celico nn z A" - bo vpisal vrednost v celico na naslovu nn. Če bo nn naslov celice v ROM-u, bo ukaz seveda brez učinka.

Naslov nn, ki nastopa v teh ukazih, je neko določeno število in ne spremenljivka. Zato moramo že med pisanjem programa vedeti, kateri naslov bomo uporabili. Ukaza torej služita za prenašanje "spremenljivk" - podatkov iz pomnilnika v register A in iz registra A v naključni pomnilnik (RAM). Vzemimo za primer program - igro Pristajanje na Luni. V njem potrebujete podatke o hitrosti svoje vesoljske ladje, o njeni višini in količini goriva. Ob vnašanju programa boste morali določiti, katere pomnilniške celice bodo služile shranjevanju teh vrednosti. Ko bi pisali program, bi torej še lahko napisali: LD A, (gorivo). Pri vnašanju strojnega koda ali uporabi zbirnika pa bi morali nadomestiti "gorivo" z naslovom izbrane pomnilniške celice. Na primer:

30000 = hitrost
30001 = višina
30002 = gorivo

Šele tako bi lahko izvedli del programa, v katerem bi pogledali, koliko je še goriva, to količino zmanjšali in jo zopet spravili.

In kaj če ne vemo natančnega naslova celice, ki vsebuje podatek? Recimo, da lahko le izračunamo, kje bo podatek shranjen ? Ker vsak naslov zasede 16 bitov, moramo izračunano vrednost začasno shraniti ali v enega izmed registrskih parov (BC, DE, HL) ali v enega indeksnih registrov (IX, IY). Prvi način: naslov v registrskem paru, imenujemo registrsko posredno naslavljanje. Ukazi so:

ukaz		opis
LD r, (HL)	-	napolni register z vsebino celice na naslovu HL,
LD A, (BC)	-	napolni A z vsebino celice na naslovu BC ter
LD A, (DE)	-	napolni A z vsebino celice na naslovu DE.

Registrski par HL je prednostni par, podobno kot je register A prednostni enojni register. Kadar shranimo naslov v HL, lahko zato prenašamo podatke v katerikoli register, celo v H ali v L, čeprav izgleda čudno. Kadar pa uporabljamo BC ali DE, lahko podatke nalagamo le v register A. Tudi pri teh ukazih srečujemo somernost: na podoben način shranjujemo podatke v pomnilnik:


```
LD (HL), A
LD (BC), A
LD (DE), A.
```

Pri drugem načinu posrednega naslavljanja shranimo naslov v enega od indeksnih registrov. To imenujemo indeksno naslavljanje. Indeksna registra IX in IY lahko uporabimo kot kazalca za cele skupine podatkov. Na kratko zapišemo to takole

```
LD r, (IX +dis)
LD r, (IY +dis)
```

Znak r spet označuje katerikoli 8-bitni register (razen F), dis pa pomeni odmik od naslova, ki ga vsebujeta IX oziroma IY. Tako lahko shranimo npr. 1.podatek neke skupine (dis = 1), 10. podatek (dis = 10), 137. podatek (dis = 137) in tako naprej. Odmik dis je 8-bitno število, ki ne more biti spremenljivka, zato ga moramo določiti med programiranjem. To je pomanjkljivost teh ukazov in jih običajno rabimo za branje ali pisanje tabel ali seznamov podatkov. Na voljo sta tudi obratna niza ukazov

```
LD (IX +dis), r
LD (IY +dis), r
```

Indeksni način naslavljanja je nekoliko zapleten in zato manj pogosto rabljen.

Izmed vseh skupin so najhitrejši (4 - 7 T stanj) in najkrajši (1 zlog) ukazi registrskega ter registrskega posrednega naslavljanja. Ukazi ostalih dveh skupin so daljši (3 - 4 zloge) in precej počasnejši (16 - 20 T stanj).

Z80 omogoča sestavljanje nekaterih opisanih načinov naslavljanja, npr. sprotnega (tj., da določite število) in zunanjega (tj., da s pomočjo registrskega para določite naslov). Tako dobite - kdo bi si mislil? - sprotno zunanje naslavljanje. Naslov lahko žal določate le s parom HL. Kratki zapis ukaza je:

```
LD (HL), n.
```

Kljub omejitvi je ukaz zelo uporaben, ker je shranjevanje neposredno, brez uporabe 8-bitnega registra.