Thus, a fixed-point coding sequence

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

would correspond to the floating-point coding sequence

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

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 which appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine which prints out a message to the effect that MIX's overflow toggle was unexpectedly found to be "on."

```
01   EXP   EQU   1:1          Definition of exponent field.
02   FSUB  STA   TEMP         Floating-point subtraction subroutine:
03         LDAN  TEMP         Change sign of operand.
04   FADD  STJ   EXITF        Floating-point addition subroutine:
05         JOV   OFLO         Ensure overflow is off.
06         STA   TEMP         TEMP ← v.
07         LDX   ACC          rX ← u.
08         CMPA  ACC(EXP)     Steps A1, A2, A3 are combined here:
09         JLE   1F           Jump if e_v ≤ e_u.
10         STA   FU(0:4)      FU ← ± f f f f 0 (u, v interchanged).
11         LD2   TEMP(EXP)    rI2 ← e_w.
12         STX   FV(0:4)
13         LD1N  ACC(EXP)     rI1 ← −e_v.
14         JMP   4F
15   1H    STX   FU(0:4)      FU ← ± f f f f 0.
16         LD2   ACC(EXP)     rI2 ← e_w.
17         JE    3F           Jump if e_u = e_v (saves time).
18         STA   FV(0:4)
19         LD1N  TEMP(EXP)    rI1 ← −e_v.
20   4H    INC1  0,2          rI1 ← e_u − e_v. (Step A4 unnecessary.)
21   5H    LDA   FV           A5. Scale right.
22         LDX   FV(0:0)      Set rX to zero with sign of v.
23         SRAX  0,1          Shift right e_u − e_v places.
24         JMP   6F
25   3H    SLA   1            Align radix point.
26         ENTX  0            Clear rX.
27   6H    ADD   FU           A6. Add.
28         JOV   N4           A7. Normalize. Jump if fraction overflow.
29         CMPA  =0=(1:1)
30         JNE   N5           Jump if f is normalized.
```

be expressed as a normalized floating point number in the required range, special action is necessary.)

N7. [Pack.] Put e and f together into the desired output representation. ∎

Some simple examples of floating point addition are given in exercise 4.

The following MIX subroutines, for addition and subtraction of numbers having the form (4), show how Algorithms A and N can be expressed as computer programs. The subroutines below are designed to take one input u from symbolic location ACC, and the other input v comes from register A upon entrance to the subroutine. The output w appears both in register A and location ACC. Thus, a fixed point coding sequence

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

would correspond to the floating point coding sequence

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

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 others 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.
04  FADD  STJ  EXITF         Floating point addition subroutine:
05        JOV  OFLO          Ensure overflow is off.
06        STA  TEMP          TEMP ← v.
07        LDX  ACC           rX ← u.
08        CMPA ACC(EXP)      Steps A1, A2, A3 are combined here:
09        JGE  1F            Jump if e_v ≥ e_u.
10        STX  FU(0:4)       FU ← ±fffff0.
11        LD2  ACC(EXP)      rI2 ← e_w.
12        STA  FV(0:4)
13        LD1N TEMP(EXP)     rI1 ← −e_v.
14        JMP  4F
15  1H    STA  FU(0:4)       FU ← ±fffff0 (u, v interchanged).
16        LD2  TEMP(EXP)     rI2 ← e_w.
17        STX  FV(0:4)
18        LD1N ACC(EXP)      rI1 ← −e_v.
```

A specially trained typist would key in most of the formula by making two passes: First the letters and symbols on the main line would be entered, and their superscripts (namely the characters

$$|\det(a\quad)| \leq \qquad\qquad a^2$$

in this case); then a second pass was made for the subscripts (namely the characters '$_{ij}$', repeated twice here). The keyboard operator had to know the width of each character so that there would be 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 $\prod$ and $\sum$) by hand.

Only a few dozen people in the world knew how to typeset mathematical formulas with Monotype. I once had the pleasure and privilege 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.

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 which appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine which prints out a message to the effect that MIX's overflow toggle was unexpectedly found to be "on."

