

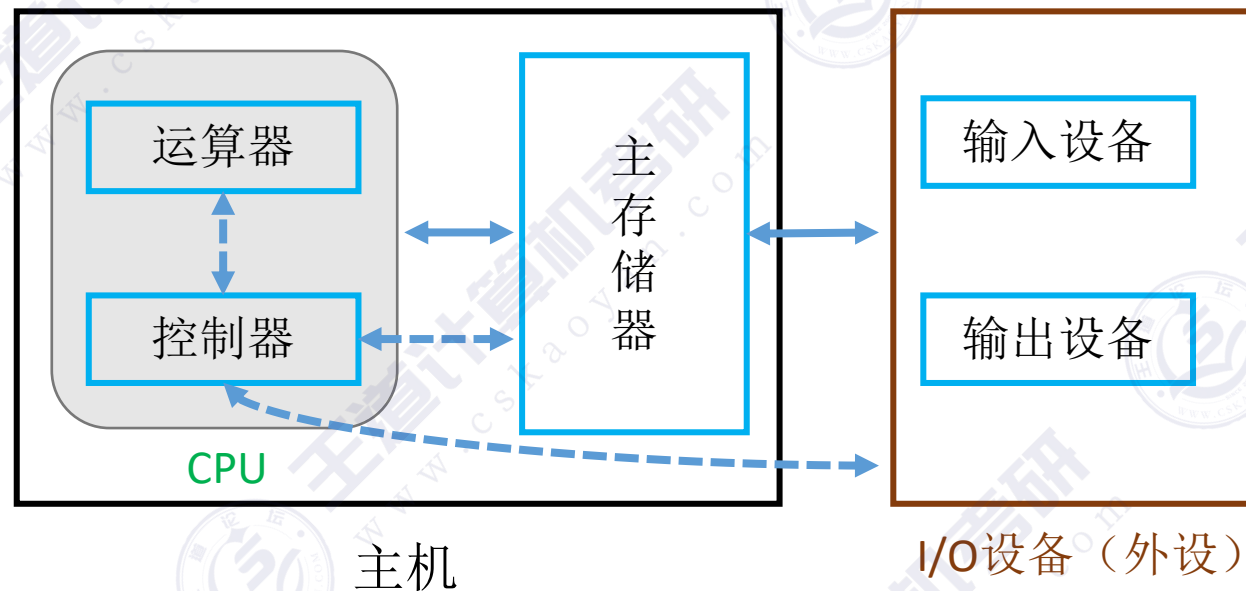
本节内容

各个硬件 的工作原理

知识总览



内部
细节



主存储器的基本组成

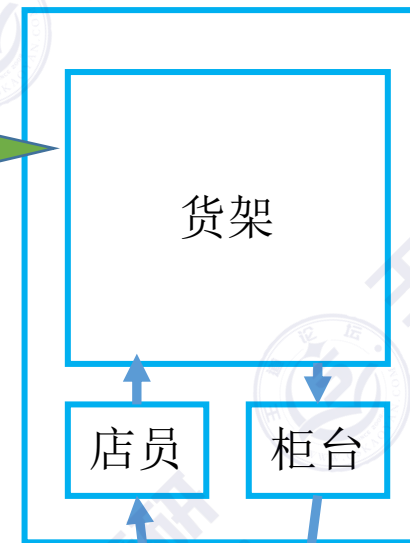
主存储器



Memory Address Register
(存储地址寄存器)

Memory Data Register
(存储数据寄存器)

