relPrime – 0x0000

GCD – 0x0014

whileLoop – 0x0002

whileLoopGCD -- 0x0029

else – 0x0033

break -- 0x0039

aEqualsZero – 0x003a

0x0000 – 0000 0101 000 100 xx

0x0001 – 1000 000000000010

0x0002 – 0110 010 000000000

0x0003 – 0000 0110 000 110 xx

0x0004 – 0000 0101 000 100 xx

0x0005 – 0110 010 0000000001

0x0006 – 0000 0101 000 001 xx

0x0007 – 0110 010 000000010

0x0008 – 0000 0101 010 000 xx

0x0009 – 0001 010 000000011

0x000a – 0000 0101 000 010

0x000b – 0000 0101 000 010 xx

0x000c – 0000 0101 000 001 xx

0x000d – 0000 0101 000 001 xx

0x000e – 0110 010 000000000

0x000f – 0000 0101 000 010 xx

0x0010 – 0001 000000000001

0x0011 – 0000 0101 000 010 xx

0x0012 – 0000 0101 000 001 xx

0x0013 – 0100 000000000001

0x0014 – 0000 0110 000 100 xx

0x0015 – 0101 010 111111110

0x0016 – 0000 0110 000 110 xx

0x0017 – 0101 010 111111111

0x0018 – 0000 0110 000 101 xx

0x0019 – 0101 010 000000000

0x001a – 0000 0110 000 001 xx

0x001b – 0000 0110 010 001 xx

0x001c – 0001 111111111100

0x001d – 0000 0110 000 010 xx

0x001e – 0000 0110 100 000 xx

0x001f – 0010 100 000011110

0x0020 – 0111 001 xxxxxxxxx

0x0021 – 0000 0101 000 100 xx

0x0022 – 1000 000000000000

0x0023 – 0000 0101 000 100 xx

0x0024 – 0010 100 000010110

0x0025 – 0000 0101 000 110 xx

0x0026 -- 0010 100 000010011

0x0027 – 0000 1001 110 101 xx

0x0028 – 0000 0101 111 000 xx

0x0029 – 1000 000000101010

0x002a – 0000 0101 111 000 xx

0x002b – 0000 0101 000 100 xx

0x002c – 0011 111 000000111

0x002d – 0000 0101 101 000 xx

0x002f – 0000 1010 000 110 xx

0x0030 – 0000 0110 000 110 xx

0x0031 – 0000 0101 101 000 xx

0x0032 – 0000 0101 000 110 xx

0x0033 – 0100 000000101001

0x0034 – 0000 0101 101 000 xx

0x0035 – 0000 1010 101 110 xx

0x0036 – 0000 0110 000 110 xx

0x0037 – 0000 0101 110 000 xx

0x0038 – 0000 0101 110 101 xx

0x0039 – 0100 000000101001

0x003a – 0000 0111 001 xxxxxxxxx

0x003b – 0000 0101 000 110 xx

0x003c – 0000 0111 001 xxxxxxxxx

0x003d – 0000 01101 000 111 xx

0x003e – 1000 000000000001

relPrime: # n is input in $m as an argument

swap $m, $t0 # store n DONE

li 2 # m = 2 DONE

whileLoop:

sw $sp[0] # stores m at address ($sp + 0) in mem DONE

copy $m, $t2 # copies m to $t2, the second argument reg DONE

swap $m, $t0 # loads n DONE

sw $sp[1] # stores n at address ($sp + 1) in mem DONE

swap $m, $ra DONE

sw $sp[2] # stores the return address in the stack at $sp + 2 DONE

swap $sp, $m DONE

addi $sp, 3 # adds 3 to the stack pointer. DONE

swap $m, $sp # saves the stack pointer, gets back $ra DONE

swap $m, $ra # loads n as first argument. DONE

jal GCD # jump to GCD ( call gcd(n[$m], m[$t2] ) pseudo DONE

copy $m, $s # saves GCD DONE

li 1 DONE

copy $m, $t0 # compare to stored.

lw $sp[-2] # restores M

copy $m, $t2

lw $sp[-1] # restores N

copy $m $t1

lw $sp[0] # restore $ra

copy $m, $ra

copy $sp, $ra

addi -3 # reallocate

copy $m, $sp

copy $t0, $m # set main to GCD

beq $t0, whileLoop

jr $ra

GCD:

swap $m, $t0 DONE

li 0 DONE

swap $m, $t0 DONE

beq $t0, aEqZero # if a ($m) == 0, DONE

swap $m, $t2 # load b DONE

whileLoopGCD:

beq $t0, break # if $m = 0, break DONE

slt $t2, $t1 # if (a > b) $t1 = 1, else $t1 = 0 DONE

swap $s, $m # store b DONE

li 1 # set compare to DONE

swap $s, $m # flag in $t1, compare to in $s DONE

swap $m, $t1 DONE

bne $s else # if not, than else DONE

swap $t1, $m # restore b DONE

sub $t2, $t1 # a = a – b DONE

copy $m, $t2 # store a DONE

swap $t1, $tm # restore b DONE

swap $m, $t2 # place b and a in correct regs. DONE

j whileLoopGCD DONE

else:

swap $t1, $m # restore b DONE

sub $t1, $t2 # b = b - a DONE

copy $m, $t2 # store b DONE

swap $t2, $t1 # restore a DONE

j whileLoopGCD

break:

jr $ra # return

aEqualsZero:

swap $m, $t2 # loads b (second arg, $t2)

jr $ra