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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>1</td>
<td>mode set</td>
</tr>
<tr>
<td>D6</td>
<td>00</td>
<td>Mode 0 for Group A</td>
</tr>
<tr>
<td>D4</td>
<td>1</td>
<td>Port A Input</td>
</tr>
<tr>
<td>D3</td>
<td>0</td>
<td>Port C (upper) Output</td>
</tr>
<tr>
<td>D2</td>
<td>0</td>
<td>Mode 0 from Group B</td>
</tr>
<tr>
<td>D1</td>
<td>1</td>
<td>Port B Input</td>
</tr>
<tr>
<td>D0</td>
<td>0</td>
<td>Port C (lower) output</td>
</tr>
</tbody>
</table>

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 00027h

\[
\begin{array}{c|c|c|c|c|c|c|c}
A_5 & A_0 & \text{New Port (94h)} & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
& & \text{Existing Ports (10h)} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
& & (27h) & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}
\]

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 $A_{15} - A_{16}$ 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.

Line $A_0 - 5$ 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 $3 \times 8$ decoder.

\[
\text{LS138}
\]

Any address which fits the following pattern is valid:

\[
A_5, xxx, xxx, xx01, 0100
\]

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.

```
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

MOV AX, 0000h
MOV SI, 0000h
MOV CX, 8000h

OUTER: MOV DX, CX
MOV [SI], AX
ADD SI, 2
MOV CX, OFFFH

INNER: LOOP INNER

\[
\begin{align*}
& \text{INNER LOOP Clocks:} \\
& C_I = \text{FFFFh} \times 17 + 5 \\
& = 65535 \times 17 + 5 = 1114100 \\

& \text{OUTER LOOP Clocks:} \\
& C_O = 8000h \times \left(2 + 14 + 4 + 2 + 17 + C_I\right) + 5 \\
& = 32769 \times (1114143) + 5 \\
& = 36508237829 \\

& \text{TOTAL} = C_O + 4 + 4 + 4 \\
& = 36508237831 \text{ clocks} \\

\end{align*}
\]

\[
\text{Total Time} = \text{clocks} \times \text{period of one clock} \\
= \text{clocks} / \text{frequency of clock} \\
= 7301.65 \text{ seconds} \\
= 2 \text{ hours} 1 \text{ minute} 41.65 \text{ seconds.}
\]

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

\[
8000h \times \text{FFFFh} \times 17 \times 200ns \\
= 7301.3 \text{ sec}
\]
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 clockspeed.

1. \[ \text{MOV CS[2], AX} \] 
   \[ \text{clocks} = 14 + 4 \quad (\text{word operand} \text{ on 8088}) \]

\[ \therefore C_0 = 32768 \times (114 + 4) + 5 \]
\[ = 36508 \text{ 369 901} \]

2. \[ \text{Clock period} = \frac{1}{4.77 \text{MHz}} = 0.2096 \text{ns} \]

\[ \therefore \text{New time} = 36508 \text{ 369 913} \times 0.2096 \text{ns} \]
\[ = 7652.75 \text{ seconds} \]
\[ = 2 \text{ hr} \ 7 \text{ min} \ 33.79 \text{ sec} \]

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 and 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 cannot 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 buffered, 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?
Because the EPROM’s 16-bit wide, the \( A_{15} \) line should not be connected, use \( A_{14} \rightarrow A_{13}, A_{15} \rightarrow A_{14} \), etc.

This line conflicts with existing RAM

