



中央处理器 (CPU)

关注者

14,728

被浏览

5,124,051

知乎日报也关注了该问题

CPU 的工作原理是什么?

看到很多介绍 CPU 的介绍，很多是工艺流程的介绍，模糊知道有各种指令集，各种门进行着各种运算，但还是不知道这运算的物理原理。还有那些所谓的 01 又…显示全部 ▾

关注问题

写回答

邀请回答

好问题 335

16 条评论

分享

...

查看全部 158 个回答



码农的荒岛求生



《计算机底层的秘密》作者

专业 已有 1 人赠与了专业徽章 >

14,581 人赞同了该回答

想了解CPU的工作原理莫过于从头开始用最基础的元素打造一个简单CPU。

接下来我会从最简单的晶体管开始一步步讲解CPU是如何构造出来的，明白了这个过程理解CPU的工作原理不在话下，在此之后我会从最基础的二进制机器指令一步步讲解高级语言的基本原理，通读本文后你将彻底明白CPU与高级语言的工作原理。

以下内容出自我的两篇文章《你管这破玩意叫CPU?》《你管这破玩意叫编程语言?》。

每次回家开灯时你有没有想过，用你按的简单开关实际上能打造出复杂的CPU来，只不过需要的数量会比较多，也就几十亿个吧。

伟大的发明

过去200年人类最重要的发明是什么？蒸汽机？电灯？火箭？这些可能都不是，最重要的也许是这个小东西：



知乎 @码农的荒岛求生

这个小东西就叫晶体管，你可能会问，晶体管有什么用呢？

实际上晶体管的功能简单到不能再简单，给一端通上电，那么电流可以从另外两端通过，否则不能通过，其本质就是一个开关。

就是这个小东西的发明让三个人获得了诺贝尔物理学奖，可见其举足轻重的地位。**无论程序员编写的程序多么复杂，软件承载的功能最终都是通过这个小东西简单的开闭完成的，**
想不出其它词来。

▲ 赞同 1.4 万

▼

550 条评论

分享

收藏

喜

知乎 2021 年度精选

知乎年度 100 问 · 第 48 问
有没有发现本次奥运会国人不再唯金牌论了？
2021 年 7-8 月 · 2020 年东京奥运会

前排围观

关于作者



码农的荒岛求生



《计算机底层的秘密》作者

回答

110

文章

57

关注者

22,886

关注

发私信

被收藏 15,500 次

实用技能

10,544 人关注

SUNNY CHAN 创建

愉悦的杂谈

3,667 人关注

Breeze先生 创建

故事

2,613 人关注

仙庭之主 创建

思考

962 人关注

黎竹岩 创建

计算机类经典回答

497 人关注

克里斯托弗 创建

相关问题

知乎

[首页](#)
[会员](#)
[发现](#)
[等你来答](#)

有没有男主是动物的小说推荐



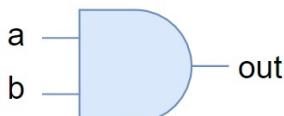
提问



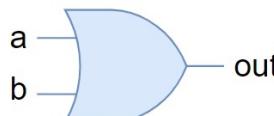
现在有了晶体管，也就是开关，在此基础之上就可以搭积木了，你随手搭建出来这样三种组合：

- 两个开关只有同时打开电流才会通过，灯才会亮
- 两个开关中只要有一个打开电流就能通过，灯就会亮
- 当开关关闭时电流通过灯会亮，打开开关灯反而电流不能通过灯会灭

天赋异禀^o的你搭建的上述组合分别就是：与门，AND Gate、或门，OR gate、非门，NOT gate，用符号表示就是这样：

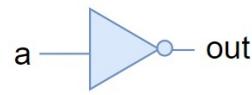


AND



OR

知乎 @码农的荒岛求生



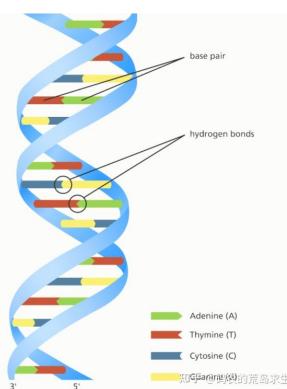
NOT

道生一、一生二、二生三、三生万物

最神奇的是，你随手搭建的三种电路竟然有一种很amazing的特性，那就是：任何一个逻辑函数最终都可以通过AND、OR以及NOT表达出来，这就是所谓的逻辑完备性，就是这么神奇。

也就是说给定足够的AND、OR以及NOT门，就可以实现任何一个逻辑函数，除此之外我们不需要任何其它类型的逻辑门电路，这时我们认为{AND、OR、NOT}就是逻辑完备的。

这一结论的得出吹响了计算机革命的号角，这个结论告诉我们计算机最终可以通过简单的{AND、OR、NOT}门构造出来，就好比基因。



老子有云：道生一、一生二、二生三、三生万物，实乃异曲同工之妙。虽然，我们可以用AND、OR、NOT来实现所有的逻辑运算，但我们真的需要把所有的逻辑运算都用AND、OR、NOT门实现出来吗？显然不是，而且这也不太可行。

计算能力是怎么来的

现在能生成万物的基础元素与或非门出现了，接下来我们着手设计CPU最重要的能力：计算，以加法为例。由于CPU只认知 0 和 1，也就是二进制，那么二进制的加法有哪些组合呢：

