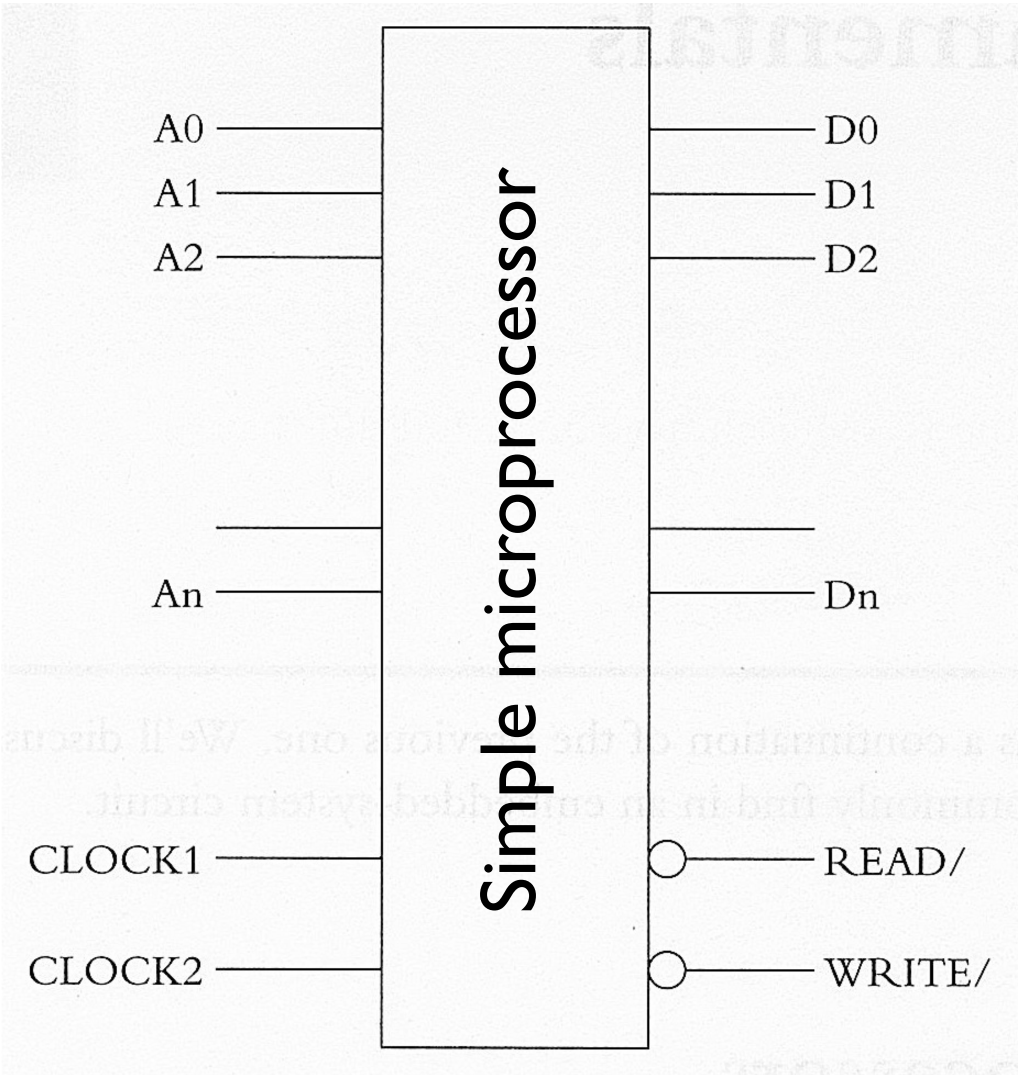
Advanced hardware fundamentals

Reference: Simon chapter 3

Overview of a very basic microprocessor

Every microprocessor contains:

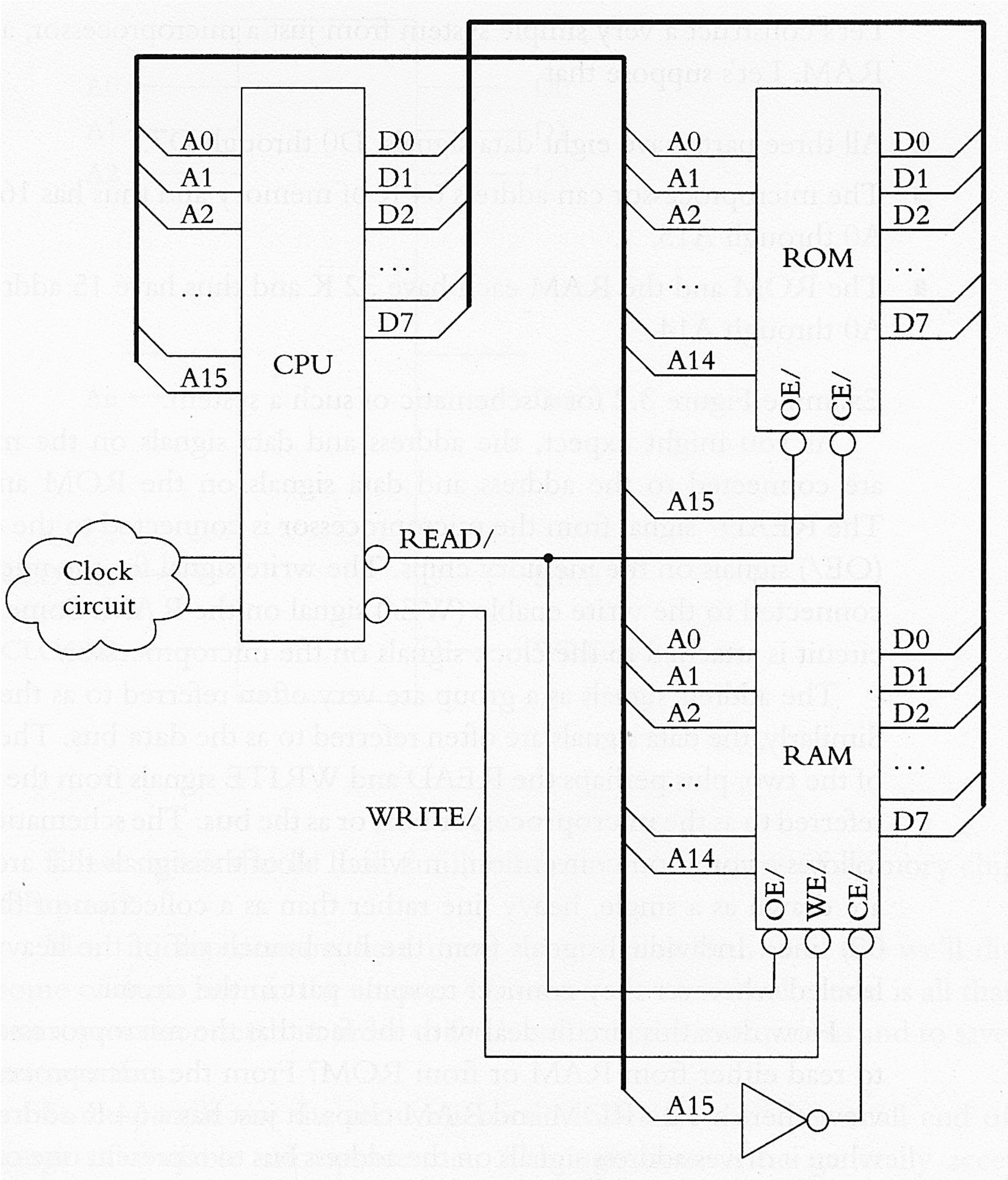
- **Address signals:** specifies which other parts of the circuit (e.g. memory) the addresses it wants to read from or write to.
- **Data signals:** used to get data from and to send data into other parts in the circuit.
- A **READ/** line, which goes LOW when the microprocessor wants to get data, and a **WRITE/** line, which goes LOW when it wants to write data out.



•Clock: However, keep in mind some microprocessors have internal clocks.

