(1) (10 points) The input and output functionality of a computer is implemented using memory mapped locations and spin wait loops. Write the code fragment that the controller would use when a key is pressed on the keyboard.

Important details:
-- Assume that MAL is the language used by the controller for a keyboard, and that the controller has access to its own registers, \$0-\$10.
--The controller's register \$2 contains the ASCII code of the character pressed.
--The memory mapped locations for the keyboard are called `KeyboardStatus` and `KeyboardData`. The status word is defined such that a most significant bit equal to 1 means that a key has been pressed and the device is ready. (Note: this is the same definition as in the text, pages 298-299.)

(2) (8 points) Assume that a MIPS RISC architecture implementation is pipelined in the following steps, where each step takes 1 time unit.

1. instruction fetch and PC update
2. decode and get operands from registers
3. arithmetic operation
4. memory access or returning results to registers

Assuming that the pipeline is empty to start, how many time units does it take to execute the following fragment of TAL code? Justify your answer with a diagram.

```
 .text
addi  $8, $9, 0x0008
mult  $14, $15
lui   $12, 0xab00
ori   $12, $12, 0xcc04
```

(3) (6 points) Assume that a base computer design spends one quarter of its time executing load and store instructions. You come up with an enhanced design that reduces the time to do a load or store instruction to one third of its unenhanced time. Write an expression that gives the speedup for this enhancement (that is, the enhanced computer's speed divided by the base computer's speed). You do NOT need to solve the expression, since calculators are not allowed.

(4) (6 points) If the nonreentrant exception handler used in assignment 9 were to be made reentrant, identify one item that would need to be saved on a stack. Why? (What would go wrong if it were not saved?)

(5) (6 points) The MIPS RISC processor implements delayed branches. Here is some TAL code before assembly. Which instruction(s) can the assembler move into the `beq` instruction's delay slot?

```
     .text
    label1:  sw    $4, 0($12)
             add   $8, $9, $10
             bne   $8, $0, label1
             mult  $14, $15
             lui   $12, 0xab00
             ori   $12, $12, 0xcc04
             add   $18, $16, $20
             beq   $12, $6, label2
             sub   $23, $24, $28
             sll   $10, $12, 2
    label2:  # more code here
```