05 年开始接触嵌入式技术，这 10 几年里经历了很多项目，遇到了无数技术点，从一个涉事不深的初学者，成长为了现在还算称得上是“资深”的技术者。这期间我还办演着另一个角色--技术的传授者和解惑者。我乐于这种技术者之间的交流，它时常会带给我反思、领悟和动力，让我一直保持追求新技术新高度的热情和信念。

在交流中，我无数次的被问到一个问题：“如何才能学好嵌入式 C 语言？”我也确实深深感觉到：C 语言的掌握和理解程度严重制约着嵌入式技术者的研发能力。其实，我的研发水平很大程度上得益于我 C 语言的扎实基础和对其深入细致的理解。我结合我早期的学习经历来讲述嵌入式 C 语言应该如何学好。也许，我的经历不易复制，但它作为一种学习的方式，大家可以多少借鉴。

引用我的启蒙老师的一句话：“C 语言，学得多精都不为过！”

性格决定你所能从事的事业，也决定了你的命运。我的性格是对新鲜事物有极大的好奇，而且这种好奇会发展为兴趣，并最终狂热。当我脑子里出现一个想法，那我会迫不及待地去实现它，而且不看到它最后的样子，不论成败，不会轻易放弃。在我记忆里，我小时候就是这样的。有一次我看到一根铁丝，正好我衣兜里有一个皮筋，于是我就作了一个弹弓。后来一发不可收拾，我迷上了作弹弓，各种各样，大大小小。还不乏创新和发挥，我想到在电影里见过的弩(其实我还不知道它叫弩)，于是接下来的很长时间我一直在研究如何用铁丝制造一把弩。最终，弩出现了。原来玩弹弓时的纸子弹被我换成了石子，随着扣动扳机，我的屁股也开花了。

上初中的时候，我通过学校开设的兴趣班接触计算机，第一次知道了 DOS、Windows98、WPS、输入法、全角半角这些东西，迅速燃起对计算机的好奇和兴趣。开始渴望拥有自己的计算机。碍于当时的经济条件，还有毕竟文化课学习为重，最后家里买了一台学习机。用它可以模拟 DOS 环境、练习指法，还有一些简单的编程，如 LOGO、BASIC 等。从此，我开始有了最基本的编程意识：程序就是一行行挨个运行语句。但是对循环、条件判断还很陌生。

对计算机的兴趣不像其它事情一样，一段时间玩透了，也就放下了。我发现计算机要学的东西非常多，而且它好像一直都在变化出新，这些新的东西又会再一次掀起我的兴趣。对计算机的狂热从上了高中后开始了。长期基于学习机的练习，我的指法已经足够熟练，但是用拼音输入文字速度太慢，所以我报班学了五笔，一直延用至今，虽然我已把字根忘光了。然后在软磨硬泡之下，我拥有了第一台自己的奔 IV 电脑，从此我的“折腾”开始了。

平时一有时间，就研究 QBasic、VB、软件加解密、网络攻防这些东西，还订阅了杂志《电脑爱好者》，期期不落。到高考前，我应该可以称得上是半个“业余电脑专家”了，也已经可以使用 VB 开发一些小的桌面软件，比如计算器、小游戏。计算机让我的好奇心得到了很大的满足，也使我的创造力得到了施展。

基于我对计算机的浓厚兴趣，报志愿的时候，我四个院校全部报了计算机专业，从那时起，注定了我将以计算机为伴、为业。

原以为进了大学，就能马上接受正统的计算机课程教育了，其实并不是。计算机专业一开始并不直接学编程，而是学数学。我当时比较迷茫，觉得学计算机不教编程，上学有什么用？其实我知道专业课程安排的用意，计算机科学的基础是数学，应该先打基础。但是又有多少学生真正去好好学习这些基础性的枯燥的东西。导致很多人整个大一的宝贵时间都浪费在游戏上，估计他们已经忘了自己为什么要学计算机了。我也怕会变成这样。

