

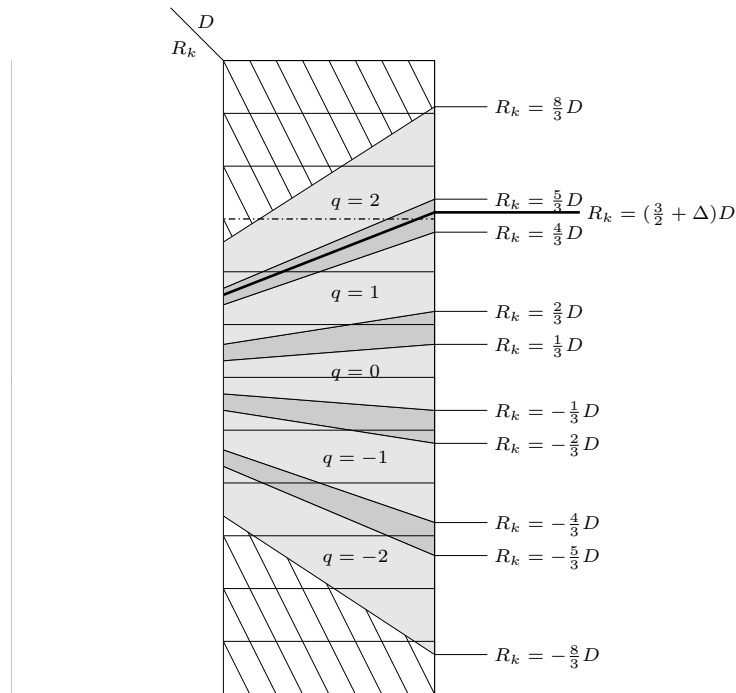
# Homework 3

Prof. Loh  
 CS3220 - Processor Design - Spring 2005  
 Handed Out: 10 Feb 2005  
 Due: 22 Feb 2005

## 1. SRT Division

In the Intel Pentium “FDIV” bug, five entries in the SRT lookup table were removed. This effectively set the entries to zero when they should have contained 2’s. Consider the region of the table between  $\frac{R_k}{D} = \frac{4}{3}$  and  $\frac{5}{3}$  where the table could return either 2 or 1 and still result in a correct answer. In the version of the SRT algorithm that we covered in class, we divided this region such that any entry above the midpoint ( $\frac{R_k}{D} = \frac{3}{2}$ ) returns a value of 2, and any entry below returns 1.

Let us suppose that the engineer in charge of the SRT lookup table is not good at plotting straight lines, and ends up using a line that is slightly “higher” than  $\frac{R_k}{D} = \frac{3}{2}$  (shown as a bold line in the figure below – the “correct” line would have terminated slightly lower where the dash-dot horizontal line hits the end of the table). That is, he uses the line  $\frac{R_k}{D} = \frac{3}{2} + \Delta$  (see the figure below). How would this affect the correctness of the divider? Assuming you cannot change the lookup table, how would you modify the algorithm to fix the problem?



Hint: Consider that (1) the region is supposed to be able to return either 1 or 2, and (2) the table is not indexed with  $R_k$  and  $D$ , but rather by the approximated versions  $\tilde{R}_k$  and  $\tilde{D}$ .

## 2. Floating Point

In class we covered the IEEE single- and double-precision floating point formats. Let us consider a “half-precision” format:

Sign		Exp				Mantissa									
s	e	e	e	e	e	m	m	m	m	m	m	m	m	m	m

Except for the different field sizes, assume that the format is in all other respects the same as the single- and double-precision FP formats.

- (a) What is the largest representable real number using this format? (not counting  $+\infty$ )
- (b) What is the smallest representable (normalized) non-zero number?
- (c) What is the base-10 value of the half-precision number 1110011001011000?
- (d) What is the half-precision binary representation of  $\frac{13}{32}$ ?

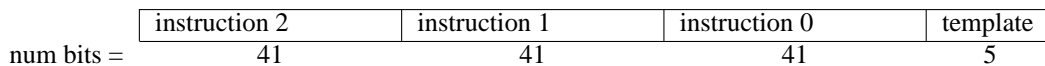
**(Problem 3 on next page.)**

### 3. ISA Design Tradeoffs

VLIW (Very Long Instruction Word) instruction set architectures have gotten a lot of press over the last several years as a result of the joint HP/Intel efforts to develop “IA64” for the Itanium processors. VLIW instruction sets are also very popular in embedded media processors and digital signal processing applications.

The idea behind VLIW is to make the compiler find instructions that are independent and could be executed in parallel, and then pack them all together. The theory is that the hardware can then execute these instructions in parallel without devoting expensive and complex hardware to discover the parallelism on its own.

An IA64 (very long) instruction is 128 bits long and has the following format:



The template bits specify what kinds of instructions fill the three instruction slots. Some example templates:

Template (in hex)	Slot 0	Slot 1	Slot 2
00	M-unit	I-unit	I-unit
08	M-unit	M-unit	I-unit
0C	M-unit	F-unit	I-unit
0E	M-unit	M-unit	F-unit
10	M-unit	I-unit	B-unit
16	B-unit	B-unit	B-unit

Where I-, F-, M- and B-units are integer, floating point, memory and branch units (as in execution units like ALUs), respectively. Instruction types are then divided based on what types of units they can execute on. For example, simple integer ALU operations can execute on either I- or M-units, where as memory instructions can only execute on M-units. The compiler chooses a template that best matches the available instructions to minimize the number of unused instruction slots (unused slots are filled with no-ops).

Each of the 41-bit instruction slots encodes an instruction in a RISC-like format, including three register specifiers (for operations like add which take two sources and one destination) plus opcode bits. IA64 has 128 general purpose integer registers, and so each register specifier requires 7 bits.

- What advantages does a VLIW architecture like IA64 have over a CISC architecture like x86?
- What advantages does a VLIW architecture have in common with a RISC architecture like MIPS or Alpha?
- What complexities does VLIW introduce over RISC architectures?
- What advantages does a CISC architecture hold over VLIW?
- Is VLIW a good idea? Why and/or why not?