My Project

Generado por Doxygen 1.8.10

Martes, 20 de Octubre de 2015 07:54:40

Índice general

Capítulo 1

Índice de estructura de datos

1.1. Estructura de datos

Lista de estructuras con una breve descripción:

Capítulo 2

Indice de archivos

2.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

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

Funciones

- 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 no actualiza banderas
- 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. Documentación de las funciones

4.1.1.1. void ACTNZ (uint32_t * *Rd*, uint32_t * *banderas*)

funcion que actualiza las banderas N y Z

Parámetros

**Rd*	resultado
**banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*Rd*	lugar donde se almacenara el resultado

Devuelve

no hay retorno

4.1.1.4. void ADDS (uint32_t ∗ *Rd,* uint32_t *Rm,* uint32_t *Rn,* uint32_t ∗ *banderas*)

funcion que suma

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*Rd*	lugar donde se almacenara el resultado
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*Rd*	lugar donde se almacenara el resultado
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	Registro de entrada
∗*Rd*	lugar donde se almacenara el resultado.
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*banderas*	direccion en memoria de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	Registro de entrada
∗*Rd*	lugar donde se almacenara el resultado.
∗*banderas*	direccion de memoria de las banderas

Devuelve

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 no actualiza banderas

Parámetros

Rn	Registro que se va a mover
∗*Rm*	lugar donde se almacenaria el resultado.

Devuelve

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

Parámetros

Rn	Registro que se mueve
∗*Rm*	lugar donde se almacenara el resultado.
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

∗*Rd*	registro donde se guarda el resultado
Rm	registro a multiplicar
Rn	registro a multiplicar
∗*banderas*	direccion de memoria de las bandera

Devuelve

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

Parámetros

Rm	Registro de entrada
∗*Rd*	lugar donde se almacenara el resultado
∗*banderas*	direccion de memoria de las bandera

Devuelve

no hay retorno

4.1.1.14. void NOP ( )

funcion para no hacer nada

Parámetros

no	tiene parametros de entrada

Devuelve

no hay retorno

4.1.1.15. void ORRS (uint32_t ∗ *Rd,* uint32_t *Rm,* uint32_t *Rn,* uint32_t ∗ *banderas*)

funcion OR

Parámetros

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

Devuelve

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

Parámetros

∗*Rd*	registro donde se guarda el resultado
Rm	registro al que se le aplicara el complemento
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

∗*Rd*	registro donde se guarda el resultado
Rm	registro a de entrada
Rn	registro a restar con el de entrada
∗*banderas*	direccion de memoria de las banderas

Devuelve

no hay retorno

4.1.1.18. void SUB (uint32_t ∗ *Rd,* uint32_t *Rm,* uint32_t *Rn*)

funcion que resta

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*Rd*	lugar donde se almacenara el resultado

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*Rd*	lugar donde se almacenara el resultado
∗*banderas*	direccion de memora de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*banderas*	direccion de memoria de las banderas

Devuelve

no hay retorno

4.2. Referencia del Archivo ALU.h

Funciones

- 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 no actualiza banderas
- 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. Documentación de las funciones

4.2.1.1. void ACTNZ (uint32_t ∗ *Rd,* uint32_t ∗ *banderas*)

funcion que actualiza las banderas N y Z

Parámetros

∗*Rd*	resultado
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*Rd*	lugar donde se almacenara el resultado

Devuelve

no hay retorno

4.2.1.4. void ADDS (uint32_t ∗ *Rd,* uint32_t *Rm,* uint32_t *Rn,* uint32_t ∗ *banderas*)

funcion que suma

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*Rd*	lugar donde se almacenara el resultado
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*Rd*	lugar donde se almacenara el resultado
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	Registro de entrada
∗*Rd*	lugar donde se almacenara el resultado.
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*banderas*	direccion en memoria de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	Registro de entrada
∗*Rd*	lugar donde se almacenara el resultado.
∗*banderas*	direccion de memoria de las banderas

Devuelve

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 no actualiza banderas

Parámetros

Rn	Registro que se va a mover
∗*Rm*	lugar donde se almacenaria el resultado.