```
01   EXP    EQU   1:1        Definition of exponent field.
02   FSUB   STA   TEMP       Floating-point subtraction subroutine:
03          LDAN  TEMP       Change sign of operand.
```

Slide 9.

Books on computer science have added a new complication to the difficulties that printers already faced in mathematical typesetting: Computer scientists need to use a special style of type called `typewriter type`, in order to represent the textual material that machines deal with. For example [SLIDE 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.

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 which appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine which 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.

```
01 EXP    EQU   1:1          Definition of exponent field.
02 FSUB   STA   TEMP         Floating-point subtraction subroutine.
03        LDAN  TEMP         Change sign of operand.
```

Slide 10.

But the results were very disappointing. For example [SLIDE 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 "NORM" 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,

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 $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

```
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.
```

Slide 12.

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 [SLIDE 13] didn't look much better than the version I had rejected before starting my typography project.

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 others 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 $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

```
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.
```

Slide 13.

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 rarely 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 "METAFLOPS." For example [SLIDE 14], 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 [SLIDE 15]: 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 [SLIDE 16]. I swear that I was *not* thinking of yen when I did this!

be expressed as a normalized floating point number in the required range, special action is necessary.)

N7. [Pack.] Put e and f together into the desired output representation. ∎

Some simple examples of floating point addition are given in exercise 4.

The following MIX subroutines, for addition and subtraction of numbers having the form (4), show how Algorithms A and N can be expressed as computer programs. The subroutines below are designed to take one input u from symbolic location ACC, and the other input v comes from register A upon entrance to the subroutine. The output w appears both in register A and location ACC. Thus, a fixed point coding sequence

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

would correspond to the floating point coding sequence

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

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 others 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.
04  FADD  STJ  EXITF         Floating point addition subroutine:
05        JOV  OFLO          Ensure overflow is off.
06        STA  TEMP          TEMP ← v.
07        LDX  ACC           rX ← u.
08        CMPA ACC(EXP)      Steps A1, A2, A3 are combined here:
09        JGE  1F            Jump if e_v ≥ e_u.
10        STX  FU(0:4)       FU ← ±fffff0.
11        LD2  ACC(EXP)      rI2 ← e_w.
12        STA  FV(0:4)
13        LD1N TEMP(EXP)     rI1 ← −e_v.
14        JMP  4F
15  1H    STA  FU(0:4)       FU ← ±fffff0 (u, v interchanged).
16        LD2  TEMP(EXP)     rI2 ← e_w.
17        STX  FV(0:4)
18        LD1N ACC(EXP)      rI1 ← −e_v.
```

Thus, a fixed-point coding sequence

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

would correspond to the floating-point coding sequence

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

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 which appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine which prints out a message to the effect that MIX's overflow toggle was unexpectedly found to be "on."

```
01   EXP   EQU   1:1          Definition of exponent field.
02   FSUB  STA   TEMP         Floating-point subtraction subroutine:
03         LDAN  TEMP         Change sign of operand.
04   FADD  STJ   EXITF        Floating-point addition subroutine:
05         JOV   OFLO         Ensure overflow is off.
06         STA   TEMP         TEMP ← v.
07         LDX   ACC          rX ← u.
08         CMPA  ACC(EXP)     Steps A1, A2, A3 are combined here:
09         JLE   1F           Jump if e_v ≤ e_u.
10         STA   FU(0:4)      FU ← ± f f f f 0 (u, v interchanged).
11         LD2   TEMP(EXP)    rI2 ← e_w.
12         STX   FV(0:4)
13         LD1N  ACC(EXP)     rI1 ← −e_v.
14         JMP   4F
15   1H    STX   FU(0:4)      FU ← ± f f f f 0.
16         LD2   ACC(EXP)     rI2 ← e_w.
17         JE    3F           Jump if e_u = e_v (saves time).
18         STA   FV(0:4)
19         LD1N  TEMP(EXP)    rI1 ← −e_v.
20   4H    INC1  0,2          rI1 ← e_u − e_v. (Step A4 unnecessary.)
21   5H    LDA   FV           A5. Scale right.
22         LDX   FV(0:0)      Set rX to zero with sign of v.
23         SRAX  0,1          Shift right e_u − e_v places.
24         JMP   6F
25   3H    SLA   1            Align radix point.
26         ENTX  0            Clear rX.
27   6H    ADD   FU           A6. Add.
28         JOV   N4           A7. Normalize. Jump if fraction overflow.
29         CMPA  =0=(1:1)
30         JNE   N5           Jump if f is normalized.
```