- 0 + 0，结果为0，进位为0
- 0 + 1，结果为1，进位为0
- 1 + 0，结果为1，进位为0
- 1 + 1，结果为0，进位为1，二进制嘛！

计算机专业的学生，笔记本cpu一定要intel家的吗，amd的行不行？听说amd指令集缺少会有麻烦？7个回答

为什么不设计面向对象的指令集和CPU？6个回答

CPU 的指令集存放在什么地方？25个回答

相关推荐

**Intel Galileo Essentials**

Richard Grimmett

0人读过

阅读

**Home Automation with Intel Galileo**

1人读过

阅读

**地球至强男人**

甘十九

1人读过

阅读



哈佛大学入学条件

刘看山 · 知乎指南 · 知乎协议 ·

知乎隐私保护指引

应用 · 工作 · 申请开通知乎机构号

侵权举报 · 网上有害信息举报专区

京 ICP 证 110745 号

京 ICP 备 13052560 号 - 1

京公网安备 11010802020088 号

互联网药品信息服务资格证书

(京) - 非经营性 - 2017 - 0067

服务热线：400-919-0001

违法和不良信息举报：010-82716601

举报邮箱：jubao@zhihu.com

儿童色情信息举报专区

信息安全漏洞反馈专区

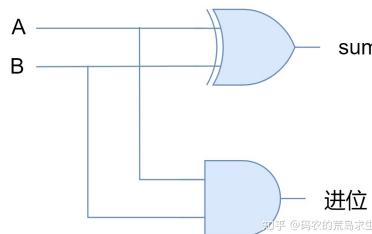
内容从业人员违法违规行为举报

证照中心 · Investor Relations

联系我们 © 2022 知乎

这就是异或啊，有没有！

我们说过与或非门是逻辑完备可以生万物，异或逻辑当然不在话下，用一个与门和一个异或门就可以实现二进制加法：



上述电路就是一个简单的加法器，就问你神奇不神奇，加法可以用与或非门实现，其它的也一样能实现，逻辑完备嘛。

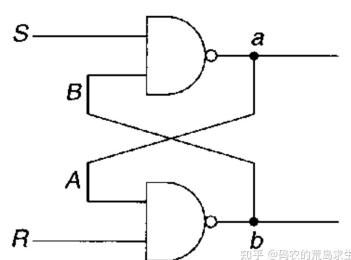
除了加法，我们也可以根据需要将不同的算数运算设计出来，负责计算的电路有一个统称，这就是所谓的arithmetic/logic unit, ALU, CPU 中专门负责运算的模块，本质上和上面的简单电路没什么区别，就是更加复杂而已。

现在，通过与或非门的组合我们获得了计算能力，计算能力就是这么来的。但，只有计算能力是不够的，电路需要能记得住信息。

神奇的记忆能力

到目前为止，你设计的组合电路比如加法器天生是没有办法存储信息的，它们只是简单的根据输入得出输出，但输入输出总的有个地方能够保存起来，这就是需要电路能保存信息。

电路怎么能保存信息呢？你不知道该怎么设计，这个问题解决不了你寝食难安，吃饭时在思考、走路时在思考，蹲坑时在思考，直到有一天你在梦中遇一位英国物理学家，他给了你这样一个简单但极其神奇的电路：



这是两个NAND门的组合，不要紧张，NAND也是有你设计的与或非门组合而成的，所谓NAND门就是与非门，先与然后取非，比如给定输入1和0，那么与运算后为0，非运算后为1，这就是与非门，这些不重要。

比较独特的是该电路的组合方式，一个NAND门的输出是两个NAND门的输入，该电路的组合方式会自带一种很有趣的特性，只要给S和R段输入1，那么这个电路只会有两种状态：

- 要么a端为1，此时B=0、A=1、b=0；
- 要么a端为0，此时B=1、A=0、b=1；

不会再有其他可能了，我们把a端的值作为电路的输出。

此后，你把S端置为0的话(R保持为1)，那么电路的输出也就是a端永远为1，这时就可以说我们把1存到电路中了；而如果你把R端置为0的话(S保持为1)，那么此时电路的输出也就是a端永远为0，此时我们可以说把0存到电路中了。

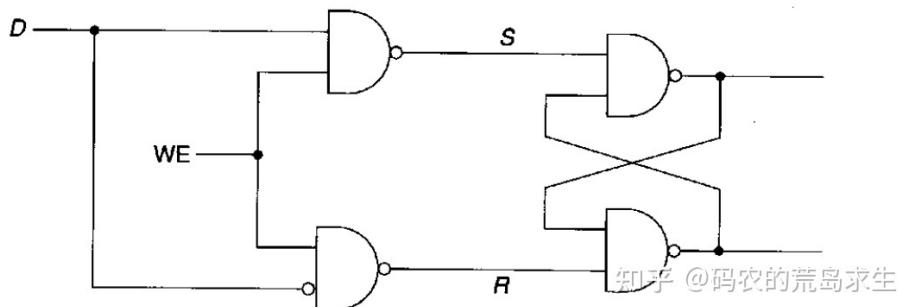
▲ 赞同 1.4 万

550 条评论

分享

收藏

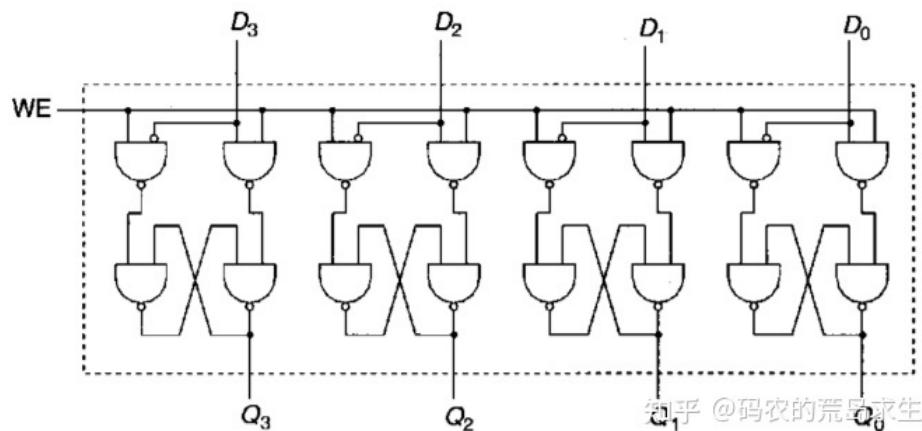
喜



这样，当D为0时，整个电路保存的就是0，否则就是1。

寄存器与内存的诞生

现在你的电路能存储一个比特位了，想存储多个比特位还不简单，复制粘贴就可以了：



我们管这个组合电路就叫**寄存器**，你没有看错，我们常说的寄存器就是这个东西。

你不满足，还要继续搭建更加复杂的电路以存储更多信息，同时提供寻址功能，就这样**内存**也诞生了。

寄存器及内存都离不开上一节那个简单电路，只要通电，这个电路中就保存信息，但是断电后很显然保存的信息就丢掉了，现在你应该明白为什么内存存在断电后就不能保存数据了吧。

