Memory Mapped I/O

Memory-mapped I/O

In memory-mapped I/O, both memory and I/O devices use the same address space.

We assign some of the memory addresses to I/O devices. The CPU treats I/O devices like computer memory.

The CPU either communicates with computer memory or some I/O devices depending on the address.

Therefore, we reserve a part of the address space for I/O devices, which is not available for computer memory.

Memory-mapped I/O

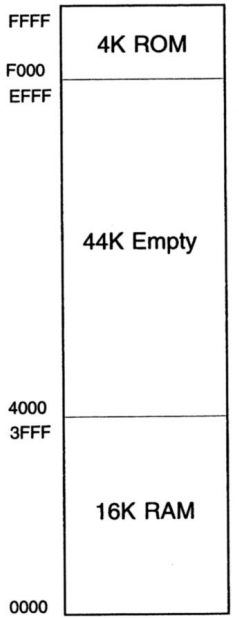
In the case of memory-mapped I/O, all the buses are the same for both memory and I/O devices. Therefore, building a CPU that uses memory-mapped I/O is easier and cheaper.

Additionally, such CPUs consume less power due to reduced complexity.

One advantage of memory-mapped I/O is that we don’t need separate instruction sets for accessing I/O devices. Instructions used for accessing memory can be easily used for accessing I/O devices.

A memory map

- A simple memory map for 16 bit system:
  - A 16 Bit system has 64K “Chip space”
  - 16 bit means $2^{16} = 65536_{10} = 0xFFFF$
  - 0x0000 ~ 0xFFFF



Given the start address and the size of the chip,

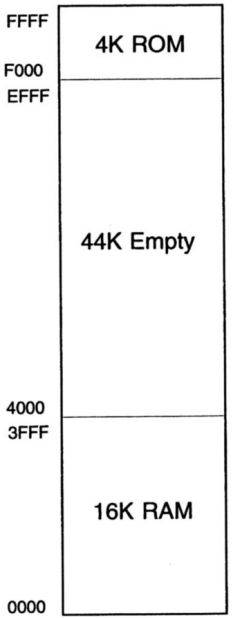
- we need to find the end address.
- Map a Chip Select from the address bus.

16K RAM starts at 0x0000; (K = Thousand = 1024)

$16 \text{ X } 1024 = 2^4 \text{ X } 2^{10} = 2^{14} = 16384_{10} = 0x4000_{16}$

Starting address = 0x0000

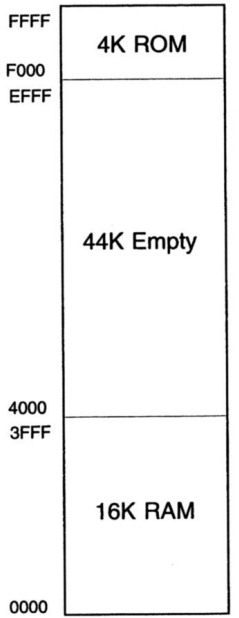
Ending address = 0x4000 - 1 = 0x3FFF



4K ROM starts at 0xEFFF; (K = Thousand = 1024)

$4 \text{ X } 1024 = 2^2 \text{ X } 2^{10} = 2^{12} = 4096_{10} = 0x1000_{16}$

0xEFFF + 0x1000 = end address = 0xFFFF



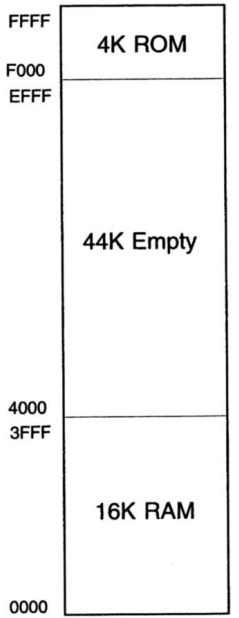
16K RAM start 0x0000

16K RAM end 0x3FFF

4K ROM start 0xEFFF

4K ROM end 0xFFFF

Note: We subtract 1 at the end for Ram and at the start for ROM



With 16 address lines, the chip space is the address bus in memory mapped I/O.

Address Bus = $\{A_{15}A_{14}A_{13}A_{12}A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0\}$

The chip select is the subset of the address bus.

	$A_{15}A_{14}A_{13}A_{12}A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0$
Start 16K	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
End 16K	0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Start 4K	1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1
End 4K	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Recall: From Boolean logic, if the bits change they are needed and if not then the **non-changing** part can be used to identify the chip.

Chip Select 16K = $CS_{16K} = \overline{(A_{15}.A_{14})}$

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
Start 16K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
End 16K	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Start 4K	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
End 4K	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Recall: From Boolean logic, if the bits change they are needed and if not then the **non-changing** part can be used to identify the chip.

Chip Select 4K = $CS_{4K} = (A_{15}A_{14}A_{13})$

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
Start 16K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
End 16K	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Start 4K	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
End 4K	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Example: The Arduino Atmega 2560 has a 16K RAM and an 8 line UART(Universal asynchronous receiver transmitter) with mapped on the address bus using memory mapped I/O. Design a chip select for each chip when,

16K RAM starts at address 0x4000

UART starts at address 0xFC00

Example: The Arduino Atmega 2560 has a 16K RAM and an 8 line UART(Universal asynchronous receiver transmitter) with mapped on the address bus using memory mapped I/O. Design a chip select for each chip when,

16K RAM starts at address 0x4000

UART starts at address 0xFC00

16K Start = 0x4000

16K end = 16 X 1024 = 16384_{10} = $0x4000_{16}$

End address = 0x4000 + 0x4000 -0x0001 (for zero) = 0x7FFF

UART start address = 0xFC00

UART end address = 0xFC00 + 0x0008 - 0x0001 (for zero) = 0xFC07

	$A_{15}A_{14}A_{13}A_{12}A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0$
Start 16K	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
End 16K	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Start UART	1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
End UART	1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1

Chip Select 16K = $CS_{16K} = \overline{(A_{15}.A_{14})}$

Chip Select UART = $CS_{UART} = \left(A_{15}.A_{14}.A_{13}.A_{12}.A_{11}.\overline{A_{10}.A_9.A_8.A_7.A_6.A_5.A_4.A_3}\right)$