DIGITAL TYPOGRAPHY

Donald Ervin Knuth

I have been in love with books ever since I can remember. At first, my parents read to me a lot—an unusual practice in America at the time, because the prevailing "wisdom" of the 1940s was that a child who is exposed to intellectual things at an early age will be bored later when entering school. Thanks to my parents, I became at age four the youngest member of the Book Worm Club at the Milwaukee Public Library [photo 1].

That early experience with books is probably responsible for the fact that I don't remember ever being bored, throughout my education. In fact, I think contemporary society is all mixed up in its concept of "boredom": People often say to each other that they are bored, but to me this is almost a shocking, shameful admission. Why should it be somebody else's duty to entertain us? People who can't find anything of interest in what they are doing, who constantly need external sources of stimulation and amusement, are missing most of life's pleasures.

With me it has always been the opposite: I tend to err in the other direction. I often get so interested in Chapter 1 of the books that I'm reading or studying, I don't have much time to read the final chapters.

Once, when I was five years old, my parents let me take the streetcar to the downtown library by myself, and I was absolutely fascinated by the children's books. When I didn't come home on time, my parents were worried and phoned the library. One of the night staff went looking and found me in the stacks, reading happily—I had no idea that the library was closed and that everyone else had gone home! Even today my wife knows I go to the library, I'll probably come home late.

In fact, not only have I always loved books, I've also been in love with the individual *letters* in books. Here's a page from the first ABC alphabet book that I had when I was little [photo 2]. Curiously, I marked each serif in the letters with a little x, and I counted the serifs: The letter K has 7 serifs. The letter P [photo 3] has 4; the letter O [photo 4] has none.

From this you can see that I like numbers as well as letters. By the time I became a professor at Stanford I had learned that my main talents were associated with computer programming, and I had begun to write books of my own. My first book, Volume 1 of *The Art of Computer Programming*, came out in 1968, and Volume 2 was ready a year later [photo 5].

I was excited to see these volumes not only because I was pleased with the information they contained, but also because of the beautiful typography and layout. These books were produced with the best, time-tested methods known for the presentation of technical material. They appeared in the same classic style that had been used in my favorite college textbooks. So it was a pleasure to look at these volumes as well as to read them.

They were produced with 19th-century technology called Monotype, involving two kinds of machines. First, there was a complex pneumatic keyboard with 284 keys [photo 6]. This machine produced a punched paper tape something like a player-piano roll; you can see this tape at the top of the picture. The paper tape was then used to control at special casting machine [photo 7] that produced individual pieces of type from hot molten lead.

The process of typesetting mathematics with such machines was very complicated [photo 8]. First, a specially trained typist would key in the formula by making two pases: One for the letters and symbols on the main line, and their superscripts (the characters‘ $|\ \det(a\ )| \leqq a^{2'}$ high-lighted in yellow); a second pass was made for the subscripts (the characters ‘ij’ in this example). The keyboard operator had to know the width of each character so that he could leave just enough space to make the subscripts line up properly. After the formula had been cast into metal, another specially trained technician inserted the remaining large symbols (the big parentheses and symbols like Π and Σ) by hand. Only a few dozen people in the world knew how to typeset mathematical formulas with Monotype. I once had the privilege and pleasure of meeting Eric, the compositor who did the keyboarding for Volumes 1 and 2; I was surprised to discover that he spoke with a very strong London-Cockney accent, although he lived in America and was responsible for some of the world's most advanced books in mathematics.

Books on computer science added a new complication to the difficulties that printers already faced in mathematical typesetting: Computer scientist need to use a special style of type called typewriter type, in order to represent the textual material that machines deal with. For example [photo 9], here's another portion of a page from

Volume 2, part of a computer program. I needed to combine typewriter type like the word 'OFLO' with the ordinary style of letters. At first I was told that an extra alphabet would be impossible with Monotype, because traditional math formulas were already stretching Monotype technology to its limits. But later, Eric and his supervisor figured out how to do it. Notice that I needed a new, squarish looking letter O in the typewriter style, in order to make a clean distinction between O (oh) and 0 (zero).

New machines based on photography began to replace hot-lead machines like the Monotype in the 1960s. The new machines created pages by exposing a photographic plate, one letter at a time, using an ingenious combination of rotating disks and lenses to put each character in its proper position. Shortly after Volume 3 of *The Art of Computer Programming* came out in 1973, my publisher sold its Monotype machine and Eric had to find another job. New printings of Volume 1 and Volume 3 were published in 1975, correcting errors that readers had found in the earlier printings; these corrections were typeset in Europe, where Monotype technology still survived.

I had also prepared a second edition of Volume 2, which required typesetting that entire book all over again. My publishers found that it was too expensive in 1976 to produce a book the way it had been done in 1969. Moreover, the style of type that had been used in the original books was not available on photo-optical typesetting machines. I flew from California to Massachusetts for a crisis meeting. The publishers agreed that quality typography was of the utmost importance; and in the next months they tried hard to obtain new fonts that would match the old ones.

But the results were very disappointing. For example [photo 10], here's some of the type from the second, "tuned up" version of their new fonts. These were much improved from the first attempt, but still unacceptable. The "N" in "NOAM" was tipped; the "ff" in "effect" was much too dark; the letters "ip" in "multiple" were too close together; and so on.

I didn't know what to do. I had spent 15 years writing those books, but if they were going to look awful I didn't want to write any more. How could I be proud of such a product?

A possible way out of this dilemma presented itself a few months later, when I learned about another radical change in printing technology. The newest machines made images on film by digital instead of analog means—something like the difference between television and real movies. The shapes of letters were now made from tiny little dots, based on electronic pulses that were either ON or OFF [photo 11]. Aha! This was something I could understand! It was very simple, like the lights on a scoreboard at a sports match.

Metallurgy and hot lead have always been complete mysteries to me; neither have I understood lenses or mechanical alignment devices. But letters made of little dots—that's computer science! That's just bits, binary digits, 0s and 1s! Put a 1 where you want ink, put a 0 where you don't want ink, and you can print a page of a book!

I had seen digital letterforms before, but only on crude machines. Computer scientists had been experimenting for many years with a machine called the Xerox Graphics Printer, which had been invented in England about 1961 but not controlled by computers until the 70s. This machine made letters out of dots, but the dots weren't very small. There were only about 180 dots per inch, so the letters had lots of jagged edges. It was fun to play with the Xerox Graphics Printer, but I never expected that such a machine could produce real books. It seemed too simple, capable only of making cheap imitations—like the difference between an electronic synthesizer and a real piano or violin.

But in February 1977 I saw for the first time the output of a high-quality digital typesetter, which had more than 1,000 dots per inch... and it looked perfect, every bit as good as the best metal typography I had ever seen. Suddenly I saw that dots of ink will form smooth-looking curves if the dots are small enough, by the laws of physics. And 1 remembered that human eyes are inherently digital, made from individual rod and cone cells. Therefore I learned for the first time that a digital typesetting machine was indeed capable of producing books of the highest conceivable quality.

Digital cameras don't capture all the sharp details of traditional photographs. High-definition television can't match the quality of a Vista Vision movie. But for ink on paper, a digital approach is as good as any other.

In other words, the problem of printing beautiful books had changed from a problem of metallurgy to a problem of optics to a problem of computer science. The fact that Gutenberg had made books from movable metal type was suddenly only a 500-year-long footnote to history. The new machines have made the old mechanical approaches essentially irrelevant: The future of typography depends on the people who know the most about creating patterns of 0s and 1s; it depends on mathematicians and computer scientists.

When I realized this, I couldn't resist tackling the typography problem myself. I dropped everything else I was doing—I had just finished writing the first 100 pages of Volume 4—and decided to write computer programs that would generate the patterns of 0s and 1s that my publishers and I needed for the new edition of Volume 2.

