四川省自动化与仪器仪表学会
科学仪器专业委员会资料

智能仪器设计

—微处理器在仪器仪表中的应用

下 册

成都科学仪器厂
科学仪器专业委员会
成都电讯工程学院

目　录

第一章　智能仪器设计的基本问题

第二章　输入输出

第三章　D/A、A/D转换器与微处理器的接口及其编程

第四章　测量算法

第五章 通用接口设计

第六章 软件设计

第七章 开发和调试设备的利用

附　录

编　者　话

下册《智能仪器设计》是为对μP已有一定基础的读者写的，在本书上册内容的基础上，下册进一步阐述μP在仪器实际应用中的一系列关键问题。目的在于为读者提供解决实际设计问题所必须的知识，使他们领会如何将μP用于他们的设计，来代替一大堆逻辑电路；如果利用μP来改善模拟式硬件的性能，或降低对硬件的技术要求；如何利用算法来扩展设备的功能；如何实现设备的自动化和智能化。

本书由成都电讯工程学院电子仪器与测量教究室的张世箕、郭成生、杨安禄、王树青，陈光禹分章执笔。本书的这种写法，可以说是一种新的尝试，由于作者们的水平和经验所限，不当之处尚祈读者惠予指正，于此预致谢意。

本书承四川省自动化与仪器仪表学会科学仪器专业委员会及成都科学仪器厂负责出版事宜，才能促成本书与广大读者见面。学会的宋水庆、孟宪武等同志劬劳尤多。于此一并致谢。

作者

1983年8月于成都

第一章　智能仪器设计的基本问题

1·1 智能仪器

什么是智能仪器，至今未有确切定义。从字面上说，所谓智能，意味着仪器中含有一定的人工智能，即是利用电路代替人的一部分脑力劳动。然而，事实上，迄今智能仪器中所具备的人工智能并不很多。特别在视觉（图形及色彩判读）、听觉（语音辨别及语言的领悟）、思维（推理、判断、学习、联想）等方面的能力仍十分有限，甚至阙如。美国习用的一个通俗名词，叫做smart instruments，包含着时髦漂亮、聪明伶俐、精明能干、机敏灵巧等意思。实质上是用以形容新的一代电子测量仪器，它们与传统仪器的区别主要在于：

（1）具有较完善的可程控能力，并常配有GPIB接口（IEC—625，IEEE—488等准接口），多以ASCII码（ISO—646，SI—939—75）输入输出；

（2）面板控制（本地控制）采用灵活的功能和数字按键，或兼有这类按键的编程能力；

（3）面板显示十进数字，或兼能显示若干拉丁字符、甚至全部ASCII字符；近来用CRT显示字符及图形的情况日渐多见，个别的附有小型打印机；

（4）具有一定的数据处理能力（计算、变换、误差修正等），从而提高仪器的精确度，并扩展其测量功能（通过间接测量方法）；

（5）具有一定的可编程自动化能力，包括指令和数据存贮、自动调零、自检、自校、等等；个别仪器可用小型盒式磁带输入程序；

（6）以上许多特色主要依靠在仪器内采用微处理器系统及相应的大规模集成电路而获得。

因此，所谓智能仪器，也可以理解为μP化的仪器，即是以微处理器为基础而设计制造出来的一代新型仪器。本书就是在这个意义上来论述这一代新仪器设计的一些基本问题。其中许多问题的解决办法，同样也适用于其他许多μP化的电子设备。因此，可以把仪器一词理解为广义的仪器，即是一个独立的电子设备（不是配件，不是成套系统，不过可作为系统中的一员），例如通信系统中的一台接收机、雷达系统中的一个激励器、医疗护理系统中的一台心电监测仪、商店中的一台电子称或计价收款机、工业过程中的一台控制器、等等。

本书假定读者对于仪器的模拟硬件、测量和控制原理、以及μP系统本身，均已具备必要的知识。总之，本书的论述，集中于如何把μP系统应用到电子仪器的设计中，以改善仪器的性能，使之跻于新一代仪器的行列，即基本具备上列的一些特色。

