每条指令对应一个唯一的操作码

|  |  |
| --- | --- |
| 指令集结构 |  |
| 指令  (机器指令) | 一条指令是一组二进制代码  计算机执行某种操作的命令，是计算机运行的最小功能单位    一个指令是一个机器代码如：10010011，那么这是一个8bit(1B)的指令  除了单字节的指令，还有双字节的、三字节的指令    一个指令代表CPU的一个操作，执行该指令就表示执行其对应的操作  程序是若干指令的有序集合，程序运行的过程就是按一定的次序逐条执行指令的过程；  举例说明一个程序如：  C:\Users\ADMINI~1\AppData\Local\Temp\msohtmlclip1\02\clip_image001.png  这是一个由3个指令构成的程序，一般一个程序的指令会放在相邻的存储单元中  如假设指令1放在程序存储器中地址为11H的存储单元中，那么指令2就在12H，指令3在15H |
| 指令系统 | 一台计算机的所有指令的集合构成该计算机的指令系统，也称为指令集  指令集位于硬件与软件的交集 |
| 基本格式 | ……    操作码：位于一条指令的高位部分，表示该指令代表的操作和功能，操作码区分指令的类型，任意两种指令的操作码必然不同  地址码：位于一条指令的低位部分  操作数的地址  地址码可能包含多个地址，即该指令包含多个操作数  设某个操作数地址是n位，那么该操作数的寻址范围就是 |
| 指令长度  (指令字长) | 一条指令所包含的二进制代码的bit数  指令字长取决于：操作码的长度、操作数的地址码的长度、操作数地址的个数  因为主存一般是按字节编址，为了便于编址和取指，指令长度多为字节的整数倍    指令长度与机器字长没有固定的关系：  指令长度与地址个数没有固定的关系；即使是单地址指令可能由于单地址的寻址方式的不同而导致指令长度的不同    单字长指令：指令长度等于机器字长  半字长指令：指令长度等于半个机器字长  双字长指令：指令长度等于两个机器字长    定长指令字结构：在一个指令系统中，若所有指令的长度都是相等的，执行速度快，控制简单  变长指令字结构：各种指令长度随指令的功能而异 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 分类 | 零地址 | 操作码OP | | | | | | | | | 空操作指令、停机指令、关中断指令  零地址运算指令：仅用在堆栈计算机中  参与运算的两个操作数从栈顶和次栈顶弹出，送到运算器进行运算，结果再压入堆栈中 |
| 单地址 | OP | 寻址特征码M | | 地址码A | | | | | | 从该地址读取目的操作数  单操作数指令：执行单操作数运算，结果仍保存到该地址  双操作数指令：使用隐含寻址得到源操作数，执行运算，结果可放在该地址，也可放在隐含地址 |
|  | 表示地址码A的寻址方式 | | 目的操作数、结果 | | | | | |
| 双地址 | OP | M1 | A1 | M2 | A2 | | | | | 算数和逻辑运算指令 |
|  |  | 目的操作数、结果 |  | 源操作数 | | | | |
| 三地址 | OP | M1 | A1 | M2 | A2 | M3 | A3 | | |  |
|  |  | 目的操作数 |  | 源操作数 |  | 结果 | | |  |
| 四地址 | OP | M1 | A1 | M2 | A2 | M3 | A3 | M4 | A4 |  |
|  |  | 目的操作数 |  | 源操作数 |  | 结果 |  | 下址 | 下址：下一条要执行的指令的地址 |

|  |  |
| --- | --- |
| 定长操作码 | n位定长操作码：该指令系统最大包含条指令  优点：简化硬件设计，提高指令译码和识别速度 |
| 扩展操作码 | 指令按操作码的位数分类：  位操作码指令，个，共 种，从中选取种  位操作码指令，个，共 种，从中选取种  …………  操作码的高位不能和位操作码指令的操作码相同  优点：在指令字长固定的前提下，保持丰富的指令种类；在满足需要的前提下，有效地缩短指令字长  缺点：增加指令译码和分析的难度，控制器的设计复杂化  注意：  不允许短码是长码的前缀，即短操作码不能与长操作码的前面部分的代码相同  通常，使用频率较高的指令，分配短操作码  使用频率较低的指令，分配长操作码  这样可以减少指令译码和分析的时间  各指令的操作码一定不能重复 |

