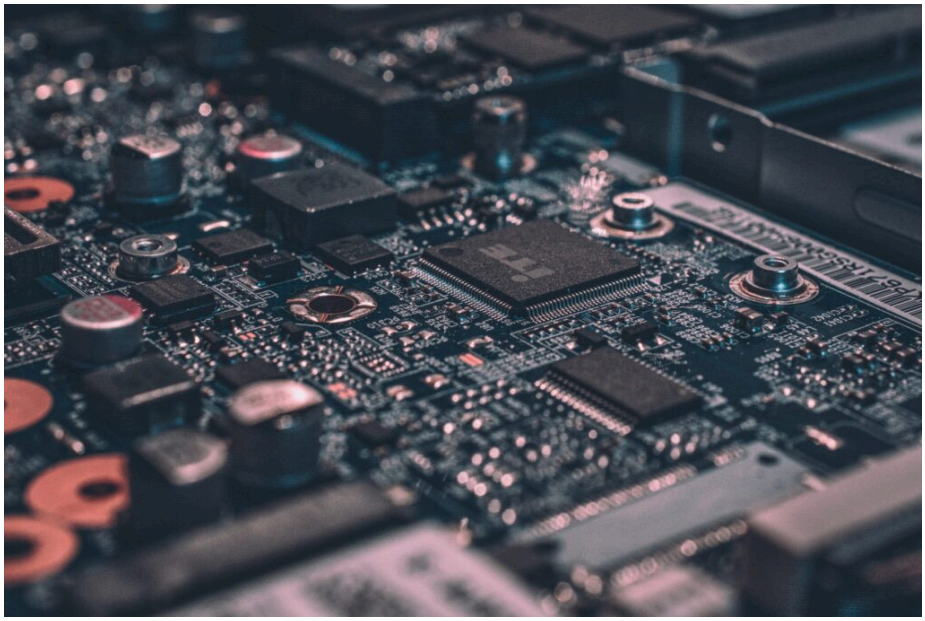
Issue 026: Hardware

Graham Lee

November 2nd, 2020



Welcome to the twenty-sixth issue of *De Programmatica Ipsum*, focused on the subject of *Hardware*. In this edition:

- Adrian argues that hardware has lost the marketing appeal[1] it had decades ago.
- Graham investigates the reasons behind the disappearance of workstations[2] from the computer landscape.
- In the Library section[3], Adrian remembers the IBM PC and Peter Norton's hallmark "Pink Shirt" book[4].

Enjoy this issue! Please subscribe to our free newsletter[5] to stay updated about new releases, share the articles on social media, or contribute[6] if you would like to support our work.

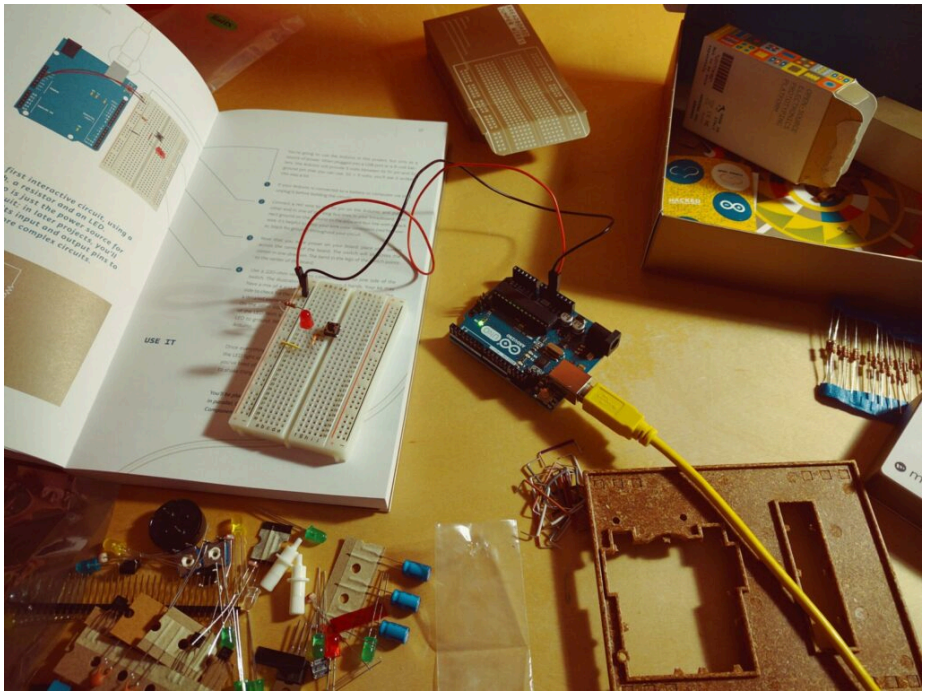
Cover photo by Alexandre Debiève[7] on Unsplash[8].

