Lesson 2
Information

ICS Seminar #9

张龄心

Sept 20, 2023

原码、反码和补码

有符号数的3种二进制表示方式

- 原码 (sign-magnitude)
    - 二进制数 + 符号位
- 反码 (ones' complement)
    - 正数的反码为本身，负数的反码: 除符号位以外其他各位取反
- 补码 (two's complement)
- 正数的补码为其本身，负数的补码等于反码 + 1

- 原码和反码中 0 有 +0 和 -0 两种表示

二进制相关

整型(以32位为例)

- Tmin = -2147483648 = -2^31, Tmax = 2147483647 = 2^31-1
    - Tmin+Tmin=0
- -x=(~x)+1

浮点数: 符号位+阶码+小数位

- float: 1+8+23
- double: 1+11+52

例题

下面表达式中为真的是：

A．(unsigned) -1 < -2

B．2147483647 > (int) 2147483648U

C．(0x80005942 >> 4) == 0x09005942

D．2147483647 + 1 != 2147483648

例题

下面表达式中为真的是：

A．(unsigned) -1 < -2　LRS=[100..0]=2^31 (如果是32位机器)

B．2147483647 > (int) 2147483648U　RHS=[100..0]=TMin

C．(0x80005942 >> 4) == 0x09005942

D．2147483647 + 1 != 2147483648　(?)

32 位 C90 机器会将右侧识别为 **unsigned**, 所以左右相等，不能选 D

64 位 C99 以上机器将右侧识别为 **long**, 所以左右不相等（左边会先溢出为 -0x80000000 再转成 long），可以选 D

例题

（不定项） 假设下列 int 和 unsigned 数均为 32 位

```
int x = 0x80000000;
unsigned y = 0x00000001;
int z = 0x80000001;
```

以下表达式正确的是

A. (-x) < 0

B. (-1) > y

C. (z<<3) == (z*8)

D. y*24 == z<<5 - y<<3

例题

（不定项） 假设下列 int 和 unsigned 数均为 32 位

int x = 0x80000000; = [1000...0]=Tmin. Tmin + Tmin = 0

unsigned y = 0x00000001;

int z = 0x80000001;

以下表达式正确的是

A. (-x) < 0

B. (-1) > y unsigned与signed参与运算，都转换成unsigned

C. (z<<3) == (z*8)

D. y*24 == z<<5 - y<<3 运算优先级: 四则运算>位移

运算符

优先级	运算符	结合律
1	后缀运算符：[] () . -> ++ --(类型名称){列表}	从左到右
2	一元运算符：++ -- ! ~ + - * & sizeof_Alignof	从右到左
3	类型转换运算符：(类型名称)	从右到左
4	乘除法运算符：* / %	从左到右
5	加减法运算符：+ -	从左到右
6	移位运算符：<< >>	从左到右
7	关系运算符：<<= >>=	从左到右
8	相等运算符：== !=	从左到右
9	位运算符 AND：&	从左到右
10	位运算符 XOR：^	从左到右
11	位运算符 OR：|	从左到右
12	逻辑运算符 AND：&&	从左到右
13	逻辑运算符 OR：||	从左到右
14	条件运算符：?:	从右到左
15	赋值运算符：= += -= *= /= %= &= ^= |= <<= >>=	从右到左
16	逗号运算符：，	从左到右

例题

2． 以下说法正确的是：

A．负数加上负数结果都为负数

B．正数加上正数结果都为正数

C．用&和~可以表示所有的逻辑与或非操作

D．用&和|可以表示所有的逻辑与或非操作

例题

2． 以下说法正确的是：

A．负数加上负数结果都为负数

B．正数加上正数结果都为正数

C．用&和~可以表示所有的逻辑与或非操作

D．用&和|可以表示所有的逻辑与或非操作

C

a | b = ~(~a & ~b)

a ^ b = (a & b) | (~a & ~b), 已知 | 可以表出

例题

(　)3.下面关于布尔代数的叙述,错误的是_____.

A.设x,y,z是整型,则x^y^z==y^z^x.

B.任何逻辑运算都可以由与运算(&)和异或运算(^)组合得到.

