```
!ASM
**END OF PASS 1

**END OF PASS 2

0800                1  *
0800                2  * PI_SPIGOT.S
0800                3  * Compute digits of PI on Apple II
0800                4  * Algorithm: Rabinowitz-Wagon Decimal Spigot
0800                5  * Assembler: LISA 2.5D / DOS 3.3
0800                6  * Target: Apple II 128K (only ~22KB RAM needed)
0800                7  *
0800                8  * ------------------------------------------------
0800                9  * MEMORY MAP
0800               10  * ------------------------------------------------
0800               11  *   $0000-$001B  Zero page variables
0800               12  *   $0900        Program origin
0800               13  *   $3FFE-$3FFF  Cell 0 (special leading cell)
0800               14  *   $4000-$A82B  Working array, 13334 cells x 2 bytes
0800               15  *
0800               16  * ------------------------------------------------
0800               17  * ALGORITHM SUMMARY
0800               18  * ------------------------------------------------
0800               19  * Array A[0..LEN] initialized to 2 in every cell.
0800               20  * Each pass (right to left) produces one predigit:
0800               21  *
0800               22  *   carry = 0
0800               23  *   for i = LEN downto 1:
0800               24  *     x      = A[i] * 10 + carry
0800               25  *     A[i]   = x MOD (2i+1)
0800               26  *     carry  = (x DIV (2i+1)) * i
0800               27  *   x        = A[0] * 10 + carry
0800               28  *   A[0]     = x MOD 10
0800               29  *   predigit = x DIV 2
0800               30  *
0800               31  * Predigits buffered to handle 9->10 carry.
0800               32  *
0800               33  * ------------------------------------------------
0800               34  * CARRY BOUND - why 24-bit math is sufficient
0800               35  * ------------------------------------------------
0800               36  * Invariant: A[i] < 2i+1 always holds.
0800               37  * Max x at cell i is approximately 30*i.
0800               38  * At i=13334, x < 400000, fits in 19 bits.
0800               39  * 24-bit intermediates provide ample headroom.
0800               40  *
0800               41  * ------------------------------------------------
0800               42  * ZERO PAGE EQUATES
0800               43  * ------------------------------------------------
0000               44  ARLO     EQU $00
0001               45  ARHI     EQU $01
0002               46  DNLO     EQU $02
0003               47  DNHI     EQU $03
0004               48  IXLO     EQU $04
0005               49  IXHI     EQU $05
0006               50  XL       EQU $06
0007               51  XM       EQU $07
0008               52  XH       EQU $08
0009               53  CL       EQU $09
000A               54  CM       EQU $0A
000B               55  CH       EQU $0B
000C               56  QL       EQU $0C
000D               57  QM       EQU $0D
000E               58  QH       EQU $0E
000F               59  PRDIG    EQU $0F
```

```
0010              60  NINES    EQU $10
0011              61  DCLO     EQU $11
```

```
0012             62  DCHI     EQU $12
0013             63  COL      EQU $13
0014             64  TMP      EQU $14
0015             65  DVLO     EQU $15
0016             66  DVHI     EQU $16
0017             67  RELO     EQU $17
0018             68  REHI     EQU $18
0019             69  SAVDIG   EQU $19
001A             70  MLLO     EQU $1A
001B             71  MLHI     EQU $1B
001C             72  GOTSAV   EQU $1C
0800             73  *
0800             74  * ------------------------------------------------
0800             75  * PRECOMPUTED SPLIT CONSTANTS
0800             76  * LISA 2.5D has no < > byte extraction operators.
0800             77  * All 16-bit values split manually here.
