船用微型计算机原理与接口技术
第一章知识整理

前言

个人整理总结，仅作为复习参考使用，并不权威，如有错误纰漏，欢迎指正，请联系 y0r4h21@whut.edu.cn 或 2124436512@qq.com。此文档完全免费。

第1章 微型计算机基础概论

（本章内容大致了解即可）

第一节 计算机系统

计算机系统分为硬件系统以及软件系统。硬件系统包括主机设备和外部设备，其中，主机设备包含 CPU、存储器、输入输出接口、总线。外部设备包括所有可以通过输入输出接口与计算机进行信息交换的电子设备。

微处理器

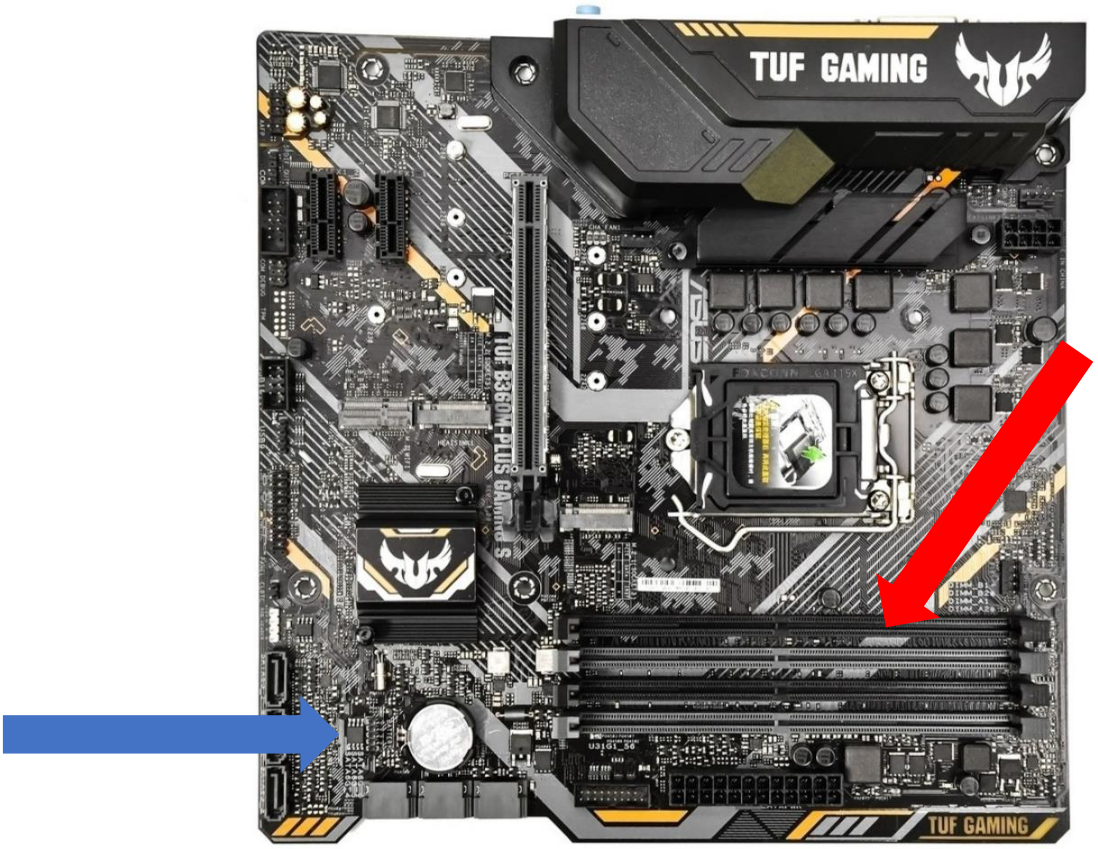
简称 CPU，包括运算器、控制器、寄存器组。

存储器

包括内存储器和外存储器。

外存储器分为联机内存和脱机内存，联机内存即为各种硬磁盘。脱机外存即为各种移动存储设备。

内存储器：内存按单元组织，每单元对应唯一地址。每个内存单元中存放 1 字节数据（1 字节即为 1Btye，每 8 个二进制位称为 1 字节）。内存单元个数称为内存容量。按工作方式将内存分为 RAM（随机存取存储器）和 ROM（只读存储器）。如图所示的主板，可以粗略的说，红色箭头所指的是内存条插槽，用来插 RAM 的；蓝色箭头所指的是 ROM。



