X86 部分

一、填空题（18 分）

1） 已知某 32 位整数 X，其值为-102，则其 16 进制补码为 0xFFFFFF9A （2 分），另一 32 位整数 Y 的补码为 0xFFFFFF68H，则 X+Y 的 16 进制补码(32 位)为 0xFFFFFF02，X-Y 的 16 进制补码为 0x32。

2） X86 32 位 linux 系统下的 float 类型的数据对齐要求是 4 字节对齐，double 类型的是 4 字节对齐；X86 32 位 Windows 系统下的 double 类型数据是 8 字节对齐。

3） 假设存在一种 16 位的浮点数表示，exp 位数是 7，frac 位数是 8，符号位为 1，其所能表示的绝对值最小的规格化数的 exp 是 0000001，frac 是 全 0；251 的 exp 是 1000110，frac 是 11111110。

4） 在 X86-32 位体系结构中，C 语言过程调用的默认传参规则是将过程参数从 右 至 左 压入栈，过程返回值（32 位）通过 eax 寄存器传出。

5） 给出 13/8 这一数字的 32 位浮点数（符合 IEEE 754 标准）表示，即 exp= 01111111；frac= 1010…0。

二、判断题（10 分）1 分 1 个

已知 int x = …; int y = …; float f = …; double d = …; unsigned int ux = x; unsigned int uy = y;

判断下面的等式（或不等式或推导）是否成立。

（1）x == (int)(float) x 错
（2）x == (int)(double) x 对
（3）f == (float)(double) f 对
（4）d == (float) d 错
（5）d > f -> -f > -d 对
（6）(d+f)-d == f 错
（7）(x>y) == (-x<-y) 错
（8）(x|-x)>>31 == -1 错
（9）~x+~y == ~(x+y) 错
（10）(int)(ux-uy) == -(y-x) 对

三、简答题 （54 分）

1）（4 分）已知一个 c 语言结构类型的定义如下，请问在 X86 32 位 Linux 系统下变量 p 所占的空间大小是多少，对齐的要求如何？（2 分）+（2 分）

24 字节；4 对齐。

```
struct   S1 {
   char c;
   int i[2];
   my_struct v;
} p;
```

```
typedef   TagStruct {
   int k[2];
   char c2;
} my_struct;
```

2)(8 分)下图给出了一个结构数组 a[10]，同时给出了一个用于访问其中某元素的 C 函数(**get_j**)

及其编译出来的汇编语言的关键部分。

```
struct S6 {
   short i;
   float v;
   short j;
} a[10];

short get_j(int idx)
{
   return a[idx].j;
}
```

```
# %eax = idx
leal (%eax,%eax,2),%eax # 3*idx
movswl a_start_addr+8(,%eax,4),%eax
```

（这儿用 *X*_start_addr 来表示 *X* 数组的首地址，movswl 就是将内存中的一个 16 位数带符号扩展后，传送至目的寄存器）。请仿效上述汇编语言实例，给出访问另外一个结构数组 b[10]中的元素的二个 C 函数的汇编关键代码。（X86-32 位代码，long long 类型是 64 位宽，8 字节对齐；函数返回值为 64 位数据时，高 32 位置于 edx。）

```
struct B{
   char c;
   int j[2];
   long long v;
} b[10];
```

```
long long  get_v (int i)
{
    return b[i].v;
} （4分）
int* get_j (int i)
{
    return b[i].j;
} （4 分）
```

1）X86 32 位体系结构中的条件跳转指令 jg 是用于符号数比较还是无符号数比较的？其产生跳转的成立条件是~(SF^OF)&~ZF 为真，请解释为何是这一条件。（4 分）

符号数比较

SF：结果正负，SF=1,表示运算结果为负数

OF：溢出标志位，OF=1，表示溢出

ZF：结果是否为零，ZF=1，表示结果为 0

即当 SF=OF 且 ZF=0 时，进行跳转

如果前后两数都为正，不发生溢出，则 SF=0，OF=0

如果前后两数都为负，不发生溢出，则 SF=0，OF=0

如果前正后负，则可发生溢出，不发生溢出则为 SF=0，OF=0；否则为 SF=1，OF=1

所以为 SF 和 OF 需要同或，最后需要保证前后不相等，ZF=0

2）下图给出了一个 C 函数，并由 gcc 编译成相应的汇编代码（AT&T 语法格式），请补全这段代码里头被省去的部分。（32 位 X86 代码，10 分）

```
int arith(int x, int y, int z)
{
    int t1= x+y;
    int t2 =z+t1;
    int t3=x+4;
    int t4=y*48;
    int t5=t3+t4;
    int rval=t2*t5;
    return rval;
}
```

