Memory in a computer is a hardware combination of logic elements which is totally independent of the software but which the software must take into account. The structure and organization of the addressable area is far more than simply a matter of getting the appearance right. This is one of the least understood characteristics of computers, and yet it plays an essential role in the operation of the machine, in the layout of the software, and even in adding memory extensions or peripherals, such as input/output modules.

address decoding

why and how an addressable area is organized

The memory of a computer could be compared to a large library: the information, or data if you prefer, is the books and their contents, which we will only mention briefly here. What interests us in this library is its filing system, and especially the way that it is laid out, with its groups, categories, sub-groups and so on. In other words, it is the reference system that we are interested in.

The value of the information

Imagine a catalogue of several billion works dealing with the most varied and different subjects. Our library, of course, contains books on electronics. These are gathered under the reference 'E'. Books about digital electronics are located under the reference of 'ED', whereas those concerning analogue subjects are classified under 'EA'. In data terms we would call the letter 'E' the most significant bit of the references 'ED' and 'EA', and 'D' and 'A' are less significant bits. This distinction is easily seen as the letter 'E' here signifies all works dealing with electronics in our imaginary library, whereas the letters 'D' and 'A' refer only to a certain number of these books. If we continue to make our references even more detailed, the next character (which is less significant again than the previous two) could, for example, be used to distinguish between works in English and those that are not. So a book filed under 'EDE' is in English and deals with digital electronics, while a book with the reference 'EAF' deals with analogue electronics and is written in French. This last character (*English or not*) is less significant than its predecessor (*digital or analogue*): within the category of 'electronic works', the distinction between 'digital' works and 'analogue' works is more *important* than between works written in English and those written in French.

To finish with this attempt to clarify the idea of the significance (or importance) of information, here is a little example. It has to do with the prices displayed by shopkeepers on their merchandise. They would much rather ask £ 9999.99 than £ 10000.00 for a product. Why is that? The most significant information (the number of thousands of pounds *seems* cheaper between one price and the other, but in fact the difference is insignificant as it only involves a very slight change in the least significant information character.

Subdivision and double addressing

Let us now turn to computer memories. These appear as a stack of compartments (called memory cells), each containing 8 irreducible units in the systems most familiar to us, that is 8-bit microcomputers. These discrete units, the bits, are not separately accessible: they constitute an eight-bit word called a byte, and their logic values make up the data. This word travels to the interior of the system via the data bus, which consists of eight lines numbered D7 . . . DØ, each corresponding to one data bit. The words in the memory are accessed by the processor via an address bus, consisting of 16 lines numbered A15 . . . AØ, along which our compartments are arranged. This organization could be compared to that of the library in the preceding example. In figure 1 we have represented the six least significant address bits (A5 . . . AØ) as corridors with successive branches as it could be imagined in a library. Whether a left or right turn is taken in these corridors, the end is reached little by little. The decision to go 'left or right' in an address line is indicated by its high or low logic level (indicated as '1' or '0'), which are the only two states possible. The more the binary 'weight' of an address bit is increased, the more important the zone covered by it becomes. Because bits 5 and 4 in figure 1 are both '0', a '0' at bit 3 means that the area from ØØ to Ø7 is selected, whereas if bit 3 is '1' the zone from Ø8 to ØF is accessed. If bit 4 then changes to '1' with 5 still being '0', the decision of bit 3 selects between zone 1Ø . . . 17 and 18 . . . 1F. Assume that in a specific application the logic level of bit 3 is not defined while bits 4 and 5 are both '0', then the result is that the zones mentioned before are no longer differentiated. Zone ØØ . . . Ø7 will be confused with zone Ø8 . . . ØF. This is called *double addressing*. Depending on the binary weight of the undefined bit, the range of the doubly addressed zones will be more or less important.