硬件还是软件？

现在我们的电路可以计算数据、也可以存储信息，但现在还有一个问题，那就是尽管我们可以用AND、OR、NOT表达出所有的逻辑函数，但我们真的有必要把所有的逻辑运算都用与或非门实现出来吗？这显然是不现实的。

这就好比厨师，你没有听说只专做一道菜的厨师然后酒店要把各个菜系厨师雇全才能做出一桌菜来吧！

中国菜系博大精深，千差万别，但制作每道菜品的方式大同小异，其中包括刀工、颠勺技术等，这些是基本功，制作每道菜品都要经过这些步骤，变化的也无非就是食材、火候、调料等，这些放到菜谱中即可，这样给厨师一个菜谱他就能制作出任意的菜来，在这里厨师就好比硬件，菜谱就好比软件。



首页 会员 发现 等你来答

有没有男主是动物的小说推荐



提问



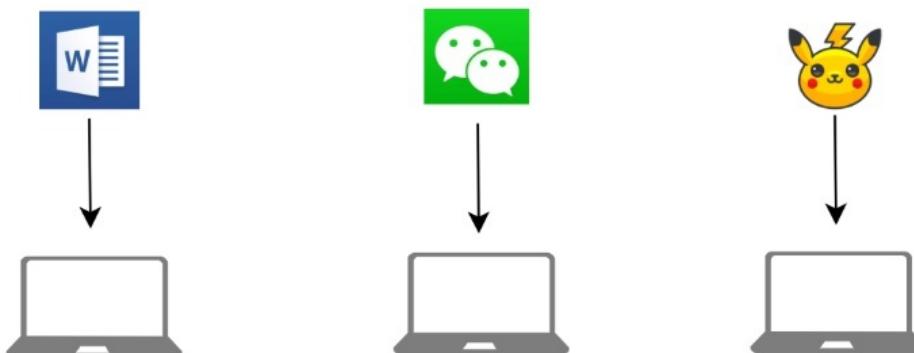
知乎 @码农的荒岛求生

同样的道理，我们没有必要为所有的计算逻辑实现出对应的硬件，硬件只需要提供最基本的功能，最终所有的计算逻辑都通过这些最基本的功能用软件表达出来就好，这就是所谓的软件一词的来源，**硬件不可变，但软件可变，不变的是硬件但提供不同的软件就能让硬件实现全新的功能，无比天才的思想**，人类真的是太聪明了。

同样一台计算机硬件，安装上word你就能编辑文档，安装上微信你就能在公号中读到码农的荒岛求生、安装上游戏你就能玩王者荣耀，硬件还是那套硬件，提供不同的软件就是实现不同的功能，**每次打开电脑使用各种App时没有在内心高呼一声牛逼你都对不起计算机这么伟大的发明创造**，这就是为什么计算机被称为通用计算设备的原因，这一思想是计算机科学祖师爷图灵提出的。

说到牛逼的通用设备，在这样也推荐一份牛逼的算法刷题资料，除了本文讲到的底层技术，，想进BAT、TMD、快手这样的一线大厂算法绝不可忽视，认认真真过上一遍这份资料，这些大厂算法面试一关大部分题目都不在话下：

Github疯传！阿里P8大佬写的Leetcode刷题笔记，秒杀80%的算法题！
mp.weixin.qq.com/s/A-HPH3Tkl8KOvZgdR...



文本编辑器

聊天工具

知乎 @码农的荒岛求生 游戏机

扯远了，接下来我们看下硬件是怎么提供所谓的基本功能的。

硬件的基本功

▲ 赞同 1.4 万

550 条评论

分享

收藏

喜



首页 会员 发现 等你来答

有没有男主是动物的小说推荐



提问



很显然，你得告诉CPU，该怎么告诉呢？还记得上一节中给厨师的菜谱吗？没错，CPU也需要一张菜谱告诉自己该接下来该干啥，在这里菜谱就是机器指令，指令通过我们上述实现的组合电路来执行。

接下来我们面临另一个问题，那就是这样的指令应该会很多吧，废话，还是以加法指令为例，你可以让CPU计算 $1+1$ ，也可以计算 $1+2$ 等等，实际上单单加法指令就可以有无数种组合，显然CPU不可能去实现所有的指令。

实际上CPU只需要提供加法操作，你提供操作数就可以了，CPU说：“我可以打人”，你告诉CPU该打谁、CPU说：“我可以唱歌”，你告诉CPU唱什么，CPU说我可以做饭，你告诉CPU该做什么饭，CPU说：“我可以炒股”，你告诉CPU快滚一边去吧韭菜。因此我们可以看到CPU只提供机制或者说功能(打人、唱歌、炒菜，加法、减法、跳转)，我们提供策略(打谁、歌名、菜名，操作数，跳转地址)。

CPU 表达机制就通过[指令集](#)来实现的。

指令集

指令集告诉我们CPU可以执行什么指令，每种指令需要提供什么样的操作数。不同类型的CPU会有不同的指令集。指令集中的指令其实都非常简单，画风大体上是这样的：

- 从内存中读一个数，地址是abc
- 对两个数加和
- 检查一个数是不是大于6
- 把这数存储到内存，地址是abc
- 等等

看上去很像碎碎念有没有，这就是机器指令，我们用高级语言编写的程序，比如对一个数组进行排序，最终都会等价转换为上面的碎碎念指令，然后CPU一条一条的去执行，很神奇有没有。接下来我们看一条可能的机器指令：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	0	1	0	0	0	0	1	1	0

