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	十三
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      #define N 4
3      int val=0;
4      int sum()
5      {
6           int i;
7           for (i=0; i<N; 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？简要说明每个步骤的工作内容。**（4 分）**

参考答案：　（略）

二、已知数组 a 在虚拟空间中首址为 0x080496dc，则 0x080496e0 到 0x080496e3 这 4 个单元内容依次是什么？若 a 改为 float 型（即 main.c 第 4 行的 int 改为 float），则这 4 个单元内容依次是什么？ **（6 分）**

参考答案：

在 0x080496e0 到 0x080496e3 这 4 个单元中存放的是-100，-100=-(64+32+4)=-1100100B，其对应的 8 位补码表示为 10011100，在机器中的 32 位补码表示为 FFFFFF9CH。因为 IA-32 是小端方式，因此，在 0x080496e0 到 0x080496e3 这 4 个单元的内容依次为：9CH、FFH、FFH、FFH。（2 分）

若 a 改为 float 类型，则-100 用 IEEE 754 单精度格式表示，因为-1100100B=-1.1001B×2^6，因此其机器数为 1 10000101 100 1000 0000 0000 0000 0000，用十六进制表示为 C2C80000H，因此，在 0x080496e0 到 0x080496e3 这 4 个单元的内容依次为：00H、00H、C8H、C2H。（4 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）**

