计算思维与计算科学

计算的本质

计算的本质

从逻辑到程序

计算的本质

- 逻辑及其物理实现
- 从逻辑到运算
- 从运算到计算
- 从指令到程序

目录

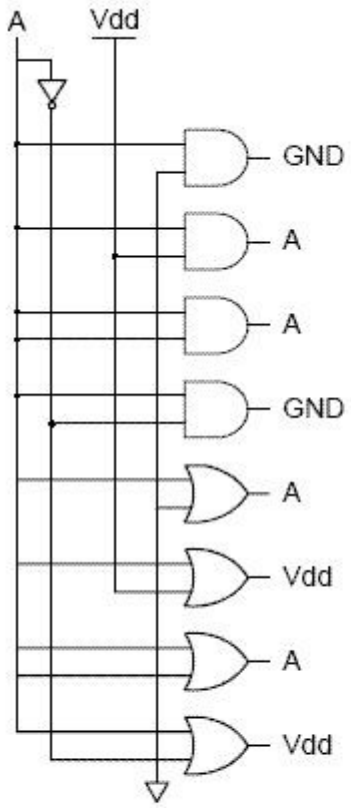
逻辑

- 逻辑学
  - 形式逻辑
  - 辩证逻辑
- 数理逻辑
  - 命题逻辑
  - 谓词逻辑
- 数字逻辑
  - 组合逻辑
  - 时序逻辑

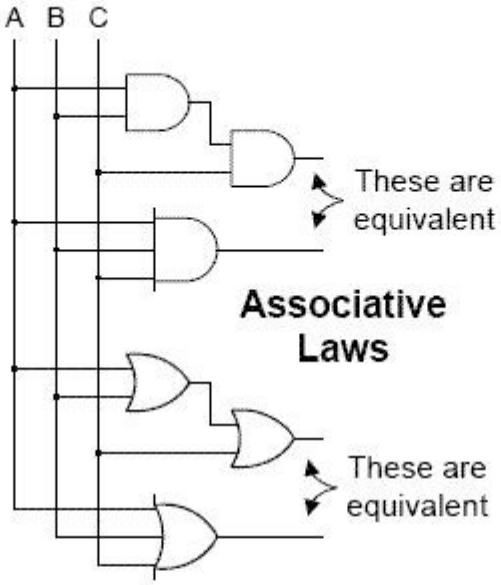
数理逻辑简介

- 布尔代数基本内容
- 运算
  - 真值表
  - 基本运算：AND OR NOT
  - 运算律
    - 交换律、结合律、分配律、德摩根定律…
- 谓词逻辑
  - 全称量词、存在量词
  - 一阶谓词逻辑

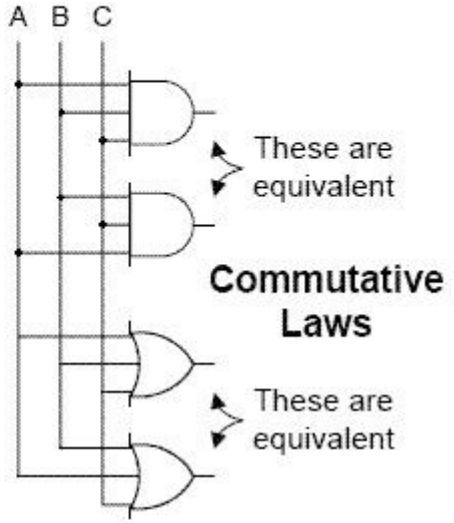
布尔代数运算定律



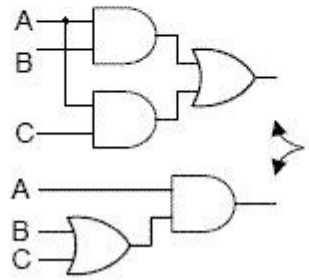
AND/OR Laws



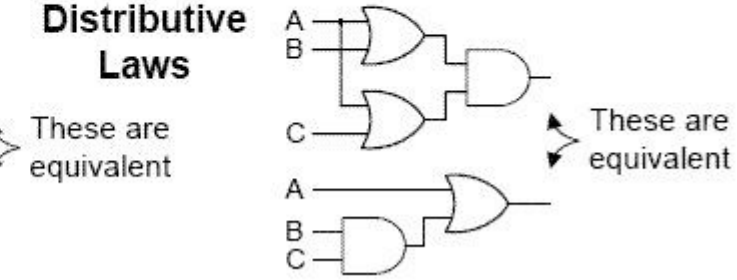
Associative Laws



Commutative Laws



Distributive Laws



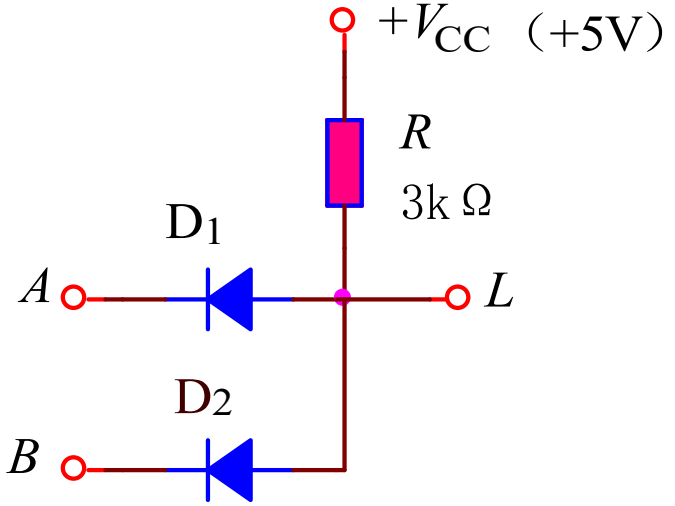
逻辑的物理实现

- 开关电路
- 继电器
- 三极管

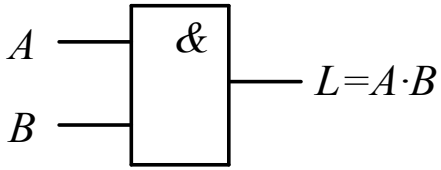
其他实现方式

- 开关实现逻辑
  - https://blog.csdn.net/booksyhay/article/details/807023l4
- 继电器门电路
  - http://www.jdzj.com/diangong/article/2018-2-3/95480-1.htm
- 杠杆实现的逻辑门
  - https://www.zhihu.com/question/323850529/answer/1035500032

