Memorial University of Newfoundland

Engineering 4862 MICROPROCESSORS

Assignment 5 Solution

Please note: Part A is due next Friday (July 20) and Part B is due on July 27. The last question is a bonus question that worth 2 Marks of the final total mark (Equivalent to the mark of one assignment)

-- Part A --

0. What control byte will configure an 8255A PPI such that Ports A & B are input, and Port C is output?

Solution: From the PPI data sheet, we can get

Bit	Value	Reason
D_7	1	mode set
D_6 D_5	00	Mode 0 for Group A
D_4	1	Port A Input
D_3	0	Port C (upper) Output
D_2	0	Mode 0 fro Group B
D_1	1	Port B input
D_0	0	Port C (lower) output

Control byte 1001 0010 (92H)

1. Sketch a diagram of an address decoder for an 8-bit input port. The assembly language instruction IN AL, 94h should fetch a byte from the port. There should be no more than 1024 mirror images, and it should not interfere with operation of I/O ports at 0010h and 0037h.

Solution: Next page

2. What is the advantage and penalty of multiplexing and demultiplexing the address/data in the 8088/86 CPUs?

Solution:

It uses less pins and consequently has a smaller package. But it takes longer time because data and address need to be demultiplexed.

3. When the input signal RESET in the 8088/86 is activated, what are the contents of the CS and IP register? Why ROM is always arranged to the high/upper end of the memory map (closer to the FFFFFH) for Intel 80x86 serial microprocessors?

Solution:

After reset, for 8086/88, CS=FFFFH, IP=0000H, that gives a physical address FFFF0H. That is where the reset vector is stored. CPU will always go to that address to fetch the first instruction. Because at the time, nothing has been loaded to the RAM, so we have to arrange a ROM to cover that location. Usually, we arrange the ROM from the highest address FFFFFH downward and make the address continuous.

* 8-bit input port with address 00094h
* max 1024 mirror images ∴ max 10 unused address lines
* All I/O addresses only use 16 address lines
* Cannot interfere with ports at 00010h and 00037h

	A_{15}			A_0
New Port (94h)	0000	0000	1001	0100
Existing Ports (10h)	0000	0000	0001	0000
(37h)	0000	0000	0011	0111

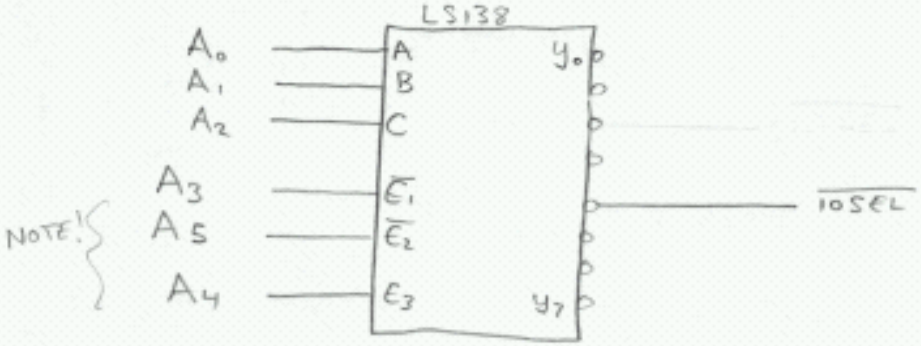
We need to use at least 6 lines, such that we can select the new port, but not the existing ports.

For example, choosing lines A_{10}–A_{15} is a very poor choice. For all three ports, these lines have the value zero. There is no way to differentiate each port.

So choose 6 lines that have different values for each port. Lines A_{0-5} will work fine.

$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$
0 1 0 1 0 0

So one possibility for decoding is to use a 3x8 decoder.



