Joe Stanley

ECE 541 - HWK1

Problem 1:

For a certain important application involving matrices, analysis reveals that floating point instructions account for 40% of the total execution time, and the floating point transpose function alone is responsible for 50% of the total floating point time. As a result of this analysis, two mutually exclusive enhancements are being considered: 1) to add a FPTRANS instruction to the current floating point unit that will accelerate the transpose computation by a factor of 5, or 2) to speed up the entire floating point unit by a factor of two. Which proposal will provide the higher performance?

In [13]:
```
# Define Characteristics
fp_inst = 0.40
fp_trans = 0.50

# Evaluate Performance of Each
standard = fp_inst
impr_1 = fp_inst + (fp_trans/5 - fp_trans) * fp_inst
impr_2 = fp_inst / 2

latex1 = (r'$$ \frac{40\%+(\frac{50\%}{5}-50\%)*40\%}{40\%} =  \frac{'+
          str(impr_1)+r'}{'+str(standard)+r'} = '+str(impr_1/standard)+r'= '

latex2 = (r'$$ \frac{\frac{40\%}{2}}{40\%} =  \frac{'+
          str(impr_2)+r'}{'+str(standard)+r'} = '+str(impr_2/standard)+r'= '
```

SOLUTION:

1. Performance Improvement of Type 1: {{latex1}}
2. Performance Improvement of Type 2: {{latex2}}

Problem 2:

Your task 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:

- A = B + C;
- B = A + C;
- D = A - B;

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?

SOLUTION:

1)

A = B + C	Accumulator	Mem-Mem	Stack	Load-Store
	REG < LOAD B	MEM (A) < STORE B + C	REG < POP A	REG (B) < LOAD B
	REG < ADD C		REG < POP B	REG (C) < LOAD C
	MEM < STORE A		REG < POP C	REG (A) < ADD B,C
			REG < ADD B,C	MEM (A) < STORE A
			MEM < PUSH A	

2)

B = A + C	Accumulator	Mem-Mem	Stack	Load-Store
	REG < LOAD A	MEM (B) < STORE A + C	REG < POP A	REG (A) < LOAD A
	REG < ADD C		REG < POP B	REG (C) < LOAD C
	MEM < STORE B		REG < POP C	REG (B) < ADD A,C
			REG < ADD A,C	MEM (B) < STORE B
			MEM < PUSH A	

B = A + C	Accumulator	Mem-Mem	Stack	Load-Store
			MEM < PUSH B	

3)

D = A - B	Accumulator	Mem-Mem	Stack	Load-Store
	REG < LOAD B	MEM (D) < STORE A - B	REG < POP A	REG (A) < LOAD A
	REG < NEG B		REG < POP B	REG (B) < LOAD B
	REG < ADD A		REG < NEG B	REG (B) < NEG B
	MEM < STORE A		REG < ADD A,B	REG (D) < ADD A,B
			MEM < PUSH A	MEM < STORE D
			MEM < PUSH B	
			MEM < PUSH C	
			MEM < PUSH D	

According to code size, Memory-Memory Architecture is the most efficient method. However, according to total bandwidth, Accumulator is perhaps the most efficient (by inspection).