At first I thought it would be easy; I expected that the job could be done in a few months. In March of 1977 I wrote to my publishers that I thought I would have first proofs ready in July. Boy, was I wrong! All my life I have underestimated the difficulty of the project I've embarked on, but this was a new personal record for being too optimistic.

In the first place, almost nobody else in computer science was doing this kind of work, so it was difficult to get financial support. The typesetting machine was very expensive, too much for our university budget. Moreover, that machine was designed to be run 24 hours per day by trained operators; I was just a single individual with strange mathematical ideas and no experience in the printing industry. Still, I assumed that if I could get my computer program working, I'd be able to borrow time on some digital typesetting machine.

There also was a chicken-and-egg problem. I couldn't set type until I had fonts of letters and mathematical symbols, but the fonts I needed did not exist in digital form. And I could not readily design the fonts until I could set type with them. I needed both things at once. Other fonts had been digitized, but I had resolved to define the fonts by myself, using purely mathematical formulas under my own control. Then I could never have to face the problem that another change in technology might upset the applecart again. With my own computer program controlling all aspects of the 0s and 1s on the pages, I would be able to define the appearance of my books once and for all.

My publishers provided me with original copies of the Monotype images that had been used to make the first edition of Volume 1 [photo 12]. So I thought it would be easy to find mathematical formulas to describe the shapes of the letters. I had seen John Warnock doing similar things at Xerox Research Center, so I asked if I could use Xerox's lab facilities to create my fonts. The answer was yes, but there was a catch: Xerox insisted on all rights to the use of any fonts that I developed with their equipment. Of course that was their privilege, but such a deal was unacceptable to me: A mathematical formula should never be “owned” by anybody! Mathematics belongs to God.

So I went to Stanford’s Artificial Intelligence lab, which had a television camera that I could use to magnify the letters and capture them in digital form. Unfortunately, the television camera did not give a true picture—the image was distorted. Even worse, a tiny change in the brightness of the room lights made a tremendous change in the television images. There was no way I could get consistent data from one letter to another. With that TV camera my fonts would look much worse than the fonts I had rejected from the non-digital machine.

I tried photographing the pages and magnifying them by projecting the images on the wall of my house, tracing the enlarged outlines with pencil and paper. But that didn’t work either.

Finally, a simple thought struck me. *Those letters were designed by people*. If I could understand what those people had in their minds when they were drawing the letters, then I could program a computer to carry out the same ideas. Instead of merely copying the form of the letters, my new goal was therefore to copy the intelligence underlying that form. I decided to learn what type designers knew, and to teach that knowledge to a computer.

That train of thought led to my computer system called METAFONT, which I want to try to show you now. Here is the way I finally desided to create the letter A, for example, using a computer program. All the key points of the letter are based on a grid that is displayed here [plate 1], although of course the grid is really invisible. Based on this grid and the specification of a normal text font, the computer first draws the main stem stroke [plate 2]. Part of this stroke needs to be erased because it’s too thick at the top [plate 3]. Then the left diagonal stroke is added [plate 4], and the crossbar [plate 5]. It’s time now to add a serif at the bottom left [plate 6, plate7], and to erase a little at the bottom [plate 8] so that the serif doesn’t make the letter too heavy. A similar serif is drawn at the bottom right [plate 9-plate 11]. This completes the letter A.

The same program will draw infinitely many different A’s if we change the specifications. For examples, here’s darker, boldface variant: [plate 12-plate 22]. And here’s a small A suitable for fine print [plate 23-plate 33]. Simply shrinking the original A by 50% would not produce such a legible character at a small size; good typography requires small letters to have subtly different shapes from their larger cousins.

Even the typewriter style A can be drawn with same program. This time we specify that the thick strokes and thin strokes are identical, and the corners of the serifs are rounded [plate 34-plate 42]. The resulting A went into my first typewriter-style font, but I learned later that such an A was a bit darker than it should be. To solve the problem, I moved the two diagonal strokes slightly apart, and cut a "notch" in the interior so as to open the inside a bit [plate 43-plate 49]. This is the nice typewriter-style A that I use today. I didn't learn such tricks until several years after I started to study type design.

Here is an example of the way my first draft fonts looked on the Xerox Graphics Printer, about one and a half years after I had begun to work on typography [photo 13]. Two years later, with some financial help from my publishers, my project was finally able to obtain a high-resolution digital typesetting machine, and I could print the new edition of Volume 2 [photo 14]. The proofs for that book looked so much better than the xerographic proofs I had been working with, I thought my goals for quality typography had finally been reached.

But when I received the first printed copy of the new Volume 2 in its familiar binding, and opened the pages, I burned with disappointment. The book did not look at all as I had hoped. After four years of hard work, I still hadn't figured out how to generate the patterns of 0s and 1s that are demanded by fine printing. The published second edition didn't look much better than the version I had rejected before starting my typography project.

Meanwhile I had had the good fortune to meet many of the world's leading type designers. They graciously gave me the instruction and criticism I needed as I continued to make improvements. After five more years went by, I finally was able to produce books of which I could feel proud.

I don't want to give the impression that those nine years of work were nothing but drudgery. (As I said before, I raely seem to get bored.) Font design is in fact lots of fun, especially when you make mistakes. The computer tends to draw delightfully creative images that no human being would ever dream up. I call these "meta-flops." For example [photo 15], here's an ffi ligature combination in which the f at the left reaches all the way over to the dot on the i at the right. And here's another weird ffi [photo 16]: I call it "the ffilling station."

In one of my first attempts to do a capital typewriter-style Y, I put the upper right serif in the wrong place [photo 17]. I swear that I was *not* thinking of yen when I did this!

Does METAFONT work for Japanese characters as well as for Roman letters? I think it does, but I haven't been able to develop a good eye for Asian letterforms myself. My student John Hobby did some promising experiments together with Gu Guoan of the Shanghai Printing Company, and I'd like to give you a taste of what they did. First they wrote 13 computer programs for basic strokes. For example, here are two "teardrop" shapes produced by one of their programs [photo 18]. A font designer specifies the top, the bottom, and the edge of the bulb; the computer does the rest. Here [photo 19] are some more examples of teardrops, together with variants of three other basic strokes.

Hobby and Gu used their stroke routines to design 128 Chinese characters. And they did it in such a way that you could get three different styles of letters simply by using three different versions of the 13 basic strokes. Here [photo 20] are five characters rendered in Song style, Long Song style, and Bold style. And here [photo 21] are examples of the 13 basic strokes in all three styles.

With the METAFONT system for type design, and the TEX system for putting letters and symbols into the right positions on a page, anybody who wants to write a beautiful book can now do so singlehandedly with a reasonable amount of effort. These systems give an author total control over the patterns of 0s and 1s that are needed to define the pages. I have made special efforts to ensure that TEX and METAFONT will give exactly the same results on all computers, and to ensure that they will give the same results 50 years from today as they do today. Furthermore I have published all of the details and put all of my programs in the public domain, so that nobody has to pay for using them. Of course, many people who offer additional services will charge money for their expertise, but the main point is that a dedicated author now has the power to prepare books that previously were prohibitively expensive.

I can't resist showing you samples from some of the books that I've received in recent years from their authors. Here's one from the Czech Republic [photo 22], showing another font done with the METAFONT SYSTEM. Here's one from Ethiopia [photo 23], telling the people of that country how to use the TEX system. Here [photo 24] is part of the Russian translation of my own book on TEX. And here [photo 25] is the same passage in Japanese translation. By the way, if I had lived in Japan, I'm sure I never would have been inclined to invent TEX or MATAFONT, because I wouldn't have felt the need: The standards of typography in this country never declined as they did in America and Europe. However, I'm glad to see that TEX is now widely used in Japanese publishing.

