揭开电脑的头盖骨

小易子

一、引子 1
二、掀开电脑的面纱 2
三、计算机的体系结构 5
四、什么是计算、怎么计算？ 6
五、计算机芯片的核心部件---加法器 12
五、芯片啊，芯片 21
六、电脑的语言 28
七、电脑的逻辑推理 30
八、电脑带来的挑战：人就是机器？ 35

一、引子

这是信息的时代，这是计算机的时代，这是电脑的时代。

生活在这个时代的每个人，都直接或间接地用着计算机、受到电脑的影响---各种大大小小、形态各异的计算机遍布工作、生活的各个角落。上至宇航飞行、棋牌训练，下至智能门锁、家用电器，就别说人人都离不开的手机了。每个人都在用电脑，似乎人人都懂计算机。不信，您问一问您的发小："老帽，你这一天至少有十几个个小时都扑在电脑及手机上，这个电脑东东是咋个转的呢？"

您的这位发小老帽，肯定开始不屑一顾："嗨，知道汽车不？汽车里有台发动机，发动机吃油烧气，将油里存储的能量转换成能跑的动能。一个样，这电脑里有个小小的魔术精灵，叫做芯片的；芯片一通电，就将电能就变成不同的信号了！"

如果您继续追问：“这发动机，俺倒是略知一二。里面有什么活塞气缸的，汽油一点着，气缸里面的气体压力变化推动活塞运动，然后这个运动就经过什么传动装置传出来，汽车就可以跑了。老帽，能不能将这个芯片拆解一下，就像倒腾发动机一样，给俺捋一捋？”

十有八九，这个老帽肯定一脸朦胧，“靠，这个，…… ……”

上面这个故事虽是杜撰，但是相同的情形随处可见。特别是，随着中美贸易战的升级，芯片一词变得家喻户晓，都知道这芯片是计算机电脑里最最重要的部件，就像大脑对于人一样。可是，一问这芯片到底是什么具体东西、怎样运作的，估计大多数人都会像上面那个老帽一样，一脸朦胧。

这篇短文就是要消除读者可能存在的关于芯片的疑问，让读者知道一点芯片的最基本知识。希望读者读完本文后，像知道发动机原理一样知道计算机电脑是怎么运行的。

二、掀开电脑的面纱

人类制作使用计算装置的历史非常悠久。据说公元前两千多年的时候，古代巴比伦就有了某种形式的“算盘”；而中国大规模使用算盘达一千多年，直到上个世纪八十年代；在工业革命时期，出现了各种机械装置的计算器具；二次大战期间，英国、德国、美国各自研究制作了以电子电器设备为元件的不同形态类型的“电子计算机”。不过，具有现代计算机体系结构的现代计算机鼻祖、也是世界上的第一台通用计算机，当属 1946 年在美国建成的埃尼阿克，英文简称 ENIAC，即电子数值积分计算机(Electronic Numerical Integrator and Computer) 的英文名称的缩写。

自从那时起，每隔几年，计算机的计算速度、存储信息容量就呈指数级别增长；而计算机的体积则急剧缩小。到了现在，计算机的应用如此广泛，它的功能如此强大，以致大多数人都觉得计算机会思考，像人的大脑一样---很多人就直接称计算机为“电脑”了。笔者也随波逐流，在这篇文章里，交互使用“计算机”与“电脑”两个单词。

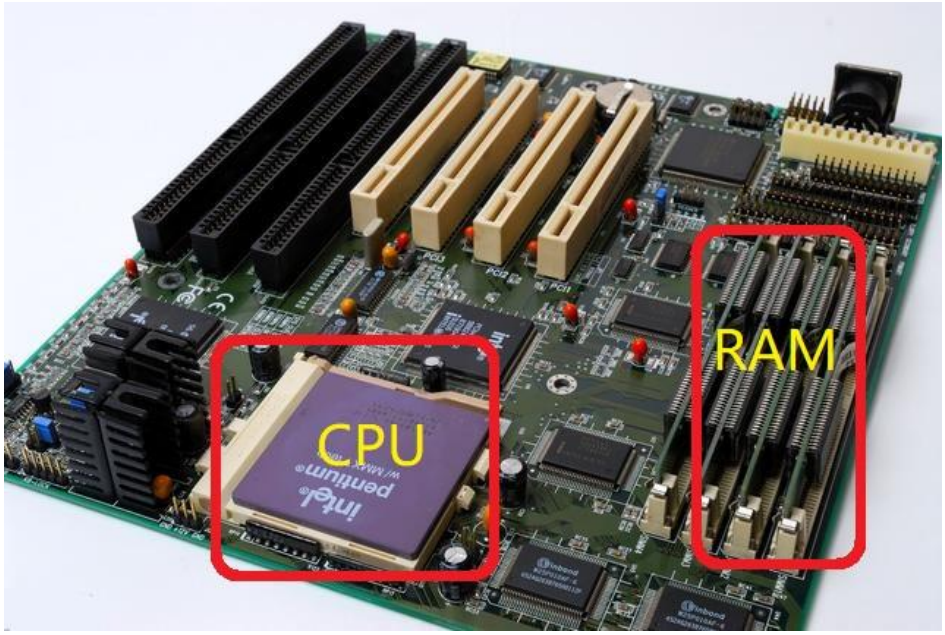
虽然计算机性能不一、大小有别，价格也千差万别，但是所有计算机的工作原理、组成结构都是一样，区别仅仅在于组成部件数量上的多与少、性

能方面的高与低。为了揭示计算机运行的奥秘，笔者用大多数读者都熟悉的台式电脑作样品来剖析计算机的结构。

一架台式计算机，通常看起来是一个黑匣子白盒子或其它颜色小箱子什么的，外连着键盘、鼠标、显示器、音响、打印机、摄像机、扫描仪等一些外部设备。用计算机学科术语称呼，所有这些外联设备，叫做输入/输出设备（即英文 Inputs/Outputs，简称 I/Os）。对于这些输入/输出设备，我们不仅耳熟能详 ，而且都有操作运用的实际经验，所以笔者按下不再赘述。

让我们先来看看这个机箱盒子里面到底有什么东西。笔者用一个较为老式个人台式电脑作为例子。打开这个黑匣子白盒子---也就是计算机的机壳，我们就揭开了电脑最外面的那层薄薄的面纱，可以看到下面的图片，它显示了计算机内部装置与结构。

这个机箱里面的部件有风扇、一些小的电器部件、上面带有插槽的印刷电路板，当然还有很多电线接头。那些外部设备就是通过这一些电线或插槽来连到某个电路板上面的。只要我们仔细一瞧，就发现有一块大的印刷电路板，上面的插槽更多，并且，几乎所有的电线或者小的电路板都与它连在一起。这个电路板，在计算机术语中叫做母版或主板（Motherboard）。为了看的更清楚，笔者用一个比上面计算机主板更小的主板来解说。



在这个主板上，除了连接输入-输出设备的众多插槽及一些小型电子线路控制部件外，有两个最最重要的部件，在上面示意图中用红框边线及黄色字体标出：（1）CPU，即中央处理器（英文 Central Processing Unit 的简称）；（2）RAM，即常说的内存（Random Access Memory，随机存取存储器的缩写）。

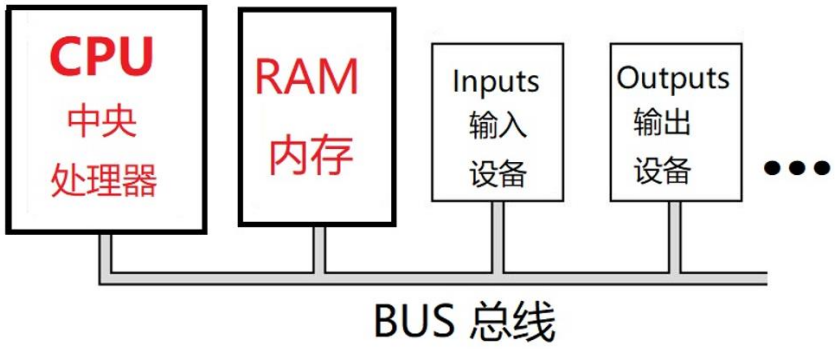
这个 RAM 内存用来存储计算机要处理的所有信息及操作命令。人们在上古时期没有文字的时候就开始记录、存储信息了，比如结绳记事、记数；后来有了文字，我们就用纸莎草植物及后来发明的纸张来记录、存储文献；到了现代，我们熟悉的磁带、DVD、U 盘、硬盘存储媒介都大量地与计算机一道使用。但是这个内存 RAM 与那些外连的存储器有一个很大的不同点：RAM 体积小且存储读取速度非常快。它看起来小小的个头，宽不过 1 厘米、长度从 1 厘米到十几厘米不等、而厚度不过几个毫米，却包含了千百万个微小的电子“元件”。这个内存 RAM 就是我们常常挂在嘴边的芯片中的一种。

与 RAM 内存片一比，CPU 中央处理器这个芯片，更加复杂！仅仅看它们的名字就知道，RAM 是用来存储信息的，就像密密麻麻的邮政信箱，暂时存放信件，其工作性质非常简单；而 CPU 则不同，它是中央处理器，是专门处理信息的，不仅要做数学上的加减乘除等众多复杂程度不一的算术运算，还要作一些逻辑演算、协调控制信息的读取及演算结果的存放。计算机中有大量的各种不同的芯片，CPU 这芯片属于精品中的精品，制作起来更是难上加难，也难怪普通大众就直接把 CPU 芯片当做计算机唯一的芯片了。

现在，我们已经将计算机的机壳打开，将电脑的最外面的画皮撕掉，知道了它们的主要组成部件；下一步，让我们来探索这些主要部件，初步了解它们的工作机制及原理。

三、计算机的体系结构

从上面计算机主板的图解里我们知道，现代计算机有四大类部件：CPU 中央处理器、RAM 内存、众多的 I/O 输入-输出设备、连接这些设备的主板。为了简单起见，我们将这个主板用一组电线表示，这组电线我们称之为 Bus 总线，可以认为这个总线就是去掉了 CPU、RAM、及所有输入输出连接的主板。这样的话，整个计算机系统就可以用下面的框图清晰简明地表示了。



别看这个示意图比幼儿园小朋友的画画还简单，它却恰恰呈现了现代计算机的体系结构；而且它有个五雷轰顶的名字：冯·诺伊曼结构 （von Neumann Architecture），又称冯·诺伊曼模型（von Neumann Model）。冯·诺伊曼出生于匈牙利、曾执教于德国、后入籍美国，是上个世纪上半叶一位真正的天才加科学全才。他在数学、物理等领域做出了开创性的贡献；还被称为现代计算机之父、博弈论之父；他在二战期间参与了曼哈顿计划，第一课原子弹的研制成功他功不可没。

曼哈顿计划的原子弹研制涉及到大量的复杂运算。在使用两台当时的“计算机”过程中，冯·诺伊曼意识到将计算机的指令程序（就是告诉计算机怎么具体操作的一些命令集合）保存在快速的内存中的重要性，就与别的科学家一起描述了计算机应该具有的体系结构，其大致内容及工作原理如下：

1. 计算机指令程序及某些数据存放在内存 RAM；
2. 中央处理器 CPU 包含逻辑-算术运算器，用来执行各种计算机指令；
3. CPU 还包含特别控制元件，用来读取程序指令；
4. CPU 从内存里通过总线一条一条地读取指令、执行相应的操作，将结果返回内存；
5. 中央处理器、内存、输入-输出设备通过总线连接，中央处理器可以对内存及输入-输出设备进行控制操作。

冯·诺伊曼计算机模型的中心思想就是内存存储程序指令、以 CPU 及 RAM 为核心来规范计算机运行的逻辑框架。其实在他之前，就有人提出了类似的计算机结构雏形或概念。但是不管怎么说，冯·诺伊曼及他的同事清晰描述了这个程序存储逻辑框架，随后这个原理大规模应用于计算机的制作并成为现代计算机的模型，人们自然而然地将这个计算机体系结构称作冯·诺伊曼结构，尽管他这个天才及科学全才根本不需要这一个锦上添花的头衔。

从这个结构图示及其解释，我们认识到中央处理器 CPU 及内存 RAM 是最重要的部件，它们两个一起可以比作人的大脑；至于输入-输出设备，好比人的四肢及其它器官。一个人如果有某些残疾，比如肢体不全，甚至耳朵或眼睛残疾，这个人通过某种辅助设施帮助，仍然可以工作、正常生活，尽管会有不同程度上的不方便；但是，一个人患了脑瘫，则无法正常生活、只能依靠上天的怜悯了。在下面，我们的注意力就放在 CPU 中央处理器及 RAM 内存上；不过在解释它们的具体工作原理及物理实现以前，我们还要做一下准备工作。

四、什么是计算、怎么计算?

