1. (17 pts) Consider this sequence of instructions.

```
        CC 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
1 lw $t0, 4($sp)
2 lw $t1, 8($sp)
3 add $t2, $t0, $t1
4 sub $t2, $t2, $t1
5 addi $t2, $t2, 1
6 sw $t2, 12($sp)
```

(a) Draw a pipeline diagram, similar to the ones in the lecture notes we have been drawing in class, showing the stages of each instruction as it moves through the pipeline. Assume *the only way we have of resolving data hazards is to stall the pipeline by inserting bubbles.*

(b) What is the average IPC (instructions per clock). Show your work. *Hint: bubbles are* nop *instructions which move through all five pipeline stages just like any other instruction.*

(c) Assume all of these data hazards *could* be eliminated by redesigning the hardware, so no bubbles would be required. In that case, the average IPC would increase. Calculate the IPC for this sequence of instructions under ideal conditions, i.e., all data hazards are eliminated by clever hardware design. Show or explain your work. You do not need to include the pipeline diagram in your solution.

(d) How many fewer clocks, expressed as a percentage relative to the sequence of (a), does the ideal sequence of (c) take. For example, if $clocks_a$ and $clocks_c$ are the number of clock cycles to execute the sequences of Exercises (a) and (c), then the question is to calculate $|clocks_a - clocks_c| \div clocks_a$. Show or explain your work.

(e) How much faster, expressed as a percentage relative to sequence (a), is the instruction sequence of Exercise (c), i.e., the instruction is asking you to calculate $|clocks_c - clocks_a| \div clocks_c$. Show or explain your work.

2. (17 pts) Consider this sequence of instructions, which requires bubbles because there are data hazards.

```
            CC 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
1 lw $t0, 4($sp)
2 lw $t1, 8($sp)
3 add $t2, $t0, $t1
4 sw $t3, 12($sp)
5 lw $t4, 16($sp)
6 addi $t4, $t4, -1
```

For this sequence of instructions:
(a) As it is, determine how many bubbles are required if we have no way of resolving data hazards other than stalling. Note: you do not have to draw the pipeline diagram for your solution, but you may still need to draw it to determine how many bubbles are required.

(b) What is the IPC? Show your work.

(c) One way we discussed of eliminating bubbles is instruction reordering. It is possible to reorder these instructions to eliminate all of the data hazards. Do so by writing the numbers of the original instructions in the proper order, e.g., some - thing like 1, 5, 2, 3, 6, 4 for the reordered sequence. Note: assume there are instructions above and below instructions 1 and 6; in particular, some instruction above instruction 1 has loaded $t3 with the value that sw will write to memory. Note: there is more than one way to reorder these instructions to eliminate all hazards. To reduce the number of ways this can be done for grading purposes, for any instructions which are not reordered (i.e., if 1 and 2 in the original sequence are not moved and no instruction is inserted in between them so they remain 1 and 2 in the reordered sequence),

then keep the instructions in the same sequence, i.e., do not write 2, 1, ... as your solution. Or, as another example, if you decide to move 5 up, do not move it above 1 (so your sequence would be 5, 1, 2, ...), *unless* that is absolutely required to eliminate a bubble that would otherwise occur. Both of these *would* be the beginnings of correct sequences, but they will be counted as incorrect. If you do it this way, I believe there is only one reordering that will be counted as correct[1].

(d) After reordering, what is the IPC? Show your work

(e) How much faster is the reordered instruction sequence expressed as percentage relative to the sequence requiring bubbles? That is, if $clocks_a$ is the number of clocks to execute the instruction sequence of Exercise (a) and $clocks_c$ is the number of clocks to execute the sequence of Exercise (c), then you must calculate $|clocks_c - clocks_a| / clocks_c$. Show or explain your work.

3. (16 pts) In the Ch. 4 lecture notes starting on p. 38, we discussed EX and MEM data hazards. On p. 38, I categorized the four type of EX and MEM hazards.
1. Type 1a EX data hazard: the conflict is between the register of an instruction that is writing to a *destination register* and the *first source register* of the following instruction. For an EX hazard, the value that is needed is coming from the EX stage of the prior instruction.
Example,
sub **$t0**, $t1, $t2 # $t0 is destination register
and $t3, **$t0**, $t4 # $t0 is first source register

2. Type 1b EX data hazard: the conflict is between the register of an instruction that is writing to a *destination register* and the *second* source register of the following instruction
sub **$t4**, $t1, $t2 # $t4 is destination register
and $t3, $t0, **$t4** # $t4 is second source register

