

Proyecto de Micros

Generado por Doxygen 1.8.10

Domingo, 18 de Octubre de 2015 05:54:59

Índice general

| | | |
|----------|---|----------|
| 1 | Índice de estructura de datos | 1 |
| 1.1 | Estructura de datos | 1 |
| 2 | Índice de archivos | 3 |
| 2.1 | Lista de archivos | 3 |
| 3 | Documentación de las estructuras de datos | 5 |
| 3.1 | Referencia de la Estructura <code>ins_t</code> | 5 |
| 3.1.1 | Documentación de los campos | 5 |
| 3.1.1.1 | array | 5 |
| 3.2 | Referencia de la Estructura <code>instruction_t</code> | 5 |
| 3.2.1 | Documentación de los campos | 5 |
| 3.2.1.1 | mnemonic | 5 |
| 3.2.1.2 | op1_type | 6 |
| 3.2.1.3 | op1_value | 6 |
| 3.2.1.4 | op2_type | 6 |
| 3.2.1.5 | op2_value | 6 |
| 3.2.1.6 | op3_type | 6 |
| 3.2.1.7 | op3_value | 6 |
| 3.2.1.8 | registers_list | 6 |
| 3.3 | Referencia de la Estructura <code>port_t</code> | 6 |
| 3.3.1 | Documentación de los campos | 6 |
| 3.3.1.1 | DDR | 6 |
| 3.3.1.2 | Interrupts | 6 |
| 3.3.1.3 | PIN | 6 |
| 3.3.1.4 | Pins | 6 |
| 3.3.1.5 | PORT | 6 |
| 4 | Documentación de archivos | 7 |
| 4.1 | Referencia del Archivo <code>alu.c</code> | 7 |
| 4.1.1 | Documentación de las funciones | 8 |
| 4.1.1.1 | <code>ADCS(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)</code> | 8 |

| | | |
|----------|--|----|
| 4.1.1.2 | ADD(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 9 |
| 4.1.1.3 | ADDS(uint32_t *Rd, uint32_t *Rm, uint32_t Rn) | 10 |
| 4.1.1.4 | AND(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 10 |
| 4.1.1.5 | ASRS(uint32_t *Rm, uint32_t Rn) | 10 |
| 4.1.1.6 | BICS(uint32_t *Rm, uint32_t Rn) | 10 |
| 4.1.1.7 | CMN(uint32_t *Rm, uint32_t *Rn) | 11 |
| 4.1.1.8 | CMP(uint32_t *Rm, uint32_t *Rn) | 11 |
| 4.1.1.9 | EOR(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 11 |
| 4.1.1.10 | LSL(uint32_t *Rm, uint32_t *Rn) | 11 |
| 4.1.1.11 | LSLS(uint32_t *Rd, uint32_t *Rm, uint32_t Rn) | 11 |
| 4.1.1.12 | LSR(uint32_t *Rm, uint32_t *Rn) | 12 |
| 4.1.1.13 | LSRS(uint32_t *Rd, uint32_t *Rm, uint32_t Rn) | 13 |
| 4.1.1.14 | MOV(uint32_t *Rm, uint32_t *Rn) | 13 |
| 4.1.1.15 | MOVS(uint32_t *Rm, uint32_t Rn) | 13 |
| 4.1.1.16 | MULS(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 13 |
| 4.1.1.17 | MVN(uint32_t *Rm, uint32_t *Rn) | 13 |
| 4.1.1.18 | NOP(void) | 14 |
| 4.1.1.19 | obtenerBandera(bool *bands) | 14 |
| 4.1.1.20 | ORR(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 14 |
| 4.1.1.21 | REV(uint32_t *Rm, uint32_t *Rn) | 14 |
| 4.1.1.22 | REV16(uint32_t *Rm, uint32_t *Rn) | 14 |
| 4.1.1.23 | RORS(uint32_t *Rm, uint32_t Rn) | 14 |
| 4.1.1.24 | RSBS(uint32_t *Rm, uint32_t *Rn) | 15 |
| 4.1.1.25 | SalvarBanderas(bool *bands) | 15 |
| 4.1.1.26 | SBC(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 15 |
| 4.1.1.27 | SUB(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 15 |
| 4.1.1.28 | SUBS(uint32_t *Rd, uint32_t *Rm, uint32_t Rn) | 15 |
| 4.1.1.29 | TST(uint32_t *Rm, uint32_t *Rn) | 16 |
| 4.1.2 | Documentación de las variables | 16 |
| 4.1.2.1 | banderas | 16 |
| 4.1.2.2 | comp | 16 |
| 4.2 | Referencia del Archivo alu.h | 16 |
| 4.2.1 | Documentación de las funciones | 17 |
| 4.2.1.1 | ADCS(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 17 |
| 4.2.1.2 | ADD(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 18 |
| 4.2.1.3 | ADDS(uint32_t *Rd, uint32_t *Rm, uint32_t Rn) | 18 |
| 4.2.1.4 | AND(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 18 |
| 4.2.1.5 | ASRS(uint32_t *Rm, uint32_t Rn) | 18 |
| 4.2.1.6 | BICS(uint32_t *Rm, uint32_t Rn) | 19 |
| 4.2.1.7 | CMN(uint32_t *Rm, uint32_t *Rn) | 19 |

| | | |
|----------|--|----|
| 4.2.1.8 | CMP(uint32_t *Rm, uint32_t *Rn) | 19 |
| 4.2.1.9 | EOR(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 19 |
| 4.2.1.10 | LSL(uint32_t *Rm, uint32_t *Rn) | 19 |
| 4.2.1.11 | LSLS(uint32_t *Rd, uint32_t *Rm, uint32_t Rn) | 20 |
| 4.2.1.12 | LSR(uint32_t *Rm, uint32_t *Rn) | 21 |
| 4.2.1.13 | LSRS(uint32_t *Rd, uint32_t *Rm, uint32_t Rn) | 21 |
| 4.2.1.14 | MOV(uint32_t *Rm, uint32_t *Rn) | 21 |
| 4.2.1.15 | MOVS(uint32_t *Rm, uint32_t Rn) | 21 |
| 4.2.1.16 | MULS(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 21 |
| 4.2.1.17 | MVN(uint32_t *Rm, uint32_t *Rn) | 22 |
| 4.2.1.18 | NOP(void) | 22 |
| 4.2.1.19 | obtenerBandera(bool *bands) | 22 |
| 4.2.1.20 | ORR(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 22 |
| 4.2.1.21 | REV(uint32_t *Rm, uint32_t *Rn) | 22 |
| 4.2.1.22 | REV16(uint32_t *Rm, uint32_t *Rn) | 22 |
| 4.2.1.23 | RORS(uint32_t *Rm, uint32_t Rn) | 23 |
| 4.2.1.24 | RSBS(uint32_t *Rm, uint32_t *Rn) | 23 |
| 4.2.1.25 | SalvarBanderas(bool *bands) | 23 |
| 4.2.1.26 | SBC(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 23 |
| 4.2.1.27 | SUB(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn) | 23 |
| 4.2.1.28 | SUBS(uint32_t *Rd, uint32_t *Rm, uint32_t Rn) | 24 |
| 4.2.1.29 | TST(uint32_t *Rm, uint32_t *Rn) | 24 |
| 4.3 | Referencia del Archivo colors.h | 24 |
| 4.3.1 | Documentación de los 'defines' | 25 |
| 4.3.1.1 | AQUA | 25 |
| 4.3.1.2 | BLACK | 25 |
| 4.3.1.3 | BLUE | 25 |
| 4.3.1.4 | BRIGHT_WHITE | 25 |
| 4.3.1.5 | GRAY | 25 |
| 4.3.1.6 | GREEN | 25 |
| 4.3.1.7 | LIGHT_AQUA | 25 |
| 4.3.1.8 | LIGHT_BLUE | 25 |
| 4.3.1.9 | LIGHT_GREEN | 25 |
| 4.3.1.10 | LIGHT_PURPLE | 25 |
| 4.3.1.11 | LIGHT_RED | 25 |
| 4.3.1.12 | LIGHT_YELLOW | 25 |
| 4.3.1.13 | PURPLE | 25 |
| 4.3.1.14 | RED | 25 |
| 4.3.1.15 | WHITE | 25 |
| 4.3.1.16 | YELLOW | 25 |

