

Am9511A

Arithmetic Processor

Am9511A

2

DISTINCTIVE CHARACTERISTICS

- 2, 3 and 4MHz operation; fixed point 16-bit and 32-bit operations
- Floating point 32-bit operations; binary data formats
- Add, Subtract, Multiply and Divide; trigonometric and inverse trigonometric functions
- Square roots, logarithms, exponentiation; float to fixed and fixed to float conversions
- Stack-oriented operand storage; DMA or programmed I/O data transfers
- End signal simplifies concurrent processing; Synchronous/Asynchronous operations
- General purpose 8-bit data bus interface; standard 24-pin package
- +12 volt and +5 volt power supplies; advanced N-channel silicon gate MOS technology

GENERAL DESCRIPTION

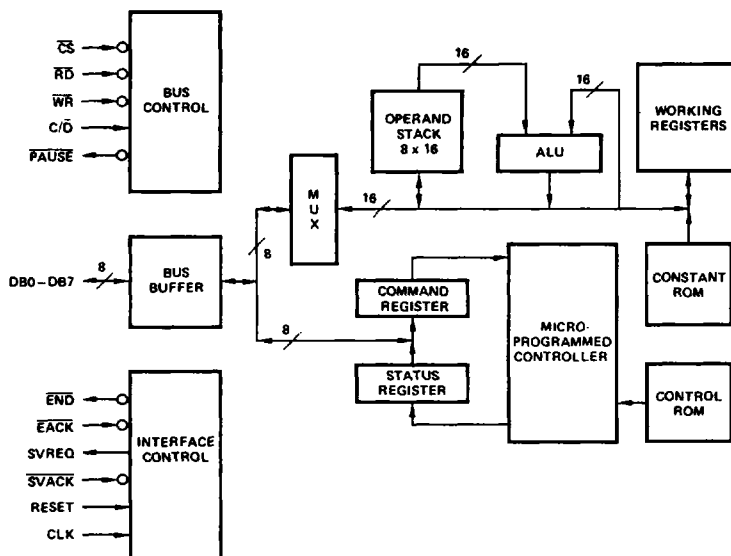
The Am9511A Arithmetic Processing Unit (APU) is a monolithic MOS/LSI device that provides high performance fixed and floating point arithmetic and a variety of floating point trigonometric and mathematical operations. It may be used to enhance the computational capability of a wide variety of processor-oriented systems.

All transfers, including operand, result, status and command information, take place over an 8-bit bidirectional data bus. Operands are pushed onto an internal stack, and a command is issued to perform operations on the data in

the stack. Results are then available to be retrieved from the stack, or additional commands may be entered.

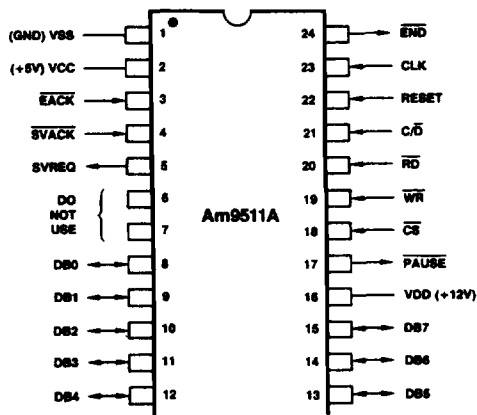
Transfers to and from the APU may be handled by the associated processor using conventional programmed I/O or may be handled by a direct memory access controller for improved performance. Upon completion of each command, the APU issues an end of execution signal that may be used as an interrupt by the CPU to help coordinate program execution.

BLOCK DIAGRAM



BD003340

CONNECTION DIAGRAM Top View D-24-2

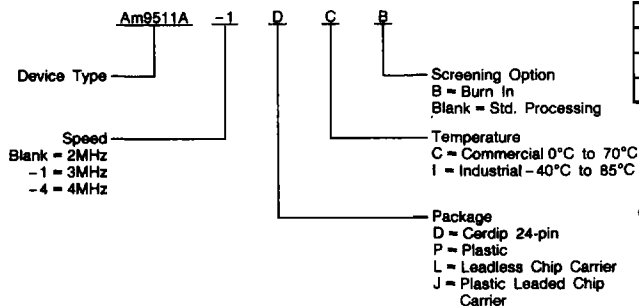


CD005170

Note: Pin 1 is marked for orientation

ORDERING INFORMATION

AMD products are available in several packages and operating ranges. The order number is formed by a combination of the following: Device number, speed option (if applicable), package type, operating range and screening option (if desired).



Valid Combinations

Am9511A	DC, DCB, DI, DIB,
Am9511A-1	LC, LCB, LI, LIB,
Am9511A-4	/BJA

Valid Combinations

Consult the local AMD sales office to confirm availability of specific valid combinations, check for newly released valid combinations and/or obtain additional data on AMD's standard military grade product.

PIN DESCRIPTION

Pin No.	Name	I/O	Description																								
2	V _{CC}		+5V Power Supply.																								
16	V _{DD}		+12V Power Supply.																								
1	V _{SS}		Ground.																								
23	CLK	I	(Clock). An external timing source connected to the CLK input provides the necessary clocking. The CLK input can be asynchronous to the RD and WR control signals.																								
22	RESET	I	(Reset). A HIGH on this input causes initialization. Reset terminates any operation in progress and clears the status register to zero. The internal stack pointer is initialized, and the contents of the stack may be affected, but the command register is not affected by the reset operation. After a reset the END output will be HIGH, and the SVREQ output will be LOW. For proper initialization, the RESET input must be HIGH for at least five CLK periods following stable power supply voltages and stable clock.																								
21	C/D	I	(Command/Data Select). The C/D input, with the RD and WR inputs, determines the type of transfer to be performed on the data bus as follows: <table border="1"> <thead> <tr> <th>C/D</th><th>RD</th><th>WR</th><th>Function</th></tr> </thead> <tbody> <tr> <td>L</td><td>H</td><td>L</td><td>Push data byte into the stack</td></tr> <tr> <td>L</td><td>L</td><td>H</td><td>Pop data byte from the stack</td></tr> <tr> <td>H</td><td>H</td><td>L</td><td>Enter command byte from the data bus</td></tr> <tr> <td>H</td><td>L</td><td>H</td><td>Read Status</td></tr> <tr> <td>X</td><td>L</td><td>L</td><td>Undefined</td></tr> </tbody> </table> <p> L = LOW H = HIGH X = DON'T CARE </p>	C/D	RD	WR	Function	L	H	L	Push data byte into the stack	L	L	H	Pop data byte from the stack	H	H	L	Enter command byte from the data bus	H	L	H	Read Status	X	L	L	Undefined
C/D	RD	WR	Function																								
L	H	L	Push data byte into the stack																								
L	L	H	Pop data byte from the stack																								
H	H	L	Enter command byte from the data bus																								
H	L	H	Read Status																								
X	L	L	Undefined																								
24	END	O	(End of Execution). A LOW on this output indicates that execution of the current command is complete. This output will be cleared HIGH by activating the EACK input LOW or performing any read or write operation or device initialization using the RESET. If EACK is tied LOW, the END output will be a pulse (see EACK description). This is an open drain output and requires a pull up to +5V. Reading the status register while a command execution is in progress is allowed. However, any read or write operation clears the flip-flop that generates the END output. Thus, such continuous reading could conflict with internal logic setting the END flip-flop at the completion of command execution.																								
3	EACK	I	(End Acknowledge). This input when LOW makes the END output go HIGH. As mentioned earlier, LOW on the END output signals completion of a command execution. The END output signal is derived from an internal flip-flop which is clocked at the completion of a command. This flip-flop is clocked to the reset state when EACK is LOW. Consequently, if the EACK is tied LOW, the END output will be a pulse that is approximately one CLK period wide.																								
5	SVREQ	O	(Service Request). A HIGH on this output indicates completion of a command. In this sense, this output is same as the END output. However, whether the SVREQ output will go HIGH at the completion of a command or not is determined by a service request bit in the command register. This bit must be 1 for SVREQ to go HIGH. The SVREQ can be cleared (i.e., go LOW) by activating the SVACK input LOW or initializing the device using the RESET. Also, the SVREQ will be automatically cleared after completion of any command that has the service request bit as 0.																								
4	SVACK	I	(Service Acknowledge). A LOW on this input activates the reset input of the flip-flop generating the SVREQ output. If the SVACK input is permanently tied LOW, it will conflict with the internal setting of the flip-flop to generate the SVREQ output. Thus the SVREQ indication cannot be relied upon if the SVACK is tied LOW.																								
8-15	DB0-DB7	I/O	(Bidirectional Data Bus). These eight bidirectional lines are used to transfer command, status and operand information between the device and the host processor. DB0 is the least significant, and DB7 is the most significant bit position. HIGH on the data bus line corresponds to 1, and LOW corresponds to 0. When pushing operands on the stack using the data bus, the least significant byte must be pushed first and most significant byte last. When popping the stack to read the result of an operation, the most significant byte will be available on the data bus first, and the least significant byte will be the last. Moreover, for pushing operands and popping results, the number of transactions must be equal to the proper number of bytes appropriate for the chosen format. Otherwise, the internal byte pointer will not be aligned properly. The Am9511A single precision format requires 2 bytes; double precision and floating-point formats require 4 bytes.																								

