

أنظمة التشغيل

OPERATING SYSTEMS

الدكتور
زياد القاضي



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

أنظمة التشغيل

أنظمة التشغيل

الدكتور
زياد القاضي



المحتويات

رقم الصفحة

المحتوى

- المقدمة 9

الوحدة الأولى

مقدمة عامة

- البيانات والمعلومات 13
- مكونات نظام الحاسوب 14
- دورة حياة العملية 24
- التطور التاريخي لنظم التشغيل 26

الوحدة الثانية

إدارة الذاكرة الرئيسية

- مقدمة 33
- إدارة الذاكرة المفردة والمتماصة 34
- إدارة الذاكرة بالتجزئة 37
- الثابتة 39
- المتغيرة 44
- المتعددة 53
- إدارة الذاكرة بإعادة ترتيب المواقع 61
- إدارة الذاكرة بالصفحات 67
- إدارات الذاكرة الافتراضية : 67
- الصفحات المطلوبة 68

79	الشرائح .
79	الصفحات المطلوبة والشرائح .
83	- طرق أخرى في إدارة الذاكرة :
83	المبادلة .
88	التغطية .
90	الذاكرة المؤقتة .

الوحدة الثالثة

برمجيات إدارة وحدة المعالجة المركزية

95	- مكونات برمجيات إدارة المعالج ووظائفها .
96	- جدولة الأعمال .
97	- طرق جدولة الأعمال .
104	- جدولة العمليات .
106	- أنظمة الحواسيب متعددة المعالجات .
107	- المشاكل المتعلقة بالمعالجات والعمليات :
107	مشكلة السباق .
110	الوضع الهامد .
114	- ميكانيكية التزامن .

الوحدة الرابعة

إدارة الإدخال والإخراج وإدارة وحدات الإدخال والإخراج

121	- إدارة عمليات الإدخال والإخراج :
123	مفهوم الاعتراض .
130	قنوات الإدخال والإخراج .
134	- إدارة وحدات الإدخال والإخراج :
135	طرق استخدام وحدات الإدخال والإخراج .
136	خصائص وحدات الإدخال والإخراج .

147	وحدات التحكم والقنوات .
150	مكونات برمجيات إدارة وحدات الإدخال والإخراج .
154	الوحدات الافتراضية .
160	النظام الافتراضي ونظام التمرير .

الوحدة الخامسة

برمجيات إدارة المعلومات (نظام الملفات)

171	- مقدمة عامة .
172	- وظائف نظام الملف ومكوناته .

دراسة لحالة واقعية

179	- نظام التشغيل DOS .
249	- نظام التشغيل UNIX .
259	- تطبيق عملي .
303	- المراجع .

المقدمة

لقد أصبح استخدام الحاسوب في معالجة البيانات ضرورة لا غنى عنها نظراً لتنوع البيانات وكبر حجمها واختلاف طرق معالجتها .

لكن كيف تتعاون أجزاء الحاسوب المختلفة في معالجة البيانات ؟ ما هي البرمجيات التي تسهم في عملية المعالجة ؟

هذه مجموعة من الأسئلة سوف تتمكن من الإجابة عليها بعد دراسة هذا الكتاب ويتوقع من خلال دراستك لهذا الكتاب أن تحقق الفوائد والأهداف التالية :

- تتعرف مكونات الحاسوب الأساسية وكيفية ارتباطها معاً لتنفيذ عمليات معالجة البيانات المختلفة .

- تطلع على مكونات نظام التشغيل الأساسية وتتعرف على وظائفها المختلفة .

- تتعرف على مفهوم عملية الإدخال والإخراج وتفهم آلية إدارة وحدات الإدخال والإخراج .

- تتعرف على الذاكرة الرئيسية وكيفية إدارتها لتخزين البيانات والنتائج والتعليمات .

- تفهم آلية تنفيذ العمليات من قبل وحدة المعالجة المركزية وكيفية ربط هذه الوحدة مع العملية وكيفية استخدامها وإدارتها .

ولتحقيق هذه الفوائد والأهداف فقد ضم الكتاب مجموعة من

الوحدات تغطي في حجمها متطلبات عملية تنفيذ المعالجة للبيانات المحددة،
بحيث جاء الكتاب مغنياً للمواضيع الأساسية التالية :

- إدارة عمليات الإدخال والإخراج وإدارة وتنظيم وحدات الإدخال والإخراج .
- إدارة الذاكرة الرئيسية وتنظيمها .
- إدارة وحدة المعالجة المركزية وتنظيمها .
- إدارة المعلومات والبيانات وكيفية الرجوع إليها على وحدات التخزين .

ولتحقيق الفائدة فقد عرضنا في نهاية الكتاب بعض نماذج أنظمة التشغيل المنتشرة وخاصة DOS و UNIX أملاً في ربط المادة النظرية بواقع استخدام أنظمة التشغيل .

والله ولي التوفيق

المؤلف

عمان 1997

الوحدة الأولى

مقدمة عامة

- البيانات والمعلومات .
- مكونات نظام الحاسوب .
- دورة حياة العملية .
- التطور التاريخي لنظم التشغيل .

مقدمة عامة

تهدف هذه الوحدة الى اطلاعك على نظم التشغيل وأهميتها وتعريفك بدور نظم التشغيل في عمليات المعالجة. وحتى تلم بعملية المعالجة لا بد أولاً من تعريف البيانات، المعلومات وتحديد المتطلبات الأساسية لعملية المعالجة .

1 - البيانات والمعلومات :

تعرف البيانات بأنها مجموعة من الأفكار والحقائق التي تصف شيئاً محدداً، وتوجد البيانات بأشكال مختلفة فقد تكون بيانات رمزية مؤلفة من حروف وأرقام ورموز خاصة ، أو تكون تمثيلية كالأمواج الصوتية ، درجة الحرارة وغيرها .

وأياً كان شكل البيانات المستخدم في عمليات المعالجة فإنه تتم عملية تحويل هذه البيانات الى صورة قابلة للمعالجة من قبل الحاسوب الرقمي وتسمى البيانات الناتجة عن عملية المعالجة بالمعلومات. وهناك طرق عدة لعمليات المعالجة أهمها :

- تنفيذ العمليات الحسابية والمنطقية على البيانات .

- ترتيب البيانات ليسهل استرجاعها .

- جدولة البيانات وتجميع البيانات المتشابهة وتصنيفها .

- ترميز البيانات وذلك لاختصار حجمها .

- تخزين البيانات لاسترجاعها لاحقاً .

- تحويل البيانات من شكل لآخر .

- تعديل البيانات وحذفها وإضافة بيانات جديدة .

وحتى تتم عملية معالجة البيانات باستخدام الحاسوب لا بد من توفر ما يلي:

1 - وحدة معالجة مركزية .

2 - ذاكرة القراءة والكتابة (Random access memory RAM) .

3 - ذاكرة القراءة فقط .

4 - برمجيات نظام التشغيل .

5 - برمجيات المعالجة (برنامج المستخدم) .

6 - وحدات الادخال والاخراج .

وترتبط هذه المعدات والبرمجيات معاً لتحقيق عملية المعالجة .

فوحدات الادخال والاخراج تستخدم لادخال البيانات المراد معالجتها وتخزين نتائج المعالجة أما ذاكرة القراءة والكتابة فيخزن فيها برنامج المستخدم وبعض البيانات والنتائج أما وحدة المعالجة فتعمل على استقبال التعليمات وتفسيرها ومعالجتها حسابياً ومنطقياً وذلك بتوجيه من برمجيات نظم التشغيل والتي قد تكون متواجدة في ذاكرة القراءة، هذا وسوف نتطرق الى دور نظام التشغيل في عملية إدارة وتنظيم وحدات الحاسوب المختلفة والتي تشترك في عمليات المعالجة في الوحدات اللاحقة ان شاء الله .

1 - 2 مكونات نظام الحاسوب :

يتألف نظام الحاسوب (Computer system) من مجموعة من الوحدات الفيزيائية (المعدات - Hardware) ومجموعة من البرمجيات (Software) ترتبط معاً لتحقيق عمليات المعالجة المختلفة للبيانات .

وفي هذا البند سوف نطلعك على مكونات الحاسوب المختلفة .

1 - 2 - 1 المعدات :

يقصد بالمعدات مجموعة الوحدات الالكترونية المكونة للحاسوب ومن هذه الوحدات :

1 - وحدة المعالجة المركزية :

تتولى هذه الوحدة عملية تنفيذ التعليمات المختلفة والاشراف على تسلسلها وتتكون هذه الوحدة كما هو مبين في الشكل (1 - 1) من مجموعة من الوحدات والعناصر الوظيفية أهمها :

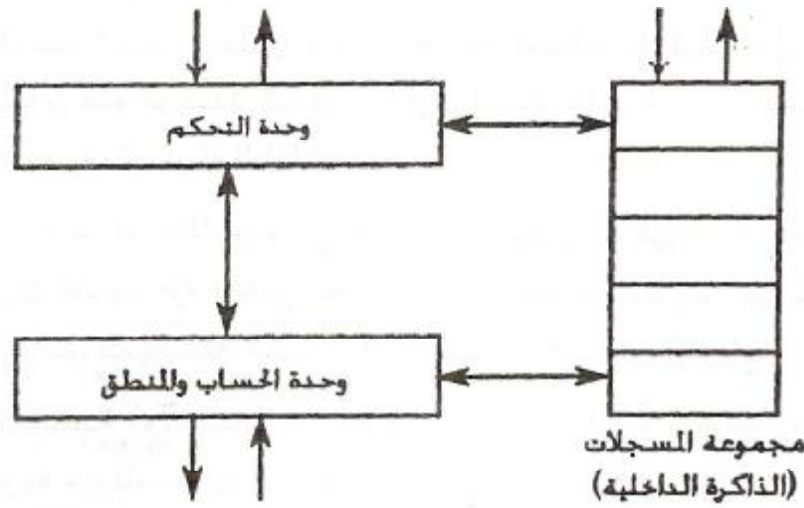
1 - وحدة الحساب والمنطق والتي تتكون من مجموعة من البوابات المنطقية والدارات التجميعية (جامع كامل ، مقارن ، محولات ...) ، حيث تعمل هذه الوحدة على تنفيذ كافة العمليات المنطقية والحسابية على البيانات .

2 - ذاكرة داخلية (Internal memory) تستخدم للقراءة والكتابة وتتكون من مجموعة من المسجلات (Registers) تستخدم لحفظ البيانات (Instructions, addresses, data) حفظاً مؤقتاً ومن أهم هذه المسجلات .

- مسجلات عامة الأغراض (general purpose registers) تستخدم في العمليات الحسابية وحفظ البيانات مثل المرمك .

- مسجلات خاصة (Special registers) تستخدم لأغراض خاصة ومحددة مثل مسجل العنوان (address register) والذي يستخدم لحفظ عنوان التعليمة أو عنوان البيانات ، مسجل التعليمة (Instruction register) والذي يستخدم لحفظ التعليمة المراد تنفيذها ، مسجل (عداد البرنامج) (Program counter) والذي يستخدم في حفظ عنوان التعليمة المراد تنفيذها لاحقاً، مسجل مؤشر الحزمة (Stack pointer) والذي يستخدم للإشارة إلى موقع في ذاكرة القراءة والكتابة يحدد بداية الذاكرة المستخدمة كحزمة لأغراض الحفظ المؤقت ومسجلات التآشير (Index register) والتي تستخدم في عمليات عنونة البيانات والتعليمات .

يتألف المسجل عادة من مجموعة من الخلايا الثنائية (bit) تسمى بالمراجيح (Flip - flop) وعادة ما يعتمد عدد المسجلات وأطوالها على نوع وحدة المعالجة المركزية إذ أن كل وحدة معالجة تمتلك التركيب الخاص بها .



الشكل (1 - 1) : وحدة المعالجة المركزية

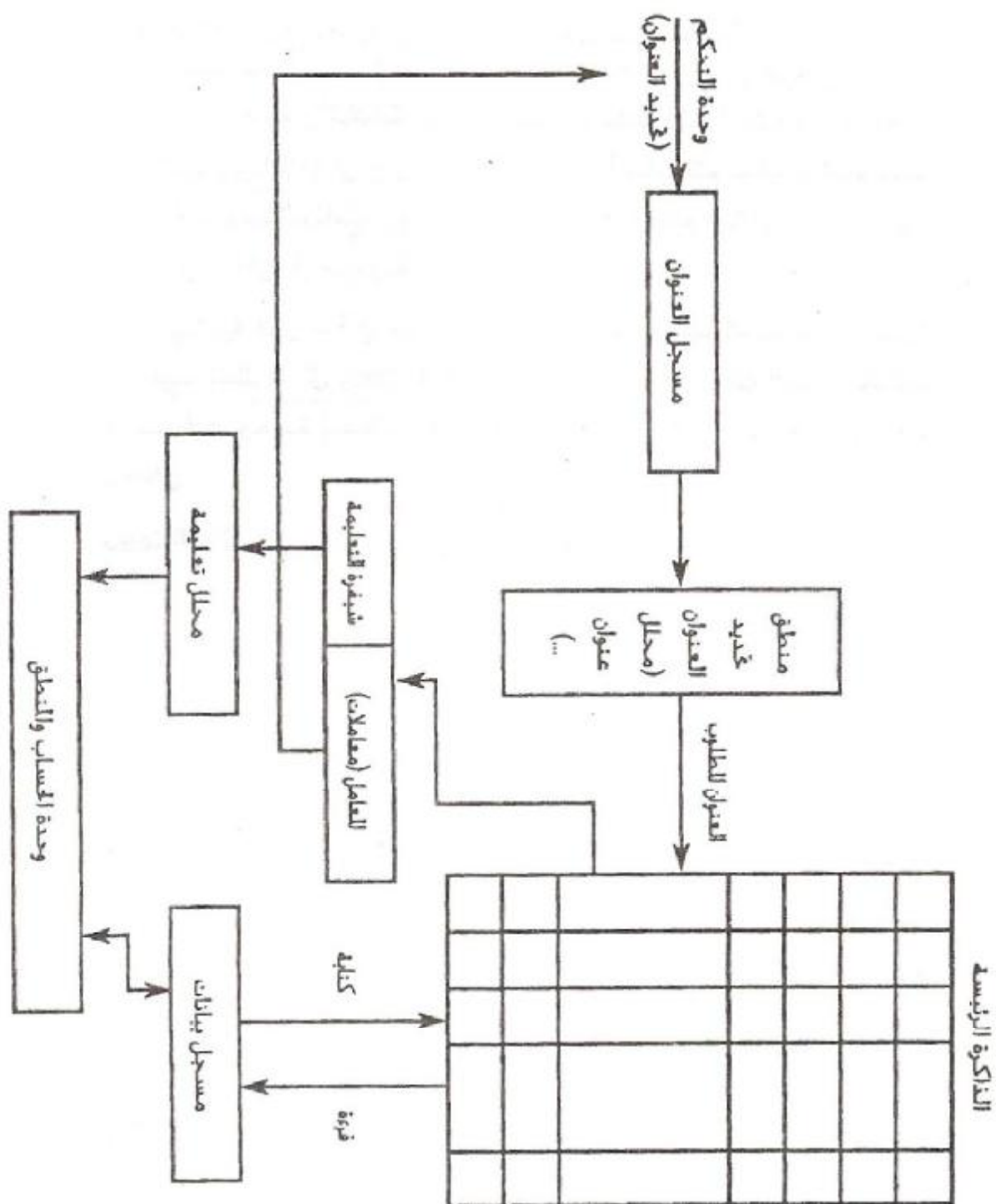
3 - وحدة التحكم (Control unit) ، حيث تتولى هذه الوحدة متابعة عملية التنفيذ والاشراف عليها وتعمل هذه الوحدة على توليد الاشارات (النبضات) الزمنية اللازمة لتحقيق عملية تسلسل تنفيذ العمليات وتؤدي هذه الوحدة بشكل عام الوظائف الاساسية التالية :

- تحديد عنوان التعليمات المراد تنفيذها والاشراف على جلب هذه التعليمات من الذاكرة الاساسية .
- تحليل التعليمات (شيفرة التعليمات) لتحديد نوعها .
- الاشراف على جلب البيانات اللازمة لتنفيذ التعليمات .
- الايعاز لوحدة الحساب والمنطق بتنفيذ التعليمات المحددة على البيانات التي تم احضارها من الذاكرة والاشراف على تخزين النتائج في مواقعها .
- زيادة عداد البرنامج بمقدار مساوٍ لطول التعليمات المنفذة للانتقال الى عنوان التعليمات التالية .

ب - وحدة الذاكرة الرئيسية، والتي ترتبط مباشرة بوحدة المعالجة المركزية وتتألف هذه الذاكرة من مجموعة من المواقع (Locations) ويضم الموقع الواحد مجموعة من الخلايا الثنائية (bits) وتستخدم عادة هذه الذاكرة لحفظ العمل (البرنامج وبياناته) أو جزء منه حفظاً مؤقتاً حتى تتم عملية تنفيذه وتتم عملية تخزين البرنامج وبياناته في الذاكرة باستخدام النظام الثنائي بحيث تخزن في الذاكرة في مجموعة من المواقع المحددة .

ولتنفيذ أية تعليمة في البرنامج يكفي تحديد عنوانها ثم تمرير شيفرة التعليمة المطلوبة الى وحدة المعالجة المركزية وتمرر البيانات اللازمة للتنفيذ لمسجلات وحدة المعالجة لتتم عملية معالجتها باستخدام وحدة الحساب والمنطق .

وبين الشكل (1 - 2) كيفية ارتباط الذاكرة بوحدة المعالجة المركزية .



الشكل (1 - 2) : ارتباط الذاكرة بوحدة المعالجة

- هذا وسوف نتطرق الى دورة تنفيذ التعليمات لاحقاً في هذه الوحدة ان شاء الله.
- ج - وحدات الادخال والاخراج الاساسية والتي تستخدم لادخال البيانات أو استقبال النتائج مثل لوحة المفاتيح ووحدات التخزين المغناطيسية .
- د - وحدات الادخال والاخراج غير الاساسية مثل الطابعة والشريط المغناطيسي والراسم وغيرها وهي وحدات ليس من الضرورة أن تكون متوفرة ويعتمد توفرها على حاجة المستخدم بها .

1 - 2 - 2 البرمجيات :

- تضم البرمجيات مجموعة من البرامج التي يمكن تقسيمها الى :
- 1 - نظم التشغيل (Operating system) :

تعرف نظم التشغيل على أنها مجموعة من البرمجيات الخاصة والتي تعمل على الاشراف على كافة معدات الحاسوب وذلك بضبط عملها والتحكم بإدارتها لاستخدامها في عمليات التنفيذ المختلفة وتحتل هذه البرمجيات حيزاً في الذاكرة الاساسية للحواسيب الصغيرة والكبيرة وقد تخزن على اسطوانات ممغنطة كما هو الحال في الحواسيب الشخصية .

وتلعب نظم التشغيل دوراً كبيراً ومميزاً في تحسين أداء الحاسوب وزيادة فعاليته ذلك ان نظم التشغيل تعمل على إدارة وحدات الحاسوب المختلفة وإدارة البيانات وذلك لاستخدام هذه الوحدات بشكل أفضل .

ولما كانت عملية التنفيذ تتطلب وحدات لادخال البيانات واستقبال النتائج، وذاكرة لتخزين تعليمات البرنامج قيد التنفيذ ووحدة معالجة مركزية لتنفيذ كافة العمليات الحسابية والمنطقية على البيانات فلا بد من توفر البرمجيات الخاصة واللازمة للاشراف على هذه الوحدات وإدارتها وربطها في عملية التنفيذ عند الحاجة اليها .

وبشكل عام فان إدارة وحدات الحاسوب المختلفة وتنظيمها والتحكم بعملية ربطها بالبرنامج المنفذ تؤثر بشكل كبير على فعالية الحاسوب ورفع قدرته

التنفيذية ، لذا فإن نظم التشغيل تلعب دوراً أساسياً في تحديد كفاءة الحاسوب نظراً لأنها تعمل على تنظيم عملية التنفيذ من خلال توفر مجموعة من البرمجيات الخاصة والتي تتكفل بإدارة وتنظيم عمل وحدات الحاسوب المختلفة .

من ما سبق يتضح لنا ان نظم التشغيل يجب أن تتألف من البرمجيات الأساسية التالية :

1 - برمجيات إدارة الذاكرة الرئيسية (Memory management programs) ، حيث تعمل هذه البرمجيات على إدارة الذاكرة وتحميل البرامج (البرنامج) القابل للتنفيذ وحفظه مؤقتاً حتى تتم عملية تنفيذه ثم اعادته، الى الموقع الذي أتى منه (قرص مغناطيسي مثلاً) . ولما كانت عملية التحميل تتطلب مواقعاً في الذاكرة الرئيسية فلا بد لهذه البرمجيات أن تكون قادرة على حل كثير من الأسئلة والمتعلقة بعملية التحميل مثل :

- هل هناك حيز كافٍ في الذاكرة لتخزين البرنامج وبياناته .
- هل يمكن تحميل البرنامج في مناطق مختلفة في الذاكرة .
- هل تتطلب عملية التنفيذ تحميل البرنامج كاملاً أو جزءاً منه .
- هل يمكن تحميل أكثر من برنامج في الذاكرة .

هذه مجموعة من الأسئلة وغيرها الكثير يجب على برمجيات إدارة الذاكرة حلها وسوف نتطرق اليها لاحقاً في وحدة إدارة الذاكرة الرئيسية .

2 - برمجيات إدارة وحدة المعالجة المركزية (Processor management) تعتبر وحدة المعالجة المركزية متطلباً لاتمام عملية التنفيذ فهي تعمل على تنفيذ العمليات الحسابية والمنطقية على البيانات المحددة وربطها بالبرنامج المنفذ ضروري لاتمام عملية التنفيذ .

يخزن البرنامج (البرامج) القابل للتنفيذ في الذاكرة الرئيسية، ولما كانت وحدة المعالجة لا تستطيع تنفيذ أكثر من عملية واحدة في اللحظة الزمنية، إذ

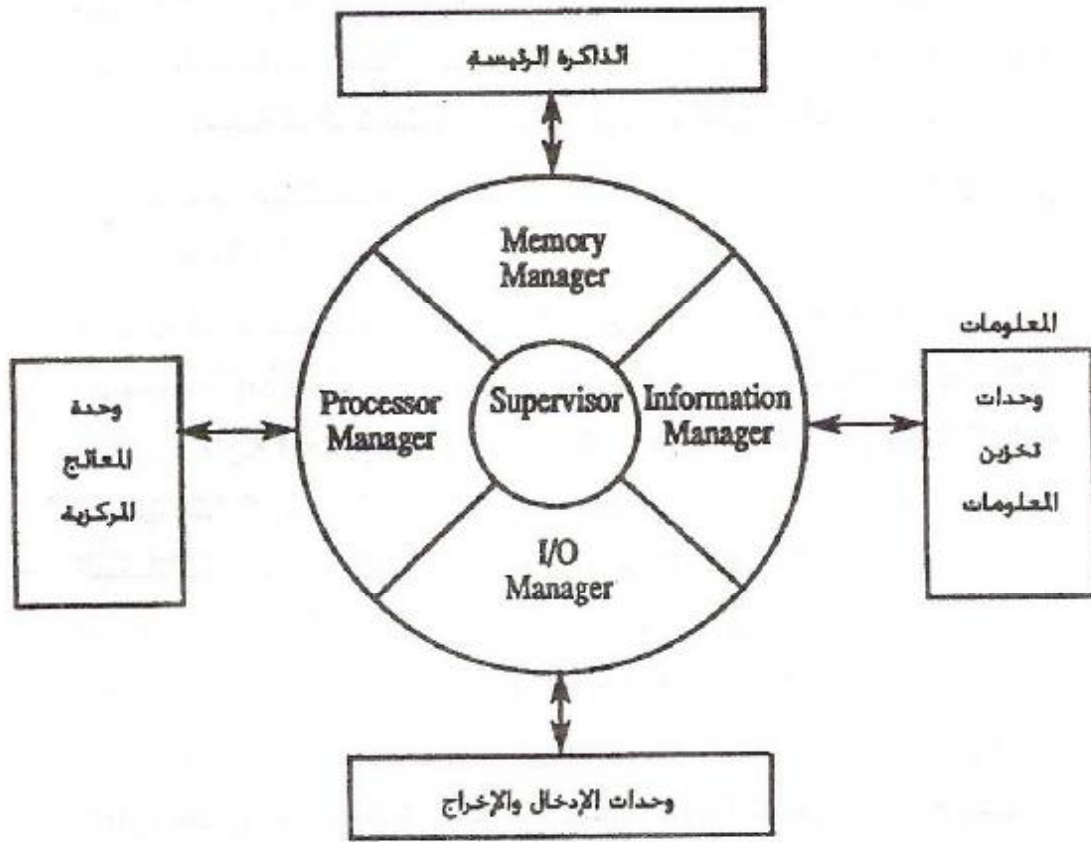
لا بد من وجود آلية معينة لتحديد عملية ربط العملية المراد تنفيذها مع وحدة المعالجة وجدولة البرامج وتحديد أولويات تنفيذها، لذا يجب أن تضم نظم التشغيل برمجيات خاصة تعمل على إدارة وحدة المعالجة المركزية لربطها بالعملية المراد تنفيذها في الوقت المحدد ولفترة زمنية محددة .
هذا وسوف نستعرض برمجيات إدارة وحدة المعالجة المركزية لاحقاً في وحدة خاصة .

3 - برمجيات إدارة عمليات الإدخال والإخراج (I/O operation management) وبرمجيات إدارة وحدات الإدخال والإخراج (I/O device management) تتطلب عملية المعالجة وجود البيانات اللازمة، وينتج عن عملية المعالجة نتائج، لكن كيف يرتبط المعالج بوحدات الإدخال والإخراج، وكيف يستقبل البيانات، وما هي الطريقة المستخدمة في الإدخال والإخراج، هذه مجموعة من الأسئلة يجب أن تحلها برمجيات خاصة تسمى برمجيات إدارة الإدخال والإخراج والتي سوف نستعرضها لاحقاً في وحدة خاصة .

4 - برمجيات إدارة المعلومات (Information management programs) تخزن المعلومات والملفات بشكل دائم في وحدات خاصة تسمى بوحدات التخزين المغناطيسية أو الضوئية، لكن كيف يتم تخزين الملفات على هذه الوحدات، كيف تعالج وكيف تتم عملية الوصول إليها، هذه مجموعة من الأسئلة تناط الإجابة عليها لبرمجيات إدارة المعلومات والتي تتكفل بكافة المهام المتعلقة بحفظ المعلومات وإيجاد الحيز المناسب لحفظها وكذلك كيفية تحديد مواقع هذه المعلومات من أجل الرجوع إليها، هذا وسوف نستعرض هذه البرمجيات في وحدة خاصة لاحقاً إن شاء الله .

5 - برامج المنفعة (Utility program) وهي برامج خاصة وينفذ كل منها مجموعة وظائف محددة ويختلف عددها من نظام تشغيل لآخر ومن أهم هذه البرامج برامج الفرز والدمج والمترجمات (Compilers) والتي تستخدم لتحويل برنامج المستخدم المصدري إلى شيفرة ثنائية قابلة للتنفيذ (برنامج تنفيذي EXE file) شريطة أن يكون البرنامج المصدري خالياً من الأخطاء الإملائية.

والقواعدية ويبين الشكل (1 - 3) مكونات نظم التشغيل .



الشكل (1 - 3) : مكونات نظام التشغيل

يعمل البرنامج المشرف (Supervisor) بتنظيم عمل برمجيات نظم التشغيل المختلفة ويعمل على التنسيق بين هذه البرمجيات ويطلق في بعض الأحيان اسم المشرف على نظام التشغيل وهناك تسميات عديدة له أهمها المراقب (monitor) أو النظام التنفيذي (Executive system) ويغض النظر عن مكونات نظام التشغيل وأجزائها المختلفة إلا أنها تعمل على تنفيذ وأداء مهام وواجبات محددة هي

- مراقبة حالة مصادر (معدات) الحاسوب المستخدمة وذلك لتحديد إمكانية ربطها بالعمل المراد تنفيذه ويقصد بحالة المصدر : هل هو مشغول (غير

متوفر) أو غير مشغول (متوفر) .

- جدولة المصادر واتخاذ سياسة محددة لحجز المصدر المعني لاشراكه في عملية التنفيذ .

- حجز المصدر وربطه بالعمل لانتهاء عملية التنفيذ .

- تحرير المصدر المشترك في عملية التنفيذ بعد انتهاء عملية التنفيذ أو بعد انتهاء الفترة اللازمة لعملية الحجز والربط .

ب - البرامج التطبيقية، ويطلق عليها أحياناً برامج المستخدم (User programs) وتكتب هذه البرامج من قبل المستخدم بأحدى لغات البرمجة العالية المستوى (مثل بيسك، كوبول، باسكال، قاعدة البيانات) أو متدنية المستوى (مثل اسمبلي) وعادة ما تخضع هذا البرامج لمراحل مختلفة تنتهي بتحويل البرنامج المكتوب الى برنامج تنفيذي ومن هذه المراحل :

- مرحلة اعداد البرنامج والتي تتمثل بادخال تعليمات البرنامج المصدري باستخدام محرر النصوص وحفظ هذه التعليمات في ملف .

- ترجمة البرنامج المصدري لشفيرة ثنائية والتأكد من خلوه من الأخطاء الاملائية والقواعدية وتنتهي هذه المرحلة بانشاء البرنامج الهدفي .

- ربط تعليمات البرنامج الهدفي واحتساب العناوين الحقيقية وانشاء برنامج قابل للتنفيذ (Exe - file) .

ومهما كانت لغة البرمجة المستخدمة في كتابة البرنامج المصدري إلا أن هذا البرنامج يتحول الى برنامج تنفيذي يتألف من مجموعة من التعليمات بحيث يتم تخزين هذا البرنامج (أو جزء منه) قبل عملية التنفيذ في الذاكرة الرئيسة بحيث تحتل كل تعليمة فيه موقع أو أكثر (اعتماداً على نوع التعليمة) بحيث تنقسم التعليمة الى حقل أو أكثر ويخصص مجموعة من الحلايا الثنائية للحقل الواحد .

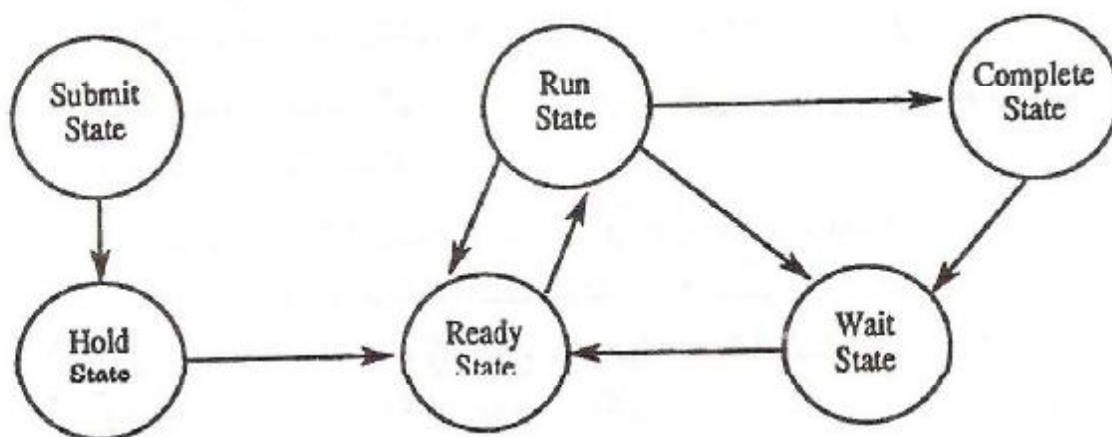
يمثل الحقل الاول في التعليمة شيفرة العملية (Operation code) ويستخدم

هذا الحقل لتحديد نوع التعليمة المراد تنفيذها، أما الحقول الأخرى فتشير الى البيانات اللازمة أو إلا عناوين هذه البيانات .

ج - حزم البرامج الجاهزة (Packages) ويتوفر منها الآن مجموعة كبيرة ، تخصص كل مجموعة منها لمعالجة بيانات محددة، وتمثل هذه البرمجيات برامج جاهزة يعمل المستخدم على تحميلها في الذاكرة وتنفيذها عند الحاجة ومن الأمثلة على هذه البرامج : برامج معالجة النصوص ، برامج معالجة الجداول الالكترونية، برامج التصميم الهندسي وغيرها الكثير .

1 - 3 دورة حياة العملية (Process life cycle) :

تمر العملية المراد تنفيذها بمراحل عدة كما هو مبين في الشكل (1 - 4) :



الشكل (1 - 4) : دورة حياة العملية

1 - مرحلة العرض (Submit state) وتبدأ هذه المرحلة لحظة بدء المستخدم بإدخال تعليمات العمل وذلك لتحضيرها لعملية التنفيذ .

2 - مرحلة التعليق (Hold state) تبدأ هذه العملية عندما يكون العمل قابلاً للتنفيذ وذلك فور تخزين العمل على إحدى وسائط التخزين المتوفرة ويصبح العمل في حالة تعليق أي ليس منفذاً وليس مرفوضاً .

3 - مرحلة الاستعداد (Ready state) وتبدأ هذه المرحلة فور انتقال العمل إلى الذاكرة الرئيسية وفي هذه المرحلة يصبح العمل جاهزاً ومتحفظاً للارتباط بوحدة المعالجة المركزية .

4 - مرحلة التنفيذ (Run state) تبدأ هذه المرحلة فور توفر كافة المصادر اللازمة لتنفيذ العمل وذلك بتنفيذ تعليماته التعليمية تلو التعليمية وتمر عادة هذه المرحلة بدورتين وتسمى هذه المرحلة دورة تنفيذ العملية : (Processing cycle)

أ - مرحلة الاحضار (Fetch state) وفي هذه المرحلة تتم عملية احضار التعليمية من الذاكرة الرئيسية إلى وحدة المعالجة، حيث يتم تحليلها لتحديد نوع العملية المراد تنفيذها وتحديد عناوين البيانات التي يجب أن تمرر من الذاكرة إلى مسجلات وحدة المعالجة المركزية، بعد ذلك تمرر البيانات اللازمة ويزاد عداد البرنامج بقيمة مساوية لطول التعليمية لتحديد عنوان التعليمية المراد تنفيذها لاحقاً .

ب - مرحلة التنفيذ (exectution state) ويتم في هذه المرحلة تنفيذ العملية المحددة على البيانات المحددة .

5 - مرحلة الانتظار (Wait state) وتقع العملية في هذه الحالة عندما تكون في حالة التنفيذ وفي لحظة ما تحتاج العملية إلى إدخال بيانات عبر قنوات الإدخال والإخراج، أو تحتاج العملية إلى إخراج معلومات وتنشأ حالة الانتظار عادة لكون وحدة المعالجة المركزية أسرع من وحدات الإدخال والإخراج أو لاحتمال عدم جاهزية أو توفر وحدة الإدخال والإخراج المطلوبة .

6 - مرحلة الاكتمال (Complete state) وهي المرحلة التي يكتمل بها تنفيذ العمل حيث ينقل العمل وبياناته الى مصادرها التي أت منها .

1 - 5 التطور التاريخي لنظم التشغيل :

كانت الحواسيب في أول عهدها غاية في البساطة، من حيث الوظيفة وتركيب المعدات، وتجهيزات البرمجيات إذا ما تمت مقارنتها بتلك الحواسيب الحالية. فمن الناحية الوظيفية ، لم يكن باستطاعة الحواسيب تخزين الكم الهائل من المعلومات ، فاقترنت بذلك وظيفة الحواسيب على المعالجة الحسابية والمنطقية الفورية ، فكانت أقل بكثير من الآلات الحاسبة المتوفرة الآن في الاسواق بكثرة .

أما من الناحية التركيبية ، فقد بدأت الحواسيب بالصمامات المفرغة ثم «الترانزستورات» ، وبعد ذلك بالدوائر التكاملية "Integrated circuits" ، ومن ثم الشرائح المصغرة "Micro chips" ، فأصبحت بذلك غاية في التعقيد، والتطور .

أما بالنسبة لأحجام الحواسيب ، فقد تطورت كثيراً ، بحيث تم اختصار الأحجام الكبيرة جداً الى الصغيرة ، حتى وصلت الى ما هي عليه الآن "LARGE COMPUTER, MINI COMPUTERS, SMALL COMPUTERS, MICRO COMPUTERS, HAND HELD COMPUTERS" مقارنة بما كانت عليه في أول عهدها .

وبالنسبة للتجهيزات البرمجية، فقد حدثت تطورات هائلة على البرمجيات المختلفة، خاصة برمجيات نظم التشغيل، وإلك المراحل التي مرّت بها نظم التشغيل .

1 - لم يكن هناك نظم تشغيل في أول عهد صناعة الحواسيب، وذلك لعدم وجود ذاكرة أساسية في الحواسيب .

2 - ظهر في عام 1953 ملمح برمجي لنظم التشغيل عرف هذا العمل البرمجي

بنظام معالجة العمل الواحد "Job by job processing" . وقد امتاز هذا النظام ببساطته الشديد ، حيث كان يحتاج الى مساعدة بالعمل اليدوي، سواء كان ذلك لعمليات الادخال أو عمليات الاخراج، وعلى مساوىء هذا النظام، الا أنه يعتبر بحق البنية الأساس في بناء نظم التشغيل .

3 - في عام 1956 ظهر نظام المعالجة الحزمية المبكر "Early batch processing" ، وهو نظام معالجة غير مباشر "Off line" ، يقوم على معالجة أكثر من عمل بالتوالي بشرط أن يفصل هذه الأعمال أوامر تحكم ، عرفت باسم "Job control cards" أو بطاقات التحكم بالعمل. وهذه البطاقات وظيقتها :

أ - تحديد بداية العمل .

ب - تحديد لغة الكتابة للبرنامج المصدري .

ج - تحديد أرقام وأسماء وحدات الادخال أو الاخراج التي تلزم العمل .

د - تحديد اسم المبرمج، وتاريخ كتابة البرنامج، وما الى ذلك من أمور تثيقية للبرنامج .

هـ - تحديد بداية بيانات العمل .

و - تحديد نهاية بيانات العمل .

ز - تحديد نهاية البرنامج، وبداية التنفيذ .

وتحمل كل بطاقة أمراً واحداً فقط من أوامر التحكم بالعمل، وترتبط هذه الأوامر بالبطاقات المثقبة لتلائم شيوعتها مع زمن انتشار نظام المعالجة الحزمية المبكر. حيث تعتبر البطاقات المثقبة الأداة الأولى التي استخدمت لتخزين المعلومات .

4 - في عام 1960 ظهر تطور كبير في وحدات الادخال والاخراج، فكان لزاماً أن يرافق ذلك التطور تطور آخر في نظم التشغيل، وذلك لخدمة وحدات الادخال والاخراج، وقد عرف هذا النظام بنظام خدمات التحكم بوحدة الادخال والاخراج (I/O control services : IOSCS) ، أو ما يسمى بالنظام التنفيذي .

5 - وبعد عام 1960، بدأت تطورات عديدة تحدث على وسائط التخزين المساندة، ورافق ذلك تطورات على الذاكرة الأساسية ، بحيث أصبح بمقدور الذاكرة الأساسية استقبال أكثر من برنامج في نفس الوقت ، وعرف هذا المفهوم بتعدد البرامج (Multiprogramming) فظهر إثر هذا التطور نظام تشغيل قادر على التعامل مع أكثر من برنامج في نفس الوقت ، سمي نظام تشغيل تعدد البرامج (Multiprogramming OS) .

ومع تطور معدات الحاسوب ، صار بالإمكان اشتراك أكثر من مستخدم بنفس الذاكرة ، بحيث يعطي كل مشترك شريحة زمنية من الوقت الكلي للحاسوب ، بحيث ينفذ عمل كل مستخدم بمقدار تلك الشريحة الزمنية المعطاة له ، ومن ثم يعطي مرة أخرى كل مشترك لم ينفذ عمله شريحة زمنية إضافية لينفذ جزء آخر من عمل مستخدم بنفس مقدار الشريحة الزمنية، وهكذا حتى تنتهي جميع الأعمال المرتبطة بالذاكرة الأساسية، وقد عرف هذا النظام بنظام المشاركة بالزمن "Time sharing system" ، فظهر نظام تشغيل عرف باسم "Comprehensive information management and timesharing" .

