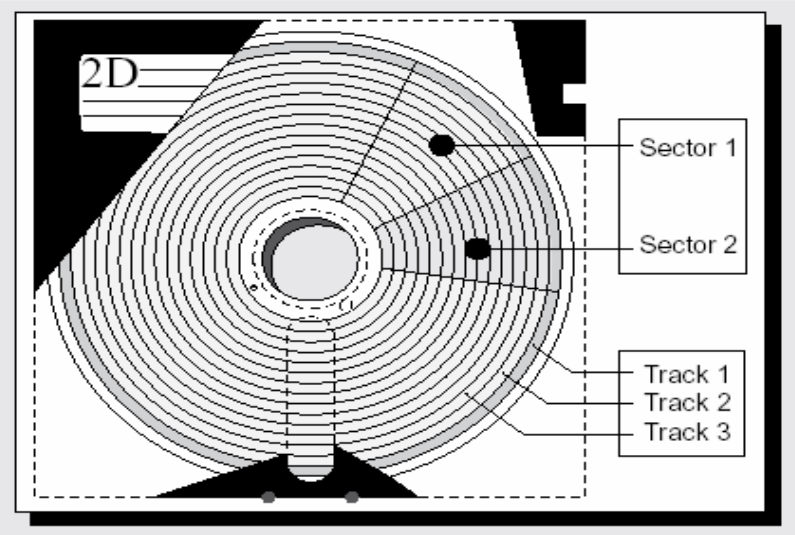


سواقة القرص المرن و القرص الصلب

Diskettes and Hard Drives

محمد سامر سروجي
محمد أبو حويلي
محمد جبلاق

الاربعاء، 23 آذار، 2005
www.aleposoft.com

الفهرس

مقدمة :

إن الهدف الأساسي من إجرائيات الـــBIOS هو إنجاز وتنفيـــذ التوابـــع ذات المـــستوى المنخفض نيابة عن الـــDOS .فعلى سبيل المثال, تستطيع إجرائيات الـــBIOS تهيئـــة ســـطح القرص المرن أو الولوج إلى قطاعاتsectors السواقة الصلبة .و لكن يبقى الـــDOS هو الموجه الرئيسي لهذه العمليات .

تقوم غالبية التطبيقات بإنجاز عمليات القرص الصلب في مستوى الـــDOS بدلا مـــن مـــستوى الـــBIOS . و على الرغم من ذلك يوجد بعض الاستثناءات . على سبيل المثال هنـــاك بعـــض البرامج الخدمية للقرص الصلب مثل PC Tools أو Norton , تقوم بـــالولوج إلـــى القـــرص الصلب بواسطة إجرائيات الـــBIOS . ولكن بشكل عام هذه البرامج التخصصية نادرة .

سنتناول في هذا البحث كيفية الولوج إلى سواقات الأقراص بواسطة توابع الـــBIOS . و نظـــرا لأنه لا يتم برمجة جميع متحكمات الأقراص بشكل مماثل , فإننا لا نستطيع برمجة موجه القرص بشكل مباشر .إن جميع التوابع التي يمكنك إنجازها على مستوى موجه القرص يمكنك أيـــضا أن تنجزها على مستوى الـــBIOS, لذلك فإنه من الجدير بالاهتمام استخدام توابع الـــBIOS لتجنب البرمجة التي تعتمد بشكل مباشر على موجه القرص .

سوف نناقش أيضا بالإضافة إلى توابع الـــBIOS بعض الموضـــوعات الأخـــرى ذات الـــصلة بالسواقة الصلبة , و سنقوم بشرح الأنواع المختلفة لمتحكمات السواقة الصلبة و ســـنرى أيـــضا كيف تسجل السواقات الصلبة البيانات .و سوف ننهي البحث بمناقشة عن كيفية تجزئـــة القـــرص الصلب و التي تتيح للمستخدم أن يقسم السواقة الصلبة إلى عدة سواقات منطقية .

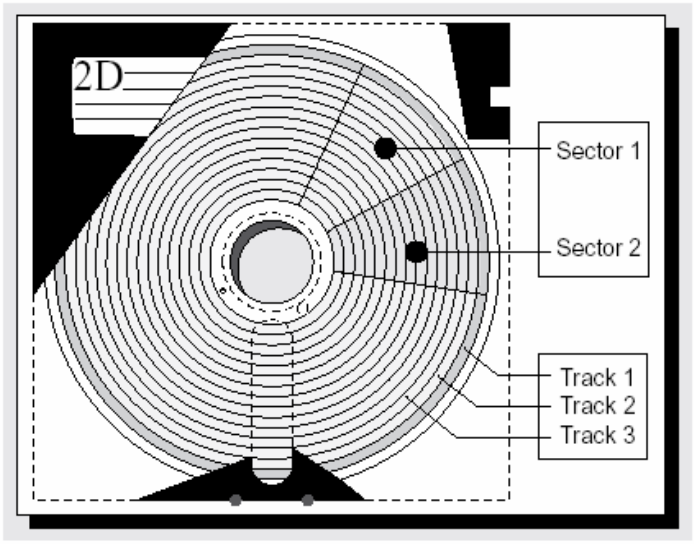
1) بنية القرص المرن و السواقة الصلبة

بداية لنلقي نظرة على الخصائص المشتركة للأقراص المرنة و السواقات الصلبة. تمتلك الأقراص المرنة و السواقات الصلبة مواصفات مماثلة و التي يعبر عنها بأرقام توابع الـBIOS و التي تطبق عليهما معا .لو تخيلنا و تأملنا أن القرص المرن هو إصدار ثنائي البعد من السواقة الصلبة , فإن التشابهات بينهما ستكون واضحة جلية .

1.1) بنية القرص المرن :

يتكون القرص المرن من مسارات tracks مستقلة مرتبة كدوائر متحدة المركز و بفواصل متساوية على سطح القرص المغناطيسي . هذه المسارات tracks مصنفة و مرقمة من 0 إلى N ; حيث أن N تمثل العدد الكلي للمسارات tracks مطروحا منه 1 و تتنوع اعتمادا على الشكل و البنية . يرمز المسار الأقصى (الأبعد) عادة بـ 0 , و المسار الذي يليه بـ 1 وهكذا . و تستمر العملية للوصول إلى الجزء الأعمق (الأقرب) .

Structure of a 5.25-inch diskette



يقسم كل مسار track ثانية إلى عدد ثابت من القطاعات sectors , و كل قطاع sector يملك نفس الحجم من المعطيات .

ترقم القطاعات من 1 إلى N ; حيث أن N تمثل العدد الكلي للقطاعات في المسار .

يعتمد العدد الأعظمي للقطاعات في المسار على نوع سواقة القرص المرن و بنية القرص المرن . يحتوي كل قطاع على 512 bytes و هذه هي أصغر كمية من المعطيات و التي يستطيع أي برنامج الوصول لها و التعامل معها . بمعنى آخر , يجب عليك أن تقرأ أو تكتب قطاعا كاملا في المرة الواحدة . حيث أنه لا توجد إمكانية لقراءة أو كتابة بايت وحيد من القرص المرن .

و بشكل مماثل للأقراص المرنة ذات الكثافة المزدوجة.يتم تسجيل المعطيات في كل قطاع باستخدام تقنية FM أو تقنية MFM , و هذه التقنيات هي نفسها المستخدمة في تسجيل المعطيات في السواقات الصلبة . و ينبغي عليك كمبرمج أن لا تقلق أبدا حيال تفاصيل التسجيل هذه التي تتم في سواقة الأقراص .

استعمل المعادلة التالية لحساب سعة القرص المرن , و تذكر أن هذه المعادلة لجهة واحدة فقط من القرص المرن . إذا كانت سواقة القرص المرن رأسي قراءة/ كتابة (مثل غالبية سواقات الأقراص المرنة حاليا) , فإنه ينبغي عليك مضاعفة هذه القيمة . يشير الـDOS إلى جهات القرص المرن و يميزها بـ side 0 و side 1 .

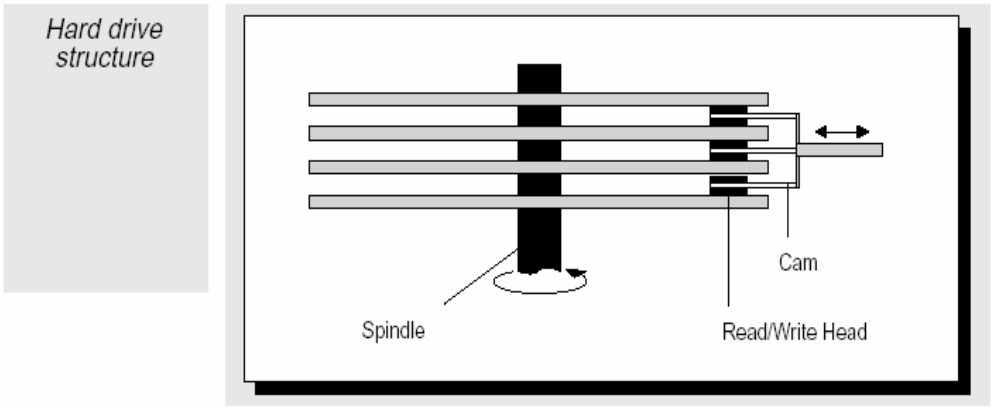
المعادلة هي : Sectors * tracks_per_sector * 512 [bytes per sector]

إن عدد القطاعات في المسار يؤثر أيضا في معدل نقل البيانات . يعبر معدل نقل البيانات عن سرعة الكترونيات سواقة القرص المرن وموجهها . مع سرعة دوران ثابتة تقدر بـ300 دورة في الدقيقة , فإن رأس القراءة / الكتابة يقوم بالمرور على بتات أكثر في وحدة الزمن و بالتالي يمكن كتابة قطاعات أكثر في المسار .

2.1) بنية السواقة الصلبة :

بما أن السواقة الصلبة تدور بشكل أسرع من القرص المرن بعشر مرات , فإن معدل نقل بياناتها سيكون على الأقل أكبر بعشر مرات من معدل نقل البيانات في القرص المرن . يزداد معدل نقل البيانات بمعدل ثانية مرفوعة للقوة 10 , لأن السواقات الصلبة ذات 3.5-inch تستطيع تخزين 100 قطاع لكل مسار .

هذه الخصائص لا تغير البنية الأساسية للسواقة الصلبة . بإمكاننا تخيل السواقة الصلبة بأنها مجموعة من الأقراص المغناطيسية المكدسة فوق بعضها البعض . كل قرص مغناطيسي يشبه القرص المرن , و له جهتين , كما أنه مقسم إلى مسارات , و كل مسار يقسم إلى قطاعات . كما أنه يوجد فوق سطح كل جهة من القرص رأس قراءة / كتابة و الذي يقوم بالولوج إلى البيانات . تصطف الأقراص بحيث أن المسار 0 لقرص ما يكون فوق المسار 0 للقرص الآخر تماما و تقوم ذراع القراءة/الكتابة بربط جميع رؤوس القراءة/ الكتابة مع بعضها . للوصول إلى مسار محدد لواحد من الأقراص , تحرك الذراع كل رؤوس القراءة/ الكتابة للمسار المحدد . و نظرا لأن هذا التنظيم يتطلب فقط آلية توضع وحيدة (ذراع القراءة/ الكتابة) , فإنه يبسط التصميم و يقلل من كلفة السواقة الصلبة . ولكن من ناحية أخرى , في هذا التنظيم , يجب أن تتحرك جميع رؤوس القراءة / الكتابة لتصل إلى البيانات في مسار مختلف . لذلك , لقراءة البيانات من المسار 1 لقرص ما , و بعدها البيانات من المسار 50 لقرص آخر و أخيرا البيانات من المسار 1 للقرص الأول مرة أخرى , فإن ذراع القراءة/الكتابة يجب أن تتحرك مرتين . إن توضع الذراع بهذا الشكل كمية هامة من الزمن فيما يخص زمن نقل البيانات .



لتقليل الزمن المطلوب للوصول إلى البيانات , يجب عليك أن تمنع البيانات من أن تنتشر عبر

مسارات عديدة . هناك طريقة لتحسين زمن الوصول لمجموعة من البيانات ألا و هي كتابة البيانات بشكل متسلسل على المسار . و إذا لم يكف مسار واحد لكتابة هذه البيانات , تتم الكتابة على نفس المسار من قرص آخر . بهذه الطريقة , فإنه لا يوجد حاجة لأن تتحرك ذراع

القراءة/الكتابة . حيث أن اختيار (تغيير) الرؤوس أسرع بكثير من التحريك الفيزيائي لذراع القراءة/الكتابة الميكانيكية لتغيير المسارات .

يستخدم مصطلح أسطوانة cylinder للتعبير عن أقراص متعددة متكدسة فوق بعضها البعض. و تشير هنا الأسطوانة cylinder لكل المسارات التي لها نفس الأرقام و لكنها متوضعة على أقراص مختلفة.

2) أشكال السواقات الصلبة و الأقراص المرنة

قبل وصف توابع الـBIOS للأقراص المرنة , سنقوم بإلقاء نظرة عامة على أشكال الأقراص المرنة .

تمتلك الحواسيب القوية في هذه الأيام على الأقل واحدة من أشكال الأقراص المرنة التالية :

3.5-inch, 1.44 Meg high-density (HD)

3.5-inch Super high density

5.25-inch, 1.2 Meg high-density (HD)

حاليا , يعتبر الشكل الأول من أكثر الأشكال الثلاثة شيوعا .

1.2) أشكال الأقراص المرنة

الأقراص المرنة الغير متوافقة مع الـDOS لا يمكن قراءتها بواسطة الـDOS ما لم تتوفر سواقة جهاز جديدة . تظهر الجداول التالية الأشكال المتوفرة للأقراص المرنة ذات 3.5- inch و 5.25 inch .

3.5-inch diskettes formats					
Type	Density	Capacity	Tracks	Sectors	DOS Version
DS DD	135 tpi	720K	80	9	3.2
DS DD	135 tpi	1.44 Meg	80	18	3.3
DS DD	270 tpi	2.88 Meg	80	36	3.3

5.25-inch diskettes formats					
Type	Density	Capacity	Tracks	Sectors	DOS Version
SS SD	40 tpi	160K	40	8	1.0
SS SD	40 tpi	100K	40	9	2.0
DS SD	40 tpi	320K	40	8	1.1
DS SD	40 tpi	360K	40	9	2.0
DS HD	96 tpi	1.2 Meg	80	15	3.0

1.1.2) سواقات الأقراص المرنة ذات 5.25-inch :

كانت سواقة القرص المرن ذات 5.25-inch السواقة القياسية للأقراص المرنة . بل إنها لاتزال قياسية للعديد من الحواسيب الشخصية . إن القرص المرن العادي ذو 5.25-inch ذو الكثافة المفردة له 4 قطاعات في المسار و40 مسار لكل جهة . هذا يزود سعة مقدارها 80K لكل جهة أو 160K لسواقة القرص المرن مع رئسي قراءة/كتابة . إن معدل نقله للبيانات 125K/sec يعد بطيء جدا مقارنة بالمقياس الحالي .

أتبع القرص المرن ذو الكثافة المفردة بالقرص المرن ذو الكثافة المزدوجة . ضاعفت هذه الأقراص المرنة عدد القطاعات في المسار إلى 8 بينما أبقت العدد الكلي للمسارات لكل جهة (40) . و بالتالي فإن سعة القرص المرن قد تم زيادتها إلى 160K للقرص المرن وحيد الجهة , و320K للقرص المرن الذي له جهتين . و هذا أدى أيضا إلى مضاعفة معدل نقل البيانات إلى 250K/sec .

هذا الشكل ذو السعة 320K لم يبق طويلا . حيث أن القطاعات الثمانية لا تملأ تماما المسار في الأقراص المرنة ذات السعة المزدوجة . و بالتالي , لا يزال هناك حيز لقطاع تاسع . و بالتالي فإن إضافة قطاع تاسع لكل مسار أدت إلى ازدياد السعة إلى 360K .

