

Floating-Point Arithmetic with the TMS32010

APPLICATION REPORT: SPRA002

*Author: Ray Simar, Jr.
Digital Signal Processing – Semiconductor Group*

*Digital Signal Processing Solutions
1989*



IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Floating-Point Arithmetic with the TMS32010

Abstract

This report presents algorithms and code implementing floating-point addition, subtraction, multiplication, and division with the TMS320. The support of floating-point operations by the TI processors has made possible some applications, such as the implementation of the CCITT Adaptive Differential Pulse Code Modulation (ADPCM) algorithm and image/graphics operations.



Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

INTRODUCTION

The TMS32010 Digital Signal Processor is a fixed-point 16/32-bit microprocessor. However, it can also perform floating-point computations at a speed comparable to dedicated floating-point processors.

The purpose of this application report is to analyze an implementation of floating-point addition and multiplication on the TMS32010. The floating-point single-precision standard proposed by the IEEE will be examined. Using this standard, the TMS32010 performs a floating-point multiplication in 8.4 microseconds and a floating-point addition in 17.2 microseconds.

To illustrate floating-point formats and the tradeoffs involved in making a choice between different floating-point formats, a review of floating-point arithmetic notation and of addition and multiplication algorithms is first presented.

FLOATING-POINT NOTATION

The floating-point number f may be written in floating-point format as

$$f = m \times b^e$$

where

m = mantissa

b = base

e = exponent

For example, 6,789,320 may be written as

$$0.6789320 \times 10^7$$

In this case,

$$m = 0.6789320$$

$$b = 10$$

$$e = 7$$

The two floating-point numbers f_1 and f_2 may be written as

$$f_1 = m_1 \times b^{e_1}$$

$$f_2 = m_2 \times b^{e_2}$$

Floating-point addition/subtraction, multiplication, and division for f_1 and f_2 are defined as follows:

$$f_1 \pm f_2 = (m_1 \pm m_2 \times b^{-(e_1 - e_2)}) \times b^{e_1} \quad \text{if } e_1 \geq e_2 \quad (1)$$

or

$$= (m_1 \times b^{-(e_2 - e_1)} \pm m_2) \times b^{e_2} \quad \text{if } e_1 < e_2$$

$$f_1 \times f_2 = m_1 \times m_2 \times b^{(e_1 + e_2)} \quad (2)$$

$$f_1 / f_2 = (m_1 / m_2) \times b^{(e_1 - e_2)} \quad (3)$$

A cursory examination of these expressions reveals some of the factors involved in the implementation of floating-point arithmetic. For addition, it is necessary to shift the mantissa of the floating-point number which has the smaller exponent to the right by the difference in the magnitude of the two exponents. This is shown in the multiplication by the terms

$$b^{-(e_1 - e_2)} \quad \text{and} \quad b^{-(e_2 - e_1)}$$

This right shift can result in mantissa underflow. There are also possibilities for mantissa overflow. Addition and subtraction of exponents can lead to exponent underflow and overflow. To alleviate underflow and overflow, it is necessary to decide on some scheme for roundoff. For a detailed description and analysis of underflow and overflow conditions and rounding schemes, see reference 1.

It is desirable to have all numbers normalized, i.e., the mantissas of f_1 and f_2 have the most significant digit in the leftmost position. This provides the representation with the greatest accuracy possible for a fixed mantissa length. The result of any floating-point operation must also be normalized. The factors associated with normalization, overflow, and other characteristics of floating-point implementations are best illustrated with a few examples.

Consider the addition of two binary floating-point numbers f_1 and f_2 where

$$f_1 = 0.10100 \times 2^{11}$$

$$f_2 = 0.11100 \times 2^{01}$$

Both of these numbers are normalized, i.e., the first bit after the binary point is a 1. Addition requires equal exponents, so the fractions are aligned by shifting right the one with the smaller exponent and adjusting the smaller exponent. This yields

$$f_2 = 0.00111 \times 2^{11}$$

Then,

$$\begin{aligned} f_1 + f_2 &= 0.10100 \times 2^{11} + 0.00111 \times 2^{11} \\ &= 0.11011 \times 2^{11} = f_3 \end{aligned}$$

The sum may overflow the left end by one digit, thus requiring a postaddition adjustment or renormalization step. Since it is assumed that the register is only of a finite length, this renormalization will result in the loss of the lowest order bit.

Another example illustrates the overflow past the most significant bit. With an assumed register length of five, let

$$\begin{aligned} f_1 &= 0.11100 \times 2^{11} \\ f_2 &= 0.10101 \times 2^{001} \end{aligned}$$

Then,

$$\begin{array}{r} 0.11100 \times 2^{11} = f_1 \\ + 0.0010101 \times 2^{11} = f_2 \\ \hline 1.0000101 \times 2^{11} = f_3 \end{array}$$

The significance of the two digits underlined in the right part of the mantissa is suspect, since it is assumed that the corresponding bits of f_1 are zero. The left underlined digit is the overflow past the most significant bit. To finish the addition, f_3 is shifted to the right and the exponent adjusted accordingly. Thus,

$$1.0000101 \times 2^{11} = f_3$$

The shift of the fraction and the adjustment of the exponent yield

$$0.10000101 \times 2^{100} = f_3$$

The result may be rounded, giving

$$0.10001 \times 2^{100} = f_3$$

or truncated, giving

$$0.10000 \times 2^{100} = f_3$$

FLOATING-POINT ALGORITHMS

Multiplication Algorithm

The algorithm for normalized floating-point multiplication is illustrated in Figure 1. This algorithm is an implementation of Equation 2 in the section on floating-point notation.

The floating-point numbers being multiplied are A and B written as

$$A = m_A \times b^{e_A} \quad \text{and} \quad B = m_B \times b^{e_B}$$

The result is

$$C = m_C \times b^{e_C}$$

For the resulting m_C , there are three special cases. The m_C may be zero, in which case there is a branch to Step 10 to set $C = 0$. If $m_C \neq 0$, then the most significant bit will be in either the first or second leftmost bit. If the most significant bit is in the second leftmost bit, then a left shift of m_C is necessary (see Step 5). Otherwise, C is already in normalized form, and there is a branch to Step 6.

Step 6 implements the desired rounding scheme. After this rounding, it is possible that m_C will overflow (see Step 7). In this case, it is necessary to right-shift m_C one bit (see Step 8). Step 9 checks for special cases of e_C . If there is an overflow or underflow of e_C , it is handled in Step 10. Otherwise, the result is in range, and the calculation is complete.

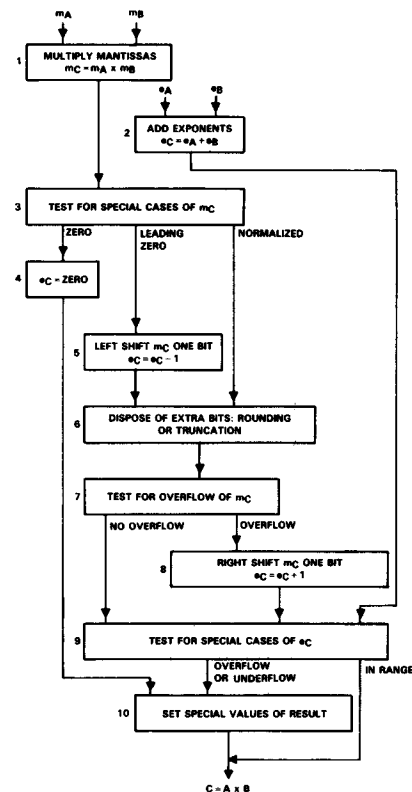


Figure 1. Floating-Point Multiplication

Addition Algorithm

The implementation of normalized floating-point addition is more involved than for multiplication. This addition algorithm, outlined in Figure 2, is an implementation of Equation 1 in the section on floating-point notation.

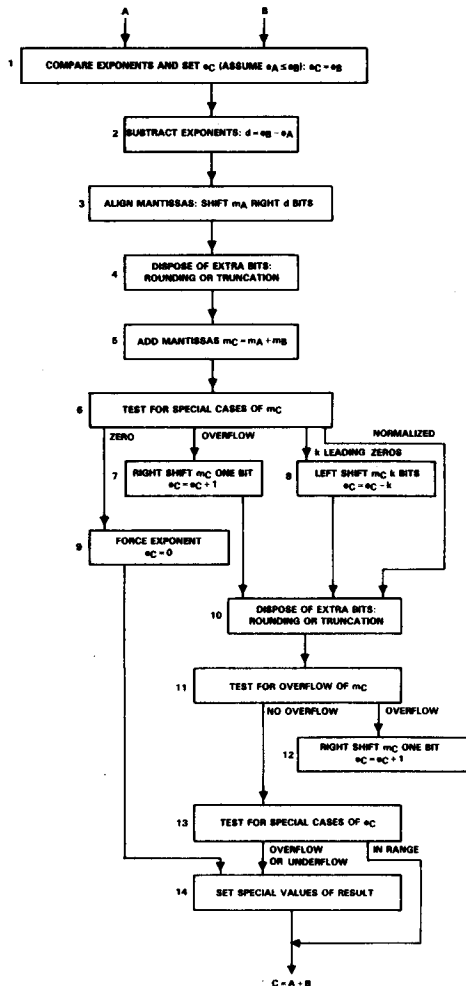


Figure 2. Floating-Point Addition

Step 1 compares e_A and e_B for determining e_C . For this illustration of the algorithm, it is assumed that $e_A \leq e_B$. Step 2 determines the right shift (d) that is required to align m_A . Step 3 implements this right shift of m_A . Step 4 disposes of the extra bits of m_A by using the desired rounding technique. The mantissas of A and B are then added in Step 5.

