Az MBR-től a GPT-ig

| Tartalom |

Bevezetés

Egyetlen PC-ben lévő merevlemezt sem vehetünk használatba anélkül, hogy az ne lenne particionálva. A particionálásról szinte minden programozással foglalkoz(gat)ó ember hallott már, de valószínűleg sokkal kevesebben vannak, akik mélyrehatóbban is tudják, hogyan is szerveződnek a háttértárolókon lévő partíciók és mi az az MBR, vagyis mester boot rekord. Ez a cikk azok számára készült, akik rendelkeznek alapfokú assembly és számítógépes ismeretekkel és szeretnének legalább egy olvasás erejéig többet megtudni ezekről a dolgokról. (Amiket mindig szerettek volna megkérdezni, de sosem mertek.) Példákat leginkább a Microsoft operációs rendszereiből hozok, bár az alapelvek minden más rendszer esetén igazak. Ahol másképp nem jelzem, az informatikai mértékegységeknél (KB, MB, GB, stb.) mindig 1024 a váltószám. A merevlemezgyártók üzleti megfontolásokból 1000-et használnak, ezért az iparban a 90-es évek végén megpróbálták bevezetni a KiB, MiB, stb. mértékegységeket (ezeknél rögzítve az 1024-et váltószámként), de én itt csakazértis maradok a régieknél, mert ezek a *mebibájt, gibibájt* elnevezések még viccnek is rosszak.

1983 márciusában jelent meg az IBM PC DOS 2.0, ami az újdonsült IBM Personal Computer XT-ben lévő 10 MB-os merevlemezeket támogatta. Ez még a FAT12 fájlrendszert használta, de immár megjelent az MBR és ezzel a particionált média.

A számítógépes adathordozókon az adatok tárolására használt legkisebb egység az operációs rendszer és a BIOS szemszögéből a szektor. Egy szektorban hagyományosan 512 bájt adatot lehet tárolni, bár léteznek nagyobb szektorméretet használó eszközök is. A mester boot rekord (MBR) egy particionált PC-kompatibilis tárolóeszköz legelső, speciális használatú szektora. Ez tartalmazza azokat az információkat, amik megmutatják, hogy az adott médiumon hogyan vannak felépítve a fájlrendszereket tartalmazó partíciók és van benne egy végrehajtható kód is, ami a telepített operációs rendszer betöltésére szolgál; általában arra, hogy átadja a vezérlést a rendszerbetöltő második fokozatának, ami legtöbbször egy partíció kötet boot rekordjában van (VBR - volume boot record; a partíció legelső szektora).

Az MBR eredeti változatát az IBM dolgozója, David Litton írta 1982 júniusában. A partíciós tábla ennél legfeljebb négy elsődleges (primary) partíciót támogatott, ezek közül a DOS egyet tudott használni. Ez nem változott akkor sem, amikor a DOS 3.0-ban megjelent a FAT16. Kiterjesztett partíció támogatása a DOS 3.2-ben jelent meg, ez egy speciális típusú partíció ami egyéb partíciókat tartalmaz. A kiterjesztett partíciókon belüli beágyazott logikai meghajtók pedig a DOS 3.30-cal jöttek világra. Mivel az MS-DOS, a PC DOS, az OS/2 és a Windows soha nem tette lehetővé, hogy ezekről bootolni lehessen, az MBR formátuma és a boot kód majdnem változatlan funkcionalitással megmaradt a DOS és OS/2 érában 1996-ig, kivéve néhány külső gyártó implementációját.

1996-ban a Windows 95B és a DOS 7.10 alkalmazni kezdte a logikai blokkcímzést (LBA), hogy használni lehessen a 8 GB-nál nagyobb lemezeket is. Itt jelentek meg a lemez időbélyegek (timestamps), amiknek sajnos nem tudni a pontos funkcióját. Időközben felmerült,

hogy az MBR-nek operációs rendszer- és fájlrendszerfüggetlennek kellene lennie, de ezt a szabályt azért az MBR újabb Microsoft implementációi megszegik, mert ott a CHS elérés a FAT16B és a 06B/0BH típusú FAT32 rendszerekhez van kikényszerítve, az LBA pedig a 0EH/0CH esetén használatos. (A partíció típusokról később.)

Az MBR formátumot néhány lényegi tulajdonságának gyakran szegényes dokumentációja ellenére (amik szeretnek kompatibilitási problémákat okozni) mégis széles körűen, gyakorlatilag ipari szabványként használják. Olyannyira, hogy egyéb platformokhoz való operációs rendszerek is támogatják, akár már más, létező keresztplatformos bootolási és particionálási szabványok mellett.

Az MBR partíciók méretét adó bejegyzések 32 bitesek. Így az 512 bájtos szektorokat használó lemezeken a legnagyobb partícióméret az MBR particionálási módszerrel (nem szabványos eljárások nélkül) 2 TB-ra van korlátozva. Az MBR-ben pedig a partíciós tábla szervezése egy 512 bájtos szektorokkal rendelkező lemez címezhető tárolóterületét 3,99 TB-ra adja meg ($2^{32} * 512 + 2^{32} * 512$ bájt). Emiatt aztán a 2010 óta egyre inkább terjedő nagyobb lemezekhez másféle particionálási megoldás szükséges. Ilyen például az újabb PC-kben az MBR-t felváltó GUID partíciós tábla (GPT). Emellett többféle hibrid MBR-t is kitaláltak egyéb gyártók, hogy párhuzamosan fenntartsák a tárolók első fizikai 2 TB-jában lévő MBR partíciókat és/vagy lehetővé tegyék a régebbi operációs rendszereknek, hogy GPT partíciók mellé is bootolhassanak. Ezek a nem szabványos megoldások azonban kompatibilitási problémát okozhatnak.

MBR-ek nincsenek a nemparticionált tárolókon, mint például a kislemezeken (floppy) vagy azokhoz hasonlóan viselkedő eszközökön. A lemez legelső szektora viszont ezeknél is tartalmazhat boot rekordot betöltővel az esetlegesen lemezen lévő operációs rendszerhez.

Összefoglalva tehát az MBR egy merevlemez (vagy SSD) nulladik szektorának 512 vagy több bájtjában helyezkedik el. Tartalmazhat

- egy vagy több partíciós táblát, ami leírja a tárolóeszköz partícióit. Ebben a felfogásban a bootszektor partíciós szektorként is értelmezhető.
- betöltő kódot, ami meghatározza a bekonfigurált bootolható partíciót, aztán egy láncolt betöltés folyamán betölti és végrehajtja annak kötet boot rekordját (VBR).
- opcionális 32 bites lemez időbélyeget
- opcionális lemez aláírást

Láncolt betöltés (chain loading)

A láncolt betöltés olyan programfuttatási módszer, aminek során az aktuálisan futó programot lecserélik egy újjal, közös adatterületeket használva arra, hogy információt adjanak át az aktuális programtól az újnak. Hasonló az átfedésekhez, de attól eltérően a láncolt betöltés teljesen lecseréli az aktuálisan futó programot, míg az átfedés (overlay) csak a futó program egy részét. Az operációs rendszerek boot menedzsereinél a láncolt betöltés van arra használva, hogy átadja a vezérlést a boot menedzsertől egy bootszektornak. A kívánt bootszektor háttértárolóról töltődik be, lecseréli a memóriában lévő korábbi bootszektort, amiben a korábbi boot menedzser foglalt helyet és hajtódott végre.

Az `fdisk` eszközt az IBM PC DOS 2.0 vezette be az MBR partíciók beállítására és karbantartására. Amikor egy tárolóegységet ezzel partícionálnak, az MBR tartalmazza a partíciós táblát, ami egybefüggő területek, vagyis partíciók helyét, méretét és egyéb attribútumait tartalmazza. A partíciók maguk is tartalmazhatnak olyan adatokat, amik leírnak összetettebb particionálási rendszereket, mint például

kiterjesztett boot rekordok (EBR), BSD disklabelek vagy Logical Disk Manager metaadat partíciókat. Az MBR nem egy partíción van, hanem az eszköz első szektorában (0. fizikai offszet) csücsül, ami megelőzi az első partíciót. (Egy nem particionált eszközön vagy egy adott partíción belüli boot szektort kötet boot rekordnak (VBR) hívnak.) Azokban az esetekben, amikor a gép DDO BIOS kiterjesztést vagy boot managert használ, a partíciós táblát más fizikai helyre is el lehet mozgatni.

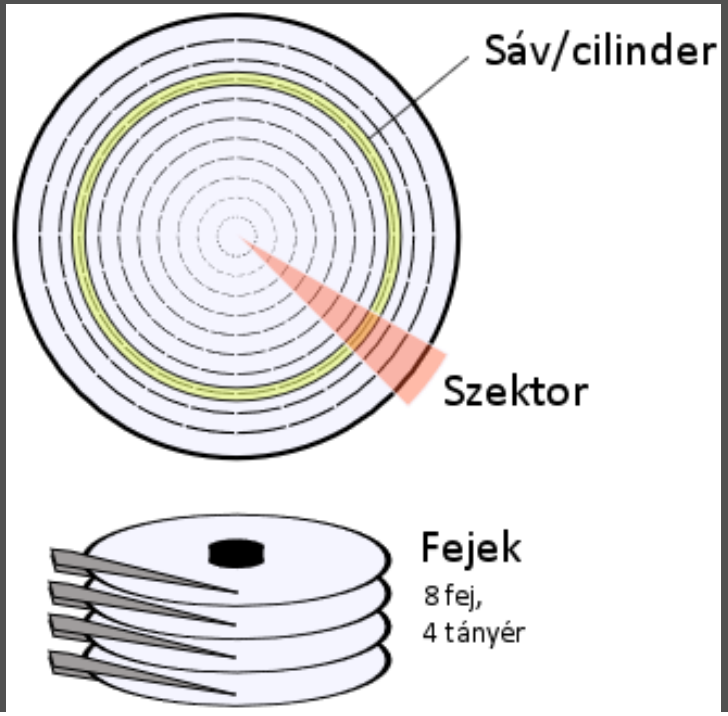
Blokkcímzés

Cilinder, fej, szektor

A CHS (Cylinder, Head, Sector; cilinder, fej, szektor) az egyik legrégebbi módszere egy merevlemezen (illetve a floppy lemezeken) lévő egyes fizikai blokkok megcímzésének. Bár a CHS értékeknek már régóta semmi köze nincs a modern merevlemezeken tárolt adat fizikai elhelyezkedéséhez (a floppy lemezek esete kivétel, ott végig megmaradt ez a kapcsolat), a virtuális CHS értékeket (amiket a meghajtó hardvere vagy szoftvere fordít le valódi értékekre) még továbbra is sok segédprogram és fájlrendszer használja. Ez a módszer a lemezen lévő egyedi szektorokat úgy címzi, hogy előbb meghatározza a cilinder és fej értékek segítségével annak a sávját. Ennek a rövidítésnek a kifejezései gyakorlatilag a nagyobb mennyiségtől a kisebb felé haladva adnak egyre kisebb mennyiségket (a lemez címzésekor a szektor a legkisebb egység). A lemezvezérlők később bevezették a címszámítás fogalmát, ahol a szoftver által adott logikai pozíciókat fizikaiakká fordítják le. Erre azért volt szügkség, mert a zónázott rögzítés kevesebb szektort rögzít a rövidebb (belső) sávokban, a fizikai lemezes formátumok pedig nem szükségszerűen cilinderesek és egy sávon lévő szektorszámok lehetnek aszimmetrikusak is.

Története

A CHS formátumot IBM mainframe-eken már a 60-as években használták a Count Key Data (CKD) merevlemezek. Az akkori formátum hasonló volt a PC-ken használatos CHS-hez, kivéve, hogy a szektorméret nem volt rögzített, hanem sávról sávra változott minden egyes alkalmazás igényeinek megfelelően. Manapság már a tárolóeszköz firmverje emulálja a mainframe-eknek mutatott lemezes geometriát is, az már nincs kapcsolatban a fizikai lemez felépítésével. A PC-kben használt korai merevlemezek (MFM és RLL meghajtók) mindegyik cilindert azonos számú szektorra osztották, úgyhogy a CHS értékek megfeleltek a meghajtó fizikai tulajdonságainak. A CHS értékeket ún. értékhármasok adják meg. Egy ilyen régi meghajtó esetén a hármas lehetett például 500 4 32, ekkor 500 sávja volt minden lemezoldalon, két lemeze volt (4 fej) és 32 szektor volt sávonként, így 32 768 000 bájtot tudott tárolni (31,25 MB).



Az ATA/IDE meghajtók lecserélték az elavult MFM és RLL meghajtókat és sokkal gazdaságosabban tárolták az adatot: zóna bit rögzítést (ZBR) használnak, ahol a sávonkénti szektorok száma a sáv lemezoldalon belüli elhelyezkedésétől függ. A lemeztányér szélén nagyobb hely van, tehát oda több sáv fér, mint a lemez közepe felé. Ezáltal a CHS címzés az ilyen meghajtók esetén a lemeztányér különböző területein elhelyezkedő különböző számú szektor miatt már nem volt teljesen szinkronban a lemez fizikai felépítésével. Emiatt aztán nagyon sok meghajtónak mindig van néhány

fennmaradó szektora a meghajtó végén (kevesebb mint egy cilindernyi), mivel a szektorok összessége ritkán végződik cilinderhatáron (ha egyáltalán). Az ATA/IDE meghajtók a BIOS-ban bármilyen cilinder, fej és szektorszámmal beállíthatóak, ha azok nem lépik túl a lemez kapacitását (vagy a BIOS-ét), mert a meghajtó bármilyen CHS értéket át fog konvertálni saját hardveres specifikációjának megfelelően. Ez persze okozhat kompatibilitási problémákat.

Az olyan operációs rendszereknél, mint az MS-DOS vagy a Windows régebbi verziói minden partíciónak cilinderhatáron kell kezdődnie és végződnie. Néhány modern operációs rendszer (mint például a Windows XP) már figyelmen kívül hagyja ezt a szabályt, bár ez dual-boot esetén szintén lehet kompatibilitási probléma oka. A Microsoft mindazonáltal a Windows Vista óta már nem követi ezt a szabályt a saját lemezes particionáló eszközeinél sem. (Arról nem is beszélve, hogy SSD esetén az egész CHS-bulinak már aztán tényleg semmi értelme sincs.)

Szektorok

A floppy lemezek és lemezes vezérlők 128, 256, 512 vagy 1024 bájtos fizikai szektorméretet használnak, ezek közül az 512 bájtos szektorméret vált dominánssá a 80-as években. Merevlemezek esetén manapság a leggyakoribb szektorméret 512 bájt, de vannak használatban nem-IBM kompatibilis gépekben 520 bájtos szektorméretű lemezek is. Egyes Seagate meghajtók 2005-ben 1024 bájtos szekorméretet kezdtek használni, 2009 vége óta pedig az Advanced Format meghajtók (ebben a Western Digital volt az első) már 4096 bájos fizikai szektorméretet használnak, de képesek emulálni az 512 bájtos szektorokat is. (Ha közelebbről megvizsgálnánk a szektorokat, azt látnánk, hogy a lemez tulajdonképpen nagy mennyiségű adatot tárol a szektorok között is. Ez a sok többletbájt arra való, hogy a lemez firmverje észrevegyen és javítani tudjon alacsony szintű hibákat. Ahogy a lemezek kapacitása növekedett, egyre nagyobb és nagyobb mennyiségű információt tároltak egy négyzetcentiméteren és ez egyre több alacsony szintű hibát eredményezett, ami jócskán megdolgoztatja a meghajtó firmverjének hibajavítási képességeit. Ennek egyik ellenszere az, ha növelik a szektorméretet 512 bájtról valami nagyobbra, mert így sokkal hatékonyabb hibajavítási algoritmusokat lehet használni. Ezek az algoritmusok bájtonként kevesebb adatot használnak a komoly hibák javításához, mint azt tették az 512 bájtos szektorok esetén.)

CHS címzésnél a szektorszámok mindig 1-gyel kezdődnek, nincs 0. szektor. Ez zavart okozhat az erőben, mert a logikai szektorcímzési módszerek (például az LBA vagy a DOS-ban a "relatív szektorcímzés") tipikusan 0-val kezdik a számolást. A fizikai lemezgeometriák esetén a maximális szektorszámot a lemez alacsony szintű formátuma határozza meg, de a PC-kompatibilis gépek BIOS-ában a szektorszámot 6 biten tárolják, így 63 a legnagyobb használható érték. Ez a maximális érték jelenleg is használatban van a virtuális CHS geometriáknál. Windows esetén egy partícióról a következő utasítással lehet információkat, például a használt szektorméretet megtudni:

```
fsutil fsinfo ntfsinfo c:
```

Sávok

A sávok a szektorokból álló vékony koncentrikus körök. Egy sáv olvasásához legalább egy fejre szükség van. A lemezek geometriája következtében a sáv és cilinder kifejezések között szoros a kapcsolat. Egy vagy kétoldalas floppy lemezek esetén a sáv általánosan használt kifejezés és több mint két fej esetén cilinderről beszélünk.

Cilinderek

Egy lemezes meghajtó esetén a cilinderek száma pontosan megegyezik a lemez egyik felületének sávjaival. Ez a fogalom egyesíti a

mindegyik lemezoldalon lévő sávokat, a cilindereket függőlegesen adják ki a sávok. Vagyis a 12. sáv a 0. lemezoldalon, a 12. sáv az 1. lemezoldalon, stb. összesen adják a 12. cilindert.

Fejek

A fej írja és olvassa a mágneslemezen lévő adatokat. Egy lemeznek két oldala van, vagyis két felület tartalmazhat adatokat, ezért általában 2 fej van egy lemez esetén: mindkét oldalán egy-egy. A PC kompatibilis BIOS-okban a CHS címzés 8 bitet használ a fej megadására, vagyis elméletben 256 fejet lehet használni. Viszont van egy hiba az MS-DOS és IBM PC DOS minden verziójában (egészen 7.10-ig), ezért ezek az operációs rendszerek képtelenek bebootolni olyan köteteken, ahol 256 fej van megadva. Így aztán minden kompatibilis BIOS csak 255 fejig használ leképezéseket. Ez a történelmi furcsaság hatással van arra, hogy a régi BIOS INT 13H hívás esetén valamint minden régi PC-s DOS esetén a legnagyobb használható lemezméret:

(512 bájt/szektor)*(63 szektor/sáv)*(255 fej)*(1024 cilinder)=8032,5 MB, de tulajdonképpen az (512 bájt/szektor)*63*256*1024=8064 MB hozta magával az ismert 8 GB-os határt. Ebben a rendszerben a 8 GB=8192 MB nem elérhető, mert ehhez sávonként 64 szektorra lenne szükség.

A sávok és cilinderek 0-tól számozódnak, a 0. sáv az első (legkülső) egy lemezen. A régi BIOS kódok 10 bitet használnak a CHS címzésnél, ami 1024 cilindert eredményez. A szektorokhoz használt 6 bit és a fejekhez használt 8 bit hozzáadásával jön ki, hogy a 13H BIOS megszakítás 24 bites címeket támogat. Ha kivonjuk a nem használt 0. szektorértéket, ahol 1024*256 sáv megfelel 128 MB-nak 512 bájtos szektorok esetén, a 8192-128=8064 MB-os határ ismét kijön.

(Az INT 13H a 20. megszakításvektorra rakott BIOS megszakítás hívása. A BIOS ide általában egy valós módú megszakításkezelőt köt be, ami a szektor-alapú CHS címzést használó lemezes eszközök írására és olvasására való szolgáltatásokat nyújt. Az olvasandó szektor mellett a megszakításnak szüksége van annak az eszköznek az azonosítójára is, amiről a hívó olvasni szeretne.)

