第 5 章　中央处理器

中央处理器(CPU)是整个计算机的核心,它包括运算器和控制器。本章着重讨论 CPU 的功能和组成,指令的执行过程、控制器的功能与工作原理以及指令流水线的基本概念。

5.1　408 考纲要求

(一) CPU 的功能和基本结构

(二) 指令执行过程

(三) 数据通路的功能和基本结构

(四) 控制器的功能和工作原理

1. 硬布线控制器

2. 微程序控制器

微程序、微指令和微命令,微指令格式,微命令的编码方式,微地址的形成方式。

(五) 指令流水线

1. 指令流水线的基本概念

2. 指令流水线的基本实现

3. 超标量和动态流水线的基本概念

5.2 知识结构梳理

5.2.1 CPU的功能和基本结构

1. CPU的功能

CPU负责协调并控制计算机各部件执行程序的指令序列，并对数据进行加工。

2. CPU的基本结构

CPU由运算器和控制器两大部分组成。运算器是对信息进行处理和运算的部件，它的主要功能是执行所有的算术运算和逻辑运算；控制器是整个计算机的指挥中心，它的主要功能是按照人们预先确定的操作步骤，控制整个计算机的各部件有条不紊地自动工作。

3. CPU中的寄存器

CPU中的寄存器是用来暂时保存运算和控制过程中的中间结果、最终结果以及控制信息和状态信息的，可分为通用寄存器和专用寄存器两大类。

5.2.2 指令执行过程

一条指令的执行过程通常可以分为3个阶段：取指令阶段、分析取数阶段和执行阶段。

5.2.3 数据通路的功能和基本结构

数据通路是数据在功能部件之间传送的路径，路径上的部件称为数据通路部件，如算术逻辑单元(ALU)、通用寄存器等。数据通路的功能是实现CPU内部的运算器与寄存器以及各寄存器之间的数据交换。不同计算机的数据通路可以是完全不同的，只有明确了计算机的数据通路，才能确定相应的微操作控制信号。

控制器在实现一条指令的功能时，将每条指令分解成为一系列时间上先后有序的最基本、最简单的微操作，形成微操作序列。微操作序列是与CPU的内部数据通路密切相关的，不同的数据通路就有不同的微操作序列。

5.2.4 控制器的功能和工作原理

1. 硬布线控制器

传统的控制器称为硬布线控制器，是采用组合逻辑技术来实现的，其控制单元是由门电路组成的复杂树形网络。这种方法是分立元件时代的产物，以使用最少的器件和取得最高的操作速度为设计目标。

硬布线控制器的最大优点是速度快。但是控制单元的结构不规整，使得设计、调试、维修较困难，难以实现设计自动化；控制单元构成之后，要想增加新的控制功能是不可能的。

2. 微程序控制器

微程序控制器是采用存储逻辑实现的，也就是把微操作信号代码化，使每条机器指令转化成为一段微程序并存入一个专门的存储器（控制存储器）中，微操作控制信号由微指令产生。

微程序控制器的设计思想和硬布线控制器的设计思想截然不同。微程序控制器具有设计规整、调试、维修以及更改、扩充指令方便的优点，易于实现自动化设计，但是，由于它增加了一级控制存储器，所以指令的执行速度比硬布线控制器慢。

5.2.5 指令流水线

1. 指令流水线的基本概念

流水线是将一个较复杂的处理过程分成 m 个复杂程度相当、处理时间大致相等的子过程，每个子过程由一个独立的功能部件来完成，处理对象在各子过程连成的线路上连续流动。

2. 指令流水线的基本实现

在同一时间，m 个部件同时进行不同的操作，完成对不同对象的处理。这种方式类似于现代工厂的生产流水线，在那里每隔一段时间（Δt）从流水线上流出一个产品，而生产这个产品的总时间要比 Δt 大得多。由于流水线上各部件并行工作，机器的吞吐率将大大提高。

3. 超标量和动态流水线的基本概念

超标量技术是通过重复设置多个功能部件，并让这些功能部件同时工作来提高指令的执行速度，实际上是以增加硬件资源为代价来换取处理器性能的。使用超标量技术的处理器在一个时钟周期内可以发射多条指令。

多功能流水线按工作方式可分为动态流水线和静态流水线两种。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，以完成不同的功能；而静态流水线在同一时间内各段只能以一种功能连接流水。

