◆高等教育应用型本科人才培养系列教材

计算机系统结构

JISUANJI XITONG JIEGOU

张佳琳　主编

哈尔滨工程大学出版社
Harbin Engineering University Press

J型本科人才培养系列教材

计算机系统结构

张佳琳 主 编

哈尔滨工程大学出版社
Harbin Engineering University Press

内容简介

本书为研究者从事的各种实际工作提供必要的软件基础知识，以便能得心应手地用好和管好计算机，更好地完成各种计算机应用任务，为进一步学好数据库系统、计算机网络和分布式系统等课程奠定理论基础。

本书可作为计算机相关专业的学生及教师的学习和参考用书，也可作为计算机专业的专业基础课和必修课的参考教材。

图书在版编目(CIP)数据

计算机系统结构/张佳琳主编. —哈尔滨：哈尔滨工程大学出版社，2018.7

ISBN 978-7-5661-2008-3

Ⅰ.①计… Ⅱ.①张… Ⅲ.①计算机体系结构—高等学校—教材 Ⅳ.①TP303

中国版本图书馆 CIP 数据核字(2018)第 150740 号

选题策划 夏飞洋
责任编辑 张忠远
封面设计 刘长友

出版发行 哈尔滨工程大学出版社
社　　址 哈尔滨市南岗区南通大街 145 号
邮政编码 150001
发行电话 0451-82519328
传　　真 0451-82519699
经　　销 新华书店
印　　刷 哈尔滨市石桥印务有限公司
开　　本 787 mm×1 092 mm　1/16
印　　张 20.25
字　　数 534 千字
版　　次 2018 年 7 月第 1 版
印　　次 2018 年 7 月第 1 次印刷
定　　价 58.00 元

http://www.hrbeupress.com
E-mail:heupress@hrbeu.edu.cn

前　言

操作系统犹如人之灵魂，支配着各种资源，了解和掌握操作系统理论是开启信息世界的一把钥匙。操作系统也是计算机相关专业的重要专业基础课和必修课，理论性和实践性都比较强。操作系统是计算机系统中不可缺少的基础系统软件，用于管理和控制计算机系统中软、硬件资源，是计算机系统直接与用户打交道的界面，是计算机系统的灵魂和核心。

通过本课程的学习使学生掌握操作系统设计理论和方法的基本知识，具有分析和维护系统等方面的初步能力。了解操作系统对整个计算机系统的管理和控制功能，以及用户与操作系统的接口，对常用计算机操作系统（DOS、Windows 和 Linux）能进行基本的操作使用。为今后从事的各种实际工作提供必要的软件基础知识，以便能得心应手地用好和管好计算机，更好地完成各种计算机应用任务，并为进一步学好数据库系统、计算机网络和分布式系统等课程奠定理论基础。

编　者

2018 年 6 月

目　录

计算机系统结构自学考试大纲

第1章 概 论

1.1 计算机系统的层次结构

计算机系统是由不同的层级组成的。计算机系统层次结构实际上分为软件层次和硬件层次两种，我们从计算机语言的角度来细致分层可分为六层，分别是微程序机器层次、传统语言机器层次、操作系统机器层次、汇编语言机器层次、高级语言机器层次和应用语言机器层次，整个计算机系统层次结构如图1-1所示。