输入输出接口

接口是 CPU 与外部设备的桥梁。

主要功能：数据缓冲寄存，信号电平或类型的转换，实现主机与外设之间的运行匹配。（了解即可）

总线

一组导线和相关的控制、驱动电路的集合。

是计算机系统各部件之间传输地址、数据、控制信息的通道。

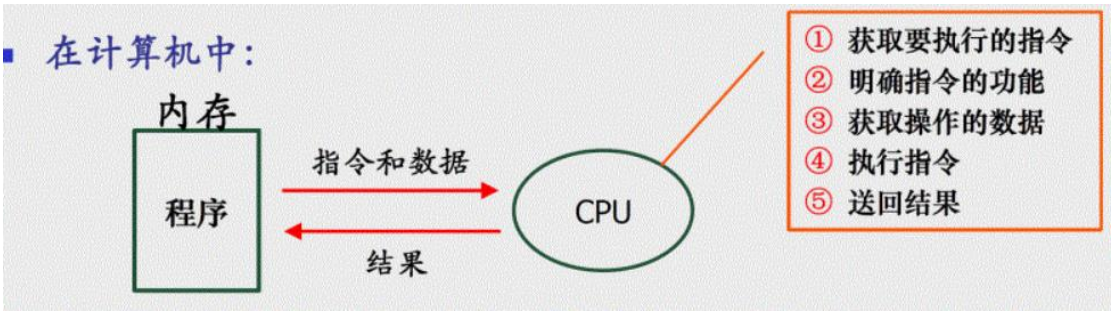
分为 AB（地址总线），DB（数据总线）,CB（控制总线）

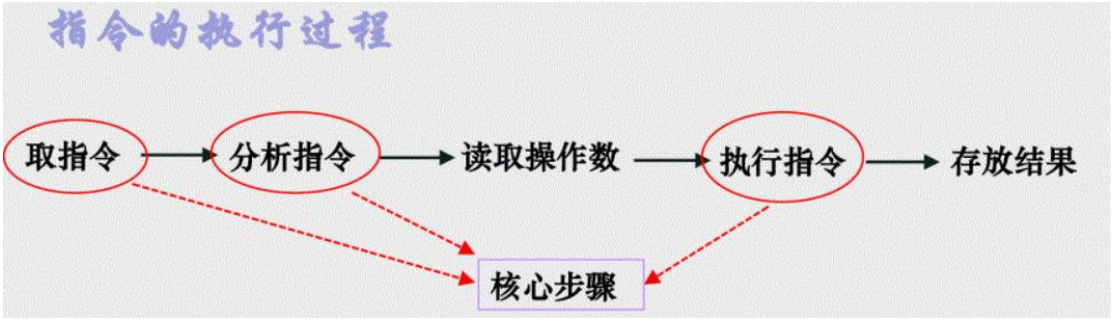
软件系统：略

第二节 计算机中指令执行过程

指令：由人向计算机发出，能够被计算机识别的命令。

程序是指令的序列，计算机的工作过程就是执行指令的过程。





指令分顺序执行和并行执行。

顺序执行：一条指令执行完再执行下一条。（由第二章知识会知道这里其实是指各功能部件交替工作，按顺序完成指令执行）

并行执行：同时执行多条指令。（各功能部件并行工作）

顺序执行时，执行时间=取指令时间+分析指令时间+执行指令时间=3t

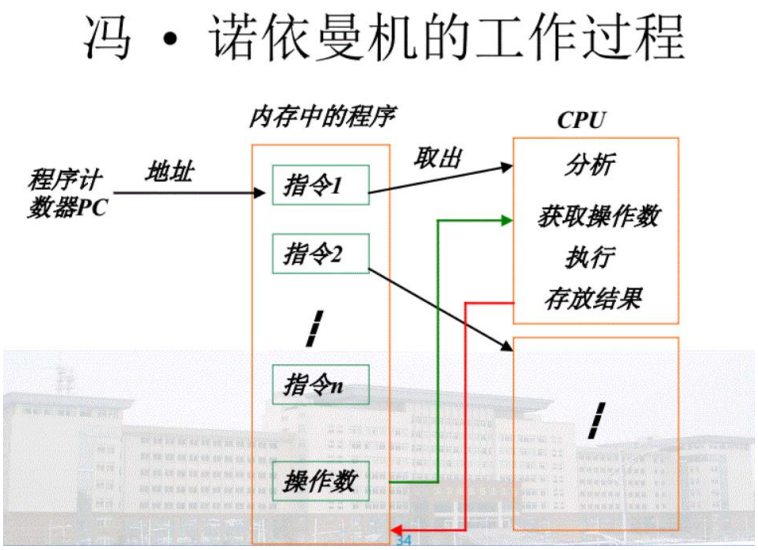
并行执行时，仅第一条指令需要 3t，之后每经过 t 就有一条指令执行结束。

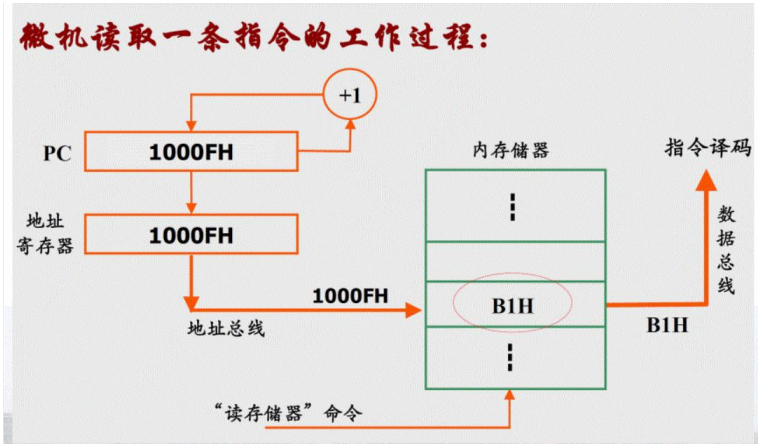
第三节 计算机结构

主要有冯诺依曼结构和哈佛结构。

冯诺依曼计算机工作原理：存储程序（或者叫程序存储）工作原理。

冯诺依曼计算机结构特点：以运算器为核心。



