

ARM_Cortex_M0

Generado por Doxygen 1.8.10

Sábado, 19 de Septiembre de 2015 09:04:02

Índice general

1	Emulador del procesador ARM Cortex -M0	1
2	Índice de estructura de datos	3
2.1	Estructura de datos	3
3	Índice de archivos	5
3.1	Lista de archivos	5
4	Documentación de las estructuras de datos	7
4.1	Referencia de la Estructura flags	7
4.1.1	Descripción detallada	7
4.1.2	Documentación de los campos	7
4.1.2.1	C	7
4.1.2.2	N	7
4.1.2.3	V	7
4.1.2.4	Z	7
4.2	Referencia de la Estructura ins_t	8
4.2.1	Descripción detallada	8
4.2.2	Documentación de los campos	8
4.2.2.1	array	8
4.3	Referencia de la Estructura instruction_t	8
4.3.1	Descripción detallada	8
4.3.2	Documentación de los campos	8
4.3.2.1	mnemonic	8
4.3.2.2	op1_type	9
4.3.2.3	op1_value	9
4.3.2.4	op2_type	9
4.3.2.5	op2_value	9
4.3.2.6	op3_type	9
4.3.2.7	op3_value	9
5	Documentación de archivos	11
5.1	Referencia del Archivo banderas.c	11

5.1.1	Descripción detallada	11
5.1.2	Documentación de las funciones	11
5.1.2.1	flags_aritmetica(uint32_t Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	11
5.1.2.2	flags_global(uint32_t Rd, flags_t *bandera)	11
5.2	Referencia del Archivo banderas.h	12
5.2.1	Descripción detallada	12
5.2.2	Documentación de los 'defines'	12
5.2.2.1	HALF	12
5.2.3	Documentación de las funciones	12
5.2.3.1	flags_aritmetica(uint32_t Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	12
5.2.3.2	flags_global(uint32_t Rd, flags_t *bandera)	13
5.3	Referencia del Archivo conversion.c	13
5.3.1	Descripción detallada	13
5.3.2	Documentación de las funciones	13
5.3.2.1	bin2reg(uint32_t *R, uint32_t *Bit)	13
5.3.2.2	reg2bin(uint32_t R, uint32_t *Bit)	14
5.4	Referencia del Archivo conversion.h	14
5.4.1	Descripción detallada	14
5.4.2	Documentación de las funciones	14
5.4.2.1	bin2reg(uint32_t *R, uint32_t *Bit)	14
5.4.2.2	reg2bin(uint32_t R, uint32_t *Bit)	15
5.5	Referencia del Archivo decoder.c	15
5.5.1	Descripción detallada	15
5.5.2	Documentación de las funciones	15
5.5.2.1	countLines(FILE *fp)	15
5.5.2.2	decodeInstruction(instruction_t instruction, uint32_t *registro, flags_t *bandera) .	15
5.5.2.3	getInstruction(char *instStr)	16
5.5.2.4	readFile(char *filename, ins_t *instructions)	16
5.6	Referencia del Archivo decoder.h	16
5.6.1	Descripción detallada	16
5.6.2	Documentación de las funciones	17
5.6.2.1	countLines(FILE *fp)	17
5.6.2.2	decodeInstruction(instruction_t instruction, uint32_t *registro, flags_t *bandera) .	17
5.6.2.3	getInstruction(char *instStr)	17
5.6.2.4	readFile(char *filename, ins_t *instructions)	17
5.7	Referencia del Archivo instrucciones.c	17
5.7.1	Descripción detallada	18
5.7.2	Documentación de las funciones	18
5.7.2.1	ADCS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	18
5.7.2.2	ADDS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	19

5.7.2.3	ANDS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	19
5.7.2.4	ASRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	19
5.7.2.5	BICS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	19
5.7.2.6	CMN(uint32_t Rn, uint32_t Rm, flags_t *bandera)	20
5.7.2.7	CMP(uint32_t Rn, uint32_t Rm, flags_t *bandera)	20
5.7.2.8	EORS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	20
5.7.2.9	LSLS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	20
5.7.2.10	LSRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	21
5.7.2.11	MOV(uint32_t *Rd, uint32_t Rm)	21
5.7.2.12	MOVS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	21
5.7.2.13	MUL(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	21
5.7.2.14	MVNS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	22
5.7.2.15	NOP()	22
5.7.2.16	ORRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	22
5.7.2.17	REV(uint32_t *Rd, uint32_t Rm)	22
5.7.2.18	REV16(uint32_t *Rd, uint32_t Rm)	23
5.7.2.19	REVSH(uint32_t *Rd, uint32_t Rm)	23
5.7.2.20	RORS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	23
5.7.2.21	RSBS(uint32_t *Rd, uint32_t Rn, flags_t *bandera)	23
5.7.2.22	SBCS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	24
5.7.2.23	SUBS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	24
5.7.2.24	TST(uint32_t Rn, uint32_t Rm, flags_t *bandera)	24
5.8	Referencia del Archivo instrucciones.h	25
5.8.1	Descripción detallada	26
5.8.2	Documentación de los 'typedefs'	26
5.8.2.1	flags_t	26
5.8.3	Documentación de las funciones	26
5.8.3.1	ADCS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	26
5.8.3.2	ADDS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	26
5.8.3.3	ANDS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	27
5.8.3.4	ASRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	27
5.8.3.5	BICS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	27
5.8.3.6	CMN(uint32_t Rn, uint32_t Rm, flags_t *bandera)	27
5.8.3.7	CMP(uint32_t Rn, uint32_t Rm, flags_t *bandera)	28
5.8.3.8	EORS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	28
5.8.3.9	LSLS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	28
5.8.3.10	LSRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	28
5.8.3.11	MOV(uint32_t *Rd, uint32_t Rm)	29
5.8.3.12	MOVS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	29
5.8.3.13	MUL(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	29