Devuelve

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

Parámetros

Rn	Registro que se mueve
∗*Rm*	lugar donde se almacenara el resultado.
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

∗*Rd*	registro donde se guarda el resultado
Rm	registro a multiplicar
Rn	registro a multiplicar
∗*banderas*	direccion de memoria de las bandera

Devuelve

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

Parámetros

Rm	Registro de entrada
∗*Rd*	lugar donde se almacenara el resultado
∗*banderas*	direccion de memoria de las bandera

Devuelve

no hay retorno

4.2.1.14. void NOP ( )

funcion para no hacer nada

Parámetros

no	tiene parametros de entrada

Devuelve

no hay retorno

4.2.1.15. void ORRS (uint32_t ∗ *Rd,* uint32_t *Rm,* uint32_t *Rn,* uint32_t ∗ *banderas*)

funcion OR

Parámetros

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

Devuelve

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

Parámetros

∗*Rd*	registro donde se guarda el resultado
Rm	registro al que se le aplicara el complemento
∗*banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

∗*Rd*	registro donde se guarda el resultado
Rm	registro a de entrada
Rn	registro a restar con el de entrada
∗*banderas*	direccion de memoria de las banderas

Devuelve

no hay retorno

4.2.1.18. void SUB (uint32_t ∗ *Rd,* uint32_t *Rm,* uint32_t *Rn*)

funcion que resta

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*Rd*	lugar donde se almacenara el resultado

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*Rd*	lugar donde se almacenara el resultado
∗*banderas*	direccion de memora de las banderas

Devuelve

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

Parámetros

Rn	Registro de entrada
Rm	registro de entrada
∗*banderas*	direccion de memoria de las banderas

Devuelve

no hay retorno

4.3. Referencia del Archivo branch.c

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

Funciones

- 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 posisiones 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. Documentación de las funciones

4.3.1.1. void B (uint32_t ∗ *R,* uint32_t *b*)

funcion que aumenta o disminuye b posiciones el pc

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

no hay retorno

4.3.1.3. void BCC (uint32_t ∗ *R,* uint32_t *b,* uint32_t ∗ *banderas*)

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

salto	el numero de direcciones que avanzara desde su posicion actual
∗PC	tomara la direccion de la subrutina ∗PC=Reg[15]
∗LR	tomara el valor de la siguiente direccion apartir de la posicion actual ∗LR=Reg[14]

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

direccion	es un registro cuyo valor es la direccion de la subrutina
∗*PC*	tomara la direccion de la subrutina ∗PC=Reg[15]
∗*LR*	tomara el valor de la siguiente direccion apartir de la posicion actual ∗LR=Reg[14]

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

direccion	es un registro cuyo valor es la direccion
∗PC	tomara la direccion de la subrutina ∗PC=Reg[15]

Devuelve

no hay retorno

4.4. Referencia del Archivo branch.h

Funciones

- 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 posisiones 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. Documentación de las funciones

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

funcion que aumenta o disminuye b posiciones el pc

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

no hay retorno

4.4.1.3. void BCC (uint32_t ∗ *R*, uint32_t *b*, uint32_t ∗ *banderas*)

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

salto	el numero de direcciones que avanzara desde su posicion actual
∗*PC*	tomara la direccion de la subrutina ∗PC=Reg[15]
∗*LR*	tomara el valor de la siguiente direccion apartir de la posicion actual ∗LR=Reg[14]

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

direccion	es un registro cuyo valor es la direccion de la subrutina
∗*PC*	tomara la direccion de la subrutina ∗PC=Reg[15]
∗*LR*	tomara el valor de la siguiente direccion apartir de la posicion actual ∗LR=Reg[14]

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

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

Devuelve

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

Parámetros

direccion	es un registro cuyo valor es la direccion
∗*PC*	tomara la direccion de la subrutina ∗PC=Reg[15]

Devuelve

no hay retorno

4.5. Referencia del Archivo decoder.c

```
#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"
#include "LoadStore.h"
#include "PILA.h"
```

Funciones

- void decodeInstruction (instruction_t instruction, uint32_t *Reg, uint32_t *banderas, uint8_t *SR, uint16_t *operacion)
- 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. Documentación de las funciones

4.5.1.1. int countLines (FILE * *fp*)

4.5.1.2. void decodeInstruction (instruction_t *instruction,* uint32_t * *Reg,* uint32_t * *banderas,* uint8_t * *SR,* uint16_t * *operacion*)

4.5.1.3. instruction_t getInstruction (char * *instStr*)

Obtiene la instrucción separada por partes.

