ECE 153a/253
Homework 3

Due: Wed Oct. 27, 2010

Problems:

1. MicroBlaze Processor:

a. Describe how branch delay slot instruction improve the code throughput.

b. If interrupt latency is of key importance to a design, why might support for hardware divide instructions be an issue?

c. Consider a mcroblaze with initial stack pointer at address 0x0a0 on program start. For a series of depth-first recursive function calls on small functions with 1-2 parameters each, estimate how many calls could you support before serious trouble is likely? What trouble?

d. Consider a function call whose parameter is a small 6x6 array of long ints which is declared in the parent as simply “long int”. Where, physically (i.e. on the stack, in the heap, in initialized memory?), is the storage for the array allocated? Where is it stored if declared as “static”?

e. A bug in an otherwise carefully checked embedded system is diagnosed as the stack growing too large during some rare execution traces. To fix this problem, the data segment of the program is moved from the shared BRAM to the much larger DRAM device on the Spartan 3. Despite the much large space, the new version exhibits a variety of odd bugs including missing some interrupts. What might be happening?

f. Describe the what is done in the prologue and epilogue code for a function call to meet the GCC/MicroBlaze ABI. How does this differ from handling an interrupt? (Assume in this case, single source interrupt dispatch).

2. Consider the following code fragment:

```
static int var;
void bar (void){
    var = 0;
    while (var != 255) ;
}
```

Most compilers will notice that no other code can possibly change the value stored in var, and will assume that it will remain equal to 0 at all times. The compiler will therefore replace the function body with an infinite loop similar to this:

```
void bar_optimized (void){
    var = 0;
    while (var != 255) ;
}
```

However, var might represent a location that can be changed by other elements of the computer system at any time, such as a hardware register of a device connected to the CPU. The above code would never detect such a change. Rewrite to show how to prevent that from happening.

3. Construct the Coke Machine 1.0 in C using the nested switch FSM implementation pattern.