Miért éppen 244 opkódot értelmezett a 8080 mikroprocesszor?

Dr. Madarász László okl. villamosmérnök, madarasz@3lan.hu

A mikroprocesszorok történetének első fejezete (a PMOS technológiájú processzorok időszaka) az Intel 4 bites áramköreinek megjelenésével kezdődött (1971-ben), és a nyolcbites 8008-cal zárult (még abban az évben, decemberben). A mikroprocesszoros fejlesztés menedzsere *Marcian (Ted) Hoff* volt **(1. kép),** a munka legnagyobb részét *Stanley Mazor* **(2. kép)** és *Federico Faggin* **(3. kép)** végezte el. Mivel az első mikroprocesszort egy japán cég (Busicom) által megrendelt asztali számológéphez dolgozták ki, ennek a cégnek egy mérnöke (*Masatoshi Shima*, **4. kép**) is ott szorgoskodott a fejlesztők körül. Később Shima belépett az Intelhez.

A második szakaszt is az Intel nyitotta meg, az 1973-ban kifejlesztett, 1974-ben piacra került NMOS 8080 áramkörével **(5. kép).** Sokan ezt tekintik az első valódi mikroprocesszornak (a korábbi áramkörök többé-kevésbé célprocesszorok voltak). Az új processzor nemcsak a felhasználók fantáziáját mozgatta meg, de a mikroprocesszor-gyártást is alaposan fellendítette, hamarosan sok további félvezetőgyár is bejelentette saját áramkörét. Ezeknek a nyolcbites mikroprocesszoroknak a tervezői mind az Intel mérnökgárdája által megkezdett úton haladtak, így alakult ki a '70-es évek közepén a „nagy nyolcbites mikroprocesszor-nemzedék". Ennek legismertebb tagjai az i8080 mellett a Motorola 6800, Zilog Z80, MOS Technology 6502 majd az Intel új eszköze, a 8085.

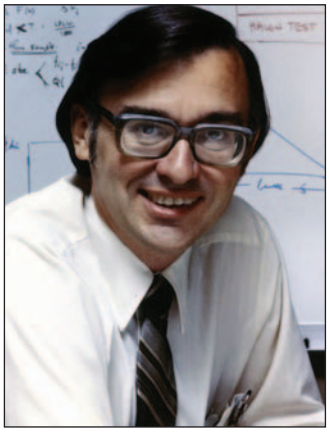
2. kép

A 8080 utasításkészletének sok érdekes sajátossága van, ezek egyike az, hogy minden utasításkód, opkód nyocbites, azaz egy bájtos. Maguk az utasítások lehetnek egy, két vagy három bájt hosszúságúak, de minden esetben csak az első 8 bit képezi az opkódot.

Nyolc biten pedig 256 különböző bitkombinációt lehet képezni, azaz a 8080-nak 256 féle utasításkódja lehetne. Lehetne, mert csak 244 opkódot értelmez. 12 lehetséges opkód kihasználatlan maradt, ezek (hexadecimális alakban) a következők:

08, 10, 18, 20, 28, 30, 38, CB, D9, DD, ED, FD.

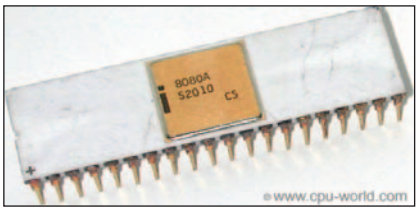
A 12 kihasználatlan opkód a későbbiekben több esetben kedvező volt a fejlesztők számára. A rövidesen kifejlesztett Zilog Z80 mikroprocesszor **(6. kép)** az i8080 minden utasítását ismeri és végrehajtja, mégpedig az Intel processzorral azonos opkódok alapján. Ugyanakkor nagyszámú további utasításra van szüksége, mert sok új belső áramköri részlete és működési módja van. Mivel 100-nál jóval több új utasítást tudnak a Z80 processzorok értelmezni, itt az utasításkészlet már nem lehetett egybájtos. A 8080 által nem használt kódok közül nyolcat új utasítások opkódjaként használtak fel (08, 10, 18, 20, 28, 30, 38, D9), négyet azonban különleges szereppel ruháztak fel. A CB, DD, ED és FD kód azt jelenti, hogy az adott utasítás opkódja két bájtból áll! Az egymást követő DDCB illetve FDCB bájt-párosok pedig hárombájtos utasításkódot jeleznek. Így szinte végtelen sok új utasítást lehetett kialakítani.



1. kép

3. kép

4. kép



5. ábra

Amikor a Zilog megjelent az igen hatékony Z80-CPU processzorral, az Intel is elhatározta, hogy egy újszerű áramkört készít, amivel biztosítja a piaci elsőséget magának. Ekkor tervezték meg a 8085 mikroprocesszort **(7. kép).** Ez az áramkör minden olyan részletet tartalmaz, amit a 8080-ban megtalálhatunk, ezért a 8080 minden utasítását kezeli, azonos opkódok alapján. Az új áramköri részletek kezelésére mindössze két új utasításra volt szükség, ezekhez a 20 és a 30 hexadecimális kódértékeket rendelték hozzá. (A 8085 „titokban" a további opkódokat is használja, vannak „nem dokumentált" utasításai, ezekről a *Rádiótechnika 2015-ös Évkönyvében* olvashatunk további érdekességeket [1].)

