

# Documentacion Proyecto

Generado por Doxygen 1.8.10

Martes, 20 de Octubre de 2015 07:55:33



# Índice general

<b>1</b>	<b>Índice de estructura de datos</b>	<b>1</b>
1.1	Estructura de datos	1
<b>2</b>	<b>Índice de archivos</b>	<b>3</b>
2.1	Lista de archivos	3
<b>3</b>	<b>Documentación de las estructuras de datos</b>	<b>5</b>
3.1	Referencia de la Estructura <code>ins_t</code>	5
3.1.1	Documentación de los campos	5
3.1.1.1	array	5
3.2	Referencia de la Estructura <code>instruction_t</code>	5
3.2.1	Documentación de los campos	5
3.2.1.1	mnemonic	5
3.2.1.2	op1_type	6
3.2.1.3	op1_value	6
3.2.1.4	op2_type	6
3.2.1.5	op2_value	6
3.2.1.6	op3_type	6
3.2.1.7	op3_value	6
3.2.1.8	registers_list	6
3.3	Referencia de la Estructura <code>port_t</code>	6
3.3.1	Documentación de los campos	6
3.3.1.1	DDR	6
3.3.1.2	Interrupts	6
3.3.1.3	PIN	6
3.3.1.4	Pins	6
3.3.1.5	PORT	6
<b>4</b>	<b>Documentación de archivos</b>	<b>7</b>
4.1	Referencia del Archivo <code>alu.c</code>	7
4.1.1	Documentación de las funciones	7
4.1.1.1	<code>ADD(uint32_t *Ra, uint32_t Rb, uint32_t Rc, bool flg[], int *pc)</code>	7

4.1.1.2	AND(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool *flg, int *pc)	8
4.1.1.3	EOR(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool *flg, int *pc)	9
4.1.1.4	MOV(uint32_t *Ra, uint32_t *Rb, bool *flg, int *pc)	9
4.1.1.5	MUL(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool *flg, int *pc)	9
4.1.1.6	OR(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool *flg, int *pc)	9
4.1.1.7	SUB(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool *flg, int *pc)	10
4.2	Referencia del Archivo alu.h	10
4.2.1	Documentación de las funciones	10
4.2.1.1	ADD(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool flg[], int *pc)	10
4.2.1.2	AND(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool *flg, int *pc)	11
4.2.1.3	EOR(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool *flg, int *pc)	11
4.2.1.4	MOV(uint32_t *Ra, uint32_t *Rb, bool *flg, int *pc)	11
4.2.1.5	MUL(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool *flg, int *pc)	11
4.2.1.6	OR(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool *flg, int *pc)	12
4.2.1.7	SUB(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, bool *flg, int *pc)	12
4.3	Referencia del Archivo banderas.c	12
4.3.1	Documentación de las funciones	12
4.3.1.1	banderas(uint32_t Ra, uint32_t Rb, uint32_t Rc, bool flg[])	12
4.4	Referencia del Archivo banderas.h	13
4.4.1	Documentación de las funciones	13
4.4.1.1	banderas(uint32_t Ra, uint32_t Rb, uint32_t Rc, bool flg[])	13
4.5	Referencia del Archivo decoder.c	13
4.5.1	Documentación de las funciones	14
4.5.1.1	countLines(FILE *fp)	14
4.5.1.2	decodeInstruction(instruction_t instruction, uint32_t *Rd, uint32_t *Rm, uint32_t *Rr, bool flg[], int *pc, uint8_t pila)	14
4.5.1.3	getInstruction(char *instStr)	14
4.5.1.4	readFile(char *filename, ins_t *instructions)	14
4.6	Referencia del Archivo decoder.h	14
4.6.1	Documentación de las funciones	14
4.6.1.1	countLines(FILE *fp)	14
4.6.1.2	readFile(char *filename, ins_t *instructions)	14
4.7	Referencia del Archivo InstruccionesDesplazamiento.c	14
4.7.1	Documentación de las funciones	15
4.7.1.1	ASRS(uint32_t *Ra, uint32_t *Rb, int *pc)	15
4.7.1.2	BIC(uint32_t *Ra, uint32_t *Rb, int *pc)	15
4.7.1.3	CMN(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, int *pc)	15
4.7.1.4	CMP(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, int *pc)	16
4.7.1.5	LSL(uint32_t *Ra, uint32_t *Rb, uint32_t inmediato, int *pc)	16
4.7.1.6	LSR(uint32_t *Ra, uint32_t *Rb, uint32_t inmediato, int *pc)	16

4.7.1.7	MVN(uint32_t *Ra, uint32_t *Rb, int *pc)	16
4.7.1.8	NOP(int *pc)	17
4.7.1.9	ROR(uint32_t *Ra, uint32_t *Rb, int *pc)	17
4.7.1.10	RSBS(uint32_t *Ra, uint32_t *Rb, int *pc)	17
4.8	Referencia del Archivo InstruccionesDesplazamiento.h	17
4.8.1	Documentación de las funciones	18
4.8.1.1	ASRS(uint32_t *Ra, uint32_t *Rb, int *pc)	18
4.8.1.2	BIC(uint32_t *Ra, uint32_t *Rb, int *pc)	18
4.8.1.3	CMN(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, int *pc)	18
4.8.1.4	CMP(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc, int *pc)	19
4.8.1.5	LSL(uint32_t *Ra, uint32_t *Rb, uint32_t inmediato, int *pc)	19
4.8.1.6	LSR(uint32_t *Ra, uint32_t *Rb, uint32_t inmediato, int *pc)	19
4.8.1.7	MVN(uint32_t *Ra, uint32_t *Rb, int *pc)	19
4.8.1.8	NOP(int *pc)	20
4.8.1.9	ROR(uint32_t *Ra, uint32_t *Rb, int *pc)	20
4.8.1.10	RSBS(uint32_t *Ra, uint32_t *Rb, int *pc)	20
4.9	Referencia del Archivo interrupciones.c	20
4.10	Referencia del Archivo interrupciones.h	20
4.11	Referencia del Archivo io.c	20
4.11.1	Documentación de las funciones	21
4.11.1.1	changePinPortA(uint8_t pin, uint8_t value)	21
4.11.1.2	changePinPortB(uint8_t pin, uint8_t value)	21
4.11.1.3	initIO(void)	21
4.11.1.4	IOAccess(uint8_t address, uint8_t *data, uint8_t r_w)	21
4.11.1.5	showFrame(int x, int y, int w, int h)	21
4.11.1.6	showPorts(void)	21
4.11.2	Documentación de las variables	21
4.11.2.1	irq	21
4.11.2.2	PORTA	21
4.11.2.3	PORTB	21
4.12	Referencia del Archivo io.h	21
4.12.1	Documentación de los 'defines'	22
4.12.1.1	BLUEBLACK	22
4.12.1.2	HIGH	22
4.12.1.3	LOW	22
4.12.1.4	Read	22
4.12.1.5	REDBLACK	22
4.12.1.6	WHITEBLACK	22
4.12.1.7	Write	22
4.12.1.8	XINIT	22