مع ظهور كومبيوتر IBM ظهر شكل جديد للقرص المرن ويدعى هذا الشكل القرص المرن ذو الكثافة العالية , حيث أن سعته 1.2 Meg . وفي هذا النوع تم زيادة عدد القطاعات في المسار إلى 15 و تم مضاعفة عدد المسارات إلى 80 مسار لكل وجه . يشبه هذا النوع تماما الأقراص المرنة ذات الكثافة المزدوجة بأنه يتم استخدام كلا الجهتين للقرص المرن . نظريا , نجد أنه من الممكن وجود 16 قطاع لكل مسار . و لكن في الحياة العملية قرر المطورون أن يضعوا 15 قطاع لكل مسار و ذلك لضمان الوثوقية لسواقة القرص المرن . إن معدل الدوران في هذا النوع يساو 360 دورة في الدقيقة . و كما ذكرنا أيضا أنه في هذه الأقراص المرنة ذات الكثافة العالية فقد تم ضغط المسارات إلى 80 مسارا بدلا من 40 و التي كانت في الأقراص ذات السعة المزدوجة , و بالتالي فإن سواقات الأقراص المرنة ذات السعة المزدوجة القياسية لا تستطيع قراءة هذه الأقراص أو الكتابة عليها . ولكن يوجد نوع وحيد من السواقات و التي تدعى السواقات متعددة الوظائف أو سواقات MF التي هي اختصار لـ(multifunction) قادرة على القراءة من او الكتابة إلى هذا النوع من الأقراص كما أنها قادرة على التعامل مع الأقراص ذات السعة المزدوجة . حيث أن ذلك يتم بضبط معدل نقل البيانات و تقليل سرعة الدوران على السرعة العادية و التي تساوي 300 دورة في الدقيقة .

إن زيادة سعة التسجيل من كثافة مزدوجة إلى كثافة عالية ليست بالأمر البسيط الذي يتعلق بأمور الكترونية في السواقة . بل إنها بشكل أساسي مسألة البرغلة للمادة المغناطيسية على القرص المرن . حيث أنه كلما صغرت الرقاقات المغناطيسية المفردة كلما ازدادت كمية المعلومات التي يمكن تسجيلها على السطح . الآن بإمكاننا أن نطرح السؤال التالي و نجيب عليه.

السؤال : لماذا لا يمكن أبدا تهيئة الأقراص ذات السعة المزدوجة تهيئة خالية من الأخطاء في سواقات الأقراص ذات السعة 1.2 Meg ؟

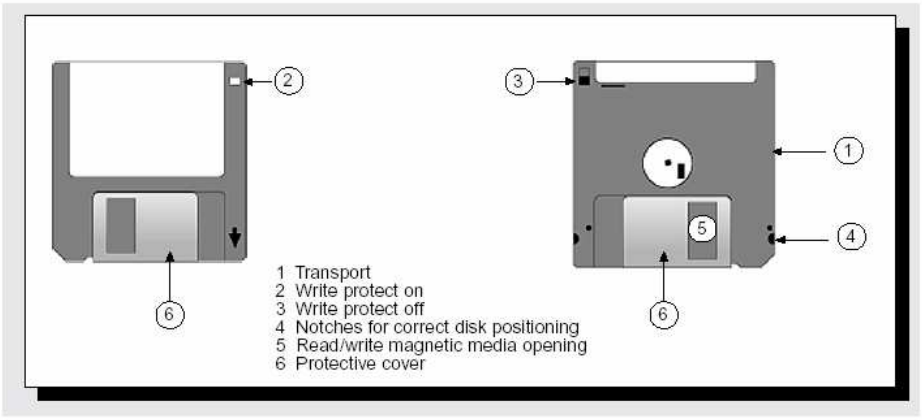
الجواب بكل بساطة : لأن البرغلة في هذه الأقراص خشنة جدا .

طالما أنه يمكن تهيئة الأقراص المرنة ذات السعة المزدوجة في السواقات ذات السعة العالية و بالتالي فإن هذه الأقراص يمكن أن تقرأ فقط من قبل الكومبيوتر الشخصي الذي تم تهيئتها منه . ستعطي الكومبيوترات الأخرى بكل بساطة مشاكل في القراءة مما يجعل من المستحيل استخدام هذه الأقراص المرنة . و يعود السبب للتفاوتات في توضع رؤوس القراءة و الكتابة من سواقة لأخرى . قد يحدث ذلك أيضا عند محاولة تهيئة القرص المرن ذو الكثافة العالية في سواقة ذات كثافة مزدوجة .

2.1.2) سواقات الأقراص المرنة ذات 3.5-inch

تعبر هذه السواقات الصغيرة الحجم مقارنة بسابقتها أكثر انتشارا في الحواسيب الشخصية و المحمولة . و تعتبر من السواقات المفضلة نظرا لحجمها الملائم و لمتانة الغلاف البلاستيكي الذي يحيطها . تقوم هذه السواقات بتسجيل البيانات إما بطريقة الكثافة المزدوجة أو الكثافة العالية . تكون السعة في الكثافة المزدوجة 720 K , حيث أن عدد القطاعات 9 لكل مسار وعدد المسارات 80 في الجهة الواحدة . و لكن الشكل الأكثر انتشارا من هذا النوع من السواقات هي السواقة 3.5-inch ذات السعة 1.44 Meg , حيث أنها تملك أيضا 80 مسارا في كل جهة و لكن تم مضاعفة عدد القطاعات إلى 18 قطاع لكل مسار . هذا يؤدي إلى مضاعفة معدل نقل البيانات .

Structure of a 3.5-inch diskette



تعمل سواقات الأقراص 3.5-inch ذات الكثافة العالية تماما كالسواقات 5.25-inch من النوع MF حيث أنها يمكن أن تضبط لتستطيع قراءة , كتابة وتهيئة الأقراص مزدوجة الكثافة .

يوجد حاليا سواقات 3.5-inch بسعة 2.88 Meg تدعى الأقراص ذات الكثافة العالية الإضافية (ED) . الفرق فيها أنه تم مضاعفة عدد القطاعات إلى 36 عن سابقتها و يمكن قراءتها فقط عن طريق سواقات أقراص (ED) الجديدة و هذه السواقات قادرة على معالجة الكثافة المزدوجة و الكثافة العالية .

إن سواقات الأقراص المرنة التي تستطيع قراءة وكتابة أشكال مختلفة من الأقراص يجب أن تكون قادرة على تحديد نوع القرص و تقوم باستخلاصها من القرص , بعد ذلك تستطيع تمرير هذه المعلومات للـBIOS قبل الولوج للبيانات على القرص المرن .

إن إيجاد القرص المرن 5.25-inch ليس بالأمر السهل أبدا , لأنه يتم تحديد المعلومات اللازمة فقط عن طريق قراءة البيانات . و بالتالي , يجب أن يكون القرص مهيأ بشكل مسبق. من ناحية أخرى , يمكن تحديد القرص المرن 3.5-inch عن طريق ثقب صغير, يتوضع هذا الثقب على الجهة الأخرى من جهة التي يوجد بها الحماية للقرص من الكتابة . يوجد ضمن سواقة القرص المرن نفسها حساس يستطيع اكتشاف حضور أو غياب الثقب , و على عكس القرص المرن ذو الكثافة العالية , فإن القرص المرن ذو الكثافة المزدوجة لا يمتلك هذا الثقب و كما أن القرص المرن ذو الكثافة المرتفعة الإضافية يمتلك ثقبا و لكنه موجود في مكان آخر.

2.2) سواقات القرص و متحكماتها :

كما رأينا أن السواقة المرنة تتكون من محرك يتولى عملية تحرك القرص بمعدل 300 دورة في الدقيقة , و تقنية لتحريك رأس القراءة و الكتابة بالإضافة إلى مكون الكتروني يدعى فاصل البيانات data separator , و الذي يقوم بتحويل الجهد إلى حزمة من البيانات الثنائية.يتم التحكم و توجيه القرص الصلب عن طريق موجه القرص المرن diskette controller , والذي إما أن يكون جزء من اللوحة الأم للكومبيوتر أو بطاقة دخل/خرج في واحد من شقوق التوسيع expansion slot للكومبيوتر .

3) الوصول إلى سواقات الأقراص المرنة عن طريق BIOS

يوجد مجموعة كاملة من توابع BIOS و التي تمكننا من التعامل و الوصول إلى سواقات الأقراص المرنة . و تستخدم المقاطعة 13H لاستدعاء هذه التوابع . كما أن هذه المقاطعة تعتبر

واجهة interface لتخديم السواقات الصلبة . و تتشارك غالبا السواقات الصلبة و المرنة في نفس أرقام التوابع , و للتمييز فيما إذا كانت السواقة التي تقوم باستدعاء التابع هي السواقة الـصلبة أو المرنة فإنه ينبغي علينا تمرير نوع السواقة للتابع الموجود في المسجل DL .

بالنسبة للسواقة المرنة يتم استخدام القيمة 0 (للسواقة A) و القيمة 1 (للسواقة B) , و هناك بعض المتحكمات تدعم 4 سواقات مرنة بحيث يقبل الـBIOS القيمة 2 و 3 للسواقات المرنة الأخرى . أما بالنسبة للسواقات الصلبة فإنه يتم تحديدها بالقيم 80H و 81H .

و هنا يجب التمييز أيضا بين PC/XT-BIOS و AT-BIOS. , الجدول التالي يعـرض توابـع القرص المرن للمقاطعة 13H .لاحظ أن التابعان 17H و 18H يقومان بنفس الوظيفة , هذا ليس خطأ , حيث أن التابع 18H تم تقديمه للسواقات المرنة 3.5-inch , لأن التابع الأقدم 17H غيـر قادر على دعم هذه السواقة , لذلك تم استبداله بالتابع الجديد .

Diskette functions of the BIOS interrupt 13H							
No.	Tasks	PC/XT	AT	No.	Tasks	PC/XT	AT
00H	Reset	Yes	Yes	08H	Request Format	Yes	Yes
01H	Read status	Yes	Yes	15H	Define drive type	No	Yes
02H	Read	Yes	Yes	16H	Detect diskette change	No	Yes
03H	Write	Yes	Yes	17H	Determine diskette format	No	Yes
04H	Verify	Yes	Yes	18H	Determine diskette format	No	Yes
05H	Format	Yes	Yes				

1.3) حالات السواقة

يوجد وظيفة أخرى لتوابع BIOS حيث أنها تقوم بإرجاع الحالات أو شيفرة الخطـأ , و يتم إرجاع هذا الشيفرة للمستدعي في المسجل AH . إن القيمة الصفرية و تفعيـل علـم الحمـل يشيران لوجود خطأ .

Status and error codes of the BIOS diskette functions			
Code	Meaning	Code	Meaning
00H	No error	08H	DMA overflow
01H	Illegal function number	09H	Data transfer past the segment limit
02H	Address marking not found	10H	Read error
03H	Attempt to write to write-protected diskette	20H	Diskette controller error
04H	Addressed sector not found	40H	Track not found
06H	Diskette was changed	80H	Time out error, drives does not respond

بإمكانك أيضا تحديد حالة القرص المرن عن طريق التابع 01H . و يتم ذلك بأن تمرر رقم التابع 01H إلى المسجل AH و نوع السواقة (صلبة أو مرنة) إلى المسجل DL , و هذا يمثل التهيئة الابتدائية , و من ثم نقوم باستدعاء التابع , و بعد استدعاء التابع فإنه يتم إرجاع حالة السواقة إلى المسجل إليك عن طريق المسجل AH .

2.3) إعادة تهيئة Resetting القرص المرن

يجب عليك أن تعيد تهيئة reset سواقة القرص المرن بعد حدوث أي خطأ . للقيام بـــذلك استعمل التابع 00H . قم بتمرير رقم التابع 00H للمسجل AH و نوع السواقة للمسجل DL . و بعد استدعاء التابع فإنه سوف يتم إرجاع حالة السواقة الحالية في المسجل AH . لا يوجد هنـــاك أي مشكلة إذا مررت 0 أو 1 كنوع للسواقة , ذلك لأنه سيتم إعادة تهيئة جميع السواقات المرنـــة . لكن انتبه جيدا للقيمة التي تمررها للمسجل DL فإنها تؤخذ بعين الاعتبار , حيث إذا مررت قيمة أكثر من 80H " يعطيك العافية " , لأنه سيتم إعادة تهيئة السواقة الصلبة .

3.3) نمط و شكل سواقة الأقراص المرنة :

إن البرنامج بحاجة إلى أن يعرف نمط و شكل سواقة الأقراص المرنة للتعامـــل معهـــا . لتحديد هذه المعلومات يجب عليك استخدام التوابع 08H و 15H . حيث أن التابع 08H يستخدم للتمييز بين الأنواع المختلفة للسواقات المرنة . و كما تعلمنا قم بتمرير رقـــم التـــابع 08H إلـــى المسجل AH و نوع السواقة (صلبة أو مرنة) على المسجل DL .

أما بالنسبة للمعلومات المرجعة فهي تكون على الشكل التالي :

Information returned by the 08H function	
Register	**Information**
BL	Drive Type 01H = 5.25-inch, 360K 02H = 5.25-inch, 1.2 Meg 03H = 3.5-inch, 720K 04H = 3.5-inch, 1.44 Meg
DH	Maximum number of sides (always 1)
CH	Maximum number of tracks
CL	Maximum number of sectors
ES:DI	Pointers to DDPT

إذا كان يوجد أي خطأ فسيتم إرجاع شيفرته إلى المسجل AH و تفعيل علم الحمل .

سيتم إرجاع قيمة إلى المسجل BL هذه القيمة هامة جدا فهي تبين نوع السواقة فيما إذا كانت 5.25-inch أو 3.5-inch و الأهم من ذلك أنها تبين الكثافة (مزدوجة أو عالية) , و لكن انتبه فليس من الضروري أن تصف هذه القيمة كثافة القرص الموجود في السواقة , و لكنها تصف أكبر كثافة ممكنة ,معلومات عن عدد الجهات و المسارات و القطاعات في المسجلات DH , CL , CH.مؤشر موجود في زوج المسجلات ES:DI يؤشر إلى DDPT و الذي يمثل جدول لباراميترات سواقة الأقراص .

سنتحدث الآن عن التابع 15H , هذا التابع مدعوم فقط من قبل ATs و سواقاتها التي من النوع FM , فهذه السواقات قادرة على اكتشاف فيما إذا تم تغيير القرص المرن أم لا. هذه الخاصية مهمة جدا للبرامج التي تعتمد على معالجة القرص الحالي و التعامل معه عندما يكون القرص موجودا في السواقة , و ينطبق هذا الكلام خاصة على DOS , حيث أن هذه البرامج تقوم بقراءة جدول FAT قبل الولوج لبيانات القرص و تحدد أي القطاعات في هذا القرص محجوزة و أيها لا يزال غير مستخدم. لذلك إذا تم تغيير القرص من غير أن يعلم DOS بذلك فإن DOS سيتابع باستعمال المعلومات الخاصة بالقرص السابق و يتعامل مع القرص الحالي على هذا الأساس و يمكن أن يقوم و بشكل غير مقصود بالكتابة فوق معلومات سابقة overwrite و إفساد محتويات هذا القرص . لذلك كان لا بد من معرفة DOS إذا تم تغيير القرص أم لا. لنعود على التابع 15H , إن هذا التابع هو من توابع BIOS كما قلنا و يقوم بتحديد فيما إذا تم تبديل القرص أم لا و يتم تهيئة القيم الابتدائية في المسجلات كما في التوابع السابقة و من ثم إرجاع القيمة المطلوبة, و يعرض الجدول التالي المعلومات التي يتم إرجاعها بعد استدعاء هذا التابع .

Drive codes of function 15H	
Code	Meaning
AH = 00H	Drive not present
AH = 01H	Disk drive, does not recognize diskette changes
AH = 02H	Disk drive, recognizes diskette changes
AH = 03H	Hard drive

4.3) قراءة قطاعات القرص المرن

يقوم التابع 02H بقراءة قطاعات القرص . إن هذا التابع يستطيع قراءة أكثر من قطاع بشرط أن تكون هذه القطاعات على نفس المسار و بجانب بعضها البعض .

و تجدر الإشارة إلى أن البيانات لا يتم نقلها إلى عنوان ذاكري ثابت , لذلك فإنه يتم تمرير عنوان الـbuffer إلى زوج المسجلات ES:BX , حيث أن المسجل ES يحتوي عنوان القطعة segment للـbuffer , و المسجل BX يحتوي على عنوان إزاحة الـbuffer .

