某生写了一个 C 语言程序，用于对一个数据索引文件按关键字进行排序。该程序有两个源文件：main.c 和 sort.c，它们的内容如下图所示（注：取消了写文件部分）。

```c
/* main.c */
#include <stdio.h>
#include <stdlib.h>
typedef struct record {
    ......
} RECORD;
typedef struct index {
    unsigned char key;
    RECORD *pdata;
} INDEX;
extern void sort();
INDEX rec_idx[256];
const int rec_num = 256;
void main(int argc, char *argv[])
{   FILE *fp = fopen(argv[1], "rb");
    if (fp) {
        fread(rec_idx, sizeof(INDEX), rec_num, fp);
        fclose(fp);
    } else exit(1);
    ort();
```

```c
/* sort.c：bubble sort */
extern INDEX rec_idx[];
extern const int rec_num;

void sort()
{
    int i, swapped;
    INDEX temp;
    do { swapped = 0;
        for(i=0; i<rec_num-1; i++)
            if (rec_idx[i].key > rec_idx[i+1].key) {
                temp.key = rec_idx[i].key;
                temp.pdata = rec_idx[i].pdata;
                rec_idx[i].key = rec_idx[i+1].key;
                rec_idx[i].pdata = rec_idx[i+1].pdata;
                rec_idx[i+1].key = temp.key;
                rec_idx[i+1].pdata = temp.pdata;
                swapped = 1;
            }
    } while (swapped);
}
```

-32/Linux 平台上用 GCC 编译驱动程序处理，main.c 和 sort.c 的可重定位目标文件名分别是

假设在 IA-32/Linux 平台上用 GCC 编译驱动程序处理，main.c 和 sort.c 的可重定位目标文件名分别是 main.o 和 sort.o，生成的可执行文件名为 bubsort。使用“objdump –d sort”得到反汇编部分结果如下。

```
0  08048530 <sort>:
1  8048530: 55                          push   %ebp
2  8048531: 89 e5                       mov    %esp,%ebp
3  8048533: 83 ec 10                    sub    $0x10,%esp
4  8048536: c7 45 f8 00 00 00 00        movl   $0x0,-0x8(%ebp)
5  804853d: c7 45 fc 00 00 00 00        movl   $0x0,-0x4(%ebp)
6  8048544: e9 93 00 00 00              jmp    80485dc <sort+0xac>
7  8048549: 8b 45 fc                    mov    -0x4(%ebp),%eax
8  804854c: 0f b6 14 c5 60 a0 04 08     movzbl 0x804a060(,%eax,8),%edx
9  8048554: 8b 45 fc                    mov    -0x4(%ebp),%eax
10 8048557: 83 c0 01                    add    $0x1,%eax
11 804855a: 0f b6 04 c5 60 a0 04 08     movzbl 0x804a060(,%eax,8),%eax
12 8048562: 38 c2                       cmp    %al,%dl
13 8048564: 76 72                       jbe    80485d8 <sort+0xa8>
14 8048566: 8b 45 fc                    mov    -0x4(%ebp),%eax
15 8048569: 0f b6 04 c5 60 a0 04 08     movzbl 0x804a060(,%eax,8),%eax
16 8048571: 88 45 f0                    mov    %al,-0x10(%ebp)
17 8048574: 8b 45 fc                    mov    -0x4(%ebp),%eax
18 8048577: 8b 04 c5 64 a0 04 08        mov    0x804a064(,%eax,8),%eax
19 804857e: 89 45 f4                    mov    %eax,-0xc(%ebp)
.....  ..............  ..........  ......................
38 80485d1: c7 45 f8 01 00 00 00        movl   $0x1,-0x8(%ebp)
39 80485d8: 83 45 fc 01                 addl   $0x1,-0x4(%ebp)
40 80485dc: a1 80 86 04 08              mov    0x8048680,%eax
41 80485e1: 83 e8 01                    sub    $0x1,%eax
42 80485e4: 3b 45 fc                    cmp    -0x4(%ebp),%eax
43 80485e7: 0f 8f 5c ff ff ff           jg     8048549 <sort+0x19>
44 80485ed: 83 7d f8 00                 cmpl   $0x0,-0x8(%ebp)
45 80485f1: 0f 85 3f ff ff ff           jne    8048536 <sort+0x6>
46 80485f7: 90                          nop
47 80485f8: c9                          leave
48 80485f9: c3                          ret
```