5.8.3.14	MVNS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	29
5.8.3.15	NOP()	30
5.8.3.16	ORRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	30
5.8.3.17	REV(uint32_t *Rd, uint32_t Rm)	30
5.8.3.18	REV16(uint32_t *Rd, uint32_t Rm)	30
5.8.3.19	REVSH(uint32_t *Rd, uint32_t Rm)	31
5.8.3.20	RORS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	31
5.8.3.21	RSBS(uint32_t *Rd, uint32_t Rn, flags_t *bandera)	31
5.8.3.22	SBCS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	31
5.8.3.23	SUBS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	32
5.8.3.24	TST(uint32_t Rn, uint32_t Rm, flags_t *bandera)	32
5.9	Referencia del Archivo instrucciones_salto.c	32
5.9.1	Descripción detallada	33
5.9.2	Documentación de las funciones	33
5.9.2.1	B(uint32_t *registro, int salto)	33
5.9.2.2	BAL(uint32_t *registro, int salto, flags_t bandera)	34
5.9.2.3	BCC(uint32_t *registro, int salto, flags_t bandera)	34
5.9.2.4	BCS(uint32_t *registro, int salto, flags_t bandera)	34
5.9.2.5	BEQ(uint32_t *registro, int salto, flags_t bandera)	34
5.9.2.6	BGE(uint32_t *registro, int salto, flags_t bandera)	35
5.9.2.7	BGT(uint32_t *registro, int salto, flags_t bandera)	35
5.9.2.8	BHI(uint32_t *registro, int salto, flags_t bandera)	35
5.9.2.9	BL(uint32_t *registro, int salto)	35
5.9.2.10	BLE(uint32_t *registro, int salto, flags_t bandera)	36
5.9.2.11	BLS(uint32_t *registro, int salto, flags_t bandera)	36
5.9.2.12	BLT(uint32_t *registro, int salto, flags_t bandera)	36
5.9.2.13	BLX(uint32_t *registro, uint32_t R)	36
5.9.2.14	BMI(uint32_t *registro, int salto, flags_t bandera)	37
5.9.2.15	BNE(uint32_t *registro, int salto, flags_t bandera)	37
5.9.2.16	BPL(uint32_t *registro, int salto, flags_t bandera)	37
5.9.2.17	BVC(uint32_t *registro, int salto, flags_t bandera)	37
5.9.2.18	BVS(uint32_t *registro, int salto, flags_t bandera)	38
5.9.2.19	BX(uint32_t *registro, uint32_t R)	38
5.10	Referencia del Archivo instrucciones_salto.h	38
5.10.1	Descripción detallada	39
5.10.2	Documentación de las funciones	39
5.10.2.1	B(uint32_t *registro, int salto)	39
5.10.2.2	BAL(uint32_t *registro, int salto, flags_t bandera)	39
5.10.2.3	BCC(uint32_t *registro, int salto, flags_t bandera)	40
5.10.2.4	BCS(uint32_t *registro, int salto, flags_t bandera)	40

5.10.2.5	BEQ(uint32_t *registro, int salto, flags_t bandera)	40
5.10.2.6	BGE(uint32_t *registro, int salto, flags_t bandera)	40
5.10.2.7	BGT(uint32_t *registro, int salto, flags_t bandera)	41
5.10.2.8	BHI(uint32_t *registro, int salto, flags_t bandera)	41
5.10.2.9	BL(uint32_t *registro, int salto)	41
5.10.2.10	BLE(uint32_t *registro, int salto, flags_t bandera)	41
5.10.2.11	BLS(uint32_t *registro, int salto, flags_t bandera)	42
5.10.2.12	BLT(uint32_t *registro, int salto, flags_t bandera)	42
5.10.2.13	BLX(uint32_t *registro, uint32_t R)	42
5.10.2.14	BMI(uint32_t *registro, int salto, flags_t bandera)	42
5.10.2.15	BNE(uint32_t *registro, int salto, flags_t bandera)	43
5.10.2.16	BPL(uint32_t *registro, int salto, flags_t bandera)	43
5.10.2.17	BVC(uint32_t *registro, int salto, flags_t bandera)	43
5.10.2.18	BVS(uint32_t *registro, int salto, flags_t bandera)	43
5.10.2.19	BX(uint32_t *registro, uint32_t R)	44
5.11	Referencia del Archivo main.c	44
5.11.1	Descripción detallada	44
5.11.2	Documentación de las funciones	44
5.11.2.1	main()	44

Capítulo 1

Emulador del procesador ARM Cortex -M0

Esta es la documentacion para un software que simula el procesador ARM Cortex -M0 el cual es el procesador ARM mas pequeno disponible con un rendimiento de 32 bits. Este software codificado en lenguaje C, con ayuda del compilador codeblocks para su desarrollo y la libreria curses.h para la presentacion de su interfaz se basa en leer las instrucciones del archivo code.txt, que se encuentran en lenguaje de maquina y convertir estas instrucciones en un lenguaje de medio nivel como lo es el lenguaje C. En este se tradujeron 24 instrucciones de lenguaje de maquina a lenguaje C, con las respectivas modificaciones de banderas; tambien se llevo a cabo la traduccion de las funciones de salto.

Para el desarrollo del software se implementaron 15 registros cada uno de 32 bits sin signo, 12 de ellos son utilizados para almacenamiento, uno para el program counter(PC) y otro para link register(LR), ademas se implemento una estructura para el manejo de las banderas de negativo, de cero, de acarreo y bandera de sobreflujo.

