尚德机构
SUNLANDS

密训·资料

计算机系统结构（上海）

1904

MI XUN ZI LIAO

编前语

试卷说明：

计算机系统结构科目,为上海计算机本科专业。考试题型有单项选择题、多项选择题、填空题、名词解释、简答题和论述题。

题型	分值
单选题	10 个×1 分=10 分
填空题	10 个×2 分=20 分
简答题	5 个×6 分=30 分
简单应用题	2 个×10 分=20 分
综合应用题	2 个×10 分=20 分

资料说明：

本资料按照章节和题型对知识点进行分类整理，并对相应题型可能会考的知识点加注了星标。星标是在参考大纲、课后题、历年真题等资料后，根据题型、分值、考试频率加注的。原则上星标数目越多，知识点越重要。本资料是为了指导学员考前做最后的复习，帮助学员理解，加速记忆。学员在拿到资料后，可以根据资料里的重点标记，有针对性的重点复习。

星标数	说明
一星	单选题
两星	填空题
三星	简答题
四星	简单应用题
五星	综合应用题

目录

第一章 概论.....3
第二章 数据表示、寻址方式与指令系统.....5
第三章 存储、中断、总线与 I/O 系统.....10
第四章 存储体系.....14
第五章 标量处理机.....16
第六章 向量处理机.....17
第七章 多处理机.....20
第八章 数据流计算机和归约机.....23

第一章 概论

1.1 计算机系统的层次结构

层次结构 ★★	（1）从使用语言的角度，一台由软、硬件组成的通用计算机系统可以被看成是按**功能**划分的多层**机器级**组成的层次结构。层次结构由**高到低**依次为**应用语言**机器级、**高级语言**机器级、**汇编语言**机器级、**操作系统**机器级、**传统机器语言**机器级和**微程序**机器级： ①第 0 级：微程序机器 M0（微指令系统），**微指令**由硬件直接执行； ②第 1 级：传统机器语言机器 M1（机器指令系统），用微指令程序解释机器指令；③第 2 级：操作系统机器 M2（作业控制语言等），一般用机器**语言程序解释**作业控制语句等；④第 3 级：汇编语言机器 M3（汇编语言），汇编语言程序经汇编程序翻译成机器语言程序；⑤第 4 级：高级语言机器 M4（高级语言），高级语言程序经编译程序翻译成汇编语言程序；⑥第 5 级：应用语言机器 M5（应用语言），应用语言程序经应用程序包翻译成高级语言程序。 （2）只有二进制机器指令，即传统所讲的机器语言与机器硬件直接对应，方可直接被硬件识别和执行。
“机器” ★★	被定义为能存储和执行相应语言程序的**算法**和**数据结构**的集合体。

1.2.1 计算机系统结构的定义和内涵

含义★★：从计算机的层次结构角度来看，系统结构是对计算机系统中各级界面的定义及其上下的功能分配。

计算机系统结构的属性★：（1）硬件能直接识别和处理的数据类型及格式等的数据表示；（2）最小可寻址单位、寻址种类、地址计算等的寻址方式；（3）通用/专用寄存器的设置、数量、字长、使用约定等的寄存器组织；（4）二进制或汇编指令的操作类型、格式、排序方式、控制机构等的指令系统；（5）主存的最小编址单位、编址方式、容量、最大可编址空间等的存储系统组织；（6）中断的分类与分级、中断处理程序功能及入口地址等的中断机构；（7）系统机器级的管态和用户态的定义与切换；（8）输入/输出设备的连接、使用方式、流量、操作结束、出错指示等的机器级 I/O 结构；（9）系统各部分的信息保护方式和保护机构等属性。

1.2.2 计算机组成与计算机实现的定义和内涵

计算机组成 ★★	（1）含义：计算机系统结构的**逻辑**实现，包括机器级内部的**数据流**和**控制流**的组成以及逻辑设计等； （2）计算机组成着眼于机器级内部各事件的排序方式与**控制机构**、各部件的**功能**及各部件间的联系； （3）要确定的方面：①数据通路宽度；②专用部件的设置；③各种操作对部件的共享程度；④功能部件的并行度；⑤控制机构的组成方式；⑥缓冲和排队技术；⑦预估、预判技术；⑧可靠性技术。

计算机实现 **★★**	（1）含义：计算机组成的物理实现，包括处理机、主存等部件的物理结构，器件的集成度和速度，器件、模块、插件、底板的划分与连接，专用器件的设计，微组装技术，信号传输，电源、冷却及整机装配技术等； （2）算机实现的设计着眼于**器件技术**和**微组装技术**，其中，器件技术起着主导作用。

1.3.1 软、硬件取舍的基本原则

从原理上讲，软件的功能可以用**硬件或固**件完成，硬件的功能也可以用软件模拟完成，只是它们在性能、价格、实现的难易程度上是不同的。★★

原则 **★★★**	（1）应考虑在现有**硬、器件**（主要是逻辑器件和存储器件）**条件下**，系统要有高的**性能价格比**，主要从实现费用、速度和其他性能要求来综合考虑；（2）要**考虑到准备采用和可能采用的组成技术**,使之尽可能不要过多或不合理地限制各种组成、实现技术的采用；（3）不能**仅从“硬”的角度考虑**如何便于应用组成技术的成果和便于发挥器件技术的进展,还应**从“软”的角度**把如何为编译和操作系统的实现以及为高级语言程序的设计提供更多、更好的硬件支持放在首位。
软、硬件功能分配比例对计算机系统性能的影响★★★	一般来说,提高硬件功能的比例可提高解题速度,减少程序所需的存储空间,但会增加硬件成本,降低硬件利用率和计算机系统的灵活性及适应性;而提高软件功能的比例可降低硬件成本,提高系统的灵活性、适应性,但解题速度会下降,软件设计费用和所需的存储器用量增加。

1.3.3 计算机系统设计的主要任务和方法

主要任务 **★★**	（1）主要任务包括系统结构、组成和实现的设计；（2）计算机系统设计首先要根据市场和应用情况，确定用户对计算机系统的功能、性能和价格的要求，其中，**应用软件对功能的确定其主要作用。**
方法 **★★**	（1）“由上往下”设计；（2）“由下往上”设计；（3）“从中间开始”向两边设计（通用机一般采用的方法）

1.4.1 软件发展对系统结构的影响

实现软件移植的技术★	（1）统一高级语言；（2）采用系列机；（3）模拟和仿真。
系列机	（1）采取系列机途径的办法：在软、硬件界面上设定好一种系统结构,其后,软件设计者按此设计软件;硬件设计者根据机器速度、性能、价格的不同,选择不同的器件、硬件和组成、实现技术,研制并提供不同档次的机器★★★；（2）意义：系列机较好地解决了软件环境要求相对稳定和硬、器件技术迅速发展的矛盾。软件环境相对稳定就可不断积累、丰富、完善软件，使软件产量、质量不断提高，同时又能不断采用新的器件和硬件技术，使之短期内便可提供新的、性能不断提高的机器★★★；（3）系列机软件必须保证向后兼容，力争向前兼容。向上同向前★★。（4）系列机技术只能应用在结构相同或相似的机器之间的汇编程序的软件移植。

模拟和仿真	（1）仿真：用微程序直接解释另一种及其指令系统的方法★★★； （2）仿真与模拟的主要区别在于解释用的语言。仿真是用微程序解释，其解释程序存储于控制存储器中；而模拟是用机器语言程序解释，其解释程序存储于主存中。★★★

