Reflections on Computer Organization

I guess everyone would accept that the computer has changed our habits and thoughts as very few instruments have done, in fact the very course of human civilisation. Yes, the computer is an exceptional device. But far more exceptional than the computer *as a device* is the computer *as an idea.*

Many years ago E W Dijkstra stated this as a challenge and a vision: *In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity for intellectual challenge, they are without precedent in the cultural history of mankind.*

Now, what is this challenge? What is it that makes a computer so special, what makes a computer essentially different from other technology like a wrist watch, a washing machine, a car? Simple things like a pen, non electronic things like a violin, all these are technologies. What is it that makes a computer different? If we want the essential characteristics and not merely incidental or superficial ones, we need to proceed from first principles. To start truly from first principles we must examine the notion of

1. “First Principles” from First Principles

The most common use of the phrase ‘first principles’ at least in the western tradition, are

1.1 Logical First Principles

Here one postulates a very few axioms that are hopefully (a) atomic (b) obvious and (c) complete. This tradition is at its greatest in Euclid who, starting from 5 axioms most of which were incontestable and trivially obvious, built the whole grand edifice of geometry. But by the turn of the 20th century this is in a shambles. Euclid collapses. Hilbert the voice of European desperation seeking solid foundations. Godel smashes Hilbert’s dream. A young student, Alan Turing, impressed by Godel’s theorem but not too happy with his philosophy, envisages a machine towards countering Godel – a universal machine. The idea of the computer is born though Godel remains invincible. But this is a story for another time.

Suffice it to say here that the conception of the modern computer directly follows the deep and difficult philosophical activity of investigating first principles. This interrogation reduces to the following question: *In my concept hierarchy, which are the primitive concepts and which derived?* Or more simply: *What comes first logically?*

If we remove the *logically* epithet above we get

1.2 Historical First Principles

The question here is simply: *What came first?*

Why should one consider historical sequence?

Much of the hoopla of today has a core of ideas that was discovered/invented 30,40 years ago. If we know some history, we are not conned. eg garbage collection in java, XML as a universal data rep language can both be found in LISP. As Wadler says: *XML is Lisp’s bastard nephew, with uglier syntax and no semantics.* At a slightly more personal level Java’s portability via an abstract machine was there in the Pascal compiler which Nori wrote (30 years back?). Hence my irritation at the Java hype!

Those who forget the past are condemned to repeat it. [@Who said this?] Some facets:

1.2.1 Accidents of History For almost 40 years after the invention of the computer, the computer community could be split into ‘scientific’ and ‘business data applications’. However the first computers were made for scientific (so-called) applications during the world-war. That was the reason for the name *computer.* If the data processing guys had got there first the name would have been very different, perhaps *datamaton* as recommended by Peter Naur. Now the name computer is there to stay but we should know it to be a misnomer: *datamaton* describes its significance more accurately.

1.2.2 Prisoners of History Most of us think in the terms we were taught to think in. And when revolutions happen, our thinking becomes obsolete. Now the computer revolution happened before (most of us) were born. So we feel we are ‘with it.’ We forget that after the computer revolution there were at least two more: the PC revolution and the internet revolution. If we were educated more than 10 years ago, it was before these two revolutions but even now the CS education being given out is usually obsolete. For example, 20 years ago, CPUs of course existed and there was the UNIBUS of PDP-11 but CPUs were studied, buses were not. This was ok then because there were no competing and complementing bus standards/protocols. Today we continue the 20 year old habit of studying CPUs but not buses. But is Pentium or SPARC architecture more important or PCI or USB? So when our CS education seems most current and topical it may be most obsolete.

1.2.3 "Progress" may be Degeneration The first book on data structures – Knuth – is still the best and in CO, Bell, Newell, Siewiorek (70s) to Tanenbaum (80s) to Hennesey Patterson (90s) is a steady decline.