5.3 重点难点分析

1. CPU 的基本结构

CPU 由运算器和控制器两大部分组成。图 5-1 给出了 CPU 的模型。

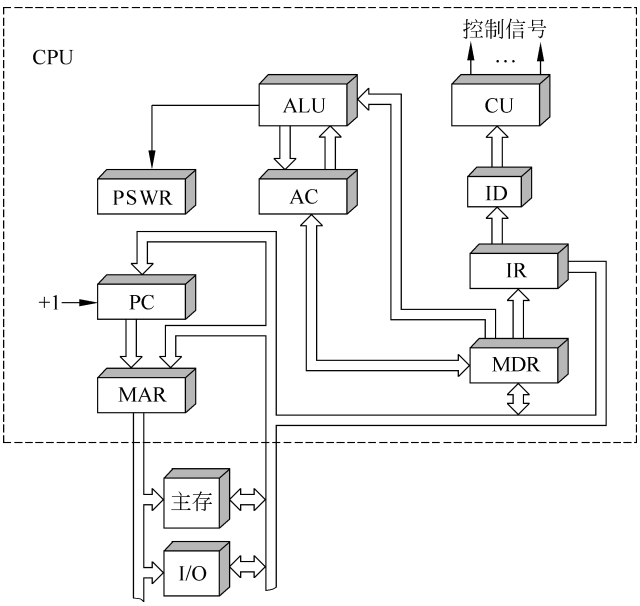

图 5-1 CPU 模型

在图 5-1 中，算术逻辑部件(ALU)和累加寄存器(AC)属于运算器，它们的主要功能是执行所有的算术运算和逻辑运算。程序计数器(PC)、指令寄存器(IR)、程序状态字寄存器(PSWR)、指令译码器(ID)和控制单元(CU)都属于控制器，它们的主要功能是正确地从主存中取出指令，并对指令进行译码或测试，产生相应的操作控制信号。虚线框内还有存储器地址寄存器(MAR)和存储器数据寄存器(MDR)，它们是主存和 CPU 的接口，通常也属于 CPU 的范畴。

2. CPU 中寄存器的设置

1) 通用寄存器

通用寄存器也就是用于处理的寄存器，它们可提供操作数并存放运算结果，或作为地址指针，或作为基址寄存器、变址寄存器，或作为计数器，等等。在指令系统中为这些寄存器分

配了编号，可以编程指定使用其中的某个寄存器，对程序员来说是“看得见”的寄存器。

在对这组寄存器的设计上，有的计算机将它们设计成基本通用的，如 PDP-11 中的通用寄存器命名为 R_0，R_1，R_2，…，它们可被指定完成各种工作，大部分寄存器没有特定的任务。有的计算机则为这组寄存器分别规定了某一基本任务，并按各自的基本任务命名，如 Intel 80x86 中设置了累加器 AX、基址寄存器 BX、数据寄存器 DX 等。

CPU 中还常设置一些用户不能直接访问的寄存器组，用来暂存信息，称为暂存器。在指令系统中没有为它们分配编号，因而不能直接编程访问。对程序员来说，它们是看不见的。

2）专用寄存器

专用寄存器是专门用来完成某一种特殊功能的寄存器。CPU 至少有 5 个专用寄存器，它们又分为用于控制的寄存器和用于主存接口的寄存器。

控制寄存器在 CPU 中起着控制操作的作用。控制寄存器有以下 3 个：

（1）程序计数器（PC），又称为指令指针（IP），用来存放指令地址。为了保证程序能自动连续执行，CPU 必须能自动确定下一条指令的地址。在程序开始执行前，将程序的第一条指令所在的存储单元地址送入 PC；在程序执行过程中，CPU 将自动修改 PC 的内容，使其保存的总是将要执行的下一条指令的地址。顺序执行时，PC 增量计数（加 1）；遇到转移指令，则将转移地址送至 PC。所谓 PC 加 1 中的“1”，代表的是一条指令，而不一定是一个存储单元，这取决于主存的编址方式，若主存按字节编址，则增量值为指令所占的字节数，即，指令占 1 字节时 PC＋1，指令占 2 字节时 PC＋2，以此类推。

（2）指令寄存器（IR），用来存放现行指令。当执行一条指令时，首先从主存将指令取出，送到指令寄存器中去，直到这一条指令执行完毕，再放入下一条指令。为了提高指令的执行速度，现在大多数计算机都将指令寄存器扩充为指令队列（指令栈），允许预取若干条指令。