PIN DESCRIPTION (Cont.)

Pin No.	Name	I/O	Description
18	CS	I	<p>(Chip Select). This input must be LOW to accomplish any read or write operation to the Am9511A.</p> <p>To perform a write operation, data is presented on DB0 through DB7 lines, C/D is driven to an appropriate level and the CS input is made LOW. However, actual writing into the Am9511A cannot start until WR is made LOW. After initiating the write operation by a WR HIGH-to-LOW transition, the PAUSE output will go LOW momentarily (TPPWW).</p> <p>The WR input can go HIGH after PAUSE goes HIGH. The data lines, C/D input and the CS input can change when appropriate hold time requirements are satisfied. See Write Timing diagram for details.</p> <p>To perform a read operation, an appropriate logic level is established on the C/D input, and CS is made LOW. The read operation does not start until the RD input goes LOW. PAUSE will go LOW for a period of TPPWR. When PAUSE goes back HIGH again, it indicates that the read operation is complete, and the required information is available on the DB0 through DB7 lines. This information will remain on the data lines as long as the RD input is LOW. The RD input can return HIGH anytime after PAUSE goes HIGH. The CS input and C/D inputs can change anytime after RD returns HIGH. See Read Timing diagram for details.</p>
20	RD	I	<p>(Read). A LOW on this input is used to read information from an internal location and gate that information on to the data bus. The CS input must be LOW to accomplish the read operation. The C/D input determines what internal location is of interest. See C/D, CS input descriptions and Read Timing diagram for details. If the END output was LOW, performing any read operation will make the END output go HIGH after the HIGH-to-LOW transition of the RD input (assuming CS is LOW).</p>
19	WR	I	<p>(Write). A LOW on this input is used to transfer information from the data bus into an internal location. The CS must be LOW to accomplish the write operation. The C/D determines which internal location is to be written. See C/D, CS input descriptions and Write Timing diagram for details.</p> <p>If the END output was LOW, performing any write operation will make the END output go HIGH after the LOW-to-HIGH transition of the WR input (assuming CS is LOW).</p>
17	PAUSE	O	<p>(Pause). This output is a handshake signal used while performing read or write transactions with the Am9511A. A LOW at this output indicates that the Am9511A has not yet completed its information transfer with the host over the data bus. During a read operation, after CS went LOW, the PAUSE will become LOW shortly (TRP) after RD goes LOW. PAUSE will return high only after the data bus contains valid output data. The CS and RD should remain LOW when PAUSE is LOW. The RD may go high anytime after PAUSE goes HIGH. During a write operation, after CS went LOW, the PAUSE will be LOW for a very short duration (TPPWN) after WR goes LOW. Since the minimum of TPPWW is 0, the PAUSE may not go LOW at all for fast devices. WR may go HIGH anytime after PAUSE goes HIGH.</p>

DETAILED DESCRIPTION

Major functional units of the Am9511A are shown in the block diagram. The Am9511A employs a microprogram controlled stack oriented architecture with 16-bit wide data paths.

The Arithmetic Logic Unit (ALU) receives one of its operands from the Operand Stack. This stack is an 8-word by 16-bit 2-port memory with last in-first out (LIFO) attributes. The second operand to the ALU is supplied by the internal 16-bit bus. In addition to supplying the second operand, this bidirectional bus also carries the results from the output of the ALU when required. Writing into the Operand Stack takes place from this internal 16-bit bus when required. Also connected to this bus are the Constant ROM and Working Registers. The ROM provides the required constants to perform the mathematical operations (Chebyshev Algorithms), while the Working Registers provide storage for the intermediate values during command execution.

Communication between the external world and the Am9511A takes place on eight bidirectional input/output lines DB0 through DB7 (Data Bus). These signals are gated to the internal eight-bit bus through appropriate interface and buffer circuitry. Multiplexing facilities exist for bidirectional communication between the internal eight- and sixteen-bit buses. The Status Register and Command Register are also accessible via the eight-bit bus.

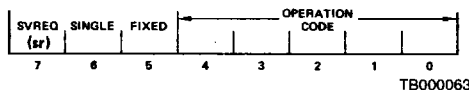
The Am9511A operations are controlled by the microprogram contained in the Control ROM. The Program Counter supplies the microprogram addresses and can be partially loaded from the Command Register. Associated with the Program Counter is the Subroutine Stack where return addresses are held during subroutine calls in the microprogram. The Microinstruction Register holds the current microinstruction being executed.

ed. This register facilitates pipelined microprogram execution. The Instruction Decode logic generates various internal control signals needed for the Am9511A operation.

The Interface Control logic receives several external inputs and provides handshake related outputs to facilitate interfacing the Am9511A to microprocessors.

Command Format

Each command entered into the Am9511A consists of a single 8-bit byte having the format illustrated below:



Bits 0 – 4 select the operation to be performed as shown in the table. Bits 5 – 6 select the data format for the operation. If bit 5 is a 1, a fixed point data format is specified. If bit 5 is a 0, floating point format is specified. Bit 6 selects the precision of the data to be operated on by fixed point commands (if bit 5 = 0, bit 6 must be 0). If bit 6 is a 1, single-precision (16-bit) operands are indicated; if bit 6 is a 0, double-precision (32-bit) operands are indicated. Results are undefined for all illegal combinations of bits in the command byte. Bit 7 indicates whether a service request is to be issued after the command is executed. If bit 7 is a 1, the service request output (SVREQ) will go HIGH at the conclusion of the command and will remain HIGH until reset by a LOW level on the service acknowledge pin (SVACK) or until completion of execution of a succeeding command where bit 7 is 0. Each command issued to the Am9511A requests post execution service based upon the state of bit 7 in the command byte. When bit 7 is a 0, SVREQ remains LOW.

