**8086 System Connections and Bus Timing**

Table of Contents

[Microprocessor Operation 3](#_Toc83227086)

[Clock Generation 4](#_Toc83227087)

[Duty Cycle 5](#_Toc83227088)

[Clock States 6](#_Toc83227089)

[Read Bus Cycle 7](#_Toc83227090)

[WRITE Bus Timing 12](#_Toc83227091)

For this chapter, we will consider that the 8086 microprocessor is operating in the **minimum mode**, since that is easier to handle.

## Microprocessor Operation

A microprocessor essentially has three tasks, **fetching** instructions from either the memory or an I/O device, **decoding** the instructions and finally **executing** the instructions. We are considering the writing of output back to external devices to be part of fetching, i.e. the memory or I/O device is the thing being fetched. Amongst these, the decoding and execution steps are internal, since they happen inside the microprocessor, while the fetching step must interact with external devices.

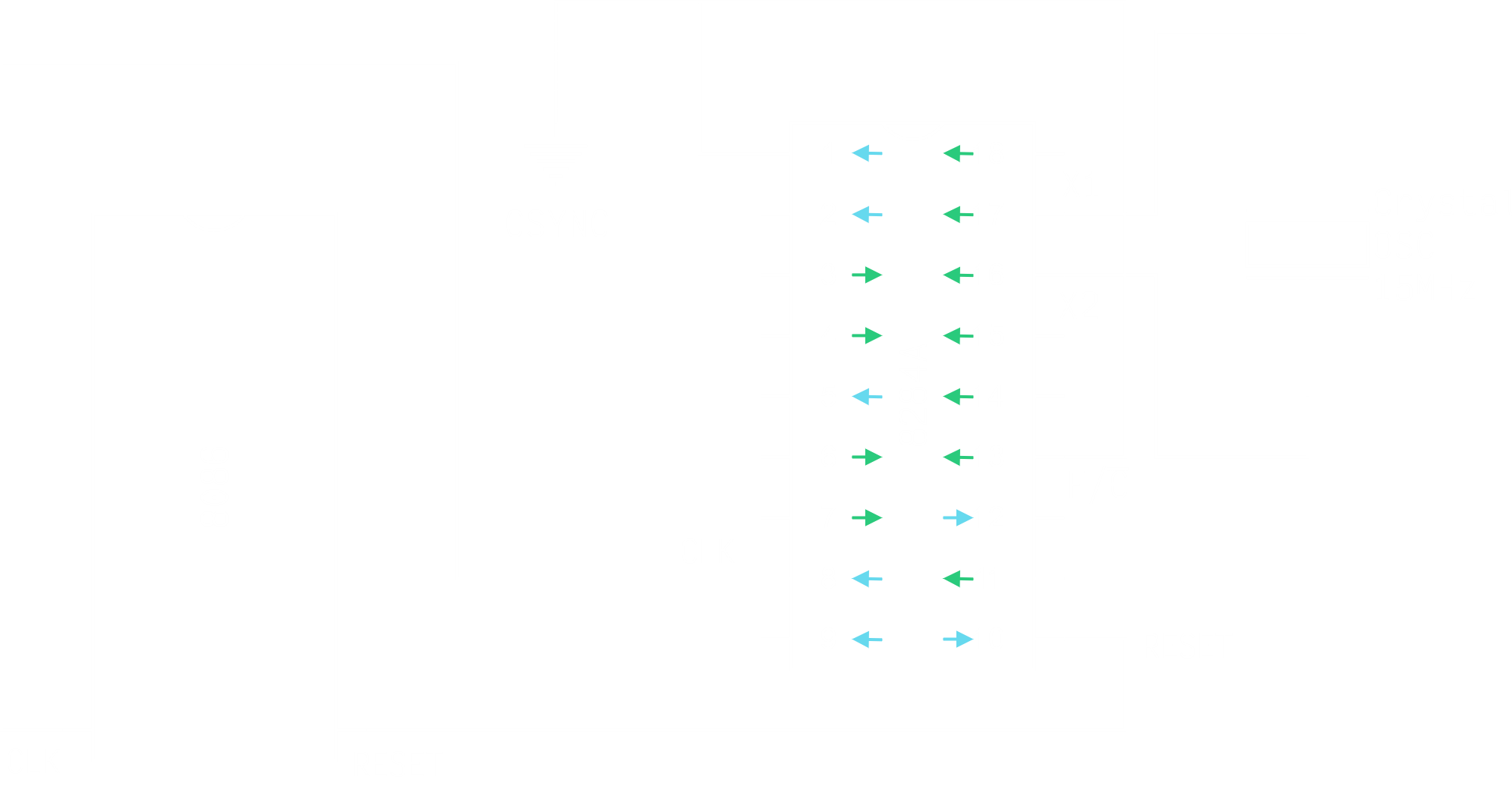
The **time** a microprocessor requires to complete a **fetch-decode-execute** operation of a single instruction is called an **instruction cycle**. Note that an instruction cycle consists of one or more **machine cycles**.