0800             78  *
0800             79  * ARRBASE = $4000
0800             80  * ARRLEN  = 13334 = $3416
0800             81  * MAXDIG  = 4000  = $0FA0
0800             82  * Last cell ptr  = $A82A  ($4000+(13334-1)*2)
0800             83  * Last denom     = 26669  = $682D  (2*13334+1)
0800             84  * ------------------------------------------------
0000             85  ARBASLO  EQU $00
0040             86  ARBASHI  EQU $40
0016             87  ARLENLO  EQU $16
0034             88  ARLENHI  EQU $34
00A0             89  MXDIGLO  EQU $A0
000F             90  MXDIGHI  EQU $0F
002A             91  LSTPLO   EQU $2A
00A8             92  LSTPHI   EQU $A8
002D             93  LSTDLO   EQU $2D
0068             94  LSTDHI   EQU $68
0800             95  *
0800             96  * Apple II ROM entry points
FDED             97  COUT     EQU $FDED
FC58             98  HOME     EQU $FC58
0800             99  *
0800            100  * ------------------------------------------------
0800            101  * PROGRAM ENTRY POINT
0800            102  * ------------------------------------------------
0900            103           ORG $0900
0900            104  *
0900 20 58 FC   105  START    JSR HOME
0903 20 90 0C   106           JSR BANNER
0906 20 2A 09   107           JSR INITARR
0909 20 76 09   108           JSR INITZP
090C            109  *
090C            110  * Main digit loop.
090C            111  * Each pass produces one predigit.
090C            112  * EMIT converts predigits to confirmed digits.
090C            113  * Loop until DCHI:DCLO reaches MAXDIG.
090C            114  *
090C 20 97 09   115  DIGLOOP  JSR PASS
090F 20 CC 0B   116           JSR EMIT
0912 AD 11 00   117           LDA DCLO
0915 C9 A0      118           CMP #MXDIGLO
0917 D0 F3      119           BNE DIGLOOP
```

```
0919 AD 12 00   120           LDA DCHI
091C C9 0F      121           CMP #MXDIGHI
091E D0 EC      122           BNE DIGLOOP
0920            123  *
0920 20 34 0C   124           JSR FLUSH
0923 20 85 0C   125           JSR NEWLIN
0926 20 D2 0C   126           JSR DONE
0929 60         127           RTS
092A            128  *
092A            129  * ---------------------------------------------
092A            130  * INITARR
092A            131  * Fill all cells with value 2.
092A            132  * Cell 0 at $3FFE/$3FFF.
092A            133  * Cells 1..ARRLEN at ARRBASE+(i-1)*2.
092A            134  * Each cell: low byte first, high byte second.
092A            135  *
092A            136  * 16-bit counter IXHI:IXLO counts from ARRLEN
092A            137  * down to zero. Borrow handled explicitly:
092A            138  * if IXLO is zero before decrement, IXHI goes
092A            139  * first. Loop exits when both bytes are zero.
092A            140  * ---------------------------------------------
092A            141  *
092A A9 02      142  INITARR  LDA #2
092C 8D FE 3F   143           STA $3FFE
092F A9 00      144           LDA #0
0931 8D FF 3F   145           STA $3FFF
0934            146  *
0934 A9 00      147           LDA #ARBASLO
0936 8D 00 00   148           STA ARLO
0939 A9 40      149           LDA #ARBASHI
093B 8D 01 00   150           STA ARHI
093E            151  *
093E A9 16      152           LDA #ARLENLO
0940 8D 04 00   153           STA IXLO
0943 A9 34      154           LDA #ARLENHI
0945 8D 05 00   155           STA IXHI
0948            156  *
0948 A0 00      157  INILP    LDY #0
094A A9 02      158           LDA #2
094C 91 00      159           STA ($00),Y
094E C8         160           INY
094F A9 00      161           LDA #0
0951 91 00      162           STA ($00),Y
0953            163  *
0953 18         164           CLC
0954 AD 00 00   165           LDA ARLO
0957 69 02      166           ADC #2
0959 8D 00 00   167           STA ARLO
095C 90 03      168           BCC INISKP
095E EE 01 00   169           INC ARHI
0961 EA         170  INISKP   NOP
0962            171  *
0962 AD 04 00   172           LDA IXLO
0965 D0 03      173           BNE INIDEC
0967 CE 05 00   174           DEC IXHI
096A CE 04 00   175  INIDEC   DEC IXLO
096D            176  *
096D AD 04 00   177           LDA IXLO
```

```
0970 0D 05 00   178           ORA IXHI
0973 D0 D3      179           BNE INILP
0975 60         180           RTS
0976            181  *
0976            182  * ---------------------------------------------
0976            183  * INITZP
0976            184  * Zero all working zero page variables.