6 - في عام 1972 حدثت ثورة هائلة في تصنيع الذاكرة المساندة بحيث صار بالإمكان اعتبار الذاكرة المساندة جزء من الذاكرة الأساسية ، وذلك حسب ما يقتض حجم الأعمال الداخلة الى الذاكرة الأساسية . وتعرف هذه الذاكرة باسم الذاكرة الفعالة "Virtual storage" . وبالمقابل حدث تطور آخر على وحدات الإدخال والإخراج بحيث أصبح بمقدورها أن تخدم أكثر من مستخدم في نفس الوقت ، وتسمى هذه الوحدات باسم (I/O virtual devices) حيث تمتاز هذه الوحدات بالسرعة الفائقة والقدرة الهائلة على التخزين ونقاء تقنيها وصناعتها .

الأمر الذي أحدث تقدماً سريعاً في نظم التشغيل ، فظهرت مفاهيم جديدة ، واختفت مفاهيم أخرى ، ودمجت مفاهيم ، فكانت قوالب جديدة ومستحدثة لنظم تشغيل مبتكرة. بحيث أصبح كل حاسوب يملك نظام التشغيل الخاص

به، على درجة كبيرة من التطور .

ومن أنظمة التشغيل ذائعة الصيت والشهرة من الناحية التجارية : AOS/
VS, MSDOS, PRIMOS, CP/M, UNIX, OS, APPLEOS ، وقد انتشرت
نظم التشغيل انتشاراً هائلاً ، بسبب التقدم الهائل الذي أحدثته صناعة
الحواسيب المصغرة "Mirco computer" .

الوحدة الثانية

إدارة الذاكرة الرئيسية MEMORY MANAGMENT

- مقدمة .
- إدارة الذاكرة المفردة والمتماثلة .
- إدارة الذاكرة بالتجزئة .
- الثابتة .
- المتغيرة .
- المتعددة .
- إدارة الذاكرة بإعادة ترتيب المواقع .
- إدارة الذاكرة بالصفحات .
- إدارة الذاكرة الافتراضية :
 الصفحات المطلوبة .
- الشرائح .
- الصفحات المطلوبة والشرائح .
- طرق أخرى في إدارة الذاكرة :
 المبادلة .
- التغطية .
- الذاكرة المؤقتة .

إدارة الذاكرة الرئيسية

MEMORY MANAGMENT

مقدمة :

تعتبر الذاكرة الرئيسية متطلباً أساسياً لتنفيذ العمل، ولتنفيذ العمل يجب أن يحمل في الذاكرة الرئيسية (أو جزء منه) وعند حديثنا عن إدارة الذاكرة فإننا نعني ذاكرة القراءة والكتابة (RAM) ويطلق على هذه الذاكرة اسم الذاكرة الرئيسية (Main memory) أو الذاكرة الأساسية وتتكون هذه الذاكرة من مجموعة من (Primary memory) الخلايا الثنائية موزعة بطريقة تشبه المصفوفة حيث يحدد عدد الخلايا في السطر الأول طول الكلمة ويحمل كل سطر عنوان (رقم) فريد لا يتكرر من خلاله تستطيع وحدة المعالجة تحديد الكلمة المطلوبة في الذاكرة .

يقاس حجم الذاكرة عادة بمجموع الخلايا الثنائية المتوفرة ويعتبر حجم الذاكرة عاملاً أساسياً في قياس كفاءة أداء الحاسوب فكلما زاد حجم الذاكرة زادت قدرة الذاكرة على استيعاب أعمال وبرامج بأحجام كبيرة كما أن طريقة إدارة خلايا الذاكرة وكيفية حجزها تؤثر على أداء الحاسوب .

وفي هذه الوحدة سوف نطلعك على طرق إدارة الذاكرة المختلفة وذلك بهدف:

- 1 - التعرف على طرق إدارة الذاكرة المختلفة .
 - 2 - تعريف القارئ مفهوم كل طريقة من طرق إدارة الذاكرة .
 - 3 - اطلاع القارئ على الاحتياجات الأساسية لكل إدارة وتعريفه بحسنات ومساوئ كل إدارة من الإدارات المختلفة .
- تتنوع طرق إدارة الذاكرة الأساسية بتنوع الحواسيب من حيث سعة الذاكرة وطريقة صنعها وقوة وكفاءة برمجيات إدارة الذاكرة .

وتتوفر طرق عدة لإدارة الذاكرة سوف نطلع عليها في هذه الوحدة أهمها :

- 1 - إدارة الذاكرة المفردة المتماسكة .
- 2 - إدارة الذاكرة بالتجزئة الثابتة والتجزئة المتغيرة والتجزئة المتعددة .
- 3 - إدارة الذاكرة بإعادة توطين الأجزاء وضغطها .
- 4 - إدارة الذاكرة بالصفحات .
- 5 - إدارة الذاكرة الافتراضية :
- إدارة الذاكرة بالصفحات المطلوبة
- إدارة الذاكرة بالقطع (الشرائح)
- إدارة الذاكرة بالقطع والصفحات المطلوبة .

ومهما كانت الطريقة المستخدمة في إدارة الذاكرة إلا أن برمجيات إدارة الذاكرة تتولى تنفيذ مهام محددة هي :

- 1 - متابعة كافة مواقع الذاكرة وكافة الأعمال الداخلة اليها والمستقرة فيها والخارجة منها .
- 2 - رسم سياسة لمتابعة الذاكرة وذلك لتحديد المساحات التي يمكن اشغالها ومتى يسمح للأعمال بالدخول الى الذاكرة والخروج منها .
- 3 - توطين الأعمال القابلة للتنفيذ في الذاكرة .
- 4 - تحرير مواقع الذاكرة والمخصصة للأعمال التي تم تنفيذها وذلك بترحيل هذه الأعمال الى مصادرها الأساسية كالذاكرة الخارجية (الأقراص المغناطيسية مثلاً) .

2 - 1 إدارة الذاكرة المفردة المتماسكة :

Single contiguous memory management

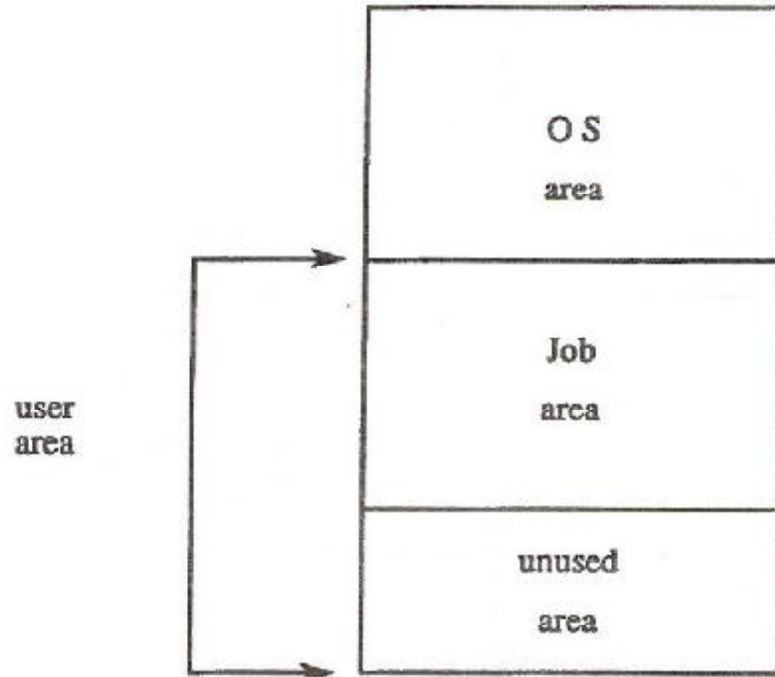
تعتبر هذه الطريقة من أسهل الطرق في إدارة الذاكرة ويرتبط مفهوم هذه الإدارة بالحواسيب المصغرة (الشخصية) التي لا تستطيع في الفترة الزمنية

الواحدة استقبـال أكثر من عمل واسمها يدل على خصائصها الأساسية وهي:

- 1 - تستوعب الذاكرة عملاً واحداً فقط للتنفيذ .
- 2 - يجب أن يخزن العمل في مواقع متتالية ومتسلسلة أي يجب أن يكون العمل متماسكاً ولا يجوز بعثرته في مواقع مختلفة .

تقسم الذاكرة الرئيسة في هذه الإدارة الى قسمين أساسيين :

- 1 - قسم يخصص لتخزين تعليمات نظام التشغيل .
- 2 - قسم يخصص لاستقبال العمل المراد تنفيذه ويسمى بمنطقة المستخدم (USER AREA) وفي بعض الأحيان قد تنشأ منطقة غير مستخدمة وذلك عندما يكون حجم العمل أقل من منطقة المستخدم ويبين الشكل (2 - 1) تقسيمات الذاكرة في إدارة الذاكرة المفردة .

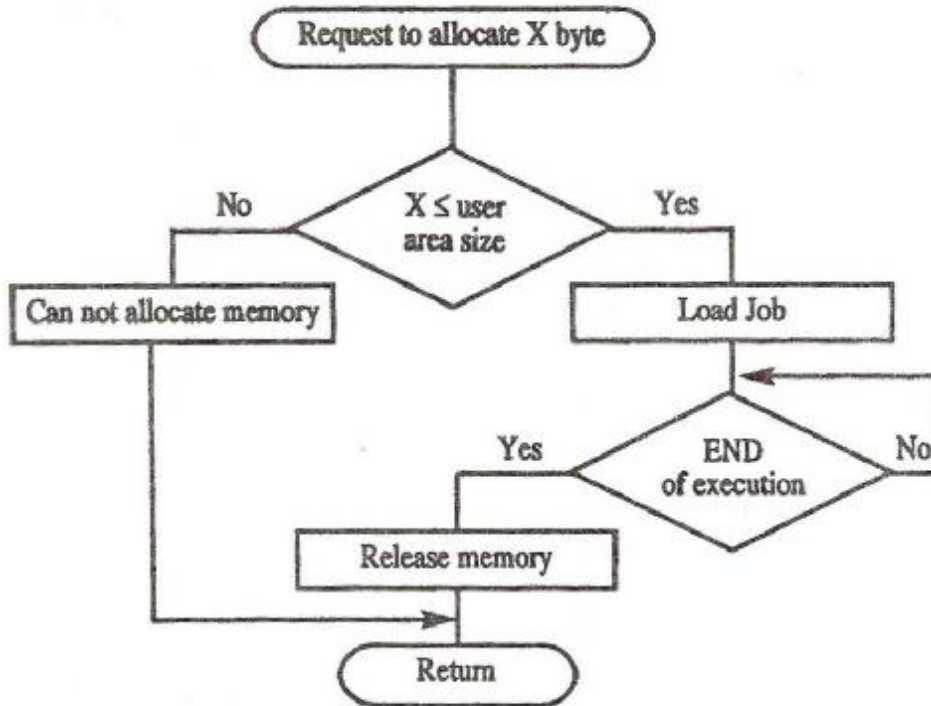


الشكل (2 - 1) : تقسيمات الذاكرة

2 - 1 - 1 ميكانيكية حجز المواقع وتحريرها :

تتلخص إدارة الذاكرة بالخطوات التالية :

- 1 - يتم تحديد حجم العمل المراد تنفيذه .
- 2 - يقارن حجم العمل بحجم منطقة المستخدم فإذا كان حجم العمل أقل أو يساوي حجم المنطقة يحمل العمل في الذاكرة ويبقى متركزاً فيها حتى تنتهي عملية تنفيذه ويبين الشكل (2 - 2) خوارزمية الحجز وإعادة الحجز في إدارة الذاكرة المفردة .



الشكل (2 - 2) : خوارزمية الحجز والتحرير

وحتى تتم حماية منطقة نظام التشغيل لا بد من توفر مسجل خاص يسمى بمسجل الحماية (Protection register) يخزن فيه قيمة آخر عنوان في منطقة نظام التشغيل وعند تنفيذ أية تعليمة في البرنامج المنفذ يجب أن لا تتضمن هذه التعليمة أي عنوان تكون قيمته أقل أو يساوي القيمة المخزنة في المسجل لأن هذا يعني محاولة الوصول الى عنوان في منطقة نظام التشغيل حيث يتم رفض هذه التعليمة .

2 - 1 - 2 حسنات الإدارة ومساوئها :

من حسنات هذه الإدارة سهولة خوارزمية الحجز والترحيل وقلة المعدات المستخدمة أما عيوب هذه الإدارة فتكمن في كما يلي :

1 - عدم استغلال الذاكرة جيداً وذلك للأسباب التالية :

أ - امكانية وجود منطقة غير مستخدمة اطلاقاً .

ب - امكانية احتواء البرنامج على تعليمات وبيانات تستخدم بشكل نادر الأمر الذي يتطلب تواجدها مع البرنامج لعدم التمكن من تجزئة البرنامج لأن هذه الإدارة تتطلب وجود البرنامج كاملاً في الذاكرة .

2 - تقيد حجم الذاكرة المراد تنفيذه بحجم الذاكرة المتوفر (منطقة المستخدم) .

3 - عدم استغلال وحدة المعالجة المركزية جيداً وذلك لاحتمال انتظار وحدة المعالجة المركزية انتهاء عمليات الادخال والاخراج المرتبطة بالبرنامج .

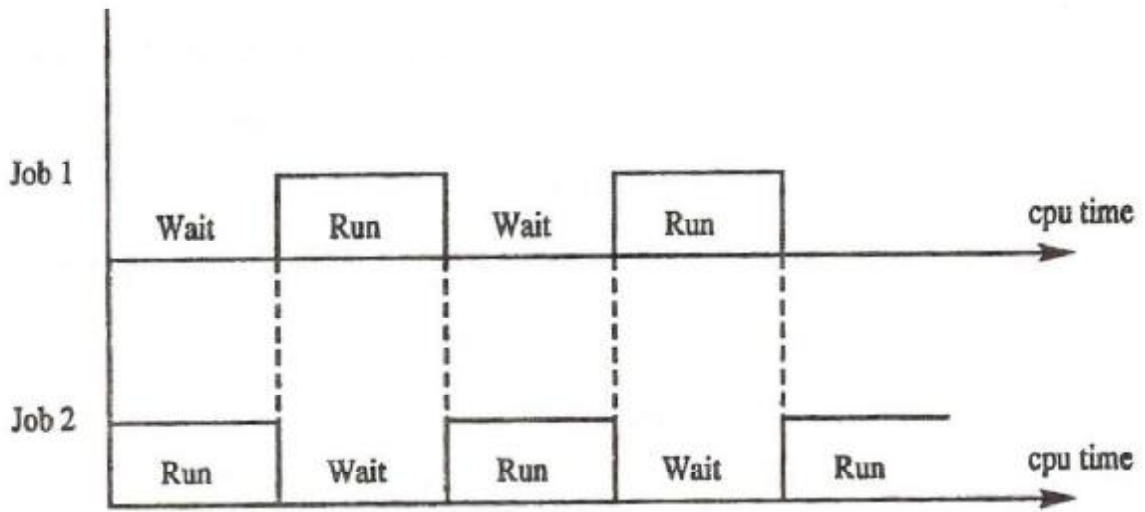
2 - 2 إدارة الذاكرة بالتجزئة Partitioned memory managment :

قبل الحديث عن تجزئة الذاكرة لا بد أولاً من اطلعك على مفهوم تعدد البرامج (Multiprogramming) والذي يقصد به قدرة الذاكرة على استيعاب أكثر من عمل في نفس اللحظة الزمنية .

ويستفاد من تعدد البرامج في اختصار وقت انتظار وحدة المعالجة المركزية وبالتالي اختصار وقت المعالجة وذلك من خلال تنفيذ العمليات الحسابية

والمنطقية وعمليات الادخال والاخراج على التوازي وكما لاحظنا سابقاً فان انتظار وحدة المعالجة يؤدي الى ضياع الوقت وبالتالي عدم استغلال هذه الوحدة جيداً .

تتم عملية تنفيذ الأعمال كما هو مبين في الشكل (2 - 3) على فرض ان الذاكرة الرئيسية تحتوي على عمليتين .



الشكل : (2 - 3) : عملية التنفيذ بتعدد البرامج

نلاحظ من الشكل (2 - 3) أن وحدة المعالجة دائماً عاملة ومشغولة بتنفيذ العمليات الحسابية والمنطقية لعمل ما بينما يقوم العمل الآخر بتنفيذ عمليات أخرى كعمليات الادخال والاخراج .

وتختلف فكرة تعدد البرامج عن فكرة المشاركة الزمنية في أن مبدأ المشاركة الزمنية يتقبل برنامج واحد في الذاكرة الرئيسية في حين تبقى البرامج الأخرى في الذاكرة الثانوية حيث يعطى كل برنامج فترة (شريحة) زمنية للتنفيذ وعند انتهاء هذه الشريحة يدخل البرنامج الى الذاكرة الثانوية (حتى لو لم تتم عملية معالجته) لاحتضار برنامج آخر حيث يعطى نفس الشريحة

الزمنية للتنفيذ وهكذا ، ويبين الجدول (2 - 1) أهم الفروقات بين مفهوم تعدد البرامج ومفهوم المشاركة الزمنية .

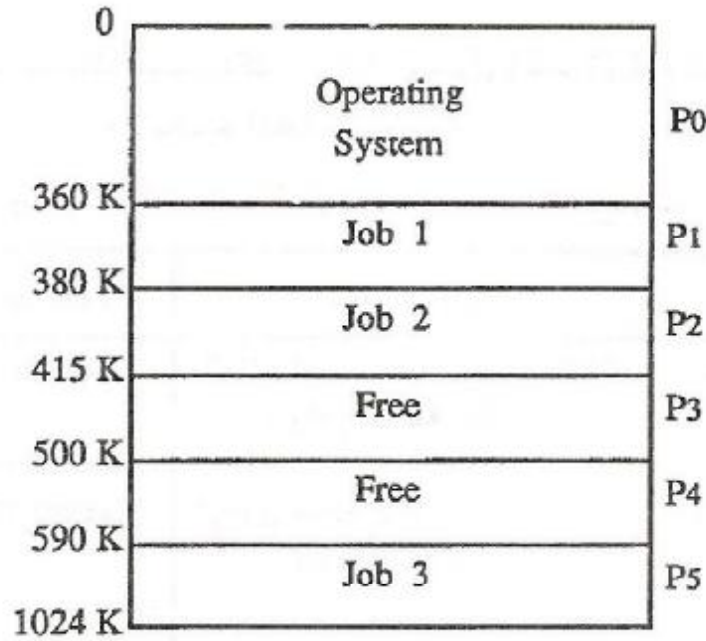
الجدول (2 - 1) : أهم الفروقات بين تعدد البرامج والمشاركة الزمنية .

وجه الاختلاف	تعدد البرامج	المشاركة الزمنية
حجم الذاكرة	تحتاج لحجم كبير لاستيعاب كافة البرامج القابلة للتنفيذ .	حجم الذاكرة أقل
جدولة تنفيذ الأعمال	تجدول عملية التنفيذ حسب عمليات الإدخال والإخراج .	تجدول عملية التنفيذ حسب الشريحة الزمنية
المبادلة مع الذاكرة الثانوية	لا تحتاج الى عملية مبادلة كون كافة البرامج موجودة في الذاكرة الرئيسية .	تحتاج الى عمليات مبادلة
أفضلية الاستخدام	في البرامج التي تكثر بها عمليات الإدخال والإخراج كأنظمة الحجز .	في البرامج التي يكثر بها العمليات الحسابية

2 - 2 - 1 إدارة الذاكرة بالتجزئة الثابتة :

Fixed partitioned memory management

تستخدم هذه الإدارة فكرة تعدد البرامج وذلك من خلال تقسيم الذاكرة الى أجزاء (Partitions) ثابتة الحجم قبل التنفيذ بحيث تختلف الأجزاء عن بعضها البعض في الحجم وذلك لاتاحة الفرصة تحميل برامج بأحجام مختلفة اليها. وعادة ما يكون عدد الأجزاء ثابتاً ويحدد عدد الأجزاء درجة تعدد البرامج فمثلاً الذاكرة المبينة في الشكل (2 - 4) تتيح الفرصة لتحميل خمسة أعمال في نفس اللحظة الزمنية .



الشكل (2 - 4) : الذاكرة بالتجزئة الثابتة (مثال)

ويتم في هذه الإدارة تخصيص مسجلين لكل جزء تسمى مسجلات الحدود (Bound register) وذلك لحماية الأعمال والمحافظة على عدم تداخلها حيث يخزن في المسجل الأول عنوان البداية للجزء وفي المسجل الثاني يخزن عنوان النهاية لنفس الجزء وعند تنفيذ تعليمة لبرنامج معين يتم مقارنة العنوان الذي ترتبط به التعليمة بالقيم المخزنة في المسجل الأول والمسجل الثاني للتأكد من أن العنوان يقع في المدى المخصص للجزء الذي يحتفظ بالبرنامج المحدد .

ولتنفيذ عمليات التوطين في الذاكرة لنفترض وجود الأجزاء المبينة في الشكل (2 - 4) ، ولنفرض أنه في لحظة ما كانت هذه الأجزاء خالية من البرامج .

تتم عملية متابعة الذاكرة وتنفيذ عمليات التوطين والترحيل وذلك باستخدام جدول الحجز (allocation table) والذي يحتوي على مجموعة من المداخل : رقم الجزء ، حجم الجزء ، موقع الجزء ، حالة الجزء مستخدم أو غير مستخدم.

الجدول (2 - 2) : جدول التوظيف :

P. number	Size	location	status
1	20	360	Free
2	35	380	Free
3	85	415	Free
4	90	500	Free
5	434	590	Free

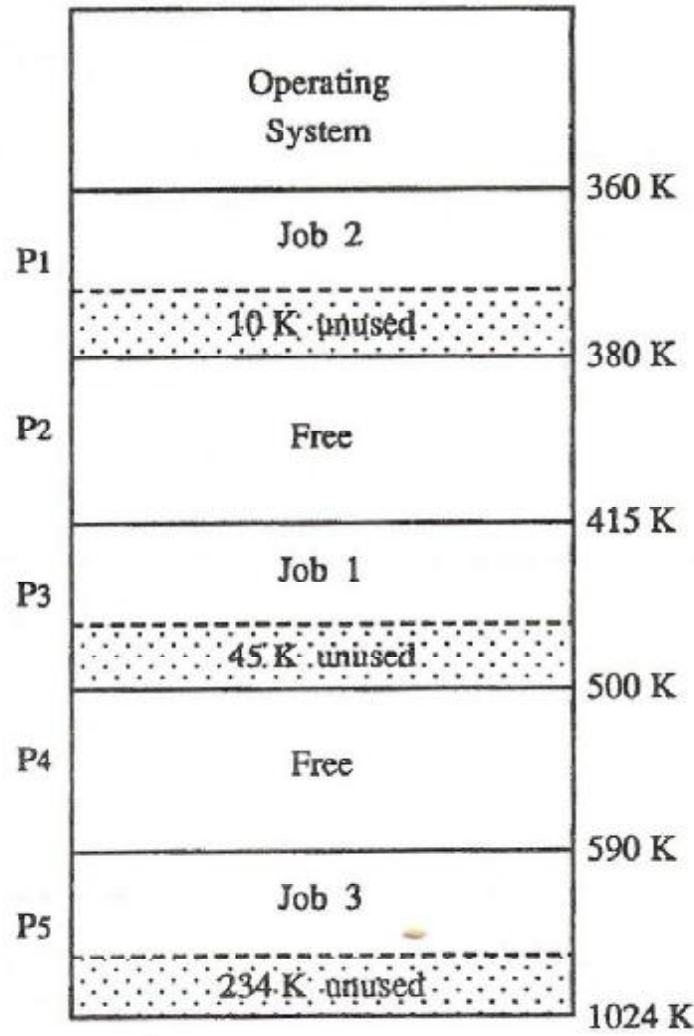
لنفرض الآن أنه طلب تنفيذ الأعمال الآتية

Job 1 size = 40

Job 2 size = 10

Job 3 size = 200

لتحميل العمل الأول يتم البحث في جدول التوظيف عن أول جزء حجمه أكبر أو يساوي الحجم 40 وبهذا يحمل هذا العمل في الجزء الثالث وتعديل حالة هذا الجزء إلى مستخدم وبهذه الطريقة يحمل العمل الثاني في الجزء الأول والعمل الثالث في الجزء الخامس وتظهر الذاكرة بعد عملية التحميل كما هو مبين في الشكل (2 - 5) وتبقى بهذا الفرصة مهيأة لتحميل عمليتين أخريين في الجزء الثاني والرابع .



الشكل (2 - 5) : الذاكرة بعد عملية التوطين

لاحظ إنه بعد عملية الحجز تم توطين الأعمال الثلاثة في الذاكرة، لاحظ أيضاً أن الجزء الأول يحتوي على فراغ غير مستغل حجمه 10K والجزء الثالث على فراغ حجمه 45K والجزء الخامس على فراغ حجمه 234K . وفي هذه الإدارة تبقى هذه الفراغات ضمن الجزء ولا يمكن استغلالها ويسمى هذا الفراغ بالفراغ الداخلي (Internal fragment) فلو افترضنا الآن وجود عمل بحجم 250K ويحتاج هذا العمل للتنفيذ فانه لن يتفقد بالرغم من وجود فراغات ومساحات حرة حجمها (مجموع) أكبر من حجم العمل المطلوب .

أما آلية الحجز فيمكن تلخيصها في الخطوات التالية :

- 1 - طلب بتوطين عمل بحجم X بايت .
- 2 - ابدأ بالجزء الأول في جدول التوطين وقارن حجم العمل بحجم الجزء اذا كانت حالته (Free) فاذا كان حجم الجزء أكبر أو يساوي حجم العمل يتم توطين هذا العمل في الجزء ويحدث حقل الحالة لهذا الجزء الى مستخدم (allocated) وبهذا تنتهي عملية توطين العمل .
- أما اذا كانت حالة الجزء مستخدم أو اذا كان حجم العمل أكبر من حجم الجزء فيتم الانتقال الى الخطوة التالية :
- 3 - يتم زيادة رقم الجزء بمقدار واحد وتنفذ الخطوة السابقة .
- 4 - اذا تم الوصول الى نهاية جدول التوطين ولم يوطن العمل فيؤخر تنفيذه إذا كان هناك جزء مشغول يسمح باستيعاب العمل أو يلغى تنفيذ العمل إذا كان حجمه أكبر من حجم أكبر جزء متوفر .

أما عملية الترحيل فتتم بعد انتهاء تنفيذ العمل وحسب الآلية التالية :

- 1 - حدد رقم الجزء المطلوب .
- 2 - انتقل الى رقم الجزء المطلوب في جدول التوطين وعدل حالة الجزء الى حر (Free) .
- 3 - اذا انتهى الجدول ولم يصادف الرقم المطلوب فهذا يعني وجود خطأ في رقم الجزء المراد تحريره .

مزايا وعيوب إدارة الذاكرة بالتجزئة الثابتة :

من حسنات هذه الإدارة زيادة تعدد البرامج وذلك باتاحة الفرصة تحميل أكثر من عمل قابل للتنفيذ أما مساوئ هذه الإدارة فتتضمن ما يلي :

- 1 - وجود الفراغات الداخلية والتي تؤدي الى ضياع وهدر جزء من الذاكرة وهدر وقت وحدة المعالجة المركزية في استعراض مواقع الجزء كاملة فيما فيها الفراغات .

2 - تتطلب عملية التوطين والترحيل وقتاً لاستعراض جدول التوطين وإجراء عمليات التحديث اللازمة له .

3 - تتطلب عملية التنفيذ إجراء عمليات المقارنة اللازمة للعناوين في الجزء مع القيم المخزنة في مسجلي الحدود الخاصة بالجزء .

4 - بالرغم من تعدد البرامج إلا أن حجم العمل لا زال مقيداً بحجم أكبر جزء متوفر وكذلك ضرورة تماسك العمل في الجزء وضرورة تحميله بالكامل في الجزء المعني .

5 - عملية التوطين تعتمد على أول مطابقة (First fit) فلو كان حجم الجزء الأول 200 وكانت حالته حرة وطلب توطين عمل بحجم 10 فهذا يعني توليد حجم 190 ك بايت غير مستغل .

2 - 2 - إدارة الذاكرة بالتجزئة الحيوية (المتغيرة) :

Dynamic partitioned memory managment

تختلف هذه الإدارة عن الإدارة بالتجزئة الثابتة في أن الذاكرة تقسم الى أجزاء مختلفة الحجم اثناء عملية التنفيذ ويتم تقسيم الذاكرة هنا تبعاً لحجم العمل القادم للتنفيذ فيصبح الحجم المشغول بالعمل جزءاً وبهذا يتم إلغاء الفراغات الداخلية .

تتطلب إدارة الذاكرة بالتجزئة المتغيرة توفر مجموعة من الوحدات تتمثل في مسجلات الحدود حيث يخصص مسجلان لكل جزء كما هو الحال في إدارة الذاكرة بالتجزئة الثابتة إلا أن قيم المسجلات قد تتغير نظراً لتغير موقع وحجم الأجزاء ، إضافة الى المسجلات فان عملية الإدارة تتطلب توفر جدولين :

1 - جدول المساحات المحجوزة (Allocated area table)

ب - جدول المساحات الحرة (Free area table)

ويتألف كل جدول من هذه الجداول من أربعة مداخل هي رقم الجزء أو المساحة الحرة، الحجم، الموقع، والحالة .

وتعتمد درجة تعدد البرامج على عدد البرامج التي يمكن تخزينها في الذاكرة وعادة ما يكون هذا العدد محدداً بحيث لا يجوز تجاوزه .

فلو افترضنا الذاكرة المبينة في الشكل (2 - 6) وافترضنا ان أكبر عدد من البرنامج الذي يمكن تحميله في الذاكرة هو 10 فإن جدول المساحات المشغولة وجدول الأجزاء الحرة يظهر كما هو في الجداول (2 - 3) و (2 - 4) .

0	Operating System	P0
32 K	Job 1	P1
80 K	Free	P2
110 K	Job 2	P3
142 K	Job 3	P4
270 K	Free	P5
370 K	Job 4	P6
420 K	Free	P7
512 K		

الشكل (2 - 6) : تقسيم الذاكرة (مثال)

الجدول (2 = 3) : جدول المساحات المشغولة

Partition No.	Size	location	status
1	48	32	used
2	30	80	unused
3	32	110	used
4	128	142	used
5	100	270	unused
6	50	370	used
7	92	420	unused

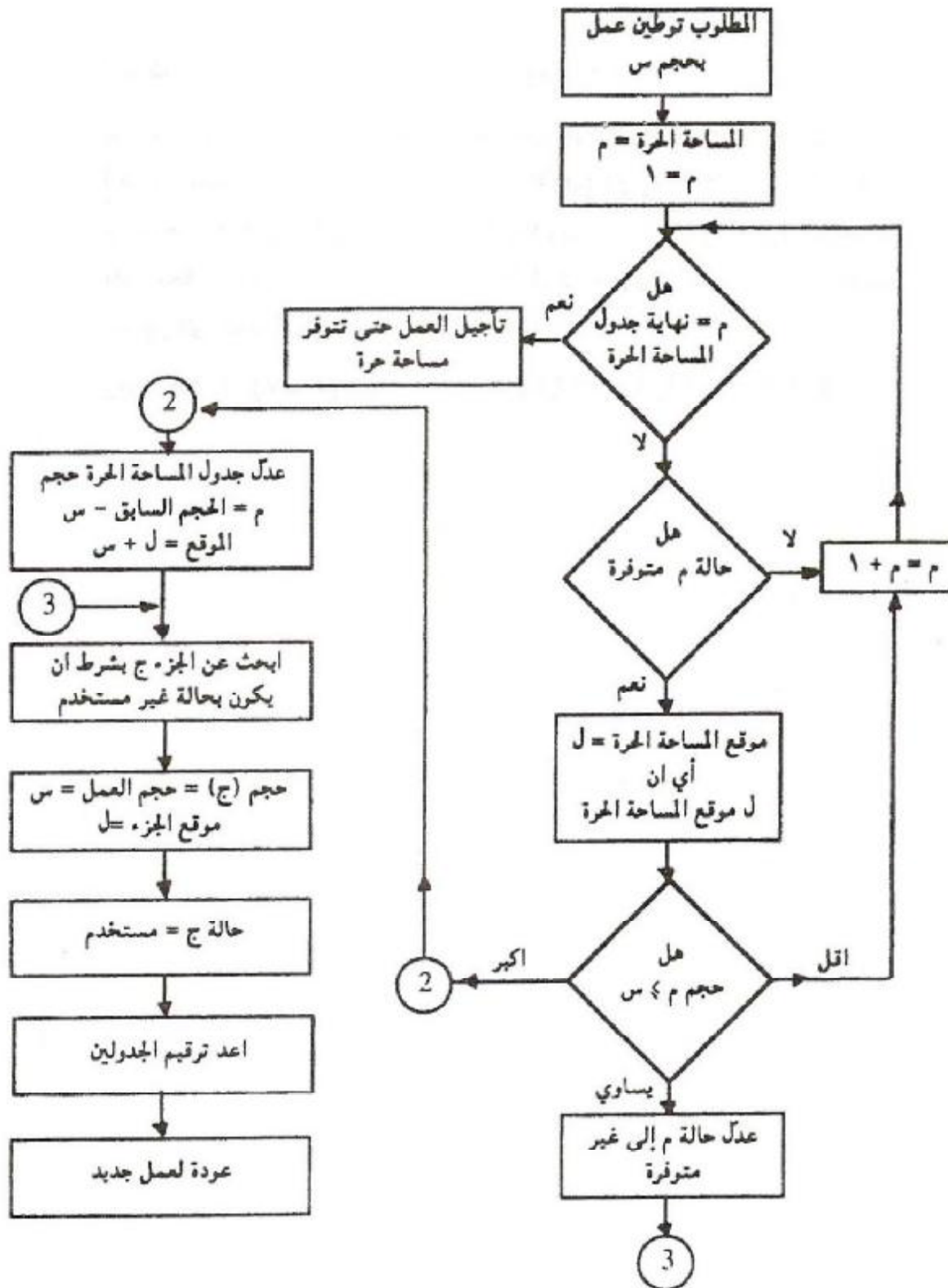
الجدول (2 - 4) : جدول المساحات الحرة

Partition No.	Size	location	status
1	30	80	Free
2	100	270	Free
3	92	420	Free
4	-	-	not available
5	-	-	not available
6	-	-	not available

لنفرض الآن طلب ادخال عمل جديد بحجم 25 K

يتم البحث في جدول المساحات الحرة عن جزء حجمه أكبر أو يساوي حجم العمل المطلوب (في هذه الحالة الجزء الأول) ثم يحمل العمل في هذا الجزء بدءاً من الموقع 80 ويعدل بعد هذا جدول المساحات المشغولة و جدول المساحات الحرة حيث يعدل المدخل الأول في جدول المساحات الحرة بتعديل الحجم الى خمسة والموقع الى 105 .

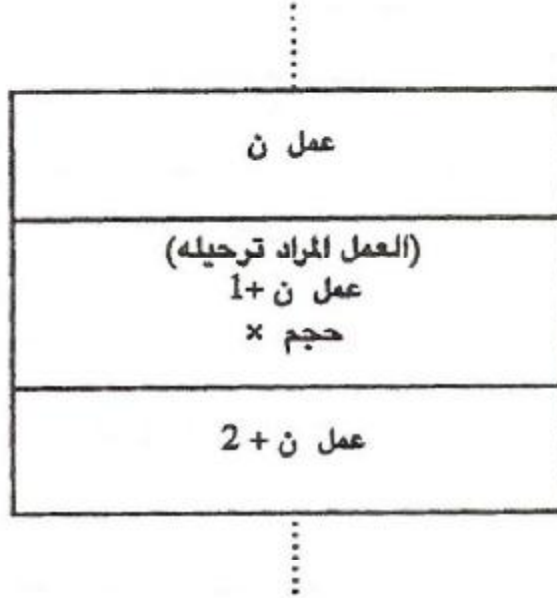
ويبين الشكل (2 - 7) آلية التوطين في إدارة الذاكرة بالتجزئة المتغيرة .



الشكل (2 - 7) : مخطط سير عمليات توطين عمل في الجزء (ج)

أما عملية الترحيل فقد تصادف ثلاثة احتمالات هي :

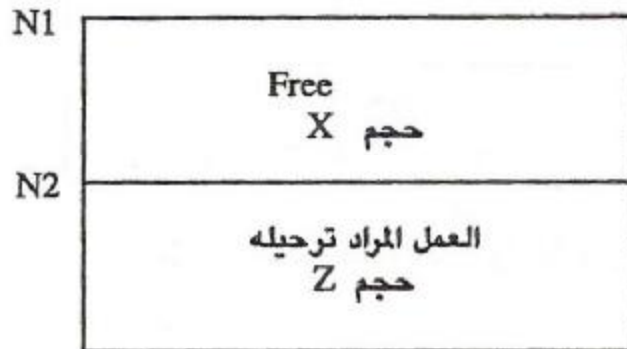
- 1 - العمل المراد ترحيله يقع بين عمليين لا زالوا تحت التنفيذ كما هو مبين في الشكل (2 - 8) .



الشكل (2 - 8) : حالة الترحيل الأولى

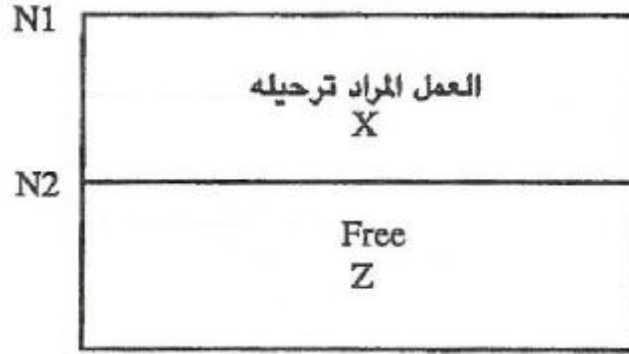
تنتج هذه الحالة مساحة حرة حجمها X وموقعها هو Y

- 2 - العمل المراد ترحيله يقع أعلى مساحة حرة كما هو مبين في الشكل (2 - 9) .



الشكل (2 - 9) : الحالة الثانية

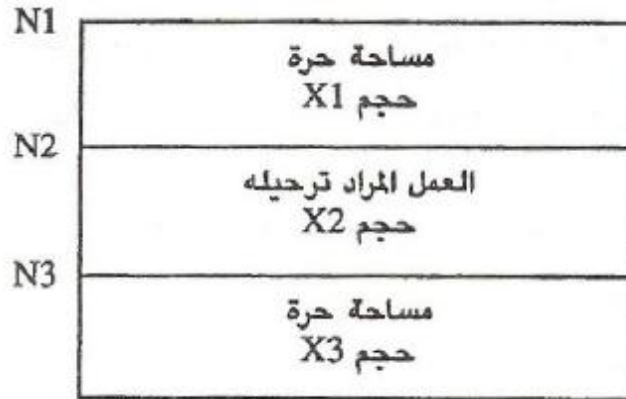
بعد ترحيل العمل المطلوب فإنه ينتج منطقة حرة موقعها هو $N1$ وحجمها هو $X+Z$. أو قد تصادف الحالة التالية المبينة في الشكل (2 - 10) .



