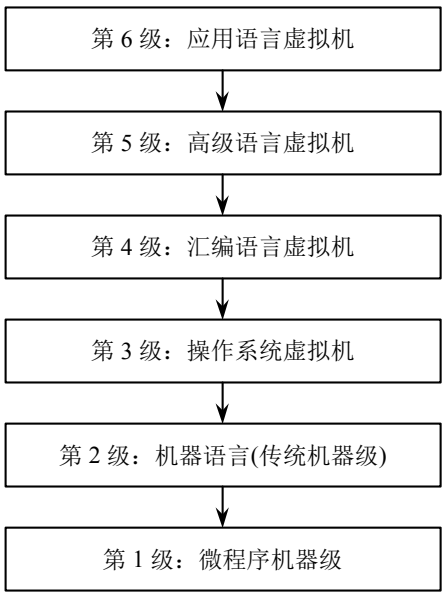
第一章 计算机体系结构的基本概念

1.1 名词解释：

1. 层次结构——计算机系统可以按语言的功能划分为多级层次结构，每一层以不同的语言为特征。



2. 翻译——（基于层次结构）先把 N+1 级程序全部变换成 N 级程序之后，再去执行 N 级程序，在执行过程中，N+1 级程序不再被访问。
3. 解释——每当一条 N+1 级指令被译码后，就直接去执行一串等效的 N 级指令，然后再去取下一条 N+1 级指令，依此重复执行。
4. 体系结构——程序员所看到的计算机的属性，即概念性结构与功能特性。
5. 透明性——在计算机技术中，对本来存在的事物或属性，从某一角度来看又好像不存在的概念称为透明性。
6. 系列机——在一个厂家生产的具有相同的体系结构，但具有不同的组成和实现的一系列不同型号的机器。
7. 软件兼容——同一个软件可以不加修改地运行于体系结构相同的各档机器上，而且它们所获得的结果一样，差别只在于运行的时间不同。
8. 兼容机——不同厂家生产的、具有相同体系结构的计算机。
9. 计算机组成——计算机体系结构的逻辑实现。
10. 计算机实现——计算机组成的物理实现。
11. 存储程序计算机（冯·诺依曼结构）——采用存储程序原理，将程序和数据存放在同一存储器中。指令在存储器中按其执行顺序存储，由指令计数器指明每条指令所在的单元地址。
12. 并行性——在同一时刻或同一时间间隔内完成两种或两种以上性质相同或不同的工作。
13. 时间重叠——在并行性中引入时间因素，即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。
14. 资源重复——在并行性中引入时间因素，是根据“以数量取胜”的原则，通过重复设

置资源，尤其是硬件资源，大幅度提高计算机系统的性能。

15. 资源共享——是一种软件方法，它使多个任务按一定的时间顺序轮流使用同一套硬件设备。
16. 同构型多处理机——由多个同种类型、至少同等功能的处理机组成、同时处理同一作业中能并行执行的多个任务的机器。
17. 异构型多处理机——由多个不同类型、功能不同的处理机组成、串行完成同一作业中不同任务的机器。
18. 最低耦合——是耦合度最低的系统，除通过某种中间存储介质之外，各计算机之间没有物理连接、也无共享的联机硬件资源。
19. 松散耦合——一般是通过通道或通信线路实现计算机之间连接、共享某些外围设备（例如磁盘、磁带），机器间的相互作用是在文件或数据集一级进行。
20. 紧密耦合——一般是指机间物理连接的频带较高，它们往往通过总线或高速开关实现互连，可以共享主存。
21. 响应时间——从事件开始到结束之间的时间，也称执行时间。
22. 测试程序——用于测试计算机性能的程序，可分为四类：真实程序、核心程序、小测试程序、合成测试程序。
23. 测试程序组件——选择一个各个方面有代表性的测试程序，组成一个通用的测试程序集合。这个通用的测试程序集合称为测试程序组件。
24. 大概率事件优先——此原则是计算机体系结构中最重要和最常用的原则。对于大概率事件（最常见的事件），赋予它优先的处理权和资源使用权，以获得全局的最优结果。
25. 系统加速比——系统改进前与改进后总执行时间之比。
26. Amdahl 定律——加快某部件执行速度所获得的系统性能加速比，受限于该部件在系统中的所占的重要性。
27. 程序的局部性原理——程序在执行时所访问的地址不是随机的，而是相对簇聚；这种簇聚包括指令和数据两部分。
28. CPI——指令时钟数（Cycles per Instruction）。

1.2 假设有一个计算机系统分为四级，每一级指令都比它下一级指令在功能上强 M 倍，即一条 r+1 级指令能够完成 M 条 r 指令的工作，且一条 r+1 级指令需要 N 条 r 级指令解释。对于一段在第一级执行时间为 K 的程序，在第二、第三、第四级上的一段等效程序需要执行多少时间？

解：

假设在第一级上用时间 K 执行了该级 IC 条指令。

对第二级而言，为了完成 IC 条指令的功能，第二级指令的条数为：$\frac{IC}{M}$。为了执行第二级 $\frac{IC}{M}$ 条指令，需要执行 $\frac{IC}{M}N$ 条第一级的指令对其进行解释，所以对于第二级而言，等效程序的执行时间是：

$$T_2 = \left[\frac{IC}{M}M + \frac{IC}{M}N\right]\frac{K}{IC}$$

$$= \left[1 + \frac{N}{M}\right]K$$

对于第三级而言，为了完成IC条指令的功能，第三级指令的条数为：$\frac{IC}{M^2}$。为了执行第三级$\frac{IC}{M^2}$条指令，需要执行$\frac{IC}{M^2}N$条第二级的指令对其进行解释。那么对第二级而言，总的指令条数为：

$$\frac{IC}{M^2}+\frac{IC}{M^2}N$$

而第二级$\frac{IC}{M^2}+\frac{IC}{M^2}N$等效于第一级$\left[\frac{IC}{M^2}+\frac{IC}{M^2}N\right]M$条指令，同时还需要$\left[\frac{IC}{M^2}+\frac{IC}{M^2}N\right]N$条第一级指令进行解释，所以第三级等效程序的执行时间是：

$$T_3=\left[\left[\frac{IC}{M^2}M+\frac{IC}{M^2}N\right]M+\left[\frac{IC}{M^2}M+\frac{IC}{M^2}N\right]N\right]\frac{K}{IC}$$

$$=\left[1+\frac{N}{M}\right]^2K$$

按照同样的逐层递推关系，不难求得第四级等效程序的总的执行时间为：

$$T_4=\left[1+\frac{N}{M}\right]^3K$$

1.2 传统存储程序计算机的主要特征是什么？存在的主要问题是什么？目前的计算机系统是如何改进的？

存储程序计算机在体系结构上的主要特点：

(1) 机器以运算器为中心。

(2) 采用存储程序原理。程序(指令)和数据放在同一存储器中，并且没有对两者加以区分。指令和数据一样可以送到运算器进行运算，即由指令组成的程序自身是可以修改的。

(3) 存储器是按地址访问的、线性编址的空间。

(4) 控制流由指令流产生。

(5) 指令由操作码和地址码组成。操作码指明本指令的操作类型，地址码指明操作数和操作结果的地址。

(6) 数据以二进制编码表示，采用二进制运算。

传统存储程序计算机体系结构存在的主要问题及其改进：

（1）分布的I/O处理能力

存储程序计算机以运算器为中心、所有部件的操作都由控制器集中控制，这一特点带来了慢速输入输出操作占用快速运算器的矛盾。为了克服这一缺点，人们先后提出各种输入/输出方式。

（2）保护的存储器空间

把指令和数据放在同一存储器中有优缺点。现在绝大多数计算机都规定：在执行过程中不准修改程序。

（3）存储器组织结构的发展

按地址访问的存储器具有结构简单、价格便宜、存取速度快等优点。但是在数据

处理时，往往要求查找具有某种内容特点的信息。但由于访问存储器的次数较多而影响计算机系统的性能。

采用了通用寄存器的概念、设置高速缓冲存储器 Cache、构成了以相联存储器为核心的相联处理机。

（4）并行处理技术

传统的存储程序计算机解题算法是顺序型的，即使问题本身可以并行处理，由于程序的执行受程序计数器控制，故只能是串行、顺序地执行。

如何挖掘传统机器中的并行性？

改进 CPU 的组成

在体系结构上使本来可以并行计算的题目能并行计算

多机并行处理系统

（5）指令集的发展

计算机系统指令的种类愈来愈多，这种计算机称为复杂指令集计算机 CISC。日趋庞杂的指令集不但不容易实现，而且还可能降低计算机系统的性能。

把指令集设计成只包含那些使用频率高的少量指令，并提供一些必要的指令以支持操作系统和高级语言。按照这个原则而构成的计算机称为精简指令集计算机 RISC。

1.4 对于一台 400MHz 计算机执行标准测试程序，程序中指令类型，执行数量和平均时钟周期数如下：

指令类型	指令执行数量	平均时钟周期数
整数	45000	1
数据传送	75000	2
浮点	8000	4
分支	1500	2

求该计算机的有效 CPI、MIPS 和程序执行时间。

解：$CPI = \sum (IC_i \times CPI_i) / IC$

$$CPI = \frac{45000 \times 1 + 75000 \times 2 + 8000 \times 4 + 1500 \times 2}{129500} = 1.776$$

$$MIPS速率 = \frac{f}{CPI \times 10^6} = \frac{400 \times 10^6}{1.776 \times 10^6} = 225.225 MIPS$$