[1] https://deprogrammaticaipsum.com/breaking-the-3-ghz-barrier/
[2] https://deprogrammaticaipsum.com/the-untimely-demise-of-workstations/
[3] https://deprogrammaticaipsum.com/category/library/
[4] https://deprogrammaticaipsum.com/peter-norton/
[5] https://deprogrammaticaipsum.com/newsletter/
[6] https://deprogrammaticaipsum.com/contribute/
[7] https://unsplash.com/@alexkixa?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[8] https://unsplash.com/s/photos/hardware?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Breaking The 3 GHz Barrier

Adrian Kosmaczewski

November 2nd, 2020



My first serious attempt at understanding computer hardware happened during college, in 1994. One of the labs consisted in wiring a 4-bit processor to a series of switches and a LED display. The objective was to make a very simple operation: starting from zero, make the display increment one digit every time the switch is activated. Or, as it is commonly know, to make a bare bones half adder[1].

That CPU was intended for teaching purposes; a squared wafer of silicon about 30 cm wide and large, with big capacitors and a whole circuitry system easily viewable with the naked eye. We could plug and unplug components of different resistance, capacity and voltage. The teacher explained to us that this contraption was a much simplified component, loosely based on the architecture of the Intel 4004[2], the first commercially available microprocessor, released in *annus mirabilis* 1971.

To be honest, I could not really wire everything properly without some help from my classmates. I was puzzled, wrongfully lost in the perception that computers could not be that simple. Yet, with a bit of wiring and patience, indeed, the LED displayed the required bits (no pun intended) of a very simple calculator in front of my eyes.

Seeing a set of wires and lights perform the simplest of all mathematical operations had quite an impact on me. Maths were no longer an abstract concept; with the proper hardware,

[1] https://en.wikipedia.org/wiki/Adder_(electronics)
[2] https://en.wikipedia.org/wiki/Intel_4004

maths could literally come to life. I felt the excitement and the wonder of those scientists who, in the 1940s and 1950s, discovered this fact and built a new science around it.

This CPU, as simple as it was, showed me what the soul of this new machine was made of.

TANSTAAFL

These days we read in the press about supercomputers built with CPUs featuring names seemingly borrowed from a science fiction novel. Take the POWER9[3], for example. According to Wikipedia, it is a "superscalar, multithreading, symmetric multiprocessor." Not even HAL 9000[4] received such attractive marketing buzzwords. The POWER9 is a real beast, used as the brains of one of the fastest supercomputers ever made[5], at the time of this writing at least.

In 2004, Herb Sutter wrote "The Free Lunch Is Over[6]", a seminal article announcing the end of the path for Moore's Law[7]. This quote stands out:

> That willingness is simply a clear indicator of the extreme pressure the chip designers face to deliver ever-faster CPUs; they're under so much pressure that they'll risk changing the meaning of your program, and possibly break it, in order to make it run faster.

In the eyes of this author, three major changes followed this article.

The first was quite visible in computer magazines. Computer marketing had no other solution but to drop CPU speeds as a differentiating factor among competitors. After all, most CPUs have been capped at around 3 GHz, and for almost 20 years now. As Graham observed during the discussions around this edition of the magazine, Steve Jobs never got his 3 GHz PowerPC G5 CPU from IBM, so he jumped ship to Intel. But he still did not immediately get a 3 GHz CPU, but rather, something called a "Core Duo".

