ECEn 528
Study Guide – VLIW and SIMD

- Read Section 3.7 of H&P
  - Things to focus on
    - Nothing in particular

  - Clarifications
    - They really like to put stuff in Appendix H! Here's a few quick summary bits about the IA64 architecture
      - It was a VLIW machine which grouped 3 instructions into what was called a bundle. A bundle was 128 bits: 3 instructions of 41 bits and 5 bits of extra decoding and scheduling information.
      - To make issue easier, the compiler had to tell the compiler whether the instructions in the bundle had any data dependences with respect to each other and w.r.t. the bundle before them. If there were, a "stop marker" was inserted which would say to the HW, "don't issue the instructions after the marker in the same cycle as the instructions before the marker". These markers were encoded in the extra 5 bits.
      - Instructions from two bundles could be issued in each cycle, allowing up to 6 instructions to issue. There were lots of functional units, so there was some flexibility in what kinds of instructions could issue together.
      - The floating point units were particularly interesting: fpadd and fpmult both took 4 cycles. There was no divider; instead there was a reciprocal approximation instruction that gave an estimate of the reciprocal of the divisor; then the programmer would use Newton-Raphson iteration to converge to the quotient.
    - There's a lot more to be said about static scheduling and hardware support for it. See Appendix H if you are interested.

  - Answer the following questions:

1. What techniques can be used in conjunction with a VLIW machine to better fill the instruction packets and thus improve performance?

Loop unrolling, scheduling within a single larger loop body

2. What happens in a VLIW machine if there is a true data dependence between instructions in the long instruction word? What about anti-dependences? Hint: you'll need to think about it here and just make a guess at what needs to be done; the book doesn't tell you!

No ordering, so true dependences must be in separate bundles; no anti-dependences because it's in-order

- Read Section 4.3 of H& P
  - Things to focus on
    - The roofline model
  - Clarifications
    - Intel likes to pretend that they invented SIMD extensions. They were 3rd (shipping in 1997) beaten by HP (MAX for PA-RISC architecture in 1995) and Sun (VIS for SPARC in 1995). Other SIMD sets: AltiVec for PowerPC (1998), NEON for ARM, 3DNow! for AMD's x86-64 (1998), MDMX for MIPS. The Cell processor's SPUs are SIMD as well.
    -
  - Answer the following questions:

1. Problem 4.9 (a)

floating point operations per iteration = 4 mults + 1 add + 1 sub = 6
memory references per iteration = 4 reads + 2 writes = 6 references = 24 bytes
Arithmetic intensity = 6/24 = 0.25

2. Problem 4.12 (a)-(d). For (c), state the floating-point performance above which the kernel becomes memory-bound. For (d), assume that the 85 GFLOP/sec is single-precision performance.

a) (1 add + 3 subs + 2 mult + 2 div) / (40 reads + 4 writes) = 8/44
b) IDx is indexed, making SIMD more complicated, but possible. Ex, dH1, dH2, Ca, and Cb, could all be SIMD or vector registers
c) (8/44)*30E9 = 5.4 GFLOPS, so assuming the processor has a higher FP performance, it is likely to be memory bound
d) (85 GFLOP/s) / (4 bytes / FP) = 21.25 FLOPs/byte

3. In the roofline model, if the peak memory bandwidth of a system increases, which way does the "ridge point" move?

To the left