（3）程序状态字寄存器（PSWR），用来存放程序状态字。程序状态字的各位表征程序和计算机运行的状态，它是参与控制程序执行的重要依据之一。它主要包括两部分内容：一是状态标志，例如进位标志（C）、结果为零标志（Z）等，许多指令的执行将会自动修改这些标志；二是控制标志，例如中断标志、陷阱标志等。

主存接口寄存器用于 CPU 与主存储器的数据交换。主存接口寄存器有以下两个：

（1）存储器地址寄存器（MAR），用来接收指令地址（PC 的内容）、操作数地址或结果数据地址，以确定要访问的单元。

（2）存储器数据寄存器（MDR），也可称为存储器数据缓冲寄存器（MBR）。它是向主存写入数据或从主存读出指令或数据的缓冲部件。

3. 指令执行的基本过程

一条指令执行的基本过程通常可以分为 3 个阶段：取指令阶段、分析取数阶段和执行阶段。

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。具体操

作如下：

(1) 将程序计数器中的内容送至存储器地址寄存器，并送至地址总线。

(2) 向存储器发读命令。

(3) 从主存中取出的指令通过数据总线送到存储器数据寄存器。

(4) 将存储器数据寄存器的内容送至指令寄存器。

(5) 将程序计数器的内容递增，为取下一条指令做好准备。

以上这些操作对任何一条指令来说都是必须执行的，所以称为公共操作。完成取指令阶段任务的时间称为取指周期。

2) 分析取数阶段

取出指令后，指令译码器可识别和区分出不同的指令类型。此时计算机进入分析取数阶段，以获取操作数。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。

对于无操作数指令，只要识别出是哪一条具体的指令即可转执行阶段。而有操作数指令还需要读取操作数，首先要计算出操作数的有效地址。如果操作数在通用寄存器中，则不需要再访问主存；如果操作数在主存中，则要到主存中取数。对于不同的寻址方式，有效地址的计算方法是不同的，有时要多次访问主存才能取出操作数。另外，单操作数指令和双操作数指令由于需要的操作数的个数不同，分析取数阶段的操作也不同。完成分析取数阶段任务的时间又可以细分为间址周期、取数周期等。

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。完成执行阶段任务的时间称为执行周期。

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

4. CPU 在取指周期将 PC 加 1 的原因

通常，CPU 在取指周期中要递增 PC 的值，为什么 CPU 要在取指周期中递增 PC 的值？如果不增加会出现什么样的结果呢？例如，假设 CPU 从 10 号地址单元取某条指令，在取指周期它完成的操作如下：

```
(PC)→MAR
Read
M(MAR)→MDR→IR
(PC)+1→PC
```

取出指令之后，CPU 对指令进行译码，然后执行该指令，接下来返回取指周期继续取下一条指令，如果此时 PC 的值仍然是 10，那么 CPU 将不断地取出、分析、执行同一条指令！

下一条将被执行的指令存放在 11 号地址单元中。事实上，CPU 只要在它返回取指令阶段之前将 PC 加 1 即可。为了实现它，设计者有两种方案可以选择：①在取指周期使 PC 加 1；②在执行周期使 PC 加 1。相比之下，方案①比方案②的实现要容易得多，因为取指周

期是公共操作,所有指令的取指操作都是相同的;而执行周期各条指令是不相同的,有的复杂,有的简单。所以 CPU 通常采用在取指周期将 PC 加 1 的方案。

另外,现代计算机普遍采用流水线技术,第 i 条指令的执行阶段是与第 $i+1$ 条指令的分析阶段和第 $i+2$ 条指令的取指阶段同时进行的,如果不在取指周期进行 PC 加 1,同样会不断地取出、分析、执行同一条指令,指令流水线将无法正常工作,这是采用流水线技术的 CPU 必须在取指周期将 PC 加 1 的另一个原因。

5. 数据通路与控制信号

数据通路是 CPU 中算术逻辑单元(ALU)、控制单元(CU)以及寄存器之间的连接线路。不同计算机的数据通路可以是完全不同的。只有明确了机器的数据通路,才能确定相应的微操作控制信号。事实上,要写出指令的微操作控制信号,首先需要给出相应的 CPU 结构和数据通路图,严格按要求建立信息在计算机各部件之间流动的时间和空间关系,而不是凭空编造。如果 CPU 内部采用单总线结构,还要考虑总线冲突的问题,相应的微操作控制信号必须与给出的数据通路结构一致,且时序上要有先后顺序。

