C1 计算机系统概述

①冯·诺依曼结构：

硬件（5个）

以运算器为中心

（软件）工作方式：取指令 — 分析 ～ — 执行 ～ — 计算下条 ～ 地址.（必须顺序逐条执行）

⇒ 程序和指令预先存在MEM中，自动、逐条取出并执行

② 对冯·诺依曼的改进

（现代计算机结构）

- 以存储器为中心（I/O、计算可并行）→ 缓冲 + DMA → 原串行.
- 多种存储器共存（CPU访问主存，提高性能、降本.（层次化，程序访问局部性原理）

- 存储器
- 运算器
- 控制器（PC、IR、ID、时序信号、控制信号）
- I/O 接口设备.

③ 工作过程：☆准备：{ 程序装入主存；程序入口写入PC

执行：取 ～ — 分析 ～ — 执行～

（PC←PC+1）（转移型 重写PC）

⑤ 技术指标 / 性能指标：

硬件：机器字长、存储容量、CPU主频.

性能：响应时间 $= T_{CPU} + T_{等} = I_N \cdot CPI \cdot T_c + T_{等}$.（→ 针对任务，可能不止一个指令 $\Sigma I_i \cdot CPI_i$）

吞吐率：$T_P = \frac{\Sigma W_i}{T_n}$（$\Sigma W_i$ — 工作量，T_n — 总时间）

{ $MIPS = \frac{指令数}{10^6 \cdot T}$；$MFLOPS = \frac{浮点次数}{10^6 \cdot T}$

C2：数 & 码，表示 & 运算

①数制转换 { 整、小；{ 十、八、二、十六

②编码：{ 原、补、反、移

$-2^{n-1} \sim 2^{n-1}-1$（原码 $-2^{n-1}+1 \sim 2^{n-1}-1$）→ $[x]_补 = [x]_补$ 带符号位取反 +1

→ 比原码多一位.

③数据表示：

整数 { 定点表示：约定 { 有符整、无符整、纯小数 } 无符号位.

整数 { 无符号 → 二进制、定点无符；有符号 → 二进制定点 补码（便于减法）.

类型转换 { 位拓展 { 0拓展；符号拓展. ⇒ 与门实现. 所补位 $= OP \cdot D_{n-1}$（最高位）（0 - 0拓，1 - 符号拓.）

浮点数：

阶符 - 阶值 - 数符 - 数值.（→ 在后面补0.）

看Ex：

规格化 → 使尾数最高位为有效数字 { 原：1；补：与符号反.

IEEE754

④运算

字节：逻辑运算 → 与标志位有关. ⇒（☆有符号数运算）

☆定点数 加减运算：{ 无符号；补码.

补码运算 $[A+B]_补 = [A]_补 + [B]_补$

（竖式）$[A-B]_补 = [A]_补 + [-B]_补$（再置一次）（不溢出时进位舍弃）

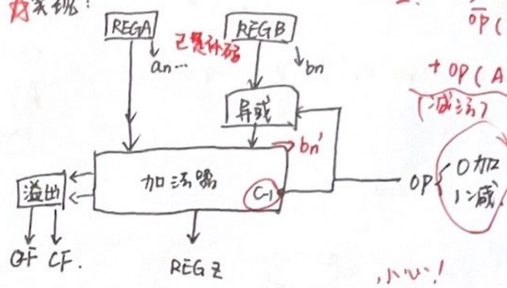
溢出判断：①必同号加，异号减.

② $OF = \{ (a_{n-1} \oplus z_{n-1}) (b'_{n-1} \oplus z_{n-1})$；$C_{n-1} \oplus C_{n-2}$ → 进位.

$b_i \oplus O_p = b'_i$

⇒ $OP(\bar{A}\bar{B}_{n}z_{n-1} + AB\bar{z})$ $+ \overline{OP}(A\bar{B}\bar{z} + \bar{A}Bz)$（减法）

☆实现：



无符号运算：只有结果解释为无符号数（小心！）

& 溢出（CF）不同：$CF = OP \oplus C_{n-1}$（0加1减）（进位）（最高位）

移位运算：{ 算术：补0 >>A；逻辑：补码右移补符号位，其余补0. >>L.