程序执行时间=（$45000 \times 1 + 75000 \times 2 + 8000 \times 4 + 1500 \times 2$） / 400=575⌈s

1.5 计算机系统有三个部件可以改进，这三个部件的加速比如下：

部件加速比 1＝30； 部件加速比 2＝20； 部件加速比 3＝10；

(1) 如果部件 1 和部件 2 的可改进比例为 30%，那么当部件 3 的可改进比例为多少时，系统的加速比才可以达到 10？

(2) 如果三个部件的可改进比例为 30%、30%和 20%，三个部件同时改进，那么系统中不可加速部分的执行时间在总执行时间中占的比例是多少？

解：在多个部件可改进情况下 Amdahl 定理的扩展:

$$T_e = T_o\left[(1-f_e)+\frac{f_e}{S_e}\right]$$

$$S=\frac{1}{(1-f_e)+\frac{f_e}{S_e}}$$

$$S=\frac{1}{(1-\sum_i f_i)+\sum_i \frac{f_i}{S_i}}$$

式中，fi 为可加速部件 i 在未优化系统中所占的比例；Si 是部件 i 的加速比。

$$S=\left\{[1-(f_1+f_2+f_3)]+\frac{f_1}{S_1}+\frac{f_2}{S_2}+\frac{f_3}{S_3}\right\}^{-1}$$

$$10=\left\{[1-(0.3+0.3+f_3)]+\frac{0.3}{30}+\frac{0.3}{20}+\frac{f_3}{30}\right\}^{-1}$$

$$f_3=\frac{65}{180}=0.36$$

$$p=\frac{[1-(0.3+0.3+0.2)]T}{\frac{0.3T}{30}+\frac{0.3T}{20}+\frac{0.2T}{10}+0.2T}$$

$$=\frac{0.2}{\frac{0.3}{30}+\frac{0.3}{20}+\frac{0.2}{10}+0.2}$$

$$=\frac{0.2}{\frac{0.6}{60}+\frac{0.9}{60}+\frac{1.2}{60}+\frac{12}{60}}$$

$$=\frac{12}{14.7}=0.82$$

第二章　计算机指令集结构设计

2.1 名词解释

1. 堆栈型机器——CPU 中存储操作数的单元是堆栈的机器。
2. 累加型机器——CPU 中存储操作数的单元是累加器的机器。
3. 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。
4. CISC——复杂指令集计算机。
5. RISC——精简指令集计算机。

2.2 堆栈型机器、累加器型机器和通用寄存器型机器各有什么优缺点？

指令集结构类型	优点	缺点
堆栈型	是一种表示计算的简单模型；指令短小。	堆栈不能被随机访问，从而很难生成有效代码。同时，由于堆栈是瓶颈，所以很难被高效地实现。
累加器型	减小了机器的内部状态；指令短小。	由于累加器是唯一的暂存器，这种机器的存储器通信开销最大。
寄存器型	是代码生成最一般的模型。	所有操作数均需命名，且显式表示，因而指令比较长。

2.3 常见的三种通用寄存器型机器的优缺点各有哪些？

指令集结构类型	优　点	缺　点
寄存器－寄存器型（0,3）	简单，指令字长固定，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。	和指令中含有对存储器操作数访问的结构相比，指令条数多，因而其目标代码较大。
寄存器－存储器型（1,2）	可以直接对存储器操作数进行访问，容易对指令进行编码，且其目标代码较小。	指令中的操作数类型不同。在一条指令中同时对一个寄存器操作数和存储器操作数进行编码，将限制指令所能够表示的寄存器个数。由于指令的操作数可以存储在不同类型的存储器单元，所以每条指令的执行时钟周期数也不尽相同。
存储器－存储器型（3,3）	是一种最紧密的编码方式，无需“浪费”寄存器保存变量。	指令字长多种多样。每条指令的执行时钟周期数也大不一样，对存储器的频繁访问将导致存储器访问瓶颈问题。

2.4 指令集结构设计所涉及的内容有哪些？

(1) 指令集功能设计：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度，根据适用频度设置相应必要的寻址方式；

(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（可以采用 IEEE 754 标准）、整型数据类型（8 位、16 位、32 位的表示方法）、字符型（8 位）、十进制数据类型（压缩十进制和非压缩十进制数据表示）等等。

(4) 寻址方式的表示：可以将寻址方式编码与操作码中，也可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：有固定长度编码方式、可变长编码方式和混合编码方式三种选择。

2.5 简述 CISC 计算机结构指令集功能设计的主要目标。从当前的计算机技术观点来看，CISC 结构有什么缺点？

CISC 结构追求的目标是强化指令功能，减少程序的指令条数，以达到提高性能的目的。从目前的计算机技术观点来看，CISC 结构存在以下几个缺点：

(1) 在 CISC 结构的指令系统中，各种指令的使用频率相差悬殊。

(2) CISC 结构的指令系统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 结构的指令系统的复杂性给 VLSI 设计带来了很大负担，不利于单片集成。

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

(5) 在结构的指令系统中，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。

2.6 简述 RISC 结构的设计原则。

(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。

2.7 简述操作数的类型及其相应的表示方法。

操作数的类型主要有：整数（定点）、浮点、十进制、字符、字符串、向量、堆栈等。

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，这也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。

2.8 表示寻址方式的主要方法有哪些？简述这些方法的优缺点。

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。

2.9 通常有哪几种指令格式？简述其适用范围。

（1） 变长编码格式。如果体系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。

2.10 为了对编译器设计提供支持，在进行指令集设计时，应考虑哪些问题？

(1) 规整性。

(2) 提供基本指令，而非解决方案。

(3) “简化方案的折中取舍标准”。

(4) “对于在编译时已经知道的量，提供将其变为常数的指令”。

2.11 试就指令格式、寻址方式和每条指令的周期数（CPI）等方面比较 RISC 和 CISC 处理机的指令系统结构。

比较内容	CISC	RISC
指令格式	变长编码	定长编码
寻址方式	各种都有	只有 load/store 指令可以访存
CPI	远远大于 1	为 1

2.12 现有如下 C 语言源代码：

```
for (I=0;i<=100,i++)
{A[i]=B[i]+C;}
```

其中，A 和 B 是两个 32 位整数的数组，C 和 i 均是 32 位整数。假设所有数据的值及其地址均保存在存储器中，A 和 B 的起始地址分别是 0 和 5000。C 和 i 的地址分别是 1500 和 2000。在循环的两次迭代之间不将任何数据保存在寄存器中。

（1）请写出该 C 语言源程序的 DLX 实现代码。

（2）该程序段共执行了多少条指令。

（3）程序对存储器中的数据访问了多少次？

（4）DLX 代码的大小是多少？

解：

```
（1）ADDI   R1,R0,#1          ; 初始化 i
     SW     2000（R0）,R1     ; 存储 i
loop:
     LW     R1,2000(R0)       ; 得到 i 的值
     MULT   R2,R1,#4          ; R2 = B 的字偏移
     ADDI   R3,R2,#5000       ; 对 R2 加上基址

     LW     R4, 0(R3)         ; LD B[i]的值
     LW     R5,1500(R0)       ; LD C 的值

     ADD    R6,R4,R5      ; B[i] + C

     LW     R1,2000(R0)       ; 得到 i 的值
     MULT   R2,R1,#4          ; R2 = A 的字偏移
     ADDI   R7,R2,#0          ; 对 R2 加上基址
     SW     0(R7),R6          ;A[i] <—B[i] + C

     LW     R1,2000(R0)       ; 得到 i 的值
     ADDI   R1,R1,#1          ; 增加 i
     SW     2000(R0),R1       ;存储 i

     LW     R1,2000(,R0)      ;得到 i 的值
     ADDI   R8,R1,#-101       ;计数器是否为 101
```

```
        BNEZ    R8,loop             ;不是 101，重复
```

