Az első pillantás az Apple G4-re

Matthew – a Linux-telepítés és a programfordítás kapcsán – az új Apple G4-én szerzett tapasztalatait osztja meg velünk.

Amikor először olvastam arról, hogy az Apple G4-alapú személyi számítógépet tervez, fogalmam sem volt róla, mi is az a G4. Szuperszámítógéphez fogható teljesítmény? Gigaflopos feldolgozási sebesség? Hogyan lehetséges ez? A G4-ben Motorola 7400-as processzor található AltiVec-egységgel. Az AltiVec az új PowerPC processzorokban található vektorműveleteket feldolgozó egység márkaneve. A Motorola a 7410 és 7450-es modelleket is bejelentette, amelyekben L2-es és hatalmas L3-as gyorstár, sebesebb CPU-mag és mélyebb, hétlépcsős utasítás-csővezeték lesz. Az AltiVec javított egészszámos és lebegőpontos műveletfeldolgozó egység. Új 128-bites feldolgozóegységgel rendelkezik, továbbá 32 vektorregiszterrel, és több mint 160 új utasítással, melyek a csővezetékben várakozó adat feldolgozását segítik elő.

A kiszolgáló gép egy Apple kétprocesszoros G4 450 MHz-es PowerPC volt 512 MB RAM-mal, 30 GB-os Quantum Fireball IDE merevlemezzel, MultiDrive-val (ez DVD-R-író, mely az összes egyéb lemezformátumot írja és olvassa), két IEEE 1394-gyel (Firewire-csatoló), 100 Mb/mp ethernettel, és egy csomó USB-csatlakozással. A billentyűzet és az egér egyaránt USB-n keresztül csatlakozott. Az Apple „Új Világ”-ra keresztelte ezt a gépet, és annak ellenére, hogy az elnevezés meglehetősen „marketingszagú”, az Apple a kifejezést azokra a gépeire használja, amelyekben a boot a ROM-programban található (a „Régi Világ” gépeiben ugyanezt a PROM-ban tárolták).

Nem tudom pontosan, hogy milyen indíttatástól vezérelve, de a Yellow Dog Linux-változat mellett döntöttem. Több lehetőség is kínálkozott volna (SuSE, LinuxPPC), de a YDL RedHat-alapú, aminek köszönhetően neve nem csengett teljesen ismeretlenül.

A YDL a Terra Soft Solutions terméke. A cég egy további terjesztéssel is bír, ami a Black Lab Linux nevet viseli, de én inkább a YDL-t ajánlom. Letöltöttem a YDL Champion Server 1.2.1 két CD-jét a Terra Soft egyik tükörkiszolgálójáról. Az első telepítőlemezként szolgál, a második neve pedig „Tasty Morsels” – ezen találhatjuk a vészrendszer ISO-fájlját, valamint néhány további PPC-s programot. Ezen a ponton merült fel a kérdés: hogyan tovább?

Elolvastam a YDL telepítési útmutatóját, amelyben kiderült, hogy yabootra van szükségem (yaboot – yet another boot loader), amit a HPFS-lemezrésze (a Mac félére) kell telepíteni, tehát létre kell hoznom egyet a maces rendszerprogramokkal.

A Mac OS9 újratelepítéséhez, majd a Linux telepítéséhez a következő lépéseket hajtottam végre:

- HFS-lemezrész (4 GB) létrehozása a yabootnak (és az OS9-nek);
- az OS9 újratelepítése gyári CD-ről;
- a CS (Champion Server) 1.2.1 CD-ről a yaboot, a yaboot.conf és a vmlinux.gz rendszermappába történő másolása.

A yaboot.conf nagyon hasonlít a lilo.conf-ra. Amikor a yaboot elindul, nyomjunk TAB-ot, hogy a rendszer kiírja a választható operációs rendszerek nevét – ezek valamelyikét kell beírnunk a promptnál.

