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

```
int main()
{
   int m;
   scanf("%d",&m);
   m++;
   printf("%d\n",m);
}
```

Running this through **gcc -S** yields:[1]

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

(a) (15) Fill blank (a).

(b) (10) Fill blank (b).

(c) (10) Fill blank (c).

(d) (10) Fill blank (c).

[1] I've removed some extraneous material.

(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

```
.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