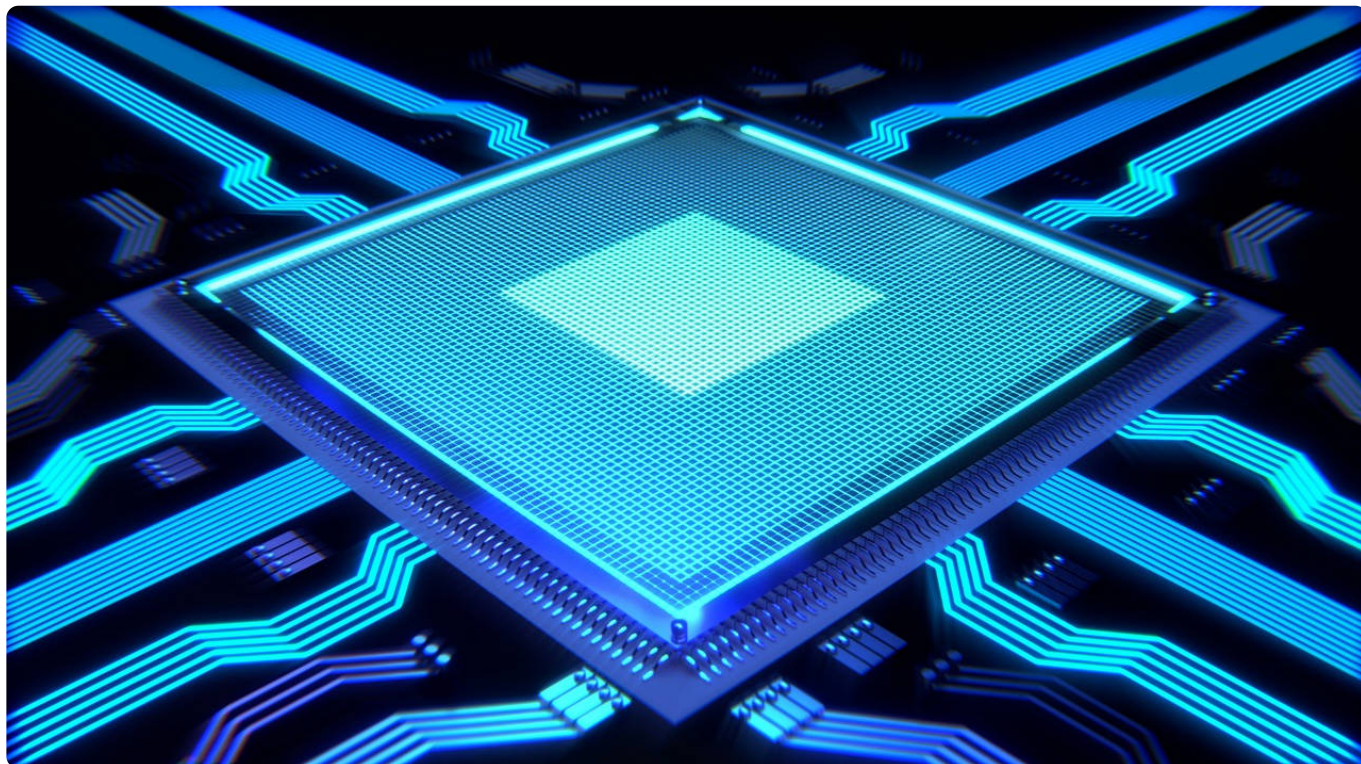


## 19 | 建立数据通路（下）：指令+运算=CPU

2019-06-07 徐文浩

深入浅出计算机组成原理

[进入课程 >](#)



讲述：徐文浩

时长 11:07 大小 10.19M



上一讲，我们讲解了时钟信号是怎么实现的，以及怎么利用这个时钟信号，来控制数据的读写，可以使得我们能把需要的数据“存储”下来。那么，这一讲，我们要让计算机“自动”跑起来。

