Computer Architecture and Organization

(Number) representation

Lecture 3

Learning objective:

You understand the different number representations

Topics of this lecture

- **Character representation**
- **Number representation**

Representation of Information

- **Text**
  - ASCII; 7-bits→128 representations (often extend to 8-bits)
  - Unicode-16; 16 bits →65536 representations; e.g. ø **and** æ
- **Numbers**
  - BCD
  - Positive numbers only (unsigned)
  - Positive and negative numbers

    1-complement, 2-complement (signed), sign magnitude, excess
  - fixed point numbers
  - floating point numbers
- **A computer program (instructions)**

BCD

- **Binary-coded decimal**
- **Each decimal digit is represented with 4 bits.**
- **$(294)_{10}$ = $(0010\ 1001\ 0100)_{BCD}$**
- **Only 10 of the 16 bit patterns used!**
- **Arithmetic operations not easy**
- **Exact representation of fixed point numbers.**

Numbers in base 8, 10 and 16

- **Base 8 (octal); digits 0..7**
  463_8 → decimal value $4\times8^2 + 6\times8^1 + 3\times8^0 = 307$

- **Base 10 (decimal); digits 0..9**
  235_{10} = decimal value $2\times10^2 + 3\times10^1 + 5\times10^0 = 235$

- **Base 16 (hexadecimal); digits 0..9, A, B, C, D, E and F**
  $4A3_{16}$ → decimal value $4\times16^2 + 10\times16^1 + 3\times16^0 = 1187$

Decimal to binary

213_{10}		$= ??_2$	
213	- 128	= 85	(2^7)
85	- 64	= 21	(2^6)
21	- 16	= 5	(2^4)
5	- 4	= 1	(2^2)
1	- 1	= 0	(2^0)

$213_{10} = 2^7 + 2^6 + 2^4 + 2^2 + 2^0$
$= 11010101_2$

0.8_{10}		$= ??_2$	
0.8	- 0.5	= 0.3	(2^{-1})
0.3	- 0.25	= 0.05	(2^{-2})
0.05	- 0.03125	= 0.01875	(2^{-5})
0.01875 ..			

$0.8_{10} = 2^{-1} + 2^{-2} + 2^{-5} + \ldots$
$= .11001\ldots_2$

Alternative

213 ÷ 2	= 106	remainder 1	LSB (Least Significant Bit)
106 ÷ 2	= 53	remainder 0	
53 ÷ 2	= 26	remainder 1	
26 ÷ 2	= 13	remainder 0	
13 ÷ 2	= 6	remainder 1	
6 ÷ 2	= 3	remainder 0	
3 ÷ 2	= 1	remainder 1	
1 ÷ 2	= 0	remainder 1	MSB (Most Significant Bit)

$213_{10} = 11010101_2$

0.8 * 2	= 1.6	bit 1, remainder 1.6-1=0.6	(MSB; is bit with weight 2^{-1})
0.6 * 2	= 1.2	bit 1, remainder 1.2-1=0.2	
0.2 * 2	= 0.4	bit 0	
0.4 * 2	= 0.8	bit 0	
0.8 * 2	= 1.6	bit 1, remainder 1.6-1=0.6	

….

$0.8_{10} = .11001100110011.._2$

Negative number representations

- Signed magnitude
- 1's complement (or one's complement)
- 2's complement (or two's complement)
- Excess (or biased)

Signed magnitude/sign magnitude

- N bits
- Left most bit is sign bit
  - '1' is negative
  - '0' is positive
- Other N-1 bits represent the absolute value
- Disadvantage: 2 representations for decimal 0
- This notation is used as part of the floating point representation

$$V = (-1)^{b_{N-1}} \times \sum_{i=0}^{N-2} (b_i \times 2^i)$$

1's complement

- N-bits
- Left most bit is sign bit
  - '1' is negative
  - '0' is positive
- If sign bit is '0' decimal value is $V = \sum_{i=0}^{N-1} (b_i \times 2^i)$
- if sign bit is '1' decimal value is $V = -\sum_{i=0}^{N-1} (\bar{b}_i \times 2^i)$
- Seldom used
- Disadvantages:
  - 2 representations for decimal 0
  - Arithmetic operations not ease, i.e. "end around carry" (page 67)

2's complement (often called *signed*)

Note: Signed magnitude ≠ Signed

- **N-bits**
- **Left most bit is sign bit**
  - **'1' is negative**
  - **'0' is positive**
- **Represented decimal value** $V = -b_{N-1} \times 2^{N-1} + \sum_{i=0}^{N-2} (b_i \times 2^i)$
- **Disadvantage:**
  - **The number of negative values is one larger then the number of positive values.**
- **Advantage:**
  - **Same hardware can be used for addition of unsigned en 2's complement numbers. (next lecture!)**

Excess (or biased)

V = e - M

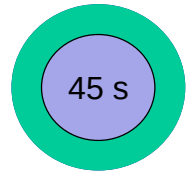
with e is the decimal value of N-bit pattern interpreted as unsigned and V is the represented decimal value in Excess representation

