

# 第二次作业

作业发布时间：2023/05/08 星期一

本次作业要求如下：

1.截止日期：2023/06/01 周四晚 24:00

2.后面发布提交作业网址

3.命名格式：附件和邮件命名统一为“第二次作业+学号+姓名”，作业为 PDF 格式；

4.注意：

(1) 选择、判断和填空只需要写答案，大题要求有详细过程，过程算分。

(2) 答案请用另一种颜色的笔回答，便于批改，否则视为无效答案。

(3) 大题的过程最好在纸上写了拍照，放到 word 里。

5.本次作业遇到问题请联系课程群里的助教。

## 一. 选择题

1. 位于存储器层次结构中的最顶部的是(A)。

A. 寄存器 B. 主存 C. 磁盘 D. 高速缓存

2. 关于 Intel 的现代 X86-64 CPU，说法正确的是(B)。

A. 属于 RISC B. 属于 CISC C. 属于 MISC D. 属于 AVX

3. 操作系统在管理硬件时使用了几个抽象概念，其中(A)是对处理器、主存和 I/O 设备的抽象表示。

A. 进程 B. 虚拟存储器 C. 文件 D. 虚拟机

4. 以下有关编程语言的叙述中，正确的是(D)。

A. 计算机能直接执行高级语言程序和汇编语言程序

B. 机器语言可以通过汇编过程变成汇编语言

C. 汇编语言比高级语言有更好的可读性

D. 汇编语言和机器语言都与计算机系统结构相关

5. 给定字长的整数  $x$  和  $y$  按补码相加，和为  $s$ ，则发生正溢出的情况是(A)

A.  $x>0, y>0, s\leq 0$  B.  $x>0, y<0, s\leq 0$

C.  $x>0, y<0, s\geq 0$  D.  $x<0, y<0, s\geq 0$

6. 设机器数字长 8 位（含 1 位符号位），若机器数 DAH 为补码，分别对其进行算术左移一位和算术右移一位，其结果分别为 (A)

A. B4H, EDH

B. B5H, 6DH

C. B4H, 6DH

D. B5H, EDH

7. -1029 的 16 位补码用十六进制表示为( C )。

A. 8405H

B. 0405H

C. FBFBH

D. 7BFBH

8. 已知两个正浮点数,  $N_1 = 2^{j_1} \times S_1$ ,  $N_2 = 2^{j_2} \times S_2$ , 当下列 ( A ) 成立时,  $N_1 < N_2$ 。

A.  $S_1$ 和 $S_2$ 均为规格化数, 且 $J_1 < J_2$

B.  $S_1 < S_2$

C.  $S_1$ 和 $S_2$ 均为规格化数, 且 $J_1 > J_2$

D.  $J_1 < J_2$

9. C 程序执行到整数或浮点变量除以 0 可能发生( A )。

A. 显示除法溢出错直接退出

B. 程序不提示任何错误

C. 可由用户程序确定处理方法

D. 以上都可能

10. 补码加法运算的溢出判别中, 以下说法正确的是( D )

A. 符号相同的两个数相加必定不会发生溢出

B. 符号不同的两个数相加可能发生溢出

C. 符号相同的两个数相加必定发生溢出

D. 符号不同的两个数相加不可能发生溢出

11. 假定变量 i、f 的数据类型分别是 int、float。已知 i=12345, f=1.2345e3, 则在一个 32 位机器中执行下列表达式时, 结果为“假”的是( C )。

A.  $i == (\text{int})(\text{float})i$

B.  $i == (\text{int})(\text{double})i$

C.  $f == (\text{float})(\text{int})f$

D.  $f == (\text{float})(\text{double})f$

12. 以下关系表达式, 结果为“真”的是( B )。

A.  $2147483647\text{U} > -2147483648$

B.  $(\text{unsigned}) -1 > -2$

C.  $-1 < 0\text{U}$

D.  $2147483647 < (\text{int}) 2147483648\text{U}$

13. 假定某数采用 IEEE 754 单精度浮点数格式表示为 00000001H, 则该数的值是( B )。

A. NaN (非数)

B.  $1.0 \times 2^{(-149)}$

C.  $1.00...01 \times 2^{(-127)}$

D.  $1.0 \times 2^{(-150)}$

14. C 语言程序如下，下列说法叙述正确的是( C )。

```
#include <stdio.h>
#define DELTA sizeof(int)
int main(){
    int i;
    for (i = 40; i - DELTA >= 0; i -= DELTA)
        printf("%d ",i);
}
```

