12. Mass memory

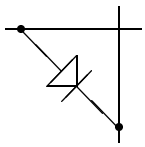
Any computer, from μ to super (vector or other arquitectures), contains from few registers to many hundreds. These registers store data temporarily, to transfer information between internal functional blocks.
To use a computer efficiently, having some sort of application, we need to store the program itself and data going into or coming out of the program. The principle may be the same (e.g. use of D flip-flops, ...), but now we need many "registers": mass memory. We will concentrate on so-called main memory. We will not discuss serial devices such as disks and tapes.
There are 4 items to be discussed: memory type, organization, speed and usage (read, write, address decoding).

12.1 Memory types

RAM random access memory - any word can be read/written.

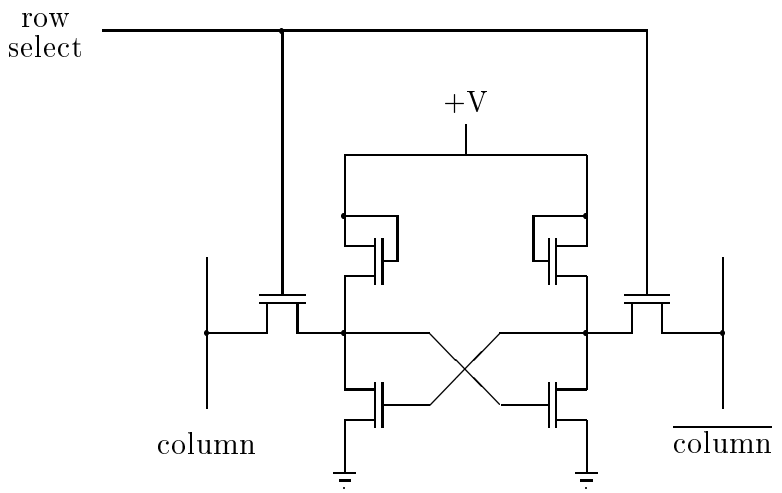
ROM read only memory - can only be read, so must be "programmed" in the factory (one can order these).

PROM programmable read only - can be programmed by the user

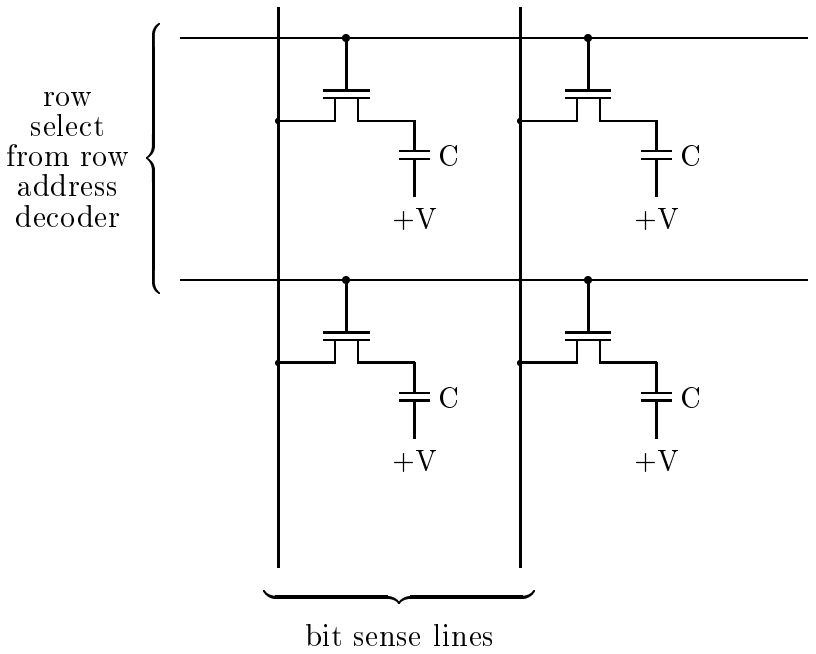


- fusible-link PROM: each matrix element (a diode in a line matrix) has a low current fuse that can be melted by applying a pulse which exceeds a critical current.
- EPROM: UV-light-erasable PROM is most common, can be reprogrammed many times. **Drawbacks**: chip must be removed from board to apply UV light; byte erasure is impossible - all information is lost; quartz windows make production more expensive.
- E^2PROM (electrically-erasable PROM): a newer development which eliminates the disadvantages of EPROMs.
  NOTE: for all these types special circuits are required.

STATIC RAM - each cell consists of a simple FF like a SR one, so each bit requires 4–6 transistors. Each bit is accessed by a row and column select signal, generated by the address decoder. It looks like:



DYNAMIC RAM - each cell consists of a simple select MOS (metal oxide semiconductor) transistor plus a small capacitor:



Advantage: very high density; more bits/chip than static RAM's.
Disadvantage: C leaks, so periodically each bit must be refreshed and this every 2 msec!

Refreshing is not difficult: each column has its own sense amplifier and this can be used to recharge a cell in a given row. Row-select sequencing is applied: first select row 1, then row 2 and so on. Because of the extra circuitry required, special chips were made to do

this job (DRAM controllers). Furthermore, the chip layout is designed such that different blocks can be accessed at the same time, for example in 128×128 bit-cell quadrants with common row addresses. But nowadays all necessary circuitry is integrated in the memory chips

Static and dynamic RAM are both volatile. When the power is switched off, all data is lost. CMOS static RAMs consume very little power in "stand-by" mode. In some systems CMOS-SRAM is used in combination with a battery (battery-backed memory) in order to keep all information while the computer is switched off. For IBM compatibles one can buy such systems as a plug-in board (which are even made such that they behave as a hard disk - one can boot from a memory board!).

12.2 Memory organization

This refers to the number of bits in an input/output word. Important is the total number of bits/chip, but also the word length. Most static RAMs are available with a word length of 8 bits, so a data bus and all bus drivers, etc. may also be 8 bits wide. If a 16-bit μ-processor is used (an 8086 from Intel© for example), two "banks" of 8 bits must be used in parallel, one for the high-order byte and one for the low-order byte (well, not really - there are controllers that can transfer a 16-bit processor into a machine that accesses 8-bit wide memory).

Because of the 8-bit wide word length, less IC pins are available for the address lines. This is not a problem for STATIC RAMs because they have a smaller capacity relative to dynamic RAMs.
For dynamic RAMs, the capacity is higher and the refresh must be dealt with. For these reasons the number of address lines is greater so that, for the same number of IC pins of a static RAM, fewer data pins are used. Often only 1! This means that an 8-bit system requires 8 DRAM chips in parallel.

In summary, for static RAMs organizations of 2K×8 to 64K×8 are common, whereas dynamic RAMs organizations of 16K×1 to 4M×1 are also found.
However, there exist DRAMs of e.g. 16K×8 which have all refresh control logic "on board". EPROMs have always 8 bit words because of the required external programming equipment (which is standard...).

12.3 Memory speed

There is a tendency towards larger capacity as well as speed, both for SRAMs and DRAMs. SRAM design is concerned with capacity and speed, whereas DRAM design is mainly concerned with capacity.
For this reason, the capacity of DRAMs is about 16× the capacity of SRAMs and the speed of SRAMs is 8–16× the speed of DRAMs. For fast applications SRAMs are always used.

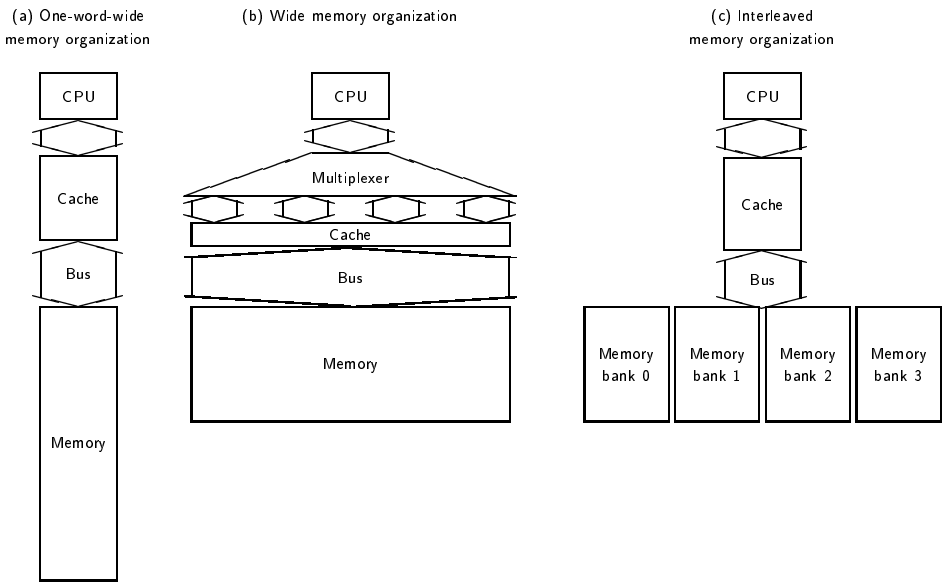
There are additional techniques to speed-up performance of a system, apart from selecting SRAMs instead of DRAMs: caching and interleaving.

Caching means that a part of a program/data in memory is placed in a special buffer; the same principle as virtual memory (a page can be on disk or in main memory).

A few typical numbers:

	CPU registers	cache	main memory	disk
typical size	< 1Kb	< 512Kb	< 512Mb	> 100Mb
access time (ns)	10	20	100	10–20,000,000
bandwidth (Mb/sec)	800	200	133	4
controlled by	compiler	hardware	operating system	op-syst/hdware

Interleaving means that memory chips are organized in banks which are 1-word wide, but the banks are addressed in parallel, such that multiple words can be read/written at a higher speed.



In this figure:

(a) the same word length for CPU, cache and memory. The fast cache allows fast transfers, but transfers between cache and memory are relatively slow.

(b) the cache and memory are broader than the CPU word length. Because of the cache and memory width, the effective memory transfer is faster.

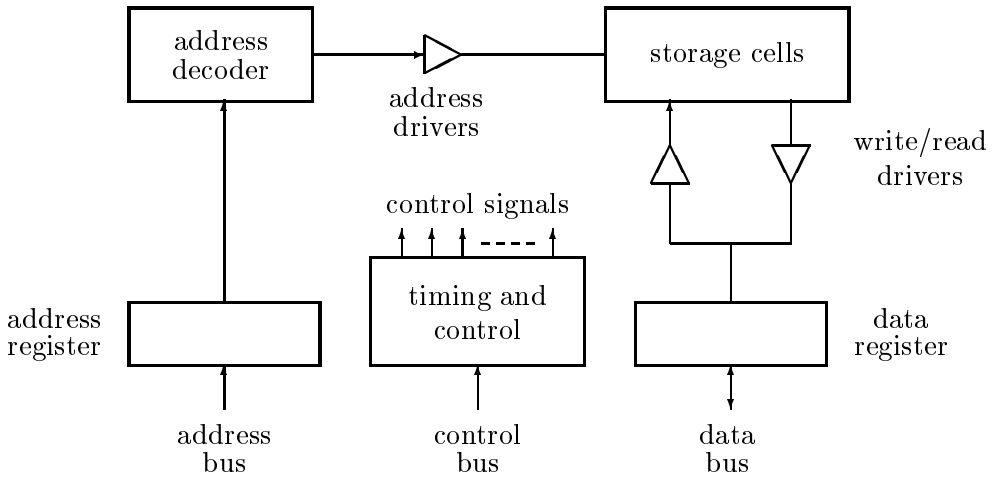
(c) Interleaving: 4 words can be read/written from/to memory at a higher speed.

(b) and (c) are more or less equivalent: special hardware is required such as a fast multiplexer and a bank-select mechanism. This is done by generating extra address codes (b) or by using the two least significant address bits for selecting the memory bank (c). Address decoding is discussed below.

Caching and interleaving requires optimization: if the CPU wants to access code/data which is not in cache memory, the specific part containing the code/data must be loaded from main memory. The question is then how big the parts must be.

12.4 Memory usage (read, write, address decoding)

Memory chips are functional blocks which can be seen as a black-box, the user does not need to know how exactly it is organised. A simplified scheme generally looks like this for a RAM chip:

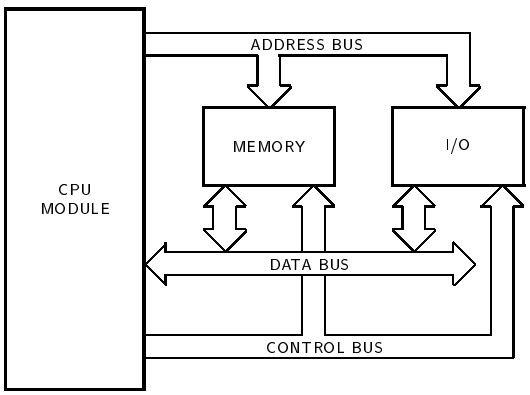


Data bus: a number of lines on which data is transferred into or output of the memory chip.

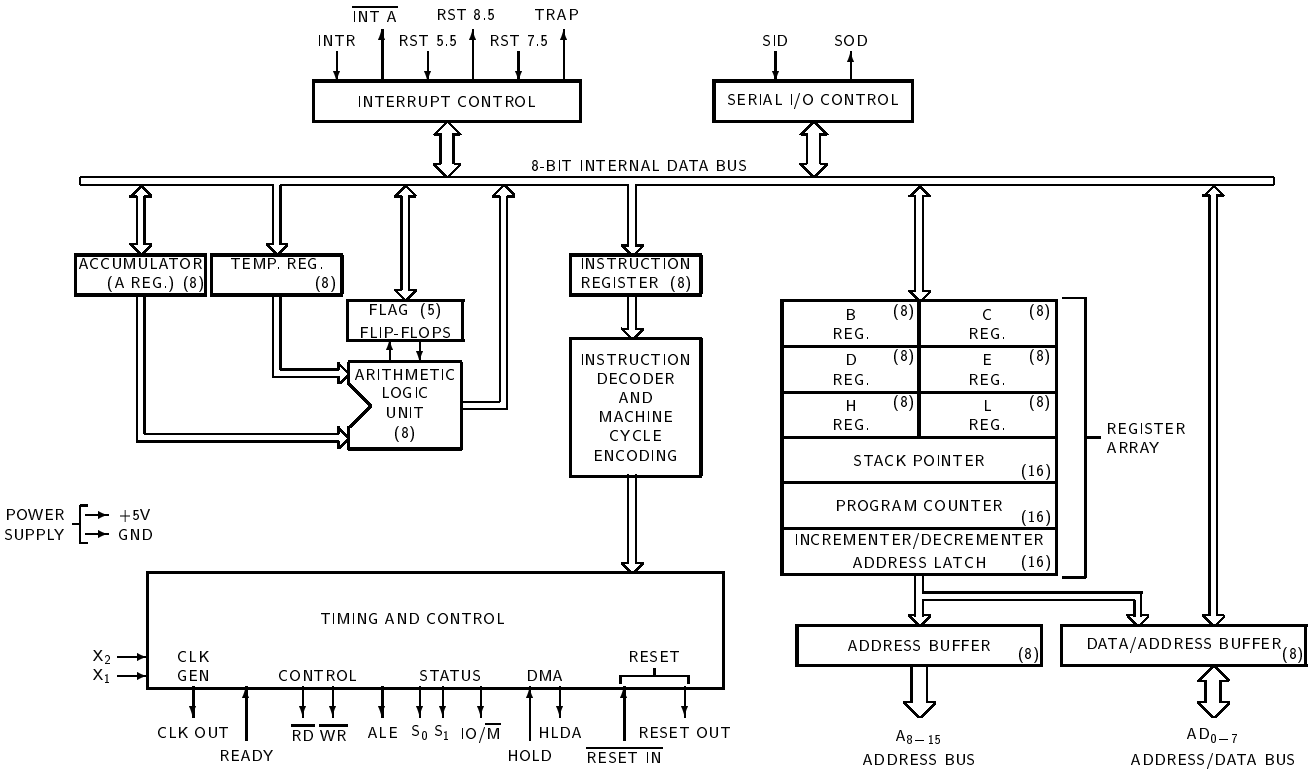
Address bus: a number of lines which give the encoded address of a data word in the storage cells.

Control bus: a few lines which indicate what has to be done, generally read, write, output enable and chip select.

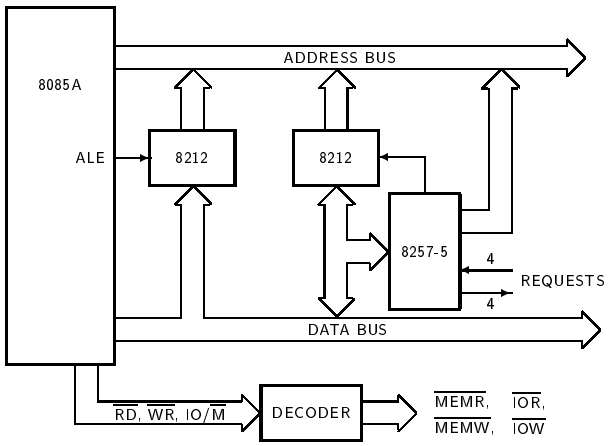
On the other hand, a CPU is also a functional block which uses data, address and control busses to communicate with memory and peripherals:



An example is the Intel 8085A processor:



For now we are only interested in a few control signals and the data and address busses:



The 8257-5 is a DMA controller (direct memory access); this will not be explained in this course.

The 8085A is an 8-bit processor, so the data bus is 8-bits wide. Because of the large amount of other control signals, not all 16 data addressing bits could be assigned an output pin. This was solved by multiplexing the data bus and the lower 8 address bits.

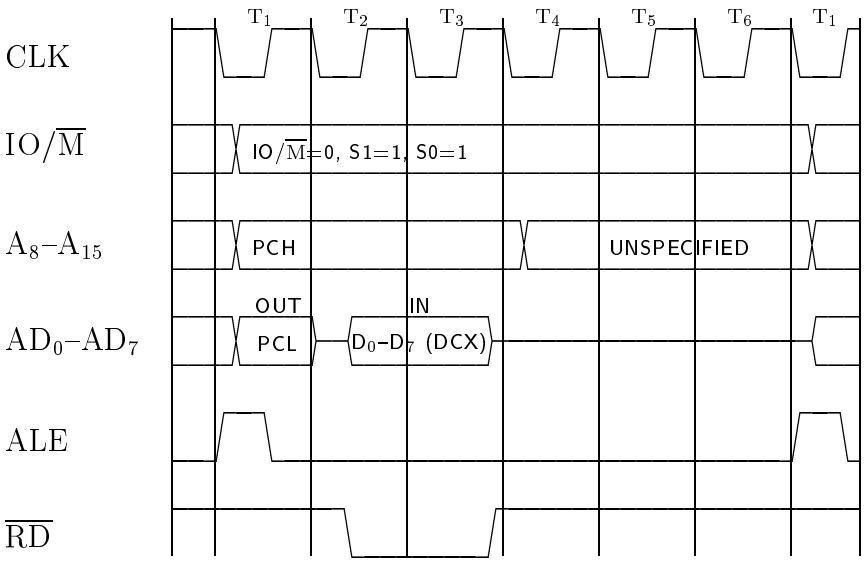
So, AD_{0-7} are combined address and data lines and A_{8-15} are address lines. Before a word in memory can be read or written, the CPU puts the low address byte (A_{0-7}) onto the AD_{0-7} lines and generates then a special signal such that the low-order address byte can be clocked into an 8-bit register or latch. In the previous figure, the latch is indicated

by the block called "8212" (this is one chip) and the special control signal to store the low-order address byte is ALE (address latch enable).

