

Ej6.pdf



GeXx_



Estructura de Computadores



2º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

te imaginas

No pagar ni primera
ni segunda matrícula??

participa
aquí

Si consigues subir
más apuntes que
tus compañeros te
regalamos una
matrícula
valorada en 1000€

WUOLAH



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ventajas

PRO



Di adiós a la publi en los apuntes y en la web



Participa gratis en todos los sorteos



Descarga carpetas completas

estudia sin publi
WUOLAH PRO

```
.set GPBASE, 0x3F200000
.set GPFSEL0, 0x00
.set GPSET0, 0x1c
.set GPCLR0, 0x28
.set STBASE, 0x3F003000
.set STCLO, 0x04

.text
mov r0, #0b11010011
msr cpsr_c, r0
mov sp, #0x08000000 @ Init stack in SVC mode
ldr r4, =GPBASE
mov r5, #0b00001000000000000000000000000000
str r5, [r4, #GPFSEL0] @ Configure GPIO9
mov r5, #0b0000000000000000000000001000000000 @ dejamos en r5 la
direccion del led rojo
ldr r0, =STBASE @ r0 is an input parameter (ST base address)
ldr r1, =1000000 @ r1 is an input parameter (waiting time in
microseconds)
ldr r2, =500000 @ r2 is an input parameter (waiting time in
microseconds)
ldr r3, =250000 @ r3 is an input parameter (waiting time in
microseconds)
ldr r7, =2000000 @ r7 tiempo de espera entre secuencias
bucle:
str r5, [r4, #GPSET0] @ Turn LED on
bl espera @ Call waiting routine
str r5, [r4, #GPCLR0] @ Turn LED off
bl espera3 @ Call waiting routine
str r5, [r4, #GPSET0] @ Turn LED on
bl espera1 @ Call waiting routine
str r5, [r4, #GPCLR0] @ Turn LED off
bl espera3 @ Call waiting routine
str r5, [r4, #GPSET0] @ Turn LED on
bl espera2 @ Call waiting routine
str r5, [r4, #GPCLR0] @ Turn LED off
bl espera3 @ Call waiting routine

b bucle

espera: push {r4, r5} @ Save r4 and r5 in the stack
ldr r4, [r0, #STCLO] @ Load CLO timer
add r4, r1 @ Add waiting time -> this is our ending
time
ret: ldr r5, [r0, #STCLO] @ Enter waiting loop: load current CLO
timer
cmp r5, r4 @ Compare current time with ending time
blo ret @ If lower, go back to read timer again
pop {r4, r5} @ Restore r4 and r5
bx lr @ Return from routine

espera1: push {r4, r5} @ Save r4 and r5 in the stack
ldr r4, [r0, #STCLO] @ Load CLO timer
add r4, r2 @ Add waiting time -> this is our ending
time
ret1: ldr r5, [r0, #STCLO] @ Enter waiting loop: load current CLO
timer
cmp r5, r4 @ Compare current time with ending time
blo ret1 @ If lower, go back to read timer again
```

```

        pop {r4, r5}          @ Restore r4 and r5
        bx lr                 @ Return from routine

espera2: push {r4, r5}        @ Save r4 and r5 in the stack
        ldr r4, [r0, #STCLO]  @ Load CLO timer
        add r4, r3            @ Add waiting time -> this is our ending
time
ret2:   ldr r5, [r0, #STCLO]  @ Enter waiting loop: load current CLO
timer
        cmp r5, r4           @ Compare current time with ending time
        blo ret2             @ If lower, go back to read timer again
        pop {r4, r5}         @ Restore r4 and r5
        bx lr                 @ Return from routine

espera3: push {r4, r5}        @ Save r4 and r5 in the stack
        ldr r4, [r0, #STCLO]  @ Load CLO timer
        add r4, r7            @ Add waiting time -> this is our ending
time
ret3:   ldr r5, [r0, #STCLO]  @ Enter waiting loop: load current CLO
timer
        cmp r5, r4           @ Compare current time with ending time
        blo ret2             @ If lower, go back to read timer again
        pop {r4, r5}         @ Restore r4 and r5
        bx lr                 @ Return from routine

```