Appendix A
References

AUP 4	Mercury Orion Atlas Autocode.
CS 271	Operating Instructions for the Teletype Punch.
CS 294A	Punched Tape Codes.
CS 298A	Atlas Magnetic Tape.
CS 308B	Punched Tape Codes (5- and 7-track).
CS 309A	Extracode Functions.
CS 318B	Making a Fortran II Program suitable for use with the Atlas Fortran Compiler.
CS 339	Operating Instructions for the Model 'B' Flexowriter.
CS 345	List of Atlas Basic Instructions.
CS 349A	Atlas Programming Exercises.
CS 360	Telex Data Links.
CS 361	Keyboard used with Ferranti Computers.
CS 362	Operator's Conventions for Punched Tapes.
CS 363	Specifications of Paper for Punched Tapes.
CS 364	Specifications of Dimensions for Punched Tapes.
CS 365	Creed Teleprinter with Integral Tape Reader and Punch.
CS 366	Model 'S' Flexowriter.
CS 367	Creed Keyboard Punch for 7-track tapes.
CS 368	Creed Verifier for 7-track tapes.
CS 377	Algol 60 Report.
CS 378A	Reference manual for Atlas Algol (provisional).
CS 379A	A primer of Algol 60 for Atlas.
CS 384	Summarised programming information.
CS 390	Primer of Fortran Programming.
CS 401	The Analysis of Plane Structural Frames.
CS 402/5052	Extended Mercury Autocode for Orion and Atlas.
CS 405/5055	Traffic Assignment.
CS 411	The Atlas 1 Computer System Operators' Manual.
CS 428	Atlas user's description of the L.P. input scheme.
R 40	An Assembly programme for a Phrase-Structure Language.
R 55	The Manchester University Atlas operating system and Users' description.

R 58	The Atlas supervisor.
R 67	One-level storage system.
R 68	The Atlas scheduling system.
R 70	Processing commercial data on Atlas.
R 74	Central Control Unit of the Atlas Computer.
R 76	The Compiler Compiler.

Appendix B

Notation

Most symbols are used with two completely different meanings. The interpretation to be given to a symbol depends on its context. A convenient division is whether it is used in describing the arithmetic or the basic language, so the notation is listed under these two headings.

1. ARITHMETIC

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

(a) Suffices

x	the argument of the prefixed location
y	the exponent of the prefixed location
:	two consecutive registers, starting with the one suffixed
*	the register following that suffixed

(b) Accumulator

A	the full double-length accumulator, holding a*x* 79-bit mantissa a*x* and 8-bit exponent a*y*
al	the 48-bit floating-point number formed from l, ls and a*y*
am	the 48-bit floating-point number formed from m and a*y*
aq	the single-length number which is obtained by standardising, rounding and truncating the contents of the accumulator
L	the less-significant half of A*x*, of 39 bits with no sign
Ls	the sign bit associated with L
M	the more-significant half of A*x*, of 40 bits, the most-significant bit being the sign digit
AO	accumulator overflow
DO	division overflow
E or EO	exponent overflow
Q	standardised
R	rounded, by forcing a 1 in the least-significant digit of M if L is not clear
R+	rounded, by adding a 1 to the least-significant digit of M if the most-significant bit of L is a 1

(c) General

B	any B-register (index register)
Ba	the B-register specified by the Ba digits of an instruction
Bc	the B-carry digit
Bm	the B-register specified by the Bm digits of an instruction
Bt	the B-test register
C	the main control register (B127)
C()	the contents of the location specified within the brackets
E	the extracode control register (B126)
F	the function digits of an instruction
G	the logical accumulator (B98 and B99)
I	the interrupt control register (B125)
N	the unmodified address part of an instruction (a 24-bit number with the point one octal place from the least-significant end).
n	the modified address part of an instruction regarded as a 24-bit number with the point one octal place from the least-significant end
S	the address of a store location. A full-word address in accumulator instructions (digits 21-23 ignored), a half-word address in B-register instructions (digits 22 and 23 ignored), or a 6-bit character address (all digits relevant)
V	the V-store
Vα	register α of the V-store
X	signifies extracodes suitable for fixed-point working.

2. BASIC LANGUAGE

In practice no distinction is made between capital and small letters, though capitals are used here. However, as an aid to clarity, it is sometimes advantageous to use lower case letters for the separators m, n, v, x and q. Small Greek letters α, β, are used for 21-bit decimal integers, k for the octal number in the 3 least-significant digits of a 24-bit address, and σ for a general octal number of up to 8 octal digits.

Aα	Parameter α, (0 ≤ α ≤ 3999), of the current routine
B	An operator in an expression, causing bits 12-23 of the previous element to be set to zero i.e. it gives a "Block Address"
C	Introduces a string of characters on the next line which are translated into internal code and placed in successive character positions

Dα	An operator, causing the previous element to be logically shifted down α places
E	The enter directive, causes the compiler to evaluate any used parameters etc., insert library routines, delete the compiler and enter the program
ER	As E but the compiler is not deleted and may be used again
EX	The enter interlude directive, to enter a short program for any reason during the compiling process
F	Introduces one or more floating-point numbers on a line, after some expressions otherwise interpreted. Also, if necessary, increases * to the next full-word address
Gα	Global Parameter α (0 ≤ α ≤ 3999)
H	Subsequent expressions on a line are interpreted as 24-bit words and, if necessary, * is increased to the next half-word address
Jσ	σ is octally justified to the left, i.e. the most-significant digit goes into bits 0-2, the next into 3-5 etc.
Kσ	σ is octally justified to the right, to bit 20, i.e. the least-significant digit goes into bits 18-20, the next into 15-17 etc.
Kσ.k	As Kσ, with k going into bits 21-23
Lα.k	Library routine number α, copy k. (.k is omitted if only one copy of the routine is wanted) (α = 1 to 1999, k = 1 to 1999)
M	Alternative to &
N	A separator which non-equivalences the element before it with the element after it in an expression
Pα	Preset parameter α (0 ≤ α ≤ 99 normally)
Q	A separator in an expression which divides the element before it by the element after it, placing the result in digits 0-20
Rα	A directive defining the beginning of routine α (0 ≤ α ≤ 3999)
S	Expressions after the directive S are interpreted as 6-bit characters, i.e. only bits 15-20 of the expression are used
Tα or Tα-β	The title is copied to output channel α or channels α to β inclusive
Uα	An operator causing the previous element to be logically shifted up α places
Uα or Uα-β	Unset preset parameter α, or α to β inclusive

