EEL 4712
Midterm 3 – Spring 2011
VERSION 1

Name: Solution

UFID: ______________________

Sign your name here if you would like for your test to be returned in class:

---

**IMPORTANT:**
- Please be neat and write (or draw) carefully. If we cannot read it with a reasonable effort, it is assumed wrong.
- As always, the best answer gets the most points.

**COVER SHEET:**

<table>
<thead>
<tr>
<th>Problem#</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (6 points)</td>
<td></td>
</tr>
<tr>
<td>2 (14 points)</td>
<td></td>
</tr>
<tr>
<td>3 (6 points)</td>
<td></td>
</tr>
<tr>
<td>4 (12 points)</td>
<td></td>
</tr>
<tr>
<td>5 (6 points)</td>
<td></td>
</tr>
<tr>
<td>6 (6 points)</td>
<td></td>
</tr>
<tr>
<td>7 (6 points)</td>
<td></td>
</tr>
<tr>
<td>8 (20 points)</td>
<td></td>
</tr>
<tr>
<td>9 (24 points)</td>
<td></td>
</tr>
<tr>
<td>10 (5 extra credit points)</td>
<td></td>
</tr>
</tbody>
</table>

**Total:**

**Regrade Info:**

---
1) (6 points) Explain how a return instruction for the Small8 determines the return address of a function call.

The return instruction reads the return address from the stack.

2) (14 points) Write the assembly code to implement the following code. You only need to show the actual instructions, not the entire assembly program (e.g. lab test cases). Make sure to list any assumptions.

```assembly
i=0;
while (i < 100) {
    i++;
}
LOAD $00
STAR
CONDITION:
LOAD 100
SETL
SUBR
BPLA O00100
LOAD $01
CURC
ACLR
STAR
CURC
BCC A CONDITION
```

3) (6 points) Draw a schematic of the FPGA circuit that would be synthesized for the following bus structure.

![Bus Schematic](image.png)
4) (12 points) List each step of an opcode fetch. Include all 8-bit, internal bus transfers.
   1) Put PC on internal bus
   2) Store PC into AR
   3) Put PC into internal bus
   4) Store P[1] into A [0]
   5) Read from memory
   6) Put memory data on internal bus
   7) Store memory data into IR

5) (6 points) Fix the metastability problems in the following circuit:

   ![Circuit Diagram]

6) (6 points) What is the main limitation of a mux-recirculation synchronizer and how does a handshake address this limitation?
   
   **Limitation:** Source domain needs to know dest clk frequency to ensure data is held stable long enough
   
   **Handshake:** Uses protocol to tell source domain when data was received

7) (6 points) Which of the following explains why a dual-flop synchronizer cannot be used to synchronize multiple-bit signals?

   (a) Multiple bits take longer to stabilize
   (b) Different bits may stabilize to different values
   (c) Routing delays between clock domains may cause each bit to arrive in different cycles in the destination domain
   (d) b and c
   (e) a, b, and c
8) (20 points) Create a memory initialization file for the following assembly code. Add a comment at the beginning of each instruction. You will likely need to break your answer up into two columns to fit on the page.

INPORTO EQU $FFFF
OUTPORTO EQU $FFFF
MASK EQU $0F

BEGIN:
    LDA A INPORTO
    STA R D
    LDAA MASK
    AND R D
    STA R D
    LDAA VALUE1
    CLCR
    ADCR R D
    STA A OUTPORTO

INFINITE_LOOP:
    CLRC
    BCCCA INFINITE_LOOP

* Data Area
VALUE1: dc.b $55

END BEGIN

Depth = 256;
Width = 8;
Address radix = hex;
Data radix = hex;
% Program RAM Data %
Content
Begin

0000: 28 //LDA A
0001: FF
0002: FE
0003: F1 //STA R D
0004: 84 //LDAA
0005: 0F
0006: 21 //AND R D
0007: F1 //STA R D
0008: 88 //LDAA
0009: 14
000A: 00
000B: F9 //CLCR
000C: 01 //ADCR
000D: F6 //STA A
000E: FF
000F: FE
0010: F9 //CLCR
0011: 00 //BCEP
0012: 10
0013: 00
9) a. (6 points) For the following pseudo-code and schedule for the body of the loop, what is the minimum number of adders and multipliers required by the datapath to execute this schedule, assuming each state corresponds to a different cycle? Do not change the schedule (it is purposely suboptimal).

```plaintext
for (i=0; i < 1000000; i++) {
    a[i] = b[i]*82 + b[i+1]*41 + b[i+2]*55 + b[i+3]*12;
}
```

1. Load b[i], Load [i+1]
2. Load b[i+2], Load b[i+3], Multiply1 (b[i]*82), Multiply2 (b[i+1]*41)
3. Multiply3 (b[i+2]*55), Multiply4 (b[i+3]*12)
4. Add1 (Multiply1 + Multiply2), Add2 (Multiply3 + Multiply4),
5. Add3 (Add1+Add2)
6. Store a[i]

2 adders, 2 multipliers

b. (6 points) Optimize the schedule to reduce the required number of adders by 1, without affecting the total number of cycles. You only need to specify the changes.

More add1 or add2 to cycle 3

c. (6 points) Estimate the performance of the circuit (in # of cycles) with the schedule shown in part a. Assume memory latency is 1 cycle.

\[ 6 \times 1000000 = 6000000 \text{ cycles} \]
d. (6 points) For the following pipelined datapath, what is the approximate performance of the loop in terms of number of cycles? Ignore initialization states. Assume memory bandwidth is sufficient to handle all datapath inputs/outputs.

```
1st iteration: 4 cycles
other iterations: 1 cycle each

4 + 999999 = \lfloor 1000003 \rfloor cycles
```

10) (5 extra credit points) Shift registers are commonly used as delays in pipelined circuits. For deep pipelines, these shift registers can sometimes have a length of greater than 100, which requires many flip-flops in the FPGA. For long shift registers, a more efficient implementation is possible using RAMs. Explain how a block RAM and other additional logic can be used to implement a shift register.