ADD R6 R2 知乎 @码农的荒岛求生

这条指令占据16比特，其中前四个比特告诉CPU这是加法指令，这意味着该CPU的指令集中可以包含 2^4 也就是16个机器指令，这四个比特位告诉CPU该做什么，剩下的bit告诉CPU该怎么做，也就是把寄存器R6和寄存器R2中的值相加然后写到寄存器R6中。

可以看到，机器指令是非常繁琐的，现代程序员都使用高级语言来编写程序，关于[高级程序语言](#)以及机器指令的话题请参见[《你管这破玩意叫编程语言？》](#)。

码农的荒岛求生：你管这破玩意叫编程语言？

2837 赞同 · 116 评论 文章



指挥家：让我们演奏一曲

现在我们的电路有了计算功能、存储功能，还可以通过指令告诉该电路执行什么操作。还有一个问题没有解决。

▲ 赞同 1.4 万

550 条评论

分享

收藏

喜

工作时R1和R2中在这一时刻保存的都是1而不是其它数?

即，我们靠什么来协调或者说靠什么来同步电路各个部分让它们协同工作呢？就像一场成功的交响乐演离不开指挥家一样，我们的计算组合电路也需要这样一个指挥家。



负责指挥角色的就是[时钟信号](#)。

时钟信号就像指挥家里拿的指挥棒，[指挥棒挥动一下整个乐队会整齐划一的有个相应动作](#)，同样的，时钟信号每一次电压改变，整个电路中的各个寄存器(也就是整个电路的状态)会更新一下，这样我们就能确保整个电路协同工作不会这里提到的问题。

现在你应该知道CPU的主频是什么意思了吧，主频是说一秒钟指挥棒挥动了多少次，显然主频越高CPU在一秒内完成的操作也就越多。

大功告成

现在我们有了可以完成各种计算的ALU、可以存储信息的寄存器以及控制它们协同工作的时钟信号，这些统称 **Central Processing Unit**，简称就是 **CPU**。

接下来我们看一下CPU与高级语言。

创世纪：聪明的笨蛋

CPU相当原始，就像单细胞生物一样，只能把数据从一个地方搬到另一个地方、简单的加一下，没有任何高难度动作，这些操作虽然看上去很简单很笨，但CPU有一个无与伦比的优势，那就是一个字：快，这是人类比不了的，**CPU出现后人类开始拥有第二个大脑**。就是这样原始的一个物种开始支配起另一个叫做程序员的物种。

干活的是大爷

一般来说两个不同的物种要想交流，比如人和鸟，就会有两种方式：要不就是鸟说人话，让人听懂；要不就是人说鸟语，让鸟听懂；就看谁厉害了。最开始CPU胜出，程序员开始说鸟语并认真感受CPU的支配地位，好让CPU大爷可以工作，感受一下最开始的程序员是怎么说鸟语的：

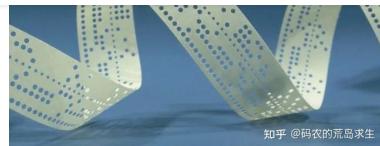


首页 会员 发现 等你来答

有没有男主是动物的小说推荐



提问



程序员按照 CPU 的旨意直接用0和1编写指令，你没有看错，这破玩意就是代码了，就是这么原生态，然后放到打孔纸带上输入给CPU，CPU 开始工作，这时的程序可真的是看得见摸得着，就是有点浪费纸。

这时程序员必须站在 CPU 的角度来写代码，画风是这样的：

```
1101101010011010
1001001100101001
1100100011011110
1011101101010010
```

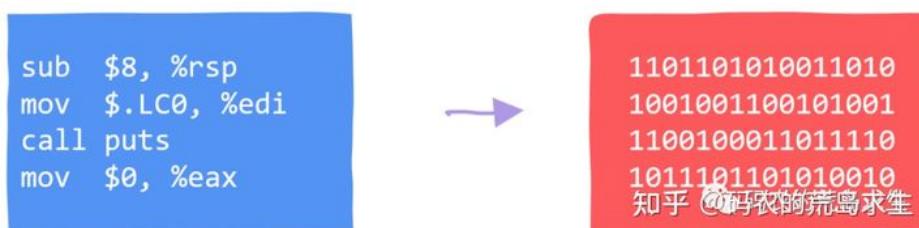
乍一看你知道这是什么意思吗？你不知道，心想：“这是什么破玩意？”，但 CPU 知道，心想“这就简直就是世界上最美的语言”。

天降大任

终于有一天程序员受够了说鸟语，好歹也是灵长类，叽叽喳喳说鸟语太没面子，你被委以重任：让程序员说人话。你没有苦其心志劳其筋骨，而是仔细研究了一下 CPU，发现 CPU 执行的指令集来来回回就那么几个指令，比如加法指令、跳转指令等等，因此你把机器指令和对应的具体操作做了一个简单的映射，**把机器指令映射到人类能看懂的单词**，这样上面的01串就变成了：

```
sub $8, %rsp
mov $.LC0, %edi
call puts
mov $0, %eax
```

这样，程序员不必生硬的记住1011.....，而是记住人类可以认识的ADD SUB MUL DIV等这样的单词即可。



汇编语言就这样诞生了，编程语言中首次出现了人类可以认识的东西。

这时程序员终于不用再“叽叽喳喳。。”，而是升级为“阿巴阿巴。。”，虽然人类认知“阿巴阿巴”这几个字，但这和人类的语言在形式上差别还是有点大。

细节 VS 抽象

尽管汇编语言已经有人类可以认识的单词，但汇编语言和机器语言一样都属于低级语言。所谓低级语言是说你需要**关心所有细节**。关心什么细节呢？我们说过，CPU 是非常原始的东西，只知道把数据从一个地方搬到另一个地方，简单的操作一下再从一个地方搬到另一地方。因此级语言来编程的话，你需要使用多个“**把数据从一个地方搬到另一个地方**，简单