菜鸟驿站



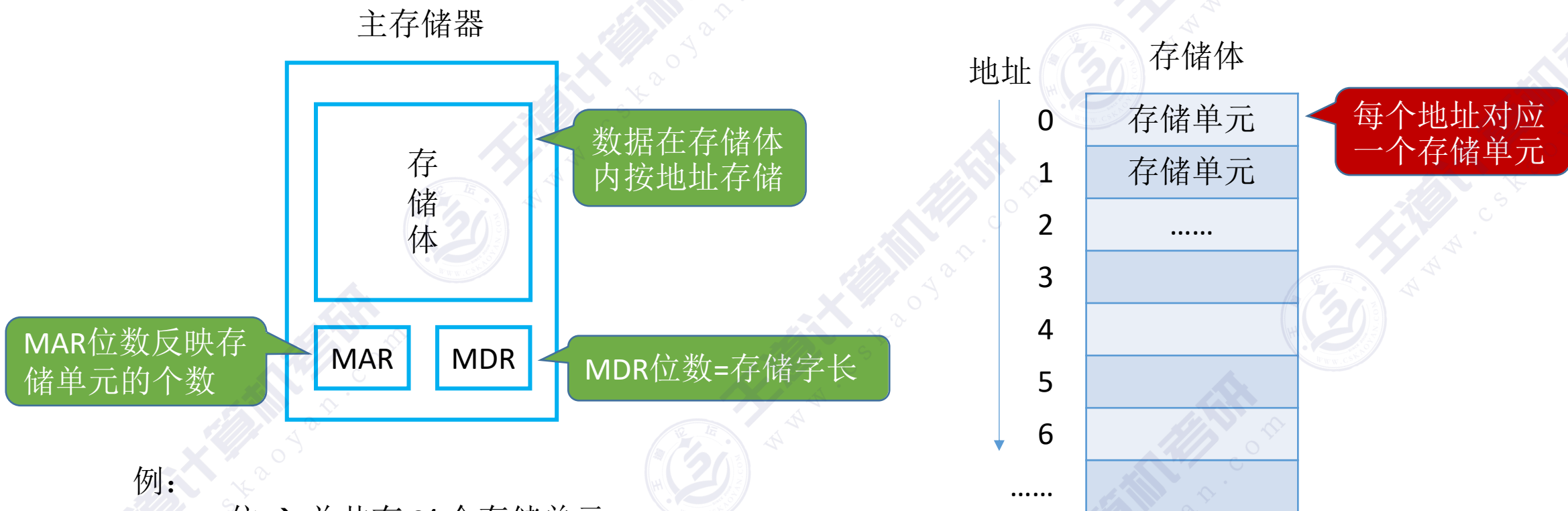
包裹在货架上
按取件号摆放

取件号

包裹



主存储器的基本组成



例:

MAR=4位 → 总共有 2^4 个存储单元

MDR=16位 → 每个存储单元可存放16bit,

1个**字(word)** = 16bit

易混淆: 1个**字节(Byte)** = 8bit

1B=1个字节, 1b=1个bit

存储单元: 每个存储单元存放一串二进制代码

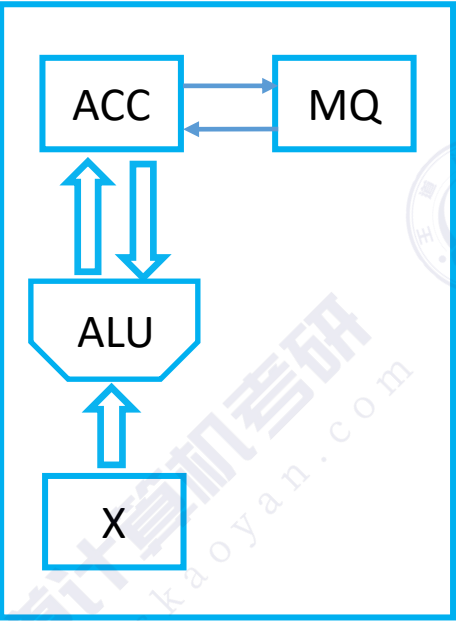
存储字(word): 存储单元中二进制代码的组合

存储字长: 存储单元中二进制代码的位数

存储元: 即存储二进制的电子元件, 每个存储元可存 1bit

运算器的基本组成

运算器



运算器：用于实现算术运算（如：加减乘除）、逻辑运算（如：与或非）

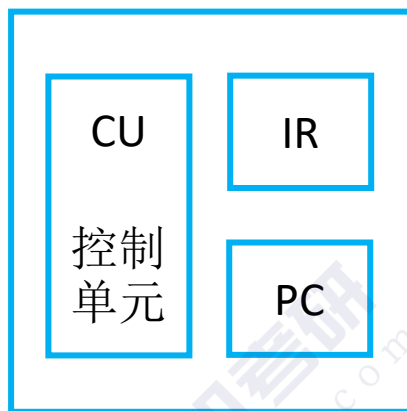
- ACC: 累加器，用于存放操作数，或运算结果。
- MQ: 乘商寄存器，在乘、除运算时，用于存放操作数或运算结果。
- X: 通用的操作数寄存器，用于存放操作数
- ALU: 算术逻辑单元，通过内部复杂的电路实现算数运算、逻辑运算