0976            185  * ---------------------------------------------
0976            186  *
0976 A9 00      187  INITZP   LDA #0
0978 8D 09 00   188           STA CL
097B 8D 0A 00   189           STA CM
097E 8D 0B 00   190           STA CH
0981 8D 0F 00   191           STA PRDIG
0984 8D 10 00   192           STA NINES
0987 8D 19 00   193           STA SAVDIG
098A 8D 11 00   194           STA DCLO
098D 8D 12 00   195           STA DCHI
0990 8D 13 00   196           STA COL
0993 8D 1C 00   197           STA GOTSAV
0996 60         198           RTS
0997            199  *
0997            200  * ---------------------------------------------
0997            201  * PASS
0997            202  * One full right-to-left spigot pass.
0997            203  * On return PRDIG holds raw predigit (0-10).
0997            204  * ---------------------------------------------
0997            205  *
0997 A9 00      206  PASS     LDA #0
0999 8D 09 00   207           STA CL
099C 8D 0A 00   208           STA CM
099F 8D 0B 00   209           STA CH
09A2            210  *
09A2 A9 16      211           LDA #ARLENLO
09A4 8D 04 00   212           STA IXLO
09A7 A9 34      213           LDA #ARLENHI
09A9 8D 05 00   214           STA IXHI
09AC            215  *
09AC A9 2A      216           LDA #LSTPLO
09AE 8D 00 00   217           STA ARLO
09B1 A9 A8      218           LDA #LSTPHI
09B3 8D 01 00   219           STA ARHI
09B6            220  *
09B6 A9 2D      221           LDA #LSTDLO
09B8 8D 02 00   222           STA DNLO
09BB A9 68      223           LDA #LSTDHI
09BD 8D 03 00   224           STA DNHI
09C0            225  *
09C0            226  * ---------------------------------------------
09C0            227  * PASSLP - one cell per iteration
09C0            228  * ---------------------------------------------
09C0            229  *
09C0 A0 00      230  PASSLP   LDY #0
09C2 B1 00      231           LDA ($00),Y
09C4 8D 06 00   232           STA XL
09C7 C8         233           INY
09C8 B1 00      234           LDA ($00),Y
09CA 8D 07 00   235           STA XM
```

```
09CD A9 00      236           LDA #0
09CF 8D 08 00   237           STA XH
09D2            238  *
09D2            239  * XH:XM:XL = A[i] * 10
09D2            240  * x*10 = (x*2) + (x*8)
09D2            241  * Shift left once -> x*2, save in QH:QM:QL
09D2            242  * Shift left twice more -> x*8
09D2            243  * Add saved x*2 to get x*10
09D2            244  *
09D2 0E 06 00   245           ASL XL
09D5 2E 07 00   246           ROL XM
09D8 2E 08 00   247           ROL XH
09DB AD 06 00   248           LDA XL
09DE 8D 0C 00   249           STA QL
09E1 AD 07 00   250           LDA XM
09E4 8D 0D 00   251           STA QM
09E7 AD 08 00   252           LDA XH
09EA 8D 0E 00   253           STA QH
09ED 0E 06 00   254           ASL XL
09F0 2E 07 00   255           ROL XM
09F3 2E 08 00   256           ROL XH
09F6 0E 06 00   257           ASL XL
09F9 2E 07 00   258           ROL XM
09FC 2E 08 00   259           ROL XH
09FF 18         260           CLC
0A00 AD 06 00   261           LDA XL
0A03 6D 0C 00   262           ADC QL
0A06 8D 06 00   263           STA XL
0A09 AD 07 00   264           LDA XM
0A0C 6D 0D 00   265           ADC QM
0A0F 8D 07 00   266           STA XM
0A12 AD 08 00   267           LDA XH
0A15 6D 0E 00   268           ADC QH
0A18 8D 08 00   269           STA XH
0A1B            270  *
0A1B            271  * XH:XM:XL = A[i]*10 + carry
0A1B 18         272           CLC
0A1C AD 06 00   273           LDA XL
0A1F 6D 09 00   274           ADC CL
0A22 8D 06 00   275           STA XL
0A25 AD 07 00   276           LDA XM
0A28 6D 0A 00   277           ADC CM
0A2B 8D 07 00   278           STA XM
0A2E AD 08 00   279           LDA XH
0A31 6D 0B 00   280           ADC CH
