Advanced Micro Devices

Algorithm Details for The Am9511 Arithmetic Processing Unit

By Richard O. Parker and Joseph H. Kroeger

AM-PUB072

TABLE OF CONTENTS

INTRODUCTION

The Am9511 APU is a complete, high performance, complex arithmetic processor contained within a single chip. It is designed to enhance the number manipulation capability of a wide variety of processor systems. It includes not only floating-point operations but fixed-point as well; not only basic add, subtract, multiply and divide operations, but a group of transcendental derived functions plus control and conversion commands as well. This Application Brief provides detailed descriptions of all the commands that can be executed by the Am9511 and indicates the error performance of the derived functions.

The Am9511 is packaged in a standard, 24 pin, dual in-line package with .6 inch between rows. Figure 1 shows the package pin assignments. Details on the operation of each interface pin will be found in the data sheet.

The block diagram in Figure 2 shows the internal structure of the APU. The part is addressed as two ports selected by the C/$\overline{D}$ control line. When C/$\overline{D}$ is high (Control Port), a read op-

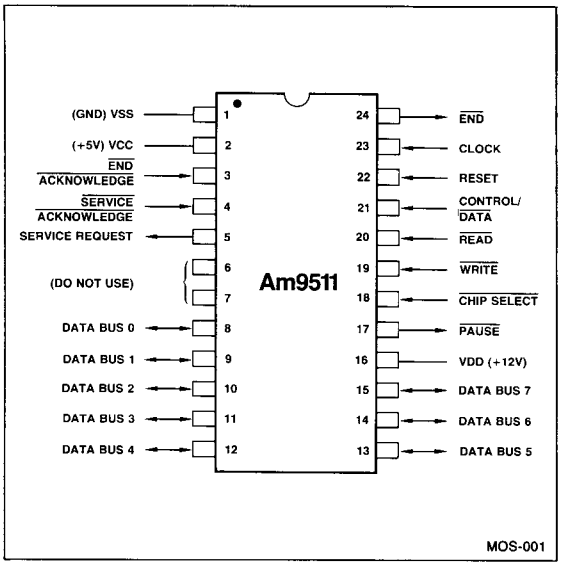


Figure 1. Connection Diagram.

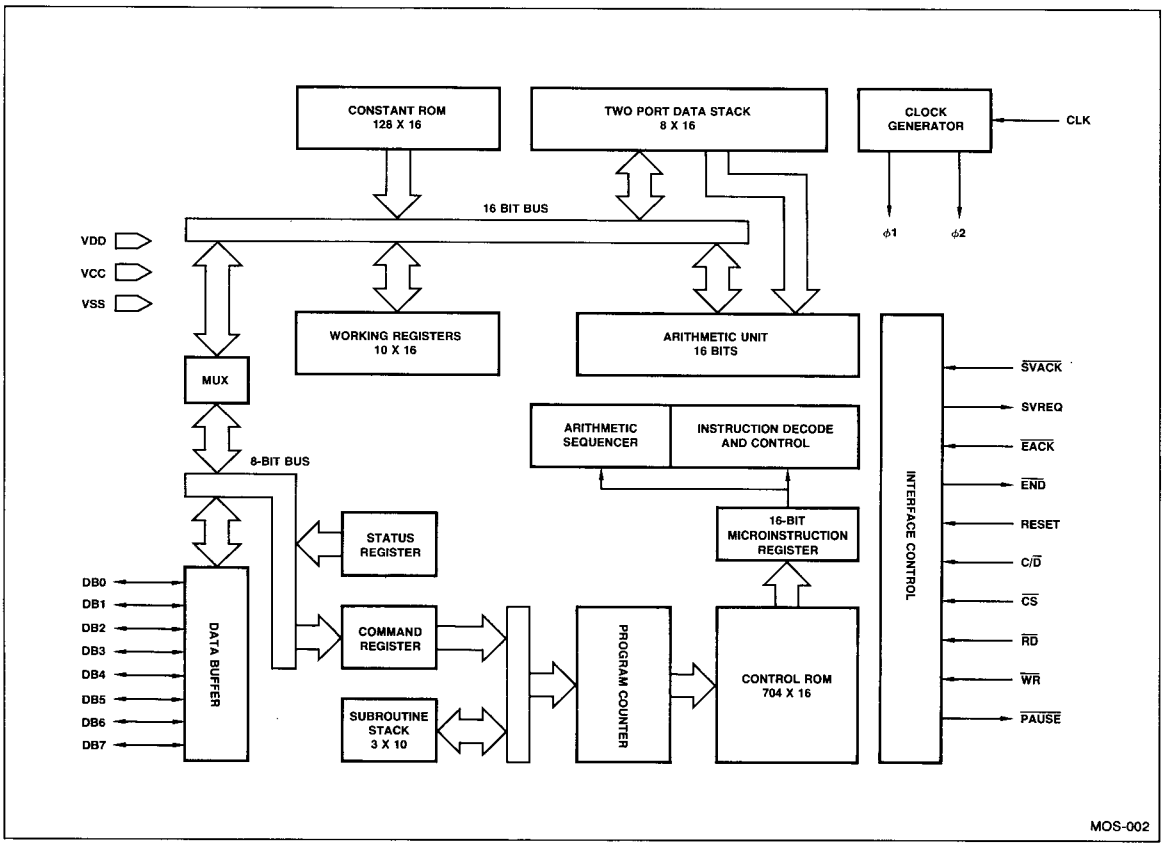


Figure 2. Arithmetic Processing Unit Block Diagram.

eration accesses the status register and a write operation enters a command. When C/$\overline{D}$ is low (Data Port), a read operation accesses data from the top of the data stack and a write operation enters data into the top of the data stack.

Data Formats

The APU executes both 16- and 32-bit fixed-point operations. All fixed-point operands and results are represented as binary two's complement integer values. The 16-bit format can express numbers with a range of −32,768 to +32,767. The 32-bit format can express numbers with a range of −2,147,483,648 to +2,147,483,647.

The floating-point format uses a 32-bit word with fields as shown in Figure 3. The most significant bit (bit 31) indicates the sign of the mantissa. The next seven bits form the exponent and the remaining 24 bits form the mantissa value.

The exponent of the base 2 is an unbiased two's complement number with a range of −64 to +63. The mantissa is a sign-magnitude number with an assumed binary point just to the left of the most significant mantissa bit (bit 23). All floating-point values must be normalized, which makes bit 23 always equal to 1 except when representing a value of zero. The number Zero is represented with binary zeros in all 32 bit positions.

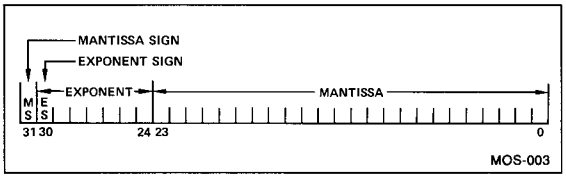


Figure 3. Floating Point Format.

Status Register

The Am9511 Status register format is shown in Figure 4. When the Busy bit (bit 7) is high, the APU is processing a previously entered command and the balance of the Status register should not be considered valid. When the Busy bit is low, the operation is complete and the other status bits are valid.

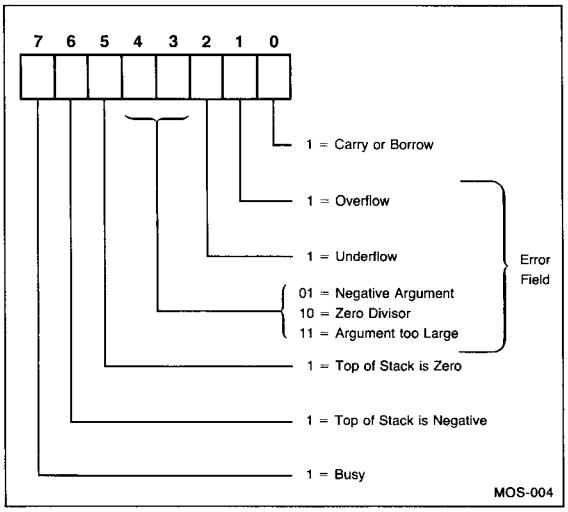


Figure 4. Status Register.

Data Stack

Figure 5 shows the two logical organizations of the internal data stack. It operates as a true push-down stack or FILO stack. That is, the data first written in will be the data last read out. Within each stack entry, the least significant byte is entered first and retrieved last.

Figure 6 shows a typical sequence for 32 bit operations. 6a represents the stack prior to entry of data. 6b shows the stack following entry of the LS Byte of operand C. 6c illustrates the stack contents following the entry of four bytes of operand C. When operands C, B and A are all fully entered the stack appears as in 6e. If a command is then issued, to add B to A for example, the stack contents look like 6f where R is the result of B + A. When the first (MSB) byte of R is removed the stack appears as in 6g. 6h shows the stack following the complete retrieval or R. An even number of bytes should always be transferred for any data operation.

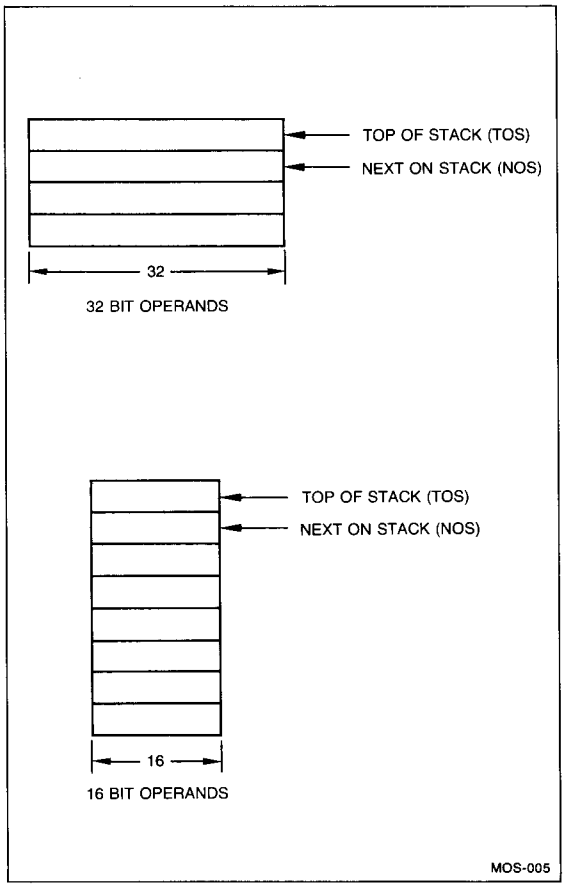


Figure 5. Stack Configurations.