▲ 赞同 1.4 万

550 条评论

分享

收藏

喜



首页 会员 发现 等你来答

有没有男主是动物的小说推荐



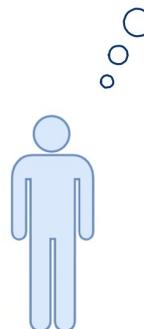
提问

45

3



给我端杯水



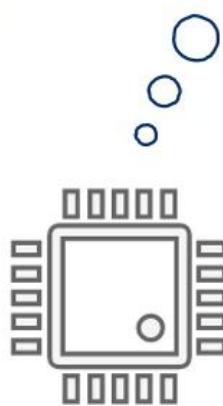
知乎 @勇敢的荒岛求生

如果你用汇编这种低级语言就得这样实现：

```

迈出右腿
停住
迈出左腿
停住
重复上述步骤直到饮水旁
找到水杯
抬起你的右手
抓住水杯
移动到出水口
伸出左手
打开出水开关
如果水没有接满
继续等待
如果水已经满了
关闭开关
向后翻转180度
迈出右腿
停住
迈出左腿
停住
重复上述步骤直到回来

```



知乎 @勇敢的荒岛求生

我想你已经 Get 到了。

弥补差异

▲ 赞同 1.4 万



● 550 条评论

▼ 分享

★ 收藏

♥ 喜欢



首页 会员 发现 等你来答

有没有男主是动物的小说推荐



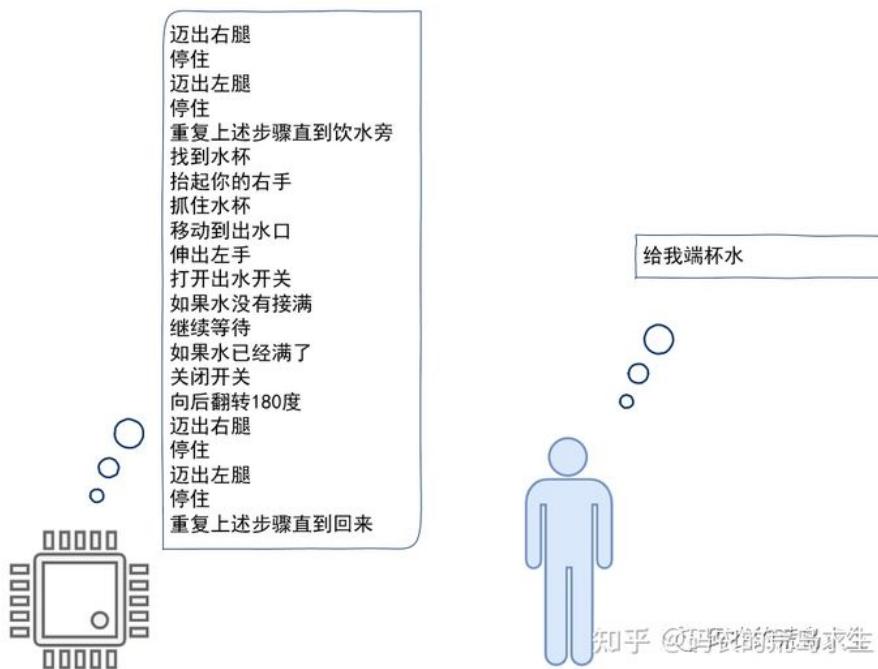
提问

45

3



动把人类抽象的表达转为 CPU 可以理解的具体实现，这显然可以极大增强程序员的生产力，现在，
这个问题需要你来解决。



套路，都是套路

思来想去你都不知道该怎么把人类的抽象自动转为 CPU 能理解的具体实现，就在要放弃的时候你又
看了一眼 CPU 可以理解的一堆细节：

▲ 赞同 1.4 万

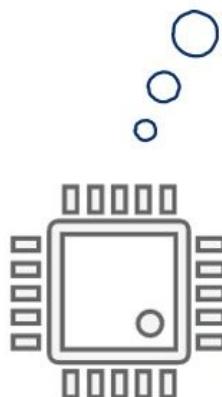


● 550 条评论

▼ 分享

★ 收藏

♥ 喜



停住
迈出左腿
停住
重复上述步骤直到饮水旁
找到水杯
抬起你的右手
抓住水杯
移动到出水口
伸出左手
打开出水开关
如果水没有接满
继续等待
如果水已经满了
关闭开关
向后翻转180度
迈出右腿
停住
迈出左腿
停住
重复上述步骤直到回来

知乎 @阳朔的海岛先生

电光火石之间灵光乍现，你发现了满满的套路，或者说模式。大部分情况下 CPU 执行的指令[平铺直叙](#)的，就像这样：



首页 会员 发现 等你来答

有没有男主是动物的小说推荐



提问

45

3



- 1, 1往左
- 3, 迈出左腿
- 4, 停住
- 5, 重复上述步骤直到饮水旁
- 6, 找到水杯
- 7, 抬起你的右手
- 8, 抓住水杯
- 9, 移动到出水口
- 10, 伸出左手
- 11, 打开出水开关
- 12, 如果水没有接满
- 13, 继续等待
- 14, 如果水已经满了
- 15, 关闭开关
- 16, 向后翻转180度
- 17, 迈出右腿
- 18, 停住
- 19, 迈出左腿
- 20, 停住
- 21, 重复上述步骤直到回来

知乎 @阿强的菜鸟人生

这些都是告诉 CPU 完成某个特定动作，你给这些平铺直叙的指令起了个名字，姑且就叫陈述句吧，statement。

除此之外，你还发现了这样的套路，那就是需要根据某种特定状态决定走哪段指令，这个套路在人看来就是“如果。。。就。。。否则。。。就。。。”：

```
if ***
    blablabla
else ***
    blablabla
```