$2^{16} = 65536$

The six most significant address lines are shown in figure 2, which also indicates their contribution to splitting up the addressable area. Quantities indicated by the sign 'K' are *always multiples of 1024* (not 1000), which is the number of memory cells ac-

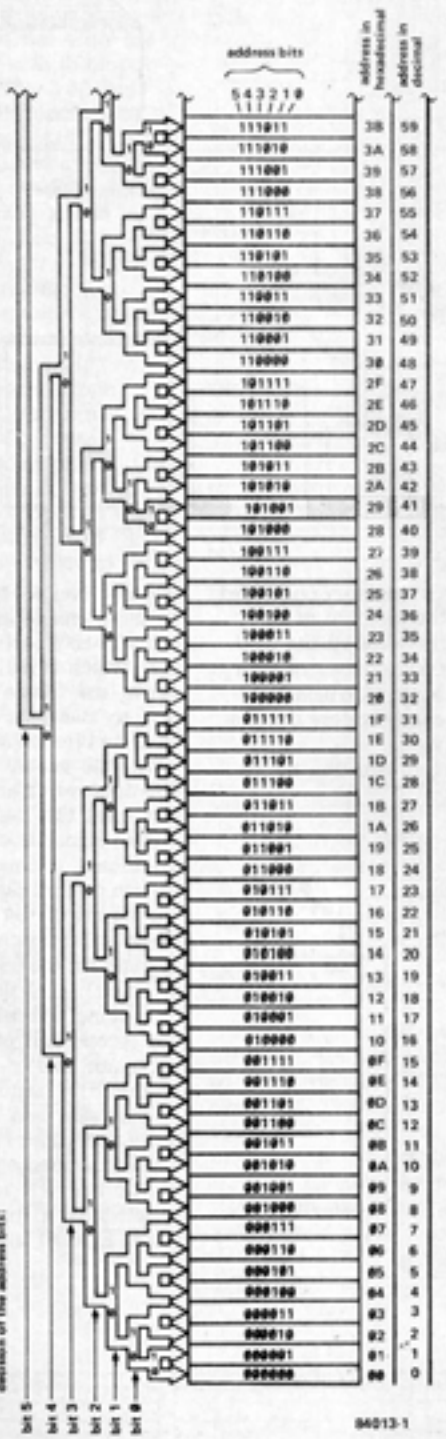


Figure 1. This binary 'tree' of the six least significant bits of an address shows how the decision of a bit (high or low logic level) determines the decoding of a zone whose size depends on the binary 'weight' of the bit.

Table 1.

ADRESSES DEC.	ADRESSES HEX	A15	A14	A13	A12
0 ↓ 15	0000 ↓ 000F	0000	0000	0000	0000 ↓ 1111
16 ↓ 31	0010 ↓ 001F	0000	0000	0001	0000 ↓ 1111
32 ↓ 63	0020 ↓ 003F	0000	0000	0010 ↓ 0011	0000 ↓ 1111
64 ↓ 127	0040 ↓ 007F	0000	0000	0100 ↓ 0111	0000 ↓ 1111
128 ↓ 255	0080 ↓ 00FF	0000	0000	1000 ↓ 1111	0000 ↓ 1111
256 ↓ 511	0100 ↓ 01FF	0000	0001	0000 ↓ 1111	0000 ↓ 1111
512 ↓ 1023	0200 ↓ 03FF	0000	0010 ↓ 0011	0000 ↓ 1111	0000 ↓ 1111
1K ↔ 1024 ↓ 2047	0400 ↓ 07FF	0000	0100 ↓ 0111	0000 ↓ 1111	0000 ↓ 1111
2K ↔ 2048 ↓ 4095	0800 ↓ 0FFF	0000	1000 ↓ 1111	0000 ↓ 1111	0000 ↓ 1111
4K ↔ 4096 ↓ 8191	1000 ↓ 1FFF	0001	0000 ↓ 1111	0000 ↓ 1111	0000 ↓ 1111
8K ↔ 8192 ↓ 16383	2000 ↓ 3FFF	0010 ↓ 0011	0000 ↓ 1111	0000 ↓ 1111	0000 ↓ 1111
16K ↔ 16384 ↓ 32767	4000 ↓ 7FFF	0100 ↓ 0111	0000 ↓ 1111	0000 ↓ 1111	0000 ↓ 1111
32K ↔ 32768 ↓ 65535	8000 ↓ FFFF	1000 ↓ 1111	0000 ↓ 1111	0000 ↓ 1111	0000 ↓ 1111
64K ↔					