C.设m,n是char*类型的指针,则下面三条语句

“*n=*m^*n;*m=*m^*n;*m=*m^*n;“可以交换*m和*n的值.

D.已知a,b是整型,且a+b+1==0为真.则a^b+1==0为真.

例题

(　)3.下面关于布尔代数的叙述,错误的是_____.

A.设x,y,z是整型,则x^y^z==y^z^x.

B.任何逻辑运算都可以由与运算(&)和异或运算(^)组合得到.

C.设m,n是char*类型的指针,则下面三条语句

“*n=*m^*n;*m=*m^*n;*m=*m^*n;“可以交换*m和*n的值.

D.已知a,b是整型,且a+b+1==0为真.则a^b+1==0为真.

D

A. 异或满足结合律和交换律

B. 正确(课本习题)

C. 正确(指针*运算符的优先级很高)

D. 四则运算的优先级>位运算

例题

在 x86-64 机器上有以下程序片段，

```
int x = foo();
int y = bar();
unsigned ux = x;
unsigned uy = y;
```

请选出右侧表达式中恒为真的选项。

1. x / 8 == x >> 3
2. ux / 8 == ux >> 3
3. x * x >= 0
4. ux * ux >= 0
5. x <= 0 || -x < 0
6. x >= 0 || -x > 0
7. !x || (x | -x) >> 31 == -1
8. ux > -1

例题

在 x86-64 机器上有以下程序片段，

```
int x = foo();
int y = bar();
unsigned ux = x;
unsigned uy = y;
```

请选出右侧表达式中恒为真的选项。

2, 4, 5, 7

1. x / 8 == x >> 3 x<0时不成立
2. **ux / 8 == ux >> 3**
3. x * x >= 0 溢出为负数时不成立
4. **ux * ux >= 0**
5. **x <= 0 || -x < 0**
6. x >= 0 || -x > 0 x=Tmin时, -x=Tmin
7. **!x || (x | -x) >> 31 == -1**

   x!=0时, [11...1]=-1
8. ux > -1 统一转换成unsigned比较

例题

(不定项)选出下列表达式中恒成立的选项（x 为整型）

A. ((x >> 1) << 1) <= x

B. ((x / 2) * 2) <= x

C. -(x ^ y ^ (~x)) - y == -(y ^ x ^ (~y)) - x

例题

(不定项)选出下列表达式中恒成立的选项（x 为整型）

A. ((x >> 1) << 1) <= x

B. ((x / 2) * 2) <= x　x<0且为奇数时不成立

C. -(x ^ y ^ (~x)) - y == -(y ^ x ^ (~y)) – x

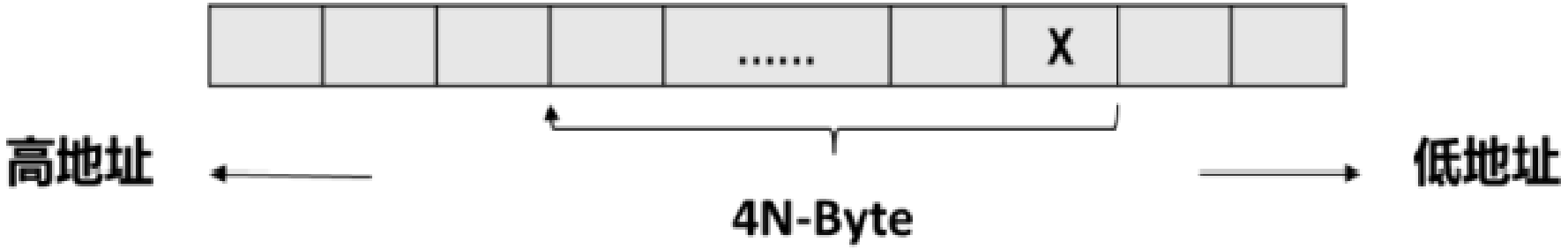
注: 异或运算满足**交换律和结合律**

大端法和小端法

- 大端法: 符合阅读习惯/直觉(从低地址读到高地址)
  - IBM / Oracle
- 小端法: 不符合直觉(反过来)
  - Intel / Android / IOS
- 双端法: 可以配置成大端/小端机器
  - ARM

