Computer Arithmetic

Unsigned Integers

Binary representation

$k_2 = a_{n-1} \;\; a_{n-2} \;\; a_{n-3} \;\; \ldots \;\; a_1 \;\; a_0 \qquad a_i \in \{0,1\}$

$\underbrace{00 \ldots 0}_{n} \le k_2 \le \underbrace{11 \ldots 1}_{n}$

Decimal value

$K(k_2) =$ decimal value of k_2

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

$(k_2)_{\min} = 00 \ldots 0 \quad \Rightarrow \quad K((k_2)_{\min}) = 0$

$(k_2)_{\max} = \underbrace{11 \ldots 1}_{n} \quad \Rightarrow \quad (k_2)_{\max} + 1 = 1\underbrace{00 \ldots 0}_{n}$

$K((k_2)_{\max} + 1) = (1 \times 2^n + 0 \times 2^{n-2} + \ \ldots \ + 0 \times 2^0) = 2^n$

$K((k_2)_{\max}) = 2^n - 1,$

$0 \le k_{10} = K(k_2) \le 2^n - 1$

$k > 2^n - 1$ cannot be represented in n bits

Signed Integers

Binary representation (two's complement)

$k_2 = a_{n-1} \;\; a_{n-2} \;\; a_{n-3} \;\; \ldots \;\; a_1 \;\; a_0 \qquad a_i \in \{0,1\}$

Non-negative values

$0 \le k_{10} \le 2^{n-1} - 1$

$k_{10} = K(k_2) = K(0 \;\; a_{n-2} \;\; a_{n-3} \;\; \ldots \;\; a_1 \;\; a_0)$

$= 0 \times 2^{n-1} + a_{n-2} \times 2^{n-2} + a_{n-3} \times 2^{n-3} + \ \ldots \ + a_1 \times 2^1 + a_0 \times 2^0$

Negative values

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

$k_2 = 11 \ \ldots 1 \Rightarrow k_{10} = -K((11 \ \ldots 1)'+1) = -K((00 \ \ldots 0)+1) = -1$

$k_2 = 11 \ \ldots 0 \Rightarrow k_{10} = -K((11 \ \ldots 0)'+1) = -K((00 \ \ldots 1)+1) = -2$

$\vdots$

$k_2 = 10 \ \ldots 0 \Rightarrow k_{10} = -K((10 \ \ldots 0)'+1) = -K((01 \ \ldots 1)+1) = -2^{n-1}$

Negative Signed Numbers

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

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

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

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

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

$= (2^n - 1) + 1$

$= 2^n$ which has no representation in n bits

$K((k_2)'+1) = 2^n - K(k_2) = -K(k_2) +$ overflow bit (ignored for signed)

General Formula for Signed Numbers

$$k_{10} = \begin{cases} K(0\ a_{n-2}\ \dots\ a_0) & , \quad a_{n-1} = 0 \\ -K(a_{n-1}\ a_{n-2}\ \dots\ a_0) & , \quad a_{n-1} = 1 \end{cases}$$

$$K\left((k_2)'+1\right) = 2^n - K(k_2)$$

$$= \begin{cases} 0\times 2^{n-1} + a_{n-2}\times 2^{n-2} + \ \dots\ + a_0 & , \quad a_{n-1} = 0 \\ -\left[2^n - \left(1\times 2^{n-1} + a_{n-2}\times 2^{n-2} + \ \dots\ + a_0\right)\right] & , \quad a_{n-1} = 1 \end{cases}$$

$$= \begin{cases} 0\times 2^{n-1} + a_{n-2}\times 2^{n-2} + \ \dots\ + a_0 & , \quad a_{n-1} = 0 \\ -\left[2\times 2^{n-1} - 1\times 2^{n-1}\right] + a_{n-2}\times 2^{n-2} + \ \dots\ + a_0 & , \quad a_{n-1} = 1 \end{cases}$$

$$= \begin{cases} 0\times 2^{n-1} + a_{n-2}\times 2^{n-2} + \ \dots\ + a_0 & , \quad a_{n-1} = 0 \\ -1\times 2^{n-1} + a_{n-2}\times 2^{n-2} + \ \dots\ + a_0 & , \quad a_{n-1} = 1 \end{cases}$$