A CHS címzés tehát 0 0 1-nél kezdődik és a maximális értéke 1023 255 63 vagy 1023 254 63 a 255 fejes esetben. A CHS hármasok valaha megadták a lemez geometriáját, de ahogy a lemezek egyre nagyobb kapacitásúak lettek, a cilinder is logikai értéké vált, amit szabványosítottak 16056 szekrorra (16065=255*63). A 28 bites CHS címzés (EIDE és ATA-2) 8 bitet biztosít a szektorokhoz, de még mindig 1-gyel kezdődően, így már durván 128 GB-os mérethatárt biztosít (130560 MB-ot 512 bájtos szektorok esetén).

Logikai blokkcímzés

A logikai blokkcímzés (LBA) mára elterjedt módja a tárolóeszközökön (főként merevlemezeken) való adatblokkok címzésének. Az LBA egy egyszerű, lineáris címzési módszer, a blokkokat egy egész index címzi, az első az LBA 0. Az LBA tehát csak egyetlen számot használ címzésre és minden lineáris báziscím egyetlen blokkot ad meg. Az IDE szabvány a 22 bites LBA címést még opcionálisként vezette be, ezt 1994-ben az ATA-1 megjelenésével kiterjesztették 28 bitre, aztán 2003-ban az ATA-6-ban 48 bitesre. Az 1996 óta gyártott merevlemezes meghajtók legtöbbje már támogatja a logikai blokkcímzést.

Az LBA-t egyébként először a SCSI-ben vezették be absztrakciós szintként. Míg a meghajtó vezérlője továbbra is CHS módszerrel címezte meg az adatblokkokat, ezt az információt a SCSI eszközmeghajtó, az operációs rendszer, a fájlrendszer kódja vagy az alkalmazások (például adatbázisok) már nem használták még úgy sem, hogy egyébként alacsony szinten elérték a meghajtót. A rendszerhívások immár LBA címeket adtak át a tárolóeszköz meghajtóprogramjának és a legegyszerűbb esetekben (ahol egy kötet egy

fizikai meghajtón foglalt helyet) ezt az LBA címet aztán közvetlenül adták át a meghajtó vezérlőjének. A jelenlegi 48 bites LBA címzés 128 PB táhelyet támogat (petabájt, a terabájtot követő nagyságrend), ez egy darabig még vélhetően elég lesz. A jelenlegi PC-kompatibilis számítógépek BIOS-a támogatja a 64 bites struktúrákat használó LBA címzést, bár a modern operációs rendszerek saját közvetlen lemezelérési módszereket implementálnak és a boot idejét kivéve nem használják a BIOS-t.

CHS-LBA konvertálás

A CHS hármasok a következő képlettel számíthatók át LBA címekké:

```
A=(c*Nf+h)*Ns+(s-1)
```

ahol A az LBA cím, Nf a fejek száma, Ns a szektorok száma sávonként, a c, h és s pedig a CHS cím.

Ez a képlet nem használja a cilinderek számát, viszont a fejek és a sávonkénti szektorok számát igen, mert ugyanaz a CHS hármas különböző logikai szektorszámokat adhat a geometriától függően. Például:

- Egy lemez 1020 16 63 geometriával 1028160 szektorral: a CHS 3 2 1 = LBA 3150 = (3× 16+2)×63
- Egy lemez 1008 4 255 geometriával 1028160 szektorral: a CHS 3 2 1 = LBA 3570 = (3× 4+2)×255
- Egy lemez 64 255 63 geometriával 1028160 szektorral: a CHS 3 2 1 = LBA 48321 = (3×255+2)×63
- Egy lemez 2142 15 32 geometriával 1028160 szektorral: a CHS 3 2 1 = LBA 1504 = (3× 15+2)×32

Segíthet vizualizálni a szektorok besorolását a lineáris LBA modellbe, ha megjegyezzük, hogy:

- az első LBA szektor a 0. sorszámú, míg CHS modellben ez az 1. sorszámú.
- mindegyik fej/sáv összes szektora megszámlálódik, mielőtt tovább mennénk a következő fejre/sávra.
- ugyanannak a cilindernek az összes feje/sávja megszámlálódik, mielőtt tovább mennénk a következő cilinderre.
- a meghajtó külső fele lesz a meghajtó első fele.

Határok

Az elmúlt évtizedek különböző operációs rendszerei és lemezkezelő eszközei a fentebb leírt rendszerben a legváltozatosabb hibákat és inkompatibilitásokat voltak képesek produkálni, de léteztek olyan korlátok, amelyek szinte mindet érintették, hiszen ezek a lemezvezérlő rendszerekben és/vagy a BIOS-okban voltak. Az alábbiak merültek fel a történelem folyamán:

- **ATA specifikáció az IDE lemezekhez (127,5 GB)**: legfeljebb 65536 cilinder, 16 fej és 255 szektor sávonként, 512 bájtos szektorokkal.
- **BIOS INT 13H hívás (8 GB)**: legfeljebb 1024 cilinder, 256 fej, 63 szektor sávonként.
- **A DOS 504 MB-os határa**: a gyakorlatban az ATA specifikáció és a BIOS megszakításhívás korlátai összevonódtak és mivel egyiknek sem lehetett túllépni a korlátain, mindkettőből a kisebb határok adták meg a küszöböt. Ha az előző két eset maximális CHS hármasát összevonjuk a kisebb értékekre, akkor 1024 cilindert, 16 fejet és sávonként 63 szektort tudunk használni. Ennek áthidalására egyes lemezkezelő eszközök kikerülték a BIOS-t és közvetlenül a hardverhez fordultak, ahol visszajött a 8 GB-os korlát.
- **Az 1,9 GB-os korlát**: néhány régi BIOS csak 12 bitet használ a CMOS RAM-ban a cilinderek tárolására. Így aztán azoknál csak 4096 cilinder lehetséges.
- **A 3,2 GB-os korlát**: volt egy hiba a Phoenix 4.03 és 4.04 BIOS-okban, ami miatt a rendszer nem tette lehetővé 3277 MB-nál

nagyobb méret beállítását a meghajtókhoz.

- **A 4 GB-os korlát**: létezett egy egyszerű BIOS átfordítás, amit ECHS-nek hívtak (Extended CHS, de néha "Large disk support"-ként is szerepelt a BIOS-okban). Ez úgy működött, hogy ciklikusan felezték a DOS számára mutatott cilinderértéket és duplázták a fejértéket addig, amíg a cilinderek száma legfeljebb 1024 nem lett. Mivel azonban a DOS és a Windows 95 nem tudott 255 fejnél többet kezelni, ha a meghajtó 16 fejet jelentett a BIOS felé, akkor ez a módszer csak 8192 * 16 * 63 * 512 = 4032 MB kapacitást tett lehetővé. (Úgy, hogy a DOS felé 1024 cilindert, 128 fejet és 63 szektort hazudott sávonként.)
- **A 7,4 GB-os korlát**: egy kicsit okosabb BIOS-ok az előző módszert úgy módosították, hogy előbb a fejek számát 15-nek vették, így 1024, 240, 63 CHS értéket tudtak hazudni.
- **A 8 GB-os korlát**: ha mindent megtettünk, amit csak lehetett, akkor maradt az 1024, 255, 63 geometria mint felső határ. Ez 8032,5 MB-ot tesz lehetővé.
- **A 31,5 GB-os korlát**: egy időben a nagy lemezek 16 fejet, sávonként 63 szektort és 16383 cilindert mutattak a BIOS felé. Sok BIOS a cilinderek értékét úgy számolta ezután ki, hogy a lemez méretét elosztotta 16*63-mal. 31,5 GB-nál nagyobb lemezek esetén ez 65535-nél nagyobb értéket adott és ezt a BIOS már nem tudta kezelni.
- **2 GB-os LBA korlát**: az IDE szabvány eredetileg a 22 bites LBA címzést vezette be.
- **128 GB-os LBA korlát**: 1994-ben az ATA-1 szabvány 28 bitre terjesztette ki az LBA címzést.
- **128 PB-os LBA korlát**: 2003-ban vezette be az ATA-6 szabvány a 48 bites címzést, jelenleg ez van érvényben.

Az MBR felépítése

Alább látható a hagyományos Master Boot Record felépítése, az MS-DOS 3.30 óta:

Cím		Leírás	Méret (bájt)
000H	Betöltő kód területe		446
1BEH	1. partíciós bejegyzés	Partíciós tábla (négy elsődleges partíció)	16
1CEH	2. partíciós bejegyzés		16
1DEH	3. partíciós bejegyzés		16
1EEH	3. partíciós bejegyzés		16
1FEH	55H	Boot rekord aláírás (little endian rendben. Vagyis értéke: AA55H)	2
1FFH	AAH		

Megegyezés alapján az MBR partíciós tábla sémájában pontosan négy elsődleges partíció táblabejegyzése van, bár néhány operációs rendszer és rendszereszköz képes ezt kiterjeszteni ötre (Advanced Active Partitions a PTS-DOS 6.60-nál és a DR-DOS 7.07-nél), nyolcra (AST és NEC MS-DOS 3.x) vagy 16-ra (Ontrack Disk Manager).

Egy 16 bájtos partíciós bejegyzés a következőképpen néz ki (minden több bájtos bejegyzés little-endian rendben):

Offszet	Hossz (bájt)	Leírás
0H	1	Státusz vagy fizikai meghajtó (a 7. bit mutatja, hogy aktív-e (1). Régi MBR-ek csak 80H és 00H értéket fogadnak el)
1H	3	A partíció első abszolút szektorának CHS címe. Az alábbi három sor mutatja a formátumát
	1	Fej érték (8 bit)
	1	A szektor érték a 0.-5. bitekben, a 6. és 7. bitek tartalmazzák a cilinder felső két bitjét (8. és 9.)
	1	A cilinder érték alsó 8 bitje
4H	1	A partíció típusa
5H	3	A partíció utolsó abszolút szektorának CHS címe. Az alábbi három sor mutatja a formátumát
	1	Fej érték (8 bit)
	1	A szektor érték a 0.-5. bitekben, a 6. és 7. bitek tartalmazzák a cilinder felső két bitjét (8. és 9.)
	1	A cilinder érték alsó 8 bitje
8H	4	A partíció első abszolút szektorának logikai címe
CH	4	A partíción lévő szektorok mennyisége

Ha egy szektor CHS címe már túl nagy ahhoz, hogy beférjen a specifikációba, akkor a partíciós táblába az 1023, 254, 63 hármas kerül (a lemezen ezek az értékek: FE FF FF). Mivel a logikai szektorcím és a méret 32 bites értékek, az 512 bájtos szektorokat használó eszközökön a legnagyobb partícióméret és egy partíció lehetséges legutolsó kezdőcíme nem lépheti túl a 2 TB-512 bájtot (2 199 023 255 040 bájt vagyis 4 294 967 295 ($2^{32}-1$) szektor * 512 (2^9) bájt szektoronként). Nem szabad elfelejteni, hogy a lemezen a tárolás little-endian, vagyis a lemezen lévő bájtokban például a 3F 2D 10 00 érték valójában 102D3FH címet takarja.

(Ha valakinek eszébe jut egy 137 GB-os (valójában 127 GB) határ is, az nem a véletlen műve. Attól függetlenül, hogy mit tárol a partíciós tábla, a régi ATA szabványt támogató BIOS-ok csak 28 bitet használnak a szektorok címzésére, ezért nem tudtak ennél nagyobb területeket használni.)

Mivel az MBR-ben tárolt particionálási információ kezdőblokk címet és hossz információt tárol, elméletben lehetséges megadni partíciókat úgy, hogy egy 512 bájtos szektorméretet használó lemezen a teljes lefoglalt terület 4 TB legyen, ha egyet kivéve az összes a 2 TB-os határ alatt helyezkedik el, az utolsó pedig a 2^{32}-1 címen kezdődik és 2^{32}-1 méret van neki beállítva. Így olyan partíció jön létre, aminek az eléréséhez 32 helyett 33 biten szükséges a szektorokat címezni. Gyakorlatban azonban csak néhány olyan LBA-48-at használó

operációs rendszer támogatja ezt, amik belsőleg 64 bites szektorcímeket használnak. Ilyen például a Linux, FreeBSD és a Windows 7. Az MBR kódterületének szigorú megszorítása illetve amiatt, hogy csak 32 bites értékeket támogat, a boot szektorok, még ha engedélyezett is az LBA-48 támogatása, csak 32 bites számításokat használnak, kivéve ha speciálisan arra tervezték őket, hogy támogassák az LBA-48 teljes címezhető tartományát vagy pedig speciálisan 64 bites platformokhoz készültek. Minden olyan boot kód vagy operációs rendszer, ami belsőleg 32 bites szektorcímeket használ, úgy járhat, hogy a címzés ilyen partíciónál átfordul komoly adatvesztést okozva.

Olyan lemezeknél, amik 512 bájttól eltérő szektorméretet használnak, mint például a külső USB meghajtók, szintén vannak korlátozások. 4096 bájtos szektorméret nyolcszoros méretnövekedést okoz az MBR által definiált partíciókban, így lehetővé tesz 16 TB-os (2^{32} * 4096 bájt) méretű partíciókat. A Windows XP utáni verziói valamint a Mac OS X támogatják a nagyobb szektorméretet, akárcsak a 2.6.32 utáni Linux kernelek. Bootloaderekkel, particionáló eszközökkel és BIOS implementációkkal kapcsolatban viszont könnyen lehetnek ilyen esetben problémák, hiszen ezek gyakran beégetve tartalmazzák az 512 bájtos szektorméretet és ezeknek megfelelő buffert, így nagyobb szektorméret esetén nemkívánatos memóriafelülírás léphet fel.

Partíciótípusok

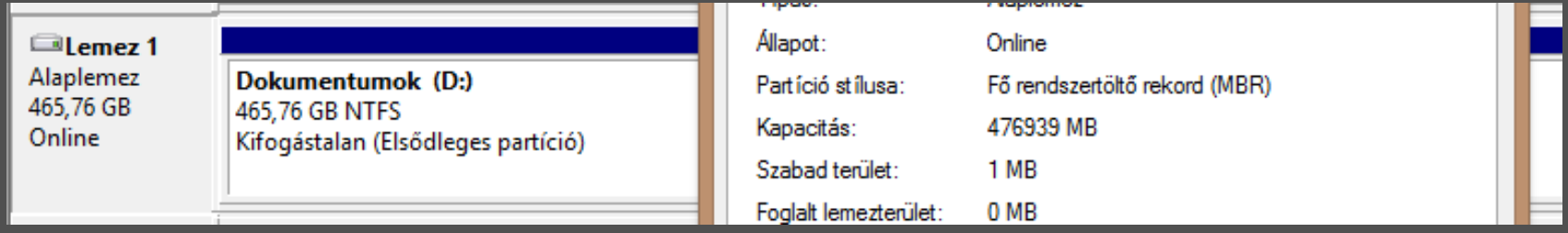
A partíciós bejegyzésben lévő egybájtos partíció típusa érték mutatja meg az adott partíció fájlrendszerét. Ezen belül tartalmazhat speciális jelzőbiteket is, hogy az adott fájlrendszer milyen elérési módszert igényel (CHS, LBA, rejtett partíció, titkosított partíció, stb.)

A partíciótípusok listáját az IBM és a Microsoft eredetileg belsőleg kezelte. Amikor a PC operációs rendszerek és lemezkezelő eszközök piaca virágzásnak indult, egyéb gyártók számára is szükségessé vált egyedi partíciótípus bevezetése. Mivel a Microsoft nem dokumentált minden általuk használt partíciótípust és nem akart kívülről érkező igényeket kezelni, a többi cég egyszerűen elkezdett ötletszerűen és próba-szerencse alapon egyedi típusazonosítókat alkalmazni. Ez persze kompatibilitási problémákhoz vezetett. A 90-es években ezért iparági kezdeményezés indult, hogy ezeket a problémákat legalább lecsökkentsék és létrehozzanak egy mindenki által elfogadott listát. A teljes lista megtalálható itt. Amint látható, ebben már nem nagyon van szabad hely új szereplőknek.

Az érték konkrét kezelése persze az operációs rendszer betöltőjén és kerneljén múlik. Ezek számukra ismeretlen partíciótípusokat úgy kell kezeljenek, mintha fenntartott lemezterület lenne.

Példák

Nézzünk meg most már egy élő példát, mondjuk az én fél terás HDD-m MBR-jének partíciós tábláját!



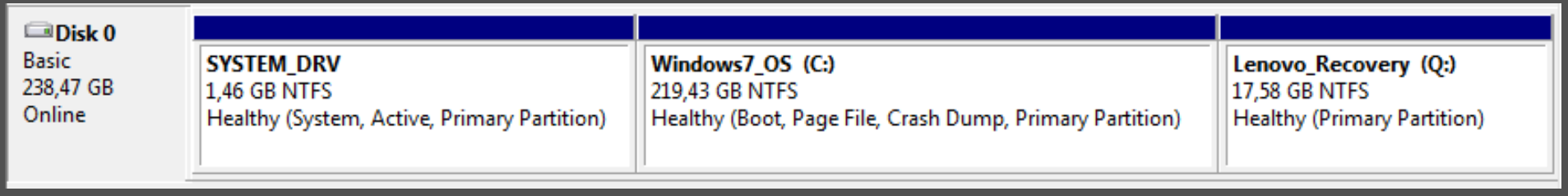
A példa rendkívül egyszerű; a lemezen egyetlen elsődleges NTFS partíció van, ami a teljes szabad területet elfoglalja. Alább látható az MBR partíciós táblájának tartalma.

```
      0|  1|  2|  3|  4|  5|  6|  7|  8|  9|  A|  B|  C|  D|  E|  F|
00   00| 20| 21| 00| 07| FE| FF| FF| 00| 08| 00| 00| 00| 50| 38| 3A|
10   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|
20   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|
30   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|
```

Látható, hogy a nem használt három bejegyzés teljes egészében nullákat tartalmaz. Az első bejegyzés pedig:

- **00H**: 00H az értéke, vagyis nem aktív fizikai meghajtó
- **01H**: 20H 21H és 00H, vagyis a partíció első szektorának CHS hármasa: 0 32 33
- **04H**: 07H az értéke, ez az NTFS partíció típusjele.
- **05H**: FEH FFH és FFH, vagyis ahogy fentebb is olvasható volt, ez a legnagyobb CHS hármas, itt azt jelenti, hogy a partíció mérete túlnyúlik a CHS által ábrázolható tartományon.
- **08H**: a partíció első abszolút szektorának LBA címe: 800H (2048)
- **0CH**: a partíción lévő szektorok mennyisége: 3A385000H (976769024). Mivel a szektorméret 512 bájt, kiszámolható a méret: 465 GB. A figyelmes olvasó észreveheti, hogy a Windows ennél picit többet mutat a képen! A különbség éppen 2048 szektornyi. A lemez méretét helyesen látni, hiszen abba beletartozik ez a terület is a lemez elején (a tulajdonságok panel is ezt mutatja), azonban a partíció méretét már illene helyesen mutatni. Ehelyett ott is a lemez méretét látni (ha a partíció tulajdonságait néznénk meg, az már helyes értéket mutatna).

Ez pedig egy Windows 7 rendszert tartalmazó 244197 MB-os SSD három elsődleges partícióval:




```
      0|  1|  2|  3|  4|  5|  6|  7|  8|  9|  A|  B|  C|  D|  E|  F|
000  80| 20| 21| 00| 07| 38| 39| BF| 00| 08| 00| 00| 00| D8| 2E| 00|
010  00| 38| 3A| BF| 07| FE| FF| FF| 00| E0| 2E| 00| 00| D0| 6D| 1B|
020  00| FE| FF| FF| 07| FE| FF| FF| 00| B0| 9C| 1B| 00| 78| 32| 02|
030  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|
```

- Látszik, hogy csak az első partíció aktív (80H).
- Az első partíció CHS hármasa: 0 32 33, a másodiké 191 58 56, a harmadik már túl van minden határon...
- Mindhárom partíció NTFS típusú: 07H a típusa.
- Az első partíció még befér az ábrázolható tartományba, de a másik kettő már nem.
- Az első partíció a 800H szektorcímen kezdődik és a 2ED800H szektort tartalmaz, vagyis 1499 MB méretű

- A második partíció a 2EE000H szektorcímen kezdődik és a 1B6DD000H szektort tartalmaz, vagyis 219,43 GB méretű
- A harmadik partíció a 1B9CB000H szektorcímen kezdődik és a 2327800H szektort tartalmaz, vagyis 17,58 GB méretű

De hát hol marad a közbeeső 2047 szektor?

