4. STORE ACCESS CONTROL

4.1 Need for Store Access Control

In the type of storage system used in LEO III there is only one access point to the store. Thus only one word can be copied from or into the store at a time.

However, there may be times when more than one of the units of the computer may require access to the store, and some control system must be provided for choosing which shall be given access first.

4.2 Units which require Access

The units which may require access to the store are as follows:

Assemblers for peripheral equipment

Clocks

Arithmetic unit and coordinator

Store monitor display

These units can be divided into two classes: those which can wait for an indefinite period without an error occurring due to the delay; and those which can only wait for a limited period without an error occurring.

A typical unit of the latter class is a punched card reader. It cannot stop reading until the end of a card is reached, and so there is no pause between the reading of any two columns. However, the assembler buffer register holds only five characters at a time, and so these must be transferred to the store within the time taken to read the next character and place it in the buffer; otherwise information already in the buffer will be corrupted.

4.3 Relative Access Requirements of the Different Units

4.3.1 Units which cannot wait indefinitely for access

The units which cannot wait indefinitely for access to the store without information being corrupted are:

Unit		Maximum waiting time
(i)	Magnetic tape deck	105 μsecs.
(ii)	Anelex printer (when linked to the standard Anelex assembler or general purpose output assembler)	150 μsecs (average)
(iii)	Card reader	1.4 m.secs.
(iv)	Card punch	4.0 m.secs.

Amendment No. 8
December 1964

4.3.2 Units which can wait indefinitely for access

These units are, in order of frequency of calling access during the reading or writing process:

Unit	Frequency of calling access
(i) Millisecond timer	1 m.sec.
(ii) Anelex printer (when linked to the special Anelex printer assembler)	1.65 m.secs.
(iii) Paper tape reader	5 m.secs.
(iv) Paper tape punch	110 m.secs.
(v) Digital clock	6 seconds
(vi) Main frame	-

4.4 Store Access Priority

4.4.1 LEO III System

Each assembler, clock, arithmetic unit, etc. is connected to store access control by an information pathway known as a channel. There are 10 channels, of which 8 are available for peripheral equipment. Whenever access is requested by any channel, a 'store access requested' condition is set up for that channel.

As soon as the store becomes free (or immediately, it it is not busy), the 'access requested' conditions of all channels are examined simultaneously and, of the channels requesting access, that which has the highest priority is then given access. The 'access requested' condition for that channel is then reset. The priority given to each channel is determined by its channel number, lowest numbers having highest priority.

Channel numbers are fixed during the construction of the machine, and the different assemblers are assigned to channels 0 to 7 according to the priority they require. This is dependent on the types of equipment linked to them, as follows:

Highest Priority

Magnetic tape assemblers (maximum of 4 to 1 store access control unit)

General purpose output assembler with Anelex printer - or standard Anelex assembler

Card/tape input assembler with card readers

General purpose output assembler without Anelex printer

Special Anelex assembler

Card/tape input assembler without card readers

Paper tape output assembler

Millisecond timer

Lowest Priority

Channel 8 is reserved for the store display on the engineers' control desk and for manual stacking of numbers in store, and channel 9 for the main frame.

4.4.2 LEO 326 System

Peripheral Priority Control (PPC)

There are 10 channels connected to the PPC: 0 to 7 for assemblers, channel 8 for the store monitor, and a special 'Read only' channel (channel 9) which is used when the coordinator wishes to engage a peripheral device. The PPC accepts Call Access signals from these ten channels, deciding between competing calls on the basis of permanently assigned channel-priorities. (Channel 0 has the highest priority and channel 9 the lowest). The PPC then transfers the call to the CPC.

Highest Priority

Magnetic tape assemblers

General purpose assembler with Anelex printer

Card/tape input assembler with card readers

General purpose output assembler without Anelex printer

Special Anelex assembler

Card/tape input assembler without card readers

Paper tape output assembler

100-microsecond timer

Lowest Priority

Store monitor

Channel 9 - a Read only channel

Calculator Priority Control (CPC)

The CPC controls the transfer of information along three channels: the Special Peripheral channel, and those formed by the PPC and the calculator (which comprises the coordinator and the arithmetic unit); these are designated channels 0 to 2. The CPC decides between competing calls on the basis of fixed channel-priorities in descending order from channel 0 to channel 2.

The two stores, which are designated channels 3 and 4, are completely autonomous and associated with separate priority controls. It is thus possible for both stores to be connected to channels at the same instant. But no two channels can ever be simultaneously connected to one store; nor may one channel be simultaneously connected to both stores.

Store Priority Control

Since there are three input channels and two store channels, there are six possible paths. The three paths associated with the main store are under the control of the Main Store Priority Control and the remaining three are under the control of the Second Store Priority Control. Under the action of a store priority control a path is set up between the particular store and the channel that has gained access. When such a path is set up, all address data, information data, and control data is made available to the store concerned.

Division Nomination

The Division Nomination is a minor unit within the CPC which interprets the division data for each channel and passes on the request for access to the appropriate store. A separate priority control is associated with each store. Acting on the information obtained from the division nomination, each priority control will allow only the input/output channels that require access to the specific store under its own control to compete for access. Thus independent action and control is secured for each store.