（2）总共执行的指令数是设置（setup）指令数加上循环中重复的指令条数：

执行的指令 ＝2+（16×101）=1618

为了计算数据访问的次数，可以用循环次数乘以每次循环数据访问次数再加上设置代码的次数：

（3）数据访问次数 ＝1+(8×10)×101= 809

代码大小就是程序中汇编指令数乘以 4（DLX 中每条指令占 4 字节）：

（4）代码大小 ＝4×18 = 72

2.13 参考 2.12 题，现在假设将 i 的值和数组变量的地址在程序运行过程中，只要有可能就一直保存在寄存器中。

（1）请写出该 C 语言源程序的 DLX 实现代码。

（2）该程序断共执行了多少条指令。

（3）程序对存储器中的数据访问了多少次？

（4）DLX 代码的大小是多少？

解：本题对上题再次讨论，只不过是在对机器代码进行基本的优化后。特别的，寄存器中的值可以重用，并且循环变量的代码应该提到循环之外。

注意到循环下标变量 i 的值仅在最后一次循环中存储，而且和以前一样，变量的地址小于 16 位，可以用立即数指令 load 地址。对 C 代码进行优化后的一个可能 DLX 代码如下：

```
        ADDI    R1,R0,#1            ; 初始化 i
        ADDI    R3,R0,#0            ; A 的基址
        ADDI    R4,R0,#5000         ; B 的基址
        LW      R5,1500（R0）       ; LD C 的值
loop:
        MULT    R2,R1,#4            ; 计算字偏移

        ADD     R6,R2,R4    ; 计算 B[i]地址
        LW      R7，0(R6)           ; LD B[i]的值

        ADD     R8,R7,R5    ; B[i] + C

        ADD     R9，R2，R3          ; 计算 A[i]的地址
        SW      0(R9),R8            ;A[i] <- B[i] + C

        ADDI    R1,R1,#1            ; 增加 i

        ADDI    R10,R1,#-101        ;计数器是否为 101
        BNEZ    R10,loop            ;不是 101，重复

out_of_loop:
```

SW　　2000(R0),R1　　；存储最后的 i 值

（2）总共执行的指令数是设置指令（setup）加上循环次数和循环指令数的乘积再加上清除指令数（cleanup）：　执行指令数 = 4 +（9×101）+1 = 914

（3）计算数据访问的次数可以用每次循环数据访问次数乘以循环次数再加上设置代码中的数据访问：

数据访问次数 = 1 + (2×101) +1 = 204

（4）代码大小是程序中汇编指令数乘以 4（DLX 中每条指令占 4 字节）：

代码大小 = (4×14) = 56

2.14 读写存储器的频率、访问指令和沪剧的频率是设计存储器系统的重要依据之一。参考表 2.16 中的整型平均指标，求：

（1）所有对数据的存储器访问所占的百分比；

（2）所有数据访问中读操作所占的百分比；

（3）所有存储器访问中读操作所占的百分比。

解：

（1）　所有对数据的存储器访问所占的比例为：26%;

（2）　所有数据访问中读操作所占的比例为：74%;

（3）　所有存储器访问中读操作所占的比例为：93%。

2.15 对表 2.16 中的所有类型指令，通过测量其 CPI，得到如下结果：

指令	时钟周期
所有的 ALU 指令	1
Load/Store 指令	1.4
成功的条件分支指令	2.0
失败的条件分支指令	1.5
跳转指令	1.2

假设 60%的条件分支指令转移成功，同时将上题表中其它一些类别的指令（没有被包含在上述类别中的指令）看作是 ALU 指令，请根据 gcc 和 espresso 基准程序计算上述各种类型指令出现的平均频率，以及这两个基准程序的有效（等效）CPI。

解：

上述指令所占的百分比如下表所示：

指令	时钟周期	百分比
所有 ALU	1	52%
Load/Store 指令	1.4	31.6%
成功的条件分支指令	2.0	9.8%
失败的条件分支指令	1.5	5.2%
跳转指令	1.2	2.7%

其等效 CPI 为：1×52%＋1.4×31.6%＋2.0×9.8%＋1.5×5.2%＋1.2×2.7% ＝ 1.23

第三章　　流水线技术

3.1 名词解释

1. 流水线——将一个重复的时序过程，分解为若干个子过程，而每一个子过程都可有效地在其专用功能段上与其他子过程同时执行。
2. 单功能流水线——只能完成一种固定功能的流水线。
3. 多功能流水线——流水线的各段可以进行不同的连接，从而使流水线在不同的时间，或者在同一时间完成不同的功能。
4. 静态流水线——同一时间内，流水线的各段只能按同一种功能的连接方式工作。
5. 动态流水线——同一时间内，当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。
6. 部件级流水线——（运算操作流水线）把处理机的算术逻辑部件分段，以便为各种数据类型进行流水操作。
7. 处理机级流水线——（指令流水线）把解释指令的过程按照流水方式处理。
8. 处理机间流水线——（宏流水线）由两个以上的处理机串行地对同一数据流进行处理，每一个处理机完成一项任务。
9. 线性流水线——指流水线的各段串行连接，没有反馈回路。
10. 非线性流水线——指流水线中除有串行连接的通路外，还有反馈回路。
11. 标量流水处理机——处理机不具有向量数据表示，仅对标量数据进行流水处理。
12. 向量流水处理机——处理机具有向量数据表示，并通过向量指令对向量的各元素进行处理。
13. 结构相关——某些指令组合在流水线中重叠执行时，发生资源冲突，则称该流水线有结构相关。
14. 数据相关——当指令在流水线中重叠执行时，流水线有可能改变指令读/写操作的顺序，使得读/写操作顺序不同于它们非流水实现时的顺序，将导致数据相关。
15. 定向——将计算结果从其产生的地方直接送到其他指令需要它的地方，或所有需要它的功能单元，避免暂停。
16. RAW——两条指令 i，j，i 在 j 前进入流水线，j 执行要用到 i 的结果，但当其在流水线中重叠执行时，j 可能在 i 写入其结果之前就先行对保存该结果的寄存器进行读操作，得到错误值。
17. WAW——两条指令 i，j，i 在 j 前进入流水线，j、i 的操作数一样，在流水线中重叠执行时，j 可能在 i 写入其结果之前就先行对保存该结果的寄存器进行写操作，导致写错误。
18. WAR——两条指令 i，j，i 在 j 前进入流水线，j 可能在 i 读某个寄存器之前对该寄存器进行写操作，导致 i 读出数据错误。

3.2 简述流水线技术的特点。

（1） 流水过程由多个相联系的子过程组成，每个过程称为流水线的“级”或“段”；
（2） 每个子过程由专用的功能段实现；
（3） 各个功能段所需时间应尽量相等，否则，时间长的功能段将成为流水线的瓶颈，会造成流水线的“堵塞”和“断流”；
（4） 流水线需要有“通过时间”(第一个任务流出结果所需的时间)，在此之后流水过程才进入稳定工作状态，每一个时钟周期(拍)流出一个结果；
（5） 流水技术适合于大量重复的时序过程，只有在输入端能连续地提供任务，流水线的效率才能充分发挥。

3.3 请画出 DLX 基本流水线，并简述其工作原理。

工作原理：把一条 DLX 指令在 5 个周期内实现，将每一个时钟周期看作是流水线的一个时钟周期，硬件每个时钟周期启动一条新的指令，并执行 5 条不同指令中的某一部分。每条指令虽仍需 5 个时钟周期完成，但提高了吞吐率，实现了流水。

指令/时钟	1	2	3	4	5	6	7	8	9
I	IF	ID	EX	MEM	WB				
I+1		IF	ID	EX	MEM	WB			
I+2			IF	ID	EX	MEM	WB		
I+3				IF	ID	EX	MEM	WB	
I+4					IF	ID	EX	MEM	WB

3.5 解决流水线结构相关的方法有哪些？

（1） 流水化功能单元
（2） 资源重复
（3） 暂停流水线

3.6 降低流水线分支损失的方法有哪些？

（1）在流水线中尽早判断出分支转移是否成功；
（2）尽早计算出分支转移成功时的 PC 值（即分支的目标地址）

- “冻结”或“排空”流水线的方法
- 预测分支失败
- 预测分支成功
- 延迟分支

3.7 请对延迟分支办法中的三种调度策略进行评价。

1．从前调动：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。
2．从目标处调度：若分支转移失败，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，可提高流水线性能。但由于复制指令，可能加大程序空间。
3．从失败处调度：若分支转移成功，必须保证被调度的指令对程序的执行无影响。

分支转移失败时，可提高流水线性能。

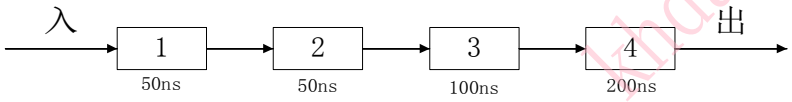
3.8 简述三种向量处理方法，它们对向量处理机的结构要求有什么不同？

1．水平处理方式：若向量长度为 N，则水平处理方式相当于执行 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。