溢出：①左移才溢出

{ 补：移出与符号位相反数.；原：移1.

☆乘法 { 无符号 → 符号+右移，走一遍（n次加法 & 右移）；原码 → 逐算右移（n-1次判断 - 加法 - 移位）（不熟）☆拓展位.

溢出：$OF = Z_{2n-1} + \dots + Z_n$. → 都是0才不溢出.

浮点数加减：{ 对阶，$max(E_A, E_B)$；尾数加减；规格化（0.5）关注负数补码）；舍入（舍入法）；溢出判断.

补码乘：n位循环，加法 +0 / $A_补$ / $[-A]_补$. 算术右移

● 标志位表示逻辑运算（关系）（结果 符号：0正1负）

① 先用减法 A-B 产生 ZF、SF、OF 有符号

② $A \geq B$：$\overline{OF \oplus SF}$

$A \leq B$：$OF \oplus SF + ZF$

$A > B$：$\overline{OF \oplus SF} \cdot \overline{ZF}$

$A < B$：$OF \oplus SF$

$A = B$：ZF

C3 存储系统　　MMU

绪论：技术指标 { 存储容量、读写速度（存取）、传输带宽（Mbps）$= \frac{W}{T}$

层次结构 { Cache－主存；主存－辅存。

工作过程. ① 程序在辅存中
② 程序执行时部分装入主存
③ 会按逻辑地址访问.（虚存）→（cache）

☆SRAM：存储矩阵＋译码器＋读写电路＋控制电路.
非破坏性读（利用差分放大器）

引脚组织：
Ex：单元长度 w，单元个数：2^n.
⇒ 容量：$2^n \cdot w = S$

引脚 {
- 数据：单向：2w，双向：w.
- 地址：单向输入：$\log 2^n = n$ 根（→ $\frac{S}{w}$）
- 控制：1×片选 ＋ 1×读/写　刷漏

读写时序：看笔记.看书
{ 读：首先发地址，稳定后 $\overline{CS}$（$\overline{WE}$ 无效）.
写：首先发地址，然后 $\overline{CS}$ 和 $\overline{WE}$
稳定 tW 后. → 防止未稳定译码出错.地址

DRAM：存储矩阵、译码器、读写电路. 地址锁存器时序控制、再生电路.
破坏性读，要刷新，定时再生.（读出时写入）

引脚组织：
{
- 数据：单向 2W，双向 w
- 地址：单向输入：$\frac{\log_2 \frac{S}{w}}{2}$ → 行、列分时传输
- 控制：选通 ×2（$\overline{RAS}$、$\overline{CAS}$）、读/写 ×1. → 实现片选

读写时序：
{ 读：先 RAS，再 CAS，~~再 WE~~，并同时撤销
写：$\overline{RAS}$ → 一直有. $\overline{WE}$ － $\overline{CAS}$ 写. 数据先于 $\overline{CAS}$，撤晚于 $\overline{WE}$

刷新：原因（电容电荷泄露）
行刷新 {
- 集中式（先读后刷所有行）（刷新周期）
- 分散式（一半读写，一半刷新）无死区（存储周期）
- 异步式（效果最好）（死区为一个存取周期，均匀分散刷新周期）

非易失，启动时最先的程序 → 提高访问速度.

主存.（ROM＋RAM）
{ 结构：主存单元长度、主存地址空间（＝CPU寻址空间）
用户：主存单元个数（总容量 < 最大容量）

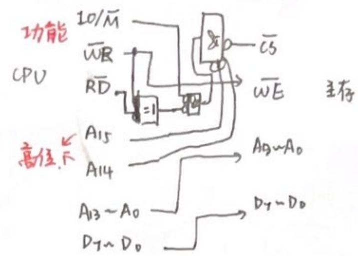
☆组合拓展（注意 SRAM、DRAM 区别）
{ 字（改地址～）
位（改 Data 引脚）
字位.
看例题，手绘模拟一下.

小测题.

（$A_n \sim A_0$，$D_k \sim D_0$，{ $\overline{CS}$，$\overline{WE}$；$\overline{RAS}$，$\overline{CAS}$，$\overline{WE}$ }）
2-4译码/多$\overline{RAS}$（DRAMC）

☆CPU与主存连接：
笔记例题.

