第 3 章 信息加工的流水线技术

流水线技术的应用极其广泛，随处可见，如工业领域、管理工作等，在计算机体系结构中也得到普遍应用。本章以用于信息加工的处理机流水线为基础，介绍流水线及其表示方法与特点分类，阐明流水线处理机实现的基本结构，分析线性流水线的性能与非线性流水线的调度策略，讨论流水线的相关及其处理方法。

3.1 流水线及其特点与分类

【问题小贴士】 在工作和生活中，常常听到流水线这个词，也不知不觉中应用流水线技术。如新生入学报到就采用了流水线技术，学校构建新生入学报到流水线分为两步：①将新生入学报到工作分解并按某种时序将它们线性排列：打印报到流程表→办理缴费或助学贷款→到院系报名（含领校园卡、登记信息、递交材料等）→购买或领取日常用品；②新生入学报到的各项子工作由不同部门指派人去负责完成。这时学生报到需要经过多人多处，每处的子工作是专业化的，相对于报到整体工作要简单得多。与采用集中一处报到相比，其最大优势为学生报到的速度快，即单位时间内可以有更多学生完成报到。①根据其他课程讲述的“指令处理过程”和时间重叠概念，当处理机处理指令序列（运行程序）时，是否也可以采用流水线技术呢？如果可以，如何构建？这样能够带来什么好处，它们是怎样形成的？怎样定义流水线呢？②根据浮点加减法运算步骤，浮点加减法运算是否可以采用流水线来执行呢？由此，有哪些不同特性的流水线呢？③对于新生入学报到流水线，大部分学生直接刷卡缴费，少数学生需要助学贷款，这时会带来什么问题？由此，怎样使流水线速度快、效率高，并且如何充分发挥流水线效率呢？④为了便于流水线技术的应用，自然需要采用一定的方法来描述它。那么，流水线有哪些表示方法？各种方法有什么优势和适用性？

3.1.1 指令序列处理及其流水线概念

1. 单条指令的处理过程

单条指令的处理过程可以分为若干个阶段，最简单、直观地可以分为取指令、分析指令和执行指令三个阶段，其处理过程如图 3-1 所示。取指令阶段的任务为：按照程序计数器的地址内容访问主存储器，读出一条指令（二进制代码）并送到指令寄存器。分析指令阶段的任务为：对指令寄存器中的操作码字段进行译码，分析指令的功能操作；分析地址码字段中的寻址方式，并进行形式地址变换，生成操作数存储单元地址（立即寻址除

外），并获取操作数；同时程序计数器自动产生一个增量，形成下一条指令的主存单元地址。执行指令阶段的任务为：根据功能操作要求，对操作数进行运算操作，并把结果送到指定的地址中。

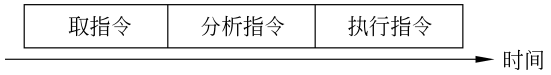

图 3-1　单条指令的处理过程

2. 指令序列的处理方式

当在同一个处理器中处理一段程序的指令序列时，若假设三个阶段所花费的时间均为 Δt，则可以采用顺序、一次重叠和二次重叠三种处理方式。

1）顺序处理

顺序处理是指在任何时刻处理器至多仅能处理一条指令，指令之间是完全串行处理的，即第 k 条指令处理结束后，再处理第 $k+1$ 条指令，以此类推。顺序处理过程如图 3-2 所示，若程序包含 N 条指令，程序运行所需要的时间为：$T=3N\Delta t$。

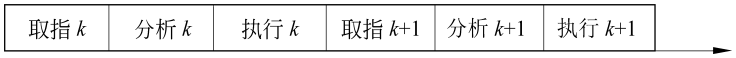

图 3-2　指令序列的顺序处理过程

顺序处理的优点在于控制简单、实现成本较低。而主要缺点有两个：一是指令处理的速度慢，指令之间是完全串行处理；二是功能部件的利用率低，如取指令时主存是忙碌的，处理器则是空闲的。

2）一次重叠处理

一次重叠处理是指在任何时刻处理器至多仅能处理两条指令，指令之间可能有两条指令在处理，即第 k 条指令的执行与第 $k+1$ 条指令的取指令同时发生。一次重叠处理过程如图 3-3 所示，若程序包含 N 条指令，程序运行所需要的时间为：$T=(2N+1)\Delta t$。

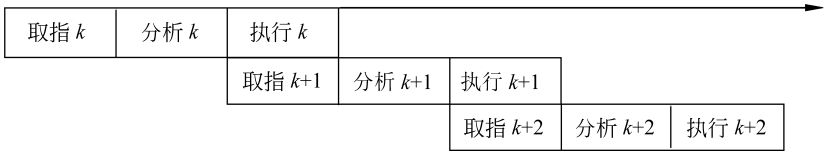

图 3-3　指令序列的一次重叠处理过程

一次重叠处理与顺序处理相比，主要优点有两个：一是处理 N 条指令所需要的时间缩短近 1/2；二是功能部件的利用率明显提高，如主存基本处于忙碌状态。但一次重叠处理的实现需要增加一些硬件，而付出了一定的代价，过程控制也变得复杂些。如为了在第 k 条指令执行的同时可以取第 $k+1$ 条指令，必须增加一个指令寄存器，原来的指令寄存器用于存放第 k 条指令，新增的指令寄存器用于存放第 $k+1$ 条指令。

3）二次重叠处理

二次重叠处理是指在任何时刻处理器至多仅能处理三条指令，指令之间可能有三条

指令在处理，即第 k-1 条指令的执行与第 k 条指令的分析以及第 $k+1$ 条指令的取指令同时发生。二次重叠处理过程如图 3-4 所示，若程序包含 N 条指令，程序运行所需要的时间为：$T=(N+2)\Delta t$。

图 3-4　指令序列的二次重叠处理过程

二次重叠处理进一步提高了指令序列的执行速度，相对于顺序处理，处理 N 条指令所需要的时间缩短近 2/3，但过程控制更加复杂，也需要付出更高代价。

可见，采用重叠方式处理指令序列，使程序运行如同工业生产流水线一样，源源不断地处理指令，有效地提高了指令序列的处理速度。当然，也带来许多问题，如指令处理过程各阶段不相等、指令之间的相关性等，这就需要先行控制、相关处理等新技术的支持。

3. 流水线及其优势实现基础

流水线在工业生产中随处可见。如汽车装配流水线，把汽车装配分为多道工序（子过程），每道工序的任务由一人或多人完成，各道工序所花费时间也大致相等。当汽车装配流水线启动之后，每隔一定的时间（一道工序的时间）就有一辆汽车下线。如果跟踪一辆汽车装配的全过程，就会发现每辆汽车装配都经过了每道工序，装配总时间并没有减少，但由于多辆汽车在时间上重叠装配，使得单位时间内有更多的汽车下线，装配速度得到极大提高。计算机中的流水线与工业生产中的流水线十分相似。

一般说来，流水线技术是指把一个重复的任务处理过程分解为若干个子过程，当每个子过程均设置一个功能部件来实现时，一个过程的子过程可以与其他过程的不同子过程同时进行，实现多个不同过程在时间上重叠工作。将任务处理过程所包含的所有子过程或实现子过程的所有功能部件按一定次序连接在一起，则称为流水线（Pipelining）。任务从流水线的一端进入，经过流水线上的功能部件，从流水线的另一端排出。

从本质上讲，流水线技术是一种时间并行技术，是通过时间重叠的技术途径实现并行处理（时间并发性），换句话说，流水线是通过多个小功能部件并行工作来提高处理速度。把一个重复的任务处理过程分解为若干个子过程，由此为每个子过程设置一个对应独立的功能部件来实现，这些小功能部件是由大功能部件分解而来的，使它们并行工作就可以提高任务的处理速度。