2．垂直处理方式：将整个向量按相同的运算处理完毕之后，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联，构成 M-M 型的运算流水线。

3．分组处理方式：把长度为 N 的向量分为若干组，每组长度为 n，组内按纵向方式处理，依次处理各组，组数为 $\left\lceil \frac{N}{n} \right\rceil$，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联，构成 R-R 型运算流水线。

3.9 有一条流水线如下所示。



（1）求连续输入 10 条指令，该流水线的实际吞吐率和效率；

（2）该流水线的瓶颈在哪一段？请采取三种不同的措施消除此“瓶颈”。对于你所给出的新流水线，计算连续输入 10 条指令时，其实际吞吐率和效率。

解：（1）

$$T_{pipeline} = \sum_{i=1}^{m} \Delta t_i + (n-1)\Delta t_{max}$$

$$= (50 + 50 + 100 + 200) + 9 \times 200$$

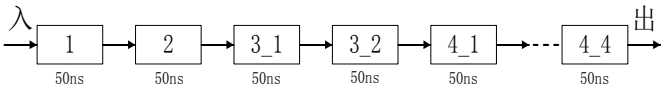
$$= 2200(ns)$$

$$TP = n / T_{pipeline} = 1/220 (ns^{-1})$$

$$E = TP \cdot \frac{\sum_{i=1}^{m} \Delta t_i}{m} = TP \cdot \frac{400}{4} = \frac{5}{11} \approx 45.45\%$$

（2）瓶颈在 3、4 段。

- 变成八级流水线（细分）



$$T_{pipeline} = \sum_{i=1}^{m} \Delta t_i + (n-1)\Delta t_{max}$$

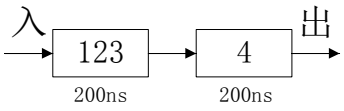
$$= 50 \times 8 + 9 \times 50$$

$$= 850(ns)$$

$$TP = n / T_{pipeline} = 1/85 (ns^{-1})$$

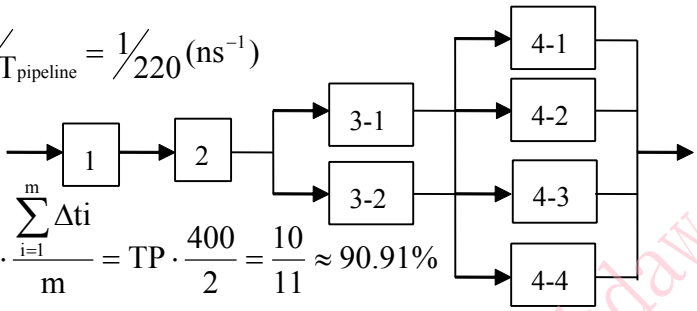
$$E = TP \cdot \frac{\sum_{i=1}^{m} \Delta ti}{m} = TP \cdot \frac{400}{8} = \frac{10}{17} \approx 58.82\%$$

- 变成两级流水线（合并）



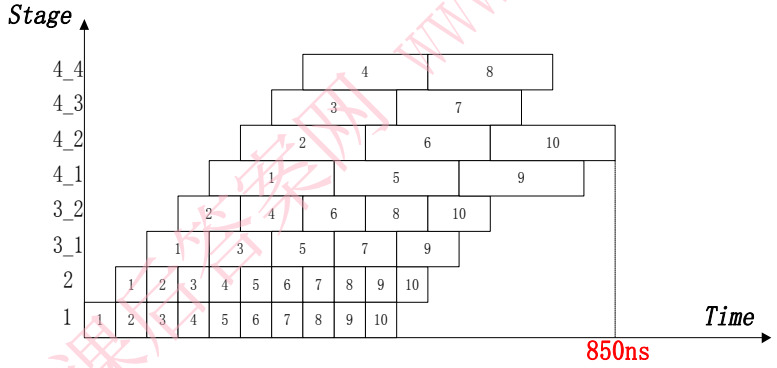
$$T_{pipeline} = \sum_{i=1}^{m} \Delta t_i + (n-1)\Delta t_{max}$$
$$= 200 \times 2 + 9 \times 200$$
$$= 2200(ns)$$

$$TP = n / T_{pipeline} = 1 / 220 (ns^{-1})$$



$$E = TP \cdot \frac{\sum_{i=1}^{m} \Delta ti}{m} = TP \cdot \frac{400}{2} = \frac{10}{11} \approx 90.91\%$$

- 重复设置部件



$$TP = n / T_{pipeline} = 1 / 85 (ns^{-1})$$
$$E = 400 \times 10 / 850 \times 8 = 10 / 17 \approx 58.82\%$$

3.10 一个流水线由四段组成，其中每当流经第三段时，总要在该段循环一次才能流到第四段。如果每段经过一次的时间都是△t，问：

（1）当在流水线的输入端每△t 时间输入任务时，该流水线会发生什么情况？

（2）此流水线的实际吞吐率为多少？如果每 2△t 输入一个任务，连续处理 10 个任务的实际吞吐率和效率是多少？

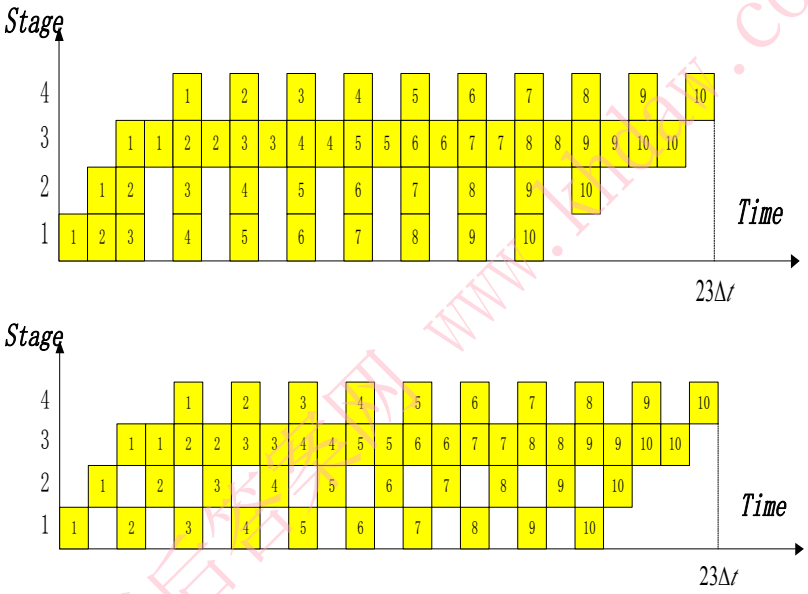
（3）当每段时间不变时，如何提高该流水线的吞吐率？仍连续处理 10 个任务 时，其吞吐率提高多少？

解：（1）会发生流水线阻塞情况。

Instr.1	stage1	stage2	stage3	stage3	stage4						
instr.2		stage1	stage2	stall	stage3	stage3	stage4				
instr.3			stage1	stall	stage2	stall	stage3	stage3	stage4		
instr.4				stall	stage1	stall	stage2	stall	stage3	stage3	stage4

（2）

	0t	1t	2t	3t	4t	5t	6t	7t	8t
Instr.1	stage1	stage2	stage3	stage3	stage4				
instr.2			stage1	stage2	stage3	stage3	stage4		
instr.3					stage1	stage2	stage3	stage3	stage4



