This homework assignment must be completed by you alone. You may not copy from others. However, you may study with others or use external references to determine general solutions. Then you must complete the problems as your own work, not copying others’ work.

Questions about this homework should be addressed to your TA first. You can find your TA’s email at the class website: http://www.cise.ufl.edu/~mssz/CompOrg/TA-hours.html

This homework has four parts: (I) Regular Questions, (II) Pipeline Hazards, (III) Pipeline Scheduling (IV) Extra Credit.

Complete all the work you can – there is no penalty for guessing.

Parts I-IV will be submitted as hardcopy in class. No electronic submissions, please.

Part I. Regular Questions (hand in as hardcopy) [10 points total]

1. Vocabulary: (terms you need to know to discuss the subject intelligently) – Define the following terms using 1-3 sentences (and a diagram, if needed): [2 points each]
   a. Data Hazard (in a pipeline)
   b. WAR (data dependency)
   c. WAW (data dependency)
   d. RAW (data dependency)
   e. Structural Hazard (in a pipeline)

Part II. Pipeline Hazards [10 points total]

2. Data Hazard Identification: List and describe the data hazards in the following fragment of MIPS code: [5 pts total]
   
   L.1  subi $t0, $zero, 24 ;
   L.2  addi $t1, $t0, 28 ;
   L.3  lw  $s1, 0($t1) ;
   L.4  sw  $s1, 0($s1) ;

   Document (comment) each line of the above code fully to get full credit, then list where each data dependency occurs and what its type and variable(s) is(are).

   For example,  L.4: RAW on $t0 from L.1
3. **Control Hazard Explanation:** Given the following MIPS code fragment, (a) identify the statements in which control hazard(s) could occur (use the labels L.5 through L.8 that are provided), then (b) describe each data hazard and control hazard in terms of why it occurs, and what its effect on the MIPS pipeline IF-ID-EXE-MEM-WB would be if the branch was taken or not-taken. [5 pts total]

L.5  subi $t0, $zero, 24 ;
L.6  subi $t1, $t0, 28 ;
L.7  beq $t0, $t1, L.22 ;
L.8  bne $t1, $t1, L.6 ; # Two uses of “t1” are intended

**Part III. Pipeline Scheduling Problems** [20 points total]

4. **Pipeline Hazards and Scheduling:** Given one pass through the following MIPS code fragment, (a) identify the statements in which data hazards occur (as in Problem 2, above), then (b) draw a schedule for the execution of this code fragment on the MIPS pipeline IF-ID-EXE-MEM-WB if we assume that the branch is not-taken, and (c) compute the CPI of this code fragment running on the five-segment MIPS pipeline. Assume that the base address has been previously assigned to $s0. [10 pts total]

L.9  lw  $t0, 24($s0) ;
L.10 lw  $t1, 28($t0) ; # yes, this is $t0, not $so
L.11 sub  $t1, $t0, $t1 ;
L.12 sw  $t1, 32($t1) ;
L.13 beq $t1, $t0, L.10 ;

5. **Pipeline Hazards and Scheduling:** Given all iterations of the following MIPS code fragment, (a) identify the statements in which data dependencies occur and list these dependencies (as in Problem 2, above), then (b) draw a schedule for the execution of this code fragment on the MIPS pipeline IF-ID-EXE-MEM-WB if we assume that the branch is not taken, and (c) compute the CPI of this code fragment running on the five-segment MIPS pipeline. Assume that initial base address in $t0 is assigned before execution. **Hint:** Be careful in your assumptions; check out the lw ... sw instructions. [10 pts total]

L.10  subi $t1, $zero, 10 ;
L.11  addi $t1, $t1, 1 ; [the value “1” is correct...]
L.12  lw  $t2, 32($t1) ;
L.13  addi $t2, $t2, $t0 ;
L.14  beq $t2, $t1, L.11 ;
L.15  sw  $t2, 32($t2) ;

**NOTE:** This is not as simple as Problem 4 – you will have to determine (i) what the code is doing (it will help to comment and flowchart the MIPS code), then determine (ii) under what condition(s) the branch would be taken or not-taken (if such conditions exist), then (iii) determine dependencies, and finally (d) solve for each case (i.e., taken or not-taken).
Part IV. Extra Credit [30 points total]

6. Discuss your answers in detail (with diagram, if needed) to get full credit: [10 points each]

(a) Under what assumptions do we assume *branch-taken* when we do branch prediction in MIPS (as discussed in class)? How does this affect the Branch Delay Slot?

(b) What is the difference in MIPS five-segment pipeline performance between *branch-taken* and *branch-not-taken* assumptions? Why does this occur (discuss in detail)?

(c) Suppose our branch prediction mechanism is *assume branch-taken*, and that for a program $P$ running on the MIPS pipeline $M$ (IF-ID-EXE-MEM-WB), this prediction is correct 52% of the time. If $P$ has 22,487 instructions, of which 23 percent are branches, and $P$ has 2.5% of its instructions that incur an exception in the EXE stage (at a cost of 2 cycles to handle the exception) then what is the estimated CPI of $P$ running on $M$?

*Hint:* Use the pipeline performance model presented in the “chalk talk” in class, and in the Web notes on Pipeline Performance...