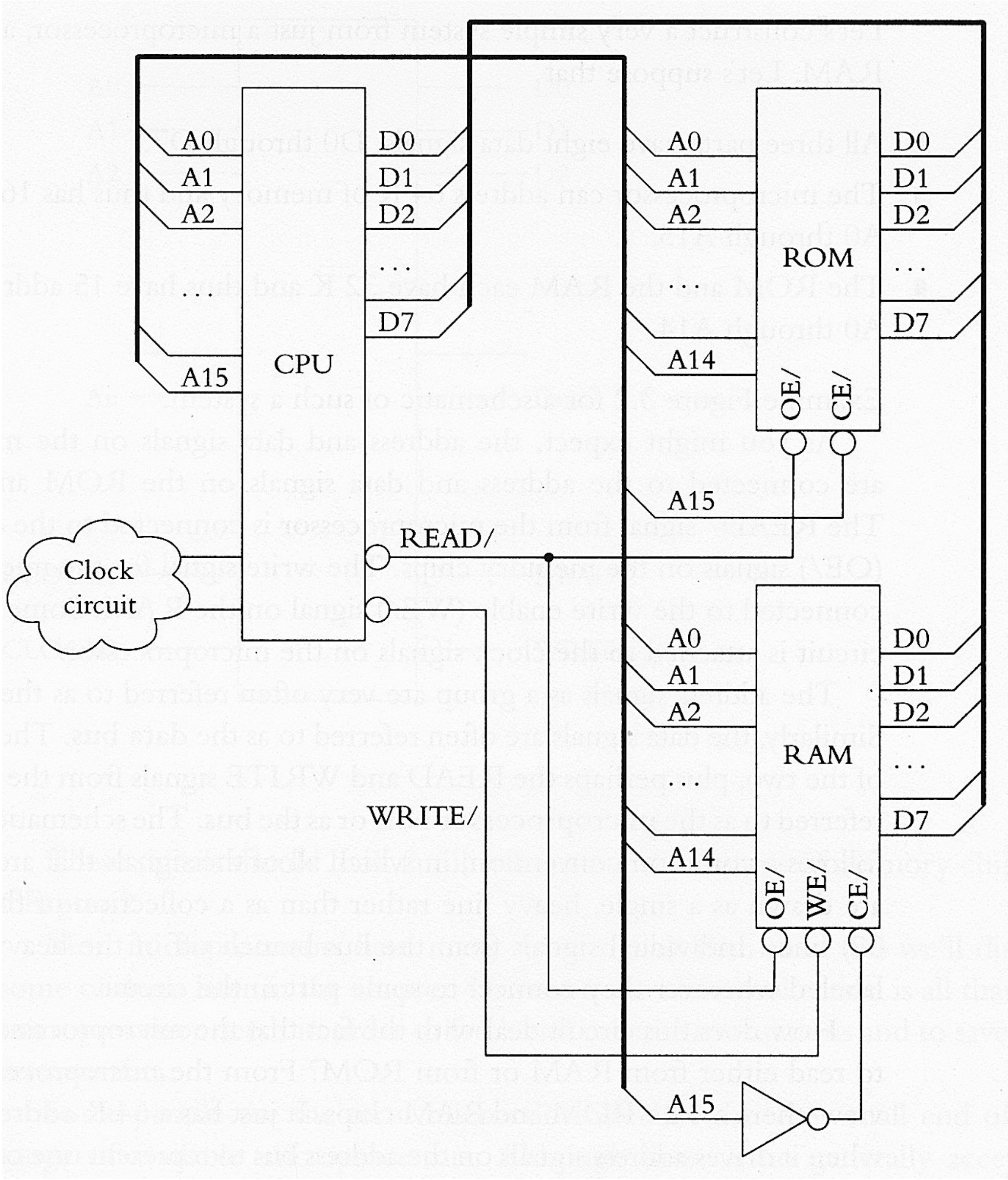
System example

- Our design has a micro-processor, a ROM and a RAM
- All three components have eight data signals, D0 through D7.
- The microprocessor has 16 address lines, A0 through A15, so it can address 64k of memory (2^{16} > 64k).
- The ROM and the RAM each have 32 K and thus have 15 address lines each, A0 through A14.



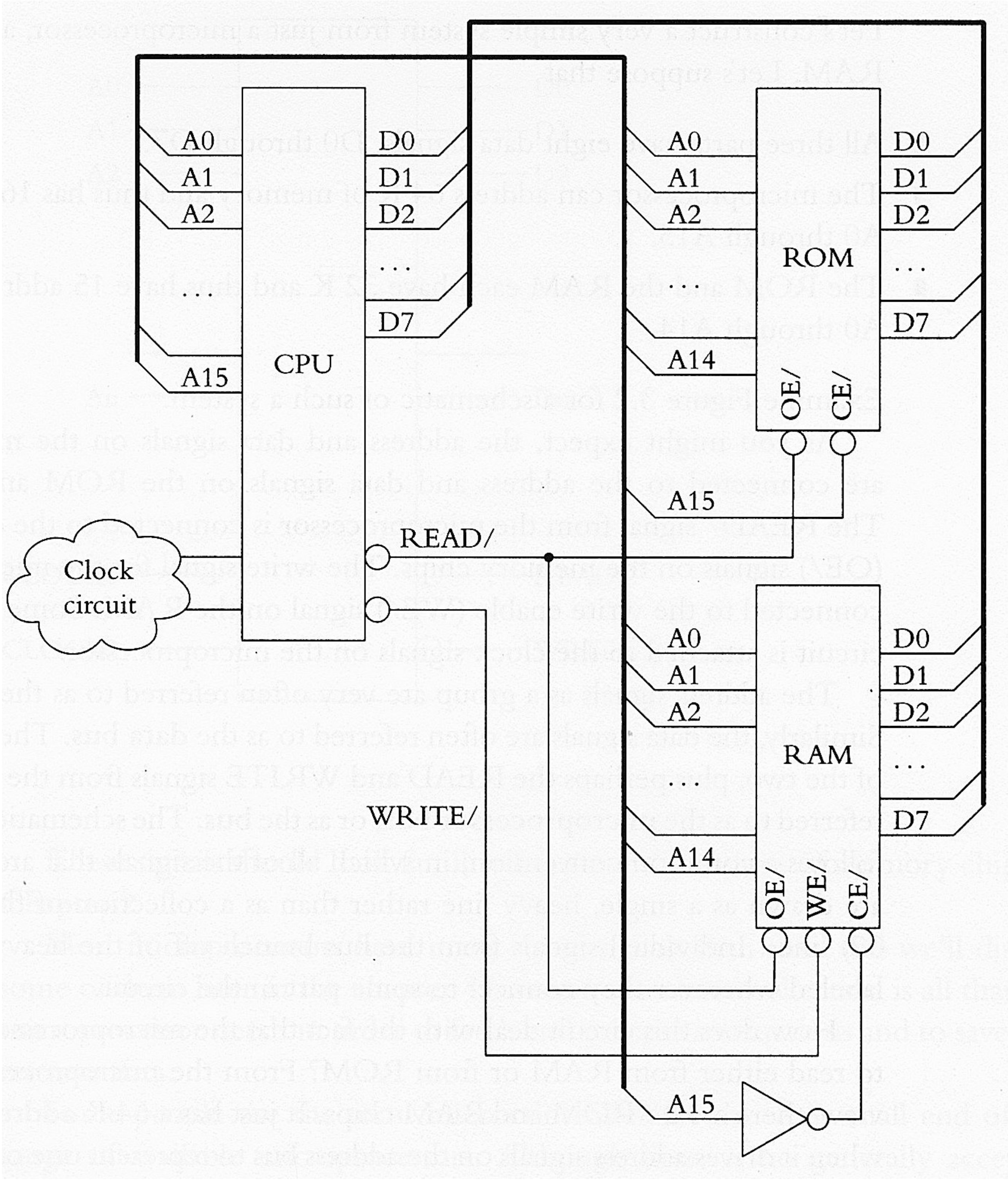
System connections

- The address and data signals on the microprocessor are connected to the address and data signals on the ROM and the RAM.
- The **READ/** signal from the microprocessor is connected to the output enable (**OE/**) signals on the memory chips.
- The write signal for the microprocessor is connected to the write enable (**WE/**) signal on the RAM.



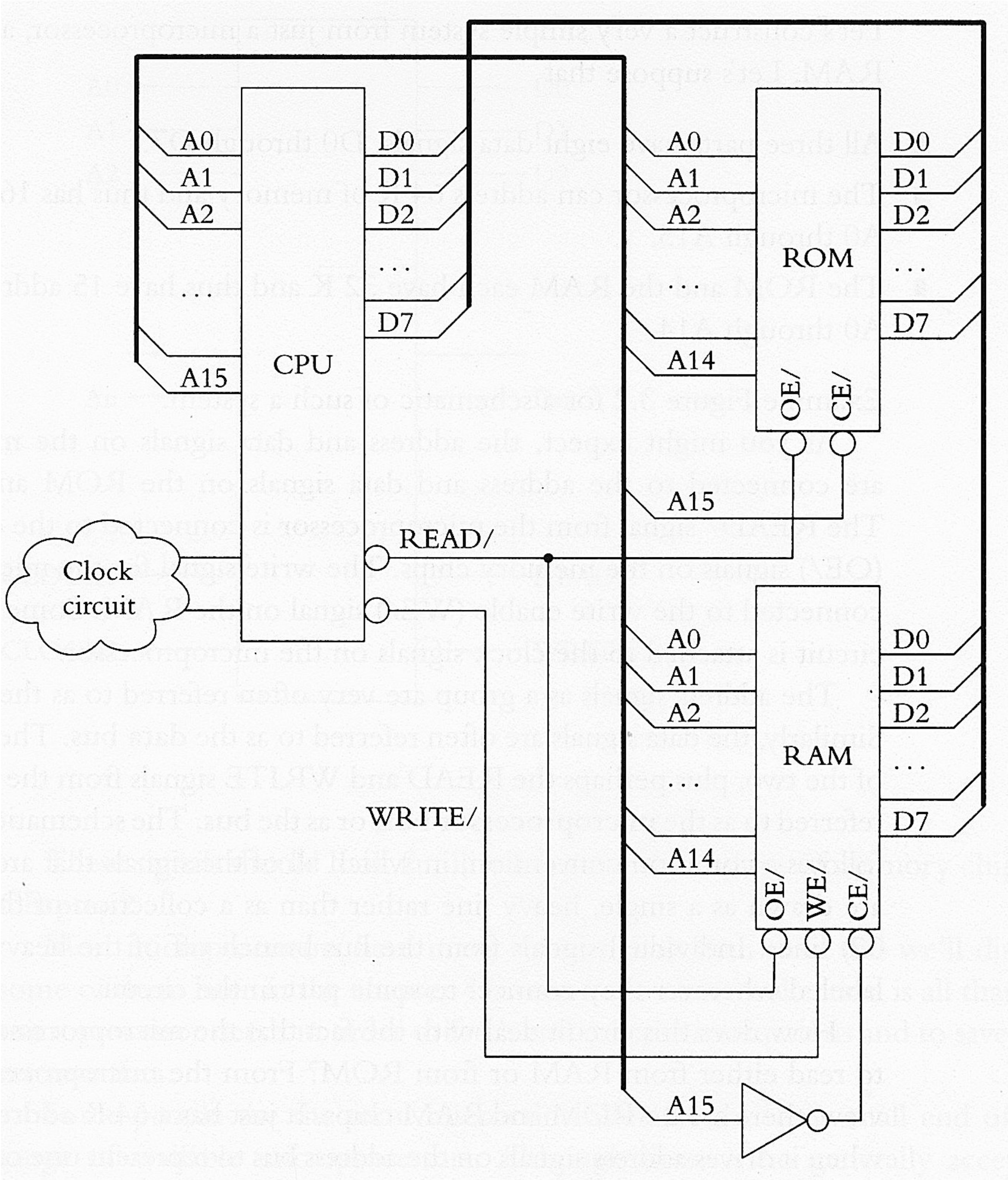
System must be clocked

- Some kind of clock circuit is attached to the clock signals on the microprocessor.