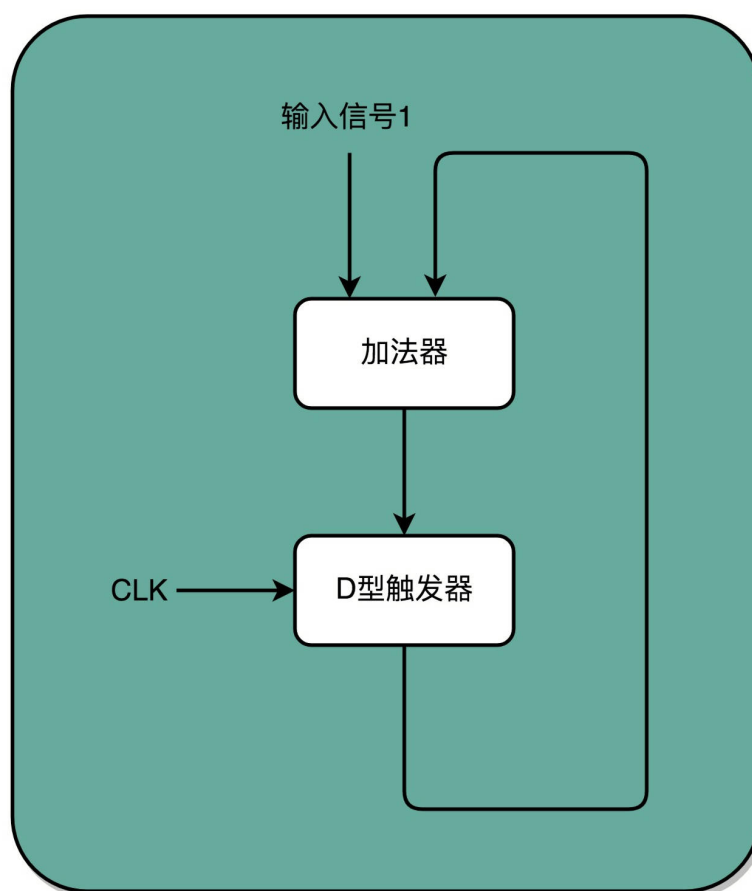
通过一个时钟信号，我们可以实现计数器，这个会成为我们的 PC 寄存器。然后，我们还需要一个能够帮我们在内存里面寻找指定数据地址的译码器，以及解析读取到的机器指令的译码器。这样，我们就能把所有学习到的硬件组件串联起来，变成一个 CPU，实现我们在计算机指令的执行部分的运行步骤。

### PC 寄存器所需要的计数器

我们常说的 PC 寄存器，还有个名字叫程序计数器。下面我们就来看看，它为什么叫作程序计数器。

有了时钟信号，我们可以提供定时的输入；有了 D 型触发器，我们可以在时钟信号控制的时间点写入数据。我们把这两个功能组合起来，就可以实现一个自动的计数器了。

加法器的两个输入，一个始终设置成 1，另外一个来自于一个 D 型触发器 A。我们把加法器的输出结果，写到这个 D 型触发器 A 里面。于是，D 型触发器里面的数据就会在固定的时钟信号为 1 的时候更新一次。



这样，我们就有了一个每过一个时钟周期，就能固定自增 1 的自动计数器了。这个自动计数器，可以拿来当我们的 PC 寄存器。事实上，PC 寄存器的这个 PC，英文就是 Program Counter，也就是**程序计数器**的意思。

每次自增之后，我们可以去对应的 D 型触发器里面取值，这也是我们下一条需要运行指令的地址。前面第 5 讲我们讲过，同一个程序的指令应该要顺序地存放在内存里面。这里就和前面对应上了，顺序地存放指令，就是为了让通过程序计数器就能定时地不断执行新指令。

