

| Befehl | Op1 Dest | Op2 | 15:13 OC(3:1) | 12 OC(0) | Imm= OC(0) | 11:8 Subcode Subcode Subcode Imm(3:0) Imm(15:12) Bedingung | 7:4 RegA RegA RegA Imm(11:8) Disp(7:4) | 3:0 RegB Imm(3:0) Subsubcode RegB Imm(7:4) Disp(3:0) | Flags | Reg_Mux | Intern_Mux | Operation |
|--------|----------|-------|------------------|-------------|---------------|--|---|--|------------|---------|------------|---|
| sub | reg | reg | 0 | 0 | 0 | 0000 | Op1 | Op2 | C, O, N, Z | ADDSUB | xxx | Op1 := Op1 - Op2 imm(15:4) := 0 |
| subi | reg | i4/16 | 0 | 1 | 1 | 0000 | Op1 | Op2 | C, O, N, Z | ADDSUB | xxx | Op1 := Op1 - Op2 imm(15:4) := 0 |
| sbb | reg | reg | 0 | 0 | 0 | 0001 | Op1 | Op2 | C, O, N, Z | ADDSUB | xxx | Op1 := Op1 - Op2 - CF imm(15:4) := 0 |
| sbbi | reg | i4/16 | 0 | 1 | 1 | 0000 | Op1 | Op2 | C, O, N, Z | ADDSUB | xxx | Op1 := Op1 - Op2 - CF imm(15:4) := 0 |
| add | reg | reg | 0 | 0 | 0 | 0010 | Op1 | Op2 | C, O, N, Z | ADDSUB | xxx | Op1 := Op1 + Op2 imm(15:4) := 0 |
| addi | reg | i4/16 | 0 | 1 | 1 | 0010 | Op1 | Op2 | C, O, N, Z | ADDSUB | xxx | Op1 := Op1 + Op2 imm(15:4) := 0 |
| adc | reg | reg | 0 | 0 | 0 | 0011 | Op1 | Op2 | C, O, N, Z | ADDSUB | xxx | Op1 := Op1 + Op2 + CF imm(15:4) := 0 |
| adci | reg | i4/16 | 0 | 1 | 1 | 0011 | Op1 | Op2 | C, O, N, Z | ADDSUB | xxx | Op1 := Op1 + Op2 + CF imm(15:4) := 0 |
| cmp | reg | reg | 0 | 0 | 0 | 0100 | Op1 | Op2 | C, O, N, Z | xxx | xxx | Op1 - Op2 imm(15:4) := 0 |
| cmpi | reg | i4/16 | 0 | 1 | 1 | 0100 | Op1 | Op2 | C, O, N, Z | xxx | xxx | Op1 - Op2 imm(15:4) := 0 |
| | | | | | | | | | | | | |
| and | reg | reg | 0 | 0 | 0 | 1000 | Op1 | Op2 | N, Z | LOGIK | xxx | Op1 := Op1 & Op2 imm(15:4) := 0 |
| andi | reg | i4/16 | 0 | 1 | 1 | 1000 | Op1 | Op2 | N, Z | LOGIK | xxx | Op1 := Op1 & Op2 imm(15:4) := 0 |
| or | reg | reg | 0 | 0 | 0 | 1001 | Op1 | Op2 | N, Z | LOGIK | xxx | Op1 := Op1 + Op2 imm(15:4) := 0 |
| ori | reg | i4/16 | 0 | 1 | 1 | 1001 | Op1 | Op2 | N, Z | LOGIK | xxx | Op1 := Op1 + Op2 imm(15:4) := 0 |
| xor | reg | reg | 0 | 0 | 0 | 1010 | Op1 | Op2 | N, Z | LOGIK | xxx | Op1 := Op1 (+) Op2 imm(15:4) := 0 |
| xori | reg | i4/16 | 0 | 1 | 1 | 1010 | Op1 | Op2 | N, Z | LOGIK | xxx | Op1 := Op1 (+) Op2 imm(15:4) := 0 |
| andn | reg | reg | 0 | 0 | 0 | 1011 | Op1 | Op2 | N, Z | LOGIK | xxx | Op1 := Op1 & ~Op2 imm(15:4) := 0 |
