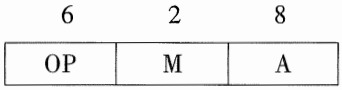


其中，OP 为操作码，6 位不变；M 为寻址模式，2 位不变；A 为位移量 8 位，对应位移量为-128～+127。

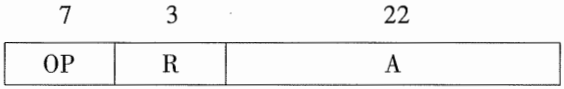
例 7.7 设某机共能完成 110 种操作，CPU 有 8 个通用寄存器（16 位），主存容量为 4 M 字，采用寄存器-存储器型指令。

（1）欲使指令可直接访问主存的任一地址，指令字长应取多少位？画出指令格式。

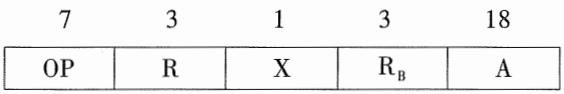
（2）若在上述设计的指令字中设置一寻址特征位 X，且 X=1 表示某个寄存器作基址寄存器，画出指令格式。试问基址寻址可否访问主存的任一单元？为什么？如果不能，提出一种方案，使其可访问主存的任一位置。

（3）若主存容量扩大到 4 G 字，且存储字长等于指令字长，则在不改变上述硬件结构的前提下，可采用什么方法使指令可访问存储器的任一位置？

解：（1）欲使指令可直接访问 4 M 字存储器的任一单元，采用寄存器-存储器型指令，该机指令应包括 22 位的地址码、3 位寄存器编号和 7 位操作码，即指令字长取 22+3+7=32 位，指令格式如下：



（2）在上述指令格式中增设一寻址特征位，且 X=1 表示某个寄存器作基址寄存器 R_B。其指令格式如下：



由于通用寄存器仅 16 位，形式地址 18 位，不足以覆盖 4 M 地址空间，可将 R_B 寄存器内容左移 6 位，低位补 0，形成 22 位基地址，然后与形式地址相加，所得的有效地址即可访问 4 M 字存储器的任一单元。

（3）若主存容量扩大到 4 G 字，且存储字长等于指令字长，则在不改变上述硬件结构的前提下，采用一次间接寻址即可访问存储器的任一单元，因为间接寻址后得到的有效地址为 32 位，$2^{32}=4$ G。

7.5 RISC 技术

RISC 即精简指令系统计算机（Reduced Instruction Set Computer），与其对应的是 CISC，即复杂指令系统计算机（Complex Instruction Set Computer）。

7.5.1 RISC的产生和发展

计算机发展至今,机器的功能越来越强,硬件结构越来越复杂。尤其是随着集成电路技术的发展及计算机应用领域的不断扩大,计算机系统的软件价格相对而言在不断提高。为了节省开销,人们希望已开发的软件能被继承、兼容,这就希望新机种的指令系统和寻址方式一定能包含旧机种所有的指令和寻址方式。通过向上兼容不仅可降低新机种的开发周期和代价,还可吸引更多的新、老用户,于是出现了同类型的系列机。在系列机的发展过程中,致使同一系列计算机指令系统变得越来越复杂,某些机器的指令系统竟可包含几百条指令。例如,DEC公司的VAX-11/780有16种寻址方式、9种数据格式、303条指令。又如,32位的68020微型计算机指令种数比6800多两倍,寻址方式多11种,达18种之多,指令长度从一个字(16位)发展到16个字。这类机器被称为复杂指令系统计算机,简称CISC。

通常对指令系统的改进都是围绕着缩小与高级语言语义的差异和有利于操作系统的优化而进行的。由于编写编译器的人们的任务是为每一条高级语言的语句编制一系列的机器指令,如果机器指令能类似于高级语言的语句,显然编写编译器的任务就变得十分简单了。于是人们产生了用增加复杂指令的办法来缩短与语义的差距。后来又发现,倘若编译器过多依赖复杂指令,同样会出现新的矛盾。例如,对减少机器代码、降低指令执行数以及为提高流水性能而优化生成代码等都是非常不利的。尤其当指令过于复杂时,机器的设计周期会很长,资金耗费会更大。例如,Intel 80386 32位机器耗资达1.5亿美元,开发时间长达三年多,结果正确性还很难保证,维护也很困难。最值得一提的例子是,1975年IBM公司投资10亿美元研制的高速机器FS机,最终以“复杂结构不宜构成高速计算机”的结论宣告研制失败。

为了解决这些问题,20世纪70年代中期,人们开始进一步分析研究CISC,发现一个80-20规律,即典型程序中80%的语句仅仅使用处理机中20%的指令,而且这些指令都是属于简单指令,如取数、加、转移等。这一点告诫人们,付出再大的代价增添复杂指令,也仅有20%的使用概率,而且当执行频度高的简单指令时,因复杂指令的存在,致使执行速度也无法提高。表7.3是HP公司对IBM 370高级语言中指令使用频度的分析结果。Marathe在1978年对PDP-11机在五种不同应用领域中的指令进行混合测试,也得出了类似的结论。

