Homework Assignment 1: Answers

Profs. John McLeod & Arash Reyhani

ECE3375, Winter 2022

Problem: Design a memory map for a system with $64\,\mathrm{KB}$ of memory. Assume RAM occupies locations between `0x0000` and `0x5FFF` (inclusive) and ROM occupies locations between `0xC000` and `0xFFFF` (inclusive). Assume you have only the following chips:

- Two (2) $8\,\mathrm{K}{\times}8$ ROM chips.
- Two (2) $8\,\mathrm{K}{\times}8$ RAM chips.
- Two (2) $4\,\mathrm{K}{\times}8$ RAM chips.

In this homework assignment, you must assign an address range to each chip according to the above specifications, and provide the chip select logic.
There is more than one valid solution to this problem.

Solution: The memory range `0x0000` to `0x5FFF` spans 24 K, which is the exact amount of RAM available. The four available chips can be inserted into this range in any arbitrary order (this is what is meant by the problems stating that "there is more than one valid solution to this problem").

- I will put the two big RAM chips at the bottom of this memory range, and the two small RAM chips at the top.

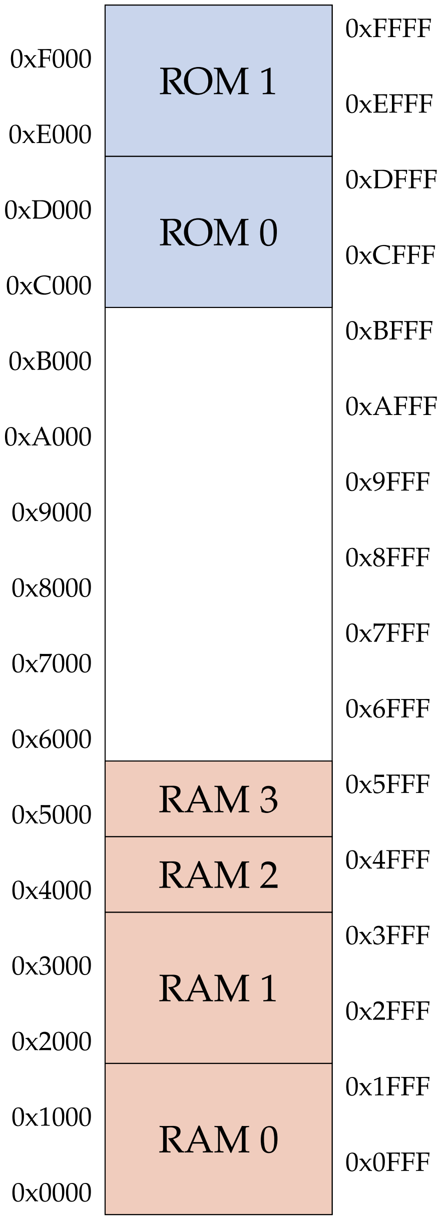
The memory range `0xC000` and `0xFFFF` spans 16 K, which is the exact amount of ROM available.

- Here there is no choice on how to arrange the chips, as there is only two identical chips available.

These chips span the following memory range:

Name	Chip	Start Address	End Address
RAM 0	8 K×8	`0x0000`	`0x1FFF`
RAM 1	8 K×8	`0x2000`	`0x3FFF`
RAM 2	4 K×8	`0x4000`	`0x4FFF`
RAM 3	4 K×8	`0x5000`	`0x5FFF`
ROM 0	8 K×8	`0xC000`	`0xDFFF`
ROM 1	8 K×8	`0xE000`	`0xFFFF`

To design the chip select logic, it should be pretty clear based on the start and end addresses for each chip that only the most significant hex digit will be used for the chip select logic. But I will still write out the full binary address to make sure everything is clear.



Name	Start Address	End Address
RAM 0	`0b0000 0000 0000 0000`	`0b0001 1111 1111 1111`
RAM 1	`0b0010 0000 0000 0000`	`0b0011 1111 1111 1111`
RAM 2	`0b0100 0000 0000 0000`	`0b0100 1111 1111 1111`
RAM 3	`0b0101 0000 0000 0000`	`0b0101 1111 1111 1111`
ROM 0	`0b1100 0000 0000 0000`	`0b1101 1111 1111 1111`
ROM 1	`0b1110 0000 0000 0000`	`0b1111 1111 1111 1111`

