ELEC 464 : Microcomputer System Design (Fall 1995)

Mid-Term Exam
Solutions

Question 1

Requirement		Guaranteed		Excess	Met
Symbol	Value	Expression	Value		?
t_{SW}	50	$t_{AVRL} + t_{RLSL}$	20 + 80 = 100	50	met
t_{WA}	10	t_{SHAZ}	30	20	met
t_{WW}	80	$t_{SL(w)}$	115	35	met
t_{DW}	80	$t_{DOSL} + t_{SL(w)}$	30 + 115 = 145	65	met
t_{WD}	30	t_{SHDOI}	30	0	met

Question 2

The circuit can be in two states depending on whether the next read will be from port1 or port2. This will be encoded in the variable Q (0=port1, 1=port2). The truth table (showing truth values) is given below. When CS* is not asserted neither output is asserted and the state does not change. When CS* is asserted and WR* is not asserted then either E1 or E2 is asserted depending on the current state. When CS* is asserted and WR* is asserted neither enable is asserted and the next state is Q=0.

Inputs			Outputs		
Q	CS*	WR*	E1*	E2*	Q'
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	0	1	0
1	1	1	0	0	0

The CUPL input could be as follows:

```
Pin  1 =   CLK ;
Pin  2 = ! WR ;
Pin  3 = ! CS ;

Pin 14 =   Q ;
Pin 15 = ! E1 ;
Pin 16 = ! E2 ;

 E1 = ! Q & CS & ! WR ;
 E2 =   Q & CS & ! WR ;
Q.D = ! Q & CS & ! WR  #  Q & ! CS ;
```

Question 3

(a) 1. When the IRQ* pin on the peripheral is asserted IPL0* and IPL2* will be asserted and IPL1* will not. This corresponds to interrupt level 5.

2. All of FC0, FC1, FC2 and AS* must be asserted for the NAND gate output to go low and assert VPA*. The required signal levels are H, H, H, and L respectively.

3. This corresponds to an interrupt acknowledge cycle (see table at top right of page 878 in the Appendix).

4. Since VPA* is asserted instead of DTACK* this is an autovectored interrupt. The exception vector for a level 5 autovectored interrupt is vector 29 and is found at address 116 (decimal) or $074 (hex).

(b) An exception handler for this device could be as follows:

```
* interrupt handler at (hex) 2000

        ORG     $2000

* push (save) registers that will be used

        MOVE.L  A0,-(A7)
        MOVE.L  D0,-(A7)

* add 1 to the 4-byte value at memory location $1000

        MOVE.L  $1000,D0
        ADD.L   #1,D0
        MOVE.L  D0,$1000

* read a byte from memory location $FE0000 and write it to the
* address given by memory location $1000

        MOVE.B  $FE0000,D0
        MOVE.L  $1000,A0
        MOVE.B  D0,(A0)

* restore registers

        MOVE.L  (A7)+,D0
        MOVE.L  (A7)+,A0

* return control to the program that was interrupted

        RTE
```