Ha a 8080 fejlesztői képesek lettek volna mindezt előre látni, természetes lett volna, hogy a 12 opkódot szabadon hagyták. De az is lehetne a magyarázat, hogy egyszerűen bíztak az áramkörük későbbi továbbfejlesztésében. A valóság azonban más, a szerző véleménye szerint érdekesebb, emberibb. A 8080 fejlesztésében szereplő mérnökök későbbi visszaemlékezéseiből, beszámolóiból megismerhetjük a valódi okot.

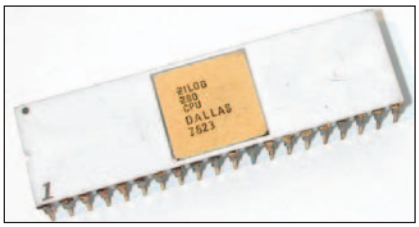
Az i8080 utasításkészletének méretéről még egy adat közismert: a mikroprocesszornak 74 különböző utasítása van. Hogyan használ el 74 utasítás 244 opkódot? Rögtön eláruljuk!

Aki fejlesztői tevékenységet folytat, ismeri azt a helyzetet, hogy a munka során a megrendelő, látva a megoldott részleteket, újabb és újabb további ötletekkel, igényekkel lép fel. Talán ennél is veszélyesebb, ha maga a fejlesztő mérnök akar újabb és újabb csodálatos képességeket beépíteni a keze alatt alakuló eszközbe, áramkörbe. A 8080 esetében ez az utóbbi történt.

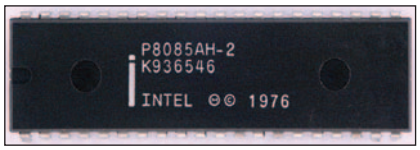
Az elképzelt utasítások rövid leírását, opkódját a tervezés elején rögzítették, majd nekiálltak az egyes utasítások részletes működésének, a végrehajtás elemi lépéseinek a kidolgozásához. Kövessük végig egy utasítás tervezésének lépéseit! Érdemes megfigyelni a kialakításának menetét. Az egyik legelső, legegyszerűbb működésű utasítás a MOV r1,r2 volt a tervezés során, amelyik egy nyolcbites adatot a jelenlegi helyéről (egy adatregiszterből, r2-ből) átmásol egy új helyre (az r1 adatregiszterbe), miközben az eredeti helyén is megmarad (így az utasítást végrehajtva az r1 és az r2 tartalma azonos, az r1 korábbi tartalma pedig elveszett). Ez egy meglehetősen egyszerű működés, a 8080 belső regiszterei lehetnek a kiindulási (forrás, Source) és a befogadó (cél, Destination) elemek. A mikroprocesszorban hét adatregisztert alakítottak ki (A, B, C, D, E, H, L), ezeket kezelheti ez az utasítás, pl. MOV D,A. A forrást három biten lehet kiválasztani, a célt hasonlóképpen. De így a regiszter kiválasztására szolgáló három bit egyik kombinációja nincs kihasználva, hiszen három biten összesen nyolcféle lehetséges bitelrendezést lehet képezni (000 … 111). Adódott tehát, hogy még egy adattárolási lehetőséget is fel tud dolgozni ez az utasítás. Ez a további adattárolási hely pedig a mikroprocesszor által kezelt külső memória, mely 16 biten címezhető, így 64 Ki méretű. A címet az utasítás használata előtt a H,L regiszterpárban kell elhelyezni, ez a regiszterpár más esetben is tárol memóriacímet.

Ezek után az utasítás lehetséges változatainak opkódja a következőképpen alakul:

01DDDSSS, ahol DDD a cél, azaz az r1, az SSS pedig a forrás (r2) kiválasztására szolgáló három



6. ábra



7. ábra

bit. Így a MOV utasításnak a lehetséges opkódjai (binárisan) a következők: 01000000 … 01111111

Ezek között vannak olyanok, amiket „feleslegesnek" vélhetünk, pl. a MOV A,A utasítás, amelyik az A tartalmát önmagába másolja, de meghagyták ezeket a lehetőségeket is. Egyetlen opkód nem használható a lehetséges 64 közül (hiszen a DDDSSS hatbites részlet 64 féle bitkombinációt alkothatna).

A MOV utasítások, ha belső regiszterekre hivatkoznak, nagyon gyorsan végrehajthatók. Az utasítás beolvasását követő dekódolás után a mikroprocesszoron belül megtörténik az adatmozgatás, és más feladat nincs is (mindezt egyetlen „gépi ciklus"-ban elvégzi a processzor).

Más a helyzet, ha a forrás vagy a cél a külső memória rekesze. A H,L-ben lévő címet ki kell küldeni a címbuszon és az adatbuszon meg kell valósítani az adatmozgatást, mindez egy második gépi ciklust igényel. Ha tehát a forrás vagy a cél a külső memória egy rekesze, az utasítás végrehajtása lelassul, két gépi ciklust vesz igénybe.

A MOV A,A-hoz hasonló MOV M,M (ahol M utal arra, hogy a H,L által címzett memóriarekeszt kell használni) már három gépi ciklust igényelne (egy az utasítás beolvasása, egy a külső memória olvasása, egy pedig az írása), és mint tudjuk, ez a rendkívül hosszú idő alatt végrehajtható utasítás teljesen felesleges, mert az eredeti helyére pakolja az adatot. Ezért ezt az utasítás-kombinációt nem engedi meg a 8080 utasításkészlete. Így a MOV utasítás egymaga 63 opkódot használ el.

Érdemes még egy kicsit elidőzni a nem létező MOV M,M utasításnál! Annak a hárombites jelzése, hogy a cél vagy a forrás a külső memóriában van: 110. A MOV M,M opkódja ezért binárisan így alakulna: 01110110, aminek a hexadecimális megfelelője 76. Ez az