| | | |
|----------|---|----|
| 4.4 | Referencia del Archivo decoder.c | 25 |
| 4.4.1 | Documentación de las funciones | 26 |
| 4.4.1.1 | countLines(FILE *fp) | 26 |
| 4.4.1.2 | decodeInstruction(instruction_t instruction) | 26 |
| 4.4.1.3 | getInstruction(char *instStr) | 26 |
| 4.4.1.4 | iniciaram(void) | 26 |
| 4.4.1.5 | obtener_memoria(uint32_t *pcount) | 26 |
| 4.4.1.6 | obtener_registros(uint32_t *pcount) | 26 |
| 4.4.1.7 | obtenerPC(uint32_t *pcount) | 27 |
| 4.4.1.8 | readFile(char *filename, ins_t *instructions) | 27 |
| 4.4.2 | Documentación de las variables | 27 |
| 4.4.2.1 | bin | 27 |
| 4.4.2.2 | exnum | 27 |
| 4.4.2.3 | guardar | 27 |
| 4.4.2.4 | i | 27 |
| 4.4.2.5 | indicador | 27 |
| 4.4.2.6 | memoria | 27 |
| 4.4.2.7 | registers | 27 |
| 4.5 | Referencia del Archivo decoder.h | 27 |
| 4.5.1 | Documentación de las funciones | 28 |
| 4.5.1.1 | countLines(FILE *fp) | 28 |
| 4.5.1.2 | decodeInstruction(instruction_t instruction) | 28 |
| 4.5.1.3 | getInstruction(char *instStr) | 28 |
| 4.5.1.4 | iniciaram(void) | 28 |
| 4.5.1.5 | obtener_memoria(uint32_t *pcount) | 28 |
| 4.5.1.6 | obtener_registros(uint32_t *pcount) | 28 |
| 4.5.1.7 | obtenerPC(uint32_t *pcount) | 28 |
| 4.5.1.8 | POPHINTERRUPT(registers, memoria, guardar) | 28 |
| 4.5.1.9 | PUSHINTERRUPT(registers, memoria, guardar) | 29 |
| 4.5.1.10 | readFile(char *filename, ins_t *instructions) | 30 |
| 4.6 | Referencia del Archivo flags.c | 30 |
| 4.6.1 | Documentación de las funciones | 30 |
| 4.6.1.1 | flag(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn, bool *banderas, uint8_t *compar) | 30 |
| 4.6.2 | Documentación de las variables | 30 |
| 4.6.2.1 | H | 30 |
| 4.7 | Referencia del Archivo flags.h | 30 |
| 4.7.1 | Documentación de los 'defines' | 31 |
| 4.7.1.1 | C | 31 |
| 4.7.1.2 | N | 31 |
| 4.7.1.3 | V | 31 |

| | | |
|----------|---|----|
| 4.7.1.4 | Z | 31 |
| 4.7.2 | Documentación de las funciones | 31 |
| 4.7.2.1 | flag(uint32_t *Rd, uint32_t *Rm, uint32_t *Rn, bool *banderas, uint8_t *compar) | 31 |
| 4.8 | Referencia del Archivo io.c | 31 |
| 4.8.1 | Documentación de las funciones | 32 |
| 4.8.1.1 | changePinPortA(uint8_t pin, uint8_t value) | 32 |
| 4.8.1.2 | changePinPortB(uint8_t pin, uint8_t value) | 32 |
| 4.8.1.3 | initIO(void) | 32 |
| 4.8.1.4 | IOAccess(uint8_t address, uint8_t *data, uint8_t r_w) | 32 |
| 4.8.1.5 | showFrame(int x, int y, int w, int h) | 32 |
| 4.8.1.6 | showPorts(void) | 32 |
| 4.8.2 | Documentación de las variables | 32 |
| 4.8.2.1 | irq | 32 |
| 4.8.2.2 | PORTA | 32 |
| 4.8.2.3 | PORTB | 32 |
| 4.9 | Referencia del Archivo io.h | 32 |
| 4.9.1 | Documentación de los 'defines' | 33 |
| 4.9.1.1 | BLUEBLACK | 33 |
| 4.9.1.2 | HIGH | 33 |
| 4.9.1.3 | LOW | 33 |
| 4.9.1.4 | Read | 33 |
| 4.9.1.5 | REDBLACK | 33 |
| 4.9.1.6 | WHITEBLACK | 33 |
| 4.9.1.7 | Write | 33 |
| 4.9.1.8 | XINIT | 33 |
| 4.9.1.9 | YINIT | 33 |
| 4.9.2 | Documentación de las funciones | 33 |
| 4.9.2.1 | changePinPortA(uint8_t pin, uint8_t value) | 33 |
| 4.9.2.2 | changePinPortB(uint8_t pin, uint8_t value) | 33 |
| 4.9.2.3 | initIO(void) | 33 |
| 4.9.2.4 | IOAccess(uint8_t address, uint8_t *data, uint8_t r_w) | 33 |
| 4.9.2.5 | showFrame(int x, int y, int w, int h) | 33 |
| 4.9.2.6 | showPorts(void) | 33 |
| 4.10 | Referencia del Archivo main.c | 33 |
| 4.10.1 | Documentación de los 'defines' | 34 |
| 4.10.1.1 | AZUL | 34 |
| 4.10.1.2 | BLANCO | 34 |
| 4.10.1.3 | BRILLO | 34 |
| 4.10.1.4 | CELESTE | 34 |
| 4.10.1.5 | INTERMITENTE | 34 |

| | | |
|-----------|--|----|
| 4.10.1.6 | INVERSO | 34 |
| 4.10.1.7 | MARRON | 34 |
| 4.10.1.8 | MAXCAD | 34 |
| 4.10.1.9 | MAXLINEAS | 35 |
| 4.10.1.10 | NEGRO | 35 |
| 4.10.1.11 | NORMAL | 35 |
| 4.10.1.12 | ROJO | 35 |
| 4.10.1.13 | ROSA | 35 |
| 4.10.1.14 | SEMIBRILLO | 35 |
| 4.10.1.15 | VERDE | 35 |
| 4.10.2 | Documentación de las funciones | 35 |
| 4.10.2.1 | Disp(void) | 35 |
| 4.10.2.2 | Exit(void) | 35 |
| 4.10.2.3 | Inivideo(void) | 35 |
| 4.10.2.4 | IniVideo(void) | 35 |
| 4.10.2.5 | main(void) | 35 |
| 4.10.3 | Documentación de las variables | 35 |
| 4.10.3.1 | data | 35 |
| 4.10.3.2 | memoria | 35 |
| 4.10.3.3 | win | 35 |
| 4.10.3.4 | wine | 35 |
| 4.11 | Referencia del Archivo NVIC.c | 35 |
| 4.11.1 | Documentación de las funciones | 35 |
| 4.11.1.1 | NVIC_DisableIRQ(uint8_t *Exnumb, uint8_t numb) | 35 |
| 4.11.1.2 | NVIC_EnableIRQ(uint8_t *Exnumb, uint8_t numb) | 36 |
| 4.12 | Referencia del Archivo NVIC.h | 36 |
| 4.12.1 | Documentación de las funciones | 36 |
| 4.12.1.1 | NVIC_DisableIRQ(uint8_t *Exnumb, uint8_t numb) | 36 |
| 4.12.1.2 | NVIC_EnableIRQ(uint8_t *Exnumb, uint8_t numb) | 36 |
| 4.13 | Referencia del Archivo ram.c | 36 |
| 4.13.1 | Documentación de las funciones | 37 |
| 4.13.1.1 | bitcount(uint32_t *R) | 37 |
| 4.13.1.2 | inimemoria(uint32_t *memoria, int tama) | 37 |
| 4.13.1.3 | LDR(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 37 |
| 4.13.1.4 | LDRB(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 38 |
| 4.13.1.5 | LDRH(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 38 |
| 4.13.1.6 | mostrar_memoria(uint32_t *memoria, int tama) | 38 |
| 4.13.1.7 | POP(uint32_t *registros, uint32_t *memory, uint32_t *res) | 38 |
| 4.13.1.8 | POPINTERRUPT(uint32_t *registros, uint32_t *memory, uint32_t *res) | 38 |
| 4.13.1.9 | PUSH(uint32_t *registros, uint32_t *memory, uint32_t *res) | 39 |

| | | |
|-----------|---|----|
| 4.13.1.10 | PUSHINTERRUPT(uint32_t *registros, uint32_t *memory, uint32_t *res) | 39 |
| 4.13.1.11 | STR(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 39 |
| 4.13.1.12 | STRB(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 39 |
| 4.13.1.13 | STRH(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 39 |
| 4.14 | Referencia del Archivo ram.h | 40 |
| 4.14.1 | Documentación de los 'defines' | 41 |
| 4.14.1.1 | DIRMAXMEM | 41 |
| 4.14.1.2 | MEMORIA | 41 |
| 4.14.2 | Documentación de las funciones | 41 |
| 4.14.2.1 | bitcount(uint32_t *R) | 41 |
| 4.14.2.2 | inimemoria(uint32_t *memoria, int tama) | 41 |
| 4.14.2.3 | LDR(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 41 |
| 4.14.2.4 | LDRB(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 41 |
| 4.14.2.5 | LDRH(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 41 |
| 4.14.2.6 | mostrar_memoria(uint32_t *memoria, int tama) | 42 |
| 4.14.2.7 | POP(uint32_t *registros, uint32_t *memory, uint32_t *res) | 43 |
| 4.14.2.8 | POPINTERRUPT(uint32_t *registros, uint32_t *memory, uint32_t *res) | 43 |
| 4.14.2.9 | PUSH(uint32_t *registros, uint32_t *memory, uint32_t *res) | 43 |
| 4.14.2.10 | PUSHINTERRUPT(uint32_t *registros, uint32_t *memory, uint32_t *res) | 43 |
| 4.14.2.11 | STR(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 43 |
| 4.14.2.12 | STRB(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 44 |
| 4.14.2.13 | STRH(uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory) | 44 |
| 4.15 | Referencia del Archivo salto.c | 44 |
| 4.15.1 | Documentación de las funciones | 45 |
| 4.15.1.1 | B(uint32_t *pc, uint32_t valor) | 45 |
| 4.15.1.2 | BAL(uint32_t *pc, uint32_t valor) | 45 |
| 4.15.1.3 | BCC(uint32_t *pc, uint32_t valor) | 45 |
| 4.15.1.4 | BCS(uint32_t *pc, uint32_t valor) | 46 |
| 4.15.1.5 | BEQ(uint32_t *pc, uint32_t valor) | 46 |
| 4.15.1.6 | BGE(uint32_t *pc, uint32_t valor) | 46 |
| 4.15.1.7 | BGT(uint32_t *pc, uint32_t valor) | 46 |
| 4.15.1.8 | BHI(uint32_t *pc, uint32_t valor) | 46 |
| 4.15.1.9 | BL(uint32_t *pc, uint32_t valor, uint32_t *LR) | 46 |
| 4.15.1.10 | BLE(uint32_t *pc, uint32_t valor) | 47 |
| 4.15.1.11 | BLS(uint32_t *pc, uint32_t valor) | 47 |
| 4.15.1.12 | BLT(uint32_t *pc, uint32_t valor) | 47 |
| 4.15.1.13 | BMI(uint32_t *pc, uint32_t valor) | 47 |
| 4.15.1.14 | BNE(uint32_t *pc, uint32_t valor) | 47 |
| 4.15.1.15 | BPL(uint32_t *pc, uint32_t valor) | 47 |
| 4.15.1.16 | BVC(uint32_t *pc, uint32_t valor) | 48 |

