计算机组成与体系结构

第7章 中央处理器

张 钰
ZJSS@BIT.EDU.CN
北京理工大学计算机学院
SCHOOL OF COMPUTER SCIENCE & TECHNOLOGY
BEIJING INSTITUTE OF TECHNOLOGY

第7章 中央处理器

7.1 中央处理器概述

7.1.1 CPU的功能

从程序运行的角度来看，CPU的基本功能就是对指令流和数据流在时间与空间上实施正确的控制，主要包含以下几种具体功能：

指令控制：能自动从存储器中取出指令，控制指令的执行序列，如顺序、跳转等。

操作控制：分析指令，产生完成指令所需要的控制信号，协调并控制计算机各部件执行指令的操作。

时序控制：对各种操作加以时间上的控制。

数据加工：对数据进行算术运算、逻辑运算或者逻辑测试。

中断处理：处理计算机运行过程中出现的异常情况和特殊要求。

7.1 中央处理器概述

7.1.2 CPU的组成与主要寄存器

1.CPU的组成

CPU=控制器+运算器

控制器的主要功能如下：

① 从主存中取出一条指令，并指出下一条指令在主存中的位置。

② 对指令进行译码或测试，产生相应的操作控制信号，以便启动规定的动作。

③ 指挥并控制CPU、主存和输入输出设备之间的数据流动方向。

运算器的主要功能如下：

① 执行所有的算术运算。

② 执行所有的逻辑运算，并进行逻辑测试。

7.1 中央处理器概述

控制单元：CU
指令译码器：ID
指令寄存器：IR
存储器数据寄存器：MDR
算术逻辑单元：ALU
累加器：AC
程序状态字寄存器：PSWR
程序计数器：PC
存储器地址寄存器：MAR
输入输出设备：I/O

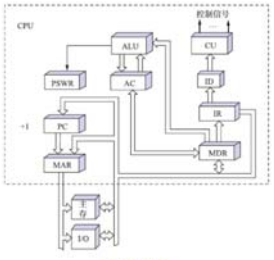
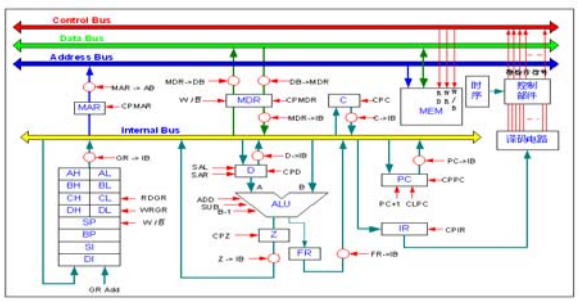


图7-1 CPU模型



2.CPU中的主要寄存器

对程序设计者可见的寄存器和“透明”的寄存器；

按功能分为通用的和专用的寄存器。

表 7-1 CPU 中常见的专用寄存器及功能

专用寄存器	功能
程序计数器	保存当前正在执行的指令地址以及下一条将要执行的指令地址
指令寄存器	保存当前正在执行的指令，一般在执行阶段不能改，也不能为程序设计者直接访问
程序状态字寄存器	存放程序和机器运行的状态
存储器地址寄存器	缓冲送往地址总线的地址
存储器数据寄存器	缓冲数据总线上的数据

7.1.3 CPU的主要技术参数

1.核心数量

通过芯片的主频和晶体管提升计算机性能基本抵达了物理极限，因此多核技术已经成为当前处理器的主流。一般而言物理核心越多，处理器性能越强，但并不绝对，感兴趣的同学可以了解下“功耗墙”问题。

2.主频、外频与倍频

主频是内部工作频率又称为内频，是衡量CPU速度的重要参数。CPU的主频表示在CPU内数字脉冲信号震荡的速度，与CPU实际的运算能力并没有直接关系。因此主频仅是CPU性能表现的一个方面，而不代表CPU的整体性能。

CPU除了主频之外，还有另一种工作频率，称为外部工作频率，它是由主板为CPU提供的基准时钟频率。

早期CPU的内频等于外频，现代微机采用倍频技术。内频、外频和倍频三者之间的关系是：

内频=外频×倍频

理论上倍频是从1.5一直到无限，以0.5为一个间隔单位。

3.片内高速缓冲存储器

CPU片内高速缓冲存储器（Cache）是为了解决CPU和主存之间工作的速度差异而设置的。片内Cache容量是衡量CPU的性能的重要指标之一，一般容量越大越好。现代主流的CPU 的片内高速缓冲存储器的组织采用层次结构。

4.电压与功耗

CPU的工作电压指的是CPU正常工作所需的电压。CPU的工作电压总的发展趋势有一个非常明显的下降变化，更低的电压，带来更低的功耗，更适合现代移动和便携式计算；更低的功耗，带来发热量减少，能够减缓器件的老化，延长机器的使用寿命；最重要的方面是降低电压有利于提高主频从而提高CPU性能。当然电压不是无限制地越低越好，实验表明较高的电压能带来更好的信号稳定性。

5.制造工艺

线宽是指芯片内电路与电路之间的距离，可以用线宽来描述制造工艺。线宽越小，意味着芯片上包括的晶体管数目越多。

Pentium Ⅱ的线宽是0.35μm，晶体管数达到7.5兆个；Pentium Ⅲ的线宽是0.25μm，晶体管数达到9.5兆个；Pentium 4的线宽是0.18μm，晶体管数达到42兆个。近年来线宽已由28nm、14nm、7nm发展到5nm，可以预计未来可能出现更小尺寸的线宽，但由于受到物理条件的限制，不可能无限小。

7.2.1基本概念

1.指令周期和机器周期

时序系统是控制器的心脏，其功能是为指令的执行提供各种定时信号。

指令周期：取指令、分析指令到执行完该指令所需的全部时间。指令功能和复杂度不同，各种指令周期不尽相同。

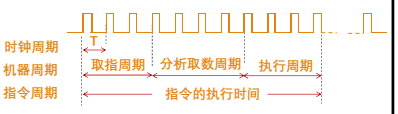
机器周期：每个机器周期完成一个基本操作，如取指周期。基本操作复杂度不同，机器周期不尽相同。

