Instruction sets and instruction times for the Elliott 502 computer.

Note: all references are listed in section E4/X5.

Elliott 502 instruction set.

The instruction set for the Elliott 502 offers a rich variety of addressing modes for a relatively straightforward repertoire of ALU operations. Each 20-bit instruction occupies one word, as follows:

1	1	6	2	10
L	**M**	**F**	**K**	**N**
Interrupt lock	Store mode	Op code	Modifier source	Literal or address

If L = 1 then interrupts are inhibited, or locked, until the succeeding instruction has been obeyed. (Note that, somewhat confusingly, the symbol S was used for this bit in the original Elliott documentation).

M is the *Store Mode* bit, whose interpretation also depends upon the actual instruction being obeyed (as explained below). If M = 0, N represents either an address in fast RAM or N represents a literal (constant). If M = 1, N represents either an address in main RAM or an address in fast RAM that holds the address of an operand in main RAM (ie indirect addressing into main RAM).

F is the 6-bit Function (or Op.code), specifying two octal digits as used in the tables below.

K specifies the source of the address-modifier, thus:

if K = 0, no modification takes place;
if K = 1, use the contents of the B register;
if K = 2, use the contents of the accumulator, A;
if K = 3, use the contents of the Sequence Control Register (or Program Counter), S, which always points to the *next* instruction to be obeyed.

The N bits in the Elliott 502 are generally interpreted in one of four possible ways, depending upon the M bit and the instruction being obeyed, as follows:

for Group 0 & 1 orders: when M = 0 then N is a positive fraction (ie literal);
when M = 1 then N is a main store address;
for Group 2 & 3 orders: when M = 0 then N is a fast store address;
when M = 1 then the contents of the fast store address is used indirectly to address a location in main store;
for Group 7 instructions, N is used to specify a peripheral device.

In the Elliott 502 instruction tables given below, the following abbreviations are used:

a, a'	the old and new contents of the accumulator register, A;
b, b'	the old and new contents of the modifier register, B;
s, s'	the old and new contents of the Sequence Control Register, S;
r, r'	the old and new contents of the auxiliary register, R;
ar, a'r'	the old and new contents of the double-length accumulator {A,R}
N	the N-bits when used as a constant (ie literal);
(N), (N)'	the old and new contents of main store address N;
(n), (n)'	the old and new contents of fast store address N;
((n)), ((n))'	the old and new contents of main store address (n), ie when the main store is being accessed indirectly via a fast store location.
OV	the Overflow indicator (a single-bit register).
T	data (eg a 5-bit character) read in from an input device.

Instructions in Groups 0 to 3 include simple short arithmetical operations on the accumulator and on the B register. The action of instructions in Groups 0 to 3 is dependent upon the setting of the store-mode bit, M, as follows.

Function	***when M = 0***	***when M = 1***	***explanation***
00	b' = b + *N*	b' = b + (N)	ADD B
01	b' = b' – *N*	b' = b – (N)	SUBTRACT B
02	b' = *N* – b	b' = (N) – b	REVERSE SUBTRACT B
03	b' = *N*	b' = (N)	LOAD B
04	-	(N)' = b + (N)	ADD B to store
05	-	(N)' = b – (N)	SUBTRACT B from store
06	-	(N)' = (N) – b	REVERSE SUBTRACT B from store
07	-	(N)' = b	STORE B
10	a' = a + *N*	a' = a + (N)	ADD A
11	a' = a – *N*	a' = a – (N)	SUBTRACT A
12	a' = *N* – a	a' = (N) – a	REVERSE SUBTRACT A
13	a' = *N*	a' = (N)	LOAD A
14	-	(N)' = a + (N)	ADD A to store
15	-	(N)' = a – (N)	SUB A from store
16	-	(N)' = (N) – a	REV SUBTRACT A from store
17	-	(N)' = a	STORE A
20	b' = b +(n)	b' = b + ((n))	ADD B
21	b' = b – (n)	b' = b – ((n))	SUBTRACT B
22	b' = (n) – b	b' = ((n)) – b	REVERSE SUBTRACT B
23	b' = (n)	b' = ((n))	LOAD B
24	(n)' = b + (n)	((n))' = b + ((n))	ADD B to store
25	(n)' = b – (n)	((n))' = b – ((n))	SUBTRACT B from store
26	(n)' = (n) – b	((n))' = ((n)) – b	REVERSE SUBTRACT B from store
27	(n)' = b	((n))' = b	STORE B
30	a' = a + (n)	a' = a + ((n))	ADD A
31	a' = a – (n)	a' = a – ((n))	SUBTRACT A

