PROGRAMMING FOR EDSAC 2 WITH MAIN STORE

by

D. W. Barron

Second Edition November 1962

1. INTRODUCTION

The MAIN STORE (so called to distinguish it from the existing FREE and RESERVED stores) is a magnetic-core of 16,384 words. The registers are numbered from 0 to 16383 (counting in ones, unlike the free store). Associated with the main store is a modifier register b, which can count from 0 to 16383. Orders are provided for setting, changing, and storing the content of this modifier, but there are no conditional orders. The main store can be used as a data store, or it can be used to hold sections of program which must be transferred to the free store before they can be obeyed. Transfers between main store and magnetic tape must be made using a buffer in the free store: a Library subroutine is provided to facilitate this. Operation times are the same for the main store as for the free store.

Since the address part of an order cannot exceed 2047, special arrangements are necessary to address registers in the main store. Orders with functions in the range 48 to 95 (inclusive) cannot refer to the main store: all other orders may have a main-store address, but an order referring to a half-register should not normally be used with such an address. (But see Section 5 below.)

Main store addresses are of four types, as follows:

(i) DIRECT. The address in the order (between 0 and 511 inclusive) is the main-store register referred to.

(ii) DIRECT, MODIFIED. The address in the order (between 0 and 511 inclusive) is added to the content of modifier b to give the main-store register.

(iii) INDIRECT. The address in the order (between 0 and 255 inclusive) gives the main-store register containing the address of the register to be referred to.

(iv) INDIRECT MODIFIED. The content of modifier b is added to the address obtained as in (iii) above to give the main-store register.

Thus the first 512 registers are immediately addressable, and of these the first 256 can hold an indirect address. The number in modifier b can be added to either a direct or indirect address. The indirect address register holds the actual address as a 14-bit positive integer, stored in digits 6 to 19. The form in which main-store addresses are written is described in Section 2.3. Examples of the use of the various modes of addressing the main store are given in Section 7.

Orders with functions between 48 and 95 may be r-modified in the usual way: if other orders are r-modified the address must lie in the range -256 to +255 inclusive, otherwise the order is interpreted as a main-store order (see Section 4).

2. THE ASSEMBLY ROUTINE

2.1 The Assembly Routine described in Chapter 4 of "Programming for EDSAC 2" (Third Edition) includes facilities for using the main store. It is assumed that the reader is familiar with the use of the assembly routine for free store programs.

2.2 The following notation is used.

m address of a register in the free store
$\underline{m}$ address of a register in the main store

d obeyed address (i.e. taking account of preceding 2, 3, 4 and 5 orders)

b content of main store modifier register

A(m) the positive integer held in digits 6 to 19 of register m.

2.3 Main store addresses are indicated by the use of special "modifier letters" as follows:

Modifier	Effective address
ff	$\underline{d}$
f*	$\underline{d} + b$
rf	$A(\underline{d})$
r*	$A(\underline{d}) + b$

$0 \leq d < 512$

$0 \leq d < 256$

Modifiers ff, f*, rf, r* correspond to the four modes of addressing the main store described in the Introduction, thus

ff = DIRECT
f* = DIRECT, MODIFIED
rf = INDIRECT
r* = INDIRECT, MODIFIED

Note that * indicates modification, and r indicates indirect addressing.

A modifier which is a priori unreasonable in its context causes a report. In orders referring to the main store, an address in

excess of the permitted maximum causes a report.

If it is required to modify a main store address by s or t, this must be done by a preceding 2s0 or 3s0 order. Great care must be taken to ensure that this modification does not cause the obeyed address d to exceed the allowed limit. Such an error does not lead to a report stop of itself, but causes the wrong operand to be used.

2.4 Parameters 1 and 2 may be set to a main-store address less than 512 by a directive

$$\begin{matrix} p1 = ff\ d \\ \text{or} \quad p2 = ff\ d \end{matrix} \Big) \quad 3 \leq d < 512$$

This enables indirect addresses to be set up, or enables the main store to be used for constants read in by the asterisk facility. A report occurs if p1 or p2 is set to a main-store address outside the permitted range. A report also occurs if an attempt is made to read orders with p1 set to a main-store address, or if a label is attached to an item which is being placed directly in the main store, or if p1 or p2 are included in an address (e.g. 10f -2p2) when set to a main-store value.

Orders can be placed in the main store by the use of the directive

p1 = a/b

The effect of this is to assemble a program as if it were going into the free store starting at register a, but to place it in the main store starting at register b. In this way a large program can conveniently be broken up into blocks. The address a can include (set) parameters; b is a positive integer less than 16384. (In practice it must be less than 15104, or the working of the A.R. will be disturbed.)

In a program broken up in this way, all forward references must be local to the block in which they appear, that is to say, a parameter cannot be used in one block and subsequently set in another. Cross-references between blocks using set parameters are permissible.

3. ORDERS CONTROLLING b

These have functions 105, 108, 112.

105	Load b. b' = A(m),	m in free or main store.
108	Store b. N(m)' = b,	remainder of register cleared to zeros, i.e. N(m)' = 2^{20}b)
		m in free or main store.
112	Set or increment b.	Usually has a main-store address. Its effect is:

112	ff	d	b' = d	$0 \leq d < 512$
112	f*	d	b' = d+b	
112	rf	d	b' = A(d)	$0 \leq d < 256$
112	r*	d	b' = A(d) + b	

Note that if d < 256, 112 rf d is equivalent to 105 ff d.

A 112 order with a free-store address causes b to be set to the last main-store address used.

Operation times for these orders are:

105	27 μsec
108	30 μsec
112	16 μsec.