指令周期 = i×机器周期

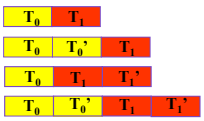
i为整数。

2.时钟周期及选取方案

时钟周期：处理器内数字脉冲震荡信号周期，主频的倒数。微机中常用指令周期-机器周期-时钟周期三级时序系统。



常见的时序选取方案：
(1) 定长CPU周期
(2) 不定长CPU周期
(3) 时钟周期插入法



(1) 定长CPU周期

以最复杂的机器周期为准定出时钟个数（节拍数），每一节拍时间的长短也以最繁的微操作作为标准。这种方法采用统一的、具有相等时间间隔和相同数目的时钟，使得所有的机器周期长度都是相等的，因此称为定长CPU周期。

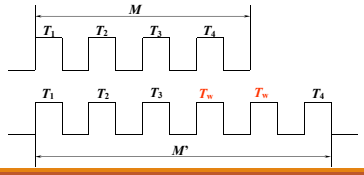
(2)不定长CPU周期

按照机器周期的实际需要安排时钟周期数，需要多少节拍，就发出多少节拍，这样可以避免浪费，提高时间利用率。由于各机器周期长度不同，又称为不定长CPU周期。

(3)时钟周期插入法

前两种方式的折中实现的思路。在照顾多数机器周期要求的情况下，选取适当的时钟周期个数（节拍数），作为基本时长，如果在某个机器周期内基本时长内无法完成该周期的全部微操作，则可通过插入时钟周期。

8086总线周期



7.2.2控制方式

1.同步控制方式

又称集中控制方式或中央控制方式。各操作均由统一时序信号驱动。

特点：设计简单，容易实现，有时间浪费。

2.异步控制方式

又称分散控制方式或局部控制方式。各项操作不采用统一的时序信号控制，而根据指令或部件的具体情况交互决定。

特点：控制复杂，没有时间上的浪费，机器效率高。

3.联合控制方式

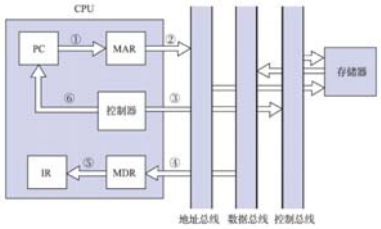
这是同步控制和异步控制相结合的方式。实际上现代计算机中几乎没有完全采用同步或完全采用异步的控制方式，大多数是采用联合控制方式。

通常的设计思想是：在功能部件内部采用同步方式或以同步方式为主的控制方式，在功能部件之间采用异步方式。

7.2 时序系统

7.2.3 指令运行的基本过程

一条指令运行的基本过程可以分为三个阶段：取指令阶段、分析取数阶段和执行阶段。



7.2 时序系统

1.取指令阶段

取指令阶段完成的任务是将现行指令从主存中取出来并送至指令寄存器中去。具体的操作为：

①将程序计数器（PC）中的内容送至存储器地址寄存器（MAR），并送地址总线（AB）。(PC)→MAR

②由控制单元（CU）经控制总线（CB）向主存发读命令。 Read

③从主存中取出的指令通过数据总线（DB）送到存储器数据寄存器（MDR）。M(MAR) →MDR

④将MDR的内容送至指令寄存器（IR）中。 (MDR)→IR

⑤将PC的内容递增，为取下一条指令做好准备。 (PC) + 1→PC

以上这些操作对任何一条指令来说都是必须要执行的操作，所以称为公共操作。

7.2 时序系统

2.分析取数阶段

取出指令后，机器立即进入分析指令阶段，指令译码器ID可识别和区分不同的指令类型及各种获取操作数的方法。由于各条指令功能不同，寻址方式也不同，所以分析取数阶段的操作是各不相同的。

3.执行阶段

执行阶段完成指令规定的各种操作，形成稳定的运算结果，并将其存储起来。

计算机的基本工作过程可以概括成为取指令、分析取数、执行指令，然后再取下一条指令，……。如此周而复始，直至遇到停机指令或外来的干预为止。

第7章 中央处理器

7.3 数据通路的组成与实现方法

7.3.1 数据通路概念

数据通路源于数字系统设计，指数据在各模块或者各子系统之间传送的路径。

主要模块包含：运算器、控制器、寄存器组和选择器等；

模块的连接形式可以采用：

基于总线的结构，或者基于专用数据通路直连结构。

7.3 数据通路的组成与实现方法

7.3.2 数据通路结构及其设计

数据通路设计的一般步骤为：

1. 拟定指令系统，确定指令的格式与功能，寻址方式等。

   如规整型指令、寄存器寻址方式。

2. 确定总体结构，分析每条指令的执行流程，确定该指令信息流经的功能部件，如PC，IR，ALU，MAR，MDR等以及连接方式。

3. 安排好工作时序，拟定指令流程与微命令序列。

4.形成控制逻辑实现，如采用组合逻辑型或者微程序控制型。

7.3 数据通路的组成与实现方法

7.3.3 数据通路实例分析

寄存器间接寻址　寄存器寻址　源操作数地址　目的操作数地址

加法指令ADD @R0，R1

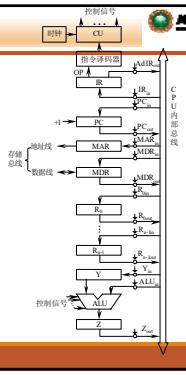
这条指令完成的功能是把R_0的内容作为地址送到主存以取得第一操作数，再与R_1的内容相加，最后将结果送回主存中。即实现：