Microprocessor Bus

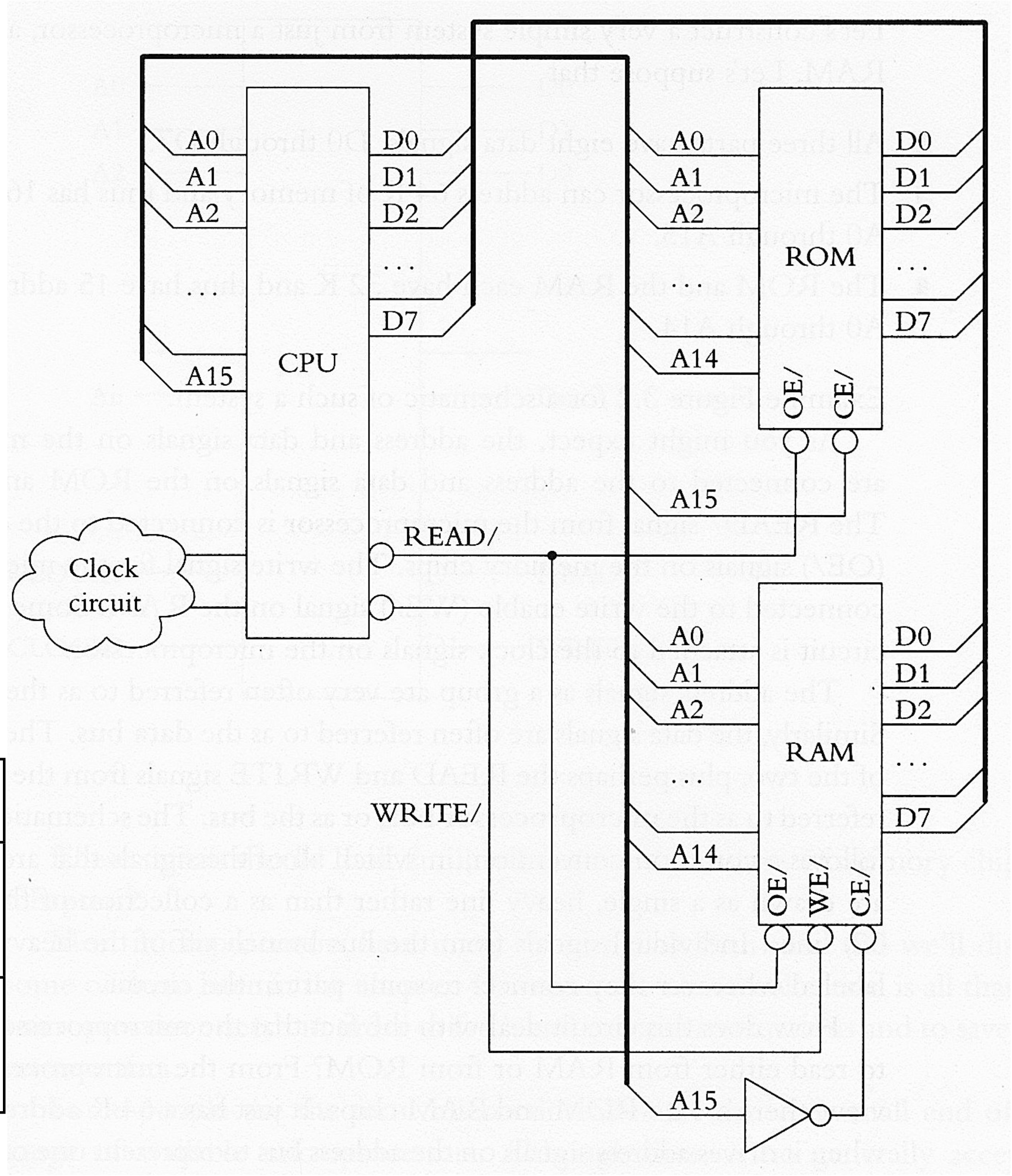
- Some kind of clock circuit is attached to the clock signals on the microprocessor.
- Address signals as a group are referred to as the **address bus**.
- Data signals are referred to as the **data bus**.
- We call the **microprocessor bus** to the combination of the address bus and data bus.



Memory space division

- From the microprocessor's point of view, there are no ROM and RAM chips.
- There is just a 64 K address space.
- We can perform the following division:

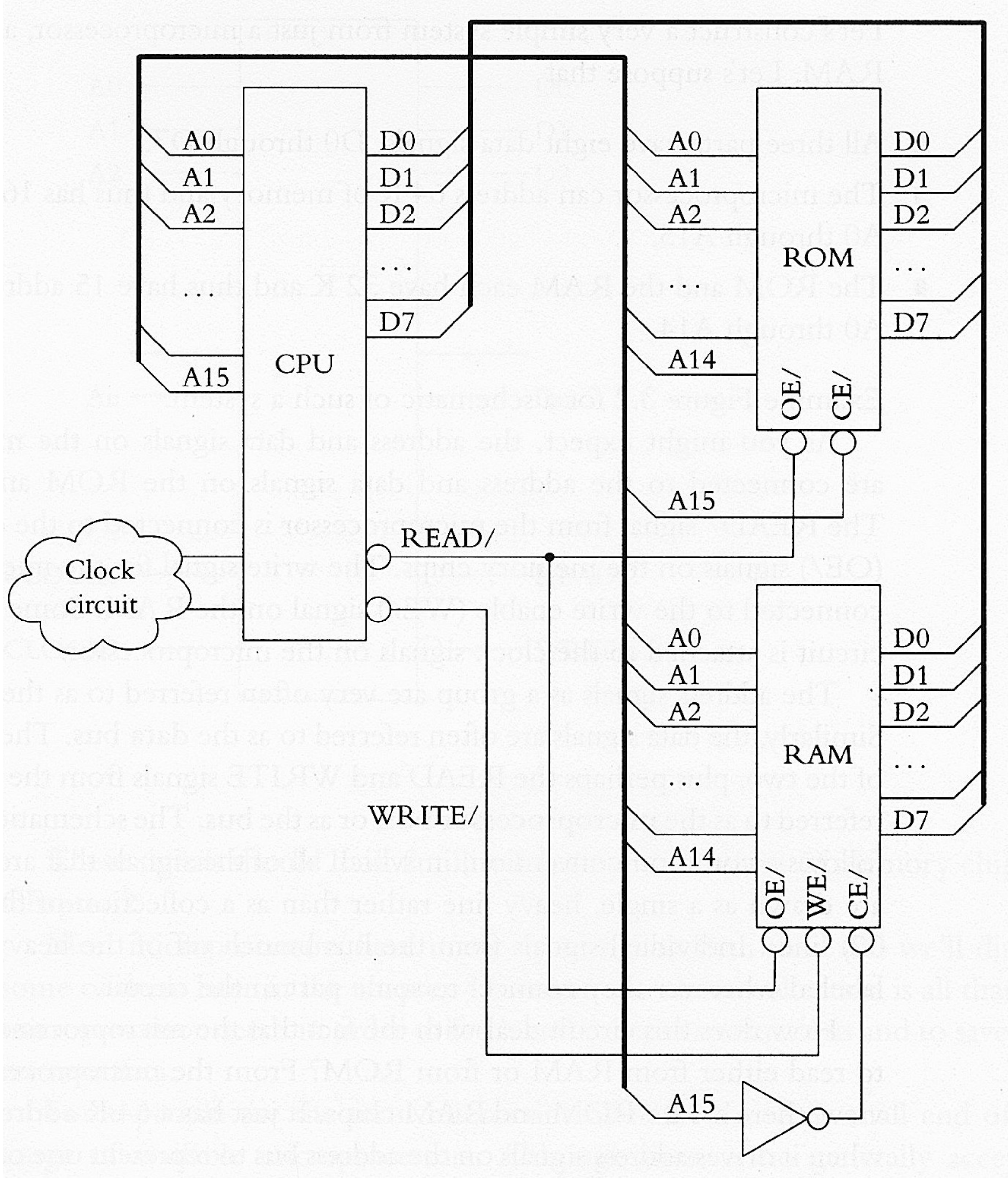
	Low Address	High Address
ROM	0x0000 0000-0000-0000-0000	0x7fff 0111-1111-1111-1111
RAM	0x8000 1000-0000-0000-0000	0xffff 1111-1111-1111-1111



