☆ THE ART OF COMPUTER PROGRAMMING ☆

☆☆☆☆ ERRATA TO VOLUME 1 FASCICLE 1 ☆☆☆☆

This document is a transcript of the notes that I have been making in my personal copy of *The Art of Computer Programming*, Volume 1, Fascicle 1, since it was first printed in 2005.

Four levels of updates — "errors," "amendments," "plans," and "improvements" — appear, indicated by four different typographic conventions:

▸**Page 666** line 1 —————————————————— 04 Jul 1776

Technical or typographical errors (aka bugs) are the most critical items, so they are flagged with a '▸' preceding the page number. The date on which I first was told about the bug is shown; this is the effective date on which I paid the finder's fee. The necessary corrections are indicated in a straightforward way. If, for example, the book says 'n' where it should have said '$n+1$', the change is shown thus:

$n \leadsto n+1$

Page 666 line 2 —————————————————— 14 Jul 1789

Amendments to the text appear in the same format as bugs, but without the '▸'. These are things I wish I had known about or thought of when I wrote the original text, so I added them later. The date is the date I drafted the new text.

Page 666 line 3 . . . . 20 Nov 1917

Plans for the future represent a third kind of item. In such notes I sketched my intentions about things that I wasn't ready to flesh out further when I wrote them down. You can identify these items because they're written in slanted type, and preceded by a bunch of dots '. . . . .' leading to the date on which I recorded the plan in my files.

Page 666 line 4 —————————————————— 10 Jan 1938

The fourth and final category — indicated by page and line number in smaller, slanted type — consists of minor corrections or improvements that most readers don't want to know about, because they are so trivial. You wouldn't even be seeing these items if you hadn't specifically chosen to print the complete errata list in all its gory details. Are you sure you wanted to do that?

The ultimate, glorious, future editions of Volumes 1–5 are works in progress. Please let me know of any improvements that you think I ought to make. Send your comments either by snail mail to D. E. Knuth, Computer Science, Gates Building 1B, Stanford University, Stanford CA 94305-9015, or by email to `taocp@cs.stanford.edu`*. (Use email for book suggestions only, please — all other correspondence is returned unread to the sender, or discarded, because I have no time to read ordinary email.) Although I'm working full time on*

Volume 4 these days, I will try to reply to all such messages within a year of receipt. Current news about The Art of Computer Programming is posted on

`http://www-cs-faculty.stanford.edu/~knuth/taocp.html`

and updated regularly.

— Don Knuth, February 2005

For a work of such scope as this,
the first word of the author should be an apology
for what is doubtless the too ambitious effort
of a single writer.

— EDWARD W. BYRN (1900)

☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆ **MMIX** ☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆

Last updated 8 January 2025

Page ii new line to follow line 6 —— 01 Jan 2006

Unicode is a trademark of Unicode, Inc.

Page ii line −15 —— 07 Aug 2013

`http://mmixmasters.sourceforge.net` ⇝ `http://mmix.cs.hm.edu/`

Page iii line −11 —— 09 Oct 2015

gratefully pay a reward of \$2.56 ⇝ gratefully award one hexadecimal dollar (#`$1.00` = \$2.56)

▸**Page iv** lines 7 and 8 —— 20 Jun 2024

MixMaster ⇝ Mixmaster

Page v line 20 —— 07 Aug 2013

`mmixmasters.sourceforge.net` ⇝ `mmix.cs.hm.edu/mmixmasters/`.

Page v new material to insert before the quotation —— 29 Jun 2020

News flash: I'm pleased to report that the first phase of the `MMIX`masters project was successfully completed in September 2012, under the inspired leadership of Martin Ruckert. Every instance of `MIX` code in Volumes 1, 2, and 3 now has a suitable `MMIX` equivalent. Martin's book, *The `MMIX` Supplement* (Addison–Wesley, 2015), makes it easy for readers to substitute this 21st-century material wherever needed. Volunteers are still, however, encouraged to submit improvements, so that the ultimate programs will have the highest possible quality. (See `http://mmix.cs.hm.edu/supplement/index.html` for further information.)

