.cpu arm7tdmi

.eabi\_attribute 20, 1

.eabi\_attribute 21, 1

.eabi\_attribute 23, 3

.eabi\_attribute 24, 1

.eabi\_attribute 25, 1

.eabi\_attribute 26, 1

.eabi\_attribute 30, 6

.eabi\_attribute 34, 0

.eabi\_attribute 18, 4

.file "gpio\_set.c"

.text

;declaracion de funcion “GPIO\_SetPinDIR”

;void GPIO\_SetPinDIR(GPIO\_T \*pGPIO, unsigned char port, unsigned char pin, IO\_t IO)

;{ if(IO==OUTPUT) (pGPIO->DIR)[port]|=(1<<pin);

; else if(IO==INPUT) (pGPIO->DIR)[port]&= ~(1<<pin); }

.align 2

.global GPIO\_SetPinDIR

.syntax unified

.arm

.fpu softvfp

.type GPIO\_SetPinDIR, %function

GPIO\_SetPinDIR:

@ Function supports interworking.

@ args = 0, pretend = 0, frame = 8

@ frame\_needed = 1, uses\_anonymous\_args = 0

@ link register save eliminated.

str fp, [sp, #-4]!

add fp, sp, #0

sub sp, sp, #12

str r0, [fp, #-8] ;guarda el puntero pGPIO

mov r0, r1

mov r1, r2

mov r2, r3

mov r3, r0

strb r3, [fp, #-9] ;guarda el argumento unsigned char port

mov r3, r1

strb r3, [fp, #-10] ;guarda el argumento unsigned char pin

mov r3, r2

strb r3, [fp, #-11] ;guarda el argumento IO\_t IO

ldrb r3, [fp, #-11] @ zero\_extendqisi2 ;carga IO en r3

;if(IO==OUTPUT)

cmp r3, #1 ;compara IO con 1 (output)

bne .L2 ;si el resultado es false → salta a el else if

;(pGPIO->SET)[port] |= (1<<pin);

ldrb r2, [fp, #-9] @ zero\_extendqisi2 ;carga el argumento port en r2

ldrb r1, [fp, #-9] @ zero\_extendqisi2 ;carga el argumento port en r1

ldr r3, [fp, #-8] ;carga el puntero pGPIO en r3

add r1, r1, #2048 ;r1=r1+0b100000000000

ldr r3, [r3, r1, lsl #2] ;carga la direccion de pGPIO[port]

ldrb r1, [fp, #-10] @ zero\_extendqisi2 ;carga pin en r1

mov r0, #1

lsl r1, r0, r1 ;(1<<pin)

;or y asignacion → “ |=”

orr r1, r3, r1

ldr r3, [fp, #-8]

add r2, r2, #2048

str r1, [r3, r2, lsl #2]

b .L4 ;branch incondicional a la salida de la función

.L2: ;else if(IO==INPUT)

ldrb r3, [fp, #-11] @ zero\_extendqisi2 ;carga IO en r3

cmp r3, #0 ;compara IO con 0 (input)

bne .L4 ;si el resultado es false → sale del if (y de la funcion)

;(pGPIO->DIR)[port]&= ~(1<<pin);

ldrb r2, [fp, #-9] @ zero\_extendqisi2 ;carga el argumento port en r2

ldrb r1, [fp, #-9] @ zero\_extendqisi2 ;carga el argumento port en r1

ldr r3, [fp, #-8] ;carga el puntero pGPIO en r2

add r1, r1, #2048 ;r1=r1+0b100000000000

ldr r3, [r3, r1, lsl #2]

;1<<pin

ldrb r1, [fp, #-10] @ zero\_extendqisi2

mov r0, #1

lsl r1, r0, r1

; ~(1<<pin)

mvn r1, r1 ;niega r1

;and y asignacion → “&=“

and r1, r1, r3

ldr r3, [fp, #-8]

add r2, r2, #2048 ;r2=r2+0b100000000000

str r1, [r3, r2, lsl #2] ;guarda el nuevo valor de (pGPIO->DIR)[port]

.L4: ;fuera del if

nop

add sp, fp, #0

@ sp needed

ldr fp, [sp], #4

bx lr

.size GPIO\_SetPinDIR, .-GPIO\_SetPinDIR

;Las funciones siguientes funcionan similar a GPIO\_SetPinDIR

;declaracion de funcion “GPIO\_SetPin”

.align 2

.global GPIO\_SetPin

.syntax unified

.arm

.fpu softvfp

.type GPIO\_SetPin, %function

GPIO\_SetPin:

@ Function supports interworking.

