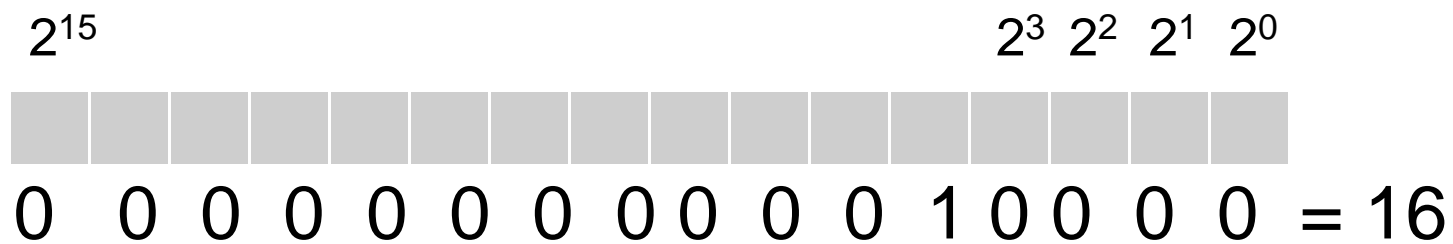

EECE 218

Microcontrollers

Fixedpoint Arithmetic

Signed/unsigned 16 bit integers

- Unsigned numbers



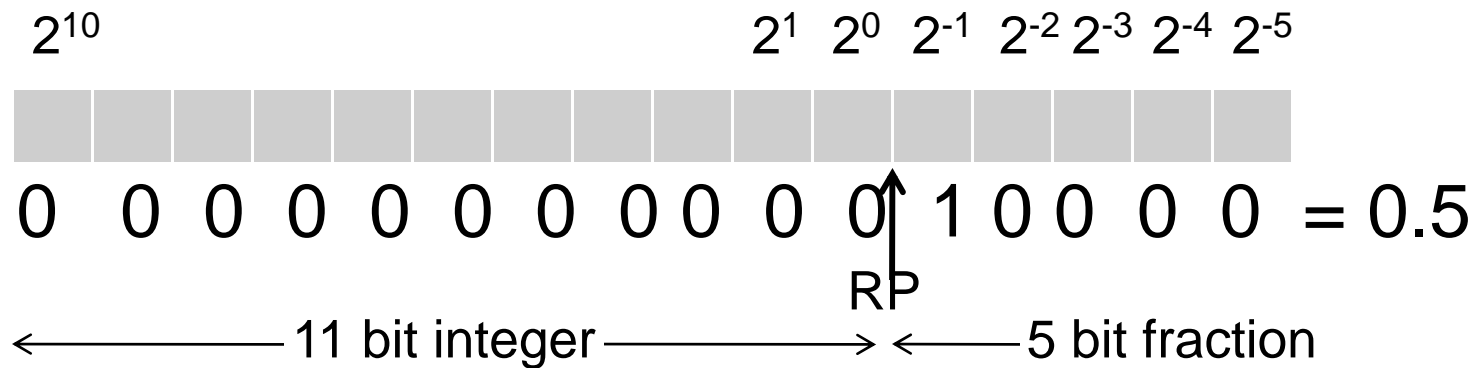
- Signed



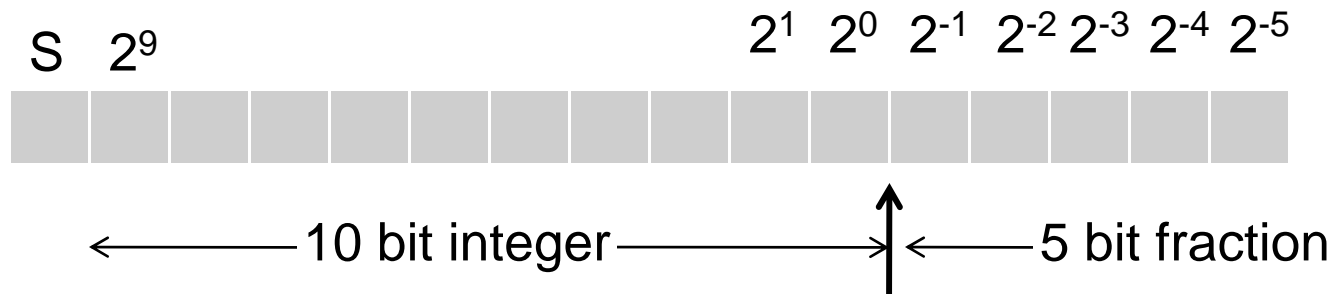
Similar to: sign + 15 bit integer

Radix point at a different place

- Unsigned 16 bit numbers (*)



