
What operations can you perform on a computer memory ???

- You can do exactly **2 operations** on the **computer memory**:

  1. **Read** the **memory** (has nothing to do with mindreading).... and
  2. **Write** to **memory**...

  The entity that performs the read/write operation is usually the **CPU**.

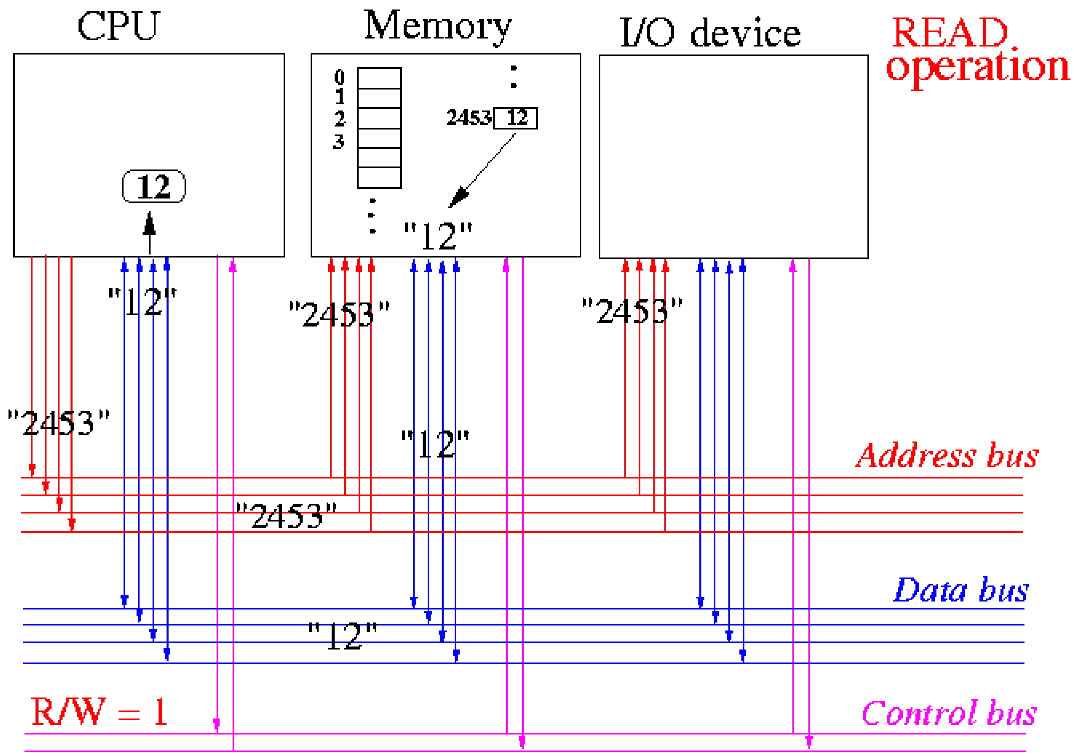
- Some computer jargon:
  - The CPU "reads from memory location X" means
    - **copy** the value (the bits) stored at memory location (address) X into the CPU
  - The CPU "writes to memory location X" means
    - **copy** the value (the bits) stored in the CPU to the memory location (address) X

- CPU can read or write one or **more** bytes of memory (1 byte, 2 bytes, 4 bytes and recently event 8 bytes at a time).

- Since the whole computer memory consists of billions of bytes, the CPU needs to **specify** the location of the bytes that it wants to read from or write to:
  - The **memory location** is specified by an address.

    The **Address Bus** is used to convey the **address information** in a read/write operation
  - The **size** is specified using some signals on the control bus
  - The **Data Bus** is used to **transport** the **value** between the CPU and the memory
  - **In addition**, there is a special signals on the **control bus** that is used to indicate whether the CPU wants to perform a **READ** or a **WRITE** operation.

- The following is an example where the CPU reads some value from memory location 2453:

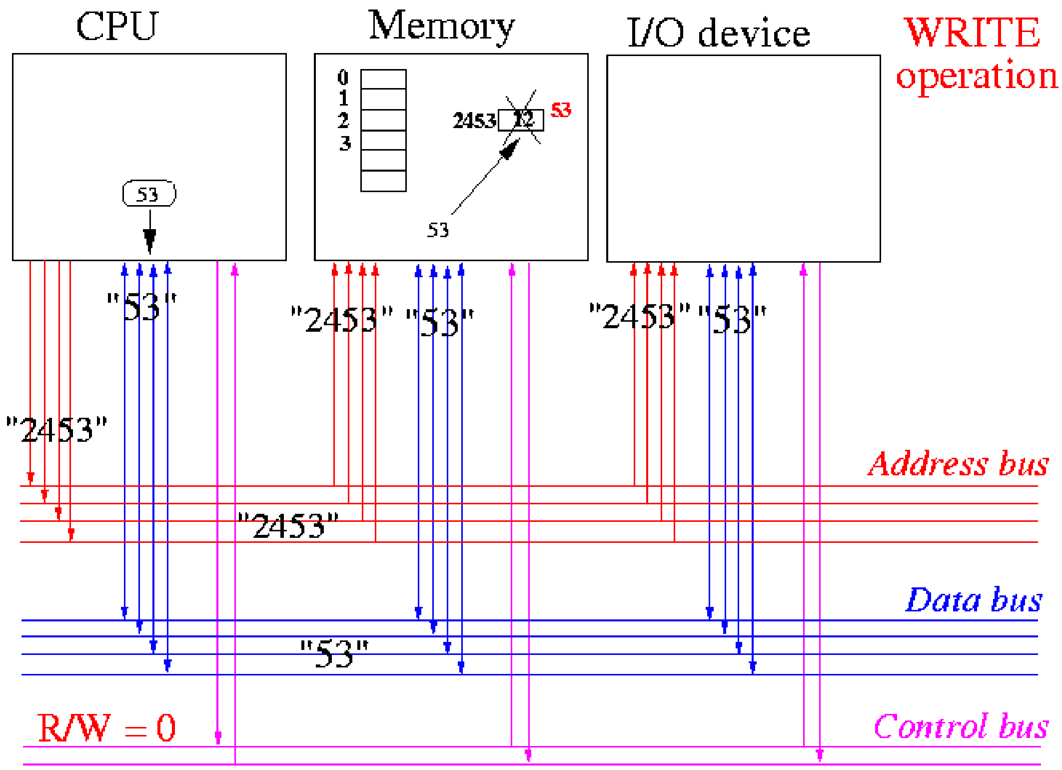


- CPU sends out the address value 2453 on the address bus
- CPU sends out the signal **R/W = 1** on the control bus, which indicates a READ operation
- CPU then waits for the data from memory on the data bus
- The **R/W = 1** signal and the address bus value 2453 will cause the memory to retrieve the value at memory location 2453 to be sent out on the data bus

- NOTE: I have used "decimal" values to illustrate the read operation.
- You will see soon that computers do not use "decimal" values, but "binary" values

---

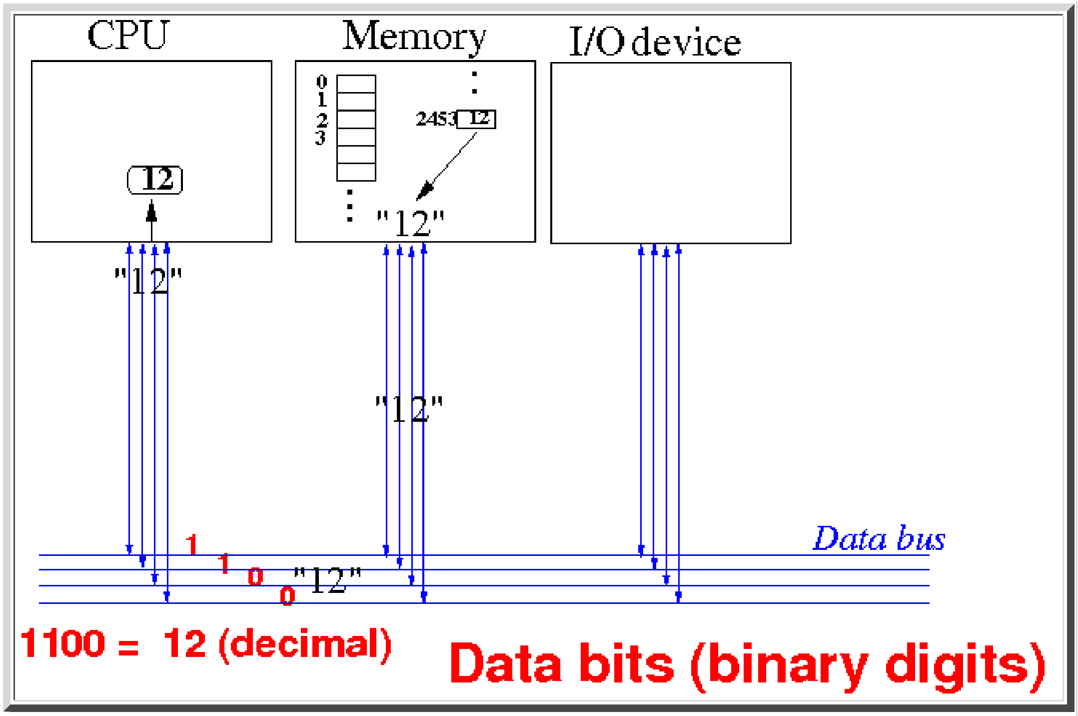
- The following is an example where the CPU write the value 53 to memory location 2453:



- CPU sends out the address value 2453 on the address bus
- CPU also sends out the value 53 on the data bus
- CPU now sends out the signal **R/W = 0**, which indicating a WRITE operation
- The **R/W = 0** signal along with the address bus value 2453 and data bus value 53 will cause the memory to store the value 53 at the location 2453...

