F E R R A N T I   L T D

FERRANTI ATLAS COMPUTER

EXTRACODE FUNCTIONS

Computer Department

London Computer Centre:	68 Newman Street, London, W.1. Telephone: Museum 5040.
	21 Portland Place, London, W.1. Telephone: Langham 9211.
Offices, Laboratories & Works:	West Gorton, Manchester, 12. Telephone: East 1301.
Research Laboratories:	Western Road, Bracknell, Berks.

List CS 309A
December 1962

ATLAS EXTRACODE FUNCTIONS

1. INTRODUCTION

This document describes the way in which the instruction code on Atlas has been extended by the technique of extracode instructions, and gives descriptions of these extracodes.

2. NOTATION

Lower case letters are used for suffixes and for the contents of a location The result of an operation is denoted by a prime. Thus, s is the contents of address S, and $s' = s + am$ means the contents of S after the operation is equal to the contents of S before plus the contents of Am.

(i) Suffixes

x	the fractional part of the suffixed location
y	the exponent of the suffixed location
:	the consecutive pair of locations starting with that suffixed
*	the register following that suffixed

(ii) General

A	the full double length floating point accumulator, of 79 bit fractional part Ax and 8 bit exponent Ay
A1	the 48 bit floating point register consisting of L, Ls and Ay
Am	the 48 bit floating point register consisting of M and Ay
AO	the accumulator overflow register
B	a general B-register
Ba	the B-register specified by the Ba digits of an instruction
Bm	the B-register specified by the Bm digits of an instruction
Bc	the B-carry
Bt	the B-test register
C	the main control register B127
C( )	the contents of the bracketed location
E	the extracode control register B126
EO	the exponent overflow register
F	the function digits
G	the logical accumulator, consisting of B98 and B99
H	the modified address part of an instruction regarded as a half word address
K	the least significant octal fraction of an address
L	the less significant half of Ax; 39 bits with no sign
Ls	the sign bit associated with L
M	the most significant half of Ax; sign bit and 39 bits
M/E	a control flip-flop between main and extracode control
N	the 24 bit address digits of an instruction
n	the modified address part of an instruction regarded as a 21 bit integer, with a fractional part consisting of one octal digit
P	the block address part of an address, i.e. digits 1 - 11 of S
Q	thw word address within a block, i.e. digits 12-20 of S
S	the modified address part of an instruction regarded as a full word address
Vr	line r of the V-store, where r is an integer
X	signifies those accumulator extracodes suitable for fixed point working

The 24 digits of a half word are numbered from 0 to 23, digit 0 being the most significant, digit 1 the next, etc. In an instruction, the function digits (0-9) are referred to as f_0 - f_9.

3. ATLAS INSTRUCTIONS

An instruction on Atlas occupies a full 48 bit register. It consists of a function F, two B-registers Ba and Bm, and an address N.

```
F(10 bits)   Ba(7 bits)   Bm(7 bits)              N(24 bits)
..........    .......      .......     ........................
0   -   9    10  -  16    17  -  23    0              -              23
```

The function consists of a most significant binary bit f_0, followed by three octal numbers f_1-f_3, f_4-f_6, f_7-f_9. Ba and Bm specify 24 bit B-registers in the range 0 - 127. N is usually regarded as a 21 bit integer with a least significant octal fraction.

Atlas instructions are of two kinds, basic or extracode.

3.1 Basic instructions

Basic instructions are the operations which the computer has been designed to perform directly. They are recognised by f_0 being zero, and are of two types.

(i) A-type are accumulator instructions. N is doubly modified to give a full word address $S = N + ba + bm$.

(ii) B-type are B-register instructions. Ba is used as an operand; the other operand is obtained by singly modifying N. This operand may be used directly as a number $n = N + bm$ or as the address $H = N + bm$ of a half word h. In some B-type instructions, such as test instructions, Bm is used as an operand, and so N is not modified.

The basic instructions are described in CS 345.

3.2 Extracode instructions

Extracode instructions cause more complicated operations to be performed which the machine has not been designed to execute directly. They cause automatic entry to basic instruction subroutines in the fixed store.

Extracode instructions are recognised by $f_0 = 1$, and the other function digits determine the particular subroutine entry address required. Normally the subroutines end by exiting automatically to the instruction following the extracode instruction which caused entry to them. Thus extracode instructions can be treated for most purposes as if they were basic instructions.

In detail, the following action takes place when an extracode instruction is encountered.

(i) Main control B127 is advanced by one to the address of the next program instruction.

(ii) N is modified according to type (N + ba + bm for A-type, N + bm for B-type) and the result placed in B119.

(iii) The seven Ba digits are placed in bits 15-21 of B121 so the extracode can refer to Ba by using B122. In the special case of the extracode instruction specifying Ba as B122, B121 is left unaltered.