4.5 Parity Checks (with LEO 326 read 'CPC' for 'SACU' and 'calculator' for 'main frame')

4.5.1 When information is placed in the store:

Every word received by the SACU from an assembler is accompanied by a parity bit, which is checked by the SACU. The register holding the word in the assembler is not cleared until a signal is received showing that the parity check at the SACU has been satisfied. Parity failure stops the assembler, which must then be reset by an engineer. This corrective action sets up in the assembler a condition known as 'doubtful block', indicating to the programme that the last block transferred is suspect. If parity is found on checking to be correct, the word is placed in the store with correct parity for each short word. Words received from the main frame are not checked, but are placed in the store with correct short-word parity.

4.5.2 When information is taken from the store:

Each word is checked for correct parity in both its constituent short words. To prevent information being corrupted, parity failure immediately stops the store, magnetic tape equipment and the equipment that called access (without waiting for the current instruction to be completed), and other equipment stops as soon as it next calls access. The address of the offending compartment is displayed on the engineers' control panel.

If parity is found to be correct, the word is sent, according to the instruction, to:

(a) The main frame - without parity bit.

(b) An assembler (PPC on LEO 326) - parity bit in each word, checked on arrival)

(c) Store monitor - with parity bits in each short word.

4.6 Capacity of Store Access Control Unit

A store access unit can handle:

(i) A single store of up to 4 divisions

(ii) 8 input/output channels

(iii) One main frame and one store display/stack in store channel

If more than this maximum capacity is required, or if more concurrent magnetic tapes are to be operated than one store access control can deal with (see section 4.4), a second store access control can be added. This must have some store associated with it. Only the main frame and store display/stack in store channels can have access to both stores. Any assembler is restricted to using the store attached to its own store access control.

5. COORDINATOR AND INSTRUCTIONS

5.1 Introduction

The coordinator controls the sequence of selection and manner of execution of instructions. The method by which this is done is described in this chapter, and the effect of individual instructions in sections 6, 7, 15 and 16.

5.2 Registers of the Coordinator

The coordinator has two control registers in which each bit is held on a flip-flop. They are:

(i) The Instruction Register - 23 bits

This register is used to hold the instruction currently being obeyed. It can be considered for some purposes as two registers - the action part (more significant 8 bits) and the address part (less significant 15 bits), referred to in this manual as OI and OA respectively.

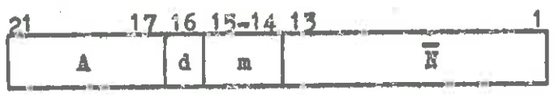
(ii) The Sequence Control Register - 15 bits

This register normally holds the location of the next instruction to be obeyed and is used for selecting that instruction. It is referred to in this manual as SC.

The purpose and operation of these registers is described in detail later in the section.

5.3 Form of Instruction

Each instruction is held in a short compartment of the store as a 21-bit pattern arranged as follows:



where:

A : Action Number (5 bits)

d : Discriminant (1 bit)

m : Modification Bits (2 bits)

$\overline{N}$: Address (13 bits)

Instructions are of the same form whether in LEO III or LEO 326. The two machines employ action codes which are virtually identical; differences chiefly concern side-effects of certain arithmetic instructions, and are noted in the specifications of the actions affected (sections 6, 7 and 16).

5.3.1 Action Number

A 5-bit binary number (in the range 0 to 31) which identifies the function of the action.

5.3.2 Address

A 13-bit binary number (in the range 0 to 8191) which usually gives the address (within a division) of the compartment of store referred to by the instruction. The reference is always to a compartment in the same division as the instruction (except in special cases - see section 5.5): to ensure this, the division number is automatically copied from the sequence control register into bits 14 and 15 of the address part of the instruction register at the same time as the instruction address N is copied into bits 1 to 13 thus giving a 15-bit address which is unique within a 4-division store: this 15-bit address is referred to as N (see section 5.6).

In some cases the address bits may be used to contain a literal value to be used in arithmetic. In other cases the address may be considered as three parts referred to as:

N_1 : bits 1 to 6

N_2 : bits 7 to 12

N_3 : bit 13

These uses of the address are described in the definition of the instructions which use them.

5.3.3 Discriminant and Modification Bits

These often have special uses as detailed in sections 5.4 and 5.5 but in some cases may be used to specify a variant of the action number (see Appendix B).

5.4 Discriminant

Most instructions involve transfer of information from the arithmetic registers (see section 6.2) to the store or vice versa. In order to specify if a short or long compartment is to be used a special bit (the discriminant) is provided.

5.4.1 The type of transfer of information depends on the discriminant and the address. In general if the discriminant is zero a short compartment is indicated, if the discriminant is 1 a long compartment is indicated. The type of transfer is as follows:

From arithmetic unit to store (after conversion from sign and complement form to sign and modulus form)

d = 0 Q1 to Q5 of the arithmetic unit register are copied into Q1 to Q5 of the short compartment. The sign bit of the arithmetic unit register is copied into S.

d = 1 (N even) Q1 to Q10 of the arithmetic unit register are copied into Q1 to Q10 of the long compartment. The sign bit of the arithmetic unit register is copied into S2. S1 is cleared.