☆ $\overline{CS} = \overline{(\overline{RD} \oplus \overline{WR}) \cdot \overline{IO/\overline{M}} \cdot \overline{A_{15}} \cdot \overline{A_{14}}}$



提高访存速度.
① 增强型 MEM
{
- SDRAM：{ 同步工作：基于时钟，不握手，控制更快. 突发传输：一次操作可访问多个地址连续的数据
- ☆多一个片选引脚！
- DDR SDRAM：{ CLK 上升、下降沿都 I/O；缓冲区预取 } ⇒ SDRAM 2倍

DDR2：SDRAM 4倍.

② 多体交叉 MEM（访问）
{ 交叉：轮流访问，突发传输 看图.
并行：同时访问（W个）

☆Cache
CPU-Cache：字单位　Cache－主存：块单位（n个主存字）
命中率（H）. 平均访问时间（TA）$= T_{命中} H + (1-H) \cdot T_{缺失}$ → 也包含一次命中.
$= T_{命中} + (1-H) T_{缺失}$

每种映射重点　Cache：有效位　标记　行号　块内地址
主存：块号　块内地址.

{ 映射　{
- 直接：冲突率最高. 速度↑，成本↓，性能↓（1个MUX）
- 全：冲突率↓，速度－. 成本↑，性能↑（多个MUX每行一个）
- 组：冲突－，速度－，成本－，性能－ ↓ n个MUX，n路 n个.

块－组－行（直接）（全）⇒ 看例题.
n路组相联.

替换
写

替换 { RAND、FIFO
LRU ⇒ Cache 多 LRU位.（不论是否命中计数器都更新）→ 由组数定. $\log_2 n$
初全为1，被访问清0，值<被访问则+1，每次访问值最大的　相同则按地址序

写 { 全写 H不好（命中 & 缺失）
写回 H好 → Cache 多一位脏位.

虚存：主要是 MMU → 在CPU中，实现地址变换和快表
TLB ⇒ 未命中访问页表.
{ 逻辑地址 ← 程序
物理地址 ← 主存
工作：① 虚存－主存地址变换
② 虚存－辅存 ～，辅存调入主存

{ 段式（大）（段表在主存中）
页式（页表在主存中）（小）
段页式（要访存2次.
⇒ 都是基址＋段/页内地址 查表知.

C4 指令系统

指令格式：操作码 + 地址码 { 零～多地址 } ☆约定.

指令字长 与机器字长无关（单、半、双…）

操作数的存放

MEM { 端序：大端、小端 ↔ 正常顺序 / 反序
对齐：不对齐、对齐 ↔ 指定位置开始

大：N N+1 N+2 N+3
正常顺序：B3 B2 B1 B0
小：N+3 N+2 N+1 N

→ 边界对齐：2^n 位数据，ADDR 低 n 位为 0

寻址：{ 指令（2种）{ 顺序 / 跳跃 }
数据.（8、9种）

⇒ 要掌握地址计算方法

无 / REG / MEM {
立即 OPD = Imme.
寄存 OPD = Ri
直接、间接、REG间接 先算EA，OPD = M[EA]. 看附注存.
基址、变址. EA = REG' + A，OPD = M[EA]
隐含（无地址码）.
}

相对寻址是指令 EA = PC + A. 要考虑 PC ← PC+1! 有符号数. 有时取一个节加一次!

{ CISC (Complex)：REG-MEM，多，复杂，时间长，寻址多
RISC (Reduced)：REG-REG，少，简单，有大量寄存器.

☆C5 中央处理器

CPU组成：看笔记图.

BIU / MMU ↔ MDR / MAR
PC、IR、ID
CU
REGs
ALU

指令周期+中断周期.

专用REG 用户不可见（PC、IR、MAR、MDR、控制REG）.

功能：循环执行指令，检测并处理异常&中断.

原理：CU产生控制信号，数据通路实现操作.

取指令 - 分析指令 - 执行指令.

原子操作 {
REG传送 Rx ← Rx Rxout、Ryin
MEM读 MDR ← M[MAR] Read、WMFC
MEM写 M[MAR] ← (MDR) Write、WMFC
算逻 Z ← Rx op Y Rxout、op=xx、Zin（先Y!）
}

数据通路.