Any address which fits the following pattern is valid:

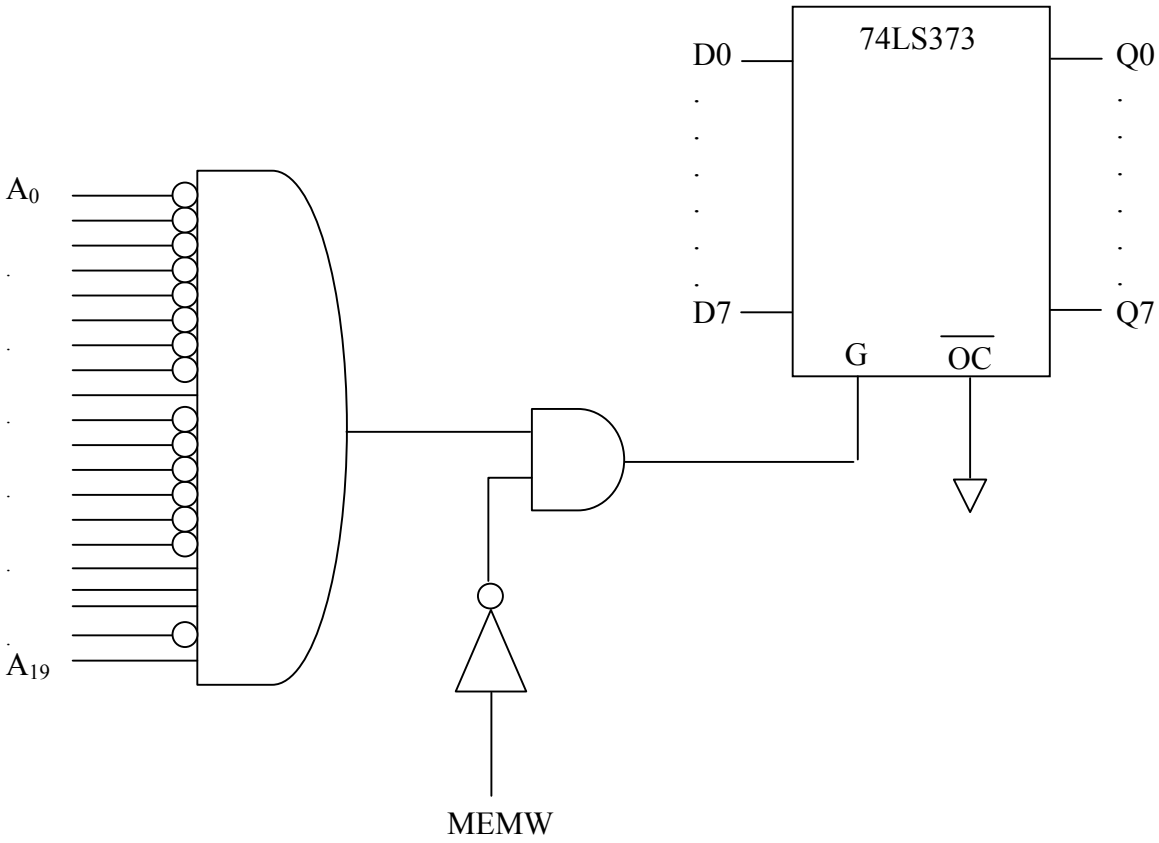
A_{15} A_0
XXXX XXXX XX01 0100

Thus port 14h will also access the same input port. That is okay; as long as port 94h works (it does), the solution meets the specification.

There are many different correct solutions.

4. Show the logical design of "MOV [0100], AL" for memory mapped I/O using AND and Inverter gates and a 74LS373 latch. Assume that DS=B800H.

The PA=B8000+0100 = B8100H. It use memory mapped I/O so MEMR or MEMW is used. Because we move data to the memory, therefore, MEMW is used.



