RISC-V Audiomark - Coding Challenge

Implimented by Shravan A Y <shravanay205@gmail.com>

This repo contains implementation the following function in C:

`void q15_axpy_rvv(const int16_t *a, const int16_t *b, int16_t *y, int n, int16_t alpha);`

that computes,

```
for all i in [0..n):
    y[i]=sat_q15(a[i]+α*b[i])y[i]
```

Implementation Overview

Scalar Reference:

The scalar implementation performs:

1. Q15 × Q15 multiply
2. Accumulation with a Q15 addend
3. Saturation to the int16_t range

This version is used as a reference for verification.

The RVV implementation uses:

1. Vector agnostic loops via `vsetvl`
2. 16-bit vector load (`vle16`)
3. Widen multiply (`vwmul`)
4. Widen accumulation in 32-bit lanes
5. Saturation and narrowing back to Q15
6. Vector store (`vse16`)

Design Choices

1. Widening is done to prevent overflow during multiplication and accumilation
2. Explicit saturation is used to ensure bit-for-bit equivalence with scalar reference
3. The implementation is vector length agnostic using `vsetvl`

4. No assumptions are made about VLEN or microarchitecture

Building:

`make`

Running:

`qemu-riscv32 axpy-rvv-q15.elf`

Results

```
$ qemu-riscv32 axpy-rvv-q15.elf
Cycles ref: 443980
Verify RVV: OK (max diff = 0)
Cycles RVV: 210920
```

Since there is randomness, i took the average

```
$ python avg-res.py
Average scalar cycles : 363844.00
Average RVV cycles : 172988.40
Speedup : 2.10x
```

Implimentation github link

https://github.com/ShravanAYG/q15_saturate/tree/main