数据表示、运算和运算器

冯诺依曼结构：存储程序、顺序执行指令

数据表示

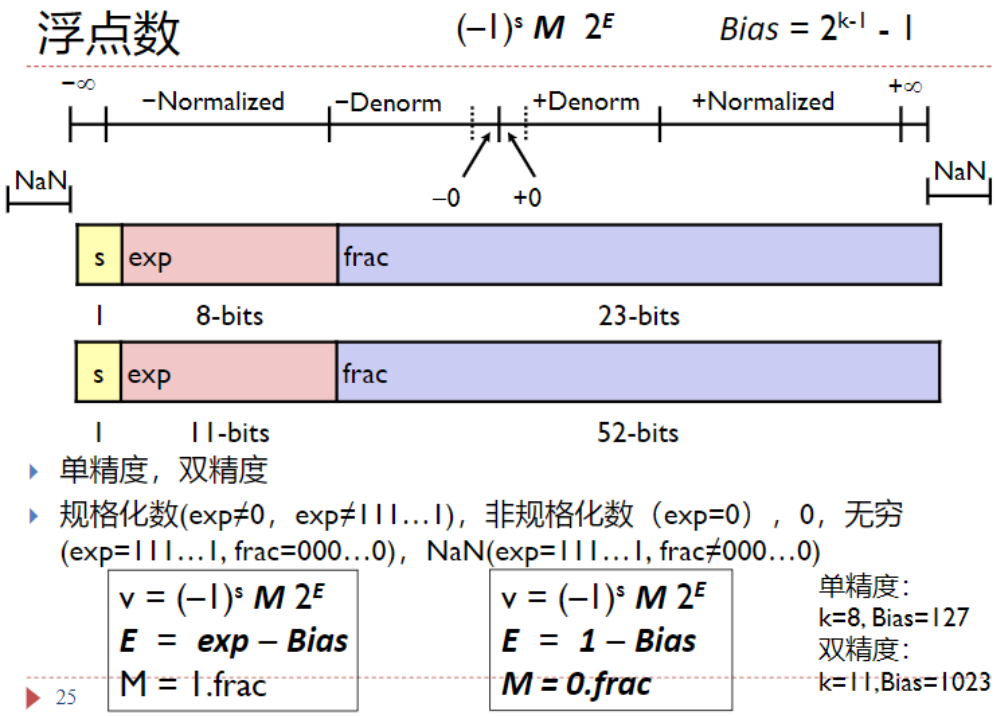
字符：

- ASCII (8 bit)，UNICODE (16 bit)，UTF-8 (变长)
- 点阵字体，矢量字体

整数：原码、反码、补码

- 正数原码、反码、补码表示均相同，
- 负数原码、反码、补码均不同，但有一个负数的原码和补码是相同的。

浮点数：



纠错检错

码距（最小码距）指任意两个合法码之间至少有几个二进制位不相同。

奇偶校验

- 在 k 位数据码之外增加 1 位校验，使 K+1 位码中 1 的总数为偶数（偶校验）或奇数（奇校验）
- 码距为 2

海明校验

- $2^r \geq m + r + 1$，发现并改正一位错
- $2^{r-1} \geq m + r$，发现并改正一位错，并发现两位错
- 码距为 4.

