

ARM_Cortex_M0

Generado por Doxygen 1.8.10

Lunes, 19 de Octubre de 2015 22:08:23

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
4.3.2.8	registers_list	9
4.4	Referencia de la Estructura port_t	9

4.4.1	Descripción detallada	9
4.4.2	Documentación de los campos	10
4.4.2.1	DDR	10
4.4.2.2	Interrupts	10
4.4.2.3	PIN	10
4.4.2.4	Pins	10
4.4.2.5	PORT	10
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	16
5.5.2.1	countLines(FILE *fp)	16
5.5.2.2	decodeInstruction(instruction_t instruction, uint32_t *registro, flags_t *bandera, uint8_t *SRAM, uint16_t *codificacion, char **Flash)	16
5.5.2.3	getInstruction(char *instStr)	16
5.5.2.4	readFile(char *filename, ins_t *instructions)	16
5.5.3	Documentación de las variables	16

5.5.3.1	irq	16
5.6	Referencia del Archivo decoder.h	16
5.6.1	Descripción detallada	17
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, uint8_t *RAM, uint16_t *codificacion, char **Flash)	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	ADD(uint32_t *Rd, uint32_t Rn, uint32_t Rm)	19
5.7.2.3	ADDS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	19
5.7.2.4	ANDS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	19
5.7.2.5	ASRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	19
5.7.2.6	BICS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	20
5.7.2.7	CMN(uint32_t Rn, uint32_t Rm, flags_t *bandera)	20
5.7.2.8	CMP(uint32_t Rn, uint32_t Rm, flags_t *bandera)	20
5.7.2.9	EORS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	20
5.7.2.10	LSLS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	21
5.7.2.11	LSRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	21
5.7.2.12	MOV(uint32_t *Rd, uint32_t Rm)	21
5.7.2.13	MOVS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	21
5.7.2.14	MULS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	22
5.7.2.15	MVNS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	22
5.7.2.16	NOP()	22
5.7.2.17	ORRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	22
5.7.2.18	REV(uint32_t *Rd, uint32_t Rm)	23
5.7.2.19	REV16(uint32_t *Rd, uint32_t Rm)	23
5.7.2.20	REVSH(uint32_t *Rd, uint32_t Rm)	23
5.7.2.21	RORS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	23
5.7.2.22	RSBS(uint32_t *Rd, uint32_t Rn, flags_t *bandera)	24
5.7.2.23	SBCS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	24
5.7.2.24	SUB(uint32_t *Rd, uint32_t Rn, uint32_t Rm)	24
5.7.2.25	SUBS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	24
5.7.2.26	TST(uint32_t Rn, uint32_t Rm, flags_t *bandera)	25
5.8	Referencia del Archivo instrucciones.h	25
5.8.1	Descripción detallada	26

5.8.2	Documentación de los 'typedefs'	27
5.8.2.1	flags_t	27
5.8.3	Documentación de las funciones	27
5.8.3.1	ADCS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	27
5.8.3.2	ADD(uint32_t *Rd, uint32_t Rn, uint32_t Rm)	28
5.8.3.3	ADDS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	28
5.8.3.4	ANDS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	28
5.8.3.5	ASRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	28
5.8.3.6	BICS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	29
5.8.3.7	CMN(uint32_t Rn, uint32_t Rm, flags_t *bandera)	29
5.8.3.8	CMP(uint32_t Rn, uint32_t Rm, flags_t *bandera)	29
5.8.3.9	EORS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	29
5.8.3.10	LSLS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	30
5.8.3.11	LSRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	30
5.8.3.12	MOV(uint32_t *Rd, uint32_t Rm)	30
5.8.3.13	MOVS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	30
5.8.3.14	MULS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	31
5.8.3.15	MVNS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	31
5.8.3.16	NOP()	31
5.8.3.17	ORRS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	31
5.8.3.18	REV(uint32_t *Rd, uint32_t Rm)	32
5.8.3.19	REV16(uint32_t *Rd, uint32_t Rm)	32
5.8.3.20	REVSH(uint32_t *Rd, uint32_t Rm)	32
5.8.3.21	RORS(uint32_t *Rd, uint32_t Rm, flags_t *bandera)	32
5.8.3.22	RSBS(uint32_t *Rd, uint32_t Rn, flags_t *bandera)	33
5.8.3.23	SBCS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	33
5.8.3.24	SUB(uint32_t *Rd, uint32_t Rn, uint32_t Rm)	33
5.8.3.25	SUBS(uint32_t *Rd, uint32_t Rn, uint32_t Rm, flags_t *bandera)	33
5.8.3.26	TST(uint32_t Rn, uint32_t Rm, flags_t *bandera)	34
5.9	Referencia del Archivo instrucciones_salto.c	34
5.9.1	Descripción detallada	35
5.9.2	Documentación de las funciones	35
5.9.2.1	B(uint32_t *registro, int salto)	35
5.9.2.2	BAL(uint32_t *registro, int salto, flags_t bandera)	35
5.9.2.3	BCC(uint32_t *registro, int salto, flags_t bandera)	36
5.9.2.4	BCS(uint32_t *registro, int salto, flags_t bandera)	36
5.9.2.5	BEQ(uint32_t *registro, int salto, flags_t bandera)	36
5.9.2.6	BGE(uint32_t *registro, int salto, flags_t bandera)	36
5.9.2.7	BGT(uint32_t *registro, int salto, flags_t bandera)	37
5.9.2.8	BHI(uint32_t *registro, int salto, flags_t bandera)	37

