计算机系统结构习题答案

目录

[第一章 计算机系统结构的基础知识 2](#_Toc484597127)

[第二章 指令系统的设计 7](#_Toc484597128)

[第三章 流水线技术 12](#_Toc484597129)

[第四章 向量处理机 22](#_Toc484597130)

[第五章 指令级并行及其开发——硬件方法 25](#_Toc484597131)

[第九章 互联网络 31](#_Toc484597132)

[第十章 多处理机 37](#_Toc484597133)

# 第一章 计算机系统结构的基础知识

多级层次结构：按照计算机语言从低级到高级的次序，把计算机系统按功能划分成多级层次结构，每一层以一种不同的语言为特征。这些层次依次为：微程序机器级，传统机器语言机器级，汇编语言机器级，高级语言机器级，应用语言机器级等。

虚拟机：用软件实现的机器。

计算机系统结构：传统机器程序员所看到的计算机属性，即概念性结构与功能特性。

透明性：在计算机技术中，把这种本来存在的事物或属性，但从某种角度看又好像不存在的概念称为透明性。

翻译：先用转换程序把高一级机器上的程序转换为低一级机器上等效的程序，然后再在这低一级机器上运行，实现程序的功能。

解释：对于高一级机器上的程序中的每一条语句或指令，都是转去执行低一级机器上的一段等效程序。执行完后，再去高一级机器取下一条语句或指令，再进行解释执行，如此反复，直到解释执行完整个程序。

摩尔定律：集成电路芯片上所集成的晶体管数目每隔18个月就翻一番。这使得越来越多的功能可以在一块芯片上实现，而且芯片的性能/价格比也越来越高。

系列机：由同一厂家生产的具有相同系统结构、但具有不同组成和实现的一系列不同型号的计算机。

Amdahl定律：当对一个系统中的某个部件进行改进后，所能获得的整个系统性能的提高，受限于该部件的执行时间占总执行时间的百分比。

程序的局部性原理：程序执行时所访问的存储器地址不是随机分布的，而是相对地簇聚。包括时间局部性和空间局部性。

软件兼容：一个软件可以不经修改或者只需少量修改就可以由一台计算机移植到另一台计算机上运行。差别只是执行时间的不同。

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。它包括同时性与并发性两种含义。

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，以加快硬件周转而赢得速度。

资源重复：在并行性概念中引入空间因素，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。

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。

耦合度：反映多机系统中各计算机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：又称直接耦合系统。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：又称间接耦合系统，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）。计算机之间的相互作用是在文件或数据集一级上进行。

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。

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中能并行执行的多个任务。

1.2

答：

如在设计主存系统时，确定主存容量、编址方式、寻址范围等属于计算机系统结构。

确定主存周期、逻辑上是否采用并行主存、逻辑设计等属于计算机组成。选择存储芯片类型、微组装技术、线路设计等属于计算机实现。

计算机组成是计算机系统结构的逻辑实现。计算机实现是计算机组成的物理实现。一种体系结构可以有多种组成。一种组成可以有多种实现。

1.3

答：

常见的计算机系统结构分类法有三种：Flynn分类法、冯氏分类法和Handler分类法。

Flynn分类法是按照指令流和数据流的多倍性进行分类的，分为以下四类：

1. 单指令流单数据流（SISD）；
2. 单指令流多数据流（SIMD）；
3. 多指令流但数据流（MISD）；
4. 多指令流多数据流（MIMD）；

冯氏分类发是用系统的最大并行度对计算机进行分类的，分为以下四类：

1. 字串位串：n=1，m=1。这是第一代计算机发展初期的纯串行计算机。
2. 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。
3. 字并位串：n=1，m>1。同时处理多个字的同一位（位片）。
4. 字并位并：n>1，m>1。同时处理多个字的多个位。

Handler分类法是以计算机的硬件结构进行分类的，分为三个层次，并考虑它们的可并行-流水处理程度。三个层次分别为;

1. 程序控制部件（PCU）的个数k；
2. 算数逻辑部件（ALU）或处理部件（PE）的个数d；
3. 每个算术逻辑部件包含基本逻辑线路（ELC）的套数w。

这样就可以把一个计算机系统的结构用以下公式表示：

t（系统型号）=(k,d,w)

1.4

答：

这里的“中间”是指层次结构中的软硬件的交界面，目前一般是在传统机器语言机器级与操作系统机器级之间。采用这种方法，首先要进行软硬件功能分配，确定好这个界面。然后从这个界面开始软件设计者开始往上设计操作系统、汇编、编译系统等，硬件设计者开始往下设计传统机器级、微程序机器级等。这种设计可以解决软硬件设计分离和脱节的问题，可以缩短设计周期，设计过程中还可以交流协调，是一种交互式的、很好的设计方法。

1.5

答：

实现软件可移植性的常用方法有三种：

统一高级语言：由于高级语言是面向问题和算法的，与计算机的具体结构关系不大，如果各计算机能采用同一种高级语言，那么用这种语言编写的应用软件和系统软件（或部分系统软件）的可移植问题就解决了。

采用系列机：系列机是指由同一厂家生产的具有相同的系统结构，但具有不同组成和实现的一系列不同型号的计算机。系列机的关键是要先对软/硬件分工进行充分的考虑，先设计好一种系统结构，之后软件设计者按此设计软件，硬件设计者则根据速度、性能、价格等不同的要求，选择不同的器件，采用不同的硬件技术和组成、实现技术，研制并提供不同档次的计算机。系列机能较好地解决了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。

模拟和仿真：为了使软件能在具有不同系统结构的计算机之间互相移植，可以通过在一种系统结构上实现另一种系统结构来实现。模拟是指用软件的方法在一台现有的计算机上实现另一台计算机的指令系统。仿真是指用一台现有的计算机上的微程序去解释实现另一台计算机的失灵系统。这个微程序是专门为实现目标机指令系统而设计的，称为仿真微程序。

1.6

答：

从处理数据的角度来看，并行性等级从低到高可分为：

（1）字串位串：每次只对一个字的一位进行处理。这是最基本的串行处理方式，不存在并行性；

（2）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。已开始出现并行性；

（3）字并位串：同时对许多字的同一位（称为位片）进行处理。这种方式具有较高的并行性；

（4）全并行：同时对许多字的全部位或部分位进行处理。这是最高一级的并行。

从执行程序的角度来看，并行性等级从低到高可分为：

（1）指令内部并行：单条指令中各微操作之间的并行；

（2）指令级并行：并行执行两条或两条以上的指令；

（3）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；

（4）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；

（5）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。

1.7

解：

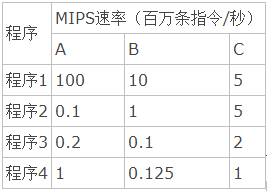
（1）CPI ＝(45000×1＋75000×2＋8000×4＋1500×2) / 129500＝1.776

（2）MIPS速率＝f/ CPI ＝400/1.776 ＝225.225MIPS

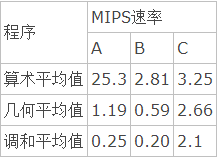
（3）程序执行时间= (45000×1＋75000×2＋8000×4＋1500×2)／400=575s

1.8

解：容易得到：



所以由算术平均值、几何平均值以及调和平均值的公式有：



1.9

解：

由题可知： 可改进比例 = 40% = 0.4 部件加速比 = 20

根据Amdahl定律可知：

系统加速比= = 1.6129

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1.6129倍。

1.10

解：

在多个部件可改进情况下，Amdahl定理的扩展：



已知S1＝30，S2＝20，S3＝10，Sn＝10，F1＝0.3，F2＝0.3，得：



得F3＝0.36，即部件3的可改进比例为36%。

（2）设系统改进前的执行时间为T，则3个部件改进前的执行时间为：（0.3+0.3+0.2）T = 0.8T，不可改进部分的执行时间为0.2T。

已知3个部件改进后的加速比分别为S1＝30，S2＝20，S3＝10，因此3个部件改进后的执行时间为：



改进后整个系统的执行时间为：Tn = 0.045T+0.2T = 0.245T

那么系统中不可改进部分的执行时间在总执行时间中占的比例是：



1.11

解：

改进之前，系统指令平均始终周期CPI为CPI =∑(CPIi(Ii / IC))= (530%)+(1.2570%) = 2.375.

