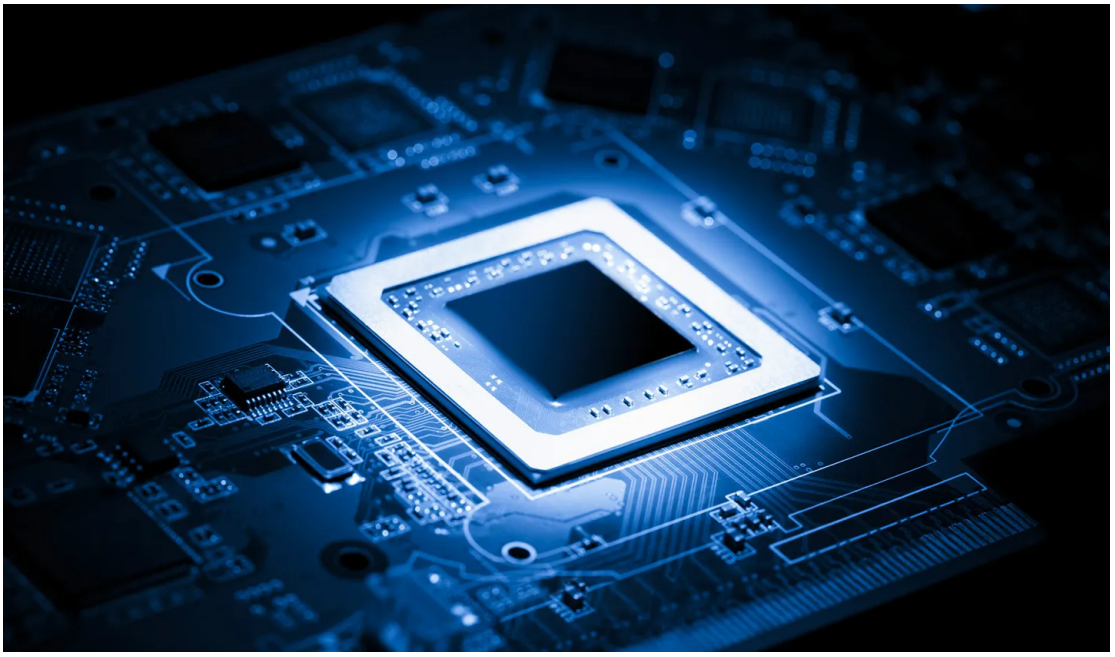


处理器架构与指令集

BruceOu
设计师

11 人赞同了该文章

大家天天都在使用手机，你知道你的手机使用的什么处理器？处理器又是何种架构呢？今天笔者就来谈谈处理器的架构和指令集。

我们知道一台手机最重要的就是处理器，也就是处理器，那么什么是处理器呢？

处理器就是一堆数字电路（架构）以高低电平的各种组合实现了各种基本的运算（指令）。

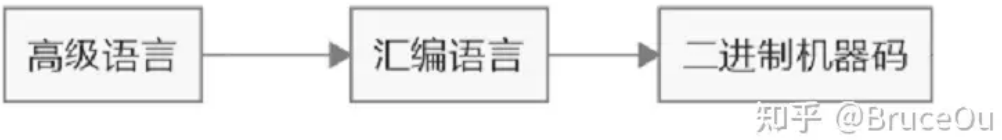
接下来我们看看要想设计出处理器，需要哪些东西，先看下图。

要想设计出处理器，就需要有指令集，也就是规定处理器的相应操作，通过指令集去控制处理器实现相应功能。但处理器是一堆硬件电路，只能识别二进制数据，所以指令集是由一堆二进制数据组成。处理器通过指令集是给用户提供的一个接口。指令集架构(Instructuon Set ArchitectureI, ISA)和处理器架构(Micro architecture)是上下级的关系。

而二进制数据对人类来说读起来很麻烦。为了方便人类操作指令集，发明了汇编语言来描述指令集。汇编语言类似人类语言，读起来方便多了。虽然汇编语言读起来方便了，但也有缺陷。首先汇编语言操作起来还是挺麻烦的。其次汇编语言对应一条条指令集，所以当指令集改变时，就得修改