在某些情况下还需要不断重复一些指令，这个套路看起来就是原地打转：

```
while ***
    blablabla
```

最后就是这里有很多看起来差不多的指令，就像这里：

▲ 赞同 1.4 万

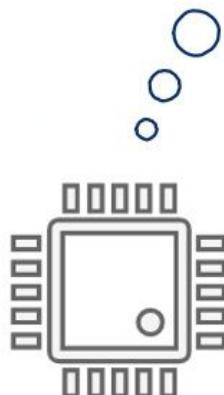
▼

550 条评论

分享

收藏

喜



2, 1停住
 3, 迈出左腿
 4, 停住
 5, 重复上述步骤直到饮水旁
 6, 找到水杯
 7, 抬起你的右手
 8, 抓住水杯
 9, 移动到出水口
 10, 伸出左手
 11, 打开出水开关
 12, 如果水没有接满
 13, 继续等待
 14, 如果水已经满了
 15, 关闭开关
 16, 向后翻转180度
 17, 迈出右腿
 18, 停住
 19, 迈出左腿
 20, 停住
 21, 重复上述步骤直到回来

知乎·@程序员的日常生活

这些指令是重复的，只是个别细节有所差异，把这些差异提取出来，剩下的指令打包到一起，用一个代号来指定这些指令就好了，这要有个名字，就叫函数吧：

```
func abc:  

    blablabla
```

现在你发现了所有套路：

```
// 条件转移  

if ***  

    blablabla  

else ***  

    blablabla  

// 循环  

while ***  

    blablabla  

// 函数  

func abc:  

    blablabla
```

这些相比汇编语言已经有了质的飞跃，因为这已经和人类的语言非常接近了。接下来你发现自己面临两个问题：

1. 这里的blablabla该是什么呢？
2. 该怎样把上面的人类可以认识的字符串转换为CPU可以认识的机器指令

▲ 赞同 1.4 万

● 550 条评论

▼ 分享

★ 收藏

♥ 喜

盗梦空间

你想起来了，上文说过大部分代码都是平铺直叙的陈述句，statement，这里的blablabla 仅仅就是一堆陈述句吗？显然不是，blablabla 可以是陈述句，当然也可以是条件转移if else，也可以是循环while，也可以是调用函数，这样才合理。虽然这样合理，很快你就发现了另一个严重的问题：blablabla中可以包含 if else 等语句，而if else等语句中又可以包含blablabla，blablabla中反过来又双可能会包含if else等语句，if else等语句又双可能会包含blablabla，blablabla又双可能会包含if else等语句。。。。



就像盗梦空间一样，一层梦中还有一层梦，梦中之梦，梦中之梦中之梦。。。一层嵌套一层，子子孙孙无穷匮也。。。

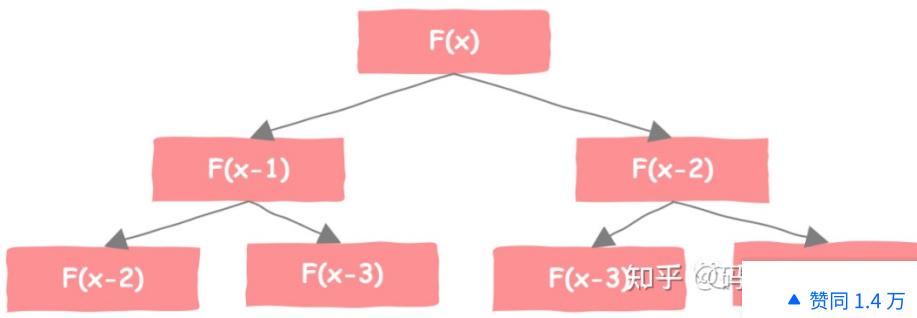


此时你已经明显感觉脑细胞不够用了，这也太复杂了吧，绝望开始吞噬你，上帝以及老天爷啊，谁来救救我！

此时你的高中老师过来拍了拍你的肩膀，递给了你一本高中数学课本，你恼羞成怒，给我这破玩意干什么，我现在想的问题这么高深，岂是一本破高中数学能解决的了的，抓过来一把扔在了地上。此时一阵妖风吹过，教材停留在了这样一页，上面有这样一个数列表达：

$$f(x) = f(x-1) + f(x-2)$$

这个递归公式在表达什么呢？ $f(x)$ 的值依赖 $f(x-1)$ ， $f(x-1)$ 的值又依赖 $f(x-2)$ ， $f(x-2)$ 的值又依赖。。。。



▲ 赞同 1.4 万

550 条评论

分享

收藏

喜

知乎

首页 会员 发现 等你来答

有没有男主是动物的小说推荐



提问

45

3



等一下，这不就是递归嘛，上面看似无穷无尽的嵌套也可以用递归表达啊！你的数学老师仰天大笑，too young too simple，留下羞愧的你佛手而去，看似高科技的东西竟然用高中数学就解决了，一时震惊的目瞪狗带不知所措无地自容。有了递归这个概念加持，聪明的智商又开始占领高地了。

递归：代码的本质

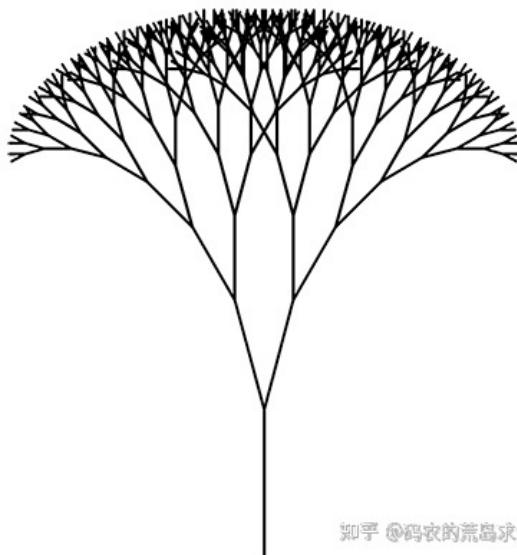
不就是嵌套嘛，一层套一层嘛，递归天生就是来表达这玩意的 (提示：这里的表达并不完备，真实的编程语言不会这么简单，这里仅仅用作示例)：

```
if : if bool statement else statements
for: while bool statements
statement: if | for | statement
```