如果采用A方案：FPSQR操作的CPI减至3，则整个系统的平均时钟周期数为：

CPIA = CPI - (CPIFPSQR - CPI'FPSQR)4% = 2.375 - (20 - 3) 4% = 1.695

如果采用B方案：把所有的FP操作的CPI减至3，则整个系统的平均时钟周期数为：

CPIB = CPI - (CPIFP - CPI'FP) 4% = 2.375 - (5-3) 30% = 1.775

从降低整个系统的指令平均时钟周期数的程度来看，方案A要优于B。

另外，分别计算两种方案的加速比：（SA=改进前CPU的执行时间/A的CPU执行时间=(IC时钟周期CPI)/(IC时钟周期CPIA)=CPI/CPIA）

SA=2.375/1.695=1.4

SB=2.375/1.775=1.34

由此也可知，方案A优于方案B。

# 第二章 指令系统的设计

2.1

堆栈型机器：CPU 中存储操作数的单元是堆栈的机器。

累加器型机器：CPU 中存储操作数的单元是累加器的机器。

通用寄存器型机器：CPU 中存储操作数的单元是通用寄存器的机器。

CISC：复杂指令集计算机

RISC：精简指令集计算机

寻址方式：指令系统中如何形成所要访问的数据的地址。一般来说，寻址方式可以指明指令中的操作数是一个常数、一个寄存器操作数或者是一个存储器操作数。

数据表示：硬件结构能够识别、指令系统可以直接调用的那些数据结构。

2.2

答：

区别不同指令集结构的主要因素是CPU中用来存储操作数的存储单元。据此可将指令系统结构分为堆栈结构、累加器结构和通用寄存器结构。

2.3

答：

|  |  |  |
| --- | --- | --- |
| 指令系统结构类型 | 优 点 | 缺 点 |
| 寄存器-寄存器型  （0，3） | 指令字长固定，指令结构简洁，是一种简单的代码生成模型，各种指令的执行时钟周期数相近。 | 与指令中含存储器操作数的指令系统结构相比，指令条数多，目标代码不够紧凑，因而程序占用的空间比较大。 |
| 寄存器-存储器型  （1，2） | 可以在ALU指令中直接对存储器操作数进行引用，而不必先用load指令进行加载。容易对指令进行编码，目标代码比较紧凑。 | 由于有一个操作数的内容将被破坏，所以指令中的两个操作数不对称。在一条指令中同时对寄存器操作数和存储器操作数进行编码，有可能限制指令所能够表示的寄存器个数。指令的执行时钟周期数因操作数的来源（寄存器或存储器）不同而差别比较大。 |
| 存储器-存储器型  （2，2）或（3，3） | 目标代码最紧凑，不需要设置寄存器来保存变量。 | 指令字长变化很大，特别是3操作数指令。而且每条指令完成的工作也差别很大。对存储器的频繁访问会使存储器成为瓶颈。这种类型的指令系统现在已不用了。 |

2.4

答：

对指令集的基本要求是：完整性、规整性、高效率和兼容性。

完整性是指在一个有限可用的存储空间内，对于任何可解的问题，编制计算程序时，指令集所提供的指令足够使用。

规整性主要包括对称性和均匀性。对称性是指所有与指令集有关的存储单元的使用、操作码的设置等都是对称的。均匀性是指对于各种不同的操作数类型、字长、操作种类和数据存储单元，指令的设置都要同等对待。

高效率是指指令的执行速度快、使用频度高。

2.5

答：

指令集功能设计：

（1）主要有RISC和CISC两种技术发展方向；

（2）寻址方式的设计：设置寻址方式可以通过对基准程序进行测试统计，察看各种寻址方式的使用频率，根据适用频率设置必要的寻址方式。

（3）操作数表示和操作数类型：主要的操作数类型和操作数表示的选择有：浮点数据类型、整型数据类型、字符型、十进制数据类型等等。

（4）寻址方式的表示：可以将寻址方式编码于操作码中，也可以将寻址方式作为一个单独的域来表示。

（5）指令集格式的设计：有变长编码格式、固定长度编码格式和混合型编码格式3种。

2.6

答：

主要目标是增强指令功能，把越来越多的功能交由硬件来实现，并且指令的数量也是越来越多。

缺点：

（1）CISC结构的指令集中，各种指令的使用频率相差悬殊。

（2）CISC结构指令的复杂性带来了计算机体系结构的复杂性，这不仅增加了研制时间和成本，而且还容易造成设计错误。

（3）CISC结构指令集的复杂性给VLSI设计增加了很大负担，不利于单片集成。

（4）CISC结构的指令集中，许多复杂指令需要很复杂的操作，因而运行速度慢。

（5）在CISC结构的指令集中，由于各条指令的功能不均衡性，不利于采用先进的计算机体系结构技术（如流水技术）来提高系统的性能。

2.7

答：

（1） 选取使用频率最高的指令，并补充一些最有用的指令；

（2）每条指令的功能应尽可能简单，并在一个机器周期内完成；

（3）所有指令长度均相同；

（4）只有Load和Store操作指令才访问存储器，其它指令操作均在寄存器之间进行； （5）以简单有效的方式支持高级语言。

2.8

答：

操作数类型有两种表示方法：

（1）操作数的类型由操作码的编码指定，这是最常见的一种方法；

（2）数据可以附上由硬件解释的标记，由这些标记指定操作数的类型，从而选择适当的运算。

2.9

答：

表示寻址方式有两种常用的方法：

（1）将寻址方式编于操作码中，由操作码在描述指令的同时也描述了相应的寻址方式。这种方式译码快，但操作码和寻址方式的结合不仅增加了指令的条数，导致了指令的多样性，而且增加了CPU对指令译码的难度。

（2）为每个操作数设置一个地址描述符，由该地址描述符表示相应操作数的寻址方式。这种方式译码较慢，但操作码和寻址独立，易于指令扩展。

2.10

答：

（1）变长编码格式。如果系统结构设计者感兴趣的是程序的目标代码大小，而不是性能，就可以采用变长编码格式。

（2）固定长度编码格式。如果感兴趣的是性能，而不是程序的目标代码大小，则可以选择固定长度编码格式。

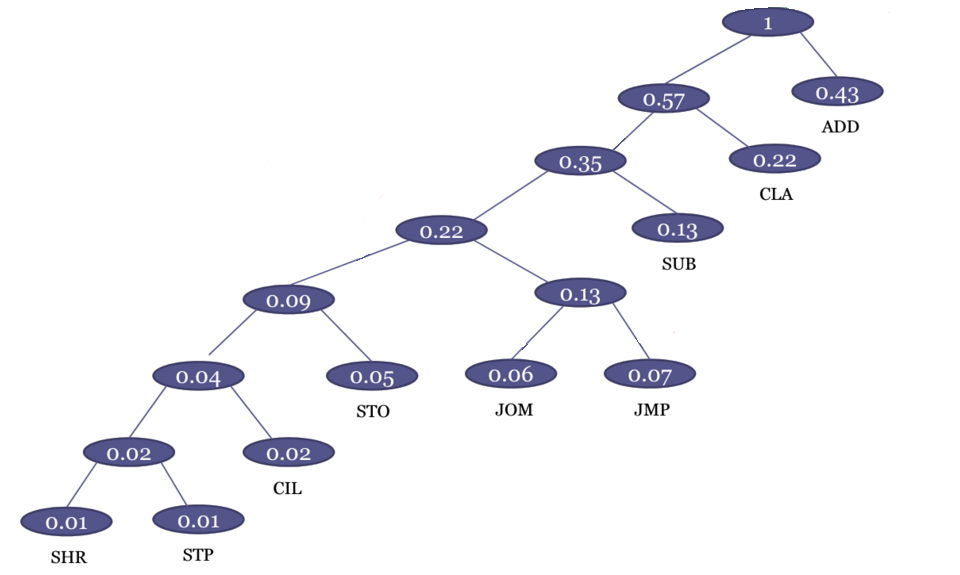
（3）混合型编码格式。需要兼顾降低目标代码长度和降低译码复杂度时，可以采用混合型编码格式。