The second visible effect, and quite dramatic this time, was Meltdown and Spectre[8]. Security vulnerabilities using *precisely* those advanced CPU features, as attack vectors of unprecedented risk and virulence.

But from the perspective of the programmer, the third visible effect was the spread of multicore CPUs, now commonplace even in smartphones and small boards like the Raspberry Pi[9], abundant in the drawers and behind the TV sets of most software developers reading these lines.

Multicore

The PC revolution happened on single-core CPUs running first single-threaded, then multithreaded applications. Is cloud and smartphone revolution currently based on multiple-core CPUs running single-threaded applications?

The availability of multicore CPUs at the end of the first decade of this millenium unleashed a new era in the programming of multithreaded apps.

[3] https://en.wikipedia.org/wiki/POWER9
[4] https://en.wikipedia.org/wiki/HAL_9000
[5] https://en.wikipedia.org/wiki/Summit_(supercomputer)
[6] http://www.gotw.ca/publications/concurrency-ddj.htm
[7] https://en.wikipedia.org/wiki/Moore%27s_law
[8] https://meltdownattack.com/
[9] https://www.raspberrypi.org/

First, technologies such as OpenMP[10] and libdispatch[11] simplified the conceptual work of designing and writing applications for multicore CPUs, avoiding the requirement to write threading code, dealing with race conditions, semaphores, shared data, and other concepts.

Second, event loop libraries and runtimes such as Node.js[12], libuv[13] and libevent[14] provided a different solution to the problem, avoiding multiprocessing altogether, at least from the point of view of the developer. Single threading and event loops for everyone.

Third, there was a the functional programming renaissance. This led in turn to two interesting trends: on one side the rise, of new and old languages such as Scala[15], Haskell[16], and F#[17]; on the other, "classical" programming languages including functional programming concepts. To name a few: C++[18], C#[19], Java[20], PHP[21], Objective-C[22]; they all got lambdas and functional idioms retrofitted into them in one way or another.

The rising popularities of Python, Ruby and JavaScript during the Web 2.0 era are arguably also to blame in this evolution; not to mention the spread of the MapReduce[23] processing model (2004) and the respective availability of Hadoop[24] and other similar technologies.

Virtualization

Developers are quite detached from hardware considerations these days. I have made a career in a software world increasingly disconnected from that of hardware. Most of the time, I would say 90% of the time, the computers running my code were completely, absolutely, and hopelessly virtual.

Let us consider, for example, a canonical "full stack" web developer, with some knowledge of cloud native apps; quite a common resumé these days. The person will surely use a rather high-level programming language, most probably JavaScript, Java, PHP, or C#. They will create a Docker container out of that code, and this container would run inside some Kubernetes cluster, itself running inside a virtual machine running in some cloud provider. In the middle, maybe, a CI/CD pipeline running in yet another virtual machine; most probably GitLab running the "Docker-in-Docker" image[25], building other images.

The actual hardware is, of course, nowhere to be seen. It is virtual machines all over, and legend has it that at some point there is an actual CPU executing those instructions. A CPU, you know, labeled with the 64 bits moniker, built with silicon, plugged to a socket, and hopelessly capped at 3 GHz.

[10] https://www.openmp.org/
[11] https://apple.github.io/swift-corelibs-libdispatch/
[12] https://nodejs.org/en/
[13] https://libuv.org/
[14] https://libevent.org/
[15] https://scala-lang.org/
[16] http://learnyouahaskell.com/
[17] https://fsharp.org/
[18] https://en.cppreference.com/w/cpp/language/lambda
[19] https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-expressions
[20] https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html
[21] https://www.php.net/manual/en/functions.anonymous.php
[22] https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/WorkingwithBlocks/WorkingwithBlocks.html
[23] http://static.googleusercontent.com/media/research.google.com/es/us/archive/mapreduce-osdi04.pdf
[24] https://hadoop.apache.org/
[25] https://docs.gitlab.com/ee/ci/docker/using_docker_build.html

Cover photo by Spencer[26] on Unsplash[27].