From store to arithmetic unit (before conversion to sign and complement form)

d = 0 Q1 to Q5 and S of the short compartment are copied into Q1 to Q5 and the sign bit of the arithmetic unit register. Q6 to Q10 of the arithmetic unit register are cleared.

d = 1 (N even) Q1 to Q10 and S2 of the long compartment are copied into Q1 to Q10 and the sign bit of the arithmetic unit register. S1 of the long compartment is ignored.

5.4.2 In order to carry out arithmetic on numbers held in alpha form (see section 3.4) and also to store numbers in alpha form, a special type of transfer is provided where the address is <u>odd</u> and the discriminant is 1, as follows:

From arithmetic unit to store

Q1 to Q5 of the arithmetic unit register are copied into basic quartets Q1, Q3, Q5, Q7 and Q9 respectively of the long compartment specified by the even address (N-1). A 4 (0100) is inserted into the control quartet of any non-zero digit.

The sign bit of the arithmetic unit register is copied into the least significant bit of Q10 of the long compartment.

From store to arithmetic unit

Q1, Q3, Q5, Q7 and Q9 of the long compartment specified by the even address (N-1) are copied into Q1 to Q5 respectively of the arithmetic unit register. If the least significant two bits of Q10 of the long compartment hold the pattern '01' then the arithmetic unit register sign bit is set. Q6 to Q10 of the arithmetic unit register are cleared.

5.4.3 In those actions which allow no choice as to the type of information transfer (if any), the discriminant is often used to describe a variant of the action.

If in the action definitions in this manual an action is written as (for example):

Action 6 d m

the 'd' indicates that the discriminant is used to specify type of information transfer. If however the discriminant specifies a variant of the action, then each variant is stated separately, e.g.

Action 26 1 m

5.5 Modification

The address of an instruction can be modified by adding a 15-bit binary number to the address part of the instruction register after the instruction address has been placed there. This can be done in one of two ways, direct or indirect modification.

5.5.1 Direct Modification

Certain compartments in division 0 of the store are set aside as 'modification registers'. These are arranged in four groups numbered 0 to 3 as shown in Appendix E. In each group there are three such registers numbered 1 to 3.

The least significant 15 bits of any of these compartments may be added to the address in the instruction register. The 'group number' required is specified by a special register (which can be set by programme instruction), the 'modification register number' within the group is specified by the modification bits of the instruction.

In actions that are not directly modifiable and which do not 'set' or 'step on' modification registers (see Section 7.3) the modification digits may be used to specify a variant of the action.

If in the action definitions in this manual an action is written (for example):

Action 2 d m

the 'm' indicates that the modification digits refer to a modification register. If an action is written as (for example):

Action 0 d 2

then the action is not directly modifiable and the number in the modification bits is used to specify a variant of the action. In such cases each variant is specified separately.

5.5.2 Indirect Modification

The address part of all actions (including directly modifiable ones) can be modified indirectly by the least significant 15 bits of any compartment of the store.

This is done by inserting an extra instruction before the instruction to be modified. It is thus less efficient than direct modification. A full description of the system is given in section 8.4.

5.6 Carrying out an Instruction

The coordinator carries out instructions taken one at a time from the store. Unless a sequence change instruction is encountered, instructions are taken from consecutive compartments. At the beginning of every instruction, the instruction register holds the action, discriminant, and modifier bits of the instruction, and a 15-bit address composed of the instruction address N and the current division number in bits 14 and 15. The sequence control register contains the 15-bit address of the store compartment from which this instruction was taken.

The action bits of the instruction register are used to select a 'built in' programme for the action which is held in special core stores known as 'microplanes' in the coordinator. This 'micro-programme' performs a sequence of operations as follows:

(i) The sequence control register is stepped on by one - provided the repetition condition (see section 5.8) is not set - to specify the location of the next instruction . (This is later altered if there is to be a sequence change).

(ii) If the instruction being carried out is directly modifiable, the modification bits in the instruction register are examined and the address modified if necessary.

(iii) The action is executed in a series of basic steps.

(iv) The next instruction is placed in the instruction register as described below in section 5.7 (see also 5.8). Control is then given to this next instruction.

5.7 Selecting the next Instruction

At the end of each instruction the next instruction is placed in the instruction register by the microprogramme provided that the computer is not about to halt, and that external or internal interruption is not about to take place (see section 5.8 and 16), in which case the halt or interruption takes place instead.

The instruction is taken from the store compartment specified by the sequence control register. The address bits are copied into the bits 1 to 13 of the instruction register. Bits 14 and 15 of the sequence control register (the current division number) are copied into bits 14 and 15 of the instruction register. The action, discriminant and modification bits are copied into the action part of the instruction register.

Control is then automatically given to the microprogramme specified by the action - and sometimes discriminant and modification - bits.

5.8 Conditions Affecting the Execution of an Instruction

Unless otherwise stated these conditions are specified by flip-flops in the coordinator.

5.8.1 The 'Halt' Condition

If this condition is set, the computer halts immediately before selecting the next instruction. The condition is set by action 0.0.0 (see section 17.4) or by pressing the Halt Key on the operators' or engineers' control desk. The computer can be restarted by pressing the Restart Key on the control desk, and the next instruction is then selected.