我开始自学很多计算机方面的知识，但是又漫无目的，直到我碰到一个“能人”。据说他小学开始学计算机，初中已经可以独立开发软件，高中时因为开发了一个网络软件，被什么软件平台收录，并评为五星软件，而被免试特招。它智商高，但似乎情商不是太高，经常容易得罪人，有一些让别人不太舒服的作事风格。有一次我们偶然聊天，他提到国际 ACM 程序设计竞赛的事情，问我有没有兴趣参加，说已经集结了五六个人，组成小组，参加比赛。

从此我开始有了动力，开始自学 C 语言，学习算法，参加团队集训，下载往年竞赛题目模拟竞技，相互交流经验。当时专业课还没有开 C 语言，但是我们已经是都是 C 语言的高手了。也许，应该在这里放一道 ACM 竞赛的试题给大家解闷(这是一道陈年老题，感兴趣的话可以百度)。

对！ACM 试题是全英文的。

Description

Background

For years, computer scientists have been trying to find efficient solutions to different computing problems. For some of them efficient algorithms are already available, these are the "easy" problems like sorting, evaluating a polynomial or finding the shortest path in a graph. For the "hard" ones only exponential-time algorithms are known. The traveling-salesman problem belongs to this latter group. Given a set of N towns and roads between these towns, the problem is to compute the shortest path allowing a salesman to visit each of the towns once and only once and return to the starting point.

Problem

The president of Gridland has hired you to design a program that calculates the length of the shortest traveling-salesman tour for the towns in the country. In Gridland, there is one town at each of the points of a rectangular grid. Roads run from every town in the directions North, Northwest, West, Southwest, South, Southeast, East, and Northeast, provided that there is a neighbouring town in that direction. The distance between neighbouring towns in directions North-South or East-West is 1 unit. The length of the roads is measured by the Euclidean distance. For example, Figure 7 shows 2*3-Gridland, i.e., a rectangular grid of dimensions 2 by 3. In 2 * 3-Gridland, the shortest tour has length 6.

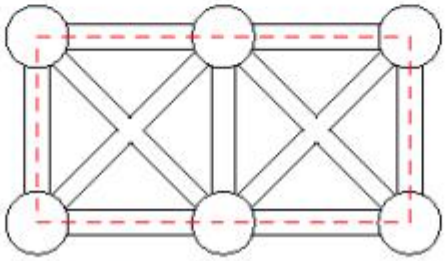


Figure 7: A traveling-salesman tour in 2*3-Gridland.

图 1.1 国际 ACM 程序设计竞赛历届真题

大二下学期，C 语言专业课开了。很多人并不知道 C 语言的什么用，带着迷茫上课、考试、通过。我开始慢慢深入感觉到 C 语言的奇妙，它有自己严格的语法规则，但是又不作过多限定，这让它非常灵活而实用。同一个逻辑功能，可以有很多种 C 语言的表达方式，它一定程度上体现出了编程者自身的习惯和素质。代码可以写得很乱，也可以写得很优雅；可以写得冗长啰嗦，也可以写得如蜻蜓点水，几行了事。但是乱也可以错落有致，寥寥几行也可以大显功底。我意识到，C 语言没那么简单，不仅仅是一门语言而已，它会伴随我一生，正如后来有人所说的“程序如人生”。

我的性格仍然在发挥着巨大的作用。不断的学 C 语言，用 C 语言，我越觉得 C 语言的乐趣多多，如同挖矿，永远都有那些未曾遍及的角落，永远都有没有见过的另类写法，永远都有富含创意的智慧的流露。