5.9.2.9	BL(uint32_t *registro, int salto)	37
5.9.2.10	BLE(uint32_t *registro, int salto, flags_t bandera)	37
5.9.2.11	BLS(uint32_t *registro, int salto, flags_t bandera)	38
5.9.2.12	BLT(uint32_t *registro, int salto, flags_t bandera)	38
5.9.2.13	BLX(uint32_t *registro, uint32_t R)	38
5.9.2.14	BMI(uint32_t *registro, int salto, flags_t bandera)	38
5.9.2.15	BNE(uint32_t *registro, int salto, flags_t bandera)	39
5.9.2.16	BPL(uint32_t *registro, int salto, flags_t bandera)	39
5.9.2.17	BVC(uint32_t *registro, int salto, flags_t bandera)	39
5.9.2.18	BVS(uint32_t *registro, int salto, flags_t bandera)	39
5.9.2.19	BX(uint32_t *registro, uint32_t R)	40
5.10	Referencia del Archivo instrucciones_salto.h	40
5.10.1	Descripción detallada	41
5.10.2	Documentación de las funciones	41
5.10.2.1	B(uint32_t *registro, int salto)	41
5.10.2.2	BAL(uint32_t *registro, int salto, flags_t bandera)	41
5.10.2.3	BCC(uint32_t *registro, int salto, flags_t bandera)	41
5.10.2.4	BCS(uint32_t *registro, int salto, flags_t bandera)	42
5.10.2.5	BEQ(uint32_t *registro, int salto, flags_t bandera)	42
5.10.2.6	BGE(uint32_t *registro, int salto, flags_t bandera)	42
5.10.2.7	BGT(uint32_t *registro, int salto, flags_t bandera)	42
5.10.2.8	BHI(uint32_t *registro, int salto, flags_t bandera)	43
5.10.2.9	BL(uint32_t *registro, int salto)	43
5.10.2.10	BLE(uint32_t *registro, int salto, flags_t bandera)	43
5.10.2.11	BLS(uint32_t *registro, int salto, flags_t bandera)	43
5.10.2.12	BLT(uint32_t *registro, int salto, flags_t bandera)	44
5.10.2.13	BLX(uint32_t *registro, uint32_t R)	44
5.10.2.14	BMI(uint32_t *registro, int salto, flags_t bandera)	44
5.10.2.15	BNE(uint32_t *registro, int salto, flags_t bandera)	44
5.10.2.16	BPL(uint32_t *registro, int salto, flags_t bandera)	45
5.10.2.17	BVC(uint32_t *registro, int salto, flags_t bandera)	45
5.10.2.18	BVS(uint32_t *registro, int salto, flags_t bandera)	45
5.10.2.19	BX(uint32_t *registro, uint32_t R)	45
5.11	Referencia del Archivo io.c	46
5.11.1	Descripción detallada	46
5.11.2	Documentación de las funciones	46
5.11.2.1	changePinPortA(uint8_t pin, uint8_t value)	46
5.11.2.2	changePinPortB(uint8_t pin, uint8_t value)	47
5.11.2.3	initIO(void)	47
5.11.2.4	IOAccess(uint8_t address, uint8_t *data, uint8_t r_w)	47