1.4.2 应用的发展对系统结构的影响★★

计算机应用可归纳为向上升级的 4 类，它们是**数据**处理、**信息**处理、**知识**处理和**智能**处理。

口诀：数信知智

1.5.1 并行性的概念与开发

含义 ★★	并行性包含同时性和并发性二重含义，并发性指两个或多个事件在同一时间间隔内发生。
等级 ★★★	（1）从计算机系统执行程序的角度来看，并行性等级由低到高可分为四级。分别是：①**指令内**部：一条指令内部各个微操作之间的并行执行；②**指令之间**：多条指令的并行执行；③**任务或进程之间**：多个任务或程序段的并行执行；④**作业或程序之间**：多个作业或多道程序的并行执行； （2）从计算机系统中处理数据的角度来看，并行性等级从低到高可以分为四级。分别是：①**位串字串**：同时只对一个字的一位进行处理，这通常是指传统的串行单处理机， 没有并行性；②**位并字串**：同时对一个字的全部位进行处理，这通常是指传统的并行单处理机，开始出现并行性；③**位片串字并**：同时对许多字的同一位（称位片）进行处理，开始进入并行处理领域；④**全并行**——同时对许多字的全部或部分位组进行处理。
并行性开发的途径★★	（1）**时**间重**叠**：在并行性概 念中引入时间因素，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，加快硬件周转来赢得速度；（2）**资**源**重**复：在并行概念中引入空间因素，通过重复设置硬件资源来提高可靠性或性能。（3）**资**源共**享**：用软件方法，让多个用户按一定时间顺序轮流使用 同一套资源来提高资源利用率，相应地也就提高了系统的性能。 口诀：时叠资重资享

1.5.2 计算机系统的分类★★

把计算机系统分成单指令流单数据流（SISD）、单指令流多数据流（SIMD）、多指令流单数据流（MISD）和多指令流多数据流（MIMD）四大类。

（1）**SISD** 系统：传统的**单处理**器计算机；（2）**SIMD** 系统：具有代表性的例子是**陈列处理机**和**相联处理机**；(3)MISD 系统：有人把处理机间的宏流水及脉动陈列流水机都归属于 MISD 系统；（4）**MIMD** 系统：能实现**作业、任务、指令、数组**各级全面并行的**多机系统**。

第二章 数据表示、寻址方式与指令系统

2.1.1 数据表示与数据结构

数据表示★★：数据表示指的是能由计算机硬件**识别**和**引用**的数据类型，表现在它有对这种类型的数据进行操作的指令和运算部件。

计算机的**运算**类指令和**运算器**结构主要是按机器有什么样的数据表示来确定的。★★

2.1.2 高级数据表示

高级数据表示包括：（1）自定义数据表述；（2）向量、数组数据表示；（3）堆栈数据表示。

自定义数据表示★★	（1）包括：标志符数据表示和数据描述符； （2）标志符数据表示的优点★★★：①简化了指令系统和程序设计；②简化了编译程序；③便于实现一致性校验；④能由硬件自动变换数据类型；⑤支持数据库系统的实现与数据类型无关的要求，使程序不用修改即可处理多种不同类型的数据；⑥为软件调试和应用软件开发提供了支持。
数据描述符★★★	数据描述符和标志符的差别：标志符是和每个数据相连的，合存在一个存储单元中，描述单个数据的类型特征；数据描述符则是与数据分开存放，用于描述所要访问的数据是整块的还是单个的，访问该数据块或数据元素所要的地址以及其他信息等。
堆栈计算机★★★	（1）含义：有堆栈数据表示的计算机称为堆找计算机； （2）特点：①由高速寄存器组成的硬件堆栈，并附加控制电路，让它与主存中的堆栈区在逻辑上构成整体，使堆栈的访问速度是寄存器的，容量是主存的；②有丰富的堆栈操作指令且功能很强，可直接对堆栈中的数据进行各种运算和处理；③有力地支持了高级语言程序的编译。有力地支持了子程序的嵌套和递归调用；④有力地支持了子程序的嵌套和递归调用。

2.1.3 引入数据表示的原则

原则★★	（1）看系统的效率是否有显著提高，包括实现时间和存储空间是否有显著减少；（2）看引入这种数据表示后，其通用性和利用率是否提高。

2.1.4.1 浮点数尾数基值的选择★★★★★

条件：非负阶、正尾数、规格化	公式
最小尾数值	$1\times r_m^{-1}$
最大尾数值	$1\text{-}1\times r_m^{-1}$
最大阶值	2^p-1
最小值	r_m^{-1}
最大值	$r_m^{(2^p-1)}\times(1-r_m^{-m'})$
尾数个数	$r_m^{m'}\times(r_m-1)/r_m$
阶的个数	2^p
数的个数	$2^p\times r_m^{m'}\times(r_m-1)/r_m$

上述表格中中：p 为阶值位数，m 为尾数位数，r_m 为尾数基值，m′ 为以 rm 为基的尾数位数。

（1）浮点数阶值的位数 p 主要影响两个可表示区的大小，即可表示数的范围大小， 而尾数的位数 m 主要影响在可表示区中能表示值的精度。★★

（2）当阶值位数 p 一定时，尾数采用什么进制都会影响到数的可表示范围、精度及数在数轴上分布的**离散程度**。★★

（3）规格化正尾数：正尾数小数点后的第 1 个 r_m 进制数位不是 0 的数。★★

（4）r_m 越大，数在数轴上的分布越稀，数的表示精度自然就下降。可表示数的精度随 r_m 增大而单调下降。运算中的精度损失是运算中尾数右移出计算机字长，使有效数字丢失造成的，因此它不同于可表示数的精度。r_m 越大，尾数右移的机会越小，精度的损失就越小。★★

（5）表示比 e 指的是相同 p、m 位数时，在 r_m=2 的可表示最大值以内，采用 r_m>2 的可表示浮点数个数与 r_m=2 的可表示浮点数个数之比。r_m越大，在与 r_m=2 的浮点数相重叠的范围内，数的密度分布要稀得多。

2.1.4.2 浮点数尾数的下溢处理方法

截断法 ★★★★	（1）含义：将尾数超出计算机字长的部分截取； （2）最大误差在整数时接近于 1 (如“11：11…1”截断成“11：”），在分数时接近于 2^{-m} (本例为 2^{-2},如“.01 11…1”截断成“.01 ；对于正数总是产生负误差，除非那些圆点处无误差。因圆点间距相等，截断误差在不同尾数值时是均匀分布的。统计平均误差为负且较大，无法调节。 （3）优点：**实现最简单**，不增加硬件，不需要处理时间；缺点：平均误差大且无法调节。
舍入法 ★★★★	（1）含义：舍入法是在计算机运算的规定字长之外增设一位附加位，存放溢出部分的最高位，每当进行尾数下溢处理时，将附加位加 1 (二进制整数相当于加 0.5,二进制小数相当于加 $2^{-(m+1)}$）； （2）最大误差整数为 0.5 (如“10：10…0”舍入成“11：”），分数为 2^{-m}) (本例为 2^{-2},如“.01：10…0”舍人成“.10：”）。对于正数，误差有正有负 (如“.10 01". 1”舍入成“.10 ”，造成负误差；“.10 10…0 “舍入成“.11 ”，造成正误差；“.01 00…0”舍入成“.01 ”，无误差)。统计平均误差趋于 0 但略偏正，平均误差无法调节。 （3）优点：实现简单，增加的硬件很少，最大误差小，平均误差接近于 0。
恒置“1”法 ★★	恒置“1”法是将计算机运算的规定字长的最低位恒置为“1”。这种方法的好处是实现最简单，不需要增加硬件和处理时间，**平均误差趋于 0**。主要缺点是最大误差最大，比截断法的还要大。多用于中、高速计算机。
查表舍入法 ★★	ROM 查表舍入法速度较快，**平均误差**可**调节到 0**，是较好的方法。