$((R_0)) + (R_1) \rightarrow (R_0)$

7.3 数据通路的组成与实现方法

字母加下标in表示该部件的接收控制信号，实际上就是该部件的输入开门信号；

字母加下标out表示该部件的发送控制信号，实际上就是该部件的输出开门信号。



7.3 数据通路的组成与实现方法

1.取指阶段的数据通路

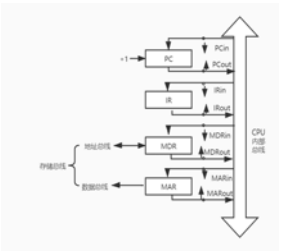
取指阶段的任务：

从主存读当前指令送到送到指令寄存器IR中，并将程序计数器PC更新为下一条指令地址。

涉及到的功能部件：

PC→MAR→MEM→MDR→IR

假定采用单总线方式连接各个功能部件。



7.3 数据通路的组成与实现方法

此加法指令的取指阶段具体操作如下：

① PCout和MARin有效，完成PC经CPU内部总线送至MAR的操作，记作(PC)→MAR；

② 通过控制总线（图中未画出）向主存发读命令，记作Read；

③ 存储器通过数据总线将MAR所指单元的内容（指令）送至MDR，记作M(MAR)→MDR；

④ MDRout和IRin有效，将MDR的内容送至IR，记作(MDR)→IR。至此，指令被从主存中取出，其操作码字段开始控制CU。

⑤ 使PC内容加1，记作(PC)+1→PC。

7.3 数据通路的组成与实现方法

2.分析取数阶段的数据通路

分析取数阶段操作完成取操作数的任务，与具体指令相关。如ADD @R_0,R_1指令第一个操作数采用的是寄存器间接寻址，第二个操作数采用的是寄存器寻址。

涉及到的功能部件：

取操作数涉及寄存器模块R_0和R_1，其中第二个操作数已经在R_1中。假定寄存器间接寻址取回的操作数送往暂存器Y保存，则该指令的间接取数阶段的数据通路为R_0→MAR→MEM→MDR→Y。

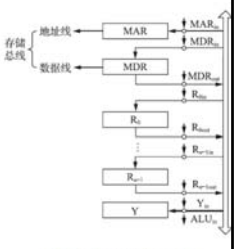


图 7-9　间接取数阶段的数据通路

7.3 数据通路的组成与实现方法

此加法指令的分析取数阶段具体操作如下：

被加数在主存中，加数已放在通用寄存器R1中。

① R0out和MARin有效，完成将被加数地址送至MAR的操作，记作(R0)→MAR；

② 向主存发读命令，记作Read；

③ 存储器通过数据总线将MAR所指单元的内容（数据）送至MDR，同时MDRout和Yin有效，记作M(MAR)→MDR→Y；

7.3 数据通路的组成与实现方法

3.执行阶段的数据通路

加法指令在此阶段的任务是完成加法运算，并将结果写回主存。

假定加法结果在寄存器Z中暂存，因此执行加法运算时流经的数据通路为(R_1)+(Y)→Z→MDR，其中ALU实现加法运算。

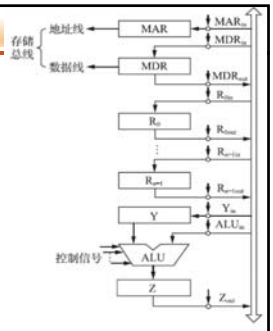


图 7-10 执行阶段的数据通路

7.3 数据通路的组成与实现方法

此加法指令的执行阶段具体操作如下：

① R_{1out}和ALU_{in}有效，同时CU向ALU发"ADD"控制信号，使R_1的内容和Y的内容相加，结果送寄存器Z，记作(R_1)+(Y)→Z。

② Z_{out}和MDR_{in}有效，将运算结果送MDR，记作(Z)→MDR。

③ 向主存发写命令，记作Write。

至此，运算结果被写入主存单元。

7.3 数据通路的组成与实现方法

① **PCout，MARin，ALUin，Read，0→Y，1→C0，"+"，Zin；**

② **Zout，PCin，Wait for MFC；**

③ **MDRout，IRin；**

④ **R_0out，MARin，Read；**

⑤ **R_1out，Yin，Wait for MFC；**

⑥ **MDRout，ALUin，"+"，Zin；**

⑦ **Zout，MDRin；**

⑧ **Write**

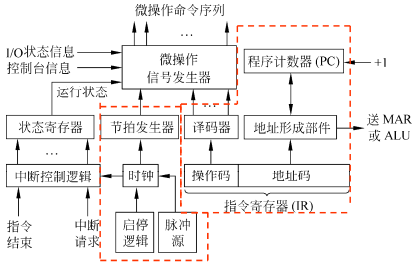
⑨ **END**

第7章 中央处理器

7.4控制器原理与实现方法

7.4.1控制器的基本组成

控制器是计算机系统的指挥中心，它把运算器、存储器、输入/输出设备等部件组成一个有机的整体，然后根据指令的要求指挥全机的工作。



7.4.1控制器的基本组成

指令部件：主要任务是完成取指令并分析指令。

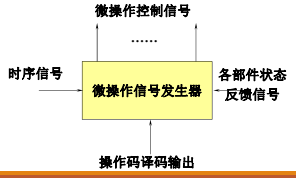
时序部件：能产生一定的时序信号，以保证机器的各功能部件有节奏地进行信息传送、加工及信息存储。

中断控制逻辑：用来控制中断处理的硬件逻辑。

微操作信号发生器：将一条指令的取出和执行可以分解成很多最基本的操作，这种最基本的不可再分割的操作称为微操作。微操作信号发生器也称为控制单元（CU）。

7.4.2 控制器的硬件实现方法

控制器的输入是机器指令代码，输出是微操作控制信号，因此微操作信号发生器是控制器的核心。根据产生微操作控制信号的方式不同，控制器可分为3种，它们的根本区别在于微操作信号发生器的实现方法不同，而控制器中的其它部分基本上是大同小异的。