Table 1. Using 16 address lines 65536 words can be addressed. This table shows how the decision of each bit affects which address is to be decoded.

cessible with the first ten address lines (A9 . . . AØ; 2^{10} = 1024). Consequently, when talking about memory, *the sign 'K' designates 1024 bytes* and not 1024 bits. Depending on whether address line A15 is at a high or low logic level, one of the two 32768-word halves of the total memory addressable with 16 lines (2^{16} = 65536) is selected. Within each of these blocks, line A14 differentiates between two blocks of 16384 words . . . and so on until line A10 which allows two blocks of 1024 words to be selected within a block of 2048 words decoded by A11. As we mentioned before, if the logic level of one of the address lines is undefined two normally distinct blocks are confused. So if the logic level of A15 is not specified, address Ø and address 32768 are mixed up. The same applies for address 1 and address 32769, and so on. Don't forget that for addressing, no matter what the base (binary, decimal or hexadecimal), the count always starts from Ø.

This leads us to table 1, which shows the 16 address lines, their 65536 possible combinations and the corresponding addresses. Despite the apparent linearity of the progression of this table, the weight of the address lines increases from right to left, and in line with this increase the range of the zones covered by the decision of an address bit becomes more important. This is shown at the extreme left of the table where the ranges of the zones decoded are indicated.

Generating enable signals

So far we have considered the problem of addressing purely as a matter of topography. Looking at the integrated circuits that we must manipulate, we see that the most common ones do not have 16 address lines but a lesser number, proportional to their capacity. As can be deduced from figure 2, a chip containing 4 K (such as a 2732 EPROM) must have 12 address lines (A11 . . . AØ). Addressing each of the 4096 words is achieved by means of an internal address decoder incorporated in the IC. In the same way an IC containing 2 K of memory (for example the still common 6116 RAM) will have 11 address lines (A10 . . . AØ) which will enable the internal decoder to distinguish between the 2048 memory cells. What is called address decoding is not, strictly speaking, this internal

2

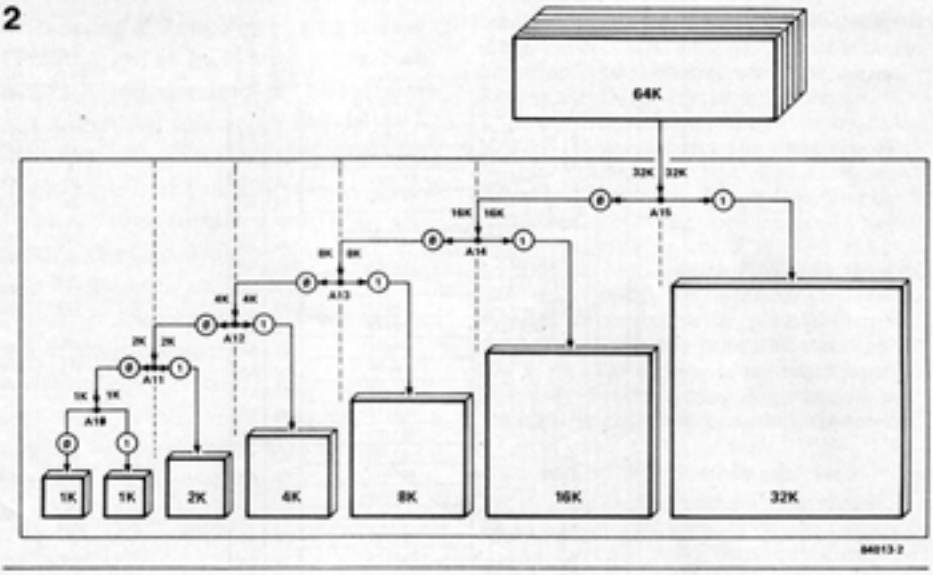


Figure 2. The levels of the most significant bits determine how the addressable area is broken up into blocks that fit inside one another. So, line A15 distinguishes two bloks of 32 K inside each of which A14 can select between two 16 K blocks, and so on.

