Memory

1 cell = 1 bit
8 cells = 8 bits = 1 byte
4 bytes = 1 word (in MIPS)

Read: uses MUX
Write: In order to get to a specific location: uses decoders
Memory is built from small components to bigger components

To access a specific memory location: we access the bigger structures fist (big → small access strategy)

Addressing

* MIPS uses byte Addressing

To represent the number of addressable units in a system:

$2^k \times m$ where k is the Address size, m = word size

2^k distinct addresses → 2^k distinct words

Array of RAM chips

We can increase memory chips or increase the size of a word

If we have the RAM config $1K \times 8 = 2^{10} \times 8$ ($1K = 2^{10} = 1024$ → has 1024 words, each word being 8 bits)

Say we want to increase our memory capacity by 10 → we want to have 10240 words

We'll put 10 chips together → C_0 to C_9

- Say we then want to access word 3079, then we need to have a method for how we can access the correct memory location.
- Since we have 1024-word chips, then we consider that if we want to find word 1023, it will be on the fist chip, C_0, and word 1024 on C_1

↳ $3079 = 1024 \times 3 + 7$ ⇒ our target word is on chip 3 with a word offset of 7

$$3079 = 3 \times 1024 + 7 = \underset{2^{11}}{2 \times 1024} + \underset{2^{10}}{1024} + \underset{111}{7}$$

Absolute Address: 1 1 0 0 0 0 0 0 0 1 1 1

11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	0	1	1	1

If we have 32-bit Addresses, then we'll have to pad the absolute address with a bunch of zeros

The absolute Address is partitioned (from right to left) into fields

the chip #	offset inside the chip

The size of the offset = size of the chip's address
↳ in our case, the originally stated memory config was $1K \times 8$ → the address is 10 bits (since $1K = 2^{10}$)

32 bit Address:

000 ... 0011	0000000111
chip #	offset (10 bits)

We'll implement a decoder to act as our chip selector. Since we have 10 chips, we need to implement a decoder that accepts a 4-bit input to accommodate all possible chip selections 0 - 9

* CS = Chip Select