The floating point standard

- Single Precision

- Value of bits stored in representation is:
  - If e=255 and f /= 0, then v is NaN regardless of s
  - If e=255 and f = 0, then v = $(-1)^s\ \infty$
  - If 0 < e < 255, then v = $(-1)^s\ 2^{e-127}\ (1.f)$ – normalized number
  - If e = 0 and f /= 0, the v = $(-1)^s\ 2^{-126}\ (0.f)$
    - Denormalized numbers – allow for graceful underflow
  - If e = 0 and f = 0 the v = $(-1)^s\ 0$ (zero)

Ref: Bryant and O'Hallaron, Computer Systems: A Programmer's , Third Edition

Single precision example-Covert 100_{10}

- Step 1 – convert to binary - 0110 0100
  - Binary representation form of 1.xxx have
  - 0110 0100 = 1.100100 x 2^6

Step 2

1.1001 x 2^6 is binary for 100

Thus the exponent is a 6

Biased exponent will be 6+127=133 = 1000 0101

Sign will be a 0 for positive

Stored fractional part f will be 1001

Thus we have

s e f

0 <u>100 0 010 1</u> 1 00 1000….

4 2 C 8 0 0 0 0 in hexadecimal

$42C8 0000 is representation for 100

- **Another example:**
  - Representation for -175

- **Convert $C32F 0000 into decimal**

- **Extract components from**
- **1100 0011 0010 1111**
- **S = 1**
- **Exponent = 1000 0110 = 128+4+2 = 134**
- **unbias   134 – 127 =7**
- **f = 0101111 so mantissa is 1.0101111**
- **Adjust by exponent 1010 1111 (move binary pt 7 places)**
- **Or 128+32+15 = 175**
- **Sign is negative so -175**

- **Convert $41C8 0000 to decimal**

Arithmetic with floating point numbers

- Add op1 \$42C8 0000 and op2 \$41C8 0000
- First divide into component parts
  - Op1 \$42C8 0000 =0100 0010 1100 1000 0000 ….
    - S = 0
    - E = 1000 0101 = 133 – 127 = 6
    - M_{op1} = 1.10010000…
  - Op2 \$41C8 0000 =0100 0001 1100 1000 0000 ….
    - S = 0
    - E = 1000 0011 = 131 – 127 = 4
    - M_{op2} = 1.10010000…

Arithmetic with floating point numbers

- Add op1 \$42C8 0000 and op2 \$41C8 0000
- First divide into component parts
  - Op1 \$42C8 0000 =0100 0010 1100 1000 0000 ….
    - S = 0
    - E = 1000 0101 = 133 – 127 = 6
    - M_{op1} = 1.10010000…
  - Op2 \$41C8 0000 =0100 0001 1100 1000 0000 ….
    - S = 0
    - E = 1000 0011 = 131 – 127 = 4
    - M_{op2} = 1.10010000…

Now add the mantissas

- But first align the mantissas
  - Op1 1.1001000….
  - Op2 1.1001000…. Which is the smaller number and needs to be aligned
  - Exponent difference between op1 and op2 is 2
  - So shift op2 by 2 binary places or
  - Op2 becomes 0.0110010000…

Add

- Add op1 mantissa with the aligned op2 mantissa
  - `1.1001000000`…
  - `0.0110010000`…
  - `1.1111010000`
- Result exponent is 6
- Value is 1111101 or 64+32+16+8+4+1=125
- **Values added were 100 and 25**

Constructing Result Value

- Sign 0
- Exponent 6   E = 1000 0101 = 133 – 127 = 6
- Mantissa of Result `1.1111010000`
- Fractional Part   1111010000….

- Constructed Value
  - 0 100 0010 1 111 1010 0000 0000 0000 0000
  - $4 2 F A 0 0 0 0   (125)

Floating point representation of 125

- Positive so s is 0
- Exponent is 6 + 127 = 133 = 1000 0101
- Fractional part from mantissa of
  - 1.111101   or 111101
- Constructed value
  - 0 1000 0101 111101 00000000000000000
  - $42FA 0000