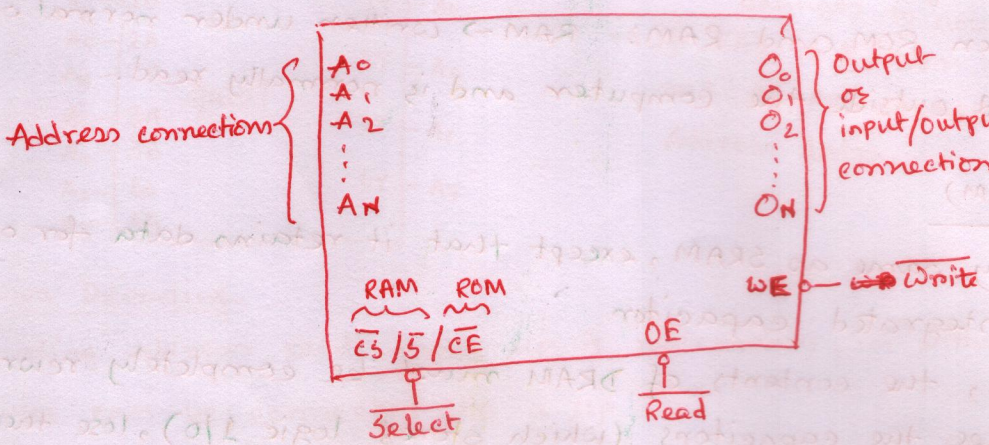
L-2

Memory Interface

- 4 common types of memory: read-only memory (ROM), flash memory (EEPROM), static random access memory (SRAM), and dynamic random access memory (DRAM).
- Pin connections common to all memory devices: address inputs, data outputs (input/outputs), some type of selection input, and at least one control input used to select a read or write operation



- Control connections:

  ROM → Only one control input ($\overline{OE}$ or $\overline{G}$) (Output Enable) (Gate)

  RAM → One ($R/\overline{W}$) or two ($\overline{WE}/\overline{W}$ and $\overline{OE}/\overline{G}$) control input do not get activated at the same time

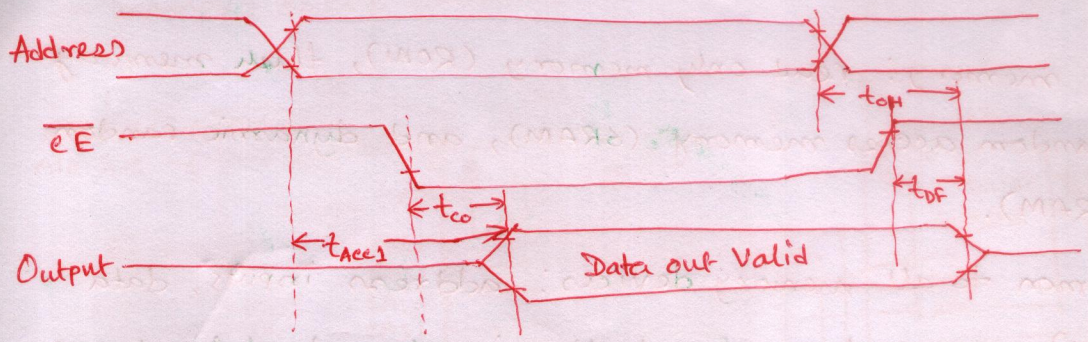
ROM Memory (nonvolatile memory)

- Permanently stores programs and data that are resident to the system and must not change when power supply is disconnected (permanently programmed). [for example BIOS].
- EPROM (Erasable Programmable ROM): programmed using a device called EPROM programmer; erasable if exposed to high-intensity UV light for about 20 minutes or less
- PROM (Programmable ROM): programmed by burning open tiny ~~NI-chrome~~ NI-chrome or silicon oxide fuses; once programmed, it cannot be erased.
- Read Mostly Memory (RMM) or flash memory or EEPROM (Electrically Erasable Programmable ROM) or EAROM (Electrically Alterable ROM) or NOVRAM (Nonvolatile RAM): Electrically erasable, however, needs more time to erase than a normal RAM

⇒ Delays in operation of an EPROM:

t_{acc1}: Address to output delay; t_{co}: Chip select to ~~Output~~ Output delay

t_{oh}: Address to Output hold; t_{DF}: Chip Deselect to Output Float



Timing Diagram of EPROM

Static memory or Static RAM (SRAM) or Volatile memory

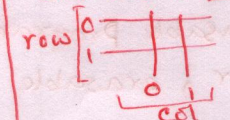
- Retain data as long as DC power is applied (no data without power)
- Difference between ROM and RAM: RAM → written under normal operation; ROM → Programmed outside the computer and is normally read.

Dynamic RAM (DRAM)

- DRAM is essentially same as SRAM, except that it retains data for only 2 or 4 ms on an integrated capacitor
- After 2 or 4 ms, the contents of DRAM must be completely rewritten (refreshed) because the capacitors (which stores logic 1/0), lose their charges.
- Refreshing also occurs during a write, a read, or during a special refresh cycle.
- Have much larger sizes compared to SRAM. Its obvious disadvantage is requirement of many address pins. To reduce the number of address pins, the address inputs are multiplexed.
- Example of multiplexed address pins: 64K X 4 DRAM

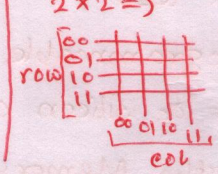
A0–A7: Address Inputs
$\overline{CAS}$: Column Address Strobe
DQ1–DQ4: Data-In/Data-Out
$\overline{G}$: Output Enable
$\overline{RAS}$: Row Address Strobe
V_{DD}: +5V Supply
V_{SS}: Ground
$\overline{W}$: Write Enable

From own: 1X1 bits ⇒ [Orientation in row & col retains same # of bits]



$2 \times 2 = 4 = 2^2$

2X2 ⇒

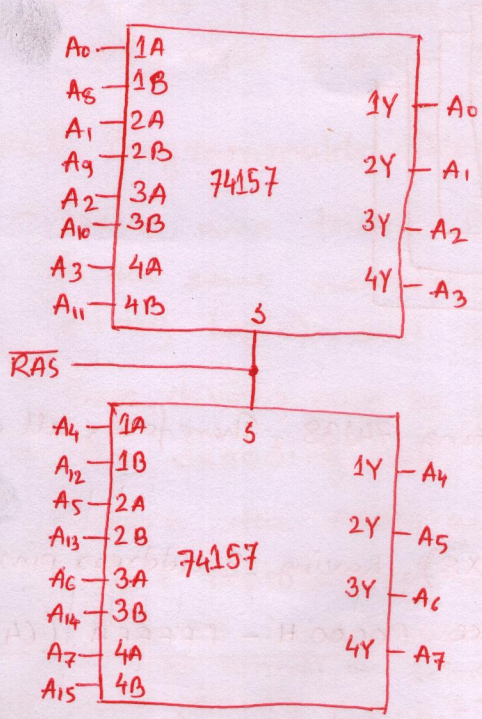


$4 \times 4 = 2^2 \times 2^2 = 2^4$

⇒ Here, for 64K, 16 address pins are required; however, 8 address pins are used through multiplexing.

