# Cortex -M0

Generado por Doxygen 1.8.7

Domingo, 18 de Octubre de 2015 01:21:08

# Índice general


**1 Cortex-M0** **1**

**2 Índice de estructura de datos** **3**

2.1 Estructura de datos . . . 3

**3 Indice de archivos** **5**

3.1 Lista de archivos . . . 5

**4 Documentación de las estructuras de datos** **7**

4.1 Referencia de la Estructura ins_t . . . 7

4.1.1 Documentación de los campos . . . 7

4.1.1.1 array . . . 7

4.2 Referencia de la Estructura instruction_t . . . 7

4.2.1 Documentación de los campos . . . 7

4.2.1.1 mnemonic . . . 7

4.2.1.2 op1_type . . . 8

4.2.1.3 op1_value . . . 8

4.2.1.4 op2_type . . . 8

4.2.1.5 op2_value . . . 8

4.2.1.6 op3_type . . . 8

4.2.1.7 op3_value . . . 8

4.2.1.8 registers_list . . . 8

4.3 Referencia de la Estructura port_t . . . 8

4.3.1 Documentación de los campos . . . 8

4.3.1.1 DDR . . . 8

4.3.1.2 Interrupts . . . 8

4.3.1.3 PIN . . . 8

4.3.1.4 Pins . . . 8

4.3.1.5 PORT . . . 8

**5 Documentación de archivos** **9**

5.1 Referencia del Archivo branch.c . . . 9

5.1.1 Documentación de los 'defines' . . . 10

5.1.1.1 C . . . 10
5.1.1.2 LR . . . 10
5.1.1.3 N . . . 10
5.1.1.4 PC . . . 10
5.1.1.5 V . . . 10
5.1.1.6 Z . . . 10
5.1.2 Documentación de las funciones . . . 10
5.1.2.1 BAL . . . 10
5.1.2.2 BCC . . . 10
5.1.2.3 BCS . . . 11
5.1.2.4 BEQ . . . 11
5.1.2.5 BGE . . . 11
5.1.2.6 BGT . . . 11
5.1.2.7 BHI . . . 12
5.1.2.8 BL . . . 12
5.1.2.9 BLE . . . 12
5.1.2.10 BLS . . . 12
5.1.2.11 BLT . . . 13
5.1.2.12 BMI . . . 13
5.1.2.13 BNE . . . 13
5.1.2.14 BPL . . . 13
5.1.2.15 BVC . . . 14
5.1.2.16 BVS . . . 14
5.1.2.17 BX . . . 14
5.1.2.18 SB . . . 14
5.2 Referencia del Archivo branch.h . . . 15
5.2.1 Documentación de las funciones . . . 15
5.2.1.1 BAL . . . 15
5.2.1.2 BCC . . . 16
5.2.1.3 BCS . . . 16
5.2.1.4 BEQ . . . 16
5.2.1.5 BGE . . . 16
5.2.1.6 BGT . . . 17
5.2.1.7 BHI . . . 17
5.2.1.8 BL . . . 17
5.2.1.9 BLE . . . 17
5.2.1.10 BLS . . . 18
5.2.1.11 BLT . . . 18
5.2.1.12 BMI . . . 18
5.2.1.13 BNE . . . 18

5.2.1.14 BPL . . . 19
5.2.1.15 BVC . . . 19
5.2.1.16 BVS . . . 19
5.2.1.17 BX . . . 19
5.2.1.18 SB . . . 20
5.3 Referencia del Archivo decoder.c . . . 20
5.3.1 Documentación de las funciones . . . 20
5.3.1.1 countLines . . . 20
5.3.1.2 decodeInstruction . . . 20
5.3.1.3 getInstruction . . . 20
5.3.1.4 readFile . . . 20
5.4 Referencia del Archivo decoder.h . . . 21
5.4.1 Documentación de las funciones . . . 21
5.4.1.1 countLines . . . 21
5.4.1.2 decodeInstruction . . . 21
5.4.1.3 getInstruction . . . 21
5.4.1.4 readFile . . . 21
5.5 Referencia del Archivo flags.c . . . 21
5.5.1 Documentación de las funciones . . . 22
5.5.1.1 BanderaC . . . 22
5.5.1.2 BanderaN . . . 22
5.5.1.3 BanderaV . . . 22
5.5.1.4 BanderaZ . . . 22
5.6 Referencia del Archivo flags.h . . . 22
5.6.1 Documentación de las funciones . . . 23
5.6.1.1 BanderaC . . . 23
5.6.1.2 BanderaN . . . 23
5.6.1.3 BanderaV . . . 23
5.6.1.4 BanderaZ . . . 23
5.7 Referencia del Archivo instrucciones.c . . . 23
5.7.1 Documentación de los 'defines' . . . 25
5.7.1.1 C . . . 25
5.7.1.2 N . . . 25
5.7.1.3 PC . . . 25
5.7.1.4 V . . . 25
5.7.1.5 Z . . . 25
5.7.2 Documentación de las funciones . . . 25
5.7.2.1 ADCS . . . 25
5.7.2.2 ADD . . . 25
5.7.2.3 ADDS . . . 25

5.7.2.4 AND . . . 26
5.7.2.5 ASRS . . . 26
5.7.2.6 BICS . . . 26
5.7.2.7 CMN . . . 26
5.7.2.8 CMP . . . 27
5.7.2.9 EOR . . . 27
5.7.2.10 LSL . . . 27
5.7.2.11 LSLS . . . 27
5.7.2.12 LSR . . . 28
5.7.2.13 LSRS . . . 28
5.7.2.14 MOVS . . . 28
5.7.2.15 MUL . . . 28
5.7.2.16 MVN . . . 29
5.7.2.17 NOP . . . 29
5.7.2.18 ORR . . . 29
5.7.2.19 REV . . . 29
5.7.2.20 REV16 . . . 30
5.7.2.21 REVSH . . . 30
5.7.2.22 ROR . . . 30
5.7.2.23 RSBS . . . 31
5.7.2.24 SUB . . . 31
5.7.2.25 TST . . . 32
5.8 Referencia del Archivo instrucciones.h . . . 32
5.8.1 Documentación de las funciones . . . 33
5.8.1.1 ADCS . . . 33
5.8.1.2 ADD . . . 33
5.8.1.3 ADDS . . . 34
5.8.1.4 AND . . . 34
5.8.1.5 ASR . . . 34
5.8.1.6 ASRS . . . 35
5.8.1.7 BICS . . . 36
5.8.1.8 CMN . . . 36
5.8.1.9 CMP . . . 36
5.8.1.10 EOR . . . 36
5.8.1.11 LSL . . . 37
5.8.1.12 LSLS . . . 37
5.8.1.13 LSR . . . 37
5.8.1.14 LSRS . . . 37
5.8.1.15 MOVS . . . 38
5.8.1.16 MUL . . . 38