A specially trained typist would key in most of the formula by making two passes: First the letters and symbols on the main line would be entered, and their superscripts (namely the characters

$$|\det(a\ \ )| \leq \qquad\qquad a^2$$

in this case); then a second pass was made for the subscripts (namely the characters '$_{ij}$', repeated twice here). The keyboard operator had to know the width of each character so that there would be 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 $\prod$ and $\sum$) by hand.

Only a few dozen people in the world knew how to typeset mathematical formulas with Monotype. I once had the pleasure and privilege 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.

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 which appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine which prints out a message to the effect that MIX's overflow toggle was unexpectedly found to be "on."

```
01   EXP   EQU   1:1        Definition of exponent field.
02   FSUB  STA   TEMP       Floating-point subtraction subroutine:
03         LDAN  TEMP       Change sign of operand.
```

Slide 9.

Books on computer science have added a new complication to the difficulties that printers already faced in mathematical typesetting: Computer scientists need to use a special style of type called **typewriter type**, in order to represent the textual material that machines deal with. For example [SLIDE 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.

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 which appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine which 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 $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

```
01 EXP    EQU   1:1          Definition of exponent field.
02 FSUB   STA   TEMP         Floating-point subtraction subroutine.
03        LDAN  TEMP         Change sign of operand.
```

Slide 10.

But the results were very disappointing. For example [SLIDE 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 "NORM" 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,

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 $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

```
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.
```

Slide 12.

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 [SLIDE 13] didn't look much better than the version I had rejected before starting my typography project.

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 others 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 $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

```
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.
```

Slide 13.

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 rarely 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 [SLIDE 14], 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 [SLIDE 15]: 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 [SLIDE 16]. I swear that I was *not* thinking of yen when I did this!

Thus, a fixed-point coding sequence

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

would correspond to the floating-point coding sequence

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

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 which appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine which prints out a message to the effect that MIX's overflow toggle was unexpectedly found to be "on."

```
01   EXP   EQU   1:1          Definition of exponent field.
02   FSUB  STA   TEMP         Floating-point subtraction subroutine:
03         LDAN  TEMP         Change sign of operand.
04   FADD  STJ   EXITF        Floating-point addition subroutine:
05         JOV   OFLO         Ensure overflow is off.
06         STA   TEMP         TEMP ← v.
07         LDX   ACC          rX ← u.
08         CMPA  ACC(EXP)     Steps A1, A2, A3 are combined here:
09         JLE   1F           Jump if e_v ≤ e_u.
10         STA   FU(0:4)      FU ← ± f f f f 0 (u, v interchanged).
11         LD2   TEMP(EXP)    rI2 ← e_w.
12         STX   FV(0:4)
13         LD1N  ACC(EXP)     rI1 ← −e_v.
14         JMP   4F
15   1H    STX   FU(0:4)      FU ← ± f f f f 0.
16         LD2   ACC(EXP)     rI2 ← e_w.
17         JE    3F           Jump if e_u = e_v (saves time).
18         STA   FV(0:4)
19         LD1N  TEMP(EXP)    rI1 ← −e_v.
20   4H    INC1  0,2          rI1 ← e_u − e_v. (Step A4 unnecessary.)
21   5H    LDA   FV           A5. Scale right.
22         LDX   FV(0:0)      Set rX to zero with sign of v.
23         SRAX  0,1          Shift right e_u − e_v places.
24         JMP   6F
25   3H    SLA   1            Align radix point.
26         ENTX  0            Clear rX.
27   6H    ADD   FU           A6. Add.
28         JOV   N4           A7. Normalize. Jump if fraction overflow.
29         CMPA  =0=(1:1)
30         JNE   N5           Jump if f is normalized.
```