| andni | reg | i4/16 | 0 | 1 | 1 | 1011 | Op1 | Op2 | N, Z | LOGIK | xxx | Op1 := Op1 & ~Op2 imm(15:4) := 0 |
| test | reg | reg | 0 | 0 | 0 | 1100 | Op1 | Op2 | N, Z | xxx | xxx | Op1 * Op2 imm(15:4) := 0 |

| | | | | | | | | | | | | |
|---------|-----|---|---|---|---|------|-----|--|---------|--------|---|--|
| testi | reg | i4/16 | 0 | 1 | 1 | 1100 | Op1 | Op2 | N, Z | xxx | xxx | Op1 * Op2 imm(15:4) := 0 |
| srl | reg | | 1 | 0 | x | 0000 | Op1 | 0000 | N, Z, C | SHIFTR | xxx | Op1(14:0) := Op1(15:1) Op1(15) := 0 imm(15:4) := 0 |
| sra | reg | | 1 | 0 | x | 0000 | Op1 | 0001 | N, Z, C | SHIFTR | xxx | Op1(14:0) := Op1(15:1) Op1(15) := Op1(15) imm(15:4) := 0 |
| src | reg | | 1 | 0 | x | 0000 | Op1 | 0010 | N, Z, C | SHIFTR | xxx | Op1(14:0) := Op1(15:1) Op1(15) := Carry imm(15:4) := 0 |
| shl | reg | | 1 | 0 | x | 0000 | Op1 | 0100 | N, Z, C | SHIFTL | xxx | Op1(15:1) := Op1(14:0) Op1(0) := 0 imm(15:4) := 0 |
| slc | reg | | 1 | 0 | x | 0000 | Op1 | 0110 | N, Z, C | SHIFTL | xxx | Op1(15:1) := Op1(14:0) Op1(0) := Carry imm(15:4) := 0 |
| bitswap | reg | | 1 | 0 | x | 1001 | Op1 | 1101 | | MUX2 | BITSWAP | Op1(15:0) := Op1(0:15) imm(15:4) := 0 |
| bytswap | reg | | 1 | 0 | x | 1001 | Op1 | 1110 | | MUX2 | BYTESWAP | Op1(7:0) := Op1(15:8) Op1(15:8) := Op1(7:0) imm(15:4) := 0 |
| movs | reg | MULTH MULTL QUOT REST FLAGS IP | 1 | 0 | x | 1001 | Op1 | 1000 1001 1010 1011 1100 1111 | | MUX2 | MULTH MULTL QUOT REST FLAGS IP | Op1 := Special register imm(15:4) := 0 |
| setf | reg | | 1 | 0 | x | 01xx | Op1 | xxxx | | xxx | xxx | Flags := Op1 imm(15:4) := 0 |
| mul | reg | reg | 1 | 0 | 0 | 1100 | Op1 | Op2 | | xxx | xxx | multl:multl := Op1 * Op2 (unsigned) imm(15:4) := 0 |
| multi | reg | i4/16 | 1 | 1 | 1 | 1100 | Op1 | Op2 | | xxx | xxx | multl:multl := Op1 * Op2 (unsigned) imm(15:4) := 0 |
| imul | reg | reg | 1 | 0 | 0 | 1101 | Op1 | Op2 | | xxx | xxx | multl:multl := Op1 * Op2 (signed) imm(15:4) := 0 |
| imuli | reg | i4/16 | 1 | 1 | 1 | 1101 | Op1 | Op2 | | xxx | xxx | multl:multl := Op1 * Op2 (signed) imm(15:4) := 0 |
| div | reg | reg | 1 | 0 | 0 | 1110 | Op1 | Op2 | | xxx | xxx | quot := Op1 div Op2 (unsigned) rest := Op1 mod Op2 (unsigned) imm(15:4) := 0 |
| divi | reg | i4/16 | 1 | 1 | 1 | 1110 | Op1 | Op2 | | xxx | xxx | quot := Op1 div Op2 (unsigned) rest := Op1 mod Op2 (unsigned) imm(15:4) := 0 |