运算性指令寻址的是操作数

转移性指令寻址的是下次与执行的指令的地址

不同的计算机，操作类型也不同，即该计算机有哪些操作，各种操作对应的操作码是什么，这都是由计算机本身决定的

下面介绍的是几类典型的操作

|  |  |  |
| --- | --- | --- |
| 运算 | 算数运算 | 加、减、乘、除、增1、减1、比较、求补  高档机还支持浮点运算、十进制运算等 |
| 逻辑运算 | 与、或、非、异或等 |
| 位运算 | 位测试、位清除、位求反，等 |
| 移位 | 算数移位 | 对有符号数和无符号数的左移(乘) 和 右移(除以) |
| 逻辑移位 | 对有符号数和无符号数的左移(乘) 和 右移(除以) |
| 循环移位 |  |
| 转移 | 无条件转移 |  |
| 条件转移 | 条件码 |
| 调用 |  |
| 返回 |  |
| 陷阱 |  |
| 其他 |  |  |
|  |  |
| 特权指令 | 这是仅用于操作系统或其他系统软件的指令。  为确保系统与数据的安全，这类指令不提供给用户使用 |

## 指令执行

|  |  |
| --- | --- |
|  | 计算机工作的过程，就是执行指令的过程 |
|  | 解释指令：取值、分析、、取操作数、执行、保存结果  保证指令的有序执行    完成一条指令：  取指令：PC寄存器：存放当前欲执行指令的地址，具有技术功能  分析指令IR：存放当前欲执行的指令  执行指令：CU |
| 控制信号 | 1bit的控制信号，1表示有效，0表示无效  多bit的控制信号，各个值对应不同的操作    控制信号只在一个时钟周期内有效，因此使能某个信号后，不必关闭，一个时钟周期结束，信号自动关闭  但如果要保留这个信号产生的数据，则需要将输入保存到寄存器中 |
| 微操作 | 微操作计算机中最基本的、不可再分解的操作  一台计算机有哪些微操作是确定的，这是由硬件决定的  计算机工作本质：选择微操作，并执行 |
| 指令 | 指令是一个固定的微操作序列  所有指令构成计算机的指令系统，是确定的，其包含的指令也是固定不变的，因此  根据计算机的指令系统列出所有的微操作  指令数据称为指令字 |
| 控制信号 | 控制信号是微操作的具体实施 |
| 时钟周期 | 使用机器主振电路，如晶体振荡器发出的脉冲信号经整形(倍频或分频)，产生时钟信号  时钟信号的频率就是CPU的主频  时钟周期是CPU操作的最基本单位  在一个时钟周期内机器执行一个微操作，或几个同时执行的微操作  因为逻辑器件仅在时钟上升沿改变状态 |
| 指令周期 | 指令周期：一条指令，从取指令开始，到指令执行结束，所经历的时间 |
| 单周期CPU | 指令周期=时钟周期  每条指令的指令周期固定  时钟周期长度等于 |
| 多周期CPU | 指令周期=N×时钟周期 每条指令的指令周期不固定 |

|  |  |  |
| --- | --- | --- |
| 微操作 | 内容 | 数据传输 |
| (PC)+1→PC | (PC)+1 | PC寄存器的值，加1 |
| (PC)→MAR | 指令地址 | PC→片内总线→MAR→系统地址总线→内存 |
| (IR)→CU | 操作码 | IR→片内总线→CU |
| (IR)→MAR | 地址码 | IR→片内总线→MAR→系统地址总线→内存 |
| M(MAR)→MDR | 数据地址  数据 | MAR→系统地址总线→内存  内存→系统数据总线→MDR |
| (MDR)→IR | 指令 | 内存→系统数据总线→MDR→片内总线→IR |
| (MDR)→ACC | 操作数 | MDR→片内总线→ACC |
| (MDR)→M(MAR) |  | MAR→系统地址总线→主存  MDR→系统数据总线→主存 |
| MemRead | 读命令 | CU→系统控制总线→内存 |
| MemWrite | 写命令 | CU→系统控制总线→内存 |
| (ACC)→MDR |  | ACC→片内总线→MDR |
| (R)→MDR |  |  |

