1.3 计算机系统设计者的主要任务

计算机系统结构的设计者必须设计能够满足包括价格、功耗、性能和可用性指标的计算机。

大致需要3个方面的工作

① 确定用户对计算机系统的功能、价格和性能的要求

- 由市场需求激发的特定功能
- 应用软件通常能帮助我们筛选出特定的功能要求

设计者需要考虑的一些主要因素

Functional requirements	Typical features required or supported
Application area	*Target of computer*
Personal mobile device	Real-time performance for a range of tasks, including interactive performance for graphics, video, and audio; energy efficiency (Ch. 2, 3, 4, 5; App. A)
General-purpose desktop	Balanced performance for a range of tasks, including interactive performance for graphics, video, and audio (Ch. 2, 3, 4, 5; App. A)
Servers	Support for databases and transaction processing; enhancements for reliability and availability; support for scalability (Ch. 2, 5; App. A, D, F)
Clusters/warehouse-scale computers	Throughput performance for many independent tasks; error correction for memory; energy proportionality (Ch 2, 6; App. F)
Embedded computing	Often requires special support for graphics or video (or other application-specific extension); power limitations and power control may be required; real-time constraints (Ch. 2, 3, 5; App. A, E)
Level of software compatibility	*Determines amount of existing software for computer*
At programming language	Most flexible for designer; need new compiler (Ch. 3, 5; App. A)
Object code or binary compatible	Instruction set architecture is completely defined—little flexibility—but no investment needed in software or porting programs (App. A)
Operating system requirements	*Necessary features to support chosen OS (Ch. 2; App. B)*
Size of address space	Very important feature (Ch. 2); may limit applications
Memory management	Required for modern OS; may be paged or segmented (Ch. 2)
Protection	Different OS and application needs: page vs. segment; virtual machines (Ch. 2)
Standards	*Certain standards may be required by marketplace*
Floating point	Format and arithmetic: IEEE 754 standard (App. J), special arithmetic for graphics or signal processing
I/O interfaces	For I/O devices: Serial ATA, Serial Attached SCSI, PCI Express (App. D, F)
Operating systems	UNIX, Windows, Linux, CISCO IOS
Networks	Support required for different networks: Ethernet, Infiniband (App. F)
Programming languages	Languages (ANSI C, C++, Java, Fortran) affect instruction set (App. A)

- **应用领域**

  专用还是通用？面向科学计算还是面向商用处理？

- ✓ 个人移动设备
- ✓ 桌面机Desktop
- ✓ 服务器Server
- ✓ 集群/超大规模集群（数据中心）
- ✓ 嵌入式

Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Embedded
Price of system	$100–$1000	$300–$2500	$5000–$10,000,000	$100,000–$200,000,000	$10–$100,000
Price of micro-processor	$10–$100	$50–$500	$200–$2000	$50–$250	$0.01–$100
Critical system design issues	Cost, energy, media performance, responsiveness	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, energy, application-specific performance

设计者需要考虑的一些主要因素

Functional requirements	Typical features required or supported
Application area	*Target of computer*
Personal mobile device	Real-time performance for a range of tasks, including interactive performance for graphics, video, and audio; energy efficiency (Ch. 2, 3, 4, 5; App. A)
General-purpose desktop	Balanced performance for a range of tasks, including interactive performance for graphics, video, and audio (Ch. 2, 3, 4, 5; App. A)
Servers	Support for databases and transaction processing; enhancements for reliability and availability; support for scalability (Ch. 2, 5; App. A, D, F)
Clusters/warehouse-scale computers	Throughput performance for many independent tasks; error correction for memory; energy proportionality (Ch 2, 6; App. F)
Embedded computing	Often requires special support for graphics or video (or other application-specific extension); power limitations and power control may be required; real-time constraints (Ch. 2, 3, 5; App. A, E)
Level of software compatibility	*Determines amount of existing software for computer*
At programming language	Most flexible for designer; need new compiler (Ch. 3, 5; App. A)
Object code or binary compatible	Instruction set architecture is completely defined—little flexibility—but no investment needed in software or porting programs (App. A)
Operating system requirements	*Necessary features to support chosen OS (Ch. 2; App. B)*
Size of address space	Very important feature (Ch. 2); may limit applications
Memory management	Required for modern OS; may be paged or segmented (Ch. 2)
Protection	Different OS and application needs: page vs. segment; virtual machines (Ch. 2)
Standards	*Certain standards may be required by marketplace*
Floating point	Format and arithmetic: IEEE 754 standard (App. J), special arithmetic for graphics or signal processing
I/O interfaces	For I/O devices: Serial ATA, Serial Attached SCSI, PCI Express (App. D, F)
Operating systems	UNIX, Windows, Linux, CISCO IOS
Networks	Support required for different networks: Ethernet, Infiniband (App. F)
Programming languages	Languages (ANSI C, C++, Java, Fortran) affect instruction set (App. A)

- **软件兼容性的选择**

  软件兼容是指一台计算机上的程序不加修改就可以移植另一台计算机上正常运行。

- ✓ 高级语言层面的兼容：设计更加灵活，需要设计新编译器
- ✓ 目标代码或二进制层面的兼容：限制比较严格
  - ✓ 指令集兼容
  - ✓ 软件可移植性

- **软件可移植性的实现**

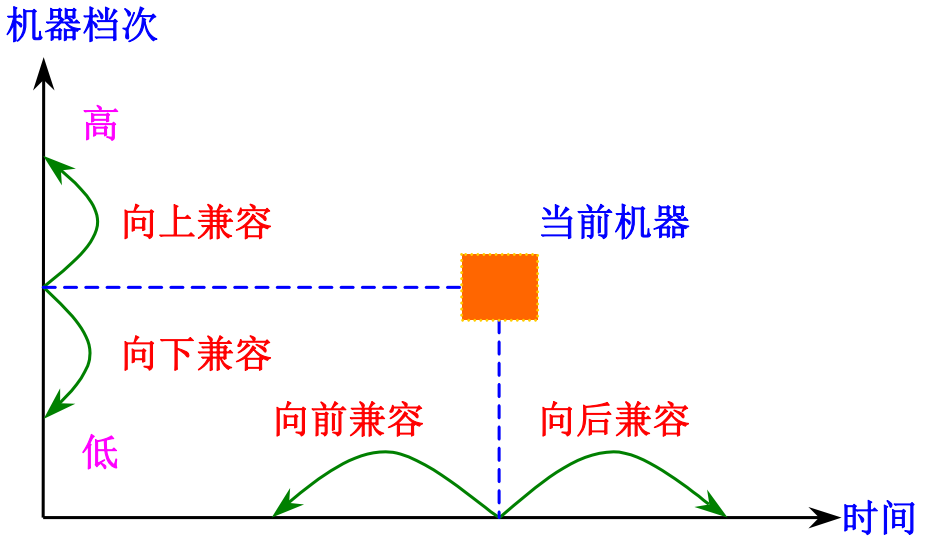
1. 统一高级语言
2. 系列机
3. 模拟和仿真

1. 统一高级语言
   - 实现软件移植的一种理想的方法
   - 较难实现
2. 系列机

   由同一厂家生产的具有相同的系统结构(ISA)，但具有不同组成和实现的一系列不同型号的机器

   - 较好地解决软件开发要求系统结构相对稳定与器件、硬件技术迅速发展的矛盾。

- 软件兼容
  - 向上（下）兼容：按某档机器编制的程序，不加修改就能运行于比它高（低）档的机器。
  - 向前（后）兼容：按某个时期投入市场的某种型号机器编制的程序，不加修改地就能运行于在它之前（后）投入市场的机器。
  - 向后兼容是系列机的根本特征。
- 兼容机：由不同公司厂家生产的具有相同系统结构的计算机 。