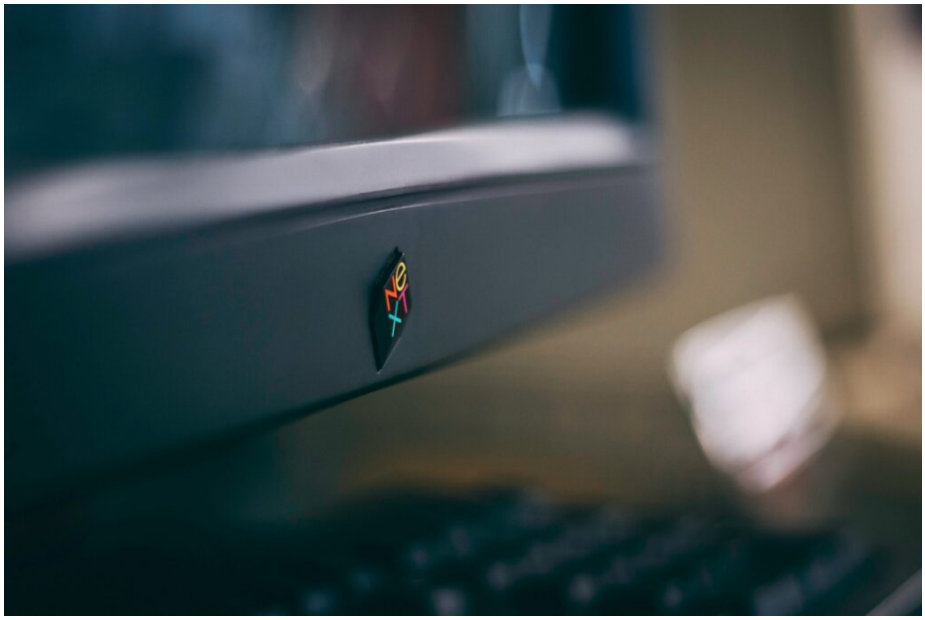
[26]https://unsplash.com/@spen?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[27]https://unsplash.com/s/photos/circuit?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

The Untimely Demise Of Workstations

Graham Lee

November 2nd, 2020



Last month's news that IBM would do a Hewlett-Packard[1] and divide into two—an IT consultancy and a buzzword compliance unit—marks the end of "business as usual" for yet another of the great workstation companies.

A quick aside on computing history. You can imagine personal computing being driven by two distinct schools of thought. The "top down" school, represented by research-led organisations including Xerox PARC, Bell Labs,academia and the military, asked "what would the world be like if everyone had their own minicomputer"? They took large, time-sharing systems like UNIX and installed them first under, then on, employees' desks for their own personal use.

The "bottom up" school was made up of hobbyists who asked "can we make an interesting computer out of inexpensive components"? Thus companies like Apple and MITS in the US, Acorn and Sinclair Radionics in the UK, and others took chips that were usually used as peripherals controllers in "real" computers and built interactive programming systems around them. The microcomputer revolution came from the bottom-up school, as they made home computing affordable. The workstation revolution came from the top-down school, as they made powerful on-demand computing feasible.

The two schools came into very close proximity in the 1980s, when the Motorola 68000 family of CPUs (along with the 68881/68882 FPU and 68851 MMU) were the processors of choice in everything from entry-level PCs like the Atari 520ST, through games consoles

[1] https://arstechnica.com/information-technology/2020/10/ibm-to-split-into-two-companies-by-the-end-of-2021/

like the Sega Mega Drive (Genesis in the US), to the most expensive UNIX workstations from NeXT Computer, Sun Microsystems, and Apollo Computer.

But then the workstation makers invested heavily in their own CPU architectures based on RISC design principles and again the two diverged. The workstation market became highly differentiated: RS/6000 from IBM (later PowerPC), Alpha from Digital Equipment Corp, MIPS from, well, MIPS, SPARC from Sun, PA-RISC from HP. The software on these workstations, while superficially very similar, was also differentiated and surprisingly incompatible. Take a program from HP-UX and you'll have difficulty running it on NeXTSTEP, unless the authors shared the source code and used the nascent GNU autotools to support portable building. As Yoda said: begun, the UNIX wars[2] have.