1.2.4 The Pedantry of the A Priori When TCP/IP was invented it was just an implementation – a 'hack'. OSI was the proper thing for respectable people to study. History proved otherwise. Another classic instance is the 1992 argument between Linus Torvalds and Andrew Tanenbaum in which Tanenbaum famously declared, "Linux is obsolete!" and gave mighty reasons for this declaration. Sometimes we teachers (pedants!) need to open our eyes to see whether the world is listening to our commands. Studying such history may save some embarrassments.

So we study history to be free from it!

1.3 Structural First Principles

Now the question *What came first?* can be asked literally or somewhat metaphorically. For example earth comes before bricks and cement which come before walls which come before buildings. This kind of questioning leads to 'Structural first principles.'

Note that the metaphorical nature of this question results in a dispute. One order is earth, bricks, walls, buildings but another is owner, builder, architect, engineer, mason. Observe that the two orders are almost diametrically opposite. Which is the correct one?

Depending on your answer you will be a 'bottom-up' (reductionistic) or 'top-down' (wholistic) person. Perhaps the two indefinite terms 'computer organization' and 'computer architecture' should be distinguished by saying that the bottom-up view is structure or organization, whereas the architecture is the wholistic view.

What is more important is for a CSist to be able to function in both modes with equal ease. Computers are (if understood non-physically) the purveyors of machine language which is the basis for assembly language which is the basis for C which is the basis for C++ Haskell and such HLLs – the Tanenbaum layering. But for the purposes of teaching the best order is (almost) the reverse: Haskell, C, assembly. Which brings us to the next two views.

1.4 Linguistic First Principles

In ages past human-beings had more dignity – they were the chosen ones of God, made in the image of God, sometimes they were even the collaborators with God or identical with God, at the least they had a soul. Today none of these edifying views seems to hold any more; we are merely the grandchildren of apes and our only hope is to desperately keep 'progressing' whatever that may mean. But if we ignore a degrading past and an imaginary future and ask, "What are we *now* that distinguishes us from other living beings?" There remains one outpost of hope:

The mystery of language

In the study of language there are two names that dominate in the last 50 years: Chomsky and Whorf. Chomsky is a universalist, where Whorf revels in relativity. Chomsky established the grand temple of language where all humans testify to their humanity, Whorf made us more humane by teaching us to enjoy differences. The question that these two linguists together make us ask is: "Does language merely relfect the world or do our talking and hearing create our world?" More simply: *Does the world come first or our language?*

Whorf pointed out that what we call our science is just the elaboration of our common sense and this in turn follows from our language. He shocked the scientific community by showing that language shapes our thinking, our life, the world we live in more than we can imagine. That he continues to be abused 50 years down the line proves that he still continues to shock.

The following draws from http://modern-thinker.co.uk/5%20-%20Benjamin%20Lee%20Whorf.htm and http://www.enformy.com/ahlford@

> Before Whorf, Western thinkers assumed that the use of language merely followed rational and intelligent thinking. Thought was supposed to depend on laws of logic or reason, and in turn these laws were supposed to be the same for everyone, no matter what language was used. Whorf points out that this view of language was universal in the West because no one knew of any exceptions to it.
>
> As a small example, in English there are two words *wife* and *woman* whereas in French *femme* does the job of both. Not surprising then that a Frenchman distinguishes wife and woman less than an Englishman does.
>
> Language has two aspects: vocabulary and grammar. The function of grammar is to indicate relationships whereas the vocabulary indicates things and hence creates identity. Indo-European languages analyse reality into two sections: there are 'things' (that is, nouns) and there are their attributes, or what they are or what they do, 'subject-predicate' languages.
>
> One language that illustrates a marked difference from Indo-European languages is Hopi which analyses reality mainly in terms of events rather than nouns ie things. The civilizational significance of this difference is profound. *Nouns (and hence names) give an identity.* If things – nouns – are not important to a person, then they are not likely to feature prominently in that person's understanding of the world.
>
> Western languages, with their thing-attribute division, can be thought of as being object-oriented. Whereas languages like Hopi are process-relation orientated. The most significant difference between these two orientations is over the issue of identity.

What does object-orientation mean?

An apple is not an mango.
If an apple is here it cannot be there.
If an apple is here, a mango cannot be here. etc.
A word and its meaning are completely unrelated except by convention. (A rose by any other name would smell as sweet). At the other extreme are sacred languages like sanskrit wherein a mantra, its vibrational effect and the deity it invokes are inseparable.