表7.3 IBM 370机指令的使用频度(%)

指令类型	转移	逻辑	数据存取	存-存传送	整数运算	浮点运算	十进制运算	其他
COBOL	24.6	14.6	40.2	12.4	6.4	0.0	1.6	0.6
FORTRAN	18.0	8.1	48.7	2.1	11.0	11.9	0.0	0.2
Pascal	18.4	9.9	54.0	4.8	7.0	6.8	0.0	0.1

另一方面,在20世纪70年代末80年代初,计算机的器件已进入VLSI时代,复杂的指令系统需要复杂的控制器,这需要占用较多的芯片面积。统计表明,典型的CISC计算机中,控制器约占60%的芯片面积,而且使设计、验证和实现都更加困难。

人们从80-20规律中得到启示:能否仅仅用最常用的20%的简单指令,重新组合不常用的80%的指令功能呢?这便引发出RISC技术。

1975年IBM公司John Cocke提出了精简指令系统的设想,1982年美国加州大学伯克利分校的研究人员专门研究了如何有效利用VLSIC(超大规模集成电路)的有效空间。RISC由于设计的指令条数有限,相对而言,它只需用较小的芯片空间便可制作逻辑控制电路,更多的芯片空间可用来增强处理机的性能或使其功能多样化。他们用大部分芯片空间做成寄存器,并且用它们作为暂时数据存储的快速存储区,从而有效地降低了RISC机器在调用子程序时所需付出的时间。他们研制的RISC Ⅰ(后来又出现RISC Ⅱ),采用VLSI CPU芯片上的晶体管数量达44 000个,线宽为3 μm,字长为32位,其中有128个寄存器(而用户只能见到32个),仅有31条指令和两种寻址方式,访存指令只有两条,即取数(LOAD)和存数(STORE)。显然其指令系统极为简单,但它们的功能已超过VAX-11/780和M68000,其速度比VAX-11/780快了1倍。

与此同时,美国斯坦福大学RISC研究的课题是MIPS(Micro Processor without Interlocking Pipeline Stages),即消除流水线各段互锁的微处理器。他们把IBM公司对优化编译程序的研究与加州大学伯克利分校对VLSI有效空间利用的思想结合在一起,最终的研究成果后来转化为MIPS公司RX000的系列产品。IBM公司又继其IBM801型机、IBM RT/PC后,于1990年推出了著名的IBM RS/6000系列产品。加州大学伯克利分校的研究成果最后发展成Sun微系统公司的RISC芯片,称为SPARC(Scalable Processor ARChitecture)。

到目前为止,RISC体系结构的芯片可以说已经历了3代:第一代以32位数据通路为代表,支持Cache,软件支持较少,性能与CISC体系结构的产品相当,如RISC Ⅰ、MIPS、IBM801等。第二代产品提高了集成度,增加了对多处理机系统的支持,提高了时钟频率,建立了完善的存储管理体系,软件支持系统也逐渐完善。它们已具有单指令流水线,可同时执行多条指令,每个时钟周期发出一条指令(有关流水线的概念详见8.3节)。例如,MIPS公司的R3000处理器,时钟频率为25 MHz和33 MHz,集成度达11.5万个晶体管,字长为32位。第三代RISC产品为64位微处理器,采用了巨型计算机或大型计算机的设计技术——超级流水线(Superpipelining)技术和超标量(Superscalar)技术,提高了指令级的并行处理能力,每个时钟周期发出2条或3条指令,使RISC处理器的整体性能更好。例如,MIPS公司的R4000处理器采用50 MHz和75 MHz的外部时钟频率,内部流水时钟达100 MHz和150 MHz,芯片集成度高达110万个晶体管,字长64位,并有16 KB的片内Cache。它有R4000PC、R4000SC和R4000MC三种版本,对应不同的时钟频率,分别提供给台式系统、高性能服务器和多处理器环境下使用。表7.4列出了MIPS公司R系列RISC处理器的几项指标。

表 7.4 MIPS 公司 R 系列 RISC 处理器比较

机 种	R2000	R3000	R4000
宣布时间	1986	1988	1991
时钟频率	16.67 MHz	25/33 MHz	50/75 MHz
芯片规模	10 万晶体管	11.5 万晶体管	110 万晶体管
结构形式	流水线	流水线	超级流水线
寄存器集	32×32 位	32×32 位	32×64 位, 16×64 位
片上 Cache	—	—	16 KB
片外 Cache	最大 128 KB	最大 512 KB	128 KB~4 MB
工艺	2 μm CMOS	1.2 μm CMOS	0.8 μm CMOS
功耗	3W	3.5W	
SPEC 分	11.2	17.6(25 MHz)	63(50 MHz)

