1103 NORMALIZED

FLOATING POINT ARITHMETIC ALGORITHMS

21 October 1955

Prepared by: H. Osofsky
G. Weeg

Issued by: Systems Analysis Department
Mathematics and Design Guidance Groups

Remington Rand
DIVISION OF SPERRY RAND CORPORATION
ENGINEERING RESEARCH ASSOCIATES DIVISION
1902 WEST MINNEHAHA AVE ST PAUL W4 MINNESOTA

1103 NORMALIZED
FLOATING POINT ARITHMETIC ALGORITHMS

Introduction

This report presents a set of algorithms intended to accomplish the following four instructions:

1. Floating Add (FAuv) Form in Q the normalized rounded packed floating point sum of the contents of u and the contents of v.

2. Floating Subtract (FSuv) Form in Q the normalized rounded packed floating representation of the contents of u minus the contents of v.

3. Floating Multiply (FMuv) Form in Q the normalized rounded packed floating point product of the contents of u by the contents of v.

4. Floating Divide (FDuv) Form in Q the normalized rounded packed floating point quotient obtained by the division of the contents of u by the contents of v.

Conventions

Any real number can be represented, at least approximately, by $(x \cdot 2^{27}) \cdot 2^y$, where x is a 27-bit real number such that $\frac{1}{2} \leq |x| < 1$, and where y is an integer or zero. The 1103 packed normalized floating point representation selects that subset of the above approximations such that $-128 \leq y \leq 127$.

The floating point word structure is:

1	8	27
S	C	M

where

$$S = 0 \text{ if } x \geq 0,\ S = 1 \text{ if } x < 0;$$

$$x = (-1)^S \left[(1 - 2^{-27}) S + (-1)^S M \cdot 2^{-27} \right] \text{and}$$

$$y = \left[(2^8 - 1) S + (-1)^S C \right] - 128,$$

where if $x = 0$ then $S = C = M = 0$.

That is, S and M define the 1's complement representation of $x \cdot 2^{27}$, while C is the representation of $y + 128$, or, when $S = 1$, C is the 1's complement form of $y + 128$.

If the mantissa becomes zero, or if the characteristic became negative, as the result of one of the four floating point instructions, the result left in Q would be a positive zero.

If the characteristic exceeded 255, an alarm is initiated.

Registers

Besides the registers now found in the arithmetic section of the 1103, these algorithms require the following additional registers:

C register, an 8-bit register, the contents of which are always treated as non-negative. It carries no sign bit. The bit positions of the C register are designated

$$C_7\ C_6\ C_5\ C_4\ C_3\ C_2\ C_1\ C_0.$$

D register, an 8-bit register similar to C. Its bit positions are denoted

$$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0.$$

S register, a 9-bit subtractive accumulator. The bit positions of S are denoted

$$S_8\ S_7\ S_6\ S_5\ S_4\ S_3\ S_2\ S_1\ S_0.$$

The arithmetic in the S register is assumed to be ones complement. S_8 is used as both an overflow bit and a sign bit. This register must be able to be complemented.

Transmission Paths

The following transmission paths are also required:

From (X_{34} --- X_{27}) to C.