  改变任何一个数据位，成为另外一个合法数据后，至少改变的有两个海明码的校验位和总的奇偶校验位，再加上自身，两个合法数据表示之间至少有四位不同。

数据运算

运算器：由控制器产生的控制信号驱动，取得操作数，完成算术、逻辑运算，得到运算结果的状态，输出、存放运算结果，暂存运算的中间结果。

布斯算法

计算补码的乘积。

记$[x]_{补} = (x_{n-1}x_{n-2}\cdots x_1x_0)_2$，

$$x = -2^{n-1}x_{n-1} + \sum_{i=0}^{n-2} 2^i x_i$$

约定$y_{-1} = 0$,

$$\begin{aligned}
[x*y]_{补} &= [x]_{补} * (-2^{n-1}y_{n-1} + \sum_{i=0}^{n-2} 2^i y_i) \\
&= [x]_{补} * [-2^{n-1}y_{n-1} + \sum_{i=0}^{n-2} (2^{i+1} - 2^i) y_i] \\
&= [x]_{补} * [2^{n-1}(y_{n-2} - y_{n-1}) + 2^{n-2}(y_{n-3} - y_{n-2}) + \cdots + 2^0(y_{-1} - y_0)] \\
&= [x]_{补} * \sum_{i=0}^{n-1} 2^i (y_{i-1} - y_i)
\end{aligned}$$

```
# 被乘数、乘数为 n bit，部分积寄存器 2n bit.
部分积 = 乘数
附加位 = 0
for _ in range(n):
    cond = 附加位 - 部分积最低位
    if cond == 1:
        部分积 += 被乘数 << n
    elif cond == -1:
        部分积 -= 被乘数 << n
    附加位 = 部分积最低位
    部分积 >>= 1
```

原码除法：

- 恢复余数法
- 加减交替法: 恢复余数法步数不固定，控制复杂。常用不恢复余数法，又称**加减交替法**。其特点是运算过程中如出现不够减，则不必恢复余数，根据余数符号，可以继续往下运算，步数固定，控制简单。

```
# 被除数、除数均用绝对值补码，所有数均为 n bit
余数 = 被除数 - 除数
商 = 0
for _ in range(n):
    if 余数 > 0:
        商 += 1
        余数 = (余数 << 1) - 除数
    elif 余数 < 0:
        余数 = (余数 << 1) + 除数
    商 <<= 1
# 商的符号位 = 被除数符号位 ^ 除数符号位
```

控制器原理与设计

控制器：正确执行指令规定的功能；自动、连续执行指令。

执行每条指令平均使用的CPU周期数称为CPI.

两种不同类型的控制器：

- 组合逻辑控制器：采用组合逻辑线路、依据指令及其执行步骤直接产生控制信号。用于 RISC.
- 微程序控制器：采用存储器电路把控制信号存储起来，依据指令执行的步骤读出要用到的信号组合。用于 CISC.

  微程序是由微码组成的硬件指令集合。

指令系统

软件-硬件接口。计算机运行的最小的功能单元。

包括：算术/逻辑，访存，分支跳转

寻址方式：

- 立即数寻址：操作数直接由立即数给出
- 直接寻址：直接给出所需数据的地址
- 寄存器寻址：操作数在寄存器中
- 变址寻址：Reg + 偏移
- 相对寻址：PC + 偏移
- 间接寻址：直接给出所需数据指针的地址
- 基址寻址：专用基址寄存器 + 偏移
- 堆栈寻址

在 RISC-V 中，有立即数寻址，寄存器寻址，变址寻址，相对寻址 (jal label)。

RISC-V 指令格式：

- R 型，寄存器型，如 add
- I 型，立即数型，如 addi
- S 型，存储型，包括 lw/sw
- B 型，分支型，包括 bez
- J 型，跳转型，如 jal
- U 型，大立即数型，如 lui

流水线

五级流水：IF(取址)，ID(译码)，EXE(执行)，MEM(访存)，WB(写回)

- n 条指令，k 阶段流水线至少执行 $n+k-1$ 个周期。
- 在 RISC-V 中，可以将 lw/sw 指令的变址寻址变成立即数寻址，这样 EX/MEM 可合并，五周期将变为四周期。
  - 流水线时钟将不会变换，因为最长的流水阶段没有变换。
  - 一方面，流水的阶段变短了，另一方面访存需要的指令变多了，对性能的影响是未知的。

结构冲突

两条或者多条在流水线中的指令去访问相同的物理资源。

例子：

ID 读寄存器、WB 写寄存器

- 通过独立的读端口和写端口进行支持
- 双沿访问：前半周期写入，后半周期读出。

IF 读内存，MEM 读/写内存 (lw, sw)