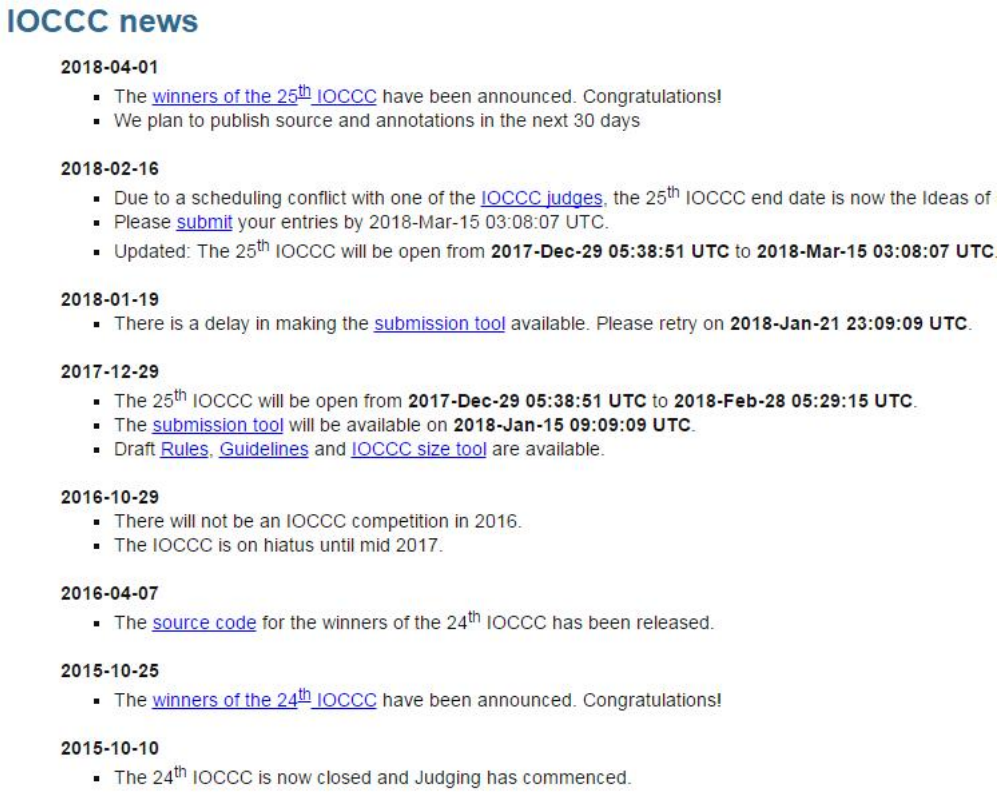
在 C 语言专业课上，老师告诫我们：“C 语言，学得多精都不为过！”很多人可能当时并

没有完全理解这句话，但是我却深深的赞同。当然，我对 C 语言的理解和编程水平，是受到老师的赞赏的。后来，我们的参赛组和我们对编程的热情，最终感染了整个计算机学院，而这位 C 语言老师，也成为了我们的集训老师。最后来，我们的举动，带动了更多人参加 ACM 程序设计竞赛的意愿，学院、学校、乃至哈尔滨市、黑龙江省、东北三省。最终国际 ACM 委员会委任我们学校为国际 ACM 中国东北赛区承办方(南方赛区是浙大)。当时，全校到处都挂满了条幅：“Program Your World!”

关于 ACM 程序设计竞赛，当年不有一个关于浙大的传奇故事(浙大是当年的 ACM 世界总冠军)。我们知道写程序要经历编程、编译、查错修改、再编译，如此往复，若干次。这个往复的次数，与程序的难度与程序员的能力有很大关系。但是要作到所有一切一次成功，极难。当年浙大参加总决赛，只剩十分钟，还有最后一道试题。参赛队员，打开记事本，直接写代码，直接提交，一次通过。这件事情，在圈子里流传，也许有夸张的成分。但是，也足以显示我们与顶尖编程高手之间的巨大差距。

ACM 竞赛，就是一群疯狂热爱计算机和编程的人们，一起正在作的事情。这些人的技术基础夯实，他们以不断猎奇、不断学习、不断完成新的目标为最大乐趣。

而我，私下在关注另一个国际编程竞赛，IOCCC(国际混乱 C 代码大赛，官方网站 http://www.ioccc.org/，如图 1.1)。其实很多人都不知道这个比赛，我也是偶然间发现的。



