

电子数字计算机原理

第二册

北京大学计算机科学技术系 编著

科学出版社

电子数字计算机原理

第二册

北京大学计算机科学技术系 编著

科学出版社

1982

内 容 简 介

《电子数字计算机原理》第二册共分四章，第一章介绍常规控制的计算机的运算器和控制器，有三个附录，介绍加法器设计，两种舍入规则和计算机速度估算的一种方法。第二章介绍微程序控制的计算机中央处理器和微程序设计技术，有四个附录，分别介绍模型机指令表，增码规则，模型机微命令和部分微程序。第三章介绍故障诊断概念，讨论布尔差分对组合线路和时序线路的故障诊断，还介绍奇偶校验与奇偶预测。第四章介绍大型机设计的几个重要问题，如交叉存贮、先行控制、快速运算、数组处理机和多道程序，存贮器管理等。

本书可作为计算机专业的教学参考书，也可供有关部门的科技工作者阅读。

电子数字计算机原理

第二册

北京大学计算机科学技术系 编著

*

科 学 出 版 社 出 版

北京朝阳门内大街 137 号

中 国 科 学 院 印 刷 厂 印 刷

新华书店北京发行所发行 各地新华书店经售

*

1980年11月第一版 开本 850×1168 1/32

1982年2月第二次印刷 印张 22 7/8 插页 3

印数：23,701—37,900 字数：606,000

统一书号：15031·247

本社书号：1511·15—8

定 价： 2.90 元

编 者 的 话

《电子数字计算机原理》第二册是为计算机专业学生写的一本教学参考书,其主要目的是让学生了解与计算机的中央处理器(即运算器和控制器)有关的一些基本的总体设计和逻辑设计的知识。计算机总体设计的内容很多,涉及的面很广,限于篇幅不可能作较多的介绍,另外如果仅仅讲一些总体设计的思想则可能使初学者感到抽象,影响对具体的中央处理器的掌握。因此,我们以一些计算机为背景,既讲一些总体设计的思想,又讲一些由这些思想而产生的逻辑设计方法。这样也许便于初学者把握具体的计算机的概貌和了解一些总体思想以及它的实现方法。

全书分四章,前两章有各自的附录。

第一章介绍常规控制方式的计算机中央处理器,其中第一、二、三节和第十节讨论了总体设计中的几个基本问题,如指令、地址方式、数据、总框图和中断系统。其他各节侧重于逻辑设计,如控制的组织、数据流程的组织等。加法器的设计是很重要的内容,本应放在本书的第一册中论述,在此我们以附录的形式出现。

第二章介绍微程序控制的计算机和微程序设计,还对系列机,“概念上的计算机”等作了说明。若读者只想了解微程序计算机,可以看完第一章的一、二、三节和第十节之后,直接读第二章。本章的一些附录可为那些想深入掌握微程序设计的人们提供一些材料。

第三章介绍 RAS 技术中奇偶检测与故障诊断的部分内容。

第四章讨论高速大型机中的几个重要问题。其中第五节的多道程序、存贮保护等内容也可以放在第一章,读者可以在看完第一章后提前看本章第五节。

本书由周炜、祝明发和郑胜利三同志执笔。它不仅反映了我

们学校的教学经验和科研成果，而且还吸收了兄弟单位的科研成果。在编写过程中，参考了1972年王攻本、杨天锡和许卓群同志写的《电子数字计算机原理》(讲义)第二册，采用了其中的某些内容；同时参考了1975年王攻本和宋万寿同志未完的讲义修改稿，也采用了其中的某些内容(如加法器设计等)。此外，还参考了200系列机和其它一些机关的有关资料。在写作过程中与校内外许多同志进行过认真讨论，听取了许多宝贵意见。黄险峰、张华和张静波同志已参加了第一章的编审工作。在此对他们给予的热情支持深表谢意。

本书于1976年夏动笔，1977年底脱稿。因时间关系加之我们的业务水平有限，书中的缺点、错误和不妥之处在所难免，希望读者批评指出。

1980年元月

目 录

第一章 控制器与运算器	1
第一节 数、指令与指令系统	1
一、数	1
二、指令	6
三、指令系统	15
第二节 地址的形成	36
一、操作地址的形成	36
二、指令地址的形成	48
三、间接地址	52
第三节 运算器与控制器的逻辑结构	54
一、运算器	55
二、控制器的信息通路	57
三、时标系统(SB)	61
四、微操作控制部件(WK)	62
五、中断系统	63
第四节 时标系统	63
一、指令周期与节拍	63
二、指令的基本周期	64
三、基本周期的节拍划分	65
四、指令的重叠	69
五、局部操作	70
六、脉冲(节拍)分配器	72
七、启停控制	74
第五节 公共操作	77
一、为什么要单独考虑公共操作	77
二、模型机的公共操作及其安排	79
第六节 运算器数据通路	85
一、寄存器	85

