考试科目名称 计算机系统基础 （A 卷）

2021 —2022 学年第 1 学期 教师 唐杰 苏生 戴鹏鹏 朱光辉 汪亮 蒋岩岩 王慧妍 考试方式：开卷

系（专业）_____ 年级_____ 班级_____

学号_____ 姓名_____ 成绩_____

题号	一	二	三	四	五	六	七	八
分数								

一个 C 语言程序的源文件 m1.c 和 m2.c 的内容以及在 IA-32/Linux 平台上使用 GCC 编译源文件并对生成的可重定位文件进行反汇编的结果如图所示。请根据图示代码回答题一至题七。

一、 根据 C 源程序文件及其可重定位目标文件的反汇编结果，画出 main 函数调用 getbyte 函数时包含两个函数的栈帧的栈结构内容，并在其中标出每一成员的起始地址（相对于相应函数主体执行时的 EBP 值）及其对应的 C 变量名或表达式。（10 分）

二、 回答下列问题：

（1） IA-32 采用的是小端还是大端方式？从 m2.o 文件中哪条指令能看出？为什么？（3 分）

（2） 结构 Record 在内存中占多少字节的存储空间？画出该结构中各成员在内存中的布局（标出各成员的存储位置相对于结构起始存储位置的偏移量），并解释编译器如何确定结构 Record 及其中成员所占的字节数。（5 分）

（3） 基于变量 st_head 和 ST 的初始值，如对 C 表达式“st_head->data.w[1] + st_head->data.w[2]”进行求值，则其结果值采用补码表示为 32 位十六进制有符号整数是什么（需给出计算过程）？（3 分）

（4） 结构数组 ST 中，针对成员 key 的初始值等于 0xcd 的结构，假设将其字节数组 b 的初始字节序列值作为一个双精度浮点数 d 的采用 IEEE-754 标准在 IA-32/Linux 平台上的编码表示结果，则 C 表达式“(int) d”的值的十六进制补码表示是什么？需给出计算步骤说明。（4 分）

三、 回答下列问题：

（1） 假设执行模块 m1.o 中偏移量为 0x3f 处的 cmp 指令时，寄存器%eax 中的值是 0x804d040，栈地址-0x4(%ebp)起始的四个字节存储单元的内容依次是 0x20、0xd1、0x04、0x08，则 cmp 指令执行后，标志位 SF、CF 和 OF 的值分别是什么（需给出计算过程）？（4 分）

（2） 模块 m1.o 中偏移量 0x1e 处的 movzbl 指令的源操作数采用了什么寻址方式？该指令的功能是什么？对应程序中哪个 C 表达式（或其一部分）的实现？（4 分）

（3） 针对 getbyte 函数的机器指令，位于 m1.o 中偏移量 0x79 处的 jne 指令中标记为“??”的 1 字节整数操作数的十六进制补码表示是什么？解释得出该结果的依据。（4 分）

（4） m1.c 中 getbyte 函数中 C 语句“b = p->data.b[idx]”是由图中所示哪几条机器指令实现的？逐一解释每条机器指令的功能作用。（6 分）

四、 回答下列问题：

（1） 变量 st_head 和 getbyte 函数中变量 b 的运行时存储空间分别位于程序运行内存映像中的哪个段中？这两个变量的初始值分别保存于可执行程序文件的哪个节中？分别在程序装载与运行过程中的什么时间点完成初始化？（6 分）

（2） 程序链接后运行时 printf 函数的输出是什么？请给出具体解释。（5 分）

源程序文件 m1.c

```
struct Record {
  char key;
  union {
    char b[8];
    short w[4];
  } data;
  struct Record *next;
};

struct Record ST[256] = {
  {0x12, {0xfe, 0xdc, 0xba, 0x98,
0x76, 0x54, 0x32, 0x10}, &ST[1]},
  {0xcd, {0x00, 0x00, 0x00, 0x64,
0x22, 0xac, 0xc2, 0x41}, &ST[2]},
  /* 其他初始值，各结构 key 值不同 */
  /* 下为数组最后一个结构的初始值 */
  {0xab, {0x01, 0x23, 0x45, 0x67,
0x89, 0xab, 0xcd, 0xef}, 0} };

struct Record *st_head = ST;

char getbyte( int key, int idx )
{
  struct Record *p, *pp;
  char b = 0xff;

  p = st_head; pp = 0;
  while ( p ) {
    if ( p->key == key ) {
      b = p->data.b[idx];
      if ( p != st_head ) {
        pp->next = p->next;
        p->next = st_head;
        st_head = p;
      }
      break;
    }
    pp = p; p = p->next;
  }

  return b;
}
```

