## Register file and its addressing with Rb/Ra fields

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rb/Ra = 1111 | | **RF/GF** | | Rb/Ra = 0001, G=111 | | G-index, 3 bits |
| Rb/Ra = 1110 | | **RE/GR** | | Rb/Ra = 0000, G=111 | | | |
| Rb/Ra = 1101 | | **RD/GD** | | Rb/Ra = 0001, G=110 | | | |
| Rb/Ra = 1100 | | **RC/GC** | | Rb/Ra = 0000, G=110 | | | |
| Rb/Ra = 1011 | **RB** | | **GB** | | Rb/Ra = 0001, G=101 | | |
| Rb/Ra = 1010 | **RA** | | **GF** | | Rb/Ra = 0000, G=101 | | |
| Rb/Ra = 1001 | **R9** | | **G9** | | Rb/Ra = 0001, G=100 | | |
| Rb/Ra = 1000 | **R8** | | **G8** | | Rb/Ra = 0000, G=100 | | |
| Rb/Ra = 0111 | **R7** | | **G7** | | Rb/Ra = 0001, G=011 | | |
| Rb/Ra = 0110 | **R6** | | **G6** | | Rb/Ra= 0000, G=011 | | |
| Rb/Ra = 0101 | **R5** | | **G5** | | Rb/Ra = 0001, G=010 | | |
| Rb/Ra = 0100 | **R4** | | **G4** | | Rb/Ra = 0000, G=010 | | |
| Rb/Ra = 0011 | **R3** | | **G3** | | Rb/Ra = 0001, G=001 | | |
| Rb/Ra = 0010 | **R2** | | **G2** | | Rb/Ra= 0000, G=001 | | |
|  |  | | **G1** | | Rb/Ra = 0001, G=000 | | |
|  |  | | **G0** | | Rb/Ra = 0000, G=000 | | |

## Arithmetic operation flags

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **NB** | **ZB** | **C4** | **C8** | **N** | **Z** | **V** | **C** |

|  |  |  |
| --- | --- | --- |
| **NB** | This flag is set if the most significant bit of the result of a byte or a word operation is a 1. This flag is cleared otherwise (except for SRW and SRWC). | always updated |
| **ZB** | This flag is set if the result of a byte or a word operation is "0". This flag is cleared otherwise. | always updated |
| **C4** | This flag is used only for operations involving decimal arithmetic. It is set if a carry from the third bit to the fourth bit is a "1". This flag is cleared otherwise. | arithmetic ops |
| **C8** | This flag is set if the carry from the most significant bit is a "1" or if the result of a shift operation is to shift off a "1". This flag is cleared otherwise. Note that this status bit is not set to borrow for subtract as in the case with the C Condition Code Flag. | arithmetic ops |
| **N** | This flag is set if the most significant bit of the result of a byte or a word operation is a "1". This flag is cleared otherwise. | updated if s |
| **Z** | This flag is set if the result of a byte or a word operation is zero. This flag is cleared otherwise. | updated if s |
| **V** | This flag is set if an arithmetic operation results in an overflow. This flag is cleared if no overflow occurs or if operation performed is not an arithmetic operation. | arithmetic ops |
| **C** | This flag monitors bits which are carried, borrowed, or shifted as follows:   * Add and Increment - this flag is set if there is a carry from the most significant bit of a byte or a word operation. This flag is cleared otherwise. * Subtract and Decrement - this flag is set if there is a borrow (complement of carry) from the most significant bit of a byte or a word operation. This flag is cleared otherwise. * Shift - This flag is set if the result of a right or left shift operation causes a "1" to shift off the end of the byte or word. This flag is cleared otherwise.   Note that the C flag is affected only for operations in the areas listed above. | add/, inc/dec,  shifts |

**Register pairs Rb:Ra for word opearions**

|  |  |  |
| --- | --- | --- |
|  | **All, except word right shifts** | **Word right shift** |
| **aaa0** | normal operation | not recommended |
| **aaa1** | swap bytes Ra:Ra-1 | normal operation |
| **bbb0** | normal operation | not recommended |
| **bbb1** | [sign extension of Rb]:R­b | normal operation |