加法计数、内存取值，乃至后面的命令执行，最终其实都是由我们一开始讲的时钟信号，来控制执行时间点和先后顺序的，这也是我们需要时序电路最核心的原因。

在最简单的情况下，我们需要让每一条指令，从程序计数，到获取指令、执行指令，都在一个时钟周期内完成。如果 PC 寄存器自增地太快，程序就会出错。因为前一次的运算结果还没有写回到对应的寄存器里面的时候，后面一条指令已经开始读取里面的数据来做下一次计算了。这个时候，如果我们的指令使用同样的寄存器，前一条指令的计算就会没有效果，计算结果就错了。

在这种设计下，我们需要在一个时钟周期里，确保执行完一条最复杂的 CPU 指令，也就是耗时最长的一条 CPU 指令。这样的 CPU 设计，我们称之为**单指令周期处理器**（Single Cycle Processor）。

很显然，这样的设计有点儿浪费。因为即便只调用一条非常简单的指令，我们也需要等待整个时钟周期的时间走完，才能执行下一条指令。在后面章节里我们会讲到，通过流水线技术进行性能优化，可以减少需要等待的时间，这里我们暂且说到这里。

## 读写数据所需要的译码器

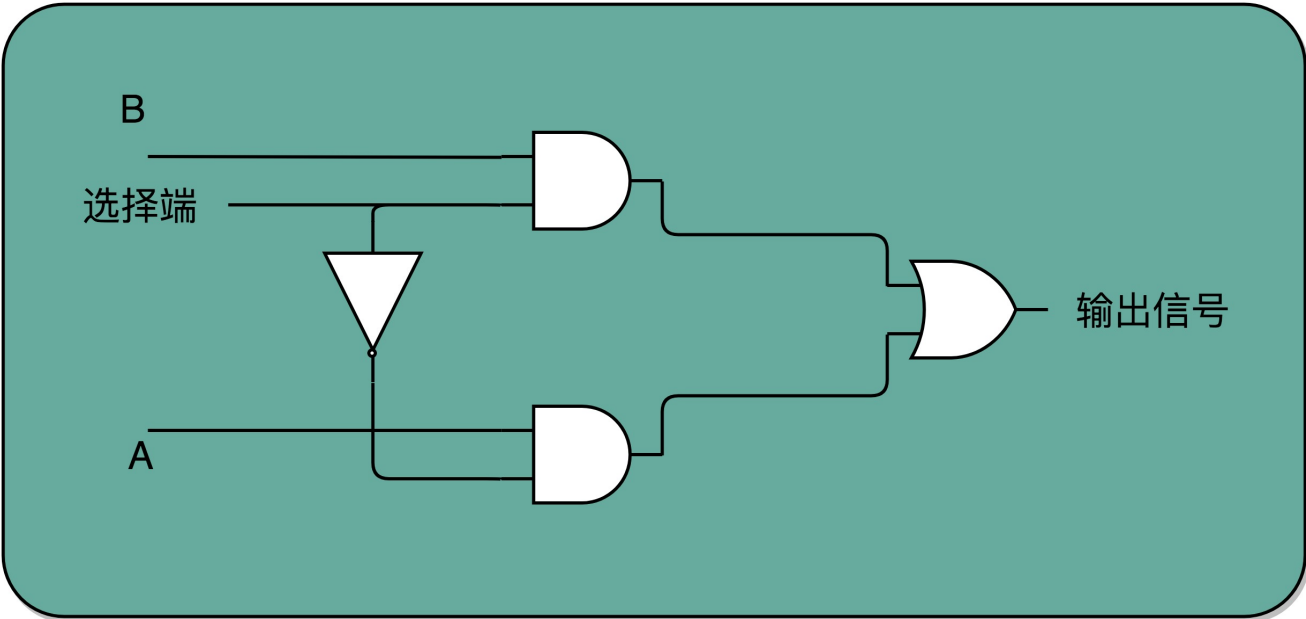
现在，我们的数据能够存储在 D 型触发器里了。如果我们把很多个 D 型触发器放在一起，就可以形成一块很大的存储空间，甚至可以当成一块内存来用。像我现在手头这台电脑，有 16G 内存。那我们怎么才能知道，写入和读取的数据，是在这么大的内存的哪几个比特呢？