IOCCC news

2018-04-01

- The winners of the 25th IOCCC have been announced. Congratulations!
- We plan to publish source and annotations in the next 30 days

2018-02-16

- Due to a scheduling conflict with one of the IOCCC judges, the 25th IOCCC end date is now the Ideas of
- Please submit your entries by 2018-Mar-15 03:08:07 UTC.
- Updated: The 25th IOCCC will be open from **2017-Dec-29 05:38:51 UTC** to **2018-Mar-15 03:08:07 UTC**.

2018-01-19

- There is a delay in making the submission tool available. Please retry on **2018-Jan-21 23:09:09 UTC**.

2017-12-29

- The 25th IOCCC will be open from **2017-Dec-29 05:38:51 UTC** to **2018-Feb-28 05:29:15 UTC**.
- The submission tool will be available on **2018-Jan-15 09:09:09 UTC**.
- Draft Rules, Guidelines and IOCCC size tool are available.

2016-10-29

- There will not be an IOCCC competition in 2016.
- The IOCCC is on hiatus until mid 2017.

2016-04-07

- The source code for the winners of the 24th IOCCC has been released.

2015-10-25

- The winners of the 24th IOCCC have been announced. Congratulations!

2015-10-10

- The 24th IOCCC is now closed and Judging has commenced.

图 1.2 IOCCC 官网历届比赛的消息公布

为什么会关注这样一个似乎不太正经的比赛？它不比算法，也不比代码的质量和效率，而是比谁的代码最乱，但是乱得要艺术，要能编译，要能实现正确的功能，如图 1.2 所示代码。

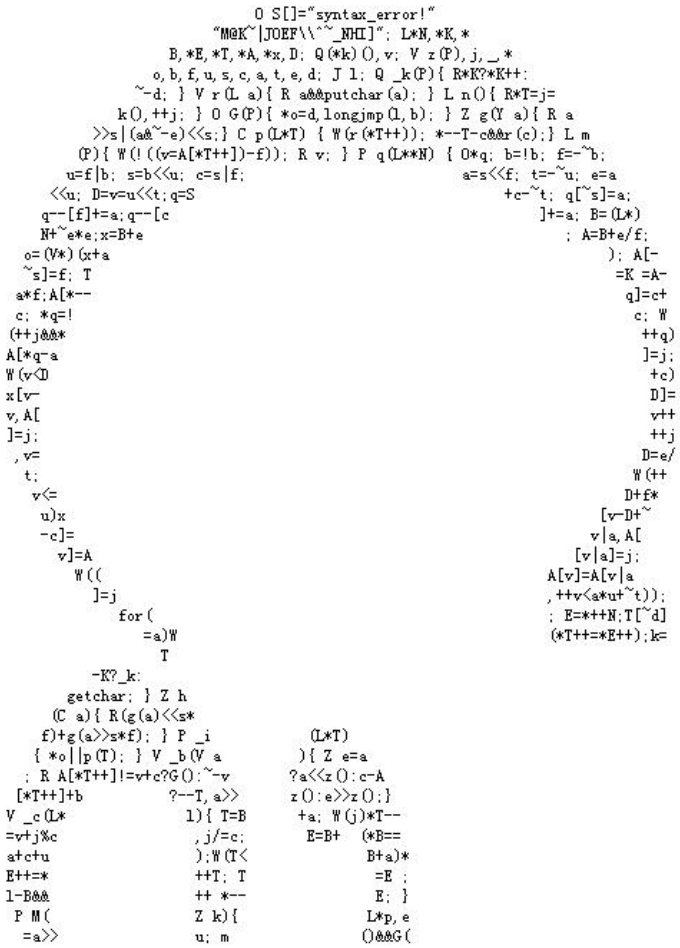


图 1.3 第 24 届 IOCCC(2015 年)参赛代码

C 语言代码还能写得如此任性？它体现了 C 语言在形式上的灵活性。当然，也不是单纯用代码来画画，就能被称为“乱”的，它有更多更深层的编程技巧。在这里，你可以看到 C 语言世界的无奇不有，各种挥挥洒洒的编程风格，以及映射出来的代码背后的那个“高手”。