计算机组成设计必须注意解决好数的下溢处理，因为这种精度损失对系统程序和应用程序设计者都是透明的，设计得不好，同样的题目在用不同下溢处理办法的计算机上会得到不同的运算结果。★★

2.2.1 寻址方式的三种面向★★

多数计算机都将主存、寄存器、堆栈分类编址，分别有**面向主存**、**面向寄存器**和**面向堆栈**的寻址方式。

2.2.3 程序在主存中的定位技术

定位技术 ★★	逻辑地址是程序员编程用的地址，主存物理地址是程序在主存中的实际地址。
静态再定位 ★★★	含义：利用 Von Neumann 型机器指令可修改的特点，在目的程序装入主存时，由装入程序用软件方法把目的程序的逻辑地址变换成物理地址，程序执行时，物理地址不再改变。
动态再定位 ★★★	含义：在执行每条指令时才形成访存物理地址的方法。

2.2.4 物理主存中信息的存储分布

为了使任何时候所需的信息都只用一个存储周期访问到，要求信息在主存中存放的地址必须是该信息宽度的整数倍。★★

2.3 指令系统的设计和优化

指令系统的设计★★	（1）指令系统的包括指令的功能和指令格式的设计； （2）指令类型一般分非特权和特权型两类：①非特权指令主要供应用程序员使用，也可供系统程序员使用；②特权型指令只供系统程序员使用，用户无权使用。
编译程序设计者要求指令系统应设计具有的特点★★★	（1）规整性。对相似的操作做相同的规定；（2）对称性；（3）独立性和全能性；（4）正交性；（5）可组合性；（6）可扩充性。
指令 ★★	含义：指令是由操作码和地址码两部分组成的。
指令字格式化优化的措施 ★★★	（1）采用扩展操作码，并根据指令的频度 pi 的分布状况选择合适的编码方式，以缩短操作码的平均码长；（2）采用诸如基址、变址、相对、寄存器、寄存器间接、段式存放、隐式指明等多种寻址方式，以缩短地址码的长度，并在有限的地址长度内提供更多的地址信息；（3）采用 0、1、2、3 等多种地址制 ,以增强指令的功能 ,这样从宏观上就越能缩短程序的长度，并加快程序的执行速度；（4）在同种地址制内再采用多种地址形式，如寄存器-寄存器、寄存器-主存、主存-主存等，让每种地址字段可以有多种长度，且让长操作码与短地址码进行组配；（5）在维持指令字在存储器中按整数边界存储的前提下，使用多种不同的指令字长度。

2.4 .1 两种途径和方向

途径 ★★	（1）如何进一步增强原有指令的功能以及设置更为复杂的新指令以取代原先由软件 子程序完成的功能，实现软件功能的硬化。按此方向发展，机器指令系统日益庞大和复杂。因此，称用这种途径设计 CPU 的计算机为复杂指令系统计算机（CISC）。这可从面向**目标程序**、面向**高级语言**、面向**操作系统**三个方面的优化实现来考虑； （2）如何通过减少指令种数和简化指令功能来降低硬件设计的复杂度，提高指令 的执行速度。按此方向发展，使机器指令系统精简，因此，称通过这种途径设计 CPU 的计算机为精简指令系统计算机。

2.4.2 按 CISC 方向发展和改进指令系统

面向目标程序的优化实现改进的途径 ★★★	（1）途径 1 通过对大量已有机器的机器语言程序及其执行情况，统计各种指令和指令串 的使用频度来加以分析和改进；（2）增设强功能复合指令来取代原先由常用宏指令或子程序（如双倍长运算、三角函数、开方、指数、二-十进制数转换、编辑、翻译等子程序）实现的功能，由微程序解释实现，不仅大大提高了运算速度，减少了程序调用的额外开销，也减少了子程序所占的主存空间。
面向高级语言的优化实现改进的途径 ★★★	（1） 通过对源程序中各种高级语言语句的使用频度进行统计来分析改进；（2） 如何面向编译，优化代码生成来改进。由于目前计算机上运行的绝大多数目标程序都是经编译系统生成的，从优化代码生成上考虑，应当增强系统结构的规整性，尽量减少例外或特殊的情况和用法，让所有运算都对称、均匀地在存储（寄存器）单元间进行。
面向操作系统的优化实现改进的途径 ★★★	（1） 通过对操作系统中常用指令和指令串的使用频度进行统计分析来改进。但这种改进的效果很有限；（2） 考虑如何增设专用于操作系统的新指令；（3）把操作系统中频繁使用的，对速度影响大的机构型软件子程序硬化或固化，改为直接用硬件或微程序解释实现；（4）发展让操作系统由专门的处理机来执行的功能分布处理系统结构。

2.4.3 按 RISC 方向发展和改进指令系统

CISC 存在的问题 ★★★	（1）指令系统庞大，一般指令在 200 条以上；（2）许多指令的操作繁杂，执行速度很低；（3）由于指令系统庞大，使高级语言编译程序选择目标指令的范围太大，因此，难以优化生成高效机器语言程序，编译程序也太长、太复杂；（4）由于指令系统庞大，各种指令的使用频度都不会太高，且差别很大，其中相当一部分指令的利用率很低。
设计 RISC 的基本原则 ★★★	（1）确定指令系统时，只选择使用频度很高的那些指令，再增加少量能有效支持操作系统、高级语言实现及其他功能的指令，大大减少指令条数，一般使之不超过 100 条；（2）减少指令系统所用寻址方式种类，一般不超过两种。简化指令的格式限制在两种之内，并让全部指令都是相同长度；（3）让所有指令都在一个机器周期内完成；（4）扩大通用寄存器数，一般不少于 32 个，尽量减少访存，所有指令只有存、取指令访存，其他指令一律只对寄存器操作；（5）为提高指令执行速度，大多数指令都用硬联控制实现，少数指令才用微程序实现；（6）通过精简指令和优化设计编译程序，简单、有效地支持高级语言的实现。

设计 RISC 结构的重叠寄存器窗口技术 ★★★	为减少访存，尽量让指令的操作在寄存器之间进行，以提高执行速度，缩短指令周期，简化寻址方式和指令格式；为更简单、有效地支持高级语言中大量出现的过程调用，减少过程调用中为保存主调过程现场，建立被调过程新现场，以及返回时恢复主调过程现场等所需的辅助操作；也为了能更简单、更直接地实现过程间的参数传递，大多数 RISC 计算机的 CPU 中都设有大量寄存器，让每个过程使用一个有限量的寄存器窗口，并让各过程的寄存器窗口部分重叠。

第三章 存储、中断、总线与 I/O 系统

3.1 存储系统的基本要求和并行主存系统

存储系统的基本要求 ★★	对存储系统的基本要求是**大容**量、**高速**度和**低价**格。 口诀：大容、高速、低价
最大频宽 ★★	最大频宽 B_M 是存储器连续访问时的频宽。

在存储器所用器件一定的条件下，容量越**大**，因其延迟增大会使速度**越低**；容量越大，存储器总价格会越大；存储器速度越快，价格也越高。★★

为了弥补 CPU 与存储器在速度上的差距，一条途径是在组成上引入并行和重叠技术，构成并行主存系统，在保持每位价格基本不变的情况下，使主存的频宽得到较大的提高。★★

并行主存系统 ★★★★★

（1）主存最大频宽（单体主存）：$B_M=W/T_M$

例：设主存采用模 m 多分体交叉存取，每个分体的存取周期为 T_M=2μs，要求主存实际频宽为 8MB/s，但实际频宽只能达到最大频宽的 0.6 倍。

①若分体宽度 W=4 字节，则主存模数应取多少才能满足要求?(m 取 2 的幂)

答案：主存最大频宽 Bm=m×W/TM

0.6×m×4/2≥8；解得 m≥6.667

所以：主存模数应取 8 才能满足要求。

②若主存模数为 8，则分体宽度应为多少才能满足要求?