4.12.1.9	YINIT	22
4.12.2	Documentación de las funciones	22
4.12.2.1	changePinPortA(uint8_t pin, uint8_t value)	22
4.12.2.2	changePinPortB(uint8_t pin, uint8_t value)	22
4.12.2.3	initIO(void)	22
4.12.2.4	IOAccess(uint8_t address, uint8_t *data, uint8_t r_w)	22
4.12.2.5	showFrame(int x, int y, int w, int h)	22
4.12.2.6	showPorts(void)	22
4.13	Referencia del Archivo Load_Store.c	22
4.13.1	Documentación de las funciones	23
4.13.1.1	LDR(uint32_t *Rt, uint32_t Rn, uint32_t inmed, int *pc)	23
4.13.1.2	LDRB(uint32_t *Rt, uint32_t Rn, uint32_t inmed, int *pc)	23
4.13.1.3	LDRH(uint32_t *Rt, uint32_t Rn, uint32_t inmed, int *pc)	23
4.13.1.4	LDRSB(uint32_t *Rt, uint32_t Rn, uint32_t inmed, int *pc)	23
4.13.1.5	LDRSH(uint32_t *Rt, uint32_t Rn, uint32_t inmed, int *pc)	23
4.13.1.6	STR(uint32_t *Rt, uint32_t Rn, uint32_t inmed, uint32_t SP, int *pc)	23
4.13.1.7	STRB(uint32_t *Rt, uint32_t Rn, uint32_t inmed, uint32_t SP, int *pc)	23
4.13.1.8	STRH(uint32_t *Rt, uint32_t Rn, uint32_t inmed, uint32_t SP, int *pc)	23
4.14	Referencia del Archivo Load_Store.h	23
4.14.1	Documentación de las funciones	24
4.14.1.1	LDR(uint32_t Rt, uint32_t Rn, uint32_t inmed)	24
4.14.1.2	LDRB(uint32_t Rt, uint32_t Rn, uint32_t inmed)	24
4.14.1.3	LDRH(uint32_t Rt, uint32_t Rn, uint32_t inmed)	24
4.14.1.4	LDRSB(uint32_t Rt, uint32_t Rn, uint32_t inmed)	24
4.14.1.5	LDRSH(uint32_t Rt, uint32_t Rn, uint32_t inmed)	25
4.14.1.6	STR(uint32_t Rt, uint32_t Rn, uint32_t inmed, uint32_t SP)	25
4.14.1.7	STRB(uint32_t Rt, uint32_t Rn, uint32_t inmed, uint32_t SP)	25
4.14.1.8	STRH(uint32_t Rt, uint32_t Rn, uint32_t inmed, uint32_t SP)	25
4.15	Referencia del Archivo main.c	26
4.15.1	Documentación de las funciones	26
4.15.1.1	main(void)	26
4.16	Referencia del Archivo ports.c	26
4.16.1	Documentación de las funciones	26
4.16.1.1	main(void)	26
4.16.2	Documentación de las variables	26
4.16.2.1	irq	26
4.17	Referencia del Archivo RAM.c	27
4.17.1	Documentación de las funciones	27
4.17.1.1	bitcount(uint8_t listaregistros[])	27
4.17.1.2	pop(uint8_t *pila, uint32_t *R, uint8_t *ram, uint8_t listaregistros[])	27

4.17.1.3	push(uint8_t *pila, uint32_t *R, uint8_t *ram, uint8_t listaregistros[])	27
4.18	Referencia del Archivo RAM.h	27
4.18.1	Documentación de los 'defines'	27
4.18.1.1	Mema	27
4.18.2	Documentación de las funciones	27
4.18.2.1	bitcount(uint8_t listaregistros[])	27
4.18.2.2	pop(uint8_t *pila, uint32_t *R, uint8_t *ram, uint8_t listaregistros[])	27
4.18.2.3	push(uint8_t *pila, uint32_t *R, uint8_t *ram, uint8_t listaregistros[])	27
4.19	Referencia del Archivo registros.c	27
4.19.1	Documentación de las funciones	28
4.19.1.1	Registros(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc)	28
4.20	Referencia del Archivo registros.h	28
4.20.1	Documentación de las funciones	28
4.20.1.1	Registros(uint32_t *Ra, uint32_t *Rb, uint32_t *Rc)	28
4.21	Referencia del Archivo saltos.c	28
4.21.1	Documentación de las funciones	29
4.21.1.1	B(uint32_t valor, int *pc, bool *flg)	29
4.21.2	Documentación de las variables	29
4.21.2.1	AUX	29
4.22	Referencia del Archivo saltos.h	29
4.22.1	Documentación de las funciones	29
4.22.1.1	B(uint32_t valor, int *pc, bool *flg)	29
4.22.1.2	BAL(uint32_t valor, int *pc, bool *flg)	29
4.22.1.3	BCC(uint32_t valor, int *pc, bool *flg)	29
4.22.1.4	BCS(uint32_t valor, int *pc, bool *flg)	30
4.22.1.5	BEQ(uint32_t valor, int *pc, bool *flg)	30
4.22.1.6	BGE(uint32_t valor, int *pc, bool *flg)	30
4.22.1.7	BGT(uint32_t valor, int *pc, bool *flg)	30
4.22.1.8	BHI(uint32_t valor, int *pc, bool *flg)	30
4.22.1.9	BL(uint32_t valor, int *pc, bool *flg)	30
4.22.1.10	BLE(uint32_t valor, int *pc, bool *flg)	30
4.22.1.11	BLS(uint32_t valor, int *pc, bool *flg)	30
4.22.1.12	BLT(uint32_t valor, int *pc, bool *flg)	30
4.22.1.13	BMI(uint32_t valor, int *pc, bool *flg)	30
4.22.1.14	BNE(uint32_t valor, int *pc, bool *flg)	30
4.22.1.15	BPL(uint32_t valor, int *pc, bool *flg)	30
4.22.1.16	BVC(uint32_t valor, int *pc, bool *flg)	30
4.22.1.17	BVS(uint32_t valor, int *pc, bool *flg)	30
4.22.1.18	BX(uint32_t *pc, bool *flg)	30
4.23	Referencia del Archivo test.c	30