上面一层嵌套一层的盗梦空间原来可以这么简洁的几句表达出来啊，你给这几句表达起了高端的名字，语法。数学，就是可以让一切都变得这么优雅。世界上所有的代码，不管有多么复杂最终都可以归结到语法上，原因也很简单，所有的代码都是按照语法的形式写出来的嘛。至此，你发明了真正的人类可以认识的编程语言。之前提到的第一个问题解决了，但仅有语言还是不够的。

让计算机理解递归

现在还差一个问题，怎样才能把这语言最终转化为 CPU 可以认识的机器指令呢？人类可以按照语法写出代码，这些代码其实就是一串字符，怎么让计算机也能认识用递归语法表达的一串字符呢？这是一项事关人类命运的事情，你不禁感到责任重大，但这最后一步又看似困难重重，你不禁仰天长叹，计算机可太难了。此时你的初中老师过来拍了拍你的肩膀，递给了你一本初中植物学课本，你恼羞成怒，给我这破玩意干什么，我现在想的问题这么高深，岂是一本破初中教科书能解决的了的，抓过来一把扔在了地上。此时又一阵妖风刮过，书被翻到了介绍树的一章，你望着这一页不禁发起呆来：



知乎 @码农的荒岛求生

树干下面是树枝，树枝下是树叶，树枝下也可以是树枝，树枝下还可以是树枝、吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮，哎？这句不对，回到上面这句，树干生树枝，树枝还可以生树枝，一层套一层、梦中之梦、子子孙孙无穷匮、高中数学老师，等一下，这也是递归啊！！！我们可以把根据递归语法写出来的代码用树来表示啊！

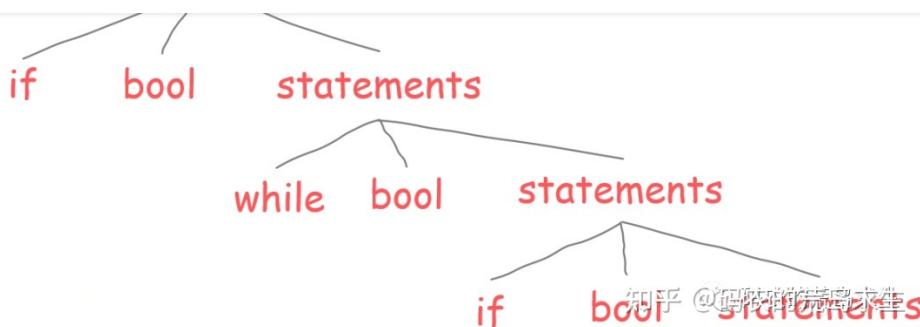
▲ 赞同 1.4 万

550 条评论

分享

收藏

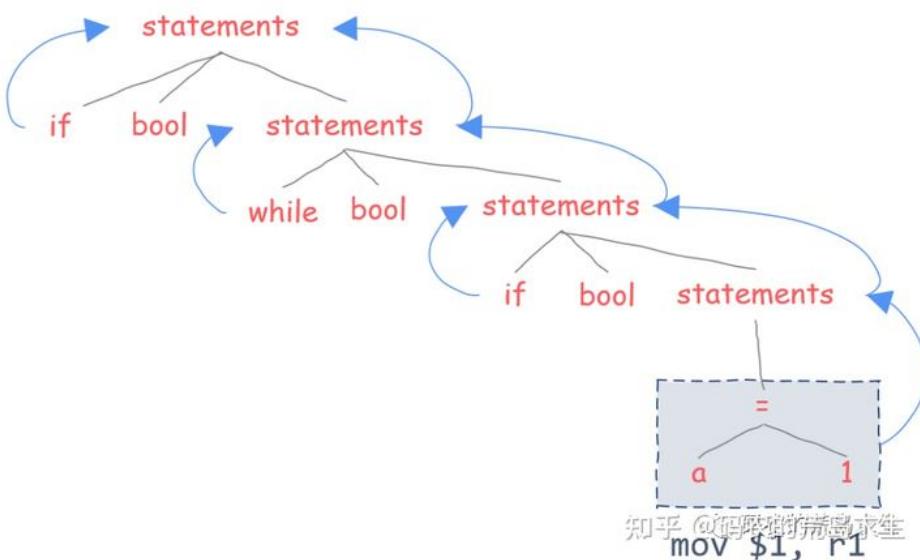
喜



你的初中老师仰天大笑，图样图森破，看似高科技的东西竟然靠初中知识就解决了。

优秀的翻译官^o

计算机处理编程语言时可以按照递归定义把代码用树的形式组织起来，由于这棵树是按照语法生成的，姑且就叫语法树吧。现在代码被表示成了树的形式，你仔细观察后发现，其实叶子节点的表达是非常简单的，可以很简单的翻译成对应的机器指令，只要叶子节点翻译成了机器指令，你就可以把此结果应用到叶子节点的父节点，父节点又可以把翻译结果引用到父节点的父节点，一层层向上传递，最终整颗树都可以翻译成具体的机器指令。



完成这个工作的程序也要有个名字，根据“弄不懂原则”(该原则的解释见下文：)

看完这篇还不懂高并发中的线程与线程池你
来打我(内含20张图)

mp.weixin.qq.com/s?__biz=MzU2NTYyOT...



，你给这个类似翻译的程序起了个不怎么响亮的名字，编译器，compiler。

现在你还觉得二叉树^o之类的数据结构没啥用吗？对了，说到二叉树，在这样也推荐一份牛逼的算法刷题资料，想进BAT、TMD、快手这样的一线大厂算法绝不可忽视，认认真真过上一遍这份资料，这些大厂算法面试一关大部分题目都不在话下：

Github疯传！阿里P8大佬写的Leetcode刷题笔记，秒杀80%的算法题！

mp.weixin.qq.com/s/A-HPH3Tkl8K0vZgdR...