2.11

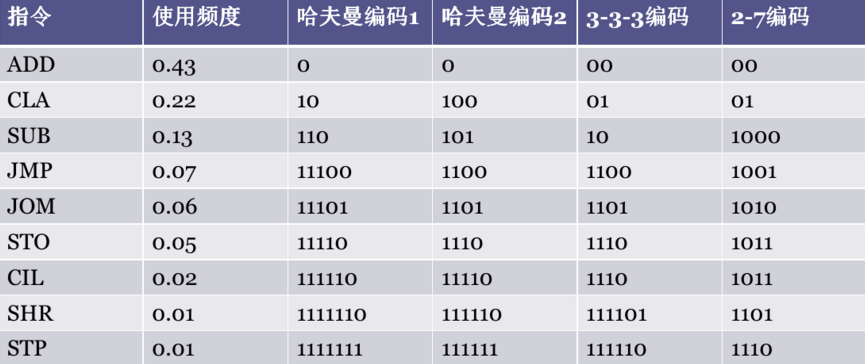
解：

特别注意哈夫曼编码树：

1. 最小概率合并
2. 所有指令在叶子结点上
3. 编码不唯一
4. 平均码长唯一



则编码结果如下表：



哈夫曼编码的平均码长为：2.42位

3-3-3扩展编码的平均码长为：2.52位

2-7扩展编码的平均码长为：2.70位

2.12

解：

双地址指令的结构为（4位操作码）+（6位地址码A1）+（6位地址码A2）

单地址指令的结构为（10位操作码）+（6位地址码A）

若只作为双地址指令的话，一共有=16条指令

每减少一条双地址指令将其分配给单地址指令，则可以增加条指令，所以此时双地址指令有A条，则单地址指令最多可以有(16 - A)条。

2.13

解：

指令字长为12位，每隔地址码长度为3位；

则三地址最多可以为=8条，此时三地址指令为4条，则可以将剩余8 – 4 = 4条用于扩展单地址指令；

所以此情况下的单地址最多为4=256条；

而单地址指令为255条，则剩余256 – 255 = 1条用于扩展零地址指令；

则零地址指令最多为1=8条；

8<16，所以不能扩展编码为其操作码编码。

如果要求单地址指令为254条，则剩余256 – 254 = 2条用于扩展零地址指令；

则零地址指令最多可以为2=16 条，刚好合适；

所以此时可以对其操作码扩展编码。

# 第三章 流水线技术

3.1

流水线技术：将一个重复的时序过程，分解成为若干个子过程，而每一个子过程都可有效地在其专用功能段上与其它子过程同时执行。

通过时间：流水线中第一个任务从进入流水线到流出结果的那个时间段。

排空时间：流水线中最后一个任务从进入流水线到流出结果的那个时间段。

定向技术：用来解决写后读冲突的。在发生写后读相关的情况下，在计算结果尚未出来之前，后面等待使用该结果的指令并不见得是马上就要用该结果。如果能够将该计算结果从其产生的地方直接送到其它指令需要它的地方，那么就可以避免停顿。

部件级流水线：把处理机中的部件进行分段，再把这些部件分段相互连接而成。它使得运算操作能够按流水方式进行。这种流水线也称为运算操作流水线。

指令流水线：为提高处理器执行指令的效率，把一条指令的操作分成多个细小的步骤，每个步骤由专门的电路完成的方式。

系统级流水线：又称为宏流水线。它是把多个处理机串行连接起来，对同一数据流进行处理，每个处理机完成整个任务中的一部分。前一个处理机的输出结果存入存储器中，作为后一个处理机的输入。

单功能流水线：指流水线的各段之间的连接固定不变、只能完成一种固定功能的流水线。

多功能流水线：指各段可以进行不同的连接，以实现不同的功能的流水线。

静态流水线：指在同一时间内，多功能流水线中的各段只能按同一种功能的连接方式工作的流水线。当流水线要切换到另一种功能时，必须等前面的任务都流出流水线之后，才能改变连接。

动态流水线：指在同一时间内，多功能流水线中的各段可以按照不同的方式连接，同时执行多种功能的流水线。它允许在某些段正在实现某种运算时，另一些段却在实现另一种运算。

线性流水线：指各段串行连接、没有反馈回路的流水线。数据通过流水线中的各段时，每一个段最多只流过一次。

非线性流水线：指各段除了有串行的连接外，还有反馈回路的流水线。

顺序流水线：流水线输出端任务流出的顺序与输入端任务流入的顺序完全相同。

乱序流水线：流水线输出端任务流出的顺序与输入端任务流入的顺序可以不同，允许后进入流水线的任务先完成。这种流水线又称为无序流水线、错序流水线、异步流水线。

吞吐率：在单位时间内流水线所完成的任务数量或输出结果的数量。

流水线的加速比：使用顺序处理方式处理一批任务所用的时间与按流水处理方式处理同一批任务所用的时间之比。

流水线的效率：即流水线设备的利用率，它是指流水线中的设备实际使用时间与整个运行时间的比值。

相关：是指两条指令之间存在某种依赖关系。

数据相关**：**考虑两条指令*i*和*j*，*i*在*j*的前面，如果下述条件之一成立，则称指令*j*与指令*i*数据相关：

（1）指令*j*使用指令*i*产生的结果；

（2）指令*j*与指令*k*数据相关，而指令*k*又与指令*i*数据相关。

名相关：如果两条指令使用了相同的名，但是它们之间并没有数据流动，则称这两条指令存在名相关。

反相关：考虑两条指令*i*和*j*，*i*在*j*的前面，如果指令*j*所写的名与指令*i*所读的名相同，则称指令*i*和*j*发生了反相关。

输出相关：考虑两条指令*i*和*j*，*i*在*j*的前面，如果指令*j*和指令*i*所写的名相同，则称指令*i*和*j*发生了输出相关。

换名技术：名相关的两条指令之间并没有数据的传送，只是使用了相同的名。可以把其中一条指令所使用的名换成别的，以此来消除名相关。

控制相关：是指由分支指令引起的相关。它需要根据分支指令的执行结果来确定后面该执行哪个分支上的指令。

流水线冲突：是指对于具体的流水线来说，由于相关的存在，使得指令流中的下一条指令不能在指定的时钟周期开始执行。

结构冲突：因硬件资源满足不了指令重叠执行的要求而发生的冲突。

数据冲突：当指令在流水线中重叠执行时，因需要用到前面指令的执行结果而发生的冲突。

控制冲突：流水线遇到分支指令或其它会改变PC值的指令所引起的冲突。

写后读冲突：考虑两条指令i和j，且i在j之前进入流水线，指令j用到指令i的计算结果，而且在i将结果写入寄存器之前就去读该寄存器，因而得到的是旧值。

写后写冲突：考虑两条指令i和j，且i在j之前进入流水线，，指令j和指令i的结果单元（寄存器或存储器单元）相同，而且j在i写入之前就先对该单元进行了写入操作，从而导致写入顺序错误。这时在结果单元中留下的是i写入的值，而不是j写入的。

读后写冲突：考虑两条指令i和j，且i在j之前进入流水线，指令j的目的寄存器和指令i的源操作数寄存器相同，而且j在i读取该寄存器之前就先对它进行了写操作，导致i读到的值是错误的。

3.2

答：

流水技术有以下特点：

（1） 流水线把一个处理过程分解为若干个子过程，每个子过程由一个专门的功能部件来实现。因此，流水线实际上是把一个大的处理功能部件分解为多个独立的功能部件，并依靠它们的并行工作来提高吞吐率。

（2） 流水线中各段的时间应尽可能相等，否则将引起流水线堵塞和断流。

（3） 流水线每一个功能部件的前面都要有一个缓冲寄存器，称为流水寄存器。

（4） 流水技术适合于大量重复的时序过程，只有在输入端不断地提供任务，才能充分发挥流水线的效率。

（5） 流水线需要有通过时间和排空时间。在这两个时间段中，流水线都不是满负荷工作。

3.3

答：

通过软件（编译器）来减少分支延迟的方法有：

1. 预测分支失败
2. 预测分支成功
3. 延迟分支

共同特点：它们对分支的处理方法在程序的执行过程中始终保持不变。它们要么总是预测分支成功，要么总是预测分支失败。

3.4

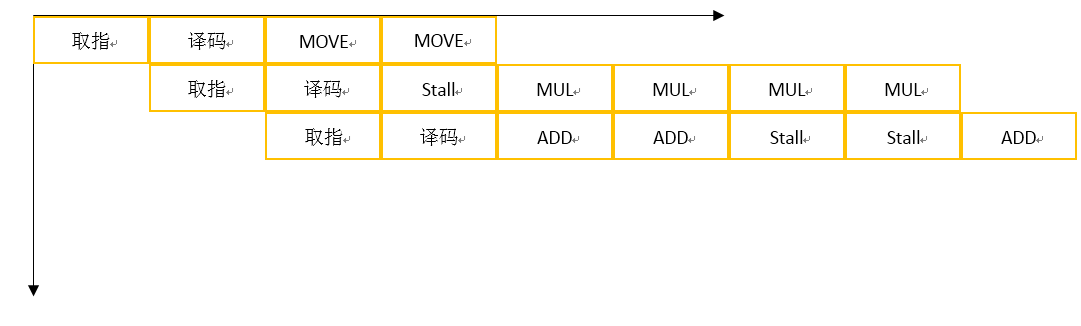
答：

|  |  |  |
| --- | --- | --- |
| 调度策略 | 对调度的要求 | 对流水线性能改善的影响 |
| 从前调度 | 分支必须不依赖于被调度的指令 | 总是可以有效提高流水线性能 |
| 从目标处调度 | 如果分支转移失败，必须保证被调度的指令对程序的执行没有影响，可能需要复制被调度指令 | 分支转移成功时，可以提高流水线性能。但由于复制指令，可能加大程序空间 |
| 从失败处调度 | 如果分支转移成功，必须保证被调度的指令对程序的执行没有影响 | 分支转移失败时，可以提高流水线性能 |