بعد استدعاء هذا التابع فإنه سيتم إرجاع حالة الخطأ إلى المسجل AH و عدد القطاعات المقروءة إلى المسجل AL . و إذا تم تفعيل علم الحمل فإن هذا يشير إلى وقوع خطأ .

الجدول التالي يبين المسجلات التي تتأثر باستدعاء التابع 02H .

Register when calling function 02H	
Register	**Information**
AL	Number of sectors to be read
DL	Drive specification value
DH	Side (0 or 1)
CL	Sector number (1 to N)
CH	Track number (0 to N-1)
ES:BX	Address of the buffer for the data to be read
AH = 03H	Hard drive

إذا كنت تستخدم سواقات MF , بإمكانك القيام بحيلة ذكية لتحدد نوع القرص المرن . إذا جربت قراءة قرص ما و استخدمت لذلك قيمة قطاع أكبر من 9 , فعندها تستطيع أن تحدد فيما إذا كان هذا القرص DD أو HD . لأنه و كما نعلم أن العدد الأعظمي للقطاعات في المسار في الأقراص من النوع DD (الكثافة المزدوجة) 9 , و بالتالي سيقوم التابع 15H بإرجاع قيمة تدل على خطأ , طبعا قيمة المسار غير ضرورية و لكن كما نعلم يجب أن تكون أقل من 40 .

الآن انتبه عند إرجاع التابع لقيمة تدل على وقوع خطأ و لا تقوم مباشرة بإيقاف العملية الحالية التي تقوم بها, و لكن يجب عليك أن تعيد تكرار هذه العملية (سواء أكانت قراءة أو كتابة أو تهيئة

) 3 مرات على الأقل قبل أن تستسلم و تجزم بأن هناك خطأ حقيقي قد حـــدث . عـــادة , تفـــشل العملية التي تريد القيام بها في المرة الأولى , و لكنها تنجح في المحاولة الثانية أو الثالثة , و ربما يكون السبب في ذلك أن رأس القراءة /الكتابة لا يكون قد وضع في الموضع المحدد فـــي المـــرة الأولى , أو أن السواقة المرنة لم تكون قد تزامنت حتى هذه اللحظة مع الالكترونيات. لا تقلـــق بشأن صحة البيانات عند وقوع خطأ ما لأن السواقة تتأكد عن طريق تساوي و تماثل المعلومـــات لضمان صحة البيانات في كل قطاع .

5.3) الكتابة على قطاعات القرص المرن

يستخدم التابع 03H للكتابة على قطاعات مستقلة . يبين الجدول التالي البراميترات التي تمرر لهذا التابع .

Register when calling function 03H	
Register	Information
AL	Number of sectors to be written
DL	Drive specification value
DH	Side (0 or 1)
CL	Sector number (1 to N)
CH	Track number (0 to N-1)
AL	Number of sectors to be read
ES:BX	Pointer to the buffer containing the data

6.3) التأكد من قطاعات القرص المرن

يقوم التابع 04H بفحص و اختبار فيما إذا تم نقل البيانات إلى القـــرص المـــرن بـــشكل صحيح . لا يتم مقارنة البيانات الموجودة بالذاكرة بالبيانات الموجودة على القـــرص . بـــل يـــتم استخدام قيمة CRC التي تحدد فيما إذا تم نقل البيانات بالشكل الصحيح . CRC هي اختصار لـــ Cyclical Redundancy Checkو هي عبارة عن إجرائية موثوقة جدا و تقوم بالتأكد من ذلك بدقة كبيرة . حيث أنها تقوم بعمليات و معادلات رياضية معقدة و تقوم بتجميع القيم لكـــل بايـــت ضمن القطاع الواحد مع الـــcheck sum و تطبق عليها هذه المعادلات الرياضية .

إن الباراميترات للتابع 04H هي نفسها الموجودة في التوابع 02H و 03H باستثناء عنوان الـbuffer فهو لا يلزم. حالياً , تعتبر أغلب السواقات موثوقة , لذلك فإن غالبية المبرمجين يعتبروا أن هذا الإجراء غير ضروري و لا حاجة لاستعماله .

7.3) تهيئة مسارات مستقلة ضمن القرص المرن

يستخدم التابع 07H لتهيئة القرص المرن كاملا . و لكن هناك إمكانية لتهيئة مسارات مستقلة ضمنه . للقيام بذلك , ينبغي عليك أولا أن تستخدم التابع 18H لإخبار BIOS بنوع السواقة . يمرر رقم التابع في المسجل AH , و نوع السواقة في المسجل DL , و عدد المسارات في المسجل CH و عدد القطاعات في المسجل CL . و بعد استدعاء التابع فإن علم الحمل يشير إلى النوع المحدد المدعوم بواسطة السواقة المرنة . في هذه الحالة يكون زوج المسجلات ES:DI مؤشر لـDDPT و الذي تحتاجه توابع التهيئة اللاحقة .

بعد أن يقوم التابع 07H بتحديد النوع المطلوب و بعد أن يتم تمرير مؤشر DDPT لشعاع المقاطعة 1EH , عندها بإمكانك القيام بعملية التهيئة الحقيقية . لتبدأ بالتهيئة , استعمل التابع 05H , هذا التابع يقوم بتهيئة مسار كامل . و على الرغم من أنه بإمكانك تهيئة قطاعات مستقلة بسعة 512,256,128 أو حتى 1024 بايت لكل قطاع , فإنك لا تستطيع أن تهيئ إلا 512 بايت فقط تحت DOS و ذلك لأن DOS يدعم فقط هذه السعة للقطاعات .

قم باستخدام التابع 05H ومرر القيم إلى المسجلات كما هو مبين بالجدول التالي :

Register when calling function 05H	
Register	**Information**
AL	Number of sectors in the track
DL	Number of the drive
CH	Number of the track
DH	Side (0 or 1)
ES:BX	Pointer to format table

يشير زوج المسجلات ES:BX على "جدول التهيئة" . إن هذا الجدول يمثل خصائص التهيئة , و كما نرى من الشكل أن هذا الجدول هو عبارة عن مصفوفة من مؤلفة من كيانات أو سجلات لكل قطاع سيتم تهيئته , و السجل الواحد مؤلف من 4 بايتات . و على الرغم من أنه يتم تمرير رقم المسار و جهة القرص المرن إلى التابع 05H , فإنه يجب تكرار ذلك في الجدول .

ES:BX register pair "format table"	
Offset	**Meaning**
0	Track to be formatted
1	Diskette side (always 0 for one-sided diskettes): 0 = Front side 1 = Back side
2	Number of the sector
3	Number of bytes in this sector: 0 = 128 bytes 1 = 256 bytes 2 = 512 bytes 3 = 1024 bytes

يتم إنشاء القطاعات فيزيائيا بالتسلسل نفسه لمكونات الجدول . و بالتالي من الممكن تهيئة الكيان الأول على أنه القطاع رقم 1 و الكيان الثاني على أنه القطاع رقم 7. يسجل الرقم المنطقي للقطاع في ترويسةheader كل قطاع على القرص المرن.

ملاحظة : ليس من الضروري أن يكون عدد البايتات في كل قطاع هو نفسه , لذلك فإنه يجب أن يتم تحديد أعداد البايتات بشكل صريح في كل قطاع من الجدول , من هنا نستنتج أنه بإمكاننا أن نغير هذه الأعداد في كل قطاع بمعنى أن نحدد لكل قطاع عدد بايتات يختلف عن القطاع الآخر و ذلك كنوع من حماية النسخ .

8.3) جدول بارامترات سواقة القرص Disk Drive Parameter Table (DDPT)

لبرمجة موجه القرص المرن , فإن الـBIOS بحاجة لمعرفة معلومات التهيئة الفيزيائية و التي وصفناها في الأعلى , بالإضافة لمعلومات أخرى. تحتوي ROM BIOS على

جدول يحتوي كل أنواع السواقات و الأقراص المرنة التي يدعمها . و أيضا بإمكانك أن تعـــرف DDPT خاص بك , وبما أن BIOS يرجع DDPT الحالي عبر مؤشر FAR , و الذي يكـــون موجودا ضمن المواقع الذاكرية التي يتواجد فيها شعاع المقاطعة 1EH . و نظـــرا لأن DOS و العتاد الصلب للكومبيوتر الشخصي كلاهما لا يستخدمان المقاطعة 1EH , فإنه بالإمكـــان تغييـــر محتوى هذه المواقع الذاكرية .

إن DOS يقوم بإنشاء DDPT الخاص به , و مهمته زيادة سرعة الوصول إلى القرص المرن . إن حجم هذا الجدول 11 بايت , و كما نرى من الشكل التالي الذي يوضح الجـــدول أنـــه لـــيس بالإمكان تغيير كل البارامترات , إنما فقط بإمكاننا تغيير البارامترات المعلمة بالإشارة * .

Diskette functions of the BIOS interrupt 13H					
Offset	Meaning	Type	Offset	Meaning	Type
*00H	Step rate and head unload time	1 BYTE	06H	DTL (Data Length)	1 BYTE
*01H	Head load time	1 BYTE	07H	Length of GAP3 when formatting	1 BYTE
*02H	Post run-time of diskette motor	1 BYTE	*08H	Fill character for formatting	1 BYTE
03H	Sector size	1 BYTE	*09H	Head settle time	1 BYTE
04H	Sectors per track	1 BYTE	*0AH	Time to run up of diskette motor	1 BYTE
05H	Length of GAP3 when reading/writing	1 BYTE			

إن الحقل الأول من جدول DDPT له حقلين فرعيين , الأول هو معدل الخطـــوة (bits 4-7) و الثاني هو زمن تفريغ الرأس (bits 0-3) . يصف معدل الخطوة الزمن الذي يستغرقه الموجـــه لتحريك رأس القراءة/الكتابة من مسار إلى آخر , و هذه القيمة تمثل بالملي ثانية , حيث أن القيمة 0FH تمثل 1 ms , 0EH تمثل 2 ms و0DH تمثل 3 ms . أما زمن تفريغ الرأس فهو يصف الزمن الذي يستغرقه رأس القراءة/الكتابة ليرتفع عن سطح القرص المرن , و يعبر عن هذه القيمة كمعاملات لـــ 16 ms. و القيمة الافتراضية هي 0FH (240 ms) .

أما الحقل الثاني فهو أيضا ينقسم إلى حقلين فرعيين , زمن تحميل الـــرأس (bits 1-7) و علـــم DMA (bit 0) . بالنسبة لزمن تحميل الرأس فهو الزمن الذي يستغرقه رأس القـــراءة/ الكتابـــة ليستقر على سطح المسار , و يعبر عن هذه القيمة كمعاملات لـــ 2 ms. أما بالنسبة لعلم DMA فهو يجب أن يأخذ دائما القيمة 0 .

أما بالنسبة للحقل الثالث فهو يعبر عن الفترة الزمنية التي تنقضي حتى يتم توقف محرك القرص المرن عند عدم وجود أي عملية لها علاقة بالقرص المرن و ينتظر تنفيذها . و نظرا لأن محرك القرص المرن يستغرق فترة من الزمن لا بأس بها حتى يقلع بشكل كامل , فإنه لا ينبغي علينا أن نقوم بإيقافه مباشرة بعد كل عملية وصول للقرص المرن . هذه القيمة تتعلق بالدور الذي يساوي تقريبا 18 ticks per second حيث أن

(1 tick is approximately 55 ms) . و القيمة الافتراضية هي 25H و التي تساوي تقريبا 2 ms .

الحقل الرابع يحدد عدد البايتات في كل قطاع و التي يمكن استخدامها في عملية القراءة و الكتابة , وهي تتوافق بشكل رئيسي و تطابق قيم تهيئة القطاع , و لذلك فهي عادة تحتوي القيمة 3 من أجل 512 بايت لكل قطاع . للقراءة من أو للكتابة في قطاعات لها سعات أو حجوم مختلفة , ينبغي عليك أولا أن تضع القيمة المناسبة في هذا الحقل ثم تقوم بالعملية المطلوبة .

الحقل التالي الموجود عند الإزاحة الذاكرية 04H هو العدد الأعظمي للبايتات في كل قطاع , و الذي يعتمد على نوع القرص المرن .

أما بالنسبة للحقول الثلاثة التالية فهي تتعلق بالترميز و فك الترميز لمعلومات القطاع , و التي تخزن على القرص المرن مع البيانات الحقيقية . انتبه جيدا و لا تحاول العبث أو التأثير على هذه القيم أبدا .

أما بالنسبة للحقل الموجود عند الإزاحة 08H يمكن تغييره , و يحتوي هذا الحقل على رمز ASCII الذي يعبر عن المحرف الذي نريد أن نملأ به القطاعات بعد عملية التهيئة , حيث أنه عند التهيئة (و التي يتم فيها طبعا خلق القطاعات) تعطى القطاعات محتوى ثابت . و القيمة الافتراضية هي إشارة القسمة (ASCII code 246) .

يحتوي الحقل التالي وقت استقرار الرأس . فبعد أن ينتقل رأس القراءة/الكتابة من مسار لآخر , يكون بحاجة لتأخير زمني قصير ليتم تخميد الاهتزازات التي تنتج عن هذه الحركة . و بعد ذلك يستطيع رأس القراءة/الكتابة إنجاز عمليات الولوج للبيانات اللاحقة بشكل مناسب و صحيح . القيمة الافتراضية لهذا الحقل 25 ms .

يعبر الحقل الأخير عن الزمن الذي يستغرقه محرك القرص المرن لكي يصل سرعة التشغيل , و هذه القيمة من معاملات و أجزاء 1/8 seconds .

4) استعمال الـ BIOS للوصول إلى الأقراص الصلبة:

سوف نصف في هذا البحث توابع Bios الخاصة بالوصول إلى سوا قات القرص الصلب hard drives .مهما يكن,قبل أن نبدأ يجب أن نحذرك بشأن تجريب هذه التوابع.هناك اختلاف بينها وبين سواقة القرص المرن floppy disk drive التي من خلالها يمكنك أن ستعمل قرص غير مرن من أجل التجريب فلا يمكنك اختبار القرص الصلب بهذه الطريقة.سيقودك الاستخدام الغير مبالي لتوابع التهيئة والكتابة إلى فقدان البيانات وسيتعذر عليك إصلاحه.لأن هيكلية نظام التشغيل DOS سوف تفرض نفسها على القرص الصلب,حيث أن هدم قطاع واحد one sector يمكن أن يؤدي إلى اختفاء جميع المسارات و الملفات وذلك لأن نظام التشغيل DOS لم يعد يعرف أين مكانهم على القرص الصلب hard disk .

لذلك إذا أردت أن تختبر توابع الـ BIOS تأكد من تنشأ نسخة احتياطية backup لقرصك الصلب مقدما.أو تستخدم حاسب آخر إذا كان ذلك متاحا لك.تلك هي الطريقة الوحيدة لتتجنب ضياع المعلومات,لأن حتى خدمة القرص الصلب المحكم قد لا تستطيع مساعدتك.

1.4) مقاطعة القرص الصلب من أجل الـ BIOS :

كما ذكرنا سابقاً تتشارك سواقات القرص الصلب hard drives مع سواقات القرص المرن floppy drive من حلال المقاطعة 13H .على الرغم من أن التوابع الخاصة بسواقة القرص الصلب hard drive والقرص المرن floppy drive هي متطابقة,فإن الـ BIOS تتحكم بسواقة القرص الصلب hard drive بشكل مختلف عن تحكمها بسواقة القرص المرن floppy drive .لهذا السبب تتضمن الـ BIOS نموذجا للتحكم بسواقة القرص الصلب hard drive منفصل عن نموذج التحكم بسواقة القرص المرن.

عندما يتم استدعاء المقاطعة 13H فإن رقم الجهاز في المسجل DL يحدد فيما إذا كان هي عنونة للقرص الصلب أو القرص المرن..تمثل القيمة 80H أن سواقة القرص الصلب الأولى هي

المطلوبة في حين تمثل القيمة 81H أن سواقة القرص الصلب الثانية هي المطلوبة.لا يمكـــن أن نقوم بعنونة أكثر من سواقتي قرص صلب من خلال الـــ BIOS.

