232 Project Journal:

M1:

➔ Translated Algorithm into RISC-V to get an impression of what instructions are required:

Euclid's algorithm in RISC-V assembly:

```
relPrime:   addi sp, sp, -16
            sw ra, 0(sp)
            sw s0, 4(sp)
            sw s1, 8(sp)
            addi s0, x0, 2
            addi s1, x0, 1
            sw a0, 12(sp)
loop:       addi a1, s0, 0
            lw a0, 12(sp)
            jal ra, gcd
            beq a0, s1, done
            addi s0, s0, 1
            jal x0, loop
done:       addi a0, s0, 0
            lw ra, 0(sp)
            lw s0, 4(sp)
            lw s1, 8(sp)
            addi sp, sp, 16
            jalr x0, ra
```

```
gcd:        addi sp, sp, -4
            sw ra, 0(sp)
            beq a0, x0, returnb
loop:       beq a1, x0, returna
            blt a1, a0, agreater
            sub a1, a1, a0
            jal x0, loop
agreater:   sub a0, a0, a1
            jal x0, loop
returnb:    addi a0, a1, 0
returna:    lw ra, 0(sp)
            addi sp, sp, 4
            jalr x0, ra
```

Required Instructions:

```
addi rd, rs, imm
sub rd, rs1, rs2
sw rd, imm (rs)
lw rd, imm (rs)
beq rs1, rs2, imm
blt rs1, rs2, imm
jal rd, imm
jalr rd, imm
```

➔ Few optimization strategies for further thought:

Optimization:

Do not assign zero register but rather use value 1 in `x0`. Enables imm values for stack frame building by using shift instructions. Also enables beq without first assigning value to register. Introducing, bez (branch equal zero) instruction.