5.11.2.5	showFrame(int x, int y, int w, int h)	47
5.11.2.6	showPorts(void)	48
5.11.3	Documentación de las variables	48
5.11.3.1	irq	48
5.11.3.2	PORTA	48
5.11.3.3	PORTB	48
5.12	Referencia del Archivo io.h	48
5.12.1	Descripción detallada	49
5.12.2	Documentación de los 'defines'	49
5.12.2.1	BLUEBLACK	49
5.12.2.2	HIGH	49
5.12.2.3	LOW	49
5.12.2.4	Read	49
5.12.2.5	REDBLACK	49
5.12.2.6	WHITEBLACK	50
5.12.2.7	Write	50
5.12.2.8	XINIT	50
5.12.2.9	YINIT	50
5.12.3	Documentación de las funciones	50
5.12.3.1	changePinPortA(uint8_t pin, uint8_t value)	50
5.12.3.2	changePinPortB(uint8_t pin, uint8_t value)	50
5.12.3.3	initIO(void)	50
5.12.3.4	IOAccess(uint8_t address, uint8_t *data, uint8_t r_w)	50
5.12.3.5	showFrame(int x, int y, int w, int h)	51
5.12.3.6	showPorts(void)	51
5.13	Referencia del Archivo main.c	51
5.13.1	Descripción detallada	52
5.13.2	Documentación de las funciones	52
5.13.2.1	main()	52
5.13.3	Documentación de las variables	52
5.13.3.1	irq	52
5.14	Referencia del Archivo nvic.c	52
5.14.1	Descripción detallada	52
5.14.2	Documentación de las funciones	52
5.14.2.1	NVIC(uint8_t *interrupcion, bool *FlagInt, uint32_t *registro, flags_t *bandera, uint8_t *SRAM)	52
5.14.2.2	POPP(uint32_t *registro, uint8_t *SRAM, flags_t *bandera)	53
5.14.2.3	PUSHH(uint32_t *registro, uint8_t *SRAM, flags_t *bandera)	53
5.15	Referencia del Archivo nvic.h	53
5.15.1	Descripción detallada	54

5.15.2 Documentación de las funciones	54
5.15.2.1 NVIC(uint8_t *interrupcion, bool *FlagInt, uint32_t *registro, flags_t *bandera, uint8_t *SRAM)	54
5.15.2.2 POPP(uint32_t *registro, uint8_t *SRAM, flags_t *bandera)	54
5.15.2.3 PUSHH(uint32_t *registro, uint8_t *SRAM, flags_t *bandera)	54
5.16 Referencia del Archivo ram.c	54
5.16.1 Descripción detallada	55
5.16.2 Documentación de las funciones	55
5.16.2.1 LDR(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	55
5.16.2.2 LDRB(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	55
5.16.2.3 LDRH(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	56
5.16.2.4 LDRSB(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	56
5.16.2.5 LDRSH(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	56
5.16.2.6 POP(uint32_t *registro, uint8_t *SRAM, uint8_t *registers_list)	57
5.16.2.7 PUSH(uint32_t *registro, uint8_t *SRAM, uint8_t *registers_list)	58
5.16.2.8 STR(uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	58
5.16.2.9 STRB(uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	58
5.16.2.10 STRH(uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	58
5.17 Referencia del Archivo ram.h	59
5.17.1 Descripción detallada	59
5.17.2 Documentación de las funciones	59
5.17.2.1 LDR(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	59
5.17.2.2 LDRB(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	60
5.17.2.3 LDRH(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	60
5.17.2.4 LDRSB(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	60
5.17.2.5 LDRSH(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	61
5.17.2.6 POP(uint32_t *registro, uint8_t *SRAM, uint8_t *registers_list)	62
5.17.2.7 PUSH(uint32_t *registro, uint8_t *SRAM, uint8_t *registers_list)	62
5.17.2.8 STR(uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	62
5.17.2.9 STRB(uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	62
5.17.2.10 STRH(uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)	63

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.

Ademas cuenta con la emulacion de los distintos modulos existentes en el Micropocesador, como lo son el modulo NVIC el cual se encarga de administrar las interrupciones en el sistema y el modulo para utilizar los puertos de entrada y de salida. La inclusion de estos modulos convierten el emulador en un Microcontralor dando la posibilidad de disponer de pines de entrada y salida para la conexion de sistemas externos que podrian ser utilizados para sensores y/o actuadores en la practica.

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
port_t	Estructura en donde los miembros definen los puertos de entrada y salida	9

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_saltos.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	34
instrucciones_saltos.h	Documento en donde estan definidas los tipos de saltos	40
io.c	Documento utilizado modificar el estado de PORT, Pins, DDR y PIN para administrar los puertos de entrada y salida	46
io.h	Documento en el cual se define la estructura port_t utilizada para representar los puertos de entrada y salida, tambien se definen funciones para modificar los miembros de la estructura port_t	48
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	51

nvic.c	Documento que contiene el desarrollo de las funciones encargadas de detectar las interrupciones y guardar registros especificos y banderas en la SRAM cuando empieza la interrupcion, y extraerlas de la SRAM cuando termina esta	52
nvic.h	Documento utilizado para definir las funciones que actuan en las interrupciones	53
ram.c	Documento en el cual se desarrollan las funciones relacionadas con el manejo de RAM	54
ram.h	Documento utilizado para definir las funciones de manejo de la RAM	59

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](#)
- uint8_t [registers_list](#) [16]

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

4.3.2.8. uint8_t registers_list[16]

arreglo de ceros y unos

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

- [decoder.h](#)

4.4. Referencia de la Estructura port_t

Estructura en donde los miembros definen los puertos de entrada y salida.

