课后作业评讲

马立平
maliping@swust.edu.cn

第一章 计算机系统结构的概念作业

1.1（层次结构、系统结构、计算机组成、计算机实现、Amdahl定律、程序的局部性原理、异构型多处理、同构型多处理机）

1.3，1.5，1.6，1.7，**1.8，1.9**

1.1 层次结构、系统结构、计算机组成、计算机实现、Amdahl定律、程序的局部性原理、异构型多处理、同构型多处理机

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，机器语言（传统机器级），操作系统虚拟机，汇编语言虚拟机，高级语言虚拟机，应用语言虚拟机等。

计算机系统结构——指机器语言程序员所看到的计算机属性，即概念性结构与功能特性。

计算机组成——指的是计算机系统结构的逻辑实现，包含物理机器级中的数据流和控制流的组成以及逻辑设计等。

计算机实现——指的是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，模块、插件、底板的划分与连接，信号传输，电源、冷却及整机装配技术等。

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。

1.1 层次结构、系统结构、计算机组成、计算机实现、Amdahl定律、程序的局部性原理、异构型多处理、同构型多处理机

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。

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。

第一章 计算机系统结构的概念作业

1.3 解

Flynn 分类法是按照指令流和数据流的多倍性进行分类。把计算机系统的结构分为：

（1） 单指令流单数据流 SISD

（2） 单指令流多数据流 SIMD

（3） 多指令流单数据流 MISD

（4） 多指令流多数据流 MIMD

1.5 解

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。

1.6 解：

（1）CPI $=(45000\times1+75000\times2+8000\times4+1500\times2)/129500=1.776$

（或$\frac{460}{259}$）

（2）MIPS 速率$=f/\text{CPI}=400/1.776=225.225\text{MIPS}$（或$\frac{5180}{259}$MIPS）

（3）程序执行时间$=(45000\times1+75000\times2+8000\times4+1500\times2)/400=575\mu s$

由题可知，可改进比例 $=40\%=0.4$，部件加速比 $=10$。

根据 Amdahl 定律可知，

$$\text{系统加速比}=\frac{1}{(1-0.4)+\frac{0.4}{10}}=1.562\,5$$

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 1.562 5 倍。

1.8 解：

（1）在多个部件可改进情况下，Amdahl 定理的扩展：

$$S_n = \frac{1}{(1-\sum F_i)+\sum \frac{F_i}{S_i}}$$

已知 $S_1=30$，$S_2=20$，$S_3=10$，$S_n=10$，$F_1=0.3$，$F_2=0.3$，得：

$$10 = \frac{1}{1-(0.3+0.3+F_3)+(0.3/30+0.3/20+F_3/10)}$$

得 $F_3=0.36$，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 $S_1=30$，$S_2=20$，$S_3=10$，因此 3 个部件改进后的执行时间为：

$$T_n' = \frac{0.3T}{30}+\frac{0.3T}{20}+\frac{0.2T}{10} = 0.045T$$

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：

$$\frac{0.2T}{0.245T} = 0.82$$

1.9 解：

根据 Amdahl 定律 $S_n = \dfrac{1}{(1-F_e)+\dfrac{F_e}{S_e}}$，可得如表 1.4 所示的 4 类操作的加速比和程序获得的加速比。

操作类型	各类操作的指令条数在程序中所占的比例 F_i	各类操作的加速比 S_i	各类操作单独改进后，程序获得的加速比
操作 1	11.1%	2	1.06
操作 2	33.3%	1.33	1.09
操作 3	38.9%	3.33	1.37
操作 4	16.7%	4	1.14

4 类操作均改进后，整个程序的加速比为

$$S_n = \frac{1}{(1-\sum F_i)+\sum \dfrac{F_i}{S_i}} \approx 2.15$$

第二章 计算机指令级结构课后作业

2.1（通过寄存器型计算机）、2.3、**2.5**，**2.7**，2.9，2.11

通用寄存器型计算机——CPU 中存储操作数的单元是通用寄存器的计算机。

2.3、解：

① 寄存器－寄存器型。

优点：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。

缺点：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。

② 寄存器－存储器型。

优点：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。

缺点：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。

③ 存储器－存储器型。

优点：目标代码最紧凑，不需要设置寄存器来保存变量。

缺点：指令字长变化很大，特别是三操作数指令。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。

第二章 计算机指令级结构课后作业

2.5、解：

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，查看各种寻址方式的使用频率，根据使用频率设置必要的寻址方式。③ 操作数表示和操作数类型。可选择浮点型数据类型、整型数据类型、字符型、十进制数据类型等。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字段来表示。⑤ 指令格式的设计。有变长编码格式、定长编码格式和混合型编码格式 3 种。

