odmik od trenutnega naslova, tj. dolžina skoka. Tudi ta ukaz lahko preoblikujemo tako, da postane odvisen od določenega pogoja:

```
JR cc, dis .
```

cc je pogoj, ki mora biti izpolnjen. Pri dolgih skokih lahko uporabimo osem različnih pogojev, pri relativnih pa le štiri: Z, NZ, C ali NC.

Vrednost odmika (dis) se *prišteje* programskemu števcu (ga torej ne nadomesti, tako kot pri dolgem skoku). Pri odmiku CP vedno upošteva pravilo "iztegnjen palec - negativno število". Zato je odmik lahko pozitiven (skok naprej) ali negativen (skok nazaj) in ima lahko le vrednosti od -128 do +127.

Pozor ! Ko CP začne izvajati relativni skok, programski števec že kaže na ukaz, ki neposredno sledi skoku. Kadar namreč CP pride do ukaza JR, ve da ta ukaz zaseda dva zloga. Zato poveča programski števec za 2 - tako da ta kaže na ukaz za skokom. Poglejmo primer

```
naslov      ukaz
 ...         ....
30000       ADD A, B
30001       JR Z, 02H
30003       LD B, 0
30005       LD HL, 4000H
 ...         ....
```

CP izvaja ta del programa takole:

1. korak Poglej vsebino pomnilniške celice na naslovu 30000. Ker je to en zlog dolg ukaz, povečaj programski števec na 30001. Izpolni ukaz.
2. korak Poglej vsebino celice na naslovu 30001. Ker je to del ukaza, dolgega dva zloga, povečaj programski števec na 30003, poglej še vsebino naslednje celice in izpolni ukaz.

Tu se mora CP odločiti, kaj naj stori s programskim števcem:
- če je ničelna zastavica dvignjena, prišteje k programskemu števcu 2 in tako skoči na naslov 30005;
- če je ničelna zastavica spuščena, ne spreminja programskega števca in nadaljuje z naslednjim ukazom (na naslovu 30003).

Glede na stanje zastavice se bo izvršil ali ukaz na naslovu 30005 ali na 30003.

Relativni skok je lahko usmerjen tudi nazaj. V tem primeru je odmik negativna vrednost.

OREH: Relativni skok zaseda dva zloga, programski števec pa kaže na ukaz, ki mu neposredno sledi. Kakšen bi bil torej učinek ukaza JR FEH (= JR -2) ?

ZANKE V STROJNEM JEZIKU

Iz basica dobro poznamo "FOR ... NEXT" zanke:

```
FOR I = 1 TO 6
LET C = C + 1
NEXT I
```

Premislimo, kako bi v strojnem jeziku napisali takšno zanko, če bi uporabili aritmetične operacije in relativne skoke:

```
        LD B, 1         ;postavi števec na 1
        LD A, 7         ;mejna vrednost
Zanka   INC C           ;C = C + 1
        INC B           ;povečaj števec
        CP B            ;je B enak A ?
        JR NZ, Zanka    ;če ni, ponovi zanko
```

Program bo deloval, vendar zaseda tri registre: enega za število, drugega za povečevanje števca in tretjega za shranjevanje mejne vrednosti. Poleg tega ukaz, ki povečuje števec, ne spremeni nobene zastavice, ko je naloga izpolnjena (tj. ko števec doseže mejno vrednost).

Mnogo bolje bi bilo šteti navzdol!

Vemo, da moramo zanko ponoviti šestkrat. Zakaj ne bi dali registru B vrednosti 6 in šteli navzdol ?

```
0606        LD B, 6         ;nastavi števec
0C   Zanka  INC C           ;C = C + 1
05          DEC B           ;zmanjšaj števec
20FC        JR NZ, Zanka    ;če še ni konec, ponovi zanko
C9          RET             ;vrni se
```

To je dosti bolj učinkovit način.