Q1)

```
#include <stdio.h>
int main() {
    int number, square;
    printf("Enter a number: ");
    scanf("%d",  &number);
    square = number * number;
    printf("The square of %d is %d.\n", number, square);
    return 0;
}
```

ISA AND Compiler	No Of Instructions
RISC-V 32-bits gcc 14.2.0	28
RISC-V 64-bits gcc (trunk)	29
RISC-V rv32gc gcc (trunk)	Not Available
RISC-V rv32gc clang(trunk)	27
RISC-V rv64gc clang(trunk)	27

x86-64 clang 12.0.0	23
X86-64 gcc 14.2.0	24
ARM64 gcc 14.2.0	23
ARM32 gcc 14.2.0	23
Armv8-a-clang 19.1.0	26
MIPS64 gcc 5.4	44

Q2)

```
#include<stdio.h>
int main(){
    int num, fact=1;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
    for (int i = 1; i <=num; i++)
    {
        fact = fact*i
}
      printf("The factorial of %d is %d",num,fact);
    return 0;
}
```

ISA AND Compiler	No Of Instructions
RISC-V 32-bits gcc 14.2.0	39
RISC-V 64-bits gcc (trunk)	42
RISC-V rv32gc gcc (trunk)	Not Available
RISC-V rv32gc clang(trunk)	42
RISC-V rv64gc clang(trunk)	42

x86-64 clang 12.0.0	32
X86-64 gcc 14.2.0	30
ARM64 gcc 14.2.0	35
ARM32 gcc 14.2.0	35
Armv8-a-clang 19.1.0	40
MIPS64 gcc 5.4	59

Q3)

ISA AND Compiler	**No Of Instructions**
RISC-V 32-bits gcc 14.2.0	70
RISC-V 64-bits gcc (trunk)	73
RISC-V rv32gc gcc (trunk)	Not Available
RISC-V rv32gc clang(trunk)	75
RISC-V rv64gc clang(trunk)	75
x86-64 clang 12.0.0	48
X86-64 gcc 14.2.0	49
ARM64 gcc 14.2.0	54
ARM32 gcc 14.2.0	69
Armv8-a-clang 19.1.0	57
MIPS64 gcc 5.4	82

Q4)

ISA AND Compiler	
RISC-V 32-bits gcc 14.2.0	30
RISC-V 64-bits gcc (trunk)	32
RISC-V rv32gc gcc (trunk)	Not Available
RISC-V rv32gc clang(trunk)	35
RISC-V rv64gc clang(trunk)	35
x86-64 clang 12.0.0	23
X86-64 gcc 14.2.0	21
ARM64 gcc 14.2.0	26
ARM32 gcc 14.2.0	33
Armv8-a-clang 19.1.0	29
MIPS64 gcc 5.4	35

Q5)

ISA AND Compiler	No Of Instructions
RISC-V 32-bits gcc 14.2.0	55
RISC-V 64-bits gcc (trunk)	53
RISC-V rv32gc gcc (trunk)	Not Available
RISC-V rv32gc clang(trunk)	56
RISC-V rv64gc clang(trunk)	56
x86-64 clang 12.0.0	39
X86-64 gcc 14.2.0	47
ARM64 gcc 14.2.0	46
ARM32 gcc 14.2.0	55
Armv8-a-clang 19.1.0	44
MIPS64 gcc 5.4	74

SUMMARY:

A)

For the same high-level program, the instruction sequence for different compilers and different machine architectures (ISAs) - Observations:

1. **RISC-V:**
    - The number of instructions varies significantly based on bit architecture (32-bit or 64-bit) and compiler versions (e.g., GCC trunk or Clang trunk).
    - Clang trunk typically generates fewer instructions compared to GCC trunk for the same ISA and bit architecture.
2. **x86-64:**
    - Clang consistently generates fewer instructions compared to GCC.

- Both compilers produce highly optimized instruction sequences relative to other architectures.

3. ARM (64-bit and 32-bit):

- ARM64 generates fewer instructions compared to ARM32, consistent with optimizations in modern ARM64 architecture.
- Instruction sequences differ across GCC and Clang, with slight variation in instruction counts.

4 . MIPS64:

- MIPS64 generates the most instructions among the ISAs, reflecting a less compact ISA design compared to others.

B)

For the same ISA but for different compilers - Observations:

1. **RISC-V:**
   - Clang generally produces fewer instructions than GCC for both 32-bit and 64-bit variants.
   - The instruction count difference between GCC and Clang is minor for some programs (e.g., factorial program).
2. **x86-64:**
   - Clang 12.0 consistently generates fewer instructions than GCC 14.2 for all tested programs, indicating Clang's efficient optimization strategies for this architecture.
3. **ARM:**
   - Both GCC and Clang produce comparable results for ARM64. For ARM32, Clang (Armv8-a) shows a trend of higher instruction counts than GCC.

C)

Observations about instruction sequences for 32-bit and 64-bit machines of the same ISA and compiler:

1. **RISC-V:**
   - 64-bit architecture generates slightly more instructions than its 32-bit counterpart for the same compiler and program. This could be attributed to differences in data representation and pointer size.
2. **x86-64:**

    - The results show minimal differences between 32-bit and 64-bit instruction counts, reflecting the optimizations in the x86-64 ISA to maintain instruction density across architectures.
3. **ARM:**
    - ARM64 instruction sequences are shorter compared to ARM32, leveraging the efficiency and enhancements in 64-bit architecture.

Key Takeaways:

1. **ISA Design Matters:** Compact ISAs like x86-64 and ARM64 tend to generate fewer instructions compared to MIPS64 and RISC-V.
2. **Compiler Optimizations:** Clang is generally more efficient in generating fewer instructions across multiple ISAs and architectures.
3. **32-bit vs. 64-bit:** 64-bit architectures tend to have marginally higher instruction counts due to larger data types but compensate with performance benefits and advanced optimizations.