二极管与门



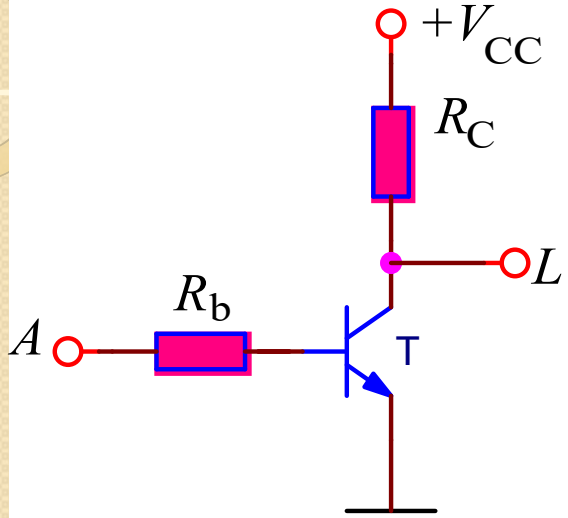
输　入		输出
V_A（V）	V_B（V）	V_L（V）
0V	0V	0V
0V	5V	0V
5V	0V	0V
5V	5V	5V

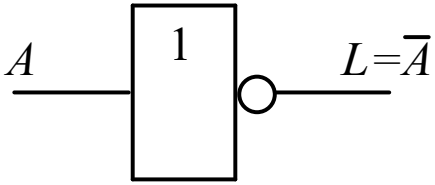


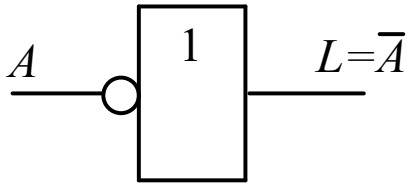
与逻辑真值表

输　入		输出
A	*B*	*L*
False	False	False
False	True	False
True	False	False
True	True	True

