

# My Project

Generated by Doxygen 1.8.10

Sat Sep 19 2015 00:46:04



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Data Structure Documentation</b>	<b>5</b>
3.1	ins_t Struct Reference . . . . .	5
3.1.1	Field Documentation . . . . .	5
3.1.1.1	array . . . . .	5
3.2	instruction_t Struct Reference . . . . .	5
3.2.1	Field Documentation . . . . .	5
3.2.1.1	mnemonic . . . . .	5
3.2.1.2	op1_type . . . . .	5
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
<b>4</b>	<b>File Documentation</b>	<b>7</b>
4.1	ALU.c File Reference . . . . .	7
4.1.1	Function Documentation . . . . .	8
4.1.1.1	ACTNZ(uint32_t *Rd, uint32_t *banderas) . . . . .	8
4.1.1.2	ADCS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas) . . . . .	8
4.1.1.3	ADD(uint32_t *Rd, uint32_t Rm, uint32_t Rn) . . . . .	8
4.1.1.4	ADDS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas) . . . . .	9
4.1.1.5	ANDS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas) . . . . .	9
4.1.1.6	BICS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas) . . . . .	9
4.1.1.7	CMN(uint32_t Rm, uint32_t Rn, uint32_t *banderas) . . . . .	10
4.1.1.8	CMP(uint32_t Rm, uint32_t Rn, uint32_t *banderas) . . . . .	11
4.1.1.9	EORS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas) . . . . .	11

4.1.1.10	MOV(uint32_t *Rm, uint32_t Rn)	11
4.1.1.11	MOVS(uint32_t *Rm, uint32_t Rn, uint32_t *banderas)	11
4.1.1.12	MULS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	12
4.1.1.13	MVNS(uint32_t *Rd, uint32_t Rm, uint32_t *banderas)	12
4.1.1.14	NOP()	12
4.1.1.15	ORRS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	12
4.1.1.16	RSBS(uint32_t *Rd, uint32_t Rm, uint32_t *banderas)	13
4.1.1.17	SBCS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	13
4.1.1.18	SUB(uint32_t *Rd, uint32_t Rm, uint32_t Rn)	13
4.1.1.19	SUBS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	13
4.1.1.20	TST(uint32_t Rm, uint32_t Rn, uint32_t *banderas)	14
4.2	ALU.h File Reference	14
4.2.1	Function Documentation	15
4.2.1.1	ACTNZ(uint32_t *Rd, uint32_t *banderas)	15
4.2.1.2	ADCS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	15
4.2.1.3	ADD(uint32_t *Rd, uint32_t Rm, uint32_t Rn)	15
4.2.1.4	ADDS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	16
4.2.1.5	ANDS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	16
4.2.1.6	BICS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	16
4.2.1.7	CMN(uint32_t Rm, uint32_t Rn, uint32_t *banderas)	17
4.2.1.8	CMP(uint32_t Rm, uint32_t Rn, uint32_t *banderas)	18
4.2.1.9	EORS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	18
4.2.1.10	MOV(uint32_t *Rm, uint32_t Rn)	18
4.2.1.11	MOVS(uint32_t *Rm, uint32_t Rn, uint32_t *banderas)	18
4.2.1.12	MULS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	19
4.2.1.13	MVNS(uint32_t *Rd, uint32_t Rm, uint32_t *banderas)	19
4.2.1.14	NOP()	19
4.2.1.15	ORRS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	19
4.2.1.16	RSBS(uint32_t *Rd, uint32_t Rm, uint32_t *banderas)	20
4.2.1.17	SBCS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	20
4.2.1.18	SUB(uint32_t *Rd, uint32_t Rm, uint32_t Rn)	20
4.2.1.19	SUBS(uint32_t *Rd, uint32_t Rm, uint32_t Rn, uint32_t *banderas)	20
4.2.1.20	TST(uint32_t Rm, uint32_t Rn, uint32_t *banderas)	21
4.3	branch.c File Reference	21
4.3.1	Function Documentation	22
4.3.1.1	B(uint32_t *R, uint32_t b)	22
4.3.1.2	BAL(uint32_t *R, uint32_t b, uint32_t *banderas)	22
4.3.1.3	BCC(uint32_t *R, uint32_t b, uint32_t *banderas)	22
4.3.1.4	BCS(uint32_t *R, uint32_t b, uint32_t *banderas)	23
4.3.1.5	BEQ(uint32_t *R, uint32_t b, uint32_t *banderas)	23

4.3.1.6	BGE(uint32_t *R, uint32_t b, uint32_t *banderas)	23
4.3.1.7	BGT(uint32_t *R, uint32_t b, uint32_t *banderas)	23
4.3.1.8	BHI(uint32_t *R, uint32_t b, uint32_t *banderas)	24
4.3.1.9	BL(uint32_t *LR, uint32_t *PC, uint32_t salto)	24
4.3.1.10	BLE(uint32_t *R, uint32_t b, uint32_t *banderas)	24
4.3.1.11	BLS(uint32_t *R, uint32_t b, uint32_t *banderas)	24
4.3.1.12	BLT(uint32_t *R, uint32_t b, uint32_t *banderas)	25
4.3.1.13	BLX(uint32_t *LR, uint32_t *PC, uint32_t direccion)	25
4.3.1.14	BMI(uint32_t *R, uint32_t b, uint32_t *banderas)	25
4.3.1.15	BNE(uint32_t *R, uint32_t b, uint32_t *banderas)	25
4.3.1.16	BPL(uint32_t *R, uint32_t b, uint32_t *banderas)	26
4.3.1.17	BVC(uint32_t *R, uint32_t b, uint32_t *banderas)	26
4.3.1.18	BVS(uint32_t *R, uint32_t b, uint32_t *banderas)	26
4.3.1.19	BX(uint32_t *PC, uint32_t direccion)	26
4.4	branch.h File Reference	27
4.4.1	Function Documentation	28
4.4.1.1	B(uint32_t *R, uint32_t b)	28
4.4.1.2	BAL(uint32_t *R, uint32_t b, uint32_t *banderas)	29
4.4.1.3	BCC(uint32_t *R, uint32_t b, uint32_t *banderas)	29
4.4.1.4	BCS(uint32_t *R, uint32_t b, uint32_t *banderas)	29
4.4.1.5	BEQ(uint32_t *R, uint32_t b, uint32_t *banderas)	29
4.4.1.6	BGE(uint32_t *R, uint32_t b, uint32_t *banderas)	30
4.4.1.7	BGT(uint32_t *R, uint32_t b, uint32_t *banderas)	30
4.4.1.8	BHI(uint32_t *R, uint32_t b, uint32_t *banderas)	30
4.4.1.9	BL(uint32_t *RL, uint32_t *PC, uint32_t salto)	30
4.4.1.10	BLE(uint32_t *R, uint32_t b, uint32_t *banderas)	31
4.4.1.11	BLS(uint32_t *R, uint32_t b, uint32_t *banderas)	31
4.4.1.12	BLT(uint32_t *R, uint32_t b, uint32_t *banderas)	31
4.4.1.13	BLX(uint32_t *RL, uint32_t *PC, uint32_t direccion)	31
4.4.1.14	BMI(uint32_t *R, uint32_t b, uint32_t *banderas)	32
4.4.1.15	BNE(uint32_t *R, uint32_t b, uint32_t *banderas)	32
4.4.1.16	BPL(uint32_t *R, uint32_t b, uint32_t *banderas)	32
4.4.1.17	BVC(uint32_t *R, uint32_t b, uint32_t *banderas)	32
4.4.1.18	BVS(uint32_t *R, uint32_t b, uint32_t *banderas)	33
4.4.1.19	BX(uint32_t *PC, uint32_t direccion)	33
4.5	decoder.c File Reference	33
4.5.1	Function Documentation	34
4.5.1.1	countLines(FILE *fp)	34
4.5.1.2	decodeInstruction(instruction_t instruction, uint32_t *Reg, uint32_t *banderas)	34
4.5.1.3	getInstruction(char *instStr)	34