2.7、解：

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，其他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。

第二章 计算机指令级结构课后作业

2.9、解：

表示寻址方式有两种常用的方法。① 将寻址方式编码于操作码中，操作码在描述指令的同时也描述相应的寻址方式。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。

2.11、解：

CPU 性能公式为 CPU 时间 = IC × CPI × T。其中，IC 为目标程序被执行的指令条数，CPI 为指令平均执行周期数，T 是时钟周期的时间。

相同功能的 CISC 目标程序的指令条数 IC_{CISC} 少于 RISC 的 IC_{RISC}，但是 CISC 的 CPI_{CISC} 和 T_{CISC} 都大于 RISC 的 CPI_{RISC} 和 T_{RISC}，因此，CISC 目标程序的执行时间比 RISC 的更长。

第三章 流水线技术课后作业

3.1（部件级流水线、处理机级流水线、处理机间流水线、数据相关、数据冲突、控制冲突、定向、链接技术、分段开采），3.2，3.5，3.9，**3.11**、**3.12**、3.13

部件级流水线——又称运算操作流水线。它将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。

指令流水线——又称处理机级流水线。它是把指令的执行过程按照流水方式进行处理，即把一条指令的执行过程分解为若干个子过程，每个子过程在独立的功能部件中执行。

处理机间流水线——又称宏流水线。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。

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。

第三章 流水线技术课后作业

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他会改变 PC 值的指令所引起的冲突。

定向技术——用于解决写后读冲突。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真正立刻就要用该结果。如果能够将该计算结果从其产生的地方直接送到其他指令需要它的地方，那么就可以避免停顿。

链接技术——具有先写后读相关的两条指令，在不出现功能部件冲突和 V_i 冲突的情况下，可以把功能部件链接起来进行流水处理，以达到加快执行的目的。

分段开采技术——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。

第三章 流水线技术课后作业

3.2 解：

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后面都要有一个缓冲寄存器，称之为流水寄存器。

④ 流水技术适用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。

3.5 解：

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，可能增大程序空间
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

3.9 解：

① $$T_{\text{pipeline}} = \sum_{i=1}^{m} \Delta t_i + (n-1)\Delta t_{\max} = (50+50+100+200) + 9 \times 200 = 2\,200$$

$$\text{TP} = n/T_{\text{pipeline}} = 1/220$$

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

② 第 3、4 段是瓶颈。

措施 1：变成八级流水线（细分），如图 3.8 所示。

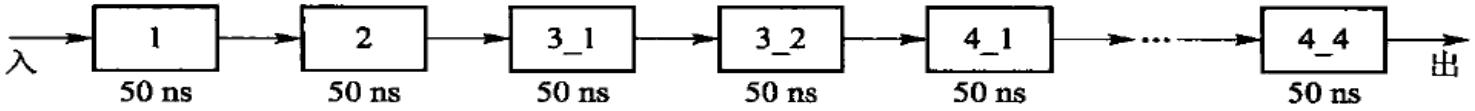

图 3.8　措施 1 的指令流水线

$$T_{\text{pipeline}} = \sum_{i=1}^{m} \Delta t_i + (n-1)\Delta t_{\max} = 50 \times 8 + 9 \times 50 = 850$$