一听到“什么是计算、怎么计算”这个问题，估计每个中国人都会不屑一顾：全世界都知道，中国人的算术水平名列前茅！即使刚刚进入幼儿园的小朋友，都会一边唱儿歌一边板着指头算数，“五五凑成一双手，四六四六一起走。……，一加九，十只小蝌蚪；二加八，十只老花鸭。……”

到了一年级，我们学会了加法；到了二年级，我们学会了减法；到了三年级，我们学会了多位数的加减混合运算及简单乘法；到了四、五年级，我们就完全掌握了多位数的四则混合运行了。现在，为了完全理解计算机是怎

样计算的，我们得回顾一下加减乘除的运算规则或具体运算方法，敬请读者忍痛耐心地等待几分钟。

首先从加法说起。给出一个两位数加法的例子：37 + 86， 这两数的和是 123，结果一下子变成了三位数。为什么？因为在做加法过程中，有了进位---就是像上面幼儿园儿歌里说的，对应的单个数字相加时，逢十进一；并且，在这个例子中，有三个进位。这个运算倒不难；可以，有一个问题，绝大多数人都忽视了：这些数字都要占用一些空间，比如一个格子放一个数。假设现在我们有三排格子，每排只有仅仅的两个格子，上面两排格子存放加数及被加数，第三排格子存放计算结果。这样的话，我们有：37 + 86 = 23----我们没有格子放和的最高进位了！如果每排格子有三位，这个运算的结果就有足够的格子存放了。依此类推，格子数量越多，可以正确表示的数就越大。这给我们敲了一个警钟：**每排存放算数的格子数量的多少，决定我们运算正确结果的大小**。

现在来看看减法。还是用上面的例子：123 - 86，其差是 37；在这里，一个三位数减去一个两位数，结果是个两位数，为什么？因为我们必须借位。如果读者稍微回忆一下小学低年级开始学这种借位减法时的情形，与加法进位相比，是不是特别讨厌这样的借位、恨不得只学加法？！我们现在就满足一下您儿时的奢望：先稍稍变通一下这个减法，去掉借位、将减法变成加法。第一步，改写算式：123 - 86 = 123 + （-86）；第二步，改造这个减数：将 （-86）用 999 - 86 + 1 = 913 + 1 = **914** 来替代 （9 减去任何个位数的运算都是一种幸福！）；第三步，回到上面说的加法： 123 + 914 = 037 = 37----最高进位丢失了，因为每一个数我们只有三个格子来表示！如果您有点怀疑上面的运算的准确性，我们再举一个例子：567 - 345 = 567 + （**999-345 +1**）= 567 + **655** = 222。在计算机学科中，我们称上面的 914 及 655 为 86 与 345 的**补码**；并且，我们从下面的介绍里马上就知道，计算机中计算补码时，用不着减法，这样的话，**减法就变成了加法**。

到了复习乘法的时候了。还是用一个例子：245 x 23，为了方便起见，假设我们有很多格子表示这些数及它们的乘积。首先让我们回忆一下儿时最惬意的乘法例子，就是什么数乘以 10, 100， 1000， ……。因为我们根本用不着运算，直接移位了事！现在，我们就将乘法转换成移位及加法运算。245 x 23 = 245 x （20 + 3）= 245 x 20 + 245 x 3 = 2450 x 2 + 245 x 3 = 2450 + 2450 + 245 + 245 + 245 = 5635。一句话，**乘法可以转换成一系列的移位及加法**。

至于除法，也类似；我们可以将除法转换成一系列的减法及移位，而这些减法又可以变成加法。为了节约读者的时间，这里就不一一地举例了。总之，在计算机的运算里面，**乘除加减都可以变成加法及移位运算**！

好了，现在我们知道怎么做加减乘除了，让我们开始设计某种“计算机”来做这些运算。 第一步，我们必须寻找某个单位“元件”来表示单位数字，就是 0、1、 ……、9。一个原始简单的想法，就是用一个带有十个齿或槽的齿轮来代表一个数字；如果作五位数字的运算，我们就用一排五个齿轮代表一个数的大小。的确，二战以前的二、三百年期间的最先进的“计算机”---机械计算器，就是利用了齿轮的这个特征。但是，这样的“计算机”计算速度、精度非常有限，造假昂贵，不可能流行开来。

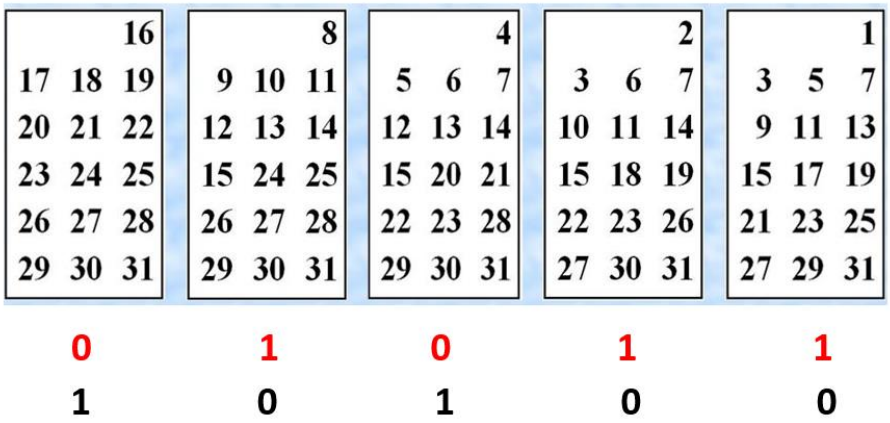
随着第二次工业革命的到来，人类进入了“电气时代”，电子技术也飞速地向前发展；寻找某个电器元件来用作计算机的“基本单位元件”也有了可能。我们可以用一个电器部件的输出输入电压值的大小来相应比照一个数字的大小（0, 1， 2，……，9），这样的话，我们必须精确无误地划定十个阈值范围。比如，在一个输出电压最高为 10 伏的电子信号，确定 0～0.99 伏的信号为 0, 1～1.99 伏的信号为 1，2～2.99 伏的信号为 2，……，9～9.99 伏的信号为 9。这样的划分有个问题：输出电压都是连续递增或递减的信号，还带有误差，如果现在信号电压值是 3 伏而误差可能会有 0.15 伏，那么这个信号到底是数字 2（对应电压是 2.85 伏）还是数字 3（对应电压是 3.15 伏）？显然，这种解决方案在实际运用中会有很多的问题，基本上是不可行的。

为了寻求新的思路，我们反问一下：为什么必须划分成十个等级？原来是按照我们日常的记数习惯：我们有十个指头，用的是十进制，相应的数字是十个，当然要用十个不同的电压值来代表这十个不同的数字了。另一方面，一个电子器件的信号，最简单的就是“有”还是“没有”信号 （比如我们用的电灯开关），即低电压与高电压；那么，如果我们只用仅仅的这两个两种信号以某种方式来表示我们使用的任何数字，那么，问题就可以迎刃而解了！唯一的疑问就是，我们怎么用两种状态来表示我们常用的数字及信息？

早在十七世纪，伟大的哲学家、数学家、历史上少见的天才---莱布尼兹就设计了一套二进制记数系统（据说，中国的《易经》对莱布尼兹的二进制

系统构造有某些影响）：仅仅只用两个数字（0 与 1）来表示任何数及进行二进制的运算。

其实，我们中国人的祖先早就有了“二进制”思想，比如“天”与“地”、“阴”与“阳”；日常生活中也到处可见，电话流行前电报的“嘀嘀嗒嗒”、各种开关按钮的“开”与“关”；就连儿时打架时的口头禅：“服”还是“不服”！为了加深对二进制的理解，我们用一个数字扑克牌游戏来解释二进制记数原理。



上面有五张数字扑克牌，每张牌上面有十六个数，其大小从 1 到 31 不等。现在请你选一个数，不告诉我你选的数，只说哪张牌有这个数，我就立刻知道你选中的数。比如，你告诉我，说从右到左，第一、二、四张牌上有你选中的数，那我即刻知道，你选的是 11；如果你说选的数仅仅出现在第三、五张牌上，我会立刻算出你选的是 20。你可能会说，我的记忆力好，我记住了每张牌上的数字分布，所以能够快速地答对。真实情况是，我的记忆力很差，只记得每张牌上的第一个数（即 1、2、4、8、16），之所以我能够答对，因为我用了二进制进行非常非常简单的运算。

当你告诉我你选中的牌时，我将你的选择用一组二进制来编码。比如，你的第一次选择，我用 01011 （即扑克牌下方的红色字码）来代表你的选择（0 表示数不在牌上面，1 则正好相反）；然后将选中的牌上面的第一个数全部加起来，就是 8 + 2 + 1 = 11，这就是你选的数了。你的第二次选择，我相应地用 10100（即扑克牌下方的黑色字码）来表示，这样的话，16 + 4 = 20 就是答案了。

扑克牌上的第一个数字（即 1、2、4、8、16）称为二进制数的权值，它们与我们十进制权值概念相同，只不过大小不一样。确切的说，十进制数，由十个不同的数字组成（0、1、2、……、9），因为十进制的基数是 10；一个十进制数，其中每个数字都有一个权值，权值的大小取决于它所在的位子，这样我们用这个“位权”可以准确地规范数的大小。比如 8537，7 是个位数，其位权是 1（10^0）；3 是十位数，其位权是 10（10^1）；5 是百位数，其位权是 100 （10^2）；以此类推，8 的位权是 1000（10^3）。正因为引入了位权，我们又有：$8537 = 8 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 7 \times 10^0 = 8 \times 1000 + 5 \times 100 + 3 \times 10 + 7 \times 1$ 。

