向量加部件时延为6拍，VREG读/写时延为1拍(即图中a=1、b=1)，向量长度为64的指令V2←V0＋V1的操作时延是多少？



解：T＝(a+m+b)+(64-1)*1＝(1+6+1)+(64-1)*1＝71

思考：多条向量指令能并行执行吗？可以，无相关性就行

三、提高向量处理机性能的常用技术

*多个功能部件并行： 无冲突的指令可并行执行(同时)

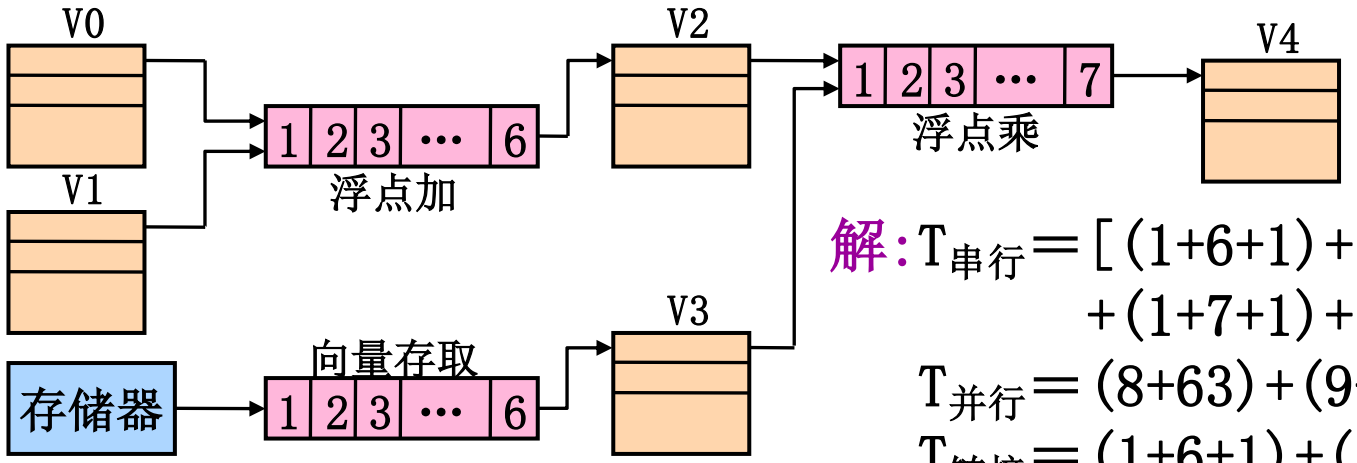
无冲突条件— 无功能部件及VREG使用冲突

VREG冲突类型— RAW、WAR、WAW、RAR(下标可能不同)

*链接技术： 有RAW冲突的指令可链接执行(重叠[早期为转发])

链接条件—指令间仅含RAW冲突，分量可用时可立即启动

例2：画出指令串①V3←M[R0] ②V2←V0+V1 ③V4←V3*V2执行的数据路径，计算串行、并行、链接的时延(n=64)



解：$T_{串行}=[(1+6+1)+(64-1)]*2$
$+(1+7+1)+(64-1)=214$

$T_{并行}=(8+63)+(9+63)=143$

$T_{链接}=(1+6+1)+(1+7+1)+63=80$

*向量屏蔽控制： 用VM(屏蔽REG)控制各个分量是否执行(条件执行)

```
for (i=0; i<64; i++)          SNESV V1,0        ;V1[i]=0时VM[i]=0、否则=1
  if (A[i]!=0)        ⇨       SUBVV V1,V1,V2    ;带屏蔽的V1←V1+V2
    A[i]=A[i]-B[i];           CVM               ;VM置全1(清除屏蔽位)
```

*使用MEM组： 满足向量存取部件的访存需求

访存需求— 多次/Tc(指令级并行)， 地址不连续(稀疏矩阵存取)

处理方案— 多个可独立访问的MEM， 交叉编址(提高带宽)

└←访问MEM的不同行

例3：某向量处理机的Tc=2ns，每个Tc最多有2次取向量、1次存向量操作，存储系统中RAM的T_M=14ns。希望向量存取不降低处理机性能，应如何配置RAM?