$$\text{TP} = n/T_{\text{pipeline}} = 1/85$$

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

第三章 流水线技术课后作业

3.9 解： 措施 2：重复设置部件，如图 3.9 所示，其时空图如图 3.10 所示。

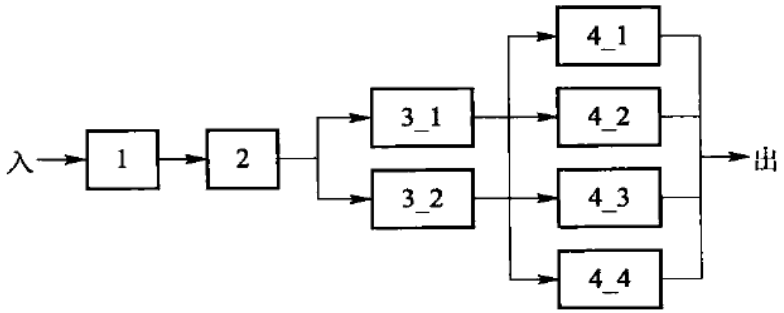

图 3.9 措施 2 的指令流水线

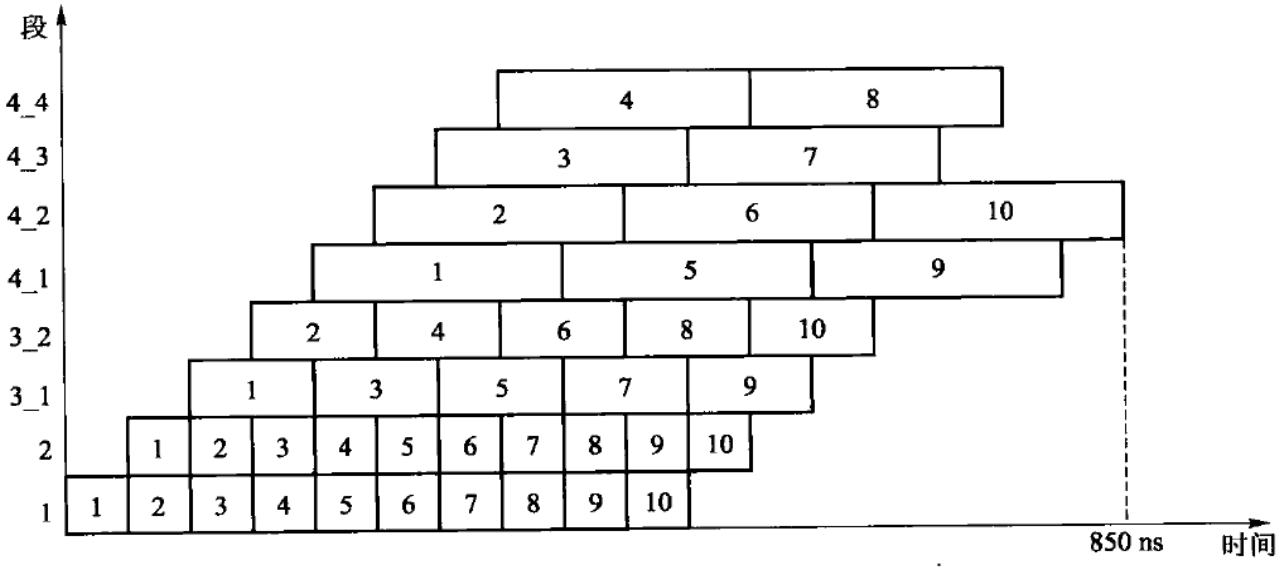

图 3.10 措施 2 时空图

$$TP = n/T_{pipeline} = 1/85$$

$$E = 400 \times 10/(850 \times 8) = 10/17 \approx 58.82\%$$

第三章 流水线技术课后作业

3.11 解：

首先，应选择适合于流水线工作的算法。对于本题，应先计算 A_1+B_1、A_2+B_2、A_3+B_3 和 A_4+B_4；再计算 $(A_1+B_1)\times(A_2+B_2)$ 和 $(A_3+B_3)\times(A_4+B_4)$；然后求总的结果。