在二进制里，基数是 2；每位二进制数的位权也取决于它的位置，比如，上面五张扑克牌代表五位二进制数的位置，它们的权值是 2^4、 2^3、2^2、2^1、2^0。继续上面的例子，二进制数 $01011 = 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 0 + 8 + 0 + 2 + 1 = 11$；二进制数 $10100 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 16 + 0 + 4 + 0 + 0 = 20$。

你也许会问，上面都是二进制整数例子，如果是小数，那怎么办？我们说，一样的按照十进制的原理。一个十进制小数，比如 $6.75 = 6 \times 10^0 + 7 \times 10^{-1} + 5 \times 10^{-2}$；一个二进制小数 $11.101 = 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 2 + 1 + 0.5 + 0 + 0.125 = 3.625$。

通过上面的演示，我们知道了什么是二进制数；以及给出一个二进制数，找出其对应的十进制数字。计算机的运算用的是二进制数，而我们人类则用的是十进制，所以，在计算机做真正运算前，必须将十进制数变成二进制数。这种转换在数学上是一系列非常简单的减法及记数运算。我们用一个例子来说明转换过程， 为了简单起见，我们只考虑整数，比如 147。首先，我们定义一组二进制权值：1，2，4，8，16，32，64，128，256，……，用 B 代表 147 的二进制数；用这个数减去小于这个数的那个最大权值，即 128，我们有，147 － 128 = 19，并且更新 B 值：B = 1。然后重复下面的运算：将上面的差值（即 19）与下个权值（即上面左边的权值）相比较，如果这个差值大于这个新的权值，那么仿照上面做减法，并且在 B 的后面添加“1”；如果这个差值不够减，就在 B 的后面添“0”；……；这样循环下去直到最小的权值 1。回到上面的例子，19 比下面的权值 64 小，那么 B = 10；19 比下一个权值 32 小，则有 B = 100；进一步，19 对比 16 （比权值 16 大！），那么，19 － 16 = 3， B 更新为 1001；3 比 8（下一个权值）小，那么 B = 10010；继续比较，3 比 4 小，B = 100100；然后，3 － 2

= 1，B = 1001001；最后一步：1 - 1 = 0；B = 10010011。这个 10010011 就是十进制数 147 的二进制编码。

其实，计算机在读取人们提供的数字及信息时，不会进行这样的数学运算---上面的演示，只是要说明要表示一个数（或某个信息），不管用二进制还是十进制，它们都是等价的，至于用什么方式，取决于具体情况。对于人类来说，习惯了十进制，所以用十进制最方便；对于计算机，因为电子元件的特征，用二进制进行，可行性更高更准确。

在计算机用二进制进行数学运算、信息处理之前，必须将人们常用的十进制数字及构成人类语言的基本字母或者“字”转换成有“0”及“1”组成的二进制信息。自从上个世纪六十年代美国电气和电子工程师协会推出一个称作 ASCII （American Standard Code for Information Interchange, 美国信息交换标准代码）的标准编码后，计算机业界都迅速采用这套编码来进行计算机信息处理。比如，日常数字“2”的 ASCII 编码是“00110010”，英文字母“B”则对应为“01000010”。这样的话，我们不用运算，就可以直接得到对应于十进制数字的二进制数码。以“2”为例，其 ASCII 编码的右边四个位码就是其二进制数（即 0010）。

现在知道了计算机是用二进制来表示数字及信息的，我们终于可以回答计算机是怎样进行计算的了。首先数学规律是普遍有效的，二进制的加减乘除运算与上面介绍的十进制运算规则完全一样，并且，因为二进制只有两个数码符号（0、1），它的运算更加简单。

首先来看一下加法。十进制是逢十进一；二进制就是逢二进一。比如做五位数的加法，01011 + 00111= 10010 （对应的十进制就是 11 + 7 = 18）。这里，有一个重要的问题，就是上面在讲十进制加法时提到的表示数的“格子”的多少。一个五位二进制数，最大的数是 11111（即十进制的 1 + 2 + 4 + 8 + 16 = 31）。如果两个数相加的和超过这个数，结果就不正确了，这就是常说的“溢出”。为了解决问题，我们可以规定，增加一个格子来表示运算的数，就不会有问题了。现代计算机，一般的基本运算位数为 64 位 （即通常说的 64 位计算机），表达的整数范围是 $[-2^{63}, 2^{63}-1]$；如果引入小数点，附加相应的规则，表示的数的范围会大许多，就可以满足一般的科学及工程计算需要。

再看减法。我们借用上面介绍的“补码”概念，将减法变成加法。在二进制里，一个数的补码的“计算”非常简单：求反加 1。用上面的例子，10010 - 01011，减数 01011 的补码是 10100 （01011 的求反） + 1 =10101。然后将这个补码与被减数相加，10010 + 10101 = 00111 （即 18 - 11 = 7） 。注意，在这个加法中，最高位产生进位，形成了一个溢出（因为我们只有五个格子来表示每个数，所以，这个最高位产生的进位无法记录）。在这个减法中，这样的溢出，不会对结果产生影响。

至于乘法，我们运用上面十进制乘法同样的规则，将二进制乘法变成一系列的移位及加法；除法，则是一系列的减法，而减法又可以变成加法。总之，计算机里的加减乘除，都可以归结到加法（另加移位）；这样的话，在设计计算机运算电路时，我们可以大大的简化运算电路的硬件实现的复杂程度。不过，要说明的是，在某些学科研究及工程项目中，有非常大量的乘除运算，有必要设计专门的乘除运算电路。在下面的例子里，我们仅展示计算机的加法“电路”的概貌。

五、计算机芯片的核心部件---加法器

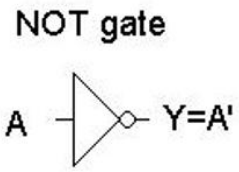
上一节介绍了计算机利用二进制进行计算的基本原理及方法，同时也揭示了可以利用某种电子器件的高低电压信号来表示二进制的两个数码（即 0、1）。将一个电子元件的输入输出信号划分成“高”、“低”两种状态，很容易实现；但是，要将这个划分用于计算机核心部件（及上面所说的 CPU 中央处理器）的制作来进行二进制运算，这种电子元件还必须满足某种“开关”特性：在某种情况下，“低”电压输入产生“高”电压的稳定输出；反之亦然。终于，在二十世纪上半叶，具有这个特性的电子管（也叫真空管）问世并开始用于工业界。经过科学家的努力，我们就有了第一代的电子计算机，即电子管计算机。二十世纪中叶，科学家发明了有同样“开关”特性的晶体管（三极管）；这个新元件比电子管体积更小、能耗更低；它立刻催生了第二代电子计算机，即晶体管计算机。以后，随着电路板上的集成度（可以大致认为单位面积上的“晶体管”的数量）越来越高，集成电路、大规模集成电路、超大规模集成电路的出现，第三代、第四代计算机就应运而生了。

我们所说的芯片，就是一种超大规模的集成电路：在一寸见方的芯片上，可以集成几百万或更多的晶体管。芯片的最重要的应用之一，就是用于

计算机 CPU 中央处理器及 RAM 内存的制作。CPU 最重要的功能之一，就是进行加法运算。下面我们通过一个简单的例子，来演示“半加器”（两个一位二进制数的加法运算器）的设计过程。

首先，电子管或晶体管这样的原始“开关”元件必须通过某种组合（比如多个晶体管，外加电阻）形成一些具有特定功能的基本元件；然后，再用这些基本元件来构造具有运算功能的某种运算器。这里，我们不考虑怎么制作这些基本元件的细节、它们的电子特性，仅仅介绍它们的功能。为了简单起见，我们借助通用的符号来标识这些基本电子元件。在计算机学科术语里，这些基本元件被称作逻辑“门”（Logic Gate），元件的功能用一个表格来描述它的定义（就像我们中学里用某个函数来定义某些输入输出特性）；这个表格被称作**真值表**（Truth-Table），它的输入输出值不是“1”就是“0”，也称作“真”或“假”，或用英文字母表示，就是“T”（True， 真）或“F”（False， 假），正好与二进制的两个数码相对应。

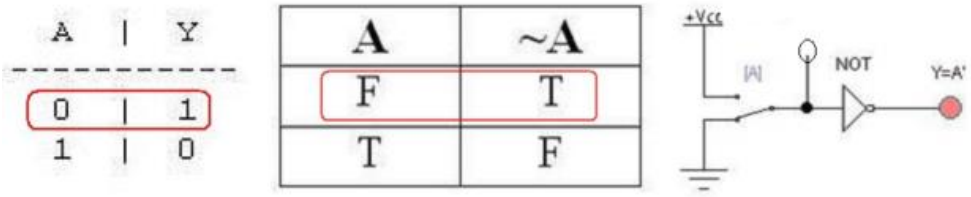
我们的第一个逻辑门叫“非门”（NOT gate），它只有一个输入、一个输出；我们用“～”或 “ ’ ”（单引号）来表示“非”操作符号（即非门的“函数”符号）。它的逻辑涵义就是“否定”，其逻辑运算非常简单：如果输入是 0 （假），那么输出就是 1 （真）；如果输入是 1 （T、真），输出就是 0 （F、假）。非门一般用一个顶端带个小圆圈的三角形来代表。下图给出了非门的图示及它的真值表(右边的两个表格，表示方式不一样的真值表）：A 表示输入，Y 是输出；真值表的左边是输入，右边是输出。



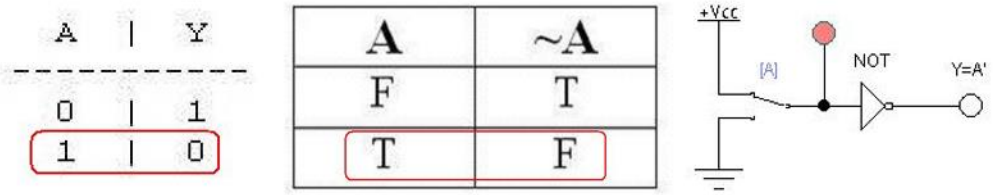

```
A | Y
----------
0 | 1
1 | 0
```

A	~A
F	T
T	F

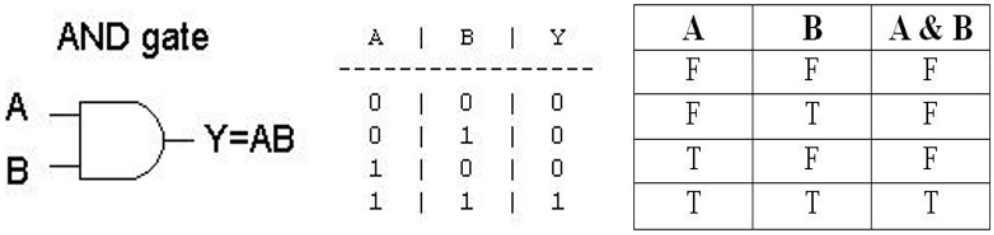
作为一个实验，我们可以设计一个简单电路来测试非门这个基本逻辑元件，同时加深我们对真值表的理解。在下面的测试电路里，Vcc 是输入电源；三个横向上下排列的短线表示接地线；两个稍大的圆圈表示电灯泡（通电或者高电压时就变亮变成红色）。首先测试真值表的第一行，当输入接地（或低电压），即输入为 0、为假（F）；非门的输出为 1、为真（T），灯泡变亮。