- **Effect of the width of the data bus**
  - **Recall** from **above**:
    - The **databus (= wires)** are used to **transfer** the **data bits (binary digits)** between the **CPU** and **memory**

      Example:



- **Fact:**
  - The **computer** will ***always*** use ***every*** **wire** in the **databus** to **transfer data** (i.e., no waste)

- **Effect** of the **width** of the **databus**:
  - A **databus** that consists of **8 bits**, can transfer **1 byte** of **data** **per read/write operation**
  - A **databus** that consists of **16 bits**, can transfer **2 bytes** of **data** **per read/write operation**
  - A **databus** that consists of **32 bits**, can transfer **4 bytes** of **data** **per read/write operation**
  - And so on

- **Conclussion:**
  - The **width** of the **databus** determines the ***amount* of data** transfered per **memory operation**

- **Current trend:**
  - **All PCs** has **at least** **32 bits** databuses (32 bit machine)
  - **Some (high end) PCs** has a **64 bits** databus (a 64 bit machine)

  Note:

  - **Databus width** is ***always*** a **power of 2**

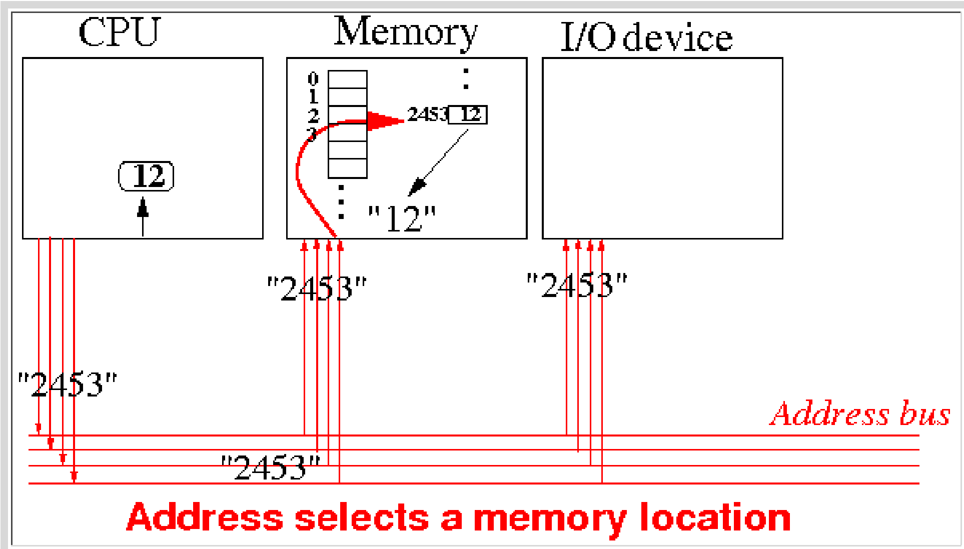
(Because of the technology reason that result in the **alignment requirement**....)

- **Effect of the width of the address bus**

  - **Recall** from **above**:

    - The **address bus (= wires)** are used to **convey** the **address (location)** in **memory**

      Example:



  - **Fact:**

    - The **computer** will ***always*** use ***every*** **wire** in the **address bus** to **identify** a **memory location**

      I.e.:

      - **Leading zeroes** are ***significant*** in an **memory** ***address*** !!!

      Example:

```
0000000000000010    =    2 (decimal)
```

  - **Effect** of the **width** of the **address bus**:

    - A **address bus** that consists of **8 bits**, can **address (= use)** a **2^8 (= 256) byte memory**
    - A **address bus** that consists of **16 bits**, can **address (= use)** a **2^{16} (= 16K) byte memory**
    - A **address bus** that consists of **32 bits**, can **address (= use)** a **2^{32} (= 4G) byte memory**
    - And so on

- **Conclussion:**

  - The **width** of the **address bus** determines the ***size* of the memory** that the **computer** can **use**

- **Current trend:**

  - **All PCs** has **at least** **32 bits** address buses and can use **4 G byte memory**
  - **Some (high end) PCs** has more than **32 bits** address bus and can use **8,16 ...** GBytes memory

* **Postscript**

  - **Later in the course**, I will **show a demo** on **how** to specify:

    - The **memory location** (= **address**)
    - The **number of bytes** of **memory**

  you want to **read/write**