V	A separator in an expression. The element before it is OR-ed with the element after it
W	An operator in an expression, causing bits 0-11 of the previous element to be set to zero, i.e. it gives an address within a block
X	A separator. The element before it is multiplied by the element after it
Yα	α is placed in bits 0-23 (instead of bits 0-20)
Z	A directive indicating the end of a routine
*	The address of the first character position in the location where the item is placed. If used in an expression on the right-hand side of a directive, then * is the address of the next character position
|	All subsequent characters up to NL are ignored (| is not a terminator)
£,π	Alternatives to |
[ ]	All characters between square brackets are ignored. Bracket nesting to any level is allowed
,	Alternative to multiple space as a terminator
&	A separator which logically ANDs the element before it with the element after
:	a special separator used in (a) an element α : β. α modulo 2^{12} goes to bits 0-11 and β modulo 2^{21} to bits 0-20, added to α (b) a floating-point number N (α : β) : γ where the value of the number is N × 10^{α} × 8^{β} and the exponent is forced to γ or standardised if γ is omitted
/	(a) in a parameter, / separates the parameter number and routine number (b) an alternative to :
'(prime)	an operator which forms the logical binary complement i.e. it replaces 1's by 0's and 0's by 1's
?	(a) in the context A α ? = expression, A α is only set to the given expression if no other definite setting of A occurs before the program is entered. Similarly for G α ? = expression (b) in the context P α ? = expression, P α is set equal to the given expression unless P α is already set, in which case the directive is ignored (c) in the context ? expression, causes the compiler to ignore the remainder of the line if the value of the expression is zero.

Appendix C

V-Store Addresses of Peripherals

Each peripheral is allocated one or more words in a part of the V-store associated with its particular type of equipment.

A V-store address is identified by having 6 as its most-significant octal digit; furthermore, since only the more significant half-words are used, the least-significant octal digit of the address is always zero.

That part of the V-store associated with the peripherals is the first 256 words of the block beginning with J6004.

To each type of equipment there corresponds 16 consecutive words, so that peripheral p of type q is allocated the V-store address

J6004 + 16q + p.

The type number q is defined in the following table:-

q	Equipment	V-store address of equipment 0 of each type
0	Card Readers	J60040000
1	Spare (London only: High Speed Data Link)	J60040200
2	TR7 Paper Tape Readers and N.E.P. Tape	J60040400
3	Graphical Outputs	J60040600
4	Anelex Line-printers	J60041000
5	I.B.M. Magnetic Tape	J60041200
6	Fast Paper Tape Punches	J60041400
7	TR5 Paper Tape Readers	J60041600
8	Teletype Punches	J60042000
9	Card Punches (and, Manchester only, X-ray Diffractometer)	J60042200
10	Spare (Manchester only: A.T. & E. On-Line Data Links)	J60042400
11	Teleprinters	J60042600

The addresses above are all for equipment 0 of the type indicated. Card Reader 1, for example, would be addressed by writing J60040010.

Appendix D

Character Codes

The following table lists all the available Atlas Internal Code characters together with their external representations in terms of punchings on 7-track tape, 5-track tape, and punched cards, using the standard Atlas character codes for these media.

The 7-track tape code is the I.C.T./Ferranti Orion/Atlas code. The 5-track code is the standard I.C.T./Ferranti code as used on Pegasus, Mercury, Sirius, Atlas and Orion. The card code is the Atlas Fortran card code.

Also in the table is an indication of which characters are available on the Anelex Line-Printer.

Some characters are designated "Unassigned". This indicates that no external printing characters have been assigned to these Internal Code characters. Most compilers and Input Routines treat these as Illegal Characters. However, these characters have had 7-track paper tape punchings assigned to them. This serves two purposes: (i) it means that, if at some later date characters are assigned to these paper tape punchings, then Internal Code characters are available and assigned to correspond to them, and (ii) it means that, since there is some internal representation for every 7-track paper tape code with odd parity, it is possible to use Internal Code representation for parity-checked "binary" information, rather than use pure "Binary" mode. However, it should be noted that, since the Supervisor treats the shift characters and the New Line characters in a special manner, the internal representation will not be an exact "image" of the external punchings. It should also be noted that some of these Internal Code characters are used by non-standard external codes to represent non-standard characters. Thus in all cases special care should be taken in using these characters.

One character (Outer Set 02) is designated "Spare". This character has no external printing assigned to it nor any 7-track paper tape code. It may however be used by non-standard external codes, and thus care should be taken over its use.

Internal Code character 00 (inner and Outer set) is designated "Not Assigned". This character is reserved for special purposes by the compilers and the Supervisor. It will never have a character assigned to it, and should never be used by normal programs.

Certain characters are not available as standard on any input medium, and may be treated as "spare" by input routines (for example, L100 treats them thus). They are

Outer Set	03	£
Outer Set	35	10
Outer Set	36	11

They have the meaning as shown when used as output characters destined for the line-printer.

Certain other characters have alternatives given in parentheses. This is because the characters are alternatives in one or more of the relevant external codes. (For example, Inner Set 13 is listed as π or £; these are alternatives on 7-track tape, 5-track tape and cards.) However, in the case of the line-printer, only the first character so listed is relevant, and, except for %, the other character appears elsewhere in the table for line-printer purposes.

Note that, in the column for 5-track tape, the 5-track tape code is given with the sprocket hole after two information holes and not three. This is the reverse of the normal convention and is done because the form as printed corresponds to the internal binary representation of the character when "Binary" mode for Input or Output is in use.

The Fault character (77 Inner Set) has no external representation. It is used under certain circumstances on Input by the Supervisor as a translation of any external character which has no Internal Code representation.

Tabulate (02 Inner Set) is treated as a single space by the Supervisor on output equipment where it does not otherwise exist.

The external shift characters (06 and 07) are ignored by the Supervisor for equipments where they have no relevance (e.g. the line-printer).

The fifteen symbols

: ' [ ] < > = _ | ? , ² α β π

are on the fourth quadrant of the line-printer wheel. If none of these symbols are used in a line, the time to print the line is $\frac{1}{1000}$ minute; otherwise it is $\frac{1}{810}$ minute

Notation:

- FS Figure Shift
- LC Lower Case
- LS Letter Shift
- UC Upper Case
- ** A paper-tape character appearing in both shifts
- .. The character concerned is not available on this peripheral.