真值表的第二行，即输入为 1（为真、T、高电压），非门的输出就变成了 0（为假、F、低电压），灯泡不亮。下面的图示给出了这个测试结果。



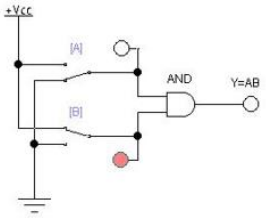
接下来，我们介绍逻辑“与门”（AND gate）。最基本的与门有两个输入（这里用 A、B 来表示），一个输出（用 Y 表示），相当于二元逻辑的合取运算；与门的“函数”符号用“&”表示（有时像代数表示乘积那样，不用任何符号，仅仅将 AB 写在一起表示二者的“与”操作）。与门“函数”规定当且仅当两个输入同时为 1 时，它的输出才是 1；对于输入的任何其它组合，其输出都是 0（对应于“乘法”结果）。下面的图示给出了与门符号及其真值表的定义。



从上面的真值表可以看出，与门的两个输入，可以组合成四种不同的输入信号，即 00、01、10、11，它们对应的输出为 0、0、0、1。我们选择真值表的第二行及第四行来进行测试。先看第二行，A 端口输入为 0（接地低电位）、B 端口输入为 1（高电位），根据与门“函数”的定义，其对应输出为 0，见下图。

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

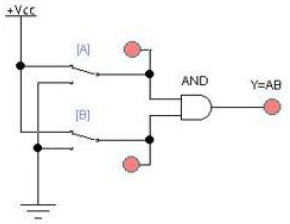
A	B	A & B
F	F	F
F	T	F
T	F	F
T	T	T



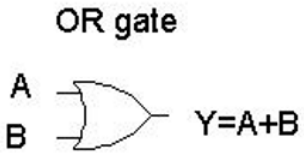
当 A、B 两个输入端口都是 1（高电位）时，即真值表中的第四行，与门的输出为 1（高电位），测试电路见下图。

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A & B
F	F	F
F	T	F
T	F	F
T	T	T



现在，我们介绍制作半加器需要的最后一个逻辑门----“或门”（OR gate）。最简单的或门有两个输入端口、一个输出端口，其“函数”符号用“+”或“|”来表示。当任何一个输入端是 1（高电位）时，或门的输出端就是 1；仅仅当两个输入都为 0 时，它的输出才是 0。这里要注意的是，我们现在谈的是逻辑运算，请不要与算术运算相混淆，尽管二者有点类似。特别是，当两个输入都是 1 时，输出是 1（逻辑“真”）而不是 0（逻辑“假”）。我们可以用一个日常生活的例子来加深我们对“或门”的理解：如果你说“我今天抽签中奖，我就请客吃饭；或者，公司发给我额外奖金，我就请客吃饭。”那么，假设你今天抽签中奖了并且公司又发给了你额外奖，你肯定得请客！与“非门”及“与门”一样，我们用真值表来定义或门“函数”。下图给出了代表或门的符号及其真值表。



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A | B
F	F	F
F	T	T
T	F	T
T	T	T

需要说明的是，这里仅仅介绍只有两个输入的“与门”及“或门”。实际上，在应用中，会有很多输入端口的“与门”及“或门”，它们都具有与

其对应的两个输入端口的逻辑门相同的性质，那就是，对 N 位输入的“与门”，只有当所有这 N 个输入都是 1（高电位）时，输出才是 1；对 N 位“或门”，任何一个输入是 1，则其输出就是 1。

有了上面的准备工作，现在我们终于可设计制作计算机 CPU（中央处理器）中最基本的运算部件了，那就是“半加器”---两个一位二进制数的加法运算器。这里说的半加器，就像做十进制加法时，仅仅考虑两个数的个位数相加，比如在 835 + 189 里面，仅仅考虑个位数 5 + 9 的运算。如果是其它位上的数字相加（比如十位、百位上的两个数字），则必须考虑低位数相加后产生的进位，比如上面的例子，十位上的两个数相加（3 + 8）时，必须将个位数的进位加进去，这些数位的加法器，叫做“全加器”。二进制的加法也是类似。有了半加器，就可以非常方便的制作全加器。

在半加器里，仅仅有两个输入端口（即两个一位二进制数），表示加数及被加数，而输出则有两个端口：一个我们称为“和”，另一个我们称为“进位”。我们现在用 A、B 表示这两个二进制数输入，用 Sum 表示这两个数相加后的“和”、Carry 表示相应的“进位”。A、B 输入有四种不同的组合，即 00、01、10、11。A 与 B 相加后的和及进位，可以用下面的真值表（也称作半加器真值表）来表示。

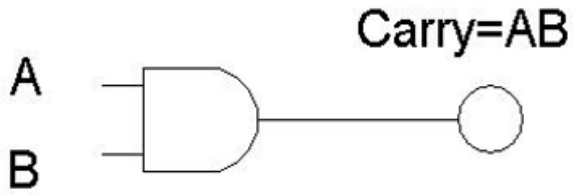
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

我们现在的任务就是要设计一个由上面介绍的“与门”、“或门”、及“非门”组成的一组电路来完成“半加器”规定的加法运算；在这个电路中，有两个输入（A、B），有两个输出（和 Sum、进位 Carry）。

先从进位开始。从上面的半加器真值表中，将表示“和”（Sum）的那一列划去， 我们有下面的“进位”真值表：

A	B	Carry	Sum
0	0	0	0
0	1	0	
1	0	0	1
1	1	1	0

仔细考察这个“进位”真值表：当且仅当两个输入为 1 时，进位（Carry）才是 1---好一个似曾相识的“与门”归来的感觉！没错，我们可以用一个“与门”来实现进位的逻辑电路，其逻辑表达式就是 AB，图示如下：

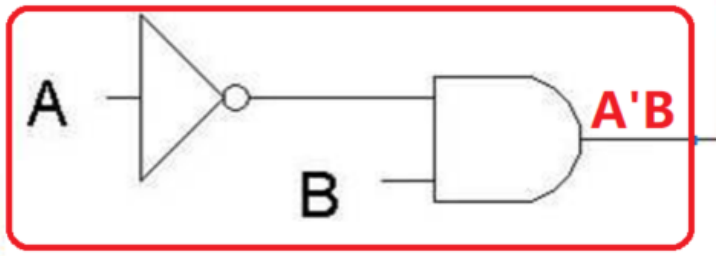


下一步，我们来设计“和”运算的逻辑电路。从半加器真值表中将“进位”（Carry）那一列划去，有下面的“和”（Sum）的真值表：

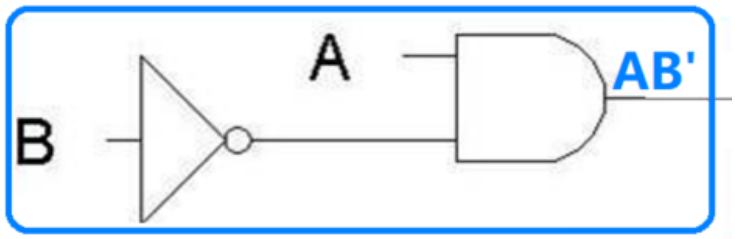
A	B	Carry	Sum
0	0	0	0
0	1		1
1	0	0	1
1	1	1	0

在这个“和”真值表中，我们只关注“和”值为 1 的对应的 A、B 输入行，即图中的第二行（用红色线框描出）及第三行（蓝色线框标记）。红色线框，对应的 A、B 输入是当且仅当 A 是 0 并且 B 是 1 时。这个情况下，A 是 0，我们对 A 进行“非门”操作，其结果就是 A’（或者～A）为 1。那么，上面的表述就转换成：“对应的 A、B 输入是当且仅当 A’ 是 1 并且 B 是 1 时”--- 又一个似曾相识的“与门”归来的感觉！不过这里的与门的输入是

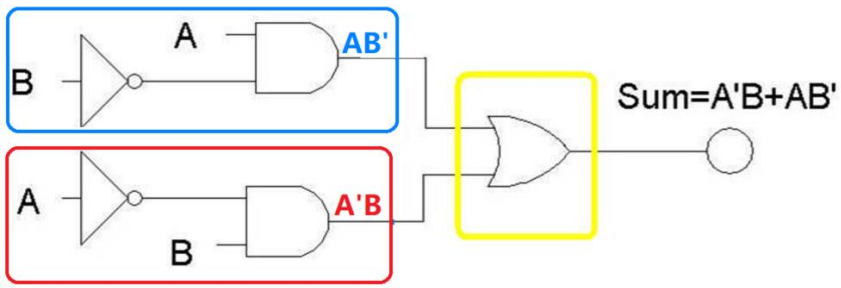
A’（及 A 的“非”运算的结果）及 B，逻辑表达式就是 A’B。那么我们有如下的电路实现这个红色线框表达的“和”：



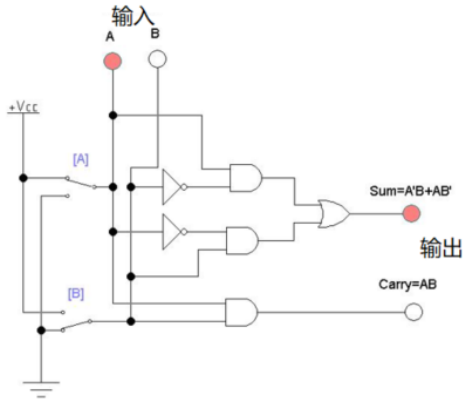
对于蓝色线框里的“和”情形，我们施以相同的方法，可以得到如下的逻辑电路（相应的逻辑表达式是 AB’）：



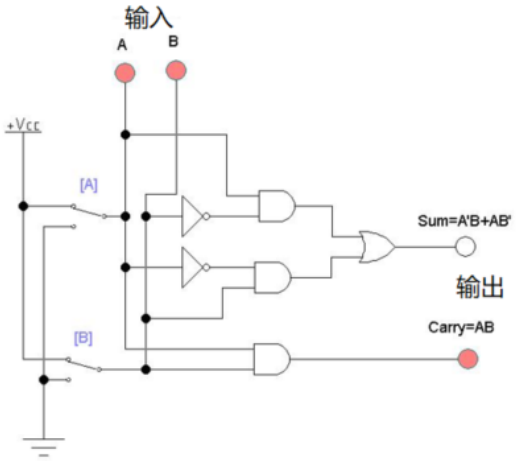
现在将红、蓝色电路合在一起，就回到了什么“如果抽签中奖或者得到额外奖金，就请客吃饭”相同情景，即“红色线框”或者“蓝色线框”都产生“和”；我们用一个“或门”来涵盖这两种情况。那么，我们就有“和”的逻辑表达式：Sum = A’B + AB’（这里的“+”是逻辑“或”运算符），其逻辑电路见下面。



我们来验证我们设计的半加器。当 A 的输入为 1、B 的输入为 0 时，那么，二者相加的“和”（Sum）是 1，“进位”（Carry）是 0.下面的电路测试结果符合预期。



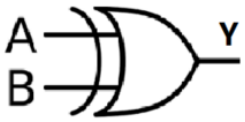
再测试 A、B 的输入都是 1 时，就是二进制的 1 加 1，根据二进制的加法逢二进一的原理，结果应该是 10（即进位数 Carry 是 1，和位数 Sum 为 0），下面的电路测试图也验证了我们的半加器的正确运算结果。