我对 C 语言的学习热情是自始至终的，现在也还是在学习。记得大学时候我们宿舍有一个习惯，就是大考之后的晚上要倾巢出动去网吧包宿。当时流行玩 CS，他们联网打得热闹。但是我对游戏毫无兴趣，就窝在一个靠边的位置上，上网看 C 语言代码。当时特别热衷逛论坛，什么 csdn、pudn 等，还喜欢把代码包全下载下来，看看别人的代码是怎么写的。看了代码，就想编译试试，于是就在网吧的电脑上安装 VC6.0。室友过来看我在干啥，然后就惊呆了：“都考完了，你还在看 C 语言？”

接下来，我继续学了 C++，还有后来的 MATLAB、VHDL 和 Verlog(其实当时对硬件、数字信号处理和仿真没什么概念，所以对于后者没有多少热情)。在学 C++之前，我使用 VC6.0 已经有一定经验了，尤其是 MFC(当时有人建议我学.NET，比如 C#，说 MFC 已经过时，说 MFC 的意思就是 Maybe Finally Canceled，即最终会被微软取消)。在系统学了 C++之后，我对 MFC(微软基础类库)有了深入的理解，开始阅读这方面的一些专业书籍，已经可以编写一些功能复杂的多层级的应用软件了。

到这里，我对编程的学习开始出现瓶颈，感觉到迷茫。C 语言，很优雅，很强大，它的父集 C++，面向对象的编程模式，可以开发专业的桌面软件。然后呢？似乎其它人用 C#或 JAVA，开发软件的速度更快，作得更好。优雅不能当饭吃，在这种驱使下，我开始转入 C#、JAVA、PHP、JSP、ASP 这些上层应用级语言的学习，准备努力成为一个出色的软件工程师。

在这个岔路口上，我遇到了我的启蒙老师，让我再一次打开好奇之门，从而走上了嵌入式技术的道路。他就是杜撰(化名)：他以年龄最小、学历最低的身份，代表黑龙江省参加全国“挑战杯”科技创新大赛，凭自己设计的“仿生蛇”获二等奖。获奖后，他把制作理论与技术全部无偿交给了国防科工大。曾作为“小崔说事”栏目的特邀嘉宾接受专访。

图 1.4 我的启蒙老师杜撰(化名)与它的“仿生蛇”

当时我对单片机完全不了解，只知道在他的“仿生蛇”里使用了单片机，而且单片机可以用 C 语言进行编程开发。单片机可以作出如此强大的东西，它远比在电脑上写桌面软件要有趣的多。正是这一点，深深吸引了我。

“你从 51 开始学吧，先焊个最小系统，然后点个灯！”他给了我一个最小系统板、芯片和一些配件，还有一个叫《平凡的 C51 教程》的电子文档。“你就在我这学吧，有问题问我。”

在学会使用汇编语言后，学习 C 语言编程是一件比较容易的事，我们将通过一系列的实例介绍 C 语言编程的方法。图 1-1 所示电路图使用 89S52 单片机作为主芯片，这种单片机性属于 80C51 系列，其内部有 8K 的 FLASH ROM,可以反复擦写，并有 ISP 功能，支持在线下载，非常适于做实验。89S52 的 P1 引脚上接 8 个发光二极管，P3.2~P3.4 引脚上接 4 个按钮开关，我们的任务是让接在 P1 引脚上的发光二极管按要求发光。