(iv) The function digits $f_1 - f_9$ are placed in extracode control B126 as shown below.

Bit	012	3 - 8	9 10 11	12 13 14	15 16 17	18 19 20	21 - 23
Value	100	0	0 f_1 f_2	f_3 0 0	f_4 f_5 f_6	f_7 f_8 f_9	0

(v) Control is switched to extracode control.

The next instruction to be obeyed is thus in the fixed store at an address determined by the function digits, and under extracode control.

It is in one of 64 registers (given by $f_4 - f_9$) in one of 8 tables at intervals of 256 words (given by $f_1 - f_3$). These tables are called "jump tables". This instruction will be, in general, an unconditional jump into a routine in the fixed store which will perform the required operation. These routines, which are written in basic instructions, are called extracodes. They end with an instruction in which $f_1 = f_3 = 1$, (e.g. functions 521 or 720) which is obeyed as if $f_1 = 0$ (i.e. as 121 or 320) after which control is re-switched to main control. The next instruction to be obeyed is then given by main control and, except for extracodes which may cause conditional jumps, is that immediately following the extracode instruction.

4. ALLOCATION OF FUNCTIONS

There are 512 function numbers available for extracodes, 1000 - 1777. Of these 1000 - 1477 are singly modified instructions (B-type) and 1500 - 1777 are doubly modified (A-type).

The extracodes are divided into sections as shown below:

1000 - 1077	Magnetic tape, input and output routines
1100 - 1177	Organisation routines
1200 - 1277	Test instruction and 6-bit character operations
1300 - 1377	B-register operations
1400 - 1477	Complex arithmetic, vector arithmetic and miscellaneous B-type accumulator routines
1500 - 1577	Double length arithmetic and accumulator operations using the address as an operand.
1600 - 1677	Logical accumulator operations and half word packing
1700 - 1777	Arithmetic functions (log., exp., sq.rt., sin, cos, tan, etc.) and similar miscellaneous A-type accumulator operations.

5. UNALLOCATED FUNCTIONS

Not all of the extracode functions have been allocated, and, where convenient, extracode programs and constants have been packed into the unallocated jump table locations. This means that use of a non-existent extracode may result in an unassigned function interrupt or may cause some extracode to be entered wrongly. The latter case would give the programmer wrong results.

The extracode 1000 always causes an unassigned function interrupt. It can be entered by "obeying" floating point zero as an instruction, which is equivalent to 1000 0 0 0. The relevant entry in the jump table, i.e. the first location in the fixed store which causes this interrupt, is the floating point number $\frac{1}{2}$.

6. EXTRACODE SPECIFICATIONS

The extracode function is listed at the left of the page and followed by a short description. The number of basic instructions obeyed is given at the right of the page. This number includes the extracode instruction and its entry in the jump table; where necessary a range or formula is given.

In the arithmetic extracodes, where possible, the last two octal function digits correspond to those of similar basic instructions. Accumulator operations are rounded floating point unless marked X when they are suitable for fixed point working.

6.1 Magnetic Tape Routines 1001 - 1047

The octal fraction of the address, K, is used in many instructions as a count. $0 \leq K \leq 7$. In general, for any K, K+1 blocks are involved in the transfer. This allows transfers of from 1 to 8 blocks.

6.1.1 Block Transfers.

1001 Search for section n on tape Ba

1002 Read next K+1 sections from tape Ba into store blocks P, P+1, •••, P+K

1003 Read previous K+1 sections from tape Ba into store blocks
P+K,••••, P+1, P

1004 Write store blocks P, P+1,•••, P+K on to the next K+1 sections on tape Ba.

1005 Move tape Ba forward K+1 sections

1006 Move tape Ba backwards K+1 sections

6.1.2 Organisation.

1007 Mount next reel of file Ba and allocate number n to it.

1010 Mount
Allocate number Ba to tape with title in locations starting at S. If tape is not already available instruct operator to mount it.

1011 Mount free
Select a free tape (instruct operator to mount one if necessary), write the title from location S onwards on to section 0 of this tape and allocate it as tape Ba.

1012 Mount on logical channel K.

1013 Mount free on logical channel K.

1012 and 1013 are similar to 1010 and 1011 respectively, but K specifies a logical channel number in the range 0 to 7. If this channel has not been previously defined (by use of 1012 or 1013), where ever possible, the tape is mounted on a channel different to any which have been defined. This channel is then designated as program channel K. If K has been previously defined, where ever possible, the tape, is mounted on this channel. Thus these extracodes allow the programmer to mount up to 8 tapes on different channels.

1014 Write title
Write on section 0 of tape Ba the title stored from S.

1015 Read title
Read the title of tape Ba from section 0 to locations from S.

1016 Unload, preserve for later use.
Rewind tape Ba and disengage. Instruct operator to remove, check title on reel and store.