0A34 8D 08 00   281           STA XH
0A37            282  *
0A37            283  * Divide by denominator (2i+1)
0A37 AD 02 00   284           LDA DNLO
0A3A 8D 15 00   285           STA DVLO
0A3D AD 03 00   286           LDA DNHI
0A40 8D 16 00   287           STA DVHI
0A43 20 32 0B   288           JSR DIV24
0A46            289  *
0A46            290  * Store remainder back into A[i]
0A46 A0 00      291           LDY #0
0A48 AD 17 00   292           LDA RELO
0A4B 91 00      293           STA ($00),Y
```

```
0A4D C8          294           INY
0A4E AD 18 00    295           LDA REHI
0A51 91 00       296           STA ($00),Y
0A53             297  *
0A53             298  * New carry = quotient * i
0A53             299  * QL holds quotient (always < 11, fits in 8 bits)
0A53             300  * Copy i to MLLO/MLHI to protect IXLO/IXHI
0A53 AD 04 00    301           LDA IXLO
0A56 8D 1A 00    302           STA MLLO
0A59 AD 05 00    303           LDA IXHI
0A5C 8D 1B 00    304           STA MLHI
0A5F AD 0C 00    305           LDA QL
0A62 8D 14 00    306           STA TMP
0A65 20 95 0B    307           JSR MUL16CRY
0A68             308  *
0A68             309  * Decrement i (16-bit, explicit borrow)
0A68 AD 04 00    310           LDA IXLO
0A6B D0 03       311           BNE IXDEC
0A6D CE 05 00    312           DEC IXHI
0A70 CE 04 00    313  IXDEC    DEC IXLO
0A73             314  *
0A73             315  * Denominator for next cell = current - 2
0A73 AD 02 00    316           LDA DNLO
0A76 38          317           SEC
0A77 E9 02       318           SBC #2
0A79 8D 02 00    319           STA DNLO
0A7C B0 03       320           BCS DNDONE
0A7E CE 03 00    321           DEC DNHI
0A81 EA          322  DNDONE   NOP
0A82             323  *
0A82             324  * Array pointer back by 2 bytes
0A82 AD 00 00    325           LDA ARLO
0A85 38          326           SEC
0A86 E9 02       327           SBC #2
0A88 8D 00 00    328           STA ARLO
0A8B B0 03       329           BCS ARDONE
0A8D CE 01 00    330           DEC ARHI
0A90 EA          331  ARDONE   NOP
0A91             332  *
0A91             333  * Continue while i != 0
0A91             334  * BNE PASSLP would be out of range (loop is >128 bytes).
0A91             335  * Invert: branch over a JMP when zero (done),
0A91             336  * otherwise JMP back to top of loop.
0A91 AD 04 00    337           LDA IXLO
0A94 0D 05 00    338           ORA IXHI
0A97 F0 03       339           BEQ PASSDONE
0A99 4C C0 09    340           JMP PASSLP
0A9C EA          341  PASSDONE NOP
0A9D             342  *
0A9D             343  * ------------------------------------------------
0A9D             344  * Cell 0 at $3FFE/$3FFF, denominator = 10
0A9D             345  * predigit = (A[0]*10 + carry) DIV 10
0A9D             346  * A[0]     = (A[0]*10 + carry) MOD 10
0A9D             347  * ------------------------------------------------
0A9D             348  *
0A9D AD FE 3F    349           LDA $3FFE
0AA0 8D 06 00    350           STA XL
0AA3 AD FF 3F    351           LDA $3FFF
```

```
0AA6 8D 07 00   352           STA XM
0AA9 A9 00      353           LDA #0
0AAB 8D 08 00   354           STA XH
0AAE            355  *
0AAE 0E 06 00   356           ASL XL
0AB1 2E 07 00   357           ROL XM
0AB4 2E 08 00   358           ROL XH
0AB7 AD 06 00   359           LDA XL
0ABA 8D 0C 00   360           STA QL
0ABD AD 07 00   361           LDA XM
0AC0 8D 0D 00   362           STA QM
0AC3 AD 08 00   363           LDA XH
0AC6 8D 0E 00   364           STA QH
0AC9 0E 06 00   365           ASL XL
0ACC 2E 07 00   366           ROL XM
0ACF 2E 08 00   367           ROL XH
0AD2 0E 06 00   368           ASL XL
0AD5 2E 07 00   369           ROL XM
0AD8 2E 08 00   370           ROL XH
0ADB 18         371           CLC
