

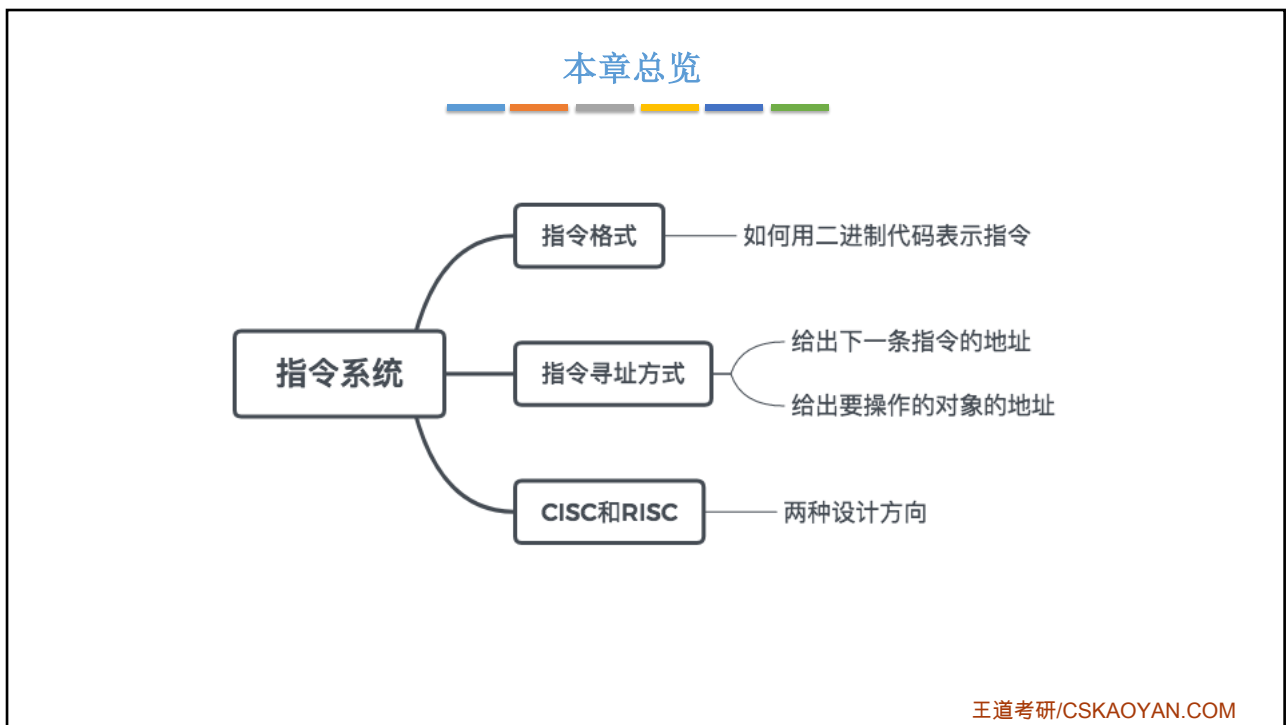
本节内容

指令系统

CISC和RISC

王道考研/CSKAOYAN.COM

1



2

类比：有很多库函数的C语言

CISC和RISC

类比：没有库函数的C语言

CISC: Complex Instruction Set Computer

设计思路：一条指令完成一个复杂的基本功能。

代表：x86架构，主要用于笔记本、台式机等

RISC: Reduced Instruction Set Computer

设计思路：一条指令完成一个基本“动作”；
多条指令组合完成一个复杂的基本功能。

代表：ARM架构，主要用于手机、平板等

80-20规律：典型程序中 80% 的语句仅仅使用处理器中 20% 的指令

比如设计一套能实现整数、矩阵加/减/乘运算的指令集：

CISC的思路：除了提供整数的加减乘指令之外，还提供矩阵的加法指令、矩阵的减法指令、矩阵的乘法指令

RISC的思路：只提供整数的加减乘指令

一条指令可以由一个专门的电路完成

一条指令一个电路，电路设计相对简单，功耗更低

有的复杂指令用纯硬件实现很困难

“并行”、“流水线”

→ 采用“存储程序”的设计思想，由一个比较通用的电路配合存储部件完成一条指令

王道考研/CSKAOYAN.COM

3

CISC和RISC

| 对比项目 \ 类别 | CISC | RISC |
|-----------|--------------------|--------------------|
| 指令系统 | 复杂，庞大 | 简单，精简 |
| 指令数目 | 一般大于200条 | 一般小于100条 |
| 指令字长 | 不固定 | 定长 |
| 可访存指令 | 不加限制 | 只有Load/Store指令 |
| 各种指令执行时间 | 相差较大 | 绝大多数在一个周期内完成 |
| 各种指令使用频度 | 相差很大 | 都比较常用 |
| 通用寄存器数量 | 较少 | 多 |
| 目标代码 | 难以用优化编译生成高效的目标代码程序 | 采用优化的编译程序，生成代码较为高效 |
| 控制方式 | 绝大多数为微程序控制 | 绝大多数为组合逻辑控制 |
| 指令流水线 | 可以通过一定方式实现 | 必须实现 |

王道考研/CSKAOYAN.COM

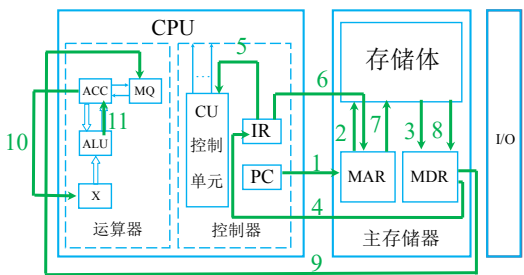
4

王道考 研,cskaoyan.com

2

计算机的工作过程

乘法指令可以访存，一定是CISC



| 主存地址 | 指令 | | 注释 |
|------|------------------|------------|----------------|
| | 操作码 | 地址码 | |
| 0 | 000001 | 0000000101 | 取数a至ACC |
| 1 | 000100 | 0000000110 | 乘b得ab,存于ACC中 |
| 2 | 000011 | 0000000111 | 加c得ab+c,存于ACC中 |
| 3 | 000010 | 0000001000 | 将ab+c,存于主存单元 |
| 4 | 000110 | 0000000000 | 停机 |
| 5 | 0000000000000010 | | 原始数据a=2 |
| 6 | 0000000000000011 | | 原始数据b=3 |
| 7 | 0000000000000001 | | 原始数据c=1 |
| 8 | 0000000000000000 | | 原始数据y=0 |

上一条指令取指后PC自动+1, (PC)=1; 执行后, (ACC)=2

- #1: (PC)→MAR, 导致(MAR)=1
- #3: M(MAR)→MDR, 导致(MDR)=000100 0000000110
- #4: (MDR)→IR, 导致(IR)= 000100 0000000110
- #5: OP(IR)→CU, 指令的操作码送到CU, CU分析后得知, 这是“乘法”指令
- #6: Ad(IR)→MAR, 指令的地址码送到MAR, 导致(MAR)=6
- #8: M(MAR)→MDR, 导致(MDR)=0000000000000011=3
- #9: (MDR)→MQ, 导致(MQ)=0000000000000011=3
- #10: (ACC)→X, 导致(X)=2
- #11: (MQ)*(X)→ACC, 由ALU实现乘法运算, 导致(ACC)=6, 如果乘积太大, 则需要MQ辅助存储

取指令 (#1~#4)
分析指令 (#5)
执行乘法指令 (#6~#11)

王道考研/CSKAOYAN.COM