Rendben, telepítsünk Linuxot! Helyezd be a YDL CD-t a DVD-meghajtóba és rendszerindulás közben tartsd nyomva C gombot! Így indíthatod CD-ről a rendszert. Ekkor megjelenik a YDL telepítőjének képernyője. A legtöbb esetben követheted a YDL telepítési útmutatójában leírtakat, de azért adnék egy jó tanácsot a lemezterületekről (hacsak nem volt már Linux a Maceden) – létre kell hoznod őket. Ezúttal már nem *ext2* lemezterületeket létesítesz, hanem *Apple_UNIX_SVR2*-típusúakat. A program neve is módosult egy kicsit: nem `fdsik`, hanem `pdisk`. A `p` paranccsal a lemezterületeket itt is megjelenítheted. Ha követted a tanácsaimat, kilenc lemezrészt kell látnod. Ezeket alapértelmezésben a rendszer hozza létre, és lehetőleg ne nyúljunk hozzájuk.

1. táblázat **Merevlemez felosztása (512 bájtos blokkokkal) a /dev/hda merevlemezen**

#	type name	length	base	(size)
1:	Apple_partition_map Apple	63	1	
2:	Apple_Driver43*Macintosh	54	64	
3:	Apple_Driver43*Macintosh	74	118	
4:	Apple_Driver_ATA*Macintosh	54	192	
5:	Apple_Driver_ATA*Macintosh	74	246	
6:	Apple_FWDriver Macintosh	200	320	
7:	Apple_Driver_IOKit Macintosh	512	520	
8:	Apple_Patches Patch Partition	512	1032	
9:	Apple_HFS untitled	5217260	1544	(2,5 G)
10:	Apple_UNIX_SVR2 root	4194304	5218804	(2,0 G)
11:	Apple_UNIX_SVR2 usr	16777216	9413108	(8,0 G)
12:	Apple_UNIX_SVR2 opt	8388608	26190324	(4,0 G)
13:	Apple_UNIX_SVR2 swap	262144	34578932	(128,0 MB)
14:	Apple_UNIX_SVR2 home	8388608	34841076	(4,0 G)
15:	Apple_Free Extra	15403650	43229684	(7,3 G)

Eljött az ideje, hogy kialakítod a szokványos lemezterületeket. A magam részéről úgy döntöttem, hogy külön lemezrészeket hozok létre a `/`, `/usr/`, `/opt/`, és `/home` könyvtáraknak és a lapozóterületnek. Lehet, hogy az eltérő összeállítás mellett döntesz, az enyémről listát találhatsz az *1. táblázatban*. Miután a változásokat a `w` utasítással kiírtuk a lemezre, lépjünk ki a `q` paranccsal, majd indítsuk újra a gépet, ellenkező esetben a program nem érzékeli a lemezen történt változásokat. Kezdd újra a telepítést, indulás közben ismét nyomva tartva a C gombot. Add meg az újonnan létrehozott csatolási pontokat, majd ezután elkezdheted a csomagok válogatását, ahogyan ezt egy szokványos RedHat-rendszeren is tennéd. A telepítés befejeztével ismét újra kell indítanod a gépet. Ez alkalommal ne nyomj le semmilyen gombot, hiszen most a MacOS-t szeretnénk indítani.

Ismét a MacOS-ben találjuk magunkat. Nyisd meg a yaboot.conf fájlt, amit CD-ről másoltál a rendszermappába, és vess rá egy pillantást! A ➲ http://www.linuxvilag.hu/G4.html címen az én fájlomat a 2. listában láthatod.

Figyeld meg a „Linux” címkéjét! A CD-n lévő yaboot.conf hibás, ezért a yaboot elé egy további `\\`-t kell írnom. Használd most a ALMA-POT-O-F billentyűkombinációt, és lépj be az Open Firmware-be. A „0 >” jelnél írd be a következőt:

Az Open Firmware

Szólnom kell néhány szót az Open Firmware-ről. Az Open Firmware-t az IEEE 1275-os szabvány írja le. Az Apple-gépet illetően ez az első érdekes újdonságok egyike. Nem láttam addog, amíg nem volt rá szükségem. A Mac bekapcsolás után 880 Hz-es sípolással jelzi, hogy sikeresen túljutott az indulás POST részén (alkatrész-ellenőrzés), és felkészül az operációs rendszer betöltésére. Ennél a pontnál az indulás folyamata az Alma-Opt-o-f billentyűk lenyomásával leállítható. Ha minden jól megy, a következő üdvözlőszöveget pillantjuk meg:

```
Apple PowerMac 3,3 3.4f1 BootROM built on
08/08/00 at 22:02:19
Copyright 1994-2000 Apple Computer, Inc.

Welcome to Open Firmware.
To continue booting, type "mac-boot"
and press return
To shut down, type "shut-down"
and press return

 ok
0 > _
```

A „0 >” egy készenléti jel legbelül az Open Firmware Forth-értelmező. A Forth veremalapú nyelv – hogy megértsük a logikáját, vegyük a következő példát:

```
0 > 3  [RETURN]
1 > 4  [RETURN]
2 > +  [RETURN]
1 > .  [RETURN]
```

Megjelenik az eredmény: `0 > 7`
Az első parancs „3”-at helyezett el a veremben. A készenléti jelben megjelenő számból tudhatjuk meg, hány elem található a veremben. Ezek után leraktunk „4”-et a verembe, majd utasítottuk az értelmezőt, hogy adja össze a számot. Most már csak egyetlen elem van a veremben. A „.” művelet a pillanatnyilag veremben lévő elemet adja vissza és megjeleníti értékét, jelen esetben „7”-t.
Az Open Firmware használatával igen sokat láthatsz a gépedről: megtekintheted például alapértelmezett beállításait, ha az alábbit írod be:

```
0 > printenv
```

Az 1. lista mutatja (➲ http://www.linuxvilag.hu/G4.html) a beépített környezeti változókat és alapértelmezett értékeiket. További jó adatforrás a `devalias` parancs. Írd be, majd nyomd le a Return billentyűt. Figyeld meg a hd értékét! Ez az első IDE merevlemezed fizikai címe, a hd pedig álnév a printenv által kijelzett teljes címre.

```
0 > boot hd:,\\yaboot
```

Némi villogás után megjelenik a LILO jele és elindul a Linux. Most már láthatod, mire képes az Open Firmware! A fenti parancs segítségével úgy futtathatsz fájlt a merevlemezről, hogy még nincs is operációs rendszer betöltve a gépen! Az X-et (Xfree86 3.3.6) a telepítés során állítottam be az `Xconfigurator`-ral: 1024x768-as felbontást választottam 24-bites színmélységgel. A yaboot.conf-hoz a következő sort adtam hozzá:

2. táblázat **AltiVec-ismerő gcc: új adattípusok**

Kulcsszó	*Méret*	*Adattípus*
vector unsigned char	16	unsigned char
vector signed char	16	signed char
vector bool char	16	unsigned char
vector unsigned short	8	unsigned short
vector unsigned short int	8	unsigned short
vector signed short	8	signed short
vector signed short int	8	signed short
vector bool short	8	unsigned short
vector bool short int	8	unsigned short
vector unsigned int	4	unsigned int
vector unsigned long	4	unsigned int
vector unsigned long int	4	unsigned int
vector signed int	4	signed int
vector signed long	4	signed int
vector signed long int	4	signed int
vector bool int	4	unsigned int
vector bool long	4	unsigned int
vector bool long int	4	unsigned int
vector float	4	float
vector pixel	8	unsigned short

```
append="video=aty128fb:vmode:17,cmode:24"
```

A rendszermag ezáltal megfelelően felismeri a gépen található ATI videokártyát. Ezt követően megnyitottam a `/etc/X11/XF86Config` fájlt, és a „Screen” részbe beírtam a `DefaultBitsPerPixel 24`-et, így a színmélységet nem szükséges minden X-indításnál kézzel megadnom.