# Capítulo 1

## Índice de estructura de datos

### 1.1. Estructura de datos

Lista de estructuras con una breve descripción:

<a href="#">ins_t</a> . . . . .	5
<a href="#">instruction_t</a> . . . . .	5
<a href="#">port_t</a> . . . . .	6



## Capítulo 2

# Indice de archivos

### 2.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

<a href="#">alu.c</a>	7
<a href="#">alu.h</a>	10
<a href="#">banderas.c</a>	12
<a href="#">banderas.h</a>	13
<a href="#">decoder.c</a>	13
<a href="#">decoder.h</a>	14
<a href="#">InstruccionesDesplazamiento.c</a>	14
<a href="#">InstruccionesDesplazamiento.h</a>	17
<a href="#">interrupciones.c</a>	20
<a href="#">interrupciones.h</a>	20
<a href="#">io.c</a>	20
<a href="#">io.h</a>	21
<a href="#">Load_Store.c</a>	22
<a href="#">Load_Store.h</a>	23
<a href="#">main.c</a>	26
<a href="#">ports.c</a>	26
<a href="#">RAM.c</a>	27
<a href="#">RAM.h</a>	27
<a href="#">registros.c</a>	27
<a href="#">registros.h</a>	28
<a href="#">saltos.c</a>	28
<a href="#">saltos.h</a>	29
<a href="#">test.c</a>	30



## Capítulo 3

# Documentación de las estructuras de datos

### 3.1. Referencia de la Estructura ins\_t

```
#include <decoder.h>
```

#### Campos de datos

- char \*\* [array](#)

#### 3.1.1. Documentación de los campos

##### 3.1.1.1. char\*\* array

La documentación para esta estructura fue generada a partir del siguiente fichero:

- [decoder.h](#)

### 3.2. Referencia de la Estructura instruction\_t

```
#include <decoder.h>
```

#### Campos de datos

- char [mnemonic](#) [10]
- char [op1\\_type](#)
- char [op2\\_type](#)
- char [op3\\_type](#)
- uint32\_t [op1\\_value](#)
- uint32\_t [op2\\_value](#)
- uint32\_t [op3\\_value](#)
- uint8\_t [registers\\_list](#) [16]

#### 3.2.1. Documentación de los campos

##### 3.2.1.1. char mnemonic[10]

3.2.1.2. char op1\_type

3.2.1.3. uint32\_t op1\_value

3.2.1.4. char op2\_type

3.2.1.5. uint32\_t op2\_value

3.2.1.6. char op3\_type

3.2.1.7. uint32\_t op3\_value

3.2.1.8. uint8\_t registers\_list[16]

La documentación para esta estructura fue generada a partir del siguiente fichero:

- [decoder.h](#)

### 3.3. Referencia de la Estructura port\_t

```
#include <io.h>
```

#### Campos de datos

- uint8\_t [DDR](#)
- uint8\_t [PORT](#)
- uint8\_t [PIN](#)
- uint8\_t [Pins](#)
- uint8\_t [Interrupts](#)

#### 3.3.1. Documentación de los campos

3.3.1.1. uint8\_t DDR

3.3.1.2. uint8\_t Interrupts

3.3.1.3. uint8\_t PIN

3.3.1.4. uint8\_t Pins

3.3.1.5. uint8\_t PORT

La documentación para esta estructura fue generada a partir del siguiente fichero:

- [io.h](#)

## Capítulo 4

# Documentación de archivos

### 4.1. Referencia del Archivo alu.c

```
#include "alu.h"
#include <curses.h>
```

#### Funciones

- void **ADD** (uint32\_t \*Ra, uint32\_t Rb, uint32\_t Rc, bool flg[], int \*pc)  
*funcion que suma*
- void **SUB** (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc, bool \*flg, int \*pc)  
*funcion que resta*
- void **MUL** (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc, bool \*flg, int \*pc)  
*funcion que multiplica dos registros*
- void **AND** (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc, bool \*flg, int \*pc)  
*funcion de producto logico*
- void **OR** (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc, bool \*flg, int \*pc)  
*funcion de suma logica*
- void **EOR** (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc, bool \*flg, int \*pc)  
*funcion de or exclusiva*
- void **MOV** (uint32\_t \*Ra, uint32\_t \*Rb, bool \*flg, int \*pc)  
*funcion que escribe un valor de un registro en otro registro*