- 暂停流水线

  注意不是插入 nop，因为 nop 也需要取址

- 将指令内存和数据内存分开

In general，结构冲突无法用静态调度（汇编器替换）的方法减少。

数据冲突

重叠执行的几条指令中，一条指令依赖于前面指令执行结果数据，但是又在指定的数据源中得不到时发生的冲突。

RISC-V 只需考虑"写后读"冲突，一个例子：

```
add t0, t1, t2
sub t4, t0, t3
and t5, t0, t6
```

第一条语句的结果在 ALU 段生成，但 WB 段才写回，而后两条语句在 ID 段就需要读取 t0 的值。

- 数据旁路：结果可用时即前传
- 暂停流水/插入 NOP

  lw 指令，MEM 段才得到结果，下一条指令必须等待一周期。

- 静态调度：汇编替换
- 动态调度：处理器调度，多发射 (multi-issue)、乱序执行

控制冲突

流水线中的分支指令或者其他需要改写PC的指令造成的冲突

影响最大，又称全局冲突。

- 暂停流水。为提高性能，将转移地址/条件的计算从 ALU 段前移到 ID 段。
- 分支预测：静态，动态预测

动态分支预测 BTB (Branch target buffer)

用一个缓存表记录分支语句的地址及其转移成功时的跳转地址，并根据之前的转移是否成功来动态预测。

层次存储系统

摩尔定律：芯片上集成的晶体管数量每18个月翻一番.

局部性原理：

- 时间局部性：最近被访问过的程序和数据很可能再次被访问
- 空间局部性：被访问的程序和数据往往集中在一小片存储区

原则：一致性和包含性

SRAM 和 DRAM

	SRAM	DRAM
存储方式	触发器	电容
破坏性读出/预充电延迟	否	是
定时刷新	否	是
送行列地址	一次	两次
速度/成本/发热	快/高/多	慢/低/少

- DRAM 的快速分页组织：连续用到相同行地址，将行地址锁存，只送列地址

Cache

- 全相连
- 直接映射
- 多路组相联

一般情况下，容量为 N 直接映射方式的 cache 缺失率与容量为 N/2 两路组相联 cache 缺失率相当。

一致性保证 命中 + 不命中：

- 写直达 + 写分配/非写分配
- 拖后写 + 写分配

Cache 缺失的原因：

- 必然缺失：首次访问
- 容量缺失
- 冲突缺失
- 无效缺失

缺失的优化：

- 块 (Line) 大小：较大的块更能利用空间局部性，但也会增大缺失损失和缺失率。
- 块替换策略：在全相连和多路组相联中，选择一组中的哪一块替换？
    - 最近最少使用 LRU
    - 先进先出 FIFO
    - 随机替换
- 多级 cache

接入策略：侧像法和隔断法

虚存

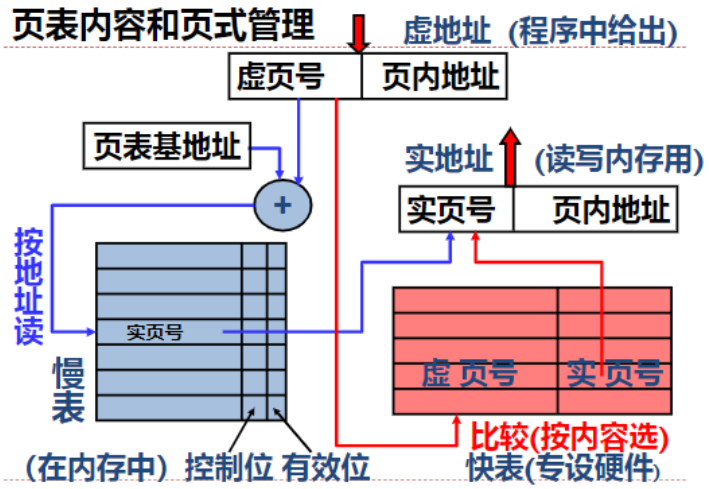
一些概念：

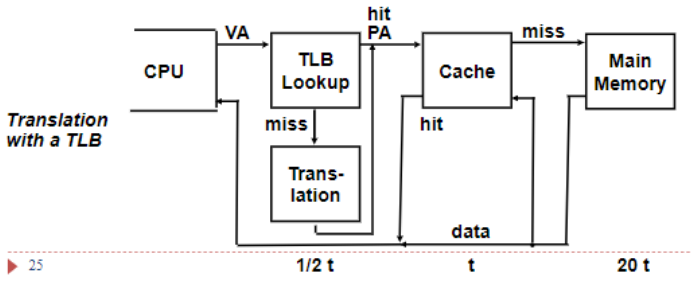
- 虚拟地址、逻辑地址：编程使用的地址
- 存储器地址、物理地址、实地址：物理存储器的地址