三极管非门电路





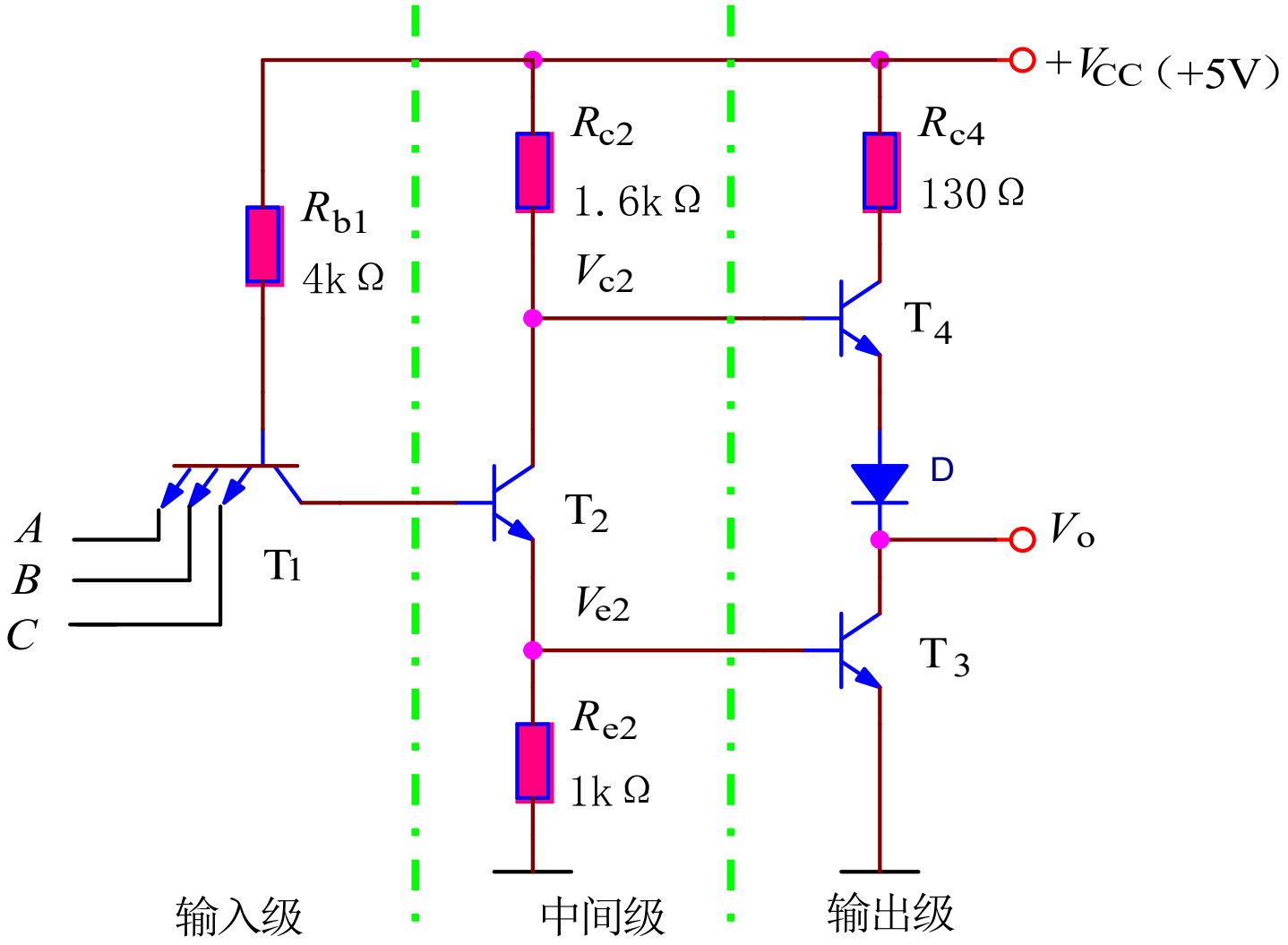


输　入	输 出
V_A（V）	V_L（V）
0V	5V
5V	0V

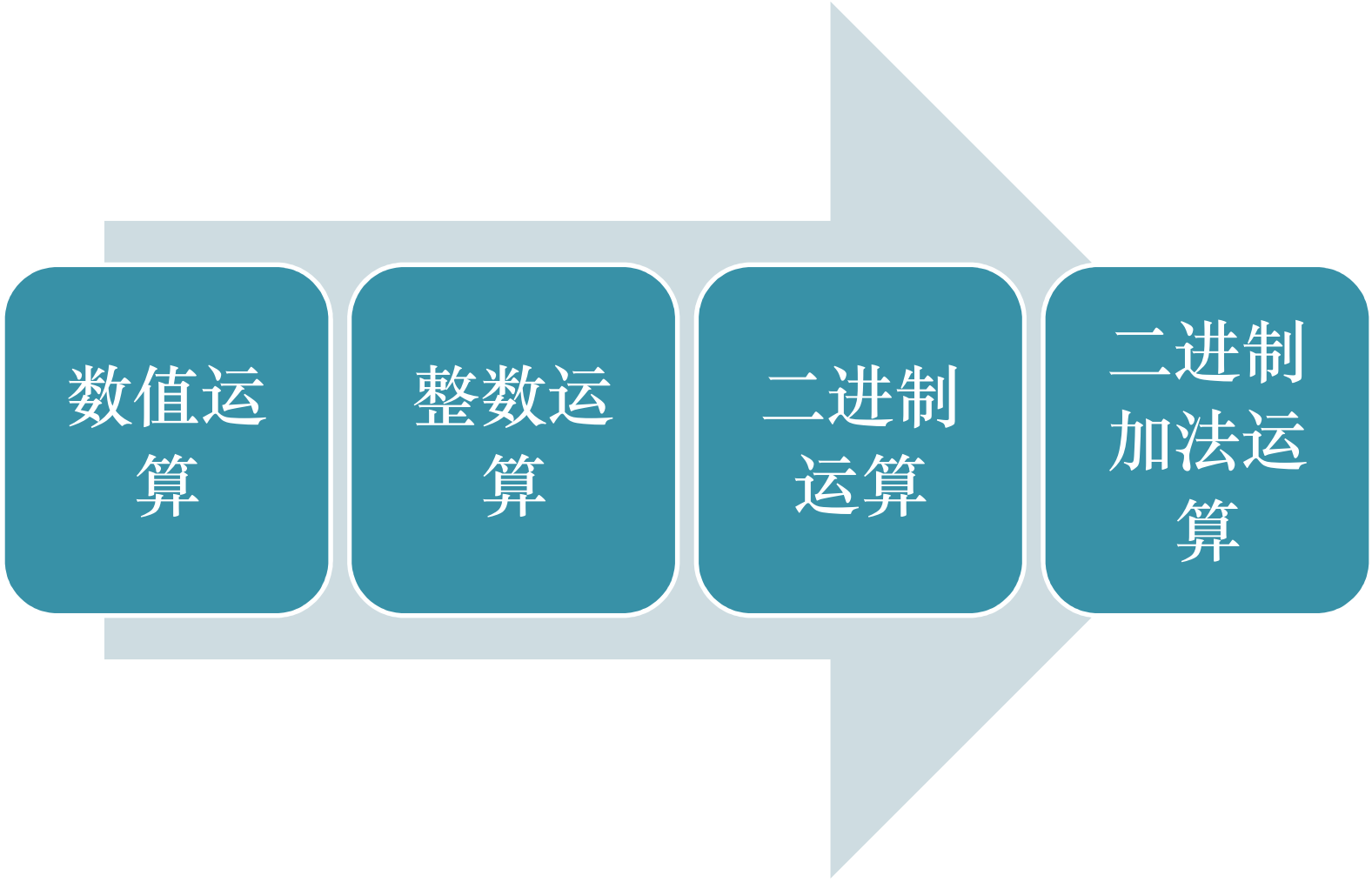
非逻辑真值表

输　入	输 出
A	*L*
False	True
True	False

TTL与非门的基本结构



运算(not 计算)的规约

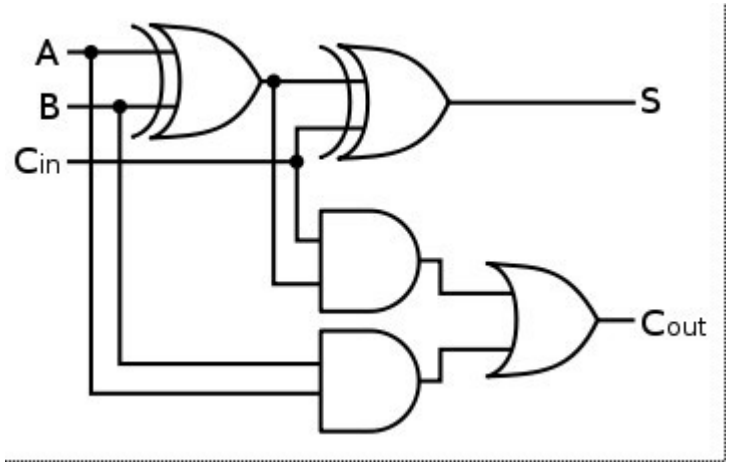


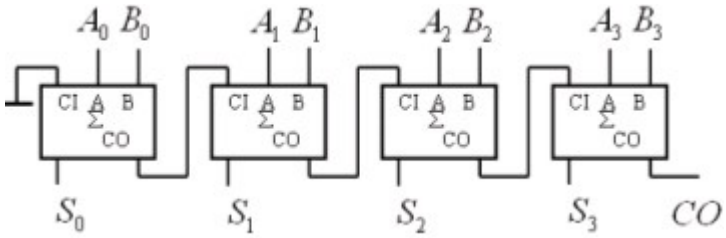
使用逻辑实现二进制加法

半加器真值表