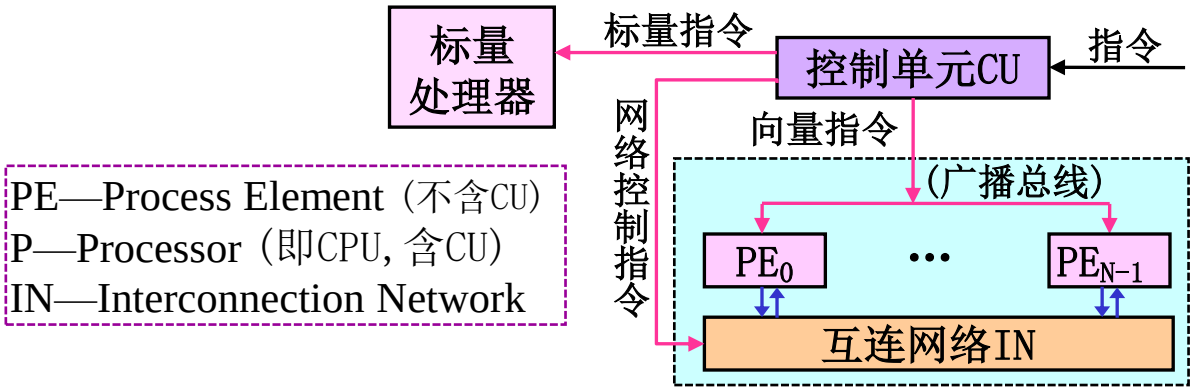
解：为满足访存操作的带宽需求，每组RAM个数＝14/2＝7 个；
为满足访存操作的并行需求，组数≥3 个。

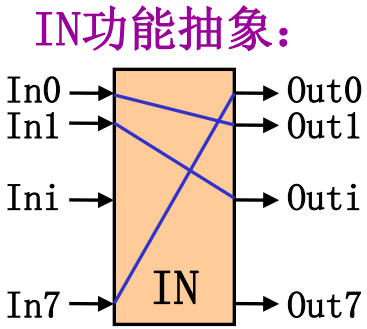
第2节　阵列处理机

※主要内容：操作模型，处理机结构，并行算法；多媒体SIMD技术

一、阵列处理机的操作模型　　（并行方式处理各分量）

*操作模型：1个CU管理多个PE并行操作(OP相同)，PE间通过IN通信





表示—阵列机＝{N, C, I, M, R}，　N—PE数，C—CU直接执行的指令集，I—PE执行的指令集，M—PE屏蔽方案集，R—IN寻径功能集

IN功能—所有的入端-出端同时互连，有多种映像(同时仅1种)

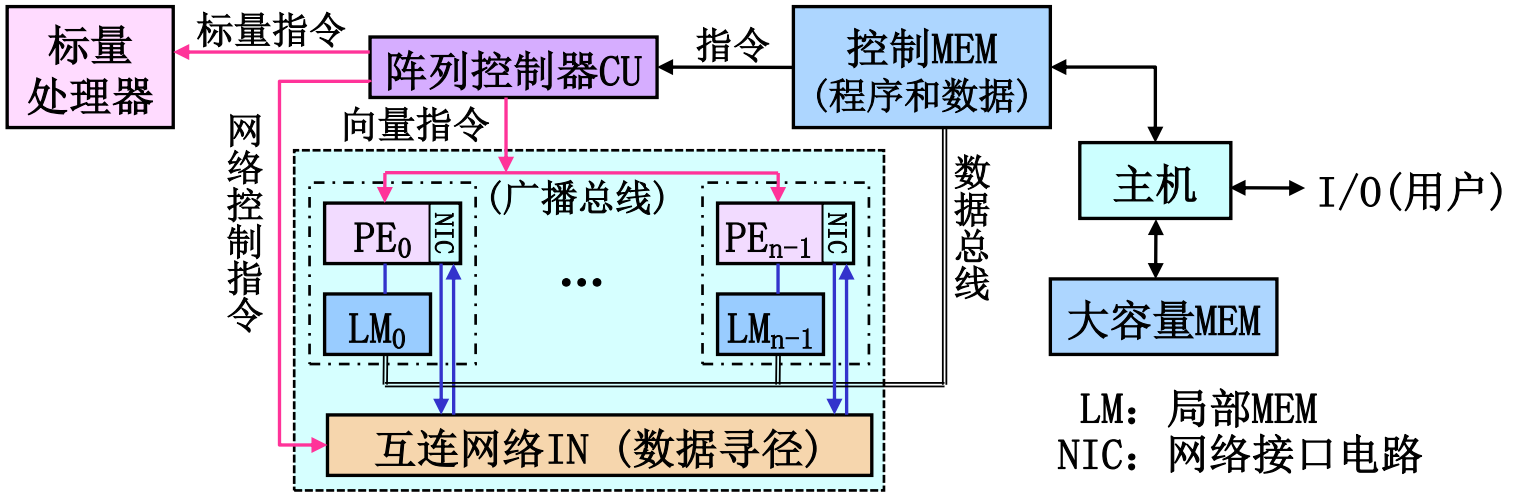
*特点：①开发并行性的同时性(又称并行计算机或SIMD计算机)