组成 { 取指：PC、IR … ID不属于DP
执行：GPRs、ALU、PSR… }

结构 { 总线：简单、分时，仅多周期CPU，GPRs 一个读端，三态门 ↗ 传上只一个数据
专用：复杂，同时，单、多周期均可，有多路MUX，GPRs 2个读端. }

功能 写μOP，μOPCmd，看例题. { 取：t_1-t_3
执：t_4 - … } 记得自己写写 End、ALU、ExtO

专用的也看看.

设计 {
单周期：{ 只专用，减少μOP
不可复用部件，哈佛结构.
时钟数为最复杂指令的. }
多周期 { 可专用可总线.
可复用.
指令周期时钟数可不同 }

控制器

{ 指令部件
CU 核心 { 石反布线 / 微程序 }
中断机构.

CU = ID + 时序电路 + 产生控制信号电路

⇒ 循环产生 μOPCmd，在正确时间

2级时序（硬布线：组合逻辑）
1组时序（微程序：存在主机）

时序信号形成（组织）：

{ 指令周期 = 多个机器周期 ↗ 中断.
机器周期 = 多个节拍 → 比最复杂多
节拍周期 = 多个工作脉冲，一个节拍一个μOP ← 用于定时 CPU最小时间单位.

早期：机器 - 节拍 - 工作 3级
现代：节拍 - 工作 2级.

序列组成 { 定长 / 变长.

时序信号电路 { 定时 / 定序.

定时：{ 同步：用时钟（CPU主频），CP = CLK
异步：应答或握手，CP = Ack，要一根应答信号线
联合：两者都用，用WMFC看现在用哪种

定序：环形信号发生器. { 移位REG / 计数+译码器.

CU组成对比的图 和设计步骤、实现方法

异常&中断 ← CPU内部异常 / CPU外部设备.

异常（随时处理）{ 故障：当前/终止
陷阱：下条指令
终止：终止程序/关机

Ex：缺页、除溢出、断点、无效表、右更件

JNZ { ZF=0 { 双字长 Addr / 单：PC ← PC + Disp } PC ← Addr.
ZF=1，无事发生.

中断 { 可屏蔽：可暂不管，可多个后处理. 用IF表子屏蔽.
不可屏蔽：要立即管，当前周期结束后处理.

外部机环…

电源故障、校验错…

☆处理.

① 保存断点及程序状态.
⇒ 用专用寄存器存，返回时恢复.

② 关中断（IF=0）⇒ 处理时不再中断.

③ 识别事件类型并转入处理程序. 识别中断要访问外设
⇒ 将正确处理程序入口地址 → PC.

{ 单向量：共用一个，直接写
向量：用一个中断向量表 IVT 写所有入口地址，在主存.
先到优（最紧急）→ 查表 → 写入PC.

Ex：响应可屏蔽中断条件（3个）

Cb.总线 { 数据总线（双向）
地址～（单向）
控制～（单向）{主：控制 从：状态
片内（只数据）
系统
通信（无地址）

特性 { 物理
功能
电气
时间

性能指标

①总线宽度：数据总线位数 w（多少位）.

②～带宽：$B=\frac{w\times f}{m}$ → 时钟频率 ☆ 1B=8bits.
→ 每个数据要的时钟.

③负载能力

操作过程：1.申请及分配阶段（仲裁使用权，只有主设备）
2.寻址（选择从设备，主发送地址&命令）
3.数据传送（读写Data）
4.结束（撤销信号）

仲裁（集中式）{ BR-总线请求 BG-总线允许.

①链式查询：自动轮询各主设备 主设备自行传递.
⇒ BS、BR、BG，一直向下去直到 BS=1.
只用2根信号线，优先级有距离.（固定） → 不公平.
⇒ 易断链.
最少的一种
$\overline{BS}\cdot BR=1$ 开始
BS=1 结束

②计数器定时查询方式：
⇒ 设用设备号总线（log2n）根.
⇒ 逐次定时查询各个设备.
公平. ⇒ 固定优先级，按顺序，但不会断链

③独立请求
⇒ 无须询问，主设请求线分别连至仲裁器，仲裁器结果直接发. → 2n根.
⇒ 隐藏式仲裁（结束阶段做）.
⇒ 公平可动可静优先级 ⇒ 效率最高.
已BAi=1，总线空闲开始.
BGi ← 1，结束.