5.8.1.17 MVN . . . 38
5.8.1.18 NOP . . . 39
5.8.1.19 ORR . . . 39
5.8.1.20 REV . . . 39
5.8.1.21 REV16 . . . 39
5.8.1.22 REVSH . . . 39
5.8.1.23 ROR . . . 40
5.8.1.24 RSB . . . 40
5.8.1.25 SBC . . . 40
5.8.1.26 SUB . . . 40
5.8.1.27 TST . . . 41
5.9 Referencia del Archivo io.c . . . 41
5.9.1 Documentación de las funciones . . . 41
5.9.1.1 changePinPortA . . . 41
5.9.1.2 changePinPortB . . . 41
5.9.1.3 initIO . . . 41
5.9.1.4 IOAccess . . . 42
5.9.1.5 showFrame . . . 42
5.9.1.6 showPorts . . . 42
5.9.2 Documentación de las variables . . . 42
5.9.2.1 irq . . . 42
5.9.2.2 PORTA . . . 42
5.9.2.3 PORTB . . . 42
5.10 Referencia del Archivo io.h . . . 42
5.10.1 Documentación de los 'defines' . . . 42
5.10.1.1 BLUEBLACK . . . 42
5.10.1.2 HIGH . . . 42
5.10.1.3 LOW . . . 42
5.10.1.4 Read . . . 43
5.10.1.5 REDBLACK . . . 43
5.10.1.6 WHITEBLACK . . . 43
5.10.1.7 Write . . . 43
5.10.1.8 XINIT . . . 43
5.10.1.9 YINIT . . . 43
5.10.2 Documentación de las funciones . . . 43
5.10.2.1 changePinPortA . . . 43
5.10.2.2 changePinPortB . . . 43
5.10.2.3 initIO . . . 43
5.10.2.4 IOAccess . . . 43
5.10.2.5 showFrame . . . 43

5.10.2.6 showPorts . . . 43
5.11 Referencia del Archivo main.c . . . 43
5.11.1 Documentación de los 'defines' . . . 43
5.11.1.1 PC . . . 43
5.11.2 Documentación de las funciones . . . 43
5.11.2.1 main . . . 44
5.12 Referencia del Archivo ports.c . . . 44
5.13 Referencia del Archivo README.md . . . 44
5.14 Referencia del Archivo SRAM.c . . . 44
5.14.1 Documentación de las funciones . . . 44
5.14.1.1 BitCount . . . 44
5.14.1.2 LDR . . . 44
5.14.1.3 LDRB . . . 44
5.14.1.4 LDRH . . . 44
5.14.1.5 LDRSB . . . 45
5.14.1.6 LDRSH . . . 45
5.14.1.7 POP . . . 45
5.14.1.8 PUSH . . . 45
5.14.1.9 STR . . . 45
5.14.1.10 STRB . . . 45
5.14.1.11 STRH . . . 45
5.15 Referencia del Archivo SRAM.h . . . 45
5.15.1 Documentación de los 'defines' . . . 46
5.15.1.1 Mema . . . 46
5.15.1.2 PC . . . 46
5.15.2 Documentación de las funciones . . . 46
5.15.2.1 BitCount . . . 46
5.15.2.2 LDR . . . 46
5.15.2.3 LDRB . . . 46
5.15.2.4 LDRH . . . 46
5.15.2.5 LDRSB . . . 46
5.15.2.6 LDRSH . . . 46
5.15.2.7 POP . . . 46
5.15.2.8 PUSH . . . 47
5.15.2.9 STR . . . 47
5.15.2.10 STRB . . . 47
5.15.2.11 STRH . . . 47
5.16 Referencia del Archivo test.c . . . 47
5.17 Referencia del Archivo visualizacion.c . . . 47
5.17.1 Documentación de los 'defines' . . . 47

5.17.1.1 PC . . . . . 47
5.17.2 Documentación de las funciones . . . . . 47
5.17.2.1 visualizacion_registro . . . . . 47
5.18 Referencia del Archivo visualizacion.h . . . . . 48
5.18.1 Documentación de las funciones . . . . . 48
5.18.1.1 visualizacion_registro . . . . . 48

# Capítulo 1

# Cortex-M0

Emulador Procesador Cortex M0 en C Documentación: Nicolás Arias. Desarrrollador: Andrés Felipe Sánchez

# Capítulo 2

# Índice de estructura de datos

## 2.1. Estructura de datos

Lista de estructuras con una breve descripción:

ins_t . . . . . 7
instruction_t . . . . . 7
port_t . . . . . 8

# Capítulo 3

# Indice de archivos

## 3.1. Lista de archivos

Lista de todos los archivos con descripciones breves:


branch.c . . . 9
branch.h . . . 15
decoder.c . . . 20
decoder.h . . . 21
flags.c . . . 21
flags.h . . . 22
instrucciones.c . . . 23
instrucciones.h . . . 32
io.c . . . 41
io.h . . . 42
main.c . . . 43
ports.c . . . 44
SRAM.c . . . 44
SRAM.h . . . 45
test.c . . . 47
visualizacion.c . . . 47
visualizacion.h . . . 48

# Capítulo 4

# Documentación de las estructuras de datos

## 4.1. Referencia de la Estructura ins_t

`#include <decoder.h>`

### Campos de datos

- char ** array

#### 4.1.1. Documentación de los campos

##### 4.1.1.1. char** array

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

- decoder.h

## 4.2. 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]

#### 4.2.1. Documentación de los campos

##### 4.2.1.1. char mnemonic[10]

#### 4.2.1.2. char op1_type

#### 4.2.1.3. uint32_t op1_value

#### 4.2.1.4. char op2_type

#### 4.2.1.5. uint32_t op2_value

#### 4.2.1.6. char op3_type

#### 4.2.1.7. uint32_t op3_value

#### 4.2.1.8. uint8_t registers_list[16]

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

- decoder.h

## 4.3. 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

### 4.3.1. Documentación de los campos

#### 4.3.1.1. uint8_t DDR

#### 4.3.1.2. uint8_t Interrupts

#### 4.3.1.3. uint8_t PIN

#### 4.3.1.4. uint8_t Pins

#### 4.3.1.5. uint8_t PORT

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

- io.h

# Capítulo 5

# Documentación de archivos

## 5.1. Referencia del Archivo branch.c

```
#include "branch.h"
#include "flags.h"
```

### 'defines'

- #define PC 15
- #define LR 14
- #define N 0
- #define Z 1
- #define C 2
- #define V 3

### Funciones

- void BEQ (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que realiza un salto si es igual.*
- void BNE (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que reliza un salto si es diferente.*
- void BCS (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que reliza un salto si es mayor o igual (sin signo)*
- void BCC (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcin que reliza un salto si es menor (sin signo)*
- void BMI (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que realiza un salto si es negativo.*
- void BPL (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que realiza un salto si es positivo.*
- void BVS (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que realiza un salto si hay sobreflujo.*
- void BVC (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que realiza un salto si no hay sobreflujo.*
- void BHI (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que realiza un salto si es mayor (sin signo)*
- void BLS (uint32_t *B, uint32_t *R, uint32_t Posicion)

*Funcion que realiza un salto si es menor o igual (sin signo)*

- void BGE (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que genera un salto si es mayor o igual (con signo)*
- void BLT (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que realiza un salto si es menor (con signo)*
- void BGT (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que realiza un salto si es mayor (con signo)*
- void BLE (uint32_t *B, uint32_t *R, uint32_t Posicion)

  *Funcion que realizaun un salto si es menor o igual (con signo)*
- void BAL (uint32_t *R, uint32_t Posicion)

  *Funcion que realiza un salto sin condicion, de cualquier forma.*
- void BL (uint32_t *R, uint32_t Posicion)

  *Funcion que realiza un salto guardando posicion instruccion siguiente en LR.*
- void BX (uint32_t *R)

  *Funcion que realiza un salto a la posicion guardada en LR.*
- void SB (uint32_t *R, uint32_t Posicion)

### 5.1.1. Documentación de los 'defines'

#### 5.1.1.1. #define C 2

#### 5.1.1.2. #define LR 14

#### 5.1.1.3. #define N 0

#### 5.1.1.4. #define PC 15

#### 5.1.1.5. #define V 3

#### 5.1.1.6. #define Z 1

### 5.1.2. Documentación de las funciones

#### 5.1.2.1. void BAL ( uint32_t * *R*, uint32_t *Posicion* )

Funcion que realiza un salto sin condicion, de cualquier forma.

