* * * * * * * * * * * *

"small" TEX

Lance Carnes

In the previous issue, I discussed several IBM PC TEX versions that were in progress. By October, 1984, two versions were running, one mine, and one by David Fuchs. By the time you read this, the marketing groups supporting both versions will have sent you at least one piece of mail deascribing their products. Additional articles and advertisements on these two versions are found elsewhere in this issue.

The two versions are similar in that they execute on an 8088- or 8086-based computer running MS-DOS. Each requires at least 512Kb of memory (640Kb to run LaTeX) and a 10Mb hard disk. David's version was done with Lattice-C and requires an 8087 coprocessor. My implementation was done with MS-Pascal and does not need an 8087 coprocessor. Both versions compile the TEXbook at an average of 25 seconds per page.

Barry Smith of Kellerman and Smith reports that the Apple Macintosh version is well underway, and they expect to have a running version by the time you read this. They intend to use the "Fat Mac" (512Kb memory version) and the Macintosh XL, and direct output to the recently announced Laserwriter.

Below is the grid for known small TEX implementations. All versions are TEX82 unless otherwise noted. Let me know if you have additional information.

"small" TEX implementations

Computer	Processor	Contact	Organization, Address
Hewlett-Packard 3000	16-bit	Lance Carnes	TeXeT, 163 Linden Lane, Mill Valley, CA 94941; 415-388-88
Hewlett-Packard 1000	16-bit	John Johnson	JDJ Wordware, Box 354, Cupertino, CA 95015; 415-965-32
DEC PDP-11/44 Plexus, Onyx IBM PC	16-bit[1] Z8000[1] 8086/88[1]	Dick Gauthier	TYX, 11250 Roger Bacon Dr., Suite 16, Reston, VA 22090; 703-471-0233
Apollo	MC68000	Thom Hickey	OCLC, Box 7777, Dublin OH 43017; 614-764-6075
		Bill Gropp	Dept. of Computer Science, Yale University, Box 2158, Y Station, New Haven, CT 06520; 203-436-3761
		Pierre Clouthier	COS Information, 6272 Notre Dame West, Montreal, H4C 1 P.Q.; 514-935-4222
Hewlett-Packard 9836	MC68000	Jim Crumly	Hewlett-Packard, Box 15, Boise, ID 83707; 208-376-6000 x2
Sun Microsystems	MC68000	Jim Sterken	Textset, Box 7993, Ann Arbor, MI 48107; 313-996-3566
		Rich Furuta	University of Washington, Computer Science, FR-35, Sea WA 98195; 206-543-7798
Cyb	MC68000[2]	Norman Naugle	Mathematics Dept., Texas A&M University, College Station, 77843; 409-845-3104
Apple MacIntosh, Lisa	MC68000[3]	Barry Smith Dave Kellerman	Kellerman & Smith, 2343 SE 45th Av., Portland, OR 97215; 503-232-4799
Masscomp	MC68000	Bart Childs	Dept. of Computer Science, Texas A&M University, Coll Station, TX 77843; 409-845-5470
Synapse	MC68000	Dick Wallenstein	Comcon, 5 Underwood Ct., Delran, NJ 08075; 609-764-1720
PERQ/ICL		Jaap van 't Ooster	Océ, St. Urbanusweg 43, 5900 MA Venlo, Holland
IBM PC, XT, AT	8088, 80286	Lance Carnes	TeXeT, 163 Linden Lane, Mill Valley, CA 94941; 415-388-88
IBM XT, AT	8088, 80286	David Fuchs	Dept. of Computer Science, Stanford University, Stanford, 94305
IBM XT, AT	8088, 80286[3]	Ronny Bar-Gadda	446 College Av., Palo Alto, CA 94306; 415-326-1275

[1] not TEX82 [2] currently unimplementable [3] in progress or recently completed

TEX Now on Microcomputers

At the January meeting of AMS, Addison-Wesley Publishing Company demonstrated a full implementation of TEX82 for the IBM PC/XT, or IBM PC with hard disk. This implementation, called MicroTEX^{TM}, was developed by David Fuchs, who had to work quite a few tricks to shoehorn the system into the PC/XT and then to achieve high performance. The MicroTEX package also includes Addison-Wesley's driver for IBM and Epson dot matrix printers (IBM Graphics and Matrix, Epson MX-, RX-, and FX-80 and 100).