Command Format

Each command executed by the APU is specified by a single byte with the format shown in Figure 7. Bits 0 through 4 indicate the operation to be performed. Bits 5 and 6 specify the data format. Bit 7 is used to control the Service Request interface line. When bit 7 is a one, the SVREQ output will go true when the execution of the command is complete.

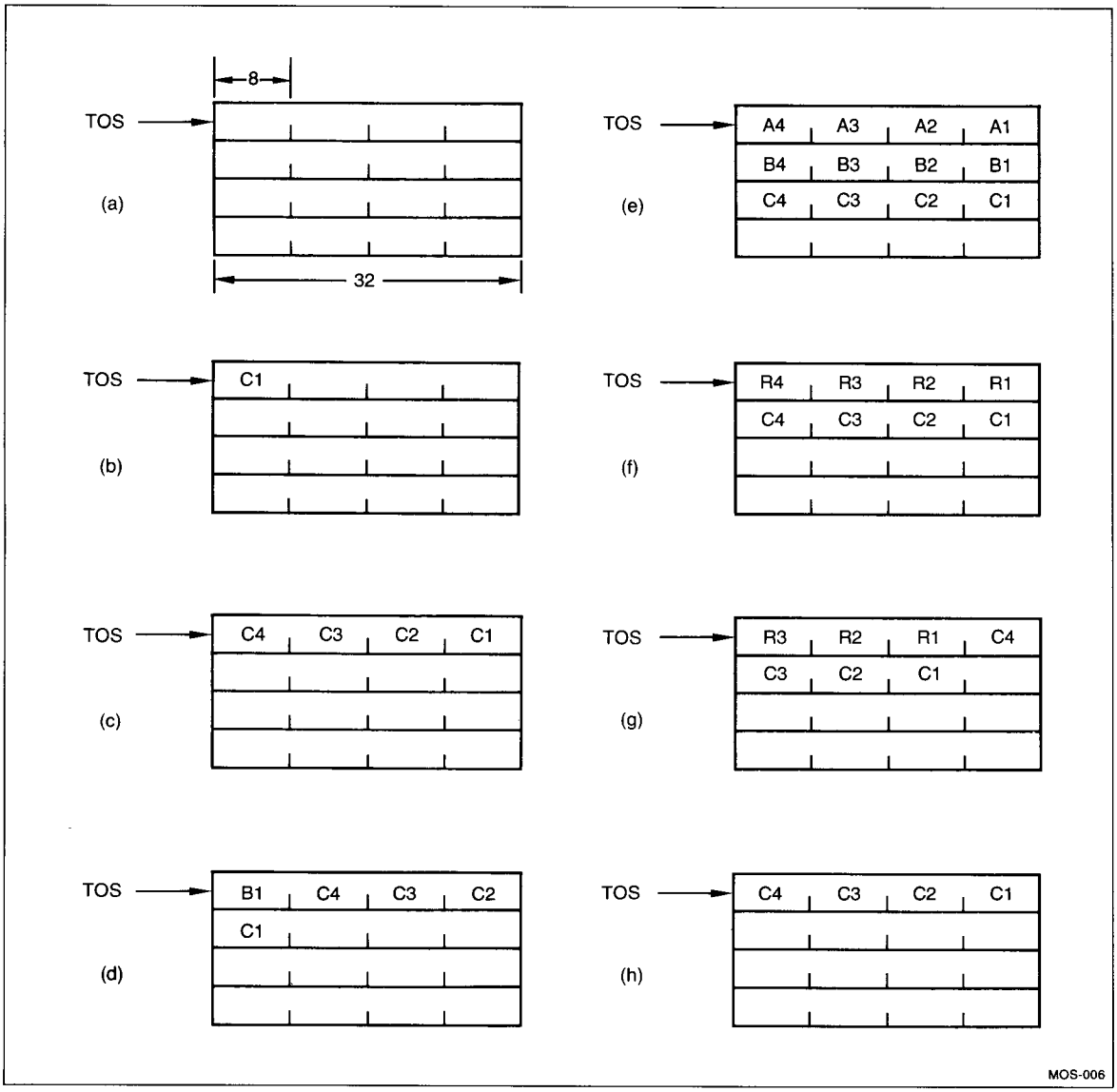


Figure 6. Stack Data Sequence Example.

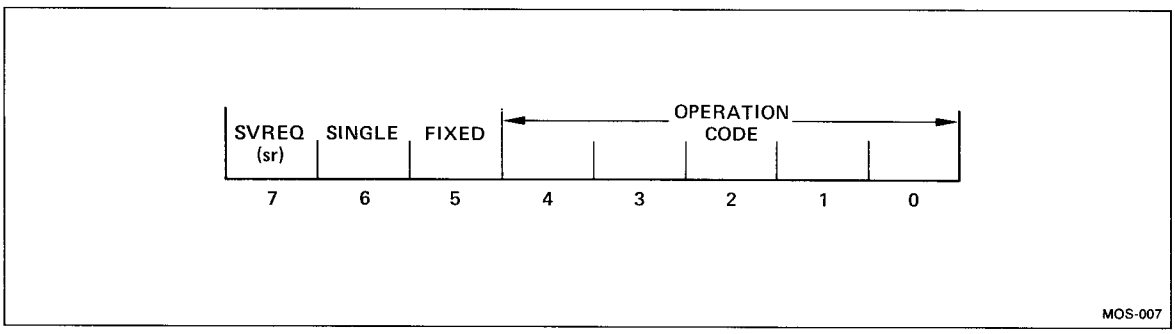


Figure 7. Command Format.

Command Mnemonic	Hex Code (sr = 1)	Hex Code (sr = 0)	Execution Cycles	Summary Description
16-BIT FIXED-POINT OPERATIONS				
SADD	EC	6C	16-18	Add TOS to NOS. Result to NOS. Pop Stack.
SSUB	ED	6D	30-32	Subtract TOS from NOS. Result to NOS. Pop Stack.
SMUL	EE	6E	84-94	Multiply NOS by TOS. Lower result to NOS. Pop Stack.
SMUU	F6	76	80-98	Multiply NOS by TOS. Upper result to NOS. Pop Stack.
SDIV	EF	6F	84-94	Divide NOS by TOS. Result to NOS. Pop Stack.
32-BIT FIXED-POINT OPERATIONS				
DADD	AC	2C	20-22	Add TOS to NOS. Result to NOS. Pop Stack.
DSUB	AD	2D	38-40	Subtract TOS from NOS. Result to NOS. Pop Stack.
DMUL	AE	2E	194-210	Multiply NOS by TOS. Lower result to NOS. Pop Stack.
DMUU	B6	36	182-218	Multiply NOS by TOS. Upper result to NOS. Pop Stack.
DDIV	AF	2F	196-210	Divide NOS by TOS. Result to NOS. Pop Stack.
32-BIT FLOATING-POINT PRIMARY OPERATIONS				
FADD	90	10	54-368	Add TOS to NOS. Result to NOS. Pop Stack.
FSUB	91	11	70-370	Subtract TOS from NOS. Result to NOS. Pop Stack.
FMUL	92	12	146-168	Multiply NOS by TOS. Result to NOS. Pop Stack.
FDIV	93	13	154-184	Divide NOS by TOS. Result to NOS. Pop Stack.
32-BIT FLOATING-POINT DERIVED OPERATIONS				
SQRT	81	01	782-870	Square Root of TOS. Result to TOS.
SIN	82	02	3796-4808	Sine of TOS. Result to TOS.
COS	83	03	3840-4878	Cosine of TOS. Result to TOS.
TAN	84	04	4894-5886	Tangent of TOS. Result to TOS.
ASIN	85	05	6230-7938	Inverse Sine of TOS. Result to TOS.
ACOS	86	06	6304-8284	Inverse Cosine of TOS. Result to TOS.
ATAN	87	07	4992-6536	Inverse Tangent of TOS. Result to TOS.
LOG	88	08	4474-7132	Common Logarithm of TOS. Result to TOS.
LN	89	09	4298-6956	Natural Logarithm of TOS. Result to TOS.
EXP	8A	0A	3794-4878	e raised to power in TOS. Result to TOS.
PWR	8B	0B	8290-12032	NOS raised to power in TOS. Result to NOS. Pop Stack.
DATA AND STACK MANIPULATION OPERATIONS				
NOP	80	00	4	No Operation. Clear or set SVREQ.
FIXS	9F	1F	90-214	Convert TOS from floating point format to fixed point format.
FIXD	9E	1E	90-336	
FLTS	9D	1D	62-156	Convert TOS from fixed point format to floating point format.
FLTD	9C	1C	56-342	
CHSS	F4	74	22-24	Change sign of fixed point operand on TOS.
CHSD	B4	34	26-28	
CHSF	95	15	16-20	Change sign of floating point operand on TOS.
PTOS	F7	77	16	Push stack. Duplicate NOS in TOS.
PTOD	B7	37	20	
PTOF	97	17	20	
POPS	F8	78	10	Pop stack. Old NOS becomes new TOS. Old TOS rotates to bottom.
POPD	B8	38	12	
POPF	98	18	12	
XCHS	F9	79	18	Exchange TOS and NOS.
XCHD	B9	39	26	
XCHF	99	19	26	
PUPI	9A	1A	16	Push floating point constant π onto TOS. Previous TOS becomes NOS.

Figure 8.

ALGORITHM DISCUSSION

Computer approximations of transcendental functions are often based on some form of polynomial equation, such as:

$$F(X) = A_0 + A_1X + A_2X^2 + A_3X^3 + A_4X^4 \ldots . \quad (1\text{-}1)$$

The primary shortcoming of an approximation in this form is that it typically exhibits very large errors when the magnitude of $|X|$ is large, although the errors are small when $|X|$ is small. With polynomials in this form, the error distribution is markedly uneven over any arbitrary interval.