Capítulo 2

Índice de estructura de datos

2.1. Estructura de datos

Lista de estructuras con una breve descripción:

flags	Estructura que contiene las banderas	7
ins_t	Estructura que contiene las instrucciones del code.txt	8
instruction_t	Estructura que contiene las instrucciones en segmentos del code.txt	8

Capítulo 3

Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

banderas.c	Documento en donde se especifica cada una de las condiciones para activar o desactivar las banderas	11
banderas.h	Documento en donde estan definidas las funciones que modifican banderas	12
conversion.c	Documento en donde se convierte de decimal a binario y viceversa	13
conversion.h	Documento utilizado prar definir las funciones que convierten de decimal a binario y viceversa	14
decoder.c	Documento en el cual se realiza el proceso de extraer las instrucciones del code.txt, segmen- tarlas y comparar el mnemonic con el nombre de cada instruccion, asi realizar la instruccion deseada	15
decoder.h	Documento en donde esta definida la estructura ins_t y la estructura instruction_t y ademas se definen las instrucciones de manejo del code.txt	16
instrucciones.c	Documento en el cual se realizan las operaciones necesarias para realizar cad instruccion, ademas se definen cuales banderas se modifican con cada instruccion	17
instrucciones.h	Documento en donde esta definida la estructura flags y las funciones	25
instrucciones_salto.c	Documento utilizado para definir PC y RL en cada instruccion a traves de los saltos y ademas utilizar las banderas para identificar si se realiza o no un salto	32
instrucciones_salto.h	Documento en donde estan definidas los tipos de saltos	38
main.c	Documento que utiliza la libreria curses.h para presentar la interfaz, tambien se definen los registros, la estructura bandera y ademas en ella se trabaja con las instrucciones adquiridas del code.txt	44

Capítulo 4

Documentación de las estructuras de datos

4.1. Referencia de la Estructura flags

Estructura que contiene las banderas.

```
#include <instrucciones.h>
```

Campos de datos

- char **N**
- char **Z**
- char **C**
- char **V**

4.1.1. Descripción detallada

Estructura que contiene las banderas.

4.1.2. Documentación de los campos

4.1.2.1. char C

bandera de carry, si hay un carry en una operacion entonces C=1;

4.1.2.2. char N

bandera de un resultado negativo, si este es negativo entonces N=1

4.1.2.3. char V

bandera de sobreflujo, si en una operación los bits mas significativos de los operandos son iguales y el bit mas significativo del resultado es el complemento de los bits de los operandos, se tiene un sobreflujo y V=1

4.1.2.4. char Z

bandera de un resultado igual a 0, si este es cero entonces Z=1;

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

- [instrucciones.h](#)

4.2. Referencia de la Estructura `ins_t`

Estructura que contiene las instrucciones del `code.txt`.

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

Campos de datos

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

4.2.1. Descripción detallada

Estructura que contiene las instrucciones del `code.txt`.

4.2.2. Documentación de los campos

4.2.2.1. `char** array`

Arreglo que utiliza memoria dinamica para obtener las lineas de codigo del `code.txt`

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

- [decoder.h](#)