临时寄存器A和B用于暂存从寄存器中读出的操作数

T0表示第一个时钟周期，T1表示第二个时钟周期，……

(R)：表示寄存器R，存储的数据

M[R]：表示以寄存器R，存储的数据，作为地址，的内存单元，存储的数据

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | T0 | T1 | T2 | T3 | T4 | T5 |
| 取指周期  指令字长=存储字长 | (PC)→MAR  MemRead | M(MAR)→MDR | (MDR)→IR  OP(IR)→CU | (PC)+1→PC |  |  |
| 取指周期  指令字长=存储字长×2  需要两次访存 | PC→MAR  MemRead | M(MAR)→MDR  (PC)+1→PC | (MDR)→IR  OP(IR)→CU | PC→MAR  MemRead | M(MAR)→MDR  (PC)+1→PC | MDR→IR  OP(IR)→CU |
| 间址周期  一次间址周期 | IR→MAR  MemRead | M(MAR)→MDR | (MDR)→Ad(IR) |  |  |  |
| 间址周期  二次间址周期 | Ad(IR)→MAR  MemRead | M(MAR)→MDR | (MDR)→Ad(IR) | Ad(IR)→MAR  MemRead | M(MAR)→MDR | MDR→Ad(IR) |
| 取操作数  LDA X | Ad(IR)→MAR  MemRead | M(MAR)→MDR | (MDR) →ACC |  |  |  |
| 存操作数  STA X | Ad(IR)→MAR  MemWrite | ACC→MDR | MDR→M(MAR) |  |  |  |
| 中断周期 |  |  |  |  |  |  |

## 控制信号

|  |  |  |
| --- | --- | --- |
| 控制信号与微操作 | 控制信号又称为微操作命令或微命令；  在计算机系统中，每一个微操作，对应一个控制信号，前面已经讲到机器有哪些微操作也是确定的，所以机器有哪些控制信号也是确定的。  微操作是微命令的执行过程，所有微操作的执行都是在控制单元发出的控制信号(微操作命令)控制下完成的，即先由控制单元发出控制信号，再执行微操作。 | |
| 相容性与互斥型 | 相容性微命令：对应的微操作可以同时执行  互斥性微命令：对应的微操作不可以同时执行  相容和互斥都是相对的，一个微命令可以和一些微命令相容，和另一些微命令互斥 | |
| 控制信号的作用 | 控制信号在每个时钟周期的上升沿，由控制单元CU发出；  控制信号必须按一定次序发出，才能使机器有序的工作。  其具体的实现方式可以见《控制单元》 | |
| 控制信号 | 不采用CPU内部总线 | CPU内各个部件之间直接相连，  控制信号控制门电路，门电路控制部件之间的数据传输的通断  如：表示控制MAR，的控制信号  有效：打开MAR送往地址总线的输出门，执行微操作-M(MAR)àMDR |
| 采用CPU内部总线 | ：控制信号控制寄存器到总线或总线到寄存器的输入控制门  ：控制从程序计数器到内部总线的输出控制门  表示输入控制，表示输出控制    与不采用片内总线相比，采用片内总线的方式，多用了两个寄存器：Y和Z  这是由于ALU是一个组合逻辑电路，再起运算过程中必须保持两个输入端不变，其中一个输入可以从Y寄存器中获得，另一个输入可以从内部总线上获得。  当CPU内有多个通用寄存器时，由于设置了寄存器Y，可实现任意两个寄存器之间的算术逻辑运算  此外，ALU的输出不能直接与内部总线相连，因为其输出又会通过总线反馈到ALU的输入，影响运算的正确性，故用寄存器Z暂存运算结果，再根据需要送至指定的目标 |
| 读/写信号 | CU发送至内存的控制信号 |

