.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 "interruptions.c"

.text

;declaracion de funcion Enable NVIC

;void Enable\_NVIC(interrupt\_t interrnum)

.align 2

.global Enable\_NVIC

.syntax unified

.arm

.fpu softvfp

.type Enable\_NVIC, %function

Enable\_NVIC:

@ Function supports interworking.

@ args = 0, pretend = 0, frame = 16

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

@ link register save eliminated.

str fp, [sp, #-4]! ;guarda el “frame pointer” y el “Link register” en el stack

;y espacio para el puntero

add fp, sp, #0

sub sp, sp, #20 ;espacio para lo que ocupa el frame

mov r3, r0 ;carga r0 en r3

;en r3 está el el parámetro de entrada interrnum

strb r3, [fp, #-13] ;guarda en la posición fp-13,que ahora corresponde a la

;variable interrnum

ldrsb r3, [fp, #-13]

;if(0<=interrnum && interrnum<=31)

;0<=interrnum

cmp r3, #0

blt .L2 ;si es false hace branch al próximo elseif

ldrsb r3, [fp, #-13]

;interrnum<=31

cmp r3, #31

bgt .L2 ;si es false hace branch al próximo elseif

;dentro del if

;isernum=0;

mov r3, #0

strb r3, [fp, #-5] ;guarda en la posición fp-5,que ahora corresponde a la

;variable isernum

b .L3 ;sale del if

.L2: ;en C: else if (32<=interrnum && interrnum<=63)

;32<=interrnum

ldrsb r3, [fp, #-13]

cmp r3, #31

ble .L6 ;próximo else

;interrnum<=63

ldrsb r3, [fp, #-13]

cmp r3, #63

bgt .L6 ;próximo else

;isernum=1;

mov r3, #1

strb r3, [fp, #-5]

ldrb r3, [fp, #-13] @ zero\_extendqisi2

;interrnum -= 32;

sub r3, r3, #32

and r3, r3, #255

strb r3, [fp, #-13]

.L3:

;(NVIC->ISER)[isernum]|=(1<<(unsigned int)interrnum);

ldr r0, .L7 ;carga en r0 la direccion de NVIC

ldrb r3, [fp, #-5] @ zero\_extendqisi2 ;carga en r3 isernum

ldr r1, .L7 ;carga en r0 la direccion de NVIC

ldrb r2, [fp, #-5] @ zero\_extendqisi2 ;carga en r2 isernum

ldr r1, [r1, r2, lsl #2] ;r1 <-- direccion de r1 con un offset de (r2<<2)

;r2 tiene isernum

;se mueve 2 por ser “ISER”, el segundo campo

;de la estructura NVIC

ldrsb r2, [fp, #-13] ;load de interrnum en r2

mov ip, #1 ;en ip un 1

;1<<(unsigned int)interrnum

lsl r2, ip, r2

;carga y or → “|=”

orr r2, r1, r2

str r2, [r0, r3, lsl #2] ;r2 --> direccion de r0 con un offset de (r3<<2)

;en r0 esta NVIC, y el offset es de

;isernum con corrimiento de 2 por “ISER

b .L1

.L6: ;en C else{return;}

nop ;no hace nada

.L1:

add sp, fp, #0

@ sp needed

ldr fp, [sp], #4

bx lr

.L8:

.align 2

.L7:

.word -536813312

.size Enable\_NVIC, .-Enable\_NVIC

;declaracion de funcion SelectEdge\_PIN\_INT

;void SelectEdge\_PIN\_INT(int ISELnum)

.align 2

.global SelectEdge\_PIN\_INT

.syntax unified

.arm

.fpu softvfp

.type SelectEdge\_PIN\_INT, %function

SelectEdge\_PIN\_INT:

@ 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 argumento de la funcion(ISELnum) en fp-8

ldr r3, [fp, #-8] ;el argumento en r3

;if(ISELnum <=7)

cmp r3, #7 ;comparación r3 (ISELnum) contra 7

bgt .L11 ;sale del if

;dentro del if

;(PIN\_INT->ISEL) &= ~(0x01<<ISELnum);

ldr r1, .L12 ;la dirección de PIN\_INT en r1

ldr r3, .L12 ;la dirección dePIN\_INT en r2

ldr r2, [r3] ;carga en r2 la dirección en r3

mov r0, #1 ;un 1 en r0

ldr r3, [fp, #-8] ;carga de ISELnum en r3

;0x01<<ISELnum

lsl r3, r0, r3 ;r3←(r0 corrido r3)

;es decir un 1 corrido ISELnum veces

;negacion → “~”

mvn r3, r3 ;(move not) → negación de r3

;and → “&= “

and r3, r3, r2 ;r3 & r2 → r3

;~(0x01<<ISELnum) & (PIN\_INT->ISEL) → (PIN\_INT->ISEL)

str r3, [r1] ;se guarda en el resultado

.L11:

nop

add sp, fp, #0

@ sp needed

ldr fp, [sp], #4

bx lr

.L13:

.align 2

.L12:

.word 1074294784

.size SelectEdge\_PIN\_INT, .-SelectEdge\_PIN\_INT

;declaración de función Enable\_PIN\_INT

;void Enable\_PIN\_INT(int SIENRnum)

.align 2

.global Enable\_PIN\_INT

.syntax unified

.arm

.fpu softvfp

.type Enable\_PIN\_INT, %function

Enable\_PIN\_INT:

@ Function supports interworking.

