Teaching and Teaching Computer Architecture: Two Very different topics (Some Opinions about each)

Yale N. Patt
teacher, The University of Texas at Austin

Abstract

This year's Computer Architecture Education workshop is remarkable in its recognition that to teach computer architecture well, one has to pay attention to two things (a) teaching and (b) computer architecture. Having been doing both for a good number of years, I harbor a fair number of opinions on what one should do and what one should not do with respect to each. This talk will get into some of those opinions. With respect to teaching, I will discuss some of my Ten Commandments of Good Teaching, what I think of distance learning, political correctness, emphasis on memorization, the inability of American students to write English, the value of having students study in groups, and what I feel is often the sad misuse of technology. Most importantly, I will discuss my motivated bottom up approach to learning. With respect to teaching computer architecture, I believe the single most important point to get across is that computer architecture, if it is a science at all, is a science of tradeoffs. The student is best served if he/she thoroughly understands the fundamental principles so as to be able to make the appropriate tradeoffs in reaching a particular design objective. I also plan to discuss the use (and too often, misuse) of measurements, simulation, and real ISAs as opposed to concocted ISAs.

1 Introduction

I have been asked to provide copies of the transparencies I will use in my Keynote Address. I have added some annotated text to hopefully provide some context.

Knowing from past experience that the dynamic schedule of my talks usually bears only casual resemblance to the static schedule that I prepared ahead of time, I provide these copies somewhat timidly.

Figure 1 is from a talk I gave last fall at the annual Visions Lecture of the Computer Sciences Department at The University of Texas at Austin. The subject was Education. I was asked to give my dream for an ideal future with respect to education.

- Distance Learning produces better education, not cheaper education
- We pay teachers enough that those who would opt for this career don't opt for medical school instead
- We teach high school English teachers enough English that students at the University can write two consecutive coherent sentences
- We get past this insane preoccupation with political correctness, so we can get on with the business of teaching and learning
- We stop canonizing the use of high tech education. Bad pedagogy is NOT good pedagogy if draped in technology
- We stop rewarding memorization ability, so maybe students will learn to think,...and perhaps understand

Figure 1. Visions (Re: Education)

The remaining figures deal with the two parts of my talk, a focus on teaching, and a focus on teaching computer architecture.

2 Focus on Teaching

Teaching involves at least three things: how to teach (Section 2.1), what to teach (Section 2.2), and what aids to use in the process (Section 2.3).

2.1 My Ten Commandments of Good Teaching

Someone suggested I come up with a set of commandments for good teaching. For historical reasons, they thought ten would be a good number. So, I set out to do it, and came up with nine. Ergo, note the tenth one. On sober reflection, that one in itself is a tenth command-

ment. So, the list on Figure 2 really does contain ten, not nine as some have commented.

- Know the material
- Want to teach
- Genuinely respect your students and show it
- Set the bar high; students will measure up
- Emphasize understanding; de-emphasize memorization
- Take responsibility for what is covered
- Don't even try to cover the material
- Encourage interruptions; don't be afraid to digress
- Don't forget those three little words
- Reserved for future use

Figure 2. My Ten Commandments of Good Teaching

2.2 Emphasis on the Fundamentals

My emphasis on teaching the fundamentals has been part of me forever. No doubt my PhD students get tired of hearing about it. For example, I believe that an appropriate PhD qualifier is not a written test on some advanced course in the graduate curriculum, but rather an oral exam on the fundamental concepts found in relevant senior level undergraduate courses.

My view is simple: research, development, problem solving are all about breaking problems down into small pieces and working with the small pieces until the

- Top-down design, Bottom-up learning for understanding
- Abstraction is vital, but...
- Not bottom-up, but "motivated" bottom-up
- Engineering is about DESIGN, first understand the components
- From Concrete to Abstract (Dijkstra notwithstanding)
- Cut through protective layers
- Memorizing is not understanding
- Students do better working in groups

Figure 3. Some thoughts on what is important

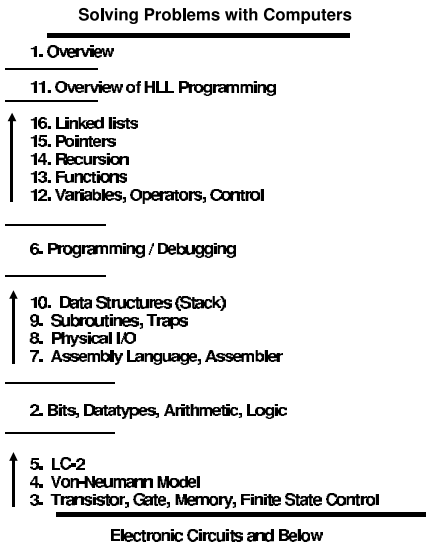

Figure 4. My motivated bottom-up approach

"Aha!" happens. How well someone can do that depends on how well that person has mastered the fundamentals.

