B0911006Y-01: Computer Organization and Design

2023 Spring

Homework 3 — March 15

Lecturer: Ke Zhang Completed by: 吉骏雄

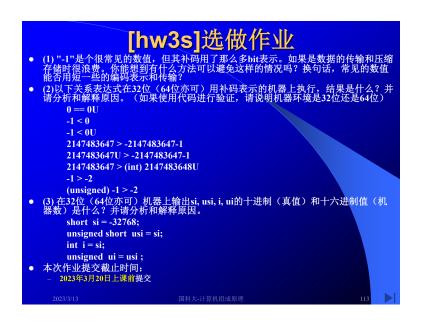


图 3.1. hw3s 选做作业

(1) "-1" 是个很常见的数值,但其补码用了那么多 bit 表示。如果是数据的传输和压缩存储时很浪费。你能想到有什么方法可以避免这样的情况吗? 换句话,常见的数值能宜角短一些的编码表示和传输?

解 可以使用变长码, 较小的数字使用更短的编码存储. 例如:

- 0xxxxxxx 表示最短的数字, 仅使用 1bit 符号位和 6bit 数字位. 00000001 表示 1, 01111111 表示 -1.
- 10xxxxxx xxxxxxx 表示更长的的数字, 使用 1bit 符号位和 13bit 数字位. 10000001 00000001 表示 257, 10111110 11111111 表示 -257.
- 110xxxxx xxxxxxxx xxxxxxxx 表示更长的的数字. 并以此类推.
- (2) 以下关系表达式在 32 位 (64 位亦可) 用补码表示的机器上执行, 结果是什么? 并请分析和解释原因. (如果使用代码进行验证, 请说明机器环境是 32 位还是 64 位)

解 结果如下 (1 代表正确, 0 代表错误):

1: 0 == 0 U 1: -1 < 0 0: -1 < 0 U 1: 2147483647 > -2147483647-1 0: 2147483647 U > -2147483647-1 1: 2147483647 > (int) 2147483648 U 1: -1 > -2 1: (unsigned) -1 > -2

使用的是 32 位 Ubuntu 虚拟机, Windows10 以及 wsl 上的 Ubuntu 系统.

-1 < 0U 的比较中, 整体被转换成无符号整数比较, -1 对应的是 fffffff, 非常大.

2147483647U > -2147483647-1 的比较中同理, 后者没有溢出, 但是按照无符号数比较, 前者大.

2147483647 > (int) 2147483648U 的比较被强制转换成整型, 按照整形比较自然是正数比负数大 (后者被转换成负数-2147483648)

(unsigned) -1 > -2 中,即便使用无符号数两者都会溢出,但是前者的模更大 (ffffffff)