4.3. Referencia de la Estructura `instruction_t`

Estructura que contiene las instrucciones en segmentos del `code.txt`.

```
#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](#)

4.3.1. Descripción detallada

Estructura que contiene las instrucciones en segmentos del `code.txt`.

4.3.2. Documentación de los campos

4.3.2.1. `char mnemonic[10]`

Contiene el nombre de la instruccion

4.3.2.2. char op1_type

Contiene el tipo del primer parametro

4.3.2.3. uint32_t op1_value

Contiene el numero del registro a operar del primer parametro

4.3.2.4. char op2_type

Contiene el tipo del segundo parametro

4.3.2.5. uint32_t op2_value

Contiene el numero del registro a operar del segundo parametro o un numero

4.3.2.6. char op3_type

Contiene el tipo del tercer parametro

4.3.2.7. uint32_t op3_value

Contiene el numero del registro a operar del tercer parametro o un numero

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

- [decoder.h](#)

Capítulo 5

Documentación de archivos

5.1. Referencia del Archivo banderas.c

Documento en donde se especifica cada una de las condiciones para activar o desactivar las banderas.

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

Funciones

- void [flags_aritmetica](#) (uint32_t Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
Funcion que modifica banderas para funciones aritmeticas como la suma, resta.
- void [flags_global](#) (uint32_t Rd, [flags_t](#) *bandera)
Funcion que modifica bandera de negativo y bandera de cero.

5.1.1. Descripción detallada

Documento en donde se especifica cada una de las condiciones para activar o desactivar las banderas.

5.1.2. Documentación de las funciones

5.1.2.1. void [flags_aritmetica](#) (uint32_t *Rd*, uint32_t *Rn*, uint32_t *Rm*, [flags_t](#) * *bandera*)

Funcion que modifica banderas para funciones aritmeticas como la suma, resta.

Parámetros

<i>Rd</i>	registro donde se guardo el resultado
<i>Rn</i>	registro que se opera con Rm
<i>Rm</i>	registro o numero que se opera con Rn
<i>bandera</i>	puntero de la estructura flags_t bandera

Devuelve

La Funcion no tiene retorno

5.1.2.2. void [flags_global](#) (uint32_t *Rd*, [flags_t](#) * *bandera*)

Funcion que modifica bandera de negativo y bandera de cero.

Parámetros

<i>Rd</i>	registro donde se guardo el resultado
<i>bandera</i>	puntero de la estructura flags_t bandera

Devuelve

La Funcion no tiene retorno

5.2. Referencia del Archivo banderas.h

Documento en donde estan definidas las funciones que modifican banderas.

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

'defines'

- #define HALF 2147483648UL
numero igual a 2^{31} , utilizado en [banderas.c](#)

Funciones

- void flags_aritmetica (uint32_t Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)
Funcion que modifica banderas para funciones aritmeticas como la suma, resta.
- void flags_global (uint32_t Rd, flags_t *bandera)
Funcion que modifica bandera de negativo y bandera de cero.

5.2.1. Descripción detallada

Documento en donde estan definidas las funciones que modifican banderas.

5.2.2. Documentación de los 'defines'

5.2.2.1. #define HALF 2147483648UL

numero igual a 2^{31} , utilizado en [banderas.c](#)

5.2.3. Documentación de las funciones

5.2.3.1. void flags_aritmetica (uint32_t Rd, uint32_t Rn, uint32_t Rm, flags_t * bandera)

Funcion que modifica banderas para funciones aritmeticas como la suma, resta.

Parámetros

<i>Rd</i>	registro donde se guardo el resultado
<i>Rn</i>	registro que se opera con Rm

<i>Rm</i>	registro o numero que se opera con Rn
<i>bandera</i>	puntero de la estructura flags_t bandera

Devuelve

La Funcion no tiene retorno

5.2.3.2. void flags_global (uint32_t Rd, flags_t * bandera)

Funcion que modifica bandera de negativo y bandera de cero.

Parámetros

<i>Rd</i>	registro donde se guardo el resultado
<i>bandera</i>	puntero de la estructutura flags_t bandera

Devuelve

La Funcion no tiene retorno

5.3. Referencia del Archivo conversion.c

Documento en donde se convierte de decimal a binario y viceversa.

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

Funciones

- void [reg2bin](#) (uint32_t R, uint32_t *Bit)

La funcion cumple el deber de convertir un decimal a binario, en donde cada elemento del arreglo bit corresponde a un bit de R.

- void [bin2reg](#) (uint32_t *R, uint32_t *Bit)

La funcion cumple el deber de convertir un binario a decimal, en donde cada elemento del arreglo bit corresponde a un bit de R.

5.3.1. Descripción detallada

Documento en donde se convierte de decimal a binario y viceversa.

5.3.2. Documentación de las funciones

5.3.2.1. void bin2reg (uint32_t * R, uint32_t * Bit)

La funcion cumple el deber de convertir un binario a decimal, en donde cada elemento del arreglo bit corresponde a un bit de R.

Parámetros

<i>R</i>	puntero al registro a guardar la conversion
<i>Bit</i>	puntero a un arreglo que posee 32 elementos

Devuelve

La funcion no retorna nada

5.3.2.2. void reg2bin (uint32_t R, uint32_t * Bit)

La funcion cumple el deber de convertir un decimal a binario, en donde cada elemento del arreglo bit corresponde a un bit de R.

Parámetros

<i>R</i>	Registro a convertir
<i>Bit</i>	puntero a un arreglo que posee 32 elementos

Devuelve

La funcion no retorna nada

5.4. Referencia del Archivo conversion.h

Documento utilizado prar definir las funciones que convierten de decimal a binario y viceversa.

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

Funciones

- void [reg2bin](#) (uint32_t R, uint32_t *Bit)

La funcion cumple el deber de convertir un decimal a binario, en donde cada elemento del arreglo bit corresponde a un bit de R.

- void [bin2reg](#) (uint32_t *R, uint32_t *Bit)

La funcion cumple el deber de convertir un binario a decimal, en donde cada elemento del arreglo bit corresponde a un bit de R.

5.4.1. Descripción detallada

Documento utilizado prar definir las funciones que convierten de decimal a binario y viceversa.

5.4.2. Documentación de las funciones**5.4.2.1. void bin2reg (uint32_t * R, uint32_t * Bit)**

La funcion cumple el deber de convertir un binario a decimal, en donde cada elemento del arreglo bit corresponde a un bit de R.

Parámetros

<i>R</i>	puntero al registro a guardar la conversion
<i>Bit</i>	puntero a un arreglo que posee 32 elementos

Devuelve

La funcion no retorna nada

5.4.2.2. void reg2bin (uint32_t R, uint32_t * Bit)

La funcion cumple el deber de convertir un decimal a binario, en donde cada elemento del arreglo bit corresponde a un bit de R.

Parámetros

<i>R</i>	Registro a convertir
<i>Bit</i>	puntero a un arreglo que posee 32 elementos

Devuelve

La funcion no retorna nada

5.5. Referencia del Archivo decoder.c

Documento en el cual se realiza el proceso de extraer las instrucciones del code.txt, segmentarlas y comparar el mnemonic con el nombre de cada instruccion, asi realizar la instruccion deseada.

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

Funciones

- void `decodeInstruction` (`instruction_t` instruction, `uint32_t` *registro, `flags_t` *bandera)
Decodifica la instrucción y la ejecuta.
- `instruction_t` `getInstruction` (char *instStr)
Obtiene la instrucción separada por partes.
- int `readFile` (char *filename, `ins_t` *instructions)
- int `countLines` (FILE *fp)

5.5.1. Descripción detallada

Documento en el cual se realiza el proceso de extraer las instrucciones del code.txt, segmentarlas y comparar el mnemonic con el nombre de cada instruccion, asi realizar la instruccion deseada.

5.5.2. Documentación de las funciones**5.5.2.1. int countLines (FILE * fp)****5.5.2.2. void decodeInstruction (instruction_t instruction, uint32_t * registro, flags_t * bandera)**

Decodifica la instrucción y la ejecuta.

Parámetros

<i>instruction</i>	instrucción a decodificar y ejecutar.
--------------------	---------------------------------------

5.5.2.3. `instruction_t getInstruction (char * instStr)`

Obtiene la instrucción separada por partes.

Parámetros

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

Devuelve

`instruction_t` la instrucción separada por partes.

5.5.2.4. `int readFile (char * filename, ins_t * instructions)`5.6. Referencia del Archivo `decoder.h`

Documento en donde esta definida la estructura `ins_t` y la estructura `instruction_t` y ademas se definen las instrucciones de manejo del `code.txt`.

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

Estructuras de datos

- struct `ins_t`
Estructura que contiene las instrucciones del code.txt.
- struct `instruction_t`
Estructura que contiene las instrucciones en segmentos del code.txt.

Funciones

- void `decodeInstruction (instruction_t instruction, uint32_t *registro, flags_t *bandera)`
Decodifica la instrucción y la ejecuta.
- `instruction_t getInstruction (char *instStr)`
Obtiene la instrucción separada por partes.
- int `readFile (char *filename, ins_t *instructions)`
- int `countLines (FILE *fp)`

5.6.1. Descripción detallada

Documento en donde esta definida la estructura `ins_t` y la estructura `instruction_t` y ademas se definen las instrucciones de manejo del `code.txt`.

5.6.2. Documentación de las funciones

5.6.2.1. `int countLines (FILE * fp)`

5.6.2.2. `void decodeInstruction (instruction_t instruction, uint32_t * registro, flags_t * bandera)`

Decodifica la instrucción y la ejecuta.

Parámetros

<i>instruction</i>	instrucción a decodificar y ejecutar.
--------------------	---------------------------------------

5.6.2.3. `instruction_t getInstruction (char * instStr)`

Obtiene la instrucción separada por partes.

Parámetros

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

Devuelve

`instruction_t` la instrucción separada por partes.

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

5.7. Referencia del Archivo instrucciones.c

Documento en el cual se realizan las operaciones necesarias para realizar cada instrucción, además se definen cuales banderas se modifican con cada instrucción.