二、加法器 Q	90
第七节 运算型指令举例	93
一、代码传送指令	94
二、浮点规舍加法指令	95
三、浮点规舍乘法指令	103
四、逻辑运算	109
五、浮点全字长比较指令	113
第八节 转移型指令举例	115
一、逢负转移指令 ($\uparrow\omega = 1$)	115
二、转子指令和返回指令	116
三、循环开门和循环关门指令	117
第九节 操作时间表和微操作综合	121
一、操作时间表	121
二、微操作的综合	121
第十节 中断系统	141
一、中断的概念	141
二、中断的种类	145
三、中断的记录方式	148
四、中断的建立、中断的开放与屏蔽	150
五、中断的响应	154
六、中断的处理	163
附录 I 加法器	167
一、进位链	196
二、单重分组跳跃进位	173
三、多重分组跳跃进位	176
四、和数的形成	180
五、加法器设计举例	183
附录 II 舍入规则	197
一、基本舍入方法	198
二、两种舍入方法的比较	199
三、“0”舍“1”入规则的改进	204
附录 III 计算机运算速度估算	205
第二章 微程序	208
第一节 什么是微程序	208

一、组合逻辑控制(常规控制)方式	210
二、微程序控制方式	213
三、一个微程序设计的例子——双倍字长积整数乘法	215
第二节 微指令编码	219
一、分段显式编码	220
二、分段隐式编码	224
第三节 微指令地址	228
一、微程序基本操作流程	228
二、S-Z 型微地址	233
三、C-D 型微地址	234
四、微地址举例	236
第四节 模型机	238
一、模型机结构要点	238
二、模型机的框图	249
三、模型机的时标系统	256
四、模型机的微指令格式	264
第五节 微程序循环	268
第六节 公用微程序	278
一、对阶微子程序	279
二、微子程序的引用方法	284
三、规格化公用微程序	285
第七节 模型机公共操作微程序	289
一、取指令微程序	289
二、单字整数、逻辑数类指令取操作数微程序 ($\theta = 5X, DX$)	293
三、长浮点数、双字整数类指令取操作数微程序 ($\theta = 7X, FX$)	300
四、指令重叠执行	300
五、单字整数加微程序 ($\theta = 5A, DA$)	304
第八节 模型机微程序举例	307
一、单字逻辑乘微程序 ($\theta = 54, D4$)	307
二、32 位整数乘微子程序	309
三、单字整数乘(双字积)微程序 ($\theta = 5E, DE$)	310
四、短浮点减微程序 ($\theta = 6B, EB$)	313
五、长浮点除微程序 ($\theta = 7D, FD$)	314
六、长浮点乘微程序 ($\theta = 7C, FC$)	317
七、转移组指令微程序	323

八、“执行”指令微程序 ($\theta = 4D, CD$)	323
九、响应二级中断微程序	329
第九节 微中断	331
一、什么叫微中断	331
二、微中断的过程	333
三、一个微中断的例子	334
结束语	339
附录 I 模型机指令表 (部分)	341
附录 II 增码及其算法	371
附录 III 模型机微命令表	375
附录 IV 模型机部分微程序	398
第三章 奇偶校验与故障诊断	477
第一节 故障诊断的概念	479
一、为什么考虑故障诊断	479
二、几个术语	481
三、通路敏化法	482
四、故障定位	490
第二节 布尔差分及其在故障测试中的应用	497
一、异或运算	497
二、什么是布尔差分	498
三、布尔差分的性质与基本公式	500
四、用布尔差分产生组合线路的测试集	513
五、用布尔差分产生两种触发器的测试集	525
第三节 奇偶校验	540
一、什么是奇偶校验	540
二、为什么采用奇偶校验	543
三、译码器的奇偶校验	546
四、计数器的奇偶预测	548
五、加法器的奇偶预测	553
第四章 大型计算机的若干问题	567
概述	567
一、中央处理机和内存速度的匹配	567
二、中央处理机的并行重叠操作	572
第一节 访内分配器 数存	580