1·2 智能仪器设计的计划

在进行具体设计之前，首先要拟订设计的计划。计划阶段包括图1·1中所示的一些工

作内容。在计划阶段中，仪器的软件和硬件设计人员在一起共同商讨，然后作出决定。

大多数 μP 化产品的研制，其决策常受下列因素所左右：该产品用来作什么工作？怎样工作？为何要这样作？生产批量有多大？对于一个电子玩具，或者一台 μP 化的新型家用缝纫机，其设计的构思和计划当然大不相同。对于一个电子玩具而言，这将是一个彻头彻尾全新的产品，在硬件和软件方面都大有游刃余地。对于家用缝纫机，其主要机械硬件都是现成的、典型化了的，这给设计限定了许多约束，主宰着软件的开发。

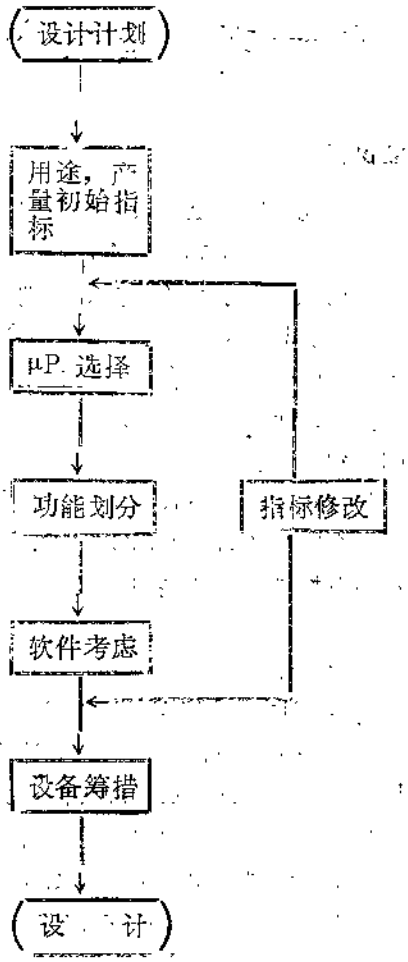


图1·1智能仪器设计的计划阶段

（1）初始考虑和指标

产品生产的批量常是一个主要考虑的因素。随着集成电路工业的发展，电路硬件价格日益低廉，而软件劳务则总是昂贵的。国内目前许多人对软件成本尚未建立起正确的概念。应该强调指出，μP 化仪器的重要特点之一，是把对模拟硬件的过高要求转化为依靠 μP 软件来解决。因此，μP 化仪器设计的劳务主要是在软件方面。事实亦表明，软件成本的比重日益增长，现已凌驾于硬件成本之上，甚至占总成本的3/4 以上。不过应注意，当产品批量很少时，主要依靠硬件解决问题还是适宜的。当产品批量较大时，则主要依靠软件来解决问题较为明智。因为大批产品的软件设计是一次过完成的，其成本由大批产品来分摊；批量愈大，则每件产品摊到的软件成本就愈低廉。而每个产品的硬件成本则是固定的，几乎不因批量大小而变；充其量只不过是大批硬件元件进货的批发价略为便宜一点而已。

此外，当然还有产品的性能/价格比的问题。这取决于产品的重要性及可靠性方面的要求。例如，电子玩具在这方面可以要求相当低，属于大路货的低档仪器要求可较低，高级精密多功能仪器要求就较高，而航天等尖端技术方面的应用则要求极高。敏感的医疗护理仪器、工业上关键的测试、监控仪器，保安和报警设备等，其失效可能导致生命攸关，或会造成巨大危险或损失，这就要求集中更多的智慧和劳力，以保证创造高水平的性能。

产品的技术指标，是根据产品的重要性、用途、和使用方法来决定的。技术指标当然主宰着设计。不过，在设计之初，技术指标往往是相对地粗略、笼统的，仅包括最少

量的必要数据。随着设计的深入进行，指标将不断修改完善。

（2）μP的选择

关于μP的选择，应由软、硬件主管人员共同研究决定。应分析研究下述各种因素。首先当然要考虑有关的技术要求，如所需并行处理的比特宽度（字节长度），所需的支持电路，要求的执行速度，时钟速率，所需电源的组数，系统的兼容性，中断能力，抗噪声和抗干扰能力，功耗，对环境方面的要求，等等。此外，还应考虑其价格和货源，本厂的采购政策，软件支持条件，本厂工程技术人员对该型号μP的熟悉程度，以及对该型μP所作的典型试验所得结果，等等。