- At first, A_0–A_7 are stored in internal row latch as row address through enabling $\overline{RAS}$
- Next, A_8–A_{15} are stored in internal column latch as column address through enabling $\overline{CAS}$

⇒ Address multiplexer of 64K X4



→ If $\overline{RAS}$ is 0, then the A pins ~~are~~ get connected and A0-A7 are stored in Internal Row Address latch

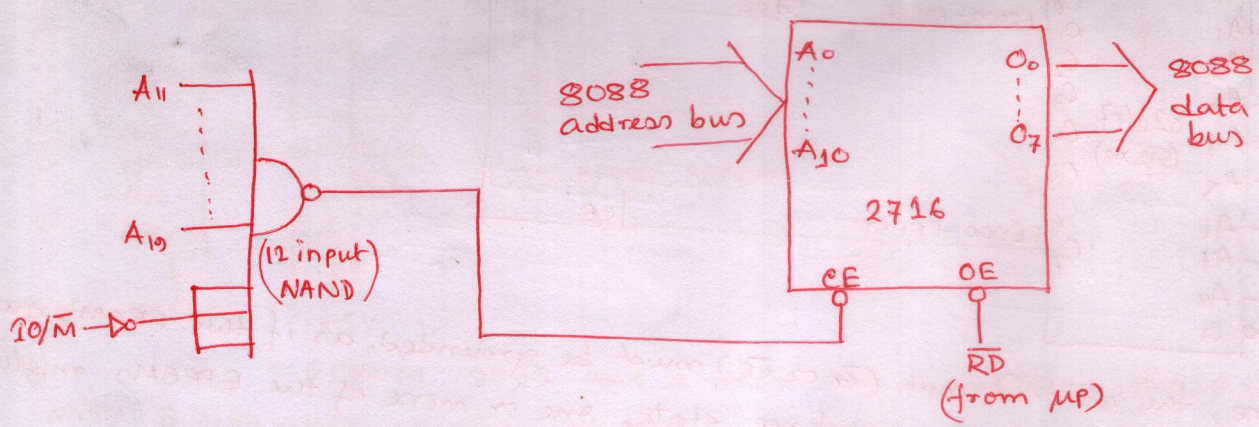
⇒ The Internal Row Address latch is edge triggered, and therefore, the row address gets captured into the latch before the address changes to column address

→ If $\overline{RAS}$ is 1, then the B pins ~~are~~ get connected and A8-A15 are stored in Internal column Address latch.

Address Decoding

- Decoding address sent from µp is necessary, as without it, only one memory device can be connected to a µp.
- Another reason of decoding is possible mismatch (or difference) between the # of address connections in µp and memory
  - → Example: A 2KX8 EPROM (2716) has 11 address pins, whereas the µp 8088 has 20 address pins.

⇒ Simple NAND Gate Decoder:

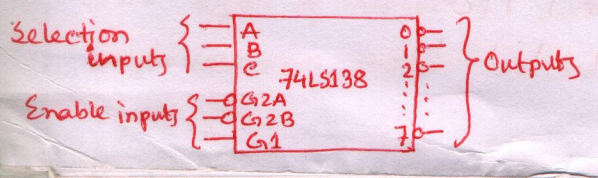


→ Here, the 2K EPROM is decoded at memory address locations FF800H - FFFFFH. (The FF8 corresponds to 1's in nine left positions)

→ In such a decoding, one NAND Gate Decoder selects one 2K EPROM out of many 2K EPROMs that appear in 1M address space

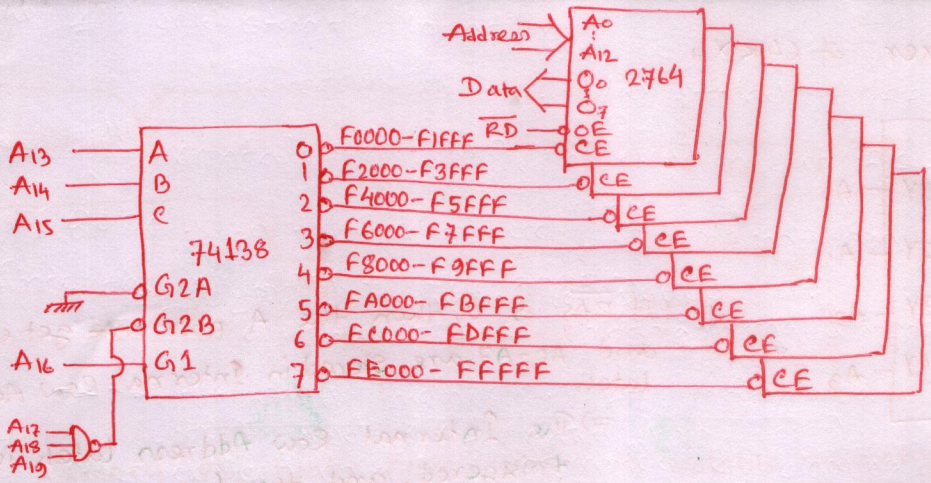
→ The obvious disadvantage is that each EPROM needs one NAND gate decoder

⇒ 3-to-8 Line Decoder (74LS138)



⇒ Input: Enable = 001 and Selection input (CBA) = n
Output: nth output = 0, all rest outputs = 1

⇒ Input: Enable ≠ 001
Output: all outputs = 1



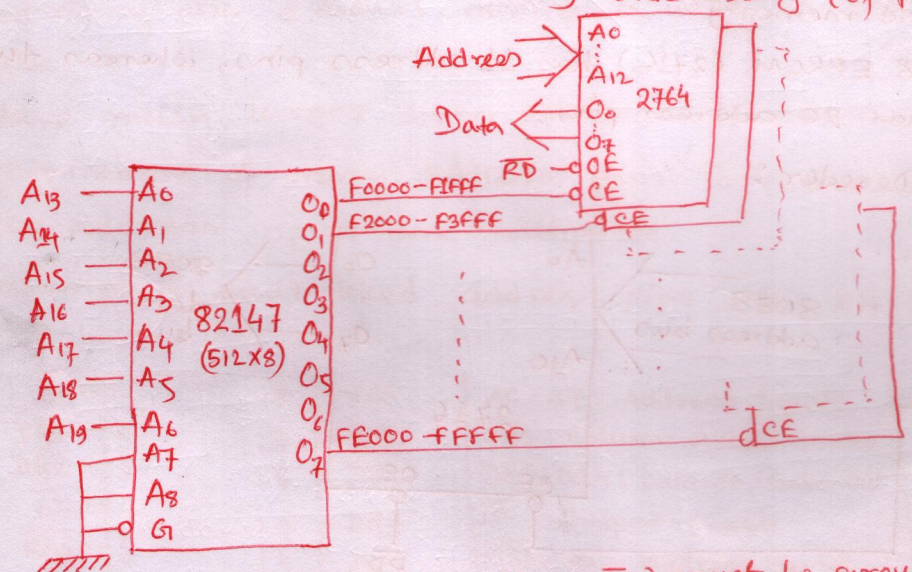
→ A_{16} - A_{19} must have 1 for activating 74138; Therefore, all addresses begin with 'f' at the left

→ # Eight 2764 EPROMs (each 8K X 8, having 13 address pins) are decoded # over address space F0000H - FFFFFH (64K X 8).

⇒ PROM Address decoder:

- Bipolar PROM is used because of its larger number of input connections which reduces the number of other circuits required in the decoding system

- Example: 825147 (512 X 8) PROM has 10 input connections and 8 output connections. Among the input connections 9 are used for ~~mer~~ internal memory addressing (of PROM) with one control (G) input



- Here, the control input ($\overline{G}$ or $\overline{CE}$) must be grounded, as if this PROM's outputs float to their high-impedance state, one or more of the EPROMs might be selected by noise impulse in the system (***)

- Programming pattern of the PROM

Inputs										Outputs							
$\overline{OE}$	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	0	0	1	1	0	1	1	1	1	1	1
0	0	0	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1
0	0	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1
0	0	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1
0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1
0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
all other combinations										1	1	1	1	1	1	1	1

- Main advantage of using a PROM: address map is easily changed
- As the PROM comes with all the locations programed as Logic 1, only eight of the 512 locations must be programmed.

