

03 | 汇编：编程语言的诞生

2019-04-23 许式伟

许式伟的架构课

[进入课程 >](#)



讲述：姚迪迈

时长 08:56 大小 8.19M



你好，我是七牛云许式伟。

在上一篇文章中，我们一起解剖了架构大厦的地基：冯·诺依曼体系。接下来，我们就开始沿着这座大厦攀登，一起来聊聊编程语言。

对于现代计算机来说，虽然 CPU 指令是一个很有限的指令集，但是 CPU 执行的指令序列（或者叫“程序”）并不是固定的，它依赖于保存在存储中的数据，由软件工程师（或者叫“程序员”）编写的软件决定。

从上一篇文章中，我们可以知道，计算机的程序可能被保存在计算机主板的 ROM 上（这段程序也叫计算机的启动程序），也可能被保存在外置的存储设备（比如硬盘）上，并在合适的时机加载执行。

程序称得上是计算机的灵魂。指令序列的可能性是无穷的，程序的可能性就是无穷的。今天计算机创造的世界如此多姿多彩，正是得益于程序无穷的可能性。

那么，软件工程师是怎么编写程序的？

编程的史前时代

在第一门面向程序员的编程语言出现前，人们只能通过理解 CPU 指令的二进制表示，将程序以二进制数据方式刻录到存储（比如 ROM 或硬盘）上。

这个时期的编程无疑是痛苦的，效率是极其低下的：且不说我们怎么去修改和迭代我们的程序，光将我们的想法表达出来就极其困难。

我们首先要把表达的执行指令翻译成二进制的比特数据，然后再把这些数据刻录到存储上。

这个时候软件和硬件的边界还非常模糊，并不存在所谓软件工程师（或者叫“程序员”）这样的职业。写程序也并不是一个纯软件的行为，把程序刻录到存储上往往还涉及了硬件的电气操作。

为了解决编程效率的问题，汇编语言（和解释它的编译器）诞生了。汇编语言的编译器将汇编语言写的程序编译成为 CPU 指令序列，并将其保存到外置的存储设备（比如硬盘）上。

汇编语言非常接近计算机的 CPU 指令，一条汇编指令基本上和 CPU 指令一一对应。

与机器对话

汇编语言的出现，让写程序（编程）成为一个纯软件行为（出现“程序员”这个分工的标志），人们可以反复修改程序，然后通过汇编编译器将其翻译成机器语言，并写入到外置的存储设备（比如硬盘）。并且，程序员可以按需执行该程序。

在表达能力上，汇编语言主要做了如下效率优化。

用文本符号（symbol）表达机器指令，例如 add 表示加法运算，而不用记忆对应的 CPU 指令的二进制表示。

用文本符号（symbol）表达要操作的内存地址，并支持内存地址的自动分配。比如我们在程序中使用了“Hello” 这样一段文本，那么汇编编译器将为程序开辟一段静态存储区

(通常我们叫“数据段”)来存放这段文本,并用一个文本符号(也就是“变量名 - variable”)指向它。用变量名去表达一段内存数据,这样我们就不用去关注内存的物理地址,而把精力放在程序的逻辑表达上。

用文本符号(symbol)表达要调用的函数(function,也叫“过程 - procedure”)地址。对 CPU 指令来说,函数只有地址没有名字。但从编程的角度,函数是机器指令的扩展,和机器指令需要用文本符号来助记一样,函数的名称也需要用文本符号来助记。

用文本符号(symbol)表达要跳转的目标地址。高级语言里面,流程控制的语法有很多,比如 goto、if ... else、for、while、until 等等。但是从汇编角度来说,只有两种基本的跳转指令:无条件跳转(jmp)和条件跳转(je、jne)。同样,跳转的目标地址用文本符号(也就是“标签 - label”)有助于程序逻辑的表达,而不是让人把精力放在具体的指令跳转地址上。

总结来说,汇编从指令能力上来说,和机器指令并无二致,它只不过把人们从物理硬件地址中解脱出来,以便专注于程序逻辑的表达。

但是,这一步所解放的生产力是惊人的,毕竟如果有选择的话,没有人会愿意用 0101 这样的东西来表达自己的思想。

可自我迭代的计算机

从探究历史的角度,你可能会期望了解最真实的历史发展过程。比如:怎么产生了现代计算机(以键盘作为输入,显示器作为输出)?怎么产生了汇编语言?怎么产生了操作系统?

不过,本专栏是以架构设计为目的,我们目的并不是还原最真实的历史。架构的意义在于创造。我们甚至可以设想一个有趣的场景:假设今天我们的信息科技的一切尚不存在,那么从架构设计角度,我们从工程上来说,如何更高效地完成从 0 到 1 的信息科技的构建?

