Sample Questions

1. Consider the following program:

```
ldr r3, [r4], #4
add r1, r2, r5
sub r2,r3,r4
ldrb r8, [r10, #1]
```

a. Draw the diagram showing the pipeline progression. (See Quiz 4 for a sample of the expected format for the table.)

b. How many cycles are required (from start to finish) to fully execute the lines?

c. Rewrite this problem to make it harder by adding a dependency between the first two lines, then repeat (a) and (b).

2. Consider the following subroutine, called find, which searches for a number in an array of words, and returns the number of occurrences of the number. It takes as parameter: (1) The 32-bit value to be searched for; (2) The address of the array; (3) The number of entries in the array. It returns the number of times the value is found in the array.

Here is one implementation:

```
find:
        stmfd sp!, {r4,r5}
        mov r2, #0 ; loop counter
        mov r3, #0 ; number of times item found

s_loop:
        cmp r2, r1
        bge s_done

        ldr r5, [r4], #4
        cmp r5, r0
        bne skip
        add r3, r3, #1
skip:
        add r2, r2, #1
        b s_loop

s_done:
        ldmfd sp!, {r4,r5}
        mov r0, r3
        bx lr
```

Optimize this subroutine by reducing the number of cycles it take to run. (You should write out an optimized version of this subroutine.) In addition, write a short paragraph describing the optimizations you did and how many cycles your implementations saves over the original.

3. Random questions on exceptions and interrupts

a) Why is a fast interrupt exception faster to process than a regular interrupt exception?

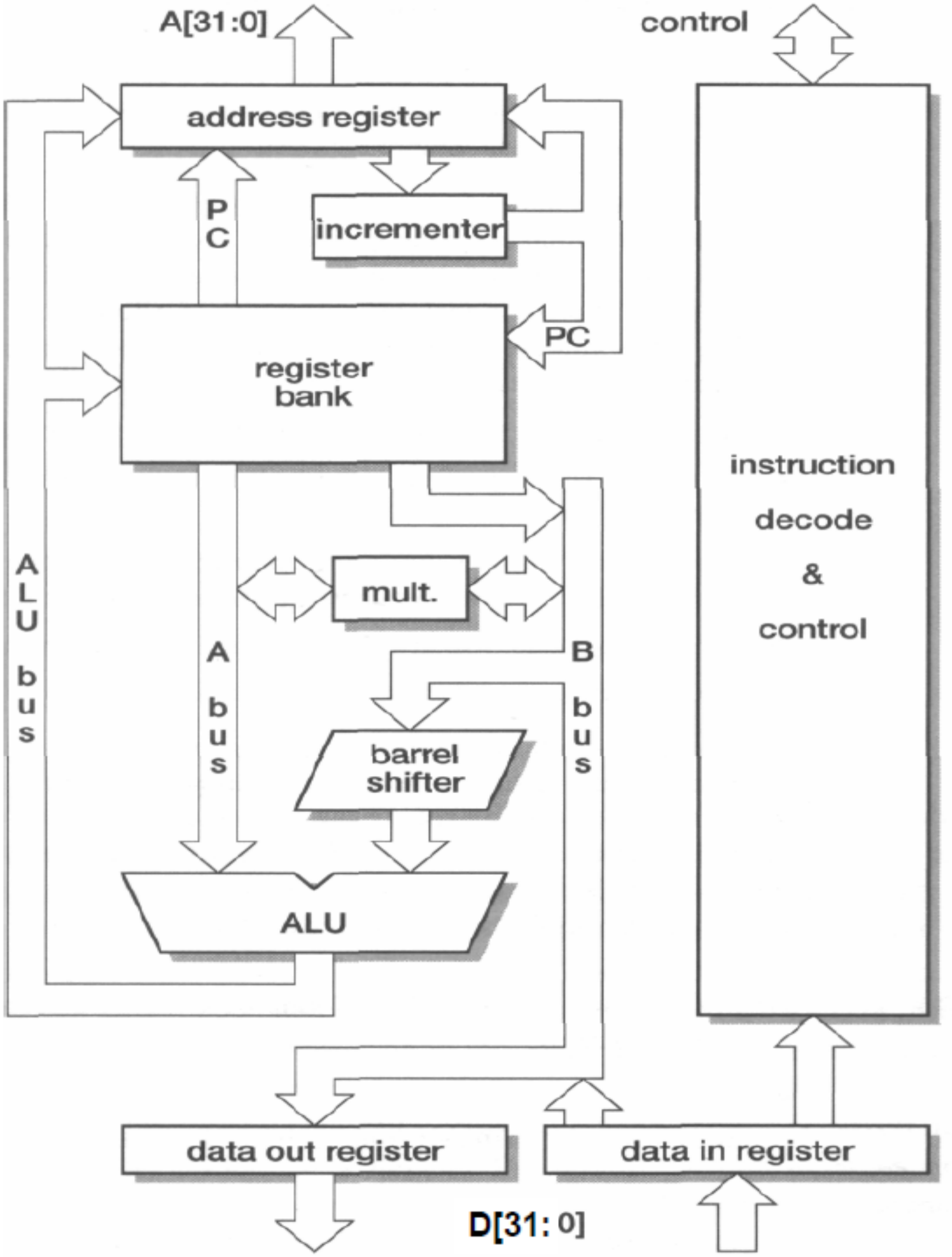
b) Special instructions (MRS and MSR) are needed to enable and disable interrupts. Why?

c) Why should an interrupt handler disable interrupts?

d) What is the difference between a processor mode and an exception?

4. Show the steps of using trial subtraction in 8-bits to calculate 2000/45.

5. Consider the following diagram for a 3-stage pipeline ARM processor:



Shade in the portions of the datapath that are used when the following instruction executes:

```
mov r0, r1, r2, ASR #6
```