其次，画出完成该计算的时空图，如图 3.15 所示，图中阴影部分表示该段在工作。

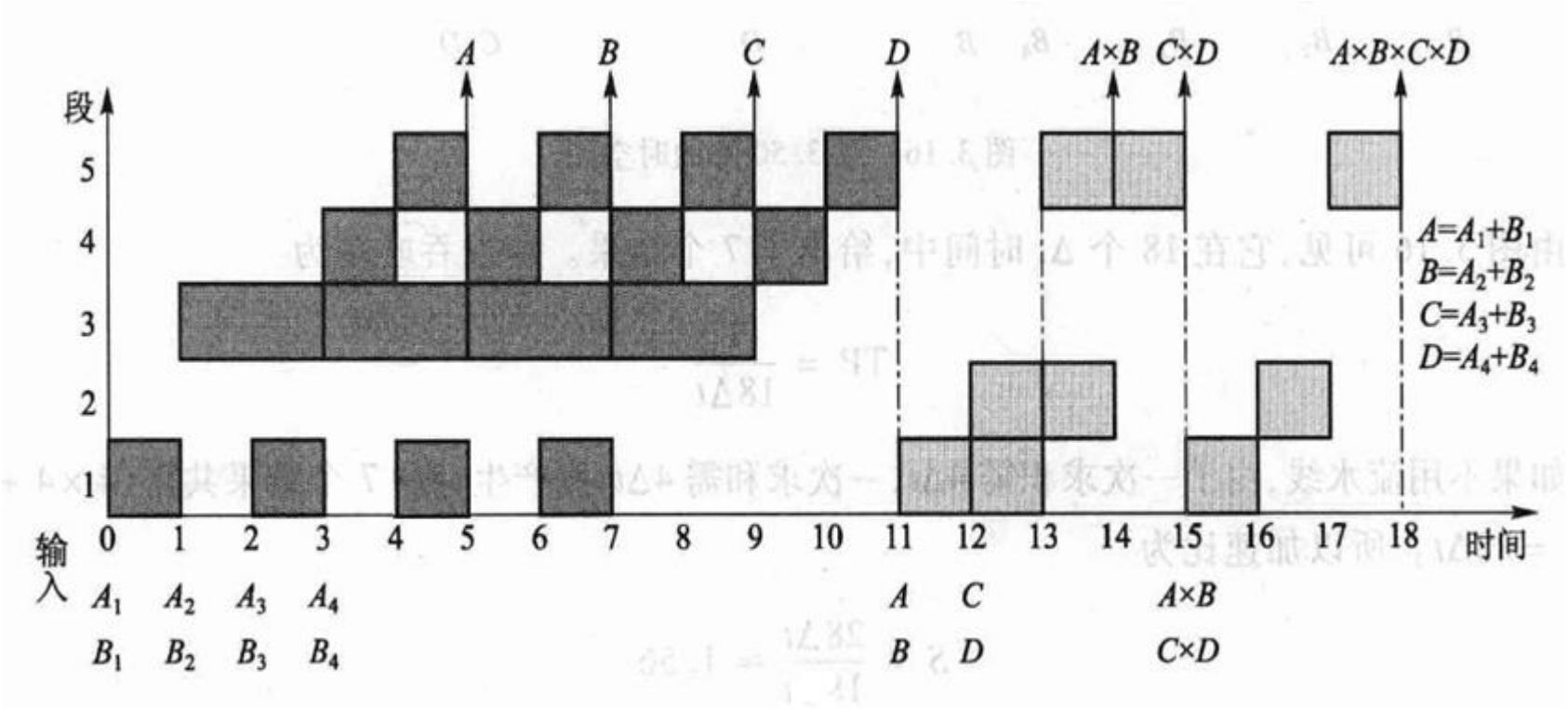

图 3.15 时空图

第三章 流水线技术课后作业

3.11 解：

由图 3.15 可见，它在 18 个 Δt 时间中，给出了 7 个结果。所以吞吐率为

$$TP = \frac{7}{18\Delta t}$$

如果不用流水线，由于一次求积需 $3\Delta t$，一次求和需 $5\Delta t$，则产生上述 7 个结果共需 $(4 \times 5 + 3 \times 3)\Delta t = 29\Delta t$。所以加速比为

