Computer Systems
baiCOSY06, Fall 2015
SIMD Lab: Achieving Greater Parallelism with SIMD Instructions
Assigned: Oct. 06, Due: Wed., Oct. 16 23:59

Arnoud Visser (A.Visser@uva.nl) is the lead person for this assignment. Ysbrand Galama, Jan Geestman, Eva van Weel and Nick de Wolf will be the teaching assistants.

1 Overview

The webside OPT:SIMD[1] describes how you can use the multimedia instructions of your processor, such as the SSE (streaming extensions) and AVX (advanced vector extensions) instructions, to enhance the performance of your C programs. Your task is to optimize a polynomial evaluation function with vector code, multiple accumulators and Horner's method.

2 Downloading the assignment

You can find the the file `simdlab.tar` at the location `https://staff.fnwi.uva.nl/a.visser/education/CS/opt/simdlab.tar`.

Start by copying `simdlab.tar` to a protected Linux directory in which you plan to do your work. Then give the command

```
linux> tar xvf simdlab.tar
```

This will create a directory called `simdlab` that contains a number of files. Go into this directory, and download `addon.tar`. This will add the files `poly.c` and `pbenchmark.c` to your directory.

For the first part of the assignment your have to analyze the file `combine.c`. To compile these file, type:

```
linux> make clean
linux> make
```

[1] http://csapp.cs.cmu.edu/3e/waside/waside-simd.pdf

The result of the make are 8 `??bench` programs, which created 24 analysis results in the directory `results`. The difference between the `*.tab`, `*.txt` and `*.vtab` is their aggregation level. You would find the results of `*.vtab` files quite illustrative.

For the second part of the assignment, you need to modify the file `poly.c`. As described in the file `poly_README.txt`, you can benchmark this code with the commands:

```
linux> make -f Makefile.poly clean
linux> make -f Makefile.poly
```

3 Description

The webaside OPT:SIMD contains six problems. Possible solutions for the first five problems are given, but those solutions are machine dependent. It is not guaranteed that this solutions show the same performance on acheron or your local (virtual) machine.

3.1 Study the effect of vector code, multiple accumulators, reassociation

Reproduce the effect of vector code, multiple accumulators and reassociation on acheron or your local machine (based on the provided solutions for the combine function) and describe this in your labbook.

3.2 Write a SIMD version of the polynomial evalation function

Study the polynomial evaluation function described in the textbook [**?**] (Problem 5.5). Write a version of this function based on vector code which is at least two times as fast as the scalar version. Describe your design decisions in your labbook.

3.3 Labbook

The labbook should make the improvement of the performance of the transpose-function traceable. The Labbook will be evaluated on content, structure, wording and completeness, as described in

`http://www.practicumav.nl/onderzoeken/labboek.html`

The balance in the grading between labbook and code performance will be 12 to 24.

4 Handing in Your Work

Two BlackBoard will be created, one for your labbook and one for your code. If you have a better implementation, feel free to increase the version number and do another submission.

References

[1] R. E. Bryant and D. R. O'Hallaron, *Computer Systems: A Programmer's Perspective*, Addison-Wesley Publishing Company, USA, 2nd edition, 2010, ISBN 0136108040, 9780136108047.