Ferranti Atlas 1 & 2 – Instruction Set & Times

Version 1: 1 November 2003

Contents

1. The Instruction Set
2. Instruction Times
3. References

1. The Instruction Set

An Atlas instruction has the following format:

Field:	Function	Index register 1	Index register 2	Address
Bits:	10	7	7	24
Atlas Notation:	F	Ba	Bm	S

By convention, the contents of registers or store locations are referred to by the corresponding lower-case letters, e.g. ba (or just b), bm and s.

S can be taken as a literal operand in some instructions, in which case it is known as 'n'. When used as an address, the most significant bit is ignored and the next 20 bits specify a word address in main store. Depending on the function F, the next bit can specify a half-word and the remaining two bits a 6-bit character within the half-word. Within any program, main store addresses begin at zero and go up to some programmer-defined limit in 512-word units. At run-time they are mapped on to actual storage blocks assigned to the program by the Supervisor by the page address registers.

F consists of a single binary digit followed by three octal digits. For all basic instructions the binary digit is zero (and is normally omitted in the written form); a one in this position signifies an *extracode* – see below. The basic instructions fall into three categories depending on the first octal digit. If this digit is 1, the function operates on the B registers; if 2, it is a test instruction; and if 3 it acts on the accumulator.

A copy of the original Atlas Summary of Basic Instructions appears at the end of this section.

The main floating-point accumulator A holds a double-length floating-point number. The 8-bit octal exponent is called Ay, and the mantissa Ax. The most significant part (i.e. a standard Atlas floating-point number) is called Am and the least significant Al. There are 45 different accumulator instructions, including different types of arithmetic, transfers, shifts, etc. and 4 accumulator tests.

There are 128 index or B registers called B0 to B127. B120 to B127 are special, and B0 is always zero. The rest are general purpose 24-bit registers with their own arithmetic unit, although some of these are used by library subroutines or extracodes. There are 33 B-register instructions including (fixed-point) addition, subtraction, logical operations, shifts, etc., plus 18 test instructions on B registers. The B registers are used in different ways, depending on the instruction type. In accumulator instructions, the contents of both Ba and Bm, treated as being the same format as S, are added S. Thus the actual address used is S+ba+bm. In most B-register instructions, the Ba position is used as an operand, and only Bm is used to modify S (there are two exceptions to this – F=164 and 165 - in which Bm is also used as an operand so no modification is done). In extracodes, Ba and Bm are treated in special ways.

There is a special test B register Bt, whose value may be set by a number of B-register instructions, and then tested by test instructions.

In the Summary of Basic Instructions, the following additional conventions are used:

- the prime symbol ‘ is used to signify the result of obeying the instruction. Thus for instruction 102, b’=b-s means that the new value of the register specified in the ba position will be its old value minus the contents of S(+bm, if the Bm field is not zero)
- & represents the Boolean operation AND, v represents OR and ! NON-EQUIVALENCE in the B instructions in column 6, but not-equals elsewhere
- the 0.4 used in the test instructions is an octal half and represents a modification of an address in a B-register by a half-word
- M containing m is the mantissa of Am and L (containing l) the mantissa of Al
- in the instructions marked as circular shifts, ½ represents a one-bit shift down, while 64 represents a 6-bit shift up (to permit character manipulation)
- in some Accumulator operations (footnote Q) floating point numbers are standardised (see the Q footnote). This means that the exponent is set to the value giving the greatest possible accuracy in the mantissa. However, non-standardising instructions are also available
- Normal rounding on Atlas (footnote R) is done by ORing a 1 into the least significant bit. This gives an unbiased result. Conventional rounding (R+) is also provided

Note that there are no specific jump instructions. This is because the program counter is B127 so setting this to an instruction location is equivalent to a jump to that location.

The basic order code is extended by the provision of *extracodes* which have a 1 in the first bit of F. These are routines held in the fixed store which are obeyed automatically when such a function is encountered. They carry out many of the operations that would otherwise use a subroutine library – such as double precision operations, logarithms, square roots, trigonometric functions - or extend the basic instruction set to provide, for example, extra fixed point arithmetic or accumulator operations using n as a floating-point literal number. An important group of extracodes deal with input, output and magnetic tape transfers.