Fortunately, a set of approximating functions exists that not only minimizes the maximum error but also provides an even distribution of errors within the selected data representation interval. These are known as Chebyshev Polynomials and are based upon cosine functions.[1,2] These functions are defined as follows:

$$T_n(X) = \text{Cos } n\theta; \text{ where } n = 0,1,2 \ldots \quad (1\text{-}2)$$
$$\theta = \text{Cos}^{-1}X$$

The various terms of the Chebyshev series can be computed as shown below:

$$T_0(X) = \text{Cos } (0 \cdot \theta) = \text{Cos } (0) = 1 \quad (1\text{-}4)$$
$$T_1(X) = \text{Cos } (\text{Cos}^{-1}X) = X \quad (1\text{-}5)$$
$$T_2(X) = \text{Cos } 2\theta = 2\text{Cos}^2\,\theta - 1 = 2\text{Cos}^2\,(\text{Cos}^{-1}X) - 1 = 2X^2 - 1 \quad (1\text{-}6)$$

In general, the next term in the Chebyshev series can be recursively derived from the previous term as follows:

$$T_n(X) = 2X\left[T_n{-}1(X)\right] - T_n{-}2\,(X);\ n \geq 2 \quad (1\text{-}7)$$

The terms T_3, T_4, T_5 and T_6 are given below for reference:

$$T_3 = 4X^3 - 3X \quad (1\text{-}8)$$
$$T_4 = 8X^4 - 8X^2 + 1 \quad (1\text{-}9)$$
$$T_5 = 16X^5 - 20X^3 + 5X \quad (1\text{-}10)$$
$$T_6 = 32X^6 - 48X^4 + 18X^2 - 1 \quad (1\text{-}11)$$

Chebyshev polynomials can be directly substituted for corresponding terms of a power series expansion by simple algebraic manipulation:

$$1 = T_0 \quad (1\text{-}12)$$
$$X = T_1 \quad (1\text{-}13)$$
$$X^2 = 1/2\,(T_0 + T_2) \quad (1\text{-}14)$$
$$X^3 = 1/4\,(3T_1 + T_3) \quad (1\text{-}15)$$
$$X^4 = 1/8\,(3T_0 + 4T_2 + T_4) \quad (1\text{-}16)$$
$$X^5 = 1/16\,(10T_1 + 5T_3 + T_5) \quad (1\text{-}17)$$
$$X^6 = 1/32\,(10T_0 + 15T_2 + 6T_4 + T_6) \quad (1\text{-}18)$$

Each of the derived functions except square root implemented in the Am9511 APU has been reduced to Chebyshev polynomial form. A sufficient number of terms has been used to provide a mean relative error of about one part in 10^7.

Each of the functions is implemented as a three-step process. The first step involves range reduction. That is, the input argument to the function is transformed to fall within a range of values for which the function can compute a valid result. For example, since functions like sine and cosine are periodic for multiples of $\pi/2$ radians, input arguments for these functions are converted to lie within the range of $-\pi/2$ to $+\pi/2$. Processing of the range-reduced input argument according to the appropriate Chebyshev expansion is done in the second step. The third step includes any necessary post processing of the result, such as sign correction in sine or cosine for a particular quadrant. Range reduction and post processing are unique to each of the functions, while processing the Chebyshev expansion is performed by an algorithm that is common to all functions.

DERIVED FUNCTION ERROR PERFORMANCE

Since each of the derived functions is an approximation of the true function, results computed by the Am9511 are not always exact. In order to more comprehensively quantify the error performance of the component, the following graphs have been prepared. Each function has been executed with a statistically significant number of diverse data values, spanning the allowable input data range, and resulting errors have been tabulated. Absolute errors (that is, the number of bits in error) have been converted to relative errors according to the following equation:

$$\text{Relative Error} = \frac{\text{Absolute Error}}{\text{True Result}}$$

This conversion permits the error to be viewed with respect to the magnitude of the true result. This provides a more objective measurement of error performance since it directly translates to a measure of significant digits of algorithm accuracy.

For example, if a given absolute error is 0.001 and the true result is also 0.001, it is clear that the relative error is equal to 1.0 (which implies that even the first significant digit of the result is wrong). However, if the same absolute error is computed for a true result of 10000.0, then the first six significant digits of the result are correct (0.001/10000 = 0.0000001).

Each of the following graphs was prepared to illustrate relative algorithm error as a function of input data range. Natural Logarithm is the only exception; since logarithms are typically additive, absolute error is plotted for this function.

Two graphs have not been included in the following figures: common logarithms and the power function (X^Y). Common logarithms are computed by multiplication of the natural logarithm by the conversion factor 0.43429448 and the error function is therefore the same as that for natural logarithm. The power function is realized by combination of natural log and exponential functions according to the equation:

$$X^Y = e^{Y\text{Ln}x}.$$

The error for the power function is a combination of that for the logarithm and exponential functions. Specifically, the relative error for PWR is expressed as follows:

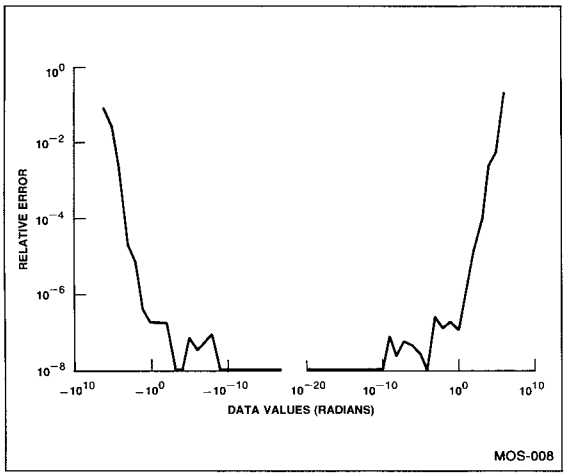
$$|RE_{PWR}| = |RE_{EXP}| + |X\,(AE_{LN})|$$

where:

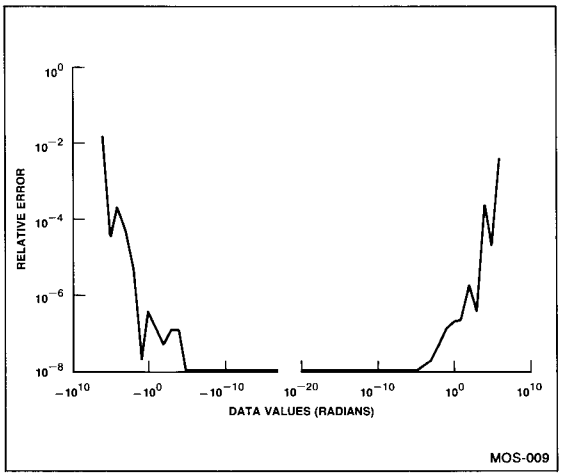
RE_{PWR} = relative error for power function
RE_{EXP} = relative error for exponential function
AE_{LN} = absolute error for natural logarithm
X = value of independent variable in X^Y

Notes:

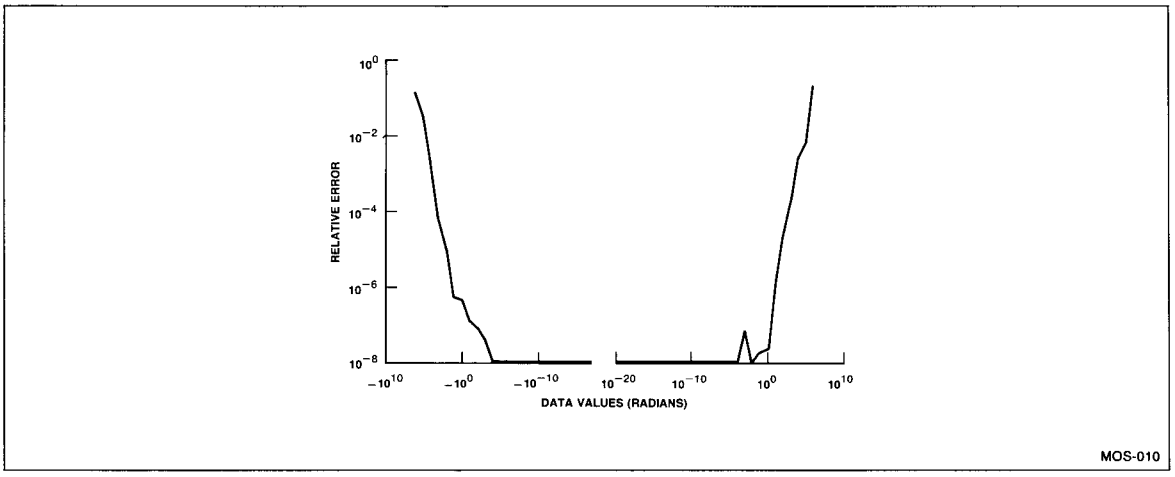
1. Properties of Chebyshev polynomials taken from: Applied Numerical Methods; Carnahan, Luther, Wikes; John Wiley & Sons, Inc.; 1969.
2. Derived function algorithms adapted from: Algorithms for Special Functions (I and II); Numerische Mathematic (1963); Clenshaw, Miller, Woodger.