答案：0.6×8×W/2≥8；解得 W≥3.333

所以：分体宽度应取 4 字节。

（2）每个存储周期所能访问到的平均字数：$B=\dfrac{1-(1-\lambda)^m}{\lambda}$

例：程序存放在模 32 单字交叉存储器中，设访存申请队的转移概率 λ=25%

①求每个存储周期能访问到的平均字数

答案：由于每个存储周期能访问到的平均字数：$B=\dfrac{1-(1-\lambda)^m}{\lambda}$

故将 m=32，λ=25%，代入公式可得：$B=\dfrac{1-(1-\lambda)^m}{\lambda}=\dfrac{1-(1-25\%)^{32}}{25\%}\approx 4$

②当模为 16 呢?由此可得到什么结论?

答案：由于每个存储周期能访问到的平均字数：$B=\dfrac{1-(1-\lambda)^m}{\lambda}$

故将 m=16，λ=25%，代入公式可得：$B=\dfrac{1-(1-\lambda)^m}{\lambda}=\dfrac{1-(1-25\%)^{16}}{25\%}\approx 3.96$

3.2.1 中断的分类和分级

分类 ★★★	（1）BM370 系统将中断分成机器校验、管理程序调用、程序性、外部、输入/输出的重新启动 **6** 类。机器校验中断是告诉程序发生了设备故障。可用 64 位机器**校验中断码**指明故障原因和严重性,更为详细的中断原因和故障位置可由机器校验保存区内容提供。这里包含有电源故障、运算电路的误动作、**主存出错**、通道动作故障、处理器的各种硬件故障等； （2）访管中断：在用户程序需要操作系统介入时，通过执行“访管”指令时发生的，访管原因由“访管”指令中的 8 位码指明。 （3）程序性中断：包括指令和数据的格式错、程序执行中出现异常（非法指令、目态下使用管态指令、主存访问方式保护、寻址超过主存容量、各种溢出、除数为 0、有效位为 0 等）以及程序的事件记录、监督程序对事件的检测引起的中断等； （4）外部中断：来自计算机外部，它包括各种定时器中断、外部信号中断及中断键中断； （5）输入/输出中断：CPU 与 I/O 设备及通道联系的工具，在输入/输出操作完成或者 I/O 通道或者设备产生故障时发出。 （6）重新启动中断：为操作员或另一台 CPU 要启动一个程序所用，CPU 不能禁止这种中断。
分级 ★★★	（1）分级原因：由于中断源相互独立而随机地发出中断请求，因此常常会同时发生多个中断请求。中断系统按中断源的级别高低来响应； （2）同一类的各中断请求的响应和处理的优先次序，一般不是由中断系统的硬件管理，而是由其软件或通道来管理的。而不同类的中断就要根据中断的性质、紧迫性、重要性以及软件处理的方便性把它们分成不同的级别。

3.2.2 中断的相应次序与处理次序

中断处理次序和中断响应次序的不同点 ★★★	中断响应的次序用排队器硬件实现，次序是由高到低固定的。为了能根据需要，由操作系统灵活改变实际的中断处理次序，很多计算机都设置了中断级屏蔽位寄存器，以决定某级中断请求能否进入中断响应排队器。只要能进入的，总是让高级别的优先响应。
终端系统的软、硬件功能的实质★★	终端系统的软、硬件功能的实质是**中断处理程序软件和中断相应硬件的功能分配。**

3.2.3 终端系统的软、硬件功能分配

中断系统的主要功能和要求★★★：中断系统的功能包括中断请求的保存和清除、优先级的确定、中断断点及现场的保存、对中断请求的分析和处理以及中断返回等。中断系统主要是要有高的中断响应速度，即从发出中断请求到进入中断处理程序的中断响应时间要短；其次是中断处理的灵活性。

3.3.1 总线的分类

分类 ★★	（1）总线按在系统中的位置分**芯片级**、**板级**和**系统级**等 3 级； （2）就总线**允许信息传送的方向**来说，可以有**单向**传输和**双向**传输两种，双向传输又有双向和全双向的不同； （3）总线按**用法**可分为**专用**和**非专用**两类。
专用总线 ★★★	（1）含义：只连接一对物理部件的总线；（2）优点：多个部件可以同时收/发信息，不争用总线，系统流量高；通信时不用指明源和目的，控制简单；任何总线的失效只会使连于该总线的两个部件不能直接通信，但它们仍可通过其他部件间接通信，因而系统可靠。

3.3.2 总线的控制方式

集中式总线控制★★	集中式总线控制按优先次序的确定可以有**串行链接**、**定时查询**和**独立请求** 3 种不同的方式。
集中式串行链接方式★★★	（1）**分配过程**：所有部件都经公共的“总线请求”线向总线控制器发出要求使用总线的申请。只有当“总线忙”信号未建立（即 总线空闲）时，“总线请求”才被总线控制器响应，送出“总线可用”信号，它串行地通过每个部件。如果某个部件接收到“总线可用”信号，但未发出过“总线请求”时，就将该信号继续送往下一个部件；如果该部件接收到“总线可用”信号并发出过“总线请求”时，则停止传送“总线可用”信号。该部件建立“总线忙”，并去除其“总线请求”，意即该部件获得了使用总线的权利，之后即可准备数据的传送； （2）**优点**：选择算法简单，用于解决总线控制分配的控制线的线数少，只需要 3 根，且不取决于部件的数量；部件的增减容易，只需简单地把它连到总线上或从总线上去掉即可，可扩充性好；由于逻辑简单，容易通过重复设置提高可靠性； （3）**缺点**：限制了总线的分配速度，增减和移动部件也受到限制。
集中式定时查询方式★★★	（1）**分配过程**：总线上的每个部件通过“总线请求”线发出请求，若总线处于空闲，“总线忙”信号未建立，则总线控制器收到请求后，让计数器开始计数，定时查询各部件以确定是谁发的请求。当查询线上的计数值与发出请求的部件号一致时，该部件就建立“总线忙”，使计数器停止计数，也即控制器中止查询。同时，去除该部件的“总线请求”，让该部件获得总线使用权，准备传送数据，直至该部件完成传送为止，去除“总线忙”。之后，“总线请求”线上仍有新的请求，就开始下一个总线分配过程； （2）**优缺点**：①优点：灵活性强；不会因某个部件失效而影响其他部件对总线的使用，可靠性高；②**缺点**：控制线的线数较多，需 $2+\lceil \log_2 N \rceil$ 根；可以共享总线的部件数受限于定时查询线的线数（编址能力），扩展性稍差；控制较为复杂；总线分配的速度取决于计数信号的频率和部件数，不能很高。