Of course we know that the (desktop) computing world today is mostly Intel and that workstations are mostly fancy PCs, rather than bespoke designs by vertically-integrated companies, Apple being the two trillion dollar outlier. How we got here was that the commodity parts got good enough that there was no evident advantage to workstation-grade hardware. A high-end PC could easily run a workstation OS like System V UNIX (Solaris was an early example), BSD (386BSD which later became FreeBSD, or NeXTSTEP) or Windows NT.

Along the way, the workstation companies consolidated (Apollo and eventually DEC got absorbed into HP; MIPS into SGI) or disappeared altogether (Sun became Oracle Hardware; SGI went bankrupt and sold its assets to sgi; Symbolics did similar—incidentally Symbolics was the first company with a .com domain). IBM long ago stopped even making its own brand PCs, and the news of its split means that there are now very few workstation companies trading in the same form they had "back in the day". The only ones I can think of that have not had major changes to their corporate structures are Xerox and Sony, whose management may not even have known that they sold workstations.

What's got lost alongside the death of the workstation is the business model where you sell expensive computers as part of an integrated solution into a particular vertical market, where that expensive solution will cost a lot less than cobbling something together out of cheap PCs. Why? I think people have a lower expectation and higher pain threshold when using computers now; they expect an amount of friction based on their own experience and translate that expectation into realms where it doesn't belong. As I described way back in issue 2, computing is a lemon market[3].

Organisations would go to the workstation vendors because they solved particular problems very well. If you're in AI, you need Symbolics. Computer graphics, SGI. Telecoms, that'd be Sun. If you want to write software in Ada for the military-industrial complex, you'll be buying a Rational workstation. Yes, the first IDE was a completely integrated package of hardware and software. And, of course, Apple for Desktop Publishing, the Mac being a workstation of sorts itself. People would buy computers *because* applications like AutoCAD, Quark or Mathematica ran well on them. They wouldn't buy the computer then browse the App Store to see whether it could do anything useful.

And the strange thing is that catering to those vertical markets with integrated solutions is easier than ever now. The wide availability of free software means that the basic job of "being a desktop computer" is taken care of at zero cost, so business can focus on contributing valuable bespoke behaviour. And hardware costs are lower than ever: the availability of high-capability SoCs and single-board computers like the Raspberry Pi and Rock64 should make

[2] https://www.livinginternet.com/i/iw_unix_war.htm
[3] https://deprogrammaticaipsum.com/the-various-meanings-of-quality/

it a no-brainer to sell the computers as accessories for the applications, not the other way around.

In high-tech domains, an engineer could readily have a toolchest of suitable computers in the same way that a mechanic has different tools for their tasks. This one has an FPGA connected by both PCI-E and JTAG to allow for quick hardware prototyping. This one is connected to a high-throughput GPU for visualisations; that one to a high-capacity GPU for scientific simulations.

The general purpose hardware vendors want us to believe that an okay-at-anything computer is the best for everything: you don't need a truck, so here's a car. But when you're hauling a ton of goods, you'll find it cheaper and more satisfying to shell out more for a truck. Okay-at-anything is good for nothing.

