Hansu Kim

1. 7 Layers
   1. Problem
      1. Things that computers will be or should be able to deal with
   2. Algorithm
      1. How the system is going solve the problem
   3. Program
      1. The thing that allows the user to do what they want to do
   4. ISA
      1. The thing that the computer understands what to do
   5. Micro-architecture
      1. The way the cpu translates the instructions into actions.
   6. Circuits
      1. The thing that allows different voltage levels, 1s and 0s, that is the most basic level. “language of electron”
   7. Electrons
      1. Raw horse power of the chips.
2. Microprocessor from 1971 evolved by creating smaller transistors and larger chips, therefore following Moore’s Law, doubling size every 2 to 3 years. 71 to 92, went from 2300 transistors to 3.1Million transistors, 2013 5billion transistors. 106KHz, to 66MHz, to 5Gz.
3. As time went on, development moved forward, Cache has increased exponentially and microprocessor has increased a bit. At one point developed duel, to double processors, then quad.
4. Branch predictor, and now multi core chips.
5. Cache is smaller and faster memory that allows the processor to reduce the time and energy to access memory
6. No, too many cores and processors on the chips consuming too much energy. Limited availability due to limited bandwidth.
7. The cores on the chips will be different, doing different processes, Heterogeneous. And different depending on the task.
8. We need to break the layers, between problem to the electrons, and we do that with education.
9. More than one interface, “Organic” run time system, and ILP cores.
10. Either code is written in a higher language and is not that optimal for systems, or is written in a more computer friendly language and can be optimized per system. To write in a higher language,it is easier for people to understand and test, taking less time and resources to finish products. However, it may contains bugs and optimization issues with other architectures or systems. On the other hand, writing in MIPS or in assembly, it is more a direct language to the architecture, but is a lot more difficult to understand. Furthermore, it is time consuming and unique per architecture. Demanding heavy weight cores.