COMMAND SUMMARY										
Command Code								Command Mnemonic	Command Description	
7	6	5	4	3	2	1	0			
FIXED-POINT 16-BIT										
Sr	1	1	0	1	1	0	0	SADD	Add TOS to NOS. Result to NOS. Pop Stack.	
Sr	1	1	0	1	1	0	1	SSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.	
Sr	1	1	0	1	1	1	0	SMUL	Multiply NOS by TOS. Lower half of result to NOS. Pop Stack.	
Sr	1	1	1	0	1	1	0	SMUJ	Multiply NOS by TOS. Upper half of result to NOS. Pop Stack.	
Sr	1	1	0	1	1	1	1	SDIV	Divide NOS by TOS. Result to NOS. Pop Stack.	
FIXED-POINT 32-BIT										
Sr	0	1	0	1	1	0	0	DADD	Add TOS to NOS. Result to NOS. Pop Stack.	
Sr	0	1	0	1	1	0	1	DSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.	
Sr	0	1	0	1	1	1	0	DMUL	Multiply NOS by TOS. Lower half of result to NOS. Pop Stack.	
Sr	0	1	1	0	1	1	0	DMUJ	Multiply NOS by TOS. Upper half of result to NOS. Pop Stack.	
Sr	0	1	0	1	1	1	1	DDIV	Divide NOS by TOS. Result to NOS. Pop Stack.	
FLOATING-POINT 32-BIT										
Sr	0	0	1	0	0	0	0	FADD	Add TOS to NOS. Result to NOS. Pop Stack.	
Sr	0	0	1	0	0	0	1	FSUB	Subtract TOS from NOS. Result to NOS. Pop Stack.	
Sr	0	0	1	0	0	1	0	FMUL	Multiply NOS by TOS. Result to NOS. Pop Stack.	
Sr	0	0	1	0	0	1	1	FDIV	Divide NOS by TOS. Result to NOS. Pop Stack.	
DERIVED FLOATING-POINT FUNCTIONS										
Sr	0	0	0	0	0	0	1	SQRT	Square Root of TOS. Result in TOS.	
Sr	0	0	0	0	0	1	0	SIN	Sine of TOS. Result in TOS.	
Sr	0	0	0	0	0	1	1	COS	Cosine of TOS. Result in TOS.	
Sr	0	0	0	0	1	0	0	TAN	Tangent of TOS. Result in TOS.	
Sr	0	0	0	0	1	0	1	ASIN	Inverse Sine of TOS. Result in TOS.	
Sr	0	0	0	0	1	1	0	ACOS	Inverse Cosine of TOS. Result in TOS.	
Sr	0	0	0	0	1	1	1	ATAN	Inverse Tangent of TOS. Result in TOS.	
Sr	0	0	0	1	0	0	0	LOG	Common Logarithm (base 10) of TOS. Result in TOS.	
Sr	0	0	0	1	0	0	1	LN	Natural Logarithm (base e) of TOS. Result in TOS.	
Sr	0	0	0	1	0	1	0	EXP	Exponential (e ^x) of TOS. Result in TOS.	
Sr	0	0	0	1	0	1	1	PWR	NOS raised to the power in TOS. Result in NOS. Pop Stack.	
DATA MANIPULATION COMMANDS										
Sr	0	0	0	0	0	0	0	NOP	No Operation.	
Sr	0	0	1	1	1	1	1	FIXS	Convert TOS from floating-point to 16-bit fixed-point format.	
Sr	0	0	1	1	1	1	0	FIXD	Convert TOS from floating-point to 32-bit fixed-point format.	
Sr	0	0	1	1	1	0	1	FLTS	Convert TOS from 16-bit fixed-point to floating-point format.	
Sr	0	0	1	1	1	0	0	FLTD	Convert TOS from 32-bit fixed-point to floating-point format.	
Sr	1	1	1	0	1	0	0	CHSS	Change sign of 16-bit fixed-point operand on TOS.	
Sr	0	1	1	0	1	0	0	CHSD	Change sign of 32-bit fixed-point operand on TOS.	
Sr	0	0	1	0	1	0	1	CHSF	Change sign of floating-point operand on TOS.	
Sr	1	1	1	0	1	1	1	PTOS	Push 16-bit fixed-point operand on TOS to NOS. (Copy.)	
Sr	0	1	1	0	1	1	1	PTOD	Push 32-bit fixed-point operand on TOS to NOS. (Copy.)	
Sr	0	0	1	0	1	1	1	PTOF	Push floating-point operand on TOS to NOS. (Copy.)	
Sr	1	1	1	1	0	0	0	POPS	Pop 16-bit fixed-point operand from TOS. NOS becomes TOS.	
Sr	0	1	1	1	0	0	0	POPD	Pop 32-bit fixed-point operand from TOS. NOS becomes TOS.	
Sr	0	0	1	1	0	0	0	POPF	Pop floating-point operand from TOS. NOS becomes TOS.	
Sr	1	1	1	1	0	0	1	XCHS	Exchange 16-bit fixed-point operands TOS and NOS.	
Sr	0	1	1	1	0	0	1	XCHD	Exchange 32-bit fixed-point operands TOS and NOS.	
Sr	0	0	1	1	0	0	1	XCHF	Exchange floating-point operands TOS and NOS.	
Sr	0	0	1	1	0	1	0	PUPJ	Push floating-point constant "r" onto TOS. Previous TOS becomes NOS.	

Notes: 1. TOS means Top of Stack. NOS means Next on Stack.

2. AMD Application Brief "Algorithm Details for the Am9511A APU" provides detailed descriptions of each command function, including data ranges, accuracies, stack configurations, etc.

3. Many commands destroy one stack location (bottom of stack) during development of the result. The derived functions may destroy several stack locations. See Application Brief for details.

4. The trigonometric functions handle angles in radians, not degrees.

5. No remainder is available for the fixed-point divide functions.

6. Results will be undefined for any combination of command coding bits not specified in this table.

Command Initiation

After properly positioning the required operands on the stack, a command may be issued. The procedure for initiating a command execution is as follows:

1. Enter the appropriate command on the DB0 - DB7 lines.
2. Establish HIGH on the C/\bar{D} input.
3. Establish LOW on the \bar{CS} input.
4. Establish LOW on the WR input after an appropriate set-up time (see Timing diagrams).
5. Sometime after the HIGH-to-LOW level transition of WR input, the PAUSE output will become LOW. After a delay of TPPWW, it will go HIGH to acknowledge the write operation. The WR input can return to HIGH anytime after PAUSE going HIGH. The DB0 - DB7, C/\bar{D} and \bar{CS} inputs are allowed to change after the

hold time requirements are satisfied (see Timing diagram).

An attempt to issue a new command while the current command execution is in progress is allowed. Under these circumstances, the PAUSE output will not go HIGH until the current command execution is completed.

Operand Entry

The Am9511A commands operate on the operands located at the TOS and NOS, and results are returned to the stack at NOS and then popped to TOS. The operands required for the Am9511A are one of three formats - single precision fixed-point (2 bytes), double precision fixed-point (4 bytes) or floating-point (4 bytes). The result of an operation has the

same format as the operands except for float to fix or fix to float commands.

Operands are always entered into the stack least significant byte first and most significant byte last. The following procedure must be followed to enter operands onto the stack:

1. The lower significant operand byte is established on the DB0 - DB7 lines.
2. A LOW is established on the C/D input to specify that data is to be entered into the stack.
3. The CS input is made LOW.
4. After appropriate set-up time (see Timing diagrams), the WR input is made LOW. The PAUSE output will become LOW.
5. Sometime after this event, the PAUSE will return HIGH to indicate that the write operation has been acknowledged.
6. Anytime after the PAUSE output goes HIGH, the WR input can be made HIGH. The DB0 - DB7, C/D and CS inputs can change after appropriate hold time requirements are satisfied (see Timing diagrams).

The above procedure must be repeated until all bytes of the operand are pushed onto the stack. It should be noted that for single precision fixed-point operands, 2 bytes should be pushed and 4 bytes must be pushed for double precision fixed-point or floating-point. Not pushing all the bytes of a quantity will result in byte pointer misalignment.

The Am9511A stack can accommodate 8 single precision fixed-point quantities or 4 double precision fixed-point or floating-point quantities. Pushing more quantities than the capacity of the stack will result in loss of data which is usual with any LIFO stack.

Data Removal

Result from an operation will be available at the TOS. Results can be transferred from the stack to the data bus by reading the stack. When the stack is popped for results, the most significant byte is available first and the least significant byte last. A result is always of the same precision as the operands that produced it except for format conversion commands. Thus when the result is taken from the stack, the total number of bytes popped out should agree with the precision - single precision results are 2 bytes, and double precision and floating-point results are 4 bytes. The following procedure must be used for reading the result from the stack:

1. A LOW is established on the C/D input.
2. The CS input is made LOW.
3. After appropriate set-up time (see Timing diagrams), the RD input is made LOW. The PAUSE will become LOW.
4. Sometime after this, PAUSE will return HIGH, indicating that the data is available on the DB0 - DB7 lines. This data will remain on the DB0 - DB7 lines as long as the RD input remains LOW.
5. Anytime after PAUSE goes HIGH, the RD input can return HIGH to complete transaction.
6. The CS and C/D inputs can change after appropriate hold time requirements are satisfied (see Timing diagram).
7. Repeat this procedure until all bytes appropriate for the precision of the result are popped out.

Reading of the stack does not alter its data; it only adjusts the byte pointer. If more data is popped than the capacity of the stack, the internal byte pointer will wrap around and older data will be read again, consistent with the LIFO stack.

Status Read

The Am9511A status register can be read without any regard to whether a command is in progress or not. The only implication that has to be considered is the effect this might have on the END output discussed in the signal descriptions.

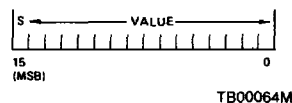
The following procedure must be followed to accomplish status register reading:

1. Establish HIGH on the C/D input.
2. Establish LOW on the CS input.
3. After appropriate set-up time (see Timing diagram) RD input is made LOW. The PAUSE will become LOW.
4. Sometime after the HIGH-to-LOW transition of RD input, the PAUSE will become HIGH, indicating that status register contents are available on the DB0 - DB7 lines. The status data will remain on DB0 - DB7 as long as RD input is LOW.
5. The RD input can be returned HIGH anytime after PAUSE goes HIGH.
6. The C/D input and CS input can change after satisfying appropriate hold time requirements (see Timing diagram).

