Datalab

Integer

bitXor(x, y)

这个任务等于说是要用与门和非门实现异或门。可以画个卡诺图得到与或非式然后化简成与非式，即 $x \oplus y = \neg((x \wedge y) \vee (\neg x \wedge \neg y)) = (\neg(x \wedge y) \wedge (\neg(\neg x \wedge \neg y)))$。

tmin()

返回int32的最小值，也就是`0b100...0`（符号位为1，其余都为0）。

isTmax(x)

判断x是否为int32的最大值，也就是`0b011...1`（符号位为0，其余都为1）。注意到它有一个性质：加一和取反后相等。如果x是tmax，应该满足`x+1 == ~x`。具有此性质的还有一个数负一，因此只要再判断一下x不是负一（`!!~x`）即可。

allOddBits(x)

对x所有的奇数位取与。可以用较小的立即数和移位构造一个mask，用`x & mask`来提取出x所有的奇数位，再检测它是否与mask相等。

negate(x)

求`-x`，也就是x的2's complement，即取反后加一。

isAsciiDigit(x)

判断x是否在一个范围内。构造mask提取出x的两个十六进制位分别判断：较大位判等，较小位判小于A（十）。可以通过检测不等号两侧差的符号位来判断不等式。

conditional(x, y, z)

这题想了一会才想到利用逻辑移位的trick。先用`!x`把x的信息压到最低位上（全零则为1，否则为0），然后左移到符号位上再右移，利用逻辑右移的补充特性置好所有位。如果x原先为0，此时x所有位都被置为1。最后通过`(y & ~x) | (z & x)`完成选择。

isLessOrEqual(x, y)

判断$x \leq y$。直接相减可能溢出，需要分情况讨论：如果x和y异号，可以通过各自符号直接得出结果；如果同号，则x-y不会溢出，可以检测x-y的符号位进行比较。

logicalNeg(x)

不用`!`实现`!`。这题本来思路聚焦在怎么收集到各个非零位，后来想到可以转而判断x是否为0。0的特性之一是`0 == 0<<1`，至于判等，可以通过检查左减右和右减左的符号位，两个符号位都为零当且仅当左右相等。

howManyBits(x)

给出需要多少二进制位表示x，也就是问x的信息量。这题很有趣，可以用类似二分的思路。由于能用多少位表示一个数显然也能用这么多位表示其补码，可以把x先转换成正的便于处理（最后需要加上符号位）。主体是进行一次二分查找，每次把x分成两半，检查较大的半边是否有非零位，如果有，我们就至少需要表示半边所需的bit数，把它加到答案里，然后继续在较大半边内二分；否则加0，然后在较小半边内二分。因为x有$31 \approx 2^5$个可能位，需要这样重复五次。

Float

floatScale2(uf)

对给定浮点数乘2。处理浮点数的题都是先把三个部分提取出来然后分情况处理。如果exp是最大值`0xFF`，说明是Inf或者NaN，不作处理；如果是denormalized，直接在小数部分左移（有可能从denormalized到normalized，但这时小数部分会直接overflow到指数部分，所以神奇地不需要做特别处理）；如果是normalized，直接在指数部分加一。

floatFloat2Int(uf)

把给定浮点数转为整数。同样把三个部分提出来分情况处理。如果指数部分小于127说明是小数，直接返回0；如果减完偏置大于30说明超范围了，返回-2^31（如果转换完就应该是这个数其实没有超范围，但反正都是返回这个数）；剩下的就是Norm的情况，normalize完再移位即可。

floatPower2(x)

返回2^x的浮点表示。指数部分表示Norm数的范围是$[1, 254]$，偏置完对应的指数范围是$[-126, 127]$，因此如果x大于127直接返回Inf；如果x在此范围内就直接用x设好exp；如果x $\in [-149, -127]$则是Denorm，直接用x设好小数部分；如果x比-149还小就返回0。