1017 Free tape, not required again
Erase title on tape Ba, return tape to Supervisor for general use.

1020 Release tape, pass it to another program
Delete tape Ba from program allocation and make it available for another program.

If n $\neq$ 0 in 1016 - 1020, the number of tape mechanisms reserved for the program is reduced by one.

1021 Release mechanisms
Reduce by n the number of tape mechanisms reserved for use by the program.

1022 Re-allocate
Allocate the number n to the program magnetic tape previously referred to as Ba

1023 How long?
h$'$ = number of 512 word sections available on tape Ba, excluding section 0

1024 Where am I?
(i) After block transfer orders:
s$'$ = address of next section on tape Ba, going forwards.
This is in the full word position of the first half word; the second half word is cleared.
(ii) After variable length transfers: (see under)

6.1.3 Variable Length Instructions. Variable length working must always be initiated by a start instruction. The information is stored in strings of words with 24 bit markers at the ends of each string. These markers give the number of words in the string and a separation level marker is a number in the range 1-7 which is set by the programmer to indicate different levels of units of the information, e.g. level 1 for ordinary strings, level 2 for a group of records, level 3 for complete files, etc.

Each writing transfer writes one string. A reading transfer may read a specified number of words or up to the end of a string (the markers are not read). In this section K is the least significant octal fraction of n unless otherwise stated.

1030 Start reading forwards
Select tape Ba to be read forwards starting at the next word on the tape. Therafter ensure the buffer is kept replenished. The buffer is in blocks P, P+1,...,P+K.

1031 Start reading backwards
Select tape Ba, starting at previous word.
Then as 1030

1032 Start writing forwards
Select tape Ba to be written forwards from the next word on the tape. Up to K+1 buffer blocks are used as required. A marker Q($0 \leq Q \leq 7$) is written before the first word of information. The buffer is in blocks P, P+1,...., P+K

1033 Select
Select tape Ba for succeeding variable length operations in the mode previously specified for that tape

1034 Start reading forwards from fixed blocks

1035 Start reading backwards from fixed blocks

1036 Ba$'$ = selected magnetic tape

1037 $h' =$ mode of magnetic tape Ba
$h' = 0$ for variable length read forward transfer using strings
$h' = 1$ for variable length read backwards transfer using strings
$h' = 2$ for variable length write transfers using strings
$h' = 3$ for fixed block transfer
$h' = 4$ for variable length read forwards transfers from fixed blocks
$h' = 5$ for variable length read backwards transfers from fixed blocks

1040 Transfer the ba words between store addresses starting at S and the selected tape in the appropriate selected mode. On writing, the octal marker K at the foot of ba is written after the last word.
On reading, the transfer continues until ba words have been read or until a marker $m \geq K$ is encountered, whichever is sooner. Then, $ba' =$ no. of words actually read, with $K' = 0$ if no marker $\geq K$ was met, or $K' = m$ if a marker $m \geq K$ terminated the transfer.

Notes:

(i) when reading backwards the words read occupy store locations S+ba-1, S+ba-2 ...

(ii) if $ba = 0$, the transfer continues until the first marker is encountered

1041 Skip
Skip ba words, or skip until a marker $m \geq K$ is encountered, whichever is sooner. Skip operates as transfer except that no words are transferred. (1041 is much less efficient than 1044 (search) for positioning the tape.)

1042 Mark
Marker $K(0 \leq K \leq 7)$ is written after the last word on the selected tape. 1042 is ignored if in reading mode.

1043 Stop
Stop variable length operations on tape Ba. This releases associated buffer blocks, and after writing operations causes the last part section to be written immediately. (This could also be done by start, search, unload, release tape, or end program).

1024 Where am I?
(i) After variable length transfer
$s' =$ halfwords A and W, defined below:
Less significant half of $s' =$ address (W) within the section of the current marker on tape Ba (or if not on a marker, of the next forward word)
More significant half of $s' =$ address (A) of the section containing the address described above.

(ii) For after block transfers see under Organisation (6.1.2.)

1044 Word search
Search for word W, section A of tape B where S contains A and W as defined in 1024

1046 Read Orion tape forwards
Read the next block on Orion tape Ba into store blocks P, P+1,... P+K.
A check is made that tape Ba is an Orion tape; the program is monitored if insufficient pages are reserved for the transfer.
The maximum transfer is 4096 words; no indication is given of how many words have been read.

1047 Read Orion tape backwards
Read the previous block on Orion tape Ba to store blocks P + K, P + K-1,....,P. As 1046, except the first word read is stored in address 511 of block P + K, the next in 510 and so on.