```
#include <stdint.h>
#include "instrucciones.h"
#include "banderas.h"
#include "conversion.h"
```

Funciones

- void `ADCS` (uint32_t *Rd, uint32_t Rn, uint32_t Rm, `flags_t` *bandera)
*funcion que realiza la suma entre Rn y Rm y lo guarda en *Rd*
- void `ADDS` (uint32_t *Rd, uint32_t Rn, uint32_t Rm, `flags_t` *bandera)
*funcion que realiza la suma entre Rn y Rm y lo guarda en *Rd*
- void `ANDS` (uint32_t *Rd, uint32_t Rn, uint32_t Rm, `flags_t` *bandera)
*funcion que realiza operacion AND entre *Rd y Rm*
- void `ASRS` (uint32_t *Rd, uint32_t Rn, uint32_t Rm, `flags_t` *bandera)
funcion que realiza desplazamiento aritmetico del registro Rd, Rm veces
- void `BICS` (uint32_t *Rd, uint32_t Rn, uint32_t Rm, `flags_t` *bandera)
funcion que realiza la operacion AND entre Rd y el complemento de un numero o un registro Rm
- void `CMN` (uint32_t Rn, uint32_t Rm, `flags_t` *bandera)
funcion que realiza la suma entre Rn y Rm, pero no modifica Rn
- void `CMP` (uint32_t Rn, uint32_t Rm, `flags_t` *bandera)
funcion que realiza la resta entre Rn y Rm, pero no modifica Rn
- void `EORS` (uint32_t *Rd, uint32_t Rn, uint32_t Rm, `flags_t` *bandera)

- funcion que realiza operación XOR entre *Rd y Rm*
- void **LSLS** (uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)
funcion que realiza desplazamiento lógico del registro Rd a la izquierda, Rm veces
- void **LSRS** (uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)
funcion que realiza desplazamiento lógico del registro Rd a la derecha, Rm veces
- void **MOV** (uint32_t *Rd, uint32_t Rm)
*funcion que realiza una copia de Rm en *Rd*
- void **MOVS** (uint32_t *Rd, uint32_t Rm, flags_t *bandera)
*funcion que realiza una copia de Rm en *Rd*
- void **MUL** (uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)
*funcion que realiza la multiplicacion entre Rn y Rm y lo guarda en *Rd*
- void **MVNS** (uint32_t *Rd, uint32_t Rm, flags_t *bandera)
funcion que guarda el complemento de un numero o registro Rm y lo guarda en Rd
- void **NOP** ()
funcion que no realiza nada en un ciclo de reloj
- void **ORRS** (uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)
*funcion que realiza operación OR entre *Rd y Rm*
- void **REV** (uint32_t *Rd, uint32_t Rm)
funcion que cambia el orden de los bytes
- void **REV16** (uint32_t *Rd, uint32_t Rm)
funcion que cambia el orden de los Bytes en cada halfword de 16 bits
- void **REVSH** (uint32_t *Rd, uint32_t Rm)
funcion que realiza extension de signo y cambia el orden de los Bytes del halfword bajo
- void **RORS** (uint32_t *Rd, uint32_t Rm, flags_t *bandera)
funcion que realiza rotacion a la derecha del registro Rd, Rm veces
- void **RSBS** (uint32_t *Rd, uint32_t Rn, flags_t *bandera)
funcion que obtiene el complemento a dos de un numero o registro Rn y lo guarda en Rd
- void **SBCS** (uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)
*funcion que realiza la resta entre Rn y Rm y lo guarda en *Rd*
- void **SUBS** (uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)
*funcion que realiza la resta entre Rn y Rm y lo guarda en *Rd*
- void **TST** (uint32_t Rn, uint32_t Rm, flags_t *bandera)
funcion que realiza operacion AND bit a bit entre Rn y Rm pero no modifica Rn

5.7.1. Descripción detallada

Documento en el cual se realizan las operaciones necesarias para realizar cada instrucción, además se definen cuales banderas se modifican con cada instrucción.

5.7.2. Documentación de las funciones

5.7.2.1. void ADCS (uint32_t * Rd, uint32_t Rn, uint32_t Rm, flags_t * bandera)

funcion que realiza la suma entre Rn y Rm y lo guarda en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
-----------	-------------------------

<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.2. void ADDS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la suma entre *Rn* y *Rm* y lo guarda en **Rd*

Parámetros

<i>Rd</i>	puntero del registro <i>Rd</i>
<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.3. void ANDS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza operacion AND entre **Rd* y *Rm*

Parámetros

<i>Rd</i>	puntero del registro <i>Rd</i>
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.4. void ASRS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza desplazamiento aritmetico del registro *Rd*, *Rm* veces

Parámetros

<i>Rd</i>	puntero del registro <i>Rd</i>
<i>Rm</i>	numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.5. void BICS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la operacion AND entre *Rd* y el complemento de un numero o un registro *Rm*

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro n o un numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.6. void CMN (uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la suma entre Rn y Rm, pero no modifica Rn

Parámetros

<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.7. void CMP (uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la resta entre Rn y Rm, pero no modifica Rn

Parámetros

<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.8. void EORS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza operación XOR entre *Rd y Rm

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.9. void LSLS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza desplazamiento lógico del registro Rd a la izquierda, Rm veces

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.10. void LSRS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza desplazamiento lógico del registro Rd a la derecha, Rm veces

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.11. void MOV (uint32_t * *Rd*, uint32_t *Rm*)

funcion que realiza una copia de Rm en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o numero

Devuelve

La funcion no tiene retorno

5.7.2.12. void MOVS (uint32_t * *Rd*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza una copia de Rm en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.13. void MUL (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la multiplicacion entre Rn y Rm y lo guarda en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.14. void MVNS (uint32_t * *Rd*, uint32_t *Rm*, flags_t * *bandera*)

funcion que guarda el complemento de un numero o registro Rm y lo guarda en Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro n o un numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.15. void NOP ()

funcion que no realiza nada en un ciclo de reloj

Devuelve

La funcion no tiene retorno

5.7.2.16. void ORRS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza operación OR entre *Rd y Rm

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

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

funcion que cambia el orden de los bytes

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o un numero

Devuelve

La funcion no tiene retorno

5.7.2.18. void REV16 (uint32_t * *Rd*, uint32_t *Rm*)

funcion que cambia el orden de los Bytes en cada halfword de 16 bits

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o un numero

Devuelve

La funcion no tiene retorno

5.7.2.19. void REVSH (uint32_t * *Rd*, uint32_t *Rm*)

funcion que realiza extension de signo y cambia el orden de los Bytes del halfword bajo

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o un numero

