# UNIVERSITY OF CALIFORNIA, RIVERSIDE

**Department of Computer Science and Engineering**
**Department of Electrical Engineering**
**CS/EE120B – Introduction to Embedded Systems**
**Final**
**December 13, 2001 8:00AM – 11:00AM**

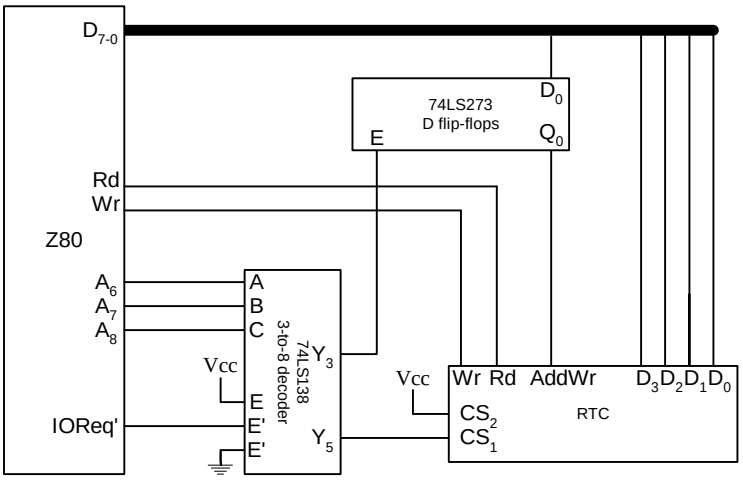
28

**Name:** Solution Key **Student ID#:**
*Please print legibly*

**Lab Section: 21** (WF 2-6):_____ **22** (MW 6-10):_____

(Numbers in parenthesis denote total possible points for question.)

Questions 1 to 4 are based on the following circuit diagram. The circuit shows a Real Time Clock IC (RTC) connected to the Z80 through a decoder and D flip-flops.




1. What is a lowest and highest address for asserting line $Y_5$ of the 74LS138 decoder? (2)

***Answer***

lowest address is **0140 hex**

| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

highest address is **FF7F hex**

| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

2. Explain why the RTC cannot be accessed using memory instructions such as Load and Store? What changes need to be done in order to access the RTC using Load and Store instructions? (2)

***Answer***

In/Out instructions assert the IOReq′ line while the Load/Store instructions assert the MemReq′ line.

The RTC currently can only be accessed using the In/Out instructions because the decoder is enabled with the IOReq′ line. Instead of connecting the IOReq′ line to the E′ of the decoder, connect the MemReq′ line to the E′.

3. Using the I/O instructions

IN destination, source
OUT source, destination

write the instructions needed to write the value $18 into internal register 4 in the RTC. Write comments next to each instruction to say what the instruction is doing. (4)

***Answer***

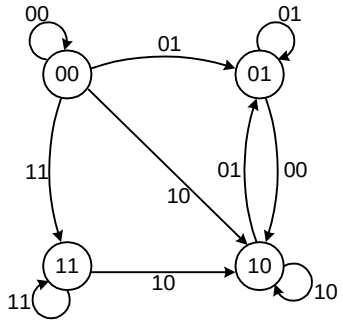

```
OUT $01, $00C0      // assert AddWr by sending a 1 to the ff to prepare to send address
                    // note that $00C0 is one address that will assert the Y3 line.
                    // any address with A6 and A7 being a 1 will work.
OUT $04, $0140      // send address for register 4 to the RTC
OUT $00, $00C0      //deassert AddWr by sending a 1 to the ff to prepare to send data
OUT $18, $0140      // send data to the RTC.
                    // some of you will use the data $14 instead. That's ok.
```

4. Using the same I/O instructions from question 3 above, write the instructions needed to read the value from register 5 in the RTC into the CPU's accumulator (Acc). Write comments next to each instruction to say what the instruction is doing. (4)

***Answer***

```
OUT $01, $00C0      // assert AddWr by sending a 1 to the ff to prepare to send address
OUT $05, $0140      // send address for register 5 to the RTC
OUT $00, $00C0      //deassert AddWr by sending a 1 to the ff to prepare to read data
IN Acc, $0140       // read in the data from the RTC to the Acc
```

5. Synthesize a FSM circuit based on the following state diagram using D flip-flops. Simplify your equations and draw the schematic diagram. (8)




***Answer***

Next-state / Implementation table $Q_{1next}\ Q_{0next}$: = $D_1 D_0$

| | *AB* | | | |
|---|---|---|---|---|
| $Q_1\ Q_0$ | 0 0 | 0 1 | 1 0 | 1 1 |
| 0 0 | 00 | 01 | 10 | 11 |
| 0 1 | 10 | 01 | ×× | ×× |
| 1 0 | ×× | 01 | 10 | ×× |
| 1 1 | ×× | ×× | 10 | 11 |