**Parámetros**

| | |
|---|---|
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void.

#### 5.1.2.2. void BCC ( uint32_t * *B*, uint32_t * *R*, uint32_t *Posicion* )

Funcin que reliza un salto si es menor (sin signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.3. void BCS ( uint32_t * *B*, uint32_t * *R*, uint32_t *Posicion* )

Funcion que reliza un salto si es mayor o igual (sin signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.4. void BEQ ( uint32_t * *B*, uint32_t * *R*, uint32_t *Posicion* )

Funcion que realiza un salto si es igual.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.5. void BGE ( uint32_t * *B*, uint32_t * *R*, uint32_t *Posicion* )

Funcion que genera un salto si es mayor o igual (con signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.6. void BGT ( uint32_t * *B*, uint32_t * *R*, uint32_t *Posicion* )

Funcion que realiza un salto si es mayor (con signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void.

#### 5.1.2.7. void BHI ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es mayor (sin signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.8. void BL ( uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto guardando posicion instruccion siguiente en LR.

**Parámetros**

| | |
|---|---|
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.9. void BLE ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realizaun un salto si es menor o igual (con signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.10. void BLS ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es menor o igual (sin signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.11. void BLT ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es menor (con signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.12. void BMI ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es negativo.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.13. void BNE ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que reliza un salto si es diferente.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.14. void BPL ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es positivo.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.15. void BVC ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si no hay sobreflujo.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void.

#### 5.1.2.16. void BVS ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si hay sobreflujo.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.1.2.17. void BX ( uint32_t * *R* )

Funcion que realiza un salto a la posicion guardada en LR.

**Parámetros**

| | |
|---|---|
| *R | Puntero a el registro PC |

**Devuelve**

void

#### 5.1.2.18. void SB ( uint32_t * *R,* uint32_t *Posicion* )

\ brief Funcion que incrementa la posicion en 2 \ param PC puntero a el registro PC \ param Posicion siguiente posicion \ return void

## 5.2. Referencia del Archivo branch.h

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

### Funciones

- void BEQ (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto si es igual.*
- void BNE (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que reliza un salto si es diferente.*
- void BCS (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que reliza un salto si es mayor o igual (sin signo)*
- void BCC (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcin que reliza un salto si es menor (sin signo)*
- void BMI (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto si es negativo.*
- void BPL (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto si es positivo.*
- void BVS (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto si hay sobreflujo.*
- void BVC (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto si no hay sobreflujo.*
- void BHI (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto si es mayor (sin signo)*
- void BLS (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto si es menor o igual (sin signo)*
- void BGE (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que genera un salto si es mayor o igual (con signo)*
- void BLT (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto si es menor (con signo)*
- void BGT (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto si es mayor (con signo)*
- void BLE (uint32_t *B, uint32_t *R, uint32_t Posicion)
  *Funcion que realizaun un salto si es menor o igual (con signo)*
- void BAL (uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto sin condicion, de cualquier forma.*
- void BL (uint32_t *R, uint32_t Posicion)
  *Funcion que realiza un salto guardando posicion instruccion siguiente en LR.*
- void BX (uint32_t *R)
  *Funcion que realiza un salto a la posicion guardada en LR.*
- void SB (uint32_t *R, uint32_t Posicion)

### 5.2.1. Documentación de las funciones

#### 5.2.1.1. void BAL ( uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto sin condicion, de cualquier forma.

**Parámetros**

| | |
|---|---|
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void.

#### 5.2.1.2. void BCC ( uint32_t * *B*, uint32_t * *R*, uint32_t *Posicion* )

Funcin que reliza un salto si es menor (sin signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.3. void BCS ( uint32_t * *B*, uint32_t * *R*, uint32_t *Posicion* )

Funcion que reliza un salto si es mayor o igual (sin signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.4. void BEQ ( uint32_t * *B*, uint32_t * *R*, uint32_t *Posicion* )

Funcion que realiza un salto si es igual.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.5. void BGE ( uint32_t * *B*, uint32_t * *R*, uint32_t *Posicion* )

Funcion que genera un salto si es mayor o igual (con signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.6. void BGT ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es mayor (con signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void.

#### 5.2.1.7. void BHI ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es mayor (sin signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.8. void BL ( uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto guardando posicion instruccion siguiente en LR.

**Parámetros**

| | |
|---|---|
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.9. void BLE ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realizaun un salto si es menor o igual (con signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.10. void BLS ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es menor o igual (sin signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.11. void BLT ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es menor (con signo)

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.12. void BMI ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es negativo.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.13. void BNE ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que reliza un salto si es diferente.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.14. void BPL ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si es positivo.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.15. void BVC ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si no hay sobreflujo.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void.

#### 5.2.1.16. void BVS ( uint32_t * *B,* uint32_t * *R,* uint32_t *Posicion* )

Funcion que realiza un salto si hay sobreflujo.

**Parámetros**

| | |
|---|---|
| *Banderas* | Puntero para el registro de las banderas |
| *PC* | Puntero a el registro PC |
| *Posicion* | Siguiente posicion |

**Devuelve**

void

#### 5.2.1.17. void BX ( uint32_t * *R* )

Funcion que realiza un salto a la posicion guardada en LR.

**Parámetros**

| | |
|---|---|
| *∗R* | Puntero a el registro PC |

**Devuelve**

void

#### 5.2.1.18. void SB ( uint32_t ∗ *R,* uint32_t *Posicion* )

\ brief Funcion que incrementa la posicion en 2 \ param PC puntero a el registro PC \ param Posicion siguiente posicion \ return void

## 5.3. Referencia del Archivo decoder.c

```
#include "decoder.h"
#include "SRAM.h"
#include "flags.h"
#include "branch.h"
```

### Funciones

- void decodeInstruction (instruction_t instruction, uint32_t ∗R, uint32_t ∗B)
- instruction_t getInstruction (char ∗instStr)

  *Obtiene la instrucción separada por partes.*
- int readFile (char ∗filename, ins_t ∗instructions)
- int countLines (FILE ∗fp)

### 5.3.1. Documentación de las funciones

#### 5.3.1.1. int countLines ( FILE ∗ *fp* )

#### 5.3.1.2. void decodeInstruction ( instruction_t *instruction,* uint32_t ∗ *R,* uint32_t ∗ *B* )

#### 5.3.1.3. instruction_t getInstruction ( char ∗ *instStr* )

Obtiene la instrucción separada por partes.

