Computer Science 252 Quiz #1 **Name:_____**

1. The cost/performance of two microprocessors is to be examined, each running the same instruction set.

The first option is a Galium Arsenide (GaAs) microprocessor. A GaAs wafer that is 10 cm (≈4 inches) in diameter costs $1000. The manufacturing process creates 4 defects per square centimeter. The microprocessor fabricated in this technology is expect to have a clock cycle rate of 250 MHz, with an average clock cycles per instruction of 1.5 if we assume an infinitely fast memory system. The size of the GaAs microprocessor is 1.0 cm by 1.0 cm. (Assume α is 2.0 for GaAs.)

The second option is a CMOS microprocessor. A 15 cm (≈6 inch) wafer with 2 defects per square centimeter costs $500. The 1.0 cm by 2.0 cm microprocessor executes multiple instructions per clock cycle to achieve an average clock cycles per instruction of 0.75, assuming an infinitely fast memory while achieving a clock rate of 50 MHz. (The microprocessor is larger because of on chip caches and executing multiple instructions per clock cycle.)

You found that a memory system can be built for either microprocessor, and that the average extra time for memory accesses per instruction (for instructions and data) is the same for each microprocessor: 10 ns.

a) What is the cost of an untested GaAs die for this microprocessor? (Use information from the book to supplement the information here if anything is missing.) Show your work. **[4 points]**

b) What is the cost of an untested die for the CMOS microprocessor? (Use information from the book to supplement the information here if anything is missing.) Show your work. **[4 points]**

c) What is the native MIPS rating of each microprocessor? Which is faster and by what percent? Show your work. **[6 points]**

d) Based solely on costs of the microprocessors calculated above, what is the cost/performance of each option? What is the ratio of cost/performance of the CMOS to GaAs microprocessors? **[4 points]**

Name:_____

2. A *predicted-branch* instruction is as a way to improve the performance of delayed branches. The idea is to include another branch instruction that indicates that the instruction in the delay slot should be aborted if the branch is mispredicted. The advantage of predicted branches is that the delay slot can **always** be filled, since the branch can abort the contents of the delay slot if mispredicted. The compiler need not worry about whether the instruction is OK to execute when the branch is mispredicted.

A simple version of predicted branches does not execute the instruction in the delay slot if the branch is not taken; assume this type of predicted branch, which we call *branch likely*, is added to DLX. Assume that 27% of the branch-delay slots are filled with useful instruction using standard delayed branches, and that the rest of the slots are filled using branch likely instructions where the instruction put into the delay slot is from the target of the branch.

a) What is the relative effectiveness of the schemes? First assume the likely branches are taken with the frequency quoted in the book. Next assume that the compilers are changed to try to change the code so that the branch likely instructions are taken 80% of the time. Show the percentage of delayed slots filled and percentage of delayed slots **usefully** filled for GCC. **[8 points]**

b) How much faster would a DLX machine run GCC that added this style of branch, assuming there is no clock-speed penalty compared to a machine with only delayed branches? Show the results for both assumptions on fraction of taken branch likely instructions. **[8 points]**

Name:_____

3. You are charged with investigating a new addressing mode that allows one source operand to be in memory for ALU instructions on DLX. To offset this increase in complexity you restrict all memory addressing to be register indirect only (e.g., no displacement addressing.)

a) Propose a change to the DLX pipeline in Chapter 6 to include this addressing mode. Draw the new pipeline and explain the changes. **[8 points]**

Stage	ALU instruction	Load or store instruction	Branch instruction
IF	IMAR←PC; IR←Mem[IMAR]; PC←PC+4;	IMAR←PC; IR←Mem[IMAR]; PC←PC+4;	IMAR←PC; IR←Mem[IMAR]; PC←PC+4;
ID	A ← DMAR ←Rs1; B ← MDR ←Rs2;	A ← DMAR ←Rs1; B ← MDR ←Rs2;	A ← DMAR ←Rs1; B ← MDR ←Rs2;
MEM	MDR2 ← Mem[DMAR];	MDR2← Mem[DMAR]; or Mem[DMAR]←MDR;	if (cond) PC←ALUoutput;
EX	ALUoutput ← A *op* B; or ALUoutput←A *op* $((IR_{16})^{16}\#\#IR_{16..31})$; or ALUoutput ← MDR2 *op* B; or ALUoutput ← MDR2 *op* $((IR_{16})^{16}\#\#IR_{16..31})$;	MDR3 ← MDR2	ALUoutput←PC + $((IR_{16})^{16}\#\#IR_{16..31})$; cond←(Rs1 *op* 0);
WB	Rd ← ALUoutput;	Rd ← MDR3;	

b) What new data hazards, if any, are created by this addressing mode? Give an example of each, classify the type of data hazard, and show the conflict using the data above.

c) List all changes or additions must be made to the hardware to accommodate this mode. **[4 points]**

d) List all questions that must be answered to estimate the impact of this change on performance. **[4 points]**

Extra credit for no points: if you have the time, try to see if you can find evidence from the statistics on DLX or other architectures to answer the questions on the impact on performance.