可重定位目标文件 m1.o 反汇编结果

```
00000000 <getbyte>:
 0:   55                      push   %ebp
 1:   89 e5                   mov    %esp,%ebp
 3:   83 ec 10                sub    $0x10,%esp
 6:   c6 45 f7 ff             movb   $0xff,-0x9(%ebp)
 a:   a1 00 00 00 00          mov    0x0,%eax
 f:   89 45 fc                mov    %eax,-0x4(%ebp)
12:   c7 45 f8 00 00 00 00    movl   $0x0,-0x8(%ebp)
19:   eb 5a                   jmp    75 <getbyte+0x75>
1b:   8b 45 fc                mov    -0x4(%ebp),%eax
1e:   0f b6 00                movzbl (%eax),%eax
21:   0f be c0                movsbl %al,%eax
24:   39 45 08                cmp    %eax,0x8(%ebp)
27:   75 3d                   jne    66 <getbyte+0x66>
29:   8b 55 fc                mov    -0x4(%ebp),%edx
2c:   8b 45 0c                mov    0xc(%ebp),%eax
2f:   01 d0                   add    %edx,%eax
31:   83 c0 02                add    $0x2,%eax
34:   0f b6 00                movzbl (%eax),%eax
37:   88 45 f7                mov    %al,-0x9(%ebp)
3a:   a1 00 00 00 00          mov    0x0,%eax
3f:   39 45 fc                cmp    %eax,-0x4(%ebp)
42:   74 39                   je     7d <getbyte+0x7d>
44:   8b 45 fc                mov    -0x4(%ebp),%eax
47:   8b 50 0c                mov    0xc(%eax),%edx
4a:   8b 45 f8                mov    -0x8(%ebp),%eax
4d:   89 50 0c                mov    %edx,0xc(%eax)
50:   8b 15 00 00 00 00       mov    0x0,%edx
56:   8b 45 fc                mov    -0x4(%ebp),%eax
59:   89 50 0c                mov    %edx,0xc(%eax)
5c:   8b 45 fc                mov    -0x4(%ebp),%eax
5f:   a3 00 00 00 00          mov    %eax,0x0
64:   eb 17                   jmp    7d <getbyte+0x7d>
66:   8b 45 fc                mov    -0x4(%ebp),%eax
69:   89 45 f8                mov    %eax,-0x8(%ebp)
6c:   8b 45 fc                mov    -0x4(%ebp),%eax
6f:   8b 40 0c                mov    0xc(%eax),%eax
72:   89 45 fc                mov    %eax,-0x4(%ebp)
75:   83 7d fc 00             cmpl   $0x0,-0x4(%ebp)
79:   75 ??                   jne    ************
7b:   eb 01                   jmp    7e <getbyte+0x7e>
7d:   90                      nop
7e:   0f b6 45 f7             movzbl -0x9(%ebp),%eax
82:   c9                      leave
83:   c3                      ret
```

源程序文件 m2.c

```
#include <stdio.h>

char getbyte( int, int );

void main()
{
  char c = getbyte( 0xab, 0x2 );

  printf( "%d", c );
}
```

可重定位目标文件 m2.o 反汇编结果

```
00000000 <main>:
.....
 a:   55                      push   %ebp
 b:   89 e5                   mov    %esp,%ebp
 d:   51                      push   %ecx
 e:   83 ec 14                sub    $0x14,%esp
11:   83 ec 08                sub    $0x8,%esp
14:   6a 02                   push   $0x2
16:   68 ab 00 00 00          push   $0xab
1b:   e8 fc ff ff ff          call   1c <main+0x1c>
20:   83 c4 10                add    $0x10,%esp
23:   88 45 f7                mov    %al,-0x9(%ebp)
26:   0f be 45 f7             movsbl -0x9(%ebp),%eax
2a:   83 ec 08                sub    $0x8,%esp
2d:   50                      push   %eax
2e:   68 00 00 00 00          push   $0x0
33:   e8 fc ff ff ff          call   34 <main+0x34>
```