Parámetros

instrStr	cadena que contiene la instrucción.
*operacion	almacenara la direccion asignada en memoria para una instruccion especifica

Devuelve

instruction_t la instrucción separada por partes.

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>
```

Estructuras de datos

- struct ins_t
- struct instruction_t

Funciones

- void decodeInstruction (instruction_t instruction, uint32_t *Reg, uint32_t *banderas, uint8_t *pila, uint16_t *operacion)
- 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. Documentación de las funciones

4.6.1.1. int countLines (FILE * *fp*)

4.6.1.2. void decodeInstruction (instruction_t *instruction,* uint32_t * *Reg,* uint32_t * *banderas,* uint8_t * *pila,* uint16_t * *operacion*)

4.6.1.3. instruction_t getInstruction (char * *instStr*)

Obtiene la instrucción separada por partes.

Parámetros

instrStr	cadena que contiene la instrucción.
**operacion*	almacenara la direccion asignada en memoria para una instruccion especifica

Devuelve

instruction_t la instrucción separada por partes.

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

4.7. Referencia del Archivo desplazamiento.c

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

Funciones

- 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. Documentación de las funciones

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

Parámetros

∗Rd	variable donde se almacena el desplazamiento
Rm	registro que se desplazara
num	numero de bits que se desplazara
∗banderas	direccion de memoria de la las banderas

Devuelve

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

Parámetros

∗Rd	registro donde se guardara el dato desplazado
Rm	registro que se desplazara
num	numero de bits que se desplazara el dato Rm
∗banderas	direccion de memoria de las banderas

Devuelve

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

Parámetros

∗Rd	registro donde se guardara el dato desplazado
Rm	registro que se desplazara
num	numero de bits que se desplazara el dato Rm
∗banderas	direccion de memoria de las banderas

Devuelve

no hay retorno

4.7.1.4. void REV (uint32_t ∗ *Rd,* uint32_t *Rm*)

funcion para desplazar paquetes de 8 bits

Parámetros

∗Rd	variable donde se almacena el desplazamiento
Rm	registro que se desplazara

Devuelve

no hay retorno

4.7.1.5. void REV16 (uint32_t ∗ *Rd,* uint32_t *Rm*)

funcion para desplazar paquetes de 16 bits

Parámetros

∗Rd	variable donde se almacena el desplazamiento
Rm	registro que se desplazara

Devuelve

no hay retorno

4.7.1.6. void REVSH (uint32_t ∗ *Rd,* uint32_t *Rm*)

funcion para intercambiar los dos byts menos significativos.

Parámetros

∗Rd	variable donde se almacena el desplazamiento
Rm	registro que se desplazara

Devuelve

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

Parámetros

∗Rd	variable donde se almacena la rotacion
Rm	registro que se rotara
num	numero de bits que se rotara el dato Rm
∗banderas	direccion de memoria de las banderas

Devuelve

no hay retorno

4.8. Referencia del Archivo desplazamiento.h

Funciones

- 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. Documentación de las funciones

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

Parámetros

**Rd*	variable donde se almacena el desplazamiento
Rm	registro que se desplazara
num	numero de bits que se desplazara
**banderas*	direccion de memoria de la las banderas

Devuelve

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

Parámetros

**Rd*	registro donde se guardara el dato desplazado
Rm	registro que se desplazara
num	numero de bits que se desplazara el dato Rm
**banderas*	direccion de memoria de las banderas

Devuelve

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

Parámetros

**Rd*	registro donde se guardara el dato desplazado
Rm	registro que se desplazara