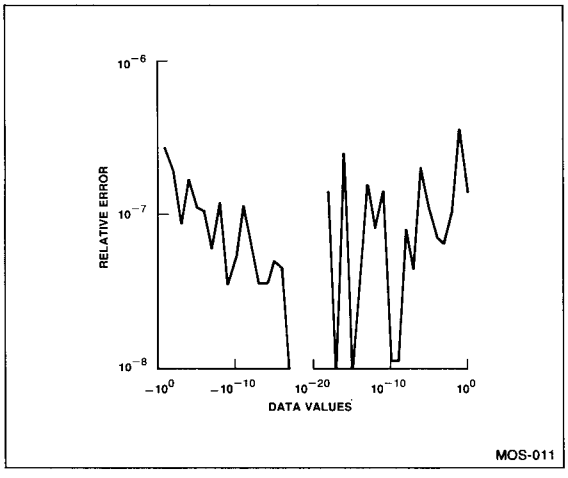
SINE



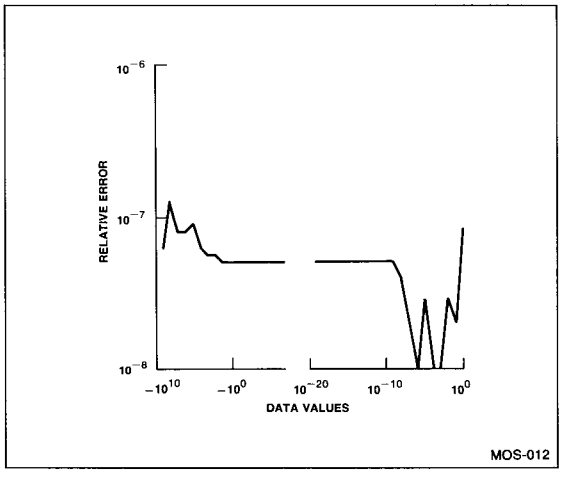
COSINE



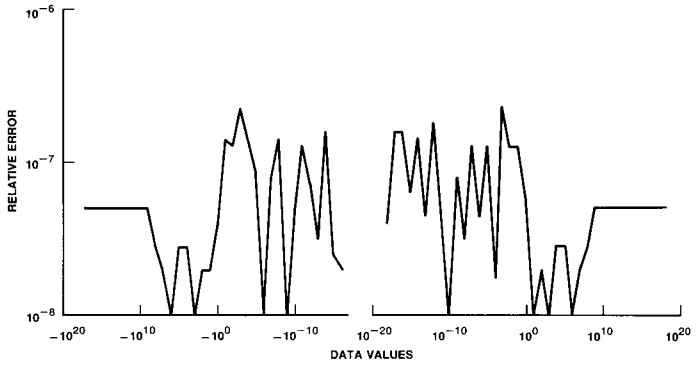
TANGENT



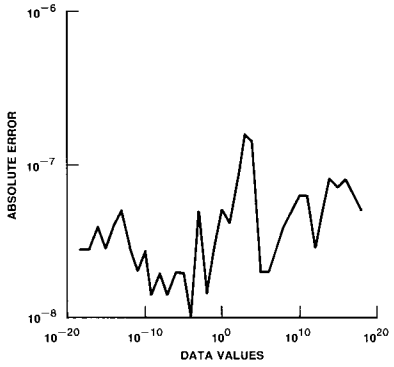
INVERSE SINE



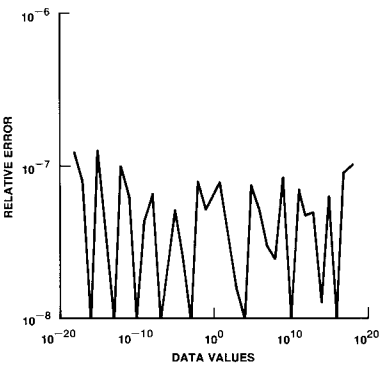
INVERSE COSINE



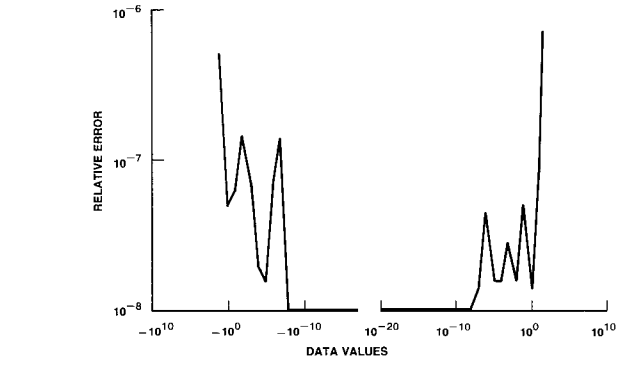
INVERSE TANGENT



NATURAL LOG



SQUARE ROOT



e^x

COMMAND DESCRIPTIONS

This section contains detailed descriptions of the APU commands. They are arranged in alphabetical order by command mnemonic. In the descriptions, TOS means Top Of Stack and NOS means Next On Stack.

All derived functions except Square Root use Chebyshev polynomial approximating algorithms. This approach is used to help minimize the internal microprogram, to minimize the maximum error values and to provide a relatively even distribution of errors over the data range. The basic arithmetic operations are used by the derived functions to compute the various Chebyshev terms. The basic operations may produce error codes in the status register as a result.

Execution times are listed in terms of clock cycles and may be converted into time values by multiplying by the clock period used. For example, an execution time of 44 clock cycles when running at a 4MHz rate translates to 11 microseconds (44 x .25μs = 11μs). Variations in execution cycles reflect the data dependency of the algorithms.

In some operations exponent overflow or underflow may be possible. When this occurs, the exponent returned in the result will be 128 greater or smaller than its true value.

Many of the functions use portions of the data stack as scratch storage during development of the results. Thus previous values in those stack locations will be lost. Scratch locations destroyed are listed in the command descriptions and shown with the crossed-out locations in the Stack Contents After diagram.

Figure 8 is a summary of all the Am9511 commands. It shows the hex codes for each command, the mnemonic abbreviation, a brief description and the execution time in clock cycles. The commands are grouped by functional classes.

Figure 9 lists the command mnemonics in alphabetical order.

ACOS	ARCCOSINE	LOG	COMMON LOGARITHM
ASIN	ARCSINE	LN	NATURAL LOGARITHM
ATAN	ARCTANGENT	NOP	NO OPERATION
CHSD	CHANGE SIGN DOUBLE	POPD	POP STACK DOUBLE
CHSF	CHANGE SIGN FLOATING	POPF	POP STACK FLOATING
CHSS	CHANGE SIGN SINGLE	POPS	POP STACK SINGLE
COS	COSINE	PTOD	PUSH STACK DOUBLE
DADD	DOUBLE ADD	PTOF	PUSH STACK FLOATING
DDIV	DOUBLE DIVIDE	PTOS	PUSH STACK SINGLE
DMUL	DOUBLE MULTIPLY LOWER	PUPI	PUSH π
DMUU	DOUBLE MULTIPLY UPPER	PWR	POWER (X^Y)
DSUB	DOUBLE SUBTRACT	SADD	SINGLE ADD
EXP	EXPONENTIATION (e^x)	SDIV	SINGLE DIVIDE
FADD	FLOATING ADD	SIN	SINE
FDIV	FLOATING DIVIDE	SMUL	SINGLE MULTIPLY LOWER
FIXD	FIX DOUBLE	SMUU	SINGLE MULTIPLY UPPER
FIXS	FIX SINGLE	SQRT	SQUARE ROOT
FLTD	FLOAT DOUBLE	SSUB	SINGLE SUBTRACT
FLTS	FLOAT SINGLE	TAN	TANGENT
FMUL	FLOATING MULTIPLY	XCHD	EXCHANGE OPERANDS DOUBLE
FSUB	FLOATING SUBTRACT	XCHF	EXCHANGE OPERANDS FLOATING
		XCHS	EXCHANGE OPERANDS SINGLE

Figure 9. Command Mnemonics in Alphabetical Order.

ACOS

32-BIT FLOATING-POINT INVERSE COSINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	1	0

Hex Coding: 86 with sr = 1
06 with sr = 0

Execution Time: 6304 to 8284 clock cycles

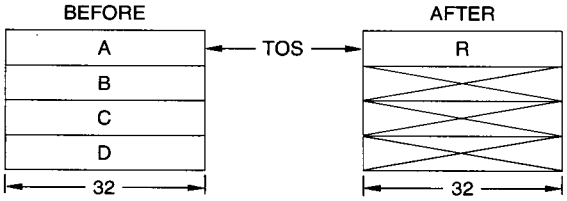
Description:

The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point inverse cosine of A. The result R is a value in radians between 0 and π. Initial operands A, B, C and D are lost. ACOS will accept all input data values within the range of -1.0 to $+1.0$. Values outside this range will return an error code of 1100 in the status register.

Accuracy: ACOS exhibits a maximum relative error of 2.0×10^{-7} over the valid input data range.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



ASIN

32-BIT FLOATING-POINT INVERSE SINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	0	1

Hex Coding: 85 with sr = 1
05 with sr = 0

Execution Time: 6230 to 7938 clock cycles

Description:

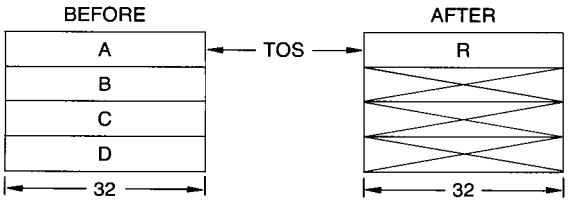
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point inverse sine of A. The result R is a value in radians between $-\pi/2$ and $+\pi/2$. Initial operands A, B, C and D are lost.

ASIN will accept all input data values within the range of -1.0 to $+1.0$. Values outside this range will return an error code of 1100 in the status register.

Accuracy: ASIN exhibits a maximum relative error of 4.0×10^{-7} over the valid input data range.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



ATAN

32-BIT FLOATING-POINT INVERSE TANGENT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	1	1

Hex Coding: 87 with sr = 1
07 with sr = 0

Execution Time: 4992 to 6536 clock cycles

Description:

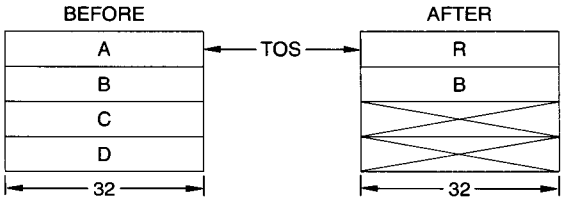
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point inverse tangent of A. The result R is a value in radians between $-\pi/2$ and $+\pi/2$. Initial operands A, C and D are lost. Operand B is unchanged.

ATAN will accept all input data values that can be represented in the floating point format.

Accuracy: ATAN exhibits a maximum relative error of 3.0×10^{-7} over the input data range.

