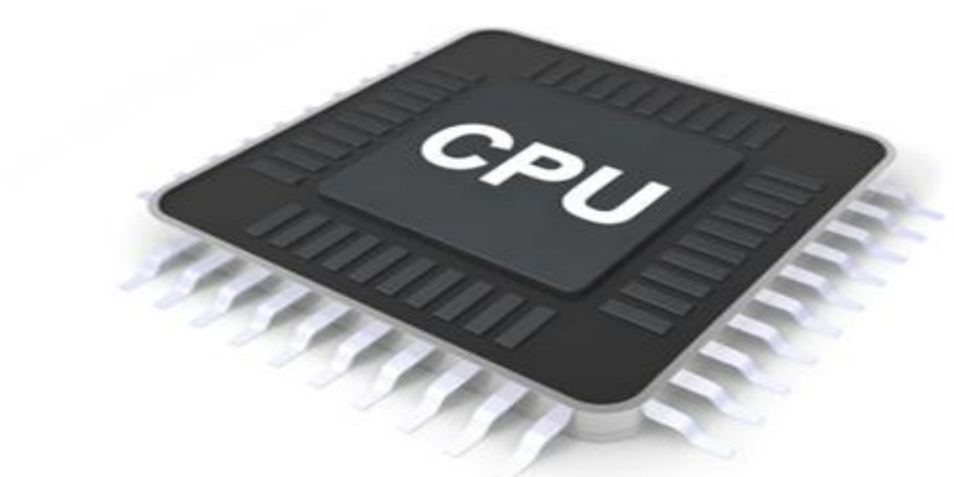

Floating Point Binary

ENEE 3582

Microp

Real Binary

- Given any unsigned real number in base x:

$$(a_{n-1}a_{n-2}a_{n-3}...a_2a_1a_0).(a_{-1}a_{-2}a_{-3}...a_{-(m-2)}a_{-(m-1)}a_{-m})$$

  a_i = 0, 1, …., x-1.

  n numbers before the decimal, m numbers after decimal.

  Binary: x = 2 ; a_i = 0 or 1.

- To convert to base 10:

$$(a_{n-1}x^{n-1} + a_{n-2}x^{n-2} +\dots a_1x^1 + a_0x^0)+(a_{-1}x^{-1} + a_{-2}x^{-2} +\dots a_{-(m-1)}x^{-(m-2)} + a_{-m}x^{-m})$$

- Example: 1011.1011 b

m :					1	2	3	4
n :	4	3	2	1				
a_i :	a_3	a_2	a_1	a_0	a_{-1}	a_{-2}	a_{-3}	a_{-4}
	1	0	1	1 .	1	0	1	1
	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}

Converting from Binary to Decimal

2^n	Binary	Fraction	Decimal
2^{-1}	0b0.1	1/2	0.5
2^{-2}	0b0.01	1/4	0.25
2^{-3}	0b0.001	1/8	0.125
2^{-4}	0b0.0001	1/16	0.0625
2^{-5}	0b0.00001	1/32	0.03125
2^{-6}	0b0.000001	1/64	0.015625
2^{-7}	0b0.0000001	1/128	0.0078125

0b1011.1011 = $1*2^3+0*2^2+1*2^1+1*2^0+1*2^{-1}+0*2^{-2}+1*2^{-3}+1*2^{-4}$

= 8 +2 +1 +1/2 +1/8 +1/16

= 11.6875 or 11 11/16

Converting to decimal is easy. Example:

0b1101.100101 $\rightarrow$ 0b1101 = 13, 0.100101 = 0.578125

FP Decimal to FP Binary

- FP Decimal Format: N.P
  - N = integer
  - P = fraction
  - E.g. 123.456 => N = 123, P = 0.456
- Convert N to decimal using successive division by 2
  - Use remainder of division to form binary integer
  - Stop when the quotient is 0
- Convert P to binary using successive multiplication by 2
  - Use integer of multiplication to form binary fraction
  - Stop when the multiplication is 0 or after specific number of bits
  - Will need 16 bits of fraction to get good precision

