SOLIDAC PROGRAMMING MANUAL

ELECTRICAL ENGINEERING DEPARTMENT
GLASGOW UNIVERSITY

12TH NOVEMBER 1965.

G.H.R.

1. INTRODUCTION

It is assumed that the reader has some knowledge of the representation and manipulation of binary numbers. The following comments apply to conventions which have been adopted for use in SOLIDAC.

Negative Binary Numbers

SOLIDAC uses negative binary numbers in complement form. This is the binary equivalent of the "9's complement" form used to represent negative numbers on most desk calculating machines.

Example: To obtain − 5 in binary in a 6 binary digit register we subtract +5 from 0 as follows:

$$\begin{array}{rl} 0 & = 0\,0\,0\,0\,0\,0 \\ +5 & = 0\,0\,0\,1\,0\,1 \\ \hline -5 & = 1\,1\,1\,0\,1\,1 \end{array}$$

We might be led to suppose that this result is not − 5 but + 59 by adding up the appropriate powers of 2, but this ambiguity is overcome if we make the rule that the leftmost digit of the register is an indication of sign, being 1 for negative numbers and 0 for positive numbers. In the 6 digit register this implies that the largest positive number that the register can hold is

$$0\,1\,1\,1\,1\,1 = 31$$

SOLIDAC has a word length of 20 binary digits and hence the largest positive integer we can hold in a register is

$$2^{19} - 1 = 524287$$

The lower bound for negative numbers is

$$-2^{19} = 524288.$$

and this is represented by a 1 in the sign digit position (the leftmost digit position) and zeros everywhere else.

The contents of a register are often thought of as representing fractional numbers with the binary point between the 20th (sign) digit and the 19th digit. In this case we can represent numbers x in the range:

$$-1 \leq x \leq 1 - \frac{1}{2}^{19}$$

Evaluating Negative Binary Numbers

This is conveniently done by considering the digits to the right of the sign digit as a positive number and evaluating it by adding up the appropriate powers of two. The value of the sign digit is then added to this.

//

The value of the sign digit is 0 for all positive number, -2^{19} for negative integers and -1 for negative fractions.

Examples

For a 6 binary digit register (in which the sign digit for negative integers is -2^5) we would have:-

+ Integer	0 1 1 1 0 1	= 0 + 29 = 29
- Integer	1 1 1 0 1 1	= -2^5 + 27 = -32 + 27 = -5
+ Fraction	0. 1 0 1 0 1	= 0 + 0.65625 = 0.65625
- Fraction	1. 1 1 0 0 0	= -1 + o.75 = -0.25

Overflow

In the course of a calculation it is possible for a number to be generated which cannot be represented by the 20 binary digits available in SOLIDAC's registers. Such a condition is called overflow.

Three registers in SOLIDAC have a 21st digit position and because of this there are two types of overflow.

1) Strong Overflow. This implies that information has been lost, i.e. either a number in a 20 bit register has become too large or a number in a 21 bit register has become too large to be represented even by 21 bits. The computer regards this type of overflow as a catastrophy and will not continue beyond the point at which overflow of this nature is encountered.

2) Weak Overflow. This can only occur in a 21 bit register. In this case the contents of the register has exceeded 20 bit capacity but because of the extra digit no information has been lost.

Considering the contents of a 21 digit register as representing a number to 19 binary places, weak overflow occurs when numbers are encountered in the range,

$$-2 \leqslant x \leqslant -1 - (\tfrac{1}{2})^{19} \text{ or } 1 \leqslant x \leqslant 2 - (\tfrac{1}{2})^{19}$$

Just as the 20th bit in a register is known as the sign bit, the 21st bit in those registers where it exists, is known as the overflow bit.

All in range numbers in 21 bit registers are such that the sign bit = the overflow bit.