集中式独立请求方式★★★	（1）**分配过程**：共享总线的每个部件各自有一对“总线请求”和“总线准许”线。当部件请求使用总线时，送“总线请求”信号到总线控制器。只要总线闲着（“总线已被分配”线无信号），总线控制器就可以根据某种算法对同时送来的多个请求进行仲裁，以确定哪个部件可使用总线，并立即通过相应的“总线准许”线送回该部件，去除其请求，建立“总线已被分配”，该部件获得总线使用权，总线分配过程结束； （2）**优缺点**：①优点：总线分配速度快，所有部件的总线请求同时送到总线控制器，不用查询；控制器可以使用程序可控的预定方式、自适应方式、循环方式或它们的混合方式灵活确定下一个使用总线的部件；能方便地隔离失效部件的请求；②**缺点**：控制线数量过大，为控制 N 个设备必须有 2N+ 1 根控制线，而且总线控制器要复杂得多。

3.3.3 总线的通信技术

信息在总线上的传送方法基本上可分为**同步**和**异步**两种。★★

3.3.4 数据宽度与总线线数

数据宽度 ★★	（1）含义：I/O 设备取得 I/O 总线后所传达数据的总量； （2）单字（单字节）宽度适合于输入机、打印机等低速设备；
总线线数 ★★	在满足性能前提下应**尽量减少线数**。总线线数可通过用线的组合、编码及并/串一串/并转换来减少，但一般会降低总线的流量。
总线标准 ★★	总线标准一般包括**机械、功能、电气及过程**（同步）4 个方面的标准。

3.4.1 I/O 系统概述★★

I/O 系统：（1）I/O（输入/输出）系统包括输入/输出设备、设备控制器及与输入/输出操作有关的软、硬件；（2）输入/输出系统的发展经历了 3 个阶段，相对应于 3 种方式，即程序控制 I/O（包括全软件的、程序查询的、中断驱动的）、直接存储器访问（DMA）及 I/O 处理机方式。它们可分别用于不同的计算机系统，也可用于同一系统；（3）输入/输出设备分外存和传输设备两大类：①外存：磁盘、磁带、光盘等；②传输设备有键盘、鼠标、光笔、显示器、各种打印/印字机、声音输入/输出设备、图形扫描器、网络驱动器等。

3.4.2.1 通道处理机的工作原理

根据通道数据传送期中信息传送方式的不同，可分为**字节多路**、**数组多路**和**选择** 3 类通道。

字节多路通道	字节多路通道适用于连接**大量**的像光电机等字符类**低速设备**。★★它们传送一个字符（字节）的时间很短，但字符（字节）间的等待时间很长。因此，通道数据宽度为单字节，以字节交叉方式轮流为多台低速设备服务，使效率提高。字节多路通道又可以有多个子通道，各子通道能独立执行通道指令，并行地操作，以字节宽度分时进出通道。接在每个子通道上的多台设备也能分时使用子通道。★★★
数组多路通道 ★★	数组多路通道适合于连接多台磁盘等高速设备。

选择通道 ★★	选择通道适合于连接优先级高的磁盘等高速设备，让它独占通道，只能执行一道通道程序。

3.4.2.2 通道流量的设计

通道流量 ★★★★★	（1）含义：通道在数据传送期内，单位时间内传送的字节数； （2）字节多路通道没选择一台设备只传送一个字节，其通道极限流量★★★★★：$f_{\max \cdot byte}=\frac{1}{T_s+T_D}$，其中 T_S 为设备的时间，T_D 为传送一个字节的时间。 （3）数组多路通道每选择一台设备可传送完 K 个字节。如果要传送 N 个字节，就得分⌈N/K⌉次传送才行，每次传送都要选一次设备，通道极限流量 $$f_{\max \bullet block}=\frac{1}{T_S+KT_D}=\frac{1}{\frac{T_S}{K}+T_D}$$ 选择通道每选择一台设备就把 N 个字节全部传送完，通道极限流量：、 $$f_{\max \bullet select}=\frac{N}{T_S+NT_D}=\frac{1}{\frac{T_S}{N}+T_D}$$ （3）如果字节多路通道上所挂设备台数为 m，设备的速率 f_i 实际就是设备发出字节传送请求的间隔时间的倒数。M 台相同设备，其速率之和为 mf_i，这样，为了不丢失信息，就应满足 $\frac{1}{T_s+T_D}\geq f_i$，于是可求得在字节多路通道上能挂的设备台数 m，应满足 $$m\leq\frac{1}{(T_s+T_D)\cdot f}$$

第四章 存储体系

4.1.1 存储体系及其分支

虚拟存储器 ★★	（1）虚拟存储器是因主存容量满足不了要求而提出来的；（2）从 CPU 上看，速度是接近于主存的，容量是辅存的，每位价格是接近于辅存的。
Cache 存储器 ★★	（1）因主存速度满足不了要求而引出了 Cache 存储器；（2）含义：在 CPU 和主存之间增设高速、小容量、每位价格较高的 Cache，用辅助硬件将 Cache 和主存构成整体。

4.1.2 存储体系的构成依据

存储层次构成的主要依据是程序的局部性。

4.2.1 虚拟存储器的管理方式

根据存储映像算法的不同，可有多种不同的存储管理方式的虚拟存储器，其中主要有段式、页式、段页式 3 种：

段式管理 ★★	程序都有模块性，一个复杂的大程序总可以分解成多个在逻辑上相对独立的模块。这些模块可以是主程序、子程序或过程，也可以是数据块。

页式管理 ★★	页式存储是把主存空间和程序空间都机械地等分成固定大小的页（页面大小随计算机而异，一般在 512B 到几 KB 之间），按页顺序编号。
段页式管理	段页式存储是把实（主）存机械地等分成固定大小的页，程序按模块分段，每个段又分成与实主存页面大小相同的页。每道程序通过一个段表和相应的一组页表进行定位。

4.2.2 页式虚拟存储器的构成

地址的映像 ★★★	地址的映像是将每个虚存单元按某种规则（算法）装入（定位于）实主存，建立起 多用户虚地址与实（主）存地址之间的对应关系。
目录表法 ★★	把页表压缩成只存放已装入主存的那些虚页（用基号 b 和标识）与实页位置的对应关系。

按内容访问的相联存储器不同于按地址访问的随机存储器。★★

页面替算法 ★★	替换算法的确定主要看主存是否有高的命中率，也要看算法是否便于实现，辅助软、硬件成本是否低。
随机算法	软的或硬的随机数产生器产生主存中要被替换页的页号。
先进先出算法	选择最早装入主存的页作为被替换的页。
近期最少使用算法★★	含义：选择近期最少访问的页作为被替换页。
堆栈型的替算法★★★★★	（1）LRU 算法在主存中保留的是 n 个最近使用的页，它们又总是被包含在 n+1 个最近使用的页中，所以 LRU 算法是堆栈型算法★★； （2）如果替算法满足： $n < L_t$，时，$B_t(n) \subset B_t(n+1)$ $n \geq L_t$，时，$B_t(n) = B_t(n+1)$ 用堆栈模拟时，主存在 t 时间点的状况用堆栈 S_t 表示，S_t 是 L_t 个不同页面号在堆栈中的有序集，S_t（1）是 t 时间点的 S_t 的栈顶项，S_t（2）是 t 时间点的 S_t 的次栈顶项，依次类推。由于堆栈型算法的包含性，必有 $n < L_t$，时，$B_t(n) = \{B_t(1), B_t(2), \ldots, B_t(n)\}$ $n \geq L_t$，时，$B_t(n) = \{B_t(1), B_t(2), \ldots, B_t(L_t)\}$

