1. **What are the key concepts introduced in the lecture? Which one is attracting you the most? Why?**

   (a) Some assumptions in computer architecture, such as Moore's and Amdahl's Law or Dennard Scaling, come to an end: What this means is that software and hardware should not be viewed as separate concepts for performance improvements in this era, if we want to step further and further to develop more efficient computers and meet the needs of this and future periods in terms of computers.

   (b) A consequence regarding (a), parallelism is now more important: A fruit of the marriage of software and hardware.

   (c) The importance of considering security on the basis of computer architecture: A quote from John Hennessy is also crucial, he says that if airplanes malfunctioned as often as computers malfunctioned nobody would be at this conference who didn't live in Southern California. But once upon a time, security was perceived as only a software problem.

2. **Which part of the lecture is clear and not clear for you?**

   I understand, although not completely, the first part which they mentioned the historical development of computer architecture. The some of terms are strange to me for taking the architecture course 1^{st} time. But also what the clear to me is why RISC architecture is distinguished and glared from other architectures, one of them CISC.

   I could not understand how security could be handled from the computer architecture perspective and where open-source architecture design would fit in this context.

3. **What did you like or dislike about the lecture?**

   They, John and Dave, emphasize interactions between software and hardware intensely. I like it that's because we talk about software, developing software, or abstract stuff. But these run on an electronic device and must correspond to concrete notions to implement abstract things.

   I remembered while watching Turing Award Lecture of John and Dave, in an interview between Oğuz Ergin and Bilgem Çakır, I really liked the quote from Mike Acton by Bilgem Çakır: "Software is not an object floating in the ether. Software is something that runs on actual physical processors and memory."

4. **What is the question you want to ask to the lecturers? Why?**

   What solutions can we offer for progress when the laws we mentioned lose their effect completely? The reason I ask this question is that these laws and/or acceptances have been in our lives for roughly 50-60 years, that is, since microprocessors have existed. We are now coming to the end of these laws and I think we may need other methods or inventions for long-term solutions.