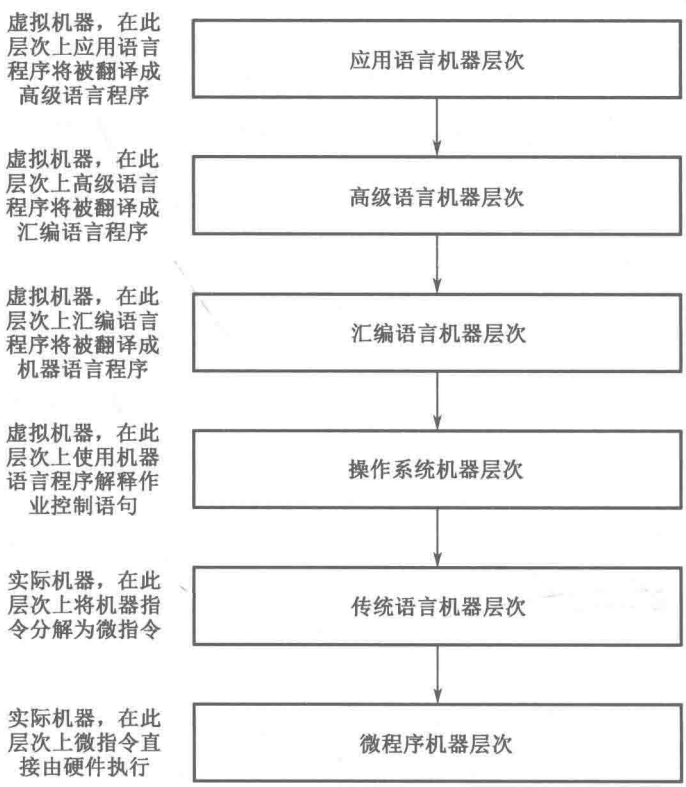

图1-1 计算机系统层次结构

软件层次是指用软件来实现的机器，称之为虚拟机器，在六层结构中虚拟机器包括操作系统机器层次、汇编语言机器层次、高级语言机器层次和应用语言机器层次。当然，虚拟机器并不一定完全是由软件来实现的，有些操作也有可能使用固件和硬件来实现，如操作系统的某些指令可以用微程序固件或者硬件来直接实现以加快系统的运行速度。

硬件层次是指用硬件或者固件来实现的机器，称之为实际机器，在六层结构中实际机器包括微程序机器层次、传统机器层次。在这里我们给出了固件的定义："使用只读、读写存储器来实现软件功能的硬件。"目前只读、读写存储器一般基于超大规模集成电路，只是在蚀刻精度方面可以比 CPU、显卡、内存的精度稍低，InteL VPU、AMD CPU 和 AMD 显卡的蚀刻精度已全部达到 14 nm，英伟达显卡的蚀刻精度是 16 nm，内存的蚀刻精度不同；而固件的蚀刻精度则是根据实际应用的需求来采用，如果 65 nm 和 130 nm 能够满足实际应用需要，就不会采用价格更高的高精度芯片。

各个层次之间通过翻译或者解释来进行命令的转换。翻译是使用转换程序将上一层次的程序转换为下一层次上的等效程序，代表性语言是 C、C++、Fortran 等；解释是使用下一层次的指令来逐句模仿最终达成上一层次程序所能实现的功能，代表性语言是 Basic，Perl 等。另外，还有先进行编译再使用解释完成执行的，代表性语言是 Java。

1.1.1 微程序机器层次

从计算机语言角度进行的计算机系统分层的第一级是微程序机器层次，它属于计算机系统结构和计算机组成原理这两个学科，在计算机组成原理的书籍中一般将这一层次称为微指令系统。

微指令是由多个微命令（即控制信号）组成的，这些信号是由不同的门电路进行逻辑运算得到的。在同一个 CPU 周期中，由不同的微指令组成了一组能够实现某种操作的指令组合，这个指令组合就被称为微程序。按照次序执行微指令就实现了微程序，每一个微程序对应一条机器指令（第二层次），机器指令和微指令是通过解释来进行操作转换的，微指令的编译方法决定了微指令的格式，因此微指令可以分为水平型微指令和垂直型微指令。

水平型微指令是指在一次运行中可以多个并行操作的微命令。水平微指令的格式如图 1－2所示，包括控制字段、判别测试字段和下地址字段。控制字段用来表明相应操作的代码，判别测试字段是条件判断逻辑所占字段，下地址字段是控制存储单元格式的二进制表示。控制字段又可以分为直接控制和编码控制，直接控制是统计操作数量，操作数量就是控制字段的位数；编码控制是将操作分类，统计每一类操作的个数并计算其二进制位数，控制字段的位数就是所有类别的二进制位数的相加求和，我们以一道例题来说明直接控制和编码控制的区别。

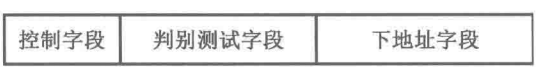

图 1－2　水平型微指令格式

【例 1－1】 某种计算机采用水平型微指令，已知机器一共有 35 条微命令，可以分为三个相斥类别，每类微命令数量为 20、6 和 9，判别测试字段为 6 位，存储控制单元为 4 096 个，请计算直接控制和编码控制水平微指令的各字段位数。