▸**Page 1** and all the following pages in the 2017 printing —— 11 Nov 2017

[Page 1 was mistakenly printed as a *left-hand* page, and all subsequent pages were fouled up too — odd numbered pages at the left of even-numbered pages!]

Page 2 line 24 —— 14 Dec 2006

100% natural ⇝ 100% natural, organic

▸**Page 3** top line of Heller quote —— 27 Apr 2024

256th squadron ⇝ 256th Squadron

▸**Page 3** lines −11, −10, and −9 —— 22 Feb 2007

became an international . . . includes ⇝ has also been developed. This code, known informally as Unicode, includes

Page 3 line −1 ———————— 22 Feb 2007

à l'Unicode. $\leadsto$ *à l'Unicode.* [The mature Unicode Standard has in fact outgrown its 16-bit origins; it now contains many more than 2^{16} code points, which can be represented in 8-bit, 16-bit, or 32-bit encodings, as discussed in Section 7.1.3. But its 16-bit "Basic Multilingual Plane" does include most of the common characters.]

Page 4 line 2 ———————— 22 Feb 2007

Unicode $\leadsto$ Unicode's basic plane

Page 6 line 13 ———————— 07 May 2009

if N is too large or too small to fit $\leadsto$ if the representation of N needs too many bits to fit

Page 7 line −7 ———————— 09 Nov 2018

Overflow is possible if $\leadsto$ Overflow occurs if

Page 9 line 5 ———————— 23 Sep 2007

the divisor, $\leadsto$ the divisor \$Z

Page 10 lines 6, 10, 17, 21 ———————— 01 Jun 2006

$\mathrm{s}(\$\mathrm{Y}) \bmod 2 \leadsto \$\mathrm{Y} \bmod 2$

Page 10 lines −17 through −4 ———————— 19 May 2005

change all occurrences of '$\wedge$' to '&' and all occurrences of '$\vee$' to '$|$' (16 changes altogether)

Page 11 near the top ———————— 11 Sep 2006

line 1: $\mathrm{v}(\$\mathrm{X}) \leftarrow \bigl(\mathrm{v}(\$\mathrm{Y}) \wedge \mathrm{v}(\mathrm{rM})\bigr) \vee \bigl(\mathrm{v}(\$\mathrm{Z}) \wedge \bar{\mathrm{v}}(\mathrm{rM})\bigr) \leadsto$
$\mathrm{v}(\$\mathrm{X}) \leftarrow \bigl(\mathrm{v}(\$\mathrm{Y}) \mathbin{\&} \mathrm{v}(\mathrm{rM})\bigr) \mid \bigl(\mathrm{v}(\$\mathrm{Z}) \mathbin{\&} \bar{\mathrm{v}}(\mathrm{rM})\bigr)$
line 4: $\sum\bigl(\mathrm{v}(\$\mathrm{Y}) \wedge \bar{\mathrm{v}}(\$\mathrm{Z})\bigr) \leadsto \nu\bigl(\mathrm{v}(\$\mathrm{Y}) \mathbin{\&} \bar{\mathrm{v}}(\$\mathrm{Z})\bigr)$

Page 11 line 15 ———————— 09 Jul 2006

denotes the operation of *saturating subtraction*, $\leadsto$ denotes *saturating subtraction*, also known as "dot minus" or "monus":

Page 12 near the top ———————— 19 May 2005

lines 1 and 2: $\vee \leadsto |$ (four changes)
lines 6, 7, and 9: $\overset{\vee}{\times} \leadsto \overset{|}{\times}$ (three changes)

Page 12 line 20 ———————— 04 Mar 2008

Exercises 33–37 $\leadsto$ Exercises 31–37

Page 12 line −11 ———————— 16 Jul 2006

(denormal) $\leadsto$ (subnormal)

Page 14 line 2 in the first printing ———————— 17 Sep 2005

by the number Z $\leadsto$ *the nonnegative number* Z

Page 14 line 5 in the first printing — 17 Sep 2005

'`SRU` $\rightsquigarrow$ '`NEG $X,$Z`' has a counterpart '`NEG $X,Z`' meaning s($X) $\leftarrow -$Z; '`SRU`