6.2. Input and Output Routines 1050 - 1072

6.2.1 Input:

1050 Select input n
Succeeding read orders refer to the date called Input n in the Job Description. Input 0 is assumed if read orders occur without previous use of 1050

1051 Find selected input
ba′ = number of currently selected input

1052 Find peripheral equipment number
ba′ = V-store address of the peripheral used for the currently selected input. (ba′ = 0 if this input originated as output from another program)

1053 Test binary/internal code
ba′ = n if next character read is a binary character. (ba unaltered if in internal code)

1054 ba′ = next character/c′ = n at end of record.
Read the next 6-bit character to digits 18-23 of Ba, clearing bits 0-17. For binary input, which is in 12-bit quarter words, the first use of 1054 reads the more significant 6 bits, the next gives the less significant 6 bits. If end of record has just been exceeded, c′ = n and ba′ = carriage control character in bits 18-23.

Carriage control character:-

Octal value:	Meaning:
00	ignored, used to end binary records
01 to 17	1 to 15 line feeds, no carriage return
20	carriage return
21 to 37	carriage return and 1 to 15 line feeds
40 to 47	paper throw, no carriage return. Home on channels 0 to 7
50 to 57	paper throw with carriage return. Home on channels 0 to 7
60 to 67	Spare

1055 ba′ = number of blocks read

1056 Read ba half words to S onwards
ba′ is unchanged except for bit 0 set = 1 and bits 22, 23 set = 0
If end of record is reached, ba′ = number of characters read.

1057 Read next record to S onwards
On exit, ba′ = number of characters read

6.2.2 Output.

1060 Select output n
Succeeding write orders are to the peripheral called Output n in the Job Description.
Bit 23 of n = 1 if binary, 0 if internal code
Output 0 is assumed if write orders occur before any use of 1060.

1061 Find selected output
ba′ = number of currently selected output, (with bit 23 as in 1060).

1062 Find peripheral equipment type
ba′ = V-store address of equipment number 0 of the peripheral type currently selected for output (ba′ = 0 if currently to any peripheral).

1064 Write character n
Write the character in bits 18-23 of n. If binary mode, 1064 has to be used twice to write the more and less significant halves.

1065 End this record
Writes the carriage control character in digits 18-23 of n to the selected output and terminates the record. (If binary mode, n = 0 usually; this last character is always ignored at the time of printing).

1066 Write ba halfwords from S
If bit 0 of ba = 1 the record is not ended; if bit 0 is = 0 the record is ended and the last character (in S+ba-0.1) is the carriage control character.

1067 Write a record from S
The effect is the same as 1066 with bit 0 of ba equal to zero.

1070 Rename output n as input ba

1071 Break output n

1072 Define output n, with ba = maximum amount of output (in blocks) and ba* = destination of output.

6.3 Subroutine Entry, Branching, Monitoring and Miscellaneous Transfers

6.3.1 Subroutine Entry.

1100	Enter subroutine at s, $ba' = c + 1$	6
1101	Enter subroutine at n, $ba' = c + 1$	5
1102	Enter subroutine at bm, $ba' = c + 1$	6

6.3.2 Branch instructions

1103 Establish Ba branches. Each branch will preserve the accumulator and the contents of B-registers Bn, B(n+1), B(n+2),··, B 99, B 119, B 121, B 126, B 127; where Bn is given in bits 15 - 21 of n

1104 Start branch with number Ba at n ($0 \leq Ba \leq 63$).
The branch is given priority Ba, but has lower priority than any branches already defined with number Ba. The main program is automatically defined as a branch 0.

1105 Kill all branches with number Ba. If Ba = 64 kill the current branch only.

1106 If any branch with number Ba is active, halt current branch.

1107 $c' = n$ if any branch with number Ba is active.

6.3.3 Monitor.

1112 Set monitor jump to n
If the program is terminated other than by extracode 1117 (end program) enter a private monitor sequence at n. The octal fraction of n, (K), has the following effects.

If $K = 0$, $c' = n$ after printing the type of fault
$K = 1$, $c' = n$ immediately
$K = 2$, $c' = n$ after standard post mortem printing

1113 Set restart
Preserve Supervisor working registers associated with this program. Should a future restart be necessary, recover these and re-enter this program with $c' = n$

1114 Exit from trap
After trapping a computer error or an off line failure, permit resumption of the program and recovery of working registers. n is interpreted as follows:

If $n < 0$, monitor and end program
$n = 0$, restart program completely
$n > 0$, and odd, resume program by $c' = n$
$n > 0$ and even, recover extracode working registers, then $c' = n$

1115 If monitored, dump on to tape Ba from section n

1116 If monitored, do not dump

1117 End program

6.3.4 Miscellaneous Transfers.

1120 $ba' = $ clock

ba digits	0-3,	4-7,	8-11,	12-15,	16-19,	20-23
value	Tens	Units	Tens	Units	Tens	Units
	Hours		Minutes		Seconds	

1121 ba′ = date

ba digits	0-3,	4-7,	8-11,	12-15,	16-19,	20-23
value	Tens	Units	Tens	Units	Tens	Units
	Day		Month		Year	

1122 ba′ = local instruction counter
Set ba′ = number of instructions to be obeyed (units of 2048) before the local instruction counter is exceeded.