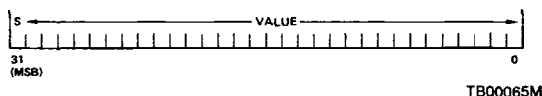
Data Formats

The Am9511A Arithmetic Processing Unit handles operands in both fixed-point and floating-point formats. Fixed-point operands may be represented in either single (16-bit operands) or double precision (32-bit operands), and are always represented as binary, two's complement values.

16-BIT FIXED-POINT FORMAT



32-BIT FIXED-POINT FORMAT



The sign (positive or negative) of the operand is located in the most significant bit (MSB). Positive values are represented by a sign bit of zero (S = 0). Negative values are represented by the two's complement of the corresponding positive value with a sign bit equal to 1 (S = 1). The range of values that may be accommodated by each of these formats is -32,767 to +32,767 for single precision and -2,147,483,647 to +2,147,483,647 for double precision.

Floating point binary values are represented in a format that permits arithmetic to be performed in a fashion analogous to operations with decimal values expressed in scientific notation.

$$(5.83 \times 10^2)(8.16 \times 10^1) = (4.75728 \times 10^4)$$

In the decimal system, data may be expressed as values between 0 and 10 times 10 raised to a power that effectively shifts the implied decimal point right or left the number of places necessary to express the result in conventional form (e.g., 47,572.8). The value-portion of the data is called the mantissa. The exponent may be either negative or positive.

The concept of floating point notation has both a gain and a loss associated with it. The gain is the ability to represent the significant digits of data with values spanning a large dynamic range limited only by the capacity of the exponent field. For

example, in decimal notation if the exponent field is two digits wide and the mantissa is five digits, a range of values (positive or negative) from 1.000×10^{-99} to $9.9999 \times 10^{+99}$ can be accommodated. The loss is that only the significant digits of the value can be represented. Thus there is no distinction in this representation between the values 123451 and 123452, for example, since each would be expressed as: 1.2345×10^5 . The sixth digit has been discarded. In most applications where the dynamic range of values to be represented is large, the loss of significance, and hence accuracy of results, is a minor consideration. For greater precision a fixed point format could be chosen, although with a loss of potential dynamic range.

The Am9511 is a binary arithmetic processor and requires that floating point data be represented by a fractional mantissa value between .5 and 1 multiplied by 2 raised to an appropriate power. This is expressed as follows:

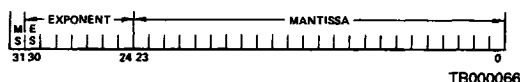
$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

For example, the value 100.5 expressed in this form is 0.11001001×2^7 . The decimal equivalent of this value may be computed by summing the components (powers of two) of the mantissa and then multiplying by the exponent as shown below:

$$\begin{aligned} \text{value} &= (2^{-1} + 2^{-2} + 2^{-5} + 2^{-8}) \times 2^7 \\ &= (0.5 + 0.25 + 0.03125 + 0.00290625) \times 128 \\ &= 0.78515625 \times 128 \\ &= 100.5 \end{aligned}$$

Floating Point Format

The format for floating-point values in the Am9511A is given below. The mantissa is expressed as a 24-bit (fractional) value; the exponent is expressed as an unbiased two's complement 7-bit value having a range of -64 to +63. The most significant bit is the sign of the mantissa (0 = positive, 1 = negative), for a total of 32 bits. The binary point is assumed to be to the left of the most significant mantissa bit (bit 23). All floating-point data values must be normalized. Bit 23 must be equal to 1, except for the value zero, which is represented by all zeros.



The range of values that can be represented in this format is $\pm(2.7 \times 10^{-20}$ to $9.2 \times 10^{18})$ and zero.

Status Register

The Am9511A contains an eight bit status register with the following bit assignments:

BUSY	SIGN	ZERO	ERROR CODE	CARRY			
7	6	5	4	3	2	1	0

BUSY: Indicates that Am9511A is currently executing a command (1 = Busy).

SIGN: Indicates that the value on the top of stack is negative (1 = Negative).

ZERO: Indicates that the value on the top of stack is zero (1 = Value is zero).

ERROR

CODE: This field contains an indication of the validity of the result of the last operation. The error codes are:

- 0000 - No error
- 1000 - Divide by error
- 0100 - Square root or log of negative number
- 1100 - Argument of inverse sine, cosine, or e^x too large

XX10 - Underflow

XX01 - Overflow

CARRY: Previous operation resulted in carry or borrow from most significant bit (1 = Carry/Borrow, 0 = No Carry/No Borrow).

If the BUSY bit in the status register is a one, the other status bits are not defined; if zero, indicating not busy, the operation is complete, and the other status bits are defined as given above.

Table 1.

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	Increase 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	4296-6956	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	
PTOD	B7	37	20	} Push stack. Duplicate NOS in TOS.
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	
PUP1	9A	1A	16	Push floating-point constant π onto TOS. Previous TOS becomes NOS.

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 3MHz rate translates to 14 microseconds

($44 \times 32\mu s = 14\mu s$). 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.

Table 1 is a summary of all the Am9511A 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.

The command mnemonics in alphabetical order are shown below in Table 2.

Table 2.
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	EXPONENTATION (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

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 k cycles clock cycles

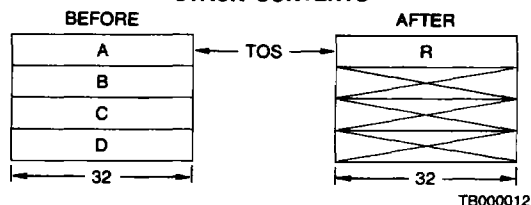
Description:

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



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 8536 clock cycles

Description:

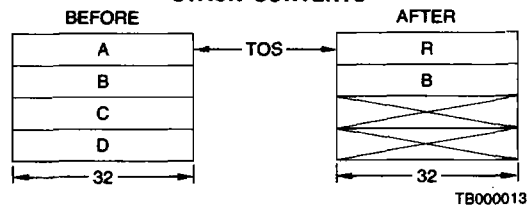
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point inverse tangent of A. The result R is a value in radians between $-\pi/2$ and $+\pi/2$. Initial operands A, C, and D are lost. Operand B is unchanged.

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



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 k cycles clock cycles

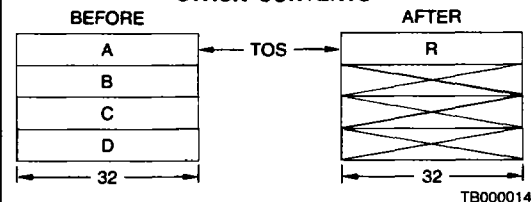
Description:

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



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

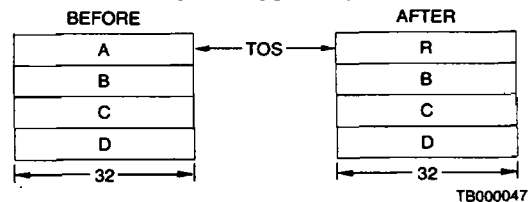
Description:

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

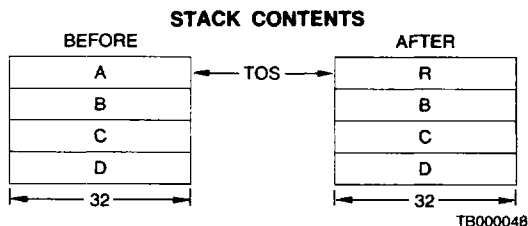
Execution Time: 16 to 20 clock cycles

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



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

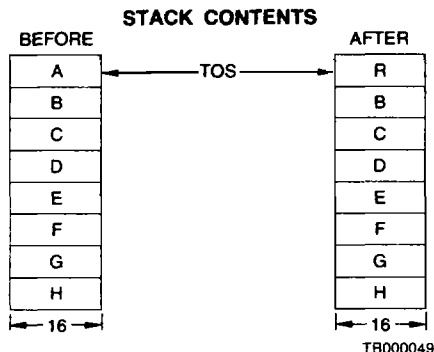
Execution Time: 22 to 24 clock cycles

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



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

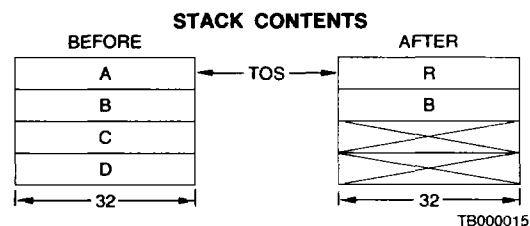
Description:

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 5.0×10^{-7} for all input data values in the range of -2π to $+2\pi$ radians.