因为半加器仅仅只能做个位数的加法，显然它不能用于真正的计算。参与运算的数，一般都是多位数；除了个位数外，其它数位的加法，都必须加上低位数相加后产生的进位：为了不失一般性，在没有进位 （比如 1 加 0）的情况下，我们认为有进位，不过这个进位是 0。这样的加法就只能靠全加器了。在二进制的全加器（Full Adder）里，输入是三个二进制数（两个参与相加的一位二进制数及来自低数位相加后产生的进位）；而输出只是两个

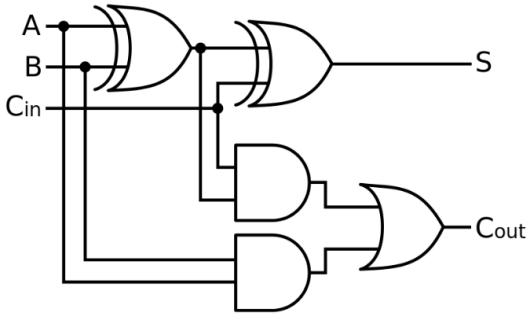
二进制数（进位及和），因为三个最大二进制数相加（即三个 1 相加），结果是 11，所以两个输出位就足够了。

为了精简二进制全加器的设计，我们引入另一个基本逻辑门，叫做“异或门”（XOR gate，XOR 是英文 eXclusive OR 的简缩写）。异或门通用代表符合及其逻辑运算真值表见下面图示。



输　入		输出
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

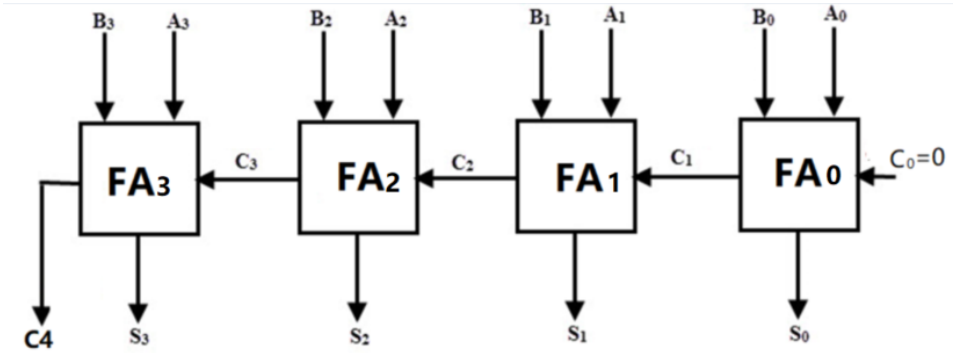
依照我们上面半加器的设计思路及流程，我们可以很方便地得出二进制全加器的逻辑电路（见下图）。在这个全加器里，三个输入是两个加数（A、B）及 Cin（低位数进位），输出是 Cout（相加产生的进位）及 S（和）。



现在，万事俱备，我们可以制作任何位数的加法器了。用一个四位二进制数加法器来说明。仿照十进制运算的习惯，我们将个位数放在最右边，数的最高位在最左边；这样两个四位二进制数可以写成 $A_3A_2A_1A_0$、$B_3B_2B_1B_0$。我们用四个全加器（FA_3、FA_2、FA_1、FA_0）来实现这个四位数的加法，其中个数上的全加器（FA_0）的进位输入是 0（没有进位），低数位的数相加后产生的进位作为左边加法器（即更高数位加法器）的进位输入。例如，假设 $A_3A_2A_1A_0$=0101、$B_3B_2B_1B_0$=1101，那么，在加法器 FA_0 上，A_0 与 B_0 相加（即 1 加 1，此处输入进位 C_0 为 0），其结果的和 S_0 为 0，相加产生的进位为 1，这个进位作为左边加法器（FA_1）的输入，即 C_1=1。在加法器 FA_1 上，三个输入 A_1、B_1、C_1（0、0、1）相加，其结果的和 S_1 为 1，相加产生的进位为 0，这个

进位作为左边加法器（FA_2）的输入，即 C_2=0。依此类推，这两个二进制数相加的结果是 $C_4S_3S_2S_1S_0$ = 10010，加法器 FA_3 产生的进位 C_4=1。在这个例子里，两个四位二进制的数相加，结果是一个五位数。

下面图示非常清楚地展示了四位二进制加法器的工作原理。



4位加法器

依照相同的原理，我们可以设计制作任何 N 位数的加法器来运算任何两个 N 位数的加法。并且，根据上面介绍的计算机计算原理，有了加法器，我们可以做任何加减乘除及其它算术运算了。当然真正能够在实际生活中运行的计算器的设计，比上面的会复杂千万倍。上面的介绍，仅仅是蜻蜓点水，用来展示一下计算机计算器的最底层、最基本的逻辑电路的设计思想及电路实现，让我们消除计算机怎么计算的神秘感。

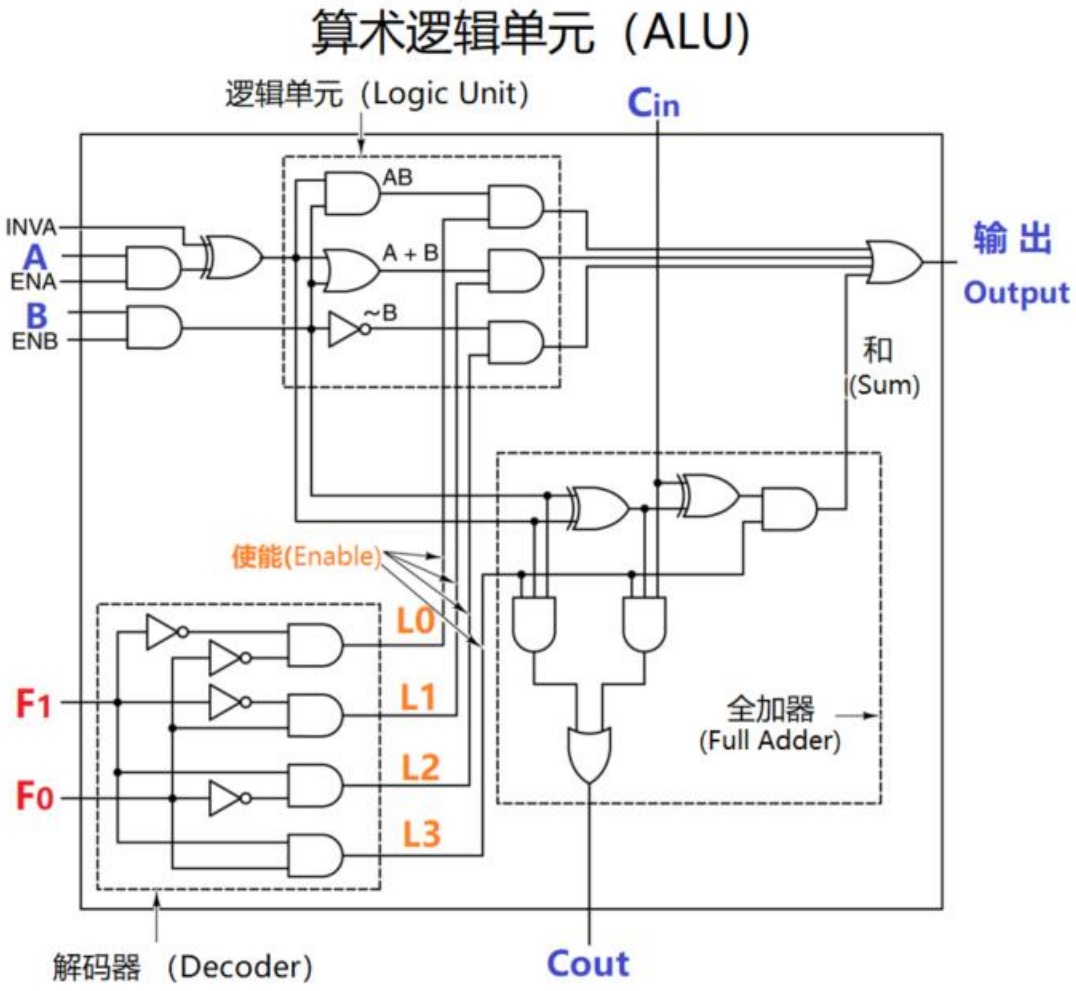
五、芯片啊，芯片

从上面计算机体系结构的介绍里，我们知道了在计算机所有部件中，最重要的有两个，就是 CPU 中央处理器及 RAM 内存，也就是人人挂在口上的芯片：RAM 存放所有信息，包括数据及计算机运算的所有程序指令；CPU 将这些数据、指令从 RAM 中读进来然后解释、执行这些命令，然后将结果存回 RAM。与 RAM 相比，CPU 芯片更加重要、更加复杂。

如果将 CPU 中央处理器按照其功能划分，可以认为它由三个部分组成：算术逻辑单元（Arithmetic Logic Unit，简称 ALU）、控制单元（Control Unit,简称 CU）、及寄存器（Registers）。寄存器是 CPU 内的高速存储器，

用于临时存放用于计算或要处理的信息（即一组二进制代码），其数量不等，十几个到几百个不等。ALU，如其名字所言，就是进行逻辑及算术运算的；控制单元 CU 掌控信息数码在内存及寄存器上的读取存放，驾驭信息处理、算术运算的具体操作。在这三个部分中，逻辑算术单元 ALU 凸显 CPU 中央处理器的功能，寄存器最简单而控制单元最为复杂。下面我们简要介绍 ALU 逻辑算术单元的工作原理。

从 ALU 的名字就可以知道，它进行的是逻辑运算（“与”、“或”、“非”等等）及算术运算（加减乘除等等）。ALU 功能可以用下面一个简单的逻辑电路图来描述，我们以此管窥计算机系统中 CPU 中央处理器的复杂程度。



（示意图取自于 Andrew Tanenbaum 的《结构化的计算机组织》（Structured Computer Organization）一书，略加编注而成。）

这是一个两个一位二进制数码的算术逻辑运算单元（ALU））示意图。这个 ALU 由三部分组成（见图中虚线方框）：逻辑单元，全加器，及解码器。逻辑单元部分，就是我们介绍的三个逻辑运算：“与”、“或”、“非”；只不过，每个逻辑门的输出都接入一个“与门”，而这个“与门”的另外一个输入则来自某个“使能”（Enable）控制信号。根据“与门”的特性，如果这个“使能”控制信号是 0，则相应的逻辑门输出被屏蔽。这三个逻辑门的输出经过三个控制“与门”接入一个有四个输入端口的或门，然后输出。要重复强调一下，通过**“使能”控制线**，我们可以在任何时刻，保证只有仅仅一个正确输出，因为来自其它的输出都被控制线屏蔽了。

全加器与我们上面介绍的一模一样，只不过原来“进位”电路在的两个输入端口的与门变成了三个输入端口的与门，这增加的第三个输入来自某个“使能”控制；“和”电路的输出增加了一个“与门”来控制其输出。在这里，相同的“使能”线控制全加器的“进位”及“和”的运算。