自 1983 年开始出现商品化的 RISC 机以来,比较著名的 RISC 机有 IBM 公司的 IBM RT 系列,HP 公司的精密结构计算机(HPPA)、MIPS R3000、Motorola M88000、Intel 80960、INMOS Transputer、AMD AM29000、Fairchild Clipper 等。其中,Clipper 兼顾了 RISC 和 CISC 两方面的特点,又称为类 RISC 机。在计算机工作站方面,Sun Microsystems 公司于 1987 年推出 SPARC,速度达7~10 MIPS。1988 年 Apollo 公司推出 Series 10000 个人超级计算机,称为并行精简指令系统多处理机 PRISM(Parallel Reduced Instruction Set Multiprocessor),单机系统速度达 15~25 MIPS,四处理机则可达 60~100 MIPS,后来 HP 合并了 Apollo 公司,继续发展工作站。

较为著名的第三代 RISC 机的有关性能指标如表 7.5 所示。

表 7.5 第三代 RISC 处理器的性能比较

机 种	R4000	Alpha	Motorola 88110	Super SPARC	RS/6000	i860	C400
公司名称	MIPS	DEC	Motorola	Sun/TI	IBM	Intel	Intergraph
时钟频率 (MHz)	50/75	150/200	50	50/100	33	25/40/50	50
集成度 (万晶体管)	110	168	130	310	120	255	30
结构形式	超流水线	超标量	超标量	超标量	超标量	超长指令	超标量
寄存器集	32×64 16×64 浮	32×64 32×64 浮	32×64 32×64 (32×80)	32×32	32×64 32×64	32×32 16×64	32×32 16×64

续表

机 种	R4000	Alpha	Motorola 88110	Super SPARC	RS/6000	i860	C400
片上 Cache	16 KB	16 KB	16 KB	36 KB	8 KB	32 KB	
片外 Cache	128 KB ~1 MB	最大可达 8 MB	256 KB ~1 MB	2 MB			128 KB
工艺	0.8 μm CMOS	0.75 μm CMOS	1 μm CMOS	0.8 μm CMOS			
功耗		23 W		8 W	4 W		7 W
SPEC 分	63 (50 MHz)	100(估计)	63.7(估计)	75(估计)	25.9	42	42

注:表中空项待查。

7.5.2 RISC 的主要特征

由上分析可知,RISC 技术是用 20% 的简单指令的组合来实现不常用的 80% 的那些指令功能,但这不意味着 RISC 技术就是简单地精简其指令集。在提高性能方面,RISC 技术还采取了许多有效措施,最有效的方法就是减少指令的执行周期数。

计算机执行程序所需的时间 P 可用下式表述:

$$P = I \times C \times T$$

其中,I 是高级语言程序编译后在机器上运行的机器指令数;C 为执行每条机器指令所需的平均机器周期;T 是每个机器周期的执行时间。

表 7.6 列出了第二代 RISC 机与 CISC 机的 I、C、T 统计,其中 I、T 为比值,C 为实际周期数。

表 7.6 RISC/CISC 的 I、C、T 统计比较

	I	C	T
RISC	1.2~1.4	1.3~1.7	<1
CISC	1	4~10	1

由于 RISC 指令比较简单,用这些简单指令编制出的子程序来代替 CISC 机中比较复杂的指令,因此 RISC 中的 I 比 CISC 多 20%~40%。但 RISC 的大多数指令仅用一个机器周期完成,C 的值比 CISC 小得多。而且 RISC 结构简单,完成一个操作所经过的数据通路较短,使 T 值也大大下降。因此总折算结果,RISC 的性能仍优于 CISC 2~5 倍。

由于计算机的硬件和软件在逻辑上的等效性,使得指令系统的精简成为可能。曾有人在 1956 年就证明,只要用一条“把主存中指定地址的内容同累加器中的内容求差,把结果留在累加

器中并存入主存原来地址中”的指令，就可以编出通用程序。又有人提出，只要用一条“条件传送(CMOVE)”指令就可以做出一台计算机；并且在1982年某大学做出了一台8位的CMOVE系统结构样机，称为SIC(单指令计算机)。而且，指令系统所精简的部分可以通过其他部件以及软件(编译程序)的功能来替代，因此，实现RISC技术是完全可能的。