Status Affected: Sign, Zero



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

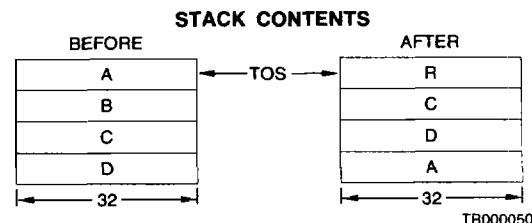
Execution Time: 20 to 22 clock cycles

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



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

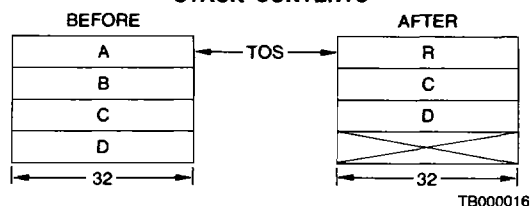
18 clock cycles when A = 0.

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 divided-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

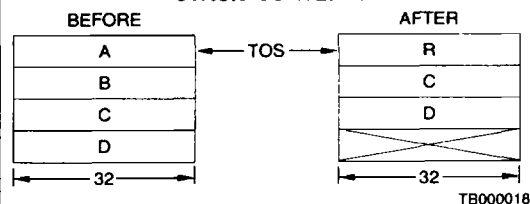
Description:

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

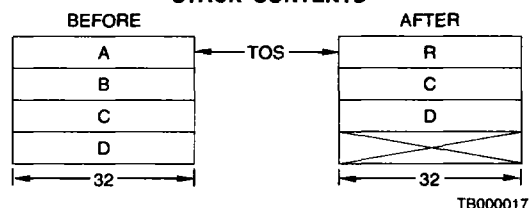
Description:

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

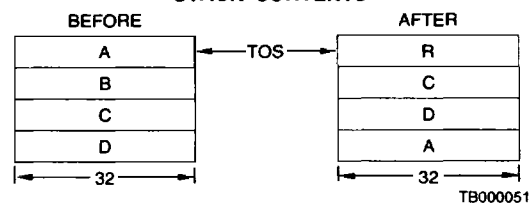
Description:

The 32-bit fixed-point two's complement integer operand A at the TOS is subtracted from the 32-bit fixed-point two's complement integer 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
sr	0	0	0	0	1	0	1	0

Binary Coding:

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:**

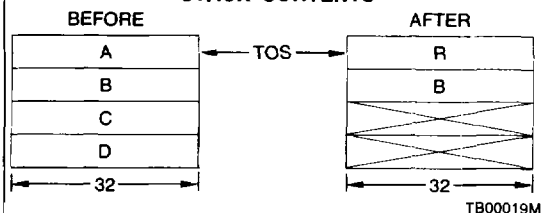
The base of natural logarithms, e , is raised to an exponent value specified by the 32-bit floating-point operand A at the TOS. The result R of e^A replaces A. Operands A, C and D are lost. Operand B is unchanged.

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



FDIV

32-BIT FLOATING-POINT DIVIDE

	7	6	5	4	3	2	1	0
sr	0	0	1	0	0	1	1	

Binary Coding:

Hex Coding:

93 with sr = 1

13 with sr = 0

Execution Time:

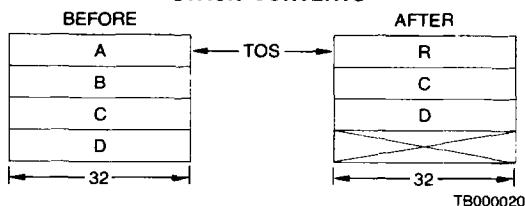
154 to 184 clock cycles for $A \neq 0$ 22 clock cycles for $A = 0$ **Description:**

32-bit floating-point operand B at the 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



FADD

32-BIT FLOATING-POINT ADD

	7	6	5	4	3	2	1	0
sr	0	0	1	0	0	0	0	0

Binary Coding:

Hex Coding:

90 with sr = 1

10 with sr = 0

Execution Time:

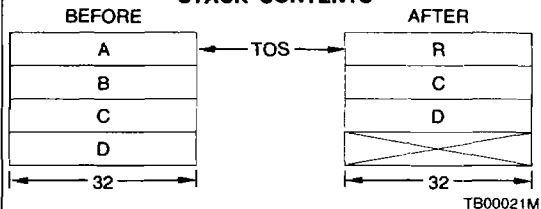
54 to 368 clock cycles for $A \neq 0$ 24 clock cycles for $A = 0$ **Description:**

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



FIXD

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

	7	6	5	4	3	2	1	0
sr	0	0	1	1	1	1	1	0

Binary Coding:

Hex Coding:

9E with sr = 1

1E with sr = 0

Execution Time:

90 to 336 clock cycles

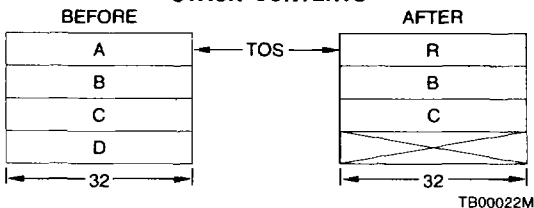
Description:

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:

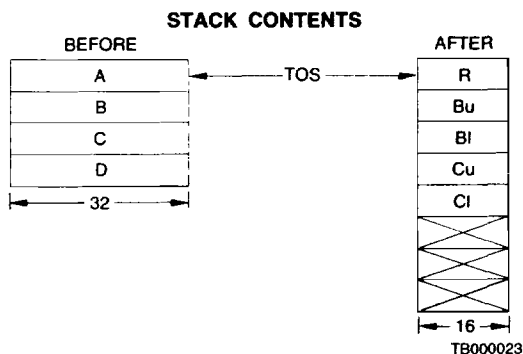
90 to 214 clock cycles

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



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 1 0 1

Hex Coding:

9D with sr = 1

1D with sr = 0

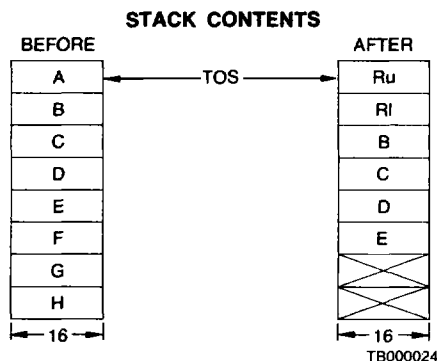
Execution Time:

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



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 1 0 0

Hex Coding:

9C with sr = 1

1C with sr = 0

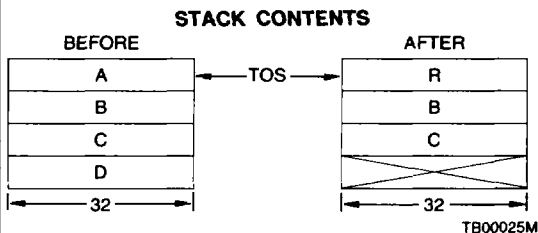
Execution Time:

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



FMUL

32-BIT FLOATING-POINT MULTIPLY

7 6 5 4 3 2 1 0

Binary Coding:

sr 0 0 1 0 0 1 0 0

Hex Coding:

92 with sr = 1

12 with sr = 0

Execution Time:

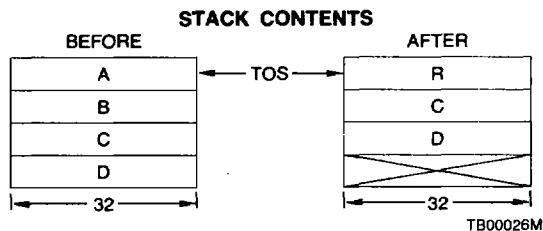
146 to 168 clock cycles

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



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:

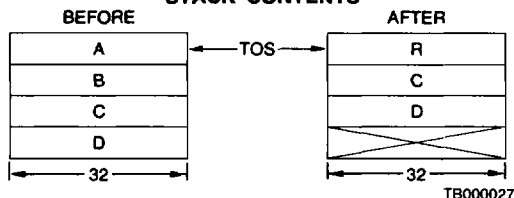
70 to 370 clock cycles for $A \neq 0$ 26 clock cycles for $A = 0$

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)

STACK CONTENTS



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 $A \leq 0$

Description:

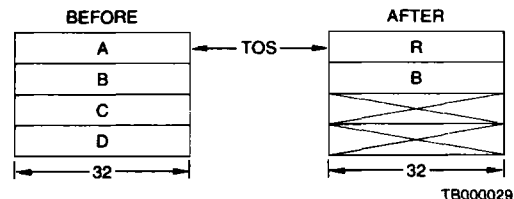
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point common logarithm (base 10) of A. Operands A, C and D are lost. Operand B is unchanged.