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

Campos de datos

- uint8_t [DDR](#)
- uint8_t [PORT](#)
- uint8_t [PIN](#)
- uint8_t [Pins](#)
- uint8_t [Interrupts](#)

4.4.1. Descripción detallada

Estructura en donde los miembros definen los puertos de entrada y salida.

4.4.2. Documentación de los campos

4.4.2.1. uint8_t DDR

Indica cuales son los pines de entrada y salida

4.4.2.2. uint8_t Interrupts

Registro que indica si se pueden realizar o no las interrupciones

4.4.2.3. uint8_t PIN

Lee tanto las entradas como salidas

4.4.2.4. uint8_t Pins

Pines externos a los puertos de entrada y salida

4.4.2.5. uint8_t PORT

Registro que permite escribir en las salidas

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

- [io.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, uint8_t *SRAM, uint16_t *codificacion, char **Flash)
- [instruction_t getInstruction](#) (char *instStr)

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

Variables

- uint8_t [irq](#) [16]

Arreglo que indica las interrupciones activas.

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, uint8_t * SRAM, uint16_t * codificacion, char ** Flash)`

5.5.2.3. `instruction_t getInstruction (char * instStr)`

Obtiene la instrucción separada por partes.

Parámetros

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

Devuelve

`instruction_t` la instrucción separada por partes.

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

5.5.3. Documentación de las variables

5.5.3.1. `irq[16]`

Arreglo que indica las interrupciones activas.

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 "io.h"
#include "instrucciones.h"
#include "instrucciones_saltos.h"
#include "ram.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, uint8_t *RAM, uint16_t *codificacion, char **Flash)`
- `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, uint8_t * RAM, uint16_t * codificacion, char ** Flash)

5.6.2.3. instruction_t getInstruction (char * instStr)

Obtiene la instrucción separada por partes.

Parámetros

<code>instStr</code>	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, ademas 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 `ADD` (uint32_t *Rd, uint32_t Rn, uint32_t Rm)
*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 **MULS** (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 **SUB** (uint32_t *Rd, uint32_t Rn, uint32_t Rm)
*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 instruccion, ademas se definen cuales banderas se modifican con cada instruccion.

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 ADD (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*)

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

Devuelve

La funcion no tiene retorno

5.7.2.3. 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

5.7.2.4. 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

5.7.2.5. 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

5.7.2.6. 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

5.7.2.7. 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

5.7.2.8. 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

5.7.2.9. 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

5.7.2.10. void LSL (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

5.7.2.11. void LSR (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

5.7.2.12. 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

5.7.2.13. 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

5.7.2.14. void MULS (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

5.7.2.15. 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

5.7.2.16. void NOP ()

funcion que no realiza nada en un ciclo de reloj

Devuelve

La funcion no tiene retorno

5.7.2.17. 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

5.7.2.18. 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

5.7.2.19. 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

5.7.2.20. 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

5.7.2.21. 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

5.7.2.22. 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

5.7.2.23. 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

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

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

Devuelve

La funcion no tiene retorno

5.7.2.25. 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

5.7.2.26. 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 [ADD](#) (uint32_t *Rd, uint32_t Rn, uint32_t Rm)
*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 **MULS** (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 **SUB** (uint32_t *Rd, uint32_t Rn, uint32_t Rm)
*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 ADD (uint32_t * Rd, uint32_t Rn, uint32_t Rm)

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

Devuelve

La funcion no tiene retorno

5.8.3.3. 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

5.8.3.4. 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

5.8.3.5. 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

5.8.3.6. 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

5.8.3.7. 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

5.8.3.8. 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

5.8.3.9. 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

5.8.3.10. void LSL (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

5.8.3.11. void LSR (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

5.8.3.12. 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

5.8.3.13. 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

5.8.3.14. void MULS (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

5.8.3.15. 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

5.8.3.16. void NOP ()

funcion que no realiza nada en un ciclo de reloj

Devuelve

La funcion no tiene retorno

5.8.3.17. 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

5.8.3.18. 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

5.8.3.19. 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

5.8.3.20. 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

5.8.3.21. 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

5.8.3.22. 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

5.8.3.23. 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

5.8.3.24. void SUB (uint32_t * *Rd*, uint32_t *Rn*, uint32_t *Rm*)

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

Devuelve

La funcion no tiene retorno

5.8.3.25. 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

5.8.3.26. 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 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 lineas 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 lineas 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 lineas 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 lineas 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 lineas 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 líneas 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 líneas 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 líneas 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 líneas 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 lineas 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 lineas 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 lineas 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 lineas 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 lineas 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 lineas 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 lineas 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 lineas 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 posición a saltar

Devuelve

La función no tiene retorno

5.11. Referencia del Archivo io.c

Documento utilizado para modificar el estado de PORT, Pins, DDR y PIN para administrar los puertos de entrada y salida.