$$S = \frac{29\Delta t}{18\Delta t} \approx 1.61$$

该流水线的效率可由阴影区的面积和 5 个段总时空区的面积的比值求得

$$E = \frac{4 \times 5 + 3 \times 3}{5 \times 18} \approx 0.322$$

第三章 流水线技术课后作业

3.12 解：

① 一共要做 10 次乘法，4 次加法。其时空图如图 3.17 所示。

②

$$TP = \frac{14}{22\Delta t}$$

$$加速比 = \frac{14 \times 4}{22\Delta t} \approx 2.55$$

$$效率 = \frac{14 \times 4}{22 \times 6} \approx 42.42\%$$

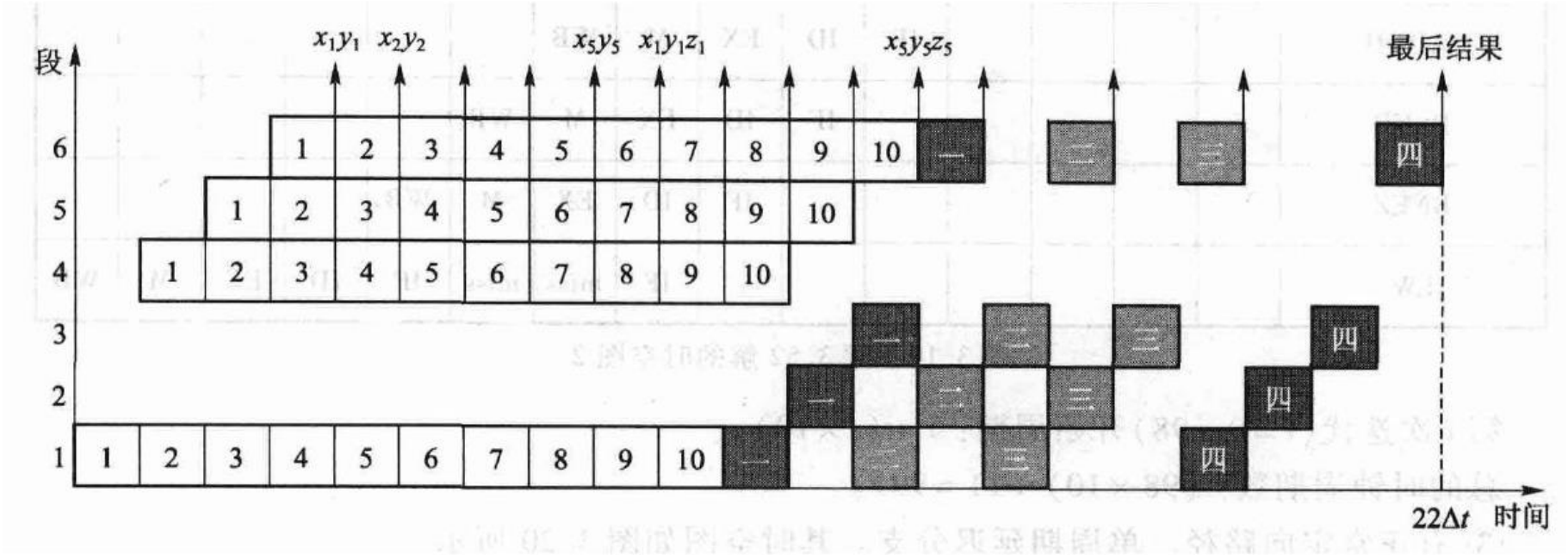

图 3.17 时空图

第三章 流水线技术课后作业

3.13 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。其时空图如图 3.18 所示。

指令	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
LW	IF	ID	EX	M	WB																	
DADDIU		IF	S	S	ID	EX	M	WB														
SW					IF	S	S	ID	EX	M	WB											
DADDIU								IF	ID	EX	M	WB										
DSUB									IF	S	S	ID	EX	M	WB							
BNEZ												IF	S	S	ID	EX	M	WB				
LW															IF	S	S	IF	ID	EX	M	WB

图 3.18 时空图 1

第 i 次迭代（$i=0\sim98$）开始周期：$1+(i\times17)$。

总的时钟周期数：$(98\times17)+18=1\,684$。

第三章 流水线技术课后作业

3.13 解：

② 有正常定向路径，预测分支失败。其时空图如图 3.19 所示。

指令	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LW	IF	ID	EX	M	WB										
DADDIU		IF	ID	S	EX	M	WB								
SW			IF	S	ID	EX	M	WB							
DADDIU					IF	ID	EX	M	WB						
DSUB						IF	ID	EX	M	WB					
BNEZ							IF	ID	EX	M	WB				
LW								IF	miss	miss	IF	ID	EX	M	WB

图 3.19 时空图 2

第 i 次迭代（$i=0\sim98$）开始周期：$1+(i\times10)$。

总的时钟周期数：$(98\times10)+11=991$。

第三章 流水线技术课后作业

③ 有正常定向路径。单周期延迟分支。其时空图如图 3.20 所示。

```
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。

指令	1	2	3	4	5	6	7	8	9	10	11
LW	IF	ID	EX	M	WB						
DADDIU		IF	ID	EX	M	WB					
DADDIU			IF	ID	EX	M	WB				
DSUB				IF	ID	EX	M	WB			
BNEZ					IF	ID	EX	M	WB		
SW						IF	ID	EX	M	WB	
LW							IF	ID	EX	M	WB

图 3.20 时空图 3

第四章 指令级并行课后作业

4.1（指令级并行、**指令的动态调度、指令的静态调度**、动态分支预测技术、循环展开），4.2，**4.3**，4.4，4.5，**4.7**，4.9

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。

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，减少数据相关导致的停顿。

指令的静态调度——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。

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，预测结果也随之改变。

循环展开：是一种增加指令间并行性最简单和最常用的方法。它将循环展开若干遍后，通过重命名和指令调度来开发更多的并行性。

第四章 指令级并行课后作业

4.2 解：

其核心思想如下。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小。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。

其基本思想如下。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，而不是从寄存器中去取。指令的执行结果也是直接送到等待数据的其他保留站中去。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对应于该寄存器保留站的名称(编号)。

4.3 解：

将循环展开两次，进行指令调度，即可以消除延迟，代码如下：