流水线中的子过程或功能部件称为流水段或功能段，也可以称为流水节拍、流水步骤等，流水线中的功能段数量称为流水线的深度。流水线技术是一种非常经济而有效的技术，已成为计算机中普遍应用的一种并行处理技术。采用流水线技术只需要增加少量的硬件，就可以把处理器的运算速度提高许多倍。

3.1.2　流水线的表示

流水线通常有三种表示方法：连接图、时空图和预约表。其中时空图用于表示线性

流水线；预约表用于表示非线性流水线；而连接图既可以表示线性流水线，也可以表示非线性流水线。现假设一条指令的处理过程分为取指令、译码、执行、保存结果 4 个子过程，相应处理指令的流水线则包含取指令、译码、执行、保存结果 4 个功能段，且对于某些指令，执行功能段还需要重复使用。

1. 连接图表示法

所谓连接图实质是将带执行时间标记的各功能段按照任务在功能段上的执行顺序从左到右排列，并用带箭头的直线把它们连接起来。上述包含 4 个功能段的线性指令流水线的连接图如图 3-5 所示，而执行功能段还需要重复使用的非线性指令流水线的连接图如图 3-6 所示。显然，通过连接图可以看出流水线所包含功能段的结构关系和任务在流水线上的处理顺序；反过来，分析任务处理过程并划分出过程阶段是画出流水线连接图的基础。

图 3-5　线性指令流水线的连接图

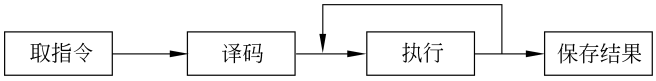

图 3-6　非线性指令流水线的连接图

2. 时空图表示法

所谓时空图实质是利用平面直角坐标的第一象限，以横坐标为时间、纵坐标为空间（功能段），由此在一定范围的平面区域内标记由流水线处理的任务编号，以表示该区域对应的时间段通过对应的功能段来处理对应编号的任务。上述包含 4 个功能段的线性指令流水线的时空图如图 3-7 所示，且当流水线中各功能段执行时间相等时，横坐标被分割成长度相等的时间段。时空图是描述线性流水线最常用、最直观有效的表示方法，横坐标表示输入流水线的任务在流水线中所经历的时间，纵坐标表示输入流水线的任务在流水线中所经过的功能段。

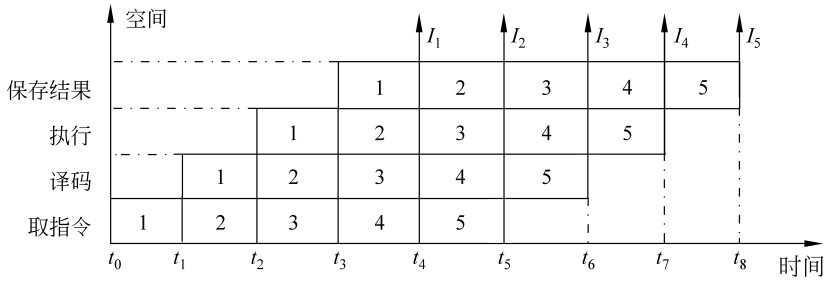

图 3-7　线性指令流水线的时空图

通过时空图可以看出任务在任一时刻所处的处理状态，以及同一时间段有哪些功能段在处理任务。如从图 3-7 的横坐标来看，5 条指令依次分别在 t_0、t_1、t_2、t_3、t_4 时刻进入

流水线，在 t_4、t_5、t_6、t_7、t_8 时刻排出流水线；在 t_4 时刻以前，每隔一个时间段就有一条指令进入流水线，从 t_4 时刻开始，每隔一个时间段就有一条指令排出流水线。另外，从图 3-7 的纵坐标来看，在 $t_4 \sim t_5$ 时间段，取指令、译码、执行、保存结果 4 个功能段分别在处理第 5、4、3、2 条指令的对应子过程。反过来，流水线连接图是建立流水线时空图的基础。

3. 预约表表示法

所谓预约表实质是利用一张表，其中行为空间（功能段）、列为时间，当仅描述一个任务在流水线上的处理过程时，则在某个单元格内标记“×”，以表示该单元格对应功能段在对应时间段被该任务占用；当描述序列任务在流水线上的处理过程时，则在某个单元格内标记“$\times_j$”，以表示该单元格对应的时间段通过对应的功能段来处理对应下标编号的任务。上述包含 4 个功能段的非线性指令流水线的单指令预约表如图 3-8 所示，序列指令预约表见 3.5 节。预约表是描述非线性流水线最常用、最直观有效的表示方法，行表示输入流水线的任务在流水线中所经历的时间，列表示输入流水线的任务在流水线中所经过的功能段。

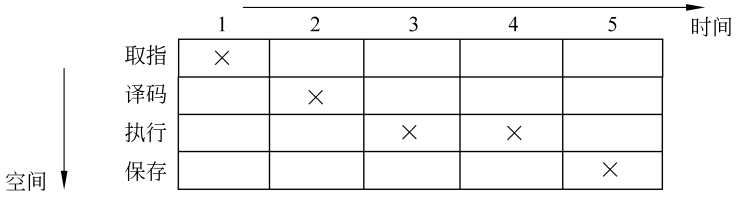

图 3-8　非线性指令流水线的单指令预约表

通过单任务预约表可以看出，任务处理需要多少个时间段，以及在任务处理过程中哪些功能段被重复使用。从图 3-8 来看，预约表有 5 列，则指令处理需要 5 个时间段，且执行功能段在第 3、4 时间段被重复使用。特别地，有的单任务预约表可能同一列有两个单元格带“×”，表明在同一时间段，任务可能进行不同处理，如分支指令，由于条件不同，分支后的时间段进行的是不同的操作。

3.1.3　流水线的分类

流水线种类很多，从不同的角度可以将流水线分为若干不同类型，以反映流水线在某方面的结构、特点或性能等。

1. 按流水线功能多寡来分类

按照功能多寡来分，可以将流水线分为单功能流水线和多功能流水线两种。

1）单功能流水线

单功能流水线（Unifunction Pipelining）是指流水线各功能段之间的连接固定不变，仅能够用于处理功能特性相同的任务。如流水线浮点加法器专门用于浮点加法运算，流水线浮点乘法器专门用于浮点乘法运算，如图 3-9 所示的是浮点加法器的流水线连接图，其所包含的 6 个功能段之间以固定形式连接在一起，实现唯一的浮点加法运算。

2）多功能流水线

多功能流水线（Multifunction Pipelining）是指流水线各功能段之间的连接可以改变，

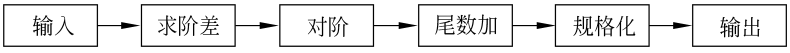

图 3-9　浮点加法器的线性单功能流水线的连接图

在不同时间或同一时间内，通过不同的连接可以处理不同功能特性的任务。典型的多功能流水线是 texas 仪器公司的高级科学计算机 ASC，其处理器的流水线包含 8 个功能段，依次为输入、求阶差、对阶、尾数加、规格化、尾数乘、累加和输出。在一台 ASC 处理器内有 4 条相同的流水线，每条流水线都可以通过不同连接来分别实现整数加减法运算、整数乘法运算、浮点加法运算、浮点乘法运算和逻辑运算、移位操作、数据转换等。如图 3-10 所示的是 ASC 计算机中处理器流水线的部分连接图，功能段之间可以建立不同的连接来实现不同的运算。另外，ASC 计算机除支持标量运算之外，还支持向量运算，可以构建非线性流水线，图 3-10(c)所示的是对两个浮点向量求点积的流水线连接，它就是一个带有反馈回路的非线性流水线。