输入		和	进位
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

S = A xor B
C = A and B





其他运算

- 补码与减法（取反）
- 乘法（移位）
- 除法（比较）
- 其他逻辑运算
  - 按位的逻辑操作
  - 移位
- 浮点运算

ALU-74181

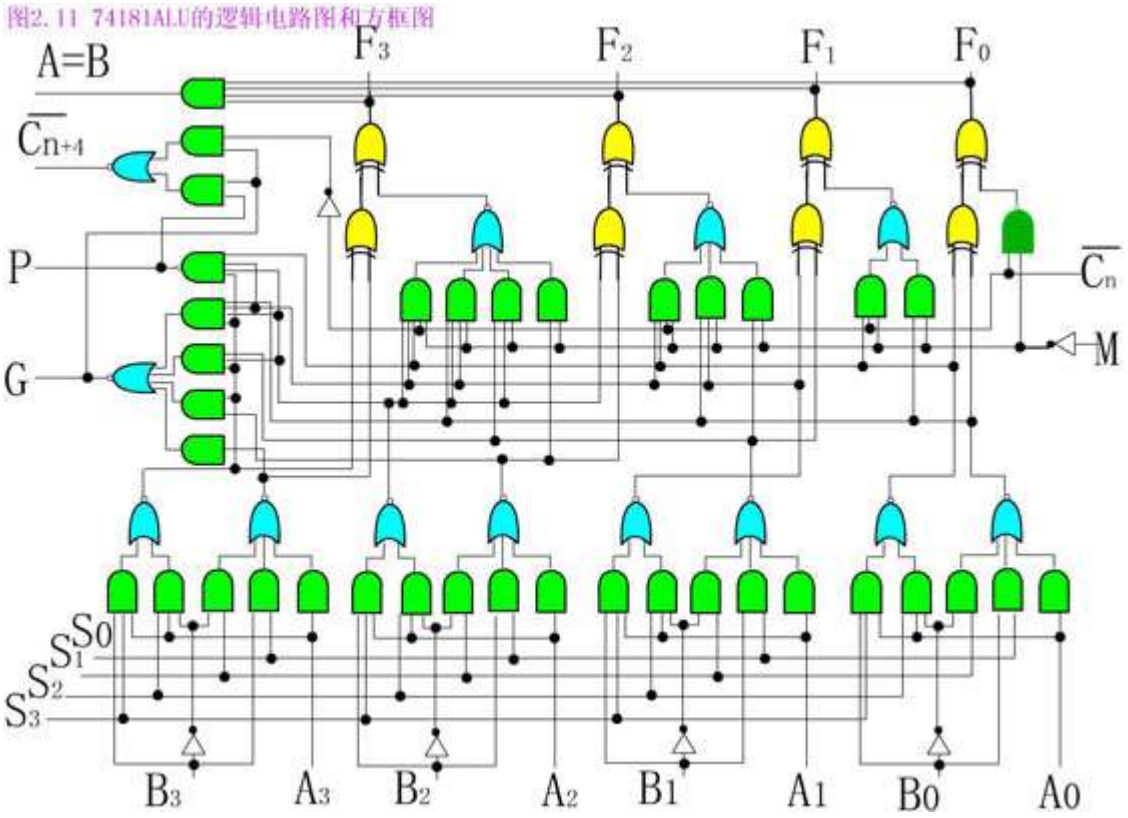


图2.11 74181ALU的逻辑电路图和方框图
A=B
F3
F2
F1
F0
Cn+4
P
G
Cn
M
S0
S1
S2
S3
B3
A3
B2
A2
B1
A1
B0
A0