MicroTEX is an exact translation of TEX82. David first wrote a translator to convert Pascal code to C, added overlay facilities to allow TEX to operate effectively with its fonts in 512K of memory, and then compiled this program for an MS-DOS environment. In the process, David had to work around the limitations, for his purposes, of the best available C compiler. Among other things, he rewrote its math, I/O, and memory management runtimes, plus a post-compiler code optimizer and FMT file preloader.

The resulting MicroTEX includes all the commands of TEX and plain TEX, as well as all the fonts of plain TEX. Since MicroTEX *is* TEX, it produces .DVI files identical to those produced by any implementation of TEX82. As proof, MicroTEX-generated files that were printed locally by Addison-Wesley on an Epson printer were sent to Textset, Inc., in Ann Arbor, Michigan, and printed without a hitch on their QMS laser printer and APS5 phototypesetter.

MicroTEX runs under DOS 2.1 and requires 512K of memory, much of which is required to hold macros; the 8087 co-processor is not required. With this configuration, it has been timed to process *The TEXbook* in 3 hours 37 minutes, or 26.4 seconds per page for difficult and dense text. This speed is more than acceptable for most purposes, and even compares favorably with processing times on larger machines in actual timeshared environments. For less dense text, you can decrease the processing time to under 20 seconds.

When running TEX, you will get the "**" prompt 12 seconds after you invoke TEX on the XT. This is a fully pre-loaded "plain TEX." In the 512K of memory, you have a 30,000 word "mem" array, and a 20,000 word "font" array; 640K will make those arrays 60,000 and 22,000 words, respectively. All other arrays are full-size, even in 512K; that is, the arrays are the same size as on mainframe and minicomputer implementations of TEX. All data for TEX are kept in RAM; overlays are used only for infrequently used code, to minimize the cost of these overlays.

The 16 plain TEX typefaces included in the MicroTEX package are each provided at 2 dot densities, 240 dpi and 120 dpi, and each of these 32 densities, in turn, is provided at 6 magsteps: 1200, 1315, 1440, 1728, 2074, and 2488 for the 240 dpi fonts, and 600, 657, 720, 864, 1037, and 1244 for 120 dpi. The complete MicroTEX package, including TEX, fonts, and system utilities, resides in 4 megabytes of hard disk space. The package is installed from 8 floppy disks by an installation program.

When running Addison-Wesley's driver to print a .DVI file, the user can specify values for numerous processing and printing options. These options include: print quality, magstep, horizontal and vertical offsets, level of error analysis to display on the screen, where to find font files, and which pages or portions of pages in the file to process. The values hold only for that run of the driver program; the program uses default values for any options not specified. Addison-Wesley's configuration program allows the user to set values for all but one of these options, and thereby make them the new default values.

The user can select from 5 levels of print quality and can also specify the definitions of any or all print quality levels. A draft quality level provides a very readable output of a .DVI file at almost the normal speed of the dot matrix printer; highest resolution print quality takes longer, because it requires more passes of the printer, but the results are well worth it. The user may also choose to process only a portion of a .DVI file and can specify the starting position of the starting page, and the ending position of the ending page.

The MicroTEX package includes several other programs. PXL-EPF converts font files from a pixel format to a column-oriented format more useful to Epson type printers. Another program expands font files that had been stored in compressed form on the floppy disks; the program then erases the compressed font file from the hard disk, to reclaim storage space. The installation program transfers MicroTEX, the font files and the other programs in the package to the hard disk; it then invokes the pro-

gram to expand the font files.