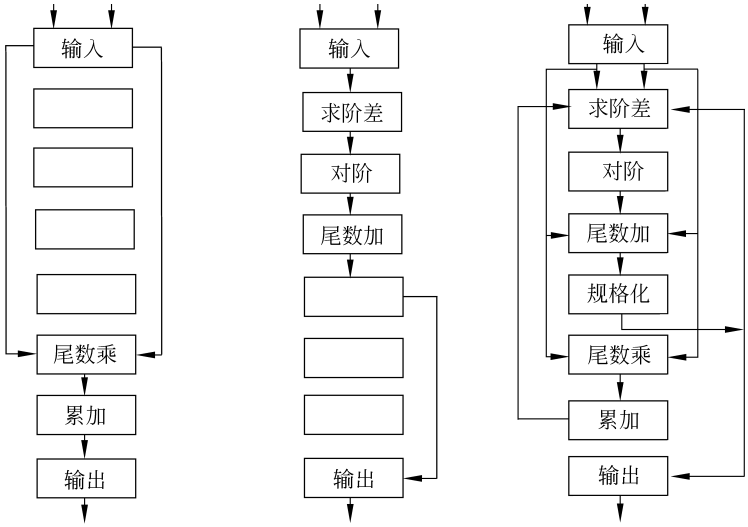

图 3-10　TI-ASC 计算机处理器的多功能流水线的连接图

2. 多功能流水线按在同一段时间内的功能段间连接是否可变来分类

对于多功能流水线，按照在同一段时间内是否可以实现多种连接以同时处理多种不同功能的任务，可将其分为静态流水线和动态流水线两种。

1) 静态流水线

静态流水线(Static Pipelining)是指在同一段时间内，多功能流水线功能段之间仅能够按照一种方式连接，并且处理相同功能特性的任务。对于静态流水线，只有当按照这种连接流入的所有任务都流出之后，多功能流水线才能改变连接以处理其他功能特性的任务。如图 3-10 所示的 8 段多功能流水线，如果按照图 3-11 所示的时空图处理任务，那么它就是一种静态流水线。由图 3-11 可看出，开始时流水线按照浮点加减运算实现来连接，当 N 个浮点加减运算全部处理结束即第 N 个浮点加减运算排出之后，多功能流水线

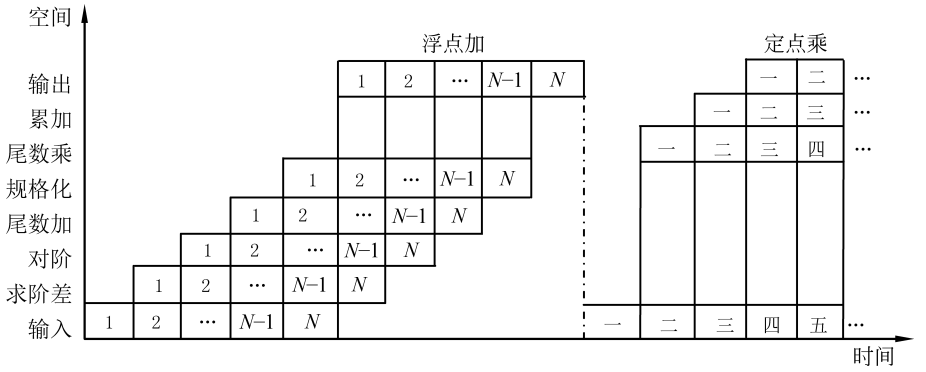

图 3-11　TI-ASC 计算机处理器的静态流水线的时空图

才改变为按照定点乘运算实现来连接。

2）动态流水线

动态流水线(Dynamic Pipelining)是指在同一段时间内，多功能流水线的功能段之间可以按照多种方式连接，从而可以处理不同功能特性的任务，当然任何一个功能段仅能参与到一种连接中。如图 3-10 所示的 8 段多功能流水线，如果按照图 3-12 所示的时空图处理任务，那么它就是一种动态流水线。由图 3-12 可以看出，浮点加减运算还未处理结束排空，流水线就已按照定点乘运算实现进行连接，并开始定点乘法运算。所以，在同一段时间内和同一条多功能流水线上，两种运算分别使用不同的功能段来同时实现。

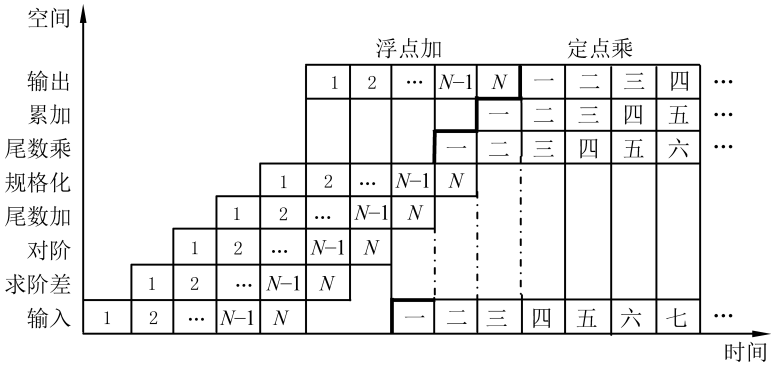

图 3-12　TI-ASC 计算机处理器的动态流水线的时空图

特别地，对于静态流水线，只有连续向其输入功能特性相同的任务，效率才能得到充分的发挥。如果连续输入静态流水线的是功能特性不同的任务，如输入的一串操作是浮点加、定点乘、浮点加、定点乘……静态流水线将如同顺序处理方式一样，效率得不到发挥。而对于动态流水线，由于其允许两种功能特性不同的任务同时进行处理，因此动态流水线的效率高于静态流水线，但动态流水线的控制极其复杂，目前大多数处理器均采用静态流水线。

3. 按流水线的级别来分类

按照级别来分，可将流水线分为功能部件级、处理机级和处理机之间级三种。

1）指令流水线

指令流水线(Instruction Pipelining)又称为处理机级流水线，它是把指令处理过程分解为多个子过程，每个子过程在独立的功能部件中执行，使指令处理实现流水线化。如图 3-5和图 3-6 所示的 4 功能段指令流水线即是将指令处理过程分为取指令、译码、执行、保存结果 4 个子过程的指令流水线。

2）运算操作流水线

运算操作流水线(Arithmetic Pipelining)又称为功能部件级流水线，它是把运算分解为多个子过程，每个子过程在独立的部件中执行，使运算操作实现流水线化。对于较复杂的运算往往采用流水线来实现，如浮点加运算与浮点乘运算等。图 3-9 所示的 6 功能段线性单功能流水线即是浮点加的运算操作流水线；图 3-10 所示的多功能流水线即是多功能运算操作流水线。

3）宏流水线

宏流水线(Macro Pipelining)又称为处理机之间级流水线，它是由两台或以上的处理机通过存储器串行连接起来，每台处理机分别对任务中的一部分工作进行处理。如图 3-13 所示的是宏流水线，在逻辑上将前一台处理机的输出存入存储器中，作为后一台处理机的输入，这样同一数据流的不同部分或不同变换将分别在不同的处理机上实现。

图 3-13　宏流水线基本结构

4. 按流水线功能段之间是否存在反馈来分类

按照流水线功能段之间是否存在反馈来分，可以把流水线分为线性流水线和非线性流水线两种。

1）线性流水线

线性流水线(Linear Pipelining)是指流水线各功能段之间串行连接，数据按顺序流经流水线各功能段一次，且仅流经一次。图 3-5 所示的指令流水线和图 3-9 所示的运算操作流水线均为线性流水线。

2）非线性流水线

非线性流水线(Nonlinear Pipelining)是指在流水线各功能段之间除串行连接外，还存在反馈回路，任务可以多次流经存在反馈回路的功能段。图 3-6 所示的指令流水线是非线性流水线，其中功能段 S_3 存在反馈回路，表示 S_3 可以被多次使用；图 3-10 的浮点点积运算流水线也是非线性流水线。