0ADC AD 06 00   372           LDA XL
0ADF 6D 0C 00   373           ADC QL
0AE2 8D 06 00   374           STA XL
0AE5 AD 07 00   375           LDA XM
0AE8 6D 0D 00   376           ADC QM
0AEB 8D 07 00   377           STA XM
0AEE AD 08 00   378           LDA XH
0AF1 6D 0E 00   379           ADC QH
0AF4 8D 08 00   380           STA XH
0AF7            381  *
0AF7 18         382           CLC
0AF8 AD 06 00   383           LDA XL
0AFB 6D 09 00   384           ADC CL
0AFE 8D 06 00   385           STA XL
0B01 AD 07 00   386           LDA XM
0B04 6D 0A 00   387           ADC CM
0B07 8D 07 00   388           STA XM
0B0A AD 08 00   389           LDA XH
0B0D 6D 0B 00   390           ADC CH
0B10 8D 08 00   391           STA XH
0B13            392  *
0B13 A9 0A      393           LDA #10
0B15 8D 15 00   394           STA DVLO
0B18 A9 00      395           LDA #0
0B1A 8D 16 00   396           STA DVHI
0B1D 20 32 0B   397           JSR DIV24
0B20            398  *
0B20            399  * Store remainder (mod 10) back to A[0]
0B20 AD 17 00   400           LDA RELO
0B23 8D FE 3F   401           STA $3FFE
0B26 A9 00      402           LDA #0
0B28 8D FF 3F   403           STA $3FFF
0B2B            404  *
0B2B AD 0C 00   405           LDA QL
0B2E 8D 0F 00   406           STA PRDIG
0B31 60         407           RTS
0B32            408  *
0B32            409  * ---------------------------------------------
```

```
0B32            410  * DIV24
0B32            411  * 24-bit / 16-bit division.
0B32            412  *
0B32            413  * Input:  XH:XM:XL  = 24-bit dividend
0B32            414  *         DVHI:DVLO = 16-bit divisor
0B32            415  * Output: QH:QM:QL  = 24-bit quotient
0B32            416  *         REHI:RELO = 16-bit remainder
0B32            417  *
0B32            418  * Method: shift-and-subtract, 24 iterations.
0B32            419  * X register used as counter, destroyed on exit.
0B32            420  * ------------------------------------------------
0B32            421  *
0B32 A9 00      422  DIV24    LDA #0
0B34 8D 0C 00   423           STA QL
0B37 8D 0D 00   424           STA QM
0B3A 8D 0E 00   425           STA QH
0B3D 8D 17 00   426           STA RELO
0B40 8D 18 00   427           STA REHI
0B43 A2 18      428           LDX #24
0B45            429  *
0B45 0E 06 00   430  D24LP    ASL XL
0B48 2E 07 00   431           ROL XM
0B4B 2E 08 00   432           ROL XH
0B4E 2E 17 00   433           ROL RELO
0B51 2E 18 00   434           ROL REHI
0B54 AD 18 00   435           LDA REHI
0B57 CD 16 00   436           CMP DVHI
0B5A 90 2B      437           BCC D24NO
0B5C D0 08      438           BNE D24YES
0B5E AD 17 00   439           LDA RELO
0B61 CD 15 00   440           CMP DVLO
0B64 90 21      441           BCC D24NO
0B66            442  *
0B66 38         443  D24YES   SEC
0B67 AD 17 00   444           LDA RELO
0B6A ED 15 00   445           SBC DVLO
0B6D 8D 17 00   446           STA RELO
0B70 AD 18 00   447           LDA REHI
0B73 ED 16 00   448           SBC DVHI
0B76 8D 18 00   449           STA REHI
0B79 38         450           SEC
0B7A 2E 0C 00   451           ROL QL
0B7D 2E 0D 00   452           ROL QM
0B80 2E 0E 00   453           ROL QH
0B83 CA         454           DEX
0B84 D0 BF      455           BNE D24LP
0B86 60         456           RTS
0B87            457  *
0B87 18         458  D24NO    CLC
0B88 2E 0C 00   459           ROL QL
0B8B 2E 0D 00   460           ROL QM
0B8E 2E 0E 00   461           ROL QH
0B91 CA         462           DEX
0B92 D0 B1      463           BNE D24LP
0B94 60         464           RTS
0B95            465  *
0B95            466  * ------------------------------------------------
0B95            467  * MUL16CRY
```

```
0B95            468  * 8-bit * 16-bit = 24-bit multiply.