五、　对照两个 C 源程序模块及其可重定位目标模块的反汇编结果，列出两个可重定位目标模块的.text 节中需要重定位的所有引用的偏移量、目标符号和重定位类型。（7 分）

可重定位目标模块	引用偏移量	目标符号名	重定位类型

六、 已知 IA-32 页大小为 4KB（4096 字节），主存地址位数为 32 位。假设指令和数据有各自独立的 Cache，其中数据 Cache 的数据区大小为 4KB，采用 4 路组相联映射、LRU 替换算法和回写（write-back）与写分配（write allocate）策略，主存块大小为 64 字节。系统使用一个包含 128 个页表项、采用 8 路组相联映射方式的 TLB。在链接后生成的可执行程序中，数组 ST 的虚拟起始地址为 0x804c040。系统中没有其他用户进程在执行。

回答下列问题：

（1） 在 32 位虚拟地址中，哪几位表示虚拟页号？虚拟页号中哪几位表示 TLB 组索引、哪几位表示 TLB 标记？（3 分）

（2） 假设 getbyte 函数已经顺序访问了数组 ST 中的前 128 个结构元素，则在访问后续 128 个结构元素的过程中是否会发生缺页异常？为什么？（3 分）

（3） 数组 ST 中第一个和最后一个结构所在主存块分别映射到数据 Cache 的哪一组？请给出计算过程。（4 分）

（4） 假设程序执行到 getbyte 函数中偏移量为 0x6 的 movb 指令时，EBP 寄存器的值为 0xbffff5d8，则按照图中所给出的 main 函数调用 getbyte 函数时传递的实际参数值，getbyte 函数自执行偏移量 0x19 处的 jmp 指令到偏移量 0x7d 处的 nop 指令之间，对函数参数 key 和 idx 的访问是否会发生数据 Cache 缺失？请给出详细解释。（5 分）

（5） 详细描述在 IA-32/Linux 平台上，CPU 在读取虚拟地址 0x804c040 处的字节内容过程中的地址转换和存储访问过程。（5 分）

七、 具体描述所示程序执行printf函数的整个过程（包括从调用printf库函数开始、对应哪个系统调用、如何从用户态陷入内核态、内核的大致处理过程等，150字左右）。（9分）

八、 选择题（每小题2分，共10分）

（1） 考虑以下C语言代码：

```
int i = -137;
unsigned short usi = i;
```

执行上述程序段后，usi的值是（ ）

A. 2^{32}-137　　B. 2^{31}-137　　C. 2^{16}-137　　D. 2^{15}-137

（2） 假设R[eax] = 0000 01F3H，R[ebx] = EF89 0010H，执行指令“mulw %bx”后，寄存器内容变化为（ ）

A. R[eax] = 0000 0000H，R[dx] = 1F30H　　B. R[eax] = 0000 1F30H，R[dx] = 0000H

C. R[eax] = 0000 1F30H，R[bx] = 0000H　　D. R[eax] = 0000 1F30H，其余不变

（3） 指令序列如下（左边为指令地址，中间为机器代码，右边为汇编指令）：

```
804866c:   39 c2      cmpl  %eax, %edx
804866e:   7e f8      jle   xxxxxxxx
```

（3） 指令序列如下（左边为指令地址，中间为机器代码，右边为汇编指令）：

```
804866c:   39 c2      cmpl  %eax, %edx
804866e:   7e f8      jle   xxxxxxxx
```

若执行到上述cmpl指令时，R[edx] = 78，R[eax] = 66，则jle指令执行后将会执行（ ）处指令

A. 8048670　　B. 8048668　　C. 8048664　　D. 8048662

（4） 假定静态int型二维数组b和指针数组pb的声明如下：

```
static int b[4][4] = { {2,9,-1,5}, {3,1,-6,2}, {0,23,0,10},{0,20,3,12} };
static int *pb[4] = { b[0], b[1], b[2], b[3] };
```

若 b 的首地址为0x8049820，则 &pb[2] 和 pb[2] 分别是（ ）

A. 0x8049868、0x8049860　　B. 0x8049884、0x8049840

C. 0x8049868、0x8049840　　D. 0x8049884、0x8049868

（5） 以下选项中，不属于“故障”类的异常是（ ）

A. 非法指令操作码　　B. 整数除0　　C. 缺页　　D.断点设置