الشكل (2 - 10) : الحالة الثانية

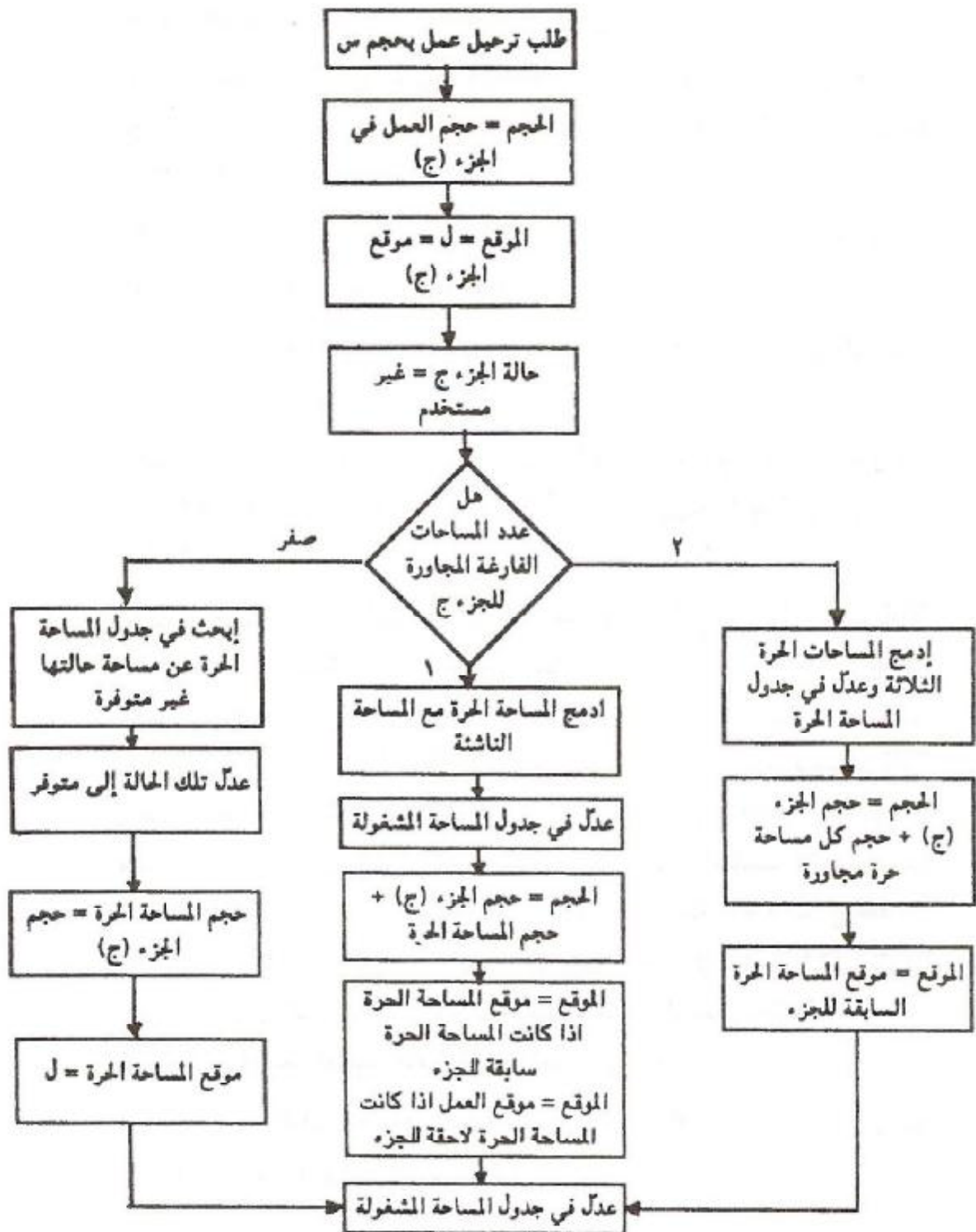
وبعد ترحيل العمل المطلوب ينتج منطقة حرة موقعها هو موقع العمل المراد ترحيله وحجمها مساو لحجم العمل مضافاً إليه حجم المساحة الحرة .

3 - العمل المراد ترحيله يقع بين مساحتين حرتين كما هو مبين في الشكل (2 - 11) .



بعد عملية الترحيل يتم إنتاج منطقة حرة واحدة حجمها هو مجموع $X3$, $X2$, $X1$ وموقعها هو $N1$.

ويبين الشكل (2 - 12) آلية تحرير العمل في إدارة الذاكرة بالتجزئة المتغيرة .



الشكل (2 - 12) : مخطط ترحيل عمل في إدارة الذاكرة المجزأة بالتجزئة المتغيرة

مزايا وعيوب إدارة الذاكرة بالتجزئة المتغيرة

من حسنات هذه الطريقة زيادة أداء وقدرة الحاسوب بزيادة درجة تعدد البرامج والتخلص من مشكلة الكسور (الفراغات) الداخلية لكن هذه الإدارة تمتلك بعض العيوب أهمها :

- 1 - لا زال حجم العمل مقيد بحجم الجزء المتوفر ولتنفيذ العمل يجب أن يكون العمل مخزناً بالكامل في الجزء ومتماسكاً .
- 2 - هدر الوقت لمعالجة الجداول ومتابعة مسجلات الحماية (الحدود) وذلك لتغير قيمها بين اللحظة والأخرى .
- 3 - لا زالت مشكلة الفراغات موجودة ونسمى الفراغات هنا بالفراغات الخارجية وتنشأ هذه الفراغات عن وجود مساحات حرة موزعة لا يستطيع أي منها استيعاب عمل محدد .

لنفترض وجود منطقتين حرتين الأولى بحجم 120K والثانية بحجم 50K وهما متباعدتان وتطلب الأمر تحميل عمل بحجم 45K في هذه الحالة يحمل العمل في المنطقة الحرة الأولى ويأخذ حجماً مساوياً لـ 45K منتجاً منطقة حرة حجمها 75K وبهذا يتبقى في الذاكرة منطقتان حرتان بحجم 75K و حجم 50K . لكن ماذا لو طلب الآن تحميل عمل بحجم 80K . يستحيل في هذه الحالة تحميل العمل لعدم توفر المنطقة الحرة المناسبة وتسمى هذه المناطق بالكسور الخارجية والتي لا يمكن استغلال أي منها في عمليات التحميل كما أشرنا سابقاً فإن طريقة التحميل تعتمد على أول مطابقة وتتوفر طريقة أخرى لتنفيذ عملية التحميل تسمى طريقة أحسن مطابقة (best fit) تتلخص في ترتيب المواقع الحرة ترتيباً تصاعدياً حسب حجمها .

فلو أخذنا الحالة السابقة فإن عملية تحميل العمل الثاني تصبح ممكنة وذلك لأن المناطق الحرة ترتب حسب الموقع :

المنطقة الأولى بحجم 50K ، والمنطقة الثانية بحجم 120K .

يحمل العمل الأول في المنطقة الأولى تاركاً منطقة حرة بحجم 5K وعند البدء

بتحميل العمل الثاني (حجم 80K) فإنه يتوفر منطقة حرة بحجم 120K وبهذا يحمل هذا العمل في هذه المنطقة مخلفاً منطقة حرة بحجم 40K .

2 - 2 - 3 إدارة الذاكرة بالتجزئة المتعددة

Multiple partition memory management

في إدارة الذاكرة بالتجزئة الثابتة والتجزئة المتغيرة تطلب تنفيذ العمل ضرورة تماسكه في الجزء الواحد أما في إدارة الذاكرة بالتجزئة المتعددة فإن شرط التماسك ليس ضرورياً وذلك لامكانية تقسيم العمل المراد تنفيذه الى أجزاء متفاوتة في الحجم شريطة أن يكون حجم الجزء الواحد من مضاعفات الرقم 2 وذلك لتسهيل عملية تحديد مواقع الأجزاء المختلفة في الذاكرة. ويشترط في إدارة الذاكرة بالتجزئة المتعددة توفر عدد من المساحات الحرة مناسب في الحجم مساوٍ لعدد أجزاء العمل . وتتم معاملة الجزء الواحد في العمل تماماً كمعاملة العمل كاملاً في إدارة الذاكرة بالتجزئة المتغيرة أو الثابتة .

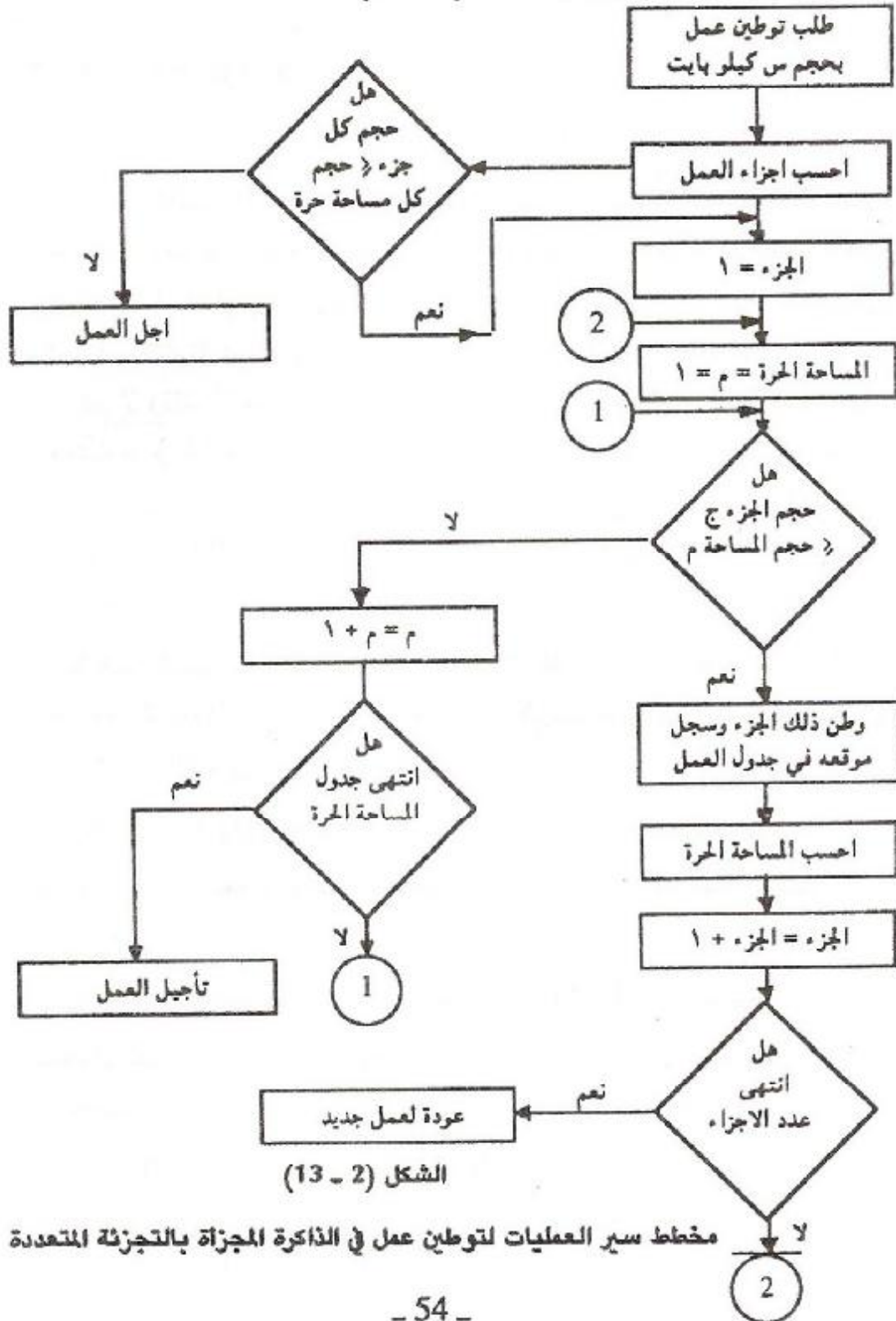
ولما كان العمل مقسماً الى مجموعة أجزاء فإن عملية متابعته سوف تزداد صعوبة ولهذا لا بد من توفر المعدات اللازمة لتحقيق عملية التوطين في الذاكرة أو الترحيل منها ومن هذه المعدات :

- 1 - مسجلات حماية ولكل عمل عدد مساوٍ لعدد أجزاء العمل مضروباً في 2 .
- 2 - مسجل تعريف العمل (Job identification register) ولكل عمل بحيث يشير محتوى هذا المسجل الى جدول العمل (Job table) .
- 3 - جدول عمل ولكل عمل يحتوي على رقم الجزء موقع الجزء وحجمه .
- 4 - جدول المساحات الحرة يحتوي على رقم الجزء الحر ، موقعه في الذاكرة وحجمه .

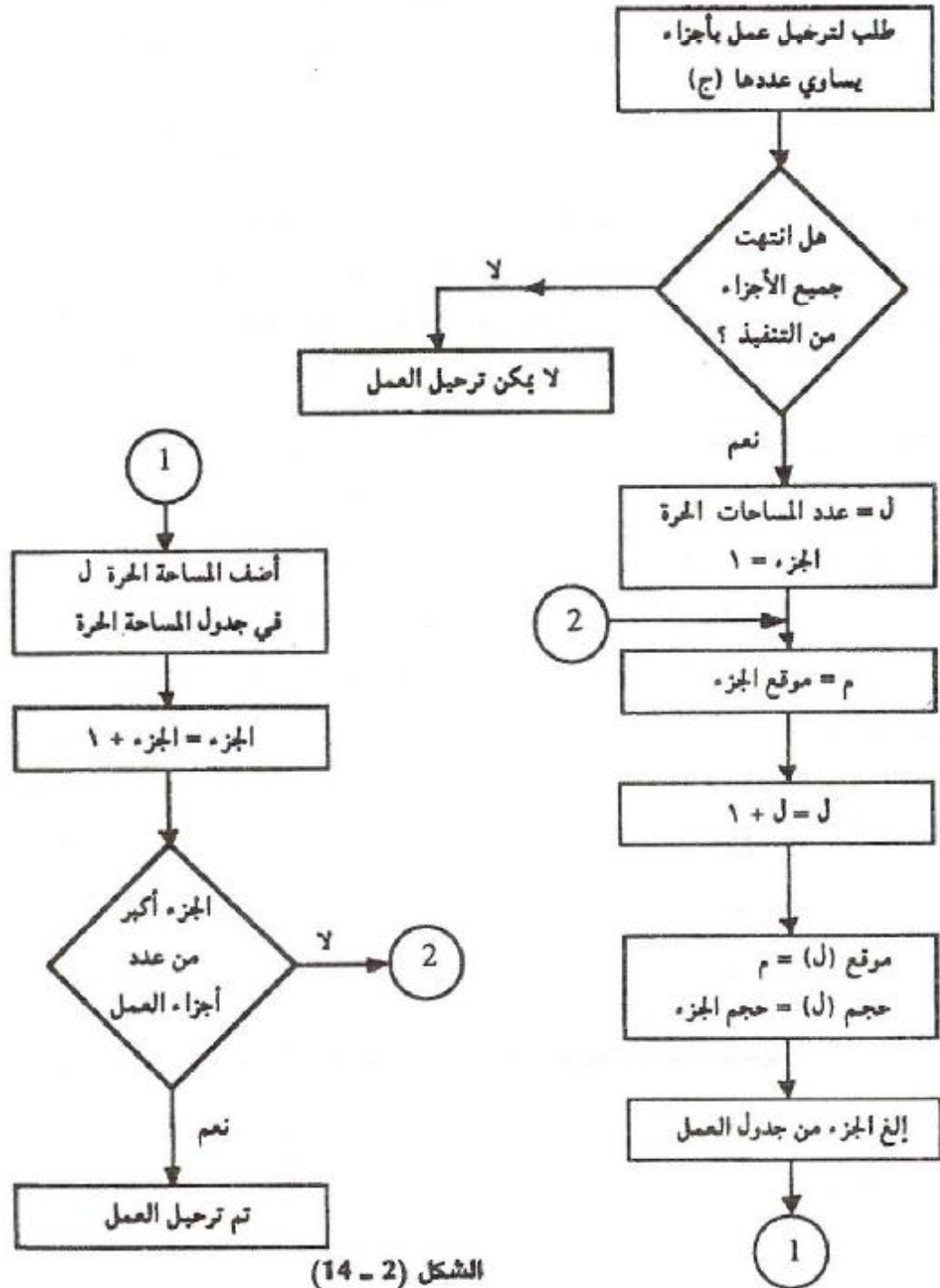
أما آلية التوطين فتتم حسب الخطوات التالية :

- 1 - يقسم العمل الى أجزاء ويتم بناء جدول عمل خاص بالعمل المطلوب توطينه .

2 - يحسب عدد أجزاء العمل ثم يستعرض جدول المساحات الحرة لتحديد امكانية التوطين كما هو مبين في الشكل (2 - 13) :



أما آلية الترحيل فهي مبينة في الشكل (2 - 14) :



مخطط سير عمليات ترحيل عمل من الذاكرة المجزأة بالتجزئة المتعددة

مزايا وعيوب إدارة الذاكرة بالتجزئة المتعددة :

من حسنات إدارة الذاكرة بالتجزئة المتعددة الميزة أنها تزيد من درجة تعدد البرامج وذلك بالغائها شرط تماسك العمل لتنفيذه وحل مشكلة الفراغات حلاً جزئياً، لكن هذه الإدارة تمتاز بالعيوب التالية :

1 - لا زال حجم العمل مقيد بحجم الأجزاء الحرة المتوفرة في الذاكرة ولا زالت الذاكرة تتطلب توطين العمل كاملاً والذي بدوره يؤدي الى عدم استغلال الذاكرة بشكل جيد وهدر وقت وحدة المعالجة المركزية في استعراض مواقع في الذاكرة غير مستخدمة أو تستخدم بشكل قليل ونادر .

ب - كثرة المعدات الفيزيائية المستخدمة والتي تحتاج الى حيز لتخزينها ووقت من وحدة المعالجة لمتابعتها من حيث الانشاء والتعديل والحذف .

ج - لا يسمح بترحيل العمل من الذاكرة الا بعد انتهاء تنفيذه كاملاً والذي بدوره يؤدي الى وجود أجزاء غير عاملة وغير مستغلة في الذاكرة .

د - احتمالية وجود الفراغات الداخلية في جزء العمل كونه حجم الجزء من مضاعفات الرقم 2. فلو كان حجم العمل فرضاً 27K فإن أحد الحلول لتقسيم العمل الى أجزاء هو : 2 ، 8 ، 8 ، 8 ، 2 وبالتالي فإن الجزء الأخير سوف يحتوي على فراغاً داخلياً قدره 1 K ويبقى هذا الفراغ مصاحباً للعمل عند توطينه في الذاكرة .

2 - 3 إدارة الذاكرة باستخدام عملية الضغط

(إعادة توزيع المواقع Relocatable memory managment)

لنفترض الذاكرة المبينة في الشكل (2 - 15) .

	Operating System
512	Job 1
520	Free
552	Job 2
568	Free
576	Job 3
584	Free
600	

الشكل (2 - 15) : حالة الذاكرة (فرضية)

لاحظ ان جدول المساحات الحرة يحتوي على ثلاثة مناطق بحجم 32,16,8 ، أي ان مجموع المساحات الحرة هو 56 .

ماذا لو تطلب الأمر الآن تنفيذ عمل بحجم 40K وباستخدام إدارة الذاكرة بالتجزئة المتغيرة .

لا شك ان عملية التنفيذ سوف ترفض نظراً لعدم توفر جزء يتسع للعمل مع العلم ان مجموع المساحات الحرة تكفي لتحميل العمل المراد تنفيذه .

وللاستفادة من الأجزاء الحرة المتوفرة في الذاكرة ولتحميل أعمال أخرى ولزيادة درجة تعدد البرامج والتقليل من الفراغات ما أمكن تستخدم إدارة الذاكرة بالضغط (Compaction) أو ما يسمى باعادة توزيع المواقع .

وتتلخص فكرة هذه الإدارة في انه قبل تنفيذ العمل يحمل هذا في الذاكرة اذا توفر جزء يستوعب هذا العمل أما إذا لم يتوفر جزء حر فيتم حساب

مجموع المساحات الحرة فإذا كان حجمها أكبر أو يساوي حجم العمل يتم تجميع الفراغات وذلك بضغط الذاكرة لانتاج منطقة حرة واحدة تستطيع استيعاب العمل المراد تنفيذه .

لنفترض الحالة السابقة وأنه تم تنفيذ عملية الضغط وبهذا تصبح الذاكرة كما هو مبين في الشكل (2 - 16) .

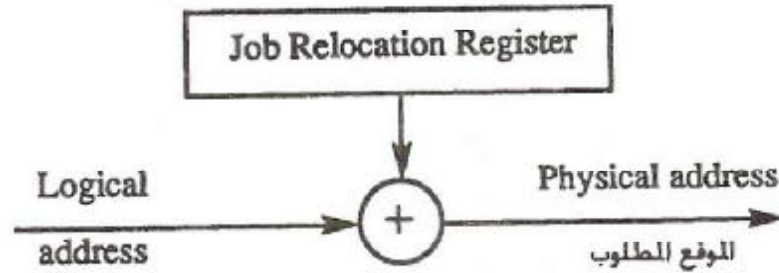
	Operating System
512	Job 1
520	Job 2
536	Job 3
544	Free
600	

الشكل (2 - 16) : الذاكرة بعد عملية الضغط

لاحظ أن العمل الثاني تمت إزاحته بمقدار 32K للأعلى وأن العمل الثالث تمت إزاحته بمقدار 40K وللأعلى .

عند إزاحة العمل في الذاكرة تبقى محتويات البرنامج ثابتة ، أي أن تعليمات البرنامج لا تتغير بحيث تبقى العناوين التي تتعامل معها التعليمات كما كانت قبل عملية الضغط وتسمى هذه العناوين : العناوين المنطقية أو الافتراضية (Virtual or logical addresses) أما العناوين الفيزيائية (Physical adress) للأعمال فقد اختلفت وذلك لحدوث تغيير على مواقع هذه الأعمال بعد تنفيذ عملية الضغط، لذا لا بد من تخصيص مسجل خاص لكل

عمل يسمى مسجل اعادة ترتيب المواقع (relocation register) يخزن فيه مقدار الازاحة للعمل وعادة ما يكون مقدار الازاحة سالباً أو صفراً . وبهذا فان عملية تنفيذ الامر (التعليمة) في العمل تتطلب باستمرار احتساب العنوان الفيزيائي كما هو مبين في الشكل (2 - 17) .

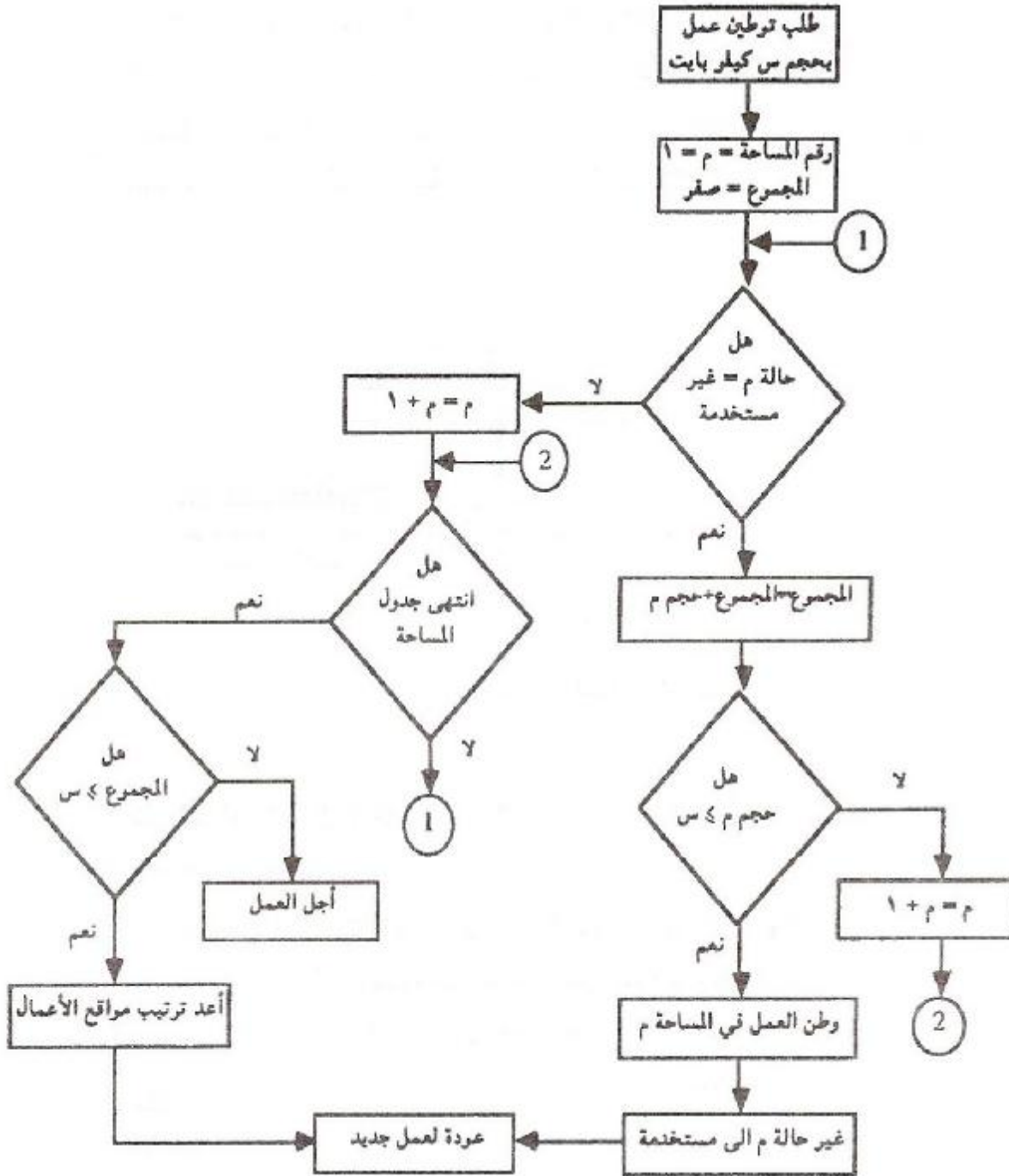


الشكل (2 - 17) : احتساب العنوان الفيزيائي

فلو أخذنا العمل 2 فإن القيمة التي يجب أن تكون في مسجل اعادة ترتيب المواقع هي 32 - .

تتطلب إدارة الذاكرة باعادة ترتيب المواقع مجموعة من المعدات ممثلة في مسجلات الحماية ومسجلات اعادة ترتيب المواقع وجدول بالمساحات الحرة حتى تتم متابعة الذاكرة بصورة صحيحة .

ويبين الشكل (2 - 18) مخطط سير العمليات اللازم لتوطين عمل في إدارة الذاكرة باعادة ترتيب المواقع .



الشكل (2 - 18)

مخطط سير العمليات لتوظيف عمل في إدارة الذاكرة باعادة ترتيب المواقع

مزاياء وعيوب إدارة الذاكرة باعادة ترتيب المواقع

ان ما يميز هذه الإدارة عن الإدارات السابقة هي امكانية تنفيذ عملية الضغط والتي يؤدي تنفيذها الى الحصول على الفائدة التالية :

زيادة درجة تعدد البرامج وذلك بالتقليل من الفراغات ما أمكن ومحاولة تجميعها للاستفادة منها لتحميل عمل آخر .

أما عيوب عملية الضغط فتكمن في :

1 - الحاجة الى توفر مسجلات خاصة وبالتالي استغلال مواقع في الذاكرة لحفظ قيم الازاحة .

ب - هدر وقت وحدة المعالجة المركزية في : -

- تنفيذ عملية الضغط .

- احتساب قيم الازاحة لكل عمل .

- احتساب العنوان الفيزيائي عند تنفيذ كل تعليمة من تعليمات العمل .

2 - 4 إدارة الذاكرة بالصفحات

Paged memory managment

يتلخص مفهوم إدارة الذاكرة بالصفحات فيما يلي :

1 - يتم تقسيم العمل الى أجزاء متساوية في الحجم (وبحجم من مضاعفات الرقم 2 لتسهيل عملية العنونة) يسمى كل جزء بالصفحة وبالمناظر يتم تقسيم الذاكرة الى أجزاء متساوية في الحجم وبحجم الصفحة ويسمى كل جزء الكتلة BLOCK .

2 - تعرف الصفحة على أنها وحدة فيزيائية للبيانات حيث تقرأ الصفحة أو تكتب ويمكن أن تحتوي على فراغات داخلية خاصة اذا كان حجم العمل المراد تحميله في الذاكرة ليس من مضاعفات الرقم 2 .

- 3 - حتى يتم تنفيذ العمل لا بد من تحميله كاملاً في الذاكرة، لذا لا بد من توفر عدد من الكتل مساوي أو أكبر من عدد صفحات العمل بحيث تكون هذه الكتل غير مشغولة .
- 4 - ليس من الضرورة تماسك العمل اذ يمكن تحميل صفحات العمل في كتل متباعدة .
- 5 - ترحيل العمل يتم دفعة واحدة فبعد الانتهاء من تنفيذ العمل ترحل كافة الصفحات التابعة للعمل من الذاكرة الرئيسية .
- 6 - تتيح إدارة الذاكرة بالصفحات تعدد البرامج، اذ يمكن أن يُحمل أكثر من عمل في الذاكرة شريطة توافر الكتل المطلوبة .
- 7 - حجم العمل المراد تحميله يعتمد على حجم الذاكرة المتوفر .

المعدات الفيزيائية :

تتطلب إدارة الذاكرة بالصفحات توفر المعدات التالية :

- 1 - جدول الأعمال (Job table) ويتضمن هذا الجدول رقم العمل، موقع مسجل جدول دالة الصفحات، حجم العمل، حالة العمل (موطن، غير موطن) .
 - 2 - جدول دالة الصفحات (Page Map table : PMT) ولكل عمل يخصص جدول يحتوي على رقم الصفحة ورقم الكتلة في الذاكرة والتي تقع فيها الصفحة .
 - 3 - جدول الكتل (block table) في الذاكرة ويحتوي على رقم الكتلة وحالتها .
 - 4 - مسجل عنوان جدول دالة الصفحات ويخصص مسجل لكل عمل يحتوي على رقم العمل وموقع دالة الصفحات الخاصة بالعمل .
- ويبين الشكل (2 - 19) المعدات اللازمة لإدارة الذاكرة بالصفحات .

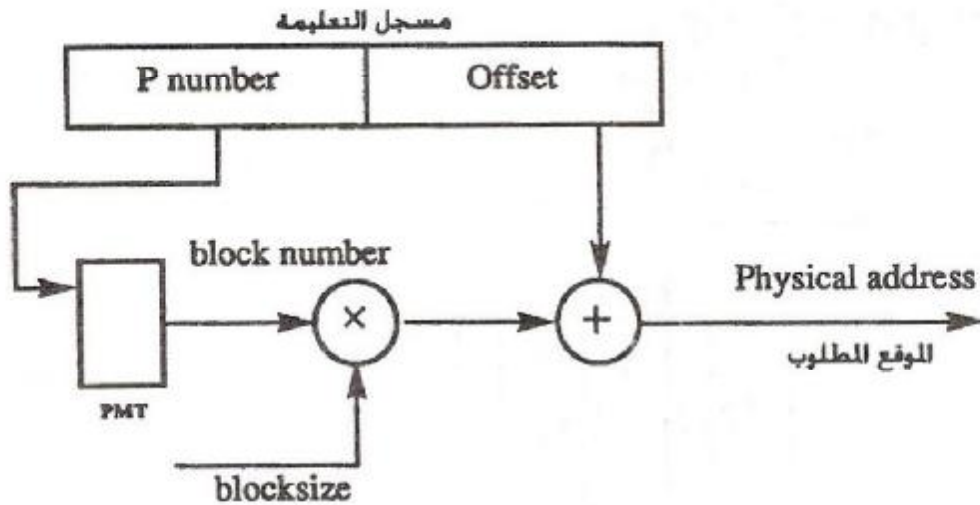
تتطلب عملية تنفيذ أي أمر (تعليلة) في الصفحة تحويل العنوان المنطقي والمخزن في مسجل التعليلة الى عنوان فيزيائي .

يحتوي العنوان المنطقي على حقلين :

الأول : يشير الى رقم الصفحة (Page number)

الثاني : يشير الى مقدار الازاحة (displacement : offset) داخل الصفحة ويشترط أن تكون قيمة الازاحة مساوية أو أقل من حجم الصفحة وإلا سوف يكون هناك خطأ في العنوان .

تتمثل عملية تحويل العنوان المنطقي الى عنوان فيزيائي وذلك بتحديد رقم الصفحة ومعرفة حجم الصفحة ويبين الشكل (2 - 20) كيفية تحويل العنوان المنطقي الى عنوان فيزيائي .



الشكل (2 - 20) : آلية احتساب العنوان الفيزيائي

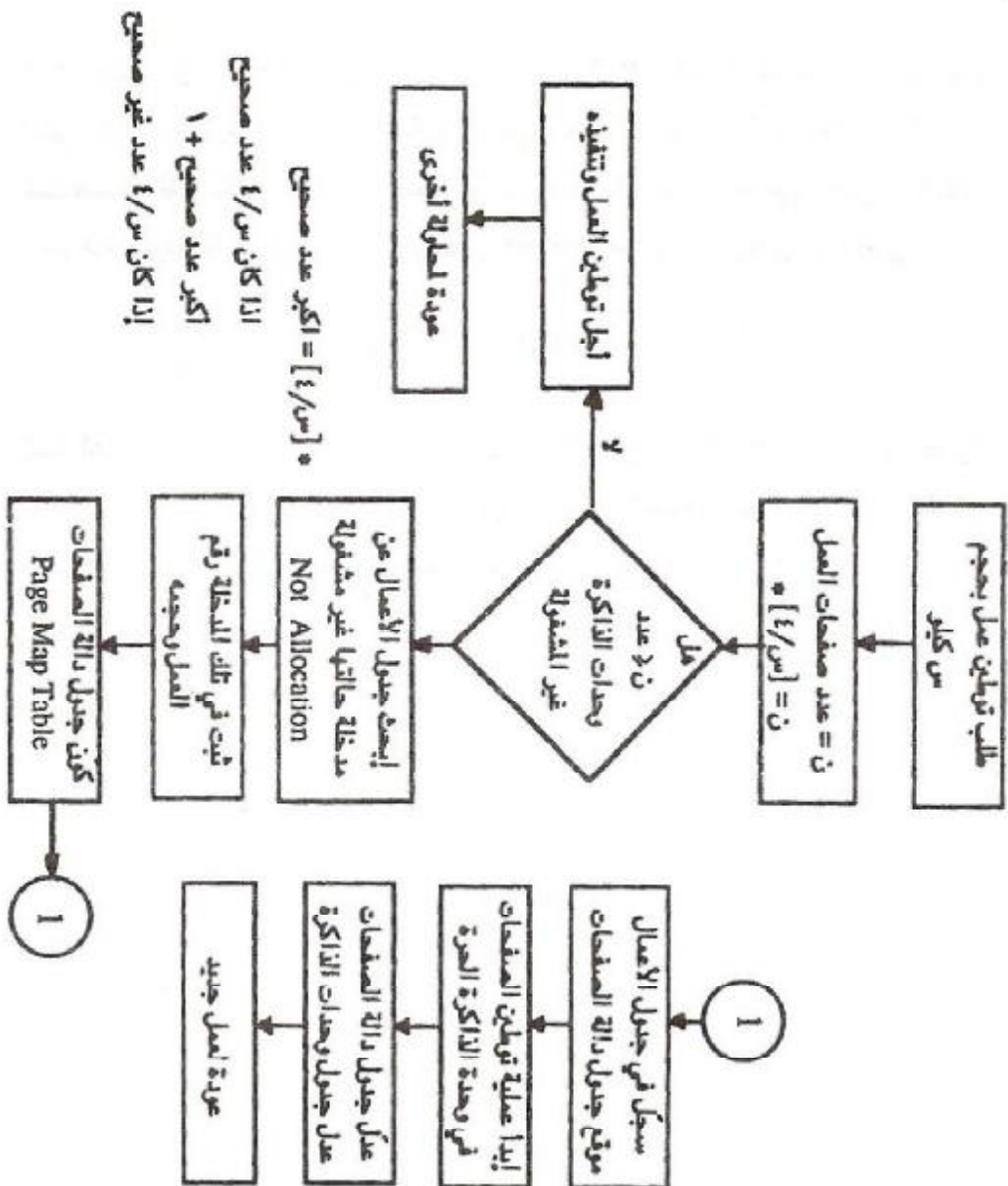
وللشكل (2 - 19) اذا افترضنا ان العنوان المنطقي هو :

3	500
---	-----

فهذا يعني ان الصفحة هي الثالثة ، من جدول الصفحات نحدد رقم الوحدة التي تقع فيها هذه الصفحة الا وهي الوحدة السادسة ، لنفترض أنه تم تخصيص 40 كتلة لنظام التشغيل وبهذا فان الصفحة المطلوبة تقع في الكتلة رقم 46. ولما كان حجم الصفحة هو 8K إذا العنوان الفيزيائي يساوي :

$$46 \times 8K + 500$$

كما اشرنا سابقاً تستخدم المعدات المختلفة لإدارة الذاكرة وتحقيق عملية التوطين اللازمة للعمل المراد تنفيذه ويبين الشكل (2 - 21) آلية عملية التوطين في إدارة الذاكرة بالصفحات .



الشكل (2 - 21) توطين العمل في إدارة الذاكرة

مزايا وعيوب إدارة الذاكرة بالصفحات

من مزايا وعيوب هذه الإدارة زيادة درجة تعدد البرامج والتقليل ما أمكن من الفراغات الخارجية وبالتالي تحسين أداء المعالج وتحسين استعمال الذاكرة أما عيوب هذه الإدارة فتنحصر في ما يلي :

- 1 - لا زال حجم العمل مقيداً بحجم الذاكرة ولا زالت عملية التنفيذ تتطلب تحميل العمل كاملاً في الذاكرة .
- 2 - وجود الفراغات الداخلية داخل الصفحة .
- 3 - كثرة المعدات الفيزيائية .

2 - 5 إدارة الذاكرة الفعالة

Virtual storage management

لاحظنا عند دراستنا للإدارات السابقة بأن حجم العمل المراد تنفيذه مرتبط بحجم الذاكرة المتوفر وحتى تتم عملية التنفيذ لا بد من تحميل العمل كاملاً في الذاكرة .

تستخدم بعض أنواع إدارات الذاكرة القرص المغناطيسي في عملية التنفيذ ويكتفى بتحميل جزء من العمل إلى الذاكرة الرئيسية وإبقاء الأجزاء الأخرى في القرص المغناطيسي إذا لم يكن هناك مستسعاً في الذاكرة الرئيسية وفي هذه الحالة يعتبر القرص امتداداً للذاكرة الرئيسية ويسمى الذاكرة الفعالة، وبهذا يلغي هذا النوع من الإدارات ارتباط حجم العمل بحجم الذاكرة المتوفر ويقال إن الذاكرة تستغل بأكثر من 100% .

هذا وسوف نطلع في هذا البند على بعض أنواع الإدارات والتي تستخدم مفهوم الذاكرة الفعالة .

2 - 5 - 1 إدارة الذاكرة بالصفحات المطلوبة

Demand paged memory managment

تشبه هذه الإدارة إدارة الذاكرة بالصفحات من حيث تقسيم الذاكرة الى كتل متساوية في الحجم ومن مضاعفات الرقم 2 وكذلك تقسيم العمل الى صفحات متساوية في الحجم ومن مضاعفات الرقم 2 ويمكن تنفيذ العمل وذلك بتحميله كلياً أو جزئياً في الذاكرة ويمكن أن ينفذ العمل الموجود صفحة واحدة منه في الذاكرة أما باقي الصفحات فيمكن الاحتفاظ بها في الذاكرة الفعالة (القرص المغناطيسي) وتشبه آلية تنفيذ الأمر والحصول على العنوان المنطقي الآلية المستخدمة في إدارة الذاكرة بالصفحات وذلك في حالة كون الصفحة متوفرة في الذاكرة، أما إذا كانت حالة الصفحة المطلوبة غير موطنة فهذا يستدعي تنفيذ اعتراض يسمى اعتراض الصفحة (Page fault Interrupt) ويؤدي تنفيذ هذا الاعتراض الى تنفيذ الخطوات التالية :

- 1 - تحديد موقع الصفحة المطلوبة في القرص المغناطيسي .
- 2 - البحث عن كتلة في الذاكرة الرئيسة حالتها حرة .
- 3 - في حالة توفر الكتلة ترحل الصفحة المطلوبة الى الذاكرة وتعديل الجداول اللازمة وتتابع عملية تنفيذ الأمر باحتساب العنوان الفيزيائي .
- 4 - في حالة عدم توفر كتلة في الذاكرة يتم البحث في الذاكرة عن صفحة (كتلة) معينة (بأحدى طرق الترحيل التي سوف نطلع عليها لاحقاً) لتنفيذ عملية المبادلة (swapping) وذلك بترحيل صفحة من الذاكرة الرئيسة لتحل محلها الصفحة المطلوبة وتعديل الجداول اللازمة ثم تتابع عملية التنفيذ باحتساب العنوان الفيزيائي .

ولمتابعة الذاكرة وإدارتها لا بد من توفر المعدات التالية :

- 1 - جدول دالة الصفحات (PMT) وتحتوي كل مدخلة في هذا الجدول على ثلاثة حقول هي :

- حقل رقم الصفحة .

- حقل حالة الصفحة هل هي موطنة في الذاكرة الرئيسية أم لا .

