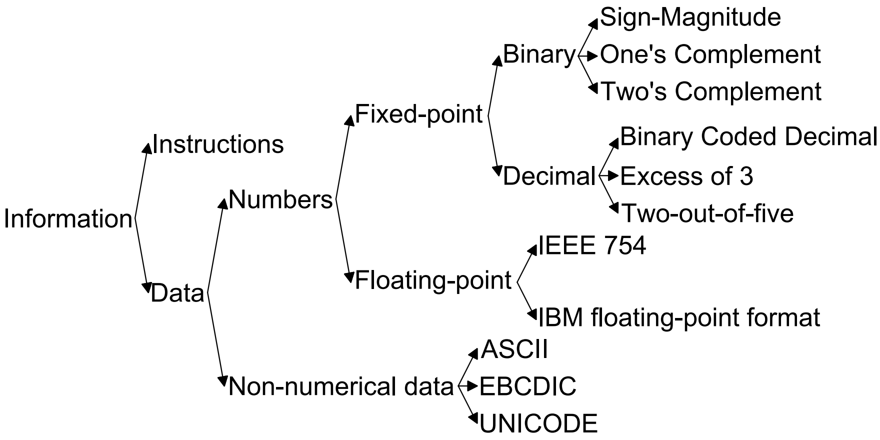
Computer Architecture

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

October 5, 2021

Chap. 1 Representation of numbers in computer systems

Information tree:



Bit: binary digit

- byte
- words

Codes for non-numerical data

- American Standard Code for Information Interchange (ASCII)
- Extended Binary Coded Decimal Interchange Code (EBCDIC)
- UNICODE

Fixed-point numbers

- integers
- fractionals

Floating point numbers:

- approximate representation
  - Consider value $1e20$
    - $(3.14 + 1e20) - 1e20 = 0$
    - $3.14 + (1e20 - 1e20) = 3.14$
  - $\Rightarrow$ Floating-point addition: not associative!

Number X represented in radix $\boldsymbol{r}$:

- $X = x_{n-1}x_{n-2}\cdots x_1x_0.x_{-1}x_{-2}\cdots x_{-m}$
  - integer part: $x_{n-1}x_{n-2}\cdots x_1x_0$
  - fractional part: $x_{-1}x_{-2}\cdots x_{-m}$
  - Most Significant Bit (MSB): x_{n-1}
  - Least Significant Bit (LSB): x_{-m}

Value of X represented in radix $\boldsymbol{r}$:

- $X = \sum_{j=-m}^{n-1} x_j * r^j$, $0 \leq x_j < r$
  - r^j: weight of digit x_j
  - **positional** representation

When radix $r = 2 \rightarrow$ binary representation system.

Example:

$$
\begin{array}{rccccccc}
103_{(10)} = & 1 & 1 & 0 & 0 & 1 & 1 & 1\,._{(2)} \\
\textit{weights} & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
68_{(10)} = & 1 & 0 & 0 & 0 & 1 & 0 & 0\,._{(2)}
\end{array}
$$

Position of the binary point:

- convention for representing integers and fractions
  - avoids encoding point's position in the number representation
  - save more bits for precision

For integers, the binary point is situated to the right of the LSB.

- $X = x_{n-1}x_{n-2}\cdots x_1x_0. = \sum_{i=0}^{n-1} x_i * 2^i$, for X on n bits

For fractionals, the binary point is situated to the left of the MSB.

- $X = .x_{n-1}x_{n-2}\cdots x_1x_0 = \sum_{i=0}^{n-1} x_i * 2^{i-n}$, for X on n bits

Example:

$$\frac{103}{128}{}_{(10)} = .1\ 1\ 0\ 0\ 1\ 1\ 1_{(2)}$$

weights $2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ 2^{-5}\ 2^{-6}\ 2^{-7}$

$$\frac{17}{32}{}_{(10)} = .1\ 0\ 0\ 0\ 1_{(2)}$$

The MSB encodes the sign of the number. Sign convention:

- positives have a sign bit of 0
- negatives have a sign bit of 1

Number X, on n bits, is represented in Sign-Magnitude as:

- $X = x_{n-1}\ \ x_{n-2}x_{n-3}\cdots x_1x_0$, with
  - x_{n-1} being the sign of X
  - $x_{n-2}x_{n-3}\cdots x_1x_0$ being the magnitude of X

Example:
$+103_{10}\ =\ 0\ 1100111_{(SM)}$
$-103_{10}\ =\ 1\ 1100111_{(SM)}$

Range of values:

- largest integer on n bits, in Sign-Magnitude:
  - $MAXINT_{SM}$: $0\,\underbrace{11\cdots111}_{n-1 \text{ bits}}$.
  - value of $MAXINT_{SM} = 2^{n-1} - 1$
- $\Rightarrow$ range of values for n-bit integers: $[1 - 2^{n-1}; 2^{n-1} - 1]$

- largest fractional on n bits, in Sign-Magnitude:
  - $MAXFRA_{SM}$: $0\,.\underbrace{11\cdots111}_{n-1 \text{ bits}}$
  - value of $MAXFRA_{SM} = 1 - 2^{-n+1}$
- $\Rightarrow$ range of values for n-bit fractionals: $[2^{-n+1} - 1; 1 - 2^{-n+1}]$

Precision:

- consider Sign-Magnitude numbers on n bits
- how many decimal digits are required for representing any of the Sign-Magnitude numbers on n bits?
  - $2^{n-1} - 1 = 10^p$, with p being the *precision*
  - it follows that $p = \lceil log_{10}(2^{n-1} - 1) \rceil$
  - thus $p < \lceil log_{10}(2^{n-1}) \rceil =$
  - and ultimately, $p \approx \lceil (n - 1) * 0.3 \rceil =$

Example: consider Sign-Magnitude numbers on 10 bits.