The choice of which decoder outputs to use is arbitrary.

The chip’s base

Controls must be connected to the most significant 8 bits of the data bus (DB2-15)

(b) Ram - existing: 1111 11?? ???? ??? = FFFFFFF

\[ 16K = 2 \times 8 \times 1K \]

- new: 110 ? ??? ??? ??? ??? = C0000 - DFFFF

RAM - existing: 0000 0000 ???? ??? ??? ??? = 00000 - 1FFFF

\[ 4K = 2 \times 2 \times 2 \times 1K \]

- new: 00?x ??? ??? ??? ??? ??? = 20000 - 2FFFF

30000 - 2FFFF 2 Mirrors

Images
(e) **SEERAM - READ**

1. **Address valid to data valid:**
   
   \[ T_{\text{CCCL}} - T_{\text{CAVL}} - T_{\text{DCHL}} - T_{\text{DR}} - T_{\text{RTE}} = 3 \times 100 - 30 - 30 - 5 = 185\text{ns} \]

   \[ t_{\text{RA}} < 185\text{ns} \] : OKAY (NO WAIT STATES)

2. **Chip Select Access Time:** (Chip enable to data valid)
   
   \[ T_{\text{CCCL}} - T_{\text{CAVL}} - T_{\text{DCHL}} - T_{\text{RTE}} - T_{\text{DRA}} = 300 - 50 - 30 - 30 - 5 \]

   \[ t_{\text{RAC}} < 145\text{ns} \] : OKAY (NO WAIT STATES)

(f) **Output enable to data valid:**

\[ T_{\text{CCCL}} - T_{\text{CAVL}} - T_{\text{DCHL}} - T_{\text{DR}} - T_{\text{RTE}} = 200 - 70 - 5 - 10 - 30 \]

\[ t_{\text{OE}} < 83\text{ns} \] : OKAY (NO WAIT STATE)

**SEERAM WRITE**

1. **Write Pulse width:**
   
   \[ T_{\text{CCCL}} - 35 = 200 - 35 = 165\text{ns} \]

   \[ t_{\text{WR}} < 165\text{ns} \] : OKAY

2. **Setup Time:**

   \[ t_{\text{WDS}} + T_{\text{CAVL}} + T_{\text{DCHL}} + T_{\text{DR}} - T_{\text{RTE}} = 165 + 10 + 50 + 10 + 30 = 105\text{ns} \]

   \[ t_{\text{WDS}} < 105\text{ns} \] : OKAY

3. **Hold Time (for data)**

   \[ T_{\text{DHS}} - T_{\text{DR}} - T_{\text{RTE}} = T_{\text{CHL}} - 25 - 10 + 30 = 55 - 5 = 40\text{ns} \]

   \[ t_{\text{DH}} < 40\text{ns} \] : OKAY (Calculation was unnecessary)

See question 2 for more details on calculating these values.
(ii) EPROM:
1. Address to data valid: \( t_{AC} = 2.00 \text{ns} > 1.85 \text{ns} \) (from SRAM calculation).
2. Chip Enable to data valid: \( t_{OE} = 2.00 \text{ns} > 1.45 \text{ns} \) : WAIT STATE
3. Output Enable to data valid: \( t_{OE} = 1.75 \text{ns} < 2.0 \text{ns} \) : OK

One wait state (or 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) \( A_4 \) is selected by \( A_0 \), \( A_2 \), \( A_3 = 100 \)

- Connect the same line to both PPI's.
- Since 16-bit EPROM, don't use \( A_0 \); use \( A_2 \) and \( A_3 \).

\[
100X \quad XXXX \quad XXXX \quad XX0V \quad \Rightarrow \text{Address of Port A}
\]

\[
\begin{align*}
2^{14} & \text{Mirror Images} \\
\{ & 0000 \\
\{ & 0008 \\
\{ & 0010 \\
\{ & 0018 \\
\{ & 9FF8
\end{align*}
\]

(q) CODE1

```
SEGMENT
ASSUME CS: CODE1, DS: DATA1

MOV AX, 8000H ; Port A
MOV DS, AX
MOV SI, 0
MOV AX, OCOOH ; Start of Array
MOV ES, AX
MOV DI, 0

CLD
MOV CX, 100 ; Increment through array
JNE Samples
; get one sample
END:
```

```
REP1: MOV SW
DEC SI, 2 ; Port address must stay the same
CALL Delay
LOOP Rep1
HLT
```

```
Delay: MOV DX, count
DEC DI
JNZ Rep2
RET
```

```
REP2: MOV
DEC
JNZ Rep2
RET
```
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.