#### 4.1.1. Documentación de las funciones

4.1.1.1. void **ADD** ( uint32\_t \* *Ra*, uint32\_t *Rb*, uint32\_t *Rc*, bool *flg*[], int \* *pc* )

funcion que suma

##### Parámetros

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

##### Devuelve

no retorna nada

4.1.1.2. void AND ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, bool \* *flg*, int \* *pc* )

funcion de producto logico



## Parámetros

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

## Devuelve

No retorna nada

4.1.1.3. void EOR ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, bool \* *flg*, int \* *pc* )

funcion de or exclusiva

## Parámetros

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

## Devuelve

No retorna nada

4.1.1.4. void MOV ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, bool \* *flg*, int \* *pc* )

funcion que escribe un valor de un registro en otro registro

## Parámetros

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1

## Devuelve

No retorna nada

4.1.1.5. void MUL ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, bool \* *flg*, int \* *pc* )

funcion que multiplica dos registros

## Parámetros

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

## Devuelve

No retorna nada

4.1.1.6. void OR ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, bool \* *flg*, int \* *pc* )

funcion de suma logica

**Parámetros**

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

**Devuelve**

No retorna nada

4.1.1.7. void SUB ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, bool \* *flg*, int \* *pc* )

funcion que resta

**Parámetros**

<i>Ra</i>	guaRaa el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

**Devuelve**

No retorna nada

## 4.2. Referencia del Archivo alu.h

```
#include "banderas.h"
#include <curses.h>
```

**Funciones**

- void **ADD** (uint32\_t \**Ra*, uint32\_t *Rb*, uint32\_t *Rc*, bool *flg*[], int \**pc*)  
*funcion que suma*
- void **SUB** (uint32\_t \**Ra*, uint32\_t \**Rb*, uint32\_t \**Rc*, bool \**flg*, int \**pc*)  
*funcion que resta*
- void **AND** (uint32\_t \**Ra*, uint32\_t \**Rb*, uint32\_t \**Rc*, bool \**flg*, int \**pc*)  
*funcion de producto logico*
- void **OR** (uint32\_t \**Ra*, uint32\_t \**Rb*, uint32\_t \**Rc*, bool \**flg*, int \**pc*)  
*funcion de suma logica*
- void **EOR** (uint32\_t \**Ra*, uint32\_t \**Rb*, uint32\_t \**Rc*, bool \**flg*, int \**pc*)  
*funcion de or exclusiva*
- void **MOV** (uint32\_t \**Ra*, uint32\_t \**Rb*, bool \**flg*, int \**pc*)  
*funcion que escribe un valor de un registro en otro registro*
- void **MUL** (uint32\_t \**Ra*, uint32\_t \**Rb*, uint32\_t \**Rc*, bool \**flg*, int \**pc*)  
*funcion que multiplica dos registros*

### 4.2.1. Documentación de las funciones

4.2.1.1. void ADD ( uint32\_t \* *Ra*, uint32\_t *Rb*, uint32\_t *Rc*, bool *flg*[], int \* *pc* )

funcion que suma

## Parámetros

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

## Devuelve

no retorna nada

4.2.1.2. void AND ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, bool \* *flg*, int \* *pc* )

funcion de producto logico

## Parámetros

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

## Devuelve

No retorna nada

4.2.1.3. void EOR ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, bool \* *flg*, int \* *pc* )

funcion de or exclusiva

## Parámetros

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

## Devuelve

No retorna nada

4.2.1.4. void MOV ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, bool \* *flg*, int \* *pc* )

funcion que escribe un valor de un registro en otro registro

## Parámetros

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1

## Devuelve

No retorna nada

4.2.1.5. void MUL ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, bool \* *flg*, int \* *pc* )

funcion que multiplica dos registros

**Parámetros**

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

**Devuelve**

No retorna nada

4.2.1.6. void OR ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, bool \* *flg*, int \* *pc* )

funcion de suma logica

**Parámetros**

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

**Devuelve**

No retorna nada

4.2.1.7. void SUB ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, bool \* *flg*, int \* *pc* )

funcion que resta

**Parámetros**

<i>Ra</i>	guarda el resultado
<i>Rb</i>	operando 1
<i>Rc</i>	operando 2

**Devuelve**

No retorna nada

## 4.3. Referencia del Archivo banderas.c

```
#include "banderas.h"
```

**Funciones**

- void [banderas](#) (uint32\_t *Ra*, uint32\_t *Rb*, uint32\_t *Rc*, bool *flg*[ ])   
*funcion que registran los estados del microprocesador*

### 4.3.1. Documentación de las funciones

4.3.1.1. void [banderas](#) ( uint32\_t *Ra*, uint32\_t *Rb*, uint32\_t *Rc*, bool *flg*[ ] )

funcion que registran los estados del microprocesador

## Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2
<i>Rc</i>	operando 3

## Devuelve

no retorna nada

## 4.4. Referencia del Archivo banderas.h

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <curses.h>
```

## Funciones

- void [banderas](#) (uint32\_t Ra, uint32\_t Rb, uint32\_t Rc, bool flg[])  
*funcion que registran los estados del microprocesador*

### 4.4.1. Documentación de las funciones

4.4.1.1. void [banderas](#) ( uint32\_t Ra, uint32\_t Rb, uint32\_t Rc, bool flg[ ] )

funcion que registran los estados del microprocesador

## Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2
<i>Rc</i>	operando 3

## Devuelve

no retorna nada

## 4.5. Referencia del Archivo decoder.c

```
#include "decoder.h"
#include "alu.h"
#include "banderas.h"
#include "InstruccionesDesplazamiento.h"
#include <string.h>
#include "RAM.h"
```

## Funciones

- void [decodeInstruction](#) (instruction\_t instruction, uint32\_t \*Rd, uint32\_t \*Rm, uint32\_t \*Rr, bool flg[ ], int \*pc, uint8\_t pila)

- `instruction_t getInstruction` (char \*instStr)
- int `readFile` (char \*filename, `ins_t` \*instructions)
- int `countLines` (FILE \*fp)

#### 4.5.1. Documentación de las funciones

4.5.1.1. `int countLines ( FILE * fp )`

4.5.1.2. `void decodeInstruction ( instruction_t instruction, uint32_t * Rd, uint32_t * Rm, uint32_t * Rr, bool flg[], int * pc, uint8_t pila )`

4.5.1.3. `instruction_t getInstruction ( char * instStr )`

4.5.1.4. `int readFile ( char * filename, ins_t * instructions )`

### 4.6. Referencia del Archivo decoder.h

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <curses.h>
```

