This homework is based on pipelining concepts described in chapter 6 of the text.

We recommend solving the following problems from the textbook

**Textbook Problems**

- 6.2: a problem on data-forwarding paths
- 6.3: an interesting problem on delayed branch slots
- 6.28: extends the MIPS ISA in an interesting manner. Besides solving the problem as required in the text, attempt to recreate the same functionality (movn and movz) using regular mips instructions based on branching.
- 6.29: describe the performance difference between the two approaches to ISA. Note that 6.28 and 6.29 are problems highlight the difference between a regular branching ISA and a “Conditional Instruction set”. This is the fundamental difference between the ARM ISA and the “traditional” ISAs such as the MIPS.

**Additionally**

Consider the following code-fragment

```c
void addOne(char *p)
{
    int i;
    for(i=0; i<100; i++) p[i]+=1;
}
```

a. Write MIPS assembly code for this. Assume relevant input and output register requirements.
b. Describe the performance of this assembly in the case of a 5 stage pipelined machine described in the text. Carefully describe the timing; the various hazards; and the solutions you have used to fix each hazard. Describe how many cycles does this implementation take to complete. Mention how many of those cycles are basically stalls (bubbles) wasted due to various dependencies.
c. Re-order the assembly code to optimize the solution and reduce some of the dependencies and improve performance. Describe the new performance.
d. Additionally assume the architecture includes a one-bit branch predictor. Describe how you would use it. Describe how much of a speed up do you get due to the branch predictor. How many cycles are now being wasted due to bubbles?