CS 2734, Computer Organization II
Spring Semester, 2004
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 **`sw`** instruction on this single-cycle model. You should use a highlighter in one color 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.) Then without using a higlighter (you may use another color of highlighter if you wish), show the *relevant* control signals. (Do *not* show control signals that serve to keep "dead-end" paths from having an effect. The input to ALU control is 00, and from that component, the input to the ALU is 010(add).) **(30)**

   Use the following specific instruction: **`sw $t1, 12($t0)`**, or in machine language form:

```
0xad2a000c                          (in hexadecimal)
101011 01000 01001 0000000000001100 (fields in binary)
    43     8     9               12 (fields in decimal)
```

   Start at the left side, showing the PC 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) **`$t0`** and **`$t1`** are registers **`8`** and **`9`** (decimal), respectively.

   (b) The contents of register **`8`** is **`64`** (decimal), and of register **`9`** is **`144`** (decimal).

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

   (Don't forget to handle the PC as well as the rest of the instruction.)

---

2. For this problem, you are to use a xerox of Figure 5.33. (Final datapath for the *multi*-cycle implementation of MIPS.) You will be tracing through the path of the **`beq`** 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. **(30)**

   For this diagram, you do *not* need to 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:

```
beq   $t2, $t5, LabelA
```

   or in machine language form:

```
0x114d0004                            (in hexadecimal)
000100 01010 01101 00000 00000 000100 (fields in binary)
     4    10    13                 4 (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.

(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. *Be sure to identify the different cycles.*

---

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

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Bit value	1	1	0	1	1	0	1	0	1	0	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 **`5`** 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.

(d) Using the original bits above, determine the value of the check bit in position **`0`**. What is this bit used for?

---

4. This question is concerned with *exceptions* and *interrupts*, as described in the text for the multi-cycle implementation. **(30)**

(a) Of the two methods of handling interrups: use of a *Cause Register*, and use of *Vectored Interrupts*, which one does MIPS use? (Just say which one.)

(b) Consider the case of an interrupt because of an *undefined instruction*. Here the Control component detects the problem, and at the end of the second cycle (after decoding the instruction), goes into a new state number 10, which represents the fact that an undefined instruction has occured. This state sets the following control signals:

```
IntCause = 0
CauseWrite = 1
ALUSrcA = 0
ALUSrcB = 01
ALUOp = 01 (which tells the ALU to subtract)
EPCWrite = 1
PCWrite = 1
PCSource = 11
```

Assume the value of the incremented program counter is **`16`**. Use these components to decide what the hardware does. Referring to the xerox of Figure 5.48 from your text, trace through the signals, and decide on their effects.

i. What value is stored into the EPC component and why?

ii. What value is stored into the Cause register, and why?

iii. What value is stored into the PC component, and where does it come from?