第 2 次作业

作业文件命名格式： 专业班级_学号_姓名_第几次作业.pdf

例如：CS2201_U202212345_张三_2.pdf

提交到 qq 邮箱：**2112745268@qq.com**

提交截止时间：2024.03.21

1、Intel X86-64 (即 x64) 位 CPU 中，有哪些通用的 64 位寄存器？通用的 32 位的寄存器？通用的 16 位寄存器？通用的 8 位寄存器？（只需要列出符号名即可，不用给出寄存器的中文名称）

64 位通用寄存器： rax、rbx、rcx、rdx、rbp、rsi、rdi、rsp、r8 ~ r15

32 位通用寄存器： eax、ebx、ecx、edx、ebp、esi、edi、esp、r8d ~ r15d

16 位通用寄存器： ax、 bx、cx、dx、bp、si、di、sp、r8w ~ r15w

8 位通用 寄存器： al、bl、cl、dl、bpl、sil、dil、spl、r8b ~ r15b

2、Intel X86-64 CPU 中，64 位的指令指示器 RIP 中存放的是什么？程序运行的基本过程是什么？（包含 RIP 是何时变化的）

RIP 存放的是下一条指令的偏移地址。

程序运行的基本过程： CPU 根据 CS:RIP 的内容，从主存中读取一条指令的机器码，同时修改 RIP 的内容（下一条指令的偏移地址）。CPU 的译码部件对取到 CPU 的指令的机器码进行译码，然后交给执行部件执行该指令（如果是转移指令则会修正 RIP 的值）。最后 CPU 将执行结果写回主存或寄存器。

3、编译器在生成执行程序时，可以做哪些优化工作？为什么做相应的工作可以提高程序的执行速度？

将分支转移优化为顺序执行（提高 cache 的命中率）、尽量使用 SIMD 指令、用寄存器表示变量、尽量形成并行流水线操作。

4、已知 8 位二进制数 x1 和 x2 的值，请写出 [x1]补、[x2]补 各是多少？

[x1]补＋[x2]补 后的结果是多少，以及标志位 SF、ZF、CF、OF 各是多少？

(1) x1＝＋0110011B； x2＝＋1011010B

(2) x1＝－0101001B； x2＝－1011101B

(3) x1＝＋1100101B； x2＝－1011101B

(1) x1＝＋0110011B；　x2＝＋1011010B

[x1]补　=　x1　=　0011 0011

[x2]补　=　x2　=　0101 1010

[x1]补 + [x1]补 = 1000 1101

SF = 1、ZF = 0、CF = 0、OF = 1

(2) x1＝－0101001B；　x2＝－1011101B

x1　=　－0010 1001　　x2　=　－0101 1101

[x1]补　= 1101 0111　　[x2]补　= 1010 0011

[x1]补 + [x1]补 = 0111 1010

SF = 0、ZF = 0、CF = 1、OF = 1

(3) x1＝＋1100101B；　x2＝－1011101B

[x1]补　=　x1　=　0110 0101

x2　=　－0101 1101　　[x2]补　=　1010 0011

[x1]补 + [x1]补 = 0000 1000

SF = 0、ZF = 0、CF = 1、OF = 0

5、对如下 C语言程序，用VS2019（Intel CPU，x86-debug）编译、链接、调试运行。

```
int main( )
{
    int  a = 100;    //0x64
    int  b = 0x12345678;
    int  r = 0;
    char msg[6] = "abcde";  // 'a'的ASCII是 0x61
    return 0;
}
```

在 return 处设置断点调试时，在监视窗口中看到变量 a 的地址（即 &a）为 0x010ffe98；变量 b 的地址（即 &b）为 0x010ffe94；变量 r 的地址为 0x010ffe90，数组 msg 的起始地址为 0x010ffe88。