Devuelve

La funcion no tiene retorno

5.7.2.20. void RORS (uint32_t * *Rd*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza rotacion a la derecha del registro Rd, Rm veces

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.21. void RSBS (uint32_t * *Rd*, uint32_t *Rn*, flags_t * *bandera*)

funcion que obtiene el complemento a dos de un numero o registro Rn y lo guarda en Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rn</i>	registro n o un numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.22. void SBCS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la resta entre Rn y Rm y lo guarda en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.23. void SUBS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la resta entre Rn y Rm y lo guarda en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.7.2.24. void TST (uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza operacion AND bit a bit entre Rn y Rm pero no modifica Rn

Parámetros

<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8. Referencia del Archivo instrucciones.h

Documento en donde esta definida la estructura flags y las funciones.

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

Estructuras de datos

- struct [flags](#)

Estructura que contiene las banderas.

'typedefs'

- typedef struct [flags](#) [flags_t](#)

Funciones

- void [ADCS](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
*funcion que realiza la suma entre Rn y Rm y lo guarda en *Rd*
- void [ADDS](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
*funcion que realiza la suma entre Rn y Rm y lo guarda en *Rd*
- void [ANDS](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
*funcion que realiza operacion AND entre *Rd y Rm*
- void [ASRS](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
funcion que realiza desplazamiento aritmetico del registro Rd, Rm veces
- void [BICS](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
funcion que realiza la operacion AND entre Rd y el complemento de un numero o un registro Rm
- void [CMN](#) (uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
funcion que realiza la suma entre Rn y Rm, pero no modifica Rn
- void [CMP](#) (uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
funcion que realiza la resta entre Rn y Rm, pero no modifica Rn
- void [EORS](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
*funcion que realiza operación XOR entre *Rd y Rm*
- void [LSLS](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
funcion que realiza desplazamiento lógico del registro Rd a la izquierda, Rm veces
- void [LSRS](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
funcion que realiza desplazamiento lógico del registro Rd a la derecha, Rm veces
- void [MOV](#) (uint32_t *Rd, uint32_t Rm)
*funcion que realiza una copia de Rm en *Rd*
- void [MOVS](#) (uint32_t *Rd, uint32_t Rm, [flags_t](#) *bandera)
*funcion que realiza una copia de Rm en *Rd*
- void [MUL](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
*funcion que realiza la multiplicacion entre Rn y Rm y lo guarda en *Rd*
- void [MVNS](#) (uint32_t *Rd, uint32_t Rm, [flags_t](#) *bandera)
funcion que guarda el complemento de un numero o registro Rm y lo guarda en Rd
- void [NOP](#) ()
funcion que no realiza nada en un ciclo de reloj
- void [ORRS](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm, [flags_t](#) *bandera)
*funcion que realiza operación OR entre *Rd y Rm*
- void [REV](#) (uint32_t *Rd, uint32_t Rm)

- funcion que cambia el orden de los bytes*
- void **REV16** (uint32_t *Rd, uint32_t Rm)
- funcion que cambia el orden de los Bytes en cada halfword de 16 bits*
- void **REVSH** (uint32_t *Rd, uint32_t Rm)
- funcion que realiza extension de signo y cambia el orden de los Bytes del halfword bajo*
- void **RORS** (uint32_t *Rd, uint32_t Rm, flags_t *bandera)
- funcion que realiza rotacion a la derecha del registro Rd, Rm veces*
- void **RSBS** (uint32_t *Rd, uint32_t Rn, flags_t *bandera)
- funcion que obtiene el complemento a dos de un numero o registro Rn y lo guarda en Rd*
- void **SBCS** (uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)
- funcion que realiza la resta entre Rn y Rm y lo guarda en *Rd*
- void **SUBS** (uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)
- funcion que realiza la resta entre Rn y Rm y lo guarda en *Rd*
- void **TST** (uint32_t Rn, uint32_t Rm, flags_t *bandera)
- funcion que realiza operacion AND bit a bit entre Rn y Rm pero no modifica Rn*

5.8.1. Descripción detallada

Documento en donde esta definida la estructura flags y las funciones.

5.8.2. Documentación de los 'typedefs'

5.8.2.1. typedef struct flags flags_t

5.8.3. Documentación de las funciones

5.8.3.1. void ADCS (uint32_t * Rd, uint32_t Rn, uint32_t Rm, flags_t * bandera)

funcion que realiza la suma entre Rn y Rm y lo guarda en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.2. void ADDS (uint32_t * Rd, uint32_t Rn, uint32_t Rm, flags_t * bandera)

funcion que realiza la suma entre Rn y Rm y lo guarda en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rn</i>	registro n

<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.3. void ANDS (uint32_t * Rd, uint32_t Rn, uint32_t Rm, flags_t * bandera)

funcion que realiza operacion AND entre *Rd y Rm

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.4. void ASRS (uint32_t * Rd, uint32_t Rn, uint32_t Rm, flags_t * bandera)

funcion que realiza desplazamiento aritmetico del registro Rd, Rm veces

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.5. void BICS (uint32_t * Rd, uint32_t Rn, uint32_t Rm, flags_t * bandera)

funcion que realiza la operacion AND entre Rd y el complemento de un numero o un registro Rm

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro n o un numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.6. void CMN (uint32_t Rn, uint32_t Rm, flags_t * bandera)

funcion que realiza la suma entre Rn y Rm, pero no modifica Rn

Parámetros

<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.7. void CMP (uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la resta entre *Rn* y *Rm*, pero no modifica *Rn*

Parámetros

<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.8. void EORS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza operación XOR entre **Rd* y *Rm*

Parámetros

<i>Rd</i>	puntero del registro <i>Rd</i>
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.9. void LSLS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza desplazamiento lógico del registro *Rd* a la izquierda, *Rm* veces

Parámetros

<i>Rd</i>	puntero del registro <i>Rd</i>
<i>Rm</i>	numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.10. void LSRS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza desplazamiento lógico del registro *Rd* a la derecha, *Rm* veces

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.11. void MOV (uint32_t * *Rd*, uint32_t *Rm*)

funcion que realiza una copia de Rm en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o numero

Devuelve

La funcion no tiene retorno

5.8.3.12. void MOVS (uint32_t * *Rd*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza una copia de Rm en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.13. void MUL (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la multiplicacion entre Rn y Rm y lo guarda en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.14. void MVNS (uint32_t * *Rd*, uint32_t *Rm*, flags_t * *bandera*)