- precision $p \approx \lceil 9 * 0.3 \rceil = \lceil 2.7 \rceil = 3$
- largest integer in Sign-Magnitude on 10 bits is $+511$, which requires 3 decimal digits

Hardware complexity of Sign-Magnitude representation:

- moderate HW complexity
- favours multiplication

Disadvantages:

- there are **two** binary configurations for 0 in Sign-Magnitude
  - $+0: \;\; 0\,00\cdots000$
  - $-0: \;\; 1\,00\cdots000$

Disadvantages:

- Sign-Magnitude addition
  - consider operands $X = 5$ and $Y = 2$, on 4 bits
  - the four possible sign configurations for addition

$$
\begin{array}{rr|l}
X=+5: & 0\,1\,0\,1_{SM} & \\
Y=+2: & 0\,0\,1\,0_{SM} & + \\
\hline
 & 0\,1\,1\,1_{SM} & =+7_{SM}
\end{array}
\qquad
\begin{array}{rr|l}
X=+5: & 0\,1\,0\,1_{SM} & \\
Y=-2: & 1\,0\,1\,0_{SM} & + \\
\hline
 & 1\,1\,1\,1_{SM} & =\cancel{-7_{SM}}
\end{array}
$$

$$
\begin{array}{rr|l}
X=-5: & 1\,1\,0\,1_{SM} & \\
Y=+2: & 0\,0\,1\,0_{SM} & + \\
\hline
 & 1\,1\,1\,1_{SM} & =\cancel{+7_{SM}}
\end{array}
\qquad
\begin{array}{rr|l}
X=-5: & 1\,1\,0\,1_{SM} & \\
Y=-2: & 1\,0\,1\,0_{SM} & + \\
\hline
\cancel{1} & 0\,1\,1\,1_{SM} & =\cancel{+}7_{SM}
\end{array}
$$

The MSB encodes the sign (same convention as for SM).

Number X, on n bits, is represented in One's Complement as:

- $\overline{X} = \begin{cases} 0 \;\; x_{n-2}x_{n-3}\cdots x_1 x_0 & X \geq 0 \\ 1 \;\; \overline{x_{n-2}}\,\overline{x_{n-3}}\cdots \overline{x_1}\,\overline{x_0} & X \leq 0 \end{cases}$, with
  - $\overline{x_i}$ representing the complement of x_i: $\overline{x_i} = 1 - x_i$

Example:

$+103_{10} \;=\; 0\,1100111_{(C1)}$

$-103_{10} \;=\; 1\,0011000_{(C1)}$

1.2.2 - One's Complement (contd.)

Range of values:

- same as for Sign-Magnitude
  - for a given number of bits n, Sign-Magnitude and One's Complement encode the same number of values

Precision:

- same as for Sign-Magnitude
  - since encoding the same number of values, One's Complement has the same precision as Sign-Magnitude

Hardware complexity:

- the HW complexity is somewat higher for One's Complement
  - not favouring anymore multiplication

Disadvantages:

- there are **two** binary configurations for 0 in One's Complement
  - $+0: \quad 0\,00\cdots000$
  - $-0: \quad 1\,11\cdots111$

Disadvantages:

- One's Complement addition
  - consider same operands $X = 5$ and $Y = 2$, on 4 bits
  - the four possible sign configurations for addition

$$
\begin{array}{lr|l}
X=+5: & 0\,1\,0\,1_{C1} & \\
Y=+2: & 0\,0\,1\,0_{C1} & + \\
\hline
 & 0\,1\,1\,1_{C1} & =+7_{C1}
\end{array}
$$

$$
\begin{array}{lr|l}
X=+5: & 0\,1\,0\,1_{C1} & \\
Y=-2: & 1\,1\,0\,1_{C1} & + \\
\hline
\text{end around carry} & 1\;0\,0\,1\,0_{C1} & =\cancel{+2}_{C1} \\
 & 1 & + \\
\hline
 & 0\,0\,1\,1_{C1} & =+3_{C1}
\end{array}
$$

$$
\begin{array}{lr|l}
X=-5: & 1\,0\,1\,0_{C1} & \\
Y=+2: & 0\,0\,1\,0_{C1} & + \\
\hline
 & 1\,1\,0\,0_{C1} & =-3_{C1} \\
 & 1\,0\,1\,1_{SM} & =-3_{SM}
\end{array}
$$

$$
\begin{array}{lr|l}
X=-5: & 1\,0\,1\,0_{C1} & \\
Y=-2: & 1\,1\,0\,1_{C1} & + \\
\hline
 & 1\;0\,1\,1\,1_{C1} & =\cancel{+}7_{C1} \\
 & 1 & + \\
\hline
 & 1\,0\,0\,0_{C1} & =-7_{C1} \\
 & 1\,1\,1\,1_{SM} & =-7_{SM}
\end{array}
$$

Are there disadvantages regarding One's Complement addition?