Section 2: Iron Law & Instruction Encoding

CS 152
Spring 2010

TA: Andrew Waterman

Problem 1

Suppose we are designing the next version of our company's killer microprocessor. We have made the following measurements from an important customer's favorite floating-point intensive program:

Frequency of floating-point (FP) instructions = 25%
Average CPI of FP instructions (excluding square-root) = 4.0
Average CPI of integer instructions = 1.33
Frequency of floating-point square root (FPSQR) = 2%
CPI of FPSQR = 20

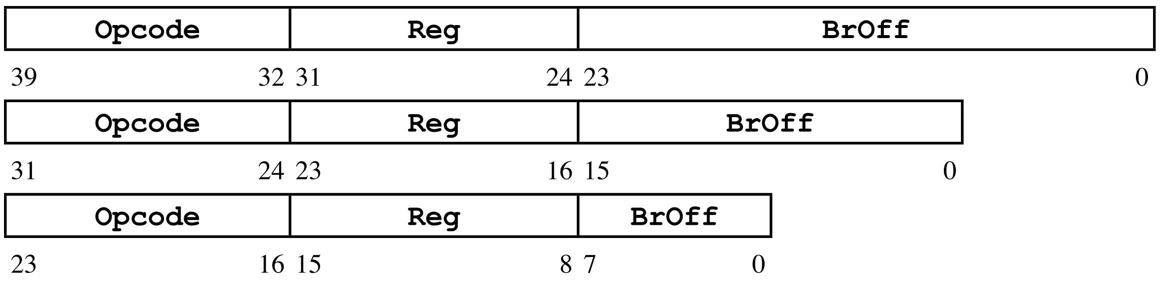
We have devised two design alternatives to improve this program's performance. One will decrease the CPI of FPSQR to 2 and the other will decrease the average CPI of all other FP operations to 2.5. Which one will yield the greatest overall improvement in performance? Hint: Consider the Iron Law of Processor Performance.

Problem 2

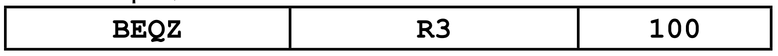
Ben Bitdiddle is designing a handheld device. However, because the device's storage capacity and battery life are limited, he needs to reduce the size of his code. Therefore, he decides to design variable-length instruction set formats for the handheld device to produce more compact code.

Part A

Ben is trying to decide whether it is worthwhile to have multiple offset lengths for branches. He considers the following formats for a branch instruction.



For example,



would mean if **`R3`** is zero, branch to location **`100 + PC`**.

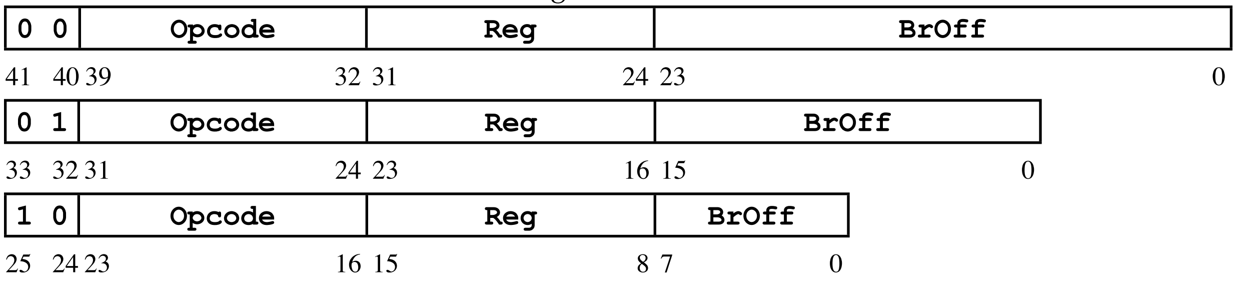
He also has the following statistics reflecting the cumulative percentage of branch instructions that can be accommodated with the corresponding number of bits needed to encode the offset.

# Offset Magnitude Bits	Cumulative Branches
1	2.80%
2	10.50%
3	22.90%
4	36.50%
5	57.40%
6	72.40%
7	85.20%
8	90.50%
9	93.10%
10	95.10%
11	96%
12	96.80%
13	97.40%
14	98.10%
15	98.50%
16	99.50%
17	99.50%
18	99.80%
19	100%
20	100%

On average, how many bits is a branch instruction reduced by using this variable-length offset encoding, compared with the fixed 24-bit offset? Suppose branch instructions account for 10% of the static code. How much do the variable-sized branch offset encodings reduce the total code?

Part B

In order to implement the above variable-length instruction encoding, Ben needs to specify each instruction's length. Since there are 3 possible instruction lengths, he decides to add 2 more bits as the instruction length field.



0 0 | Opcode | Reg | BrOff
41 40 39 32 31 24 23 0
0 1 | Opcode | Reg | BrOff
33 32 31 24 23 16 15 0
1 0 | Opcode | Reg | BrOff
25 24 23 16 15 8 7 0

What is the overhead cost of adding these 2 bits? Describe an alternative encoding method to specify the instruction length.

Part C

Without re-running the calculations, do you expect that it would be worthwhile to provide a finer granularity of branch offset length--say, providing offsets of 4,8,12,...,24 bits?

Part D

What are some disadvantages of using variable-length instructions?