```
LOOP:  L.D     F0，0（R1）
       L.D     F10，-8（R1）
       MUL.D   F0，F0，F2
       MUL.D   F10，F10，F2
       L.D     F4，0（R2）
       L.D     F14，-8（R2）
       ADD.D   F0，F0，F4
       ADD.D   F10，F10，F14
       DSUBI   R1，R1，16
       S.D     0（R2），F0
       DSUBI   R2，R2，16
       BNEZ    R1，LOOP
       S.D     8（R2），F10
```

4.4 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。

第四章 指令级并行课后作业

4.5 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 $= 1 + 2 \times 5\% \times (1 - 90\%) = 1.01$

4.7 解（1）（答案不唯一，只要满足题目条件就可以）

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)	

第四章 指令级并行课后作业

4.7 解（2）（答案不唯一，只要满足题目条件就可以）

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)	

4.7 解（3）（答案不唯一，只要满足题目条件就可以）

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)		

4.8 解：

解：展开 7 遍循环

访存指令 1	访存指令 2	浮点指令 1	浮点指令 2	整数/转移指令
L.D F0,0（R1）	L.D F6,-8（R1）			
L.D F10,-16（R1）	L.D F14,-24（R1）			
L.D F18,-32（R1）	L.D F22,-40（R1）	ADD.D F4,F0,F2	ADD.D F8,F6,F2	
L.D F26,-48（R1）		ADD.D F12,F10,F2	ADD.D F16,F14,F2	
		ADD.D F20,F18,F2	ADD.D F24,F22,F2	
S.D F4,0（R1）	S.D F8,-8（R1）	ADD.D F28,F26,F2		
S.D F12,-16（R1）	S.D F16,–24（R1）			
S.D F20,-32（R1）	S.D F24,–40（R1）			DADDIU R1,R1,# -56
S.D F28,8（R1）				BNE R1,Loop

这段程序的运行时间为 9 个时钟周期，每遍循环平均约 1.28 个时钟周期。9 个时钟周期内流出了 23 条指令，每个时钟周期 2.55 条。9 个时钟周期共有操作槽 9×5=45 个，有效槽的比例为 51.1%。

第五章 存储系统课后作业

5.1（多级存储层次、全相联映像、直接映像、组相联映像、替换算法、TLB），5.3，5.5，**5.9（教材例5.3）**，**5.10**，5.13

多级存储层次——由一组采用不同技术实现的存储器构成的一个存储系统。在存储层次中，各存储器之间一般满足包容关系，即任何一层存储器中的内容都是其下一层（离 CPU 更远的一层）存储器中内容的子集。其目标是：从 CPU 看，该存储系统的速度接近于离 CPU 最近的存储器的速度，而容量接近于离 CPU 最远的存储器的容量。

全相联映像——主存中的任一块可以被放置到 Cache 中任意一个地方。

直接映像——主存中的每一块只能被放置到 Cache 中唯一的一个地方。

组相联映像——将 Cache 分成若干个组，每组由若干块构成。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。

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，会出现该块所映像到的一组（或一个）Cache 块已全部被占用的情况。这时，需被迫腾出其中的某一块，以接纳新调入的块。

TLB——一个专用高速缓冲器，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。

第五章 存储系统课后作业

5.3 解：

① 随机法。简单、易于用硬件实现，但这种方法没有考虑 Cache 块过去被使用的情况，反映不了程序的局部性，所以其失效率比 LRU 的高。

② 先进先出法。容易实现。它虽然利用了同一组中各块进入 Cache 的顺序这一“历史”信息，但还是不能正确地反映程序的局部性。

③ 最近最少使用法(LRU)。失效率最低。但是 LRU 比较复杂，硬件实现比较困难。

5.5 解：

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，程序不是按数据在存储器中的顺序访问。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顺序进行访问。

③ 循环融合。有些程序含有几部分独立的程序段，它们用相同的循环访问同样的数组，对相同的数据做不同的运算。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。