虚存	Cache
克服存储容量不足	解决CPU和主存的性能差距
由 OS 管理	由硬件实现

页表，存放虚页号到实页号的对应关系，由 OS 管理

TLB 是页表的 cache. 通常 N 路组相联，LRU 替换。





外存

容量大、成本低、断电后还可以保存信息，能脱机保存信息，弥补了主存的不足。

磁盘

可移动的读/写头来访问。

访问时间 = 寻道时间 + 平均旋转延迟(转半周时间，单位 RPM 每**分钟**转数) + 传输时间 + 磁盘控制器延迟

- 大部分为额外开销：一次传输大量数据
- 将页面存储于相邻扇区可避免额外寻到开销

对磁盘的访问总是由page fault引起的。

可靠性和可用性

- 可靠性：**设备**故障概率，可通过提高各部件可靠性、减少组件
- 可用性：**系统**正常概率，可通过硬件冗余来提高
    - 可用性提高可能带来可靠性的降低

RAID：用冗余阵列提高磁盘性能和可用性。

- RAID0：[1, 3, 5, 7], [2, 4, 6, 8]
- RAID1：[1, 2, 3, 4], [1, 2, 3, 4]
- RAID4：对带校验，增加一块磁盘

- RAID5：RAID4 基础上将校验位循环分布于所有驱动器。
- RAID6：RAID5 基础上增加一块磁盘做校验

SSD

FLASH + DRAM，能擦除的次数是有限的。

- block 是最小的擦除单位
- 一个 block 包含多个页，页是最小的读写单位

FTL (Flash Translation Layer)，将逻辑地址转为物理地址，帮助完成磨损均衡。

RISC-V 异常处理

同步异常：访问错误、断点、环境调用、非法指令、地址未对齐等。

异步异常 (中断)：外部中断、软件中断、时钟中断等。

异常处理：在 CSR 中记录异常种类 (mcause)、异常发生指令 (mepc) 和附加信息 （mtval），跳转到 mtvec 运行异常处理程序。

总线和 IO 设备

输入/输出方式

程序直接控制：成本低，效率低，严重占用 cpu 资源。

程序中断：对 CPU 干扰较大，可管理多个外部设备，适用于传输量不高、速度低的情况

- 中断处理过程：

  关中断、保存断点、判中断源转中断服务、开中断、执行中断服务程序、关中断、恢复断点、开中断、返回断点