5.8.2 The 'Repetition Condition

This is an engineers test condition. If it is set the stepping on of sequence control of the beginning of the instruction is automatically suppressed, so that at the end of the instruction the same instruction is selected again.

5.8.3 The 'Overflow' and 'Lockout' Conditions

The overflow and lockout conditions cause a break in the normal sequence of instructions.

Lockout is described in section 17 and overflow in section 6.6.

5.8.4 The 'External Interruption Requested' and the 'External Interruption Permitted' Conditions

The 'external interruption requested' condition is specified by a flip-flop in the coordinator.

The 'external interruption permitted' condition is specified by bit 13 of the Indicator register (see section 16.1). When this bit is set at 1, the 'external interruption permitted' condition is set and the computer is said to be in 'interruptible mode'. When the bit is set at zero, the computer is said to be in 'non-interruptible mode'.

If both these conditions are present, the normal sequence of instructions is broken as specified in section 16.

6. ARITHMETIC

6.1 Introduction

The arithmetic unit is controlled by the coordinator and is used for arithmetical and logical operations. It consists of:

(i) Registers (temporary storage devices) used to hold numbers;

(ii) An adder;

(iii) Circuitry to enable data to be moved between registers and the adder.

The arithmetic units of LEO III and LEO 326 differ to a sufficient extent to warrant separate descriptions in this volume, and these are given in the two sub-sections that follow.

6.2 The Arithmetic Unit - LEO III

6.2.1 Registers

The arithmetic unit contains the following registers:

Register A - 40 bits and sign
Register B - 40 bits and sign
Register C - 20 bits
Register D - 40 bits and sign
Register E - 40 bits and sign
Register F - 40 bits and sign
Register K - 10 bits
Register T - 4 bits

The sequence control register (SC) and the less significant half of the instruction register (OA), both of 15 bits, are used by some actions as arithmetic registers. The normal functions of these registers are described in section 5.

For the purpose of arithmetic operations all these registers except K and T may be considered as being divided into quartets of 4 bits, each quartet containing one digit of a number as described in section 3.4.

Registers A and B are the main registers with which the programmer is concerned. They may be used as separate registers, or as one register AB of 20 quartets and sign, in which case B is less significant than A and the sign bit of B is not used. These registers are known as the upper and lower accumulator respectively and are used for the formation of results.

Register C is also of importance to the programmer: it is used to hold 'excess constants', as described in section 6.4.

The remaining registers are used as working registers by most actions, and are not of great importance to the programmer. Their main uses are described briefly here:

Register D - Used to hold numbers received from the store

Register E - Used to hold numbers taken from the 'sum' output of the adder (see section 6.2.2).

Register F - Used to hold numbers to be copied to store

Register EF - E and F may be used as one register, in a similar manner to AB. Their main use then is in shifting numbers to left and right.

Register K - Used to note when carries between quartets have occurred in addition - see sections 6.2.2 and 6.3.1.

Register T - This register is used in connection with the store reservation system (see section 16) to hold the programme tag number.

6.2.2 The Adder

The adder is a device which adds two binary numbers, each of 40 bits and sign, producing a 'sum' output of 40 bits and sign and a 'carry' output. The 'carry' output records carries which have occurred between quartets, and is placed in register K.

6.2.3 Alignment of Numbers

For arithmetic purposes all numbers are aligned to the right (less significant) end of the register containing them. In the case of transfers when the destination and source are of different sizes, the following rules apply:

If destination is larger than source, the more significant quartets of the destination are set at zero;

If destination is shorter than source, the more significant quartets of the source are not copied to the destination;

The sign bit of the source is always copied to the sign bit of the destination, irrespective of differences in register lengths.

6.3 The Arithmetic Unit - LEO 326

6.3.1 Registers

The arithmetic unit contains the following registers:

Register A - 40 bits and sign
Register B - 40 bits and sign
Register C - 20 bits
Register D - 40 bits and sign
Register E - 40 bits and sign
Register F - 40 bits and sign
Register R - 10 bits
Register I - 13 bits
Register J - 4 bits
Register T - 4 bits
Register DIV- 3 bits
Register MG - 2 bits

The sequence control register and the less significant half of the instruction register are used by some actions as arithmetic registers.

Registers A and B are the main registers that concern the programmer. (see section 6.2.1 for further details).

Register C is used to hold 'excess constants', as described in section 6.4.

Registers D, E and F are full-length (40 bits and sign) registers variously used for holding data to be used in calculation, intermediate results, and final results. Registers E and F form a double-length register for use in shift operations.

Register K holds carries generated at the outputs of mill quartets (see section 6.3.2).

Register I is an indicator register holding 12 bits from the operators' control desk or engineers' control unit keys; or it may be used with register MG to hold the output from the mill.

The remaining registers are used as working registers by various actions.

6.3.2 The Mill

The mill is a 41-bit parallel adder/subtracter. Complements (see section 6.4.3) are formed in the mill itself, all numbers being held in the arithmetic registers in sign and modulus form.

6.3.3 The Exchange

The exchange receives data from the CPC or the mill output, and transfers it, realigned as necessary, to registers OA, B, D, SC, DIV, and C, in the arithmetic unit, and SR, IR and X in the coordinator.