1. RISC的主要特点

通过对RISC各种产品的分析，可归纳出RISC机应具有如下一些特点。

① 选取使用频度较高的一些简单指令以及一些很有用但又不复杂的指令，让复杂指令的功能由频度高的简单指令的组合来实现。

② 指令长度固定，指令格式种类少，寻址方式种类少。

③ 只有取数/存数(LOAD/STORE)指令访问存储器，其余指令的操作都在寄存器内完成。

④ CPU中有多个通用寄存器。

⑤ 采用流水线技术，大部分指令在一个时钟周期内完成。采用超标量和超流水线技术，可使每条指令的平均执行时间小于一个时钟周期。

⑥ 控制器采用组合逻辑控制，不用微程序控制。

⑦ 采用优化的编译程序。

值得注意的是，商品化的RISC机通常不会是纯RISC机，故上述这些特点不是所有RISC机全部具备的。

相比之下，CISC的指令系统复杂庞大，各种指令使用频度相差很大；指令字长不固定，指令格式多，寻址方式多；可以访存的指令不受限制；CPU中设有专用寄存器；绝大多数指令需要多个时钟周期方可执行完毕；采用微程序控制器；难以用优化编译生成高效的目标代码。

表7.7列出了一些RISC机指令系统的指令条数。

表7.7 一些RISC机的指令条数

机器名	指令数	机器名	指令数
RISC Ⅱ	39	ACORN	44
MIPS	31	INMOS	111
IBM 801	120	IBM RT	118
MIRIS	64	HPPA	140
PYRAMID	128	CLIPPER	101
RIDGE	128	SPARC	89

下面以RISC Ⅱ为例，着重分析其指令种类和指令格式。

2. RISC Ⅱ指令系统举例

(1) 指令种类

RISC Ⅱ共有39条指令，分为以下4类。

① 寄存器-寄存器操作:移位、逻辑、算术(整数)运算等 12 条。

② 取/存数指令:取存字节、半字、字等 16 条。

③ 控制转移指令:条件转移、调用/返回等 6 条。

④ 其他:存取程序状态字 PSW 和程序计数器等 5 条。

在 RISC Ⅱ机中,有一些常用指令未被选中,但用上述这些指令并在硬件系统的辅助下,足以实现其他一些指令的功能。例如,该机约定 R_0 寄存器内容恒为 0,这样加法指令可替代寄存器间的传送指令,即

$(R_s)+(R_0)\rightarrow R_d$,替代了 $R_s\rightarrow R_d$

加法指令还可替代清除寄存器指令,即

$(R_0)+(R_0)\rightarrow R_d$,替代了 $0\rightarrow R_d$

减法指令可替代取负数指令,即

$(R_0)-(R_s)\rightarrow R_d$,替代了 R_d 寄存器内容取负

此外,该机可用立即数作为一个操作数,这样当立即数取 1 时,再用加法(或减法)指令就可替代寄存器内容增 1(减 1)指令,即

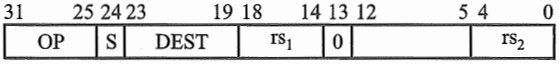
$(R_s)+1\rightarrow R_d$

当立即数取-1 时,异或指令可替代求反码指令,即

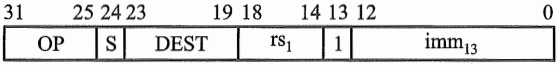
$R_s\oplus(-1)\rightarrow R_d$替代 $\overline{R}_s\rightarrow R_d$

(2) 指令格式

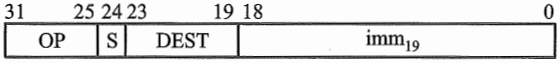
RISC 机的指令格式比较简单,寻址方式也比较少,如 RISC Ⅱ的指令格式有两种:短立即数格式和长立即数格式。指令字长固定为 32 位,指令字中每个字段都有固定位置,如图 7.23 所示。



(a) 第二源操作数在寄存器中的短立即数格式



(b) 第二源操作数为 imm_{13} 的短立即数格式



(c) 长立即数格式

图 7.23　RISC Ⅱ的指令格式

短立即数格式指令主要用于算逻运算,其中第 31 位~25 位为操作码;两个操作数一个在 rs_1 中,另一个操作数的来源由指令的第 13 位决定。当其为 0 时(如图 7.23(a)所示),第二个操作数在寄存器 rs_2 中(只用 0~4 位);当其为 1 时(如图 7.23(b)所示),第二个操作数为 13 位的立

即数 imm_{13}。运算结果存放在 DEST 所指示的寄存器 r_d 中(共 32 个)。指令字中的第 24 位 S 用来表示是否需要根据运算结果置状态位,S=1 表示置状态位。RISC Ⅱ机有 4 个状态位,即零标志位 Z、负标志位 N、溢出标志位 V、进位标志位 C。

指令中的 DEST 字段在条件转移指令中用第 22~19 位作为转移条件,第 23 位无用。对于图 7.23(b)所示的短立即数指令格式,其 imm_{13}即为相对转移位移量。