- حقل رقم الكتلة في الذاكرة الرئيسية والتي تقع فيها الصفحة .

ويتم تخصيص جدول لكل عمل يراد تنفيذه .

2 - جدول الكتل (block table) وتحتوي كل مدخلة في هذا الجدول على حقلين هما رقم الكتلة وحالتها (مشغولة أو غير مشغولة) .

3 - جدول دالة الملف (File map table) ويتم تخصيص جدول لكل عمل بحيث تحتوي كل مدخلة في هذا الجدول على حقلين هما رقم الصفحة وحقل عنوان الصفحة في الذاكرة الفعالة (القرص المغناطيسي) وتخزن في هذا الجدول أرقام صفحات العمل غير المحملة في الذاكرة الرئيسية .

4 - مسجل عنوان جدول دالة الصفحات ويخصص مسجل لكل عمل ويحتوي هذا المسجل على موقع جدول دالة الصفحات الخاص بالعمل المحدد .

5 - ترتبط بالصفحة المحملة في الذاكرة الرئيسية 3 خلايا ثنائية تسمى بالأقفال وتخصص كل خلية لوظيفة محددة وهذه الخلايا هي :

1 - خلية الاسناد (Reference bit) وتأخذ هذه الخلية القيمة واحد عندما تستخدم الصفحة (أو تعليمة فيها) في التنفيذ وتصفر إذا كانت الصفحة المعنية غير خاضعة للتنفيذ .

ويمكن ربط الصفحة بعدد (Counter) يشير الى عدد مرات استخدام الصفحة حيث تزداد قيمة العداد بمقدار واحد مجرد تغير قيمة خلية الاسناد الى الواحد ويمكن أن يستخدم هذا العداد لغايات ترحيل الصفحات .

ب - خلية التغير (Change bit) أو خلية المبادلة وتصبح قيمة هذه الخلية مساوية للواحد عند اختيار الصفحة المحددة لتنفيذ عملية المبادلة وذلك بترحيل الصفحة المحددة الى الذاكرة الثانوية لتحل محلها صفحة مطلوبة للتنفيذ .

ج - خلية الحماية (Protection bit) وتكون قيمة هذه الخلية مساوية للواحد اذا كانت الصفحة محمية والا فان قيمة هذه الخلية تكون مساوية للواحد .

ولترحيل الصفحات يمكن استخدام إحدى الطرق التالية :

1 - طريقة الداخل أولاً خارج أولاً (First in first out)

ترتب الصفحات المحملة في الذاكرة في جدول ترتيباً تصاعدياً حسب زمن قدوم الصفحة للذاكرة الرئيسية وعند تنفيذ عملية الترحيل يتم اختيار الصفحة الاولى .

2 - طريقة الأقل استخداماً (Least recently used)

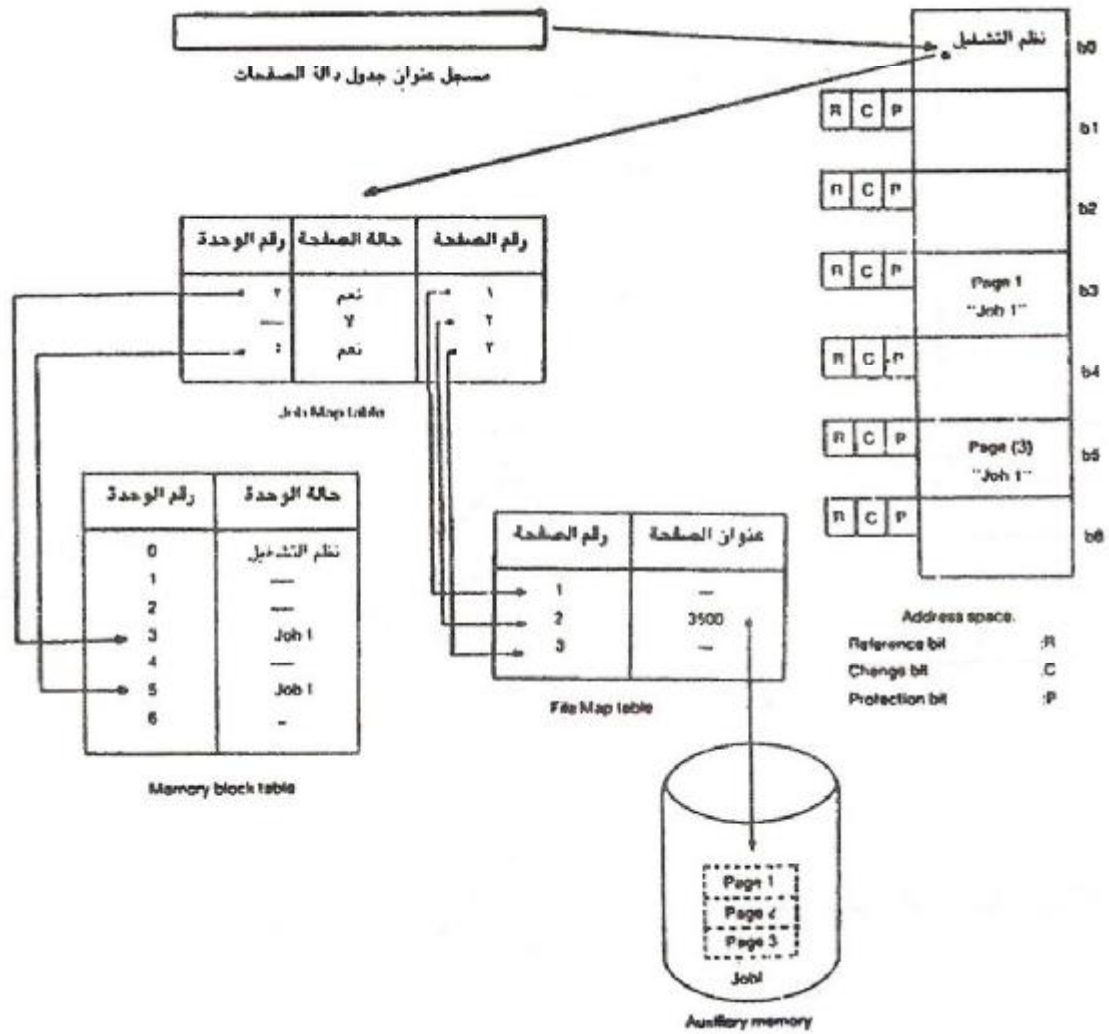
يستخدم في هذه الطريقة عداد عدد مرات استخدام الصفحة وترتب الصفحات المتواجدة في الذاكرة ترتيباً تصاعدياً حسب قيم العدادات الخاصة بالصفحات وعند تنفيذ عملية الترحيل يتم اختيار الصفحة الاولى في الجدول وهي الصفحة الأقل استخداماً .

3 - طريقة المزاوجة والربط (Touple coupling)

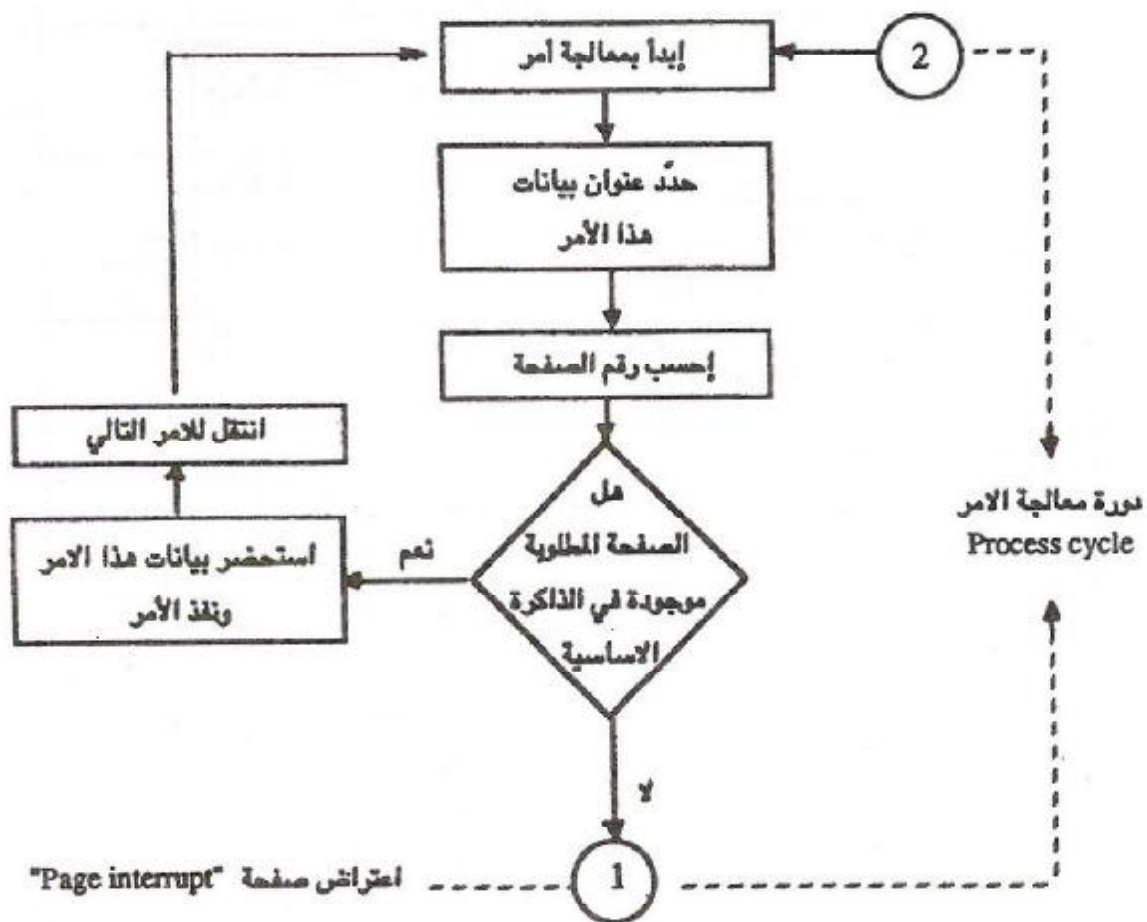
في هذه الطريقة تحدد الصفحة المراد ترحيلها ويتم ربطها بالصفحة الأقل استخداماً في لحظة ما بحيث ترحل الصفحة المحددة والصفحة الأقل استخداماً معاً .

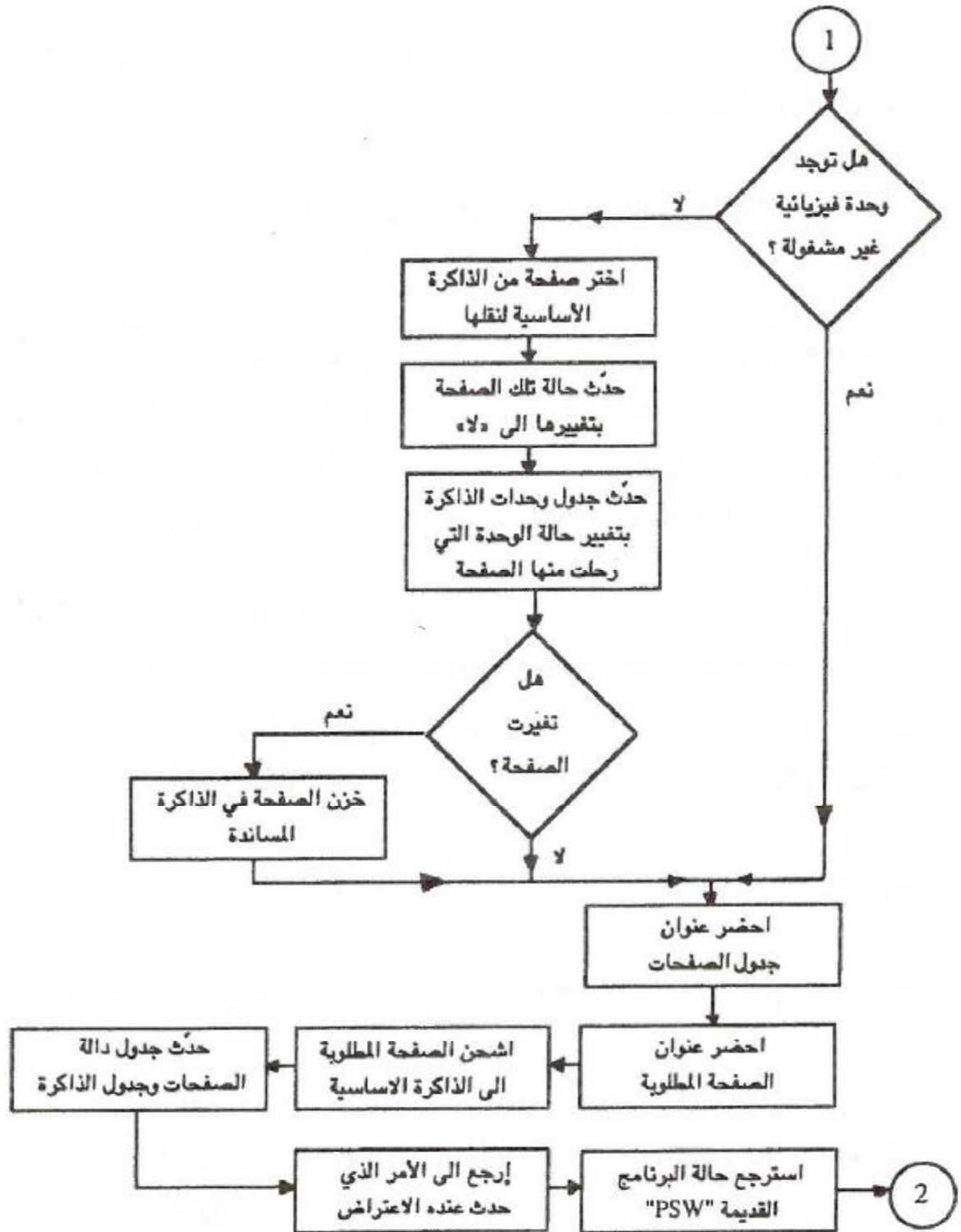
ويبين الشكل (2 - 22) المعدات اللازمة لإدارة الذاكرة بالصفحات المطلوبة .

كما ويبين الشكل (2 - 23) آلية تنفيذ الأمر في إدارة الذاكرة بالصفحات المطلوبة .



الشكل (2 - 22) : المعدات الفيزيائية لإدارة الذاكرة بالصفحات المطلوبة





الشكل (2 - 23) : مخطط سير عمليات طلب صفحة

مزايا وعيوب إدارة الذاكرة بالصفحات المطلوبة :

من مزايا هذه الإدارة امكانية استخدام ذاكرة رئيسة بحجم صغير نسبياً نظراً لاستخدام الذاكرة الفعالة كامتداد للذاكرة الرئيسية ومن المزايا الرئيسية لهذه الإدارة هي إلغاء تقيد حجم العمل المراد تنفيذه بحجم الذاكرة المتوفر وبالتالي تحسين استخدام الذاكرة وزيادة درجة تعدد البرامج أما مساوئ هذه الإدارة فتكمن في :

- 1 - كثرة المعدات الفيزيائية والتي تتطلب حيزاً في الذاكرة ووقتاً من وحدة المعالجة المركزية لمتابعتها وذلك باستحداثها أو تعديل محتوياتها عند الحاجة .
- 2 - ارتفاع زمن المعالجة وذلك لتطلب عملية التنفيذ زمناً لاحتساب العنوان الفيزيائي وزمناً لأجراء عملية الترحيل والبحث عن الصفحة .

2 - 5 - إدارة الذاكرة بالقطع (الشرائح)

Segmentation : segmented memory management

أشرنا سابقاً الى تساوي الصفحات في الحجم وهذا بدوره يؤدي الى تسهيل عملية احتساب العنوان الفيزيائي ولكن قد يؤدي الى ظهور مشكلة الفراغات الداخلية .

وللتخلص من مشكلة الفراغات الداخلية يمكن استخدام الشرائح وفيما يلي أهم مفاهيم إدارة الذاكرة بالشرائح :

- 1 - تعرف الشريحة على أنها وحدة منطقية للبيانات ولا تحتوي على فراغات داخلية .
- 2 - شرائح العمل الواحد غير متساوية في الحجم .
- 3 - يمكن إخضاع الشريحة للحماية كأن تكون للقراءة فقط أو للكتابة فقط أو للقراءة والكتابة .
- 4 - امكانية ربط الشريحة بأكثر من عمل وتشارك أعمال مختلفة في الشريحة .

5 - إمكانية تجميع الفراغات الخارجية وذلك من خلال عملية تجميع المناطق الحرة وضغط الذاكرة .

6 - استخدام الذاكرة الفعالة كامتداد للذاكرة الرئيسية .

ومن الأمثلة على الشرائح : الاقترانات المكتبية الذاتية، البرامج الفرعية والماكروية، الجداول، مناطق التخزين المؤقتة .

ولإدارة الذاكرة بالشرائح لابد من توفر المعدات التالية :

1 - جدول دالة القطع (Segment map table : SMT) ويتم تخصيص جدول لكل عمل وتحتوي كل مدخلة في هذا الجدول على الحقول التالية :

- حقل رقم الشريحة .

- حقل نوع الوصول ويحدد هذا الحقل طريقة معالجة الشريحة هل هي للقراءة فقط أو للقراءة والكتابة .

- حقل الحالة ويحدد هذا الحقل تواجد الشريحة في الذاكرة الرئيسية أو في الذاكرة الثانوية (أي ان الشريحة موطنة في الذاكرة الرئيسية أم لا) .

- حقل الموقع ويحدد هذا الحقل موقع الشريحة في الذاكرة الرئيسية .

- حقل الحجم ويحدد هذا الحقل حجم الشريحة .

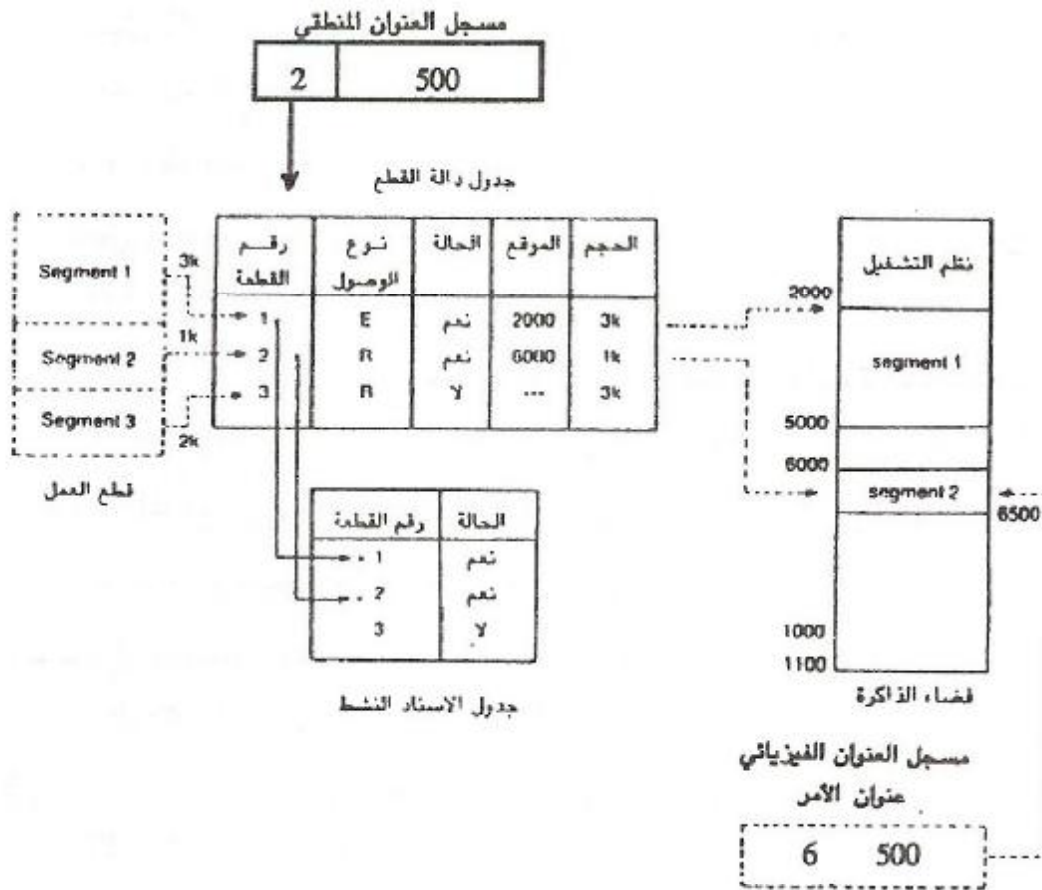
2 - جدول الاسناد النشط (effective reference table) ويحتوي هذا الجدول على معلومات تضم رقم الشريحة وحالتها .

3 - مسجل جدول دالة القطع (الشرائح) ويشير محتوى هذا المسجل الى موقع SMT الخاص بالعمل .

4 - مسجل العنوان المنطقي ويحتوي هذا المسجل على عنوان التعليم المراد تنفيذها وسوف نبين لاحقاً كيفية تحويل العنوان المنطقي الى عنوان فيزيائي .

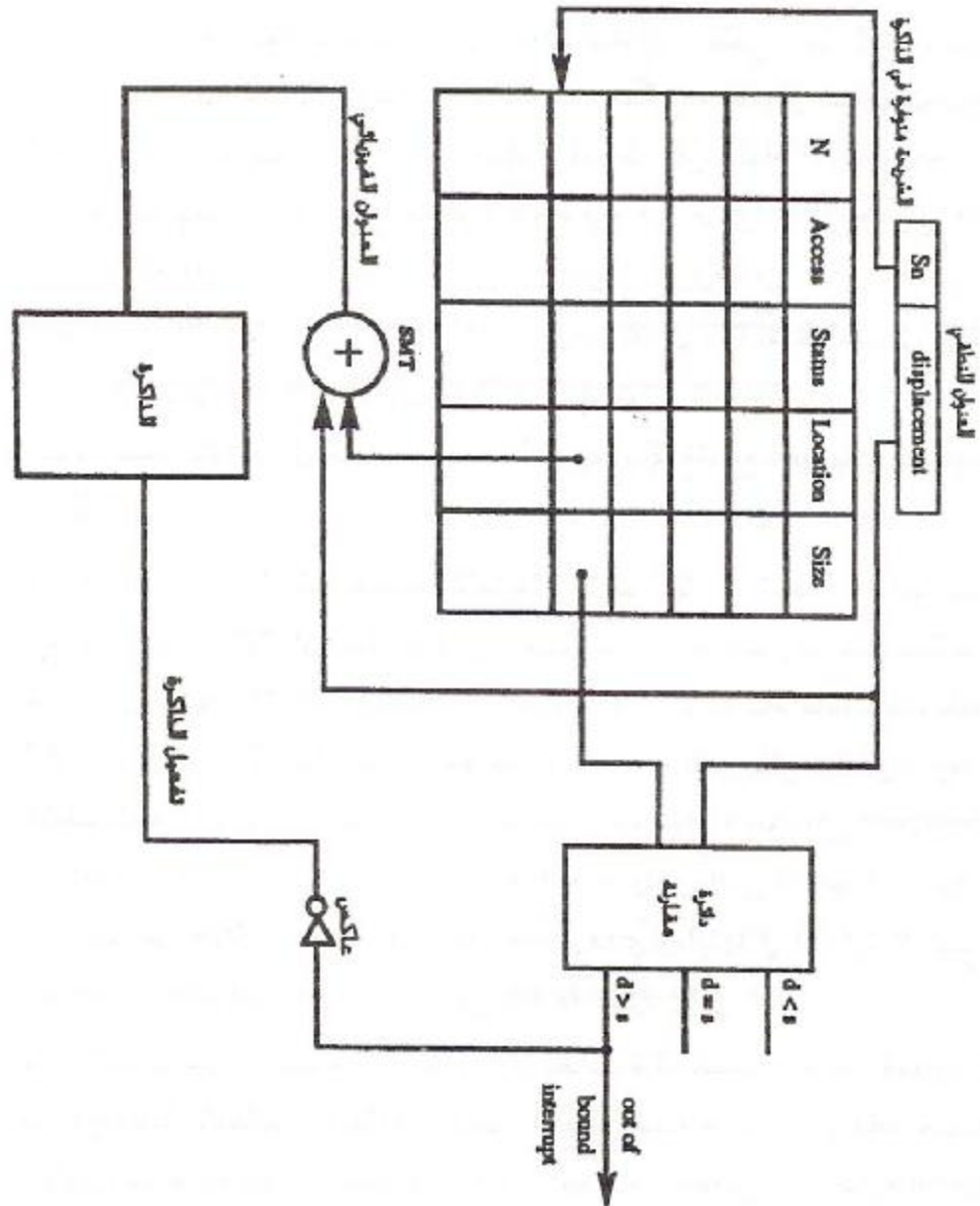
5 - مسجل العنوان الفعلي ويحتوي هذا المسجل على عنوان التعليم الفيزيائي في الذاكرة الرئيسية .

ويبين الشكل (2 - 24) المعدات المستخدمة في إدارة الذاكرة باستخدام الشرائح .



الشكل (2 - 24) : المعدات الفيزيائية لإدارة الذاكرة بالقطع

وفيما يلي نبين آلية تنفيذ أمر ما في إدارة الذاكرة بالشرائح :
 تتلخص آلية التنفيذ بتحويل العنوان المنطقي الى عنوان فيزيائي كما هو مبين
 في الشكل (2 - 25) .



الشكل (2 - 25) : آلية تنفيذ الأمر

وفيما يلي توضيحاً لما يتم من عمليات :

1 - يؤخذ رقم الشريحة من العنوان المنطقي ويستعرض جدول الاسناد النشط فإذا كانت الشريحة متوفرة في الذاكرة يحدد حجم الشريحة من الجدول . SMT

2 - يقارن حجم الشريحة بمقدار الازاحة في العنوان المنطقي فإذا كانت قيمة الازاحة أقل أو يساوي قيمة حجم الشريحة يتم اصدار اشارة لتنشيط الذاكرة ويتم الانتقال الى الخطوة التالية. أما اذا كان مقدار الازاحة أكبر من حجم الشريحة فهذا يعني محاولة الخروج عن حدود الشريحة وفي هذه الحالة يتم اصدار اعتراض يسمى اعتراض الخروج عن حدود الشريحة (Segment out of bound interrupt) ويؤدي تقبل هذا الاعتراض الى وقف تنفيذ الامر والغائه نظراً لوجود خطأ في العنوان .

3 - تضاف قيمة الازاحة الى موقع الشريحة للوصول الى الموقع الفيزيائي للتعليمية في الذاكرة .

وقبل البدء باحتساب العنوان الفيزيائي لا بد أولاً من تحديد ان الشريحة متوفرة في الذاكرة الرئيسية أم لا وفي هذه الحالة يتم أخذ رقم الشريحة من مسجل العنوان المنطقي ويستعرض جدول الاسناد النشط فإذا كانت حالة الشريحة متوطنة ننقل الى خطوة حساب العنوان الفيزيائي وإلا يتم وقف التنفيذ مؤقتاً باعتراض يسمى اعتراض الشريحة الاستثنائي (Segment exception interrupt) بحيث يشير هذا الاعتراض الى الحاجة الى جلب الشريحة من الذاكرة الثانوية، وبعد احضارها وتوطينها في الذاكرة الرئيسية تتم عملية احتساب العنوان الفيزيائي لمتابعة تنفيذ الأمر .

بعد تحديد عنوان التعليم المطلوبة في الذاكرة الرئيسية تحمل التعليمية في مسجل التعليمية وتقارن شيفرة التعليمية بنوع الوصول فإذا حدث تطابق تتم عملية متابعة التنفيذ أما إذا لم يحدث التطابق فإن عملية التنفيذ تلغى وذلك باعتراض يسمى اعتراض حماية الشريحة (Segment protection interrupt) .

عيوب إدارة الذاكرة بالشرائح :

- اضافة الى كثرة المعدات المستخدمة وما تحتاجه هذه المعدات من حيز ووقت للمتابعة إلا أن هناك زمناً اضافياً يمكن أن ينفق على :
- احتساب العنوان الفيزيائي والتأكد من صحته .
- تنفيذ عملية المبادلة بين الذاكرة الرئيسية والثانوية .
- تجميع الفراغات المبعثرة في الذاكرة وإجراء عملية الضغط تماماً كما هو الحال في إدارة الذاكرة بإعادة ترتيب المواقع .

2 - 5 - 3 إدارة الذاكرة بالشرائح والصفحات المطلوبة

Segment and demanded paged memory managment

تجمع هذه الإدارة بين خصائص إدارة الذاكرة بالشرائح وخصائص إدارة الذاكرة بالصفحات المطلوبة .

ينقسم العمل في هذا النوع من الإدارة الى قطع مختلفة الأحجام ، وكل قطعة تنقسم الى صفحات محددة الحجم بحيث يقل حجم الصفحة عن حجم القطعة ، ثم تنقسم الذاكرة الأساسية الى وحدات فيزيائية "Hardware blocks" بحيث يكون حجم الوحدة الواحدة مساوياً لحجم الصفحة .

بهذا الأسلوب نكون قد تخلصنا من العيوب التالية في إدارة الذاكرة بالقطع :

- 1 - فكرة إعادة ترتيب مواقع القطع "segment relocation" .
- 2 - صعوبة إدارة الأحجام المختلفة وغير الثابتة من القطع سواء كان ذلك في الذاكرة الأساسية أو الذاكرة المساندة .
- 3 - تبعية حجم القطعة الى حجم الذاكرة الأساسية وتتضح للقارئ طريقة التعامل مع القطعة أو الصفحة، فهي تتشابه مع ما ذكر في إدارة الذاكرة بالصفحات المطلوبة، باختلاف بسيط جداً، نجم عن عدم ادخال القطعة الى الذاكرة دفعة واحدة بل الى الجزء المطلوب فقط .

ولمتابعة الذاكرة الرئيسية وإدارتها لا بد من توفر المعدات الفيزيائية اللازمة وتختلف أنواع المعدات الفيزيائية التي تلزم هذا النوع من الإدارة باختلاف الحاسوب، لكنها تشترك بنفس أفكار المعالجة .

وهذه المعدات هي :

1 - مسجل العنوان الفعلي "Effective address register" وظيفته تحديد موقع الأمر المطلوب تنفيذه داخل صفحة ما من قطعة ما، بحيث تتحدد هذه الأمور في الحقول الثلاثة التالية :

أ - حقل رقم القطعة المطلوبة "Segmen number" .

ب - حقل رقم الصفحة المطلوبة "Page number" .

ج - حقل عنوان الأمر المطلوب من الصفحة "Byte number" .

2 - جدول دالة الصفحة "Page map table" ، وظيفته تحديد أرقام صفحات القطعة الواحدة وحالاتها إن كانت داخل الذاكرة الأساسية أو لا وأرقام الوحدات الفيزيائية المقابلة "Block number" .

3 - جدول دالة القطعة "Segment map table" ، تحتوي مدخلة هذا الجدول على الحقول التالية :

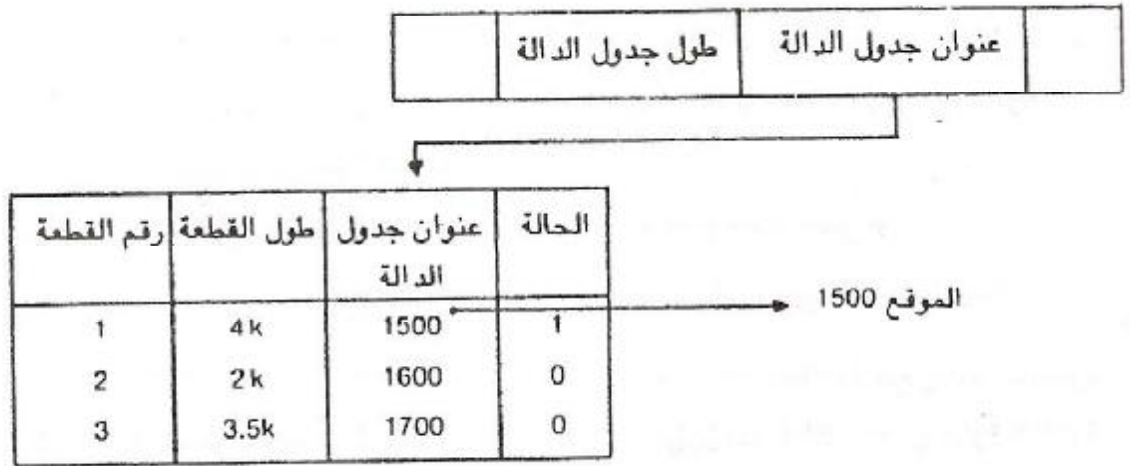
أ - رقم القطعة "Segment number" .

ب - طول القطعة "Segment length" .

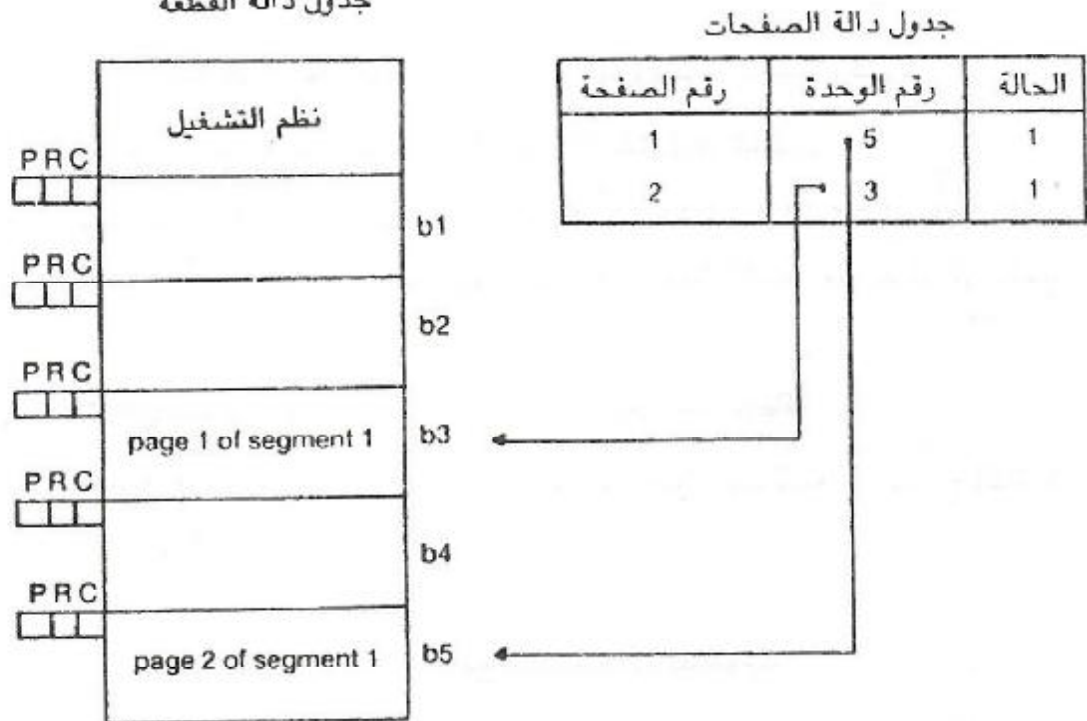
ج - عنوان جدول دالة الصفحة "Page table adress" .

د - رمز يدل على حالة القطعة إن كانت داخل الذاكرة الأساسية أم لا، ويعرف باسم "Segment exception code" .

ويبين الشكل (2 - 26) هذه المعدات .



جدول دالة القطعة



فضاء الذاكرة

الشكل (2 - 26) : المعدات اللازمة لإدارة الذاكرة بالصفحات والقطع المطلوبة .

4 - قفل حماية على كل وحدة فيزيائية "Protection lock" وخلية ثنائية تبين حالة الاسناد تعرف بخلية الاسناد "Reference" .

بالاضافة الى خلية التغير "Change bit" التي تبين حالة الصفحة الواحدة إن كانت واقعة تحت عملية المبادلة أم لا .

5 - مسجل عنوان جدول دالة القطعة "Segment map table address register" ، ويشتمل على حقلين أساسيين هما :

أ - حقل طول جدول دالة القطعة "Segment map table length" .

ب - حقل عنوان جدول دالة الصفحة "Segment map table address" .

تشابه إدارة الذاكرة بالقطع والصفحات المطلوبة مع إدارة الذاكرة بالصفحات المطلوبة، لذلك نرى أن الخوارزمية المتبعة في إدارة الذاكرة بالصفحات المطلوبة يمكن أن تتبع هنا .

وتجب ملاحظة أنواع الاعتراضات التي تطرأ أثناء المعالجة وهي :

1 - اعتراض الربط الاستثنائي (Linkage exception interrupt) :

يحدث هذا الاعتراض كلما حدث استدعاء لقطعة من الذاكرة المساندة لأجل تنفيذها داخل الذاكرة الأساسية، حيث تعدل حالة القطعة من صفر الى واحد في جدول دالة القطعة .

2 - اعتراض الصفحة الاستثنائي (Page exception interrupt) :

يحدث هذا النوع من الاعتراض كلما استدعيت صفحة من خارج الذاكرة الأساسية .

مميزات وعيوب إدارة الذاكرة بالقطع والصفحات المطلوبة

المميزات :

استطعنا في إدارة الذاكرة بالقطع والصفحات المطلوبة الحصول على ميزات إدارة الذاكرة بالقطع، وإدارة الذاكرة بالصفحات المطلوبة .

العيوب :

- 1 - ارتفاع كلفة الحاسوب، لزيادة المعدات اللازمة لإدارة الذاكرة .
 - 2 - صعوبة متابعة الجداول المختلفة الخاصة بالصفحات والقطع المطلوبة .
 - 3 - يمكن أن تنشأ كسور جزئية داخلية، كما هو مبين في المثال الموضح في شكل (3 - 25) . وذلك عندما يتم استدعاء القطعة رقم 3، حيث بلغ حجم القطعة 3,5 كيلوبايت ، فإذا انقسمت الى صفحات ، بحيث كان حجم الصفحة الواحدة 2 كيلوبايت ، نشأ لدينا صفحتان إحداهما شغلت فعلياً 1,5 كيلوبايت، وبقي ما حجمه كيلوبايت فارغاً من أي عمل .
- هذه هي العيوب المختلفة التي تنشأ عن إدارة الذاكرة بالقطع والصفحات المطلوبة ، وبالرغم من وجود هذه العيوب إلا أن معظم الحواسيب الكبيرة والصغيرة تستخدم هذا النوع من الإدارة .

2 - 6 : طرق أخرى في إدارة الذاكرة

بالإضافة الى مجموعة الطرق السابقة في إدارة الذاكرة توجد عدة طرق لمعالجة الذاكرة الأساسية في الحاسوب، ومن هذه الطرق ما يلي :

1 - المبادلة : "Swapping"

هي ترحيل عمل أو جزء عمل (صفحة، قطعة، جزء) من الذاكرة الأساسية الى الذاكرة المساندة وبالعكس ، وتوطين العمل أو جزء العمل في المكان المناسب من الذاكرة المساندة في نفس مواقع العمل أو جزء العمل المرحل .

أي ان العمل أو جزء العمل يتبادل موقعه مع العمل نفسه أو عمل آخر فيما بين الذاكرة الأساسية والمساندة .

لذا نجد ان عملية المبادلة تنقسم الى مرحلتين :

- 1 - مرحلة سحب العمل أو جزء العمل من الذاكرة المساندة الى الذاكرة الأساسية، تعرف باسم «الدحرجة الى الذاكرة الأساسية» أو "Rolling in" .

2 - مرحلة سحب العمل أو جزء العمل من الذاكرة الأساسية الى الذاكرة المساندة، وتعرف هذه العملية باسم «الدحرجة خارج الذاكرة» أو "Rolling out" بحيث تتم المرحلتان على نفس الموقع من الذاكرة الأساسية في الوقت ذاته .
وتحدث عملية المبادلة في كافة أنواع إدارة الذاكرة، ويمكن أن يزيد مفهوم المبادلة ما يعرف باسم المشاركة بالزمن "Time sharing" حيث تحدث المبادلة بكثرة .