From (S_7 --- S_0) to (X_{34}--- X_{27}). transfer input

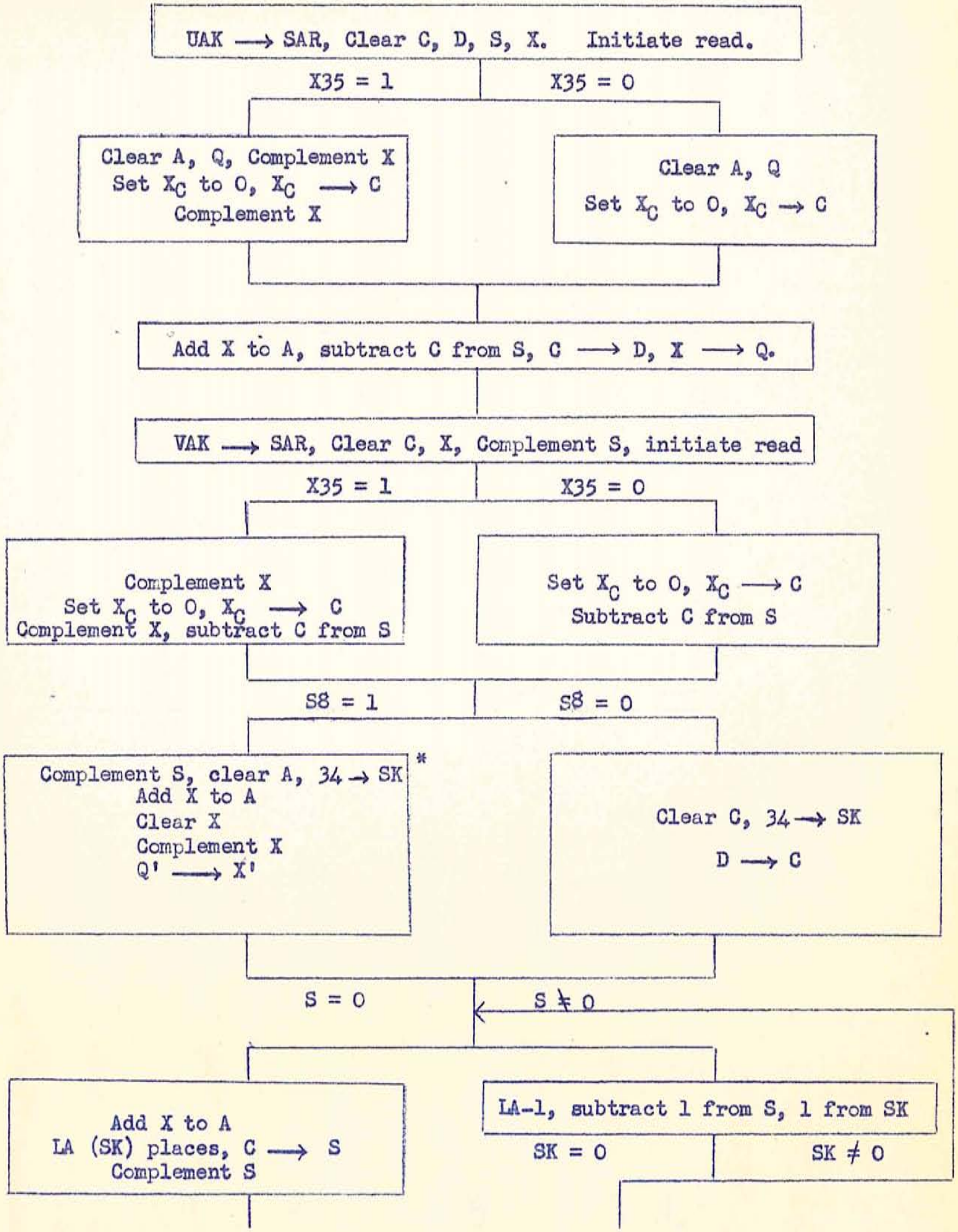
From C to D.

From C subtractively to S.

From D to C.