A Windows 7 és 8, akárcsak a Vista, ha üres lemezre telepítik, akkor az első partíciót a 2048-as szektorcímen kezdi (a Windows XP a 63.-on, ami 3FH). Ez pontosan egy megabájt végének offszetje a lemezen. Arról, hogy miért hagy ki ennyi szabad helyet, a Microsoft ezt mondja: a fájlrendszer, a kötetkezelő és a Windows Vista rendszerben lévő tárolóverem más részei a nagy szektorméretű merevlemez-meghajtókkal való kompatibilitás biztosítása érdekében frissültek. A Windows korábbi verzióiban a merevlemez-meghajtó első partíciója alapértelmezés szerint a 3FH szektornál kezdődött (kezdeti eltolás). Mivel a kezdeti eltolás mértéke páratlan szám volt, ez a nagy szektorméretű meghajtókon a partíció és a fizikai szektorok helytelen igazítása miatt teljesítményproblémákat okozna. A Windows Vista rendszerben az alapértelmezett kezdeti eltolás értéke a 800H-s szektorra esik. A speciális szektorigazítású meghajtókon a kezdeti eltolás ettől eltérhet. (Visszamenőleges kompatibilitás miatt a legtöbb régi operációs rendszer, mint a DOS, OS/2 és a Vista előtti Windows verziók az MBR partíciók kezdetét mindig sávhatárra, végét pedig cilinderhatárra teszik. (Emulált CHS geometriáknál és LBA-val kezelt partícióknál is.) A kiterjesztett partícióknak pedig mindig cilinderhatáron kell kezdődniük. Ezért kezdődik az első partíció a 63-as LBA címen.)

Azt tudjuk, hogy az Advanced Format meghajtók már 4096 bájtos szektorméretet használnak. Felmerül a kérdés, hogy ha a rossz illeszkedés miatti lehetséges teljesítményproblémák miatt választottak a Vistánál új értéket, minek kellett ekkorát választani, ami 256 ilyen nagy méretű szektor? Miért nem lehetett 32, 64 vagy 128 KB-ra illeszteni? A Microsoft biztos akart lenni, hogy akár még nagyobb szektorméretű lemezekhez is illeszkedni fog a partíció? Ezt már nem tudni, habár én gyanítom azért, mert a partíció méreteket ezek a rendszerek MB-ra kerekítik.

A Windows 7 óta a Windows operációs rendszerek egyébként létrehoznak több partíciót is a rendszer számára, ha üres lemezre telepítjük őket, de azok céljának tárgyalása nem tartozik a partíciós táblák témakörbe.

A betöltő kód

Az IBM PC-kompatibilis számítógépeken a ROM BIOS-ban lévő rendszerindító program tölti be és hajtja végre a master boot rekordot. A PC/XT 5160 Intel 8088 processzort használt. A kompatibilitás miatt minden x86-alapú rendszerben egy ún. valós módnak nevezett üzemmódban indul el a processzor, ami kompatibilis az Intel 8088-cal. A BIOS beolvassa az MBR-t a tárolóeszközről a fizikai memóriába, majd átadja a vezérlést a boot kódra. Mivel a BIOS nem vált üzemmódot a processzoron, ezért valós módban fut, tehát elvárás, hogy az MBR is valós módú programot tartalmazzon. Láthatjuk, hogy milyen pici hely (446 bájt) van az MBR-ben a kód számára, emiatt itt csak nagyon kis program fér el. Ez majd a tárolóeszközről (BIOS-hívások segítségével) további kódot (például boot loadert) fog betölteni. A vezérlés aztán átadódik erre a kódra, ez felel a tulajdonképpeni operációs rendszer betöltéséért. Ezt hívják láncolt betöltésnek.

A széles körben használt MBR programokat arra hozták létre, hogy PC DOS és MS-DOS operációs rendszereket betöltsenek. Ezek megnézik a partíciók listáját az MBR beágyazott partíciós táblájában, megkeresik azt az egyet, ami meg van jelölve aktív jelzőbittel. Ha ez megvan, akkor betöltik és futtatják az aktív partíció kötet boot rekordját.

Persze vannak alternatív boot kód implementációk is, boot menedzserek is telepíthetnek ilyeneket. Van olyan, ami a lemez első sávjáról

további kódot tölt be a boot menedzserhez (ez első sávról feltételezi, hogy "szabad" terület), majd végrehajtja azt. (Persze ezzel olyan tulajdonságra alapoz, ami egyáltalán nem általános.) Olyan is létezik, ami felhasználói beavatkozást igényel; a felhasználó kijelölheti, hogy melyik lemez melyik partíciójáról szeretne bootolni. Ez átadhatja a vezérlést akár másik meghajtó MBR-jének is. Más alternatív MBR kódok tartalmaznak listát egyes lemezes pozíciókról (ezek gyakran megfelelnek egy fájlrendszerben egy fájl tartalmának), majd onnan betöltik a boot menedzser kód további részét. Ilyen esetben persze szükség van rá, hogy a beágyazott lemezpozíciókat frissítse valami, amikor a fájlrendszerben a kód további részének pozíciója változik.

Olyan, merőben nem szabványos megoldások is készültek, amik nem feltétlenül a rendszerindításra szolgálnak. Az MBR-ben fut a világ egyik legkisebb méretű sakkprogramja, a 487 bájtos BootChess, de a legenda szerint készült egy játszható tetris játék is, ami szintén elfér a boot szektorban, sőt ennek egy változata csak akkor tölti be az operációs rendszert, amikor 10 sort sikerül kirakni.

A BIOS-ban a rendszerindító folyamat be fogja tölteni az első érvényes MBR-t a 0000H:7C00H címre. A BIOS kódban az utolsó végrehajtott utastás egy ugrás lesz erre a címre. A legtöbb BIOS esetén a fő ellenőrzés az aláírás megléte a +1FEH offszeten, bár a BIOS fejlesztője kiegészítő ellenőrzéseket is írhat, ha van hozzá kedve. Ellenőrizheti például, hogy az MBR tartalmaz-e érvényes partíciós tábla bejegyzéseket olyan szektorokkal, amik nem hivatkoznak túl a lemez fizikai kapacitásán.

A boot szektor kódja feltételezi, hogy a 0000H:7C00H címen fog kezdődni, de ettől a fizikai címtől kezdődően gyakorlatilag minden memória elérhető valós módban, az első 640 KB végéig. Az INT 12H BIOS megszakítás segíthet meghatározni, hogy mennyi memóriát lehet biztonsággal lefoglalni (alapesetben ez egyszerűen kiolvassa a 0040H:0013H címről a memóriaméretet KB-ban, de persze ez a megszakítás felül lehet írva BIOS-t kiterjesztő pre-boot programok vagy akár vírusok által is, amik lecsökkentik a visszajelzett elérhető memória mennyiségét, hogy megakadályozzák, hogy a betöltési fázisban futó programok, mint a boot szektor felülírják őket).

Az MBR kód kommunikálhat a felhasználóval, megvizsgálhatja a partíciós táblát, de végül mégiscsak el kell végeznie a fő feladatát, vagyis betölteni (általában INT 13H BIOS hívásokkal) azt a programot, ami elvégzi a rendszerbetöltési folyamat következő fázisát. A betöltött operációs rendszer és az MBR kód között egyébként nem feltétlenül szükséges éles elkülönülés; az MBR vagy egy része simán a RAM-ban maradhat és akár a betöltött program használhatja is, miután az MBR átadta neki a vezérlést. Ugyanez igaz a kötet boot rekordra is, függetlenül attól, hogy az a kötet floppy lemezen vagy merevlemezen van-e. A gyakorlatban mindenesetre tipikus, hogy a betöltött program felülírja ezt, tehát egyetlen funkciója a rendszerbetöltési láncolat első láncszemének lenni.

Fontos megjegyezni, hogy egy MBR és egy kötet boot rekordja között csak a BIOS feletti felhasználói program szemszögéből van különbség (itt a felhasználói program jelenthet operációs rendszert is). A BIOS számára a cserélhető és a beépített lemezek között gyakorlatilag nincs különbség. A BIOS mindkét esetben betölti az eszköz első fizikai szektorát a 7C00H abszolút címre, ellenőrzi a két utolsó bájtban lévő aláírást és ha minden rendben lévőnek tűnik, átadja a vezérlést egy JMP utasítással az első bájtjára.

A hagyományos MBR rendszerindító kód tehát betölti majd futtatja a - rendszerbetöltő- vagy operációs rendszer-függő - kötet boot rekordot, ami az "aktív" partíció kezdetén található. Egy hagyományos kötet boot rekord is belefér az 512 bájtos szektorba, de az MBR simán betölthet akár több szektort is, ha szükség van rá. Akárcsak az MBR, a kötet boot rekord is feltételezi, hogy a 0000H:7C00H címre töltötték be. Ez azért van, mert a VBR elképzelése a nem particionált médiából ered, ahol a BIOS közvetlenül a VBR-t töltötte be és ahogyan fentebb már olvasható volt, a BIOS semmiféle különbséget nem tesz a VBR és MBR között. Ezért az MBR egyik első feladata, hogy átmásolja magát a memóriába valahová máshová. Ezt a területet is az MBR határozza meg, de leggyakrabban a 0000H:0600H (az MS-DOS, PC DOS, OS/2 és Windows MBR kódjánál) vagy 0060H:0000H (a legtöbb DR-DOS MBR esetén) címen szokott kezdődni.

(Mégha valós módban mindkét szegmentált cím ugyanarra a fizikai címre fordítódik is le, az Apple Darwin indulásához az MBR-t a 0000H:0600H-ra kell áthelyezni a 0060H:0000H helyett, mert a kód a DS:SI mutatótól függ ugyan, de hibásan csak a 0000H:SI-re hivatkozik.) Ezek a szabványos címek azért is fontosak, mert a VBR-ek feltételeznek némi szabványos memóriaelrendezést, mikor betöltik a boot fájljukat.

A partíciós tábla rekordjának *status* mezője használatos az aktív partíció jelzésére. A szabványnak megfelelő MBR-ek csak egy aktívnak jelölt partíciót engedélyeznek és ezt a feltételt akár az érvényes partíciós tábla ellenőrzéséhez is felhasználhatják. Ha ez nem teljesül, akkor hibaüzenetet dobnak. Néhány nem szabványos MBR azonban ezt nem tekinti hibának, egyszerűen az elsőként aktívnak jelölt partíciót fogja használni.

A 00H (nem aktív) 80H (aktív) értékektől eltérő eseteket hagyományosan érvénytelennek tekintették és a rendszertöltő program hibaüzenetet jelenített meg, ha ilyesmivel találkozott. A Plug and Play BIOS szabvány és a BIOS Boot Specifikáció (BBS) 1994 óta lehetővé teszi, hogy egyéb eszközökről is lehessen bootolni. Ezért az MS-DOS 7.10 bevezetésével (Windows 95B) az MBR elkezdte aktív jelzőbitként kezelni a beállított 7-es bitet és csak a 01H-7FH értékekre adott hibaüzenetet. Mivel továbbra is úgy tekintettek a bejegyzésre, hogy az aktuális fizikai eszközt kell használni a megfelelő partíció VBR-jének betöltéséhez, most már 80H-tól eltérő boot meghajtkat is elfogadtak érvényesnek (a boot meghajtó azonosítójáról a DL regiszterben kap információkat az MBR kódja, erről lásd lejjebb). Az aktuális fizikai meghajtószám partíciós táblában való tárolása általában nem okoz visszamenőleges kompatibilitási problémákat, mert az érték csak az elsőtől eltérő meghajtókon fog eltérni 80H-tól (ami korábban egyébként sem volt bootolható). De még ha a rendszerek lehetővé is tették, hogy bootolni lehessen más eszközökről, a kiterjesztés azért nem működik annyira univerzálisan. Ha például a fizikai meghajtók BIOS-beli elrendezése megváltozik, vagy amikor meghajtókat eltávolítanak, hozzáadtak vagy kicserélnek, az MBR-ben tárolt azonosító már nem egyezik a DL regiszterben kapottal. Így aztán a BIOS Boot Specification (BBS) szerint a legjobb módszer egy modern MBR számára, ami a 7. bitet aktív flagnek fogadja el, hogy továbbadja az eredetileg BIOS által adott DL értéket a partíciós táblában szereplő helyett.

BIOS-MBR kapcsolat

Miután az MBR betöltődött a 0000H:7C00H címre, vezérlésátadáskor a BIOS a következő regisztereket hagyja beállítva:

- **CS:IP = 0000H:7C00H** (rögzített érték) Néhány régi Compaq BIOS hibásan a 07C0H:0000H értéket használta. Bár ez ugyanarra a fizikai memóriaterületre hivatkozik, nem szabványos és kerülendő, mert az MBR kód feltételez bizonyos regiszterértékeket.
- **DL = az a boot eszköz, ahonnan az MBR-t betöltötte a BIOS** (merevlemezek esetén: 80H=első, 81H=második, stb FEH-ig; floppik / szuperfloppik esetén: 00H=első, 01H=második, stb 7EH-ig. A 7FH és FFH értékek ROM és távoli eszközökhöz vannak fenntartva és lemezeken nem szabad használni). Amikor az MBR további szektorokat akar majd betölteni arról az eszközről, ahonnan őt is betöltötték, ezt az értéket a BIOS INT 13H megszakításnak át kell adnia. A tipikusan cserélhető merevlemezként konfigurált USB pendrive-ok DL = 80H, 81H, stb. értékeket kapnak. De néhány ritka BIOS ezeket is hibásan jelöli DL=01H-nak, mintha szuperfloppiknak lennének konfigurálva. (Merevlemezeknek nem csak a gépbe beépített lemezeket tekintjük itt, hanem például a mobil rack-ekbe helyezett HDD-k is ebbe a kategóriába esnek, amik természetesen cserélhetők. Azok is bootolhatók és particionáltak. A szuperfloppik a floppi lemez leváltására tett kísérletek nagyobb kapacitású technikákkal, mint például a Zip Drive 1994-ből 100 MB és afölötti kapacitással. De a Microsoft terminológiájában a CD-ROM és DVD-ROM is szuperfloppinak minősül.)

Egy szabványkövető BIOS 80H-nál nem kisebb értékeket kizárólag rögzített és cserélhető merevlemezekhez társít és tradicionálisan csak 80H és 00H értékek adódtak át a fizikai eszközöknek a boot során. Mivel csak a merevlemezek particionáltak, így az egyetlen DL

érték, amit az MBR láthatott, az a 80H volt. Sok MBR-t úgy kódoltak, hogy ne vegye figyelembe a DL tartalmát és minden esetben beégetett értékkel (általában 80H) dolgozzon. Egyéb eszközök a specifikáció szerint 1994-től váltak bootolhatóvá. Ez már azt ajánlja, hogy az MBR és a VBR kód a DL értékét használja, mert ez még számos nem szabványos elrendezéssel is biztosítja a kompatibilitást (lásd fent). Az El Torito szabványnak megfelelő bootolható CD-ROM-ok például tartalmazhatnak olyan BIOS által felcsatolandó lemezképeket, amik floppiként vagy szuperfloppiként viselkednek. 00H és 01H DL értékeket szintén használ például a megbízható (trusted) módban futó Protected Area Run Time Interface Extension Services (PARTIES) és a Trusted Computing Group (TCG) BIOS kiterjesztés, hogy elérjen egyébként láthatatlan PARTIES partíciókat, lemezkép fájlokat a Boot Engineering Extension-ön (BEER) keresztül a merevlemez Host Protected Area (HPA)-jának utolsó fizikai szektorában. Ha arra tervezték, hogy floppikat vagy szuperfloppikat is emulálni tudjon, az MBR kód elfogadja ezeket a nem szabványos DL értékeket, hogy lehetővé tegye ezeknek a lemezképeknek a használatát particionált médián, legalábbis az operációs rendszerek boot fázisa során.

- **DH 5. bit**: ha nulla, azt jelenti, hogy az eszköz az INT 13H-n keresztüli használata támogatott, egyébként pedig nem. A DH-t néhány IBM BIOS támogatja.
- az eredeti IBM BIOS-ok esetén néhány egyéb regiszter is tartalmazhat bizonyos értékeket (DS, ES, SS=0000H; SP=0400H), de erre nem ajánlatos számítani, mert más BIOS-ok meg más értékeket hagynak ezekben a regiszterekben. Ezért az IBM, Microsoft, Digital Reseach, stb. MBR kódjai soha nem függenek ezektől.
- A Plug and Play BIOS-ok vagy BBS támogatás a DL-hez csatolva még biztosítanak egy mutatót a PnP telepítési adatszerkezetre is az ES:DI regiszterpárban. Ez az információ lehetővé teszi a betöltő kódnak, hogy aktívan közreműködjön a BIOS-szal többek között például a boot sorrend módosításában. Mindazonáltal ezt az információt a legtöbb szabványos MBR és VBR kód figyelmen kívül hagyja. Az eredeti elképzelés az volt, hogy ez ES:DI értéke továbbadódik a VBR-nek az operációs rendszer általi további felhasználásra, de a PnP-t használó operációs rendszereknek általában megvan a módszerük arra, hogy megkapják a BIOS-tól a PnP belépési pontot később is, így a legtöbb operációs rendszer ezt nem várja itt el.

Host Protected Area (HPA)

A HPA olyan terület a merevlemezen, ami általában nem látható az operációs rendszer számára. Elsőként 2001-ben az ATA-4 szabványban jelent meg.

Az IDE vezérlőnek vannak regiszterei, amiket le lehet kérdezni ATA parancsokkal. Ezek a parancsok a vezérlőre csatlakoztatott meghajtóról adnak információt. A HPA létrehozásához három ATA parancsot kell ismerni:

- IDENTIFY DEVICE
- SET MAX ADDRESS
- READ NATIVE MAX ADDRESS

Az operációs rendszerek az IDENTIFY DEVICE parancsot használják arra, hogy meghatározzák a merevlemez címezhető tárterületét. Ez a parancs egy IDE vezérlőben lévő speciális regiszter értékét adja vissza. Ezt az értéket a regiszterben viszont simán meg lehet változtatni a SET MAX ADDRESS paranccsal. Ha a regiszterben lévő érték kisebb, mint az eszköz tulajdonképpeni mérete, akkor valójában létrejött egy védett rejtett terület, amit HPA-nak hívnak. Ez azért lesz védett, mert az operációs rendszer csak a regiszterben

lévő értéket fogja használni, amit az IDENTIFY DEVICE visszaad és így nem fogja megcímezni azt a területet, ami a HPA-ban van. A HPA-nak persze csak akkor van értelme, ha más szoftver (például a BIOS vagy egyéb firmware) viszont tudja használni. Az ilyen szoftvert HPA képes-nek hívnak. Ezek a READ NATIVE MAX ADDRESS parancsot használják, ami olyan regisztert ér el, amiben már a merevlemez valódi mérete van. A terület használatához a HPA-képes program megváltoztatja az IDENTIFY DEVICE által lekérdezett regiszterben lévő értéket arra, amit a READ NATIVE MAX ADDRESS visszaadott.

A HPA-t számos bootoló és diagnosztikai eszköz használja általában a BIOS-szal együttműködve.

