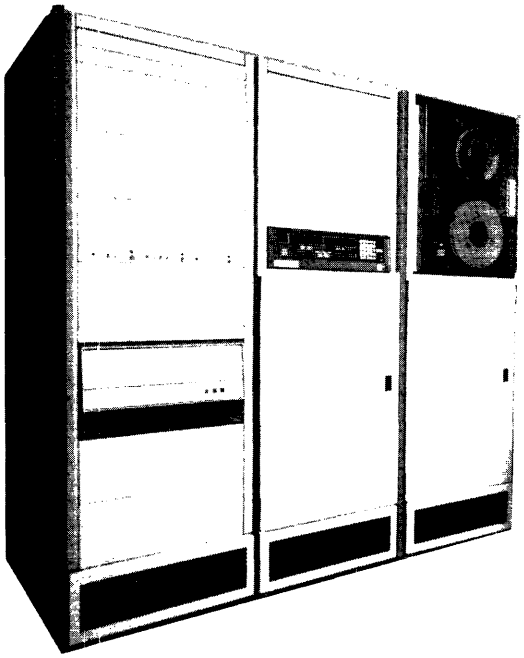
INTERDATA

PROCESSOR ENHANCEMENTS



High Performance Floating Point for the Model 8/32 Megamini®

PRODUCT DESCRIPTION

The M83-111 is a truly high performance floating point arithmetic unit for the Interdata 8/32-C Megamini: high performance in number of instructions to control floating point operations, in precision, and in speed. The Interdata 8/32 Megamini features a 32-bit word and up to one million bytes of 750-nanosecond core memory. It gives large scale computer performance at minicomputer prices.

Thirty instructions provide Load, Store, and Compare in addition to the standard floating point arithmetic instructions of Add, Subtract, Multiply, and Divide. Operands can be either single or double precision. Both operands can be located in the floating point registers, or one operand can be in the floating point registers and the other in memory. Load and Store have a third format for loading and storing multiple floating point registers.

Four additional instructions perform Fix and Float conversion between the standard single precision integer and single or double precision floating point formats.

Because the 8/32-C uses a 32-bit word, single precision numbers are the equivalent of 7.2 decimal digits. Double precision numbers are the equivalent of 16.7 decimal digits.

Speeds for single precision floating point arithmetic are 0.40 microsecond for Compare Register, 1.00 microsecond for Add Register, 1.75 microseconds for Multiply Register, and 3.60 microseconds for Divide Register. Double precision arithmetic times are 0.60 microsecond for Compare Register Double, 1.04 microseconds for Add Register Double, 2.50 microseconds for Multiply Register Double, and 6.70 for Divide Register Double. These times are the fastest possible instruction execution times.

Table 1 lists all the floating point arithmetic instructions with execution times.

FEATURES

- Single Precision (7.2 decimal digits)
- Double Precision (16.7 decimal digits)
- Thirty Floating Point Instructions
- Eight 32-Bit Registers (single precision)
- Eight 64-Bit Registers (double precision)