وُجدت توابع الـــ BIOS الخاصة بالقرص الصلب منذ مقدمة XT .إن الـــــ BIOS الخاصـــة بالحاسب العادية لا تمتلكهم.في عام 1981 لم يفكر أحد في وضع سواقات القرص الـــصلب فـــي الحواسب المصغرة.عندما تم طرح الحواسب AT & PS/2 من قبل شركة IBM فقـــد أضـــيفت بعض التوابع الإضافية كما هو مبين في الجدول التالي:

Function	Task	Origin	Function	Task	Origin
00H	Reset	XT	0CH	Move read/write head	XT
01H	Read status	XT	0DH	Reset	XT
02H	Read	XT	0EH	Controller read test	only PS/2
03H	Write	XT	0FH	Controller write test	only PS/2
04H	Verify	XT	10H	Drive ready?	XT
05H	Format	XT	11H	Recalibrate drive	XT
08H	Check format	XT	12H	Controller RAM test	only PS/2
09H	Adapt to foreign drives	XT	13H	Drive test	only PS/2
0AH	Extended read	XT	14H	Controller diagnostic	XT
0BH	Extended write	XT	15H	Determine drive type	AT

لتتصل البرامج التطبيقية العادية إلى سواقة القرص الصلب من خلال الـــ BIOS .سوف نوضح فقط التوابع الأكثر أهمية في هذه البحث.

2.4) شيفرة الحالة

تستخدم توابع القرص الصلب علم الحمل carry flag لتشير إلى وجود خطأ.إذا كان علم الحمل يساوي واحد إذا هناك خطأ قد حدث وتُعاد شيفرة الحالة للخطأ في المـــسجل AH .يبـــين الجدول التالي معاني هذه الشيفرات:

شيفرات الخطأ عندماتستدعي BIOS المقاطعة 13 من أجل الوصول للقرص الصلب			
الشيفرة	الوظيفة	الشيفرة	الوظيفة
00h	لا يوجد خطأ	10h	خطأ قراءة
01h	رقم التابع أو المشغل غير مسموح	11h	تم تصحيح خطأ قراءة من قبل ECC
02h	لم يجد العنوان	20h	يوجد خلل بالمتحكم
04h	لم يجد عنوان القطاع	40h	فشلت عملية البحث
05h	خطأ في عملية Reset للمتحكم	80h	انتهى الوقت,الوحدة لم تستجيب
07h	خطأ أثناء تهيئة المتحكم	AAh	الوحدة غير جاهزة
09h	خطأ إرسال DMA ,تم تجاوز حدود القطعة	CCh	خطأ كتابة
0Ah	يوجد خلل في القطاع		

عندما يحدث أحد هذه الأخطاء(عد الخطأ الأول) فيجب عليك أن تقوم بعملية reset لسواقة القرص الصلب ومن ثم تجرب التابع من جديد.عادة تكون العملية ناجحة.إذا أعيد الخطأ 11H بعد تابع القراءة فإنه ليس من الضروري أن تكون البيانات لاغية.بشكل فعلي فإن شيفرة هذه الحالة تشير إلى أن قد تم كشف خطأ قراءة لكن يمكن تصحيحه باستعمال خوارزمية شيفرة تصحيح الخطأ ECC(Error Correction Code) .هذه الإجرائية شبيهة بإجرائية CRC التي يتم استخدامها من قبل سواقة القرص المرن.تُحسب البايتات الفردية من القطاع sector من خلال صيغة رياضية معقدة.يكتب المجموع الناتج إلى القطاع على القرص الصلب كأربعة بايتات إضافية.إذا تم كشف خطأ قراءة يمكن تصحيحه باستعمال خوارزمية ECC .

3.4) استخدام توابع القرص الصلب:

تستخدم توابع القرص الصلب أيضا المسجلات من أجل تمرير البارامترات.يمرر رقم التابع في المسجل AH .عندما يكون من الواجب التعريف عن رقم سواقة القرص الصلب,فإن قيمته تمرر في المسجل DL .إن القيمة 80H تمثل سواقة القرص الصلب الأولى في حين تمثل

القيمة 81H سواقة القرص الصلب الثانية.تُمرر عدد رؤوس القراءة والكتابة وأي وجه من القرص(0 أو 1) في المسجل DH. يخصص المسجل CH من أجل عدد الاسطوانات cylinders .بما أنه يمكنك تمثيل فقط 256 اسطوانة بمسجل ذو 8bits و سواقة القرص الصلب لشركة XT تملك أكثر من 306 اسطوانات فإن هذا المسجل غير كافي لتمثيل عدد الاسطوانات في القرص الصلب .لهذا السبب فإن البتين السادس والسابع من المسجل CL يُضافان إلى القيمة الموجودة في المسجل CH لتحدد عدد الاسطوانات.حيث يصبح العدد الأعظمي للاسطوانات الذي يمكن تمثيله 1024 اسطوانة(ترقم من 0 إلى 1023).تحدد البتات الخمسة الأولى من المسجل CL رقم القطاع sector number (من 1 إلى 17 من أجل كل اسطوانة).إذا كان هناك أكثر من قطاع يمكن الوصول إليه في نفس الوقت فإن المسجل AL يحدد عدد القطاعات .يجب أن تحدد في عمليات القراءة والكتابة عنوان الذاكرة المؤقتة Buffer من أجل البيانات التي سيتم كتابتها أو البيانات التي سيتم نقلها.في هذه الحالة يشير المسجل ES إلى عنوان القطعة والمسجل BX إلى عنوان الإزاحة للذاكرة المؤقتة Buffer .

4.4) القيام بعملية Reset لجهاز التحكم بالقرص الصلب:

التابع الوحيد الذي لا يتطلب بارامترات هو التابع 00H الذي يشبه التابع 0dH حيث يقوم بعملية Reset لجهاز التحكم controller .مثلا بعد وقوع خطأ فإن هذا التابع يُنجز بشكل دوري قبل الوصول إلى البيانات التالية.إن البارامتر الوحيد الذي يكون بحاجة إليه هو رقم سواقة القرص الصلب الذي يُمرر بدوره في المسجل DL .

5.4) تحديد حالة سواقة القرص الصلب:

باستخدام التابع 01H يمكنك أن تحدد حالة سواقة القرص الصلب.أيضا رقم السواقة التي تريد فحص حالتها يُمرر في المسجل DL .

6.4) قراءة قطاعات القرص الصلب:

يقوم التابع 02H بقراءة قطاع أو أكثر من القرص الصلب.عند كل استدعاء لهذا التابع يمكنك قراءة 128 قطاع كعدد أعظمي.ربما أنت متعجب لماذا العدد الأعظمي هو 128 بدلا من

256 قطاع.إن جهاز التحكم الخاص بالقرص الصلب يستخدم إمكانيات ذاكرة الوصول المباشر للذاكرة DMA(Direct Memory Access) لنقل البيانات بين ذاكرة الحاسب والقرص الصلب.مهما يكن فإن عناصر DMA تستطيع أن تنقل 64K من البيانات في نفي الوقت كعدد أعظمي.هذا يكافئ 128 قطاع (64k-128 sectors * 512 byte/sector) .هناك قيد آخر هو أن عناصر DMA يمكنها أن تنقل البيانات ضممن قطعة الذاكرة الواحدة.لذلك فإن الذاكرة المؤقتة للقراءة أو الكتابة يشير عادة إلى بداية قطعة الذاكرة.تذكر بأن زوج المسجلات ES-BX يحدد الـ Buffer .في هذه الحالة سيشير المسجل ES لإلى بداية هذه القطعة أما المسجل BX فستكون قيمة الإزاحة المخزنة فيه مساوية للصفر.

عندما تستعمل التابع 02H لتقرأ أكثر من قطاع في كل استدعاء له فإنه يتم قراءة القطاعات بالشكل التالي:أولا,القطاعات في الاسطوانة المحددة وأي جانب تقرأ بترتيب متصاعد(بواسطة رقم القطاع).

عندما يتم الوصول إلى نهاية الاسطوانة فإن القطاع الأول على نفس الاسطوانة لكن على الرأس التالي يتم قراءته.لن يتم قراءة القطاعات على الاسطوانة التالية حتى بعد أن يتم الوصول إلى آخر رأس في نفس الاسطوانة ولم يبقى هناك أي قاعات أخرى للقراءة.

7.4) كتابة قطاعات على القرص الصلب:

يستخدم التابع 03H من أجل كتابة قطاع أو أكثر على القرص الصلب.هذا التابع مشابه للتابع 02H ماعدا أن البيانات تُكتب من الذاكرة المؤقتة Buffer لإلى القرص الصلب.

8.4) التحقق على قطاعات القرص الصلب:

يعرف التابع 04H قطاعات كل اسطوانة.مهما يكن فإن البيانات على القرص الصلب تُقارن مع قيمة ECC بدلا من المقارنة مع البيانات الموجودة ف الذاكرة(التي ليس من الضروري أن نحدد عنوان Buffer في زوج المسجلات ES-BX).يحدد عدد القطاعات التي نود التأكد منها ضمن المسجل AL .

9.4) تهيئة اسطوانات القرص الصلب:

يجب أن تتم تهيئة القرص الصلب قبل استخدامه.ينجز التابع 05H هذه المهمة.هذا التابع يشبه التابع المستعمل من أجل تهيئة القرص المرن.عنوان الـ Buffer يُمرر إلى زوج المسجلات ES-BX .يجب أن يكون حجم هذا الـ Buffer هو 512Bytes على الرغم من أن فقط أول 34Byte تُستخدم.يتألف الـ Buffer من مدخلين كل منها بحجم 1Byte من أجل كل قطاع من القطاعات الذي يبلغ عددها 17 قطاع لكل اسطوانة لتتم تهيئته.يشير أول بايت فيما إذا كان القطاع جيد أم سيئ.قبل استدعاء هذا التابع نفترض أن كل قطاع هو جيد.لذلك نخزن القيمة 0 هنا.أما البايت الثاني فهو يشير إلى رقم القطاع المنطقي.

يستخدم البايت الأول والثاني من الجدول عندما يكون القطاع الفيزيائي الأول من الاسطوانة قد تمت تهيئته. يستخدم البايت الثالث والرابع من الجدول عندما يكون القطاع الفيزيائي الثاني من الاسطوانة قد تمت تهيئته,وهكذا....بينما التسلسل الفيزيائي ثابت.يُعرف التتالي المنطقي للقطاعات من قبل بايتين من القطاع يُحددا في هذا الجدول.

إن الطريقة الأكثر وضوحا والمستخدمة في عملية التهيئة هي التي تشير إلى رقم القطاع الفيزيائي للقطاع إلى كل قطاع منطقي.مهما يكن فإن تقنية تستدعي sector interleaving تستخدم فعليا لتسريع أداء القرص الصلب.

يحوي البايت الأول من كل table entry على القيمة 00H التي تشير إلى أن القطاع جيد أو القيمة 80H التي تشير إلى أن القطاع سيئ.أثناء عملية التهيئة Formatting يتم نفل هذا البايت إلى الجزء sector marker الذي يشير إلى أن نظام التشغيل DOS لن يستخدم هذا القطاع ليخزن البيانات.

10.4) تحديد بارامترات القرص الصلب:

بشكل غير مشابه للأقراص المرنة فإن الأقراص الصلبة لا تمتلك خصائص موحدة.يكون من المهم لبعض البرامج أن تعرف بارامترات القرص الصلب.لفعل هذا استخدم التابع 08H لتمرر رقم القرص الصلب في المسجل DL .

بعد استدعاء التابع فإن المسجل DL يحوي عدد الأقراص الصلبة المتصلة بجهاز التحكم.يتم إرجاع القيمة إما 0 أو 1 أو 2 .يحوي المسجل DH عدد رؤوس القراءة والكتابة.بما أن هذه

القيمة تبدأ من الصفر فإن القيمة 7 تعني وجود ثمانية رؤوس.يتم إرجاع عدد الاسطوانات في المسجل CL و البتين العلويين من المسجل CH .أيضا هذه القيمة تبدأ من الصفر.أخيرا يتم إرجاع عدد القطاعات في كل مسار track في البتات الست الأقل أهمية من المسجل CH لكن هذه القيمة تبدأ من 1 وليس من الصفر.

11.4) تشغيل قرص صلب نوعيته غريبة:

إن الـ BIOS في كل حاسب تتضمن مواصفات عدد من الأقراص الصلبة المتنوعة.هذا يجعل من السهل أن تختار مواصفات القرص الصلب أثناء عملية التنصيب setup .مهما يكن افترض أن مواصفات من أجل قرص صلب محدد غير موجودة ضمن الـ BIOS .عندها يوجد طريقة أخرى لجعل الـ BIOS تتعرف على مواصفات هذا القرص.أولا ينشأ جدول يحتوي على المواصفات.ومن ثم يخزن عنوان الجدول عند المقاطعة 41H أو المقاطعة 46H وذلك اعتمادا على فيما إذا كان رقم القرص الصلب المراد تشغيله 0 أو 1 .تنجز صيغة الجدول من قبل الـ BIOS ويصف خصائص القرص الصلب.

أخيرا يستدعى التابع 09H الذي يشغل جهاز التحكم مع خصائص القرص الصلب الجديد.يُمرر رقم السواقة (80H أو 81H) في المسجل DL .عادة يزود مشغل الجهاز من قبل الشركة المنتجة للقرص الصلب.

12.4) كتابة/قراءة قطاع قرص صلب موسع:

إن التوابع 0AH & 0BH شبيه بالتوابع 02H & 03H المخصصة لقراءة وكتابة القطاعات.مهما يكن فهناك اختلاف واحد فقط هو أنه بالإضافة إلى الـ512Bytes المخصصة للمعطيات من أجل كل قطاع ينم نقله فإنه يوجد أربعة بايتات إضافية مخصصة لخوارزمية ECC عند نهاية كل قطاع يتم نقلها أيضا.بما أن كل قطاع أصبح يتكون من 516Bytes بدلا من 512Bytes فإن العدد الأعظمي للقطاعات التي يمكن قراءتها أو كتابتها في نفس الوقت هو 127 قطاع ,بينما كانت التوابع 02H,03H تعالج 128 قطاع.

يفحص التابع 10H فيما إذا كان القرص الصلب الذي مُرر رقمه في المسجل DL هو جاهز لأن ينفذ أوامر.إذا كان علم الحمل يحوي القيمة 1 فذلك يشير إلى أن السواقة غير جاهزة,وعندها فإن المسجل AH سيحوي قيمة شيفرة الخطأ.

13.4) معايرة القرص الصلب

يستخدم التابع 0BH لمعايرة القرص الصلب.بعد استدعاء التابع ,يعيد هذا التابع حالة الخطأ متماشيا مع رقم القرص في المسجل DL .

14.4) اختبار جهاز التحكم بالقرص الصلب

يستخدم التابع 14H لعملية الاختبار,إذا اجتاز جهاز التحكم هذا الاختبار فإن علم الحمل سيتضمن القيمة 0 .

تابع المقاطعة الأخير المخصص للقرص الصلب هو 15H الذي يكون متوفرا فقط على ATs وليس على XTs .تعيد هذه المقاطعة نوع القرص.يمرر رقم القرص(80 or 81) في المسجل DL .إذا لم يكن القرص متوفرا سيتم إرجاع القيمة 0 في المسجل AH .تشير القيمة 2 1or إلى سواقة القرص المرنة.بينما تشير القيمة 3 لإلى سواقة القرص الصلب.في هذه الحالة تحتوي المسجلات CX and DX على عدد القطاعات لهذا القرص الصلب.يشكل هذا المسجلان 32bit ,حيث يحوي المسجل CX البايت الأكثر أهمية بينما يحتوي المسجل DX البايت الأقل أهمية.

5) سواقات الأقراص الصلبة وأجهزة التحكم الخاصة بها:

إن التغيرات الإيجابية التي طرأت على القرص الصلب متعلقة بأربعة أنواع من المتحكمات هي:ST506,ESDI,SCSI,IDE. لا تعتمد الصيغة التي تخزن بها البيانات على القرص الصلب على جهاز التحكم controller لكن تعتمد أيضا على معدل نقل البيانات بين الحاسب والقرص الصلب.