El Torito

Az El Torito bootolható CD specifikáció az ISO 9660 1994 novemberében bejelentett kiterjesztése és arra szolgál, hogy lehetővé tegye a számítógépnek, hogy CD-ROM-ról bootoljon. A szabvány két különböző módot biztosít erre. Az egyik a merevlemezes emuláció, amikor a boot információt közvetlenül a CD-ről olvassa be a gép, a másik a floppy emuláció, amikor a boot információt egy floppy lemezképben tárolják a CD-n. Ezt a BIOS beolvassa és aztán úgy kezeli, mint egy virtuális floppy meghajtót. Ez az eset a nagyjából 1999 előtt gyártott számítógépeknél hasznos, amik csak floppyról tudnak bootolni. A mai gépeknél az emulálás nélküli megoldás sokkal megbízhatóbb. A BIOS egy meghajtószámot fog társítani a CD meghajtóhoz is. Ez a szám lehet 80H (merevlemezes emulálás), 00H (floppy emulálás), vagy tetszőleges érték, ha a BIOS nem akar emulációt. A szabvány a legenda szerint onnan kapta a nevét, hogy az IBM és a Phoenix Technologies két mérnöke a Kaliforniai Irvine-ban lévő El Torito étteremben találták ki.

MBR-VBR kapcsolat

A szabályok szerint egy szabványos MBR egy felétel nélküli ugrással átadja a vezérlést a sikeresen betöltött VBR-nek a 0000H:7C00H címre. Eközben a processzor továbbra is valós módban marad és a következő regiszterek értéke a lényeges:

- **CS:IP = 0000H:7C00H**
- **DL = a boot eszköz (lásd fent)**. Az MS-DOS 2.0-7.0 és PC DOS 2.0-6.3 MBR-jei nem adják tovább a megkapott DL értéket, hanem a kiválasztott elsődleges partíció boot státusz bejegyzését használják fel. Miután ez a szabvány szerint a legtöbb MBR partíciós táblában 80H, ez nem változtatja meg a dolgokat, hacsak a BIOS nem próbálkozik olyan fizikai eszközről bootolni, ami nem az első merevlemez a listában. Ez az oka annak, hogy miért nem tudnak ezek az operációs rendszerek például második merevlemezről bootolni. Néhány fdisk eszköz lehetővé teszi, hogy másodlagos lemezeken lévő partíciókat is aktívnak jelöljünk. Ebben a helyzetben, tudva, hogy ezek az operációs rendszerek képtelenek másik meghajtóról bootolni, néhányuk továbbra is használja a 80H értéket aktív jelölőnek, míg mások az aktuálisan társított fizikai eszköznek megfelelő értéket használnak (81H, 82H, stb.), így lehetővé teszik a bootolást másik meghajtóról is, legalábbis elméletben. Valójában ez csak néhány MBR kódnál működik, mégpedig azoknál, amik a boot státusz bejegyzés beállított 7. bitjét veszik aktív jelzőbitnek, nem pedig a teljes 80H értékre támaszkodnak. Az MS-DOS, PC-DOS MBR-jeibe be van égetve a fix 80H érték használata. A tulajdonképpeni fizikai eszközszámnak a partíciós táblában való használata szintén tud problémákat okozni, amikor a fizikai meghajtók BIOS-beli hozzárendelései megváltoznak. Például amikor meghajtót eltávolítanak, hozzáadnak vagy kicserélnek. Ennélfogva egy normál MBR-nek az biztosítja a legnagyobb rugalmasságot, ha csak a 7. bitet fogadja el aktív flag-nek és egyébként a BIOS által eredetileg adott DL-beli értéket használja és adja tovább a VBR felé. Az MS-DOS 7.1-8.0 MBR-jei már megváltoztak, csak a 7. bitet figyelik, a 01H-7FH értékeket érvénytelennek tekintik, viszont még mindig a

partíciós táblából veszik a fizikai eszköz számát a BIOS által beállítot DL helyett.

- a DH és az ES:DI tartalmát szintén meg kell tartania az MBR-nek a PnP támogatáshoz, viszont sok MBR, például az MS-DOS 2.0-8.0, PC-DOS 2.0-6.3 és a Windows NT, 2000, XP nem így tesz. Ez persze nem meglepő, mert ezek a DOS verziók még a PnP BIOS szabvány előtt készültek és a korábbi szabvány még csak a DL értékének megtartását ajánlotta.

Mit rejt egy valódi MBR kód?

A Windows 95B-ig az alábbi módon nézett ki a Microsoft által használt MBR az 1BDH címig, vagyis a partíciós tábla előtti utolsó bájtig:

```
      0|  1|  2|  3|  4|  5|  6|  7|  8|  9|  A|  B|  C|  D|  E|  F|0123456789ABCDEF
000  FA| 33| C0| 8E| D0| BC| 00| 7C| 8B| F4| 50| 07| 50| 1F| FB| FC|_3_____|__P_P___
010  BF| 00| 06| B9| 00| 01| F2| A5| EA| 1D| 06| 00| 00| BE| BE| 07|________________
020  B3| 04| 80| 3C| 80| 74| 0E| 80| 3C| 00| 75| 1C| 83| C6| 10| FE|___<_t__<_u_____
030  CB| 75| EF| CD| 18| 8B| 14| 8B| 4C| 02| 8B| EE| 83| C6| 10| FE|_u______L_______
040  CB| 74| 1A| 80| 3C| 00| 74| F4| BE| 8B| 06| AC| 3C| 00| 74| 0B|_t__<_t____<_t_
050  56| BB| 07| 00| B4| 0E| CD| 10| 5E| EB| F0| EB| FE| BF| 05| 00|V_______^_______
060  BB| 00| 7C| B8| 01| 02| 57| CD| 13| 5F| 73| 0C| 33| C0| CD| 13|__|___W___s_3___
070  4F| 75| ED| BE| A3| 06| EB| D3| BE| C2| 06| BF| FE| 7D| 81| 3D|Ou___________}_=
080  55| AA| 75| C7| 8B| F5| EA| 00| 7C| 00| 00| 49| 6E| 76| 61| 6C|U_u_____|__Inval
090  69| 64| 20| 70| 61| 72| 74| 69| 74| 69| 6F| 6E| 20| 74| 61| 62|id_partition_tab
0A0  6C| 65| 00| 45| 72| 72| 6F| 72| 20| 6C| 6F| 61| 64| 69| 6E| 67|le_Error_loading
0B0  20| 6F| 70| 65| 72| 61| 74| 69| 6E| 67| 20| 73| 79| 73| 74| 65|_operating_syste
0C0  6D| 00| 4D| 69| 73| 73| 69| 6E| 67| 20| 6F| 70| 65| 72| 61| 74|m_Missing_operat
0D0  69| 6E| 67| 20| 73| 79| 73| 74| 65| 6D| 00| 00| 00| 00| 00| 00|ing_system______
0E0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
0F0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
100  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
110  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
120  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
130  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
140  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
150  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
160  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
170  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
180  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
190  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
1A0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
```

```
1B0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|   |   |_______________
```

Az első 139 bájt (00H-tól 8AH-ig) a végrehajtható kód, a következő 80 bájt pedig (8BH-tól DAH-ig) a hibaüzeneteket tartalmazza (mindegyiket 0 zárja). A partíciós tábláig fennmaradó 227 bájtot az `fdisk` nullákkal tölti fel. Érdekesség, hogy a legelső partíciós táblát kezelő IBM PC DOS 2.0 által gyártott MBR-ben nem a "`Missing operating system`" szöveggel ér véget az üzenetek rész, mert ezután közvetlenül még volt egy "`Author - David Litton`" szöveg is.

Ennek az assembly változata a következőképp néz ki:

```
7C00 CLI                    ; Maszkolható megszakítások letiltása
7C01 XOR    AX,AX           ; Kinullázzuk az akkumulátort és
7C03 MOV    SS,AX           ;  a veremszegmens-regisztert.
7C05 MOV    SP,7C00         ; A veremmutatót beállítjuk a 0000:7C00 címre
7C08 MOV    SI,SP           ; Forrásindex: innen fogunk másolni...
7C0A PUSH   AX
7C0B POP    ES              ; Kinullázzuk az extraszegmens-regisztert
7C0C PUSH   AX
7C0D POP    DS              ; Kinullázzuk az adatszegmens-regisztert
7C0E STI                    ; Megszakítások engedélyezése
7C0F CLD                    ; Az irány jelzőbit törlése (DF=0).
7C10 MOV    DI,0600         ; Célindex: a kód másolata a 0000:0600 címen fog kezdődni
7C13 MOV    CX,0100         ; 256 szó (512 bájt) másolása (100H = 256)
7C16 REP                    ; A következő MOVSW utasítás ismétlése CX-szer
7C17 MOVSW                  ; Egyszerre két bájt másolása
7C18 JMP    0000:061D       ; A kód másolatára való ugrás...

; Mivel a fenti kód átmásolja önmagát és az összes további kódrészt a 0000:0600 címre,
; aztán elugrik a 0000:061D címre, a további kódlistában a címek át lettek alakítva úgy,
; hogy a kód futás közbeni állapotát mutassák.

; A kód alábbi része megpróbál megkeresni egy bejegyzést a partíciós táblában, ami
; aktívnak van jelölve, vagyis bootolható. Ennek jelzésére a bejegyzés első bájtja
; használatos. Ha ez 80H, akkor jó, ha 00H, akkor az nem bootolható. Ha a négy lehetőség
; közül egyik sem aktív, akkor hibaüzenet jelenik meg.

061D MOV    SI,07BE         ; A partíciós tábla első bejegyzésének a címe
0620 MOV    BL,04           ; Legfeljebb 4 táblabejegyzés van.
```

```
0622 CMP    BYTE PTR [SI],80; Ez bootolható (80H)?
0625 JE     0635            ; Igen, ugorjunk a következő ellenőrzésre!
0627 CMP    BYTE PTR [SI],00; Nem. Ez 00H? Ha nem, akkor ez egy érvénytelen partíciós
                            ; táblabejegyzés.
062C ADD    SI,+10          ; Nézzük a következő bejegyzést (10H = 16 bájt bejegyzésenként)
062F DEC    BL              ; Csökkentjük a számlálót.
0631 JNZ    0622            ; Minden bejegyzést megnéztünk?
0633 INT    18              ; Igen, és egyik sem bootolható, úgyhogy hívjunk egy 18H
                            ; megszakítást.
; Sok BIOS egyszerűen kiír egy "PRESS A KEY TO REBOOT" üzenetet ennél a megszakításnál.

; Találtunk aktív partíciót, úgyhogy a többi bejegyzéseket meg kell nézni, hogy nem
; bootolhatóak-e. Ha ezek között van másik bootolható, akkor hiba van, mert egyszerre csak egy
; ilyen lehet. Mielőtt ezt megtennénk, betöltjük a fej, meghajtó, cilinder és szektor adatokat a
; DX és CX regiszterekbe a 13-as megszakítás későbbi használatához.

0635 MOV    DX,[SI]           ; Meghajtó a DL-be; fej a DH-ba
; Szabványos MBR kód setén a DL mindig 80H, ami azt jelenti, hogy csak az első
; meghajtóról lehet bootolni. (A kód ezen részét gyakran megváltoztatják egyedi
; MBR-ek, hogy más meghajtóról is lehessen rendszert indítani.)

0637 MOV    CX,[SI+02]      ; Szektor a CL-be; cilinder a CH-ba
063A MOV    BP,SI           ; Az aktív bejegyzés offszetjének elmentése, hogy majd a
                            ; VBR-nek át lehessen adni.
063C ADD    SI,+10          ; A következő bejegyzésre lépünk.
063F DEC    BL              ; Ez az utolsó bejegyzés?
0641 JZ     065D            ; minden bejegyzés rendben lévőnek tűnik, úgyhogy...
                            ; ugrás a boot-rutinra!
0643 CMP    BYTE PTR [SI],00; Nem bootolható bejegyzés (00H)?
0646 JE     063C            ; Igen, nézzük a következő bejegyzést.

; Ha volt hiba, akkor ez a következő rutin megjeleníti azt az üzenetet, amire az SI mutat.
; A 0 végű ASCII sztring kiírása után a program végtelen ciklusba kerülve lezárja a gépet.

0648 MOV    SI,068B         ; "Invalid partition table"
064B LODSB                  ; AL-be töltjük az SI-nél lévő bájtot
```

```
                            ; és inkrementáljuk az SI értékét.
064C CMP    AL,00           ; Ez már nulla?
064E JE     065B            ; Ha igen, kész vagyunk. Ha nem...
0650 PUSH   SI              ; Sztringmutató tárolása a vermen.
0651 MOV    BX,0007         ; Az 10-s megszakítás 0E (szöveg írása) függvényének használa
0654 MOV    AH,0E           ; ahhoz, hogy az AL-ben lévő karaktermutató által jelölt
0656 INT    10              ; sztring a képernyőre íródjon.
0658 POP    SI
0659 JMP    064B
065B JMP    065B            ; Végtelen ciklus. Innen csak gép kikapcsolás vagy reset segít.

; Most már be tudjuk tölteni az aktív partíció első szektorát (a legtöbb meghajtón ez az első
; vagy egyetlen partíció 63-as abszolút szektora lenne. A 2.-62. szektorok általában üresek,
; hacsak nincs valami nagy MBR lecserélő kód oda telepítve.)
; A partíciós bejegyzés első két szava a partíció első szektorának meghajtó/fej és
; szektor/cilinder számai. Ez az adat olyan formátumban van, ami az alábbi INT 13H hívás számára
; szükséges.

065D MOV    DI,0005         ; 5-ször ismételjük (ha szükséges)...
0660 MOV    BX,7C00         ; Az oprendszer boot szektorának betöltése a 0000:7C00 címre.
0663 MOV    AX,0201         ; 02H-s funkció: 1 szektor betöltése.
0666 PUSH   DI
0667 INT    13              ; Megjegyzés: ez a régi INT 13H-as hívás 1024 cilinderre korlátozott.
0669 POP    DI
066A JNC    0678            ; Átvitel bit be van állítva? Ha nem, ugorjunk!
066C XOR    AX,AX           ; Igen, tehát hiba volt! Akkor...
066E INT    13              ; ... reszeteljük a meghajtót (00H-s funkció)
0670 DEC    DI              ; Számláló csökkentése
0671 JNZ    0660            ; És újra próbálkozunk.

0673 MOV    SI,06A3         ; Ha pedig még mindig nem jó, akkor írjuk ki: "Error loading
0676 JMP    064B            ; operating system" és álljunk le!

; A fenti kódrészletet gyakran megváltoztatták az egyedi MBR-ek, amik szerint épp elég egyszer
; megpróbálkozni az operációs rendszer betöltésével. Ezt a kódot még egyértelműen azokban a
; régi időkben írták, amikor a merevlemezek, memóriák és maga a boot folyamat is eléggé
```

```
; megbízhatatlan volt.

; Amint az aktív partíció boot szektora betöltődött a memóriába, meg kell nézni, hogy
; érvényes-e. Ez úgy történik, hogy egyszerűen megnézzük a szektor utolsó szavát, aminek
; AA55H-nek kell lennie.

0678 MOV    SI,06C2          ; "Missing operating system"
                             ; Beállítjuk az SI-t, ha valami hiba van a bootszektor betöltésével.
067B MOV    DI,7DFE          ; A bootszektor utolsó szavára mutat. Ennek AA55H-nek kell lennie.
067E CMP    WORD PTR [DI],AA55 ; Annyi? ("aláírás-ellenőrzés")
0682 JNE    064B             ; Ha nem, akkor hibaüzenetet jelenítünk meg és
                             ; "lezárjuk" a rendszert
0684 MOV    SI,BP            ; SI és BP legyenek azonosak az aktív partíciós bejegyzéssel,
                             ; amit az operációs rendszer boot kódja fog majd használni.
0686 JMP    0000:7C00        ; Ugrás az oprendszer boot szektor kódjára és a bootolás folytatása!
```

Látható, hogy a kód nem is tölti ki teljesen a rendelkezésre álló helyet és az talán külön kiemelés nélkül is egyértelmű, hogy a kompatibilitás teljes elvesztése nélkül csak úgy lehet módosítani az MBR-t (például további partíciós bejegyzésekkel) ha a betöltő kódnak szánt helyből vesznek el. Az idők során az MBR kód természetesen egyre bonyolultabb lett, de az egyes lépéseket idő hiányában már nem tárgyalom ilyen részletesen.

Az MBR kiegészítései

A Windows 95B MBR-je

1996-ban a Windows 95B megjelenésével bejött a FAT32 fájlrendszer és ehhez szükség volt több változtatásra a lemezkezelő segédprogramok mellett az MBR kódjában is. A változtatások tovább éltek a Windows 98, Windows ME korszakban is. Alapvetően ez az MBR abban hozott újat, hogy a FAT32 boot szektorok már olyan partíciókon is lehettek, amik túl vannak a korábbi MBR által kikötött határon (1024 cilinder). A kódban több kisebb-nagyobb változás történt és itt már az INT 13H megszakítás új, 42H-es funkciója van használva (amennyiben elérhető), ami már LBA címekkel dolgozik.

A kódba ágyazottan megjelent egy új dolog is, mégpedig a lemez időbélyeg. Ez a DAH-tól DFH-ig tartó hat bájt. Új MBR írásakor ezek értéke alapértelmezetten 0. Ezt az időbélyeget a boot során a rendszer megnézi és ha mindegyik nulla, akkor kicseréli az értékét a következőképpen:

- A DAH-DBH értéke mindig 0.
- DCH: a fizikai meghajtószám (80H: az első merevlemez, 81H: a második merevlemez, stb.) Ez persze csak azt az állapotot rögzíti, amikor az operációs rendszer arra a meghajtóra írt.
- DDH-DFH: az idő, amikor ezeket az MBR-be írták, fordított sorrendben. (36 09 17: 17:09:36)

Ennek az időbélyegnek a pontos funkciója nem ismert, viszont a szektorról szektorra készített lemezképek használatát kissé megnehezítette, mert összeakadásokat okozhatott.

A Windows 2000/XP MBR-je

A Windows 2000 az időbélyeget megszüntette, azonban megjelent egy új fogalom, a lemez azonosító. (Az MBR-be írandó kódot egyébként a Windows telepítési könyvtára alatt a system32\dmadmin.exe fájl tartalmazza.)

Az MBR kódja itt már 300 bájtosra növekedett, amit 80 bájtnyi hibaüzenet-szöveg tartalmazott (angol nyelv esetén. Más nyelvnél a hibaüzenet szövegének mérete természetesen eltért.). Ezközött és a partíciós tábla között azonban már nem csak nullák voltak, hanem az ún. lemez azonsító (1B8H-tól 1BBH-ig). Ez egy 32 bites érték, ami azonosítja a meghajtót az operációs rendszer számára. Bár a Windows NT 3.5 vezette be, ma már számos operációs rendszer használja, például a Linux 2.6 utáni kernelek is. (A Linuxos eszközök arra használják, hogy rögzítsék, melyik lemezről indították a gépet.)

A Windows NT (és a későbbi Microsoft operációs rendszerek) arra használják a lemez aláírást, hogy azonosítsák vele az összes partíciót, amit bármilyen lemezen a géphez csatlakoztattak. Ezeket az aláírásokat a regisztrációs adatbázisban tárolják és például ezek segítségével kezelik a megfeleltetéseket a partíciók és a meghajtó betűjelek között. De emellett a boot.ini fájlokban is használható arra (bár legtöbbször nem teszik), hogy megadjunk vele további bootolható Windows partíciókat. Ezek az aláírások például a következő helyen jelennek meg a Windows 2000/XP regisztrációs adatbázisában:

`HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices\`

Ha egy MBR-ben tárolt lemez aláírás A8H E1H B9H D2H (ebben a sorrendben) és annak az első partíciója megfelel a Windows alatti logikai C: meghajtónak, akkor a \DosDevices\C: kulcsa alatti REG_BINARY adat: A8H E1H B9H D2H 00H 7EH 00H 00H 00H 00H 00H 00H

Az első négy bájt a lemez aláírás. (Más kulcsokban ezek a bájtok pont ellenkező sorrendben is megjelenhetnek, mint ahogy az MBR szektorban vannak.) Ezt követi nyolc további bájt, ami egy 64 bites egésztet ad ki little endian sorrendben. Ez arra való, hogy meghatározza a partíció bájtoffszetjét. Ebben az esetben 00H 7EH a 7E00H hexadecimális értéknek felel meg (32256). Ha feltesszük, hogy a kérdéses egységen 512 bájtos a szektorméret, akkor ezt 512-vel elosztva 63 jön ki, ami a partíció első szektorának vagy LBA-ja.

