

# 位扩展与位截断、移位



# 位扩展



- 进行数据类型转换时，当短数向长数转换
- C语言没有明确的位扩展运算
- 扩展两种方式：
  - 0扩展：用于无符号数
  - 符号扩展：用于补码表示的带符号数
  - 能保留数值原有的含义



## 举例：无符号数的“0”扩展

- 假设编译器规定int和short类型长度分别为32位和16位， 若有下列C语言语句：
- `unsigned short x=65530; // : 0X FFFA`
- `unsigned int y=x;`
- 得到的y的机器数是 ( )
- A. 0000 7FFA H      B. 0000 FFFA H
- C. FFFF 7FFA H      D. FFFF FFFA H

答案 (B)

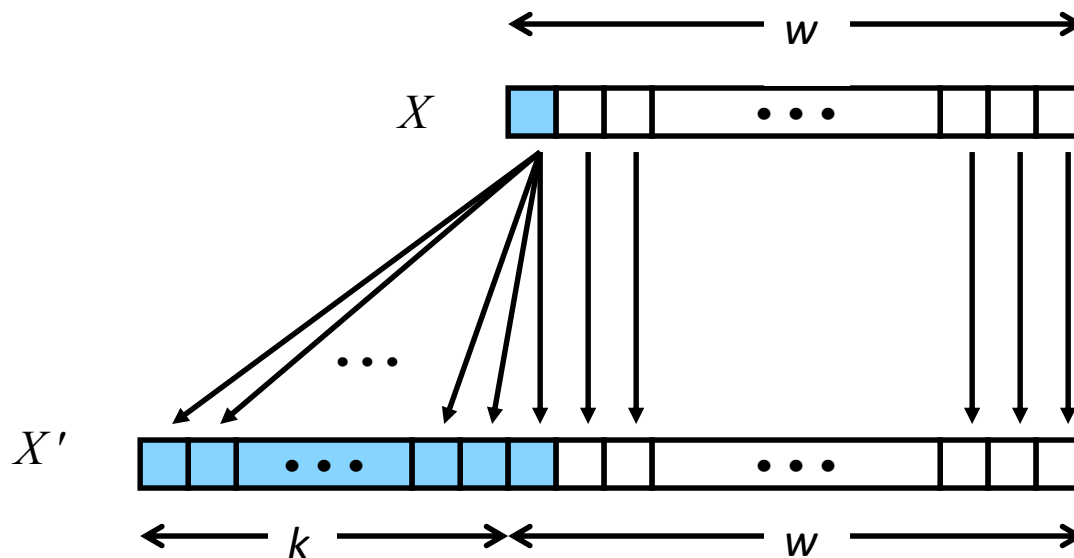
# 带符号位的扩展



- 规则：将符号位扩展到增添的位数上

$$X' = \underbrace{X_{w+k}, \dots, X_w}_{\text{对符号位进行复制}}, X_{w-1}, X_{w-2}, \dots, X_0$$

对符号位进行复制





# 举例:带符号位的扩展 Sign Extension



```
short int x = 15213;  
int      ix = (int) x;  
short int y = -15213;  
int      iy = (int) y;
```

|    | Decimal | Hex         | Binary                              |
|----|---------|-------------|-------------------------------------|
| x  | 15213   | 3B 6D       | 00111011 01101101                   |
| ix | 15213   | 00 00 3B 6D | 00000000 00000000 00111011 01101101 |
| y  | -15213  | C4 93       | 11000100 10010011                   |
| iy | -15213  | FF FF C4 93 | 11111111 11111111 11000100 10010011 |

- C语言: 自动完成符号位扩展



# 位截断: ( Truncating )

- 当强行将长数转换为短数时位截断
- 规则: 直接截断、重新解释结果
- 不一定能保留数值原有的含义
- `int i=32768; //0X 0000 8000H`
- `short si= (short) i; // 0X 8000H`
- `int j=si; // 0X FFFF 8000H , 代表-32768`
- 截断可能会导致程序出现意外错误, 但不会引起硬件异常或错误报告, 错误隐蔽性高, 需要警惕

# 本节内容



- 移位

# C语言支持的移位操作



- **左移** Left Shift:  $x \ll y$ 
  - 将位向量 **x** 左移 **y** 位
    - 高位移除，低位补0
- **右移** Right Shift:  $x \gg y$ 
  - 将位向量 **x** 右移 **y** 位
  - 低位移除
  - 逻辑右移：高位 填0
  - 算术右移：补符号位

|                   |          |
|-------------------|----------|
| Argument <b>x</b> | 01100010 |
| $\ll 3$           | 00010000 |
| Log. $\gg 2$      | 00011000 |
| Arith. $\gg 2$    | 00011000 |

|                   |          |
|-------------------|----------|
| Argument <b>x</b> | 10100010 |
| $\ll 3$           | 00010000 |
| Log. $\gg 2$      | 00101000 |
| Arith. $\gg 2$    | 11101000 |



# 移位运算的特点



## ■ 左移 <<

- 不区分逻辑左移还是算术左右，都是抛弃最高位
- 每左移一位，相当于数值乘以2
- 左移可能会发生溢出：超出数值表达范围，会得到错误值

## ■ 右移 >> (算术、逻辑)

- C语言无明确规定，算术右移 or 逻辑右移，C编译器根据数据类型（无符号、有符号补码）自动选择一种
- 每右移一位，相当于数值除以2

# 移位操作



- MIPS处理器中的移位指令
  - 对应了C语言编译器的要求
  - sll (左移)
  - sra (算术右移)
  - srl (逻辑右移)
- 当移位量大于数据字长时
  - 例如字长  $w=32$  ,  $x \ll y$
  - 等价于  $x \ll (y \bmod w)$  , 移位量不超过  $w$  位
  - 右移规则同左移

# 小结



- C语言中类型转换时，会自动实现位扩展和位截断
- C语言中有左移、右移运算符
- 硬件的设计来源于软件的需求
  - 高级语言中的运算，会转化为底层硬件的指令实现
  - 理解这些运算的实现原理，了解程序中出现的意外结果的可能性

谢谢！