People have sent me many fine books that probably never have existed without TEX and METAFONT. My favorite examples are scholarly publications, such as the interlingual text of an Eskimo language folk tale shown here [photo 26]. Here, similarly, are some footnotes from a critical edition of a Greek text [photo 27]; another, in Arabic [photo 28]; another in Sanskrit [photo 29].

Ever since I began working on TEX in 1977, I've kept a record of all the errors, large and small, that I found and removed from the program with the help of volunteers around the world. This list has now grown to 1,276 items. Perhaps TEX has thereby become one of the most thoroughly checked computer programs ever written.

I would like to conclude this talk by quoting one of my favorite poems, written by the Danish sage Piet Hein. He calls it a "grook"—it's sort of a Danish variant of *haiku*. My wife and I like it so much, we commissioned a British stonecutter to carve it in slate for the entryway of our house [photo 30]. It goes like this:

The road to wisdom? Well it's plain
and simple to express:
Err
and err
and err again
but less
and less
and less.

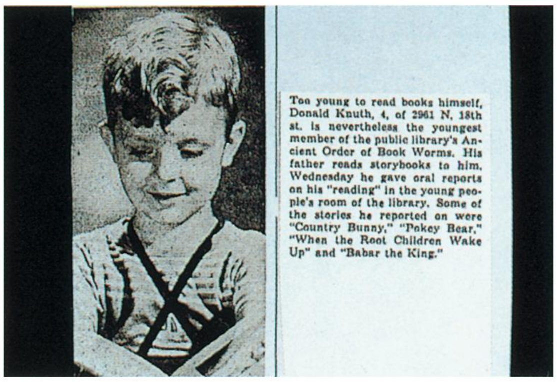

Too young to read books himself, Donald Knuth, 4, of 2961 N. 18th st. is nevertheless the youngest member of the public library's Ancient Order of Book Worms. His father reads storybooks to him. Wednesday he gave oral reports on his "reading" in the young people's room of the library. Some of the stories he reported on were "Country Bunny," "Pokey Bear," "When the Root Children Wake Up" and "Babar the King."

photo 1

photo 2

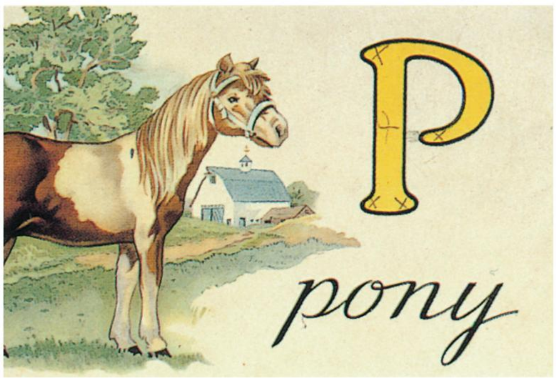


photo 3

photo 4

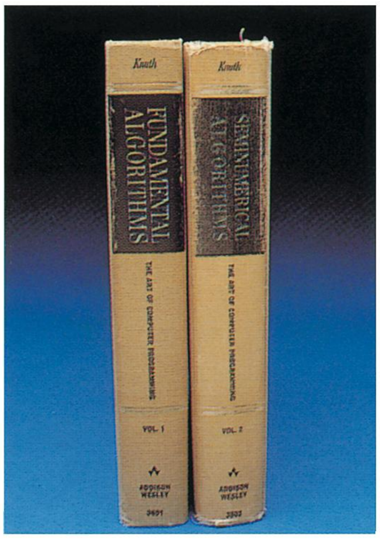


photo 5

photo 6

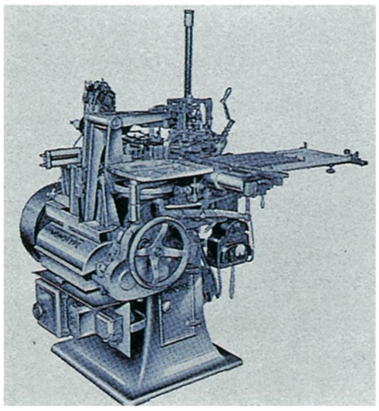
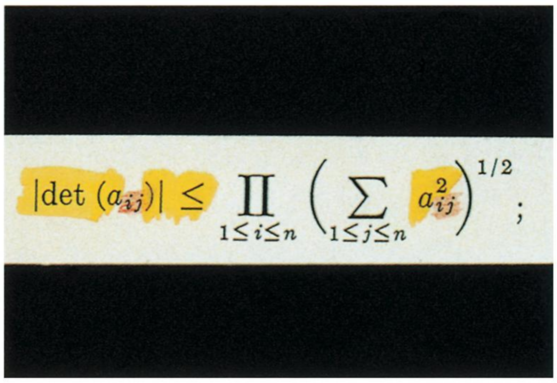
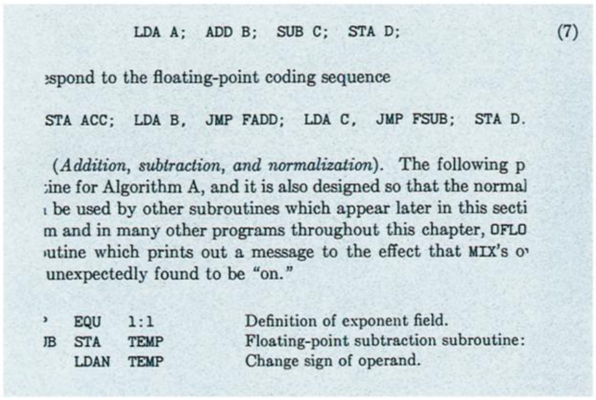


photo 7



$$|\det(a_{ij})| \le \prod_{1\le i\le n} \Big(\sum_{1\le j\le n} a_{ij}^2 \Big)^{1/2} ;$$

photo 8



```
        LDA A;  ADD B;  SUB C;  STA D;                        (7)
```

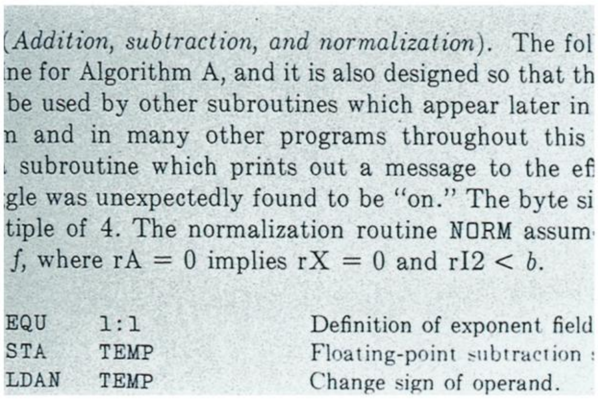
espond to the floating-point coding sequence

```
STA ACC;  LDA B,  JMP FADD;  LDA C,  JMP FSUB;  STA D.
```