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

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

push {fp, lr}

add fp, sp, #4

sub sp, sp, #8

str r0, [fp, #-8] ;guarda el argumento de la funcion(SIENRnum) en fp-8

ldr r3, [fp, #-8] ;el argumento en r3

;if(SIENRnum <=7)

cmp r3, #7

bgt .L15 ;sale del if

ldr r1, .L16 ;la dirección de PIN\_INT en r1

ldr r3, .L16 ;la dirección de PIN\_INT en r3

ldr r2, [r3, #8] ;offset de 8 sobre r3 y guardado en r2

;offset por ->SIENR

;(PIN\_INT->SIENR)|= (0x01<<SIENRnum);

mov r0, #1 ;carga de un 1 en r0

ldr r3, [fp, #-8] ;el argumento en r3

;0x01<<SIENRnum

lsl r3, r0, r3

;or →” |= “

orr r3, r2, r3

str r3, [r1, #8] ;store del resultado en PIN\_INT->SIENR

.L15:

;SelectEdge\_PIN\_INT(SIENRnum);

ldr r0, [fp, #-8]

bl SelectEdge\_PIN\_INT

nop

sub sp, fp, #4

@ sp needed

pop {fp, lr}

bx lr

.L17:

.align 2

.L16:

.word 1074294784

.size Enable\_PIN\_INT, .-Enable\_PIN\_INT

;declaración de función Select\_GPIO\_interrupt

;void Select\_GPIO\_interrupt(unsigned char port, unsigned char pin,

;unsigned char interrnum)

.align 2

.global Select\_GPIO\_interrupt

.syntax unified

.arm

.fpu softvfp

.type Select\_GPIO\_interrupt, %function

Select\_GPIO\_interrupt:

@ Function supports interworking.

@ args = 0, pretend = 0, frame = 16

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

@ link register save eliminated.

str fp, [sp, #-4]!

add fp, sp, #0

sub sp, sp, #20

mov r3, r0

strb r3, [fp, #-13] ;guarda el argumento de la funcion(port) en fp-13

mov r3, r1

strb r3, [fp, #-14] ;guarda el argumento de la funcion(pin) en fp-14

mov r3, r2

strb r3, [fp, #-15] ;guarda el argumento de la funcion(interrnum) en fp-15

ldrb r3, [fp, #-15] @ zero\_extendqisi2

;int desp=(interrnum & 3) << 3;

lsl r3, r3, #3

and r3, r3, #24

str r3, [fp, #-8] ;guarda resultado de la operación de bits en fp-8

;fp-8 ahora representa la variable desp

;(SCU->PINTSEL)[interrnum >> 2]|= ( ( ( port & 7 ) << 5) | (pin & 0x1F) ) << desp;

ldr r0, .L19 ;dirección de SCU en ro

;interrnum >> 2

ldrb r3, [fp, #-15] @ zero\_extendqisi2 ;carga en r3 de interrnum

lsr r3, r3, #2 ;rr>>2 → r3

and r3, r3, #255

mov ip, r3

ldr r2, .L19 ;carga de dirección de SCU en r2

ldrb r3, [fp, #-15] @ zero\_extendqisi2 ;carga en r3 de interrnum

lsr r3, r3, #2 ;rr>>2 → r3

and r3, r3, #255

;( port & 7 ) << 5

add r3, r3, #896 ;826=binario 1110000000

;es lo mismo que un 7 corrido 5 veces a la izq

ldr r3, [r2, r3, lsl #2]

ldrb r2, [fp, #-13] @ zero\_extendqisi2 ;carga en r2 de port

lsl r2, r2, #5

and r1, r2, #255

;pin & 0x1F

ldrb r2, [fp, #-14] @ zero\_extendqisi2 ;carga en r2 de pin

and r2, r2, #31

;or → ( port & 7 ) << 5) | (pin & 0x1F)

orr r1, r1, r2

;<< desp

ldr r2, [fp, #-8]

lsl r2, r1, r2

;or y asignación → “|=”

orr r2, r3, r2

add r3, ip, #896 ;se le aplica un offset a la dirección del SCU de 896

;(SCU->PINTSEL)

str r2, [r0, r3, lsl #2]

nop

add sp, fp, #0

@ sp needed

ldr fp, [sp], #4

bx lr

.L20:

.align 2

.L19:

.word 1074290688

.size Select\_GPIO\_interrupt, .-Select\_GPIO\_interrupt

.ident "GCC: (15:6.3.1+svn253039-1build1) 6.3.1 20170620"