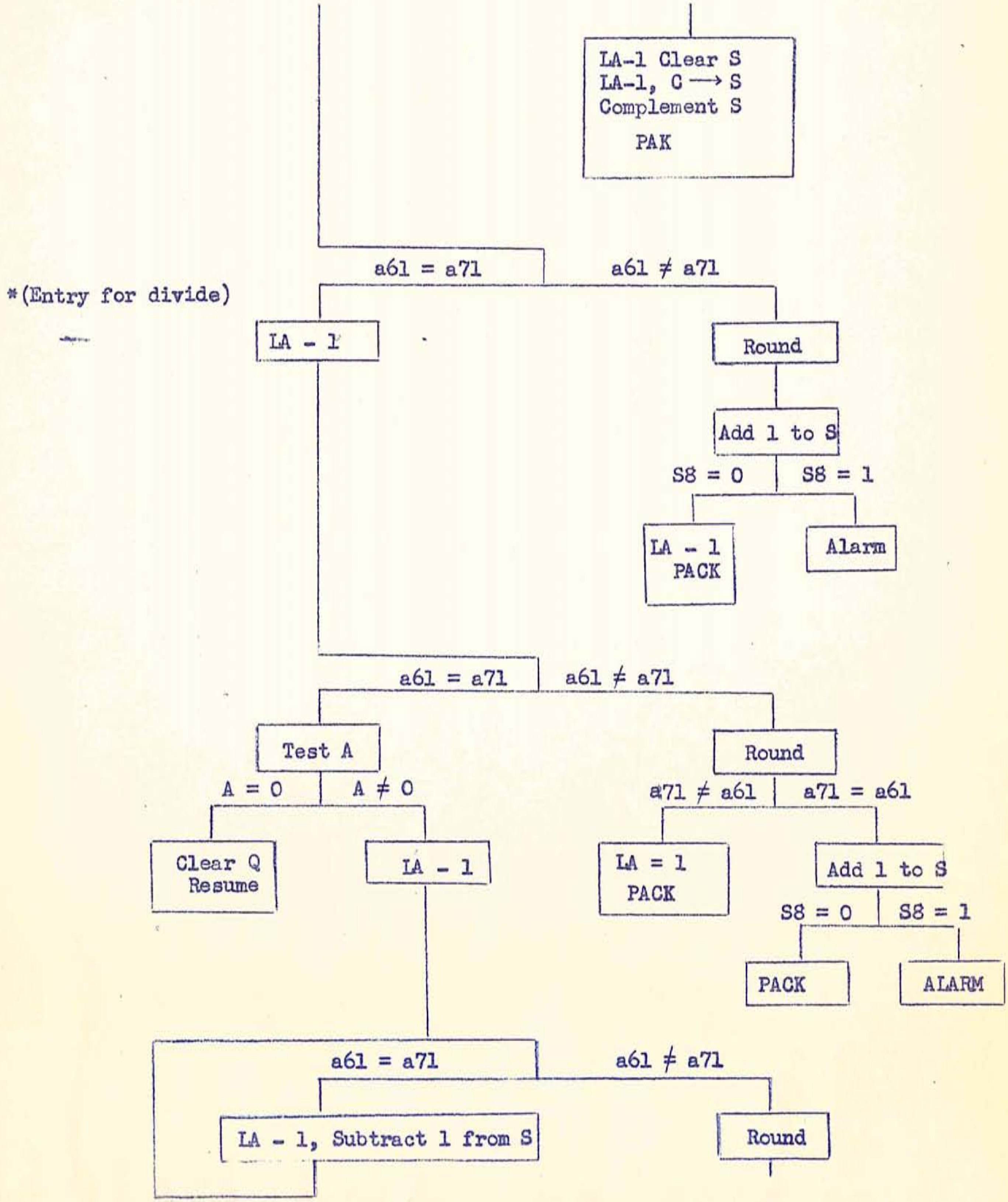
From A_L to X.

Floating Add

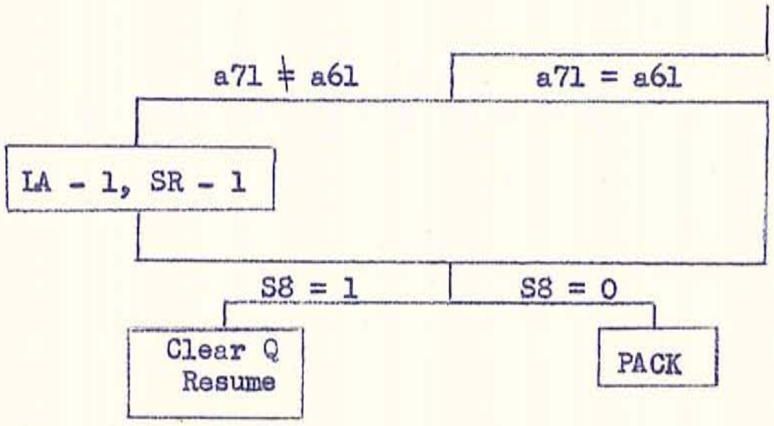

UAK → SAR, Clear C, D, S, X. Initiate read.
X35 = 1
X35 = 0
Clear A, Q, Complement X
Set X_C to 0, X_C → C
Complement X
Clear A, Q
Set X_C to 0, X_C → C
Add X to A, subtract C from S, C → D, X → Q.
VAK → SAR, Clear C, X, Complement S, initiate read
X35 = 1
X35 = 0
Complement X
Set X_C to 0, X_C → C
Complement X, subtract C from S
Set X_C to 0, X_C → C
Subtract C from S
S8 = 1
S8 = 0
Complement S, clear A, 34 → SK *
Add X to A
Clear X
Complement X
Q' → X'
Clear C, 34 → SK
D → C
S = 0
S ≠ 0
Add X to A
LA (SK) places, C → S
Complement S
LA-1, subtract 1 from S, 1 from SK
SK = 0
SK ≠ 0