| | | |
|-----------|--|----|
| 4.15.1.17 | BVS(uint32_t *pc, uint32_t valor) | 48 |
| 4.15.1.18 | BX(uint32_t *pc, uint32_t *LR) | 48 |
| 4.15.2 | Documentación de las variables | 48 |
| 4.15.2.1 | banderas | 48 |
| 4.15.2.2 | LR | 48 |
| 4.16 | Referencia del Archivo salto.h | 48 |
| 4.16.1 | Documentación de las funciones | 49 |
| 4.16.1.1 | B(uint32_t *pc, uint32_t valor) | 49 |
| 4.16.1.2 | BAL(uint32_t *pc, uint32_t valor) | 49 |
| 4.16.1.3 | BCC(uint32_t *pc, uint32_t valor) | 49 |
| 4.16.1.4 | BCS(uint32_t *pc, uint32_t valor) | 50 |
| 4.16.1.5 | BEQ(uint32_t *pc, uint32_t valor) | 50 |
| 4.16.1.6 | BGE(uint32_t *pc, uint32_t valor) | 50 |
| 4.16.1.7 | BGT(uint32_t *pc, uint32_t valor) | 50 |
| 4.16.1.8 | BHI(uint32_t *pc, uint32_t valor) | 50 |
| 4.16.1.9 | BL(uint32_t *pc, uint32_t valor, uint32_t *LR) | 50 |
| 4.16.1.10 | BLE(uint32_t *pc, uint32_t valor) | 51 |
| 4.16.1.11 | BLS(uint32_t *pc, uint32_t valor) | 51 |
| 4.16.1.12 | BLT(uint32_t *pc, uint32_t valor) | 51 |
| 4.16.1.13 | BMI(uint32_t *pc, uint32_t valor) | 51 |
| 4.16.1.14 | BNE(uint32_t *pc, uint32_t valor) | 51 |
| 4.16.1.15 | BPL(uint32_t *pc, uint32_t valor) | 51 |
| 4.16.1.16 | BVC(uint32_t *pc, uint32_t valor) | 52 |
| 4.16.1.17 | BVS(uint32_t *pc, uint32_t valor) | 52 |
| 4.16.1.18 | BX(uint32_t *pc, uint32_t *LR) | 52 |
| 4.17 | Referencia del Archivo screen.c | 52 |
| 4.17.1 | Documentación de las funciones | 52 |
| 4.17.1.1 | showRegisters(uint32_t *registers, size_t len) | 52 |
| 4.18 | Referencia del Archivo screen.h | 52 |
| 4.18.1 | Documentación de las funciones | 53 |
| 4.18.1.1 | showRegisters(uint32_t *registers, size_t len) | 53 |

Capítulo 1

Índice de estructura de datos

1.1. Estructura de datos

Lista de estructuras con una breve descripción:

| | |
|---|---|
| ins_t | 5 |
| instruction_t | 5 |
| port_t | 6 |

Capítulo 2

Indice de archivos

2.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

| | |
|-----------|----|
| alu.c | 7 |
| alu.h | 16 |
| colors.h | 24 |
| decoder.c | 25 |
| decoder.h | 27 |
| flags.c | 30 |
| flags.h | 30 |
| io.c | 31 |
| io.h | 32 |
| main.c | 33 |
| NVIC.c | 35 |
| NVIC.h | 36 |
| ram.c | 36 |
| ram.h | 40 |
| salto.c | 44 |
| salto.h | 48 |
| screen.c | 52 |
| screen.h | 52 |

Capítulo 3

Documentación de las estructuras de datos

3.1. Referencia de la Estructura ins_t

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

Campos de datos

- char ** [array](#)

3.1.1. Documentación de los campos

3.1.1.1. char** array

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

- [decoder.h](#)

3.2. Referencia de la Estructura instruction_t

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

Campos de datos

- char [mnemonic](#) [10]
- char [op1_type](#)
- char [op2_type](#)
- char [op3_type](#)
- uint32_t [op1_value](#)
- uint32_t [op2_value](#)
- uint32_t [op3_value](#)
- uint8_t [registers_list](#) [16]

3.2.1. Documentación de los campos

3.2.1.1. char mnemonic[10]

3.2.1.2. char op1_type

3.2.1.3. uint32_t op1_value

3.2.1.4. char op2_type

3.2.1.5. uint32_t op2_value

3.2.1.6. char op3_type

3.2.1.7. uint32_t op3_value

3.2.1.8. uint8_t registers_list[16]

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

- [decoder.h](#)

3.3. Referencia de la Estructura port_t

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

Campos de datos

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

3.3.1. Documentación de los campos

3.3.1.1. uint8_t DDR

3.3.1.2. uint8_t Interrupts

3.3.1.3. uint8_t PIN

3.3.1.4. uint8_t Pins

3.3.1.5. uint8_t PORT

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

- [io.h](#)

Capítulo 4

Documentación de archivos

4.1. Referencia del Archivo alu.c

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

Funciones

- void **ADD** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que suma los valores de dos direcciones.
- void **SUB** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que resta los valores de dos direcciones.
- void **AND** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que multiplica los valores de dos direcciones.
- void **ORR** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion de suma logica entre los valores de dos direcciones.
- void **EOR** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion logica exclusiva entre dos direcciones.
- void **ADCS** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que realiza suma con carry.
- void **CMN** (uint32_t *Rm, uint32_t *Rn)
Funcion que suma pero no guarda el resultado solo modifica banderas.
- void **SBC** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion de resta con carry.
- void **MOVS** (uint32_t *Rm, uint32_t Rn)
Funcion que escribe un valor inmediato en un registro.
- void **SUBS** (uint32_t *Rd, uint32_t *Rm, uint32_t Rn)
Funcion que resta un valor inmediato con el valor de direccion Rm.
- void **ADDS** (uint32_t *Rd, uint32_t *Rm, uint32_t Rn)
Funcion que suma un valor inmediato con el valor de direccion Rm.
- void **LSL** (uint32_t *Rm, uint32_t *Rn)
Funcion de desplazamiento logico a la izquierda.
- void **LSR** (uint32_t *Rm, uint32_t *Rn)
Funcion de desplazamiento logico a la izquierda.

- void **RORS** (uint32_t *Rm, uint32_t Rn)
Funcion de rotacion a la derecha.
- void **ASRS** (uint32_t *Rm, uint32_t Rn)
Funcion de desplazamiento aritmetico logico a la derecha.
- void **MOV** (uint32_t *Rm, uint32_t *Rn)
Funcion que escribe un valor en un resgistro.
- void **CMP** (uint32_t *Rm, uint32_t *Rn)
Funcion que resta pero no guarda el resultado solo modifica banderas.
- void **REV** (uint32_t *Rm, uint32_t *Rn)
Funcion que cambia el orden de los Bytes.
- void **REV16** (uint32_t *Rm, uint32_t *Rn)
Funcion que cambia de orden los bytes en cada halfword de 16 bits.
- void **BICS** (uint32_t *Rm, uint32_t Rn)
Funcion que ingresa un registro y el complementeto de otro.
- void **MVN** (uint32_t *Rm, uint32_t *Rn)
Funcion que guarda el complementeto de un registro.
- void **RSBS** (uint32_t *Rm, uint32_t *Rn)
Funcion que obtiene el complemento a dos de un numero.
- void **NOP** (void)
Funcion que no ejecuta ninguna operacion o mas bien no hace nada.
- void **TST** (uint32_t *Rm, uint32_t *Rn)
Funcion que multiplica bit a bit pero no guarad el resultado solo modifica banderas.
- void **LSLS** (uint32_t *Rd, uint32_t *Rm, uint32_t Rn)
Funcion que realiza un desplazamiento logico a la izquierda dependiendo del valor inmediato.
- void **LSRS** (uint32_t *Rd, uint32_t *Rm, uint32_t Rn)
Funcion que realiza un desplazamiento logico a la derecha dependiendo del valor inmediato.
- void **obtenerBandera** (bool *bands)
Funcion que obtiene el valor de las banderas.
- void **SalvarBanderas** (bool *bands)
Funcion que almacena el valor de las banderas.
- void **MULS** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion de multiplicacion y solo guarda los 32 bits menos significativos.