- Signed 16 bit numbers (**)



Idea

- Programmer DEFINES the position of the radix point and writes code accordingly

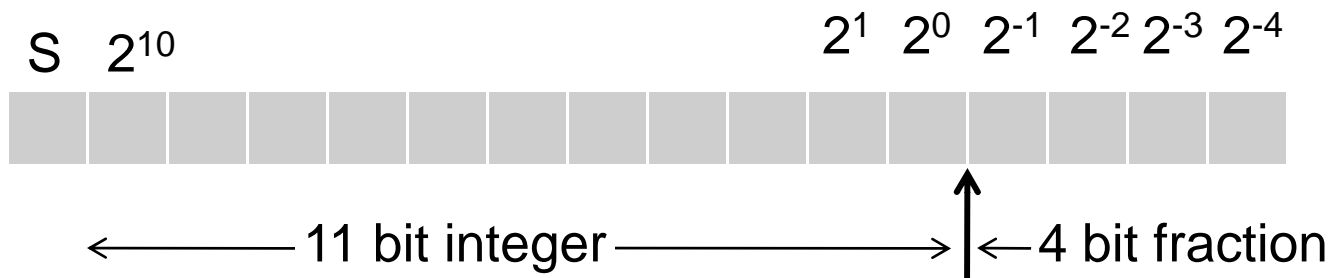
Example above:

(*) can represent 0.0 .. 2047.97865

(**) can represent -1024.0 .. 1023.96875
with 1/32 resolution

Example

- Signed, 11-bit integer/4-bit fraction



Range: -2048.0 ... 2047.xxx

Usable for representing temperatures in Fahrenheit over a large range (0 Kelvin = -459.67), with 1/16 resolution.

4 bit fraction means: 'Scaled by 2^{-4} '

Example: 11528_{10} = represents 720.5 F^o

Notation

Fixedpoint number = <mantissa>S<exponent>

Example:

5S-3 means $5 \cdot 2^{-3} = 0.625$

31S-8 means $31 \cdot 2^{-8} = 0.1211$

Notation from now:

INT(x): take the integer portion of x (no rounding!)

To scale numbers

What is the Mant/Exp for a number x , using the maximum precision possible, on 16 bits.

- Unsigned numbers: $0.0 < x \leq 65,535.0$

$$\text{Exp} = -\text{INT}(\log(65535/x) / \log 2)$$

$$\text{Mant} = \text{INT}(2^{-\text{Exp}} * x + 0.5)$$

When $X = 0$, $\text{Exp} = \text{Mant} = 0$

Example: $x = 1.2345$

$$\text{Exp} = -\text{INT}(\log 53086.2694 / \log 2) = -15$$

$$\text{Mant} = \text{INT}(2^{15} * 1.2345 + 0.5) = 40452$$

Result: $x = \underline{40452S-15}$

To scale numbers

- Unsigned numbers: $x > 65,535.0$

$$\text{Exp} = \text{INT} (\log (x/65535) / \log 2) + 1$$

$$\text{Mant} = x / (2^{\text{Exp}})$$

Example: $x = 107573$

$$\text{Exp} = \text{INT} (\log (107573/65535)/\log 2)+1 = 1$$

$$\text{Mant} = 107573/2^1 = 53786$$

Result: $x = \underline{53786S1}$

Note: We lose resolution!

To scale numbers

- Signed numbers:

$$-32,767.0 \leq x \leq +32,767 \text{ (except } x=0)$$

$$\text{Exp} = - \text{INT} (\log (32767/\text{abs}(x)) / \log 2)$$

$$\text{Mant} = 2^{-\text{Exp}} * x$$

- Signed numbers:

$$x < -32,767.0 \text{ or } x > 32,767.0$$

$$\text{Exp} = \text{INT} (\log (\text{abs}(x)/32767) / \log 2) + 1$$

$$\text{Mant} = x / (2^{\text{Exp}})$$

Operations

- Addition/subtraction: exponents must be the same (i.e. adjust numbers before operation)

Problem: $0.99 + 0.99 = 1.98$

Fixed point:

$32440S-15 + 32440S-15 = 64880S-15$

\Rightarrow Overflow! (> 32767)

Need to adjust first:

$16220S-14 + 16220S-14 = 32440S-14$

Operations

- Multiplication: (multiply mantissas, add exp-s)

$$0.6250 * 0.1211 = 0.075688$$

$$20580S-15 * 31745S-18 = 650137600S-33 \Rightarrow ???$$

Unsigned: 2 16 bit numbers yield = 32 bit result

Signed: 2 15 bit numbers yield = 30 bit result

→ The results must be scaled back to 16/15 bits!