**s** - force update flags NZVC in PSW

**bbbb** - address field for the first operand

**aaaa** - address field for the second operand

**LLLL** - literal data

**dddd** - jump target address

**xIII** - interrupt mask

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0000 0ddd dddd dddd | JMP | addr12 | LC = addr; | - | 2 |
| 0000 1xxx xxxx xxx | RFS |  | LC = RR; (return from subroutine) | - | 2 |
| 0001 cond dddd dddd | Conditional jumps within 256 microinstructions page | | | - | 2 |
| 0001 0000 dddd dddd | JZBF | addr8 | if (!ZB) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JZBT | addr8 | if (ZB) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JC8F | addr8 | if (!C8) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JC8T | addr8 | if (C8) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JIF | addr8 | if (!ICC) LC[7:0] = addr8; (indirect condition code) | - | 2 |
| 0001 0000 dddd dddd | JIT | addr8 | if (ICC) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JNBF | addr8 | if (!NB) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JNBT | addr8 | if (NB) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JZF | addr8 | if (!Z) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JZT | addr8 | if (Z) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JCF | addr8 | if (!C) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JCT | addr8 | if (C) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JVF | addr8 | if (!V) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JVT | addr8 | if (V) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JNF | addr8 | if (!N) LC[7:0] = addr8; | - | 2 |
| 0001 0000 dddd dddd | JNT | addr8 | if (N) LC[7:0] = addr8; | - | 2 |
| 0010 LLLL LLLL aaaa | AL | data8, areg | Ra = Ra + data8; NZVC; | nz48---- | 1 |
| 0011 LLLL LLLL aaaa | CL | data8, areg | Ra - data8; NZVC; | nz48---- | 1 |
| 0100 LLLL LLLL aaaa | NL | data8, areg | Ra = Ra & data8 | nz------ | 1 |
| 0101 LLLL LLLL aaaa | TL | data8, areg | Ra & data8; | nz------ | 1 |
| 0110 LLLL LLLL aaaa | LL | data8, areg | Ra = data8 | nz------ | 1 |
| 0111 0000 xIII xxxx | RI | imask8 | if imask8[6] I6=0; if imask8[5] I5=0; if imask8[4] I4 =0; | - | 1 |
| 0111 0001 xIII xxxx | SI | imask8 | if imask8[6] I6=1; if imask8[5] I5=1; if imask8[4] I4 =1; | - | 1 |
| 0111 0010 xxxx aaaa | CCF | areg | Ra = nz48NZCF; | - | 1 |