Memory space division

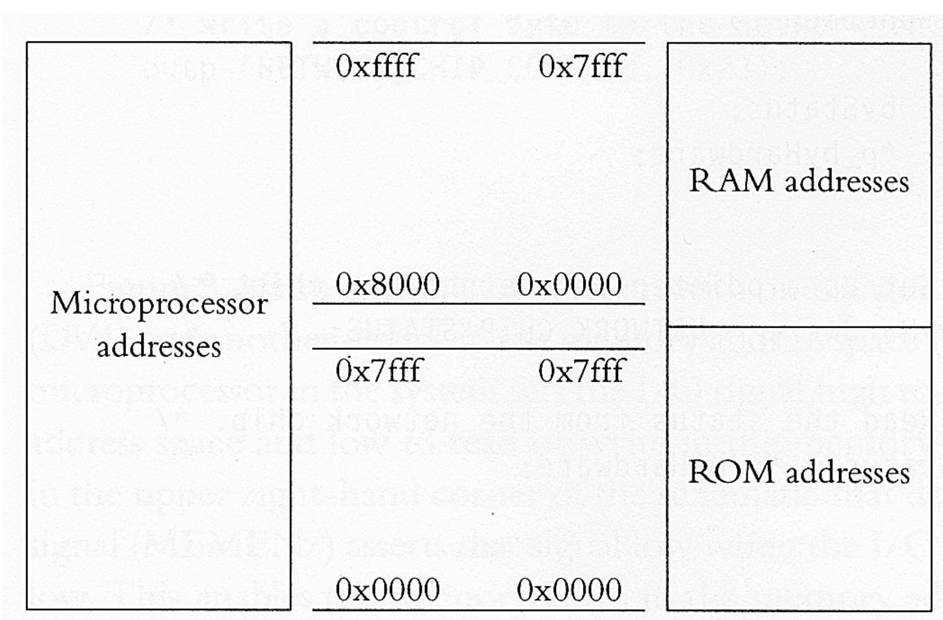
	Low Address	High Address
ROM	0x0000 0000-0000-0000-0000	0x7fff 0111-1111-1111-1111
RAM	0x8000 1000-0000-0000-0000	0xffff 1111-1111-1111-1111

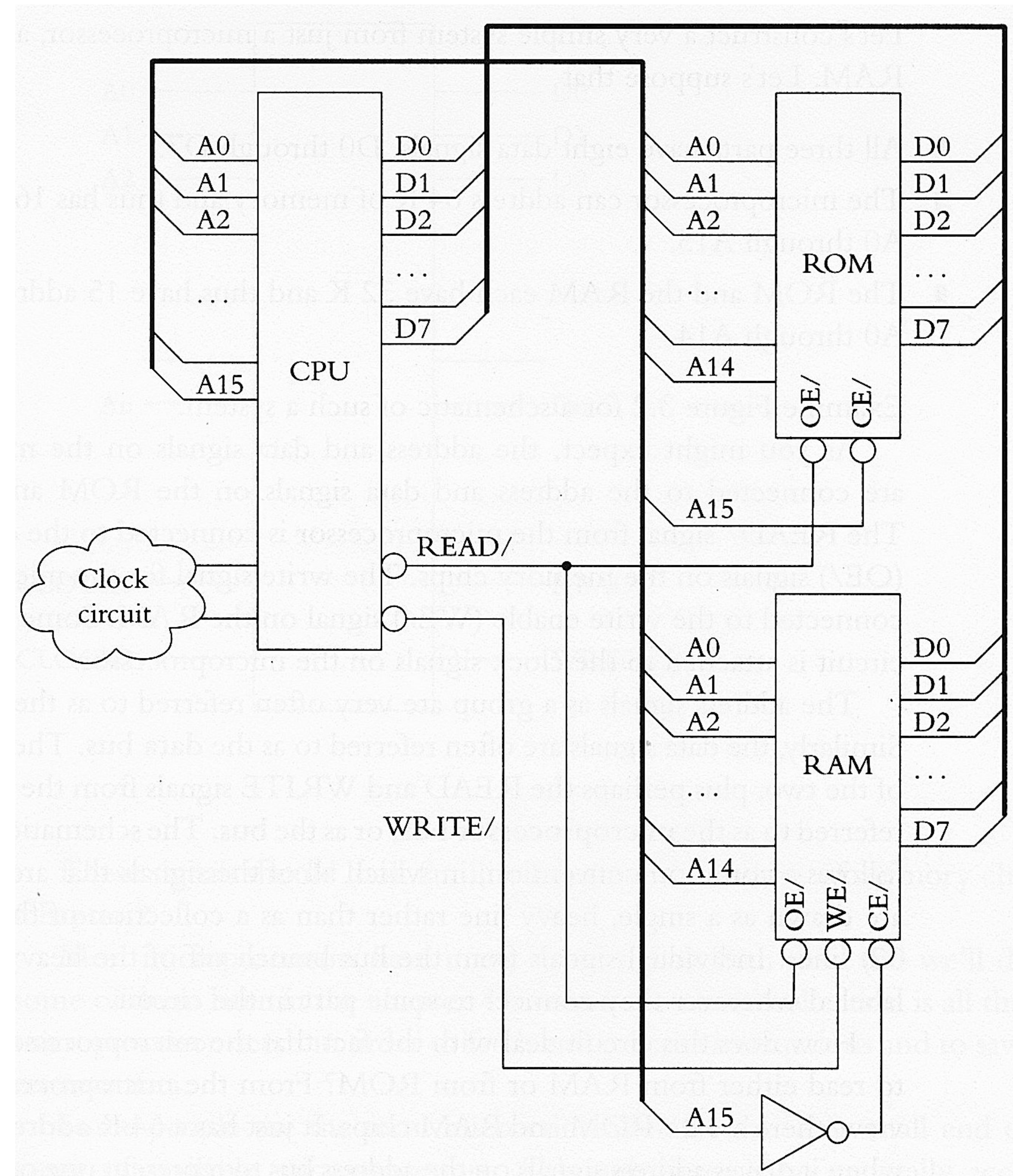
- In the ROM, the highest-order address signal (A15) is 0; whereas RAM A15 is 1.
- We can use the A15 signal to decide which of the two chips (ROM or RAM)
- A15 is attached to the chip enable (CE/) signal on the ROM, enabling it whenever A15 is 0.
- A15 signal is inverted and then attached to the chip enable signal on the RAM



Another look at this memory space

- For example if read address is 0x9123 (1001 0001 0010 0011)
- Then ROM will be disabled since the A15 signal will be a 1
- RAM will then be enabled, and data will be written to address 001 0001 0010 0011.





Memory mapping

- Some additional devices must be connected to the microprocessor data can be exchanged between them.
- The microprocessor and these devices are connected using the address and data bus.
- These devices use an address range that is not used by any of the memory parts.
- For example, the microprocessor may use the bus address (0x80000 to 0x800ff) to access the network chip.
- This is known as **memory mapping**, and its up the hardware engineer to design the proper assertions.

Code sample that uses a memory-mapped device

```
#define NETWORK_CHIP_STATUS ((BYTE *) 0x80000)
(...)

void vFunction  ()
{
   BYTE  byStatus;
   BYTE  *p_byHardware;
   (...)

   /* Set up a pointer to the network chip. */
   p_byHardware = NETW0RK_CHIP_STATUS;

   /* Read the status from the network chip. */
   byStatus = *p_byHardware;
   (...)

}
```

This means some other code would go in here.

This memory address contains the network chip status.

Additional devices on the microprocessor bus

- Some microprocessors allow two address spaces: a **memory address space** and an **I/O address space**.
- A microprocessor that supports I/O address space will have some extra pins, which specifies which address space is currently being addressed.
- The most common implementation requires a single pin: **LOW** for the memory address space and **HIGH** for the I/O address space.

Microprocessors with I/O address space

- Microprocessors that support an I/O memory space have extra assembly language instructions for doing that.
- The **MOVE** instruction reads from or writes to memory;
- The **IN** and **OUT** instructions, access devices in the I/O address space.
- C contain functions to read to and write from devices in the I/O address space
- For example *inport*, *outport*, *inp*, *outp*, *inbyte*, *inword*, *inpw*.