Now, things become somewhat more involved. The m_C may be zero, in which case there is a branch to Step 9 which sets $e_C = 0$; a branch to Step 14 sets the special value of the result. The m_C may overflow, in which case a right shift of one is necessary (see Step 7). The m_C may have k leading zeroes, in which case a left shift of k is required. This normalization step is generally the most involved and time-consuming step to perform. Steps 10, 11, and 12 round m_C , test for a possible overflow due to the rounding, and adjust e_C accordingly. Step 13 involves the determination of the special case of e_C . Finally, after Step 14, the sum $C = A + B$ is formed.

IEEE FLOATING-POINT SINGLE-PRECISION FORMAT

Of interest is a set of formats known as the IEEE standard. This IEEE recommended format consists of a variety of precision formats (single, double, single-extended, and double-extended). The IEEE has also proposed several techniques for handling special cases such as overflow, underflow, $\pm \infty$, and rounding. For complete details, the reader is referred to the proposed IEEE standard.²

The single-precision format is a 32-bit format consisting of a 1-bit sign field s , an 8-bit biased exponent e , and a 23-bit fraction f (see Figure 3). The value of a binary floating-point number X is determined as follows:

$$X = (-1)^s \times 2^{(e-127)} \times 1.f$$

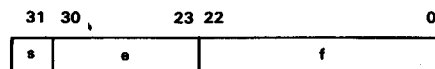


Figure 3. IEEE Floating-Point Single-Precision Format

The advantage of this format is that it is structured in such a way as to provide easy storage and straightforward input/output operations on 8-, 16- and 32-bit processors. The disadvantage with this format is that the large mantissa will generally span several words of memory.

FLOATING-POINT IMPLEMENTATION

IEEE Implementation

The IEEE single-precision format is described here as it applies to the addition and multiplication algorithms. In these floating-point routines written for the TMS32010, all results are truncated to 31 bits. This was done so that the user has more flexibility to develop a rounding scheme suitable for his application. The representations of $\pm \infty$ are ignored so that the user can decide how to handle these exceptions in a manner that is appropriate for his particular application.

I/O Considerations

The first consideration is the internal representation of the binary floating-point number. If the number is read into the TMS32010 as two 16-bit words, some processing is then necessary to put the floating-point number into a representation which is easier to process. The representation used in the TMS32010 programs in the Appendix is shown in Figure 4. This internal representation may be arrived at by a simple manipulation of the IEEE bit fields. For this particular algorithm, it is assumed that the floating-point number is input to the TMS32010 as the four 16-bit fields shown in Figure 4. However, the user can easily supply his own routine to arrive at this format from two 16-bit inputs to the TMS32010 where the inputs contain the IEEE single-precision format.

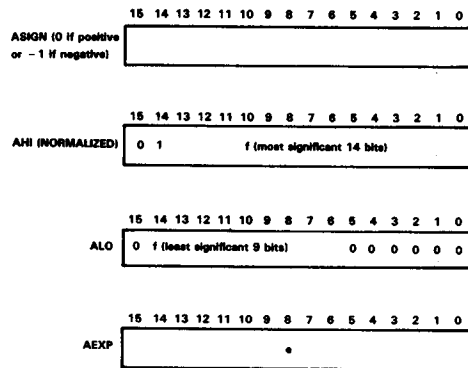


Figure 4. Floating-Point Representation

The format in Figure 4 was chosen to minimize the execution time of the floating-point addition and multiplication routines. The format of the result is shown in Figure 5. Notice that it is identical to the format in Figure 4 except for CLO. CLO has its 16 most significant bits valid for both the multiplication and addition routines.

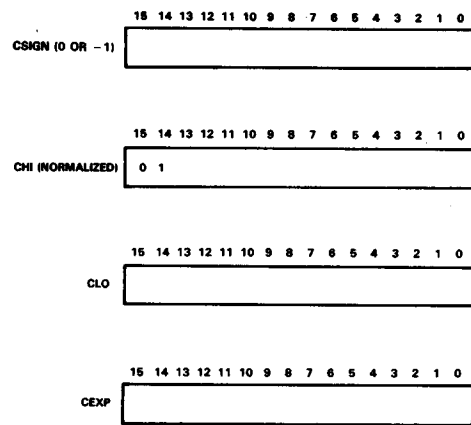


Figure 5. Result Representation

Normalization

Since the floating-point addition involves a normalization, a technique similar to a binary search algorithm is used in the addition routine in the Appendix. To begin the normalization routine, note that with the format used for the result (see Figure 5), all mantissas can be considered to be positive. The binary search for the most significant bit (the leftmost 1 since the mantissa is positive) is illustrated in Figure 6.

The first move is to split C into CHI and CLO. If CHI $\neq 0$, then the most significant bit (MSB) is in CHI; otherwise, it is in CLO. For this example, it is in CHI. The next step is to split CHI into C11 and C12. If C11 $\neq 0$, then the MSB is in the eight bits of C11; otherwise, it is in C12. For this example, the MSB is in C12. Next, split C12 into C23 and C24. Again, if C24 $\neq 0$, then the MSB is in C24; otherwise, it is in C23. Since C24 $\neq 0$, split C24 into C37 and C38. Since C37 $\neq 0$, the MSB is in C37. Finally, splitting C37, a simple bit test shows that the MSB is in position 19. Using this technique, it is possible to find the MSB in a 32-bit field with only five compares.

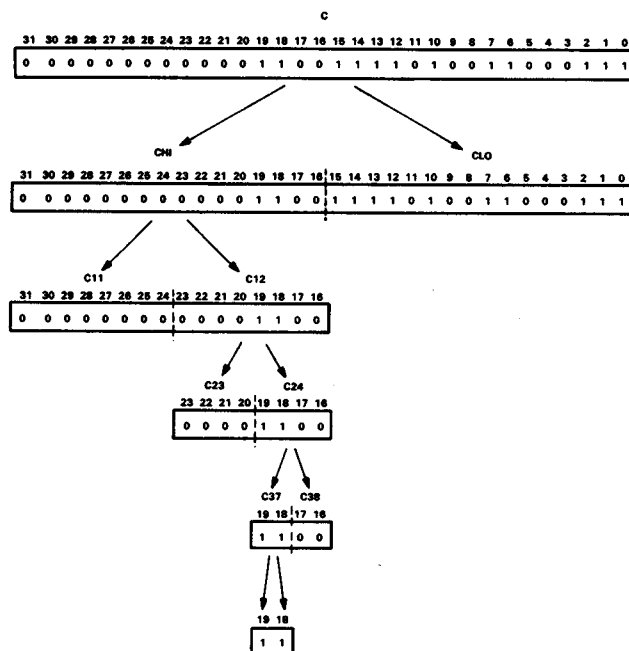


Figure 6. Binary Search

Added Precision

As illustrated in Figure 5, the 16 most significant bits of CLO are valid, i.e., C is valid for 31 places beyond the binary point. Oftentimes the user is not as concerned with the IEEE standard as in being certain that he has enough accuracy for his particular application. Since the TMS32010 uses 16-bit words, the routines in the Appendix implicitly maintain a 30-bit mantissa. They also implicitly use a 16-bit exponent. If the user desires this added accuracy and dynamic range, then it is readily implementable with no additional cost in execution time. The normalization for the addition, as mentioned previously, operates over the entire 32-bit accumulator. For the strict IEEE format, the user will only want to normalize over the 25 most significant bits of the accumulator. The structure of the normalization routine makes this modification simple.

The routines in the Appendix make no provision for the representations of $\pm \infty$ and exponent underflow and overflow. The user of the routines should consider the degree of significance of these results and the way they should be handled for his particular application. Since these routines are written to operate at maximum speed, truncation of results is used. If the user desires to implement a rounding scheme, then he will also need to check for the possibility of overflow due to the rounding scheme. This step is shown in the multiplication and addition flowcharts (see Figures 1 and 2).

SUMMARY

The TMS32010 may be used to perform floating-point operations with great accuracy, wide dynamic range, and high-speed execution. The design engineer has the responsibility of deciding what type of floating-point format is best for his application. To aid in understanding floating-point operations, several examples have been given illustrating the manipulations necessary to implement floating-point addition and multiplication algorithms. Flowcharts for these algorithms are also included. The Appendix contains the TMS32010 code for the IEEE floating-point single-precision format used in multiplication and addition. These same routines may also be used without modification to implement a format with up to a 30-bit mantissa and a 16-bit exponent without any increase in execution time.