5F27
HD74LS181P

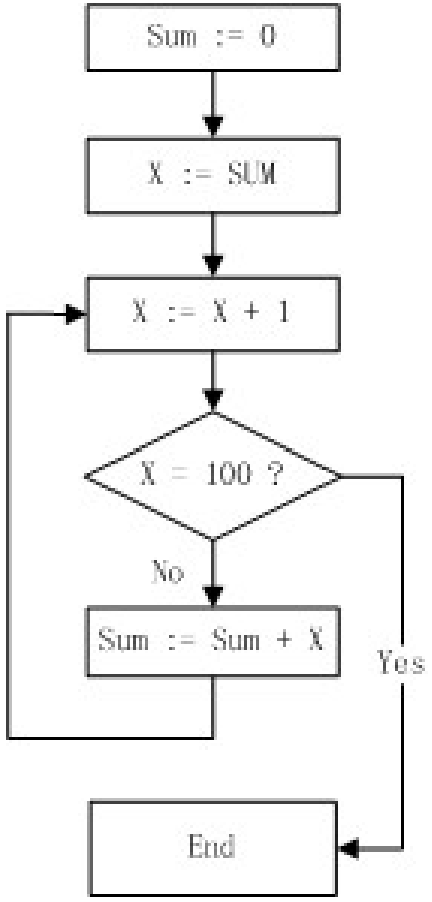
74181算术/逻辑运算功能表

工作方式选择输出 $S_3\ S_2\ S_1\ S_0$	负逻辑输入与输出		正逻辑输入与输出	
	逻辑 (M=H)	算术运算 (M=L)(C_n=L)	逻辑 (M=H)	算术运算 (M=L)(C_n=H)
L L L L	$\overline{A}$	A减1	$\overline{A}$	A
L L L H	$\overline{AB}$	AB减1	$\overline{A+B}$	A+B
L L H L	$\overline{A}+B$	$\overline{AB}$减1	$\overline{A}$ B	A+$\overline{B}$
L L H H	逻辑1	减1	逻辑0	减1
L H L L	$\overline{A+B}$	A加(A+$\overline{B}$)	$\overline{AB}$	A加A$\overline{B}$
L H L H	$\overline{B}$	AB加(A+$\overline{B}$)	$\overline{B}$	(A+$\overline{B}$)加A$\overline{B}$
L H H L	$\overline{A\oplus B}$	A减B减1	A⊕B	A减B减1
L H H H	A+$\overline{B}$	A+$\overline{B}$	A$\overline{B}$	$\overline{AB}$减1
H L L L	$\overline{A}$B	A加(A+B)	$\overline{A}$+B	A加AB
H L L H	A⊕B	A加B	$\overline{A\oplus B}$	A加B
H L H L	B	A$\overline{B}$加(A+B)	B	(A+$\overline{B}$)加AB
H L H H	A+B	A+B	AB	AB减1
H H L L	逻辑0	A加A*	逻辑1	A加A*
H H L H	A$\overline{B}$	AB加A	A+$\overline{B}$	(A+B)加A
H H H L	AB	$\overline{AB}$加A	A+B	(A+$\overline{B}$)加A
H H H H	A	A	A	A减1

从运算到计算

- 计算是按照一定规则对信息进行有限次变换的过程
  - 提供多种运算操作（指令）
  - 要能存储中间结果（含输入输出）
  - 要把多个运算按步骤排列起来
  - 要能重复计算
  - 能够根据情况按照不同分支计算

例：从1加到100

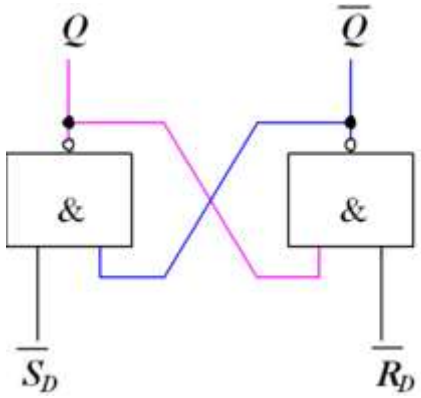


- 运算指令
  - 清零 CLR
  - 加一 INC
  - 加法 ADD
- 跳转指令
  - 无条件跳转 JMP
  - 有条件跳转
    - 比较相等跳转指令　CJE
- 内存访问指令和数据传输指令
  - MOV

需要实现的能力

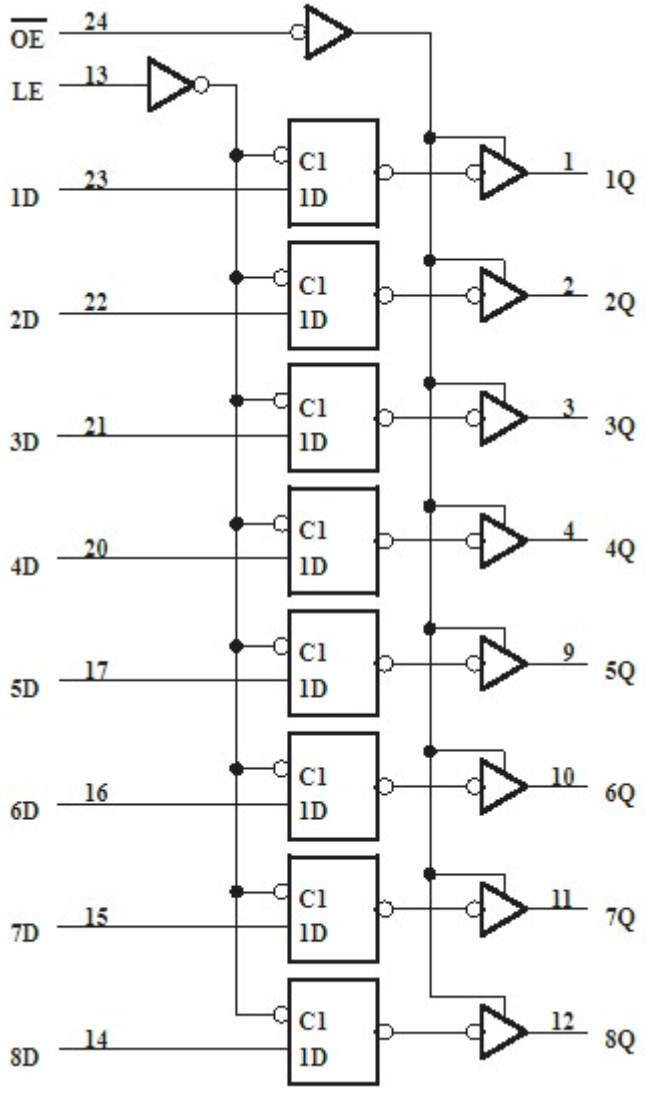
- 一、实现数据存储
  - 使用存储单元暂存数据
  - 为了区分，需要给每个存储单元以访问的标识，也就是地址
- 二、实现每个功能框的运算
  - 受控的ALU
  - 每个功能编码，控制ALU及其他部件
- 三、实现连续运行
  - 控制流（顺序执行和GOTO）

