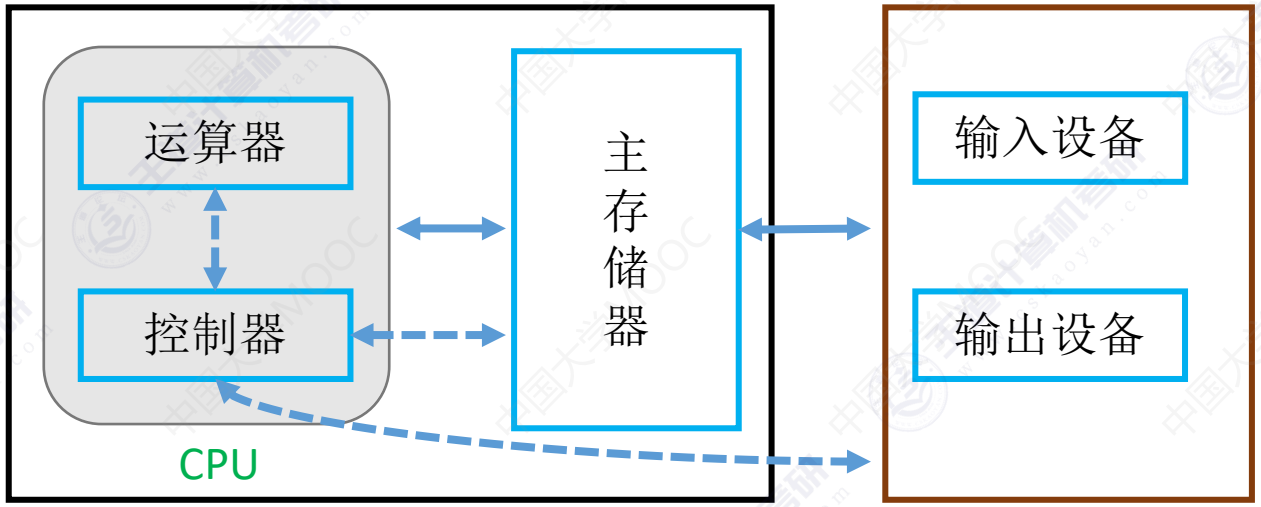
王道考研——组成原理

WWW.CSKAOYAN.COM

第四章 指令系统

现代计算机的结构

控制器！



主机

I/O设备（外设）

回忆：计算机的工作过程

高级语言

```
int a=2,b=3,c=1,y=0;
void main(){
  y=a*b+c;
}
```

我和你相爱在网络里
爱来爱去都变成回忆

编译
装入主存

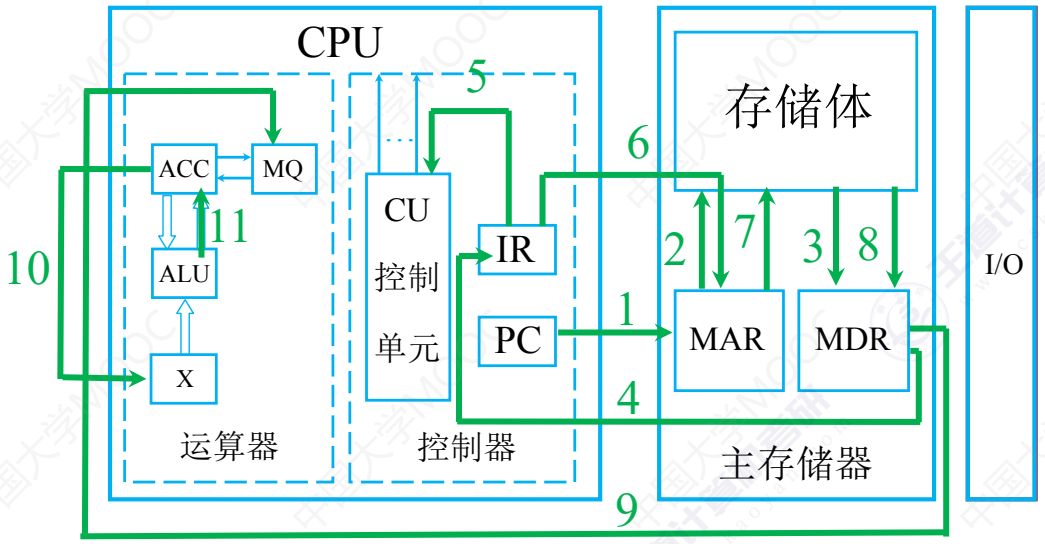
机器语言

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*

存储字长=16bit

回忆：计算机的工作过程

操作码：指明了“做什么”
地址码：指明了“对谁动手”



PC

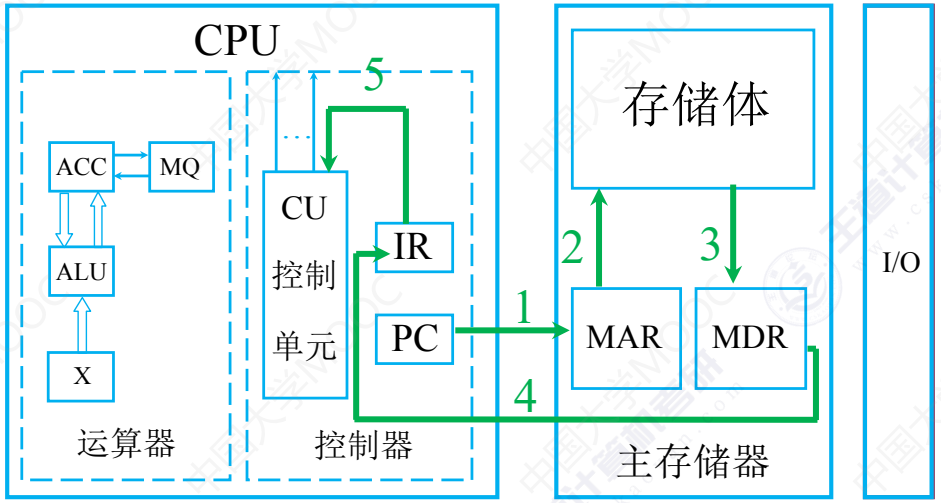
主存地址	指令		注释
	操作码	地址码	
0	000001	0000000101	取数*a*至ACC
1	000100	0000000110	乘*b*得*ab* ,存于ACC中
2	000011	0000000111	加*c*得*ab*+*c* ,存于ACC中
3	000010	0000001000	将*ab*+*c* ,存于主存单元
4	000110	0000000000	停机
5	*0000000000000010*		原始数据*a=2*
6	*0000000000000011*		原始数据*b=3*
7	*0000000000000001*		原始数据*c=1*
8	*0000000000000000*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

回忆：计算机的工作过程

操作码：指明了“做什么”
地址码：指明了“对谁动手”
有的指令不需要地址码（停机）



PC

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	*0000000000000111*		最终结果*y=7*

上一条指令取指后(PC)=4
#1：(PC)→MAR，导致(MAR)=3
#3：M(MAR)→MDR，导致(MDR)=000110 0000000000
#4：(MDR)→IR，导致(IR)= 000110 0000000000
#5：OP(IR)→CU，指令的操作码送到CU，CU分析后得知，这是“停机”指令