长立即数指令格式主要用于相对转移指令,此时 19 位的立即数 imm_{19}指出转移指令的相对位移量,与 13 位相比,可扩大相对于 PC 的转移距离。

(3) 寻址方式

RISC Ⅱ指令系统有两种访存寻址方式。一种是变址寻址,另一种是相对寻址,还可用组合方式产生其他寻址方式。若令变址寄存器内容为 0(因该机约定寄存器 R_0 内容恒为 0,所以只要指定 R_0 作为变址寄存器即可实现),则成为直接寻址方式;若令位移量为 0,则成为寄存器间接寻址方式。

对于 LOAD 指令,可根据计算所得的有效地址,从存储器中读取数据并送入 DEST 字段中指示的目的寄存器中。如短立即数指令有效地址为$(rs_1)+(rs_2)$,或为$(rs_1)+imm_{13}$。

对于 STORE 指令,是将 DEST 字段指示的源寄存器中的数取出并存入存储器中,有效地址的计算与 LOAD 指令相同。

3. RISC 指令系统的扩充

从实用角度出发,商品化的 RISC 机,因用途不同还可扩充一些指令,例如:

① 浮点指令,用于科学计算的 RISC 机。为了提高机器速度而增设浮点指令。

② 特权指令,为了便于操作系统管理机器,防止用户破坏机器的运行环境而特设特权指令。

③ 读后置数指令,完成读—修改—写,用于寄存器与存储单元交换数据等。

④ 一些简单的专用指令。例如,某些指令用得较多,实现起来又比较复杂,若用子程序来实现,占用较多的时间,则可考虑设置一条指令来缩短子程序执行时间。有些机器用乘法步指令来加快乘法运算的执行速度。

7.5.3 RISC 和 CISC 的比较

与 CISC 机相比,RISC 机的主要优点可归纳如下:

1. 充分利用 VLSI 芯片的面积

CISC 机的控制器大多采用微程序控制(详见第 10 章),其控制存储器在 CPU 芯片内所占的面积为 50%以上(如 Motorola 公司的 MC68020 占 68%)。而 RISC 机控制器采用组合逻辑控制(详见第 10 章),其硬布线逻辑只占 CPU 芯片面积的 10%左右。可见它可将空出的面积供其他功能部件用,例如用于增加大量的通用寄存器(如 Sun 微系统公司的 SPARC 有 100 多个通用寄存器),或将存储管理部件也集成到 CPU 芯片内(如 MIPS 公司的 R2000/R3000)。以上两种芯片的集成度分别小于 10 万个和 20 万个晶体管。

随着半导体工艺技术的提高，集成度可达100万至几百万个晶体管，此时无论是CISC还是RISC都将多个功能部件集成在一个芯片内。但此时RISC已占领了市场，尤其是在工作站领域占有明显的优势。

2. 提高计算机运算速度

RISC机能提高运算速度，主要反映在以下5个方面。

① RISC机的指令数、寻址方式和指令格式种类较少，而且指令的编码很有规律，因此RISC的指令译码比CISC的指令译码快。

② RISC机内通用寄存器多，减少了访存次数，可加快运行速度。

③ RISC机采用寄存器窗口重叠技术，程序嵌套时不必将寄存器内容保存到存储器中，故又提高了执行速度。

④ RISC机采用组合逻辑控制，比采用微程序控制的CISC机的延迟小，缩短了CPU的周期。

⑤ RISC机选用精简指令系统，适合于流水线工作，大多数指令在一个时钟周期内完成。

3. 便于设计，可降低成本，提高可靠性

RISC机指令系统简单，故机器设计周期短，如美国加州大学伯克利分校的RISC Ⅰ机从设计到芯片试制成功只用了十几个月，而Intel 80386处理器(CISC)的开发花了三年半时间。

RISC机逻辑简单，设计出错可能性小，有错时也容易发现，可靠性高。

4. 有效支持高级语言程序

RISC机靠优化编译来更有效地支持高级语言程序。由于RISC指令少，寻址方式少，使编译程序容易选择更有效的指令和寻址方式，而且由于RISC机的通用寄存器多，可尽量安排寄存器的操作，使编译程序的代码优化效率提高。例如，IBM的研究人员发现，IBM 801(RISC机)产生的代码大小是IBM S/370(CISC机)的90%。

有些RISC机(如Sun公司的SPARC)采用寄存器窗口重叠技术，使过程间的参数传送加快，且不必保存与恢复现场，能直接支持调用子程序和过程的高级语言程序。表7.8列出了一些CISC与RISC微处理器的特征。

表7.8　一些CISC与RISC微处理器的特征

特　征	CISC			RISC	
	IBM 370/168	VAX11/780	Intel 80486	Motorola 88000	MIPS R4000
开发年份	1973	1978	1989	1988	1991
指令数	208	303	235	51	94
指令字长/B	2~6	2~57	1~11	4	32
寻址方式	4	22	11	3	1
通用寄存器数	16	16	8	32	32
控制存储器容量/Kb	420	480	246	—	—
Cache容量/Kb	64	64	8	16	128