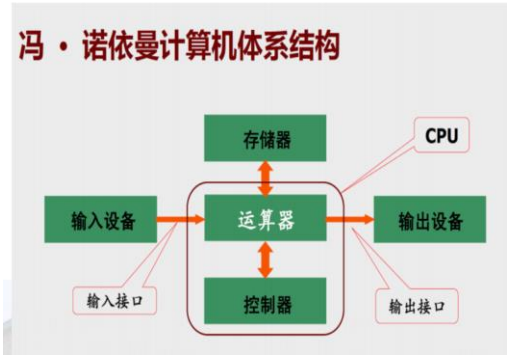


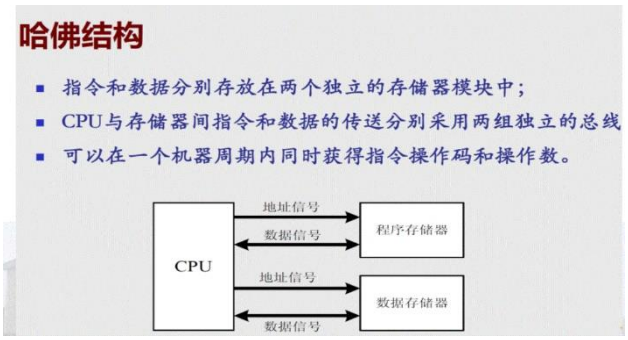
上图看不懂没关系，只需要知道这几点

1：内存中的程序以指令 1，指令 2……指令 n，操作数组成

2：存在程序计数器 PC，起初指向第一条指令的地址，随后每次自加 1，来实现连续读取下一条指令。

3：执行指令过程中，CPU 取出指令→获取操作数→执行指令→存放结果





第四节 数制及编码

常用计数制

常用十进制 D，二进制 B，十六进制 H，八进制

对任一 K 进制数 S，均可以用有权展开式来表示，如下图

$$
\begin{aligned}
(S)_k &= S_{n-1} \times K^{n-1} + S_{n-2} \times K^{n-2} + \cdots + S_0 \times K^0 + S_{-1} \times K^{-1} \\
&\quad + \cdots + S_{-m} \times K^{-m} \\
&= \sum_{i=-m}^{n-1} S_i \times K^i
\end{aligned}
$$

数制转换

非十进制转十进制：按有权展开式展开再按十进制运算规律求和即可

十进制转二进制：整数部分除 2 取余直到小于 2，再反向排列；小数部分乘 2 取整直到规定标准，再正向排列（这也是为什么十转二会丢精度，所以小数部分运算时达到所需精度或者为 0 即可停止）

