(4) Question 1. Give max number of bytes on the stack
21

(4) Question 2. Yes/no. If yes, state where
Yes, in LFP between ldd y and stx y

(2) Question 3. Choose A-F
E (partial credit B)

(2) Question 4. Choose A-F
D

(2) Question 5. Choose A-F
C

(2) Question 6. Choose A-F
A

(4) Question 7. Choose A-F
D

(4) Question 8. Specify RDR
(950,1010)

(2) Question 9. Specify 3 letters from choices A-F
A C E in any order

(2) Question 10. Choose A-H
B

(4) Question 11. ADC resolution
20mV

(4) Question 12
CPOL=0
CPHA=0

(2) Question 13. Choose A-Z, or AA-JJ
B) asyn serial

(2) Question 14. N) full duplex

(4) Question 15. Choose A-Z, or AA-JJ
A) accuracy

(2) Question 16. Choose A-Z, or AA-JJ
J) desk check

(2) Question 17. Choose A-Z, or AA-JJ
D) bandwidth

(2) Question 18. Choose A-Z, or AA-JJ
P) latency

(2) Question 19. Choose A-Z, or AA-JJ
U) polled interrupt

(2) Question 20. Choose A-Z, or AA-JJ
W) private

(2) Question 21. Choose A-Z, or AA-JJ
R) nonintrusive

(2) Question 22. Choose A-Z, or AA-JJ
H) buffered I/O

(4) Question 23. Choose A-F
F

(4) Question 24. Choose yes/no, if no then give an example
No

(4) Question 25. Choose A-Z, or AA-JJ
B

(2) Question 26. Choose A-F
C

(2) Question 27. If you deliver software with bugs, then you should implement a plan allowing customers to receive software patches to fix the errors.
(4) Question 28. Minimal-cost positive-logic address decoder

RAM $6000-$67FF 0110,0XXX,XXXX,XXXX
YourDevice $6800-$6FFF 0110,1XXX,XXXX,XXXX
ROM $E000-$FFFF 111X,XXXX,XXXX,XXXX

Need addresses A15, A11
Select = not(A15)*A11

(6) Question 29. Interface a solid-state relay

$$R = \frac{(5-V_d-V_{OL})}{I_d} = \frac{(5-2.6-0.4)}{2mA} = 1000\Omega$$

(4) Question 30. Write the C code to implement the following equation using fixed-point math

```c
// *****Multiply fixed point Z=X*Y, resolution = 1/16
// Inputs X,Y range from 0 to 4095.9375
// Outputs Z range from 0 to 4095.9375 (return 4095.9375 if overflow)
void FixMult(void){
  unsigned long product;
  product = (IX*IY)/16; // promote to higher precision
  if(product<65535){
    IZ = (unsigned short) product; // demote result
  } else{
    IZ = 65535; // overflow
  }
}
```

(8) Question 31. Design the interface between a 32k by 8-bit PROM and a 6811

**EE345L Spring 2005 Final**

(8) Question 31. Design the interface between a 32k by 8-bit PROM and a 6811 running at 2 MHz. Implement full address decoding for addresses $8000-$FFFF. Assume $t_a = 100$ ns, and $t_b = 20$ns. Assume a 10ns gate delay through any digital logic and the 74HC573 address latch. The PROM has one control signal, CE, which when high reads data at the 15-bit address. Show just the circuit including chip numbers, but not pin numbers. The timing analysis is not required for this question.

In this situation there is one control signal called CE
Select | R/W | CE | Explanation
--- | --- | --- | ---
0 0 | 0 0 | write cycle to another device
0 0 | 1 0 | read cycle from another device
1 1 | 0 0 | write cycle to our device
1 1 | 1 1 | read cycle from our device

Basically in the status table you make the memory do the necessary functions. In this case we want to turn off the device if the cycle is accesses another device. We will also turn it off if a write cycle occurs.

Select | R/W | CE | Explanation
--- | --- | --- | ---
0 0 | 0 | write cycle to another device
0 1 | 0 | read cycle from another device
1 0 | 0 | write cycle to our device
1 1 | 1 | read cycle from our device

Finally, we will activate it if there is a read cycle to our device

Select | R/W | CE | Explanation
--- | --- | --- | ---
0 0 | 0 0 | write cycle to another device
0 1 | 1 0 | read cycle from another device
1 0 | 0 0 | write cycle to our device
1 1 | 1 1 | read cycle from our device

Step 3. During the timing analysis we have to do two things. First, guarantee RDA overlaps RDR. And, second we have to make sure the memory doesn’t drive the data bus during the first half of the cycle (because the 6811 is driving the low address during the first half of the cycle.
To prevent the memory data from colliding we will only drive data out of the memory when E=1. This is called synchronizing the read operation to E. To create a combined table, we expand the status table, adding the E as an input. The data from the status table in entered in the positions where E=1.

<table>
<thead>
<tr>
<th>E</th>
<th>Select</th>
<th>R/W</th>
<th>CE</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>write cycle to another device</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>read cycle from another device</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>write cycle to our device</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>read cycle from our device</td>
</tr>
</tbody>
</table>

To make the control signal low throughout the cycles when we wish to disable the memory, we place additional zeros.
To make the control signal synchronized positive logic, we place a 0,1 in those entries for the cycle we wish to activate. See the book Figure 9.28 and Table 9.9 for the choices that can be entered into the combined table.

<table>
<thead>
<tr>
<th>E</th>
<th>Select</th>
<th>R/W</th>
<th>CE</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>write cycle to another device</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>read cycle from another device</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>write cycle to our device</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>read cycle from our device</td>
</tr>
</tbody>
</table>

The question did not ask to verify timing, but if it did, we would write equations for RDA and RDR. The gate delay in this question is 10ns.

\[
RDA = (\text{rise of CE} + t_a, \text{fall of CE} + t_b)
\]

Because CE is synchronized to the E clock, the rise of CE will be 250+gate delay = 260. Similarly, the fall of CE will be 500+gate delay = 510. \( t_a = 100 \text{ ns}, \text{ and } t_b = 20\text{ns} \)

\[
RDA = (260+100, 510+20) = (360,530)
\]

RDR comes from the 6811. At 2 MHz it is

\[
RDR = (450,510)
\]

Notice that RDA overlaps RDR

Step 4. Develop the logic equation for the control signal and build it with real gates

\[CE = A_{15} \cdot R/W \cdot E\]

Synchronized positive logic, activate PROM when R/W=1 and A_{15}=1

(6) Question 32. Write C code that arms and enables an input capture 7 interrupt

```c
// --------------IC7_Init--------------
// enable input capture 7 interrupts on falling edge
// inputs: none
// outputs: none
```
void IC7_Init(void){
    TSCR1 = 0x80;                // enable TCNT
    TIOS &= ~0x80;               // channel 7 is IC
    TCTL3 = (TCTL3&0x3F)|0x40;   // rising edge IC7
    TIE |= 0x80;                 // Arm IC7
    asm cli                        // enable interrupts
}

Main ; <=start execution here

LowPassFilter

; interrupt pushes CCR,A,B,X,Y,PC

handler