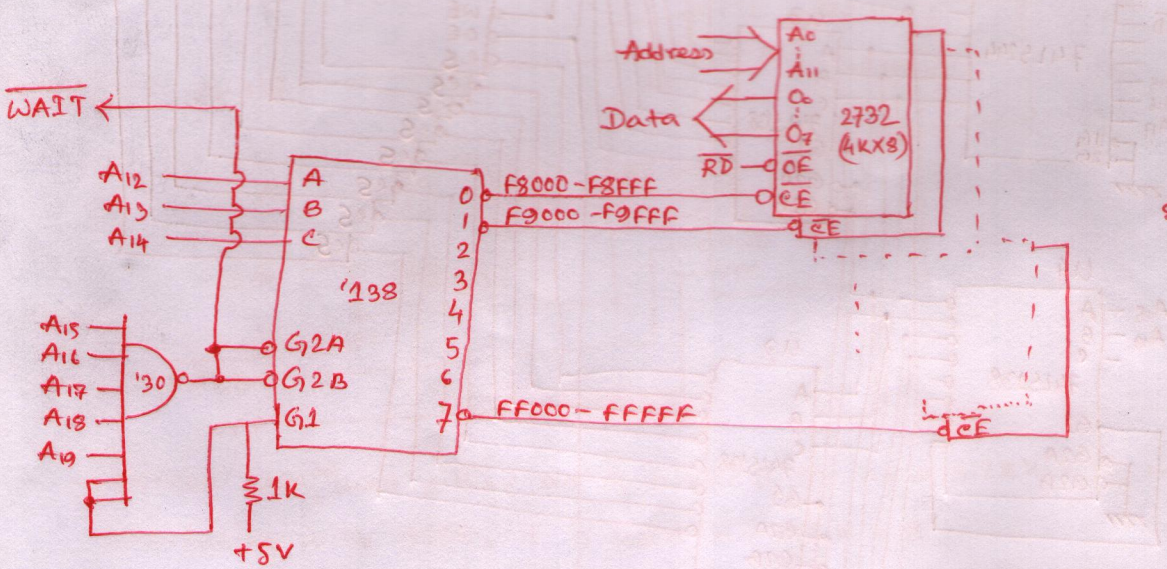
⇒ PLD Programmable Decoders:

- There are three Programmable Logic Devices (PLDs) that function in the same manner: PLA (Programmable Logic Array), PAL (Programmable Array Logic), and GAL (Gated Array Logic).
- These devices can be programmed by blowing fuses to establish connections
- The decoding ckt using a PLD is similar as that with a PROM, where the PROM gets replaced by the PLD (programmed with the logic representing the input-output pattern ~~relat~~ realized by PROM) [showed in a table in the last decoding technique]
  - No control or selection logic is required here

8088 and 80188 (8-bit) Memory Interface

- Examples presented here pertain to minimum mode ($\mu p \Rightarrow$ 20 address pins, 8 data pins, and 3 control signals consisting IO/$\overline{M}$, $\overline{RD}$, and $\overline{WR}$)

⇒ Interfacing EPROM to the 8088:

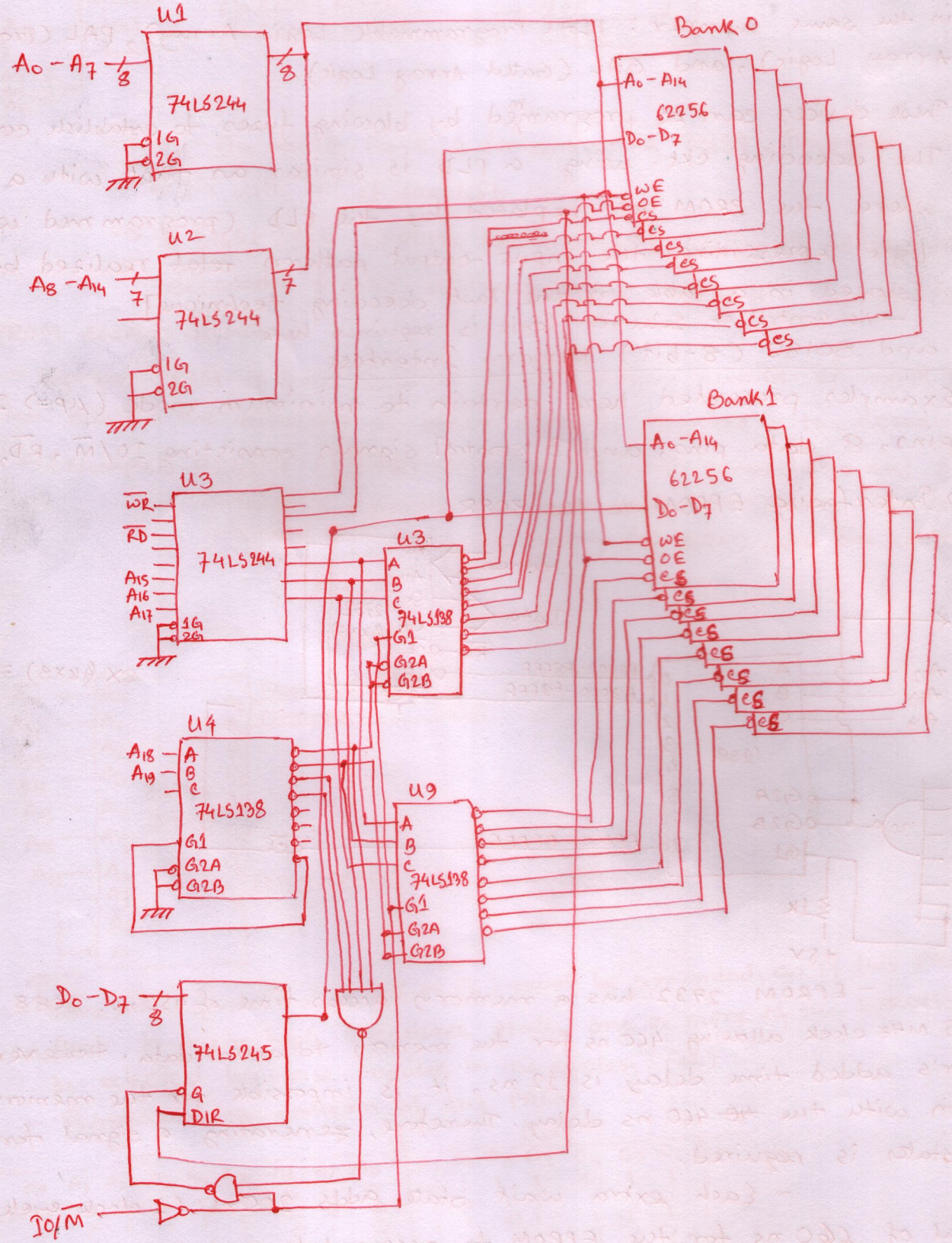


8X (4KX8) = 32KX8

- EPROM 2732 has a memory access time of 450 ns. 8088 operates with 5 MHz clock allowing 460 ns for the memory to access data. However, as decoder's added time delay is 12 ns, it is impossible for the memory to function with the ~~40~~ 460 ns delay. Therefore, generating a signal for inserting WAIT states is required.
- Each extra wait state adds 200 ns (1 clock cycle) making a total of 660 ns for the EPROM to access data.
- The ∴ address space (F8000H - FFFFFH) includes the upper 32K bytes of memory containing FFFF0H, where 8088 starts executing instructions after a hardware reset. FFFF0H location is often called

the ~~it~~ "cold start" location. s/w stored at FFFF0H location ~~generally~~ would contain a JMP instruction at FFFF0H that jumps to F8000H so the reminder of the program can execute.