另外，流水线还可以按照控制方式分为顺序流水线和乱序流水线、同步流水线和异步流水线，按照数据表示方式分为标量流水线和向量流水线。

3.1.4　流水线的特点

在处理机中，采用流水线方式处理指令与采用串行方式相比具有许多特点。在处理机与程序的设计过程中，应充分注意这些特点，以设计出高效率的处理机流水线并充分发

挥其效率。

(1) 流水线各功能段延迟时间应尽量相等才能充分发挥效率。流水线中任务的流动节拍不能快于延迟时间最长、执行速度最慢的功能段,该功能段是流水线的“瓶颈”,将引起流水线“堵塞”与“断流”,使得其他功能段将不能充分发挥其效率。

(2) 流水线必然存在不能充分发挥效率的“装入时间”与“排空时间”。“装入时间”是指第一个任务进入流水线到排出流水线的时间,“排空时间”是指最后进入流水线的任务到其排出流水线的时间。在这两个时间段内,有功能段空闲,流水线没有充满;只有流水线完全充满后,流水线的效率才能得到充分发挥。

(3) 必须连续不断地为静态流水线提供同种任务才能充分发挥效率。如为使流水线化的浮点加法器充分发挥效率,就需要连续提供浮点加法运算。但由于程序本身的原因和程序设计过程中人为的原因,如数据相关等,不可能连续为浮点加法器提供同种运算。因此,对于流水线处理机,特别是当流水线级数较多时,应该在软件和硬件等多方面尽量为流水线提供连续的任务。

(4) 在流水线每个功能段的后面应设置一个缓冲寄存器(又称锁存器、闸门寄存器等),以平滑各功能段的延迟时间。在流水线中,每个功能段的延迟时间不可能绝对相等,另外还存在电路的延迟时间与时钟偏移等,因此在功能段之间传送任务时,必须通过锁存器来保存功能段自身的执行结果,如图 3-14 所示。

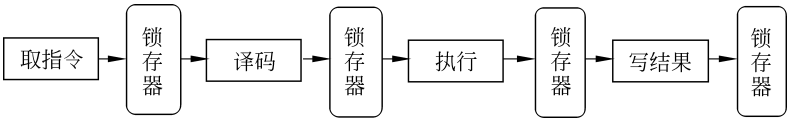

图 3-14 在流水线功能段后面设置的锁存器

显然,在指令流水线中加入锁存器之后,每条指令的实际处理时间增加了。尽管如此,采用流水线后通过多条指令并行处理可以使整个程序的运行时间缩短,但并没有真正减少每条指令的处理时间,这就限制了流水线的深度。一旦时钟周期很小,与时钟偏移和锁存器的附加开销相当时,流水线就失去了作用,在一个时钟周期内没有足够的时间用于有效工作。

3.2 流水线处理机的实现结构

【问题小贴士】 ①流水线是通过多个小功能部件并行工作来提高处理速度,这些小功能部件是由串行处理指令序列的大功能部件分解而来的。那么,当把指令处理过程分解为取指令、分析、执行三个阶段时,处理机应该具备怎样的结构才能实现多条指令的并行处理?为什么?②当多条指令并行处理时,为什么处理机对主存储器的访问会产生冲突?这时会影响处理机的工作效率吗?为什么?③针对流水线功能段的延迟时间不相等的问题,提出了先行控制技术,那么该技术可以解决哪些问题?是如何解决的?为了实现先行控制技术,处理机应该具备怎样的结构?这时,指令处理过程可以分为哪些阶段?

3.2.1　重叠处理的实现结构及其访问冲突

1. 重叠处理的实现结构

当采用二次重叠方式处理指令序列时，可以同时处理 3 条指令，且 3 条指令分别处于取指令、分析指令和执行指令三个不同的阶段。显然，为实现二次重叠处理，处理机必须具有独立的取指令部件、分析指令部件和执行指令部件，才能够使三条指令的不同阶段同时进行。自然，每个部件必须具有各自的控制逻辑，以控制实现各自的功能。由于指令分析不需要微操作，也就不需要相应的控制逻辑；由于取指令即是存储访问操作，与读写操作数的操作相同，因此取指令所需要的控制逻辑与读写操作数共享，且称之为存储控制器。把执行指令部件的控制逻辑称为运算控制器，则二次重叠处理指令序列的基本结构如图 3-15 所示。

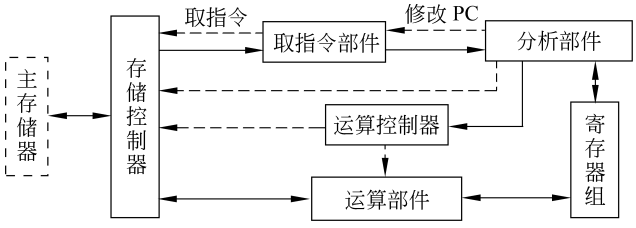

图 3-15　二次重叠处理指令序列的实现结构

在 Inter8086 微处理器中就有重叠处理指令序列的雏形，8086 微处理器的组成可分为总线接口部件和执行部件两部分。总线接口部件的任务有：从主存单元预取指令，送到指令队列缓冲器暂存，实现取指令；从指定主存单元或 I/O 端口取数据送至执行部件，或把执行部件的运算操作结果送至指定的主存单元或 I/O 端口，实现数据存取。执行部件负责从指令队列缓冲器取指令，并进行译码分析和指令执行。这两个功能部件再加上一个队列缓冲器，就可以实现取指令与分析、执行指令之间的一次重叠。

2. 主存访问冲突及其解决方法

当采用二次重叠处理指令序列的实现结构后，处理器可以并发进行取指令、分析指令和执行指令。取指令需要访问主存，分析指令可能需要访问主存来读取操作数，执行指令可能需要访问主存来写结果，这样就使得主存访问可能发生冲突。在一般的计算机中，指令与数据是混存于一个主存中的，而主存的一个存取周期仅能访问一个存储字。所以当计算机仅有一个主存储器时，往往无法实现指令之间的重叠处理。目前，重叠处理指令序列时发生主存访问冲突的解决方法有三种。

(1) 设置两个独立编址的主存储器。设置指令存储器和数据存储器，指令存储器用于存放程序的指令序列，数据存储器用于存放数据。这两个存储器可以同时独立访问，从而解决了取指令与读写操作数之间的访问冲突。如果规定指令执行阶段的运算结果仅写到通用寄存器而不直接写到主存，那么取指令、分析指令和执行指令就可以并发进行。目前，高性能微处理器的芯片内部均配置两个高速缓冲存储器(Cache)，一个是指令 Cache，一个是数据 Cache。

（2）主存储器采用多体交叉编址的并行存储器。多体交叉编址的并行存储器在一个存取周期内可以访问多个存储字，如果同时进行的取指令与读写操作数所存放的存储单元不在同一个存储体中，就可以实现指令序列的重叠处理，否则无法重叠处理指令序列。

（3）采用先行控制技术。流水线各功能段的延迟时间不可能绝对相等，也不可能在较长时间内为流水线连续不断地提供同种任务。因此，当处理器采用二次重叠处理指令序列的结构时，取指令、分析指令和执行指令等部件不可能完全处于忙碌状态，流水线效率不可能得到充分发挥。前面两种解决访问冲突的方法不仅对此无能为力，而且还没有完全解决访问冲突；设置两个主存时，操作数读与写之间的访问冲突仍然存在；采用并行存储器时，三种主存访问的存储单元必须在不同的存储体上。先行控制技术既可以有效地解决访问冲突，又可以充分发挥流水线效率。

3.2.2 先行控制及其实现结构

1. 先行控制技术的基本原理

先行控制技术最早出现在 IBM 公司研制的 STRETCH 计算机上，目前在处理器中已得到广泛应用。先行控制技术是为了连续流畅地实现指令序列重叠处理而提出的，没有先行控制技术的支持，将无法连续流畅地重叠处理指令序列，也就无法实现真正意义上的流水线。