1123 Set local instruction counter = 2048n

1124 v6′ = n
The least significant six bits of V-store line 6 are used as follows:

Digit	18	19	20	21	22	23
Value	12/13 shift on division instructions (needed to adjust remainder for 376, 377 instructions)	Sign of quotient (Qs) in Basic division instructions	A0	Bt		Bc
Set = 1	12 shift	Qs<0	A0 set	bt<0	bt≠0	Bc set
Set = 0	13 shift	Qs≥0	A0 clear	bt≥0	bt=0	Bc clear

1125 ba′ = v6 & n

1126 v7′ = n. Hoot, operated by least significant digit of n

1127 ba′ = v7 & n. Read engineers handswitches, digits 16 - 23

6.4 Searches, Traps, Compiler & Supervisor Routines. 1131-1157

6.4.1 Searches and Traps

1131 Table search for s, starting at C(ba)
ba′ = address of successful halfword, or -2^{20} if unsuccessful.
c′ = c + 2. C(c + 1) is used to specify parameters k, l, m as shown below. Up to l + 1 halfwords are selected, starting from C(ba), continuing at intervals of k halfwords, which are masked with m before comparison with s

Bits	0 - 9,	10 - 20,	21-23	0 - 23
value	k	l		m
	interval	count	spare	mask

1132 Set address of trap vector to S
Store words S onwards contain trap vector. Each word is used as follows:
More significant halfwords = trap jump address (if this is < 0, no trapping required for this entry in vector).
Less significant half, bits 15-21 = B-register in which main control is to be preserved.
If S<0, revert to no trapping.

1133 $ba' =$ trap address
$ba' =$ start of trap vector, (ba' set <0 if no trapping vector defined).

1134 Trap Ba
Jump to address given in entry Ba of the trap vector, preserving c in the B-register specified in entry Ba.

1135 $c' = n$ if block label $\geq$ ba defined
If in the future a block is newly defined by non-equivalence with a block label $\geq$ ba, then $b91' = c$ and $c' = n$

1136 $am' =$ number of instructions obeyed, from start of program, as a fixed point integer with exponent 16 as the number is in multiples of 2048

6.4.2 Compiler and Supervisor

1140 Read "parameter" Ba of program to store starting at location S

Ba	Parameter
0	Job title (10 words)
1	Computing time estimate, in seconds, in digits 0-23 (One half word)
2	Execution time estimate, " " "
3	No. of store blocks required, in digits 1-11 (One half word)
4	"Parameter" in Job Description (One half word)
5	Logical tape numbers defined (8 half words) The j^{th} digit ($0 \leq j \leq 15$) of the i^{th} half word is a 1 if tape number $16i + j$ is defined
6	Inputs defined (One half word) The i^{th} digit ($0 \leq i \leq 15$) is 1 if input stream i is defined
7	Outputs defined (One half word) As for 6

1142 End compiling. Used normally only by a compiler.
Informs the Supervisor that compilation has ended, so that the number of store blocks which may be used is now restricted to that specified in the program's Job Description

1143 Call system document s to be input stream ba

1144 Call system document s to store block ba onwards

1147 Call in library program n and set $c' =$ ba

1150 Assign ba blocks, labels P to (P+ba-1) to overflow K

1151 Set up blocks P onwards from overflow K

1156 Enter extracode control at n if the "In Supervisor" switch is set

1157 Enter Extracode control at n if the "Process" switch is set

Note: extracodes 1156 and 1157 are for use only by Supervisor routines which use Main Control. They will cause a monitor if used by ordinary programs.

6.5 Store Routines 1160 - 1177

1160 Read block P.
P is in bits 1 - 11 of n; bit 0 of n is 0 if operands section of store preferred, or 1 if instruction section preferred.
The requested transfer is inserted in the drum queue. If P was in the drum store, it is read to core store, its drum store space made empty and the learning program entered to select a page to write to the drum store.

1161 Release block P
Parameters are set so the learning program will choose this page next to write to the drum.

1162 Duplicate read
P_1 = bits 1 - 11 of n, P_2 = bits 1 - 11 of ba
A copy of block P_1 is made as P_2, always leaving P_2 in the core store.

1163 Duplicate write
P_1 and P_2 as in 1162. A copy of P_1 is made as P_2 in the drum store. P_1 is always left in the core store.

1164 Rename
P_1 and P_2 as in 1162. Block P_1 is renamed as block P_2 and the original block P_2 is lost.

1165 Write block P.
The requested transfer is inserted in the drum queue. Block P is written to the next empty sector if it was in core store and its core store space made empty. (In general, 1161 will release a block from core store faster than 1165)

1166 Read block P to absolute page p
p is an integer, in the integer position of ba, which gives the absolute core store page that block P is to occupy. The contents of p, say Px, are copied to a free page which is then called Px, and block P is transferred to page p. 1166 allows complete manipulation of the store for exceptional programs in which this is worth while.

