1. (20 points) Consider the following fragment of C code:

\[
\text{for } (i = 0; i <= 100; i + +) \\
A[i] = B[i] + C;
\]

Assume that A and B are arrays of 32-bit integers, and C and i are 32-bit integers. Assume that the arrays A and B start at addresses 0 and 4000. The value of C is stored at address 6000. The memory location 6004 is used to store the value of i. All addresses are in decimal. Assume that no values are kept in registers between iterations of the loop.

Write the code using MIPS instruction set. How many instructions are executed dynamically? How many memory data references will be executed? What is the code size in bytes?

2. (20 points) Consider the following piece of MIPS code. Assume the first instruction starts at address 0x800 (indicated as a Hex address). For each of these instructions, indicate the correct instruction format (I, R or J) being used. Also, indicate the detailed encoding for each instruction with their respective values. You can ignore the actual values for opcode, shamt and func fields.

```
DADDI    R1, R0, #128
DADD     R2, R0, R0
Loop:    LD      R3, 0x1000(R2)
           DADDI  R3, R3, #1
           SD     0x1000(R2), R3
           DADDI  R2, R2, #8
           DSUBI  R1, R1, #1
           BNEQ   R1, Loop
```

3. (15+5=20 points) Several researchers have suggested that adding a register-memory addressing mode to a load-store machine might be useful. The idea is to replace a sequences of

```
LOAD     R1, 0(Rb)
ADD      R2, R2, R1
```

by
ADD R2, 0(Rb)

Assume that the new instruction will cause the clock cycle to increase by 10%. The new instruction only affects the clock speed, but not the CPI.

(a) Consider the instruction frequency table shown in Figure B.13 of the textbook. What percentage of the loads must be eliminated for the machine with the new instruction to have at least the same performance.

(b) Show a situation in a multiple instruction sequence where a load of R1 followed immediately by the use of R1 (with some type of opcode) could not be replaced by a single instruction of the form proposed, assuming that the same opcode exists.

4. (10+10+20) = 40 points) Consider implementing stack-oriented operations **PUSH** and **POP** using MIPS instruction set. Assume register *R30* is used as a byte-addressable stack pointer (SP). An instruction **PUSH Rx** increments SP by 4 and writes the contents of register *Rx* to the memory word location pointed by SP. Similarly, the instruction **POP Rx** reads data from top of the stack to the register *Rx* and decrements SP by 4.

(a) Can you implement the above stack-oriented operations using a set of existing instructions from the MIPS instruction set? If yes, give the sequence of instructions to implement them. If not, explain why it can not be done.

(b) Consider adding the **POP Rx** instruction to the MIPS instruction set. Show the appropriate MIPS instruction format (Fig. B.22 on page B-35 of the textbook) and its fields to encode this instruction.

(c) Consider an ‘Unpipelined’ design of MIPS processor (as indicated in Slide #3 of Appendix A). Show the steps to execute this new instruction as a single instruction under the instruction format derived in part (b). How many clock cycles this instruction will take to execute. Will it be better than the solution derived in part (a)?