Status Affected: Sign, Zero

STACK CONTENTS



CHSD

32-BIT FIXED-POINT SIGN CHANGE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	0	1	0	0

Hex Coding: B4 with sr = 1
34 with sr = 0

Execution Time: 26 to 28 clock cycles

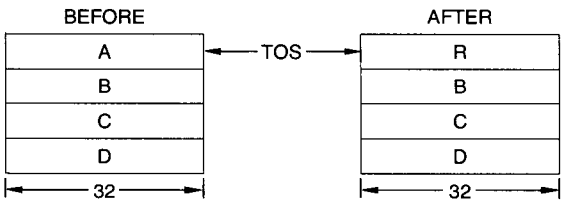
Description:

The 32-bit fixed-point two's complement integer operand A at the TOS is subtracted from zero. The result R replaces A at the TOS. Other entries in the stack are not disturbed.

Overflow status will be set and the TOS will be returned unchanged when A is input as the most negative value possible in the format since no positive equivalent exists.

Status Affected: Sign, Zero, Error Field (overflow)

STACK CONTENTS



CHSF

32-BIT FLOATING-POINT SIGN CHANGE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	1	0	1

Hex Coding: 95 with sr = 1
15 with sr = 0

Execution Time: 16 to 20 clock cycles

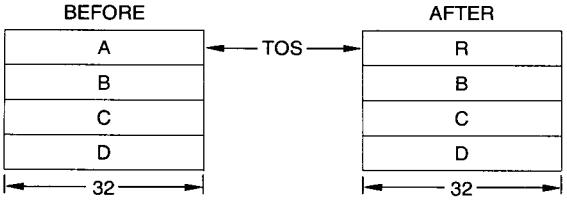
Description:

The sign of the mantissa of the 32-bit floating-point operand A at the TOS is inverted. The result R replaces A at the TOS. Other stack entries are unchanged.

If A is input as zero (mantissa MSB = 0), no change is made.

Status Affected: Sign, Zero

STACK CONTENTS



CHSS

16-BIT FIXED-POINT SIGN CHANGE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	0	1	0	0

Hex Coding: F4 with sr = 1
74 with sr = 0

Execution Time: 22 to 24 clock cycles

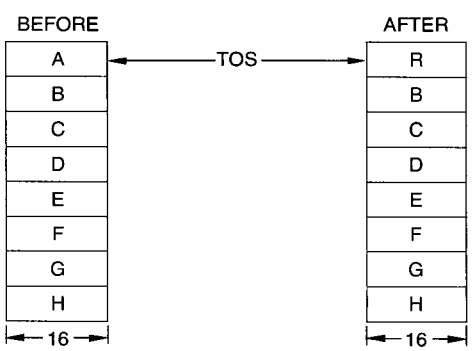
Description:

16-bit fixed-point two's complement integer operand A at the TOS is subtracted from zero. The result R replaces A at the TOS. All other operands are unchanged.

Overflow status will be set and the TOS will be returned unchanged when A is input as the most negative value possible in the format since no positive equivalent exists.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



COS

32-BIT FLOATING-POINT COSINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	1	1

Hex Coding: 83 with sr = 1
03 with sr = 0

Execution Time: 3840 to 4878 clock cycles

Description:

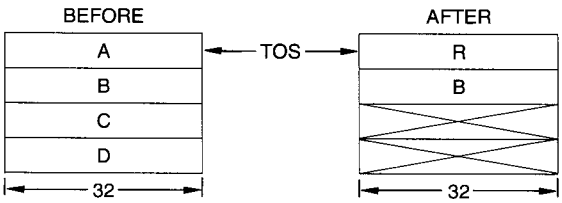
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point cosine of A. A is assumed to be in radians. Operands A, C and D are lost. B is unchanged.

The COS function can accept any input data value that can be represented in the data format. All input values are range reduced to fall within an interval of $-\pi/2$ to $+\pi/2$ radians.

Accuracy: COS exhibits a maximum relative error of 5.0 x 10^{-7} for all input data values in the range of -2π to $+2\pi$ radians.

Status Affected: Sign, Zero

STACK CONTENTS



DADD

32-BIT FIXED-POINT ADD

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	0	0

Hex Coding: AC with sr = 1
2C with sr = 0

Execution Time: 20 to 22 clock cycles

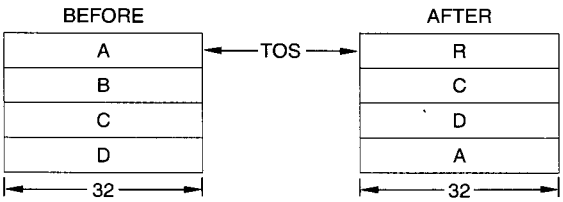
Description:

The 32-bit fixed-point two's complement integer operand A at the TOS is added to the 32-bit fixed-point two's complement integer operand B at the NOS. The result R replaces operand B and the Stack is moved up so that R occupies the TOS. Operand B is lost. Operands A, C and D are unchanged. If the addition generates a carry it is reported in the status register.

If the result is too large to be represented by the data format, the least significant 32 bits of the result are returned and overflow status is reported.

Status Affected: Sign, Zero, Carry, Error Field

STACK CONTENTS



DDIV

32-BIT FIXED-POINT DIVIDE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	1	1

Hex Coding: AF with sr = 1
2F with sr = 0

Execution Time: 196 to 210 clock cycles when A ≠ 0
18 clock cycles when A = 0.

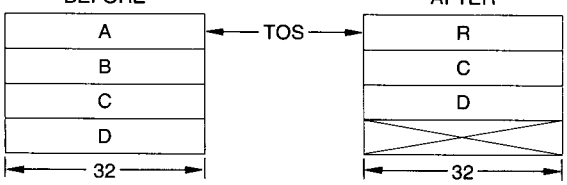
Description:

The 32-bit fixed-point two's complement integer operand B at NOS is divided by the 32-bit fixed-point two's complement integer operand A at the TOS. The 32-bit integer quotient R replaces B and the stack is moved up so that R occupies the TOS. No remainder is generated. Operands A and B are lost. Operands C and D are unchanged.

If A is zero, R is set equal to B and the divide-by-zero error status will be reported. If either A or B is the most negative value possible in the format, R will be meaningless and the overflow error status will be reported.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



DMUL

32-BIT FIXED-POINT MULTIPLY, LOWER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	1	0

Hex Coding: AE with sr = 1
2E with sr = 0

Execution Time: 194 to 210 clock cycles

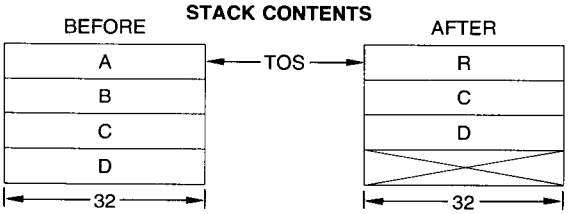
Description:

The 32-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 32-bit fixed-point two's complement integer operand B at the NOS. The 32-bit least significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The most significant half of the product is lost. Operands A and B are lost. Operands C and D are unchanged.

The overflow status bit is set if the discarded upper half was non-zero. If either A or B is the most negative value that can be represented in the format, that value is returned as R and the overflow status is set.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



DMUU

32-BIT FIXED-POINT MULTIPLY, UPPER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	0	1	1	0

Hex Coding: B6 with sr = 1
36 with sr = 0

Execution Time: 182 to 218 clock cycles

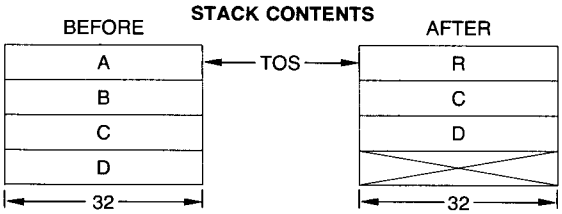
Description:

The 32-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 32-bit fixed-point two's complement integer operand B at the NOS. The 32-bit most significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The least significant half of the product is lost. Operands A and B are lost. Operands C and D are unchanged.

If A or B was the most negative value possible in the format, overflow status is set and R is meaningless.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



DSUB

32-BIT FIXED-POINT SUBTRACT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	0	1

Hex Coding: AD with sr = 1
2D with sr = 0

Execution Time: 38 to 40 clock cycles

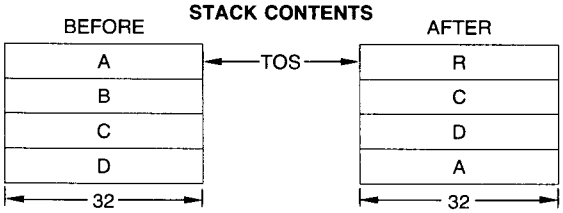
Description:

The 32-bit fixed-point two's complement operand A at the TOS is subtracted from the 32-bit fixed-point two's complement operand B at the NOS. The difference R replaces operand B and the stack is moved up so that R occupies the TOS. Operand B is lost. Operands A, C and D are unchanged.

If the subtraction generates a borrow it is reported in the carry status bit. If A is the most negative value that can be represented in the format the overflow status is set. If the result cannot be represented in the data format range, the overflow bit is set and the 32 least significant bits of the result are returned as R.

Status Affected: Sign, Zero, Carry, Overflow

STACK CONTENTS



EXP

32-BIT FLOATING-POINT e^X

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	1	0

Hex Coding: 8A with sr = 1
0A with sr = 0

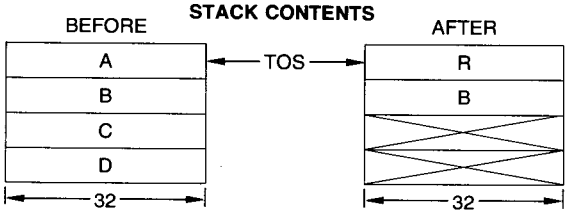
Execution Time: 3794 to 4878 clock cycles for $|A| \leq 1.0 \times 2^5$
34 clock cycles for $|A| > 1.0 \times 2^5$

Description:
The base of natural logarithms, e, is raised to an exponent value specified by the 32-bit floating-point operand A at the TOS. The result R of e^A replaces A. Operands A, C and D are lost. Operand B is unchanged.
EXP accepts all input data values within the range of $-1.0 \times 2^{+5}$ to $+1.0 \times 2^{+5}$. Input values outside this range will return a code of 1100 in the error field of the status register.