Note: before using 1166 the programmer must set a trap, in case page p is locked down and reserved by the supervisor.

1167 Lose block P
The sector or page occupied by P is made empty.

1170 Clear/not clear new blocks
This supervisor switch is set initially for each program to "clear", so new core store pages allocated to the program are cleared to floating point zero. 1170 sets and resets this switch. If $n < 0$, clear blocks not required, if $n \geq 0$ clear blocks again required.

1171 Store allocation = n blocks

1172 Pages available
ba' = the number of core store pages unallocated (at that moment).

1173 Blocks available
ba' = the number of blocks in the one-level store which are unallocated (at that moment)

1174 Reserve band d
A band of the drum store is reserved for the program and this band is allocated the number d given by bits 13-20 of n.

1175 Read K-1 blocks from band d
d = bits 13-20 of ba
P = bits 1-11 of n, K = bits 21-23 of n $(0 \leq K \leq 5)$

A multiple block transfer takes place, which reads K+1 blocks (K taken modulo 5) to form pages P, P+1,•••, P+K in the core store

1176 Write K+1 blocks to band d
d, P and K as in 1175. A multiple block transfer takes place, which writes pages P, P+1,•••, P+K to the program band d and frees these core store pages.

1177 Lose band d
The band d, given by bits 13-20 of n, is made free and returned to the one level store.

6.6 Test instruction

1200	ba′ = n if Accumulator verflow (A0) is set; clear A0	9
1201	ba′ = n if A0 is not set; clear A0	7
1206	ba′ = n if most significant character in G is zero	4
1216	ba′ = n if bm $>$ 0	5-6
1217	ba′ = n if bm $\leq$ 0	4-5
1223	ba′ = n if B-carry is set	4
1226	ba′ = n if bt $>$ 0	4-6
1227	ba′ = n if bt $\leq$ 0	3-5
1234	c′ = c + 2 if am is approximately equal to s	11
1235	c′ = c + 2 if am is not approximately = s	11
	Approximate equality is defined by $\left\|\frac{am - s}{am}\right\| < C(ba)$ with am standardised and $\neq 0$ If am = 0, am is not approx. = s.	
1236	ba′ = n if am $>$ 0	4-6
1237	ba′ = n if am $\leq$ 0	3-5
1727	c′ = c + 1, 2 or 3 as am >, =, or < s	7
1736	c′ = c + 2 if $\|am\| \geq$ s	8
1737	c′ = c + 2 if $\|am\| <$ s	7

6.7 Character Data Processing 1250 - 1253

1250	ba′ (digits 18-23) = s, ba′ (digits 0-17) = 0	7-10
1251	s′ = ba digits 18-23. Other characters in S unaltered	11-18

In 1250 and 1251, S is taken as a character address

1252	Unpack n characters, starting from character address C(ba), to half words from C(ba*), placing one character at the foot of each half word and clearing the other digits	16+int.pt.($6\frac{3}{4}$n)
1253	Pack n characters from digits 18-23 of half words starting from C(ba*) into successive character positions starting from C(ba)	18+5n

6.8 B-register Operations

1300	ba′ = integral part of s, am′ = fractional part of s	10
1301	ba′ = integral part am, am′ = fractional part am	9
1302	ba′ = ba.n	23-24
1303	ba′ = -ba.n	22-23
1304	ba′ = int.pt. (ba/n), b97′ = remainder	25-28

In 1302-1304, ba and n are 21 bit integers in digits 0-20
Octal fractions are rounded towards zero

1312	ba′ = ba.n	23-24
1313	ba′ = -ba.n	22-23
1314	ba′ = int.pt. (ba/n), b97′ = remainder	25-28

In 1312-1314, ba and n are 24 bit integers

1340	ba′ = ba.2^{-n}; unrounded arithmetic shift right	10-22
1341	ba′ = ba.2^{n}; unrounded arithmetic shift left	9-21
1342	ba′ = ba circularly shifted right n places	10-19
1343	ba′ = ba circularly shifted left n places	9-18
1344	ba′ = ba logically shifted right n places	10-21
1345	ba′ = ba logically shifted left n places	9-20
1347	h′ = h v ba	5
1353	ba′ = position of most significant 1 bit in bits 16-23 of n. (as B 123).	7
1356	bt′ = ba ≠ h	5
1357	bt′ = ba ≠ n	4
1364	ba′ = (ba & n) v (bm & n); b 119′ (bm ≠ bm) & n	6
1371	b121 = Ba, b119′ = N+bm	2
1376	bt′ = ba & h	5
1377	bt′ = ba & n	4

6.9 Complex Arithmetic

The complex accumulator, Ca, is a pair of consecutive registers, the first register having address ba. (If Ba = 0, Ca is locations 0, 1). s: is a number pair. Ca may coincide with S: but not overlap with it. a is spoiled.