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

Funciones

- void [initIO](#) (void)
Función que determina las condiciones iniciales.
- void [changePinPortA](#) (uint8_t pin, uint8_t value)
La función que cambia el miembro Pins, y activa las interrupciones dependiendo de ciertas condiciones.
- void [changePinPortB](#) (uint8_t pin, uint8_t value)
La función que cambia el miembro Pins, y activa las interrupciones dependiendo de ciertas condiciones.
- void [IOAccess](#) (uint8_t address, uint8_t *data, uint8_t r_w)
La función escribe o lee los miembros de la estructura [port_t](#).
- void [showPorts](#) (void)
Función que muestra ordenadamente los miembros de PORTA Y PORTB.
- void [showFrame](#) (int x, int y, int w, int h)
Función utilizada por la función showPorts.

Variables

- [port_t PORTA](#)
Puerto A.
- [port_t PORTB](#)
Puerto B.
- uint8_t [irq](#) [16]
Arreglo que indica las interrupciones activas.

5.11.1. Descripción detallada

Documento utilizado para modificar el estado de PORT, Pins, DDR y PIN para administrar los puertos de entrada y salida.

5.11.2. Documentación de las funciones

5.11.2.1. void changePinPortA (uint8_t pin, uint8_t value)

La función que cambia el miembro Pins, y activa las interrupciones dependiendo de ciertas condiciones.

Parámetros

<i>pin</i>	indica el pin del miembro Pins a cambiar
<i>value</i>	alto o bajo

Devuelve

La funcion no retorna nada

5.11.2.2. void changePinPortB (uint8_t *pin*, uint8_t *value*)

La funcion que cambia el miembro Pins, y activa las interrupciones dependiendo de ciertas condiciones.

Parámetros

<i>pin</i>	indica el pin del miembro Pins a cambiar
<i>value</i>	alto o bajo

Devuelve

La funcion no retorna nada

5.11.2.3. void initIO (void)

Funcion que determina las condiciones iniciales.

Devuelve

La funcion no retorna nada

5.11.2.4. void IOAccess (uint8_t *address*, uint8_t * *data*, uint8_t *r_w*)

La funcion escribe o lee los miembros de la estructura [port_t](#).

Parámetros

<i>address</i>	indica la posicion del miembro a leer o a escribir
<i>data</i>	registro que se leera en donde se guardara el miembro indicado por la direccion
<i>r_w</i>	indica si se lee o se escribe data

Devuelve

La funcion no retorna nada

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

Funcion utilizada por la funcion showPorts.

Devuelve

La funcion no retorna nada

5.11.2.6. void showPorts (void)

Funcion que muestra ordenamente los miembros de PORTA Y PORTB.

Devuelve

La funcion no retorna nada

5.11.3. Documentación de las variables

5.11.3.1. uint8_t irq[16]

Arreglo que indica las interrupciones activas.

5.11.3.2. PORTA

Puerto A.

5.11.3.3. PORTB

Puerto B.

5.12. Referencia del Archivo io.h

Documento en el cual se define la estructura `port_t` utilizada para representar los puertos de entrada y salida, tambien se definen funciones para modificar los miembros de la estructura `port_t`.

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

Estructuras de datos

- struct `port_t`

Estructura en donde los miembros definen los puertos de entrada y salida.

'defines'

- #define `XINIT` 10

Posicion en x para mostrar puertos.

- #define `YINIT` 25

Posicion en y para mostrar puertos.

- #define `HIGH` 1

Valor que indica el estado del pin en 1.

- #define `LOW` 0

Valor que indica el estado del pin en 0.

- #define `Read` 1

Dato que indica la lectura en la funcion IOAccess.

- #define `Write` 0

Dato que indica la escritura en la funcion IOAccess.

- #define `BLUEBLACK` 10

Indica configuracion de texto azul y fondo negro.

- `#define REDBLACK 20`

Indica configuracion de texto rojo y fondo negro.

- `#define WHITEBLACK 30`

Indica configuracion de texto blanco y fondo blanco.