4.3.2 地址的影响与变换

全相联映像：全相联映像法的优点是块冲突概率最低，只有当 Cache 全部装满才可能出现块冲突，所以，Cache 的空间利用率最高。★

4.3.4 Cache 存储器的透明性及性能分析

写回法 ★★★	含义：在 CPU 执行写操作时，信息只写入 Cache，仅当需要替换时，才将改写过的 Cache 块先写回主存，然后再调入新块。

写直达法 ★★★	含义：利用 Cache 存储器在处理机和主存之间的直接通路，每当处理机写入 Cache 的同时，也通过此通路直接写入主存。
Cache 的取算法★★	（1）适当选择好 Cache 的容量、块的大小、组相联的组数和组内块数，是可以保证有较高的命中率的；（2）为了便于硬件实现，通常在访问主存第 i 块（无论是否已取进 Cache）时，只预取顺序的第 i + 1 块。至于何时取进该块，可有**恒预取**和**不命中时预取**两种方法。
Cache 存储器的性能分析 ★★	评价 Cache 存储器的性能**主要是看命中率的高低**，而命中率与块的大小、块的总数（即 Cache 的总容量）、采用组相联时组的大小（组内块数）、替换算法和地址流的簇聚性等有关。

第五章 标量处理机

5.1.1 重叠原理与一次重叠

解释一条机器指令的微操作可归并成取指令、分析和执行三部分。

取指 ★★	含义：按指令计数器的内容访主存，取出该指令送到指令寄存器。
标量处理机的顺序解释★★	（1）优点：控制简单，转入下条指令的时间易于控制；（2）指令的重叠解释是在解释第 k 条指令的操作完成之前，就可以开始解释 k+1 条指令。
实现指令的重叠解释必须在计算机组成上满足的要求 ★★★	（1）要解决访主存的冲突。 （2）要解决“分析”与“执行”操作的并行。 （3）要解决“分析”与“执行”操作控制上的同步。 （4）要解决指令间各种相关的处理。
一次重叠 ★★	指令分析部件和指令分析部件任何时候只有相邻两条指令在重叠解释的方式为“一次重叠”。
数相关 ★★	数相关不只会发生在主存空间，还会发生在通用寄存器空间。

5.1.2 相关处理★★

（1）主存空间数相关是相邻两条指令之间出现对主存同一单元要求先写而后读的关联。如果让“执行与“分析 k + 1 ”在时间上重叠，就会使“分析 k+ 1 ”读出的数不是第 A: 条指令执行完应写入的结果而出错。要想不出错，只有推后“分析 k + 1 ”的读。

（2）推后“分析 k+1 ”和设置“相关专用通路”是解决重叠方式相关处理的两种基本方法。

5.2.2.2 流水的分类

流水的分类 ★★	流水按处理的级别可分为部件级、处理机级和系统级。
流水线分类 ★★	按多功能流水线的各段能否允许同时用于多种不同功能连接流水，可把流水线分为**静态流水线**和**动态流水线**。
静态流水线★	静态流水线在某一时间内各段只能按一种功能连接流水，只有等流水线全部流空后， 才能切换成按另一种功能连接流水。

动态流水线★	动态流水线的各功能段在同一时间内可按不同运算或功能连接。

标量流水机没有向量数据表示，只能用标量循环方式来处理向量和数组，如 **Amdahl470V/6** 及后面要介绍的 **IBM360/91**。

5.2.2 标量流水线的主要性能

标量流水处理机的性能主要是吞吐率 T_p、加速比 S_p 和效率 η。

吞吐率和加速比★★★★	（1）吞吐率：流水线单位时间里能流出的任务数或结果数； （2）为了提高流水线的最大吞吐率，首先要找出瓶颈，然后设法消除此瓶颈，然而，并不是所有的子过程都是能再细分的。 （3）流水线的实际吞吐率为：$T_p=\frac{n}{m\Delta t+(n-1)\Delta t}$，其中 m 为流水线个数，$\Delta t$ 为 m 段流水线的各段经过时间，n 为完成的任务个数。

5.2.3 标量流水机的相关处理和控制机构

任务在流水线中流动顺序的安排和控制可以有两种方式：①顺序流动方式或同步流动方式：让任务（指令）流出流水线的顺序保持与流人流水线的顺序一致；②异步流动方式：让流出流水线的任务（指令）顺序可以和流入流水线的顺序不同。★★

全局性相关★★	（1）含义：已进入流水线的转移指令（尤其是条件转移指令）和其后续指令之间相关； （2）常用的处理方法：①使用猜测法；②加快和提前形成条件码；③采取延迟转移；④加快短循环程序的处理
非线性流水线的调度★★★★★	（1）将流水线中所有各段对一个任务流过时会征用同一段的节拍间隔数汇集在一起，构成一个延迟禁止表 F； （2）可以用一个有 N-1 位的位向量来表示后续新任务间隔各种不同拍数送入流水线时，是否会发生功能段使用的冲突，称此位向量为冲突向量 c，冲突向量（$c_{N-1}...c_i...c_2c_1$）中第 i 位的状态表示与当时相隔 i 拍给流水线送入后续任务是否会发生功能段的使用冲突。如果不会发生冲突，令该位为“0”，表示允许送入；否则让该位为“1”，表示禁止送入。 （3）流水线最大吞吐率 $T_{P_{\max}}=$（任务/拍）
标量流水机对局部性相关的处理一般采用总线式分布方式控制管理，包括：（1）相关的判断主要是靠分布于各寄存器的“忙位”标志来管理；（2）在分散于各流水线的入、出端处设置若干保存站来缓存信息；（3）用站号控制公共数据总线的连接作相关专用通路，使之可为多个子过程的相关所共用；（4）一旦发生相关，用更换站号来推后和控制相关专用通路的连接；（5）采用多条流水线，每条流水线人端有多组保存站，以便发生相关后，可以采用异步的流动方式。	

第六章 向量处理机

6.1 向量的流水处理与向量流水处理机

向量处理机★★	（1）向量处理机是由向量数据表示的处理机，分向量流水处理机和陈列处理机两类； （2）向量流水处理机是以时间重叠途径开发的，而陈列处理机是以资源重复途径开发的。

纵向（垂直）处理方式★★	含义：采用对整个向量按相同操作都执行完之后再转去执行别的操作，才能较好地发挥流水处理的效能。
向量横向处理★★	向量横向处理是向量的处理方式，但不是向量的流水处理方式；而向量纵向处理和分组纵横处理既是向量的处理方式，也是向量的流水处理方式。

6.2.1 陈列处理机的构形和特点

构形★★	（1）分布式存储器陈列机的构形：为了高速有效地处理向量数据，这种构形要求能把数据合理地预分配到各个处理单元的局部存储器中，使各处理单元 PE_i 中的数据运算；（2）集中式共享存储器的陈列机的构形。
特点★★★	（1）阵列处理机利用的是资源重复，而不是时间重叠；利用的是并行性中的同时性，而不是并发性； （2）其设备利用率却可能没有多个单功能流水线部件的那样高，只有在硬件价格有了大幅度下降以及系统结构有了较大改进的情况下，阵列处理机才能有好的性能价格比； （3）阵列处理机提高速度主要是靠增大处理单元数，速度提高的潜力要比向量流水处理机大得多； （4）阵列处理机使用简单、规整的互连网络来确定处理单元间的连接； （5）阵列处理机在机间互连上比固定结构的单功能流水线灵活，使相当一部分专门问题上的工作性能比流水线处理机高得多，专用性强得多。
累加和★★★★	算法步骤**（举例子说明）**： **例：**求向量累积和 $S=\sum_{i=0}^{15}A(i)$，在 SISD 计算机上实现需 16 次加法。现在陈列处理机上用成对递归算法，只需 $\log_2 16=4$ 次加法，可求得前 1 个，前 2 个，...，前 16 个元素之和。设原始数据 A（i）分别存放在 PEM_i 的α单元，其中，0≤i≤15 答案：（1）置全部 PEi 为活跃状态，0≤i≤15 （2）置全部 A(i)为 PE_i 的α单元读到相应 PE_i 的累加寄存器 RGR_i ,0≤i≤15； （3）令 K=0； （4）将全部 PE_i 的（RGA_i）转送到传送寄存器 RGR_i，0≤i≤15； （5）将全部 PE_i 的（RGA_i）经过互联网络各右传送 2^k 步举 $_i$，0≤i≤15； （6）令 $j=2^k-1$； （7）置 PE_0~PE_j 为不活跃状态； （8）处理活跃状态的所有 PE_i 执行（RGA_i）:=（RGA_i）+（RGR_i）,j<i≤15； （9）K：K+1； （10）如 K<4，则转回（4）； （11）置全部 PE_i 为活跃状态，0≤i≤15； （12）将全部 PE_i 的累积寄存器内容（RGA_i）存入相应 PEi 的α+1 单元中，0≤i≤15