存储的物理实现-静态存储器

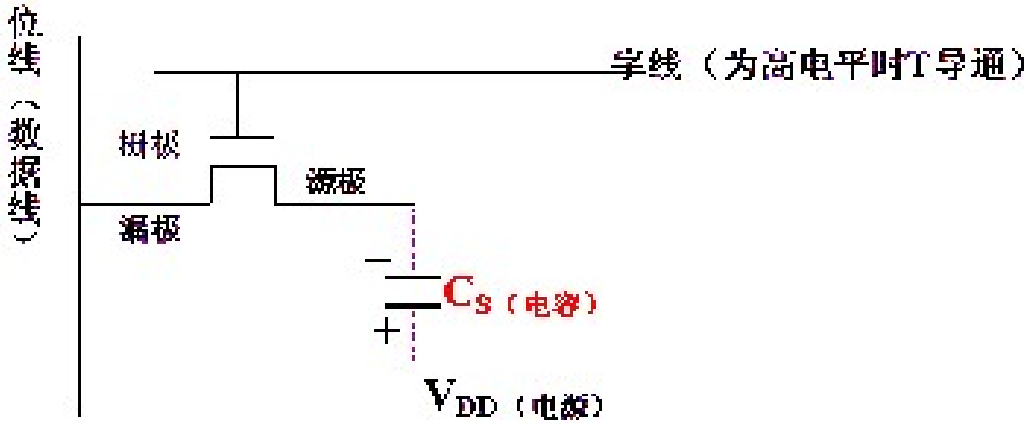


RS触发器

8位锁存器

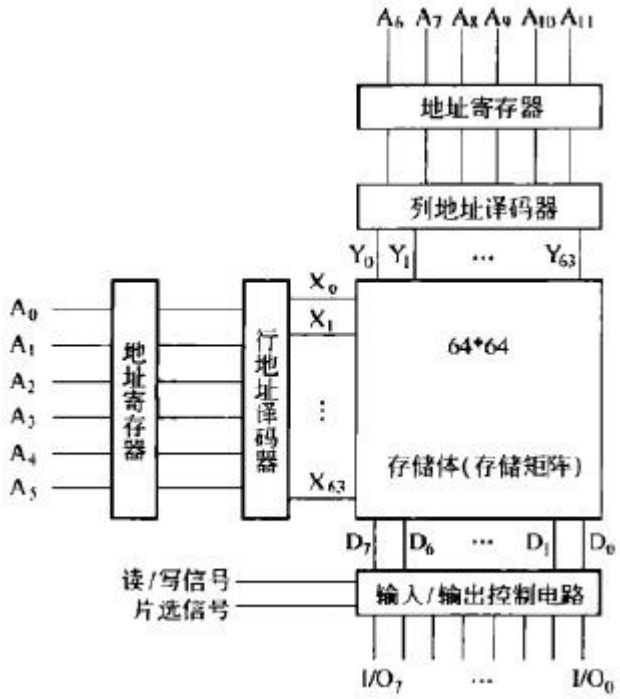


存储的物理实现-动态存储器



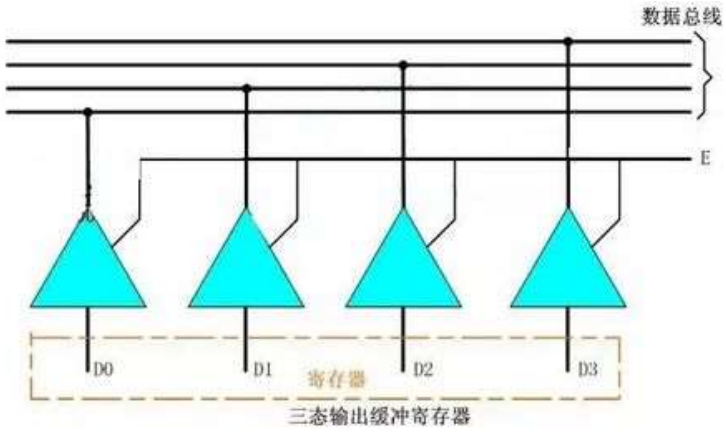
MOS单管存储单元线路

内存译码与读写


A6 A7 A8 A9 A10 A11
地址寄存器
列地址译码器
Y0 Y1 … Y63
A0
A1
A2
A3
A4
A5
地址寄存器
行地址译码器
X0
X1
X63
64*64
存储体(存储矩阵)
D7 D6 … D1 D0
读/写信号
片选信号
输入/输出控制电路
I/O7 … I/O0

总线

- 使用一条线路将各个相关部件连接在一起
- 通过控制信号控制每个部件与总线之间的连通性
- 一个时刻，只能有一个数据提供者，一般只有一个使用者
- 包括数据总线、地址总线和控制总线



功能->指令集

- 功能分类
  - 运算
    - 运算数和结果的存放位置
  - 数据传送
    - 源->目的
    - 内存单元编码
  - 执行次序
    - 顺序执行
    - 无条件转移
    - 有条件转移

功能->指令集

初始版本（V0）	增加累加器（V1）	加入标志位(C,Z)（V2）	其他指令
CLR	MOV A, addr	MOV A, #direct	NOP
INC	MOV addr，A	MOV A, addr	HALT
ADD	CLR A	MOV addr，A	
JMP	INC A	INC A	
CJE	DEC A	DEC A	
MOV	ADD A, addr	ADDC A, addr	
	SUB A, addr	SUBB A, addr	
	JMP addr	CLR A(and C)	
	CJE A, addr	CMP A, addr	
	CJG A, addr	JZ/JNZ addr	
		JC/JNC addr	
		JMP addr	

指令集编码（V2）

操作码	操作数长	指令	操作码	操作数长	指令
0000	无	NOP	1000	12	JZ addr
0001	8	MOV A, #direct	1001	12	JNZ addr
0010	12	MOV addr, A	1010	12	JC addr
0011	12	MOV A, addr	1011	12	JNC addr
0100 0000	无	INC A	1100		空白
0100 0001	无	DEC A			
0100 0010	无	CLR A			
0101	12	CMP A, addr	1101		空白
0110	12	ADDC A, addr	1110	12	JMP addr
0111	12	SUBB A, addr	1111	无	HALT

设计一个计算设备

- 实现相关运算指令
  - 将指令编码成数
  - 指令中的编码可以通过逻辑来控制ALU和其他设备的运算方式，以及各个数据通路的开启
- 实现跳转
  - 每一条指令有一个标识，如果存放在内存中，就是指令的地址
  - 编码后的指令放在存储器中
- 内存的使用
  - 存储数据（运算结果和中间变量）
  - 存储指令
  - 内存编址