在选用μP时，不应过份依赖μP厂商所给出的技术资料。"卖瓜的不说瓜苦"。μP厂商给出的高性能一般是从对该型μP最"有利"的算法得出的，对于我们设计中的若干关键算法来说，该型μP执行起来就未必会达到那么良好的结果。此外，也应注意一些似是而非的情报以及未有给出的信息，必要时应向有关厂商仔细查询。

关于比特宽度，在目前有4、8、16比特的μP可供选用。

4比特μP属于低性能的廉价低档产品。现多为不可扩展的单片控制单元，多用于产量极大的日用电器。在工业中则多单独或与强电流驱动器联合使用，来控制灯光、阀门、电磁铁等。在智能仪器中，应用颇为有限。

8比特μP大概是目前最常用的，其系统易于扩展，并可扩到多μP系统，甚适于产品今后的改型和发展。其配套用的支持硬件颇为齐全。软件积累丰富，而且兼容性甚好，往往可能只需要作不甚繁重的软件重新设计的工作。

16比特μP目前还较昂贵，支持硬件尚未充分发展出来，软件积累亦有限。迄今在智能仪器中的应用仍不甚广泛，主要用于需作实时数据处理的高档产品中。

在μP的软件开发方面，应考虑仿真和调试所需的硬件设备和软件手段。手头现有的开发系统是否适于μP的选型之用，或是否易于与之适配（添加必要的接口和软件）。如有现成合适的开发系统，当然会大大节省智能仪器的设计、初样和调试时间。

软、硬件设计人员对该型μP的熟悉程度，是选型的最重要考虑之一。若对μP及其附属支持系统的体系结构及指令集的运用十分熟悉，当然会使设计工作更快速、更准确。

最后，关于所谓典型性能试验（Benchmarking），通常是指在二种或多种μP上作试验性运行，以资比较其工作性能。在典型性能试验中，令受试的每一型μP执行若干专门编写的或经过审慎挑选的子程序。原则上，这些子程序对各受试μP的体系结构应是一视同仁、公正无偏的，并且应复盖设计对象中一些困难或颇成问题的操作内容。典型性能试验的结果应能揭示受试μP的数据处理能力。不过，讲句老实话，关于典型性能试验，一般是讲得多，做得少。究其原因，主要有下列一些。首先，μP作为智能仪器中的控制器来使用时，关键的问题大都是毫秒级或更慢的操作，而不在于较快速的数据处理速度。另一方面，也难以编写出对不同型号μP均一视同仁、不偏不倚的试验程序。再则，要写出能针对设计内容难点且有挑战性的软件，也是十分困难而费时的。最后，在许多设计中，其实并不要求μP的应用达到其最高水平，而且往往是还差得远，因而很多型号的μP都能轻而易举地满足设计上的要求。于是，μP的具体选择将取决于其他一

些因素。

如果典型性能试验是适宜的话，那么试验所用的子程序必须针对设计中的具体应用。应考虑作下列一些操作试验：执行样机原型程序中一些困难的部分，数据传入传出存贮器，子程序的调用操作，数据的复杂计算，与外围交换数据，A/D或D/A变换，对中断的处理和服务等。

（3）功能划分

在选定μP之后，根据初始技术指标及要求，一般采用自上而下的方法，把产品划分为若干个主要功能部分。每一部分又再划分为较细的功能部分。……一直细分到最下一层。最下层的每一功能，应具有若干预定的性能，而且应能用一个算法来加以描述，并且其输入和输出应能予以定义。

这样的功能划分工作，有利于最后精确指标集的修订。在多人组成的设计组中，也有助于划分软、硬件设计师各人之间的职责。

有些功能块，应该用硬件或软件来实现，可能是很明显的。有些则可能不甚分明，而主要取决于应用上的要求，例如取决于今后生产批量的大小。有些功能可能用软件（或硬件）实现起来较为容易，或稍为困难一些。有些功能则可能用软件或硬件方法实现均可，而无所轩轾。

这种计划工作，作得好时，常能节省设计的投资和时间。若计划不周，则很可能导致今后调试困难，返工甚多，耽搁很长时间。

（4）软件开发

应该尽早就筹划软件开发的问题。对智能仪器设计面临的局面进行分析，从而决定怎样来进行软件开发。采用汇编语言或采用高级语言来作开发工作，或二者兼用？使用汇编语言，在程序编制得当之时，能节约存贮空间并提高运行速度。不过其编程的过程则较慢，而且较易出错。反之，使用高级语言，则编程较易而且迅速，也不易出错；但占内存空间大得多，而且执行速度较慢。此外，最后从高级语言翻译为机器的目的码时，所得结果也许会颇为累赘冗长。当然也可以采用汇编语言和高级语言混合使用，如果这对设计适宜而且开发、支持系统也能容许的话。关于这方面，在后面第1·4节中还将进一步论述。