**Parámetros**

| | |
|---|---|
| *instrStr* | cadena que contiene la instrucción. |

**Devuelve**

instruction_t la instrucción separada por partes.

#### 5.3.1.4. int readFile ( char ∗ *filename,* ins_t ∗ *instructions* )

## 5.4. Referencia del Archivo decoder.h

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

### Estructuras de datos

- struct ins_t
- struct instruction_t

### Funciones

- void decodeInstruction (instruction_t instruction, uint32_t *R, uint32_t *B)
- instruction_t getInstruction (char *instStr)

  *Obtiene la instrucción separada por partes.*
- int readFile (char *filename, ins_t *instructions)
- int countLines (FILE *fp)

### 5.4.1. Documentación de las funciones

#### 5.4.1.1. int countLines ( FILE * *fp* )

#### 5.4.1.2. void decodeInstruction ( instruction_t *instruction,* uint32_t * *R,* uint32_t * *B* )

#### 5.4.1.3. instruction_t getInstruction ( char * *instStr* )

Obtiene la instrucción separada por partes.

**Parámetros**

| | |
|---|---|
| *instrStr* | cadena que contiene la instrucción. |

**Devuelve**

instruction_t la instrucción separada por partes.

#### 5.4.1.4. int readFile ( char * *filename,* ins_t * *instructions* )

## 5.5. Referencia del Archivo flags.c

```
#include "flags.h"
```

### Funciones

- void BanderaN (uint32_t ∗Rd, uint32_t ∗B)
- void BanderaZ (uint32_t ∗Rd, uint32_t ∗B)
- void BanderaC (uint32_t ∗Rd, uint32_t ∗B)
- void BanderaV (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗B)

### 5.5.1. Documentación de las funciones

#### 5.5.1.1. void BanderaC ( uint32_t ∗ *Rd,* uint32_t ∗ *B* )

para modificar la bandera de carry (C)

**Parámetros**

| | |
|---|---|
| *Rd* | Variable que contiene el resultado de las operaciones |
| *B* | Punntero en el cual se almacenaran las banderas. |

#### 5.5.1.2. void BanderaN ( uint32_t ∗ *Rd,* uint32_t ∗ *B* )

para modificar la bandera de negativo (N)

**Parámetros**

| | |
|---|---|
| *Rd* | Variable que contiene el resultado de las operaciones |
| *B* | Punntero en el cual se almacenaran las banderas. |

#### 5.5.1.3. void BanderaV ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *B* )

para modificar la bandera de sobre-flujo (V)

**Parámetros**

| | |
|---|---|
| *Rd* | Variable que contiene el resultado de las operaciones |
| *Rn* | Variable con el primer operando. |
| *Rm* | Variable con el segundo operando. |
| *B* | Punntero en el cual se almacenaran las banderas. |

#### 5.5.1.4. void BanderaZ ( uint32_t ∗ *Rd,* uint32_t ∗ *B* )

para modificar la bandera de cero (Z)

**Parámetros**

| | |
|---|---|
| *Rd* | Variable que contiene el resultado de las operaciones |
| *B* | Punntero en el cual se almacenaran las banderas. |

## 5.6. Referencia del Archivo flags.h

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

### Funciones

- void BanderaN (uint32_t *Rd, uint32_t *B)
- void BanderaZ (uint32_t *Rd, uint32_t *B)
- void BanderaC (uint32_t *Rd, uint32_t *B)
- void BanderaV (uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint32_t *B)

### 5.6.1. Documentación de las funciones

#### 5.6.1.1. void BanderaC ( uint32_t * *Rd,* uint32_t * *B* )

para modificar la bandera de carry (C)

**Parámetros**

| | |
|---|---|
| *Rd* | Variable que contiene el resultado de las operaciones |
| *B* | Punntero en el cual se almacenaran las banderas. |

#### 5.6.1.2. void BanderaN ( uint32_t * *Rd,* uint32_t * *B* )

para modificar la bandera de negativo (N)

**Parámetros**

| | |
|---|---|
| *Rd* | Variable que contiene el resultado de las operaciones |
| *B* | Punntero en el cual se almacenaran las banderas. |

#### 5.6.1.3. void BanderaV ( uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t * *B* )

para modificar la bandera de sobre-flujo (V)

**Parámetros**

| | |
|---|---|
| *Rd* | Variable que contiene el resultado de las operaciones |
| *Rn* | Variable con el primer operando. |
| *Rm* | Variable con el segundo operando. |
| *B* | Punntero en el cual se almacenaran las banderas. |

#### 5.6.1.4. void BanderaZ ( uint32_t * *Rd,* uint32_t * *B* )

para modificar la bandera de cero (Z)

**Parámetros**

| | |
|---|---|
| *Rd* | Variable que contiene el resultado de las operaciones |
| *B* | Punntero en el cual se almacenaran las banderas. |

## 5.7. Referencia del Archivo instrucciones.c

```
#include "instrucciones.h"
#include "flags.h"
```

### 'defines'

- #define PC 15
- #define N 0
- #define Z 1
- #define C 2
- #define V 3

## Funciones

- void ADD (uint32_t *Rd, uint32_t Rn, uint32_t Num, uint32_t *R, uint32_t *B)

  *Funcion para realizar la suma de un registro y un valor.*
- void ADDS (uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint32_t *R, uint32_t *B)

  *Funcion para realizar la suma con dos registros.*
- void ORR (uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint32_t *R, uint32_t *B)

  *Funcion para realizar ORR con dos registros.*
- void EOR (uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint32_t *R, uint32_t *B)

  *Funcion para realizar EOR con dos registros.*
- void MOVS (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)

  *Funcion para realizar MOV (reemplazar el valor de un registro en otro)*
- void AND (uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint32_t *R, uint32_t *B)

  *Funcion para realizar AND dos registros.*
- void TST (uint32_t Rn, uint32_t Rm, uint32_t *R, uint32_t *B)

  *Funcion que hace una AND bit a bit, pero solo modifica las *B.*
- void SUB (uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint32_t *R, uint32_t *B)

  *Funcion para realizar la resta entre un registro y un valor.*
- void LSL (uint32_t *Rd, uint32_t Rn, uint32_t Num, uint32_t *R, uint32_t *B)

  *Funcion desplazamiento hacia la izquierda ingresando un numero a sumar.*
- void LSLS (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)
- void LSR (uint32_t *Rd, uint32_t Rn, uint32_t Num, uint32_t *R, uint32_t *B)

  *Funcion desplazamiento hacia la derecha con un numero a sumar.*
- void LSRS (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)

  *Funcion desplazamiento hacia la Derecha con dos registros.*
- void ROR (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)

  *Funcion para rotar.*
- void ASRS (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)

  *Funcion para despalazar aritmeticamente hacia la derecha con dos registros.*
- void REV (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)

  *Funcion para cambiar el orden de los bits.*
- void REV16 (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)

  *Funcion para modificar la posicion de grupos de 16 bits.*
- void REVSH (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)

  *Funcion para modificar la posicion de a ocho bits.*
- void BICS (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)

  *Funcion que hace una and de 1 bit y con Rm negado.*
- void MVN (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)