一、内存组织	581
二、访内分配器	584
三、数存	593
第二节 先行控制	602
一、先行指令栈	603
二、关于先行指令寄存器的个数	608
三、先行操作码栈与先行读数栈	610
四、后行写数栈	616
五、设计先行控制器应考虑的几个问题	618
第三节 快速运算	628
一、多运算部件、多累加寄存器结构	628
二、快速运算部件	638
第四节 算术运算流水线和数组处理机简介	671
一、算术运算流水线	671
二、流水线数组处理机	681
三、并行数组处理机	694
第五节 多道程序运行	698
一、多道程序运行的必要性	698
二、实现多道程序运行必须解决哪些问题	703
三、存贮分配、程序浮动、存贮保护	706

第一章 控制器与运算器

控制器与运算器是电子数字计算机的重要部件之一。随计算机的使用范围不同其逻辑结构有所不同，我们仅就用于科学计算的计算机讨论其控制器与运算器的有关问题。

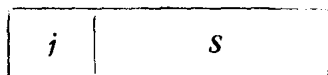
第一节 数、指令与指令系统

一、数

在计算机的数字系统中，一般设有浮点数、定点数、整数和逻辑数。所有这些数均用一串二进制代码表示，而在机器内部则表现为一串电信号。为了便于对下面的问题展开讨论，我们分别对浮点数、定点数、整数和逻辑数作一简单介绍。

1. 浮点数

浮点数由阶码 (j) 和尾数 (S) 组成，它在机器中的形式为



如果浮点数以 2 为底则可写成下面的表达式

$$N = 2^j \times S$$

浮点数的阶码 (j) 和尾数 (S) 各定为多少位取决于对计算机所要求的取值范围和精度。

若一计算机其浮点数的阶码 j 用八位二进制补码表示，其中有一位阶码符号 j_7 ，七位阶码可写成下面的形式

$$[j]_{\text{补}} = j_7 j_6 j_5 j_4 j_3 j_2 j_1$$

阶码 j 用补码表示时其取值范围见表 1.1。

这种情况下阶码取值范围是

$$-128 \leq j \leq 127.$$

表 1.1

	f_7	f_6	f_5	f_4	f_3	f_2	f_1	f_0	十进制数值
正数	0	1	1	1	1	1	1	1	2的127次方
	0	1	1	1	1	1	1	0	2的126次方
	0	1	1	1	1	1	0	1	2的125次方
									⋮
	0	0	0	0	0	0	0	1	2的1次方
	0	0	0	0	0	0	0	0	2的0次方
负数	1	1	1	1	1	1	1	1	2的-1次方
	1	1	1	1	1	1	1	0	2的-2次方
	1	1	1	1	1	1	0	1	2的-3次方
									⋮
	1	0	0	0	0	0	0	0	2的-128次方

若一浮点数阶码部份为8位;尾数 S 为40位,其中有一位符号位 S_f , S_w 有39位,用补码形式表示,那么尾数 S_w 的取值范围是

表 1.2

$S_f \backslash S_w$	S_1	S_2	S_3, \dots, S_{38}	S_{39}	小数值
0	1	1	1.....1	1	$1-2^{-39}$
0	1	1	1.....1	0	$1-2^{-38}$
0	1	1	1.....0	1	$1-(2^{-38}+2^{-39})$
⋮			⋮		⋮
0	0	0	0.....0	1	2^{-39}
0	0	0	0.....0	0	0
1	1	1	1.....1	1	-2^{-39}
1	1	1	1.....1	0	-2^{-38}
⋮			⋮		⋮
1	0	0	0.....0	0	-1

即: $-1 \leq S \leq 1 - 2^{-39}$.

注意尾数符号 S_f 表示浮点数 N 是正值还是负值。上述48位

浮点数 N 最大正值为

$$N_{\max} = 2^{+127} \times (1 - 2^{-39})$$

其最小负值为

$$N_{\min} = -2^{+127}$$

那么它对应于十进制数的范围为

$$|N| \leq 2^{127} \approx 10^{38.2}$$

(设 $2^{127} = 10^x$, 则 $\log_{10} 2^{127} = \log_{10} 10^x$, 因此, $x = 127 \log_{10} 2 \approx 38.2$)