Funciones

- void `IOAccess` (uint8_t address, uint8_t *data, uint8_t r_w)

La funcion escribe o lee los miembros de la estructura `port_t`.

- void `changePinPortA` (uint8_t pin, uint8_t value)

La funcion que cambia el miembro Pins, y activa las interrupciones dependiendo de ciertas condiciones.

- void `changePinPortB` (uint8_t pin, uint8_t value)

La funcion que cambia el miembro Pins, y activa las interrupciones dependiendo de ciertas condiciones.

- void `initIO` (void)

Funcion que determina las condiciones iniciales.

- void `showPorts` (void)

Funcion que muestra ordenamente los miembros de PORTA Y PORTB.

- void `showFrame` (int x, int y, int w, int h)

Funcion utilizada por la funcion showPorts.

5.12.1. Descripción detallada

Documento en el cual se define la estructura `port_t` utilizada para representar los puertos de entrada y salida, tambien se definen funciones para modificar los miembros de la estructura `port_t`.

5.12.2. Documentación de los 'defines'

5.12.2.1. `#define BLUEBLACK 10`

Indica configuracion de texto azul y fondo negro.

5.12.2.2. `#define HIGH 1`

Valor que indica el estado del pin en 1.

5.12.2.3. `#define LOW 0`

Valor que indica el estado del pin en 0.

5.12.2.4. `#define Read 1`

Dato que indica la lectura en la funcion IOAccess.

5.12.2.5. `#define REDBLACK 20`

Indica configuracion de texto rojo y fondo negro.

5.12.2.6. `#define WHITEBLACK 30`

Indica configuracion de texto blanco y fondo blanco.

5.12.2.7. `#define Write 0`

Dato que indica la escritura en la funcion IOAccess.

5.12.2.8. `#define XINIT 10`

Posiciion en x para mostrar puertos.

5.12.2.9. `#define YINIT 25`

Posiciion en y para mostrar puertos.

5.12.3. Documentación de las funciones

5.12.3.1. `void changePinPortA (uint8_t pin, uint8_t value)`

La funcion que cambia el miembro Pins, y activa las interrupciones dependiendo de ciertas condiciones.

Parámetros

<i>pin</i>	indica el pin del miembro Pins a cambiar
<i>value</i>	alto o bajo

Devuelve

La funcion no retorna nada

5.12.3.2. `void changePinPortB (uint8_t pin, uint8_t value)`

La funcion que cambia el miembro Pins, y activa las interrupciones dependiendo de ciertas condiciones.

Parámetros

<i>pin</i>	indica el pin del miembro Pins a cambiar
<i>value</i>	alto o bajo

Devuelve

La funcion no retorna nada

5.12.3.3. `void initIO (void)`

Funcion que determina las condiciones iniciales.

Devuelve

La funcion no retorna nada

5.12.3.4. `void IOAccess (uint8_t address, uint8_t * data, uint8_t r_w)`

La funcion escribe o lee los miembros de la estructura [port_t](#).

Parámetros

<i>address</i>	indica la posicion del miembro a leer o a escribir
<i>data</i>	registro que se leera en donde se guardara el miembro indicado por la direccion
<i>r_w</i>	indica si se lee o se escribe data

Devuelve

La funcion no retorna nada

5.12.3.5. void showFrame (int x, int y, int w, int h)

Funcion utilizada por la funcion showPorts.

Devuelve

La funcion no retorna nada

5.12.3.6. void showPorts (void)

Funcion que muestra ordenamente los miembros de PORTA Y PORTB.

Devuelve

La funcion no retorna nada

5.13. 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"
#include "nvic.h"
#include "io.h"
```

Funciones

- int [main](#) ()

Variables

- uint8_t [irq](#) [16]

Arreglo que indica las interrupciones activas.

5.13.1. 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.

5.13.2. Documentación de las funciones

5.13.2.1. `int main ()`

5.13.3. Documentación de las variables

5.13.3.1. `uint8_t irq[16]`

Arreglo que indica las interrupciones activas.

5.14. Referencia del Archivo nvic.c

Documento que contiene el desarrollo de las funciones encargadas de detectar las interrupciones y guardar registros especificos y banderas en la SRAM cuando empieza la interrupcion, y extraerlas de la SRAM cuando termina esta.