（利用中断机制通知操作系统终止该进程）

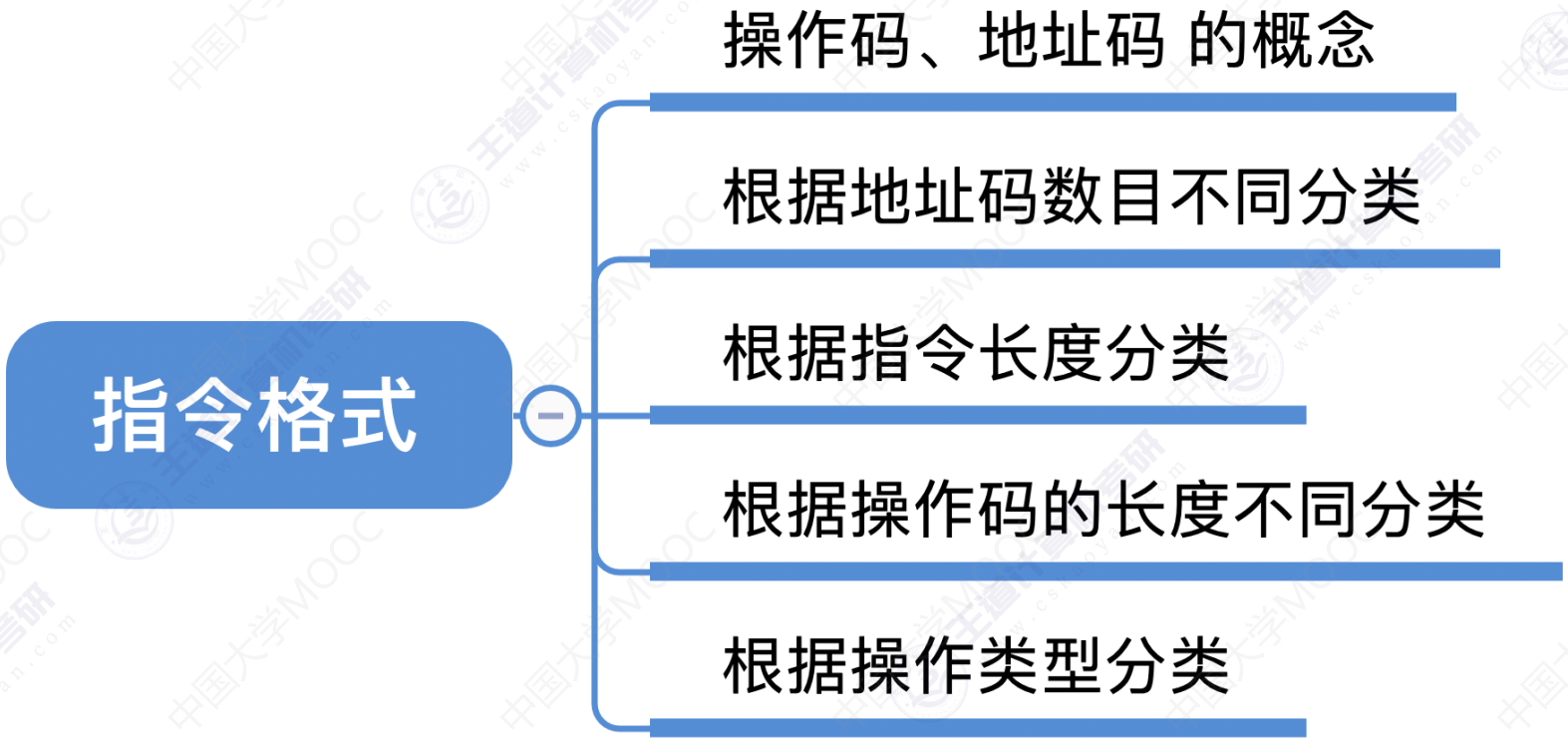
取指令（#1~#4）
分析指令（#5）
执行停机指令

本节内容

指令系统

指令格式

本节总览



指令的定义

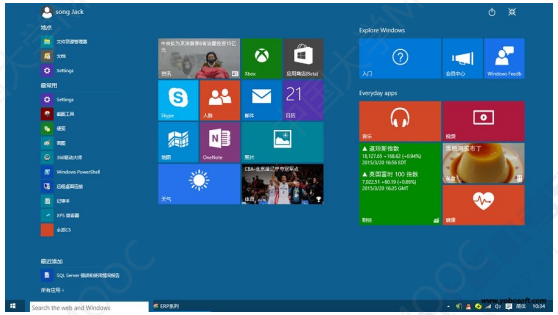
指令（又称机器指令）：

是指示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。

一台计算机的所有指令的集合构成该机的指令系统，也称为指令集。

注：一台计算机只能执行自己指令系统中的指令，不能执行其他系统的指令。

Eg：x86 架构、ARM架构



软件

10001000111110010011111110001010

指令

硬件

指令格式

一条指令就是机器语言的一个语句，它是一组有意义的二进制代码。

一条指令通常要包括操作码字段和地址码字段两部分：

