DATE: August 6, 1981

TO: R & D Personnel

FROM: Dave Waxman, Mark Johnson

SUBJECT: Quad Precision Floating Point

REFERENCE: Fox Architecture Reference--Martha August. PE-TI-748, "A New Look at Prime's Floating Point Architecture," by Suren Irrukella - May 22, 1980.

KEYWORDS: None

ABSTRACT

Quad precision floating point is an agreed essential feature for Marketing. This is a new format, 128 bits long yielding 96 bits (28 digits) of precision. It will be available in both FOX and VOLE, hopefully from FCS (First Customer Ship).

1 Introduction--(Abstract)

Quad precision floating point is an agreed essential feature for Marketing. This is a new format, 128 bits long yielding 96 bits (28 digits) of precision. It will be available in both FOX and VOLE, hopefully from FCS (first customer ship).

This paper defines the formats, instructions and implementation strategies.

2 Formats--Quad

2.1 Memory Format

N	1	Frac	16
N+1	17	Frac	32
N+2	33	Frac	48
N+3	1	Exp	16
N+4	49	Frac	64
N+5	65	Frac	80
N+6	81	Frac	96
N+7		Unused	

Storage is in blocks of 8 16-bit words in address order as shown. The exponent is a 2's complement 16 bit excess '200 representation identical to that used by double precision. The fraction is 96 bits, two's complement.

2.2 Accumulator Format

Both the I-mode floating point accumulators (FP0 and FP1 or FAC0 and FAC1) are used to represent a single QUAD floating point value. FAC1 contains the high order 64 bits from the memory format and FAC2 the low order 64 bits. This reversal from the obvious means of expressing QUAD is to permit a cleaner overlap of QUAD with double in V-mode .

Quad Floating Accumulator (QAC)

```
            ------------------------
FAC1       |  1      Frac     48   |
           |  1      Exp      16   |
           |                       |
           |------------------------
FAC0       | 49      Frac     96   |
           |       (unused)        |
           |                       |
            ------------------------
```

Note: QAC is pronounced "Quack".

2.3 Special Numbers

2.3.1 Zero

Zero is generated as 8 words of zero. Zero is checked as 96 bits of zero fraction.

2.3.2 Unnormalized Numbers

Numbers are assumed to be normalized. Operations do not prenormalize. Unnormalized numbers participate successfully in all operations except divide. If divide cannot generate correct results because of unnormalized numbers, it generates an overflow/underflow exception. All results are normalized. Load and store never change the contents of the 128 bits they move about (i.e. they never normalize).

3 Instructions--V Mode

3.1 Memory Reference--V Mode

The memory referencing instructions introduce a new form of OP code (augmented, extended memory reference) of the following form:

```
                    1   2  3   6 7    11 12 13 14 15 16 17  32
                   ____________________________________________
first 32 bits:   |  IX  |  OP  | 11000 | Y | XX | BR |  Disp  |
                   ____________________________________________
```

(from figure 2.4 FOX Ref. Manual)

```
                          1                             16
                           _____________________________
      remaining 16 bits:  |        augmentation         |
                           _____________________________
```

All but one of the memory referencing instructions share the same extended memory reference code: 0101, 10 (5,2).

The augmented op code is defined as:

0 QFLD--Quad Floating Load

The 128 bit quantity specified by EA is moved to the Quad accumulator. No other status changes. The unused 16 bits are faithfully copied into the QAC.

1 QFST--Quad Floating Store

The 128 bits of the QAC are stored into the 128 bits of memory starting from EA. No other status changes.

2 QFAD--Quad Floating Add

Adds the 128 bits, Quad precision number contained in the locations specified in EA to the QAC. Normalizes the results. The least significant 16 bits from memory are ignored. The least significant bits of the QAC are ignored during the operation and are zeroed at the end. The condition codes and LINK are indeterminate.

Floating point faults for this instruction, overflow/underflow, can occur and are treated as shown in section 5. Cbit resets to zero on normal operations.

3 QFSB--Quad Floating Subtract

Same as QFAD except subtracts the memory Quad value from the QAC.

4 QFMP--Quad Floating Multiply

Same as QFAD except multiplies the memory value to the QAC, leaving the results in the QAC.