There are 512 possible extracodes, which are arranged as follows:

1000-1077	Magnetic tape routines, and input and output routines
1100-1177	Organisational routines
1200-1277	Test instructions 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, sqrt, sin, cos, tan, etc.) and miscellaneous A-type accumulator operations.

Of these, 1000-1477 are normally singly modified, as with basic B-register instructions, and the rest are doubly modified, as with Accumulator instructions. Not all the possible functions are used; there are around 165 manipulative extracodes, plus the input, output and magnetic tape handling functions. Since the fixed-store jump table positions for unused functions may be utilised for other purposes, including such functions in a program may produce unforeseen results. For a full definition of extracodes see references [1], [44] or [58].

As examples:

- the function 1250 unpacks a 6-bit character from store and places it in Ba
- 1304 divides ba by bm, leaving the result in Ba and the remainder in B97
- 1067 prints ba
- 1670 loads the sine of a stored floating-point number into the accumulator
- 1001 searches for 512-word section n of magnetic tape deck number Ba.

2. Instruction Times

Instruction times are notoriously difficult to pin down on both Atlas 1 and Atlas 2, because instruction overlapping, multiple arithmetic units, multiple storage access mechanisms, and slave stores mean that identical instructions may take vastly different times in different circumstances.

On Atlas 1 there are several distinct phases in obeying an instruction, in general:

- Locate instruction in core store
- Decode instruction
- Find operand in core store
- Obey instruction.

The overlapping mechanisms mean that while one instruction is being obeyed, and another decoded, a third maybe being fetched from store – see Figure 1. In addition, because of the separate A and B arithmetic units, indexing may be carried out at the same time as floating-point operations, and in parallel with peripheral transfers.

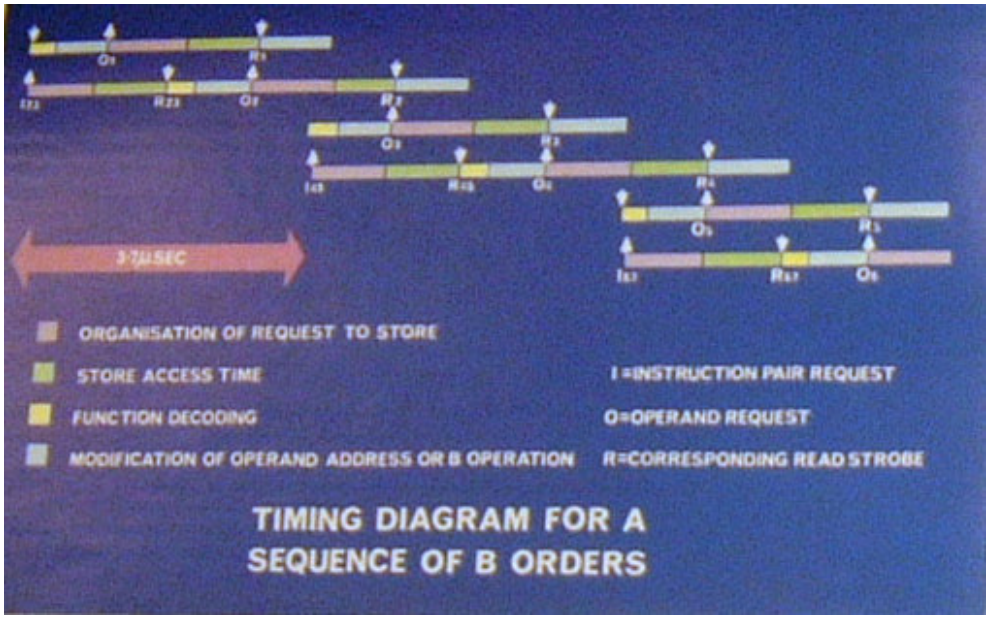

Figure 1: Instruction Overlap

[Author's note: Figures are poor quality and temporary.]