*When the number n is put into SK, this means a left shift of n places is to be called for.

next page

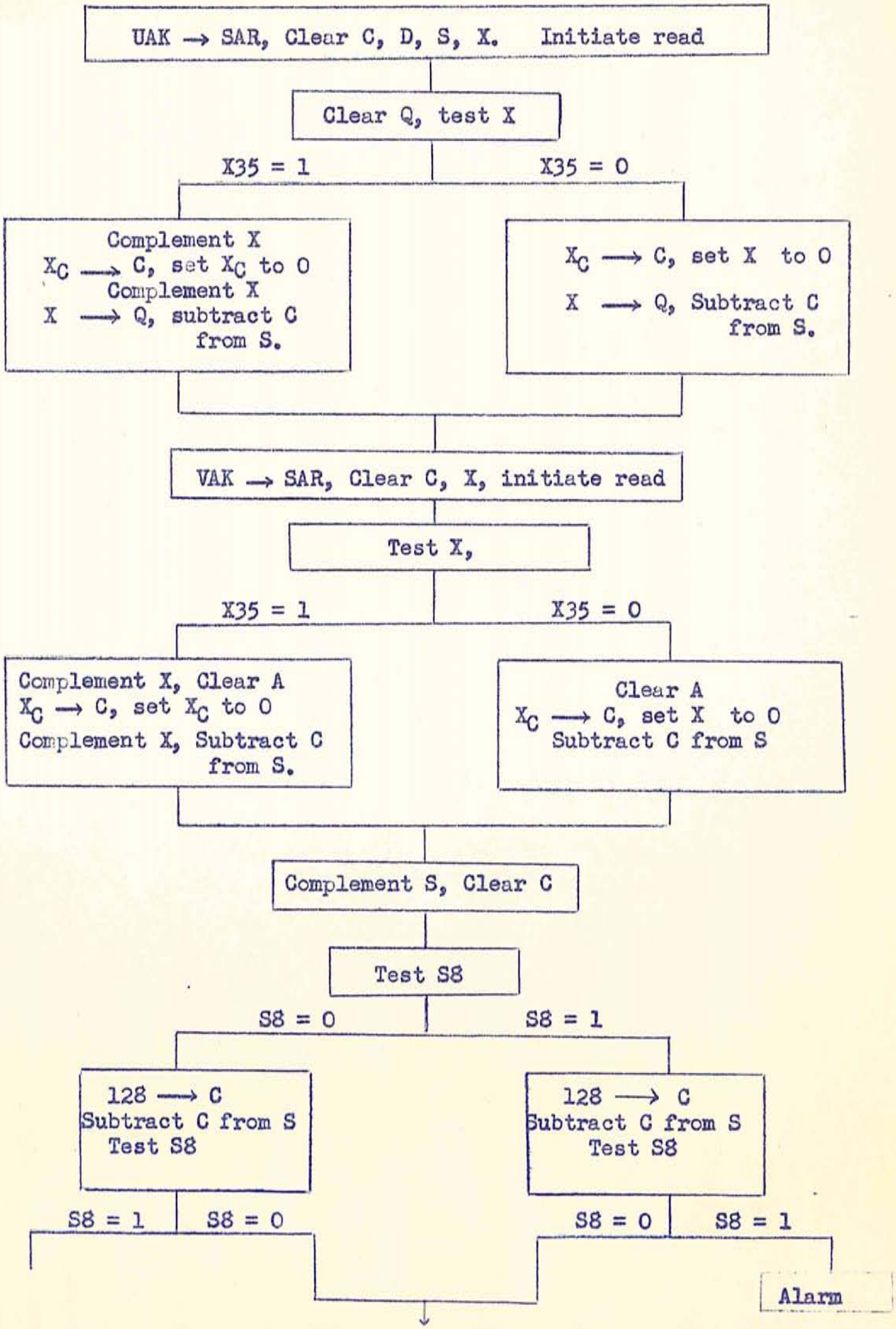

LA-1 Clear S
LA-1, C ⟶ S
Complement S
PAK
a61 = a71
a61 ≠ a71
*(Entry for divide)
LA - 1
Round
Add 1 to S
S8 = 0
S8 = 1
LA - 1
PACK
Alarm
a61 = a71
a61 ≠ a71
Test A
A = 0
A ≠ 0
Clear Q
Resume
LA - 1
Round
a71 ≠ a61
a71 = a61
LA = 1
PACK
Add 1 to S
S8 = 0
S8 = 1
PACK
ALARM
a61 = a71
a61 ≠ a71
LA - 1, Subtract 1 from S
Round