32	a' = (n) – a	a' = ((n)) – a	REVERSE SUBTRACT A
33	a' = (n)	a' = ((n))	LOAD A
34	(n)' = a + (n)	((n))' = a + ((n))	ADD A to store
35	(n)' = a – (n)	((n))' = a – ((n))	SUBTRACT A from store
36	(n)' = (n) – a	((n))' = ((n)) – a	REVERSE SUBTRACT A from store
37	(n)' = a	((n))' = a	STORE A

Note: for Group 3 instructions, the overflow marker OV is set if the result of any operation is outside the range –1 to +(1 – 2^{-19}).

Group 4 instructions: transfer of control.

40	if b ≠ 0 then b’ = b + 1 and s’ = N	Jump if b not equal to zero and INC B
41	if b ≠ 0 then b’ = b – 1 and s’ = N	Jump if b not equal to zero and DEC B
42	if b = 0 then s’ = N	Jump if b = 0
43	b’ = s and s’ = N	Subroutine entry
44	if a ≥ 0 then s’ = N	Jump if acc ≥ 0
45	if a < 0 then s' = N	Jump if acc < 0
46	if a = 0 then s' = N	Jump if acc = 0
47	if OV = 1 then s' = N and OV = 0	Jump if overflow

Note: for Group 4 instructions and for instructions 50 and 51, the jump-to address is in fast RAM when M = 0 and in main RAM when M = 1. Following the 43 instruction, the most-significant digit of B is zero if the instruction is in fast RAM and one if it is in main RAM.

Group 5 and 6 instructions: miscellaneous and double-length arithmetic.

Function	***when M = 0***	***when M = 1***	***explanation***
50	s' = (n)	s' = (N)	absolute unconditional jump
51	s' = s – (n)	s' = (N)	relative branch
52	a' = a & (n)	s' = a & (N)	logical AND
53	r' = (n)	r' = (N)	load R
54	(n)'= (n) + 1	(N)' = (N) + 1	memory INC
55	(n)' = (n) – 1	(N)' = (N) – 1	memory DEC
56	(n)' = a & (n)	(N)' = a & (N)	memory logical AND
57	(n)' = r	(N)' = r	store R
60	... shift A logically *N* places left ...		logical shift left
61	... shift A arith. *N* places left ...		arithmetical shift left
62	... shift {A,R} arith. *N* places left ...		double-length arithmetical shift
63	a' = SQRT(a + (n))	a = SQRT(a + (N))	Square root
64	a'r' = a x (n)	a'r' = a x (N)	Multiply, double-length
65	a' = a x (n)	a' = (N)	Multiply single-length; R cleared
66	{a,r}/(n) or {a,r}/(N);	r' = quotient; a' = rem.	Divide double-length
67	a' = {a,r}/(n)	a' = {a,r}/(N)	Divide single-length; R cleared

Note: at the conclusion of instruction 64, AR holds the product as a 38-bit signed fraction, the most-significant bit of R being cleared. At the start of instructions 66 and 67, AR together hold a 38-bit signed fraction.

Group 7 instructions: input/output.