  *funcion para hacer el complemento a un registro.*
- void RSBS (uint32_t *Rd, uint32_t Rn, uint32_t *R, uint32_t *B)
- void NOP (uint32_t *R)

  *Funcion que hace un retardo.*
- void CMN (uint32_t Rn, uint32_t Rm, uint32_t *R, uint32_t *B)

*Funcion que modifica las ∗B sin guardar el resultado.*

- void MUL (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion Multiplica dos registros, sin signo y almacena los 32 bits menos significativos del resultado en el regsitro de destino.*
- void CMP (uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion que modifica las ∗B comparando los registros.*
- void ADCS (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion que realiza una suma entre dos registros mas el valor de carry.*

### 5.7.1. Documentación de los 'defines'

#### 5.7.1.1. #define C 2

#### 5.7.1.2. #define N 0

#### 5.7.1.3. #define PC 15

#### 5.7.1.4. #define V 3

#### 5.7.1.5. #define Z 1

### 5.7.2. Documentación de las funciones

#### 5.7.2.1. void ADCS ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion que realiza una suma entre dos registros mas el valor de carry.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro que almacena el resultado |
| *Rn* | Primer valor a operar |
| *Rm* | Segundo valor a operar |
| *R* | el vector con los registros. |
| *B* | vector que contiene las banderas. |

#### 5.7.2.2. void ADD ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Num,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para realizar la suma de un registro y un valor.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado. |
| *Rn* | Registro de ingreso para sumar. |
| *Num* | Valor a sumar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.7.2.3. void ADDS ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para realizar la suma con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado. |
| *Rn* | Primer registro de ingreso a sumar. |
| *Rm* | Segundo registro de ingreso a sumar. |
| ∗*B* | Es el puntero que almacenas los resultados de las baderas |

**Devuelve**

void

#### 5.7.2.4. void AND ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para realizar AND dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado |
| *Rn* | Primer registro de ingreso para AND |
| *Rm* | Segundo registro de ingreso para AND |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.7.2.5. void ASRS ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para despalazar aritmeticamente hacia la derecha con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| *Rn* | numero a trasladar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.6. void BICS ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion que hace una and de 1 bit y con Rm negado.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion |
| *Rn* | Registro a negar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.7. void CMN ( uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion que modifica las ∗B sin guardar el resultado.

**Parámetros**

| | |
|---|---|
| *Rn* | Primer registro a operar. |
| *Rm* | Segundo registro a operar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.8. void CMP ( uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion que modifica las ∗B comparando los registros.

**Parámetros**

| | |
|---|---|
| *Rn* | Primer registro a comparar. |
| *Rm* | Segundo regsitro a comparar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.9. void EOR ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para realizar EOR con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado. |
| *Rn* | Primer registro de ingreso para EOR |
| *Rm* | Segundo registro de ingreso para EOR. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.7.2.10. void LSL ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Num,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion desplazamiento hacia la izquierda ingresando un numero a sumar.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| *Rn* | Valor del registro a sumar. |
| *Num* | Valor a sumar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.11. void LSLS ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion desplazamiento hacia la izquierda con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| *Rn* | Valor del registro a sumar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.12. void LSR ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Num,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion desplazamiento hacia la derecha con un numero a sumar.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion |
| *Rn* | Valor del registro a sumar. |
| *Num* | Valor a sumar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.7.2.13. void LSRS ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion desplazamiento hacia la Derecha con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| *Rn* | Valor del registro a sumar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.14. void MOVS ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para realizar MOV (reemplazar el valor de un registro en otro)

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado |
| *Rn* | Registro de ingreso para MOV (posicion en la cual se va a mover el valor del registro) |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.7.2.15. void MUL ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion Multiplica dos registros, sin signo y almacena los 32 bits menos significativos del resultado en el regsitro de destino.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado de la operacion. |
| *Rn* | Primer valor del reistro a multiplicar. |
| *Rm* | Segundo valor del registro a multiplicar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.16. void MVN ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

funcion para hacer el complemento a un registro.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.17. void NOP ( uint32_t ∗ *R* )

Funcion que hace un retardo.

**Devuelve**

void.

#### 5.7.2.18. void ORR ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para realizar ORR con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado |
| *Rn* | Primer registro de ingreso para ORR. |
| *Rm* | Segundo registro de ingreso para ORR. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.7.2.19. void REV ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para cambiar el orden de los bits.

**Parámetros**

| | |
|---|---|
| *Rb* | Registro a resultante. |
| *Rn* | valor a modificar. |
| *R* | Puntero que almacena los registros. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.20. void REV16 ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para modificar la posicion de grupos de 16 bits.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro resultante. |
| *Rn* | Registro a modificar. |
| *R* | Puntero que almacena los registros. |
| ∗*B* | Es el puntero que almacena los resultados de las banderas |

**Devuelve**

void.

#### 5.7.2.21. void REVSH ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para modificar la posicion de a ocho bits.

**Parámetros**

| | |
|---|---|
| *Rd* | Numero a modificar. |
| *Rn* | Registro a modificar. |
| *R* | Puntero que almacena los registros. |
| ∗*B* | Es el puntero que almacenas los resultados de las ∗B |

**Devuelve**

void

#### 5.7.2.22. void ROR ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para rotar.

**Parámetros**

| | |
|---|---|
| *Rdn* | Resultado de la operacion. |
| *Rn* | Numero a rotar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

5.7.2.23. void RSBS ( uint32_t * *Rd,* uint32_t *Rn,* uint32_t * *R,* uint32_t * *B* )

5.7.2.24. void SUB ( uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t * *R,* uint32_t * *B* )

Funcion para realizar la resta entre un registro y un valor.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado |
| *Rn* | Primer registro de ingreso a sumar |
| *Rm* | Segundo registro de ingreso a sumar |
| *∗B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.7.2.25. void TST ( uint32_t *Rn*, uint32_t *Rm*, uint32_t ∗ *R*, uint32_t ∗ *B* )

Funcion que hace una AND bit a bit, pero solo modifica las ∗B.

**Parámetros**

| | |
|---|---|
| *Rn* | Primer valor ingresado para hacer la operacion. |
| *Rm* | Segundo valor ingresado para hacer la operacion. |
| *∗B* | Es el puntero que almacenas los resultados de las ∗B |

**Devuelve**

void.

