Computer Architecture Lab1

PRIYANSHU RAO

January 2025

List of Tables

Contents

Note: ARM32 gcc 14.2.0 was unavailable so I used ARM gcc 14.2.0

Question 1 Table

Architecture	**Version**
RISC-V 32-bits gcc 14.2.0	27
RISC-V 64-bits gcc (trunk)	28
RISC-V rv32gc gcc (trunk)	Not available
RISC-V rv32gc clang (trunk)	24
RISC-V rv64gc clang (trunk)	24
x86-64 clang 12.0.0	21
x86-64 gcc 14.2	22
ARM64 gcc 14.2.0	22
ARM gcc 14.2.0	24
Armv8-a clang 19.1.0	24
MIPS64 gcc 5.4	42

Table 1: Computer Architecture Specifications of question 1

Question 2 Table

Architecture	Version
RISC-V 32-bits gcc 14.2.0	36
RISC-V 64-bits gcc (trunk)	39
RISC-V rv32gc gcc (trunk)	Not available
RISC-V rv32gc clang (trunk)	40
RISC-V rv64gc clang (trunk)	40
x86-64 clang 12.0.0	31
x86-64 gcc 14.2	29
ARM64 gcc 14.2.0	32
ARM gcc 14.2.0	35
Armv8-a clang 19.1.0	37
MIPS64 gcc 5.4	54

Table 2: Computer Architecture Specifications of question 2

Question 3 Table

Architecture	Version
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	49
ARM64 gcc 14.2.0	54
ARM gcc 14.2.0	69
Armv8-a clang 19.1.0	57
MIPS64 gcc 5.4	82

Table 3: Computer Architecture Specifications of question 3

Question 4 Table

Architecture	Version
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	21
ARM64 gcc 14.2.0	26
ARM gcc 14.2.0	33
Armv8-a clang 19.1.0	29
MIPS64 gcc 5.4	35

Table 4: Computer Architecture Specifications of question 4

Question 5 Table

Architecture	Version
RISC-V 32-bits gcc 14.2.0	71
RISC-V 64-bits gcc (trunk)	79
RISC-V rv32gc gcc (trunk)	Not available
RISC-V rv32gc clang (trunk)	74
RISC-V rv64gc clang (trunk)	74
x86-64 clang 12.0.0	51
x86-64 gcc 14.2	56
ARM64 gcc 14.2.0	57
ARM gcc 14.2.0	66
Armv8-a clang 19.1.0	60
MIPS64 gcc 5.4	97

Table 5: Computer Architecture Specifications of question 5

1 Q1. For the same high-level program, the instruction sequence for different compilers and different machine architecture (represented by its ISA) - What are your observations ?

ISA's categorized by label positioning:

1. **Labels before the code:**
   - RISC-V 32-bits gcc 14.2.0
   - RISC-V 64-bits gcc (trunk)
   - x86-64 gcc 14.2
   - ARM64 gcc 14.2.0
   - ARM gcc 14.2.0
2. **Labels between the lines:**
   - RISC-V rv64gc clang (trunk)
   - MIPS64 gcc 5.4
3. **Labels at the end:**
   - RISC-V rv32gc clang (trunk)
   - RISC-V rv64gc clang (trunk)
   - x86-64 clang 12.0.0

Insights:

- Different compiler-ISA combinations result in variations in the number of instruction lines.
- RISC-V `rv32gc clang (trunk)` and `RISC-V rv64gc clang (trunk)` mostly have the same number of instructions.
- MIPS consistently shows the highest number of instruction lines compared to RISC-V, ARM, and x86 architectures, while x86 shows the least.

2 Q2. Even for the same ISA, but for different compilers, can you comment about the generated the instruction sequences

Observations:

- Same ISA with different compilers shows negligible differences in the number of instruction lines.

- RISC-V `rv32gc clang (trunk)` and `RISC-V rv64gc clang (trunk)` mostly have the same number of instructions.
- x86-64 `gcc 14.2` has more instruction lines than x86-64 `clang 12.0.0` in 3 out of 5 cases in the given data.
- RISC-V 64-bits `gcc (trunk)` has more instructions than RISC-V 32-bits `gcc 14.2.0`.

3 Q3. Provide your observations about the instruction sequences generated for 32-bit and 64-bit machines of the same ISA and of the same compiler.

Comparison:

- RISC-V 64-bits `gcc (trunk)` consistently has more instructions than RISC-V 32-bits `gcc 14.2.0`.
- RISC-V `rv32gc clang (trunk)` and `rv64gc clang (trunk)` mostly produce the same number of instructions.