#### Estructuras de datos

- struct `ins_t`
- struct `instruction_t`

#### Funciones

- int `readFile` (char \*filename, `ins_t` \*instructions)
- int `countLines` (FILE \*fp)

#### 4.6.1. Documentación de las funciones

4.6.1.1. `int countLines ( FILE * fp )`

4.6.1.2. `int readFile ( char * filename, ins_t * instructions )`

### 4.7. Referencia del Archivo InstruccionesDesplazamiento.c

```
#include "InstruccionesDesplazamiento.h"
```

#### Funciones

- void `LSL` (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t inmediato, int \*pc)  
*funcion de desplazamiento logico a la izquierda*
- void `NOP` (int \*pc)  
*funcion que no hace ninguna operacion solo aumenta el valor del pc*

- void **LSR** (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t inmediato, int \*pc)  
*funcion de desplazamiento logico a la derecha*
- void **BIC** (uint32\_t \*Ra, uint32\_t \*Rb, int \*pc)  
*funcion que realiza una AND entre un registro y el complemento de otro*
- void **MVN** (uint32\_t \*Ra, uint32\_t \*Rb, int \*pc)  
*funcion que gua\*Ra el complemento de un numero*
- void **RSBS** (uint32\_t \*Ra, uint32\_t \*Rb, int \*pc)  
*funcion que obtiene el complemento a dos de un numero*
- void **ASRS** (uint32\_t \*Ra, uint32\_t \*Rb, int \*pc)  
*funcion de desplazamiento aritmetico a la derecha*
- void **ROR** (uint32\_t \*Ra, uint32\_t \*Rb, int \*pc)  
*funcion de rotacion a la derecha*
- void **CMP** (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc, int \*pc)  
*funcion de resta que solo modifica las banderas*
- void **CMN** (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc, int \*pc)  
*funcion de suma que solo modifica las banderas*

#### 4.7.1. Documentación de las funciones

##### 4.7.1.1. void ASRS ( uint32\_t \* Ra, uint32\_t \* Rb, int \* pc )

funcion de desplazamiento aritmetico a la derecha

Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2

Devuelve

no retorna nada

##### 4.7.1.2. void BIC ( uint32\_t \* Ra, uint32\_t \* Rb, int \* pc )

funcion que realiza una AND entre un registro y el complemento de otro

Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2

Devuelve

no retorna nada

##### 4.7.1.3. void CMN ( uint32\_t \* Ra, uint32\_t \* Rb, uint32\_t \* Rc, int \* pc )

funcion de suma que solo modifica las banderas

**Parámetros**

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2
<i>Rc</i>	operando 3
<i>pc</i>	operando 4

**Devuelve**

no retorna nada

4.7.1.4. void CMP ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, int \* *pc* )

funcion de resta que solo modifica las banderas

**Parámetros**

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2
<i>Rc</i>	operando 3
<i>pc</i>	operando 4

**Devuelve**

no retorna nada

4.7.1.5. void LSL ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t *inmediato*, int \* *pc* )

funcion de desplazamiento logico a la izquierda

**Parámetros**

<i>Ra</i>	operando 1, ademas guarda el resultado
<i>Rb</i>	operando 2
<i>inmediato</i>	operando 3

**Devuelve**

no retorna nada

4.7.1.6. void LSR ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t *inmediato*, int \* *pc* )

funcion de desplazamiento logico a la derecha

**Parámetros**

<i>Ra</i>	operando1, ademas guarda el resultado
<i>Rb</i>	operando 2
<i>inmediato</i>	operando 3

**Devuelve**

no retorna nada

4.7.1.7. void MVN ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, int \* *pc* )

funcion que gua\*Raa el complemento de un numero



## Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2

## Devuelve

no retorna nada

4.7.1.8. void NOP ( int \* *pc* )

funcion que no hace ninguna operacion solo aumenta el valor del pc

4.7.1.9. void ROR ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, int \* *pc* )

funcion de rotacion a la derecha

## Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2

## Devuelve

no retorna nada

4.7.1.10. void RSBS ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, int \* *pc* )