第五章 存储系统课后作业

5.9 解：

第一级 Cache 的失效率(全局和局部)是 110/3000,即约 3.67%。

第二级 Cache 的局部失效率是 55/110,即 50%,第二级 Cache 的全局失效率是 55/3000,即约1.83%。

5.10 解：

$$\text{平均访问时间} = \text{命中时间} + \text{失效率} \times \text{失效开销}$$

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

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

$$\text{访问速度比} = 3.12/3 = 1.04$$

2 路组相联映像的平均访问时间比较小。

$$\text{CPU}_{\text{时间}} = (\text{CPU}_{\text{执行}} + \text{存储等待周期}) \times \text{时钟周期}$$

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

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

$$\text{CPU}_{\text{时间-1路}} = \text{IC}(2.0 \times 2 + 1.2 \times 0.014 \times 80) = 5.344\text{IC}$$

$$\text{CPU}_{\text{时间-2路}} = \text{IC}(2.2 \times 2 + 1.2 \times 0.01 \times 80) = 5.36\text{IC}$$

$$\text{相对性能比} = \frac{\text{CPU}_{\text{时间-2路}}}{\text{CPU}_{\text{时间-1路}}} = 5.36/5.344 \approx 1.003$$

直接映像 Cache 的访问速度比 2 路组相联映像 Cache 要快 1.04 倍，而 2 路组相联映像 Cache 的平均性能比直接映像 Cache 要高 1.003 倍。因此这里选择 2 路组相联映像。

第五章 存储系统课后作业

5.13 解：

① 提高 Cache 的相联度，可以减少冲突失效的次数，但不会影响强制性失效和容量失效的次数。

已知 Cache 的失效率为 7%，程序共访问存储器 1 000 000 次，所以总的失效次数为 70 000，其中 50% 为冲突失效的次数。

因此，提高 Cache 的相联度能够消除的最大失效次数是 70 000 × 50% = 35 000。

② 当同时提高 Cache 的容量大小和相联度时，可以消除容量失效和冲突失效的次数。而这两种失效占总失效次数的 75%，所以，能够消除的最大失效次数是 70 000 × 75% = 52 500。

第六章 输入/输出系统课后作业

6.1（RAID、分离事务总线、通道、通道流量、虚拟DMA），**6.3**，**6.4**，6.6，6.8

分离事务总线——将总线事务分成请求和应答两部分。在请求和应答之间的空闲时间内，总线可以供给其他 I/O 使用。采用这种技术的总线称为分离事务总线。

通道——专门负责整个计算机系统输入/输出工作的专用处理机，能执行有限的一组输入/输出指令。

通道流量——指一个通道在数据传送期间，单位时间内能够传送的数据量。

虚拟 DMA——它允许 DMA 设备直接使用虚拟地址，并在 DMA 传送过程中由硬件将虚拟地址转换为物理地址。

6.3 解：

① 同步总线。同步总线上所有设备通过统一的总线系统时钟进行同步。同步总线的优点是成本低，因为它不需要设备之间互相确定时序逻辑。但是同步总线也有缺点，总线操作必须以相同的速度运行。

② 异步总线。异步总线上的设备之间没有统一的系统时钟，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来控制。异步总线的优点是适用于更广泛的设备类型，扩充总线时不必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。

6.4 （分别说明三种通道的在数据传输上的特点） 解：

（1）字节多路通道：用于连接多台低速或中速设备，这些设备一般是以字节为宽度进行输入/输出，相邻两次传送之间有较长时间等待。

（2）选择通道：在一段时间内只被一台高速外设独占使用，直到该设备数据传输工作全部完成，再重新选择设备。

（3）数组多路通道：以数据块为单位，分时轮流地为多台高速设备提供服务。

6.6 解：

① 通道实际流量为

$$f_{BYTE} = \sum_{i=1}^{6} f_i \approx 50 + 50 + 40 + 25 + 25 + 10 = 200 \text{ B/ms}$$

② 由于通道的最大流量等于实际工作流量，即有

$$f_{max\text{-}BYTE} = \frac{1}{T_S + T_D} = 200 \text{ B/ms}$$

可得，通道的工作周期 $T_S + T_D = 5\ \mu s$。

6.8 解：① 如果按字节多路通道设计，该通道的最大流量为

$$f_{\text{max-BYTE}} = \frac{1}{T_S + T_D} = 250 \text{ KBps}$$

