Caroline Case

Project 2

**Loop Optimization Techniques**

Loop fusion combines several loops, which reduces loop overhead, and ultimately runtime. The loops to be combined must have the same number of iterations and the iterator must start at the same number. For instance, the following code is the two loops before loop fusion:

for(int i = 1; i <= 5; i++) {

a[i] = a[i] + 1;

}

for(int i = 1; i <= 5; i++) {

b[i] = b[i] \* 2;

}

The following code is the end loop after loop fusion:

for(int i = 1; i <= 5; i++) {

a[i] = a[i] + 1;

b[i] = b[i] \* 2;

}

Loop interchange reorders nested loops. For example, the following code is a nested loop before loop interchange:

for(int i = 0; i <= 5; i++) {

for(int j = 0; j <= 10; j++) {

arr[i][j] \*= 2;

}

}

After loop interchange, the code looks as follows:

for(int j = 0; j <= 10; j++) {

for(int i = 0; i <= 5; i++) {

arr[i][j] \*= 2;

}

}

Loop unrolling is the removing of iterations. This can reduce loop overhead because it’s faster with small loops to just do each iteration by itself rather than checking for the value of i and incrementing every time. For example, the following code is before the loop unrolling:

for(int i = 0; i < 3; i++) {

cout << arr[i];

}

And this is after loop unrolling:

cout << arr[0];

cout << arr[1];

cout << arr[2];

cout << arr[3];

**Loop-Carried Dependencies**

A loop-carried dependency is when a line of code in a loop relies on a past line of code in the same loop. This can affect the ability to optimize loops because the order of the code matters in this case. The code will not execute properly if it’s not in order.

**Array Element Aliasing**

Array element aliasing is when an element of an array’s address is pointed to by different variables either directly or indirectly [IBM]. These aliases can affect the ability to optimize loops because optimizations will change one variable that’s aliased along with all other variables that point to the same address, even if those aren’t supposed to be changed. So when optimizing, the changing of an aliased array element will affect all other aliased variables, even if the programmer didn’t intend for that to happen.

**Hands-On Test and Results Report for GNU Compilers on the Whetstone Code**

Works Cited \* cite in IEEE

<https://compileroptimizations.com/category/loop_fusion.htm>

<https://www.cs.colostate.edu/~mstrout/CS553Fall09/Slides/lecture18-fusion-KellyPugh.ppt.pdf>

<https://engineering.purdue.edu/~milind/ece573/2011spring/lecture-14.pdf>

<https://www.geeksforgeeks.org/loop-unrolling/>

<https://www.ibm.com/support/knowledgecenter/SS2MB5_14.1.0/com.ibm.xlf141.bg.doc/proguide/aliasing.html>