## 5.8. Referencia del Archivo instrucciones.h

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

### Funciones

- void ADD (uint32_t ∗Rd, uint32_t Rn, uint32_t Num, uint32_t ∗R, uint32_t ∗B)

  *Funcion para realizar la suma de un registro y un valor.*
- void ADDS (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion para realizar la suma con dos registros.*
- void ORR (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion para realizar ORR con dos registros.*
- void EOR (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion para realizar EOR con dos registros.*
- void MOVS (uint32_t ∗Rd, uint32_t Rn, uint32_t ∗R, uint32_t ∗B)

  *Funcion para realizar MOV (reemplazar el valor de un registro en otro)*
- void AND (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion para realizar AND dos registros.*
- void SUB (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion para realizar la resta entre un registro y un valor.*
- void LSL (uint32_t ∗Rd, uint32_t Rn, uint32_t Num, uint32_t ∗R, uint32_t ∗B)

  *Funcion desplazamiento hacia la izquierda ingresando un numero a sumar.*
- void LSLS (uint32_t ∗Rd, uint32_t Rn, uint32_t ∗R, uint32_t ∗B)
- void LSR (uint32_t ∗Rd, uint32_t Rn, uint32_t Num, uint32_t ∗R, uint32_t ∗B)

  *Funcion desplazamiento hacia la derecha con un numero a sumar.*
- void LSRS (uint32_t ∗Rd, uint32_t Rn, uint32_t ∗R, uint32_t ∗B)

*Funcion desplazamiento hacia la Derecha con dos registros.*

- void ROR (uint32_t ∗Rd, uint32_t Rn, uint32_t ∗R, uint32_t ∗B)

  *Funcion para rotar.*
- void ASR (uint32_t ∗Rd, uint32_t Rn, uint32_t Num, uint32_t ∗R, uint32_t ∗B)

  *Funcion para desplazar aritmeticamente hacia la derecha con un registro y y un numero.*
- void ASRS (uint32_t ∗Rd, uint32_t Rn, uint32_t ∗R, uint32_t ∗B)

  *Funcion para despalazar aritmeticamente hacia la derecha con dos registros.*
- void BICS (uint32_t ∗Rd, uint32_t Rn, uint32_t ∗R, uint32_t ∗B)

  *Funcion que hace una and de 1 bit y con Rm negado.*
- void CMN (uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion que modifica las ∗B sin guardar el resultado.*
- void CMP (uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion que modifica las ∗B comparando los registros.*
- void MUL (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion Multiplica dos registros, sin signo y almacena los 32 bits menos significativos del resultado en el regsitro de destino.*
- void MVN (uint32_t ∗Rd, uint32_t Rn, uint32_t ∗R, uint32_t ∗B)

  *funcion para hacer el complemento a un registro.*
- void NOP (uint32_t ∗R)

  *Funcion que hace un retardo.*
- void REV (uint32_t ∗Rd, uint32_t Rn, uint32_t ∗R, uint32_t ∗B)

  *Funcion para cambiar el orden de los bits.*
- void REV16 (uint32_t ∗Rd, uint32_t Rn, uint32_t ∗R, uint32_t ∗B)

  *Funcion para modificar la posicion de grupos de 16 bits.*
- void REVSH (uint32_t ∗Rd, uint32_t Rn, uint32_t ∗R, uint32_t ∗B)

  *Funcion para modificar la posicion de a ocho bits.*
- void RSB (uint32_t ∗Rd, uint32_t ∗R, uint32_t ∗B)

  *Funcion para obtener el complemento a dos de un numero.*
- void SBC (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion para restar con Carry.*
- void TST (uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion que hace una AND bit a bit, pero solo modifica las ∗B.*
- void ADCS (uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint32_t ∗R, uint32_t ∗B)

  *Funcion que realiza una suma entre dos registros mas el valor de carry.*

## 5.8.1. Documentación de las funciones

### 5.8.1.1. void ADCS ( uint32_t ∗ *Rd*, uint32_t *Rn*, uint32_t *Rm*, uint32_t ∗ *R*, uint32_t ∗ *B* )

Funcion que realiza una suma entre dos registros mas el valor de carry.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro que almacena el resultado |
| *Rn* | Primer valor a operar |
| *Rm* | Segundo valor a operar |
| *R* | el vector con los registros. |
| *B* | vector que contiene las banderas. |

### 5.8.1.2. void ADD ( uint32_t ∗ *Rd*, uint32_t *Rn*, uint32_t *Num*, uint32_t ∗ *R*, uint32_t ∗ *B* )

Funcion para realizar la suma de un registro y un valor.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado. |
| *Rn* | Registro de ingreso para sumar. |
| *Num* | Valor a sumar. |
| $*B$ | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.8.1.3. void ADDS ( uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t * *R,* uint32_t * *B* )

Funcion para realizar la suma con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado. |
| *Rn* | Primer registro de ingreso a sumar. |
| *Rm* | Segundo registro de ingreso a sumar. |
| $*B$ | Es el puntero que almacenas los resultados de las baderas |

**Devuelve**

void

#### 5.8.1.4. void AND ( uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t * *R,* uint32_t * *B* )

Funcion para realizar AND dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado |
| *Rn* | Primer registro de ingreso para AND |
| *Rm* | Segundo registro de ingreso para AND |
| $*B$ | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.8.1.5. void ASR ( uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Num,* uint32_t * *R,* uint32_t * *B* )

Funcion para desplazar aritmeticamente hacia la derecha con un registro y y un numero.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| *Rn* | Valor del registro a sumar. |
| *Num* | Valor a sumar. |
| $*B$ | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.6. void ASRS ( uint32_t * *Rd,* uint32_t *Rn,* uint32_t * *R,* uint32_t * *B* )

Funcion para despalazar aritmeticamente hacia la derecha con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| *Rn* | numero a trasladar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.7. void BICS ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion que hace una and de 1 bit y con Rm negado.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion |
| *Rn* | Registro a negar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.8. void CMN ( uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion que modifica las ∗B sin guardar el resultado.

**Parámetros**

| | |
|---|---|
| *Rn* | Primer registro a operar. |
| *Rm* | Segundo registro a operar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.9. void CMP ( uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion que modifica las ∗B comparando los registros.

**Parámetros**

| | |
|---|---|
| *Rn* | Primer registro a comparar. |
| *Rm* | Segundo regsitro a comparar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.10. void EOR ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para realizar EOR con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado. |
| *Rn* | Primer registro de ingreso para EOR |
| *Rm* | Segundo registro de ingreso para EOR. |
| *∗B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.8.1.11. void LSL ( uint32_t ∗ *Rd*, uint32_t *Rn*, uint32_t *Num*, uint32_t ∗ *R*, uint32_t ∗ *B* )

Funcion desplazamiento hacia la izquierda ingresando un numero a sumar.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| *Rn* | Valor del registro a sumar. |
| *Num* | Valor a sumar. |
| *∗B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.12. void LSLS ( uint32_t ∗ *Rd*, uint32_t *Rn*, uint32_t ∗ *R*, uint32_t ∗ *B* )

Funcion desplazamiento hacia la izquierda con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| *Rn* | Valor del registro a sumar. |
| *∗B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.13. void LSR ( uint32_t ∗ *Rd*, uint32_t *Rn*, uint32_t *Num*, uint32_t ∗ *R*, uint32_t ∗ *B* )

Funcion desplazamiento hacia la derecha con un numero a sumar.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion |
| *Rn* | Valor del registro a sumar. |
| *Num* | Valor a sumar. |
| *∗B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.8.1.14. void LSRS ( uint32_t ∗ *Rd*, uint32_t *Rn*, uint32_t ∗ *R*, uint32_t ∗ *B* )

Funcion desplazamiento hacia la Derecha con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| *Rn* | Valor del registro a sumar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.15. void MOVS ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para realizar MOV (reemplazar el valor de un registro en otro)

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado |
| *Rn* | Registro de ingreso para MOV (posicion en la cual se va a mover el valor del registro) |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.8.1.16. void MUL ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion Multiplica dos registros, sin signo y almacena los 32 bits menos significativos del resultado en el regsitro de destino.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado de la operacion. |
| *Rn* | Primer valor del reistro a multiplicar. |
| *Rm* | Segundo valor del registro a multiplicar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.17. void MVN ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

funcion para hacer el complemento a un registro.