解 采用直接控制的水平微指令:35 条微命令需要 35 位,因此控制字段的位数为 35 位,已知判别测试字段位数为 6 位,控制存储单元个数为 4 096 个,可以计算出下地址字段位数为 12 位($4\ 096 = 2^{12}$),因此我们得到微指令位数总共为 35 +6 +12 =53 位。

采用编码控制水平微指令:35 条微命令分为三类,每类数量为 20、6 和 9,20 条需要 5 位($2^4 < 20 < 2^5$),6 条需要 3 位($2^2 < 6 < 2^3$),9 条需要 4 位($2^3 < 9 < 2^4$),此时控制字段的位数为 5 +3 +4 =12 位,判别测试字段位数和下地址字段位数同直接控制的水平微指令,因此我们得到的微指令位数总共为 12 +6 +12 =30 位。

垂直型微指令中设置微操作码字段,采用微操作码编译法,由微操作码规定微指令的功能。垂直型微指令的结构与机器指令结构比较类似,如图 1 -3 所示,垂直型微指令可以包括一到两个微操作码,因此每条微指令能够实现的功能很简单,由垂直型微指令组成的微程序的长度要大于水平型微指令组成的微程序。

微操作码字段	微操作码字段	操作对象地址字段

图 1 -3 垂直型微指令格式

根据上文的水平型微指令和垂直型微指令的介绍,下面从三个方面对两者的优缺点进行比较。

(1)并行操作和灵活程度方面,一条水平型微指令可以有很多条微命令,执行一条水平型微指令相当于同时执行多条微命令,而垂直型微指令一般只有 1 ~2 个微操作码,也就意味着执行一条垂直型微指令最多只能执行两条微命令,并且由于水平型微指令包含的微命令数量多,微命令组合的方式就可以很灵活。

(2)微程序长度和执行速度方面,由于水平型微指令包含很多条微命令,可以用相对较少的微指令组成微程序,冗余较少;而使用垂直型微指令则需要远大于水平型微指令的条数才能达到相同的效果,冗余较多。因此由水平型微指令组成的微程序长度较短,相同的机器上,长度短的微程序执行速度更快,水平型微指令也占据优势。

(3)掌握难度方面,水平型微指令的结构比较复杂,掌握难度较高,而垂直型微指令的结构比较简单,掌握难度较低,因此在学习难度上,垂直型微指令更有优势。

1.1.2 传统语言机器层次

计算机系统分层的第二级是传统语言机器层次,这一层次主要涉及指令集的相关内容,指令集存储在 CPU 中,是设定 CPU 操作的固定程序,CPU 指令集是计算机运行的最核心的部分。

目前,指令集包括复杂指令集(Complex Instruction Set Computing, CISC)、精简指令集(Reduced Instruction Set Computing, RISC)、显式并行指令集(Explicitly Parallel Instruction Computing, EPIC)和超长指令集(Very Long Instruction Word, VLIW)。

CISC 指令集的各个指令是串行执行的,每条指令中的各个操作也是串行执行的。顺序执行的优点是控制简单,但计算机各部分的利用率不高,执行速度慢。目前,主要采用 CISC 指令集的 CPU 包括 Intel 的 X86(IA -32)、EM64T、MMX、SSE、SSE2、SSE3、SSSE3(Super SSE3)、SSE4A、SSE4.1、SSE4.2、AVX、AVX2、AVX -512、VMX 指令集和 AMD 的 X86、X86 -64、3D -Now! 指令集。CISC 指令集存在着两个较明显的缺点:其一,在 CISC 计算机中,典

型程序的运算过程所使用的80%指令只占一个处理器指令系统的20%,这造成了大量的浪费;其二,复杂的指令系统必然带来结构的复杂性,既增加了设计的时间与成本,还容易造成设计失误。

RISC 指令集对指令数目和寻址方式都做了精简,它的指令集中所有指令的格式都是一致的,所有指令的指令周期也是相同的,采用流水线技术,使指令并行执行程度更好,编译器的效率更高。目前,主要采用 RISC 指令集的 CPU 包括了 Compaq 的 Alpha、HP 的 PA - RISC、IBM 的 PowerPC、MIPS 的 MIPS 和 SUN 的 Sparc。