十转十六：整除 16 取余，小数乘 16 取整。或者先十转二再转十六

二转十六：小数点开始向左右 4 位分一组，不足则在左/右补 0，再转为十六进制

十六转二：1 位换 4 位，多余 0 去掉

二—八互转和二—十六互转差不多，因为 2^3=8；2^4=16

这几个规则可能有点抽象，我们用例子说明一下

$$1011.11B=1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$$

$$=8+2+1+0.5+0.25$$

$$=11.75$$

- 25.5 =11001.1B =00011001.1000B =19.8H
- 11001010.0110101 0B =CA.6AH

  C A 6 A

253.25 转二进制：先看整数部分，253/2=126……1，126/2=63……0，63/2=31……1，31/2=15……1，15/2=7……1，7/2=3……1，3/2=1……1，所以整数部分为 1111101；小数部分，0.25*2=0.5 取 0，0.5*2=1 取 1，所以小数部分为 01。综上，253.25=1111101.01B

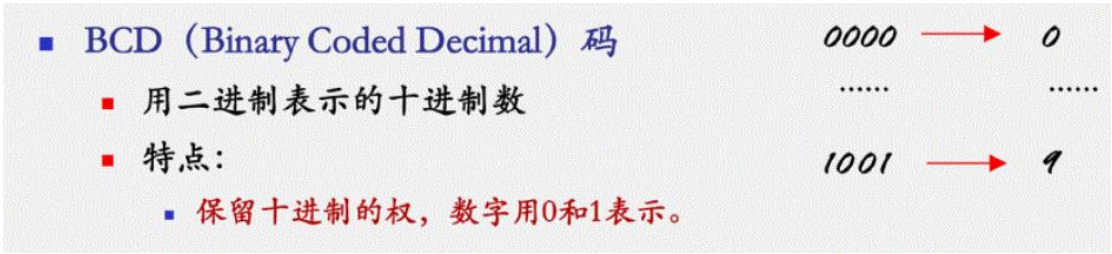
计算机中二进制数表示：略，这个比较复杂，可以自己看书

二进制编码

二进制编码的十进制数

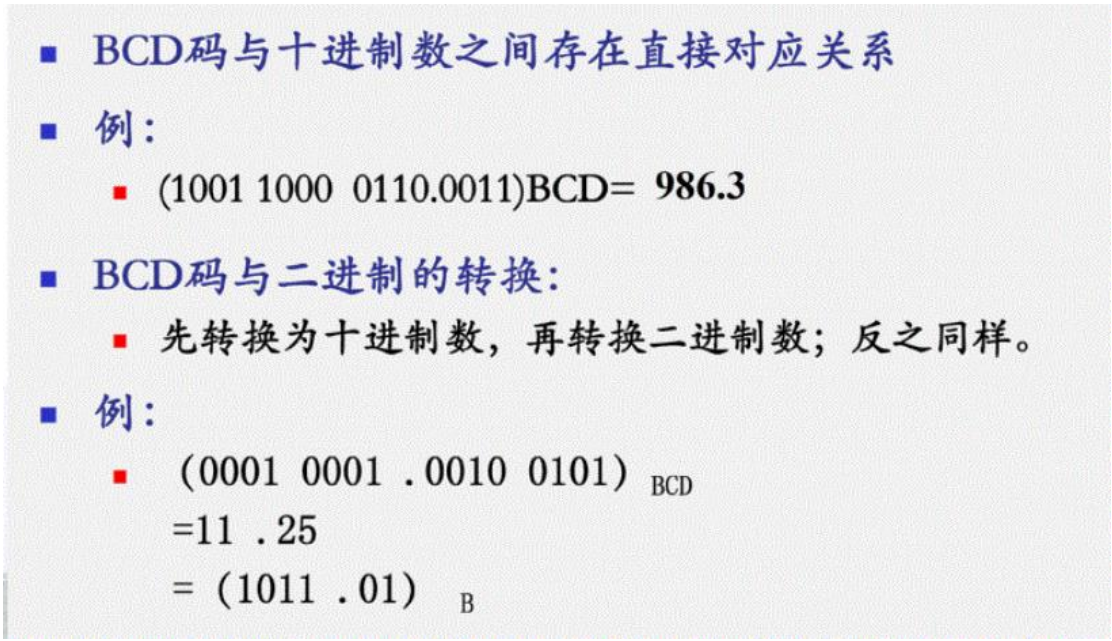
用二进制编码表示的十进制数，成为二—十进制码，即 BCD 码。

BCD 码最常用的是 8421BCD 码，用四个二进制数表示十进制数，以 BCD 作为结尾标记符，比如（00111001）$_{BCD}$（注，此代表 39）。



BCD 码只有 0001~1001 是合法的

BCD 码与十进制码之间转换只需要按位转换即可，BCD 码与二进制之间转换需要借助十进制。



BCD 码在计算机中以压缩 BCD 码和扩展 BCD 码的形式存放

压缩 BCD 码即 4 个二进制位代表 1 位 BCD 码，一个字节存放 2 位 BCD 数

扩展 BCD 码即 8 个二进制位代表 1 位 BCD 码，其中高四位为 0，低四位为有效位，一个字节存放 1 位 BCD 数