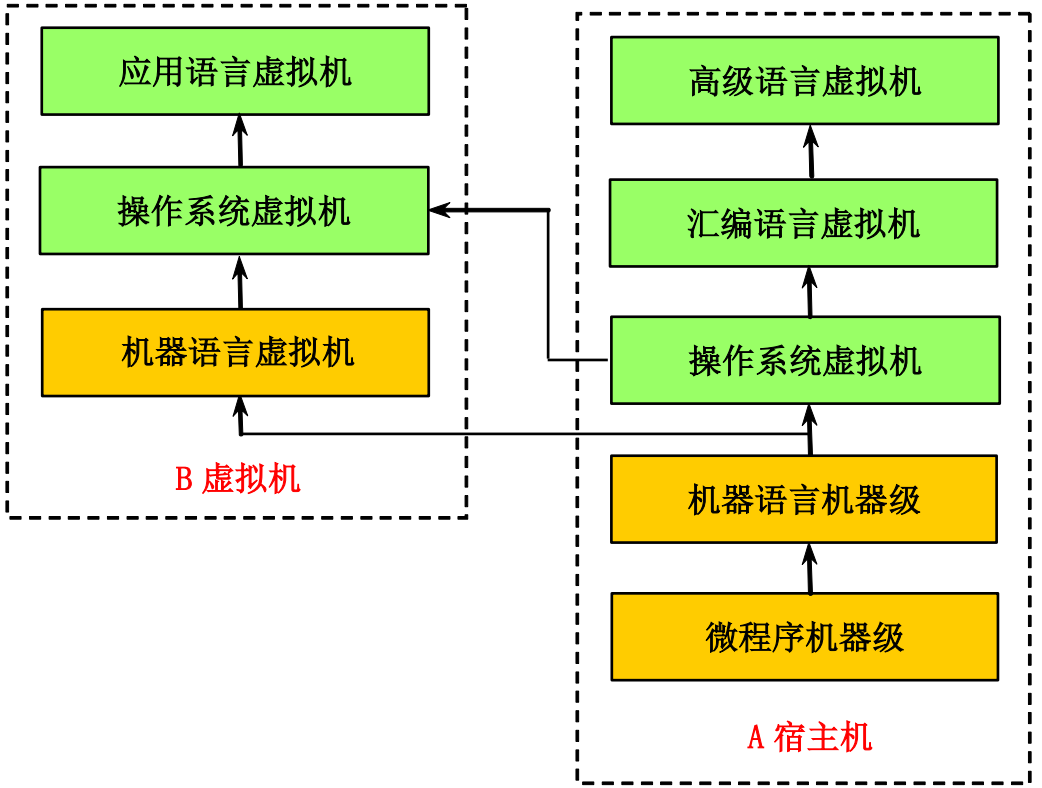
具有代表性的计算机（IBM360）

- 1961年开始研制，1964年获得成功，电脑系统用360为名， 代表着360电脑从工商业到科学界的全方位应用
- 系列产品，大中小型360计算机系统，都能处理相同的指令，使用相同的软件，配置相同的外部设备，而且能够相互连接在一起工作。IBM360共有6个型号的大、中、小型系统和44种新式的配套设备。

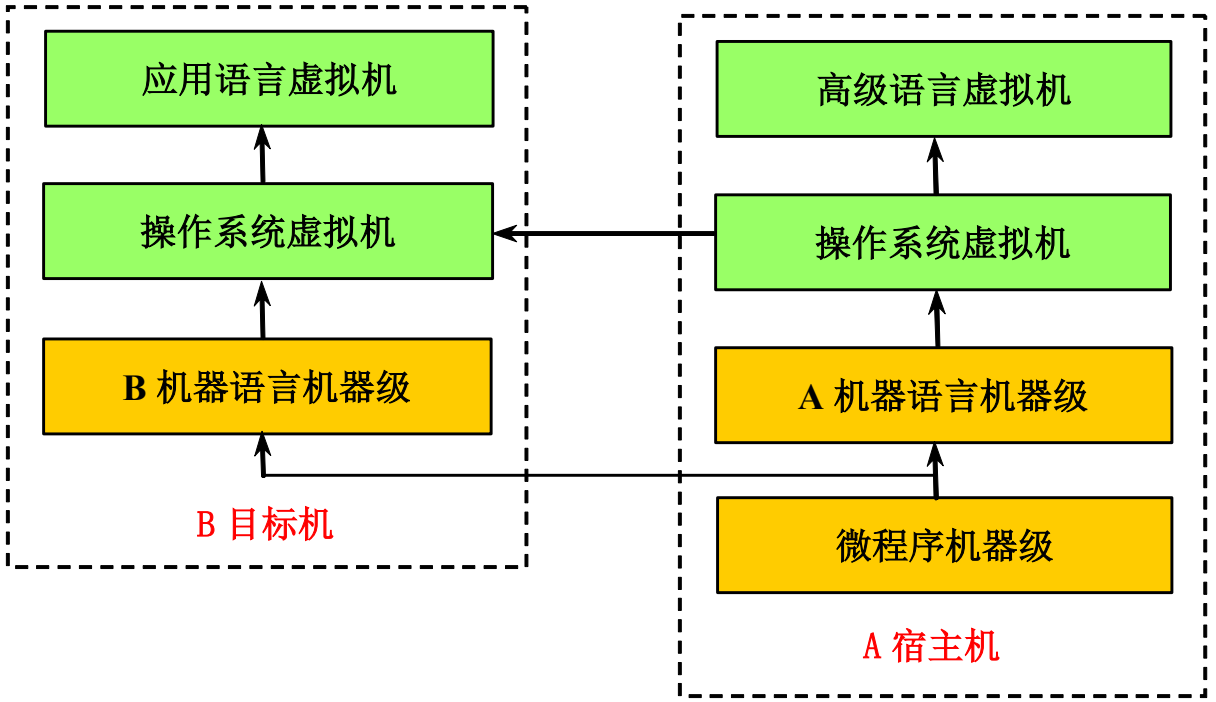
3. 模拟和仿真

- 使软件能在具有不同系统结构的机器之间相互移植。
  - 在一种系统结构上实现另一种系统结构。
  - 从指令集的角度来看，就是要在一种机器上实现另一种机器的指令集。

- **模拟**：用软件的方法在一台现有的机器（称为宿主机）上实现另一台机器（称为虚拟机）的指令集。
  - 通常用解释的方法来实现。
  - 运行速度较慢，性能较差。



- 仿真：用一台现有机器（宿主机）上的微程序去解释实现另一台机器（目标机）的指令集。
  - 运行速度比模拟方法的快
  - 仿真只能在系统结构差距不大的机器之间使用



设计者需要考虑的一些主要因素

Functional requirements	Typical features required or supported
Application area	*Target of computer*
Personal mobile device	Real-time performance for a range of tasks, including interactive performance for graphics, video, and audio; energy efficiency (Ch. 2, 3, 4, 5; App. A)
General-purpose desktop	Balanced performance for a range of tasks, including interactive performance for graphics, video, and audio (Ch. 2, 3, 4, 5; App. A)
Servers	Support for databases and transaction processing; enhancements for reliability and availability; support for scalability (Ch. 2, 5; App. A, D, F)
Clusters/warehouse-scale computers	Throughput performance for many independent tasks; error correction for memory; energy proportionality (Ch 2, 6; App. F)
Embedded computing	Often requires special support for graphics or video (or other application-specific extension); power limitations and power control may be required; real-time constraints (Ch. 2, 3, 5; App. A, E)
Level of software compatibility	*Determines amount of existing software for computer*
At programming language	Most flexible for designer; need new compiler (Ch. 3, 5; App. A)
Object code or binary compatible	Instruction set architecture is completely defined—little flexibility—but no investment needed in software or porting programs (App. A)
Operating system requirements	*Necessary features to support chosen OS (Ch. 2; App. B)*
Size of address space	Very important feature (Ch. 2); may limit applications
Memory management	Required for modern OS; may be paged or segmented (Ch. 2)
Protection	Different OS and application needs: page vs. segment; virtual machines (Ch. 2)
Standards	*Certain standards may be required by marketplace*
Floating point	Format and arithmetic: IEEE 754 standard (App. J), special arithmetic for graphics or signal processing
I/O interfaces	For I/O devices: Serial ATA, Serial Attached SCSI, PCI Express (App. D, F)
Operating systems	UNIX, Windows, Linux, CISCO IOS
Networks	Support required for different networks: Ethernet, Infiniband (App. F)
Programming languages	Languages (ANSI C, C++, Java, Fortran) affect instruction set (App. A)

- 操作系统需求

  包括地址空间大小、存储管理、保护等。从系统结构上对操作系统的需求提供支持，是很重要的一点。

- 标准

  确定系统中哪些方面要采用标准以及采用什么标准。如：浮点数标准、I/O总线标准、网络标准、程序设计语言标准等。

1.3 计算机系统设计者的主要任务

② 软硬件功能分配

- 考虑如何优化设计？

  必须考虑软硬件功能的合理分配。

- 软件和硬件在实现功能上是等价的
  - 用软件实现的优点：设计容易、修改简单，而且可以减少硬件成本。但是所实现的功能的速度较慢。
  - 用硬件实现的优点：速度快、性能高，但它修改困难，灵活性差。
- 在软硬件之间进行折中和取舍。

③ 设计出生命周期长的系统结构

- 特别注意计算机应用和计算机技术的发展趋势
- 设计出具有一定前瞻性的系统结构，以使得它具有较长的生命周期。

1.7 计算机系统的定量设计方法

1.7.1 计算机系统设计的定量原理

4个定量原理：

1. 以经常性事件为重点

- 对经常发生的情况采用优化方法的原则进行选择，以得到更多的总体上的改进。
- 优化是指分配更多的资源、达到更高的性能或者分配更多的电能等。

2. Amdahl定律

加快某部件执行速度所能获得的系统性能加速比，受限于该部件的执行时间占系统中总执行时间的百分比。

系统性能加速比：

$$\text{加速比} = \frac{\text{系统性能}_{\text{改进后}}}{\text{系统性能}_{\text{改进前}}} = \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}}$$

