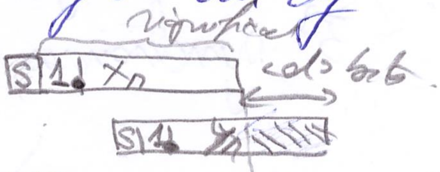
AC C10

2.3. F.p. addition without rounding

$X \pm Y = (X_m \pm Y_m \cdot 2^{Y_e - X_e}) \cdot 2^{X_e}$

- while alligning, keep all bits that were RShifted out of Y_m's storage capacity $X_e \geq Y_e$



<u>adder/subtracter 1</u>

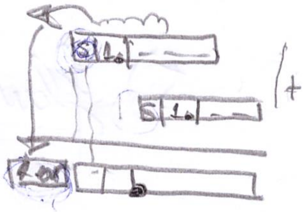
<u>declare registers</u> A[(m+1):0], M[(m+1):0], E1[(e-1):0], E2[(e-1):0], E[(e-1):0], A_COUT, ERROR;

<u>declare bus</u> INBUS[(m+e+1):0], OUTBUS[(m+e+1):0];

BEGIN: A_COUT := 0, ERROR := 0,

INPUT: A := INBUS(X_m), E1 := INBUS(X_E);

M := INBUS(Y_m), E2 := INBUS(Y_E);



COMPARE: E := E1 − E2

ALIGN: <u>if</u> E < 0 <u>then</u> A := <u>RShift</u>(A), E := E+1, <u>go to</u> ALIGN;

<u>if</u> E > 0 <u>then</u> M := <u>RShift</u>(M), E := E−1, <u>go to</u> ALIGN;

ADD/SUBTRACT: A := A ± M, E := <u>max</u>(E1, E2);

OVERFLOW: <u>if</u> A_COUT == 1 <u>then</u> <u>begin</u>

<u>if</u> E == E_{max} (== 2^e−2) <u>then</u> go to ERROR,

A := <u>RShift</u>(A), E := E+1, <u>go to</u> END;

<u>end</u>

ZERO: <u>if</u> A == 0 <u>then</u> E := 0, <u>go to</u> END;

NORMALIZE: <u>if</u> <u>isNormalized(A)</u> == 1 <u>then</u> <u>go to</u> END,

UNDERFLOW: <u>if</u> E > E_{min} (==1) <u>then</u>

A := <u>LShift</u>(A), E := E−1, <u>go to</u> NORMALIZE;

ERROR: ERROR := 1;

END:

Pseudo-language:

1) <u>declare registers</u> ⟨ name, width

- concatenation: A_COUT . A

2) declare bus ↔ name, width — $2m+2e+5$ bits
- unified bus : IOBUS — $m+e+2$ bits.

3) synchronous execution:
- non-colliding operations : concurrent execution, separated by ,
- colliding operations : sequential execution, separated by ;

4) := assignment
- hardwired operators : RShift, isNormalized, not

5) flow control
- unconditional jump : goto <label>
- conditional jump : if A==Error then goto <label>

6) simultaneous read/write from/to register/bus:
A[7] := M[A] even Q[0], Q[0] := 0;

Comments for algorithm:
A) operands $m+e+2$ bits ↔ L: sign, e: exponent, m+1: natural/significand, 1: hidden, implicit, m: fractional part

B) registers A and M:
- RShift capability
- store Sign + Natural/Significand.
- A ↔ LShift capabilities, extended with flag A-OUT.

C) register E:
- ++/-- capabilities.

D) register ERROR:
- indicate exceptions: OVERFLOW, UNDERFLOW

2.4. Rounding and normalisation rules for f.p. addition

$$X_M = 1.\, x_{m-2}\, x_{m-3} \ldots x_1\, x_0 \quad (2^{-1}, 2^{-2}, \ldots, 2^{-m+2}, 2^{-m+1};\ 2^{-m}\ 2^{-m-1}\ 2^{-m-2})$$

<m> bits

$$Y_M = 1.\, y_{m-2}\, y_{m-3} \ldots y_1\, y_0, \text{ and } X_E \geq Y_E$$

Allign Y_M by RShift with d bits, where $d = X_E - Y_E$

Y_M's allignment:
- with infinite precision: keep all bits with weight $< 2^{-m+1}$
- with finite precision: keep only **3 bits** with weight $< 2^{-m+1}$

the 3 bits == sticky bits

- g: guard bit, weight 2^{-m}, the last bit RShifted out of Y_M – guards against loss of precision.
- r: round bit, weight 2^{-m-1}, the last but one bit RShift out of Y_M
- s: sticky bit, weight 2^{-m-2}, obtained by logic OR-ing all the other bits RShifted out of Y_M (excluding g, r)

$$X_M + Y_{M_A} = Z_M$$

Consider $Z_M = z_m \,|\, z_{m-1} \cdot z_{m-2}\, z_{m-3} \ldots z_1\, z_0 \,|\, g\ r\ s$ (carry out z_m; <m>)

