考试科目名称　计算机系统基础　（A 卷）

2014—2015 学年第 1 学期　教师 袁春风 路通 苏丰 唐杰 汪亮　考试方式：开卷

系（专业）　计算机科学与技术　　年级　2013　　班级＿＿＿＿

学号＿＿＿＿＿＿＿＿　　姓名＿＿＿＿＿＿　　成绩＿＿＿＿＿

题号	一	二	三	四	五	六	七	八	九	十	十一	十二	
分数													

一个 C 语言程序有两个源文件：main.c 和 test.c，它们的内容如下图所示。

```
/* main.c */
1    #include <stdio.h>
2
3     int sum();
4     int a[4]={-1,-100,2, 3};
5     extern int val;
6     void main( )
7     {
8           val=sum();
9           printf("sum=%d\n",val);
10    }
```

```
/* test.c */
1    extern int a[];
2
3     int val=0;
4     int sum()
5     {
6          int i;
7          for (i=0; i<4; i++)
8             val += a[i];
9          return val;
10    }
```

假设在 IA-32/Linux 平台上用 GCC 编译驱动程序处理，main.c 和 test.c 的可重定位目标文件名分别是 main.o 和 test.o，生成的可执行文件名为 test。回答下列问题或完成下列任务。

（提示：IA-32 为小端方式，字长为 32 位，即 sizeof(int)=4，虚拟地址空间中的只读数据和代码段、可读写数据段都按 4KB 边界对齐）

一、从 C 语言源程序到可执行文件 test 的转换需要经过哪些步骤？**（4 分）**

二、已知数组 a 首址为 0x080496dc，则 0x080496e0 到 0x080496e3 每个单元的内容依次是什么？假设数组 a 的类型为 float，则 0x080496e0 到 0x080496e3 每个单元的内容依次是什么？**（6 分）**

三、使用‘objdump –d test’得到 sum 函数的反汇编结果如下，从反汇编结果可看出 IA-32 是 CISC 还是 RISC？为什么？**（2 分）**

四、根据 sum 函数反汇编结果画出其栈帧，要求分别用 EBP 和 ESP 标示栈帧底部和顶部并标出 i 的位置。**（4 分）**

五、cmpl 指令的执行将会影响 EFLAGS 寄存器中哪些常用标志？当 i=4 时，sum 函数中 cmpl 指令的执行结果将如何影响下条 jle 指令？**（10 分）**

```
08048448 <sum>:
 8048448:       55                      push   %ebp
 8048449:       89 e5                   mov    %esp,%ebp
 804844b:       83 ec 10                sub    $0x10,%esp
 804844e:       c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%ebp)
 8048455:       eb 1a                   jmp    8048471 <sum+0x29>
 8048457:       8b 45 fc                mov    -0x4(%ebp),%eax
 804845a:       8b 14 85 dc 96 04 08    mov    0x80496dc(,%eax,4),%edx
 8048461:       a1 f0 96 04 08          mov    0x80496f0,%eax
 8048466:       01 d0                   add    %edx,%eax
 8048468:       a3 f0 96 04 08          mov    %eax,0x80496f0
 804846d:       83 45 fc 01             addl   $0x1,-0x4(%ebp)
 8048471:       83 7d fc 03             cmpl   $0x3,-0x4(%ebp)
 8048475:       7e e0                   jle    8048457 <sum+0xf>
 8048477:       a1 f0 96 04 08          mov    0x80496f0,%eax
 804847c:       c9                      leave
 804847d:       c3                      ret
```

六、地址 0x804845a 处的 mov 指令中，源操作数采用什么寻址方式？其中，EAX 寄存器存放的是哪个变量？为何比例因子为 4？如何计算源操作数的有效地址？源操作数的访问过程需要经过哪些步骤？（要求从有效地址计算开始进行简要说明，包括何时判断及如何判断 TLB 缺失、缺页和 cache 缺失等，字数在 300~400 字左右）**（20 分）**

七、画出 test 的一个进程对应的虚拟地址空间。要求根据 sum 函数的反汇编结果，给出只读数据和代码段的起始地址、可读写数据段的起始地址，并说明符号 a、val、sum 分别定义在哪个段内。**（10 分）**

八、使用“objdump –d test”得到 main 函数的反汇编结果如下。已知分页时页大小为 4KB，主存与 cache 交换的主存块大小为 64B，则在执行 test 程序过程中，对于 sum 函数来说，访问指令的 cache 命中率是多少？**（10 分）**

```
0804841c <main>:
 804841c:       55                      push   %ebp
 804841d:       89 e5                   mov    %esp,%ebp
 804841f:       83 e4 f0                and    $0xfffffff0,%esp
 8048422:       83 ec 10                sub    $0x10,%esp
 8048425:       e8 1e 00 00 00          call   8048448 <sum>
 804842a:       a3 f0 96 04 08          mov    %eax,0x80496f0
 804842f:       a1 f0 96 04 08          mov    0x80496f0,%eax
 8048434:       89 44 24 04             mov    %eax,0x4(%esp)
 8048438:       c7 04 24 10 85 04 08    movl   $0x8048510,(%esp)
 804843f:       e8 bc fe ff ff          call   8048300 <printf@plt>
 8048444:       c9                      leave
 8048445:       c3                      ret
```

九、填写下表中各符号的情况，说明每个符号是否出现在 test.o 的符号表（.symtab 节）中，如果是的话，进一步说明定义该符号的模块是 main.o 还是 test.o、该符号的类型是全局、外部还是本地符号、该符号出现在 test.o 中的哪个节（.text、.data 或.bss）。**（6 分）**

符号	在 test.o 的符号表中?	定义模块	符号类型	节
a				
val				
sum				
i				

十、使用“objdump –d test.o”得到 sum 函数的反汇编结果如下。对照在可重定位文件 test.o 和可执行文件 test 中的两个 sum 函数的反汇编结果，说明在哪些指令中进行了重定位（可在相应指令下方划线或给出相应指令所在的位移量）。**（4 分）**

```
test.o:     file format elf32-i386
Disassembly of section .text:
00000000 <sum>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 ec 10                sub    $0x10,%esp
   6:   c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%ebp)
   d:   eb 1a                   jmp    29 <sum+0x29>
   f:   8b 45 fc                mov    -0x4(%ebp),%eax
  12:   8b 14 85 00 00 00 00    mov    0x0(,%eax,4),%edx
  19:   a1 00 00 00 00          mov    0x0,%eax
  1e:   01 d0                   add    %edx,%eax
  20:   a3 00 00 00 00          mov    %eax,0x0
  25:   83 45 fc 01             addl   $0x1,-0x4(%ebp)
  29:   83 7d fc 03             cmpl   $0x3,-0x4(%ebp)
  2d:   7e e0                   jle    f <sum+0xf>
  2f:   a1 00 00 00 00          mov    0x0,%eax
  34:   c9                      leave
  35:   c3                      ret
```

十一、为什么在 main.c 的开头需加“#include <stdio.h>”？ 为什么 main.c 中没有定义 printf()函数，也没它的原型声明，但 main()函数引用它时没有发生错误？为什么 printf()函数中未指定字符串输出目标，但执行 test 程序后会在屏幕上显示字符串？**（4 分）**

十二、main 函数中的 printf 语句所对应的指令为“call 8048300<printf@plt>”。简述从执行该指令开始到在屏幕上显示出“sum=-96”为止的大概过程。要求字数在 300 左右。**（10 分）**

十三、简述计算机系统层次结构。要求字数在 200 左右。**（10 分）**