TABLE 1. HIGH PERFORMANCE FLOATING POINT INSTRUCTIONS

Instruction	Mnemonic	Instruction Format	Execution Time, usec.*
SINGLE PRECISION			
Load Floating Point	LE	RX1, RX2, RX3	1.39
Load Floating Point Register	LER	RR	1.04
Load Floating Point Multiple	LME	RX1, RX2, RX3	3.57 + 1.34n
Store Floating Point	STE	RX1, RX3,	2.23
		RX2	2.60
Store Floating Point Multiple	STME	RX1, RX2, RX3	3.59 + 0.90n
Add Floating Point	AE	RX1, RX2, RX3	1.82
Add Floating Point Register	AER	RR	1.00
Subtract Floating Point	SE	RX1, RX2, RX3	1.82
Subtract Floating Point Register	SER	RR	1.00
Multiply Floating Point	ME	RX1, RX2, RX3	2.50
Multiply Floating Point Register	MER	RR	1.75
Divide Floating Point	DE	RX1, RX2, RX3	4.45
Divide Floating Point Register	DER	RR	3.60
Compare Floating Point	CE	RX1, RX2, RX3	1.45
Compare Floating Point Register	CER	RR	0.60
Fix Register	FXR	RR	5.35
Float Register	FLR	RR	2.00
DOUBLE PRECISION			
Load Double Precision Floating Point	LD	RX1, RX3	2.91
		RX2	3.28
Load Register Double Precision Floating Point	LDR	RR	1.04
Load Multiple Double Precision Floating Point	LMD	RX1, RX2, RX3	3.69 + 2.19n
Store Double Precision Floating Point	STD	RX1, RX2	2.75
		RX3	2.81
Store Multiple Double Precision Point	STMD	RX1, RX2, RX3	4.50 + 1.80n
Add Double Precision Floating Point	AD	RX1, RX3	3.38
		RX2	3.75
Add Register Double Precision Floating Point	ADR	RR	1.04
Subtract Double Precision Floating Point	SD	RX1, RX3	3.38
		RX2	3.75
Subtract Register Double Precision Floating Point	SDR	RR	1.04
Multiply Double Precision Floating Point	MD	RX1, RX3	4.90
		RX2	5.30
Multiply Register Double Precision Floating Point	MDR	RR	2.50
Divide Double Precision Floating Point	DD	RX1, RX3	9.20
		RX2	9.65
Divide Register Double Precision Floating Point	DDR	RR	6.70
Compare Double Precision Floating Point	CD	RX1, RX3	3.00
		RX2	3.40
Compare Register Double Precision Floating Point	CDR	RR	0.60
Fix Register Double Precision	FXDR	RR	8.10
Float Register Double Precision	FLDR	RR	2.00

*Execution times vary depending on the data in the operands in many cases and on the instruction's location in the lookahead stack for memory referencing instructions. In all cases, the listed time for an instruction is the fastest execution time. The following factors can be used to adjust the execution times:

- Normalize result (Add, Subtract, Multiply, Divide, Float, Load) — 100 nanoseconds per hexadecimal digit shifted.
- Equalize exponents (Add, Subtract) — 100 nanoseconds per hexadecimal digit shifted.
- Data with alternate 1's and 0's (Multiply only) - can increase time by up to 700 nanoseconds for single precision operands and by 1600 nanoseconds for double precision operands.
- Position of instruction in lookahead stack (all memory referencing instructions) — can increase execution time by 400 nanoseconds (maximum), if the instruction read causes the stack to try to refill from memory, or if the stack is already being filled from memory.

FLOATING POINT DATA FORMATS

The data formats for floating point operands are based on hexadecimal digits. Each operand consists of a sign, an exponent, and a fraction. The sign is one bit that designates the sign of the fraction: zero (0) for positive fraction and one (1) for negative fraction. The exponent is a 7-bit field that expresses the floating point number's exponent in hexadecimal excess 64 notation. For example, 64 (x '40') in the exponent field represents an exponent of zero, 63 (x '3F') in the exponent field represents an exponent of minus one (−1), and 65 (x '41') in the exponent field represents an exponent of plus one (+1). The fraction consists of six (single precision) or fourteen (double precision) hexadecimal digits expressed in absolute form.

The exponent of true zero (all zeros in fraction) is zero. The sign of true zero is always zero (positive). Figure 1 illustrates the floating point formats.