REFERENCES

1. Kuck, D.J., THE STRUCTURE OF COMPUTERS AND COMPUTATIONS, Volume 1. New York: John Wiley & Sons, 1978.
2. Coonen, J. et al, "A Proposed Standard for Binary Floating-Point Arithmetic," ACM SIGNUM NEWSLETTER, October, 1979, 4-12.

APPENDIX

```

IEEEMULT      320 FAMILY MACRO ASSEMBLER  2.1 83.076      16:33:40   1/18/84

0001          *****
0002          *
0003          *   THIS IS A FLOATING-POINT MULTIPLICATION ROUTINE WHICH
0004          *   IMPLEMENTS THE IEEE PROPOSED FLOATING-POINT FORMAT ON
0005          *   THE TMS32010.
0006          *
0007          *****
0008          *
0009          *   INITIAL FORMAT (ALL 16 BIT WORDS)
0010          *   -----
0011          *   | ALL 0 OR 1 | ASIGN (0 OR -1)
0012          *   -----
0013          *
0014          *   -----
0015          *   |0|. 15 BITS | AHI (NORMALIZED)
0016          *   -----
0017          *
0018          *   -----
0019          *   |0| 9 BITS |--0--| ALO
0020          *   -----
0021          *
0022          *   -----
0023          *   | AEXP (-127 TO 128)
0024          *   -----
0025          *
0026          *   TO HAVE THIS CORRESPOND TO IEEE FORMAT, INPUT 0.1F *
0027          *   2 ** (E + 1) INSTEAD OF 1.F * 2 ** E AND SUBTRACT
0028          *   127 FROM E.
0029          *
0030          *   THE FINAL FORMAT IS THE SAME AS THE INITIAL FORMAT
0031          *   EXCEPT THAT FOR CLO WE HAVE:
0032          *   -----
0033          *   | 16 BITS | CLO
0034          *   -----
0035          *
0036          *
0037          *   THE 16 BITS OF CLO ARE VALID. ANYTHING PAST THESE
0038          *   HAS BEEN TRUNCATED.
0039          *
0040          *****
0041          *
0042          *   WORST CASE (EXCLUDING INITIALIZATION AND I/O):
0043          *   8.4 MICROSECONDS
0044          *   WORDS OF PROGRAM MEMORY: 72
0045          *
0046          *****
0047          *
0048          *   IDT 'IEEEMULT'
0049          *   DSEG
0050          *
0051          *   AEXP BSS 1
0052          *   AHI BSS 1
0053          *   ALO BSS 1
0054          *   ASIGN BSS 1
0055          *   BEXP BSS 1

```

```

0056 0005      BHI BSS 1
0057 0006      BLO BSS 1
0058 0007      BSIGN BSS 1
0059 0008      CEXP BSS 1
0060 0009      CHI BSS 1
0061 000A      CLO BSS 1
0062 000B      CSIGN BSS 1
0063 000C      TLO BSS 1
0064 000D      THI BSS 1
0065 000E      CLOHI BSS 1
0066 000F      M0003 BSS 1
0067 0010      ONE BSS 1
0068 0011      NEGONE BSS 1
0069 0012      TWO BSS 1
0070           *
0071 0013      DEND
0072           *
0073 0000      PSEG      * BEGIN THE PROGRAM SEGMENT *
0074           *
0075 0000 4003      IN ASIGN,PA0      * INPUT *
0076 0001 4000      IN AEXP,PA0
0077 0002 4001      IN AHI,PA0
0078 0003 4002      IN ALO,PA0
0079 0004 4007      IN BSIGN,PA0
0080 0005 4004      IN BEXP,PA0
0081 0006 4005      IN BHI,PA0
0082 0007 4006      IN BLO,PA0
0083           *      * FINISHED THE INPUT ROUTINE *
0084           *
0085           *      * A LITTLE INITIALIZATION *
0086 0008 6E00      START      LDPK 0
0087 0009 7E01      LACK 1
0088 000A 5010      SACL ONE
0089 000B 7E03      LACK 3
0090 000C 500F      SACL M0003
0091 000D 7F89      ZAC
0092 000E 1010      SUB ONE
0093 000F 5011      SACL NEGONE
0094 0010 7E02      LACK 2
0095 0011 5012      SACL TWO
0096           *      * DONE WITH THE INITIALIZATION *
0097           *      * ADD EXPONENTS *
0098 0012 2000      LAC AEXP
0099 0013 0004      ADD BEXP
0100 0014 5008      SACL CEXP      * CEXP = AEXP + BEXP *
0101           *      * FINISHED ADDING EXPONENTS *
0102           *      * MULTIPLY MANTISSAS *
0103 0015 6A02      LT ALO      * FIRST PRODUCT, (ALO * BHI) *
0104 0016 6D05      MPY BHI
0105 0017 7F8E      PAC
0106 0018 580D      SACH THI
0107 0019 500C      SACL TLO
0108           *
0109 001A 6A01      LT AHI      * SECOND PRODUCT, (AHI * BLO) *
0110 001B 6D06      MPY BLO
0111           *
0112 001C 7F8F      APAC      * (ALO * BHI + AHI * BLO) *

```

0113		*		
0114	001D 7F8F		APAC	* HAS THE EFFECT OF
0115		*		(AHI * BLO + ALO * BHI)
0116		*		* 2 ** -15 *
0117	001E 600D		ADDH THI	
0118	001F 610C		ADDS TLO	
0119	0020 580D		SACH THI	
0120		*		
0121	0021 6D05		MPY BHI	* (AHI * BHI) *
0122	0022 7F8E		PAC	
0123	0023 610D		ADDS THI	
0124		*		
0125	0024 5909		SACH CHI,1	* GET RID OF EXTRA SIGN BIT *
0126	0025 500A		SACL CLO	
0127		*		* THE (ALO * BLO * 2 ** -30) IS LOST DUE
0128		*		TO THE IEEE FORMAT *
0129		*		
0130		*		* FINISHED MULTIPLYING THE MANTISSAS *
0131		*		
0132		*		* CHECK SPECIAL CASES AND WRAP THINGS UP
0133	0026 FE00		BNZ OK	* CHI AND CLO ARE STILL IN THE ACC *
	0027 002C'			
0134		*		
0135	0028 7F89		ZAC	* IF C IS ZERO LOAD CEXP WITH ZERO *
0136	0029 5008		SACL CEXP	
0137	002A F900		B SETSIN	* BRANCH TO SET THE SIGN *
	002B 003A'			
0138		*		
0139	002C 210A	OK	LAC CLO,1	* TAKING CARE OF EXTRA SIGN BIT
0140		*		AS ABOVE *
0141	002D 500A		SACL CLO	
0142	002E 2E10		LAC ONE,14	* MASK OFF POSSIBLE MSB *
0143	002F 7909		AND CHI	
0144	0030 FE00		BNZ SETSIN	* BRANCH IF NORMALIZATION NOT
	0031 003A'			NECESSARY *
0145		*		
0146		*		
0147	0032 2008	SHIFT1	LAC CEXP	* HERE A LEFT SHIFT OF ONE IS
0148		*		NECESSARY *
0149	0033 1010		SUB ONE	
0150	0034 5008		SACL CEXP	
0151		*		
0152	0035 6509		ZALH CHI	
0153	0036 610A		ADDS CLO	
0154	0037 5909		SACH CHI,1	
0155	0038 210A		LAC CLO,1	
0156	0039 500A		SACL CLO	
0157		*		
0158	003A 6603	SETSIN	ZALS ASIGN	
0159	003B 7807		XOR BSIGN	
0160	003C FE00		BNZ NEG	* IF ASIGN XOR BSIGN != 0
	003D 0042'			
0161		*		THE PRODUCT IS NEGATIVE *
0162		*		
0163	003E 7F89		ZAC	
0164	003F 500B		SACL CSIGN	
0165	0040 F900		B OUTPUT	

```

      0041 0044'
0166      *
0167 0042 2011 NEG      LAC NEGONE      * NEGONE = -1 *
0168      *
0169 0043 500B      SACL CSIGN
0170      *
0171 0044 490B OUTPUT OUT CSIGN,PA1
0172 0045 4908      OUT CEXP,PA1
0173 0046 4909      OUT CHI,PA1
0174 0047 490A      OUT CLO,PA1
0175      *
0176 0048 F900 SELF      B SELF
      0049 0048'
0177      *
0178      *
0179      * END THE PROGRAM SEGMENT *
END
NO ERRORS, NO WARNINGS