②∈专用机(IN功能→所支持并行算法)

思考：P7例1的阵列机时延？
T=1+6+1=8

二、阵列机基本结构

*分布式结构：每个PE自带私有MEM



特征—PE间通过IN实现通信，IN为单向网络(非对称)

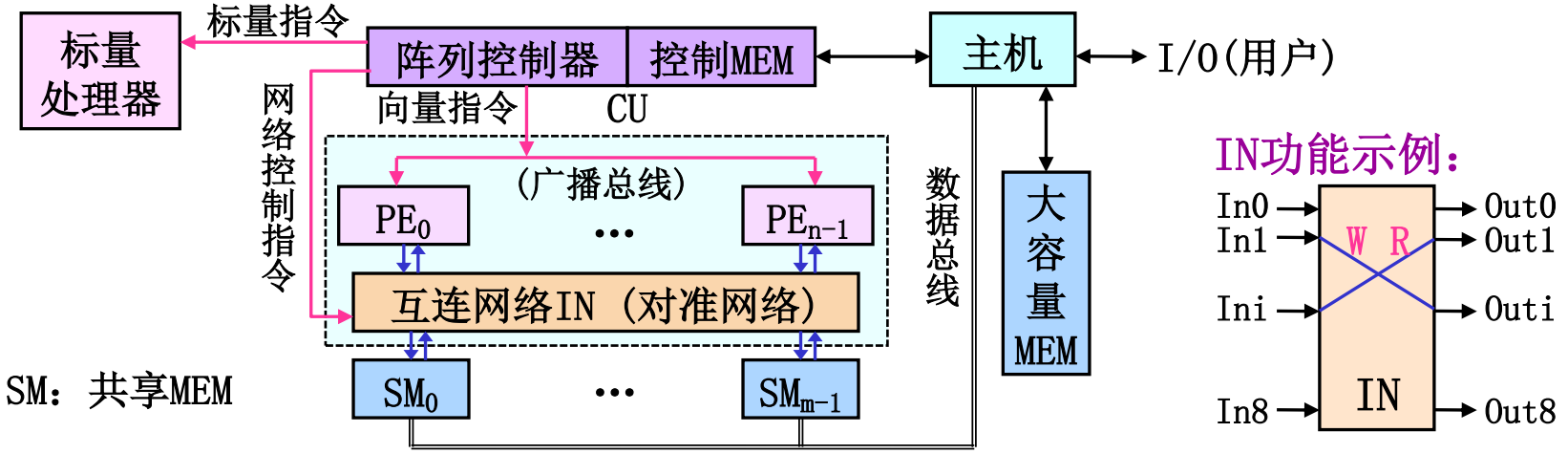
└←互连用地址实现　　└←如$PE_1 \rightarrow PE_2$、$PE_0 \rightarrow PE_1$

I/O操作—主机处理(LM-控制MEM)，LM编址方式有2种(公共/私有)

通信操作—用网络控制指令(基于目的地址)＋通信指令实现

```
例 I1: R[j]←LM[i]              ;各PE访问LM
   I2: LM[i]←R[j]+R[k]         ;各PE计算
   I3: Out[p+1]←In[p]          ;IN各端口互连
   I4: NIC←R[j],R[r]←NIC       ;各PE实现通信
```

*集中式结构：各PE共享m个MEM模块



特征—PE间通过SM实现通信，IN为双向网络(对称)

└←互连用地址实现→┘　　→┴←否则读、写不是同一个SM

I/O操作—主机处理(SM及控制MEM)，SM编址方式仅1种

通信操作—用网络控制指令(基于目的地址)＋访存指令实现

SM冲突避免—m与n互质，向量错位存放(行/列)　←同独立MEM

*应用选择：多为分布式结构(通信次数少→节点规模大)

*阵列机研究重点：并行算法对IN的适应性

四、阵列机常用并行算法

*性能影响因素： PE数(并行度)、 IN功能(通信性能[直接/转发])

*常用并行算法： (以Illiac IV阵列机为例)

有限差分— $U(x,y)=[U(x,y+h)+U(x,y-h)+U(x+h,y)+U(x-h,y)]/4$

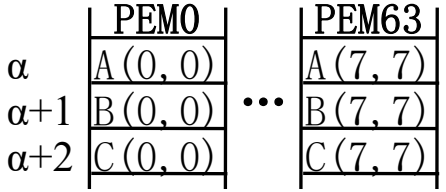
实现：每个PE计算一个坐标点，多次迭代(直到误差$\leqslant\lambda$)