| | | | | | | | | | | | | |
|-------------|-----------|-----------|---|---|---|------------|-----------|-----------|--|----------|-----|--|
| idiv (res) | reg | reg | 1 | 0 | 0 | 1111 | Op1 | Op2 | | xxx | xxx | quot := Op1 div Op2 (signed) rest := Op1 mod Op2 (signed) imm(15:4) := 0 |
| idivi (res) | reg | i4/16 | 1 | 1 | 1 | 1111 | Op1 | Op2 | | xxx | xxx | quot := Op1 div Op2 (signed) rest := Op1 mod Op2 (signed) imm(15:4) := 0 |
| st | [reg+i16] | reg | 2 | 0 | x | Imm(3:0) | Op2 | Op1.Reg | | xxx | xxx | RAM/ROM1(Op1) := Op2 imm(15:4) := 0 |
| stb | [reg+i16] | reg | 2 | 1 | x | Imm(3:0) | Op2 | Op1.Reg | | xxx | xxx | RAM/ROM1(Op1) := Op2(7:0) imm(15:4) := 0 |
| ld | reg | [reg+i16] | 3 | 0 | x | Imm(3:0) | Op1 | Op2.Reg | | BRAM | xxx | Op1 := RAM/ROM1(Op2) imm(15:4) := 0 |
| ldx | reg | [reg+i16] | 3 | 1 | x | Imm(3:0) | Op1 | Op2.Reg | | ExternIn | xxx | Op1 := RAM/ROM2(Op2) imm(15:4) := 0 |
| stx | [reg+i16] | reg | 4 | 0 | x | Imm(3:0) | Op2 | Op1.Reg | | xxx | xxx | RAM/ROM2(Op1) := Op2 imm(15:4) := 0 |
| stxb | [reg+i16] | reg | 4 | 1 | x | Imm(3:0) | Op2 | Op1.Reg | | xxx | xxx | RAM/ROM2(Op1) := Op2(7:0) imm(15:4) := 0 |
| call | reg+i16 | reg | 5 | 0 | x | Imm(3:0) | 1xxx | Op1.Reg | | MUX2 | IP | r[Op2] := IP, Op2 = (8..)15 IP := Op1 = Op1.Reg + Op1.Imm (unsigned) imm(15:4) := 0 Standard: Op2 = 15 bei call, Op2 = 14 bei Int |
| jmp (ret) | reg+i16 | | 5 | 0 | x | Imm(3:0) | 0xxx | Op1.Reg | | xxx | xxx | IP := Op1 = Op1.Reg + Op1.Imm (unsigned) imm(15:4) := 0 Op1.Reg darf nicht 14 sein (=> irect!) Standard für ret: Op1 = R15 + 0 implizit |
| irect | | | 5 | 0 | x | 0000 | 0xxx | 1110 | | xxx | xxx | IP := R14 imm(15:4) := 0 IEF := 1 |
| lea (mov) | reg | reg+i16 | 5 | 1 | x | Imm(3:0) | Op1 | Op2.Reg | | ADDR | xxx | Op1 := Op2 = Op2.Reg + Op2.Imm imm(15:4) := 0 |
| jxx = jrel | d8 | | 6 | 0 | x | Bedingung | Disp(7:4) | Disp(3:0) | | xxx | xxx | IP := IP+(sign_ext) Disp8 (signed) imm(15:4) := 0 |
| imm12 | i12 | | 6 | 1 | x | Imm(15:12) | Imm(11:8) | Imm(7:4) | | xxx | xxx | imm(15:4) := i12 |
| out | [reg+i16] | reg | 7 | 0 | x | Imm(3:0) | Op2 | Op1.Reg | | xxx | xxx | Outport(Op1) := Op2 imm(15:4) := 0 |
| in | reg | [reg+i16] | 7 | 1 | x | Imm(3:0) | Op1 | Op2.Reg | | ExternIn | xxx | Op1 := Inport(Op2) imm(15:4) := 0 |