4.5.1.4	readFile(char *filename, ins_t *instructions)	34
4.6	decoder.h File Reference	34
4.6.1	Function Documentation	34
4.6.1.1	countLines(FILE *fp)	34
4.6.1.2	decodeInstruction(instruction_t instruction, uint32_t *Reg, uint32_t *banderas)	34
4.6.1.3	getInstruction(char *instStr)	34
4.6.1.4	readFile(char *filename, ins_t *instructions)	35
4.7	desplazamiento.c File Reference	35
4.7.1	Function Documentation	35
4.7.1.1	ASRS(uint32_t *Rd, uint32_t Rm, uint32_t num, uint32_t *banderas)	35
4.7.1.2	LSLS(uint32_t *Rd, uint32_t Rm, uint32_t num, uint32_t *banderas)	36
4.7.1.3	LSRS(uint32_t *Rd, uint32_t Rm, uint32_t num, uint32_t *banderas)	37
4.7.1.4	REV(uint32_t *Rd, uint32_t Rm)	37
4.7.1.5	REV16(uint32_t *Rd, uint32_t Rm)	37
4.7.1.6	REVSH(uint32_t *Rd, uint32_t Rm)	37
4.7.1.7	RORS(uint32_t *Rd, uint32_t Rm, uint32_t num, uint32_t *banderas)	38
4.8	desplazamiento.h File Reference	38
4.8.1	Function Documentation	38
4.8.1.1	ASRS(uint32_t *Rd, uint32_t Rm, uint32_t num, uint32_t *banderas)	38
4.8.1.2	LSLS(uint32_t *Rd, uint32_t Rm, uint32_t num, uint32_t *banderas)	39
4.8.1.3	LSRS(uint32_t *Rd, uint32_t Rm, uint32_t num, uint32_t *banderas)	39
4.8.1.4	REV(uint32_t *Rd, uint32_t Rm)	39
4.8.1.5	REV16(uint32_t *Rd, uint32_t Rm)	39
4.8.1.6	REVSH(uint32_t *Rd, uint32_t Rm)	40
4.8.1.7	RORS(uint32_t *Rd, uint32_t Rm, uint32_t num, uint32_t *banderas)	40
4.9	main.c File Reference	40
4.9.1	Function Documentation	41
4.9.1.1	main(void)	41
4.10	registros.c File Reference	41
4.10.1	Function Documentation	41
4.10.1.1	mostrar_banderas(uint32_t banderas[4])	41
4.10.1.2	mostrar_operacion(char *op)	41
4.10.1.3	mostrar_registros(uint32_t *registro)	41
4.10.2	Variable Documentation	42
4.10.2.1	i	42
4.11	registros.h File Reference	42
4.11.1	Function Documentation	42
4.11.1.1	mostrar_banderas(uint32_t banderas[4])	42
4.11.1.2	mostrar_operacion(char *op)	42
4.11.1.3	mostrar_registros(uint32_t *registro)	42

---

4.12 test.c File Reference . . . . .	43
--------------------------------------	----





# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

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



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

ALU.c	7
ALU.h	14
branch.c	21
branch.h	27
decoder.c	33
decoder.h	34
desplazamiento.c	35
desplazamiento.h	38
main.c	40
registros.c	41
registros.h	42
test.c	43



## Chapter 3

# Data Structure Documentation

### 3.1 ins\_t Struct Reference

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

#### Data Fields

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

#### 3.1.1 Field Documentation

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

The documentation for this struct was generated from the following file:

- [decoder.h](#)

### 3.2 instruction\_t Struct Reference

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

#### Data Fields

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

#### 3.2.1 Field Documentation

##### 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

The documentation for this struct was generated from the following file:

- [decoder.h](#)

