考试中心填写：

___年 ___月___日
考试用

湖南大学课程考试试卷

课程名称：计算机系统　；试卷编号：A；考试时间：120 分钟

姓名：　学号：　专业班级：

题号	一	二	三	四	五	六	七	八	九	十	总分
应得分	10	10	15	20	20	25					100
实得分											
评卷人											评分：

一.简答题（10 分）

小明仿照 IEEE 754 标准，对其进行了微调，形成了 HNU 2023 标准。

微调的地方为：IEEE 754 标准中，对于 k 位阶码，其偏移值为 $2^{k-1}-1$，而 HNU 2023 标准中，规定偏移值为 2^k-1。

以 10bit 数字（其中 1bit 为符号位，4bit 为阶码位，5bit 为尾数位）为例，仅讨论**正数**的情况。

请回答：

（1）HNU 2023 标准与 IEEE754 标准相比，其表示的数字范围有何区别？

（2）对于 HNU 2023 标准与 IEEE754 标准，其不能表示的最小整数分别是多少？请写出其十进制值与二进制值表示。

二.程序填空题（10 分，每空 2 分）

如下是一个 c 语言程序及其对应的汇编代码（32 位机，小端环境下编译），请参照汇编代码，完成 c 程序的空缺部分。

c 语言程序：

```
int main()
{
    int i,j,flag;
    int sum=   (1)
    for(   (2)    ;   (3)    ;i++,j+=2)
    {
        flag=   (4)    ;
        sum+=   (5)    ;
    }
    return sum;
}
```

汇编代码如下：

```
main:
pushl  %ebp
```

```
    movl   %esp, %ebp
    subl   $16, %esp
    movl   $17, -8(%ebp)
    movl   $5, -16(%ebp)
    movl   $1, -12(%ebp)
    jmp .L2
.L5:
    movl   -16(%ebp), %eax
    andl   $1, %eax
    testl  %eax, %eax
    jne .L3
    movl   -12(%ebp), %eax
    addl   $3, %eax
    jmp .L4
.L3:
    movl   -16(%ebp), %eax
    subl   $2, %eax
.L4:
    movl   %eax, -4(%ebp)
    subl   $1, -16(%ebp)
    movl   -12(%ebp), %eax
    imull  -16(%ebp), %eax
    addl   -4(%ebp), %eax
    addl   %eax, -8(%ebp)
    addl   $1, -12(%ebp)
    addl   $1, -16(%ebp)
    addl   $2, -12(%ebp)
.L2:
    movl   -12(%ebp), %eax
    movl   -16(%ebp), %edx
    addl   %edx, %eax
    cmpl   $98, %eax
    jle .L5
    movl   -8(%ebp), %eax
    leave
    ret
```

三.程序填空题（第一空 3 分，后面三空每空 4 分，共 15 分）

如下是一个 c 语言程序及其对应的汇编代码(32 位机,小端环境下编译),请参照汇编代码,完成 c 程序的空缺部分。

c 语言程序:

```
int f(int a,int b)
{
    if(0==a) return ____(1)____;
```

```
    return    (2)    ;
}
int g(int x)
{
    if(x<=0) return 0;
    if(x==1||x==2) return 1;
    return    (3)    ;
}
int main()
{
    int x=3,y=5;
    return f(    (4)    );
    return 0;
}
```

汇编代码如下：

```
f:
    pushl  %ebp
    movl   %esp, %ebp
    subl   $24, %esp
    cmpl   $0, 8(%ebp)
    jne .L2
    movl   12(%ebp), %eax
    imull  12(%ebp), %eax
    jmp .L3
.L2:
    movl   12(%ebp), %eax
    movl   8(%ebp), %edx
    addl   %edx, %eax
    movl   %eax, (%esp)
    call   g
.L3:
    leave
    ret
g:
    pushl  %ebp
    movl   %esp, %ebp
    pushl  %ebx
    subl   $20, %esp
    cmpl   $0, 8(%ebp)
    jg  .L5
    movl   $0, %eax
    jmp .L6
.L5:
    cmpl   $1, 8(%ebp)
```

```
    je  .L7
    cmpl   $2, 8(%ebp)
    jne .L8
.L7:
    movl   $1, %eax
    jmp .L6
.L8:
    movl   8(%ebp), %eax
    subl   $2, %eax
    movl   %eax, (%esp)
    call   g
    leal   (%eax,%eax), %ebx
    movl   8(%ebp), %eax
    addl   $3, %eax
    movl   %eax, (%esp)
    call   g
    movl   %eax, %edx
    movl   %edx, %eax
    addl   %eax, %eax
    addl   %edx, %eax
    addl   %ebx, %eax
.L6:
    addl   $20, %esp
    popl   %ebx
    popl   %ebp
    ret
main:
    pushl  %ebp
    movl   %esp, %ebp
    pushl  %ebx
    andl   $-16, %esp
    subl   $32, %esp
    movl   $3, 24(%esp)
    movl   $5, 28(%esp)
    movl   28(%esp), %eax
    movl   24(%esp), %edx
    movl   %edx, %ecx
    subl   %eax, %ecx
    movl   %ecx, %eax
    movl   %eax, (%esp)
    call   g
    movl   %eax, %ebx
    movl   28(%esp), %eax
    movl   %eax, 4(%esp)
```

```
    movl   24(%esp), %eax
    movl   %eax, (%esp)
    call   f
    imull  %ebx, %eax
    movl   28(%esp), %edx
    movl   24(%esp), %ecx
    addl   %ecx, %edx
    movl   %eax, 4(%esp)
    movl   %edx, (%esp)
    call   f
    movl   -4(%ebp), %ebx
    leave
    ret
```