funcion que obtiene el complemento a dos de un numero

## Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2

## Devuelve

no retorna nada

## 4.8. Referencia del Archivo InstruccionesDesplazamiento.h

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
```

## Funciones

- void **LSL** (uint32\_t \**Ra*, uint32\_t \**Rb*, uint32\_t inmediato, int \**pc*)  
*funcion de desplazamiento logico a la izquierda*
- void **LSR** (uint32\_t \**Ra*, uint32\_t \**Rb*, uint32\_t inmediato, int \**pc*)  
*funcion de desplazamiento logico a la derecha*
- void **BIC** (uint32\_t \**Ra*, uint32\_t \**Rb*, int \**pc*)

- funcion que realiza una AND entre un registro y el complemento de otro*

  - void **MVN** (uint32\_t \*Ra, uint32\_t \*Rb, int \*pc)
- funcion que gura el complemento de un numero*

  - void **RSBS** (uint32\_t \*Ra, uint32\_t \*Rb, int \*pc)
- funcion que obtiene el complemento a dos de un numero*

  - void **NOP** (int \*pc)
- funcion que no hace ninguna operacion solo aumenta el valor del pc*

  - void **ASRS** (uint32\_t \*Ra, uint32\_t \*Rb, int \*pc)
- funcion de desplazamiento aritmetico a la derecha*

  - void **ROR** (uint32\_t \*Ra, uint32\_t \*Rb, int \*pc)
- funcion de rotacion a la derecha*

  - void **CMP** (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc, int \*pc)
- funcion de resta que solo modifica las banderas*

  - void **CMN** (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc, int \*pc)
- funcion de suma que solo modifica las banderas*

#### 4.8.1. Documentación de las funciones

##### 4.8.1.1. void ASRS ( uint32\_t \* Ra, uint32\_t \* Rb, int \* pc )

funcion de desplazamiento aritmetico a la derecha

Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2

Devuelve

no retorna nada

##### 4.8.1.2. void BIC ( uint32\_t \* Ra, uint32\_t \* Rb, int \* pc )

funcion que realiza una AND entre un registro y el complemento de otro

Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2

Devuelve

no retorna nada

##### 4.8.1.3. void CMN ( uint32\_t \* Ra, uint32\_t \* Rb, uint32\_t \* Rc, int \* pc )

funcion de suma que solo modifica las banderas

Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2
<i>Rc</i>	operando 3
<i>pc</i>	operando 4

Devuelve

no retorna nada

4.8.1.4. void CMP ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t \* *Rc*, int \* *pc* )

funcion de resta que solo modifica las banderas

Parámetros

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2
<i>Rc</i>	operando 3
<i>pc</i>	operando 4

Devuelve

no retorna nada

4.8.1.5. void LSL ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t *inmediato*, int \* *pc* )

funcion de desplazamiento logico a la izquierda

Parámetros

<i>Ra</i>	operando 1, ademas guarda el resultado
<i>Rb</i>	operando 2
<i>inmediato</i>	operando 3

Devuelve

no retorna nada

4.8.1.6. void LSR ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, uint32\_t *inmediato*, int \* *pc* )

funcion de desplazamiento logico a la derecha

Parámetros

<i>Ra</i>	operando1, ademas guarda el resultado
<i>Rb</i>	operando 2
<i>inmediato</i>	operando 3

Devuelve

no retorna nada

4.8.1.7. void MVN ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, int \* *pc* )

funcion que gua\*Ra el complemento de un numero

**Parámetros**

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2

**Devuelve**

no retorna nada

**4.8.1.8. void NOP ( int \* *pc* )**

funcion que no hace ninguna operacion solo aumenta el valor del pc

**4.8.1.9. void ROR ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, int \* *pc* )**

funcion de rotacion a la derecha

**Parámetros**

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2

**Devuelve**

no retorna nada

**4.8.1.10. void RSBS ( uint32\_t \* *Ra*, uint32\_t \* *Rb*, int \* *pc* )**

funcion que obtiene el complemento a dos de un numero

**Parámetros**

<i>Ra</i>	operando 1
<i>Rb</i>	operando 2

**Devuelve**

no retorna nada

**4.9. Referencia del Archivo interrupciones.c****4.10. Referencia del Archivo interrupciones.h**

```
#include <stdint.h>
```

**4.11. Referencia del Archivo io.c**

```
#include "io.h"
```

## Funciones

- void `initIO` (void)
- void `changePinPortA` (uint8\_t pin, uint8\_t value)
- void `changePinPortB` (uint8\_t pin, uint8\_t value)
- void `IOAccess` (uint8\_t address, uint8\_t \*data, uint8\_t r\_w)
- void `showPorts` (void)
- void `showFrame` (int x, int y, int w, int h)

## Variables

- `port_t` `PORTA`
- `port_t` `PORTB`
- `uint8_t` `irq` [16]

### 4.11.1. Documentación de las funciones

4.11.1.1. void `changePinPortA` ( uint8\_t *pin*, uint8\_t *value* )

4.11.1.2. void `changePinPortB` ( uint8\_t *pin*, uint8\_t *value* )

4.11.1.3. void `initIO` ( void )

4.11.1.4. void `IOAccess` ( uint8\_t *address*, uint8\_t \* *data*, uint8\_t *r\_w* )

4.11.1.5. void `showFrame` ( int *x*, int *y*, int *w*, int *h* )

4.11.1.6. void `showPorts` ( void )

### 4.11.2. Documentación de las variables

4.11.2.1. `uint8_t` `irq`[16]

4.11.2.2. `port_t` `PORTA`

4.11.2.3. `port_t` `PORTB`

## 4.12. Referencia del Archivo io.h

```
#include <stdint.h>
#include <curses.h>
```

## Estructuras de datos

- struct `port_t`

## 'defines'

- #define `XINIT` 10
- #define `YINIT` 5
- #define `HIGH` 1
- #define `LOW` 0

- #define Read 1
- #define Write 0
- #define BLUEBLACK 10 /\*Text Blue Background Black\*/
- #define REDBLACK 20 /\*Text Red Background Black\*/
- #define WHITEBLACK 30 /\*Text White Background White\*/

## Funciones

- void IOAccess (uint8\_t address, uint8\_t \*data, uint8\_t r\_w)
- void changePinPortA (uint8\_t pin, uint8\_t value)
- void changePinPortB (uint8\_t pin, uint8\_t value)
- void initIO (void)
- void showPorts (void)
- void showFrame (int x, int y, int w, int h)

### 4.12.1. Documentación de los 'defines'

4.12.1.1. #define BLUEBLACK 10 /\*Text Blue Background Black\*/

4.12.1.2. #define HIGH 1

4.12.1.3. #define LOW 0

4.12.1.4. #define Read 1

4.12.1.5. #define REDBLACK 20 /\*Text Red Background Black\*/

4.12.1.6. #define WHITEBLACK 30 /\*Text White Background White\*/

4.12.1.7. #define Write 0

4.12.1.8. #define XINIT 10

4.12.1.9. #define YINIT 5

### 4.12.2. Documentación de las funciones

4.12.2.1. void changePinPortA ( uint8\_t *pin*, uint8\_t *value* )

4.12.2.2. void changePinPortB ( uint8\_t *pin*, uint8\_t *value* )

4.12.2.3. void initIO ( void )

4.12.2.4. void IOAccess ( uint8\_t *address*, uint8\_t \* *data*, uint8\_t *r\_w* )

4.12.2.5. void showFrame ( int *x*, int *y*, int *w*, int *h* )

4.12.2.6. void showPorts ( void )

## 4.13. Referencia del Archivo Load\_Store.c