6.3 SIMD 计算机的互连网络

SIMD 系统的互连网络的设计目标★★★	结构不要过分复杂，以降低成本；互连要灵活，以满足算法和应用的需要；处理单元间信息交换所需的传送步数要尽可能少，以提高速度性能；能用规整单一的基本构件组合而成，或者经多次通过或者经多级连接来实现复杂的互连，使模块性好，以便于用 VLSI 实现并满足系统的可扩充性。
操作方式★★	有同步、异步及同步组合三种。
网络的拓扑结构	互连网络入、出端可以连接的模式，有静态和动态两种。
动态网络	有单级和多级两类。
三维的立方体单级网络★★	（1）三维的立方体单级网络有 3 种互连函数：$Cube_0$、$Cube_1$和 $Cube_2$，$Cube_i$函数表示相连的入端和出端的二进制编号只在右起第 i 位（i= 0, 1，2)上 0、1 互反，其余各位代码都相同； （2）互连函数：$Cube_i(P_{n-1}...P_i...P_1P_0)=P_{n-1}...P_i...P_1P_0$ ★★★★★
PM2I 单级网络★★★★★	（1）PM2I 单级网络是“加减 2^i”（Plus-Minus2^i）单级网络的简称。能实现与 j 号处理单元直接相连的是号为 $j\pm 2^i$ 的处理单元，即： $$\begin{cases} PM2_{+i}(i)=j+2^i \mod N \\ PM2_{-i}(i)=j-2^i \mod N \end{cases}$$ （2）ILLIACIV 处理单元的互连也是 PM2I 的特例，采用了其中的 $PM2_{\pm 0}$ 和 $PM2\pm_{n/2}$ (即 $PM2_{\pm 3}$) 4 个互连函数。
混洗交换单级网络★★★★★	（1）互连函数：$Shuffle\ (P_{n-1}P_{n-2}...P_1P_0)=P_{n-2}...P_1P_0P_{n-1}$ （2）在混洗交换网络中，最远的两个入、出端号是全“0”和全“1”，它们的连接需要次交换和 n - 1 次混洗，所以其最大距离为 2n- 1。
基本的多级互连网络★★	不同的多级互连网络，在所用的交换开关、拓扑结构和控制方式上各有不同。
多级立方体网络★★	多级立方体网络有 STARAN 网络、间接二进制 n 方体网络等。
STARAN 网络★★	STARAN 网络用作交换网络时，采用级控制，实现的是交换函数。

6.4 共享主存构形的陈列处理机中并行存储器的无冲突访问

为了能使行或列的各元素都能并行访问，采取将数据在存储器中**错位存放**，但是该方案可造成主对角线上各元素的并行访问冲突，致使实际频宽下降一半；次对角线上各元素的访问则都发生冲突，使实际频宽降低成与串行一样。★★

6.5 脉动陈列流水处理机

脉动陈列结构★★	（1）脉动阵列结构是由一组处理单元（PE）构成的阵列；（2）为了执行多种计算，脉动型系统内的输入数据流和结果数据流可以在多个不同方向上以不同速度向前搏动。

脉动陈列结构的特点★★★	（1）结构简单、规整，模块化强，可扩充性好，非常适合用超大规模集成电路实现； （2）PE 间数据通信距离短、规则，使数据流和控制流的设计、同步控制等均简单规整； （3）脉动阵列中所有 PE 能同时运算，具有极高的计算并行性，可通过流水获得很高的运算效率和吞吐率； （4）脉动阵列结构的构形与特定计算任务和算法密切相关，具有某种专用性，限制了应用范围，这对 VLSI 是不利的。
发展通用脉动陈列结构的途径	（1）通过增设附加的硬件，对阵列的拓扑结构和互连方式用可编程开关进行重构； （2）用软件把不同的算法映像到固定的阵列结构上； （3）探寻与问题大小无关的脉动处理方法，以及 VLSI 运算系统的分割矩阵算法，使它们可以克服阵列只能求解固定大小题目的缺陷，同时探寻发展适合一类计算问题的通用算法和相应的设置方案。

第七章 多处理机

7.1 多处理机的概念、问题和硬件结构

多处理机	（1）含义：有**两台以上**的处理机，**共享 I/O 子系统**，机间经共享主存或高速通信网络通信，在统一操作系统控制下，协同求解大而复杂问题的计算机系统。 （2）使用多处理机的目的：①通过多台处理机对多个作业、任务进行并行执行来提高解题速度，从而提高系统的整体性能；②使用冗余的多个处理机通过重新组织来提高系统的可靠性、适应性和可用性； （3）多处理机是属于多指令流多数据流的系统★★； （4）与单指令流多数据流的陈列处理机的区别★★★：①在硬件结构上，它的多个处理机要用多个指令部件分别控制，通过共享主存或机间互连网络实现异步通信；②在算法上，不限于向量、数组处理，还要挖掘和实现更多通用算法中隐含的并行性；③在系统管理上，要更多地依靠操作系统等软件手段，有效地解决资源分析和管理，特别是任务分配、处理机调度、进程的同步和通信等问题； （5）种类：多处理机可以有同构 型异构型和分布型 3 种。★★ （6）多处理机执行并发任务所需的处理机的机数是**不固定的**。各处理机进入或退出任务的时间及所需资源的变化比较大。必须研究如何较好地解决动态的资源分配和任务调度，让各处理机的负荷尽可能**均衡**，并要防止**死锁**。

多处理机的构形★★	（1）多处理机有紧耦合和松耦合两种不同的构形：①紧耦合多处理机是通过共享主存实现处理机间通信的，其通信速率受限于主存频宽；②松耦合多处理机可分为**非层次型**和**层次型**两种构形。
多处理机的互连一般采用**总线**、**环形互连**、**交叉开关**、**多端口存储器或蠕虫穿洞寻径网络**等几种形式。随着技术的发展，当处理机的机数较多时，也有类似 SIMD 的多级网络。★★	

