Question 2

Code Sequences

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

1. ```
   // zero-address machine
   PUSH A;
   PUSH C;
   ADD;
   POP A;
   PUSH B;
   PUSH C;
   SUB;
   POP B;
   PUSH B;
   PUSH A;
   SUB;
   POP D;
   PUSH D;
   PUSH A;
   ADD;
   POP C;
   ```

2. ```
   // one-address machine
   LOAD A;
   ADD C;
   STORE A;
   LOAD B;
   SUB C;
   STORE B;
   SUB A; // Assume that STORE B leaves the value in the accumulator
   STORE D;
   ADD A; // Assume that STORE D leaves the value in the accumulator
   STORE C;
   ```

3. ```
   // two-address machine
   ADD A, C;
   SUB B, C;
   SUB D, D; // Extra operation to clear D
   ```
   ```

```
ADD D, B;
SUB D, A;
SUB C, C; // Extra operation to clear C
ADD C, A;
ADD C, D;
```

4.
```
// three-address machine on memory
ADD A, A, C;
SUB B, B, C;
SUB D, B, A;
ADD C, D, A;
```

5.
```
// three-address machine on registers
LOAD R1, A;
LOAD R2, C;
ADD R1, R1, R2; // A = A + C
LOAD R3, B;
SUB R3, R3, R2; // B = B - C
SUB R4, R3, R1; // D = B - A, note that there was no need to load D
ADD R2, R4, R1; // C = D + A
STORE R1, A;
STORE R2, C;
STORE R3, B;
STORE R4, D;
```

Instruction bytes fetched and memory bytes transferred

Note that we have variable length instructions, so include only what we need.

1. Zero-address machine
   - We have 4 opcode only instructions and 12 opcode + memory address instructions. Instructions bytes fetched = 4 * 1 + 12 * 5 = 64 bytes.
   - From above, a total of 12 instructions involve memory access (read/write) and so 12 * 4 = 48 bytes transferred.
   - Total: 64 + 48 = 112 bytes.
2. One-address machine
   - All 10 instructions are opcode + memory address instructions. Instructions bytes fetched = 10 * 5 = 50 bytes.
   - A total of 10 instructions involve memory access (read/write) and so 10 * 4 = 40 bytes transferred.
   - Total: 50 + 40 = 90 bytes.
3. Two-address machine
   - All 8 instructions are opcode + two memory addresses instructions. Instructions bytes fetched = 8 * 9 = 72 bytes.

    - A total of 8 instructions involve memory access (read/write), each accessing two memory locations, so 8 * 2 * 4 = 64 bytes transferred.
    - Total: 72 + 64 = 136 bytes.
4. Three-address machine on memory
    - All 4 instructions are opcode + three memory addresses instructions. Instructions bytes fetched = 4 * 13 = 52 bytes.
    - A total of 4 instructions involve memory access, each accessing three memory locations, so 4 * 3 * 4 = 48 bytes transferred.
    - Total: 52 + 48 = 100 bytes.
5. Three-address machine on registers
    - 3 `LOAD` and 4 `STORE` instructions are opcode + register + memory address instructions. Remaining 4 arithmetic instructions are opcode + three registers instructions. Instructions bytes fetched = 7 * 6 + 4 * 4 = 58 bytes.
    - A total of 7 instructions involve memory access (load/store), so 7 * 4 = 28 bytes transferred.
    - Total: 58 + 28 = 86 bytes.

Most Efficient Code Size

The three-address machine on memory is the most efficient in terms of code size, with only 4 instructions.

Most Efficient Memory Transfer (Instructions Bytes Fetched and Memory Bytes Transferred)

The three-address machine on registers is the most efficient in terms of memory transfer, with a total of 86 bytes for both instruction and memory-data bytes.