此外,从指令系统兼容性看,CISC 大多能实现软件兼容,即高档机包含了低档机的全部指令,并可加以扩充。但 RISC 机简化了指令系统,指令数量少,格式也不同于老机器,因此大多数 RISC 机不能与老机器兼容。

PowerPC 是 IBM、Apple、Motorola 三家公司于 1991 年联合,用 Motorola 的芯片制造经验、Apple的微型计算机软件支持、IBM 的体系结构及其世界计算机市场霸主的地位,向长期被 Intel 占据的微处理器市场挑战而开发的 RISC 产品。

PowerPC 中的"PC"意为"Powerful Chip",其中"Power"基于 20 世纪 80 年代后期,IBM 在其 801 小型机的基础上开发的工作站和服务器中的 Power 体系,意为"Performance Optimization With Enhanced RISC(性能优化的增强型 RISC)"。PowerPC 具有超高的性能、价廉、易仿真 CISC 指令集、可运行大量的现代 CISC 计算机应用软件,即集工作站的卓越性能、PC 机的低成本及运行众多的软件等优点于一身。此外,PowerPC 扩展性强,可覆盖 PDA(个人数字助理)到多处理、超并行的中大型机,用单芯片提供整个解决方案。

多年来计算机体系结构和组织发展的趋势是增加 CPU 的复杂性,即使用更多的寻址方式及更加专门的寄存器等。RISC 的出现象征着与这种趋势根本决裂,自然地引起了 RISC 与 CISC 的争端。随着技术不断发展,RISC 与 CISC 还不能说是截然不同的两大体系,很难对它们做出明确的评价。最近几年,RISC 与 CISC 的争端已减少了很多。原因在于这两种技术已逐渐融合。特别是芯片集成度和硬件速度的增加,RISC 系统也越来越复杂。与此同时,在努力挖掘最大性能的过程中,CISC 的设计已集中到和 RISC 相关联的主题上来,例如增加通用寄存器数以及更加强调指令流水线设计,所以更难去评价它们的优越性了。

RISC 技术发展很快,有关 RISC 体系结构、RISC 流水、RISC 编译系统、RISC、CISC 和 VLIW (Very Long Instruction Word,超长指令字)技术的融合等方面的资料不少。读者若想深入了解,可以查阅有关文献。

思考题与习题

7.1 什么叫机器指令?什么叫指令系统?为什么说指令系统与机器的主要功能以及与硬件结构之间存在着密切的关系?

7.2 什么叫寻址方式?为什么要学习寻址方式?

7.3 什么是指令字长、机器字长和存储字长?

7.4 零地址指令的操作数来自哪里?在一地址指令中,另一个操作数的地址通常可采用什么寻址方式获得?各举一例说明。

7.5 对于二地址指令而言,操作数的物理地址可安排在什么地方?举例说明。

7.6 某指令系统字长为 16 位,地址码取 4 位,试提出一种方案,使该指令系统有 8 条三地址指令、16 条二地址指令、100 条一地址指令。

7.7 设指令字长为 16 位,采用扩展操作码技术,每个操作数的地址为 6 位。如果定义了 13 条二地址指令,

试问还可安排多少条一地址指令?

7.8 某机指令字长16位,每个操作数的地址码为6位,设操作码长度固定,指令分为零地址、一地址和二地址三种格式。若零地址指令有 M 种,一地址指令有 N 种,则二地址指令最多有几种?若操作码位数可变,则二地址指令最多允许有几种?

7.9 试比较间接寻址和寄存器间接寻址。

7.10 试比较基址寻址和变址寻址。

7.11 画出先变址再间址及先间址再变址的寻址过程示意图。

7.12 画出"SUB @R1"指令对操作数的寻址及减法过程的流程图。设被减数和结果存于ACC中,@表示间接寻址,R1寄存器的内容为2074H。

7.13 画出执行"ADD *-5"指令(*为相对寻址特征)的信息流程图。设另一个操作数和结果存于ACC中,并假设(PC)=4000H。

7.14 设相对寻址的转移指令占两个字节,第一个字节是操作码,第二个字节是相对位移量,用补码表示。假设当前转移指令第一字节所在的地址为2000H,且CPU每取出一个字节便自动完成(PC)+1→PC的操作。试问当执行"JMP *+8"和"JMP *-9"指令时,转移指令第二字节的内容各为多少?

7.15 一相对寻址的转移指令占3个字节,第一字节是操作码,第二、三字节为相对位移量,而且数据在存储器中采用以高字节地址为字地址的存放方式。假设PC当前值是4000H。试问当结果为0,执行"JZ *+35"和"JZ *-17"指令时,该指令的第二、第三字节的机器代码各为多少?