6.3.4 Alignment of Numbers

Numbers are aligned in the same manner as for LEO III (see section 6.2.3).

6.4 System of Arithmetic

6.4.1 Excess Constants

The adder adds together two binary numbers. Carries between quartets will thus occur when the value of a quartet in the sum exceeds 15.

If numbers in other radices than binary, held as described in section 3.4, are to be operated upon, then the carry between quartets must be made to occur at some other value (e.g. for decimal - when any quartet in the sum exceeds 9).

In order to do this, a number x must be added in each quartet position, where x is the difference between 16 and the required carry value for that quartet position, e.g.

for decimal the numbers to be added in the ten quartet positions are:

6666666666

for sterling they are:

6666666 $\overline{14}$ 6 4

These numbers are called 'excess constants'.

The excess constants are held in register C and are set up by a computer code action. Only the least significant five quartets are held in the register.

When addition is performed the more significant excess constants are given the value of the most significant quartet in the register. This means that any mixed radices used (e.g. sterling) may only vary in the five least significant digit positions.

If binary numbers are being used the excess constants will all be zero, as binary numbers, if considered as being divided into quartets, are in effect in radix 16.

6.4.2 Example of use of Excess Constants

This example illustrates the effect of using excess constants in sterling arithmetic to add £117.11.8 to £8124.14.7.

Operation	Contents of Registers (Sign Bit)										
Initially the augend is in the accumulator:	+	0	0	0	8	1	2	4	1	4	7
First the excess constants are added	+	6	6	6	6	6	6	6	$\bar{14}$	6	4
giving:	+	6	6	6	$\bar{14}$	7	8	$\bar{10}$	$\bar{15}$	$\bar{10}$	$\bar{11}$
(Note that no carry has occurred between quartets and no carry can have occurred if the augend has been held in the specified radix)											
Next the addend is taken from store and added:	+	0	0	0	0	1	1	7	1	1	8
giving:	+	6	6	6	$\bar{14}$	8	$\bar{10}$	2 *	0 *	$\bar{12}$	3 *
(* denotes a carry from a quartet - these are recorded in register K)											
Now the correction procedure is effectively to <u>subtract</u> excess constants where no carry occurred by adding the complement form (see 6.4.3 and 6.4.4)	-	9	9	9	9	9	9	$\bar{15}$	$\bar{15}$	$\bar{10}$	0
	+	0	0	0	8	2	4	2	0	6	3

which is the correct answer : £8242.6.3

6.4.3 Negative Numbers and Subtraction

Negative numbers in the store are held in sign and modulus form as described in section 3.6.

Negative numbers in the arithmetic registers are held in sign and complement form, i.e. each quartet takes the value x, where x equals 15 minus (the corresponding quartet of the excess constants) minus (the corresponding quartet of the modulus), and the sign bit is set. 1 is then added to the result. This provides for the correct manipulation of negative numbers, which are converted to this form when placed in registers, and reconverted when returned to the store. To subtract A from B, (-A) is added to B.

6.4.4 Example of Subtraction using Excess Constants

This example illustrates the effect of using excess constants in decimal arithmetic to subtract 1234 from 18025.

Operation	Contents of Registers (Sign Bit)									
Initially the minuend is in the accumulator:	+	0	0	0	0	0	1	8	0	2	5
The subtrahend is then obtained from store, set negative and converted to sign and decimal complement form giving:	-	9	9	9	9	9	9	8	7	6	6
and the excess constants added:	+	6	6	6	6	6	6	6	6	6	6
giving:	-	$\bar{15}$	$\bar{15}$	$\bar{15}$	$\bar{15}$	$\bar{15}$	$\bar{15}$	$\bar{14}$	$\bar{13}$	$\bar{12}$	$\bar{12}$
(Note that these two steps are actually done at the same time by taking a 'reflection' of the positive subtrahend (i.e. interchanging 0's and 1's) and adding 1).											
The sign and complement form of the subtrahend is now added to the minuend giving:	+	0	0	0	0	0	1	6	$\bar{13}$	$\bar{15}$	1
and carries are recorded:		*	*	*	*	*	*	*			*
The excess constants are now subtracted where no carry occurred. This is done by adding the binary complement (reflection + 1):	-	$\bar{15}$	$\bar{15}$	$\bar{15}$	$\bar{15}$	$\bar{15}$	$\bar{15}$	$\bar{15}$	9	$\bar{10}$	0
giving:	+	0	0	0	0	0	1	6	7	9	1

which is the correct answer.

6.5 Overflow - LEO III

Overflow is an alarm condition which may be generated in connection with an arithmetic operation. The presence of the overflow condition is indicated by the setting of an overflow flip-flop in the coordinator. The overflow flip-flop is set when an arithmetic operation produces a carry into bit 41 (sign bit) of the register holding the result of the operation. This in itself is not always a fault, and in computer code actions which allow this possibility the flip-flop is reset before completion of the action.

When overflow is not expected (for the addition of two numbers of the same sign for instance) the flip-flop remains set at the end of the action and control is passed to the Master Programme.

