

Study Questions for Patterson & Hennessey, Sections 2.15–18

On Wednesday, laboratory homework #2 is due, but you are still responsible for the readings.

Section 2.15

This section is intimidating if you have not studied pointers in C, since Java purposely doesn't have pointers. Here is the only thing that you need to know, a point made when we were discussing Section 2.13:

When calculating an entry in an integer array, it is always easier when possible to add 4 to the previous subscript than to calculate the subscript from scratch by a multiplication by 4 and then an addition.

Section 2.16

This section, too, is intimidating if you think I require that you know it all. Skim it just quickly enough to be able to answer the following question:

What are some evidences that the Intel 32-bit Architecture (IA-32) chip is a Complex Instruction Set Computer (CISC), and not RISC?

In particular, take careful note of Figures 2.40 and 2.45.

Although the history of Intel's attempt to maintain backward compatibility while still introducing some RISC concepts, as described on pages 135–136, makes interesting reading, you will not be tested on it. It will be especially interesting to those of you who wonder whether you'll be getting maximum performance out of Intel chips in the future, whether you should buy the cheaper AMD chip instead of the pricier Intel chip, and what Intel is doing in the area of graphics/multimedia on a chip.

Section 2.17

Fallacy #1 reminds you that RISC is better than CISC, for all of the four reasons summarized in the next section.

Fallacy #2 reminds you that there are people who write compilers who are more clever than we are at turning high level code into assembly language.

Pitfall #1 is to remind you always to add 4 as you step by words through byte-addressable memory.

Pitfall #2 is to remind you that you may put things on the stack, but after you leave the procedure you're in, the stack will be cleared, so don't count on the data still existing.

Section 2.18

Review the four design principles that were sprinkled through the text. If you need their page sources, they are

1. Simplicity favors regularity, p. 50.
2. Smaller is faster, p. 53.
3. Make the common case fast, p. 58.
4. Good design demands good compromises, p. 63.

Be able to give an example of each principle in use in the design of the MIPS CPU.