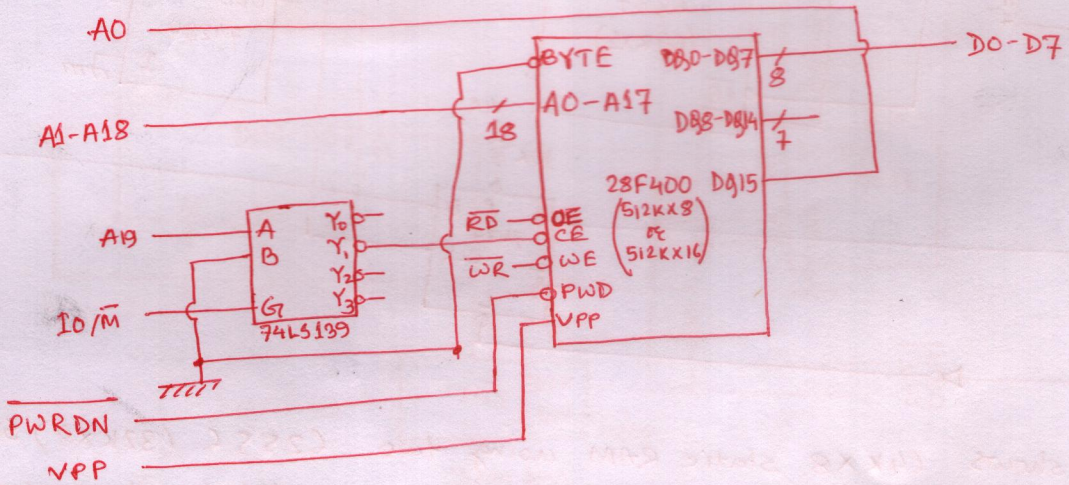
⇒ Interfacing RAM to the 8088:



- Most RAMs do not require wiat states.
- Interrupt vectors (often modified by s/w packages) reside in RAMs
- In the example, 16 62256 (32K×8) static RAMs are interfaced to 8088 beginning at memory location 00000H, to 7FFFFH (512KB). Here, two decoders select from 16 different RAM components and a third to select the other decoders for appropriate memory sections.
  - U4 selects the other two decoders. U3 selects addresses beginning with 00 and U9 selects addresses beginning with 01. Extra pins of U4 remain for future extension.
  - All address, data, control (RD and WR) are buffered. Buffering is important when many devices appear in a single system. Excessive load without buffering can cause logic 0 output to rise above 0·8 V, maximum allowed in a system.

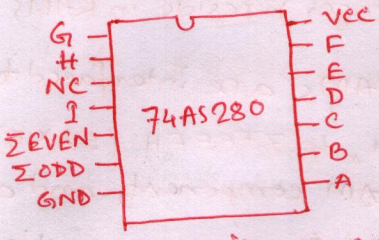
⇒ Interfacing flash memory:

- Flash memory requires 12V programming voltage to erase or write new data. The 12V can be available either at the power supply or through 5V-to-12V converter.



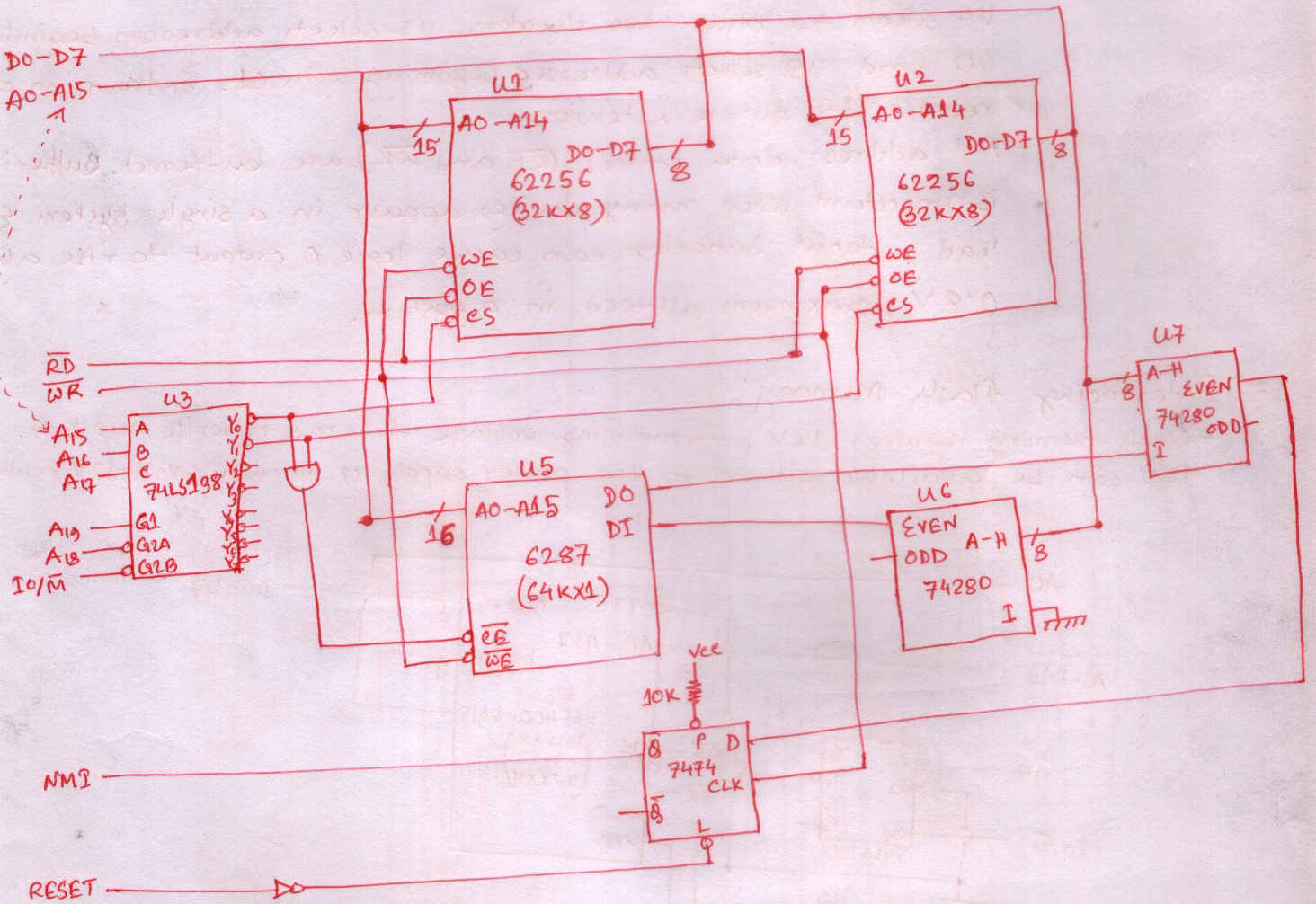
- 28F400 can be used either as a 512K×8 or as a 512K×16 memory device. For interfacing with 8088, it is used as 512K×8
- New pins in flash memory compared to SRAM: i) VPP, which is connected to 12V for erase and programming; ii) $\overline{PWD}$, which selects power-down mode when a logic 0 is applied and is also used for programming; and iii) BYTE, which selects byte (0) or word (1) operation.
- Pin DQ15 functions as the least-significant address input when operated in the byte mode.
- Flash memory is much slower than SRAM (can need around 10^7 times more time)
- The single decoder (74LS139) used address connection A19 and IO/$\overline{M}$ as inputs. (location in the example: 80000H – FFFFFH).

