CS 2734, Computer Organization II
Spring Semester, 2003
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 **`lw`** 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.) Then write in the values for the *relevant* control signals. (Do *not* give control signals that serve to keep "dead-end" paths from having an effect.) **(30)**

   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)
   ```

   Don't forget the handling of the PC by this instruction. 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`**.

---

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 **`add`** instruction on this multi-cycle model. You should use several colors of highlighters 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. **(40)**

   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:

```
add $s1, $s3, $s1
```

or in machine language form:

```
0x02718820                                (in hexadecimal)
000000 10011 10001 10001 00000 100000     (fields in binary)
     0    19    17    17           32     (fields in decimal)
```

Don't forget the PC. Show what values are traveling along the different lines, assuming the following initial values:

(a) **`$s1`** and **`$s3`** are registers numbers **`17`** and **`19`** (decimal), respectively.

(b) The contents of register **`17`** is **`104`** (decimal), and of register **`19`** is **`57`** (decimal).

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

---

3. This question is concerned with *exceptions* and *interrupts*, as described in the text for the multi-cycle implementation. You are also to use a xerox of Figure 5.48. (The multicycle datapath with the additions needed to implement exceptions.) **(30)**

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

(b) What part of the hardware gives values to the various control signals in case of an exception?

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

(d) Suppose an *undefined instruction* exception occurs, which requires a **`1`** in the Cause register.

On the sheet show the *relevant* control signals needed to handle this exception. Using a highlighter, trace the data flowing though the diagram, and show the values, assuming that the address of the undefined instruction is **`32`**. (There will be new values for the **`PC`**, the **`EPC`**, and for **`Cause`**.)