Accumulator
Multiple-Quotient Register
Arithmetic and Logic Unit

	加	减	乘	除
ACC	被加数、和	被减数、差	乘积高位	被除数、余数
MQ			乘数、乘积低位	商
X	加数	减数	被乘数	除数

控制器的基本组成

控制器



CU: 控制单元, 分析指令, 给出控制信号

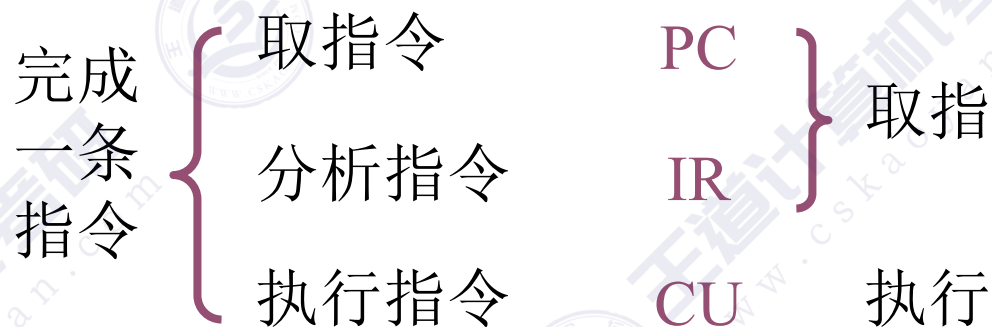
IR: 指令寄存器, 存放当前执行的指令

PC: 程序计数器, 存放下一条指令地址, 有自动加1功能

Control Unit

Instruction Register

Program Counter



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;
}
```

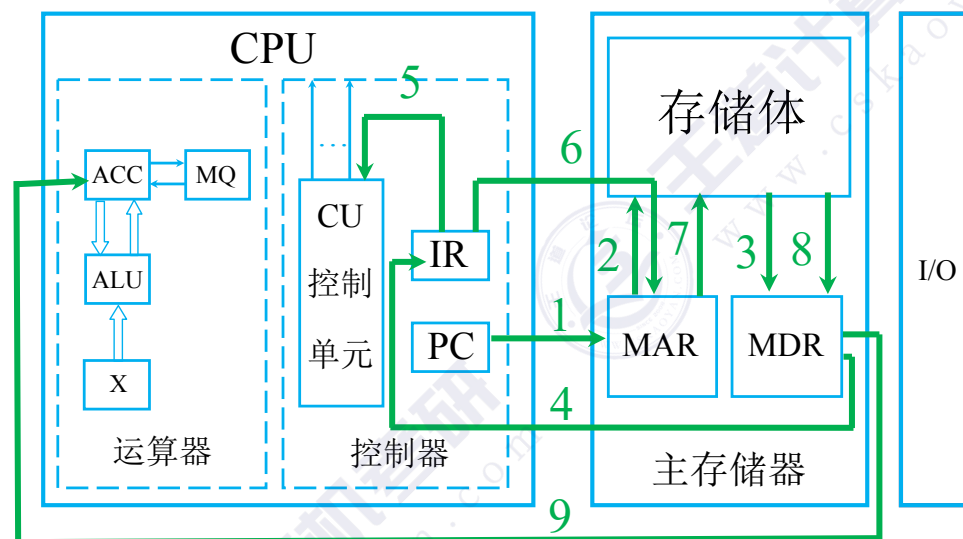
编译
装入主存

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

计算机的工作过程



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	00000000000000010		原始数据 $a=2$
6	00000000000000011		原始数据 $b=3$
7	00000000000000001		原始数据 $c=1$
8	00000000000000000		原始数据 $y=0$

初: $(PC)=0$, 指向第一条指令的存储地址

#1: $(PC) \rightarrow MAR$, 导致 $(MAR)=0$