- A. 程序有编译错误
- B. 程序输出 10 个数：40 36 32 28 24 20 16 12 8 4 0
- C. 程序死循环，不停地输出数值
- D. 以上都不对

15. 若 int 型变量 x 的最高有效字节全变 0，其余各位不变，则对应 C 语言表达式为( A )。

- A.  $((\text{unsigned}) x \ll 8) \gg 8$
- B.  $((\text{unsigned}) x \gg 8) \ll 8$
- C.  $(x \ll 8) \gg 8$
- D.  $(x \gg 8) \ll 8$

## 二. 填空题

1. 64 位系统中 short 数 -2 的机器数二进制表示 1111 1111 1111 1110。
2. 判断整型变量 n 的位 7 为 1 的 C 语言表达式是  $(n \gg 7) \& 1 == 1$ 。
3. -1024 采用 IEEE 754 单精度浮点数格式按内存地址从低到高表示的结果（十六进制表示，小端模式）是 0x 00 00 80 c4。
4. C 语言中的 double 类型浮点数用 64 位表示。
5. 64 位系统中，整型变量 x = -7，其在内存从低地址到高地址依次存放的数是 f9 ff ff ff (十六进制表示，小端模式)。

## 三. 判断题

1. ( X ) C 浮点常数 IEEE754 编码的缺省舍入规则是四舍五入。
2. ( V ) 浮点数 IEEE754 标准中，规格化数比非规格化数多。
3. ( V ) 对 unsigned int x,  $(x * x) \geq 0$  总成立。
4. ( X ) CPU 无法判断加法运算的和是否溢出。
5. ( X ) C 语言中的有符号数强制转换成无符号数时位模式不会改变。
6. ( X ) C 语言中数值从 int 转换成 double 后，数值虽然不会溢出，但是可能是不精确的。
7. ( X ) C 语言中从 double 转换成 float 时，值可能溢出，但不可能被舍入。

## 四. 分析题

1. 请说明 float 类型编码格式，并按步骤计算 -10.1 的各部分内容，写出 -10.1 在内存从低地址到高地址的存储字节内容（小端系统）。

将-10.1 转化为二进制, 有  $-10.1 = -(1010.0011\ 0011\ \dots) = -(1.0100\ 0011\ 0011\ \dots) \times 2^3$ .

因此, 符号位为 1, 指数位为  $127+3 = 130 = 1000\ 0010$ , 底数为  $0100\ 0011\ 0011\ 0011\ 0011\ 0011\ 010$ , 则二进制编码为  $1\ 1000\ 0010\ 0100\ 0011\ 0011\ 0011\ 0011\ 0011\ 010$  在小端系统中, 为 9a 99 21 c1.

2. 写出负 short 型整数  $x$  除以 2 的整数幂 ( $k$ ) 的商的公式, 当  $x = -17231$  时, 计算  $x/2^6$  的值的十六进制表示, 写出结果在内存从低地址到高地址的存储字节内容 (小端系统)。

公式:  $(x + (1 \ll k) - 1) \gg k$

当  $x = -17231 = 1011\ 1100\ 1011\ 0001$ ,  $x + (1 \ll 6) - 1 = 1011\ 1100\ 1101\ 0000$ ,

因此结果为  $1111\ 1110\ 1111\ 0011$ .