于是，我们就需要有一个电路，来完成“寻址”的工作。这个“寻址”电路，就是我们接下来要讲的译码器。

在现在实际使用的计算机里面，内存所使用的 DRAM，并不是通过上面的 D 型触发器来实现的，而是使用了一种 CMOS 芯片来实现的。不过，这并不影响我们从基础原理方面来理解译码器。在这里，我们还是可以把内存芯片，当成是很多个连在一起的 D 型触发器来实现的。

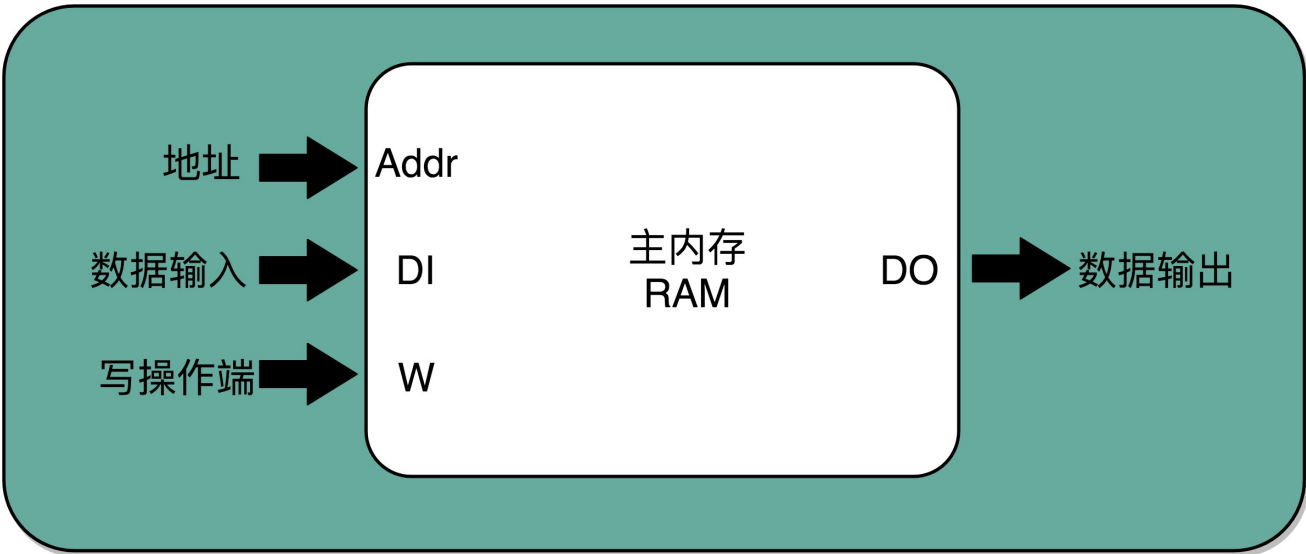
如果把“寻址”这件事情退化到最简单的情况，就是在两个地址中，去选择一个地址。这样的电路，我们叫作**2-1 选择器**。我把它的电路实现画在了这里。

我们通过一个反相器、两个与门和一个或门，就可以实现一个 2-1 选择器。通过控制反相器的输入是 0 还是 1，能够决定对应的输出信号，是和地址 A，还是地址 B 的输入信号一致。



2-1 选择器电路示意图

一个反向器只能有 0 和 1 这样两个状态，所以我们只能从两个地址中选择一个。如果输入的信号有三个不同的开关，我们就能从  $2^3$ ，也就是 8 个地址中选择一个了。这样的电路，我们就叫 **3-8 译码器**。现代的计算机，如果 CPU 是 64 位的，就意味着我们的寻址空间也是  $2^{64}$ ，那么我们就需要一个有 64 个开关的译码器。

