Branch Prediction Example

Consider the following code:

```
int c;
int main () {
        int     i, j;

        for (i=0; i<1000; i++) {
                for (j=0; j<4; j++) {
                        c++;
                }
        }
        return c;
}
```

The compiler generates this assembly (more-or-less):

```
main:
        leal    4(%esp), %ecx           ; function overhead
        andl    $-16, %esp
        pushl   -4(%ecx)
        pushl   %ecx
        xorl    %ecx, %ecx              ; i = 0 (%ecx)
.L2:
        xorl    %edx, %edx              ; j = 0 (%edx)
.L3:
        movl    c, %eax                 ; %eax = c
        addl    $1, %edx                ; j++
        addl    $1, %eax                ; %eax++
        movl    %eax, c                 ; c = %eax
        cmpl    $4, %edx                ; if j < 4 then goto L3
        jne     .L3                     ; INNER LOOP BRANCH
        addl    $1, %ecx                ; i++
        cmpl    $1000, %ecx             ; if i < 1000 then goto L2
        jne     .L2                     ; OUTER LOOP BRANCH
        movl    c, %eax                 ; return c
        popl    %ecx                    ; function overhead
        leal    -4(%ecx), %esp
        ret
```

Assuming no conflicts between branch address bits, and assuming all entries are initially set to 0, how many conditional branches would be mispredicted:

1. For a branch history table (BHT) with 1-bit entries?
2. For a branch history table (BHT) with 2-bit saturating counters?
3. For a (4,2) correlating predictor? Give an answer within a tolerance of 1%.

Solution:

1. We consider two entries: the one for the inner loop branch and the one for the outer loop branch. The inner loop iterates 4 times. The first time through the outer loop, the entry for inner is initialized to zero. Subsequent times through the outer loop, that entry is also zero because, on the previous iteration, the last iteration of the inner loop was not taken. So every time through the outer loop, the pattern of taken/not for the inner loop taken will be:

   ```
   T, T, T, NT
   ```

   while the pattern of predictions will be:

```
NT, T, T, T
```

So there will be 1000 * 2 = 2000 mispredictions due to the inner loop. The outer loop branch will be mispredicted as not taken the first time, and mispredicted as taken the last time, so will add another 2 mispredictions. So there are 2002 mispredictions total.

2. Again we consider the same two entries, this time two-bit entries. The first iteration of the outer loop will see the following pattern of outcomes and predictions for the inner loop branch:

```
outcome   : T,  T,  T, NT
prediction: NT, NT, T, T
old value : 0   1   2  3
new value : 1   2   3  2
```

Every time after that, the pattern will be this:

```
outcome   : T,  T,  T, NT
prediction: T,  T,  T, T
old value : 2   3   3  3
new value : 3   3   3  2
```

So for the first iteration of the outer loop, there will be 3 mispredictions due to the inner loop. The subsequent 999 iterations, there will be 1 misprediction due to the inner loop. So there will be 3 + 999 = 1002 mispredictions due to the inner loop. The outer loop will be mispredicted the first 2 times and the last time, so it will add 3 more mispredictions for a total of 1005 mispredictions.

3. Now there will be many entries in the table for each branch. We assume that they do not interfere with one another. Assume also that the history register is initially all 0s. Note that this example uses *global* history, i.e., the history of all branches. We could also work the example using *per-branch* i.e. *local* history, using only the history of a particular branch to predict that branch.

The global history will look like this:

```
i  i  i  i   o  i  i  i  i   o  i  i  i  i   o (i=inner, o=outer)
T, T, T, NT, T, T, T, T, NT, T, T, T, T, NT, T ...
```

So the values in the history register look like this:

```
        after this branch       this branch outcome     next branch outcome
0000 initial
0001 first inner loop branch    taken                   taken
0011 second inner loop branch   taken                   taken
0111 third inner loop branch    taken                   not taken
1110 fourth inner loop branch   not taken               taken
1101 outer loop branch          taken                   taken

1011 first inner loop branch    taken                   taken
0111 second inner loop branch   taken                   taken
1111 third inner loop branch    taken                   not taken
1110 fourth inner loop branch   not taken               taken
1101 outer loop branch          taken                   taken
1011 first inner loop branch    taken                   taken
...
```

Let's look at the values in the table for each of the histories of the branch for the inner loop at the time the very last branch is predicted:

```
0000 1
0001 1
0010 ?
0011 1
0100 ?
0101 ?
0110 ?
```

```
0111 3
1000 ?
1001 ?
1010 ?
1011 3
1100 ?
1101 3
1110 3
1111 0
```

The history pattern 1111 leads to the next branch always being not taken, so the counters for these patterns will saturate to 0. The history patterns 1011, 0111, 1110, and 1101 always lead to the next branch being taken. So, except for a few initial mispredictions the inner loop will have no mispredictions. The outer loop always has the history 1110 and will be taken 999/1000 times, so it will have very few mispredictions (3 to be exact).