A **basic microprocessor operation**, such as reading or writing a byte from or to memory or an I/O device is called a machine cycle or a bus cycle.

A machine or bus cycle consists of at least **four clock cycles**, called **T states**. One cycle of a clock is called a **state**.

## Clock Generation

The microprocessor itself cannot generate clock pulses. However, it has a pin, **Pin 19**, called the **CLK pin**, through which it receives clock pulses. The actual clock pulses are generated by a different circuit, the **8284A IC**, which generates clock pulses for multiple ICs, one of which is the microprocessor.

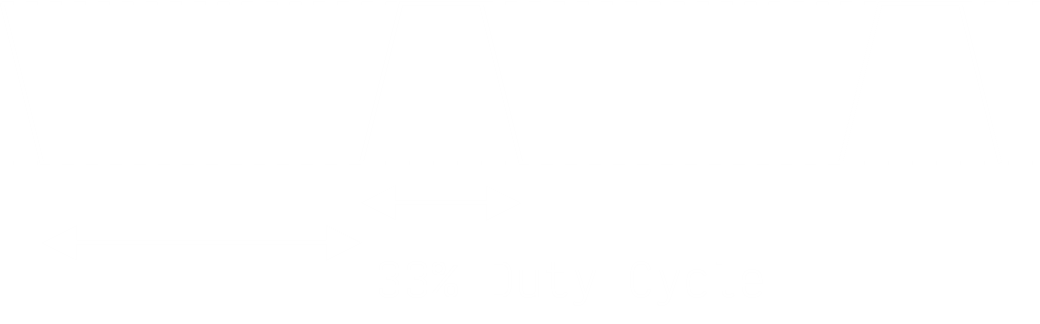


The clock generation IC has multiple pins, some input pins and some output pins. The output pins are marked in **blue** and those are the ones through which clock pulses are sent out. It generates these clock pulses with the help of a **crystal oscillator**, and the maximum clock pulse rate it can generate is **15MHz**.

The 8086 microprocessor itself, however, operates between **5MHz and 10MHz**. At **5MHz**, the **clock pulses** will be of **200ns**, meaning a complete **bus cycle** will be **800ns**. At **10MHz**, clock pulses will be of **100ns** and bus cycles will be of **400ns**. Each read and write operation takes **1 bus cycle** (ignoring the information about even and odd banks, which will give us better results).

### Duty Cycle

The ratio of the duration for which the clock pulse is high to the total duration is called the **duty cycle**.



For example, say the ALE pin is high for 66ms for a 100ms clock cycle. Thus, it has a 66% duty cycle.

In 8086, a 33% duty cycle will cause a high signal to be acknowledged.