EPIC 指令集的代表是 Intel 的 IA - 64,它基于超长指令集架构(Very Long Instruction Word,VLIW),克服了 VLIW 指令集独立性太高的缺陷,但是其发展势头并不好,目前仅有 HP 依然坚守在 Itanium 阵营。

由于 EPIC 和 VLIW 的缺陷和发展趋势等原因,市场上的主流指令集还是 RISC 和 CISC。RISC 指令集相对于 CISC 指令集具有如下优点:

(1)RISC 指令集使用统一的指令周期,避免了 CISC 指令周期不一致造成的系统运行不稳定性;

(2)RISC 采用了流水线技术,并发性能较 CISC 有了很大的提高;

(3)RISC 的指令格式标准化,大幅提高了流水线技术的效率;

(4)RISC 基于超大规模集成电路采用了面向寄存器堆的指令,效率大幅超过 CISC;

(5)RISC 使用装入/存储指令完成对内存的访问,避免了 CISC 大量访问内存造成的时间损耗。

基于以上优点的分析,绝大多数 Linux 服务器工作站都是采用 RISC 指令集的 CPU。

1.1.3 操作系统机器层次

计算机系统分层的第三级是操作系统机器层次,它的主要工作是管理计算机的软件资源和硬件资源。这一层次主要涉及宽泛指令,这里的宽泛指令是指操作系统定义和解释的软件指令,这些指令包括进程控制语言、PV 原理作业控制语言等。由于操作系统是一门单独的课程,因此,本节仅对进程控制语言进行简单的介绍。

进程控制语言(PV 原语)是通过操作信号量来完成进程之间的同步和互斥的。信号量的概念在 1965 年由荷兰科学家艾兹格·迪杰斯特拉提出,信号量被用来完成进程之间的同步和互斥的时候,进程不能够赋值,它完全由操作系统来管理,进程只能通过初始化和阻塞环型原语进行访问。

PV 原语使用 sem 变量来记录某类当前可用资源的数量,这个变量有两种表示方法。

(1)sem 变量的取值必须是大于或者等于 0 的整数,当变量值大于 0 的时候就说明系统存在可用的资源,而等于 0 的时候则说明系统没有可用的资源。

(2)sem 变量的取值可以为任意正负整数,当变量值大于或者等于 0 的时候与第一种表示方式含义相同,当变量值小于 0 的时候则表示有多少个进程在等待使用资源。

P 原语和 V 原语是设置进程状态的操作原语,PV 原语必须成对使用并且在执行期间绝对不允许有任何中断发生,临界区是访问共享资源的等待区域,在此区域外的进程没有访问共享资源的权限。

P 原语是阻塞原语,其工作内容是把进程的状态设置为阻塞,直到有某类资源才唤醒这个进程。P 原语的操作过程如下:

(1)进程申请一个空闲资源,将信号量 sem 的值减 1;

(2)判断是否成功,如果成功,则是成功申请到资源,如果不成功,则说明没有空闲资源,将进程的状态设置为阻塞并放入需求此类资源的阻塞队列,转到进程调度。

V 原语是唤醒原语,其工作内容是把一个被阻塞的进程的状态设置为唤醒。V 原语的操作过程如下:

(1)其他进程释放了一个空闲资源,此时将信号量 sem 的值加 1;

(2)在需求此类资源的阻塞队列中唤醒一个进程,然后再返回原来的进程继续执行操作或者转到进程调度。

PV 原语的三种应用类型如下。

(1)实现对一个共享变量的互斥访问。下面的原语操作流程中 mutex 的初始值设为 1。

P(mutex);

V(mutex);

非临界区;

(2)实现对一类资源的互斥访问,在下面原语操作流程中 resource 的初始值为该资源的个数 N。

P(resource);

V(resource);

非临界区;

(3)作为进程同步工具。

临界区 C1;

P(sem);

V(sem);

临界区 C2;