2. Description of SOLIDAC

2.1 Store

Solidac has a storage capacity of 1024 "words" each word consisting of a 20 bit pattern. This store is composed of a three dimensional/

array of magnetic cores. Each core is used to represent a 0 or a 1 and the cores are arranged in groups of 20. Each group represents a word and is uniquely determined by an integer in the range.

$$0 \leqslant N \leqslant 1023$$

This integer is known as the address of a word. This store is used to hold instructions for the computer and data for the instructions to operate upon.

A second store consists of 96 words of fixed instructions. These form a program which is used to read information from punched paper tape and to convert this information to binary.

2.2 ARITHMETIC UNIT

This is composed of 10 interconnected registers.

1) The Accumulator. This is a 40 bit register which may be considered as a 39 bit register with an overflow bit, or as two separate registers. The most significant half of the accumulator consists of the topmost 20 bits and the overflow bit and is called the M-Register. The bottom 19 bits form the L-Register.

2) The D-Register. This is a 20 bit register with an overflow bit.

2) The B-Registers or Modifier Registers. There are seven B-Registers and each is uniquely associated with integer in the range

$$1 \leqslant B \leqslant 7.$$

4) The Store Register - This is a 20 digit register with an overflow bit which is used to buffer information between the store and the rest of the computer. All information passing between the store and any other part of the machine must pass thro' the store register.

2.3 INSTRUCTION FORMAL AND OPERATION MODE

An instruction consists of a pattern of 20 bits which is subdivided as follows:

F	B	N
(6 bits)	(3 bits)	(11 bits)

F is an integer in the range	$0 \leqslant F \leqslant 63$
B is an " " " "	$0 \leqslant B \leqslant 7$
N is an " " " "	$0 \leqslant N \leqslant 2047$

Outside the computer, when punching instructions on paper tape the instructions are written in decimal as

F. B. N.

/

If B = 0 the instruction may be written as

F, n or

F, 0, n or

F, n,

The most significant 6 bits define an integer F which corresponds to some function the computer is to perform e.g. "add", "divide" "fetch from store" etc.

The next 3 digits define an integer B which, if it is non zero specifies one of the 7 B-registers. A value of B=0 implies that no B register is specified. The action of the computer when a B-register is specified is described under 3.2.

The bottom 11 digits of an instruction may be considered in two ways.

1) as a positive integer in which event

$$0 \leqslant n \leqslant 2047$$

2) as a signed integer with the 11th digit as the sign digit. In this case

$$-1024 \leqslant n \leqslant 1023.$$

As in many digital computers, instructions are interpreted and obeyed by a train of voltage pulses which are used to open and close electronic "gates" and which shift the contents of the registers along the many paths available inside the computer. This train of pulses forms a cycle which is performed everytime an instruction is obeyed. This cycle is in two parts.

1) The Instruction Word cycle or I.W. cycle. This performs the task of modification described under 3.2. and prepares the instruction for decoding in the following cycle.

2) The operation word cycle or O.W. cycle. This causes the instruction to be interpreted and obeyed.

Each time an instruction is obeyed an 11 bit counter in the control unit of the computer is incremented by 1, and provides the address from which the next instruction is to come. From this it will be seen that instructions are obeyed one at a time and that the instruction in the store location with address n will be followed by the instruction in the store location with address n + 1. Jump instructions are an exception to this sequential rule as will be seen later.

3. Order Code and Description of Orders.

3.1 Notation

An instruction will be considered as follows

Digit position :	20,19 ------ 15	14,13,12	11,10 ------ 2,1
Contents :	$0\leqslant F\leqslant 63$	$0\leqslant B\leqslant 7$	$0\leqslant N\leqslant 2047$

The following table describes the symbols used in the description of the instructions.