由此可见, 阶码位数决定浮点数的取值范围。阶码位数越多对应的浮点数取值范围越大, 反之则小。根据实际情况, 8 位阶码基本够用, 所以大部分机器采用 8 位阶码, 也有少数机器用到 11 位阶码的。

尾数 S_w 的位数主要用来决定浮点数的精度即有效数字的位数。上述的尾数 S_w 为 39 位二进制数, 相当于十进制数的 11.7 位, 也就是说上例中的浮点数折合为十进制数最多可能有 11.7 位有效数字。计算机中浮点数的精度是由计算对象的要求决定的。

目前常用的浮点数的长度有以下几种(如表 1.3):

注意一个浮点数的取值范围与有效数字的位数, 它们是两个不同概念。前者由阶码长度决定, 而后者由 S_w 长度决定。

表 1.3

字长	阶符 i_f	阶码	数符 S_f	尾数	数值范围	精度(有效数字)
24 位	1	7	1	15	$-10^{38.2} - 10^{38.2}$	十进制 4.5 位
32 位	1	7	1	23	同上	十进制 6.9 位
48 位	1	7	1	39	同上	十进制 11.7 位
64 位	1	7	1	55	同上	十进制 16.6 位

现在有一些计算机的浮点数中阶码用增码表示。

若阶码仍为 8 位, 用增码表示, 其表达式为

$$N = 2^i \times 2^{128} \times S = 2^{i+128} \times S$$

用补码表示和用增码表示的对应关系如表 1.4。

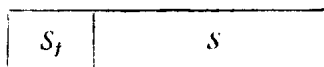
表 1.4

阶码为补码	阶码为增码	取 值
$f_7 f_6 f_5 f_4 f_3 f_2 f_1$	$f_7 f_6 f_5 f_4 f_3 f_2 f_1$	
0 1 1 1 1 1 1	1 1 1 1 1 1 1	2 的 127 次方
0 1 1 1 1 1 0	1 1 1 1 1 1 0	2 的 126 次方
⋮	⋮	⋮
0 0 0 0 0 0 1	1 0 0 0 0 0 1	2 的 1 次方
0 0 0 0 0 0 0	1 0 0 0 0 0 0	2 的 0 次方
1 1 1 1 1 1 1	0 1 1 1 1 1 1	2 的 -1 次方
1 1 1 1 1 1 0	0 1 1 1 1 1 0	2 的 -2 次方
⋮	⋮	⋮
1 0 0 0 0 0 1	0 0 0 0 0 0 1	2 的 -127 次方
1 0 0 0 0 0 0	0 0 0 0 0 0 0	2 的 -128 次方

阶码采用增码表示时，增码的最小值（即全 0）正好表示了所能表示的真值的最小值（注意在补码表示阶码时，此时阶码为 10000000，不是全“0”）。这样判“0”线路简单。关于增码的运算规则见第二章附录 II。

2. 定点数、整数

数 N 的表示为



S_f 为数的符号（1 位）。

S 为数值部份。

若规定小数点在 S_f 与 S 之间，则 N 为定点数。

若规定小数点在数 N 的最后，则 N 为整数。

采用定点运算的计算机在运算时无需对阶，所以结构简单。定点数若与浮点位数相同，由于无阶码部份，因而有效位数多，精度高。但是定点数所表示的数的范围比浮点制要小得多。一台计算机是采用浮点制还是定点制主要由机器的规模和应用场合所决定。通常用于自动控制系统的专用计算机及小型计算机大多采用

定点制,而科学计算用大、中型计算机大多采用浮点制。

机器中存贮器单元号码(即地址码)、变址值和计数器的值,都可以看成是无符号位的正整数。

机器中的浮点数或定点数长度确定之后,整数长度与之相适应,或与其等长,或为它的一半,或为它的双倍长。

用补码表示的 24 位长的定点数与整数的取值情况见表 1.5。

表 1.5

$S_7 S_1 S_2 S_3 \dots S_{23} S_{24}$	整数值	小数值
0 1 1 1 1 1	$2^{23} - 1 = 8388607$	$1 - 2^{-23}$
0 1 1 1 1 0	$2^{23} - 2 = 8388606$	$1 - 2^{-22}$
⋮	⋮	⋮
0 0 0 0 0 1	1	$+2^{-23}$
0 0 0 0 0 0	0	0
1 1 1 1 1 1	-1	-2^{-23}
1 1 1 1 1 0	-2	-2^{-22}
⋮	⋮	⋮
1 0 0 0 0 0	$-2^{23} = -8388608$	-1