Parity for memory error detection



9 bit parity generator/detector

Input: # of Logic 1 in A-I (9 pins)	Outputs ΣEVEN	ΣODD
0, 2, 4, 6, 8	H	L
1, 3, 5, 7, 9	L	H



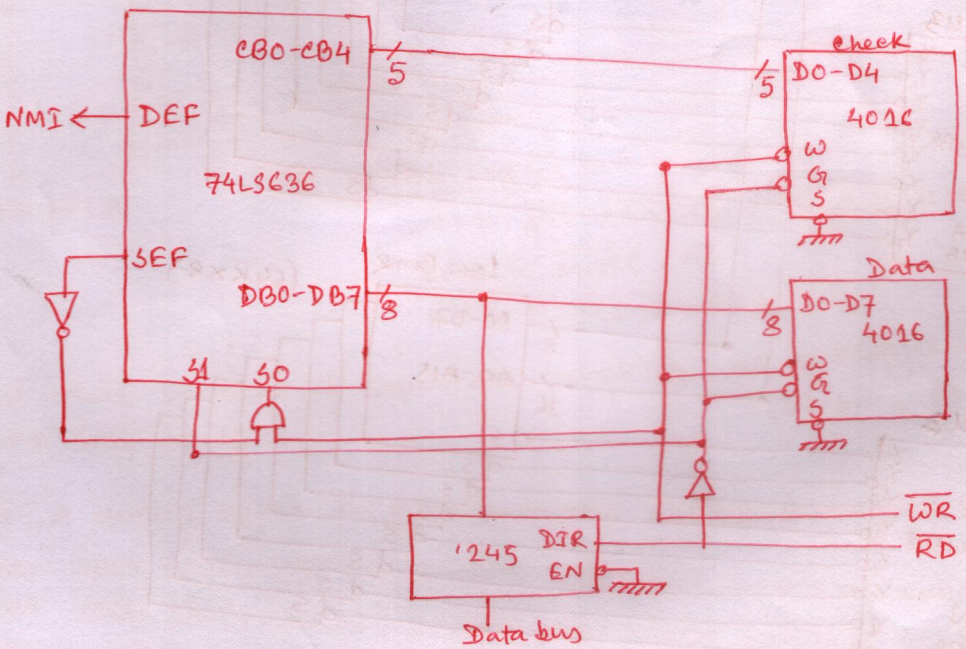
- Example shows 64K x 8 static RAM using two 62256 (32K x 8) SRAMs having a parity bit stored in 6287 (64K x 1) SRAM generated (parity generated) by 74AS280.
- Decoded memory space is at 80000H - 8FFFFH.
- As I of U6 is grounded, a 1 is stored in parity RAM (U5) if an even # of 1's appear in data bus (connected to A-H). Thus, including the stored parity bit, an odd ~~# of~~ parity is stored for each byte.
  - Parity SRAM: No $\overline{OE}$ connection for $\overline{RD}$; It reads data from output pin when selected ($\overline{CE}$ is enabled) and writes data when selected with $\overline{WE}=0$
- If overall parity is odd, (everything is correct), the even parity output of U7 is 0.
- If any bit of info read from the memory changes for any reason, the even output of U7 will become logic 1. The parity output is connected to 8088's NMI.
- Data read from the memory are settled to the final state before an NMI ~~input~~ input (error detection) occurs through being timed by a D flip flop.
  - The D FF latches output of the parity checker (U7) at the end of an $\overline{RD}$ cycle on the memory.

ror Correction

- 74LS636: An 8-bit error correction and detection circuit that corrects any single bit memory read error and flags any 2-bit error.
  - Corrects errors by storing five parity bits with each byte of memory data.
  - If more than two bits are in error (rarely happened), the circuit may not detect it.
  - Whenever a memory components fail completely, its bits are all high or all low. In this case, the circuit flags the processor with a multi-bit error detection.
  - 8 data pins, 5 check bit pins, 2 control inputs (S0 and S1), and 2 error outputs (Single Error Flag - SEF and Double Error Flag - DEF).

S0	S1	function	SEF	DEF
0	0	Write check word	0	0
0	1	Correct data word	*	*
1	0	Read data	0	0
1	1	Latch data	*	*

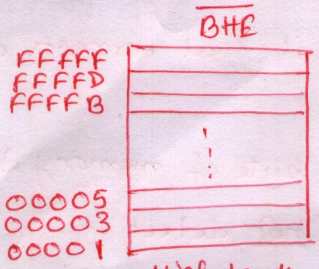
*: These levels are determined by the type of error.

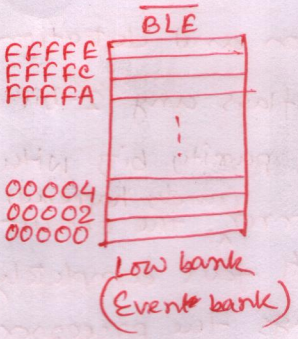


- When a single error is detected, the 74LS636 goes through an error correction cycle: it places 01 on S0 and S1 by causing a wait and then a read following error correction.
- Difference in connection in memory component: $\overline{S}$ is grounded, which enables data to be accessed from memory before $\overline{RD}$ goes low
- On the next negative edge of the clock after an $\overline{RD}$, the 74LS636 checks SEF. If a single bit error is detected, a correction cycle causes the single-bit error to be corrected. If a double-bit error is detected, DEF generates an NMI.

8086, 80186, 80286, and 80386 (16-bit) memory interface

- 80286/80386: 24-bit address bus; $\overline{MRDC}$ and $\overline{MWTC}$ instead of $\overline{RD}$ and $\overline{WR}$
- These µPs must be able to write data to any 16-bit or 8-bit location. Therefore, the 16-bit data bus must be divided into two separate sections (banks), which are 8-bit wide.