Computers, while not as distant from object-orientation as sacred languages, are nevertheless quite un-object-oriented. eg. Things sitting next to each other in memory may be 'logically' very far apart and conversely. A word may be an integer or a pointer or a character. It may even be a pun and be all 3. Above all a program and its data are indistinguishable and in fact a program may treat itself as its data.

We are now in a position to give a first approximate answer to Dijkstra's challenge: *Investigate how computers will change not just the outer forms of our lives but their intellectual content as well.*

I would like to conjecture that a civilization that draws its analogies and inspirations from the computer (as Newton drew from clocks) will be drastically different from the western civilization of the last 400 years. If we treat the computer merely as a device it can only intensify the object-orientation of our lives but if we also see the mind-bending ideas it engenders, maybe it reduces? Key to getting out of OO thinking when dealing with a jazzy object of modern culture like the computer is to see it not primarily as thing with 'powerful' attributes but as a 'languager.'

1.5 Pedagogical First Principles

A child is a classic languager. It starts out dumb and bawling and gets articulate with the passage of time. This transformation could not happen if the child did not keep learning and the parents/teachers keep teaching.

Now a teacher will say that the first principles one must study are abcd. Obviously for teaching, pedagogical first principles need to come first. In other words a teacher must teach in a sequence which is most efficient for the purposes of teaching. You cant effectively teach the foundations of mathematics to a student who does not know any mathematics. The question here is: *In what order should this subject be taught/learnt?*

However it is the business of a teacher who is committed to not mass-producing slaves for the software (or any other) industry to open the students mind to the other types of first principles. eg Rajaraman taught hvs (mid 60s) assembly as the first programming language and then taught FORTRAN as 'advanced' and high-level. But then someone somewhere (where?) introspected and said FORTRAN is more first principles and assembly is more derived and so high level programing became programming and assembly programming became low level programming. Unless someone sits and reflects philosophically these paradigm shifts cant happen. If we want to make students who are not merely thrown around by these paradigm shifts but can envision and create these shifts themselves the study of all these forms of first principles is essential.

1.6 Psychological First Principles

Which of these different forms of first principles has primacy? Which is first? A logician will say logical, a historian will say historical, an engineer will say sructural, a teacher will say pedagogical. What should we say? Let me be more direct. What do I say?

I say 'I'.

Let us (let me!) elaborate.

- No sentence can be without a subject although the object is optional.
- Of all subjects, the first person comes before the second and third.
- Grammatical cases like 'me' 'my' 'mine' can only be defined with respect to the first case 'I'.
- I must say 'I' before I say anything else. Look at this sentence or the first sentence of the article.
- I imagine that I am a minute particle in a vast universe but in fact space is an intellectual abstraction consequent on the operation of my senses. Since space is in my mind and my mind is in me, space is in me.
- Likewise I imagine that I was born and will one day die but in fact the tenses of past and future are relative to the present – to Now. It may seem paradoxical but the present is prior to the past and future because past and future need memory and imagination ie mind to exist. But for anything including mind to exist it must exist now.

In short the most absolute statement I (or any being that says I) can make is:

Here Now I Am.

The reason why we vigorously object to this is that we are all overwhelmed by our sense of limitation. On investigation, we find that this sense is just a sense, an assumption. The solution then is this investigation, the core question is:*Who am I?*

Why is this question more basic than all the others?

Who am I? is the most basic question because our concepts are in the mind and the mind is in us. It is prior to structure questions because structure is based on the assumption of an existing world order but the world is our construct as Whorf showed. It comes before *What came first?* because time is in us and not the other way round. And it is more basic than issues of teaching/studying order because teaching and studying is contingent to relations of teacher and student. But before I can enter into a relation with anyone as teacher or student I must exist.