一条指令可能包含 0个、1个、2个、3个、4个 地址码…
根据地址码数目不同，可以将指令分为 零地址指令、一地址指令、二地址指令…

零地址指令

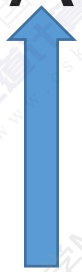
零地址指令 | OP

1. 不需要操作数，如空操作、停机、关中断等指令
2. 堆栈计算机，两个操作数隐含存放在栈顶和次栈顶，计算结果压回栈顶

数据结构：“后缀表达式”

A + B - C * D / E + F

A B + C D * E / - F +



零地址指令

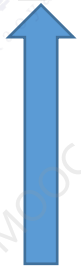
零地址指令 OP

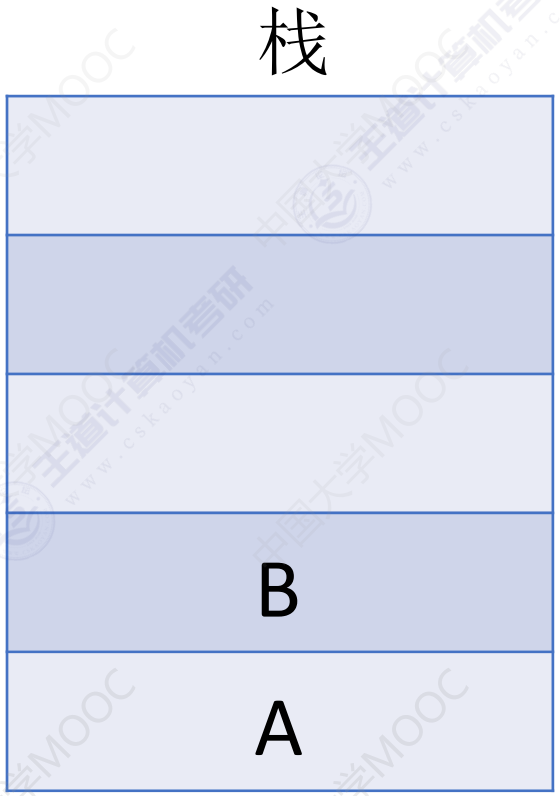
1. 不需要操作数，如空操作、停机、关中断等指令
2. 堆栈计算机，两个操作数隐含存放在栈顶和次栈顶，计算结果压回栈顶

数据结构：“后缀表达式”