小端系统中用 f3 ef 表示。

3. 向量元素和计算的相关程序如下, 请改写或重写计算函数 `vector_sum`, 进行速度优化, 并简要说明优化的依据。(如果能自己动手在电脑上测试一下, 优化前后性能提升了多少会有额外加分, 贴上截图, 注明机器型号)

```
/*向量的数据结构定义 */
typedef struct{
    int len;    //向量长度, 即元素的个数
    float *data; //向量元素的存储地址
} vec;

/*获取向量长度*/
int vec_length(vec *v){return v->len;}

/* 获取向量中指定下标的元素值, 保存在指针参数 val 中*/
int get_vec_element(*vec v, size_t idx, float *val){
    if (idx >= v->len)
        return 0;
    *val = v->data[idx];
    return 1;
}

/*计算向量元素的和*/
void vector_sum(vec *v, float *sum){
    long int i;
    *sum = 0; //初始化为 0
    for (i = 0; i < vec_length(v); i++) {
        float val;
        get_vec_element(v, i, &val); //获取向量 v 中第 i 个元素的值, 存入 val 中
        *sum = *sum + val;    //将 val 累加到 sum 中
    }
}
```

1. 将 `vec_length` 从循环中取出，保存在 `len` 变量当中，减少调用。
2. 定义局部变量 `my_sum`，加到 `my_sum` 上，最后赋值给 `sum`。减少访存。
3. 将 `val` 移到循环外，减少栈空间分配/释放。

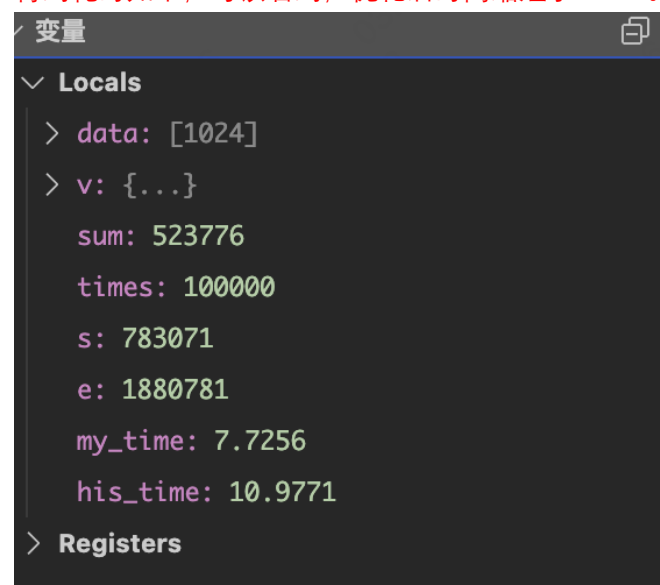
修改后代码如下：

```
void my_vector_sum(vec *v, float *sum){
    long int i;
    int len = vec_length(v);
    float my_sum = 0;
    float val;
    for ( i = 0; i < len; i++)
    {
        get_vec_element(v, i, &val);
        my_sum += val;
    }
    *sum = my_sum;
}
```

测试代码如下：

```
int main(){
    float data[1024];
    for (size_t i = 0; i < 1024; i++)
    {
        data[i] = (float) i;
    }
    vec v;
    v.len = 1024;
    v.data = data;
    float sum;
    int times = 100000;
    clock_t s = clock();
    for (size_t i = 0; i < times; i++)
    {
        my_vector_sum(&v, &sum);
    }
    clock_t e = clock();
    double my_time = (double)(e - s)/times;
    s = clock();
    for (size_t i = 0; i < times; i++)
    {
        vector_sum(&v, &sum);
    }
    e = clock();
    double his_time = (double)(e - s)/times;
    return 0;
}
```

得到耗时如下，可以看到，优化后时间缩短了 29.6%。



The screenshot shows a variable inspector window with a dark background. The title bar is labeled '变量' (Variables) with a copy icon on the right. The main content area is titled 'Locals' and contains a list of variables with their values. The variables are: data: [1024], v: {...}, sum: 523776, times: 100000, s: 783071, e: 1880781, my\_time: 7.7256, and his\_time: 10.9771. Below the 'Locals' section is a 'Registers' section, which is currently collapsed.

```
变量
Locals
> data: [1024]
> v: {...}
    sum: 523776
    times: 100000
    s: 783071
    e: 1880781
    my_time: 7.7256
    his_time: 10.9771
> Registers
```

测试机器：Macbook Pro CPU: 2.6 GHz 六核 Intel Core i7