23 Address pointers, indirection operators

The Electron's memory

The computer's memory consists of 65536 locations (0 to 65535), each containing 1 byte (8 binary digits). Half of the computer's memory can be written to or read from (called RAM); the other half can only be read from (called ROM).

Each location in memory has a unique address (like the address of your house) which is a four digit hexadecimal number. Location 0 in memory is &0000 and location 65535 is &FFFF. The & sign means that the numbers which follow are in hexadecimal. The easiest way to look at the computer's memory is on a memory map. Overleaf is a simplified memory map for the Electron, showing on the right the address of each location.

Looking at the memory map, see how it is divided into the two types, RAM and ROM. All the programming which went into making the machine work is stored in the upper half of the memory, from &8000 to &FFFF. Your BASIC programs are stored, unless the computer is told otherwise, starting at location &0E00. This position where the program starts is assigned to a resident variable called **PAGE**. **PAGE** is an *address pointer*; it tells the computer at which address to start executing a program when you tell it to **RUN**. The location at which the BASIC program finishes is also assigned to a variable, called **TOP**. If you type:

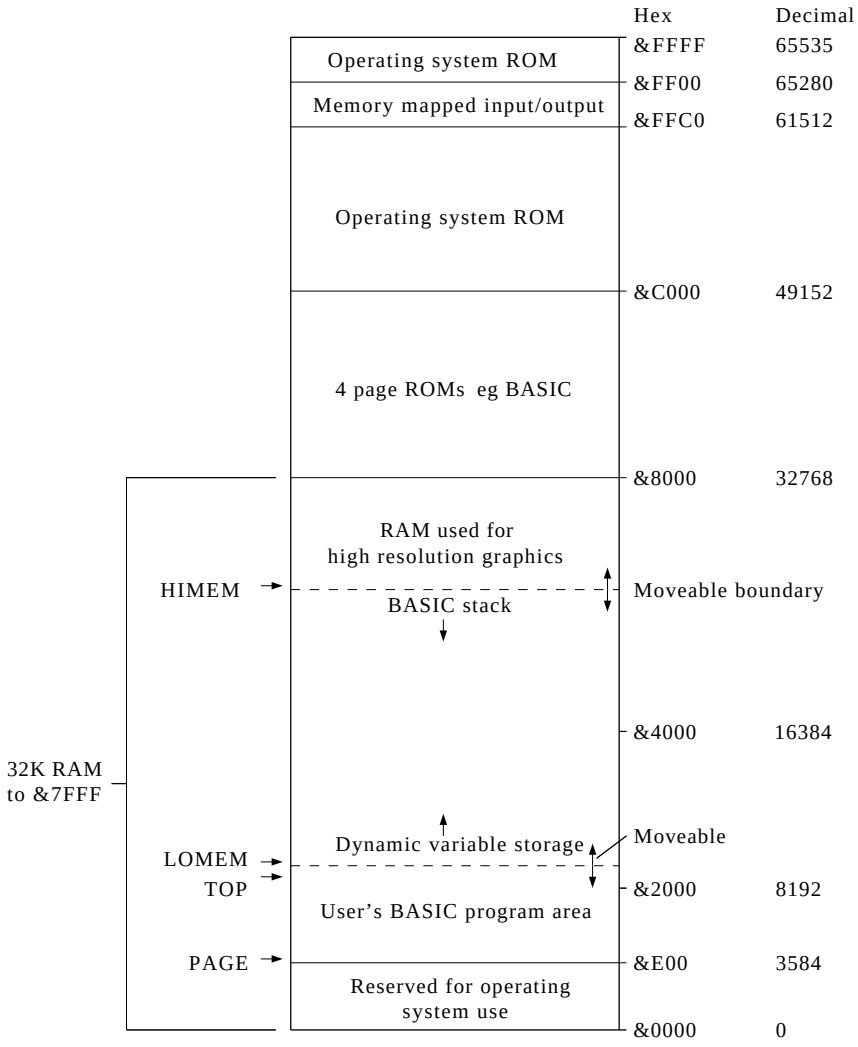
```
PRINT TOP-PAGE RETURN
```

the computer tells you how many bytes of memory your program fills.

Note that future expansions of the Electron, eg a disc filing system, will move **PAGE** up.

The next address pointer is **LOMEM**. This tells the computer where it can store the variables which are used by your program, and is usually the same value as **TOP**.

Memory map



The last address pointer is **HIMEM**. **HIMEM** shows the position of the bottom of the screen memory, so any program or variables must be kept below this value.

Programs are normally loaded at &0E00, but they can be put higher up the memory by altering the value of **PAGE**. For example, if you make **PAGE** = &1000, and then you **LOAD** a program from tape, it will be situated at address &1000. When you do this, **TOP** and **LOMEM** are moved to their new position above the program. Another way in which to re-situate a program is to use ***LOAD**.

```
*LOAD "program name" 1000 RETURN
```

LOAD the program from tape into the memory at location &1000. This instruction does not alter **PAGE**, and if you want to run the program you must make **PAGE** = &1000.

Any section of memory can also be saved by using ***SAVE**.

```
*SAVE "file name" SSSS FFFF EEEE RETURN
```

SSSS is the hex address from which you wish to start saving.

FFFF is the hex address plus 1 at which you wish to fmish.

EEEE is the hex address at which execution should commence.

Indirection operators

Individual memory locations can be accessed from BASIC by using three indirection operators:

Symbol	Purpose	Number of bytes affected
?	Byte indirection operator	1
!	Double-word indirection operator	4
$	String indirection operator	1 to 256

To illustrate this, set a variable to an address in memory, for example:

```
A = &1000 RETURN
```

?A will give the contents of location A, so the contents of location &1000 can be set by typing

```
?A = 100 RETURN
```

(Of course, because a location is a single byte, it cannot be set to a fractional number, or any integer above 255 decimal, which is &FF. If this is done, the least significant byte is stored in the memory location specified.)

To check the contents of &1000, type

```
PRINT ?A RETURN
```

BASIC integer variables, such as age%, are stored in four consecutive bytes of memory, and four bytes can be accessed using !.

```
!A = 70965 RETURN
```

Strings can be placed direct in memory, each character's ASCII code being stored in 1 byte of memory.

```
$A = "STRING"
```

The $ indirection operator appends a carriage return to the end of the string, so the above command would give 6 bytes of ASCII code for the word 'STRING', plus a byte containing &D which is the ASCII code for RETURN.

Notice that the indirection operator is **$A**, and not **A$** which is a BASIC string variable. The string in memory can, however, be assigned to a BASIC variable:

```
name$ = $A RETURN
```

Another way of using **?** is with both a variable and a number.

A?6 gives the contents of location **A+6**, in this case location &1006.

To look at the contents of a group of memory locations, write a small program:

```
10 FOR I = 0 TO 15
20 PRINT "CONTENTS OF ";A+I;" ARE ";A?I
30 NEXT
```

The indirection operators described above are used and explained more thoroughly in the chapter on Assembly Language.