num	numero de bits que se desplazara el dato Rm
**banderas*	direccion de memoria de las banderas

Devuelve

no hay retorno

4.8.1.4. void REV (uint32_t * *Rd,* uint32_t *Rm*)

funcion para desplazar paquetes de 8 bits

Parámetros

∗Rd	variable donde se almacena el desplazamiento
Rm	registro que se desplazara

Devuelve

no hay retorno

4.8.1.5. void REV16 (uint32_t ∗ *Rd,* uint32_t *Rm*)

funcion para desplazar paquetes de 16 bits

Parámetros

∗Rd	variable donde se almacena el desplazamiento
Rm	registro que se desplazara

Devuelve

no hay retorno

4.8.1.6. void REVSH (uint32_t ∗ *Rd,* uint32_t *Rm*)

funcion para intercambiar los dos byts menos significativos.

Parámetros

∗Rd	variable donde se almacena el desplazamiento
Rm	registro que se desplazara

Devuelve

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

Parámetros

∗Rd	variable donde se almacena la rotacion
Rm	registro que se rotara
num	numero de bits que se rotara el dato Rm
∗banderas	direccion de memoria de las banderas

Devuelve

no hay retorno

4.9. Referencia del Archivo interrupciones.c

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

Funciones

- void NVIC (uint32_t ∗Reg, uint32_t ∗banderas, uint8_t ∗SR, uint8_t ∗Rin)

  funcion que me controla la interrupcion que se va a ejecutar

4.9.1. Documentación de las funciones

4.9.1.1. void NVIC (uint32_t ∗ *Reg,* uint32_t ∗ *banderas,* uint8_t ∗ *SR,* uint8_t ∗ *Rin*)

funcion que me controla la interrupcion que se va a ejecutar

Parámetros

∗*Reg*	direccion en memoria de los registros
∗*banderas*	direccion en memoria de las banderas
∗*SR*	direccion en memoria de la SRAM
∗*Rin*	direccion en memoria del arreglo donde se encuentra la interrupccion que se va a ejecutar

Devuelve

no hay retorno

4.10. Referencia del Archivo interrupciones.h

Funciones

- void NVIC (uint32_t ∗Reg, uint32_t ∗banderas, uint8_t ∗SR, uint8_t ∗Rin)

  funcion que me controla la interrupcion que se va a ejecutar

4.10.1. Documentación de las funciones

4.10.1.1. void NVIC (uint32_t ∗ *Reg,* uint32_t ∗ *banderas,* uint8_t ∗ *SR,* uint8_t ∗ *Rin*)

funcion que me controla la interrupcion que se va a ejecutar

Parámetros

∗*Reg*	direccion en memoria de los registros
∗*banderas*	direccion en memoria de las banderas
∗*SR*	direccion en memoria de la SRAM
∗*Rin*	direccion en memoria del arreglo donde se encuentra la interrupccion que se va a ejecutar

Devuelve

no hay retorno

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 40
- #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 40

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 LoadStore.c

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

Funciones

- void LDR (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM, uint32_t imm)
  funcion para cargar un dato de la RAM al registro Rt
- void LDRB (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
  funcion para cargar 8 bits de la RAM al registro Rt
- void LDRH (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM, uint32_t imm)
  funcion para cargar 16 bits de la RAM al registro Rt
- void LDRSB (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
  funcion para cargar 8 bits de la RAM al registro Rt con extension de signo
- void LDRSH (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
  funcion para cargar 16 bits de la RAM al registro Rt con extension de signo
- void STR (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM, uint32_t imm)
  funcion que toma un registro y lo escribe en cierta direccion de memoria
- void STRB (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
  funcion almacena los primeros 8 bits de un registro en cierta direccion de la RAM
- void STRH (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM, uint32_t imm)