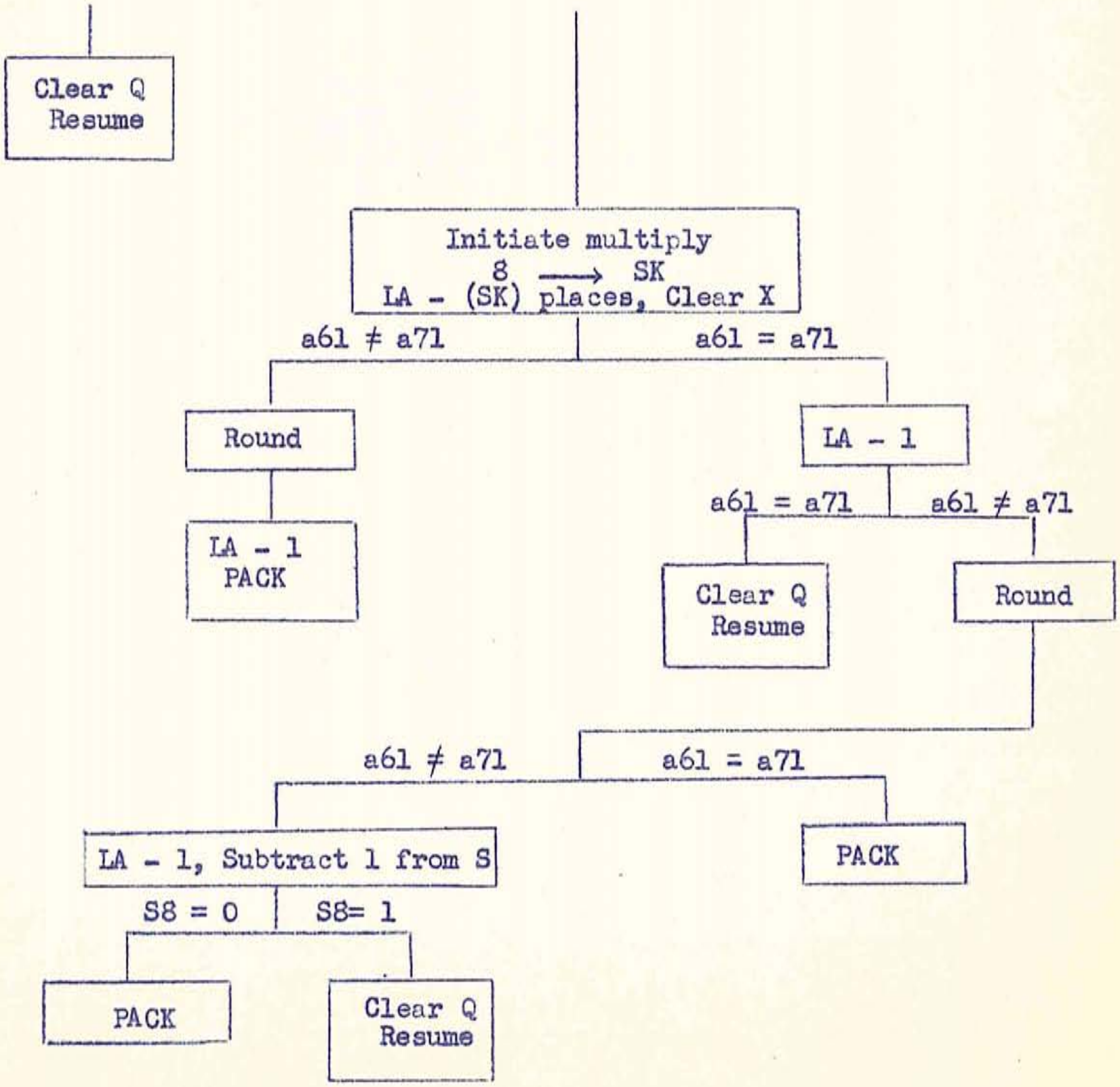
next page



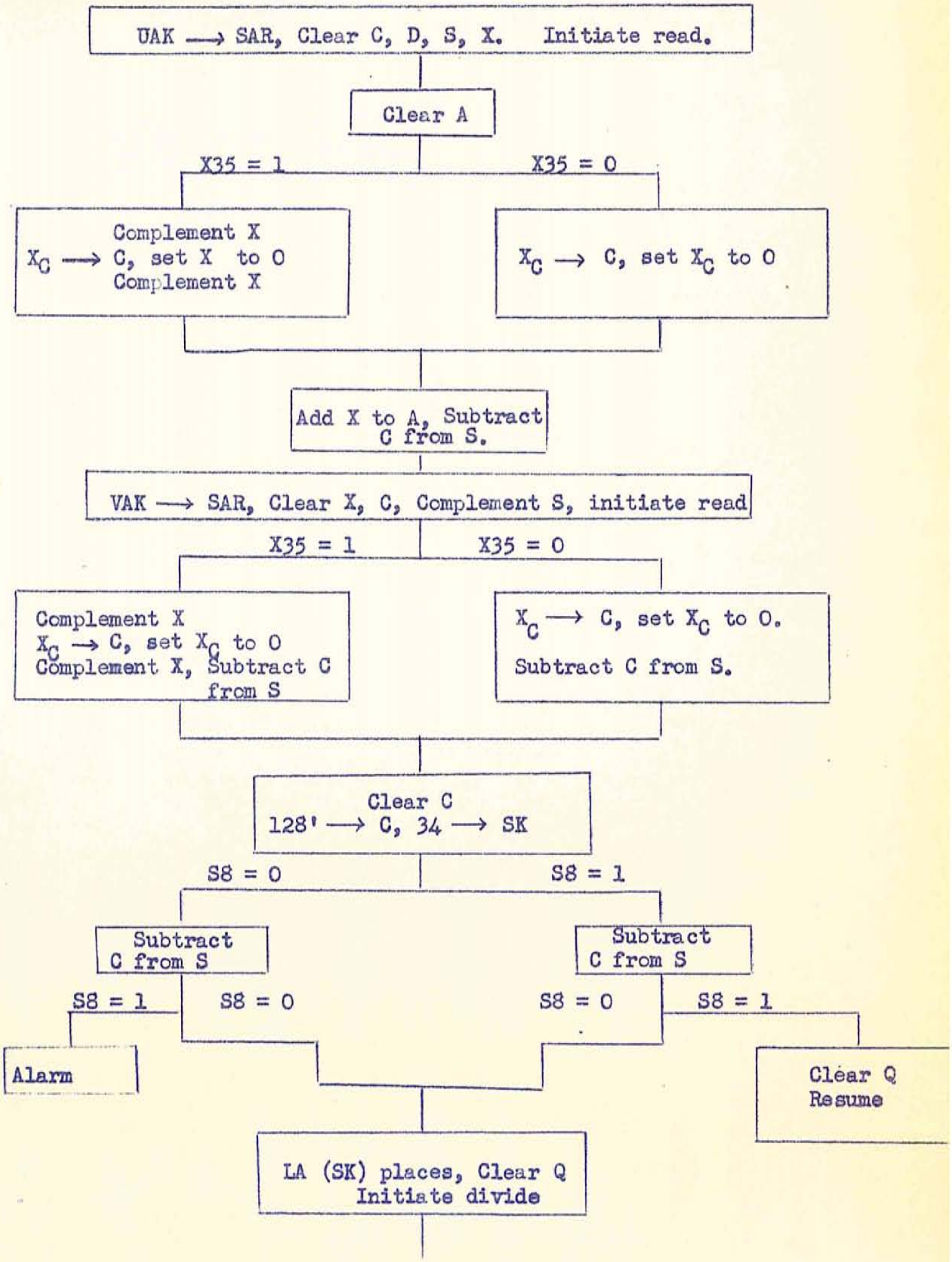
Floating subtract is the same as floating add, except that where (v) is entered into X in add, (v)' is entered in subtract.