My views crystallized particularly strongly with the development of our "new" introduction to computing, which we first developed at Michigan in the mid-90s[1] and later turned into a textbook, published by McGraw-Hill[2]. My view of what is important is expressed in Figure 3. More specific detail of the motivated bottom-up approach is shown in Figure 4.

2.3 High Tech in the Classroom

There seems to be a preoccupation with using technology in the classroom. Certainly, there is much that can be done with technology to improve learning. I am concerned that in our leap to technologize everything, we are developing some very bad pedagogy under the umbrella of "using technology." Figure 5 shows some common uses. Figure 6 lists some serious concerns.

3 Focus on Teaching Computer Architecture

We are lucky. We get to teach computer architecture. Some will tell you that computer architecture is dead, that the microprocessor is to computing like a brick is to buildings. Wrong. Computer architecture is the interface between what technology can provide and what the marketplace demands. Technology continues to provide more and more. We are told that within a very few

- Email
- Web site
- Power Point
- Document Reader
- Animations
- Plato, vintage 2003
- Clever attendance mechanism
- Other bookkeeping
- Text+Voice (WOW Factor, see Shriver's CDROM)[3]

Figure 5. Some uses of high tech

- Baseline Power Point
- Cost
- Extemporaneous Effect
- Visual/voice disconnect
- Attendance vs. Participation

Figure 6. Some caveats associated with using high tech

years, each chip will contain more than a billion transistors. And the marketplace continues to demand more. In fact, the higher and high performance chip becomes an important enabler. As we develop more, the marketplace dreams of more things it needs.

Contrary to being dead, computer architecture is in a constant state of high volatility. The state-of-the-art examples we studied yesterday are boring today.

Computer architecture will always be alive and healthy as long as people continue to dream up new needs for our future products. The design points may change. Not just higher performance, but higher reliability, availability, cheaper cost, and more power-sensitive designs, for example.

Within that framework, what do we teach. In my view, we focus on three things: the fundamental principles (which do not change, or change very, very slowly), the tradeoffs that always result, and the concrete implementation of those principles.

Figure 7 identifies a number of the fundamentals, Figure 8 (levels of transformation) and Figure 9 (three parts of a microarchitecture) elaborate on two of them. Figure 10 lists some concerns.

- The transformation hierarchy
- Three parts of a Microarchitecture
- The DSI
- IPC vs. cycle time
- Partitioning

Figure 7. Some fundamentals of computer architecture

- Problems
- Algorithms
- Programs
- ISA
- Microarchitecture
- Circuits
- Devices

Figure 8. Levels of Transformation

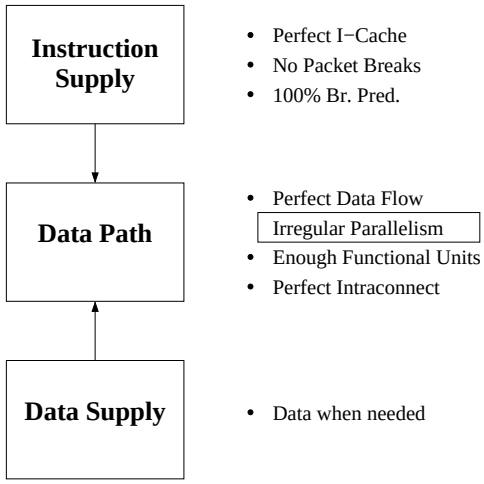

Figure 9. The Microarchitecture

- Focus on Measurements
- Use of Simulation
- Real ISA vs. Concocted ISA

Figure 10. Some concerns

- the new data path
- internal fault tolerance
- asynch and synch co-existing
- different cycle times for different functions
- SSMT (aka helper threads)
- Block-structured ISA
- uarch support for CAD
- greater use of microcode
- greater impact of the compiler
- compiler/uarch communication

Figure 11. The Microprocessor ten years from now (perhaps)

Finally, because a course in Computer Architecture is not only about what is, but also about preparing the student for what will be, it should also give our best current guess into the future (see Figure 11).

References

[1] Y. N. Patt. The First Computing Course for CS, CE, and EE Majors at Michigan. In *The Interface*, pages 1–3, November 1998.

[2] Y. N. Patt and S. J. Patel. *Introduction to Computing Systems: From Bits and Gates to C and Beyond*. McGraw-Hill, 2001.

[3] B. D. Shriver and B. Smith. *The Anatomy of a High Performance Microprocessor: A Systems Perspective*. IEEE Computer Society Press, 1998.