下面我们用例子来说明 PV 原语的应用类型。

【例 1-2】 某超市门口为顾客准备了 20 辆手推车,每位顾客进去买东西时取一辆推车,在买完东西结完账以后再把推车还回去。试用 P、V 操作正确实现顾客进程的同步互斥关系。

分析 在这道例题中我们可以把手推车视为某类系统资源,每个顾客则是需要互斥访问该类资源的进程。因此例 1-2 实际上是 PV 原语的第二种应用类型,当这类资源的数量简化为 1 的时候,那么就会变成 PV 原语的第一种应用类型。

解 semaphore S_CartNum;//设置信号量表示空闲手推车数量,初始值为 20

```
void consumer(void)//设置进程:申请空闲手推车,操作,释放手推车
{
  P(S_CartNum);
  买东西;
  结账;
  V(S_CartNum);
}
```

【例1-3】 桌子上有一个果盘，每一次可以往里面放入一瓣橘子。妈妈负责向盘子中放橘子，女儿负责吃盘子中的橘子。把妈妈、女儿看作两个进程，试用P、V操作使这两个进程能正确地并发执行。

分析 妈妈和女儿的操作是相互制约的，妈妈进程在执行完放入橘子后，女儿进程才能执行吃橘子的操作，说明该问题实际为第三种应用类型。

解
```
semaphore S_CartNum;//设置信号量表示空闲手推车数量,初始值为20
void consumer(void)//设置进程:申请空闲手推车,操作,释放手推车
{
  P(S_CartNum);
  买东西;
  结账;
  V(S_CartNum);
}
```

1.1.4 汇编语言机器层次

计算机系统分层的第四级是汇编语言机器层次，这一层次主要涉及汇编语言的相关内容，汇编语言分为数据传送、算术运算、逻辑运算、串操作、控制转移和处理器控制这六类，机器指令的格式如图1-4所示，包括地址码和机器码，一般长度都为8位。

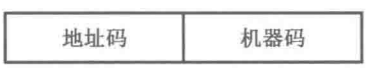

图1-4 机器指令格式

数据传送指令包括基本传送指令(MOV)、入栈指令(PUSH)、出栈指令(POP)、交换指令(XCHG)、查表指令(XLAT)、输入指令(IN)、输出指令(OUT)、取有效地址指令(LEA)、地址指针装入DS指令(LDS)、地址指针装入ES指令(LES)、标志装入AH指令(LAHF)、设置标志指令(SAHF)、标志压入堆栈指令(PUSHF)和标志弹出堆栈指令(POPF)。

算术运算指令包括不带进位的加减法指令(ADD,SUB)、带进位的加减法指令(ADC,SBB)、加法和减法的ASCII码调整指令(AAA,DAA,AAS,DAS)、自加自减指令(INC,DEC)、求补及比较指令(NEG,CMP)、无符号数乘法指令(MUL)、符号数乘法指令(IMUL)、无符号数除法指令(DIV)、符号数除法指令(IDIV)、转换指令(CWB,CWD)。

逻辑运算指令包括非指令(NOT)、与指令(AND)、或指令(OR)、异或指令(XOR)、测试指令(TEST)、移位指令(SHL/SHR,SAL/SAR,ROL/ROR,RCL/RCR)。

串操作指令包括重复前缀指令(REP)、串传送指令(MOVS)、串存储指令(STOS)、串读取指令(LODS)、串比较指令(CMPS)。

控制转移指令包括无条件转移指令(AJMP,SJMP,LJMP,JMP)、条件转移指令(JCXZ),循环指令(LOOP)、中断指令(INT,INTO,IRET)、过程指令(RET,CALL)。

处理器控制指令包括CF设置指令(CLC,STC,CMC)、IF设置指令(CLI,STI)、DF设置指令(CLD,STD)、暂停指令(HLT)、空操作指令(NOP)。

下面对一些典型的机器指令进行介绍。

INC指令和DEC指令:INC eax指令是将寄存器eax中的数加1,DEC指令是将寄存器

eax 中的数减 1，表 1 -1 是 INC 指令的地址码、机器码和汇编指令，地址码和机器码都是 16 进制编码。

表 1 -1 INC 指令

地址码	机器码	汇编指令
00000000	40	inc eax
00000001	41	inc ecx
00000002	42	inc edx
00000003	43	inc ebx
00000004	44	inc esp
00000005	45	inc ebp
00000006	46	inc esi
00000007	47	inc edi