Ha ezen a lemezen van egy másik partíció, aminél 00H F8H 93H 71H 02H követi a lemezaláírást (mondjuk a \DosDevices\D: kulcs alatt), akkor az a 00027193F800H (10 495 457 280) offszeten kezdődik, ami a 20 498 940-es fizikai szektor első bájtja.

A Windows Vistával kezdődően a lemezaláírástól függ a boot folyamat is, ezért az eltárolódik a Boot Configuration Data (BCD) nevű tárhelyen is. Ha a lemez aláírása megváltozik, nem található vagy pedig ütközik valamivel, akkor a Windows nem tud betöltődni.

A Windows 7-8 MBR-je

A Windows 7, 8 és 8.1 ugyanazt az MBR-t használja. A kód immár majdnem a teljes rendelkezésre álló helyet kitölti. A partíciós tábla természetesen nem változott, ugyanakkor az aktív és bootolható partíció kifejezések némileg tisztázásra szorulnak. A Windows 7 előtt ezek a kifejezések egymás szinonímái voltak. Akkor volt egy partíció bootolható, ha a partíciós tábla első bájtjának értéke 80H volt. A Windows 7 azonban üres lemezre egy rendszerpartíciót is létrehoz az operációs rendszer meghajtóján kívül, ami a boot manager kódot és a BCD adatbázist tartalmazza és valakik a Microsoftnál úgy döntöttek, hogy az operációs rendszert tartalmazó partíciót nevezik el

boot partíciónak, a bootoláskor használtat pedig aktívnak. Egy Windows 7 MBR az 1BDH címig, vagyis a partíciós tábla előtti utolsó bájtig így néz ki:

```
      0|  1|  2|  3|  4|  5|  6|  7|  8|  9|  A|  B|  C|  D|  E|  F|0123456789ABCDEF
000  33| C0| 8E| D0| BC| 00| 7C| 8E| C0| 8E| D8| BE| 00| 7C| BF| 00|3_____|______|__
010  06| B9| 00| 02| FC| F3| A4| 50| 68| 1C| 06| CB| FB| B9| 04| 00|_______Ph_______
020  BD| BE| 07| 80| 7E| 00| 00| 7C| 0B| 0F| 85| 0E| 01| 83| C5| 10|_______|________
030  E2| F1| CD| 18| 88| 56| 00| 55| C6| 46| 11| 05| C6| 46| 10| 00|_____V_U_F___F__
040  B4| 41| BB| AA| 55| CD| 13| 5D| 72| 0F| 81| FB| 55| AA| 75| 09|_A__U__]r___U_u_
050  F7| C1| 01| 00| 74| 03| FE| 46| 10| 66| 60| 80| 7E| 10| 00| 74|____t__F_f`____t
060  26| 66| 68| 00| 00| 00| 00| 66| FF| 76| 08| 68| 00| 00| 68| 00|&fh____f_v_h__h_
070  7C| 68| 01| 00| 68| 10| 00| B4| 42| 8A| 56| 00| 8B| F4| CD| 13||h__h___B_V_____
080  9F| 83| C4| 10| 9E| EB| 14| B8| 01| 02| BB| 00| 7C| 8A| 56| 00|____________|_V_
090  8A| 76| 01| 8A| 4E| 02| 8A| 6E| 03| CD| 13| 66| 61| 73| 1C| FE|_v__N__n___fas__
0A0  4E| 11| 75| 0C| 80| 7E| 00| 80| 0F| 84| 8A| 00| B2| 80| EB| 84|N_u_____________
0B0  55| 32| E4| 8A| 56| 00| CD| 13| 5D| EB| 9E| 81| 3E| FE| 7D| 55|U2__V___]___>_}U
0C0  AA| 75| 6E| FF| 76| 00| E8| 8D| 00| 75| 17| FA| B0| D1| E6| 64|_un_v____u_____d
0D0  E8| 83| 00| B0| DF| E6| 60| E8| 7C| 00| B0| FF| E6| 64| E8| 75|______`_|____d_u
0E0  00| FB| B8| 00| BB| CD| 1A| 66| 23| C0| 75| 3B| 66| 81| FB| 54|_______f#_u;f__T
0F0  43| 50| 41| 75| 32| 81| F9| 02| 01| 72| 2C| 66| 68| 07| BB| 00|CPAu2____r,fh___
100  00| 66| 68| 00| 02| 00| 00| 66| 68| 08| 00| 00| 00| 66| 53| 66|_fh____fh____fSf
110  53| 66| 55| 66| 68| 00| 00| 00| 00| 66| 68| 00| 7C| 00| 00| 66|SfUfh____fh_|__f
120  61| 68| 00| 00| 07| CD| 1A| 5A| 32| F6| EA| 00| 7C| 00| 00| CD|ah_____Z2___|___
130  18| A0| B7| 07| EB| 08| A0| B6| 07| EB| 03| A0| B5| 07| 32| E4|_______________2_
140  05| 00| 07| 8B| F0| AC| 3C| 00| 74| 09| BB| 07| 00| B4| 0E| CD|______<_t_______
150  10| EB| F2| F4| EB| FD| 2B| C9| E4| 64| EB| 00| 24| 02| E0| F8|______+__d__$____
160  24| 02| C3| 49| 6E| 76| 61| 6C| 69| 64| 20| 70| 61| 72| 74| 69|$__Invalid_parti
170  74| 69| 6F| 6E| 20| 74| 61| 62| 6C| 65| 00| 45| 72| 72| 6F| 72|tion_table_Error
180  20| 6C| 6F| 61| 64| 69| 6E| 67| 20| 6F| 70| 65| 72| 61| 74| 69|_loading_operati
190  6E| 67| 20| 73| 79| 73| 74| 65| 6D| 00| 4D| 69| 73| 73| 69| 6E|ng_system_Missin
1A0  67| 20| 6F| 70| 65| 72| 61| 74| 69| 6E| 67| 20| 73| 79| 73| 74|g_operating_syst
1B0  65| 6D| 00| 00| 00| 63| 7B| 9A| 81| CD| A3| A7| 00| 00|   |   |em___c{_______
```

Enhanced Disk Drive Services

Az INT 13H még nagyobb címzési módokra való támogatását a Western Digital és a Phoenix Technologies

fejlesztette ki és BIOS Enhanced Disk Drive Services (EDD) néven ismeretes. Az EDD már 64 bites LBA címeket is kezel. Az új INT 13H EDD funkció egy ún. csomagot használó hívás és egy megadott szerkezetnek megfelelő információcsomagra hivatkozó mutatót vár, nem pedig feltöltött regisztereket, mint az INT 13H más funkciói. Ez a csomag tartalmazza az interfész verziót, adatméretet és az LBA-kat.

BitLocker és TPM

A Microsoft rövid leírásából állítottam össze az alábbi ismertetést erről a két fogalomról.

A Windows BitLocker titkosít minden adatot a Windowst tartalmazó köteten. (Egy kötet egy vagy több merevlemez egy vagy több partíciójából állhat. A BitLocker egyszerű kötetekkel dolgozik, ahol egy kötet egy partíció.)

A platformmegbízhatósági modul (TPM) a számítógépbe beépített mikrocsip. Ez a csip kriptográfiai információk, például a titkosítási kulcsok tárolására szolgál. A TPM általában az asztali- vagy hordozható számítógépek alaplapjára van építve és a hardver buszon keresztül kommunikál a rendszer többi részével. A platformmegbízhatósági modullal rendelkező számítógépek létre tudnak hozni kriptográfiai kulcsokat, és titkosítani is tudják azokat. Ezek után már csak a TPM tudja visszafejteni ezeket a kulcsokat. Ez a gyakran a kulcs "kötésének" vagy "burkolásának" nevezett folyamat segít megelőzni, hogy kitudódjon a kulcs. Minden TPM tartalmaz egy fő burkolókulcsot, az úgynevezett tárolási gyökérkulcsot (Storage Root Key - SRK), amely magában a platformmegbízhatósági modulban tárolódik. Egy TPM által készített kulcs titkos része nem elérhető semmilyen másik összetevő, szoftver, folyamat vagy személy számára.

A platformmegbízhatósági modullal rendelkező számítógépek létre tudnak hozni olyan kulcsokat is, amelyek nemcsak hogy be vannak burkolva, de amelyek használata bizonyos hardver és szoftver feltételekhez is van rögzítve. Ezt nevezik a kulcs "lezárásának". A lezárt kulcs első létrehozása során a TPM készít egy pillanatképet a konfigurációs értékekről és a fájlkivonatokról. Egy lezárt kulcsot csak akkor lehet "feloldani", ha az aktuális rendszerértékek megegyeznek a pillanatképben tároltakkal.. A BitLocker lezárt kulcsokat használ, hogy a Windows integritása elleni támadásokat is fel tudja ismerni. A TPM használatával a kulcspárok titkos része az operációs rendszer által irányított memóriától függetlenül tárolódik. Mivel a TPM saját belső vezérlőprogrammal és logikai áramkörökkel rendelkezik a folyamatok feldolgozásához, így nem függ az operációs rendszertől, és nem befolyásolják a külső szoftverek biztonsági hibái.

A BitLocker használható TPM nélkül is. Ha a számítógép rendelkezik kompatibilis TPM modullal, akkor a BitLocker a TPM modullal zárolja az adatokat védő titkosítási kulcsokat. Ennek eredményeképp a kulcsok nem érhetők el addig, amíg a TPM nem ellenőrizte a számítógép állapotát. Mivel az adatok visszafejtéséhez szükséges kulcsokat a TPM tárolja, a támadó nem férhet az adatokhoz úgy, ha eltávolítja a merevlemezt, és egy másik számítógépre próbálja telepíteni. A rendszerindítási folyamat során a TPM csak azután nyitja meg a titkosított partíció visszafejtéséhez szükséges kulcsot, miután összevetette az operációs rendszer fontos konfigurációs értékeit tartalmazó kivonatot egy korábban készített pillanatképpel. Ez igazolja, hogy a Windows rendszerindítási folyamat nem sérült. A TPM nem oldja ki a kulcsot, ha a Windows telepítési

beállítások módosítását észleli.

A fenti bájtsorozat assemblyben a következőképp néz ki:

```
7C00 XOR    AX,AX           ; Kinullázzuk az akkumulátort és
7C02 MOV    SS,AX           ;  a veremszegmens-regisztert.
7C04 MOV    SP,7C00         ; A veremmutatót beállítjuk a 0000:7C00 címre
7C07 MOV    ES,AX           ; Mivel az AX már nulla, kinullázzuk az extraszegmens-regisztert
7C09 MOV    DS,AX           ;  és az adatszegmens-regisztert is.
7C0B MOV    SI,7C00         ; Forrásindex: innen fogunk másolni
7C0E MOV    DI,0600         ; Célindex: a kód másolata a 0000:0600 címen fog kezdődni
7C11 MOV    CX,0200         ; Beállítjuk a számlálót (CX) mind az 512 bájt (200H) másolásához
7C14 CLD                    ; Az irány jelzőbit törlése (DF=0).
7C15 REP                    ; A következő MOVSB utasítás ismétlése CX-szer
7C16 MOVSB                  ; Egyszerre egy bájt másolása
7C17 PUSH   AX              ; Beállítjuk a szegmenst (AX) és az offszetet (DI)
7C18 PUSH   061C            ;  a 0000:061C-re való ugráshoz
7C1B RETF                   ; A RETF-et használjuk az átmásolt kódra való ugráshoz

; Mivel a fenti kód átmásolja önmagát és az összes további kódrészt a 0000:0600 címre,
; aztán elugrik a 0000:061C címre, a további kódlistában a címek át lettek alakítva úgy,
; hogy a kód futás közbeni állapotát mutassák.

; A kód alábbi része megpróbál keresni egy aktív, vagyis bootolható bejegyzést a partíciós
; táblában. Ennek jelzésére a bejegyzés első bájtja használatos. Ha ez 80H,
; akkor jó. Ha 00H, akkor az nem bootolható. Egyéb esetek érvénytelen partíciós táblát
; jelentenek. Ha a négy partíció közül egyik sem aktív, akkor hibaüzenet jelenik meg.
; A Windows 2000/XP előtti MBR kódok a BP helyett az SI regisztert használták.

061C STI                    ; Megszakítások engedélyezése
061D MOV    CX,0004         ; Legfeljebb négy bejegyzés van
0620 MOV    BP,07BE         ; A partíciós tábla első bejegyzésének címe
0623 CMP    BYTE PTR [BP+00],00 ; A bejegyzés első bájtjának vizsgálata az SS:[BP+00] címen.
                            ;  Bármi 80H-tól FFH-ig kisebb mint nulla, tehát:
0627 JL     0634            ; Lehetséges bejegyzést találtunk, ezért azt bővebben megnézzük a
                            ;  0634H-n
0629 JNZ    073B            ; De ha nem nulla (és nagyobb), akkor hiba van, mert 01H-től 79H-ig
```

```
                            ;  tartalmaz értéket. Jön az "Invalid partition table"
                            ;  üzenet. Egyébként nullát kaptunk, ilyenkor folytatjuk a keresést.
062D ADD    BP,+10          ; A következő bejegyzés (egy bejegyzés 16 bájt, ami 10H).
0630 LOOP   0623            ; Vissza a következő bejegyzésre, hacsak nem nulla már a CL
                            ;  ami azt jelenti, mind a négyet kipróbáltuk.
0632 INT    18              ; Mind a négyet kipróbáltuk, egyik sem bootolható, úgyhogy hívjunk
                            ;  egy 18H megszakítást.
; Sok BIOS egyszerűen kiír egy "PRESS A KEY TO REBOOT" üzenetet ennél a megszakításnál.

0634 MOV    [BP+00],DL      ; A DL-t a BIOS már beállította 80H-ra.
0637 PUSH   BP              ; A bázismutatót a veremre rakjuk.
0638 MOV    BYTE PTR [BP+11],05 ; Adatot tárolunk a 069FH-en lévő utasítás esetleges
                            ;  használatához.
063C MOV    BYTE PTR [BP+10],00 ; Flagként és/vagy számlálóként van használva a telepített
                            ;  INT 13H kiterjesztés használatához. Lásd a 0656H és 065BH címeket
                            ;  lentebb.
0640 MOV    AH,41
0642 MOV    BX,55AA
0645 INT    13              ; INT 13H, 41H-es funkció (BX=55AAH értékkel):
                            ;  Ellenőrizzük az INT 13H kiterjesztést a BIOS-ban.
; Ha a CF bit nulla lett és a BX megváltozott AA55H-ra, akkor megvan a kiterjesztés.
; A főverzió az AH-ban: 01H=1.x; 20H=2.0/EDD-1.0; 21H=2.1/EDD-1.1; 30H=EDD-3.0
; A CX az API alcsoport támogatási térképe: ha a 0. bit be van állítva (CX=1, 3, 5, stb;
; páratlan), akkor támogatottak a kiterjesztett lemezkezelési funkciók (AH=42H-44H, 47H, 48H).
; Csak akkor lesz hibás 0650H-n a TEST, ha a kiterjesztett támogatás nem elérhető.

0647 POP    BP              ; Visszavesszük az eredeti bázismutatót.
0648 JB     0659            ; Kisebb? Ha igen, CF=1, nincs INT 13H kiterjesztés, ugorjunk!
064A CMP    BX,AA55         ; A BX tartalma megváltozott?
064E JNZ    0659            ; Ha nem, ugorjunk a 0659H-re.
0650 TEST   CX,0001         ; Utolsó teszt az INT 13H kiterjesztésre!
                            ; Ha a 0. bit nincs beállítva, akkor ez hibás lesz,
0654 JZ     0659            ;  tehát ugorjunk a következő sorra...
0656 INC    BYTE PTR [BP+10];  vagy növeljük a [BP+10h]-t egyel.
0659 PUSHAD                 ; Veremre mentjük az összes 32 bites regisztert ebben a sorrendben:
                            ;  eax, ecx, edx, ebx, esp, ebp, esi, edi.
```

```
065B CMP    BYTE PTR [BP+10],00 ; Összehasonlítjuk a [BP+10H]-t nullával,
065F JZ     0687                ;  ha 0, akkor nem tudjuk használni a kiterjesztéseket.

; A következő kód az INT 13H 42H "Kiterjesztett olvasás" funkciót használja a
; bootolható partíció első szektorának (VBR) 7C00H címre beolvasására. Ezt úgy csinálja,
; hogy először fordított sorrendben a veremre tesz egy ún. "lemezcím csomagot"
; (Disk Address Packet; DAP). Így a 00H (fenntartott) és a 10H bájtok lesznek a veremre tett
; utolsó bájtok a 0674H címen.
; Offszet Méret A lemezcím csomag tartalma
; ------- ----- ----------------------------------------------------------
;  00H    BYTE  A csomag mérete (10H vagy 18H; 16 vagy 24 bájt).
;  01H    BYTE  Fenntartott (00).
;  02H    WORD  Az olvasni kívánt blokkok száma (Jelen esetben csak 1).
;  04H    DWORD Az olvasási bufferre mutat (jelen esetben 0000:7C00).
;  08H    QWORD A kezdeti abszolút szektor (a partíciós táblából:
;               (00000000 + DWORD PTR [BP+08]). Ez legfeljebb 32 bit lehet.
;  10H    QWORD Itt nem használt. (Az EDD-3.0-ban opcionális; az olvasási buffer 64 bites
;               lapos címei. Csak akkor használatos, ha a 04H-n a DWORD értéke FFFF:FFFF.)

0661 PUSH   00000000        ; 4 bájt (32 bit) a veremre a VBR kezdőszektorának kiterjesztése
                            ;  miatt.
0667 PUSH   DWORD PTR [BP+08]; A VBR szektor helye
066B PUSH   0000            ; A szegmens, aztán az offszet, tehát:
066E PUSH   7C00            ; A szektor másolása a memóriába a 7C00 helyre
0671 PUSH   0001            ; Csak 1 szektor másolása.
0674 PUSH   0010            ; Csomagméret (16 bájt).
0677 MOV    AH,42           ; 42H-es funkció.
0679 MOV    DL,[BP+00]      ; Meghajtó azonosító
067C MOV    SI,SP           ; A DS:SI-nek a vermen lévő lemez cím csomagra kell mutatni.
067E INT    13              ; Megpróbáljuk beolvasni a VBR szektort a lemezről.

; Ha sikerült, akkor a CF (átvitel) 0 és az AH is nullára van állítva.
; Ha hiba volt, akkor a CF értéke 1 és az AH hibakódot tartalmaz.
; Bárelyik esetben a DAP blokkszám mezője a ténylegesen átvitt blokkok számát tartalmazza.

0680 LAHF                   ; Betöltjük a státusz jelzőbitet az AH-ba.
```

```
0681 ADD    SP,+10          ; Szépen eltávolítja az összes DAP bájtot a veremről a veremmutató
                            ;  módosításával.
0684 SAHF                   ; AH-t elmentjük a flagregiszterbe, így nem változnak meg a státusz
                            ;  flagek.
0685 JMP    069B

; Az MBR itt a szabványos INT 13H "Szektorok olvasása" funkciót használja, mert
; az INT 13H kiterjesztett funkciók a fentiek szerint (065F) nem elérhetők a BIOS-ban.

0687 MOV    AX,0201         ; 02H funkció, csak 1 szektort olvas.
068A MOV    BX,7C00         ; Olvasási buffer 7C00H-n kezdődik.
068D MOV    DL,[BP+00]      ; DL = meghajtó
0690 MOV    DH,[BP+01]      ; DH = fej száma (sose használj FFH-t).
0693 MOV    CL,[BP+02]      ; CL 0.-5. bitje (legnagyobb értéke 3FH) a szektorszám
0696 MOV    CH,[BP+03]      ; A CL 6-7. bitje a cilinder két legmagasabb bitje lesz (8-9.)
                            ; a CH 0.-7. bitjeivel. Legnagyobb értéke 3FFH.
0699 INT    13              ; INT 13H 02H funkciója: szektorok olvasása a memóriába az ES:BX
                            ; címtől

; Akár van kiterjesztés, akár nincs, mindkét rutin itt ér véget:

069B POPAD                  ; Az összes 32 bites regiszter visszaállítása a veremről,
                            ;  amiket 0659H-nél elmentettünk.
069D JNB    06BB
069F DEC    BYTE PTR [BP+11]; 0638H-tól 05H-val kezdődik.
06A2 JNZ    06B0            ; Ha 0, megpróbáljuk a meghajtóról 5-ször felolvasni a VBR-t.
06A4 CMP    BYTE PTR [BP+00],80
06A8 JZ     0736            ; "Error loading operating system"
06AC MOV    DL,80
06AE JMP    0634
06B0 PUSH   BP
06B1 XOR    AH,AH
06B3 MOV    DL,[BP+00]
06B6 INT    13
06B8 POP    BP
06B9 JMP    0659
```

```
06BB CMP    WORD PTR [7DFE],AA55 ; Aláírás vizsgálata
06C1 JNZ    0731                 ; Ha nem látjuk, hiba van! "Missing operating system"
06C3 PUSH   WORD PTR [BP+00]; Újra a DL-be téve a 0727-en (80H-t tartalmaz, ha az 1. meghajtó).

; A 06C6-tól 0726-ig lévő kód megnézi, hogy a TPM 1.2 interfész működik-e a rendszerben,
; mert ezt használhatja a BitLocker az operációs rendszer bootolásának lehetővé tételére
; miután megvizsgálta a PC kezdeti indítási komponenseinek integritását.
; Valamilyen TPM specifikáció miatt a Microsoft úgy gondolta, hogy bármilyen TMP INT 1AH
; funkcióba való belépés előtt elérhetővé kell tenni az 1 MB fölötti memóriát
; az A20 vonallal. Ezért az alábbi kód tulajdonképpen egy alprogram az 1 MB fölötti
; területek elérésének megszerzéséhez (A20 vonal engedélyezése néven is ismert).

; A címvonalak lehetővé teszik 2^n bájt memória elérését: az A0-A15 2^16=64 KB memóriát.
; Az A20 vonal lehetővé teszi 2^20 (1 MB) memóriaterület helyett 2^21 (2 MB) elérését.
; De a számítógépek úgy vannak tervezve, hogy az A20 vonal egyszerű engedélyezésével lehetővé
; válik 1 MB fölött bármennyi memória elérése ha mind a CPU mind a kód erre alkalmasak (amint
; valós módból kiléptek). Megjegyzendő, hogy ez a kód a 06C6-06E1-en
; és a szubrutin a 0756-on apróbb módosításokkal gyakorlatilag ugyanazok.

06C6 CALL   0756
06C9 JNZ    06E2
06CB CLI                         ; Törli az IF bitet, a CPU már nem fogad maszkolható megszakításokat.
06CC MOV    AL,D1
06CE OUT    64,AL
06D0 CALL   0756
06D3 MOV    AL,DF
06D5 OUT    60,AL
06D7 CALL   0756
06DA MOV    AL,FF
06DC OUT    64,AL
06DE CALL   0756
06E1 STI                         ; Maszkolható megszakításokat újra fogadunk.
06E2 MOV    AX,BB00              ; AH = BBH, AL = 00H
06E5 INT    1A                   ; INT 1A a TCG_StatusCheck funkció
06E7 AND    EAX,EAX              ; Ha az EAX nem nulla, akkor
06EA JNZ    0727                 ; nincs TCG BIOS támogatás
```

```
06EC CMP    EBX,41504354    ; Az EBX-nek is tartalmazni kell a "TCPA" ASCII karaktereket
                            ; (54, 43, 50, 41) további biztonsági ellenőrzés végett.
06F3 JNZ    0727            ; Ha nem így van, kilépünk a TCG kódból.
06F5 CMP    CX,0102         ; 1.2-es verzió vagy újabb?
06F9 JB     0727            ; Ha nem, kilépünk a TCG kódból.

; Ha van TPM 1.2, akkor végrehajtunk egy "TCG_CompactHashLogExtendEvent" eljárást.

06FB PUSH   0000BB07        ; Az INT 1AH híváshoz beállítás: AH = BBH, AL = 07H parancs.
0701 PUSH   00000200
0707 PUSH   00000008
070D PUSH   EBX
070F PUSH   EBX
0711 PUSH   EBP
0713 PUSH   00000000
0719 PUSH   00007C00
071F POPAD
0721 PUSH   0000
0724 POP    ES
0725 INT    1A

; Visszatérés után az EAX egy státuszkódot, az EDX pedig egy esemény-azonosítót tartalmaz.

0727 POP    DX              ; A [BP+00]-tól a 06C3H-n; gyakran 80H.
0728 XOR    DH,DH           ; Csak a DL számít.
072A JMP    0000:7C00       ; Ugrás a betöltött VBR kódra
072F INT    18              ; Ez talán a TPM 1.2-nek valami specifikációja miatt kell ide?
; A szokványos "INT18 ha nincs lemez" már fent van egyszer a 0632H címen.

; Ha egy hibaüzenet utolsó karaktere is megjelent, akkor a 0748H, 0753H és 0754H végtelen
; ciklusba teszik a gépet! Innen csak újraindítás segít.
; Az INT 10H 0EH funkciója (Teletype kimenet) használatos a hibaüzenetek karaktereinek
; megjelenítésére.

0731 MOV    AL,[07B7]       ; (A 7B7 értéke 9A) + 700 = 79AH
0734 JMP    073E            ; "Missing operating system" hibaüzenet.
```

```
0736 MOV    AL,[07B6]       ; (A 7B6 értéke 7B) + 700 = 77BH
0739 JMP    073E            ; "Error loading operating system" hibaüzenet.
073B MOV    AL,[07B5]       ; (A 7B5 értéke 63) + 700 = 763H
                            ;  "Invalid partition table" hibaüzenet.
073E XOR    AH,AH           ; Kinullázzuk az AH regisztert.
0740 ADD    AX,0700         ; A fentiekhez hozzáadjuk a 700H offszetet
0743 MOV    SI,AX           ; Az üzenet offszetje a forrás index regiszterbe.
0745 LODSB                  ; Karakter betöltése az SI által mutatott helyről az AL-be.
0746 CMP    AL,00           ; Elértük az üzenet végét jelző nullát?
0748 JZ     0753            ; Ha igen, akkor...
074A MOV    BX,0007         ; 0. oldal megjelenítése, normál fehér karakterekkel.
074D MOV    AH,0E           ; Teletype kimenet
074F INT    10              ; Egyszerre csak egy karaktert jelenít meg.
0751 JMP    0745            ; Menjünk vissza a következő karakterért.
0753 HLT
0754 JMP    0753            ; Csak akkor kerülünk ide, ha egy NMI lép fel, de nem hagyjuk
                            ; magunkat és újra visszaugrunk a HLT-re!

; Szubrutin: - Az A20 engedélyező kód része. Ellenőrzés és várakozás a billentyűzet vezérlőre.

0756 SUB    CX,CX           ; CX = 0 legyen.
0758 IN     AL,64           ; 64H port ellenőrzése.
075A JMP    075C            ; Páratlannak tűnik, de így lesz kész.
075C AND    AL,02           ; Csak az 1. bit nem beállított állapotának vizsgálata.
075E LOOPNE 0758            ; Folytassuk az ellenőrzést, amíg CX = 0 (és ZF = 1) nem lesz.
                            ; Ekkor kész.
0760 AND    AL,02
0762 RET
```