funcion que guarda el complemento de un numero o registro Rm y lo guarda en Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro n o un numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.15. void NOP ()

funcion que no realiza nada en un ciclo de reloj

Devuelve

La funcion no tiene retorno

5.8.3.16. void ORRS (uint32_t * Rd, uint32_t Rn, uint32_t Rm, flags_t * bandera)

funcion que realiza operación OR entre *Rd y Rm

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

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

funcion que cambia el orden de los bytes

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o un numero

Devuelve

La funcion no tiene retorno

5.8.3.18. void REV16 (uint32_t * Rd, uint32_t Rm)

funcion que cambia el orden de los Bytes en cada halfword de 16 bits

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o un numero

Devuelve

La funcion no tiene retorno

5.8.3.19. void REVSH (uint32_t * *Rd*, uint32_t *Rm*)

funcion que realiza extension de signo y cambia el orden de los Bytes del halfword bajo

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	registro m o un numero

Devuelve

La funcion no tiene retorno

5.8.3.20. void RORS (uint32_t * *Rd*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza rotacion a la derecha del registro Rd, Rm veces

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rm</i>	numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.21. void RSBS (uint32_t * *Rd*, uint32_t *Rn*, flags_t * *bandera*)

funcion que obtiene el complemento a dos de un numero o registro Rn y lo guarda en Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rn</i>	registro n o un numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.22. void SBCS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la resta entre Rn y Rm y lo guarda en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.23. void SUBS (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza la resta entre Rn y Rm y lo guarda en *Rd

Parámetros

<i>Rd</i>	puntero del registro Rd
<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.8.3.24. void TST (uint32_t *Rn*, uint32_t *Rm*, flags_t * *bandera*)

funcion que realiza operacion AND bit a bit entre Rn y Rm pero no modifica Rn

Parámetros

<i>Rn</i>	registro n
<i>Rm</i>	registro m o numero
<i>bandera</i>	puntero a la estructura bandera de entrada

Devuelve

La funcion no tiene retorno

5.9. Referencia del Archivo instrucciones_salto.c

Documento utilizado para definir PC y RL en cada instruccion a traves de los saltos y ademas utilizar las banderas para identificar si se realiza o no un salto.

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

Funciones

- void **B** (uint32_t *registro, int salto)
Funcion que realiza el salto en el emulador.
- void **BL** (uint32_t *registro, int salto)
Funcion que guarda en LR el valor de PC+2 y luego realiza el salto en el emulador.

- void **BLX** (uint32_t *registro, uint32_t R)
Funcion que guarda en LR el valor de PC+2 y luego realiza el salto en el emulador.
- void **BX** (uint32_t *registro, uint32_t R)
Funcion que conduce a cierta posicion en el emulador.
- void **BEQ** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es 0.
- void **BNE** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior no es 0.
- void **BCS** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es mayor o igual (sin signo)
- void **BCC** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es menor (sin signo)
- void **BMI** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es negativo.
- void **BPL** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es positivo.
- void **BVS** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior hubo sobreflujo.
- void **BVC** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior no hubo sobreflujo.
- void **BHI** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es mayor (sin signo)
- void **BLS** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es menor o igual (sin signo)
- void **BGE** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es mayor o igual (signo)
- void **BLT** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es menor (signo)
- void **BGT** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es mayor (signo)
- void **BLE** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza el salto si el resultado anterior es menor o igual (signo)
- void **BAL** (uint32_t *registro, int salto, flags_t bandera)
Funcion que realiza siempre el salto.

5.9.1. Descripción detallada

Documento utilizado para definir PC y RL en cada instruccion a traves de los saltos y ademas utilizar las banderas para identificar si se realiza o no un salto.

5.9.2. Documentación de las funciones

5.9.2.1. void B (uint32_t * registro, int salto)

Funcion que realiza el salto en el emulador.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar

Devuelve

La funcion no tiene retorno

5.9.2.2. void BAL (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza siempre el salto.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.3. void BCC (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es menor (sin signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.4. void BCS (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es mayor o igual (sin signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.5. void BEQ (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es 0.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.6. void BGE (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es mayor o igual (signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.7. void BGT (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es mayor (signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.8. void BHI (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es mayor (sin signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.9. void BL (uint32_t * *registro*, int *salto*)

Funcion que guarda en LR el valor de PC+2 y luego realiza el salto en el emulador.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC y realiza el salto
<i>salto</i>	cantidad de líneas a saltar

Devuelve

La funcion no tiene retorno

5.9.2.10. void BLE (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es menor o igual (signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.11. void BLS (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es menor o igual (sin signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.12. void BLT (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es menor (signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.13. void BLX (uint32_t * *registro*, uint32_t *R*)

Funcion que guarda en LR el valor de PC+2 y luego realiza el salto en el emulador.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>R</i>	registro que contiene las lineas a saltar

Devuelve

La funcion no tiene retorno