指令串—(IN控制+通信)*4+加法*3+除法*1+误差&循环控制

矩阵加— $C(x,y)=A(x,y)+B(x,y)$

实现：每个PE计算一个元素，一次完成

指令串—①LDA α ②ADD α+1 ③STA α+2



累加求和— $S=A[0]+A[1]+\cdots+A[N-1]$

实现：各PE同时计算、调整步距&屏蔽值，$\log_2 N$次迭代

while (2^k ＜ N) //每轮步距为2^k，共$\log_2 N$轮

{ 置PE_0～PE_{2^k-1}不活跃； PE_{i+2^k} += PE_i ； k++; }

指令串—屏蔽指令+(IN控制+通信)+加法+循环控制

五、多媒体SIMD技术 --简化版阵列机

思考①：阵列机没有广泛使用的原因？ 成本增加大、性能提升小(使用频率低)

1、MMX(Multi Media eXtension)技术概述

Pentium的像素为8位/16位/32位

*多媒体数据的表示： 向量(分量类型有几种) →需扩展指令系统

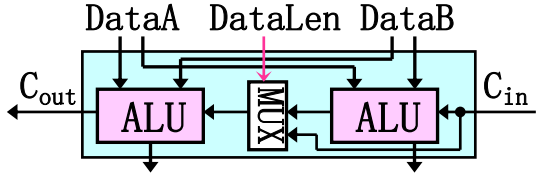
*MMX基本思想： ←需扩展系统结构

①向量用阵列机方式处理— 性能好(并行＞流水)

②不增加控制/状态REG— 实现软件兼容(含OS) ←目标是锦上添花

*MMX实现策略：

向量数据打包(长度固定)，同一部件并行处理各分量(多个片段)



*指令集的扩展： (以x86为例)

思考②：扩展IS需增加的内容？

增加4种64位数据表示、8个64位MMX寄存器、57条MMX指令

└←向量数据类型 (标量为32位)

└←向量寄存器组 (地址码长不变)

└←向量指令 (整数运算)

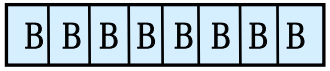
注：向量处理部件∉IS、∈CO(有多种方法)

思考：增加数据表示、指令功能、向量REG

2、MMX的数据表示及存放　（以x86为例）

***MMX数据的表示：** 4种64位无符号整数包　←通过操作码指明

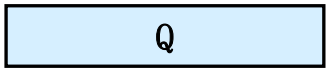
紧缩字节类型—8个字节(B)数据包



紧缩字类型—4个字(W)数据包

紧缩双字类型—2个双字(D)数据包

四字类型—1个四字(Q)数据



***MMX数据的REG存放：** 使用8个64位寄存器MM0-MM7，占全部位

例：$R_{MMX1} \leftarrow R_{MMX1}$ OP R_{MMX2}

Pentium—借用浮点寄存器FP，用别名方法实现

└←MMX指令将MMi映射到FPi

└→浮点运算与MMX运算互斥

思考：当时寄存器很便宜，为何不增设？　仅定点运算，初期降低成本

PIII起—增设MMX寄存器　←MMX支持向量浮点运算

***MMX数据的MEM存放：** 同标量数据(小端/对齐方式)

思考：执行MMX程序时（如在看电影），进行浮点运算的概率很小，初期开发尽量降低成本

3、MMX的指令集 （以x86为例）

*MMX指令功能：共7组，算术、逻辑、移位、比较、转换、传送、清除MMX状态(EMMS)

*MMX指令格式：利用指令前缀(值为0FH)表示，融入x86指令集

*MMX指令的特征：

向量指令— 部件可并行处理多个分量(数据短)

饱和运算— 溢出时不算作异常，结果为极限值(多媒体的需求)

如:结果＜255时为实际值，≥255时为255←┘

积和运算— 点积功能(Σa_i*b_i)，适于傅里叶变换等运算

比较指令— 比较结果(多个)放在MMX寄存器中(不增加标志位)，分支(多个)用带屏蔽的运算指令实现

转换指令— 数据精度转换(B/W/D/Q)，类型有紧缩、解紧缩，适于像素点间插值、色彩空间转换等运算

4、SIMD技术的发展

*发展历程：

即SIMD操作流水化(MMX为串行)

种类	时间	版本更新
MMX(Multi Media eXtension)	1996	
SSE(Streaming SIMD Extensions)	1999	SSE2(2001)，SSE3(2004)，SE4(2008)
AVX(Advanced Vector eXtensions)	2010	AVX2(2013)