```
#include "nvic.h"
```

Funciones

- void **NVIC** (`uint8_t *interrupcion`, `bool *FlagInt`, `uint32_t *registro`, `flags_t *bandera`, `uint8_t *SRAM`)
funcion utilizada para detectar interrupciones
- void **PUSHH** (`uint32_t *registro`, `uint8_t *SRAM`, `flags_t *bandera`)
funcion PUSH modificada para guardar registros especificos y banderas
- void **POPP** (`uint32_t *registro`, `uint8_t *SRAM`, `flags_t *bandera`)
funcion POP modificada para guardar registros especificos y banderas

5.14.1. Descripción detallada

Documento que contiene el desarrollo de las funciones encargadas de detectar las interrupciones y guardar registros especificos y banderas en la SRAM cuando empieza la interrupcion, y extraerlas de la SRAM cuando termina esta.

5.14.2. Documentación de las funciones

5.14.2.1. `void NVIC (uint8_t * interrupcion, bool * FlagInt, uint32_t * registro, flags_t * bandera, uint8_t * SRAM)`

funcion utilizada para detectar interrupciones

Parámetros

<i>interrupcion</i>	puntero que define la veracidad de la interrupcion
<i>FlagInt</i>	indicador de banderas

<i>registro</i>	Puntero con la primera posicion de los registros
<i>flags_t</i>	Puntero para operar la estructura de banderas
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

5.14.2.2. void POPP (uint32_t * *registro*, uint8_t * *SRAM*, flags_t * *bandera*)

funcion POP modificada para guardar registros especificos y banderas

Parámetros

<i>registro</i>	Puntero con la primera posicion de los registros
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM
<i>flags_t</i>	Puntero para operar la estructura de banderas

Devuelve

la funcion no retorna nada

5.14.2.3. void PUSHH (uint32_t * *registro*, uint8_t * *SRAM*, flags_t * *bandera*)

funcion PUSH modificada para guardar registros especificos y banderas

Parámetros

<i>registro</i>	Puntero con la primera posicion de los registros
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM
<i>flags_t</i>	Puntero para operar la estructura de banderas

Devuelve

la funcion no retorna nada

5.15. Referencia del Archivo nvic.h

Documento utilizado para definir las funciones que actuan en las interrupciones.

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

Funciones

- void **NVIC** (uint8_t *interrupcion, bool *FlagInt, uint32_t *registro, flags_t *bandera, uint8_t *SRAM)
funcion utilizada para detectar interrupciones
- void **PUSHH** (uint32_t *registro, uint8_t *SRAM, flags_t *bandera)
funcion PUSH modificada para guardar registros especificos y banderas
- void **POPP** (uint32_t *registro, uint8_t *SRAM, flags_t *bandera)
funcion POP modificada para guardar registros especificos y banderas

5.15.1. Descripción detallada

Documento utilizado para definir las funciones que actuan en las interrupciones.

5.15.2. Documentación de las funciones

5.15.2.1. void NVIC (uint8_t * *interrupcion*, bool * *FlagInt*, uint32_t * *registro*, flags_t * *bandera*, uint8_t * *SRAM*)

funcion utilizada para detectar interrupciones

Parámetros

<i>interrupcion</i>	puntero que define la veracidad de la interrupcion
<i>FlagInt</i>	indicador de banderas
<i>registro</i>	Puntero con la primera posicion de los registros
<i>flags_t</i>	Puntero para operar la estructura de banderas
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

5.15.2.2. void POPP (uint32_t * *registro*, uint8_t * *SRAM*, flags_t * *bandera*)

funcion POP modificada para guardar registros especificos y banderas

Parámetros

<i>registro</i>	Puntero con la primera posicion de los registros
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM
<i>flags_t</i>	Puntero para operar la estructura de banderas

Devuelve

la funcion no retorna nada

5.15.2.3. void PUSHH (uint32_t * *registro*, uint8_t * *SRAM*, flags_t * *bandera*)

funcion PUSH modificada para guardar registros especificos y banderas

Parámetros

<i>registro</i>	Puntero con la primera posicion de los registros
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM
<i>flags_t</i>	Puntero para operar la estructura de banderas

Devuelve

la funcion no retorna nada

5.16. Referencia del Archivo ram.c

Documento en el cual se desarrollan las funciones relacionadas con el manejo de RAM.

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


Funciones

- void **PUSH** (uint32_t *registro, uint8_t *SRAM, uint8_t *registers_list)
funcion utilizada para guardar registros en la SRAM
- void **POP** (uint32_t *registro, uint8_t *SRAM, uint8_t *registers_list)
funcion utilizada para extraer registros en la SRAM
- void **LDR** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para la carga de 2 bytes de la RAM
- void **LDRB** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para la carga de 4 bits de la RAM
- void **LDRH** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para la carga de 1 byte de la RAM
- void **LDRSB** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para la carga de 4 bits de la RAM y hacer extension de signo
- void **LDRSH** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para la carga de 1 byte de la RAM y hacer extension de signo
- void **STR** (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para el almacenamiento de 2 bytes de la RAM
- void **STRB** (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para el almacenamiento de 4 bits de la RAM
- void **STRH** (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para el almacenamiento de 1 byte de la RAM

5.16.1. Descripción detallada