(*Addition, subtraction, and normalization*). The following p
ine for Algorithm A, and it is also designed so that the normal
be used by other subroutines which appear later in this secti
m and in many other programs throughout this chapter, OFLO
utine which prints out a message to the effect that MIX's o
unexpectedly found to be "on."

```
   EQU   1:1          Definition of exponent field.
JB STA   TEMP         Floating-point subtraction subroutine:
   LDAN  TEMP         Change sign of operand.
```

photo 9



(*Addition, subtraction, and normalization*). The fol
ne for Algorithm A, and it is also designed so that th
be used by other subroutines which appear later in
n and in many other programs throughout this
subroutine which prints out a message to the ef
gle was unexpectedly found to be "on." The byte si
tiple of 4. The normalization routine NORM assum
f, where rA = 0 implies rX = 0 and rI2 < *b*.

```
EQU    1:1        Definition of exponent field
STA    TEMP       Floating-point subtraction
LDAN   TEMP       Change sign of operand.
```

photo 10

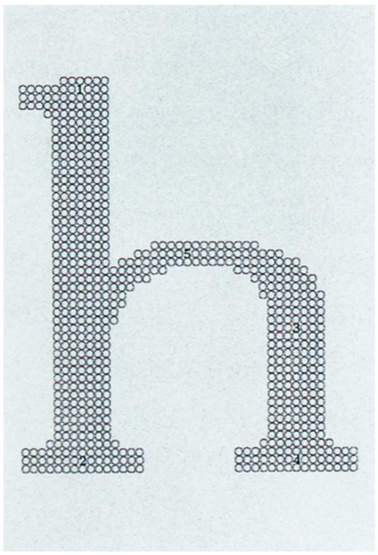
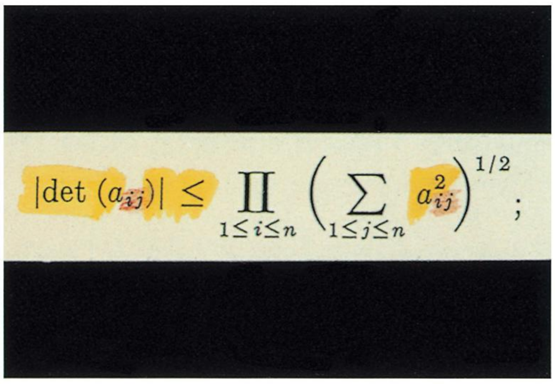
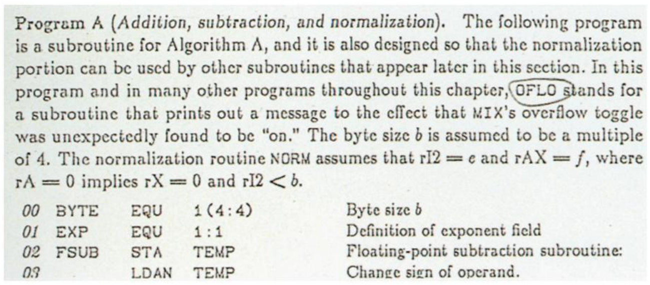


photo 11



$$|\det(a_{ij})| \le \prod_{1\le i\le n} \Big(\sum_{1\le j\le n} a_{ij}^2 \Big)^{1/2} ;$$

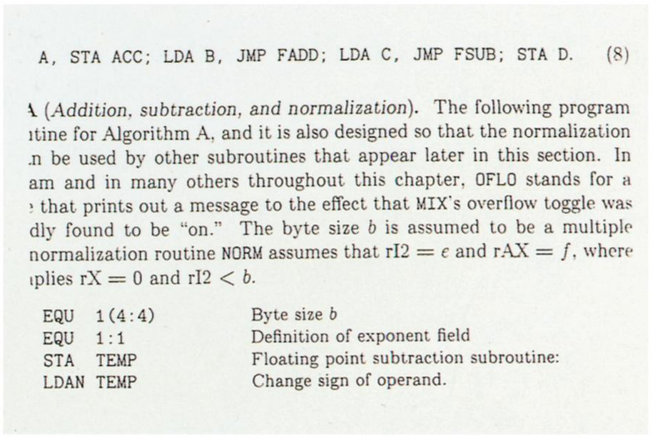
photo 12



Program A (*Addition, subtraction, and normalization*). The following program is a subroutine for Algorithm A, and it is also designed so that the normalization portion can be used by other subroutines that appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine that prints out a message to the effect that MIX's overflow toggle was unexpectedly found to be "on." The byte size b is assumed to be a multiple of 4. The normalization routine NORM assumes that rI2 $= e$ and rAX $= f$, where rA $= 0$ implies rX $= 0$ and rI2 $< b$.

```
00  BYTE   EQU   1(4:4)          Byte size b
01  EXP    EQU   1:1             Definition of exponent field
02  FSUB   STA   TEMP            Floating-point subtraction subroutine:
03         LDAN  TEMP            Change sign of operand.
```

photo 13



```
A, STA ACC; LDA B, JMP FADD; LDA C, JMP FSUB; STA D.     (8)
```

A (*Addition, subtraction, and normalization*). The following program
itine for Algorithm A, and it is also designed so that the normalization
.n be used by other subroutines that appear later in this section. In
am and in many others throughout this chapter, OFLO stands for a
: that prints out a message to the effect that MIX's overflow toggle was
dly found to be "on." The byte size b is assumed to be a multiple
normalization routine NORM assumes that rI2 $= e$ and rAX $= f$, where
iplies rX $= 0$ and rI2 $< b$.

```
EQU   1(4:4)        Byte size b
EQU   1:1           Definition of exponent field
STA   TEMP          Floating point subtraction subroutine:
LDAN  TEMP          Change sign of operand.
```