*技术核心：增加分量个数，优化运算、存取、IN功能

	数据包	指令集特点
MMX	64位	定点运算(借用浮点REG)
SSE	128位	浮点运算(单→双、设置XMM)，流式数据存取(访存绕过Cache/预取)
AVX	256位	VEX编码(各种前缀统一编码)，3/4个OPD(增强功能/代码高效)， 数据重排(改变分量次序[IN功能])、不对齐访存(多种分量长度)
AVX2		256位定点运算，FMA指令(融合乘加)， 离散数据加载(按分量下标访问[含IN功能])

定点运算位128位

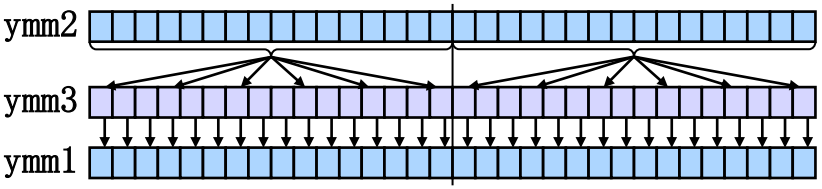
时间局部性差

AVX指令功能示例：

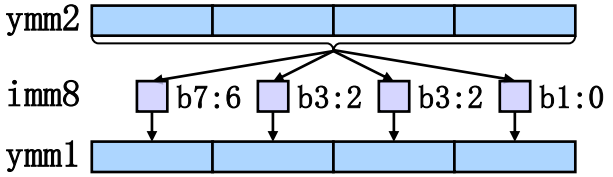
←源自群中资料

数据重排—按索引重新排列(IN功能)

←ymmi为寄存器

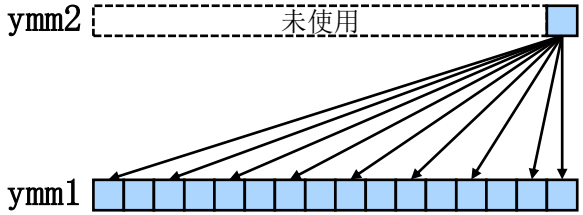


功能：ymm1[i]＝ymm2[ymm3[i]]

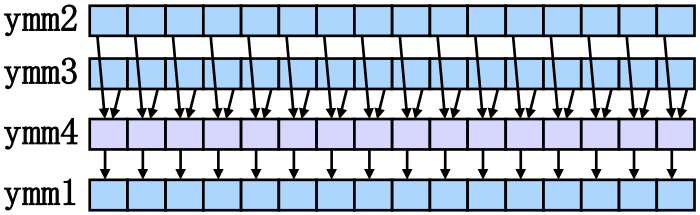


功能：ymm1[i]＝ymm2[imm8$_{2i+1\sim 2i}$]

数据广播/选择—复制最低分量，按mask最高位选择

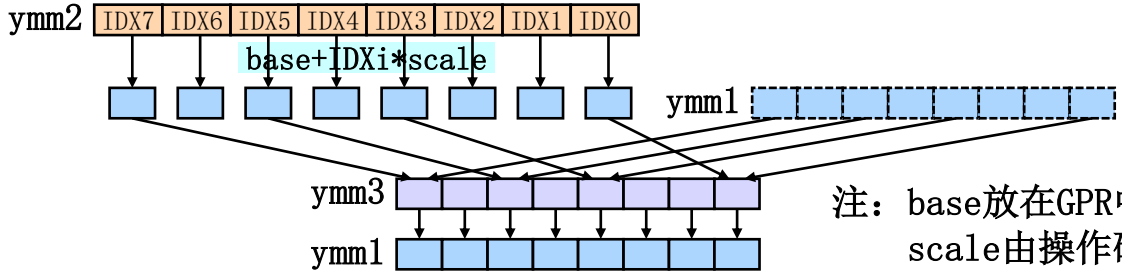


功能：ymm1[i]＝ymm2[0]



功能：ymm1[i]＝ymm4[i]$_7$? ymm2[i] : ymm3[i]

不对齐访存—按索引向量计算各分量地址(scale=1/2/4/8)



注：base放在GPR中，
scale由操作码指明

功能：ymm1[i]＝ymm3[i]? Mem[base+ymm2[i]*scale] : ymm1[i]