相应汇编语言，导致其可移植性很差，不能跨平台使用，如ARM架构的汇编语言与Intel X86架构的就不同。这时人们就想开发一种更方便操作，超越指令集的语言，于是有了C，C++等高级语言。

处理器只能识别二进制码，那怎么能识别高级语言呢？于是人们开发了编译器，依照如下顺序，将高级语言翻译成二进制码：

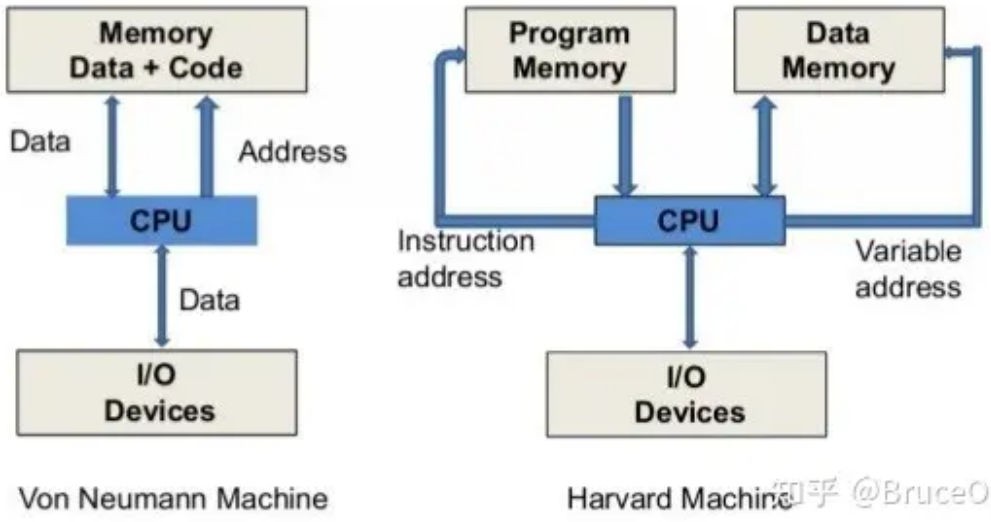


知乎 @BruceOu

至此，人类就可以方便的利用高级语言编写程序，控制处理器完成相应工作。然后程序员这个职业就此大规模诞生了。哈哈，惊不惊喜！好了，接下来我们谈谈处理架构和指令集。

1冯·诺依曼架构与哈佛架构

对于处理器架构，目前已经演变出很多架构，但基本上都是由**冯·诺依曼架构和哈佛架构**演变而来。



知乎 @BruceOu

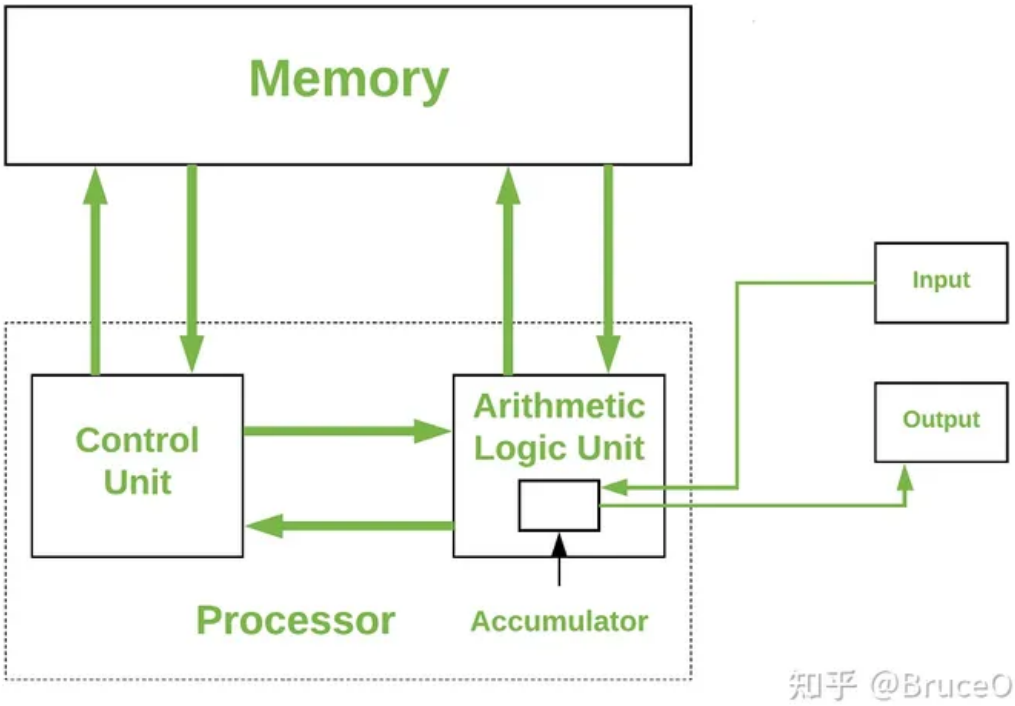
1.1冯·诺伊曼架构

1945 年，约翰·冯·诺依曼 (John von Neumann) 在他的著名《First Draft of a Report on the EDVAC》提出了电子数字计算机，也就是冯·诺伊曼架构。

冯·诺伊曼架构(Von Neumann Architecture)，也称普林斯顿结构，是一种将**程序指令存储器和数据存储器合并在一起的处理器架构**。