Az AltiVec

Most, hogy a Linux települt, nézzük meg közelebbről az AltiVecet. Az AltiVec lebegőpontos és egészszámos műveleteket végző egység, amely 32 darab 128 bites regiszterben tárolt adattal dolgozik. A vektorokat kezelő egység SIMD-módszerrel (egy utasítás, több adat) dolgozza fel a vektorregiszterekben található adatokat. A processzor egyetlen utasítással egyszerre 4, 8 vagy 16 adategységen tud dolgozni. Lássunk erre egy példát!
A Motorola 162 új assembly-utasítást hozott létre, hogy a programozók kihasználhassák az AltiVec új lehetőségeit. Ezekről az utasításokról részletesen olvashatsz az „AltiVec Technology Programming Environments Manual”-ban (altivec_pem-ben).
A magasabb szintű C-utasítások, amelyek ezeket az új assembly-parancsokat használják, megtalálhatók az „AltiVec Technology Programming Interface Manual”-ban (altivec_pim-ben). Mindkettő letölthető pdf formátumban a Motorola weblapjáról, vagy pedig a ➲ http://www.altivec.org webhelyről.
Következő lépésem az AltiVec RPM-ek letöltése és telepítése volt. Ezekben a csomagokban a gcc (2.95.2) módosított változata rejlik, ami ezeket az új lehetőségeket használja. Telepítése a következőképpen zajlik:

```
rpm -U binutils-2.9.5.0.22-6.vec.ppc.rpm
rpm -i gcc-altivec-2.95.2-1i.ppc.rpm
rpm -i gcc-altivec-c++-2.95.2-1i.ppc.rpm
```

Telepítés után az új `gcc`-t így tudtam használni:

```
gcc-vec program.c -o program
```

A `gcc` a /opt/bin könyvtárba települ, így nem zavarja az alapértelmezett `gcc` programot. Az RPM-csomag létrehoz egy `gcc-vec` hivatkozást a /usr/bin könyvtárban, ami a /opt-ban található altiveces `gcc`-re mutat.
Az új vektorparancsok használatához olyan alkalmazásokat kell írnod, amelyek használják őket, és olyan `gcc`-vel kell fordítanod, ami ismeri az új utasításokat. Hiába fordítod le a szabványos C forráskódot az új `gcc`-vel, nem számíthatsz az AltiVec-egység okozta teljesítménynövekedésre. Az AliVecet ismerő `gcc` tud az új kulcsszavakról és függvényekről. Az első hely, amit tanulmányoznod kell, az altivec_pim, ebből tanulhatod meg a `gcc-vec` által ismert új parancsokat. A vektoradattípusokat a *2. táblázatban* tekintheted meg. Az `altivec_pim` alapján az AltiVec-ismerő fordítóknak a következő makróval kell szolgálniuk:

```
#define __VEC__
```

Ha olyan kódot szándékozol írni, amit többféle rendszeren is le lehet fordítani, és ami mégis kihasználja az AltiVec lehetőségeit, amennyiben ez utóbbi megtalálható a rendszerben, az alábbihoz hasonlót írj:

```
#ifdef __VEC__
        /* Ide jön a kódod */
        /* ... */
#else
        /* ha nincs más lehetőség, a régi
           módszert választjuk */
        /* ... */
#endif
```

Mellékeltem egy egyszerű példaprogramot az AltiVecet ismerő `gcc` használatához, ez a ➲ http://www.linuxvilag.hu/G4.html címen érhető el (3. lista).
Először figyeld meg a `typedef union` meghatározásokat! Mint már korábban említettem, az AltiVec regiszterei 128 bitesek. Ezek a meghatározások egyrészt biztosítják, hogy a fordító az ilyen típusú adatokat 128-bites határokra igazítja, másrészt azt is, hogy a `vektor` adattípus egyes elemeit egyszerűen elérjük. Végül az sem elhanyagolható, hogy a `union` adattípus használatával és a `printf()` függvény segítségével könnyen betekinthetünk a regiszterek tartalmába. Az `altivec_pim` szolgáltat formázott bemenetet és kimenetet a `scanf()/printf()` függvénypár segítségével. Elméletileg a következő C-kóddal egy lebegőpontos számokból álló vektorregisztert nyomtathatnál ki:

```
vector float f32 = (vector float)(1.1, 2.2, 3.3,
4.4);
printf( "%,vf\n", f32 );
```