```
#include "Load_Store.h"
#include "ram.h"
```

## Funciones

- void **LDR** (uint32\_t \*Rt, uint32\_t Rn, uint32\_t inmed, int \*pc)
- void **LDRB** (uint32\_t \*Rt, uint32\_t Rn, uint32\_t inmed, int \*pc)
- void **LDRH** (uint32\_t \*Rt, uint32\_t Rn, uint32\_t inmed, int \*pc)
- void **LDRSB** (uint32\_t \*Rt, uint32\_t Rn, uint32\_t inmed, int \*pc)
- void **LDRSH** (uint32\_t \*Rt, uint32\_t Rn, uint32\_t inmed, int \*pc)
- void **STR** (uint32\_t \*Rt, uint32\_t Rn, uint32\_t inmed, uint32\_t SP, int \*pc)
- void **STRB** (uint32\_t \*Rt, uint32\_t Rn, uint32\_t inmed, uint32\_t SP, int \*pc)
- void **STRH** (uint32\_t \*Rt, uint32\_t Rn, uint32\_t inmed, uint32\_t SP, int \*pc)

### 4.13.1. Documentación de las funciones

4.13.1.1. void LDR ( uint32\_t \* *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, int \* *pc* )

4.13.1.2. void LDRB ( uint32\_t \* *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, int \* *pc* )

4.13.1.3. void LDRH ( uint32\_t \* *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, int \* *pc* )

4.13.1.4. void LDRSB ( uint32\_t \* *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, int \* *pc* )

4.13.1.5. void LDRSH ( uint32\_t \* *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, int \* *pc* )

4.13.1.6. void STR ( uint32\_t \* *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, uint32\_t *SP*, int \* *pc* )

4.13.1.7. void STRB ( uint32\_t \* *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, uint32\_t *SP*, int \* *pc* )

4.13.1.8. void STRH ( uint32\_t \* *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, uint32\_t *SP*, int \* *pc* )

## 4.14. Referencia del Archivo Load\_Store.h

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
```

## Funciones

- void **LDR** (uint32\_t Rt, uint32\_t Rn, uint32\_t inmed)  
*funcion de Carga*
- void **LDRB** (uint32\_t Rt, uint32\_t Rn, uint32\_t inmed)  
*funcion de Carga con extension de cero*
- void **LDRH** (uint32\_t Rt, uint32\_t Rn, uint32\_t inmed)  
*funcion de Carga con extension de cero*
- void **LDRSB** (uint32\_t Rt, uint32\_t Rn, uint32\_t inmed)  
*funcion de Carga con extension de signo*
- void **LDRSH** (uint32\_t Rt, uint32\_t Rn, uint32\_t inmed)  
*funcion de Carga con extension de signo*
- void **STR** (uint32\_t Rt, uint32\_t Rn, uint32\_t inmed, uint32\_t SP)  
*funcion de almacenamiento*
- void **STRB** (uint32\_t Rt, uint32\_t Rn, uint32\_t inmed, uint32\_t SP)  
*funcion de almacenamiento*
- void **STRH** (uint32\_t Rt, uint32\_t Rn, uint32\_t inmed, uint32\_t SP)  
*funcion de almacenamiento*

#### 4.14.1. Documentación de las funciones

##### 4.14.1.1. void LDR ( uint32\_t *Rt*, uint32\_t *Rn*, uint32\_t *inmed* )

funcion de Carga

Parámetros

<i>Rt</i>	guarda el resultado
<i>Rn</i>	operando 1
<i>inmed</i>	operando 2

Devuelve

no retorna nada

##### 4.14.1.2. void LDRB ( uint32\_t *Rt*, uint32\_t *Rn*, uint32\_t *inmed* )

funcion de Carga con extension de cero

Parámetros

<i>Rt</i>	guarda el resultado
<i>Rn</i>	operando 1
<i>inmed</i>	operando 2

Devuelve

no retorna nada

##### 4.14.1.3. void LDRH ( uint32\_t *Rt*, uint32\_t *Rn*, uint32\_t *inmed* )

funcion de Carga con extension de cero

Parámetros

<i>Rt</i>	guarda el resultado
<i>Rn</i>	operando 1
<i>inmed</i>	operando 2

Devuelve

no retorna nada

##### 4.14.1.4. void LDRSB ( uint32\_t *Rt*, uint32\_t *Rn*, uint32\_t *inmed* )

funcion de Carga con extension de signo

Parámetros

<i>Rt</i>	guarda el resultado
<i>Rn</i>	operando 1



<i>inmed</i>	operando 2
--------------	------------

Devuelve

no retorna nada

4.14.1.5. void LDRSH ( uint32\_t *Rt*, uint32\_t *Rn*, uint32\_t *inmed* )

funcion de Carga con extension de signo

Parámetros

<i>Rt</i>	guarda el resultado
<i>Rn</i>	operando 1
<i>inmed</i>	operando 2

Devuelve

no retorna nada

4.14.1.6. void STR ( uint32\_t *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, uint32\_t *SP* )

funcion de almacenamiento

Parámetros

<i>Rt</i>	operando 1
<i>Rn</i>	operando 2
<i>inmed</i>	operando 3
<i>SP</i>	operando 4

Devuelve

no retorna nada

4.14.1.7. void STRB ( uint32\_t *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, uint32\_t *SP* )

funcion de almacenamiento

Parámetros

<i>Rt</i>	operando 1
<i>Rn</i>	operando 2
<i>inmed</i>	operando 3
<i>SP</i>	operando 4

Devuelve

no retorna nada

4.14.1.8. void STRH ( uint32\_t *Rt*, uint32\_t *Rn*, uint32\_t *inmed*, uint32\_t *SP* )

funcion de almacenamiento

**Parámetros**

<i>Rt</i>	operando 1
<i>Rn</i>	operando 2
<i>inmed</i>	operando 3
<i>SP</i>	operando 4

**Devuelve**

no retorna nada

## 4.15. Referencia del Archivo main.c

```
#include "registros.h"  
#include "decoder.h"  
#include <curses.h>
```

**Funciones**

- int [main](#) (void)

### 4.15.1. Documentación de las funciones

4.15.1.1. int main ( void )

## 4.16. Referencia del Archivo ports.c

```
#include "io.h"
```

**Funciones**

- int [main](#) (void)