Page 14 lines -18 through -7 — 19 May 2005

change all occurrences of '$\land$' to '&' and all occurrences of '$\lor$' to '|' (eight changes altogether)

Page 15 lines -14, -10, -2 — 15 Jul 2006

s($X) mod 2 $\rightsquigarrow$ $X mod 2

Page 16 line 3 — 15 Jul 2006

s($X) mod 2 $\rightsquigarrow$ $X mod 2

Page 19 line 5 — 19 May 2005

$\mathrm{rK} \land \mathrm{rQ} \neq 0$ $\rightsquigarrow$ $\mathrm{rK} \,\&\, \mathrm{rQ} \neq 0$

Page 19 lines -4 and -3 — 03 Jun 2011

a "clock register" …advancing; $\rightsquigarrow$ a *continuation register*, rC, used by the operating system for stack overflow; a *failure register*, rF, to help diagnose hardware errors;

▸**Page 20** line -10 — 11 Nov 2005

increases by 1 $\rightsquigarrow$ increases

Page 21 line 6 — 03 Jun 2011

cycle counter $\rightsquigarrow$ continuation register

Page 23 lines -12 and -11 — 24 Jan 2020

Commercial machines do not (yet?) have the powerful `MOR` and `MXOR` operations. They usually have a half dozen or so $\rightsquigarrow$ Commercial machines rarely have the powerful `MOR` and `MXOR` operations (although the situation is slowly improving). They sometimes have a few

Page 25 line -2 — 19 May 2005

$\land$ and $\lor$ $\rightsquigarrow$ & and |

▸**Page 29** line 7 — 17 May 2005

`#55 00 ff fa` $\rightsquigarrow$ `#55 02 ff fa`

Page 29 insert new paragraph as the bottom two lines — 09 Feb 2020

[Why '`LOC #100`'? It's generally best to start at location `#100` because the smaller locations are used for trip handlers and other special purposes.]

Page 30 line 20 — 23 Apr 2007

set to 100. $\rightsquigarrow$ set to 100. Using principles explained below in Section 1.4.1′,

▸**Page 32** lines 10–14 — 29 Mar 2023

[these lines should all be indented so that '`0002`' lies under '`t Fi`']

Page 32 step P1 ——— 21 May 2005

In this program ... keeps track $\leadsto$ In the following steps, n will run through the odd numbers that are candidates for primes; j will keep track

Page 34 line 7 ——— 21 May 2005

Initialize m. $\leadsto$ Initialize m by setting `mm` $\leftarrow -2$.

Page 37 line −17 ——— 22 Jan 2010

hash mark `#` $\leadsto$ number sign or hash mark `#`

▸**Page 39** lines 12–15 from the bottom ——— 21 Sep 2006

equivalent to the largest ... not change that global register. $\leadsto$ equivalent to the number of a global register that will contain x when the program begins. If $x \neq 0$, the value of x is considered to be a *base address*, and the program should not change that global register. If $x = 0$, or if x is a base address that hasn't already occurred, a new global register is allocated (as large as possible).

Page 40 line 11 ——— 29 Jul 2006

`TRAP 0,Halt,0` $\leadsto$ `TRAP 0,Halt,0;`

Page 42 line 5 ——— 22 Feb 2007

an error $\leadsto$ a serious error

▸**Page 44** line 2 of exercise 12 ——— 05 Jan 2009

`Blank` $\leadsto$ `Blanks`

▸**Page 44** replacement for line 2 of exercise 13 ——— 10 Jan 2010

أول خمسمائة عدر أولي

Page 48 line 9 of exercise 30 ——— 15 Mar 2024

10^n. Then we wish $\leadsto$ 10^n, although that formula doesn't work for arbitrary x. Then we wish

Page 50 line 7 ——— 21 Mar 2011

two-bit codes $\leadsto$ individual codes

Page 50 lines 8 and 9 become like RGB codes ——— 03 Jun 2011

`#80` amber $\leadsto$ `#c0` amber
`#c0` red $\leadsto$ `#80` red
`#20` amber $\leadsto$ `#30` amber
`#30` red $\leadsto$ `#20` red

