DataLab实验解析

转载

xuzhezhaozhao 于 2013-04-17 16:50:04 发布 5692 收藏 8
分类专栏： CSAPP 文章标签： DATA LAB CSAPP 计算机

CSAPP 专栏收录该内容
11 篇文章 0 订阅
订阅专栏

清华大学的计算机课程个个都很变态，本想这学期选点不考试的课程好轻松一下，不巧的是你看我选的这门课，微计算机系统设计，老师第一节课就说该课程不考试，就做点试验，也不写论文。当时俺还暗自庆幸可选到了一门轻松的课了，然后老师把大学里的汇编语言的知识讲了下。选这课的共*17*人，老师也不点名，因此几乎除了第一节课外其他时候就只有*10*个左右的学生来听课。倒数几节课就*5~6*人来了，最后一节课老师就直接给我们放假了，不上了。

可是哪有天上掉馅饼的事啊，*5*月中旬的时候，老师说别忘了咱们还有一个*Project* 和*5*个实验，上完这节课服务器上就上传实验了。

对于我来说做那实验简直是头疼死了，不是说耗费的时间问题，而是我们做实验必须到实验室去，然后登陆到服务器上来做，实验室有时还不开。

还有那个*Project*：*investegating X64 mode* （娘的，做个操作系统了最后）

废话不说了看看第一个实验是啥吧

"Lab Assignment L1: Manipulating Bits" 哎呀 这是*CMU*大学的实验嘛不是，这有难度了。

看下实验目标

" The purpose of this assignment is to become more familiar with bit-level representations and manipulations.You'll do this by solving a series of programming "puzzles". Many of these puzzles are quite arti?cial, butyou'll ?nd yourself thinking much more about bits in working your way through them. "

妈呀是猜谜语吗？？

实验要求看下呗

" The ?le btest.c allows you to evaluate the functional correctness of your code. The ?le README containsadditional documentation about btest. Use the command make btest to generate the test code and runit with the command ./btest. The ?le dlc is a compiler binary that you can use to check your solutionsfor compliance with the coding rules. The remaining ?les are used to build btest.Looking at the ?le bits.c you'll notice a C structure team into which you should insert the requestedidentifying information about yourself. Do this right away so you don't forget.

The bits.c ?le also contains a skeleton for each of the 15 programming puzzles. Your assignment is tocomplete each function skeleton using only straightline code (i.e., no loops or conditionals) and a limitednumber of C arithmetic and logical operators. Speci?cally, you are only allowed to use the following eight operators: ! ? & ? | + << >>"

这不是只能用几个破符号做题啊！有意思哈！

根据实验要求看下*bits.c*卖的是什么药吧！

打开文件*bits.c*首先看到的就是编写规范

" CODING RULES:

Replace the "return" statement in each function with one

or more lines of C code that implements the function. Your code *must conform to the following style:*

```
int Funct(arg1, arg2, ...) {
    /* brief description of how your implementation works */
    int var1 = Expr1;
    ...
    int varM = ExprM;

    varJ = ExprJ;
    ...
    varN = ExprN;
    return ExprR;
}
```

Each "Expr" is an expression using ONLY the following:

1. Integer constants 0 through 255 (0xFF), inclusive. You are

not allowed to use big constants such as 0xffffffff.

2. Function arguments and local variables (no global variables).

3. Unary integer operations ! ~

4. Binary integer operations & ^ | + << >>

Some of the problems restrict the set of allowed operators even further.

Each "Expr" may consist of multiple operators. You are not restricted to

one operator per line.

You are expressly forbidden to:

1. Use any control constructs such as if, do, while, for, switch, etc.

2. Define or use any macros.

3. Define any additional functions in this file.

4. Call any functions.

5. Use any other operations, such as &&, ||, -, or ?:

6. Use any form of casting.

You may assume that your machine:

1. Uses 2s complement, 32-bit representations of integers.

2. Performs right shifts arithmetically.

3. Has unpredictable behavior when shifting an integer by more

than the word size.”

看明白了吧！接下来看题吧！

```
/*
 * bitNor - ~(x|y) using only ~ and &
 *   Example: bitNor(0x6, 0x5) = 0xFFFFFFF8
 *   Legal ops: ~ &
 *   Max ops: 8
 *   Rating: 1
 */
int bitNor(int x, int y) {
  return 2;
}
```

这个题要求是用~和&实现位级或非这个不难吧，最多*8*个符号，分数*1*分， 但是要注意的是，现实是现实，计算机是计算机，这是两码子事，计算机中数值的表达采用的是补码。要注意哦！

这个我做了

```
return (~x)&(~y);
```

第二个

```
/*
 * bitXor - x^y using only ~ and &
 *   Example: bitXor(4, 5) = 1
 *   Legal ops: ~ &
 *   Max ops: 14
 *   Rating: 2
 */
int bitXor(int x, int y) {
  return 2;
}
```

类似上题

```
  int a=(~x)&y;
  int b=x&(~y);
  return ~(~a&~b);
```

第三个

```
/*
 * isNotEqual - return 0 if x == y, and 1 otherwise
 *   Examples: isNotEqual(5,5) = 0, isNotEqual(4,5) = 1
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 6
 *   Rating: 2
 */
int isNotEqual(int x, int y) {
  return 2;
}
```

答案：return !(!(x+(~y)+1));