示意图上的蓝色标识是三个输入端口（两个数码 A、B，及来自低位的进位 Cin）和两个输出端口（“输出 Output”及相加产生的进位 Cout）。那些黑色标记输入线（INVA、ENA、ENB）是用来控制 A、B 的，这里不作探讨。要特别注意的是，输入信号 A 及 B 连接到 ALU 的不同部件，即某些不同功能的部件共享这两个输入；而标记为“输出”（Output）的单个输出端口整合了四个不同元件的输出（三个逻辑运算的结果及一个算术相加运算的结果“和 Sum”）。一个显而易见的问题就是，怎么区分这个单一的“输出”信号到底代表哪一个元件的输出？仔细一看，就发现起“输出”整合作用的是一个“或门”，根据“或门”的特性，只要有一个输入有信号，其输出就会有信号。而所有这四个输入来自四个不同的逻辑操作电路的输出，正好有四根“使能”控制线，分别控制那四个逻辑运算电路；只要保证这四根“使能”控制线在任何一个时刻仅仅只有一个为高电位，那么，那个共享的“输出”就会正确地输出相对应的运算结果。这个保证就靠“**解码器**”了。

图中的解码器，是这个 ALU 的看点！红色标记的 F1、F0 是解码器的输入；橙色标记的 L0、L1、L2、L3、L4 是解码器的输出---“使能”控制线；其中的三根“使能”线（L0、L1、L2）“选通”及“屏蔽”逻辑单元的输出；另外一根（L3）控制全加器。

现在，我们仔细考察一下这个解码器电路。根据逻辑“非门”及“与门”的特点，就知道不管 F1、F0 的什么输入组合（即可能的四种之一：00、01、10、11），解码器的输出中（L0、L1、L2、L3）中必须有一个且仅仅只有一个为 1；这个唯一的为 1 的“使能选通线”**开通**了逻辑单元-全加器的某个输出，同时，其它三根为 0 的“使能线”正好“**断开**”了其它的输出！例如，当 F1、F0 为 01 时，L1 变成 1，L0、L2、L3 都是 0，这个是 1 的 L1“使能控制线”开通了逻辑单元在“或门”的输出（即 A+B，这里的“+”是逻辑“或”运算）：这个“或门”的输出与 L1 作为一个控制与门的输入，现在 L1 为 1，那么根据与门的性质，这个与门的输出完全取决于逻辑“或门”的输出结果了（即 A+B），所以我们说，这个 L1“使能线”选通了那个逻辑运算“或门”的输出。同样的，L0、L2、L3 都是 0，将所有与其他逻辑运算器（“与门”、“非门”）及全加器的输出相连的三个“与门”的输出全部变成了 0，翻译成大众语言，就是说，将这三个运算器的输出全部“断开”了---因为，即使这些运算器有输出 1，但是与之相连的与门的输出已经被那三根为 0 的“使能线”给变成 0 了。

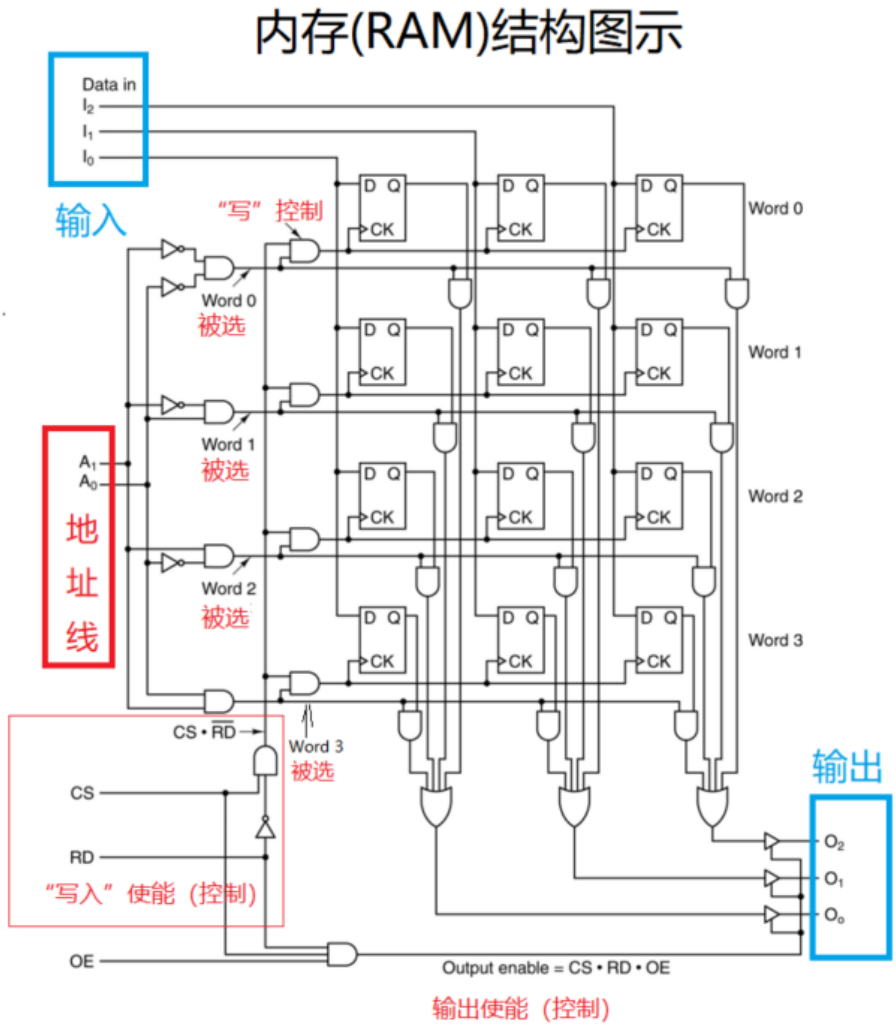
经过上面的解释，我们知道这个解码器掌控了 ALU 的输出---间接地操纵算术逻辑运算，它在 CPU 中央处理器中扮演了极其重要的角色，所以我们用下面的一个表格来总结这个解码器的输入码与 ALU 运算输出的关系。

F1 F0	L3 L2 L1 L0	“输出”结果
0 0	0 0 0 1	逻辑“与”
0 1	0 0 1 0	逻辑“或”
1 0	0 1 0 0	逻辑“非”
1 1	1 0 0 0	全加器“和”

上面我们用一个示意图例子，简单地介绍了 CPU 中央处理器的最基本最原始的功能。真正实用的 CPU 的内部结构，比这个图示的复杂程度高出千万倍。尽管这个粗略的 CPU 图解对我们理解计算机的工作原理有帮助，然而，从这个图示里，我们不能直接体会到芯片的复杂性。下面，我们演示一下比 CPU 更简单的 RAM 内存的结构，来亲身感受芯片的复杂性。

计算机的内存(RAM)是存储芯片，用来存放 CPU 要处理计算的所有信息数据---它通过总线 Bus 与 CPU 及输入输出设备相连接（见上面“计算机体系结构”一节的介绍），是仅仅次于 CPU 的重要芯片。在耗电、散热、稳定性

能方面相同的话，内存片越小、能够存储的数据越多，那么这样的内存片肯定更好。RAM 内存结构比 CPU 的简单很多，因为只要求它保存信息、随时可以对它进行读出、写入操作。我们用下面一个简单的内存结构示意图来了解它的工作原理。



(示意图取自于 Andrew Tanenbaum 的《结构化的计算机组织》（Structured Computer Organization）一书，略加编注而成。)

正如这个内存示意图显示的那样，可以将 RAM 内存片看成一排接一排的信箱，它们排列的非常整齐，每个信箱里只能存储一个二进制数（0 或 1）。这个存储信箱就是图中的那个有标有三个接口（即 D、Q、及 CK）的小方框。这个小方框代表的也是一个由晶体三极管组成的基本逻辑元件，学名叫触发器（Flip-flop）。这个图示中的触发器是很多种类中的一种，称为 D 触发器。在这种触发器里，端口 D 是输入接口，Q 是输出，CK 用于控制更新触发器存储的信息（即一个一位二进制数,0 或 1）：如果 CK 是 1，那么输入

信号（0 或 1）被写入触发器并被保存；如果 CK 是 0，触发器则被锁定，其存储内容不变，也就是不管输入端口的信号 D 是 0 还是 1，触发器的存储的信息不受影响。就是触发器的这个“**锁定**”特性使得它广泛用于存储器。

这个内存图里，有四排信箱（触发器），每一排称作一个“字”（Word），一般来说，“字”的大小（即一排触发器可以存储多少位的 0 或 1）取决于整个计算机系统，特别是 CPU 的规格，一般来说，“字”的大小以 8 位、16 位、32 位、64 位等计量。这里为了简单起见，是 3 位。这也说明了，这个存储片可以每次存取三位的二进制数，所以，这个内存有三个数据输入端口（I_2、I_1、I_0）、三个数据输出端口（Q_2、Q_1、Q_0）），见图中天蓝色标记的方框。

红色方框标记的是地址选择线(包括地址线 A_1、A_0及对应的控制选线)。我们这里有四排触发器，其存放的是二进制数码；二进制的基数（底数）是 2，并且 $4 = 2^2$（这个算式里的指数 2 对应的就是地址线的多少），所以我们需要两根地址线来选定四排触发器中的哪一排。假设我们有 16 排信箱，我们必须要有 4 根地址线来确定选择哪一排，因为 $16 = 2^4$。

这个地址选择电路，与 CPU 里的解码器一样：不管地址线 A_1A_0是四种信号（00、01、10、11）的哪一种，在四个输出线中，有一个且仅仅一个输出为 1，其它三个全是 0；这个为 1 的控制线正好连接到对应的那一排触发器。例如，当 A_1A_0是 00 时，图中标记为“Word0 被选”那根输出线的信号为 1，这个信号同时用作触发器的“写”与“读”的控制：（1）它成为标记为“写控制”的与门的一个输入，如果来自图左下方的“写入”使能控制信号是 1，那么，这个与门输出是 1，这个 1 信号就“开通”了第一排触发器的 CK 端口，正好启动这些触发器的“写操作”---输入信号 $I_2I_1I_0$将被写入这一排触发器；（2）这个“Word0 被选”信号又与第一排三个触发器的输出端（即触发器的 Q 端口）接入相应的三个与门，而这三个与门的输出又通过相应的三个或门正好连接到图右下方的输出，最终到达蓝色方框的内存“输出”（即 $Q_2Q_1Q_0$）；这就是 RAM 的“读”控制（即 RAM 这个时刻的输出来自 Word0）。

以此类推，当 A_1A_0是 01、10、11 时，我们可以对 Word1（第二排触发器）、Word2（第三排触发器）、Word3（第四批触发器）来进行“写入”、“读出”控制操作。

这样的话，通过地址线的不同输入，可以产生不同的“被选”控制信号（即“Word0 被选”、“Word1 被选”、“Word2 被选”、“Word3 被选”），也就是“写入”使能、“读出”使能控制线，从而我们可以准确地控制所有触发器的读、写操作（以一排三个触发器为单位）。这也就是一般计算机 RAM 内存的存储、读、写操作的工作原理及操作过程。