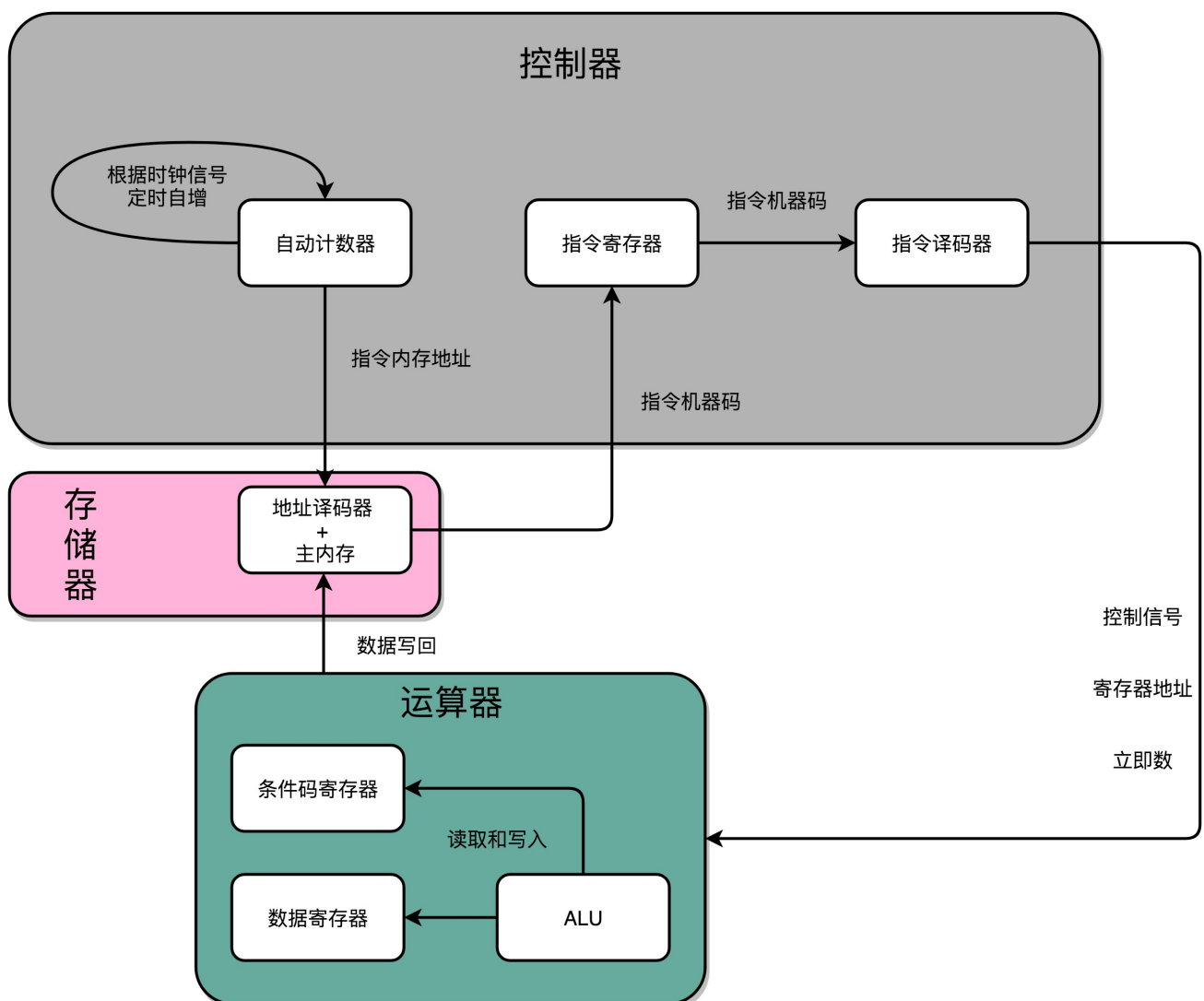


当我们把译码器和内存连到一起时，通常会组成这样一个电路

所以说，其实译码器的本质，就是从输入的多个位的信号中，根据一定的开关和电路组合，选择出自己想要的信号。除了能够进行“寻址”之外，我们还可以把对应的需要运行的指令码，同样通过译码器，找出我们期望执行的指令，也就是在之前我们讲到过的 opcode，以及后面对应的操作数或者寄存器地址。只是，这样的“译码器”，比起 2-1 选择器和 3-8 译码器，要复杂的多。