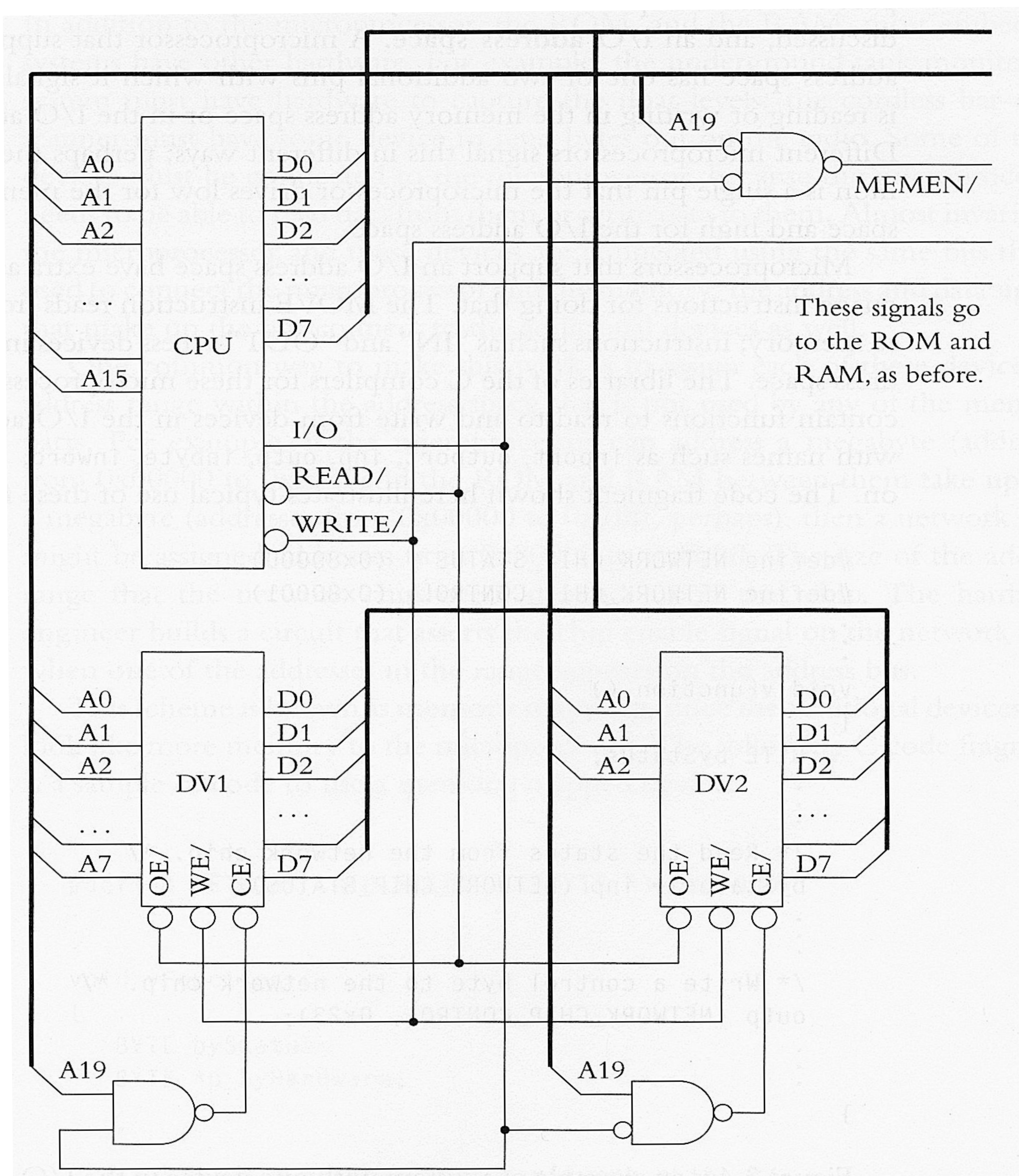
Code sample calls an I/O address space

```
#define NETWORK_CHIP_STATUS (0x80000)
#define NETWORK_CHIP_CONTROL (0x80001)
(...)
void vFunction  ()
{
  BYTE  byStatus;
  (...)

  // Read the status  from the network chip.
  byStatus = inp (NETWORK_CHIP_STATUS);
  (...)

  // Write a control byte to the network chip.
  outp (NETWORK_CHIP_CONTROL, 0x23);
  (...)
  }
```

This memory address contains the network chip status.

This C instruction (inp) will collect information from a particular address in the I/O address space.

Example of a system with I/O address space



- DV1 is a device in the I/O address space
- DV2 is a device in the memory address space
- DV1 is enabled when A19 and I/O are both high
- DV2 is enabled when A19 is high and I/O is low.

Bus handshaking

Bus handshaking

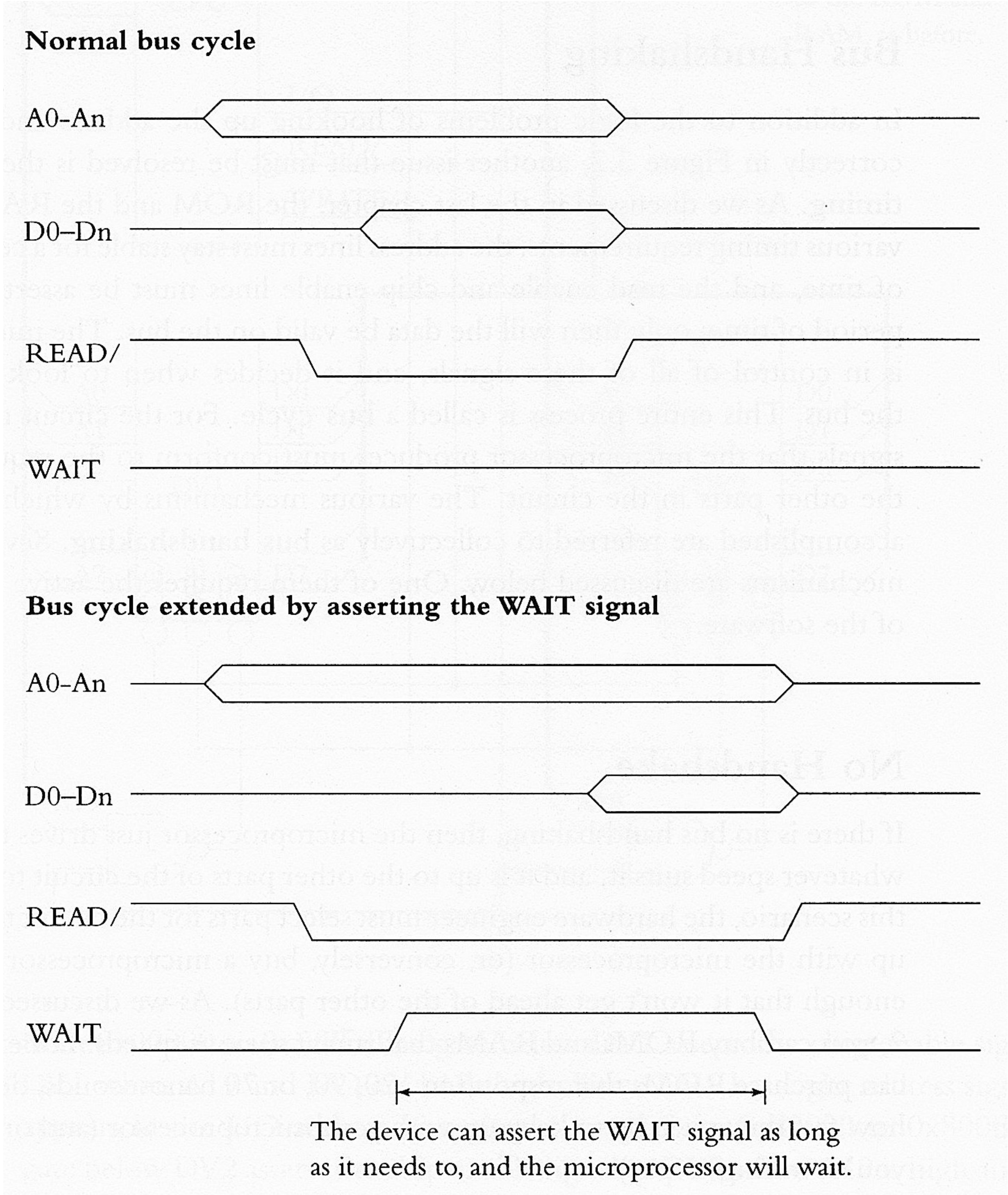
- Both RAM and ROM have different timing requirements.
- The data is valid on the bus if:
  1. The **address lines** are stable for a certain period of time.
  2. The **read enable** and **chip enable** lines are asserted for some period of time.
- This period of signal stability is called a **bus cycle**.
- Microprocessor must conform to timing requirements of other components too. The various mechanisms are called **bus handshaking**.

Handshaking method #1: no handshake

- With no bus handshaking, the microprocessor just drives the signals at whatever speed it can.
- It is up to the other parts of the circuit to keep up.
- The hardware engineer will buy ROMs and RAMs that run at the desired speeds (e.g. 120, 90, or 70 ns).

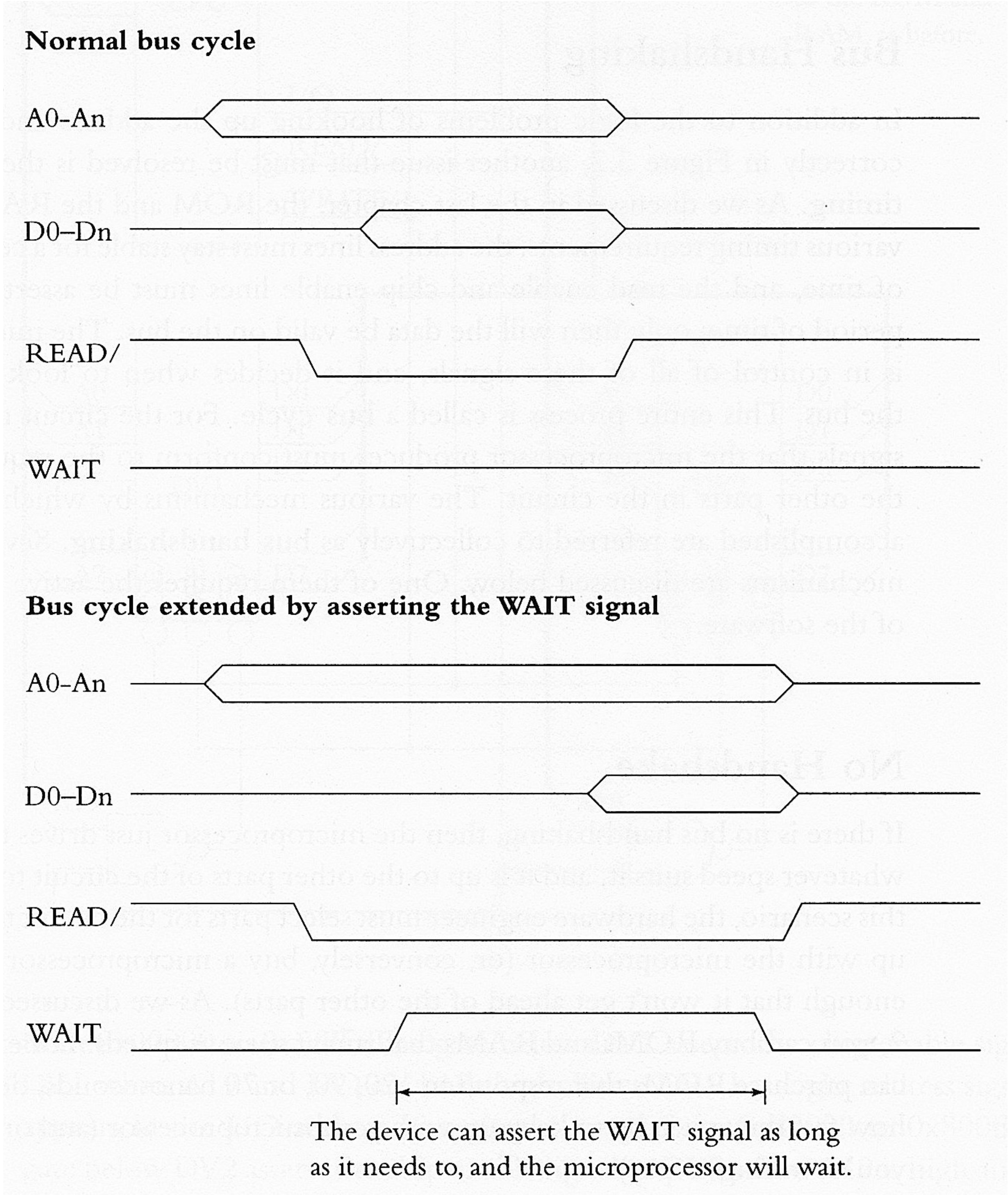
Handshaking method #2: wait signals

- Some microprocessors have a WAIT input signal that the memory can use to extend the bus cycle as needed.
- If a device cannot respond as quickly as that diagram requires, it can assert the WAIT signal to make the microprocessor extend the bus cycle.