يبين الجدول التالي معدلات نقل البيانات العظمى الخاصة بالمتحكمات.تصور مجموعة من البيانات وهي بطريقها من القرص الصلب ليتم إظهارها على الشاشة (مثلا عندما تحمل مستند إلى

معالج نصوص).بالإضافة إلى جهاز التحكم,على البرنامج أن يكون له interface مع عدة مستويات مثلا BIOS وDOS وبرنامج التطبيق وربما مع واحد أو أكثر من البرامج المقيمة في الذاكرة TSR .

معدلات نقل البيانات العظمى لمتحكمات الأقراص الصلبة المختلفة			
المتحكم	معدلات نقل البيانات الأعظمي	المتحكم	معدلات نقل البيانات الأعظمي
ST506	1 ميغا بت كل ثانية	IDE	4 ميغا بت كل ثانية
ESDI	2.5 ميغا بت كل ثانية	SCSI	5 ميغا بت كل ثانية

هناك عامل آخر يؤثر على معدل نقل البيانات هو سرعة الباص speed of bus .هذا يحد من سرعة المتحكمات بالقرص الصلب لأنه على الأقل الباص من النوع ISA لا يزال يعمل عند التردد 8MHZ على الرغم من أن سرعة وحدة المعا لجة المركزية قد وصلت إلى 100MHZ تقريبا.سنتحدث عن هذه المتحكمات بشيء من التفصيل.

1.5) المتحكم ST506 :

صممت الأقراص الصلبة لتركب على متحكمات ST506 التي ميزت باللافتة MFM/RLL .تشير هذه الأحرف إلى طريقتي تسجيل التي بواسطتها تقوم المتحكمات بحفظ البيانات على القرص الصلب.عادة يمكنك استخدام DIP switches لتختار الشكل الذي تريد استخدامه من المتحكمات.كما أن متحكمات ESDI,IDE تتوافق مع هذه المتحكمات بعدة طرق.

في هذا النوع من المتحكمات يكون القرص الصلب والمتحكم جهازان منفصلان.حيث يكون المتحكم على كرت منفصل.

يمكن لهذا المتحكم أن يدير قرصان صلبان عادة.اثنان من أنواع الكابلات يتصل بهما المتحكم مع كل قرص,يتصل كل قرص صلب مع المتحكم من خلال 20pins من كبل المعطيات الخاص به.وفي حين تم وصل قرصان صلبان فهما يتشاركا في pins 34 من كبل المتحكم.يرسل الكبل إشارات كهر بائية إلى القرص الصلب ليختار رؤوس القراءة والكتابة التي يريدها,والبحث عن الاسطوانة التي يريد القراءة منها أو الكتابة عليها.إن المعطيات التي يتم قراءتها أو كتابتها تنقل

عبر الكبل بشكل تشابهي وتسلسلي.يمكن للمتحكم أن يحول المعلومات الرقمية على الاسطوانة إلى سلسلة من البتات.

تتواجد المعلومات على القرص المغناطيسي كوحدات و أصفار.يمكن للمتحكم أن يقلب القيم الرقمية حسب الحاجة وتدعى هذه المعالجة عكس التدفق flux reversal .

يمكن أن يصل معدل نقل البيانات إلى megabits/second 5 كقيمة عظمى باستخدام تقنية التسجيل MFM و megabits/second 7.5 باستخدام تقنية التسجيل RLL ,حيث أن هذه القيم النظرية تتجاهل عوامل كثيرة كزمن اختيار القرص الصلب,زمن الوصول إلى الاسطوانة,وتفترض أن جميع القطاعات التي تريد قراءتها متجاورة.

أول ما ظهرت الحواسب الشخصية كانت فقط المتحكمات من النوع ST506 متوافرة لكن الإمكانيات المحدودة لهذه المتحكمات جعلت توابع القرص الصلب محدودة الإمكانيات أيضا مثلا عدد الأقراص الصلبة 2 ,العدد الأعظم للاسطوانات 1024 ,العدد الأعظم للقطاعات من أجل كل مسار track هو 63 قطاع,والعدد الأعظمي للرؤوس هو 16 ,أيضا حجم القطاع ثُبت على الرقم 512Byte,وكل هذه الأرقام جعلت السعة العظمى للقرص الصلب هي 504Meg. إن المتحكم ST506 يكون بطيء جدا في التعامل مع الأقراص الصلبة ذات الحجم الكبير لذلك فإنه ليس من العملي وصل أقراص صلبة ذات حجم كبير مع متحكمات من النوع ST506 .

2.5) متحكمات ESDI :

إن ESDI عبارة عن اختصار لـ Enhanced Small Devices Interface أي وصلة الأدوات الصغيرة المحسنة وهي تتواجد في أجهزة IBM PS/2 .وهي مطورة عن المتحكم ST506 .لا تنقل المتحكمات ESDI كل تدفق عكسي بشكل تسلسلي عن طريق خط البيانات كما هو في متحكمات ST506 وإنما بدلا من ذلك جزء من decoding logic يدعى data separator يكون متواجدا على القرص الصلب.يحضر هذا الجزء المعطيات التي تمت قراءتها من القرص الصلب وينقل فقط البيانات المفيدة بشكل رقمي إلى المتحكم.

بما أن المتحكم controller و مقسم المعطيات data separator يعملان بشكل تفرعي parallel فإن معدل نقل البيانات يمكن أن يصل إلى 10megabits/second كحد أعظمي.هذا هو التحسين

الذي قدمته هذه المتحكمات.تستعمل عادة هذه المتحكمات قطاع ذاكـــرة مؤقتـــة sector buffer الذي يجعل العامل interleave يصل إلى 1 تقريبا.ذلك يعني أنه في الأقراص ذات المتحكمـــات ST506 مع عامل interleave يساوي 6 أي ذلك يتطلـــب 6 دورات لنقـــل محتويـــات مـــسار كامل,بينما في المتحكمات النوع ESDI فهي قادرة على أن تنجز نفس المهمة بدورة واحدة.هـــذا ناتج عن الزيادة في سرعة الوصول نتيجة factor interleave من 3 إلى 6. يمكن أن يـــصل سرعة نقل البيانات في بعض أنظمة متحكمات ESDI إلى 15,20 حتى 24megabits/second .لكن هـــذه المتحكمـــات قليلـــة وغاليـــة.لـــذلك فـــإن معظـــم متحكمـــات ESDI تعمـــل بـــــ 10megabits/second .

إن القرص الصلب الذي يتعامل مع متحكمات ESDI يخزن أيضا معلومات عن شكله الفيزيائي و عناوين القطاعات التي يوجد فيها خلل ويرسل هذه المعلومات إلى المتحكم.ومن ثم يمكن للمتحكم أن ينجز عملية التنصيب الخاصة به.

العلاقة بين BIOS ومتحكمات ESDI :

إن المعلومات الخاصة بالقرص الصلب يتم تخزينها على ذاكرة CMOS لأجهـــزة AT .يجب على الـــ BIOS أن تعرف بارامترات القرص الصلب وتمرر هذه المعلومات إلى مـــشغل النظام DOS . لأنه يمكن لمتحكمات ESDI أن تطلب هذه المعلومات من القرص الصلب,وليس من الضروري أن تكون هذه المعلومات متلائمة مع خصائص القرص الصلب الفعلية وطبعـــا لا يمكن تجنب هذا التناقض,لأن كل BIOS تعرف عددا محـــددا مـــن أنـــواع الأقـــراص الـــصلبة وخصائصها.

عليك أن تختار القرص الصلب من القائمة الموجودة ضمن الـــ BIOS والتي تكون خصائـــصها قريبة جدا من خصائص القرص الصلب الذي تود تنصيبه.وذلك يعني طبعا ضياع بعض من حجم القرص.عليك أن تختار القرص الذي يملك اسطوانات ومسارات ورؤوس أقل وذلـــك فـــي خـــال التعامل مع متحكمات من نوع ST506 وإلا ربما يحاول النظام أن يصل إلـــى قطاعـــات غيـــر متواجدة على القرص الصلب وذلك بدوره يسبب ظهور خطأ.

يوجد مشكلة أخرى عندما لاتملك الـ BIOS نوع القرص الصلب الـذي يعمـل بـنفس عـدد القطاعات في كل مسار.عندها يجب أن تختار القرص الذي يملك حجم القطاع الأصـغر التـالي وذلك يعني ضياع بعض القطاعات المتاحة في كل مسار على القرص الصلب وبالتالي يزداد عدد القطاعات الغير مستخدمة إلى المستوى الغير مرغوب.

هذه المشكلة غير متواجدة في مشغلات ST506 لأنها تعمل فقط إما بـ 17 قطاع أو 26 قطـاع ,وذلك يعتمد على طريقة التسجيل.تملك مشغلات ESDI إما 34 أو 36 قطاع في كل مسار.لذلك إذا كانت تحتوي الـ BIOS تعريفات لأقراص ذات 26 قطاع فربما تضيع تقريبـا ثلـث سـعة القرص الصلب.

لحسن الحظ يمكن لمتحكمات ESDI أن تتجنب هذه المشكلة,حيث يمكن للأقراص الـصلبة التـي تتعامل مع هذا النوع من المتحكمات أن ترسل مواصفاتها إلى المتحكم ESDI عندها يتمكن هـذا المتحكم من التعرف على المواصفات الفيزيائية للقر,باستخدام ميزة خاصة تدعى ترجمة القطـاع sector translation,يمكن للمتحكم ESDI أن يحول مواصفات المشغل المنطقي الخـاص بـ BIOS إلى مواصفات فيزيائية للقرص الصلب.يأخذ هذا التحويل وقتا أطول لكن يضمن قـرص ESDI أن يستخدم تقريبا أي تعريف موجود ضمن الـ BIOS دون ضياع الكثيـر مـن سـعة القرص الصلب.

تعتمد إمكانية إنجاز هذا التحويل على المتحكم ESDI .تدعم بعض المتحكمـات فقـط تعريفـات BIOS معينة بينما المتحكمات الأخرى أكثر مرونة ويمكنها أن تقبل أي شكل.هـذا أيـضا أحـد التحسينات التي تمتلكها متحكمات ESDI في حال واجهت مشكلة الإمكانيات المحدودة مـن قبـل توابع القرص الصلب الموجودة لدى BIOS .

3.5) متحكمات SCSI :

هي أداة تمكنك من أن تصل ثماني أجهزة مختلفة إلى الحاسب الشخصي.يمكنك أن توصل إلى جانب الأقراص الصلبة الأجهزة التالية tape backup streams, CD-ROM, scanners. وهي اختصار للعبارة التالية Small Computer System Interface ولا تتواجد فقـط علـى الحواسب الشخصية ولكن على أكثر من 68000 نظام(Macintosh and Atari ST) & large

workstation .هناك سبب وحيد فقط يجعلك تلجأ إليها وهو أنها تمكنك من ربط أو فك ربط الأجهزة مع SCSI interface لأن الأجهزة تتصل مع المتحكم من خلال bus الذي يكون منفصلا عن الباص الخاص بالحاسب PC bus.

إن الباص الخاص بالمتحكم SCSI الذي يربط عدة أجهزة مع بعضها عادة يكون عبارة عن كابل في نهايته وصلة تمتلك 80 pin .يسمح هذا الباص بنقل معطيات تفرعي على شكل 8 bits ويوجد الآن إصدار جديد من هذا المتحكم يسمح بنقل المعطيات تفرعي على 16 bits. وإن معدل نقل البيانات لهذه المتحكمات هو مابين 1.5 & 2 mega per second على الرغم من أن السرعة النظرية تصل إلى 4 or 5 mega per second ولكن المشكلة في سرعة الأقراص الصلبة التي تتصل مع هذه المتحكمات.

تعتمد أنظمة SCSI على هيكلتها الخاصة ضمن BIOS.من وجهة نظر البرمجية فإن أنظمة SCSI لا تدعم أنظمة ST506 القياسية.لسوء الحظ فإن SCSI BIOS غير نافعة في النمط المحمي والذي فيه تصل بيئات التشغيل مثل Novell or OS/2 مباشرة إلى القرص وتدعم فقط المتحكمات ST506 القياسية.حيث يتطلب ذلك device drive معين الذي ربما يكون غير متوفر من أجل بيئات تشغيل معينة.يعتبر هذا الموضوع من مساوئ متحكمات SCSI .

على أية حال إذا كنت تملك المشغل المطلوب,فإن مواصفات القرص الصلب آلية.ولكي تنصب مشغل SCSI جديد فقم بوصله إلى الباص ببساطة,عندها يعالج المتحكم SCSI ما بقي لديه من خلال استخدام driver ware الصحيح.

4.5) متحكمات IDE :

هي اختصار Intelligent Drive Electronics أي الكترونيات الدفع الذكية وهي متواجدة في أغلب الحواسب الشخصية الجديدة.وتكون هذه المتحكمات متصلة مع باص النظام مباشرة. يمكن أن تعمل هذه المتحكمات بـ 1:1 interleave factor الذي يؤمن سرعة وصول أكبر.

يكون استهلاك الطاقة لمشغلات IDE منخفض جدا لذلك فهي مناسبة من أجل الأجهزة المحمولة و دفاتر الملاحظات الالكترونية.إن أغلب مشغلات IDE يكون لها أوامر خاصة من أجل عمل

الأجهزة المحمولة ودفاتر الملاحظات.على سبيل المثال يمكن أن تضع هذه الأوامـــر notebook computer في حالة sleep وذلك يخفض استهلاك الطاقة إلى أقل مستوى له.

5.5) من المتحكم إلى الذاكرة

بغض النظر عن سرعة القرص الصلب والمتحكم فإن الطريقة التي تنقل بها البيانات من قبل المتحكم إلى الذاكرة تتحدد بالسرعة الفعالة للتركيب hard drive-controller .يمكن أن تستخدم أربع طرق لفعل هذا:

1. أجهزة الدخل الخرج المبرمجة programmed I/O(PIO).
2. الوصول المباشر للذاكرة DMA .
3. خريطة ذاكرة الدخل/خرج Memory mapped I/O .
4. الباص الرئيسي للوصول المباشر للذاكرةBusmaster DMA .

- أجهزة الدخل الخرج المبرمجة programmed I/O(PIO):تدير منافذ الدخل والخرج المختلفـــة للمتحكم كلا من أوامر المشغل و نقل المعطيات من المتحكم إلى الذاكرة الرئيسية.إذا اســـتخدمت منافذ الدخل/الخرج المبرمجة فليك أن تستعمل تعليمـــات لغـــة الأســـمبلي الخاصـــة بالإدخـــال والإخراج.هذا يعني أن كل بايت أو كل كلمة يجـــب أن تحـــول مـــن خـــلال وحـــدة المعالجـــة المركزية.إن معدل نقل البيانات سيكون محدد بباص الحاسب وأداء وحدة المعالجة المركزية.بينما يسمح لك باص ISA بمعدل نقل بيانات يصل إلى 5.33 mega per second, هذا المعدل بعيـــد المنال من أجل وحدات المعالجة المركزية المستخدمة حاليا.إن معدل نقل البيانات محدد ب 3 or 4 Meg من أجل معالجات 386es,486es,Pentiums, ويمكن الوصول إلى هذه السرعة فقط من خلال أقراص صلبة سريعة وغالية.
- الوصول المباشر للذاكرة DMA :باستخدام هذه الطريقة فإن أي جهاز (hard drive ,floppy disk drive ,CD-ROM) يمكنها أن تنقل البيانات مباشرة إلى ذاكرة الحاسب,حيث يتم تجـــاوز وحدة المعالجة المركزية.لتستخدم الطريقة DMA فأنت بحاجة فقط إلى برنامج يخبر DMA كم بايت يجب أن تنقل من موضع إلى آخر.وذلك بجعل DMA الطريقة الأفضل لنقل البيانات.

إن جهاز التحكم الخاص بـــ DMA الموجود في الحاسب غير مرن وبطئ.إن معدل نقل البيانات محدد تقريبا 2 mega per second باستعمال هذه الطريقة.