Kiterjesztett partíciók

Amikor az egyre nagyobb merevlemezek megjelenésével az MBR által biztosított négy partíció kezdett szűk keresztmetszet lenni, valami olyasmit kellett kitalálni, ami a kompatibilitást megtartva lehetővé teszi, hogy igény szerint mégiscsak lehessen négynél több partíciót definiálni. A megoldás a kiterjesztett partíció lett. Ez gyakorlatilag egy speciális partíciótípus az MBR-ben:

- 05H: kiterjesztett DOS partíció CHS címzéssel. Egy ilyennek a lemez első fizikai 8 GB-jában kell lennie.
- 0FH: kiterjesztett DOS partíció LBA címzéssel.
- 85H: kiterjesztett Linux partíció

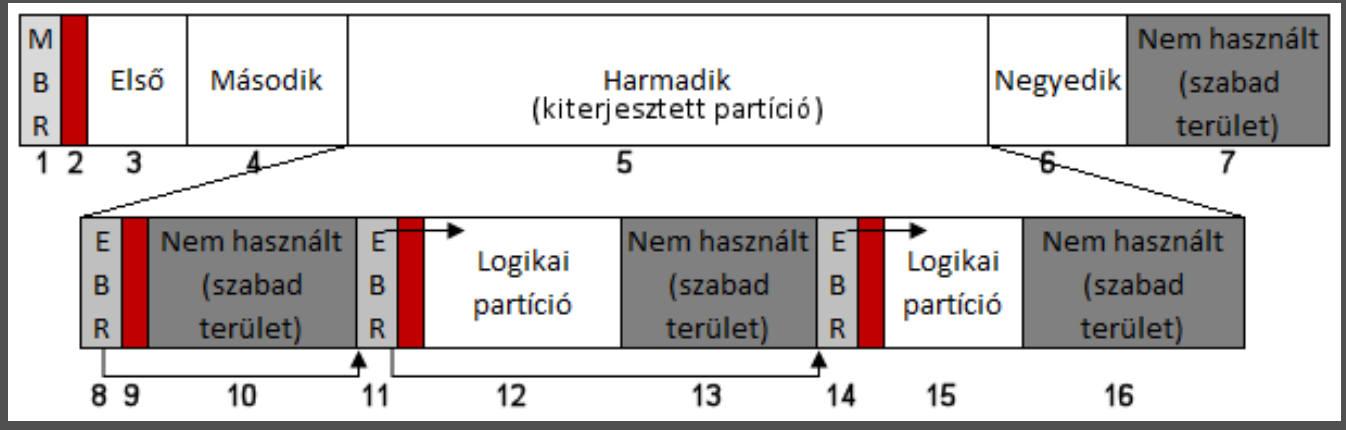
- C5H, D5H: védett kiterjesztett partíció

A kiterjesztett partíció az MBR szemszögéből tehát gyakorlatilag egy konténer, a típusát kivéve semmiben sem különbözik a többitől. Az MS-DOS 3.30 és későbbi operációs rendszerek alatt egy kiterjesztett partíció 23 logikai meghajtót tartalmazhat (Windows 2000 óta illetve Linux alatt még többet). A logikai partíciók a kiterjesztett partíción egy láncolt listában helyezkednek el, ezért az egyetlen módja annak, hogy kiderítsük, mennyi van belőlük az, hogy végigugrálunk mindegyik kiterjesztett boot rekordon (EBR), míg meg nem találjuk az utolsót.

Kiterjesztett boot rekord

A kiterjesztett boot rekord az MBR-hez hasonló szerkezetű, de többnyire csak a partíciós táblázatot és az AA55H aláírást tartalmazza (az első 446 bájt általában nullákal van feltöltve, hacsak a rendszerprogramozó úgy nem dönt, hogy valamire felhasználja).

Cím	Leírás	Méret (bájt)
000H	Általában kihasználatlan	446
1BEH	Partíciós bejegyzés az aktuális logikai partícióhoz	16
1CEH	Link a következő logikai partícióra	16
1DEH	Általában kihasználatlan	32
1FEH	A boot rekord aláírás (little endian rendben. Vagyis értéke: AA55H)	2



Az ábrán látható területekhez tartozó magyarázat:

1. Az MBR mindig a lemez első szektorában helyezkedik el (0. LBA szektor).
2. A pirossal jelzett terület nem használt rész, erről korábban már volt szó. A Windows Vistáig 62 szektor volt a mérete, azóta 2047 szektor. Attól függetlenül, hogy nem használtnak nevezik, előfordulhat, hogy egyes betöltőprogramok (például a GRUB) azon részei, amik nem férnek be az MBR-be, ide kerülnek.

A 3., 4. és 6. terület az első, második és negyedik partíció, melyeknek a mérete és pozíciója az MBR partíciós táblájának 1., 2. és 4. bejegyzésében található. (Megjegyzendő, hogy régi `fdisk` eszközök nem engednek egynél több elsődleges partíciót létrehozni a kiterjesztett partíció előtt. A Windows 2000 és későbbi rendszerek lemezkezelő eszközeinek már nincs ilyen megszorítása.)

5. Ez a partíciós tábla 3. bejegyzése: egy kiterjesztett partíció. A tartalma külön részletezve van.

7. Ez csak egy nagyobb, a felhasználó által nem partícionált terület.
8. A kiterjesztett partíció első szektora az EBR.
9. Partíción belüli nem használt terület. (Ilyen látható a 11. és 14. számmal jelölve is.) Ez az EBR szektor és annak partíciója között rész hasonló az MBR-t követő területhez. Mérete általában a sávonkénti szektorokkal egyezik meg, tehát alapesetben 62 szektor.
10. Nem használt terület. Ez eredetileg az első logikai partíció volt, de már nem használt terület, mert később törölték. Törléskor az első EBR partíciós bejegyzése törlődik, de a következő EBR-re mutató hivatkozás megmarad.
11. Logikai partíciót definiáló EBR, egy linkkel a következő EBR-re.
12. Érvényes logikai partíció a kiterjesztett partíción belül. Lehet formatált vagy formatálatlan is, típusát a megelőző MBR tartalmazza.
13. Nem használt terület. Ez akár a 12. számmal jelölt logikai partíció része is lehetett korábban, míg egy lemezkezelő eszköz annak méretét le nem csökkentette. De az is lehet, hogy volt itt korábban egy másik logikai meghajtó is, amit töröltek, majd egy lemezkezelő program annak EBR-jét is törölte, a korábbi EBR linkjét pedig módosította (vannak eszközök, amik ezt megcsinálják, de vannak, amik nem).
14. Az kiterjesztett partíció utolsó EBR-je.
15. Egy újabb logikai partíció.
16. Egy kis nem használt terület a kiterjesztett partíció végén.

Elég volt az MBR-ből!

Látható, hogy lényegében több, mint 30 éve nem változott a merevlemezek particionálására szolgáló módszer. (Ha pedig megpróbálták módosítani, az inkompatibilitás okozott gondot.) A 2000-es években az MBR korlátai egyre nyilvánvalóbbak kezdtek lenni:

- a rendszert eredetileg négy partíció létrehozására találták ki. Ennek kicselezésére persze létrejött a kiterjesztett és logikai partíciók rendszere, de az nem az igazi, mert néhány operációs rendszer csak elsődleges partícióról tud indulni és nem egyszerű a partíciók közötti konvertálás.
- az MBR egybájtos partíció típuskódot használ arra, hogy jelezze az operációs rendszernek, mire is való az a partíció. Ez eléggé korlátozza a lehetőségeket és néha ütközések is vannak benne (mint például a 82H, ami egyszerre jelent Linux swap és Solaris disklabel partíciót).
- az MBR eredetileg CHS címzést használt a partíciók címzésére. Ez problémát okozott, ha különböző BIOS-ok különböző módon értelmezték a CHS geometriát. Emellett a CHS címzésnek nagyjából 8 GB volt a felső határa. A 32 bites LBA címek megoldották ezt a problémát, de két különféle rendszer párhuzamos használata már eleve melegágya a problémák jelentkezésének. A modern operációs rendszerek figyelmen kívül hagyják a CHS értékeket. (És így is kell tenniük, ha egy partíció kezdete vagy vége túllép a 8 GB-os határon.)
- a majdnem általánosan használt 512 bájtos szektorméret mellett a 32 bites LBA mutatók 2 TB-os határt jelentenek, ami az MBR egyszerű kiterjesztésével már nem küszöbölhető ki: egyszerűen már nem maradt hely az MBR-ben ilyesmire. A nagy kapacitású meghajtók megjelenésével pedig a probléma kezdett különösen komollyá válni.
- az MBR partíciók érzékenyek a sérülésre. Az elsődleges partíciós tábla kizárólag a lemez első szektorában tárolódik és ha az megsérül,

akkor nagyon nehéz visszaállítani a lemez partícióit. A logikai partíciók pedig egy láncolt lista struktúrában tárolódnak a kiterjesztett partíciókon és ha egy láncszem megsérül, a további logiki partíciókhoz való hozzáférés is elvész.

Vannak persze technikák, amivel ki lehet tolni az MBR életciklusának végét, de ezek egyike sem az igazi. Kell valami a leváltására.

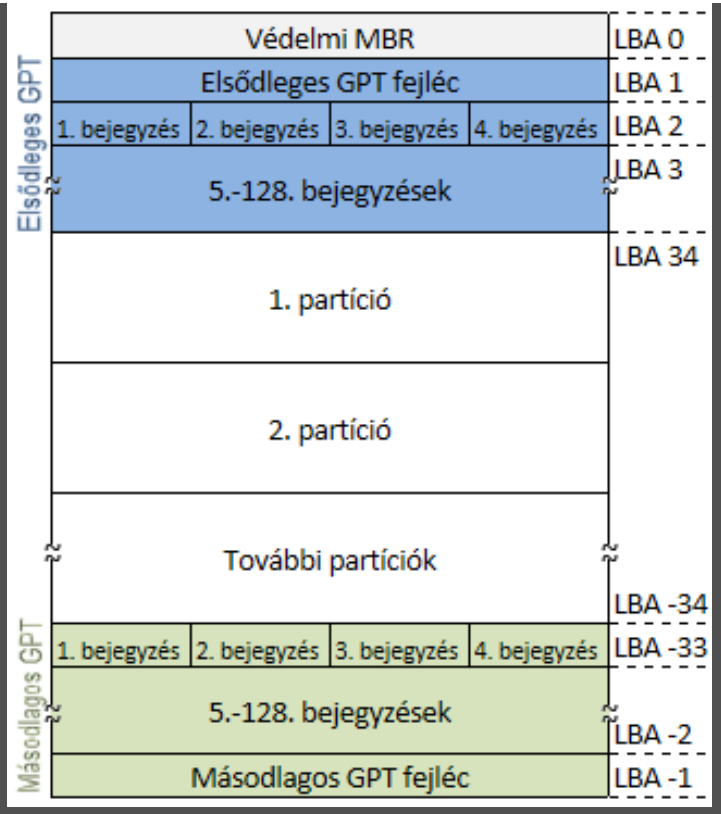
Színre lép a GPT

A történet 1998 tájékán kezdődött, amikor az Intel, a Microsoft és a HP más cégekkel együttműködve elkezdték az első Itanium alapú rendszereket tervezni. A BIOS egészen az IBM PC-k kezdete óta használatban volt, de az Itanium által megcélzott nagygépes környezetben nem tűnt célszerűnek a bevezetése. Egyrészt olyan specifikus hardvereket igényelt, mint a 8254-es időzítő vagy a 8259-es megszakításvezérlő, másrészt 1 MB memóriára volt korlátozva. A 16 bites architektúrája pedig pláne nem illett a 64 bites Itaniumhoz. Más nagygépes környezetek persze már kialakították a saját megoldásaikat, de ezek egyikét sem találták megfelelőnek az Itanium számára, ezért az Intel egyedi megoldást kezdett fejleszteni. Az elképzelést eredetileg IBI-nek hívták (Intel Boot Initiative), de később átkeresztelték EFI-re (Extensible Firmware Interface), ami 1999-ben jelent meg. Az EFI lehetővé tette a magas szintű nyelveken való fejlesztést, megfelelő hardveres absztrakciót adott és biztosította a jövőbeli továbbfejlődés lehetőségét is. Az előnyök olyan meggyőzőek voltak, hogy a Microsoft mellett az egész ipar ezt választotta az Itanium-alapú rendszerek egyedüli boot-mechanizmusának.