مفهوم المشاركة بالزمن : "Time sharing"

هو أحد الأمثلة العملية على التشغيل المباشر، أو النظام المباشر "On-line system" . حيث يمكن لأكثر من مستفيد أو مستخدم أن يعمل على الجهاز في آن واحد، بحيث يعطى كل مستخدم قدرًا زمنيًا "Quantum or slice of time" لتنفيذ جزء من عمله بحيث يتولى تنفيذ أعمال كافة المستخدمين تباعاً بنفس القدر الزمني المعطى لكل منهم، ثم يبدأ النظام دورة أخرى مبتدئاً من المستخدم الأول وحتى المستخدم الأخير لتنفيذ قدرًا زمنيًا آخر من أعمال المستخدمين جميعاً ، فإذا تم انتهاء عمل ما من مجموعة الأعمال المرتبطة في النظام، خرج هذا العمل ووزع هذا القدر على مجموعة الأعمال الباقية .

ولتوضيح ذلك نضرب المثال التالي :

لدينا 3 مشتركين على جهاز ما، بحيث كانت الأوقات الزمنية اللازمة لتنفيذ الأعمال معطاة حسب الجدول التالي :

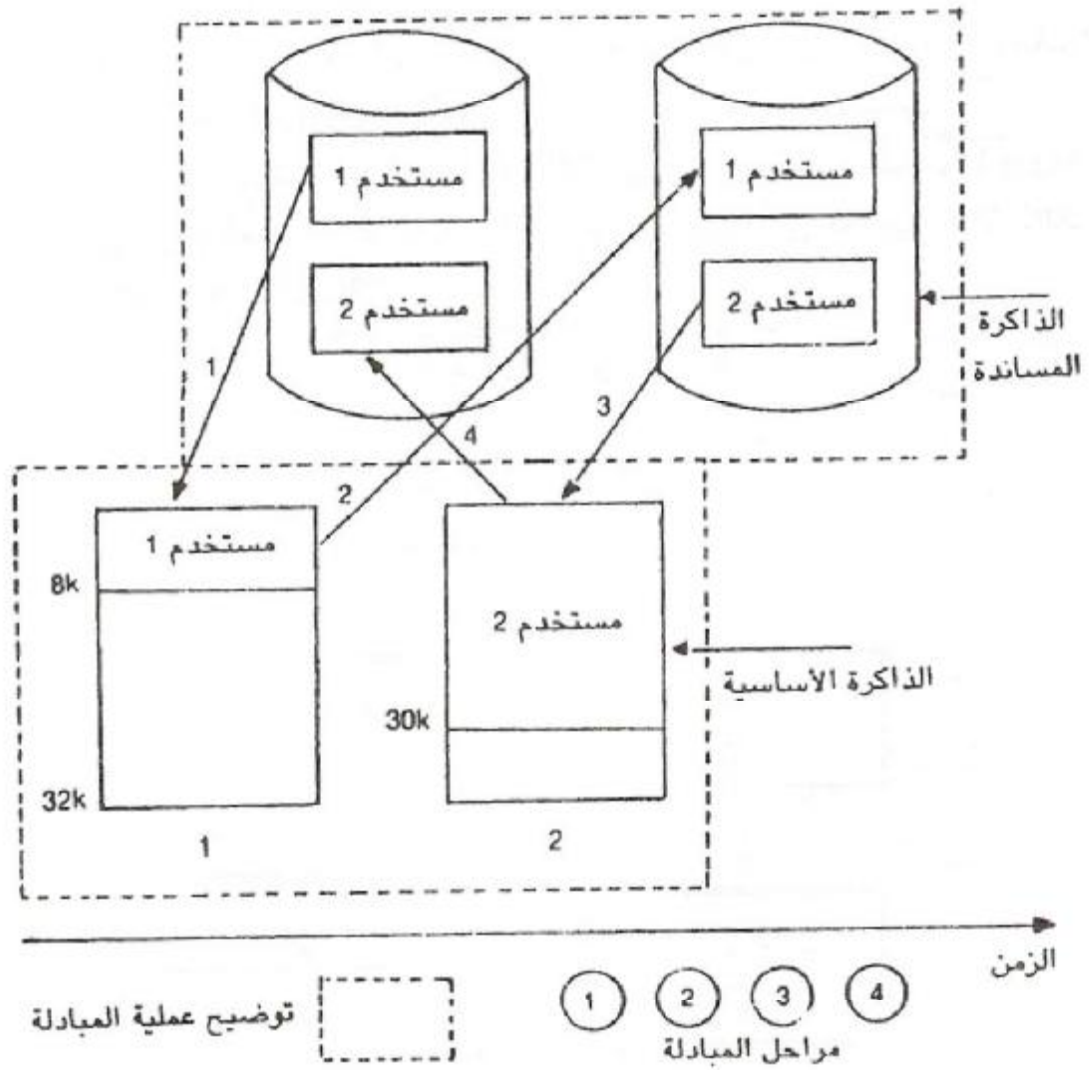
العمل	الزمن
1	15 دقيقة
2	8 دقائق
3	10 دقائق

وكان القدر الزمني "Quantum of time" خمس دقائق لكل عمل، على ذلك يكون تنفيذ الأعمال على النحو التالي :

المرحلة	رقم العمل	ترتيب العمل في التنفيذ	الزمن المستغرق في التنفيذ	الزمن الباقي
دورة 1	1	1	5	10
	2	-	-	8
	3	-	-	10
	1	1	5	10
	2	2	5	3
	3	-	-	10
	1	1	5	10
	2	2	5	3
	3	3	5	5
دورة 2	1	4	10	5
	2	2	5	3
	3	3	5	5
	1	4	10	5
	2	5	8	-
	3	3	5	-
	1	4	10	5
	2	5	8	-
	3	6	10	-
دورة 3	1	7	15	-
	2	5	8	-
	3	6	10	-

من الملاحظ ان المراحل المختلفة تعبر عن 3 دورات، مدة كل دورة 15 ثانية على الاكثر وبهذا يتفق مفهوم المشاركة بالزمن مع مفهوم تعدد البرامج باعطاء الفرصة لأكثر من مستفيد أن يستخدم جهاز الحاسوب في آن واحد، ولكن يختلفان في أن مفهوم المشاركة بالزمن لا يشترط قدرة الجهاز على استقبال أكثر من عمل في آن واحد، فقد يحدث أن ينفذ الحاسوب جزء العمل الأول في الوقت الذي تكون فيه الأعمال الأخرى خارج الذاكرة الأساسية. بينما مفهوم تعدد البرامج يشترط قدرة الحاسوب على استقبال أكثر من عمل في آن واحد، وادخالها الى الذاكرة، أما الأمر الثاني فان مفهوم تعدد البرامج لا يقتضي توقف تنفيذ الأعمال في لحظة تنفيذ عمل ما، بل يقتضي تنفيذ أجزاء الأعمال الأخرى بعمليات أخرى مثل عمليات الإدخال والإخراج مثلاً، في حين أن أجزاء أخرى من أعمال أخرى تقوم بتنفيذ عمليات المعالجة الحسابية والمنطقية وما الى ذلك .

ويبين الشكل (2 - 27) : مفهوم المبادلة في نظام المشاركة بالزمن .



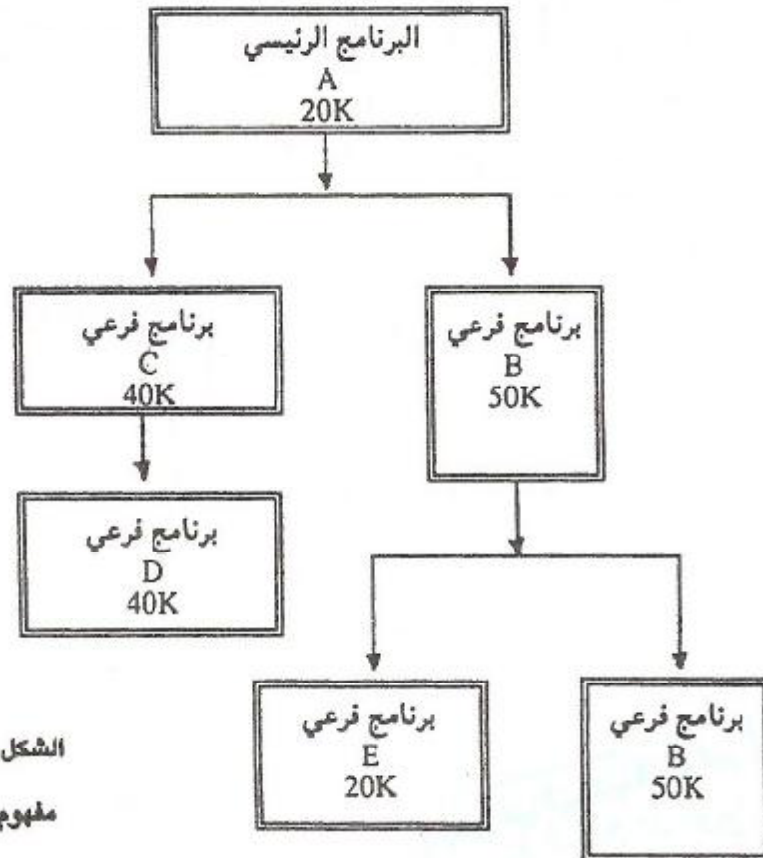
الشكل (2 - 27) : مفهوم عملية المبادلة

ب - التغطية "Over Laying"

هي عملية محسنة لعملية المبادلة، حيث يتم سحب أجزاء العمل إلى الذاكرة الأساسية على نفس الموانع التي كانت مشغولة من الأعمال التي تم ترحيلها إلى الذاكرة المساندة، سواء كانت أحجام أجزاء الأعمال المرحلة أكبر أو أصغر أو تساوي أحجام المواقع المفرغة .

لتوضيح ذلك سنعتبر العمل التالي الذي يتكون من مجموعة روتينيات، بعضها مستقل والبعض الآخر معتمد على أجزاء أخرى من العمل .

هذا العمل يوضحه الشكل (2 - 28) ويبين فيه التوزيع الهرمي لمجموعة روتينيات العمل A, B, C, D, E, F التي أحجامها على الترتيب 30K, 20K, 40K, 20K, 50K, 20K



الشكل (2 - 28)
مفهوم التغطية

لاحظ أن أكبر حجم للبرنامج الرئيسي وتفرعاته هو 100 كيلوبايت، فإذا لم تتوفر مساحة ذاكرة على الأقل بحجم «100» كيلوبايت فإن الروتين F لن ينفذ على الإطلاق، وذلك عند استدعاء F,B,A .

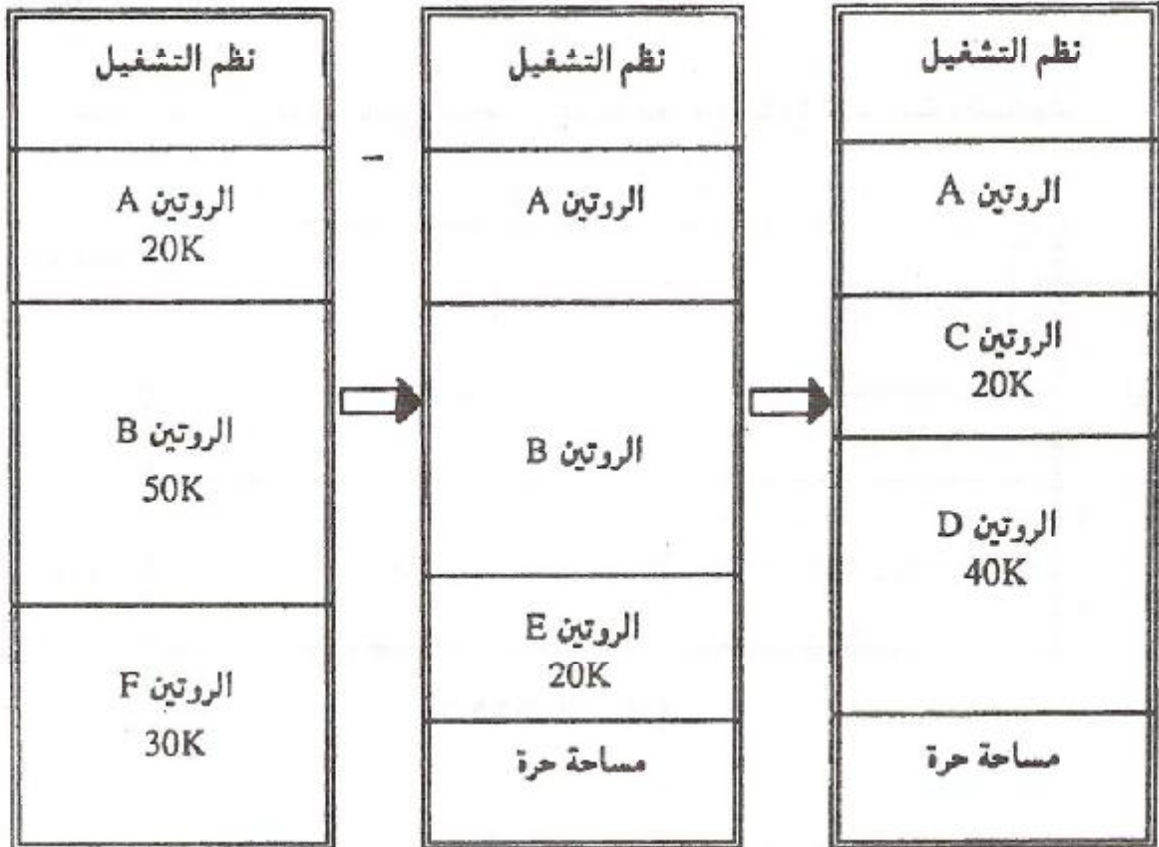
والآن لنفترض أننا قمنا بتنفيذ البرنامج والبرامج الفرعية على 3 مراحل هي:

أ - تنفيذ F,B,A

ب - تنفيذ E,B,A

ج - تنفيذ D,C,A .

سيظهر الشكل (2 - 29) هذه المراحل الثلاثة، ويبين كيف أن الروتين F قد سحب من الذاكرة الأساسية وغطيت مساحة بحجم الروتين E من المساحة التي كان يحتلها الروتين F . وكذلك نرى في الروتين E,B والروتين D,C . أنظر الشكل .



أ - عند استدعاء (ABF)

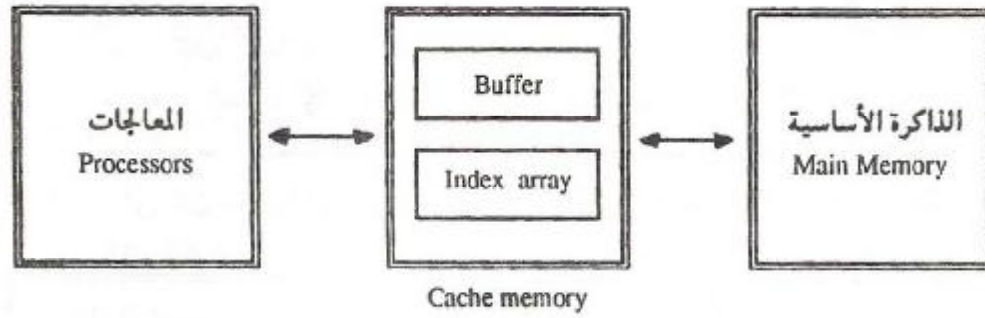
ب - عند استدعاء (ABE)

ج - عند استدعاء (ACD)

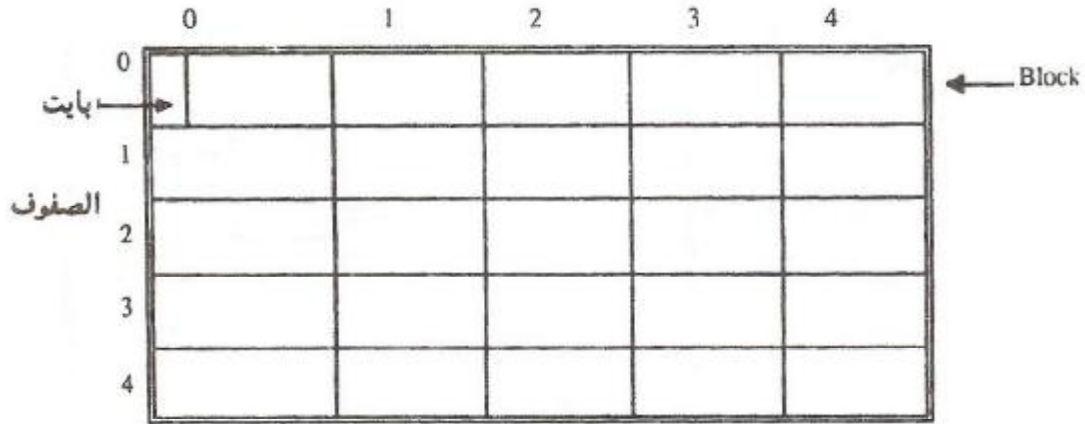
الشكل (3 - 29) : كيفية سحب وتغطية الاعمال

جـ - إدارة الذاكرة المؤقتة : "Management of buffering memory"

تعرف هذه الذاكرة بإسم الذاكرة المخزنة (Chach memory) وتقع الذاكرة المؤقتة بين المعالجات CPU processors، والذاكرة الأساسية كما هو موضح في الشكل (2 - 30) .



الشكل (2 - 30) موقع الذاكرة المؤقتة بالنسبة لكل من الذاكرة الأساسية والمعالجات



الشكل (2 - 31)

أما الشكل (3 - 31) فيبين تركيب الذاكرة المؤقتة، التي تتألف من وحدات فيزيائية (Blocks) مرتبة في صفوف وأعمدة مرقمة من صفر وحتى عددها

وكل وحدة فيزيائية مكونة من 32 بايت، أما عن تركيب منظومة الفهرس / Index array فإن كل مدخلة في هذه المنظومة تتألف من ثلاث حقول هي :

أ - حقل رقم سطر الوحدة : "Row number of block"

ب - حقل رقم عمود الوحدة : "Column number of block"

ج - حقل رقم البايت : "Byte number" .

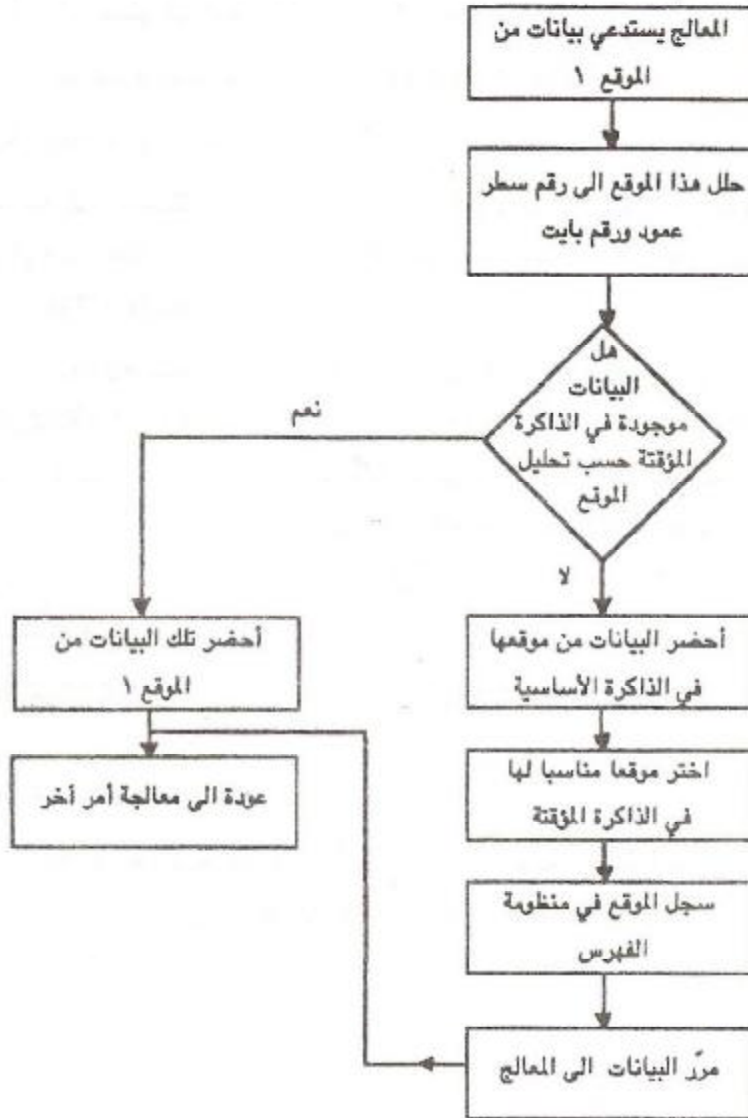
بهذه الحقول الثلاثة نستطيع تحديد موقع أي بايت في الذاكرة المؤقتة، لذلك يلزم تركيب الذاكرة المؤقتة منظومة الفهرس "Index number" . وتعمل الذاكرة المؤقتة على النحو التالي :

عند استدعاء مجموعة من البيانات من الذاكرة إلى المعالج، يتم فحص الذاكرة المؤقتة فيما إذا كانت تحتوي على البيانات المطلوبة أم لا، وذلك بفحص الموقع المطلوب عبر منظومة الفهرس . فإذا كانت جميع خلايا مدخلات هذه المنظومة خالية من البيانات، استدعيت البيانات من الذاكرة الأساسية إلى المواقع الفارغة في الذاكرة المؤقتة، وتم تعيين المواقع التي تم إشغالها في منظومة الفهرس .

ويوضح الشكل (2 - 32) مخطط سير عمليات إدارة الذاكرة المؤقتة .

مميزات الذاكرة المؤقتة :

تعد الذاكرة المؤقتة حلاً وسطاً بين الذاكرة السريعة عالية الثمن والذاكرة البطيئة رخيصة الثمن، بالإضافة إلى أنها تشكل الفكرة الأولى للذاكرة الفعالة "Virtual memory" .



الشكل (2 - 32) : إدارة الذاكرة المؤقتة .

الوحدة الثالثة

برمجيات إدارة وحدة المعالجة المركزية

Processor Managment

- مكونات برمجيات إدارة المعالج ووظائفها .
- جدولة الأعمال .
- طرق جدولة الأعمال .
- جدولة العمليات .
- أنظمة الحواسيب متعددة المعالجات .
- المشاكل المتعلقة بالمعالجات والعمليات :
 - مشكلة السباق .
 - الوضع الهامد .
- ميكانيكية التزامن .

برمجيات إدارة وحدة المعالجة المركزية Processor Managment

3 - 1 مكونات برمجيات إدارة وحدة المعالجة ووظائفها :

تتكون برمجيات إدارة المعالج من الأجزاء الرئيسة التالية :

أ - جدول الاعمال والبرامج (Job Scheduler)

ب - جدول المعالجات والعمليات (Process scheduler)

ج - ضابط التحكم (Traffic controller)

وترتبط هذه الأجزاء معاً لتحقيق وتنفيذ الوظائف التالية :

- 1 - استطلاع الذاكرة المساندة وذلك لانتقاء الأعمال المستعدة للتنفيذ ويقوم بهذه المهمة جدول الاعمال الذي يتصل ببرمجيات إدارة المعلومات وذلك لتمرير البرامج أو الملفات المستعدة للتنفيذ الى الذاكرة الرئيسة حيث يتم تقسيم هذه البرامج والملفات الى خطوات عمل مؤلفة من عمليات يعمل جدول الاعمال على جدولتها وتنظيمها من أجل تنفيذها لاحقاً .
- 2 - متابعة المعالج والعمليات أثناء تنفيذها وجدولة عملية التنفيذ ويقوم بهذه المهمة جدول العمليات .
- 3 - تحقيق عملية التزامن بين المعالج والعمليات وتحديد حالة المعالج ويقوم بهذه المهمة ضابط التحكم .
- 4 - رسم سياسة متابعة العمليات والمعالجات وتحديد طريقة ربط المعالج بالعملية وفي الزمن المحدد ولفترة محددة ويتولى تنفيذ هذه المهمة جدول العمليات .
- 5 - ربط المعالج بالعملية لفترة زمنية محددة .
- 6 - تحرير المعالج من عملية الربط وذلك عند الانتهاء من تنفيذ عملية معينة

وترحيل العملية الى الذاكرة الرئيسة حتى يستطيع المعالج الارتباط بعملية أخرى لتنفيذها .

3 - 2 جدولة الأعمال :

إذا كان الحاسوب يمتلك المقدرة على تعدد البرامج فانه قد تكون هناك أعمال في الذاكرة المساندة مستعدة للتنفيذ ويقصد بجدولة الأعمال وضع هذه الأعمال في قائمة الاستعداد بحيث ترتب هذه الأعمال في الجدول ضمن أولويات محددة ويتم متابعة العمل من خلال مسجل خاص يسمى مسجل معرف العمل (job control block register) ويحتوي هذا المسجل على الحقول التالية :

- 1 - معرف أو اسم العمل .
- 2 - الحالة الحالية للعمل .
- 3 - أولوية العمل .
- 4 - الزمن اللازم (أو المتبقي) للتنفيذ .

عندما تصبح الأعمال قابلة للتنفيذ تنتقل من حالة التعليق الى حالة الاستعداد ويقوم مجدول الأعمال باستطلاع مسجل التعريف لكافة الأعمال للتعرف على حالة كل عمل ويختار الأعمال التي في حالة استعداد ويكون من هذه الأعمال قائمة الاستعداد بحيث ترتب هذه القائمة باستخدام طريقة جدولة محددة .

تتم عملية جدولة الأعمال وذلك للاعتبارات التالية :

- 1 - عدد المصادر المادية في الحاسوب محدود، لذا لا بد من جدولة الأعمال وذلك لتحقيق عملية الاستخدام الأمثل لهذه المصادر .
- 2 - بعض وحدات الحاسوب لا يمكن أن تكون مشتركة من قبل أكثر من عمل وخوفاً من مشكلة السباق والصراع على هذه الوحدات لا بد من جدولة

الاعمال بطريقة لا تسمح بوجود السباق أو الصراع .

3 - نظر لتعدد البرامج فإن تكلفة تنفيذ الأعمال ترتفع وذلك لزيادة زمن التنفيذ وزمن الانتظار لذلك لا بد من جدولة الأعمال لتحقيق التوازن بين عمليات المعالجة الحسابية وعمليات الادخال والاخراج ولتحديد وقت انتظار العمل بحيث لا يتجاوز هذا الوقت قيمة محددة .

3 - 3 طرق جدولة الأعمال "Methods of scheduling"

نهتم عند جدولة الأعمال بمبدايي ثلاثة ، كل منها يعتني بتقليل وقت الانتظار لكل عمل، ومعدل وقت الإنتظار لكافة الأعمال، وهذه المباديء هي :

1 - مبدأ الداخل أولاً هو الخارج أولاً "First in First out"

ولتوضيح هذا المبدأ لناخذ البيانات المبينة في الجدول (3 - 1)

الزمن مقدر بالساعة وبكسر عشري من الساعة أي أن 0,1 ساعة يساوي $60 \times 0,1 = 6$ دقائق .

الجدول (3 - 1) وصول الاعمال وفترات تنفيذها

رقم العمل	وقت الوصول	فترة التنفيذ
1	10.0	0.2
2	10.1	0.3
3	10.2	0.1
4	10.3	0.4

فإذا عرفنا وقت الانتظار بالرمز W ووقت الوصول بالرمز A ووقت الإنتهاء بالرمز F ، فيكون وقت الانتظار معرّفاً على النحو التالي :

وقت الانتظار = وقت الإنتهاء - وقت الوصول .

أو بالرموز $(W=F-A)$.

$$\frac{\text{مجموع أوقات الانتظار}}{\text{عدد الأعمال}} = \text{ويكون معدل وقت الانتظار}$$

$$\bar{W} = \frac{W}{N} \quad \text{أو بالرموز}$$

حيث $N = \text{عدد الأعمال}$

وإذا أردنا حساب وقت الانتظار W لكل عمل في الجدول السابق، ومعدل وقت الانتظار W فإننا نرى الجدول (3 - 2) الذي تم حساب الأوقات فيه بناءً على مبدأ "FIFO"

$$\bar{W} = \frac{1.7}{4} = 0.425 = \text{معدل وقت الانتظار}$$

الجدول (3 - 2) : حساب وقت الانتظار

رقم العمل	وقت الوصول	وقت بدء التنفيذ	وقت الإنتهاء	وقت الانتظار
1	10,0	10,0	10,2	0,2
2	10,1	10,2	10,5	0,4
3	10,2	10,5	10,6	0,4
4	10,3	10,6	11,0	0,7

ب - مبدأ تنفيذ أقصر عمل أولاً "Shortest job first"

ويعني هذا المبدأ ترتيب أولويات تنفيذ الأعمال حسب وقت التنفيذ اللازم لكل منها، بحيث يتم تنفيذ العمل الذي وصل أولاً ، ومن ثم يبدأ إختيار الأعمال التي تلي تبعاً لقصر مدة التنفيذ ، كما هو مبين في الجدول رقم (3 - 3) والمعتمدة على الجدول رقم (2 - 1)

الجدول (3 - 3) : وقت الانتظار

رقم العمل	وقت الوصول	وقت بدء التنفيذ	وقت الإنتهاء	وقت الإنتظار
1	10,0	10,0	10,2	0,2
2	10,1	10,3	10,6	0,5
3	10,2	10,2	10,3	0,1
4	10,3	10,6	11	0,4

$$\bar{W} = \frac{1.2}{4} = 0.3 = \text{معدل وقت الانتظار}$$

لاحظ ان الأعمال قد ترتبت حسب وقت التنفيذ لكل منها. وذلك حسب ما يبينه الجدول (3 - 4) .

الجدول (3 - 4) تسلسل الأعمال حسب وقت البدء

رقم العمل	1	3	2	4
زمن البدء	10,0	10,2	10,3	10,6
مدة التنفيذ	0,2	0,1	0,3	0,4

لاحظ أن معدل الانتظار في المبدأ الثاني أقل من معدل الانتظار في المبدأ الأول.

جـ - المبدأ الثالث: التنفيذ وفق معرفة المستقبل "FUTURE KNOWLEDGE"

في هذا المبدأ يتم جعل الحاسوب خاملاً من دون عمل حتى زمن وصول آخر الأعمال، ومن ثم ترتب كافة الأعمال تتابعياً من العمل ذي الوقت الأقصر للتنفيذ حتى العمل ذي الوقت الأطول للتنفيذ .

ويوضح الجدول (3 - 5) عمل هذا المبدأ ، والذي يعتمد على معلومات الجدول (3 - 1) :

الجدول (3 - 5) : وقت الانتظار

رقم العمل	وقت الوصول	وقت بدء التنفيذ	وقت الإنتهاء	وقت الإنتظار
1	10,0	10,4	10,6	0,6
2	10,1	10,6	10,9	0,8
3	10,2	10,3	10,4	0,2
4	10,3	10,8	11,2	0,9

$$\bar{W} = \frac{2.0}{4} = 0.625 = \text{معدل وقت الانتظار}$$

مدة فترة خمول الجهاز = 0.2

ومن الملاحظ أن معدل وقت الانتظار قد زاد عما هو عليه فيما سبق من المبادئ وليس بالضرورة أن يزيد معدل وقت الإنتظار عما هو عليه في المبدأين السابقين .

إن مبادئ الجدولة السابقة، قد تمت على إفتراض عدم قدرة الحاسوب على إستيعاب أكثر من عمل واحد في آن واحد .

وإذا أردنا أن نقوم بجدولة الاعمال المعطاة في الجدول (3 - 1) بإفتراض قدرة الحاسوب على تعدد البرامج سنجد ما يلي :

Scheduling when multiprogramming تعدد البرامج

إن قدرة الحاسوب على تعدد البرامج "Multiprogramming" تعني قدرة الذاكرة الاساسية على إستيعاب أكثر من برنامج في نفس الوقت، وقدرة الحاسوب على توزيع مصادره «الأشرطة، الأقراص، الطابعات، ...» بين هذه الاعمال .

وبالتالي سنجد أن زمن المعالجات سيتوزع بين الاعمال المختلفة، وهنا سنفترض توزع هذه الأوقات بالمعدل، أي أنه إذا وجد عملان في الذاكرة فإن وقت المعالجة سيتوزع مناصفة، وإذا كان هنالك 3 أعمال فإن وقت المعالجة سيتوزع بالثلث لكل عمل وهكذا، وذلك لتسهيل تصور مبدأ تعدد البرامج "Multiprogramming" .

الجدول (3 - 6) : الجدولة باستخدام تعدد البرامج

الزمن	الحدث	عدد الأعمال في الذاكرة	نسبة وقت المعالجة	الزمن المستغرق	الزمن المستنفذ لكل عمل	الأعمال	الزمن الباقى
10	وصول العمل (1)	—	—	—	—	1	0,02
10,1	وصول العمل (2)	1	$1 = \%100$	0,1	0,1	1	0,01
10,2	وصول العمل (3)	2	$1/2 = \%50$	0,1	0,05	2	0,3
10,3	وصول العمل (4)	3	$1/3 = \%33,3$	0,1	0,033	3	0,01
10,368	انتهاء العمل (1)	4	$1/4 = \%25$	0,068	0,017	4	0,017
10,518	انتهاء العمل (2)	3	$1/3 = \%33,3$	0,15	0,05	3	0,217
10,818	انتهاء العمل (3)	2	$1/2 = \%50$	0,3	0,15	2	0,067
10,848	انتهاء العمل (4)	1	$1 = \%100$	0,033	0,033	1	0,4

حيث أن : نسبة وقت المعالجة = $1 \div \text{عدد الأعمال في الذاكرة}$.

عدد الأعمال في الذاكرة = عدد الأعمال التي تم إدخالها الى الذاكرة قبل
وقوع الحدث مشار اليه .

الزمن المستغرق = زمن وقوع الحادث - الزمن المبين في الجدول .

= الفرق بين زمني حدثين متتالين .

الزمن المستفيد لكل عمل = نسبة وقت المعالجة × الزمن المستغرق .
الزمن الباقي = الزمن الباقي في الحدث السابق — الزمن المستنفذ لكل عمل
عند وقوع حدث معين .

ومن الجدول (3 - 6) يمكن ان نحسب أوقات الانتظار لكل عمل من الاعمال السابقة، ومعدل وقت الانتظار، وذلك حسب ما يشير إليه الجدول (3 - 7).

$$\bar{W} = \frac{1,944}{4} = 0,486 = \text{معدل وقت الانتظار}$$

يجدر بالقارئ ان يلاحظ ان أولويات تنفيذ الأعمال في حالة تعدد البرامج بأولويات ذات نمط متغير تعتمد على عدد الأعمال المنفذة في الذاكرة، وأوقات تنفيذ تلك الأعمال، وقد استخدمنا مبدأ الداخل أولاً هو الخارج أولاً "FIRST IN FIRST OUT".

الجدول (3 - 7) : وقت الانتظار باستخدام تعدد البرامج

رقم العمل	وقت الوصول	وقت بدء التنفيذ	وقت الإنتهاء	وقت الإنتظار
1	10,0	10,0	10,368	0,368
2	10,1	10,1	10,818	0,718
3	10,2	10,2	10,518	0,318
4	10,3	10,3	10,848	0,548

3 - 4 : جدولة العمليات : "Process scheduling"

ونعني بها وضع جميع المعالجات والعمليات حسب توافق زمني ولدد معينة، ويقوم بهذه الجدولة برنامج "Process scheduling" والذي يسمى أحياناً بالمنجز "Dispatcher".

ولمتابعة المعالجات والعمليات نستعمل المسجل "Register" الذي يحدد حالة العملية، ويعرف هذا المسجل باسم وحدة ضبط العملية "Process control block" ولها التوصيف التالي :

Process identification	Current state	Pointer to the list of the same status	Copy of active register	etc
------------------------	---------------	--	-------------------------	-----

Process control block

وهذه الحقول على الترتيب تعرف :

- 1 - العملية .
- 2 - حالة العملية وحالة استعداد ، إنتظار، تنفيذ، اكتمال، تعليق، .
- 3 - مؤشر يحدد موقع القائمة التي تحتوي على جميع العمليات ذات نفس الحالة .
- 4 - نسخة بالمسجلات النشطة .
- 5 - معلومات أخرى .

وهذه المعلومات تشير الى ان جميع العمليات ذات نفس الحالة توضع في قائمة واحدة تسمى باسم الحالة ، كقائمة الاستعداد "Ready List" وقائمة الانتظار "Wait state" وهكذا ...

وعند إجراء الجدولة الزمنية لربط العمليات بالمعالجات المناسبة فإنه تؤخذ عدة سياسات بعين الاعتبار .

سياسات جدولة العمليات :

تنظم العمليات في قائمة الاستعداد بنفس الأسلوب الذي تنظم به الأعمال في القائمة المناظرة، أي أن جميع العمليات ذات حالة الاستعداد تنتظم عشوائياً أو تتابعياً وببنفس أسلوب تنظيم الأعمال في قائمة الاستعداد .

والأمور الواجب مراعاتها عند جدولة هذ العمليات وإدخالها الى قائمة التنفيذ "Run List" هي :

1 - إتباع مبدأ الدور والمعروف بمبدأ "Round robin" والذي يعنى إعطاء كل عملية قدراً معيناً من الزمن ، كأن يكون مثلاً 100 ميكرو ثانية (Micro second) .

2 - مبدأ عكس باقي الدور : بحيث اذا استغرقت العملية القدر المحدد لها من الزمن بكامله، قفزت هذه العملية الى منتصف القائمة، وهكذا، وذلك لإعطاء باقي العمليات أزمّة متكافئة .

3 - مبدأ مضاعفة القدر الزمني : تُعطى كل عملية دخلت الى قائمة التنفيذ مجدداً عدداً من الاقدار الزمنية يساوي عدد المقادير الزمنية التي استغرقتها العمليات الموجودة في قائمة التنفيذ سابقاً، وذلك لكي تنفذ هذه العملية بنفس المقادير الزمنية التي نفذت بها العمليات السابقة أو لكي تعطى نفس أولوية التنفيذ فيما يلي من المقادير الزمنية .

4 - مبدأ الأولوية : تعطى بعض العمليات أولويات تنفيذ لأهمية هذه العمليات .

5 - مبدأ العدد المحدود من المقادير الزمنية، يبقى مبدأ إعطاء الدور Round robin قائماً لعدد محدود من المرات ثم تنفذ العمليات طبيعياً .

6 - مبدأ توازن عمليات الادخال أو الاخراج مع عمليات المعالجة الحسابية والمنطقية .

7 - مبدأ إعطاء الأفضلية لكافة العمليات بالأعمال النشطة أو ما يسمى "Interactive jobs" الأولوية العليا .

8 - تعطى الأولوية للعمل ذو عبء الإدخال أو الإخراج الأكبر . أحياناً حتى يحدث
إتزان في العمل على الحاسوب .

وتبعاً لهذه المبادئ السابقة تتم جدولة العمليات، وهي خطوة ضرورية
لتنفيذ الأعمال، وقد أشرنا إلى أن مرحلة جدولة العمليات تقع كخطوة تالية
لجدولة الأعمال .

ويمكن إتباع مبادئ جدولة الأعمال في جدولة العمليات .

3 - 5 : أنظمة الحواسيب متعددة المعالجات : Multiprocessors systems

لزيادة فعالية الحواسيب، ومرونة أدائها وزيادة كفاءتها الحسابية والمنطقية
تضاف مجموعة من المعالجات إلى الحاسوب، وقد تكون هذه المعالجات ذات
وظائف متخصصة، أو قد تكون وحدة معالجة مركزية مستقلة بنفسها،
وترتبط بها مجموعة من ملحقات الحاسوب "Comptures peripherals" ،
أو قد تكون معالجات موازية ، ومن أساليب ربط هذه المعالجات
العديدة ، والتي تسمى أنظمتها بأنظمة الحواسيب متعددة المعالجات
"Multiprocesser systems" ما يلي :

1 - معالجات متباعدة "Seperate systems" :

وتكون على شكل حواسيب مستقلة بذاتها، كل منها مكتمل التركيب من
ذاكرة مستقلة وطرفيات حسب اللازم .
وتستطيع المؤسسة التي تقني مثل هذه الأجهزة أن تنفذ الأعمال على
الحواسيب، مستقلة عن بعضها البعض دون أدنى إتصال .

2 - معالجة موجهة "Coordinated systems" :

وكل معالج يقرر بوظيفة خاصة به، ويرتبط بجهاز مركزي واحد .

3 - معالجات العبد / السيد "Master / slave system" :

وتخضع جميع المعالجات هنا إلى سيطرة مهيمنة من قبل معالج واحد، يعمل
عمل السيد مع العبيد .

أي أن المعالجات الأخرى لا تستطيع التنقل بين المعالجات الأخرى إلا من خلال المعالج مع السيد .