```

```

0001 *****
0002 *
0003 * THIS IS A FLOATING POINT ADDITION ROUTINE WHICH
0004 * IMPLEMENTS THE IEEE PROPOSED FLOATING POINT FORMAT
0005 * ON THE TMS32010.
0006 *
0007 *****
0008 *
0009 * INITIAL FORMAT (ALL 16 BIT WORDS)
0010 *
0011 * | ALL 0 OR ALL 1 | ASSIGN (0 OR -1)
0012 * -----
0013 *
0014 *
0015 * | 0 | 15 BITS | AHI (NORMALIZED)
0016 * -----
0017 *
0018 *
0019 * | 0 | 9 BITS | --0- | ALO
0020 * -----
0021 *
0022 *
0023 * | | AEXP (-127 TO 128)
0024 * -----
0025 *
0026 * TO HAVE THIS CORRESPOND TO IEEE FORMAT, INPUT
0027 * 0.1F * 2 ** (E + 1)
0028 * INSTEAD OF 1.F * 2 ** E AND SUBTRACT 127 FROM E.
0029 *
0030 * THE FINAL FORMAT IS THE SAME AS THE INITIAL FORMAT
0031 * EXCEPT THAT FOR CLO WE HAVE:
0032 *
0033 *
0034 * | 16 BITS | CLO
0035 * -----
0036 *
0037 * ALL 16 BITS OF CLO ARE VALID. ANYTHING PAST THESE
0038 * HAS BEEN TRUNCATED.
0039 *
0040 *****
0041 *
0042 * WORST CASE (EXCLUDING INITIALIZATION AND I/O):
0043 * 17.2 MICROSECONDS.
0044 * THIS FIGURE INCLUDES THE NORMALIZATION.
0045 * WORDS OF PROGRAM MEMORY: 768
0046 *
0047 *****
0048 *
0049 IDT 'IEEEADD'
0050 0000 DORG 0
0051 *
0052 0000 AEXP BSS 1
0053 0001 AHI BSS 1
0054 0002 ALO BSS 1

```


0055 0003	ASIGN BSS 1
0056 0004	BEXP BSS 1
0057 0005	BHI BSS 1
0058 0006	BLO BSS 1
0059 0007	BSIGN BSS 1
0060 0008	CEXP BSS 1
0061 0009	CHI BSS 1
0062 000A	CLO BSS 1
0063 000B	CSIGN BSS 1
0064 000C	C11 BSS 1
0065 000D	C12 BSS 1
0066 000E	C21 BSS 1
0067 000F	C22 BSS 1
0068 0010	C31 BSS 1
0069 0011	C32 BSS 1
0070 0012	C33 BSS 1
0071 0013	C34 BSS 1
0072 0014	C23 BSS 1
0073 0015	C24 BSS 1
0074 0016	C35 BSS 1
0075 0017	C36 BSS 1
0076 0018	C37 BSS 1
0077 0019	C38 BSS 1
0078 001A	CTEMP BSS 1
0079 001B	D BSS 1
0080 001C	SHIFT1 BSS 1
0081 001D	SHIFT2 BSS 1
0082 001E	AHITL BSS 1
0083 001F	BHITL BSS 1
0084 0020	N1 BSS 1
0085 0021	N2 BSS 1
0086 0022	N3 BSS 1
0087 0023	N4 BSS 1
0088 0024	N5 BSS 1
0089 0025	N6 BSS 1
0090 0026	N7 BSS 1
0091 0027	N8 BSS 1
0092 0028	N9 BSS 1
0093 0029	N10 BSS 1
0094 002A	N11 BSS 1
0095 002B	N12 BSS 1
0096 002C	N13 BSS 1
0097 002D	N14 BSS 1
0098 002E	N15 BSS 1
0099 002F	N16 BSS 1
0100 0030	N17 BSS 1
0101 0031	N18 BSS 1
0102 0032	N19 BSS 1
0103 0033	N20 BSS 1
0104 0034	N21 BSS 1
0105 0035	N22 BSS 1
0106 0036	N23 BSS 1
0107 0037	N24 BSS 1
0108 0038	N25 BSS 1
0109 0039	N26 BSS 1
0110 003A	N27 BSS 1

```

0111 003B      N28 BSS 1
0112 003C      N29 BSS 1
0113 003D      N30 BSS 1
0114 003E      ONE BSS 1
0115 003F      TWO BSS 1
0116 0040      FIFTEEN BSS 1
0117 0041      SIXTEEN BSS 1
0118 0042      M0002 BSS 1
0119 0043      M0003 BSS 1
0120 0044      M000F BSS 1
0121 0045      M001F BSS 1
0122 0046      M007F BSS 1
0123 0047      M00FF BSS 1
0124 0048      M01FF BSS 1
0125 0049      M03FF BSS 1
0126 004A      M07FF BSS 1
0127 004B      M0FFF BSS 1
0128 004C      M1FFF BSS 1
0129 004D      M3FFF BSS 1
0130 004E      M7FFF BSS 1
0131 004F      M8000 BSS 1
0132 0050      OS BSS 1
0133          *
0134          *
0135          *      BEGIN THE PROGRAM SEGMENT *
0136          *
0137 0000          AORG >0
0138          *
0139 0000 F900      B START
          0001 0011
0140          *
0141 0002 0002      SHIFTS  DATA 2
0142 0003 0004      DATA 4
0143 0004 0008      DATA 8
0144 0005 0010      DATA 16
0145 0006 0020      DATA 32
0146 0007 0040      DATA 64
0147 0008 0080      DATA 128
0148 0009 0100      DATA 256
0149 000A 0200      DATA 512
0150 000B 0400      DATA 1024
0151 000C 0800      DATA 2048
0152 000D 1000      DATA 4096
0153 000E 2000      DATA 8192
0154 000F 4000      DATA 16384
0155 0010 8000      DATA 32768
0156          *
0157          *      * A LITTLE INITIALIZATION *
0158 0011 6E00      START  LDPK 0
0159 0012 7E01      LACK 1
0160 0013 503E      SACL ONE
0161 0014 7020      LARK ARO,N1
0162 0015 711D      LARK ARI,29
0163 0016 7F89      ZAC
0164 0017 103E      LOOP  SUB ONE
0165 0018 6880      LARP ARO

```

0166 0019 50A1	SACL *+,0,ARI	
0167 001A F400	BANZ LOOP	
001B 0017		
0168 001C 7E02	LACK 2	
0169 001D 5042	SACL M0002	
0170 001E 503F	SACL TWO	
0171 001F 7E03	LACK 3	
0172 0020 5043	SACL M0003	
0173 0021 7E0F	LACK 15	
0174 0022 5044	SACL M000F	
0175 0023 5040	SACL FIFTEEN	
0176 0024 7E10	LACK 16	
0177 0025 5041	SACL SIXTEN	
0178 0026 7E1F	LACK 31	
0179 0027 5045	SACL M001F	
0180 0028 7E7F	LACK 127	
0181 0029 5046	SACL M007F	
0182 002A 2444	LAC M000F,4	
0183 002B 0044	ADD M000F	
0184 002C 5047	SACL M00FF	
0185 002D 083E	ADD ONE,8	
0186 002E 5048	SACL M01FF	
0187 002F 093E	ADD ONE,9	
0188 0030 5049	SACL M03FF	
0189 0031 0A3E	ADD ONE,10	
0190 0032 504A	SACL M07FF	
0191 0033 083E	ADD ONE,11	
0192 0034 504B	SACL M0FFF	
0193 0035 0C3E	ADD ONE,12	
0194 0036 504C	SACL M1FFF	
0195 0037 003E	ADD ONE,13	
0196 0038 504D	SACL M3FFF	
0197 0039 0E3E	ADD ONE,14	
0198 003A 504E	SACL M7FFF	
0199 003B 2F3E	LAC ONE,15	
0200 003C 504F	SACL M8000	
0201 003D 6A3E	LT ONE	
0202 003E 8002	MPYK SHIFTS	
0203 003F 7F8E	PAC	
0204 0040 5050	SACL OS	
0205	*	
0206	*	
0207	*	* FINISHED INITIALIZATION *
0208 0041 4003	IN ASIGN,PA0	
0209 0042 4000	IN AEXP,PA0	
0210 0043 4001	IN AHI,PA0	
0211 0044 4002	IN ALO,PA0	
0212 0045 4007	IN BSIGN,PA0	
0213 0046 4004	IN BEXP,PA0	
0214 0047 4005	IN BHI,PA0	
0215 0048 4006	IN BLO,PA0	
0216	*	
0217 0049 2000	LAC AEXP	
0218 004A 1004	SUB BEXP	
0219 004B FA00	BLZ ALTB	* BRANCH IF AEXP < BEXP *
004C 0091		

```

0220      *
0221 004D FF00      BZ AEQB      * BRANCH IF AEXP = BEXP *
      004E 00E8
0222      *
0223 004F 501B      SACL D      * D IS THE RIGHT SHIFT NEEDED FOR B *
0224      *
0225 0050 2000      LAC AEXP
0226 0051 5008      SACL CEXP      * THE EXP FOR THE RESULT IS AEXP *
0227      *
0228 0052 2040 AGTB  LAC FIFTEN      * A > B SO SHIFT B TO THE RIGHT D *
0229 0053 101B      SUB D
0230 0054 FB00      BLEZ BHIZER      * IF D >= 15 BHI IS ZERO *
      0055 0069
0231      *
0232 0056 0050      ADD OS
0233 0057 671C      TBLR SHIFTL      | 0 | 15 BITS | BHI
0234      *
0235      *
0236      *      | 0 | 8 BITS | | BLO
0237      *
0238 0058 6A05      LT BHI      || IS
0239 0059 6D1C      MPY SHIFTL      || CHANGED
0240 005A 7F8E      PAC      VV TO
0241 005B 7F88      ABS      VV
0242 005C 5805      SACH BHI      VV
0243 005D 501F      SACL BHITL
0244      *      | -0- | 15-0 | BHI
0245 005E 6A06      LT BLO
0246 005F 6D1C      MPY SHIFTL
0247 0060 7F8E      PAC      | D | 16-0 | BLO
0248 0061 7F8F      APAC
0249 0062 7F88      ABS
0250 0063 601F      ADDH BHITL
0251 0064 5806      SACH BLO
0252 0065 2102      LAC ALO,1
0253 0066 5002      SACL ALO
0254      *
0255 0067 F900      B DUN      * FINISHED SHIFT OF B BY D TO THE RIGHT
      0068 00EE
0256      *
0257 0069 003E BHIZER ADD ONE      * IF D >= 16 BLO LOSES BITS *
0258 006A FB00      BLEZ BLOLUZ
      006B 0077
0259      *      * WE ONLY GET HERE IF D = 15 *
0260      *
0261      *      | 0 | 15 BITS | BHI
0262 006C 2206      LAC BLO,2
0263 006D 5806      SACH BLO
0264 006E 6606      ZALS BLO      | 0 | 8 BITS | | BLO
0265 006F 0105      ADD BHI,1
0266 0070 5006      SACL BLO      ||
0267 0071 7F89      ZAC      VV
0268 0072 5005      SACL BHI
0269      *      | -0- | BHI
0270      *