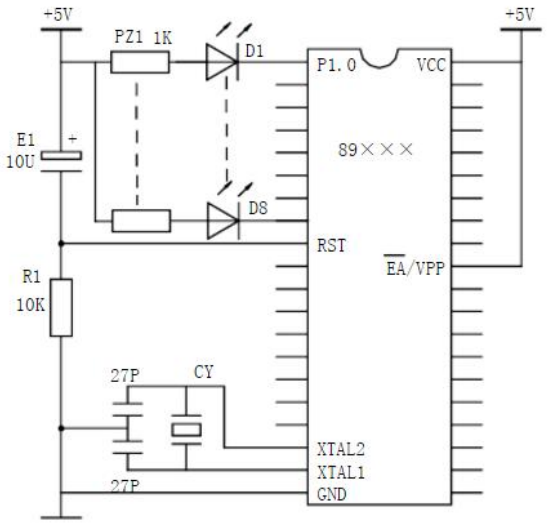


图 1-1 接有 LED 的单片机基本电路

图 1.5 我的第一个单片机实验(此图摘自网文《单片机 Keil C51 编程入门》)

我依葫芦画瓢的焊完了我人生中第一个电路板，虽然惨不忍睹，但是经他过目之后，评价是“还不错，能用。”然后，就让我去学 C51。其实我对这个被称为“最小系统”的电路为什么要用到这些元件，为什么要焊成这样，完成没有概念。带着诸多的迷惑不解开始了我的单片机 C 语言学习之旅，让我开始慢慢明白了“C 语言是最贴近硬件的高级语言”。

C 语言，不论是变量函数，还是分支循环，乃至于算法，哪怕是最复杂的算法，说到底无非就是在读写存储器(内存或硬盘)，不论是读取指令还是读写数据。这是我起初对 C 语言的理解。那如何让 C 语言去操控硬件产生物理效果呢？小到点亮一个灯；大到“仿生蛇”产生一系列的动作；更大的比如控制火箭发动机开始点火。这让我百思不得其解。C 语言与硬件之间似乎有一条鸿沟，它到底是如何逾越的？

```
sfr P1=0x80;

void main(void)
{
 P1=0x55;
 while(1);
}
```

这是我亲手写过的第一个嵌入式 C 语言代码。它为我第一次揭示了 C 语言与硬件之间的交互方式----特殊功能寄存器(Special Function Register)。

register 在标准 C 语言中是一个修饰符。一个变量在定义的时候如何加入这个修饰符，编译器便不会把它分配在内存里，而是直接放在 CPU 内部寄存器里。它的目的是为了加快变量的访问速度，尤其是那些需要被频繁修改的变量。

```
register int i;
register int sum=0;

for(i=1;i<=100;i++)
{
 sum+=i;
}
```

这面段代码(计算 1 到 100 的加和)的执行效率就比没有 register 要高。

51 单片机中的 SFR 本质上也是一些随机存储单元，它们的访问速度很快(与 CPU 之间采用直接寻址)。但其又有特殊之处，它们都是一些电路的门户出入口。向这些寄存器写入数值，会直接影响相关电路的运行和输出。

51 单片机的 C 语言中(归根结底是 C51 编译器)，为这些有特殊功能的寄存器，专门增加了一个修饰符----SFR。由它定义的标识(类似变量名)，是可以访问到相应的特殊功能寄存器的，即电路的出入口，从而达到控制电路的目的。诸多的电路，具有各自的功能，它们纷纷留出数据接口，形成一系列的 SFR。通过 CPU 统一调配、有机控制，最终就可以完成复杂而有序的各种总体功能。这就是单片机，乃至于更高端的嵌入式 CPU，如 ARM、DSP 等均采用的运作机理。而这些电路，连同 CPU 内核，还有存储器，当然还有连接它们之间的总线，被塑料封在一起(即封装)，再把电路(被称为片内外设)的相关外部信号通过引脚引出，这就是我们所看到的单片机芯片了(MCU、微控制器)。其实它就是一个完整的计算机。这也大大拓宽了我起初对计算机认识的范畴：凡是拥有独立计算能力，具备输入输出和存储功能的设