decoding in the block of memory contained in an IC, but rather the location of this block in the area addressable by the CPU. For our examples we will concentrate on the 6502 and Z 80, both of which have 16 address lines and can therefore decode up to 64 K of memory.
Every memory IC has, in addition to the address lines we have just mentioned, one or more enable inputs. These have to be brought to a certain logic level (generally low, which is indicated by a negation bar above the 'name' of the corresponding pin) to make the chip active. This means that the internal addressing only takes place when the enable signal is present, and the data words are not placed on the data bus until this condition is fulfilled. This enable signal is obtained using the most significant address lines, combined with certain control signals that are essential for the timing of the operations (see figure 3). These control signals are different for each system; for the 6502 they are:

- clock signal Φ2 which only permits reading and writing operations during the second half of each clock cycle of the processor, and
- the R/$\overline{W}$ signal which distinguishes between read operations (Read) and write operations ($\overline{\text{Write}}$).

The corresponding signals in the Z 80 are:

- $\overline{WE}$ and $\overline{RE}$ to distinguish between writing ($\overline{\text{Write Enable}}$) and reading ($\overline{\text{Read Enable}}$), and

3

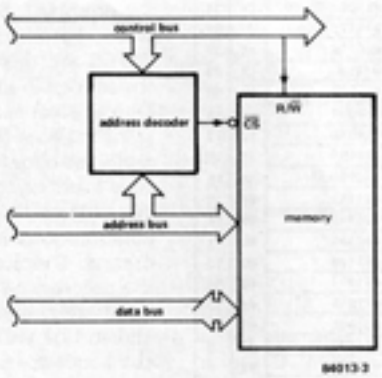


Figure 3. The data and address buses are not all that is needed for addressing the memory; a certain number of control signals are also essential to ensure the correct timing of the read and write operations.