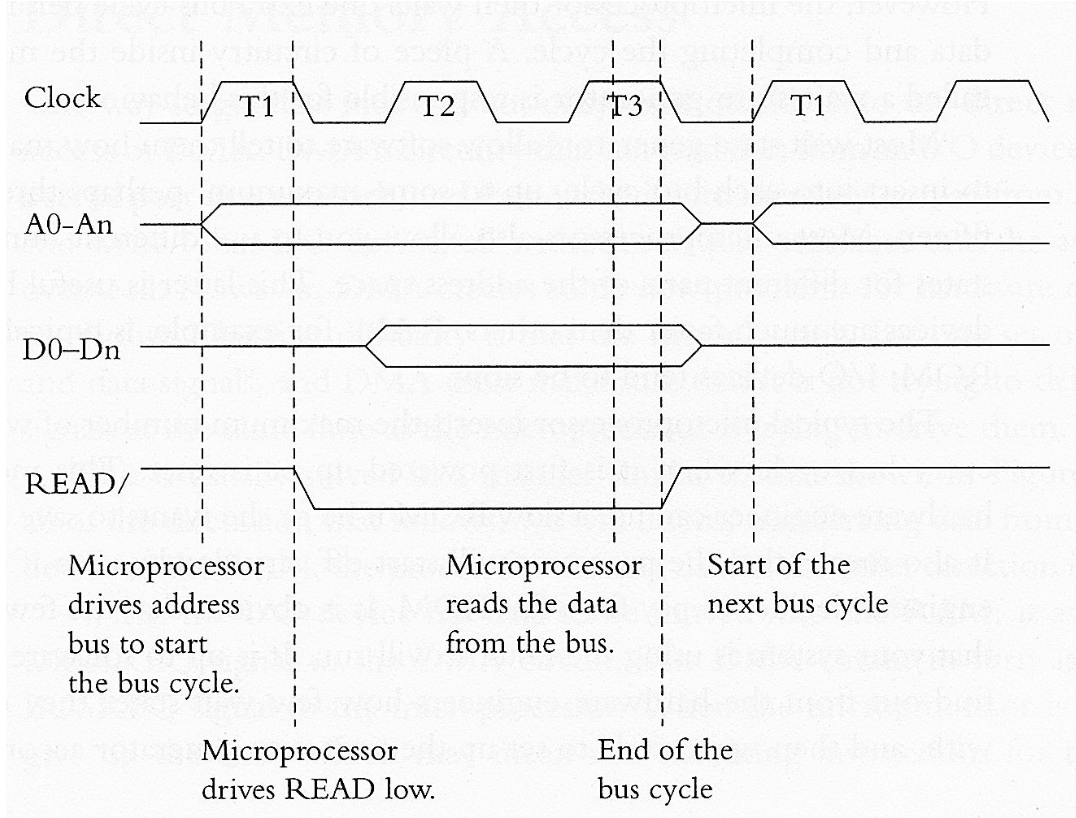
Wait signal can put a device on hold indefinitely

- **READ/** means the microprocessor wants data!
- As long as the WAIT signal is asserted, the microprocessor will wait indefinitely for the device to put the data on the bus.
- Unfortunately, standard ROMs and RAMs don't come with a wait signal, so engineers need to build some external circuitry to drive the wait signal correctly.



Handshaking method #3: wait states

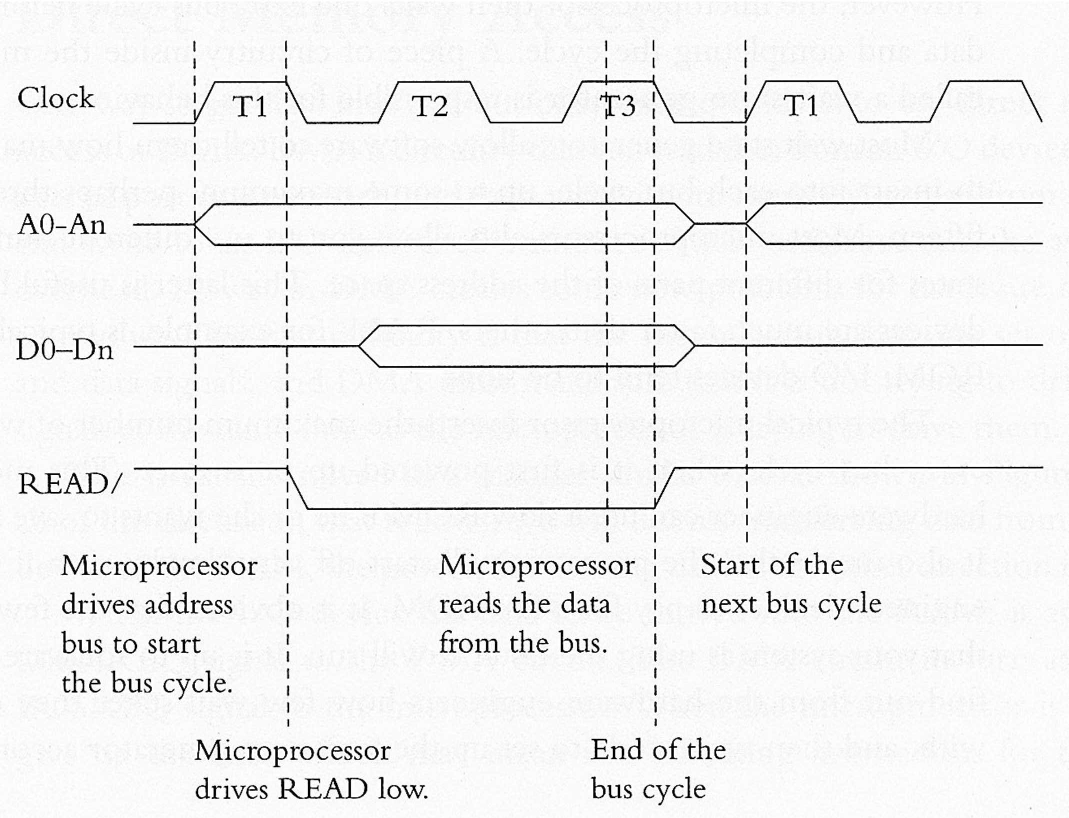
- The microprocessor has a clock input, which is used to time all activities.
- Each of the signal changes happens at a certain time in relation the the microprocessor input clock signal.
- The bus cycles are called T1, T2, T3 and so on...



How a micro-processor works

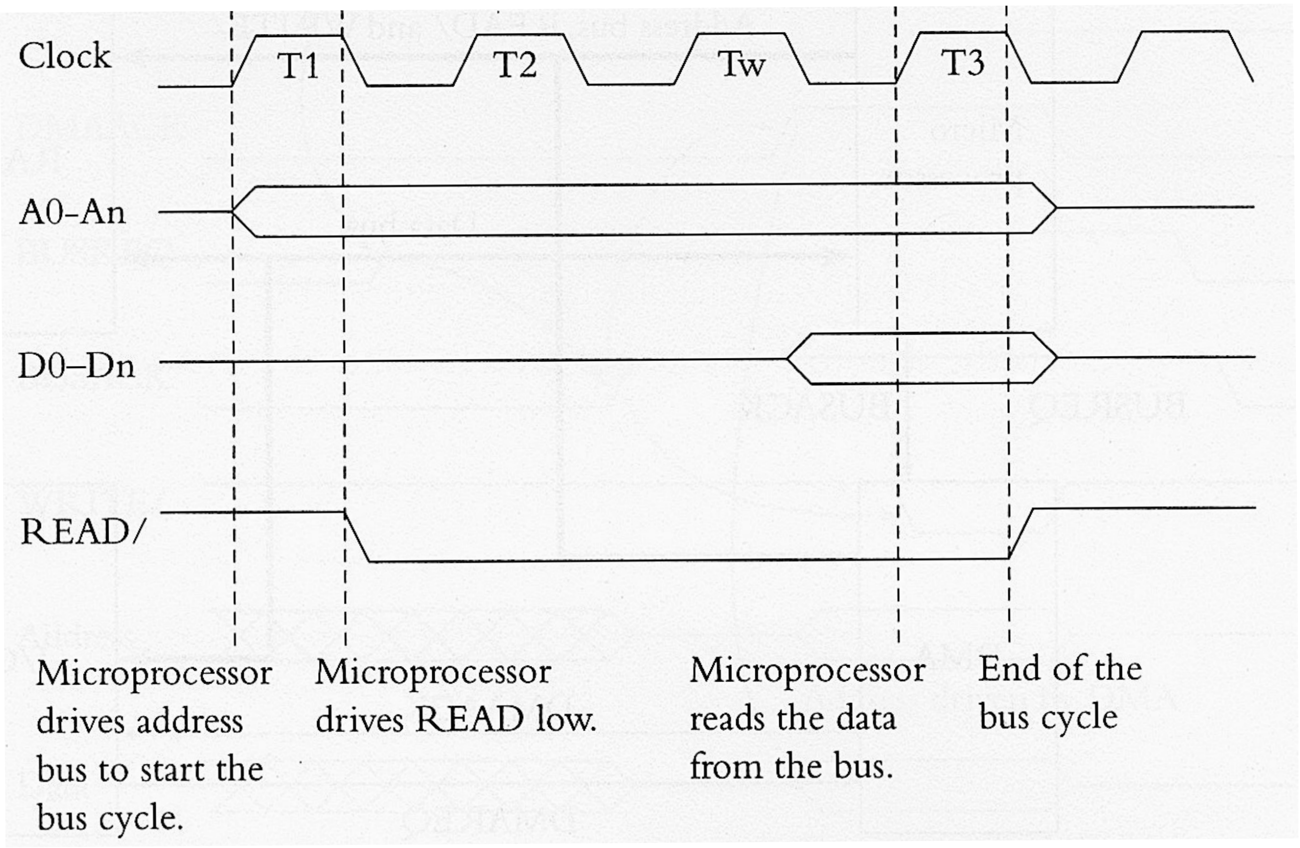
The microprocessor behaves as follows:

- @ rising edge of T1 it outputs the address
- @ falling edge of T1, asserts the **READ/** line.
- @ rising edge of T3 it takes the data in (since it should be valid).
- @ falling edge of T3, it de-asserts the **READ/** line.



Using wait states

- If the microprocessor is able to use wait states, then it can insert clock cycles between T2 and T3.
- The microprocessor will wait another clock cycle (TW) for the data to be ready.
- **Wait state generator** is the piece of hardware that inserts wait states.
- In software we can specify the number of desired wait states.



Direct Memory Access (DMA)

What is direct memory access (DMA)?

DMA is a piece of circuitry that, without software assistance, can:

- Read data from an I/O device, such as a serial port or a network, and then write it into memory

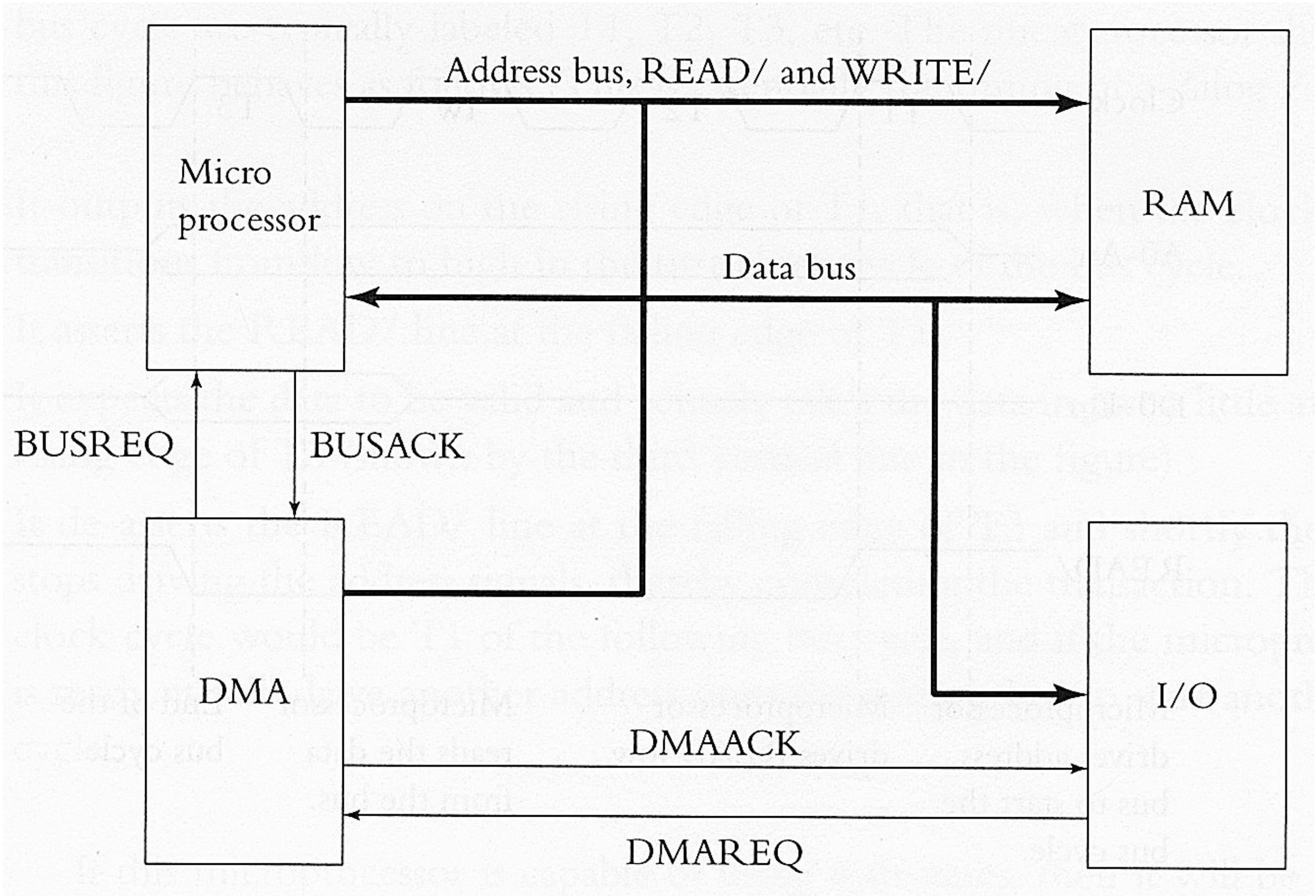
or

- Read from memory and write its contents into an I/O device

Caution: memory only has one set of address and data signals. The DMA must make sure that it is not driving those signals while the microprocessor is also driving them.

Using DMA: we want to transfer the data from the I/O into RAM

1. **DMAREQ** signal is asserted.
2. DMA circuit asserts **BUSREQ** to the microprocessor.
3. When microprocessor is ready to give up the bus, it asserts **BUSACK**.
4. DMA circuit puts address in the address bus.
5. **DMAACK** and **WRITE/** are asserted.
6. I/O places data in the data bus.

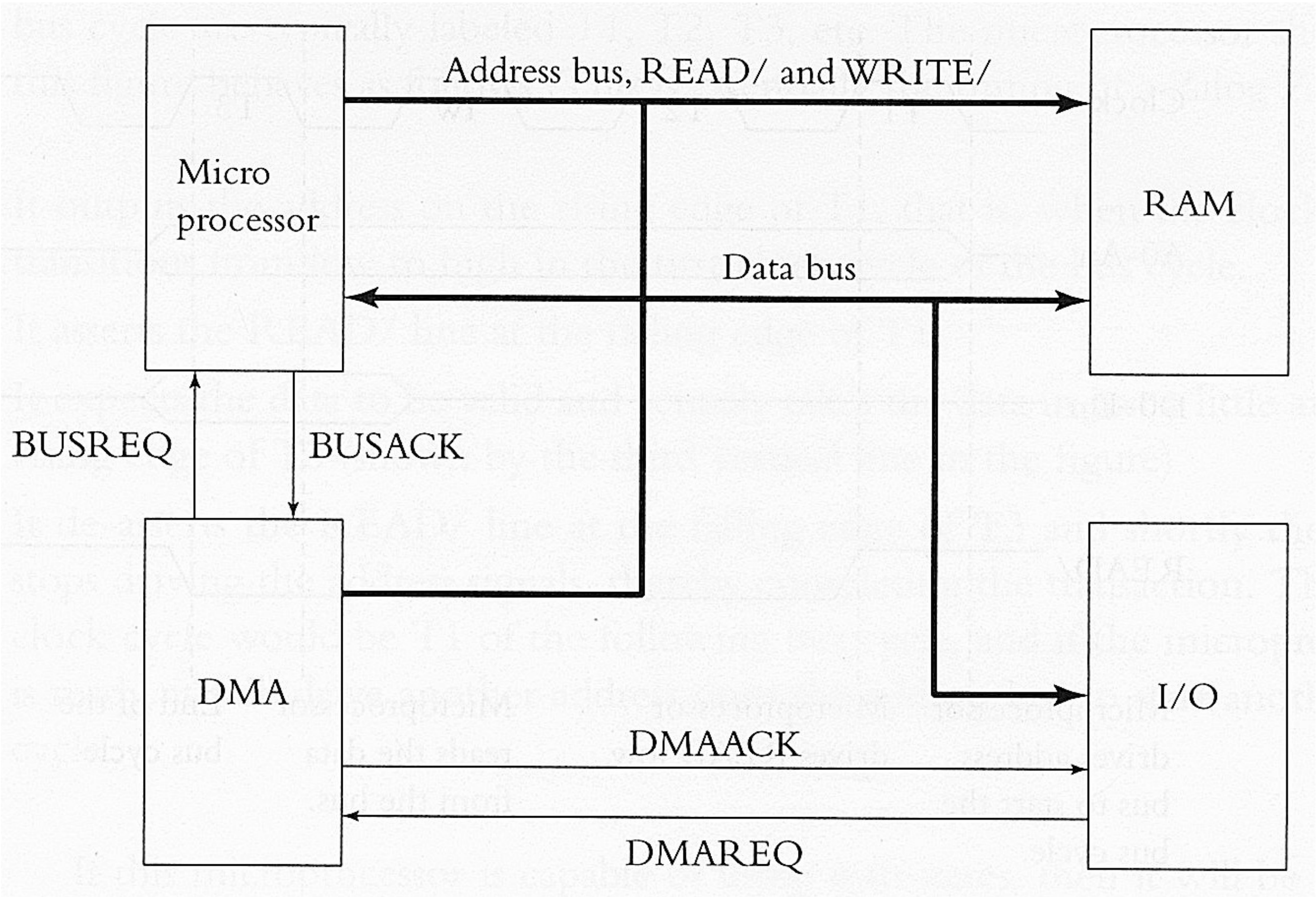


What's next?