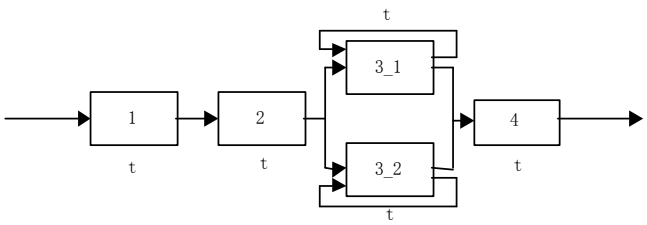
$$TP_{max} = \frac{1}{2\Delta t}$$

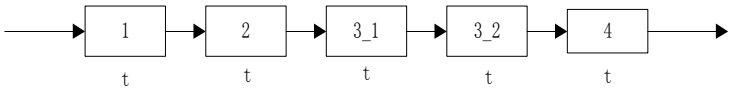
$$T_{pipeline} = 23\Delta t$$

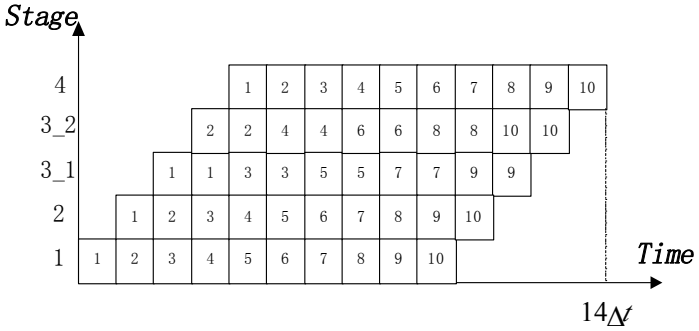
$$Tp = n / T_{pipeline} = 10 / 23\Delta t$$

$$\Delta E = TP \cdot 5\Delta t / 4 = 50 / 92 \approx 54.35\%$$

（3）重复设置部件







$$TP = n/T_{pipeline} = 10/14 \cdot \Delta t = 5/7 \cdot \Delta t$$

$$\text{吞吐率提高倍数} = \frac{5/7\Delta t}{10/23\Delta t} = 1.64$$

3.11　如果流水线有 m 段，各段的处理时间分别是 t_i（i=1，2，…，m），现在有 n 个任务需要完成，且每个任务均需流水线各段实现，请计算：

（1）流水线完成这 n 个任务所需要的时间；

（2）和非流水线实现相比，这 n 个任务流水实现的加速比是多少？加速比的峰值是多少？

解：（1）

$$T_{pipeline} = \sum_{i=1}^{m} t_i + (n-1) \cdot t_{max}$$

（2）

$$T_{nopipeline} = n \cdot \sum_{i=1}^{m} t_i$$

$$Speedup = T_{nopipeline}/T_{pipeline} = n \cdot \sum_{i=1}^{m} t_i \Big/ \sum_{i=1}^{m} t_i + (n-1) \cdot t_{max}$$

$$Speedup_{max} = m \cdot n/m + n - 1 \quad (t_i = t_0)$$

$$(n >> m, Speedup \to m)$$

3.12　在改进的 DLX 流水线上运行如下代码序列：

```
LOOP:  LW      R1, 0(R2)
       ADDI    R1, R1, #1
       SW      0(R2), R1
       ADDI    R2, R2, #4
```

SUB R4, R3, R2

BNZ R4, LOOP

其中，R3 的初始值是 R2＋396。假设：在整个代码序列的运行过程中，所有的存储器访问都是命中的，并且在一个时钟周期中对同一个寄存器的读操作和写操作可以通过寄存器“定向”。问：

（1）在没有任何其它定向（或旁路）硬件的支持下，请画出该指令序列执行的流水线时空图。假设采用排空流水线的策略处理分支指令，且所有的存储器访问都可以命中 Cache，那么执行上述循环需要多少个时钟周期？

（2）假设该 DLX 流水线有正常的定向路径，请画出该指令序列执行的流水线时空图。假设采用预测分支失败的策略处理分支指令，且所有的存储器访问都可以命中 Cache，那么执行上述循环需要多少个时钟周期？

（3）假设该 DLX 流水线有正常的定向路径，请对该循环中的指令进行调度。注意可以重新组织指令的顺序，也可以修改指令的操作数，但是不能增加指令的条数。请画出该指令序列执行的流水线时空图，并计算执行上述循环需要的时钟周期数？

解：

（1）寄存器读写可以定向，无其他旁路硬件支持。排空流水线。

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
lw r1,0(r2)	IF	ID	EX	M	WB																	
addi r1,r1,#1		IF	S	S	ID	EX	M	WB														
sw r1,0(r2)					IF	S	S	ID	EX	M	WB											
addi r2,r2,#4								IF	ID	EX	M	WB										
sub r4,r3,r2									IF	S	S	ID	EX	M	WB							
bnz r4,loop												IF	S	S	ID	EX	M	WB				
lw r1,0(r2)															IF	S	S	IF	ID	EX	M	WB

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

（2）有正常定向路径，预测分支失败。

Instruction	1	2	3	4	5	6	7	8	9	10	11	1	13	14	15
lw r1,0(r2)	IF	ID	EX	M	WB										
addi r1,r1,#1		IF	ID	S	EX	M	WB								
sw r1,0(r2)			IF	S	ID	EX	M	WB							
addi r2,r2,#4					IF	ID	EX	M	WB						
sub r4,r3,r2						IF	ID	EX	M	WB					
bnz r4,loop							IF	ID	EX	M	WB				
lw r1,0(r2)								IF	miss	miss	IF	ID	EX	M	WB