photo 14

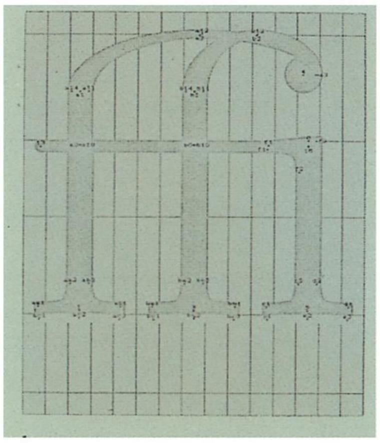


photo 15

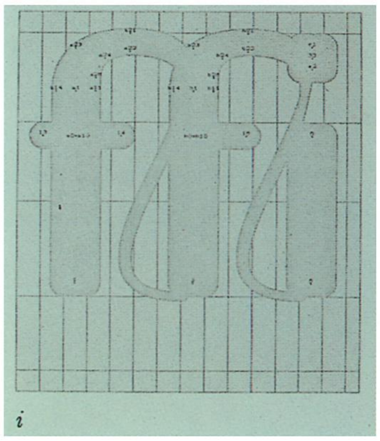


photo 16

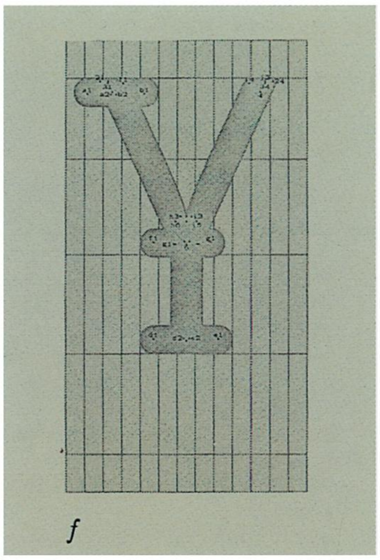


photo 17

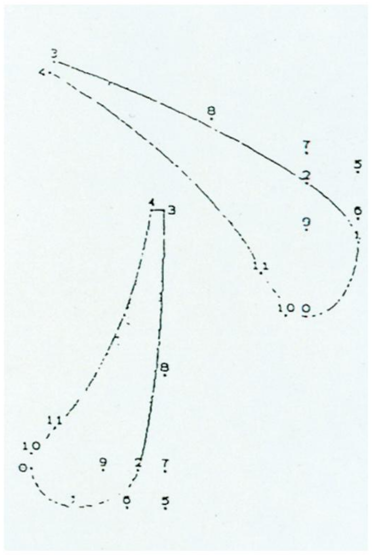


photo 18

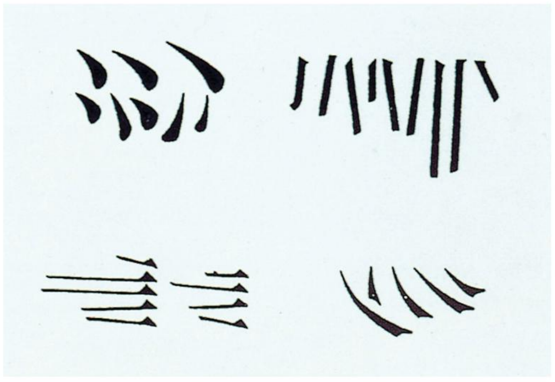


photo 19

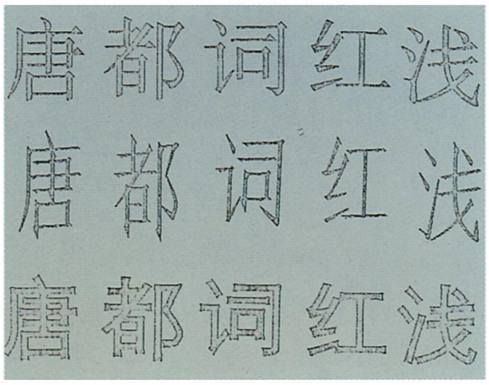


photo 20

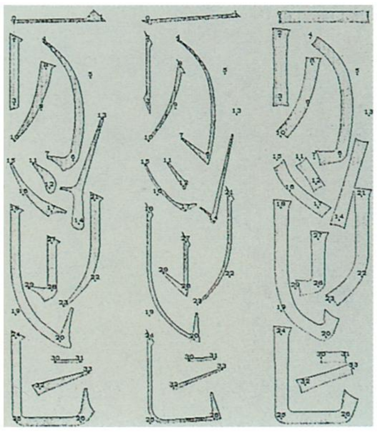
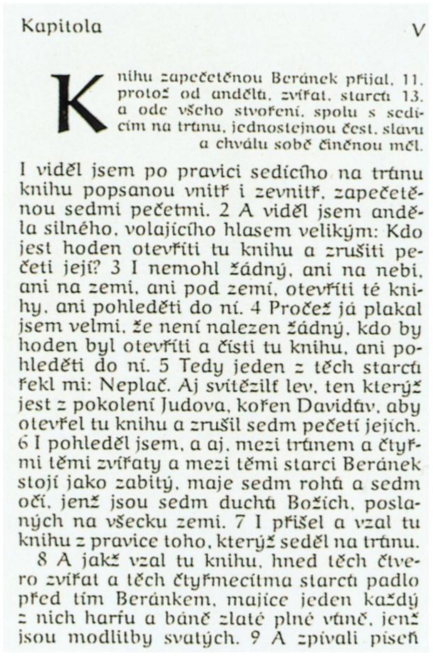


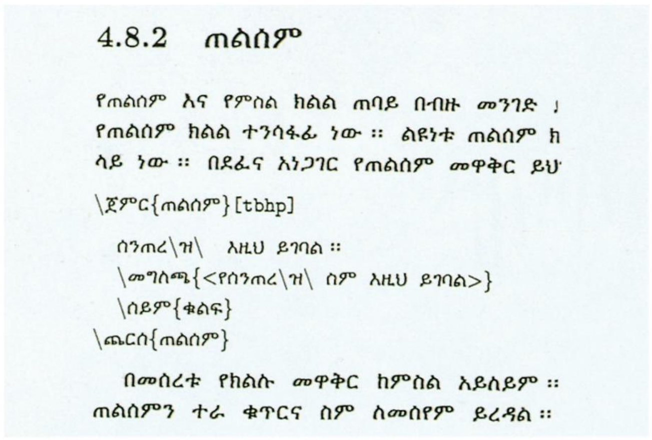
photo 21


Kapitola V

Knihu zapečetěnou Beránek přijal, 11. protož od andělů, zvířat, starců 13. a ode všeho stvoření, spolu s sedícím na trůnu, jednostejnou čest, slávu a chválu sobě činěnou měl.

I viděl jsem po pravici sedícího na trůnu knihu popsanou vnitř i zevnitř, zapečetěnou sedmi pečetmi. 2 A viděl jsem anděla silného, volajícího hlasem velikým: Kdo jest hoden otevříti tu knihu a zrušiti pečeti její? 3 I nemohl žádný, ani na nebi, ani na zemi, ani pod zemí, otevříti té knihy, ani pohleděti do ní. 4 Pročež já plakal jsem velmi, že není nalezen žádný, kdo by hoden byl otevříti a čísti tu knihu, ani pohleděti do ní. 5 Tedy jeden z těch starců řekl mi: Neplač. Aj svítězilť lev, ten kterýž jest z pokolení Judova, kořen Davidův, aby otevřel tu knihu a zrušil sedm pečetí jejích. 6 I pohleděl jsem, a aj, mezi trůnem a čtyřmi těmi zvířaty a mezi těmi starci Beránek stojí jako zabitý, maje sedm rohů a sedm očí, jenž jsou sedm duchů Božích, poslaných na všecku zemi. 7 I přišel a vzal tu knihu z pravice toho, kterýž seděl na trůnu.
8 A jakž vzal tu knihu, hned těch čtvero zvířat a těch čtyřmecítma starců padlo před tím Beránkem, majíce jeden každý z nich harfu a báně zlaté plné vůně, jenž jsou modlitby svatých. 9 A zpívali píseň

photo 22

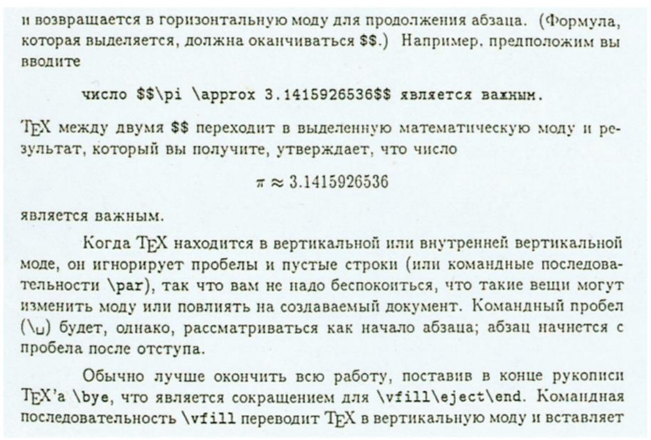

4.8.2 ጠልሰም

የጠልሰም እና የምስል ክልል ጠባይ በብዙ መንገድ ፣ የጠልሰም ክልል ተንሳፋፊ ነው ። ልዩነቱ ጠልሰም ክ ላይ ነው ። በደፈና አነጋገር የጠልሰም መዋቅር ይህ፡

```
\ጀምር{ጠልሰም}[tbhp]
  ሰንጠረ\ዥ\  እዚህ ይገባል ።
  \መግለጫ{<የሰንጠረ\ዥ\ ስም እዚህ ይገባል>}
  \ሰይም{ቁልፍ}
\ጨርስ{ጠልሰም}
```

በመሰረቱ የክልሉ መዋቅር ከምስል አይለይም ። ጠልሰምን ተራ ቁጥርና ስም ለመሰየም ይረዳል ።

photo 23


и возвращается в горизонтальную моду для продолжения абзаца. (Формула, которая выделяется, должна оканчиваться `$$`.) Например, предположим вы вводите

```
число $$\pi \approx 3.1415926536$$ является важным.
```

TEX между двумя `$$` переходит в выделенную математическую моду и результат, который вы получите, утверждает, что число

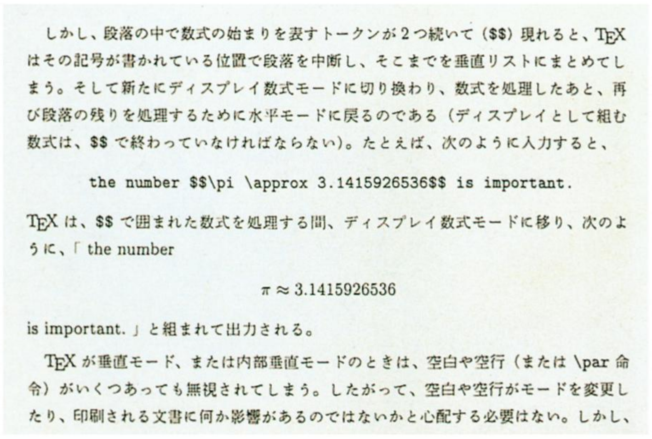
$$\pi \approx 3.1415926536$$

является важным.