## Clock States

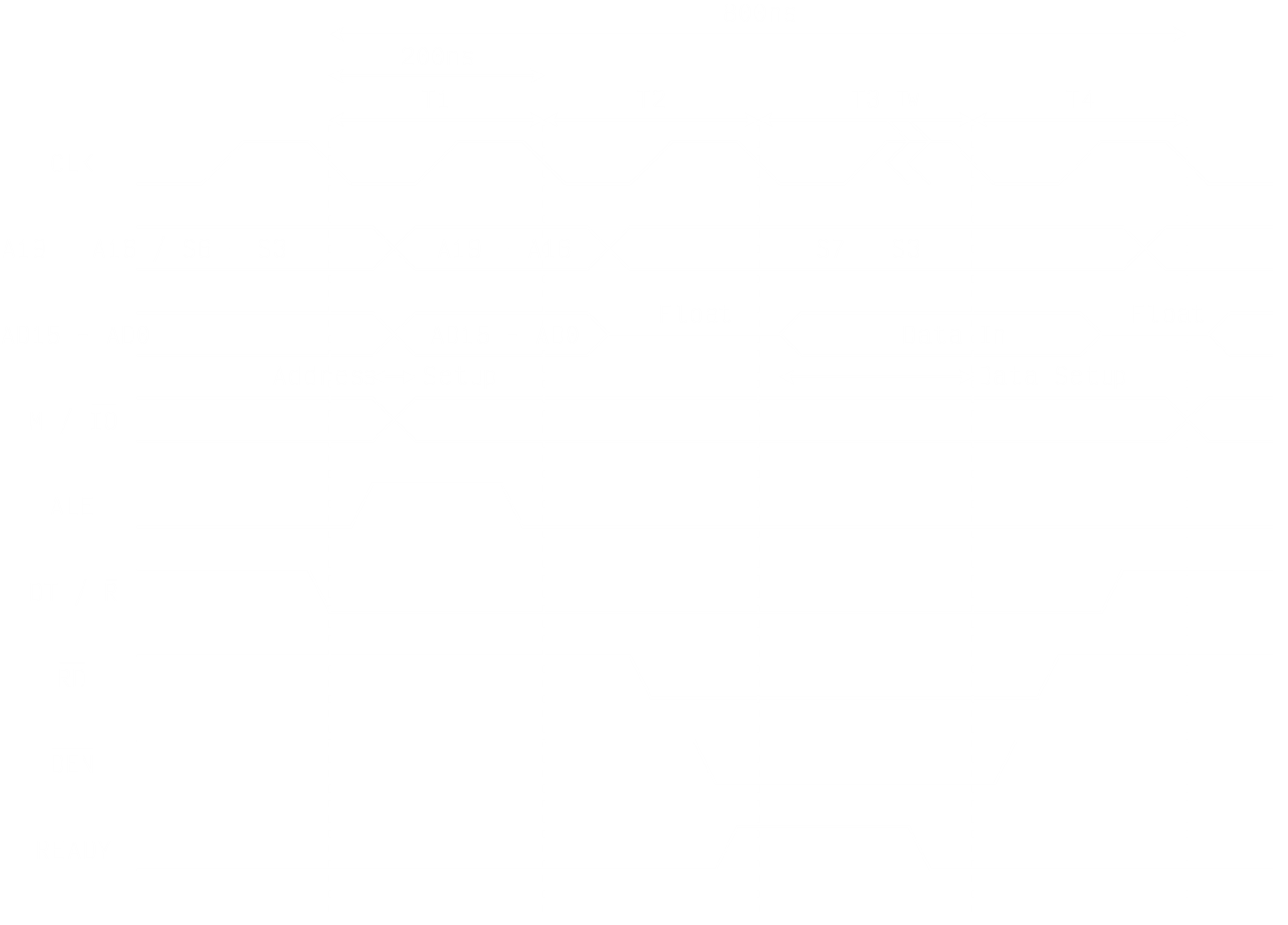
We have seen that we can have multiple **T States**. This is because:

* In the 8086, the address and data lines are **multiplexed**. Thus, the address lines can be used in one state while the data lines can be used in another state.
* The microprocessor needs time to **change signals** during each bus cycle.
* The memory devices need time to interpret the address value and then read or write the data. This is called **access time**.

