Instruction sets and times for the Ferranti Poseidon, Hermes, Apollo and Argus computers.

1. Ferranti Poseidon

Notation.

The operation denoted as: *a' = b + c* places the sum of the contents of addresses b and c in the address a. (Note that b or c may be a).

Various registers were referred to by their Greek symbols. For clarity, these registers are referred to by the full name of the Greek letter (e.g. lambda) in the description below. The symbol # is used to represent the 'not equivalent to' operation (ie exclusive OR).

The 24-bit Poseidon instruction had nine function (ie op code) bits, split up as three octal digits, U, V and W. The remaining 15 bits were generally used as three 5-bit address fields A, B and C.

Group 0 (Arithmetic and Logical Operations).

	U VW A B C	***Effect of Order***	
1.	0 0 0 a b c	a' = b + c	
2.	0 0 1	a' = b - c	
3.	0 0 2	a' = b # c	
4.	0 0 3	a' = b # NOTc	
5.	0 0 4	a' = b + c	) with collate effective if
6.	0 0 5	a' = b - c	) a, b or c = 3 (for D.S.)
7.	0 0 6	a' = b & c	
8.	0 0 7	a' = b & NOTc	
9.	0 1 W a b c	Instruction as above, but simultaneous count in N; N '= N + 1 if Q10 clear; N' = N + 2 if Q10 set.	
10.	0 2 W a b c	Instruction as above for V=0, supplemented by count in N as for V=1 with the extra facility lambda' = lambda + 1 with a jump to the address in register L if lambda negative before the count.	
11.	0 3 0 a b c	a' = b + c if condition register (C.R.) records last adder output as >= 0; else a' = b - c if C.R. records < 0.	
12.	0 3 2 a b c	a' = b + c if C.R. records >= 0; else Null order if C.R. records < 0.	
13.	0 3 4 a b c	a' = b - c if C.R. records >= 0; else	

		a' = b + c if C.R. records < 0.
14.	0 3 6 a b c	a’ = b - c if C.R. records >= 0; else Null order if C.R. records < 0.
15.	0 4 W a b c	Return to outer mode after completion of order as defined for V==0.
16.	0 5 W a b c	If b or c = 2, the fast program store will be read at address N. If b or c is not= 2, the order will be followed by a fast jump to address N.
17.	0 6 W a b c	Order basically as for V=O, but supplemented by the count and test lambda facility as in V=2 (but no count in N).
18.	0 7 W 3 b c	Digits 0-4 of Highway A sent to tape punch.
19.	0 7 W a 3 c	Tape reader character read onto digits 0-4 of Highway B.
20.	0 7 W a b 3	Handswitches read onto Highway C.

The rest of the Poseidon’s Instruction Set is now given in outline only.

Group 1 (Miscellaneous Arithmetic).
This includes transfers between various internal registers, a modulus function, exchange, etc.

Group 2 (Shifts).
This includes left and right, arithmetic and logical, single and double length shifts (the latter using register mu as the least significant part). Multiply and divide are also included and functions to normalise the contents of a register and to count in lambda the number of shifts to do so. Various supplementary functions may be added and there were a number of ‘shift-and-test’ functions.

Group 3 (Stops, etc.).
This includes a Stop function, an Optional Stop (depending on a Handswitch setting) and sending one hoot pulse. There were also functions to Wait for or send a synchronising signal to an external device called PROCTOR. This was very configuration-dependant and dealt with connecting one Poseidon to others and to other external devices.

Group 4 (Conditional Jumps).
In some jump orders, the jump address is made up of the 10-bit field bc. In others, the jump is to the address in N. Tests are generally on the result of an operation on a and/or other registers, or on the state of the Condition Register (C.R.) which records whether the result of a previous arithmetic operation performed was = 0, not = 0, > 0, < 0, >= 0, <= 0 and if the action caused Overflow (OVR set).

Group 5 (Stat. Operations)
The state of devices in the outside world or switching instructions to those devices,

were held in staticisers (ie flip-flops) in PROCTOR (see above). Poseidon used Group 5 operations to Clear, Set and Test these staticisers (if meaningful) or its own Q Stats.

Group 6 (Specified Bit Operations).
These could Clear, Set and Test individual bits of Register 1, the Data Store contents at address in N, alpha or kappa.

Group 7 (13-bit Operations and Unconditional Jumps).
To accommodate data stores up to 16384, 13-bit address registers are required. Accordingly N, L and lambda are 13 bits long. Operations are provided to handle the13-bit field made up of bc and the three lower bits of a, designated p. Jump functions can also change between Inner and Outer Mode, with jump address from N, L or p.

Timing
The computer clock speed was 1 MHz and most simple Inner Order Code instructions (together with any subsidiary functions required) were completed in 1 microsecond (or 1 'beat'). The fast program store ran on a 2 microsecond cycle and the core store on a 6 microsecond cycle, so instructions referring to these took correspondingly longer. Multiplication took 24 beats and division 27 beats.