> Unfortunately the above facts are so natural as be axiomatic for easterners whereas for westerners they appear not just difficult but insane. Unfortunate because the greatest of European philosophers – Immanuel Kant – said just this: When I see an object, 'I see the object' is the fact whereas the object is an imagination. In fact all 'facts' are conventional only perception is fundamental. (That most easterners today are western and increasingly westerners are beginning to see the above wisdom is not contradictory: east and west are in the mind). Paradoxical that the doctrine of absolute subjectivity of our existence is objectively the same whether we get it from Kant, Vedanta, Matrix Reloaded —

Or the study of Computers.

The idea of the modern computer is usually traced back to Turing. Now Turing's ideas were a peculiar mixture of human and machine. They did not have any physical elements of the computer as it would develop 10 years later or what it is today. But what was the itch that drove the genuis of Turing? When Turing was hardly 20 he lost a close friend Christopher Morcom. What follows is from his biography: http://www.turing.org.uk/bio/.

> Through a long crisis of 3 years his thoughts turned to the question of how the human mind, and Christopher's mind in particular, was embodied in matter; and whether accordingly it could be released from matter by death. His originality and daring came from his imagining a machine whose transitions were analogous to the 'states of mind' of a human being performing a mental process.
>
> The triple correspondence between logical instructions, the action of the mind, and a machine which could in principle be embodied in a practical physical form, was Turing's definitive contribution. Turing worked in isolation from the powerful school of logical theory centred on Church at Princeton University, and his work emerged as that of a complete outsider. One can only speculate, but it looks as if Turing found in the concept of the Turing machine something that would satisfy the fascination with the problem of Mind that Christopher Morcom had sparked; his total originality lay in seeing the relevance of mathematical logic to a problem originally seen as one of physics.

Two important conclusions:

1. People who make the most important scientific contributions, especially in CS, are those whose minds are untrammeled by existing mindsets.
2. The design of modern computers followed from the pursuit of the mystery of human consciousness. Just as *Who am I* is the most fundamental question about myself *What is a computer* is the most basic question about computers.

So, What is a computer?

The standard icon suggests that the computer is a monitor. If we accept that, it would mean that a computer is just a TV. Slightly more educated persons may point to the dubba next to (or under) the monitor as *the computer*. Is any box a computer? Academics seem to believe that a computer is a CPU. This is discussed in *Prisoners of History* above. Others may associate a computer with the motherboard.

A recent strange experience illustrates the subtleties involved.

> Ive a computer whose OS I wanted to upgrade without disturbing the existing setup. Decided to fit a new hard disk with a new OS. Installed the OS on a new hard disk, fitted the *new* hard disk into the *old* computer and rebooted. The messages that started coming were: New Hardware detected: monitor, mouse, network card etc etc. but not new disk!
>
> Strange! The only one thing new is not seen as new but all the old things are seen as new.

So whats happening? From the point of view of the OS the disk from which the OS is loaded is 'I', the rest are peripherals. Yes *from the OS point of view* it is the hard disk, from the casual users point of view it is the monitor (icon), from the point of view of the person assembling machines it is the motherboard and from the point of view of academics it is the CPU.

Perhaps the view: computer is monitor+box+keyboard+mouse connected to mains and modem/ethernet is the most balanced? Well, this is the system administrator's (or janitor's) view.

Really the only absolute in all these conflicting views is that there are different views:

Facts are conventional, Perception is fundamental.

So a second answer to Dijkstra's challenge.

For nearly 400 years the world has been under the thrall of physics: a subject is considered worthwhile – scientific – to the extent that it is physics-like. So physics is king, biology is next, sociology is inferior and psychology is not respectable at all.

Is a completely different – even opposite – view possible? The computer is based on physics but provides a behavior that is essentially logical and not physical. Maybe the study of the computer, not as a device but as an idea, will provide a bridge from a physics that is fascinated with dead bodies to the study of the self that is too difficult for us to conceive apart from the body.

In the rest of CS we study the more logical and psychological parts, in CO the more physical aspects that provide the foundation for the other.

2. Fundamental characteristics of a computer

2.1 Digital vs Analog

Take a weighing machine, a voltmeter, a record player. These are analog devices. Analog Vs digital. Discrete Vs continuous. What are these 2 dichotomies? A single variable is discrete or continuous and a function which transforms one into another is analog or digital. [@Actually 4 possibilities DD, DC, CD, CC. Which is which?]