Ehhez azonban a C programkönyvtárnak (glibc*) ismernie kell a vektorformátum-meghatározásokat. A GNU C programkönyvtár jelenlegi megvalósítása (2.2) nem ismeri ezeket, valószínűleg nem is fogja. Ezért remélem, marad időm és módosítani tudom a GNU libc-t, hogy megfelelő legyen erre a célra. (Ha tanáccsal tudsz szolgálni vagy érdekel a dolog, nyugodtan keress meg!) Figyeld meg továbbá a vektortípusok meghatározásának két különböző módját. Az első módszer állandó vektort határoz meg, az értékeket `cVals`-, `sVals`-, `iVals`- és `fVals`-okban tárolja, ahol a vektoradattípust ugyanabban az állításban adjuk és határozzuk meg. Ez a példa azt mutatja, hogyan tároljunk állandókat (amik futásidőben nem változnak) vektorokban. A másik módszer lényege, hogy megad egy `union`-típust, majd a vektort futásidőben elemenként tölti fel. Ezen eljárás segítségével adatokat olvashatsz be egy átmeneti tárolóból, azokat vektortípusú változóba másolhatod be, majd ezt a változót átadhatod a vektort ismerő függvényeidnek.
Végül figyeld meg a `vec_add( )` függvény alakját! Minden esetben ugyanazt a `vec_add( )` függvényt használtam, és mindig jó eredménnyel tért vissza, függetlenül attól, hogy változói `vector short`, `vector int` vagy `vector float` típusúak voltak-e (a két összeadandó azonos kell legyen). Ebben az esetben a fordító értelmezte az adattípusokat, melyeket a `vec_add( )` függvénynek adtam át, és létrehozta számomra a megfelelő `vadd` assembly-utasítást. Az alábbi példában láthatjuk, hogyan képes a fordító a megfelelő egyeztetést létrehozni:

```
vector float a,b,c;

/* Assign a,b */

/*     ...      */

c = vec_add( a, b);
```

Ezt a következő assembly-utasítás hozza létre:

```
vaddfp c,a,b
```

A dolog egyre egyszerűbbé válik. A program fordításához használd a következő parancsot:

```
gcc-vec -fvec vecdemo1.c -o vecdemo1
```

A fordító `-fvec` kapcsolója a vektorparancsok értelmezésére utasítja. Ha nem adod meg a `-fvec` kapcsolót, a fordító nem ismeri fel a vektoradatokat és programfordulás közben hibát jelez, ez emlékeztet majd arra, hogy legközelebb ne felejtsd el használni a kapcsolót.
A ➲ http://www.linuxvilag.hu/G4.html címen a 4. listán látható a program kimenete.
A fentiekben rövid bevezetőt kíséreltem meg nyújtani a Linux PowerMacen történő használatáról, továbbá a linuxos programozók számára elérhető AltiVec-lehetőségekről.
Köszönetet szeretnék mondani az AltiVec-fórum tagjainak. A levelezési lista segítsége felbecsülhetetlenül értékes volt, e nélkül sokkal nehezebb lett volna elindulnom. Ugyancsak köszönettel tartozom az AltiVec-fejlesztőknek, amiért számtalan fejlesztői eszközt biztosítottak, és lehetővé tették, hogy a programozók könnyen dolgozhassanak egy olyan hatékony rendszeren, mint a G4-es.

Matthew Fite
(mattfite@yahoo.com) feleségével Észak-Virginiában él, és beágyazott programokat fejleszt. Annak ellenére, hogy napközben kereskedelmi RTOS-szel kell foglalkoznia, titkon arról álmodik, hogy egyszer bevezetheti az RTLinuxot. Mindig új tervei születnek, bár a felesége szerint már így is eleget foglalkozik a gépekkel.

Kapcsolódó címek

AltiVec ➲ http://www.altivec.org/
Motorola ➲ http://www.mot.com/AltiVec/
Apple Developer Connection ➲ http://developer.apple.com