Addison-Wesley also expects to have a version of MicroTEX for the IBM PC/AT available soon. Enhancements to the package, such as LATEX, the availability of additional drivers, and the publication of related software and books will be announced at appropriate times in the coming months. Addison-Wesley welcomes suggestions from TUG members on what is needed to make TEX even more useful and broadly available, particularly in the microcomputer environment, and welcomes discussion with you on work you may be doing in this area yourself. Contact Addison-Wesley Publishing Company, Reading, MA 01867. 617-944-3700, extension 2677.

(NOTE: This article was printed, on an Epson FX-80 printer, at 240 by 216 dpi.)

PC TEX

Lance Carnes
Personal TEX, Inc.

Having successfully ported TEX to the HP3000, I wanted to bring it to a wider market. Last summer, I formed a business with Scott Guthery and Michael Ballantyne, two TEX users from Austin, Texas, to bring up TEX on the IBM PC.

I started the project in September, 1984. Getting a version running on the PC, however, was not so simple a matter as merely compiling the `TEX.PAS` source from TANGLE. The Microsoft Pascal compiler (version 3.20) would not compile the "vanilla" Pascal code as written, and I had to create a translation program to modify the code so that it would compile cleanly, let alone run. By October, I had a version of TEX running that could at least completely process Don Knuth's `TRIP` test, though not correctly. (I have since eliminated the eight or ten minor problems.)

This first, inefficient version of PC TEX ran agonizingly slowly, taking 40 to 60 seconds to compile each page of a document. I recognized that not only would I have to eliminate the `TRIP` bugs, I must optimize the run-time performance.

I spent the next months debugging and optimizing. I was able to make use of much of the considerable amount of data I had amassed about my HP3000 version of TEX, particularly run-time characteristics. Since TEX is not optimized for particular machines, I was not surprised to find that a majority of the program's execution time on the PC was spent in file I/O, memory management, and in the modules identified as "inner loop" in the index of the TEX82 Manual.

I was able to improve file I/O performance through direct DOS calls rather than using the Microsoft Pascal run-time package. This speeded things by 10 to 15%. For example, one improvement was in the way TEX reads source input files. In the original code, these files are read one character at a time. The optimization consisted of reading the file in large blocks and de-blocking these into records by dedicated routines.

The most significant problem in optimizing memory management appeared in addressing the two arrays most often used by TEX, *mem* and *font_info*. In most implementations of TEX, these arrays are assigned storage greater than 64K. (*mem* is usually allotted 120 to 240K, and *font_info* 80 to 120.) However, Microsoft Pascal (in common with most other compilers for the PC) cannot address arrays larger than 64K. To properly implement TEX on the PC, I had to devise methods of getting around the 64K limitation. This I did by judicious use of segmented pointer types and by writing efficient machine-level address calculation routines. These memory management optimizations speed up run time by 16 to 20%.

Other miscellaneous optimizations included replacing unnecessary division and multiplication operations with other implementation-dependent operations, and changing data types to those more efficient for the particular machine. Each reduced run time by a few percent.

My last step was to eliminate the final few remaining bugs. As an interesting sidelight, during the entire development period, I discovered only one bug in the Microsoft Pascal compiler (32-bit integer division).

PC TEX is now a completed product. It compiles the TEXbook at an average rate of 25 seconds per page. We timed it on an IBM PC/XT with 512K memory (without an 8087 coprocessor). It should run three times as fast on an AT, that is, about eight seconds per page.

During the time that I was writing Pascal code, and learning all about the 8088 processor and MS-DOS, my partners and I met frequently to discuss marketing and distributing PC TEX. Dana Ballantyne, Michael's brother, brought his business expertise to the company, joining us as a partner in December.

Personal TEX, Inc., offers a full implementation of the TEX document compiler. PC TEX includes

LATEX, AMSTEX, and our own macro package that lets beginners quickly begin producing documents themselves, plus drivers for the IBM Graphics printer, and Epson RX, FX, and LQ series printers.