Integer: Decimal to Binary

N	2	Q	R
123	2	61	1
61	2	30	1
30	2	15	0
15	2	7	1
7	2	3	1
3	2	1	1
1	2	0	1

123 = 0b1111011

Stop when the Quotient of Division is 0

Fraction: Decimal to Binary

P	2		N
0.375	2	0.75	0
0.75	2	1.5	1
0.5	2	1.0	1
0.0	2	0	0

$0.375 = 0b0.0110 = \frac{0b0110}{2^4} = \frac{6}{16}$

P	2		N
0.345	2	0.69	0
0.69	2	1.38	1
0.38	2	0.76	0
0.76	2	1.52	1
0.52	2	1.04	1
0.04	2	0.08	0
0.08	2	0.16	0

$0.345 = 0b0.0101100 = \frac{0b0101100}{2^7} = \frac{44}{128} = 0.34375$

Stop when the result of Multiplication 0 or you run out of bits

Algorithm: Convert Decimal Fraction to Binary

- Fraction is expressed as 2 integers: P/Q
  - E.g. 0.23 = 23/100
  - If P/Q * 2 > 1 then N = 1 is the same as: If P*2 > Q then N = 1
- Algorithm:

```
for i = 0 to M
    Multiply P by 2: P= P*2
    If P >= Q
        N[i] = 1
        P = P - Q
    else
        N[i] = 0
```

Code: Convert Decimal Fraction to Binary

```
        LDI Zh, high(2*P)
        LDI Zl, low(2*P)
        LPM R16, Z

        LDI Zh, high(2*Q)
        LDI Zl, low(2*Q)
        LPM R17, Z

        LDI Xh, high(N)
        LDI Xl, low(N)
        LDI R20, 1          ;TO WRITE 1
        LDI R21, 0          ;TO WRITE 0
        LDI R18, 2
        LDI R19, LEN
L1:     MUL R16, R18        ;R1:R0
        TST R1
        BREQ CHKR0
        BRSH N1
CHKR0:  CP R0, R17          ;>Q ?
        BRSH N1
        ST X+, R21
        RJMP NEXT
N1:     ST X+, R20

        SUB R0, R17         ;SUBTRACT Q
        SBC R1, R22
NEXT:   MOV R16, R0         ;P = 2*P
        DEC R19
        TST R19
        BRNE L1

P:      .DB 23
Q:      .DB 100
        .DSEG
.EQU LEN = 16
N:      .BYTE LEN
```

IEEE-754

- Format: $\pm 1.a_1 a_2 a_3 \ldots a_m \times 2^{exp}$
- Scientific Notation
  - Convert the number into scientific notation

Sign		Exponent	Fraction

- Sign bit: 0,1 (pos, neg)
  - Doesn’t follow 2’s compement
- Exponent
  - Signed binary
  - Stored exponent = actual exponent + “bias”
- Fraction: of the scientific notation
- Integer “1” of scientific notation is not stored

IEEE 754 - float

- 32-bit, aka single precision
- Sign: 1 bit. 1 = negative, 0 = positive
- Fraction: 23 bits.
- Exponent: 8 bits.
  - Exponent stored = actual exponent+127.
- Example: 19.375d = 10011.011b = 1.0011011×2^4
- Example: -591
- Example: 0.3
- Example: -591.3

IEEE 754 - double

- Double precision (long real): 64 bits
- Sign: 1 bit. 1 = negative, 0 = positive
- Fraction: 52 bits.
- Exponent: 11 bits
  - Exponent stored = actual exponent+1023.
- Example: 19.375d = 10011.011b = 1.0011011×2^4
- Example: -591
- Example: 0.3
- Example: -591.3

Coding Exercise: Determine Sci Not

- Given unsigned byte N, P that makeup a real number as such: N.P. Determine the scientific notation form 1.F exponent (E), such that F and E are also bytes.
- Algorithm to determine E:
  1. If N > 0:

     Find the number of significant digits in N

     Count the the number of leading 0 (left side)

     Significant digits = 8 - count

     E = sig dig - 1 = 8 - count - 1
  2. If N = 0

     Count the number of leading 0 in P

     E = -count - 1

Coding Exercise: continued

- Algorithm to determine E:
  1. If N > 0: ignore all leading 0 from N

     shift N to left so all the leading 0 are gone,

     then shift left again to get rid of most sig

     count the number of shifts

     shift P to the right (8 - count)

     F = N+P

  1. If N = 0: ignore all leading 0s from P

     Shift P to the left so all leading 0s are gone,

     then shift left again to get rid of the most sig

```
          ;R16 = N
          ;R17 = P

          CLR R18              ;SIG DIGIT COUNT - 1

          TST R16
          BREQ CASE2N
CASE1N:   LSL R16, 1           ;N<<1
          INC R18
          BRCC CASE1N
          ST X, R18            ;mem[X] = EXPONENT
                               ;EXPONENT = COUNT-1

CASE1P:   INC R18              ;COUNT
          LSR R17
          DEC R18
          TST R18
          BRNE CASE1P
          ADD R16, R17         ;F=N+P
          ST Y, R16            ;mem[Y] = F
          RJMP DONE

CASE2N:   LSL R17,1
          INC R18
          BRCC CASE2N
          NEG R18
          ST X, R18            ;mem[x] = E

CASE2P:   ST Y,R17             ;mem[y] = F
```