3. Type 2a MEM data hazard: the conflict occurs between an instruction that is in the MEM stage which has a value which will be written to a *destination register* when the instruction moves to the WB stage and the *first source register* of the instruction following the following instruction. For a MEM hazard, the value that is needed is coming from the MEM stage of the instruction preceding the preceding instruction.

sub **$t0**, $t1, $t2 # $t0 is destination register
ori $t7, $t7, 1 # For a MEM hazard, there must be an intervening instruction
and $t3, **$t0**, $t4 # $t0 is first source register

4. Type 2b MEM data hazard: the conflict occurs between an instruction that is in the MEM stage which has a value which will be written to a *destination register* when the instruction moves to the WB stage and the *second source register* of the instruction following the following instruction.
sub **$t4**, $t1, $t2 # $t4 is destination register
ori $t7, $t7, 1 # For a MEM hazard, there must be an intervening instruction
and $t3, $t0, **$t4** # $t4 is first source register

For the final exam, you should learn how to quickly identify these hazards, so study the examples on p. 39 and above, where the conflicting registers are in boldface. Next (see pp. 29-30), we discussed how forwarding (also called bypassing)
can be used to resolve data hazards. See p. 45 of the lecture notes which shows the final pipeline design. The *forwarding unit* in the lower right corner is responsible for forwarding the needed values so they can become either the
A input of the ALU (the top input) or the B input (the bottom input). For the instructions above, the hazards will be resolved in this manner,

1. Type 1a EX data hazard. The forwarding unit will forward *ALUResult*, which is $t1 - $t2, produced in clock 3 of the sub instruction so that $t1 - $t2 becomes the **A** input to the ALU for the and instruction in clock 4. For an EX hazard, the value that is needed by the and instruction is coming from the EX stage of the prior instruction. 1 2 3 4 5

```
                        1  2   3    4    5
      sub $t0, $t1, $t2 IF ID EX ME WB
\
and       $t3, $t0, $t4 IF ID EX ME WB
```

2. Type 1b EX data hazard. The forwarding unit will forward *ALUResult*, which is $t1 - $t2, produced in clock 3 of the sub instruction so that $t1 - $t2 becomes the **B** input to the ALU for the and instruction in clock 4.

```
                    1   2   3    4    5
sub $t4, $t1, $t2 IF ID EX ME WB
\
and $t3, $t0, $t4 IF ID EX ME WB
```

3. Type 2a MEM data hazard. The forwarding unit will forward *ALUResult*, which is $t1 - $t2, produced in clock 3 of the sub instruction and carried forward to the sub MEM stage, where $t1 - $t2 will become the **A** input to the ALU for the and instruction in clock 5.

```
                    1   2   3    4    5
sub $t0, $t1, $t2 IF ID EX ME WB
ori $t7, $t7, 1         IF ID   \    ME WB
and $t3, $t0, $t4          IF    ID  EX  ME WB
```

4. Type 2b MEM data hazard. The forwarding unit will forward *ALUResult*, which is $t1 - $t2, produced in clock 3 of the sub instruction and carried forward to the sub MEM stage, where $t1 - $t2 will become the **B** input to the ALU for the and instruction in clock 5.

```
                     1   2   3    4    5
sub $t4, $t1, $t2  IF ID EX ME WB
ori $t7, $t7,    1       IF ID   \    ME WB
and $t3, $t0, $t4           IF   ID  EX  ME WB
```

Now to the exercise. Consider the sequence of instructions shown below. Assume the instructions are pipelined under ideal conditions, i.e., assume there are no hazards so each subsequent instruction enters the pipeline in the clock cycle following the previous instruction. However, there are data hazards present, so the exercise is to document all data hazards as they exist in the ideal pipelined sequence, i.e., specify the data hazard, but do not resolve it. For each hazard, specify the involved instructions (e.g., 1 and 3) and the type of the hazard (e.g., type 1a, 1b, 2a, or 2b). You do not need to include the pipeline diagram in your solution. For example, the first hazard is between instructions 1 and
3 and it is a Type 2a MEM hazard (write **Instructions 1 and 3: Type 2a MEM Hazard** for your answer). Why? lw obtains the value in the MEM stage that will be written to $t0 in lw's WB stage and that value is needed as the first source register for addi when it reaches the EX stage.

```
1  lw     $t0, 0($sp)
2  and   $t9, $t9, $t9
3  addi  $t0, $t0, 1
4  sub   $t1, $t2, $t0
5   ori    $t1, $t1, $t0
6  andi  $t1, $t1, 0x80
```