CS 311 - Computer Architecture
Lab 1
Part A - Comparing ISAs

Raghuveer Verma

CS23BT041

15 January, 2025

Important Note

- **RISC-V rv32gc gcc (trunk)** is not available on godbolt
- **ARM32 gcc 14.2.0** is is not available on godbolt and is replaced by **ARM gcc 14.2.0** in this report

Question 1

C program to take an integer value as an input and print out the square of the value along with the original number.

Code used:

```
#include <stdio.h>

int main(){
    int a;
    scanf("%d", &a);
    printf("Value = %d, Square = %d", a, a*a);
    return 0;
}
```

ISA and Compiler	No. of Instructions
RISC-V 32-bits gcc 14.2.0	24
RISC-V 64-bits gcc (trunk)	25
RISC-V rv32gc clang (trunk)	21
RISC-V rv64gc clang (trunk)	21
x86-64 clang 12.0.0	18
x86-64 gcc 14.2.0	19
ARM64 gcc 14.2.0	19
ARM gcc 14.2.0	21
Armv8-a-clang 19.1.0	21
MIPS64 gcc 5.4	36

Question 2

C program to take a positive integer as an input and iteratively compute the factorial of the integer.

Code used:

```
#include <stdio.h>

int main(){
    int a, fact = 1;
    scanf("%d",&a);
    if(a <= 0){
        printf("Please enter a positive integer");
        return 0;
    }
    else{
        for(int i = 1; i <= a; i++){
            fact = fact*i;
        }
    }
    printf("Factorial of %d = %d", a, fact);
    return 0;
}
```

ISA and Compiler	No. of Instructions
RISC-V 32-bits gcc 14.2.0	43
RISC-V 64-bits gcc (trunk)	46
RISC-V rv32gc clang (trunk)	53
RISC-V rv64gc clang (trunk)	53
x86-64 clang 12.0.0	38
x86-64 gcc 14.2.0	35
ARM64 gcc 14.2.0	40
ARM gcc 14.2.0	43
Armv8-a-clang 19.1.0	48
MIPS64 gcc 5.4	64

Question 3

Code used:

```
int x[10];
void init(int *x){
    for(int i = 0; i < 10; i++)
        x[i] = i;
}

int sumofarray(int *x){
    int sum = 0;
    for(int i = 0; i < 10; i++)
        sum = sum + x[i];

    return sum;
}

int main(){
    init(x);
    int sum = sumofarray(x);
    return sum;
}
```

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 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
ARM gcc 14.2.0	69
Armv8-a-clang 19.1.0	57
MIPS64 gcc 5.4	82

Question 4

Code used:

```
int N = 8;

int main(){
    int t1 = 0, t2 = 1, nextTerm;

    for(int i = 0; i < N; i++){
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;
    }

    return nextTerm;
}
```

ISA and Compiler	**No. of Instructions**
RISC-V 32-bits gcc 14.2.0	30
RISC-V 64-bits gcc (trunk)	32
RISC-V rv32gc clang (trunk)	34
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
ARM gcc 14.2.0	33
Armv8-a-clang 19.1.0	29
MIPS64 gcc 5.4	35

Question 5

Code used:

```
int source[8] = {0, 1, 2, 3, 4, 5, 6, 7};
int dest[8];
int calculate(int source) { return source*source; }

void loop(int *source, int *dest, int N){
    for(int k = 0; k < N; k++){
        int a = 10;

        if(source[k] != 0)
            dest[k] = calculate(source[k]) + 10;
    }
}

int main(){
    loop(source, dest, 8);

    return 0;
}
```

ISA and Compiler	**No. of Instructions**
RISC-V 32-bits gcc 14.2.0	71
RISC-V 64-bits gcc (trunk)	79
RISC-V rv32gc clang (trunk)	74
RISC-V rv64gc clang (trunk)	74
x86-64 clang 12.0.0	51
x86-64 gcc 14.2.0	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

Results

Comparing different ISAs and different compilers for same program

- **MIPS64 gcc** had the most number of instructions for all the programs.
- **RISC-V rv32gc clang** and **RISC-V rv64gc clang** had the same number of instructions for every program.
- **x86-64 gcc** and **x86-64 clang** needed relatively less number of instructions.

Comparing different compilers for the same ISAs

- The compiler **clang** puts the initializations at the end of the assembly code, while **gcc** puts the initializations at the beginning of the assembly code.

Comparing instructions generated for 32-bit and 64-bit machines of the same ISA and the same compiler

RISC-V 32-bits gcc	**RISC-V 64-bits gcc**
Makes use of instructions such as sw, lw etc. for operations on 32-bit data	Makes use of instructions such as sd, ld etc. for operations on 64-bit data
Registers contain 32-bit data	There exists an instruction signed extend word (sext.w) to extend 32-bit word to 64 bits

RISC-V rv32gc clang	**RISC-V rv64gc clang**
Makes use of instructions such as sw, lw etc. for operations on 32-bit data	Makes use of instructions such as sd, ld etc. for operations on 64-bit data

ARM gcc	**ARM64 gcc**
Uses registers r0, r1, r2, ...	Uses registers x0, x1, x2, ...
Uses push and pop commands for stack management	Uses stp and ldp for stack management
Uses .ascii for strings	Uses .string for strings