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

STACK CONTENTS



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 $A \leq 0$

Description:

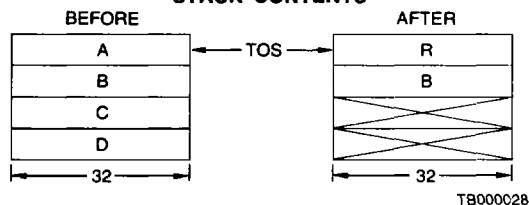
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point natural logarithm (base e) of A. Operands A, C and D are lost. Operand B is unchanged.

The LN function accepts all positive input data value that can be represented by the data format. If LN of a non-positive value 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

STACK CONTENTS



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:

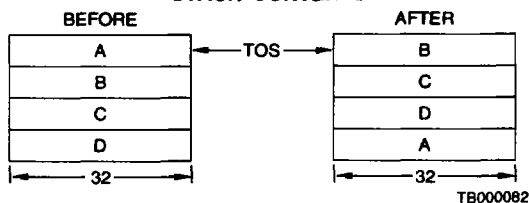
12 clock cycles

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:

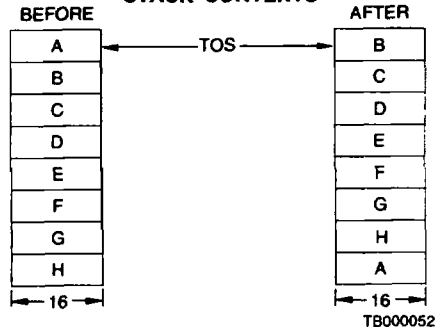
10 clock cycles

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:

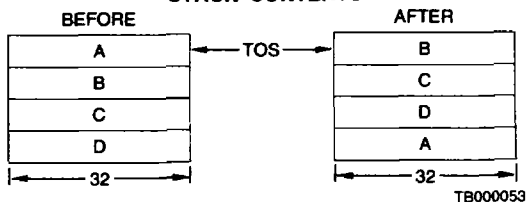
12 clock cycles

Description:

The 32-bit stack is moved up so that the old NOS becomes the new TOS. The previous 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:

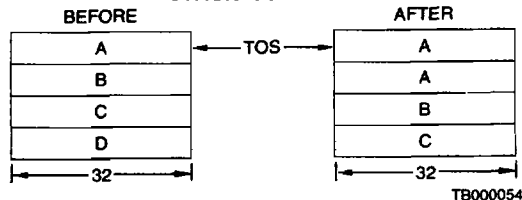
20 clock cycles

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:

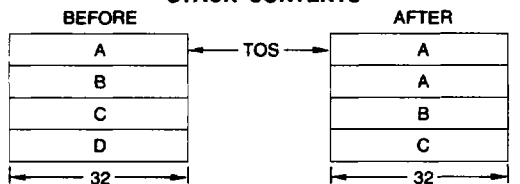
20 clock cycles

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



TB000055

PUIP

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:

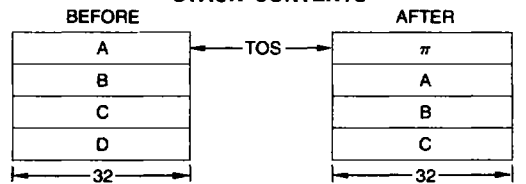
16 clock cycles

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



TB000056

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:

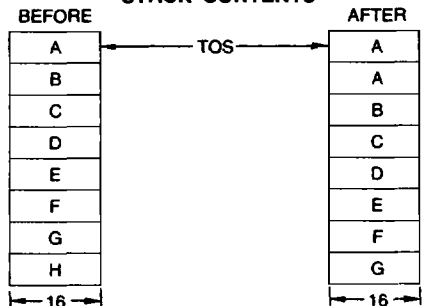
16 clock cycles

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



TB000057

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.

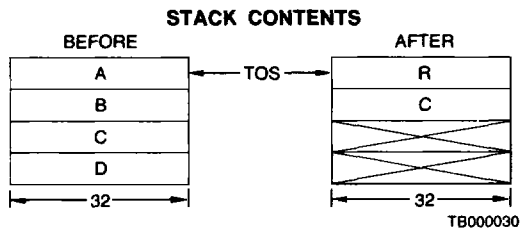
The PWR function accepts all input data values that can be represented in the data format for operand A and all positive values for operand B. If operand B is non-positive, an error status of 0100 will be returned. The EXP and LN functions are used to implement PWR using the relationship $B^A = \text{EXP}[A(\text{LN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

$$|(\text{Related Error})_{\text{PWR}}| = |(\text{Relative Error})_{\text{EXP}} + |A|(\text{Absolute Error})_{\text{LN}}|$$

The maximum relative error for PWR occurs when A is at its maximum value while $[A(\text{LN } B)]$ 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



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:

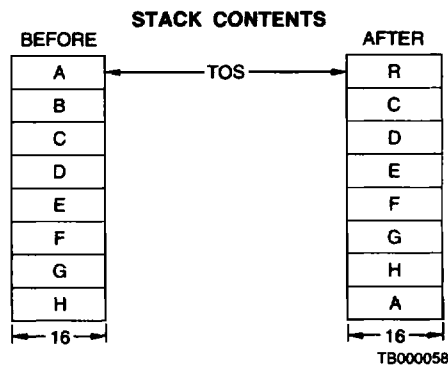
16 to 18 clock cycles

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



SDIV

16-BIT
FIXED-POINT DIVIDE

7	6	5	4	3	2	1	0
sr	1	1	0	1	1	1	1

Binary Coding:

Hex Coding:

EF with sr = 1

6F with sr = 0

Execution Time:

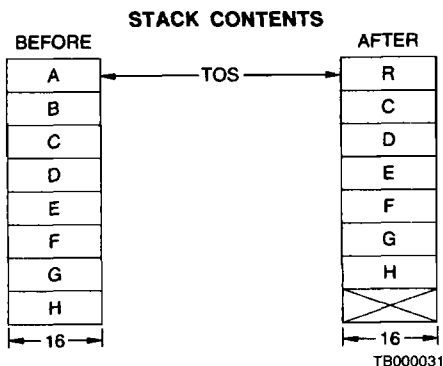
84 to 94 clock cycles for $A \neq 0$ 14 clock cycles for $A = 0$

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 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



SIN

32-BIT
FLOATING-POINT SINE

7	6	5	4	3	2	1	0
sr	0	0	0	0	0	1	0

Binary Coding:

Hex Coding:

82 with sr = 1

02 with sr = 0

Execution Time:

3796 to 4808 clock cycles for $|A| > 2^{-12}$ radians30 clock cycles for $|A| \leq 2^{-12}$ radians

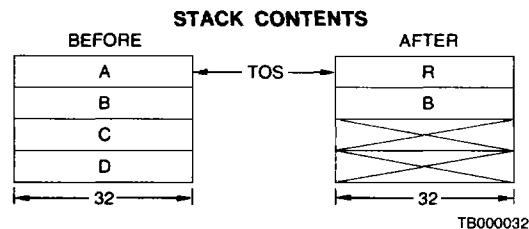
Description:

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



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:

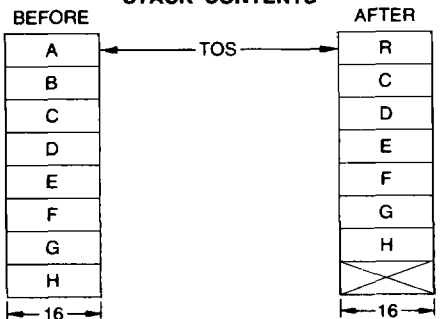
84 to 94 clock cycles

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



TB000033

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

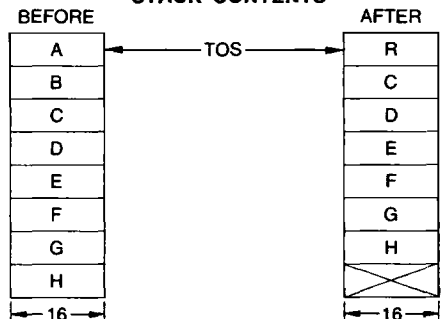
Description:

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



TB000034

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

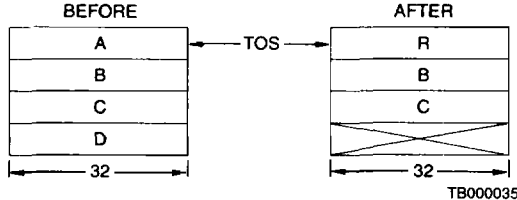
Description:

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 unchanged.