在二次重叠处理指令序列中，假设取指令、分析指令和执行指令的延迟时间相等，则流水线连续流畅，否则功能段之间存在相互等待现象。但实际上功能段的延迟时间往往是不相等的，若取指令、分析指令和执行指令的时间分别为 Δt_1、Δt_2、Δt_3，且 $\Delta t_2 > \Delta t_3 > \Delta t_1$，采用二次重叠处理指令序列的过程如图 3-16 所示。从图 3-16 可以看出，取指功能段在取出第 $k+1$ 条指令后，由于分析功能段对第 k 条指令的分析还未结束，第 $k+1$ 条指令则不能进入分析功能段而产生堵塞，导致取指功能段空闲；执行功能段在执行第 k 条指令后，由于分析功能段对第 $k+1$ 条指令的分析还未结束，执行功能段得不到第 $k+1$ 条指令而产生断流，又导致执行功能段空闲。另外，当指令之间存在相关时，重叠也会被打断。如若第 k 条指令是转移指令，则按顺序取来的第 $k+1$ 条指令可能无效，从而产生停顿。

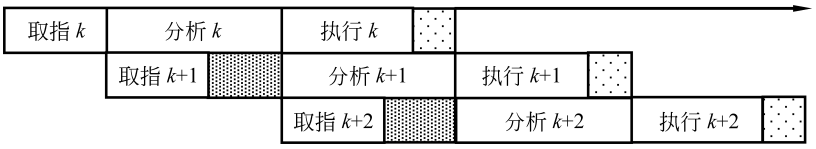

图 3-16 各功能段延迟时间不相等时二次重叠处理的时空图

为使流水线连续流畅而不产生停顿，当流水线某功能段的延迟时间较长时，则在该功能段前面设置预处理栈或在其后面设置后行处理栈，或者同时设置预处理栈和后行处理栈。这样功能段之间相互隔离，所有功能段各自处于忙碌状态，当任务在流水线上流动时，即使功能段之间存在等待时间，但各自的任务流动是连续的。这就是先行控制技术的基本原理。预处理栈一方面对后继任务进行预处理，另一方面用于缓存预处理好的后继

任务；而后行处理栈一方面对任务进行后行处理，另一方面用于缓存后行处理好的任务。当然，如果功能段后行处理栈的缓冲器已满，那么该功能段则产生堵塞；如果功能段预处理栈的缓冲器为空，那么该功能段则产生断流；这时流水线仍不是连续流畅的，但可以使功能段空闲降到最低。

先行控制技术不仅使得任务在流水线上连续流畅地流动，而且还有效地解决了主存访问冲突。当多个功能段同时访问主存时，根据具体情况，让其中一个功能段访问主存，该功能段前面的功能段则对后继任务进行预处理，其后面的功能段则对任务进行后行处理。

2. 先行控制定义

先行控制是通过对任务进行预处理和缓冲，以平滑功能部件（功能段）之间工作速度上的差异，使功能部件各自独立地工作，始终处于忙碌状态，提高任务序列的处理速度。显然，先行控制技术实质是缓冲技术和预处理技术相结合的结果，这两种技术是先行控制的关键。

缓冲技术是在工作速度不固定的两个功能段之间设置缓冲栈，用以平滑它们的工作速度的差异。预处理技术是把需要进入某功能段的任务进行外围处理，以减少任务在该功能段中的延迟时间。如把需要进入运算器的指令均处理为寄存器-寄存器型（RR 型）指令，结合缓冲技术，则可以为进入运算器的指令准备好所需要的全部操作数。

3. 先行控制的实现结构

将指令处理分为取指令、分析指令和执行指令三个阶段，执行指令所包含的微操作复杂多变，延迟时间也最长，而取指令与分析指令所包含的微操作相对固定简单。为此，在二次重叠处理指令序列的基本结构基础后，增设 4 个先进先出的先行缓冲栈。在分析指令部件与运算控制器之间增设后行操作栈，在主存储器与执行指令部件之间增设先行读数栈，在执行指令部件后面增设先行写数栈，并使取指令部件增加缓冲功能，先行控制实现的基本结构如图 3-17 所示。特别地，通常把先行指令缓冲栈、先行读数栈、先行操作栈和后行写数栈统称为先行控制器，把先行控制器与指令分析器合称为指令控制部件，而把运算部件及运算控制器合称为指令执行部件。

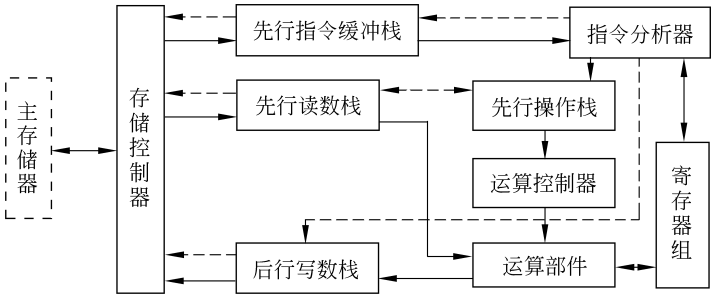

图 3-17　先行控制实现的基本结构

1）先行指令缓冲栈

先行指令缓冲栈是主存与指令分析器之间的一个缓冲部件，用于平滑它们之间工作

速度上的差异。存储控制器一般将取指令的优先级设为最低，其次是操作数的读写，输入输出的优先级最高。只要主存有空闲，就通过预取从主存中取多条指令存放于先行指令缓冲栈中。而当指令分析器空闲时，它就从先行指令缓冲栈中得到所需要的指令。

由于正在处理的指令与从主存中取出的指令是不同的，所以需要设置两个程序计数器——先行程序计数器 PC_1 和现行程序计数器 PC_2，二者计数的顺序是一致的，以维持正确的指令序列。先行指令缓冲栈的组成结构如图 3-18 所示。存储控制器根据先行程序计数器 PC_1 的内容，从主存中取指令；指令分析器则根据现行程序计数器 PC_2 的内容，从先行指令缓冲栈取指令。

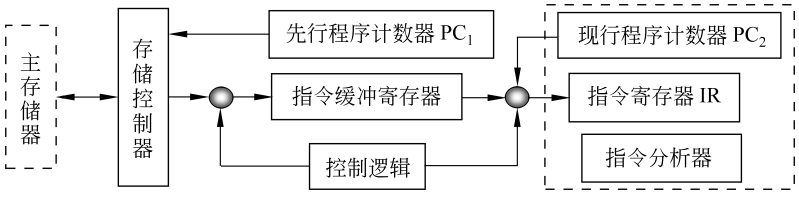

图 3-18　先行指令缓冲栈的组成结构

2）先行操作栈

先行操作栈是指令分析器和运算控制器之间的一个缓冲部件，用于平滑它们之间工作速度上的差异。当运算部件空闲时，运算控制器就从先行操作栈取出一条 RR * 型指令（由“*”来区分真正的 RR 型指令），该指令所需要的操作数则来自于先行读数栈或通用寄存器中。RR * 型指令是先行操作栈预处理的结果，其预处理是将变址型（RS 型）、存储器型（SS 型）等指令转换为 RR * 型指令。

3）先行读数栈

先行读数栈是主存与运算部件之间的一个缓冲部件，用于平滑它们之间工作速度上的差异。如果先行操作栈遇到从主存读取源操作数的指令，则由先行操作栈计算出主存有效地址后送到先行读数栈，先行读数栈从主存读取源操作数，并把它送到 RR * 型指令所指示的寄存器。这样，运算部件执行指令时，均不需要访问主存来获取源操作数，从而缩短了指令的执行时间。

4）后行写数栈

后行写数栈与先行读数栈一样，是主存与运算部件之间的一个缓冲部件。如果先行操作栈遇到向主存写入操作数的指令，则由先行操作栈计算出主存有效地址后送到后行写数栈；当运算部件执行 RR * 型指令后，将结果操作数送入后行写数栈，由后行写数栈把结果操作数写入主存。这样，运算部件执行指令时，均不需要将结果操作数写入主存，从而缩短了指令的处理时间。