## 微操作与时序

|  |  |
| --- | --- |
| 微操作 | 微操作是计算机中最基本的、不可再分解的操作  每条指令都可以看作一个微操作序列  由于计算机的指令系统是确定的，其包含的指令也是固定不变的，因此机器有哪些微操作也是确定的  根据计算机的指令系统列出所有的微操作  那么计算机工作其实就是从这些微操作中进行选择并执行。 |
| 时钟周期 | 使用机器主振电路，如晶体振荡器发出的脉冲信号经整形(倍频或分频)，产生时钟信号  时钟信号的频率就是CPU的主频  时钟周期是CPU操作的最基本单位  在一个时钟周期内机器执行一个微操作，或几个同时执行的微操作 |

取指周期

指令字长一般都是取存储字长的整数倍；

定长指令字结构，所有指令的取指操作是相同的；

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令字长=存储字长 | 微操作 |  |  |  |
|  | 节拍 | 传输内容 | 传输路径 | 记作 |
|  | T0 | 指令地址 | PCà片内总线àMAR | PCàMAR |
|  |  | 读命令 | CUà系统控制总线à主存 | 1àR |
|  | T1 | 存储地址 | MARà系统地址总线à主存 | M(MAR)àMDR |
|  |  | 存储数据 | 主存à系统数据总线àMDR |  |
|  |  | (PC)+1 | PC | (PC)+1àPC |
|  | T2 | 存储数据 | MDRà片内总线àIR | MDRàIR |
|  |  | 操作码 | IRà片内总线àCU | OP(IR)àCU |
|  |  |  |  |  |
| 指令字长=存储字长×2  需要两次访存 | 微操作 |  |  |  |
|  | 节拍 | 传输内容 | 传输路径 | 记作 |
|  | T0 | 指令地址 | PCà片内总线àMAR | PCàMAR |
|  |  | 读命令 | CUà系统控制总线à主存 | 1àR |
|  | T1 | 存储地址 | MARà系统地址总线à主存 | M(MAR)àMDR |
|  |  | 存储数据 | 主存à系统数据总线àMDR |  |
|  |  | (PC)+1 | PC | (PC)+1àPC |
|  | T2 | 存储数据 | MDRà片内总线àIR | MDRàIR |
|  |  | 操作码 | IRà片内总线àCU | OP(IR)àCU |
|  |  |  |  |  |
|  | T3 | 指令地址 | PCà片内总线àMAR | PCàMAR |
|  |  | 读命令 | CUà系统控制总线à主存 | 1àR |
|  | T4 | 存储地址 | MARà系统地址总线à主存 | M(MAR)àMDR |
|  |  | 存储数据 | 主存à系统数据总线àMDR |  |
|  |  | (PC)+1 | PC | (PC)+1àPC |
|  | T5 | 存储数据 | MDRà片内总线àIR | MDRàIR |
|  |  | 操作码 | IRà片内总线àCU | OP(IR)àCU |

间址周期(分析指令)