Lets take a voltmeter or a spring-balance. What do they do? There is a voltage input which is continuously varying and if it is an old fashioned voltmeter with a meter going like this (picture for this), it also is continuously varying: the meter position is the direct analog of the voltage. Likewise in a spring balance, the position of the marker is an analog of the weight. This analog transformation need not be an identity function. The old 78 rpm records were more or less identity functions. The later (33 rpm) record players were not identity functions because the bass was reduced in the record and a corresponding correction was made which pushed up the bass and pushed down the treble because high bass would tear the groves. So there is an unequalisation made when it is put there and a correction made (the RIAA curve). This is an example of a non-identity analog function.

Is the world discrete or continuous?

The voltmeter without the markings is pure analog and almost useless. Until you put the markings, you cannot say this is 3.3 V. Now between 3.3 and 3.4 there may be no marking and there the analogness shows. There is an infinite range there but it is useless because it could be somewhere anywhere between 3.3 and 3.4; we dont know where. In practice one reads 3.3 though in principle the voltmeter has an infinite range of possibilities.

So there is this continuous tradeoff between the continuous and the discrete − the play between analog and digital. The voltage is analog, the meter is analog but the readings are shown in some digital way. In fact the digital mode is so convenient that nowadays we get digital voltmeters: just the number, no meter, no analog equivalent.

Now when we say that things like voltages and weights and currents are purely analog there is a subtle problem here. For example we may say that the current of 1 amp flowing in a wire is (say) 1 trillion electrons per sec. Now when we say 1 trillion, do we really mean exactly one trillion? If I say 1 trillion plus 1 is it more than one amp? Of course not! Depending on the least count of my meter, 1 trillion plus 1 billion may show and if it is a very very sensitive meter, one trillion plus one million may show because it is a million times more sensitive.

Of course a million is not such a big thing: eg a micrometer is million times more sensitive than a milestone. There can be different grades of sensitivity but only upto a point. Note that this is not just a technical or scientific but a philosophical problem: the electron is at the boundary of measurable things. According to the Heisenberg principle you cant really ever see an electron in same sense that you see me. It may be possible to go from 1 trillion ± 1 billion to 1 trillion ± 1 million but we cant go much beyond that. Somewhere there is the limit depending on quantum physics constants.

So when we say that 1 amp is 1 trillion electrons/second we are actually indicating a range which depends on the measuring capacity of the meter. It is this range or a tighter or looser range but it can never be a single figure. So when we go from amps to counts of electrons we are apparently back in a discrete world. This is a linguistic trap. The trap is in our language wherein we are forced to say *an electron* or *the electron* like I can say, 'I ate *an apple* today.' And then go on '*The apple* I ate was very tasty.' Underlying such words are very strong assumptions: an apple (see the language!) is clearly recognisable and distinguishable from other things and yet there is some ‘appleness’ common to all apples. An electron is not a thing like an apple.

There is a multiplicative problem here. Our language is object-oriented, physics is object science (*padartha shastra*) In the world of CO, if we take care that when we say 3 volts we mean 3 volts ± ½ volt we dont get into trouble and we can align with the ‘commonsense’ view that voltages, currents, pressures, weights are perfectly analog – continuously varying quantities. The computer is a digital or discrete device. And so our basic engineering problem is *How to build this discrete device on top of a continuous world.* Key to a solution is a convention wherein we shall associate one range of voltages with one logic value and another with the other logic value. This is called the *Digital Abstraction* and is a cornerstone of computer engineering.

One of the purposes of this book is to meditate on *the digital abstraction* through the lens of different categories. A basic thesis that we shall elaborate further is that the digital abstraction is at once deep science, brilliant engineering, effective management and sound economics. But all this would not have been possible were it not for the profound philosophical understanding of those who originally envisioned and created it. (Leibnitz, Pascal, Babbage, Godel, Turing, Zuse, von Neumann, Aitken, Eckert-Mauchly, McCarthy, Backus, Dijkstra).