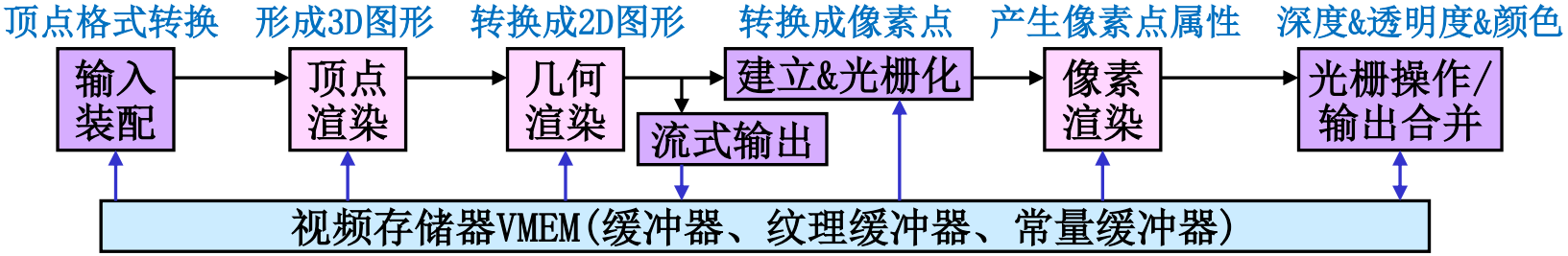
第3节　图像处理单元GPU

※主要内容：GPU应用概述，GPU基本结构，GPU编程模型

一、GPU应用概述

*GPU的产生：　(前身是VGA控制器)

功能—支持2D/3D图形处理的可编程处理器　←∈专用结构



注：顶点/几何/像素渲染(又称着色)为可编程流水段

编程—基于openGL、DirectX模型　←开放标准、微软API

*GPU的发展：　共用PE阵列

功能—支持图形处理/可视化计算/GPU计算的并行处理器　←∈通用结构

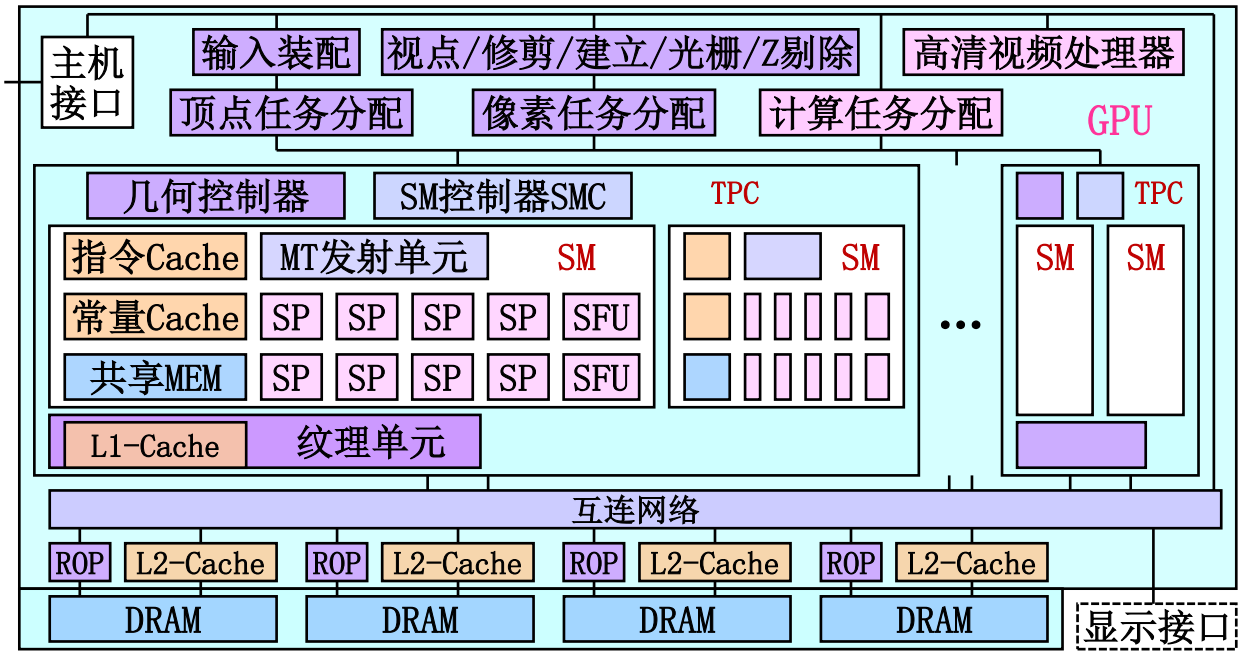
└←视频　└←通用并行

编程—基于CUDA(Compute Unified Device Architecture)　←编程模型和软件平台

└←编程时为单线程程序，执行时实例化为多个并发线程

二、GPU基本结构

*基本结构： 图形流水线(无渲染段)＋处理器阵列(渲染段/GPU计算)＋显存



TPC—纹理/处理器簇
(Texture Process Cluster)
SM—流多处理器
(Streaming Multiprocessor)
SP—流处理器
(Streaming Processor)
SFU—特殊功能单元
(Special Function Unit)
ROP—光栅操作处理器
(Raster Operation Processor)

SIMT—Single Instruction Multiple Thread

SM—多线程的流式多处理器，运行同一程序(≈1个CPU核)