总之，开发所用语言选择的倾向性，取决于要求或容许软件工作费时的长短，要求达到的软件效率的高低，可供使用的存贮空间大小，已有的开发经验，手头现有的开发设备，等等。软件开发手段可能有四种：利用本单位的主计算机；利用本单位的一台微计算机；利用本地计算机网的分时服务；利用现成的μP开发系统。

在上述各项筹划之后，设计对象的初始技术指标可能需要作某些修改或细致化。可能需要几次反复，技术指标随之逐步更臻完善，最后可成为具体设计的指南。

（5）设备筹措

软件开发途径决定了之后，即可着手筹措设计所需用的设备。可能需要考虑下列一些设备：万用表、模拟式示波器、逻辑探头、数字锁存器、数字延时单元、逻辑分析仪、μP开发系统等。μP开发系统是强有力的工具，但也比较复杂。若决定使用开发系统，则应及早先行准备，以便有充裕时间熟悉其使用。

1·3 早期硬件设计

关于早期的硬件设计，这里只能作一般化的概论性的讨论，因为硬件设计的具体步骤，与设计对象的具体应用紧密相关，常不能一概而论。一般而言，早期硬件设计过程，大体上如图1.2所示意。兹将各阶级工作分述如下：

（1）设计的筹划

如前面所述那样，采取自上而下的办法，对硬件进一步细分功能块。最终应达到如下目的：

(a) 明确应完成一些什么样的硬件任务，及其完成的先后次序，以便作出进度计划及人力安排；

(b) 据之拟出计划使用开发设备的时间表；

(c) 反映出所需元部件，以便能及早先行订购远期元部件，并作出经费概算。

（2）设计的评审

评审整个设计计划是否符合设计的总目标及总决策。在计划阶级中，一般是一点一滴地逐步寻求折衷和协调的。尽管每一步的折衷、协调单独看来可能合理，但总合起来后，也可能与总目标不尽相符。因此在这一阶级的评审，有便于在开始具体设计之前能有修改余地。