Variables

- bool **banderas** [4]
- uint8_t **comp** =0

4.1.1. Documentación de las funciones

4.1.1.1. void ADCS (uint32_t * Rd, uint32_t * Rm, uint32_t * Rn)

Funcion que realiza suma con carry.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.2. void ADD (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que suma los valores de dos direcciones.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.3. void ADDS (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t *Rn*)

Funcion que suma un valor inmediato con el valor de direccion Rm.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.4. void AND (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que multiplica los valores de dos direcciones.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.5. void ASRS (uint32_t * *Rm*, uint32_t *Rn*)

Funcion de desplazamiento aritmetico logico a la derecha.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

4.1.1.6. void BICS (uint32_t * *Rm*, uint32_t *Rn*)

Funcion que ingresa un registro y el complemento de otro.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
|-----------|--|

| | |
|-----------|-------------------------------|
| <i>Rn</i> | Operador sin signo de 32 bits |
|-----------|-------------------------------|

4.1.1.7. void CMN (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que suma pero no guarda el resultado solo modifica banderas.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.1.1.8. void CMP (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que resta pero no guarda el resultado solo modifica banderas.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.1.1.9. void EOR (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion logica exclusiva entre dos direcciones.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.10. void LSL (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de desplazamiento logico a la izquierda.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.1.1.11. void LSLS (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que realiza un desplazamiento logico a la izquierda dependiendo del valor inmediato.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | <i>Rn</i> Operador sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.12. void LSR (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de desplazamiento logico a la izquierda.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.1.1.13. void LSRS (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t *Rn*)

Funcion que realiza un desplazamiento logico a la derecha dependiendo del valor inmediato.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Rn Operador sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.14. void MOV (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que escribe un valor en un resgistro.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.1.1.15. void MOVS (uint32_t * *Rm*, uint32_t *Rn*)

Funcion que escribe un valor inmediato en un registro.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

4.1.1.16. void MULS (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de multiplicacion y solo guarda los 32 bits menos significativos.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.17. void MVN (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que guarda el complementeto de un registro.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.1.1.18. void NOP (void)

Funcion que no ejecuta ninguna operacion o mas bien no hace nada.

4.1.1.19. void obtenerBandera (bool * *bands*)

Funcion que obtiene el valor de las banderas.

Parámetros

| | |
|--------------|--|
| <i>bands</i> | direccion que obtiene el valor de las banderas |
|--------------|--|

4.1.1.20. void ORR (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de suma logica entre los valores de dos direcciones.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.21. void REV (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que cambia el orden de los Bytes.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.1.1.22. void REV16 (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que cambia de orden los bytes en cada halfword de 16 bits.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.1.1.23. void RORS (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de rotacion a la derecha.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

4.1.1.24. void RSBS (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que obtiene el complemento a dos de un numero.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.1.1.25. void SalvarBanderas (bool * *bands*)

Funcion que almacena el valor de las banderas.

Parámetros

| | |
|--------------|--|
| <i>bands</i> | direccion que obtiene el valor de las banderas |
|--------------|--|

4.1.1.26. void SBC (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de resta con carry.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

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

Funcion que resta los valores de dos direcciones.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.28. void SUBS (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que resta un valor inmediato con el valor de direccion Rm.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.1.1.29. void TST (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que multiplica bit a bit pero no guarad el resultado solo modifica banderas.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.1.2. Documentación de las variables

4.1.2.1. bool banderas[4]

4.1.2.2. uint8_t comp =0

4.2. Referencia del Archivo alu.h

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

Funciones

- void **ADD** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que suma los valores de dos direcciones.
- void **SUB** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que resta los valores de dos direcciones.
- void **AND** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que multiplica los valores de dos direcciones.
- void **ORR** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion de suma logica entre los valores de dos direcciones.
- void **EOR** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion logica exclusiva entre dos direcciones.
- void **LSL** (uint32_t *Rm, uint32_t *Rn)
Funcion de desplazamiento logico a la izquierda.
- void **LSR** (uint32_t *Rm, uint32_t *Rn)
Funcion de desplazamiento logico a la izquierda.
- void **RORS** (uint32_t *Rm, uint32_t Rn)
Funcion de rotacion a la derecha.
- void **ASRS** (uint32_t *Rm, uint32_t Rn)
Funcion de desplazamiento aritmetico logico a la derecha.
- void **REV** (uint32_t *Rm, uint32_t *Rn)
Funcion que cambia el orden de los Bytes.

- void **REV16** (uint32_t *Rm, uint32_t *Rn)
Funcion que cambia de orden los bytes en cada halfword de 16 bits.
- void **BICS** (uint32_t *Rm, uint32_t *Rn)
Funcion que ingresa un registro y el complemento de otro.
- void **MVN** (uint32_t *Rm, uint32_t *Rn)
Funcion que guarda el complemento de un registro.
- void **RSBS** (uint32_t *Rm, uint32_t *Rn)
Funcion que obtiene el complemento a dos de un numero.
- void **NOP** (void)
Funcion que no ejecuta ninguna operacion o mas bien no hace nada.
- void **MOV** (uint32_t *Rm, uint32_t *Rn)
Funcion que escribe un valor en un registro.
- void **CMP** (uint32_t *Rm, uint32_t *Rn)
Funcion que resta pero no guarda el resultado solo modifica banderas.
- void **ADCS** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que realiza suma con carry.
- void **CMN** (uint32_t *Rm, uint32_t *Rn)
Funcion que suma pero no guarda el resultado solo modifica banderas.
- void **SBC** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion de resta con carry.
- void **MOVS** (uint32_t *Rm, uint32_t *Rn)
Funcion que escribe un valor inmediato en un registro.
- void **SUBS** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que resta un valor inmediato con el valor de direccion Rm.
- void **ADDS** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que suma un valor inmediato con el valor de direccion Rm.
- void **TST** (uint32_t *Rm, uint32_t *Rn)
Funcion que multiplica bit a bit pero no guarda el resultado solo modifica banderas.
- void **LSLS** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que realiza un desplazamiento logico a la izquierda dependiendo del valor inmediato.
- void **LSRS** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion que realiza un desplazamiento logico a la derecha dependiendo del valor inmediato.
- void **obtenerBandera** (bool *bands)
Funcion que obtiene el valor de las banderas.
- void **SalvarBanderas** (bool *bands)
Funcion que almacena el valor de las banderas.
- void **MULS** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn)
Funcion de multiplicacion y solo guarda los 32 bits menos significativos.

4.2.1. Documentación de las funciones

4.2.1.1. void ADCS (uint32_t * Rd, uint32_t * Rm, uint32_t * Rn)

Funcion que realiza suma con carry.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
|-----------|--|

| | |
|-----------|--|
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |
|-----------|--|

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.2. void ADD (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que suma los valores de dos direcciones.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.3. void ADDS (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t *Rn*)

Funcion que suma un valor inmediato con el valor de direccion Rm.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.4. void AND (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que multiplica los valores de dos direcciones.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.5. void ASRS (uint32_t * *Rm*, uint32_t *Rn*)

Funcion de desplazamiento aritmetico logico a la derecha.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

4.2.1.6. void BICS (uint32_t * *Rm*, uint32_t *Rn*)

Funcion que ingresa un registro y el complemento de otro.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

4.2.1.7. void CMN (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que suma pero no guarda el resultado solo modifica banderas.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.2.1.8. void CMP (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que resta pero no guarda el resultado solo modifica banderas.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.2.1.9. void EOR (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion logica exclusiva entre dos direcciones.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.10. void LSL (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de desplazamiento logico a la izquierda.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.2.1.11. void LSLS (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t *Rn*)

Funcion que realiza un desplazamiento logico a la izquierda dependiendo del valor inmediato.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Rn Operador sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.12. void LSR (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de desplazamiento logico a la izquierda.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.2.1.13. void LSRS (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t *Rn*)

Funcion que realiza un desplazamiento logico a la derecha dependiendo del valor inmediato.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Rn Operador sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.14. void MOV (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que escribe un valor en un resgistro.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.2.1.15. void MOVS (uint32_t * *Rm*, uint32_t *Rn*)

Funcion que escribe un valor inmediato en un registro.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

4.2.1.16. void MULS (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de multiplicacion y solo guarda los 32 bits menos significativos.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.17. void MVN (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que guarda el complementeto de un registro.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.2.1.18. void NOP (void)

Funcion que no ejecuta ninguna operacion o mas bien no hace nada.

4.2.1.19. void obtenerBandera (bool * *bands*)

Funcion que obtiene el valor de las banderas.