1400	ca′ = log s:	
1402	ca′ = exp. s:	140
1403	ca′ = conj. s:	5
1410	ca′ = $\sqrt{s}$:	Max.117
1411	am′ = arg. s: radians	
1412	am′ = mod. s:	Max.53
1413	ca′ = s cos s*, s sin s*	95
1414	ca′ = recip. s:	15
1420	ca′ = ca + s:	8
1421	ca′ = ca - s:	8
1424	ca′ = s:	6
1425	ca′ = - s:	6
1456	s:′ = ca	5
1462	ca′ = ca.s:	18

6.10 Vector Operations 1430 - 1437

The vectors are of order n. $\underline{s_1}$ is stored in consecutive locations from ba, and $\underline{s_2}$ from ba.* a is spoiled.

1430	$\underline{s_1}' = \underline{s_1} + \underline{s_2}$	9 + 4n
1431	$\underline{s_1}' = \underline{s_1} - \underline{s_2}$	9 + 4n
1432	$\underline{s_1}' = am. \underline{s_2}$	10 + 4n
1433	$\underline{s_1}' = \underline{s_1} + am. \underline{s_2}$	10 + 5n
1434	$\underline{s_1}' = \underline{s_2}$ (forwards or backwards)	13 + 3n
1436	$am' = \sum_{i=0}^{n-1} s_{1\,i} . s_{2\,i}$	10 + 5n
1437	$a' = \sum_{i=0}^{n-1} s_{1\,i} . s_{2\,i}$	10 + 13n

6.11 Miscellaneous B-type Accumulator Operations

1452 $m' = m.xs.\ 8^{ya+ys-ba}$, $ya' = ba$. (X) 19 - 23

1415 Generate pseudo random numbers (PRN's) in A and in S (or S*) from numbers in S and S*.
This extracode may be used in several ways.

1. With digit 21 of $S = 0$, the PRN is placed in S and A.

(i) If $s^*y = 0$, $sx>0$ and $s^*x>0$, then s' will be a PRN in the range 0 to 8^{sy}, rectangularly distributed and fixed-point (i.e. sx' is a fixed-point PRN and $sy' = sy$). a' will be a PRN in the range 0 to $s^*x.\ 8^{sy}$ (with $al' = s'$).

(ii) If $s^*y = 0$, $sx<0$ and $s^*x>0$, then as (i) except that the ranges become -8^{sy} to 0 and $-s^*x.8^{sy-1}$ to 0 respectively.

(iii) If $s^*y = 0$ and $s^*x<0$, then as (i) except that the PRN's alternate in sign.

2. With digit 21 of $s = 1$, the PRN's are generated in S* and A instead of S and A. The cases are as for 1., interchanging S and S* throughout.

3. Two successive uses of the extraoode, with digit 21 of S first $= 0$ and then $= 1$, and with $sy = s^*y = 0$, will set PRN's in S and S*, both rectangularly distributed in the range 0 to 1. A will contain the product of two PRN's and so will be distributed in the range 0 to 1 with the probability $-\log x.\delta x$ of being in the neighbourhood δx of x.

In all cases the generation process must be started with Sx and S*x containing numbers with a random mixture of binary digits, and with their least significant bits set to 1.

1466 $a' = C(N + bm + ba).C(N + bm) + a$ 18

1467 $am' = \sum_{r=0}^{ba} s._r\ am^r$ where $S_r = S + r$, 6 + 3n

1473 $m' = (xa/xs).\ 8^{ay-sy-ba}$; $ay' = ba$. (X) 24 - 28

1474 $C(ba)' =$ quotient (am/s), $am' =$ remainder. (X) 20 - 29

1475 $C(ba)' =$ quotient (a/s), $am' =$ remainder (X) 19 - 28

1476 $C(ba)' =$ quotient $([\text{int.pt.am}]/s)$, $am' =$ remainder (X) 28 - 37

1477 Remainder and adjusted integral quotient when used after extracode divisions 1574, 1575, 1774 or 1775 (with no other extracode in between and am not altered)

$s' =$ adjusted integral quotient, $am' =$ remainder.

The type of remainder is determined by the Ba digits as follows:

Ba	Sign of Remainder
0	Same as denominator
1	Opposite to denominator
2	Same as numerator
3	Opposite to numerator
4	Same as quotient
5	Opposite to quotient
6	Positive
7	Negative

6.12 Double Length Arithmetic

The double length number is $s{:} = s + s^*$, where $sy - 13 \geq sy^*$. s: and al are assumed to be positive numbers

1500	$a' = a + s{:}$	10
1501	$a' = a - s{:}$	10
1502	$a' = - a + s{:}$	14
1504	$a' = s{:}$	4
1505	$a' = - s{:}$	3
1542	$a' = a.s{:}$	15
1543	$a' = -a.s{:}$	19
1556	$s{:}' = a$	5
1565	$a' = - a$	5
1566	$a' = \|a\|$	4-6
1567	$a' = \|s{:}\|$	5
1576	$a' = a/s$	19

6.13 Arithmetic Using Address as Operand

The address is taken as a 21 bit integer with one octal fractional place. Fixed point operations imply an exponent of 12.