四.综合应用题（20 分）

一个函数调用的 C 代码如下：

```
#include "stdio.h"

void test(int *xp,int *yp,int i,int j)
{
    int size=sizeof(int), step=(i*(3*size)+j*size)/4;

    int *addr1=(int*)(xp+step);
    int *addr2=(int*)(yp+step);

    int m=*addr1, n=*addr2;

    *addr1=n;
    *addr2=m;
}
int main()
{
    int A[3][3]={{1,4,7},{2,5,8},{3,6,9}};
    int B[3][3]={{10,20,30},{40,50,60},{70,80,90}};
    int *PA=*A;
    int *PB=*B;

    test(PA,PB,1,2);

    printf("A[1][2]=%d\n B[1][2]=%d\n", A[1][2],B[1][2]);
    return 0;
}
```

1．该程序运行后，打印在屏幕上的结果是_____(5 分)；

2．将返回地址“Rtn Addr”，填入栈帧图中准确位置（3 分）；

3．在主函数栈帧中的正确位置，填写 4 个传递参数（12 分）；
*说明：当前 ESP 与 EBP 指向子函数栈帧，图中每一格为 4 字节。

地址	内容
0XBFFFF108	OLD EBP
……	……
0FC	B[2][2]=90
……	……
0DC	B[0][0]=10
0D8	A[2][2]=9
……	……
0B8	A[0][0]=1
……	
0AC	
0A8	
0A4	
0A0	
09C	
(EBP) → 098	OLD EBP
……	
(ESP) → 078	

五.综合应用题（20 分）
给定一个 32 位 Linux 系统，其高速缓存的大小为 64 字节，2 路组相联，每个块 16 字节。高速缓存采用的写策略是直写，替换策略为 LRU(最近最少使用)。
A)（2 分）基础知识
A1) 高速缓存中有多少个组_____?
A2) 每个组中有多少缓存行_____?

B）(10 分)假设如下：
- 访问一次内存消耗 100 ns
- 访问一次 cache 的开销是 1 ns
- 忽略可能发生的其他时间开销(回收、存储到高速缓存等)
- 如果使用高速缓存，我们总是会先访问高速缓存(因此，在这种情况下，高速缓存未命中的开销为 101 ns)

给定一个整数数组：
```
int Arr[6][4];
```
Arr 数组从地址 0x00000000 开始。
如果我们要逐一访问数组中的以下元素：
B1) 用"H"或"M"填空，分别表示高速缓存命中和高速缓存未命中。

Access	Cache Result
Arr[0][0]	
Arr[0][2]	
Arr[0][3]	
Arr[1][1]	
Arr[1][3]	

Arr[2][1]	

B2）如果不使用高速缓存，时间开销是多少 _____ ？
B3）如果使用高速缓存，时间开销是多少 _____？

C)（8 分）如果我们使用以下程序访问数组：

```
int i, j;
for ( i = 0; i < 4; i++) {
    for ( j = 0; j < 6; j++) {
    Arr[j][i]++;
    }
}
```

C1)如果不使用高速缓存，时间开销是多少_____？
C2)如果使用高速缓存，时间开销是多少 _____？

六.综合分析题（25 分）
现有如下 C 代码片断：

……

```
extern int number=2
int sharedV（）{ return number };

int main ()
{
  int i,child_status;
  pid_t pid[sharedV];

  void myHandler(int sig) {
     printf("Process received signal\n");
exit(0); }


  signal(SIGINT, myHandler);

  for(i=0;i< sharedV(); i++) {
    if ((pid[i]=fork())==0) { while(1);} }

  for (i=0; i<sharedV(); i++) {
    kill(pid[i],SIGINT);}

  for (i=0;i<sharedV(); i++) {

  pid_t wpid=wait(&child_status);
```

```
  if (WIFEXITED(child_status))
    printf("Process                                                    %d:
Hello,World!%d\n",wpid,WEXITSTATUS(child_status));}
```

请阅读分析上述 C 代码后，尝试解答下列问题：

（1）这段代码的输出结果可能是什么？为什么会有这样的输出？

（2）在这段代码中 wait 函数的作用是什么？如果改用 waitpid 函数可能会产生什么影响？

（3）当观察该代码对应的可重定位目标文件时，发现如下信息：

```
fc: e8 fc ff ff ff call fd<main+0xda>
                        fd: R_386_PC32 kill
……
108：a1 00 00 00 00   mov 0x0,%eaX
                        109: R386_32    sharedV
……
```

请问这两行是什么意思？作用是什么？在可执行文件里面这两行会产生什么变化？

（4）对可执行文件进行反汇编观察时，发现在<main>部分对应 printf 函数调用的地方出现了如下汇编代码：

```
call 80483e0 <printf@plt>
```

继而查看 0X80483e0 地址处的内容，发现出现如下语句：

```
jmp  *0x804a00c
push  $0x0
jmp 80483d0 <_init+0x24>
```

请结合你学习的 PIC 的相关知识尝试解答这几条语句是干什么用的，以及后续对 printf 的重定位过程。

（5）对于题干中的代码，不考虑数据结构和算法的变更，做什么修订可能能改善该代码的执行性能？