Internal Code - Inner Set

Character	Internal code (octal)	7-track code (binary bits and case)	5-track code (binary bits and shift)	Atlas Fortran card code (holes punched)	Anelex Line-Printer (availability)
(Not Assigned)	00	..	..	..	..
Space	01	** 0010.000	FS 01.110	None	Yes
Tabulate	02	** 0000.100	..	..	..
Backspace	03	** 0010.101	..	..	..
Shift to outer set	04	..	..	..	..
Shift to inner set	05	..	..	..	..
Shift to LC/LS	06	** 0010.110	** 11.011	..	..
Shift to UC/FS	07	** 0000.111	** 00.000	..	..
(Open brackets	10	LC 0111.000	FS 10.100	0,8,4	Yes
) Close brackets	11	LC 0101.001	FS 01.100	10,8,4	Yes
, Comma	12	LC 0101.111	FS 11.110	0,8,3	Yes
π (£) Pi (Pounds)	13	LC 0111.011	LS 01.111	11,8,3	Yes
? Query	14	LC 0101.100	LS 10.111	11,8,5	Yes
& Ampersand	15	LC 0111.101	..	8,5	Yes
* Asterisk	16	LC 0111.110	FS 11.000	11,8,4	Yes
/ Oblique	17	UC 0011.111	FS 11.101	0,1	Yes
0 Zero	20	UC 0100.000	FS 00.001	0	Yes
1	21	UC 0110.001	FS 10.000	1	Yes
2	22	UC 0110.010	FS 01.000	2	Yes
3	23	UC 0100.011	FS 11.001	3	Yes
4	24	UC.0110.100	FS 00.100	4	Yes
5	25	UC 0100.101.	FS 10.101	5	Yes
6	26	UC 0100.110	FS 01.101	6	Yes
7	27	UC 0110.111	FS 11.100	7	Yes
8	30	UC 0111.000	FS 00.010	8	Yes
9	31	UC 0101.001	FS 10.011	9	Yes
< Less than	32	LC 0100.011	..	0,8,5	Yes
> Greater than	33	LC 0110.100	FS 10.001	10,8,5	Yes
= Equals	34	LC 0100.101	FS 01.010	8,3	Yes
+ Plus	35	UC 0111.101	FS 01.011	10	Yes
- Minus	36	UC 0111.110	FS 11.010	11	Yes
. Point	37	UC 0101.111	** 00.111	10,8,3	Yes

Inner set (continued)

Character	Internal code (octal)	7-track code (binary bits and case)	5-track code (binary bits and shift)	Atlas Fortran card code (holes punched)	Anelex Line-Printer (availability)
'Prime (alternative: **n (letter n) on** 5-track tape only)	40	LC 0100.000	FS 10.111	8,4	Yes
A	41	UC 1010.001	LS 10.000	10,1	Yes
B	42	UC 1010.010	LS 01.000	10,2	Yes
C	43	UC 1000.011	LS 11.000	10,3	Yes
D	44	UC 1010.100	LS 00.100	10,4	Yes
E	45	UC 1000.101	LS 10.100	10,5	Yes
F	46	UC 1000.110	LS 01.100	10,6	Yes
G	47	UC 1010.111	LS 11.100	10,7	Yes
H	50	UC 1011.000	LS 00.010	10,8	Yes
I	51	UC 1001,001	LS 10.010	10,9	Yes
J	52	UC 1001.010	LS 01.010	11,1	Yes
K	53	UC 1011.011	LS 11.010	11,2	Yes
L	54	UC 1001.100	LS 00.110	11,3	Yes
M	55	UC 1011.101	LS 10.110	11,4	Yes
N	56	UC 1011.110	LS 01.110	11,5	Yes
O	57	UC 1001.111	LS 11.110	11,6	Yes
P	60	UC 1110.000	LS 00.001	11,7	Yes
Q	61	UC 1100.001	LS 10.001	11,8	Yes
R	62	UC 1100.010	LS 01.001	11,9	Yes
S	63	UC 1110.011	LS 11.001	0,2	Yes
T	64	UC 1100.100	LS 00.101	0,3	Yes
U	65	UC 1110.101	LS 10.101	0,4	Yes
V	66	UC 1110.110	LS 01.101	0,5	Yes
W	67	UC 1100.111	LS 11.101	0,6	Yes
X	70	UC 1101.000	LS 00.011	0,7	Yes
Y	71	UC 1111.001	LS 10.011	0,8	Yes
Z	72	UC 1111.010	LS 01.011	0,9	Yes
(Unassigned)	73	(UC 1101.011)	..	..	..
(Unassigned)	74	(UC 1111.100)	..	..	..
(Unassigned)	75	(UC 1101.101)	..	..	..
(Unassigned)	76	(UC 1101.110)	..	..	..
Fault	77	..	..	..	..

Internal Code - Outer Set

Character	Internal code (octal)	7-track code (binary bits and case)	5-track code (binary bits and shift)	Atlas Fortran card code (holes punched)	Anelex Line-Printer (availability)
(Not Assigned)	00	..	..	..	..
Space	01	** 0010.000	FS 01.110	None	Yes
(Spare)	02	..	..	..	..
£ Pounds	03	..	..	..	Yes
Shift to outer set	04	..	..	..	..
Shift to inner set	05	..	..	..	..
Shift to LC/LS	06	** 0010.110	** 11.011	..	..
Shift to UC/FS	07	** 0000.111	** 00.000	..	..
(Unassigned)	10	(** 0001.000)	..	..	..
(Unassigned)	11	(** 0011.001)	..	..	..
(Unassigned)	12	(** 0011.010)	..	..	..
(Unassigned)	13	(** 0001.011)	..	..	..
Stop	14	** 0011.100	..	..	..
Punch On	15	** 0001.101	..	..	..
Punch Off	16	** 0001.110	..	..	..
: Colon	17	LC 0011.111	..	.6,8	Yes
∅ (x) Phi (letter x)	20	..	FS 00.011	..	..
[Open square brackets	21	LC 0110.001	..	11,7,8	Yes
] Close square brackets	22	LC 0110.010	..	11,6,8	Yes
→ Arrow	23	..	FS 00.101	..	..
≥ Greater than or equal	24	..	FS 01.001	..	..
≠ Not equal	25	..	FS 10.010	..	..
_ Underline	26	LC 0100.110	..	10,6,8	Yes
| Vertical bar	27	LC 0110.111	..	10,7,8	Yes
² (%) Superscript 2 (Percent)	30	LC 0101.010	..	..	Yes
≈ (v) Curly equal (letter v)	31	..	FS 00.110	..	..
α (10) Alpha (Ten)	32	UC 0101.010	..	..	Yes
β (11) Beta (Eleven)	33	UC 0111.011	..	..	Yes
½ Half	34	UC 0101.100	..	..	Yes
10 Ten	35	..	..	..	Yes
11 Eleven	36	..	..	..	Yes
(Unassigned)	37	(UC 1000.000)	..	..	..