| 0111 0011 bbbb aaaa | LCF | mask4, areg | nz48 = Ra[7:4], NZVC = NZVC & ~bbbb | Ra & bbbb; | nz48NZVC | 1 |
| 0111 0100 xxxx xxxx | RTSR |  | TSR[2:0] = 0 | - | 1 |
| 0111 0101 xxxx aaaa | LGL | areg | G[2:0] = Ra; | - | 1 |
| 0111 0110 xxxx aaaa | CIB | areg | if (C8) {Ra = Rb+1; NZVC;} | nz48---- | 1 |
| 0111 0111 xxxx aaaa | CDB | areg | if (C8) {Ra = Rb-1; NZVC;} | nz48---- | 1 |
| 0111 1xxx xxxx xxxx | - |  |  |  |  |
| 1000 000s bbbb aaaa | MBs | breg, areg | Ra = Rb; NZ, V=0; | nz--NZ0- | 1 |
| 1000 001s bbbb aaaa | MWs | breg, areg | Ra+1:Ra = Rb+1Rb; NZ, V=0; | nz--NZ0- | 2 |
| 1000 010s bbbb aaaa | CMBs | breg, areg | if (C) {Ra = Rb; NZ, V=0;} | nz--NZ0- | 1 |
| 1000 011s bbbb aaaa | CMWs | breg, areg | if (C) { Ra+1:Ra = Rb+1Rb ; NZ, V=0;} | nz--NZ0- | 2 |
| 1000 100s bbbb aaaa | SLBCs | breg, areg | Ra = (Rb<< 1) | C; NZVC; | nz48NZVC | 1 |
| 1000 101s bbbb aaaa | SLWCs | breg, areg | Ra+1:Ra = (Rb+1:Rb+1 << 1) | C; NZVC; | nz48NZVC | 2 |
| 1000 110s bbbb aaaa | SLBs | breg, areg | Ra = Rb<< 1; NZVC; | nz48NZVC | 1 |
| 1000 111s bbbb aaaa | SLWs | breg, areg | Ra+1:Ra = Rb+1:Rb+1 << 1; NZVC; | nz48NZVC | 2 |
| 1001 000s bbbb aaaa | ICB1s | breg, areg | Ra = Rb+1; NZVC; | nz48NZVC | 1 |
| 1001 001s bbbb aaaa | ICW1s | breg, areg | Ra+1:Ra = Rb+1:Rb+1; NZVC; | nz48NZVC | 2 |
| 1001 010s bbbb aaaa | ICB2s | breg, areg | Ra = Rb+2; NZVC; | nz48NZVC | 1 |
| 1001 011s bbbb aaaa | ICW2s | breg, areg | Ra+1:Ra = Rb+1:Rb+2; NZVC; | nz48NZVC | 2 |
| 1001 100s bbbb aaaa | TCBs | breg, areg | Ra = ~Rb+1; NZVC; | nz48NZVC | 1 |
| 1001 101s bbbb aaaa | TCWs | breg, areg | Ra+1:Ra = ~Rb+1:~Rb+1; NZVC; | nz48NZVC | 2 |
| 1001 110s bbbb aaaa | OCBs | breg, areg | Ra = ~Rb; NZ, V=0; | nz00NZ00 | 1 |
| 1001 111s bbbb aaaa | OCWs | breg, areg | Ra+1:Ra = ~Rb+1:~Rb; NZ, V=0; | nz00NZ00 | 2 |
| 1010 000s bbbb aaaa | ABs | breg, areg | Ra = Ra + Rb; NZVC; | nz48NZVC | 1 |
| 1010 001s bbbb aaaa | AWs | breg, areg | Ra+1:Ra = Ra+1:Ra + Rb+1:Rb; NZVC; | nz48NZVC | 2 |
| 1010 010s bbbb aaaa | CABs | breg, areg | if (C) {Ra = Ra + Rb; NZVC;} | nz48NZVC | 1 |
| 1010 011s bbbb aaaa | CAWs | breg, areg | if (C) {Ra+1:Ra = Ra+1:Ra + Rb+1:Rb; NZVC;} | nz48NZVC | 2 |
| 1010 100s bbbb aaaa | ABCs | breg, areg | Ra = Ra + Rb + C; NZVC; | nz48NZVC | 1 |
| 1010 101s bbbb aaaa | AWCs | breg, areg | Ra+1:Ra = Ra+1:Ra + Rb+1:Rb + C; NZVC; | nz48NZVC | 2 |
| 1010 1100 bbbb aaaa | CAD | breg, areg | if (!C4) Ra[3:0] += Rb[3:0]; if (!C8) Ra[7:4] += Rb[7:4];NZVC; | nz48NZ01 | 1 |
