ENGN 2910A Homework 02 (200 points) – Due Date: Thursday Oct 8 2015

Professor: Sherief Reda

School of Engineering, Brown University

In this HW you will explore the x86 ISA in 32 bits. You can download the Jasmin x86 assembly interpreter from the class website and use it do these exercise. There are many guides to x86 assembly language. There are many good x86 tutorials available online. One good one is at http://www.cs.virginia.edu/ evans/cs216/guides/x86.html

1. [50 points] Write an assembly program that computes the first N Fibonacci numbers. The input number N should be at memory location 0, and the output computed Fibonacci numbers should be stored in subsequent memory locations starting at location 4. Include a screen shot from your code and memory for $N = 5$.

2. [50 points] Write an assembly program that **recursively** computes the factorial of N. The input number N should be at memory location 0, and the result should be stored at location 4. Include a screen shot from your code for $N = 5$. Note that the recursive requirement means you have to use subroutine instructions such as `call` and `ret`. Include a screen shot from your code and memory. 5 points out of the 50 are reserved for the HWs that use the fewest instructions to implement the functionality.

3. [50 points] The pipelined MIPS processor we student in class is running the following program.

```
add $t0, $t0, $t1
sub $s1, $t2, $t3
and $s2, $s0, $s1
or $s3, $t4, $t5
slt $s4, $s2, $s3
```

a. [10 points] Which registers are being written, and which are being read in the sixth clock cycle?

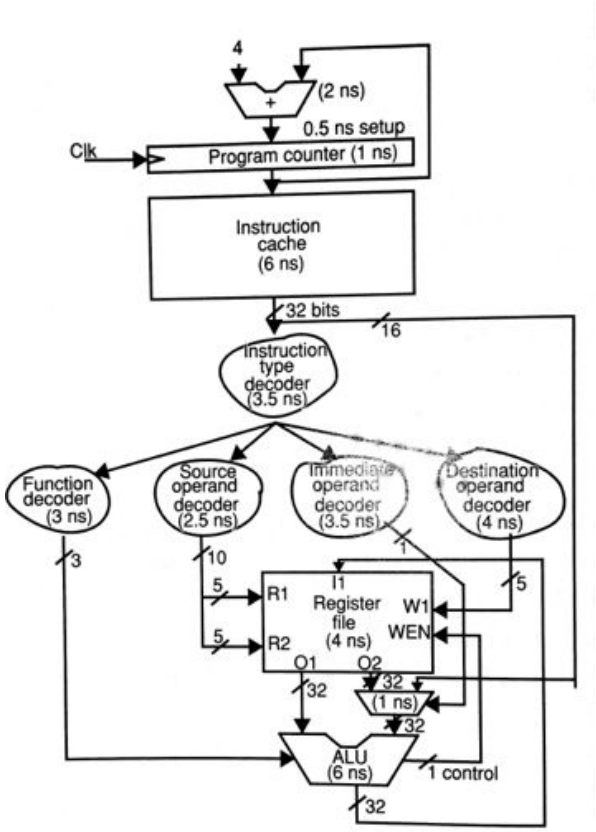
b. [10 points] Identify all Read After Write (RAW) hazards.

c. [10 points] Modify the program to eliminate the hazards by inserting nop instructions in the program to eliminate any RAW hazards. How many cycles does it need to complete the modified program?

d. [10 points] Reorder the instructions in the program, while maintaining correctness, to eliminate the RAW hazards. How many cycles does it need to complete the modified program?

e. [10 points] Explain how a hazard detection unit can eliminate the RAW hazards in the program without the need for software modifications. Comment on the incurred costs of using a hazard detection unit with forwarding.

4. [from Debois *et al.* – 50 points] Consider the non-pipelined implementation of a simple processor that executes only ALU instructions in the figure. The simple microprocessor has to perform several tasks. First, it computes the address of the next instruction to fetch by incrementing the PC. Second, it uses the PC to access the I-cache. Then the instruction is decoded. The instruction decoder itself is divided into smaller tasks. First, it has to decode the instruction type, Once the opcode is decoded, it has to decode what functional units are needed for executing the instruction. Concurrently, it also decodes what source registers or immediate operands are used by the instruction and which destination register is written to. Once the decode process is complete, the register file is accessed (the immediate data are accessed from the instruction itself) to get the source data. Then the appropriate ALU function is activated to compute the results, which are then written back to the destination register. Note that the delay of every block is shown in the figure. For instance, it takes 6 ns to access I-cache , 4 ns to access register file, etc.



a. Generate a 5-stage (IF, ID1, ID2, EX, WB) pipelined implementation of the processor that balances each pipeline stage, ignoring all data hazards. Each sub-block in the diagram is a primitive unit that cannot be further partitioned into smaller ones. The original functionality must be maintained in the pipelined implementation. In other words, there should be no difference to a programmer writing code whether this machine is pipelined or otherwise. Show the diagram of your pipelined implementation.1

b. What are the latencies in (in nanoseconds) of the instruction cycle of the non-pipelined and pipelined implementations? Assume each pipeline register has a delay of 0.5 ns.

c. What are the machine cycle times (in nanoseconds) of the non-pipelined and the pipelined implementations?

d. What is the potential speedup of the pipelined implementation over the original non-pipelined implementation?

e. Which microarchitectural techniques could be used to reduced further the machine cycle time of pipelined design? Identify bottlenecks. Explain how the machine cycle time is reduced.