COSC 530, Fall 2013, Exam 2 Study Guide

Chapter 3 Concepts and Problems

Learn to identify, name, and work with/around data hazards: RAW, WAR, and WAW. See exercises in Appendix C such as C.1 page C-82 and Figure C.24.

Learn to work with 5-stage RISC pipeline (IF, ID, EX, MEM, WB) and its extensions. Be familiar with pipeline diagrams such as the one in Figure C.38 and in Exercise C.1 page C-82.

Learn how to map RISC-like assembly into pipeline diagrams like the one in Exercise C.12 page C-86 or slide 7 (on the left) in Chapter 4 slides.

Know how to rearrange the code to better fit the pipeline as shown in slide 17 and 25 from Chapter 3 and in the book in Figure C.14.

Chapter 4 Concepts and Problems

Know how to transform symbolic vector operations or C code to vector assembly (you can use VecLoad, VecAdd, etc. for vector RISC instructions). For example, to turn Y=a*Y+X into vector assembly see slide 7 in Chapter 4 slides.

Learn how to turn a generic loop into a vector friendly loop. For an example, see slide 14 in Chapter 4 slides where it is done for a 64-element vector RISC.

Familiarize yourself with simple CUDA code like the code shown in slide 29 of Chapter 4 slides. What does `fnc_daxpy<<<x_threads, y_threads>>>(n, ...)` mean? What do __device__ and __host__ keywords do? How to take a loop over n array entries and map it to GPU? What about doubly nested loop? Here is an example:

```
for (int i = 0; i < n; ++i)
     for (int j = 0; j < n; ++j)
          y[j] = a * x[i] + y[j];
```

we can map it as follows to function `muladd()`:

```
__host__ muladd<<<mblocks, nblocks>>>(n, a, x, y);
__device__ muladd(int n, double a, double *x, double *y) {
     int I = blockIdx.x + threadIdx.x; j = blockidx.y + threadIdx.y;
     if (i < n && j < n) y[j] = a*x[i] + y[j];
}
```

Note that we don't worry much about performance of memory accesses (coalescing) and the exact meaning of all the syntax. Focus on multiple levels of threading.

Learn to analyze and remove loop dependences such as done in slide 44 (rewrite of loop body) and 48 (renaming) in Chapter 4 slides.

Know how to parallelize reductions such as sums, max/min etc. as was done on slide 49 in Chapter 4 slides.