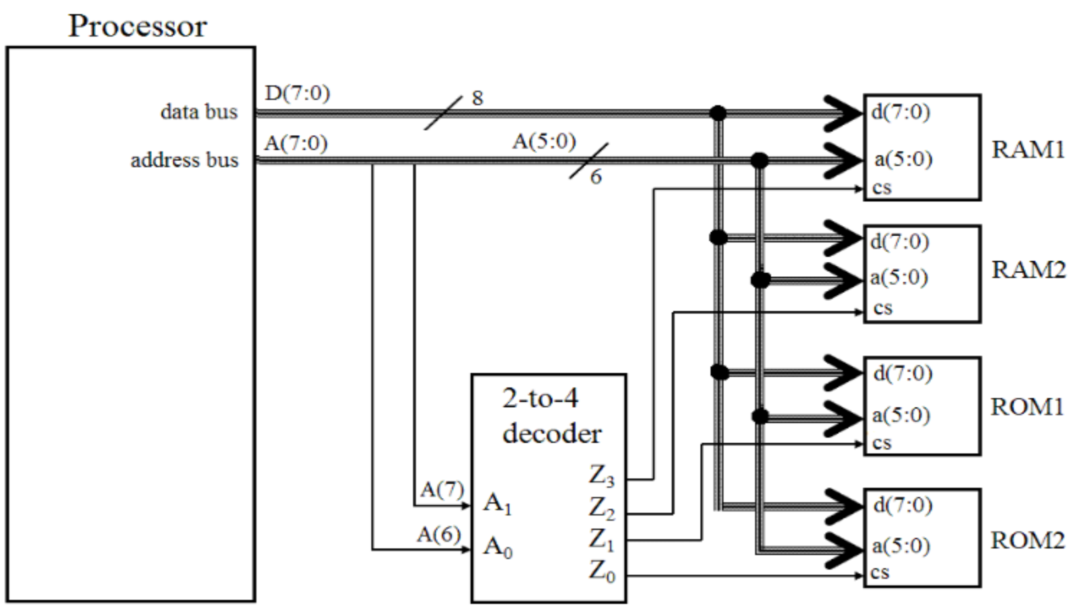
EEE225 Solution Sheet 4

1. Decode the two most significant bits of the address to enable the required peripheral. The remaining six address bits are fed to each peripheral.

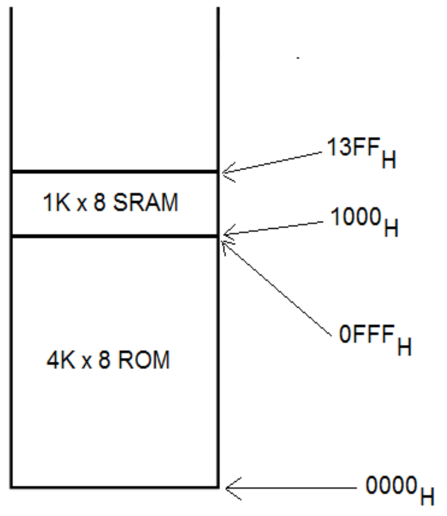


2. Isolated I/O or as it is sometimes called Port Mapped I/O separates I/O access from memory access. This is useful for CPUs with limited addressing capability because it leaves the full address space for memory. The CPU would set its MEM/$\overline{\text{IO}}$ control line to indicate that isolated I/O is being used ('0' in this case). It would then use dedicated INPUT and OUTPUT instructions to communicate with the device. (IN and OUT instructions for Intel devices)

   Memory mapped I/O uses the same address bus to address both memory and I/O devices. There is a single address space for I/O devices, RAM and ROM. The CPU would first set its MEM/$\overline{\text{IO}}$ control line to indicate that memory mapped I/O is now being used ('1' in this case). The CPU now uses the same instructions to access memory or the device. The problem with this scheme is that I/O devices are generally slower than main memory and this will slow down the memory access if the address and data bus are shared.

   See lecture notes for diagrams.

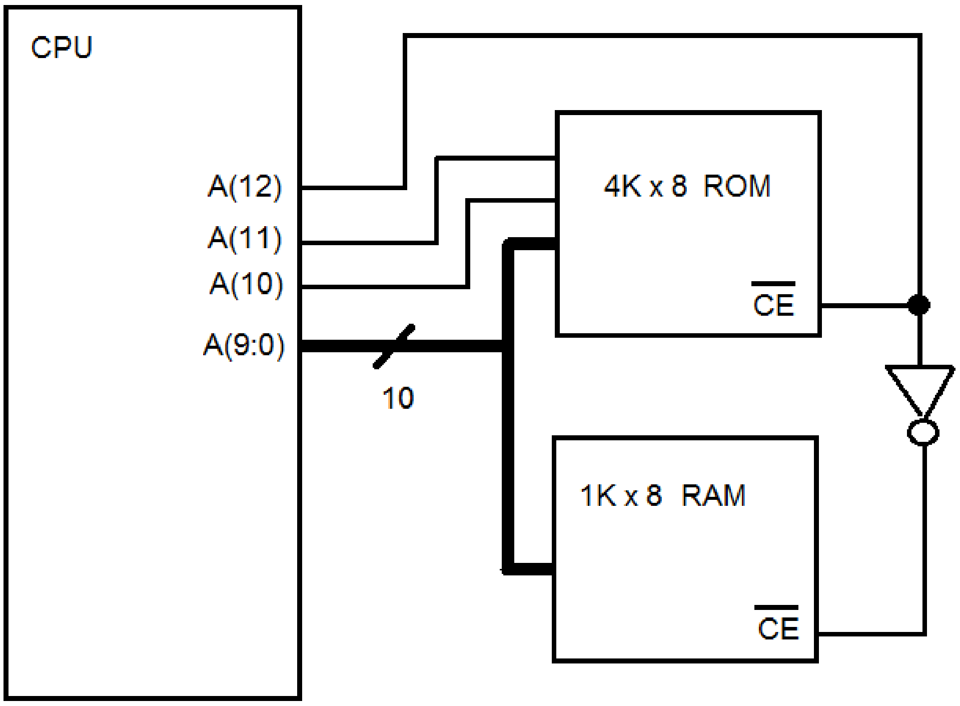
3. The memory map for the proposed system would be:



The ROM would occupy locations 0000_H to $0FFF_H$ inclusive and the RAM would occupy locations 1000_H to $13FF_H$ inclusive. Note that the first address in the memory space has index 0000_H.
The design of the decoding circuitry can be approached by observing that the bottom three (hexadecimal) digits of the address reference the memory cells *within* each chip whereas the fourth digit differentiates between chips. That is, for the ROM chip the fourth digit is always one. Thus we can use (decode) the fourth hexadecimal digit to give the necessary enable signals for the memory devices. In binary terms, each hexadecimal digit is made up from a group of four binary digits; thus the three bottom hex digits of the address are made up from the bottom twelve binary digits of the address line (A_0 through A_{11} – bearing in mind that by convention, we start numbering from zero). These lines can be fed directly to the two chips except that the 1K RAM chip has only ten address lines ($2^{10} = 1024 = 1K$), not twelve. Given that we only have 1024 locations in the 1K chip means that we can ignore the top two address lines – they have no influence on the 1K chip.

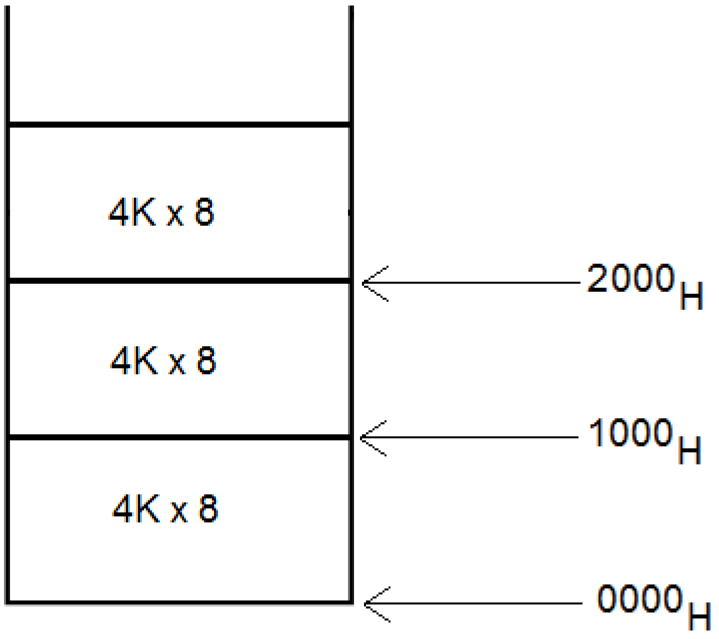
To obtain the chip enable signals ($\overline{CE}$), we need to decode the fourth hex digit of the address… but instead of decoding all four binary bits ($A_{15} – A_{12}$) which make up the fourth hex digit, we need only bother with $A_{12.}$ Address bits $A_{15} – A_{13}$ will always be zero; when we are addressing the ROM chip, $A_{12} = 0$ and for the RAM chip, $A_{12} = 1$. Thus $\overline{CE}$ for the ROM can be obtained by connecting A_{12} directly to the chip enable pin and for the RAM. CE can be obtained by inverting $A_{12.}$

A possible diagram for the memory interfacing circuit could look something like:



In such a system – indeed most computer systems – the ROM chip would be located at 0000_H because during power-up or reset, most CPUs force the contents of the program counter to 0. Thus placing the (initialisation) program in ROM starting at location 0000_H means that the computer system will start automatically without any human intervention.