字符的编码

西文字符采用 ASCII 码，用 7 位二进制码表示 128 个字符与符号（但是有 8 个二进制位），字节最高位 D_7 位默认为 0（不代表它一直是 0，奇偶校验时便会体现）（D_7 不是第七位啊，别读着读着搞混了）

ASCII 码在传输的时候常常采用奇偶校验法。

奇偶校验：偶校验代表含校验位在内的 8 位二进制码中“1”的个数为偶数。奇校验反之。

奇偶校验和最高位是 1 还是 0 相关，根据传送的字符来定，举个例子：A 的 ASCII 码为 1000001B（写全的话其实是 01000001B），如果我们将 A 以奇校验发送，则需要将“1”的个数变成奇数，所以将最高位变成 1，发送的其实是 11000001B。如果我们将 A 以偶校验发送，观察到“1”的个数已经是偶数了，所以最高位不用动，直接发送 1000001B（最高位 0 忽略掉了）

第五节 无符号二进制数运算

算术运算

加减法运算略，乘除法运算规则其实和十进制是一模一样的。与此同时，乘/除法还可以用移位加减的思想来做（其实十进制中也是这样）

K 进制数移位加减结论：左移一位等效于乘 K，右移一位等效于除 K。左移就是在数最右边加个 0，右移最右边就是扣一个数

例如 1100B*1001B，实际上就是 1100B*（2^3+2^0）=1100000B+1100B=1101100B

注 1：这个放十进制里一看就懂，就是 256*1000=256000 的道理

注 2：实在不理解乘法就换成十进制再算吧

无符号数表示范围

$$0 \leqslant X \leqslant 2^n-1$$

若运算结果超出这个范围，则产生溢出。

对无符号数：运算时，当最高位向更高位有进位（或借位）时则产生溢出。

乘运算不会产生溢出，除运算时若除数过小就会溢出

逻辑运算

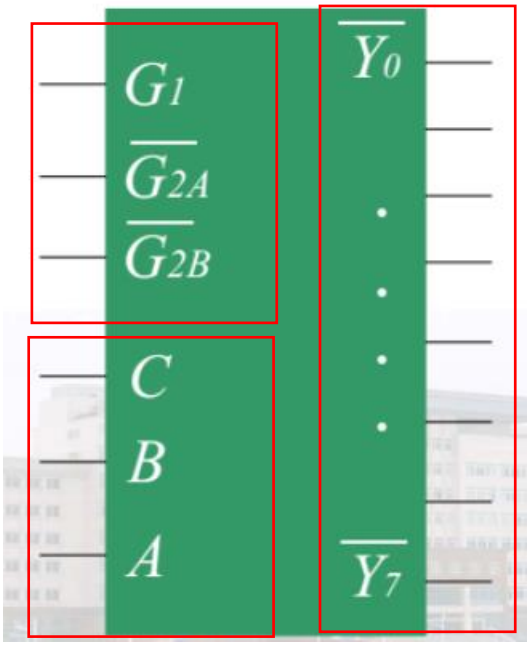
1：与运算：全 1 则 1。
2：或运算：有 1 则 1。
3：非运算：1 为 0，0 为 1。
4：异或运算：相同为 0 不同为 1。
注：这里描述符号比较难打，就自己看书吧。（抱歉）
1,2,4 运算是两数之间的运算，3 是一个数按位取反。

