Institut für Informatik — TUM
Scientific Computing in Computer Science
R. Wittmann

WT 2015

Tutorial (Advanced Programming) Worksheet 13:

Assignment 1: Code Optimization and Performance

Consider the following code snippet:

```
void myfunc(float* a, float* b, float* c, size_t n) {
    for (size_t i=0; i < n; ++i) {
        if (n > 100) {
            if (i < n/2) {
                c[i]=a[i]*b[i/2];
            } else {
                c[i]=c[i-n/2]+a[i]*b[(i-n/2)/2+1];
            }
        }
    }
}
```

- Let `n=1000` in the function call from above. How many float elements from each array are read/written from/to main memory?

  You may assume that read- and write-units are perfectly distinct (i.e., a write does not invoke a read-operation at the same time) and that elements remain in cache once they are read. However, data which was written has to be read back from main memory again if it is required for another computation.

arrays	a	b	c
read			
write			

- Assume your compiler does not perform any kinds of code optimizations, but translates source code instructions line by line into (non-vectorized) machine code. Which instructions can be considered major bottlenecks in the code snippet of `myfunc`? Rewrite the code snippet without these bottlenecks to speed up its execution (e.g. for `n=1000`). Name/Explain the strategies you have used to optimize it (1 sentence).

- Name three other optimization techniques how scientific source code may be restructured to yield faster program execution.

Homework Assignment 2: Using Auto-Vectorization for Adding Two Vectors

Write a program with a simple loop that adds two float vectors of equal length. Make sure the vector size is sufficiently large so vectorized code can be generated. Then, disassemble the compiled executable and look whether the `xmm` or `ymm` registers are used and how. Further, compile the program with different optimization flags (`-O` for GCC and `/O` for Visual Studio). How does the disassembled code change?

Class Assignment 3: Performance Aware Computing

In the following, four code fragments with simple, non-nested loops are given. Assume the compiler knows for all arrays passed as function parameters, that they do not alias each other.
Discuss likely behavior of the auto-vectorizer for every fragment ($\leq$ 3 sentences each).

```
void product( float* a, float* b, float* c ) {
  for( int i = 0; i < 400; i++ ) {
    c[i] += a[i] * b[i];
    if( a[i] < 0.0f )
      break;
  }
}

void compute( double* a ) {
  for( int i = 0; i < 800; i++ ) {
    a[i] = a[i] * a[i];
    a[i] = a[i] + a[i];
  }
}

void mixed( float* a, float* b, float* c ) {
  for( int i = 0; i < 4096; i++ ) {
    if( a[i] > 0.0f ) {
      c[i] = a[i] * b[i];
    }
    else {
      c[i] = a[i] + b[i];
    }
  }
}

void sum( float* a ) {
  for( int i = 1; i < 8192; i++ ) {
    a[i] += a[i-1];
  }
}
```