触发标志：IND=1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 一次间址周期 | 微操作 |  |  |  |
|  | 节拍 | 信号内容 | 传输路径 | 记作 |
|  | T0 | 形式地址码 | IRà片内总线àMAR | Ad(IR)àMAR |
|  |  | 读命令 | CUà系统控制总线à主存 | 1àR |
|  | T1 | 形式地址码 | MARà系统地址总线à主存 | M(MAR)àMDR |
|  |  | 有效地址码 | 主存à系统数据总线àMDR |  |
|  | T2 | 有效地址码 | MDRà片内总线àIR | MDRàAd(IR) |
|  |  |  |  |  |
| 二次间址周期 | 微操作 |  |  |  |
|  | 节拍 | 信号内容 | 传输路径 | 记作 |
|  | T0 | 形式地址码1 | IRà片内总线àMAR | Ad(IR)àMAR |
|  |  | 读命令 | CUà系统控制总线à主存 | 1àR |
|  | T1 | 形式地址码1 | MARà系统地址总线à主存 | M(MAR)àMDR |
|  |  | 形式地址码2 | 主存à系统数据总线àMDR |  |
|  | T2 | 形式地址码2 | MDRà片内总线àIR | MDRàAd(IR) |
|  |  |  |  |  |
|  | T3 | 形式地址码2 | IRà片内总线àMAR | Ad(IR)àMAR |
|  |  | 读命令 | CUà系统控制总线à主存 | 1àR |
|  | T4 | 形式地址码2 | MARà系统地址总线à主存 | M(MAR)àMDR |
|  |  | 有效地址码 | 主存à系统数据总线àMDR |  |
|  | T5 | 有效地址码 | MDRà片内总线àIR | MDRàAd(IR) |

指令的间址周期由一次间址、二次间址、多次间址，因此它们的操作是不同的

存储器间址通过形式地址访存

寄存器间址通过寄存器内容访存

在间址周期阶段，间址次数可以间址周期标志触发器IND确定，当它为0时，表示间址寻址结束

执行周期

触发标志：EX=1

在执行周期开头，IR存储的地址码必然是有效地址

不同指令的执行周期的微操作不同，计算机在设计时可以根据需求创建指令，即创建指令所包含的微操做；

下面是一些常用指令的执行周期的微操作序列

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令 | 微操作 |  |  |  |
|  | 节拍 | 信号内容 | 传输路径 | 记作 |
| 取操作数  LDA X | T0 | 有效地址码 | IRà片内总线àMAR | Ad(IR)àMAR |
|  |  | 读命令 | CUà系统控制总线à主存 | 1àR |
|  | T1 | 有效地址码 | MARà系统地址总线à主存 | M(MAR)àMDR |
|  |  | 操作数 | 主存à系统数据总线àMDR |  |
|  | T2 | 操作数 | MDRà片内总线àACC | MDRàACC |
|  |  |  |  |  |
| 存操作数  STA X | T0 | 有效地址码 | IRà片内总线àMAR | Ad(IR)àMAR |
|  |  | 写命令 | CUà系统控制总线à主存 | 1àW |
|  | T1 | 操作数 | ACCà片内总线àMDR | ACCàMDR |
|  | T2 | 有效地址码 | MARà系统地址总线à主存 | MDRàM(MAR) |
|  |  | 操作数 | MDRà系统数据总线à主存 |  |
|  |  |  |  |  |
| 加法指令  ADD X | T | 目标操作数 | ACCà片内总线àALU | (ACC)+(MDR)àACC |
|  |  | 源操作数 | MDRà片内总线àALU |  |
|  |  | 结果 | ALUà片内总线àACC |  |
|  |  |  |  |  |
| 无条件转移  JMP X | T | 指令地址 | Yà片内总线àPC | Ad(IR)àPC |
| 条件(负)转移  BAN X | T |  |  | àPC |
| 条件(零)转移  BAZ X | T | 指令地址 | Yà片内总线àPC | àPC |
|  |  |  |  |  |
| 清除累加器  CLA | T | 0 | ACC | 0àACC |
| 累加器取反  COM | T |  | ACC | àACC |
| 算数右移一位  SHR | T |  | ACC | L(ACC)àR(ACC)，à(符号位不变) |
| 循环左移一位  CSL | T |  | ACC | R(ACC)àL(ACC)，à[或] |
| 停机  STP | T | 0 | G | 0àG |