该架构将储存装置与中央处理器分开的概念，因此依该架构设计出的计算机又称**储存程式型电脑**。

冯诺依曼架构如下图所示：



1.CPU：CPU是负责执行计算机程序指令的电子电路单元。CPU是处理指令的计算机系统的主要组件。它运行操作系统和应用程序，不断接收来自用户或应用程序的输入。

2.寄存器：它是一种存储器，用于快速接收、存储和传输中央处理器立即使用的数据和指令。

3 . ALU： Arithmetic Logic Unit 负责所有所需的计算，如乘法、加法、减法、比较、逻辑运算符和其他算术运算符。

4.控制单元（CU）： CU也是中央处理器（CPU）的一部分。CU 操作系统内的所有处理器控制信号、输入/输出设备、数据和指令的移动。

5.总线：用作指令和数据的信号通信。

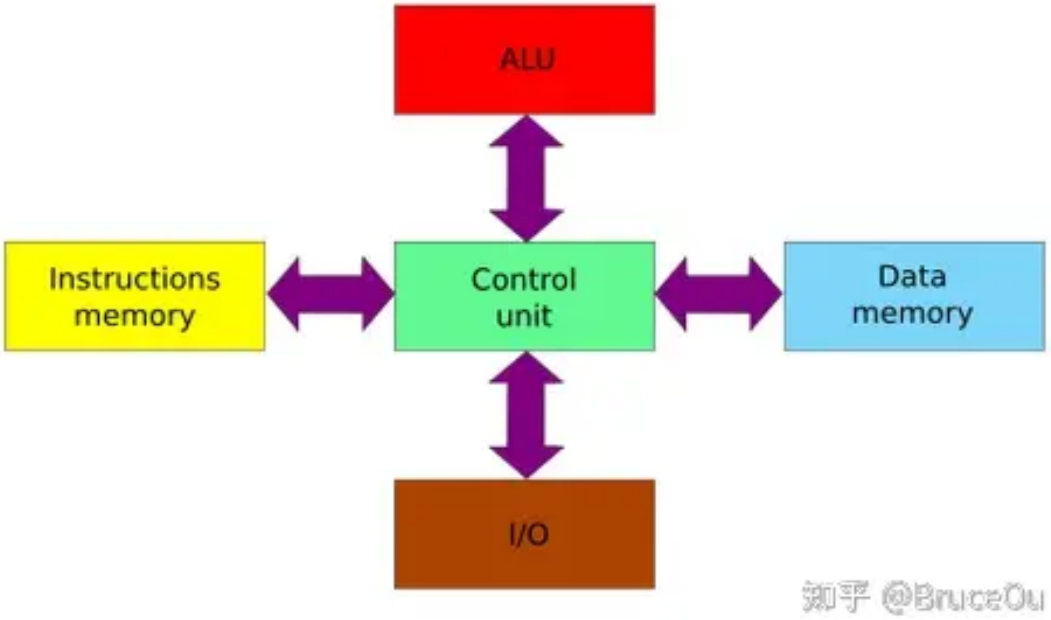
6.内存单元：内存单元由随机存取内存 (RAM) 组成，有时也称为主内存或主内存。与硬盘（二级内存）不同，这种内存速度快，而且 CPU 可以直接访问。RAM 被分成多个分区。它的每个分区都由一个地址及其内容组成。该地址将唯一标识内存中的每个位置。将数据从永久内存（硬盘）加载到更快且可直接访问的临时内存 (RAM) 中，可使 CPU 工作得更快。