A + B - C * D / E + F

A B + C D * E / - F +





零地址指令

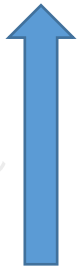
零地址指令 OP

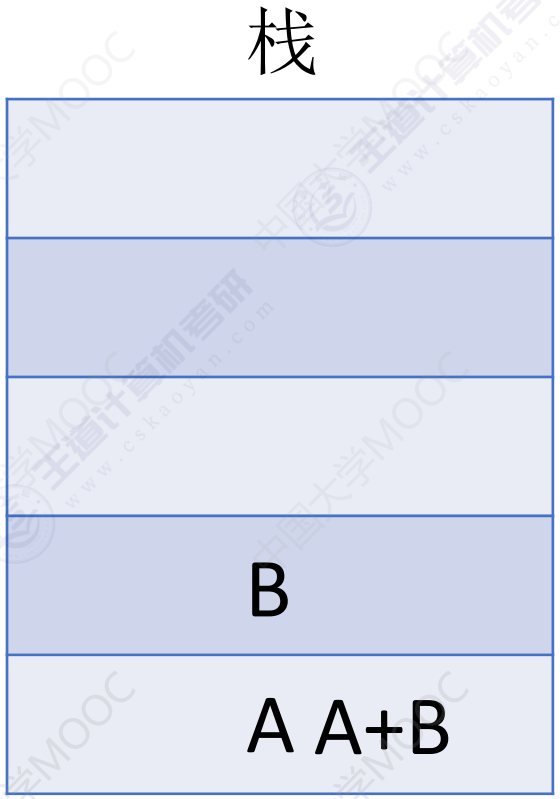
1. 不需要操作数，如空操作、停机、关中断等指令
2. 堆栈计算机，两个操作数隐含存放在栈顶和次栈顶，计算结果压回栈顶

数据结构：“后缀表达式”

A + B - C * D / E + F

A B + C D * E / - F +





零地址指令

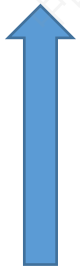
零地址指令 | OP

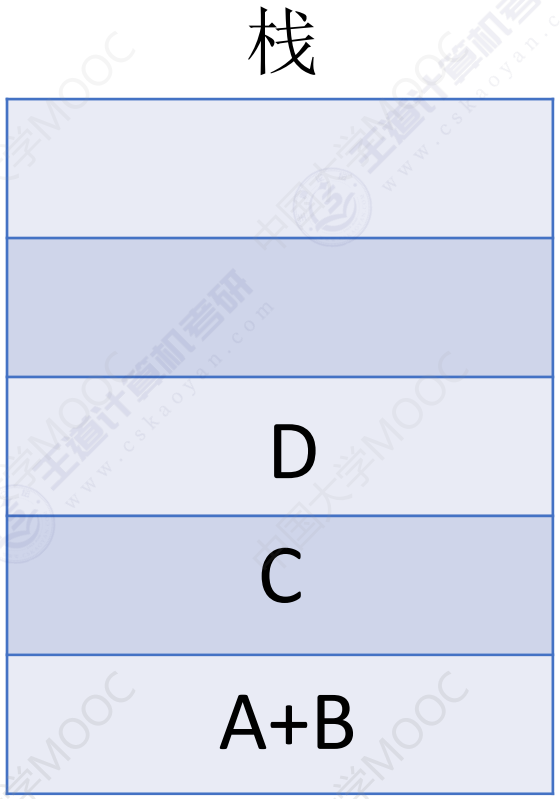
1. 不需要操作数，如空操作、停机、关中断等指令
2. 堆栈计算机，两个操作数隐含存放在栈顶和次栈顶，计算结果压回栈顶

数据结构：“后缀表达式”