- 加速比依赖于两个因素
  - 可改进比例（F_e）：在改进前的系统中，可改进部分的执行时间在总的执行时间中所占的比例。

    它总是小于等于1。

    例如：一个需运行60秒的程序中有20秒的运算可以加速，那么这个比例就是20/60。
  - 部件加速比（S_e）：可改进部分改进以后性能提高的倍数。它是改进前所需的执行时间与改进后执行时间的比。

    一般情况下部件加速比是大于1的。

    例如：若系统改进后，可改进部分的执行时间是2秒，而改进前其执行时间为5秒，则部件加速比为5/2。

- 改进后程序的总执行时间T_n

$$T_n = T_0\left(1 - Fe + \frac{Fe}{Se}\right)$$

  - T_0：改进前整个程序的执行时间
  - $1-F_e$：不可改进比例

系统加速比S_n为改进前与改进后总执行时间之比：

$$S_n = \frac{T_0}{T_n} = \frac{1}{(1 - Fe) + \frac{Fe}{Se}}$$

例1.1 我们分析一个用于网页服务器系统的处理器的性能。假定采用以下的增强方式，新的CPU在计算性能上的运行速度是原来处理器中的15倍，同时假定此CPU有40％的时间用于计算，另外60％的时间用于I/O操作。那么增强性能后总的加速比是多少？

解 由题可知： $F_e = 40\% = 0.4$

$S_e = 15$

根据Amdahl定律可知：

$$S_n = \frac{1}{(1-Fe)+\frac{Fe}{Se}} = \frac{1}{(1-0.4)+\frac{0.4}{15}} \approx 1.6$$

采用此增强功能方法后，能使整个系统的性能提高到原来的1.6倍。

例1.2　求浮点数（FP）平方根的不同实现方法在性能上可能有很大差异。假定求浮点数平方根（FPSQR）的操作在某台机器上的一个标准测试程序中占总执行时间的20%。一种方法是增加专门的FPSQR硬件，可以将FPSQR的操作速度提高为原来的10倍。另一种方法是提高所有的FP运算指令的执行速度；FP运算指令在总执行时间中占50%。设计小组认为可以把所有的FP指令的执行速度提高为原来的1.6倍从而达到提高求浮点数平方根操作的速度。试比较这两种方法。

解：

$$SPEEDUP_{FPSQR}=\frac{1}{(1-0.2)+\frac{0.2}{10}}=\frac{1}{0.82}=1.22$$

$$SPEEDUP_{FP}=\frac{1}{(1-0.5)+\frac{0.5}{1.6}}=\frac{1}{0.8125}=1.23$$

- Amdahl定律：一种性能改进的递减规则
  - 如果仅仅对计算任务中的一部分做性能改进，则改进得越多，所得到的总体性能的提升就越有限。
- 重要推论：如果只针对整个任务的一部分进行改进和优化，那么所获得的加速比不超过：

1/（1－可改进比例）

$$\frac{1}{1-F_e}$$

3. CPU性能公式

- 执行一个程序所需的CPU时间

CPU时间 = 执行程序所需的时钟周期数×时钟周期时间

其中：时钟周期时间是系统时钟频率的倒数。

- 每条指令执行的平均时钟周期数CPI (Cycles Per Instruction)

CPI = 执行程序所需的时钟周期数 / IC

IC：所执行的指令条数

- 程序执行的CPU时间可以写成

CPU时间 = IC ×CPI ×时钟周期时间

  - 时钟周期时间：取决于硬件实现技术和计算机组成；
  - CPI：取决于计算机组成和指令系统的结构；
  - IC：取决于指令系统的结构和编译技术

- 对CPU性能公式进行进一步细化

假设：计算机系统有n种指令；

CPI_i ：第i种指令的处理时间；

IC_i ：在程序中第i种指令出现的次数；

则：

$$\text{CPU时钟周期数} = \sum_{i=1}^{n} (CPI_i \times IC_i)$$

CPU时间 = 执行程序所需的时钟周期数×时钟周期时间

$$= \sum_{i=1}^{n} (CPI_i \times IC_i) \times \text{时钟周期时间}$$

CPI可以表示为：

$$CPI = \frac{\text{时钟周期数}}{IC} = \frac{\sum_{i=1}^{n} (CPI_i \times IC_i)}{IC} = \sum_{i=1}^{n} (CPI_i \times \frac{IC_i}{IC})$$

其中：(IC_i/IC)反映了第i种指令在程序中所占的比例。

例1.3 假设FP指令的比例为25%，其中，FPSQR占全部指令的比例为2%，FP操作的CPI为4，FPSQR操作的CPI为20，其他指令的平均CPI为1.33。现有两种改进方案，第一种是把FPSQR操作的CPI减至2，第二种是把所有的FP操作的CPI减至2.5，试比较两种方案对系统性能的提高程度。

解 没有改进之前，每条指令的平均时钟周期CPI为：

$$CPI = \sum_{i=1}^{n}\left(CPI_i \times \frac{IC_i}{IC}\right) = (4 \times 25\%) + (1.33 \times 75\%) = 2$$

（1）采用第一种方案

FPSQR操作的CPI由CPI_{FPSQR}＝20减至CPI'_{FPSQR}＝2，则整个系统的指令平均时钟周期数为：

$$CPI_1 = CPI - (CPI_{FPSQR} - CPI'_{FPSQR}) \times 2\%$$

$$= 2 - (20 - 2) \times 2\% = 1.64$$

（2）采用第二种方案

所有FP操作的CPI由CPI_{FP}＝4减至CPI'_{FP}＝2.5，则整个系统的指令平均时钟周期数为：

$$CPI_2 = CPI - (CPI_{FP} - CPI'_{FP}) \times 25\%$$

$$= 2 - (4 - 2.5) \times 25\% = 1.625$$

从降低整个系统的指令平均时钟周期数的程度来看，第二种方案优于第一种方案。

例1.4 考虑条件分支指令的两种不同设计方法：

（1）CPU_1：通过比较指令设置条件码，然后测试条件码进行分支。

（2）CPU_2：在分支指令中包括比较过程。

在这两种CPU中，条件分支指令都占用2个时钟周期，而所有其它指令占用1个时钟周期。对于CPU_1，执行的指令中分支指令占30%；由于每条分支指令之前都需要有比较指令，因此比较指令也占30%。由于CPU_1在分支时不需要比较，因此假设CPU_2的时钟周期时间是CPU_1的1.15倍。问：哪一个CPU更快？