$$= -a_{n-1}\times 2^{n-1} + a_{n-2}\times 2^{n-2} + \ \dots\ + a_0$$

Multiplying Unsigned Integers

Long Multiplication

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

Algorithm

Operands $a = a_{n-1}\ a_{n-2}\ a_{n-3}\ \cdots\ a_0$

$b = b_{n-1}\ b_{n-2}\ b_{n-3}\ \cdots\ b_0$

Zero temporary register $P = P_{n-1}\ P_{n-2}\ P_{n-3}\ \cdots\ P_0 \leftarrow 0$ $\quad c_{out} \leftarrow 0$

n times {

$c_{out}\ P \leftarrow \begin{cases} P, \text{ if } a_0 = 0 \\ P + b, \text{ if } a_0 = 1 \end{cases}$

shift right bits $c_{out}\left[P_{n-1}\ P_{n-2}\ P_{n-3}\ \cdots\ P_0 \quad a_{n-1}\ a_{n-2}\ a_{n-3}\ \cdots\ a_0\right]$

to form new $\left[P_{n-1}\ P_{n-2}\ P_{n-3}\ \cdots\ P_0 \quad a_{n-1}\ a_{n-2}\ a_{n-3}\ \cdots\ a_0\right]$

}

Result is found in bits $P_{n-1}\ P_{n-2}\ P_{n-3}\ \cdots\ P_0 \quad a_{n-1}\ a_{n-2}\ a_{n-3}\ \cdots\ a_0$

Multiplying Signed Integers

Algorithm

Operands $a = a_{n-1}\ a_{n-2}\ a_{n-3}\ \cdots\ a_0$ (non-negative)

$b = b_{n-1}\ b_{n-2}\ b_{n-3}\ \cdots\ b_0$

Zero temporary register $P = P_{n-1}\ P_{n-2}\ P_{n-3}\ \cdots\ P_0 \leftarrow 0$

n times {

$P \leftarrow \begin{cases} P, \text{ if } a_0 = 0 \\ P + b, \text{ if } a_0 = 1 \end{cases}$

Uses P_{n-1} instead of c_{out}

shift right bits $P_{n-1}\left[P_{n-1}\ P_{n-2}\ P_{n-3}\ \cdots\ P_0 \quad a_{n-1}\ a_{n-2}\ a_{n-3}\ \cdots\ a_0\right]$

to form new $P_{n-1}\ P_{n-2}\ P_{n-3}\ \cdots\ P_0 \quad a_{n-1}\ a_{n-2}\ a_{n-3}\ \cdots\ a_0$

}

Result is found in bits $P_{n-1}\ P_{n-2}\ P_{n-3}\ \cdots\ P_0 \quad a_{n-1}\ a_{n-2}\ a_{n-3}\ \cdots\ a_0$

Dividing Unsigned Integers

Long Division

```
1 1 | 1 1 1 1      1 1 | 1 1 1 1      1 1 | 1 1 1 1      1 1 | 1 1 1 1
      1 1                1 1                1 1                1 1
      0 0                0 0 1              0 0 1              0 0 1
                               0                  0                  0
                             1 1                1 1                1 1
                                                1 1                1 1
                                                                   0
```

Algorithm for a / b

Operands $a = a_{n-1}\ a_{n-2}\ a_{n-3}\ \cdots\ a_0$

$b = b_{n-1}\ b_{n-2}\ b_{n-3}\ \cdots\ b_0$

Zero temporary register $P = P_{n-1}\ P_{n-2}\ P_{n-3}\ \cdots\ P_0 \leftarrow 0$

n times {

shift bits $\left[P\ a\right]\ 0$ left to form $\underbrace{P_{n-1}}_{discard}\left[\underbrace{P_{n-2}\ P_{n-3}\ \cdots\ P_0\ a_{n-1}}_{new\ P} \quad \underbrace{a_{n-2}\ a_{n-3}\ \cdots\ a_0\ 0}_{new\ a}\right]$

if $P \geq b$, $\begin{cases} P \leftarrow P - b \\ a_{n-2}\ \cdots\ a_0\ 0 \leftarrow a_{n-2}\ \cdots\ a_0\ 1 \end{cases}$

}

Remainder is in P and Quotient is in a

Floating Point Numbers (IEEE-754 Standard)

s	e	f

Single precision

s — 1 bit
e — 8 bits
f — 23 bits

$$N = (-1)^s \times (1.f)_2 \times 2^{e-127}, \qquad 1 \le e \le 254, \quad -126 \le e-127 \le 127$$

Double precision

s — 1 bit
e — 11 bits
f — 52 bits

$$N = (-1)^s \times (1.f)_2 \times 2^{e-1023}, \qquad 1 \le e \le 2046, \quad -1022 \le e-1023 \le 1023$$