Normalize $Z_M \Longrightarrow Z_{M_n}$

$$Z_{M_n} = 1.\, z_{m-2_n}\, z_{m-3_n} \ldots z_{1_n}\, z_{0_n} \,|\, R\ S$$

<m>

$z_\Pi = z_m . z_{m-1} \cdot z_{m-2} z_{m-3} --- z_1 z_0 \mid g \ r \ s$

$z_{\Pi_m} =$

1.	z_{m-2_m}	z_{m-3_m}	---	z_{1_m}	z_{0_m}	R	S
1.	z_{m-1}	z_{m-2}	---	z_2	z_1	z_0	(g or r or s) $++z_E$
1.	z_{m-2}	z_{m-3}	---	z_1	z_0	g	(r or s)
1.	z_{m-3}	z_{m-4}	---	z_0	g	r	s $\rightarrow --z_E$
1.	z_{m-4}	z_{m-5}	---	g	0	0	0 $\rightarrow z_E -= 2$

Normalization cases:

1) $z_m = 1$; 1-bit RShift
2) $z_m = 0, z_{m-1} = 1$; z_Π is normalized.
3) $z_m = 0, z_{m-1} = 0, z_{m-2} = 1$ 1-bit LShift.
4) $z_m = 0, z_{m-1} = 0, z_{m-2} = 0, z_{m-3} = 1$ 2-bit LShift

if a LShift is required by 2 or more bits of z_Π then after appending g, z_Π is completed with 0s

R and S are used for rounding of z_{Π_m} – only 2 bits are needed out of those to be eliminated for implementing the 4 rounding modes of IEEE 754

Rounding mode	$z_{\Pi_m} \geq 0$	$z_{\Pi_m} < 0$
to 0	(discard R, S)	(discard R, S)
towards $-\infty$	——	if (R or S) then $z_{\Pi_m} - 1$
towards $+\infty$	if (R or S) then $z_{\Pi_m} + 1$	——
to nearest even	if (R and (S or z_{0m})) then $z_{\Pi_m} + 1$	if (R and (S or z_{0m})) then $z_{\Pi_m} - 1$

$-1.000005 \downarrow -\infty$ -2

$.11_{(2)} = 0.75_{(10)}$

$.10_{(2)} = 0.5_{(10)}$

?(+1) → partea fr. $> \frac{1}{2}$ (R=1, S=1)
→ partea fr. $== \frac{1}{2}$ (R=1, S=0)
și partea int să fie impară ($z_{0m}=1$)

partea "întreagă" | parte "fracționară"

1. ... z_{0m} | R S

$R \cdot S + R \cdot \bar{S} \cdot z_{0m} = R \cdot (S + z_{0m})$

2.5. F.p. addition/subtraction with rounding

- Rounding error is not correlated with exponents' difference.

$1.75 \times 2^{32} + 1.25 \times 2^{4}$

$\varepsilon\ 1.75 \times 2^{32} + 1.25 \times 2^{31}$

- Use a IEEE 754 format scaled down:
  - 1 – sign
  - 3 – exponent → bias $= 2^{3-1} - 1 = 3$
  - 4 – significand (3 – fractional part of significand)

$X = 0.5625_{(10)} = 0.1001 \times 2^{0} = 1.001 \times 2^{-1}$

$Y = -3.75_{(10)} = 11.11 \times 2^{0} = 1.111 \times 2^{+1}$

Packed operands

X

0	0	1	0	0	0	1	

Y

1	1	0	0	1	1	1	

F.p. addition with rounding Algorithm

Step 1 ⇒ unpack operands
- add the hidden bit to significand.
- check for special cases (zero, ±∞, NaN)

X

0	0	1	0	1	0	0	1

X_E, X_M

Y

1	1	0	0	1	1	1	1

Y_E, Y_M

Step 2 Compute exponents' difference $d = X_E - Y_E$
- if $d < 0 \Rightarrow |X| < |Y| \Rightarrow$ SWAP $X \leftrightarrow Y$
  chose Y_E as result's exponent
- if $d \geq 0 \Rightarrow$ chose X_E as result's exponent

! Only Y_M has RShift capabilities (in order to save area)

$d = x_0 - y_0 = 010_{(2)} - 100_{(2)} = 2_{(10)} - 4_{(10)} = -2 < 0$

$\Rightarrow$ SWAP $x \leftrightarrow y$.

$z_0 = y_0 \rightarrow z_0 = 4$

x

y