解 用CPU性能公式。占用2个时钟周期的分支指令占总指令的30%，剩下的指令占用1个时钟周期。所以

$$CPI_1 = 0.3 \times 2 + 0.70 \times 1 = 1.3$$

则CPU_1性能为：

$$总CPU时间_1 = IC_1 \times 1.3 \times 时钟周期_1$$

根据假设，有：

$$时钟周期_2 = 1.15 \times 时钟周期_1$$

在CPU_2中没有独立的比较指令，所以CPU_2的程序量为CPU_1的70%，分支指令的比例为：

$$30\%/70\% = 42.8\%$$

这些分支指令占用2个时钟周期，而剩下的57.2%的指令占用1个时钟周期，

因此：

$$CPI_2 = 0.428 \times 2 + 0.572 \times 1 = 1.428$$

因为CPU_2不执行比较，故：

$$IC_2 = 0.7 \times IC_1$$

因此CPU_2性能为：

$$\text{总CPU时间}_2 = IC_2 \times CPI_2 \times \text{时钟周期}_2$$

$$= 0.7 \times IC_1 \times 1.428 \times (1.15 \times \text{时钟周期}_1)$$

$$= 1.15 \times IC_1 \times \text{时钟周期}_1$$

当CPU1比CPU2快30%以上，CPU1将会执行的更快。

4. 程序的局部性原理

程序执行时所访问的存储器地址分布不是随机的，而是相对地簇聚。

- 常用的一个经验规则

  程序执行时间的90%都是在执行程序中10%的代码。

- 程序的时间局部性

  程序即将用到的信息很可能就是目前正在使用的信息。

- 程序的空间局部性

  程序即将用到的信息很可能与目前正在使用的信息在空间上相邻或者临近。

1.5 计算机系统结构中并行性的发展

1.5.1 并行性的概念

1. 并行性：计算机系统在同一时刻或者同一时间间隔内进行多种运算或操作。

只要在时间上相互重叠，就存在并行性。

- 同时性：两个或两个以上的事件在同一时刻发生。
- 并发性：两个或两个以上的事件在同一时间间隔内发生。

2. 从处理数据的角度来看，并行性等级从低到高可分为：

- 字串位串：每次只对一个字的一位进行处理。

  最基本的串行处理方式，不存在并行性。

- 字串位并：同时对一个字的全部位进行处理，不同字之间是串行的。

  开始出现并行性。

- 字并位串：同时对许多字的同一位（称为位片）进行处理。

  具有较高的并行性。

- 全并行：同时对许多字的全部位或部分位进行处理。

  最高一级的并行。

3. 从执行程序的角度来看，并行性等级从低到高可分为：

- 指令内部并行：单条指令中各微操作之间的并行。
- 指令级并行：并行执行两条或两条以上的指令。
- 线程级并行：并行执行两个或两个以上的线程。

  通常是以一个进程内派生的多个线程为调度单位。
- 任务级或过程级并行：并行执行两个或两个以上的过程或任务（程序段）

  以子程序或进程为调度单元。
- 作业或程序级并行：并行执行两个或两个以上的作业或程序。

1.5.2 提高并行性的技术途径

三种途径：

1. 时间重叠

   引入时间因素，让多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分，以加快硬件周转而赢得速度。

2. 资源重复

   引入空间因素，以数量取胜。通过重复设置硬件资源，大幅度地提高计算机系统的性能。

3. 资源共享

   这是一种软件方法，它使多个任务按一定时间顺序轮流使用同一套硬件设备。

1.5.3 单机系统中并行性的发展

1. 在发展高性能单处理机过程中，起主导作用的是时间重叠原理。

实现时间重叠的基础：部件功能专用化

- 把一件工作按功能分割为若干相互联系的部分；
- 把每一部分指定给专门的部件完成；
- 然后按时间重叠原理把各部分的执行过程在时间上重叠起来，使所有部件依次分工完成一组同样的工作。

2. 在单处理机中，资源重复原理的运用也已经十分普遍。

- 多体存储器
- 多操作部件
  - 通用部件被分解成若干个专用部件，如加法部件、乘法部件、除法部件、逻辑运算部件等，而且同一种部件也可以重复设置多个。
  - 只要指令所需的操作部件空闲，就可以开始执行这条指令（如果操作数已准备好的话）。
- 阵列处理机（并行处理机）

  更进一步，设置许多相同的处理单元，让它们在同一个控制器的指挥下，按照同一条指令的要求，对向量或数组的各元素同时进行同一操作，就形成了阵列处理机 。

3. 在单处理机中，资源共享的概念实质上是用单处理机模拟多处理机的功能

1.5.4 多机系统中并行性的发展

1. 多机系统遵循时间重叠、资源重复、资源共享原理，发展为3种不同的多处理机：

   同构型多处理机、异构型多处理机、分布式系统

2. 耦合度

   反映多机系统中各机器之间物理连接的紧密程度和交互作用能力的强弱。

   - 紧密耦合系统（直接耦合系统）：在这种系统中，计算机之间的物理连接的频带较高，一般是通过总线或高速开关互连，可以共享主存。

- 松散耦合系统（间接耦合系统）：一般是通过通道或通信线路实现计算机之间的互连，可以共享外存设备（磁盘、磁带等）。机器之间的相互作用是在文件或数据集一级上进行。

  表现为两种形式：

  - 多台计算机和共享的外存设备连接，不同机器之间实现功能上的分工（功能专用化），机器处理的结果以文件或数据集的形式送到共享外存设备，供其它机器继续处理。
  - 计算机网，通过通信线路连接，实现更大范围的资源共享。

3. 功能专用化（实现时间重叠）

- 专用外围处理机

  例如：输入/输出功能的分离

- 专用处理机

  如数组运算、高级语言翻译、数据库管理等，分离出来。

- 异构型多处理机系统

  由多个不同类型、至少担负不同功能的处理机组成，它们按照作业要求的顺序，利用时间重叠原理，依次对它们的多个任务进行加工，各自完成规定的功能动作。

4. 同构型多处理机系统

由多个同类型或至少担负同等功能的处理机组成，它们同时处理同一作业，能并行执行的多个任务

- 容错系统
  - 可重构系统

对计算机之间互连网络的性能提出了更高的要求。高带宽、低延迟、低开销的机间互连网络是高效实现程序或任务一级并行处理的前提条件

1.5.5 并行机的发展变化

并行机的发展可分为4个阶段。

1. 并行机的萌芽阶段（1964年～

- 20世纪60年代初期
  - **CDC6600**：非对称的共享存储结构，中央处理机采用了双**CPU**，并连接了多个外部处理器。
- 60年代后期，一个重要的突破
  - 在处理器中使用流水线和重复设置功能单元，所获得的性能提高是明显的，并比单纯地提高时钟频率来提高性能更有效。
- 在1972年，Illinois大学和Burroughs公司联合研制Illiac IV SIMD计算机（64个处理单元构成的）

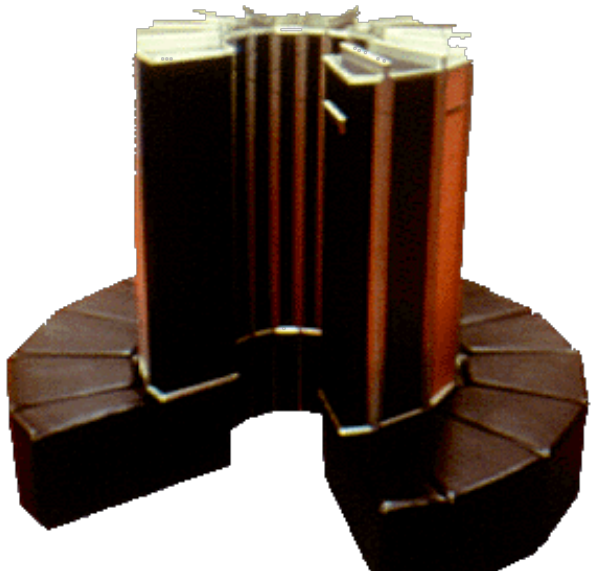
  在1975年 Illiac IV系统 （16个处理单元构成）

2. 向量机的发展和鼎盛阶段（1976年～1990年）

- 1976年，Cray公司推出了第一台向量计算机Cray-1
- 在随后的10年中，不断地推出新的向量计算机。

  包括：CDC的Cyber205、Fujitsu的VP1000/VP2000、NEC的SX1/SX2以、我国的YH-1等

- 向量计算机的发展呈两大趋势
  - 提高单处理器的速度
  - 研制多处理器系统



3. MPP出现和蓬勃发展阶段（1990年～1995年）

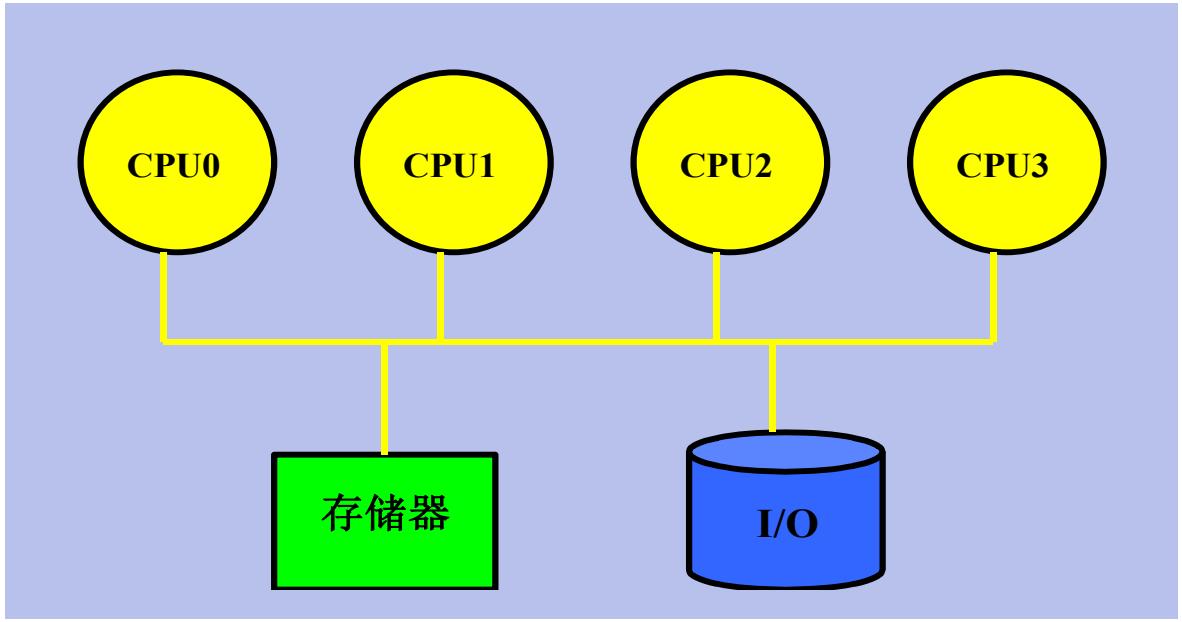
- 早期的MPP
  - **TC2000（1989年）、Touchstone Delta、Intel Paragon（1992年）、KSR1、Cray T3D（1993年）、IBM SP2（1994年）和我国的曙光-1000（1995年）等。（分布存储的MIMD计算机）**
- MPP的高端机器
  - **1996年，Intel公司的ASCI Red和1997年SGI Cray公司的T3E900 （万亿次高性能并行计算机）**
- 90年代的中期，在中、低档市场上，SMP以其更优的性能/价格比代替了MPP。

