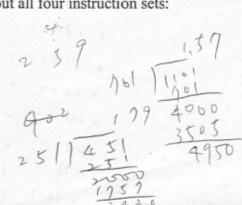1. (14%) [performance] There are two possible ways to improve the performance of a computer: (1) make multiply instructions run _four times faster_ than before, or (2) make memory access instructions run _two times faster_. You repeatedly run a program that takes 100 seconds to execute. Of this time, 20% is used for multiplication, 50% for memory access, and 30% for other tasks.
   (a) Which option will give you the best speedup?
   (b) What will the speedup be if both improvements are made?
   <Note> speedup = (run time before improvement) / (run time after improvement)

2. (36%) [instruction set and performance] Consider the following MIPS code segment:

```
        addi $a1,$zero,200
        add  $t0,$zero,$zero   # i=0
loop1:  add  $t1,$t0,$t0       # $t1=i*2
        add  $t1,$t1,$t1       # $t1=i*4
        add  $t2,$a0,$t1       # $t2=address of array[i]
        sw   $zero,0($t2)
        addi $t0,$t0,1
        slt  $t3,$t0,$a1
        bne  $t3,$zero,loop1
```

   (a) Suppose addi, add and slt take one cycle, sw takes four cycles, and bne takes two cycles to execute. How long does it take to execute the above code segment on a processor with a 100 MHz clock?
   (b) What is the CPI of this code segment on the above processor?
   (c) If we multiply $a1 by 4 in the very beginning, then we can add $t0 by 4 instead of doing $t0*4 inside the loop. Show the improved code.
   (d) Find the CPI of the improved code and show that the compiler is a factor affecting CPI.

3. (10%) [instruction set design] Compare the memory efficiency of four different styles of instruction sets for one code sequence. The architecture styles are:
   • Accumulator
   • Memory-Memory – All three operands of each instruction are in memory
   • Stack – All operations occur on top of the stack. Only push and pop access memory and all other instructions remove their operands from the stack and replace them with the result. The implementation uses a stack for the top two entries, accesses that use other stack positions are memory references.
   • Load/Store – All operations occur in registers and register-to-register instructions have three operands per instruction. There are 16 general purpose registers and register specifiers are 4 bits long.

   To measure memory efficiency, make the following assumptions about all four instruction sets:
   – The opcode is always 1 byte
   – All memory addresses are 2 bytes
   – All data operands are 4 bytes
   – All instructions are an integral number of bytes in length
   – There are no optimizations to reduce memory traffic

(a) For the following code segment, write the code sequence for each of the four instruction sets. Suppose variables A, B, and C are initially in memory.

A = B + C;

(b) For each code sequence, calculate the instruction bytes fetched and memory-data bytes transferred. Which architecture is most efficient as measured by code size? Which architecture is most efficient as measured by total memory bandwidth required (code + data).

4. (20%) [arithmetic unit] Calculate the relative performance of adders. Assume that hardware corresponding to any equation of OR or AND terms (one level logic) takes one time unit T. Equation that is in sum-of-product form (two level logic) and that has each sum or product term having at most 5 inputs takes 2T. Calculate the relative speeds of a 64-bit adder using
   (a) ripple carry
   (b) ripple carry of 4-bit groups that use carry lookahead
   (c) carry lookahead of 4-bit groups
   Give the delay of the most significant carry bit only.

5. (20%) [floating point representation]
   (e) Let the 32-bit pattern, 1001100101110...0, represent a single precision floating point, where bit 31 is a sign bit, bits 30 to 23 represent the exponent, bit 22 to 0 represent the significant, bias 127 is used for exponent, and hidden 1 is used. What is the number?
   (f) By the above representation, show how to represent a decimal floating point 0.0472.