**Variables**

- uint8\_t [irq](#) [16]

### 4.16.1. Documentación de las funciones

4.16.1.1. int main ( void )

### 4.16.2. Documentación de las variables

4.16.2.1. uint8\_t irq[16]

## 4.17. Referencia del Archivo RAM.c

```
#include <stdint.h>
#include "RAM.h"
#include <stdio.h>
```

### Funciones

- uint8\_t **bitcount** (uint8\_t listaregistros[])
- void **push** (uint8\_t \*pila, uint32\_t \*R, uint8\_t \*ram, uint8\_t listaregistros[])
- void **pop** (uint8\_t \*pila, uint32\_t \*R, uint8\_t \*ram, uint8\_t listaregistros[])

#### 4.17.1. Documentación de las funciones

4.17.1.1. uint8\_t bitcount ( uint8\_t *listaregistros*[] )

4.17.1.2. void pop ( uint8\_t \* *pila*, uint32\_t \* *R*, uint8\_t \* *ram*, uint8\_t *listaregistros*[] )

4.17.1.3. void push ( uint8\_t \* *pila*, uint32\_t \* *R*, uint8\_t \* *ram*, uint8\_t *listaregistros*[] )

## 4.18. Referencia del Archivo RAM.h

```
#include <stdint.h>
```

### 'defines'

- #define **Mema** 64

### Funciones

- void **push** (uint8\_t \*pila, uint32\_t \*R, uint8\_t \*ram, uint8\_t listaregistros[])
- void **pop** (uint8\_t \*pila, uint32\_t \*R, uint8\_t \*ram, uint8\_t listaregistros[])
- uint8\_t **bitcount** (uint8\_t listaregistros[])

#### 4.18.1. Documentación de los 'defines'

4.18.1.1. #define Mema 64

#### 4.18.2. Documentación de las funciones

4.18.2.1. uint8\_t bitcount ( uint8\_t *listaregistros*[] )

4.18.2.2. void pop ( uint8\_t \* *pila*, uint32\_t \* *R*, uint8\_t \* *ram*, uint8\_t *listaregistros*[] )

4.18.2.3. void push ( uint8\_t \* *pila*, uint32\_t \* *R*, uint8\_t \* *ram*, uint8\_t *listaregistros*[] )

## 4.19. Referencia del Archivo registros.c

```
#include "registros.h"
```

## Funciones

- void [Registros](#) (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc)  
*funcion que muestra los registros*

### 4.19.1. Documentación de las funciones

4.19.1.1. void Registros ( uint32\_t \* Ra, uint32\_t \* Rb, uint32\_t \* Rc )

funcion que muestra los registros

#### Parámetros

<i>Rd</i>	operando 1
<i>Rm</i>	operando 2
<i>Rr</i>	operando 3

#### Devuelve

no retorna nada

## 4.20. Referencia del Archivo registros.h

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
```

## Funciones

- void [Registros](#) (uint32\_t \*Ra, uint32\_t \*Rb, uint32\_t \*Rc)  
*funcion que muestra los registros*

### 4.20.1. Documentación de las funciones

4.20.1.1. void Registros ( uint32\_t \* Ra, uint32\_t \* Rb, uint32\_t \* Rc )

funcion que muestra los registros

#### Parámetros

<i>Rd</i>	operando 1
<i>Rm</i>	operando 2
<i>Rr</i>	operando 3

#### Devuelve

no retorna nada

## 4.21. Referencia del Archivo saltos.c

```
#include "banderas.h"
#include "alu.h"
#include <curses.h>
```

## Funciones

- void **B** (uint32\_t valor, int \*pc, bool \*flg)

## Variables

- uint32\_t **AUX**

### 4.21.1. Documentación de las funciones

4.21.1.1. void **B** ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )

### 4.21.2. Documentación de las variables

4.21.2.1. uint32\_t **AUX**

## 4.22. Referencia del Archivo saltos.h

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
```

## Funciones

- void **B** (uint32\_t valor, int \*pc, bool \*flg)
- void **BEQ** (uint32\_t valor, int \*pc, bool \*flg)
- void **BNE** (uint32\_t valor, int \*pc, bool \*flg)
- void **BCS** (uint32\_t valor, int \*pc, bool \*flg)
- void **BCC** (uint32\_t valor, int \*pc, bool \*flg)
- void **BMI** (uint32\_t valor, int \*pc, bool \*flg)
- void **BPL** (uint32\_t valor, int \*pc, bool \*flg)
- void **BVS** (uint32\_t valor, int \*pc, bool \*flg)
- void **BVC** (uint32\_t valor, int \*pc, bool \*flg)
- void **BHI** (uint32\_t valor, int \*pc, bool \*flg)
- void **BLS** (uint32\_t valor, int \*pc, bool \*flg)
- void **BGE** (uint32\_t valor, int \*pc, bool \*flg)
- void **BLT** (uint32\_t valor, int \*pc, bool \*flg)
- void **BGT** (uint32\_t valor, int \*pc, bool \*flg)
- void **BLE** (uint32\_t valor, int \*pc, bool \*flg)
- void **BAL** (uint32\_t valor, int \*pc, bool \*flg)
- void **BL** (uint32\_t valor, int \*pc, bool \*flg)
- void **BX** (uint32\_t \*pc, bool \*flg)

### 4.22.1. Documentación de las funciones

4.22.1.1. void **B** ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )

4.22.1.2. void **BAL** ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )

4.22.1.3. void **BCC** ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )

- 4.22.1.4. void BCS ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.5. void BEQ ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.6. void BGE ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.7. void BGT ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.8. void BHI ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.9. void BL ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.10. void BLE ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.11. void BLS ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.12. void BLT ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.13. void BMI ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.14. void BNE ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.15. void BPL ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.16. void BVC ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.17. void BVS ( uint32\_t *valor*, int \* *pc*, bool \* *flg* )
- 4.22.1.18. void BX ( uint32\_t \* *pc*, bool \* *flg* )

## 4.23. Referencia del Archivo test.c