2.2 Universality

The second factor that strongly distinguishes a computer from a pen or a voltmeter or a piano or gramophone is its *universality* – a profound concept that permeates every aspect of computer science. And CO is the point where we take common current technology which is used to build all kinds of *specific* devices viz electronics, and turn it round to build a *universal* device.

To understand universality we need to see in what sense these other devices are not universal. Cassette and cassette player: no possible mixup of cassette and cassette player.

Table@

2.3 Programming

But when it comes to sufficiently sophisticated cassettes (or simple players) the two become indistinguishable (@questionable): The object over which the device functions and the device itself become the same. Now at a philosophical level this is very exciting but at a pragmatic level it makes things difficult and useless because the computer as it stands is a useless device: For the computer to behave like a gramophone or a pen or a piano it has to be specialized. In its universal mode it is useless because it is just a potential to be any other device/machine you can imagine, it is not yet that machine. So a process of converting a universal machine into a specialized machine is required, a process we call programming: the third very special(!) aspect of what characterises a computer.

3. Other characteristics of Computers

The three characteristics we have seen are (1) computers are digital, (2) they are universal and to make that universality specifically usable (3) they are programmable.

Now these fundamentals have a number of implications.

3.1 Language

Everyone knows that central to CS are programming languages. However languages and mathematical notations occur in a wide variety of contexts: specification languages, program generating languages, verification logics, system adminstration shells etc. Relevant to CO are machine description languages. Bell Newell and Siewiorek pioneered PMS and ISP to describe machines whose modern indirect descendents are Verilog, VHDL etc. But in CO the ubiquitous presence of language is manifests as systems of

3.2 Coding

When we cut groves in the record that makes sounds when it is played, it is quite obvious that the groves and the sounds correspond: the groves encode the sounds but it seems unnecessarily pedantic to say this. Likewise when I get my passbook updated by the bank, it seems not just unnecessary but ridiculous to assert formally that the numbers correspond to dates and the corresponding bank balance amount in rupees. What else could they be? But when the same machine — the computer — can encode both bank balances and sounds and the text of this book and countless other concepts, the facts and the subtleties of *coding* become difficult and inescapable.

Universal to specific implies the need for *coding.*

But what is most crucial (and amazing) is the fact that this universal-to-specific transition, viz. a program, itself is encoded in much the same way as the data that the program works on.

The above is related to information theory. It may not be really necessary or relevant to do it rigorously but the general idea of coding pervades from microprogramming, nanoprogramming to RISC to XML. It is central to the structure of instruction sets. It is fashionable today to teach RISC instruction sets as though they were the only ones that exist and not teach the architectural design of instruction sets at all.

3.3 Parameterization

Other machines(technology) are also *parameterizable*. Take a car. Its basic business is to get you from one place to another but you can change other characteristics. At the least you can go fast or slow. You can turn up or down the windows depending on the climate. You can go a step further and tune the engine for increased performance or increased efficiency – but within a limit. It is only in computing that the parameterization is essentially limitless and hence limits put up in practice are usually seen (in 20/20 hindsight!) to accrue from the lack of imagination of its creators and not from any fundamental limitations.

4. Incidental Aspects

From a philosophical viewpoint this is all fine but from a practical (engineering) viewpoint this is all philosophy! Engineering is the union of technology, economics and management. Technology gives us what we can do – a positive. Economics puts pragmatic limits on these abilities – a negative. Management finds solutions – a reconciliation.

Above we have tried to see the nature of computers in an abstract (eternal) sense. However, given current technology we build computers in a certain way. Below we look at those aspects which are more dependent on current technology.

4.1 Electronics

The fact that we use electronics to build our computers follows from on the fact that I am speaking in the year 2003. 50-60 years ago the computers that were built (around WW-II) used electro-mechanical devices. One of the earliest computers, the Mark-1 had a clock speed of 4 cycles/second(!!). Aitken suggested valves then there were transistors and then ICs all within the domain of electronics and so we imagine that electronics is made by God but maybe 10-15 years from now people will use optical devices which is exciting because light travels twice as fast as electricity but is not yet workable because we dont know how to make optical devices anywhere as small as electronic devices. Note and observe that these are engineering considerations and not fundamental ones.
@Maybe quantum computers will involve fundamental scientific problems?

