计算机组成 Lecture Notes 2

本文主要概述计算机的发展历史、重要的计算机结构，以及计算机性能的衡量尺度。

计算机发展历程

第一代：真空管计算机

ENIAC 是世界上第一台通用电子数字计算机。

- 它是一台十进制机器，即数字采用十进制表示。
- 它使用真空管存储数字。
- 它的编程方式非常原始，需要人为设置开关、插拔电缆头来实现。

IAS 也属于这一代，见后。

UNIVAC I 是第一台成功的商业计算机。

IBM 701 是其首台应用了存储程序概念的计算机。

第二代：晶体管计算机

- 1947 年晶体管在贝尔实验室被发明。
- 电子计算机的第一个主要改变是使用晶体管替代电子管，晶体管的使用是第二代计算机的标志。
- 这一代存储用的是磁环。
- 这个时候出现了数据设备公司（DEC）。

第三代：集成电路

- 1958 年集成电路被发明，它定义了第三代计算机。
- 这一代存储改用芯片。
- 芯片由半导体产业生产，原料是薄硅晶片（wafer）。芯片上有大量的晶体管。

集成电路规模发展非常快。曾经有**摩尔定律（Moore's Law）**：单芯片上晶体管数目每 18-24 个月翻倍，性能也将因而翻倍。现在其已难以维持。

摩尔定律的影响：

- 密度上升则电路缩短，速度提升。
- 芯片体积缩小，更灵活。
- 降低了功耗和散热需求。
- 更少的相互连接提升了可靠性。

现在：超大规模集成电路

现在的 VLSI（超大规模集成电路，Very Large Scale Integration）可以做到一个芯片上有十万，乃至一亿个晶体管。

现代计算机的基础结构

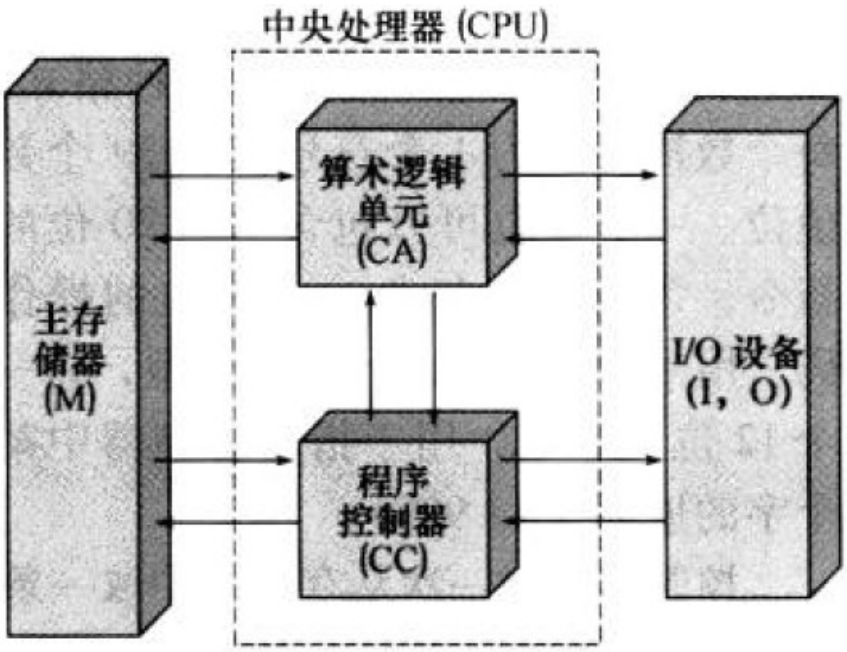
在 ENIAC 诞生后不久，冯诺依曼就提出了冯诺依曼结构。现在的通用计算机基本都还采用的是这种结构。