- خريطة ذاكرة الدخل/خرج Memory mapped I/O يمكن لوحدة المعالجة المركزية أن تعالج المعطيات بشكل أسرع من disk controller من أن تخزنها في منطقة ذاكرة محددة.يمكن أن تنقل المعطيات الموجودة في الذاكرة المخصصة للبرنامج بشكل أسرع باستخدام تعليمات النقل MOV .وذلك سيكون أسرع من الوصول إلى منافذ الدخل/الخرج باستخدام تعليمات IN,OUT .
- الباص الرئيسي للوصول المباشر للذاكرةBusmaster DMA :إن هذه الطريقة هي شكل آخر من الوصول المباشر للذاكرة,لكن ليس له علاقة بالدارة DMA الموجودة على اللوحة الأم للحاسب.باستخدام هذه الطريقة,فإن المتحكم الخاص بالقرص الصلب يقوم بفصل وحدة المعالجة المركزية عن الباص وينقل المعطيات إلى الذاكرة على الباص الخاص به باستخدامه المتحكم Busmaster DMA .تصل معدلات النقل إلى 8 meg per second .

النمط المحمي :

إن طرق النقل عن طريق DMA هي محددة من أجل برامج النمط الحقيقي.بدون تدخل خاص الوصول المباشر للذاكرة لا يمكنه أن يستخدم في النمط المحمي أو الافتراضي.وإن المسئول عن هذا التقييد هو كيف تُنجز دارة الذاكرة الافتراضية من قبل وحدة إدارة الذاكرة الخاصة بوحدة المعالجة المركزية Memory Management unit(MMU) .تقوم هذه الوحدة بتحويل عناوين الذاكرة الافتراضية الخاصة بالبرنامج إلى عناوين ذاكرة فيزيائية حقيقية. لا يحقق برنامج النمط المحمي أو الافتراضي هذا لأنه لا يكون مدركا أبدا للعناوين الفيزيائية,حيث يتعامل فقط مع العناوين الافتراضية.عندما يريد هذا البرنامج أن ينجز مهمة نقل للبيانات عن طريق DMA فإنه يمرر العناوين الافتراضية إلى شريحة DMA .مهما يكن,بما أن الوحدة MMU لا تصبح طرفا في عملية النقل عن طريق DMA فإنها لا تستطيع أن تجدول هذا العنوان الافتراضي إلى عنوان فيزيائي.كنتيجة لذلك يتم نقل البيانات إلى منطقة ذاكرية مختلفة.عندها فإن النظام سيكون قد تعرض لحالة crash لأنه تم الكتابة فوق مناطق ذاكرية مهمة. هذه ميزة أنظمة التشغيل التي تعمل في النمط المحمي مثل Windows and OS/2 .

إن الحل هو أن تراقب المتحكم DMA .ولفعل ذلك في النمط المحمي هو أن تتحكم بجميع منافذ الدخل/الخرج.في نظام التشغيل windows مثلا يتم تنصيب مراقب تحكم وهمي في الخلفية

background ليراقب برمجة المتحكم DMA عن طريق الـ BIOS أو برنامج آخر.يحول هذا المراقب العناوين الفيزيائية الفعلية قبل أن تتم كتابتها إلى المسجلات الخاصة بالمتحكم DMA .

6) ميزات سواقات الأقراص الصلبة:

من أهم الميزات الحديثة لسواقات الأقراص الصلبة هو أنه أصبح أسرع و أرخص و أصـــغر حجماً فعلى سبيل المثال:

- تم تقليل زمن الوصول في النماذج ذات الأداء الحديث من حوالي 30ms إلى 10ms
- زيادة سعة التخزين الأعظمية فقد أصبحت سواقات الأقراص الصلبة ذات السعات أكثر من 1GigaByte كثير الانتشار
- أصبحت أسعار سواقات الأقراص الصلبة منخفضة جداً

و سنناقش فيما يلي أهم أسباب تحسين أداء سواقات القرص الصلب.

1.6) ***التوزيع و عامل عملية التوزيع***

حالياً متحكمات سواقة القرص الصلب سريعة جداً بحيث أنها تستطيع مسار كامل حتى لو كانت المعطيات المطلوبة موجودة في قطاع واحد. و هكذا إذا طلبت المعطيات الموجـــودة فـــي القطاع التالي فلا يقوم المتحكم بالقراءة من القطاع و لكن يتم أخذ المعطيات مع الذاكرة الداخليـــة للمتحكم و هذا يزيد من سرعة الوصول للمعطيات بشكل كبيـــر. و تملـــك المتحكمـــات ST506 ذاكرة داخلية تتسع لقطاع واحد و لكن المتحكمات الحديثة من النوع (SCI ,ESDI ,and IDE) تملك ذاكرة داخلية تتسع لمسار كامل .

Interleaving in the first XT and AT hard drives from IBM			
AT		XT	
Physical sectors	Logical sectors	Physical sectors	Logical sectors
1	1	1	1
2	7	2	4
3	13	3	7
4	2	4	10
5	8	5	13
6	14	6	16
7	3	7	2
8	9	8	5
9	15	9	8
10	4	10	11
11	10	11	14
12	16	12	17
13	5	13	3
14	11	14	6
15	17	15	9
16	6	16	12
17	12	17	15

و لا يتم إعادة ملى الذاكرة ذات سعة القطاع إلا بعد نقل آخر بايت إلى .CPU وهذا يتطلب مزيداً من الوقت وهذا الوقت كافي ليمر القطاع التالي من تحت رؤوس القراءة و الكتابة و بالتالي لقراءة المعطيات من القطاع التالي سيتطلب هذا انتظار حوالي دورة كاملة للقرص الصلب و بالتالي إعادة تنفيذ هذه العملية سيودي إلى بطيء ملحوظ في زمن الوصول.

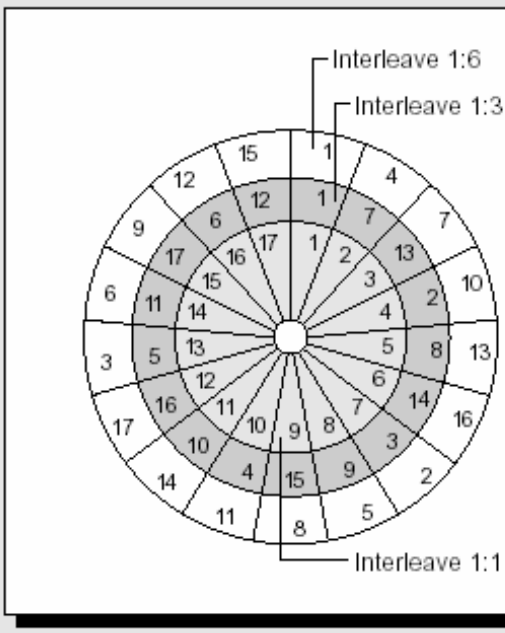
و بالتالي لتقليل هذا التأخير يتم استخدام ما يسمى عملية التوزيعInterleaving لنشر القطاعات في المسار. و بتوزيع القطاعات منطقياً يتاح وقت كافي للمتحكم لمعالجة و إرسال المعطيات من قطاع واحد قبل أن يمر القطاع ذو الترتيب المنطقي التالي تحت رأس القراءة, و بهذا يتم تجنب انتظار دورة كاملة قبل قراءة القطاع التالي. و يتم قياس التوزيع بعامل عملية التوزيع و قيمته هي عدد القطاعات و التي تم إزاحتها برقم القطاع المنطقي عن رقم القطاع الفيزيائي. في سواقة القرص الصلب لجهاز XT يكون عامل عملية التوزيع مساوياً لـ 1:6 أما في أجهزة AT فهو 1:3 و يمكن أن يقلل إلى 1:2 مما يزيد سرعة الوصول. و حالياً القيمة 1:1 لهذا العامل منتشرة

جداً مما يعني أنه لا يوجد توزيع على الإطلاق كما في الجدول السابق حيث يوجد 17 قطاع , و بغض النظر عن أهمية التوزيع فإن أفضل عملية التوزيع تكون عندما لا توجد حاجة للتوزيـــع. و بذلك يمكن قراءة مسار كامل ضمن دورة واحدة للقرص فقط. أما بوجود توزيع فـــسيكون هنـــاك حاجة لعدد من الدورات 3,2 و ذلك بحسب قيمة عامل عملية التوزيع.

2.6) إسناد عملية التوزيع

يتم ذلك خلال عملية التهيئة ذات المستوى المنخفض لسواقة القرص الصلب

Different interleave factors and their influence on the formatting of a track



و بما أن الرقم المنطقي للقطاع يتم تحديده باستخدام برامج التهيئة ذات المستوى المنخفض فمـــن الممكن إزاحة القطاعات. و معظم البرامج المخدمة لسواقة القرص الصلب ستعطيك عامل عملية التوزيع المناسب. إذا كانت القيمة سيئة سيؤدي هذا إلى بطء شديد في الوصـــول إلـــى الملفـــات الموجودة على سواقة القرص الصلب.

3.6) انحراف الأسطوانة و المسار :

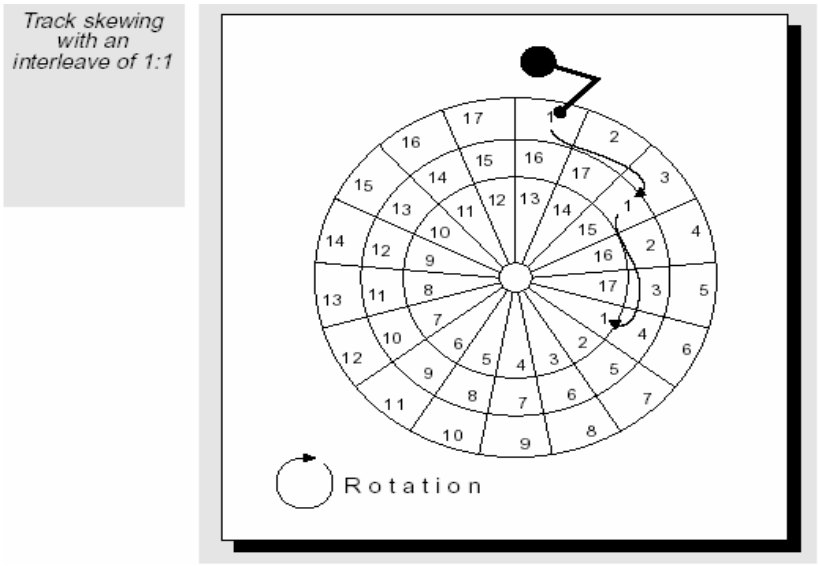
اختيار عامل عملية التوزيع المناسب مهم جداً لتحسين أداء سواقة القرص الصلب. و بعد قراءة المعطيات من مسار معين يتم عادة قراءة المعطيات من على نفس الأسطوانة و لكـــن مـــن

Cylinder skewing

رأس قراءة آخر و بينما يقوم المتحكم بالتبديل بين رؤوس القراءة يستمر القرص الصلب بالدوران و بعد قراءة آخر قطاع من المسار فإن القطاع الأول من الرأس التالي سيكون قد مر من تحت رأس القراءة و لذلك يجب الانتظار حوالي دورة كاملة.

و لمنع هذا من الحدوث يتم استخدام انحراف الاسطوانة حيث يتم إزاحة كل القطاعات على المسار التالي, و بالتالي بعد تبديل رأس القراءة يتم قراءة القطاع الأول بدون تأخير دوراني. ويتم تحديد انحراف الاسطوانة خلال عملية التهيئة ذات المستوى المنخفض.

و بالإضافة لعملية انحراف الاسطوانة توجد عملية انحراف المسار و هي مشابهة لعملية انحراف الاسطوانة و لكنها تقوم بالاهتمام بالوقت اللازم لسواقة القرص الصلب لنقل ذراع القراءة إلى المسار التالي



4.6) تصحيح الخطأ

تعتبر الموثوقية من أهم مؤشرات عملية التخزين. إن حدوث التلوث في المادة المغناطيسية شيء لا يمكن تجنبه و يمكن أن يسبب عدم استخدام لبعض الأجزاء من سواقة

القرص الصلب.في ما مضى كانت تطبع لائحة بالقطاعات المصابة على حافظة (غلاف) سواقة القرص الصلب كما تم تحديدها من قبل المصنعين و خلال عملية التهيئة ذات المستوى المنخفض يقوم المستخدم بإدخال هذه الأرقام ليتعرف عليها نظام التشغيل و يتم تعليمها على أنها مصابة في جدول توزيع الملفات FAT و يتجنب تخزين المعطيات فيها .

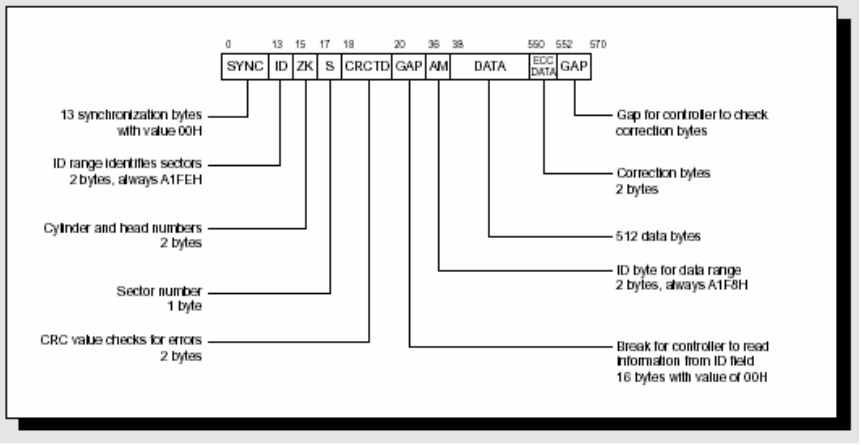
أما النماذج الحديثة لـ IDE ليس لها لائحة قطاعات مصابة و يتم تخزينها على مسار معين منفصل و تتعرف عليها خلال عملية التهيئة ذات المستوى المنخفض. و العديد من السواقات تستطيع تمييز قطاعات مصابة عن طريق عمليات القراءة و الكتابة و هذا هو هدف ECC (شفرة تحديد الخطأ) و التي يتم توليدها أوتوماتيكياً و تخزينها على كل قطاع و هذه الميزة موجودة في سواقة القرص الصلب IDE.

5.6) مكونات سواقة القرص الصلب الأخرى :

عندما يتم تهيئة سواقة القرص الصلب فإنه يتم تخزين مجموعة معطيات كاملة على سواقة القرص الصلب و نمط هذه المعطيات يختلف بحسب المتحكم (مثل رقم الأسطوانة, رقم ECC , ...) حيث يبدأ كل قطاع بسلسلة من 13 بايت متزامنة مع القيمة 00H (حقل البدء) و يلي هذه القيمة حقل رقم التعريف ID و الذي يعرف القطاع. بالإضافة إلى أرقام الأسطوانة, الرأس, القطاع(C,H,S) و تبدأ هذه المعطيات ببايت ID مخصص و ينتهي ببايت شفرة الخطأ تستخدم للتأكد من صحة المعطيات (حقل CRCID)

و من ثم منطقة فارغة gap تعطي المتحكم فرصة قبل بدء بحقل المعطيات و يستخدم المتحكم هذه الفرصة لقراءة قيمة ID و التأكد هل هذا هو القطاع المطلوب. و من ثم يبدأ 512 بايت من المعطيات (حقل Data) و من ثم هناك بايتين يستخدمهما المتحكم للتأكد من صحة المعطيات(حقل ECC-Data) و من ثم منطقة فارغة ثانية gap مملوءة بالقيمة 4EH تعطي المتحكم فرصة للتأكد من صحة المعطيات و بذلك يكون الطول الفعلي الكلي لكل قطاع هي 570 بايت.

Sector format created by an ST506-type controller



مسارات إضافية : و هناك مناطق محجوزة أخرى فهناك مسارات مؤازرة و تستخدمها الدارة الالكترونية لسواقة القرص الصلب لتزامن نفسها مع ساعة المعطيات. و هناك بعض المسارات حتى BIOS لا تراها و هي محجوزة كبديل عن القطاعات المصابة. و أخيراً هناك منطقة الاستراحة (Park area) تستخدم عندما يطفئ جهاز الكمبيوتر حيث تستقر رؤوس القراءة و الكتابة على سطح منطقة الاستراحة. لأنه إذا استقرت رؤوس القراءة و الكتابة على مسار يحتوي معطيات فإنه قد يضر بهذه المعطيات, و حيث أنه لا يوجد معطيات في منطقة الاستراحة.