Accuracy: EXP exhibits a maximum relative error of 5.0×10^{-7} over the valid input data range.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



FADD

32-BIT FLOATING-POINT ADD

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	0	0

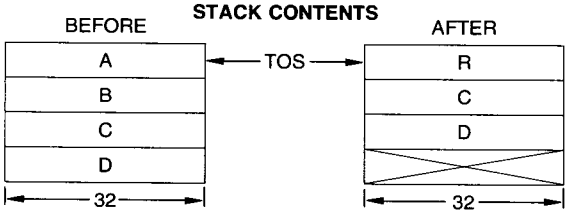
Hex Coding: 90 with sr = 1
10 with sr = 0

Execution Time: 54 to 368 clock cycles for A ≠ 0
24 clock cycles for A = 0

Description:
32-bit floating-point operand A at the TOS is added to 32-bit floating-point operand B at the NOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.
Exponent alignment before the addition and normalization of the result accounts for the variation in execution time. Exponent overflow and underflow are reported in the status register, in which case the mantissa is correct and the exponent is offset by 128.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



FDIV

32-BIT FLOATING-POINT DIVIDE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	1	1

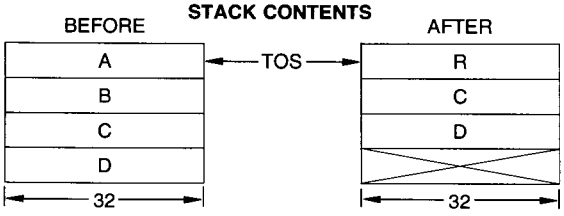
Hex Coding: 93 with sr = 1
13 with sr = 0

Execution Time: 154 to 184 clock cycles for A ≠ 0
22 clock cycles for A = 0

Description:
32-bit floating-point operand B at NOS is divided by 32-bit floating-point operand A at the TOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.
If operand A is zero, R is set equal to B and the divide-by-zero error is reported in the status register. Exponent overflow or underflow is reported in the status register, in which case the mantissa portion of the result is correct and the exponent portion is offset by 128.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



FIXD

32-BIT FLOATING-POINT TO 32-BIT FIXED-POINT CONVERSION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	1	0

Hex Coding: 9E with sr = 1
1E with sr = 0

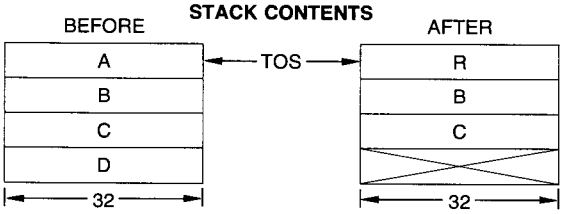
Execution Time: 90 to 336 clock cycles

Description:
32-bit floating-point operand A at the TOS is converted to a 32-bit fixed-point two's complement integer. The result R replaces A. Operands A and D are lost. Operands B and C are unchanged.

If the integer portion of A is larger than 31 bits when converted, the overflow status will be set and A will not be changed. Operand D, however, will still be lost.

Status Affected: Sign, Zero Overflow

STACK CONTENTS



FIXS

32-BIT FLOATING-POINT TO 16-BIT FIXED-POINT CONVERSION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	1	1

Hex Coding: 9F with sr = 1
1F with sr = 0

Execution Time: 90 to 214 clock cycles

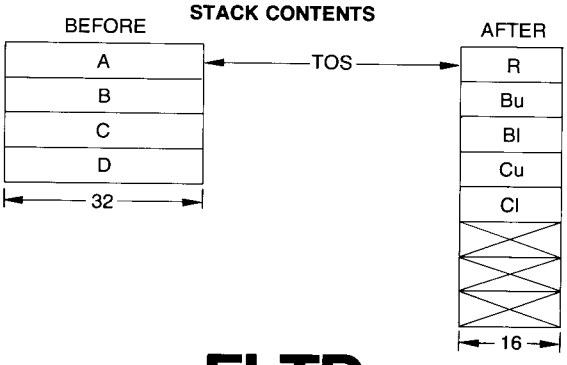
Description:

32-bit floating-point operand A at the TOS is converted to a 16-bit fixed-point two's complement integer. The result R replaces the lower half of A and the stack is moved up by two bytes so that R occupies the TOS. Operands A and D are lost. Operands B and C are unchanged, but appear as upper (u) and lower (l) halves on the 16-bit wide stack if they are 32-bit operands.

If the integer portion of A is larger than 15 bits when converted, the overflow status will be set and A will not be changed. Operand D, however, will still be lost.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



FLTD

32-BIT FIXED-POINT TO 32-BIT FLOATING-POINT CONVERSION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	0	0

Hex Coding: 9C with sr = 1
1C with sr = 0

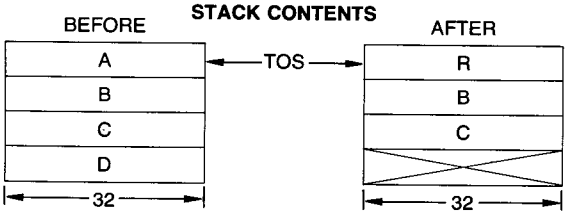
Execution Time: 56 to 342 clock cycles

Description:

32-bit fixed-point two's complement integer operand A at the TOS is converted to a 32-bit floating-point number. The result R replaces A at the TOS. Operands A and D are lost. Operands B and C are unchanged.

Status Affected: Sign, Zero

STACK CONTENTS



FLTS

16-BIT FIXED-POINT TO 32-BIT FLOATING-POINT CONVERSION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	0	1

Hex Coding: 9D with sr = 1
1D with sr = 0

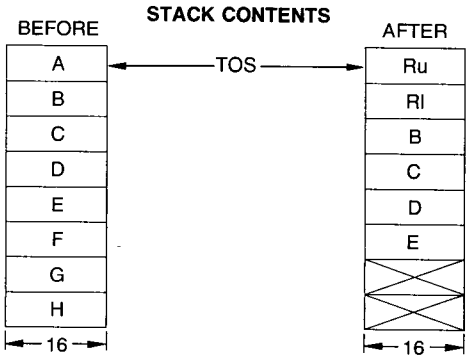
Execution Time: 62 to 156 clock cycles

Description:

16-bit fixed-point two's complement integer A at the TOS is converted to a 32-bit floating-point number. The lower half of the result R (Rl) replaces A, the upper half (Ru) replaces H and the stack is moved down so that Ru occupies the TOS. Operands A, F, G and H are lost. Operands B, C, D and E are unchanged.

Status Affected: Sign, Zero

STACK CONTENTS



FMUL

32-BIT FLOATING-POINT MULTIPLY

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	1	0

Hex Coding: 92 with sr = 1
12 with sr = 0

Execution Time: 146 to 168 clock cycles

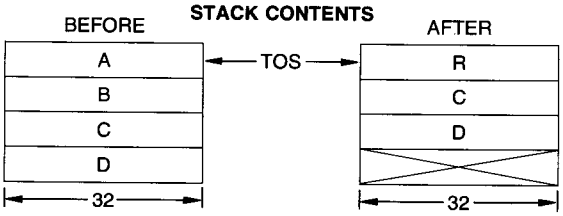
Description:

32-bit floating-point operand A at the TOS is multiplied by the 32-bit floating-point operand B at the NOS. The normalized result R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.

Exponent overflow or underflow is reported in the status register, in which case the mantissa portion of the result is correct and the exponent portion is offset by 128.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



FSUB

32-BIT FLOATING-POINT SUBTRACTION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	0	1

Hex Coding: 91 with sr = 1
11 with sr = 0

Execution Time: 70 to 370 clock cycles for A ≠ 0
26 clock cycles for A = 0

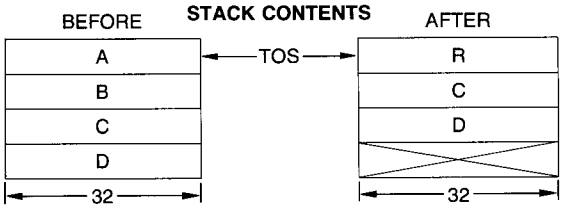
Description:

32-bit floating-point operand A at the TOS is subtracted from 32-bit floating-point operand B at the NOS. The normalized difference R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.

Exponent alignment before the subtraction and normalization of the result account for the variation in execution time.

Exponent overflow or underflow is reported in the status register in which case the mantissa portion of the result is correct and the exponent portion is offset by 128.

Status Affected: Sign, Zero, Error Field (overflow)



LOG

32-BIT FLOATING-POINT COMMON LOGARITHM

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	0	0

Hex Coding: 88 with sr = 1
08 with sr = 0

Execution Time: 4474 to 7132 clock cycles for A > 0
20 clock cycles for A ≤ 0

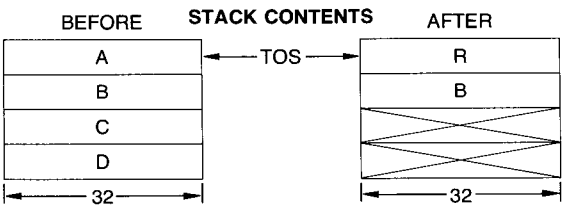
Description:

The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point common logarithm (base 10) of A. Operands A, C and D are lost. Operand B is unchanged.

The LOG function accepts any positive input data value that can be represented by the data format. If LOG of a non-positive value is attempted an error status of 0100 is returned.

Accuracy: LOG exhibits a maximum absolute error of 2.0×10^{-7} for the input range from 0.1 to 10, and a maximum relative error of 2.0×10^{-7} for positive values less than 0.1 or greater than 10.

