Chapter 7

Computer Arithmetic

עד עכשיו לא דיברנו על פרטי ה- ALU ו- FPU. רק הזכרנו שהרכיבים האלה קיימים במעבד ומשחקים תפקיד מרכזי. בפרק הזה נעשה חזרה קצרה על חשבון במחשבים, תחום ששיך למערכות ספרתיות.

Slide 2 **Unsigned Integers**

The standard n-bit representation of an unsigned binary number will be written as

$$k_2 = a_{n-1}\ \ a_{n-2}\ \ a_{n-3}\ \ \dots\ \ a_1\ \ a_0 \qquad a_i \in \{0,1\}$$
$$\underbrace{00\ \dots\ 0}_{n} \le k_2 \le \underbrace{11\ \dots\ 1}_{n}$$

We write the decimal value of k_2 as $k_{10} = K(k_2)$

$$k_{10} = K(k_2) = a_{n-1}\times 2^{n-1} + a_{n-2}\times 2^{n-2} + a_{n-3}\times 2^{n-3} + \ \dots\ + a_1\times 2^1 + a_0\times 2^0$$
$$(k_2)_{\min} = 00\ \dots\ 0 \ \Rightarrow\ K\left((k_2)_{\min}\right) = 0$$
$$(k_2)_{\max} = \underbrace{11\ \dots\ 1}_{n} \ \Rightarrow\ (k_2)_{\max} + 1 = 1\underbrace{00\ \dots\ 0}_{n}$$
$$K\left((k_2)_{\max} + 1\right) = \left(1\times 2^n + 0\times 2^{n-2} + \ \dots\ + 0\times 2^0\right) = 2^n$$
$$K\left((k_2)_{\max}\right) = 2^n - 1,$$
$$0 \le k_{10} = K(k_2) \le 2^n - 1$$

A value $k > 2^n - 1$ cannot be represented by in n-bits and may cause overflow in an ALU.

Slides 3 – 5 **Signed Integers**

Signed numbers are written and stored in digital systems in the same way as unsigned numbers but are interpreted according to the 2's-complement method.

The value $k_2 = a_{n-1}\ \ a_{n-2}\ \ a_{n-3}\ \ \dots\ \ a_1\ \ a_0 \qquad a_i \in \{0,1\}$

may represents a non-negative value

$$0 \le k_{10} \le 2^{n-1} - 1$$
$$k_{10} = K(k_2) = K(0\ \ a_{n-2}\ \ a_{n-3}\ \ \dots\ \ a_1\ \ a_0)$$
$$= 0\times 2^{n-1} + a_{n-2}\times 2^{n-2} + a_{n-3}\times 2^{n-3} + \ \dots\ + a_1\times 2^1 + a_0\times 2^0$$

or a negative value (minus the 2's-complement)

$$0 > k_{10} \ge -2^{n-1} \Rightarrow k_{10} = -K\left((k_2)'+1\right)$$
$$k_2 = 11\ \dots\ 1 \Rightarrow k_{10} = -K\left((11\ \dots\ 1)'+1\right) = -K\left((00\ \dots\ 0)+1\right) = -1$$
$$k_2 = 11\ \dots\ 0 \Rightarrow k_{10} = -K\left((11\ \dots\ 0)'+1\right) = -K\left((00\ \dots\ 1)+1\right) = -2$$
$$\vdots$$
$$k_2 = 10\ \dots\ 0 \Rightarrow k_{10} = -K\left((10\ \dots\ 0)'+1\right) = -K\left((01\ \dots\ 1)+1\right) = -2^{n-1}$$

The high-order bit — the sign bit — is 0 for non-negative and 1 for negative values.