**Parámetros**

| | |
|---|---|
| *Rd* | Resultado de la operacion. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.18. void NOP ( uint32_t * R )

Funcion que hace un retardo.

**Devuelve**

void.

#### 5.8.1.19. void ORR ( uint32_t * Rd, uint32_t Rn, uint32_t Rm, uint32_t * R, uint32_t * B )

Funcion para realizar ORR con dos registros.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado |
| *Rn* | Primer registro de ingreso para ORR. |
| *Rm* | Segundo registro de ingreso para ORR. |
| **B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.8.1.20. void REV ( uint32_t * Rd, uint32_t Rn, uint32_t * R, uint32_t * B )

Funcion para cambiar el orden de los bits.

**Parámetros**

| | |
|---|---|
| *Rb* | Registro a resultante. |
| *Rn* | valor a modificar. |
| *R* | Puntero que almacena los registros. |
| **B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.21. void REV16 ( uint32_t * Rd, uint32_t Rn, uint32_t * R, uint32_t * B )

Funcion para modificar la posicion de grupos de 16 bits.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro resultante. |
| *Rn* | Registro a modificar. |
| *R* | Puntero que almacena los registros. |
| **B* | Es el puntero que almacena los resultados de las banderas |

**Devuelve**

void.

#### 5.8.1.22. void REVSH ( uint32_t * Rd, uint32_t Rn, uint32_t * R, uint32_t * B )

Funcion para modificar la posicion de a ocho bits.

**Parámetros**

| | |
|---|---|
| *Rd* | Numero a modificar. |
| *Rn* | Registro a modificar. |
| *R* | Puntero que almacena los registros. |
| ∗*B* | Es el puntero que almacenas los resultados de las ∗B |

**Devuelve**

void

#### 5.8.1.23. void ROR ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para rotar.

**Parámetros**

| | |
|---|---|
| *Rdn* | Resultado de la operacion. |
| *Rn* | Numero a rotar. |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.8.1.24. void RSB ( uint32_t ∗ *Rd,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para obtener el complemento a dos de un numero.

**Parámetros**

| | |
|---|---|
| *Rb* | Registro al cual se le hara el complemento. |
| ∗*B* | Es el puntero que almacenas los resultados de las ∗B |

**Devuelve**

void.

#### 5.8.1.25. void SBC ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para restar con Carry.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado de la operacion. |
| *Rn* | Primer valor a operar |
| *Rm* | Segundo valor a operar |
| ∗*B* | Es el puntero que almacenas los resultados de las ∗B |

**Devuelve**

void.

#### 5.8.1.26. void SUB ( uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint32_t ∗ *R,* uint32_t ∗ *B* )

Funcion para realizar la resta entre un registro y un valor.

**Parámetros**

| | |
|---|---|
| *Rd* | Registro para el resultado |
| *Rn* | Primer registro de ingreso a sumar |
| *Rm* | Segundo registro de ingreso a sumar |
| ∗*B* | Es el puntero que almacenas los resultados de las banderas |

**Devuelve**

void

#### 5.8.1.27. void TST ( uint32_t *Rn*, uint32_t *Rm*, uint32_t ∗ *R*, uint32_t ∗ *B* )

Funcion que hace una AND bit a bit, pero solo modifica las ∗B.

**Parámetros**

| | |
|---|---|
| *Rn* | Primer valor ingresado para hacer la operacion. |
| *Rm* | Segundo valor ingresado para hacer la operacion. |
| ∗*B* | Es el puntero que almacenas los resultados de las ∗B |

**Devuelve**

void.

## 5.9. 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]

### 5.9.1. Documentación de las funciones

#### 5.9.1.1. void changePinPortA ( uint8_t *pin*, uint8_t *value* )

#### 5.9.1.2. void changePinPortB ( uint8_t *pin*, uint8_t *value* )

#### 5.9.1.3. void initIO ( void )

#### 5.9.1.4. void IOAccess ( uint8_t *address*, uint8_t * *data*, uint8_t *r_w* )

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

#### 5.9.1.6. void showPorts ( void )

### 5.9.2. Documentación de las variables

#### 5.9.2.1. uint8_t irq[16]

#### 5.9.2.2. port_t PORTA

#### 5.9.2.3. port_t PORTB

## 5.10. Referencia del Archivo io.h

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

### Estructuras de datos

- struct port_t

### 'defines'

- #define XINIT 10
- #define YINIT 5
- #define HIGH 1
- #define LOW 0
- #define Read 1
- #define Write 0
- #define BLUEBLACK 10 /*Text Blue Background Black*/
- #define REDBLACK 20 /*Text Red Background Black*/
- #define WHITEBLACK 30 /*Text White Background White*/

### Funciones

- void IOAccess (uint8_t address, uint8_t *data, uint8_t r_w)
- void changePinPortA (uint8_t pin, uint8_t value)
- void changePinPortB (uint8_t pin, uint8_t value)
- void initIO (void)
- void showPorts (void)
- void showFrame (int x, int y, int w, int h)

### 5.10.1. Documentación de los 'defines'

#### 5.10.1.1. #define BLUEBLACK 10 /*Text Blue Background Black*/

#### 5.10.1.2. #define HIGH 1

#### 5.10.1.3. #define LOW 0

#### 5.10.1.4. #define Read 1

#### 5.10.1.5. #define REDBLACK 20 /∗Text Red Background Black∗/

#### 5.10.1.6. #define WHITEBLACK 30 /∗Text White Background White∗/

#### 5.10.1.7. #define Write 0

#### 5.10.1.8. #define XINIT 10

#### 5.10.1.9. #define YINIT 5

### 5.10.2. Documentación de las funciones

#### 5.10.2.1. void changePinPortA ( uint8_t *pin,* uint8_t *value* )

#### 5.10.2.2. void changePinPortB ( uint8_t *pin,* uint8_t *value* )

#### 5.10.2.3. void initIO ( void )

#### 5.10.2.4. void IOAccess ( uint8_t *address,* uint8_t ∗ *data,* uint8_t *r_w* )

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

#### 5.10.2.6. void showPorts ( void )

## 5.11. Referencia del Archivo main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "visualizacion.h"
#include "instrucciones.h"
#include "decoder.h"
#include "flags.h"
#include "SRAM.h"
#include "io.h"
```

### 'defines'

- #define PC 15

### Funciones

- int main ()

### 5.11.1. Documentación de los 'defines'

#### 5.11.1.1. #define PC 15

### 5.11.2. Documentación de las funciones

#### 5.11.2.1. int main ( )

\ Arreglo registro[] Datos recibidos del microcontrolador. \ Variable i Contador.

## 5.12. Referencia del Archivo ports.c

## 5.13. Referencia del Archivo README.md

## 5.14. Referencia del Archivo SRAM.c