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

（3）有正常定向路径。单周期延迟分支。

```
Loop: lw   r1,0(r2)
addi   r2,r2,#4
addi r1,r1,#1
sub r4,r3,r2
bnz 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

Instruction	1	2	3	4	5	6	7	8	9	10	11
lw r1,0(r2)	IF	ID	EX	M	WB						
addi r2,r2,#4		IF	ID	EX	M	WB					
addi r1,r1,#1			IF	ID	EX	M	WB				
sub r4,r3,r2				IF	ID	EX	M	WB			
bnz r4,loop					IF	ID	EX	M	WB		
sw r1,-4(r2)						IF	ID	EX	M	WB	
lw r1,0(r2)							IF	ID	EX	M	WB

3.13　假设各种分支所占指令数地百分比如下表所示：

条件分支	20%（其中 60%是成功的）
跳转和调用	5%

现有一深度为 4 地流水线（流水线有 4 段），无条件分支在第二个时钟周期结束时就被解析出来，而条件分支要到第三个时钟周期结束时才能被解析出来。第一个流水段是完全独立于指令类型的，即所有的指令都必须经过第一个流水段的处理。请问在没有任何结构相关地情况下，该流水线相对于存在上述结构相关情况下地加速比是多少？

解：在不存在结构相关时，每条指令的平均执行时间是 1 个时钟周期，而存在上述条件相关的情况下，并假设条件分支预测成功，那么无条件分支和成功的条件分支的等待时间都是 1，而不成功地条件分支等待时间是 2 个周期；所以加速比就等于存在相关的每条指令的平均执行时间和不存在相关的每条指令的执行时间 1 的比值：

$$加速比 = 1 + C = 1 + f \times P_{分支}$$

$P_{无条件分支}$＝1stall

$P_{条件分支}$＝2stall

每条指令的平均等待时间：

$$C = f_{条件分支} \times P_{条件分支} + f_{无条件分支} \times P_{无条件分支}$$

$$= 20\% \times 60\% \times 2 + 20\% \times 40\% \times 1 + 5\% \times 1$$

$$= 0.37$$

所以：$加速比 = 1.37$

3.14 CRAY-1 机器上，按照链接方式执行下述 4 条向量指令（括号中给出了相应功能部件的时间），如果向量寄存器和功能部件之间数据传输需要 1 拍，试求此链接流水线的通过时间是多少拍？如果向量长度为 64，则需要多少拍才能得到全部结果。

V_0←存储器　　（从存储器中取数：7 拍）

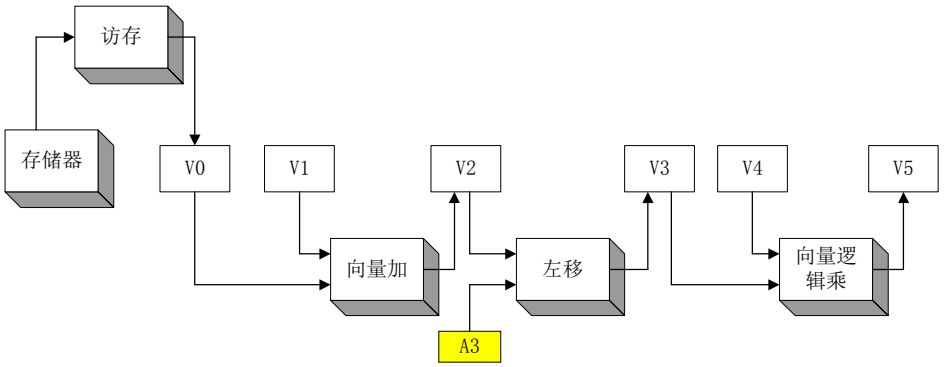
V_2←V_0＋V_1　　（向量加：3 拍）

V_2←V_2 < A_3　　（按(A3)左移：4 拍）

V_5←V_3∧V_4　　（向量逻辑乘：2 拍）

解：通过时间就是每条向量指令的第一个操作数执行完毕需要的时间，也就是各功

能流水线由空到满的时间，具体过程如下图所示。要得到全部结果，在流水线充满之后，向量中后继操作数继续以流水方式执行，直到整组向量执行完毕。



$T_{通过}$＝（7+1）＋（1+3+1）＋（1+4+1）＋（1+2+1）＝23(拍)

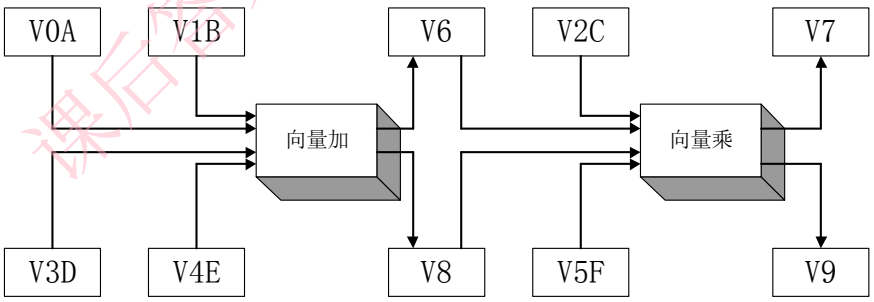
$T_{总共}$ = $T_{通过}$＋（64−1）＝23+63=86(拍)

3.15 向量处理机有 16 个向量寄存器，其中 V0～V5 中分别存放有向量 A、B、C、D、E、F，向量长度均为 8，向量各元素均为浮点数；处理部件采用两个单功能流水线，加法功能部件时间为 2 拍，乘法功能部件时间为 3 拍。采用类似 CRAY-1 的链接技术，先计算（A＋B）*C，在流水线不停留的情况下，接着计算（D＋E）*F。

（1）求此链接流水线的通过时间为多少拍？（设寄存器入、出各需 1 拍）

（2）假如每拍时间为 50ns，完成这些计算并把结果存进相应寄存器，此处理部件地实际吞吐率为多少 MFLOPS？

解：（1）我们在这里假设 A＋B 的中间结果放在 V6 中，（A＋B）*C 地最后结果放在 V7 中，D＋E 地中间结果放在 V8 中，（D＋E）*F 的最后结果放在 V9 中。具体实现参考下图：



通过时间应该为前者（（A＋B）*C）通过的时间：

$T_{通过}$= (1+2+1)+(1+3+1) =9（拍）

（2）在做完（A＋B）*C 之后，作（C＋D）*E 就不需要通过时间了。

$$V6 = A + B;$$

$$V7 = V6 * C;$$

$$V8 = D + E;$$

$$V9 = V8 * F;$$

$$T = T_{通过} + (8-1) + 8 = 24(拍) = 1200(ns)$$

$$TP = \frac{32}{T} = 26.67MFLOPS$$

第五章　存储层次

5.1 名词解释

1．存储层次——采用不同的技术实现的存储器，处在离 CPU 不同距离的层次上，目标是达到离 CPU 最近的存储器的速度，最远的存储器的容量。

2．全相联映象——主存中的任一块可以被放置到 Cache 中任意一个地方。

3．直接映象——主存中的每一块只能被放置到 Cache 中唯一的一个地方。

4．组相联映象——主存中的每一块可以放置到 Cache 中唯一的一组中任何一个地方（Cache 分成若干组，每组由若干块构成）。

5．替换算法——由于主存中的块比 Cache 中的块多，所以当要从主存中调一个块到 Cache 中时，会出现该块所映象到的一组（或一个）Cache 块已全部被占用的情况。这时，需要被迫腾出其中的某一块，以接纳新调入的块。

6．LRU——选择最近最少被访问的块作为被替换的块。实际实现都是选择最久没有被访问的块作为被替换的块。

7．写直达法——在执行写操作时，不仅把信息写入 Cache 中相应的块，而且也写入下一级存储器中相应的块。

8．写回法——只把信息写入 Cache 中相应块，该块只有被替换时，才被写回主存。

9．按写分配法——写失效时，先把所写单元所在的块调入 Cache，然后再进行写入。

10．不按写分配法——写失效时，直接写入下一级存储器中，而不把相应的块调入 Cache。

11．写合并——在往缓冲器写入地址和数据时，如果缓冲器中存在被修改过的块，就检查其地址，看看本次写入数据的地址是否和缓冲器内某个有效块的地址匹配。如果匹配，就将新数据与该块合并。

12．命中时间——访问 Cache 命中时所用的时间。

13．失效率——CPU 访存时，在一级存储器中找不到所需信息的概率。

14．失效开销——CPU 向二级存储器发出访问请求到把这个数据调入一级存储器所需的时间。

15．强制性失效——当第一次访问一个块时，该块不在 Cache 中，需要从下一级存储器中调入 Cache，这就是强制性失效。

16．容量失效——如果程序在执行时，所需要的块不能全部调入 Cache 中，则当某些块被替换后又重新被访问，就会产生失效，这种失效就称作容量失效。

17．冲突失效——在组相联或直接映象 Cache 中，若太多的块映象到同一组（块）中，则会出现该组中某个块被别的块替换（即使别的组或块有空闲位置），然后又被重新访问的情况。

18．2：1Cache 经验规则——大小为 N 的直接映象 Cache 的失效率约等于大小为 N /2 的两路组相联 Cache 的实效率。

19．相联度——在组相联中，每组 Cache 中的块数。

20．Victim Cache——位于 Cache 和存储器之间的又一级 Cache，容量小，采用全相联策略。用于存放由于失效而被丢弃（替换）的那些块。每当失效发生时，在访问下一级存储器之前，先检查 Victim Cache 中是否含有所需块。

21．伪相联 Cache——一种既能获得多路组相联 Cache 的低失效率，又能获得直接映象 Cache

的命中速度的相联办法。

22．故障性预取——在预取时，若出现虚地址故障或违反保护权限，就会发生异常。

23．非故障性预取——在预取时，若出现虚地址故障或违反保护权限，不发生异常。

24．非阻塞 Cache——Cache 在等待预取数据返回时，还能继续提供指令和数据。

25．子块放置技术——把一个 Cache 块划分为若干小块，称为子块（sub-blocks），并为每个子块赋予一位有效值，用于说明该子块中的数据是否有效。失效时，只需从下一级存储器调入一个子块。

26．尽早重启动——在请求字没有到达时，CPU 处于等待状态。一旦请求字到达，就立即发送给 CPU，让等待的 CPU 尽早重启动，继续执行。

27．请求字优先——调 块时，首先向存储器请求 CPU 所要的请求字。请求字一旦到达，就立即送往 CPU，让 CPU 继续执行，同时从存储器调入该块的其余部分。

28．多级包容性——一级存储器（Cache）中的数据总位于下一级存储器中。

29．虚拟 Cache——地址使用虚地址的 Cache。

30．多体交叉技术——具有多个存储体，各体之间按字交叉的存储技术。

31．存储体冲突——多个请求要访问同一个体。

32．TLB——一个专用高速存储器，用于存放近期经常使用的页表项，其内容是页表部分内容的一个副本。

5.2 简述**“Cache—主存”和“主存—辅存”层次的区别。**

存储层次 / 比较项目	“Cache—主存”层次	“主存—辅存”层次