例题

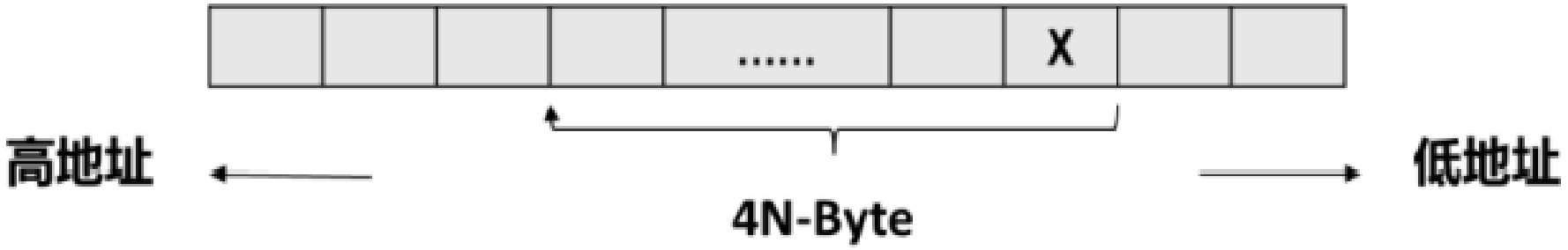
1. 假定一个特殊设计的计算机，将 int 型数据的长度从 4-Byte 扩展为 4N-Byte，采用大端法（Big Endian）。现将该 int 型所能表示的最小负数写入内存中，如下图所示。其中每个小矩形代表一个 Byte，请问 X 位置这个 Byte 中的值是多少？



A. 00000000_2　　B. 01111111_2　　C. 10000000_2　　D. 11111111_2

例题

1. 假定一个特殊设计的计算机，将 int 型数据的长度从 4-Byte 扩展为 4N-Byte，采用大端法（Big Endian）。现将该 int 型所能表示的最小负数写入内存中，如下图所示。其中每个小矩形代表一个 Byte，请问 X 位置这个 Byte 中的值是多少？



A. 00000000_2　　B. 01111111_2　　C. 10000000_2　　D. 11111111_2

C

注意本题中高地址在左低地址在右(与阅读习惯不同)

例题

1. 一个 IPv4 地址就是一个 32 位无符号整数。例如，10.2.155.253 对应的地址是 0x0a029bfd。协议规定，无论主机字节顺序如何，IP 地址在内存中总是以大端法来存放的。下面代码要实现的功能是检验 IP 是否符合 192.168.56.xx（xx 表示任意 0~255 的数）的模式，如果满足，则执行 if 语句内部指令。那么，在小端法的机器上，应该分别补充的数字是：

```
unsigned ip, mask;
// set ip
mask = ____________;
if(ip & mask == ____________)
{
    // do something
}
```

A. 0x00ffffff　　0x0038a8c0

B. 0x00ffffff　　0x00838a0c

C. 0xffffff00　　0xc0a83800

D. 0xffffff00　　0x0c8a8300

例题

1. 一个 IPv4 地址就是一个 32 位无符号整数。例如，10.2.155.253 对应的地址是 0x0a029bfd。协议规定，无论主机字节顺序如何，IP 地址在内存中总是以大端法来存放的。下面代码要实现的功能是检验 IP 是否符合 192.168.56.xx（xx 表示任意 0~255 的数）的模式，如果满足，则执行 if 语句内部指令。那么，在小端法的机器上，应该分别补充的数字是：

```
unsigned ip, mask;
// set ip
mask = ____________;
if(ip & mask == ____________)
{
    // do something
}
```

A. 0x00ffffff　　0x0038a8c0
B. 0x00ffffff　　0x00838a0c
C. 0xffffff00　　0xc0a83800
D. 0xffffff00　　0x0c8a8300

A

注: 192.168.56.xx = 0xc0 a8 38 ??