| 1010 1101 xxxx xxxx | - |  |  |  |  |
| 1010 111s bbbb aaaa | CAWIs | breg, areg | if (ICC) {Ra+1:Ra = Ra+1:Ra + Rb+1:Rb; NZVC;} | nz48NZ01 | 2 |
| 1011 000s bbbb aaaa | SBs | breg, areg | Ra = Ra - Rb; NZVC; | nz48NZVC | 1 |
| 1011 001s bbbb aaaa | SWs | breg, areg | Ra+1:Ra = Ra+1:Ra - Rb+1:Rb; NZVC; | nz48NZVC | 2 |
| 1011 010s bbbb aaaa | CBs | breg, areg | Ra - Rb; NZVC; | nz48NZVC | 1 |
| 1011 011s bbbb aaaa | CWs | breg, areg | Ra+1:Ra - Rb+1:Rb; NZVC; | nz48NZVC | 2 |
| 1011 100s bbbb aaaa | SBCs | breg, areg | Ra = Ra - Rb - C; NZVC; | nz48NZVC | 1 |
| 1011 101s bbbb aaaa | SWCs | breg, areg | Ra+1:Ra = Ra+1:Ra - Rb+1:Rb - C; NZVC; | nz48NZVC | 2 |
| 1011 110s bbbb aaaa | DB1s | breg, areg | Ra = Rb-1; NZVC; | nz48NZ01 | 1 |
| 1011 111s bbbb aaaa | DW1s | breg, areg | Ra+1:Ra = Rb+1:Rb-1; NZVC; | nz48NZ01 | 2 |
| 1100 000s bbbb aaaa | NBs | breg, areg | Ra = Ra & Rb; NZ, V=0; | nz--NZ0- | 1 |
| 1100 001s bbbb aaaa | NWs | breg, areg | Ra+1:Ra = Ra+1:Ra & Rb+1:Rb; NZ, V=0; | nz--NZ0- | 2 |
| 1100 010s bbbb aaaa | TBs | breg, areg | Ra & Rb; NZ, V=0; | nz--NZ0- | 1 |
| 1100 011s bbbb aaaa | TWs | breg, areg | Ra+1:Ra & Rb+1:Rb; NZ, V=0; | nz--NZ0- | 2 |
| 1100 100s bbbb aaaa | ORBs | breg, areg | Ra = Ra | Rb; NZ, V=0; | nz--NZ0- | 1 |
| 1100 101s bbbb aaaa | ORWs | breg, areg | Ra+1:Ra = Ra+1:Ra | Rb+1:Rb; NZ, V=0; | nz--NZ0- | 2 |
| 1100 110s bbbb aaaa | XBs | breg, areg | Ra = Ra ^ Rb; NZ, V=0; | nz--NZ0- | 1 |
| 1100 111s bbbb aaaa | XWs | breg, areg | Ra+1:Ra = Ra+1:Ra ^ Rb+1:Rb; NZ, V=0; | nz--NZ0- | 2 |
| 1101 000s bbbb aaaa | NCBs | breg, areg | Ra = Ra & ~Rb; NZ, V=0; | nz--NZ0- | 1 |
| 1101 001s bbbb aaaa | NCWs | breg, areg | Ra+1:Ra = Ra+1:Ra & ~(Rb+1:Rb); NZ, V=0; | nz--NZ0- | 2 |
| 1101 01xx xxxx xxxx | - |  |  |  |  |
| 1101 100s bbbb aaaa | SRBCs | breg, areg | Ra = (Rb >> 1) | (C << 7); NZVC; | nz08NZ0- | 1 |
| 1101 101s bbbb aaaa | SRWCs | breg, areg | Ra+1:Ra = (Rb+1:Rb >> 1) | ( C<<15); NZVC; | nz08NZ0- | 2 |
| 1101 110s bbbb aaaa | SRBs | breg, areg | Ra = Rb >> 1; NZVC; | nz08NZ0- | 1 |
| 1101 111s bbbb aaaa | SRWs | breg, areg | Ra+1:Ra = Rb+1:Rb >> 1; NZVC; | nz08NZ0- | 2 |
| 1110 000s xbbb aaaa | IBs | breg, areg | Ra = DAL[7:0] or DAT[15:8]; NZ, V=0; depending on Rb field:  - 0 - upper byte, DAL[15:8]  - 1 - lower byte, DAL[7:0]  - 2 - upper byte if A[0]=1,lower byte if A[0]=0  - 3 - lower byte if A[0]=1, upper byte if A[0]=0  - 4 - upper byte, DAL[15:8], RMW  - 5- lower byte, DAL[7:0], RMW  - 6 - upper byte if A[0]=1, lower byte if A[0]=0, RMW  - 7 - lower byte if A[0]=1, upper byte if A[0]=0, RMW | nz--NZ0- | 1 |