定时

①同步定时方式
⇒ 用时钟CLK定时，一次传送为几个周期
⇒ 每步固定时长，为最复一步时间.
⇒ 总线长度长短，设备时间相近用.

②异步定时方式
⇒ 通过握手(Ack)定时，时长可变.
⇒ 传输周期长，一共4个阶段，
每个阶段都要握手.

③半同步定时 ☆注意时注意：以CLK为Tc为基准，所有要取整！
⇒ 同步用于传输定时CLK，
异步用于控制定时（是否Ready…）
$T_{总}=\lceil\frac{T}{T_c}\rceil\times T_c$.

总线结构：
{ 单总线（控制简单、性能差
多总线 { 双：Ex HOST+I/O总线
多
⇒ 用总线桥进行管理（仲裁、发起、响应…）

总线互连：
⇒ 总线接口电路：
各个设备连入总线的转接电路.
{ 控制、缓冲
记录状态、
格式转换.

C7 I/O系统（主机与外设信息交换）

性能 { 响应时间
吞吐率
I/O占CPU时间

硬件/软件

外设与主机的连接 ⇒ 总线 GPR → I/O端口.

工作过程：① 主机 -(一对多)- 接口 → 总线操作，接口存信息在REGs

② 接口 -(一对一)- 外设 → 通信操作，(自动实现)(特设备就绪)

操作码 | 设备码 | 命令码

编址：

①统一编址：

主存/I/O共用一个地址空间.

⇒ 不加机器指令数，控制线只有MEM写&读(2条)

⇒ 主存空间↓，地址译码复杂.

②独立编址：

主存/I/O从0开始编.

⇒ 多加I/O指令，控制线4条 MEM/IO 写&读.

⇒ 不影响向主存、地址译码简单.

外设识别：

I/O接口比较总线上地址码与自己REG中唯一的设备号.

数据传送 { 无条件：随时可传，不用启动/响应(简单设备)
有条件：先启动，就绪后传 (复杂设备)
↳ 增加信号线传状态.

外设联络 { 立即响应：无信号传 (无条件传送)
异步联络：2根请求/应答 } 条件传送
同步联络：时钟线

☆I/O传送控制

①程序直接控制I/O (字符设备，1个Data/次)

{ 直接传送：只占一个指令周期，无条件，不启动 (用并行接口只有数据口)
程序查询：CPU一直查询设备是否齐备，就绪后传 (条件)
占多个指令周期，CPU与外设串行
启动-完成. 等待外设时CPU不工作.
{ 独占：启动后立刻开始
定时：启动后等一会再开始 ↳ 部分并行.

⇒ I/O接口：设置控制口、状态口，支持 启动① 查询② 传送③ 数据库.

②程序中断I/O (字符设备，1Data/次)

⇒ CPU启动外设，继续执行原程序部分并行.

⇒ 外设就绪后向CPU发请求.

⇒ CPU响应，开始I/O

{ 屏蔽&不可
单重&多重 → 中断响应.
向量&非向量 → 中断处理
不同外设 ↳同一外设

识别中断源：
判优 { 软件排队
硬件排队
访问外设，返回最紧急请求，
查IVT，加写入PC.

I/O接口：

比程序查询多了中断 { 中断允许位EI
中断请求位INT.

⇒ 支持3+1(中断方式)子操作.

③DMA (块设备，一批Data/次).

⇒ CPU要多一个DMA接口硬件

要求CPU { 负责准备&结束工作.
传送时让出总线权
但可用Cache
DMA接口为主设备

1) CPU发送请求+启动，回去原程序

2) DMA接口实现 数据传送，结束后向CPU请求

3) CPU响应.&结束工作.

完全并行.

传送方式 { CPU停止访问.(每个Data传时申请，完成后放弃).
周期挪用.(准备时DMA放弃，就绪时申请)
分时支援访问.(定时轮流分配).
↳ DMA如何申请总线权.

I/O接口

⇒ 外设接口-主机：并行

⇒ 接口-外设：可串可并，看设备

(不用串，并用并)