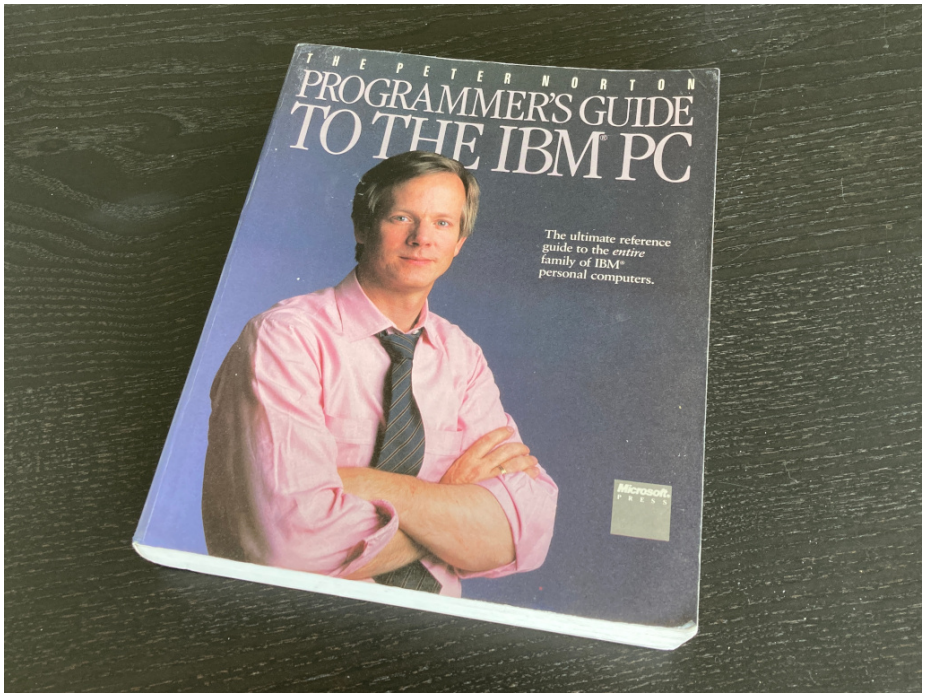
Cover photo by Serhii Butenko[4] on Unsplash[5].

[4] https://unsplash.com/@serejahh
[5] https://unsplash.com/photos/zx2Vc1zPDIs

Peter Norton

Adrian Kosmaczewski

November 2nd, 2020



Some successful computer books have earned memorable nicknames. There is the "K&R[1]" book, the "Gang of Four[2]" book, and, to please generations of board and role game players[3], there are also the "Wizard Book[4]", the "Dragon Book[5]", and the "Dinosaur Book[6]". There's the "Camel[7]" book and the "Pickaxe[8]" book. And then, with a decidedly more corporate look and feel, let us talk today about the "Pink Shirt" book, officially titled "The Peter Norton Programmer's Guide to the IBM PC." More corporate, yes, because the PC was after all a business machine coming from a business corporation.

As a reference for our younger readers, many of which might be reading this publication on an IBM PC (nowadays more commonly referred to as Windows PCs,) it is important to mention one historic fact. This very Mr. Peter Norton[9], the one appearing in the cover of the book, is the same that gave his name to famed products such as Norton Antivirus, Norton Commander (which begat a popular heir[10]), Norton Utilities (the most ground-breaking of

[1] https://en.wikipedia.org/wiki/The_C_Programming_Language
[2] https://en.wikipedia.org/wiki/Design_Patterns
[3] https://mark-gently.tumblr.com/post/154355213324/techieyuckyboy-wizards-dragons-and-dinosaurs
[4] https://mitpress.mit.edu/sites/default/files/sicp/index.html
[5] https://en.wikipedia.org/wiki/Dragon_Book_%28computer_science%29
[6] https://www.amazon.in/gp/product/0471694665/
[7] https://en.wikipedia.org/wiki/Programming_Perl
[8] https://pragprog.com/titles/ruby/programming-ruby-2nd-edition/
[9] https://www.technologizer.com/2014/06/05/where-have-you-gone-peter-norton/
[10] https://midnight-commander.org/

which was undoubedly `UNERASE`), culminating with the Norton 360 suite[11] available at the time of this writing.

The Norton name played a double role in the PC industry of the 1980s: on one side, representing the aforementioned products, which helped millions of IBM PC users in recovering deleted files and preventing virus infections. On the other side, the pre-Twitch, pre-YouTubber, pre-blogger, pre-web influencer, pre-Guy Kawasaki evangelist. The role that nowadays Scott Hanselman[12], Kelsey Hightower[13] and John Sundell[14] play for .NET, Kubernetes, and Swift, respectively.

Joel Spolsky mentioned[15] the Pink Shirt book (twice[16], actually) as one of the major references that shaped his career.

The IBM PC

Why, among the huge numbers of different architectures and standards in the market, did the IBM PC, built around the x86 CPU, became synonym with desktop computing? As explained by Norm DeWitt during his guest appearance in the March 19th, 1985 episode of The Computer Chronicles[17], by 1984 IBM had captured 26% of the worldwide PC market.