5.9.2.14. void BMI (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es negativo.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de lineas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.15. void BNE (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior no es 0.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de lineas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.16. void BPL (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es positivo.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de lineas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.17. void BVC (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior no hubo sobreflujo.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.18. void BVS (uint32_t * registro, int salto, flags_t bandera)

Funcion que realiza el salto si el resultado anterior hubo sobreflujo.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.9.2.19. void BX (uint32_t * registro, uint32_t R)

Funcion que conduce a cierta posicion en el emulador.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>R</i>	registro que contiene la posicion a saltar

Devuelve

La funcion no tiene retorno

5.10. Referencia del Archivo instrucciones_salto.h

Documento en donde estan definidas los tipos de saltos.

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

Funciones

- void **B** (uint32_t *registro, int salto)
Funcion que realiza el salto en el emulador.
- void **BL** (uint32_t *registro, int salto)
Funcion que guarda en LR el valor de PC+2 y luego realiza el salto en el emulador.
- void **BLX** (uint32_t *registro, uint32_t R)
Funcion que que guarda en LR el valor de PC+2 y luego realiza el salto en el emulador.
- void **BX** (uint32_t *registro, uint32_t R)

- Funcion que conduce a cierta posicion en el emulador.*
- void **BEQ** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es 0.*
- void **BNE** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior no es 0.*
- void **BCS** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es mayor o igual (sin signo)*
- void **BCC** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es menor (sin signo)*
- void **BMI** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es negativo.*
- void **BPL** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es positivo.*
- void **BVS** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior hubo sobreflujo.*
- void **BVC** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior no hubo sobreflujo.*
- void **BHI** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es mayor (sin signo)*
- void **BLS** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es menor o igual (sin signo)*
- void **BGE** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es mayor o igual (signo)*
- void **BLT** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es menor (signo)*
- void **BGT** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es mayor (signo)*
- void **BLE** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza el salto si el resultado anterior es menor o igual (signo)*
- void **BAL** (uint32_t *registro, int salto, **flags_t** bandera)
- Funcion que realiza siempre el salto.*

5.10.1. Descripción detallada

Documento en donde estan definidas los tipos de saltos.

5.10.2. Documentación de las funciones

5.10.2.1. void B (uint32_t * registro, int salto)

Funcion que realiza el salto en el emulador.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de lineas a saltar

Devuelve

La funcion no tiene retorno

5.10.2.2. void BAL (uint32_t * registro, int salto, flags_t bandera)

Funcion que realiza siempre el salto.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.3. void BCC (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es menor (sin signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.4. void BCS (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es mayor o igual (sin signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.5. void BEQ (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es 0.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.6. void BGE (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es mayor o igual (signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.7. void BGT (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es mayor (signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.8. void BHI (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es mayor (sin signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.9. void BL (uint32_t * *registro*, int *salto*)

Funcion que guarda en LR el valor de PC+2 y luego realiza el salto en el emulador.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC y realiza el salto
<i>salto</i>	cantidad de líneas a saltar

Devuelve

La funcion no tiene retorno

5.10.2.10. void BLE (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es menor o igual (signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.11. void BLS (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es menor o igual (sin signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.12. void BLT (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es menor (signo)

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.13. void BLX (uint32_t * *registro*, uint32_t *R*)

Funcion que guarda en LR el valor de PC+2 y luego realiza el salto en el emulador.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>R</i>	registro que contiene las lineas a saltar

Devuelve

La funcion no tiene retorno

5.10.2.14. void BMI (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es negativo.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.15. void BNE (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior no es 0.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.16. void BPL (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior es positivo.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.17. void BVC (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior no hubo sobreflujo.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.18. void BVS (uint32_t * *registro*, int *salto*, flags_t *bandera*)

Funcion que realiza el salto si el resultado anterior hubo sobreflujo.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>salto</i>	cantidad de líneas a saltar
<i>bandera</i>	estructura en donde estan las banderas

Devuelve

La funcion no tiene retorno

5.10.2.19. void BX (uint32_t * *registro*, uint32_t *R*)

Funcion que conduce a cierta posicion en el emulador.

Parámetros

<i>registro</i>	puntero al arreglo que contiene PC
<i>R</i>	registro que contiene la posicion a saltar

Devuelve

La funcion no tiene retorno

5.11. Referencia del Archivo main.c

Documento que utiliza la libreria curses.h para presentar la interfaz, tambien se definen los registros, la estructura bandera y ademas en ella se trabaja con las instrucciones adquiridas del code.txt.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include "decoder.h"
#include "banderas.h"
#include "instrucciones.h"
#include "instrucciones_saltos.h"
#include "curses.h"
```

Funciones

- int [main](#) ()

5.11.1. Descripción detallada

Documento que utiliza la libreria curses.h para presentar la interfaz, tambien se definen los registros, la estructura bandera y ademas en ella se trabaja con las instrucciones adquiridas del code.txt.

5.11.2. Documentación de las funciones**5.11.2.1. int main ()**