@ args = 0, pretend = 0, frame = 8

@ frame\_needed = 1, uses\_anonymous\_args = 0

@ link register save eliminated.

str fp, [sp, #-4]!

add fp, sp, #0

sub sp, sp, #12

str r0, [fp, #-8]

mov r0, r1

mov r1, r2

mov r2, r3

mov r3, r0

strb r3, [fp, #-9]

mov r3, r1

strb r3, [fp, #-10]

mov r3, r2

strb r3, [fp, #-11]

ldrb r3, [fp, #-11] @ zero\_extendqisi2

cmp r3, #1

bne .L6

ldrb r2, [fp, #-9] @ zero\_extendqisi2

ldrb r1, [fp, #-9] @ zero\_extendqisi2

ldr r3, [fp, #-8]

add r1, r1, #2176

ldr r3, [r3, r1, lsl #2]

ldrb r1, [fp, #-10] @ zero\_extendqisi2

mov r0, #1

lsl r1, r0, r1

orr r1, r3, r1

ldr r3, [fp, #-8]

add r2, r2, #2176

str r1, [r3, r2, lsl #2]

b .L8

.L6:

ldrb r3, [fp, #-11] @ zero\_extendqisi2

cmp r3, #0

bne .L8

ldrb r2, [fp, #-9] @ zero\_extendqisi2

ldrb r1, [fp, #-9] @ zero\_extendqisi2

ldr r3, [fp, #-8]

add r1, r1, #2208

ldr r3, [r3, r1, lsl #2]

ldrb r1, [fp, #-10] @ zero\_extendqisi2

mov r0, #1

lsl r1, r0, r1

orr r1, r3, r1

ldr r3, [fp, #-8]

add r2, r2, #2208

str r1, [r3, r2, lsl #2]

.L8:

nop

add sp, fp, #0

@ sp needed

ldr fp, [sp], #4

bx lr

.size GPIO\_SetPin, .-GPIO\_SetPin

;declaracion de funcion “GPIO\_GetPinState”

.align 2

.global GPIO\_GetPinState

.syntax unified

.arm

.fpu softvfp

.type GPIO\_GetPinState, %function

GPIO\_GetPinState:

@ Function supports interworking.

@ args = 0, pretend = 0, frame = 8

@ frame\_needed = 1, uses\_anonymous\_args = 0

@ link register save eliminated.

str fp, [sp, #-4]!

add fp, sp, #0

sub sp, sp, #12

str r0, [fp, #-8]

mov r3, r1

strb r3, [fp, #-9]

mov r3, r2

strb r3, [fp, #-10]

ldrb r2, [fp, #-9] @ zero\_extendqisi2

ldrb r3, [fp, #-10] @ zero\_extendqisi2

ldr r1, [fp, #-8]

lsl r2, r2, #5

add r2, r1, r2

add r3, r2, r3

ldrb r3, [r3] @ zero\_extendqisi2

mov r0, r3

add sp, fp, #0

@ sp needed

ldr fp, [sp], #4

bx lr

.size GPIO\_GetPinState, .-GPIO\_GetPinState

;declaración de funcion “GPIO\_TogglePin”

.align 2

.global GPIO\_TogglePin

.syntax unified

.arm

.fpu softvfp

.type GPIO\_TogglePin, %function

GPIO\_TogglePin:

@ Function supports interworking.

@ args = 0, pretend = 0, frame = 8

@ frame\_needed = 1, uses\_anonymous\_args = 0

@ link register save eliminated.

str fp, [sp, #-4]!

add fp, sp, #0

sub sp, sp, #12

str r0, [fp, #-8]

mov r3, r1

strb r3, [fp, #-9]

mov r3, r2

strb r3, [fp, #-10]

ldrb r2, [fp, #-9] @ zero\_extendqisi2

ldrb r1, [fp, #-9] @ zero\_extendqisi2

ldr r3, [fp, #-8]

add r1, r1, #2240

ldr r3, [r3, r1, lsl #2]

ldrb r1, [fp, #-10] @ zero\_extendqisi2

mov r0, #1

lsl r1, r0, r1

orr r1, r3, r1

ldr r3, [fp, #-8]

add r2, r2, #2240

str r1, [r3, r2, lsl #2]

nop

add sp, fp, #0

@ sp needed

ldr fp, [sp], #4

bx lr