(By the way, that video includes a demo of IBM TopView[18], a series of unfortunate predictions by Mr. DeWitt, a discussion about Digital Research's Concurrent DOS[19] by Gary Kildall himself, and a fascinating discussion of the relationship between IBM and other smaller partners in the industry. Totally worth a watch. But I digress.)

Emerson W. Pugh, in his excellent book "Building IBM[20]" (MIT Press, 1995) dedicates the last few words of the last chapter to this new machine, born in the Boca Raton, Florida IBU (Independent Business Unit).

> To achieve broad market acceptance, for example, the PC was to have an open rather than proprietary architecture. To reduce development time and costs, the operating system was to be purchased from an independent software firm, and hardware components were to be open for competitive bids from inside and outside the company. (...) Spurred on by a brilliant advertising campaign that featured Charlie Chaplin's little tramp, sales of the PC soared beyond all expectations.

The introduction of the IBM PC was the defining moment of a whole market. Software vendors such as Microsoft, Software Arts[21], and Digital Research were all invited to provide software for this new machine. Byte Magazine started its January 1982 issue[22] with a glowing statement[23]:

[11] https://norton.com/
[12] https://www.hanselman.com/
[13] https://github.com/kelseyhightower/kubernetes-the-hard-way
[14] https://www.swiftbysundell.com/
[15] https://www.joelonsoftware.com/2002/12/11/lord-palmerston-on-programming/
[16] https://www.joelonsoftware.com/2004/05/05/mike-gunderloys-coder-to-developer/
[17] https://archive.org/details/Profileo1985
[18] https://en.wikipedia.org/wiki/IBM_TopView
[19] https://en.wikipedia.org/wiki/Multiuser_DOS#Concurrent_DOS
[20] https://mitpress.mit.edu/books/building-ibm
[21] https://www.youtube.com/watch?v=uDsU9wyQ5Ks
[22] https://archive.org/details/byte-magazine-1982-01
[23] https://archive.org/details/byte-magazine-1982-01/page/n37/mode/2up

> What microcomputer has color graphics like the Apple II, an 80-column display like the TRS-80 Model II, a redefinable character set like the Atari 800, a 16-bit microprocessor like the Texas Instruments TI 99/4, an expanded memory space like the Apple III, a full-function uppercase and lowercase keyboard like the TRS-80 Model III, and BASIC color graphics like the TRS-80 Color Computer? Answer: the IBM Personal Computer, which is a synthesis of the best the microcomputer industry has offered to date.

Of course, not everybody was as enthusiastic at first. Dr. Dobb's Journal first mentions the IBM PC in a small note in page 45 of its October 1981 edition[24], in a rather laconic fashion:

> Late News From IBM: On August 12th, IBM announced the IBM Personal Computer, a small home machine. It will be sold, IBM announced, through Computerland stores and Sears Business Machines outlets. The machine consists of a console with keyboard, a separate video monitor (which may be a home TV), and cassette and disk units. It uses 5-inch disks (...)

The Pink Shirt book covers subjects applicable to the five kinds of IBM PCs available in 1985: The 5150 PC[25], the 5160 XT[26], the 5170 AT[27], the 5155 Portable[28], and what was arguably the first major flop in the PC industry, the IBM PCjr[29].

To give an idea of the type of hardware we are talking about: all of these machines used the Intel 8088 CPU, except for the AT, which used the 80286. None included the 8087 / 80287 "arithmetic coprocessor" (sold separately!) for which the PCjr did not even have a dedicated slot in the motherboard! Let us be clear: the Intel 8088 could only perform integer math, and floating-point operations were executed in software. Gaming on a PC was not going to be a thing until almost a decade later. This is the world this book was born into.

GPUs were not even a science fiction thing back then.