Parámetros

| | |
|--------------|--|
| <i>bands</i> | direccion que obtiene el valor de las banderas |
|--------------|--|

4.2.1.20. void ORR (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de suma logica entre los valores de dos direcciones.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.21. void REV (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que cambia el orden de los Bytes.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.2.1.22. void REV16 (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que cambia de orden los bytes en cada halfword de 16 bits.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.2.1.23. void RORS (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de rotacion a la derecha.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

4.2.1.24. void RSBS (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que obtiene el complemento a dos de un numero.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.2.1.25. void SalvarBanderas (bool * *bands*)

Funcion que almacena el valor de las banderas.

Parámetros

| | |
|--------------|--|
| <i>bands</i> | direccion que obtiene el valor de las banderas |
|--------------|--|

4.2.1.26. void SBC (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*)

Funcion de resta con carry.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

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

Funcion que resta los valores de dos direcciones.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
|-----------|--|

| | |
|-----------|--|
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |
|-----------|--|

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.28. void SUBS (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t *Rn*)

Funcion que resta un valor inmediato con el valor de direccion Rm.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Operador sin signo de 32 bits |

Devuelve

Rd Direccion que contiene el valor a retornar

4.2.1.29. void TST (uint32_t * *Rm*, uint32_t * *Rn*)

Funcion que multiplica bit a bit pero no guarad el resultado solo modifica banderas.

Parámetros

| | |
|-----------|--|
| <i>Rm</i> | Direccion que contiene el valor sin signo de 32 bits |
| <i>Rn</i> | Direccion que contiene el valor sin signo de 32 bits |

4.3. Referencia del Archivo colors.h

'defines'

- #define BLACK 0 /* Black */
- #define BLUE 1 /* Blue */
- #define GREEN 2 /* Green */
- #define AQUA 3 /* Aqua */
- #define RED 4 /* Red */
- #define PURPLE 5 /* Purple */
- #define YELLOW 6 /* Yellow */
- #define WHITE 7 /* White */
- #define GRAY 8 /* Gray */
- #define LIGHT_BLUE 9 /* Light Blue */
- #define LIGHT_GREEN A /* Light Green */
- #define LIGHT_AQUA B /* Light Aqua */
- #define LIGHT_RED C /* Light Red */
- #define LIGHT_PURPLE D /* Light Purple */
- #define LIGHT_YELLOW E /* Light Yellow */
- #define BRIGHT_WHITE F /* Bright White */

4.3.1. Documentación de los 'defines'

4.3.1.1. `#define AQUA 3 /* Aqua */`

4.3.1.2. `#define BLACK 0 /* Black */`

4.3.1.3. `#define BLUE 1 /* Blue */`

4.3.1.4. `#define BRIGHT_WHITE F /* Bright White */`

4.3.1.5. `#define GRAY 8 /* Gray */`

4.3.1.6. `#define GREEN 2 /* Green */`

4.3.1.7. `#define LIGHT_AQUA B /* Light Aqua */`

4.3.1.8. `#define LIGHT_BLUE 9 /* Light Blue */`

4.3.1.9. `#define LIGHT_GREEN A /* Light Green */`

4.3.1.10. `#define LIGHT_PURPLE D /* Light Purple */`

4.3.1.11. `#define LIGHT_RED C /* Light Red */`

4.3.1.12. `#define LIGHT_YELLOW E /* Light Yellow */`

4.3.1.13. `#define PURPLE 5 /* Purple */`

4.3.1.14. `#define RED 4 /* Red */`

4.3.1.15. `#define WHITE 7 /* White */`

4.3.1.16. `#define YELLOW 6 /* Yellow */`

4.4. Referencia del Archivo decoder.c

```
#include "decoder.h"
#include "alu.h"
#include "salto.h"
#include <stdint.h>
#include "screen.h"
#include "ram.h"
#include "NVIC.h"
```

Funciones

- void `iniciaram` (void)
Funcion que inicia la ram con FFFF.
- void `decodeInstruction` (`instruction_t` instruction)
- void `obtenerPC` (`uint32_t` *pcount)
- void `obtener_registros` (`uint32_t` *pcount)
Funcion que obtiene de las direcciones el valor de los registro.
- void `obtener_memoria` (`uint32_t` *pcount)
Funcion que obtiene los registros de memoria.

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

Variables

- uint32_t `registers` [15]
- uint32_t `memoria` [MEMORIA]
- uint8_t `guardar` [16] = {1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,1}
- uint8_t `i`
- uint8_t `indicador` =0
- uint8_t `exnum` [16] = {0}
- uint16_t `bin` =0

4.4.1. Documentación de las funciones

4.4.1.1. int `countLines` (FILE * *fp*)

4.4.1.2. void `decodeInstruction` (`instruction_t` *instruction*)

4.4.1.3. `instruction_t` `getInstruction` (char * *instStr*)

Obtiene la instrucción separada por partes.

Parámetros

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

Devuelve

`instruction_t` la instrucción separada por partes

4.4.1.4. void `iniciaram` (void)

Funcion que inicia la ram con FFFF..

4.4.1.5. void `obtener_memoria` (uint32_t * *pcount*)

Funcion que obtiene los registros de memoria.

Parámetros

| | |
|-----------|-------------------------|
| <i>fp</i> | Direccion de la memoria |
|-----------|-------------------------|

4.4.1.6. void `obtener_registros` (uint32_t * *pcount*)

Funcion que obtiene de las direcciones el valor de los registro.

Parámetros

| | |
|---------------|-------------------------|
| <i>pcount</i> | Dirección de la memoria |
|---------------|-------------------------|

4.4.1.7. void obtenerPC (uint32_t * pcount)

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

4.4.2. Documentación de las variables

4.4.2.1. uint16_t bin =0

4.4.2.2. uint8_t exnum[16]={0}

4.4.2.3. uint8_t guardar[16] ={1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,1}

4.4.2.4. uint8_t i

4.4.2.5. uint8_t indicador =0

4.4.2.6. uint32_t memoria[MEMORIA]

4.4.2.7. uint32_t registers[15]

4.5. Referencia del Archivo decoder.h

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

Estructuras de datos

- struct [ins_t](#)
- struct [instruction_t](#)

Funciones

- [instruction_t](#) getInstruction (char *instStr)
Obtiene la instrucción separada por partes.
- int [readFile](#) (char *filename, [ins_t](#) *instructions)
- int [countLines](#) (FILE *fp)
- void [obtenerPC](#) (uint32_t *pcount)
- void [decodeInstruction](#) ([instruction_t](#) instruction)
- void [iniciaram](#) (void)
Funcion que inicia la ram con FFFF.
- void [obtener_registros](#) (uint32_t *pcount)
Funcion que obtiene de las direcciones el valor de los registro.
- void [obtener_memoria](#) (uint32_t *pcount)
Funcion que obtiene los registros de memoria.
- void [PUSHINTERRUPT](#) ([registers](#), [memoria](#), [guardar](#))

Funcion PUSH dentro de la interrupcion.

- void **POPHINTERRUPT** (**registers**, **memoria**, **guardar**)

Funcion POP dentro de la interrupcion.

4.5.1. Documentación de las funciones

4.5.1.1. int countLines (FILE * *fp*)

4.5.1.2. void decodeInstruction (**instruction_t** *instruction*)

4.5.1.3. **instruction_t** getInstruction (char * *instStr*)

Obtiene la instrucción separada por partes.

Parámetros

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

Devuelve

instruction_t la instrucción separada por partes

4.5.1.4. void iniciaram (void)

Funcion que inicia la ram con FFFF..

4.5.1.5. void obtener_memoria (uint32_t * *pcount*)

Funcion que obtiene los registros de memoria.

Parámetros

| | |
|-----------|-------------------------|
| <i>fp</i> | Direccion de la memoria |
|-----------|-------------------------|

4.5.1.6. void obtener_registros (uint32_t * *pcount*)

Funcion que obtiene de las direcciones el valor de los registro.

Parámetros

| | |
|---------------|-------------------------|
| <i>pcount</i> | Direccion de la memoria |
|---------------|-------------------------|

4.5.1.7. void obtenerPC (uint32_t * *pcount*)

4.5.1.8. void **POPHINTERRUPT** (**registers** , **memoria** , **guardar**)

Funcion POP dentro de la interrupcion.

Parámetros

| | |
|------------------|--|
| <i>registers</i> | |
|------------------|--|

| | |
|----------------|--|
| <i>memoria</i> | |
| <i>guardar</i> | |

4.5.1.9. void PUSHINTERRUPT (registers , memoria , guardar)

Funcion PUSH dentro de la interrupcion.

Parámetros

| | |
|------------------|--|
| <i>registers</i> | |
| <i>memoria</i> | |
| <i>guardar</i> | |

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

4.6. Referencia del Archivo flags.c

```
#include <stdint.h>
#include <stdio.h>
#include "flags.h"
#include "alu.h"
#include <curses.h>
```

Funciones

- void **flag** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn, bool *banderas, uint8_t *compar)
Función que determina las banderas (C,Z y N).

Variables

- uint32_t **H** =2147483648UL

4.6.1. Documentación de las funciones

4.6.1.1. void flag (uint32_t * Rd, uint32_t * Rm, uint32_t * Rn, bool * banderas, uint8_t * compar)

Función que determina las banderas (C,Z y N).