The point is that the use of electronics is not a fundamental requirement but a consequence of current technology. As Dijkstra said, *Whether the computer works on electronics, pneumatics or magic is irrelevant...* @exact quote?

A little more fundamental than electronics but still more managerial than technical is the question of

4.2 Timing

For some strange reason computers are always studied as though synchronous, globally timed devices are the only ones possible, whereas in fact there are four quadrants: Synchronous vs asynchronous circuits, locally vs globally timed devices. (@See Ward and Halstead)

4.3 Abstration layers

Ideally we all want to know everything about everything but in practice to make a hi-tech civilization like ours happen we need thousands of specializations and hence people need to specialize in areas of knowledge. This implies that we must be able to break down a complex discipline into many levels − the so-called abstraction levels. Tanenbaum in his book talks of some 5 levels: The digital abstraction level, the microprogramming level, the machine- language level, the OS level, and the Hll level. Actually the digital abstraction level is a way up level, it is not hardware in the literal − hard − sense at all. Amar Mukherjee breaks up digital (CMOS) design into 5 layers (1) physical (2) layout (3) circuit and logic (4) function (5) system

So there are of the order of 10 levels. The very word *discipline* has associations with the army with the 10 levels from jawan to general − all rigorously separated. So this is the general line of *computers organization*. We need to keep our eyes open to the difference between the specific details of building modern day computers and the general principles of engineering involved here viz. that the engineering of complex artifacts is as much subservient to the laws of economics as to the laws of science and the practical business of engineering an artifact is as much a managerial issue as a technical one (maybe more). Knuth (Out of their minds) says that the ability to view things at various levels is key to being a computer scientist.

Most people assume that this is good. Is it?

Economics is related to management and central to management is separating hand and heart: In olden more honest times it was called slavery. In today's politically correct times we talk of *deskilling* or simply management. Ancient India was the most sane, humane and level-headed about this: specialization ≡ sudraness.

> *Sudra* is usually associated with the caste system. This − varna ≡ caste − is a malicious mistranslation. A more appropriate translation for varna would be 'psychological type': the sudra *chooses* to specialise, and hence *chooses* to reduce his humanity. 'Sudra' itself is quite untranslatable (vide. Whorf above) but if we want English approximations the following 4 add up roughly to sudra: (1) specialist (2) technician (technologist, technocrat are synonymous modulo hubris) (3) professional (4) consumer (in the sense of consumerism). The upper 'caste' is easier to translate: master. Even here though, there are subtleties: master as opposed to slave is kshatriya, master in the sense of mastery is brahmin. The brahmin is at the highest level because he does not need to do or manifest his mastery — he is naturally a master.
>
> From the point of view of academics the meaning is quite unequivocal: The only valid goal of study is mastery. Any other goal leads to slavery.

The author's prejudices should be clear: I do not like specialization. I do not see that if the world swarms over with technicians or professionals it will become a paradise — on the contrary. Those who are specialised in a narrow area become illiterate in another, those who are technicians or professionals in one field become consumers in general. It was almost 30 years ago that Dijkstra pointed out that the word *user* in computer jargon stands for what was later to become consumer in a capitalist, consumerist sense. He is still worth hearing:

> The computer "user" isn't a real person of flesh and blood, with passions and brains. No, he is a mythical figure, and not a pleasant one either. A kind of mongrel with money but without taste, an ugly caricature that is uninspiring to work for. He is, as a matter of fact, such an uninspiring idiot that his stupidity alone is sufficient explanation for the ugliness of most computer systems. And oh! Is he uneducated!

Those who dont agree with this are unlikely to find this book helpful. For those who wish to continue, the principle in short is: Between *philosophy* which is grand but empty and *technology* which is powerful but too detailed and variegated and confusing and narrow, is *science* which bridges the gap. We seek to study all three in a suitable sequence.