多个SP→SIMT(代码相同)→SIMD(数据不同)

SP支持多线程(代码相同/数据不同)、流式处理(部件流水)

SP—多线程的流式处理器，无指令单元，REG较多，部件流水化

(处理标量) (SM统一控制) (如1024个) (利于数据流)

VMEM—GPU的MEM，层次结构(Cache+DRAM)

*SIMT的实现思路：　（效果＝SIMD）

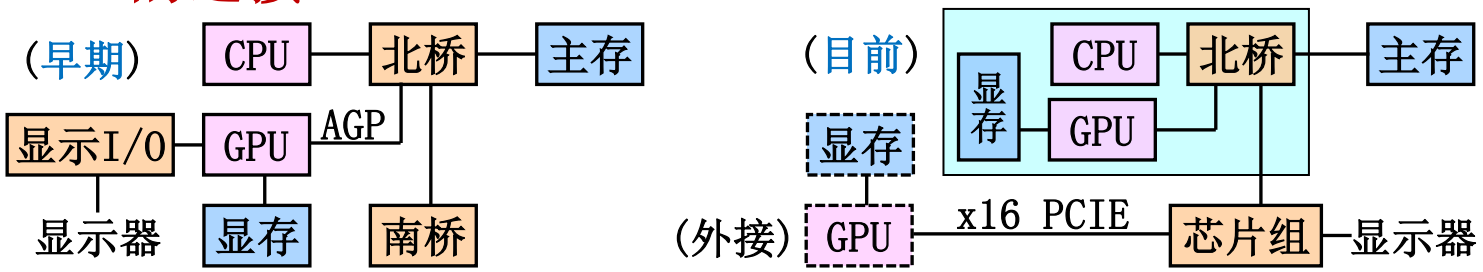
SM负责取指&译码，创建、调度、执行(发射)多个并发线程；

（1条）　为减少次数→┗━━━━━┛→每个SP≥1个线程

各SP负责执行所分配的多个线程(指令相同/数据不同)

（同时）　└←流水方式←┘

*GPU-CPU的连接：

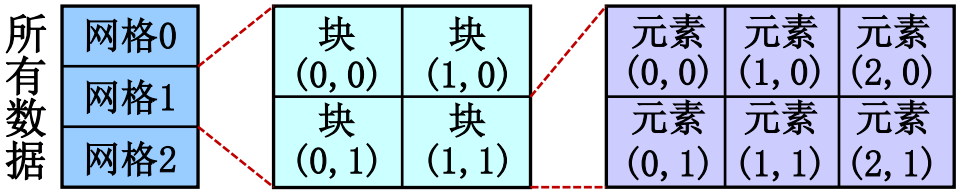


主机接口—CPU内部总线，或多通道PCI-Express总线

MEM访问—均可访问主存&显存(显存空间∈主存空间)

三、CUDA编程模型

*数据并行问题分解： 3个层次



*软件结构：

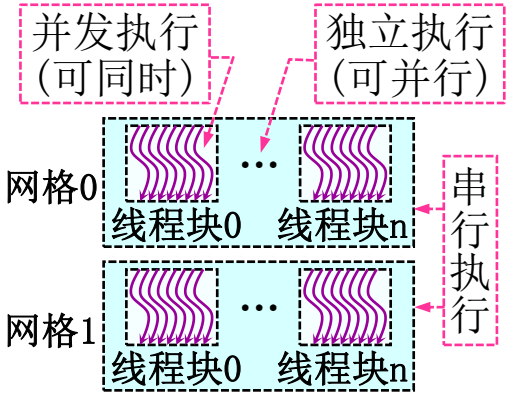
编程{ 内　核—适合单个线程的程序/函数(SI)，可被多个线程执行(MT)

执行{ 线　程—处理1个元素的程序执行实例(SP控制[指令串行])

执行{ 线程块—执行相同线程程序、可同步&通信的1组并发线程(SM控制)

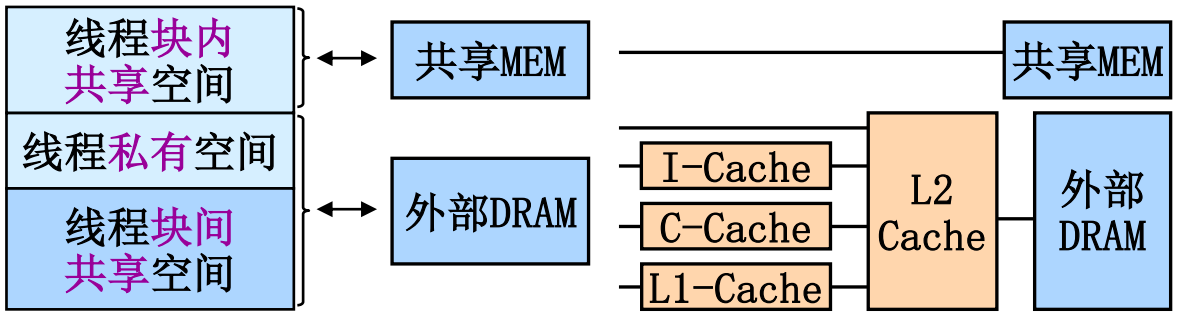
执行{ 网　格—执行相同内核程序、可独立执行的1组线程块(TPC控制)