4a

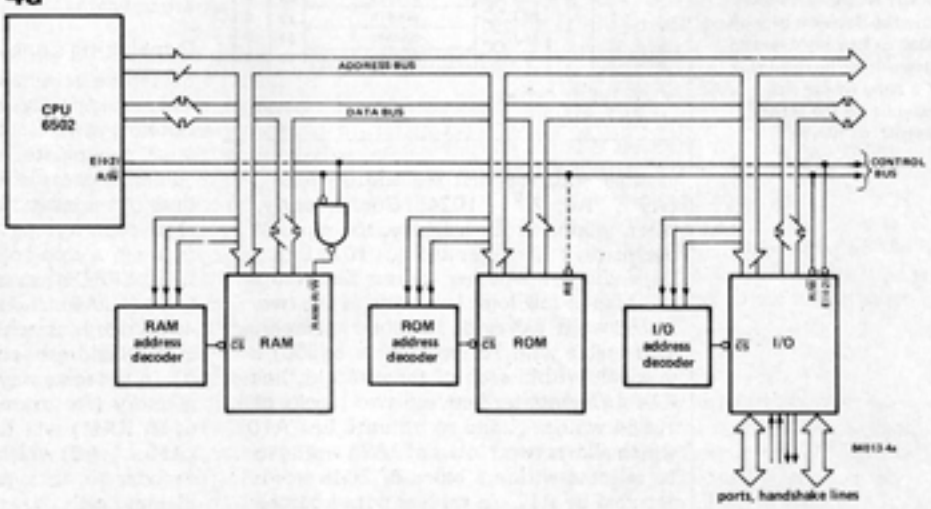


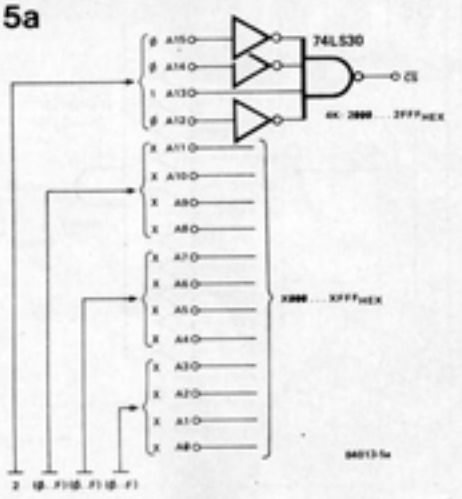
Figure 4a. The 6502 has no specific instructions or signals to distinguish the memory from the input/output modules. The control signals needed to enable the operations are clock Φ2 and the read/$\overline{\text{write}}$ (R/$\overline{W}$) signal.

■ MREQ and IOREQ to distinguish between operations carried out with the memory and those dealing with the input/output module for which the Z 80 has specific instructions. The differences between the two processors are clarified by figures 4a and 4b. The validation signals, obtained from the most significant address signals and the control signals, are all referred to here as CS (Chip Select). Just for the sake of making things easier to follow, we will assume that they are always active at the low logic level. However, depending on the system and the manufacturer, it is possible to find some signals, including the enable signal, which are active high. Before getting on to the logic combinations which will allow these enable signals to be generated it is no harm to emphasize the importance of the hexadecimal base. We have sixteen address lines grouped as 4 x 4 lines. There is a hexadecimal figure (Ø . . . F; 0 . . . 15 in decimal) corresponding to each group of four lines. In address 4A2F, for example, the 4 corresponds to the binary word for lines A15, A14, A13 and A12 (Ø1ØØ), the A corresponds to the binary word on lines A11, A10, A9 and A8 (1Ø1Ø), the 2 to the word on lines A7, A6, A5 and A4 (ØØ1Ø) and the F to that on A3, A2, A1 and AØ (1111). This simple conversion allows the configuration of the 16 address lines, corresponding to an address given in hexadecimal, to be easily found.

Fixed logic combinations

Now we will start looking at the address decoding proper, achieved by means of more or less complex logic combinations. Imagine a memory circuit to be enabled between addresses 2ØØØ and 2FFF. Lines A11 . . . AØ decode 4098 memory cells between XØØØ and XFFF. Combining the A15 . . . A12 lines as shown in figure 5a provides a CS signal active (at logic zero) *only* when the configuration of the lines is 'ØØ1Ø', that is the number 2. The example of figure 5b shows more precise decoding. The enable signal CS, obtained by combining lines A15 . . . A11 logically, is only active when the configuration of these lines gives the values EØ . . . E7. The other address lines allow each of the 2048 addresses between EØØØ and E7FF to be addressed. The decoding obtained with the combination shown in figure 5c is even more precise: CS is only at logic zero when A3 . . . A15 give the hexadecimal value C1Ø; while the three remaining lines are used for addressing the eight bytes between C1ØØ and C1Ø7.

5a



5b

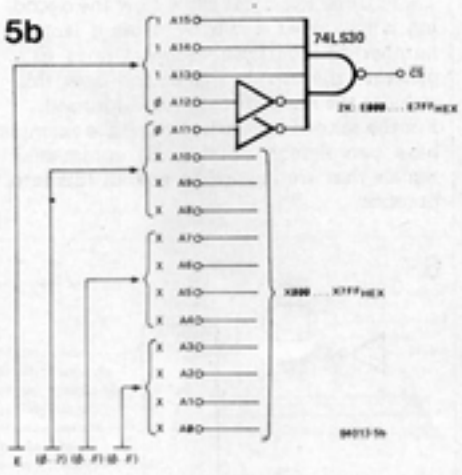


Figure 5a & 5b. Examples of fixed address decoding, of 4 K and 2 K bytes. As the zone addressed becomes smaller, so the number of address signals combined becomes larger.