控制器在实现一条指令的功能时,总要把每条指令分解成为一系列时间上先后有序的最基本、最简单的微操作,即微操作序列。微操作序列是与 CPU 的内部数据通路密切相关的,不同的数据通路就有不同的微操作序列。

一条指令的取出和执行可以分解成很多最基本的操作,这种不可再分割的最基本的操作称为微操作。微操作信号发生器也称为控制单元。真正控制各部件工作的微操作信号是由指令部件提供的操作信号、时序部件提供的时序信号、被控制功能部件反馈的状态及条件信号综合形成的。不同的机器指令具有不同的微操作序列。

6. 控制器的核心——控制单元

控制器主要由以下几部分组成:

(1) 指令部件。包括程序计数器、指令寄存器、指令译码器、地址形成部件等。

(2) 时序部件。包括脉冲源、启停控制逻辑、节拍信号发生器等。

(3) 微操作信号发生器(控制单元)。

(4) 中断控制逻辑。

需要注意的是,存放在指令寄存器中的指令,其操作码部分经译码后才能识别出当前要执行的指令是一条什么样的指令,也就是说,指令译码器仅对指令中的操作码字段进行译码,而不对整条指令进行译码。

控制器的核心是控制单元(即微操作信号发生器),计算机无论执行什么任务,都是在控制单元的控制下完成的。控制单元通常有 3 种实现方法:组合逻辑电路、存储逻辑电路和可编程逻辑阵列(PLA)。根据控制单元实现方法的不同,控制器可分为组合逻辑(硬布线)型、存储逻辑型、组合逻辑与存储逻辑结合型 3 种。这 3 种控制器仅仅是控制单元的实现不同,而控制器中的其他部分基本上是大同小异的。

控制单元的输入包括时序信号、机器指令操作码、各部件状态反馈信号等,输出的微操作控制信号又可以细分为 CPU 内的控制信号和送至主存或外设的控制信号。CPU 内部的

控制信号用于控制寄存器间的数据传送，使 ALU 完成指定的功能以及其他的内部操作。发送至 CPU 外部的控制信号用于控制 CPU 与主存和外设交换数据。

控制单元的一般模型如图 5-2 所示，该模型表示了控制单元的输入和输出信号之间的关系。

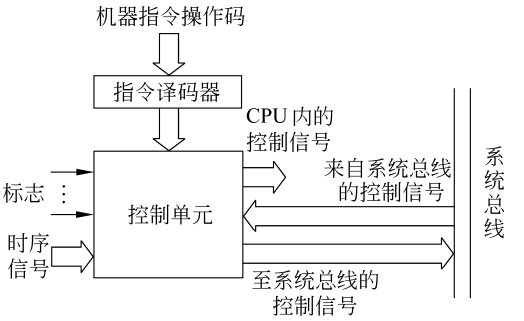

图 5-2　控制单元模型

7. 组合逻辑控制和微程序控制的比较

组合逻辑(硬布线)控制和微程序控制的主要区别在于控制单元的实现方法不同。具体说明如下：

(1) 组合逻辑控制的控制功能是由组合逻辑电路实现的，由于各个微操作控制信号的逻辑表达式的繁简程度不同，由此组成的控制电路零乱、复杂。而微程序控制的控制功能是由存放微程序的控制存储器和存放当前正在执行的微指令的寄存器实现的，控制电路比较规整。

(2) 对组合逻辑控制来说，因为所有控制信号的逻辑表达式用硬连线固定下来，当需要修改和增加指令时就很麻烦，有时甚至可能需要重新进行设计。而在微程序控制中，各条指令的微操作控制信号的差别仅反映在控制存储器的内容上。如果想扩展和改变计算机的功能，只需在控制存储器中增加新的微指令或修改某些原来的微指令即可。

(3) 在同样的半导体工艺条件下，组合逻辑控制的速度比微程序控制的速度快。这是因为组合逻辑控制的速度主要取决于逻辑电路的延迟，而微程序控制执行的每条微指令都要从控制存储器中读取，影响了速度。

8. 微程序控制的基本概念

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，形成控制字(微指令)，再把这些微指令按时间先后排列起来构成微程序，存放在一个只读的控制存储器中。

下面给出微程序设计技术涉及的基本术语。

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。在微程序控制的计算机中，将控制部件向执行部件发出的各种控制命令叫作微命令。因此，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。

微命令有兼容性微命令和互斥性微命令之分。兼容性微命令是指那些可以同时产生，共同完成某些微操作的微命令；而互斥性微命令是指在计算机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。对于一个微命令，谈论其兼容和互斥都是没有意义的。

