Homework 5: Solutions Dominik Gresch

Exercise 1: Mini-MMM

The system used for this exercise is an Intel Core i7-4700MQ @ 2.4 GHz
(32 kB L1 cache, 256 kB L2 cache, ADD tp 1, MUL tp 2)

a) - d) The files can be found in `impl/src/`

e) The working set fits into cache if

$$\left\lceil \frac{N_B^2}{B_1} \right\rceil + 3 \cdot \left\lceil \frac{N_B \cdot M_U}{B_1} \right\rceil + \left\lceil \frac{M_U \cdot N_U}{B_1} \right\rceil \leq \frac{C_1}{B_1}$$

This gives $N_B \leq 87$ for `code1` and `code2` (which has to be rounded to $N_B = 86$) and $N_B = 88$ for `code3` (see calculations in the `Mathematica` file).
I get the following results:

version	performance $[\frac{\texttt{flop}}{\texttt{cycle}}]$
`code1`	0.86
`code2`	1.44
`code3`	1.30

It turns out that `code2` is by far the fastest.

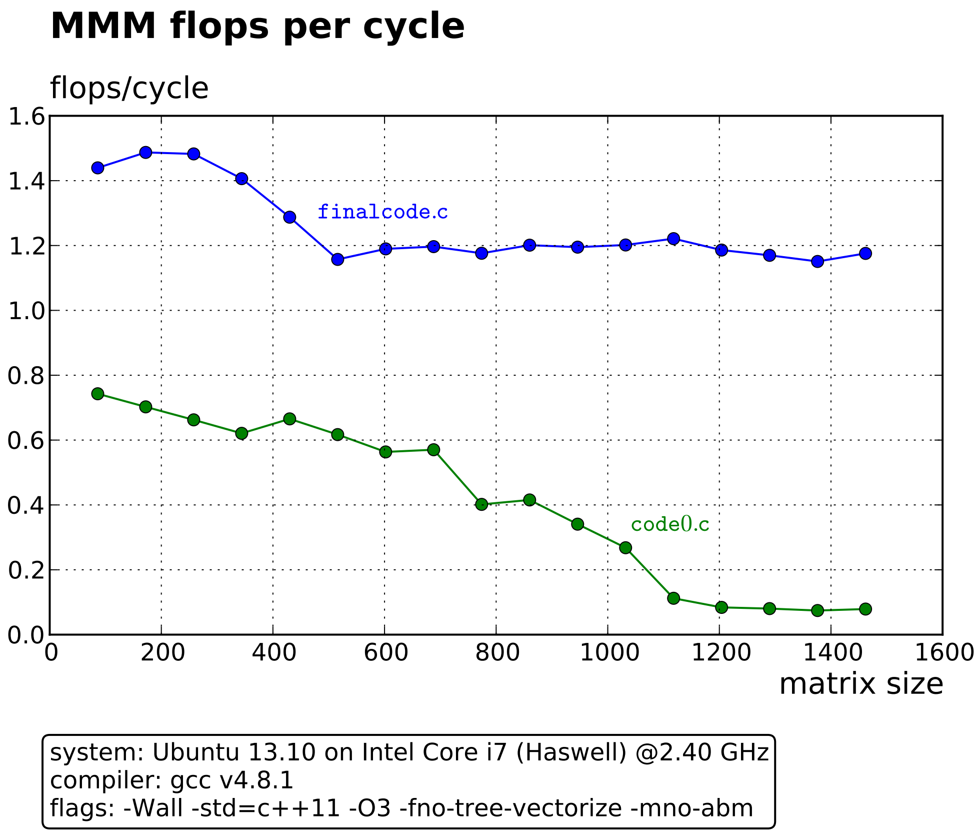
f) Now, we get $N_B \leq 177$ for `code1` and `code2` (round to $N_B = 176$) and $N_B \leq 179$ (round to $N_B = 176$) for `code3`.

version	performance $[\frac{\texttt{flop}}{\texttt{cycle}}]$
`code1`	0.86
`code2`	1.18
`code3`	1.18

Amongst those versions, `code2` is again the fastest (on par with `code3`, within error). Overall, however, `code2` for $N_B = 86$ gives the best performance.

Exercise 2: MMM

`finalcode.c` can again be found in `impl/src/`



Exercise 3: Roofline

a) The processor can issue 2 mults and 1 add per cycle, hence the peak performance is

$$\Rightarrow \pi = 3\ \frac{\texttt{flop}}{\texttt{cycle}}$$

b) The maximum bandwith is $25.6\,\mathrm{GB} \cdot \mathrm{s}^{-1}$ at a frequency of $3.2\,\mathrm{GHz}$. This means the maximum bandwidth is

$$\beta = \frac{25.6}{3.2}\ \frac{\texttt{byte}}{\texttt{cycle}} = 8\ \frac{\texttt{byte}}{\texttt{cycle}}$$

c) MMM cannot reach peak performance on this system, because the flops are not balanced: There are equally many additions and multiplications, whereas the processor can issue twice as many multiplications as additions. Therefore the maximum performance for MMM is limited by the throughput of ADD, i.e. only one multiplication can actually be issued per cycle (limited by the dependency on the adds).
The performance of MMM is hence limited by

$$\pi' = 2\ \frac{\texttt{flop}}{\texttt{cycle}}$$

d) For `fjump(x, 3*1024*1024)`, the computation is memory bound:
For each iteration in `i`, at least $\frac{3 \cdot 1024 \cdot 1024}{16} \cdot 64\ \texttt{byte} = 12\ \texttt{MB}$ of data are read (one cacheline

containing `x[j]` and `x[j+1]`; the `x` for the next `j`-iteration cannot be on the same cacheline).
Since the LLC is only 6 MB, this means that access to `x[j]` will always be a miss. Accessing `x[j+1]` will be a hit as long as it is on the same cacheline as `x[j]` (this depends on the alignment of `x`). A hard lower bound for Q is

$$Q \geq 16 \cdot 12 \texttt{ MB} = 192 \texttt{ MB}$$

Which can be used to find an upper bound for the operational intensity:

$$W = 2 \cdot 3 \cdot 1024 \cdot 1024 \texttt{ flop} = 6 \texttt{ Mflop} \Rightarrow I \leq \frac{6}{192} \frac{\texttt{flop}}{\texttt{byte}} = \frac{1}{32} \frac{\texttt{flop}}{\texttt{byte}}$$

This then gives a tighter bound for the performance:

$$\pi'' \leq \beta \cdot I = \frac{1}{4} \frac{\texttt{flop}}{\texttt{cycle}}$$