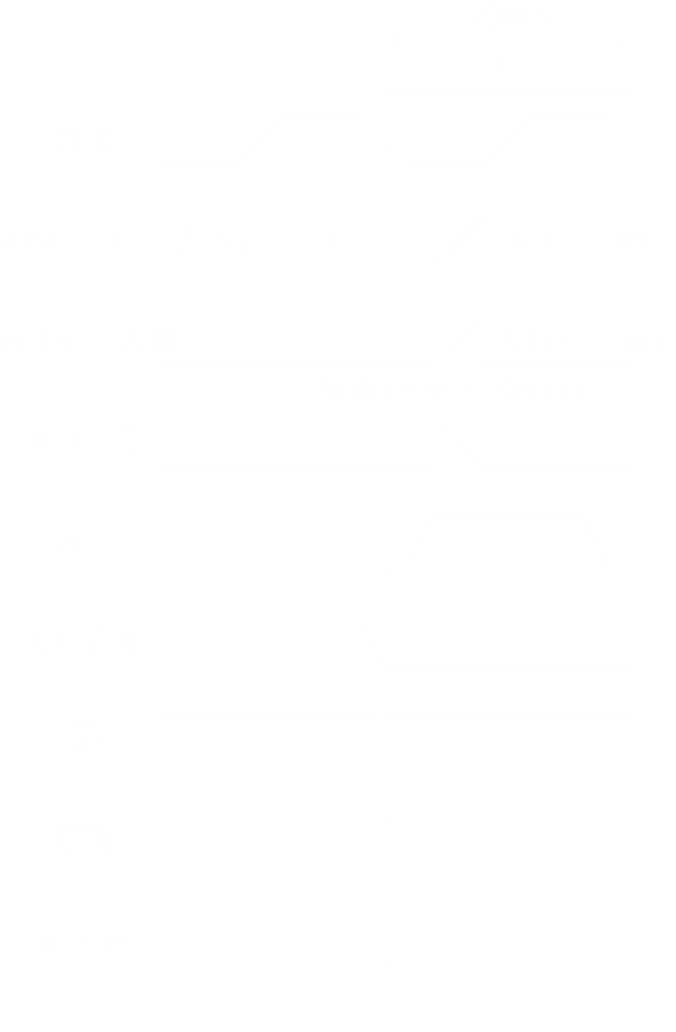
As mentioned before, a single bus cycle consists of at least **4 T states**, denoted as to . Each of these states correspond to a single operation:

* – **Address Output** – The address of memory or I/O is sent out by the microprocessor via the address bus. The , and pins may show output.
* – **Bus Cycle Type** – The microprocessor issues either the , or signal. Without the pin, data cannot be placed on the data bus. In case of , the data to be written also appears on the data bus.
* – **Data Supplied** – At the end of , the pin is sampled. If it is low, is a wait state, meaning there is no operation. If it is high, in the READ bus cycle, the data bus is sampled at the end of .
* – **Data Latched** – All bus signals are deactivated in preparation for the next bus cycle.

### Read Bus Cycle



The complete Bus Timing diagram for a **READ operation** is quite complicated to look at, so instead, we will go step by step.