  funcion almacena los primeros 16 bits de un registro en cierta direccion de la RAM

Variables

- uint32_t adress
- uint8_t mem

4.13.1. Documentación de las funciones

4.13.1.1. void LDR (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint8_t * *SRAM*, uint32_t *imm*)

funcion para cargar un dato de la RAM al registro Rt

Parámetros

*Rt	resultado
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram puede ser un dato o un inmediato
imm	es 1 si Rm es un inmediato cero en caso de ser un registro
*SRAM	posicion cero de la memoria RAM

Devuelve

no hay retorno

4.13.1.2. void LDRB (uint32_t * *Rt,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *SRAM*)

funcion para cargar 8 bits de la RAM al registro Rt

Parámetros

$*Rt$	resultado
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram puede ser un dato o un inmediato
$*SRAM$	posicion cero de la memoria RAM

Devuelve

no hay retorno

4.13.1.3. void LDRH (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint8_t * *SRAM*, uint32_t *imm*)

funcion para cargar 16 bits de la RAM al registro Rt

Parámetros

$*Rt$	resultado
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram puede ser un dato o un inmediato
$*SRAM$	posicion cero de la memoria RAM
imm	es 1 si Rm es un inmediato cero en caso de ser un registro

Devuelve

no hay retorno

4.13.1.4. void LDRSB (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint8_t * *SRAM*)

funcion para cargar 8 bits de la RAM al registro Rt con extension de signo

Parámetros

$*Rt$	resultado
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram
$*SRAM$	posicion cero de la memoria RAM

Devuelve

no hay retorno

4.13.1.5. void LDRSH (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint8_t * *SRAM*)

funcion para cargar 16 bits de la RAM al registro Rt con extension de signo

Parámetros

$*Rt$	resultado
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram
$*SRAM$	posicion cero de la memoria RAM

Devuelve

no hay retorno

4.13.1.6. void STR (uint32_t *Rt,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *SRAM,* uint32_t *imm*)

funcion que toma un registro y lo escribe en cierta direccion de memoria

Parámetros

Rt	dato a escribir
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram
∗*SRAM*	posicion cero de la memoria RAM
imm	indica si Rm es inmediato si es 1 cero de lo contrario

Devuelve

no hay retorno

4.13.1.7. void STRB (uint32_t *Rt,* uint32_t *Rn,* uint32_t *Rm,* uint8_t ∗ *SRAM*)

funcion almacena los primeros 8 bits de un registro en cierta direccion de la RAM

Parámetros

Rt	dato a escribir
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram
∗*SRAM*	posicion cero de la memoria RAM

Devuelve

no hay retorno

4.13.1.8. void STRH (uint32_t *Rt,* uint32_t *Rn,* uint32_t *Rm,* uint8_t ∗ *SRAM,* uint32_t *imm*)

funcion almacena los primeros 16 bits de un registro en cierta direccion de la RAM

Parámetros

Rt	dato a escribir
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram
∗*SRAM*	posicion cero de la memoria RAM
imm	indica si Rm es inmediato si es 1 cero de lo contrario

Devuelve

no hay retorno

4.13.2. Documentación de las variables

4.13.2.1. uint32_t adress

4.13.2.2. uint8_t mem

4.14. Referencia del Archivo LoadStore.h

Funciones

- void LDR (uint32_t ∗Rt, uint32_t Rn, uint32_t Rm, uint8_t ∗SRAM, uint32_t imm)

  funcion para cargar un dato de la RAM al registro Rt

- void LDRB (uint32_t ∗Rt, uint32_t Rn, uint32_t Rm, uint8_t ∗SRAM)

  funcion para cargar 8 bits de la RAM al registro Rt
- void LDRH (uint32_t ∗Rt, uint32_t Rn, uint32_t Rm, uint8_t ∗SRAM, uint32_t imm)

  funcion para cargar 16 bits de la RAM al registro Rt
- void LDRSB (uint32_t ∗Rt, uint32_t Rn, uint32_t Rm, uint8_t ∗SRAM)

