

## Codigo DSP:

LIST

; en X estan los bits de parte entera  
; en Y estan los bits fraccionarios

ORG X:\$999  
VARIABLE\_X\_1 DS 500

ORG X:\$1999  
VARIABLE\_Y\_1 DS 500

ORG P:\$E000

```
main  move  #$1000,r0 ; posicion inicial de X
      move  #$2000,r1 ; posicion inicial de Y

      ; inicializamos x(0)=1 para la respuesta al impulso

      move  #$100000,a0
      move  a0,x:(r0)

      ; reseteo posiciones iniciales

      move  #$1000,r0 ; posicion inicial de X
      move  #$2000,r1 ; posicion inicial de Y

      ; al iniciar el loop r0 apunta a X(n), r1 apunta a Y(n)

      ; empieza el ciclo
      ;  $Y(n) = X(n) + X(n-1) * e1 + y(n-1) * e1 * e2$   $n \geq 0$ 

      DO    #10,END1

      ;Primer paso a  $\leq X(n)$ 

      move  x:(r0),a
```

; Segundo paso  $a += X(n-1) * e1$

move x:-(r0),x0

move #\$780000,y0 ; asigno  $e1=0.5+0.25+0.125+0.0625=0.9375$

mac y0,x0,a ; realizo  $a+=x0 * y0$

; Tercer paso  $a += Y(n-1) * e1 * e2$

move x:-(r1),x0

move x:(r1)+,x0 ;r1++

move #\$780000,y0 ; asigno e1

move #\$780000,y1 ; asigno  $e2=e1$

mpy y0,y1,b

move b1,y0

mac x0,y0,a

; Cuarto paso  $Y(n) = a$

move a1,x:(r1)

move x:(r0)+,a ;r0++

move x:(r0)+,a ;r0++

move x:(r1)+,a ;r1++

END1

vvv nop

end main

### Código Python:

```
from scipy import signal
import matplotlib.pyplot as plt

import numpy
```

```

def generateSystem(e1, e2):
    return signal.dlti([1, e1], [1, -e1*e2], dt=1)

def addPlot(e1, e2):
    input = [0.125]

    for x in range(99):
        input.append(0)

    system = generateSystem(e1, e2)

    x_val_input = [x for x in range(len(input))]

    x_val_output, output = system.output(input, x_val_input)

    plt.plot(x_val_output, output, label = "Values = %d, %d" % (e1, e2))

def computeValues(e1, e2, x_input , input):
    system = generateSystem(e1, e2)

    x_val_output, output = system.output(input, x_input)

    return x_val_output, output

input = [0.125, 0, 0, 0, 0, 0, 0, 0, 0, 0]
x_input = [x for x in range(len(input))]

addPlot(0.9375, 0.9375)

x_val_output, output = computeValues(0.9375, 0.9375, x_input, input)

print(x_input, input)
print(x_val_output, output)

#addPlot(0, 0)
#addPlot(1, 0)
#addPlot(2, 0)
#addPlot(3, 0)
#addPlot(4, 0)

#addPlot(0.999, 0.999)

#addPlot(1, 1)

```

```
plt.legend()
plt.show()
```

Validación:

Código DSP:

```

cyc=003298 ictr= 000123 cnt1= 000000 cnt2= 000000 cnt3= 000000 cnt4=
p:$e018 546100 = move a1,x:(r1)
evaluate X:$2000
Hex:100000 Uns:1048576 Fract: 0.125 Bin:000100000000000000000000
evaluate X:$2001
Hex:1d1000 Uns:1904640 Fract: 0.2270508 Bin:000111010001000000000000
evaluate X:$2002
Hex:198b10 Uns:1674000 Fract: 0.1995564 Bin:000110011000101100010000
evaluate X:$2003
Hex:167339 Uns:1471289 Fract: 0.1753913 Bin:000101100111001100111001
evaluate X:$2004
Hex:13bb45 Uns:1293125 Fract: 0.1541525 Bin:000100111011101101000101
evaluate X:$2005
Hex:115797 Uns:1136535 Fract: 0.1354855 Bin:000100010101011110010111
evaluate X:2006
Hex:000000 Uns:000000 Fract: 0 Bin:000000000000000000000000
0>

```

Código Python:

```

[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.] [[0.125      ]
 [0.22705078]
 [0.19955635]
 [0.17539132]
 [0.15415253]
 [0.13548562]
 [0.11907916]
 [0.10465942]
 [0.09198582]
 [0.08084691]]

```