4. 各种系统结构并存阶段（1995年～2000年）

- 从1995年以后，PVP（并行向量处理机）、MPP、SMP、DSM（分布式共享存储多处理机）、COW等各种系统结构进入并存发展的阶段。
- MPP系统在全世界前500强最快的计算机中的占有量继续稳固上升，其性能也得到了进一步的提高。

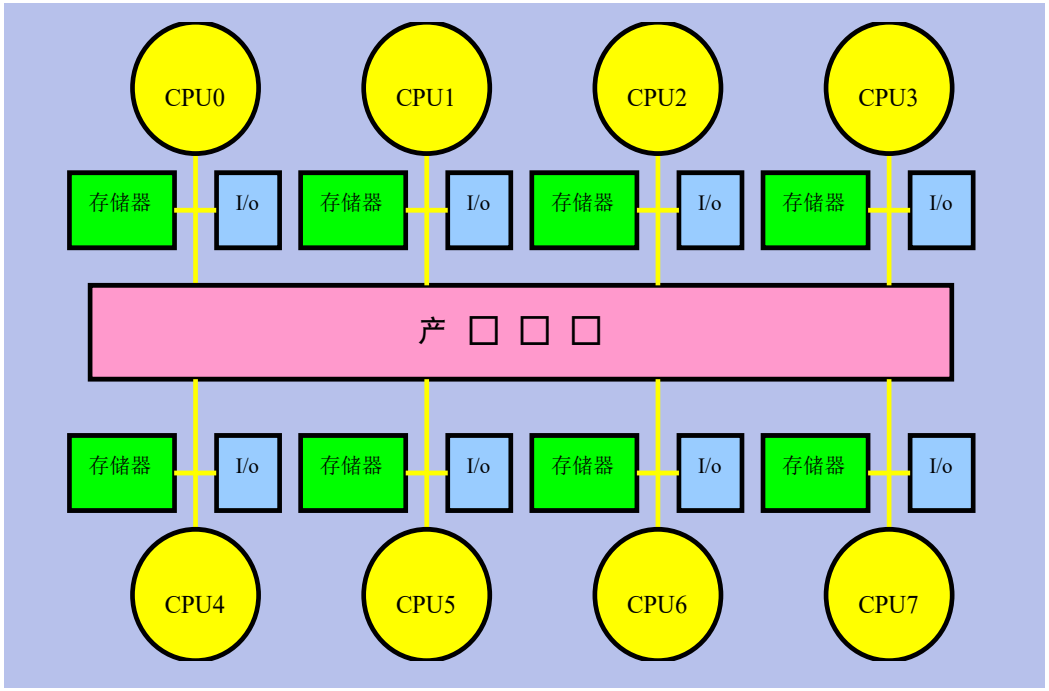
  如：**ASCI Red**的理论峰值速度已达到了**1Tflop/s**

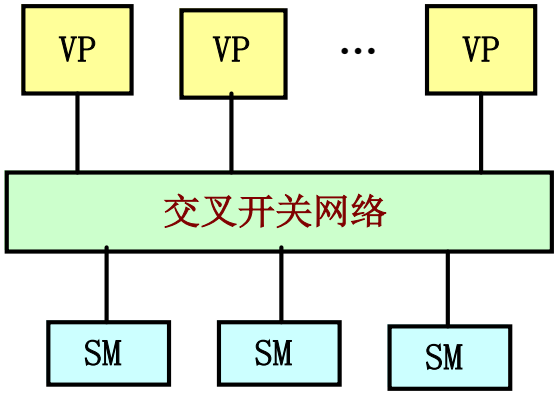
  SX4和**VPP700**等的理论峰值速度也都达到了**1Tflop/s**