逻辑门：略，自行学习

译码器

作用：将一组输入信号转换为某一时刻的一个确定的输出信号

一种常用的 3—8 线译码器为 74LS138 译码器，如图



左上框为使能端，左下框为译码输出端，右框为译码输出端

使能端的三个引脚共同决定译码器是否被允许工作，若 G1=1，G2A=G2B=0，则工作，处于使能（enable）状态，否则均处于禁止（disable）状态。

CBA 输入线可以代表 8 种不同的输入状态

Y0~Y7 根据 CBA 状态输出内容

具体输入输出关系如图所示

值表）如表 1-8 所示，表中电平为正逻辑，即高电平表示逻辑 1，低电平表示逻辑 0，×表示不定，＃表示该信号低电平有效（与上横线标注¯含义相同）。

表 1-8　74LS138 功能表

使　能　端			输入端			输　出　端							
G_1	$\#G_{2A}$	$\#G_{2B}$	C	B	A	$\#Y_0$	$\#Y_1$	$\#Y_2$	$\#Y_3$	$\#Y_4$	$\#Y_5$	$\#Y_6$	$\#Y_7$
×	1	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	×	1	1	1	1	1	1	1	1
1	**0**	**0**	**0**	**0**	**0**	**0**	1	1	1	1	1	1	1
1	**0**	**0**	**0**	**0**	**1**	1	**0**	1	1	1	1	1	1
1	**0**	**0**	**0**	**1**	**0**	1	1	**0**	1	1	1	1	1
1	**0**	**0**	**0**	**1**	**1**	1	1	1	**0**	1	1	1	1
1	**0**	**0**	**1**	**0**	**0**	1	1	1	1	**0**	1	1	1
1	**0**	**0**	**1**	**0**	**1**	1	1	1	1	1	**0**	1	1
1	**0**	**0**	**1**	**1**	**0**	1	1	1	1	1	1	**0**	1
1	**0**	**0**	**1**	**1**	**1**	1	1	1	1	1	1	1	**0**

第六节 有符号二进制数的表示及运算

符号数表示

符号数+真值，符号数部分为“0”代表正，“1”代表负。通常符号数+真值共 8 位长。

符号数有三种表示方法，原码反码和补码

原码

最高位为符号位（用“0”表示正，用“1”表示负），其余为真值部分。

反码

对一个机器数X：

若X>0，则 $[X]_{反}=[X]_{原}$

若X<0，则 $[X]_{反}$= 对应原码的符号位不变，数值部分按位求反

补码

定义：

若X>0，则$[X]_{补}=[X]_{反}=[X]_{原}$

若X<0，则$[X]_{补}=[X]_{反}+1$

存在一个特殊数字 10000000，若它为无符号数，则它表示 128。

若它为原码，则它表示的数为-0；若它为反码，则表示的数为-127（即原码为 11111111），若它为补码，则它表示的数为-128（这个可以说是人为规定的）

注意：原码为 10000000 的数的补码所代表的数不是-128，它的补码是 00000000，有一位溢出了。而 10000000 为补码时则规定其表示-128。

符号数表示范围

对8位二进制数：

原码： -127～+127

反码： -127～+127

补码： -128～+127

符号二进制数与十进制数的转换

（个人总结）规律：补码与其对应真值的符号是相同的。
对于补码为正的数，真值即为后七位
对于补码为负的数，再对其求补码就可以得到真值

符号数算术运算

通过引进补码，可将减法运算转换为加法运算。

即：$[X+Y]_{补}=[X]_{补}+[Y]_{补}$

$[X-Y]_{补}=[X+(-Y)]_{补}$

$=[X]_{补}+[-Y]_{补}$

注：运算时符号位须对齐

【-Y】补 可以对【Y】补 求变补或者直接对-Y 求补得到
变补：对【Y】补 按位取反（包括符号位）后加 1 即可得到【-Y】补

符号数算数溢出问题

两个带符号的二进制数相加减时，若运算结果超出可表达范围就会产生溢出
溢出的判断方法：最高位与次高位进借位状态不同便产生溢出
例如（本题中第七位向第八位进了一位，而第八位没有进位行为）

若：X=01111000， Y=01101001

则：X+Y=

$$\begin{array}{r} 01111000 \\ +\ 01101001 \\ \hline 11100001 \end{array}$$

次高位向最高位有进位，而最高位向前无进位，产生溢出。

注：除法运算溢出时，会产生“除数为 0”中断（这个是后面的内容）
乘运算不会产生溢出