The following average times for floating-point instructions were obtained from the Manchester University Atlas 1 on 7th December 1962:

- Addition, unmodified 1.61③s
- Addition, singly modified 1.94③s
- Addition, doubly modified 2.61③s

- Multiplication, all cases 4.97③s
- Division, all cases, minimum 10.66③s
- Division, all cases, maximum 29.80③s

A similar overlapping scheme is adopted on Atlas 2. Here such methods are even more important, since the machine can use a relatively slow core store with an access time of 5③s. To overcome this delay, the store is divided into four interleaved stacks, each with its own access system. The first contains addresses 0, 4, 8, 16…, the second 1, 5. 9…, and so on. This means that access for consecutive instructions can never clash, and there is a reasonable chance that operand access will also be overlapped.

A further mechanism used on Atlas 2 is the slave store. This contains 32 fast-access registers. As an instruction is fetched from core store to be obeyed, a copy of it, together with the most significant 12 bits of its address, is placed in the register which has the same number as the least significant 5 bits of the address. If any of these instructions are obeyed again, they are automatically fetched from the slave, rather than core storage, taking about only 0.2③s to access. Thus, program loops of up to 32 instructions will be obeyed extremely fast, and some benefit will be given to loops of up to 64 instructions. The operations of the slave store, which are outside the programmer's knowledge and control, are illustrated in Figure 2.