SYMBOL	DESCRIPTION:
B	This refers to digit positions 12, 13 and 14 of an instruction, whose contents specify an integer in the range: $0\leqslant B\leqslant 7$. if B 0 then this integer is the address of a modifier register (B-register). Modifier registers can hold 11 binary digits.
N	This refers to the contents of the bottom 11 digits of an instruction. In some instructions this integer is considered to be positive, in the range $0\leqslant N\leqslant 2047$, and in others the 11th digit is used as a sign digit, in which case the range of the integer N becomes $-1024\leqslant N\leqslant 1023$.
S	This is the address of a register in the core store and is an integer in the range $0\leqslant S\leqslant 2047$. At present there are only 1024 store selections and they have addressed in the range $0\leqslant S\leqslant 1023$. The integer S is held in the bottom 11 digits of the instruction.
C	This refers to the control counter, which is an 11 digit register whose contents are the address from which the next instruction is taken. After normal instructions 1 is added to this register automatically and the result provides the next instruction address. Jump instructions from an exception to this.
F	Is an integer contained in the top 6 digits of an instruction. This integer specifies the action which the instruction is to perform.
A	This refers to the double length accumulator register which has 39 digit positions and an overflow digit.
M	This refers to the most significant half of the accumulator which is a 20 digit register with an overflow digit.
L	This refers to the least significant half of the accumulator which is a 19 digit register.
D	This refers to a 20 digit register which has an overflow digit.

SL This refers to the signal lights register which is connected to a row of 20 lights on the console of the computer.

HS This refers to a 20 digit register whose contents are set by 20 switches on the console of the computer.
A 1 is set into this register if the switch in the corresponding digit position is down, otherwise 0 is set.

1 This refers to the input buffer register. This is a 5 digit register between the paper tape reader and the core store.

0 This refers to the output buffer register. This is a 5 digit register between the core store and the paper tape punch.

Other Conventions.

The contents of a register will be denoted by the letter C followed by a symbol in brackets.

Example :-

The contents of the store location with address S = C(S).

The contents of a register before and after an instruction is obeyed will be distinguished by putting a superscript after the last bracket in the latter case.

Example :-

The final contents of modifier register B = C(B)'.

Subscripts will be used to denote the digital positions of a register affected by an instruction. If all the positions are effected the subscripts will be omitted.

Example :-

The final contents of digit positions 1 to 11 of the D-register = $C(D)'_{1-11}$

3.2 Modification by B-Registers.

This only applies to those instructions which have a value of F in the range $16 \leq F \leq 63$.

If such an instruction specifies a value of B in the range 1 B 7 (i.e. B ≠ 0), then before the instruction is obeyed it will have the contents of the B register which is specified added to it's bottom 11 digits. The resulting instruction is then obeyed. In the process of addition of the modifier to the instruction any carry which might occur beyond the 11th digit of the instruction is inhibited

3.3 Examples of Instructions

1) 17.5.301

This has F = 17, b = 5 and S = 301

If B5 (i.e. modifier register with address 5) contained the integer 81 when this instruction was selected from the store, then the instruction which would be obeyed would be

17.5.(301 + 81) i.e. 17.5.382

2) 8.4.63

In this case F = 8, B = 4, S = 63.
As F is less then 16 no modification would take place and the instruction would be obeyed as it stands.

3.4 Order Code.

3.4.1 Stops There are three instructions which can be used to stop the computer when it is obeying a program. All use the value 0 for F but they differ in the values used in the B and N parts of the instruction.

1) Absolute stop. This has B and N both zero, i.e. it is the instruction,

0 . 0. 0

It is not possible to continue beyond an absolute stop

2) Normal stop. This has B = 0 but N ≠ 0. It is possible to continue beyond a normal stop by pushing a button on the console of the computer. (The "GO" button)

3) Optional stop. This has B ≠ 0 and N ≠ 0. This will only cause a stop if a switch on the console of the computer is in the position marked "optional stop". The program may be continued by restarting the computer manually as in 2).

3.4.2. B Register operations

It will be remembered that instructions with values of F in the range 0 F 15 are not modifiable. Instructions with F in the range 1 F 15 perform operations upon the contents of the B register specified in the instruction.

