Name: ________________

Directions: **Work only on this sheet** (on both sides, if needed). **MAKE SURE TO COPY YOUR ANSWERS TO A SEPARATE SHEET FOR SENDING ME AN ELECTRONIC COPY LATER.**

On all Tests, 32-bit word size on Intel machines running Linux is assumed unless otherwise stated.

1. (15) The execution (not fetch) of an IRET instruction makes ___________ ___________ (fill the first blank with a number, the second with *more* or *fewer*, and the third with *read* or *write* accesses to memory than does an RET. Assume no cache.

2. (15) Page replacement policy is set by (i) hardware; (ii) system software; (iii) the application programmer; (iv) a combination of (i) and (ii); (v) a combination of (i) and (iii); (vi) a combination of (ii) and (iii); (vii) a combination of (i), (ii) and (iii).

3. Consider the code
   ```c
   int main()
   {
       int m;
       scanf("%d", &m);
       m++;
       printf("%d\n", m);
   }
   ```

   Running this through `gcc -S` yields:

   ```
   1 .LC0: .string "%d"
   2 .LC1: .string "%d\n"
   3 .text
   4 main:
   5     pushl %ebp
   6     movl %esp, %ebp
   7     andl $-16, %esp
   8     subl $32, %esp
   9     leal 28(%esp), %eax
   10    movl %eax, 4(%esp)
   11    call scanf
   12    movl 28(%esp), %eax
   13    addl $1, %eax
   14    movl %eax, 28(%esp)
   15    movl 28(% esp), %eax
   16    movl %eax, 4(% esp)
   17    movl $.LC1, (% esp)
   18    call printf
   19    leave
   20    ret
   ```

(a) (15) Fill blank (a).
(b) (10) Fill blank (b).
(c) (10) Fill blank (c).
(d) (10) Fill blank (c).

(e) (10) Fill blank (e).
(f) (15) Give the line numbers in the assembly code which could cause a page fault when the instruction is executed (not fetched). Do NOT include any lines containing a blank, and do NOT include the even-numbered lines.

Solutions:

1. 2 more reads
2. (ii)
3.a-e

```c
   .LC0: .string "%d"
   .LC1: .string "%d\n"
   .text
   main:
   pushl %ebp
   movl %esp, %ebp
   andl $-16, %esp
   subl $32, %esp
   leal 28(%esp), %eax
   movl %eax, 4(%esp)
   movl $.LC0, (%esp)
   call scanf
   movl 28(%esp), %eax
   addl $1, %eax
   movl %eax, 28(%esp)
   movl 28(%esp), %eax
   movl %eax, 4(%esp)
   movl $.LC1, (%esp)
   call printf
   leave
   ret
   ```

3.f 7, 15, 17, 21, 23

---

I've removed some extraneous material.