```
                .data

FloatXYZ:       .float  156.125
FloatX:         .float   48.125
FloatY:         .float   1024.
Float_X_plus_Y: .float
X:              .word
Y:              .word
X_F:            .word
X_E:            .word
Y_F:            .word
Y_E:            .word
X_time_Y:       .word
X_plus_Y:       .word
X_plus_Y_F:     .word
X_plus_Y_E:     .word
X_plus_Y_S:     .word
small_F:        .word
diff:           .word
F_Mask:         .word   0x007fffff
E_Mask:         .word   0x7f800000
S_Mask:         .word   0x80000000
Hidden_one:     .word   0x00800000
zero:           .word   0
max_F:          .word   0x01000000
                .text
                                                        # Extract E (exponent)
and F (significand).
__start:        move    X, FloatX
                and     X_F, X, F_Mask                  # get X_F
                or      X_F, X_F, Hidden_one            # add hidden bit
                bgtz    X, DoX_E                        # skip if positive
                sub     X_F, zero, X_F                  # convert to 2’s comp.
DoX_E:          and     X_E, X, E_Mask                  # get X_E
                srl     X_E, X_E, 23                    # align
                sub     X_E, X_E, 127                   # convert to 2’s comp.
                move    Y, FloatY
                and     Y_F, Y, F_Mask                  # get Y_F
                or      Y_F, Y_F, Hidden_one            # add hidden bit
                bgtz    Y, DoY_E                        # skip if positive
                sub     Y_F, zero, Y_F                  # convert to 2’s comp.
DoY_E:          and     Y_E, Y, E_Mask                  # get Y_E
                srl     Y_E, Y_E, 23                    # align
                sub     Y_E, Y_E, 127                   # convert to 2’s comp.
                                                        # Determine which input
is smaller
                sub     diff, Y_E, X_E
                bltz    diff, X_bigger
                move    X_plus_Y_E, Y_E
                move    X_plus_Y_F, Y_F
                move    small_F, X_F
                b       LittleF
X_bigger:       move    X_plus_Y_E, X_E
                move    X_plus_Y_F, X_F
                move    small_F, Y_F
                sub     diff, zero, diff
LittleF:        sra     small_F, small_F, diff                  # denormalize
little F
                add     X_plus_Y_F, small_F, X_plus_Y_F         # add Fs
                and     X_plus_Y_S, X_plus_Y_F, S_Mask
                beqz    X_plus_Y_F, Zero
                bgez    X_plus_Y_F, L1                          # skip if
positive
                sub     X_plus_Y_F, zero, X_plus_Y_F            # convert to
sign/mag
L1:             move    X_plus_Y_E, X_plus_Y_E
                blt     X_plus_Y_F, max_F, NotTooBig            # skip if no
overflow
                srl     X_plus_Y_F, X_plus_Y_F, 1               # divide F by 2
                add     X_plus_Y_E, X_plus_Y_E, 1               # adjust E
                b       Normalized
Zero:           move    Float_X_plus_Y, 0
                b       Finished
TooSmall:       sll     X_plus_Y_F, X_plus_Y_F, 1               # multiply F by
2
                sub     X_plus_Y_E, X_plus_Y_E, 1               # adjust E
NotTooBig:      blt     X_plus_Y_F, Hidden_one, TooSmall        # check if still
too big
Normalized:     sub     X_plus_Y_F, X_plus_Y_F, Hidden_one      # delete hidden
one
                add     X_plus_Y_E, X_plus_Y_E, 127             # convert to
bias-127
                sll     X_plus_Y_E, X_plus_Y_E, 23              # align properly

                or      X_plus_Y, X_plus_Y_E, X_plus_Y_F        # merge E, F
                or      X_plus_Y, X_plus_Y, X_plus_Y_S          # merge S
                move    Float_X_plus_Y, X_plus_Y                # move to
floating point
Finished:       put     Float_X_plus_Y
                done
```