Outer set (continued)

Character	Internal code (octal)	7-track code (binary bits and case)	5-track code (binary bits and shift)	Atlas Fortran card code (holes punched)	Anelex Line-Printer (availability)
(Unassigned)	40	(LC 1000.000)	..	..	..
a	41	LC 1010.001	..	..	..
b	42	LC 1010.010	..	..	..
c	43	LC 1000.011	..	..	..
d	44	LC 1010.100	..	..	..
e	45	LC 1000.101	..	..	..
f	46	LC 1000.110	..	..	..
g	47	LC 1010.111	..	..	..
h	50	LC 1011.000	..	..	..
i	51	LC 1001.001	..	..	..
j	52	LC 1001.010	..	..	..
k	53	LC 1011.011	..	..	..
l	54	LC 1001.100	..	..	..
m	55	LC 1011.101	..	..	..
n	56	LC 1011.110	..	..	..
o	57	LC 1001.111	..	..	..
p	60	LC 1110.000	..	..	..
q	61	LC 1100.001	..	..	..
r	62	LC 1100.010	..	..	..
s	63	LC 1110.011	..	..	..
t	64	LC 1100.100	..	..	..
u	65	LC 1110.101.	..	..	..
v	66	LC 1110.110	..	..	..
w	67	LC 1100.111	..	..	..
x	70	LC 1101.000	..	..	..
y	71	LC 1111.001	..	..	..
z	72	LC 1111.010	..	..	..
(Unassigned)	73	(LC 1101.011)	..	..	..
(Unassigned)	74	(LC 1111.100)	..	..	..
(Unassigned)	75	(LC 1101.101)	..	..	..
(Unassigned)	76	(LC 1101.110)	..	..	..
Erase	77	** 1111.111 **	11.111	..	..

Appendix E

Summary of Extracodes

Allocation of Function Numbers

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

The extracodes are divided into sections as shown below:

1000 - 1077	Peripheral routines.
1100 - 1177	Organisational routines
1200 - 1277	Test instructions and character data-processing.
1300 - 1377	B-register operations.
1400 - 1477	Complex arithmetic, Vector arithmetic, and other B-type accumulator functions.
1500 - 1577	Double-length arithmetic and accumulator operations using the address as an operand.
1600 - 1677	Logical accumulator operations, trigonometric routines and half-word packing.
1700 - 1777	Logarithm, exponential, square root etc., and miscellaneous arithmetic operations.

Where possible, the last two octal function digits correspond to those of similar basic operations.

The extracode function is listed at the left of the page and followed by a reference and a 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.

The extracodes are listed in numerical order, and are also classified by type; some extracodes are therefore given twice.

Extracode	Ref.	Description	Instructions Obeyed

Organisational and Peripheral Extracodes

E.1 Magnetic tape

Block Transfers

1001	9.3.1	Search for section n on tape B_a	
1002	9.3.1	Read next K+1 sections from tape B_a to store blocks, P, P+1...., P+K	
1003	9.3.1	Read previous K sections from B_a to P+K, ..., P.	
1004	9.3.1	Write P, P+1,.., P+K to next K+1 sections on B_a	

Extracode	Ref.	Description	Instructions Obeyed
1005	9.3.1	Move tape B_a forwards K+1 sections	
1006	9.3.1	Move tape Ba backwards K+1 sections	
Organisational Instructions			
1007	9.5.1	Mount next reel of file Ba	
1010	9.5.1	Mount	
1011	9.5.1	Mount free	
1012	9.5.1	Mount on logical channel K	
1013	9.5.1	Mount free on logical channel K	
1014	9.5.2	Write title	
1015	9.5.2	Read title or number	
1016	9.5.2	Unload	
1017	9.5.2	Free tape	
1020	9.5.2	Release tape (pass to another program)	
1021	9.5.2	Release mechanisms	
1022	9.5.2	Re-allocate	
1023	9.5.2	How long?	
1024	9.5.2	Where am I?	
Variable Length Organisation			
1030	9.4.2	Start reading forwards	
1031	9.4.2	Start reading backwards	
1032	9.4.2	Start writing forwards	
1033	9.4.2	Select tape Ba	
1034	9.4.2	Start reading forwards from fixed blocks	
1035	9.4.2	Start reading backwards from fixed blocks	
1036	9.4.3	ba' = selected magnetic tape	
1037	9.4.3	s' = mode of magnetic tape Ba	
Variable Length Transfers			
1040	9.4.3	Transfer	
1041	9.4.3	Skip	
1042	9.4.3	Mark	
1043	9.4.3	Stop	
1044	9.4.3	Word search	
1046	9.7	Read next block on Orion tape	
1047	9.7	Read previous block on Orion tape	
E.2 Input			
1050	8.4	Select input n	
1051	8.4	Find selected input	
1052	8.14	Find peripheral equipment number	
1053	8.14	Test whether binary or internal code	
1054	8.14	Read next character to Ba. Jump to n at end of record	
1055	8.14	ba' = number of blocks read	
1056	8.14	Read ba half-words to S	
1057	8.14	Read next record to S	