## 建立数据通路，构造一个最简单的 CPU

D 触发器、自动计数以及译码器，再加上一个我们之前说过的 ALU，我们就凑齐了一个拼装一个 CPU 必须要的零件了。下面，我们就来看一看，怎么把这些零件组合起来，才能实现指令执行和算术逻辑计算的 CPU。



CPU 实现的抽象逻辑图



1. 首先，我们有一个自动计数器。这个自动计数器会随着时钟主频不断地自增，来作为我们的 PC 寄存器。
2. 在这个自动计数器的后面，我们连上一个译码器。译码器还要同时连着我们通过大量的 D 触发器组成的内存。
3. 自动计数器会随着时钟主频不断自增，从译码器当中，找到对应的计数器所表示的内存地址，然后读取里面的 CPU 指令。
4. 读取出来的 CPU 指令会通过我们的 CPU 时钟的控制，写入到一个由 D 触发器组成的寄存器，也就是指令寄存器当中。
5. 在指令寄存器后面，我们可以再跟一个译码器。这个译码器不再是用来寻址的了，而是把我们拿到的指令，解析成 opcode 和对应的操作数。
6. 当我们拿到对应的 opcode 和操作数，对应的输出线路就要连接 ALU，开始进行各种算术和逻辑运算。对应的计算结果，则会再写回到 D 触发器组成的寄存器或者内存当中。

这样的一个完整的通路，也就完成了我们的 CPU 的一条指令的执行过程。在这个过程中，你会发现这样几个有意思的问题。

第一个，是我们之前在 [第 6 讲](#) 讲过的程序跳转所使用的条件码寄存器。那时，讲计算机的指令执行的时候，我们说高级语言中的 if...else，其实是变成了一条 cmp 指令和一条 jmp 指令。cmp 指令是在进行对应的比较，比较的结果会更新到条件码寄存器当中。jmp 指令则是根据条件码寄存器当中的标志位，来决定是否进行跳转以及跳转到什么地址。

不知道你当时看到这个知识点的时候，有没有一些疑惑，为什么我们的 if...else 会变成这样两条指令，而不是设计成一个复杂的电路，变成一条指令？到这里，我们就可以解释了。这样分成两个指令实现，完全匹配好了我们在电路层面，“译码 - 执行 - 更新寄存器”这样的步骤。

cmp 指令的执行结果放到了条件码寄存器里面，我们的条件跳转指令也是在 ALU 层面执行的，而不是在控制器里面执行的。这样的实现方式在电路层面非常直观，我们不需要一个非常复杂的电路，就能实现 if...else 的功能。

第二个，是关于我们在 [第 17 讲](#) 里讲到的指令周期、CPU 周期和时钟周期的差异。在上面的抽象的逻辑模型中，你很容易发现，我们执行一条指令，其实可以不放在一个时钟周期里面，可以直接拆分到多个时钟周期。

我们可以在一个时钟周期里面，去自增 PC 寄存器的值，也就是指令对应的内存地址。然后，我们要根据这个地址从 D 触发器里面读取指令，这个还是可以在刚才那个时钟周期内。但是对应的指令写入到指令寄存器，我们可以放在一个新的时钟周期里面。指令译码给到 ALU 之后的计算结果，要写回到寄存器，又可以放到另一个新的时钟周期。所以，执行一条计算机指令，其实可以拆分到很多个时钟周期，而不是必须使用单指令周期处理器的设计。

因为从内存里面读取指令时间很长，所以如果使用单指令周期处理器，就意味着我们的指令都要去等待一些慢速的操作。这些不同指令执行速度的差异，也正是计算机指令有指令周期、CPU 周期和时钟周期之分的原因。因此，现代我们优化 CPU 的性能时，用的 CPU 都不是单指令周期处理器，而是通过流水线、分支预测等技术，来实现在一个周期里同时执行多个指令。

