1. There are 16 General-Purpose registers in ARM vs RISC-V has 32 General Purpose registers
    a. RISC-V Advantages
        i. Programmer:
            1. Application: More registers mean more opportunities for assignments and more flexibility with what you can store outside of the main memory for a program.
            2. Compiler developer: More registers mean you have more opportunities to save time by not having to move things around with loads and stores as much from main memory. This is good.
        ii. CPU Writer: More registers make the architecture more flexible. Additionally, having more registers available makes it easier for the CPU to achieve its fastest possible speed due to less memory access. The compiler writer has more opportunities which is beneficial both for problem solving and efficiency.
    b. ARM Disadvantage if register size was increased to match RISC-V
        i. Programmer:
            1. Application: The downside for the programmer is that if the CPU writer and compiler writer have not adapted the architecture to the new register sizes, you are going to have a hard time knowing what values you can assign into what registers.
            2. Compiler developer: Having registers in the ARM architecture that are larger would be a waste of space and time. You are loading in larger blocks of memory or not getting good use of the extra size.
        ii. CPU Writer: Arm's registers are normally word-aligned and changing the size of the registers would create a mismatch in the alignment.
2. Compare memory efficiency of Accumulator, memory-memory, stack, load-store
    a. For each architecture, write the best assembly lenguage code for this high level language code sequence.
        i. Accumulator
            1. A=B+C
                a. Load B
                b. ADD C
                c. Store A
            2. B=A+C
                a. Load A
                b. ADD C
                c. Store B
            3. D=A-B
                a. Load A
                b. SUB B
                c. Store D
        ii. Memory-memory
            1. A=B+C
                a. ADD A, B, C
            2. B=A+C

            a. ADD B, A, C
        3. D=A-B
            a. SUB D, A, B
    iii. Stack
        1. A=B+C
            a. Push B
            b. Push C
            c. ADD
            d. Pop A
        2. B=A+C
            a. Push A
            b. Push C
            c. ADD
            d. Pop B
        3. D=A-B
            a. Push A
            b. Push C
            c. SUB
            d. Pop D
    iv. Load store
        1. A=B+C
            a. Load R1, B
            b. Load R2, C
            c. ADD R3, R1, R2
            d. Store R3, A
        2. B=A+C
            a. Load R1, R3
            b. ADD R3, R1, R2
            c. Store R3, B
        3. D=A-B
            a. Load R2, R3
            b. SUB R3, R1, R2
            c. Store R3, D

c. Assume a 16 bit memory address and data operands. 16 general-purpose registers.

Will assume each op-code is 1 byte, assume register name is 4 bits

1. Accumulator – 45 Bytes
    i. How many instruction bytes are fetched?
        1. 1+2+1+2+1+2, 1+2+1+2+1+2, 1+2+1+2+1+2 = 9*3 = 27 bytes
    ii. How many bytes of data are transferred to and from memory?
        1. (2bytes*3 memory calls)*3 statements = 18 bytes
1. Memory-Memory – 39 Bytes
    i. How many instruction bytes are fetched?
        1. 1+2+2+2, 1+2+2+2, 1+2+2+2 = 21

    ii. How many bytes of data are transferred to and from memory?
        1. (2bytes*3 memory calls)*3 statements = 18 bytes

1. Stack – 48 Bytes
    i. How many instruction bytes are fetched?
        1. 1+2+1+2+1+2+1, 1+2+1+2+1+2+1, 1+2+1+2+1+2+1 = 10*3 = 30Bytes
    ii. How many bytes of data are transferred to and from memory?
        1. (2bytes*3 memory calls)*3 statements = 18 bytes

1. Load-Store - 39Bytes
    i. How many instruction bytes are fetched?
        1. 3bytes for register name +1+2+1+2+1+1+2, 3+1+1+1+2, 3+1+1+1+2 = 29 Bytes
    ii. How many bytes of data are transferred to and from memory?
        1. 2+2+2+2+2 = 10 bytes

1. Most efficient architecture:
    i. Load-Store and Memory-memory have the smallest total byte usages, but Load-Store has a much smaller transfer of data to and from memory which make it much more efficient.