1.组合逻辑型，又称常规控制器或硬布线控制器

由门电路组成产生微操作控制信号的复杂树形网络。

优点:速度快。缺点：结构不规整，使得设计、调试、维修较困难。

2.存储逻辑型，又称微程序控制器

把微操作信号代码化，存入控制存储器中，微操作控制信号由微指令产生。

优点：设计规整、调试、维修以及更改、扩充指令方便，易于实现自动化设计。缺点：指令执行速度比组合逻辑控制器慢。

3.组合逻辑和存储逻辑结合型，又称PLA控制器。

它是组合逻辑技术和存储逻辑技术结合的产物，它克服了两者的缺点，是一种较有前途的方法。

7.4.3单周期处理器的控制原理

表 7-2 寄存器寻址指令格式

操作码（6位）	源寄存器1（5位）	源寄存器2（5位）	目的寄存器2（5位）	移位（5位）	功能号（6位）
opcode	rs	rt	rd	shamt	funct

表 7-3 立即寻址指令格式

操作码（6位）	源寄存器1（5位）	源寄存器2（5位）	立即数（16位）
opcode	rs	rt	immediate

表 7-4 CPU 模型中 6 条指令功能和编码

序号	操作码	助记符	功能	描述
1	10 0000	Add	R[rd]←R[rs]+R[rt]; PC←PC+4	加法
2	10 0010	Sub	R[rd]←R[rs]-R[rt]; PC←PC+4	减法
3	00 1101	Ori	R[rt]←R[rs]+zf(imm16); PC←PC+4	立即数或运算
4	10 0011	Lw	R[rt]←MEM[R[rs]+sf(imm16)]; PC←PC+4	读存储器内容读入寄存器
5	10 1011	Sw	MEM[R[rs]+sf(imm16)]←R[rt]; PC←PC+4	寄存器内容写入存储器
6	00 0100	Beq	if(R[rs]==R[rt]) then PC←PC+4+[sf(imm16)]00] else PC←PC+4	相等则跳转

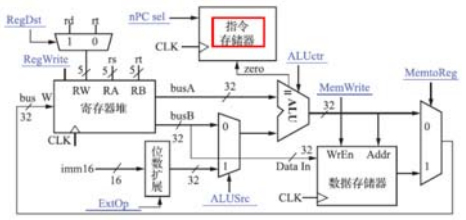


图 7-13 单周期数据通路示例

表 7-5 模型机的控制信号列表

操作码op	10 0000	10 0010	00 1101	10 0011	10 1011	00 0100
信号/指令	Add	Sub	Ori	Lw	Sw	Beq
ExtOp	X	X	0	1	1	X
ALUSrc	0	0	1	1	1	0
ALUctr[1:0]	00(add)	01(sub)	10(or)	00(add)	00(add)	01(sub)
nPC_sel	0	0	0	0	0	1
MemWrite	0	0	0	0	1	0
MemtoReg	0	0	0	1	X	X
RegDst	1	1	0	0	X	X
RegWrite	1	1	1	1	0	0

7.4控制器原理与实现方法

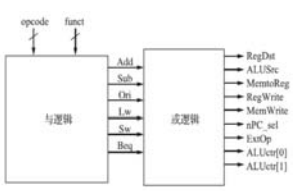


图7-14 单周期控制器

图7-15 组合数据通路与控制器

7.4控制器原理与实现方法

单周期控制器特点：

简单、CPI=1、效率低、无法器件复用…

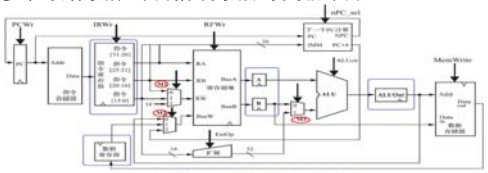
思考：

如何提升效率？

7.4控制器原理与实现方法

7.4.4多周期处理器的控制原理

单周期效率低，改进方式之一把指令的执行细分成多个基本操作，每个基本操作在一个时钟周期内完成，此时一个指令周期包含多个时钟周期，不同指令周期长度可能不同。



7.4控制器原理与实现方法

基于状态机的多周期控制器设计：

状态机不唯一。状态机设计时，首先将指令划分为若干大类。取指公操作是所有指令都有的，因此指令的状态分支在译码阶段判定，按照增量方式，由一条指令开始分析，根据指令的执行流程设计状态以及状态转换，若某条指令归入任一分类都感觉不合理时，则新增一个状态处理，直至指令集中的所有指令都处理完毕。如图7-175个状态的状态机。

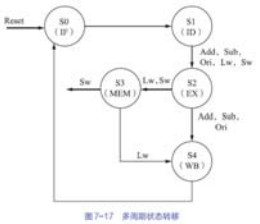


图7-17 多周期状态转移

第7章 中央处理器

7.1中央处理器概述

7.2时序系统

7.3数据通路的组成与实现方法

7.4控制器原理与实现方法

7.5微程序控制原理

7.6流水线技术

7.5微程序控制原理

7.5.1微程序控制的基本概念

- 1951年英国剑桥大学的M.V.Wilkes教授提出微程序设计的概念和原理。
- 1964年，IBM公司在IBM360系列机上成功地采用了微程序设计技术，解决了指令系统的兼容问题。
- 70年代以来，由于VLSI技术的发展，推动了微程序设计技术的发展和应用，目前，大多数计算机都采用微程序设计技术。

微程序设计技术的实质是将程序设计技术和存储技术相结合，即用程序设计的思想方法来组织操作控制逻辑，将微操作控制信号按一定规则进行信息编码（代码化），形成控制字（微指令），再把这些微指令按时间先后排列起来，存放在一个只读存储器中。

7.5微程序控制原理

1.微命令和微操作

一条机器指令可以分解成一个微操作序列，这些微操作是计算机中最基本的、不可再分解的操作。微命令是控制计算机各部件完成某个基本微操作的命令。

微命令和微操作是一一对应的。微命令是微操作的控制信号，微操作是微命令的操作过程。

微命令有兼容性和互斥性之分，兼容性微命令是指那些可以同时产生，共同完成某一些微操作的微命令；而互斥性微命令是指在机器中不允许同时出现的微命令。兼容和互斥都是相对的，一个微命令可以和一些微命令兼容，和另一些微命令互斥。对于单独一个微命令，谈论其兼容和互斥都是没有意义的。