## Chapter 4

# File Documentation

### 4.1 ALU.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "ALU.h"
#include <string.h>
```

#### Functions

- void **ADDS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion que suma*
- void **ADD** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn)  
*funcion que suma pero no actualiza banderas*
- void **SUBS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion que resta y actualiza las banderas*
- void **SUB** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn)  
*funcion que resta*
- void **ANDS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion que producto logico a nivel de bit*
- void **ORRS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion OR*
- void **MOVS** (uint32\_t \*Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion que mueve datos de un registro a otro*
- void **MOV** (uint32\_t \*Rm, uint32\_t Rn)  
*funcion que mueve datos de un registro a otro pero solo lo hace para actualizar banderas, no guarda este resultado*
- void **BICS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion para ejecutar una AND entre un registro y el negado del otro*
- void **EORS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion para ejecutar una exclusive OR bit a bit con los dos registros de entrada*
- void **MVNS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t \*banderas)  
*funcion para ejecutar la operacion logica NOT a Rm*
- void **NOP** ()  
*funcion para no hacer nada*
- void **CMN** (uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion sumar pero no guarda el resultado*

- void **CMP** (uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion restar pero no guarda el valor solo actualiza banderas*
- void **TST** (uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion realiza la operacion AND pero no guarda el valor solo actualiza banderas*
- void **ACTNZ** (uint32\_t \*Rd, uint32\_t \*banderas)  
*funcion que actualiza las banderas N y Z*
- void **ADCS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion que efectua la suma normal y suma el carry*
- void **SBCS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion que efectua la resta normal y resta el carry*
- void **RSBS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t \*banderas)  
*funcion que efectua el complemento a dos de un registro*
- void **MULS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion que efectue el producto de dos registros sin signo*

#### 4.1.1 Function Documentation

##### 4.1.1.1 void ACTNZ ( uint32\_t \* Rd, uint32\_t \* banderas )

funcion que actualiza las banderas N y Z

###### Parameters

*Rd	resultado
*banderas	direccion de memoria de la bandera cero "N"
*op	se identifica la operacion

###### Returns

no hay retorno

##### 4.1.1.2 void ADCS ( uint32\_t \* Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \* banderas )

funcion que efectua la suma normal y suma el carry

###### Parameters

*Rd	registro donde se guarda el resultado
Rm	registro a sumar
Rn	registro a sumar
*banderas	direccion de memoria de la bandera cero "N"
*op	se identifica la operacion

###### Returns

no hay retorno

##### 4.1.1.3 void ADD ( uint32\_t \* Rd, uint32\_t Rm, uint32\_t Rn )

funcion que suma pero no actualiza banderas



## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado

## Returns

no hay retorno

4.1.1.4 void ADDS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que suma

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado
<i>*banderas</i>	

## Returns

no hay retorno

4.1.1.5 void ANDS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que producto logico a nivel de bit

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado
<i>*banderas</i>	direccion de memoria de la bandera cero "N"

## Returns

no hay retorno

4.1.1.6 void BICS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion para ejecutar una AND entre un registro y el negado del otro

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	Registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado.
<i>*banderas</i>	direccion de memoria de la bandera cero "N"
<i>*op</i>	se identifica la operacion

## Returns

no hay retorno

4.1.1.7 void CMN ( uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t\* *banderas* )

funcion sumar pero no guarda el resultado

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*banderas</i>	direccion de memoria de la bandera cero "N"

## Returns

no hay retorno

4.1.1.8 void CMP ( uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion restar pero no guarda el valor solo actualiza banderas

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*banderas</i>	

## Returns

no hay retorno

4.1.1.9 void EORS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion para ejecutar una exclusive OR bit a bit con los dos registros de entrada

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	Registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado.
<i>*banderas</i>	direccion de memoria de la bandera cero "N"

## Returns

no hay retorno

4.1.1.10 void MOV ( uint32\_t \* *Rm*, uint32\_t *Rn* )

funcion que mueve datos de un registro a otro pero solo lo hace para actualizar banderas, no guarda este resultado

## Parameters

<i>Rn</i>	Registro que se va a mover
<i>*Rm</i>	lugar donde se almacenaria el resultado.
<i>*banderas</i>	

## Returns

no hay retorno

4.1.1.11 void MOVS ( uint32\_t \* *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que mueve datos de un registro a otro

## Parameters

<i>Rn</i>	Registro de entrada
<i>*Rm</i>	lugar donde se almacenara el resultado.
<i>*banderas</i>	direccion de memoria de la bandera cero "N"

## Returns

no hay retorno

4.1.1.12 void MULS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que efectue el producto de dos registros sin signo

## Parameters

<i>*Rd</i>	registro donde se guarda el resultado
<i>Rm</i>	registro a multiplicar
<i>Rn</i>	registro a multiplicar
<i>*banderas</i>	direccion de memoria de las bandera

## Returns

no hay retorno

4.1.1.13 void MVNS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t \* *banderas* )

funcion para ejecutar la operacion logica NOT a Rm

## Parameters

<i>Rm</i>	Registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado.
<i>*op</i>	se identifica la operacion

## Returns

no hay retorno

4.1.1.14 void NOP ( )

funcion para no hacer nada

## Parameters

<i>no</i>	tiene parametros de entrada
-----------	-----------------------------

## Returns

no hay retorno

4.1.1.15 void ORRS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion OR

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	Registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado
<i>*banderas</i>	direccion de memoria de la bandera cero "N"

## Returns

no hay retorno

4.1.1.16 void RSBS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t \* *banderas* )

funcion que efectua el complemento a dos de un registro

## Parameters

<i>*Rd</i>	registro donde se guarda el resultado
<i>Rm</i>	registro al que se le aplicara el complemento
<i>*banderas</i>	direccion de memoria de las banderas

## Returns

no hay retorno

4.1.1.17 void SBCS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que efectua la resta normal y resta el carry

## Parameters

<i>*Rd</i>	registro donde se guarda el resultado
<i>Rm</i>	registro a de entrada
<i>Rn</i>	registro a restar con el de entrada
<i>*banderas</i>	direccion de memoria de las banderas

## Returns

no hay retorno

4.1.1.18 void SUB ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn* )

funcion que resta

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado

## Returns

no hay retorno

4.1.1.19 void SUBS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que resta y actualiza las banderas

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado
<i>*banderas</i>	

## Returns

no hay retorno

4.1.1.20 void TST ( uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion realiza la operacion AND pero no guarda el valor solo actualiza banderas

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*banderas</i>	direccion de memoria de las banderas

## Returns

no hay retorno

## 4.2 ALU.h File Reference

## Functions

- void **ADCS** (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \**banderas*)  
*funcion que efectua la suma normal y suma el carry*
- void **ADDS** (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \**banderas*)  
*funcion que suma*
- void **ADD** (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *Rn*)  
*funcion que suma pero no actualiza banderas*
- void **ANDS** (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \**banderas*)  
*funcion que producto logico a nivel de bit*
- void **BICS** (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \**banderas*)  
*funcion para ejecutar una AND entre un registro y el negado del otro*
- void **CMN** (uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \**banderas*)  
*funcion sumar pero no guarda el resultado*
- void **CMP** (uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \**banderas*)  
*funcion restar pero no guarda el valor solo actualiza banderas*
- void **EORS** (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \**banderas*)  
*funcion para ejecutar una exclusive OR bit a bit con los dos registros de entrada*
- void **MOV** (uint32\_t \**Rm*, uint32\_t *Rn*)  
*funcion que mueve datos de un registro a otro pero solo lo hace para actualizar banderas, no guarda este resultado*
- void **MOVS** (uint32\_t \**Rm*, uint32\_t *Rn*, uint32\_t \**banderas*)  
*funcion que mueve datos de un registro a otro*
- void **MULS** (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \**banderas*)  
*funcion que efectue el producto de dos registros sin signo*
- void **MVNS** (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t \**banderas*)  