Parámetros

| | |
|-----------|--|
| <i>Rd</i> | Dirección que contiene el operador sin signo de 32 bits. |
| <i>Rm</i> | Dirección que contiene el operador sin signo de 32 bits. |
| <i>Rn</i> | Dirección que contiene el operador sin signo de 32 bits. |

Devuelve

banderas Un booleano con el resultado (0 y 1) dependiendo el caso.

4.6.2. Documentación de las variables

4.6.2.1. uint32_t H =2147483648UL

4.7. Referencia del Archivo flags.h

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

'defines'

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

Funciones

- void **flag** (uint32_t *Rd, uint32_t *Rm, uint32_t *Rn, bool *banderas, uint8_t *compar)
Función que determina las banderas (C,Z y N).

4.7.1. Documentación de los 'defines'

4.7.1.1. #define C 0

4.7.1.2. #define N 2

4.7.1.3. #define V 3

4.7.1.4. #define Z 1

4.7.2. Documentación de las funciones

4.7.2.1. void **flag** (uint32_t * *Rd*, uint32_t * *Rm*, uint32_t * *Rn*, bool * *banderas*, uint8_t * *compar*)

Función que determina las banderas (C,Z y N).

Parámetros

| | |
|-----------|--|
| <i>Rd</i> | Dirección que contiene el operador sin signo de 32 bits. |
| <i>Rm</i> | Dirección que contiene el operador sin signo de 32 bits. |
| <i>Rn</i> | Dirección que contiene el operador sin signo de 32 bits. |

Devuelve

banderas Un booleano con el resultado (0 y 1) dependiendo el caso.

4.8. Referencia del Archivo io.c

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

Funciones

- void **initIO** (void)
- void **changePinPortA** (uint8_t pin, uint8_t value)

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

Variables

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

4.8.1. Documentación de las funciones

4.8.1.1. void `changePinPortA` (uint8_t *pin*, uint8_t *value*)

4.8.1.2. void `changePinPortB` (uint8_t *pin*, uint8_t *value*)

4.8.1.3. void `initIO` (void)

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

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

4.8.1.6. void `showPorts` (void)

4.8.2. Documentación de las variables

4.8.2.1. uint8_t `irq`[16]

4.8.2.2. `port_t` PORTA

4.8.2.3. `port_t` PORTB

4.9. Referencia del Archivo io.h

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

Estructuras de datos

- struct `port_t`

'defines'

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

Funciones

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

4.9.1. Documentación de los 'defines'

4.9.1.1. `#define BLUEBLACK 10 /*Text Blue Background Black*/`

4.9.1.2. `#define HIGH 1`

4.9.1.3. `#define LOW 0`

4.9.1.4. `#define Read 1`

4.9.1.5. `#define REDBLACK 20 /*Text Red Background Black*/`

4.9.1.6. `#define WHITEBLACK 30 /*Text White Background White*/`

4.9.1.7. `#define Write 0`

4.9.1.8. `#define XINIT 10`

4.9.1.9. `#define YINIT 5`

4.9.2. Documentación de las funciones

4.9.2.1. void `changePinPortA` (uint8_t *pin*, uint8_t *value*)

4.9.2.2. void `changePinPortB` (uint8_t *pin*, uint8_t *value*)

4.9.2.3. void `initIO` (void)

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

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

4.9.2.6. void `showPorts` (void)