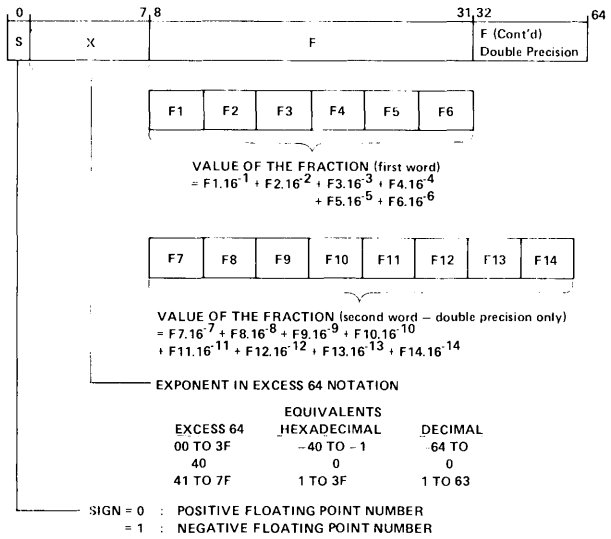


FIGURE 1. FLOATING POINT ARITHMETIC DATA FORMATS

Because of the hexadecimal format, normalized numbers can have up to three leading zeros in the fraction. Minimum precision for single precision operands is equivalent to 21 bits in binary format or 6.2 decimal digits. Minimum precision for double precision operands is equivalent to 53 bits binary format or 15.9 decimal digits.

Results of all floating point operations including the Float instructions are normalized in hexadecimal format. All Load instructions except Load Multiple normalize numbers transferred into the floating point registers. Other floating point instructions assume normalized operands.

The result of a single precision floating point arithmetic operation is rounded up rather than truncated. The result is calculated to 7 hexadecimal digits (the seventh digit is called a guard digit). If the seventh hexadecimal digit is equal to 8 or above, a one (1) is added to the sixth hexadecimal digit. The result of a double precision arithmetic operation is simply truncated.

CONDITION CODES

The Condition Code (CC) bits are set for the floating point arithmetic operations and for Fix and Float instructions to indicate the characteristics of the result. They are:

- Result is zero
- Result is less than zero
- Result is greater than zero
- Exponent overflow (greater than +63), result is negative
- Exponent overflow (greater than +63), result is positive
  Result is forced to maximum value (all 1's).
- Exponent underflow (less than −64). Result is forced to zero.
  Divide has one additional condition code setting:
- Divisor equal to zero.

For Fix instructions, Exponent Underflow does not occur. For Float instructions, neither Exponent Overflow nor Underflow occurs. All the Load instructions except the Load Multiple instructions use the same condition codes as the arithmetic instructions, but Exponent Overflow does not occur. Load Multiple instructions leave the condition codes unchanged from the previous operation.

All the Store instructions also leave the condition codes unchanged from the previous operations.

The Compare instructions use the condition code settings to indicate the following characteristics of the result:

- Operands are equal.
- First operand is less than second operand.
- First operand is greater than second operand.

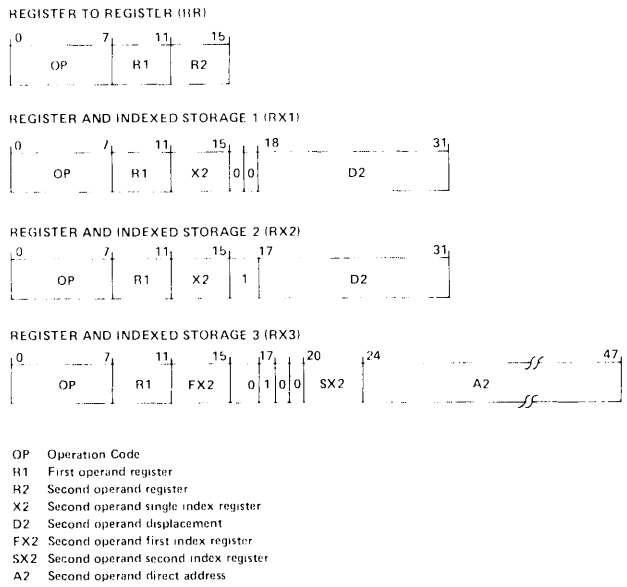
All Exponent Underflow or Overflow conditions except as the result of a Fix instruction generate an Arithmetic Fault Interrupt. A Fix instruction Exponent Overflow condition does not generate an interrupt.