6.6 Overflow - LEO 326

The general definition of overflow as given for LEO III also applies to LEO 326; but there are differences of detail in the conditions that may cause the alarm to be generated; these are noted in the Action Specifications where necessary.

Note that LEO 326 can generate overflow when working with numbers incompatible with the radix specified by C, whereas LEO III can not.

6.7 Arithmetic Actions

The computer code actions used to perform arithmetic according to the rules given in section 6.4 are defined in this section. Addition and subtraction of excess constants, and conversion to and from sign and complement form, are carried out automatically unless stated otherwise.

The notation previously described in sections 5 and 6.2 is used in all action descriptions. Note that brackets surrounding a reference to a register or store address are used to denote the contents of that register or compartment.

A brief description of the process involved accompanies the definitions of more complex actions.

6.7.1 Arithmetic on Variables

The following actions are used to perform arithmetic on numbers contained in compartments.

Action 2 d m : TRANSFER

Definition: Transfer (A) to N

Notes: (i) A is cleared

(ii) The radix set in C is immaterial for positive numbers.

Action 3 d m : COPY

Definition. Copy (A) to N

Notes: (i) (A) is unchanged

(ii) The radix set in C is immaterial for positive numbers.

Action 4 d m : ADD

Definition Add (N) to (A)

Action 5 d m : SUBTRACT

Definition: Subtract (N) from (A)

Action 6 d m : SELECT

Definition Select (N) into A

Notes: (i) The radix set in C is immaterial for positive numbers.

Action 7 d m : AUGMENT

Definition: Augment (N) by (A)

Notes:
(i) (A) is unchanged
(ii) Overflow occurs if the original contents of N are zero, and these of A minus zero.

Action 9 d m : MULTIPLY IN COMMON UNIFORM RADIX

Definition: Multiply (A) by (N) and place the result in AB.

Notes:
(i) (A) and (N) must both be in the uniform radix specified by (C). (A uniform radix is one in which the values of the excess constants are the same in each digit position).
(ii) The original contents of A may be up to nine digits.

Procedure: This action is carried out as follows:

1. A test is made to determine if the process described in 2 below has already been carried out for this multiplicand in this radix; this is done by testing to see if (A) = (34') and (C) = (158). If so 2 is omitted.
2. Multiples (A) x 1 to (A) x (R-1) are placed in long compartments 34' onwards in sign and complement form, where R is the radix. (C) is placed in 158 and copies of the most significant quartet of (C) are placed in five quartets of 159.
3. A is cleared.
4. The multiplier (N) is obtained and put in B, (in sign and modulus form if negative).
5. The multiple corresponding to the lowest digit of (B) is obtained and added into A, or subtracted from (A) if the multiplier is negative
6. The contents of AB are shifted right one quartet.

7. Steps 5 and 6 are then repeated 9 more times when all the digits of the multiplier will have disappeared from B and the lowest digit of the product will have moved to the lowest place in B, i.e. with the full product in AB.

Action 10 d m : MULTIPLY AND ADD

Definition: Multiply (N) by (B) and add the product to (A).

Notes:

(i) (B) must be a decimal number in sign and modulus form - action 0.d.2 may be used to achieve this.

(ii) (N) is in the radix set in C.

(iii) B is cleared.

(iv) Overflow occurs if:

(a) N contains minus zero, and B is negative.

(b) B has x significant digits, and 10^xx (N) > $2^{40}-1$.

(v) When the result of reading from the store is minus zero, it will be treated as plus zero by LEO 326, whereas on LEO III overflow may occur.

Procedure: The action is carried out as follows:

1. If (B) is positive (N) is placed in working registers D and F (in complement form if negative). If (B) is negative the sign bit of B is set at zero and -(N) is placed in working registers D and F (in complement form if negative).
2. If the lowest quartet Q1 of (B) is zero, steps 3 and 4 are omitted.
3. (F) is added to (D) till (D) = (Q1 of B) x (F)
4. (D) is added to (A)
5. (F) is added to (D) till (D) = 10(F).
6. (D) is copied to F
7. (B) is shifted one digit right, dropping the least significant digit.
8. A test is made on (B): if (B) is zero the process is complete, otherwise the above steps are repeated starting at 2.

Action 11 d m : MULTIPLY AND SUBTRACT

Definition: Multiply (N) by (B) and subtract the product from (A).

Notes:

(i) (B) must be a decimal number in sign and modulus form - action 0.d.2 may be used to achieve this.

(ii) (N) is in the radix set in C.

(iii) B is cleared.

(iv) Overflow occurs if B has x significant digits and 10^{x} (N) > $2^{40}-1$.

(v) When the result of reading from the store is minus zero, it will be treated as plus zero by LEO 326, whereas on LEO III overflow may occur.

Procedure: Action is carried out as follows:

1. If (B) is positive -(N) is placed in working registers D and F (in complement form if negative). If (B) is negative, the sign bit of B is set at zero and (N) is placed in working registers D and F (in complement form if negative).

2. to 8. As for action 10.d.m.

Action 0 d 2 : REPLACE (B) BY (N)

Definition: Place (N) in B in sign and modulus form.

Action 13 d m : DIVIDE

Definition: Divide the contents of the more significant 19 quartets of AB by (N), placing the quotient in A and the remainder in the more significant 9 quartets of B.