目　　的	为了弥补主存速度的不足	为了弥补主存容量的不足
存储管理实现	全部由专用硬件实现	主要由软件实现
访问速度的比值（第一级比第二级）	几比一	几百比一
典型的块（页）大小	几十个字节	几百到几千个字节
CPU 对第二级的访问方式	可直接访问	均通过第一级
失效时 CPU 是否切换	不切换	切换到其它进程

5.3 降低 Cache 失效率有哪几种方法？简述其基本思想。

常用的降低 Cache 失效率的方法有下面几种：

（1）增加 Cache 块大小。增加块大小利用了程序的空间局部性。

（2）提高相联度，降低冲突失效。

（3）Victim Cache，降低冲突失效。

（4）伪相联 Cache，降低冲突失效。

（5）硬件预取技术，指令和数据都可以在处理器提出访问请求前进行预取。

（6）由编译器控制的预取，硬件预取的替代方法，在编译时加入预取的指令，在数据被用到之前发出预取请求。

（7）编译器优化，通过对软件的优化来降低失效率。

5.4 简述减小 Cache 失效的几种方法。

（1）让读失效优先于写。
（2）子块放置技术。
（3）请求字处理技术。
（4）非阻塞 Cache 技术。
（5）采用两级 Cache、

5.5 通过编译器对程序优化来改进 Cache 性能的方法有哪几种？简述其基本思想。

（1）数组合并，通过提高空间局部性来减少失效次数。有些程序同时用相同的索引来访问若干个数组的同一维，这些访问可能会相互干扰，导致冲突失效，可以将这些相互独立的数组合并成一个复合数组，使得一个 Cache 块中能包含全部所需元素。

（2）内外循环交换。循环嵌套时，程序没有按数据在存储器中的循序访问。只要简单地交换内外循环，就能使程序按数据在存储器中的存储循序进行访问。

（3）循环融合。有些程序含有几部分独立的程序断，它们用相同的循环访问同样的数组，对相同的数据作不同的运算。通过将它们融合成一个单一循环，能使读入 Cache 的数据被替换出去之前得到反复的使用。

（4）分块。通过改进时间局部性来减少失效。分块不是对数组的整行或整列进行访问，而是对子矩阵或块进行操作。

5.6 在**“Cache—主存”**层次中，主存的更新算法有哪几种？？它们各有什么特点？

（1） 写直达法

易于实现，而且下一级存储器中的数据总是最新的。

（2） 写回法

速度块，“写”操作能以 Cache 存储器的速度进行。而且对于同一单元的多个写最后只需一次写回下一级存储器，有些“写”只到达 Cache，不到达主存，因而所使用的存储器频带较低。

5.7 组相联 Cache 比相同容量的之直接映象 Cache 的失效率低。由此是否可以得出结论：采用组相联 Cache 一定能带来性能上的提高？为什么？

答：不一定。因为组相联命中率的提高是以增加命中时间为代价的，组相联需要增加多路选择开关。

5.8 写出三级 Cache 的平均访问时间 T_A 的公式。

平均访存时间 ＝ 命中时间＋失效率×失效开销

只有第 I 层的失效时才会访问第 I＋1

设三级 Cache 的命中率分别为 H_{L1}、 H_{12}、 H_{L3}，失效率分别为 M_{11}、M_{12}、M_{L3}，第三级 Cache 的失效开销为 P_{L3}。

平均访问时间 $T_A = H_{L1}+M_{11}\{H_{12}+M_{12}(H_{L3}+M_{L3}\times P_{L3})\}$

5.9 给定以下的假设，试计算直接映象 Cache 和两路组相联 Cache 的平均访问时间以及 CPU 的性能。由计算结果能得出什么结论？

（1）理想 Cache 情况下的 CPI 为 2.0，时钟周期为 2ns，平均每条指令访存 1.2 次；

（2）两者 Cache 容量均为 64KB，块大小都是 32 字节；

（3）组相联 Cache 中的多路选择器使 CPU 的时钟周期增加了 10%；

（4）这两种 Cache 的失效开销都是 80ns；

（5）命中时间为 1 个时钟周期；

（6）64KB 直接映象 Cache 的失效率为 1.4%，64KB 两路组相联 Cache 的失效率为 1.0%。

解：　平均访问时间＝命中时间＋失效率×失效开销

平均访问时间 $_{1\text{-路}}$=2.0+1.4% *80=3.12ns

平均访问时间 $_{2\text{-路}}$=2.0*(1+10%)+1.0% *80=3.0ns

两路组相联的平均访问时间比较低

CPU_{time}=（CPU $_{执行}$+存储等待周期）*时钟周期

CPU $_{time}$=IC（CPI $_{执行}$+总失效次数/指令总数*失效开销）　*时钟周期

=IC（（CPI $_{执行}$*时钟周期）+（每条指令的访存次数*失效率*失效开销*时钟周期））

CPU $_{time\ 1\text{-}way}$=IC(2.0*2+1.2*0.014*80)＝5.344IC

CPU $_{time\ 2\text{-}way}$=IC(2.2*2+1.2*0.01*80)＝5.36IC

相对性能比：$\dfrac{CPU_{time-2way}}{CPU_{time-1way}}=5.36/5.344=1.003$

直接映象 cache 的访问速度比两路组相联 cache 要快 1.04 倍，而两路组相联 Cache 的平均性能比直接映象 cache 要高 1.003 倍。因此这里选择两路组相联。

5.10 假设一台计算机具有以下特性：

（1） 95%的访存在 Cache 中命中；

（2） 块大小为两个字，且失效时整个块被调入；

（3） CPU 发出访存请求的速率为 10^9 字/秒；

（4） 25%的访存为写访问；

（5） 存储器的最大流量为 10^9 字/秒（包括读和写）；

（6） 主存每次只能读或写一个字；

（7） 在任何时候，Cache 中 有 30%的块被修改过；

（8） 写失效时，Cache 采用写分配法。

现欲给计算机增添一台外设，为此想先知道主存的频带已经使用了多少。试对于以下两种情况计算主存频带的平均使用比例。

（1）写直达 Cache；

（2）写回法 Cache。

解：采用按写分配

（1）写直达 cache 访问命中，有两种情况：

读命中，不访问主存；

写命中，更新 cache 和主存，访问主存一次。

访问失效，有两种情况：

读失效，将主存中的块调入 cache 中，访问主存两次；

写失效，将要写的块调入 cache，访问主存两次，再将修改的数据写入 cache

和主存，访问主存一次，共三次。上述分析如下表所示。

访问命中	访问类型	频率	访存次数
Y	读	95%*75%=71.3%	0
Y	写	95%*25%=23.8%	1
N	读	5%*75%=3.8%	2
N	写	5%*25%=1.3%	3

一次访存请求最后真正的平均访存次数=(71.3%*0)+(23.8%*1)+(3.8%*2)+(1.3%*3)＝0.35

已用带宽=$0.35\times10^9/10^9$=35.0%

（2）写回法 cache 访问命中,有两种情况：

读命中，不访问主存；

写命中，不访问主存。采用写回法，只有当修改的 cache 块被换出时，才写入主存；

访问失效,有一个块将被换出，这也有两种情况：

如果被替换的块没有修改过，将主存中的块调入 cache 块中，访问主存两次；

如果被替换的块修改过，则首先将修改的块写入主存，需要访问主存两次；然后将主存中的块调入 cache 块中，需要访问主存两次，共四次访问主存。

访问命中	块为脏	频率	访存次数
Y	N	95%*70%=66.5%	0
Y	Y	95%*30%=28.5%	0
N	N	5%*70%=3.5%	2
N	Y	5%*30%=1.5%	4

所以：

一次访存请求最后真正的平均访存次数=66.5％*0＋28.5%*0+3.5%*2+1.5%*4=0.13

已用带宽＝$0.13\times10^9/10^9$＝13%

5.11 伪相联中，假设在直接映象位置没有发现匹配，而在另一个位置才找到数据（伪命中）时，需要 1 个额外的周期，而且不交换两个 Cache 中的数据，失效开销为 50 个时钟周期。试求：

（1）推导出平均访存的时间公式。

（2）利用（1）中得到的公式，对于 2KBCache 和 128KBCache，重新计算伪相联的平均访存时间。请问哪一种伪相联更快？

假设 2KB 直接映象 Cache 的总失效率为 0.098，2 路相联的总失效率为 0.076；

128KB 直接映象 Cache 的总失效率为 0.010，2 路相联的总失效率为 0.007。

解：

不管作了何种改进，失效开销相同。不管是否交换内容，在同一“伪相联”组中的两块都是用同一个索引得到的，因此失效率相同，即：失效率$_{伪相联}$＝失效率$_{2路}$。

伪相联 cache 的命中时间等于直接映象 cache 的命中时间加上伪相联查找过程中的命中时间*该命中所需的额外开销。

命中时间$_{伪相联}$＝命中时间$_{1路}$＋伪命中率$_{伪相联}\times1$

交换或不交换内容，伪相联的命中率都是由于在第一次失效时，将地址取反，再在第二次查找带来的。

因此 伪命中率$_{伪相联}$＝命中率$_{2路}$－命中率$_{1路}$＝（1－失效率$_{2路}$）－（1－失效率$_{1路}$）

＝失效率$_{1路}$－失效率$_{2路}$。交换内容需要增加伪相联的额外开销。

平均访存时间$_{伪相联}$＝命中时间$_{1路}$＋（失效率$_{1路}$－失效率$_{2路}$）×1

＋失效率$_{2路}$×失效开销$_{1路}$

将题设中的数据带入计算，得到：

