CS2610
Lab 2 : Multi-Precision Arithmetic

Various algorithms utilize multi-byte integers and perform arithmetic operations on such integers. For example, cryptographic algorithms like RSA perform computations on 1024 bits integers. Register sizes in modem processors however vary from 8-64 bits. For example, you are using a 64-bit RISCV toolchain and can only store 64-bit values in a register. This necessitates use of specialized techniques capable of handling multi-byte or multi-precision arithmetic.
The following instructions may be helpful for this assignment:

1. `mul rd, rs1, rs2` performs an XLEN-bit$\times$XLEN-bit multiplication of rs1 by rs2 and places the lower XLEN bits in the destination register. (Here, XLEN = 64)
2. `mulhu rd, rs1, rs2` performs the same multiplication but places the upper XLEN bits of the product in the destination register.
3. `sltu rd, rs1, rs2` compares two unsigned integers and sets the destination register to 1 if the first operand is less than the second, otherwise 0. This can be used to detect overflow if the result of addition is less than one of the operands.

Note: Data in memory should be stored in little-endian format.

Problem 1: 64-bit × 64-bit multiplication (40 points)

Write an assembly program that reads two 64-bit numbers from memory, performs a multiplication operation on these numbers and stores the 128-bit result in memory.

Test your code

val1 = 0x1234567887654321
val2 = 0x8765432112345678
result = 0x9a0cd05b99fe92eff39ef8670b88d78

Problem 2: 128-bit x 128-bit multiplication (60 points)

Write an assembly program that reads two 128-bit numbers from memory, performs a multiplication operation on these numbers and stores the 256-bit result in memory. Since each register can only store 64-bit values, you may use two registers to store each operand and perform four partial multiplications to get the final result.

Test your code

val1 = 0x12345678876543211234567887654321
val2 = 0x87654321123456788765432112345678
result = 0x09a0cd05b99fe92f127b8991e3f85fd60814ac129b11041eff39ef8670b88d78

What you need to submit:

1. Code files
2. Screenshots of the output

Note:

1. All the files should be submitted in a zipped folder through Moodle.
2. The zipped folder should be named $< Roll_No >$_Lab2.zip.

Resources

`file:///home/saltanat/Downloads/riscv-unprivileged.pdf`