Notes:

(i) (AB) and (N) must both be in the uniform radix specified by C.

(ii) The least significant quartet of (B) is unchanged if (AB) was initially positive, otherwise the least significant quartet of (B) at the end of the action is the complement of its original value.

(iii) Overflow occurs if the divisor is zero, minus zero or greater than nine digits; or if AB contains minus zero originally; or if the quotient exceeds ten digits.

Procedure: The Division process is carried out by successive subtraction of the divisor from that part of the dividend which is in A, moving the dividend one digit position left after each digit of the quotient has been obtained. To get the quotient in the right digit positions the dividend must first be shifted in AB so that its least significant digit is in the second lowest digit position of B. Since a maximum of 10 digits is permitted in a quotient, the number of digits in the dividend must not exceed that in the divisor by more than 10, or, if the digits of the dividend in A form a number greater than the divisor, by not more than 9.

This is carried out as follows:

1. The divisor (N) is placed in working register D (in sign and complement form if (N) is negative).
2. If (AB) is negative it is replaced by -(AB) and the sign of D is changed.
3. The modulus of (D) is then subtracted from (A) repeatedly until (A) becomes negative, taking say, x times. (x must be less than the radix otherwise the overflow alarm is set).
4. The modulus of (D) is added to (A). (The top digit of (A) must now be zero, otherwise the overflow alarm is set).
5. (AB) is shifted one digit left.
6. x - 1 is placed in the least significant digit of B.
7. Steps 3 to 6 are repeated nine more times.

8. (A) and (B) are interchanged. Thus the quotient is in A so that it is readily available for further arithmetic and the modulus of the remainder is now in the top nine digits of B.

9. (A) is replaced by -(A) if (D) is negative.

6.7.2 Arithmetic on Literals

The following actions are used to perform arithmetic on literals. The actual address part of the instruction, and not the contents of a specified compartment, is the number operated upon in each case. Since the address of an instruction is 13 bits only, literals are limited to three quartets and a fourth (most significant) digit which may take only the values 0 or 1.

Literals are assumed by these actions to be positive.

Action 1.1.1 : ADD LITERAL

Definition: Add $\overline{N}$ to (A)

Notes: (i) The store division bits from SC are not added to N.

Action 1.1.2 : SUBTRACT LITERAL

Definition: Subtract $\overline{N}$ from (A)

Note: (i) The store division bits from SC are not added to N.

Action 1.1.3 : SELECT LITERAL

Definition: Select $\overline{N}$ into A

Notes: (i) The store division bits from SC are not added to $\overline{N}$.

(ii) The radix set in C is immaterial.

6.7.3 Rounding and Shifting

The following actions give facilities for rounding, shifting and scaling numbers in registers needed in conjunction with multiplication and division.

Action 1.0.2 : ROUND OFF

Definition: Add 1 to (A) if the contents of the most significant quartet of B are greater than or equal to N (literal).

Notes:

(i) The store division bits from SC are not added to N.

(ii) B is cleared.

(iii) (A) must not exceed $R^{10}-2$, where R is the radix.

Action 18.0.0 : SHIFT (A) LOGICALLY

Action 18.0.1 : SHIFT (A) ARITHMETICALLY

Action 18.1.0 : SHIFT (AB) LOGICALLY

Action 18.1.1 : SHIFT (AB) ARITHMETICALLY

Definition: Shift (A) or (AB) right or left the number of quartet positions specified by N.

Notes:

(i) Left shift is indicated by N3 = 0. Bits 1 to 5 of N specify in binary the number of quartet positions to be shifted. Zeros are inserted into the quartets left vacant by the shift. There is no difference between arithmetic and logical shift.

(ii) Right shift is indicated by N3 = 1. The binary complement of bits 1 to 5 of N specifies the number of quartet positions to be shifted. In logical shifts, zeros are inserted into the quartets left vacant by the shift. In the arithmetic shift a value x is inserted into each quartet left vacant by the shift if (A) is negative, where x = (15 minus the most significant quartet of (C)), otherwise zeros are inserted.

(iii) The sign bits of A and B are not shifted.

(iv) The maximum number of quartet positions shifted is 31.

Action 18.0.2 : SCALE NUMERATOR

Definition: Shift (AB) until the digit corresponding to the most significant non-zero digit of the sign and modulus form of (AB) is in the second most significant quartet of A. Subtract 1 from (N) for each quartet left shift, or add 1 to (N) for each quartet right shift, which occurs.

Notes:

(i) Addition and subtraction are in binary and act only on bits 1 to 20 of (N).

(ii) N is treated as a positive number and the sign bit, if set, is cleared.

(iii) If (AB) is zero it is shifted 20 quartets but N is not changed.

(iv) (AB) must not consist of a sign bit only.

Action 18.1.2 : SCALE DENOMINATOR

Definition: Shift (AB) until the digit corresponding to the most significant non-zero digit of the sign and modulus form of (AB) is in the second most significant quartet of A. Add 1 to (N) for each quartet left shift, or subtract 1 from (N) for each quartet right shift, that occurs.