```

```

0271      *
0272      *
0273      *
0274 0073 2102      LAC ALO,I
0275 0074 5002      SACL ALO
0276      *
0277 0075 F900      B DUN      * FINISHED SHIFT OF B
0278      *                  BY D TO THE RIGHT *
0279 0077 FF00 BLOLUZ BZ SHB16
0280 0079 0040      ADD FIFTEN
0281 007A FB00      BLEZ BZERO      IF D >= 31, THEN B IS ZERO.
0282      *
0283 007C 0050      ADD OS      {0} 15 BITS { } BHI
0284 007D 671C      TBLR SHIF1
0285      *
0286 007E 6A05      LT BHI      {0} 8 BITS { } BLO
0287 007F 6D1C      MPY SHIF1
0288 0080 7F8E      PAC
0289 0081 7F88      ABS
0290 0082 5806      SACH BLO
0291 0083 7F89      ZAC
0292 0084 5005      SACL BHI      {0} ---0--- { } BHI
0293      *
0294      *
0295      *
0296      *
0297 0085 2102      LAC ALO,I
0298 0086 5002      SACL ALO
0299      *
0300 0087 F900      B DUN
0301      *
0302 0089 2005 SHB16 LAC BHI
0303 008A 5006      SACL BLO
0304 008B 7F89      ZAC
0305 008C 5005      SACL BHI
0306      *
0307 008D 2102      LAC ALO,I
0308 008E 5002      SACL ALO
0309      *
0310 008F F900      B DUN
0311      *
0312 0091 7F88 ALTB  ABS      * TO SEE WHAT IS GOING ON LOOK AT
0313      *                  THE PREVIOUS CASE FOR AGTB *
0314 0092 501B      SACL D
0315      *
0316 0093 2004      LAC BEXP
0317 0094 5008      SACL CEXP
0318      *
0319 0095 2040      LAC FIFTEN
0320 0096 101B      SUB D
0321 0097 FB00      BLEZ AHIZER

```

```

0098 00AC
0322 *
0323 0099 0050 ADD OS
0324 009A 671C TBLR SHIFT1
0325 *
0326 009B 6A01 LT AHI
0327 009C 6D1C MPY SHIFT1
0328 009D 7F8E PAC
0329 009E 7F88 ABS
0330 009F 5801 SACH AHI
0331 00A0 501E SACL AHITL
0332 *
0333 00A1 6A02 LT ALO
0334 00A2 6D1C MPY SHIFT1
0335 00A3 7F8E PAC
0336 00A4 7F8F APAC
0337 00A5 7F88 ABS
0338 00A6 601E ADDH AHITL
0339 00A7 5802 SACH ALO
0340 00A8 2106 LAC BLO,1
0341 00A9 5006 SACL BLO
0342 *
0343 00AA F900 B DUN
00AB 00EE
0344 *
0345 00AC 003E AHIZER ADD ONE
0346 00AD FB00 BLEZ ALOLUZ
00AE 00BA
0347 *
0348 00AF 2202 LAC ALO,2
0349 00B0 5802 SACH ALO
0350 00B1 2002 LAC ALO
0351 00B2 0101 ADD AHI,1
0352 00B3 5002 SACL ALO
0353 00B4 7F89 ZAC
0354 00B5 5001 SACL AHI
0355 00B6 2106 LAC BLO,1
0356 00B7 5006 SACL BLO
0357 *
0358 00B8 F900 B DUN
00B9 00EE
0359 *
0360 00BA FF00 ALOLUZ BZ SHA16
00BB 00CC
0361 00BC 0040 ADD FIFTEN
0362 00BD FB00 BLEZ AZERO
00BE 0004
0363 *
0364 00BF 0050 ADD OS
0365 00C0 671C TBLR SHIFT1
0366 *
0367 00C1 6A01 LT AHI
0368 00C2 6D1C MPY SHIFT1
0369 00C3 7F8E PAC
0370 00C4 7F88 ABS
0371 00C5 5802 SACH ALO

```

```

0372 00C6 7F89      ZAC
0373 00C7 5001      SACL AHI
0374 00C8 2106      LAC BLO,I
0375 00C9 5006      SACL BLO
0376                *
0377 00CA F900      B DUN
      00CB 00EE
0378                *
0379 00CC 2001  SHA16 LAC AHI
0380 00CD 5002      SACL ALO
0381 00CE 7F89      ZAC
0382 00CF 5001      SACL AHI
0383                *
0384 00D0 2106      LAC BLO,I
0385 00D1 5006      SACL BLO
0386                *
0387 00D2 F900      B DUN
      00D3 00EE
0388                *
0389 00D4 2005  AZERO LAC BHI
0390 00D5 5009      SACL CHI
0391 00D6 2004      LAC BEXP
0392 00D7 5008      SACL CEXP
0393 00D8 2106      LAC BLO,I
0394 00D9 500A      SACL CLO
0395 00DA 2007      LAC BSIGN
0396 00DB 5008      SACL CSIGN
0397 00DC F900      B OUTPUT
      00DD 02FB
0398                *
0399 00DE 2001  BZERO LAC AHI
0400 00DF 5009      SACL CHI
0401 00E0 2000      LAC AEXP
0402 00E1 5008      SACL CEXP
0403 00E2 2102      LAC ALO,I
0404 00E3 500A      SACL CLO
0405 00E4 2003      LAC ASIGN
0406 00E5 5008      SACL CSIGN
0407 00E6 F900      B OUTPUT
      00E7 02FB
0408                *
0409 00E8 2102  AEQB  LAC ALO,I
0410 00E9 5002      SACL ALO
0411 00EA 2106      LAC BLO,I
0412 00EB 5006      SACL BLO
0413 00EC 2000      LAC AEXP
0414 00ED 5008      SACL CEXP
0415                *
0416                *
0417                *
0418 00EE 2003  DUN  LAC ASIGN
0419 00EF 1007      SUB BSIGN
0420 00F0 FE00      BNZ DIFSIN
      00F1 0113
0421                *
0422 00F2 6501      ZALH AHI