制作 RAM 内存芯片的最基本“元件”就是上面介绍的逻辑元件：与门、或门、非门、异或门、触发器等；而这些逻辑元件又是由上面提到的晶体管（三级管）组成。我们现在来估算一下，一个常用的计算机或手机的存储芯片最少需要多少三极管。

上面的内存示意图有 3x4 = 12 个触发器，另外加上至少相同数量的其它逻辑元件；再假设每个触发器或其它逻辑元仅仅需要两个三极管，这样，上面的“内存芯片”需要 24x2 = 48 个三极管。这样的话，存储一个一位二进制数需要 4（48÷12 = 4）个晶体管。

对于一个当代通常流行的智能手机，8G 内存算是一个普通的配置。这个 8G，指的是 8G 个“字”，而每个“字”相当于一个 8 位的二进制数码，“G”的大小定义为 2^{30}。这样，8G = 8 x 8 x 2^{30} = 2^3 x 2^3 x 2^{30} = 2^{36} = 68 719 476 736。也就是说，一个 8G 的内存片，可以存储一个 687 亿位的二进制数！如果要计算制作这个芯片所需要的三极管，那么，这个 687 亿天文数字还要乘以 4。这两个数的乘积，在下不敢写出来，要不然，您看了可能会眼花。

如果您觉得毕竟手机还是有点大，可以将里面的存储芯片做的大一点。也许，我们大多数人没有实打实地见过存储芯片；但是，大家应该都知道很多手机可以配置额外的存储卡，一个拇指甲大小的小卡片，它的存储量是 32G、64G、或更高。想象一下，在一个拇指甲大小的卡片上，要植入上面那个未写出来的天文数字再乘以 4、8、或更大的一个数的数量的三极管！

这还不是尽头。相比排列布局这些三极管，连接这些三极管、控制这些三极管，从而我们可以精准地读取每个触发器上的信息，更是难上加难！

还有，相比 RAM 存储芯片的单一功能，CPU 中央处理器芯片要进行众多的逻辑、算术运算，它的控制线路更加复杂……

写到这里，笔者的头都开始发晕了，只能昏沉沉地、重重地感叹一声：芯片啊，芯片！

六、电脑的语言

如果您向一百个人发问：“什么是语言？哪些关键字可以描述语言的定义？”您可能会收到一百个不太一样的回复，但是，大多数都会包含下面一些关键字：听、说、写、读、交流，……。并且，很多人都想当然地认为，语言是人类特有的，只要是个人，都天生具有语言能力。是的，聋哑人、盲人，他们也会用某种语言的特定方式互相沟通。即使是几个月大的婴儿，也能与母亲做某种形式的交流互动；还有，宠物爱好者肯定还会补充一下，说不仅动物之间存在特殊“语言”，并且宠物与主人间也可以互相“对话”！

语言一直是一些哲学家、语言学家、逻辑学家、计算机专家的研究课题。虽然每个学科专家给语言的定义不太一样，但是大都同意语言是一个交流工具，它有一套完备的符号系统、规范的语法定义及语义范畴。我们现在就用这几条标准来考察一下计算机，看看电脑有没有语言功能。

通过上面的介绍，我们知道计算机是通过大量的一系列逻辑门组成的；这些逻辑门的各个端口的信号表示为“有”与“无”，即 1 与 0。这些物理硬件的特性，决定了计算机使用二进制来进行算术运算及信息处理。一串串的 0 与 1 相接的二进制数串就是计算机的符号系统。

再回头看一下 CPU 中央处理器里的“解码器”。当解码器的两个控制端口（F1、F0）为 00 时，CPU 的输出结果是 CPU 输入信号 A、B 的逻辑“与”结果；当 F1、F0 为 11 是，CPU 变成了一个全加器，输出是二进制数 A、B 相加的“和”及“进位”。这 F1、F0 两个控制端口所组合的四种不同信号（00、01、10、11）清楚严格地定义了这个简单中央处理器的不同功能。换句话说，这四个不同的两位二进制数，**规定**了这个简单 CPU 的四种不同的运算；它们相当于这个 CPU 的四种不同“**指令**”，告诉 CPU 做什么具体操作。比如，我们先后发出“命令”：00、01、11，那么，我们的这个简单 CPU 就会严格按照这些命令一步一步地行事：第一步，输出 A 与 B 的逻辑“与”结果（命令 00）；第二步，输出 A 与 B 的逻辑“或”的结果（命令 01）；第三步，输出 A 与 B 相加的算术运算结果（命令 11）。就是说，我们根据实际需要，“告诉”我们的计算机做什么，而 CPU 就进行具体的操作；计算机

“听从”我们发出的指示，不折不扣地完成任务。这些 0 及 1 组成的命令，就像我们说的“开关”操作，直接作用在 CPU 硬件机器上，我们用一个专业术语，“**机器语言**”（Machine Language），来概括它们。这个简单的 CPU 总共有四个命令，我们称之为这部电脑的 CPU 的“**指令集**”。

当然，我们的 CPU 太简陋，只有仅仅四条指令，不能用于任何实际场合。上个世纪七十年代后期，因特尔公司（Intel）推出了 8086CPU 芯片。这个 CPU 拥有一百多条指令，这些指令与 CPU 内专用的存储器（寄存器）相组合，可以形成数量更多、功能更强大的指令集合。例如，指令 01000110（某个寄存器加 1）；01001001（某个寄存器减 1）；00010101（对某两个特定的寄存器做“带进位加法”）；……。我们可以用这些机器语言的指令来编写“程序”，也就是开发软件。

但是，除非是专攻某个 CPU 的技术人员，要记住这些机器语言的 0-1 指令字串及它们的确切定义与特定操作细节，来编写应用程序软件，即使对一般的计算机工作人员，都是一项具大的难以胜任的任务。所以，计算机专家就将这些 0-1 字串指令用缩减的英文单词或词组来替代。这些与机器语言指令一一对应的英文助记符组成的指令集成，我们称之为“**汇编语言**”（Assembly Language）。比如，上面的几条机器语言指令可以用下面的汇编语言语句来代表：INC SI(寄存器 SI 加 1)，DEC CX（寄存器 CX 减 1），ADC （带进位加法）。懂英文的计算机技术人员，如果花点时间学习一下 8086CPU 的体系结构，就可以编写程序了，这比用机器语言编程要简单方便很多。

机器语言与汇编语言的指令集取决于特定的 CPU 体系结构，不同的 CPU 就有不同的指令集。如果仅仅局限于用机器语言或汇编语言编写程序，那么软件开发的时间与成本将不可估量，因为不仅需要花费大量的时间来训练计算机技术人员来学习某个特定的 CPU 体系结构；并且，在某个计算机（CPU）上开发的软件无法直接移植在另一台有不同的 CPU 的计算机上来运行。

所以，早在上个世纪五十年代中后期，计算机**高级语言**就问世了（汇编语言及机器语言被称作为低级语言）。高级语言的有限“词汇”，大多数都是英文单词或英语单词的简写；用高级语言编写的程序经过特定的计算机程序进行自动“翻译”，可以变成不同 CPU 体系结构的机器语言程序，从而在各种不同的计算机上运行。这样，普通大众，经过短期的编程训练，就可以

写程序、开发软件。几十年来，流行的计算机高级语言有几十种，比如 Fortran，C/C++， Pascal， ……。下面是一个用 Java 语言编写的完整程序：首先提示输入两个整数，然后将它们相加，最后输出显示两数相加后的结果。

```
import java.util.Scanner;

public class Sum {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("请输入第一个整数: ");
        int a = sc.nextInt();

        System.out.println("请输入第二个整数:");
        int  b =  sc.nextInt();

        int sum = a + b;

        System.out.println("这两个数的和为: "+sum);
    }
}
```

各种计算机高级语言的问世及发展，极大地推动了计算机在各行各业中的广泛应用。越来越多的来自不同学科、不同专业的技术人员，经过短时期的自习或培训，就可以用高级语言编写软件，处理他们工作方面或日常生活中的事务；并且，这些软件在各种各样的计算机上都可以运行。现在，绝大多数计算机工作人员的工作都是与“软件”有关，用高级语言编写程序，根本不需要事先了解某个特定计算机 CPU 的特殊结构。这些软件，早就不局限在“计算”方面了；它们用在现实生活的各个方面。

七、电脑的逻辑推理

常识告诉我们，生物与非生物的最大区别就是生物可以进行新陈代谢（有序的化学反应）；动物学家和心理学家的研究还揭示，位居生物进化链高端的动物还有喜怒哀乐情感；而站在进化顶点的“人”，不仅具有丰富多彩的感情，而且还能进行创造性的、理性的思维活动。

哲学家们早就开始注意并研究人类的这种理性思维活动，提取出了一些思维活动必须遵守的逻辑演义规则。著名的三段论就是其中之一。试举一个

人人都熟知且自动遵守的三段论例子：如果我饿了，我就吃饭；现在我饿了；那么我吃饭。

人类的很多智力活动都自然地服从那些活动特有的规则。比如下棋，在某种局面下，走哪个棋子、走到哪个位置才是最佳的选择。说到下棋，大家的兴趣可能一下子就提上来了，因为现在大家都知道，对于绝大多数的各种棋类，人类棋手都下不过计算机电脑！特别是那些知道一点点围棋及懂一点计算机的朋友们都清楚，让电脑下围棋并战胜人类顶尖棋手，多少年了，都是计算机专家想要攀登的珠穆朗玛峰，因为围棋的计算复杂性已经从数学理论上被证明：不管用多快的计算机、也不管“算”多少年（甚至直到整个宇宙不存在了），计算机也不可能算出下一步走哪里是最佳的选择！可是，几年前，谷歌的围棋电脑“阿尔法狗”（AlphaGo）横空问世，让世界上几个最顶尖的围棋棋手泪洒棋盘、输的一点脾气也没有！

计算机在人类最为自豪的智力活动范围里将人类击败，让大多数人都觉得电脑真的是太深奥、太神秘。其实，计算机无外乎是遵从相同的一些逻辑规则，利用自己快速的计算能力及应用某些人类开发的专用软件模型（比如神经网络），从而战胜人类。下面，笔者就举一个简单的电脑逻辑推理的例子，让读者朋友一窥豹斑。

首先，在已经介绍的基本逻辑运算（“与”、“或”、“非”等）上面，我们引入一个非常重要的逻辑操作“函数”，叫做“蕴含”(Implication），就是上面三段论例子里提及的“如果……那么……。”它的逻辑操作符号用右箭头→或者→来表示。并且，为了简单起见，我们规定基本语句用英文字母表示，比如“我饿了”用 P 表示，“我吃饭”用 Q 表示，那么，这句话“如果我饿了，我就吃饭”就变成了 P→Q 逻辑表达式了。

与其它逻辑运算一样，我们用真值表来定义逻辑蕴涵函数。假设有语句 P、Q，则 P→Q 用下面的真值表来定义：

P	Q	P->Q
F	F	T
F	T	T
T	F	F
T	T	T

从这个真值表中可以看到，如果 P 为真（即“我饿了”）并且 Q 为真（即“我吃饭”），那么 P→Q 为真（即“如果我饿了，我就吃饭”这个复杂语句为真）。唯一让 P→Q 为假的情况发生在下面的情景下：我饿了（P 为真），却没有吃饭（Q 为假）。这样，也符合我们的常识理解（对其它两种情况，这里不作具体举例解释）。