然后，硬件主管人员再与软件主管人员会同联合审核，以保证软、硬件设计的配合。

（3）准备工作

具体设计的准备工作，包括准备场所、设备、人员，订购远期元部件，作出经费预算，拟定工作进度计划等。

人力安排、调配，可能伸延到今后好几步工作中，可能视具体情况而定而需作调整。

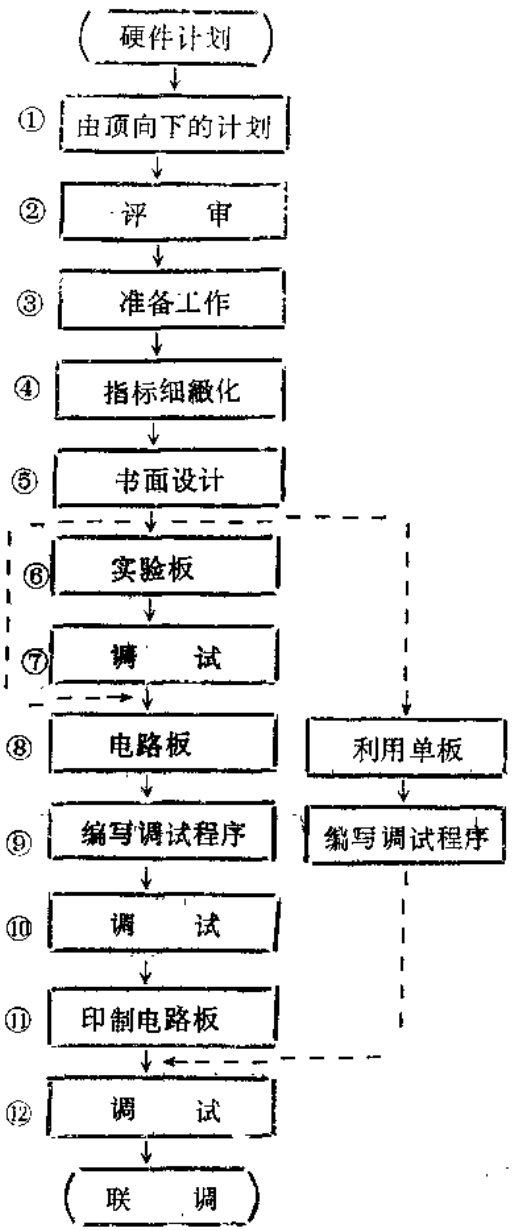


图1.2 早期硬件设计过程

准备工作中最重要的一环是进度计划。在开发初样的总时间中，可能一半以上的时间将会化在组装及联调阶级。经验不足的工程师往往可能过于乐观，其实最好是多准备些时间给后来的软、硬件联调。

(4)指标的细致化

指标的细微化工作，将会在开发过程中多次反复修改、逐步深入，难以逆料。正因为如此，所以更应尽早及时细微修订，不应拖延。否则可能导致设计及支援工作上的混乱，甚至使整个设计工作失控，难以收拾。

(5)拟定电路图

硬件设计者应作出完善的电路图。首先当然是作纸面设计。当设计具有重大革新或创新性质时，则可能要在实验板与电路图之间反复折腾。最终的电路图将是理论与实践结合的产物。

(6)实验板工作

电路的纸面设计是一回事，而保证它能付诸实际工作则往往又是另一回事。因此，大都需要制作电路实验板，以资验证，并帮助修改电路图，使之终于完善。电路实验板可能是一个模块（一个电路单元），代表若干步逻辑乃至一系列功能，其规模的大小常取决于具体应用，亦反映设计师个人的设计风格。一般，电路实验板上元件的安装排列布置及走线并不是主要的考虑，主要着重考虑的是其电气功能。

在某些紧急的设计场合，可以利用买来的现成单板 μP 系统，而不另行自制实验板。单板 μP 系统中，若干支持电路及几乎一切联线都是现成的，设计者无需在信号的定时、电路的布置、电源的电平等问题上费神费事。若所设计的是单台产品，或者甚至中等批量的产品，则利用 μP 单板作实验可能比自行制作实验板更为节约。在大批量产品的设计中，利用现成 μP 单板作实验，则未必合适。

利用μP单板作实验板时，软件人员也就不必为硬件的指标规定而操心，因为与软件有关的一些参量都是现成地可明确规定的。此外，不少生产μP单板的厂商已在提供一些软件包，来供设计者使用。这也可以节省不少软件劳务。

利用现成的 μP 单板来作设计，其不足之处，除了成本上的考虑之外，在于可供选择的单板型号为数不多，软件包所提供的支持亦相当有限。此外，随着微计算机发展步伐的加速，所能获得的单板常常落后于先进水平。

(7)实验板调试

制作实验板的目的，是供调试之用，以便据之修正、改进并验证设计。请注意，在图1·2中的(7)、(10)、(12)各步均需作调试工作。应该尽可能在后期设计的软、硬件联调开始之前，把一切硬件毛病都找出来，并予以排除。否则在今后联调时就不好办。

(8)组装连线电路板

如前所述，在作实验板时，通常专注于功能试验，对结构布置不甚在意。对实验板初步调试修改好之后，才组装较为正规的连线电路板。通常是利用布有密集排列的连线插孔的板子，仔细安排元件位置，注意结构及走线。