- range of e: 0 to 2^N-1
- range of the represented value V: -M to 2^N-M-1
- The value of M is often 2^{N-1}
- Disadvantage: a correction is needed if you add numbers
  e1 + e2 = (V1 + M) + (V2 + M) = (V1 + V2) + M + M
- This notation is used in the floating point representation

Question (90 seconds)

What is the bit pattern of the decimal value -35 using (all representations use 8 bits)

- **Sign magnitude**
- **1's complement**
- **2's complement**
- **Excess 50**
- **Excess 128**



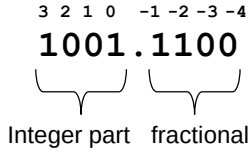
What are the bit patterns in hex?

Topics of this lecture

- **Review number representation**
- **Fixed point numbers ⇦**
  - **Unsigned fixed point**
  - **Signed fixed point**
- **Floating point numbers**

fixed point numbers

- **The point is on a fixed position, P, from the right**



- **Decimal value of unsigned fixed point number**

  $2^3 + 2^0 + 2^{-1} + 2^{-2}$

  or

  $(2^7 + 2^4 + 2^3 + 2^2)/2^4$

$$V = \sum_{i=0}^{N-1} (b_i \times 2^i) \times 2^{-P}$$

N is number of bits

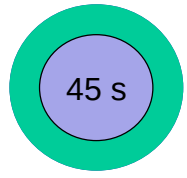
- **Decimal value of signed fixed point number**

  $-2^3 + 2^0 + 2^{-1} + 2^{-2}$

$$V = -b_{N-1} \times 2^{(N-1)-P} + \sum_{i=0}^{N-2} (b_i \times 2^{i-P})$$

Question (90 seconds)

- **Signed fixed point, N=8, P=3**
  - **What is the decimal value of 01010101 ?**
  - **What is the bit pattern for decimal -12.75 ?**
  - **What is the bit pattern for decimal 10.4 ?**



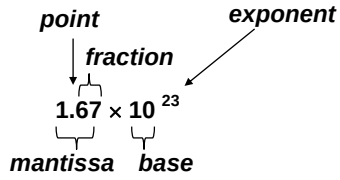
Rounding/Truncation (explained for base 10)

- **Rounding** To round to k decimal places means that we must discard the $(k+1)^{th}$ plus decimals. The rules for rounding are:
  - Rule (a) If the discarded number is less than $\frac{1}{2}\ 10^{-k}$, so that the first discarded digit is less than 5, leave the k^{th} decimal unchanged. This is rounding down.
  - Rule (b) If the discarded number is greater than $\frac{1}{2}\ 10^{-k}$, so that the first discarded digit is greater than 5 add one to the k^{th} decimal. This is rounding up.
  - Rule (c) If the discarded number is exactly $\frac{1}{2}\ 10^{-k}$ then round to the nearest even decimal !
- **Truncation**: skip the $(k+1)^{th}$ plus decimals (=rounding down)

Topics of this lecture

- Review number representation
- Fixed point numbers
- Floating point numbers ⇦
  - Only base 2 for this course

Floating point



- **Floating point numbers allow very large and very small numbers to be represented using only a few digits, at the expense of precision.**
- **The precision is primarily determined by the number of digits in the mantissa**
- **The range is primarily determined by the number of digits in the exponent.**

Normalization

- The base 10 number 254 can be represented in floating point form as 254×10^0, or equivalently as:

  25.4×10^1 or 2.54×10^2 or

  $.254 \times 10^3$ or $.0254 \times 10^4$

- Floating point numbers are usually *normalized*, in which the radix point is located in only one position for a given number.
- Usually the normalized representation places the radix point immediately to the left, or to the right, of the leftmost, nonzero digit, 0.254×10^3 respectively 2.54×10^2

Floating Point: Normalization with base 2

- NORMALIZATION: only one valid representation (different normalization schemes are possible)
  - Remove the leading zero's.
  - The point is on the right (or left) of the first digit not equal 0 of the mantissa (also called significant).

In binary notation the normalized mantissa is:
1.xxxx.. with x ∈{0,1}
or
0.1xxxx.. with x ∈{0,1}

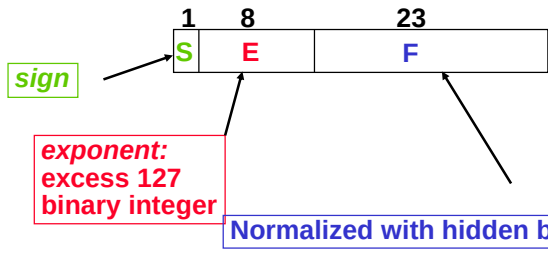
- Since the first bit is always '1' for binary this bit is not stored: hidden bit !

Exceptions

- How to represent 0 ?
- How to represent infinity ?