INSTRUCTION	OPERATION	COMMENT
1. B. S.	$C(S)'_{1-11} = C(B)$	Only the bottom 11 digits of S are altered.
2. B. S.	$C(B)' = C(S)_{1-11}$	
3. B. S.	$C(B)' = C(B) + C(S)_{1-11}$	
4. B. S.	$C(B)' = C(B) - C(S)_{1-11}$	
5. B. N.	$C(B)' = N$	Puts the integer N into B-Register.
6. B. N.	$C(B)' = C(B) + N$	Adds N to B Register
7. B. N.	$C(B)' = C(B) - N$	Subtracts N from B-register
8. B. N.	$C(B)' = C(B) \quad C(S)_{1-11}$	Replaces B by the logical product of B and S.
9. B. S.	$C(B)' = C(3)_{1-11}$ $C(S)'_{1-11} = C(B)$	exchanges contents of B and bottom 11 digits of S

3.4.3 Input Instruction

INSTRUCTION	OPERATION	COMMENT
10. B. S.	$C(B)'_{1-5} = C(S)'_{1-5} = I$	reads a character from the paper tape reader to B and S and clears the remainder of B and S.

3.4.4. B Register Conditional Jumps

The control counter C is used to provide the address of the next instruction to be obeyed. Normally this counter is increased by one at the end of each instruction so that the instructions are obeyed sequentially. In jump instructions it is possible for this process to be omitted and instead the contents of the control counter may be replaced by the integer specified in the N-part of the jump instruction.

/

This integer therefore specified the address of the next instruction.

Jumps are normally conditional i.e. if a specified condition holds then the jump is performed, otherwise the next instruction is selected in sequence as usual.

INSTRUCTION	OPERATION	COMMENT
12. B. N.	C(C)' = N if C(B) 0	Jump if contents of 11th digit of B is 0.
13. B. N.	C(C)' = N if C(B) 0	Jump of contents of B register are non zero.
14. B. N.	C(C)' = N if C(B)' = C(B) - 1 0	subtract 1 from the contents of the B register. If the result is non zero then jump.
15. B. N.	C(C)' = N if C(B)' = C(B) - 2 0	subtract 2 from the contents of the B register. If the result is non zero then jump.

Example

A frequent use of B register jump instructions is in the execution of loops in a program. Suppose it is required to obey a block of instructions a given number of times. This can be done conveniently using a B register to count the number of times the instructions in the block are performed. The program might then look as follows

location	contents	comment
100	5.3.138	Set C (B3) = 138
101	- - - -	
"	"	
"	"	
"	"	Block of instructions to
"	"	be obeyed.
"	"	
190	- - - -	
191	7. 3. 1.	Subtract 1 from C(B3)
192	13. 3.101	Jump to 101 if C(B3) 0.

This program could be reduced by one instruction by replacing the instructions in locations 191 and 192 by the following instruction in location 191

14. 3. 101

The subtracts one from B3 each time it is obeyed, and returns to 101 until such time as B3 becomes zero.

/

3.4.5 Special Modify Instructions

We have seen that instructions with F in the range 0 F 15 are not modifiable by B-registers because they use the B-registers for other operations. The special modify instructions permit the modification of all instructions and extend the range of modifiers from the 11 digits of a B-register to 20 digits.

INSTRUCTION		COMMENT
16.	N	Add the integer N to the next instruction. Inhibit any carry occurrring beyond 11th digit during the addition.
17.	S	Add C(S) to next instruction

Examples: Set C(B4) = C(B7)

The following program will do this

LOCATION	CONTENTS	COMMENT
n	16 . 7 . 0	Add N = C(B7) to next order
n+1	5. 4 . 0	Set C(B4) = N = C(B7)

Two types of modification can be included in one instruction as follows

17 . 5 . 13

This would cause C(13 + C(B5)) to be added to the next order.

3.4.6. Output Instruction

INSTRUCTION		OPERATION	COMMENT
20.	S	$C(O)' = C(S)_{1-5}$	S is unaltered. $C(S)_{1-5}$ are punched on paper tape.