-- Part B --

5. Calculate, using execution times, the total time required by the following program. Assume the program is running on a system with 5-MHz 8086 microprocessor (with the help of Intel User Manual).

```
MOV AX, 0000h
MOV SI, 0000h
MOV CX, 8000h
OUTER: MOV DX, CX
MOV [SI], AX
ADD SI, 2
MOV CX, 0FFFFh
INNER: LOOP INNER
MOV CX, DX
LOOP OUTER
```

	INSTRUCTION		CLOCKS
	MOV	AX, 0000h	4
	MOV	SI, 0000h	4
	MOV	CX, 8000h	4
OUTER:	MOV	DX, CX	2
	MOV	[SI], AX	9+EA = 9+5 = 14 (SI is even)
	ADD	SI, 2	4
	MOV	CX, OFFFFh	4
INNER:	LOOP	INNER	{ 17 if jump; 5 if no jump
	MOV	CX, DX	2
	LOOP	OUTER	{ 17 if jump; 5 if no jump

INNER Loop Clocks:

$$C_I = FFFFh \times 17 + 5$$
$$= 65535 \times 17 + 5 = 1\,114\,100$$

OUTER Loop Clocks:

$$C_O = 8000h \times [2+14+4+4+2+17+C_I] + 5$$
$$= 32768 \times (1114143) + 5$$
$$= 36\,508\,237\,829$$

$$TOTAL = C_O + 4+4+4$$
$$= 36\,508\,237\,831 \text{ clocks}$$

$\therefore$ Total Time = clocks × period of one clock ← $\frac{1}{5MHz} = 200ns$
= clocks / frequency of clock
$\doteq$ 7301.65 seconds
$\doteq$ 2 hours 1 minute 41.65 seconds.

Note that simply using the time for the inner loop × # of outer loops is accurate to within half a second, and would be acceptable:

$$8000h \times FFFFh \times 17 \times 200ns$$
$$= 7301.3 \text{ sec}$$

6. Repeat the previous question for a system with a 4.77-MHz 8088 microprocessor. (Only recalculate values that change.)

There are two differences: one is due to µPro used (8088), and the other is due to clock speed

① MOV [SI], AX clocks = 14+4 (word operand on an 8088)

$\therefore C_0 = 32768 \times (1\,114\,143 + 4) + 5$
$= 36\,508\,368\,901$

② clock period = $\frac{1}{4.77\text{MHz}}$ = 209.6ns

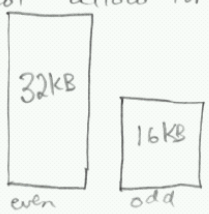
$\therefore$ New time = $36\,508\,368\,913 \times 209.6\text{ns}$
$\doteq 7653.75$ seconds
= 2 hr 7 min 33.75 sec

7. Is it possible to create an 8086-based computer system with one 32KB (8-bit wide) RAM device and one 16KB (8-bit wide) RAM device? Explain.

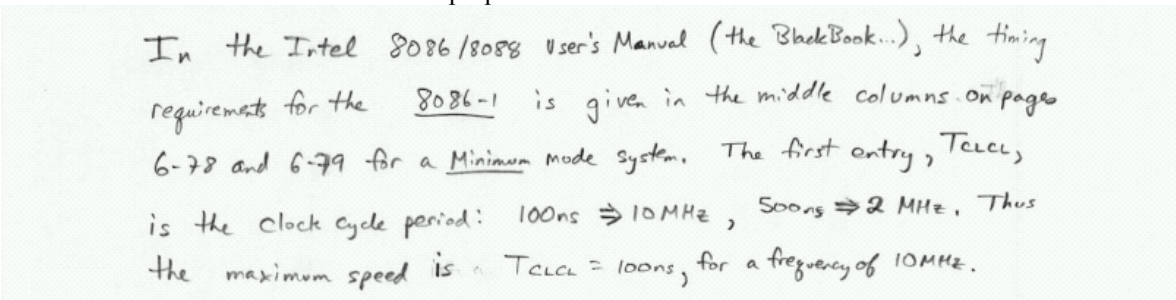
An 8086-based system uses separate even & odd memory banks. This means that all memory cells that are accessed with even 20-bit addresses are grouped together. Likewise for odd addresses.

Because these addresses are interleaved, it makes no sense to have a range of even addressed memory cells without the corresponding range of odd addressed cells. A standard, single RAM IC can not be used for both banks because it only has an eight-bit data path.