也有不少有经验的工程师，由于胸有成竹，而绕过上述第(6)、(7)两步，而直接从组装连线电路板的工作开始实验工作。

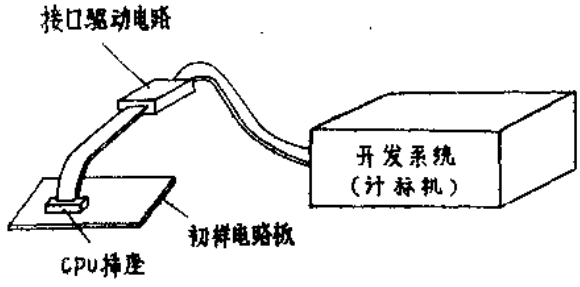
(9)编写调试程序

一旦对所设计的电路硬件排除了故障之后，就要看它能否工作于初样程序的某些段落或子程序，或者专门设计出一些调试程序段来对硬件进行检查。大多数硬件设计师都

是自己动手，用汇编语言来自行编写检查调试用的程序。因为在这个阶段，软件人员大概正忙于自己的工作，也许不能及时提供初步调试所需用的程序。

(10)用开发系统来作调试

把专供调试用的程序装入 μP 开发系统，把开发系统的仿真探头插入初样电路板的CPU插座中来取代CPU，如图1·3所示。然后让初样硬件来执行这些调试程序。



图·13　　用开发系统进行仿真调试

虽然在设计过程的后期有专门的调试阶段，但事实上从硬件设计之始就陆续不断地进行调试工作。我们不能把一切问题都留到最后调试阶段去解决。在现阶段，应力求排除草样电路板上的一切逻辑不正常现象，并解决一些主要的定时问题。

由于连线电路板不及印制电路板那么精密，其阻抗特性将会与正规印制电路板不完全一样。不过，印制电路板却含有其固有的寄生电容耦合，这一点亦应注意。

(11)加工印制电路板

印制电路板的制作一般要费较长时间，在车间生产可能需5—7个星期（连安装元件及核查）。有专门的试制车间及相应人员时，可能也需2—3星期才能完成。如果板子要反复修改重作三、四次，则将会占了总开发时间的相当大部分，从而影响整个进度及人力安排。

技术主管人员都希望硬、软件设计工作进度能同时完成，以便及时进行总联调。平均而言，软件工作约占样机总设计工作量的75%左右或更多。初样印制板工作所费时间，一般大概不致成为严重问题。

(12)印制电路板的调试

初样印制板加工完毕后，大概先要作一些初步的功能及逻辑检验，肯定它能工作后，才在 μP 开发系统上作电路内仿真（In-circuit emulation）。开发系统内装入必要的调试程序。若受试印制板上含有 CPU，则可将开发系统的仿真探头插入受试印制板的CPU插座上以取代CPU。若CPU 不在受试板上而在另一块印制电路板上，则应把开发系统的仿真探头插在后者的 CPU 插座上，再令该板与受试板联起来作仿真调试。

调试中一般大概会发现有硬件问题，这时就需回到实验桌上去仔细研究虚假信号、竞争状态及其他不正常操作，设法加以排除，然后再重新调试。直到不出问题时为止。

(13)可维修性设计

最后应强调指出，在产品设计的计划阶段即应考虑今后产品的可维修性问题。在早期设计中同样应考虑如何作到使产品便于维修。甚至可以设计在产品中加入若干故障检查、寻踪手段。这当然会增加生产成本，但可以节约今后产品运行中的维修费用。这两方面当然要作一定的折衷考虑。

1·4 早期软件设计

关于软件的开发，这里假定是用高级语言（例如BASIC或PASCAL 等）来编写，并利用汇编语言来作程序优化。这样就可以有机会同时讨论二类语言的软件开发问题。

软件开发肯定是要针对设计对象的应用来进行，也同样不可完全一概而论。图1·4示软件开发过程的概略。图中（1）至（4）步当然可以交错起来。而由第（5）步开始的以后各步，则会受到具体开发系统的影响。若使用另一种不同的开发系统，则图 1·4 整个框图的结构就可能不大一样。不过，所涉及的一些基本原理和原则，则是共同的。

在对图 1·4 各方框的内容进行阐述之前，有必要先讨论一下这里为何要使用两类语言？为何要用汇编语言作优化？在什么地方可以进行优化？

读者对于汇编语言编程已有初步基础，大概知道在使用面向机器的汇编语言编程时，事实上是在控制着 μP 内部一步接一步的动作。因此，就有可能使过程达到最优化，即是最好地利用 μP 系统的资源、尽可能节省内存空间、加速整个程序的运行速度。汇编语言的缺点在于要编写出大量代码（助记符），而且时常不免出错。一般而言，不论用何种语言编程，出错总是难免的，而且出错的次数一般正比于程序代码的行数。由于高级语言的一行，常相当于汇编语言程序的许多行代码，所以汇编语言程序的编写过程中将会出现更多的错误，从而会大大增加随后对程序的调试时间。