Special values

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

Floating Point Operations

Addition

$$1.010\times2^{-3} + 1.010\times2^{-4} = 10100\times2^{-7} + 1010\times2^{-7}$$

```
  10100
+ 01010
  11110
```

$$11110\times2^{-7} = 1.1110\times2^{-3}$$

Multiplication

$$1.010\times2^{-3} \times 1.010\times2^{-4} = 10100\times2^{-7} \times 1010\times2^{-7}$$

```
  10100
× 01010
 11001000
```

$$11001000\times2^{-7}\times2^{-7} = 1.1001000\times2^{-7} \rightarrow 1.100\times2^{-7}$$

Floating Point Multiplication

$$x = (-1)^{s_1} \times 1.f_1 \times 2^{e_1-127}$$
$$y = (-1)^{s_2} \times 1.f_2 \times 2^{e_2-127}$$
$$x \times y = (-1)^{s_1+s_2} \times [1.f_1 \times 1.f_2] \times 2^{(e_1+e_2-127)-127} = (-1)^s \times [1.f_1 \times 1.f_2] \times 2^{e-127}$$

Multiply unsigned numbers

$$P_{n-1}\ P_{n-2}\ P_{n-3}\ \cdots\ P_0 \quad a_{n-1}\ a_{n-2}\ a_{n-3}\ \cdots\ a_0 \quad \leftarrow \quad 1.f_1 \times 1.f_2$$

Rounding algorithm

If $P_{n-1} = 0$ assemble

X_{n-1}	X_{n-2}	...	X_1	X_0	r
P_{n-2}	P_{n-3}		P_0	a_{n-1}	a_{n-2}

If $P_{n-1} = 1$ assemble

X_{n-1}	X_{n-2}	...	X_1	X_0	r	
P_{n-1}	P_{n-2}		P_1	P_0	a_{n-1}	$e \leftarrow e+1$

$X_{n-1}\ .\ X_{n-2}\ \ldots X_0 \ + \ r \times (2^{-(n-1)})$ r is rounding bit

Floating Point Addition — 1

$$\left.\begin{aligned} x_1 &= (-1)^{s_1} \times 1.f_1 \times 2^{e_1-127} \\ x_2 &= (-1)^{s_2} \times 1.f_2 \times 2^{e_2-127} \end{aligned}\right\} \quad X = x_1 + x_2 = (-1)^s \times 1.f \times 2^{e-127}$$

1a. If $e_2 > e_1$ then $x_1 \leftrightarrow x_2$
1b. $d = e_1 - e_2$
1c. $e = e_1$
1d. $w = 0$
2. If $s_1 \ne s_2$ then $x_2 \leftarrow (x_2)$" (two's complement) and $w \leftarrow 1$

3a. Construct from x_2

P_{n-1}	.	P_{n-2}	P_{n-3}	...	P_0
x_{n-1}	.	x_{n-2}	x_{n-3}	...	x_0

3b. Shift P right by d places, shifting in w from left

P_{n-1}	.	P_{n-2}	...	P_{n-d}	P_{n-1-d}	...	P_0	g	r
w	.	w	...	w	P_{n-1}	...	P_d	P_{d-1}	P_{d-2}

4. Add $P \leftarrow P + 1.f_1$

Floating Point Addition — 2

$$\left.\begin{aligned} x_1 &= (-1)^{s_1} \times 1.f_1 \times 2^{e_1-127} \\ x_2 &= (-1)^{s_2} \times 1.f_2 \times 2^{e_2-127} \end{aligned}\right\} \quad X = x_1 + x_2 = (-1)^{s} \times 1.f \times 2^{e-127}$$

5. If $s_1 \neq s_2$ and $P_{n-1} = 1$ and $c_{out} = 0$ then $P \leftarrow (P)$"
6. If $s_1 = s_2$ and $c_{out} = 1$ then

   shift right $P_{n-1}\, P_{n-2}\, P_{n-3}\, \ldots\, P_1\, P_0 \;\leftarrow\; 1\, P_{n-1}\, P_{n-2}\, P_{n-3}\, \ldots\, P_1$

   $r \leftarrow P_0$

   $e \leftarrow e + 1$

   Otherwise shift P left L times until $P_{n-1} = 1$

   $P = 0\,0\,0 \ldots 1\, P_s \ldots S_1\, P_0 \;\Rightarrow\; P \leftarrow 1\, P_s\, P_{s-1}\, P_{s-2} \ldots P_0\; g\; r\; 0 \ldots 0$