So the given chips do not allow for a feasible 8086-based system.



8. An 8086-1 microprocessor based system operating in minimum mode at its maximum speed already includes 4 kilobytes of RAM (two 2K x 8 devices) and 16 kilobytes of EPROM (two 8K x 8 devices), which are mapped in appropriate locations with no mirror images. This system needs to be expanded by adding two 51256S-10 SRAM devices and one 27210-200V10 EPROM device. A new address decoder should be added which would cater to the expansion devices; the most significant three address lines with one 74LS138 should be used for this purpose.



a. *Assuming that the data lines needs to be buffere*d, sketch a circuit diagram illustrating the interface between the CPU and the expansion memory. Show all the connections to the pins of the memory devices; include latches and all the necessary TTL devices. The existing memory devices (4K RAM and 16K ROM) and their address decoders **need not** be included in your diagram.

Diagram see next page.

b. List the address ranges (for both ROM and RWM) corresponding to all memory devices in the system – existing as well as additional devices. List all mirror images.

c. Sketch the memory map of the system – including the existing as well as newly added memory – and clearly indicate all mirror images.

d. Consider the address line A13. Find out the additional load – current as well as capacitance load – placed on this line due to the expansion. Which device should be capable of driving this additional load?

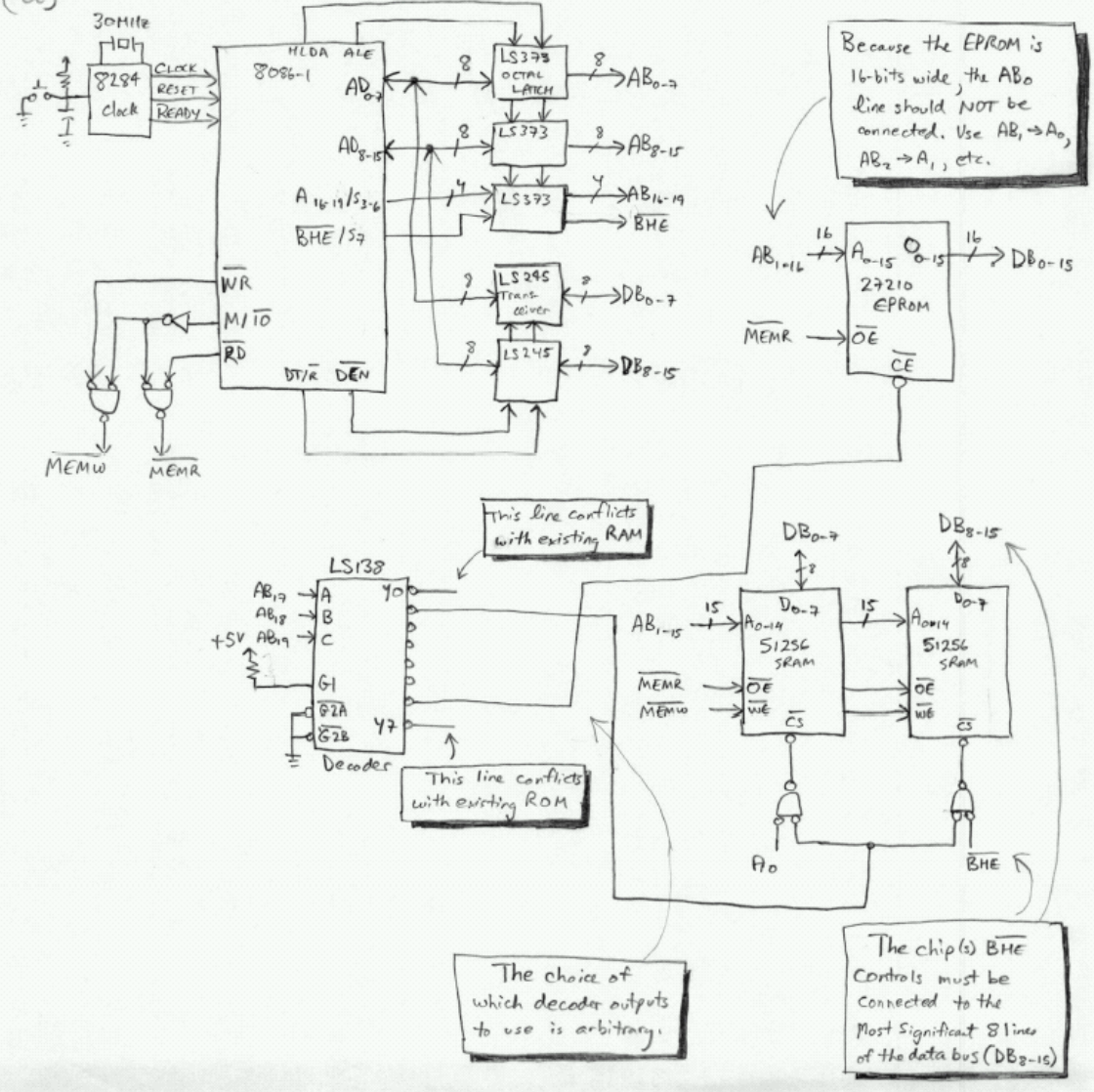
e. Determine the number of wait states required to interface the (i) SRAM and the (ii) EPROM. Suggested propagation delays are: decoder ~40ns, latch ~30ns, buffer ~25ns, transceiver ~30ns, and each logic gate ~10ns.