Floating point faults for this instruction, overflow/underflow, can occur and are treated as shown in section 5. Cbit resets to zero on normal operations.

5 QFDV--Quad Floating Divide

Divides the 128 bit Quad precision number contained in the location specified by EA into the QAC (QAC/[EA]). Leaves the quotient in the QAC. The condition codes and link are indeterminate. The unused parts of the format do not participate. The QAC unused piece is zeroed on completion.

Floating point faults for Quad: overflow/underflow and divide by zero can occur and are treated as shown in section 5. The Cbit is reset upon successful completion of the operation.

6 QFCS--Quad Floating Compare

Compares the 128 bit contents of the QAC to the 128 bit memory value. Skips according to this table:

Test	Action
QAC>[EA]	no skip
QAC=[EA]	skip 1 word
QAC<[EA]	skip 2 words

The unused parts do not affect the comparison and are not modified. CC's, CBit, and Link are indeterminate.

7'177777 not used--generate a UII

3.1.1 QFLX

A second memory reference code has been used in an unextended way to implement QFLX.

QFLX--Quad Floating Load Index

01101, 11 type: extended memory reference

This instruction is from the extended LDX class so it cannot be indexed.

An effective address is generated. The 16 bit contents of the effective address is loaded into the X register after multiplying it by eight. (A left shift of three) zeros are shifted in on the right. Data is shifted out through bit 2 and then bit 1. Leaves the Cbit, Link and condition codes unchanged.

3.2 Generics--V Mode

The GENA group of generic instructions has the following additions: (these are the old FRAC op codes so they will UII on all existing 50 series and 400 architecture machines).

QFCM Quad Floating Complement

140570

The contents of QAC is arthmetically complemented. The unused portion of QAC is zeroed. Link and CC's are indeterminate.

Overflow/underflow is possible and controlled as described in section 5. Cbit is zeroed on a normal operation.

FCDQ Floating Convert Double to Quad

140571

The contents of FACO (FPO) are cleared. Cbit, Link and Condition codes are unchanged.

QINQ Quad to Integer in Quad Convert

140572

The contents of QAC are examined and the integer portion of the value is returned as a Quad floating point value.

if exponent:	then
≥ '200 + 96	Fault: conversion error (see section 5)
< '200 + 96 and > '200	Strip fractional part if positive. If negative strip fractional part, incrementing integer part if fractional part = 0.
= '200	If positive, return zero If negative, return -1 if bits 2 --> 96 = 0.
< '200	Return 0

The condition codes and link are indeterminate.

Normal operations reset the Cbit. Faults do not change the value of QAC.

The Cbit is modified as described in section 5 on faults.

QIQR Quad to Integer, in Quad Convert Rounded

140573

Same as QINQ except round by adding 1 to the integer part (if it exists) if the MSB of the fractional part = 1. For the special case of exponent = 95 + '200, set the MSP of the fractional part = 0.

3.3 Rounding Control Generics

The Quad format also introduces the ability to direct rounding for unbiased double precision results and round up and round down. The full set of rounding is included in the proposal for completeness and to pave the way for the IEEE floating point hardware rounding. The op codes are taken from the class of shifts which would be arithmetic rotates, but are unimplemented (of course).

A new concept which these instructions will permit is the automatic generation of interval arithmetic. This could be done by defining a date type as INTERVAL *4 X, Y, Z to FORTRAN and generating a double precision floating point pair which maintained the high and low intervals. The instructions emmitted then use the appropriate round instruction before storing. The advantage of this definition mode over any other is the ability to correctly mainpulate the interval quantities. If the 0th register and first half of the 64 bit quantity contain the least interval, then:

```
INTERVAL *4 X, Y, Z

X = Y + Z

Z = Y/X

generates:

    FL     0,Y                    /*Least interval
    FL     1,Y+2                  /*Greatest
    FA     0,Z                    /*L + Least = Least
    FA     1,Z+2                  /*G + Greatest = Greatest
    FRNM     0                    /*Round to - infinity
    FRNP     1                    /*Round to + infinity
    FST    0,X
    FST    1, X+2                 /*Now the divide
    FL     0,Y
    FL     1,Y+2
    FD     0,X+2                  /*L/G = New Least
    FD     1,X                    /*G/L = New Greatest
    FRNM     0
    FRNP     1
    FST    0,Z
    FST    1,Z+2
```

This is a unique and powerful feature. With it, PR1ME can offer a reasonably rigorous, fast and very easy check of algorithms. The real infinite precision correct answer will always lie within the interval specified by the number pair.The feature also exploits in a direct and gainful way the extended precision we routinely carry today.