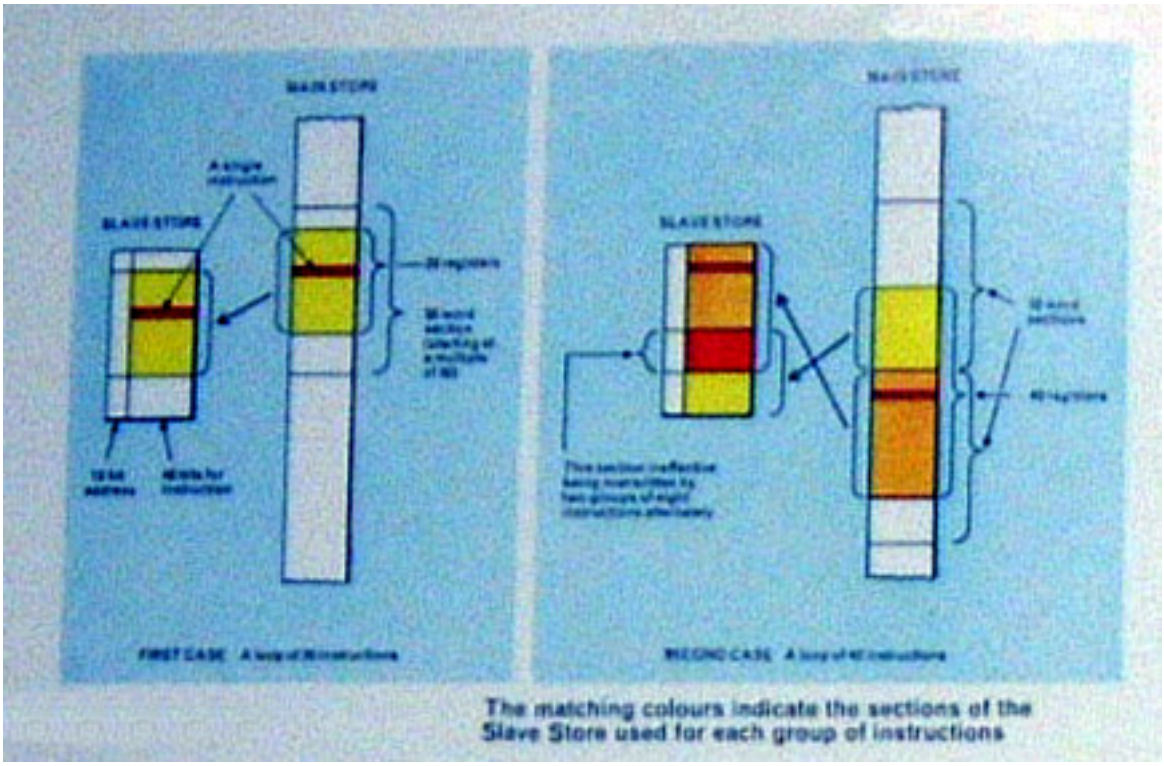

Figure 2: Atlas 2 Slave Store Operation

The Atlas 2 fast operand registers provide a similar improvement for operand access, but under the programmer's control They are addressed as the first eight addresses of the program storage and can be used for data for which fast access is needed, e.g. partial results. Access time is very small and the possibility of stack clashes further reduced.

The following examples of timing are given for an Atlas 2 with a 2.5③s or 5③s cycle core store. The first and third columns assume use of both the slave store and operand registers, the second and fourth assume neither. Times are in ③s.

Single instructions	**2.5③s**		**5③s**	
Repeated indexing operation or floating point addition	2.0	2.8	2.0	4.6
Repeated floating point multiplication	5.0	5.0	5.0	5.5
Repeated floating point division		10.7 to 29.8		
Simple loops				
Scalar product of two n-vectors	11.9n	15.0n	14.3n	25.9n
Sum polynomial to n terms	7.4n	9.2n	8.3n	13.7n

3. References

The following references in section **X5** are particularly relevant to this section:

1, 6, 14, 20, 32, 44, 51, 52.

FERRANTI LTD

SUMMARY OF ATLAS BASIC INSTRUCTIONS

	0	1	2	3	4	5	6	7	
10	b' = s - b	b' = s	b' = b - s	b' = -s	b' = b + s	b' = (64b) + s	b' = b ≠ s	b' = b & s	10
11	s' = s - b	s' = -b	s' = b - s	s' = b	s' = b + s		s' = b ≠ s	s' = b & s	11
12	b' = n - b	b' = n	b' = b - n	b' = -n	b' = b + n	b' = (64b) + n	b' = b ≠ n	b' = b & n	12
13									13
14				b' = (½b) - s				b' = b ∨ s	14
15	bt' = s - b		bt' = b - s						15
16				b' = (½b) - n	b'=b+(bm&n)	b' = bm & n		b' = b ∨ n	16
17	bt' = n - b		bt' = b - n						17
20	If bm≠0, b'=n & bm'=bm+0.4	If bm≠0, b'=n & bm'=bm+1.0	If bm≠0, b'=n & bm'=bm-0.4	If bm≠0, b'=n & bm'=bm-1.0					20
21	If bm odd, b'=n	If bm even, b'=n			If bm=0, b'=n	If bm≠0, b'=n	If bm≥0, b'=n	If bm<0, b'=n	21
22	If bt≠0, b'=n & bm'=bm+0.4	If bt≠0, b'=n & bm'=bm+1.0	If bt≠0, b'=n & bm'=bm-0.4	If bt≠0, b'=n & bm'=bm-1.0	If bt=0, b'=n	If bt≠0, b'=n	If bt≥0, b'=n	If bt<0, b'=n	22
23					If ax=0, b'=n	If ax≠0, b'=n	If ax≥0, b'=n	If ax<0, b'=n	23
30	a'=am+s QE	a'=am-s QE	a'=-am+s QE						30
31	a'=am+s N1 QE	a'=am-s N1 QE			am'=s N1	am'=-s N1 AO			31
32	am'=am+s QRE	am'=am-s QRE	am'=-am+s QRE		a'=s Q	a'=-s QE			32
33	a'=am+s AO	a'=am-s AO	a'=-am+s AO		a'=s	a'=-s AO			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			l'=m.s AO E	l'=-m.s E	am'=a R+ AO	a'=a.1.8^{-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			a'=am.s AO E	a'=-am.s 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

○ Circle denotes circular shift

N1 L not cleared

Q Accumulator standardised

R Accumulator rounded

R+ Rounded by adding

E Exponent overflow may occur

DO Division overflow may occur

AO Accumulator overflow may occur

TM

24th July, 1962