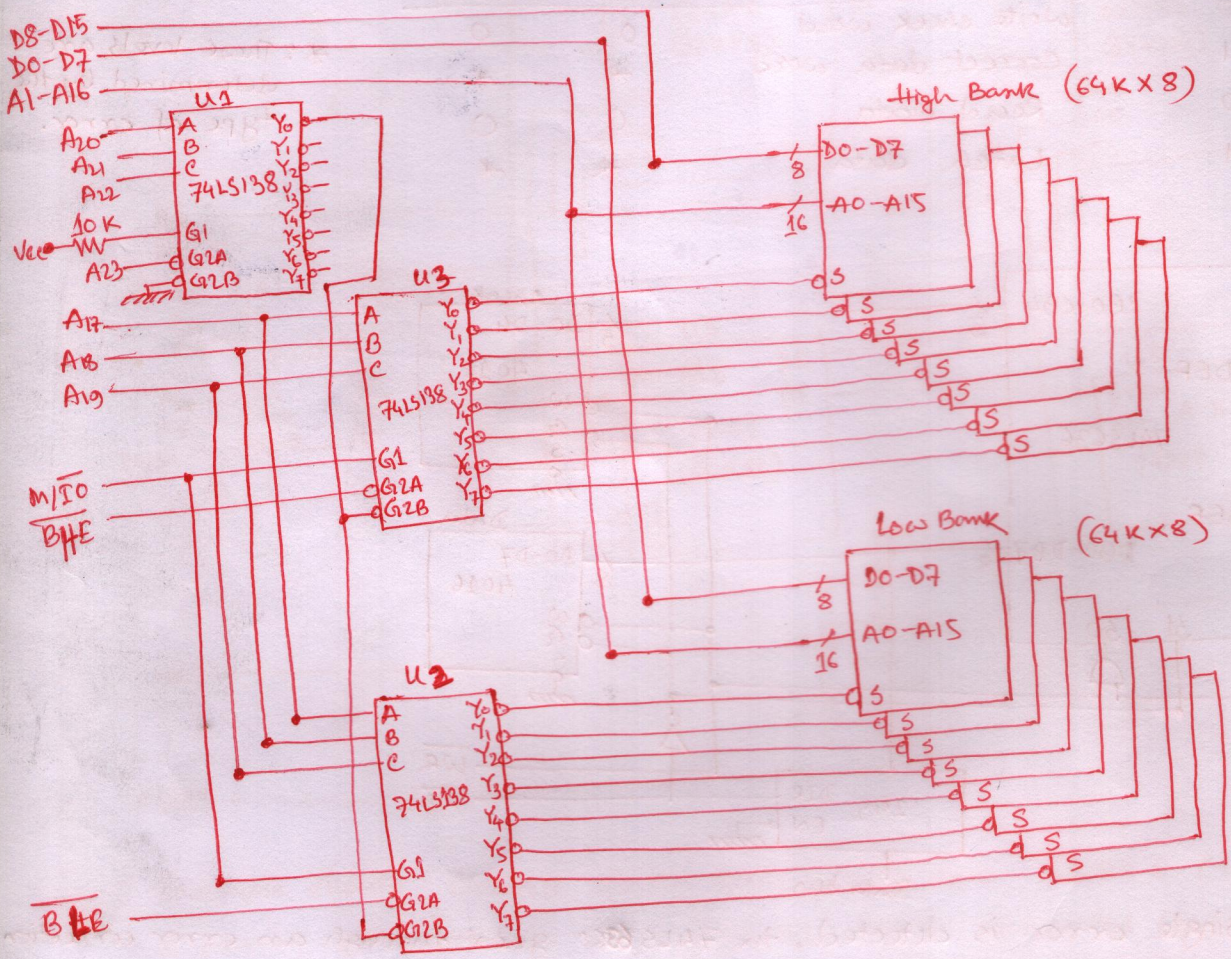




$\overline{BHE}$	$\overline{BLE}$ (A0)	Functions
0	0	Both banks are enabled for a 16-bit transfer
0	1	High bank is enabled for an 8-bit transfer
1	0	Low " " " " " "
1	1	No bank is enabled

- Bank selection is accomplished in two ways: (1) a separate write signal is developed to select a write to each bank (~~least~~ costly), or (2) separate decoders are used for each bank.

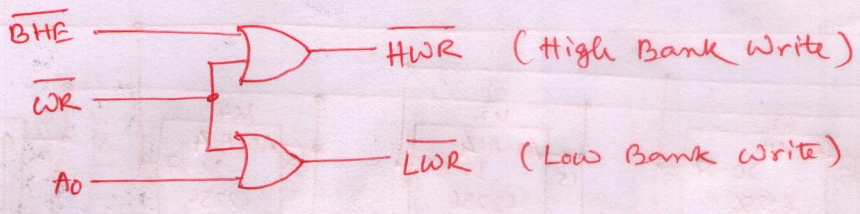
Separate Bank Decoders



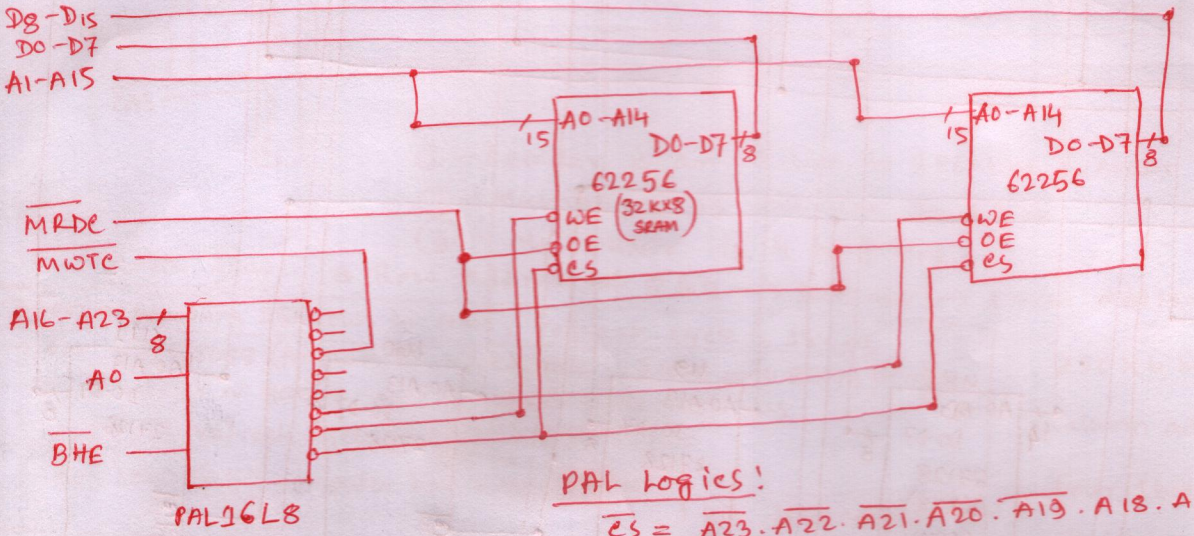
- Two 74LS138 decoders are used to select 64K RAM for 80386SX μp (24-bit addr).
- An enable pin (G2A) ~~is~~ of U2 is enabled by $\overline{BLE}$ (A0) and of ~~U3~~ U3 is enabled by $\overline{BHE}$.
- ~~U3~~ U2 controls enabling the low bank and ~~U2~~ U3 controls enabling the high bank.
- Decoded memory location range is 000000H - 0FFFFFH (1M).
- A0 (or $\overline{BLE}$) from the μp is not connected to the memory to ensure having separate memory addresses (odd or even) in separate banks. If it would be connected to the memory address pin, then half of the memory will be wasted. [A0 is NOT even a pin in 80386SX μp]

Separate Bank Write Strobes

- Generation of separate 8086 write strobes for memory: $\overline{WR}$ combines with A0 for low bank selection ($\overline{LWR}$), and $\overline{BHE}$ for high bank selection ($\overline{HWR}$)
  - For 80286/80386: $\overline{MWTC}$ is used instead of $\overline{WR}$



- Why not also generate read strobe for each memory bank?: It is usually unnecessary as 8086, 80286, 80386 µps read only the byte of data they need at any given time ignoring the other parts without ~~eas~~ causing any problem.
- Example: 16-bit memory stored at locations 060000H – 06FFFFH for 80286 or 80386 µp using PAL



PAL Logics:

$\overline{CS} = \overline{A23} \cdot \overline{A22} \cdot \overline{A21} \cdot \overline{A20} \cdot \overline{A19} \cdot A18 \cdot A17 \cdot \overline{A16}$

$\overline{LWR} = \overline{MWTC} \cdot \overline{A0}$

$\overline{HWR} = \overline{MWTC} \cdot \overline{BHE}$

- Extend this concept of separate banking: all subsequent designs up to Section 10-6 (32-bit, 64-bit, ---)