  funcion para cargar 8 bits de la RAM al registro Rt con extension de signo
- void LDRSH (uint32_t ∗Rt, uint32_t Rn, uint32_t Rm, uint8_t ∗SRAM)

  funcion para cargar 16 bits de la RAM al registro Rt con extension de signo
- void STR (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t ∗SRAM, uint32_t imm)

  funcion que toma un registro y lo escribe en cierta direccion de memoria
- void STRB (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t ∗SRAM)

  funcion almacena los primeros 8 bits de un registro en cierta direccion de la RAM
- void STRH (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t ∗SRAM, uint32_t imm)

  funcion almacena los primeros 16 bits de un registro en cierta direccion de la RAM

4.14.1. Documentación de las funciones

4.14.1.1. void LDR (uint32_t ∗ *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint8_t ∗ *SRAM*, uint32_t *imm*)

funcion para cargar un dato de la RAM al registro Rt

Parámetros

∗*Rt*	resultado
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram puede ser un dato o un inmediato
imm	es 1 si Rm es un inmediato cero en caso de ser un registro
∗*SRAM*	posicion cero de la memoria RAM

Devuelve

no hay retorno

4.14.1.2. void LDRB (uint32_t ∗ *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint8_t ∗ *SRAM*)

funcion para cargar 8 bits de la RAM al registro Rt

Parámetros

∗*Rt*	resultado
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram puede ser un dato o un inmediato
∗*SRAM*	posicion cero de la memoria RAM

Devuelve

no hay retorno

4.14.1.3. void LDRH (uint32_t ∗ *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint8_t ∗ *SRAM*, uint32_t *imm*)

funcion para cargar 16 bits de la RAM al registro Rt

Parámetros

$*Rt$	resultado
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram puede ser un dato o un inmediato
$*SRAM$	posicion cero de la memoria RAM
imm	es 1 si Rm es un inmediato cero en caso de ser un registro

Devuelve

no hay retorno

4.14.1.4. void LDRSB (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint8_t * *SRAM*)

funcion para cargar 8 bits de la RAM al registro Rt con extension de signo

Parámetros

$*Rt$	resultado
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram
$*SRAM$	posicion cero de la memoria RAM

Devuelve

no hay retorno

4.14.1.5. void LDRSH (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint8_t * *SRAM*)

funcion para cargar 16 bits de la RAM al registro Rt con extension de signo

Parámetros

$*Rt$	resultado
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram
$*SRAM$	posicion cero de la memoria RAM

Devuelve

no hay retorno

4.14.1.6. void STR (uint32_t *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint8_t * *SRAM*, uint32_t *imm*)

funcion que toma un registro y lo escribe en cierta direccion de memoria

Parámetros

Rt	dato a escribir
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram
$*SRAM$	posicion cero de la memoria RAM

imm	indica si Rm es inmediato si es 1 cero de lo contrario

Devuelve

no hay retorno

4.14.1.7. void STRB (uint32_t *Rt,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *SRAM*)

funcion almacena los primeros 8 bits de un registro en cierta direccion de la RAM

Parámetros

Rt	dato a escribir
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram
**SRAM*	posicion cero de la memoria RAM

Devuelve

no hay retorno

4.14.1.8. void STRH (uint32_t *Rt,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *SRAM,* uint32_t *imm*)

funcion almacena los primeros 16 bits de un registro en cierta direccion de la RAM

Parámetros

Rt	dato a escribir
Rn	posicion de memoria en ram
Rm	posicion de memoria en ram
**SRAM*	posicion cero de la memoria RAM
imm	indica si Rm es inmediato si es 1 cero de lo contrario

Devuelve

no hay retorno

4.15. Referencia del Archivo main.c