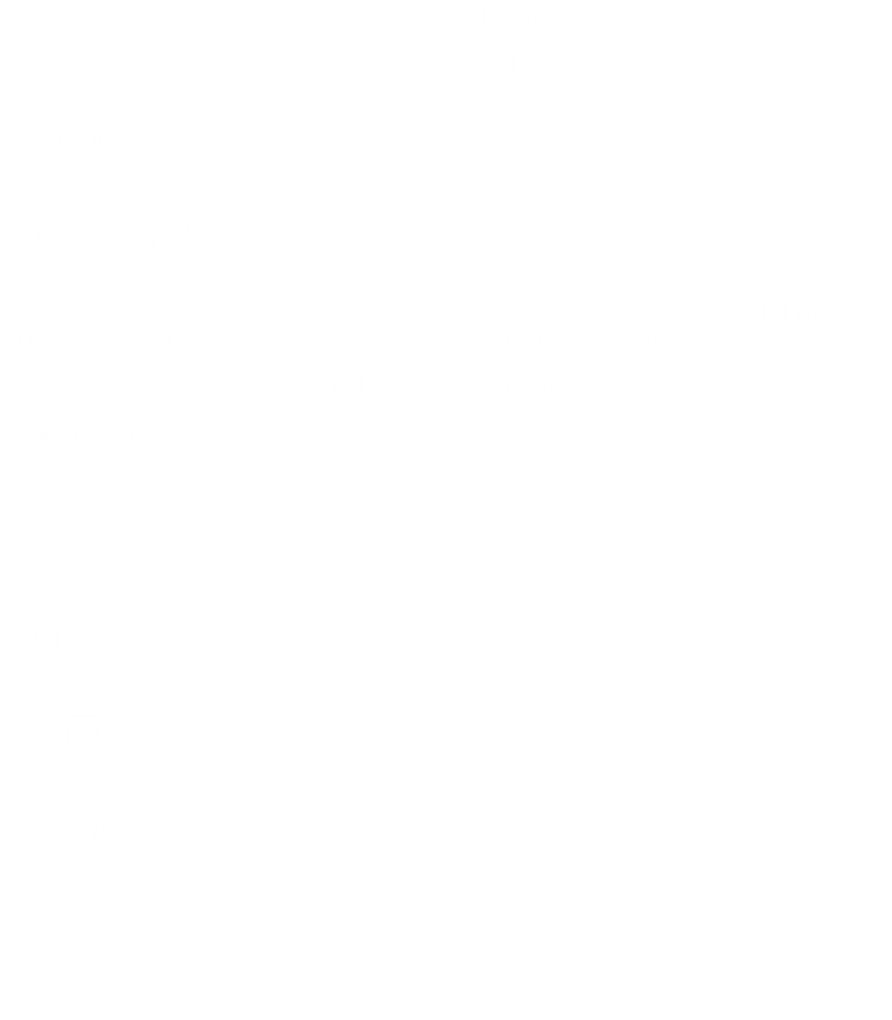


For , we are placing the **address** on the **address bus**. The signal and the signal are both high. However, there is a weird cross-like shape here to indicate that the address is a combination of 0s and 1s, meaning both high and low signals are present. If we have an all 0 address, then these signals would constantly be low. If we have an all 1 address, then these signals would constantly be high.

Similarly, the pin has both options enabled. If we specified either the memory or I/O, the signal would be high or low respectively.

The pin is high, since to are being used with the **address bus**.