4b

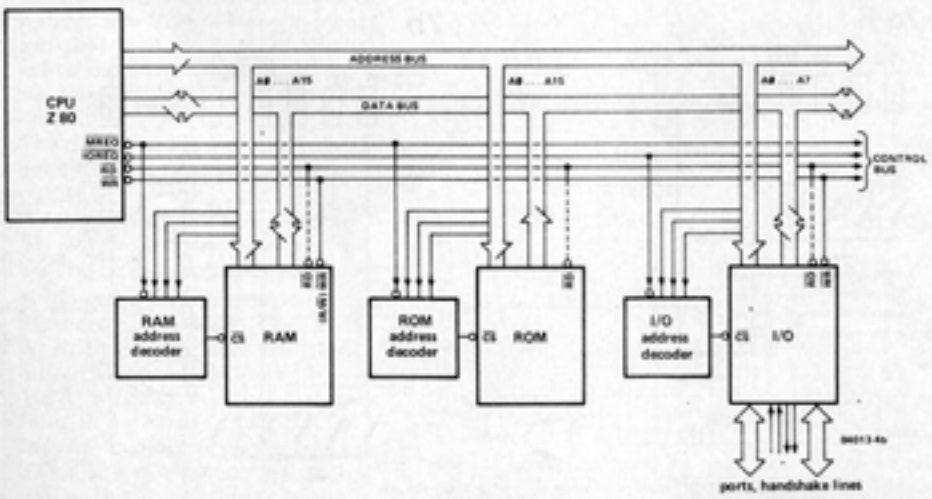


Figure 4b. The internal structure of a Z 80 system is quite similar to that of a 6502, except that it has more (and more specific) control signals. It is beyond the scope of this article to discuss the problems associated with timing these signals.

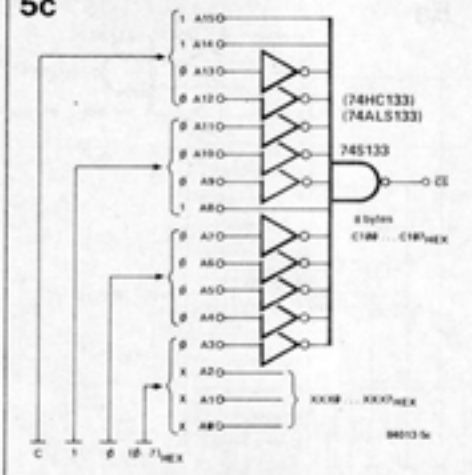


Figure 5c. Another example of fixed address decoding, in this case 8 bytes are decoded.

These three examples show how the decoding is narrowed down by using a larger number of significant address lines to generate the enable signal, and how this reduces the range of the zone addressed. For the sake of simplification, these examples have completely ignored the command signals that are needed to put all this into practice.

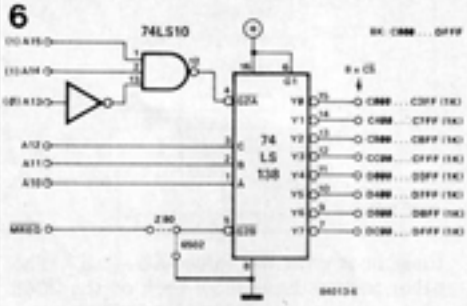


Figure 6. The 74LS138 decoder allows an 8 K block (decoded using A13 . . . A15) to be easily split up into blocks of 1 K, each with its own $\overline{CS}$ signal. The second enable input is treated differently depending on whether it is used with the Z 80 or the 6502.