*funcion para ejecutar la operacion logica NOT a *Rm**

- void **NOP** ()  
*funcion para no hacer nada*
- void **ORRS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion OR*
- void **RSBS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t \*banderas)  
*funcion que efectua el complemento a dos de un registro*
- void **SBCS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion que efectua la resta normal y resta el carry*
- void **SUBS** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion que resta y actualiza las banderas*
- void **SUB** (uint32\_t \*Rd, uint32\_t Rm, uint32\_t Rn)  
*funcion que resta*
- void **TST** (uint32\_t Rm, uint32\_t Rn, uint32\_t \*banderas)  
*funcion realiza la operacion AND pero no guarda el valor solo actualiza banderas*
- void **ACTNZ** (uint32\_t \*Rd, uint32\_t \*banderas)  
*funcion que actualiza las banderas N y Z*

## 4.2.1 Function Documentation

### 4.2.1.1 void ACTNZ ( uint32\_t \* Rd, uint32\_t \* banderas )

funcion que actualiza las banderas N y Z

#### Parameters

*Rd	resultado
*banderas	direccion de memoria de la bandera cero "N"
*op	se identifica la operacion

#### Returns

no hay retorno

### 4.2.1.2 void ADCS ( uint32\_t \* Rd, uint32\_t Rm, uint32\_t Rn, uint32\_t \* banderas )

funcion que efectua la suma normal y suma el carry

#### Parameters

*Rd	registro donde se guarda el resultado
Rm	registro a sumar
Rn	registro a sumar
*banderas	direccion de memoria de la bandera cero "N"
*op	se identifica la operacion

#### Returns

no hay retorno

### 4.2.1.3 void ADD ( uint32\_t \* Rd, uint32\_t Rm, uint32\_t Rn )

funcion que suma pero no actualiza banderas

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado

## Returns

no hay retorno

4.2.1.4 void ADDS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que suma

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado
<i>*banderas</i>	

## Returns

no hay retorno

4.2.1.5 void ANDS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que producto logico a nivel de bit

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado
<i>*banderas</i>	direccion de memoria de la bandera cero "N"

## Returns

no hay retorno

4.2.1.6 void BICS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion para ejecutar una AND entre un registro y el negado del otro

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	Registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado.
<i>*banderas</i>	direccion de memoria de la bandera cero "N"
<i>*op</i>	se identifica la operacion

## Returns

no hay retorno



4.2.1.7 void CMN ( uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t\* *banderas* )

funcion sumar pero no guarda el resultado

**Parameters**

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*banderas</i>	direccion de memoria de la bandera cero "N"

**Returns**

no hay retorno

4.2.1.8 void CMP ( uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion restar pero no guarda el valor solo actualiza banderas

**Parameters**

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*banderas</i>	

**Returns**

no hay retorno

4.2.1.9 void EORS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion para ejecutar una exclusive OR bit a bit con los dos registros de entrada

**Parameters**

<i>Rn</i>	Registro de entrada
<i>Rm</i>	Registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado.
<i>*banderas</i>	direccion de memoria de la bandera cero "N"

**Returns**

no hay retorno

4.2.1.10 void MOV ( uint32\_t \* *Rm*, uint32\_t *Rn* )

funcion que mueve datos de un registro a otro pero solo lo hace para actualizar banderas, no guarda este resultado

**Parameters**

<i>Rn</i>	Registro que se va a mover
<i>*Rm</i>	lugar donde se almacenaria el resultado.
<i>*banderas</i>	

**Returns**

no hay retorno

4.2.1.11 void MOVS ( uint32\_t \* *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que mueve datos de un registro a otro

## Parameters

<i>Rn</i>	Registro de entrada
* <i>Rm</i>	lugar donde se almacenara el resultado.
* <i>banderas</i>	direccion de memoria de la bandera cero "N"

## Returns

no hay retorno

4.2.1.12 void MULS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que efectue el producto de dos registros sin signo

## Parameters

* <i>Rd</i>	registro donde se guarda el resultado
<i>Rm</i>	registro a multiplicar
<i>Rn</i>	registro a multiplicar
* <i>banderas</i>	direccion de memoria de las bandera

## Returns

no hay retorno

4.2.1.13 void MVNS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t \* *banderas* )

funcion para ejecutar la operacion logica NOT a *Rm*

## Parameters

<i>Rm</i>	Registro de entrada
* <i>Rd</i>	lugar donde se almacenara el resultado.
* <i>op</i>	se identifica la operacion

## Returns

no hay retorno

## 4.2.1.14 void NOP ( )

funcion para no hacer nada

## Parameters

<i>no</i>	tiene parametros de entrada
-----------	-----------------------------

## Returns

no hay retorno

4.2.1.15 void ORRS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion OR

**Parameters**

<i>Rn</i>	Registro de entrada
<i>Rm</i>	Registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado
<i>*banderas</i>	direccion de memoria de la bandera cero "N"

**Returns**

no hay retorno

4.2.1.16 void RSBS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t \* *banderas* )

funcion que efectua el complemento a dos de un registro

**Parameters**

<i>*Rd</i>	registro donde se guarda el resultado
<i>Rm</i>	registro al que se le aplicara el complemento
<i>*banderas</i>	direccion de memoria de las banderas

**Returns**

no hay retorno

4.2.1.17 void SBCS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que efectua la resta normal y resta el carry

**Parameters**

<i>*Rd</i>	registro donde se guarda el resultado
<i>Rm</i>	registro a de entrada
<i>Rn</i>	registro a restar con el de entrada
<i>*banderas</i>	direccion de memoria de las banderas

**Returns**

no hay retorno

4.2.1.18 void SUB ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn* )

funcion que resta

**Parameters**

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado

**Returns**

no hay retorno

4.2.1.19 void SUBS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion que resta y actualiza las banderas

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*Rd</i>	lugar donde se almacenara el resultado
<i>*banderas</i>	

## Returns

no hay retorno

4.2.1.20 void TST ( uint32\_t *Rm*, uint32\_t *Rn*, uint32\_t \* *banderas* )

funcion realiza la operacion AND pero no guarda el valor solo actualiza banderas

## Parameters

<i>Rn</i>	Registro de entrada
<i>Rm</i>	registro de entrada
<i>*banderas</i>	direccion de memoria de las banderas

## Returns

no hay retorno

## 4.3 branch.c File Reference

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

## Functions

- void **B** (uint32\_t \**R*, uint32\_t *b*)  
*funcion que aumenta o disminuye b posiciones el pc*
- void **BEQ** (uint32\_t \**R*, uint32\_t *b*, uint32\_t \**banderas*)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera Z esta en uno*
- void **BNE** (uint32\_t \**R*, uint32\_t *b*, uint32\_t \**banderas*)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera Z esta en cero*
- void **BCS** (uint32\_t \**R*, uint32\_t *b*, uint32\_t \**banderas*)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en uno*
- void **BCC** (uint32\_t \**R*, uint32\_t *b*, uint32\_t \**banderas*)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en cero*
- void **BMI** (uint32\_t \**R*, uint32\_t *b*, uint32\_t \**banderas*)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera N esta en uno*
- void **BPL** (uint32\_t \**R*, uint32\_t *b*, uint32\_t \**banderas*)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera N esta en cero*
- void **BVS** (uint32\_t \**R*, uint32\_t *b*, uint32\_t \**banderas*)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera V esta en uno*
- void **BVC** (uint32\_t \**R*, uint32\_t *b*, uint32\_t \**banderas*)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera V esta en cero*

- void **BHI** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en uno y la Z en cero*
- void **BLS** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en cero o la Z en uno*
- void **BGE** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera N y V son iguales*
- void **BLT** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera N y V son diferentes*
- void **BGT** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera Z es cero y N igual a V*
- void **BLE** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera Z es cero 0 N diferente de V*
- void **BAL** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc*
- void **BL** (uint32\_t \*LR, uint32\_t \*PC, uint32\_t salto)  
*funcion que llama una subrutina que esta en una direccion relativa al pc*
- void **BLX** (uint32\_t \*LR, uint32\_t \*PC, uint32\_t direccion)  
*funcion que llama una subrutina que esta en la posicion direccion y RL avanza a la siguiente direccion*
- void **BX** (uint32\_t \*PC, uint32\_t direccion)  
*funcion que salta a una direccion espesifica por un registro*

### 4.3.1 Function Documentation

#### 4.3.1.1 void B ( uint32\_t \* R, uint32\_t b )

funcion que aumenta o disminuye b posiciones el pc

Parameters

*R	direccion del PC
b	valor que se va a cambiar el PC

Returns

no hay retorno

#### 4.3.1.2 void BAL ( uint32\_t \* R, uint32\_t b, uint32\_t \* banderas )

funcion que aumenta o disminuye b posiciones el pc

Parameters

*R	direccion del PC
b	valor que va a cambiar el PC
*banderas	Direccion de las banderas utilizada para evaluar la condicion

Returns

no hay retorno

#### 4.3.1.3 void BCC ( uint32\_t \* R, uint32\_t b, uint32\_t \* banderas )

funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en cero

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.3.1.4 void BCS ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en uno