7.16 某机主存容量为4 M×16位,且存储字长等于指令字长,若该机指令系统可完成108种操作,操作码位数固定,且具有直接、间接、变址、基址、相对、立即等六种寻址方式,试回答以下问题。

(1) 画出一地址指令格式并指出各字段的作用。

(2) 该指令直接寻址的最大范围。

(3) 一次间接寻址和多次间接寻址的寻址范围。

(4) 立即数的范围(十进制表示)。

(5) 相对寻址的位移量(十进制表示)。

(6) 上述六种寻址方式的指令中哪一种执行时间最短,哪一种最长,为什么?哪一种便于程序浮动,哪一种最适合处理数组问题?

(7) 如何修改指令格式,使指令的寻址范围可扩大到4 M?

(8) 为使一条转移指令能转移到主存的任一位置,可采取什么措施?简要说明之。

7.17 举例说明哪几种寻址方式在指令的执行阶段不访问存储器,哪几种寻址方式在指令的执行阶段只需访问一次存储器?完成什么样的指令,包括取指令在内共访问存储器4次?

7.18 某机器共能完成78种操作,若指令字长为16位,试问一地址格式的指令地址码可取几位?若想使指令寻址范围扩大到 2^{16},可采用什么方法?举出三种不同例子加以说明。

7.19 CPU内有32个32位的通用寄存器,设计一种能容纳64种操作的指令系统。假设指令字长等于机器字长,试回答以下问题。

(1) 如果主存可直接或间接寻址,采用寄存器-存储器型指令,能直接寻址的最大存储空间是多少?画出指令格式并说明各字段的含义。

(2) 在满足(1)的前提下,如果采用通用寄存器作基址寄存器,则上述寄存器-存储器型指令的指令格式有何特点?画出指令格式并指出这类指令可访问多大的存储空间?

7.20 什么是 RISC?简述它的主要特点。

7.21 比较 RISC 和 CISC 的异同之处。

7.22 RISC 机中指令简单,有些常用的指令未被选用,它用什么方式来实现这些常用指令的功能,试举例说明。

第 8 章　CPU 的结构和功能

本章从分析 CPU 的功能和内部结构入手，详细讨论机器完成一条指令的全过程，以及为了进一步提高数据的处理能力、开发系统的并行性所采取的流水技术。此外，本章还进一步概括了中断技术在提高整机系统效能方面的作用。通过本章的学习，希望读者对 CPU 在计算机中的地位和作用以及对中断概念的理解比前面章节更加深入。

8.1　CPU 的结构

8.1.1　CPU 的功能

由第 1 章可知，CPU 实质包括运算器和控制器两大部分，第 6 章讨论了计算机内各种运算及相应的硬件配置，这里重点介绍控制器的功能。

对于冯·诺依曼结构的计算机而言，一旦程序进入存储器后，就可由计算机自动完成取指令和执行指令的任务，控制器就是专用于完成此项工作的，它负责协调并控制计算机各部件执行程序的指令序列，其基本功能是取指令、分析指令和执行指令。

1. 取指令

控制器必须具备能自动地从存储器中取出指令的功能。为此，要求控制器能自动形成指令的地址，并能发出取指令的命令，将对应此地址的指令取到控制器中。第一条指令的地址可以人为指定，也可由系统设定。

2. 分析指令

分析指令包括两部分内容：其一，分析此指令要完成什么操作，即控制器需发出什么操作命令；其二，分析参与这次操作的操作数地址，即操作数的有效地址。

3. 执行指令

执行指令就是根据分析指令产生的“操作命令”和“操作数地址”的要求，形成操作控制信号序列（不同的指令有不同的操作控制信号序列），通过对运算器、存储器以及 I/O 设备的操作，执行每条指令。

此外，控制器还必须能控制程序的输入和运算结果的输出（即控制主机与 I/O 设备交换信息）以及对总线的管理，甚至能处理机器运行过程中出现的异常情况（如掉电）和特殊请求（如打

印机请求打印一行字符),即处理中断的能力。

总之,CPU 必须具有控制程序的顺序执行(称**指令控制**)、产生完成每条指令所需的控制命令(称**操作控制**)、对各种操作加以时间上的控制(称**时间控制**)、对数据进行算术运算和逻辑运算(**数据加工**)以及**处理中断**等功能。

8.1.2 CPU 结构框图

根据 CPU 的功能不难设想,要取指令,必须有一个寄存器专用于存放当前指令的地址;要分析指令,必须有存放当前指令的寄存器和对指令操作码进行译码的部件;要执行指令,必须有一个能发出各种操作命令序列的控制部件 CU;要完成算术运算和逻辑运算,必须有存放操作数的寄存器和实现算逻运算的部件 ALU;为了处理异常情况和特殊请求,还必须有中断系统。可见,CPU 可由四大部分组成,如图 8.1 所示。将图 8.1 细化,又可得图 8.2。图中 ALU 部件实际上只对 CPU 内部寄存器的数据进行操作,有关 ALU 的内容已在第 6 章中有所介绍。