编译出的代码：

```
…
movl    8(%ebp),%eax
movl    12 (%ebp),%edx
leal    (% edx , %eax),%ecx
leal    (%edx,%edx,2),%edx
sall    $4 ,%edx
addl    16 (%ebp), %ecx
leal    4(%edx,%eax),%eax
imull   %ecx,%eax
…
```

3）C语言中过程的参数个数可以是不固定的。比如定义了如下能够产生格式化输出的过程：

```
void my_printf(const char *fmt, ...);
```

其参数个数大于等于 1,第一个参数是一个格式字符串,可以接受形如“input string %d %d”之类的字符串作为输入，其中%d 指定输出 32 位带符号整数，具体的输出整数值则由后续的参数指定（为简化起见，这个函数只能接受%d 作为格式转换）。这个函数的汇编代码如下所示，请分析这些代码并回答如下问题：（11 分）

- 这类不定参数的过程是如何传入参数的？

  通过栈传递，从右至左压栈；8(%ebp)是字符串地址，12(%ebp)为变参的起始地址

- my_printf是如何确定不定参数个数的？

  从字符串起始地址开始依次扫描每个字符，统计出现的%d个数。

```
    .section .rdata,"dr"
LC0:
    .ascii "%d\0"
    .text
.globl _my_printf
_my_printf:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp

    pushl   %esi
    leal    12(%ebp), %esi
    pushl   %ebx
    movl    8(%ebp), %ebx
    movzbl  (%ebx), %eax
    testb   %al, %al
    je      L11
L14:
    cmpb    $37, %al #'%'的ascii码值是37
    je     L5
    movsbl  %al,%eax
L8:
    movl    %eax, (%esp)
    incl       %ebx
    call      _putchar
```

```
L15:
    movzbl  (%ebx), %eax
    testb   %al, %al
    jne L14
L11:
    addl    $16, %esp
    popl    %ebx
    popl    %esi
    popl    %ebp
    ret
L5:
    incl       %ebx
    movsbl  (%ebx),%eax
    cmpl    $100, %eax
    jne L8
    movl    %esi, %eax
    incl       %ebx
    movl    (%eax), %eax
    addl    $4, %esi
    movl    $LC0, (%esp)
    movl    %eax, 4(%esp)
    call    _printf
    jmp     L15
```

6）（8 分）左侧的汇编代码段 foo1（2 分，对 c3）、foo2（3 分，对 c5）、foo3（3 分，对 c1）分别对应右侧的哪段 C 代码？

```
foo1:
      pushl %ebp
      movl %esp,%ebp
      movl 8(%ebp),%eax
      sall $4,%eax
      subl 8(%ebp),%eax
      movl %ebp,%esp
      popl %ebp
      ret

foo2:
      pushl %ebp
      movl %esp,%ebp
      movl 8(%ebp),%eax
      testl %eax,%eax
      jge .L4
      addl $15,%eax
.L4:
      sarl $4,%eax
      movl %ebp,%esp
      popl %ebp
      ret

foo3:
      pushl %ebp
      movl %esp,%ebp
      movl 8(%ebp),%eax
      shrl $31,%eax
      movl %ebp,%esp
      popl %ebp
      ret
```

```
int choice1(int x)
{
    return (x < 0);
}

int choice2(int x)
{
    return (x << 31) & 1;
}

int choice3(int x)
{
    return 15 * x;
}

int choice4(int x)
{
    return (x + 15) /4
}

int choice5(int x)
{
    return x / 16;
}

int choice6(int x)
{
    return (x >> 31);
}
```

MIPS32 部分(四)

简答题（18 分）

1） 请分别举出一个同步异常（2 分）、异步异常（2 分）的例子；以及举出两种情况（共 2 分），在这两种情况下异常返回不是回到 EPC 记录的地址（6 分）

这个很多答案了！

2）补全下列代码：（8 分）（1 个 1 分）

（2.1）lw $t6, 65536($sp)经过MIPS汇编器处理后，产生的代码如下，请补全。

lui $1, 1

addu $1, $1, $sp

lw $t6, 0($1)

（2.2）li $6, 0x245678 经过MIPS汇编器处理后，产生的代码如下，请补全

lui $1, 0x24

ori $6, $1, 0x5678

（2.3）addu $4, 0x10000经过MIPS汇编器处理后，产生的代码如下，请补全

lui $at, 1

addu $4, $4, $at

3）（4分）mips 32中异常处理过程返回用的指令是eret，其功能是返回到EPC寄存器所

存储的地址，而且置status寄存器中的相关位为0（表示运行状态回到用户态），请解释下如果这两个功能不是用一条指令实现，会出现什么问题？