INSTRUCTION FORMATS

The M83-111 High Performance Floating Point option uses four register formats: Register-to-Register (RR), Register and Indexed Storage 1 (RX1), Register and Indexed Storage 2 (RX2), and Register and Indexed Storage 3 (RX3). The Fix and Float instructions use only the RR format. All other instructions use all four formats. Figure 2 illustrates the instruction formats.

In the RR format except for Fix and Float instructions, R1 and R2 select floating point registers. Registers are even numbered for both single and double precision: 0, 2, 4, 6, 8, A, C, and E. For the Fix instructions, R1 selects a general purpose register and R2 selects a floating point register. For the Float instructions, R1 selects a floating point register and R2 selects one of the general purpose registers.

For the RX1, RX2, and RX3 instruction formats, R1 selects a floating point register that contains one operand. The second operand resides in memory. The contents of the general purpose register X2 is used as a base added to the D2 displacement to form the memory address of the second operand. In the RX3 format, the contents of two general purpose registers can be used with A2 to calculate the memory address of the second operand.



OP Operation Code
R1 First operand register
R2 Second operand register
X2 Second operand single index register
D2 Second operand displacement
FX2 Second operand first index register
SX2 Second operand second index register
A2 Second operand direct address

FIGURE 2. FLOATING POINT INSTRUCTION FORMATS

SOFTWARE SUPPORT

The OS/32MT02 Multitasking Operating System and the FORTRAN VI compiler support the M83-111 High Performance Floating Point option.

OS/32MT is a real-time, event-driven operating system. It provides fast real-time response in the foreground. The background can be used for batch processing or to develop programs for on-line tasks or applications computation.

The background can be used for batch processing or to develop programs for on-line tasks or applications computation.

OS/32MT supports three file structures, 255 levels of task priority, and dynamic memory segmentation and relocation.

FORTRAN VI is a full ANSI standard language implementation with full Purdue/ISA extensions for real-time processing. The compiler includes a re-entrant run-time library. It can handle large arrays and programs.

MANUAL CONTROL

The Hexadecimal Display Panel (see Figure 3) can display the contents of the floating point registers. The FLT key selects the single precision floating point register set if it is preceded by depressing the FN (Function) Key with number 2. The FLT key preceded by depressing FN key with number 3 selects the double precision floating point register set for display. The hexadecimal keys select which register is displayed. For double precision, odd numbers select the least significant 32-bit register and even numbers the most significant 32-bit register. For single precision, only even numbers have meaning and depressing an odd number selects the next lower even number register. Registers are numbered, 0, 2, 4, 6, 8, A, C, and E.

INTERDATA PRODUCT NUMBER

M83-111 High Performance Floating Point Option for the Model 8/32

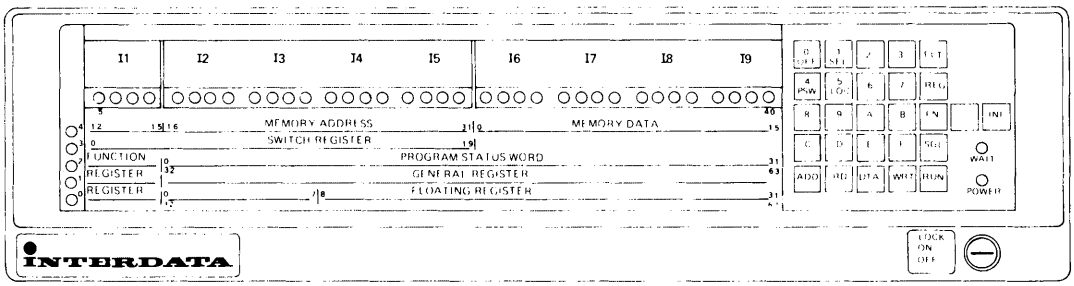


FIGURE 3. HEXADECIMAL DISPLAY PANEL

Printed in U.S.A.

INTERDATA
Subsidiary of PERKIN ELMER • Oceanport New Jersey 07757
9761033