直接存储访问（DMA）：一对一硬件，高速成组传送，对 CPU 干扰适中，无法适用大量高速设备管理。

工作方式：

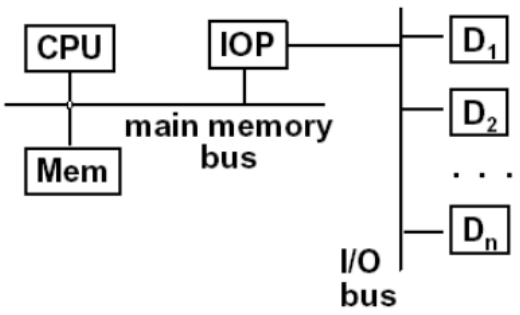
- 独占总线、周期窃取、DMA 和 CPU 交替访问内存

流程：预处理 (中断或程序控制) -> DMA 数据传送 -> 后处理

一些问题：

- DMA 使用实地址，要进行虚实地址转换
- 直接设计硬件控制，保持缓存一致性

I/O 通道：代替 CPU 管理控制外设，一对多。



类型：

- 字节多路通道：简单的共享通道、分时处理，适用于中低速设备

- 选择通道：选择一台外设独占通道，适用于高速设备
- 数组多路通道：两种方式的结合。

外围处理机：由外围计算机独立完成输入输出，通过通道 (共享内存) 与主机交互。

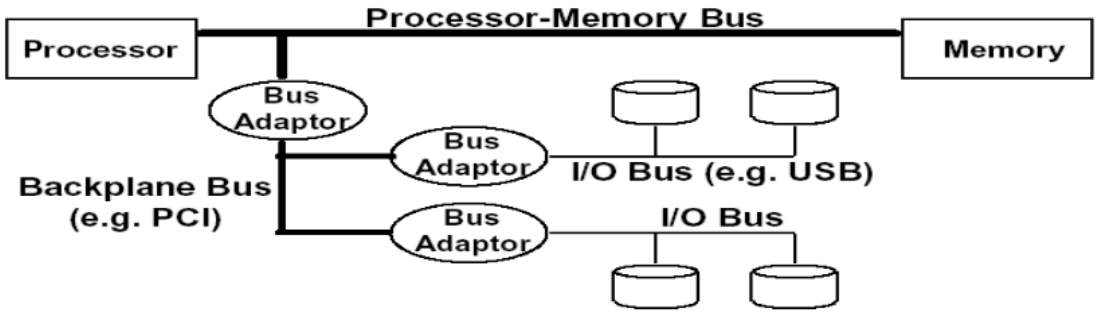
总线

处理器与其它组成部分的接口

组成：

- 控制线，标记总线事务的开始和结束，指明传输信息的类型
- 数据线，传输信息

类型：处理器/主存主线 + IO 总线 + 主板总线



总线仲裁

管理多个设备竞争使用总线。

集中仲裁：

- 菊链仲裁：所有设备共用一个总线请求信号，无法保证公平
- 集中平行仲裁

分布仲裁

数据传输模式

- 同步总线：总线上所有设备按同一时钟频率工作，传输协议根据时钟信号制定，逻辑简单高速。
  - 防止信号扭曲，高速工作时，总线距离应该够短
- 异步总线：使用握手协议。

提高总线带宽

- 增加总线宽度
- 分别设置数据总线和地址总线
- 成组传送：一个总线事务传送多个数据，每次只需要在开始的时候传送一个地址，直到数据传送完毕才释放总线。

  代价：复杂，延长后续总线请求等待时间。

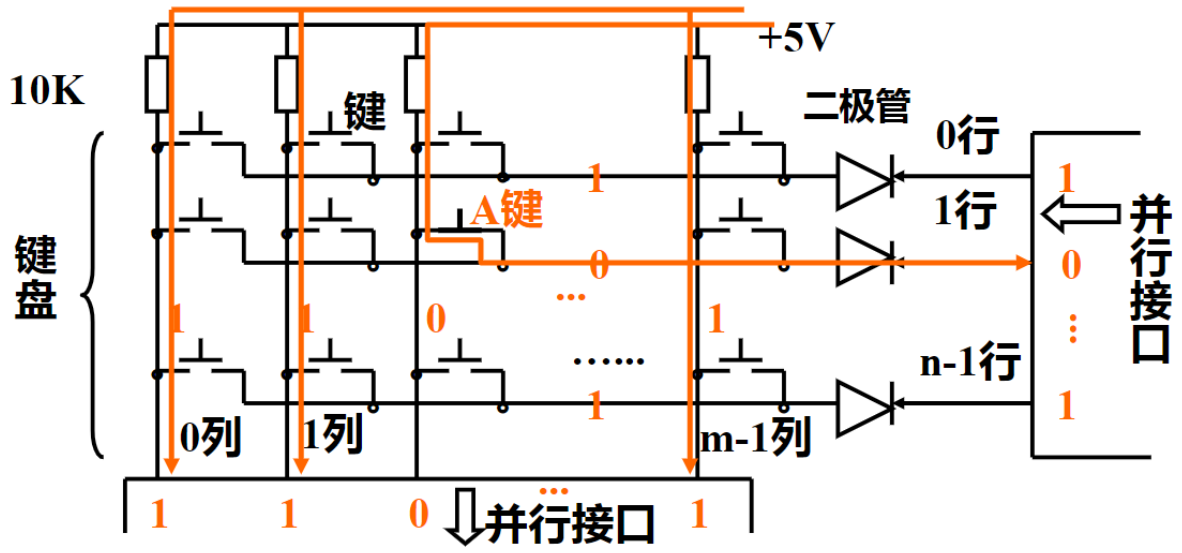
多主设备总线提高事务数量：冲裁重叠 (流水) 等

接口电路和外部设备

接口：为主机识别 IO 设备，控制通信，数据缓冲，屏蔽外部设备的差异。

- 串行接口芯片8251
- USB： <= 127 个设备，可在运行时接入，同步串行；四根线：电源，地，双数据线。

键盘：把敲击的键在键盘上的位置对应为一个编码。



鼠标

点阵输出设备——显示器，打印机