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.3.1.5 void BEQ ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera Z esta en uno

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.3.1.6 void BGE ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera N y V son iguales

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.3.1.7 void BGT ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera Z es cero y N igual a V

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.3.1.8 void BHI ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en uno y la Z en cero

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.3.1.9 void BL ( uint32\_t \* *RL*, uint32\_t \* *PC*, uint32\_t *salto* )

funcion que llama una subrutina que esta en una direccion relativa al pc

## Parameters

<i>salto</i>	el numero de direcciones que avanzara desde su posicion actual
<i>*PC</i>	tomara la direccion de la subrutina *PC=Reg[13]
<i>*LR</i>	tomara el valor de la siguiente direccion apartir de la posicion actual *LR=Reg[14]

## Returns

no hay retorno

4.3.1.10 void BLE ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera Z es cero 0 N diferente de V

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.3.1.11 void BLS ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en cero o la Z en uno



## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.3.1.12 void BLT ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera N y V son diferentes

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.3.1.13 void BLX ( uint32\_t \* *RL*, uint32\_t \* *PC*, uint32\_t *direccion* )

funcion que llama una subrutina que esta en la posicion direccion y RL avanza a la siguiente direccion

## Parameters

<i>direccion</i>	es un registro cuyo valor es la direccion de la subrutina
<i>*PC</i>	tomara la direccion de la subrutina *PC=Reg[13]
<i>*LR</i>	tomara el valor de la siguiente direccion apartir de la posicion actual *LR=Reg[14]

## Returns

no hay retorno

4.3.1.14 void BMI ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera N esta en uno

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.3.1.15 void BNE ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera Z esta en cero

**Parameters**

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

**Returns**

no hay retorno

**4.3.1.16 void BPL ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )**

funcion que aumenta o disminuye b posiciones el pc si la bandera N esta en cero

**Parameters**

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

**Returns**

no hay retorno

**4.3.1.17 void BVC ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )**

funcion que aumenta o disminuye b posiciones el pc si la bandera V esta en cero

**Parameters**

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

**Returns**

no hay retorno

**4.3.1.18 void BVS ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )**

funcion que aumenta o disminuye b posiciones el pc si la bandera V esta en uno

**Parameters**

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

**Returns**

no hay retorno

**4.3.1.19 void BX ( uint32\_t \* *PC*, uint32\_t *direccion* )**

funcion que salta a una direccion espesifica por un registro

## Parameters

<i>direccion</i>	es un registro cuyo valor es la direccion
<i>*PC</i>	tomara la direccion de la subrutina *PC=Reg[13]

## Returns

no hay retorno

## 4.4 branch.h File Reference

## Functions

- void **B** (uint32\_t \*R, uint32\_t b)  
*funcion que aumenta o disminuye b posiciones el pc*
- void **BEQ** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera Z esta en uno*
- void **BNE** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera Z esta en cero*
- void **BCS** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en uno*
- void **BCC** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en cero*
- void **BMI** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera N esta en uno*
- void **BPL** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera N esta en cero*
- void **BVS** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera V esta en uno*
- void **BVC** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera V esta en cero*
- void **BHI** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en uno y la Z en cero*
- void **BLS** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en cero o la Z en uno*
- void **BGE** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera N y V son iguales*
- void **BLT** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera N y V son diferentes*
- void **BGT** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera Z es cero y N igual a V*
- void **BLE** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc si la bandera Z es cero 0 N diferente de V*
- void **BAL** (uint32\_t \*R, uint32\_t b, uint32\_t \*banderas)  
*funcion que aumenta o disminuye b posiciones el pc*
- void **BL** (uint32\_t \*RL, uint32\_t \*PC, uint32\_t salto)  
*funcion que llama una subrutina que esta en una direccion relativa al pc*
- void **BLX** (uint32\_t \*RL, uint32\_t \*PC, uint32\_t direccion)  
*funcion que llama una subrutina que esta en la posicion direccion y RL avanza a la siguiente direccion*
- void **BX** (uint32\_t \*PC, uint32\_t direccion)  
*funcion que salta a una direccion espesifica por un registro*

#### 4.4.1 Function Documentation

4.4.1.1 `void B ( uint32_t * R, uint32_t b )`

funcion que aumenta o disminuye b posiciones el pc

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que se va a cambiar el PC

## Returns

no hay retorno

4.4.1.2 void BAL ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.4.1.3 void BCC ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en cero

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.4.1.4 void BCS ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en uno

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.4.1.5 void BEQ ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera Z esta en uno

**Parameters**

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

**Returns**

no hay retorno

**4.4.1.6 void BGE ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )**

funcion que aumenta o disminuye b posiciones el pc si la bandera N y V son iguales

**Parameters**

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

**Returns**

no hay retorno

**4.4.1.7 void BGT ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )**

funcion que aumenta o disminuye b posiciones el pc si la bandera Z es cero y N igual a V

**Parameters**

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

**Returns**

no hay retorno

**4.4.1.8 void BHI ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )**

funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en uno y la Z en cero

**Parameters**

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

**Returns**

no hay retorno

**4.4.1.9 void BL ( uint32\_t \* *RL*, uint32\_t \* *PC*, uint32\_t *salto* )**

funcion que llama una subrutina que esta en una direccion relativa al pc

## Parameters

<i>salto</i>	el numero de direcciones que avanzara desde su posicion actual
<i>*PC</i>	tomara la direccion de la subrutina *PC=Reg[13]
<i>*LR</i>	tomara el valor de la siguiente direccion apartir de la posicion actual *LR=Reg[14]

## Returns

no hay retorno

4.4.1.10 void BLE ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera Z es cero 0 N diferente de V

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.4.1.11 void BLS ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera C esta en cero o la Z en uno

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.4.1.12 void BLT ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera N y V son diferentes

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.4.1.13 void BLX ( uint32\_t \* *RL*, uint32\_t \* *PC*, uint32\_t *direccion* )

funcion que llama una subrutina que esta en la posicion direccion y RL avanza a la siguiente direccion

## Parameters

<i>direccion</i>	es un registro cuyo valor es la direccion de la subrutina
<i>*PC</i>	tomara la direccion de la subrutina <i>*PC=Reg[13]</i>
<i>*LR</i>	tomara el valor de la siguiente direccion apartir de la posicion actual <i>*LR=Reg[14]</i>

## Returns

no hay retorno

**4.4.1.14 void BMI ( uint32\_t \* R, uint32\_t b, uint32\_t \* banderas )**

funcion que aumenta o disminuye b posiciones el pc si la bandera N esta en uno

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

**4.4.1.15 void BNE ( uint32\_t \* R, uint32\_t b, uint32\_t \* banderas )**

funcion que aumenta o disminuye b posiciones el pc si la bandera Z esta en cero

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

**4.4.1.16 void BPL ( uint32\_t \* R, uint32\_t b, uint32\_t \* banderas )**

funcion que aumenta o disminuye b posiciones el pc si la bandera N esta en cero

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

**4.4.1.17 void BVC ( uint32\_t \* R, uint32\_t b, uint32\_t \* banderas )**

funcion que aumenta o disminuye b posiciones el pc si la bandera V esta en cero



## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.4.1.18 void BVS ( uint32\_t \* *R*, uint32\_t *b*, uint32\_t \* *banderas* )

funcion que aumenta o disminuye b posiciones el pc si la bandera V esta en uno

## Parameters

<i>*R</i>	direccion del PC
<i>b</i>	valor que va a cambiar el PC
<i>*banderas</i>	Direccion de las banderas utilizada para evaluar la condicion

## Returns

no hay retorno

4.4.1.19 void BX ( uint32\_t \* *PC*, uint32\_t *direccion* )

funcion que salta a una direccion espesifica por un registro

## Parameters

<i>direccion</i>	es un registro cuyo valor es la direccion
<i>*PC</i>	tomara la direccion de la subrutina *PC=Reg[13]

## Returns

no hay retorno

## 4.5 decoder.c File Reference