指令的完整实现

- ALU长度（存储单元字节）：8
- 指令长度：16位
- 操作码长度：4位
- 操作数：12位（4K）
  - 部分指令操作数为空
    - NOP、DEC、INC、CLR、HALT
  - 操作数是地址的
    - MOV、ADDC、SUBB
    - 转移指令
  - 操作数是数据的
    - MOV A, #direct（8位） （空4位）

使用机器指令（汇编语言）实现

- 定义变量的存储地址
- 编写代码
- 跳转目标地址暂用标号表示

```
;0400H: Sum
;0410H: X
;0420H: 100,常量
ORG  0000H
       CLR A
       MOV [0400H], A   ; Sum=0
       MOV [0410H], A   ; X=0
       MOV A, #100
       MOV [0420H], A
LOOP1:
       MOV A, [0410H]
       INC A          ; X=X+1
       CMP A, [0420H]
       JZ STOP
       MOV [0410H], A ;Save X
       ADD A, [0400H] ;X + Sum
       MOV [0400H], A ;Save Sum
       JMP LOOP1
STOP:
       HALT
```

汇编（汇编语言->机器码）

- 定义程序起始地址
- 将每条指令翻译为代码
  - 转移指令目标地址暂空
- 推算每个标号的地址
- 将标号地址替换到转移指令里面

实现一个CPU

- 使用同步时序电路
  - 确定基准时钟（主频）
- 每个指令操作时序
  - 读取指令译码、读取操作数、运算、写回结果
- 使用总线连接各个部件，使用三态门等控制各个部件与总线的连接
- 内部设立程序计数器PC，保存当前指令地址
- 转移指令的结果就是写入程序计数器

基本的CPU结构

- 程序计数器PC
- 指令译码器
- 累加器
- 三总线
  - 数据总线
  - 地址总线
  - 控制总线

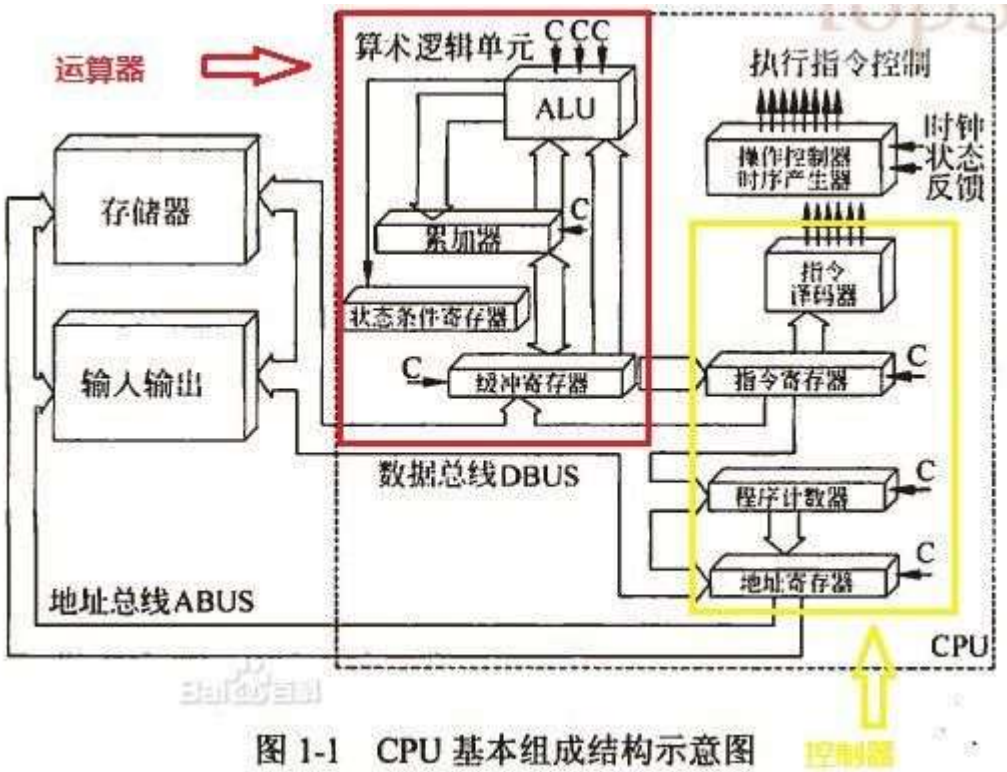


图 1-1　CPU 基本组成结构示意图

扩展CPU的能力

- 增加基本运算
  - 逻辑运算、移位
- 增加更多的CPU内部寄存器
- 程序状态字PSW
  - 保存上次计算的一些标志，或者CPU的某些状态，供后续指令使用
  - 进位标志CY、零标志Z、溢出标志OV
- 子程序调用和中断
- 寻址方式
  - 访问存储的地址不一定是常数，而是通过某种方式计算得到的

扩展指令系统

- 增加基本运算
  - 逻辑运算、移位
- 利用PSW进行判断和参与运算
- 子程序调用和堆栈指示器
  - CALL、RET
  - 递归
- 多种寻址方式
  - 内部寄存器寻址
  - 变址寻址（数组）、基址寻址（栈变量）

指令集编码（V3）

操作码	操作数长	指令	操作码	操作数长	指令
0000	0/12	NOP	1000	12/12	JZ addr
0001	8/12	MOV A, #direct	1001	12/12	JNZ addr
0010	12/12	MOV addr, A	1010	12/12	JC addr
0011	12/12	MOV A, addr	1011	12/12	JNC addr
0100 0000	0/8	INC A	1100	12/12	LOAD DPTR, addr
0100 0001	0/8	DEC A			
0100 0010	0/8	CLR AC			
0100 1000	0/8	INC DPTR	1101 0000	0/8	MOV [DPTR], A
0100 1001	0/8	DEC DPTR	1101 0001	0/8	MOV A, [A+DPTR]
0101	12/12	CMP A, addr			
0110	12/12	ADDC A, addr	1110	12/12	JMP addr
0111	12/12	SUBB A, addr	1111	0/12	HALT