7.3.1 并行算法/7.3.2 程序并行性的分析

定义	可同时执行的多个进程的集合，各进程可相互作用、协调和并发操作。
种类★★	（1）按运算基本对象，并行算法可分为数值型的和非数值型两类； （2）按并行进程间的操作顺序不同，并行算法又分为同步型、异步型和独立型 3 种：①同步型并行算法：并行的各进程间由于相关，必须顺次等待；②异步型并行算法：并行的各进程间执行相互独立，不会因相关而等待，只是根据执行情况决定中止或继续；③独立型并行算法是指并行的各进程间完全独立，进程之间不需要相互通信。 （3）根据各处理机计算任务的大小（即任务粒度）不同，并行算法又分为细粒度、中粒度和粗粒度 3 种：①细粒度并行算法一般指向量或循环级的并行；②中粒度并行算法一般指较大的循环级并行，并确保这种并行的好处可以补偿因并行带来的额外开销；③粗粒度并行算法则一般是指子任务级的并行。
多处理机并行算法研究思路★★★★★	为了评价所提出的并行算法的性能效率，用 P 表示可并行处理的处理机机数；用 T_p 表示 P 台处理机运算的级数，即树高；用多处理机的加速比 S_p，表示单处理机顺序运算的级数:T_1 与 P 台处理机并行运算的级数 T_p 之比 ;用心表示 P 台处理机的设备利用率（效率），$E_p = S_p / P$ 可见，$S_p \geq 1$ 时，会使 $E_p \leq 1$，即运算的加速总是伴随着效率的下降。
两个程序段之间若有**先写后读**的数据相关，**不能并行**，只在**特殊**情况下可以**交换串行**；若有先读后写的数据**反相关**，**可以并行执行**，但必须保证其写入共享主存时的先读后写次序，不能交换串行；若有**写-写的数据输出相关**，**可以并行执行**，但同样需保证其写入的先后次序，不能交换串行；若**同时有先写后读和先读后写两种相关**，以交换数据为目的时，必须并行执行，且**读、写要完全同步，不许顺序串行和交换串行**；若**没有任何相关**或仅有源数据相同时，**可以并行、顺序串行和交换串行。★**	

7.3.3 并行语言与并行编译

FORK 语句的表示形式★★★★★	FORK 语句的形式为 FORK m 其中 m 为新进程开始的标号。执行 FORK m 语句时， 派生出标号为 m 开始的新进程，具体为：准备好这个新进程启动和执行所必需的信息；如果是共享主存，则产生存储器指针、映像函数和访问权数据；将空闲的处理机分配给派 生的新进程，如果没有空闲处理机，则让它们排队等待；继续在原处理机上执行 FORK 语 句的原进程。

IOIN 语句的表示形式★★★★★	作为每个并发进程的终端语句 JOIN 的形式为 JOIN n，其中 n 为并发进程的个数。JOIN 语句附有一个计数器，其初始值为 0。每当执行 JOIN n 语句时，计数器的值加 1，并与 n 比较。若比较相等，表明这是执行中的第 n 个并发进程经过 JOIN 语句，于是允许该进程通过 JOIN 语句，将计数器清 0,并在其处理机上继续执行后续语句；若比较不等，计数器的值仍小于 n，表明此进程不是并发进程的最后一个，可让现在执行 JOIN 语句的这个进程先结束，把它所占用的处理机释放出来，分配给正在排队等待其他任务。

7.4 多处理机的操作系统

多处理机操作系统有 3 类。它们是**主从型、各自独立型及浮动型。★★**

主从型操作系统★★★	（1）优点：①结构比较简单；②整个管理程序只在一个处理机上运行，除非某些需递归调用或多重调用的公用程序，一般都不必是可再入的；③只有一个处理机访问执行表，不存在系统管理控制表格的访问冲突和阻塞，简化了管理控制的实现； （2）缺点：对主处理机的可靠性要求很高； （3）适用场合：适用于工作负荷固定，从处理机能力明显低于主处理机，或由功能相差很大的处理机组成的异构型多处理机。
各自独立型操作系统★★★	（1）含义：将控制功能分散给多台处理机，共同完成对整个系统的控制工作； （2）优点：①很适应分布处理的模块化结构特点，减少对大型控制专用处理机的需求；②某个处理机发生故障，不会引起整个系统瘫痪，有较高的可靠性；③每台处理机都有其专用控制表格，使访问系统表格的冲突较少，也不会有许多公用的执行表，同时控制进程和用户进程一起进行调度，能取得较高的系统效率； （3）缺点：实现复杂。 （4）适用场合：适用于松耦合多处理机。
浮动型操作系统	（1）含义：介于主从型和各自独立型操作系统之间的一种折中方式，其管理程 序可以在处理机之间浮动； （2）优点：各类资源可以较好地做到负荷平衡； （3）缺点：浮动型操作系统的设计最为困难； （4）适用场合：适用于紧耦合多处理机，特别是公共主存和 I/O 子系统的多个相同处理机组成 的同构型多处理机。

7.5 多处理机的发展

对称处理机	采用对称多处理机的典型计算机有 DEC Alpha Server 8400、SGI Power challenge、IBM R50、SUN Ultra Erterprise 1000 和我国的曙光一号等。

并行向量处理机★★	含义：并行向量处理机由若干数目不等的强功能的专用向量处理器经高带宽的纵横交叉开关 互连到若干共享的存储器模块，每个处理机系统超过1GFLOPS，这类机器一般不使用 Cache,而采用大量向量寄存器和指令缓冲存储器。
机群系统★★★	（1）含义：将多个高性能的工作站或高档微型计算机，使用高速的通信网络加以互连组成的系统； （2）优点：①系统有高的性能价格比；②系统的开发周期短；③系统的可扩展性好；④系统的资源利用率高；⑤用户投资风险小；⑥用户编程方便。

第八章 数据流计算机和归约机

8.1 数据流计算机

控制驱动的控制流方式的特点★★★	（1）通过访问共享存储单元让数据在指令之间传递；（2）指令执行的顺序性隐含于控制流中，但却可以显式地使用专门的控制操作符来实现并行处理；（3）指令执行的顺序受程序计数器控制，换句话说，是受控制令牌所支配的。
数据驱动计算★★	数据驱动计算，其操作是按输入数据**可用性**决定的次序进行的。需求驱动计算，其操作则按数据**需求**所决定的次序进行。
数据流★★	从语义上讲，数据流是基于**异步性**和**函数性**的一种计算模型。
单赋值语言具备的基本特点	（1）遵循单赋值规则；（2）有丰富的数据类型；（3）具有很强的类型性；（4）具有模块化结构的程序设计思想；（5）没有全局存储器和状态的概念；（6）程序不规定语句的执行顺序.
数据流计算机存在的问题	（1）数据流计算机的主要目的是为了提高操作级并行的开发水平，但如果题目本身数据相关性很强，内涵并行性成分不多时，就会使效率反而比传统的 Von Neumann 型机的还要低； （2）在数据流计算机中为给数据建立、识别、处理标记，需要花费较多的辅助开销和较大的存储空间（可能比 Von Neumann 型的要大出 2∽3 倍） （3）数据流计算机不保存数组； （4）数据流语言的变量代表数值，而不是存储单元位置，使程序员无法控制存储分配。 （5）数据流计算机互连网络设计困难，输入/输出系统仍不够完善。 （6）数据流计算机没有程序计数器，给诊断和维护带来了困难。

8.2 归约机

特点：（1）归约机应当是面向函数式语言，或以函数式语言为机器语言的非 Neumann 型机器，其内部结构应不同于 Neumann 型机器；
（2）具有大容量物理存储器并采用大虚存容量的虚拟存储器，具备高效的动态存储分配和管理的软、硬件支持，满足归约机对动态存储分配及所需存储空间大的要求；
（3）处理部分应当是一种有多个处理器或多个处理机并行的结构形式，以发挥函数式程序并行处理的特长；
（4）采用适合于函数式程序运行的多处理器（机）互连的结构，最好采用树形方式的互连结构或多层次复合的互连结构形式；

（5）为减少进程调度及进程间的通信开销，尽量把运行进程的结点机紧靠该进程所需用的数据安排，并使运行时需相互通信的进程所占用的处理机也靠近，让各处理机的负荷平衡；