还有另一种结构叫做哈佛结构，它还有一种改进版本。

冯诺依曼结构

普林斯顿高等研究院设计的 IAS Computer 是以此架构为代表的计算机。因此，冯诺依曼结构也被称为普林斯顿结构。

IAS 的一般结构如图所示。

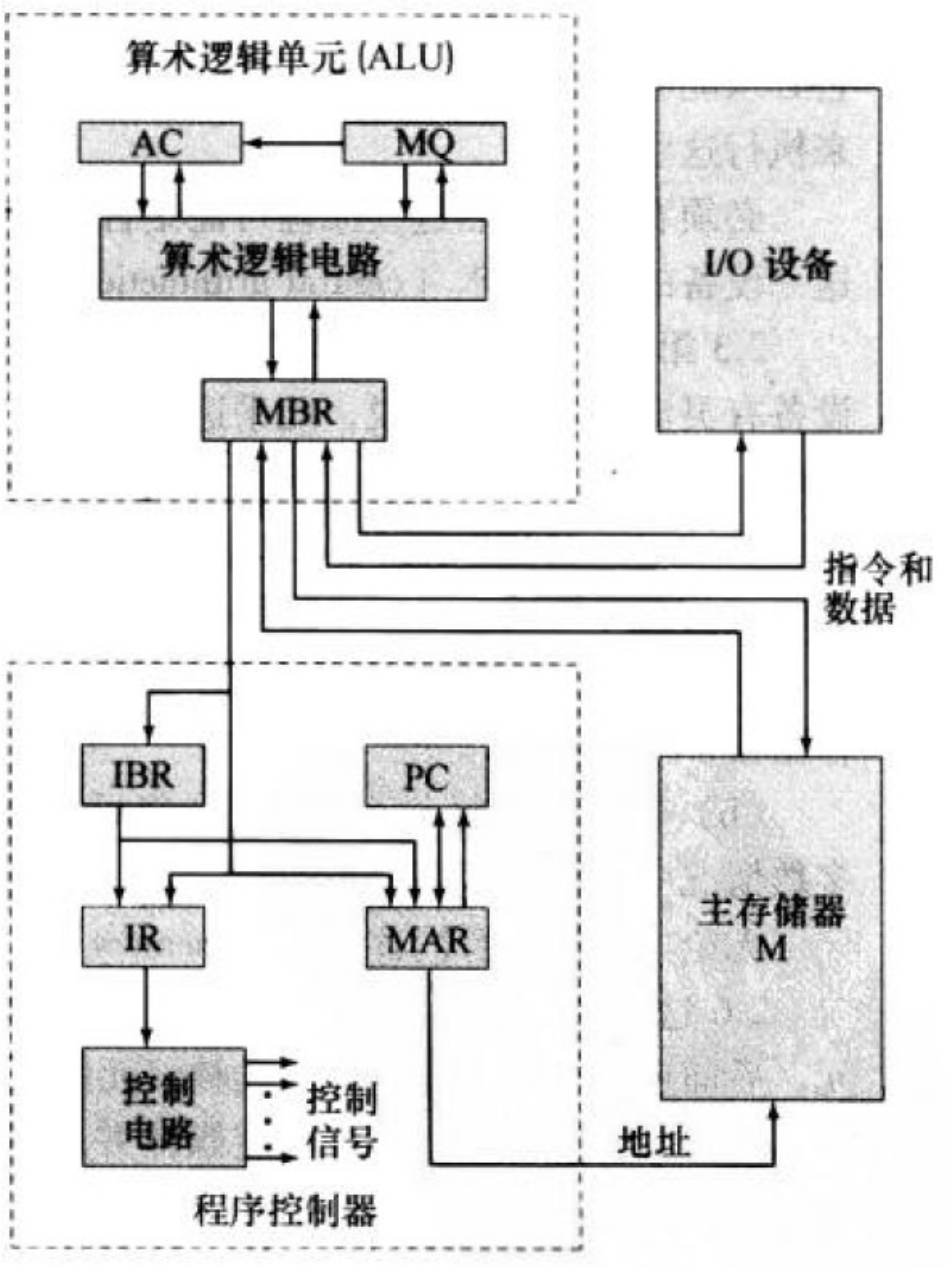


各个部分：

- 主存储器：IAS 包含 1000 个长为 40 bit 的存储单元，称为字。每个存储单元有一个长为 12 位的二进制数用于标定位置，称为地址。一个字可以保存一个数据或者两个指令。相关的*存储程序概念*会在后面提及。
- CA：处理二进制数据。
- CC：解释指令并执行。