Excitation equations:

$Q_{0next}$ = $D_0$ K-map:

| $D_0$ | *AB* | | | |
|---|---|---|---|---|
| $Q_1\ Q_0$ | 0 0 | 0 1 | 1 1 | 1 0 |
| 0 0 | 0 | 1 | 1 | 0 |
| 0 1 | 0 | 1 | × | × |
| 1 1 | × | × | 1 | 0 |
| 1 0 | × | 1 | × | 0 |

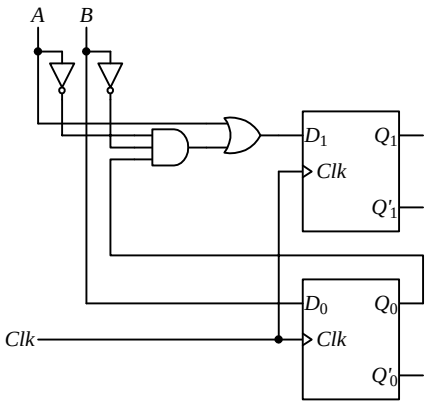
$D_0 = B$

$Q_{1next}$ = $D_1$ K-map:

| $D_1$ | *AB* | | | |
|---|---|---|---|---|
| $Q_1\ Q_0$ | 0 0 | 0 1 | 1 1 | 1 0 |
| 0 0 | 0 | 0 | 1 | 1 |
| 0 1 | 1 | 0 | × | × |
| 1 1 | × | × | 1 | 1 |
| 1 0 | × | 0 | × | 1 |

$D_1 = A + A'B'Q_0$

Circuit:




6. Design a customized datapath and a State-action table for the Mealy FSM that solves the following program. Your datapath should use as few single functional units and registers as possible. Your FSM should have a *start* and a *done* signal. You should first write the high-level pseudo-code that solves the problem.

   Input three positive integers and output the largest of these three integers. (8)

### *Answer*

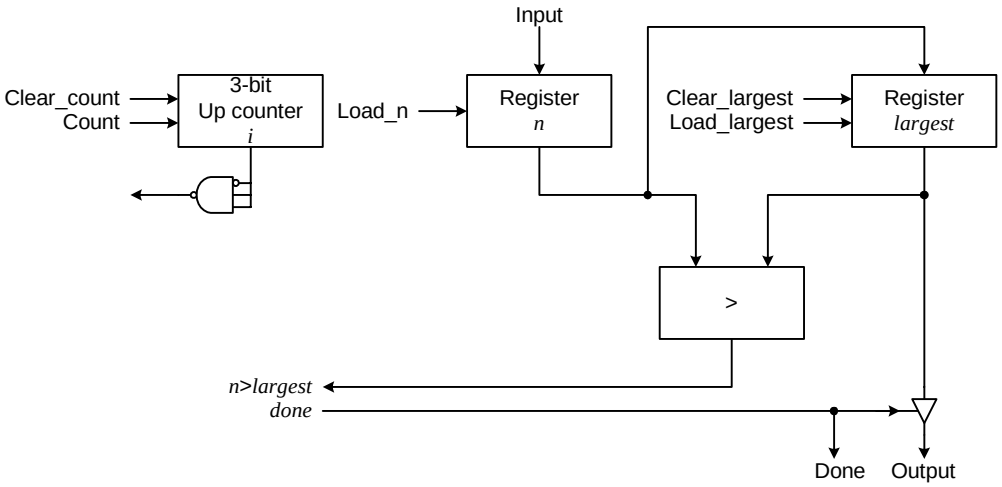
Pseudo-code:

```
largest = 0;
for(i=0; i≠3; i++){
   input n;
   if (n > largest)
      largest = n;
   }
output largest;
```

Datapath needs:

- an up counter for *i*.
- a register for *n*.
- a register for *largest*.
- a greater than comparator
- a not-equal-to-3 comparator

Datapath:




State-action table:

| State | [Condition, Next-state] | [Condition, Action] |
|---|---|---|
| $S_0$ | *Start'*, $S_0$<br>*Start*, $S_1$ | *largest* = 0<br>*i* = 0 |
| $S_1$ | $S_2$ | |
| $S_2$ | (*i*≠3), $S_1$<br>(*i*≠3)', $S_1$ | (*n*>*largest*), *largest* = *n* |
| $S_3$ | $S_0$ | output *largest*<br>output *done* |