SQRT will accept any non-negative input data 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

STACK CONTENTS



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:

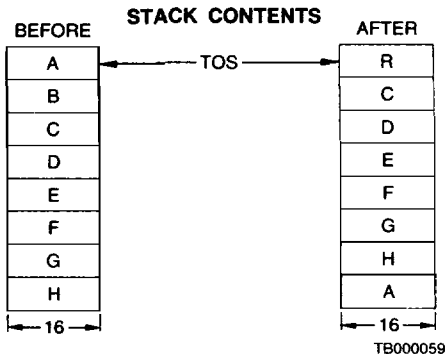
30 to 32 clock cycles

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 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 cycle for $|A| \leq 2^{-12}$ radians

Description:

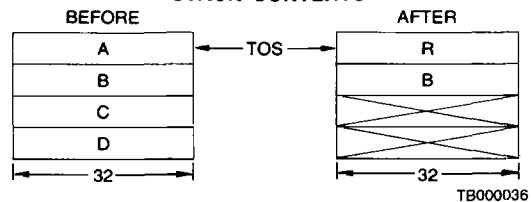
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point tangent of A. Operand A is assumed to be in radians. A, C and D are lost. B is unchanged.

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 5.0×10^{-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)

STACK CONTENTS



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:

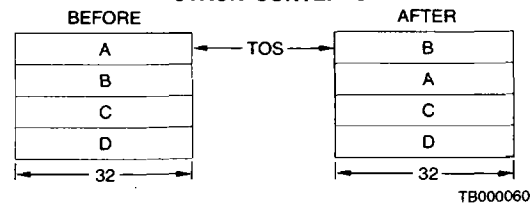
26 clock cycles

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

STACK CONTENTS



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

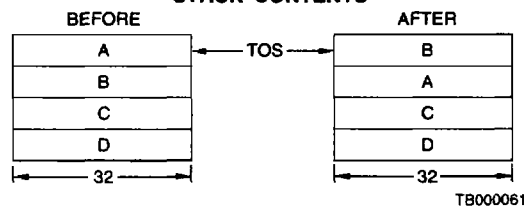
19 with sr = 0

Execution Time:

26 clock cycles

Description:

32-bit operand A at all 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

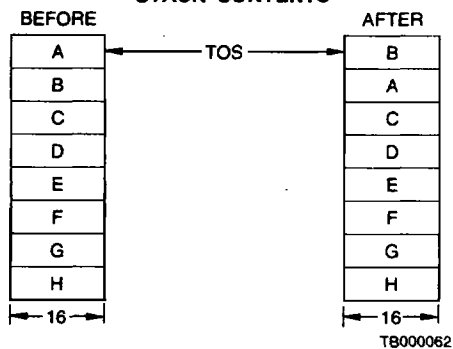
79 with sr = 0

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 operands are unchanged.

Status Affected: Sign, Zero**STACK CONTENTS**

APPLICATIONS INFORMATION

The diagram in Figure 2 shows the interface connections for the Am9511A APU with operand transfers handled by an Am9517A DMA controller and CPU coordination handled by an Am9519A Interrupt Controller. The APU interrupts the CPU to indicate that a command has been completed. When the performance enhancements provided by the DMA and Inter-

rupt operations are not required, the APU interface can be simplified as shown in Figure 1. The Am9511A APU is designed with a general purpose 8-bit data bus and interface control so that it can be conveniently used with any general 8-bit processor.

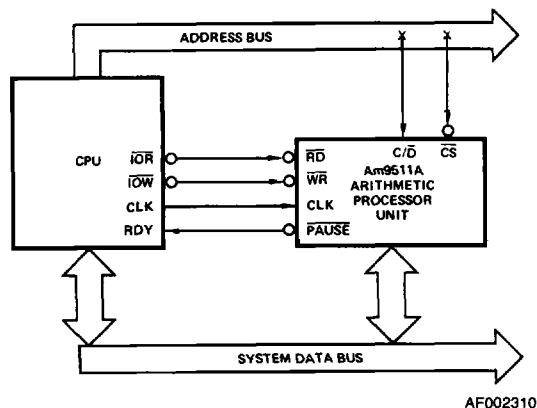


Figure 1. Am9511A Minimum Configuration Example.

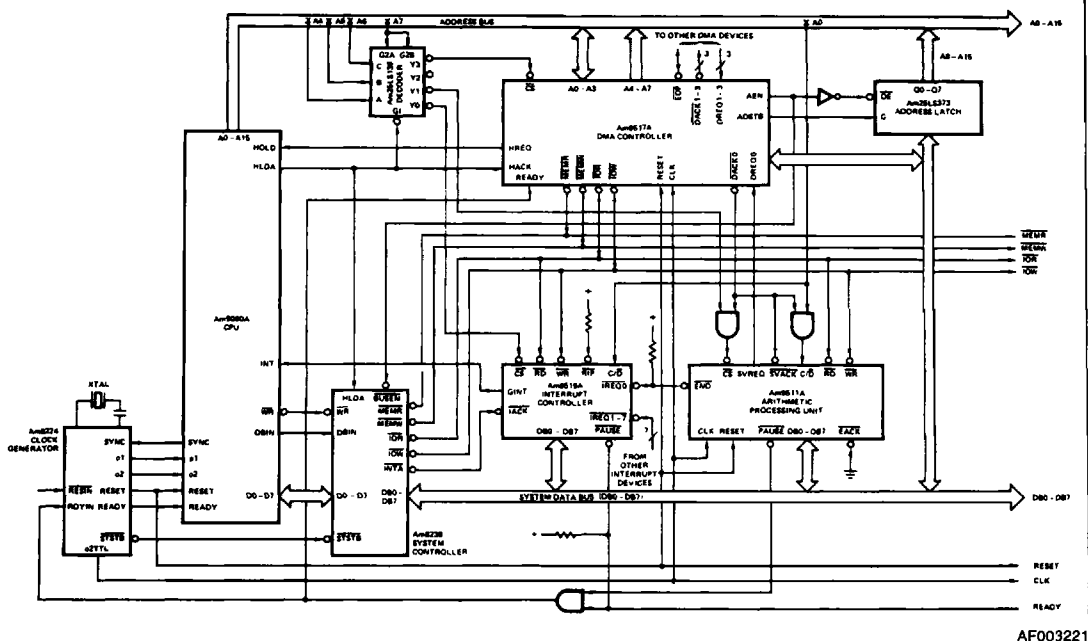


Figure 2. Am9511A High Performance Configuration Example.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature -65 to +150°C
 VDD with Respect to VSS -0.5V to +15.0V
 VCC with Respect to VSS -0.5V to +7.0V
 All Signal Voltages
 with Respect to VSS -5.0V to +7.0V
 Power Dissipation (Package Limitation) 2.0W

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Grade	T _A	V _{SS}	V _{CC}	V _{DD}
Commercial	0°C to 70°C	0V	5V ± 5%	12 ± 5%
Industrial	-40°C to 85°C	0V	5V ± 10%	12 ± 10%

Operating ranges define those limits over which the functionality of the device is guaranteed.