DRN Double Round from Quad

040300

Converts a quad value in the QAC to a double precision floating point value in the DAC using unviased rounding to the closest representable floating point number in double precision. Ties are broken by rounding to the even value. The results are normalized. CC's and link are indeterminate. The Cbit is normally reset. Overflow can occur and is treated as shown in section 5 as a QUAD fault.

Definition of unbiased round to even for QAC to DFAC.

1) If QAC = 0, done

2) If (bits 50 --> 96 not equal 0 or bit 48 = 1, then add bit 49 to 48.

3) Clear bits 49 --> 96.

4) Normalize the DAC

DRNP Double Round from QUAD Toward Plus Infinity

040301

Converts a quad value in the QAC to a double precision value in the DFAC by always rounding to the larger of the two representative values in the QAC. The link and CC's are indeterminate. The Cbit is normally reset but overflow can occur and is treated as a QAC overflow as described in section 5.

Definition of round toward plus infinity:

1) If QAC = 0, done.

2) If (bits 49 --> 96 = 0), done.

3) Add 1 to bit 48 of QAC.

4) Clear bits 49 --> 96.

5) Normalize the DAC.

DRNM Double Round from QUAD toward minus infinity

140571

(same as FCDQ)

Round the QAC towards the smallest representative value in DAC. This is equivalent to clearing bits 49 to 96 of the QAC.

DRNZ Double Round from QUAD Toward Zero

040302

Selects the representative value in DAC closest to zero for the value in QAC. CC's and link are indeterminate. The Cbit is normally zeroed but overflow can occur as a QUAD overflow. See section 5.

Definition of round toward zero.

1) If (QAC = 0), done.

2) If (bit 1 = 1 and bits 49 --> 96 not equal 0), then add 1 to bit 48.

3) Clear bits 49 --> 96.

4) Normalize the DAC.

FRN Floating Round

140534

(Exists today in slightly modified form).

Rounds the number in the DAC to its closest representative single precision value. In case of a tie, round to the even value. The CC and links are indeterminate. The Cbit is reset as a normal result. Overflow can occur and is treated as indicated in section 5.

Definition of round to nearest even:

1) If DAC = 0, done.

2) If (bits 26 --> 48 not equal 0 or bit 24 = 1), add bit 25 to 24.

3) Clear bits 25 --> 48.

4) Normalize the FAC.

Note: this is a change. OLD: Round to nearest except for ties, on ties round toward + infinity. The change to the results is a tiny but reasurable improvement in long term accuracy.

FRNP Floating Round Toward plus Infinity

040303

Converts the DAC value to a single precision value by selecting the larger of the two possible representations of the DAC value in single precision format. The link and CC's are indeterminate. Normal operations reset the Cbit. Overflow can occur and is treated as shown in section 5.

Definition:

1) If DAC = 0, done.

2) If (bits 25 - 48 = 0), done.

3) Add 1 to bit 24 of DAC.

4) Clear bits 25 - 48 of DAC.

5) Normalize the DAC.

FRNM Floating Round Toward Minus Infinity

040320

Converts the DAC value to a single precision value by selecting the smaller of the two possible representations of the DAC value in single precision format. The link, CC's and Cbit are unchanged. Overflow cannot occur.

Definition:

1) Clear bits 25 - 48 of the DAC.

FRNZ Floating Round Toward Zero

040321

Selects the representation single precision value closest to zero for the value in DAC. The condition codes and link are indeterminate. The Cbit is normally zeroed but overflow can occur as shown in section 5.

Definition of round toward zero:

1) If DAC = 0, done.

2) If (bit 1 = 1 and bits 25 - 48 not equal 0), then add 1 to bit 24.

3) Clear bits 25 - 48.

4) Normalize the DAC.

4 I Mode Instructions

The I mode instructions are functionally identical to their V mode instructions by the same name.