Extracode	Ref.	Description	Instructions Obeyed
E.3 Output			
1060	8.4	Select output n	
1061	8.4	Find selected output	
1062	8.15	Find peripheral equipment type	
1063	8.15	Delete output n	
1064	8.15	Write character n	
1065	8.15	End this record	
1066	8.15	Write ba half-words from S	
1067	8.15	Write a record from S	
1070	8.15	Rename output n as input ba	
1071	8.15	Break output n	
1072	8.15	Define output n	
E.4 Subroutine Entry			
1100	7.7	Enter subroutine at s, ba' = c+1	6
1101	7.7	Enter subroutine at S, ba' = c+1	5
1102	7.7	Enter subroutine at bm, ba' = c+1	6
1362	7.7	Enter subroutine at n, b90' = c+1	
E.5 Branching			
1103	12.3.2	Establish Ba branches	
1104	12.3.2	Start branch Ba at S	
1105	12.3.2	Kill Ba. If Ba = 64 kill current branch	
1106	12.3.2	Halt current branch if Ba is active	
1107	12.3.2	Jump to n if Ba is active	
E.6 Monitor			
1112	11.3	Set Monitor jump to n	
1113	11.4.1	Do not restart	
1117	11.1.3	End program	
E.7 Miscellaneous Transfers			
1120	7.8	ba' = clock	
1121	7.8	ba' = date	
1122	12.4	ba' = local instruction counter	
1123	12.4	set instruction counter = n 2048	
1136	12.4	Read instruction counter	
1124	7.8	v6' = n	3
1125	7.8	ba' = v6 & n	4
1126	12.9	v7' = n (hoot)	2
1127	12.9	ba' = v7 & n (read handswitches)	3
E.8 Traps			
1131	7.5.2	See E.12 Character Data Processing	
1132	11.2	Set trap/normal mode	
1133	11.2	ba' = trap address	
1134	11.2	Trap	
1135	12.1	See E.10 Store	
1136	12.1	See E.7 Miscellaneous Transfers	

Extracode	Ref.	Description	Instructions Obeyed

E.9 Compiler and Supervisor

1140	12.9	Read parameter Ba to s	
1141	12.9	Define Compiler	
1142	12.9	End compiling	
1143	12.9	Reserve Supervisor Tape	
1147	12.9	Call compiler n	
1150	12.9	Assign ba blocks, labels P to (P + ba-1), to overflow K	
1151	12.9	Set up blocks P onwards from overflow K	
1155	12.1	See E.10 Store	
1156	12.9	Enter extracode control at n if the "In Supervisor switch" is set	
1157	12.9	Enter extracode control at n if the "Process switch" is set	

E.10 Store

1135	12.1	Jump to n when block ≥ ba defined	
1155	12.1	Find smallest block label defined	
1160	12.1	Read block P	
1161	12.1	Release block P	
1162	12.1	Duplicate read	
1163	12.1	Duplicate write	
1164	12.1	Rename	
1165	12.1	Write block P	
1166	12.1	Read to absolute page	
1167	12.1	Lose block P	
1170	12.1	Clear blocks	
1171	12.1	Store allocation = n blocks	
1172	12.1	ba' = number of pages available	
1173	12.1	ba' = number of blocks available	
1174	12.1	Reserve band n	
1175	12.1	Read K + 1 blocks	
1176	12.1	Write K + 1 blocks	
1177	12.1	Lose band n	

Arithmetic and Logical Extracodes

Accumulator operations are rounded floating-point unless marked X, when they are suitable for fixed-point working.

E.11 Tests

1200	7.6.1	ba' = n if A0 set; clear A0	9
1201	7.6.1	ba' = n if A0 clear; clear A0	7
1204	7.5.3	See E.19 Logical Operations	
1206	7.6.2	ba' = n if most-significant character in g = 0	4
1216	7.6.2	ba' = n if bm > 0	5-6
1217	7.6.2	ba' = n if bm ≤ 0	4-5
1223	7.6.2	ba' = n if Bc = 1	4
1226	7.6.2	ba' = n if bt > 0	4-6
1227	7.6.2	ba' = n if bt ≤ 0	3-5
1234	7.6.1	c' = c + 2 if am approximately = s	11-12
1235	7.6.1	c' = c + 2 if am not approximately = s	11-12

Extracode	Ref.	Description	Instructions Obeyed

Approximate equality is defined by

$\left| \frac{am-s}{am} \right| < C(ba)$, with am standardised

If am = 0, am is not approximately = s

1236	7.6.1	ba' = n if ax > 0	4-6
1237	7.6.1	ba' = n if ax ≤ 0	3-5
1250 to 1253	7.5.2	See E.12 Character Data Processing	
1255	7.6.1	ba' = n if m is neither zero nor all ones	9
1265	7.5.3	See E.19 Logical Operations	
1727	7.6.1	c' = c + 1, c + 2, or c + 3 as am >, = or < s	7
1736	7.6.1	c' = c + 2 if |am| ≥ s	8
1737	7.6.1	c' = c + 2 if |am| < s	7

E.12 Character data processing

1131	7.5.2	Table search	8+6n
In 1250 and 1251, S is taken as a character address			
1250	7.5.2	ba' (digits 18-23) = s, ba' (digits 0-17) = 0	7-10
1251	7.5.2	s' = ba (digits 18-23)	11-18
1252	7.5.2	Unpack n characters starting from character address C(ba), to half-words from C(ba*)	16 + int.pt. ($6\frac{3}{4}$n)
1253	7.5.2	Pack n characters starting from half-word address C(ba*), to character address C(ba)	18 + 5n

E.13 B-register operations

1300	7.5.1	ba' = integral part of s, am' = fractional part of s	10
1301	7.5.1	ba' = integral part of am, am' = fractional part of am	9
1302	7.5.1	ba' = ba.n, rounded away from zero	23-24
1303	7.5.1	ba' = -ba.n rounded away from zero	22-23
1304	7.5.1	ba' = integral part of (ba/n), b97' = remainder	25-28

In 1302-1304, ba and n are 21-bit integers in digits 0-20

1312	7.5.1	ba' = ba.n	23-24
1313	7.5.1	ba' = -ba.n	22-23
1314	7.5.1	ba' = integral part of (ba/n) b97' = remainder	25-28

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

Extracode	Ref.	Description	Instructions Obeyed
1340	7.5.1	ba' = ba. 2^{-n}; unrounded arithmetic shift right	10-22
1341	7.5.1	ba' = ba. 2^{n} ; unrounded arithmetic shift left	9-21
1342	7.5.1	ba' = ba circularly shifted right n places	10-19
1343	7.5.1	ba' = ba circularly shifted left n places	9-18
1344	7.5.1	ba' = ba logically shifted right n places	10-21
1345	7.5.1	ba' = ba logically shifted left n places	9-20
1347	7.5.1	s' = s v ba	5
1353	7.5.1	ba' = position of most-significant 1 in bits 16-23 of n (as B123)	7
1356	7.5.1	bt' = ba ≢ s	7
1357	7.5.1	bt' = ba ≢ n	5
1362	7.7	See E.4 Subroutine Entry	
1364	7.5.1	ba' = (ba & n) v (bm & n); b119' = (ba ≢ bm) & n	4
1371	7.5.1	b121' = Ba, b119' = N + bm	2
1376	7.5.1	bt' = ba & s	5
1377	7.5.1	bt' = ba & n	4

E.14 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 otherwise overlap with it. A is spoiled.