中断周期

触发标志：INT=1

中断周期(如果有的话)，被放在一个指令周期的末尾

在中断周期开头，CPU查询是否有中断请求，如果有则进入中断周期

在中断周期中，中断隐指令会自动完成保护断点、寻找中断服务程序入口地址以及硬件关中断的操作

当CPU采用中断方式其进行信息传输时，CPU在每条指令执行结束前，都要发中断查询信号，如果有中断请求信号，就会进入中断响应阶段，称为中断周期

|  |  |  |  |
| --- | --- | --- | --- |
| 微操作 |  |  |  |
| 节拍 | 信号内容 | 传输路径 | 记作 |
| T0 | (SP)-1 | SP | SPàMAR |
|  | 写命令 | CUà系统控制总线à主存 | 1àW |
| T1 | 程序断点 | PCà片内总线àMDR | PCàMDR |
| T2 | 存储器地址 | MARà系统地址总线à主存 | MDRàM(MAR) |
|  | 程序断点 | MDRà系统数据总线à主存 |  |
|  | 0 | G | 向量地址àPC |
|  | 0 | EINT | 0àEINT |

T6时刻关闭中断，将允许中断触发器清0，该操作可直接由硬件线路完成

|  |  |
| --- | --- |
|  | T1时刻，把用于保存程序断点的存储器特殊地址  这里使用堆栈来保存程序断点的地址  进栈操作是将SP减1，这和传统意义上的进栈操作相反，因为计算机的堆栈中都是向低地址增加，所以进栈操作是减1不是加1 |

## 结构与控制

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PC | 程序计数器  存放现行指令的地址，位长取决于存储器容量  通常具有计数功能 | PCIe | PC输入使能 | |
| PCOeH | PC寄存器值，高4位，输出使能 | |
| PCOeL | PC寄存器值，低28位，输出使能 | |
| IR | 指令寄存器  存放现行指令的数据 | IRIe | IR输入使能 | |
| MAR | 存储器地址寄存器  存放将被访问的存储单元的地址 | MARIe | MAR输入使能 | |
| MDR | 存储器数据寄存器  存放将被存入存储器中的数据或  最近从存储器中读出的数据 | MDRSrc | MDR输入源选择 | |
| MDRIe | MDR输入使能 | |
| MDROe | MDR输出使能 | |
| Reg | 由程序设计者指定功能  有些机器不适用通用寄存器，而是使用专用寄存器作为基址寄存器、变址寄存器或栈指针，这么做的话，在设计指令格式的时候只需要将这类专用寄存器隐含在操作码中，而不必占用指令中的位，见p314  如p322，7.21 所示，由于用通用寄存器作为变址寄存器和基址寄存器，股灾指令中设有X和B字段，分别指出作为变址寄存器和基址寄存器的通用寄存器编号   * 累加寄存器ACC：暂时存放ALU预算的结果信息，作为加法运算的一个输入端 * 堆栈指针SP：：满足某种寻址方式所需 * 临时寄存器A：存放操作数存放地址 * 临时寄存器B：存放操作数存放地址 * 基址寄存器：满足某种寻址方式所需 * 变址寄存器：满足某种寻址方式所需 * 在寄存器间接寻址中还可以使用通用寄存器存放有效地址的地址 | AIe | | A暂存器输入使能 |
| BIe | | B暂存器输入使能 |
| ImmOeH | | 立即数，高4位，输出使能 |
| ImmOeL | | 立即数，低28位，输出使能 |
| ExtSel | | 00：将16位的立即数符号扩展为32位 01：将16位的立即数符号扩展为32位，再左移2位  10：输出立即数4  11：IR[25-0]<<2 |
| RegSel | | 通用寄存器输入源选择 |
| RegIe | | 通用寄存器输入使能 |
| RegOe | | 通用寄存器输出使能 |
| ALU | 算数逻辑单元  进行算数/逻辑运算  在CPU内部必须给ALU提供数据，因此ALU必须可直接访问MDR和用户可见的寄存器  ALU外围还可以有另一些寄存器，这些寄存器用于ALU的输入/输出以及用于和MDR及用户可见寄存器交换数据 | ALUOp | | ALU操作 |
| ALUCtrl | | ALU操作控制 |
| ALUOe | | ALU输出使能 |
| Mem |  | MemOe | | 存储器输出使能 |
| MemRd | | 存储器读 |
| MemWr | | 存储器写 |