我们将表 1 -1 中的机器码都转换为二进制数字，分别是 01000000、01000001、01000010、01000011、01000100、01000101、01000110、01000111，我们可以发现机器码的开头都是 01000，仅仅是后三位的不同，实际上后三位就是寄存器在机器中表示的二进制机器码，总结出寄存器的二进制编码，如表 1 -2 所示。

表 1 -2 寄存器的二进制编码

机器码	寄存器	机器码	寄存器
000	eax	100	esp
001	ecx	101	ebp
010	edx	110	esi
011	ebx	111	edi

DEC 指令：DEC 指令包括地址码、机器码和汇编指令。DEC 指令对应的机器码开头是 01001，DEC 指令机器码部分组成就是 DEC 指令的机器码头部与寄存器机器码，如表1 -3所示，例如 dec eax 的机器码就是机器码头部 01001 与 eax 寄存器的机器码 000 拼接生成的 01001000，转换为 16 进制就是 48，如表 1 -4 所示。

表 1 -3 拼接生成 DEC 机器码

机器码头部	寄存器机器码	拼接指令机器码	16 进制表示
01001	000	01001000	48
01001	001	01001001	49

表 1-3(续)

机器码头部	寄存器机器码	拼接指令机器码	16 进制表示
01001	010	01001010	4A
01001	011	01001011	4B
01001	100	01001100	4C
01001	101	01001101	4D

表 1-4 DEC 指令

地址码	机器码	汇编指令
00000008	48	dec eax
00000009	49	dec ecx
0000000A	4A	dec edx
0000000B	4B	dec ebx
0000000C	4C	dec esp
0000000D	4D	dec ebp
0000000E	4E	dec esi
0000000F	4F	dec edi

MOV 指令:MOV 指令是基本传送指令,其基本格式是(MOV eax,数字或寄存器),假设我们有一条 MOV eax,16H 的指令,这条指令的含义就是将 8 位数据 16H 传递给 eax 寄存器,表 1-5 是 MOV eax,reg 指令的地址码、机器码、二进制机器码和汇编指令。

表 1-5 MOV 指令

地址码	机器码	机器码后两位转换为二进制	汇编指令
00000010	8BC0	11000000	mov eax,eax
00000012	8BC1	11000001	mov eax,ecx
00000014	8BC2	11000010	mov eax,edx
00000016	8BC3	11000011	mov eax,ebx
00000018	8BC4	11000100	mov eax,esp
0000001A	8BC5	11000101	mov eax,ebp

以表 1-5 中第三行为例,二进制机器码最后 6 位是 000010,前三位 000 对应的是目标寄存器 eax 的机器码,后三位 010 对应的是源寄存器 edx 对应的机器码。

ADD 指令和 SUB 指令:ADD 指令是加法指令,其基本格式是 ADD OPRD1,OPRD2,此命令的含义是 OPRD1 中的数值加上 OPRD2 的数值并存储到 OPRD1 中,OPRD1 可以是表 1-4中的寄存器,也可以是存储器操作数;OPRD2 可以是一个数字,也可以是寄存器或者存

储器操作数,但不允许两者同时为存储器操作数,ADD 指令见表 1-6。SUB 指令是减法指令,其基本格式是 SUB OPRD1,OPRD2,这个命令的含义是 OPRD1 中的数值减去 OPRD2 的数值并存储到 OPRD1 中,SUB 指令见表 1-7。

表 1-6 MOV 指令

地址码	机器码	机器码后两位二进制	汇编指令
0000006A	03C0	11000000	add eax,eax
0000006C	03C1	11000001	add eax,ecx
0000006E	03C2	11000010	add eax,edx

表 1-7 SUB 指令

地址码	机器码	机器码后两位二进制	汇编指令
00000072	2BC8	11000000	sub eax,eax
00000074	2BC9	11000001	sub eax,ecx
00000076	2BCA	11000010	sub eax,edx