4.10. Referencia del Archivo main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <curses.h>
#include "screen.h"
#include "alu.h"
#include "flags.h"
#include "decoder.h"
#include "ram.h"
#include "io.h"
```

'defines'

- #define NORMAL 0
- #define BRILLO 1
- #define SEMIBRILLO 2
- #define INTERMITENTE 5
- #define INVERSO 7
- #define NEGRO 30
- #define ROJO 31
- #define VERDE 32
- #define MARRON 33
- #define AZUL 34
- #define ROSA 35
- #define CELESTE 36
- #define BLANCO 37
- #define MAXLINEAS 15
- #define MAXCAD 70

Funciones

- int Disp (void)
- void Inivideo (void)
- void Exit (void)
- void IniVideo (void)
- int main (void)

Variables

- WINDOW * win
- WINDOW * wine
- uint32_t memoria [MEMORIA]
- uint8_t data

4.10.1. Documentación de los 'defines'

4.10.1.1. #define AZUL 34

4.10.1.2. #define BLANCO 37

4.10.1.3. #define BRILLO 1

4.10.1.4. #define CELESTE 36

4.10.1.5. #define INTERMITENTE 5

4.10.1.6. #define INVERSO 7

4.10.1.7. #define MARRON 33

4.10.1.8. #define MAXCAD 70

4.10.1.9. `#define MAXLINEAS 15`

4.10.1.10. `#define NEGRO 30`

4.10.1.11. `#define NORMAL 0`

4.10.1.12. `#define ROJO 31`

4.10.1.13. `#define ROSA 35`

4.10.1.14. `#define SEMIBRILLO 2`

4.10.1.15. `#define VERDE 32`

4.10.2. Documentación de las funciones

4.10.2.1. `int Disp (void)`

4.10.2.2. `void Exit (void)`

4.10.2.3. `void Inivideo (void)`

4.10.2.4. `void IniVideo (void)`

4.10.2.5. `int main (void)`

4.10.3. Documentación de las variables

4.10.3.1. `uint8_t data`

4.10.3.2. `uint32_t memoria[MEMORIA]`

4.10.3.3. `WINDOW* win`

4.10.3.4. `WINDOW * wine`

4.11. Referencia del Archivo NVIC.c

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

Funciones

- `void NVIC_EnableIRQ (uint8_t *Exnumb, uint8_t numb)`
Funcion que activa las interrupciones.
- `void NVIC_DisableIRQ (uint8_t *Exnumb, uint8_t numb)`
Funcion que ejecuta la regresion al codigo principal.

4.11.1. Documentación de las funciones

4.11.1.1. `void NVIC_DisableIRQ (uint8_t * Exnumb, uint8_t numb)`

Funcion que ejecuta la regresion al codigo principal.

Parámetros

| | |
|--------------|--|
| <i>Exnum</i> | Direccion a la cual regresar al codigo principal |
| <i>numb</i> | numero sin signo de 8 bits |

4.11.1.2. void NVIC_EnableIRQ (uint8_t * *Exnumb*, uint8_t *numb*)

Funcion que activa las interrupciones.

Parámetros

| | |
|--------------|---|
| <i>Exnum</i> | Direccion a la cual accede para iniciar la interrupcion |
| <i>numb</i> | numero entero sin signo de 8 bits |

4.12. Referencia del Archivo NVIC.h

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

Funciones

- void [NVIC_EnableIRQ](#) (uint8_t **Exnumb*, uint8_t *numb*)
Funcion que activa las interrupciones.
- void [NVIC_DisableIRQ](#) (uint8_t **Exnumb*, uint8_t *numb*)
Funcion que ejecuta la regresion al codigo principal.

4.12.1. Documentación de las funciones

4.12.1.1. void NVIC_DisableIRQ (uint8_t * *Exnumb*, uint8_t *numb*)

Funcion que ejecuta la regresion al codigo principal.

Parámetros

| | |
|--------------|--|
| <i>Exnum</i> | Direccion a la cual regresar al codigo principal |
| <i>numb</i> | numero sin signo de 8 bits |

4.12.1.2. void NVIC_EnableIRQ (uint8_t * *Exnumb*, uint8_t *numb*)

Funcion que activa las interrupciones.

Parámetros

| | |
|--------------|---|
| <i>Exnum</i> | Direccion a la cual accede para iniciar la interrupcion |
| <i>numb</i> | numero entero sin signo de 8 bits |

4.13. Referencia del Archivo ram.c

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

Funciones

- `uint32_t bitcount (uint32_t *R)`
- `void PUSH (uint32_t *registros, uint32_t *memory, uint32_t *res)`
Funcion que permite realizar el PUSH en la memoria.
- `void POP (uint32_t *registros, uint32_t *memory, uint32_t *res)`
Funcion que permite realizar el POP en la memoria.
- `void mostrar_memoria (uint32_t *memoria, int tama)`
Funcion que permite mostrar los valores almacenados en la memoria.
- `void inimeroria (uint32_t *memoria, int tama)`
Funcion que inicia la memoria, es decir comienza a desde el valor maximo y va disminuye (descendente)
- `void PUSHINTERRUPT (uint32_t *registros, uint32_t *memory, uint32_t *res)`
Funcion POP dentro de la interrupcion.
- `void POPINTERRUPT (uint32_t *registros, uint32_t *memory, uint32_t *res)`
Funcion POP dentro de la interrupcion.
- `void LDR (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)`
Funcion que extrae 4 valores de la pila dependiendo de la suma de las direcciones.
- `void LDRB (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)`
Funcion que extrae un valor de la pila dependiendo de la suma de la direccion con otra direccion de registro.
- `void LDRH (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)`
Funcion que extrae dos valores de la pila dependiendo de la suma de la direccion con un valor inmediato adicionalmente se realiza extension de ceros.
- `void STR (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)`
Funcion que almacena 4 valores en la pila.
- `void STRB (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)`
Funcion que almacena un valore en la pila ya sea los primeros 8 bit o los 8 bit ultimos.
- `void STRH (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)`
Funcion que almacena dos valores en la pila ya sea los 16 primeros 16bit o los ultimos 16 bits.

4.13.1. Documentación de las funciones

4.13.1.1. `uint32_t bitcount (uint32_t * R)`

4.13.1.2. `void inimeroria (uint32_t * memoria, int tama)`

Funcion que inicia la memoria, es decir comienza a desde el valor maximo y va disminuye (descendente)

Parámetros

| | |
|----------------|---|
| <i>memoria</i> | Direccion en la cual se almacenan los datos en la memoria |
| <i>tama</i> | Es el tamaño de los datos a almacenar |

4.13.1.3. `void LDR (uint32_t * Rt, uint32_t Rn, uint32_t Rm, uint32_t * memory)`

Funcion que extrae 4 valores de la pila dependiendo de la suma de las direcciones.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la direccion de donde se extrae los valores |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicará en memoria para obtener el valor |

4.13.1.4. void LDRB (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint32_t * *memory*)

Función que extrae un valor de la pila dependiendo de la suma de la dirección con otra dirección de registro.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la direccion de donde se extrae el valor |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicará en memoria para obtener el valor |

4.13.1.5. void LDRH (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint32_t * *memory*)

Función que extrae dos valores de la pila dependiendo de la suma de la dirección con un valor inmediato adicionalmente se realiza extensión de ceros.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la direccion de donde se extrae los valores |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicará en memoria para obtener el valor |

4.13.1.6. void mostrar_memoria (uint32_t * *memoria*, int *tama*)

Función que permite mostrar los valores almacenados en la memoria.

Parámetros

| | |
|----------------|---|
| <i>memoria</i> | Dirección en la cual se almacenan los datos en la memoria |
| <i>tama</i> | Es el tamaño de los datos a almacenar |

4.13.1.7. void POP (uint32_t * *registros*, uint32_t * *memory*, uint32_t * *res*)

Función que permite realizar el POP en la memoria.

Parámetros

| | |
|------------------|---|
| <i>registros</i> | Dirección de los registros que van del 0 al 15 |
| <i>memoria</i> | Dirección en la cual se almacenan los datos en la memoria |
| <i>res</i> | Dirección de registro |

4.13.1.8. void POPINTERRUPT (uint32_t * *registros*, uint32_t * *memory*, uint32_t * *res*)

Función POP dentro de la interrupción.

Parámetros

| | |
|------------------|-----------------------|
| <i>registros</i> | Registros del 0 al 15 |
| <i>memoria</i> | Dirección de memoria |
| <i>res</i> | Registros en memoria |

4.13.1.9. void PUSH (uint32_t * *registros*, uint32_t * *memory*, uint32_t * *res*)

Función que permite realizar el PUSH en la memoria.

Parámetros

| | |
|------------------|---|
| <i>registros</i> | Dirección de los registros que van del 0 al 15 |
| <i>memoria</i> | Dirección en la cual se almacenan los datos en la memoria |
| <i>res</i> | Dirección de registro |

4.13.1.10. void PUSHINTERRUPT (uint32_t * *registros*, uint32_t * *memory*, uint32_t * *res*)

Función POP dentro de la interrupción.

Parámetros

| | |
|------------------|-----------------------|
| <i>registros</i> | Registros del 0 al 15 |
| <i>memoria</i> | Dirección de memoria |
| <i>res</i> | Registros en memoria |

4.13.1.11. void STR (uint32_t * *Rt*, uint32_t * *Rn*, uint32_t * *Rm*, uint32_t * *memory*)

Función que almacena 4 valores en la pila.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la dirección donde se almacenarán los datos |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicará en memoria para almacenar el valor |

4.13.1.12. void STRB (uint32_t * *Rt*, uint32_t * *Rn*, uint32_t * *Rm*, uint32_t * *memory*)

Función que almacena un valor en la pila ya sea los primeros 8 bits o los últimos 8 bits.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la dirección donde se almacenarán los datos |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicará en memoria para almacenar el valor |

4.13.1.13. void STRH (uint32_t * *Rt*, uint32_t * *Rn*, uint32_t * *Rm*, uint32_t * *memory*)

Función que almacena dos valores en la pila ya sea los primeros 16 bits o los últimos 16 bits.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la direccion donde se almacenaran los datos |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicará en memoria para almacenar el valor |

4.14. Referencia del Archivo ram.h

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

'defines'

- #define **MEMORIA** 64
- #define **DIRMAXMEM** 255

Funciones

- void **inimemoria** (uint32_t *memoria, int tama)
Funcion que inicia la memoria, es decir comienza a desde el valor maximo y va disminuye (descendente)
- void **mostrar_memoria** (uint32_t *memoria, int tama)
Funcion que permite mostrar los valores almacenados en la memoria.
- void **PUSH** (uint32_t *registros, uint32_t *memory, uint32_t *res)
Funcion que permite realizar el PUSH en la memoria.
- void **POP** (uint32_t *registros, uint32_t *memory, uint32_t *res)
Funcion que permite realizar el POP en la memoria.
- uint32_t **bitcount** (uint32_t *R)
- void **POPINTERRUPT** (uint32_t *registros, uint32_t *memory, uint32_t *res)
Funcion POP dentro de la interrupcion.
- void **PUSHINTERRUPT** (uint32_t *registros, uint32_t *memory, uint32_t *res)
Funcion POP dentro de la interrupcion.
- void **LDR** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)
Funcion que extrae 4 valores de la pila dependiendo de la suma de las direcciones.
- void **LDRB** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)
Funcion que extrae un valor de la pila dependiendo de la suma de la direccion con otra direccion de registro.
- void **LDRH** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)
Funcion que extrae dos valores de la pila dependiendo de la suma de la direccion con un valor inmediato adicionalmente se realiza extension de ceros.
- void **STR** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)
Funcion que almacena 4 valores en la pila.
- void **STRB** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)
Funcion que almacena un valor en la pila ya sea los primeros 8 bit o los 8 bit ultimos.
- void **STRH** (uint32_t *Rt, uint32_t Rn, uint32_t Rm, uint32_t *memory)
Funcion que almacena dos valores en la pila ya sea los 16 primeros 16bit o los ultimos 16 bits.

4.14.1. Documentación de los 'defines'

4.14.1.1. `#define DIRMAMEM 255`

4.14.1.2. `#define MEMORIA 64`

4.14.2. Documentación de las funciones

4.14.2.1. `uint32_t bitcount (uint32_t * R)`

4.14.2.2. `void inmemoria (uint32_t * memoria, int tama)`

Funcion que inicia la memoria, es decir comienza a desde el valor maximo y va disminuye (descendente)

Parámetros

| | |
|----------------|---|
| <i>memoria</i> | Direccion en la cual se almacenan los datos en la memoria |
| <i>tama</i> | Es el tamaño de los datos a almacenar |

4.14.2.3. `void LDR (uint32_t * Rt, uint32_t Rn, uint32_t Rm, uint32_t * memory)`

Funcion que extrae 4 valores de la pila dependiendo de la suma de las direcciones.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la direccion de donde se extrae los valores |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicara en memoria para obtener el valor |

4.14.2.4. `void LDRB (uint32_t * Rt, uint32_t Rn, uint32_t Rm, uint32_t * memory)`

Funcion que extrae un valor de la pila dependiendo de la suma de la direccion con otra direccion de registro.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la direccion de donde se extrae el valor |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicara en memoria para obtener el valor |

4.14.2.5. `void LDRH (uint32_t * Rt, uint32_t Rn, uint32_t Rm, uint32_t * memory)`

Funcion que extrae dos valores de la pila dependiendo de la suma de la direccion con un valor inmediato adicionalmente se realiza extension de ceros.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la direccion de donde se extrae los valores |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicara en memoria para obtener el valor |

4.14.2.6. void mostrar_memoria (uint32_t * *memoria*, int *tama*)

Funcion que permite mostrar los valores almacenados en la memoria.

Parámetros

| | |
|----------------|---|
| <i>memoria</i> | Direccion en la cual se almacenan los datos en la memoria |
| <i>tama</i> | Es el tamaño de los datos a almacenar |

4.14.2.7. void POP (uint32_t * *registros*, uint32_t * *memory*, uint32_t * *res*)

Funcion que permite realizar el POP en la memoria.

Parámetros

| | |
|------------------|---|
| <i>registros</i> | Direccion de los registros que van del 0 al 15 |
| <i>memoria</i> | Direccion en la cual se almacenan los datos en la memoria |
| <i>res</i> | Direccion de registro |

4.14.2.8. void POPINTERRUPT (uint32_t * *registros*, uint32_t * *memory*, uint32_t * *res*)

Funcion POP dentro de la interrupcion.

Parámetros

| | |
|------------------|-----------------------|
| <i>registros</i> | Registros del 0 al 15 |
| <i>memoria</i> | Direccion de memoria |
| <i>rest</i> | Registros en memoria |

4.14.2.9. void PUSH (uint32_t * *registros*, uint32_t * *memory*, uint32_t * *res*)

Funcion que permite realizar el PUSH en la memoria.

Parámetros

| | |
|------------------|---|
| <i>registros</i> | Direccion de los registros que van del 0 al 15 |
| <i>memoria</i> | Direccion en la cual se almacenan los datos en la memoria |
| <i>res</i> | Direccion de registro |

4.14.2.10. void PUSHINTERRUPT (uint32_t * *registros*, uint32_t * *memory*, uint32_t * *res*)

Funcion POP dentro de la interrupcion.

Parámetros

| | |
|------------------|-----------------------|
| <i>registros</i> | Registros del 0 al 15 |
| <i>memoria</i> | Direccion de memoria |
| <i>res</i> | Registros en memoria |

4.14.2.11. void STR (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint32_t * *memory*)

Funcion que almacena 4 valores en la pila.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la direccion donde se almacenaran los datos |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicará en memoria para almacenar el valor |

4.14.2.12. void STRB (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint32_t * *memory*)

Función que almacena un valor en la pila ya sea los primeros 8 bit o los 8 bit últimos.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la direccion donde se almacenaran los datos |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicará en memoria para almacenar el valor |

4.14.2.13. void STRH (uint32_t * *Rt*, uint32_t *Rn*, uint32_t *Rm*, uint32_t * *memory*)

Función que almacena dos valores en la pila ya sea los 16 primeros 16bit o los últimos 16 bits.

Parámetros

| | |
|---------------|---|
| <i>Rt</i> | Es la direccion donde se almacenaran los datos |
| <i>Rn</i> | Dirección que contiene el valor del registro a operar |
| <i>Rm</i> | Dirección que contiene el valor del registro a operar |
| <i>memory</i> | Dirección donde se ubicará en memoria para almacenar el valor |

4.15. Referencia del Archivo salto.c