Когда TEX находится в вертикальной или внутренней вертикальной моде, он игнорирует пробелы и пустые строки (или командные последовательности `\par`), так что вам не надо беспокоиться, что такие вещи могут изменить моду или повлиять на создаваемый документ. Командный пробел (`\␣`) будет, однако, рассматриваться как начало абзаца; абзац начнется с пробела после отступа.

Обычно лучше окончить всю работу, поставив в конце рукописи TEX'a `\bye`, что является сокращением для `\vfill\eject\end`. Командная последовательность `\vfill` переводит TEX в вертикальную моду и вставляет

photo 24



しかし、段落の中で数式の始まりを表すトークンが２つ続いて（$$）現れると、TEX はその記号が書かれている位置で段落を中断し、そこまでを垂直リストにまとめてしまう。そして新たにディスプレイ数式モードに切り換わり、数式を処理したあと、再び段落の残りを処理するために水平モードに戻るのである（ディスプレイとして組む数式は、$$ で終わっていなければならない）。たとえば、次のように入力すると、

```
the number $$\pi \approx 3.1415926536$$ is important.
```

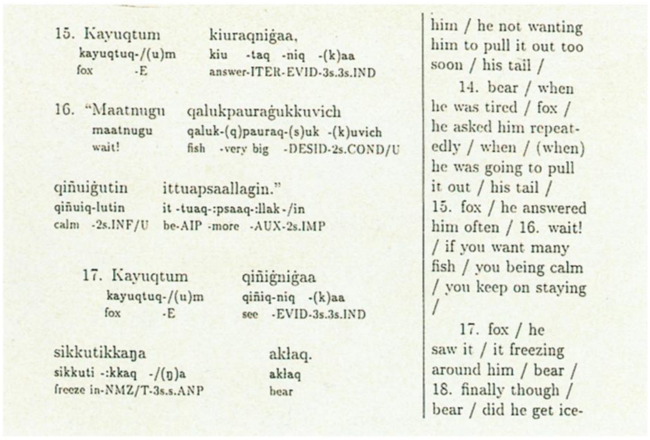
TEX は、$$ で囲まれた数式を処理する間、ディスプレイ数式モードに移り、次のように、「the number

$$\pi \approx 3.1415926536$$

is important.」と組まれて出力される。

TEX が垂直モード、または内部垂直モードのときは、空白や空行（または \par 命令）がいくつあっても無視されてしまう。したがって、空白や空行がモードを変更したり、印刷される文書に何か影響があるのではないかと心配する必要はない。しかし、

photo 25



15. Kayuqtum kiuraqniġaa,
kayuqtuq-/(u)m kiu -taq -niq -(k)aa
fox -E answer-ITER-EVID-3s.3s.IND

16. "Maatnugu qalukpauraġukkuvich
maatnugu qaluk-(q)pauraq-(s)uk -(k)uvich
wait! fish -very big -DESID-2s.COND/U

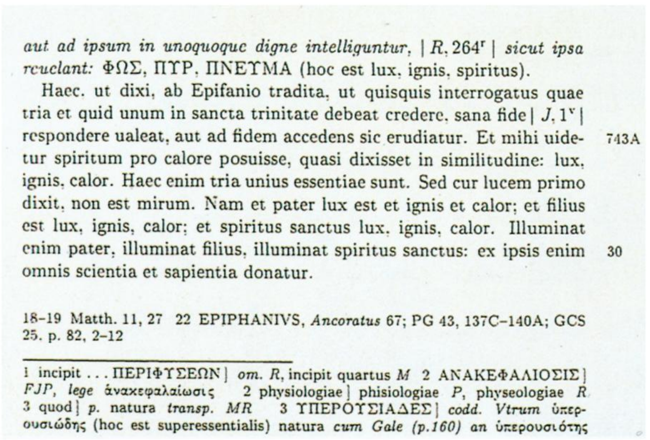
qiñuiġutin ittuapsaallagin."
qiñuiq-lutin it -tuaq-:psaaq-:llak-/in
calm -2s.INF/U be-AIP -more -AUX-2s.IMP

17. Kayuqtum qiñiġniġaa
kayuqtuq-/(u)m qiñiq-niq -(k)aa
fox -E see -EVID-3s.3s.IND

sikkutikkaŋa akłaq.
sikkuti -:kkaq -/(ŋ)a akłaq
freeze in-NMZ/T-3s.s.ANP bear

him / he not wanting him to pull it out too soon / his tail /
14. bear / when he was tired / fox / he asked him repeatedly / when / (when) he was going to pull it out / his tail / 15. fox / he answered him often / 16. wait! / if you want many fish / you being calm / you keep on staying /
17. fox / he saw it / it freezing around him / bear / 18. finally though / bear / did he get ice-

photo 26



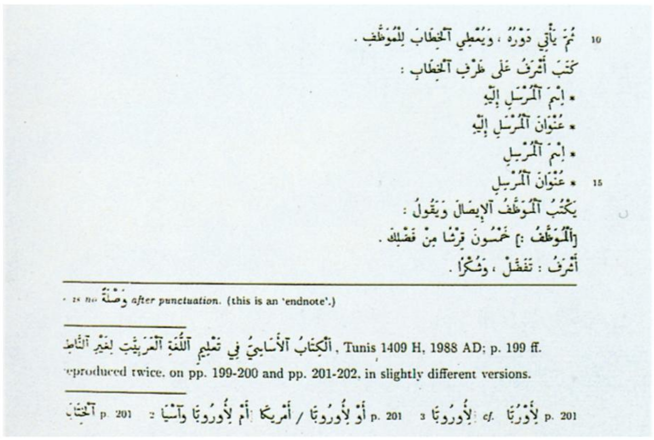
aut ad ipsum in unoquoque digne intelliguntur, | *R*, 264ʳ | *sicut ipsa reuelant:* ΦΩΣ, ΠΥΡ, ΠΝΕΥΜΑ (hoc est lux, ignis, spiritus).