(多个SM时)可并行→



*同步方式： 栅障同步 ←块内线程间、线程块间

*通信方式： 共享MEM，MEM可为分布式结构



线程可访问空间　对应的物理MEM　物理MEM的结构

*SIMT实现机制：

SM调度机制—基于Warp调度，优先流出Warp池中就已绪Warp

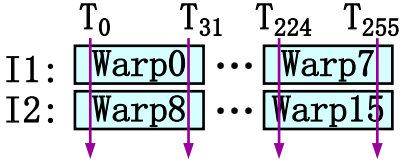
类似于保留站→┴←译码时放入、执行完移出

类似于超标量发射包→┴←为同一线程块中连续32个线程的同一条指令

示例：若SM有8个SP、Warp池大小＝16，（硬件参数）

线程块大小＝256，（软件参数）

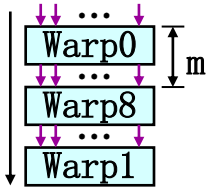
则启动内核时创建256个线程，



指令译码时产生256/32＝8个Warp、放入Warp池；

每次调度时发射1个Warp，调度间隔m≥4Tc

负责32/8＝4条指令/SP→┘　　可能有冒险→┘



SP执行机制—流水执行Warp中所负责的指令　←所有SP≈SIMD

并行效果—相当于阵列机(各SP同时执行，并行度≥SM中SP个数)，

└←可同时分配多个SM

可支持MIMD(各SM独立执行，内核程序可不同)

第六章小结&思考

(1)相对于标量指令格式，向量指令格式需增加哪些字段？
(2)向量处理机中，向量REG与标量REG的结构有何不同？向量功能部件与相同功能标量功能部件的结构有何不同？
(3)阵列处理机中，PE的功能部件是标量部件还是向量部件？
(4)同一条向量指令，在向量处理机、阵列处理机中执行时，指令周期相差多少个时钟周期？
(5)相对于向量处理机，阵列处理机的指令集还需包含哪些指令？
(6)集中式阵列处理机对互连网络IN的功能要求是什么？分布式阵列处理机有此要求吗？
(7)并行算法在不同阵列处理机中实现时，性能受哪些因素影响？
(8)多媒体SIMD技术如何保持软件兼容性的？实现方法？

(1)每个向量OPD的分量基地址，向量长度(可共用)
(2)标量REG的数组＋读地址REG及写地址REG，标量功能部件的流水线
(3)标量部件　　(4)向量长度-1
(5)网络控制指令、通信指令　　(6)双向互连(PE-MEM间)，无要求(PE-PE间)
(7)IN功能(决定通信效率)
(8)只增加指令，不增加控制/状态REG；向量总位数固定、同一个部件分段处理

第六章课后复习思考题

(1)阵列机计算机PE中的功能部件，可以用于向量处理机吗？为什么？

(2)早期的向量机中，向量指令只能在第一个结果分量写入VREG时链接，是如何实现的？现在的向量机中，只要VREG的分量可用，即可启动下条指令，VREG内部要做哪些改进即可实现？

(3)设计一个采用加、乘和通信(数据寻径)操作，实现$S=A_1*B_1+\cdots+A_{32}*B_{32}$的算法,要求在下列2种计算机中所用时间最短，并写出所用时间。假设加法、乘法时延分别为2Tc、4Tc，忽略取指令、取操作数、译码的时延，所有的指令和数据均已装入相关的PE。

a)串行计算机，PE中有一个加法器和乘法器，加法和乘法需分时实现。该计算机中，无需通信操作。

b)有8个PE(PE_0～PE_7)的阵列计算机，PE采用双向环结构互连。每个PE与相邻PE通信的时延为1Tc。操作数A_i和B_i最初存放在$PE_{i \bmod 8}$中，PE中的加法和乘法需分时实现。

(4)为了支持MMX，x86的ISA做了哪些扩展？

(5)GPU实现数据并行的机理是什么？

(6)CUDA的线程间通信，什么情况下涉及局部MEM？什么情况下涉及GPU MEM？