Having been asked to write a guest editorial for a new Australian journal of educational computing, I had to think about which of my hobby-horses it would be most appropriate to ride on this occasion. I decided that if something high-sounding and impressive was wanted, someone else would have been asked, so I've chosen the most outrageous of all the things I believe about computers in education.

The only good reason for a kid to use a computer is if it's fun for that kid. There is nothing that everyone needs to know about computers. We all know that those advertisements about Johnny flunking out of college because Mum and Dad didn't buy him a certain brand of computer are not only crass, but also utter nonsense and a truly evil encouragement of hysteria; yet some educators push the same myth in slightly less obvious form when they develop "computer awareness" curricula.

"A computer is made up of processor, memory, input, and output units." This typical curricular gem was arguably true a quarter of a century ago when computers had a card reader at one end and a printer at the other. It is not even slightly true today, when a magnetic disk drive is used as input device by one program, as output device by another, and as swapping memory by the operating system, all at the same time. What's more, inside that disk drive is a controller containing its own processor and memory. The computer itself may be built around a gate array chip that can be thought of as processor or as memory, depending on how you choose to look at it. The large data flow arrow that sweeps clearly from left to right across the textbook page from the card reader to the printer is altogether misleading. But even the curricular gems that do have a degree of truth are still awesomely irrelevant. Neither computer programmers nor spreadsheet-wielding captains of industry start their day by bowing to the East while chanting "RAM, ROM, Bit, Byte, CPU!"

What about those spreadsheets, then? Surely our children need "exposure" to application programs so that they will be prepared to find work. Sorry, but that's nonsense too. Think about it. Why did VisiCalc turn the world of business upside-down overnight, making its developers instant millionaires? Precisely because no computer expertise is needed to use it! Instead of having to hire a member of the computer priesthood to translate business terms into computersese, an executive could interact with VisiCalc in precisely the same language she had been using all along to prepare spreadsheets by hand. The only difference is that it's faster with the computer. Learning to use VisiCalc is a matter of an hour or two, not a semester. And as programming technology improves, the application programs become even easier to use, not harder. What's more, any specific program that we teach a teenager will certainly be obsolete by the time he or she actually enters the workforce.

As a kid, I used to spend a lot of time in libraries. When I was about 11 years old, I happened to read a book about hypnotism. I was fascinated. I spent all my spare time over the next couple of years tracking down references to early research reports in the field. That was when I broke free of the kids' section of the library and started reading real books. Looking back at it, I can now see that what was so fascinating about hypnotism was the idea of having power over someone. As we all know, kids don't generally have much power even over their own lives. They're surrounded by parents and teachers and other such grownups who keep telling them what to do, from the best of motives, of course. And I was the kind of kid that other kids liked to beat up.

Unfortunately, enslaving one's classmates turns out not to be a practical course of action. It's not so easy to hypnotise someone in between classes, and besides, the grownups tend to discourage the idea. My research didn't change my life in any obvious way. (It did, of course, empower me in more subtle ways that I came to appreciate later. I learned a lot about reading and writing and research and finding my way around the big city to track down obscure used book stores.)

But, see, just a few years later my dreams of domination really came true, in a way that even the grownups approved. I was about 14 years old when I discovered computers. They were meant to be enslaved! By mastering the esoteric knowledge of FORTRAN programming (this was in 1964), I could get this million-dollar machine to do whatever I wanted. At
first I was satisfied if I could just get a list of prime numbers. Later I found that the FORTRAN compiler generated debugging information that could be used to get a trace of subroutine calls if I wrote the appropriate program in assembler. That was the beginning of my continuing interest in systems programming. But the real highlight of my secondary school computing career was when I was allowed to do *real work* — something the grownups actually needed. New York University was in the process of installing a new computer, a CDC 6600, and there was a huge amount of work to be done in converting everything to the new machine. Several of us high school kids were pressed into service. My assignment was to write a disassembler, a program that would turn 6600 binary machine language programs back into assembler mnemonics.

That's what computing in schools should be about. Empowerment. Not everyone has the particular bent to find empowerment through this route. Some kids find it in playing the violin, or the electric guitar. Others find it in writing, whether poetry or journalism. Others may find it in sports. But the sad fact is that many kids don't find it at all — not because they don't have the innate ability, but because we fill up their days with meaningless jumping through hoops.

A universal computing curriculum hurts those who don't love computers by distracting them from their true work, whatever that may be. But what's worse is that the same curriculum hurts the ones who do love computers by transforming a medium of empowerment into one more boring hoop. A year 12 student who is considering studying Computer Science at university should be programming the computer, dominating it, inventing new ways to use it, not working exercises or memorizing irrelevant facts half of which aren't true anyway.