*** Self Study.

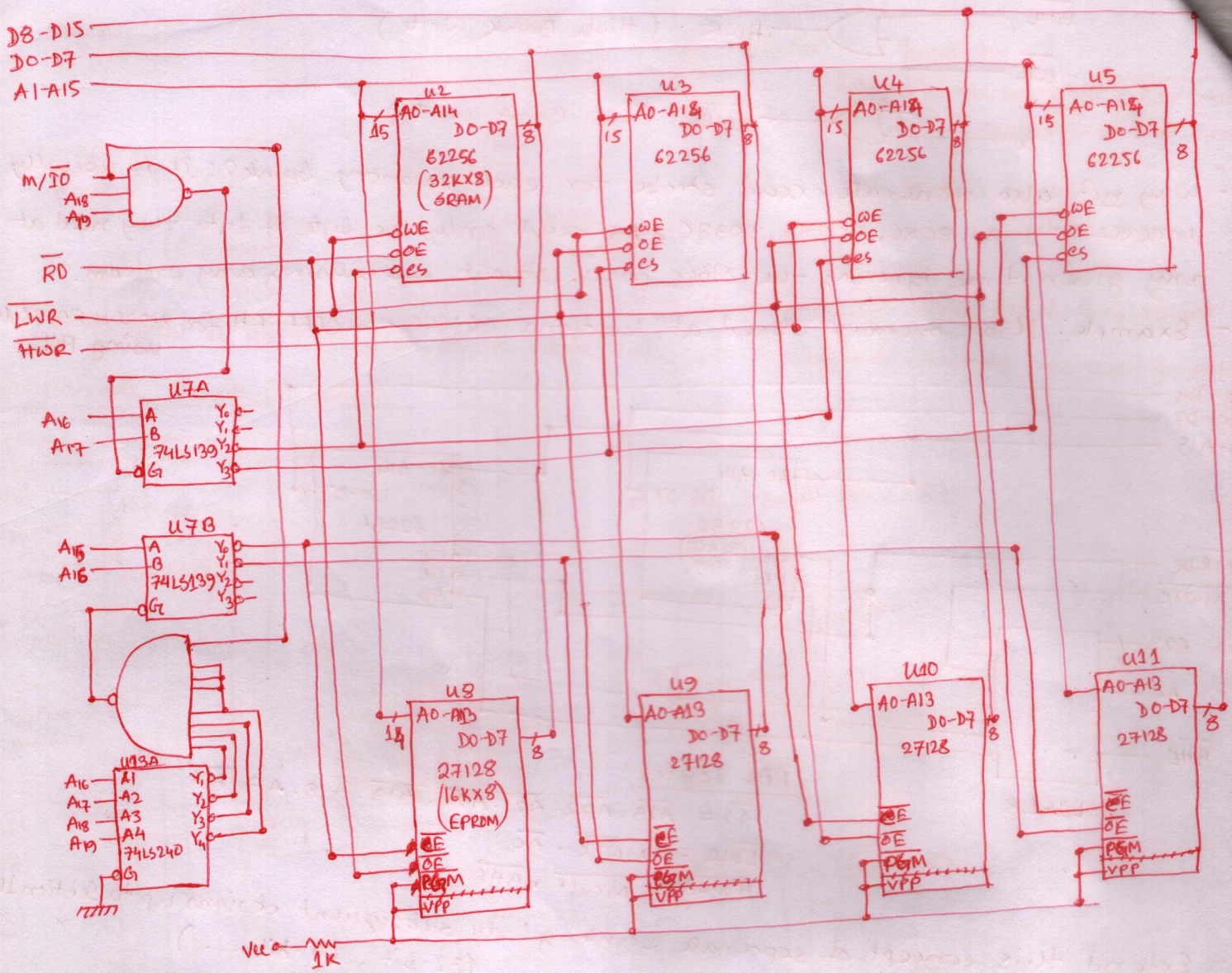
- Example: A memory system for 8086 containing 64K byte EPROM and 128K byte SRAM

In Book { four 27128 EPROMs (16K×8) composing 32K×16 bit memory at location F0000H-FFFFFH and four 62256 SRAMs (32K×8) composing 64K×16 bit memory at location 00000H-1FFFFH

(Even though the memory is 16 bit wide, it is still numbered in bytes)

* * → Actually should be: EPROM → 00000H - 0FFFFH

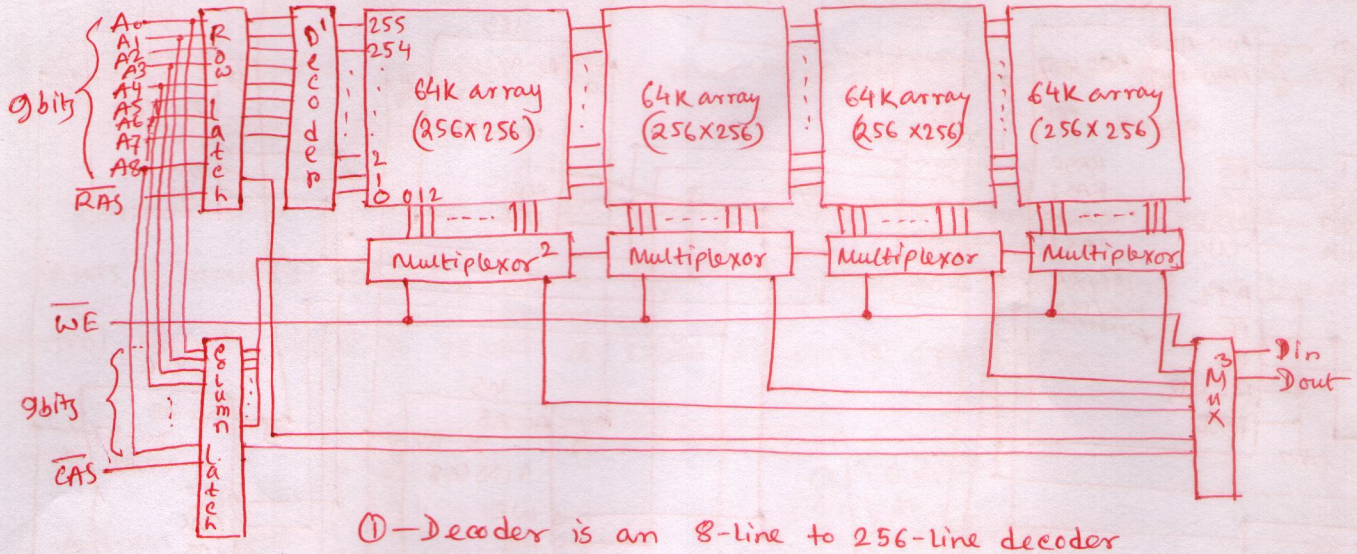
RAM → E0000H - FFFFFH



- $\overline{RD}$ is connected to all $\overline{OE}$ ~~outputs~~ inputs (enabling all 16 bits is okay while reading)
- $\overline{LWR}$ and $\overline{HWR}$ are connected to different banks of RAM. Here, for writing, 16-bit or 8-bit does matter. for 16-bit writing, both $\overline{LWR}$ and $\overline{HWR}$ go low; for 8-bit writing, either of them go low. ($\overline{LWR}$ and $\overline{HWR}$ can be obtained from $\overline{WR}$, $\overline{BHE}$, and A_0 as shown earlier; Besides, such writing is not needed for EPROM)
- A 74LS139 (dual 2-to-4 decoder) is used to select EPROM with one half and RAM with the other half. It decoded memory as ~~a~~ 16-bit wide (not 8-bit).
- EPROM's decoder signal is sent to 8086 wait state generator.