7.5微程序控制原理

2.微指令、微地址

微指令是指控制存储器中的一个单元的内容，即控制字，它是若干个微命令的集合。存放控制字的控制存储器的单元地址就称为微地址。

一条微指令通常至少包含两大部分信息：

微指令有垂直型和水平型之分，垂直型微指令接近于机器指令的格式，每条微指令只能完成一个基本操作。水平型微指令则具有良好的并行性，每条微指令可以完成较多的基本操作。

7.5微程序控制原理

3.微周期

从控制存储器中读取一条微指令并执行相应的微命令所需的全部时间称为微周期。

4.微程序

一系列微指令的有序集合就是微程序。一条机器指令对应于一段微程序。

程序 → 指令的集合 → 指令（微程序）

指令（微程序）→ 微指令的集合 → 微指令

微指令 → 微命令的集合 → 微命令

程序设计者看到的传统机器层，包括：机器指令、工作程序、主存储器；

机器设计者看到的微程序层，包括：微指令、微程序和控制存储器。

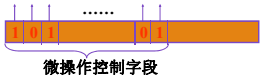
7.5微程序控制原理

7.5.2微指令编码法

微指令编码法指的就是微操作控制字段的编码方法。各类计算机的微指令编码法不同。

1.直接控制法（不译码法）

操作控制字段中的各位分别可以直接控制计算机，不需要进行译码。操作控制字段的每一个独立的二进制位代表一个微命令，该位为"1"表示这个微命令有效，为"0"表示这个微命令无效。每个微命令对应并控制数据通路中的一个微操作。



优点：结构简单，并行性强，操作速度快

缺点：微指令字太长，信息的利用率不高。

7.5微程序控制原理

2.最短编码法

最短编码微指令字最短。这种方法将所有的微命令统一编码，每条微指令只定义一个微命令。若微命令的总数为N，操作控制字段的长度为L，则：

$$L \geqslant \log_2 N$$

优点：微指令字长最短，

缺点：需要译码器译码才能得到需要的微命令。微命令数目越多，译码器就越复杂。无法并行性，微程序长，不支持某些要求在同一时刻同时动作的组合性微操作。

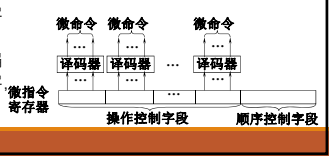
7.5微程序控制原理

3.字段编码法

前两种编码法的一个折衷的方法，既具有两者的优点，又克服了它们的缺点。这种方法将操作控制字段分为若干个小段，每段内采用最短编码法，段与段之间采用直接控制法。这种方法又可进一步分为字段直接编码法和字段间接编码法。

(1)字段直接编码法

各字段都可以独立地定义本字段的微命令，而和其他字段无关，因此又称为显式编码或单重定义编码方法。这种方法缩短了微指令字，因此得到了广泛的应用。



7.5微程序控制原理

字段编码法中操作控制字段的分段原则：

① 把互斥性的微命令分在同一段内，兼容性的微命令分在不同段内。这样不仅有助于提高信息的利用率，缩短微指令字长，而且有助于充分利用硬件所具有的并行性，加快执行的速度。

例如，运算器的输出控制信号有直传、左移、右移、半字交换等四个，这四个微命令是互斥的，它们可以安排在同一字段编码。同样，存储器的读和写命令也是一对互斥的微命令。还有象A→C、B→C（A、B、C都是寄存器）这样的一类微命令也是互斥的微命令，它们不允许在同一时刻出现。

7.5微程序控制原理

② 应与数据通路结构相适应。

③ 每个小段中包含的信息位不能太多，否则将增加译码线路的复杂性和译码时间。

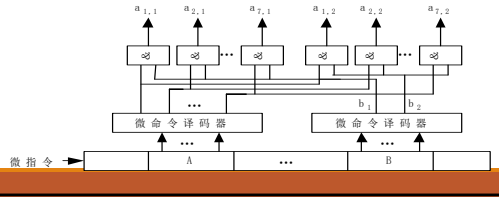
④ 一般每个小段还要留出一个状态，表示本字段不发出任何微命令。因此当某字段的长度为三位时，最多只能表示七个互斥的微命令，通常用000表示不操作。

例如：假设某计算机共有256个微命令，如果采用字段直接编码法，若4位为一个段，共需18段，操作控制字段只需72位，而且在同一时刻可以并行发出18个不同的微命令。

7.5微程序控制原理

(2)字段间接编码法

字段间接编码法是在字段直接编码法的基础上，用来进一步缩短微指令字长的方法。间接编码的含义是，一个字段的某些编码不能独立地定义某些微命令，而需要与其他字段的编码来联合定义，因此又称为隐式编码或多重定义编码方法。



7.5微程序控制原理

7.5.3.微程序控制器的基本组成

1.微程序控制器的基本组成

(1)控制存储器（CM）

核心部件，用来存放微程序。

(2)微指令寄存器（μIR）

用来存放从CM取出的正在执行的微指令。

(3)微地址形成部件

用来产生初始微地址和后继微地址。

(4)微地址寄存器（μMAR）

它接受微地址形成部件送来的微地址，为在CM中读取微指令作准备。

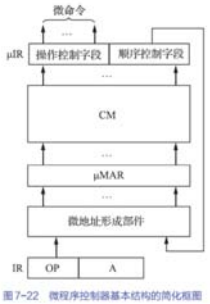
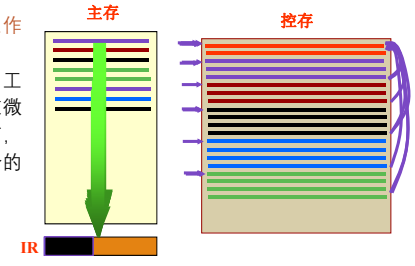