```
#include "decoder.h"
#include "ALU.h"
#include "registros.h"
#include "desplazamiento.h"
#include "curses.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include "branch.h"
```

## Functions

- void [decodeInstruction](#) ([instruction\\_t](#) instruction, uint32\_t \*Reg, uint32\_t \*banderas)
- [instruction\\_t getInstruction](#) (char \*instStr)

*Obtiene la instrucción separada por partes.*

- int [readFile](#) (char \*filename, [ins\\_t](#) \*instructions)
- int [countLines](#) (FILE \*fp)

#### 4.5.1 Function Documentation

4.5.1.1 int [countLines](#) ( FILE \* *fp* )

4.5.1.2 void [decodeInstruction](#) ( [instruction\\_t](#) *instruction*, uint32\_t \* *Reg*, uint32\_t \* *banderas* )

4.5.1.3 [instruction\\_t](#) [getInstruction](#) ( char \* *instStr* )

Obtiene la instrucción separada por partes.

Parameters

<i>instStr</i>	cadena que contiene la instrucción.
----------------	-------------------------------------

Returns

[instruction\\_t](#) la instrucción separada por partes.

4.5.1.4 int [readFile](#) ( char \* *filename*, [ins\\_t](#) \* *instructions* )

## 4.6 decoder.h File Reference

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

### Data Structures

- struct [ins\\_t](#)
- struct [instruction\\_t](#)

### Functions

- void [decodeInstruction](#) ([instruction\\_t](#) *instruction*, uint32\_t \**Reg*, uint32\_t \**banderas*)
- [instruction\\_t](#) [getInstruction](#) (char \**instStr*)  
*Obtiene la instrucción separada por partes.*
- int [readFile](#) (char \*filename, [ins\\_t](#) \*instructions)
- int [countLines](#) (FILE \*fp)

#### 4.6.1 Function Documentation

4.6.1.1 int [countLines](#) ( FILE \* *fp* )

4.6.1.2 void [decodeInstruction](#) ( [instruction\\_t](#) *instruction*, uint32\_t \* *Reg*, uint32\_t \* *banderas* )

4.6.1.3 [instruction\\_t](#) [getInstruction](#) ( char \* *instStr* )

Obtiene la instrucción separada por partes.

## Parameters

<i>instrStr</i>	cadena que contiene la instrucción.
-----------------	-------------------------------------

## Returns

[instruction\\_t](#) la instrucción separada por partes.

4.6.1.4 `int readFile ( char * filename, ins_t * instructions )`

## 4.7 desplazamiento.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "desplazamiento.h"
#include "ALU.h"
```