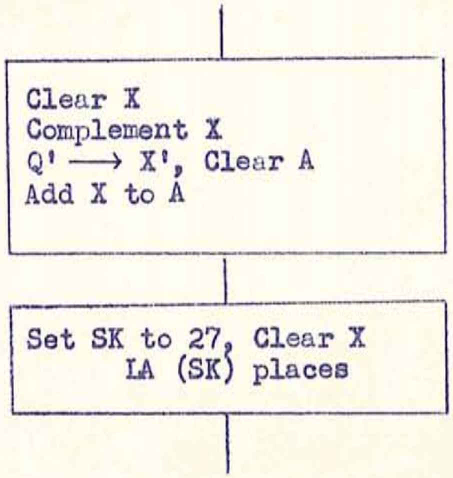
Multiply


UAK → SAR, Clear C, D, S, X. Initiate read
Clear Q, test X
X35 = 1
X35 = 0
Complement X
X_C → C, set X_C to 0
Complement X
X → Q, subtract C from S.
X_C → C, set X to 0
X → Q, Subtract C from S.
VAK → SAR, Clear C, X, initiate read
Test X,
X35 = 1
X35 = 0
Complement X, Clear A
X_C → C, set X_C to 0
Complement X, Subtract C from S.
Clear A
X_C → C, set X to 0
Subtract C from S
Complement S, Clear C
Test S8
S8 = 0
S8 = 1
128 → C
Subtract C from S
Test S8
128 → C
Subtract C from S
Test S8
S8 = 1
S8 = 0
S8 = 0
S8 = 1
Alarm


Clear Q
Resume
Initiate multiply
8 ⟶ SK
LA - (SK) places, Clear X
a61 ≠ a71
a61 = a71
Round
LA - 1
PACK
LA - 1
a61 = a71
a61 ≠ a71
Clear Q
Resume
Round
a61 ≠ a71
a61 = a71
LA - 1, Subtract 1 from S
PACK
S8 = 0
S8= 1
PACK
Clear Q
Resume

Divide


UAK ⟶ SAR, Clear C, D, S, X. Initiate read.
Clear A
X35 = 1
X35 = 0
Complement X
X_C ⟶ C, set X to 0
Complement X
X_C ⟶ C, set X_C to 0
Add X to A, Subtract C from S.
VAK ⟶ SAR, Clear X, C, Complement S, initiate read
X35 = 1
X35 = 0
Complement X
X_C → C, set X_C to 0
Complement X, Subtract C from S
X_C ⟶ C, set X_C to 0.
Subtract C from S.
Clear C
128' ⟶ C, 34 ⟶ SK
S8 = 0
S8 = 1
Subtract C from S
Subtract C from S
S8 = 1
S8 = 0
S8 = 0
S8 = 1
Alarm
Clear Q
Resume
LA (SK) places, Clear Q
Initiate divide

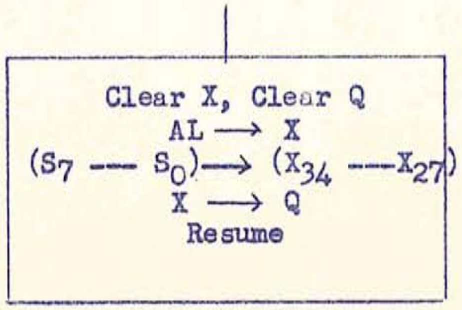


Clear X
Complement X
$Q' \longrightarrow X'$, Clear A
Add X to A

Set SK to 27, Clear X
LA (SK) places

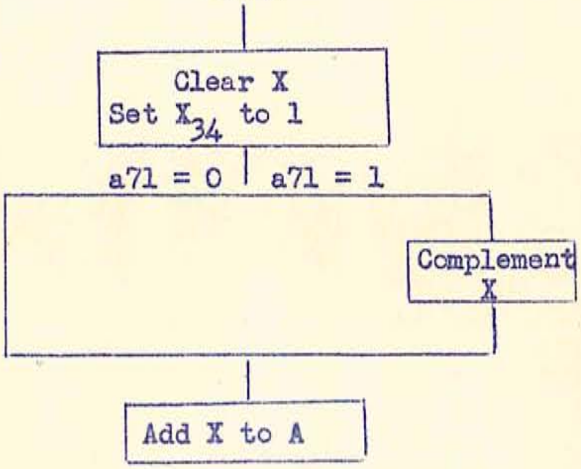
To addition end.*

PACK



Clear X, Clear Q
AL $\longrightarrow$ X
$(S_7 \text{ --- } S_0) \longrightarrow (X_{34} \text{ ---} X_{27})$
X $\longrightarrow$ Q
Resume

Round



Clear X
Set X_{34} to 1

a71 = 0 | a71 = 1

Complement X

Add X to A