be expressed as a normalized floating point number in the required range, special action is necessary.)

N7. [Pack.] Put e and f together into the desired output representation. ∎

Some simple examples of floating point addition are given in exercise 4.

The following MIX subroutines, for addition and subtraction of numbers having the form (4), show how Algorithms A and N can be expressed as computer programs. The subroutines below are designed to take one input u from symbolic location ACC, and the other input v comes from register A upon entrance to the subroutine. The output w appears both in register A and location ACC. Thus, a fixed point coding sequence

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

would correspond to the floating point coding sequence

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

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 others 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.
04  FADD  STJ  EXITF         Floating point addition subroutine:
05        JOV  OFLO          Ensure overflow is off.
06        STA  TEMP          TEMP ← v.
07        LDX  ACC           rX ← u.
08        CMPA ACC(EXP)      Steps A1, A2, A3 are combined here:
09        JGE  1F            Jump if e_v ≥ e_u.
10        STX  FU(0:4)       FU ← ±fffff0.
11        LD2  ACC(EXP)      rI2 ← e_w.
12        STA  FV(0:4)
13        LD1N TEMP(EXP)     rI1 ← −e_v.
14        JMP  4F
15  1H    STA  FU(0:4)       FU ← ±fffff0 (u, v interchanged).
16        LD2  TEMP(EXP)     rI2 ← e_w.
17        STX  FV(0:4)
18        LD1N ACC(EXP)      rI1 ← −e_v.
```

A specially trained typist would key in most of the formula by making two passes: First the letters and symbols on the main line would be entered, and their superscripts (namely the characters

$$|\det(a\ \ )| \leq \qquad\qquad a^2$$

in this case); then a second pass was made for the subscripts (namely the characters '$_{ij}$', repeated twice here). The keyboard operator had to know the width of each character so that there would be 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 $\prod$ and $\sum$) by hand.

Only a few dozen people in the world knew how to typeset mathematical formulas with Monotype. I once had the pleasure and privilege 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.

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 which appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine which prints out a message to the effect that MIX's overflow toggle was unexpectedly found to be "on."

```
01   EXP   EQU   1:1          Definition of exponent field.
02   FSUB  STA   TEMP         Floating-point subtraction subroutine:
03         LDAN  TEMP         Change sign of operand.
```

Slide 9.

Books on computer science have added a new complication to the difficulties that printers already faced in mathematical typesetting: Computer scientists need to use a special style of type called `typewriter` type, in order to represent the textual material that machines deal with. For example [SLIDE 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.

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 which appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine which 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 $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

```
01  EXP    EQU   1:1          Definition of exponent field.
02  FSUB   STA   TEMP         Floating-point subtraction subroutine.
03         LDAN  TEMP         Change sign of operand.
```

Slide 10.

But the results were very disappointing. For example [SLIDE 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 "NORM" 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,

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 $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

```
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.
```

Slide 12.

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 [SLIDE 13] didn't look much better than the version I had rejected before starting my typography project.

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 others 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 $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

```
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.
```

Slide 13.

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 rarely 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 "METAFLOPS." For example [SLIDE 14], 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 [SLIDE 15]: 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 [SLIDE 16]. I swear that I was *not* thinking of yen when I did this!