Status Affected: Sign, Zero, Error Field



LN

32-BIT FLOATING-POINT NATURAL LOGARITHM

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	0	1

Hex Coding: 89 with sr = 1
09 with sr = 0

Execution Time: 4298 to 6956 clock cycles for A > 0
20 clock cycles for A ≤ 0

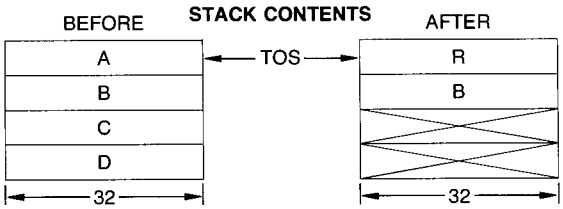
Description:

The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point natural logarithm (base e) of A. Operands A, C and D are lost. Operand B is unchanged.

The LN function accepts all positive input data values that can be represented by the data format. If LN of a non-positive number is attempted an error status of 0100 is returned.

Accuracy: LN exhibits a maximum absolute error of 2×10^{-7} for the input range from e^{-1} to e, and a maximum relative error of 2.0×10^{-7} for positive values less than e^{-1} or greater than e.

Status Affected: Sign, Zero, Error Field



NOP

NO OPERATION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	0	0

Hex Coding: 80 with sr = 1
00 with sr = 0

Execution Time: 4 clock cycles

Description:

The NOP command performs no internal data manipulations. It may be used to set or clear the service request interface line without changing the contents of the stack.

Status Affected: The status byte is cleared to all zeroes.

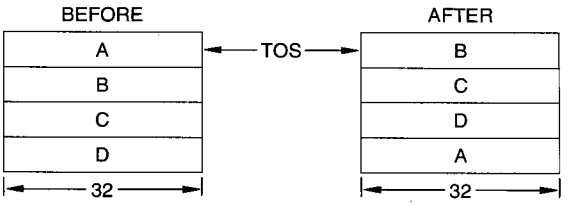
POPD

32-BIT
STACK POP

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	1	0	0	0

Hex Coding: B8 with sr = 1
38 with sr = 0
Execution Time: 12 clock cycles
Description:
The 32-bit stack is moved up so that the old NOS becomes the new TOS. The previous TOS rotates to the bottom of the stack. All operand values are unchanged. POPD and POPF execute the same operation.
Status Affected: Sign, Zero

STACK CONTENTS



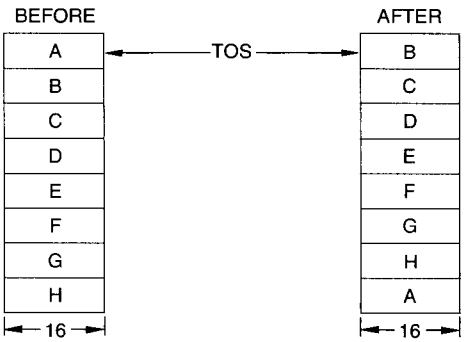
POPS

16-BIT
STACK POP

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	1	0	0	0

Hex Coding: F8 with sr = 1
78 with sr = 0
Execution Time: 10 clock cycles
Description:
The 16-bit stack is moved up so that the old NOS becomes the new TOS. The previous TOS rotates to the bottom of the stack. All operand values are unchanged.
Status Affected: Sign, Zero

STACK CONTENTS



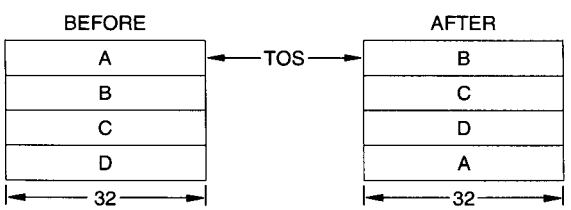
POPF

32-BIT
STACK POP

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	0	0	0

Hex Coding: 98 with sr = 1
18 with sr = 0
Execution Time: 12 clock cycles
Description:
The 32-bit stack is moved up so that the old NOS becomes the new TOS. The old TOS rotates to the bottom of the stack. All operand values are unchanged. POPF and POPD execute the same operation.
Status Affected: Sign, Zero

STACK CONTENTS



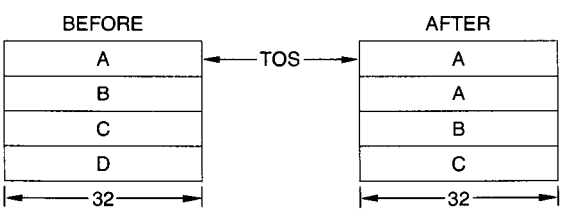
PTOD

PUSH 32-BIT
TOS ONTO STACK

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	0	1	1	1

Hex Coding: B7 with sr = 1
37 with sr = 0
Execution Time: 20 clock cycles
Description:
The 32-bit stack is moved down and the previous TOS is copied into the new TOS location. Operand D is lost. All other operand values are unchanged. PTOD and PTOF execute the same operation.
Status Affected: Sign, Zero

STACK CONTENTS



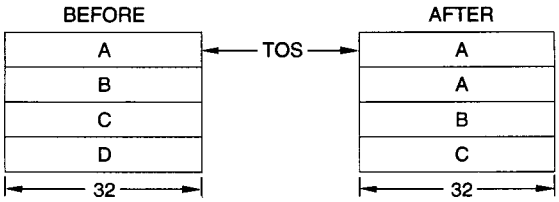
PTOF

PUSH 32-BIT TOS ONTO STACK

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	1	1	1

Hex Coding: 97 with sr = 1
17 with sr = 0
Execution Time: 20 clock cycles
Description:
The 32-bit stack is moved down and the previous TOS is copied into the new TOS location. Operand D is lost. All other operand values are unchanged. PTOF and PTOD execute the same operation.
Status Affected: Sign, Zero

STACK CONTENTS



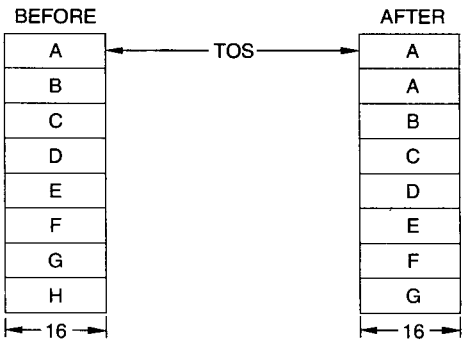
PTOS

PUSH 16-BIT TOS ONTO STACK

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	0	1	1	1

Hex Coding: F7 with sr = 1
77 with sr = 0
Execution Time: 16 clock cycles
Description:
The 16-bit stack is moved down and the previous TOS is copied into the new TOS location. Operand H is lost and all other operand values are unchanged.
Status Affected: Sign, Zero

STACK CONTENTS



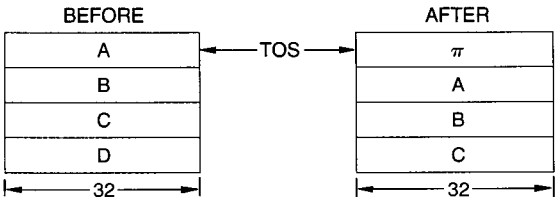
PUPI

PUSH 32-BIT FLOATING-POINT π

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	0	1	0

Hex Coding: 9A with sr = 1
1A with sr = 0
Execution Time: 16 clock cycles
Description:
The 32-bit stack is moved down so that the previous TOS occupies the new NOS location. 32-bit floating-point constant π is entered into the new TOS location. Operand D is lost. Operands A, B and C are unchanged.
Status Affected: Sign, Zero

STACK CONTENTS



PWR

32-BIT FLOATING-POINT X^Y

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	1	1

Hex Coding: 8B with sr = 1
0B with sr = 0

Execution Time: 8290 to 12032 clock cycles

Description:

32-bit floating-point operand B at the NOS is raised to the power specified by the 32-bit floating-point operand A at the TOS. The result R of B^A replaces B and the stack is moved up so that R occupies the TOS. Operands A, B, and D are lost. Operand C is unchanged.

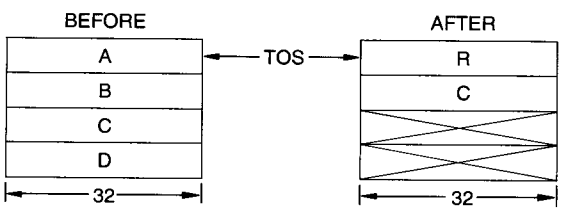
The PWR function accepts all input data values that can be represented in the data format for operand A and all positive values for operand B. If operand B is non-positive an error status of 0100 will be returned. The EXP and LN functions are used to implement PWR using the relationship $B^A = EXP\,[A(LN\ B)]$. Thus if the term $[A(LN\ B)]$ is outside the range of $-1.0 \times 2^{+5}$ to $+1.0 \times 2^{+5}$ an error status of 1100 will be returned. Underflow and overflow conditions can occur.

Accuracy: The error performance for PWR is a function of the LN and EXP performance as expressed by:
$|(\text{Relative Error})_{PWR}| = |(\text{Relative Error})_{EXP} + |A(\text{Absolute Error})_{LN}|$

The maximum relative error for PWR occurs when A is at its maximum value while $[A(LN\ B)]$ is near 1.0×2^5 and the EXP error is also at its maximum. For most practical applications the relative error for PWR will be less than 7.0×10^{-7}.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SADD

16-BIT FIXED-POINT ADD

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	0	0

Hex Coding: EC with sr = 1
6C with sr = 0

Execution Time: 16 to 18 clock cycles

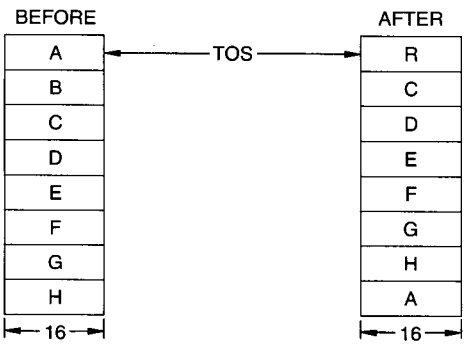
Description:

16-bit fixed-point two's complement integer operand A at the TOS is added to 16-bit fixed-point two's complement integer operand B at the NOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operand B is lost. All other operands are unchanged.

If the addition generates a carry bit it is reported in the status register. If an overflow occurs it is reported in the status register and the 16 least significant bits of the result are returned.