#3: $M(MAR) \rightarrow MDR$, 导致 $(MDR)=000001\ 0000000101$

#4: $(MDR) \rightarrow IR$, 导致 $(IR)=000001\ 0000000101$

#5: $OP(IR) \rightarrow CU$, 指令的操作码送到CU, CU分析后得知, 这是“取数”指令

#6: $Ad(IR) \rightarrow MAR$, 指令的地址码送到MAR, 导致 $(MAR)=5$

#8: $M(MAR) \rightarrow MDR$, 导致 $(MDR)=00000000000000010=2$

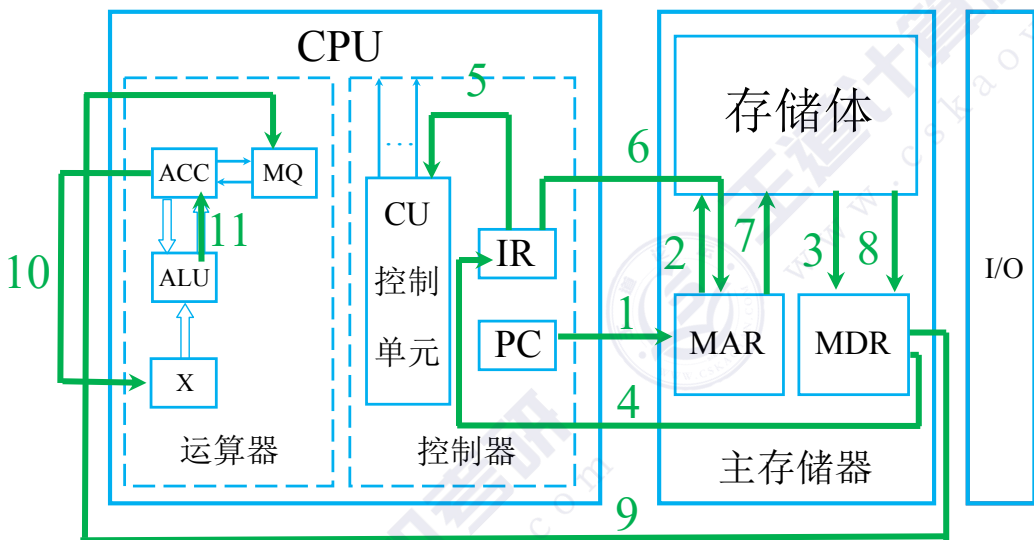
#9: $(MDR) \rightarrow ACC$, 导致 $(ACC)=00000000000000010=2$

取指令 (#1~#4)

分析指令 (#5)

执行取数指令 (#6 ~ #9)

计算机的工作过程



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	000000000000000010		原始数据 $a=2$
6	000000000000000011		原始数据 $b=3$
7	000000000000000001		原始数据 $c=1$
8	000000000000000000		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

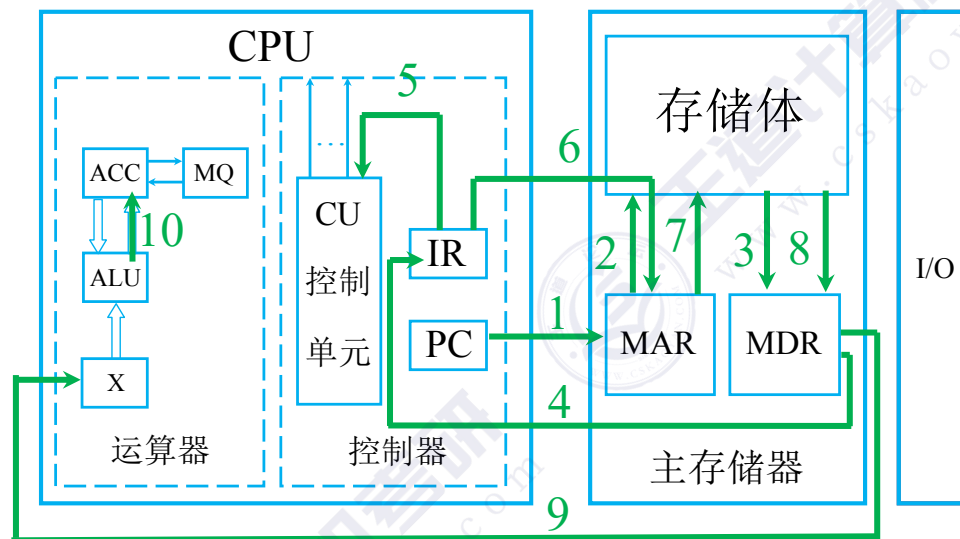
#11: (MQ)*(X)→ACC, 由ALU实现乘法运算, 导致(ACC)=6, 如果乘积太大, 则需要MQ辅助存储

取指令 (#1~#4)