平均访存时间$_{2Kb}$=1+(0.098-0.076)*1+(0.076 *50) =4.822

平均访存时间 $_{128Kb}$=1+(0.010-0.007)*1+(0.007 *50) =1.353

显然是 128KB 的伪相联 Cache 要快一些。

5.12 假设采用理想存储器系统时的基本CPI是1.5，试利用表5.5 ，分别对于下述三种Cache计算 CPI：

(1) 16KB 直接映象统一 Cache，采用写回法；

(2) 16KB 两路组相联统一 Cache，采用写回法；

(3) 32KB 直接映象统一 Cache，采用写回法。

解： CPI=CPI $_{执行}$+存储停顿周期数/指令数

存储停顿由下列原因引起：

- 从主存中取指令
- Lload 和 Store 指令访问数据
- 由 TLB 引起

$$\frac{\text{存储停顿周期数}}{\text{指令数}}=\frac{\text{取指令停顿}}{\text{指令数}}+\frac{\text{数据访问停顿}+\text{TLB停顿}}{\text{指令数}}$$

$$\frac{\text{停顿周期数}}{\text{指令数}}=\frac{\text{存储访问}}{\text{指令数}}\times\text{失效率}\times\text{失效开销}$$

$$\frac{\text{存储停顿周期数}}{\text{指令数}}=\left(R_{指令}P_{指令}\right)+（f_{数据}R_{数据}P_{数据}）+\frac{\text{TLB停顿}}{\text{指令数}}$$

.对于理想 TLB,TLB 失效开销为 0。而对于统一 Cache, R $_{指令}$=R $_{数据}$

P $_{指令}$=主存延迟＋传输一个块需要使用的时间＝40＋32/4＝48（拍）

若为读失效，P $_{数据}$＝主存延迟＋传输一个块需要使用的时间＝40＋32/4＝48（拍）

若为写失效，且块是干净的，

P $_{数据}$＝主存延迟＋传输一个块需要使用的时间＝40＋32/4＝48（拍）

若为写失效，且块是脏的，

P $_{数据}$＝主存延迟＋传输两个块需要使用的时间＝40＋64/4＝56（拍）

CPI=1.5+[RP+(RP*20%)+0]

指令访存全是读，而数据传输指令 Load 或 Store 指令，

f $_{数据}$*P $_{数据}$＝读百分比*（f $_{数据}$*P $_{数据}$）＋写百分比*（f $_{数据}$*P $_{干净数据}$*其对应的百分比

＋f $_{数据}$*P $_{脏数据}$*其对应的百分比）

＝20%*（75％×48＋25％*（50％*48+50％*（48＋16））＝50（拍）

代入上述公式计算出结果为：

配置	失效率	CPI
16KB 直接统一映象	0.029	2.95
16KB 两路统一映象	0.022	2.6
32KB 直接统一映象	0.020	2.5

第六章 输入输出系统

6.1 假设一台计算机的 I/O 处理占 10%，当其 CPU 性能改进到原来的 100 倍时，而 I/O 性能仅改进为原来的两倍时，系统总体性能会有什么改进？

解： 加速比 $= \dfrac{1}{10\%/2 + 90\%/10} = 7.14$

本题再次反映了 Amdahl 定律，要改进一个系统的性能要对各方面性能都进行改进，不然系统中最慢的地方就成为新系统的瓶颈。

6.2 假设磁盘空闲，这样没有排队延迟；公布的平均寻道时间是 9ms，传输速度是 4MB/s，转速是 5400r/min，控制器的开销是 1ms。问读或写一个 512 字节的扇区的平均时间是多少？

解：

平均磁盘访问时间

= 平均寻道时间 + 平均旋转延迟 + 传输时间 + 控制器开销

$$9\text{ms} + \frac{0.5}{5400\text{r/min}} + \frac{0.5\text{KB}}{4.0\text{MB/s}} + 1\text{ms} = 9 + 5.6 + 0.125 + 1 = 15.725\text{m}$$

假设实际测得的寻道时间是公布值的 33%，则答案是：

3ms + 4.2ms + 0.1ms + 1ms = 8.3ms

6.3 按表 6.2 中的数据，问用哪种磁盘最适合于建立 100GB 的镜像盘阵列（RAID1）存储子系统？又问哪种磁盘最适合于建立 100GB 的镜像盘阵列（RAID5）存储子系统？

6.4 盘阵列有哪些分级？各有什么特点？

RAID0 亦称数据分块，即把数据分布在多个盘上，实际上是非冗余阵列，无冗余信息。

RAID1 亦称镜像盘，使用双备份磁盘。每当数据写入一个磁盘时，将该数据也写到另一个冗余盘，这样形成信息的两份复制品。如果一个磁盘失效，系统可以到镜像盘中获得所需要的信息。镜像是最昂贵的解决方法。特点是系统可靠性很高，但效率很低。

RAID2 位交叉式海明编码阵列。原理上比较优越，但冗余信息的开销太大，因此未被广泛应用。

RAID3 位交叉奇偶校验盘阵列，是单盘容错并行传输的阵列。即数据以位或字节交叉的方式存于各盘，冗余的奇偶校验信息存储在一台专用盘上。

RAID4 专用奇偶校验独立存取盘阵列。即数据以块(块大小可变)交叉的方式存于各盘，冗余的奇偶校验信息存在一台专用盘上。

RAID5 块交叉分布式奇偶校验盘阵列，是旋转奇偶校验独立存取的阵列。即数据以块交叉的方式存于各盘，但无专用的校验盘，而是把冗余的奇偶校验信息均匀地分布在所有磁盘上。

RAID6 双维奇偶校验独立存取盘阵列。即数据以块(块大小可变)交叉的方式存于各盘，冗余的检、纠错信息均匀地分布在所有磁盘上。并且，每次写入数据都要访问一个数据盘和两个校验盘，可容忍双盘出错。

RAID7 是采用 Cache 和异步技术的 RAID6，使响应速度和传输速率有了较大提高。

6.5 磁带作为海量档案存储设备，平均每个磁带的读者很少。假设一个系统中，磁带读取速率为 9MB/s，更换一个磁带需要 30s，请计算 10 个读者全部读完 6000 个磁带，需要多长时间？

6.6 同步总线和异步总线各有什么优缺点？总线的主要参数有哪些？各是什么含义？

同步总线上所有设备通过统一的总线时钟进行同步。同步总线成本低，因为它不需要设备之间相互确定时序的逻辑。但是同步总线也有缺点，总线操作必须以相同的速度运行。由于各种设备都要精确地以公共时钟为定时参考，因此在时钟频率很高时容易产生时钟相对漂移错误。

异步总线上的设备之间没有统一的时钟，设备自己内部定时。设备之间的信息传送用总线发送器和接收器控制。异步总线容易适应更广泛的设备类型，扩充总线时不用担心时钟时序和时钟同步问题。但在传输时，异步总线需要额外的同步开销。

总线常用的参数有 3 个：

（1）Tp：总线信号传输延迟。即在总线上的每个设备都取到和识别一个信号需要的最大时间。

（2）Tsk：响应其它设备的最大时间，这个参数在同步总线中是一个重要的参数。

（3）Top：设备的操作时间。

6.7

6.9 在有 Cache 的计算机系统中，进行 I/O 操作时，会产生哪些数据不一致问题？如何克服？

（1）存储器中可能不是 CPU 产生的最新数据 ，所以 I/O 系统从存储器中取出来的是陈旧数据。

（2）I/O 系统与存储器交换数据之后，在 Cache 中，被 CPU 使用的可能就会是陈旧数据。

第一个问题可以用写直达 Cache 解决。

第二个问题操作系统可以保证 I/O 操作的数据不在 cache 中。如果不能，就作废 Cache 中相应的数据。

6.10 假设在一个计算机系统中：

(1) 每页为 32KB，Cache 块大小为 128 字节；
(2) 对应新页的地址不在 Cache 中，CPU 不访问新页中的数据；
(3) Cache 中 95%的被替换块将再次被读取，并引起一次失效；
(4) Cache 使用写回方法，平均 60%的块修改过；
(5) I/O 系统缓冲能够存储一个 Cache 完整的块（这称为速度匹配缓冲区，使存储器和 I/O 的速度得到匹配）；
(6) 访问或失效在所有的 Cache 中均匀分布；
(7) 在 CPU 和 I/O 之间，没有其它访问 Cache 的干扰；
(8) 无 I/O 时，每 100 万个时钟周期中，有 18000 次失效；
(9) 失效开销是 40 个时钟周期。如果替换块被修改过，则再加上 30 个周期用于写回主存；
(10)假设机器平均每 200 万周期处理 1 页。
分析 I/O 对于性能的影响有多大？

解：每个主存页有 32K/128＝256 块。

因为是按块传输，所以 I/O 传输本身并不引起 Cache 失效。但是它可能要替换 Cache 中的有效块。如果这些被替换块中有 60%是被修改过的，将需要（256×60%）×30＝4608 个时钟周期将这些被修改过的块写回主存。

这些被替换出去的块中，有 95%的后继需要访问，从而产生 95%×256＝244 次失效，将再次发生替换。由于这次被替换的 244 块中数据是从 I/O 直接写入 Cache 的，因此所有块都为被修改块，需要写回主存（因为 CPU 不会直接访问从 I/O 来的新页中的数据，所以它们不会立即从主存中调入 Cache），需要时间是 244×（40＋30）＝17080 个时钟周期。

没有 I/O 时，每一页平均使用 200 万个时钟周期，Cache 失效 36000 次，其中 60%被修改过，所需的处理时间为：

（36000×40%）×40＋（36000×60%）×（40＋30）＝2088000（时钟周期）

时钟 I/O 造成的额外性能损失比例为

（4608＋17080）÷（2000000＋2088000）＝0.53%

即大约产生 0.53%的性能损失。