4. MACHINE REPRESENTATION OF MAIN-STORE ORDERS

The internal representation is as follows

WRITTEN		IN MACHINE	
Modifier	Address	Modifier	Address
ff	d	r	512 + d
f*	d	r	1024 + d
rf	d	r	256 + d
r*	d	r	1536 + d

Since the address in the machine differs from the address on the program tape, great care must be taken when changing addresses within a program by 79 orders.

5. ORDERS REFERRING TO HALF-REGISTERS

(a) Functions 4, 5, 99, 100.

Let r be the location in the free store of the order, and let d be the obeyed address, taking account of preceding 2, 3, 4, 5 orders. Then d, together with the modifier letters, determines the main store address, and the odd or even half is used according as (d + r) is odd or even.

(b) Function 110.

This is treated as a 100 order if it is written with a main store address.

6. ORDERS WHICH DO NOT REFER TO THE STORE

It is meaningless to use these orders (2,3,6,7,26,46,96,107,120,121) with a main store address.

7. PROGRAMMING TECHNIQUES

If we require access to arbitrary addresses we can either set the address in b, and use an order with modifier f*, or we can set the address in one of the first 256 registers and use modifier rf. This latter method is useful if addresses are being constructed in the accumulator. Thus to place the word from register (12,000 + s) in the accumulator we could use the orders

```
46 s 0
32 * n 12000
 6 f 20
19 ff 0
10 rf 0
```

If access is required to sequential registers there are several techniques available. We can use b to sweep through the range, possibly setting a base address (less than 512) in the order. Alternatively, we can use b for the base, and sweep over a range of up to 512 by counting in s, and preceding the main-store order by a 2 order thus:

```
 2 s 0
10 f* 0 F(M)' = F(b+s)
```

Another method is to use an indirect address as a base and count through b; by use of a preceding 2-order we can use s to select different bases.

There are no conditional orders associated with b, therefore it is necessary either to count in s and/or t, or to use the Accumulator. In order to count negatively in b it is necessary to store the decrement in the main store, and to use 112 r* for the subtraction. (Note that for main-store addresses, -m is written and stored as 16384 -m.)

Some of these points are brought out in the examples that follow.

(i) Clear the first 12,288 words of the main store to zero.

```
112 ff   0      b = 0
 71  s  24
 70  t   0      s and t counts
  9  f*  0      N(b) = 0
112  f*  1      b' = b+1
 74 tr  -2
 74 sr  -4      Count words
```

(ii) Copy 100 words into main store, starting address in A(3)

```
 71  s  200     Count words
112  rf   3     b' = A(3)
 10  s   -
 19  f*   0     copy word
112  f*   1     b' = b+1
 74  sr-  3
```

(iii) Matrix operations. If the addresses (in the main store) of the beginnings of successive rows of a matrix are stored in sequence in the main store, starting at 0, and if s = i, t = j, ($i < 256$, $j < 512$) we can obtain a_{ij} by the orders

```
  2 t  0
112 ff 0  b' = j
  2 s  0
 10 r* 0
```

This illustrates the use of 2-orders to modify main-store addresses.

(iv) Chaining. Run through a chain of addresses until a word is found with sign digit 0

```
105 ff   -   b' = head of chain
 10 f*  0
105 f*  0    b' = A(b)
 55 r  -2    Test sign
```

(v) Read numbers into j, j-1, ... until solidus, where j = A(5).

This illustrates the method of counting backwards

```
110 f 1429
 19 ff   4   A(4) = -1
105 ff   5   b' = j
 70 tr   5   jump for solidus
 59 f   10   read and store
 19 f*   0
112 r*   4   b' = b-1
 50 r   -3
```

Note that we cannot decrease b by using the order 112 f* -1.

We could set A(4) = -1 from the program tape as follows:

```
p1 = ff 4
nn 16383
```

APPENDIX

REPORT NUMBERS

254	Letter shift outside title
307	Meaningless character following p or s
359	Number, parameter number, unsetting number or address too large
376	Character meaningless in context
442	pt not paired with pn
470	pn not paired with pt
472	Order following pp
475	Function $\geq$ 512
483	* in order not followed by a constant
506	Illegal modifier combination
516	sr or tr in non-modifier order
552	Illegal character in address
590	Two minus signs in exponent
668	Number too large when exponents taken into account
683	Two minus signs in address
717	Address exceeds permitted limit
728	pp when p1 is odd
750	q / directive with q $\geq$ 200
763) 769)	Magnetic tape errors when reading library subroutine
772	q / directive with p1 odd
774	q / directive with p1 set to main store address
832	* facility used when p2 is set to main store and function cannot have main store address
838	p2 exceeds 512 when set to main store
856	Attempt to use registers 0 or 2046 in free store
869	p1 exceeds 512 when set to main store
876	Short integer too long
882	Attempt to read orders with p1 set to main store
906	Attempt to use parameter 0
912	p1 or p2 in address when set to main store
931	Unset parameter in parameter setting directive
935	Unset parameter in start directive
1070	Too many forward references
1140	Second unsetting number (in = a/b) exceeds 127
1169	Attempt to unset a parameter which has been used as a forward reference but not yet set
1171	Label used when p1 set to main store
1178	Attempt to set p1 or p2 by label
1184	Set parameter used as label
1188	Attempt to set parameter 0
1205	Label attached to parameter setting directive
1311) 1354)	Illegal character in parameter setting directive
1369	Attempt to set p1 or p2 to main store address $\geq$ 512 or < 3
1512	Number with 13 or more digits
1515	Solidus in address
1523	Main store address where not allowed