Highlights

In many ways, the Pink Shirt book is akin to "The Linux Programming Interface[30]" by Michael Kerrisk. Of course, instead of Linux syscalls, you get lists of DOS interrupts. But the idea is the same; to be able to write efficient, fast software for this new kind of computer, using the lowest possible interface available in the original PC. (Ironically enough, there is a 1999 book co-written by Peter Norton about Linux[31], but we will leave it for a future edition of this magazine.)

Take, for example, the list of DOS interrupts detailed in chapter 16, called "Universal DOS Functions"; there one finds all that is needed to read and write files, command I/O from the keyboard and to peripherals. The book is sprinkled with bits and pieces of the design philosophy used in the creation of the IBM PC, most of them turning around a central theme:

> The basic philosophy of the PC family is: Let the BIOS do it; don't mess with direct control.

[24] https://archive.org/details/dr_dobbs_journal_vol_06/page/n477/mode/2up
[25] https://en.wikipedia.org/wiki/IBM_Personal_Computer
[26] https://en.wikipedia.org/wiki/IBM_Personal_Computer_XT
[27] https://en.wikipedia.org/wiki/IBM_Personal_Computer/AT
[28] https://en.wikipedia.org/wiki/IBM_Portable_Personal_Computer
[29] https://en.wikipedia.org/wiki/IBM_PCjr
[30] https://en.wikipedia.org/wiki/The_Linux_Programming_Interface
[31] https://www.amazon.com/Peter-Nortons-Complete-Guide-Norton/dp/0672315734

Mr. Norton will stress this notion several times across his text: to build durable, compatible software across all IBM PC clones, always use interrupts, follow IBM's advice, and do not circumvent this rule.

The Pink Shirt book is aimed at practicioners; its pages are filled with practical bits and pieces of information, from insider tips and tricks, to actual code snippets. Suffice to mention the "actual production code" taken from the Norton Utilities, shown in page 292, aimed at the calculation of weekdays. Or the section about "Terminate-but-Stay-Resident" (interrupt 39) in page 252, allowing a severely crippled DOS architecture to provide users with an experience similar to that of a multitasking system. Or Appendix B, providing a timeless explanation of hexadecimal arithmetic.

Coupled with its deep detail, there is a honest, down-to-earth tone throughout the book. Mr. Norton is not shy of literally saying "I don't know" when the time comes; for example in page 37, when explaining how a small snippet of BASIC code can... completly lock up a PCjr. A good example of what I called "honestization" in a previous edition[32] of this magazine.

Last but not least, the Pink Shirt book included two chapters (19 and 20) dedicated to the creation of programs, using the five most important programming languages[33] available at this point in time: assembly language, interpreted BASIC, compiled BASIC, Pascal, and of course C. A delightful reading for anyone curious about programming languages history.

Legacy

Why is the Pink Shirt book still relevant, almost 40 years later? I will let Raymond Chen answer this question with a quote from his classic book "The Old New Thing[34]" (Addison-Wesley, 2007)

> Every company has its own collection of line-of-business (LOB) applications. These are programs that the company uses for its day-to-day business, programs the company simply cannot live without. (...) If a Windows upgrade breaks a LOB application, it's game over. No upgrade. (...) And it happens that a lot of these LOB applications are 16-bit programs. Some are DOS.

This book comes from a time when computers filled entire desks, weighted tens of kilos, had noisy fans, and only a few had mice attached to them. Mr. Norton's thoughts around portability[35] and architecture are still valid and stand the test of time.

And most surprisingly, most of the assembly code snippets will still work on any standard Windows PC, in a show of perenniality[36] that is very rare in our industry. Now go, grab that old Pentium PC in your garage, install FreeDOS[37] on it, and write[38] some apps.

Cover photo by the author.

[32] https://deprogrammaticaipsum.com/less-evangelization-more-honestization/
[33] https://deprogrammaticaipsum.com/issue/issue-013-programming-languages/
[34] https://www.pearson.ch/Informatik/Addison-Wesley/EAN/9780321440303/Old-New-Thing-The
[35] https://deprogrammaticaipsum.com/issue/issue-019-cross-platform/
[36] https://deprogrammaticaipsum.com/issue/issue-018-obsolescence/
[37] http://freedos.org/
[38] https://archive.org/details/BorlandTurboPascal4.0