6.6) زمن الوصول

يقاس أداء سواقة القرص الصلب بسرعة الوصول. و يدعي عادة المصنعون أن سواقة القرص الصلب التي يصنعونها ذات سرعات وصول حوالي 10 أو 30 ميلي ثانية. و تعبر هذه القيمة عن معدل زمن الوصول إلى ملفين مختلفين.

و يستخدم أيضاً المصنعون زمن الوصول من مسار إلى مسار و زمن الوصول الأعظمي ليعبران عن سرعة سواقة القرص الصلب. حيث أن زمن الوصول من مسار إلى مسار هو الزمن اللازم لتحريك رأس القراءة و الكتابة من الأسطوانة إلى أخرى. و زمن الوصول الأعظمي هو الزمن اللازم لتحريك ذراع القراءة و الكتابة من الأسطوانة الأولى إلى الأخيرة للسواقة.

كما أن هذه العوامل لها تأثير مهم على كيفية الوصول إلى سواقة القرص الصلب من مستوى نظام التشغيل و هناك عوامل أخرى فمثلاً : قص بمواصفات سريعة جداً تكون فيه هذه المواصفات عديمة الفائدة ما لم يستطع المتحكم مجاراته. و قد تكون سواقة القرص الصلب مجزئ

جداً بحيث أن ذراع القراءة و الكتابة تقفز بشكل دائم للأمام و للخلف بين الأسطوانات المختلفة للوصول إلى المعطيات.

لذلك يجب أن يقاس الأداء اعتماداً على الزمن الذي تستغرقه طلبات القراءة و الكتابة في المستويات المختلفة للتطبيق. أي على مستوى نظام التشغيل و مشغلات السواقات و مستوى BIOS و من مستوى سواقة القرص الصلب و متحكماته. و هناك تطبيقات تقيس أداء سواقة القرص الصلب.

7) تجزئة سواقة القرص الصلب : Hard Disk Partition

عند تنفيذ تعليمة FDISK من نظام التشغيل DOS فإنها تقوم بتجزئة سواقة القرص الصلب إلى كتل منفصلة منطقياً أو عندما تريد تحميل أكثر من نظام تشغيل على قص واحد. عملية التهيئة لتحضير سواقة القرص الصلب ليتم استخدامها من قبل نظام تشغيل يجب تنفيذ ثلاث تنفيذ مهام . أولاً يجب عليك تنفيذ عملية تهيئة منخفضة المستوى . و عندما تقوم بهذا فأنك تقوم تنظيم سواقة القرص الصلب إلى سواقات و مسارات و قطاعات عن طريق كتابة علامات العنوان المناسب على سطح السواقة . يستخدم المتحكم علامات العنوان فيما بعد لتعيين القطاعات المحددة .

1.7) عملية تجزئة سواقة القرص الصلب

و هناك سببان رئيسيان للقيام بعملية التجزئة. أولاً هذا يمكنك من تحميل أكثر من نظام تشغيل على سواقة القرص الصلب. فتجزئة سواقة القرص الصلب إلى عدة مناطق منفصلة يتيح لكل نظام تشغيل أن يدير جزءه الخاص من سواقة القرص الصلب بدون حدوث اضطرابات عن اختلاف هيكلية أنظمة الملفات.

و السبب الآخر للتجزئة هو حتى يصبح من الممكن استخدام سعات إضافية للسواقات ذات السعات الكبيرة. تمتلك حواسيب XT قص بسعة 10Meg ثم ظهرت سواقات قرص صلب بسعات 40,80 Meg و كانت نسخ Dos القديمة تستطيع التعامل مع سعات 32 Meg فقط و لكن تم التخلص من هذه المشكلة في نسخة Dos 3.3 حيث أصبحت سعة 32 Meg مرتبطة بجزء واحد. حيث قدمت هذه النسخة من DOS إمكانية تجزئة سواقة القرص الصلب لجزء رئيسي

(Primary Portion) و جزء ممتد (Extended Portion) يمكن تجزئته لـــ 23 جهاز منطقي سعة كل منها الأعظمية 32 Meg مما يتيح سعة كليـــة أعظميـــة 768 Meg لـــسواقة القـــرص الصلب. و نسخة Dos 4.0 كانت تدعم أقراص صلبة بسعة GegaByte 2 و لكن المـــستخدمين استمروا بعملية التجزئة لأنهم كانوا يفضلون التعامل مع عدة أقراص منطقية على التعامـــل مـــع قرص واحد كبير السعة.

2.7) قطاع التجزئة

و هو عبارة عن البنية التي تستخدمها جميع نسخ DOS لتعريف تجزيئات سواقة قرصها الصلب. و عندما تقوم بتنفيذ برنامج FDISK لأول مرة فإنه يقوم بإنشاء قطاع التجزئة و ذلـــك على القطاع الأول لسواقة القرص الصلب المعرف بالأرقام (Cylinder0,Track0,Sector1) و يقوم BIOS في عند بدء التشغيل بتحميل قطاع التجزئة بدلاً من قطاع الإقلاع لـــ DOS و بعد أن ينتهي إتمام تشغيل نظام الحاسب يتم تحميل قطاع التجزئة إلى الذاكرة و ذلك بدءاً من العنوان 0000:7C00 و هذا إذا لم يكن هناك أقراص مرنة في السواقة :A.

و إذا وجد BIOS القيمتين 55H وتليها AAH في البايتين الأخيرين من قطاع التجزئة المؤلـــف من 512 بايت فإنه يعتبر هذا القطاع قابلاً للتنفيذ و يبدأ بتنفيذ البرنامج بدءاً من البايـــت الأول من القطاع نفسه و إلا يقوم BIOS بإظهار رسالة خطأ و من ثم يبدأ تنفيـــذ ROM-BASIC أو يدخل في حلقة لا نهائية و ذلك بحسب نسخة BIOS و مصنعيها.

و يقوم هذا البرنامج بتنظيم و تشغيل الجزء الفعال من نظام التشغيل و يتطلب القيام بذلك تحميـــل قطاع الإقلاع لنظام التشغيل و تمرير التحكم للبرنامج الموجود ضـــمن هـــذا القطـــاع و بمـــا أن البرنامج الجديد يجب تحميله إلى الذاكرة أيضاً على العنوان 0000:7C00 فإن برنـــامج قطـــاع التجزئة يتم نقله للعنوان 0000:0600 أولاً لإفساح المجال لقطاع الإقلاع بالعمل.

Structure of the partition sector of a hard drive		
Address	Contents	Type
+000h	Partition code	Code
+1BEh	1.Entry in the partition table	16 BYTE
+1CEh	2.Entry in the partition table	16 BYTE
+1DEh	3.Entry in the partition table	16 BYTE
+1EEh	4.Entry in the partition table	16 BYTE
+1FEh	IDcode(AA55h),which identifies the partition sector as such	2 BYTE
Length: 1EH (30) bytes		

3.7) جدول التجزئة

البرنامج المكتوب في قطاع التجزئة يجب أن يكون قادراً على إيجاد للجـــزء الفعـــال و للقيام بذلك فإنه يستخدم جدول التجزئة و يتم تحميل هذا الجدول في الإزاحة 1BEH من قطـــاع التجزئة.

Structure of an entry in the partition table		
Address	Contents	Type
+00h	Partition status 00h = inactive 80h = Boot-Partition	1 BYTE
+01h	Read/write head, with which the partition begins	1 BYTE
+02h	Sector and cylinder, with which the partition begins	1 WORD
+04h	Partition type * 00h = Entry not allocated 01h = DOS with 12-Bit-FAT (primary Part.) 02h = XENIX 03h = XENIX 04h = DOS with 16-Bit-FAT (primary Part.) 05h = extended DOS-Partition (DOS 3.3) 06h = DOS-4.0 partition with more than 32 Meg DBh = Concurrent DOS	1 BYTE
+05h	Read/write head, with which the partition ends	1 BYTE
+06h	Sector and cylinder, with which the partition ends	1 WORD
+08h	Removal of first sector of the partition (Boot-sector) of partition sector in sectors	1 DWORD
+0Ch	Number of sectors in this partition	1 DWORD
Length: 10h (16 Bytes)		
* Other codes possible combined with other operating systems or special driver software.		

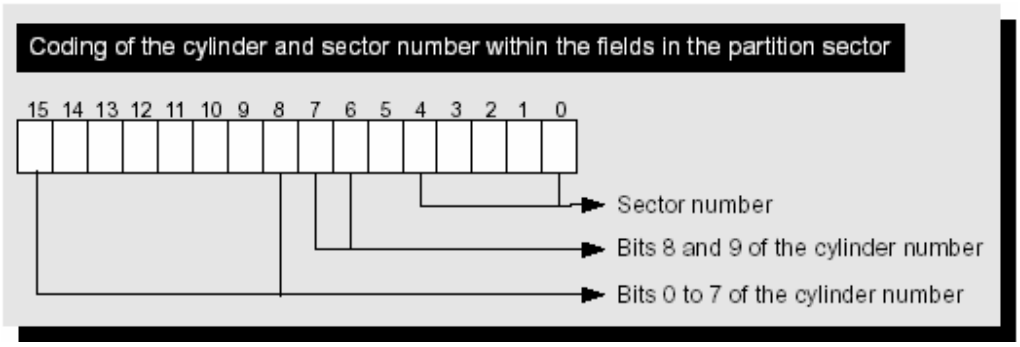
كل مدخل Entry في جدول التجزئة مكون من 16 بايت و هذا الجدول موجود في نهاية قطـــاع التجزئة متيحاً المجال لـــ 4 مداخل لذلك عدد التجزئات محدود بـــ 4 أجزاء. و لإنشاء أكثر من 4 أجزاء يلجأ بعض مصنعي سواقات الأقراص الصلبة إلى استخدام برنامج إعداد خاص.

1في بعض الأحيان يتم تعديل شفرة قطاع التجزئة مما يتيح لك الإقلاع إلى أي من أنظمة التشغيل التي لديك و هذا يجعل الاختيار أسهل بين أنظمة التشغيل عندما يبدأ الحاسب بالعمـــل. و يتـــألف جدول التجزئة للمدخل الواحد كما في الجدول السابق .

4.7) بدء تشغيل قطاع الإقلاع

الحقل الأول من كل مدخل جدول تجزئة يشير إذا ما كان الجزء فعالاً أم لا. القيمة 00H تعني أن الجزء غير فعال و القيمة 80H تعني أن الجزء فعال و يجب الإقلاع منـــه. و إذا وجـــد برنامج قطاع التجزئة أكثر من جزء واحد فعال أو أنه لا يوجد أي منها فعال فإنه لا يتابع عمليـــة الإقلاع و يظهر رسالة خطأ و يدخل في حلقة لا نهائية لا يمكن الخروج منها إلا بإعادة الإقلاع.

و عندما يحدد برنامج قطاع التجزئة جزء فعال واحد فإنه يستخدم الحقلين التاليين لتحديد مكان هذا الجزء على القرص الصلب. و يتم التعبير عن رقم الأسطوانة و القطاع تماماً كما في المقاطعـــة رقم 13 و متضمناً البت السادس و السابع من رقم القطاع.و الذي يمثل البت الثامن و التاسع من رقم الأسطوانة (أي هذين البايتين لهما الشكل الذي يمكننا مباشرة استدعاء مقاطعة BIOS رقـــم 13 بتحديد رقم الأسطوانة و رقم القطاع لها).



في هذا الوقت تكون مقاطعة BIOS رقم 13 و توابعها هي الطريقة الوحيدة للوصـــول لـــسواقة القرص الصلب لأنه من الواضح أن توابع DOS غير متاحة في هذه اللحظة.

و هكذا و بعد التعرف على سواقة القرص المرن و الصلب بإمكاننا كتابـــة البرنـــامج الملحـــق و شرحه.

8) البرنامج الملحق :

يقوم البرنامج الملحق بالعمليات التالية و ذلك بحسب السواقة المراد تطبيق الأوامر عليها

1) سواقة القرص المرن :
   a. التعرف على خصائص سواقة القرص المرن.
   b. قراءة قطاع معين من سواقة القرص المرن.
   c. كتابة قيم معينة على قطاع ما من سواقة القرص المرن.
2) سواقة القرص الصلب :
   a. التعرف على خصائص سواقة القرص الصلب.
   b. قراءة قطاع معين من سواقة القرص الصلب.
   c. قراءة قطاع التجزئة و عرض خصائص كل جزء.

ملاحظة :

بالنسبة للعمليات على القرص الصلب فإنها لا تعمل على نظام Windows XP لأن هذا النظـــام يقوم بحماية القرص الصلب و يمنع التعامل معه باستخدام توابع منخفضة المستوى فـــي عمليتـــي القراءة منه و الكتابة عليه.