PC和服务器芯片（譬如Intel AMD那些出的），ARM Cortex-A系列嵌入式芯片（譬如核心是ARM Cortex A8的三星S5PV210，譬如华为的麒麟970等手机芯片）等都是冯诺依曼结构。这些系统都需要大量内存，所以工作内存都是DRAM，因为他们更适合使用冯诺依曼系统。

1.2哈佛架构

哈佛架构(Harvard architecture)是一种**将程序指令储存和数据储存分开的存储器结构**。中央处理器首先到程序指令储存器中读取程序指令内容，解码后得到数据地址，再到相应的数据储存器中读取数据，并进行下一步的操作（通常是执行）。

哈佛架构的架构如下：



1.总线：用作指令和数据的信号通路。在这种架构中，数据和指令都有单独的总线。有不同类型的总线，它们如下：数据总线、数据地址总线、指令总线、指令地址总线。

2.操作寄存器：在哈佛架构中，使用了不同类型的寄存器，用于存储不同类型指令的地址。一些寄存器是内存数据寄存器，内存地址寄存器。

3.程序计数器：程序计数器具有要执行的下一条指令的位置。程序计数器将下一个地址传递给内存地址寄存器。

4. ALU：Arithmetic Logic Unit 负责所有所需的计算，如乘法、加法、减法、比较、逻辑运算符和其他算术运算符。

5.控制单元（CU）：CU也是中央处理器（CPU）的一部分。CU 操作系统内的所有处理器控制信号、输入/输出设备、数据和指令的移动。

6.输入/输出流：I/P设备用于在中央处理器（CUP）输入指令的帮助下将数据读入主存储器。当输出通过输出设备显示时，它提供来自系统的信息。简单地说，计算机系统在 O/P 设备的帮助下给出处理（计算）的结果。

在遵循冯诺依曼架构的普通计算机中，指令和数据都存储在同一内存中。因此使用相同的总线来获取指令和数据。这意味着 CPU 不能同时做两件事（读指令和读/写数据）。而哈佛体系结构的计算机将指令和数据分别使用独立存储和独立总线（信号路径）。它基本上是为了克服冯诺依曼架构的瓶颈而开发的。**指令和数据总线分开的主要优点是 CPU 可以同时访问指令和读/写数据**。这是哈佛架构的主要优势。

哈佛结构的微处理器通常具有较高的执行效率，因其程序指令和数据指令分开组织和储存的，执行时可以预先读取下一条指令。目前使用哈佛结构的中央处理器和微控制器有很多，比如Microchip公司的PIC系列芯片，还有摩托罗拉公司的MC68系列、Zilog公司的Z8系列、ATMEL公司的AVR系列和ARM公司的ARM9、ARM10和ARM11。MCU也就是单片机中几乎都是用哈佛结构，譬如广泛使用的51单片机、典型的STM32单片机（核心是ARM Cortex-M系列的）都是哈佛结构。