   $1.f \leftarrow P$

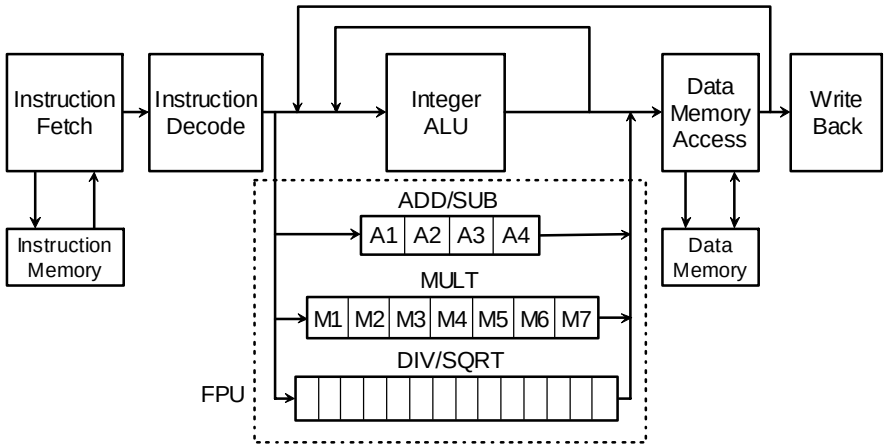
   $e \leftarrow e - L$

   $L = 0 \Rightarrow r \leftarrow g$

   $L = 1 \Rightarrow r$ is rounding bit

   $L > 1 \Rightarrow r \leftarrow 0$

DLX FP Pipeline



Floating Point Unit (FPU)

ADD/SUB: 4 pipelined stages

MULT: 7 pipelined stages

DIV/SQRT: 24 stages — 15 non-pipelined

FP Execution

F = IF D = ID X = EX M = MEM W = WB

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LD F4,0(R2)	F	D	X	M	W												
ADDD F6,F8,F10		F	D	A1	A2	A3	A4	M	W		Pipelined FP ADD						
ADDD F12,F14,F16			F	D	A1	A2	A3	A4	M	W							

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LD F4,0(R2)	F	D	X	M	W												
MULTD F2,F4,F8		F	D	M1	M2	M3	M4	M5	M6	M7	M	W		Pipelined FP MULT			
MULTD F0,F2,F6			F	D	M1	M2	M3	M4	M5	M6	M7	M	W				

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LD F4,0(R2)	F	D	X	M	W						Dependency stalls						
ADDD F2,F4,F8		F	D	D	A1	A2	A3	A4	M	W	Forwarding						
MULTD F0,F2,F6			F	F	D	D	D	D	M1	M2	M3	M4	M5	M6	M7	M	W
SD 0(R2),F0					F	F	F	F	D	X	X	X	X	X	X	M	W

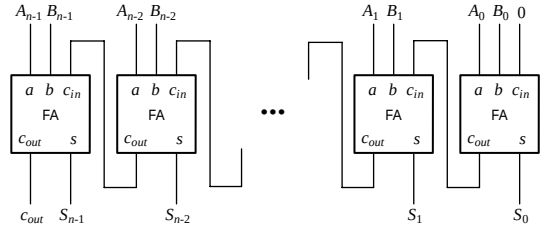
More Hardware → Fewer Steps → Speedup

Ripple Adder

Adds order by order

c_{out} → next order as c_{in}

$n-1$ propagation delays from order to order



Look Ahead Adder

Each stage produces $\begin{cases} s_i = a_i \oplus b_i \oplus c_i \\ c_i = a_{i-1}b_{i-1} + (a_{i-1} + b_{i-1})c_{i-1} = g_{i-1} + p_{i-1}c_{i-1} \end{cases}$

Calculate $n-1$ values for c_{in} in (large) dedicated hardware

$$c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + p_1 c_1 = g_1 + p_1[g_0 + p_0 c_0] = g_1 + p_1 g_0 + p_1 p_0 c_0$$

$$c_3 = g_2 + p_2[g_1 + p_1 g_0 + p_0 p_1 c_0] = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$$

$$c_k = g_{k-1} + p_{k-1}g_{k-2} + p_{k-1}p_{k-2}g_{k-3} + p_{k-1}p_{k-2}p_{k-3}g_{k-4} + p_{k-1}p_{k-2}p_{k-3}p_{k-4}g_{k-5}$$
$$+\ldots + p_{k-1}p_{k-2}\cdots p_1 g_0 + p_{k-1}p_{k-2}\cdots p_0 c_0$$