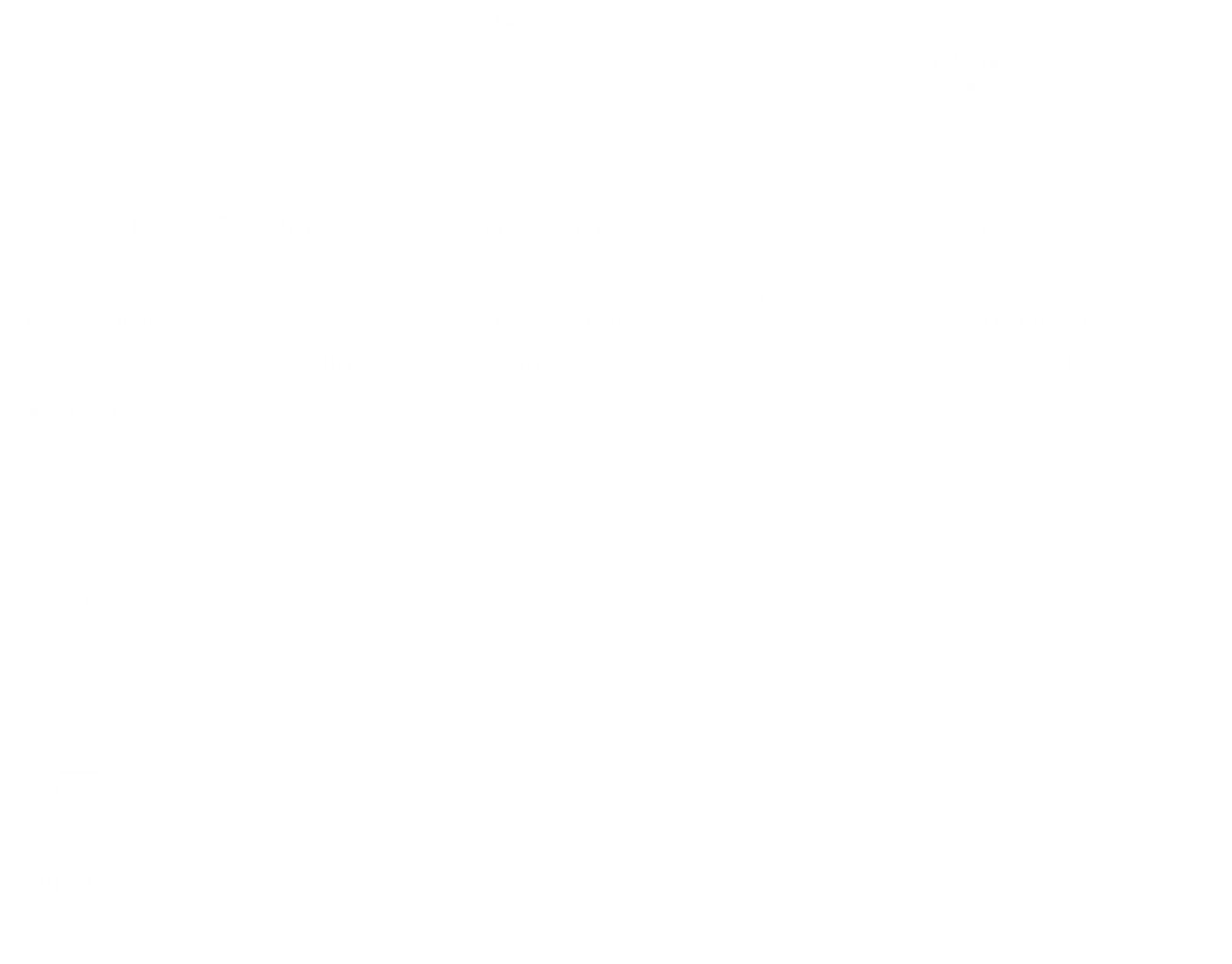
The pin is low since data is being **received**.



During , the pin is **low**, since we are no longer supplying an address. This also makes the pins above it meaningless, thus they are in the **floating** sate.

The pin is now set to **low**, since we want to read data.

The pin is set to **low** and the pin is set to **high** at the end of the state since the microprocessor wants to allow data to be placed on the data bus.