图7-22 微程序控制器基本结构的简化框图

7.5微程序控制原理

2.微程序控制器的工作过程

微程序控制器的工作过程实际上就是在微程序控制器的控制下，计算机执行机器指令的过程。



7.5微程序控制原理

(1)执行取指令公操作。取指令的公共操作通常由一段取指微程序来完成，这个取指微程序也可能仅由一条微指令组成。

具体的执行是：在机器开始运行时，自动将取指微程序的入口微地址送μMAR，并从CM中读出相应的微指令送入μIR。微指令的操作控制字段产生有关的微命令，用来控制实现取机器指令的公共操作。取指微程序的入口地址一般为CM的0号单元，当取指微程序执行完后，从主存中取出的机器指令就已存入指令寄存器IR中了。

(2)由机器指令的操作码字段通过微地址形成部件产生出该机器指令所对应的微程序的入口地址，并送入μMAR。

(3)从CM中逐条取出对应的微指令并执行之。

(4)执行完对应于一条机器指令的一段微程序后又回到取指微程序的入口地址，继续第(1)步，以完成取下条机器指令的公共操作。

以上是一条机器指令的执行过程，如此周而复始，直到整个程序执行完毕为止。

3. 机器指令对应的微程序

一条机器指令对应一个微程序。一般n条机器指令的微程序控制器中至少包含有n+1个微程序段。

7.5.4.微程序入口地址的形成

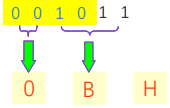
一般由机器指令的操作码字段指出各段微程序的入口地址（初始微地址），主要有三种方式。

1.一级功能转换

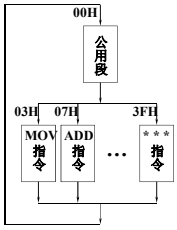
如果机器指令操作码字段的位数和位置固定，可以直接使操作码与入口地址码的部分位相对应。

例如，某机有16条机器指令，指令操作码由4位二进制数表示，现以字母θ表示操作码，令微程序的入口地址为：

θ 11B



各微程序的入口地址相差4个单元。



2.二级功能转换

当同类机器指令的操作码字段的位数和位置固定，而不同类机器指令的操作码的位数和位置不固定时，就不能再采用一级功能转换的方法。所谓二级功能转换是指第一次先按指令类型标志转移，以区分出指令属于哪一类，如：是单操作数指令，还是双操作数指令等。因为每一类机器指令中操作码字段的位数和位置是固定的，所以第二次即可按操作码区分出具体是哪条指令，以便找出相应微程序的入口微地址。

3.通过PLA电路实现功能转换

当机器指令的操作码位数和位置都不固定时，可以采用PLA电路将每条机器指令的操作码翻译成对应的微程序入口地址。这种方法对于变长度、变位置的操作码显得更有效，而且转换速度较快。

7.5.5后继微地址的形成

找到初始微地址之后，可以开始执行微程序，每条微指令执行完毕都要根据要求形成后继微地址。后继微地址的形成方法对微程序编制的灵活性影响很大，它主要有两大基本类型：增量方式和断定方式。

1.增量方式（顺序－转移型微地址）

这种方式和机器指令的控制方式很类似，它也有顺序执行、转移和转子之分。顺序执行时后继微地址就是现行微地址加上一个增量（通常为1）；转移或转子时，由微指令的顺序控制字段产生转移微地址。因此，在微程序控制器中应当有一个微程序计数器（μPC），为了降低成本，一般情况下都是将微地址寄存器μMAR改为具有计数功能的寄存器，以代替μPC。

2.断定方式

断定方式的后继微地址可由微程序设计者指定，或者根据微指令所规定的测试结果直接决定后继微地址的全部或部分值。

这是一种直接给定与测试断定相结合的方式，其顺序控制字段一般由两部分组成：非测试段和测试段。

(1)非测试段，可由设计者指定，一般是微地址的高位部分，用来指定后继微地址在CM中的某个区域内。

(2)测试段，根据有关状态的测试结果确定其地址值，一般对应微地址的低位部分。这相当于在指定区域内断定具体的分支。所依据的测试状态可能是指定的开关状态、指令操作码、状态字等。

测试段如果只有一位，则微地址将产生两个分支，若有两位，则最多可产生四个分支，依此类推，测试段为n位最多可产生2^n个分支。

7.5.6微程序设计

1.微程序设计方法

（1）水平型微指令及水平型微程序设计

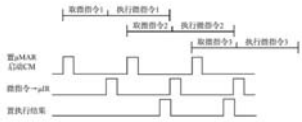
（2）垂直型微指令及垂直型微程序设计

（3）混合型微指令

2.微指令的运行方式

（1）串行方式

（2）并行方式



3.微程序仿真

图7-26 系统仿真时宿主机的主存储器和控制存储器

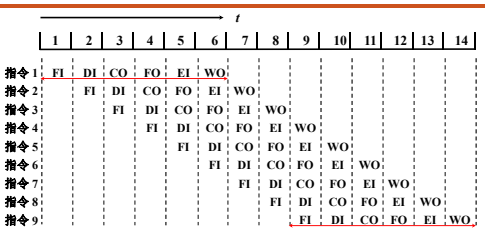
7.6 流水线的基本概念

1）流水线基本原理

流水线技术是一种显著提高指令执行速度与效率的技术。方法是：指令取指完成后，不等该指令执行完毕即可取下一条指令。

如果把一条指令的解释过程进一步细分，例如，把分析、执行两个过程分成取指FI、译码DI、计算操作数地址CO、取操作数FO、执行指令EI和写回寄存器WO六个子过程，并用六个子部件分别处理这六个子过程。

这样只需在上一指令的第一子过程处理完毕进入第二子过程处理时，在第一子部件中就开始对第二条指令的第一子过程进行处理。随着时间推移，这种重叠操作最后可达到六个子部件同时对六条指令的子过程进行操作。