```

* GO TO DUN *

* BRANCH IF THERE IS A
SIGN DIFFERENCE *

```

0423 00F3 6102      ADDS ALO
0424 00F4 6106      ADDS BLO
0425 00F5 6005      ADDH BHI
0426                *
0427 00F6 5809      SACH CHI
0428 00F7 500A      SACL CLO
0429 00F8 FF00      BZ CZERO
0429 00F9 0131
0431                *
0432 00FA FD00      BGEZ NOOV
0432 00FB 010F
0433                *
0434 00FC 2F09      LAC CHI,15
0435 00FD 5809      SACH CHI
0436 00FE 501A      SACL CTEMP
0437                *
0438 00FF 2009      LAC CHI
0439 0100 794E      AND M7FFF
0440 0101 5009      SACL CHI
0441                *
0442 0102 2F0A      LAC CLO,15
0443 0103 580A      SACH CLO
0444 0104 200A      LAC CLO
0445 0105 794E      AND M7FFF
0446 0106 611A      ADDS CTEMP
0447 0107 500A      SACL CLO
0448                *
0449 0108 2008      LAC CEXP
0450 0109 003E      ADD ONE
0451 010A 5008      SACL CEXP
0452                *
0453 010B 2003      LAC ASIGN
0454 010C 500B      SACL CSIGN
0455                *
0456 010D F900      B OUTPUT
0456 010E 02FB
0457                *
0458 010F 2003      NOOV LAC ASIGN
0459 0110 500B      SACL CSIGN
0460                *
0461 0111 F900      B NORM
0461 0112 013D
0462                *
0463 0113 FA00      DIFSIN BLZ CHAS
0463 0114 0123
0464                *
0465 0115 6501      CHBS ZALH AHI
0466 0116 6102      ADDS ALO
0467 0117 6306      SUBS BLO
0468 0118 6205      SUBH BHI
0469                *
0470 0119 FF00      BZ CZERO
0470 011A 0131
0471 011B FA00      BLZ CNEG
0471 011C 0138
0472                *

```



```

0473 011D 5809      SACH CHI
0474 011E 500A      SACL CLO
0475                *
0476 011F 7F89      ZAC
0477 0120 500B      SACL CSIGN
0478                *
0479 0121 F900      B NORM
0122 0130
0480                *
0481 0123 6505 CHAS  ZALH BHI      * DO (!B! - !A!) SINCE A < 0 AND B > 0
0482 0124 6106      ADDS BLO
0483 0125 6302      SUBS ALO
0484 0126 6201      SUBH AHI
0485                *
0486 0127 FF00      BZ CZERO
0128 0131
0487 0129 FA00      BLZ CNEG
012A 0138
0488                *
0489 012B 5809      SACH CHI
0490 012C 500A      SACL CLO
0491                *
0492 012D 7F89      ZAC
0493 012E 500B      SACL CSIGN
0494                *
0495 012F F900      B NORM
0130 013D
0496                *
0497 0131 7F89 CZERO ZAC
0498 0132 5008      SACL CEXP
0499 0133 5009      SACL CHI
0500 0134 500A      SACL CLO
0501 0135 500B      SACL CSIGN
0502 0136 F900      B OUTPUT
0137 02FB
0503                *
0504 0138 7F88 CNEG  ABS
0505 0139 5809      SACH CHI
0506 013A 500A      SACL CLO
0507 013B 2020      LAC NI
0508 013C 500B      SACL CSIGN
0509                *
0510                *
0511                *
0512                *
0513                *
0514                *
0515                *
0516                *
0517                *
0518                *
0519                *
0520                *
0521                *
0522                *
0523                *

```

NORM DOES THE NORMALIZATION. THAT IS TO SAY THAT IT
FINDS THE MSB OF CHI AND CLO. IN THIS CASE THE MSB WILL
BE THE FIRST ONE (1) FOUND. THE SEARCH FOR THIS SPECIAL
ONE IS DONE WITH A BINARY SEARCH. THE NOTATION USED
IS SUMMARIZED HERE:

!----- 16 BITS -----!

```

0524 *      |-----| CHI OR CLO |-----| CHI OR  

0525 *      |                                     | CLO  

0526 *      |-----|-----|-----|-----| 8  

0527 *      |                                     | BITS  

0528 *      |          C11          I          C12          | EACH  

0529 *      |-----|-----|-----|-----| 4  

0530 *      |          C21   I   C22   |   C23   I   C24   | BITS  

0531 *      |-----|-----|-----|-----| 2  

0532 *      |          C31 | C32 | C33 | C34 | C35 | C36 | C37 | C38 | | EACH  

0533 *      |-----|-----|-----|-----| 1  

0534 *      |          C31 | C32 | C33 | C34 | C35 | C36 | C37 | C38 | | BIT  

0535 *      |-----|-----|-----|-----| 1  

0536 *      |          C31 | C32 | C33 | C34 | C35 | C36 | C37 | C38 | | EACH  

0537 *      |-----|-----|-----|-----|  

0538 *      |          C31 | C32 | C33 | C34 | C35 | C36 | C37 | C38 | |  

0539 *      |-----|-----|-----|-----|  

0540 *      THE 1'S REPRESENT THE BOUNDARY BETWEEN THE ACC HIGH BIT  

0541 *      AND THE ACC LOW BITS WHEN THE HIGHER LEVEL IS LOADED INTO  

0542 *      THE ACC WITH THE NECESSARY SHIFT TO SPLIT BY A FACTOR OF  

0543 *      TWO.  

0544 *      WORST CASE: 6.4 MICROSECONDS.  

0545 *  

0546 *****  

0547 013D 2809 NORM LAC CHI,8  

0548 013E FF00 BZ CLOB * BRANCH IF MSB IS IN CLO *  

0549 013F 0228  

0550 *  

0550 0140 580C SACH C11 * SPLIT THE 16 BIT WORD INTO TWO  

0551 0141 500D SACL C12 8 BIT WORDS *  

0552 *  

0553 0142 2C0C LAC C11,12  

0554 0143 FF00 BZ C12B * IF THERE IS A ONE IN C11 IT WILL  

0555 0144 01A9 BE NONZERO *  

0556 *  

0556 0145 580E SACH C21  

0557 0146 500F SACL C22  

0558 *  

0559 0147 2E0E LAC C21,14  

0560 0148 FF00 BZ C22B * IF THERE IS A ONE ETC. *  

0561 0149 016C  

0562 *  

0562 014A 5810 SACH C31  

0563 014B 5011 SACL C32  

0564 *  

0565 014C 2010 LAC C31  

0566 014D FF00 BZ C32B * IF THERE IS A ONE ETC. *  

0567 014E 0151  

0568 *  

0568 014F F900 B OUTPUT * THE MSB CANNOT BE IN BIT ONE BECAUSE  

0569 0150 02FB THIS WAS HANDLED EARLIER *  

0570 *  

0570 0151 2011 C32B LAC C32  

0571 0152 794F AND M8000  

0572 0153 FF00 BZ MSB4 * MSB# MEANS THE MSB IS IN BIT # FROM  

0573 0154 0160 LEFT TO RIGHT *

```

```

0574 *
0575 0155 2020 LAC N1 * LEFT SHIFT OF 1 *
0576 0156 0008 ADD CEXP
0577 0157 5008 SACL CEXP
0578 *
0579 0158 6509 ZALH CHI
0580 0159 610A ADDS CLO
0581 015A 6009 ADDH CHI
0582 015B 610A ADDS CLO
0583 015C 5809 SACH CHI
0584 015D 500A SACL CLO
0585 *
0586 015E F900 B OUTPUT
      015F 02FB
0587 *
0588 0160 2021 MSB4 LAC N2 * LEFT SHIFT OF 2 *
0589 0161 0008 ADD CEXP
0590 0162 5008 SACL CEXP
0591 *
0592 0163 220A LAC CLO,2
0593 0164 500A SACL CLO
0594 0165 581A SACH CTEMP
0595 0166 201A LAC CTEMP
0596 0167 7943 AND M0003
0597 0168 0209 ADD CHI,2
0598 0169 5009 SACL CHI
0599 *
0600 016A F900 B OUTPUT
      016B 02FB
0601 *
0602 016C 220F C22B LAC C22,2
0603 016D 5812 SACH C33
0604 016E 5013 SACL C34
0605 *
0606 016F 2012 LAC C33
0607 0170 FF00 BZ C34B * BRANCH IF NO ONE *
      0171 0180
0608 *
0609 0172 7942 AND M0002
0610 0173 FF00 BZ MSB6
      0174 0181
0611 *
0612 0175 2022 LAC N3 * LEFT SHIFT OF THREE *
0613 0176 0008 ADD CEXP
0614 0177 5008 SACL CEXP
0615 0178 230A LAC CLO,3
0616 0179 500A SACL CLO
0617 017A 581A SACH CTEMP
0618 017B 201A LAC CTEMP
0619 017C 7944 AND M000F
0620 017D 0309 ADD CHI,3
0621 017E 5009 SACL CHI
0622 *
0623 017F F900 B OUTPUT
      0180 02FB
0624 *

