Home Work Assignment - MCA - Henrique Dantas, 4172922

1. Optimizations Performed

1.1. Algebraic simplifications

In order to optimize the program the original source code was rewritten, please see the `fir_filter` function in `fir_filter_optimized.c`. In essence the outer loop was swapped with the inner loop since this way the number of reads and stores is reduced.

1.2. Loop unrolling

In the source code there are a total of 3 loops. Assuming the only variable that may change is `TAP` the only real candidate for unrolling is the loop in the main function. In the filter, both loops' sizes are dependent on the size of `TAP` making it impossible to unroll. Moreover no more information about this variable is provided (*eg.* `TAP` is even or odd) so it is not possible to reduce the unrolling factor.

On the other hand, for the main the number of variables to print is known at compile time and consequently loop can be fully unrolled. However, there is an upper bound on the code's size and therefore a factor of 3[1] was used.

1.3. Common subexpression elimination

With the restructured algorithm it is now possible to store `b[j]` just once in a temporary variable and then use it for all elements of `y`.

This technique was also used for the expression `x[i-j]`, in particular using distributivity $8 \cdot (i - j) = i \cdot 8 - j \cdot 8$.

1.4. Code scheduling

After implementing the three aforementioned optimizations the code was reorganized by moving instructions and renaming a few registers to avoid, as much as possible, stalls and redundancies.

2. Simple vs. Optimized

TAP	Cycles	Instructions	CPI	Cycles	Instructions	CPI
4	7253	5039	1.4359	4618	2967	1.556
8	12801	8439	1.517	8962	5367	1.670
16	21401	13703	1.562	15730	9111	1.726
32	28617	18087	1.582	21586	12375	1.744

Table 1: Comparison between Simple and Optimized versions.

As table 1 indicates there is a significant improvement after optimizations. For a `TAP` of 32 the CPI speedup is 10.24% and the number of cycles is reduced by 32.6%.

[1]Since 64 is not divisible by 3 some instructions were added at the end of the loop