Andrzej Wąsowski. *Software Programmable DSP Platform Analysis.*

**Exercise Set 7**

4 May 2005

Some of the following exercises assume that your platform has a data cache and a program (instruction) cache. If you ask me to sort them in the order of interest (from my personal highest interest to the lowest) it would be: 7.4, 7.5, 7.2, 7.3, and 7.1.

**Exercise 7.1** (project task proposal) Investigate the fetch principles of cl6x (what size of blocks are fetched from the data cache). Analyze your program to find pieces of data that are used together and try (manually) allocate it in "corrrect" blocks, if the compiler fails to do so.

This may be cumbersome. One way to achieve it is to store data in a big array of characters (or a malloc block), so that you have complete control over where things are placed. Also you should not start optimizing this in doubtful cases as manual control over allocation is weak, and you can spoil register allocation significantly (getting it right requires a lot of effort).

**Exercises 7.2** (these are many tasks proposals) Skim chapter 5 of *Programmer's Guide*. This chapter proposes manual ways to improve your code. Do not be discouraged by the fact that it is explained in assembly (one cannot possibly explain it on a more abstract level). Make a list of interesting optimizations.

Analyze your project code to see whether any of potential omptimizations introduced in the previous step applies. This exercises gives you a chance to identify the transformation not applied autamatically by cl6x, and also may significantly impact the speed of your code.

**Exercises 7.3** (task proposal) Study the problem of basic block alignment in the instruction cache, as described in the lecture. Identify the blocks within busy loops in your code, that are only executed rarely[1]. Check whether the compiler has scheduled the block sequentially, or reordered them to minimize the cache-misses. Try to reorder the source code (using goto statements) to enforce a more efficient order.

**Exercise 7.4** (task proposal, makes sens also in absence of data cache). Check whether compiler efficiently applies scalar replacement in your loops. Do it yourself, if not.

**Exercise 7.5** (task proposal) Consider the applicability of blocking (computing in parts) for your code, to avoid many cache-misses. If you estimate that the program will gain speed on this, implement blocking in your project (it only makes sense, if the data cache is smaller than the data set you are working on in a given loop).

[1] If you do not have such blocks, then feel free to skip the exercise.