0B95            469  *
0B95            470  * Input:  TMP       = 8-bit multiplier (max 10)
0B95            471  *         MLHI:MLLO = 16-bit multiplicand (copy of i)
0B95            472  * Output: CH:CM:CL  = 24-bit product (becomes carry)
0B95            473  *
0B95            474  * MLLO/MLHI are shifted in place and consumed.
0B95            475  * TMP is shifted right and consumed.
0B95            476  * IXLO/IXHI are never touched by this routine.
0B95            477  * Caller must copy IXLO/IXHI to MLLO/MLHI first.
0B95            478  *
0B95            479  * Method: shift-and-add, 8 iterations.
0B95            480  * ------------------------------------------------
0B95            481  *
0B95 A9 00      482  MUL16CRY LDA #0
0B97 8D 09 00   483           STA CL
0B9A 8D 0A 00   484           STA CM
0B9D 8D 0B 00   485           STA CH
0BA0 A2 08      486           LDX #8
0BA2            487  *
0BA2 4E 14 00   488  MULLP    LSR TMP
0BA5 90 1B      489           BCC MULSHFT
0BA7 18         490           CLC
0BA8 AD 09 00   491           LDA CL
0BAB 6D 1A 00   492           ADC MLLO
0BAE 8D 09 00   493           STA CL
0BB1 AD 0A 00   494           LDA CM
0BB4 6D 1B 00   495           ADC MLHI
0BB7 8D 0A 00   496           STA CM
0BBA AD 0B 00   497           LDA CH
0BBD 69 00      498           ADC #0
0BBF 8D 0B 00   499           STA CH
0BC2            500  *
0BC2 0E 1A 00   501  MULSHFT  ASL MLLO
0BC5 2E 1B 00   502           ROL MLHI
0BC8 CA         503           DEX
0BC9 D0 D7      504           BNE MULLP
0BCB 60         505           RTS
0BCC            506  *
0BCC            507  * ------------------------------------------------
0BCC            508  * EMIT
0BCC            509  * Manages predigit buffer, emits confirmed digits.
0BCC            510  *
0BCC            511  * PRDIG = 9:   INC NINES, return
0BCC            512  * PRDIG = 10:  output SAVDIG+1, output NINES zeros,
0BCC            513  *              reset NINES and SAVDIG
0BCC            514  * PRDIG = 0-8: if GOTSAV=0 (no prior digit saved yet):
0BCC            515  *                set GOTSAV=1, save PRDIG, return
0BCC            516  *              else: output SAVDIG, output NINES nines,
0BCC            517  *                save new PRDIG, reset NINES
0BCC            518  * ------------------------------------------------
0BCC            519  *
0BCC AD 0F 00   520  EMIT     LDA PRDIG
0BCF C9 09      521           CMP #9
0BD1 F0 34      522           BEQ EMIT9
0BD3 C9 0A      523           CMP #10
0BD5 F0 34      524           BEQ EMIT10
0BD7            525  *
```

```
0BD7                526  * Normal digit 0-8: check if we have a saved digit yet
0BD7 AD 1C 00       527           LDA GOTSAV
0BDA D0 0C          528           BNE EMITNORM
0BDC                529  *
0BDC                530  * No saved digit yet -- this is the very first predigit
0BDC A9 01          531  EMITFST  LDA #1
0BDE 8D 1C 00       532           STA GOTSAV
0BE1 AD 0F 00       533           LDA PRDIG
0BE4 8D 19 00       534           STA SAVDIG
0BE7 60             535           RTS
0BE8                536  *
0BE8                537  * Normal path: flush saved digit and buffered 9s, save new
0BE8 AD 19 00       538  EMITNORM LDA SAVDIG
0BEB 20 48 0C       539           JSR OUTDIG
0BEE                540  *
0BEE AE 10 00       541           LDX NINES
0BF1 F0 08          542           BEQ EMITSAVE
0BF3 A9 09          543  ENILOOP  LDA #9
0BF5 20 48 0C       544           JSR OUTDIG
0BF8 CA             545           DEX
0BF9 D0 F8          546           BNE ENILOOP
0BFB                547  *
0BFB A9 00          548  EMITSAVE LDA #0
0BFD 8D 10 00       549           STA NINES
0C00 AD 0F 00       550           LDA PRDIG
