CORDIC ASSEMBLY

```
#t0 - Angle, t1 - It, t2 - x, t3 - y, t4 - (x>>i), t5 - (y>>i), t6 - I, t7 - sumAngle, t8 - j, t9 - temp e s3 - angleTable[i]

.text
        addi $t0, $t0, 90           # Carrega o ângulo
        sll  $t0, $t0, 24           # angle <<= 24
        addi $t2, $zero, 0x9B74ED   # Carrega valor inicial de x
        addi $t1, $t1, 32           # Carrega valor de iterações

for:
        sub $t9, $t1, $t6           # t9 <= it-i
        beq $t9, $zero, endcode     # Se t9 for igual a zero pula pro fim

        # Deslocamento de x e y
        srav $t4, $t2, $t6          # Shift aritmético do x (x>>i)
        srav $t5, $t3, $t6          # Shift aritmético do y (y>>i)

        # Leitura da memória de dados
        la  $s3, angleTable         # Carrega o endereço inicial do angleTable
        add $s3, $s3, $t8           # Endereçamento do angleTable[i]
        lw  $s3, ($s3)              # Carrega dado de angleTable[i]

        # Teste do if/else
        slt $t9, $t7, $t0           # Se angle > sumAngle, t9 <- 1
        beq $t9, $zero, else        # Se angle <= sumAngle, salta para else

if:
        sub $t2, $t2, $t5           # x = x - (y>>i)
        add $t3, $t3, $t4           # y = y + (x>>i)
        add $t7, $t7, $s3           # sA = sA + angleTable[i]

        j endfor                    # Salta incondicionalmente para endfor

else:
        add $t2, $t2, $t5           # x = x + (y>>i)
        sub $t3, $t3, $t4           # y = y - (x>>i)
        sub $t7, $t7, $s3           # sA = sA - angleTable[i]

endfor:
        addi $t6, $t6, 1            # i++ (para a comparação e deslocamento)
        addi $t8, $t8, 4            # j+4 (para endereçar memória)

        j for                       # Salto incondicional para o for

endcode:


.data
angleTable: .word 0x2d000000 0x1a90a731 0x0e094740 0x07200112 0x03938aa6 0x01ca3794
0x00e52a1a 0x007296d7 0x00394ba5 0x001ca5d9 0x000e52ed 0x00072976
0x000394bb 0x0001ca5d 0x0000e52e 0x00007297 0x0000394b 0x00001ca5
0x00000e52 0x00000729 0x00000394 0x000001ca 0x000000e5 0x00000072
0x00000039 0x0000001c 0x0000000e 0x00000007 0x00000003 0x00000001
0x00000000 0x00000000
```

TEMPOS DE EXECUÇÃO DE CADA IMPLEMENTAÇÃO DO CORDIC EM VHDL:

- Estrutural + Comportamental = 194 ciclos de clock
- Totalmente Comportamental = 66 ciclos de clock
- MIPS Multiciclo = 2069 ciclos de clock