Floating Point Representation

- **Numerical Form:**

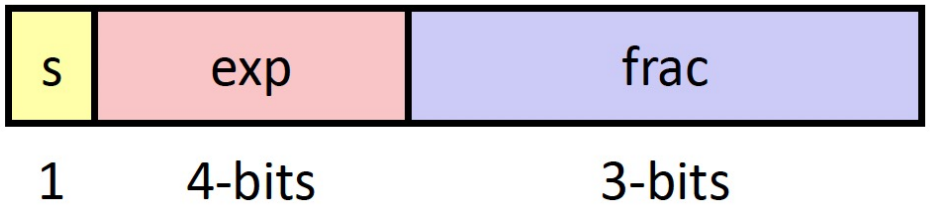
$$(-1)^s\, M\ \ 2^E$$

  - **Sign bit** ***s*** determines whether number is negative or positive
  - **Significand** ***M*** normally a fractional value in range [1.0,2.0).
  - **Exponent** ***E*** weights value by power of two

- **Encoding**
  - MSB s is sign bit ***s***
  - exp field encodes ***E*** (but is not equal to E)
  - frac field encodes ***M*** (but is not equal to M)

s	exp	frac

Tiny Floating Point Example



- **8-bit Floating Point Representation**
  - the sign bit is in the most significant bit
  - the next four bits are the exponent, with a bias of 7
  - the last three bits are the `frac`

Dynamic Range (Positive Only)

$v = (-1)^s \, M \, 2^E$

n: E = Exp – Bias

d: E = 1 – Bias

	s	exp	frac	E	Value	
	0	0000	000	-6	0	
	0	0000	001	-6	1/8*1/64 = 1/512	closest to zero
Denormalized numbers	0	0000	010	-6	2/8*1/64 = 2/512	
	…					
	0	0000	110	-6	6/8*1/64 = 6/512	
	0	0000	111	-6	7/8*1/64 = 7/512	largest denorm
	0	0001	000	-6	8/8*1/64 = 8/512	smallest norm
	0	0001	001	-6	9/8*1/64 = 9/512	
	…					
	0	0110	110	-1	14/8*1/2 = 14/16	
	0	0110	111	-1	15/8*1/2 = 15/16	closest to 1 below
Normalized numbers	0	0111	000	0	8/8*1 = 1	
	0	0111	001	0	9/8*1 = 9/8	closest to 1 above
	0	0111	010	0	10/8*1 = 10/8	
	…					
	0	1110	110	7	14/8*128 = 224	
	0	1110	111	7	15/8*128 = 240	largest norm
	0	1111	000	n/a	inf	

```c
union {
  int i;
  float f;
  unsigned int u;
  char s[4];
} payload;

int main( int argc, const char* argv[] )
{
  int i;
  unsigned int j;
  float bigf;
  float f;

  payload.f=

  printf("Contents are: 0x%x 0x%x 0x%x 0x%x\n", payload.s[0]&0xff,payload.s[1]&0xff,payload.s[2]&0xff,payload.s[3]&0xff);
  printf("Convert 0x%x %d %f %u %c\n", payload.i, payload.i, payload.f, payload.u, payload.s[3]);

  bigf=8388608;
  f=.25;

  printf("Check it: %f %f\n", (bigf+f)-bigf, f);

}
```

```c
union {
  int i;
  float f;
  unsigned int u;
  char s[4];
} payload;

int main( int argc, const char* argv[] )
{
  int i;
  unsigned int j;
  float bigf;
  float f;

  payload.f=

  printf("Contents are: 0x%x 0x%x 0x%x 0x%x\n", payload.s[0]&0xff,payload.s[1]&0xff,payload.s[2]&0xff,payload.s[3]&0xff);
  printf("Convert 0x%x %d %f %u %c\n", payload.i, payload.i, payload.f, payload.u, payload.s[3]);

  bigf=8388608;
  f=.25;

  printf("Check it: %f %f\n", (bigf+f)-bigf, f);

}
```

```
Contents are: 0x0 0x0 0x8 0xc0
```

0b 1 1000000 0 1.0001000 00000000 00000000

```c
union {
  int i;
  float f;
  unsigned int u;
  char s[4];
} payload;

int main( int argc, const char* argv[] )
{
  int i;
  unsigned int j;
  float bigf;
  float f;

  payload.f=-2.125;

  printf("Contents are: 0x%x 0x%x 0x%x 0x%x\n", payload.s[0]&0xff,payload.s[1]&0xff,payload.s[2]&0xff,payload.s[3]&0xff);
  printf("Convert 0x%x %d %f %u %c\n", payload.i, payload.i, payload.f, payload.u, payload.s[3]);

  bigf=8388608;
  f=.25;

  printf("Check it: %f %f\n", (bigf+f)-bigf, f);

}
```

```
Contents are: 0x0 0x0 0x8 0xc0
Convert 0xc0080000 -1073217536 -2.125000 3221749760 ?
Check it: 0.000000 0.250000
```