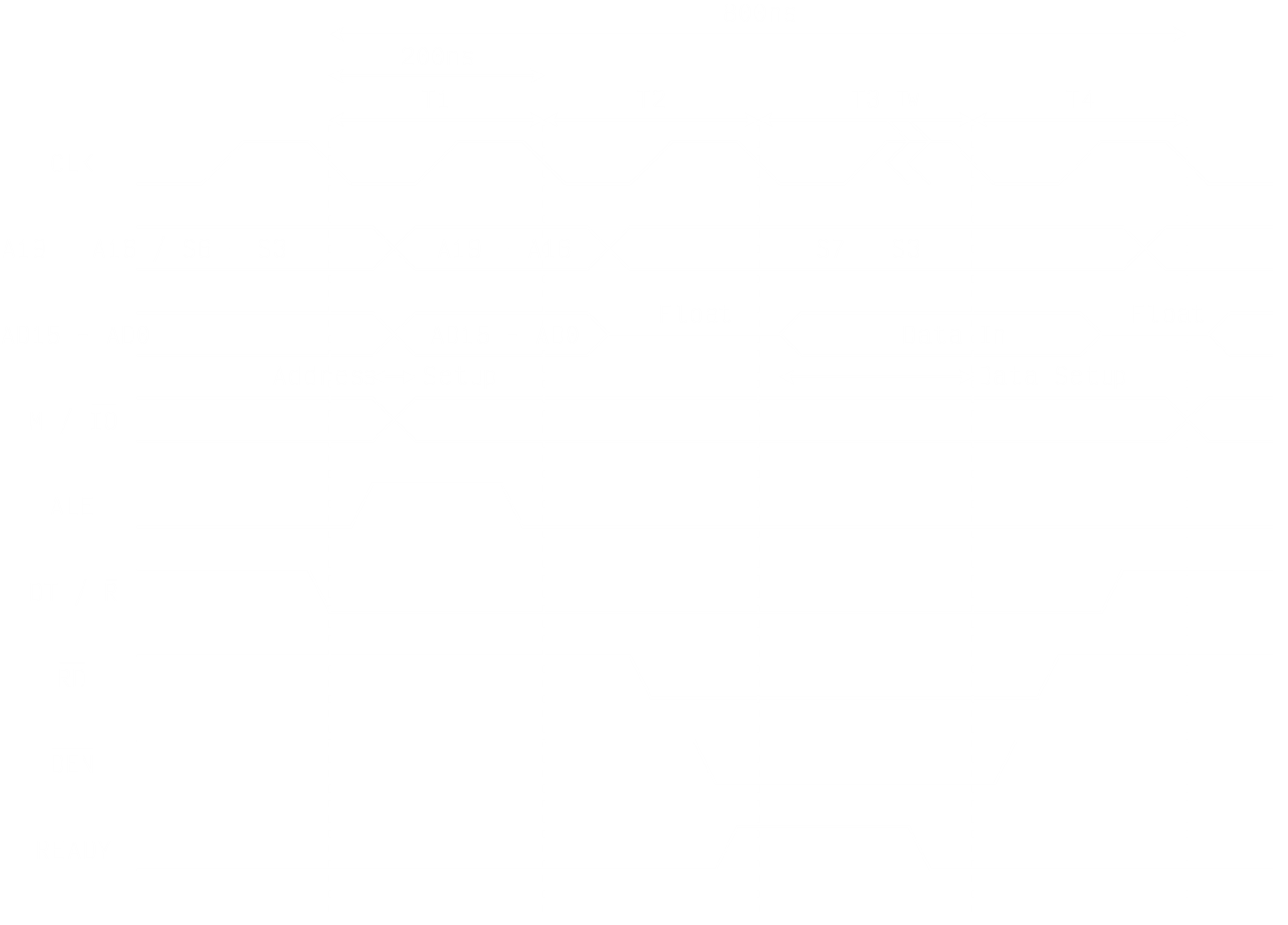


During , the data required has been placed on the **data bus**. The pin is **high** during this time, since the microprocessor is not doing anything else.

Notice that we have a label towards the end of . This stands for **Time for Wait**. It is possible that the microprocessor goes into the **waiting state**, which will also cause the total number of clock cycles in this bus cycle to **exceed four**. This can only happen during .

When the microprocessor goes into the waiting state, the signal becomes **low**. This could happen due to a **high-priority interrupt**.

At the end of , the data from the data bus is **collected** by the microprocessor.



Finally, at , all the signals are **deactivated**. The signals go to their **default state**. The microprocessor needs some time to do this, which is why we have a separate state for this.

### WRITE Bus Timing

Like the process for reading, writing data also has a bus timing diagram:

