8086 - 80386SX 16-bit Memory Interface

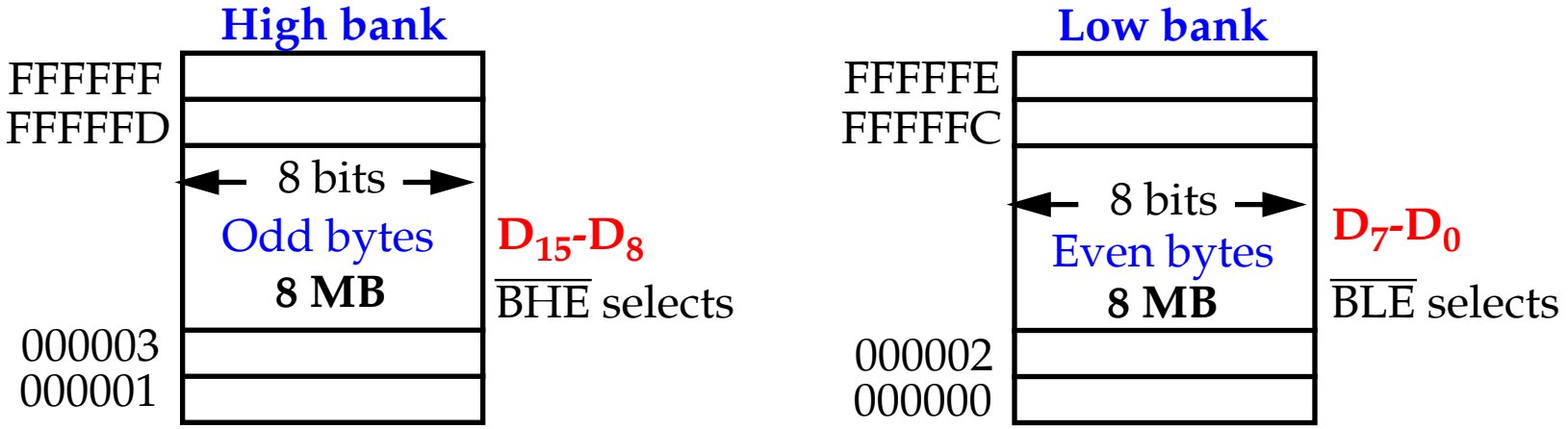
These machines differ from the 8088/80188 in several ways:

- The data bus is *16-bits* wide.
- The IO/$\overline{M}$ pin is replaced with M/$\overline{IO}$ (8086/80186) and $\overline{MRDC}$ and $\overline{MWTC}$ for 80286 and 80386SX.
- $\overline{BHE}$, **Bus High Enable,** control signal is added.
- Address pin A_0 (or $\overline{BLE}$, **Bus Low Enable**) is used differently.

The 16-bit data bus presents a new problem:

The microprocessor must be able to read and write data to any 16-bit location in addition to any 8-bit location.

The data bus and memory are divided into banks:



8086 - 80386SX 16-bit Memory Interface

BHE and BLE are used to select one or both:

$\overline{\text{BHE}}$	$\overline{\text{BLE}}$	Function
0	0	Both banks enabled for 16-bit transfer
0	1	High bank enabled for an 8-bit transfer
1	0	Low bank enabled for an 8-bit transfer
1	1	No banks selected

Bank selection can be accomplished in two ways:

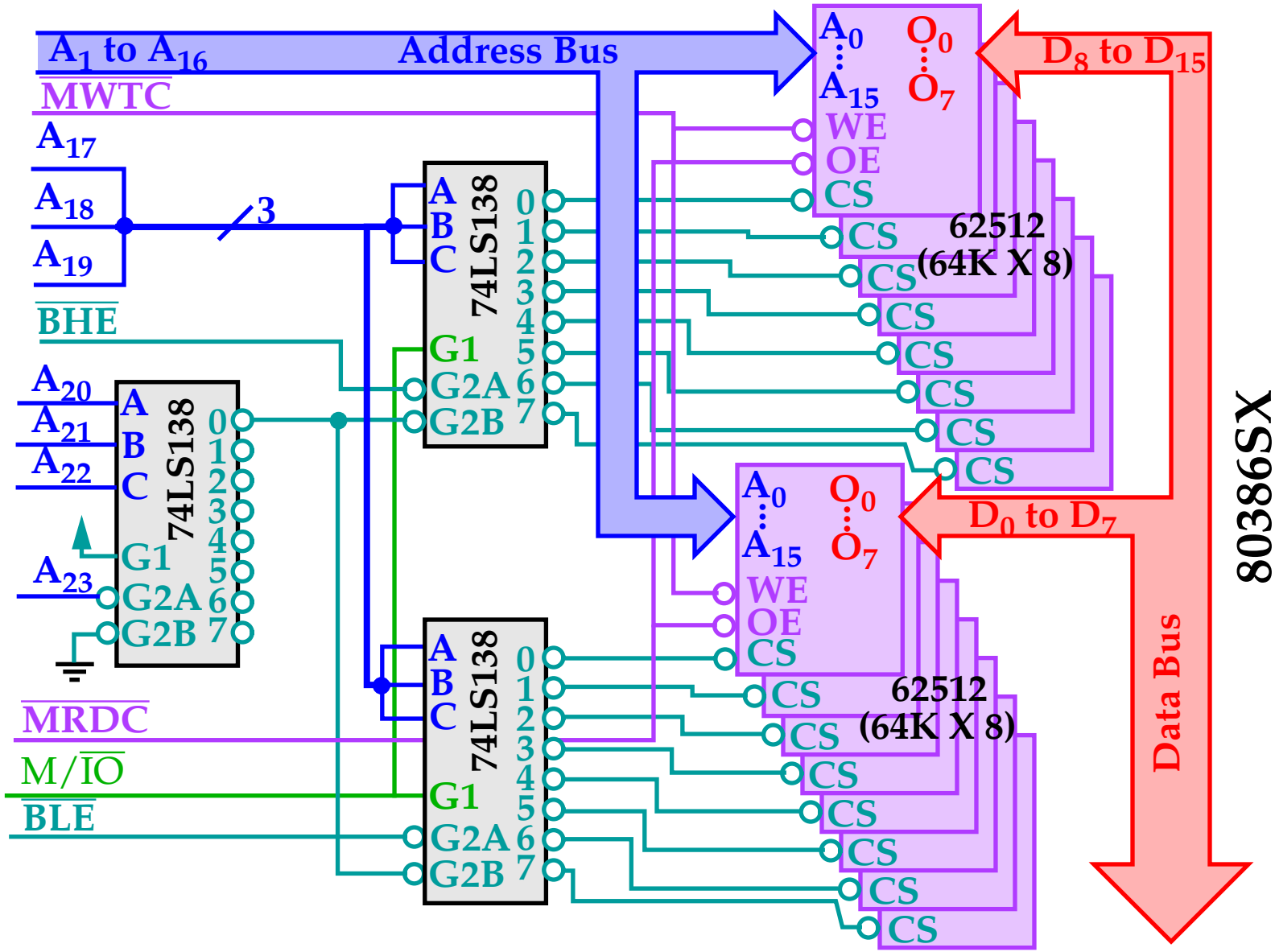
- Separate write decoders for each bank (which drive $\overline{\text{CS}}$).
- A separate write signal (strobe) to each bank (which drive $\overline{\text{WE}}$).

  Note that 8-bit read requests in this scheme are handled by the microprocessor (it selects the bits it wants to read from the 16-bits on the bus).

There does not seem to be a big difference between these methods although the book claims that there is.

Note in either method that A_0 does not connect to memory and bus wire A_1 connects to memory pin A_0, A_2 to A_1, etc.

80386SX 16-bit Memory Interface (Separate Decoders)



Memory Interfaces

See text for *Separate Write Strobe* scheme plus some examples of the integration of EPROM and SRAM in a complete system.

It is just an application of what we've been covering.

80386DX and 80486 have *32-bit* data buses and therefore 4 banks of memory.

32-bit, *16-bit* and *8-bit* transfers are accomplished by different combinations of the bank selection signals $\overline{\text{BE3}}$, $\overline{\text{BE2}}$, $\overline{\text{BE1}}$, $\overline{\text{BE0}}$.

The Address bits A_0 and A_1 are used within the microprocessor to generate these signals.

They are *don't cares* in the decoding of the 32-bit address outside the chip (using a PLD such as the **PAL 16L8**).

The high clock rates of these processors usually require **wait states** for memory access.

We will come back to this later.

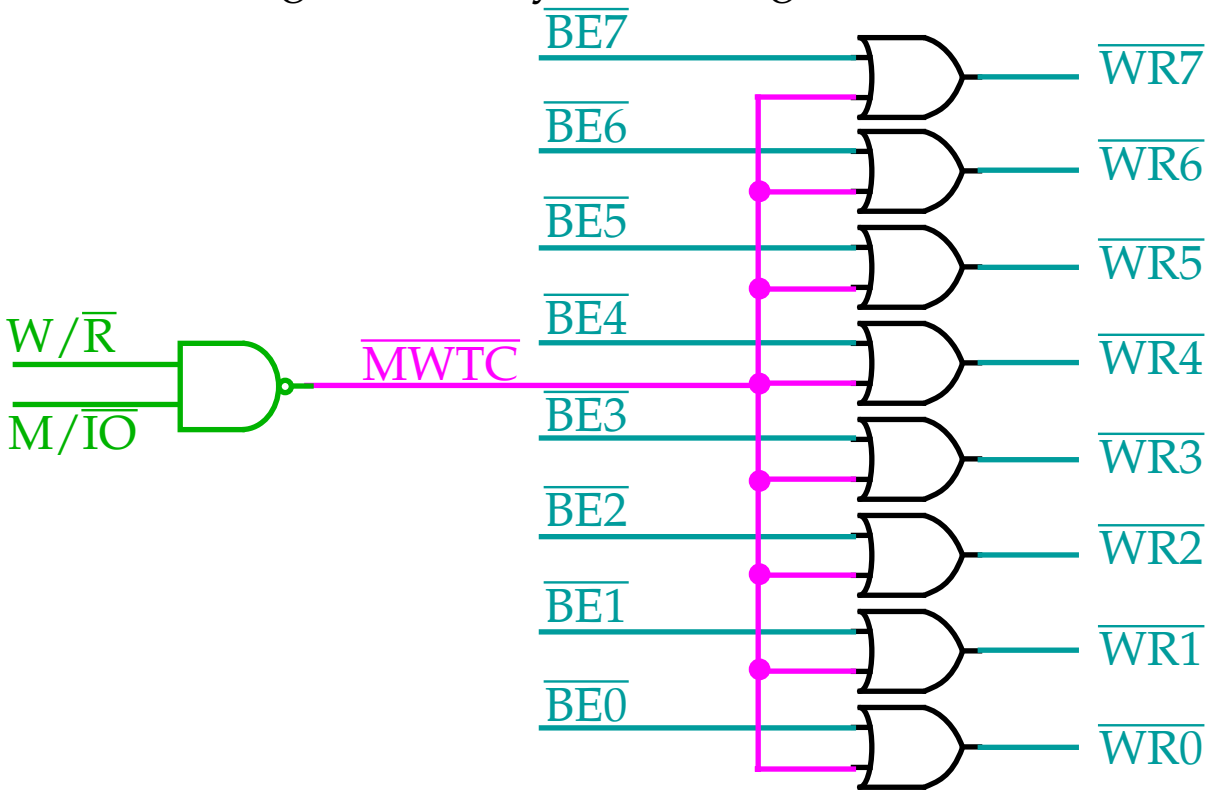
Pentium Memory Interface

The Pentium, Pentium Pro, Pentium II and III contain a 64-bit data bus.

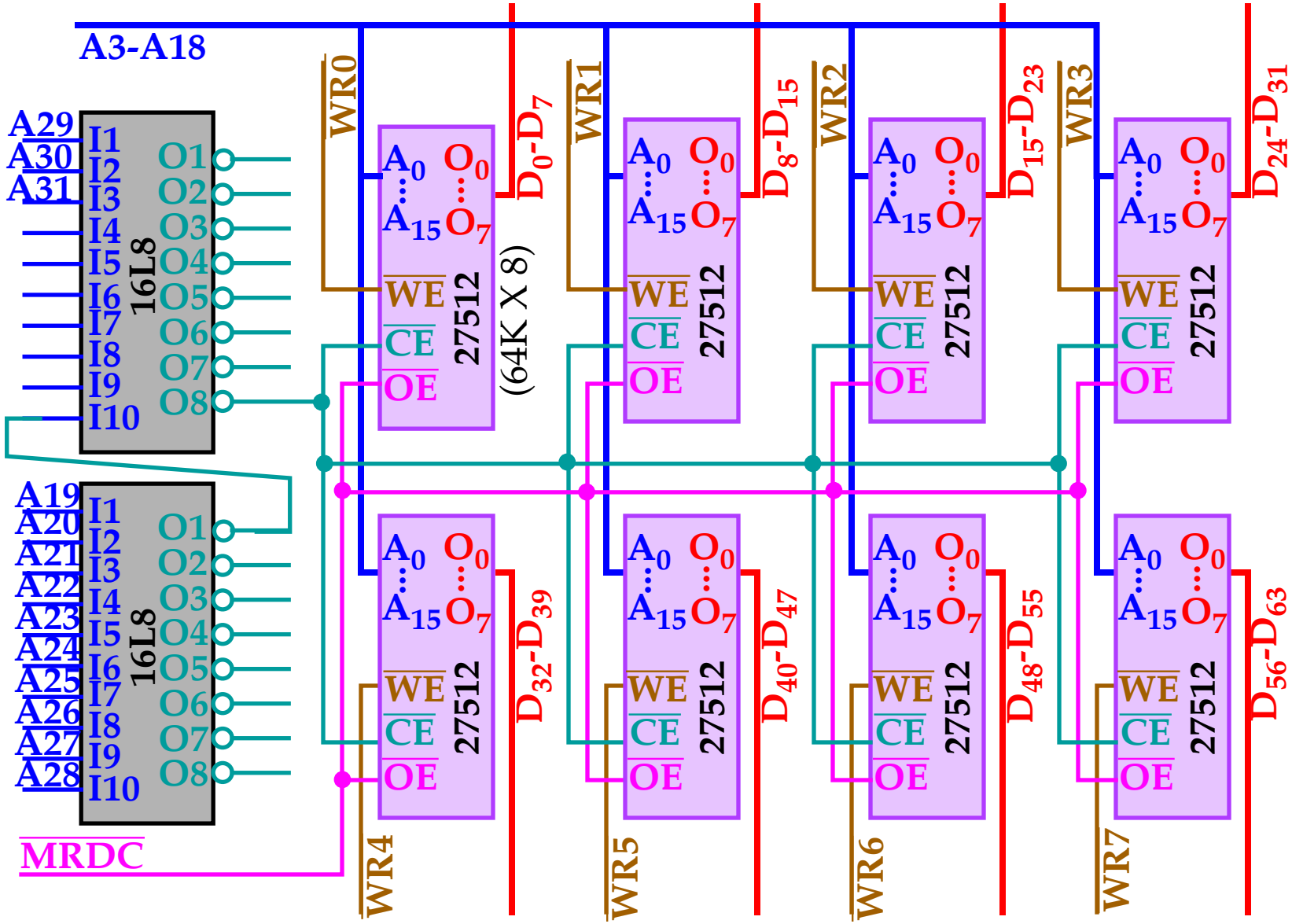
Therefore, 8 decoders or 8 write strobes are needed as well as 8 memory banks.

The write strobes are obtained by combining the bank enable signals ($\overline{BEx}$) with the $\overline{MWTC}$ signal.

$\overline{MWTC}$ is generated by combining the M/$\overline{IO}$ and W/$\overline{R}$ signals.

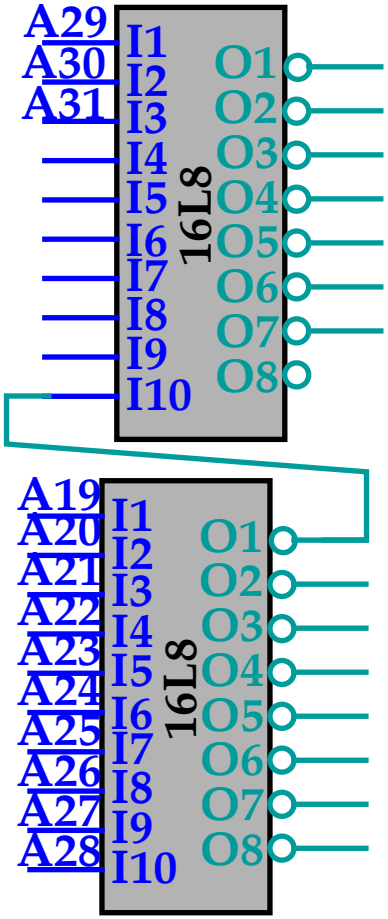


Pentium Memory Interface



Pentium Memory Interface

In order to map previous memory into addr. space *FFF80000H-FFFFFFFH*



;**pins** 1 2 3 4 5 6 7 8 9 10
A29 A30 A31 NC NC NC NC NC NC GND

;**pins** 11 12 13 14 15 16 17 18 19 20
U2 CE NC NC NC NC NC NC NC VCC

Equations:

/CE = /U2 * A29 * A30 * A31

;**pins** 1 2 3 4 5 6 7 8 9 10
A19 A20 A21 A22 A23 A24 A25 A26 A27 GND

;**pins** 11 12 13 14 15 16 17 18 19 20
A28 U2 NC NC NC NC NC NC NC VCC

Equations:

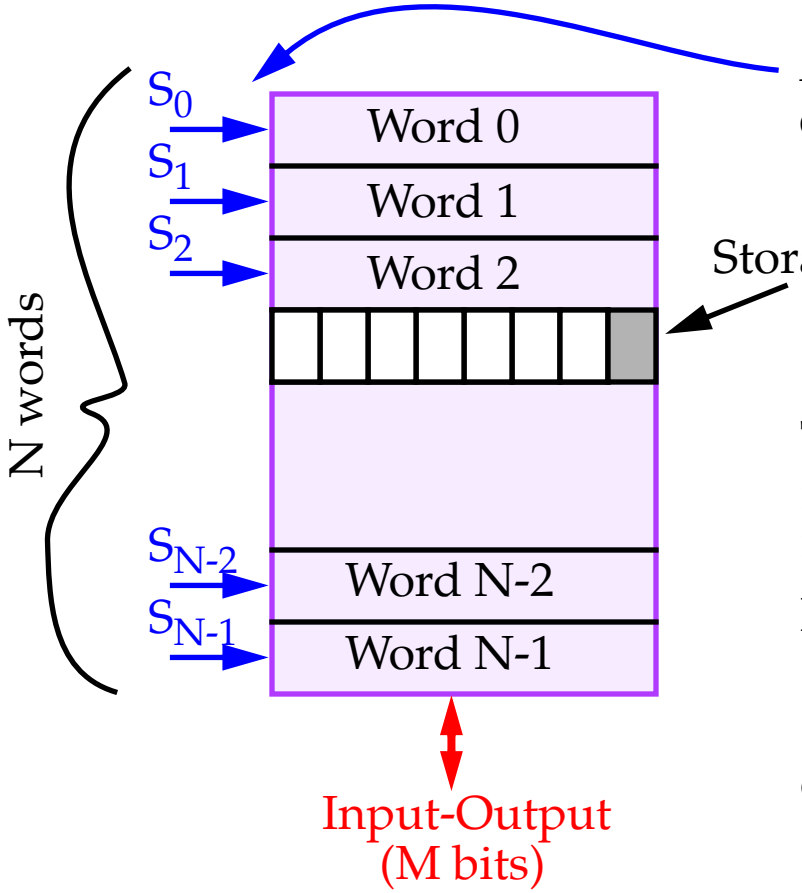
/U2 = A19 * A20 * A21 * A22 * A23 * A24 * A25 *
A26 * A27 * A28

Use a **16L8** to do the $\overline{WR0}$ - $\overline{WR7}$ decoding using $\overline{MWTC}$ and $\overline{BE0}$ - $\overline{BE7}$.

See the text -- Figure 10-35.

Memory Architecture

In order to build an *N-word* memory where each word is *M bits* wide (typically 1, 4 or 8 bits), a straightforward approach is to stack memory:



A word is selected by setting exactly **one** of the select bits, S_x, high.

Storage cell

This approach works well for small memories but has problems for large memories.

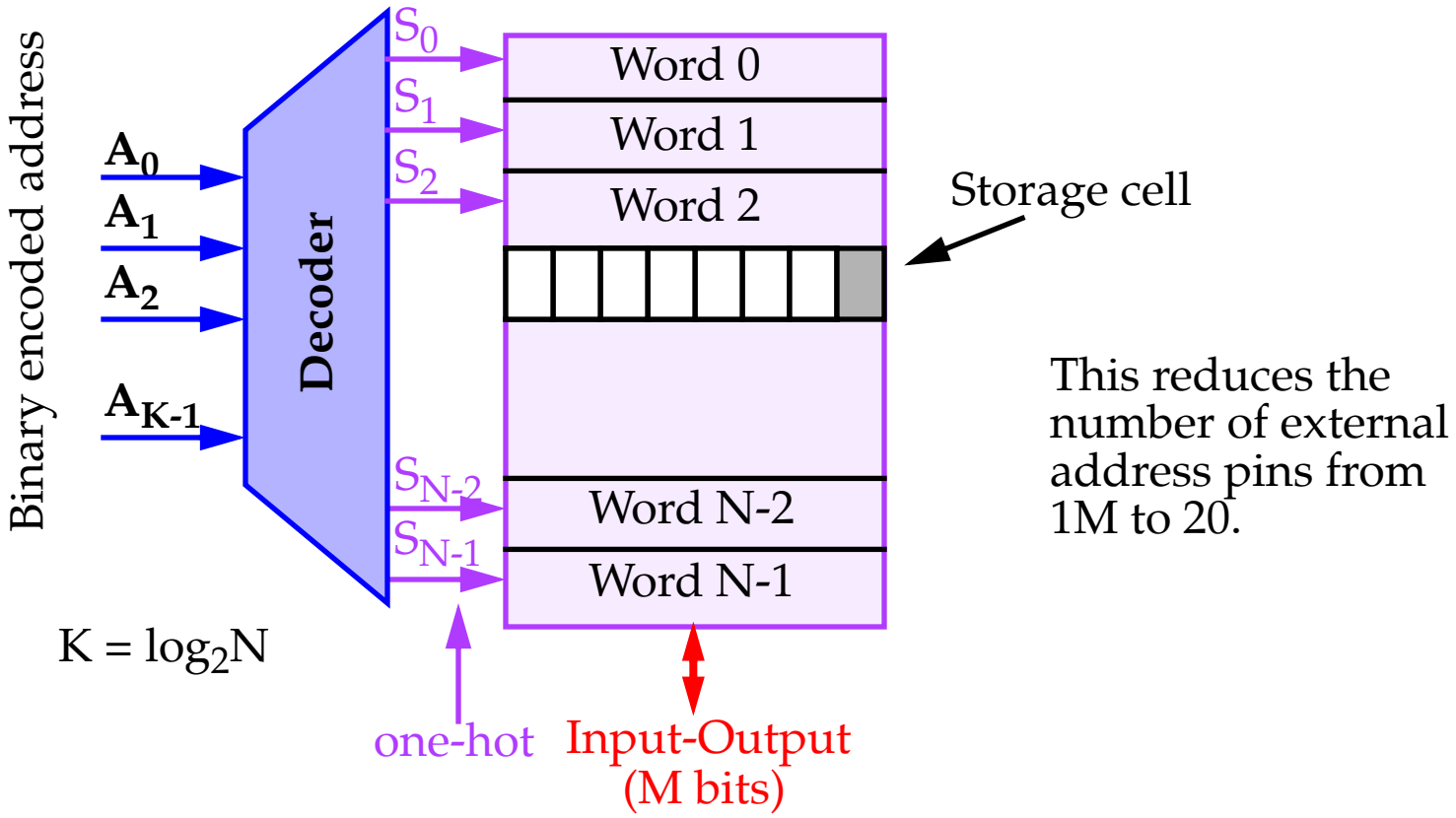
For example, to build a 1Mword (where word = 8 bits) memory, requires 1M select lines, provided by some off-chip device.

This approach is not practical.

What can we do?

Memory Architecture

Add a decoder to solve the package problem:



This reduces the number of external address pins from 1M to 20.

This does not address the **memory aspect ratio** problem:

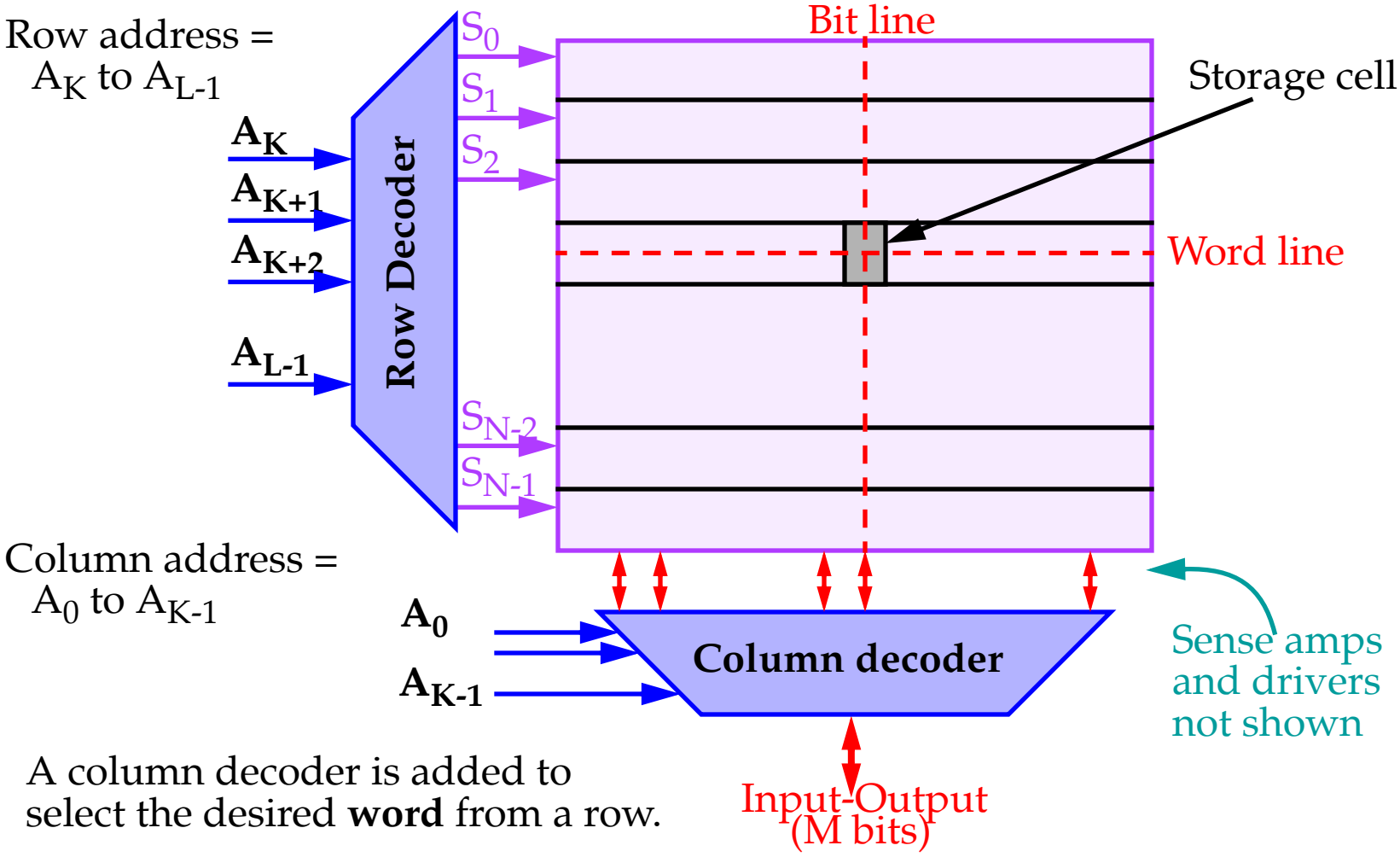
The memory is 128,000 time higher than wide ($2^{20}/2^{3}$) !

Besides the bizarre shape factor, the design is *extremely slow* since the vertical wires are VERY long (delay is at least linear to length).

Memory Architecture

The vertical and horizontal dimensions are usually very similar, for an aspect ratio of *unity*.

Multiple words are stored in each row and selected simultaneously:



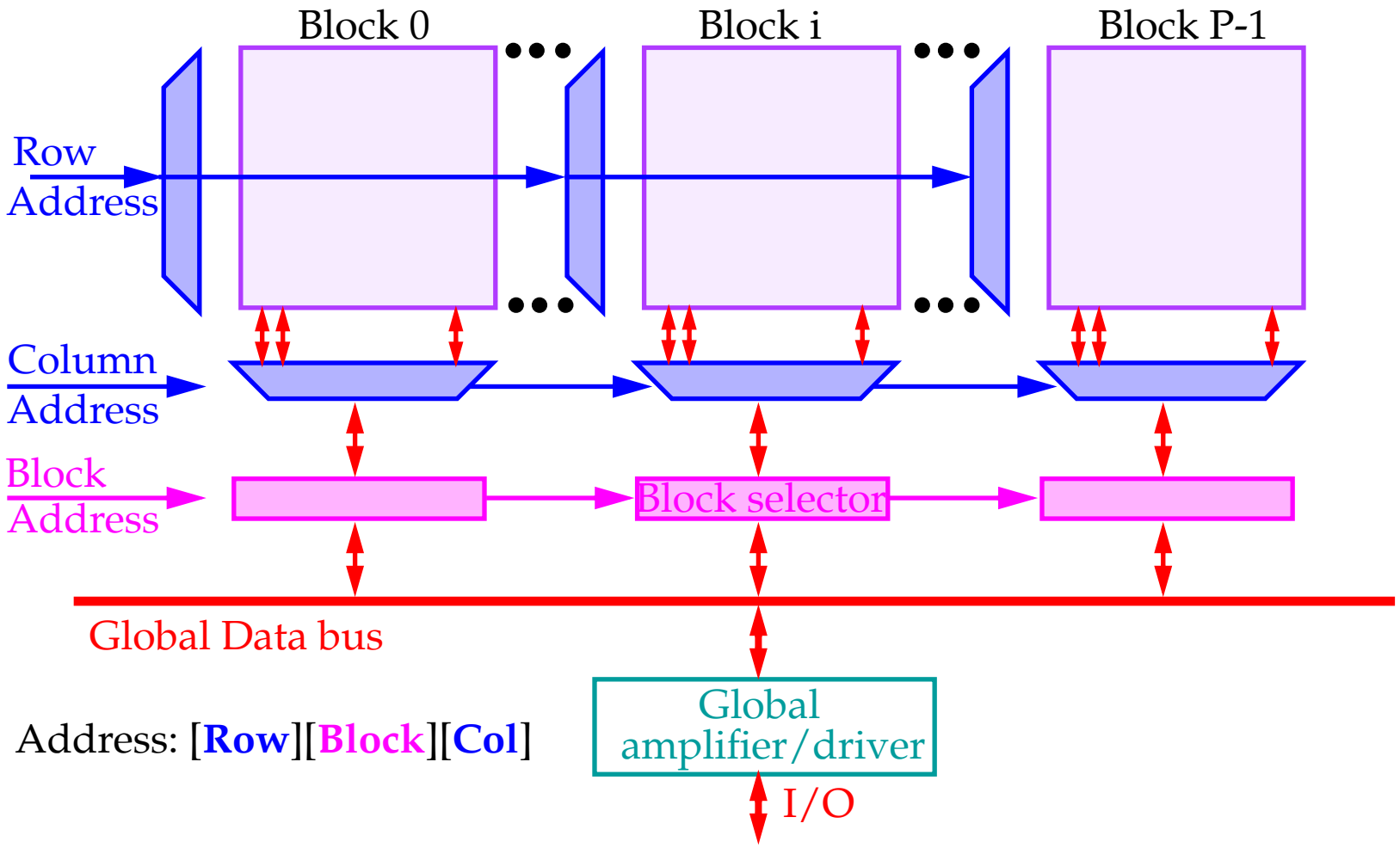
A column decoder is added to select the desired **word** from a row.

Memory Architecture

This strategy works well for memories up to 64 Kbits to 256 Kbits.

Larger memories start to suffer excess delay along bit and word lines.

A **third dimension** is added to the address space to solve this problem:



Dynamic RAM

DRAM requires refreshing every *2* to *4 ms*.

Refreshing occurs automatically during a read or write.

Internal circuitry takes care of refreshing cells that are not accessed over this interval.

This special refresh occurs **transparently** while other memory components operate and is called *transparent refresh* or *cycle stealing*.

A $\overline{\text{RAS}}$-only cycle strobes a row address into the DRAM, obtained by 7- or 8-bit binary counter.

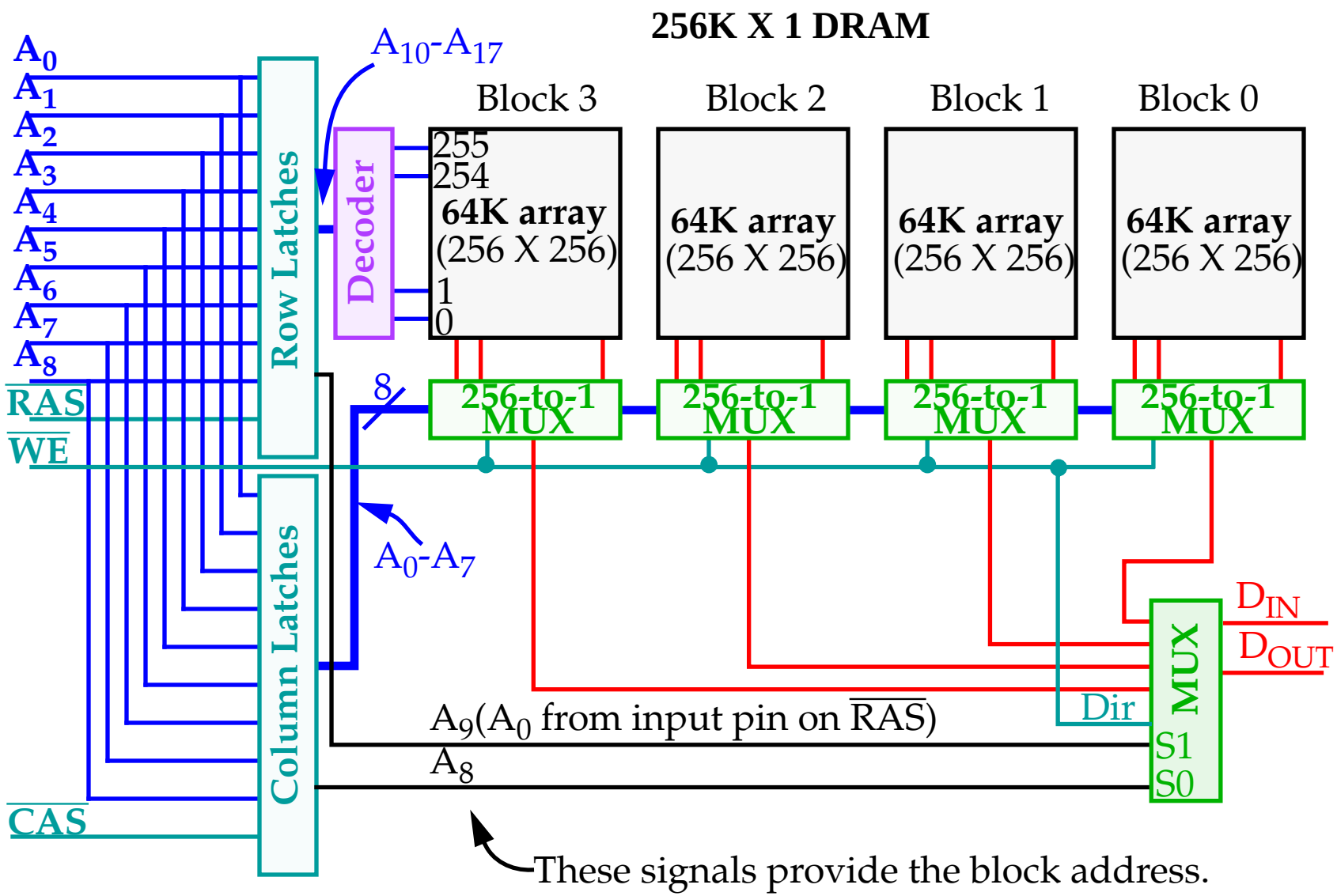
The capacitors are recharged for the selected row by reading the bits out internally and then writing them back.

For a *256K X 1* DRAM with 256 rows, a refresh must occur every *15.6us* (4ms/256).

For the 8086, a read or write occurs every *800ns*.

This allows **19** memory reads/writes per refresh or **5%** of the time.

Dynamic RAM


256K X 1 DRAM
A0
A1
A2
A3
A4
A5
A6
A7
A8
RAS
WE
CAS
Row Latches
Column Latches
Decoder
A10-A17
Block 3
Block 2
Block 1
Block 0
255
254
64K array
(256 X 256)
1
0
8
256-to-1 MUX
A0-A7
DIN
DOUT
Dir
MUX
S1
S0
A9(A0 from input pin on RAS)
A8
These signals provide the block address.

EDO and SDRAM Memory

Extended Data Output memory:

Any memory access in an **EDO** memory (including a refresh) stores the 256 bits in a set of latches.

Any subsequent access to bytes in this set are **immediately** available (*without the decode time* and therefore wait states).

This works well because of the principle of spatial locality, and improves system performance by 15 to 25 % !

Synchronous Dynamic RAM:

Access times are *10ns* (for use with 66MHz bus) and *8ns* (for use with 100MHz bus).

Standard DRAM access times are *60ns*.

However, these access times *only apply* to the **2nd**, **3rd** and **4th** *64-bit* reads -- the first takes the same time as a standard DRAM.

EDO and SDRAM Memory

Synchronous Dynamic RAM:

However, this improves performance again, particularly for reads into cache block sizes of *256 bits*.

For example, 256 bit transfer takes **three** bus cycles for the *first read* and **three** for the *next three* 64-bit words, for a total of **7** bus cycles.

This contrasts with the 3*4 or **12** bus cycles for DRAM or EDO.

Measurements show about a **10%** increase in performance.

DRAM Controllers:

A DRAM controller is usually responsible for address multiplexing and generation of the DRAM control signals.

These devices tend to get very complex.
We will focus on a simpler device, the *Intel 82C08*, which can control **two** banks of *256K X 16* DRAM memories for a total of 1 MB.

DRAM Controllers:

Intel 82C08:

Microprocessor bits A_1 through A_{18} (18 bits) drive the 9 **Address Low** (AL) and 9 **Address High** (AH) bits of the *82C08*.

9 of each of these are strobed onto the address wires A_0 through A_8 to the memories.

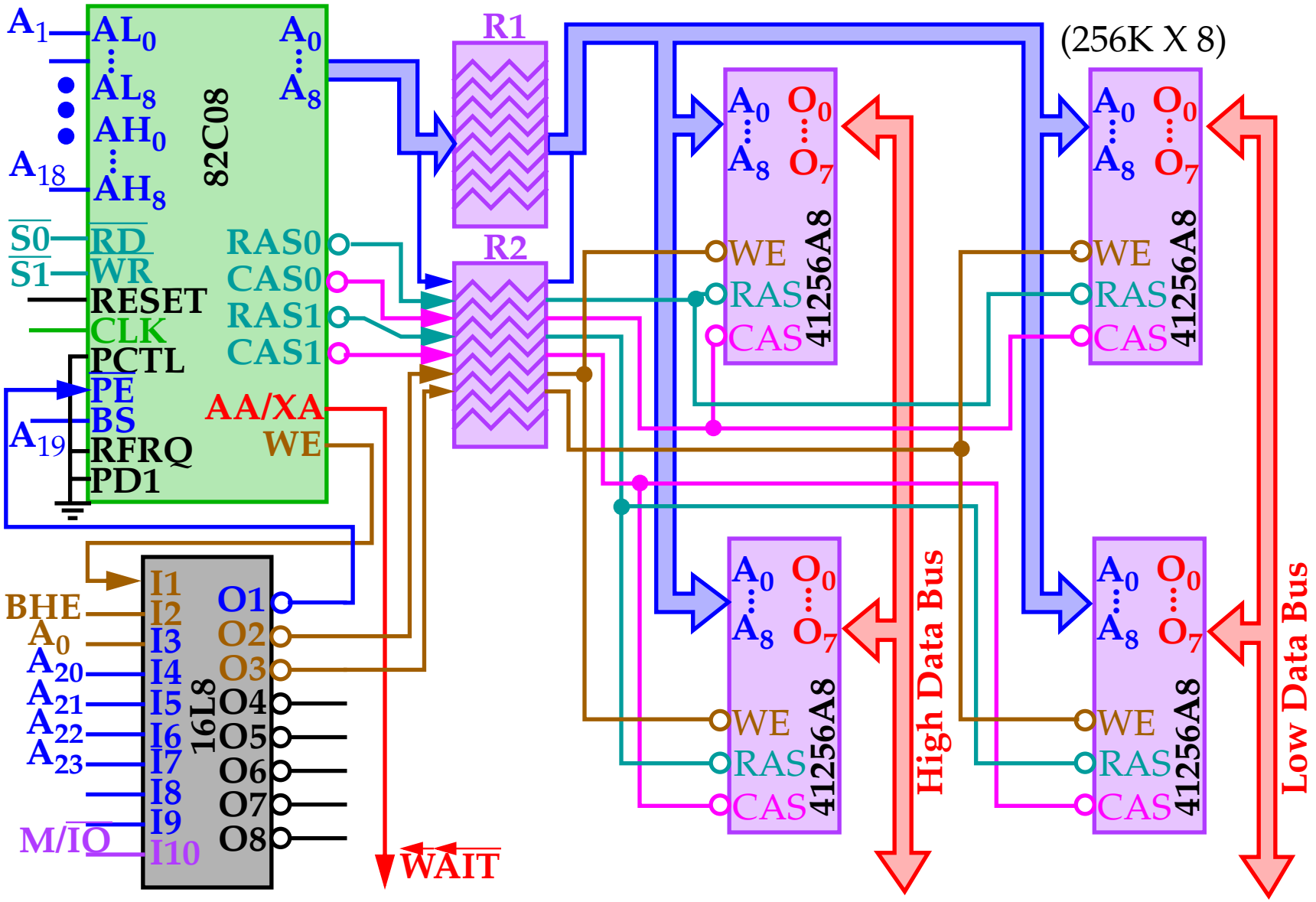
Either RAS0/CAS0 or RAS1/CAS1 are strobed depending on the address.

This drives a *16-bit* word onto the High and Low data buses (if WE is low) or writes an 8 or 16 bit word into the memory otherwise.

WE (from the 82C08), BHE and A_0 are used to determine if a write is to be performed and which byte(s) (low or high or both) is to be written.

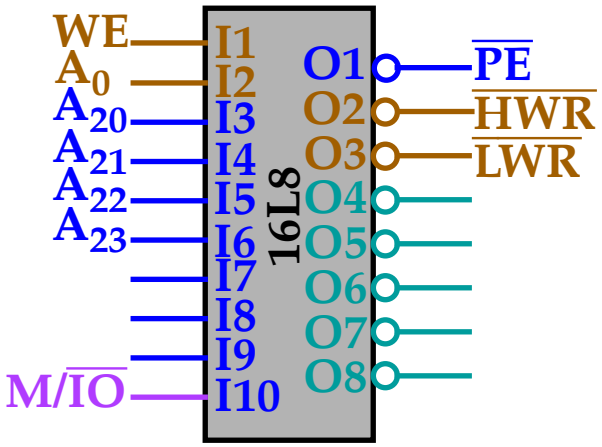
Address bit A_{20} through A_{23} along with M/$\overline{IO}$ enable these memories to map onto 1 MByte range (*000000H-0FFFFFH*).

DRAM Controllers



DRAM Controllers:

16L8 Programming:




```
;pins 1    2    3    4    5    6    7    8    9    10
      WE BHE  A0   A20  A21  A22  A23  NC   NC   GND
;pins 11   12   13   14   15   16   17   18   19   20
      MIO CE   NC   NC   NC   NC   LWR  HWR  PE   VCC
```

Equations:

/LWR = /A0 * /WE

/HWR = /BHE * /WE

/PE = /A20 * /A21 * /A22 * /A23 * MIO