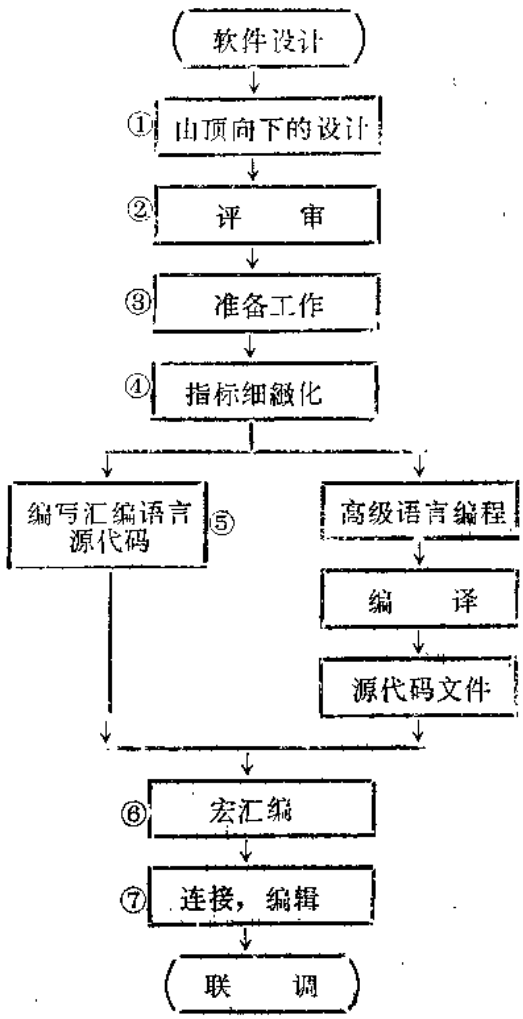


图1·4　早期软件开发过程

当然，若汇编程序具备宏指令能力，则可在一定程度上简约编程劳务。宏指令有二类操作方式，即所谓呆操作和巧操作。呆操作就是简单地用一条宏指令来固定地代替若干条简单指令的序列，实际操作是直接的在线代码扩展，即是把一条宏指令化为一列真指令。巧操作则是利用赋予宏指令的各参数来产生可选择的代码扩展，也可以设计得使宏指令对汇编过程早先发生的某事件起响应。这种巧操作宏指令，事实上是汇编语言的一种复杂巧妙的增强（扩充），要创造出这样的一些宏指令是要费相当时间精力的。不过，如果这样的一条宏指令若在程序中被使用四、五次以上，则往往值得费工夫去创造

它。要使宏指令有用而且高效，应尽可能预见编程上所需要的普适性。

宏指令有助于加速汇编语言程序的开发速度，但并一定会降低所得代码的效率，也不会妨碍我们了解最后会产生出多少行代码。汇编程序打印出来的汇编语言程序清单中，可以自动开列出宏指令的在线代码扩展。

另一方面，使用高级语言编程时，一行高级语言语句即产生出一行或多行乃至好几十行机器代码，这使编程工作迅速得多，而且减少了出错的机会。高级语言是面向解题的，它表示着算法；汇编语言是面向机器的，它表示着数据的传递和基本操作。高级语言程序所产生的代码，对于给出的算法而言总是正确的；不会象汇编语言编程时那样会出现偶然的错误，例如误把某个字节（数据）传送到一个不合适的场所。用汇编语言表达一个算法是不大容易的，而用高级语言则能容易地表达算法，这本来就是高级语言的目的。

高级语言程序易读易懂，可以一目十行地浏览一大段程序。在汇编语言程序中，一个程序转移或重复循环可能涉及到五、六页之外，一眼看不到那么远。此外，用高级语言编写书面文件也较容易。

若初样设计只含有有限的内存空间，或有特殊的高速要求，则应警惕高级语言编程的缺点。若要时刻费神留意内存空间和执行速度，这就与使用高级语言的初衷相违背。因为采用高级语言的目的，本来就是可以在编程时不必顾虑数据放在何处、如何传递，而可以专注于算法的解决。当然，有经验的编程员，如果愿意费神保持跟踪，是能够了解他的高级语言程序会产生出多长的代码的。不过一般很少会这样自找麻烦，除非是发生内存溢出时才被迫为之。