Extracode	Ref.	Description	Instructions Obeyed
1400	7.4.6	ca' = log s:	
1402	7.4.6	ca' = exp s:	140
1403	7.4.6	ca' = conj s:	5
1407	7.4.2	See E.16. Miscellaneous B-type accumulator operations	
1410	7.4.6	ca' = √ s:	Max.117
1411	7.4.6	am' = arg s: radians	
1412	7.4.6	ca' = mod s:	Max. 53
1413	7.4.6	ca' = s cos s*, s sin s*	95
1414	7.4.6	ca' = 1/s:	15
1415	7.4.2	See E.16. Miscellaneous B-type accumulator operations	
1420	7.4.6	ca' = ca + s:	8
1421	7.4.6	ca' = ca - s:	8
1424	7.4.6	ca' = s:	6
1425	7.4.6	ca' = -s:	6
1456	7.4.6	s:' = ca	5
1462	7.4.6	ca' = ca. s:	18

E.15 Vector Operations

The vectors are of order n. s_1 is stored in consecutive locations from ba, and s_2 from ba*. A is spoiled.

Extracode Ref.		Description	Instructions Obeyed
1430	7.4.7	$s_1' = s_1 + s_2$	9 + 4n
1431	7.4.7	$s_1' = s_1 - s_2$	9 + 4n
1432	7.4.7	s_1' = am. s_2	10 + 4n
1433	7.4.7	$s_1' = s_1$ + am. s_2	10 + 5n
1434	7.4.7	$s_1' = s_2$(forwards or backwards)	13 + 3n
1436	7.4.7	am' = $\sum_{i=0}^{n-1} s_{1i} . s_{2i}$	10 + 5n
1437	7.4.7	a' = $\sum_{i=0}^{n-1} s_{1i} . s_{2i}$	10 + 13n

E.16 Miscellaneous B-type accumulator operations

Extracode	Ref.	Description	Instructions Obeyed	
1407	7.4.2	Remainder and adjusted integral quotient	14-31	
1415	7.4.2	Generate pseudo-random number		
1441	7.4.5	See E.18. Arithmetic using address as Operand.		
1452	7.4.3	m' = m.sx. $8^{ay+sy-bay}$, ay' = bay	19-23	(X)
1456) 1462)	7.4.6	See E.14. Complex arithmetic		
1466	7.4.2	a' = C(N+bm+ba) ×C(N+bm) + a	18	
1467	7.4.2	am' = $\sum_{i=0}^{r} s_i x^i$ where x = am, S_i = S+i, r = ba	6 + 3n	
1473	7.4.3	m' = (ax/s x).$8^{ay-sy-bay}$, ay' = bay	24-28	(X)
1474	7.4.3	C(ba)' = quotient (am/s), am' = remainder	20-29	(X)
1475	7.4.3	C(ba)' = quotient (a/s), am' = remainder	19-28	(X)
1476	7.4.3	C(ba)' = quotient ([integral part of am/s), am' = remainder	28-37	

E.17 Double-length arithmetic

The double-length number is s: = s + s* where sy - 13 ≥ s*y. s* and al are assumed to be positive numbers.

Extracode	Ref.	Description	Instructions Obeyed
1500	7.4.4.	a' = a + s:	10
1501	7.4.4.	a' = a - s:	10
1502	7.4.4.	a' = a + s:	14
1504	7.4.4	a' = s:	4
1505	7.4.4	a' = -s:	3
1520) to) 1535)	7.4.5	See E.18. Arithmetic using address as Operand	
1542	7.4.4	a' = a.s:	15
1543	7.4.4	a' = -a.s:	19
1556	7.4.4	a:' = a	5
1562	7.4.5	See E.18. Arithmetic using address as Operand.	

Extracode	Ref	Description	Instructions Obeyed	
1565	7.4.4	a' = -a	5	
1566	7.4.4	a' = |a|	4-6	
1567	7.4.4	a' = |s:|	5	
1574) 1575)	7.4.5	See E.18 Arithmetic using address as Operand		
1576	7.4.4	a' = a/s:	19	

E.18 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.

1441	7.4.5	sx' = ba, sy' = 12	5	
1520	7.4.5	am' = am + n	10	
1521	7.4.5	am' = am - n	9	
1524	7.4.5	am' = n, 1' = 0	8	
1525	7.4.5	am' = -n, 1' = 0	7	
1534	7.4.5	am' = n, 1' = 0	10	(X)
1535	7.4.5	am' = -n, 1' = 0	9	(X)
1562	7.4.5	am' = am.n	8	
1574	7.4.5	am' = am/n	16	
1575	7.4.5	am' = aq/n	15	

E.19 Logical accumulator operations

The logical accumulator G is B98 and B99

1204	7.5.3	ba' = number of 6-bit characters from most-significant end identical in g and s	10-31	
1265	7.5.3	$g' = 2^6 g$ + n, ba' = 6-bits shifted out of g	11	
1601	7.5.3	g' = s	3	
1604	7.5.3	$g' = g$ + s	7	
1605	7.5.3	$g' = g$ + s with end around carry	12	
1606	7.5.3	$g' = g \not\equiv$ s	4	
1607	7.5.3	$g' = g$ & s	3	
1611	7.5.3	$g' = \bar{g}$	3	
1613	7.5.3	s' = g	3	
1615	7.5.3	am' = g	4	
1624) 1626)	7.4.8	See E.20 Half-word Packing		
1630	7.5.3	$g' = g$ & $\bar{s}$	5	
1635	7.5.3	g' = am	4	
1646	7.5.3	$g' = g$ v s	3	
1652	7.5.3	bt' = g - s	7-9	

E.20 Half-word packing

s has an 8-bit exponent and a 16-bit mantissa.