4 - المعالجات المتجانسة "Homogeneous system" :

يكون كل معالج قادراً على الإتصال والتحكم بالمعالجات الأخرى، وبنفس الوقت تتصل المعالجات بعضها ببعض في شبكة عمل واحدة . بذلك يتصف عمل كل معالج باللامركزية ، أي أن كل معالج يقوم بإنتقاء العملية "Process" ، من قائمة الإستعداد "Ready list" ، ليباشر بتنفيذها مستقلاً عن غيره من المعالجات . ولذلك تتطلب عملية جدولة المعالجات مهارة كبيرة ودقة عالية .

3 - 6 : المشاكل المتعلقة بالمعالجات والعمليات وطرق الحل المتبعة لذلك

تظهر المشكلات المتعلقة بالمعالجات والعمليات نتيجة لما يلي :

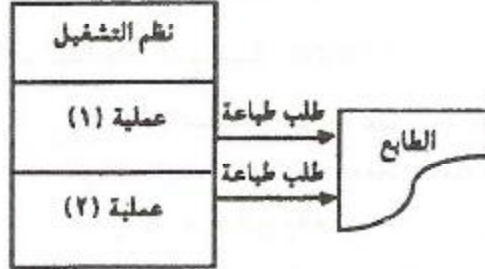
1 - تعدد المعالجات : إن العدد الكبير من المعالجات في الحاسوب يتطلب جهداً زائداً في المتابعة، سواء كانت تلك المتابعة من الناحية البرمجية أو من ناحية صيانة المعدات .

ب - جدولة المعالجات والعمليات زمنياً : إن ربط العمليات بالمعالجات لتنفيذها يحتاج توافقاً زمنياً «تزامنياً Synchronization» دقيقاً ، وتوجيهاً متقناً «Coordination» نحو الأهداف .

ومن أولى المشكلات التي تواجهنا في جدولة المعالجات والعمليات :

3 - 6 - 1 مشكلة السباق "Race condition"

تقع هذه المشكلة حينما تكون عملية جدولة عمليتين حرجة، بحيث تنتج نتائج مختلفة تبعاً لأوامر متفرقة من الجدولة. وقد تنتج حالة السباق حينما تشترك عمليتان أو أكثر ضمناً أو بشكل صريح بنفس المصدر . كأن يتم طلب طباعة سطر على نفس الطابع في نفس الوقت، كما هو موضح في الشكل التالي :



الشكل (3 - 1)

طلب الطباعة من قبل عمليتين

افترضنا في الشكل (3 - 1)
الموضح جانباً وجود
العمليتين 1 ، 2 ، وان كلا
منهما تقوم بطلب طباعة
سطر على نفس الطابع
"Printer" ونتيجة لحدوث
الطلب في نفس الوقت ، فإننا
نجد ان ما تمت طباعته على
نفس السطر خليط مبعضر من
العمليتين .

وان أحد الحلول المقترحة لهذه المشكلة هو الربط المسبق للآلة الطابعة قبل
حدوث الطلب من إحدى العمليتين ، وبالتالي إذا تم ربط العملية (1) مسبقاً
بالآلة الطابعة مثلاً ، فإنه عند طلب الطباعة من العمليتين في نفس الوقت،
تجدول العملية (2) بعد العملية (1) ، وبذلك تنتهي حالة السباق، وتكون
نتائج كل عملية مستقلة عن الأخرى .

أي ان ما إستعملناه سابقاً يقع تحت عنوان الطلب وفك الارتباط "Request
and release" بمعنى أنه قد تم طلب الطابع وربطه بالعملية الاولى حتى
إكتمال طباعة ما تم الطلب إليه ، وبعد ذلك يحرر الطابع من ذلك الطلب،
ويوجه لتنفيذ الطلب الناشئ من العملية الثانية وهكذا . إن ميكانيكية إنشاء
الطلب والتحرر "Request and release" تقع ضمن واجبات برنامج ضابط
التحكم "Traffic controller" .

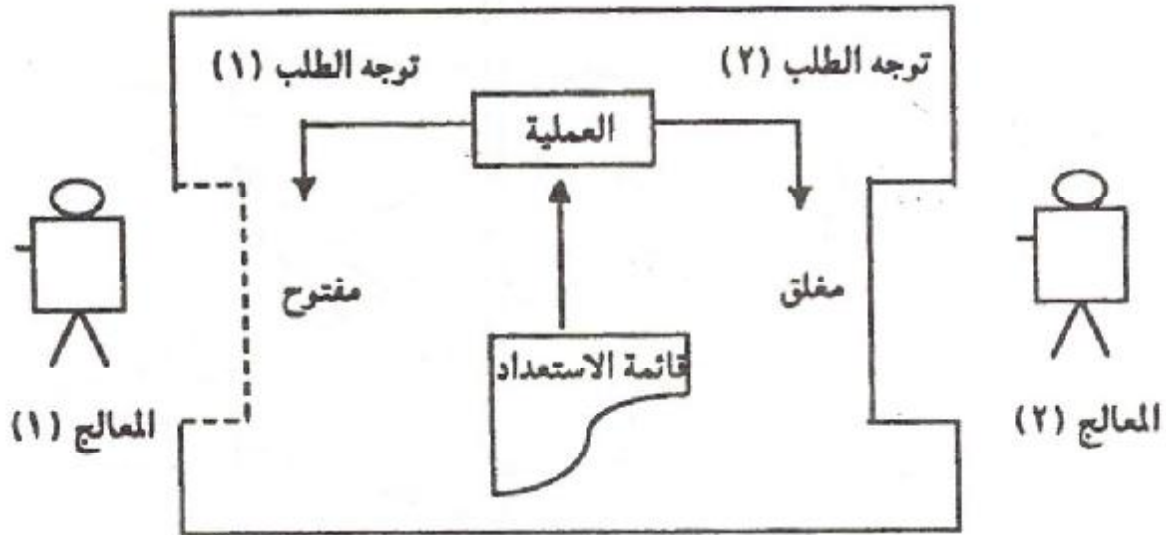
أما الحل الثاني لحالة السباق ، فيتم عن طريق نظم التمرير "Spooling
system" .

بالاضافة الى ذلك فإن حلولاً غير برمجية "Hardware solution" قد تم
وضعها لمعالجة حالة السباق ، ومن أمثلة ذلك ما يسمى بالمعالجات ذات
الاقفال "Multiple processors with lockout" .

فعند نشوء طلب لمعالج من عمليتين في نفس الوقت، يتم ربط ذلك المعالج بإحدى العمليتين فقط، وقفل المعالج الآخر .

أي أن قناة واحدة من الاتصال في الوقت الواحد يتم فتحها وجميع قنوات الاتصال في تلك اللحظة تكون مغلقة .

ويوضح الشكل (3 - 2) الفكرة السابقة :



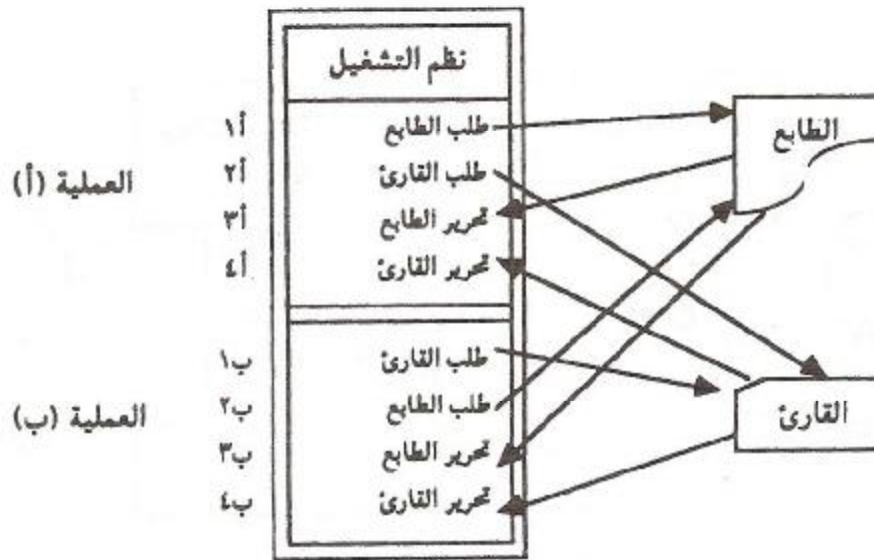
يتم ربط المعالج (1) فقط لأن باب القفل مفتوح، وبذلك ينفذ التوجه بالطلب (1) ، ولا يحدث التوجه بالطلب (2) لأن باب القفل مغلق من جهة المعالج (2) .

(Multiple processors with lockout)

الشكل (3 - 2) : عملية الربط باستخدام الاقفال

3 - 6 - 2 مشكلة العناق المميت (Deadly embrace الوضع الهامد)

وهي حالة إنتظار دائم تقع من قبل عمليتين تنتظران مصادر الحاسوب كالقارئ أو الطابع، أو محرك الأقراص، ... المرتبطة أصلاً مع كل منهما بمجموعة طلبات سابقة، وإليك المثال التالي الذي يوضح حالة الانتظار الدائم كما هو مبين في الشكل (3 - 2) .



الشكل (3 - 3) : حالة العناق المميت

يمكن جدولة خطوات كل من العمليتين (أ ، ب) على أشكال متعددة منها :

- 1 - 1A ، 2A ، 3A ، 4A ، 1B ، 2B ، 3B ، 4B على الترتيب .
- 2 - 1B ، 2B ، 3B ، 4B ، 1A ، 2A ، 3A ، 4A على الترتيب .
- 3 - 1A ، 2A ، 1B ، 3A ، 4A ، 2B ، 3B ، 4B على الترتيب .

يلاحظ القارئ أن الجدولة في كل مرة من 1 ، 2 ، 3 صحيحة ولا تحدث حالة إنتظار دائم مطلقاً ، وقد لاحظ القارئ حالة الإنتظار المؤقتة، التي

حدثت في الجدولة رقم 3 ، حيث أن طلب القارئ من العملية (ب ، ب3) وضع في حالة إنتظار مؤقت ، ريثما يتحرر بالطلب (41) ، وبذلك تصبح الجدولة رقم 3 صحيحة ، ولا تحدث حالة الإعاقة التي توقف عمليات الحاسوب .

لنفترض الآن أن الجدولة قد باتت بالعملية (11) : وهي طلب الطابع، وتبعتها العملية (ب1) طلب القارئ، فإذا تبع تلك العملية (21) طلب القارئ ، فإن (21) عليه أن ينتظر (ب1) حتى تنتهي من عملية القراءة وتصدر العملية (ب4) وهي تحرير القارئ . فإذا تبع ذلك (ب2) طلب الطابع ، فإن عملية إنتظار أخرى للعملية (ب2) ستحدث ، وذلك حتى تنتهي العملية (11) ، وتصدر العملية (31) تحرير الطابع، وبذلك إن العملية (1) تنتظر (ب) وأن العملية (ب) تنتظر العملية (1) ، لذلك كلاً من العمليتين لن تنفذ ، وهذا ما يسمى حقاً بالعناق المميت .

ولشكة العناق المميت، توجد الحلول التالية :

1 - الربط المسبق لمصادر الحاسوب قبل وقوع حالة العناق المميت، بحيث يتم جدولة العمليات المختلفة حسب حاجتها من مصدر الحاسوب، وشرط تجنب حالة العناق المميت، ولهذا الحل العيوب التالية :

أ - يحدث للمستخدم أن لا يعلم أوقات وفترات إرتباط كل مصدر من مصادر الحاسوب بالعمليات المختلفة، وذلك قبل بدء التنفيذ، وبالتالي يصعب تحديد عملية الربط المسبق لمصادر الحاسوب .

ب - عند ربط مصدر الحاسوب بعمل ما من الأعمال مسبقاً، قد يتم حجز هذا المصدر فترة طويلة دون أن يكون لهذا المصدر أي عمل وظيفي يؤديه. وبذلك يؤخر تنفيذ الأعمال الأخرى .

ج - من العبث أن يتم ربط مصدر من مصادر الحاسوب مسبقاً، خاصة إذا صغر احتمال إستخدام ذلك المصدر بذلك العمل المرتبط به، كأن يتم ربط الطابع مثلاً مسبقاً بعمل قد لا يحتاج أكثر من سطر واحد طباعة يتم طباعته في نهاية العمل .

2 - ربط المصادر بالأعمال تحت شروط Constrained allocation ، ويتفرع عن

هذا الربط ما يلي :

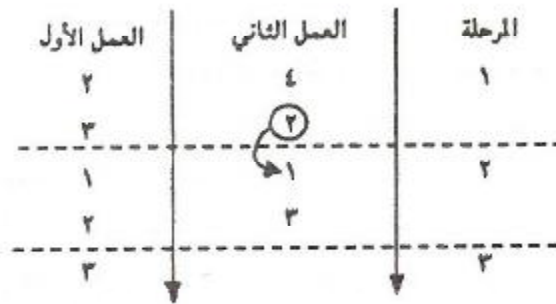
1 - الربط القياسي "Standard allocation" .

ب - الربط المتحكم به "Controlled allocation" .

أي لا يتم ربط المصادر بالأعمال إلا وفق قيود محددة، تمنع حالة العناق المميت. كأن يكون مثلاً لدينا عدداً محدوداً من الأشرطة الممغنطة، ويكون عدد الطلب على الأشرطة الممغنطة في وقت ما أكبر من العدد المتوفر لدينا، عندها نشترط برمجياً «أي ضمن برنامج التحكم بالمصادر» عدم الإستجابة لأمر الطلب الذي يتضمن عدداً من مصادر الحاسوب أكبر من العدد المتوفر لدينا .

فمثلاً إذا كان لدينا 9 أشرطة ممغنطة ولدينا العملاق 1 ، 2 بحيث يطلب العمل الأول على الترتيب 2 ، 3 ، 1 ، 2 ، 3 ، أشرطة ممغنطة .

أما العمل الثاني فكان يطلب على الترتيب : 4 ، 2 ، 1 ، 3 أشرطة ممغنطة وكلا العملاق كانا في الذاكرة الأساسية في نفس الوقت "Multiprogramming" وعلى فرض أن العاملين كان يسير كل منهما كما هو موضح في الشكل (3 - 4) .



الشكل (3 - 4) : رفض الطلب للعمل الثاني

ففي المرحلة الأولى ، يجب أن يتوقف الحاسوب عن إستجابة الطلب للعمل الثاني وذلك حين طلب شريطين ممغنتين ، حيث يبلغ في تلك اللحظة عدد الأشرطة الممغنطة المطلوبة 11 شريطاً ، وهو عدد يزيد عما هو متوفر لدى الحاسوب بشريطين، لذا إقتضت هذه المرحلة، توقف الحاسوب عن إستجابة الطلب، حتى يتم تحرير عدداً من الأشرطة أكبر أو يساوي العدد المطلوب .

لاحظ أن عدد الأشرطة المرتبطة بالعملين في المرحلتين الأولى والثانية يساوي 9 أشرطة، حيث تم نقل الطلب على الشريطين من قبل العمل الثاني في المرحلة الأولى الى المرحلة الثانية (لاحظ السهم) .

بينما نرى أن عدد الأشرطة في المرحلة الثالثة يساوي 3 أشرطة . وهذا ما يسمى بالربط المتحكم به "Controlled allocation" .

أما الربط القياسي "Standard allocation" فيعني وضع الأعمال في جدول مرتبة حسب تسلسل أرقام مصادر الحاسوب التي تدل على أولويات الربط.

3 - التحقق من حدوث حالة العناق المميت وحل تلك الحالة لحظة حدوثها "Detect and recover"

وتعني هذه الطريقة ترك العمليات تحت التنفيذ، ومراقبة حالة العناق المميت، حتى إذا وقعت تم معالجة تلك الحالة في حينه، ويلزم لذلك جدولان هما :

1 - جدول تعيين المصادر "Resource assignment table"

وظيفة هذا الجدول تحديد رقم المصدر ورقم العملية المرتبطة بذلك المصدر، فهو إذن بعمودين على النحو المبين في الشكل (3 - 5) :

Resource number	Assigned process

RAT

Resource Assignment table

الشكل (3 - 5) : جدول تعيين لمصادر

ب - جدول العمليات المنتظرة "Process wait table" .

وظيفة هذا الجدول تحديد أرقام العمليات الواقعة في حالة إنتظار وأرقام المصادر المرتبطة بتلك العمليات المنتظرة ، وهو على النحو المبين في الشكل (3 - 6)

process no	Resource no

pwt : Precess wait table

الشكل (3 - 6) : جدول العمليات

يتعاون كلا الجدولان المبينان في الشكل (3 - 5) و (3 - 6) على تحديد حالة العناق المميت، وبالتالي يقوم برنامج مجدول المعالجات بحل تلك الحالة حسب خوارزمية محددة لا مجال لذكرها هنا .

3 - 7 : ميكانيكية التزامن Synchronzation Mechanisms

تتوفر مجموعة من أنظمة التزامن متخصصة في معالجة العمليات من توجيه واتصال، وفي هذا البند سوف نقوم بعرض لبعض هذه الأنظمة .

4 - 7 - 1 : التعليمة Test and set

في معظم أنظمة التزامن تعتبر المصادر المستخدمة هي المدخل الرئيسي حيث يسمى المصدر بالسيمافور (Semaphor) ، ولهذا فإن لكل مصدر من المصادر كوحدة الإدخال والإخراج المشتركة أو قاعدة البيانات يتوفر سيمافور خاص به، فإذا كانت قيمته مساوية للصفر، فإن المصدر يعتبر

متوفرًا، ويمكن حجزه لتنفيذ عملية معينة، أما إذا كانت قيمته مساوية للواحد، فإن هذا يعني أن المصدر محجوز من قبل عملية أخرى .

وقبل البدء باستخدام المصدر المشترك فإن العمل يجب أن يقوم بما يلي :

1 - فحص قيمة السيمافور المخصصة للوحدة أو المصدر المطلوب (هل هو صفر أو واحد) ؟

2 - جعل قيمة السيمافور مساوية للواحد .

3 - إذا كانت القيمة الأصلية للسيمافور مساوية للواحد، نفذ الخطوة 1، وبعد انتهاء العمل من استخدام المصدر، فإن العمل يقوم بجعل قيمة السيمافور الخاصة بالمصدر مساوية للصفر .

وفي بعض الأحيان يطلق على السيمافور اسم بايت الإغلاق Lock byte ، فإذا أطلقنا على هذا البايث الاسم X فإن العملية المنفذة على هذا البايث هي عملية إغلاق Lock (X) أما العملية بعد انتهاء العمل من استخدام المصدر ، فيشار إليها بـ Unlock (X) ويمكن التعبير عن هذه العمليات بـ Request (X) في حالة طلب استخدام المصدر و Release (X) في حالة الانتهاء من استخدام المصدر. ويجب الانتباه هنا أنه في حالة فحص هذا البايث من قبل عمل معين فإن قيمة هذا البايث لا يمكن تغييرها من قبل عمل آخر وذلك بين الخطوة الأولى والثانية .

3 - 7 - 2 : Wait and signal

في الطريقة الأولى في التعامل مع السيمافور يستهلك زمن كبير من وحدة المعالجة المركزية في الفحص وتغيير قيمة السيمافور ، لذا يمكن استخدام مبدأ الانتظار ثم إصدار الإشارة (Wait and signal) ويتلخص هذا المبدأ فيما يلي :

A. Lock (X)

1. examine the value of X (either 0 or 1)

2. set X to 1
3. if the original value of X = 1, call wait (X)

B. Unlock (X)

1. set X to 1
2. call signal (X)

وتعتبر العمليات wait و signal من مهام مشرف المر في إدارة وحدة المعالجة المركزية حيث تقوم العملية wait (X) بربط العمل الموقوف (blocked) مع X ، وبعدها يتم اختيار عمل آخر للتنفيذ من قبل مجلد وحدة المعالجة (مجدول العمليات) . أما العملية signal (X) فتقوم بفحص الاعمال الموقوفة والمرتبطة بإيقافها بمصدر معين بايت الإغلاق له هو X ، وإذا كان هناك أعمال من هذا النوع فإن أحد هذه الاعمال سوف يتم اختياره وتغير حالته من حالة الإيقاف الى حالة الاستعداد .

3 - 7 - 3 : العمليات V, P

من العمليات الأساسية والمطبقة على السيمافور المحسوب هي العملية V, P .

فالعملية P تتلخص في الخطوات التالية :

P(S)

1. Decrement value of S ($S = S - 1$)
2. if S is less than 0, wait (S)

أما العملية V فتتخلص في الخطوات التالية :

V(S)

1. Increment value of S ($S = S + 1$)
2. if S is less than or equal 0, signal (X)

واعتماداً على قيمة السيمافور S الابتدائية وعلى عدد S المستخدم فإن العمليات V, P تستخدم لأغراض متعددة .

ومن أهم استخداماتها هي نقل وتبادل المعلومات (Communication) حيث تتم عملية التبادل بين عمليين ، وذلك عن طريق استخدام ذاكرة مؤقتة (Buffer) وسيمافور . ومن الأمثلة على هذه الحالة عملية البيع والشراء الموضحة في الجدول التالي (لاحظ أن المنتج لا يمكنه إضافة مادة في حالة كون الذاكرة ممتلئة، أما المشتري فلا يمكنه الأخذ من الذاكرة في حالة كونها فارغة) .

Event	ACTIONS		SEMAPHORE VALUES	
	P_p (Producer)	P_c (Consumer)	S_1	S_2
0	...	...	2	0
1	—	$P(S_2)$ blocked	2	1
2	$P(S_1)$	— (buffer	1	1
3	$V(S_2)$ unblocks P_c	— "empty")	1	0
4	$P(S_1)$	—	0	0
5	—	$V(S_1)$	1	0
6	$V(S_2)$	—	1	1
7	$P(S_1)$	—	0	1
8	$V(S_2)$	—	0	2
9	$P(S_1)$ blocked	—	-1	2
10	— (buffer	$P(S_2)$	1	1
11	— "full")	$V(S_1)$ unblocks P_p	0	1
etc.				

إن عملية الاتصال المباشر بين الأعمال تتم من خلال التعليمات $SEND(PR,M)$ و $RECEIVE(Ps,M)$ ، حيث تشير Ps ، Pr الى أسماء الأعمال ، أما M ، فتشير الى رسالة المعلومات المراد إرسالها من العمل الثاني .

الوحدة الرابعة

إدارة الإدخال والإخراج وإدارة وحدات الإدخال والإخراج I\ O Managment

- إدارة عمليات الإدخال والإخراج :
 - مفهوم الاعتراض .
 - قنوات الإدخال والإخراج .
- إدارة وحدات الإدخال والإخراج :
 - طرق استخدام وحدات الإدخال والإخراج .
 - خصائص وحدات الإدخال والإخراج .
 - وحدات التحكم والقنوات .
 - مكونات برمجيات إدارة وحدات الإدخال والإخراج .
 - الوحدات الافتراضية .
 - النظام الافتراضي ونظام التمرير .

إدارة الإدخال والإخراج

سوف نتناول في هذه الوحدة كيفية إدارة عمليات الإدخال والإخراج (I/O operations) وإدارة وتنظيم الوحدات الفيزيائية المخصصة لعمليات الإدخال والإخراج (I/O Devices mangment) .

4 - 1 : إدارة عمليات الإدخال والإخراج

يقصد بعمليات الإدخال والإخراج تبادل المعلومات بين وحدات الإدخال والإخراج ومسجلات وحدة المعالجة المركزية أو بين وحدات الإدخال والإخراج والذاكرة الرئيسية .

وتتوفر عدة طرق لتنفيذ عمليات الإدخال والإخراج أهمها :

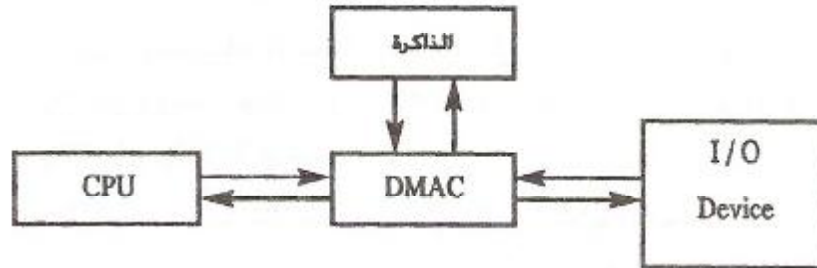
1 - استخدام طريقة المسح، حيث تزود كل وحدة ادخال / اخراج بمفتاح (Flag) يأخذ القيمة صفر أو واحد وعند حاجة هذه الوحدة للارتباط بالمعالج تعمل وحدة الإدخال / الإخراج على وضع 1 في المفتاح وتقوم وحدة المعالجة المركزية بين الفترة والآخرى باستعراض قيم هذه المفاتيح حتى ترتبط بوحدة الإدخال والإخراج التي تحتاج الى المعالج وذلك عندما تكون قيمة المفتاح لهذه الوحدة مساوياً للواحد .

2 - الإدخال والإخراج المبرمج وتتم هذه العملية من خلال استخدام تعليمات مباشرة للإدخال والإخراج يحدد فيها منافذ الإدخال والإخراج والمراد استخدامها وعناوين وحدات الإدخال المعنية في عملية الإدخال والإخراج .

3 - الإدخال والإخراج باستخدام مفهوم الاعتراض (Interrupt) وسوف نستعرض مفهوم الاعتراض لاحقاً في هذا البند .

4 - الإدخال والإخراج المباشر للذاكرة (Direct memory access : DMA) .

تستخدم هذه الطريقة لتحقيق مبدأ المعالجة على التوازي بحيث يقوم المعالج بتنفيذ العمليات الحسابية والمنطقية وفي نفس الوقت تنفذ عمليات الإدخال والإخراج بين وحدات الإدخال والإخراج والذاكرة الرئيسية . ولتنفيذ عمليات الإدخال والإخراج بهذه الطريقة لا بد من توفر وحدة خاصة تسمى متحكم الوصول المباشر للذاكرة (DMA controller) يتم ربطها مع وحدة الإدخال والإخراج والذاكرة الرئيسية ووحدة المعالجة المركزية كما هو مبين في الشكل (4 - 1) .



الشكل (4 - 1) : استخدام DMAC

يحتوي متحكم الوصول المباشر للذاكرة على بعض المكونات (أهمها مسجل العنوان ، عداد ، مسجل بيانات ، وحدة تحكم محلية) وتتم عملية الإدخال والإخراج بواسطة المتحكم كما يلي :

- 1- تعمل وحدة المعالجة على تحديد عنوان البداية في الذاكرة والمخصص لعمليات الإدخال والإخراج وذلك بوضع هذه القيمة في مسجل العنوان في DMAC .
- 2 - تحدد وحدة المعالجة المركزية عدد البايت المراد تبادلها بين الذاكرة ووحدة الإدخال / الإخراج وبعد هذا تقوم وحدة المعالجة بالانتقال الى تنفيذ العمليات الحسابية والمنطقية المخزنة في مسجلاتها وتتخل عن خطوط نقل العنوان وخطوط نقل البيانات .

- 3 - يعمل DMAC على استخدام خطوط العنوان وخطوط البيانات لنقل البايت الذي عنوانه مخزن في مسجل العنوان من الذاكرة الى وحدة الإخراج مروراً بمسجل البيانات أو بالعكس من وحدة الإدخال الى موقع الذاكرة المحدد .
- 4 - بعد نقل البايت الاول تعمل وحدة التحكم المحلية على طرح واحد من قيمة العداد وزيادة واحد الى مسجل العنوان للانتقال الى عملية النقل التالية .
- 5 - تكرر الخطوات 3 و 4 ما دامت قيمة العداد لا تساوي الصفر .
- 6 - عندما تصبح قيمة العداد مساوية للصفر يعترض DMAC وحدة المعالجة المركزية مخبراً اياها بانتهاء عملية الإدخال / الإخراج .

4 - 1 - 1 : مفهوم الاعتراض واستخدامه في عمليات الإدخال والإخراج

الاعتراض هو حدث استثنائي يضطر نظم لتشغيل عند الاستجابة له وقف تنفيذ العمل الخاضع للتنفيذ من أجل تنفيذ عمل جديد يسمى العمل المعترض . وتمتلك نظم التشغيل برمجيات خاصة لمعالجة الاعتراض (المقاطعة) تسمى بمعالج الاعتراض (Interrupt handler) وتتولى هذه البرمجيات تنفيذ الوظائف التالية :

- 1 - اصدار الاستجابة الموجبة أو السالبة بناء على طلب الاعتراض من وحدة الإدخال والإخراج .
- ويتحدد نوع الاستجابة بناء على نوع الاعتراض فإذا كان الاعتراض غير قابل للتأجيل (Non maskable) فإن الاستجابة سوف تكون موجبة (أي انه يمكن الانتقال الى خدمة الاعتراض) أما اذا كان الاعتراض قابلاً للتأجيل (maskable) فيمكن تأجيل خدمة الاعتراض باصدار استجابة سالبة .
- 2 - في حالة تقبل طلب الاعتراض تعمل برمجيات الاعتراض على حفظ حالة البرنامج (العمل) الخاضع للتنفيذ (ولنسمه العمل القديم) وذلك للرجوع اليها لاحقاً، ويتم تحميل حالة العمل المعترض الجديد بعد وقف تنفيذ العمل القديم .

3 - ينفذ العمل الجديد .

4 - بعد الانتهاء من العمل الجديد تحمل حالة البرنامج القديم لمتابعة تنفيذه من النقطة التي قُطع منها .

ويقصد بحالة البرنامج (Program Status) المراد حفظها لغاية الاسترجاع اللاحق هو قيمة عداد البرنامج والذي يحتفظ بعنوان التعليمة التالية (لا يقبل الاعتراض إلا بعد الانتهاء من تنفيذ التعليمة الحالية) ، ومجموعة من الرايات (Flags) تسمى بمسجل الرايات بحيث تشير كل خلية فيه الى حالة معينة لنتيجة التنفيذ ومن هذه الخانات :

- خانة الصفر (Zero Flag) وتكون قيمتها مساوية للواحد عندما يكون ناتج تنفيذ التعليمة الحالية صفراً .

- خانة الحمل (Carry Flag) وتكون قيمتها مساوية للواحد عندما يحدث حمل بعد تنفيذ التعليمة .

- خانة الاشارة (Sign Flag) وتكون مساوية للواحد عندما تكون نتيجة التنفيذ سالبة .

- خانة الفائض الحسابي (Over Flow Flag) وتكون قيمتها واحد عند حدوث فائض حسابي نتيجة لتنفيذ التعليمة .

- خانة الاعتراض (Interrupt Flag) وتحدد هذه الخانة امكانية الاستجابة للاعتراض بعد تنفيذ التعليمة الحالية فإذا كانت قيمة الخانة مساوية للصفر فهذا يعني امكانية الاستجابة للاعتراض وهناك خانات أخرى ليست موضوعنا الآن .

يستخدم مسجل الحالة (مسجل الرايات) في أجهزة الحاسوب الشخصية وفي بعض الأجهزة وخاصة الأجهزة المتوسطة والكبيرة والتي تعمل تحت أنظمة تشغيل أخرى تخزن حالة البرنامج في مسجل خاص يسمى مسجل حالة البرنامج (Processor status word : PSW) ويحتوي هذا المسجل على الحقول التالية :

- حقل تغطية النظام ويشير محتوى هذا الحقل الى امكانية استجابة الحاسوب للاعتراضات التي قد تحدث اثناء تنفيذ البرنامج .

- حقل تغطية البرنامج ويشير محتوى هذا الحقل الى امكانية الاستجابة لانواع اخرى من الاعتراضات قد تحدث اثناء تنفيذ البرنامج .

- حقل طول التعليم الحالية ويشير هذا الحقل الى طول التعليم قيد التنفيذ ومن هذا الحقل يمكن تحديد عنوان التعليم في البرنامج والتي يجب ان تنفذ لاحقاً .

اضافة الى بعض الحقول الاخرى والتي قد تشير الى توصيف التعليم وتحدد نوعها .

تقسم الاعتراضات من حيث آلية وميكانيكية تنفيذها إلا نوعين :

1 - الاعتراض المبرمج (Programmed interrupt) .

ينفذ هذا النوع من الاعتراضات باستخدام تعليمات مباشرة للاعتراض تكون جزءاً من البرنامج وذلك بعد تحديد وظيفة الاعتراض المراد تنفيذها فمثلاً باستخدام نظام التشغيل DOS فإن :

Mov AH, 0AH

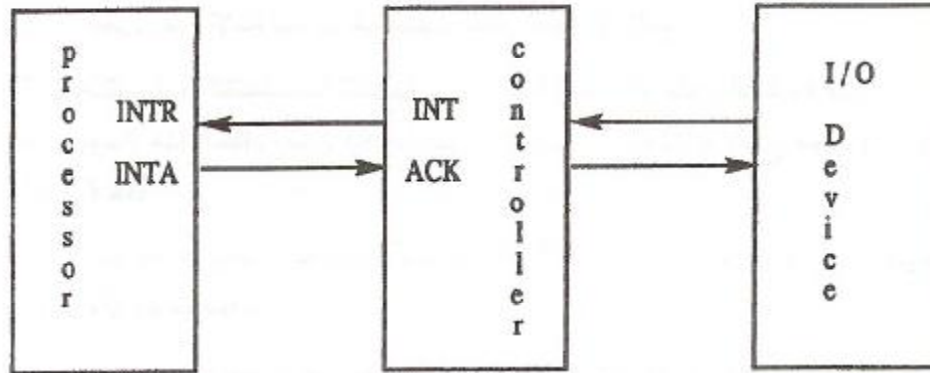
LEA DX, buffer

INT 21H

وجود التعليمات (اسميلي) السابقة في برنامج ما يؤدي الى وقف التنفيذ عند مصادفة تعليمية الاعتراض وبناء على رقم الاعتراض والوظيفة المحددة في المسجل AH يتم احتساب عنوان برنامج خدمة الاعتراض حيث يتم الانتقال الى تعليمات خدمة الاعتراض وذلك بتنفيذها لربط لوحة المفاتيح مع الذاكرة (buffer) لاجراء عملية ادخال لمجموعة من الرموز عن طريق لوحة المفاتيح محدد عددها .

2 - اعتراض المعدات (Hardware interrupt) ، في هذا النوع من الاعتراضات ترتبط

وحدة الإدخال / الإخراج مع متحكم خاص يرتبط بالمعالج كما هو مبين في الشكل (4 - 2) .



الشكل (4 - 2) : اعتراض المعدات

فعند الحاجة لتنفيذ عملية الإدخال / الإخراج يعمل المتحكم على تفعيل خط الاعتراض (INT) مشعراً بذلك المعالج بطلب الاعتراض ، وبعد استجابة المعالج لطلب الاعتراض يعمل المعالج على تفعيل خط الاستجابة للاعتراض INTA مشعراً بذلك المتحكم بالاستجابة لطلب الاعتراض (تسمى هذه الطريقة في الإدخال والإخراج بالمصافحة اليدوية handshaking) بعد هذا يقرأ المعالج محتوى مسجل عنوان الاعتراض في المتحكم ثم يعمل المعالج على توليد تعليمة خاصة (Restart) من خلالها يحدد مستوى الاعتراض ويحدد بداية العنوان في برنامج خدمة الاعتراض والمطلوب الانتقال اليه .

أنواع الإعتراضات : "Interrupt types"

تصنف الاعتراضات حسب المصدر القادمة منه إلى :

1 - إعتراض البرنامج "Program interrupt" .

ويحدث لواحد من الاسباب التالية :

أ - خطأ في أمر المعالج : "Invalid CPU instruction" .

- وقد يكون هذا الخطأ لغوياً "Lexical error" أو خطأ قواعدياً "Syntax error" أو خطأ في الشيفرة الثنائية "coding error" .

ب - خطأ الفائض الحسابي "Overflow error" .

من المعلوم أن الأرقام وكافة البيانات الرقمية في الحاسوب تمثل بواسطة الشيفرة الثنائية "Binary codes" ، بحيث لا يزيد عدد الخلايا الثنائية عن طول الكلمة الخاصة بالحاسوب، فإذا كان طول الكلمة في الحاسوب غير كاف لتمثيل بيانات رقمية ، وفاضت عدد خانات تمثيل الرقم عن طول الكلمة اعتبر ذلك فائض حسابي .

فمثلاً إذا كان طول الكلمة الخاصة بتمثيل الأرقام في الحاسوب يساوي 16 خلية ثنائية (16 bits) فإن أكبر عدد صحيح موجب يمكن أن يكون ممثلاً في هذه الكلمة يساوي $2^{15} - 1 = 32767$. وهذا الرقم يمثل كما يلي :

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0															15

وذلك بإهمال خلية الإشارة + ، - ، .

فإذا أردنا تمثيل الرقم 32768 يصبح تمثيل الرقم كما يلي :

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0															15

↑
فائض حسابي

مما يدل على حدوث فائض حسابي .

فإذا حدث مثل هذا الفائض الحسابي ، حدث إعتراض مفاجيء يضطر البرنامج للتوقف .

ج - خطأ إختراق الحماية "Storage protection violation" ، يحدث هذا الخطأ عندما يطلب من برنامج في إحدى تعليماته أن يخزن بيانات أو معلومات في الذاكرة الأساسية أو المساندة، وذلك في موقع غير مسموح الكتابة فيه .
أو أن يطلب البرنامج في إحدى تعليماته إستدعاء بيانات «قراءة بيانات» من موقع في الذاكرة الأساسية أو المساندة من غير الممكن الوصول إليها قراءة .
عند ذلك يحدث إعتراض إختراق الحماية، سواءً كانت حماية بالكتابة فقط أو حماية بالقراءة أو كليهما .

2 - إعتراض الإدخال والإخراج NO interrupt

وينشأ هذا الإعتراض في قناة الإدخال أو الإخراج، أو وحدات الإدخال والإخراج، وأسبابه واحد مما يلي أو أكثر :

1 - خطأ في أمر الإدخال أو الإخراج ، كان يكون الأمر غير صحيح لغوياً، أو غير متبع لقواعد كتابة أوامر الإدخال أو الإخراج، أو خطأ في شيفرته الثنائية، أو عدم تحديد هذا الأمر كلا من رقمي وحدة الإدخال والإخراج ورقم القناة .

ب - إنتهاء تنفيذ برنامج الإدخال أو الإخراج، وذلك بعد تنفيذ كلمة "Channel command word" في برنامج القناة، حيث يصدر نظم التشغيل الأمر المهيمن HIO لإيقاف برنامج القناة وبذلك يحدث إعتراض مفاجيء على القناة لتتوقف وتعمل أجزاء أخرى في الحاسوب .

ج - إنتهاء وحدتي الإدخال أو الإخراج من عملية تمرير ونقل البيانات، وفي هذه الحالة يتحكم برنامج ضابط التحكم بالإدخال والإخراج من ضبط عملية نقل البيانات بالإتجاهات المختلفة وذلك بالتعاون مع نظام التمرير Spooling system فإذا تمت عملية تمرير البيانات حدث إعتراض مفاجيء .

3 - إعتراض نظام التشغيل "Supervisor call interrupt" .

حيث يقوم نظام التشغيل الخاص بالحاسوب بطلب التوقف المفاجيء ، وذلك لإجراء عمليات داخلية في الحاسوب، لا علاقة للمستخدم بها، فيحدث ذلك الإعتراض .

4 - الإعتراض الخارجي "External interrupt" .

وسبب هذا الاعتراض أحد أمرين :

1 - المستخدم : إذا يقوم مستخدم بتوقيف البرنامج الواقع تحت التنفيذ بالخروج منه إضطرارياً، وذلك حينما يرى المستخدم ضرورة لذلك، كإيقاف تنفيذ البرنامج لدورات غير منتهية (Endless Loop) ، أو إيقاف تنفيذ البرنامج لإكتشاف المستخدم خطأ ما في عمل البرنامج، أو إيقاف تنفيذ البرنامج لإنهاء الفترة الزمنية التي سجلت بإسم المستخدم أو ما أشبه ذلك من أسباب .

أو قد يضطر المستخدم بتوقيف البرنامج عن التنفيذ لحظياً، وذلك لاداء مهمات أخرى تتعلق بالبرنامج، كأن يضبط المستخدم تجهيزات الآلة الطابعة وما يعلق فيها من أوراق طباعة، أو أن يجهز المستخدم وسائط تخزين معينة لتخزين أو شحن برنامج ما أو قطعة من برنامج .

كل ذلك يؤدي إلى إحداث إعتراض خارجي .