Status Affected: Sign, Zero, Carry, Error Field

STACK CONTENTS



SDIV

16-BIT FIXED-POINT DIVIDE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	1	1

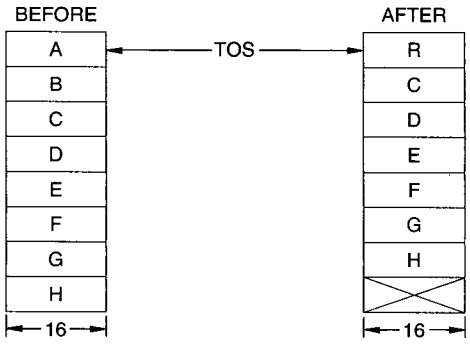
Hex Coding: EF with sr = 1
6F with sr = 0

Execution Time: 84 to 94 clock cycles for A ≠ 0
14 clock cycles for A = 0

Description:
16-bit fixed-point two's complement integer operand B at the NOS is divided by 16-bit fixed-point two's complement integer operand A at the TOS. The 16-bit integer quotient R replaces B and the stack is moved up so that R occupies the TOS. No remainder is generated. Operands A and B are lost. All other operands are unchanged.
If A is zero, R will be set equal to B and the divide-by-zero error status will be reported.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SIN

32-BIT FLOATING-POINT SINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	1	0

Hex Coding: 82 with sr = 1
02 with sr = 0

Execution Time: 3796 to 4808 clock cycles for $|A| > 2^{-12}$ radians
30 clock cycles for $|A| \leq 2^{-12}$ radians

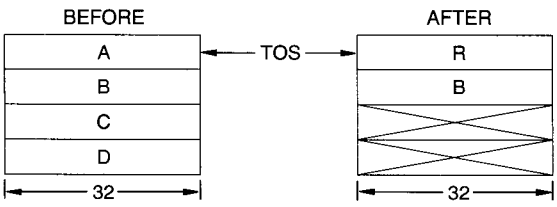
Description:
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point sine of A. A is assumed to be in radians. Operands A, C and D are lost. Operand B is unchanged.

The SIN function will accept any input data value that can be represented by the data format. All input values are range reduced to fall within the interval $-\pi/2$ to $+\pi/2$ radians.

Accuracy: SIN exhibits a maximum relative error of 5.0×10^{-7} for input values in the range of -2π to $+2\pi$ radians.

Status Affected: Sign, Zero

STACK CONTENTS



SMUL

16-BIT FIXED-POINT MULTIPLY, LOWER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	1	0

Hex Coding: EE with sr = 1
6E with sr = 0

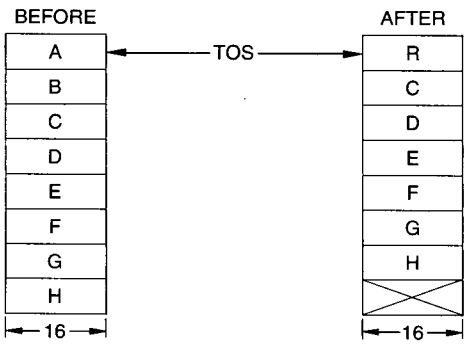
Execution Time: 84 to 94 clock cycles

Description:

16-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 16-bit fixed-point two's complement integer operand B at the NOS. The 16-bit least significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The most significant half of the product is lost. Operands A and B are lost. All other operands are unchanged. The overflow status bit is set if the discarded upper half was non-zero. If either A or B is the most negative value that can be represented in the format, that value is returned as R and the overflow status is set.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SMUU

16-BIT FIXED-POINT MULTIPLY, UPPER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	0	1	1	0

Hex Coding: F6 with sr = 1
76 with sr = 0

Execution Time: 80 to 98 clock cycles

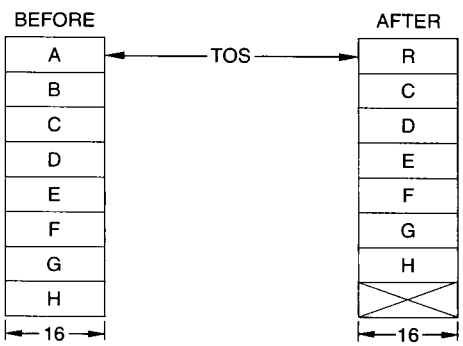
Description:

16-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 16-bit fixed-point two's complement integer operand B at the NOS. The 16-bit most significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The least significant half of the product is lost. Operands A and B are lost. All other operands are unchanged.

If either A or B is the most negative value that can be represented in the format, that value is returned as R and the overflow status is set.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SQRT

32-BIT FLOATING-POINT SQUARE ROOT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	0	1

Hex Coding: 81 with sr = 1
01 with sr = 0

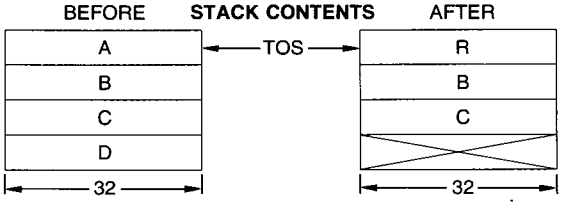
Execution Time: 782 to 870 clock cycles

Description:

32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point square root of A. Operands A and D are lost. Operands B and C are not changed.

SQRT will accept any non-negative input data value that can be represented by the data format. If A is negative an error code of 0100 will be returned in the status register.

Status Affected: Sign, Zero, Error Field



SSUB

16-BIT FIXED-POINT SUBTRACT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	0	1

Hex Coding: ED with sr = 1
6D with sr = 0

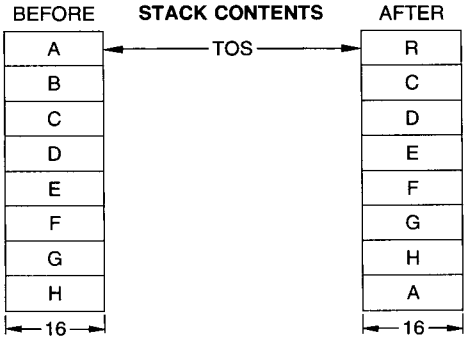
Execution Time: 30 to 32 clock cycles

Description:

16-bit fixed-point two's complement integer operand A at the TOS is subtracted from 16-bit fixed-point two's complement integer operand B at the NOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operand B is lost. All other operands are unchanged.

If the subtraction generates a borrow it is reported in the carry status bit. If A is the most negative value that can be represented in the format the overflow status is set. If the result cannot be represented in the format range, the overflow status is set and the 16 least significant bits of the result are returned as R.

Status Affected: Sign, Zero, Carry, Error Field



TAN

32-BIT FLOATING-POINT TANGENT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	0	0

Hex Coding: 84 with sr = 1
04 with sr = 0

Execution Time: 4894 to 5886 clock cycles for $|A| > 2^{-12}$ radians
30 clock cycles for $|A| \leq 2^{-12}$ radians

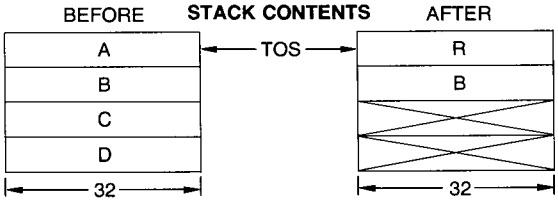
Description:

The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point tangent of A. Operand A is assumed to be in radians. A, C and D are lost. B is unchanged.

The TAN function will accept any input data value that can be represented in the data format. All input data values are range-reduced to fall within $-\pi/4$ to $+\pi/4$ radians. TAN is unbounded for input values near odd multiples of $\pi/2$ and in such cases the overflow bit is set in the status register. For angles smaller than 2^{-12} radians, TAN returns A as the tangent of A.

Accuracy: TAN exhibits a maximum relative error of 5.0 x 10^{-7} for input data values in the range of -2π to $+2\pi$ radians except for data values near odd multiples of $\pi/2$.

Status Affected: Sign, Zero, Error Field (overflow)



XCHD

EXCHANGE 32-BIT STACK OPERANDS

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	1	0	0	1

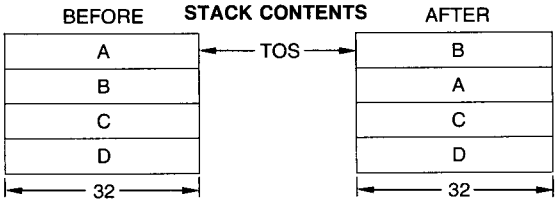
Hex Coding: B9 with sr = 1
39 with sr = 0

Execution Time: 26 clock cycles

Description:

32-bit operand A at the TOS and 32-bit operand B at the NOS are exchanged. After execution, B is at the TOS and A is at the NOS. All operands are unchanged. XCHD and XCHF execute the same operation.

Status Affected: Sign, Zero



XCHF

EXCHANGE 32-BIT STACK OPERANDS

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	0	0	1

Hex Coding: 99 with sr = 1
19 with sr = 0

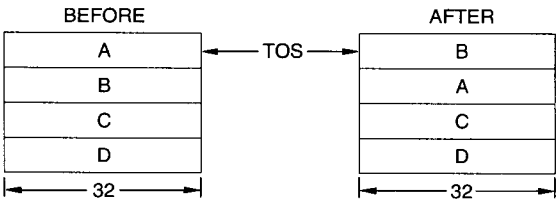
Execution Time: 26 clock cycles

Description:

32-bit operand A at the TOS and 32-bit operand B at the NOS are exchanged. After execution, B is at the TOS and A is at the NOS. All operands are unchanged. XCHD and XCHF execute the same operation.

Status Affected: Sign, Zero

STACK CONTENTS



XCHS

EXCHANGE 16-BIT STACK OPERANDS

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	1	0	0	1

Hex Coding: F9 with sr = 1
79 with sr = 0

Execution Time: 18 clock cycles

Description:

16-bit operand A at the TOS and 16-bit operand B at the NOS are exchanged. After execution, B is at the TOS and A is at the NOS. All operand values are unchanged.

Status Affected: Sign, Zero

STACK CONTENTS