最早的输入输出设备并不是键盘和显示器,而是打孔卡和打印机。用打孔卡来作为机器指令的输入,早在 18 世纪初就被用在织布机上了。早期的数字计算机就是用打孔卡来表达程序指令和输入的数据。

下图是 IBM 制造的打孔卡:

把程序的源代码写入磁盘，从磁盘读入的能力。

最早期的计算机毫无疑问是单任务的，计算的职能也多于存储的职能。每次做完任务，计算机的状态重新归零（回到初始状态）都没有关系。

但是，有了上面这样一个 BIOS 程序后，计算机就开始发展起它存储的能力：程序的源代码可以进行迭代演进了。

这一步非常非常重要。计算机的存储能力的重要性如同人类发明了纸。纸让人类存储了知识，一代代传递下去并不断演进，不断发扬光大。

而同样有了存储能力的计算机，我们的软件程序就会不断被传承，不断演进发扬光大，并最终演进到今天越来越多姿多彩的信息科技的世界。

结语

今天我们一起回到了编程的史前时代，共同回溯了编程语言诞生的历史。

为了不再用“0101”表达自己的思想，人们创造了汇编语言，这一步让编程成为一个纯软件行为，程序员这一个分工也由此诞生。

为了进一步支持程序员这个职业，我们设计了 MVP 版（最小化可行产品）的可自我迭代的计算机。有了这个计算机，我们就可以不断演进，并最终演进到今天越来越多姿多彩的信息科技的世界。

架构上的思考题

在上一篇文章中，我们谈架构思维时提到，我们在需求分析时，要区分需求的变化点和稳定点。稳定点往往是系统的核心能力，而变化点则需要对应地去考虑扩展性上的设计。

今天，我们假设要实现一个最小化的计算能力可自我迭代的计算机，需求如上所述。

那么，它的变化点和稳定点分别是什么？为此，你会怎么设计出哪些子系统，每个子系统的规格是什么？扩展性上有哪些考虑？

欢迎把你的想法告诉我，我们一起讨论。感谢你的收听，再见。

许式伟的架构课

从源头出发, 带你重新理解架构设计

许式伟
七牛云 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 02 | 大厦基石: 无生有, 有生万物

下一篇 04 | 编程语言的进化

精选留言 (69)

写留言



公号-代码...

2019-04-23

41

稳定点: 指令执行能力, 数据存储能力, 程序编辑能力, 程序编译能力

变化点: 不同CPU的指令集, 不同外设设备, 不同的编辑器, 不同程序的语法特性

子系统: 计算执行子系统, IO子系统, 编辑子系统, 编译子系统

可扩展性: 考虑微内核+插件的架构模式

展开 ∨



晓凉

2019-04-24

21

系统要与外部世界交互, 就应该有输入和输出的能力(黑洞是否只有输入没有输出); 作为人造系统, 主要的交互对象是人, 所以针对人类的输入输出能力就更重要, 显示器和键

盘就是这个角色；存储和计算能力是系统智能的核心，决定了系统如何用输入输出与外部世界交互；系统需要有演化能力，初期只能人类帮助演化（编程语言、编辑器、编译器），足够智能后才能自我演化。

展开 ▾

作者回复: 总结很赞



叶建盟(Ja...

2019-04-25

👍 11

我要做一个最小机器人系统，需要考虑需求的变化点和稳定点。该怎么考虑呢？

作者回复: 挺典型的问题。这个问法是一种典型的需求陈述误区。描述需求需要有几个典型的要素：1、用户，面向什么人群；2、他们有什么要解决的问题；3、我解决这个问题的核心系统。只有满足这几个要素的需求才能进一步讨论变化点和稳定点。最小机器人可能符合上面的3，但是用户人群和要解决的问题没有描述，也就无法进一步去思考到底哪些因素是稳定的，哪些是易变的。



姜戈

2019-04-23

👍 10

稳定点在于：计算能力；变化点在于各种输入输出方式（设备）

展开 ▾



Jay

2019-04-23

👍 5

自己的见解：

稳定点：

- 1.计算能力 cpu
- 2.存储能力 将数据写入已分配的位置
- 3.寻址能力 从硬盘和内存中找到变量指向的位置...

