```
zero:                   I 48
nine:                   C ':' ascii right after 9
newline:                I 10
start:                  F 0.0
magn:                   W 1
magn0:                  F 1.0
magn1:                  F 10.0
magn2:                  F 100.0
magn3:                  F 1000.0
big:                    F 10000.0
zero1:                  F 0.1
zero2:                  F 0.01
zero3:                  F 0.001
zero4:                  F 0.0001
add:                    C '+'
sub:                    C '-'
mul:                    C '*'
div:                    C '/'
dot:                    C '.'
equ:                    C '='
esc:                    I 27
error:                  C 'E'
result:                 W 1
op:                     W 1
initial:                F 0.0
operator:               W 1
integer:                W 1
fractional:             W 1
fraction:               W 1
checkFrac:              W 1
ready:                  W 1
executable:             W 1
divisionbyzero:         W 1
negative:               W 1
true:                   I 1
false:                  I -1
fsign:                  F -1.0
current:                W 1
decimal:                W 1
zeros:                  W 1
position0:              I 0
position1:              I 1
position2:              I 2
position3:              I 3
position4:              I 4
pitoprint:              W 1
piweight:               I 1
```

```
picharminus:                                C '-'
pichar0:                                    I 48
pimask:                                     H 80000000
piC0:                                       I 0
piC1:                                       I 1
piC10:                                      I 10
input:                                      W 1
printint:                                   W 1
FL:                                         W 1
digits:                                     W 1
W 0  #  read input, and do the calculation

main: LDA initial
       STA FL           initial FL
       LDA false
       STA operator     initial operator
       STA ready        initial ready to print
       STA executable   initial operator execution
       STA divisionbyzero
initialOp: LDA false
       STA checkFrac
       STA negative
       LDA initial
       STA integer            initial integer part
       STA fractional         initial fraction part
       LDA magn1
       STA fraction
       LDA piC0
       STA digits
loadInput: INP opsys
       STA input        stores input
       SUB zero
       AND pimask       check if the input >= zero
       BZE biggerThanZero     if it is bunch to 'biggerThanZero' if it is
       LDA input        else check '+ - * / .'
       XOR dot          check '.'
       BZE isDot        if it is bruch to 'isDot'
       LDA input        else check check '+ - * /'
       XOR add          check '+'
       BZE isOperator   if it is brunch to 'isOperator'
       LDA input        else check check ' - * /'
       XOR sub          check '-'
       BZE isOperator   if it is brunch to 'isOperator'
       LDA input        else check check ' * /'
       XOR mul          check '*'
       BZE isOperator   if it is brunch to 'isOperator'
       LDA input        else check check ' /'
       XOR div          check '/'
```

```
        BZE isOperator    if it is brunch to 'isOperator'
        BUN loadInput     invalid input, bunch to read next input
biggerThanZero: LDA input
        SUB nine
        AND pimask        check if input >9
        BZE notNumber     bunch to 'notNumber' if input > 9
        BUN isNumber      0 <= input <= 9
notNumber: LDA input
        XOR equ           check if input is '='
        BZE isEqu         bunch to 'isEqu' if it is
        BUN loadInput     invalid input, bunch to read next input

isNumber:LDA digits
        XOR position4     check if the number already has 4 digits
        BZE loadInput
        LDA piC1
        ADD digits
        STA digits
        LDA true
        STA executable    when we have input set executable to truth
        LDA input
        OUT opsys         print number '0-9' (integer part)
        SUB zero          convert ascii to number
        STA input         stores number
        CIF input         convert to float
        STA input         stores, update to float
        LDA checkFrac     check if the number is the fractional part
        AND pimask        checkFrac >= 0
        BZE isfractional  if checkFrac is true
        LDA integer       else not fractional, load input
        XOR initial       check if integer part still the initial value

        BZE isInitial     if it is bunch to 'isInitial'
        LDA integer       if it is not
        FLM magn1         integer * 10
        FLA input         + input
        STA integer       update integer
        BUN loadInput     bunch to read next input
isInitial: LDA input
        STA integer
        BUN loadInput           bunch to read next input

isfractional: LDA input         load input
        FLD fraction
        FLA fractional
        STA fractional
        LDA fraction
        FLM magn1
```

```
        STA fraction
        BUN loadInput     bunch to read next input
isDot: LDA true
        STA executable
        LDA input
        OUT opsys         print '.'