3.4.7. Modifiable Jump Instructions

INSTRUCTION		OPERATION	COMMENT
21.	N	C(C)' = N if C(A) 0	Jump to N if C(A) are negative.
22.	N	C(C)' = N if C(A) 0	Jump to N if C(A) are positive or zero.
23.	N	C(C)' = N if C(A) 0	Jump to N if C(A) are non zero.
24.	N	C(C)' = N if overshaft during normalise	See normalise order (3.4.10)
25.	N	C(C)' = N if weak overflow in A or D	Jump if weak overflow exists in A or D registers
26.	N	C(C)' = N	unconditioned jump
27.	N	C(C)' = N and change store.	" " with switch between normal & fixed stor

This last instruction requires some further explanation.

The 96 words of fixed instructions mentioned in 2.1 are called the fixed order (F.O) store. The first 96 locations of store are in fact repeated, one set being the F.O. store the other set being the bottom part of the normal order (N.O.) store. This duplication only affects the programmer when fetching information from locations with addresses in the range 0 to 95. These addresses can be assigned to either the F.O. store or the first 96 locations of the N.O. Store. An assignment existing at any time is reversed by executing a 27 order.

3.4.8 Peripheral Switches

Solidac has been provided with a facility for handling several input/output devices. Each device is assigned to a "channel" which may be disconnected or connected to the computer by means of the following two instructions. Input/output channels are referenced by means of integers in the bottom part of the instructions. Channels connected to input devices are referred to by the odd integers 1, 3, 5, 7 etc. and output channels are given the even integers 0, 2, 4 etc. At present there is a 5 hole paper tape reader on channel 1 and a 5 hole paper tape punch on channel 0, the other channels being empty. The following instructions will provide a method of selecting a peripheral device should more than one of the same type exist in the future.

INSTRUCTION		OPERATION	COMMENTS
28.	N	Connect I/O unit N	Connecting one input disconnects all others
29.	N	Disconnect Output unit N.	

It will be noted that only one input unit may be connected at any one time while any number of output units may be connected at the same time. Instructions using F = 10 or F = 20 affect only those units which are connected. When the computer is initially started input unit 1 and output unit 0 are automatically connected for use by the F.O. store instructions.

3.4.9. L-Register Instructions

/

3.4.9 L - Register Instructions

The L-register is the bottom 19 stages of the accumulator. It is often useful to hold real numbers to 19 binary places in the accumulator, so that the L-register contains the fractional part and the other stages of the accumulator (i.e. the M-register) contain the integer part. For purely fractional numbers the M-register will be all 1's (negative fraction) or all zeros (positive fraction). When adding or subtracting real numbers to 19 binary places to the accumulator it will be necessary to add or subtract -1 to or from the M-register of the number is negative. The following instructions are useful in this respect.

Instruction	Operation	Comment
31. S	$C(S)'_{1-19} = C(L)$	Store C(L) in S put sign of C(S) = 0
32. S	$C(L)' = C(S)_{1-19}$ $C(M)' = C(S)_{20}$	Fetch bottom 19 digits of C(S) to L, put sign of C(S) in M
33. S	$C(L)' = C(L) + C(S)_{1-19}$ $C(M)' = C(M) + C(S)_{20}$	Add bottom 19 digits of C(S) to L, add sign of C(S) to M
34. S	$C(L)' = C(L) - C(S)_{1-19}$ $C(M)' = C(M) - C(S)_{20}$	Subtract bottom 19 digits of C(S) from L, subtract sign of C(S) from M

3.4.10 Normalise.

It is often useful to use numbers in floating point form. Such a number is in the form

$$a \times 2^b$$

where a is called the mantissa and is a fractional number and b is an integer called the exponent. Such a number is said to be normalised if a is in the range

$$-1 \leqslant a < -\tfrac{1}{2} \quad \text{for negative a}$$
$$\text{or } \tfrac{1}{2} \leqslant a < 1 \quad \text{for positive a}$$

It will often happen that after a series of arithmetic operations the result will be in non standard floating point form with the mantissa in the accumulator.

