Q：冯诺依曼机

A:冯诺依曼机特点

1. 计算机硬件系统由**运算器、存储器、控制器、输入设备和输出设备** 5 大部件组成
2. 指令和数据以同等地位存储在存储器中，并可按地址寻访
3. 指令和数据均用二进制代码表示
4. 指令由操作码和地址码组成
5. 指令在存储器内按顺序存放
6. 早期的冯诺依曼机以**运算器**为中心，输入 / 输出设备通过运算器与存储器传送数据。现代计算机以**存储器**为中心

Q：存储程序?

A：存储程序的概念是指将指令以代码的形式事先输入计算机的主存储器，然后按其在存储器中的首地址执行程序的第一条指令，以后就按该程序的规定顺序执行其他指令，直至程序执行结束

Q：在计算机系统结构中，什么是编译？什么是解释?

A：

- 解释：在运行程序的时候才翻译，翻译一句执行一句，边翻译边执行
- 编译：将高级语言转化为汇编语言

Q：计算机系统 5 层层次结构？虚拟机是哪几层?

A：微程序机器层、传统机器语言层、操作系统层、汇编语言层、高级语言层

虚拟机：操作系统层、汇编语言层、高级语言层

Q：字长?

A：

- 机器字长：计算机能直接处理的二进制位数
- 指令字长：一个指令字中包含的二进制代码位数
- 存储字长：一个存储单元存储二进制代码长度

Q：为什么采用补码加减?

A：省去计算机判断符号位或者说判断 +/- 运算的麻烦。采用补码表示后，不管是加法还是减法都是加法运算

Q：IEEE 754 标准

按照 IEEE 754 标准，常用的浮点数的格式如图 2.14 所示。

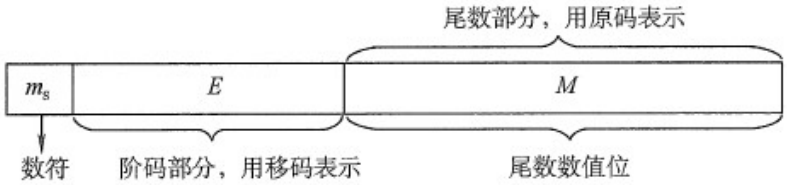

图 2.14　IEEE 754 标准浮点数的格式

IEEE 754 标准规定常用的浮点数格式有短浮点数（单精度、float 型）、长浮点数（双精度、double 型）、临时浮点数，见表 2.6。

表 2.6　IEEE 754 浮点数的格式

类型	数符	阶码	尾数数值	总位数	偏置值	
					十六进制	十进制
短浮点数	1	8	23	32	7FH	127
长浮点数	1	11	52	64	3FFH	1023
临时浮点数	1	15	64	80	3FFFH	16383

IEEE 754 标准的浮点数（除临时浮点数外），是尾数用采取隐藏位策略的原码表示，且阶码用移码表示的浮点数。

以短浮点数为例，最高位为数符位；其后是 8 位阶码，以 2 为底，用移码表示，阶码的偏置值为 $2^{8-1}-1=127$；其后 23 位是原码表示的尾数数值位。对于规格化的二进制浮点数，数值的最高位总是“1”，为了能使尾数多表示一位有效位，将这个“1”隐含，因此尾数数值实际上是 24 位。隐含的“1”是一位整数。在浮点格式中表示的 23 位尾数是纯小数。例如，$(12)_{10}=(1100)_2$，将它规格化后结果为 1.1×2^3，其中整数部分的“1”将不存储在 23 位尾数内。

注意：短浮点数与长浮点数都采用隐含尾数最高数位的方法，因此可多表示一位尾数。临时浮点数又称扩展精度浮点数，无隐含位。

A:

Q：在计算机中，为什么要采用二进制来表示数据?

A：

- 从可行性来说，采用二进制，只有 0 和 1 两个状态，能够表示 0 、1 两种状态的电子器件很多
- 从运算的简易性来说，二进制数的运算法则少，运算简单
- 从逻辑上来说，由于二进制 0 和 1 正好和逻辑代数的假 (false) 和真 (true) 相对应

Q：什么是存储单元，存储字，存储字长，存储体?

A：

- 存储单元：存储一个存储字并具有特定存储地址的存储单位。
- 存储字：一个存储单元中存放的所有二进制数据。
- 存储字长：存储字中存储的所有二进制数据的位数。
- 存储体：有多个存储单元构成的存储器件。

Q：主存储器中，什么是 MAR, 什么是 MDR，存储器的最大容量由什么决定?

A：

- MAR：指存储地址寄存器，保存需要访问的存储单元地址，反应存储单元的个数
- MDR：指存储数据寄存器，缓存读出或写入存储单元的数据，反应存储字长

存储器的最大容量由 MAR 寄存器的位数和 MDR 寄存器的位数决定

Q：多级存储系统?

A：为了解决存储系统大容量、高速度和低成本 3 个相互制约的矛盾

- “Cache - 主存”层次：解决 CPU 和主存速度不匹配的问题
- “主存 - 辅存”层次：解决存储系统的容量问题

Q：大端方式和小端方式?

