Problems 13.6:

1. Zero-Address Machine (Stack-based)

In a zero-address machine, operations use a stack. Operands are pushed onto the stack, and operations are performed on the top elements.

Program:

```
PUSH B
PUSH C
MUL
PUSH A
ADD
PUSH E
PUSH F
MUL
PUSH D
SUB
DIV
POP X
```

Explanation:

- **Numerator (A + B × C):**
  - Push B and C, multiply to get B × C, then push A and add to get A + B × C.
- **Denominator (D - E × F):**
  - Push E and F, multiply to get E × F, then push D and subtract to get D - E × F.
- **Final Step:**
  - Divide the second-top (A + B × C) by the top (D - E × F), then pop the result into X.

Instruction Count: 12

---

2. One-Address Machine (Accumulator-based)

In a one-address machine, operations use an accumulator register and memory operands.

Program:

```
LOAD E
MUL F
STORE TEMP1
LOAD D
SUB TEMP1
STORE TEMP2
LOAD B
MUL C
ADD A
DIV TEMP2
```

```
STORE X
```

Explanation:

- **Denominator (D - E × F):**
  - Load E, multiply by F, store in TEMP1.
  - Load D, subtract TEMP1, store in TEMP2.
- **Numerator (A + B × C):**
  - Load B, multiply by C, add A.
- **Final Step:**
  - Divide the accumulator by TEMP2, store the result in X.

Instruction Count: 11

---

3. Two-Address Machine

In a two-address machine, instructions specify two operands, with the first serving as both source and destination.

Program:

```
MOVE X B
MUL X C
ADD X A
MOVE TEMP1 E
MUL TEMP1 F
MOVE TEMP2 D
SUB TEMP2 TEMP1
DIV X TEMP2
```

Explanation:

- **Numerator (A + B × C):**
  - Move B to X, multiply by C, add A.
- **Denominator (D - E × F):**
  - Move E to TEMP1, multiply by F.
  - Move D to TEMP2, subtract TEMP1.
- **Final Step:**
  - Divide X by TEMP2.

Instruction Count: 8

---

4. Three-Address Machine

In a three-address machine, instructions specify two sources and one destination.

Program:

```
MUL TEMP1 B C
ADD TEMP2 A TEMP1
MUL TEMP3 E F
SUB TEMP4 D TEMP3
DIV X TEMP2 TEMP4
```

Explanation:

- Compute B × C into TEMP1.
- Add A and TEMP1 into TEMP2.
- Compute E × F into TEMP3.
- Subtract TEMP3 from D into TEMP4.
- Divide TEMP2 by TEMP4 into X.

Instruction Count: 5

Comparison of Addressing Modes

Addressing Mode	Instruction Count	Pros	Cons
0 Address	12	No registers needed; uses stack.	More instructions; stack access may be slow.
1 Address	11	Simple hardware with accumulator.	Frequent memory loads/stores.
2 Address	8	Fewer instructions; direct memory ops.	Needs temporaries; slightly complex hardware.
3 Address	5	Fewest instructions; clear operand use.	Requires complex hardware for three operands.

- **Efficiency:** The three-address machine uses the fewest instructions (5), while the zero-address machine uses the most (12).
- **Hardware:** Zero-address machines are simpler (stack-based), while three-address machines require more complex instruction decoding.
- **Memory:** The stack in zero-address mode avoids explicit temporaries, but other modes may need them.

This solution provides a clear understanding of how different addressing modes handle the same computation, highlighting their unique characteristics and trade-offs.

Problems 14.1:

1. LOAD IMMEDIATE 20

- **Value Loaded:** 20
- **Explanation:** In immediate addressing, the operand itself is the data. Thus, the value 20 is loaded directly into the accumulator.

2. LOAD DIRECT 20

- **Value Loaded:** 40
- **Explanation:** Direct addressing uses the operand as the memory address. The content of memory address 20 is 40, so 40 is loaded into the accumulator.

3. LOAD INDIRECT 20

- **Value Loaded:** 60
- **Explanation:** Indirect addressing uses the operand as an address that contains another address. Memory address 20 contains 40, and memory address 40 contains 60. Therefore, 60 is loaded into the accumulator.

4. LOAD IMMEDIATE 30

- **Value Loaded:** 30
- **Explanation:** Immediate addressing loads the operand value directly. Here, the value 30 is loaded into the accumulator.

5. LOAD DIRECT 30

- **Value Loaded:** 50
- **Explanation:** Direct addressing loads the content of the specified memory address. The content of address 30 is 50, so 50 is loaded into the accumulator.

6. LOAD INDIRECT 30

- **Value Loaded:** 70
- **Explanation:** Indirect addressing loads the content of the address stored at the operand. Memory address 30 contains 50, and memory address 50 contains 70. Thus, 70 is loaded into the accumulator.

Summary of Addressing Modes:

- **Immediate Addressing:** The operand is the data itself (e.g., 20 or 30).

- **Direct Addressing:** The operand is the address of the data (e.g., contents of address 20 or 30).
- **Indirect Addressing:** The operand is the address of an address where the data resides (e.g., contents of the address stored at 20 or 30).

Problems 14.17:

Type 1: Two 15-bit Addresses and One 3-bit Register

- **Opcode:** 000 (3 bits)
- **First Address:** Bits 3–17 (15 bits)
- **Second Address:** Bits 18–32 (15 bits)
- **Register Number:** Bits 33–35 (3 bits)
- **Total Bits:** 3 + 15 + 15 + 3 = 36 bits

Type 2: One 15-bit Address and One 3-bit Register

- **Opcode:** 001 (3 bits)
- **Address:** Bits 3–17 (15 bits)
- **Register Number:** Bits 18–20 (3 bits)
- **Unused:** Bits 21–35 (15 bits)
- **Total Bits:** 3 + 15 + 3 + 15 = 36 bits

Type 3: No Addresses or Registers

- **Opcode:** 010 (3 bits)
- **Remaining Bits:** Bits 3–35 (33 bits, optional data or unused)
- **Total Bits:** 3 + 33 = 36 bits

Summary

- **Opcode** 000**:** Indicates an instruction with two 15-bit addresses and one 3-bit register.
- **Opcode** 001**:** Indicates an instruction with one 15-bit address and one 3-bit register.
- **Opcode** 010**:** Indicates an instruction with no addresses or registers.