> 对于 I/O 设备，输入或者输出是站在 CPU 的视角来看的，数据传入 CPU 就是输入，反之就是输出。这是默认的看待方式，后面也会提及。

为了解释指令的执行，需要更详细的结构，如下图所示。



各个部分：

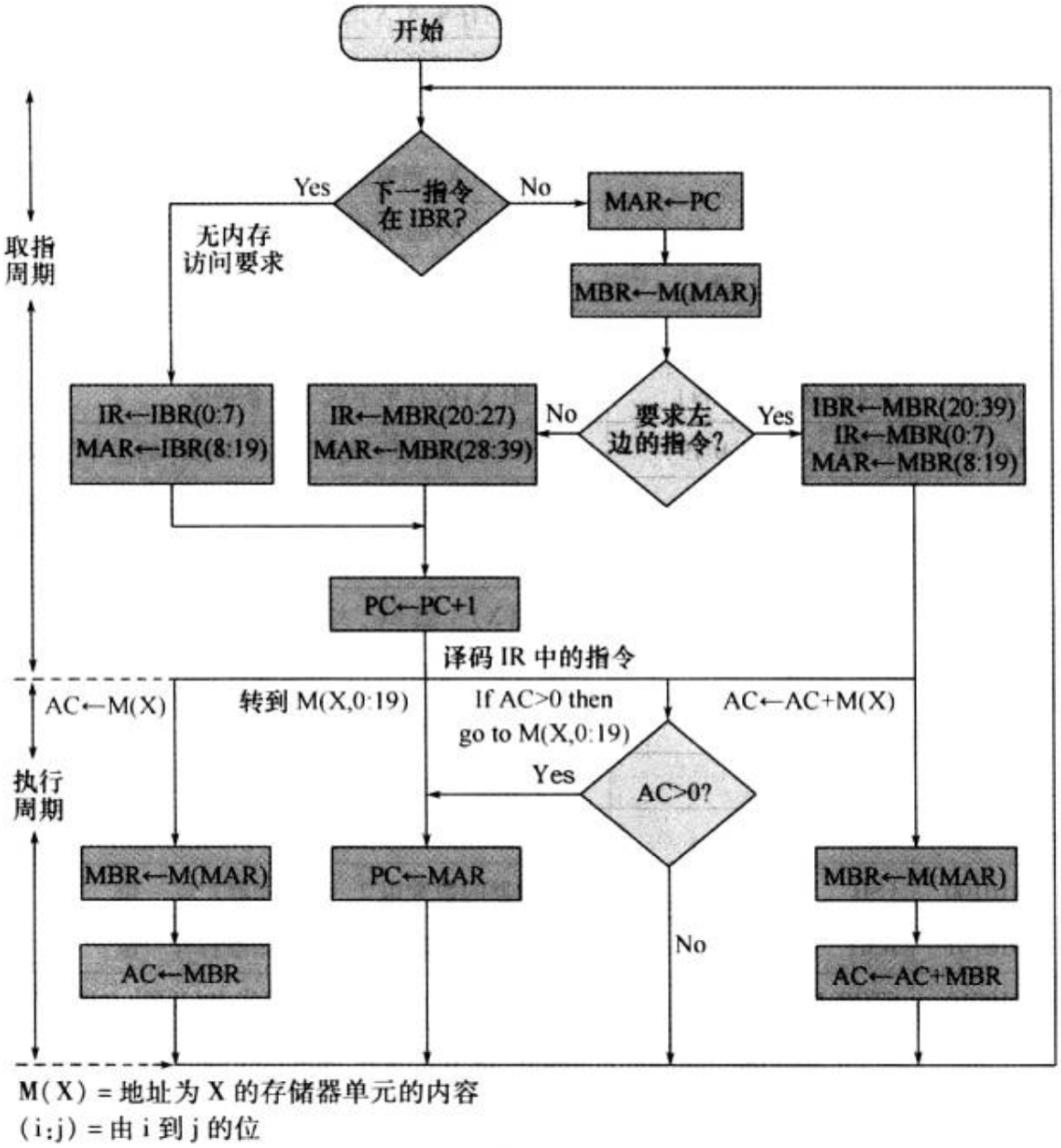
- AC，MQ：用来暂存 ALU 的操作数和结果。加减的结果放在 AC 中，乘除的结果高位在 AC，低位在 MQ。
- MBR（存储器缓冲寄存器，Memory Buffer Register）：包含将要接受自/写入到存储器或 I/O 的一个字。
- MAR（存储器地址寄存器，Memoty Address Register）：包含将要从 MBR 写入存储器，或者从存储器读到 MBR 的字的地址。
- IR（指令寄存器，Instruction Memory）：包含正在执行的指令的 *8 位操作码*。
- IBR（指令缓冲寄存器，Instruction Buffer Memory）：用来暂存一个字中右半部分指令。相当于一级缓存。
- PC：保存即将取出的下一对指令的地址。