Miután pedig az x86 architektúrát is kiterjesztették 64 bitesre, az ipar az EFI alapjára építve elkezdett a Unified EFI (UEFI) rendszeren, mint szabványos pre-boot infrastruktúrán dolgozni. (Az UEFI a BIOS-szal ellentétben egy rendkívül összetett és sokrétű *szabvány*, ami CPU-független architektúrát, sőt CPU-független meghajtóprogramokat definiál. Az UEFI immár a processzor natív módjában fut és tölti be az operációs rendszert, így nem annak kell átváltania a 8088-kompatibilis 16 bites módból 32 vagy 64 bites üzemmódba, mint a BIOS esetén.)

Eredetileg az EFI része volt, de persze az UEFI-be is átkerült az MBR-t leváltó GPT (GUID Partition Table). Ez a technika nemcsak a lemez particionálását, hanem a rendszerindítást is gyökeresen megváltoztatja és visszafelé nem kompatibilis (habár van némi védelmi mechanizmusa a régi rendszerekhez, mint azt látni fogjuk). A GPT-particionált lemezeket ma már minden operációs rendszer tudja kezelni, de ha ilyen merevlemezről (vagy SSD-ről) szeretnénk bootolni, ahhoz már UEFI kell BIOS helyett (az új alaplapokon ma már ez is alapvető).

Az alábbi ábrán látható egy GPT-vel partícionált lemez felépítése.



Az ábra egy olyan GPT sémát mutat, ahol minden logikai blokk 512 bájtos méretű és minden partíciós bejegyzés 128 bájtos. A negatív LBA címek a lemez végétől értendőek.

GPT esetén a nulladik LBA szektor az ún. védelmi MBR (protective MBR). A visszamenőleges kompatibilitást annyiban biztosítja, hogy úgy van létrehozva, hogy a régi lemezkezelő eszközök ne sérthessék meg véletlenül a GPT-t. Ezért is nevezik védelmi MBR-nek. Ebben egyetlen, olyan méretű partíció van definiálva EEH típussal, hogy magába foglalja az egész GPT meghajtót. (Az egész itt azt jelenti, hogy a meghajtónak legfeljebb akkora részét, amekkorát az MBR kezelni tud. Ha a meghajtó mérete nagyobb a 32 bites LBA által elérhetőnél, akkor MBR-rel az ezen felüli rész úgyis elérhetetlen marad.) A GPT-t nem kezelő operációs rendszerek és eszközök ezért általában úgy látják, hogy egy ismeretlen típusú partíció van a lemezen, ami ezen kívül üres területet már nem is tartalmaz és ezért nem engedik módosítani a lemezt, hacsak a felhasználó kifejezetten nem kéri ezt. A GPT-t kezelő operációs rendszerek pedig ellenőrizhetik is a védelmi MBR-t és ha abban a partíció típusa nem EEH, vagy több partíció is definiálva van benne, akkor visszautasíthatják a partíciós tábla módosítását.

Mivel az MBR 512 bájtos szektorra lett kitalálva és manapság már ennél nagyobb szektorokkal dolgozó meghajtók is vannak, ilyen esetben az MBR-ben lévő extra terület általában kihasználatlan marad.

Azoknál az operációs rendszereknél, amik a GPT-alapú meghajtóknál a BIOS-on keresztüli boot-ot támogatják (UEFI helyett), az első szektort ki lehet használni olyan betöltő kódnak, ami felismeri a GPT partíciókat.

A szektorméretttől függetlenül a GPT fejléc (header) mindig az 1. LBA címen található. Ez tartalmaz egy mutatót a tulajdonképpeni partíciós táblára. Az UEFI szabvány kiköti, hogy a szektorméretttől függetlenül legalább 16384 bájt legyen lefoglalva a partíciós táblának. 512 bájtos szektorokat használó lemezen 16384 bájtos partíciós táblával és minimálisan 128 bájtos bejegyzésekkel a lemez első felhasználható szektora az LBA 34-es címen van. Ha a partíciós tábla a 16384 bájtos minimális méretet használja és a bejegyzések az alapértelmezett 128 bájtos méretűek, akkor 128 partíciót lehet definiálni.

A globálisan egyedi azonosító (GUID) egy egyedi szám, amit a szoftverekben azonosítóként használnak. Általában egy 128 bites érték amit 32 hexadecimális számként jelenítenek meg, csoportonként kötőjellel elválasztva, például:

78DE2984-FEFA-A856-BCD1-7895423574AA

A GUID-ot akár véletlenszámokból is lehet generálni, de ebben az esetben általában 122 véletlenszám-bitet és 6 fix bitet tartalmaz, amelyek jelzik, hogy ez egy véletlenszám-GUID. Az ilyen egyedi azonosítók teljes számossága 2^{122} (nagyjából $5{,}3*10^{36}$). Ez a szám olyan nagy, hogy elenyészően kicsi annak valószínűsége, hogy egy értéket véletlenül kétszer generálnak, bár más GUID verzióknak ettől eltérő egyediségi feltételeik is lehetnek. Van ami garantálja az egyediséget, de van, aminél felmerülhet duplikálás. Ha az egyszerűség kedvéért egységes valószínűséget tételezünk fel, akkor annak valószínűsége, hogy duplikálás jön létre nagyjából 50% lenne akkor, ha a Földön 2014-ben minden ember kapna 600 millió GUID-ot.

A GUID partíciós táblán kívül ezt használja például a Microsoft a COM osztályok és interfészek azonosítására vagy például a Second Life nevű számítógépes játék a világban lévő minden eszköz azonosítására.

Egyik GUID generáló algoritmus (amint látni fogjuk) a felhasználó hálózati kártyájának MAC címét használja fel a számjegyek utolsó csoportjának alapjaként (a többi bithez pedig az aktuális időpontot). Ez azt jelenti, hogy egy ilyen GUID-dal azonosított dokumentumot vissza lehet követni addig a számítógépig, ahol azt létrehozták. Ezt a biztonsági rést használta ki annak idején a Melissa vírus.

Persze bizonyos GUID-ok megjelenhetnek újra és újra. A GPT esetén nem fordulhat elő, hogy több, mint egy lemeznek ugyanaz legyen a lemez GUID-ja, vagy egynél több partíciónak ugyanaz legyen az egyedi partíció GUID-ja, de több partíció használhatja ugyanazt a partíció típus GUID-ot.

Az EFI specifikáció a GUID formátumot a következőképpen adja meg:

Offszet (bájt)	Hossz (bájt)	Leírás
00H	4	Az időbélyeg alsó mezője
04H	2	Az időbélyeg középső mezője
06H	2	Az időbélyeg felső mezője megszorozva a verziószámmal

08H	1	Az órajelszekvencia értékének felső mezője megszorozva a variánssal
09H	1	Az órajelszekvencia alsó mezője
0AH	6	A térbelileg egyedi csomópont azonosító. Ez lehet bármilyen hálózati kártyától vett IEEE 802 cím. Ha a rendszerben nincs hálózati kártya, akkor véletlenszám-generátor.

Ez az öt csoportra osztott ábrázolással egy kicsit trükkös eredményt ad, ugyanis az első három csoport megfelel a specifikáció első három részének és little-endian módon tárolódik a lemezen. A negyedik csoport nem kétbájtos érték, hanem két egybájtos, tehát ott valójában már nem szabad little-endianként felcserélni ezeket, amikor vizsgáljuk az értéket. Az utolsó csoport pedig egy hálózati kártya (vagy véletlenszám) generátor által adott érték, ami megintcsak nem little-endian rendben tárolódik.

Az EFI specifikáció 64 bites idő mezőt definiál a GUID-hoz, ami tartalmaz egy 60 bites UTC időbélyeget. Ez 1582. október 15.-től 100 nanoszekundumokat számol (a keresztény naptárreform időpontja). Ez az érték nem fog túlcsordulni egészen 3400-ig. Az EFI kidolgozói ennek alapján fontosnak tartották megjegyezni a specifikációban, hogy egy esetleges jövőbeli specifikáció frissítésnek majd kezdeni kell valamit a 3400. év problémájával.

Az első 4096 bájtos szektorméretet használó meghajtók még 512 bájtos szektorokat jelentettek az operációs rendszer felé, ezért teljesítményveszteséggel járt, ha a meghajtó belső 4 KB-os szektorhatára nem esett egybe a nagyon sok fájlrendszer és operációs rendszer által használt 4 KB-os logikai blokkokkal, klaszterekkel vagy virtuális memória lapokkal. Ilyenkor egyetlen nem illeszkető 4 KB-os adatcsomag írásához két olvas-módosít-ír műveletre volt szükség. Ilyen illeszkedési hiba már a rendszerbe van kódolva, ha az első partíció közvetlenül a GPT után indul, mert a következő blokk az LBA 34, de a következő 4 KB-os határ az LBA 40-nél kezdődne. A GPT fejléc és partíciós tábla a lemez elejére és a lemez végére is felmásolódik, a két példány nagyobb biztonságot jelent.

Partíciós tábla fejléc

A fejléc definiálja a lemezen használt blokkokat és a partíciós táblát alkotó bejegyzések méretét és számát is. Ez tartalmazza a lemez egyedi azonosítóját (GUID) valamint rögzíti saját méretét és helyét is (ez mindig LBA 1). Rögzíti a másodlagos GPT fejléc és partíciós tábla helyét is; mindig a lemez utolsó szektoraiban. Emellett tartalmaz egy CRC32 ellenőrző összeget saját magára és a partíciós táblára vonatkozóan, amit a fimver, a rendszerbetöltő és/vagy az operációs rendszer is ellenőrizhet a rendszer betöltésekor. Alacsony szintű hexa szerkesztők viszont emiatt nem használhatók a GPT tartalmának módosítására, mert ilyen módosítás esetén az ellenőrző összeg érvénytelenné válik (hacsak nem módosítják azt is). Hibás ellenőrzőszám esetén rendszer-hejreállító szoftver felülírhatja az elsődleges GPT-t a lemez végén tárolt biztonsági másolattal. Ha mindkét GPT érvénytelen ellenőrzőszámot tartalmaz, akkor sok rendszerbetöltő és operációs rendszer megtagadja a rendszerindítást.

Offszet	Hossz (bájt)	Leírás

00H	8	Aláírás: "EFI PART" (45H 46H 49H 20H 50H 41H 52H 54H vagy 5452415020494645H little-endian gépeken)
08H	4	Revízió. 1.0-s GPT verziókhoz 00H 00H 01H 00H
0CH	4	A fejléc mérete little endian formában. Bájtokban 5CH 00H 00H 00H vagyis 92 bájt.
10H	4	A fejléchez tartozó CRC32. Számítása a 0. bájttól a fejléc méretéig tart, a számításnál ezen mező értékét 0-nak véve.
14H	4	Fenntartott, az értéke nulla.
18H	8	Az aktuális LBA (ennek a fejlécnek a címe).
20H	8	Biztonsági mentés LBA-ja (a másik fejléc címe).
28H	8	A partíciók számára használható első LBA cím (az elsődleges partíciós tábla utolsó LBA címe + 1).
30H	8	A partíciók számára használható utolsó LBA cím (a másodlagos partíciós tábla első LBA-jának címe - 1)
38H	16	A lemez GUID azonosítója
48H	8	A partíciós táblázat kezdő LBA-ja (az elsődleges partíciós táblázat esetén mindig 2).
50H	4	A partíciós bejegyzések száma
54H	4	Egy partíciós bejegyzés mérete (általában 80H vagyis 128)
58H	4	A partíciós tömb CRC32 ellenőrző összege
5CH		Fenntartott, a szektor végéig nullákat tartalmaz

Az aktuális és a biztonsági másolat fejléc LBA-jának mindig a második és az utolsó szektorba kell esnie. A másodlagos fejléc a lemez végén saját partíciós táblát ad meg, ami közvetlenül előtte helyezkedik el. Az elsődleges fejlécnek pedig az LBA 1 címen kell lennie, ebből az következik, hogy nem szükséges fizikailag azonnal követnie az MBR-t. A 4 KB-os szektorméretű lemezeken például a lemez kezdetétől 4096 bájtra kezdődik és ezért kihasználatlan terület marad közte és az MBR között. 512 bájtos lemezeknél viszont már közvetlenül az MBR-t követi, mert ezeknél az LBA 1 cím oda mutat.

Partíciós bejegyzések

A fejléc után következik a partíciós táblázat, legalább 128 bájtos bejegyzésekkel. Egy bejegyzésből az első 16 bájt a partíció típusának GUID-ját tartalmazza. A második 16 bájt a partíció egyedi GUID-ját tárolja, ezt követi a partíció kezdetének és végének 64 bites LBA címe, majd a partíció attribútumai és neve. A lemez és a partíció GUID-ja nem teszi lehetővé teljes meghajtók szektoronkénti másolását, hiszen akkor ezen azonosítók onnantól kezdve már nem lennének egyediek. A partíciók másolása viszont továbbra is lehetséges.

Offszet	Hossz (bájt)	Leírás
00H	16	A partíció típusának GUID azonosítója. Kicsit trükkös, mert az első nyolc bájt (az azonosító ábrázolásának első három csoportja) little-endian rendben van, a második nyolc bájt pedig big-endian rendben.
10H	16	Egyedi partíciós GUID
20H	8	A partíció első LBA-ja (little endian sorrendben)
28H	8	A partíció utolsó LBA-ja (általában páratlan)
30H	8	Attribútumok (a 60. bit például csak olvashatót jelent)
38H	72	A partíció neve (36 UTF-16 karakter)

A számításokban a szektorméret nem kell, hogy szükségszerűen be legyen drótozva 512 bájtra, tehát egy szektorban négy bejegyzésnél több is lehet, a jövőbeli sokkal nagyobb bejegyzések lehetőségét szem előtt tartva pedig lehetséges az is, hogy egy szektor egy bejegyzésnek csak egy részét tartalmazza. Az első két szektort kivéve a GPT specifikáció csak az adatstruktúra méretét és szervezését írja le, azt nem, hogy ez a lemezen hány szektorban tárolódjon.

Ha a partíció típus GUID értéke 00000000-0000-0000-0000-000000000000, az azt mutatja, hogy a bejegyzés nem tartalmaz partíció definíciót.

A 64 bites attribútum mező fel van osztva egy 48 bites, minden partíciótípusra érvényes részre és egy 16 bites, típus-specifikus részre, ezek bővebben az alábbi táblázatban láthatóak.

Bit	Jelentés
0	Rendszerpartíció (a partícionáló eszközöknek úgy kell hagyni ezt a partíciót, ahogy van)
1	Az EFI firmvernek ezen partíció tartalmát figyelmen kívül kell hagyni, ne olvasson róla
2	Régi BIOS által bootolható (megegyezik az MBR partíciós táblájában lévő státusz mező aktív, 7. jelzőbitjével)
3-47	Jövőbeli használatra fenntartott
48-63	Az adott partíció típus saját használatú bitjei

A Microsoft a következőképp definiálja egy adatpartícióhoz a típusspecifikus attribútumokat:

Bit	Jelentés
60	Csak olvasható
61	Árnyékmásolat

62	Rejtett
63	Nem lehet automatikusan csatolni (például meghajtójelet társítani hozzá)

A GPT lemezek esetén az MBR-hez hasonló módon szintén el lehet rejteni adatokat (még akkor is, ha a Microsoft ennek ellenkezőjét állítja több MSDN cikkében), habár a GPT eleve lehetővé teszi rejtett partíció attribútumok használatát is. De a definiált partíciók közötti rések vagy az utolsó partíció utáni kihasználatlan terület (a másodlagos GPT előtt) itt is létező lehetőség. A szabvány implementálásától függően pedig akár a GPT fejléc módosításával is lehetőség nyílik adatok elrejtésére. A partíciók számára használható területeket jelölő kezdő és vég LBA címek módosítása például tipikusan egy ilyen módszer. De akár a partíciós bejegyzések méretének növelésével is lehet felhasználható területet nyerni a partíciós táblában csakúgy, mint a fenntartottként jelölt területek kihasználásával. Ez utóbbi persze nem jelent túl sok felhasználható plusz helyet.

Hibrid MBR

Egy szabványkövető GPT lemez tehát tartalmaz egy védelmi MBR-t, amiben egyetlen partíció van definiálva, EEH típuskóddal a lemez teljes méretére (vagy a lehető legnagyobb LBA méretre, ha a lemez azt már túlhaladja). A hibrid MBR egy normál és egy védelmi MBR kombinációja. Ez is tartalmaz egy EEH típusú partíciót, de definiál legfeljebb három elsődleges partíciót is, amik ugyanarra a területre mutatnak, mint legfeljebb három GPT partíció.

Nézzük például azt az esetet, amikor van egy Macintosh számítógép dual-boot Mac OS X-szel és egy Windows-zal. Az OS X jól megvan a GPT-vel, tehát lehet használni a GPT partíciókat, de a Windows mondjuk egy régebbi verzió és az nem tudja kezelni. Ezért elsőként definiálni kell a GPT partíciókat (a Windows partíciókat is beleértve), majd módosítani kell a védelmi MBR-t, hogy az EEH típusú partíció kisebb legyen, mint a szabvány által megkövetelt és bekerüljön mellé még a többi (legfeljebb három) partíció is a Windowsnak. Ezek ugyanazokra mutatnak, mint amik a GPT-ben is definálva vannak. Ezután telepíteni lehet a Windowst ezekre a hibrid partíciókra. Hibrid MBR-ekre általában csak BIOS-alapú gépeknél van szükség.

Mivel azonban a hibrid MBR nem része a szabványnak, ez a buherálás jelentős kockázattal jár, hiszen minden egyes rendszer vagy lemezkezelő eszköz egyedi módon értelmezheti. Ezért nem is ajánlott.

Operációs rendszer támogatás

A legtöbb modern operációs rendszer már támogatja a GPT partíciók elérését, de a boothoz természetesen UEFI kell. Windows esetén a támogatás a következőképp alakul:

Verzió	Platform	Olvasás vagy írás	Boot
Windows XP	32 bit	nem	nem
Windows Server 2003	32 bit	nem	nem
Windows Server 2003 SP1	32 bit	igen	nem

Windows Vista	32 bit	igen	nem
Windows Server 2008	32 bit	igen	nem
Windows 7	32 bit	igen	nem
Windows 8	32 bit	igen	UEFI-vel
Windows 8.1	32 bit	igen	UEFI-vel
Windows 10	32 bit	igen	UEFI-vel
Windows XP Professional x64 Edition	x64	igen	nem
Windows Server 2003	x64	igen	nem
Windows Server 2003	IA-64	igen	igen
Windows Vista	x64	igen	UEFI-vel
Windows Server 2008	x64	igen	UEFI-vel
Windows Server 2008	IA-64	igen	igen
Windows 7	x64	igen	UEFI-vel
Windows Server 2008 R2	x64	igen	UEFI-vel
Windows Server 2008 R2	IA-64	igen	igen
Windows 8	x64	igen	UEFI-vel
Windows Server 2012	x64	igen	UEFI-vel
Windows 8.1	x64	igen	UEFI-vel
Windows 10	x64	igen	UEFI-vel

A maximális partíció- és lemezméret is operációs rendszerfüggő. A Windows XP és a Windows Server 2003 eredeti kiadása csak 2 TB-os fizikai lemezeket tud kezelni, függetlenül a partícióktól. A Windows Server 2003 SP1 és a Windows XP x64 valamint minden újabb verzió esetén a támogatott nyers partícióméret már 18 exabájt, bár a Windows fájlrendszerei esetén jelenleg a legnagyobb használható méret 256 terabájt.

EFI rendszerpartíció

Az EFI rendszerpartíciót (EFI System partition, ESP) az UEFI-t használó számítógépek tudják rendszerindításra használni. Az UEFI firmver képes arra, hogy közvetlenül megértse a GPT partíciós táblát valamint a FAT fájlrendszert és betöltsön, aztán futtasson az EFI rendszerpartíción lévő fájlokban tárolt programokat. Az ESP-t emiatt FAT fájlrendszerrel kell formattálni, de ezt a FAT rendszert már az UEFI specifikáció is tartalmazza (tehát nem támaszkodik külső szabványra).

Az ESP típus GUID-ja:

C12A7328-F81F-11D2-BA4B-00A0C93EC93B

(Ezt egy GUID-dekódolával visszafejtve kiderül, hogy 1999. április 21-én generálták.)