1.1.5 高级语言与应用语言机器层次

计算机系统分层的第五级是高级语言机器层次,计算机系统分层的第六级是应用语言机器层次,在这两个层级上主要涉及高级程序语言的相关内容。第五级的功能是翻译应用程序使其能够在计算机上运行,第六级的功能是为满足各种用户需求进行应用开发。例如,常用的通信软件、办公软件、游戏及其他各种应用程序都在这两个层次上进行开发和运行。

相对于低级语言(包括机器语言指令集和汇编语言),高级程序语言的语法更接近于人们的日常生活用语,能够让致力于进行程序开发的人员更容易地学习,使程序开发工作不会局限于少部分机器语言开发人员。

高级语言分类方式有很多种,但是根据编写方式的不同一般分为两类,分别是面向过程语言和面向对象语言,下面简单介绍一下这两类语言。

面向过程语言也被称为结构化程序设计语言,在整个程序设计过程中,将大的任务分解成诸多的小任务,使用函数来完成这些小任务,面向过程语言实现了与计算机硬件无关、语言更接近自然语言、模块化设计和严格的书写规定等内容,完成从低级语言到高级语言的跃迁。常见的面向过程语言包括 C 语言和 Fortran 语言,C 语言是一种通用型程序设计语言,能够以非常简单的方式进行编译,并且不需要任何运行环境的支持,于 1972 年在美国贝尔实验室开始设计并于 1973 年年初完成主体内容。Fortran 语言是一种应用于科学计算和工程计算领域的程序设计语言,1951 年,在 IBM 公司开始设计并于 1954 年完成了第一个版本。

面向对象语言在程序设计的过程中以类作为基本的结构单元、以描述对象为核心。面向对象语言有四个基本的特征,分别是抽象、继承、封装和多态性。常见的面向对象语言有

两个类别:第一个类别是纯面向对象语言,包括 Smalltalk、Eiffel 等;第二个类别是混合面向对象语言,就是在过程语言中添加面向对象的特性,包括 C++、Java、Objective-C、PHP、Python、Javascript 等。

基于不同的场合,需要根据实际情况来选择高级语言,也需要比较不同类别高级语言的优缺点。面向过程语言的优点是性能高,缺点是不易维护、复用和扩展;面向对象语言的优点是易维护、复用和扩展,缺点是性能较面向过程语言低。在网站开发的过程中,需要大量的维护、复用和扩展操作,面向对象语言在维护复用扩展方面有着明显的优势,并且在性能方面也能够满足要求,这个时候我们就会选择面向对象语言进行开发。目前,网站开发语言主要是以.net、Java 和 PHP 等面向对象的语言为主,与此相反的是嵌入式开发,对性能要求极为严格,并且复用扩展操作很少,这种情况下我们就会选择面向过程语言进行开发。目前嵌入式开发使用的语言主要以 C 语言为主,C++基于 C 语言发展而来,在性能方面表现十分出色,所以在嵌入式开发中也会使用 C++。

1.2 计算机系统结构、计算机组成和计算机实现

1.2.1 计算机系统结构的定义及发展历史

1. 计算机系统结构的定义

定义 1-1 计算机系统多级层次结构是机器语言机器级的结构,它是软件和硬件/固件的主要交界面,是机器语言目标程序能在机器上正确运行所应具有的界面结构和功能。

计算机系统结构也称为计算机体系结构,一般是开发人员能够识别的概念性结构和功能特性(计算机的属性、计算机系统的逻辑和计算机系统的功能),这些内容包括了硬件和软件之间的关系。目前,计算机系统结构主要是 1946 年正式提出的冯·诺依曼结构,它的特点是以存储器和运算器为中心来集中控制,如图 1-5 所示。计算机的功能特性包括数据类型和格式,寻址方式,寄存器的组织方式规则,指令集的类型、格式排序方式,中断类型级别,存储系统的类型、格式、容量,计算机的工作状态,输入、输出系统,以及信息安全措施等方面的内容。

我们知道 Windows 操作系统自带的计算器工具,它给我们展示了计算机系统的一个功能计算功能,而从开发人员的视角看到的是输入数据(使用键盘或者鼠标输入需要计算的数据和逻辑运算符),将数据传输给存储设备(输入的数据将直接存放到内存中),由存储设备传输给计算设备(数据从内存经过缓存进入 CPU 进行逻辑运算),随后再把计算的结果传输到存储设备和输出设备(传值给内存和显示器,显示器显示计算结果)。在这个例子中,软件开发人员看到的计算器工作流程所展示的概念性结构、计算机工作状态、CPU 使用的指令集以及内存地址的编码方式等功能特性,无论是概念性结构还是功能特性都属于计算机系统结构的范畴。