这些寄存器都是有锁存（latch）功能的，也就是可以保持数据不变。一般记住部件名字就大概知道它们的用处。

从图中大致可以看出：从 M 传指令传到 CU 的数据通路是 PC -> MAR -> M -> MBR -> IBR -> IR。

> 这部分在详细了解了 CPU 后可能会有更深刻的认识。

IAS 通过反复运行指令周期（Instruction Cycle）来运行。一个指令周期分为取指周期（装载指令入 IR）和执行周期（CU 译码并产生控制信号）。流程图大致如图所示。



这样一种结构的优秀之处在于它的通用性，在上面编写程序不需要改变硬件，而只需要改变存储器中的内容，并在译码时为指令生成控制信号。

哈佛结构

哈佛结构和冯诺依曼结构很相近，差别在于它将指令存储器和数据存储器分开来了，每个存储器独立编址、独立访问，目的是为了减轻程序运行时的访存瓶颈。

原始哈佛结构对于两个存储器各有一条地址总线和一条数据总线，改进版哈佛结构只有公用的一条地址总线和一条数据总线。

性能评价

在考虑性能时，可以从两个方面考虑：

- 响应时间（或**执行时间，Execution Time**），即运行某任务所需时间。
- 吞吐量（Throughput），即单位时间所能做任务量。

可以量化**表现（Performance）**为执行时间的倒数。因此如果说 X 的表现是 Y 的 n，那么就是说 X 的表现是 Y 的 n。

> 这里 n 可以理解成 n 倍，但为了避免类似“快 1 倍是 2 倍”的歧义不这么写。

执行时间有多种定义。

- 最直观的定义是**持续时间（Elapsed Time）**，即一个系统从执行任务开始到结束，现实世界经过的时间。它包含了系统内所有的因素。
- 另一种定义是 CPU 时间，即用于处理任务的时间，包括**用户 CPU 时间**（用于用户程序的事件）和**系统 CPU 时间**（操作系统为用户服务花去的时间）。它不包含外部因素（如 I/O）的时间以及多任务切换时，花费在别的用户的任务上的时间。

我们这里主要考虑 CPU 时间。

CPU 运转需要系统时钟驱动，各种 CPU 的操作都随着时钟的脉冲开始。一般情况下，时钟信号由水晶振子产生。

产生脉冲的速率为**时钟频率（Clock Frequency）**或**时钟速度（Clock Rate）**。相应的有**时钟周期（Clock Period）**。