A + B - C * D / E + F

A B + C D * E / - F +





C * D

一地址指令

一地址指令

1. 只需要单操作数，如加1、减1、取反、求补等

指令含义：$OP(A_1) \rightarrow A_1$，　完成一条指令需要3次访存：取指→ 读A1 →写A1

2. 需要两个操作数，但其中一个操作数隐含在某个寄存器（如隐含在ACC）

指令含义：$(ACC)OP(A_1) \rightarrow ACC$　　完成一条指令需要2次访存：取指→ 读A1

注：A_1 指某个主存地址，(A_1)表示 A_1 所指向的地址中的内容

类比：C语言指针

指针所指位置的内容

二、三地址指令

二地址指令

OP	A_1（目的操作数）	A_2（源操作数）

常用于需要两个操作数的算术运算、逻辑运算相关指令

指令含义：$(A_1)OP(A_2)\rightarrow A_1$

完成一条指令需要访存4次，取指→读A1 →读A2 →写A1

三地址指令

OP	A_1	A_2	A_3（结果）

常用于需要两个操作数的算术运算、逻辑运算相关指令

指令含义：$(A_1)OP(A_2)\rightarrow A_3$

完成一条指令需要访存4次，取指→读A1 →读A2 →写A3

四地址指令

四地址指令

OP	A_1	A_2	A_3（结果）	A_4（下址）

指令含义：$(A_1)OP(A_2)\rightarrow A_3$，$A_4$=下一条将要执行指令的地址

完成一条指令需要访存4次，取指→读A1 →读A2 →写A3

正常情况下：取指令之后 PC+1，指向下一条指令
四地址指令：执行指令后，将PC的值修改位 A_4 所指地址

地址码的位数有什么影响？

n位地址码的直接寻址范围=2^n

若指令总长度固定不变，则地址码数量越多，寻址能力越差

指令-按地址码数目分类

四地址指令

OP	A_1	A_2	A_3（结果）	A_4（下址）

指令含义：$(A_1)OP(A_2)\rightarrow A_3$，$A_4$=下一条将要执行指令的地址

三地址指令

OP	A_1	A_2	A_3（结果）

指令含义：$(A_1)OP(A_2)\rightarrow A_3$

二地址指令

OP	A_1（目的操作数）	A_2（源操作数）

指令含义：$(A_1)OP(A_2)\rightarrow A_1$

一地址指令

OP	A_1

指令含义：1. $OP(A_1)\rightarrow A_1$，如加1、减1、取反、求补等
2. $(ACC)OP(A_1)\rightarrow ACC$，隐含约定的目的地址为ACC

零地址指令

OP

指令含义：1. 不需要操作数，如空操作、停机、关中断等指令
2. 堆栈计算机，两个操作数隐含存放在栈顶和次栈顶，计算结果压回栈顶

指令-按指令长度分类

一般取字节的整数倍

指令字长：一条指令的总长度（可能会变）

机器字长：CPU进行一次整数运算所能处理的二进制数据的位数（通常和ALU直接相关）

存储字长：一个存储单元中的二进制代码位数（通常和MDR位数相同）

半字长指令、单字长指令、双字长指令　　——指令长度是机器字长的多少倍

指令字长会影响取指令所需时间。如：机器字长=存储字长=16bit，则取一条双字长指令需要两次访存

定长指令字结构：指令系统中所有指令的长度都相等

变长指令字结构：指令系统中各种指令的长度不等

指令-按操作码长度分类

定长操作码：指令系统中所有指令的操作码长度都相同

n位 → 2^n条指令

控制器的译码电路设计简单，但灵活性较低

可变长操作码：指令系统中各指令的操作码长度可变

控制器的译码电路设计复杂，但灵活性较高

定长指令字结构+可变长操作码 →扩展操作码指令格式

不同地址数的指令使用不同长度的操作码

指令—按操作类型分类

1. 数据传送

源　　目的

LOAD　作用：把**存储器**中的数据放到**寄存器**中

STORE　作用：把**寄存器**中的数据放到**存储器**中

数据传送类：进行主存与CPU之间的数据传送

2. 算术逻辑操作

算术：加、减、乘、除、增 1、减 1、求补、浮点运算、十进制运算

逻辑：与、或、非、异或、位操作、位测试、位清除、位求反

运算类

3. 移位操作

算术移位、逻辑移位、循环移位(带进位和不带进位)

