# Assembler D-RISC

## Istruzioni aritmetico logiche

| Istruzione | Semantica | Registri letti | Registri scritti |
|---|---|---|---|
| ADD Ri, Rj, Rk | Reg[i]+Reg[j] → Reg[k], IC+1→ IC | {Ri, Rj} | {Rk} |
| ADDi Ri, #n, Rk | Reg[i]+n → Reg[k], IC+1→ IC | {Ri} | {Rk} |
| SUB Ri, Rj, Rk | Reg[i]-Reg[j] → Reg[k], IC+1→ IC | {Ri, Rj} | {Rk} |
| SUBiRi, #n, Rk | Reg[i]-n → Reg[k], IC+1→ IC | {Ri} | {Rk} |
| MUL Ri, Rj, Rk | Reg[i]*Reg[j] → Reg[k], IC+1→ IC | {Ri, Rj} | {Rk} |
| MULi Ri, #n, Rk | Reg[i]*n → Reg[k], IC+1→ IC | {Ri} | {Rk} |
| DIV Ri, Rj, Rk | Reg[i]/Reg[j] → Reg[k], IC+1→ IC | {Ri, Rj} | {Rk} |
| DIVi Ri, #n, Rk | Reg[i]/n → Reg[k], IC+1→ IC | {Ri} | {Rk} |
| INC Ri | Reg[i] + 1 → Reg[i], IC+1→ IC | {Ri} | {Ri} |
| DECR Ri | Reg[i] - 1 → Reg[i], IC+1→ IC | {Ri} | {Ri} |
| SHL Ri, Rj | Shift sn di Reg[j] posizioni di Reg[i] → Reg[i] | {Ri,Rj} | {Ri} |
| SHL Ri, #n | Shift sn di n posizioni di Reg[i] → Reg[i] | {Ri,Rj} | {Ri} |
| SHR Ri, Rj | Shift ds di Reg[j] posizioni di Reg[i] → Reg[i] | {Ri,Rj} | {Ri} |
| SHR Ri, #n | Shift ds di n posizioni di Reg[i] → Reg[i] | {Ri,Rj} | {Ri} |

## Istruzioni sulla memoria

| Istruzione | Semantica | Registri letti | Registri scritti |
|---|---|---|---|
| LOAD Ri, Rj, Rk | Mem[Reg[i]+Reg[j]] → Reg[k], IC+1→ IC | {Ri, Rj} | {Rk} |
| LOADi Ri, off, Rk | Mem[Reg[i]+off] → Reg[k], IC+1→ IC | {Ri} | {Rk} |
| STORE Ri, Rj, Rk | Reg[k] → Mem[Reg[i]+Reg[j]], IC+1→ IC | {Ri, Rj} | {Rk} |
| STOREi Ri, off, Rk | Reg[k] → Mem[Reg[i]+off], IC+1→ IC | {Ri} | {Rk} |
| EXCH Ri, Rj, Rk | Scambia Reg[k] con Mem[Reg[i]+Reg[j]] | {Ri, Rj, Rk} | {Rk} |

## Istruzioni di salto incondizionato

| Istruzione | Semantica | Registri letti | Registri scritti |
|---|---|---|---|
| GOTO Ri | Reg[i] → IC | {Ri} | |
| CALL Rf, Rret | IC + 1 → Reg[ret], Reg[f] → IC | {Rf} | {Rret} |
| GOTO etich | IC + etich → IC | | |

## Istruzioni di salto condizionato

| Istruzione | Semantica | Registri letti | Registri scritti |
|---|---|---|---|
| IF< Ri, Rj, etich | (Reg[i]<Reg[j] ? IC + etich : IC+1) → IC | {Ri, Rj} | |
| IF= Ri, Rj, etich | (Reg[i]=Reg[j] ? IC + etich : IC+1) → IC | {Ri, Rj} | |
| IF> Ri, Rj, etich | (Reg[i]>Reg[j] ? IC + etich : IC+1) → IC | {Ri, Rj} | |
| IF<= Ri, Rj, etich | (Reg[i]<=Reg[j] ? IC + etich : IC+1) → IC | {Ri, Rj} | |
| IF>= Ri, Rj, etich | (Reg[i]>=Reg[j] ? IC + etich : IC+1) → IC | {Ri, Rj} | |
| IF<0 Ri, Rj, etich | (Reg[i]<0 ? IC + etich : IC+1) → IC | {Ri} | |
| IF=0 Ri, Rj, etich | (Reg[i]=0 ? IC + etich : IC+1) → IC | {Ri} | |
| IF>0 Ri, Rj, etich | (Reg[i]>0 ? IC + etich : IC+1) → IC | {Ri} | |
| IF<=0 Ri, Rj, etich | (Reg[i]<=0 ? IC + etich : IC+1) → IC | {Ri} | |
| IF>=0 Ri, Rj, etich | (Reg[i]>=0 ? IC + etich : IC+1) → IC | {Ri} | |

## “Pseudo” istruzioni

| Istruzione | Compilata come | Registri letti | Registri scritti |
|---|---|---|---|
| CLEAR Ri | ADD R0, R0, Ri | | {Ri} |
| MOV Ri, Rj | ADD Ri, R0, Rj | {Ri} | {Rj} |

## Istruzioni speciali

| Istruzione | Compilata come | Registri letti | Registri scritti |
|---|---|---|---|
| START_PROC Ra, Rb | Ra inviato a MMU, Reg[b] → IC | {Ra, Rb} | |
| NOP | Non fa nulla | | |
| MASKINT Ra | Utilizza Ra come registro maschera delle interruzioni | {Ra} | |
| DI | Disabilita interruzioni | | |
| EI | Abilita interruzioni | | |
| COPY_IC Ra | IC → Reg[a] | | {Ra} |
| END | Fine programma | | |