1. Escreva um procedimento, em assembly para o MIPS, para dividir dois números inteiros de 32 bits. Use o segundo algoritmo da divisão, apresentado em sala de aula. Escreva um programa, em assembly para o MIPS, usando este procedimento para realizar a divisão x ÷ y, com x = 0x90357274 y = 0x12341234. Repita a divisão com x = 0x12341234 y = 0x90357274. Mostre a saída da execução do seu programa no programa MARS. Verifique se o resultado da divisão apresentado pelo programa está correto.

Código do programa:

```
.text:
init:
    jal main
    addi $a0, $v0, 0
    addi $v0, $zero, 17
    syscall

# divisao(int x, int y) -> int quociente, int resto
#         $a0    $a1       $v0            $v1
# pilha (-20):
# 00: dividendo
# 04: divisor
# 08: resto
# 12: iteracao
# 16: $ra
#
# registradores:
# $t0: resto_hi
# $t1: resto_lo/dividendo
# $t2: divisor
# $t3: iteracao
# $t4: MSB resto_lo
# $t5: testes
divisao:
    # prologo:
    addi $sp, $sp, -20 #ajustando a pilha
    sw $ra, 16($sp) #salvando $ra na pilha

    # corpo:
    addi $t0, $zero, 0 #t0(resto_hi) = 0
    addi $t1, $a0, 0 #t1(resto_lo) = x
    addi $t2, $a1, 0 #t2(divisor) = y
    addi $t3, $zero, 0 #t3(iteracao) = 0

    #salvando valores na pilha
    sw $t1, 0($sp) #salvando dividendo na pilha
    sw $t2, 4($sp) #salvando divisor na pilha
    sw $t0, 8($sp) #salvando resto na pilha
    sw $t3, 12($sp) #salvando iteracao na pilha

    #printando valores (iteracao, divisor, resto_hi, resto_lo)
    addi $a0, $t3, 0
```

```
    addi $a1, $t2, 0
    addi $a2, $t0, 0
    addi $a3, $t1, 0
    jal print

    #pegando valores da pilha
    lw $t1, 0($sp) #pegando dividendo na pilha
    lw $t2, 4($sp) #pegando divisor na pilha
    lw $t0, 8($sp) #pegando resto na pilha
    lw $t3, 12($sp) #pegando iteracao na pilha

    #resto = resto << 1
    srl $t4, $t1, 31 #t4 = MSB resto_lo
    sll $t1, $t1, 1 #resto_lo = resto_lo << 1
    sll $t0, $t0, 1 #resto_hi = resto_hi << 1
    or $t0, $t0, $t4 #lsb resto_hi = msb antigo resto_lo

    #salvando valores na pilha
    sw $t1, 0($sp) #salvando dividendo na pilha
    sw $t2, 4($sp) #salvando divisor na pilha
    sw $t0, 8($sp) #salvando resto na pilha
    sw $t3, 12($sp) #salvando iteracao na pilha

    #printando valores (iteracao, divisor, resto_hi, resto_lo)
    addi $a0, $t3, 0
    addi $a1, $t2, 0
    addi $a2, $t0, 0
    addi $a3, $t1, 0
    jal print

    j divisao_for_teste

divisao_for_codigo:
    #pegando valores da pilha
    lw $t1, 0($sp) #pegando dividendo na pilha
    lw $t2, 4($sp) #pegando divisor na pilha
    lw $t0, 8($sp) #pegando resto na pilha
    lw $t3, 12($sp) #pegando iteracao na pilha
    #iteracao = iteracao + 1
    addi $t3, $t3, 1

divisao_for_if_teste:
    #if (resto_hi < divisor)
    sltu $t5, $t0, $t2
    bne $t5, $zero, divisao_for_if_v

divisao_for_if_f:
    #resto = resto - divisor
    subu $t0, $t0, $t2
    #resto = resto << 1
```

```
    srl $t4, $t1, 31 #t4 = MSB resto_lo
    sll $t1, $t1, 1 #resto_lo = resto_lo << 1
    sll $t0, $t0, 1 #resto_hi = resto_hi << 1
    or $t0, $t0, $t4 #lsb resto_hi = msb antigo resto_lo

    #resto[0] = 1
    ori $t1, $t1, 1

    j divisao_for_cont

divisao_for_if_v:
    #resto = resto << 1
    srl $t4, $t1, 31 #t4 = MSB resto_lo
    sll $t1, $t1, 1 #resto_lo = resto_lo << 1
    sll $t0, $t0, 1 #resto_hi = resto_hi << 1
    or $t0, $t0, $t4 #lsb resto_hi = msb antigo resto_lo

    #resto[0] = 0
    #vai ser 0 de qualquer jeito

divisao_for_cont:
    #salvando valores na pilha
    sw $t1, 0($sp) #salvando dividendo na pilha
    sw $t2, 4($sp) #salvando divisor na pilha
    sw $t0, 8($sp) #salvando resto na pilha
    sw $t3, 12($sp) #salvando iteracao na pilha

    #printando valores (iteracao, divisor, resto_hi, resto_lo)
    addi $a0, $t3, 0
    addi $a1, $t2, 0
    addi $a2, $t0, 0
    addi $a3, $t1, 0
    jal print

divisao_for_teste:
    #if (iteracao < 32)
    slti $t5, $t3, 32
    bne $t5, $zero, divisao_for_codigo

divisao_fim:
    #iteracao = iteracao + 1
    addi $t3, $t3, 1

    #resto_hi = resto_hi >> 1
    srl $t0, $t0, 1

    #salvando valores na pilha
    sw $t1, 0($sp) #salvando dividendo na pilha
    sw $t2, 4($sp) #salvando divisor na pilha
    sw $t0, 8($sp) #salvando resto na pilha
```

```
    sw $t3, 12($sp) #salvando iteracao na pilha

    #printando valores (iteracao, divisor, resto_hi, resto_lo)
    addi $a0, $t3, 0
    addi $a1, $t2, 0
    addi $a2, $t0, 0
    addi $a3, $t1, 0
    jal print

    #pegando valores da pilha
    lw $t1, 0($sp) #pegando dividendo na pilha
    lw $t2, 4($sp) #pegando divisor na pilha
    lw $t0, 8($sp) #pegando resto na pilha
    lw $t3, 12($sp) #pegando iteracao na pilha

    # epilogo
    addi $v0, $t1, 0 #v0 = quociente
    addi $v1, $t0, 0 #v1 = resto
    lw $ra, 16($sp) #recuperando $ra
    addi $sp, $sp, 20 #reajustando pilha
    jr $ra


#print(a0, a1, a2, a3)
print:
    # print a0
    addi $v0, $zero, 36
    syscall

    # print ' '
    addi $v0, $zero, 11
    addi $a0, $zero, 32
    syscall

    # print a1
    addi $v0, $zero, 36
    addi $a0, $a1, 0
    syscall

    # print ' '
    addi $v0, $zero, 11
    addi $a0, $zero, 32
    syscall

    # print a2
    addi $v0, $zero, 36
    addi $a0, $a2, 0
    syscall

    # print ' '
```

```
    addi $v0, $zero, 11
    addi $a0, $zero, 32
    syscall

    # print a3
    addi $v0, $zero, 36
    addi $a0, $a3, 0
    syscall

    # print '\n'
    addi $v0, $zero, 11
    addi $a0, $zero, 10
    syscall

    jr $ra


#int main()->0
#pilha: -12
#sp+00: x
#sp+04: y
#sp+08: $ra
main:
    # prologo:
    addi $sp, $sp, -12
    sw $ra, 8($sp)

    # corpo:
    #$a0 = 0x90357274
    lui $a0, 0x9035
    ori $a0, $a0, 0x7274

    #$a1 = 0x12341234
    lui $a1, 0x1234
    ori $a1, $a1, 0x1234

    jal divisao

    # print '\n'
    addi $v0, $zero, 11
    addi $a0, $zero, 10
    syscall

    #$a0 = 0x12341234
    lui $a0, 0x1234
    ori $a0, $a0, 0x1234

    #$a1 = 0x90357274
    lui $a1, 0x9035
    ori $a1, $a1, 0x7274
```

```
    jal divisao

    # epilogo:
    addi $v0, $zero, 0
    lw $ra, 8($sp)
    addi $sp, $sp, 12
    jr $ra
```

