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

1. Consider the following MIPS code fragment: **(30)**

```
        .data
A:      .word  2, 4, 6, 8, 10, 12, 14, 16, 18, 20
        .text
# insert MIPS instructions here.
```

For insertion at the comment, write MIPS instructions that will do the following:

(a) Put the starting address of `A` into register `$s1`.

(b) Use a loop to add the values of the array `A` and to leave the result in register `$s2`. [You must use a loop for this.]

(c) Print the resulting sum, using `syscall`. [Recall that `syscall` requires `$v0` equal to 1 to print the value in `$a0`.]

Your MIPS code should do what is asked for above and *nothing more*.

2. Write portions of MIPS assembler language that will call a function **`F`** with parameter **`12`**. Then your code should store the returned value in a variable **`i`** corresponding to register **`$s3`**. You should give the complete code for **`F`**, which should add its argument to itself and return the result. Your code should not invoke any syscall, and it should not have any of the rest of the main function except the call to **`F`** and the storing of the returned value into **`$s3`**. The code for **`F`** should be complete. *For full credit, you must follow the MIPS conventions for passing parameters and for returning a value from a function.* (You do not need to explicitly save any registers on the stack.) Thus you should be implementing the following C code: **(20)**

```
/* in main */
i = F(12);  /* use $s3 for i */
. . .

int F(int a)
{
    return a + a;
}
```

3. For this problem you will be including an additional instruction into the *single*cycle datapath described in the text, one not included in the examples in the text. The new instruction is **`addi`** (add immediate). **(20)**

You are to use a xerox of Figure 5.29 (the final datapath for the singlecycle implementation of MIPS). No new datapaths will be needed, but you will need different control signals. The Control unit will be expanded to include an input of 8, the opcode for **`addi`**. Show the control signals as they should be set.

Use a highlighter to show the data lines traversed during execution. You should also write in the values of all signals.

Specifically, suppose the instruction is **`addi $t3, $t2, 45`**, which in machine language form is:

```
214b002d                            (in hexadecimal)
001000 01010 01011 0000000000101101 (fields in binary)
     8    10    11               45 (fields in decimal)
```

In marking the values traveling along data lines, assume that **`$t2 = $10`** holds the value **`51`**. (Note that the ALUOp control can be two 0 bits, so that the input to the ALU from ALU control will then be 010, that is, an add.)

4. 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.*

5. Consider the xerox of Figure 6.41 from your text, showing pipelined execution where forwarding is required. There are two diagrams, an upper one and a lower one, showing successive cycles. **(20)**

   The pipelined MIPS machine is in the middle of executing the following sequence of instructions:

```
sub     $2, $1, $3
and     $4, $2, $5
or      $4, $4, $2
add     $9, $4, $2
```

(a) The above instructions require *four* instances of data forwarding. Say exactly which values are being forwarded, and to which instructions.

(b) In the lower diagram shows details of one forwarding step. Which forwarding is being handled in this cycle (Clock 4).

(c) Explain in detail exactly how the proper forwarded value is getting into the ALU.

6. Consider the xerox of Figure 6.47 from your text, showing pipelined execution, where a *stall* is needed. The instructions in execution are: **(20)**

```
lw      $2, 20($1)
and     $4, $2, $5
or      $4, $4, $2
add     $9, $4, $2
```

(a) After which instruction is a stall needed? Why is a stall needed in this spot? How many cycles must the stall last?

(b) What unit detects the need for a stall, and how does it know a stall is needed?

(c) In order to actually carry out a stall, two separate (distinct) actions must be taken. Describe them in some detail.

(d) Describe briefly what value is forwarded from the lw instruction and how it is forwarded to where it is needed.

7. Consider the xerox of Figure 6.52 from your text, showing pipelined execution, where a *stall* is needed. The instructions in execution are: **(20)**

```
sub     $10, $4, $8
beq      $1, $3,  7 # branches to lw below
and     $12, $2, $5
or      $13, $2, $6
add     $14, $4, $2
. . .
lw       $4,  50($7)
```

The figure assumes the branch is taken, that is control transfers from the **`beq`** to the **`lw`** instruction.

(a) What hardware changes does the book show to limit a stall on **`beq`** to just 1 cycle (if taken) and to avoid a stall (if not taken)?

(b) Using Figure 6.52, describe how the stall is inserted.

(15)

8. Consider the xerox of Figure 7.8 from the text, showing the diagram for the cache memory of a DECStation 3100.

   (a) What is the size of the field used for an *index* (that is, for a cache table entry)? From this, say how many words of data this cache holds.

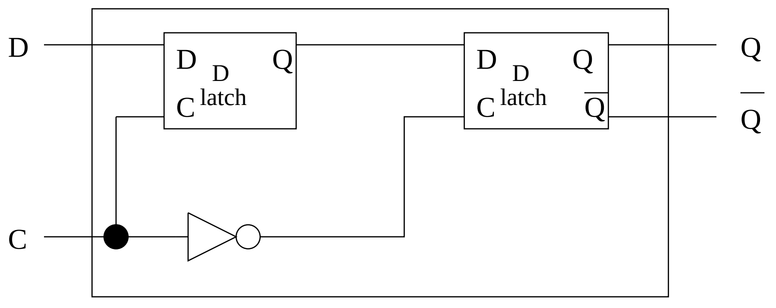
   (b) Why does this hardware not pay attention to the low order 2 bits?

   (c) Why is the *Tag* field in the cache 16 bits wide?

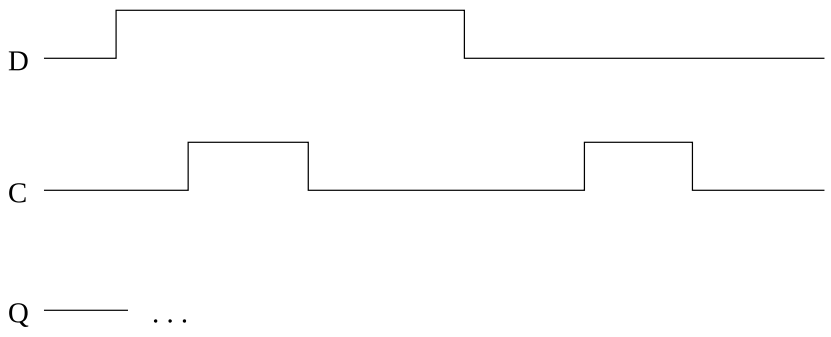
   (d) Describe the process that occurs with a cache *hit*. How does the hardware know there is a hit? (Be careful, there are *two* conditions.) How does the hardware manage to get to the correct data without some kind of search?

   (e) Describe what the text says happens on an instruction cache *miss*. (Four steps.)

9. Consider the following diagram of a D flip-flop: **(15)**



D flip-flop



This D flip-flop is constructed from two D latches. Recall that a D latch lets the signal D go through if the clock C is asserted (the D latch is *open*), and the output does not change if the clock C is deasserted (the D latch is *closed*). Such a D latch is said to be *transparent*.

   (a) Give the output Q of the D flip-flop as it would appear in the figure above. (That is, fill in the output signal Q.)

   (b) Explain very carefully why the output is what it is. (Your explanation should refer to the specifics of the diagram above, including the NOT gate and the two D latches.)