```

```

0625 0181 2023 MSB6 LAC N4 * LEFT SHIFT OF 4 *
0626 0182 0008 ADD CEXP
0627 0183 5008 SACL CEXP
0628 *
0629 0184 240A LAC CLO,4
0630 0185 500A SACL CLO
0631 0186 581A SACH CTEMP
0632 0187 201A LAC CTEMP
0633 0188 7944 AND M000F
0634 0189 0409 ADD CHI,4
0635 018A 5009 SACL CHI
0636 *
0637 018B F900 B OUTPUT
018C 02FB
0638 *
0639 018D 2013 C34B LAC C34
0640 *
0641 018E 794F AND M8000
0642 018F FF00 BZ MSB8
0190 019D
0643 *
0644 0191 2024 LAC N5 * LEFT SHIFT OF 5 *
0645 0192 0008 ADD CEXP
0646 0193 5008 SACL CEXP
0647 *
0648 0194 250A LAC CLO,5
0649 0195 500A SACL CLO
0650 0196 581A SACH CTEMP
0651 0197 201A LAC CTEMP
0652 0198 7945 AND M001F
0653 0199 0509 ADD CHI,5
0654 019A 5009 SACL CHI
0655 *
0656 019B F900 B OUTPUT
019C 02FB
0657 *
0658 019D 2025 MSB8 LAC N6 * LEFT SHIFT OF 6 *
0659 019E 0008 ADD CEXP
0660 019F 5008 SACL CEXP
0661 *
0662 01A0 260A LAC CLO,6
0663 01A1 500A SACL CLO
0664 01A2 581A SACH CTEMP
0665 01A3 201A LAC CTEMP
0666 01A4 7945 AND M001F
0667 01A5 0609 ADD CHI,6
0668 01A6 5009 SACL CHI
0669 *
0670 01A7 F900 B OUTPUT * ^^ COMPLETES TOP 8 BITS ^^ *
01A8 02FB
0671 *
0672 01A9 240D C12B LAC C12,4
0673 01AA 5814 SACH C23
0674 01AB 5015 SACL C24
0675 *
0676 01AC 2E14 LAC C23,14

```

```

0677 01AD FF00      BZ C24B
      01AE 01EB
0678 *
0679 01AF 5816      SACH C35
0680 01B0 5017      SACL C36
0681 *
0682 01B1 2016      LAC C35
0683 01B2 FF00      BZ C36B
      01B3 01CF
0684 *
0685 01B4 7942      AND M0002
0686 01B5 FF00      BZ MSB10
      01B6 01C3
0687 *
0688 01B7 2026      LAC N7      * LEFT SHIFT OF 7 *
0689 01B8 0008      ADD CEXP
0690 01B9 5008      SACL CEXP
0691 *
0692 01BA 270A      LAC CLO,7
0693 01BB 500A      SACL CLO
0694 01BC 581A      SACH CTEMP
0695 01BD 201A      LAC CTEMP
0696 01BE 7946      AND M007F
0697 01BF 0709      ADD CHI,7
0698 01C0 5009      SACL CHI
0699 *
0700 01C1 F900      B OUTPUT
      01C2 02FB
0701 *
0702 01C3 2027      MSB10      LAC N8      * LEFT SHIFT OF 8 *
0703 01C4 0008      ADD CEXP
0704 01C5 5008      SACL CEXP
0705 *
0706 01C6 280A      LAC CLO,8
0707 01C7 500A      SACL CLO
0708 01C8 581A      SACH CTEMP
0709 01C9 201A      LAC CTEMP
0710 01CA 7947      AND M00FF
0711 01CB 0809      ADD CHI,8
0712 01CC 5009      SACL CHI
0713 01CD F900      B OUTPUT
      01CE 02FB
0714 *
0715 01CF 2017      C36B      LAC C36
0716 01D0 794F      AND M8000
0717 01D1 FF00      BZ MSB12
      01D2 01DF
0718 01D3
0719 01D3 2028      LAC N9      * LEFT SHIFT OF 9 *
0720 01D4 0008      ADD CEXP
0721 01D5 5008      SACL CEXP
0722 01D6 290A      LAC CLO,9
0723 01D7 500A      SACL CLO
0724 01D8 581A      SACH CTEMP
0725 01D9 201A      LAC CTEMP
0726 01DA 7948      AND M01FF

```

0727	010B	0909		ADD CHI,9	
0728	010C	5009		SACL CHI	
0729			*		
0730	010D	F900		B OUTPUT	
	010E	02FB			
0731			*		
0732	010F	2029	MSB12	LAC N10	* LEFT SHIFT OF 10 *
0733	01E0	0008		ADD CEXP	
0734	01E1	5008		SACL CEXP	
0735			*		
0736	01E2	2A0A		LAC CLO,10	
0737	01E3	500A		SACL CLO	
0738	01E4	581A		SACH CTEMP	
0739	01E5	201A		LAC CTEMP	
0740	01E6	7949		AND M03FF	
0741	01E7	0A09		ADD CHI,10	
0742	01E8	5009		SACL CHI	
0743			*		
0744	01E9	F900		B OUTPUT	
	01EA	02FB			
0745			*		
0746	01EB	2215	C24B	LAC C24,2	
0747	01EC	5818		SACH C37	
0748	01ED	5019		SACL C38	
0749			*		
0750	01EE	2018		LAC C37	
0751	01EF	FF00		BZ C38B	
	01F0	020C			
0752			*		
0753	01F1	7942		AND M0002	
0754	01F2	FF00		BZ MSB14	
	01F3	0200			
0755			*		
0756	01F4	202A		LAC N11	* LEFT SHIFT OF 11 *
0757	01F5	0008		ADD CEXP	
0758	01F6	5008		SACL CEXP	
0759			*		
0760	01F7	280A		LAC CLO,11	
0761	01F8	500A		SACL CLO	
0762	01F9	581A		SACH CTEMP	
0763	01FA	201A		LAC CTEMP	
0764	01FB	794A		AND M07FF	
0765	01FC	0809		ADD CHI,11	
0766	01FD	5009		SACL CHI	
0767			*		
0768	01FE	F900		B OUTPUT	
	01FF	02FB			
0769			*		
0770	0200	202B	MSB14	LAC N12	* LEFT SHIFT OF 12 *
0771	0201	0008		ADD CEXP	
0772	0202	5008		SACL CEXP	
0773			*		
0774	0203	2C0A		LAC CLO,12	
0775	0204	500A		SACL CLO	
0776	0205	581A		SACH CTEMP	

```

0777 0206 201A      LAC CTEMP
0778 0207 794B      AND M0FFF
0779 0208 0C09      ADD CHI,12
0780 0209 5009      SACL CHI
0781                *
0782 020A F900      B OUTPUT
0783 020B 02FB      *
0784 020C 2019 C38B LAC C38
0785 020D 794F      AND M8000
0786 020E FF00      BZ MSB16
0787 020F 021C      *
0788 0210 202C      LAC N13      * LEFT SHIFT OF 13 *
0789 0211 0008      ADD CEXP
0790 0212 5008      SACL CEXP
0791                *
0792 0213 200A      LAC CLO,13
0793 0214 500A      SACL CLO
0794 0215 581A      SACH CTEMP
0795 0216 201A      LAC CTEMP
0796 0217 794C      AND M1FFF
0797 0218 0009      ADD CHI,13
0798 0219 5009      SACL CHI
0799                *
0800 021A F900      B OUTPUT
0801 021B 02FB      *
0802 021C 202D MSB16 LAC N14      * LEFT SHIFT OF 14 *
0803 021D 0008      ADD CEXP
0804 021E 5008      SACL CEXP
0805                *
0806 021F 2E0A      LAC CLO,14
0807 0220 500A      SACL CLO
0808 0221 581A      SACH CTEMP
0809 0222 201A      LAC CTEMP
0810 0223 794D      AND M3FFF
0811 0224 0E09      ADD CHI,14
0812 0225 5009      SACL CHI
0813                *
0814 0226 F900      B OUTPUT
0815 0227 02FB      *
0816 0228 280A CLOB LAC CLO,8      * CHI IS ZERO *
0817                *
0818 0229 580C      SACH C11      * SPLIT THE 16 BIT WORD INTO TWO
0819 022A 500D      SACL C12      8 BIT PIECES *
0820                *
0821 022B 2C0C      LAC C11,12
0822 022C FF00      BZ C12BP      * IF THERE IS A ONE ETC. *
0823 022D 0296      *
0824 022E 580E      SACH C21
0825 022F 500F      SACL C22
0826                *