Saída do programa (copiada do programa MARS):
OBS: A saída mostra iteração, divisor, resto (resto[63:32]), dividendo/quociente (resto[31:0]).

```
0 305402420 0 2419421812
0 305402420 1 543876328
1 305402420 2 1087752656
2 305402420 4 2175505312
3 305402420 9 56043328
4 305402420 18 112086656
5 305402420 36 224173312
6 305402420 72 448346624
7 305402420 144 896693248
8 305402420 288 1793386496
9 305402420 576 3586772992
10 305402420 1153 2878578688
11 305402420 2307 1462190080
12 305402420 4614 2924380160
13 305402420 9229 1553793024
14 305402420 18458 3107586048
15 305402420 36917 1920204800
16 305402420 73834 3840409600
17 305402420 147669 3385851904
18 305402420 295339 2476736512
19 305402420 590679 658505728
20 305402420 1181358 1317011456
21 305402420 2362716 2634022912
22 305402420 4725433 973078528
23 305402420 9450866 1946157056
24 305402420 18901732 3892314112
25 305402420 37803465 3489660928
26 305402420 75606931 2684354560
27 305402420 151213863 1073741824
28 305402420 302427726 2147483648
29 305402420 604855453 0
30 305402420 598906066 1
31 305402420 587007292 3
32 305402420 563209744 7
33 305402420 281604872 7

0 2419421812 0 305402420
0 2419421812 0 610804840
1 2419421812 0 1221609680
```

```
2 2419421812 0 2443219360
3 2419421812 1 591471424
4 2419421812 2 1182942848
5 2419421812 4 2365885696
6 2419421812 9 436804096
7 2419421812 18 873608192
8 2419421812 36 1747216384
9 2419421812 72 3494432768
10 2419421812 145 2693898240
11 2419421812 291 1092829184
12 2419421812 582 2185658368
13 2419421812 1165 76349440
14 2419421812 2330 152698880
15 2419421812 4660 305397760
16 2419421812 9320 610795520
17 2419421812 18640 1221591040
18 2419421812 37280 2443182080
19 2419421812 74561 591396864
20 2419421812 149122 1182793728
21 2419421812 298244 2365587456
22 2419421812 596489 436207616
23 2419421812 1192978 872415232
24 2419421812 2385956 1744830464
25 2419421812 4771912 3489660928
26 2419421812 9543825 2684354560
27 2419421812 19087651 1073741824
28 2419421812 38175302 2147483648
29 2419421812 76350605 0
30 2419421812 152701210 0
31 2419421812 305402420 0
32 2419421812 610804840 0
33 2419421812 305402420 0
```

0x90357274 = 2419421812
0x12341234 = 305402420
2419421812 // 305402420 = 7
2419421812 % 305402420 = 281604872
305402420 // 2419421812 = 0
305402420 % 2419421812 = 305402420