A：

- 大端方式：字的低位存在内存的高地址中，而字的高位存在内存的低地址中
- 小端方式：字的低位存在内存的低地址中，而字的高位存在内存的高地址中

Q：随机存储器

A：

- 静态随机存储器（SRAM）：速度快，集成度低，价格高，不需要刷新，用于高速缓存
- 动态随机存储器（DRAM）：速度慢，集成度高，价格低，需要刷新，用于内存

Q：程序访问的局部性

A：

- 时间局部性：如果一个存储项被访问，则可能该项会很快被再次访问
- 空间局部性：如果一个存储项被访问，则该项及其邻近的项也可能很快被访问

Q：Cache

A：利用程序访问的局部性原理，把程序中正在使用的部分存放在一个高速的、容量较小的临时存储器中

引入目的：解决 CPU 和主存之间运算速度的差异

Q：Cache 与主存的映射方式

A：直接映射、全相联映射、组相联映射

- 直接映射：地址变换速度快，cache 利用率不高，块冲突率高
- 全相联映射：cache 利用率高，块冲突率低，地址变换复杂

Q：Cache 替换算法

A：随机算法、先进先出算法（FIFO）、近期最少使用算法（LRU）（最优）

Q：Cache 写操作方式

A：

- 全写法：写操作既写入 Cache 又写入主存
- 写回法：只把数据写入 Cache，当 Cache 中数据被替换出去之后才写入主存

Q：虚拟存储器

A：虚拟存储器是指具有请求调入和置换功能，能从逻辑上对内存容量加以扩存的一种存储器系统

- 页式虚拟存储器：把虚拟存储空间和实际空间等分成固定大小的页
- 段式虚拟存储器：把程序按段划分
- 段页式虚拟存储器：将程序按逻辑结构分成段，每段又分成若干个页

Q：快表

A：加快地址变换，虚页号变换成主存中实页号

Q：一条指令包含哪些部分?

A：操作码字段、地址码字段

指令中的地址可以为四地址（访存 4 次），三地址（访存 4 次），二地址（访存 3 次），一地址（访存 2 次）和零地址

Q：寻址方式包括哪两类?

A：指令寻址、数据寻址

Q：各种数据寻址方式概念

A：

- 隐含寻址：不显式的给出第二操作数的地址，规定累加器作为第二操作数地址
- 立即（数）寻址：地址字段存放操作数本身，数据以补码形式存放
- 直接寻址：形式地址 A 即为操作数的有效地址 EA
- 间接寻址：形式地址是操作数有效地址所在的存储单元的地址
- 寄存器寻址：在指令字中直接给出操作数所在的寄存器编号，操作数在寄存器中
- 寄存器间接寻址：在指令字中给出操作数所在的寄存器编号，寄存器中给出的是操作数所在主存单元的地址
- 相对寻址：程序计数器 PC 的内容加上指令格式中的形式地址而形成操作数的有效地址，广泛应用于转移指令
- 基址寻址：基址寄存器 BR 的内容加上指令格式中的形式地址而形成操作数的有效地址，利于多道程序设计，可用于编制浮动程序
- 变址寻址：变址寄存器 IX 的内容加上指令格式中的形式地址而形成操作数的有效地址，适用于编制循环程序，处理数组问题

Q：CISC 和 RISC

A：

- CISC：复杂指令系统计算机，指令数目多，字长不固定，寻址方式多，寄存器数量少，一般为微程序控制
- RISC：精简指令系统计算机，指令数目少，字长固定，寻址方式少，寄存器数量多，一般为组合逻辑控制

Q：指令周期

A：取指周期、间址周期、执行周期、中断周期

Q：影响流水线性能的因素

A：数据冲突（数据旁路）、控制冲突（动态预测）

Q：总线的两大基本特征

A：分时、共享

Q：系统总线按照传输信息的不同，分成哪几类?

A：数据总线（双向）、地址总线（单向）、控制总线（单向）

Q：集中式总线判优控制有哪三种方式?

A：链式查询、计数器定时查询、独立请求方式

Q：总线通信控制

A：

- 同步通信：总线上各个部件由统一的时钟信号控制
- 异步通信：总线上各部件没有统一的时钟标准，采用应答式通信
  - 不互锁、半互锁、全互锁

Q：I/O 控制方式

A：

- 程序查询方式：由程序不断的查询外设的状态，直到外设准备就绪
- 程序中断方式：在计算机执行现行程序的过程中，出现某些急需处理的异常情况或特殊请求，CPU 暂时中止现行程序，而转去对这些异常情况或特殊请求进行处理，在处理完毕后 CPU 又自动返回到现行程序的断点处，继续执行原程序
- DMA 方式：在外设和主存之间开辟一条直接数据通道
- 通道方式：专门管理 I/O 的处理机

Q：I/O 设备编址方式

A：统一编址方式、独立编址方式

Q：中断服务程序的基本流程

A：

1. 保护现场
2. 执行中断服务程序
3. 恢复现场

4. 开中断
5. 中断返回

Q：DMA 工作过程包括哪三部分

A:

1. 预处理
2. 数据传输
3. 后处理