```

0827	0230	2E0E		LAC C21,14	
0828	0231	FF00		BZ C22BP	* IF THERE IS A ONE ETC. *
	0232	0265			
0829			*		
0830	0233	5810		SACH C31	
0831	0234	5011		SACL C32	
0832			*		
0833	0235	2010		LAC C31	
0834	0236	FF00		BZ C32BP	* IF THERE IS A ONE ETC. *
	0237	024F			
0835			*		
0836	0238	7942		AND M0002	
0837	0239	FF00		BZ MSB18	
	023A	0246			
0838			*		
0839	023B	202E		LAC N15	* LEFT SHIFT OF 15 *
0840	023C	0008		ADD CEXP	
0841	023D	5008		SACL CEXP	
0842			*		
0843	023E	2F0A		LAC CLO,15	
0844	023F	5809		SACH CHI	
0845	0240	500A		SACL CLO	
0846	0241	2009		LAC CHI	
0847	0242	794E		AND M7FFF	
0848	0243	5009		SACL CHI	
0849			*		
0850	0244	F900		B OUTPUT	
	0245	02FB			
0851			*		
0852	0246	202F	MSB18	LAC N16	* LEFT SHIFT OF 16 *
0853	0247	0008		ADD CEXP	
0854	0248	5008		SACL CEXP	
0855			*		
0856	0249	200A		LAC CLO	
0857	024A	5009		SACL CHI	
0858	024B	7F89		ZAC	
0859	024C	500A		SACL CLO	
0860			*		
0861	024D	F900		B OUTPUT	
	024E	02FB			
0862			*		
0863	024F	2011	C32BP	LAC C32	
0864	0250	794F		AND M8000	
0865	0251	FF00		BZ MSB20	
	0252	025C			
0866			*		
0867	0253	2030		LAC N17	* LEFT SHIFT OF 17 *
0868	0254	0008		ADD CEXP	
0869	0255	5008		SACL CEXP	
0870			*		
0871	0256	210A		LAC CLO,1	
0872	0257	5009		SACL CHI	
0873	0258	7F89		ZAC	
0874	0259	500A		SACL CLO	
0875			*		
0876	025A	F900		B OUTPUT	


```

025B 02FB
0877
0878 025C 2031 MSB20 LAC N18 * LEFT SHIFT OF 18 *
0879 025D 0008 ADD CEXP
0880 025E 5008 SACL CEXP
0881
0882 025F 220A LAC CLO,2
0883 0260 5009 SACL CHI
0884 0261 7F89 ZAC
0885 0262 500A SACL CLO
0886
0887 0263 F900 B OUTPUT
0264 02FB
0888
0889 0265 220F C22BP LAC C22,2
0890 0266 5812 SACL C33
0891 0267 5013 SACL C34
0892
0893 0268 2012 LAC C33
0894 0269 FF00 BZ C34BP * BRANCH IF NO ONE *
026A 0280
0895
0896 026B 7942 AND M0002
0897 026C FF00 BZ MSB22
026D 0277
0898
0899 026E 2032 LAC N19 * LEFT SHIFT OF 19 *
0900 026F 0008 ADD CEXP
0901 0270 5008 SACL CEXP
0902
0903 0271 230A LAC CLO,3
0904 0272 5009 SACL CHI
0905 0273 7F89 ZAC
0906 0274 500A SACL CLO
0907
0908 0275 F900 B OUTPUT
0276 02FB
0909
0910 0277 2033 MSB22 LAC N20 * LEFT SHIFT OF 20 *
0911 0278 0008 ADD CEXP
0912 0279 5008 SACL CEXP
0913
0914 027A 240A LAC CLO,4
0915 027B 5009 SACL CHI
0916 027C 7F89 ZAC
0917 027D 500A SACL CLO
0918
0919 027E F900 B OUTPUT
027F 02FB
0920
0921 0280 2013 C34BP LAC C34
0922 0281 794F AND M8000
0923 0282 FF00 BZ MSB24
0283 028D
0924
0925 0284 2034 LAC N21 * LEFT SHIFT OF 21 *

```

```

0926 0285 0008      ADD CEXP
0927 0286 5008      SACL CEXP
0928                *
0929 0287 250A      LAC CLO,5
0930 0288 5009      SACL CHI
0931 0289 7F89      ZAC
0932 028A 500A      SACL CLO
0933                *
0934 028B F900      B OUTPUT
      028C 02FB
0935                *
0936 028D 2035 MSB24 LAC N22      * LEFT SHIFT OF 22 *
0937 028E 0008      ADD CEXP
0938 028F 5008      SACL CEXP
0939                *
0940 0290 260A      LAC CLO,6
0941 0291 5009      SACL CHI
0942 0292 7F89      ZAC
0943 0293 500A      SACL CLO
0944                *
0945 0294 F900      B OUTPUT      * ^^ COMPLETES TOP 8 ^^ *
      0295 02FB
0946                *
0947 0296 240D C12BP LAC C12,4
0948 0297 5814      SACH C23
0949 0298 5015      SACL C24
0950                *
0951 0299 2E14      LAC C23,14
0952 029A FF00      BZ C24BP
      029B 02CC
0953 029C 5816      SACH C35
0954 029D 5017      SACL C36
0955                *
0956 029E 2016      LAC C35
0957 029F FF00      BZ C36BP
      02A0 02B6
0958                *
0959 02A1 7942      AND M0002
0960 02A2 FF00      BZ MSB26
      02A3 02AD
0961                *
0962 02A4 2036      LAC N23      * LEFT SHIFT OF 23 *
0963 02A5 0008      ADD CEXP
0964 02A6 5008      SACL CEXP
0965                *
0966 02A7 270A      LAC CLO,7
0967 02A8 5009      SACL CHI
0968 02A9 7F89      ZAC
0969 02AA 500A      SACL CLO
0970                *
0971 02AB F900      B OUTPUT
      02AC 02FB
0972                *
0973 02AD 2037 MSB26 LAC N24      * LEFT SHIFT OF 24 *
0974 02AE 0008      ADD CEXP
0975 02AF 5008      SACL CEXP

```

```

0976      *
0977 02B0 280A      LAC CLO,8
0978 02B1 5009      SACL CHI
0979 02B2 7F89      ZAC
0980 02B3 500A      SACL CLO
0981      *
0982 02B4 F900      B OUTPUT
      02B5 02FB
0983      *
0984 02B6 2017 C36BP LAC C36
0985 02B7 794F      AND M8000
0986 02B8 FF00      BZ MSB28
      02B9 02C3
0987      *
0988 02BA 2038      LAC N25      * LEFT SHIFT OF 25 *
0989 02BB 0008      ADD CEXP
0990 02BC 5008      SACL CEXP
0991 02BD 290A      LAC CLO,9
0992 02BE 5009      SACL CHI
0993 02BF 7F89      ZAC
0994 02C0 500A      SACL CLO
0995      *
0996 02C1 F900      B OUTPUT
      02C2 02FB
0997      *
0998 02C3 2039 MSB28 LAC N26      * LEFT SHIFT OF 26 *
0999 02C4 0008      ADD CEXP
1000 02C5 5008      SACL CEXP
1001      *
1002 02C6 2A0A      LAC CLO,10
1003 02C7 5009      SACL CHI
1004 02C8 7F89      ZAC
1005 02C9 500A      SACL CLO
1006      *
1007 02CA F900      B OUTPUT
      02CB 02FB
1008      *
1009 02CC 2215 C24BP LAC C24,2
1010 02CD 5818      SACL C37
1011 02CE 5019      SACL C38
1012      *
1013 02CF 2018      LAC C37
1014 02D0 FF00      BZ C38BP
      02D1 02E7
1015      *
1016 02D2 7942      AND M0002
1017 02D3 FF00      BZ MSB30
      02D4 02DE
1018      *
1019 02D5 203A      LAC N27      * LEFT SHIFT OF 27 *
1020 02D6 0008      ADD CEXP
1021 02D7 5008      SACL CEXP
1022      *
1023 02D8 2B0A      LAC CLO,11
1024 02D9 5009      SACL CHI
1025 02DA 7F89      ZAC

```

```

1026 02DB 500A      *      SACL CLO
1027
1028 02DC F900      *      B OUTPUT
      02DD 02FB
1029
1030 02DE 203B MSB30  LAC N28      * LEFT SHIFT OF 28 *
1031 02DF 0008      ADD CEXP
1032 02E0 5008      SACL CEXP
1033
1034 02E1 2C0A      LAC CLO,12
1035 02E2 5009      SACL CHI
1036 02E3 7F89      ZAC
1037 02E4 500A      SACL CLO
1038
1039 02E5 F900      *      B OUTPUT
      02E6 02FB
1040
1041 02E7 2019 C38BP  LAC C38
1042 02E8 794F      AND M8000
1043 02E9 FF00      BZ MSB32
      02EA 02F4
1044
1045 02EB 203C      *      LAC N29      * LEFT SHIFT OF 29 *
1046 02EC 0008      ADD CEXP
1047 02ED 5008      SACL CEXP
1048
1049 02EE 2D0A      *      LAC CLO,13
1050 02EF 5009      SACL CHI
1051 02F0 7F89      ZAC
1052 02F1 500A      SACL CLO
1053
1054 02F2 F900      *      B OUTPUT
      02F3 02FB
1055
1056 02F4 203D MSB32  LAC N30      * LEFT SHIFT OF 30 *
1057 02F5 0008      ADD CEXP      * SORRY, BUT THATS ALL THE ACC CAN HOLD
1058 02F6 5008      SACL CEXP
1059
1060 02F7 2E0A      *      LAC CLO,14
1061 02F8 5009      SACL CHI
1062 02F9 7F89      ZAC
1063 02FA 500A      SACL CLO
1064
1065
1066
1067 02FB 490B OUTPUT OUT CSIGN,PA1
1068 02FC 4908      OUT CEXP,PA1
1069 02FD 4909      OUT CHI,PA1
1070 02FE 490A      OUT CLO,PA1
1071
1072 02FF F900 SELF  B SELF
      0300 02FF
1073
1074
NO ERRORS, NO WARNINGS

```