4. The memory map for 4K by 8 chips would be:



Here the three chips would be located in the address ranges: 0000_H to $0FFF_H$;

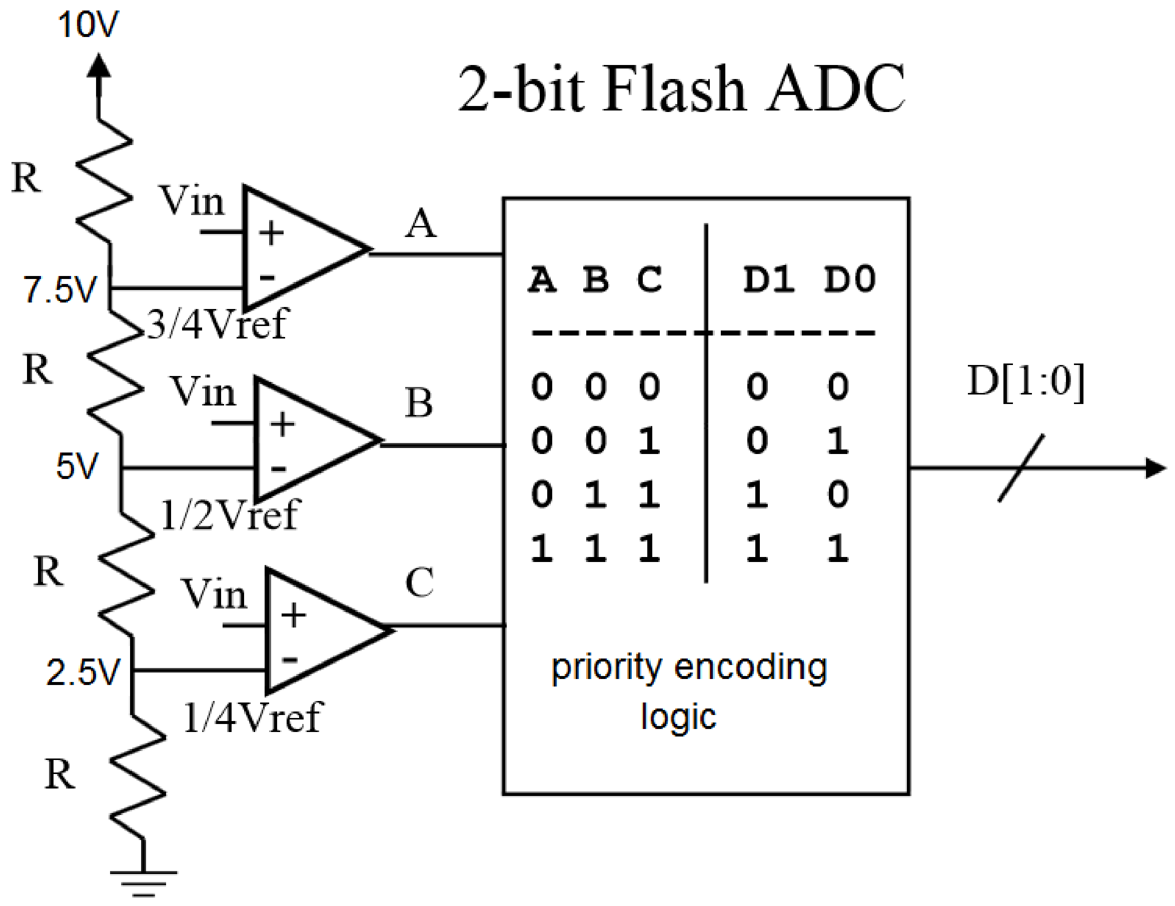
1000_H to $1FFF_H$; 2000_H to $2FFF_H$. As with the previous question, the lowest three hexadecimal digits form the *within* chip address and the fourth, most significant address digit specifies to which chip the *overall* address refers; again we can decode the most significant address bits to form the necessary enable signals. A_{15} and A_{14} are always zero so only bits A_{13} and A_{12} need to be decoded. Consider the chips to have active HIGH enable lines in this example.

$CE_0 = \overline{A_{13}}.\ \overline{A_{12}}$, $CE_1 = \overline{A_{13}}.\ A_{12}$, $CE_2 = A_{13}.\ \overline{A_{12}}$

In both questions. Large fractions of the total memory space of 64K are unused. This does not pose a problem except that the system programmer has to take some care not to generate illegal – strictly unused – program addresses. Trying to save/retrieve data to/from an unused address will probably result in a piece of garbage being processed. If the processor tries to retrieve a non-existent program instruction then the system will almost certainly crash.

5. 1023

6.



Comparator C, 8V is greater than 2.5V so C = 1.Comparator B, 8V is greater than 5V so B = 1, Comparator A, 8V is greater than 7.5V so A = 1. Thus D[1:0] = 11.

7. For 8 bits, each sample can take a value from 0 to 255.

   5/15 = d/255
   d = 85 or binary 01010101

7.50	3.75	1.875	0.938	0.469	.234	0.117	0.059

5V < 7.5V, discard	0	0	0	0	0	0	0	0	STEP1
5V > 3.75V, keep	0	1	0	0	0	0	0	0	STEP2
5V < 5.625V, discard	0	1	0	0	0	0	0	0	STEP3
5V > 4.688V, keep	0	1	0	1	0	0	0	0	STEP4
5V < 5.157V, discard	0	1	0	1	0	0	0	0	STEP5
5V > 4.922V, keep	0	1	0	1	0	1	0	0	STEP6
5V < 5.039V, discard	0	1	0	1	0	1	0	0	STEP7
5V > 4.981V, keep	0	1	0	1	0	1	0	1	STEP8