1624	7.4.8	am' = s	6	
1626	7.4.8	s'= am, with s rounded	8	

Extracode	Ref.	Description	Instructions Obeyed	

E.21 Functions and miscellaneous routines

1700	7.4.1	am' = log s		
1701	7.4.1	am' = log aq		
1702	7.4.1	am' = exp s	43	
1703	7.4.1	am' = exp aq	42	
1704	7.4.2	am' = integral part of s	5	
1705	7.4.2	am' = integral part of am	4	
1706	7.4.2	am' = sign s	5-6	
1707	7.4.2	am' = sign am	4-5	
1710	7.4.1	am' = $\sqrt{s}$	Max 42	
1711	7.4.1	am' = $\sqrt{aq}$	Max 41	
1712	7.4.1	am' = $\sqrt{aq^2 + s^2}$	Max 50	
1713	7.4.2	am' = aq^s		
1714	7.4.2	am' = 1/s	4	
1715	7.4.2	am' = 1/am	4	
1720	7.4.1	am' = arc sin s $(-\pi/2 \leq s \leq \pi/2)$		
1721	7.4.1	am' = arc sin aq		
1722	7.4.1	am' = arc cos s $(0 \leq s \leq \pi)$		
1723	7.4.1	am' = arc cos aq		
1724	7.4.1	am' = arc tan s $(-\pi/2 < s < \pi/2)$		
1725	7.4.1	am' = arc tan aq		
1726	7.4.1	am' = arc tan (aq/s) $(-\pi \leq aq \leq \pi)$		
1727	7.6.1	See E.11 Test Instructions		
1730	7.4.1	am' = sin s	41	
1731	7.4.1	am' = sin aq	40	
1732	7.4.1	am' = cos s	42	
1733	7.4.1	am' = cos aq	41	
1734	7.4.1	am' = tan s	34	
1735	7.4.1	am' = tan aq	33	
1736) 1737)	7.6.1	See E.11 Test Instructions		
1752	7.4.3	m' = $ax. 8^{12}$; ay' = ay - 12	10	(X)
1753	7.4.3	ax' = m. 8^{-12}, ay' = ay + 12	6	(X)
1754	7.4.2	Round am by adding; standardise	6	
1755	7.4.3	$ax' = ax. 8^{ay-ny}$; ay' = ny	17	(X)
1756	7.4.2	s' = am, am' = s	8	
1757	7.4.2	am' = s/am	4	
1760	7.4.2	am' = am^2	3	
1762	7.4.3	m' = $ax. 8^{12}$	9	(X)
1763	7.4.3	ax' = m. 8^{-12}	5	(X)
1764	7.4.3	$ax' = ax. 8^n$	17	(X)
1765	7.4.3	$ax' = ax. 8^{-n}$	12	(X)
1766	7.4.3	am' = |s|	4	(X)
1767	7.4.3	am' = |am|	3	(X)
1771	7.5.1	b121' = Ba, b119' = N + ba + bm	2	
1772	7.4.3	m' = $(m.sx)\ 8^{12}$; ay' = ay + s y-12	11	(X)
1773	7.4.3	m' = $(ax/sx)\ 8^{ay-sy-12}$; ay' = 12	27	(X)
1774	7.4.2	am' = am/s	10	
1775	7.4.2	am' = aq/s	9	
1776	7.4.2	Remainder	13	

Appendix F

Summary of Basic Instructions by Function

B-Line Operations

Logic Operations

106 ba′ = ba ≢ s	147 ba′ = ba v s	107 ba′ = ba & s
116 s′ = ba ≢ s	167 ba′ = ba v n	117 s′ = ba & s
126 ba′ = ba ≢ n		127 ba′ = ba & n
		164 ba′ = ba + (bm & n)
		165 ba′ = bm & n

Cyclic Shifts

143 ba′ = $\frac{1}{2}$ ba - s
163 ba′ = $\frac{1}{2}$ ba - n
105 ba′ = (64) ba + s
125 ba′ = (64) ba + n

Index Arithmetic

120 ba′ = n - ba	100 ba′ = s - ba	110 s′ = s - ba
121 ba′ = n	101 ba′ = s	111 s′ = -ba
122 ba′ = ba - n	102 ba′ = ba - s	112 s′ = ba - s
123 ba′ = -n	103 ba′ = -s	113 s′ = ba
124 ba′ = ba + n	104 ba′ = ba + s	114 s′ = ba + s

Set B-Test

150 bt′ = s - ba
152 bt′ = ba - s
170 bt′ = n - ba
172 bt′ = ba - n

Test Instructions

Count

200 If bm≠0, then ba′=n and bm′=bm+0.4	220 If bt≠0, then ba′=n and bm′=bm+0.4
201 If bm≠0, then ba′=n and bm′=bm+1.0.	221 If bt≠0, then ba′=n and bm′=bm+1.0
202 If bm≠0, then ba′=n and bm′=bm-0.4	222 If bt≠0, then ba′=n and bm′=bm-0.4
203 If bm≠0, then ba′=n and bm′=bm-1.0	223 If bt≠0, then ba′=n and bm′=bm-1.0

Test

234 If ax=0, ba′=n	214 If bm=0, ba′=n	210 If bm odd, ba′=n	224 If bt=0, ba′=n
235 If ax≠0, ba′=n	215 If bm≠0, ba′=n	211 If bm even, ba′=n	225 If bt≠0, ba′=n
236 If ax≥0, ba′=n	216 If bm≥0, ba′=n		226 If bt≥0, ba′=n
237 If ax<0, ba′=n	217 If bm<0, ba′=n		227 If bt<0, ba′=n