3. 逻辑数

一般说来,逻辑数包括两种。一种是机器中用来表示状态的二进制代码,如控制触发器,控制器中某些寄存器的信息(条件码 ω ,程序状态字……)等等,这些二进制代码只具有数的形式而无正负之分,有的也无大小之分。另一种是表示一个字符的二进制代码也是逻辑数,它纯粹是编码形式。由于逻辑数的性质不同于前面的浮点数,定点数,所以它的运算法则也不同。

现在国内使用的一种字符编码规则如表 1.6。

表 1.6

				W_7	0	0	0	0	1	1	1	1
				W_6	0	0	1	1	0	0	1	1
				W_5	0	1	0	1	0	1	0	1
W_4	W_3	W_2	W_1	列 行	0	1	2	3	4	5	6	7
0	0	0	0	0	空白 KE	转移 ZY	间隔	0	ω	P	“	p
0	0	0	1	1	序始 XS	机控 ₁ JK ₁		1	A	Q	a	q
0	0	1	0	2	文始 WS	机控 ₂ JK ₂	”	2	B	R	b	r
0	0	1	1	3	文终 WZ	机控 ₃ JK ₃	#	3	C	S	c	s
0	1	0	0	4	送毕 SB	机控 ₄ JK ₄	≠	4	D	T	d	t
0	1	0	1	5	询问 XW	否认 FR	%	5	E	U	e	u
0	1	1	0	6	承认 CR	同步 TB	∧	6	F	V	f	v
0	1	1	1	7	告警 GJ	组终 ZZ	,	7	G	W	g	w
1	0	0	0	8	退格 TG	作废 ZF	(8	H	X	h	x ₂
1	0	0	1	9	横表 HB	载终 ZTZ)	9	I	Y	i	y
1	0	1	0	A	换行 HH	取代 QD	★	:	J	Z	j	z
1	0	1	1	B	纵表 ZB	扩展 KZ	+	;	K	[k	←
1	1	0	0	C	换页 HY	卷隙 JX	.	<	L	∨	l	
1	1	0	1	D	回车 HC	群隙 QX	-	=	M]	m	→
1	1	1	0	E	移出 YC	录隙 LX	o	>	N	↑	n	—
1	1	1	1	F	移入 YR	元隙 YX	/	?	O	_	o	抹掉

二、指 令

一个教练员训练队伍时是靠教练员用声音发出一道道口令(如“立正”、“跑步走”、“卧倒”等),指挥整个队伍协调地完成一项项动作的。而计算机完成某种操作是由一串特殊的电信号(称为控制信息)控制机器的某些部件而达到的。计算机中的这种控

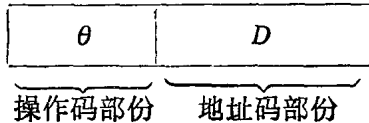
编码举例

字 符	八进制代码	二进制代码
A	41	100,0001
B	42	100,0010
5	35	011,0101
⋮	⋮	⋮

制信息就是指令。不同的控制信息表示不同的指令，一条指令能使计算机完成某种操作，如算术运算中的 $+$ ， $-$ ， \times ， \div ，逻辑运算中的逻辑乘、逻辑加，数码的传送，程序的分支(即转移)等。

一条指令完成某种控制功能，它必须包括的信息有两大部份：一部份是告诉计算机做哪一种操作；另一部份是要告诉机器从内存的那些单元取出参与运算的数(即操作数)，这次运算的结果放到何处去。前一部份叫操作码部份，后一部份叫地址码部份。

一条指令也是一串信息，它的特殊性是有控制作用(计算机中的数也是一种信息，但无控制作用)，然而作为信息的普遍性来说，它都可以用一串二进制代码来表示。指令的代码形式可写成



操作码 θ 一般占用五至八位二进制代码。 θ 占用位数取决于所需要的指令的多少，若 θ 为 N 位，最多可表示 2^N 种指令。

地址码部份变化较多，随计算机的不同有许多不同的形式。

指令的具体形式与计算机内部结构有密切关系，不同的机器结构有不同的指令形式，我们着重讨论单累加器结构的机器的指令形式。

单累加器结构的机器的运算器一般由三个寄存器(即累加器 L ，乘商寄存器 S 和接收寄存器 C)和一个加法器组成[参看本书第一册293页运算器框图]。在使用上，累加器 L 处于比较特殊的