## 总结延伸

好了，今天我们讲完了，怎么通过连接不同功能的电路，实现出一个完整的 CPU。

我们可以通过自动计数器的电路，来实现一个 PC 寄存器，不断生成下一条要执行的计算机指令的内存地址。然后通过译码器，从内存里面读出对应的指令，写入到 D 触发器实现的指令寄存器中。再通过另外一个译码器，把它解析成我们需要执行的指令和操作数的地址。这些电路，组成了我们计算机五大组成部分里面的控制器。

我们把 opcode 和对应的操作数，发送给 ALU 进行计算，得到计算结果，再写回到寄存器以及内存里面来，这个就是我们计算机五大组成部分里面的运算器。

我们的时钟信号，则提供了协调这样一条条指令的执行时间和先后顺序的机制。同样的，这也带来了一个挑战，那就是单指令周期处理器去执行一条指令的时间太长了。而这个挑战，也是我们接下来的几讲里要解答的问题。

## 推荐阅读

《编码：隐匿在计算机软硬件背后的语言》的第 17 章，用更多细节的流程来讲解了 CPU 的数据通路。《计算机组成与设计 硬件 / 软件接口》的 4.1 到 4.4 小节，从另外一个层面和角度讲解了 CPU 的数据通路的建立，推荐你阅读一下。

## 课后思考

CPU 在执行无条件跳转的时候，不需要通过运算器以及 ALU，可以直接在控制器里面完成，你能说说这是为什么吗？

欢迎在留言区写下你的思考和疑惑，你也可以把今天的内容分享给你的朋友，和他一起学习和进步。

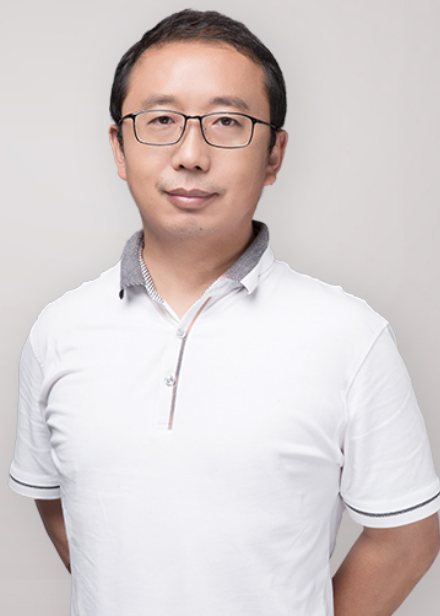
### 更多课程推荐


# 编译原理之美

手把手教你实现一个编译器

宫文学

北京物演科技CEO



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 18 | 建立数据通路（中）：指令+运算=CPU

下一篇 20 | 面向流水线的指令设计（上）：一心多用的现代CPU

## 精选留言 (18)

 写留言



小海海

2019-06-07

(手滑点了保存，接上条)无条件跳转意味着没有计算的逻辑，应该是可以不经过ALU的，但是要控制器把PC设置成跳转后的指令地址，不知我理解的对不对🤔



展开 ▾

作者回复: 小海海同学你好,

回答正确 😊



6



**鱼向北游**

2019-06-11

应该是无条件跳转没有cmp吧

展开 ▾

作者回复: 鱼向北游同学, 你好

cmp是一条单独的指令, 和跳转是否有条件的没有关系。无条件跳转不需要检查条件码寄存器哦。可以想想这样的话, 在硬件上有什么区别呢?



2



**小海海**

2019-06-07

@陈华应 我的理解无条件跳转就是goto, 是没有比较结果的

作者回复: 👍



2



**humor**

2019-06-10

一个指令的执行想用多少个时钟周期都可以吗?

