

qasm-cudasm

Programming CUDA in assembly language
using qasm and cudasm

Ruben Niederhagen

May 4, 2010

- Why do we want to use assembly language for programming graphic cards?
 - nvcc does not always do a good job
 - we want high speed tuned code

- Why do we want to use assembly language for programming graphic cards?
 - nvcc does not always do a good job
 - we want high speed tuned code
- Why not use ptx?
 - ptx code will be heavily modified by nvcc, it is not the code that is executed on the hardware!

- Why do we want to use assembly language for programming graphic cards?
 - nvcc does not always do a good job
 - we want high speed tuned code
- Why not use ptx?
 - ptx code will be heavily modified by nvcc, it is not the code that is executed on the hardware!
 - nvcc does not always do a good job
 - we want high speed tuned code

- Why do we want to use assembly language for programming graphic cards?
 - nvcc does not always do a good job
 - we want high speed tuned code
- Why not use ptx?
 - ptx code will be heavily modified by nvcc, it is not the code that is executed on the hardware!
 - nvcc does not always do a good job
 - we want high speed tuned code
- Why not use cudasm?
 - keeping track of registers is painful
 - preprocessing makes code better readable

- Why do we want to use assembly language for programming graphic cards?
 - nvcc does not always do a good job
 - we want high speed tuned code
- Why not use ptx?
 - ptx code will be heavily modified by nvcc, it is not the code that is executed on the hardware!
 - nvcc does not always do a good job
 - we want high speed tuned code
- Why not use cudasm?
 - keeping track of registers is painful
 - preprocessing makes code better readable
- Why use qhasm?

Why use qasm?

qasm offers:

- a unified syntax for all platforms
- full control over assembly code (each line in qasm is one instruction in assembly)
- powerful register allocation
- alongside with m5 it offers a powerful programming tool set

```
low32 pos
...
syncthreads                bar.sync.u32 0
pos = tid << 2              shl.u32 $r13, $r0, 2
pos += r                    add.b32 $r13, $r13, $r9
x0 = g[pos]                 mov.u32 $r14, g[$r13]
```

- qasm:
 - Curve25519: new Diffie-Hellman speed records
 - New AES software speed records
 - Fast elliptic-curve cryptography on the Cell Broadband Engine
 - ECC2K-130 on Cell CPUs
- qasm cudasm:
 - ECC2K-130 on GPUs