DRAM

- Needs to be refreshed periodically in each 2-4 ms, as data is stored on capacitors that lose their charges within a short period of time. It also needs address multiplexing.
- Any read or write automatically refreshes an entire section of the DRAM.
- Refresh cycles are done by a read, ~~or~~ a write, or a special refresh cycle that doesn't read or write. The refresh cycle is internal to DRAM and accomplished when other memory components operate. (also called hidden refresh, transparent refresh, or cycle stealing)
- DRAM's internal organization contains ~~a~~ series of rows and columns. A 256K×1 DRAM has 256 columns, each containing 256 bit, or row ~~orgized~~ organized into four sections of 64K bits each.
- whenever a memory location is addressed, the colum address selects a column (or internal memory word) or 1024 bits



① – Decoder is an 8-line to 256-line decoder
② – Multiplexor is 256 line to 1 line
③ – Multiplexor is 4 to 1 line

Data flow: A Row address of 8 bits → Decoder → 1 row address is applied to 4 sections

→ for refreshing 256 rows in 4ms, refresh cycle is 15.6μs
→ for 8086/8088 (for example), clock rate is 5 MHz with 800ns for rd/wr
So, 1 rd/wr ~ 800ns and 1 refresh cycle ~ 15.4μs
So, 1 refresh cycle ~ $\frac{15.4\mu s}{800ns} = 19$ rd/wr
So, loss of 5% ($\frac{1}{19}$) computer time ⇒ small price

256 X 4 bits inputs to 4 multiplexors
+
Column address of 8 bits to multiplexors
↓
Four 1 bit data from 4 multiplexors
+
2 higher address bits to MUX

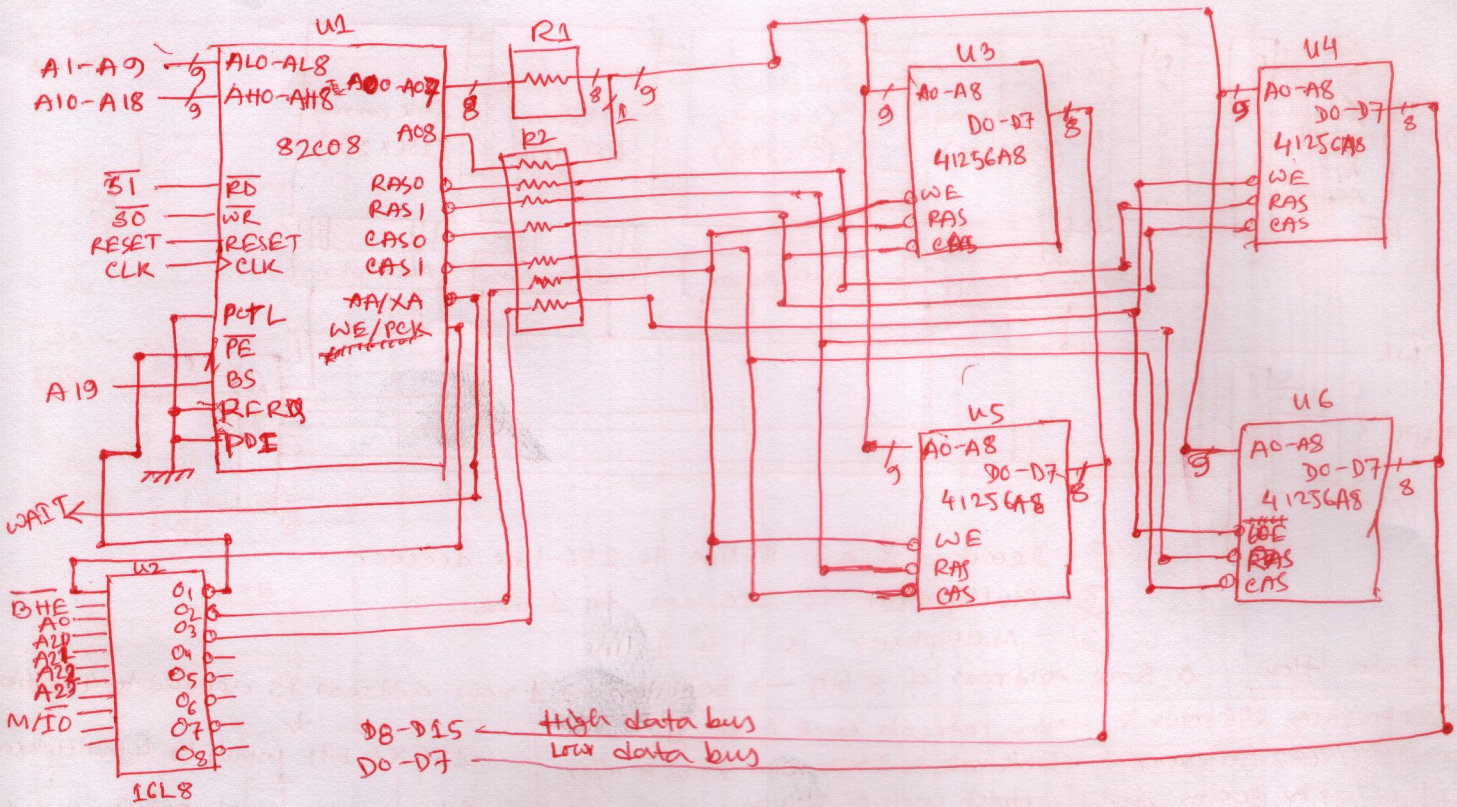
1 bit data output from the MUX ←

EDO Memory:
- A slight modification to DRAM to have EDO (Extended Data Output) DRAM.
- Any memory ~~access~~ refresh, including a refresh, stores the 256 bits selected by RAS into latches
- These latches hold the next 256 bits of information, so that data becomes available without wait state in case of sequential execution
- Increases system performance by 15–25%.

SDRAM (Synchronous Dynamic RAM)
- At the beginning of read/write, it behaves like a standard RAM with the same # of wait states
- Subsequent (second, third, and fourth) accesses need one clock cycle each.
- Therefore, to read four 64-bit numbers, a total of 3 (for 1st access) + 1+1+1 = 6 clock cycles are required. For DRAM, it is 3+3+3+3 = 12 clock cycle (for next 3 accesses)
- 10% performance increase over EDO

DRAM controllers
- Performs address multiplexing and generation of DRAM control signals
- Example: 82C08 – contains an address multiplexer to multiplex an 18-bit address onto 9 bit address connections for 256K memory. It generates CAS and RAS for DRAM. These signals are developed internally by CLK, S1, and S2. It takes S1 and S0 as its RD or WR inputs. Then AACK/XACK provides an acknowledge output that is used to indicate ready cond'n of μPs (normally connected to READY of the μP)

- Example: 82C08 connected to a series of four 256 K x 8 bits memory modules comprising 1M.



→ 82C08 is connected to a series of four 256 k x 8 bit mem

→ U3 and U5 form the high bank, and U4 and U6 form the low bank

→ PAL 16L8 combines $\overline{WE}$ and A0 to generate write signals for U4 and U6; it combines $\overline{WE}$ and $\overline{BHE}$ " " " " " U3 and U5.

→ PAL also develops the controller selection signal ($\overline{PE}$) by combining M/$\overline{IO}$ and other address lines A20 – A23.

→ Memory locations 000000H – DFFFFFFH.

→ A19 selects the upper (NOT higher) bank (U3 and U4) or the lower bank (U5 and U6) through the BS (Bank select) input to 82C08.

→ PAL logic:

$$\overline{HWR} = \overline{BHE} \cdot \overline{WE}$$

$$\overline{LWR} = \overline{A0} \cdot \overline{WE}$$

$$\overline{PE} = \overline{A20} \cdot \overline{A21} \cdot \overline{A22} \cdot \overline{A23} \cdot M\overline{IO}$$