复杂类型的阅读: 右左法则

- 从变量名开始看
- 向右读, 直到遇到圆括号时, 转为向左读. 再次遇到圆括号时, 再次转向
- 即, 每次遇到圆括号时就转向

例子:

- int a[10]
- int *a[10]
- int (*a)[10]

例题

1. char **argv;
2. int (*daytab)[13];
3. int *daytab[13];
4. void *comp();
5. void (*comp)();
6. int *(*(*a)(int))[10]
7. int *(*(*arr[5])())();
8. char (*(*x())[])();
9. char (*(*x[3])())[5];

强制类型转换

- 有符号/无符号转换: 保持底层二进制表示不变即可
- 大小转换:
    - 大转小: 直接截断
        - 若有符号, 符号位保持不变
    - 小转大
        - 有符号数: 符号扩展(前面加符号位),
        - 无符号数: 零扩展(前面加0)
- 大小和符号都需要转换: **先变大小, 再变符号**
- 不同类型进行运算:
    - 有符号/无符号类型一起运算, 统一转换成无符号
    - 不同大小的类型一起运算, 统一转换成大的类型

通用原则: 同大小类型转换, 底层二进制表示不变 (适用于union等类型)

例题

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    unsigned char uc = 128;
    char c = 128;
    printf ("%d %d\n", uc == c, uc + c);
}
```

运行结果?

例题

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    unsigned char uc = 128;
    char c = 128;
    printf ("%d %d\n", uc == c, uc + c);
}
```

0 0

char: 1 byte, 范围-128~127

unsigned char: 1 byte, 范围0~255

浮点运算

- 有符号/无符号转换: 保持底层二进制表示不变即可
- 大小转换:
    - 大转小: 直接截断
        - 若有符号, 符号位保持不变
    - 小转大
        - 有符号数: 符号扩展(前面加符号位),
        - 无符号数: 零扩展(前面加0)
- 大小和符号都需要转换: **先变大小, 再变符号**
- 不同类型进行运算:
    - 有符号/无符号类型一起运算, 统一转换成无符号
    - 不同大小的类型一起运算, 统一转换成大的类型

通用原则: 同大小类型转换, 底层二进制表示不变 (适用于union等类型)

例题

1. float的阶码可表示的指数范围是?
2. double的阶码可表示的指数范围是?
3. 某种浮点数，阶码字段m位，小数字段n位 （共1+m+n位）

   问最大非规格化数？最小规格化数？ 最大规格化数？ （仅考虑正数）

例题

1. float的阶码可表示的指数范围是? -126 ~ 127
2. double的阶码可表示的指数范围是? -1022 ~ 1023
3. 某种浮点数，阶码字段m位，小数字段n位 （共1+m+n位）

   问最大非规格化数？最小规格化数？ 1？最大规格化数？ （仅考虑正数）

   最大非规格化数：0 00..00 11..11 -> $2^{2-2^{m-1}}(1-2^{-n})$

   最小规格化数：0 00..01 00...0 -> $2^{2-2^{m-1}}$

   1：0 011...11 00..0

   最大规格化数：0 11...110 11...1 -> 表达式太复杂不写了

例题

12. 下列关于教材第二章中整数和浮点数的说法中，正确的是(　　).

A. 假设a是使用补码表示的整型，则表达式"-a == ~(a + 1)"为真.

B. 假设a和b是两个负整型，则可以通过表达式"a + b > 0"是否为真来判断a + b是否产生了溢出.

C. 假设a是浮点数，且a的阶码域为零，则a一定不是规格化数.

D. 假设a，b是两个浮点数，且它们都不是非数(NaN)，则表达式"a + b == b + a"为真.

例题

12. 下列关于教材第二章中整数和浮点数的说法中，正确的是(　　).

A. 假设a是使用补码表示的整型，则表达式"-a == ~(a + 1)"为真.

B. 假设a和b是两个负整型，则可以通过表达式"a + b > 0"是否为真来判断a + b是否产生了溢出.

C. 假设a是浮点数，且a的阶码域为零，则a一定不是规格化数.

D. 假设a，b是两个浮点数，且它们都不是非数(NaN)，则表达式"a + b == b + a"为真.

A. 正确

B. Tmin+Tmin=0

C. 阶码为0 -> 非规格化数

D. Inf+(-Inf)=NaN, NaN无法互相比较

Thank you!