分析指令 (#5)

执行乘法指令 (#6 ~ #11)

计算机的工作过程



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	000000000000000010		原始数据 $a=2$
6	000000000000000011		原始数据 $b=3$
7	000000000000000001		原始数据 $c=1$
8	000000000000000000		原始数据 $y=0$

上一条指令取指后(PC)=2, 执行后, (ACC)=6

#1: (PC)→MAR, 导致(MAR)=2

#3: M(MAR)→MDR, 导致(MDR)= 000011 0000000111

#4: (MDR)→IR, 导致(IR)= 000011 0000000111

#5: OP(IR)→CU, 指令的操作码送到CU, CU分析后得知, 这是“加法”指令

#6: Ad(IR)→MAR, 指令的地址码送到MAR, 导致(MAR)=7

#8: M(MAR)→MDR, 导致(MDR)=0000000000000001=1

#9: (MDR)→X, 导致(X)=0000000000000001=1

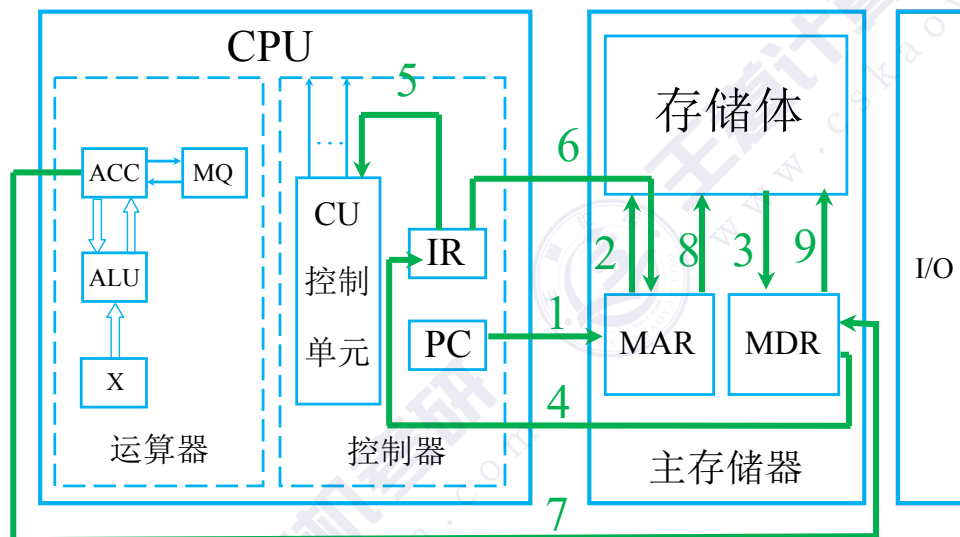
#10: (ACC)+(X)→ACC, 导致(ACC)=7, 由ALU实现加法运算

取指令 (#1~#4)

分析指令 (#5)

执行加法指令 (#6 ~ #10)

计算机的工作过程



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	00000000000000010		原始数据 $a=2$