3.5

解：

1. 就程序本身而言，可能有三种数据相关。若3条指令顺序流动，则k指令对R1寄存器的写与k+1指令对R1寄存器的读形成的“先写后读”相关。若3条指令异步流动，则k指令对R0寄存器的读与k+1指令对R0寄存器的写形成的“先读后写”相关，k+2指令对R0寄存器的写与k+1指令对R0寄存器的写形成的“写—写”相关。
2. 在程序实际执行过程中，二种数据相关会引起流水线停顿。一是“先写后读”相关，k指令对R1的写在程序执行开始后的第四个时钟；k+1指令对R1的读对指令本身是第三个时钟，但k+1指令比k指令晚一个时钟进入流水线，则在程序执行开始后的第四个时钟要读R1.不能再同一时钟周期内读写同一寄存器，因此k+1指令应推迟一个时钟进入流水线，产生了流水线停顿。二是“写—写”相关，k+1指令对R0的写对指令本身是第六个时钟，而要求该指令进入流水线应在程序执行开始后的第三个时钟，所以对R0的写是在程序执行开始后的第八个时钟。k+2指令对R0的写对指令本身是第五个时钟，而k+2指令比k+1指令晚一个时钟进入流水线，则在程序执行开始后的第四个时钟，所以对R0的写是在程序执行开始后的第八个时钟。不能在同一时钟周期内写同一个寄存器，因此k+2指令应推迟一个时钟进入流水线，产生了流水线停顿。另外，可分析“先读后写”相关不会产生流水线的停顿。
3. 由题意可认为该指令流水线由六个功能段取指、译码、取数、运一、运二和存数等组成，则程序指令执行过程的流水线时空图如下所示。若3条指令顺序流动，共需要9个时钟周期。



3.6

解：

（1）







（2）瓶颈在3、4段。

* 变成八级流水线（细分）









* 重复设置部件

1

2

3-1

3-2

4-1

4-2

4-3

4-4







3.7

解：

（1）会发生流水线阻塞情况。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 第1个任务 | S1 | S2 | S3 | S3 | S4 |  |  |  |  |  |  |
| 第2个任务 |  | S1 | S2 | stall | S3 | S3 | S4 |  |  |  |  |
| 第3个任务 |  |  | S1 | stall | S2 | stall | S3 | S3 | S4 |  |  |
| 第4个任务 |  |  |  |  | S1 | stall | S2 | stall | S3 | S3 | S4 |