与冯.诺曼结构处理器比较，哈佛结构处理器有两个明显的特点：

1.使用两个独立的存储器模块，分别存储指令和数据，每个存储模块都不允许指令和数据并存；

2.使用独立的两条总线，分别作为CPU与每个存储器之间的专用通信路径，而这两条总线之间毫无关联。

1.3总结

哈佛结构和冯诺依曼结构主要**区别在是否区分指令与数据**。在教科书里这是两种截然不同的做法。

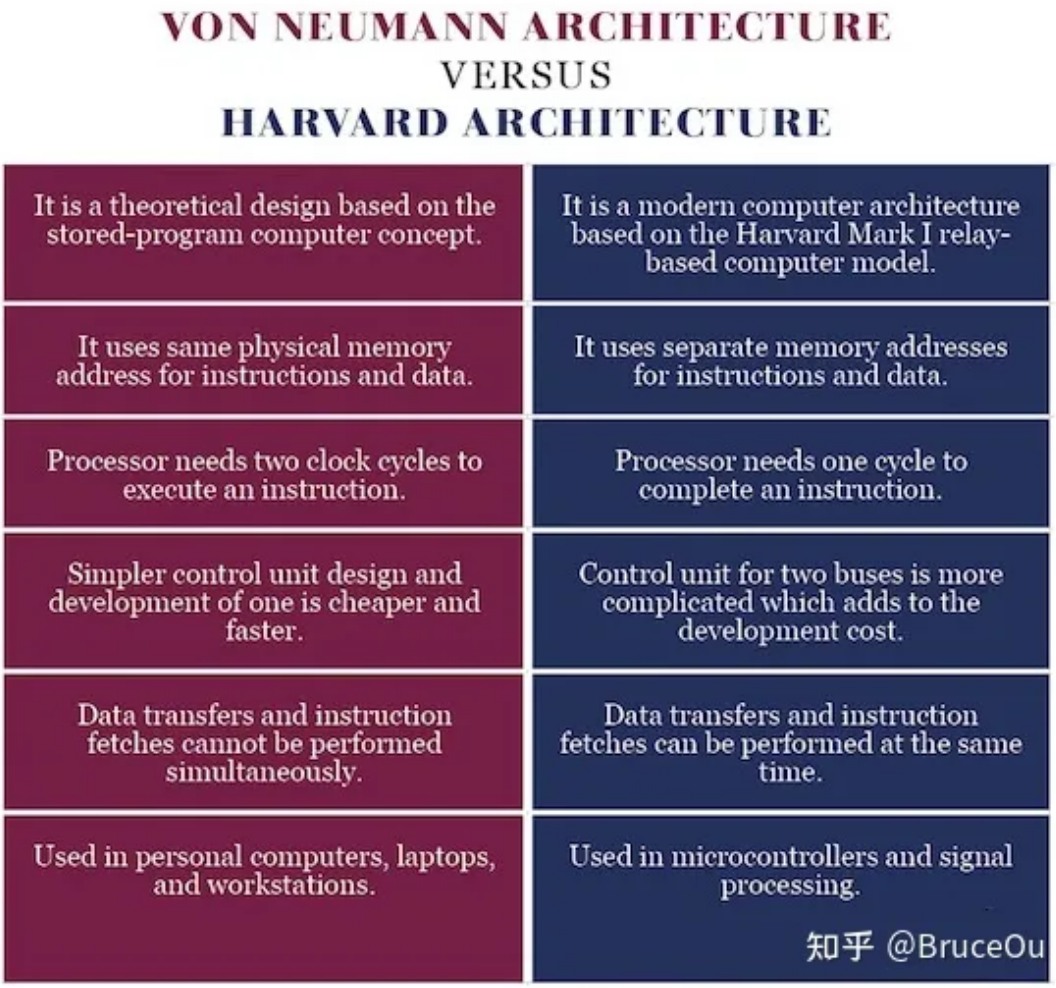
但实际上在内存里，指令和数据是在一起的。而在CPU内的缓存中，还是会区分指令缓存和数据缓存，最终执行的时候，指令和数据是从两个不同的地方出来的。你可以理解为在CPU外部，采用的是冯诺依曼模型，而在CPU内部用的是哈佛结构。

哈佛结构设计复杂，但效率高。冯诺依曼结构则比较简单，但也比较慢。CPU厂商为了提高处理速度，在CPU内增加了高速缓存。也基于同样的目的，区分了指令缓存和数据缓存。有时为了解决现实问题，究竟是什么主义真的没那么重要。

实际上，绝大多数现代计算机使用的是所谓的“Modified Harvard Architecture”，指令和数据共享同一个address space，但缓存是分开的。

在现实世界中很少有非常纯粹的概念，特别是在实际的应用里。教科书里的大多是理想化的模型，便于掌握某个概念的重点和本质，但实际中很难达到这种理想化的状态。

有一些ARM（Cortex-M系列）是哈佛结构，而另一些ARM（Cortex-A）是冯诺依曼结构（或者更准确说是混合结构），对ARM体系CPU(除ARM7)对外表现为冯.诺伊曼架构，对内则表现为哈佛架构。