| 1110 001s xbbb aaaa | IWs | breg, areg | Ra+1:Ra = DAL; NZ, V=0; depending on Rb field:  - 0 - load designated registers only  - 1 - load TR, G=DAL[6:4], set ICC flag  - 2 - load TR, G=DAL[8:6], set ICC flag  - 3 - load TR, set ICC flag  - 4 - READ-MODIFY-WRITE (RMW)  - 5 - load TR, G=DAL[6:4], set ICC flag, RMW  - 6 - load TR, G=DAL[8:6], set ICC flag. RMW  - 7 - load TR, set ICC flag, RMW | nz--NZ0- | 2 |
| 1110 010s xxbb aaaa | ISBs | breg, areg | Ra = DAL[7:0] or DAT[15:8]; NZ, V=0; depending on Rb field:  - 0/4 - upper byte, DAL[15:8]  - 1/5 - lower byte, DAL[7:0]  - 2/6 - upper byte if A[0]=1,lower byte if A[0]=0  - 3/7 - lower byte if A[0]=1, upper byte if A[0]=0 | nz--NZ0- | 1 |
| 1110 011s xxxx aaaa | ISWs | breg, areg | Ra+1:Ra = DAL; NZ, V=0; | nz--NZ0- | 2 |
| 1110 10xx xxxx xxxx | - |  |  |  |  |
| 1110 110s bbbb aaaa | MIs | breg, areg | next MI |= Rb:Ra; | - | 1 |
| 1110 1110 xxxx xxxx | LTR | breg, areg |  | - | 2 |
| 1110 1111 xxxx xxxx | - |  |  |  |  |
| 1111 0000 bbbb aaaa | RIB1 | breg, areg | DAL = Rb:Ra; Ra = Ra + 1; | nz48---- | 1 |
| 1111 0001 bbbb aaaa | WIB1 | breg, areg | DAL = Rb:Ra; Ra = Ra + 1; | nz48---- | 1 |
| 1111 0010 bbbb aaaa | RIW1 | breg, areg | DAL = Rb:Ra; Rb:Ra = Rb:Ra + 1; | nz48---- | 2 |
| 1111 0011 bbbb aaaa | WIW1 | breg, areg | DAL = Rb:Ra; Rb:Ra = Rb:Ra + 1; | nz48---- | 2 |
| 1111 0100 bbbb aaaa | RIB2 | breg, areg | DAL = Rb:Ra; Ra = Ra + 2; | nz48---- | 1 |
| 1111 0101 bbbb aaaa | WIB2 | breg, areg | DAL = Rb:Ra; Ra = Ra + 2; | nz48---- | 1 |
| 1111 0110 bbbb aaaa | RIW2 | breg, areg | DAL = Rb:Ra; Rb:Ra = Rb:Ra + 2; | nz48---- | 2 |
| 1111 0111 bbbb aaaa | WIW2 | breg, areg | DAL = Rb:Ra; Rb:Ra = Rb:Ra + 2; | nz48---- | 2 |
| 1111 1000 bbbb aaaa | R | breg, areg | DAL = Rb:Ra; | - | 1 |
| 1111 1001 bbbb aaaa | W | breg, areg | DAL = Rb:Ra; | - | 1 |
| 1111 1010 bbbb aaaa | RA | breg, areg | DAL = Rb:Ra; | - | 1 |
| 1111 1011 bbbb aaaa | WA | breg, areg | DAL = Rb:Ra; | - | 1 |
| 1111 1100 bbbb aaaa | OB | breg, areg | DAL = Rb:Ra; | - | 1 |
| 1111 1101 bbbb aaaa | OW | breg, areg | DAL = Rb:Ra; | - | 1 |
| 1111 1110 bbbb aaaa | OS | breg, areg | DAL = Rb:Ra; | - | 1 |
| 1111 1111 xxxx xxxx | NOP |  | no operation | - | 1 |