DC CHARACTERISTICS over operating range (Note 1)

Parameters	Description	Test Conditions	Min	Typ	Max	Units
VOH	Output HIGH Voltage	IOH = -200μA	3.7			Volts
VOL	Output LOW Voltage	IOL = 3.2mA			0.4	Volts
VIH	Input HIGH Voltage		2.0		VCC	Volts
VIL	Input LOW Voltage		-0.5		0.8	Volts
IIX	Input Load Current	VSS ≤ VI ≤ VCC			± 10	μA
IOZ	Data Bus Leakage	VO = 0.4V			10	μA
		VO = VCC			10	
ICC	VCC Supply Current	T _A = +25°C		50	90	mA
		T _A = 0°C			95	
		T _A = -55°C			100	
IDD	VDD Supply Current	T _A = +25°C		50	90	mA
		T _A = 0°C			95	
		T _A = -55°C			100	
CO	Output Capacitance	f _c = 1.0MHz, Inputs = 0V		8	10	pF
CI	Input Capacitance			5	8	pF
CIO	I/O Capacitance			10	12	pF

Note: 1. Typical values are for T_A = 25°C, normal supply voltages and normal processing parameters.

SWITCHING TEST INPUT WAVEFORM

WF004060

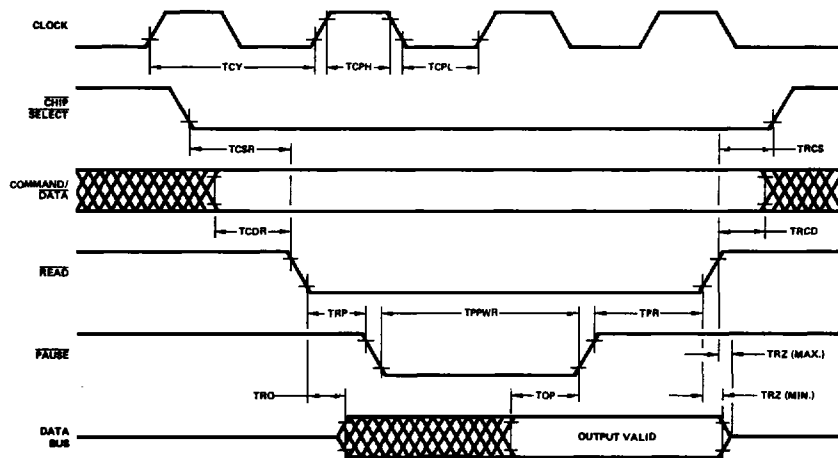
SWITCHING CHARACTERISTICS over operating range (Note 1)

Parameters	Description		Am9511A		Am9511A-1		Am9511A-4		Units
			Min	Max	Min	Max	Min	Max	
TAPW	EACK LOW Pulse Width		100		75		50		ns
TCDR	C/D to RD LOW Set-up Time		0		0		0		ns
TCDW	C/D to WR LOW Set-up Time		0		0		0		ns
TCPH	Clock Pulse HIGH Width		200		140		100		ns
TCPL	Clock Pulse LOW Width		240		160		120		ns
TCSR	CS LOW to RD LOW Set-up Time		0		0		0		ns
TCSW	CS LOW to WR LOW Set-up Time		0		0		0		ns
TCY	Clock Period		480	5000	320	3300	250	2500	ns
TDW	Data Bus Stable to WR HIGH Set-up Time		150		100 (Note 9)		100		ns
TEAE	EACK LOW to END HIGH Delay			200		175		150	ns
TEPW	END LOW Pulse Width (Note 4)		400		300		200		ns
TOP	Data Bus Output Valid to PAUSE HIGH Delay		0		0		0		ns
TPPWR	PAUSE LOW Pulse Width Read (Note 5)	Data	3.5TCY + 50	5.5TCY + 300	3.5TCY + 50	5.5TCY + 200	3.5TCY + 50	5.5TCY + 200	ns
		Status	1.5TCY + 50	3.5TCY + 300	1.5TCY + 50	3.5TCY + 200	1.5TCY + 50	3.5TCY + 200	
TPPWW	PAUSE LOW Pulse Width Write (Note 8)			50		50		50	ns
TPR	PAUSE HIGH to RD HIGH Hold Time		0		0		0		ns
TPW	PAUSE HIGH to WR HIGH Hold Time		0		0		0		ns
TRCD	RD HIGH to C/D Hold Time		0		0		0		ns
TRCS	RD HIGH to CS HIGH Hold Time		0		0		0		ns
TRO	RD LOW to Data Bus ON Delay		50		50		25		ns
TRP	RD LOW to PAUSE LOW Delay (Note 6)			150		100 (Note 9)		100	ns
TRZ	RD HIGH to Data Bus OFF Delay		50	200	50	150	25	100	ns
TSAPW	SVACK LOW Pulse Width		100		75		50		ns
TSAR	SVACK LOW to SVREQ LOW Delay			300		200		150	ns
TWCD	WR HIGH to C/D Hold Time		60		30		30		ns
TWCS	WR HIGH to CS HIGH Hold Time		60		30		30		ns
TWD	WR HIGH to Data Bus Hold Time		20		20		20		ns
TWI	Write Inactive Time	Command	3TCY		3TCY		3TCY		ns
		Data	4TCY		4TCY		4TCY		
TWP	WR LOW to PAUSE LOW Delay (Note 6)			150		100 (Note 9)		100	ns

- Notes: 1. Typical values are for $T_A = 25^\circ\text{C}$, nominal supply voltages and nominal processing parameters.
2. Switching parameters are listed in alphabetical order.
3. Test conditions assume transition times of 20ns or less, output loading of one TTL gate plus 100pF and timing reference levels of 0.8V and 2.0V.
4. END low pulse width is specified for EACK tied to VSS. Otherwise TEAE applies.
5. Minimum values shown assume no previously entered command is being executed for the data access. If a previously entered command is being executed, PAUSE LOW Pulse Width is the time to complete execution plus the time shown. Status may be read at any time without exceeding the time shown.
6. PAUSE is pulled low for both command and data operations.
7. TEX is the execution time of the current command (see the Command Execution Times table).
8. PAUSE low pulse width is less than 50ns when writing into the data port or the control port as long as the duty requirement (TWI) is observed and no previous command is being executed. TWI may be safely violated up to 500ns as long as the extended TPPWW that results is observed. If a previously entered command is being executed, PAUSE LOW Pulse Width is the time to complete execution plus the time shown.
9. 150ns for Military grade.

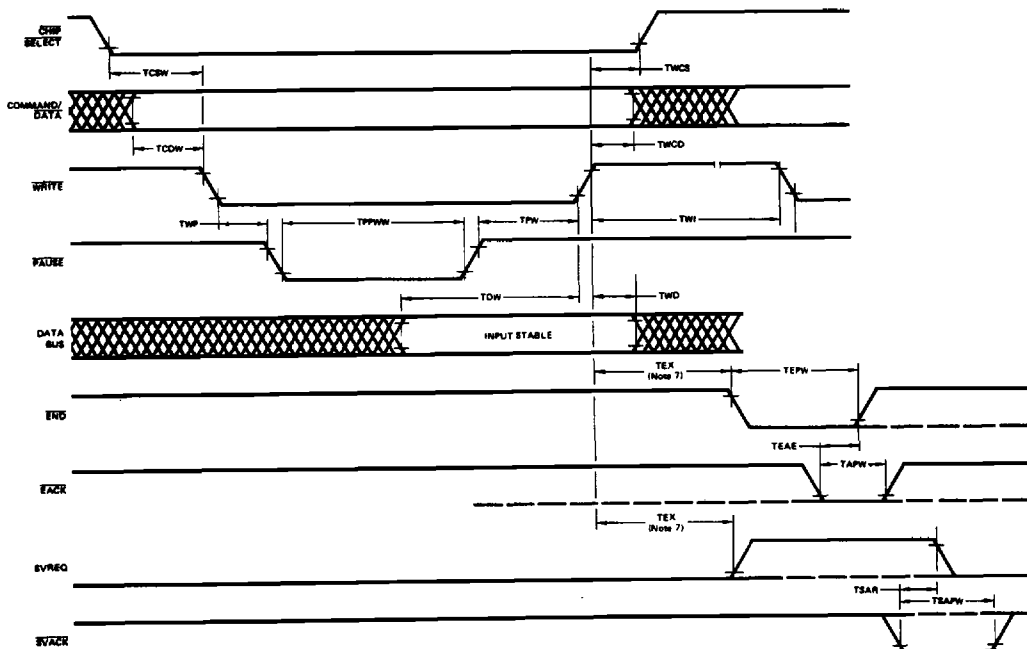
SWITCHING WAVEFORMS

READ OPERATIONS



WF004070

WRITE OPERATIONS



WF004080