Haec, ut dixi, ab Epifanio tradita, ut quisquis interrogatus quae tria et quid unum in sancta trinitate debeat credere, sana fide | *J*, 1ᵛ | respondere ualeat, aut ad fidem accedens sic erudiatur. Et mihi uidetur spiritum pro calore posuisse, quasi dixisset in similitudine: lux, ignis, calor. Haec enim tria unius essentiae sunt. Sed cur lucem primo dixit, non est mirum. Nam et pater lux est et ignis et calor; et filius est lux, ignis, calor; et spiritus sanctus lux, ignis, calor. Illuminat enim pater, illuminat filius, illuminat spiritus sanctus: ex ipsis enim omnis scientia et sapientia donatur. 743A

18–19 Matth. 11, 27 22 EPIPHANIVS, *Ancoratus* 67; PG 43, 137C–140A; GCS 25, p. 82, 2–12

1 incipit . . . ΠΕΡΙΦΥΣΕΩΝ] *om. R*, incipit quartus *M* 2 ΑΝΑΚΕΦΑΛΙΟΣΙΣ] *FJP*, *lege* ἀνακεφαλαίωσις 2 physiologiae] phisiologiae *P*, physeologiae *R* 3 quod] *p.* natura *transp. MR* 3 ΥΠΕΡΟΥΣΙΑΔΕΣ] *codd. Vtrum* ὑπερουσιώδης (hoc est superessentialis) natura *cum Gale (p.160) an* ὑπερουσιότης

photo 27



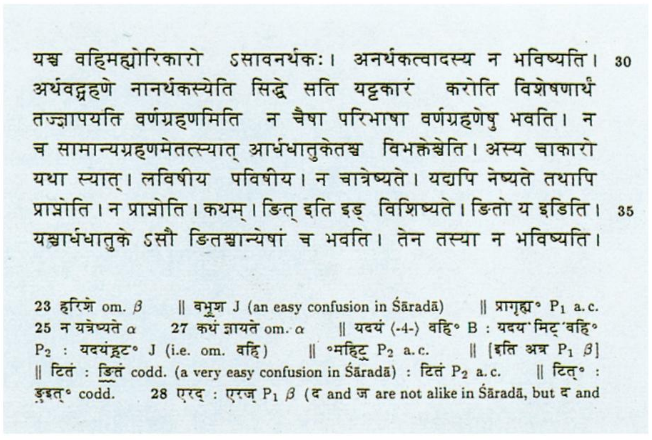
ثُمَّ يَأْتِي دَوْرُهُ ، وَيُعْطِي ٱلْخِطَابَ لِلْمُوَظَّفِ .
كَتَبَ أَشْرَفُ عَلَى ظَرْفِ ٱلْخِطَابِ :
• إِسْمَ ٱلْمُرْسَلِ إِلَيْهِ
• عُنْوَانَ ٱلْمُرْسَلِ إِلَيْهِ
• إِسْمَ ٱلْمُرْسِلِ
• عُنْوَانَ ٱلْمُرْسِلِ
يَكْتُبُ ٱلْمُوَظَّفُ ٱلْإِيصَالَ وَيَقُولُ :
[ٱلْمُوَظَّفُ :] خَمْسُونَ قِرْشًا مِنْ فَضْلِكَ .
أَشْرَفُ : تَفَضَّلْ ، وَشُكْرًا .

· is no وَصْلَة *after punctuation.* (this is an 'endnote'.)

ٱلْكِتَابُ ٱلْأَسَاسِيُّ فِي تَعْلِيمِ ٱللُّغَةِ ٱلْعَرَبِيَّةِ لِغَيْرِ ٱلنَّاطِ, Tunis 1409 H, 1988 AD; p. 199 ff.
eproduced twice, on pp. 199-200 and pp. 201-202, in slightly different versions.

لِأُورُبَّا p. 201 *cf.* لِأُورُوبَّا 3 أَوْ لِأُورُوبَّا / أَمْرِيكَا أَمْ لِأُورُوبَّا وَآسْيَا p. 201 2 ٱلْخِتَابِ p. 201

photo 28



यश्च वहिमह्योरिकारो ऽसावनर्थकः। अनर्थकत्वादस्य न भविष्यति। अर्थवद्ग्रहणे नानर्थकस्येति सिद्धे सति यट्टकारं करोति विशेषणार्थं तज्ज्ञापयति वर्णग्रहणमिति न चैषा परिभाषा वर्णग्रहणेषु भवति। न च सामान्यग्रहणमेतत्स्यात् आर्धधातुकेतश्च विभक्तेश्चेति। अस्य चाकारो यथा स्यात्। लविषीय पविषीय। न चात्रेष्यते। यद्यपि नेष्यते तथापि प्राप्नोति। न प्राप्नोति। कथम्। ङित् इति इड् विशिष्यते। ङितो य इडिति। यश्चार्धधातुके ऽसौ ङितश्चान्येषां च भवति। तेन तस्या न भविष्यति।

23 हरिशे om. β ‖ वभूश J (an easy confusion in Śāradā) ‖ प्रागृह्य° P_1 a.c. 25 न यत्रेष्यते α 27 कथं ज्ञायते om. α ‖ यदयं ⟨-4-⟩ वहि° B : यदय'मिट्'वहि° P_2 : यदयंह्रट° J (i.e. om. वहि) ‖ °महिट्ट P_2 a.c. ‖ [इति अत्र P_1 β] ‖ टितं : ङितं codd. (a very easy confusion in Śāradā) : टितं P_2 a.c. ‖ टित्° : ङ्इत्° codd. 28 एरद् : एरज् P_1 β (द and ज are not alike in Śāradā, but द and