```
#include <stdint.h>
#include <stdio.h>
#include "flags.h"
#include "alu.h"
#include <stdbool.h>
#include <curses.h>
```

Funciones

- void **B** (uint32_t *pc, uint32_t valor)
Función que realiza un salto en una dirección específica.
- void **BEQ** (uint32_t *pc, uint32_t valor)
Función que realiza un salto teniendo en cuenta que la bandera sea Z == 1.
- void **BNE** (uint32_t *pc, uint32_t valor)
Función que realiza un salto teniendo en cuenta que la bandera sea Z == 0.
- void **BCS** (uint32_t *pc, uint32_t valor)
Función que realiza un salto teniendo en cuenta que la bandera sea C == 1.
- void **BCC** (uint32_t *pc, uint32_t valor)
Función que realiza un salto teniendo en cuenta que la bandera sea C == 0.
- void **BMI** (uint32_t *pc, uint32_t valor)
Función que realiza un salto teniendo en cuenta que la bandera sea N == 1.
- void **BPL** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea $N == 0$.

- void **BVS** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea $V == 1$.

- void **BVC** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea $V == 0$.

- void **BHI** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que las banderas sean $C == 1$ y $Z == 0$.

- void **BLS** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que las banderas sean $C == 0$ o $Z == 1$.

- void **BGE** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea $N == V$.

- void **BLT** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea $N != V$.

- void **BGT** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que las banderas sean $Z == 1$ y $N == V$.

- void **BLE** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que las banderas sean $Z == 0$ o $N != V$.

- void **BAL** (uint32_t *pc, uint32_t valor)

Función Función que realiza siempre un salto no se tiene condición.

- void **BL** (uint32_t *pc, uint32_t valor, uint32_t *LR)

Función que llama a una subrutina que se encuentra en una dirección relativa al pc.

- void **BX** (uint32_t *pc, uint32_t *LR)

Función Función que realiza un salto a una dirección específica por un registro.

Variables

- bool **banderas** [4]
- uint32_t **LR**

4.15.1. Documentación de las funciones

4.15.1.1. void B (uint32_t * pc, uint32_t valor)

Función que realiza un salto en una dirección específica.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.2. void BAL (uint32_t * pc, uint32_t valor)

Función Función que realiza siempre un salto no se tiene condición.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.3. void BCC (uint32_t * pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea $C == 0$.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.4. void BCS (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea C == 1.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.5. void BEQ (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea Z == 1.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.6. void BGE (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea N == V.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.7. void BGT (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que las banderas sean Z == 1 y N == V.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.8. void BHI (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que las banderas sean C == 1 y Z == 0.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.9. void BL (uint32_t * *pc*, uint32_t *valor*, uint32_t * *LR*)

Función que llama a una subrutina que se encuentra en una dirección relativa al pc .

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.10. void BLE (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que las banderas sean Z == 0 o N != V.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.11. void BLS (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que las banderas sean C == 0 o Z==1.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.12. void BLT (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea N != V.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.13. void BMI (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea N == 1.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.14. void BNE (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea Z == 0.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.15. void BPL (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea N == 0.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.16. void BVC (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea $V == 0$.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.17. void BVS (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea $V == 1$.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.15.1.18. void BX (uint32_t * *pc*, uint32_t * *LR*)

Función Función que realiza un salto a una dirección específica por un registro.

Parámetros

| | |
|-----------|------------------------|
| <i>pc</i> | Contador del programa. |
|-----------|------------------------|

4.15.2. Documentación de las variables

4.15.2.1. bool banderas[4]

4.15.2.2. uint32_t LR

4.16. Referencia del Archivo salto.h

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

Funciones

- void **B** (uint32_t *pc, uint32_t valor)
Función que realiza un salto en una dirección específica.
- void **BEQ** (uint32_t *pc, uint32_t valor)
Función que realiza un salto teniendo en cuenta que la bandera sea $Z == 1$.
- void **BNE** (uint32_t *pc, uint32_t valor)
Función que realiza un salto teniendo en cuenta que la bandera sea $Z == 0$.
- void **BCS** (uint32_t *pc, uint32_t valor)
Función que realiza un salto teniendo en cuenta que la bandera sea $C == 1$.
- void **BCC** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea C == 0.

- void **BMI** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea N == 1.

- void **BPL** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea N == 0.

- void **BVS** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea V == 1.

- void **BVC** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea V == 0.

- void **BHI** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que las banderas sean C == 1 y Z == 0.

- void **BLS** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que las banderas sean C == 0 o Z==1.

- void **BGE** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea N == V.

- void **BLT** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea N != V.

- void **BGT** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que las banderas sean Z == 1 y N == V.

- void **BLE** (uint32_t *pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que las banderas sean Z == 0 o N != V.

- void **BAL** (uint32_t *pc, uint32_t valor)

Función Función que realiza siempre un salto no se tiene condición.

- void **BL** (uint32_t *pc, uint32_t valor, uint32_t *LR)

Función que llama a una subrutina que se encuentra en una dirección relativa al pc .

- void **BX** (uint32_t *pc, uint32_t *LR)

Función Función que realiza un salto a una dirección específica por un registro.

4.16.1. Documentación de las funciones

4.16.1.1. void B (uint32_t * pc, uint32_t valor)

Función que realiza un salto en una dirección específica.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.2. void BAL (uint32_t * pc, uint32_t valor)

Función Función que realiza siempre un salto no se tiene condición.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.3. void BCC (uint32_t * pc, uint32_t valor)

Función que realiza un salto teniendo en cuenta que la bandera sea C == 0.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.4. void BCS (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea C == 1.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.5. void BEQ (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea Z == 1.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.6. void BGE (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea N == V.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.7. void BGT (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que las banderas sean Z == 1 y N == V.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.8. void BHI (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que las banderas sean C == 1 y Z == 0.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.9. void BL (uint32_t * *pc*, uint32_t *valor*, uint32_t * *LR*)

Función que llama a una subrutina que se encuentra en una dirección relativa al pc .

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.10. void BLE (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que las banderas sean Z == 0 o N != V.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.11. void BLS (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que las banderas sean C == 0 o Z==1.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.12. void BLT (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea N != V.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.13. void BMI (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea N == 1.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.14. void BNE (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea Z == 0.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.15. void BPL (uint32_t * *pc*, uint32_t *valor*)

Función que realiza un salto teniendo en cuenta que la bandera sea N == 0.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.16. `void BVC (uint32_t * pc, uint32_t valor)`

Función que realiza un salto teniendo en cuenta que la bandera sea $V == 0$.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.17. `void BVS (uint32_t * pc, uint32_t valor)`

Función que realiza un salto teniendo en cuenta que la bandera sea $V == 1$.

Parámetros

| | |
|--------------|---|
| <i>pc</i> | Contador del programa. |
| <i>valor</i> | Es un valor inmediato sin signo de 32 bits. |

4.16.1.18. `void BX (uint32_t * pc, uint32_t * LR)`

Función Función que realiza un salto a una dirección específica por un registro.

Parámetros

| | |
|-----------|------------------------|
| <i>pc</i> | Contador del programa. |
|-----------|------------------------|

4.17. Referencia del Archivo screen.c

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

Funciones

- void [showRegisters](#) (uint32_t *registers, size_t len)

4.17.1. Documentación de las funciones

4.17.1.1. `void showRegisters (uint32_t * registers, size_t len)`

4.18. Referencia del Archivo screen.h

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

```
#include <stdlib.h>
#include <stdint.h>
#include <urses.h>
#include "colors.h"
```

Funciones

- void `showRegisters` (uint32_t **registers*, size_t *len*)

4.18.1. Documentación de las funciones

4.18.1.1. void `showRegisters` (uint32_t * *registers*, size_t *len*)