### Functions

- void [LSLS](#) (uint32\_t \*Rd, uint32\_t Rm, uint32\_t num, uint32\_t \*banderas)  
*funcion para desplazar el dato Rm cierta cantidad de veces hacia la izquierda*
- void [LSRS](#) (uint32\_t \*Rd, uint32\_t Rm, uint32\_t num, uint32\_t \*banderas)  
*funcion para desplazar el dato Rm cierta cantidad de veces hacia la derecha*
- void [RORS](#) (uint32\_t \*Rd, uint32\_t Rm, uint32\_t num, uint32\_t \*banderas)  
*funcion para rotar hacia la derecha un dato*
- void [ASRS](#) (uint32\_t \*Rd, uint32\_t Rm, uint32\_t num, uint32\_t \*banderas)  
*funcion para hacer un desplazamiento aritmetico a la derecha*
- void [REV](#) (uint32\_t \*Rd, uint32\_t Rm)  
*funcion para desplazar paquetes de 8 bits*
- void [REV16](#) (uint32\_t \*Rd, uint32\_t Rm)  
*funcion para desplazar paquetes de 16 bits*
- void [REVSH](#) (uint32\_t \*Rd, uint32\_t Rm)  
*funcion para intercambiar los dos byts menos significativos.*

### 4.7.1 Function Documentation

4.7.1.1 `void ASRS ( uint32_t * Rd, uint32_t Rm, uint32_t num, uint32_t * banderas )`

funcion para hacer un desplazamiento aritmetico a la derecha

## Parameters

<i>*Rd</i>	variable donde se almacena el desplazamiento
<i>Rm</i>	registro que se desplazara
<i>num</i>	numero de bits que se desplazara

## Returns

no hay retorno

4.7.1.2 void LSLS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *num*, uint32\_t \* *banderas* )

funcion para desplazar el dato *Rm* cierta cantidad de veces hacia la izquierda

## Parameters

<i>*Rd</i>	registro donde se guardara el dato desplazado
<i>Rm</i>	registro que se desplazara
<i>num</i>	numero de bits que se desplazara el dato Rm

## Returns

no hay retorno

## 4.7.1.3 void LSRS ( uint32\_t \* Rd, uint32\_t Rm, uint32\_t num, uint32\_t \* banderas )

funcion para desplazar el dato Rm cierta cantidad de veces hacia la derecha

## Parameters

<i>*Rd</i>	registro donde se guardara el dato desplazado
<i>Rm</i>	registro que se desplazara
<i>num</i>	numero de bits que se desplazara el dato Rm

## Returns

no hay retorno

## 4.7.1.4 void REV ( uint32\_t \* Rd, uint32\_t Rm )

funcion para desplazar paquetes de 8 bits

## Parameters

<i>*Rd</i>	variable donde se almacena el desplazamiento
<i>Rm</i>	registro que se desplazara

## Returns

no hay retorno

## 4.7.1.5 void REV16 ( uint32\_t \* Rd, uint32\_t Rm )

funcion para desplazar paquetes de 16 bits

## Parameters

<i>*Rd</i>	variable donde se almacena el desplazamiento
<i>Rm</i>	registro que se desplazara

## Returns

no hay retorno

## 4.7.1.6 void REVSH ( uint32\_t \* Rd, uint32\_t Rm )

funcion para intercambiar los dos byts menos significativos.

## Parameters

<i>*Rd</i>	variable donde se almacena el desplazamiento
<i>Rm</i>	registro que se desplazara

## Returns

no hay retorno

4.7.1.7 void RORS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *num*, uint32\_t \* *banderas* )

funcion para rotar hacia la derecha un dato

## Parameters

<i>*Rd</i>	variable donde se almacena la rotacion
<i>Rm</i>	registro que se rotara
<i>num</i>	numero de bits que se rotara el dato Rm

## Returns

no hay retorno

## 4.8 desplazamiento.h File Reference

## Functions

- void [ASRS](#) (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *num*, uint32\_t \**banderas*)  
*funcion para hacer un desplazamiento aritmetico a la derecha*
- void [LSLS](#) (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *num*, uint32\_t \**banderas*)  
*funcion para desplazar el dato Rm cierta cantidad de veces hacia la izquierda*
- void [LSRS](#) (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *num*, uint32\_t \**banderas*)  
*funcion para desplazar el dato Rm cierta cantidad de veces hacia la derecha*
- void [REV](#) (uint32\_t \**Rd*, uint32\_t *Rm*)  
*funcion para desplazar paquetes de 8 bits*
- void [REV16](#) (uint32\_t \**Rd*, uint32\_t *Rm*)  
*funcion para desplazar paquetes de 16 bits*
- void [REVSH](#) (uint32\_t \**Rd*, uint32\_t *Rm*)  
*funcion para intercambiar los dos byts menos significativos.*
- void [RORS](#) (uint32\_t \**Rd*, uint32\_t *Rm*, uint32\_t *num*, uint32\_t \**banderas*)  
*funcion para rotar hacia la derecha un dato*