4. 转移操作

程序控制类：改变程序执行的顺序

无条件转移 JMP

条件转移　JZ：结果为0；JO：结果溢出；JC：结果有进位

调用和返回 CALL和RETURN

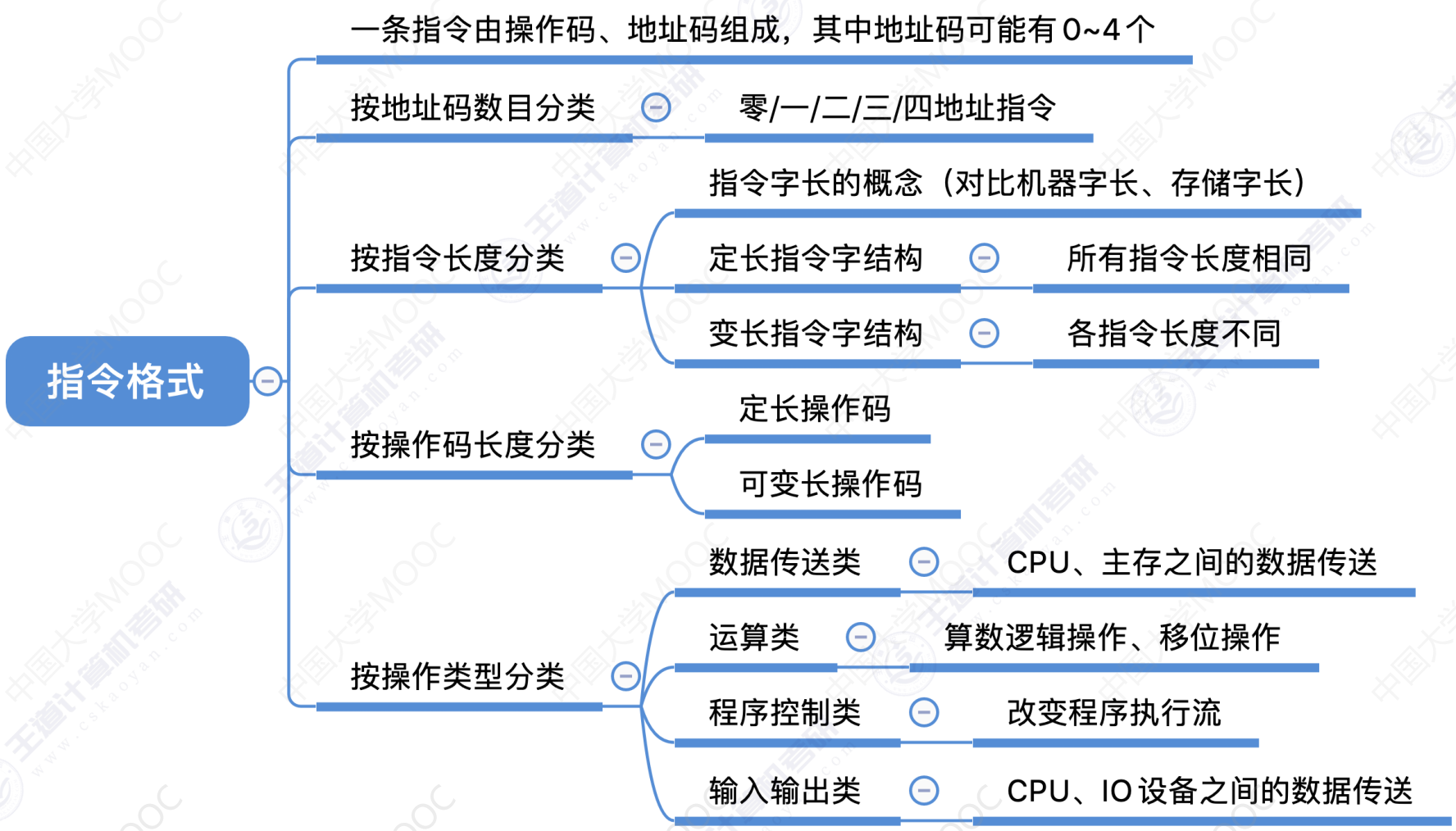
陷阱(Trap)与陷阱指令

5. 输入输出操作

输入输出类（I/O）：进行CPU和I/O设备之间的数据传送

CPU寄存器与IO端口之间的数据传送(端口即IO接口中的寄存器)

本节回顾



公众号：王道在线

b站：王道计算机教育

抖音：王道计算机考研