4. 栈缓冲深度的选择

先行控制器所包含的 4 个缓冲栈中的缓冲寄存器个数即是缓冲深度。缓冲深度小，缓冲效果不大；缓冲深度大，控制复杂，延迟时间长，还会浪费寄存器。至于选择多大的缓冲深度，一般是在静态分析的基础上，通过模拟方法来确定栈的缓冲深度。而静态分析方法是从两个极端情况来考察。

一是假设先行指令缓冲栈的缓冲深度为 D_i，且已完全充满。若此时先行指令缓冲栈输出端输出的指令较简单，指令分析器的速度快，缓冲栈流出指令的速度也快；而在先行指令缓冲栈的输入端，由于需要访存取指令，指令流入速度慢。设平均每条指令的分析时间为 t_1、访取时间为 t_2，且有 $t_2>t_1$。在先行指令缓冲栈从完全充满到全部被排空的过程中，指令分析器分析了 L_1 条指令，所花费时间为 L_1t_1；同时，从主存中读取了 L_1-D_i 条指令，所花费时间为 $(L_1-D_i)t_2$。则有：

$$L_1t_1=(L_1-D_i)t_2$$
$$D_i=\lceil L_1(t_2-t_1)/t_2 \rceil \tag{3-1}$$

由于先行指令缓冲栈为空时，指令分析器被迫处于等待状态。为使先行指令缓冲栈不被取空，其深度必须大于 $\lceil L_1(t_2-t_1)/t_2 \rceil$。

二是假设先行指令缓冲栈为空，此时先行指令缓冲栈输出端输出的指令较复杂，指令分析器的速度慢，缓冲栈流出指令的速度也慢。相比之下，访存取指速度快，先行指令缓冲栈输入端指令流入速度也快。设平均每条指令的分析时间为 t_1^*、访取时间为 t_2^*，且有 $t_1^*>t_2^*$。从先行指令缓冲栈为空到完全充满的过程中，从主存读取到先行指令缓冲栈的指令为 L_2 条，所花费时间为 $L_2t_2^*$；同时，指令分析器分析了 (L_2-D_i) 条指令，所花费时间为 $(L_2-D_i)t_1^*$。则有：

$$L_2t_2^*=(L_2-D_i)t_1^*$$
$$D_i=\lceil L_2(t_1^*-t_2^*)/t_1^* \rceil \tag{3-2}$$

由于先行指令缓冲栈充满时，缓冲栈将失去效果。为使先行指令缓冲栈不被充满，其深度必须小于 $\lceil L_2(t_1^*-t_2^*)/t_1^* \rceil$。

所以，先行指令缓冲栈的深度主要由主存速度决定。为减小先行指令缓冲栈的深度，必须使指令访取时间 t_2 尽量接近指令分析时间 t_1，即提高主存速度。由于指令分析时间一般小于指令执行时间，采用类似的方法可以计算出其他缓冲栈的深度。对于采用先行控制技术的处理机，各缓冲栈的深度有如下关系：

$$D_{\text{先行指令}} \geqslant D_{\text{先行操作}} \geqslant D_{\text{先行读数}} \geqslant D_{\text{后行写数}}$$

如 IBM/370/165 机，$D_{\text{先行指令}}=4$，$D_{\text{先行操作}}=3$，$D_{\text{先行读数}}=2$，$D_{\text{后行写数}}=1$。

3.2.3　不同级别的流水线结构

1. 先行控制的指令流水线结构

在采用先行控制的处理机中，其各个部件构成一条流水线，如图 3-19 所示。先行控制处理机处理指令序列时，把指令处理分解为 6 个子过程，即取指令、指令译码、指令变换、读操作数、运算和存结果。

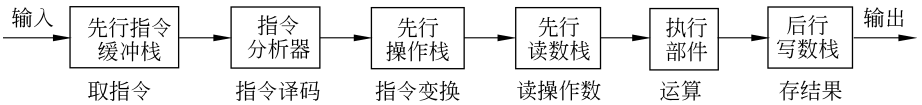

图 3-19　先行控制的指令流水线结构

如有一个程序按指令顺序共有 $i+j+k+n+m+p+q+r$ 条指令，且 $k=1$，$p=1$。

在某一时刻，运算部件正在执行第 k 条指令，而第 k 条指令之前的最前面的 i 条指令已全部处理完，结果已写到主存储器。已通过运算的 j 条指令在后行写数栈中等待把结果写到主存储器，第 k 条指令之后的 n 条指令已由先行操作栈完成预处理，以 RR * 型指令存放在先行操作栈中，所需要的操作数也已读到先行读数栈。往后的 m 条指令已生成 RR * 型指令并存放在先行操作栈中，但所需要的操作数还未读到先行读数栈。再往后的第 p 条指令正在指令分析器中分析，第 p 条指令后的 q 条指令已从主存储器预取到先行指令缓冲栈中，最后的 r 条指令则还在主存储器中，其中第一条正准备从主存储器流到 CPU 中。可见，先行控制的指令流水线上有大量的指令存在，使得任何功能段都基本上处于忙碌状态。

2. 运算操作的流水线结构

把运算过程中的各种操作为一个子过程，这样每个子过程的延迟时间不可能绝对相等，但偏移不大。因此，采用集中控制方式，使所有功能段与一个统一时钟同步，时钟周期的长短由所有运算操作中延迟时间最长的操作决定。由于运算操作的延迟时间不同，功能段之间可能会有干扰。为了避免干扰，保证功能段之间的时间宽度匹配，在各功能段之间增设门(Latch)寄存器，这样运算操作的流水线结构如图 3-20 所示。

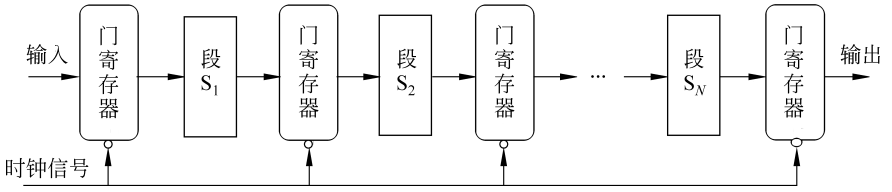

图 3-20　运算操作的流水线结构

3. 宏流水线结构

宏流水线是多台处理机对同一数据流的不同作业分别进行处理，其结构则是异构型多处理机。多处理机处理任务的效率不仅与体系结构有关，而且与算法、语言和软件等有关。宏流水线结构如图 3-13 所示。

3.2.4　流水线结构中存在的问题

1. 瓶颈段问题

当流水线各功能段延迟时间相等时，在流水线上流动的所有任务在每个节拍同步地往前流动一段。当流水线各功能段延迟时间不相等时，延迟时间最长的功能段就成为瓶颈段，统一时钟由瓶颈段的延迟时间决定。因此，在设计流水线结构时，应尽可能使各功能段延迟时间相等。

2. 额外开销问题

当在流水线上处理任务时，会产生额外的开销，这些开销在进行非流水线方式处理时是不会存在的。流水线上的额外开销包含两部分：门寄存器延迟和时钟到达偏差。由于流水线功能段之间均需要设置门寄存器，而门寄存器需要有建立时间和传输延迟。时钟到达各门寄存器的时间不是完全相同的，应该把它们的最大偏差计入统一时钟之中，使得

统一时钟周期变长。所以当采用指令流水线来处理指令序列时，不仅不可能减少单条指令的处理时间，由于额外开销的存在，反而会增加这个时间。因此，在设计流水线结构时，应尽可能减少额外开销。

3. 任务间相关问题

无论是运算操作流水线还是指令流水线或宏流水线，如果后面任务的计算需要用到前面任务的计算结果，而前面任务的计算还没有结束，则后面任务就会停止向前流动，即引起流水线的停顿。流水线上流动的前后任务只要存在关联关系，就是相关问题。因此，在设计流水线结构时，解决好相关问题可以有效地提高流水线的性能。