（2）





（3）重复设置部件



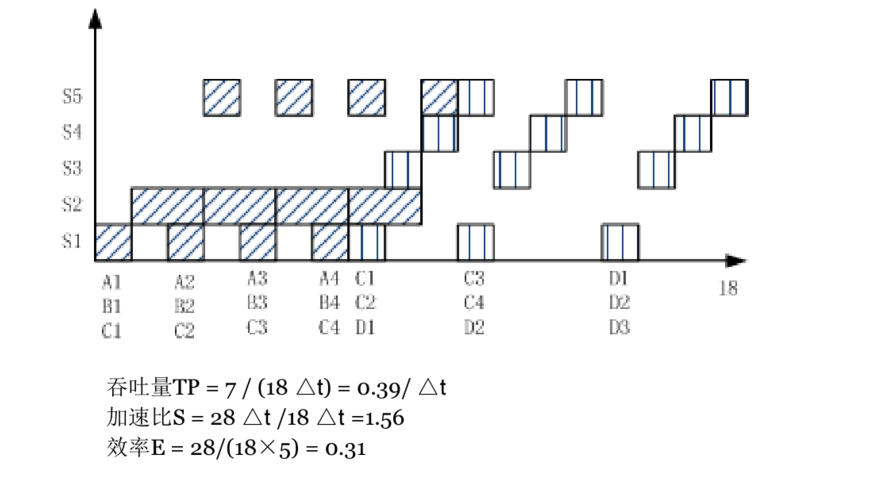




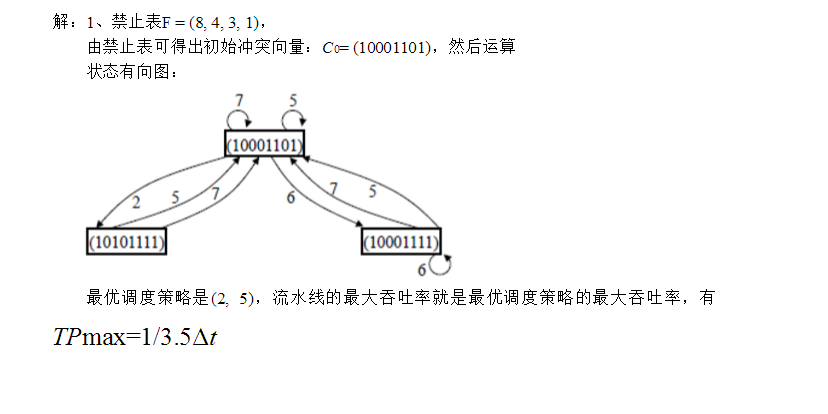
吞吐率提高倍数＝＝1.64

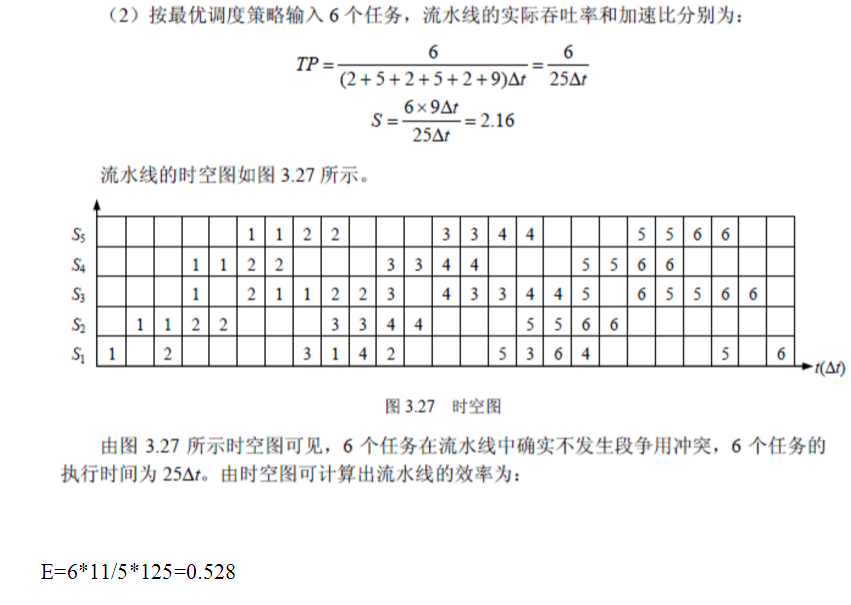
3.8

解：



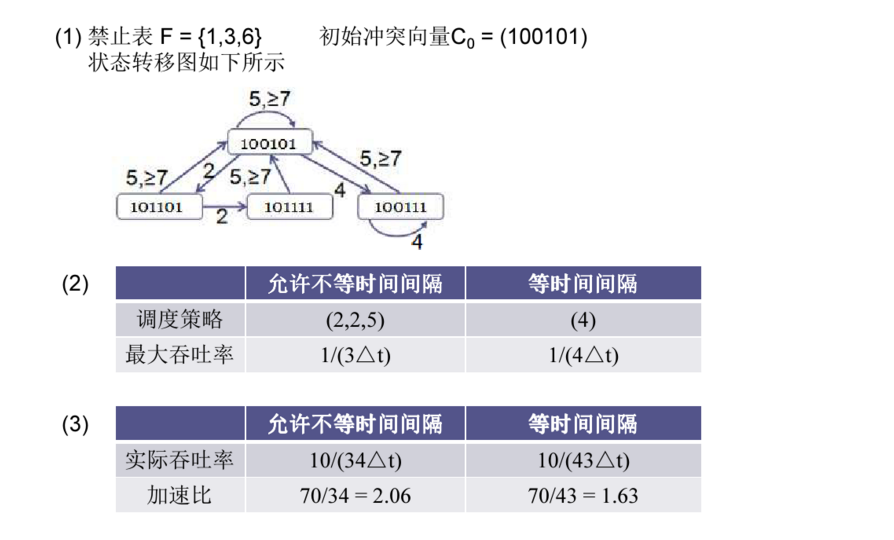
3.9





3.10

解：



3.11

解：

寄存器读写可以定向，无其他旁路硬件支持。排空流水线。



第i次迭代（i＝0..98）开始周期：1＋（i×17）

总的时钟周期数：（98×17）＋18＝1684

有正常定向路径，预测分支失败。



第i次迭代（i＝0..98）开始周期：1＋（i×10）

总的时钟周期数：（98×10）＋11＝991

有正常定向路径。单周期延迟分支。

LOOP: LW R1，0(R2)

DADDIU R2，R2，#4

DADDIU R1，R1，#1

DSUB R4，R3，R2

BNEZ R4，LOOP

SW R1，-4(R2)

第i次迭代（i ＝0..98）开始周期：1＋（i ×6 ）

总的时钟周期数：（98×6）＋10＝598



3.12

解：

没有控制相关时流水线的平均CPI＝1

存在控制相关时：由于无条件分支在第二个时钟周期结束时就被解析出来，而条件分支

要到第3个时钟周期结束时才能被解析出来。所以：

（1）若使用排空流水线的策略，则对于条件分支，有两个额外的stall，对无条件分支，有一个额外的stall：

CPI = 1+20%2+5%1 = 1.45

加速比S=CPI/1 = 1.45

（2） 若使用预测分支成功策略，则对于不成功的条件分支，有两个额外的stall，对无条件分支和成功的条件分支，有一个额外的stall 1：

CPI = 1+20%(60%1+40%2) +5%1 = 1.33

加速比S=CPI/1 = 1.33

（3）若使用预测分支失败策略，则对于成功的条件分支，有两个额外的stall；对无条件分支，有一个额外的stall；对不成功的条件分支，其目标地址已经由PC 值给出，不必等待，所以无延迟：

CPI = 1+20%(60%2 + 40%0) +5%1 = 1.29

加速比S=CPI/1 = 1.29

# 第四章 向量处理机

4.1

横向处理方式：按行的方式从左至右恒祥地进行计算。

纵向处理方式：向量计算按列的方式从上到下纵向的进行，将整个向量按相同的方式运算处理完毕之后，再去进行别的运算。

纵横处理方式：把向量分成若干组，组内按照纵向的方式处理，依次处理各组。

向量流水线链接：具有先写后读相关性的两条指令，在不出现功能部件冲突和其他Vi冲突的情况下，可以把功能部件连接起来流水处理，以加快执行。

Vi冲突：源寄存器冲突，结果寄存器冲突

功能部件冲突：两条指令使用相同的功能部件。

分段开采：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，然后循环分段处理，每一次循环只处理一个向量段。

半性能向量长度：向量处理机的性能为其最大性能 的一半时所需的向量长度。

向量长度临界值：向量流水方式的处理速度优于标量串行方式的处理速度时所需的向量长度的最小值。

4.2

答：

横向处理方式： 若向量长度为N，则水平处理方式相当于N次循环，若使用流水线，在每次循环中可能出现数据相关和功能转换，不适合对向量进行流水处理。

纵向处理方式：将整个向量按相同的运算处理完毕后，再去执行其他运算。适合对向量进行流水处理，向量运算指令的源/目的向量都放在存储器内，使得流水线运算部件的输入、输出端直接与存储器相连，构成M-M型运算流水线。

纵横处理方式：把长度为N的向量分为若干组，每组长度为N，组内按照纵向方式处理，依次处理各组，适合流水处理。可设长度为n的向量寄存器，使每组向量运算的源\目的向量都在向量寄存器中，流水线的运算部件输入、输出端与向量寄存器相连，构成R-R型运算流水线。

4.3

答：

设置多个功能部件，使它们并行工作；

采用链接技术，加快一串向量指令的执行；

采用循环开采技术，加快循环的处理；

采用多处理机系统，进一步提高性能。

4.4

解：

无Vi冲突和无功能部件冲突

只有前一条指令的第一个元素送入结果向量寄存器的哪一个时钟周期才可以进行链接

当一条向量指令的两个源操作数分别来自紧邻的两条指令结果时，要求这两条指令产生的运算结果的时间必须相等

要进行链接执行的向量指令的向量长度必须相等

4.5

解：

该链接流水线的通过时间：

1 + 7 + 1 + 1 + 1 + 3 + 1 + 1 + 4 + 1 + 1 + 2 = 23 (拍)

在向量长度为64的情况下，得到所有结果共需要 23 + 64 = 87 (拍)

4.6

解：

该链接流水线的通过时间：

1 +7 + 1 + 1 + 3 + 1 + 1 + 4 + 1 + 1 + 2 + 1 + 7 = 31 (拍)

在向量长度为64的情况下，得到所有结果共需要 31 + 64 – 1 = 94 (拍)

4.7

解：

（1）设（A+B）的中间结果放入V6，（A+B）×C结果放入V7，（D+E）中间结果放入V8，（D+E）×F放入V9中。

先计算（A+B）×C，再计算（D+E）×F，即可理解为同一条指令进两次流水。故，通过时间只需要考虑一者。

V6 ←V1+V2

V7←V6×V3

T = (1 + 2 + 1) + (1 + 3 + 1) = 9 (拍)

（2）



4.8

解：



本题中，n=200，Tloop=15，m=1，MVL=64

Tstart = e – 1 = 向量流水线通过时间 – 1 = 12 + 7 + 12 – 1 = 30

Tall = 380

4.9

解：



4.10

解：

通过时间： 2 + 4 + 1 = 7

向量长度64时，共需7 + 64 - 1 = 70（拍）

时间：70×10× =  s

# 第五章 指令级并行及其开发——硬件方法

5.1

指令级并行：简称ILP。是指指令之间存在的一种并行性，利用它，计算机可以并行执行两条或两条以上的指令。

IPC：指令调度。通过在编译时让编译器重新组织指令顺序或通过硬件在执行时调整指令顺序来消除冲突。

循环级并行性：是指处于不同循环层次的不同循环体之间的并行性。

指令的动态调度：是指在保持数据流和异常行为的情况下，通过硬件对指令执行顺序进行重新安排，以提高流水线的利用率且减少停顿现象。是由硬件在程序实际运行时实施的。

指令的静态调度：是指依靠编译器对代码进行静态调度，以减少相关和冲突。它不是在程序执行的过程中、而是在编译期间进行代码调度和优化的。

不精确异常：在多发射乱序执行的流水线 CPU 上，从指令进入流水线到异常事件的发生，期间要经过若干流水级，此时 PC 的值已指向其后的某条指令，在实现非精确异常的 CPU 上就把此时的 PC 值作为引起异常指令的所在。

精确异常：简单地说就是引起异常指令的指向就是真正引起异常的指令之所在。而实现精确异常的 CPU，则在最后指令提交时 (commit) 按指令流的顺序提交，异常的抛出也在该指令提交时，这样就能精确计算出引起异常的指令相对于当前 PC 的偏移，从而保证精确异常。

CDB：公共数据总线。

动态分支预测技术：是用硬件动态地进行分支处理的方法。在程序运行时，根据分支指令过去的表现来预测其将来的行为。如果分支行为发生了变化，预测结果也跟着改变。

BHT：分支历史表。用来记录相关分支指令最近一次或几次的执行情况是成功还是失败，并据此进行预测。

分支目标缓冲：是一种动态分支预测技术。将执行过的成功分支指令的地址以及预测的分支目标地址记录在一张硬件表中。在每次取指令的同时，用该指令的地址与表中所有项目的相应字段进行比较，以便尽早知道分支是否成功，尽早知道分支目标地址，达到减少分支开销的目的。