完成 一条指令： 6 个时间单位

串行执行： 6 × 9 = 54 个时间单位

六级流水： 14 个时间单位

7.6.1重叠与先行控制

例：假定一条指令的执行分为三个阶段：取指令、分析、执行。

下图表示了三种执行方式相邻指令间的时序关系。假定各段时间相等。

图 7-27 顺序和重叠执行方式

T=3nt

T=(1+2n)t

T=(2+n)t

7.6.2指令流水线工作原理

1.流水线表示方法

(1)连接图：

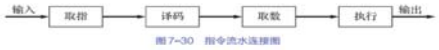


图 7-30 指令流水连接图

(2)时空图：

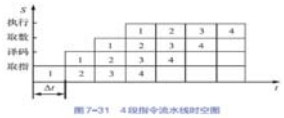


图 7-31 4段指令流水线时空图

2.流水线分类

(1)从流水处理的级别上分类:

部件级: 指构成部件内的各子部件之间的流水。

图 7-32 浮点加法器流水线连接图

指令集流水：指将指令的执行过程分为多个子过程的流水。

处理机级流水：指构成计算机系统的多个处理机之间的流水

图 7-33 处理机级流水线连接图

（2）按功能分类

单功能流水线

指只能实现一种功能的流水线。

多功能流水线

指同一流水线的各个段之间可以有多种不同的联接方式，以实现多种不同的运算或功能。

(3)按工作方式分类

静态流水线

动态流水线

- **静态流水线**
  - **同一时间内它只能以一种功能方式工作。它可以是单功能的，也可以是多功能的。**
  - **当是多功能流水线时，则从一种功能方式变为另一种功能方式时，必须先排空流水线，然后为另一种功能设置初始条件后方可使用。**
  - **不希望这种功能的转换频繁的发生，否则将严重影响流水线的处理效率。**

静态流水线示例

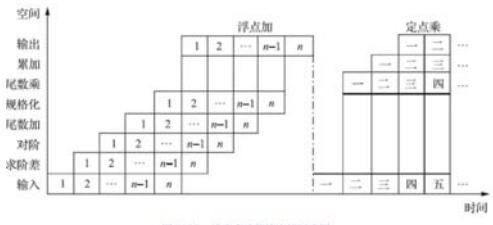


图 7-35 静态多功能流水线时空图

- 动态流水线
  - 允许在同一时间内将不同的功能段连接成不同的功能子集（前提条件是功能部件的使用不发生冲突），以完成不同的运算功能。
  - 动态流水线必是多功能流水线，而单功能流水线则必是静态的。

动态流水线示例

图 7-36 动态多功能流水线时空图

（4）按照流水线的结构分：

线性流水线(Linear Pipelining)：每个流水段都流过一次，且仅流过一次。

非线性流水线(Nonlinear Pipelining)：在流水线的某些流水段之间有反馈回路或前馈回路。

线性流水线能够用流水线连接图唯一表示，非线性流水线必须用流水线连接图流水线预约表等共同表示。

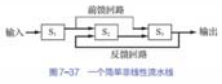


图 7-37 一个简单非线性流水线

7.6.3流水线的主要性能指标

1.吞吐率

流水线的吞吐率（TP）是指在单位时间内流水线所完成的任务数量或输出的结果数量。若流水线T_k时间完成n个任务，则吞吐率为TP = n /T_k。

流水线存在“瓶颈”流水段时，连续输入n个任务的一条k段线性流水线的实际吞吐率为：

$$TP=\frac{n}{\sum_{i=1}^{k}\Delta t_i+(n-1)\max\{\Delta t_1、\Delta t_2、\cdots、\Delta t_k\}}$$

吞吐率

单位时间内 流水线所完成指令 或 输出结果 的 数量

设 m 段的流水线各段时间为Δt

- **最大吞吐率**

$$T_{pmax}=\frac{1}{\Delta t}$$

- **实际吞吐率**

连续处理 n 条指令的吞吐率为

$$T_p=\frac{n}{m\cdot\Delta t+(n-1)\cdot\Delta t}$$

例7-1：有一指令流水线连接图和各段时间消耗如图7-38所示，问该计算机的CPU时钟至少是多少？完成10个任务的吞吐率是多少？最大吞吐率是多少？

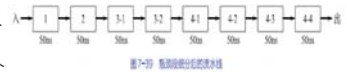
图 7-38 某指令流水线

解：这个指令流水线的各功能段执行时间是不相同的。由于各功能段的时间不同，计算机的CPU时钟周期应当以最长的功能段执行时间为准，max{50,50,100,200}，即CPU时钟至少为200ns。
实际吞吐率TP= 10/[(50+50+100+200)+9×200]=1/220条指令/秒
最大吞吐率Tpmax=1/200条指令/秒

7.6流水线技术

解决流水线瓶颈的方法：

一是将“瓶颈”部分再细分，当分成与其他时间段几乎相等的功能段时，就会每一个时间步(1个时钟周期)出一条指令，如7-39；



另一种是采用“瓶颈”段复制的方法，用数据分配器，将多条指令的“瓶颈”段(访存)并行地执行，加快执行过程。如图7-40，当然复杂度要高。

7.6流水线技术

2.加速比

不使用流水线相对使用流水线所花费的时间比值。

假定不使用流水线所用的时间为T_s，使用流水线的时间为T_p，则流水线的加速比为： $S = T_s / T_p$

当流水线的各个流水段的执行时间为△ti时，一条k段线性流水线完成n个连续任务的实际加速比为：

$$S=\frac{n\cdot\sum_{i=1}^{k}\Delta t_i}{\sum_{i=1}^{k}\Delta t_i+(n-1)\max\{\Delta t_1,\Delta t_2\cdots,\Delta t_k\}}$$

若各个流水线各段执行时间相等，n个任务理想流入并且流出流水线，则上式可以化简为：

$$S=\frac{n\cdot k\Delta t}{(k+n-1)\Delta t}=\frac{n\cdot k}{k+n-1}\qquad S_{max}=\lim_{n\to\infty}\frac{n\cdot k}{n+k-1}=k$$