3.3　线性流水线性能及其最佳段数选取

【问题小贴士】 ①对于处理机，在指令序列串行处理的基础上，采用流水线技术并行处理指令序列，可以提高指令的处理速度和功能部件的利用率，那么流水线任务处理速度和资源利用率分别利用什么指标来衡量？如何计算它们？②由于流水线功能段之间均需要设置门寄存器，由此便会产生任务传输延迟，且功能段越多延迟越大，可见并非流水线上的功能段越多越好，那么如何选取流水线功能段数量？③当流水线各功能段延迟时间不相等时，将出现“瓶颈”段，且引起流水线“堵塞”与“断流”。采用先行控制技术，可以尽可能地避免“堵塞”与“断流”，这能否从根本上解决“瓶颈”段所带来的问题？④处理机指令主要包括运算加工、数据传输和程序转移三种类型，运算指令和传输指令与后继指令的处理顺序与程序顺序始终一致，转移指令与后继指令的处理顺序与程序顺序可能一致也可能不一致，是否一致则一般在转移指令处理的后期才能决定，这使得转移指令的后继指令应该停顿延迟，那么这样对流水线效率的影响有多大？上述问题的解决涉及许多变换计算，还需要通过练习来掌握直至熟悉。

3.3.1　线性流水线的性能指标

通常用于衡量流水线性能的主要指标有吞吐率、加速比和效率，它们从不同侧面反映流水线性能。

1. 吞吐率

吞吐率(ThoughPut Rate，TP)是指在单位时间内流水线所处理的任务数量或输出的结果数量。即有：

$$TP = N/T_K \tag{3-3}$$

其中，N 为任务数，T_K 是处理 N 个任务所用的时间。式(3-3)是计算流水线吞吐率的基本公式。

1) 流水线各功能段延迟时间相等时的吞吐率

若有一条 K 个功能段的线性流水线，各功能段延迟时间均为 Δt。当 N 个任务理想地连续输入流水线时，流水线的时空图如图 3-21 所示，其处理 N 个任务所需要的时间可从两个方面来分析。一是从流水线输出端来看，用 K 个 Δt 输出第一个任务，即“装入时间”为 $K\Delta t$，之后则每隔一个 Δt 输出一个任务，其余 $N-1$ 个任务输出需要 $(N-1)\ \Delta t$

的时间。二是从流水线输入端来看，每隔一个 Δt 向流水线输入一个任务，需要 $N\Delta t$ 的时间，另外还需要 $K-1$ 个 Δt 来把任务排空，即“排空时间”为$(K-1)\ \Delta t$。因此，流水线处理 N 个任务需要的总时间为：

$$T_K=(N+K-1)\ \Delta t \tag{3-4}$$

将式(3-4)代入式(3-3)中，则得到当流水线各功能段延迟时间相等时连续输入 N 个任务到含 K 个功能段的线性流水线的实际吞吐率为：

$$TP=\frac{N}{(N+K-1)\cdot\Delta t} \tag{3-5}$$

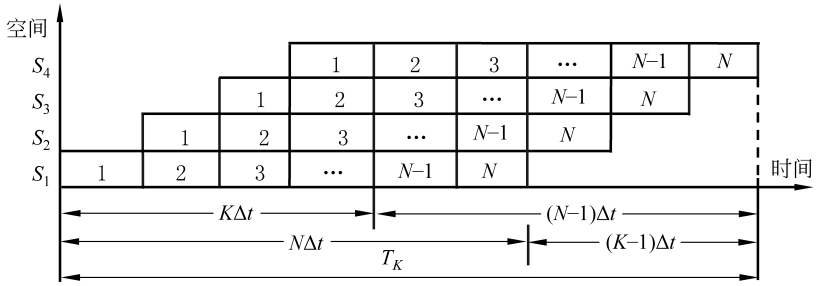

图 3-21　各功能段延迟时间均相等的流水线时空图

当 N 趋于无穷大时的最大吞吐率为：

$$TP_{\max}=\lim_{N\to\infty}\frac{N}{(N+K-1)\cdot\Delta t}=\frac{1}{\Delta t} \tag{3-6}$$

最大吞吐率与实际吞吐率的关系是：

$$TP=\frac{N}{N+K-1}=TP_{\max} \tag{3-7}$$

从式(3-7)可以看出，流水线的实际吞吐率小于最大吞吐率，它除与 Δt 有关之外，还与流水线的段数 K 和输入流水线中的任务数 N 有关。只有当 N 远大于 K 时，才有 $TP\approx TP_{\max}$。

2）流水线各功能段延迟时间不相等时的吞吐率

若有一条 K 个功能段的线性流水线，各功能段延迟时间分别为 $\Delta t_1,\Delta t_2,\cdots,\Delta t_K$，即流水线中存在“瓶颈”功能段，且“瓶颈”功能段的延迟时间为 $\max\{\Delta t_1,\Delta t_2,\cdots,\Delta t_K\}$。设 $K=4$，$\Delta t_1=\Delta t_3=\Delta t_4=\Delta t$，$\Delta t_2=3\Delta t$，则线性流水线的连接图如图 3-22 所示，相应时空图如图 3-23 所示。从图 3-23 可以看出，除第一个任务外，其余$(N-1)$ 个任务必须按“瓶颈”段延迟时间间隔连续流入流水线。因此，各功能段延迟时间不相等且连续输入任务时的线性流水线的实际吞吐率为：

$$TP=N/\left(\sum_{i=1}^{K}\Delta t_i+(N-1)\max\{\Delta t_1,\Delta t_2,\cdots,\Delta t_K\}\right) \tag{3-8}$$

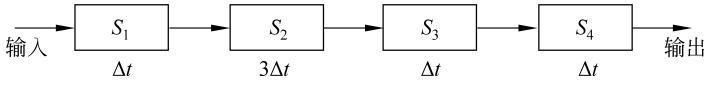

图 3-22　各功能段延迟时间不相等的流水线连接图

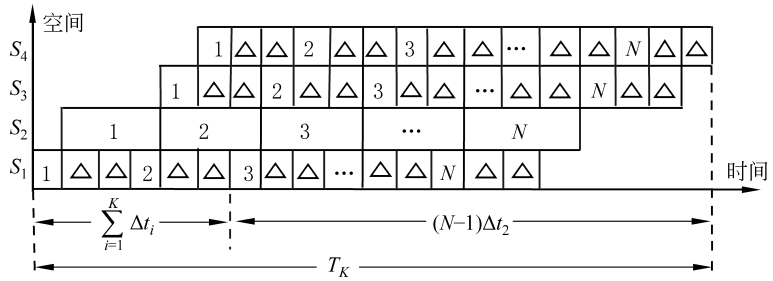

图 3-23　各功能段延迟时间不相等的流水线时空图

式(3-8)中表示总时间的分母中的第一项是流水线处理第一个任务所需要的时间，第二项是处理其余 $N-1$ 个任务所需要的时间。当 N 趋于无穷大时的最大吞吐率为：

$$TP_{\max}=\frac{1}{\max\{\Delta t_1,\Delta t_2,\cdots,\Delta t_K\}} \tag{3-9}$$

从式(3-8)和式(3-9)中可以看出，当流水线中各功能段延迟时间不相等时，流水线的最大吞吐率与实际吞吐率主要由“瓶颈”段的延迟时间决定。这时除“瓶颈”段一直忙碌外，其余功能段都有空闲时间，图 3-23 中“△”符号表示对应功能段在相应时间内是空闲的，这实际上是资源的浪费。