The decimal value of a 2's-complement signed number with sign bit = 1 is

$$0 > k_{10} \geq -2^{n-1} \Rightarrow k_{10} = -K\left(\left(k_2\right)'+1\right)$$

where k' = 1's-complement of k (reverse all 1 and 0 values). Therefore, for any value k_2

$$k_2 = a_{n-1}\ \ a_{n-2}\ \ a_{n-3}\ \ \ldots\ \ a_1\ \ a_0$$

$$\left(k_2\right)' = (1-a_{n-1})\ \ (1-a_{n-2})\ \ (1-a_{n-3})\ \ \ldots\ \ (1-a_1)\ \ (1-a_0)$$

$$K\left(k_2+\left(k_2\right)'+1\right) = K\left(a_{n-1}+(1-a_{n-1})\ \ a_{n-2}+(1-a_{n-2})\ \ \ldots\ \ a_0+(1-a_0)\right)+1$$

$$= K\left(\underbrace{11\ldots1}_{n}\right)+1$$

$$= \left(2^n-1\right)+1$$

$$= 2^n \text{ which has no representation in } n \text{ bits}$$

$$= 0$$

$$K\left(\left(k_2\right)'+1\right) = 2^n - K\left(k_2\right) = -K\left(k_2\right) + \text{overflow bit (ignored for signed)}$$

The sum k + (-k) in 2's-complement signed numbers is always 0 as required for arithmetic.

Slide 5 shows a general expression for the decimal value of a signed binary number.

Slides 6 – 8 Multiplication and Division

Binary numbers can be multiplied in long multiplication, for example

```
          0  1  1
      ×   1  0  1
    ---------------
          0  1  1
       0  0  0
    0  1  1
    ---------------
    0  1  1  1  1
```

From this procedure it is possible to generalize to the multiplication algorithm for unsigned numbers shown on slide 6 (in the example above a = 101, b = 001). The basic rule is that if $a_i = 1$ then shift and add b (as seen above in the first and last row) and if $a_i = 0$ then add 0 (as seen above in the middle row).

The algorithm for signed numbers (slide 7) is similar but the sign bit replaces the carry-out bit for unsigned numbers.

The algorithm for unsigned division is shown on slide 8.

Slides 9 – 13 Floating Point Numbers (IEEE-754 Standard)

Floating point numbers are represented by the data structure

s	e	f

where

s represents the sign of the number

e is the exponent — biased power of 2 (8 bits for float or 11 bits for double)

f is the fraction — (23 bits for float or 52 bits for double)

The value N of the represented number is

$$N = (-1)^s \times (1.f)_2 \times 2^{e-127}, \quad \begin{matrix} 1 \le e \le 254 \\ -126 \le e-127 \le 127 \end{matrix} \text{ for float}$$

$$N = (-1)^s \times (1.f)_2 \times 2^{e-1023}, \quad \begin{matrix} 1 \le e \le 2046 \\ -1022 \le e-1023 \le 1023 \end{matrix} \text{ for double}$$

For example, the value of the float

1	10000001	10100000000000000000000

is $(-1)^1$ x 1.10100000000000000000000 x $2^{129-127}$ = -1.101 x 2^2 = -110.10 = -6.50_{10}

Special values for **e** and **f** are

e	f	N	
0	0	0	
0	Not zero	$N = (-1)^s \times 0.f \times 2^{-127}$	$N = (-1)^s \times 0.f \times 2^{-1023}$
		denormal numbers — permit representation of very small numbers	
255	0	$N = (-1)^s \times \infty$	
255	Not zero	NaN (not a number — overflow/underflow)	

Slides 10 – 13 give examples of operations on floats and an indication of the algorithms used in the FPU to perform the operations.

Slides 14 – 15 DLX FP Pipeline

Because of the complexity of FP operations it is convenient to divide them into multiple pipeline stages. In the example, the ADD/SUB unit consists of 4 pipelined stages and the MULT/DIV unit has 7 pipelined stages. In these units, a new FP operation can begin on every clock cycle but it takes 4 or 7 CC for each operation to complete. The execution table for programs with FP operations is shown on slide 15. Notice that RAW data dependencies are more serious for FPU operations.

Slides 16 More Hardware

ALU speed can be limited by propagation delays, such as the time required for results to flow between bits in a ripple adder made of full adders. The **look-ahead adder** uses additional hardware to calculate complex pieces of these calculations in parallel and speed up the final result.