0C03 8D 19 00       551           STA SAVDIG
0C06 60             552           RTS
0C07                553  *
0C07 EE 10 00       554  EMIT9    INC NINES
0C0A 60             555           RTS
0C0B                556  *
0C0B AD 1C 00       557  EMIT10   LDA GOTSAV
0C0E F0 16          558           BEQ EMIT10Z         ; no saved digit yet, skip out
put
0C10 AD 19 00       559           LDA SAVDIG
0C13 18             560           CLC
0C14 69 01          561           ADC #1
0C16 20 48 0C       562           JSR OUTDIG
0C19 AE 10 00       563           LDX NINES
0C1C F0 08          564           BEQ EMIT10Z
0C1E A9 00          565  EMITZLP  LDA #0
0C20 20 48 0C       566           JSR OUTDIG
0C23 CA             567           DEX
0C24 D0 F8          568           BNE EMITZLP
0C26 A9 01          569  EMIT10Z  LDA #1
0C28 8D 1C 00       570           STA GOTSAV
0C2B A9 00          571           LDA #0
0C2D 8D 10 00       572           STA NINES
0C30 8D 19 00       573           STA SAVDIG
0C33 60             574           RTS
0C34                575  *
0C34                576  * ------------------------------------------------
0C34                577  * FLUSH
0C34                578  * Emit remaining buffered state at end of run.
0C34                579  * ------------------------------------------------
0C34                580  *
0C34 AD 19 00       581  FLUSH    LDA SAVDIG
0C37 20 48 0C       582           JSR OUTDIG
0C3A AE 10 00       583           LDX NINES
```

```
0C3D F0 08      584           BEQ FLDONE
0C3F A9 09      585  FLOOP    LDA #9
0C41 20 48 0C   586           JSR OUTDIG
0C44 CA         587           DEX
0C45 D0 F8      588           BNE FLOOP
0C47 60         589  FLDONE   RTS
0C48            590  *
0C48            591  * ---------------------------------------------
0C48            592  * OUTDIG
0C48            593  * Output one decimal digit (value 0-9) in A.
0C48            594  *
0C48            595  * First digit outputs digit then decimal point.
0C48            596  * Newline every 50 digit positions.
0C48            597  * Increments DCHI:DCLO each call.
0C48            598  *
0C48            599  * Apple II COUT requires high bit set:
0C48            600  *   digit n -> n + $B0
0C48            601  *   period  -> $AE
0C48            602  *   CR      -> $8D
0C48            603  * ---------------------------------------------
0C48            604  *
0C48 48         605  OUTDIG   PHA
0C49 AD 11 00   606           LDA DCLO
0C4C 0D 12 00   607           ORA DCHI
0C4F D0 15      608           BNE OUTDIG2
0C51            609  *
0C51 68         610           PLA
0C52 48         611           PHA
0C53 18         612           CLC
0C54 69 B0      613           ADC #$B0
0C56 20 ED FD   614           JSR COUT
0C59 A9 AE      615           LDA #$AE
0C5B 20 ED FD   616           JSR COUT
0C5E A9 02      617           LDA #2
0C60 8D 13 00   618           STA COL
0C63 4C 7B 0C   619           JMP OUTINC
0C66            620  *
0C66 68         621  OUTDIG2  PLA
0C67 48         622           PHA
0C68 18         623           CLC
0C69 69 B0      624           ADC #$B0
0C6B 20 ED FD   625           JSR COUT
0C6E EE 13 00   626           INC COL
0C71 AD 13 00   627           LDA COL
0C74 C9 32      628           CMP #50
0C76 D0 03      629           BNE OUTINC
0C78 20 85 0C   630           JSR NEWLIN
0C7B            631  *
0C7B 68         632  OUTINC   PLA
0C7C EE 11 00   633           INC DCLO
0C7F D0 03      634           BNE OUTRET
0C81 EE 12 00   635           INC DCHI
0C84 60         636  OUTRET   RTS
0C85            637  *
0C85            638  * ---------------------------------------------
0C85            639  * NEWLIN
0C85            640  * Output carriage return, reset column counter.