7.6流水线技术

加速比 S_p

m 段的 流水线的速度 与等功能的 非流水线的速度 之比

设流水线各段时间为 Δt

完成 n 条指令在 m 段流水线上共需

$$T=m\cdot\Delta t+(n-1)\cdot\Delta t$$

完成 n 条指令在等效的非流水线上共需

$$T'=nm\cdot\Delta t$$

则

$$S_p=\frac{nm\cdot\Delta t}{m\cdot\Delta t+(n-1)\cdot\Delta t}=\frac{nm}{m+n-1}$$

7.6流水线技术

例7-2 假定非流水线机器，时钟周期为10ns，ALU和分支操作需要4个时钟周期，存储器操作需要5个时钟周期，以上操作的比例相应为40%，20%，40%，流水线机器上由于存在时钟偏移和启动时间，时钟周期增加了1ns，并忽略其他的影响，求该流水线的加速比。

解：非流水线的机器上，指令的平均执行时间：

指令平均执行时间=时钟周期 × 平均CPI

=10ns× [(40%+20%)× 4 + 40%×5]

=44ns

在流水线方式下，时钟周期为10+1=11ns。

所以加速比为 44/11=4。

7.6流水线技术

3.效率

流水线的效率是指流水线的设备利用率,即流水线中设备的使用时间占整个运行时间之比，也称流水线设备的时间利用率。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非常方便。

各段的执行时间不相等的k段线性流水线，完成n个连续任务的效率为：

$$E=\frac{n\cdot\sum_{i=1}^{k}\Delta t_i}{k\cdot\left[\sum_{i=1}^{k}\Delta t_i+(n-1)\max\{\Delta t_1,\Delta t_2\cdots,\Delta t_k\}\right]}$$

各段时间相等可化简为

$$E=\frac{n\cdot k\Delta t}{k\cdot(k+n-1)\Delta t}=\frac{n}{k+n-1}$$

从时空图上来看：效率实际上就是n个任务所占用的时空区面积与k个段所占用的总的时空区的面积之比。当n>>k时，效率E趋近于1。

7.6流水线技术

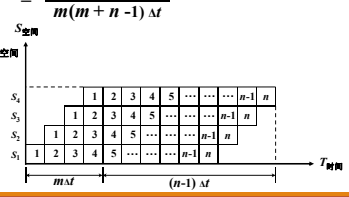
效率

指流水线中各功能段的 利用率

由于流水线有 建立时间 和 排空时间

因此各功能段的 设备不可能 一直 处于 工作 状态

$$效率 = \frac{流水线各段处于工作时间的时空区}{流水线中各段总的时空区} = \frac{mn\Delta t}{m(m+n-1)\Delta t}$$



7.6.4指令流水线的相关性问题

1.结构相关

由于多条指令在同一时刻争夺同一资源而形成的冲突称为结构相关，也称资源相关。

2.数据相关

后续指令要使用前面指令的操作结果，而这一结果尚未产生或者未送到指定的位置，从而造成后续指令无法运行的局面称为数据相关。

根据指令间对同一个寄存器读或写操作的先后次序关系，数据相关可分为RAW（写后读）、WAR（读后写）和WAW（写后写）3种类型。

3.控制相关

控制相关主要是由转移指令引起的，在遇到条件转移指令时，存在着是顺序执行还是转移执行两种可能，需要依据条件的判断结果来选择其一。在无法确定应该选择把哪一程序段安排在转移指令之后来执行的局面称为控制相关，又称指令相关。

7.6.5流水线相关性解决方案

1.结构相关的解决：可以通过插入阻塞周期暂停流水线，或者通过增加硬件的方式来解决。如：

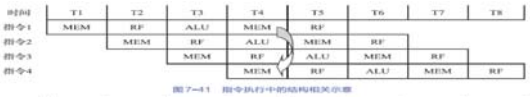


图 7-41 指令执行中的结构相关示意

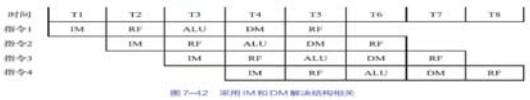


图 7-42 采用IM和DM解决结构相关

2.数据相关的解决：一般WAW冒险和WAR冒险仅存在于某些种类的处理器中，常见的为RAW型数据相关。如：

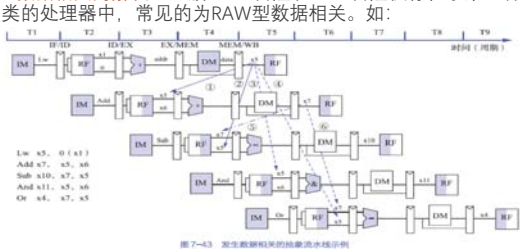


图 7-43 发生数据相关的流水线示例

使用旁路方式解决。

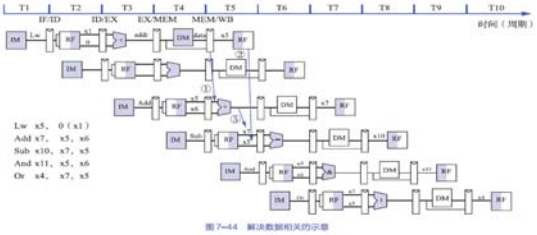


图 7-44 解决数据相关的示意

3.控制相关的解决

控制相关主要由转移指令引起。

两种广泛使用的解决方式：

（1）简单但不高效的相关解决方式：阻塞，即停顿流水线，暂停流水线一个或多个时钟周期。阻塞能解决所有相关。

（2）分支预测。

7.6.6流水线数据通路与控制器设计

数据通路在多周期数据通路的基础上进行改造，增加流水段寄存器。

两种不同类型的控制器：

集中式控制器

分布式控制器

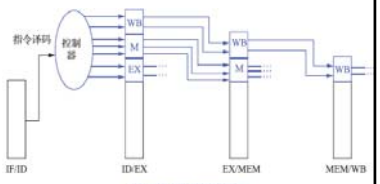


图7-45 集中式控制器示意