备都可以称为计算机。从某种范范的意义来说，算盘就是最原始的计算机，虽然它很大程度上依赖人的操作和辅助。

一直困惑我的谜雾终于变得清晰了。捅破了这层 C 语言与硬件之间的窗户纸，让我看清了硬件和嵌入式系统的本质。我觉得在硬件上，我将可以发挥更大的创造力。兴趣的泛滥再一次一发不可收拾。

基于我在 C 语言(标准 C)方面的扎实基础和深入的理解，我对单片机的学习也较为顺利。

理解了 51 单片机的 SFR，很多东西便变得简单了。对“C 语言，学得再精也不为过!”这句话有了更深的认识：C 语言不光是一门语言，它影射出了整个计算机系统技术体系的运作机制，每当硬件出现进步，甚至是革命的时候，作为首当其冲的 C 语言，必定会随之进化。不应为 C 语言如何操控硬件而产生疑惑，因为从我们使用 C 语言写下第一行代码的那一刻起，其实我们已经在操控硬件了(内存访问、数据传送、CPU 执行就是硬件行为)。

再说回到 51 单片机，它的 C 语言不光增加了 SFR，还扩展了很多关键字。这一切都服务于 51 单片机 CPU 独特的硬件体系结构。说白了，是针对特定 CPU 对标准 C 语言编译器进行了改进和订制，以便 C 语言可以更好的更准确的描述硬件并对其加以应用。下图是 51 单片机的体系架构。

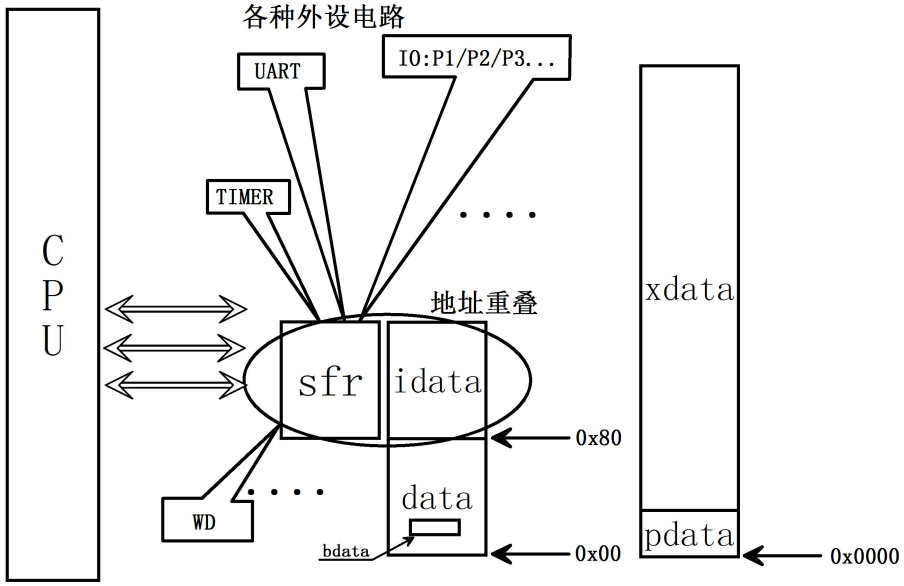


图 1.6 51 单片机的大体硬件结构

简要介绍一下这个硬件结构。51 单片机依寻址方式和效率，对存储空间进行了划分：data、idata、sfr 和 xdata(pdata)，访问速度 data/sfr>idata>pdata/xdata。因此它的 C 语言中也有对应的修饰符，用来描述变量位于哪个存储空间中：data、idata、sfr 和 xdata(pdata)。如果我们希望代码运行快一点，可以使用 data 来修饰变量的定义。

```
data usnigned char i=0;

for(i=0;i<200;i++);
```

这个循环所花费的时间一定要比使用 idata 或 xdata 要少。这也是在 51 单片机上实现 delay 函数或其它对时间较为敏感的代码的时候我们经常会遇到的问题：为什么原方不动的代码突然出错了？