The chip select logic is therefore:

$$\begin{aligned}
CSA_0 &= \overline{a}_{15}\overline{a}_{14}\overline{a}_{13} \\
CSA_1 &= \overline{a}_{15}\overline{a}_{14}a_{13} \\
CSA_2 &= \overline{a}_{15}a_{14}\overline{a}_{13}\overline{a}_{12} \\
CSA_3 &= \overline{a}_{15}a_{14}\overline{a}_{13}a_{12} \\
CSO_0 &= a_{15}a_{14}\overline{a}_{13} \\
CSO_0 &= a_{15}a_{14}a_{13},
\end{aligned}$$

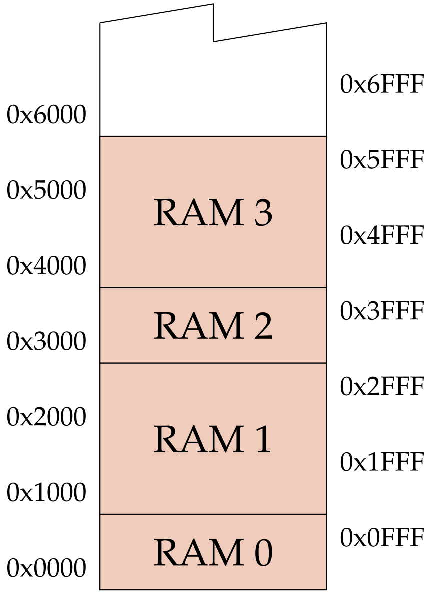
where CSA_i stands for “Chip Select R**A**M i”, and CSO_i stands for “Chip Select R**O**M i”.

Second Solution: It is possible to choose a chip arrangement for RAM that will leave the 8 K×8 chips misaligned in memory. This is still a valid memory map, it is just unnecessarily difficult.

- We can put a small RAM chip at the bottom, then a big one, then another small one, then a big one. (There are a few other ways of having misaligned memory chips, this is just one variant.)

It is not possible to misalign the ROM in this problem, so I won't consider that — there is only one valid solution for the ROM as described above. The RAM chips span the following memory range:

Name	Chip	Start Address	End Address
RAM 0	4 K×8	0x0000	0x0FFF
RAM 1	8 K×8	0x1000	0x2FFF
RAM 2	4 K×8	0x3000	0x3FFF
RAM 3	8 K×8	0x4000	0x5FFF



The chip select logic is still based on only the most significant hex digit. But again, I will still write out the full binary address to make sure everything is clear.

Name	Start Address	End Address
RAM 0	0b0000 0000 0000 0000	0b0000 1111 1111 1111
RAM 1	0b0001 0000 0000 0000	0b0010 1111 1111 1111
RAM 2	0b0011 0000 0000 0000	0b0011 1111 1111 1111
RAM 3	0b0100 0000 0000 0000	0b0101 1111 1111 1111

The tricky part is RAM 1, which requires 13 bits for addressing the internal memory cells, but the remaining three bits (in red) are not consistent from the start to the end. The solution is straightforward:

- Use the "sum of products" form for the chip select: the logical OR of *two* minterms, each minterm being the entire first 4 bits of the start and end address range.
- This is equivalent to considering the $8\,\mathrm{K}{\times}8$ chip as being two $4\,\mathrm{K}{\times}8$ chips in the same package, so instead of having separate chip selects for each, we have the OR-logic chip select for both together ("RAM 1 = RAM 1a + RAM 1b").
- Note that this results in bit 12 being used *twice*: as part of the chip select logic, and also passed into the chip's internal address bus. (Of course this isn't a problem, just extra wiring.)

Name	Start Address	End Address
RAM 1a	`0b0001 0000 0000 0000`	`0b0001 1111 1111 1111`
RAM 1b	`0b0010 0000 0000 0000`	`0b0010 1111 1111 1111`

The chip select logic is therefore:

$$\begin{aligned}
CSA_0 &= \overline{a}_{15}\overline{a}_{14}\overline{a}_{13}\overline{a}_{12} \\
CSA_1 &= \overline{a}_{15}\overline{a}_{14}\overline{a}_{13}a_{12} + \overline{a}_{15}\overline{a}_{14}a_{13}\overline{a}_{12} \\
CSA_2 &= \overline{a}_{15}\overline{a}_{14}a_{13}a_{12} \\
CSA_3 &= \overline{a}_{15}a_{14}\overline{a}_{13}.
\end{aligned}$$

There is almost never a good reason for misaligning a large memory chip like this, so if you are confused by this answer then *just don't do it*.