CS 2734, Computer Organization II
Spring Semester, 2001
Second Examination

1. For this problem, you are to use a xerox of Figure 5.29 (final datapath for the *single*-cycle implementation of MIPS). You will be tracing through the path of the **`beq`** instruction on this single-cycle model. You should use a highlighter to trace the path the instruction and associated data takes through the diagram. (Do *not* show data traveling to "dead-end" components, which will eventually have no effect.) Indicate the values of *relevent* control signals, without highlighting the control line. (Do *not* give control signals that serve to keep "dead-end" paths from having an effect.) **(30)**

   Use the following specific instruction:

   ```
   beq   $t2, $t5, Loop
   ```

   or in machine language form:

   ```
   0x114d0008                          (in hexadecimal)
   000100 01010 01101 00000 00000 001000 (fields in binary)
        4    10    13                 8 (fields in decimal)
   ```

   Start at the left side, showing the PC value coming in, and assume this instruction is read from the Instruction Memory. Show what values are traveling along the different lines, assuming the following initial values:

   (a) **`$t2`** and **`$t5`** are register numbers **`10`** and **`13`** (decimal), respectively.

   (b) Assume that the contents of each of these registers is **`5234`**, so you should assume that the branch is taken. (The branch will be taken because the two register values are equal.)

   (c) Assume the PC has value **`20`** (decimal) initially. On the proper line, give the *final* PC value, assuming the branch is taken. Don't forget to highlight the parts related to the PC as well as the rest of the instruction.

2. For this problem, you are to use one or more xeroxes of Figure 5.33 (final datapath for the *multi*-cycle implementation of MIPS). You will be tracing through the path of the **`lw`** instruction on this multi-cycle model. You should use several colors of highlighters to trace the paths the instruction and associated data takes through the diagram. (Or you can trace these paths using more than one diagram.) Do *not* show data traveling to "dead-end" components, which will eventually have no effect. In particular, do not show the computation of the branch address, which will not be used in this case. **(30)**

   For this diagram, do *not* give the values of control signals.

   Below the diagram, or in some other way, carefully identify *which cycle* (or step) of handling the instruction belongs to each part of the highlighted datapath (just for data, not control). Thus you should identify *Cycle 1*, *Cycle 2*, *Cycle 3*, and perhaps *Cycle 4* and *Cycle 5* (if the instruction uses Cycles 4 and 5).

   Use the following specific instruction:

```
lw $t5,92($t1)
```

or in machine language form:

```
0x8d2d005c                          (in hexadecimal)
100011 01001 01101 00000 00001 011100 (fields in binary)
    35     9    13                92 (fields in decimal)
```

Start at the left side, showing the PC coming in, and assume this instruction is read from the Instruction memory. *Be sure to identify the different cycles.* Don't forget the PC. Show what values are traveling along the different lines, assuming the following initial values:

(a) **`$t1`** and **`$t5`** are registers numbers **`9`** and **`13`** (decimal), respectively.

(b) The contents of register **`9`** is **`200`** (decimal).

(c) The PC has value **`16`**.

3. Suppose we have a *Hamming code* with 10 bits, in positions numbered 1, 2, 3, ..., 10. **(15)**

   (a) Which of these bits are used as check bits and which are used as data bits?

   (b) Which bit positions are checked by the first check bit?

   (c) Suppose that after transmission, the check performed by the first check bit fails. What does this say about the location of the bit in error?

   (d) Which bit positions are checked by the second check bit?

   (e) Suppose that after transmission, not only does the check performed by the first check bit fail, but the check performed by the second check bit also fails. What does this say about the location of the bit in error? (Specifically which bits might be in error?)

4. Using the MIPS multi-cycle implementation as a model, suppose there is a MIPS *exception* because of an *illegal instruction.* **(15)**

   (a) What part of the MIPS hardware (multi-cycle implementation) detects that an illegal instruction has occurred?

   (b) What value ends up in the **EPC register**?

   (c) How does the hardware manage to get the desired value into the **EPC register**?

5. Consider the exception (or trap) handlers described in the book and in class. Three of the final few instructions of the handler might be as follows. Say briefly what each of these instructions is doing. (For the second one below, you must say more that just "add 4 to register **`$k0`**.) **(10)**

```
mcf0     $k0, $14
addiu    $k0, $k0, 4
jr       $k0
```