Due at the beginning of recitation R22 on Friday November 30.

Consider a direct-mapped branch target buffer (BTB) like the one we’ve seen in lecture.

(A) Assume you have a processor that supports 32-bit addresses. Given a BTB with 8 entries in it. What would be the size of each entry of your BTB if you had a long enough tag to differentiate between each address in your addresses space? Make sure to account for the tag bits, the target address bits, and the valid bit.

Size of a BTB entry: _______ 60 bits _______

Since there are 8 entries in the BTB, we need 3 index bits in addition to the bottom two bits being zero for word alignment. So the tag is 32-5 = 27 bits. The target address is 32 bits, and the valid bit is 1 bit, for a total of 27 + 32 + 1 = 60.

(B) Now suppose that you are told that each of your code segments was at most 4k bytes long. What effect, if any, would that have on the size of your BTB entry if you want to use the minimum number of tag bits necessary to differentiate between any two addresses in a given program? Assume that the target address bits do not change.

Size of a BTB entry (with 4k max code segments): _______ 40 bits _______

If each code segment is at most 4k bytes long, then only 12 address bits are required to identify the different locations in the code. Given that we still need 5 bits of address of the index and the word alignment, that leaves us with 7 bits for the tag in order to fully identify any branch in our code segments. 7 + 32 + 1 = 40 bits.

(C) What are the implications of having only 8 entries in your BTB?

If the BTB is too small, then you are more likely to have conflict misses in your direct mapped BTB. These conflicts will generally lead to a higher misprediction rate.