/...

It is then possible to obtain the result in normalised form by transferring the exponent to the D-register and executing a normalise instruction. This causes the contents of the accumulator to be shifted until it is in standard floating point form rounded off to 19 binary places in M. The appropriate modifications are made to the exponent in the D-register. It may be that the programmer has some idea of the number of accurate places in the result of a series of arithmetic operations. He will then be in a position to specify an integer N such that if it takes more than N left shifts to obtain the result rounded off to 19 binary places in M and in normalised form, then the result will not be significant. It is therefore possible to specify an integer N in a normalise instruction and if after N left shifts the result is not normalised a condition called overshift is set by the computer. This and other normalise facilities can be obtained by appropriately specifying N as follows

i) $0 < N \leqslant 40$ This range of N will be used when the programmer wishes to be informed of overshift. N is the smallest impermissible number of shifts.

ii) $N \leqslant 0$ This will be used when the programmer does not want the condition of overshift to be set irrespective of the contents of the accumulator.

iii) $N > 40$ Overshift will only be set in the event that the accumulator is zero.

If overshift is set by a normalise order it can only be unset by the execution of an instruction with F=24 (see 3.4.7). An attempt to obey any other instruction in the presence of overshift will cause a failure.

Normalise instructions will give a correct result when the contents of the accumulator indicate weak overflow before the instruction is obeyed.

The integer N is specified by the bottom 11 digits in the normalise instruction, which is in the form

35. N

<u>3.4.11. Shift Instructions.</u>

Four instructions exist for shifting the contents of the accumulator register. The instructions using F = 36 and F = 37 are called the arithmetic shift instructions. The contents of the accumulator are considered to be

/.....

representing a number. When shifting to the right these instructions cause the sign digit to be repeated and are equivalent to a division by 2 for each shift right. Similarly arithmetic shifts to the left are equivalent to multiplications by 2 and so there is a possibility of overflow occurring. Instructions using F = 38 and F = 39 are called logical shift instructions. In this case the contents of the accumulator are considered to be representing a pattern of 39 binary digits. Shifting to the right introduces the appropriate number of zeros at the top end while shifting left can not set overflow, the digits simply being pushed off the top end of the register. After a logical shift instruction the computer arranges the overflow digit of the accumulator to be equal to the 39th digit thus ensuring that overflow can not be set by the instructions following logical shifting.

Shift instructions specify an integer N in the bottom 11 digit positions which is the number of shifts to be given.

Instruction	Operation	Comment.
36. N	$C(A)' = C(A) \times 2^N$	Arithmetic left shift may cause overflow.
37. N	$C(A)' = C(A) \div 2^N$	Arithmetic right shift may cause overflow
38. N	Pattern in A shifted N places left - logical left shift.	
39. N	Pattern in A shifted N places right - logical right shift.	

NOTE: Negative values of N are quite legitimate and result in shifts in the opposite direction to that specified by the function number. Values of N greater than 40 or less that - 40 are taken to be exactly 40 or - 40 respectively.

/....

3.4.12 M - Register Operations.

The following ten instructions are those normally required for performing fixed point single length arithmetic operations;

Instruction.	Operation	Comment;
40. S	C(S)' = C(M) C(A)' = 0	Store C(M) in S and clear the accumulator (i.e. M and L).
41. S	C(S)' = C(M)	Store C(M) in S
42. S	C(M)' = C(S)	Fetch C(S) to M - Register
43. S	C(M)' = C(M) +C(S)	Add C(S) to C(M), result in M
44. S	C(M)' = C(M)-C(S)	Subtract C(S) from C(M) result in M
45. S	C(S)' = C(S) + C(M)	Add C(M) to C(S), result in S
46. S	C(S)' = C(S) - C(M)	Subtract C(M) from C(S) result in S
47. S	C(M)' = C(M) ≢ C(S)	Logical operation "not equivalent"
48. S	C(M)' = C(M) ⊕ C(S)	Logical operation "and"
49. S	C(M)' = C(S), C(S)' = C(M)	Exchange C(M) and C(S)

The operations "not equivalent" and "and" are described by the following

if a = 1100
and b = 1010
then a ≢ b = 0110
and a ⊕ b = 1000

3.4. 13 D - Register Operations.