展开 ▾

作者回复: humor同学你好,

一个指令需要的时钟周期是固定的, 并不是多少个都可以啊。

**陈华应**

2019-06-07

无条件跳转不需要依赖于比较结果吗？

展开 ∨

作者回复: 既然是“无条件”了，当然不依赖比较结果啦。



1

**wei**

2019-11-30

老师好，“加法器的两个输入，一个始终设置成 1，另外一个来自于一个 D 型触发器 A。我们把加法器的输出结果，写到这个 D 型触发器 A 里面。于是，D 型触发器里面的数据就会在固定的时钟信号为 1 的时候更新一次。” 这里我明白，为什么时钟信号为 1 的时候只会更新一次，而不是两次或更多次？

展开 ∨

**寒玉**

2019-11-07

一个周期执行多条指令和java里面的多线程意思上是一样的吗？

**Im Robin**

2019-10-31

老师你好，有一个问题很困惑，PC寄存器自动加1，我怎么感觉应该是自动加指令字节数啊，这样寄存器里才能存下一个指令的起始地址，如果PC寄存器里只是数值，那么程序如何从PC寄存器中取出指令地址呢，麻烦老师给予解答，谢谢

展开 ∨

**Im Robin**

2019-10-30

老师你好，看了好几遍也没理解下面这句，我理解时钟周期（机器时钟）是固定值，也就是1/主频，在这样的语境下，想实现一个时钟周期内执行一个任意的CPU指令，我感觉是不可能的事情，能否帮忙解惑，谢谢

在这种设计下，我们需要在一个时钟周期里，确保执行完一条最复杂的 CPU 指令，也就...

展开 ▾

💬 1



**随心而至**

2019-10-15

真的是深入浅出，赞

展开 ▾



**活的潇洒**

2019-09-01

“怎样通过连接不同功能的电路，实现出一个完整的CPU” 讲的真好  
day19 笔记: <https://www.cnblogs.com/luoahong/p/11431367.html>



**业余爱好者**

2019-08-27

流水线让我想起了大学时的一门选修课——项目管理

展开 ▾



**小先生**

2019-08-14

为什么我们的 if...else 会变成这样两条指令，而不是设计成一个复杂的电路，变成一条指令？到这里，我们就可以解释了。这样分成两个指令实现，完全匹配好了我们在电路层面，“译码 - 执行 - 更新寄存器”这样的步骤。

请问老师这段话，是什么意思，还是没看懂。

展开 ▾

💬 1



**张立昊Leon**

2019-07-01

还记得当年数电考试分还挺高的，不知道有啥用。后面就直接学c++，嵌入式系统之类的课程，跳过了中间的计算机组成，操作系统之类的东西，万恶的电气系学科体系...嵌入式系统当时学得完全不知所云，也极大的损害了我对这些东西的兴趣，今天在这里总算补学了一些重要的东西啊

展开 ▾





**Geek\_54edc1**

2019-06-25

因为无条件跳转不需要访问条件码寄存器，而且地址跳转就是一个直接寻址的过程，因此不需要通过运算器和ALU

展开 ▾



**-W.LI-**

2019-06-22

老师好!无条件调整，直接把跳转过去的指令跟在当前指令后面可以么?变成顺序的。  
D触发器感觉用处很大啊，存储相关的都可以通过它实现，寄存器的速度比内存快很多，和硬件实现有关系么比如D触发器比CMOS快?  
高速缓存使用触发器实现的还是CMOS啊?

展开 ▾



**-W.LI-**

2019-06-22

老师好，读取数据的那个寄存器就是。当前指令地址+指令长度。每次就能取到下一条指令了是吧?这里的内存地址都是虚存地址么?物理内存很容易就不连续吧。特别是那种动态申请内存的。还是说Java这样的编程语言，编译后的机器码就是在物理内存里顺序存储的?

展开 ▾



**许童童**

2019-06-07

《编码：隐匿在计算机软硬件背后的语言》这本书这几天看完了，写得真是很不错。希望老师再多推荐一些好看的书。

展开 ▾