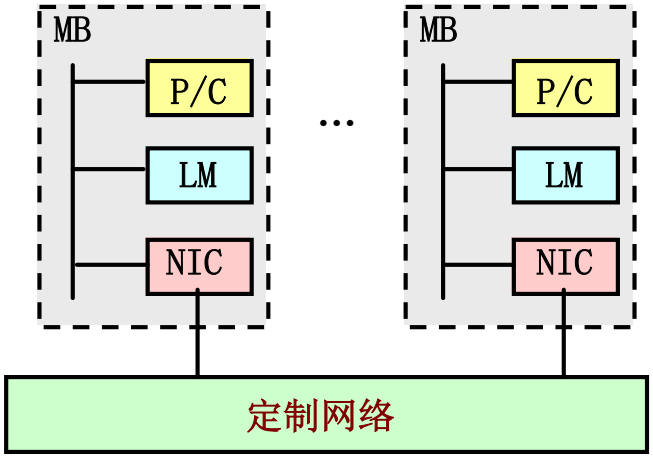
对称式共享存储器多处理机
SMP

分布式处理器多处理机DSM

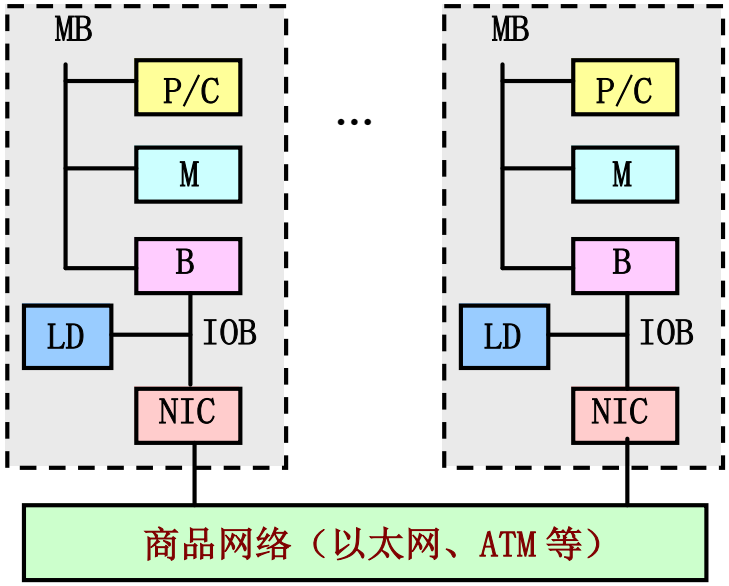




（a）PVP



（b）MPP



（c）机群

VP：向量处理器

SM：共享存储器模块

P/C：商品微处理器/Cache

LM、M：本地存储器

NIC：网络接口电路

MB：存储器总线

LD：本地磁盘

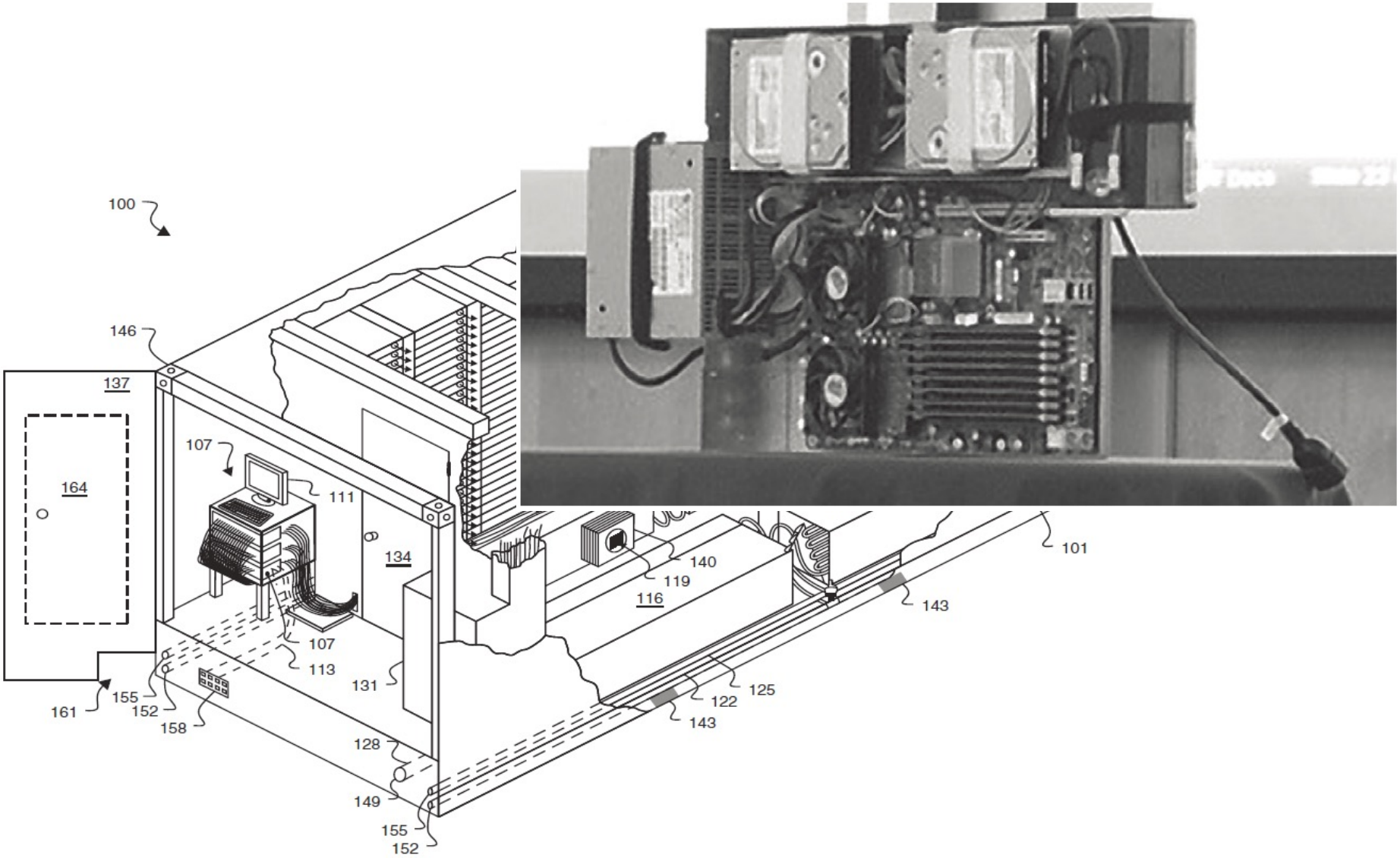
IOB：I/O 总线

B：存储总线与 I/O 总线之间

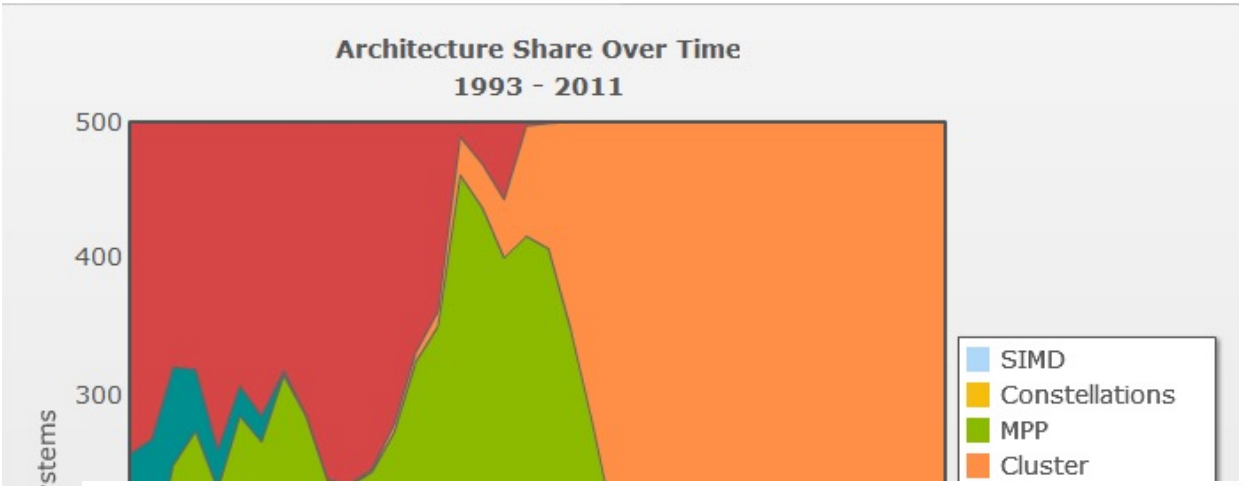
5. 机群蓬勃发展阶段（2000年以后）

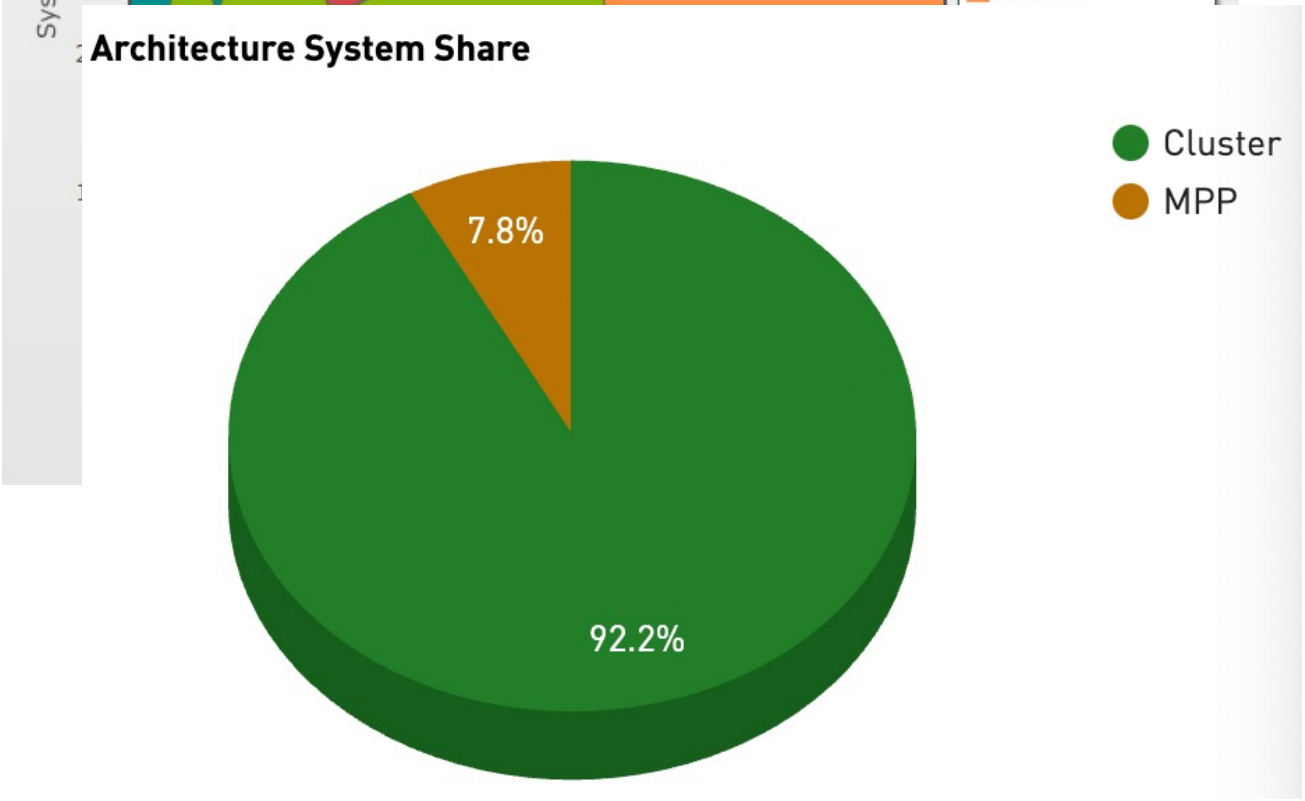
- 机群系统：将一群工作站或高档微机用某种结构的互连网络互连起来，充分利用其中各计算机的资源，统一调度、协调处理，以达到很高的峰值性能，并实现高效的并行计算。
- 1997年6月才有第一台机群结构的计算机进入Top500排名
- 2003年11月，这一数字已达到208台，机群首次成为Top500排名中比例最高的结构。
- 截至2008年6月，机群已经连续10期位居榜首，其数量已经达到400，占80%。

Google's WSC


100
146
137
164
107
111
134
107
113
131
155
152
158
161
128
149
155
152
140
119
116
143
101
125
122
143

- 机群已成为当今构建高性能并行计算机系统的最常用的结构。
- 1993年至2011年期间，Top500中机群和MPP的数量的分布情况。





1.6 计算机系统结构的分类

常见的计算机系统结构分类法有3种：

Flynn分类法、冯氏分类法和Handler分类法

1. Flynn分类法

- 按照指令流和数据流的多倍性进行分类。
  - 指令流：计算机执行的指令序列
  - 数据流：由指令流调用的数据序列
  - 多倍性：在系统最受限的部件上，同时处于同一执行阶段的指令或数据的最大数目

