FP6000 PROJECT MEMORANDUM #30

October 24, 1963
Revised May 25, 1964

Specification of Floating Point Arithmetic Unit

(Part of FP6002 and FP6004)

The following specification has been drawn up by Product Planning in conjunction with Engineering. Revisions have been underlined.

W.R. Whittall
Product Planning
Computer Systems

DISTRIBUTION LIST:

L.R. Wood

ICT Australia

ICT(U.K.) (2)

Engineering

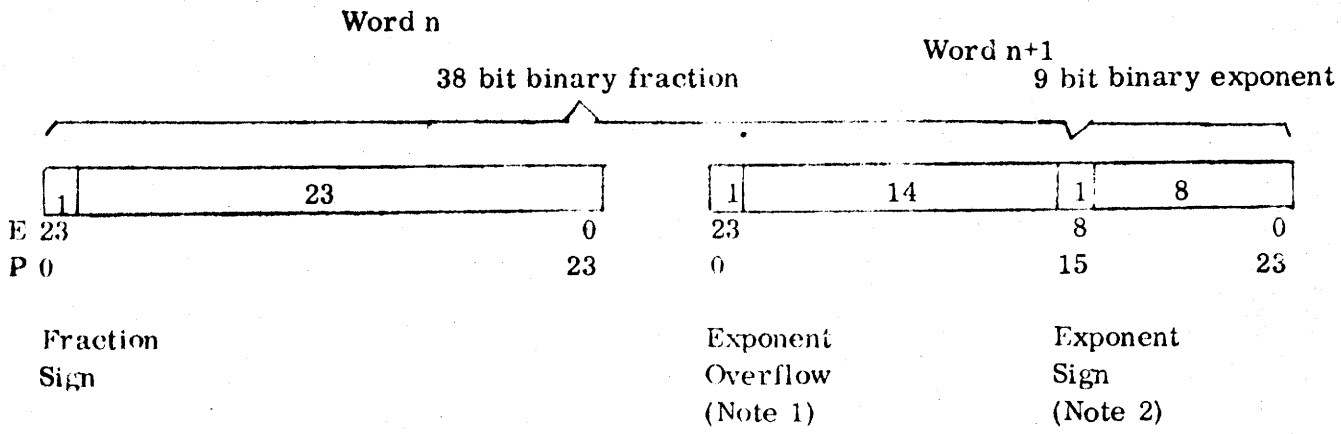
G. Collins
P. Stevens
E. Strain
I. Fekete
B. Ovenell
J. Davidson
H. Reddering
K. Bower
G. Freel
V. Caldwell
B. Cellier
R. Cass
W. Joyner
M. Rex
J. Parsonage

Marketing

D.K. Ritchie
A. Adams
P. Adams
J. Chapman
T. Croil
L. Cragg
B. Daly
H. Foulds
A. Henry
T. Hobbs
R. Johnston
L. Lawry
F. Longstaff
M. Lucas
M. Marcotty
J. McKenzie
R. Moore
E. McDorman
J. McSherry
D. Oldacre
D. Rees
R. Sadana
I. Sharp
A. Sharp
D. Smith
V. Taylor
W. Whittall (3)
J. Pollard
File

Specification of Floating Point Arithmetic Unit

1. Floating Point Number Representation



Note 1 This bit corresponds to the exponent overflow flip-flop in the floating point arithmetic unit. The flip-flop will set if a floating point operation causes exponent overflow or if the number used has a one in this position. The flip-flop can be tested by using a "store" or "store and clear" operation which sets the state of the flip-flop into the sign bit of word N+1 and sets the main machine overflow flip-flop if exponent overflow is set. The flip-flop can be cleared by loading a number which does not have a one in the sign bit of word N+1 or by using a "store and clear" operation.

Note 2 Negative exponents are represented in two's complement form. The exponent sign bit is inverted, so that an exponent of -256 is represented by nine zeros.

Note 3 If the final result of any operation (after making any corrections for temporary fraction overflow or after normalizing) has an exponent of -256 or less the whole of the floating point accumulator is cleared except for the exponent overflow bit.

2. Functions

The Floating Point Arithmetic unit is an autonomous unit containing a number of registers including a 48 bit floating-point accumulator which consists of a 38 bit signed fraction, a 9 bit signed exponent and an exponent overflow flip-flop. The available functions are as follows:-

132	Add	f + n: → f
133	Subtract	f - n: → f
134	Multiply	f x n: → f
135	Divide	f ÷ n: → f
136	Load	n: → f
137	Store	f → n:

where f is the 48-bit floating point accumulator.

The x bits are used to further define the function as follows:-

X_1
- functions 132 to 135 — Unrounded result
- function 136 — Load zeros and clear the exponent overflow flip-flop.
- function 137 — Store f and then clear f and the exponent overflow flip-flop.

X_2
- functions 132 to 135 — Unnormalized result.
- otherwise — Ignored

X_4
- functions 133, 135 — Reverse the function (i.e. n: - f→f or n: + f→f)
- otherwise — Ignored.

Notes:

1. Operands can be normalized or unnormalized.

2. Exponent overflow will be set if the fractional part of the denominator in a division is not normalized and is less than the fractional part of the numerator. The answer will be correct, however, if the fractional part of the denominator is greater than half the fractional part of the numerator.

3. Timing The floating point arithmetic unit contains its own microprogram control which runs at one megacycle independently of the main microprogram control. When a floating point instruction is encountered, the main microprogram control first reads the instruction from the store and modifies the N address if necessary.

At this point the main microprogram waits if the floating point unit is still busy with the previous operation and then transfers n and n+1 to (or from) the floating point unit which then proceeds independently with the arithmetic operation while the main microprogram is free to begin the next instruction. Thus floating point operations can be divided into three phases:-

1. Preparation of order in the main microprogram taking one or two store cycles depending on address modification.
2. Transfer of n and n+1 to the floating point unit taking two store cycles, except in the case of Store when the transfer is in the opposite direction and takes an extra 2 μs.
3. Arithmetic operation in the floating point unit. In the case of Load and Store this phase does not occur.

If the next instruction is also a floating point instruction phase 1 can begin at the end of the previous phase 2 and phase 2 can begin at the end of phase 1 or the previous phase 3 whichever is later. If the next instruction is a non floating point instruction, it will begin at the end of phase 2.

The time in microseconds for phase 3 only is as follows:-

	Min.	Typical	Max.
Add/Subtract	6	8	43
Multiply	23	24	24
Divide	45	47	49

Notes

1. Typical and Maximum figures include rounding and normalizing if the operands are normalized.
2. The typical Add/Subtract time assumes 2 shifts for exponent equalization and 1 shift for normalization.

W. R. Whittall