6	00000000000000011		原始数据 $b=3$
7	00000000000000001		原始数据 $c=1$
8	00000000000000111		最终结果 $y=7$

上一条指令取指后 $(PC)=3$, 执行后, $(ACC)=7$

#1: $(PC) \rightarrow MAR$, 导致 $(MAR)=3$

#3: $M(MAR) \rightarrow MDR$, 导致 $(MDR)=000010\ 0000001000$

#4: $(MDR) \rightarrow IR$, 导致 $(IR)=000010\ 0000001000$

#5: $OP(IR) \rightarrow CU$, 指令的操作码送到CU, CU分析后得知, 这是“存数”指令

#6: $Ad(IR) \rightarrow MAR$, 指令的地址码送到MAR, 导致 $(MAR)=8$

#7: $(ACC) \rightarrow MDR$, 导致 $(MDR)=7$

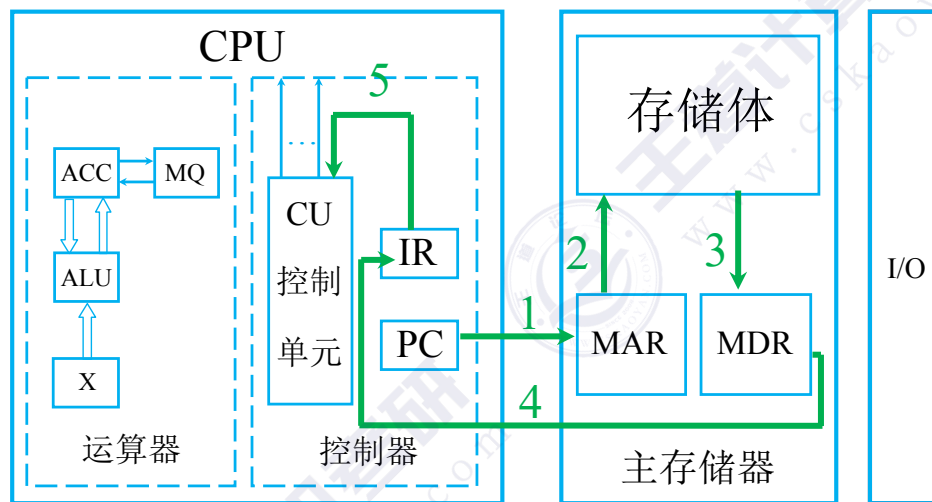
#9: $(MDR) \rightarrow$ 地址为8的存储单元, 导致 $y=7$

取指令 (#1~#4)

分析指令 (#5)

执行存数指令 (#6 ~ #9)

计算机的工作过程



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	000000000000000010		原始数据 $a=2$
6	000000000000000011		原始数据 $b=3$
7	000000000000000001		原始数据 $c=1$
8	0000000000000000111		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