Documento en el cual se desarrollan las funciones relacionadas con el manejo de RAM.

5.16.2. Documentación de las funciones

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

funcion utilizada para la carga de 2 bytes de la RAM

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para la carga de 4 bits de la RAM

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para la carga de 1 byte de la RAM

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para la carga de 4 bits de la RAM y hacer extension de signo

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para la carga de 1 byte de la RAM y hacer extension de signo

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

5.16.2.6. void POP (uint32_t * *registro*, uint8_t * *SRAM*, uint8_t * *registers_list*)

funcion utilizada para extraer registros en la SRAM

Parámetros

<i>registro</i>	puntero del primer elemento del arreglo registro
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM
<i>registers_list</i>	puntero del primer elemento del arreglo de unos y ceros, que contiene los registros a extraer

Devuelve

la funcion no retorna nada

5.16.2.7. void PUSH (uint32_t * *registro*, uint8_t * *SRAM*, uint8_t * *registers_list*)

funcion utilizada para guardar registros en la SRAM

Parámetros

<i>registro</i>	puntero del primer elemento del arreglo registro
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM
<i>registers_list</i>	puntero del primer elemento del arreglo de unos y ceros, que contiene los registros a guardar

Devuelve

la funcion no retorna nada

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

funcion utilizada para el almacenamiento de 2 bytes de la RAM

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para el almacenamiento de 4 bits de la RAM

Parámetros

<i>Rt</i>	registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para el almacenamiento de 1 byte de la RAM

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

5.17. Referencia del Archivo ram.h

Documento utilizado para definir las funciones de manejo de la RAM.

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

Funciones

- void **PUSH** (uint32_t *registro, uint8_t *SRAM, uint8_t *registers_list)
funcion utilizada para guardar registros en la SRAM
- void **POP** (uint32_t *registro, uint8_t *SRAM, uint8_t *registers_list)
funcion utilizada para extraer registros en la SRAM
- void **LDR** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para la carga de 2 bytes de la RAM
- void **LDRB** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para la carga de 4 bits de la RAM
- void **LDRH** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para la carga de 1 byte de la RAM
- void **LDRSB** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para la carga de 4 bits de la RAM y hacer extension de signo
- void **LDRSH** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para la carga de 1 byte de la RAM y hacer extension de signo
- void **STR** (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para el almacenamiento de 2 bytes de la RAM
- void **STRB** (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para el almacenamiento de 4 bits de la RAM
- void **STRH** (uint32_t Rt, uint32_t Rn, uint32_t Rm, uint8_t *SRAM)
funcion utilizada para el almacenamiento de 1 byte de la RAM

5.17.1. Descripción detallada

Documento utilizado para definir las funciones de manejo de la RAM.

5.17.2. Documentación de las funciones

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

funcion utilizada para la carga de 2 bytes de la RAM

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para la carga de 4 bits de la RAM

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para la carga de 1 byte de la RAM

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para la carga de 4 bits de la RAM y hacer extension de signo

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para la carga de 1 byte de la RAM y hacer extension de signo

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

5.17.2.6. void POP (uint32_t * *registro*, uint8_t * *SRAM*, uint8_t * *registers_list*)

funcion utilizada para extraer registros en la SRAM

Parámetros

<i>registro</i>	puntero del primer elemento del arreglo registro
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM
<i>registers_list</i>	puntero del primer elemento del arreglo de unos y ceros, que contiene los registros a extraer

Devuelve

la funcion no retorna nada

5.17.2.7. void PUSH (uint32_t * *registro*, uint8_t * *SRAM*, uint8_t * *registers_list*)

funcion utilizada para guardar registros en la SRAM

Parámetros

<i>registro</i>	puntero del primer elemento del arreglo registro
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM
<i>registers_list</i>	puntero del primer elemento del arreglo de unos y ceros, que contiene los registros a guardar

Devuelve

la funcion no retorna nada

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

funcion utilizada para el almacenamiento de 2 bytes de la RAM

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para el almacenamiento de 4 bits de la RAM

Parámetros

<i>Rt</i>	registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

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

funcion utilizada para el almacenamiento de 1 byte de la RAM

Parámetros

<i>Rt</i>	Registro para almacenar la direccion
<i>Rn</i>	registro operando
<i>Rm</i>	registro a operar
<i>SRAM</i>	puntero del primer elemento del arreglo SRAM

Devuelve

la funcion no retorna nada