The D - register is used in multiplication, division and normalise instructions and consequently it is often necessary to perform manipulations on its contents. The following instructions exist for this purpose.

Instruction	Operation	Comment.
51. S	C(S)' = D	Store C(D) in S
52. S	C(D)' = S	Fetch C(S) to D
53. S	C(D)'=C(D)+C(S)	Add C(S) to C(D) result in D
54. S	C(C)' = C(D) - C(S)	Subtract C(S) from C(D) result in D
55. N	$C(D)'_{1-11} = N$ $C(D)'_{12-20} = 0$	Puts N in bottom of 11 digits of D, clears rest of D.

.../....

3.4.14 Multiplication

In multiplication, two single length numbers (i.e. 19 digits + sign digit) are multiplied together to give a double length product (i.e. 38 digits + sign digit). The multiplier is held in the D-register, the multiplicand in a store location and the product is put into the double length accumulator. The number of binary places in the product is the sum of the binary places in the two original numbers. The D-register and store register contents are unaltered after multiplication. If both original numbers are negative and of maximum size (e.g. -1 to 19 b.p.) weak overflow will occur in the product. The multiplication instruction is

56. S — multiply the contents of S by the contents of D and put the result in A. S and D are unaltered. Weak overflow is set if contents of S and D are maximum size and negative.

3.4.15 Division

It is often difficult for the beginner to understand the process of fixed point division as performed by a computer and so an example of decimal division in a 4 stage decimal register computer is given. It is important to remember that during fixed point arithmetic operations the computer has no knowledge of the position of the binary point. The computer simply produces the digits of the result and the programmer must arrange his calculations so that he knows at each stage where the binary point is.

Consider a dividend given to double length significance in a 4 digit register decimal computer,

dividend = 04700000

and the divisor given as a single length number

divisor = 0120

The/

The division process produces the digits of the results as

quotient = 3916

and

remainder = 0080.

This works perfectly well until we encounter the following situation

dividend = 4700000

divisor = 0120

In this case the first "digit" of the result is greater than 9. In other words the quotient is out of range.

In general, it is necessary to scale the operands of a division by shifting them relative to each other so that the result is certain to be in range.

Returning to the binary number system of solidac we are given the dividend in the accumulator and the divisor in a store location. If we disregard the signs of the two operands and consider their absolute magnitudes only, in order that the result of a division will be in range, the first digit of the quotient must be zero. In other words the absolute magnitude of the most significant half of the dividend must be less than the absolute magnitude of the divisor. To arrange this it may be necessary to shift the store register to the left and/or the accumulator to the right. If after scaling, the dividend is given to p binary places and the divisor to q binary places the quotient will have p-q binary places and the remainder in the M-register will have p binary places.

The division instruction is

58.	S	Divide the contents of the double length accumulator by the contents of S and put the result in D and the remainder in M. Overflow will be set if the quotient is out of range.

3.4.15/

3.4.15 M and D Interchange

An instruction which is often of use in conjunction with multiply and divide instructions is as follows

Instruction		Operation	Comment
59.	0	D' = M, M' D	Interchange M and D.

3.4.16 Hand Switch and Signal Lights Instructions

These instructions provide a means of communication between the operator and the computer while the computer is running.

Instruction		Operation	Comment
60.	S	C(SL)' = C(S)	Put C(S) on signal lights.
61.	S	C(S)' = C(HS)	Put C(HS) into S.

3.417 Unassigned Function Numbers

Function numbers, 11, 18, 30, 50, 57, 62 and 63 have the same effect as an absolute stop instruction.