Page 50 next-to-last line of exercise 34 ——— 04 Jun 2011

special clock register rC increases $\leadsto$ interval counter rI decreases

Page 51 line −3 ——— 01 Sep 2013

in Chapters $\leadsto$ *in Section 1.4.4 and in Chapters*

Page 52 line 14 ——— 17 Sep 2005

by having less space $\leadsto$ by occupying less space

Page 53 line 8 ——— 17 Sep 2005

These `MMIX` instructions $\leadsto$ We'll see shortly that these `MMIX` instructions

▸**Page 54** line -8 ——— 29 Mar 2023

`$,n` $\leadsto$ `$,`n

▸**Page 60** line 6 ——— 17 Dec 2018

a `PUSHJ` subroutine $\leadsto$ a `PUSHJ` instruction

▸**Page 60** line -3 of (16) ——— 08 Jun 2005

`BNZ  kk,2F` $\leadsto$ `BN  kk,2F`

Page 66 line 2 of exercise 13 ——— 29 Mar 2023

given n $\leadsto$ given $n \ge 0$

Page 65 the rating of exercise 9 ——— 10 Jan 2007

20 $\leadsto$ *28*

Page 67 line -13 ——— 24 Oct 2005

the ASCII code for `' '` $\leadsto$ which is the character constant `' '` in `MMIXAL` programs

Page 74 line 22 ——— 29 Mar 2023

type-(b) interpreter $\leadsto$ type (b) interpreter

Page 74 line 26 ——— 06 Apr 2010

multiple precision arithmetic $\leadsto$ multiple-precision arithmetic

Page 75 line 14 ——— 08 Nov 2005

simulator (or sometimes an *emulator*) $\leadsto$ *simulator* (or sometimes an *emulator*)

Page 76 line -17 ——— 03 Jun 2011

Table 1.3.1′–2; one of those special registers will be the simulated clock, rC. $\leadsto$ Table 1.3.1′–2.

Page 76 lines -15 and -14 ——— 03 Jun 2011

simulated rC $\leadsto$ simulated clock (twice)

Page 80 line *062* of the program ——— 03 Jun 2011

cycle counter, rC $\leadsto$ clock

▸**Page 79** line -9 ——— 15 Feb 2019

But eight $\leadsto$ But seven

▸**Page 80** line *068* of the program ——— 19 Apr 2009

`t` $\leftarrow$ [$\$k$ is local]. $\leadsto$ Set `t` negative if $\$k$ is local.

Page 82 line 6 ——— 19 May 2005

`f` $\wedge$ `#40` $\leadsto$ `f` & `#40`

Page 82 lines *142–144* of the program ——— 19 May 2005

$\wedge$ `#ff` $\leadsto$ & `#ff`

▸**Page 83** entire page ______ 20 Jun 2017

[This entire page should be shifted to the right, so that the page number aligns with the others.]

Page 83 line *158* of the program ______ 19 May 2005

`inst` $\land$ `#ffffff` $\rightsquigarrow$ `inst` & `#ffffff`

▸**Page 84** lines *227–228* of the program ______ 19 Apr 2009

[Remove the comment on line *227*; say 'Branch if `op` is a load instruction.' on line *228*.]

Page 85 lines *254–255* of the program ______ 03 Jun 2011

rC $\rightsquigarrow$ the clock (twice)

Page 88 line 2 from the bottom ______ 15 Jan 2008

easily deduced $\rightsquigarrow$ easy to infer

Page 89 line *505* of the program ______ 19 May 2005

`exc` $\leftarrow$ `exc` $\lor$ `t` $\rightsquigarrow$ `exc` $\leftarrow$ `exc` | `t`

Page 89 line −6 ______ 16 Jul 2006

denormal $\rightsquigarrow$ subnormal

Page 89 line −4 ______ 29 Jun 2020

clocks $\rightsquigarrow$ clocks and counters

Page 90 line 1 ______ 03 Jun 2011

clock, rC. $\rightsquigarrow$ clock.