- 把计算机系统的结构分为4类
  - 单指令流单数据流SISD

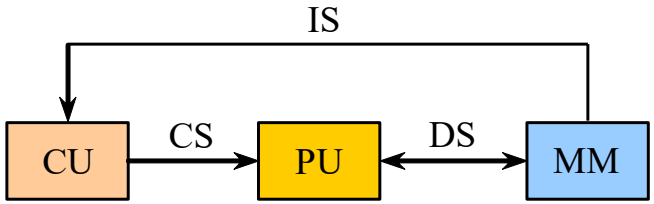
    （Single Instruction stream Single Data stream）
  - 单指令流多数据流SIMD

    （Single Instruction stream Multiple Data stream）
  - 多指令流单数据流MISD

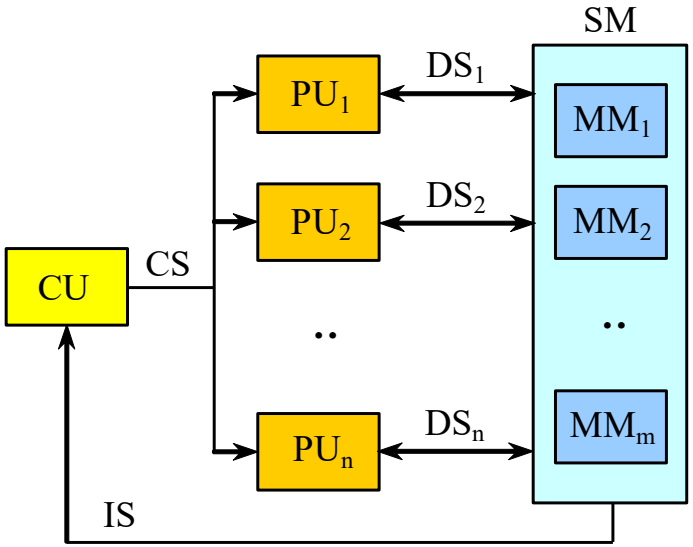
    （Multiple Instruction stream Single Data stream）
  - 多指令流多数据流MIMD

    （Multiple Instruction stream Multiple Data stream）

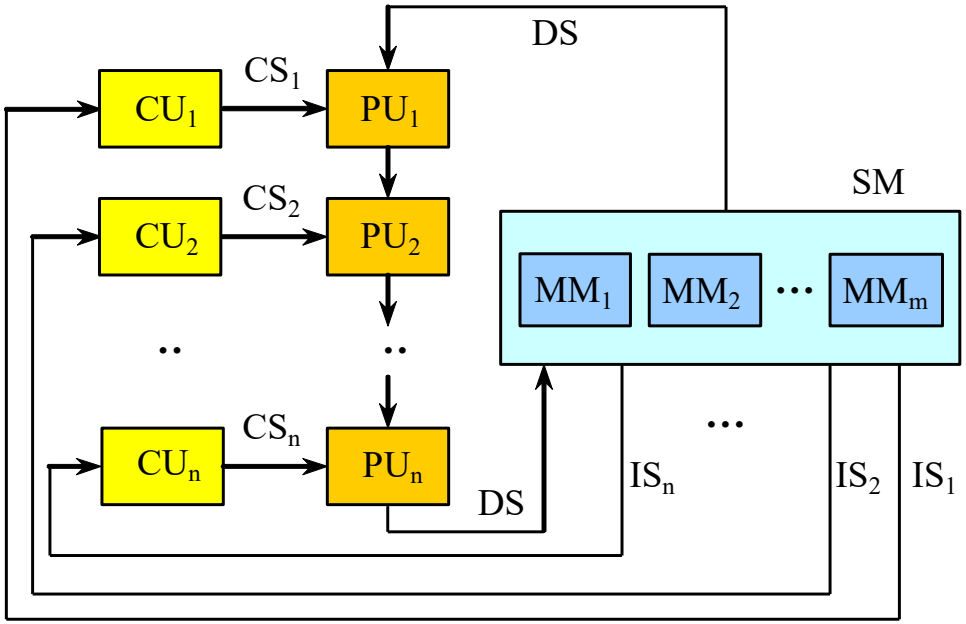
- 4类计算机的基本结构
  - IS：指令流
  - DS：数据流
  - CS：控制流
  - CU：控制部件
  - PU：处理部件
  - MM和SM：存储器