1520	$am' = am + n$	10
1521	$am' = am - n$	9
1524	$am' = n$, $l' = 0$	8
1525	$am' = - n$, $l' = 0$	7
1534	$am' = n$, $l' = 0$ (X)	10
1535	$am' = -n$. $l' = 0$ (X)	9
1562	$am' = am.n$	8
1574	$am' = am/n$	16
1575	$am' = a/n$	15

6.14 Logical Accumulator Operations

The logical accumulator G is B98 and B99

1204	$ba' =$ no. of 6 bit characters from most significant end identical in g and s	10 - 31
1206	$ba' = n$ if most significant character in G is zero	4
1265	$g' = 2^6 g + n$, $ba' =$ overflow from g.	11
1601	$g' = s$	3
1604	$g' = g + s$	7
1605	$g' = g + s$ with end around carry	12
1606	$g' = g \not\equiv s$	4
1607	$g' = g \& s$	3
1611	$g' = \overline{g}$	3
1613	$s' = g$	3
1615	$am' = g$	4
1630	$g' = g \& \overline{s}$	5
1635	$g' = am$	4
1646	$g' = g \vee s$	3
1652	$bt' = g - s$	7-9

6.15 Half word packing

h has an 8 bit exponent and a 16 bit argument

1624	$am' = h$	6
1626	$h' = am$, with h rounded	8

6.16. Functions and Miscellaneous Routines

The operand in some of these instructions is specified as aq. The associated routines all begin by standardising the accumulator and then truncating it to single length, so aq can be defined as the first 13 significant octal digits of a.

1700	$am' = \log s$	
1701	$am' = \log aq$	
1702	$am' = \exp s$	43
1703	$am' = \exp aq$	42
1704	$a' =$ int.pt. s	5
1705	$a' =$ int.pt. a	4
1706	$a' =$ sign s	5-6
1707	$a' =$ sign a	4-5
1710	$am' = \sqrt{s}$	34-36
1711	$am' = \sqrt{aq}$	34-36
1712	$am' = \sqrt{aq^2 + s^2}$	44
1713	$am' = aq^s.\ (am \geq 0)$	
1714	$am' = 1/s$	4
1715	$am' = 1/am$	4
1720	$am' = \arcsin s\ (-\pi/2 \leq am \leq \pi/2)$	
1721	$am' = \arcsin aq$	
1722	$am' = \arccos s\ (0 \leq am \leq \pi)$	
1723	$am' = \arccos aq$	
1724	$am' = \arctan s\ (^{-}\pi/2 < am < \pi/2)$	
1725	$a,' = \arctan aq$	
1726	$am' = \arctan (aq/s)\ \ (-\pi \leq am \leq \pi)$	
1730	$am' = \sin s$	31-36
1731	$am' = \sin aq$	31-36
1732	$am' = \cos s$	31-36
1733	$am' = \cos aq$	31-36
1734	$am' = \tan s$	34
1735	$am' = \tan aq$	33
1752	$m' = ax.\ 8^{12};\ ay' = ay - 12$ (X)	10
1753	$ax' = m.\ 8^{-12},\ ay' = ay + 12$ (X)	6
1754	round am by adding; standardise	6
1755	$ax' = ax.\ 8^{ay-n};\ \ ay' = n$ (X)	17
1756	$s' = am,\ am' = s$	8
1757	$am' = s/am$	4
1760	$am' = am^2$	3
1762	$m' = ax.\ 8^{12}$ (X)	9
1763	$ax' = m.\ 8^{-12}$ (X)	5

1764	$ax' = ax.8^{n}$ (X)	17
1765	$ax' = ax.8^{-n}$ (X)	12
1766	$am' = \|s\|$ (X)	4
1771	$b121' = Ba$, $b119' = N + ba + bn$	2
1772	$m' = (m.sx)\ 8^{12}$; $ay' = ay + sy - 12$ (X)	11
1773	$m' = (ax/sx)8^{ay-sy-12}$; $ay' = 12$ (X)	27
1774	$am' = am/s$	10
1775	$am' = a/s$	9
1776	Remainder and integral quotient. When used after division extracodes 1574, 1575, 1774 or 1775, with no other extracodes in between and am not altered: s' = integral quotient, am' = remainder. The remainder has the sign of the denominator. (See also 1477)	13

MHJB/dmh
18.12.62

10264056