Assignment #2

1. Suppose we are considering a change to an instruction set. The base machine initially has only loads and stores to memory, and all operations work on the registers. Measurements of the load-store machine showing the instruction mix and clock cycle counts per instruction are given in the table below:

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Frequency</th>
<th>Clock Cycle Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU Operations</td>
<td>43%</td>
<td>1</td>
</tr>
<tr>
<td>Loads</td>
<td>21%</td>
<td>2</td>
</tr>
<tr>
<td>Stores</td>
<td>12%</td>
<td>2</td>
</tr>
<tr>
<td>Branches</td>
<td>24%</td>
<td>2</td>
</tr>
</tbody>
</table>

Assume that 25% of the ALU Operations directly use a loaded operand that is not used again.

We propose adding ALU instructions that have one source operand in memory. These new register-memory instructions have a clock cycle count of 2. Suppose that the extended instruction set increases the clock cycle count for branches by 1 (so branches now take 3 cycles total), but it does not affect the clock cycle time. Would this change improve CPU performance? Justify your answer by showing your work.

2. The purpose of this exercise is to compare the memory efficiency of four different styles of instruction set architectures. The architecture styles are:

1. Accumulator - All operations occur between a single register and a memory location.
2. Memory-memory - All three operands of each instruction are in memory.
3. Stack - All operations occur on the top of the stack. Only push and pop access memory, and all other instructions remove their operands from the top of stack and replace them with the result. The implementation uses a pair of hardware registers to hold the top two entries on the stack; accesses that use other stack positions require memory references.
4. Load-store - All operations occur in registers, and register-to-register instructions have three operands per instruction. There are 16 general purpose registers, making the register specifiers 4 bits long.

To measure memory efficiency, assume the following about all four instruction sets:

- The opcode is always one byte (8 bits).
- All memory addresses are 2 bytes (16 bits).
- All data operands are 4 bytes (32 bits).
- All instructions are an integral number of bytes in length.
There are no other optimizations to reduce memory traffic, and the variables A, B, C and D are initially in memory.

Invent your own assembly language mnemonics and write the best equivalent assembly language code for the high-level-language fragments given. Write the four code sequences for:

\[
\begin{align*}
A &= B + C; \\
B &= A + C; \\
D &= A - B; \\
\end{align*}
\]

Calculate the instruction bytes fetched and the 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 (code + data) required?

3. Consider the following code fragment:

\[
\text{loop: LD R1, 0(r2) }
\text{ADDI R1, R1, #1 }
\text{ST 0(R2), R1 }
\text{ADDI R2, R2, #4 }
\text{SUB R4, R3, R2 }
\text{BNEZ R4, loop }
\]

a) Show the timing of this instruction sequence for a processor that uses the 5-stage MIPS pipeline, but without any data forwarding and without a delayed branch slot. Determine the number of cycles required to execute this loop. Show your answer in a form similar to Figure A.1.

b) Show the timing of this instruction sequence with all possible forwarding and bypassing hardware, and with a single-cycle delayed branch slot. Determine the number of cycles required to execute this loop.