Floating point systems often have special cases to take care of these kind of exceptions.
E.g. in the IEEE FP system the number is not normalized anymore if the exponent field is all 0 or all 1.

Floating-Point, an example: the IEEE standard 754 (single precision (32 bits))



actual exponent is
E = e - 127

0 < e < 255

$N = (-1)^S \times 2^{e-127} \times (1+F)$

Hidden bit, not stored in the 23 bits.

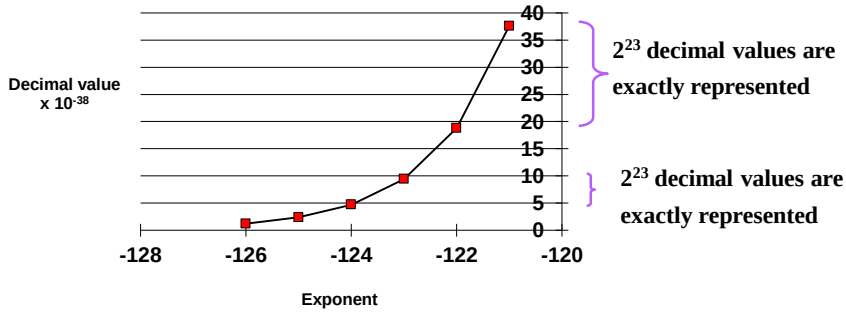
Exceptions if exponent field has pattern:
00000000 or 11111111

IEEE 754, Normalized numbers, the facts

	Bit pattern	Decimal value (V)
Mantissa minimal (M_{min}):	1.0000..0	1
Mantissa Maximal (M_{max}):	1.1111..1	$2-2^{-23} = 1.99999988$
Exponent Minimal (E_{min}):	00000001	-126
Exponent Maximum (E_{max}):	1111110	127

- Minimum positive value that can be represented: (V_{min}): $M_{min} \times 2^{E_{min}} = 1.1755 \times 10^{-38}$
- Maximum positive value that can be represented: (V_{max}): $M_{max} \times 2^{E_{max}} = 3.4028 \times 10^{38}$

IEEE 754, Normalized numbers



Floating Point: trade-off between accuracy and range.

- **Increasing the size of the fraction enhances the accuracy**
- **Increasing the size of the exponent increases the ranges**

Relative error for normalized numbers

- For IEEE format (32 bits; single precision) the distance between 2 consecutive numbers that can be exactly represented is: $\Delta = 2^{-23} \times 2^{e-127}$

- The relative error is approximately the same for all numbers

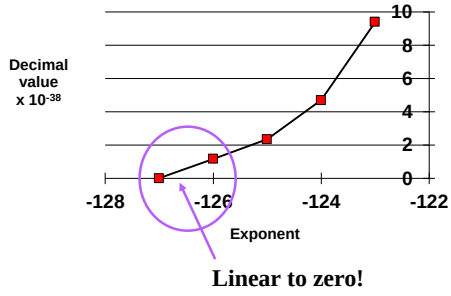
IEEE 754, Special “numbers”

E	*F*	*S*	*V*
-127	0	0	0
-127	0	1	0
-127	≠0	-	Denormalized numbers
128	0	0	+ ∞
128	0	1	- ∞
128	≠0	-	'Not a Number' (NaN)

Denormalized numbers

Try to fill the gap between ‘0’ and smallest normalized number

- smallest positive normalized number: 1.00000 .. 0000 × 2^{-126}
- smallest denormalized positive number: **0.**000000 .. 0001 × 2^{-126} (1.0×2^{-149})



Infinity and NaN

Infinity the operation results in:

- *overflow,* i.e., is larger than the largest number that can be represented.
- *underflow*, i.e., result is number than the smallest absolute value that can be represented

Not a number (NaN) is used to signify invalid operations, such as the multiplication of zero by infinity.

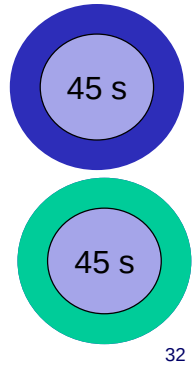
Why EXCESS coding for exponent

Historical reason!

- **Excess coding is used because two floating point numbers can be compared by bit compare**
  - **in Excess code bit pattern 0000 is the smallest and 1111 the largest**
  - **in 2-complement 1111 is not the largest and 0000 is not the smallest**

Exercise (try it yourself)

- Floating point format
  - Sign bit
  - Exponent field: 5 bits
    - Excess 20 for normalized numbers
    - All fields zero: no normalization (not used in this exercise)
  - Fraction field 6 bits (excl. hidden bit);
    - point right from hidden bit
- Questions (use powers of 2 in answer)
  - What is the decimal value of
    1 01110 010000 (45 seconds)
  - What is the representation of decimal
    4.75 (90 seconds)
  - What is the representation of decimal
    2.3



Next lecture

- What did you learn:
  - You understand the different number representations
- Next lecture:
  - Arithmetic (chapter 3)