第四个

```
/*
 * getByte - Extract byte n from word x
 *   Bytes numbered from 0 (LSB) to 3 (MSB)
 *   Examples: getByte(0x12345678,1) = 0x56
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 6
 *   Rating: 2
 */
int getByte(int x, int n) {
  return 2；
}
```

答案:return (x>>(n<<3)&0xff);

第五个

```
/*
 * copyLSB - set all bits of result to least significant bit of x
 *   Example: copyLSB(5) = 0xFFFFFFFF, copyLSB(6) = 0x00000000
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 5
 *   Rating: 2
 */
int copyLSB(int x) {
        return 2；
}
```

答案： return (x<<31)>>31;

第六个

```
/*
 * logicalShift - shift x to the right by n, using a logical shift
 *   Can assume that 1 <= n <= 31
 *   Examples: logicalShift(0x87654321,4) = 0x08765432
 *   Legal ops: ~ & ^ | + << >>
 *   Max ops: 16
 *   Rating: 3
 */
int logicalShift(int x, int n) {
  return 2;
}
```

答案：

```
  int mask=0x01<<(32+1+~n);
  mask=mask+~0;
  return (x>>n)&mask;
```

第七个（这个有点难度哈！）

```
/*
 * bitCount - returns count of number of 1's in word
 *   Examples: bitCount(5) = 2, bitCount(7) = 3
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 40
 *   Rating: 4
 */
int bitCount(int x) {
return 2;
}
```

答案：

```
int mask0=(0x11<<8)|0x11;  //0x1111
 int mask1=(mask0<<16)|mask0; //0x11111111
 int result=x&mask1;
 int mask2=(0x0f<<8)|0x0f; //0x0f0f
 result=result+((x>>1)&mask1);
 result=result+((x>>2)&mask1);
 result=result+((x>>3)&mask1);
 result=((result>>16)+result);
 int result1=result&mask2;  //result&0x0f0f
 result=(result>>4)&mask2;
 result=result+result1;
 result=(result+(result>>8))&0x3f;//最大32 用6位表示
 return result;
```

第八个

```
/*
 * bang - Compute !x without using !
 *   Examples: bang(3) = 0, bang(0) = 1
 *   Legal ops: ~ & ^ | + << >>
 *   Max ops: 12
 *   Rating: 4
 */
int bang(int x) {
 return 2；
}
```

答案：

```
return (((~x)&~(~x+1))>>31)&0x01;
```

第九个

```
/*
 * leastBitPos - return a mask that marks the position of the
 *               least significant 1 bit. If x == 0, return 0
 *   Example: leastBitPos(96) = 0x20
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 6
 *   Rating: 4
 */
int leastBitPos(int x) {
        return 2;
}
```

答案:

```
  return x&(~x+1);
```

第十个

```
/*
 * TMax - return maximum two's complement integer
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 4
 *   Rating: 1
 */
int tmax(void) {
 return 2;
}
```

答案：*return ~(0x1<<31);*

第十一个

```
/*
 * isNonNegative - return 1 if x >= 0, return 0 otherwise
 *   Example: isNonNegative(-1) = 0.  isNonNegative(0) = 1.
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 6
 *   Rating: 3
 */
int isNonNegative(int x) {
         return 2;
}
```

答案：

```
  return !((x>>31)&0x01);
```

第十二个

```
/*
 * isGreater - if x > y  then return 1, else return 0
 *   Example: isGreater(4,5) = 0, isGreater(5,4) = 1
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 24
 *   Rating: 3
 */
int isGreater(int x, int y) {
return 2;
}
```

答案：*int signofx=x>>31;*

```
 int signofy=y>>31;
 int signequal=(!(signofx ^ signofy)) & ((~y + x) >> 31);
 int signnequal=signofx & !signofy;
  return !(signequal | signnequal);
```

第十三个

```
/*
 * divpwr2 - Compute x/(2^n), for 0 <= n <= 30
 *  Round toward zero
 *   Examples: divpwr2(15,1) = 7, divpwr2(-33,4) = -2

 *   Legal ops: ! ~ & ^ | + << >>

 *   Max ops: 15

 *   Rating: 2

 */

int divpwr2(int x, int n) {

         return 2;

}
```

答案：

```
  int signofx=x>>31;

  int mask = (1 << n) + ~0;

  return ((signofx&mask)+x)>>n;
```

第十四个

```
/*

 * abs - absolute value of x (except returns TMin for TMin)

 *   Example: abs(-1) = 1.

 *   Legal ops: ! ~ & ^ | + << >>

 *   Max ops: 10

 *   Rating: 4

 */

int abs(int x) {

 return 2;

}
```

答案：

```
int sign_x = x >> 31;

  return((x ^ (sign_x)) + (1 + ( ~(sign_x))));
```

第十五个

```
/*

 * addOK - Determine if can compute x+y without overflow

 *   Example: addOK(0x80000000,0x80000000) = 0,

 *            addOK(0x80000000,0x70000000) = 1,

 *   Legal ops: ! ~ & ^ | + << >>

 *   Max ops: 20

 *   Rating: 3

 */

int addOK(int x, int y) {
 return 2;
}
```

答案：

```
int xysum = x + y;
  int signx = x >> 31;
  int signy = y >> 31;
  int signsumxy = xysum >> 31;
  return !(~(signx ^ signy) & (signx ^ signsumxy));
```

*15*个到此结束，终于是整完了，耗费了我两天的时间啊！

原文地址: http://blog.163.com/bccuypsj@126/blog/static/68107486201152195113297/