A multiple address decoding circuit is shown in figure 6. It contains a commonly used decoder IC, the 74LS138, which has three binary data inputs and two enable inputs ($\overline{G2A}$, $\overline{G2B}$). Signal $\overline{G2A}$, which is obtained from a combination of A13 . . . A15, is only active between C000 and DFFF, a block of 8 K. Input $\overline{G2B}$ picks up the $\overline{MREQ}$ signal from a Z 80, or is tied to earth (logic zero) if used with a 6502 processor. The three bit binary word created by combining A10 . . . A12 allows eight successive blocks of 1 K to be decoded. The eight $\overline{CS}$ signals thus produced could be applied to the memory, in conjunction with command signals $\overline{WE}$, $\overline{RD}$ or R/$\overline{W}$.

Variable logic combinations

The decoding examples examined so far have one thing in common, that they are invariable, but variable address decoding is also possible, as illustrated by figure 7. The main part of this diagram is the four bit magnitude comparator, a 74LS85. A binary word A0 . . . A3 is provided by address lines A12 . . . A15. This is compared by the 74LS85 with the binary word supplied by four switches connected to earth and four polarizing resistors to the high logic level. When binary word A0 . . . A3 is the same as binary word B0 . . . B3 pin 3 (A = B) goes logic high. The output of this pin is then inverted and becomes the $\overline{CS}$ signal for a 4 K memory block (X000 . . . XFFF, where X is the hexadecimal value corresponding to binary word B0 . . . B3).

The same sort of programmable address decoding could be achieved using EXNOR gates, as shown in figure 7b. The open collector outputs of the 74LS266 are all logic high only when the two inputs of each gate are at the same logic level. Each gate compares one bit of the address word formed by A12 . . . A15 with the corresponding bit of the binary word programmed using the switches and polarizing resistors. This procedure has the advantage that it adds flexibility to the address decoding. Furthermore, as the dotted lines of figure 7b suggest, it is quite easy to narrow the programmable decoding by increasing the number of significant address lines used, and thus reducing the range of the block enabled by the $\overline{CS}$ signal.

With that we will finish this article on address decoding, and, while we realize that there is much that has not been said about the subject, we hope that at least some light has been thrown on the address bus and how it works. ■

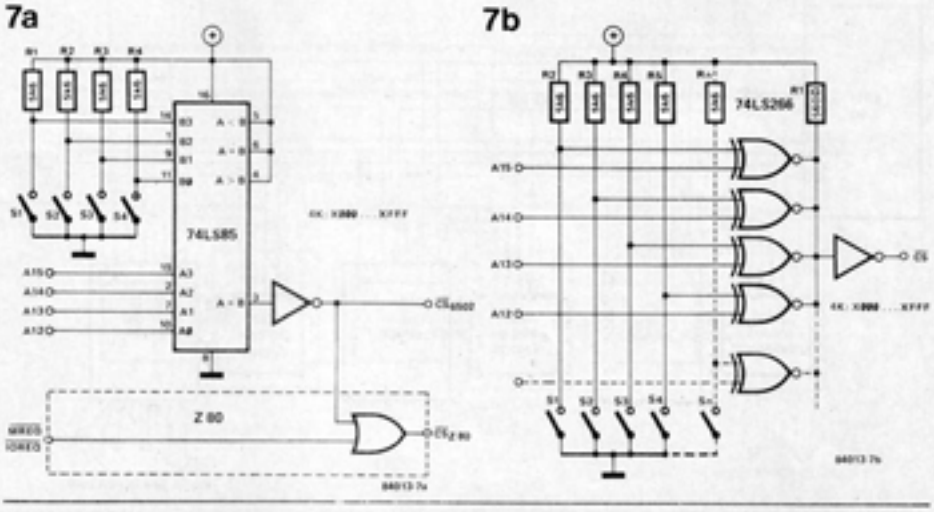


Figure 7. In certain applications it is desirable to have programmable, or at least variable, addressing. This is achieved using a magnitude comparator that determines when the binary word formed by lines A12 . . . A15 is the same as the word formed by the user with the four switches. An alternative is to use EXNOR gates, as shown in b. The outputs of the 74LS266 are all high only when the two inputs of each gate are at the same logic level.