参考答案：CISC。（1 分）

因为指令的长度参差不齐，有一个、两个、三个、四个字节等不同长度。（1 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
```

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）

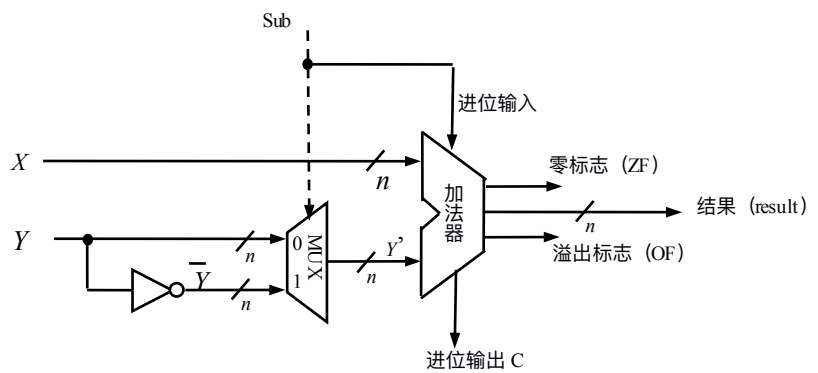
参考答案：

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）

参考答案：

cmpl 指令通过做减法来生成标志信息，其执行将影响 EFLAGS 寄存器中的 OF、CF、ZF 和 SF 这几位条件标志位。（4 分）

当 i 为 4 时，cmpl 指令中的操作数 M[R[ebp]-4]实际上就是 4，因此，cmpl 指令实际上是在以下电路中实现“4 减 3”的功能，也即，在电路中 X=0000 0004H，Y=0000 0003H，Y’=FFFF FFFCH，Sub=1，因此，result=0000 0001H，OF=0，CF=0，ZF=0，SF=0，执行到 jle 指令时，根据指令功能得知是带符号整数比较，因为 OF=SF，因而是大于关系，不满足 jle 指令的“小于或等于”关系，因而不会跳转到循环体内执行，而是跳出循环执行。（6 分）



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，在 300 字以上。）**（20 分，若能结合题目中给出的具体例子清楚描述 IA-32/Linux 中的地址转换过程，则额外加 10 分）**

参考答案：

采用“比例变址+位移量”寻址方式。（1 分）

EAX 中存放的是 i。（1 分）

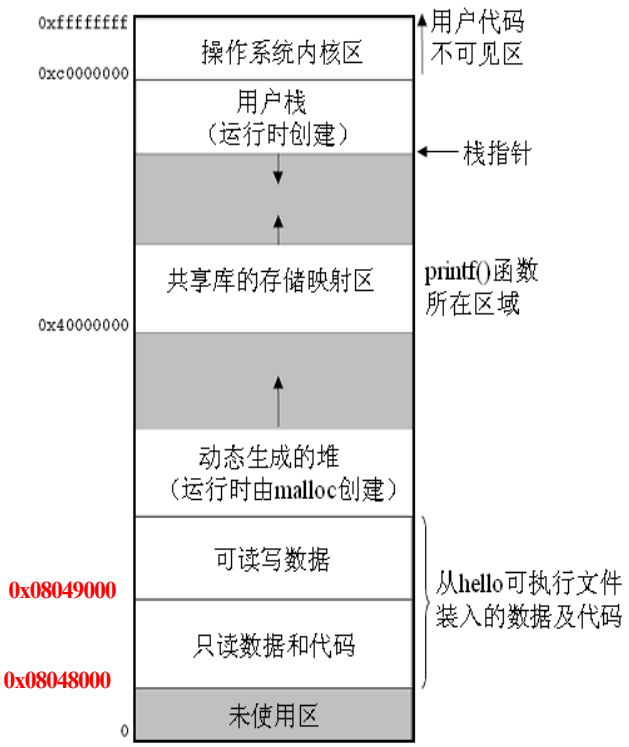
比例因子为 4 是因为 a 的每个元素占 4B，且 IA-32 按字节编址。（1 分）

有效地址 EA=R[eax]*4+0x80496dc=i*4+0x80496dc（其中 0x80496dc 是数组 a 的首地址）（1 分）

（略）

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）**

参考答案：



根据 sum 反汇编结果可知只读数据和代码段的起始地址、可读写数据段的起始地址分别是 0x8048000 和 0x8049000。（7 分）

因为符号 a 和 val 都是初始化的全局变量，所以被定义在可读写数据段中，sum 是函数名，所以被定义在只读数据和代码段。（3 分）

八、已知页大小为 4KB，主存地址位数为 32 位。假设数据 cache 的数据区大小为 32KB，采用 4 路组相联映射方式，主存块大小为 64B，则主存地址如何划分？若 main.c 中数组 a 有 100 个元素被初始化，test.c 中的 N 定义为 100，即 for 循环访问的数组元素为 a[0]~a[99]，执行 sum 函数前，数组 a 未调入 cache

但已调入主存，数组 a 在主存的首地址为 0x406dc，则数组 a 的所有元素将被复制到 cache 的哪些组中？在执行 sum 函数过程中访问数组 a 的 cache 命中率是多少？ **（10 分）**

参考答案：

数据 cache 共有 32KB/64B=512 行，共有 512/4=128 个 cache 组。因此，32 位主存地址中，块内地址占 6 位，组索引占 7 位，高 19 位是标记。（3 分）

数组 a 中有 100 个元素，每个元素占 4 个字节，因此，a 共占 400B，400B/64B=6.25。其首地址为 0x406dc，因此，其最后一个元素的地址为 0x406dc+396=0x406dc+0x18c=0x40868。（3 分）

将 0x406dc 展开后为 100 000 0011011 011100，将 0x40868 展开为 100 000 0100001 101000，因此，可以看出 a 中的元素依次映射到 cache 的第 11011B=27 组、28 组、…、100001=33 组，共 7 组。因此，共有 7 次未命中，命中率为 93/100=93%。（4 分）

九、填写下表中各标识符的情况，说明每个标识符是否出现在 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）**

参考答案：

标识符	在 test.o 的符号表中?	定义模块	符号类型	节
a	是	main.o	外部	---
val	是	test.o	全局	.data
sum	是	test.o	全局	.text
i	否	---	---	---
N	否	---	---	---

（i 和 N 各占 1 分，其余每 3 格占 1 分，共 6 分）

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）**

参考答案：（一个占 1 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( )函数中未指定输出文件名，但执行 test 程序后会在屏幕上显示字符串？ **（4 分）**

参考答案：

因为 main.c 中使用了标准 I/O 函数 printf()，这个函数的原型说明在 stdio.h 中，所以应该在 main.c 的开头加“#include <stdio.h>”。（1 分）

因为在 main.c 的开头加了“#include <stdio.h>”，因此，在预处理后，printf()函数的原型说明被嵌入到了 main.c 中，因此，在 main 函数中引用 printf 函数不会发生错误。（2 分）

因为 printf 函数隐含的输出文件名是 stdout，即标准输出，因此，执行 test 程序后会在屏幕上显示字符串。（1 分）

十二、main 函数中的 printf 语句所对应的指令为“call　8048300”。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 300 字以上。**（10 分，若能结合题目中给出的具体例子清楚描述 IA-32/Linux 中的异常/中断处理机制，则额外加 10 分）**

参考答案：

（略）

十三、已知 test 程序用于解决求和问题。请运用计算机系统层次结构概念，简要说明如何使用通用计算机系统解决求和问题。与使用普通计算器进行求和的方法比较，使用通用计算机系统解决问题的最大特点是计算机系统采用什么工作方式？ **（10 分）**

参考答案：

（略）