And so a third answer to Dijkstra's challenge: At the dawn − in European terms − of the scientific era, scientists were the true humanists who sought to bridge the gap between philosophy (= religion) and the common man. After 500 years, we are in a reverse position where the modern temples of learning − universities − are not bridges but walls between the masters who make, and the masses who use — *users*. The computer if 'used' thoughtlessly can thicken these walls, if used rightly, it can obliterate them (vide@ www.fsf.org).

5. Problems with current COs

@The following are "meta thoughts" which should probably not be written but their import woven into the text

5.1 Engineering

CO is obviously an engineering subject. But over years with the extreme separation of h/w and s/w and also with the non-availability of h/w foundaries in most places other than US, the study of CO has become abstract. Now one can as meaningfully study an engineering subject in the abstract as one can 'study' cycling or swimming. Certain subjects are *doing* subjects and removing the active component invalidates the whole enterprise, making it merely descriptive.

Descriptive subjects degrade students, teachers and the whole milieu that participates. Why, for example does no one want to study English or history? Because what happens in an English dept is: *Shakespeare said ... and X said ... about Shakespeare and Y said ... about X ad libitum.* Likewise we only study the dirty political history of a musty past but never *How to make history now.* I would hazard a guess that many young people would give a lot to become 1 hundreth of Shakespeare or (you choose your idol). And everyone would want to create history. The divorcing of a subject from me here and now loses its relevance and ultimately its validity.

CO, as taught in typical CS departments, is about as valid as numerical analysis; both historically central but today largely irrelevant. If we want to salvage CO it must become a *doing* course like IP or DFS. Today things are possible (eg Xilinx) that even a few years ago were not conceivable (in India)

I would suggest a lab component consisting of

1. 74LS gates and experiments. May be preferable to use MOS than bipolar but is it feasible? Possible? Dont know.
2. With xilinx foundation series (other CAD tools@) a student can not only do CAD style design on the computer but actually download designs onto an FPGA and test out. Scale? Cost? Feasibility?
3. 'Experiments' to put together SMPS's, motherboards, cpus, disks, maybe of different generations, configure BIOS's install and configure OS's (maybe a small network?) so that the whole picture begins to fit together.

   If the reader thinks this is irrelavant he is requested to mdeitate on the fact that CO expands to *computer organization.* Yes, admittedly all this is system admin but I believe that system admin is more central to today's computing milieu than it was even 2-3 years ago. When I studied 20 years ago (end of the main frame era) there was a compulsory half-credit course called computer centre management. That slot should today be filled by a system admin course which would be strongly linked to not just CO but LLP, syspro, networks and others.

5.2 Science

Just as there is too little engineering in the CO course, there is too little science as well. And things are getting only worse as technology replaces science (eg logic minimization is becoming unfashionable in the modern breed of books like Hen Pat). Just because Espresso is more popular than Quine McCluskey, not teaching even K-map minimization is a modern fad. In my (obsolete!) view the design and analysis of sequential circuits is a solid subject. It is a nice point for students to get the synergy between basic science – finite automata – and pragmatic technology – clocked flip-flop networks.

The replacement of foundation areas with hot technology amounts to the commitment to maximise the production of shudras and to sideline potential brahmins. Those who value humanity over technology would strenuously oppose such moves.

Other science I would like to see in the CO course.

1. Whether one uses bipolar or MOS is just technology but that one needs a transistor and cannot manage with diodes and resistors etc. is fundamental science. In fact it should be possible starting from thermodynamics, to prove that a basic logic element must be an active device and have a characteristic that is non-linear. (Ward and Halstead, Feynman lectures in computation).
2. Logic gates are taught today as though God made them or else they are hardly mentioned at all. Both options are unacceptable. The development of a logic element from a switching element provides a beautiful coincidence of maths (boolean algebra) eletronics (simple CMOS design) and

basic science (developing the MOSFET as a switch from simple semconductor ideas). And covering this takes not more than a couple of lectures.

3. Multiple views of design: Gate level, architectural, data-flow, behavioral
4. Current mode vs voltage mode circuits (ECL, Bipolar, MOS) (Dont know any of this and not sure of its relevance).