提高计算速度的几种途径

- 指令流水
- Cache
  - 通过Cache实现在不同计算速度的单元之间的匹配
  - 基本原理：访问局部性
  - 替换算法
- 专用处理单元（FPU、GPU...）
- 并行处理
  - 多核
  - 多CPU
  - 多机

一个相对复杂的CPU结构

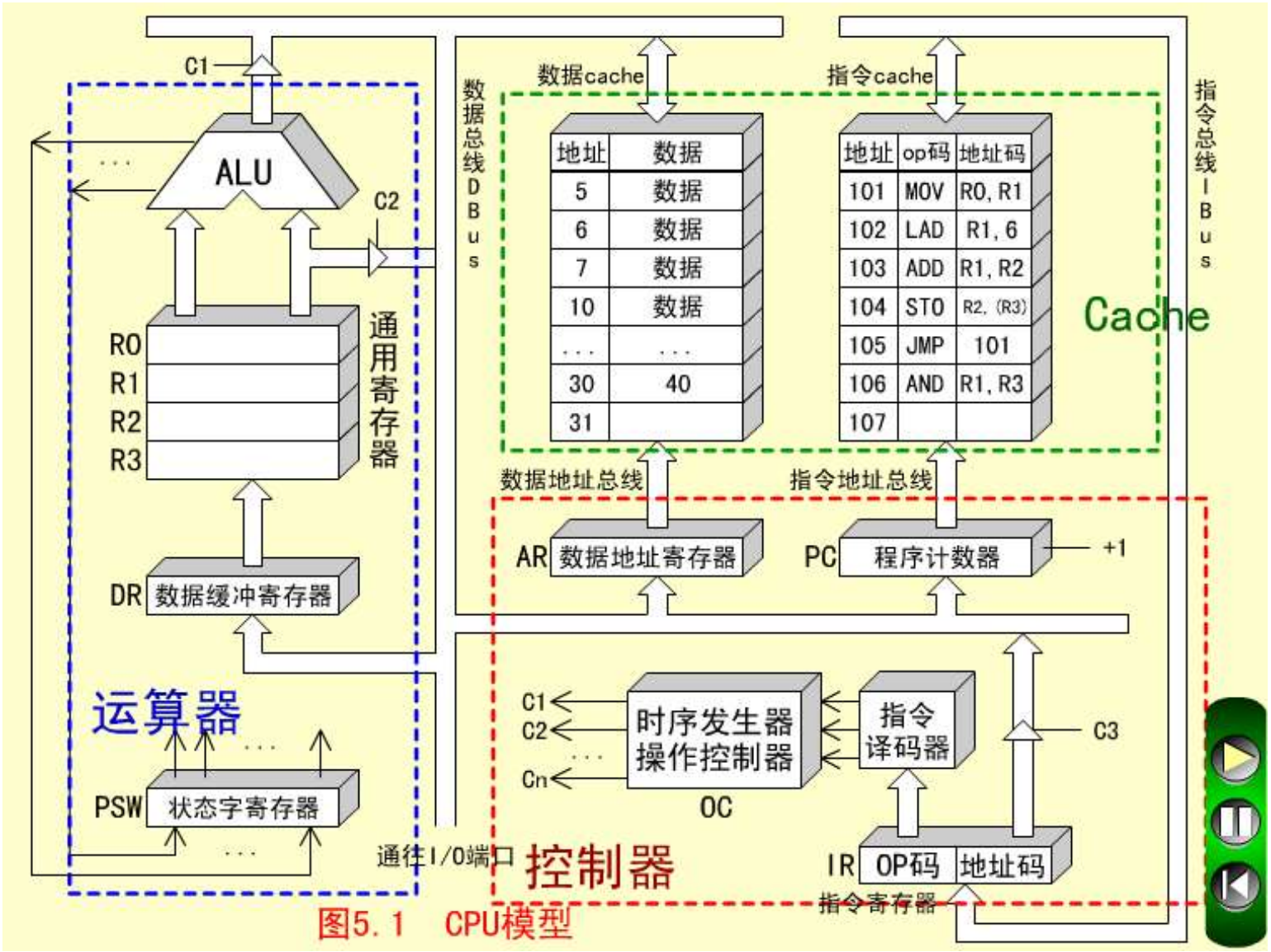


图5.1 CPU模型

从指令到程序

- 程序：完成一定功能的指令序列
- 能够完成程序要求的基本指令类型
  - 数据传送指令
  - 算术逻辑运算指令
  - 跳转，条件跳转指令
- 能够提供不同的寻址方式
  - 直接寻址
  - 间接寻址
- 能够支持程序循环和分支执行

程序存在哪里？

程序存储体系结构

冯·诺依曼体系结构

- 数据以二进制表示
- 结构上包括：运算器、控制器、存储器、输入设备和输出设备
- 存储器是线性编址的，具有唯一地址
- 指令由操作码和操作数（地址）构成
- 程序存储方式
  - 指令和数据都存放在存储器中
  - 指令构成的程序是可修改的
- 指令按照指令计数器给出的地址依次执行，但可以根据指令的含义（或外部条件）改变指令计数器的内容，从而实现跳转

冯·诺依曼体系结构的思考

- 计算是串行的
- 计算结果是确定的
- 计算过程是不确定性的减少

- 冯·诺依曼体系结构的瓶颈?

课后作业

- 一、仅使用if和goto作为控制流语句，使用C语言完成一个计算100以内素数的程序。
- 二、使用示例指令系统实现程序功能如下：统计0600H单元的字节的2进制表示中有多少个1，结果写入0610H单元，并将该程序翻译成机器码。
- 三、讨论使用示例指令系统，如何实现移位（左移、右移）、按位逻辑运算（与或非）
- 四、设计一个你认为最简化的指令系统，并说明如何实现示例指令系统的每条指令功能（NOP和HALT除外）