→ Division by 65536S-16 for unsigned

→ Shift mantissa right by 16 bits

→ Division by 32768S-15 for signed

→ Shift mantissa right by 15 bits

Operations

- Division: (divide mantissas, subtract exp-s)
 $0.2345 / -10.987 = -0.021343$
 $30736S-17 / -22501S-11 = -1S-6 \rightarrow -0.015625 ???$
Problem: $q = -1$, $rem = 8235$ (too big!)
Trick: scale dividend by $32768S-15$ (i.e. 1)
 $(30736S-17 * 32768S-15) / -22501S-11 =$
 $(1007157248S-32) / -22501S-11 =$
 $-44760S-21 = -22380S-20 (= -0.21343)$
Last step: result needs to be scaled to 15 bits!
- Compare: exponents must be the same.

Using fixed point arithmetic

- Trick: we use the built-in 16/32 fixed point instructions!
- (1) Calculate the circumference of a circle whose diameter is between 1.22 and 20.88 inches.

$$c = \text{PI} * d; \text{PI} = 51472\text{S}-14$$

d: 5 bits for mant, 11 bits for fraction → S-11

We use C code for description

short = 16-bit signed integer; ushort = 16-bit unsigned integer

long = 32-bit signed integer; ulong = 32-bit unsigned integer

short * short → long, 1234L means (long)1234

Using fixed point arithmetic

```
ushort circ (ushort diam)          /* S-11 */
{
    ushort x;
    x = (ushort) ((51742L * diam) >> 16);
    return x;
}
```

Multiplying two 16 bit numbers yields a 32 bit result. The resultant mantissa should be adjusted by dividing it by 65536 (right shift 16 times).

Using fixed point arithmetic

The exponent of the result:

PI has S-14, diam has S -11 → S-25

followed by 16 bit right shift (add +16 to exp)

→ S-9 is the result's exponent.

Checks:

min 1.22 → 2498S-11 → 1961S-9 (3.83)

max 20.8 → 42598S-11 → 33456S-9 (65.34..)

Using fixed point arithmetic

(2) Calculate volume of a cylinder

$$\text{vol} = (\text{PI} * \text{diam}^2 * \text{len}) / 4$$

		mant	exp	scale
len	9..24	5 bits	11 bits	S-11
diam	1..12	4 bits	12 bits	S-12
PI	51742S-14			

Using fixed point arithmetic

```
ushort vol (ushort len, ushort diam) /* S-11,S-12 */ {
    ushort x;
    x = (ushort) ((51472 * diam) >> 16);
    /* (S-14)*(S-12) + 16 → S-10 */
    x = (ushort) ((x * diam) >> 16);
    /* (S-10)*(S-12) + 16 → S-6 */
    x = (ushort) ((x * len) >> 16);
    /* (S-6)*(S-11) + 16 → S-1 */
    return x;
    /* S-3 --- Avoid div by 4, just change scale ! */
}
```

Using fixed point arithmetic

(3) Fahrenheit / Celsius converter

$$C = ((F - 32) * 5) / 9$$

Range: -40F .. + 250F

→ Signed 8 bit mantissa, 7 bits fraction → S-7

$$32F = 4096S-7, 5/9 = 18204S-15$$

```
short F2C(short F) {          /* S-7 */
    return (short) (((F - (short)4096) * 18204) >> 15);
    /* (S-7) * (S-15) + 15 → S-7 */
}
```

Using fixed point arithmetic

(4) Celsius / Fahrenheit converter

$$F = (C * 9/5) + 32$$

$$9/5 = 29491S-14$$

```
short C2F(short c) {          /* S-7 */
    short x;
    x = (short) ((c * 29491) >> 14); /* Not to overflow */
    /* (S-7) * (S-14) + 14 → S-7 */
    return (x + 4096)          /* (S-7) */
}
```