```
################################################
# Binary GCD algorithm
#def gcd_bit(a,b):
#    if a == b: return a #(A)
#    if a == 0: return b #(B)
#    if b == 0: return a #(C)
#    if (~a & 1): #(D)
#        if (b &1): #(E)
#            return gcd_bit(a >> 1, b) #(F)
#        else: #(G)
#            return gcd_bit(a >> 1, b >> 1) << 1 #(H)
#    if (~b & 1): #(I)
#        return gcd_bit(a, b >> 1) #(J)
#    if (a > b): #(K)
#        return gcd_bit( (a-b) >> 1, b) #(L)
#    return gcd_bit( (b-a) >> 1, a ) #(M)
################################################


 .data
 x: .word  1
 y:  .word   1
 sonuc: .word    -1

 .text
 .globl main

main:
 la $s0, x
 lw $s1,0($s0)
 lw $s2,4($s0)
 add $t0,$0,$0
geri:
 bne $s1,$s2, ileri1 # x==y
 add $s3,$s1,$0
 j son1

ileri1: bne $s1,$0,ileri3 # x==0
 add $s3,$s2,$0
 j son1

ileri3: bne $s2,$0,ileri4 # y==0
 add $s3,$s1,$0
 j son1

ileri4: nor $s4,$s1,$s1  # x cift, y tek
 andi $s4,$s4,1
 beq $s4,$0,ileri5
 andi $s4,$s2,1
 beq $s4,$0,elsepart
 sra $s1,$s1,1
 j  geri

elsepart: sra $s1,$s1,1  # x cift, y cift
 sra $s2,$s2,1
```

```
 addi $t0,$t0,1
 j geri

ileri5: nor $s4,$s2,$s2  # x tek, y cift
 andi $s4,$s4,1
 beq $s4,$0,ileri6
 sra $s2,$s2,1
 j geri

ileri6: ble $s1,$s2,ileri7 # x tek, y tek, x>y
 sub $s1,$s1,$s2
 sra $s1,$s1,1
 j geri

ileri7: add $s4,$s1,$0  # x tek, y tek, y>=x
 sub $s1,$s2,$s1
 sra $s1,$s1,1
 add $s2,$s4,$0
 j geri

son1: beq $t0,$0,son
 sll $s3,$s3,1
 addi $t0,$t0,-1
 j  son1

son: sw $s3,8($s0)
 li $v0, 10   # code for program end
 syscall
```