CS230: Digital Logic Design and Computer Architecture
Tutorial 06, [Mon 30 Sep, Tue 01 Oct, Thu 02 Oct]

Concepts tested: Control Hazards

1. **Compatibility of ideas:** Among the following ideas for dealing with control hazards, which pairs are NOT compatible with one another?

   - 3-stage branch completion
   - 2-stage branch completion
   - Assume branch not taken
   - Branch prediction
   - Branch target buffer
   - Delayed branches

2. **Filling the branch delay slot:**

   Consider the following (incomplete) code. It is supposed to execute on a MIPS machine with a 5-stage pipeline implementation (as discussed in the lectures). It has a 2-stage branch completion scheme, with a branch delay slot of exactly one instruction.

L.no:			
L1		_____	_____ # Answer below, not here
L2		beq	$s1, $zero, NO_IF
L3		nop	# delay slot is initially empty
L4		addi	$s2, $s2, 1
L5		add	$a0, $s3, $zero
L6		jal	FUNC
L6.5		nop	
L7	NO_IF:	_____	_____ # Answer below, not here

   In the following sub-questions, we will consider various possibilities to schedule the delay slot (i.e. fill a useful instruction in the delay slot). Note that these possibilities are *exclusive* of one another. Also, briefly explain your answer in each case, in a sentence or two.

   (a) Give an example non-branch instruction in line L1, which the compiler *can safely* schedule in the delay slot.

   (b) Give an example non-branch instruction in line L1, which the compiler *cannot* schedule in the delay slot.

   (c) Fill L7 with a non-branch instruction such that the compiler *can safely* schedule the instruction in L7 in the delay slot.

   (d) Fill L7 with a non-branch instruction such that the compiler *cannot* schedule the instruction in L7 in the delay slot.

   (e) Fill L7 with a non-branch instruction such that the compiler *can safely* schedule the instruction in L4 in the delay slot.

   (f) Fill L7 with a non-branch instruction such that the compiler *cannot* schedule the instruction in L4 in the delay slot.

   (g) Fill L1 & L7 with non-branch instructions such that the instruction in L1 can be safely scheduled in the delay slot, but does not end up enhancing the performance.