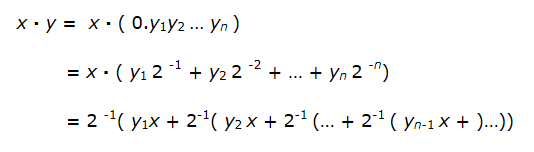
1. 冯诺依曼模型：以运算器为核心，必须顺序执行每一条指令，指令执行分为 取出指令，分析指令，执行指令（指令地址存储于PC）。
2. 机器数编码：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 原码 | -(2^(n-1)-1) ~ 2^(n-1)   |  |  |  | | --- | --- | --- | |  | 00…0 | 01…1 | | 11…1 | 10…0 |  | |
| 反码 | -(2^(n-1)-1) ~ 2^(n-1)   |  |  |  | | --- | --- | --- | |  | 00…0 | 01…1 | | 10…0 | 11…1 |  | |
| 补码 | -2^(n-1) ~ 2^(n-1)   |  |  |  |  | | --- | --- | --- | --- | | 10…0 | 11…1 | 00…0 | 01…1 | | -2^(n-1) | -1 | 0 | 2^(n-1) | |
| 移码 | -2^(n-1) ~ 2^(n-1)   |  |  |  |  | | --- | --- | --- | --- | | 00…0 | 01…1 | 10…0 | 11…1 | | -2^(n-1) | -1 | 0 | 2^(n-1) | |

1. 8421码：以四位二进制数代表一位十进制数，并在末尾用1100，1101表示+和-。
2. 原码乘法：乘数的每位与被乘数相乘并移位相加。

详细（<https://www.cnblogs.com/yjbjingcha/p/6956431.html>）



1. 浮点数加减：调整至阶数相同，将尾数相加减，规格化后进行舍入

（截断法（全舍），置1法（全入），零舍1入，查表舍入（原码：0舍1入，补码：正数（0舍1入），负数（舍（-1，-0.5]，入（-0.5，0）））。

浮点数表示：

十进制->二进制->左/右规格至0.1…->用浮点数表示

|  |  |  |  |
| --- | --- | --- | --- |
| 阶符 | 阶值 | 尾数符 | 尾数值 |
| 左/右规格 | 规格次数 | 与原数相同 | 规格后数值 |

1. 用ALU比较数的大小：将两数进行加减运算，并观察借进位，0标志。
2. 存储器：

|  |  |
| --- | --- |
| 顺序存取：SAM（磁带） | 只读存取：ROM（光盘） |
| 随机存取：RAM（内存） | 直接存取：DAM（磁盘） |

1. CPU->主存（Cache地址转化->Cache）-> 辅存 程序访问局部性原理
2. 64K \* 16b -> 字扩展(串联)8 \* 8K \* 16b -> 8 \* ( 位扩展（并联）4 \* (8K \* 4b))。
3. CPU引脚：地址（log64K）引脚（A0~A16），数据 (16b) 引脚（D0~D16），状态/控制引脚 CS（RD,WR,IO,MEM）
4. Cache：

|  |  |
| --- | --- |
| 管理区 | 数据区 |

|  |  |  |  |
| --- | --- | --- | --- |
| 有效位 | 标记位 | 状态位 | 缓存块 |

1. Cache地址映射：直接映射，全相连，组相连

组相连：

Cache：

|  |  |  |
| --- | --- | --- |
| 群号 | 群内块号 | 块内地址 |

主存：

|  |  |  |
| --- | --- | --- |
| 组号 | 组内行号 | 块内地址 |

群块号=组号，块映射到组后组内查全表，块内地址相等，tag=群号

1. Cache替换算法：RAND，先进先出，LRU（最近最少使用）

LRU：设置计数器，初始为1，每次更新时若未被访问则+1，否则置0，选取计数器值最大的行进行替换。

1. Cache写策

全写法：写入主存同时写入Cache。T命中=Tmem，T缺失=0（缺失直接写入主存）；

写回法：修改时写在Cache中并记录脏位，Cache被替换时脏位为1的行写入主存。Cache行管理信息=有效位+标记（tag）+记数器位（LRU）+脏位 （T命中=Tcache，T缺失>=Tmem（缺失时将行调入再写））

1. 指令：目的操作数<-源操作数1 OP 源操作数2（R2<-（R1）+ M [1000H]）

R2代表寄存器，其中的内容用（R2）表示，M [1000H]代表储存单元。

算术运算结果产生的进位，零，溢出，符号标志（CF,ZF,OF,SF）存放在状态寄存器中

1. 操作数存放：寄存器，存储器，IO设备，堆栈，指令寄存器。

存储器中：存放（小端/大端），排列（对齐/不对齐）。

小端方式对齐存放：数据低位存在存储器低位，数据小端靠存储器边缘。

Tips：12345678H存入按字节编址存储器中

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #5 | #4 | #3 | #2 | #1 | #0 |
| &&& | &&& | 12H | 34H | 56H | 78H |

1. CPU组成：
2. 指令控制：指令地址（PC）,指令内容（IR），指令分析（ID）
3. 时间控制：时序电路
4. 数据加工：算术逻辑运算器（ALU），浮点运算器（FPU），数据寄存器组（REG），状态寄存器
5. 外部访问：总线逻辑电路，缓冲寄存器（接口），存储器管理（MMU）
6. 异常中断：异常中断机构

指令执行过程：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 取指令（指令译码ID） | 取操作数 | ALU运算 | 保存结果 | 更改PC |
| PC->MAR | MAR->MDR | MDR取数据进ALU | MDR->IR | PC=PC+1 |

CPU数据宽度与外部数据通路宽度一致

1. 时序系统：
2. CISC（复杂指令集）：硬件层面支持高级语言语句归类后的复杂操作，指令周期采用定长周期。
3. RISC（精简指令集）：接受简单指令，采用变长指令周期。
4. 异常处理及中断：响应（保存断点及寄存器状态，关中断），处理（识别中断事件），返回（恢复断点及程序状态）
5. 指令流水线：并行执行不同指令的不同部分

拍长=max{段操作时间}+时延，时机吞吐率=单位时间执行指令个数

加速比=串行执行时间/流水线时间，效率=指令所占时空区/总时空区

|  |  |  |  |
| --- | --- | --- | --- |
| 存数 | &&& | &&& | 1(MEM) |
| 运算 | &&& | 1(ALU) | 2(ALU |
| 取数 | 1（PC） | 2(PC) | 3(PC) |
|  | 拍1 | 拍2 | 拍3 |

1. 总线仲裁：集中式（链式查询，计数器定时查询，独立请求），分布式

链式：优先级固定

计数器定时：可设置优先级，若从0开始则与链式相同

独立请求：外设发出请求信号并由中央排队电路决定总线使用权归属

分布式：每个外设均具有一个仲裁号和仲裁器，请求时仲裁器将自己的仲裁号与仲裁总线上的比较，若自己的大则发送仲裁号，否则不予响应。

1. 冒险：结构冒险，数据冒险，控制冒险

结构冒险：取操作数和取数据阶段均用到寄存器，并行时会冲突。（设置多寄存器）

数据冒险：ALU计算时数据未及时存入，所取得数据为无效数据。（ALU直通法，设置等待周期）

控制冒险：在跳转指令执行时，流水线上的指令将无效。