Now that data has been written to RAM, the DMA circuit releases:

- DMAACK
- Address bus
- BUSREQ

The microprocessor releases BUSACK and microprocessor execution resumes.



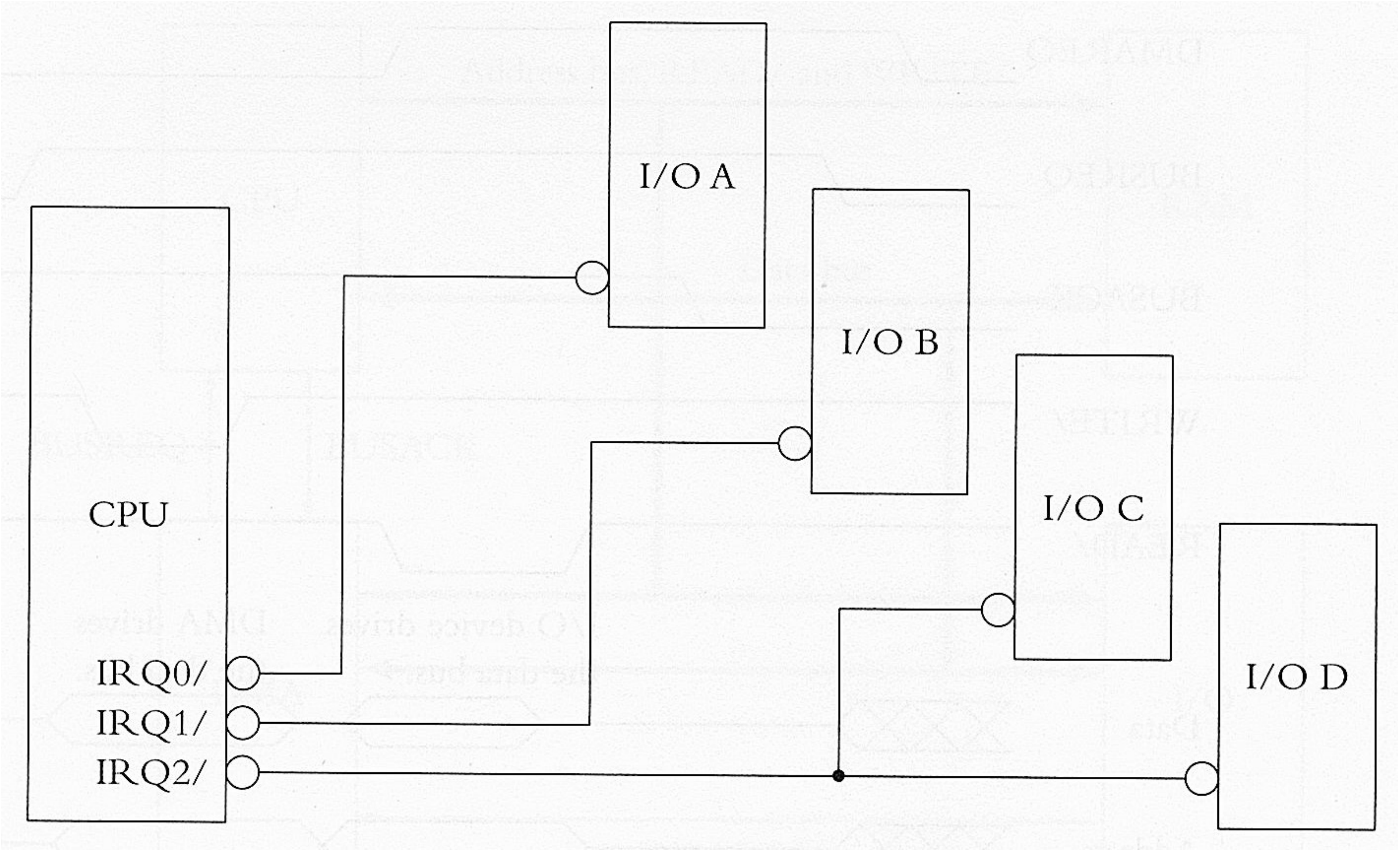
Interrupts

What is an interrupt?

- Micro-controllers can be ordered, or **interrupted**, to stop whatever they are doing, and execute another piece of code
- This other piece of code is the **interrupt routine**.
- The signal that tells the microprocessor to run the interrupt routine is the **interrupt request** (or **IRQ**).
- Interrupt request signals are typically LOW.
- It is typical for interrupt request pins on I/O devices to be open collectors, so they can share an interrupt request pin on the microprocessor.

Interrupt example

- **I/O device A** can interrupt the processor by asserting **IRQ0/**
- **I/O device B** can interrupt the processor by asserting IRQ1/
- **I/O device C and D** can interrupt the processor by IRQ2/

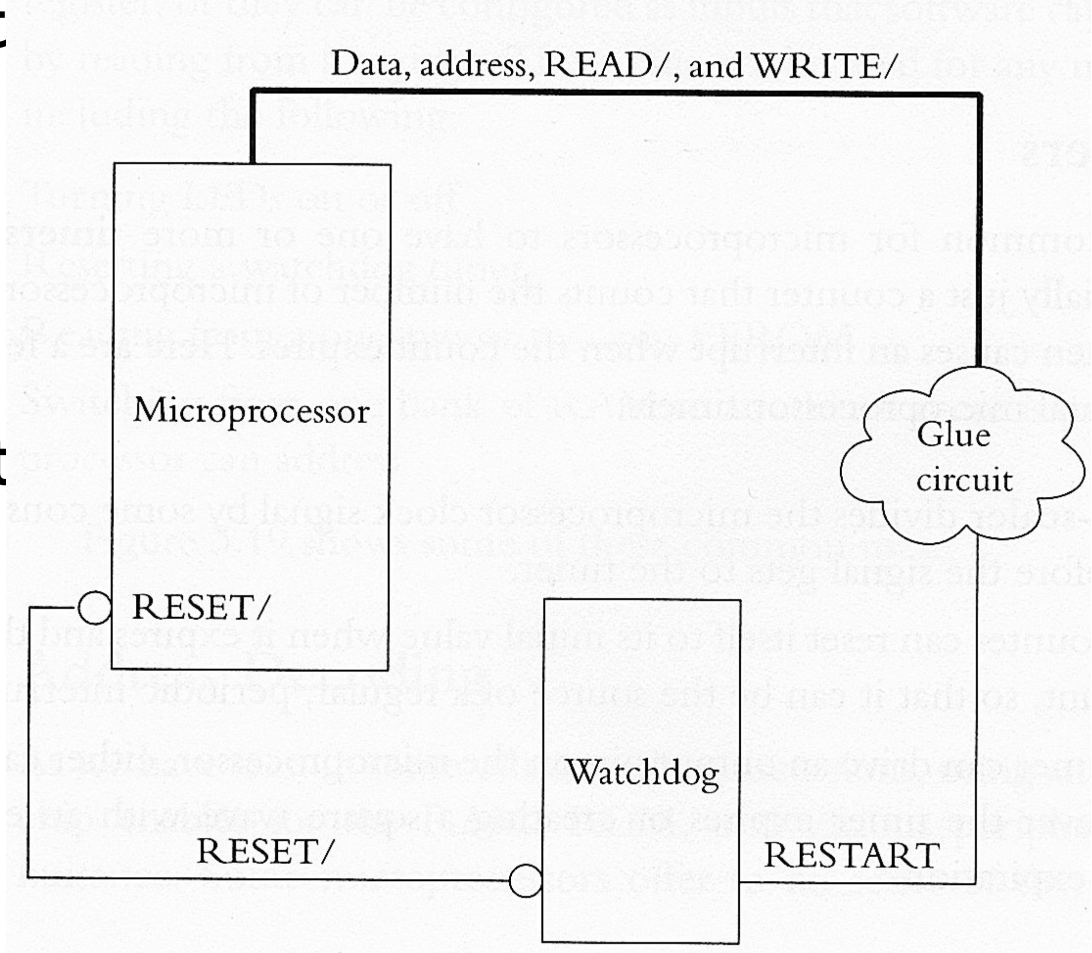


WESTERN NEW ENGLAND UNIVERSITY WNE

Watchdog timer

Watchdog timer

- A watchdog timer contains a timer that expires after a certain interval unless it is restarted.
- The watchdog timer has an output that pulses should the timer ever expire, but the idea is that the timer will never expire.
- If the timer expires, (because it was never restarted), then the software has crashed.



How is a watchdog timer connected to a circuit?

- The output of a watchdog timer is connected to the **RESET/** signal of the microprocessor.
- If the timer expires, the pulse on its output signal resets the microprocessor and starts the software from the beginning.
- Different watchdog timer circuits require different bit patterns to restart them (ergo the glue circuit).