Notes: As for action 18.0.2

Action 18.0.3 : BINARY LEFT SHIFT (A)

Action 18.1.3 : BINARY LEFT SHIFT (AB)

Definition: Shift (A) or (AB) left the number of bit positions specified by $\overline{N}$.

Notes:

(i) Bits 1 to 12 of $\overline{N}$ specify in binary the number of bit positions to be shifted. Zeros are inserted into the bits left vacant by the shift.

(ii) The sign bits of A and B are not shifted.

(iii) The maximum number of bits shifted is 4095.

(iv) (C) is immaterial.

(v) Bit 13 of $\overline{N}$ does not affect the action in LEO 326, but must be zero in LEO III.

6.7.4 Collation

Collation is a means of extracting information from specified parts of a compartment (which may hold several items of information).

The parts of the compartment to be operated upon are specified by register B. If any bits (including the sign bit) of (B) have the value 1 the corresponding bits of the compartment are operated upon, while bits of the compartment corresponding to zero bits of B are not used or changed.

Collation is defined as: compare two bit patterns. For any bit position which has the value 1 in both patterns set the corresponding bit = 1 in the result; set all other bits = 0 in the result.

Action 1.0.1 : PREPARE FOR DIGIT COLLATION

Definition: For each of the least significant ten bits of $\overline{N}$ (literal) which is 1, set the corresponding quartet of B as '1111' and for each which is zero set the corresponding quartet of B as '0000'. Copy bit 11 of N to the sign bit of B.

Notes:

(i) This action can only be used to prepare for collation involving an integral number of quartets. If other patterns are required in B, action 0.d.2 (see section 6.7.1) must be used.

Action 14.d.B : REPLACE SELECTED BITS

Definition: Replace those bits of (N) specified by (B) with the corresponding bits of the sign and modulus form of (A).

Notes:

(i) (A) and (B) are unchanged.

(ii) The radix set in C is immaterial if (A) is positive.

(iii) If discriminant 1 with address odd (see section 5.4.2) is specified, (N) is first converted to numeric form, the operation is then performed on the numeric form of (N) and the result is then reconverted back to alpha form before placing in the store. Only the least significant 5 quartets and the sign bit of B are relevant.

(iv) Minus zero in A is treated as zero.

Procedure: The action is carried out as follows:

1. (A) is placed in working register F in sign and complement form, and in working register D in sign and modulus form.
2. (D) and (B) are collated and the result placed in A.
3. (N) is obtained and placed in working register D.
4. (D) is added to (A) in binary and placed in working register E.
5. (D) and (E) are collated and the result placed in A.
6. (A) is subtracted from (E) in binary and the result stored in compartment N. (F) is restored to A.

Action 15.d.m : COLLATE AND ADD

Definition: Collate (N) with (B) and add the result to (A).

Notes:

(i) (N) and (B) are unchanged.

(ii) Arithmetic is in the radix set in C. The result of collation will be treated as negative and converted to sign and complement form for addition only if the sign bits of both N and B are set at 1.

(iii) If discriminant ' with address odd (see section 5.4.2) is specified the 'convert from alpha to numeric' operation takes place before the collation and thus collation is performed on the numeric form of (N). Only the least significant 5 quartets and the sign bit of B are relevant.

(iv) If the result of the collation is minus zero, it is treated as zero.

(v) Overflow occurs if the original contents of A are minus zero and the result of the collation is negative.

Procedure: The action is carried out as follows:

1. (A) is placed in working register F.
2. (N) is obtained and placed in working register D.
3. (D) and (N) are collated and the result placed in A.
4. (A) and (F) are added according to the radix set in C, and the result placed in A.

6.7.5 Radix Facilities

The following actions allow the radix for arithmetic operations to be set in register C, and allow conversion of numbers from one radix to another.

Action 0.0.3 : SET RADIX

Definition: Copy (N) to C

Note: The five quartets of (N) are copied to register C. The sign bit of N is ignored.

Action 12.d.m : CONVERT

Definition: Convert (N) to the radix specified by (C) and place the result in A. The relationship between (C) and the radix of (N) is defined by the table starting at (A).

Notes:

(i) Before this action occurs, the equivalent in the new radix of unity in each quartet position in the old radix must be held as a table of constants in ten successive long compartments. The address of the first of these compartments (containing the equivalent for the least significant digit position) must be placed in A before the action.

(ii) Register B is cleared.

(iii) The conversion of an n digit number ($n < 10$) involves the reference of $(n+1)$ long compartments of store in the conversion table. To avoid lockout the $(n+1)$th compartment must bear the same tag as the table.

(iv) In the particular case where $n = 10$, only 10 long compartments are referred to and note (iii) does not then apply.

Procedure: The action is carried out as follows:

1. (N) is placed in B in sign and modulus form.
2. x, the address of the first equivalent in A, is placed in OI.
3. A is cleared.
4. (x) is then added to (A) as many times as specified by the least significant digit of (B).
5. (B) is shifted right one digit.
6. x, held in the instruction address register, is increased by 2, (to get the address of the next equivalent).
7. A test is made on (B): if (B) is zero step 8 is carried out; otherwise the above steps are repeated starting at 4.
8. If (N) is negative (A) is replaced by -(A). (This is determined by examining the sign bit of B, which is then cleared).