CS 2734, Computer Organization II
Fall Semester, 2002
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 **`sw`** instruction on this multi-cycle model. You should use several colors of highlighters in one color to trace the path 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, in cycle 2 do *not* show the computation of the branch address, which will not be used in this case. **(35)**

   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:

```
sw $t2,40($t1)
```

or in machine language form:

```
0xad2a0028                          (in hexadecimal)
101011 01001 01010 0000000000101000 (fields in binary)
    43     9    10               40 (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) **`$t1`** and **`$t2`** are registers numbers **`9`** and **`10`** (decimal), respectively.

(b) The contents of register **`9`** is **`104`** (decimal), and of register **`10`** is **`57`** (decimal).

(c) The PC has value **`32`**.

---

3. Consider the following instance of the *Hamming Code*. **(20)**

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Bit value	1	1	1	0	1	0	1	0	1	0	1

(a) Which of the above bits are check bits?

(b) Verify that the rightmost check bit above has the correct value.

(c) Suppose the bit in position **`6`** is transmitted in error, that is, transmitted as a 1 instead of a 0. Show how the Hamming code will be able to correct this error.

---

4. This question is concerned with *exceptions* and *interrupts*, as described in the text for the multi-cycle implementation. **(15)**

(a) Does MIPS use vectored interrupts? (Just "Yes" or "No.")

(b) For the hardware described in the text, what two exceptional conditions are handled?

(c) Why does the hardware subtract 4 from the value in the program counter (the **PC**)?

(d) Where does the result of the subtraction end up?