The NVIDIA compilers `nvcc` and `ptxas` leave the programmer with only very limited control over register allocation, register spills, instruction selection, and instruction scheduling. In theory a programmer can gain control by writing an entire kernel in W.J. van der Laan’s `cudasm`, but this requires tedious, error-prone tracking of register assignments. We have built a higher-level assembly language that allows much faster programming while providing the same amount of control over the GPU.

Decades of advances in the design of optimizing compilers have reduced, but have not eliminated, the need for some performance-critical functions to be written in assembly language. We have built a new assembly language `qhasm-cudasm` for programming on NVIDIA’s Tesla-architecture GPUs.

We have used `qhasm-cudasm` successfully to produce highly optimized code for a major cryptanalytic computation, the “ECC2K-130” computation, an order of magnitude larger than the recently announced RSA-768 factorization. This computation has already kept thousands of CPU cores busy for several months; the addition of several GPU clusters running our code has drastically reduced the overall expected time for the computation.

We use G200b GPUs in low-cost GTX 295 graphics cards for development. We have also run our software without trouble on the T10 GPUs (in Tesla S1070-500 units) in TeraGrid’s Lincoln cluster, similar GPUs in the NCF/SARA cluster, and the FX 5800 GPUs in TeraGrid’s Longhorn cluster.