0C85            641  * ---------------------------------------------
```

```
0C85                642  *
0C85 A9 8D          643  NEWLIN   LDA #$8D
0C87 20 ED FD       644           JSR COUT
0C8A A9 00          645           LDA #0
0C8C 8D 13 00       646           STA COL
0C8F 60             647           RTS
0C90                648  *
0C90                649  * ---------------------------------------------
0C90                650  * BANNER
0C90                651  * Print startup header using indexed LDA from
0C90                652  * inline table. Zero terminated.
0C90                653  * All bytes pre-encoded with high bit set.
0C90                654  *
0C90                655  * Decodes as:
0C90                656  *   PI SPIGOT - APPLE II    (CR)
0C90                657  *   RABINOWITZ-WAGON 3000 DIGITS  (CR)(CR)
0C90                658  * ---------------------------------------------
0C90                659  *
0C90 A2 00          660  BANNER   LDX #0
0C92 BD 9E 0C       661  BANLP    LDA BANTXT,X
0C95 F0 06          662           BEQ BANDONE
0C97 20 ED FD       663           JSR COUT
0C9A E8             664           INX
0C9B D0 F5          665           BNE BANLP
0C9D 60             666  BANDONE  RTS
0C9E                667  *
0C9E                668  * "PI SPIGOT - APPLE II" + CR
0C9E                669  * P=$D0 I=$C9 ´ ´=$A0 S=$D3 P=$D0 I=$C9 G=$C7
0C9E                670  * O=$CF T=$D4 ´ ´=$A0 -=$AD ´ ´=$A0 A=$C1
0C9E                671  * P=$D0 P=$D0 L=$CC E=$C5 ´ ´=$A0 I=$C9 I=$C9
0C9E                672  * CR=$8D
0C9E D0 C9 A0       673  BANTXT   HEX D0C9A0D3D0C9C7CF
0CA1 D3 D0 C9
0CA4 C7 CF
0CA6 D4 A0 AD       674           HEX D4A0ADA0C1D0D0CC
0CA9 A0 C1 D0
0CAC D0 CC
0CAE C5 A0 C9       675           HEX C5A0C9C9
0CB1 C9
0CB2 8D             676           HEX 8D
0CB3                677  * "RABINOWITZ-WAGON 4000 DIGITS" + CR + CR + $00
0CB3 D2 C1 C2       678           HEX D2C1C2C9CECFD7C9
0CB6 C9 CE CF
0CB9 D7 C9
0CBB D4 DA AD       679           HEX D4DAADD7C1C7CFCE
0CBE D7 C1 C7
0CC1 CF CE
0CC3 A0 B4 B0       680           HEX A0B4B0B0B0A0C4C9
0CC6 B0 B0 A0
0CC9 C4 C9
0CCB C7 C9 D4       681           HEX C7C9D4D3
0CCE D3
0CCF 8D 8D          682           HEX 8D8D
0CD1 00             683           BYT 0
0CD2                684  *
0CD2                685  * ---------------------------------------------
0CD2                686  * DONE
0CD2                687  * Print completion message.
```

```
0CD2            688  * Decodes as: CR + "DONE." + CR
0CD2            689  * D=$C4 O=$CF N=$CE E=$C5 .=$AE
0CD2            690  * ------------------------------------------------
0CD2            691  *
0CD2 A2 00      692  DONE     LDX #0
0CD4 BD E0 0C   693  DONELP   LDA DONETXT,X
0CD7 F0 06      694           BEQ DONEEND
0CD9 20 ED FD   695           JSR COUT
0CDC E8         696           INX
0CDD D0 F5      697           BNE DONELP
0CDF 60         698  DONEEND  RTS
0CE0            699  *
0CE0 8D         700  DONETXT  HEX 8D
0CE1 C4 CF CE   701           HEX C4CFCEC5AE
0CE4 C5 AE
0CE6 8D         702           HEX 8D
0CE7 00         703           BYT 0
0CE8            704  *
0CE8            705           END


***** END OF ASSEMBLY

!!bsaveBBPI,A$900,L$400
BSAVEBBPI,A$900,L$400

!
```