2）微指令和微地址

微指令是指控制存储器中的一个单元的内容，即控制字，是若干微命令的集合。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：

（1）操作控制字段，又称微操作码字段，用来产生某一步操作所需的各微操作控制信号。

（2）顺序控制字段，又称微地址码字段，用来控制产生下一条要执行的微指令的地址。

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。每一条机器指令都对应一个微程序。

微程序和程序是两个不同的概念。微程序是由微指令组成的，用于描述机器指令。微程序实际上是机器指令的实时解释器，是由计算机的设计者事先编制好并存放在控制存储器中的，一般不提供给用户。对于程序员来说，计算机系统中微程序一级的结构和功能是透明的，无须知道。而程序最终由机器指令组成，是由软件设计人员事先编制好并存放在主存或辅存中的。所以说，微程序控制的计算机涉及两个层次：一个是机器语言或汇编语言程序员所看到的传统机器层，包括机器指令、工作程序和主存储器；另一个是计算机设计者看到的微程序层，包括微指令、微程序和控制存储器。

9. 微指令操作控制字段的编码法

1）直接控制法

在微指令的操作控制字段中，每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 则表示这个微命令无效。微命令的产生不必经过译码，所需的控制信号直接送到相应的控制点。

这种方法结构简单，并行性强，操作速度快；但是微指令字太长，使控存单元的位数过多，而且在给定的任何一个微指令中往往仅使用了部分微命令，造成信息利用率下降。

2）最短编码法

最短编码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，则最短编码法应满足下列关系式：

$$L \geqslant \log_2 N$$

最短编码法的微指令字长最短，但要通过一个微命令译码器译码以后才能得到需要的微命令。这种方法在同一时刻只能产生一个微命令，不能充分利用计算机硬件所具有的并

行性，使得机器指令对应的微程序变得很长，而且对于某些要求在同一时刻同时动作的组合性微操作将无法实现。

3）字段编码法

字段编码法是前述两种编码法的一个折中的方法，既具有它们的优点，又克服了它们的缺点。这种方法将操作控制字段分为若干段，每段内采用最短编码法，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。

在字段编码法中，操作控制字段的分段并非是任意的，必须遵循如下原则：

（1）应把互斥的微命令分在同一段内，将兼容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。

（2）应与数据通路结构相适应。

（3）每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）一般每段还要留出一个状态，表示本段不发出任何微命令。因此，当某段的长度为 3 位时，最多只能表示 7 个互斥的微命令，通常用 000 表示不操作。

10. 微程序控制计算机的基本结构和工作过程

微程序控制计算机与组合逻辑控制计算机的主要差别在于控制单元的不同，微程序控制计算机的控制单元部分用一个完整的存储系统（包括控制存储器、微指令寄存器、微地址寄存器等）代替了组合逻辑控制网络。图 5-3 给出了微程序控制器控制单元的基本结构。

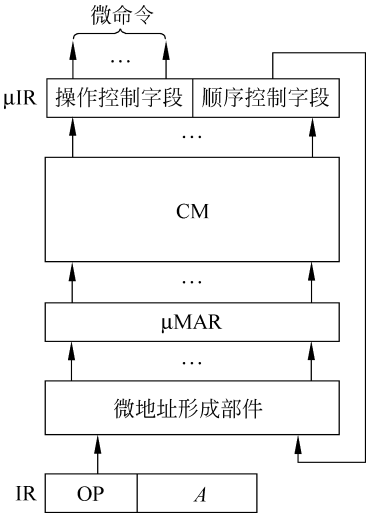

图 5-3 微程序控制器控制单元的基本结构

注意：*这里所说的存储系统与第 3 章中提到的存储系统不是一回事，这里的存储系统属于 CPU 的一部分。*

控制存储器（CM）是微程序控制器的核心部件，用来存放微程序。其性能（包括容量、速度、可靠性等）与计算机的性能密切相关。

微指令寄存器（μIR）用来存放从 CM 取出的正在执行的微指令，它的位数同微指令字长相等。

微存储器地址寄存器（μMAR）用于接收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。

微地址形成部件用来产生初始微地址和后继微地址，以保证微指令的连续执行。

微程序控制器的工作过程实际上就是计算机在微程序控制器的控制下执行机器指令的过程，这个过程可以描述如下：

（1）执行取指令公共操作。具体的操作是：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，用来控制计算机实现取机器指令的公共操作。取指微程序的入口地址一般