前瞻执行：解决控制相关的方法，它对分支指令的结果进行猜测，然后按这个猜测结果继续取指、流出和执行后续的指令。只是指令执行的结果不是写回到寄存器或存储器，而是放到一个称为ROB的缓冲器中。等到相应的指令得到“确认”（即确实是应该执行的）后，才将结果写入寄存器或存储器。

ROB：ReOrder Buffer。前瞻执行缓冲器。

超标量：一种多指令流出技术。它在每个时钟周期流出的指令条数不固定，依代码的具体情况而定，但有个上限。

超流水：在一个时钟周期内分时流出多条指令。

超长指令字：一种多指令流出技术。VLIW处理机在每个时钟周期流出的指令条数是固定的，这些指令构成一条长指令或者一个指令包，在这个指令包中，指令之间的并行性是通过指令显式地表示出来的。

5.2

答：

对于正确地执行程序来说，必须保持的最关键的两个属性是：数据流和异常行为。

保持异常行为是指：无论怎么改变指令的执行顺序，都不能改变程序中异常的发生情况。即原来程序中是怎么发生的，改变执行顺序后还是怎么发生。

数据流：指数据值从其产生者指令到其消费者指令的实际流动。

5.3

答：

记分牌中记录的信息由三部分构成。

1. 指令状态表：记录正在执行的各条指令已经进入到了哪一段。
2. 功能部件状态表：记录各个功能部件的状态。每个功能部件有一项，每一项由以下9个字段组成。

Busy：忙标志，指出功能部件是否忙。初值为no；

Op：该功能部件正在执行或将要执行的操作；

Fi：目的寄存器编号；

Fj，Fk：源寄存器编号；

Qj，Qk：指出向源寄存器Fj、Fk写数据的功能部件；

Rj，Rk：标志位，为yes表示Fj，Fk中的操作数就绪且还未被取走，否则就被置为no。

1. 结果寄存器状态表Result：每个寄存器在该表中有一项，用于指出哪个功能部件将把结果写入该寄存器。如果当前正在运行的指令都不以它为目的寄存器，则其相应项置为no。Result各项的初值为no（全零）。

5.4

答：

核心思想是：① 记录和检测指令相关，操作数一旦就绪就立即执行，把发生RAW冲突的可能性减小到最少；② 通过寄存器换名来消除WAR冲突和WAW冲突。寄存器换名是通过保留站来实现，它保存等待流出和正在流出指令所需要的操作数。

基本思想：只要操作数有效，就将其取到保留站，避免指令流出时才到寄存器中取数据，这就使得即将执行的指令从相应的保留站中取得操作数，而不是从寄存器中。指令的执行结果也是直接送到等待数据的其它保留站中去。因而，对于连续的寄存器写，只有最后一个才真正更新寄存器中的内容。一条指令流出时，存放操作数的寄存器名被换成为对应于该寄存器保留站的名称（编号）。

5.5

答：

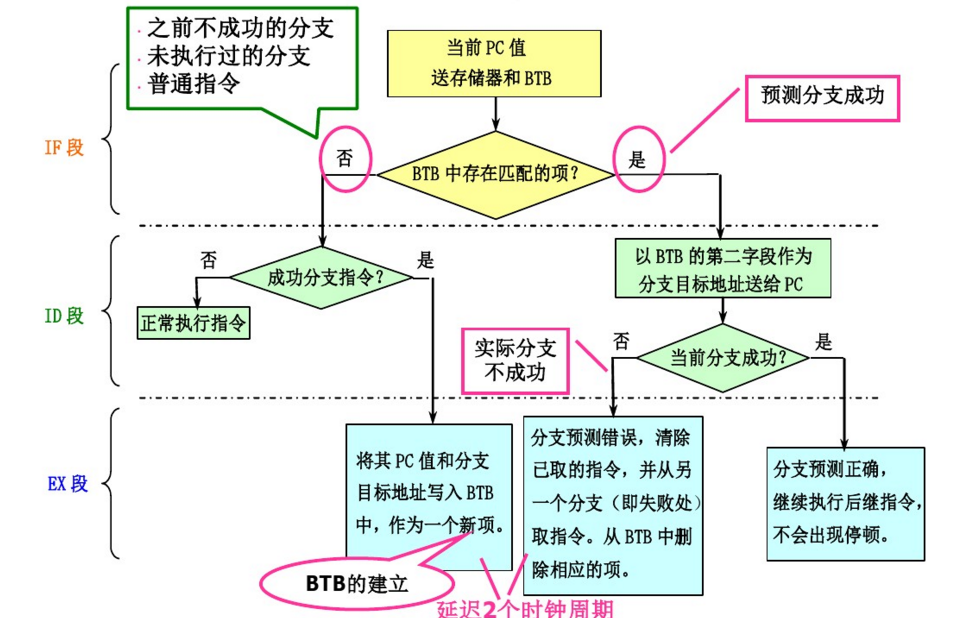
采用动态分支预测技术的目的由两个：1、预测分支是否成功。2、尽快找到分支目标地址（或者指令），从而避免控制相关造成流水线停顿。

需要解决以下两个关键问题：

1. 如何记录分支的历史信息，要记录哪些信息？
2. 如何根据这些信息来预测分支的去向，甚至提前取出分支目标处的指令？

5.6

答：



5.7

答：

基于硬件的前瞻执行是把三种思想结合在了一起。

1. 动态分支预测。用来选择后续执行的指令。
2. 在控制相关的结果尚未出来之前，前瞻地执行后续指令。
3. 用动态调度对基本块的各种组合进行跨基本块的调度。

5.8

解：

（1）程序执行的CPI = 没有分支的基本CPI（1） + 分支带来的额外开销

分支带来的额外开销是指在分支指令中，缓冲命中但预测错误带来的开销与缓冲没有命中带来的开销之和。

分支带来的额外开销= 15% \* (90%命中×10%预测错误×4 + 10％没命中×3)= 0.099

所以，程序执行的CPI ＝ 1 ＋ 0.099 = 1.099

（2）采用固定的2 个时钟周期延迟的分支处理CPI = 1 + 15%×2 = 1.3

由（1）（2）可知分支目标缓冲方法执行速度快。

5.9

解：

设每条无条件转移指令的延迟为x，则有：

1＋5%×x＝1.1

x＝2

当分支目标缓冲命中时，无条件转移指令的延迟为0。

所以 程序的CPI ＝ 1 ＋ 2 × 5% ×(1 －90%) ＝1.01

5.10

解：

（1）

|  |  |
| --- | --- |
| 第一路 | 第二路 |
| LW R4, (R5) |  |
| LW R7, (R8) |  |
|  |  |
|  |  |
| DADD R9, R4, R7 | LD R10, (R11) |
| DMUL R12, R13, R14 |  |
| DSUB R2, R3, R1 | SW R15, (R2) |
| DMUL R21, R4, R7 | SW R23, (R22) |
|  |  |
| SW R21, (R24) |  |

（2）

|  |  |
| --- | --- |
| 第一路 | 第二路 |
| LW R4, (R5) | LW R7, (R8) |
|  |  |
|  |  |
| DADD R9, R4, R7 | LD R10, (R11) |
| DMUL R12, R13, R14 | DSUB R2, R3, R1 |
| SW R15, (R2) | DMUL R21, R4, R7 |
| SW R23, (R22) |  |
| SW R21, (R24) |  |

（3）

|  |  |
| --- | --- |
| 第一路 | 第二路 |
| LW R4, (R5) | LW R7, (R8) |
| DSUB R2, R3, R1 | LD R10, (R11) |
| SW R23, (R22) | DMUL R12, R13, R14 |
| DADD R9, R4, R7 | DMUL R21, R4, R7 |
| SW R15, (R2) |  |
| SW R21, (R24) |  |

5.11

解：标量流水处理机的时空图：



执行完12条指令需T1＝14△t。

超标量流水处理机与超长指令字处理机的时空图：



超标量流水处理机中，每一个时钟周期同时启动4条指令。执行完12条指令需T2＝5△t，相对于标量流水处理机的加速比为：



超长指令字处理机中，每4条指令组成一条长指令，共形成3条长指令。执行完12条指令需T3＝5△t，相对于标量流水处理机的加速比为：



超流水处理机的时空图：



超流水处理机中，每1/4个时钟周期启动一条指令。执行完12条指令需T4＝5.75△t，相对于标量流水处理机的加速比为：



# 第九章 互联网络

9.1

互连网络：是一种由开关元件按照一定的拓扑结构和控制方法构成的网络，用来实现计算机系统中节点之间的相互连接。这些节点可以是处理器、存储模块或其他设备。在拓扑上，互连网络是输入节点到输出节点之间的一组互连或映像。