因此，两种架构各有优势，不能一概而论。就目前而言，两种现代处理器在大规模的处理都是冯诺依曼，在小规模的处理是哈佛。

介绍完了处理器架构，接下来说说指令集。

2 CISC与RISC指令集

CISC(Complex Instruction Set Computers，复杂指令集计算机)和RISC(Reduced Instruction Set Computers，精减指令集计算机)是两大类主流的CPU指令集类型。

2.1 CISC（复杂指令集计算机）

早期的CPU全部是采用CISC，它的设计目的是要用最少的机器语言指令来完成所需的计算任务。

随着集成电路的发展，特别是VLSI（超大规模集成电路）技术的迅速发展，为了软件编程方便和提高程序的运行速度，硬件工程师采用的办法是**不断增加硬件的复杂性来实现复杂功能的指令和多种灵活的编址方式**，甚至某些指令可支持高级语言语句归类后的复杂操作，至使硬件越来越复杂，造价也相应提高。为实现复杂操作，微处理器除向程序员提供类似各种寄存器和机器指令功能外，还通过存于只读存贮器(ROM)中的微程序来实现其极强的功能，处理器在分析每一条指令之后执行一系列初级指令运算来完成所需的功能，以这种指令集设计出来计算机就是CISC(complex instruction set computer)，其中以**CISC**架构设计的处理器有以Intel、AMD的X86为代表。

2.2 RISC（精简指令集计算机）

采用复杂指令系统的计算机有着较强的处理高级语言的能力，这对提高计算机的性能是有益的。当计算机的设计沿着这条道路发展时，有些人没有随波逐流，他们回过头去看一看过去走过的道路，开始怀疑这种传统的做法，IBM公司在纽约Yorktown的JhomasI.Wason研究中心于1975年组织力量研究指令系统的合理性问题，因为当时已感到，日趋庞杂的指令系统不但不易实现，而且还可能降低系统性能。

1979年以帕特逊教授为首的一批科学家也开始在美国加册大学伯克莱分校开展这一研究。结果表明，CISC存在许多缺点。首先，在这种计算机中，各种指令的使用率相差悬殊：一个典型程序的运算过程所使用的80%指令，只占一个处理器指令系统的20%，事实上最频繁使用的指令是取、存和加这些最简单的指令。这样一来，长期致力于复杂指令系统的设计，实际上是在设计一种难得在实践中用得上的指令系统的处理器。同时，复杂的指令系统必然带来结构的复杂性。这不但增加了设计的时间与成本还容易造成设计失误。

此外，尽管VLSI技术现在已达到很高的水平，但也很难把CISC的全部硬件做在一个芯片上，这也妨碍单片计算机的发展。在CISC中，许多复杂指令需要极复杂的操作，这类指令多数是某种高级语言的直接翻版，因而通用性差。由于采用二级的微码执行方式，它也降低那些被频繁调用的简单指令系统的运行速度。

因而，针对CISC的这些弊病。帕特逊等人提出了精简指令的设想即指令系统应当只包含那些使用频率很高的少量指令，并提供一些必要的指令以支持操作系统和高级语言。按照这个原则发展而成的计算机被称为**精简指令集计算机(Reduced Instruction Set Computer)**，简称RISC。RISC以ARM、Apple的Macintosh、IBM Power为代表。开源的RISC-V也是RISC指令集。

RISC的设计初衷针对CISC CPU复杂的弊端，选择一些可以在单个CPU周期完成的指令，以降低CPU的复杂度，将复杂性交给编译器。

2.3 CISC和RISC区别

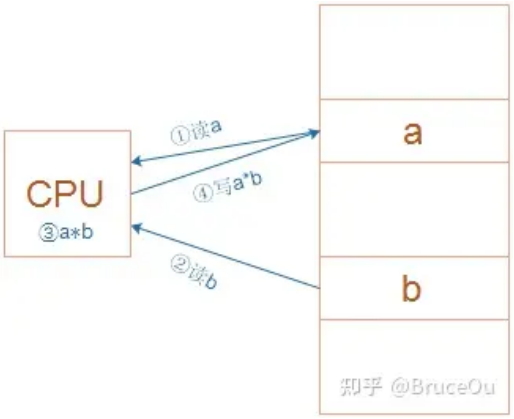
为了说明两者的区别，先举一个例子。下图是实现乘法运算的计算步骤：a = a * b。它需要4个步骤：