▸**Page 90** line 5 ______ 29 Mar 2023

Otherwise set `t` $\leftarrow$ [`op` = `RESUME`]. $\rightsquigarrow$ Otherwise set |`t`| $\leftarrow$ [`op` = `RESUME`].

Page 90 lines *580–584* of the program ______ 04 Jun 2011

`90` $\rightsquigarrow$ `f0` (four times)

Page 91 line 4 ______ 04 Jun 2011

`#90` $\rightsquigarrow$ `#f0`

Page 92 last line of exercise 12 ______ 29 Jun 2020

clocks $\rightsquigarrow$ clocks and counters

Page 94 line 3 of answer 4 ______ 29 Mar 2023

(YB), according to the 19th ... (1990). $\rightsquigarrow$ (YB), 1000 YB = 1 ronnabyte (RB), 1000 RB = 1 quettabyte (QB), according to the 27th Conférence Générale des Poids et Mesures (2022).

Page 94 answer 4: *not* an error! ______ 17 Mar 2006

[A widely disregarded "international standard" has also been adopted for binary-based units. But I think it is so preposterous, I've chosen not to even mention it. See `http://www-cs-faculty.stanford.edu/~knuth/news99.html` for further comments.]

Page 94 answer 4, end of second paragraph ——— 29 Mar 2023

binary nature. $\leadsto$ binary nature while remaining friendly to the English language.)

Page 94 new sentence at end of answer 9 ——— 02 Dec 2011

occurs. $\leadsto$ occurs. (In this answer and several others below, 'X = Y' means that the registers have the same *name*, while \$X = \$Y means that they have the same *contents*.)

Page 95 first two lines of answer 15 ——— 02 Jul 2016

$\mathrm{s}(y) = y - 2^{64}\dot{y}\ \ldots\ \mathrm{mod} 2^{128}$. $\leadsto$ $\mathrm{s}(y) = \mathrm{u}(y) - 2^{64}\dot{y}$ and $\mathrm{s}(z) = \mathrm{u}(z) - 2^{64}\dot{z}$; we want to calculate $\mathrm{s}(y)\mathrm{s}(z) \bmod 2^{128} = \bigl(\mathrm{u}(y)\mathrm{u}(z) - 2^{64}(\dot{y}\mathrm{u}(z) + \mathrm{u}(y)\dot{z})\bigr) \bmod 2^{128}$.

▸**Page 95** line 5 of answer 19 ——— 26 Oct 2006

$0 < \epsilon < 2/z \leadsto 0 \le \epsilon < 2/z$

Page 95 last line of answer 19 ——— 02 May 2021

in general.) $\leadsto$ in general. For a comprehensive discussion see T. Granlund and P. L. Montgomery, *SIGPLAN Notices* **29**, 6 (June 1994), 61–72.)

Page 95 line 1 of answer 29 ——— 29 Mar 2023

`NOR $0,$Y,0; BDIF $0,$0,$Z; NOR $X,$0,0.` $\leadsto$
`NOR $X,$Y,0; BDIF $X,$X,$Z; NOR $X,$X,0.`

Page 96 line 4 of answer 37 ——— 29 May 2006

respectively. $\leadsto$ respectively. This construction works because $x^8 + x^6 + x^5 + x^4 + 1$ is a primitive polynomial modulo 2 (see Section 3.2.2).

▸**Page 97** answer 42 ——— 08 Dec 2006

`NEGU` $\leadsto$ `NEG`

Page 97 line -2 ——— 10 Jan 2009

X is marginal $\leadsto$ \$X is marginal

Page 99 improvement to answer 8 ——— 13 Oct 2007

8. `LOC -@/16*-16` or `LOC (@+15)&-16` or `LOC -(-@>>4<<4)`, etc.