互连函数：互连网络是输入节点到输出节点之间的一组互连，这种互连可以用互连函数来表示，互连函数反映了网络输入端数组和输出端数组之间对应的置换关系或排列关系，所以互连函数有时也称为置换函数或排列函数。

网络规模：是指互连网络中节点的个数。它表示该网络所能连接的部件的数量。网络规模越大，则这个互连网络的连接能力越强，就能连接更多的部件。

节点度：是指互连网络中节点所连接的边数，包括入度和出度。进入节点的边数叫入度，从节点出来的边数叫出度。

节点距离：对于互连网络中的任意两个节点，其距离是指从一个节点出发到另一个节点终止所需要跨越的边数的最小值。

网络直径：是指互连网络中任意两个节点之间距离的最大值。网络中任意两个节点之间传送的信息通过的边数都不会大于网络直径。显然，网络直径应当尽可能的小。

等分带宽：与等分宽度对应的切平面中，所有边合起来单位时间所能传送的最大信息量称为该网络的等分带宽。

对称网络：如果从任意节点来看，网络的结构都是相同的，则称该网络为对称网络。

静态网络：是指各节点之间有固定的连接通路、且在运行中不能改变的网络。

动态网络：是指由交换开关构成、可按运行程序的要求动态改变连接状态的网络。

虚拟通道：是两个节点间的逻辑链接，它由源节点的片缓冲区、节点间的物理通道以及接收节点的片缓冲区组成。

自适应寻径：通信的通路每一次都要根据资源或者网络的情况来选择。这样就可以避开拥挤的或者有故障的节点，从而使网络的利用率得到改进。

确定寻径方法：通信路径完全由源节点地址和目的地址来决定，寻径路径是预先唯一地确定好了的，而与网络的状况无关。

线路交换：在传递一个信息之前，需要先建立一条从源节点到目的节点的物理通路，然后在传递信息。

存储转发：是最简单的分组交换方式。在这种方式中，包是信息传递的基本单位。存储转发要求所经过的每个中间节点都要设置一个包缓冲器，用于保存所传递的包。当一个包到达某个中间节点时，该节点先把这个包全部存储起来，然后在出口链路可用、而且下一个节点的包缓冲器也可用的情况下，传递给下一个节点。

虫蚀方式：把信息包“切割”称“片”，而且使信息包中各片的传送按流水方式进行，所以不仅可以减少节点中缓冲器的容量，而且还能缩短传送延迟时间。

9.2

解：

(0110) = 1110

(0110) = 0101

(0110) = 0110

(0110) = 1010

9.3

答：把第二个包暂存在缓冲区、阻塞第二个包、丢弃第二个包、绕道。

9.4

答：

时延和带宽是用来评估互连网络性能的两个基本指标。

1. 通信时延
2. 网络时延
3. 端口带宽
4. 聚集带宽
5. 等分带宽

9.5

答：

控制方式是指对各个开关模块进行控制的方式，它可以有三种。

1. 级控制：每一级的所有开关只用一个控制信号进行控制，这些开关只能同时处于一种状态；
2. 单元控制：每一个开关都有一个独立的控制信号，可各自处于不同的状态；
3. 部分级控制：第i级的所有开关分别用i+1个信号控制，0in-1，n为级数。

9.6

答：

不能实现含Cube1的配对通信，即0、1、4、5、8、9、C、D不能与2、3、6、7、A、B、E、F之间进行通信。因为Cube() 与 之间，第1级开关为“直连”，则为“0”的不能与为“1”的处理器号通信，即号为xx0x 的处理器不能与号 xx1x的处理器配对。

9.7

答：

总线互连的复杂性最低，成本也是最低。其缺点是每台处理机可用的带宽较窄。

交叉开关是最昂贵的，因为其硬件复杂性以上升，所以其成本最高。但是交叉开关的带宽和寻径性能最好。当网络的规模较小时，它是一种理想的选择。

多级互连网络的复杂度和带宽介于总线和交叉开关之间，是一种折中方案。其主要优点是采用模块化结构，可扩展性较好。不过，其时延随网络级数的增加而上升。另外，由于其硬件复杂度比总线高很多，其成本也不低。

9.8

答：

从0号到15号，需要经过4次交换，3次混洗。

9.9

解：

1. Cube2（12）＝Cube2（01100）＝（01000）2＝8，

（8）＝（01000）＝（10000）2＝16，

（9）＝（01001）＝（11000）2＝24，

PM2I+3（28）＝28+23 mod 32=4，

Cube0（（4））＝ Cube0（（00100））＝ Cube0（01000）＝

（01001）2＝9，

（Cube0（18））＝（Cube0（10010））＝（10011）＝（00111）2＝7.

1. 依题意，均匀洗牌交换网的互连函数设计为：S（E0（X））。

（Cube0（00000））＝00010，（Cube0（00001））＝00000，

（Cube 0（00010））＝00110，（Cube 0（00011））＝00100，

（Cube 0（00100））＝01010，（Cube 0（00101））＝01000，

（Cube 0（00110））＝01110，（Cube 0（00111））＝01100，

（Cube 0（01000））＝10010，（Cube 0（01001））＝10000，

（Cube 0（01010））＝10110，（Cube 0（01011））＝10100，

（Cube 0（01100））＝11010，（Cube 0（01101））＝11000，

（Cube 0（01110））＝11110，（Cube 0（01111））＝11100，

（Cube 0（10000））＝00011，（Cube 0（10001））＝00001，

（Cube 0（10010））＝00111，（Cube 0（10011））＝00101，

（Cube 0（10100））＝01011，（Cube 0（10101））＝01001，

（Cube 0（10110））＝01111，（Cube 0（10111））＝01101，

（Cube 0（11000））＝10011，（Cube 0（11001））＝10001，

（Cube 0（11010））＝10111，（Cube 0（11011））＝10101，

（Cube 0（11100））＝11011，（Cube 0（11101））＝11001，

（Cube 0（11110））＝11111，（Cube 0（11111））＝11101。

以上互连关系用图形描述如下：

0→2→6→14→30→31→29→25→17→1

3→4→10→22→15→28→27→21→9→16

5→8→18→7→12→26→23→13→24→19

11→20

所以有：网络直径=∞；从5号结点发送数据到7号结点，最短路径要经过3步，即5→8→18→7。

⑶ 采用移数函数构成互连网，结点间的互连关系是：

PM2+0：（0 1 2 3 … 29 30 31）

PM2-0：（31 30 29 28 … 2 1 0）

PM2+1：（0 2 4 6 … 28 30） （1 3 5 7 … 29 31）

PM2-1：（30 28 26 … 4 2 0） （31 29 27 … 5 3 1）

PM2+2：（0 4 8 12 16 20 24 28）（1 5 9 13 17 21 25 29）

1. 6 10 14 18 22 26 30）（3 7 11 15 19 23 27 31）

PM2-2：（28 24 20 16 12 8 4 0）（29 25 21 17 13 9 5 1）

1. 26 22 18 14 10 6 2）（31 27 23 19 15 11 7 3）

PM2+3：（0 8 16 24）（1 9 17 25）（2 10 18 26）（3 11 19 27）

1. 12 20 28）（5 13 21 29）（6 14 22 30）（7 15 23 31）

PM2-3：（24 16 8 0）（25 17 9 1）（26 18 10 2）（27 19 11 3）

1. 20 12 3）（29 21 13 5）（30 22 14 6）（31 23 15 7）

PM2±4：（0 16）（1 17）（2 18）（3 19）（4 20）（5 21）（6 22）（7 23）

1. 24）（9 25）（10 26）（11 27）（12 28）（13 29）（14 30）
2. 31）

9.10

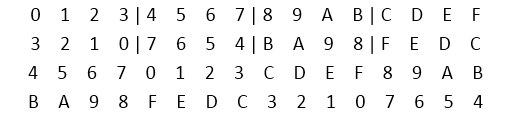
解：

不能用循环表示法表示该互联网络实现的互连。

9.11

解：

交换情况为下图：



则可知（0 B）（1 A）（2 9）（3 8）（4 F）（5 E）（6 D）（7 C）为输入端输出端组合

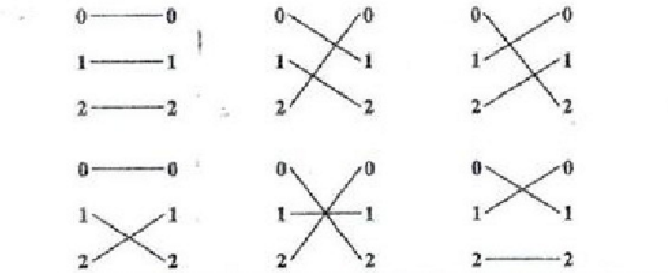
则互连函数为：

9.12

解：

1. 在不考虑网络开关状态冲突和开关输出端争用的情况下，互连挽留过的N个输入端与输出端之间所有可能的置换连接的数量就是输出端端号的不同排列的排列数，故N个输出端总共有N！种不同的排序。