对于硬件开发人员而言,他们所能够识别的系统结构是计算机的组成和实现。例如,IBM 370 系列机所有机型都具有相同的系统结构(包括相同的指令系统、字符编码等),但是采用了不同的组成和实现技术,导致了性能和价格的不同,在价格较低的机器上采用的指令运行方式是顺序执行策略,采用的数据通道位宽是较小的位宽(8 b)。而在价格较高的

机器上采用的指令运行方式包括重叠、流水线和其他并行处理方式，采用的数据通道位宽是较高的位宽(64 b)。

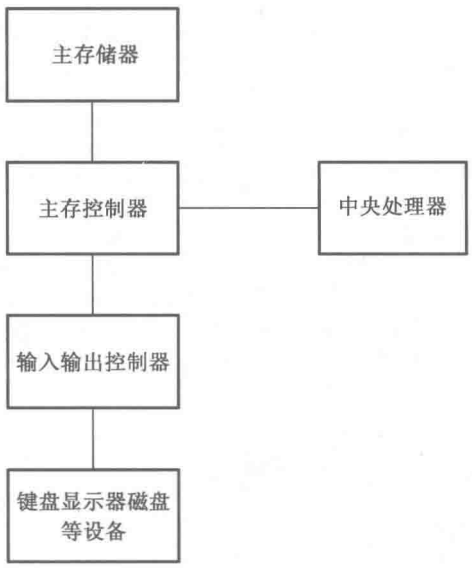

图1-5 计算机系统结构概念性结构

除此之外，有关于计算机系统结构的其他定义中比较有代表性的定义是认为计算机系统结构主要研究软硬件功能分配及对软硬件界面的确定。众所周知，计算机系统是由软件、硬件和固件组成的，软件包括操作系统软件(Windows 系列操作系统、Linux 操作系统、Mac OS操作系统等)、应用程序软件(Office 系列办公软件、Tencent 系列社交软件等)，硬件包括中央处理器(Central Processing Unit，CPU，包括 Intel 系列 CPU 和 AMD 系列 CPU)、主板(支持 Intel 系列 CPU 的主板和支持 AMD 系列 CPU 的主板)、内存(DDR3 800 MHz—DDR4 3 000 MHz)、显卡(英伟达系列显卡和 ATI 系列显卡)、硬盘(固态硬盘和机械硬盘)、显示器(扭曲向列液晶显示器、超扭曲向列液晶显示器和宽视角模式液晶显示器)、输入输出设备(键盘、鼠标、绘图板等)，如图1-6所示。固件包括主板基本输入输出系统(Basic Input Output System，BIOS)、显卡 BIOS、交换设备芯片等。同一种功能可以使用硬件来实现，也可以使用软件和固件来实现，只不过在性能和价格之间进行了取舍。Windows 操作系统里的计算器工具，如果单独使用硬件实现计算器功能，需要在某一芯片区域蚀刻能够进行科学计算的门电路，而使用软件实现仅需要进行相应的程序设计，并调用 CPU 的相应逻辑运算单元即可，两者比较而言，使用软件来实现是非常合适的。但是，若使用我国最新的分组数据加密算法 SM4 进行加密时，SM4 基本的密钥长度为 128 位，白化后的密钥长度为 384 位；若使用软件调用硬件资源的方式进行加密，那么耗时将不可接受；若仅采用硬件加密，那么加密一次就需要重新制作加密芯片，费用将无法接受。单纯的软件和硬件实现都是不可行的。因此，可以采用现场可编程门阵列(Field-Programmable Gate Array，FPGA)作为分组数据加密的实现手段，即是一种固件。上述两个例子说明了如何选择软件、硬件和固件实现相应的功能，在现实中组装一台满足实际工作和生活需求的个人电脑，就可以看作是简单的计算机系统设计和制造的过程，只是在这个过程中要选择适合自身的概念性结构、功能及计算机硬件。