▲ 赞同 1.4 万

550 条评论

分享

收藏

喜



首页 会员 发现 等你来答

有没有男主是动物的小说推荐



提问

45

3



及后续的Java、Python，这些语言现在还有一帮人在用呢。

总结

本文我们从最基本的晶体管一路讲解到CPU的工作原理，再从最低级的二进制机器指令到高级语言，相信如果你能读到这里定能有所收获。

最后，有同学问有没有书单，我也仔细回想自己认真读过的计算机数据，在这里也给出自认为很经典的几本，书单这东西贵精不贵多，我在这里精心挑选了10本，**不要贪心，如果你真能把这里推荐的10本书读通，可以说你已经能超越90%的程序员了。**

程序员必看经典书单

mp.weixin.qq.com/s/4T-ZLWasVDQVY99gC...



最后的最后，有很多知乎朋友问有没有pdf版本，我也整理出来了，绘图非常精美，这里还汇总了部分知乎问题，总计14万字，我为其专门设计了封面，并将其命名为《计算机底层的秘密》，现在免费分享给大家。



▲ 赞同 1.4 万

● 550 条评论

▼ 分享

★ 收藏

● 喜

知乎

[首页](#)
[会员](#)
[发现](#)
[等你来答](#)

有没有男主是动物的小说推荐



提问

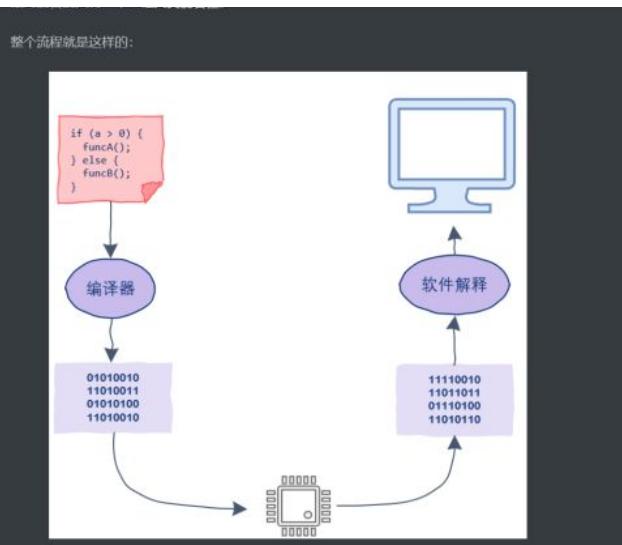


二、CPU空闲时在干嘛?
剩下的CPU时间去哪了?
程序、进程与操作系统的
调度器与进程管理
队列中的一个更好的设计
一切都是为了操作硬件
软件硬件结合
总结
关注作者

三、编译器是如何工作的?
编译器是一个普通的程序,没什么大不了的
输出出错一单词: 语法分析
这些token表达什么意思: 语义分析
语义分析不合理的: 语义分析
根据既定规则中的代码: 代码生成
中间语言优化
代码生成
总结
关注作者

四、函数运行在内存中是什么样子?
从进程、线程到函数调用
函数调用的流动轨迹: 栈
A Box
控制转移
传播参数与返回值
总结
关注作者

五、请教理解困难的我



可以使用这个下载链接: [点击下载《计算机底层的秘密》PDF](#)

如果你觉得本文有所帮助, 给点个赞呗! [@码农的荒岛求生](#)

作者: 码农的荒岛求生

Github: github.com/xfenglu/ever...

原文: [你管这破玩意叫CPU?](#)

[你管这破玩意叫编程语言?](#)

推荐:

码农的荒岛求生: CPU 空闲时在干嘛?
3124 赞同 · 145 评论 文章



编辑于 2021-12-30 17:42

更多回答

 柳两丛
ars longa, vita brevis

 专业 已有 2 人赠与了专业徽章 >

19,724 人赞同了该回答

上二年级的小明正坐在教室里。现在是数学课，下午第一节，窗外的蝉鸣、缓缓旋转的吊扇让同学们昏昏欲睡。此时，刘老师在黑板上写下了一个问题：

6324 + 244675 = ?

小明抬头看了一眼，觉得这两个数字挺眼熟。他昨天翘课去网吧了，因此错过了刘老师讲的竖式计算加法。

▲ 赞同 1.4 万

550 条评论

分享

收藏

喜



首页 会员 发现 等你来答

有没有男主是动物的小说推荐



提问



小明盯着黑板懵逼。

小学二年级的他面对这样一道世界级难题，束手无策。小明伸出了自己的左手，打算用一个古老而深邃的方法--掰手指--尝试一下。

[展开阅读全文 ▾](#)

小明发现他的每只手只能输入0-5中的正整数，和的范围仅限于0-10，离6324还十分遥远。

[▲ 赞同 1.9 万](#) [▼](#) [1,133 条评论](#) [分享](#) [收藏](#) [喜欢](#) ...



王锐超

京东方 系统集成工程师

[6 专业 已有 6 人赠与了专业徽章 >](#)

12,467 人赞同了该回答

文章比较长，不过我相信是值得花时间观看的，**一定能看到别处看不到的知识**，能对 CPU 有更为深入的理解。**第一部分**是补充背景知识的**综述部分**，我相信就算是外行耐下心来看也是可以读懂的；文章的第二部分深度会有增加，下面为第二部分的目录，有基础知识的建议直接跳到后面去阅读。

- 1，为什么 MOSFET 是逻辑电路的最基本单元，为什么电路最基本的逻辑是‘非’，以及为什么逻辑是“与非”“或非”而不是‘与’‘或’
- 2，把‘电子’装进盒子里的各种储存单元：register（俗称寄存器，L0 cache），cache（SRAM），DRAM（俗称内存），& Flash Drive（俗称‘闪存’）
- 3，TDP，主频，超频及其相关

欢迎大家指正错误！我会及时改正的。

如果大家还有感兴趣的话题在下方留言即可，答主一定会认真回复的，谢谢支持！！！

[展开阅读全文 ▾](#)

[▲ 赞同 1.2 万](#) [▼](#) [570 条评论](#) [分享](#) [收藏](#) [喜欢](#) ...

[查看全部 158 个回答](#)

[▲ 赞同 1.4 万](#) [▼](#) [550 条评论](#) [分享](#) [收藏](#) [喜](#)