2. Ferranti Hermes.
The layout of an instruction was as follows (see also section F6X2):

3	2	1	3	5	5	5
F	I	D	S	A	B	C

As an illustration, the first half of the Instruction Set (Order Code) for F = 0 is shown in the Table below. In all jump instructions (F = 0 to 5 inclusive), ← 1 indicates that the jump-to address is obtained from VA, whereas ← 2 indicates that the jump is relative A places where A is a signed integer.

F = 0

	I = 0		**I = 1**	
S	***D = 0***	***D = 1***	***D = 0***	***D = 1***
0	← 1, VB + VC > 0	← 1, VB + C > 0	VA = VB + VC	VA = VB + C
1	← 1, VB + VC < 0	← 1, VB + C < 0	N1 = VA, VN1 = VB + VC	V[N1 + A] = VB + VC
2	← 2, VB + VC > 0	← 2, VB + C > 0	N1 = VB, VA = VN1 + VC	VA = V[N1 = B] + VC
3	← 2, VB + VC < 0	← 2, VB + C < 0	N1 = VC, VA = VB + VN1	VA = VB + V[N1 = C]
4	← 1, VB = VB + VC > 0	← 1, VB = VB + C > 0	WAIT A B C	
5	← 1, VB = VB + VC < 0	← 1, VB = VB + C < 0		
6	← 2, VB = VB + VC > 0	← 2, VB = VB + C > 0	VA = VB + VC, N1 + 1	VA = VB + C, N1 + 1
7	← 2, VB = VB + VC < 0	← 2, VB = VB + C < 0	VA = VB + VC, N1 - 1	VA = VB + C, N1 - 1

The other 15 tables of the Hermes Instruction Set are omitted simply because of lack of time on the part of the compiler of this information. Sorry!

Timing.
The following Table gives an idea of typical Hermes instruction times.

Operation	***With 3 µsec. core store***	***With 6 µsec. core store***
Simple orders (eg ADD, SUBTRACT, AND, etc.)	12 µsec.	24 µsec.
Modified orders	15 µsec.	30 µsec.
Multiplication	24 – 30 µsec.	36 – 44 µsec.
Division	39 – 45 µsec.	51 – 63 µsec.
Jumps	12 – 15 µsec.	24 – 30 µsec.

Ferranti Apollo
No data currently available.

Ferranti Argus Series.
The Argus Instruction Set was derived from that of the Ferranti Pegasus computer, with the following format for the 24-bit word (the Pegasus instruction was 19 bits:

13	3	6	2
N	X	F	M
Address or constant	Acc	Function	mod

The two M bits identify a modifier (index) register. The function (ie op code) bits F were conveniently represented by two octal digits. The lists below give the instructions corresponding to the octal combinations 00 to 77. In the lists, x denotes the contents of Accumulator X, n denotes the contents of short address N when with a short accumulator; n denotes the contents of short addresses N-1, N, when with a long accumulator; x', n' denote the contents of X, N after the order has been obeyed.

Accumulator 7 is designated P, composed of P1(X4), and P2(X5). Accumulator 6 is designated Q, composed of Q1 and Q2.

00	x' = n	10	n' = x
01	x' = x + n	11	n' = n + x
02	x' = - n	12	n' = - x
03	x' = x - n	13	n' = n - x
04	x' = n - x	14	n' = x - n
05	x' = x & n	15	n' = n & x
06	x' = x # n	16	n' = n # x
07	--------	17	-------

(Note:- # represents the ‘not equivalent to’ operation, ie *exclusive OR*).

	When X is 0, 1, 2, 3, 4 or 5	***When x is 6 or 7***
20	p′ = n . x (multiplication)	(pq)′ = n . x
21	p′ = n . x (rounded up)	(pq)’ = n . x (rounded up)

22	-------	-------
23	-------	-------
24	Short division	Long division
25	Short division rounded	Long division rounded
26	As 25 but with p2’ zero	As 25 but with q zero
27	-------	-------

30 Select N to Register 20 40 x’ = N
(Single bit inputs)
31 Select & Translate Digitisers to 20 41 x’ = x + N
32 Select Analogue currents to 22 42 x’ = - N
33) 43 x’ = x - N
34) 44 x’ = N - x
35) Other types of input if reqd. 45 x’ = x & N
36) 46 x’ = x # N
37) 47 -------
50 x’ = x shifted arithmetically N places (N may be positive or negative)
51 x’ = x shifted logically N places (Up if N positive, down if negative)
52 p’ = p shifted arithmetically N places if X is 4,
(pq)’ = (pq) shifted arithmetically N places if X is 7
53 Shift p or (pq) logically N places (X is 4 or 7 only)
54 Normalize p. Number of places shifted put into X.
55 Normalize (pq). Number of places shifted put into X.
56 -------
57 -------
60 Jump to N if x = 0 70 Jump to N, store link
61 Jump to N if x no zero 71 Jump to link + 1
62 Jump if x >= 0 72 Jump to link after interruption
63 Jump to N if x < 0 73 Jump if typewriter busy
64 Jump to N if OVR clear & clear it 74 Jump if typewriter not busy
65 Jump to N if OVR set & clear it 75 -------
66 x’ = x - 2, m’ = m + 2) Jump to N 76 -------
67 x’ = x - 1, m’ = m + 1) if x’ not zero 77 Stop.