当N=3时，输入端和输出端的置换连接有3！=6种，如下图所示：



与输入端端号序列012一一对应置换连接的3个输出端端号的6种排序分别是012、120、201、021、210和102。

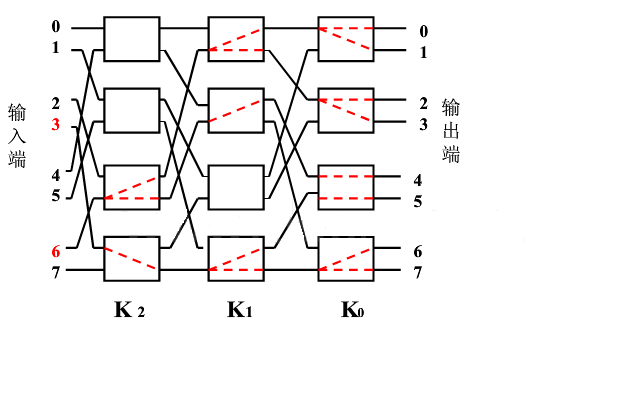
1. N个输入端、输出端的omega网络有n=级开关机，每级改观级有N/2个22的4功能开关，总共有(N/2)个开关。置换连接是指网络的输入端与输出端的一对一连接，故只考虑22开关的2个功能状态，即直连与交换。网络采用单元控制，因此，每个开关都根据连接要求处于2个功能状态中的一种状态，所以，由(N/2)个开关组成omega网络的开关状态的种数为：。

一种网络开关状态实现omega网络的一种无冲突的置换连接，所以，一次使用omega网络可以实现的N个输入端与N个输出端的无冲突的置换连接有种。

1. 若N=8，则omega网络一次使用能实现的置换连接数占全部可能的置换连接数的比例为：

9.13

解：



由图可知，它们的播送要求没有冲突，因此可以同时实现播送。

# 第十章 多处理机

10.1

集中式共享多处理机：最多由几十个处理器构成。由于处理器个数较少，各处理器可以共享一个集中式的物理存储器。

分布式共享多处理机：在这类机器中，存储器在物理上是分布的。它支持构建规模较大的多处理机系统，存储器分布在各个处理器上。

SMP：对称共享存储多处理机，只有单一主存，并且这个主存对于各个处理器的关系是对称的

多Cache一致性：如果允许共享数据进入Cache，就可能出现多个处理器的Cache中都有一个同意存储块的副本的情况，当其中某个处理器对其Cache中的数据进行修改后，就会使得其Cache中的数据与其他Cache中的数据不一致

写作废协议： 在处理器进行写入操作之前，把所有其他Cache中的副本全部作废

写更新协议：当一个处理器对某个数据项进行写入的时候，它把该数据广播给其他所有Cache

栅栏同步：强制所有到达该栅栏的进程进行等待，直到全部的进行到达栅栏，然后释放全部的进程，从而实现同步。

旋钮锁：处理器不停地请求获得使用权的锁。

同时多线程：是一种在多流出、动态调度的处理器上同时开发线程级并行和指令级并行的技术，它是多线程技术的一种改进

细粒度多线程技术：是一种实现多线程的技术。它在每条指令之间都能进行线程的切换，从而使得多个线程可以交替执行。通常以时间片轮转的方法实现这样的交替执行，在轮转的过程中跳过处于停顿的线程

粗粒度多线程技术：是一种实现多线程的技术。只有线程发生较长时间的停顿时才切换到其他线程。

MPP：即大规模并行处理，按照当前的标准，具有几百台～几千台处理机的任何机器都是大规模并行处理系统

10.2

答：

共享存储器通信的优点：

1. 与常用的对称式多处理机使用的通信机制兼容
2. 当处理器之间通信方式复杂或在执行过程中动态变化时，采用共享存储器通信，编程容易，同时在简化编译器设计方面也具有优势
3. 采用大家熟悉的共享存储器开发模型开发程序，而把重点放到解决对性能影响较大的数据访问上
4. 当通信数据量较小时，通信开销较小，带宽利用较好
5. 可以通过采用Cache技术来减少远程通信的频度

消息传递通信机制的优点：

1. 硬件简单
2. 通信是显式的，因此更容易搞清楚何时发生通信以及通信开销是多少
3. 显式通信可以让编程者重点注意并行计算的主要通信开销，使之有可能开发出结构更好、性能更高的并行程序
4. 同步很自然地与发送消息相关联，能减少不当的同步带来错误的可能性

10.3

答：

多处理机的一致性：如果允许共享数据进入Cache，就可能出现多个处理器的Cache中都有同一存储快副本的情况，当其中某个处理器对其Cache中的数据进行修改后，就会使得其Cache中的数据与其他Cache中的数据不一样的情况。

目录式协议：物理存储器中的数据块的共享状态被保存在一个称为目录的地方。目录式协议的实现开销比监听式协议稍微大一下，但可用于实现更大规模的多处理机。

监听式协议：当物理存储器中的数据块被调入Cache中时，其共享状态信息与该数据块一起放在Cache中。系统中设有集中的状态表。这些Cache通常连在共享存储器的总线上。当某个Cache需要访问存储器时，它会把请求放到总线中广播出去，其他各个Cache控制器通过监听总线来判断他们是否有总线上请求的数据块。如果有，就进行相应的操作。

10.4

答：

（1）在对同一个数据进行多次写操作而中间无读操作的情况下，写更新操作协议需要多次进行写广播操作，而写作废协议则只需要一次作废操作，

（2）在对同一Cache块的多个字进行写操作的情况下，写更新协议对于每一个写操作都要进行一次广播，而写作废协议仅在对该块的第一次写进行作废操作即可，写作废操作是针对Cache块进行操作的，而写更新则是针对字（或字节）进行的。

（3）考虑从一个处理器A进行写操作后到另一个处理器B能读到数据之间的延迟时间。写更新操作中延迟比较小，而写作废协议中延迟比较高

10.5

答：

并行向量处理机

对称式共享存储器多处理机

分布式共享存储器多处理机

大规模并行处理机

机群计算机

10.6

解：



前者是后者速度的2倍

10.8

解：

我们忽略读写锁的时间。N个处理器中的每一个都需要C个时钟周期来锁住与栅栏相关的计数器，修改它的值，然后释放锁。考虑最坏情况，所有N个处理器都要对计数器加锁并修改它的值，由于锁只能顺序访问计数器，在同一时间，只能有一个处理器修改计数器的数据。所以，总共要花NC个时钟周期使得所有的处理器都到达数据栅栏。

10.9

解：

fetch-and-increment(count)；

if (count=total){

//进程全部到达 count=0； //重置计数器 release=1； //释放进程

}

else{

//还有进程未到达 spin(release=1)； //等待信号

}

当有N个处理器时，上述代码执行fetch-and-increment操作N次，当访问释放操作的时候，有N个Cache未命中。当最后一个处理器到达栅栏条件后，release被置为“1”，此时有N-1个Cache未命中（对于最后一个到达栅栏的处理器，当它读release的时候，将在主存中命中）。所以，共有3N-1次总线传输操作。如果有10个处理器，则共有29次总线传输操作，总共需要2900个时钟周期。

10.10

解：

当实现了专门的锁广播一致性协议后，每当一把锁被释放的时候，和锁相关的值将被广播到所有处理器，这意味着在处理器对锁变量进行读操作的时候，未命中的情况永远不会发生。

假定每个Cache都有一个数据块保留锁变量的初值。通过下表可以知道，10次上锁/释放锁的平均时间是550个时钟周期，总时间是5500个时钟周期