Elsewhere in this issue is an advertisement for PC TEX that tells you how to order and how much it costs.

Future revisions will include a preview screen driver, and drivers for QMS, Imagen, Apple's LaserWriter, and other popular output devices. We will also offer customized macro packages, written by Michael Spivak, each aimed at specific needs, such as publishing, business, and education.

If you have questions about PC TEX, you can reach me weekdays from 9 a.m. until at least 6 p.m. (PST), or leave a message with my answering service.

Lance Carnes
Personal TEX, Inc.
20 Sunnyside, Suite H
Mill Valley, CA 94941 USA
(415)388-8853. TELEX 910-481-0421

Macros

MACROS FOR TWO-COLUMN FORMAT

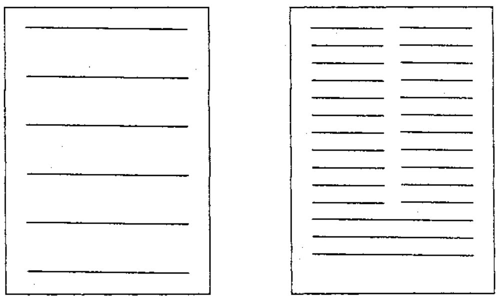
Craig Platt
University of Manitoba

In Appendix E of the TEXbook, Don Knuth presents the macros that were used for two-column formatting in Appendix I, the index. For their intended purposes they seem to have worked well enough, but there are a couple of circumstances under which they may fail. I discovered the first of these while trying to adapt the macros to another context, and in the process of working out a fix, Don came across the other.

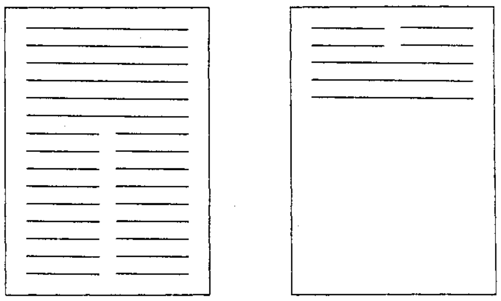
The first problem can arise when switching from single-column to double-column mode near the end of a page, and then back to single-column mode "too soon" on the next page. Referring to page 417 of The TEXbook, the `\begindoublecolumns` macro operates by first saving the current `\box255` in `\partialpage`, changing the output routine to `\doublecolumnout`, changing `\hsize` to `\colwidth`, and changing `\vsize` to `\bigcolheight`, which is a bit more than twice the original `\vsize`. This allows a very tall column to accumulate, after which, when `\doublecolumnout` is invoked, `\vsplit` is used to extract columns of the correct height.

When `\enddoublecolumns` occurs, the output routine `\balancecolumns` uses a `\loop` in an attempt to split the current `\box255` into two columns of equal height. Then `\pagesofar` packages these boxes side by side and contributes the result (along with the `\partialpage`, if any) to the current vertical list, and normal processing resumes.

The problem is that the alteration of `\vsize` by `\begindoublecolumns` doesn't take the height of `\partialpage` into account. This can allow `\box255` to grow too large, creating a situation that `\balancecolumns` can't handle. Consider a case where `\begindoublecolumns` has occured on a given page and more than enough material has accumulated in the main vertical list to fill the remainder of the page, if split equally. Since `\begindoublecolumns` has set `\vsize` for a *full* double-column page, this might not be enough material to cause the output routine to be invoked. If `\enddoublecolumns` occurs at this point, `\balancecolumns` will split the *entire* `\box255` into two equal parts, and the resulting columns won't fit onto the page along with `\partialpage`. The result is that `\partialpage` gets put onto the current page as a badly underfull `\vbox`, and the two columns get held over for the next page. You get something like this:



instead of this:



Of course, this situation never occurs in Appendix I because before `\enddoublecolumns` finally occurs, several full-size double-column pages have intervened, so that `\partialpage` is empty.