时钟速率不可能任意的快，原因：

- 物理定律有所限制。
- 信号变化需要一定时间稳定。
- 信号传输速率可能不同，但操作必须同步进行。

CPI

定义**CPI（每条指令的平均周期数，Average Cycles Per Instruction）**如下：

$$CPI = \frac{\sum_i CPI_i \times I_i}{I_c}$$

其中：

- I_c 为运行该程序时指令执行的条数。
- I_i 表示第 i 类指令的执行条数。
- CPI_i 为执行第 i 类指令的平均时钟周期数。

> 注意：I_c 是实际执行了的指令数目，而不是程序编译后的总指令数目。

那么 CPU 时间 $T = I_c \times CPI \times$ 时钟周期。可以分别从三方面提升性能。

- 使用好的算法：可以减少指令数。
- 使用好的程序语言或者编译器：可以减少指令数、CPI。
- 使用好的架构，可以减少指令数、CPI，可能减少时钟周期（提升时钟频率）。

评价性能必须对以上三个方面都做分析。

指令执行速率

CPU 性能的一个通用度量是指令执行的速率。可以以每秒百万条指令为单位，简记为 **MIPS 速率**。公式为：

$$\text{MIPS} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

其中 f 为时钟频率。

浮点性能可用每秒百万条浮点操作表示，简记为 **MFLOPS**。将上式中的 I_c 换成程序中执行浮点操作的次数、T 换成执行时间即得到 MFLOPS 的计算式。

尽管 MIPS 容易理解，且看上去确实能反映性能，但它有一些问题：

- 无法用 MIPS 比较不同指令集的计算机。
- MIPS 对 CPI 敏感，而 CPI 和具体程序有关。有可能 MIPS 随 CPI 的变化是和性能无关的。

基准程序

基准程序（Benchmarks），如 SPEC 开发的基准程序，可以更好地评测处理器的性能。其特点：

- 为了可移植性，它以高级语言编写。
- 它是面向不同领域的程序设计方式的代表。
- 它易于度量，有广泛的发行。

SPEC 基准程序不关心指令执行的速度，它对于速度度量（测量一台机器完成单个任务的能力）和频率度量（测量一台机器执行多个任务的吞吐量或频率）更感兴趣。

对于速度度量，SPEC 通过参照机器为每一个基准程序 i 定义一个参照运行时间，设为 $Tref_i$。设在受试机器上 i 的运行时间为 $Tsut_i$，那么用比值表示结果：

$$r_i = \frac{Tref_i}{Tsut_i}$$

用几何平均数将这些比值结合在一起，表示全面的性能测量。用几何平均而不用算术平均、调和平均是因为它对于比值这种归一化的量更合适。

Amdahl 定律

Amdahl 定律可以用于评价计算机系统的改进的影响。对于一项改进，加速比为：

$$\text{Speedup} = \frac{T_{old}}{T_{new}} = \frac{P_{new}}{P_{old}}$$

其中 new 表示改进后，old 表示改进前，P 表示表现（Performance），T 表示执行时间。

以程序为例，如果一个程序有 f 的部分可以理想化地无限并行，剩余部分只能串行，那么其在 N 核机器上运行比在单核上，加速比为

$$\text{Speedup} = \frac{T}{T(1-f) + \frac{Tf}{N}} = \frac{1}{1 - f + \frac{f}{N}}$$

它的意义在于说明了只加速技术或设计的一个方面，对于提升性能是有局限性的。对上面那个例子，即使 $N \to \infty$，也不会突破 $\frac{1}{1-f}$ 的瓶颈。

参考资料

本文的主要内容参考自：

- 上海交通大学《计算机组成》（课程代号：EI332）一课的课程材料。
- 《Computer Organization and Architecture (8th edition)》
- 《Computer Organization and Design: Hardware/Software Interface (5th edition)》
- 《Computer Architecture: A Quantitative Approach (6th edition)》
- 什么是冯诺依曼结构、哈佛结构、改进型哈佛结构?