f. Assume that the O4* output line of the decoder in your design is used for selecting a pair of memory mapped 8255 PPI devices. Assume also that Port A in both these devices have been configured as input, thus obtaining a 16-bit input port. What address(es) should be used to access this 16-bit port?

g. Write a program that will read the contents of this port at a known sampling rate, and store the acquired information in an array located in the expansion RAM; 100 samples are required to be collected. Hint: use the extra segment for the RAM.

h. Suppose that the rate at which samples are collected is not pre-determined, but depends on the speed of the input device connected to the 8255. Therefore, you cannot simply write a delay routine to determine when to collect input samples. What hardware/software solution would you recommend to solve this problem? That is, how will you use the 8255 device in this case?

(a)



(b)

ROM – existing: 1111 11?? ???? ???? ???? = FC000 – FFFFF

16KB = 2 × 8KB
∴ 14 lines

(Must assume contiguous

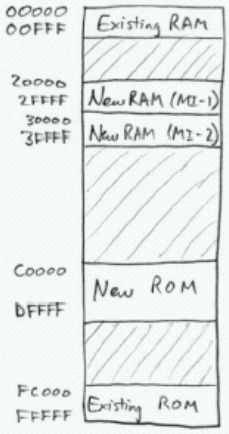
– new: 110? ???? ???? ???? ???✓ = C0000 – DFFFF

RAM – existing: 0000 0000 ???? ???? ??? = 00000 – 00FFF

4KB = 2 × 2KB
∴ 12 lines

– new: 001x ???? ???? ???? ???? = 20000 – 2FFFF, 30000 – 3FFFF } 2 Mirror Images

(c)



(d) Current Load:

ROM: $I_{LI} = 1\mu A$ ← same for both I_{IL} & I_{IH}

RAM: $I_{LI} = 1\mu A$ ←

∴ Logic Low = $3\mu A$ extra

∴ Logic High = $3\mu A$ extra

Capacitance:

ROM: $C_{IN} = 6pF$ RAM: $C_{IN} = 8pF$

∴ Additional Capacitance load = 6+16 = 22 pF

The 74LS373 OCTAL LATCH should be capable of driving this additional load.