A.读出a的值

B.读出b的值

C.相乘

D.写结果到a中



使用CISC提供的乘法指令，只需要一条指令即可完成这4步操作。当然，这一个指令需要多个CPU周期才可以完成。

而RISC不提供“一站式”的乘法指令，需调用四条单CPU周期指令完成两数相乘：内存a加载到寄存器，内存b加载到寄存器，两个寄存器中数相乘，寄存器结果存入内存a。

按照此思路，早期的设计出的RISC指令集，指令数是比CISC少些。后来，很多RISC的指令集中指令数反超了CISC。因此，应该根据指令的复杂度而非数量来区分两种指令集。

当然，CISC也是要通过操作内存、寄存器、运算器来完成复杂指令的。它在实现时，是将复杂指令转换成了一个微程序，微程序在制造CPU时就已存储于微服务存储器。一个微程序包含若干条微指令（也称微码），执行复杂指令时，实际上是在执行一个微程序。这也带来两种指令集的一个差别，微程序的执行是不可被打断的，而RISC指令之间可以被打断，所以理论上RISC可更快响应中断。

总的来说，区别如下：

1.指令能力：

CISC的指令能力强，单多数指令使用率低却增加了CPU的复杂度，指令是**可变长格式，**它必须对不等长指令进行分割，因此在执行单一指令的时候需要进行较多的处理工作。

RISC的指令大部分为**单周期指令，指令长度固定，**CPU在执行指令的时候速度较快且性能稳定。因此在并行处理方面RISC明显优于CISC，RISC可同时执行多条指令，它可将一条指令分割成若干个进程或线程，交由多个处理器同时执行。RISC对内存只有load/store操作，数据的运算都是在CPU内部实现。

2.寻址方式：

CISC支持多种寻址方式。

RISC支持的寻址方式少。

3.实现方式：

CISC通过微程序控制技术实现(微码)。

RISC增加了通用寄存器，硬布线逻辑控制为主，适合采用流水线方式执行。RISC可以优化编译，有效支持高级语言。

4.研发周期：

CISC的研制周期长。

RISC硬件简单，因而它的制造工艺简单且成本低廉。

欢迎访问我的网站：

BruceOu的哔哩哔哩：BruceOxl的个人空间_哔哩哔哩_Bilibili

BruceOu的主页：BruceOu的主页

BruceOu的博客：BruceOu的博客 - Stay Hungry. Stay Foolish！求知若渴，虚心若愚！

BruceOu的CSDN博客：不问归期的博客_Bruceoxl_

BruceOu的简书：BruceOu - 简书

BruceOu的知乎：BruceOu

发布于 2021-08-31 22:42

精简指令集（RISC） 复杂指令集（CISC） 汇编语言

写下你的评论...

2 条评论 默认 最新

Venity Gshidra

大佬能不能详细讲解下为啥RISC的硬件简单。🤔

2022-05-31 回复 喜欢

君子不渡

个人感觉可以这么理解，举个例子，假设你要计算1+2*3，复杂指令集是把这个整个计算任务作为一条完整指令（a+b*c）,因此你同时读取三个数，选择使用两个计算单元来实现两个步骤的计算，或者设计更复杂的电路使得一次性完成乘和加两个步骤，另外如果是1+2/3则复杂指令集还需要编写实现一条指令计算（a+b/c），因此电路又变得更复杂。但精简指令集可以把这些任务一律拆成加、减、乘、除因此电路和元件都会更简单🤔

2022-11-06 回复 1

推荐阅读

处理器架构和处理器指令集以及汇编语言关系

，我们知道，计算机是由5大部件组成，分别是输入设备，主存，运算器（算数逻辑单元），控制器（控制单元），输出设备，但是如何让这些设备发挥作用，按照我们的想法让其做出对应的操作呢，…

Sunny 发表于技术以及生...

关于CPU、指令集、架构、芯片的一些科普

王强 发表于没有问题的...

处理器指令集，不是一本武功秘籍

01 如果一个江湖侠客功夫很菜，不用问，肯定缺少一本武功秘籍。《倚天屠龙记》中，张无忌开场也很菜，被玄冥二老虐，被何太冲虐，甚至被殷离虐，但是自从被朱长龄逼的跳下山崖，意外在白猿…

歪睿老哥 发表于歪睿老哥和...