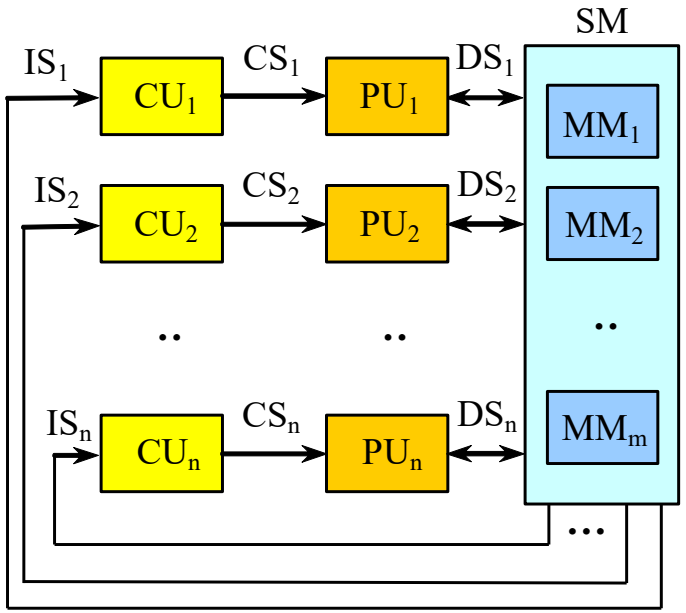
（a）SISD 计算机



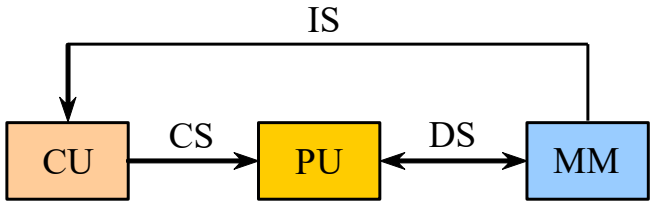
（b）SIMD 计算机



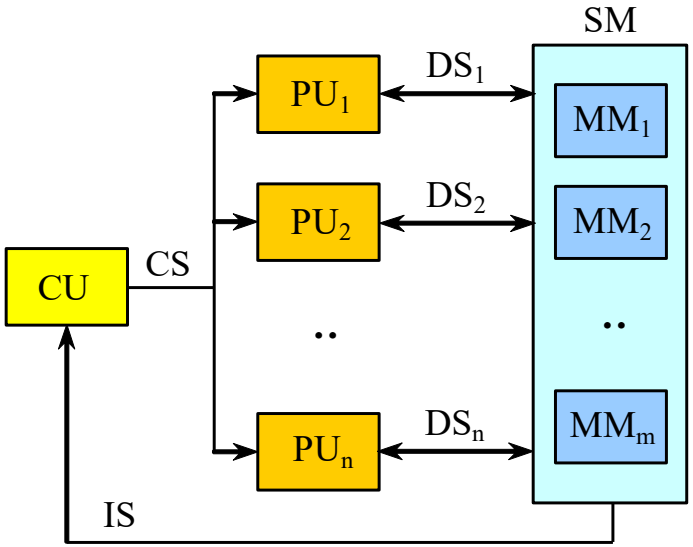
（c）MISD 计算机



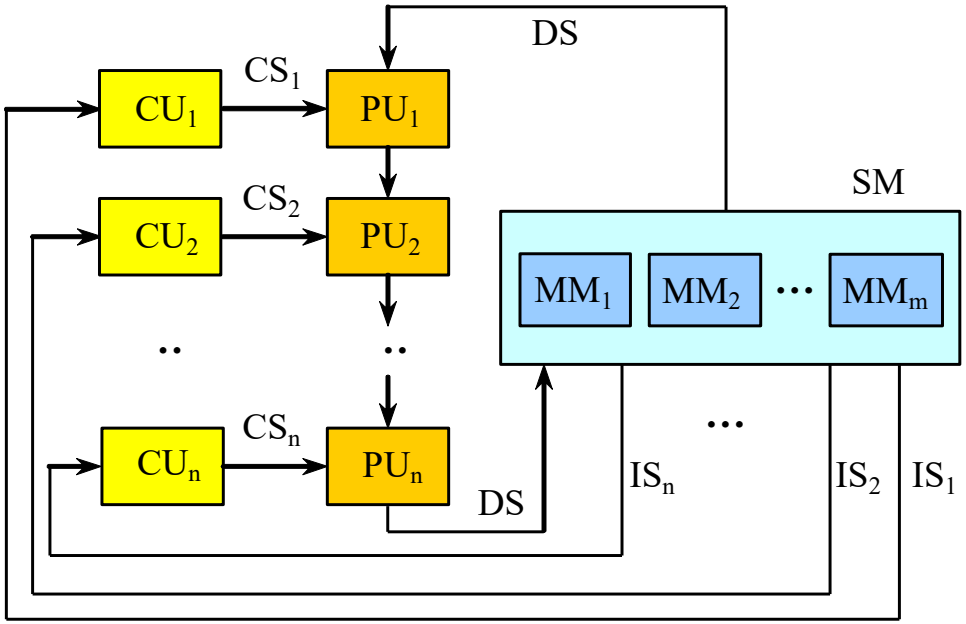
（d）MIMD 计算机



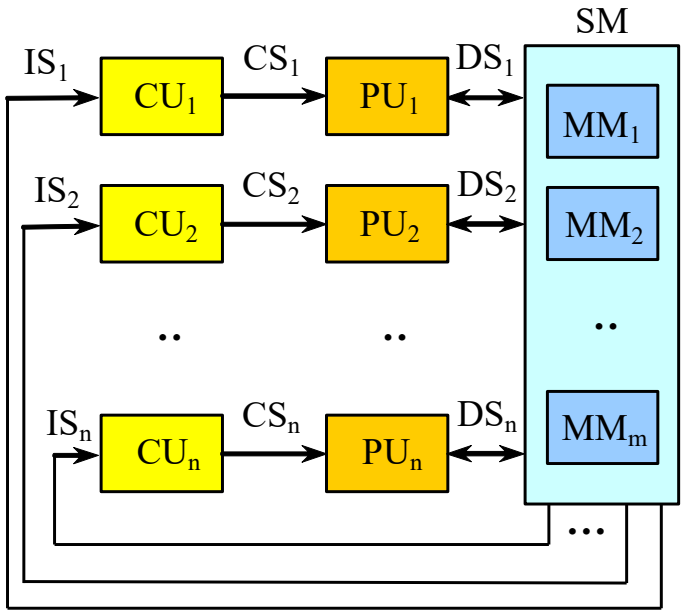
（a）SISD 计算机



（b）SIMD 计算机



（c）MISD 计算机

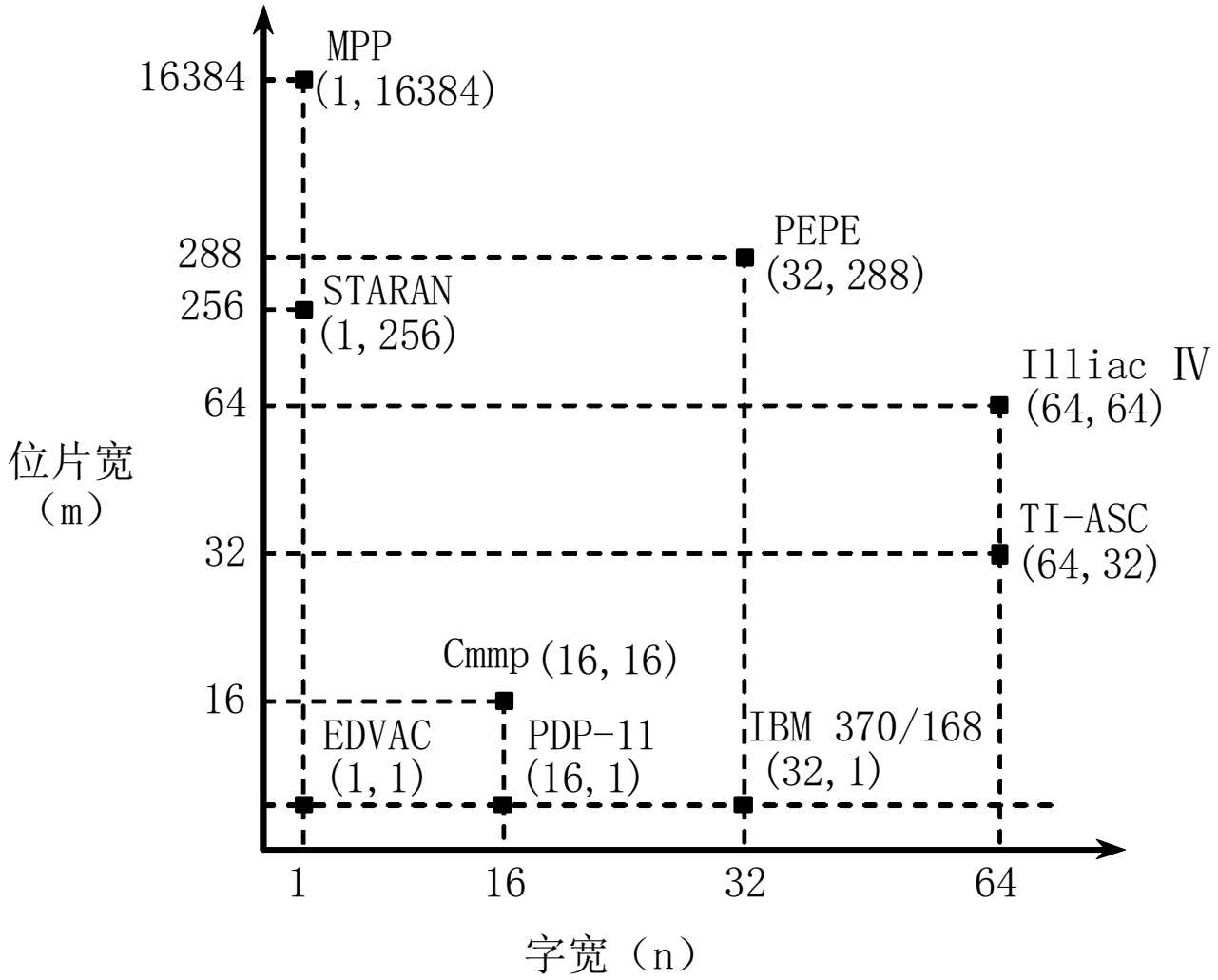


（d）MIMD 计算机

2. 冯氏分类法

- 用系统的最大并行度对计算机进行分类。
- 最大并行度：计算机系统在单位时间内能够处理的最大的二进制位数。

用平面直角坐标系中的一个点代表一个计算机系统，其横坐标表示字宽（n位），纵坐标表示一次能同时处理的字数（m字）。m×n就表示了其最大并行度。


MPP
(1, 16384)
16384
288
256
64
32
16
PEPE
(32, 288)
STARAN
(1, 256)
Illiac Ⅳ
(64, 64)
TI-ASC
(64, 32)
Cmmp (16, 16)
EDVAC
(1, 1)
PDP-11
(16, 1)
IBM 370/168
(32, 1)
位片宽
（m）
1
16
32
64
字宽（n）

- 4类不同最大并行度的计算机系统结构
  - 字串位串：n＝1，m＝1。

    (第一代计算机发展初期的纯串行计算机)
  - 字串位并： n＞1，m＝1。这是传统的单处理机，同时处理单个字的多个位，如16位、32位等。
  - 字并位串：n＝1，m＞1。同时处理多个字的同一位（位片）。
  - 字并位并：n＞1，m＞1。同时处理多个字的多个位。
- 平均并行度

  与最大并行度密切相关的一个指标。

  取决于系统的运用程度，与应用程序有关。

假设每个时钟周期内能同时处理的二进制位数为P_i，则T个时钟周期内的平均并行度为：

$$P_a = \frac{\sum_{i=1}^{T} P_i}{T}$$

系统在T个时钟周期内的平均利用率定义为：

$$\mu = \frac{P_a}{P_m} = \frac{\sum_{i=1}^{T} P_i}{TP_m}$$

3. Handler分类法

- 根据并行度和流水线对计算机进行分类。
- 把计算机的硬件结构分成3个层次
  - 程序控制部件（**PCU**）的个数**k**
  - 算术逻辑部件（**ALU**）或处理部件（**PE**）的个数**d**
  - 每个算术逻辑部件包含基本逻辑线路(**ELC**)的套数**w**
- 用公式表示

  t（系统型号）＝（k，d，w）
- 进一步改进

  t（系统型号）＝（k×k’，d×d’，w×w’）

- 进一步改进

  t（系统型号）＝（k×k’，d×d’，w×w’）

  - **k’**：宏流水线中程序控制部件的个数
  - **d’**：指令流水线中算术逻辑部件的个数
  - **w’**：操作流水线中基本逻辑线路的套数

例如：Cray-1有1个CPU，12个相当于ALU或PE的处理部件，可以最多实现8级流水线。字长为64位，可以实现1～14位流水线处理。所以Cray-1系统结构可表示为：

t（Cray-1）＝(1，12×8，64×（1～14）)

几个例子：

t（PDP-11）＝(1，1，16)

t（Illiac IV）＝(1，64，64)

t（STARAN）＝(1，8192，1)

t（Cmmp）＝(16，1，16)

t（PEPE）＝(1×3，288，32)

t（TI-ASC）＝(1，4，64×8)

自学内容

1. 计算机性能评测：执行时间，吞吐率，基准测试程序，MIPS，FLOPS…

- 作业
  - 1.7 1.10 1.11