        LDA true          bool true
        STA checkFrac     update checkFrac to true
        BUN loadInput     bunch to read next input

isOperator: LDA           executable check if it is two operations in row
        XOR false         if not executable
        BZE notExe
        LDA input
        OUT opsys         print operator
        LDA operator
        XOR false         check if it is the first operation
        BZE initialFL     if it is bunch to 'initialFL'
        LDA false
        STA executable
        BUN updateFL

notExe:LDA negative
        XOR true
        BZE loadInput     illegal input bunch to read next input
        LDA input
        XOR sub
        BZE isNegative
        BUN loadInput     illegal input bunch to read next input

isNegative:LDA true
        STA negative      set next number be negative
        LDA input
        OUT opsys
        BUN loadInput     illegal input bunch to read next input

updateFL: LDA operator
        XOR add
        BZE addFL         if the previous operator is '+'
        LDA operator
        XOR sub
        BZE subFL         if the previous operator is '-'
        LDA operator
        XOR mul
        BZE mulFL         if the previous operator is '*'
        LDA operator
        XOR div
        BZE divFL         if the previous operator is '/'
```

```
initialFL: LDA false
       STA executable
       LDA integer      else this is the first operator
       FLA fractional
       STA FL           update FL
       LDA negative
       XOR true
       BZE flipSign
backInitialFL: LDA input
       STA operator
       LDA ready
       XOR true
       BZE wait
       BUN initialOp          bunch to read next input

flipSign: LDA FL
       FLM fsign
       STA FL
       LDA false
       STA negative
       BUN backInitialFL

addFL: LDA negative           check if its adding negative number
       XOR true
       BZE negAdd
   LDA FL
       FLA integer
       FLA fractional
       STA FL
backaddFL: LDA ready
       XOR true
       BZE wait
       LDA input
       STA operator
       BUN initialOp
negAdd: LDA FL          add negative
       FLS integer
       FLS fractional
       STA FL
       LDA false
       STA negative
       BUN backaddFL

subFL: LDA negative     check if it is minus a negative number
       XOR true
       BZE negSub
       LDA FL
       FLS integer
```

```
        FLS fractional
        STA FL
backsubFL: LDA ready
        XOR true
        BZE wait
        LDA input
        STA operator
        BUN initialOp
negSub: LDA FL          sub negative number
        FLA integer
        FLA fractional
        STA FL
        LDA false
        STA negative
        BUN backsubFL

mulFL: LDA integer      check if it times a negative number
        FLA fractional
        FLM FL
        STA FL
        LDA negative
        XOR true
        BZE negMul
backmulFL: LDA ready
        XOR true
        BZE wait
        LDA input
        STA operator
        BUN initialOp
negMul: LDA FL          times negative number
        FLM fsign
        STA FL
        LDA false
        STA negative
        BUN backmulFL

divFL: LDA integer      check if div a negative number
        FLA fractional
        STA fractional
        XOR start       check if it is 0.0
        BZE divZero
        LDA fractional
        FLM fsign       check if it is -0.0
        XOR start
        BZE divZero
back: LDA FL
        FLD fractional
        STA FL
```

```
       LDA negative
       XOR true
       BZE negDiv
backdivFL: LDA ready     div a negative number
       LDA ready
       XOR true
       BZE wait
       LDA input
       STA operator
       BUN initialOp
negDiv: LDA FL
       FLM fsign
       STA FL
       LDA false
       STA negative
       BUN backdivFL
divZero: LDA true
       STA divisionbyzero
       BUN back

isEqu: LDA executable
       XOR false
       BZE loadInput
       LDA input
       OUT opsys         print '='
       LDA true
       STA ready
       BUN updateFL
wait: INP opsys          ask for output
       XOR esc
       BZE display       if the input is 'esc' display the output
       BUN wait          else print 'input Error'

W 0  #  print the results with 4 digits precision

display: LDA divisionbyzero
       XOR true
       BZE toobig
   LDA FL
       XOR start
       BZE printZero
       LDA FL
   AND pimask            check sign
   BZE postive           goto positive if >= 0.0
       LDA FL
       FLM fsign         flip the sign floating number
       STA FL            stores result
       FLS big           check if big
```

```
        AND pimask
        BZE toobig              too big >= 10000.0
        LDA picharminus
        OUT opsys
        BUN decimalPoint        not too big, bunch to check decimal point