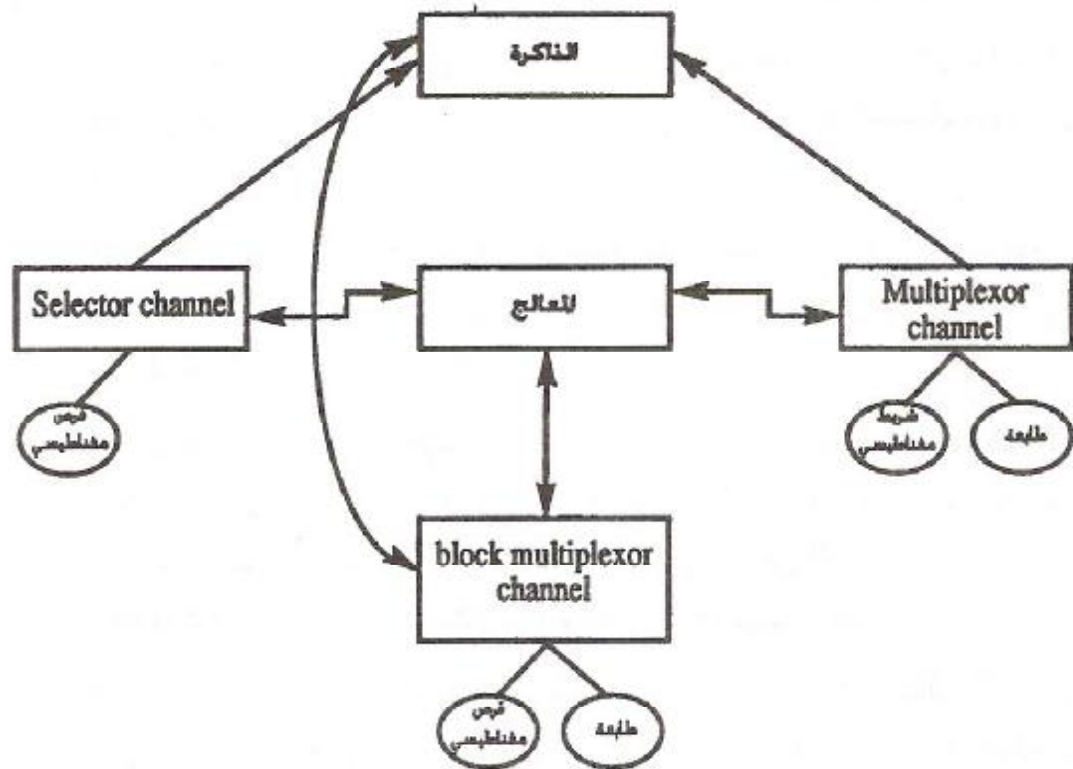
ب - المؤقتات الداخلة في تركيب الحاسوب، كإنقضاء الفترة الزمنية لتنفيذ البرنامج وخاصة في نظام المشاركة بالزمن "Time-sharing" ، أو حدوث خلل في المؤقت نفسه، أو إنعكاس القيمة الموجبة في المؤقت الى قيمة سالبة . وقد يحدث هذا النوع من الإعتراضات بسبب إتصال مكونات الحاسوب بعضها ببعض، إتصال المعالج بالمعالج، وإتصال الذاكرة بالقنوات أو أي إتصال آخر .

5 . إعتراض إختبار الآلة "Machine check interrupt" .

عند بداية تشغيل الحاسوب يقوم نظام التشغيل بإستعراض وفحص المعدات المتصلة بالحاسوب تلقائياً، بحيث لا يمكن أن يؤدي الحاسوب وظيفة ما دون الإنتهاء من هذا الإختبار ويمكن للقارئ أن يرى ذلك الإختبار في أجهزة الحواسيب المصغرة عند تشغيلها، حيث يتوفر في نظام التشغيل الخاص بهذه الأجهزة برنامج فحص ذاتي يعمل على التأكد من سلامة اتصال كافة معدات الحاسوب ببعضها ويعطي المستخدم رسالة توضيحية (في حالة وجود خلل) تفيد بوجود خطأ في عملية ربط الأجهزة وقد يوقف هذا البرنامج عملية التشغيل في حالة وجود خطأ في الذاكرة أو في المعالج .

4 - 1 - 2 قنوات الإدخال والإخراج (I/O Channels) :

قنوات الإدخال والإخراج عبارة عن أجهزة متخصصة يمكن أن تكون حاسوباً، لكنها تتخصص في الإشراف على ربط وحدات الإدخال والإخراج مع المعالج أو مع الذاكرة لاتمام عمليات الإدخال والإخراج ، وتحتوي هذه الأجهزة (القنوات) على برنامج خاص يسمى برنامج القناة (Channels program) وظيفته متابعة عمليات الإدخال والإخراج والاتصال بين وحدة الإدخال / الإخراج المعنية والمعالج أو الذاكرة . ويبين الشكل (4 - 3) كيفية استخدام قنوات الإدخال والإخراج لأجراء عمليات الربط اللازمة .



الشكل (4 - 3) : استخدام القنوات .

واضح من الشكل (4 - 3) أن عملية الاتصال يمكن أن تتم بإحدى الطرق التالية :

- 1 - الاتصال المباشر بين الذاكرة والقناة .
- 2 - الاتصال المباشر بين المعالج والقناة .
- 3 - الاتصال المباشر بين وحدات الإدخال والإخراج والقناة .

واضح من الشكل أيضاً أن قنوات الإدخال والإخراج تصنف إلى :

- 1 - Selector : وهي قنوات سريعة تعمل على ربط وحدات الإدخال والإخراج السريعة مع المعالج أو مع الذاكرة الرئيسية وتستطيع هذه القنوات نقل كتلة (block) كاملة من البيانات دفعة واحدة .

2 - Multiplexor : وتربط هذه القنوات وحدات الإدخال والإخراج البطيئة مثل الطابعة والشريط المغناطيسي وتعمل على نقل البيانات من هذه الوحدات الى المعالج أو الذاكرة أو من الذاكرة أو المعالج الى وحدات الإدخال والإخراج البايت تلو البايت .

3 - Block multiplexor : وهي قنوات خاصة تتوفر فيها خصائص النوع الأول والثاني من القنوات وتستطيع ربط وحدات الإدخال والإخراج السريعة فيها والبطيئة .

تعمل قنوات الإدخال والإخراج على الاشراف على تنفيذ عمليات الإدخال والإخراج الى / من المعالج أو الى / من الذاكرة وذلك عن طريق برنامج خاص يخزن فيها ويتألف هذا البرنامج من مجموعة من التعليمات تأخذ على عاتقها تنفيذ كل ما يتعلق بالإدخال والإخراج من عمليات أهمها :

1 - نقل البيانات (data transferring) وذلك بقراءة البيانات من وحدات الإدخال وارسالها الى الذاكرة أو مسجلات وحدة المعالجة المركزية أو كتابة البيانات على وحدات الإخراج وذلك بنقلها من الذاكرة أو من مسجلات وحدة المعالجة المركزية الى وحدة الإخراج .

2 - التحكم بوحدات الإدخال والإخراج وذلك بتنفيذ التحريلات اللازمة للبيانات للتأثير على الوحدات الميكانيكية في وحدات الإدخال والإخراج من أجل تحريك هذه الوحدات كتحريك الشريط أو تحريك رأس القراءة والكتابة أو تحريك رأس الطابعة وغيرها من عمليات ميكانيكية .

3 - نقل التحكم وتنفيذ عمليات التفرع اللازمة إذا تطلب الإدخال والإخراج هذا . وحتى تقوم قناة الإدخال والإخراج بأداء وظائفها ومهامها لا بد من توفر برنامج القناة (Channel program) والذي يتألف من التعليمات التالية:

1 - مجموعة من تعليمات (CCW : Channel command word)

بحيث تتألف كل تعليمة من هذه التعليمات من مجموعة من الحقول أهمها :

- حقل شيفرة العملية وتستخدم لتحديد نوع تعليمة الإدخال والإخراج المراد تنفيذها .

- حقل عنوان البيانات ويشير هذا الحقل الى موقع البيانات (التابعة للعملية أو المرتبطة بها) في الذاكرة الرئيسية .

- حقل العداد والذي يشير الى طول حقل البيانات المرتبط بالتعليمة .

- مجموعة من الرايات (Flage) تشير الى امكانية استخدام مواقع متعددة في الذاكرة، امكانية وقف الإدخال والإخراج، توصيف البيانات .

وحتى تنفذ مجموعة التعليمات هذه لا بد من توفر مسجلين (تعليمتين) هما:

2 - مسجل حالة القناة (تعليمة) CSW : Channel status word

وتحدد هذه التعليمة حالة القناة (مشغولة / غير مشغولة) وتحتوي على الحقول التالية :

- حقل الحماية .

- حقل يشير الى موقع CCW التالية والمراد تنفيذها .

- حقل الحالة (القناة عاملة، غير عاملة، وجود خطأ في الإدخال والإخراج) .

3 - كلمة العنوان (CAW : Channel address word) وتحتوي هذه التعليمة على مجموعة من الخلايا الثنائية يمكن من خلالها تحديد عنوان البداية لبرنامج القناة (عنوان أول CCW في برنامج القناة) .

يتم التحكم بتنفيذ برنامج القناة (عملية الإدخال والإخراج) من خلال نظام التشغيل وذلك عن طريق مجموعة من التعليمات المهيمنة (Previliged instruction) تنفذ فقط من قبل نظم التشغيل ومن هذه التعليمات :

1 - بدء التنفيذ (START INPUT OUTPUT : SIO)

ويؤدي تنفيذ هذا الأمر الى :

1 - تحديد رقم (عنوان) قناة الإدخال والإخراج .

2 - تحديد رقم (عنوان) وحدة الإخراج والادخال .

3 - تحديد أول عنوان في برنامج القناة .

ب - الفحص والاختبار (TEST INPUT OUTPUT : ITO)

ويعمل هذا الأمر على فحص حقل نوع التعليم في مسجل حالة البرنامج (PSW) للتأكد من أنها عملية ادخال / اخراج .

ج - وقف تنفيذ برنامج القناة (Halt I/O : HIO)

ويعمل هذا الأمر على إيقاف تنفيذ برنامج القناة (وقف عملية الإدخال والإخراج) .

4 - 2 : إدارة وحدات الإدخال والإخراج

سوف نتناول في هذا البند إدارة وحدات الإدخال والإخراج والتي تتضمن الوحدات الحقيقية مثل آلة الطباعة ، الأشرطة المغناطيسية والأقراص المغناطيسية، وكذلك الوحدات الداعمة مثل: وحدة التحكم ، القنوات ، وحدات تحويل المعلومات .

هذا وتشرف على هذه الوحدات برمجيات خاصة تسمى ببرمجيات إدارة وحدات الإدخال والإخراج (Device management) حيث تأخذ هذه البرمجيات على عاتقها تنفيذ المهام والوظائف التالية :

1 - متابعة وحدات الإدخال والإخراج وذلك من خلال استخدام كتلة وحدة التحكم لكل وحدة (unit control block UCB) والتي تتضمن كافة المعلومات المطلوبة عن كل وحدة ادخال واخراج مرتبطة مع الكمبيوتر. هذا وسوف نستعرض هذا لاحقاً إن شاء الله في هذا الفصل .

2 - اتخاذ سياسة معينة لحجز وحدات الادخال أو الاخراج وفي زمن معين ولفترة زمنية محددة وغالباً ما تعتمد هذه السياسة على طريقة استخدام وحدة الإدخال والإخراج (وحدات غير مشتركة، مشتركة، افتراضية) .

3 - الحجز الفيزيائي لوحدة الإدخال أو الإخراج وربط هذه الوحدة أو تلك مع العمل القابل للتنفيذ .

4 - إعادة مجز وحدات الإدخال والإخراج المرتبطة مع العمل ، وذلك بعد انتهاء تنفيذ العمل أو بعد انتهاء الفترة الزمنية المخصصة للحجز .

4 - 2 - 1 : طرق استخدام وحدات الإدخال والإخراج :

تقسم وحدات الإدخال والإخراج حسب طريقة الاستعمال الى :

1 - الوحدات غير المشتركة Dedicated I/O Devices

2 - الوحدات المشتركة Shared I/O Devices

3 - الوحدات الافتراضية Virtual I/O Devices .

الوحدات غير المشتركة :

وهي وحدات يمكن حجزها لعمل واحد فقط، ولا يمكن اشتراك أكثر من عمل قابل للتنفيذ في استخدام هذه الوحدات ، من الأمثلة على هذه الوحدات : الآلات الطابعة، قارئ البطاقات المثقبة. وعند استخدام أي من هذه الوحدات من قبل عمل معين فإن الأعمال الأخرى القابلة للتنفيذ والتي قد تحتاجها الوحدة المحجوزة عليها الانتظار حتى تتحرر الوحدة المطلوبة عند انتهاء تنفيذ العمل أو عند انتهاء الفترة الزمنية المخصصة للعمل .

الوحدات المشتركة :

وهي وحدات يمكن استخدامها من قبل أكثر من عمل قابل للتنفيذ، وفي نفس الفترة الزمنية. ومن الأمثلة على هذه الوحدات : وحدات الإدخال والإخراج ذات الوصول المباشر (Direct Access Storage Devices DASD) كالقرص المغناطيسي مثلاً. وتعتبر إدارة مثل هذه الوحدات أصعب من إدارة الوحدات غير المشتركة نظراً للحاجة الى تحديد أولوية عملية القراءة أو الكتابة للأعمال المختلفة القابلة للتنفيذ .

الوحدات الافتراضية

وهي وحدات غير مشتركة تم تحويلها الى وحدات مشتركة، وذلك باستخدام تقنيات وبرمجيات خاصة كنظام التمرير (Spooling System) مثلاً حيث يؤدي هذا النظام الى تحويل الوحدة غير المشتركة الى وحدة مشتركة، فمثلاً يمكن استخدام القرص المغناطيسي كألة طباعة، مما يؤدي بدوره الى تحقيق الفوائد التالية :

- 1 - حل مشكلة عدد الوحدات غير المشتركة، فمثلاً لو كان لدينا عدة أعمال تحت التنفيذ وكلها تحتاج الى عمليات طباعة، فإنه لتنفيذ هذه الأعمال لا بد من توفر عدد من الطابعات مساوياً لعدد الأعمال المنفذة، مما يؤدي بدوره الى تكاليف باهظة ، وباستخدام نظام التمرير يمكن الاستعانة بالقرص المغناطيسي حيث تتم عملية الطباعة على القرص ، وعند الحاجة تنقل المعلومات من القرص المغناطيسي الى الطابعة .
- 2 - زيادة سرعة المعالجة نظراً لأن سرعة تعامل وحدة المعالجة المركزية مع القرص المغناطيسي أعلى منها عند تعامل الطابعة مع وحدة المعالجة المركزية. هذا وسوف نستعرض نظام التمرير لاحقاً في هذا الفصل إن شاء الله .

4 - 2 - 2 : خصائص وحدات الإدخال والإخراج :

تقسم وحدات الإدخال والإخراج الى قسمين :

- 1 - وحدات الإدخال والإخراج : وهي الوحدات المستخدمة في ادخال البيانات الى الكمبيوتر أو استقبال النتائج . فوحدات الادخال مثلاً هي الوحدات التي تؤثر على الكمبيوتر وتجعله يتفاعل مع الوسط الخارجي ك لوحة المفاتيح مثلاً أما وحدات الاخراج فهي الوسط الذي يؤثر عليه الكمبيوتر كالشاشة أو الآلة الطباعة .

- 2 - وحدات التخزين حيث تستخدم هذه الوحدات لتسجيل البيانات وتخزينها من أجل الرجوع إليها في فترات لاحقة، ويمكن التمييز بين أنواع هذه الوحدات،

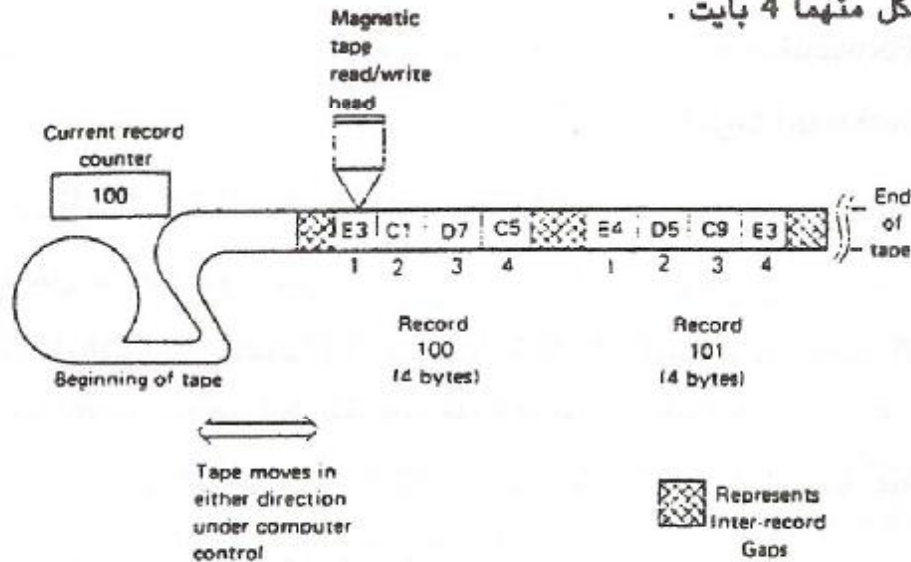
وذلك من خلال زمن المعالجة (Access time) T_{ij} (الزمن اللازم لمعالجة وحدة البيانات i وذلك عند وقوف رأس القراءة والكتابة عند وحدة البيانات i).

وتقسم وحدات التخزين اعتماداً على زمن المعالجة الى :

- وحدات التخزين التتابعية Serial storage devices
- وحدات التخزين ذات الوصول المباشر الكامل Completely direct access storage devices
- وحدات التخزين المباشرة Direct access storage devices

وحدات التخزين التتابعية :

تخزن البيانات في هذه الوحدات بشكل تتابعي، ويتم استرجاعها أيضاً بشكل تتابعي. ومن الأمثلة على هذه الوحدات : الشريط المغناطيسي . وعند تخزين البيانات على الشريط المغناطيسي، فإن وحدات البيانات (Bytes) تجمع في سجلات (Records) غير محددة الطول من (1 - 32768 bytes) ويوضح الشكل (4 - 4) مقطعاً من شريط مغناطيسي يتضمن سجلين طول كل منهما 4 بايت .



الشكل (4 - 4) : وحدة الشريط المغناطيسي

يتم التوصل (التعرف) الى سجل معين وذلك بتحديد موقعه الفيزيائي على الشريط .

فالسجل الاول يقع في الموقع الاول والثاني في الموقع الثاني، وهكذا. ويعتمد زمن الوصول الى سجل معين على موقع رأس القراءة والكتابة. وعادة ما تمتلك وحدة القرص المغناطيسي عدداً يحدد موقع رأس القراءة والكتابة. وفي الشكل نرى ان قيمة هذا العداد هي 100 وللوصول الى السجل 101 لا بد من المرور عن السجل رقم 100 ، ولو كانت قيمة هذا العداد مساوية لـ 1 (رأس القراءة والكتابة يقف عند بداية الشريط) فلا بد من المرور عن السجلات (1, 2, ..., 100) للوصول الى السجل المطلوب .

تمتلك وحدة الاشرطة المغناطيسية العديد من عمليات الإدخال والإخراج ، والتي تستخدم في الوصول الى السجلات أو في عملية كتابة هذه السجلات . ومن هذه العمليات :

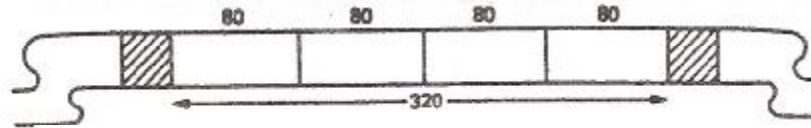
- تحريك الشريط إلى الامام أو الى الخلف Rewind
 - قراءة السجل التالي Read the next record
 - تسجيل السجل التالي Write the next record
 - القفز إلى الامام عن السجل بدون إجراء عملية قراءة أو الكتابة Skip Forward
 - القفز الخلفي عن سجل بدون إجراء عملية الكتابة أو القراءة Skip Backward
- ويمتاز الشريط المغناطيسي بالمميزات التالية :

- عرض الشريط 1/2 انش .
- كثافة التخزين (density) 1600 بايت في الانش الواحد، وفي بعض الاحيان 6250 اعتماداً على الطريقة المستخدمة في تسجيل البيانات .
- طول الشريط 2400 قدم، وبهذا فإن سعة التخزين في الشريط عند كثافة مساوية لـ 1600 بايت / انش هي 46 ميغا بايت .
- سرعة لف الشريط 200 انش/ثانية .

- الوسط الحسابي لزمن المعالجة 1 دقيقة .

- الزمن اللازم لقراءة السجل التالي = 4 ميلي ثانية .

ولتمييز السجلات عن بعضها البعض، يترك فراغ (IRG) بين كل سجل وآخر، وعادة ما يتراوح طول هذا الفراغ من $1/4 - 3/4$ الانش، وهذا بدوره يؤدي الى مدر الشريط المغناطيسي، وعدم استغلاله جيداً، وللتخلص مما أمكن من هذا العيب في الشريط تستخدم فكرة التكتل (Blocking) حيث يتم تجميع عدد من السجلات في كتلة فيزيائية واحدة على أن يترك الفراغ بين كتلة وأخرى، ويوضح الشكل (4 - 5) كيفية تجميع 4 سجلات في كتلة واحدة .



الشكل (4 - 5) : التكتل في الشريط المغناطيسي

والتكتل يحقق الفوائد التالية :

1 - الإقلال من زمن القراءة والكتابة حيث تقرا (أو تسجل) الوحدة الفيزيائية باستخدام عملية قراءة (كتابة) واحدة إذ أن مجموعة السجلات تقرا مرة واحدة باستخدام تعليمة قراءة واحدة بدلاً من قراءة كل سجل على حده، وبهذا فإن التكتل يقلل عمليات القراءة الى $n-1$ حيث n عدد السجلات في الكتلة الواحدة .

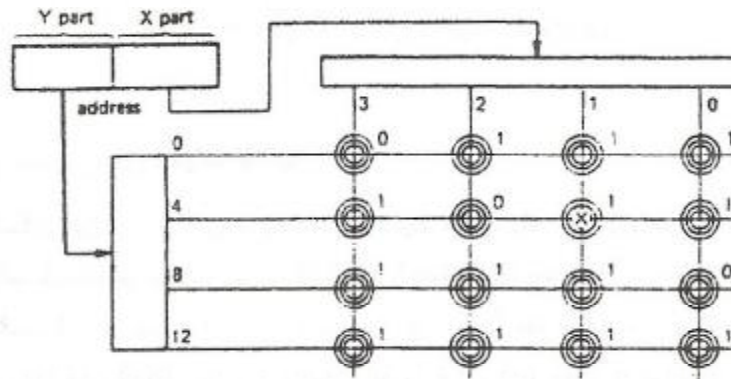
2 - استغلال الشريط المغناطيسي استغلالاً جيداً، وذلك بالإقلال من الفراغات، حيث

تخزن السجلات داخل الكتلة الواحدة بدون ترك فراغ، ويترك الفراغ فقط بين الكتلة والأخرى .

3 - الأسراع في عملية المعالجة نظراً للإقلال من الفراغات : ومن مساوئ التكتل الحاجة الى خوارزميات وبرمجيات خاصة لاجراء عملية التكتل والتفكك للوصول الى السجل المطلوب بالاضافة الى الحاجة الى مناطق تخزين مؤقتة ذات سعة عالية بدون استغلالها جيداً، فمثلاً لو احتجنا الى ذاكرة مؤقتة (buffer) بحجم 800 علماً بأننا بحاجة لسجل واحد فقط، أي ان نسبة استغلال الذاكرة المؤقتة هو 10٪ فقط .

وحدات التخزين ذات الوصول المباشر الكامل :

ومن الأمثلة على هذه الوحدات والتي تمتاز بثبات قيمة T_{ij} ذاكرة الحلقات المغناطيسية ، ذاكرة أشباه الموصلات، ذاكرة القراءة فقط (ROM) وذاكرة مصفوفة الديودات، والشكل (4 - 6) يوضح أحد أنواع هذه الوحدات .



الشكل (4 - 6) : ذاكرة الحلقات المغناطيسية

من الشكل (4 - 6) يلاحظ ان الذاكرة مكونة من 16 خلية مغناطيسية موزعة في مصفوفة مكونة من أربعة أسطر وأربعة أعمدة . وللوصول الى خلية معينة داخل هذه الذاكرة لا بد من تحديد رقم السطر ورقم العمود حيث تستخدم قيم العنوان x, y حيث تحلل قيمة x باستخدام مفكك الرموز (Decoder) لتحديد رقم السطر، وتحلل قيمة y لايجاد قيمة العمود، فمثلاً لو كان العنوان مساوياً لـ 0101 ($x = 01, y = 01$) فإن قيمة $y = x = 1$ وفي هذا النوع من الذاكرة ، فإن عملية القراءة تؤدي الى ضياع المعلومات ، لذا لا بد من إعادة كتابة المعلومات .

ويمتاز هذا النوع من الذاكرة بثبات T_{ij} ، وتتراوح قيمته بين 500 نانو ثانية الى 2 ميكرو ثانية، وعادة ما يستخدم هذا النوع في بناء الذاكرة الرئيسية، كما ويمكن استخدامها كوحدة تخزين .

تستخدم معظم أجهزة الكمبيوتر ذاكرة أشباه الموصلات، ويشبه هذا النوع من الذاكرة ذاكرة الحلقات المغناطيسية، إلا أن تكلفتها أقل، وسرعة المعالجة فيها أعلى، إذ قد تتراوح قيمة T_{ij} بين 50 و 550 نانو ثانية، ونظراً لسرعة هذه الوحدات العالية، لذا فإنها تربط عادة مع وحدة المعالجة المركزية مباشرة. هذا ويمكن ربطها مع وحدات الإدخال والإخراج عن طريق ضابط التحكم (I/O traffic controler) والذي سنعطي فكرة عنه لاحقاً في هذا الفصل .

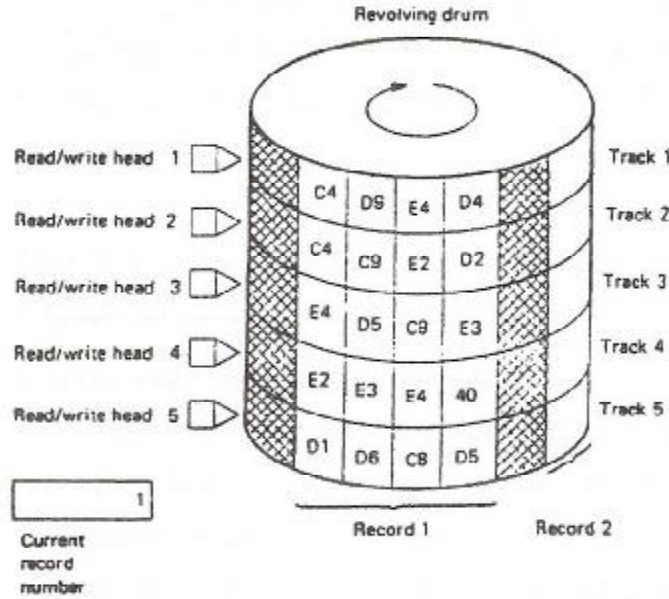
وحدات التخزين المباشرة :

تختلف هذه عن الوحدات السابقة بأن معدل التغير في قيمة T_{ij} غير ثابت، ولكنها قليلة مقارنة مع الوحدات التابعة، وتقسم هذه الوحدات الى :

- 1 - وحدات التخزين المباشرة ذات الرؤوس الثابتة .
- ب - وحدات التخزين المباشرة ذات الرؤوس المتحركة .

1 - وحدات التخزين ذات الرؤوس الثابتة :

من الأمثلة على هذه الوحدات القرص المغناطيسي أو الطبل المغناطيسي (Drum) ، والشكل (4 - 7) يوضح تركيب هذا النوع من الوحدات .



الشكل (4 - 7) : وحدة تخزين مباشرة ذات رؤوس ثابتة

يقسم القرص أو الطبل المغناطيسي الى مجموعة من المسارات (Tracks) حيث يقابل كل مسار رأس قراءة وكتابة بايت .

يتم تحديد السجل في القرص ، وذلك بتحديد رقم المسار ، ورقم السجل داخل المسار ، فمثلاً من الشكل (4 - 7) ، السجل (1,2) هو السجل XC4C9E2D2 ، وللعالجة سجل معين لا بد من تحديد عنوان هذا السجل ، أي لا بد من تحديد رقم المسار وترتيب السجل في المسار ، وبهذا فإن شكل تعليمة الإدخال والإخراج يصبح كما يلي :

READ (1) DASD record (2,1 into)

(2) Memory address (146 po)

(3) Length (80)

أي أن عملية القراءة تتضمن :

- 1 - تحديد الموقع، وذلك بتحديد رقم المسار، ورقم السجل في المسار .
- 2 - تحديد عنوان موقع الذاكرة والمراد نقل المعلومات إليه .
- 3 - تحديد طول السجل .

ومن خصائص هذا النوع من الوحدات :

- سرعة اللف : 10 ميلي ثانية .
- أكبر زمن للمعالجة : 10 ميلي ثانية .
- متوسط زمن المعالجة : 5 ميلي ثانية .
- المعالجة التتابعية (اعتماداً على السجل ، 1 ميلي ثانية) .
- سعة التخزين : 8 ميجا بايت (256 راسي * 32000 بايت / مسار) .

يختلف هذا النوع من الوحدات عن وحدات التخزين التتابعية في الأمور التالية :

- القرص أو الطبل المغناطيسي دائم الدوران في حين أن الشريط المغناطيسي من النوع Start/Stop ، لهذا فإن الفراغات في الشريط المغناطيسي تستخدم لغايات الإيقاف أو التحريك، بينما تستخدم الفراغات في القرص المغناطيسي لاعطاء وحدة المعالجة الفرصة لإجراء الحسابات اللازمة لتحديد السجل .

- باستخدام الخاصية الأولى يمكن إجراء عملية القراءة والمعالجة المحسنة للسجلات، وذلك في حالة المعالجة التتابعية، ولتفسير هذا المفهوم لناخذ المثال التالي :

لنفرض وجود 10 سجلات (A,B,...,J) مخزنة على طبل مغناطيسي يتسع المسار فيه لـ 10 سجلات .

تخزن هذه السجلات في المسار كما يلي :

Drum record	Data record
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J

لنفرض ان قراءة السجلات تتم بشكل متتابعي (يقرأ السجل A أولاً ثم B ، وهكذا) ، لنفرض أيضاً أن زمن قراءة السجل 2 ميلي ثانية، وزمن المعالجة هو 1 ميلي ثانية، لذا فبعد قراءة ومعالجة السجل A ، فإن الطبل سوف يصبح واقفاً عند السجل رقم 4 ، لذا لا بد من تحريك المسار مرة أخرى ليقف رأس القراءة والكتابة عند السجل B ، أي لا بد من زمن قدره 8 ميلي ثانية لاجراء هذا، وبهذا فإن الزمن لقراءة ومعالجة السجلات يصبح كما يلي :

$$\begin{aligned} T &= 10 (1\text{ms (read record)} + 2\text{ms (access record)} + 8 (\text{access} \\ &\quad \text{next record)}) \\ &= 110 \text{ ms} \end{aligned}$$

ويمكن تحسين عملية المعالجة هذه وذلك من خلال إعادة تخزين هذه السجلات كما يلي :

Drum record	Data record
1	A
2	H
3	E
4	B
5	I
6	F
7	C
8	J
9	G
10	D

لاحظ من هنا أنه بعد معالجة السجل A فإن رأس القراءة والكتابة سوف يصبح عند السجل B ، وبهذا فإن مجموع الزمن اللازم لمعالجة السجلات العشرة يصبح كما يلي :

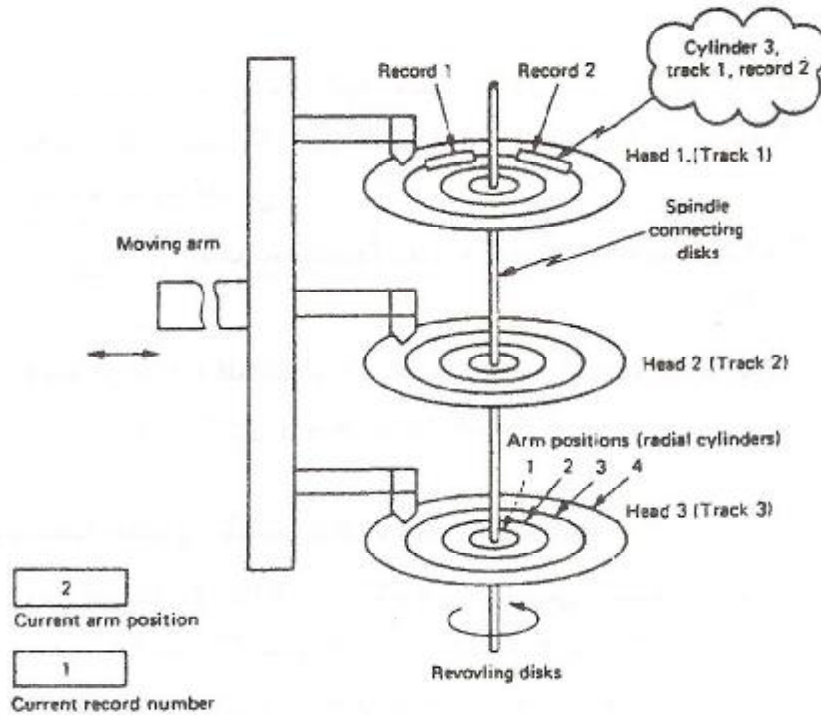
$$T = 10 (1 \text{ ms}(\text{read record}) + 2 \text{ ms} (\text{access record})) \\ = 30 \text{ ms}$$

لاحظ أن سرعة المعالجة هنا تفوق سرعة المعالجة سابقاً بأربع مرات ، وسوف يصبح الفرق واضحاً وكبيراً كلما زاد عدد السجلات .

ب - وحدات التخزين المباشر ذات الرؤوس المتحركة :

ومن الأمثلة على هذا النوع : القرص المغناطيسي ، حيث يقسم القرص الى مجموعة من الأسطح (الاسطوانات) حيث تدور كل الاسطوانات حول محور، ويقابل كل اسطوانة رأس قراءة وكتابة متحرك اماماً أو خلفاً. والشكل (4 - 8) يوضح تركيب القرص المغناطيسي .

- ويتم تحديد السجل، وذلك بتحديد رقم الاسطوانة ورقم المسار وترتيب السجل، ومن أهم خصائص هذا النوع من الوحدات :
- أقصى زمن المعالجة : 75 ميلي ثانية (55 ميلي ثانية لتحريك الرأس ، و 20 ميلي ثانية لدوران الاسطوانة) .
 - معدل زمن المعالجة : 30 ميلي ثانية .
 - زمن المعالجة التتابعية > 1 ميلي ثانية .
 - السعة : 100 ميغا بايت (400 اسطوانة × 20 رأساً × 1600 بايت / مسار) .

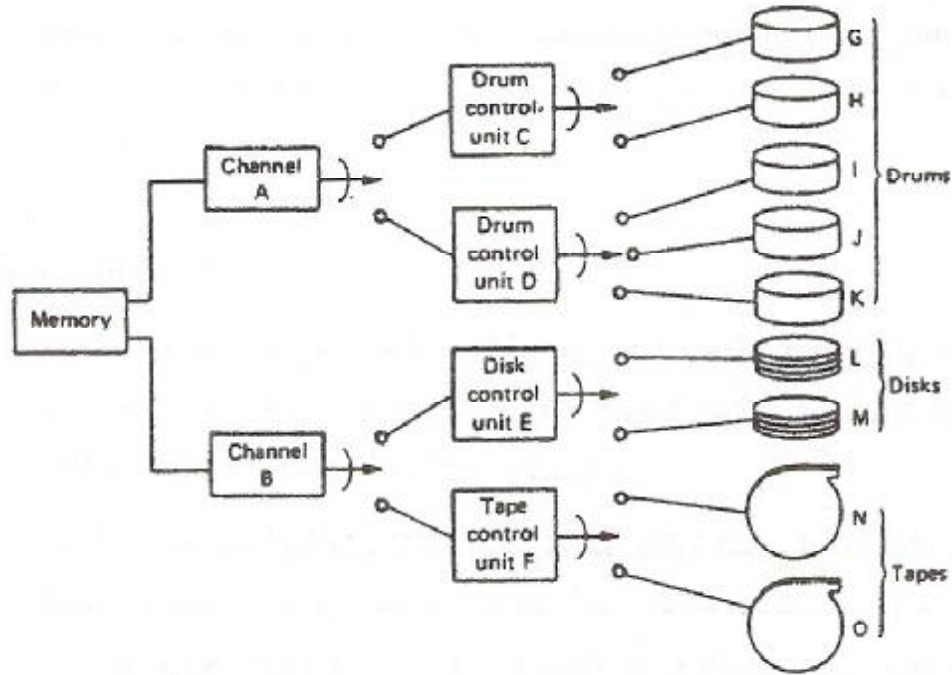


الشكل (4 - 8) : وحدة تخزين ذات رؤوس متحركة

4 - 2 - 3 : وحدات التحكم والقنوات :

نظراً لاستخدام وحدات التحكم والقنوات في عملية الإدخال والإخراج، لذا لا بد لبرمجيات إدارة وحدات الإدخال والإخراج من الاشراف على القنوات، ووحدات التحكم بالاضافة الى اشرافها على الوحدات الأخرى .

ونظراً لكون القنوات أجهزة متخصصة ومرتفعة الأسعار، فإن عددها يكون محدود في نظام الكمبيوتر عادة، لذا لا بد لبرمجيات إدارة وحدات الإدخال والإخراج من الاشراف على عملية ربط وفصل القناة مع وحدة الإدخال والإخراج حيث يوضح الشكل (4 - 9) كيفية إتصال الوحدات المختلفة مع القنوات .



الشكل (4 - 9) : وصل وحدات الإدخال والإخراج مع الذاكرة

استقلالية عمل وحدات الإدخال والإخراج :

في بعض الأحيان تمتلك بعض وحدات الادخال أو الاخراج القدرة على معالجة عملية الإدخال والإخراج تلقائياً بدون مساعدة وحدة التحكم أو القناة. فعملية البحث عن سجل في القرص المغناطيسي تتطلب 50 ميلي ثانية وتحتاج القناة فقط لنقل عنوان البحث من الذاكرة الى القرص والتي تتطلب في اسوأ الاحتمالات 20 ميكروثانية ، وبعد إرسال العنوان فإن وحدة القرص تقوم بعملية البحث بشكل مستقل .

وفي بعض الأحيان تتطلب عملية الادخال أو الاخراج اشتراك وحدة الادخال أو الاخراج ووحدة التحكم في تنفيذ العملية بدون الحاجة الى القناة .

عند استصدار أمر الإدخال أو الإخراج لا بد من تحديد ممر عبر القناة بين وحدة الإدخال أو الإخراج والذاكرة الرئيسية حيث يستخدم مبدأ الاعتراض لايجاد مثل هذا الممر ، ومن هذه الاعتراضات (Channel-end, Control) (unit-end, device-end) ، ويحدد نوع الاعتراض بواسطة تحليل كلمة القناة (CSW) .

التخزين Buffering :

عند إجراء عملية القراءة أو الكتابة من (الى) وحدات الادخال أو الاخراج تستخدم عادة وحدة تخزين مؤقتة للتوفيق بين سرعة وحدة الإدخال والإخراج وسرعة القناة أو الذاكرة الرئيسية .

فمثلاً قارئ البطاقات المثقبة يحتوي على ذاكرة تتسع لسجل واحد (80 بايت) . وعادة ما تكون هذه الذاكرة جزءاً من وحدة التحكم الخاصة بوحدة الإدخال والإخراج. وعند الحاجة لإجراء عملية القراءة، فإن القناة تتصل بالذاكرة المؤقتة لأخذ محتوياتها حيث تنتقل هذه المحتويات وتحرر القناة في فترة زمنية مساوية لفترة عملية النقل (لا تتعدى 100 ميكروثانية) .

(عادة تزود وحدات الإدخال والإخراج البطيئة بالذاكرة المؤقتة) .

الممر المتعدد Multiple path :

نظراً لتكلفة القنوات الباهظة تستخدم فكرة الاتصال المتعدد (الممر المتعدد) كما هو موضح بالشكل (4 - 10) .