```
#include "SRAM.h"
```

### Funciones

- uint8_t BitCount (uint8_t registers_list[])
- void PUSH (uint8_t *SP, uint32_t *R, uint8_t *RAM, uint8_t registers_list[])
- void POP (uint8_t *SP, uint32_t *R, uint8_t *RAM, uint8_t registers_list[])
- void LDR (uint32_t *R, uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint8_t *RAM)
- void LDRB (uint32_t *R, uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint8_t *RAM)
- void LDRH (uint32_t *R, uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint8_t *RAM)
- void LDRSB (uint32_t *R, uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint8_t *RAM)
- void LDRSH (uint32_t *R, uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint8_t *RAM)
- void STR (uint32_t *R, uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint8_t *RAM)
- void STRB (uint32_t *R, uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint8_t *RAM)
- void STRH (uint32_t *R, uint32_t *Rd, uint32_t Rn, uint32_t Rm, uint8_t *RAM)

### 5.14.1. Documentación de las funciones

#### 5.14.1.1. uint8_t BitCount ( uint8_t *registers_list[]* )

\ brief Funcion que cuenta cuantos bits estan en 1 \ param registers_list Registros a los que se les va a realizar la operacion \ return cont Numero de bits en 1

#### 5.14.1.2. void LDR ( uint32_t * *R*, uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, uint8_t * *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.14.1.3. void LDRB ( uint32_t * *R*, uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, uint8_t * *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.14.1.4. void LDRH ( uint32_t * *R*, uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, uint8_t * *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.14.1.5. void LDRSB ( uint32_t * *R,* uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.14.1.6. void LDRSH ( uint32_t * *R,* uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.14.1.7. void POP ( uint8_t * *SP,* uint32_t * *R,* uint8_t * *RAM,* uint8_t *registers_list[]* )

\ brief Funcion que saca los registros de la SRAM \ param SP Puntero de pila \ param R puntero que almacena los resgitros \ param RAM Memoria \ param registers_list Registros que se van a sacar. \ return void

#### 5.14.1.8. void PUSH ( uint8_t * *SP,* uint32_t * *R,* uint8_t * *RAM,* uint8_t *registers_list[]* )

\ brief Funcion que almacena los registros en la SRAM \ param SP Puntero de pila \ param R puntero con los registros \ param RAM Memoria \ param registers_list Registros que se requieren ingresar. \ return void

#### 5.14.1.9. void STR ( uint32_t * *R,* uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.14.1.10. void STRB ( uint32_t * *R,* uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.14.1.11. void STRH ( uint32_t * *Registro,* uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *RAM* )

\ brief \ param Registro vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

## 5.15. Referencia del Archivo SRAM.h

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

### 'defines'

- #define Mema 64
- #define PC 15

### Funciones

- void PUSH (uint8_t *SP, uint32_t *R, uint8_t *RAM, uint8_t registers_list[])
- uint8_t BitCount (uint8_t registers_list[])

- void POP (uint8_t ∗SP, uint32_t ∗R, uint8_t ∗RAM, uint8_t registers_list[])
- void LDR (uint32_t ∗R, uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint8_t ∗RAM)
- void LDRB (uint32_t ∗R, uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint8_t ∗RAM)
- void LDRH (uint32_t ∗R, uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint8_t ∗RAM)
- void LDRSB (uint32_t ∗R, uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint8_t ∗RAM)
- void LDRSH (uint32_t ∗R, uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint8_t ∗RAM)
- void STR (uint32_t ∗R, uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint8_t ∗RAM)
- void STRB (uint32_t ∗R, uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint8_t ∗RAM)
- void STRH (uint32_t ∗Registro, uint32_t ∗Rd, uint32_t Rn, uint32_t Rm, uint8_t ∗RAM)

### 5.15.1. Documentación de los 'defines'

#### 5.15.1.1. #define Mema 64

#### 5.15.1.2. #define PC 15

### 5.15.2. Documentación de las funciones

#### 5.15.2.1. uint8_t BitCount ( uint8_t *registers_list[]* )

\ brief Funcion que cuenta cuantos bits estan en 1 \ param registers_list Registros a los que se les va a realizar la operacion \ return cont Numero de bits en 1

#### 5.15.2.2. void LDR ( uint32_t ∗ *R,* uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t ∗ *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.15.2.3. void LDRB ( uint32_t ∗ *R,* uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t ∗ *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.15.2.4. void LDRH ( uint32_t ∗ *R,* uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t ∗ *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.15.2.5. void LDRSB ( uint32_t ∗ *R,* uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t ∗ *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.15.2.6. void LDRSH ( uint32_t ∗ *R,* uint32_t ∗ *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t ∗ *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.15.2.7. void POP ( uint8_t ∗ *SP,* uint32_t ∗ *R,* uint8_t ∗ *RAM,* uint8_t *registers_list[]* )

\ brief Funcion que saca los registros de la SRAM \ param SP Puntero de pila \ param R puntero que almacena los resgitros \ param RAM Memoria \ param registers_list Registros que se van a sacar. \ return void

#### 5.15.2.8. void PUSH ( uint8_t * *SP,* uint32_t * *R,* uint8_t * *RAM,* uint8_t *registers_list[ ]* )

\ brief Funcion que almacena los registros en la SRAM \ param SP Puntero de pila \ param R puntero con los registros \ param RAM Memoria \ param registers_list Registros que se requieren ingresar. \ return void

#### 5.15.2.9. void STR ( uint32_t * *R,* uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.15.2.10. void STRB ( uint32_t * *R,* uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *RAM* )

\ brief \ param R vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

#### 5.15.2.11. void STRH ( uint32_t * *Registro,* uint32_t * *Rd,* uint32_t *Rn,* uint32_t *Rm,* uint8_t * *RAM* )

\ brief \ param Registro vector que contiene los registros \ param Rd Registro de almacenamiento \ param Rn Primer operando \ param Rm Segundo operando \ param RAM Memoria RAM \ return void

## 5.16. Referencia del Archivo test.c

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

## 5.17. Referencia del Archivo visualizacion.c

```
#include "visualizacion.h"
```

### 'defines'

- #define PC 15

### Funciones

- void visualizacion_registro (uint32_t *R, uint32_t *B, uint8_t *RAM, instruction_t instruction)

### 5.17.1. Documentación de los 'defines'

#### 5.17.1.1. #define PC 15

### 5.17.2. Documentación de las funciones

#### 5.17.2.1. void visualizacion_registro ( uint32_t * *R,* uint32_t * *B,* uint8_t * *RAM,* instruction_t *instruction* )

\ brief Funcion para visualizar los registros \ param *R Puntero que señala la posicion de los registros almacenados. \ param *Banderas Corresponde a las banderas Cero, Negativo, Acarreo y Sobreflujo. \ return void

## 5.18. Referencia del Archivo visualizacion.h

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <ncurses.h>
#include "flags.h"
#include "decoder.h"
#include "io.h"
```

### Funciones

- void visualizacion_registro (uint32_t ∗R, uint32_t ∗B, uint8_t ∗RAM, instruction_t instruction)

### 5.18.1. Documentación de las funciones

#### 5.18.1.1. void visualizacion_registro ( uint32_t ∗ *R,* uint32_t ∗ *B,* uint8_t ∗ *RAM,* instruction_t *instruction* )

\ brief Funcion para visualizar los registros \ param ∗R Puntero que señala la posicion de los registros almacenados. \ param ∗Banderas Corresponde a las banderas Cero, Negativo, Acarreo y Sobreflujo. \ return void