展开 ▾



思维

2019-04-23

👍 4

首先是一台计算机，所以包括了上一讲中计算机的稳定点和变化点。2可自我迭代，代码要可编辑，代码要可以存储起来，代码要方便开发人员阅读。要完成这几点 就需要设计编辑器 编译器。代码的命令要和机器指令对应。

展开 ∨



糖果屋

2019-04-23

👍 3

1: 需求

稳定点: 计算能力, 存储能力, 输入输出能力

变化点: 计算设备+指令集, 存储设备+驱动; 输入输出设备+驱动

2: 子系统: 计算子系统, 存储操作子系统, I/O子系统...

展开 ∨



yason

2019-04-23

👍 3

最小化的计算能力可自我迭代的计算机。

首先需要明确一些概念。最小化指具备 CPU、鼠标和键盘的现代计算机，计算能力描述的是指令序列，可自我迭代:指指令序列可修改，进一步推导就是要指令序列要可存储、可...

展开 ∨



hao

2019-05-06

👍 2

存储让数据跨越时间，（今天存储明天读取）传输让数据跨越空间，计算让数据改变形式。时空一体，那么存储和传输可能也是统一的

作者回复: 📬



宁静致远

2019-04-25

👍 2

许老师，自己现在已经工作快三年了，想往架构师这个方向走，但现在自己有些迷茫，接触到的技术也算挺多了，但不知道该如何入手架构师，之前您也提到过先广度然后深度，

但我想问达到什么算广度够了，怎么进行深度学习

展开 ∨

作者回复: 架构师核心是把知识串起来，构建一个完整的认知，不留疑惑。大部分知识是不需要深入细节的，只在你需要的时候深入，但深入的时候要很深

◀ ▶



JackJin

2019-04-23

👍 2

大佬，以前并没有架构方面的经验，以至于看您的文章，没有感觉了，就光看了一遍，脑子里没有引发共鸣，作为业界大佬的您，有什么好的指导与建议吗？

展开 ∨

作者回复: 根据文章的脉络，看看能不能串起来，加深了对这个信息世界如何构建的理解，如果没有，欢迎把问题抛出来讨论

◀ ▶



辉

2019-04-29

👍 1

汇编语言是机器思维与人思维桥梁。
存储让复杂任务成了可能，让机器经验像知识一样传承下去。

作者回复: 挺好的总结

◀ ▶



若飞

2019-04-27

👍 1

对 CPU 指令来说，函数只有地址没有名字。请问老师，这句话怎么理解??

展开 ∨

作者回复: cpu调用函数的指令是“call 函数地址”，而不是“call 函数名”。对cpu而言，函数没有函数名，这是高级语言为了逻辑表达方便而加的

◀ ▶



ljf10000

2019-04-24

👍 1

计算机能运行起来就是一个奇迹，本质上是一个个硬件组件单元之间的约定创造的奇迹，软件越做越大越复杂也是靠这个约定奇迹。



Aaron Che...

2019-04-23

👍 1

打卡 03 坚持fighting

展开 ▾



CoderLim

2019-04-23

👍 1

首先需要明确 mvp 的组成部分，根据这节课的内容，

mvp 组成部分应该是cpu、磁盘、内存、bios、汇编编译器、外设暂不考虑；

稳定点都知道是计算能力cpu，那么汇编编译器也应该是稳定点，因为汇编指令与cpu是...

展开 ▾



刘晓林

2019-04-23

👍 1

MVP版的可自我迭代计算机，编辑器和汇编器好像可以不用，因为有了存储驱动，编辑器和编译器可以放在存储介质中了，并且可以迭代变化。

稳定点:计算能力，存储(cpu与存储介质交互)能力，输入(键盘)能力，输出(显示器)能力，汇编语言(或者说汇编理论)。

变化点:汇编代码的具体内容，外置设备的具体类型

展开 ▾



My Cathe...

2019-06-03

👍

计算能力可迭代的计算机，需求应该是计算，就是执行一个有输入的函数，得到输出。那么稳定能力要有cpu的执行计算，读取外部输出输出设备，读取外部存储的能力。

为了不断迭代，那么就需要计算机有更多记忆的积累，记录或删除中间变量，记录或修改基本函数，这些都需要存储在外部存储设备，计算机还需要在开机后知道这些数据(函数和数据)放在哪些位置，这些存放位置不太能放在cpu内部存储中，这里的能力可以变化迭...

展开 ▾



学习

2019-05-22



根据冯诺依曼体系原理，必然推导出计算机最基本的功能是：

- 1.驱动输入输出设备，达到可用达态。
- 2.驱动存储设备，达到可用状态。
- 3.编排待计算任务并保存。
- 4.翻译成机器码并保存。...

展开 ∨



L

2019-05-19



稳定点：计算能力，输入输出

变化点：计算逻辑及需要的数据（包括文本，键盘，鼠标，硬盘等）