应该知道，有些事情是高级语言做不到的。当设计原型要与其他器件或软件包相联时，有些东西是必须明确地知道或予以明确规定的。在使用高级语言时，遇到这种情况就可能发生困难。在高级语言编程中，一般不必管数据在什么地方，例如，在计算过程中，根本不必在意数据是否存于存贮器中或暂存在某个寄存器里，而事实上这也由不得你直接控制。这些正是高级语言令人喜爱之处。但是，如果所设计的样机要联到某个操作系统中，那么就需要对数据实行控制。例如，操作系统可能要求一些参数以某种次序存放于样机的堆栈中，而你的高级语言程序则可能并不是按这种次序把各参数置于堆栈中的。这样，就必需调用一个汇编语言子程序来重新整理堆栈内容，以满足操作系统的要求。与操作系统的接口工作完成后，又要调用另一个汇编语言子程序来把堆栈恢复到原状，以便高级语言程序能继续执行下去。在诸如此类的情况下，就不得不用汇编语言子程序来作为高级语言程序的补充，以进行必要的数据排布整理。

对于高级语言程序是否需要进行优化，这取决于设计的具体应用。如前所述，若要求运行快速，若存贮空间成问题，或者需要对数据的控制超出了所用高级语言的能力之外，则利用汇编语言来作优化是适宜的。有一句老话说：编码的20%完成了工作的80%。如果能识别这关键的20%代码，并使之优化，则肯定可以大大提高整个程序的执行速度。一个程序中，用于高速中断的部分，作高精度数字处理的部分，有严格时限的子程序，等等，都是需要考虑优化的程序段。

在表过了上面关于语言和优化的一翻概论之后，现在我们回过头来，就图 1·4 中各

方框分别加以阐述。

（1） 计划

软件人员从两个文件开始工作。其一是第1·2节中所述的自上而下的计划，这是由软、硬件主管人员联合发展而形成的；计划不仅把样机划分为有明确规定的若干个任务，也指出了其中哪些任务要由软件人员来完成。另一个文件是硬件的技术指标，这些指标提出了对硬件的要求，并随后由硬件人员再予以进一步细致化；硬件指标同时也是今后软件计划所必需的基础。

关于软件的开发方法，各说不一。当然，语言本身也自然带着一定的倾向性。例如，对于用 BASIC 语言作开发而言，线性规划法是相当自然的；而结构化编程方法，则适于使用PASCAL语言来开发（见第6章）；等等。

在对软件工作进行详细计划之前，必须知道硬件将是什么样子、它将怎样工作。要了解使用者将通过何种手段来与软件作联系——用一组开关？还是用键盘？等等。要了解样机面板上将有些什么开关、按钮，它们将起些什么作用？指示灯应在何时点亮？为何亮？诸如此类。

有了总体计划和硬件指标，软件人员就把计划中的软件部分发展到能使软件指标细致化，并达到能对软件组每一成员指派具体的开发任务的程度。这与拟定总体计划时的自上而下划分任务的办法相类似。

在自上而下划分软件任务时，初时较粗分的任务并不牵涉到具体的 μP，但逐步往下就会反映出 μP 的具体结构来了。软件任务一直细分到能开始具体编程的程度为止。要用汇编语言编程时，软件任务一般比高级语言编程更细一、二级。

在各划分级别上，都有一些数据要传送到这整个金字塔式软件模块体系的不同部分去。在每一级别上，都应记下正在使用的变量，以及哪些软件模块需要访问这些变量。这是很重要的。

自上而下划分任务后，即可对软件人员分派具体任务。这时，自然也就可以清楚看出其中哪些是需要作优化的关键部分。

（2） 评审

到了这时，软件主管人就可以回过头来重新评审软件组所作出的决策和各种决定，并与硬件主管人会议与硬件有关联的一些决定。如第1·3节所指出过的，在局部看来合理的决定，从全局来看未必一定合适。

（3） 准备工作

开始具体开发软件之前，准备工作包括准备工作场所、筹措必要设备、提出经费预算、安排人员、拟定工作进度等。与此同时，对软件的指标和要求进一步精细化。

（4） 软件开发

软件人员分到若干个划分出来的软件任务（通常是互相有关联的一组模块）之后，先考虑各模块开发的先后次序，特别是当各模块彼此有联系时，尤应分清先后。对于每一模块，首先应进行分析，不要急于马上着手编程。要考虑整个系统的工作过程应是怎样的，考虑该模块所需的输入数据及其应输出的数据，研究合适的算法及中间的软件过程，等等。这些分析研究，应写成书面材料。否则，单凭头脑瞬思默记，常会有些问题