The other signals that we are interested in are $\overline{\text{RD}}$, $\overline{\text{WR}}$ and IO/$\overline{\text{M}}$.

- IO/$\overline{\text{M}}$ indicates whether transfer is going to be to memory (IO/$\overline{\text{M}}$=0) or to other input/output devices (IO/$\overline{\text{M}}$=1).
- $\overline{\text{RD}}$ is a read signal, so if the CPU wants to read it makes $\overline{\text{RD}}$ low
- $\overline{\text{WR}}$ is a write signal, so if the CPU wants to write it makes $\overline{\text{WR}}$ low.

Normally, a control bus has slightly diferent control signals: $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$ and $\overline{\text{IOW}}$. We can use a very simple decoder to generate these signals.
How does it work? Let us take a look at a memory read cycle (in this case of an OF - opcode fetch).



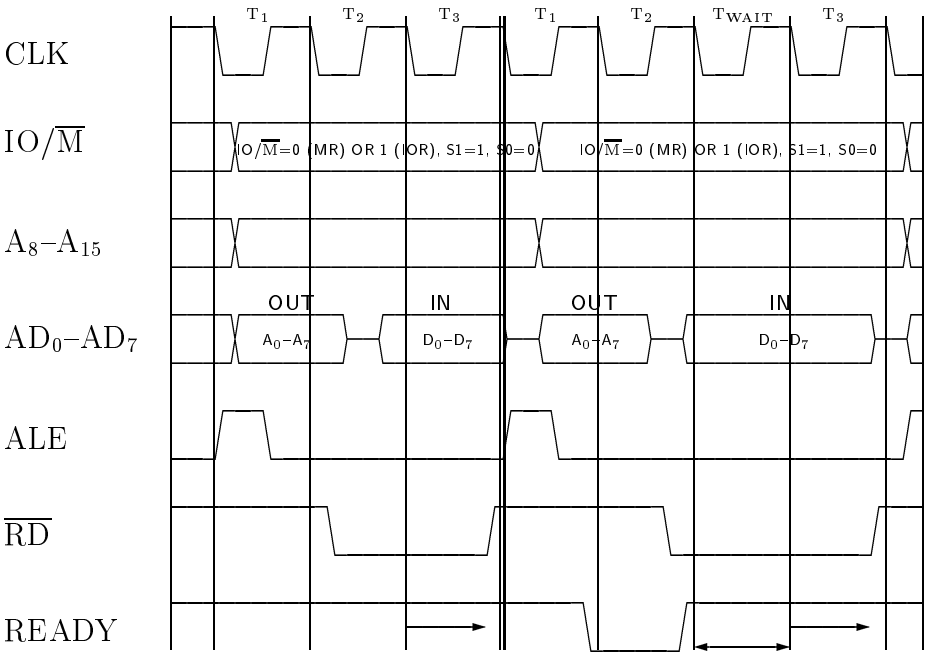
PCH=program counter high
PCL=program counter low

CLK is a periodic system clock and a memory read cycle is defined by a number of states T. Reading a data byte from memory takes normally (if the memory is fast enough) three states: T_1, T_2 and T_3.
In the figure above an OF (opcode fetch) is shown. This is an instruction which takes 6 states: T_1, T_2 and T_3 are concerned with reading an instruction from memory, whereas T_4, T_5 and T_6 are concerned with the interpretation (decoding) of the opcode.

- In T_1, IO/$\overline{\text{M}}$ goes low (memory access), A_8–A_{15} are put on the bus, AD_0–AD_7 which contains the low-order address byte as well and the negative flank of the ALE signal can be used to clock this byte into a latch. After T_1, the address bus is complete!

- In T_2, the $\overline{RD}$ signal goes low, which means that all data bus buffers or transceivers must be directed toward the CPU and the memory must provide the data word with the address indicated.
- In T_3, the $\overline{RD}$ signal goes high, which means that the CPU has read the word into an internal register and the read cycle is finished.