postive:LDA FL
        FLS big                 check if too big
        AND pimask
        BZE toobig
        BUN decimalPoint        smaller than big, bunch to check decimal point
toobig: LDA error
        OUT opsys
        BUN nextComputation    number is too big print error msg
decimalPoint: LDA FL
        FLS magn3               check if have 4 digits, >= 1000
        AND pimask
        BZE digits4             bigger than zero, has 4 digits
        LDA FL
        FLS magn2               check have 3 digits, >= 100
        AND pimask
        BZE digits3             bigger than zero, has 3 digits
        LDA FL
        FLS magn1               check have 2 digits >= 10
        AND pimask
        BZE digits2             bigger than zero, has 2 digits
        LDA FL
        FLS magn0               check have 1 digits >= 1
        AND pimask
        BZE digits1             bigger than zero, has 1 digits
        LDA FL
        FLS zero1
        AND pimask              check if smaller than 0.1
        BZE zeros1
        LDA FL
        FLS zero2
        AND pimask              check if smaller than 0.01
        BZE zeros2
        LDA FL
        FLS zero3
        AND pimask       check if smaller than 0.001
        BZE zeros3
        LDA FL
        FLS zero4
        AND pimask       check if smaller than 0.001
        BZE zeros4
        LDA zero
        OUT opsys
        BUN main
```

```
digits4:LDA position4    if the number has at least 4 digits for integer part
        STA decimal      update the decimal point
        LDA FL
        STA pitoprint
        CFI pitoprint    convert to int
        STA pitoprint
        BUN print
digits3:LDA position3    if the number has at least 3 digits for integer part
        STA decimal      update the decimal point
        LDA FL
        FLM magn1
        STA pitoprint
        CFI pitoprint
        STA pitoprint
        BUN print        print the number
digits2:LDA position2    if the number has at least 2 digits for integer part
        STA decimal      update the decimal point
        LDA FL
        FLM magn2
        STA pitoprint
        CFI pitoprint
        STA pitoprint
        BUN print
digits1:LDA position1    if the number has at least 1 digits for integer part
       STA decimal
       LDA FL
       FLM magn3
       STA pitoprint
       CFI pitoprint
       STA pitoprint
       BUN print
zeros1: LDA position0    if the number has at least 1 digits for fractional part
       STA decimal
       LDA FL
       FLM big
       STA pitoprint
       CFI pitoprint
       STA pitoprint
       BUN print
zeros2: LDA position4    if the number has at least 2 digits for fractional part
       STA decimal
       LDA FL
       FLM big
       STA pitoprint
       CFI pitoprint
       STA pitoprint
       LDA dot
       OUT opsys
```

```
      LDA zero
      OUT opsys
      BUN print
zeros3: LDA position4    if the number has at least 3 digits for fractional part
      STA decimal
      LDA FL
      FLM big
      STA pitoprint
      CFI pitoprint
      STA pitoprint
      LDA dot
      OUT opsys
      LDA zero
      OUT opsys
      OUT opsys
      BUN print
zeros4: LDA position4    if the number has at least 4 digits for fractional part
      STA decimal
      LDA FL
      FLM big
      STA pitoprint
      CFI pitoprint
      STA pitoprint
      LDA dot
      OUT opsys
      LDA zero
      OUT opsys
      OUT opsys
      OUT opsys
      BUN print
print: LDA pitoprint
      AND pimask
      BZE piwhile1
      LDA picharminus
      OUT opsys
      LDA pitoprint
      SUB pimask
      STA pitoprint
piwhile1: LDA pitoprint
      DIV piweight
      SUB piC10
      AND pimask
      BZE pibody1
      BUN piwhile2
pibody1: LDA piweight
      MUL piC10
      STA piweight
      BUN piwhile1
```

```
piwhile2: LDA pitoprint
      DIV piweight
      ADD pichar0
      STA current             store the digits to print
      LDA decimal             load decimal position
      SUB piC1
      STA decimal
      LDA decimal
      AND pimask
      BZE printcurrent        if >= 0
      LDA dot
      OUT opsys
      LDA pichar0             prevent it print again
      STA decimal
      BUN printcurrent
printcurrent: LDA current
      OUT opsys
      LDA piweight
      XOR piC1
      BZE pireturn
      LDA pitoprint
      MOD piweight
      STA pitoprint
      LDA piweight
      DIV piC10
      STA piweight
      BUN piwhile2
printZero: LDA zero
      OUT opsys
      LDA dot
      OUT opsys
pireturn: LDA decimal
      XOR piC0
      BZE printDot
nextComputation:LDA newline
      OUT opsys
      BUN main                main loop, read the next equation
printDot: LDA dot
      OUT opsys
      BUN nextComputation
```