▸**Page 100** line 1 of answer 12 ——— 10 Jan 2009

`Blank` $\leadsto$ `Blanks`

▸**Page 100** replacement for lines 6–9 of answer 13 ——— 10 Jan 2010

`Title WYDE "أول خمسمائة عدد أولي"`

when it is printed bidirectionally, but in the computer file the individual characters actually appear in "logical" order without ligatures. Thus a spelled-out sequence like

`Title WYDE 'أ','و','ل',' ','خ','م','س','م',...,'و','ل','ي'`

▸**Page 104** line 7 ——— 26 May 2012

`PBP ii,3B` $\leadsto$ `PBNN ii,3B`

Page 105 line 2 ——— 23 Jun 2020

(See the website information on page ii.) $\leadsto$ *(See page v.)*

▶**Page 105** line −3 of answer 22 ——— 29 Mar 2023

Philomathique ⇝ *philomatique*

▶**Page 110** last line of answer 30 ——— 29 Mar 2023

$R_n(m_{k+1}) = R_n(m_k - 1) \rightsquigarrow R_n(1/m_{k+1}) = R_n(1/m_k) - 1$

Page 111 line 10 ——— 27 Aug 2014

Victorius of Aquitania ⇝ Victorius of Aquitaine

Page 111 replacement for answer 34 (continues to page 113) ——— 04 Jun 2011

34. The following program follows the protocol to within a few hundred υ; this slight discrepancy is negligible, since ρ is typically more than 10^8, and $\rho\upsilon = 1\,\mathrm{sec}$. All of the computation takes place in registers, except when a byte is input.

An interrupt occurs when rI reaches zero. We assume that the operating system will return soon after, resetting rI to some not-too-small value.

```
* Traffic Signal Problem
rho         GREG   250000000              Assume 250 MHz clock rate
t           IS     $255
Sensor_Buf  IS     Data_Segment
            GREG   Sensor_Buf

            LOC    #100
            GREG   @
delta       GREG                          Number of cycles to delay
ref_time    GREG                          Internal register for Delay
delay_go    GREG                          Exit location for Delay

2H          SUBU   delta,delta,t          Decrease δ by rI.
1H          SET    ref_time,t             Remember previous value of rI.
            GET    t,rI
            CMPU   ref_time,ref_time,t
            PBP    ref_time,1B            Repeat until rI increases.
Delay       GET    t,rI                   Wait busily for δ cycles:
            SUBU   ref_time,t,delta       Compute rI − δ.
            BN     t,1F                   Branch if rI ≥ 2^63.
            BN     ref_time,2B            Branch if rI is less than δ.
1H          GET    t,rI
            SUBU   t,t,ref_time           (N.B. Not CMPU; see below)
            PBP    t,1B                   Repeat until rI passes ref_time.
            GO     delay_go,delay_go,0    Return to caller.

Lights      IS     3                      Handle for /dev/lights
Sensor      IS     4                      Handle for /dev/sensor
Lights_Name BYTE   "/dev/lights",0
Sensor_Name BYTE   "/dev/sensor",0
Lights_Args OCTA   Lights_Name,BinaryWrite
Sensor_Args OCTA   Sensor_Name,BinaryRead
Read_Sensor OCTA   Sensor_Buf,1

Dg IS #40 ;Da IS #c0 ;Dr IS #80 ;Dw IS #4 ;Dd IS #c
Bg IS #10 ;Ba IS #30 ;Br IS #20 ;Bw IS #1 ;Bd IS #3
Boulevard   BYTE   Dg|Dw|Br|Bd,0,Dg|Dd|Br|Bd,0,Dg|Br|Bd,0,Da|Dd|Br|Bd,0
Avenue      BYTE   Bg|Bw|Dr|Dd,0,Bg|Bd|Dr|Dd,0,Bg|Dr|Dd,0,Ba|Bd|Dr|Dd,0
```

```
flash_go   GREG
n          GREG   0                      Iteration counter
green      GREG   0                      Boulevard or Avenue
Flash      SET    n,8                    Subroutine to flash the lights:
1H         ADDU   t,green,2*1
           TRAP   0,Fputs,Lights         DON'T WALK
           SR     delta,rho,1
           GO     delay_go,Delay
           ADDU   t,green,2*2
           TRAP   0,Fputs,Lights         (off)
           SR     delta,rho,1
           GO     delay_go,Delay
           SUB    n,n,1
           PBP    n,1B                   Repeat eight times.
           ADDU   t,green,2*1
           TRAP   0,Fputs,Lights         DON'T WALK
           MUL    delta,rho,4
           GO     delay_go,Delay         Hold for 4 sec.
           ADDU   t,green,2*3
           TRAP   0,Fputs,Lights         DON'T WALK, amber
           GO     flash_go,flash_go,0    Return to caller.
Main       GETA   t,Lights_Args          Open the files: Fopen(Lights,
           TRAP   0,Fopen,Lights           "/dev/lights",BinaryWrite)
           GETA   t,Sensor_Args          Fopen(Sensor,
           TRAP   0,Fopen,Sensor           "/dev/sensor",BinaryRead)
           JMP    2F                     Begin with green on the boulevard.
Wait       GETA   t,Read_Sensor          Extend the 18 sec green.
           TRAP   0,Fread,Sensor
           LDB    t,Sensor_Buf
           BZ     t,Wait                 Repeat until sensor is nonzero.
           GETA   green,Boulevard
           GO     flash_go,Flash         Finish the boulevard cycle.
           MUL    delta,rho,8
           GO     delay_go,Delay         Amber for 8 sec.
           GETA   t,Avenue
           TRAP   0,Fputs,Lights         Green light for Berkeley.
           MUL    delta,rho,8
           GO     delay_go,Delay
           GETA   green,Avenue
           GO     flash_go,Flash         Finish the avenue cycle.
           GETA   t,Read_Sensor
           TRAP   0,Fread,Sensor         Ignore sensor during green time.
           MUL    delta,rho,5
           GO     delay_go,Delay         Amber for 5 sec.
2H         GETA   t,Boulevard
           TRAP   0,Fputs,Lights         Green light for Del Mar.
           MUL    delay,rho,18
           GO     delay_go,Delay         At least 18 sec to WALK.
           JMP    Wait                   ▮
```