```
#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 "PILA.h"
#include "LoadStore.h"
#include "interrupciones.h"
#include "io.h"
```

Funciones

- int main (void)

Variables

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

4.15.1. Documentación de las funciones

4.15.1.1. int main (void)

4.15.2. Documentación de las variables

4.15.2.1. uint8_t irq[16]

4.15.2.2. port_t PORTA

4.15.2.3. port_t PORTB

4.16. Referencia del Archivo PILA.c

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

Funciones

- void PUSH (uint8_t registros[16], uint32_t *Reg, uint8_t *SR)

  funcion para escribir datos en la pila
- void POP (uint8_t registros[16], uint32_t *Reg, uint8_t *SR)

  funcion para escribir datos en la pila

4.16.1. Documentación de las funciones

4.16.1.1. void POP (uint8_t *registros[16],* uint32_t * *Reg,* uint8_t * *SR*)

funcion para escribir datos en la pila

Parámetros

registros[16]	
*Reg	dato del registro cero
*PILA	valor en la primer direccion de memoria en la pila

Devuelve

no hay retorno

4.16.1.2. void PUSH (uint8_t *registros[16],* uint32_t * *Reg,* uint8_t * *SR*)

funcion para escribir datos en la pila

Parámetros

registros[16]	
∗*Reg*	dato del registro cero
∗*PILA*	valor en la primer direccion de memoria en la pila

Devuelve

no hay retorno

4.17. Referencia del Archivo PILA.h

Funciones

- void PUSH (uint8_t registros[16], uint32_t ∗Reg, uint8_t ∗SR)

  funcion para escribir datos en la pila
- void POP (uint8_t registros[16], uint32_t ∗Reg, uint8_t ∗SR)

  funcion para escribir datos en la pila

4.17.1. Documentación de las funciones

4.17.1.1. void POP (uint8_t *registros[16],* uint32_t ∗ *Reg,* uint8_t ∗ *SR*)

funcion para escribir datos en la pila

Parámetros

registros[16]	
∗*Reg*	dato del registro cero
∗*PILA*	valor en la primer direccion de memoria en la pila

Devuelve

no hay retorno

4.17.1.2. void PUSH (uint8_t *registros[16],* uint32_t ∗ *Reg,* uint8_t ∗ *SR*)

funcion para escribir datos en la pila

Parámetros

registros[16]	
∗*Reg*	dato del registro cero
∗*PILA*	valor en la primer direccion de memoria en la pila

Devuelve

no hay retorno

4.18. Referencia del Archivo ports.c

4.19. Referencia del Archivo registros.c

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

Funciones

- void mostrar_registros (uint32_t *registro)

  Funcion que toma los valores de los registros y los imprime en pantalla.
- void mostrar_SRam (uint8_t *SRam)

  Funcion que muestra los valores almacenados en la SRam.
- 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
- unsigned int j

4.19.1. Documentación de las funciones

4.19.1.1. void mostrar_banderas (uint32_t *banderas[4]*)

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

Parámetros

banderas[4]	Arreglo que contiene cada 1 de las banderas

Devuelve

No tiene retorno

4.19.1.2. void mostrar_operacion (char * *op*)

Funcion que muestra la operacion que se realizo.

Parámetros

op[5]	variable tipo caracter donde se gusrda el nombre de la operacion realizada

Devuelve

No tiene retorno

4.19.1.3. void mostrar_registros (uint32_t * *registro*)

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

Parámetros

registro[13]	Arreglo que contiene cada 1 de los registros

Devuelve

No tiene retorno

4.19.1.4. void mostrar_SRam (uint8_t * *SRam*)

Funcion que muestra los valores almacenados en la SRam.

Parámetros

SRam[65]	Memoria disponible

Devuelve

No tiene retorno

4.19.2. Documentación de las variables

4.19.2.1. unsigned int i

4.19.2.2. unsigned int j

4.20. Referencia del Archivo registros.h

Funciones

- 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.
- void mostrar_SRam (uint8_t *SRam)
  Funcion que muestra los valores almacenados en la SRam.

4.20.1. Documentación de las funciones

4.20.1.1. void mostrar_banderas (uint32_t *banderas[4]*)

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

Parámetros

banderas[4]	Arreglo que contiene cada 1 de las banderas

Devuelve

No tiene retorno

4.20.1.2. void mostrar_operacion (char * *op*)

Funcion que muestra la operacion que se realizo.

Parámetros

op[5]	variable tipo caracter donde se gusrda el nombre de la operacion realizada

Devuelve

No tiene retorno

4.20.1.3. void mostrar_registros (uint32_t * *registro*)

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

Parámetros

registro[13]	Arreglo que contiene cada 1 de los registros

Devuelve

No tiene retorno

4.20.1.4. void mostrar_SRam (uint8_t * *SRam*)

Funcion que muestra los valores almacenados en la SRam.

Parámetros

SRam[65]	Memoria disponible

Devuelve

No tiene retorno

4.21. Referencia del Archivo test.c