كما أن البرنامج لا يتيح إمكانية الكتابة على القرص الصلب لأنها عملية خطرة و اكتفينا بتخـــديم الكتابة على القرص المرن.

```
/*==Link include files =========================================*/

#include <dos.h>
#include <stdio.h>
#include <string.h>

#define SEG( p ) ( ( unsigned int ) ( ( ( long ) p ) >> 16 ) )
#define OFS( p ) ( ( unsigned int ) ( p ) )

/*== Constants ================================================*/

#define NOPE         0x4E                    /* N for No */
#define NO_DRIVE     0                  /* Drive does not exist */
#define DD_525       1                   /* Drive: 5.25" DD */
#define HD_525       2                   /* Drive: 5.25" HD */
#define DD_35        3                   /* Drive: 3.5" DD */
#define HD_35        4                   /* Drive: 3.5" HD */
#define MAXNUMTRIES  5                     /* Maximum number of tries */

#define ReadOp  0x02                /* Read Operation */
#define WriteOp 0x03                /* Write Operation */

#define TRUE  ( 1 == 1 )
#define FALSE ( 1 == 0 )

/*== Macros ===================================================*/

#define HI(x) ( *((BYTE *) (&x)+1) ) /* Returns high byte of word */
#define LO(x) ( *((BYTE *) &x) )      /* Returns low byte of word */


/*== Type declarations ========================================*/
```

```
typedef unsigned char BYTE;                    /* Data type byte */
typedef unsigned int WORD;

typedef BYTE TrackBfType[ 18 ][ 512 ];         /* Memory for a track */

typedef struct {                        /* Floppy disk drive parameters */
                 char    *stype;
                 BYTE            DSides,              /* Number of sides */
                         STrax,               /* Tracks          */
                         TSectors;             /* Sectors per track */

               } FloppyType;


typedef struct {
                 BYTE    HardNo,               /* The number of Hards */
                         RWHeads;              /* Number of R/W Heads */
                 int     Celenders,             /* Celenders       */
                         TSectors;            /* Sectors per track */

               } HardType;


typedef struct {               /* Describes a sector position */
                 BYTE Head;                /* Read/write head */
                 WORD SecCyl;      /* Sector and cylinder number */
               } SECPOS;

typedef struct {                   /* Partition table entry */
                 BYTE         Status;      /* Partition status */
                 SECPOS       StartSec;        /* First sector */
                 BYTE         PartTyp;       /* Partition type */
                 SECPOS       EndSec;          /* Last sector */
                 unsigned long SecOfs;  /* Offset of boot sector */
                 unsigned long SecNum;      /* Number of sectors */
               } PARTENTRY;

typedef struct {                /* Describes partition sector */
                 BYTE     BootCode[ 0x1BE ];
                 PARTENTRY PartTable[ 4 ];
                 WORD     IdCode;                 /* 0xAA55 */
               } PARTSEC;

typedef PARTSEC far *PARSPTR;
                        /* Ptr. to partition sector in memory */
```

```
typedef BYTE SectorSize[ 512 ] ;          /* Memory for a Sector */
typedef SectorSize far *SectorSizeP; /*Ptr. to a sector */

/*****************************************************************/
/* ResetDisk : Disk reset on all drives.                         */
/* Input     : None                                              */
/* Output    : None                                              */
/* Info      : Reset executed on all drives, regardless of drive */
/*             number loaded in DD                               */
/*****************************************************************/

void DiskReset( void )
{
 union REGS regs;          /* Processor registers for interrupt call */

 regs.h.ah = 0x00;             /* Function number for interrupt call */
 regs.h.dl = 0;                        /* Drive a: (see Info) */
 int86( 0x13, &regs, &regs );                  /* Interrupt call */
}


/*****************************************************************/
/*  GetSecCyl: Gets the combined sector/cylinder coding of the   */
/*             BIOS sector and cylinder number.                  */
/*  Input  : SecCyl   : Value to be decoded                      */
/*           Sector   : Sector variable reference                */
/*           Cylinder : Cylinder variable reference              */
/*  Output : None                                                */
/*****************************************************************/

void GetSecCyl( WORD SecCyl, int *Sector, int *Cylinder )

{
 *Sector   = SecCyl & 63;                  /* Mask bits 6 and 7 */
 *Cylinder = HI(SecCyl) + ( ( (WORD) LO(SecCyl) & 192 ) << 2 );
}

/*****************************************************************/
/* GetFloppyType: Determines the Floppy drive type.              */
/* Input        : DRIVE = Drive number                           */
/* Output       : Floppy spcefications as a struct of FloppyType */
/*****************************************************************/

FloppyType GetFloppyType (BYTE Drive)
{
 union REGS regs;        /* Processor registers for interrupt call */
```

```
FloppyType     info;
int type=0;
regs.h.ah = 0x08;                /* Function: Determine drive type */
regs.h.dl = Drive;                              /* Drive number */
int86( 0x13, &regs, &regs );                /* Call BIOS interrupt */
info.DSides= regs.h.dh;         /*save the sides number*/
info.STrax= regs.h.ch;         /*save the tracks number*/
info.TSectors= regs.h.cl;      /*save the sectors number*/
type= regs.h.bl;
switch (type)
{
case NOPE:info.stype="Nope";
            break;
case NO_DRIVE:info.stype="Drive does not exist";
               break;
case DD_525:info.stype="Drive: 5.25 DD"; /*5.25 Double Density*/
            break;
case HD_525:info.stype="Drive: 5.25 HD"; /*5.25 High Density*/
            break;
case DD_35:info.stype="Drive: 3.5 DD";   /*3.5 Double Density*/
            break;
case HD_35:info.stype="Drive: 3.5 HD";   /*3.5 High Density*/
 }

return (info);
}

/*****************************************************************/
/* GetHardType: Determines the Hard drive type.                  */
/* Input        : DRIVE = Drive number                           */
/* Output       : Hard spcefications as a struct of HardType     */
/*****************************************************************/
HardType GetHardType (BYTE Drive)
{
 union REGS regs;
 HardType    Hinfo;
 WORD cylcsec;
 regs.h.ah = 0x08;               /* Function: Determine drive type */
 regs.h.dl = Drive;                              /* Drive number */
 int86( 0x13, &regs, &regs );                /* Call BIOS interrupt */
 Hinfo.HardNo= regs.h.dl;                                  /*save hards number*/
 Hinfo.RWHeads=regs.h.dh+1;                                /*save read write heads number*/
 GetSecCyl( regs.x.cx, &Hinfo.TSectors, &Hinfo.Celenders); /*save cylinders number*/
 return (Hinfo);
}
```

```
/*****************************************************************/
/*ReadPartSec     :Reads a partition sector from the hard drive.  */
/*Input  :- DS    : BIOS code for drive (0x80, 0x81, etc.)        */
/*        - Head  : Number of read/write heads                    */
/*        - SctCyl: Sector/cylinder numbers in BIOS format        */
/*        - Buf   : Buffer to which sector is passed              */
/*Output : TRUE if sector can be read without errors,             */
/*         otherwise FALSE                                        */
/*****************************************************************/

BYTE ReadPartSec( BYTE Drive, BYTE Head, WORD SecCyl, PARSPTR Buf )

{
 union REGS   Regs;     /* Processor registers for interrupt call */
 struct SREGS SRegs;

 Regs.x.ax = 0x0201;        /* Funct. no.: READ for first sector */
 Regs.h.dl = Drive;                 /* Pass other parameters */
 Regs.h.dh = Head;                   /* to their respective   */
 Regs.x.cx = SecCyl;                 /* registers             */
 Regs.x.bx = FP_OFF( Buf );
 SRegs.es  = FP_SEG( Buf );

 int86x( 0x13, &Regs, &Regs, &SRegs );
                                    /* Call hard drive interrupt */
 return !Regs.x.cflag;
}

/*****************************************************************/
/*ReadWriteHardSec : Reads or Writes a sector from\to the hard drive. */
/*Input  : - DS    : BIOS code for drive (0x80, 0x81, etc.)       */
/*         - OpNum : Opperation Number 0x02:Read,0x03:Write       */
/*         - Sec   : Number of Sector                             */
/*         - Head  : Number of read/write heads                   */
/*         - SctCyl: Sector/cylinder numbers in BIOS format       */
/*         - Buf   : Buffer to which sector is passed             */
/*Output : TRUE if sector can be read without errors,             */
/*         otherwise FALSE                                        */
/*****************************************************************/

BYTE ReadWriteHardSec( BYTE Drive,BYTE OpNum,BYTE Sec ,BYTE Head,
WORD SecCyl,SectorSizeP Buf )

{
```

```
union REGS   Regs;     /* Processor registers for interrupt call */
struct SREGS SRegs;

Regs.h.ah = OpNum;       /* Funct. no.: READ\Write sector    */
Regs.h.al = Sec;                 /* Pass other parameters */
Regs.h.dl = Drive;               /* to their respective   */
Regs.h.dh = Head;
Regs.x.cx = SecCyl;
Regs.x.bx = FP_OFF( Buf );       /* registers             */
SRegs.es  = FP_SEG( Buf );

int86x( 0x13, &Regs, &Regs, &SRegs );
                                 /* Call hard drive interrupt */
return !Regs.x.cflag;

}

/*****************************************************************/
/*ReadWriteFloppySec : Reads or Writes a sector from\to the Floppy drive. */
/*Input  : - DS      : BIOS code for drive (0x00, 0x01, etc.)            */
/*         - OpNum   : Opperation Number 0x02:Read,0x03:Write            */
/*         - Side    : Number of Side 1st=0x00 ,2nd=0x01                 */
/*         - Sec     : Number of Sector                                  */
/*         - Track   : Number of Track                                   */
/*         - Buf     : Buffer to which sector is passed                  */
/*Output : TRUE if sector can be read without errors,                    */
/*         otherwise FALSE                                              */
/*****************************************************************/

BYTE ReadWriteFloppySec( BYTE Drive,BYTE OpNum,BYTE Side ,BYTE
Sec,BYTE Track,SectorSize* Buf)

{
union REGS   Regs;     /* Processor registers for interrupt call */
struct SREGS SRegs;

Regs.h.ah=OpNum;         /* Funct. no.: READ\Write sector        */
Regs.h.al=1;            /*Reading\writing only one sector */
Regs.h.dl=Drive;           /* Pass other parameters          */
Regs.h.dh=Side;            /* to their respective           */
Regs.h.cl=Sec;
Regs.h.ch=Track;
Regs.x.bx = OFS( *Buf );              /* Offset addr. of buffer */
SRegs.es = SEG( *Buf );               /* Segment addr.          */
int86x( 0x13, &Regs, &Regs, &SRegs );     /* Call BIOS interrupt */
return !Regs.x.cflag;
```

```
}


/*****************************************************************/
/*  ShowPartition: Displays hard drive partitioning on the screen.*/
/*  Input  : DS : Number of the corresponding hard drive         */
/*               (0, 1, etc.)                                    */
/*  Output : None                                                */
/*****************************************************************/

void ShowPartition( BYTE LW )
{
 #define AP ( ParSec.PartTable[ Entry ] )

 BYTE       Head,                  /* Head of current partition */
            Entry;                      /* Loop counter */
 int        Sector,                    /* Get sector and  */
            Cylinder;                  /* cylinder number */
 PARTSEC    ParSec;                /* Current partition sector */
 union REGS Regs;       /* Processor registers for interrupt call */

 printf("\n");
 LW |= 0x80;                   /* Prepare drive number for BIOS */
 if ( ReadPartSec( LW, 0, 1, &ParSec ) ) /* Read partition sector */
  {                              /* Sector could be read */
   Regs.h.ah = 8;                /* Funct. no.: Read drive ID */
   Regs.h.dl = LW;
   int86( 0x13, &Regs, &Regs );      /* Call hard drive interrupt */
   GetSecCyl( Regs.x.cx, &Sector, &Cylinder );
   printf( "+-----------------------------------------"
           "-----------------------------+\n");
   printf( "      Drive %2d:     %2d Heads %4d"
           " Cylinders %3d Sectors                        \n",
           LW-0x80, Regs.h.dh+1, Cylinder, Sector );
   printf( "      Partition table in Partition Sector             \n");
   printf( "-----------------------------------------"
           "-----------------------------\n");
   printf( "                          Start         End"
           "           Dist fm          \n");
   printf( "Nr Boot Type                Head Cyl. Sec  Head"
           " Cyl. Sec BootSec Total\n");
   printf( "--+----+------------------+--------------+----"
           "----------+-------+---------\n");

   /*-- Get partition table ---------------------------------------*/
   for ( Entry=0; Entry < 4; ++Entry )
```

```
    {
     printf( " %d", Entry );
     if ( AP.Status == 0x80 )              /* Partition active? */
      printf("  Yes");
     else
      printf("  No ");
     printf(" ");
     switch( AP.PartTyp )              /* Compute partition type */
      {
       case 0x00 : printf( "Not allocated      " );
                   break;
       case 0x01 : printf( "DOS, 12-bit FAT    " );
                   break;
       case 0x02 :
       case 0x03 : printf( "XENIX              " );
                   break;
       case 0x04 : printf( "DOS, 16-bit FAT    " );
                   break;
       case 0x05 : printf( "DOS, ext. partition" );
                   break;
       case 0x06 : printf( "DOS 4.0 > 32 MB    " );
                   break;
       case 0xDB : printf( "Concurrent DOS     " );
                   break;
       default   : printf( "Unknown (%3d)      ",
                           ParSec.PartTable[ Entry ].PartTyp );
      }

     /*-- Get physical and logical parameters ----------------------*/
     GetSecCyl( AP.StartSec.SecCyl, &Sector, &Cylinder );
     printf( "%2d %5d  %3d    ", AP.StartSec.Head, Cylinder, Sector );
     GetSecCyl( AP.EndSec.SecCyl, &Sector, &Cylinder );
     printf( "%2d %5d  %3d  ", AP.EndSec.Head, Cylinder, Sector );
     printf( "%6lu %6lu  \n", AP.SecOfs, AP.SecNum );
    }
   printf( "+-----------------------------------------"
           "-----------------------------+\n" );
  }
 else
  printf("Error during boot sector access\n");
}


/*****************************************************************/
/*                    MAIN PROGRAM                               */
/*****************************************************************/
```

```
void main()
{
 char c,again='n';
 FloppyType floppyinfo;
 HardType       hinfo;
 int a1,b1,c1,d1;
 int i=0,length1,from1;

 SectorSize SectorData;

 do
 {
  clrscr();
  printf("\nSelect the drive you want to deal with :");
  printf("\n1) Floppy Drive");
  printf("\n2) Hard Drive      \"doesn't work on WinXP\" \n");
  do
  {
    scanf("%c",&c);
  }while((c!='1')&&(c!='2'));
  if (c=='1')         /*Floppy Drive Operations*/
  {
            printf("Select one task of the following to be performed on the Floppy drive:");
            printf("\n1) Get properties of the floppy disk");
            printf("\n2) Read a Sector from the floppy disk");
            printf("\n3) Write a Sector to the floppy disk\n");
            do
            {
              scanf("%c",&c);
            }while((c!='1')&&(c!='2')&&(c!='3'));
          switch (c)
          {
            case '1':      /*properties of the floppy disk*/
                        floppyinfo=GetFloppyType (0);
                        printf("\n\n The Type of Floppy drive is %s",floppyinfo.stype);
                        printf("\n The number of Sides %d",floppyinfo.DSides);
                        printf("\n The number of Tracks %d",floppyinfo.STrax+1);
                        printf("\n The number of Sectors %d",floppyinfo.TSectors);
            break;
            case '2':      /*Read a Sector from the floppy disk*/
                        printf("\nEnter Side Number\n");
                        scanf("%d",&a1);
                        printf("Enter Sector Number\n");
                        scanf("%d",&b1);
                        printf("Enter Track Number\n");
```

```
                scanf("%d",&c1);
                if (ReadWriteFloppySec( 0x00,ReadOp,a1,b1,c1,&SectorData))
                {
                    i=0;
                    while (i<512)                   /*Printing the Sector*/
                    {                                       /*..*/
                      if (i%25==0)                          /*..*/
                        printf("\n");                       /*..*/
                      printf("%3x",SectorData[i]);          /*..*/
                      i++;                                  /*..*/
                    }

                }
                else
                {
                        printf("Operation Failed");   /*Error in Reading*/
                        DiskReset();
                }
        break;
        case '3': /*Write a Sector to the floppy disk*/
                printf("\nEnter Side Number\n");
                scanf("%d",&a1);
                printf("Enter Sector Number\n");
                scanf("%d",&b1);
                printf("Enter Track Number\n");
                scanf("%d",&c1);
                if (!ReadWriteFloppySec(
0x00,ReadOp,a1,b1,c1,&SectorData))/*Read the Sector Bytes*/
                {
                    printf("Operation Failed\n");  /*Error in Reading*/
                    DiskReset();
                    break;
                }
                printf("Enter How Many Bytes you want to write\n");
                scanf("%d",&length1);
                printf("Enter the number of byte you want to start from\n");
                scanf("%d",&from1);

                if ((length1<0)||(from1<0)||((length1+from1)>=512)) /*Invalid Inputs*/
                {
                    printf("Operation Failed\n");  /*Can't perform the writing
operation*/
                    break;
                }

                i=from1;                    /*modifiy the Sector Bytes*/
```

```
                    while (i<(from1+length1))
                    {
                     printf("Enter %d byte: ",i-from1);
                     scanf("%x",&SectorData[i]);
                     i++;
                    }

                    if (ReadWriteFloppySec( 0x00,WriteOp,a1,b1,c1,&SectorData))
/*Write the Sector after modification*/
                    {
                            printf("Operation Done!");
                    }
                    else
                    {
                            printf("Operation Failed");   /*Error in Writing*/
                            DiskReset();
                    }

          }
  }
  else          /*Hard Drive Operations*/
  {
           printf("Select one task of the following to be performed on the Hard drive:");
           printf("\n1) Get properties of the Hard disk");
           printf("\n2) Read a Sector from the hard disk ");
           printf("\n3) Show partion sector properties   \n");
           do
           {
            scanf("%c",&c);
           }while((c!='1')&&(c!='2')&&(c!='3'));
          switch (c)
          {
           case '1':      /*Get properties of the Hard disk*/
                    hinfo=GetHardType (0x80);
                    printf("\n\n The Number of hard drives connected to the controller
%d",hinfo.HardNo);
                    printf("\n The number of read/write heads %d",hinfo.RWHeads);
                    printf("\n The number of Celenders %4d",hinfo.Celenders);
                    printf("\n The number of Sectors %3d",hinfo.TSectors);
           break;
           case '2':      /*Read a Sector from the hard disk*/
                    printf("\nEnter Sector Number\n");
                    scanf("%d",&a1);
                    printf("Enter Head Number\n");
                    scanf("%d",&b1);
                    printf("Enter Cylinder Number\n");
```

```
                    scanf("%d",&c1);
                    if (ReadWriteHardSec( 0x80,ReadOp,a1,b1,c1,&SectorData))
                    {
                          i=0;
                          while (i<512)                          /*Printing the Sector*/
                          {                                              /*..*/
                            if (i%25==0)                                 /*..*/
                              printf("\n");                              /*..*/
                            printf("%3x",SectorData[i]);                 /*..*/
                            i++;                                         /*..*/
                          }                                             /*..*/

                    }
                    else
                    {
                              printf("Operation Failed");     /*Error in Reading*/
                              DiskReset();
                    }
             break;
             case '3':
                    ShowPartition(0x80);       /* Display partition sector */
           }

  }
printf("\n\n Do you want to do another operation? Y/yes\n"); /*Asking to do again?*/
scanf("%s",&again);
}while((again=='y')||(again=='Y'));     /*if again is not equal to ('Y'or'y') then Exit*/
}
```

9) المراجع:

PCIntern (E-book) Chapter 14

حلب
الاربعاء، 23 آذار، 2005