photo 29

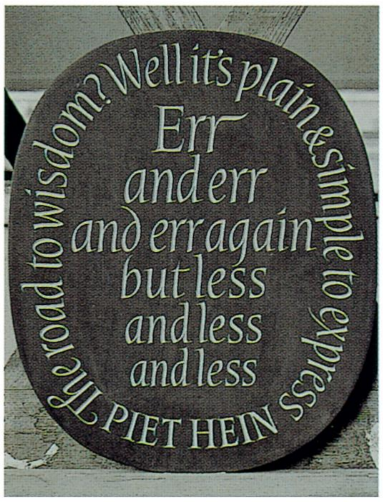


photo 30

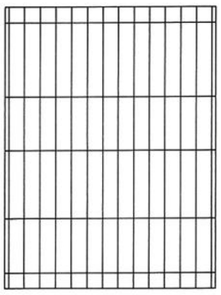


plate 1

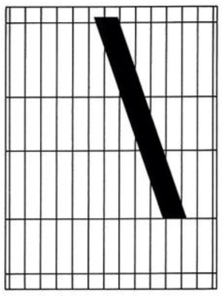


plate 2

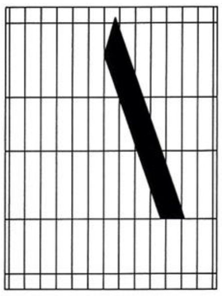


plate 3

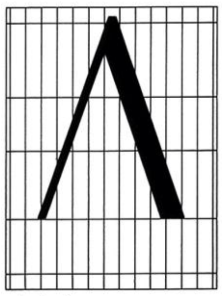


plate 4

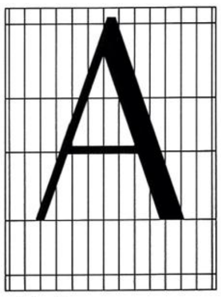


plate 5

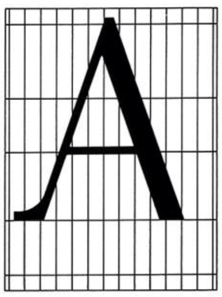


plate 6

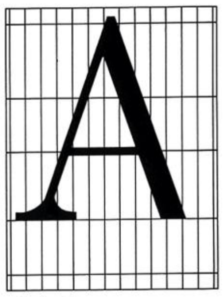


plate 7

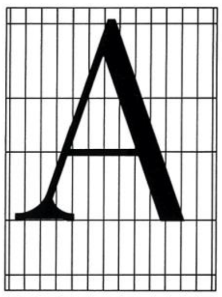


plate 8

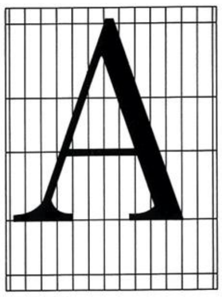


plate 9

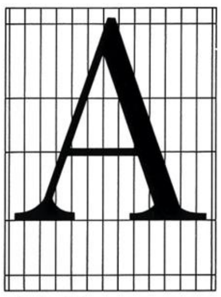


plate 10

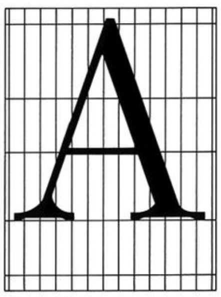


plate 11

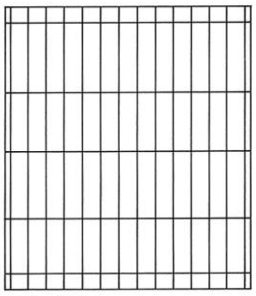


plate 12

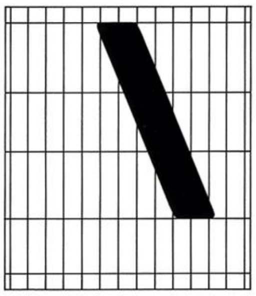


plate 13

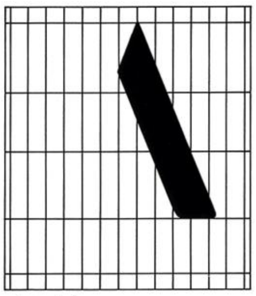


plate 14

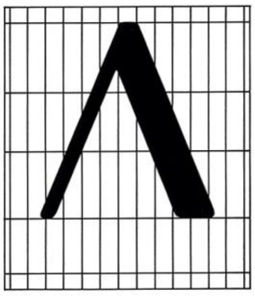


plate 15

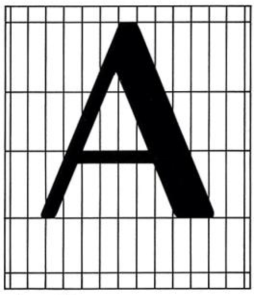


plate 16

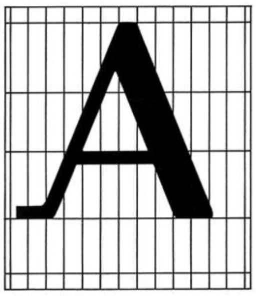


plate 17

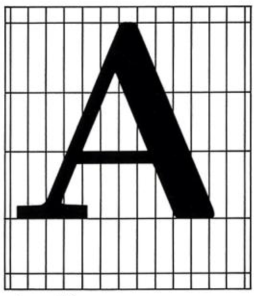


plate 18

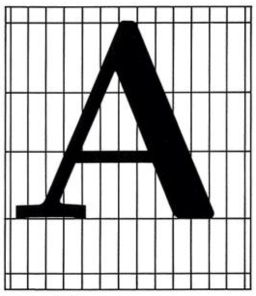


plate 19

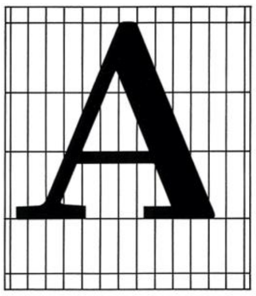


plate 20

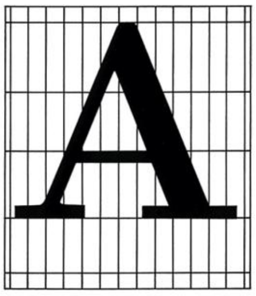


plate 21

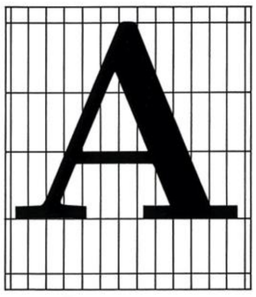


plate 22

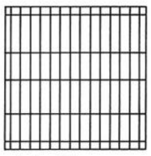


plate 23

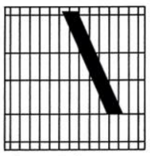


plate 24

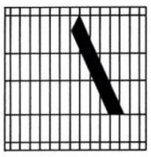


plate 25

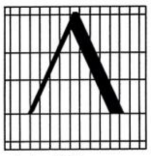


plate 26

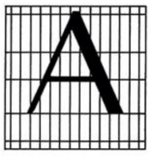


plate 27

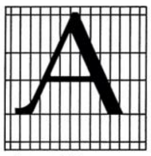


plate 28

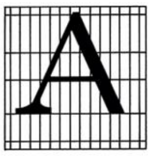


plate 29

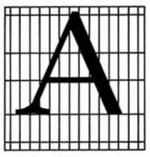


plate 30

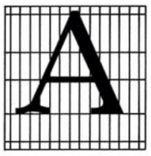


plate 31

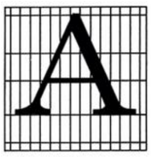


plate 32

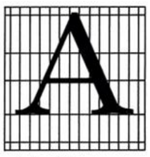


plate 33

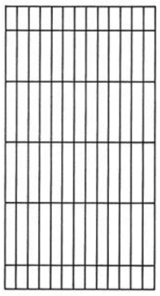


plate 34

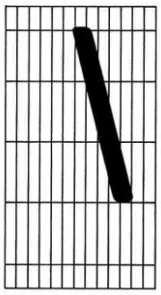


plate 35

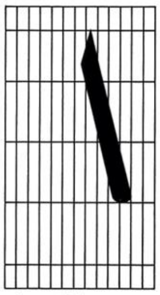


plate 36

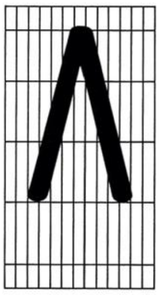


plate 37

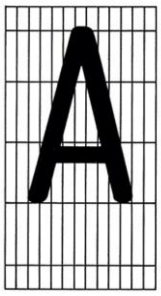


plate 38

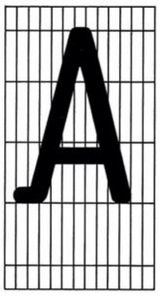


plate 39

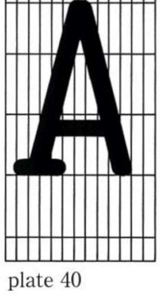


plate 40

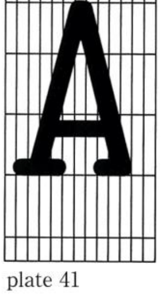


plate 41

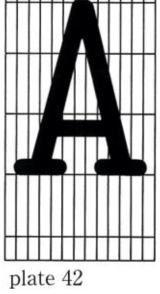


plate 42

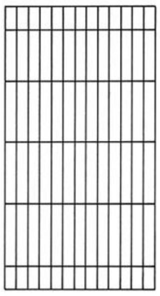


plate 43

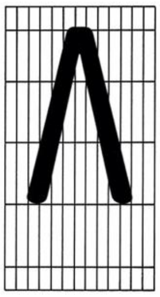


plate 44

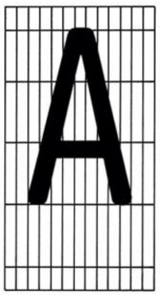


plate 45

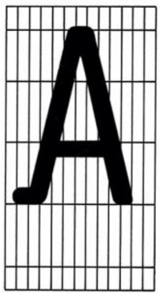


plate 46

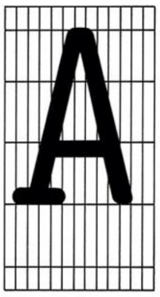


plate 47

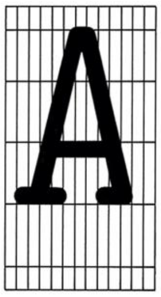


plate 48

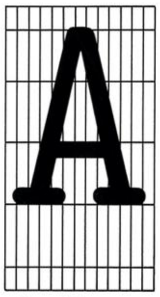


plate 49