(e) (i) SRAM – READ

① Address valid to data valid:

MAX

$$3T_{CLCL} - T_{CLAV} - t_{LATCH} - t_{TRANS} - T_{DVCL} = 3\times100 - 50 - 30 - 30 - 5 = 185ns$$

$t_{AA} = 100ns < 185ns$ ∴ OKAY (NO WAIT STATES)

② Chip Select Access Time: (Chip enable to data valid)

$$3T_{CLCL} - T_{CLAV} - t_{LATCH} - t_{DECODER} - t_{TRANS} - T_{DVCL} = 300 - 50 - 30 - 40 - 30 - 5 = 145ns$$

$t_{ACS} = 100ns < 145ns$ ∴ OKAY (NO WAIT STATES)

t_{AA} and t_{ACS} are values from the SRAM data sheet

③ Output enable to Data Valid: ($\overline{RD}$ low to data valid)

$$2T_{CLCL} - T_{CLRL} - T_{DVCL} - T_{DR} - T_{TRANS} = 200 - 70 - 5 - 10 - 30 = 85ns$$

$t_{OE} = 50ns < 85ns$ ∴ OKAY (NO WAIT STATE)

SRAM Write

① Write Pulse Width

$$2T_{CLCL} - 35 = 200 - 35 = 165ns$$

$t_{WP} = 70 < 165ns$ ∴ OKAY

See question 2 for more details on calculating these values.

② Setup Time (MIN) (MAX) Increases the end of WR

$$T_{WLWH} + T_{CVCTV} - T_{CLDV} + T_{DR} - T_{TRANS}$$
$$= 165 + 10 - 50 + 10 - 30 = 105ns$$

$t_{DW} = 40ns < 105ns$ ∴ OKAY

③ Hold Time (for Data)

$$T_{WHDX} - T_{DR} + T_{TRANS} = T_{CLCH} - 25 - 10 + 30 = 53 - 5 = 48ns$$

$t_{DH} = 0ns < 48ns$ ∴ OKAY (ie calculation was unnecessary)

(ii) EPROM:

① Address to data valid: $t_{ACC} = 200\,ns > 185\,ns$ (from SRAM calculations)
∴ WAIT STATE

② Chip Enable to data valid: $t_{CE} = 200\,ns > 145\,ns$ ∴ WAIT STATE

③ Output Enable to data valid: $t_{OE} = 75\,ns < 85\,ns$ ∴ OKAY

One wait state (an additional 100 ns) would be sufficient to provide the EPROM with adequate time.

∴ (i) SRAM needs no wait states.

(ii) EPROM needs one wait state.

(f) $\overline{O_4}$ is selected by $AB_{19}\,AB_{18}\,AB_{17} = 100$

→ Connect the same lines to both PPIs

→ like 16-bit EPROM, don't use AB_0 ∴ use AB_2 and AB_1

Don't use AB_0 when calculating # of mirror images

100x xxxx xxxx xxxx x00✓ ⇒ Address of Port A

2^{14} Mirror Images

80000
80008
80010
80018
⋮
9FFF8

2^{14} Memory Mapped I/O Mirror Images

(g)

```
CODE1   SEGMENT
        ASSUME  CS:CODE1 , ES:EXTRA1
        MOV     AX,8000H        ; Port A
        Mov     DS, AX
        Mov     SI , 0
        MOV     AX ,0C000H      ; Start of Array
        MOV     ES , AX
        MOV     DI , 0
        CLD                     ; Increment through array
        MOV     CX , 100        ; no. of Samples
REP1:   MOVSW                   ; get one Sample
        DEC     SI, 2           ; Port address must stay the Same
        CALL    Delay
        LOOP    Rep1
        HLT                     ; or INT #
Delay:  MOV     DX , count      ; "count" gives known sampling rate
Rep2:   DEC     DX
        JNZ     Rep2
        RET
```

```
CODE1     ENDS
 EXTRA1   SEGMENT
Arr1      DW      100 DUP(?)
 EXTRA 1  ENDS
```

(h) The 8255 provides handshaking capabilities which would be useful in this case. The PPI should be configured so that Port A is operating in Mode 1 as an input port. PC4 is a strobe signal from the input device. When a value is received, the PPI can interrupt the CPU and provide the input value.