MBR partícionált lemezek is tartalmazhatnak ESP-t, ezeknél EFH a típuskód.

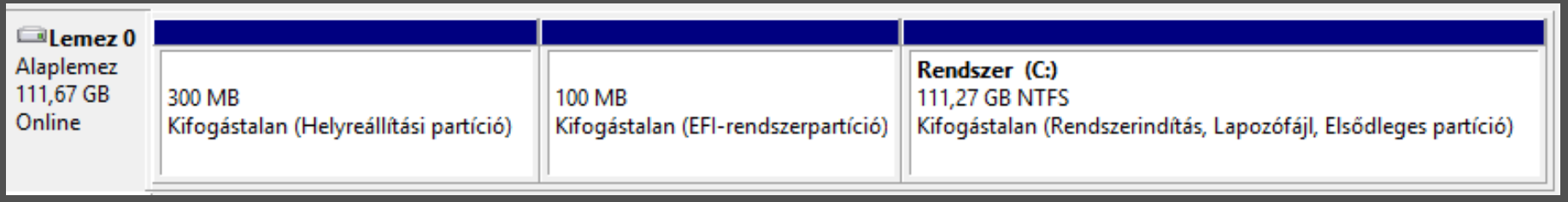
Az ESP rendszerbetöltő programokat tartalmaz mindegyik telepített operációs rendszerhez (amik egyéb partíciókon vannak ugyanazon vagy akár másik fizikai tárolóegységen). Emellett tartalmaz eszközmeghajtókat a gépben lévő hardverekhez. Windows esetén itt van például az NTLDR, a HAL (hardveres absztrakciós réteg), a boot.txt. A Microsoft azt is ajánlja, hogy az ESP a lemezen az első partíció legyen. Ennek ugyan nincs architekturális követelménye, de például lehetetlen átméretezni egyéb adatpartíciókat, ha az ESP azok között helyezkedik el.

Ez a partíció emellett tartalmazhat akár parancssori eszközöket az EFI alrendszer és a firmver közvetlen eléréséhez is.

Az UEFI a visszamenőleges kompatibilitás érdekében fenntartja az ESP első szektorát olyan programnak, ami biztosítja ezt a kompatibilitást, tehát emulálja a VBR-t. A régi, BIOS-alapú gépeken a partíció első szektorát az MBR kód a memóriába tölti és végrehajtja, de ez UEFI már nem teszi ezt meg, hacsak nincs beállítva neki, hogy a régi BIOS módban induljon.

Élő példa

Vizsgáljuk meg most a Samsung által 120 gigásnak mondott, valójában 111,67 gigás (vagy legalábbis azt hittem) SSD-men lévő Windows 8.1 GPT szerkezetét! A partíciók a lemezkezelőben így néznek ki:



Az fsutil megmondta, hogy a fizikai szektorméret 512 bájt, így ezzel számolunk a továbbiakban.

A 0. szektor a védelmi MBR:

```
       0|  1|  2|  3|  4|  5|  6|  7|  8|  9|  A|  B|  C|  D|  E|  F|0123456789ABCDEF
000   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
010   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
020   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
030   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
040   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
050   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
060   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
070   00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|
```

```
080  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
090  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
0A0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
0B0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
0C0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
0D0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
0E0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
0F0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
100  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
110  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
120  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
130  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
140  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
150  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
160  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
170  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
180  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
190  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
1A0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
1B0  00| 00| 00| 00| 00| 00| 00| 00| B3| 96| 8A| D2| 00| 00| 00| 00|________________
1C0  02| 00| EE| FF| FF| FF| 01| 00| 00| 00| FF| FF| FF| FF| 00| 00|________________
1D0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
1E0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
1F0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 55| AA|______________U_
```

Hát ez a kód bizony csupa nulla! A lemez aláírás viszont ki van töltve 01B8H-n. Persze ha egy ilyen háttértárat egy Windows XP-s gépbe raknánk, az felülírná ezt. A partíciós táblában lévő egyetlen bejegyzés (1BEH-1CDH0) típusa EEH, vagyis védelmi partíció, az első bájtja 00H vagyis nem bootolható. Érdekes módon a partíció utolsó szektorának LBA címe is a legnagyobb 32 bites értéket tartalmazza (még úgy is, hogy a lemez jóval kisebb, mint 2 TB). A védelmi MBR tehát az aláírást kivéve pontosan ugyanaz minden egyes gépen, amit Windows 7-tel vagy 8-cal telepítettünk. Érdemes megjegyezni azonban, hogy ez nem felel meg az UEFI specifikációnak, mert az azt mondja, hogy a partíció méretének a lemez mérete - 1 kellene lennie, vagy ha a lemez mérete túl nagy, akkor a legnagyobb 32 bites értéknek. (A Linux vagy Mac OS alatt inicializált lemezek már követik az UEFI előírásait ebben a kérdésben.)

Az 1. szektor a GPT fejléc:

```
      0|  1|  2|  3|  4|  5|  6|  7|  8|  9|  A|  B|  C|  D|  E|  F|0123456789ABCDEF
000  45| 46| 49| 20| 50| 41| 52| 54| 00| 00| 01| 00| 5C| 00| 00| 00|EFI_PART____\___
```

```
010  DD| B7| AD| AE| 00| 00| 00| 00| 01| 00| 00| 00| 00| 00| 00| 00|________________
020  AF| 4B| F9| 0D| 00| 00| 00| 00| 22| 00| 00| 00| 00| 00| 00| 00|_K______"_______
030  8E| 4B| F9| 0D| 00| 00| 00| 00| 1F| 66| 3A| A5| DA| 04| 12| 4D|_K_______f:____M
040  AE| 01| A0| A3| CF| 45| A5| F7| 02| 00| 00| 00| 00| 00| 00| 00|_____E__________
050  80| 00| 00| 00| 80| 00| 00| 00| 05| 8A| D5| 1C| 00| 00| 00| 00|________________
```

Bár itt már csak 5FH-ig látható a szektor tartalma, de az olvasó elhiheti nekem, hogy ezután már végig 0 a bájtok értéke. A specifikációnak megfelelően a szektor az "EFI PART"aláírással kezdődik. A szektor további tartalma:

- 08H: négybájtos revízió azonosító, eszerint ez 1.0-s GPT.
- 0CH: 4 bájt a fejléc mérete: 5CH (a lemezen little-endian!), vagyis 92 bájt.
- 10H: CRC32 ellenőrző kód
- 14H: egy fenntartott 0 bájt, ugorjunk
- 18H: az aktuális 64 bites LBA, vagyis a fejléc címe. Értéke 1, tehát jó helyen járunk.
- 20H: a biztonsági mentés 64 bites LBA-ja: 0DF94BAFH, vagyis 234441647, ezt majd megnézzük.
- 28H: a partíciók számára használható terület első LBA-ja: 22H, vagyis 34
- 30H: a partíciók számára használható terület utolsó LBA-ja: 0DF94B8EH, vagyis 234441614
- 38H: a lemez GUID azonosítója
- 48H: a partíciós táblázat kezdő LBA-ja; itt 2. Elsődleges partíciós tábla esetén ennek kell lennie.
- 50H: a partíciós bejegyzések száma, ami 128 (ebből itt nyilván nincs kihasználva mind)
- 54H: egy partíciós bejegyzés mérete 128 bájt (80H), ez is szokványos
- 58H: a partíciós tábla CRC32 ellenőrző összege, biztos jó
- 5CH-tól pedig jön a fenntartott rész nullákkal kitöltve (és mivel ez a 93. bájt, tehát valóban 92 bájtos a fejléc)

Álljunk csak meg egy pillanatra! A lemezkezelő azt mondja, hogy az SSD 111,67 GB-os, a tulajdonságait megnyitva még pontosabb értékkel szolgál a Kötetek fülön: 114345 MB. A biztonsági mentés LBA-ja a GPT fejléc szerint 0DF94BAFH, ami átszámítva már a 114473 MB után mutat. Ez a lemezkezelő adatával összevetve túl van az SSD kapacitásán! Valami nincs rendben a lemezkezelő által mutatott értékkel. De lesznek még itt furcsaságok.

A 2. szektortól kezdődik a partíciós tábla.

Tudjuk, hogy 128 darab 128 bájtos bejegyzésünk lehet, ami 16384 bájt, ez 32 szektor. A fejléc és az MBR beleszámításával tehát első használható partíciónak a 34-es LBA cím azt jelenti, hogy nincs semmi titkos hely az SSD-n errefelé. A partíciós táblában viszont nem 3, hanem 4 darab 128 bájtos érvényes bejegyzés található (az első szektorban), a többi terület nulla értékű bájtokkal van tele. A lemezkezelő még három partíciót mutatott. Hogy néznek ki a GPT-ben a bejegyzések?

```
      0|  1|  2|  3|  4|  5|  6|  7|  8|  9|  A|  B|  C|  D|  E|  F|0123456789ABCDEF
000  A4| BB| 94| DE| D1| 06| 40| 4D| A1| 6A| BF| D5| 01| 79| D6| AC|______@M_j___y__
010  90| 22| 31| 24| DF| 3F| 0A| 42| A8| 98| 84| 56| F3| FA| A9| AF|_"1$_?_B___V____
020  00| 08| 00| 00| 00| 00| 00| 00| FF| 67| 09| 00| 00| 00| 00| 00|_________g______
```

```
030  01| 00| 00| 00| 00| 00| 00| 80| 42| 00| 61| 00| 73| 00| 69| 00|________B_a_s_i_
040  63| 00| 20| 00| 64| 00| 61| 00| 74| 00| 61| 00| 20| 00| 70| 00|c___d_a_t_a___p_
050  61| 00| 72| 00| 74| 00| 69| 00| 74| 00| 69| 00| 6F| 00| 6E| 00|a_r_t_i_t_i_o_n_
060  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
070  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
080  28| 73| 2A| C1| 1F| F8| D2| 11| BA| 4B| 00| A0| C9| 3E| C9| 3B|(s*_____K___>_;
090  7E| 9B| C1| F6| 22| CA| 4B| 48| A3| A4| 5E| C0| 92| A4| 61| 63|____"_KH__^___ac
0A0  00| 68| 09| 00| 00| 00| 00| 00| FF| 87| 0C| 00| 00| 00| 00| 00|_h______________
0B0  00| 00| 00| 00| 00| 00| 00| 80| 45| 00| 46| 00| 49| 00| 20| 00|________E_F_I___
0C0  73| 00| 79| 00| 73| 00| 74| 00| 65| 00| 6D| 00| 20| 00| 70| 00|s_y_s_t_e_m___p_
0D0  61| 00| 72| 00| 74| 00| 69| 00| 74| 00| 69| 00| 6F| 00| 6E| 00|a_r_t_i_t_i_o_n_
0E0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
0F0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
100  16| E3| C9| E3| 5C| 0B| B8| 4D| 81| 7D| F9| 2D| F0| 02| 15| AE|____\__M_}_-____
110  15| D1| CD| 26| DE| D7| 48| 42| A1| F3| 6B| CE| 47| 8E| CC| EA|___&__HB__k_G___
120  00| 88| 0C| 00| 00| 00| 00| 00| FF| 87| 10| 00| 00| 00| 00| 00|________________
130  00| 00| 00| 00| 00| 00| 00| 80| 4D| 00| 69| 00| 63| 00| 72| 00|________M_i_c_r_
140  6F| 00| 73| 00| 6F| 00| 66| 00| 74| 00| 20| 00| 72| 00| 65| 00|o_s_o_f_t___r_e_
150  73| 00| 65| 00| 72| 00| 76| 00| 65| 00| 64| 00| 20| 00| 70| 00|s_e_r_v_e_d___p_
160  61| 00| 72| 00| 74| 00| 69| 00| 74| 00| 69| 00| 6F| 00| 6E| 00|a_r_t_i_t_i_o_n_
170  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
180  A2| A0| D0| EB| E5| B9| 33| 44| 87| C0| 68| B6| B7| 26| 99| C7|______3D__h__&__
190  66| 14| 96| D9| 5C| 85| 5E| 40| 94| 9A| 57| 93| 63| 6C| 82| 2E|f___\_^@__W_cl_.
1A0  00| 88| 10| 00| 00| 00| 00| 00| FF| 47| F9| 0D| 00| 00| 00| 00|________G_______
1B0  00| 00| 00| 00| 00| 00| 00| 00| 42| 00| 61| 00| 73| 00| 69| 00|________B_a_s_i_
1C0  63| 00| 20| 00| 64| 00| 61| 00| 74| 00| 61| 00| 20| 00| 70| 00|c___d_a_t_a___p_
1D0  61| 00| 72| 00| 74| 00| 69| 00| 74| 00| 69| 00| 6F| 00| 6E| 00|a_r_t_i_t_i_o_n_
1E0  00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00| 00|________________
```

Elemezzük ki ezeket a bejegyzéseket!

- a móka a partíció típusának GUID azonosítójával kezdődik, ami 16 bájt. Ezt kicsit nehézkesebb kibogarászni, de azért sikerül: DE94BBA4-06D1-4D40-A16A-BFD50179D6AC. Ez a helyreállítási partíció típusa.
- ezt követi 10H címen az egyedi partíciós GUID 16 bájtja
- 20H címen jön a partíció első LBA-ja: 800H. Már tudjuk az MBR elemzéséből, miért ennyi. Itt tehát szintén kimarad egy kis üres terület a partíciós tábla és az első partíció között.
- 28H címen jön a partíció utolsó LBA-ja: 0967FFH, vagyis 616447. A kezdetet figyelembe véve a méret bizony 300 MB.
- 30H címen jönnek az attribútumok (8000000000000001H)

- 38H-n jön a partíció neve: "Basic data partition"

A második partíciós bejegyzés 80H címtől kezdődik:

- a C12A7328-F81F-11D2-BA4B-00A0C93EC93B azonosító alapján ez egy EFI rendszerpartíció
- az első szektor LBA-ja 96800H, vagyis egyből a korábbi után következik
- az utolsó szektor LBA-ja 0C87FFH, tehát a partíció mérete 100 MB
- attribútum: 8000000000000000H
- a név: "EFI system partition"

A harmadik partíciós bejegyzés 100H címtől kezdődik:

- ez valamiféle Microsoft reserved partition (MSR) típust jelöl (E3C9E316-0B5C-4DB8-817D-F92DF00215AE)
- 0C8800H címtől 1087FF címig tart, 128 MB-os
- attribútum: 8000000000000000H
- Microsoft reserved partition a neve

Ez nem látszott a Lemezkezelőben! Ezek szerint van egy partíció a lemezen, amit a Windows 8.1 telepítője létrehoz, de a rendszer nem mutatja meg. Nos a Microsoft szerint ez egy olyan partíció, amit a rendszer későbbi operációs rendszerkomponensek általi használatra tart fenn és a korábban az MBR esetén rejtett szektorokban tárolt programok kerülnek ide.

Végül van még egy negyedik partíció, ahol maga a rendszer helyezkedik el:

- EBD0A0A2-B9E5-4433-87C0-68B6B72699C7, vagyis alap adatpartíció típusú.
- 108800H címtől a 0DF947FFH címig, vagyis 111,27 GB méretű
- attribútum: 0000000000000000H
- "Basic data partition" néven

Így tehát megvan minden, amit kerestünk, de az attribútumokat azért még érdemes megnézni. A fenti négy partíción háromféle attribútumot lehet megfigyelni. A helyreállítási partíció nulladik bitje 1, ami rendszerpartíciót jelent, ez az egyetlen ilyen. Az első három partíció legnagyobb helyiértékű bitje 1-es, ami a korábbi tábázat alapján azt jelenti, hogy a meghajtó alapértelmezetten nem kap betűjelet. Az utolsó adatpartíció semmilyen speciális tulajdonsággal nem rendelkezik. Viszont a méretekkel megint gond van. Az utolsó partíció végének címe 0DF947FFH, ami átszámítva szintén túlmutat a Windows által mutatott meghajtóméreten: 114473. MB-ra. A partíció kilóg az SSD-ből?

Biztonsági másolat

A GPT egyik előnye az MBR-el szemben, hogy a szabvány előír egy biztonsági másolatot a tárolóeszköz végén, ami akkor használható, ha az elsődleges partíciós tábla megsérül. A példában a partíciók számára használható terület végének címe: 0DF94B8EH, bár az utolsó partíció már a 0DF947FFH címen végetér. A kettő között tehát van egy 38FH szektor méretű (455 KB) üres terület, mégpedig azért, mert a Windows telepítő megabájtra kerekíti a partíciókat. A biztonsági mentés címéig pedig megint van egy kis üres rész (33 szektornyi), mert az a 0DF94BAFH címen kezdődik.

A probléma a biztonsági másolat beolvasásánál jelentkezett. A szektorok olvasására használt egyszerű HDHacker program a biztonsági másolat fejléce helyett értelmezhetetlen adatokat mutatott. Az elsődleges GPT fejléc hibás lenne? De a már említett méretbeli probléma is gyanús volt és úgy döntöttem, próbát teszek egy linuxos live boot CD-vel. Parancssori eszközökkel a Linux által a meghajtóra megjelenített szektorszám nagyobb méretet adott, mint amit a Lemezkezelőben látunk: így már 114473,46 MB volt az SSD mérete (ez már közelebb van a Samsung által 10-es számrendszeri megabájtban adott méret kihasználásához is). Ha pedig ebből levonom a 128 MB-ot (a rejtett MSR partíció mérete), akkor épp megkapom a lemezkezelőben látható értéket: 114345 MB. A Windows 8.1 tehát nemcsak a partíciót rejti el, hanem az általa használt területet is, még a szektorolvasó program elől is! És még mondja a Microsoft, hogy a GPT nem teszi lehetővé a rejtett szektorok használatát!

A Linux segítségével már látszódtak az utolsó szektorok, így megvan a

biztonsági másolat fejléce:

```
      0|  1|  2|  3|  4|  5|  6|  7|  8|  9|  A|  B|  C|  D|  E|  F|0123456789ABCDEF
000  45| 46| 49| 20| 50| 41| 52| 54| 00| 00| 01| 00| 5C| 00| 00| 00|EFI_PART____\___
010  C3| 50| A5| 75| 00| 00| 00| 00| AF| 4B| F9| 0D| 00| 00| 00| 00|_P_u_____K______
020  01| 00| 00| 00| 00| 00| 00| 00| 22| 00| 00| 00| 00| 00| 00| 00|________"_______
030  8E| 4B| F9| 0D| 00| 00| 00| 00| 1F| 66| 3A| A5| DA| 04| 12| 4D|_K_______f:____M
040  AE| 01| A0| A3| CF| 45| A5| F7| 8F| 4B| F9| 0D| 00| 00| 00| 00|_____E___K______
050  80| 00| 00| 00| 80| 00| 00| 00| 05| 8A| D5| 1C| 00| 00| 00| 00|________________
```

A fejléc nagyrészt megegyezik az elsődlegessel, a következő kivételekkel:

- 10H-tól más a CRC32 ellenőrző összeg (hiszen a tartalom is eltér)
- 18H: az aktuális cím nem 1, hanem 0DF94BAFH, ami egyben az SSD utolsó használható címe, vagyis 234441647.
- 20H: itt nem a biztonsági mentés címe van, hanem az elsődleges fejléc címe, ami 1
- 48H: a biztonsági partíciós táblázat kezdőcíme: 0DF94B8F.

5CH-tól pedig a nullákkal feltöltött rész. A partíciós tábla pedig teljesen megegyezik az elsődleges partíciós táblával.

Sipos Róbert (2015)

Az összeállításhoz használt források:

Master Boot Record
Cylinder, head, sector
Logical block addressing
The Starman's Realm
Nagy szektorméretű meghajtók támogatása
A BitLocker meghajtótitkosítás áttekintése

What's a GPT?
Beyond BIOS (Intel, 2010)
GUID Partition table
Windows and GPT FAQ
Forensic Analysis of GPT Disks and GUID Partition Tables (2009)
Extensible Firmware Interface Specification (Intel, 2002)