Accumulator Operations

	Unstandardised (Pseudo Fixed-Point)	Standardised Floating-Point: Rounded	Standardised Floating-Point: Unrounded	Pseudo Double-Length
Addition and Subtraction	330 a'=am+s AO 331 a'=am-s AO 332 a'=s-am AO	320 am'=am+s QRE 321 am'=am-s QRE 322 am'=s-am QRE	300 a'=am+s QRE 301 a'=am-s QE 302 a'=s-am QE	310 a'=am+s NQE 311 a'=am-s NQE
Transfers In	334 a'=s 335 a'=-s AO		324 a'=s Q 325 a'=-s QE	314 am'=s N 315 am'=-s NAO 344 l'=sx 345 l'=sx,m'=ss
Transfers Out	346 s'=am,a'=0 356 s'=am			347 s'=al,l'=0 357 s'=al
Multiplication	352 l'=m.s EAO 353 l'=-m.sEAO	362 am'=am.s QRE 363 am'=-am.sQRE	342 a'=am.s QE 343 a'=-am.sQE	372 a'=am.s EAO 373 a'=-am.sEAO
Division	375 al'=+a/|s| m'=rem E	374 am'=am/s l'=0 QRE DO	376 al'=+a/|s| m'=rem EDO 377 al'=|am|/|s| m'=rem EDO	
Standardisation and Rounding	361 am'=a RE 354 am'=a R+AO	360 am'=a QRE	340 a'=a QE	355 a'=al.8^{-13}Q
Octal Shifts & Moduli	364 ax'=8ax 365 ax'=ax/8		366 a'=|am| QE 367 a'=|s| QE	
Check Exponent Overflow	341 a'=a E			

AO Accumulator may overflow
N L not cleared
R+ Rounded by adding
○ Cyclic Shift
Q Accumulator standardised
R Accumulator rounded
E Exponent overflow may occur
DO Division overflow may occur

Appendix G Summary of Instructions by numbers

		0	1	2	3	4	5	6	7	
Index Instructions	10	ba'=s-ba Bc	ba' = s	ba'=ba-s Bc	ba' = -s	ba'=ba+s Bc	ba'=(64b)+s	ba' = ba≢s	ba' = ba & s	10
	11	s'=s-ba Bc	s' = -ba	s'=ba-s Bc	s' = ba	s'=ba+s Bc	(s'=ba+s) No Bc	s' = ba≢s	s' = ba & s	11
	12	ba'=n-ba Bc	ba' = n	ba'=ba-n Bc	ba' = -n	ba'=ba+n Bc	ba'=(64b)+n	ba' = ba≢n	ba' = ba & n	12
	13	(NA n-type Bc)	(NA n-type)	(NA n-type Bc)	(NA n-type)	(NA n-type Bc)	(NA n-type)	(NA n-type)	(NA n-type)	13
	14	(As 100)	(As 101)	(As 102)	ba'=(½ba)-s	(As 104)	(As 105)	(As 147)	ba' = ba v s	14
	15	bt'=s-ba Bc	(NA s-type)	bt'=ba-s Bc	(NA s-type)	(NA s-type Bc)	(NA s-type)	(NA s-type)	(NA s-type)	15
	16	(As 120)	(As 121)	(As 122)	ba'=(½ba)-n	ba'=ba+(bm&n) Bc	ba'= bm & n	(As 167)	ba'= ba v n	16
	17	bt'=n-ba Bc	(NA n-type)	bt'=ba-n Bc	(NA n-type)	(NA n-type Bc)	(NA n-type)	(NA n-type)	(NA n-type)	17
Test Instructions	20 (24)	If bm≠0, ba'=n and bm'=bm+0.4	If bm≠0, ba'=n and bm'=bm+1.0	If bm≠0, ba'=n and bm'=bm-0.4	If bm≠0, ba'=n and bm'=bm-1.0	(As 200)	(As 201)	(As 202)	(As 203)	20 (24)
	21 (25)	If bm odd, ba'=n	If bm even, ba'=n	(As 210)	(As 211)	If bm=0, ba'=n	If bm≠0, ba'=n	If bm≥ 0, ba'=n	If bm < 0, ba'=n	21 (25)
	22 (26)	If bt≠0, ba'=n and bm'=bm+0.4	If bt≠0, ba'=n and bm'=bm+1.0	If bt≠0, ba'=n and bm'=bm-0.4	If bt≠0, ba'=n and bm'=bm-1.0	If bt=0, ba'=n	If bt≠0, ba'=n	If bm ≥ 0, ba'=n	If bt < 0, ba'=n	22 (26)
	23 (27)	(As 234)	(As 245)	(As 236)	(As 237)	If ax=0, ba'=n	If ax≠0, ba'=n	If ax ≥ 0, ba'=n	If ax < 0, ba'=n	23 (27)
Accumulator Instructions	30	a'=am+s QE	a'=am-s QE	a'=am-s QE	(As 302)	(As 324)	(As 325)	(As 324)	(As 324)	30
	31	a'=am+s N QE	a'=am-s N QE	(As 302)	(As 302)	am'=s N	am'=-s N AO	(As 324)	(As 324)	31
	32	am'=am+s QRE	am'=am-s QRE	am'=-am+s QRE	(As 322)	a'=s Q	a'=-s QE	(As 324)	(As 324)	32
	33	a'=am+s AO	a'=am-s AO	a'=-am+s AO	(As 332)	a'=s	a'=-s AO	(As 324)	(As 334)	33
	34	a'=a QE	a'=a E	a'=am.s QE	a'=-am.s QE	l'=sx	l'=sx,m'=ss	s'=am,a'=0	s'=al, l'=0	34
	35	(As 340)	(As 341)	a'=am.s AO E sgn l'=sgn m'	a'=-am.s AO E sgn l'=sgn m'	am'=a R+ AO	l'=al.8^{-13}Q	s'=am	s'=al	35
	36	am'=a QRE	am'=a RE	am'=am.s QRE	am'=-am.s QRE	ax'=8ax	ax'=ax/8	a'=|am| QE	a'=|s| QE	36
	37	(As 340)	(As 341)	a'=am.s AO E	a'=-am.s AO E	am'=am/s QRE DO l'=0	al'=a/|s| m'=rem E*	al'=a/|s| m'=rem E DO*	al'=|am|/|s| m'=rem E DO	37

Legend
○ Circle denotes circular shift
N L not cleared (including Ls)
ss sign digit of sx
Q Accumulator standardised
R Accumulator rounded by Forcing
R+ Rounded by adding
E Exponent overflow may occur
DO Division overflow may occur
AO Accumulator overflow may occur

Instructions given in brackets are non-standard and should not normally be used. They are given here only for the sake of completeness.
NA means that no registers are altered as a result of these operations. However instructions designated "s-type" do make a store-reference and thus SVO or Non-equivalence may occur.
The "n-type" instructions do not make any store-reference, and are thus in effect dummy instructions.

* These division instructions are not fully described by the summary. Reference should be made to Chapter 6 before use.
Bc These instructions set B-carry