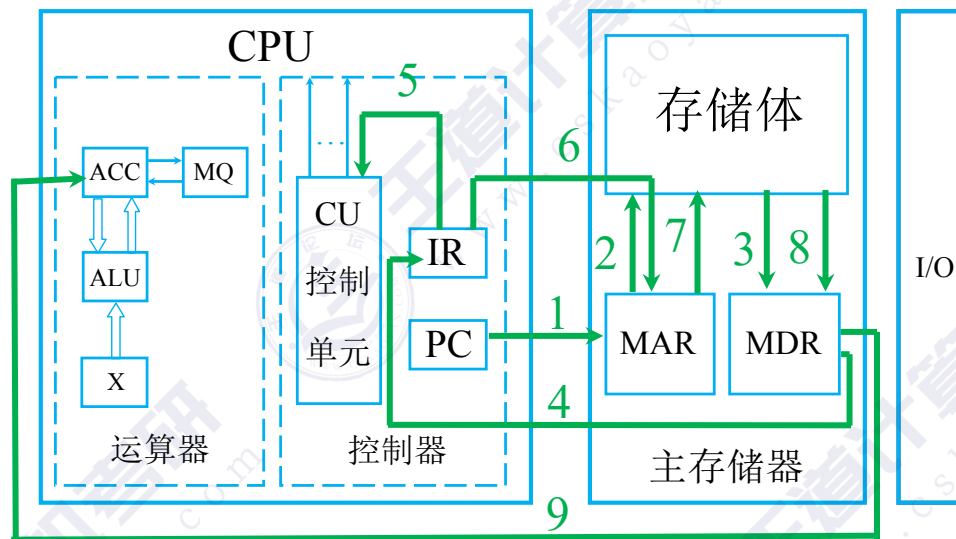
取指令 (#1~#4)

分析指令 (#5)

执行停机指令

(利用中断机制通知操作系统终止该进程)

计算机的工作过程



M: 主存中某存储单元

ACC、MQ、X、MAR、MDR...: 相应寄存器

M(MAR): 取存储单元中的数据

(ACC)...: 取相应寄存器中的数据

指令:

操作码	地址码
-----	-----

OP(IR): 取操作码

Ad(IR): 取地址码

“取数”指令的执行:
(从主存中指定地址处取数)

(PC) \rightarrow MAR

M(MAR) \rightarrow MDR

(MDR) \rightarrow IR

取指令结束 (PC)+1 \rightarrow PC

OP(IR) \rightarrow CU

分析指令结束

Ad(IR) \rightarrow MAR

M(MAR) \rightarrow MDR

(MDR) \rightarrow ACC

执行指令结束

必经步骤

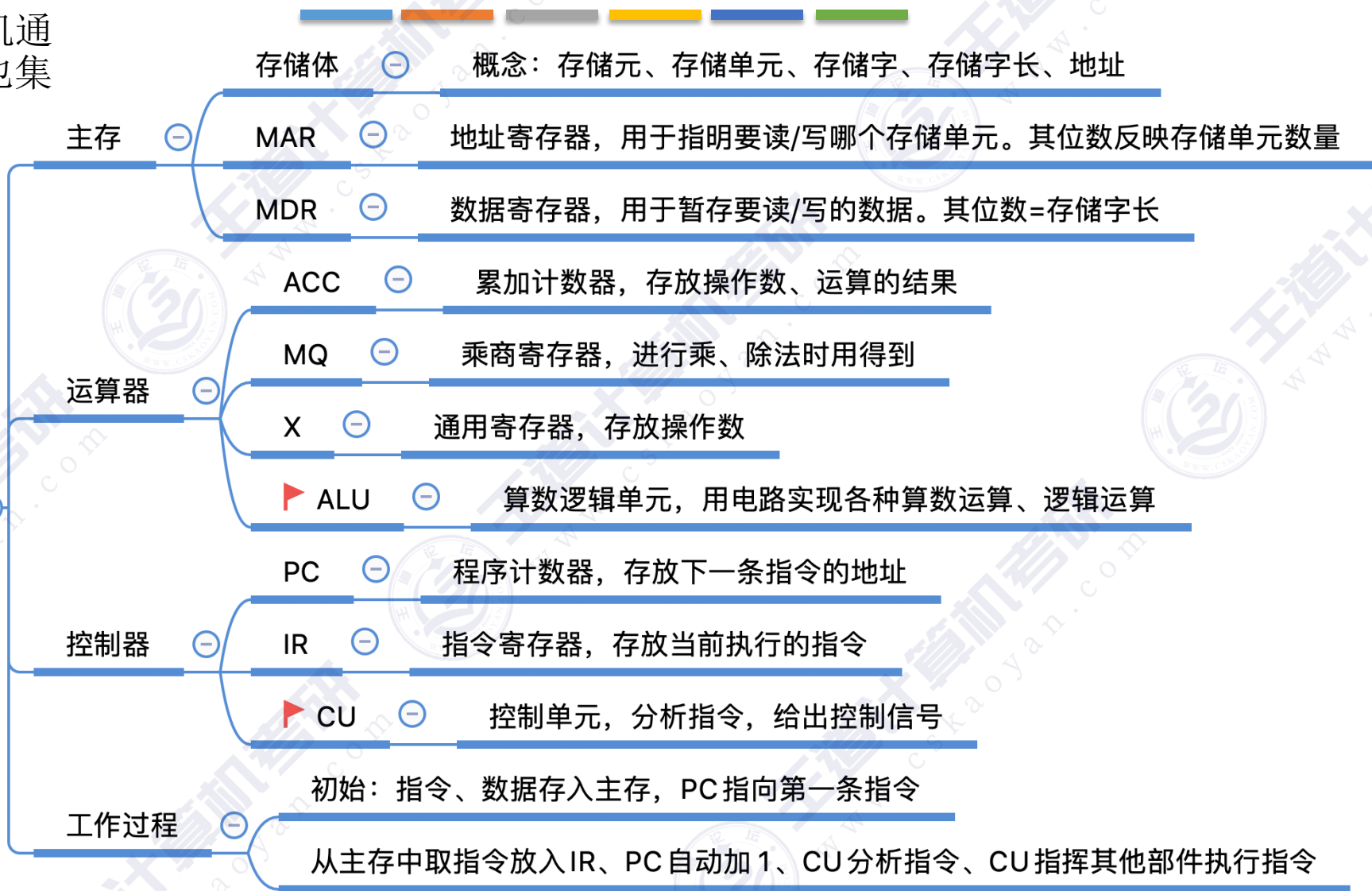
不同的指令
具体步骤不同

CPU区分指令和数据的依据: 指令周期的不同阶段

知识回顾与重要考点

注：现在的计算机通常把MAR、MDR也集成在CPU内

各硬件部件



回顾：冯诺依曼机的特点

冯·诺依曼计算机的特点：

1. 计算机由五大部件组成
2. 指令和数据以同等地位存于存储器，可按地址寻访
3. 指令和数据用二进制表示
4. 指令由操作码和地址码组成
5. 存储程序
6. 以运算器为中心（现在一般以存储器为中心）