The final `SUBU` instruction in the `Delay` subroutine is an interesting example of a case where the comparison should be done with `SUBU`, *not* with `CMPU`, in spite of the comments in answer 1.3.1′–22. The reason is that the two quantities being compared, rI and `ref_time`, "wrap around" modulo 2^{64}.

▶**Page 113** a correction to 3rd-last line of that program ——— 15 Mar 2024

`MUL delay,rho,18` $\rightsquigarrow$ `MUL delta,rho,18`

▶**Page 114** lines 2 and 3 of answer 9 ——— 07 Jan 2007

two additional instructions "`GET $255,rB; RESUME`" will $\rightsquigarrow$
"`PUT rJ,$255; GET $255,rB; RESUME`" after the `PUSHJ` will

Page 115 last line of answer 16 ——— 11 Jan 2009

of (15) $\rightsquigarrow$ of exercise 15

▶**Page 116** answer 4 ——— 29 Mar 2023

[the code should be properly indented, with `GREG`s lined up]

▶**Page 118** line 1 ——— 14 Dec 2018

line 309 of the text $\rightsquigarrow$ line 309 of the code

Page 118 line 7 of answer 8 ——— 03 Jun 2011

[delete this line, which begins '`STOU cc`']

▶**Page 119** line −5 of answer 9 ——— 15 Mar 2024

Branch if $G <$ `z`. $\rightsquigarrow$ Branch if $G >$ `z`.

Page 125 line 2 of answer 19 ——— 19 May 2005

$\texttt{l} + ((\texttt{oo} + \texttt{ll}) \wedge \texttt{lring\_mask})$ $\rightsquigarrow$ $\texttt{l} + ((\texttt{oo} + \texttt{ll}) \,\&\, \texttt{lring\_mask})$

Page 127 and following ——— 25 Feb 2005

Miscellaneous changes to the existing index of Volume 1 Fascicle 1 are collected here, including corrections and amendments to the old entries as well as new entries that are occasioned by the new material. Thus, the lines of the full index that have changed serve also as an index to the present document. However, when a correction or amendment has caused an old index entry to be deleted, the deletion is usually not indicated.