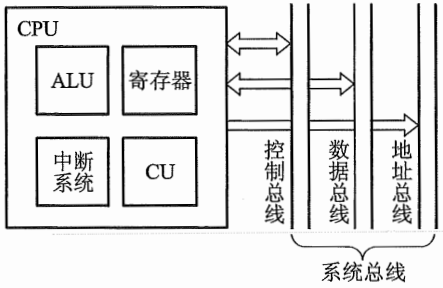


图 8.1 使用系统总线的 CPU

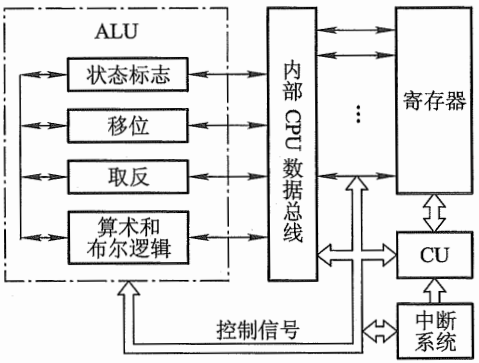


图 8.2 CPU 的内部结构

8.1.3 CPU 的寄存器

第 4 章图 4.2 示出了存储器速度、容量和位价的关系,最上层的寄存器速度最快,容量最小,位价最贵,它们通常设在 CPU 内部。CPU 中的寄存器大致可分两类:一类属于用户可见寄存器,用户可对这类寄存器编程,以及通过优化使 CPU 因使用这类寄存器而减少对主存的访问次数;另一类属于控制和状态寄存器,用户不可对这类寄存器编程,它们被控制部件使用,以控制 CPU 的操作,也可被带有特权的操作系统程序使用,从而控制程序的执行。

1. 用户可见寄存器

通常 CPU 执行机器语言访问的寄存器为用户可见寄存器,按其特征又可分为以下几类。

(1) 通用寄存器

通用寄存器可由程序设计者指定许多功能,可用于存放操作数,也可作为满足某种寻址方式所需的寄存器。例如,基址寻址所需的基址寄存器、变址寻址所需的变址寄存器和堆栈寻址所需的栈指针,都可用通用寄存器代替。寄存器间接寻址时还可用通用寄存器存放有效地址的地址。

当然,也有一些机器用专用寄存器作为基址寄存器、变址寄存器或栈指针,这样,在设计指令格式时只需将这类专用寄存器隐含在操作码中,而不必占用指令字中的位。图 7.15(a)所示的就是用专用寄存器作为基址寄存器,而图 7.15(b)是用通用寄存器作为基址寄存器,所以指令字中必须有 R 字段指出寄存器编号。又如图 7.21 所示的 IBM 360/370 指令格式中,由于用通用寄存器作为变址寄存器和基址寄存器,故在指令字中设有 X 和 B 字段,分别指出作为变址寄存器和基址寄存器的通用寄存器编号。

(2) 数据寄存器

数据寄存器用于存放操作数,其位数应满足多数数据类型的数值范围,有些机器允许使用两个连读的寄存器存放双倍字长的值。还有些机器的数据寄存器只能用于保存数据,不能用于操作数地址的计算。

(3) 地址寄存器

地址寄存器用于存放地址,其本身可以具有通用性,也可用于特殊的寻址方式,如用于基址寻址的段指针(存放基地址)、用于变址寻址的变址寄存器和用于堆栈寻址的栈指针。地址寄存器的位数必须足够长,以满足最大的地址范围。

(4) 条件码寄存器

这类寄存器中存放条件码,它们对用户来说是部分透明的。条件码是 CPU 根据运算结果由硬件设置的位,例如,算术运算会产生正、负、零或溢出等结果。条件码可被测试,作为分支运算的依据。此外,有些条件码也可被设置,例如,对于最高位进位标志 C,可用指令对它置位和复位。将条件码放到一个或多个寄存器中,就构成了条件码寄存器。

在调用子程序前,必须将所有的用户可见寄存器的内容保存起来,这种保存可由 CPU 自动完成,也可由程序员编程保存,视不同机器进行不同处理。

2. 控制和状态寄存器

CPU 中还有一类寄存器用于控制 CPU 的操作或运算。在一些机器里,大部分这类寄存器对用户是透明的。如以下四种寄存器在指令执行过程中起重要作用。

① MAR:存储器地址寄存器,用于存放将被访问的存储单元的地址。

② MDR:存储器数据寄存器,用于存放欲存入存储器中的数据或最近从存储器中读出的数据。

③ PC:程序计数器,存放现行指令的地址,通常具有计数功能。当遇到转移类指令时,PC 的值可被修改。

④ IR:指令寄存器,存放当前欲执行的指令。

通过这 4 个寄存器,CPU 和主存可交换信息。例如,将现行指令地址从 PC 送至 MAR,启动