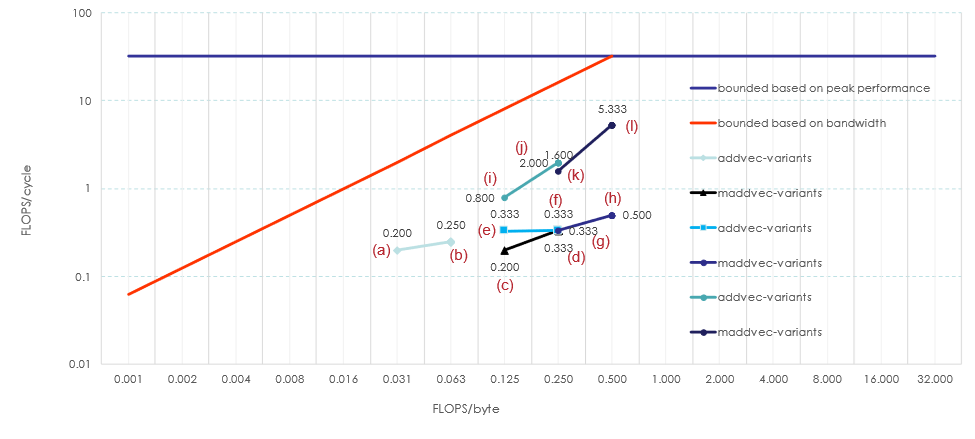
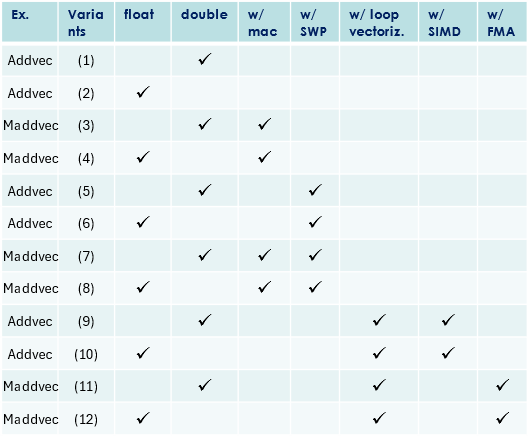
Laboratory Tutorial #2 – PART-II

Roofline Model

* *Label each point (a)…(l) in the Roofline model (below) with the variant (1)…(12) of the table (also below)*





So, let’s make the connections between points and variants:

|  |  |  |
| --- | --- | --- |
| Points in Roofline Model |  | Variants |
| (a) |  | (1) |
| (b) |  | (2) |
| (c) |  | (3) |
| (d) |  | (4) |
| (e) |  | (5) |
| (f) |  | (6) |
| (g) |  | (7) |
| (h) |  | (8) |
| (i) |  | (9) |
| (j) |  | (10) |
| (k) |  | (11) |
| (l) |  | (12 |

Laboratory Tutorial #3

Code Transformations and Optimizations

Let’s fill the table below in a collaborative way with your colleagues.

**Table III. Examples of gcc compiler optimizations.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Gcc optimization** | **Brief description** | **Possible parameters? (if yes, which ones?)** | **Identify in which optimization option, i.e., -=, -O1, -O2, -O3, -Ofast, -Ox, the optimization is included (e.g., check via gcc -Q --help=optimizers)** |
| -floop-unroll-and-jam |  |  |  |
| -ftree-loop-distribution |  |  |  |
| -floop-interchange |  |  |  |
| -funroll-loops |  |  |  |
| -funroll-all-loops |  |  |  |
| -ftree-loop-vectorize | perform loop vectorization on trees, so it does operations on multiple elements of an array at the same time inside the loop | No | O2, O3, Ofast |
| -fno-tree-loop-optimise | It disables any loop-specific optimizations, like loop vectorization, loop untrolling, etc. | No | O0, Og |
| -fmove-loop-invariants | Remove redundant calculations from a loop, that are the calculations that produce Always the same reasult because they don't depend on variables that change within the loop body | No | O1, O2, O3, Ofast, Og, Os |
| -ffast-math |  |  |  |
| -funsafe-math-optimizations |  |  |  |
| -fcrossjumping flag |  |  |  |
| -fcse-follow-jumps |  |  |  |
| -fguess-branch-probability |  |  |  |
| -fno-guess-branch-probability |  |  |  |
| -ftree-ccp |  |  |  |
| -ftree-bit-ccp |  |  |  |
| -finline-functions | Consider all functions for inlining, even if they are not declared inline. The compiler heuristically decides which functions are worth integrating in this way.  If all calls to a given function are integrated, and the function is declared static, then the function is normally not output as assembler code in its own right. | No | -O2, -O3, -Os. Also enabled by -fprofile-use and -fauto-profile |
| -fno-inline-functions | Do not expand any functions inline apart from those marked with the always\_inline attribute. Single functions can be exempted from inlining by marking them with the noinline attribute. | No | This is the default when not optimizing |
| -fipa-icf | Perform Identical Code Folding for functions and read-only variables. The optimization reduces code size and may disturb unwind stacks by replacing a function by equivalent one with a different name. The optimization works more effectively with link-time optimization enabled. | No | -O2 and -Os |
| -fipa-vrp | Perform inter-procedural propagation of value ranges | No | -O2 |
| -fipa-cp |  |  |  |