CS5214 – THE DESIGN AND IMPLMENTATION OF OPTIMISING COMPILERS

Assignment 4
(The last assignment)

Due: April 22, 2016

Notes for the programming assignment:

1. This assignment is to be done on an individual basis. While you can discuss the assignment with others as much as is necessary, plagiarism will be doubly penalized – both the copier *and* the copied will be penalized. In addition, plagiarism from the web will also be penalized.

2. The assignment is due at 11.59pm on April 22, 2016. This is a hard deadline.

Submit your answers in the form of a Zip file containing (a) a Word document or a PDF (only these two formats are acceptable – if you had hand-written it, then please scan it into PDF), and (b) the source code (for Linux please) in the "Assignment 4 submissions" subfolder of "Student Submission" in the CS5214 IVLE with your submission named as such:

<*your-name*>-assignment-4.zip

3. There is no page limit to your answer but please try to be concise and to-the-point.

Description of Assignment 4 (Fun with Matrix Multiply)

This assignment involves the good old matrix multiplication routine.

1. In C (we need C because of efficiency), write down the (single-precision) floating point matrix multiplication (MM) routine.

2. Write a tiled version of your matrix multiplication code. Make sure that your tile size is parameterized so that you can do the next step conveniently.

3. Using very large matrices (sizes in the tens of million entries – I can't exactly say how much as it will depend on the amount of memory you have on the machine you test on but do try to push it to the limit because if it is too small, you won't see the difference), test out the performance of your MM using different tile sizes. Plot the performance as a function of tile size. What do you observe? (For better accuracy, do this on a machine that is running as few applications as possible.)

4. Show how you can produce an efficient Intel SSE SIMD assembly program from your code. (Please do not copy this off the web. Not only

is this stuff fun, I predict that knowing it will come in very handy in your future work. So do try it on your own.) You will need to research a bit on the Intel SSE instruction set. To do this, you may want to consider the following link that

http://en.wikipedia.org/wiki/X86_instruction_listings
- for a summary and
http://www.intel.com/products/processor/manuals/
- full gory details in Volume 2A and 2B.

You may want to use GCC to generate a non-SSE assembly code (using "`gcc -S …`" and then replace the kernel's assembly code with SSE instructions.

5. Compare your code with that of the vectorized code that GCC produces. You will require at least GCC 4.0 and the tree-vectorize options (see the man pages).

6. (*This is more open-ended in that it is impossible to specify here a complete ISA. So write down clearly any assumptions you make.*) Now suppose you have a hypothetical VLIW machine (with a RISC instruction set). For example, use the hypothetical syntax of "**`R3 := R1 + R2`**" for the addition of `R1` to `R2` with the result placed in `R3`, "**`R1 := MEM[R2]`**" and "**`MEM[R2] := R1`**" for loads and stores. The machine can perform 2 arithmetic operations (integer or floating point), 2 memory operations, and 1 branch operation in a single cycle, i.e. it has the following instruction format:

ALU1	ALU2	MEM1	MEM2	BRANCH

Assume that 32 rotating floating point registers and 32 rotating integer registers are supported.

(a) Write down the assembly code for your matrix multiplication routine in the instruction set of this machine – which you are free to invent.

(b) Perform software pipelining and show the resultant code. You may assume predicated execution. If you assume a special branch instruction in support of software pipelining, then explain clearly what the instruction does.

Grading:

For this assignment, the grading criteria shall be as follows:

1. Correctness of solution – 20%
2. Elegance of solution (esp. in terms of simplicity) – 10%
3. Clarity of writeup – 10%

Have fun!