70	a' = a + T	add data from input device *N*
71	*Unallocated; causes a dynamic stop.*	
72	*Unallocated; causes a dynamic stop.*	
73	a' = T	read data from input device *N*
74	*Used to control autonomous transfer devices; no details have yet come to light*.	
75	*Unallocated; causes a dynamic stop.*	
76	Causes a Program Break; re-enter at n or N.	
77	a is output to device *N*	write acc to output device *N*

Note: for instructions 70, 73, 74 and 77, N specifies the identity of the peripheral device. Amongst the values of N used for standard devices for the 502, the following are known to have applied. Clearly, other peripherals specific to the classified radar application at RRE would have had their own identities.

N = 0 the 502's Operator's Console;
N = 1 the 5-track paper tape reader;
N = 2 the Program Control Register, PCR.

Further details of the PCR are given above in section 2. In summary, to quote the Elliott 502's manual (reference 3): "... PCR contains an indication of where a given program has been prevented from running; instructions 70/2 or 73/2 can be used to determine this, while instruction 77/2 can activate or inhibit a programme". Note that a Program Break normally occurs as a result of an external demand (interrupt); the 76 instruction is an alternative software method of achieving the same action.

The effect of a 76 instruction is that:

- the S register is set to n or N;
- the stimulus for the current operating priority level is cleared.

This then causes a Program Break. The new priority level will normally be lower than the current one. The address of the 76 instruction will have been stored as the re-entry address to be used when the former priority level is re-stimulated. The 502 console includes switches that can enable or mask each program stimulus (in addition to the control provided by the PCR) and a push-button for emulating each stimulus.

The Elliott 502 was required to handle significant amounts of input/out for the special real-time, on-line applications for which it was designed. The width of the highways for each form of input/output is given below.

Programme Stimuli (ie Interrupts):

Demands	7 bits	into the 502
Replies	7 bits	out of the 502

Direct Input/Output transfers and device control transfers:

Data	20 bits	bi-directional
Address	4 bits	out of the 502
Function (70/73, 74 or 77)	3 bits	out of the 502
Reply & Busy	2 bits	into the 502

Autonomous Data Transfers:

Data	40 bits	bi-directional
Control	12 bits	out of the 502
Control	12 bits	into the 502
Priority Control	36 bits	into the 502
Priority Control	12 bits	out of the 502.

The Elliott 502's autonomous transfer facility, which is similar to techniques referred to in later years as *Direct Memory Access* or *Cycle-stealing,* allows Input/Output to take place in parallel with normal program operation. The mechanism is that when a *Priority Control* signal from a peripheral indicates that transfer of a word is required the computer will, between instructions, automatically access a pointer and a counter held in memory for the peripheral concerned, transfer a word to or from the location indicated by the pointer and then adjust the pointer and counter. Thus, transfer of a block of words is effected without program action. The Elliott 502's 74 instruction is used to indicate to the peripheral the type of action required and to set it in progress. For the Elliott 502, it is up to a program to ascertain the progress of a transfer before using or changing the data concerned. However, on the Elliott 503, for compatibility with 803 programs, an additional memory bit in each word is used to inhibit access to it while the word is involved in an autonomous transfer.

Elliott 502 instruction times.

The approximate times in microseconds for various instructions (see reference 3) are as follows:

Instruction(s)	***operand***	***Instr. in fast RAM***	***Instr. in main RAM***
Groups 0 & 1	literal operand	2.3	3.8
Groups 0 & 1	main RAM operand	4.3	8.2
Groups 2 & 3	fast RAM operand	3.3	4.3
Groups 2 & 3	main RAM, indirect	4.8	8.2
Group 4; 50 & 51		1.5 to 3.0	2.5 to 4.1
52 to 57	fast RAM operand	3.3	4.3
52 to 57	main RAM operand	4.3	8.2
60 & 61		(2.8 + 0.5*N*)	(4.0 + 0.5*N*)
62		(3.2 + 0.9*N*)	(4.4 + 0.9*N*)
63		33 to 45	
64 to 67		18 to 24	
70, 73, 77		4 to 6	
76		12 to 15	
74		5 to 8	