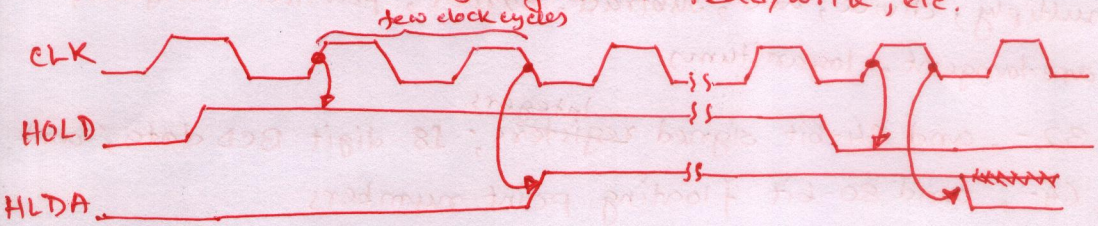
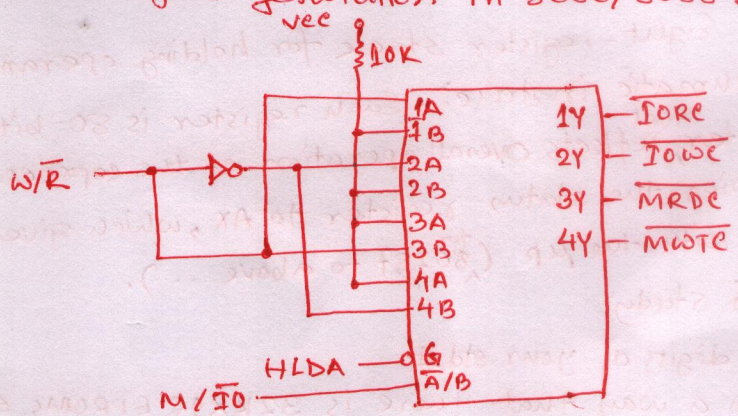
Direct Memory Access (DMA) L-5

- 3 forms of I/O: Basic, interruppt-processed, and DMA
  (Basic, interruppt-processed: Already covered)
- DMA: Provides I/O a direct access to memory while the µp is temporarily disabled
- Applica^n's: DRAM refresh, video displays for refreshing the screen, disk memory system read/write, etc.



- HOLD is sampled in the middle of any clocking cycle. As soon as the µp recognizes the hold, it ~~set~~ stops executing s/w and enters hold cycles.
- (*) HOLD has a higher priority than INTR and NMI. Interrupts take effect at the end of an instruc^n, while a HOLD takes effect in the middle of an instruc^n.
- The only µp ping having higher priority than HOLD is the RESET pin
- The HLDA becomes active to indicate that the µp has ~~entered~~ placed its buses at high-impedence states
- HOLD: DMA request input; HLDA: DMA grant output.
- DMA read: Xfers data from memory to the I/O device ($\overline{MRDC}$, $\overline{IOWC}$) } available in 80186 and above
  DMA write: " " " an I/O device " " memory ($\overline{MWTC}$, $\overline{IORC}$) }
- Control signal generation in 8086/8088:



- $\overline{IORC}$ gets enabled with $\overline{R}$ when $\overline{IO}$ gets activated
- $\overline{MWTC}$ gets ~~act~~ enabled with (W→D0) when M gets activated

M/$\overline{IO}$	W/$\overline{R}$	Output
M	W→D0	$\overline{MWTC}$
M	$\overline{R}$	$\overline{MRDC}$
$\overline{IO}$	W→D0	$\overline{IOWC}$
$\overline{IO}$	$\overline{R}$	$\overline{IORC}$

8237 DMA Controller

- A special-purpose µp whose job is high-speed data xfer between mem and I/O
- Pins of 8237:
  - DREQ3 - DREQ0: DMA request inputs (8237 is a four-channel device)
  - DACK3 - DACK0: DMA channel acknowledge
  - HRQ: HOLD request
  - HLDA: ~~HOLD~~ acknowledge
  - $\overline{EOP}$: End of Process; used to terminate a DMA process or to signal the end of a DMA xfer.

- $\overline{IOR}$, $\overline{IOW}$, $\overline{MEMR}$, $\overline{MEMW}$, DB7-DB0, A7-A4 (A3-A0 also selects an internal register), CLK, $\overline{ES}$, RESET, READY, ...
- Internal registers: Self study

Arithmatic Coprocessor

- 8087, 80287, 80387X, ...
- 80X87: Able to multiply, divide, add, subtract, sqroot, partial tangent, partial arctangent, logarithms
- Data types: 16-, 32-, and 64-bit signed ~~registers~~ integers; 18-digit BCD data; and 32-, 64-, and 80-bit floating point numbers
- Operations performed by the 80X87 generally execute many times faster than equivalent operations written with the most efficient programs that use the µp's normal instruc^n set.
- 80X87:
  - Can operate concurrently with µp
  - Coprocessor instruc^ns - escape (ESC) instruc^ns: Used by the µp to generate a memory address for the coprocessor so that the coprocessor can execute a coprocessor instruc^n
  - Divided into two major sections: Control unit and Numeric execu^n unit.
    - Control unit: Interfaces the coprocessor to the µp. Both monitor the instruc^ns stream. Coprocessor executes if it is an Esc. Instruc^n. ~~Esc~~ µP executes if it is not.
    - Numeric execu^n unit: Eight-register stack for holding operands and results of an aritumatic instruc^n. Each register is 80-bit. It has a status register that reflects overall operation of the coprocessor. "FSTSW AX" instruc^n copies the status register to AX, which gives the only way to communicate with the µp (for 80287 to above ...).

⇒ 80386 and 80486 Basics: Self study

→ X = Std id Y.Z ; YYY = Last (LSB) 3 digits of your std #

Again if X=0; interface 80386 in such a way that there is 32KX16 EPROMs starting from ~~your~~ 20####H, 64KX16 RAM starting from 20####H, and 8 I/O devices connected at consecutive ports starting from 8####8H.

if X=1, interface 80486 in such a way that there is 32KX16 EPROMs starting from 80####H, 64KX16 RAM starting from 40####H, and 8 I/O devices connected at consecutive ports starting from 10####8H

↑ # of bits ~~#~~ in address is not shown in full in all the addresses