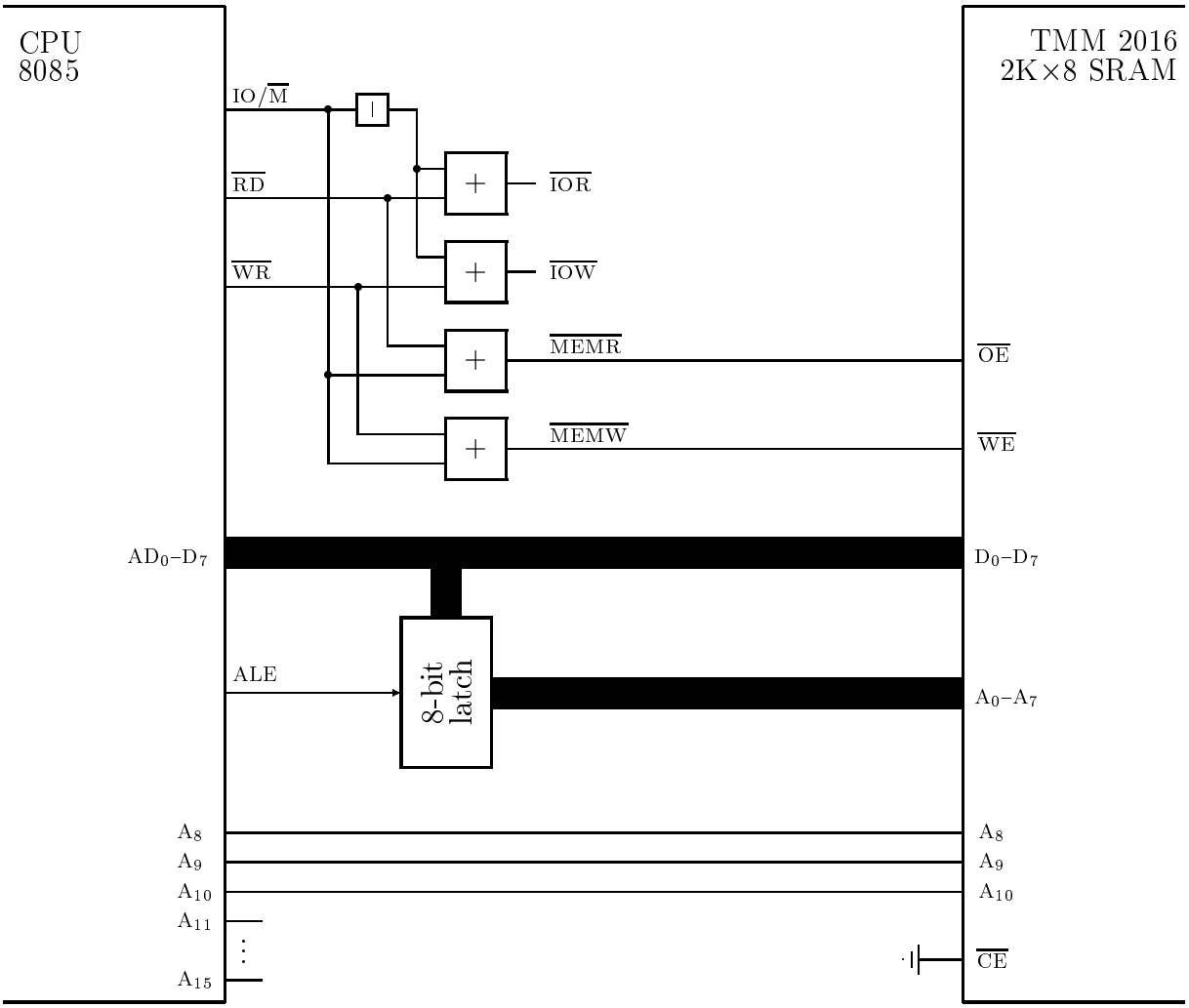
A normal memory (data) read cycle requires only 3 states T_1, T_2 and T_3 (see below left). If the memory is slow or there are many buffers in a system (CPU on a CPU board with its own data/address buffers, memory on another board with its own buffers, ...), the memory board must generate a "ready" signal in order to indicate that the CPU must wait an extra cycle before clocking-in the data on the data bus: memory with 1 wait state.



NOTE: the 8086 processor, which will be discussed in detail in the next semester, is a 16-bit processor but it uses the same principle of address-data multiplexing with an ALE signal.

12.5 CPU-memory interfacing

In this section we will briefly discuss a few examples and explain a few address decoding methods.
The simplest interface is shown below:



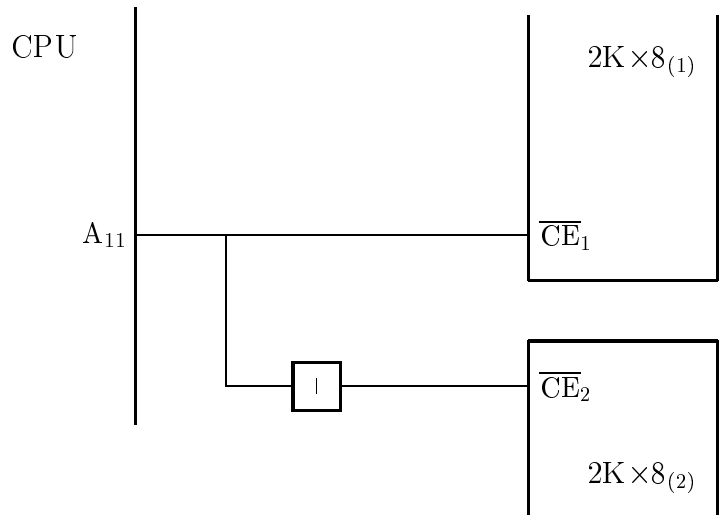
NOTES:

- $\overline{\text{MEMR}}$ goes to $\overline{\text{OE}}$ (=output enable).
- $\overline{\text{MEMW}}$ goes to $\overline{\text{WE}}$ (=write enable).
- $\overline{\text{IOR}}$ and $\overline{\text{IOW}}$ are not used, so $\overline{\text{RD}}$ and $\overline{\text{WR}}$ could be used but such a system cannot communicate with peripherals.
- 2K words means 10 address lines A_0–A_{10}. In this minimum system the entire address decoding is done by the 2016.
- A_{11}–A_{15} are not used at all (the programmer must be sure not to include jump instructions beyond 2K).
- the $\overline{\text{CE}}$ (chip enable) input of the 2016 is always low.

Of course, such a minimum system is not realistic unless we need to design a small controller board for some application, but even then the use of only RAM means *no program to execute*! At least a PROM or EPROM should be added. For other applications we will need to use more RAM and to combine more memory chips.

12.6 Address decoding

In the above example we used a 2K RAM at addresses from 00000–003FFH (hexadecimal) or 0-2047D (decimal). We could use A_{11} to A_{15} to put this memory block at higher addresses. Or, to add a second chip in order to increase the memory from 2K to 4K, occupying 00000–004FFH. To do this, we need to exploit address bit A_{11}:

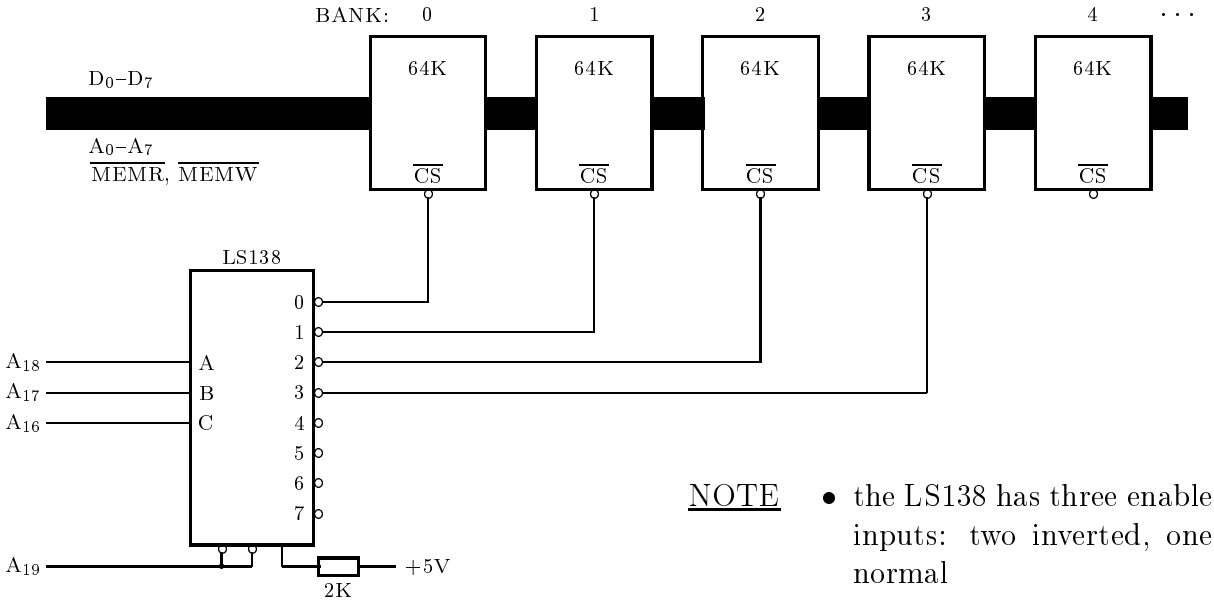


NOTES:
- all other lines go to both chip 1 and chip 2, to the same inputs and outputs.
- A_{11} to $\overline{CE}_1$ means chip 1 contains addresses 00000-003FFH.
  $\overline{A_{11}}$ to $\overline{CE}_2$ means chip 2 contains addresses 00400-004FFH.

  In other words: A_{11} has been decoded in order to address more memory and to address more memory chips.

Again, this example is not quite realistic because we still need more memory! So we also need to use A_{12} and A_{13} and... and each chip needs its specific $\overline{CE}$ signal in the memory space that it will occupy: address decoding.
Instead of using simple circuits consisting of a few gates, special circuits are then employed.

Example
- use 64K chips (64K requires 16 address lines: A_0–A_{15}.
  Hence, A_{16} and A_{17} must be decoded to use 4×64K
- use a 74LS138 (3-to-8 line decoder)



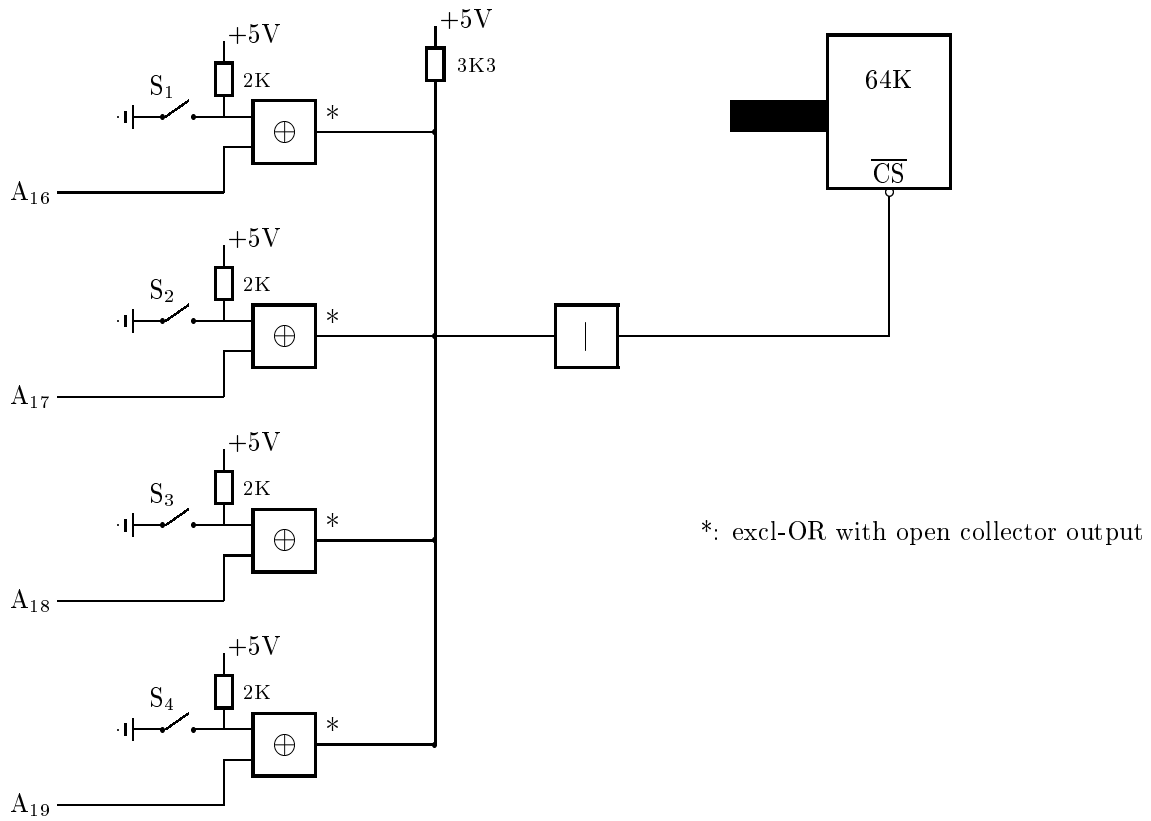
NOTE

- the LS138 has three enable inputs: two inverted, one normal
- the 8086 has 20 address lines, so A_{19} is the highest

Bank 0: 00000–0FFFFH
Bank 1: 10000–1FFFFH
Bank 2: 20000–2FFFFH
⋮
Bank 7: 70000–7FFFFH

Using one LS138 and 64K chips without using A_{19} allows to address 80000H words (512Kb). Using also A_{19}, the 8086 can address 1Mb at most.

Sometimes we may need to put a chip in a specific memory range: for example when we make a PC board which contains battery-backed CMOS SRAMS (memory chips with a very low power consumption; a small battery allows to keep the information when the computer is shut down). If we want to put this board in different PC's with certain amounts of memory, we might adapt the board address space to the host address space. One simple way to do so is by using exclusive-ORs:



When all four switches are open, A_{16}–A_{19} must equal 0000 to cause all four excl-OR open collector outputs to be high, i.e. $\overline{CS}$ becomes 1: disabled. This decoder (actually comparator) allows to put the 64K block to any of the 16 possible 64K slots.

To put the block in the space beginning at C0000H=$\overbrace{1100}^{CH}.\overbrace{0000}^{0H}\ldots$ $S_4S_3S_2S_1$ must be 1100 (only S_4 and S_3 closed).

Instead of using separate switches and exclusive-ORs we can use "dip-switches" and a comparator.