2. 加速比

加速比(Speedup Ratio，S)是指对于同一批任务采用串行处理方式所用的时间与采用流水线所用的时间之比。则流水线加速比为：

$$S=T_0/T_K \tag{3-10}$$

其中，T_0、T_K 分别为对于同样一批任务不采用流水线处理与采用流水线处理所用的时间，式(3-10)是计算流水线加速比的基本公式。

1) 流水线各功能段延迟时间相等时的加速比

若有一条 K 个功能段的线性流水线，各功能段延迟时间均为 Δt。采用流水线处理 N 个连续输入的任务所需要的时间可见式(3-4)，而不采用流水线处理这 N 个任务所需的时间为：$N\cdot K\Delta t$。因此，流水线的实际加速比为：

$$S=\frac{N\cdot K\Delta t}{(N+K-1)\Delta t}=\frac{N\cdot K}{N+K-1} \tag{3-11}$$

当 N 趋于无穷大时的最大加速比为：

$$S_{\max}=\lim_{N\to\infty}\frac{N\cdot K}{N+K-1}=K \tag{3-12}$$

从式(3-12) 中可以看出，当 N 远大于 K 时，在线性流水线各功能段延迟时间相等的情况下，流水线的最大加速比等于流水线的功能段数量。

2) 流水线各功能段延迟时间不相等时的加速比

根据流水线各功能段延迟时间不相等时吞吐率的计算分析，一条有 K 个功能段的线性流水线处理 N 个连续任务的实际加速比为：

$$S=N\cdot\sum_{i=1}^{K}\Delta t_i\Big/\sum_{i=1}^{K}\Delta t_i+(N-1)\max\{\Delta t_1,\Delta t_2,\cdots,\Delta t_K\} \tag{3-13}$$

3. 效率

效率(Efficiency,E)是指流水线上的资源利用率,在时空图上则定义为 N 个任务占用的时空区与 K 个功能段占用的总时空区之比,即有:

$$E = N \text{ 个任务占用的时空区} / K \text{ 个功能段占用的总时空区} \tag{3-14}$$

式(3-14)是计算流水线效率的基本公式,其分母是处理 N 个任务所需要的时间与 K 个功能段所围成的时空总面积 $K \cdot T_K$,分子是处理 N 个任务时实际占用的有效时空面积。因此,流水线的效率包含时间和空间两方面的因素,通过时空图来计算流水线的效率是极其必要又非常方便的。

1) 流水线各功能段延迟时间相等时的效率

若有一条 K 个功能段的线性流水线,各功能段延迟时间均为 Δt。在流水线上处理 N 个连续输入的任务所需要的时间可见式(3-4),在这段时间内 K 个功能段均被占用,则时空图的总面积为 $K \cdot (N+K-1)\Delta t$。任一任务占用的时空区为 $K \cdot \Delta t$,N 个任务实际占用的有效面积为 $N \cdot K\Delta t$。则流水线效率为:

$$E = \frac{N \cdot K\Delta t}{K \cdot (N+K-1)\Delta t} = \frac{N}{N+K-1} \tag{3-15}$$

当 N 趋于无穷大时的最大效率为:

$$E_{\max} = \lim_{N \to \infty} \frac{N}{N+K-1} = 1 \tag{3-16}$$

从式(3-16)中可以看出,当 N 远大于 K 时,流水线效率达到最大值即为 1,这时"装入时间"和"排空时间"忽略不计,流水线的各功能段均处于忙碌状态,时空图中每个时空块均被有效利用。

2) 流水线各功能段延迟时间不相等时的效率

根据流水线各功能段延迟时间不相等时吞吐率的计算分析,一条有 K 个功能段的线性流水线处理 N 个连续任务的实际效率为:

$$E = N \cdot \sum_{i=1}^{K} \Delta t_i \Big/ K \cdot \left[\sum_{i=1}^{K} \Delta t_i + (N-1)\max\{\Delta t_1, \Delta t_2, \cdots, \Delta t_K\} \right] \tag{3-17}$$

4. 吞吐率、加速比和效率之间的关系

比较式(3-5)和式(3-15),可以得到:

$$E = TP \cdot \Delta t \quad \text{或} \quad TP = E/\Delta t \tag{3-18}$$

式(3-18)说明:当流水线各功能段的延迟时间均为 Δt 时,流水线的效率与吞吐率成正比,即若采取措施提高了流水线的效率,则同时也提高了吞吐率。

比较式(3-11)和式(3-15),可以得到:

$$E = S/K \quad \text{或} \quad S = K \cdot E \tag{3-19}$$

式(3-19)说明:流水线效率是流水线实际加速比 S 与其最大加速比 $S_{\max} = K$ 之比,只有当流水线效率达到最大值 1 时,才能使实际加速比达到最大值 K。

当 N 远大于 K 时,则有:

$$E_{\max} = 1, \quad S_{\max} = K, \quad TP_{\max} = 1/\Delta t \tag{3-20}$$

当流水线各功能段的延迟时间相等时，流水线上的各个功能段始终处于忙碌状态，没有空闲时间，这时流水线的吞吐率、加速比和效率都很高，达到最大值。但实际上由于多种原因，流水线的实际吞吐率、加速比和效率远低于相应的最大值。例如，流水线存在装入和排空时间，输入的任务往往不是连续的，程序本身存在相关，多功能流水线在处理某种任务时有的功能段不需要使用等。

特别地，计算流水线吞吐率、加速比和效率有许多公式，在实际分析流水线的性能时，应该注意它们各自适用的场合，但式(3-3)、式(3-10)和式(3-14)具有普适性。另外，当任务输入不连续时，需要先画出时空图，然后利用时空图来计算各项性能指标。

3.3.2　流水线最佳段数的选取

当流水线各功能段的延迟时间相等时，由于任务处理时间是不变的，所以增加流水线功能段数量 K，会使功能段延迟时间 Δt 变小。由式(3-6)和式(3-12)可以看出，流水线的最大吞吐率和最大加速比均可以得到提高。但由于每个功能段的输出端都必须设置一个锁存器，使得在流水线功能段数量增加时，锁存器上的总延迟时间也将增加，任务处理的总时间也必将增加。另外，随着流水线功能段数量增加，锁存器数量也将增加，流水线的造价也会增加。所以，在设计流水线时，需要综合考虑各方面的因素，根据最佳性价比来选取流水线功能段的数量。

假设采用串行处理时，处理一个任务所需要的时间为 T。采用含有 K 个功能段的流水线处理时，每个功能段的延迟时间为 T/K，流水线的最大吞吐率为 $TP_{\max}=1/(T/K+D)$(D 为锁存器的总延迟时间)。又设流水线 K 个功能段的总造价为 A，每个锁存器的价格为 B，则流水线的总价格为 $C=A+BK$。由此，将流水线的性价比 PCR 定义为：

$$\mathrm{PCR}=\frac{TP_{\max}}{C}=\frac{1}{T/K+D}\cdot\frac{1}{A+BK} \tag{3-21}$$

通过对自变量 K 求导，可得到 PCR 的极值。由于大于零的极值只有一个，这个极值就是最大值。如图 3-24 所示，当性价比 PCR 取得最大值时，它所对应的流水线的功能段数量就是最佳段数 K_0：

$$K_0=\sqrt{TA/DB} \tag{3-22}$$

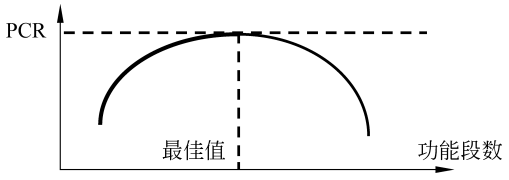

图 3-24　流水线性价比与功能段数的关系

由式(3-22)可以看出，流水线最佳段数与任务处理时间 T 和流水线功能段的造价 A 的平方根成正比，与锁存器的总延迟时间 D 和锁存器价格 B 的平方根成反比。目前，一般处理机流水线的功能段数为 2～10，极少超过 15，且一般把段数大于或等于 8 的流水线称为超流水线。