### 4.8.1 Function Documentation

4.8.1.1 void ASRS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *num*, uint32\_t \* *banderas* )

funcion para hacer un desplazamiento aritmetico a la derecha

## Parameters

<i>*Rd</i>	variable donde se almacena el desplazamiento
<i>Rm</i>	registro que se desplazara
<i>num</i>	numero de bits que se desplazara

**Returns**

no hay retorno

4.8.1.2 void LSLS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *num*, uint32\_t \* *banderas* )

funcion para desplazar el dato *Rm* cierta cantidad de veces hacia la izquierda

**Parameters**

<i>*Rd</i>	registro donde se guardara el dato desplazado
<i>Rm</i>	registro que se desplazara
<i>num</i>	numero de bits que se desplazara el dato <i>Rm</i>

**Returns**

no hay retorno

4.8.1.3 void LSRS ( uint32\_t \* *Rd*, uint32\_t *Rm*, uint32\_t *num*, uint32\_t \* *banderas* )

funcion para desplazar el dato *Rm* cierta cantidad de veces hacia la derecha

**Parameters**

<i>*Rd</i>	registro donde se guardara el dato desplazado
<i>Rm</i>	registro que se desplazara
<i>num</i>	numero de bits que se desplazara el dato <i>Rm</i>

**Returns**

no hay retorno

4.8.1.4 void REV ( uint32\_t \* *Rd*, uint32\_t *Rm* )

funcion para desplazar paquetes de 8 bits

**Parameters**

<i>*Rd</i>	variable donde se almacena el desplazamiento
<i>Rm</i>	registro que se desplazara

**Returns**

no hay retorno

4.8.1.5 void REV16 ( uint32\_t \* *Rd*, uint32\_t *Rm* )

funcion para desplazar paquetes de 16 bits

**Parameters**

<i>*Rd</i>	variable donde se almacena el desplazamiento
<i>Rm</i>	registro que se desplazara

**Returns**

no hay retorno

**4.8.1.6 void REVSH ( uint32\_t \* Rd, uint32\_t Rm )**

funcion para intercambiar los dos byts menos significativos.

**Parameters**

<i>*Rd</i>	variable donde se almacena el desplazamiento
<i>Rm</i>	registro que se desplazara

**Returns**

no hay retorno

**4.8.1.7 void RORS ( uint32\_t \* Rd, uint32\_t Rm, uint32\_t num, uint32\_t \* banderas )**

funcion para rotar hacia la derecha un dato

**Parameters**

<i>*Rd</i>	variable donde se almacena la rotacion
<i>Rm</i>	registro que se rotara
<i>num</i>	numero de bits que se rotara el dato Rm

**Returns**

no hay retorno

## 4.9 main.c File Reference

```
#include "decoder.h"
#include "ALU.h"
#include "registros.h"
#include "desplazamiento.h"
#include "curses.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdint.h>
```

**Functions**

- int [main](#) (void)



### 4.9.1 Function Documentation

#### 4.9.1.1 int main ( void )

## 4.10 registros.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "registros.h"
#include <urses.h>
```

### Functions

- void [mostrar\\_registros](#) (uint32\_t \*registro)  
*Funcion que toma los valores de los registros y los imprime en pantalla.*
- void [mostrar\\_banderas](#) (uint32\_t banderas[4])  
*Funcion que toma los valores de las banderas y los imprime en pantalla.*
- void [mostrar\\_operacion](#) (char \*op)  
*Funcion que muestra la operacion que se realizo.*

### Variables

- unsigned int [i](#)

### 4.10.1 Function Documentation

#### 4.10.1.1 void mostrar\_banderas ( uint32\_t banderas[4] )

Funcion que toma los valores de las banderas y los imprime en pantalla.

##### Parameters

<i>banderas[4]</i>	Arreglo que contiene cada 1 de las banderas
--------------------	---

##### Returns

No tiene retorno

#### 4.10.1.2 void mostrar\_operacion ( char \* op )

Funcion que muestra la operacion que se realizo.

##### Parameters

<i>op[5]</i>	variable tipo caracter donde se gusrda el nombre de la operacion realizada
--------------	--

##### Returns

No tiene retorno

#### 4.10.1.3 void mostrar\_registros ( uint32\_t \* registro )

Funcion que toma los valores de los registros y los imprime en pantalla.

## Parameters

<i>registro[13]</i>	Arreglo que contiene cada 1 de los registros
---------------------	--

## Returns

No tiene retorno

## 4.10.2 Variable Documentation

### 4.10.2.1 unsigned int i

## 4.11 registros.h File Reference

## Functions

- void [mostrar\\_registros](#) (uint32\_t \*registro)  
*Funcion que toma los valores de los registros y los imprime en pantalla.*
- void [mostrar\\_banderas](#) (uint32\_t banderas[4])  
*Funcion que toma los valores de las banderas y los imprime en pantalla.*
- void [mostrar\\_operacion](#) (char \*op)  
*Funcion que muestra la operacion que se realizo.*

### 4.11.1 Function Documentation

#### 4.11.1.1 void mostrar\_banderas ( uint32\_t banderas[4] )

Funcion que toma los valores de las banderas y los imprime en pantalla.

## Parameters

<i>banderas[4]</i>	Arreglo que contiene cada 1 de las banderas
--------------------	---

## Returns

No tiene retorno

#### 4.11.1.2 void mostrar\_operacion ( char \* op )

Funcion que muestra la operacion que se realizo.

## Parameters

<i>op[5]</i>	variable tipo caracter donde se gusrda el nombre de la operacion realizada
--------------	--

## Returns

No tiene retorno

#### 4.11.1.3 void mostrar\_registros ( uint32\_t \* registro )

Funcion que toma los valores de los registros y los imprime en pantalla.

## Parameters

<i>registro[13]</i>	Arreglo que contiene cada 1 de los registros
---------------------	--

## Returns

No tiene retorno

## 4.12 test.c File Reference