那么，计算机怎么进行逻辑推理呢？我们用 Smullyan 在他的一本书《这本书的名字是什么？》(What is the Name of This Book?)中举的“爱情中的逻辑”例子，笔者用我们熟知的真值表，来展示电脑对这个问题解答的最基本的逻辑推理过程。

假设一个恋爱中的男孩说：“我或者喜欢贝蒂，或者喜欢珍妮；如果我喜欢贝蒂，那么我肯定喜欢珍妮。”那么，我们能不能肯定这个男孩到底百分之百喜欢谁？也许读者应用我们人类特有的逻辑思考能力，已经推断出来了：可以肯定的说，这个男孩喜欢珍妮；至于喜不喜欢贝蒂，则不敢肯定。我们现在来看看，计算机是怎么一步步地运用上面介绍的一些逻辑函数真值表来得出相同的结论的。

首先，我们用 B 代表“我喜欢贝蒂”，用 J 代表“我喜欢珍妮”，那么，男孩说的两句就可以表示为：B or J （or 是逻辑“或”函数）；B→J。这两个逻辑函数的输入是 B、J，并且，根据定义，我们将两个逻辑函数用真值表表示如下：

B	J	B OR J
F	F	F
F	T	T
T	F	T
T	T	T

B	J	B → J
F	F	T
F	T	T
T	F	F
T	T	T

因为两个函数有相同的输入，为了简单起见，我们将两个真值表合成一个，并且将函数真值表的输出列用红框标记。那么，我们有：

B	J	B OR J	B → J
F	F	F	T
F	T	T	T
T	F	T	F
T	T	T	T

对于男孩的爱情宣言，我们相信是他的真情表达；所以，对他说的两句话（即真值表中的两个输出列），我们在真值表中找出使这两句同时为真的情形，用红粗线框标出：

B	J	B OR J	B → J
F	F	F	T
F	T	T	T
T	F	T	F
T	T	T	T

真值表中用红粗线标记的两行，就是我们要考量的；对于其它的情形，不在我们要考虑的范围。这样的话，我们有：

B	J	B OR J	B → J
F	F	F	T
F	T	T	T
T	F	T	F
T	T	T	T

现在，我们的注意力转移到真值表的 B（“我喜欢贝蒂”）及 J（“我喜欢珍妮”）两列上面。将红框中 B 及 J 的逻辑值（即 T/真，F/假）用方框和圆圈标出来：

B	J	B OR J	B → J
F	F	F	T
F	T	T	T
T	F	T	F
T	T	T	T

这样，我们清楚地看到，在就这个情形给定的条件下，J 的逻辑值总是“真”（即“我喜欢珍妮”总是真），而 B 的逻辑值可以“真”可以“假”（即我们不能断定“我喜欢贝蒂”到底是真还是假）！这个计算机依照逻辑规定推断出的结论与我们人类思维推理结果完全一样。

当然，计算机棋牌、游戏、其它实际应用中遇到的情况、推理过程比上面的例子要复杂的多，肯定不能仅仅用真值表来运算求解；但是，这些基本逻辑函数及对应的真值表是电脑推理的最基本起点、奠基石；有了这些起点，我们可以运用计算机的高级语言编写程序来实现复杂的逻辑推理。我们日常生活中看到听到的计算机下棋玩游戏、人脸识别、语音辨识，都是在这些奠基石的基础上发展起来的，属于人工智能（Artificial Intelligence，简称 AI）领域及神经网络（Neural Network）的应用范围。如果读者有兴趣，可以在因特网上寻找感兴趣的读物及视频。

八、电脑带来的挑战：人就是机器？

从计算机诞生时日开始算起，到现在为止不过七、八十年；就在这短短的时间里，计算机这种机器就在很多专属于人类智力活动的领域内将人类打的一败涂地、狼狈不堪。计算机学科早已不局限于研发速度更高、体积更小、应用范围更广的电脑计算机了，它形成了一门新的科学---计算机科学，对数学、哲学、心理学、甚至宗教产生了重大的影响，甚至有些笃信上帝存在的虔诚的教徒因此改变了他们的观点。

不管每个人受到的教育程度的不同，也无论每个人的宗教背景的差异，人们都认同作为生物一种的人比所有其它生物的“智力”程度高很多。同时，我们也知道，计算机不过是一些机电设备、电子元件的组合体，它怎么会有“智力”？！

可是，我们闭着眼睛想一想、睁开眼睛看一看，就发现计算机电脑在很多方面比人“聪明”；并且，还不止一点点。比如上面提及的围棋例子，手机上的众多应用软件，甚至谷歌的汉英、英汉翻译比很多学过英语多年的大学生的翻译还地道许多。

当然，作为电脑的创造者、计算机的主人，居于生物进化最高点的我们肯定心有不甘，会找出一大堆的反例，来证明计算机电脑的智力比人的智力低很多。而那些理性的科学家就会问几个看似平常却意义深奥的问题：什么是智力？怎么考量智力？怎么比较人与计算机的智力？

早在 1950 年,伟大的计算机科学之父、数学家、逻辑学家、密码专家图灵（Alan Turing）就提出了著名的图灵测试。考虑到人们对智力的理解、对智力的定义会掺杂不同的主观成分，图灵设计了一个“思想实验”（Thought Experiment， 又译臆想实验或想像实验，就是由于现有实验条件不具备，只能在大脑里“做实验”）：这个实验中有两个人（测试者、被试者）及一台计算机（被试者），两个被试者在分开的封闭房间（测试者不知道哪个被试者在哪个房间），测试者与他们通过某种装置（比如键盘、打印机）交流；测试者向被试者提出任何问题、接受他们的回答；如果测试者不能区分那些回答到底是来自计算机被试者还是人类被试者，我们就说那台计算机通过了“测试”，那台计算机具有人的智力。另一方面，在回答问题的过程中，计算机被试者能够“愚弄”测试者的次数（即图灵测试通过率）的多少，也反映出计算机的“智力”水平的高低。

随着计算机性能的日益提升及其广泛应用、认知科学（Cognitive Science）的兴起，一些科学家开始认同“强人工智能”（Strong AI），认为机器可以具备甚至超跃人的智力。上个世纪末设立的每年一次的罗布纳奖（Loebner Prize），也给了人工智能爱好者以极大的鼓舞，给他们提供参赛机会，可以向计算机科学界展示他们开发的软件的图灵测试通过率。

囿于计算机的简短历史及当今的技术水平的限制，人工智能还处于襁褓期，相当多的哲学家科学家一方面首肯计算机的高速发展、对现实生活各个方面的巨大贡献，一方面认为人与机器之间存在某种“天然”鸿沟，大致认可“弱人工智能”（Weak AI）：计算机电脑可以具有人的某些智力或“心智”，或者即使我们在图灵测试中不能区分两个被试者（即使通过了图灵测试），但是人与计算机还是有实质上的差别。

哲学家希尔勒（John Searle）就是他们的代表，他提出的叫做“中文房间”（Chinese Room）的著名“思想实验”就形象生动地阐释了“弱人工智能”观点。汉语中文，对地地道道的欧美人士来说，像是天上的文字，不仅艰深还神秘莫测，所以希尔勒用中文作为一个例子来设计他的“思想实验”：一个测试者与一个被试者分处两个封闭的房间，房间之间仅仅有一个共用信箱，用来传递写在纸上的问题及答卷；测试者仅仅懂汉语，被试者不懂汉语但可以使用中文有关的翻译工具；测试者用中文提问题，被测试者借助翻译工具琢磨着用中文回答提问。假设被试者的回答全部准确无误，通过了测试，测试者以为被试者懂中文；但是，事实上，我们还是得承认，被测试者不懂中文。

这些哲学家科学家臆想的与现实不着边际的什么实验，还有那些看似简单而深藏玄机的专有词汇，对我们普通吃瓜大众来说，有点像雾里看花、不得要领。下面，笔者就将一些科普读物上的有关数据排列展开，让读者您用自己的大脑过滤一下、忘却那些让人云里雾里的专有词汇、展望预测计算机电脑的未来。

我们生活的地球形成于近 50 亿年前；细胞作为生物体的具有完整生命力的基本单位，出现在 20 亿年前左右；在进化链上，“人”与动物的分家，大概在 200 万年前。标识人与动物区别的最重要的特征就是人的大脑容量的急剧增加，准确的说，应该是脑神经细胞、也就是神经元（Neuron）的数量增加。一个人，如果四肢受损、五官受伤，这个人仍然有正常的情感、常人

的智力，可以融入社会生活；但是，如果这个人的大脑受到伤害，那肯定不能过正常人的生活了。所以，拿“人”与某个生物或机器比较智力，如果我们使用可以量化的大脑神经元的数量来作对比，肯定具有更强的说服力。

根据生物学家神经科学家的研究，人的大脑有大约 860 亿个神经元。我们再来看被认为是计算机“大脑”的 CPU 芯片，最新苹果笔记本上的 CPU 芯片（Apple M1 Max）有 570 亿个晶体管；用电子元件模拟人脑神经元的研究还处于实验阶段，据有关报道，大概可以用几百个晶体管来模拟一个神经元。为了方便计算，我们取整数 1000。这样的话，人脑神经元的数量比一般电脑芯片上的“神经元”高出 1500 倍。这说明人脑比现在的电脑聪明很多很多！

但是且慢，人脑经过了 200 万年的进化；计算机从诞生到现在，不过区区 75 年；还有，计算机芯片上信号传播速度比神经元“通讯”速度快百万倍；不仅如此，计算机芯片上晶体管数量的增加大致遵从著名的摩尔（Moore）定律：每隔 18 个月，晶体管的数量就增加一倍，并且性能也会提升一倍。按照这个速度，几十年后，芯片上的“神经元”的数量就会超过人脑中的神经元数量。

也许有的读者会指出，人脑神经元之间的“交流”不仅仅是依赖“数字式”信号，还有模拟信号的功能，这是现在数字式计算机无法比拟的。对，没错。但是，笔者也要提醒，现在物理学家、计算机科学家正在实验研究的量子计算机（Quantum Computer），与现在的建立在二进制基础上的计算机会完全不同。量子可以同时表现为多个状态（而晶体管在某个时刻只能是一种，不是 0 就是 1），量子叠加特性让量子计算机的“计算”性能提升百万倍以上（不是仅仅在速度上的增加）！

当然，还会有读者说，即使计算机比人聪明，毕竟计算机是人创造出来的，肯定会有办法控制计算机，人比计算机肯定要高一等。对此，笔者举出一个“思想实验”，让读者您绕过难以捉摸的智力比较问题、回答一个简单的问题---“人到底是不是机器？”

现在，科学家们对单个神经元的特性已经非常了解，也可以制造模拟单个神经元（我们且称之为人工神经元）。那么，神经科学家与外科学专家合作实验，将一个被试人的大脑里的神经元一根根地取出，代之以人工神经元。当替换一根神经元时，被试者自己及检测工作人员，感觉不到任何变

化，这个被试者还是“人”，这样继续往复；替换了 10%的神经元，如果没变化，被试者还是“人”；……；如果 100%的神经元都被替换了，被试者没有任何感觉思维上的变化，那么，这个被试者到底是“人”，还是一台“机器”？！

亲爱的读者，您的回答是什么呢？