已知 IA-32 页大小为 4KB，主存地址位数为 32 位。假设代码 Cache 和数据 Cache 的数据区大小皆为 8KB，采用 2 路组相联映射、LRU 替换算法和直写（Write Through）策略，主存块大小为 64B，系统中没有其他用户进程在执行，请回答下列问题或完成下列任务。

一、第 7~13 行指令实现什么功能？对应 sort.c 中哪条语句？第 40~41 行指令实现什么功能？（4 分）

二、访问 Cache 时主存地址应如何划分？代码 Cache 的总容量为多少位？（7 分）

三、第 18 行指令的源操作数采用什么寻址方式？第一次执行该指令时，源操作数的有效地址为多少？读取该指令过程中是否发生 TLB 缺失和缺页？为什么？读取该指令过程中是否发生 Cache 缺失？为什么？用 300 字左右简述该指令的执行过程。（25 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））

四、从反汇编代码中的哪条指令可看出 INDEX 的长度？sizeof(INDEX)为多少？编译器如何确定 INDEX 所占字节数？（4 分）

五、43 行的 jg 指令采用的是什么寻址方式？为什么？从这条指令可以看出 IA-32 采用的是小端还是大端方式？为什么？该指令中偏移量的真值为多少？（7 分）

六、根据反汇编结果画出 sort 函数（过程）的栈帧，要求分别用 EBP 和 ESP 标示出 sort 函数的栈帧底部和顶部，并标出 i、swapped 和 temp 中各个成员变量的位置。（6 分）

七、数组 rec_idx 和常量 rec_num 的起始虚拟地址分别是多少？数组名 rec_idx 和常量名 rec_num 分别在 main.o 中的哪个节（section）中定义？可执行文件 bubsort 只读代码段的起始地址是多少？可读可写数据段的起始地址可能是多少？main.c 和 sort.c 中各定义了哪几个符号？这些符号分别属于 bubsort 的哪个段？（10 分）

八、若数组 rec_idx 定义为 main 函数内部的局部变量，则应如何调用 sort 函数（写出 sort 函数原型声明即可）？与题干中 main.c 的做法相比，程序包含的指令条数会增加还是减少？为什么？在被调用过程 sort 中，如何获得数组 rec_idx 的首地址，相应地，804854c 处的指令如何变化（写出两条汇编指令即可）？在 rec_idx 被定义为非静态局部数组（即 INDEX rec_idx[256];）和静态局部数组（即 static INDEX rec_idx[256];）的情况下，rec_idx 分别存放在进程虚存空间中的何处？（10 分）

九、执行第 40 行指令时，源操作数所在主存块映射到数据 Cache 哪一组？rec_idx 数组共占多少字节？rec_idx 所在主存块被映射到数据 Cache 的哪些组中？第 40 行指令源操作数所在主存块会不会被 rec_idx 所在主存块替换出来？假定 rec_idx 数组各元素已经按升序排好序，且从 main 跳转到 sort 第一条指令后 R[esp]=0xbffa003c，则 sort 栈帧所在的虚拟地址范围是什么？执行 sort 函数过程中，对 sort 的局部变量的访问是否发生 Cache 缺失？为什么？执行 sort 函数过程中，rec_idx 数组的 Cache 访问命中率为多少？（17 分）

十、在执行 bubsort 程序过程中，是否会陷入内核执行？计算机系统如何实现 fread 函数的功能（要求从调用 fread 库函数开始简要说明，包括对应哪个系统调用、如何从用户态陷入内核态、内核的大致处理过程等，300 字左右[illegible]分）