以字节为单位、用 16 进制数的形式填空，最左边是内存窗口显示的内存地址。

0x010ffe88　**_61_　_62_　_63_　_64_　_65_　_00_**　XX　XX

0x010ffe90　**_00_　_00_　_00_　_00_　_78_　_56_　_34_　_12_**

0x010ffe98　**_64_　_00_　_00_　_00_**　XX　XX　XX　XX

说明：若同学们要实验，看到变量比较紧凑的存放，可以设置编译开关：【项目属性-> C/C++ -> 代码生成 -> 基本运行时检查 设置为 默认值】【整个平台是 x86，是 32 位地址】

6、整数数据的表示

设有 short x; 除了 x = 0 外，x 有无其他值使得 x == -x？该值是多少？说明理由。

答案：要使得 x == -x，必须 x = [-x]补，记 x 的 4 个 16 进制位为 X_1X_2 X_3X_4，即
X = X_1X_2 X_3X_4
那么，[-x]补 = (15- X_1) (15- X_2) (15- X_3) (15- X_4) + 1
要使得 x 和 [-x]补的每个 16 进制位相同，首先必须满足：
15- X_2 = 15- X_3 = 15- X_4 = F
这样才能使得最高位 15- X_1 加 1。因为如果最高位 15- X_1 不加 1，要使得 X_1 = 15- X_1，必须 X_1=7.5，显然这是不对的。因此必须满足 15- X_2 = 15- X_3 = 15- X_4 = F，
这样 [-x]补 = (15- X_1+1) FFF。于是，x = [-x]补的条件是 x = 0x8000 或者 x=0x0000。

7、有符号数与无符号数

设有 short x = 0xf100; short y = 0x1234; 问 x > y 是否成立？说明理由。

设有 unsigned short u = 0xf100; unsigned short v = 0x1234; 问 u > v 是否成立？说明理由。

x > y 是不成立，因为 x 是负数（将 0xf100 解释为有符号数时表示负数），y 是正数。u > v 是成立，因为 0xf100 被解释为无符号数。

8、数据类型转换

设有 int x; float y; y = (float)x;

问 x == (int)y 是否（一定）成立，为什么？

若有 x =(int) y; y ==(float) x； 是否（一定）成立，为什么？

x == (int)y 不一定成立，因为 y = (float)x 可能会产生精度损失，(int) y 也可能会产生精度损失。

x =(int) y; y ==(float) x 不一定成立，理由同上（float 转 int、int 转 float 都有可能产生损失）。

9、字符串的表示

设有 char s[] = "…"; 在内存中观察数组 s 中存放的信息为：

31 32 33 67 6f 6f 64 00 (每个字节都是 16 进制数，31 对应的字节地址最小)。

问 char s[] = "…"，引用中的字符串是什么？

“123good”

10、浮点数的表示

给出 11.25 的单精度浮点表示（要分别给出符号位、指数部分、有效数部分的编码），以及该数在内存中的存放形式。

11.25 D = 1011.01 B = 1.01101 * 2^3
阶码 = 3 + 127 = 130 D = 82 H
内存表示： 0 1000 0010 01101 00 0000 0000 0000 0000 B

11、为什么 float 数有+0 和-0？如何判断一个 float 变量的值是+0 还是-0？

IEEE754 表示 float 数时，由于精度问题，不能精确表示 0。所以 IEEE754 将无法表示的很小负数记为-0，将无法表示的很小正数记为+0。

12、假设：

float a = 65536; //0x10000
float b;
求满足 b>a 的条件下, IEEE754 能表示的最小 b。

a = 65536 = 0x10000 = 1 0000 0000 0000 0000 B = 1.0 * 2 ^ 16
阶码 = 16 + 127 = 143 D = 8F H
内存表示： 0 1000 1111 0000 0000 0000 0000 0000 000 B
大于 a 的最小数 b 的内存表示为： 0 1000 1111 0000 0000 0000 0000 0000 001 B
即 b = 1.00…001 * 2 ^ 16 (1.00…001, 小数点后面 22 个 0)