4.1 Memory Referencing--I Mode

These instructions are from the special memory reference section (A.3 in the FOX Architecture Reference). No immediate or register-register forms are available.

QFLD	34
QFST	35
QFAD	36
QFSB	37
QFMP	45
QFDV	46
QFCS	47

The QFLX instruction is in I mode as:

```
LHL3, r, address
Load Halfword Shifted Left by 3
011101  DR/3  TM/2  SR/3  BR/2
[Displacement/16]
```

Calculates an effective address, EA. Shifts the 16 bit contents of the location specified by EA left 3 bits and stores the result in the specified r. Leaves the values of Link, Cbit and the condition codes unchanged.The RR form is available.

4.2 Generics--I Mode

The same instructions and op codes are available for I-mode as V mode:

QFCM	140570	DRN	040300
FCDQ(DRNM)	140571	DRNP	040301
QINQ	140572		
QINR	140573	DRNZ	040302

The single precision round instructions work on each of the two floating point registers. The following instructions and op codes used:

```
FRNP    0110000F01100101
        (FOX p. 425 Table A.15 FRNP 145
FRNM    0110000F01100110
        (FOX p. 425 Table A.15 FRNM 146
FRNZ    0110000F01100111
        (FOX p. 425 Table A.15 FRNZ 147
```

5 Faults

See section 11.3.8 of the FOX Architecture Reference. Quad faults are controlled by bit 7 of the keys exactly like the other floating point data types.

The table 11.10 is extended as follows:

Data Type	Exception Type	FCODEH	FADDR
Quad precision	overflow or underflow	$800	addr of faulting ins
Floating point	Divide by 0	$801	"
	QINQ exception	$803	"

Overflow may be reliably detected from underflow by testing the exponent. If the exonent is positive, underflow occurred. If the exponent underflow is negative, overflow occurred.

6 Rounding and Accuracy

The Quad format will produce 96 bits of truncated results. The rounding keys bit is ignored. Divide will produce exact results if integer values are used.

7 Changed to existing instructions

FRN has been modified as shown in section 3.3.

FCS is changed as folows:

1) In normal no rounding mode, no change.

2) In rounding mode:

Before the fractions are compared, the DAC is rounded as it is for FST in round mode. The contents of DAC are not changed, but the memory value is compared against the rounded DAC value.

This change should protect us against the obvious difficulties associated with our extended precision accumulators. Then, when a calculation is completed and stored, with rounding, a subsequent comparison of that stored value to the accumulator will compare exactly equal. Now, the comparison finds the less significant bits in the DAC and finds a not equal condition. The change of "How come I stored it away, compared it, and got a different answer?", is easier to fix than explain. (We will still have the problem, but less often).

8 Representation in PMA, F77

Real *16 is the F77 Quad data type specifier. The constant specifier for both PMA and F77 is a Q before the exponent where E is found for single precision. Example: 1 in Quad = 1.0Q0.

9 Task List

This section lists the various tasks currently identified. Latest dates correspond to requirements to make FOX FCS.

9.1 UII begin:now end:(latest) 10-15-81

The UII package is required to allow software checkout to proceed in an orderly fashion on any 50 series machine. The package can also be used by the test program as a checker. Simple routines are to be used as time to develop and confidence in the results is more important than speed of execution.

The fault simulation must be done with care. How PRIMOS currently reflects FLEX faults must be examined and possibly modified. In any event, the faults must be simulated in ring 3, so the PRIMOS handeling must be assumed.

9.2 Test Program begin:now end:(latest) 11-1-81

An extention to FCT/FOX should suffice for both FOX and VOLE as the results must be identical.

9.3 Assembler Mods begin:now end:(latest) 9-15-81

Required for the test program and UII package.

9.4 F77 Mods begin:now end:(latest) ?

9.5 Microcode begin:now end:(latest) 12-31-81

End date dictated by FOX engineering release. This date could move out only by negotiating with Manufacturing.

9.6 Libraries begin:now end:?

Issues here include:

a. Completeness of function support.

b. Required accuracy.

c. Support of Quad complex.

d. Rewrites of Double complex and Double to exploit Quad.

9.7 Interval Mode begin:? end:?