连接到字节多路通道上的 p 台设备能正常工作的条件是

$$f_{\text{max-BYTE}} \geqslant \sum_{i=1}^{p} f_i$$

故应选择外设 3、4、5、7 和 8 同时连接到通道上，因为

$$\sum_{i=1}^{5} f_i = 100 + 75 + 50 + 14 + 10 = 249 \text{ KBps} < 250 \text{ KBps}$$

② 如果按数组多路通道设计，该通道的最大流量为

$$f_{\text{max-BLOCK}} = \frac{k}{T_S + kT_D} = \frac{512}{(1 + 512) \times 2 \times 10^{-6}} = 0.499 \times 10^6 \text{ Bps} = 499 \text{ KBps}$$

连接到数组多路通道上的设备能正常工作的条件是

$$f_{\text{max-BLOCK}} \geqslant f_i$$

因此，除外设 1 外，其他外设都可以同时连接到通道上。

第八章 多处理机课后作业

8.1（集中式共享多处理机、分布式共享多处理机、多Cache一致性、同时多线程），8.2，**8.3**，8.6

集中式共享多处理机——也称为对称式共享存储器多处理机 SMP。它一般由几十个处理器构成，各处理器共享一个集中式的物理存储器，这个存储器相对于各处理器的关系是对称的。

分布式共享多处理机——它的共享存储器分布在各台处理机中，每台处理机都带有自己的本地存储器，组成一个“处理机－存储器”单元。但是这些分布在各台处理机中的实际存储器又合在一起统一编址，在逻辑上组成一个共享存储器。这些处理机存储器单元通过互连网络连接在一起，每台处理机除了能访问本地存储器外，还能通过互连网络直接访问在其他处理机存储器单元中的“远程存储器”。

多处理机 Cache 一致性——在多处理机中，当共享数据进入 Cache 后，可能出现多个处理器的 Cache 中都有同一存储器块的副本的情况，当其中某个处理器对其 Cache 中的数据进行修改后，就会使其 Cache 中的数据与其他 Cache 中的数据不一致。

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。

第八章 多处理机课后作业

8.2 解：

- 共享存储器通信的主要优点
  - 与常用的对称式多处理机使用的通信机制兼容。
  - 易于编程，同时在简化编译器设计方面也占有优势。
  - 采用大家所熟悉的共享存储器模型开发应用程序，而把重点放到解决对性能影响较大的数据访问上。
  - 当通信数据量较小时，通信开销较低，带宽利用较好。
  - 可以通过采用Cache技术来减少远程通信的频度，减少了通信延迟以及对共享数据的访问冲突。
- 消息传递通信机制的主要优点
  - 硬件较简单。
  - 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。
  - 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。
  - 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。

8.3 解：

对多个处理器维护一致性的协议称为多处理器 Cache 一致性协议。

目录协议的工作原理：采用一个集中的数据结构——目录。对于存储器中的每一个可以调入 Cache 的数据块，在目录中设置一个目录项，用于记录该块的状态以及哪些 Cache 中有副本等相关信息。目录协议根据该项目中的信息以及当前要进行的访问操作，依次对相应的 Cache 发送控制消息，并完成对目录项信息的修改。此外，还要向请求处理器发送响应信息。

监听协议的工作原理：每个 Cache 除了包含物理存储器中块的数据副本之外，也保存着各个块的共享状态信息。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上广播出去，其他各个 Cache 控制器通过监听总线来判断它们是否有总线上请求的数据块。如果有，就进行相应的操作。

第八章 多处理机课后作业

8.6 解：

已知远程访问率 $p=0.5\%$，远程访问时间 $t=2\ 000$ ns，时钟周期 $T=10$ ns，远程访问开销 $C=t/T=2\ 000\ \text{ns}/10\ \text{ns}=200$（时钟周期数）。

没有远程访问的机器的基本 $\text{CPI}_1=1.0$。

有 0.5% 远程访问的机器的实际 CPI_2 为

$$\text{CPI}_2 = \text{CPI}_1 + pC = 1.0 + 0.5\% \times 200 = 2.0$$

则 $\text{CPI}_2/\ \text{CPI}_1=2.0/1.0=2$（倍）。

因此，没有远程访问状态下的机器速度是有 0.5% 远程访问的机器速度的 2 倍。

8.7 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 5 500 个时钟周期。

事　　件	持续时间
所有处理器都读(命中)锁	0
释放锁的处理器进行写(失效)广播	100
读(命中)锁(处理器认为锁是空闲的)	0
一个处理器进行写交换广播，同时还有 9 个写广播	1 000
一个处理器得到并释放锁的总时间	1 100