من الشكل (4 - 10) يتضح بأن هناك 4 ممرات بين الذاكرة الرئيسية ووحدة الإدخال E :

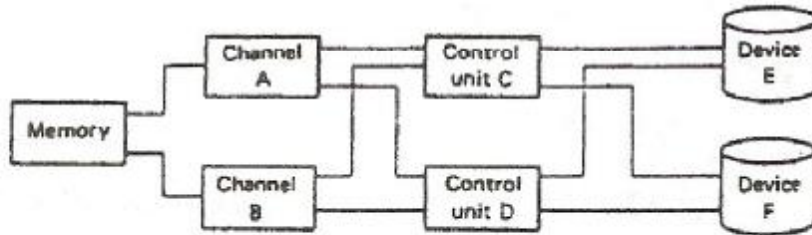
1- Channel A control unit C - Device E

1- Channel A control unit D - Device E

1- Channel B control unit C - Device E

1- Channel B control unit D - Device E

ويؤدي استخدام فكرة الممر المتعدد الى زيادة وثوقية نظام الإدخال والإخراج حيث يبقى هذا النظام عاملاً حتى عند تعطل إحدى القنوات أو إحدى وحدات التحكم .



الشكل (4 - 10) : الممر المتعدد

4 - 2 - 4 : مكونات برمجيات إدارة وحدات الإدخال والإخراج :

تتضمن برمجيات إدارة وحدات الإدخال والإخراج الأجزاء التالية :

- مشرف الممر I/O traffic controler حيث يقوم هذا الجزء بمعرفة حالة كل من وحدة الإدخال والإخراج، حالة وحدة التحكم وحالة القناة .

- مجدول الإدخال والإخراج (I/O scheduler) حيث يقوم هذا الجزء باتخاذ سياسة معينة لحجز القناة أو وحدة التحكم أو وحدة الإدخال والإخراج للعمل القابل للتنفيذ .

- قائد وحدة الإدخال والإخراج (I/O device handler : Driver) حيث يوجد قائد لكل وحدة إدخال أو إخراج يقوم بالمهام التالية :

- 1 - حجز القناة اللازمة لتنفيذ عمليات الإدخال والإخراج .
- 2 - الاشراف على عملية الإدخال والإخراج .
- 3 - الاتصال بمجدول الإدخال والإخراج لتقرير زمن حجز وحدة المعالجة المركزية ثم ربط وحدة الادخال أو الاخراج مع هذه الوحدة .

1 - مشرف الممر :

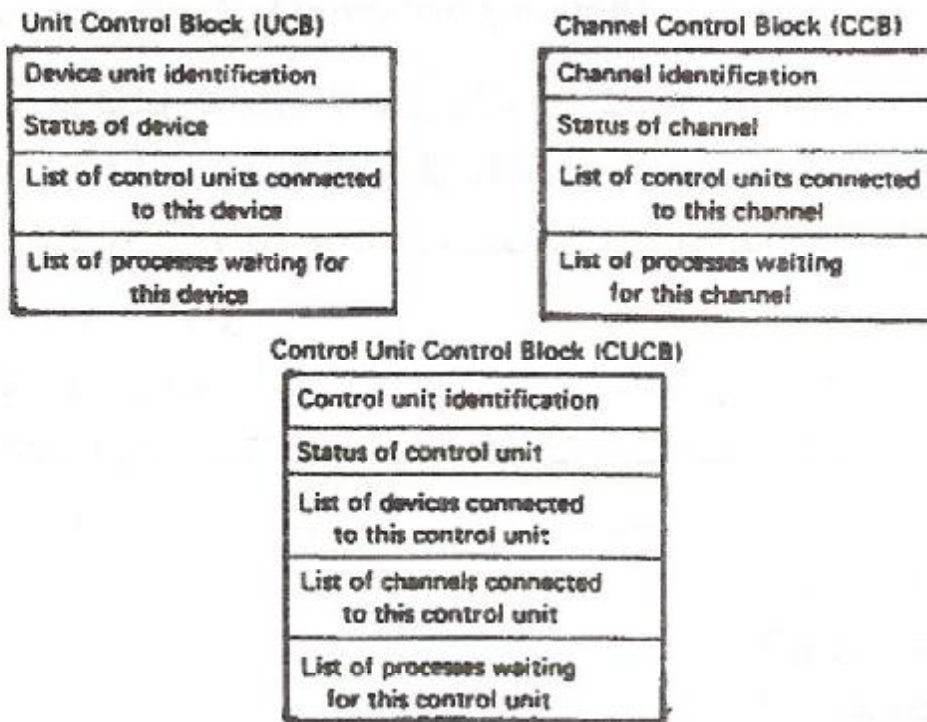
يقوم مشرف الممر كما أسلفنا سابقاً بتحديد حالة الوحدة حيث يقوم بإصدار معلومات خاصة تتعلق بحالة الوحدات المشتركة في عملية الإدخال والإخراج، ويقوم المشرف بالاجابة عن الاسئلة التالية :

- 1 - هل هناك ممر متوفر للاستجابة لطلب الإدخال والإخراج .
- 2 - هل هناك أكثر من ممر متوفر .
- 3 - اذا لم يكن هناك ممر متوفراً، فمتى يصبح أحد الممرات متوفراً .

ومن أجل الاجابة على هذه الاسئلة فإن مشرف الممر يصدر مجموعة من المعلومات المتعلقة بحالة الوحدات وإمكانية ربط الوحدات لتنفيذ عملية الإدخال والإخراج حيث تتضمن هذه المعلومات ما يلي :

- 1- (Unit control block) UCB
- 2- (Control unit control block) CUCB
- 3- (Channel control block) CCB

والشكل (4 - 11) يوضح أهم محتويات هذه الأجزاء .



الشكل (4 - 11) معلومات الحالة (وحدات التحكم)

2 - جدول الإدخال والإخراج :

إذا كان عدد طلبيات الإدخال والإخراج أكبر من عدد الممرات المتوفرة، فلا بد من تحديد سياسة معينة يتم بواسطتها اختيار العدد المطلوب من طلبيات الإدخال والإخراج والمساوي لعدد الممرات المتوفرة حيث يقوم بهذه المهمة جدول الإدخال والإخراج، وهناك سياسات متعددة لتنفيذ طلبيات الإدخال والإخراج تشبه إلى حد كبير السياسات المتبعة في تنفيذ العمليات والمنفذة من قبل جدول العمليات .

3 - قائد وحدة الإدخال والإخراج :

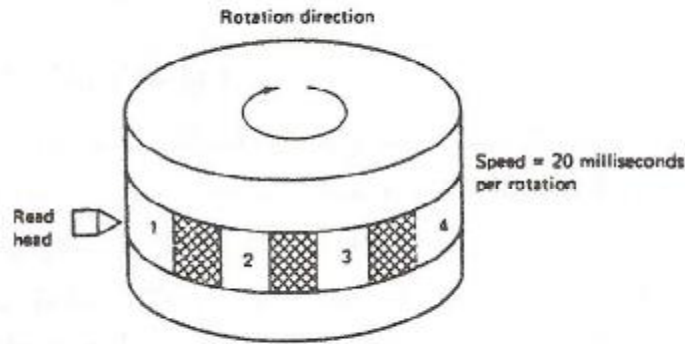
بالإضافة الى الوظائف التي أشرنا إليها سابقاً لهذا الجزء، فإن هذا الجزء مهم لاكتشاف الأخطاء أثناء عمليات الإدخال والإخراج بالإضافة الى الإستجابة لاعتراض الإدخال والإخراج، وتعتمد خواص وتركيب قائد الوحدة على نوع وحدة الإدخال والإخراج المستخدمة .

الوحدات المتحركة دائرياً (Rotating ordering)

عندما تكثر عمليات الإدخال والإخراج، فإنه سوف يتولد دوراً من الطلبات المنتظرة لنفس وحدة الإدخال والإخراج وأن تنظيم هذا سوف يؤدي بدوره الى التقليل من مجموع الزمن لخدمة طلبات الإدخال والإخراج .
لنأخذ المثال التالي :

لنفرض وجود 4 سجلات على الطبل المغناطيسي ، وأن طلبات الإدخال والإخراج التالية تم وصولها علماً بأن الممر متوفر كما هو مبين في الشكل (4 - 12) .

- 1- Read record 4
- 2- Read record 3
- 3- Read record 2
- 4- Read record 1



الشكل (4 - 12) : السجلات على الطبل المغناطيسي

1 - الطلب A :

إذا حقق جدول الأعمال الطلب بقراءة السجلات (1, 2, 3, 4) ، فإن الزمن اللازم للمعالجة سيصبح مساوياً :

$$T_{74} + T_{34} + T_{32} + T_{21} - 3 \text{ revolution} \\ (3/4 + 3.3/4 \text{ rotation})$$

والذي يساوي بدوره 60 ميلي ثانية .

فعند قراءة السجل الرابع فإننا لا نعرف موقع رأس القراءة والكتابة، لذا فإننا نفترض نصف دورة (10 ميلي ثانية) لتحديد موقعه ، و 1/4 دورة للقراءة (5 ميلي ثانية) .

2 - الطلب B :

إذا حقق الجدول طلب القراءة (1,2,3,4) فإن الزمن اللازم للمعالجة هو:

$$T_{71} + T_{12} + T_{23} + T_{34} - 1.5 \text{ revolution} \\ (3/4 + 3.1/4 \text{ rotation}) = 30 \text{ ms}$$

لاحظ أن الزمن هنا نصف الزمن في الحالة A .

3 - الطلب C :

إذا كان رأس القراءة يقف عند السجل 3 وعلم بهذا الجدول ($? = 3$) .

فإن قراءة السجلات سوف تصبح كما يلي :

$$4, 1, 2, 3$$

وبهذا فإن زمن المعالجة يصبح مساوياً لـ :

$$T_{34} + T_{41} + T_{12} + T_{23} = 1 \text{ revolution} \\ = 20 \text{ ms}$$

ولتنفيذ الطلب C لا بد للمجدول بالاحساس بموقع الدوران إذ لا بد من تزويد الجدول بالامكانية المطلوبة لتحديد الموقع وبدء عملية القراءة من هذا الموقع وذلك للوصول بزمن المعالجة الى القيمة الصغرى .

العناوين البديلة (Alternation addresses) :

تعتمد سرعة المعالجة على زمن الدوران، والذي يعتبر ثابتاً، ولزيادة سرعة المعالجة يمكن تخزين السجل في أماكن متعددة واستخدام العناوين البديلة للوصول الى السجل (تسمى هذه العملية بـ Flooding) .

لنفرض وجود وحدة يحتوي فيها المسار على 8 سجلات وان زمن الدوران مساو لـ 20 نانو ثانية .

فاذا خزن السجل A في المسار الأول ، السجل الأول ، فإن الزمن المتوقع للوصول الى هذا السجل هو زمن نصف دورة (أي 10 نانو ثانية) ، أما اذا خزنا نسخة إضافية من هذا السجل في الموقع مسار (1) سجل (5) ، فإن الزمن سوف ينقص الى 5 نانو ثانية وهكذا .

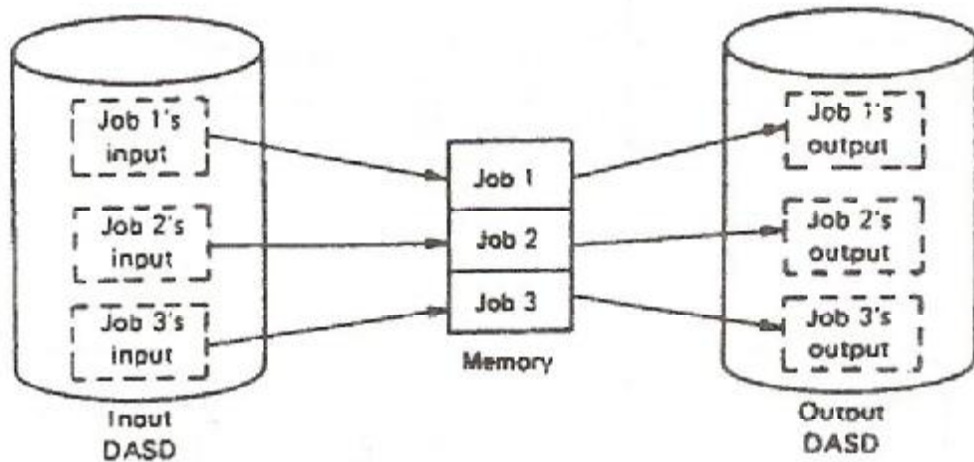
بالطبع، فإن هذا يحتاج الى حجم كبير من الذاكرة، فزيادة سرعة المعالجة هنا تتم على حساب حجم الذاكرة، فكلما زاد عدد النسخ من السجل كلما زادت سرعة المعالجة وكلما زاد الحجم المطلوب من الذاكرة .

4 - 2 - 5 : الوحدات الافتراضية Virtual device :

الوحدات الافتراضية هي عبارة عن وحدات غير مشتركة مثل الآلة الطابعة تم تحويلها الى وحدات مشتركة (مثل القرص المغناطيسي) وذلك باستخدام تقنيات وبرمجيات خاصة كنظام التمرير مثلاً (Spooling system) . كما أسلفنا فإن تحويل الوحدات غير المشتركة الى وحدات افتراضية مشتركة يؤدي الى تحقيق :

1 - حل مشكلة العدد المحدود من وحدات الإدخال والإخراج المستخدمة .

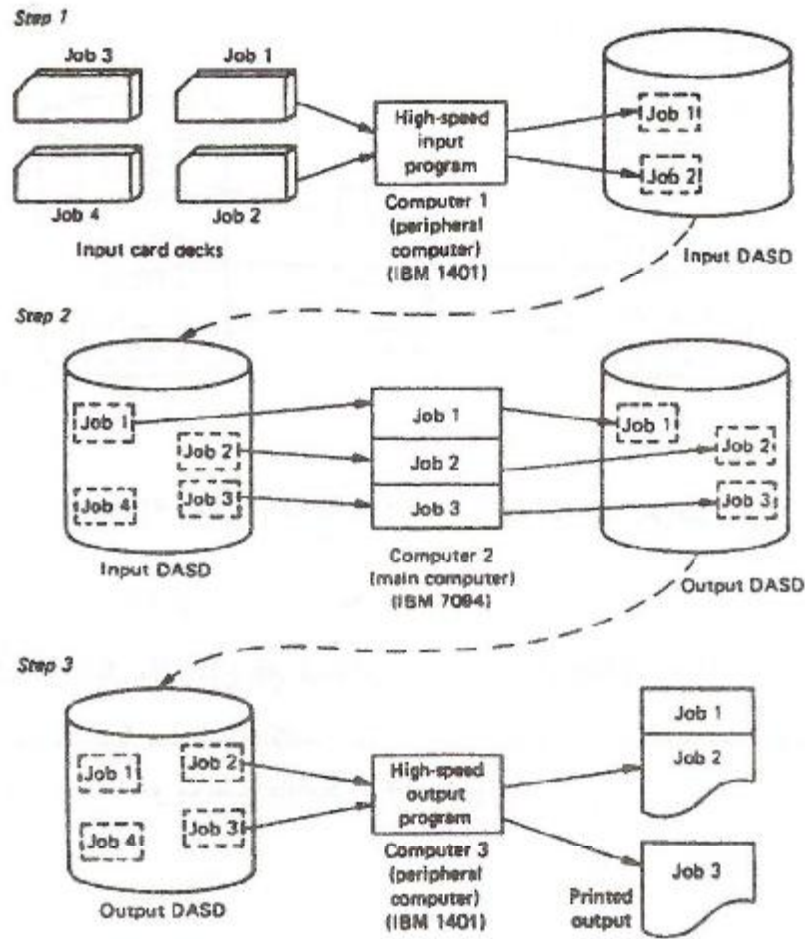
2 - حل المشكلة تفاوت السرعة بين الوحدات غير المشتركة ووحدة المعالجة المركزية، حيث يتم حل هاتين المشكلتين بتحويل الوحدات غير المشتركة الى وحدات إدخال وإخراج مباشرة كما هو موضح بالشكل (4 - 13) .



الشكل (4 - 13) : استخدام الوحدة المباشرة للإدخال والإخراج

وحدات الإدخال والإخراج غير المباشرة (OFF-Line I/O Devices)

تتم عملية معالجة الإدخال والإخراج في هذا النوع من الوحدات على ثلاث مراحل كما هو موضح بالشكل (4 - 14) .



الشكل (4 - 14) : الوحدات غير المباشرة

ففي المرحلة الاولى يتم نقل بيانات وحدة الإدخال غير المباشر بواسطة كمبيوتر متخصص الى وحدة ادخال مباشر .

اما في المرحلة الثانية فيتم معالجة البيانات بأخذها من وحدة الادخال المباشر حيث تعالج باستخدام الكمبيوتر الرئيسي .

اما في المرحلة الثالثة فيتم نقل النتائج من وحدة الادخال / الاخراج المباشر، وعن طريق كمبيوتر خاص الى وحدات الاخراج .

تقوم عملية المعالجة غير المباشر على حل المشكلتين السابقتين (محدودية وحدات الإدخال والإخراج وتدني سرعتها) إلا أن الطريقة تخلق الكثير من المشاكل ألا وهي :

1 - لا بد من تدخل العنصر البشري في عملية المعالجة .

2 - زيادة عدد أجهزة الكمبيوتر المستخدمة .

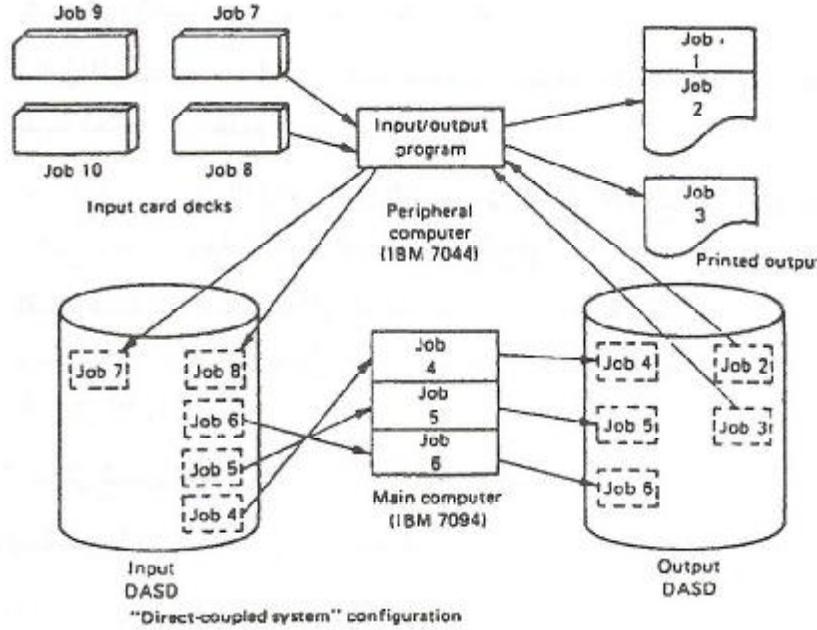
3 - جدولة عمليات الإدخال والإخراج .

عادة ما تستخدم هذه الطريقة في أنظمة الحزم (Batch system) .

الاتصال المباشر Direct - coupled system

للتقليل من عيوب الطريقة غير المباشرة في المعالجة تستخدم الطريقة المباشرة في المعالجة ، وذلك بربط وحدات الإدخال والإخراج مع وحدة المعالجة المركزية مما يؤدي بدوره الى زيادة سرعة المعالجة، والشكل (4 - 15) يوضح مفهوم الاتصال المباشر .

فباستخدام برنامج الإدخال والإخراج يمكن نقل البيانات مباشرة الى وحدة تخزين مباشرة حيث تعالج البيانات من هذه الوحدة ويتم اخراج النتائج الى وحدة تخزين مباشرة، ومنها وباستخدام برنامج الإدخال والإخراج تخرج النتائج الى وحدات الإخراج .

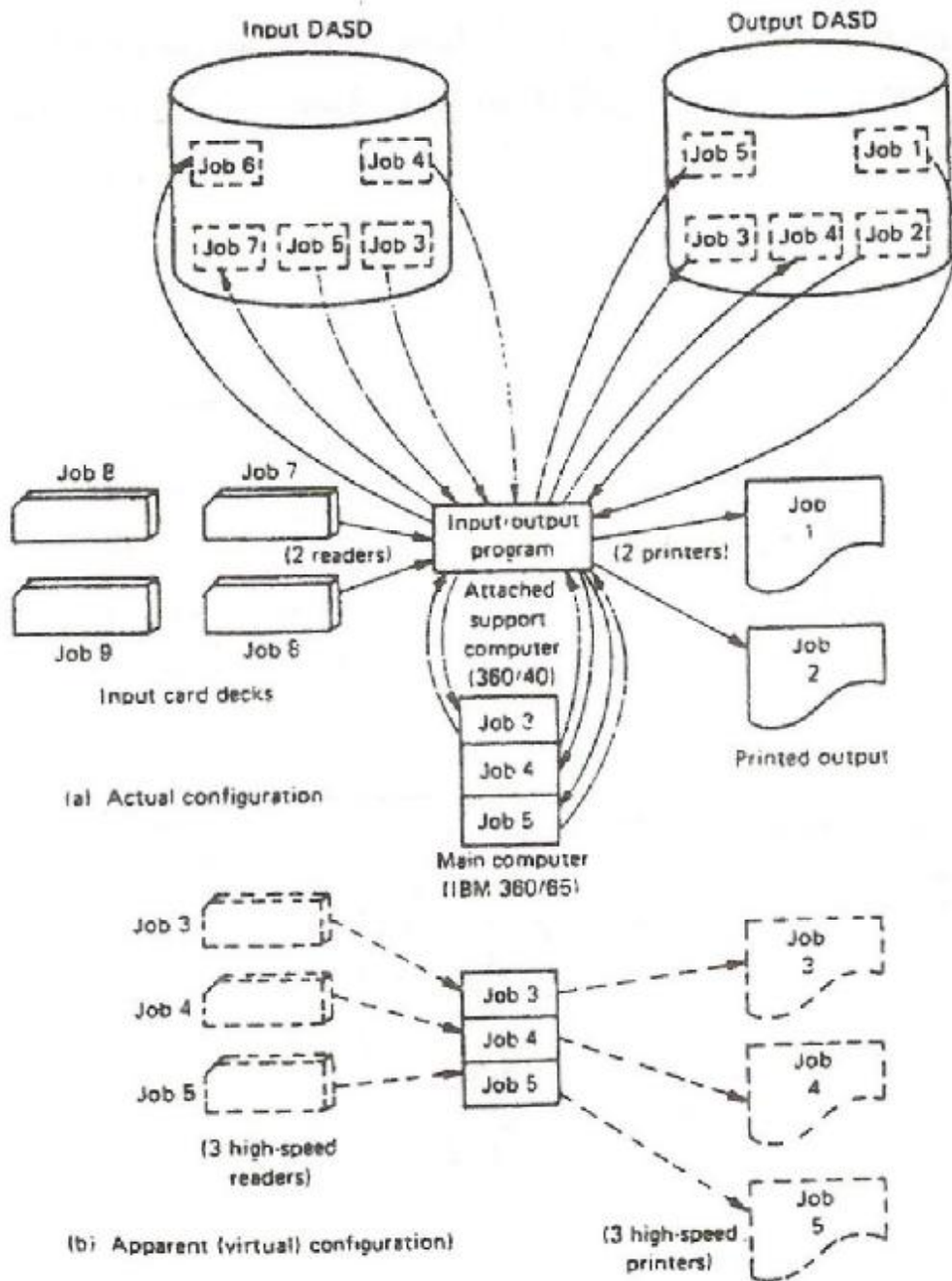


الشكل (4 - 15) : الاتصال المباشر

وحدات المعالجة الداعمة Attached Support Processor

لزيادة سرعة عملية الإدخال والإخراج يمكن ربط وحدات الإدخال والإخراج مع الكمبيوتر الرئيسي، وذلك عن طريق وحدة سريعة تسمى بوحدة المعالجة الداعمة (كما هو موضح بالشكل 4 - 16) .

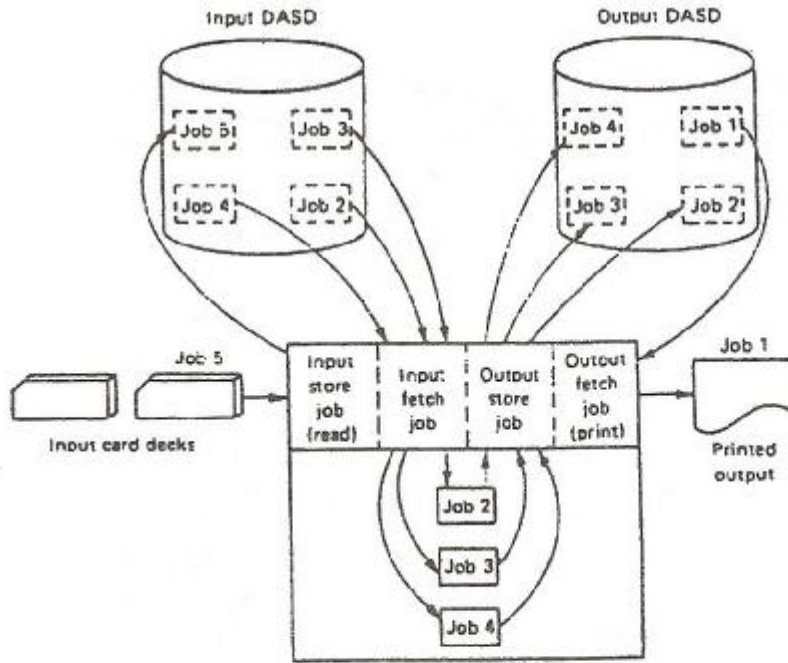
حيث تقوم هذه الوحدة بالتحكم بعمليات الإدخال والإخراج، بالإضافة إلى ذلك تقوم بعملية التخزين المؤقت وتجميع البيانات في كتل .



الشكل (4 - 16) : الوحدة الداعمة

4 - 2 - 6 : النظام الافتراضي (Virtual system)

كما أسلفنا سابقاً، فإنه يمكن تحويل وحدات الإدخال والإخراج غير المشتركة وحل المشاكل المتعلقة باستخدام هذه الوحدات وذلك من خلال تقنيات وبرمجيات خاصة تسمى بنظام التمرير (Spooling system)، كما هو موضح بالشكل (4 - 17).



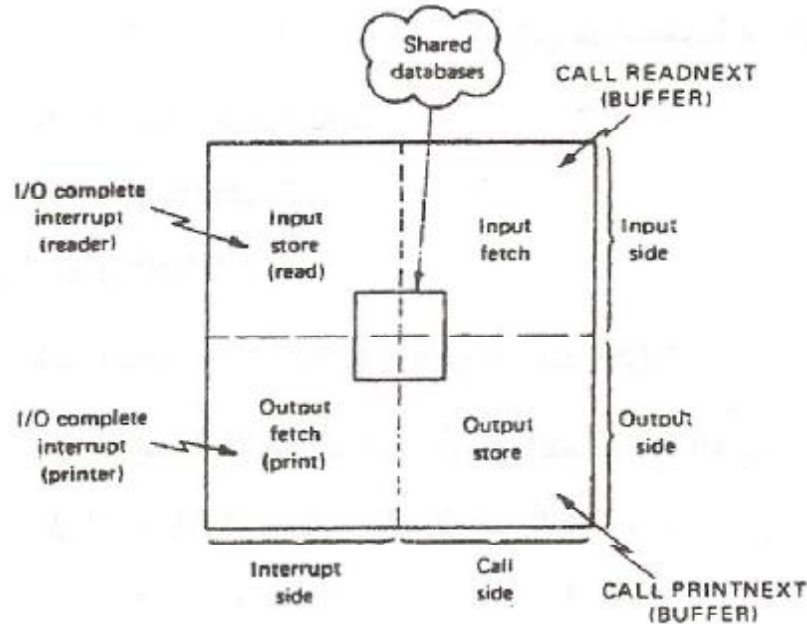
الشكل (4 - 17) : نظام التمرير

حيث يمكن تلخيص عمل نظام التمرير في ما يلي :

- 1 - فتح منطقة للادخال تستخدم لاستقبال البيانات وذلك في وحدة التخزين المباشرة .
- 2 - فتح منطقة للإخراج على وحدة التخزين المباشرة تستخدم لاستقبال النتائج .
- 3 - إجراء عملية المعالجة، وذلك باستحضار البيانات المطلوبة من منطقة الادخال .
- 4 - نقل النتائج الى منطقة الإخراج .
- 5 - استحضار النتائج المطلوب إخراجها .
- 6 - إخراج النتائج المطلوبة الى وحدة الإخراج .

تصميم نظام التمرير :

يُفترض أن تكون برمجيات نظام التمرير جزءاً من برمجيات نظم التشغيل ولكنها تقوم بالأعمال المنوطة بها بشكل مستقل وبشكل عام فإن نظام التمرير للادخال والإخراج يقسم الى أربعة أقسام كما هو موضح في الشكل (4 - 18) .



الشكل (4 - 18) : الشكل العام لنظام التمرير

- أما ميكانيكية عمل هذا النظام فهي كما يلي :
- اعتراض الإدخال يقوم بتنفيذ برنامج الإدخال .
- عملية المعالجة تتم باستدعاء برنامج قراءة السجل التالي CALL Reading .
- تخزين النتائج يتم باستدعاء برنامج التسجيل CALL Printnext .
- عملية الإخراج تتم بتقبل الاعتراض I/O Complete interrupt .

نظام الإدخال (Input system)

يقوم نظام الإدخال بأجراء العمليات الرئيسية التالية :

- 1 - قراءة كل سجل فيزيائي وتسجيله على وحدة التخزين المباشرة .
- 2 - إتاحة الفرصة لمعالجة السجل من وحدة التخزين المباشرة .

ويتم البد بتنفيذ البرنامج (نظام الإدخال) ، وذلك بعد تقبل اعتراض الإدخال والإخراج ، أما عملية المعالجة فتتم من خلال برنامج خاص Read/next ، والتي تقوم بإنشاء قاعدة للبيانات كما هو موضح بالشكل (4 - 18) حيث تتألف هذه القاعدة بالاضافة الى منطقة التخزين من مجموعة من الجداول :

- جدول الإدخال Input spool table

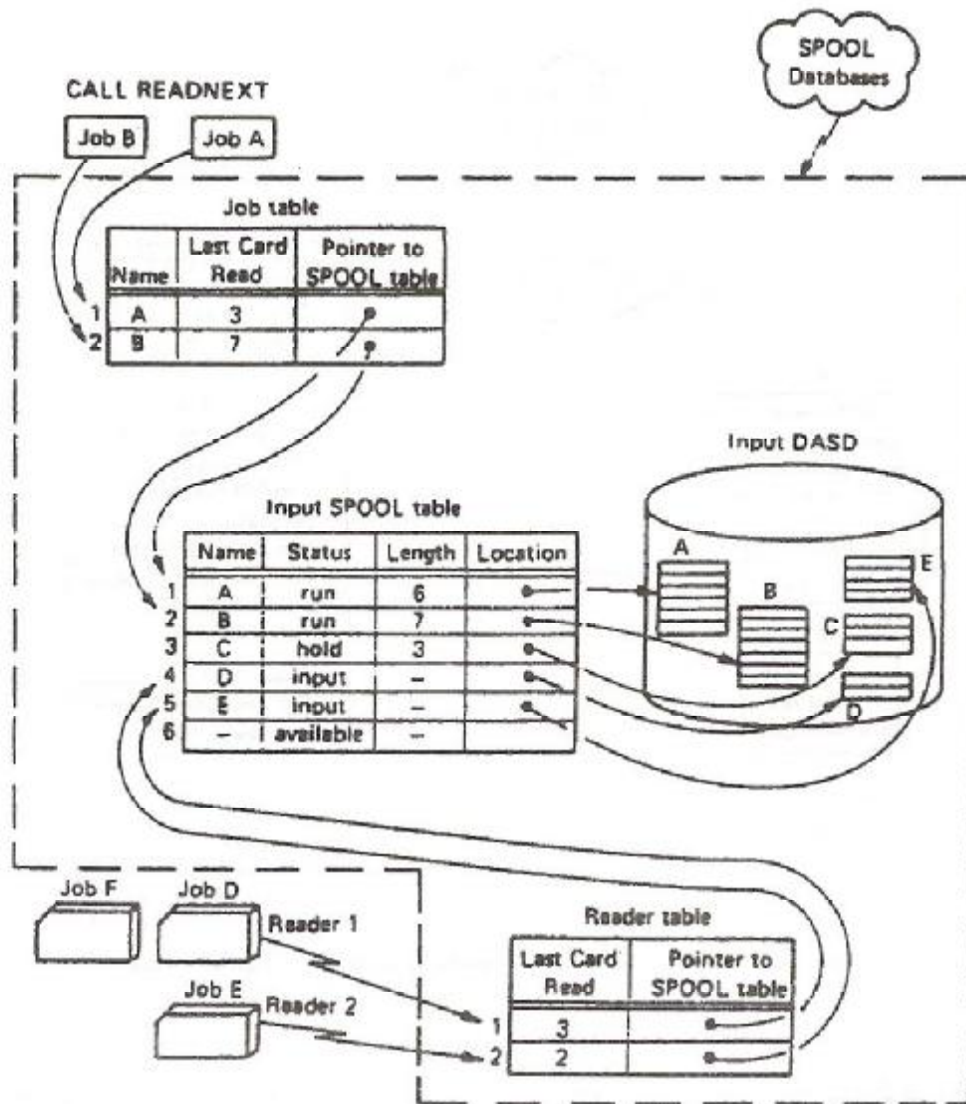
- جدول القراءة Reader table

- جدول الأعمال Job table .

حيث تشير Input : الى ان المدخلات لا زالت تحت القراءة

Hold : الى أن عملية القراءة قد تمت ولكن لم تتم معالجة العمل .

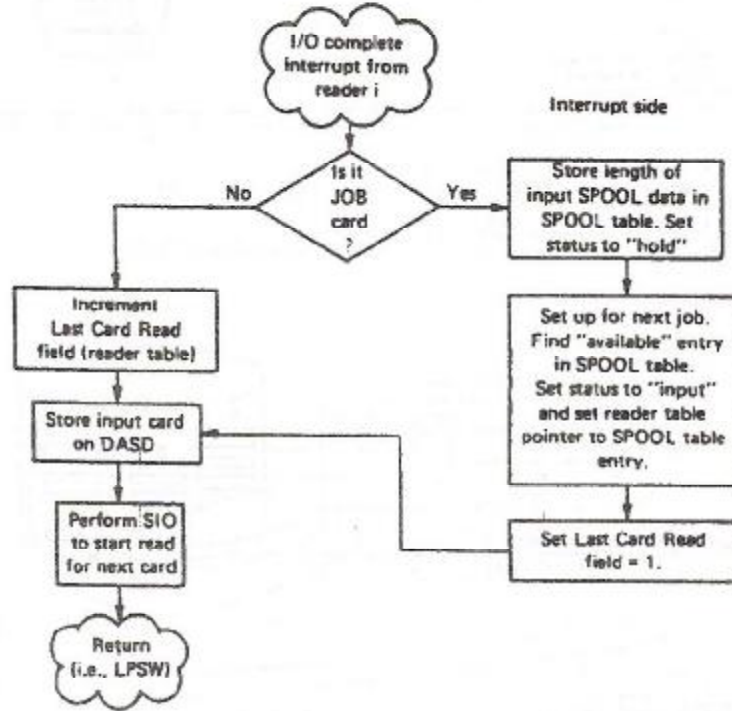
Run : الى أن القراءة تمت والعمل المناظر قيد التنفيذ .



الشكل (4 - 18) : قاعدة البيانات لنظام التمرير

خوارزمية برنامج الإدخال :

بالإضافة إلى جدول المدخلات فإن برنامج الإدخال يعالج جدول القراءة وجدول الأعمال حيث يحتوي الجدول الأول على معلومات تتضمن حالة قارئ البطاقات بالإضافة إلى مؤشر يشير إلى الموقع المخصص للعمل المطلوب في جدول الإدخال ، ويمكن تناول خوارزمية برنامج الإدخال من خلال عملية الاعتراض ومن خلال استدعاء برنامج القراءة والمخططات التالية توضح عمل برنامج الإدخال شكل (4 - 19) ، شكل (4 - 20) .

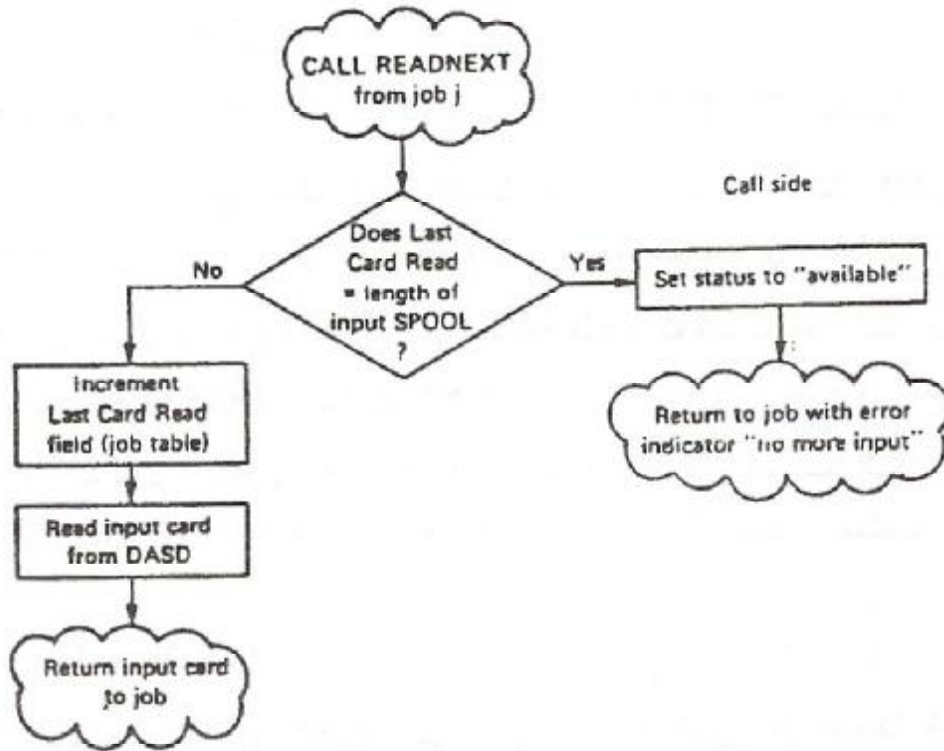


الشكل (4 - 19) : عمل برنامج الإدخال من زاوية الاعتراض

بالإضافة إلى برنامج الإدخال يتكون نظام التمرير من برنامج الإخراج، ويشبه عمل هذا البرنامج عمل برنامج الإدخال، مشرف برنامج التمرير، برامج تسلسل الأعمال، ويمكن تلخيص عمل كل برنامج من هذه البرامج كما يلي :

1 - عمل المشرف (Supervisor) :

- يقوم المشرف بإيجاد وحدة التخزين المباشرة لاستخدامها لعملية التنفيذ وقراءة المعلومات، ويطلق على هذه المنطقة بالمنطقة R .
- يقوم المشرف بإيجاد منطقة أخرى على وحدة التخزين المباشرة تستخدم لتقبل نتائج المعالجة ويطلق عليها بالمنطقة P .



الشكل (4 - 20) : عمل برنامج الإدخال من زاوية برنامج القراءة

ب - عمل برامج الإدخال :

- يقوم المشرف باعطاء برنامج الإدخال أمر البدء بالعمل وذلك بعد تقبل اعتراض الإدخال والإخراج .
- تقوم برامج الإدخال بنقل المعلومات الى المنطقة R .
- توقف هذه البرامج عملها حتى الانتهاء من عملية التمرير، وتخبر المشرف والذي بدوره يعاود أمر برامج الإدخال بالاستمرار في عملية التمرير وهكذا.
- عند وصول برنامج الإدخال الى نقطة النهاية (end) يخبر المشرف والذي بدوره يوقف هذا البرنامج .
- إذا كان هناك أعمال أخرى، فإنه يتم إعادة تنفيذ الخطوات السابقة، وإلا فإن برنامج الإدخال ينتقل الى حالة الانتظار .

ج - عمل برنامج الاخراج :

- يقوم برامج الاخراج بتمرير المعلومات من المنطقة P الى الآلة الطابعة .
- توقف هذه البرامج عملها لحين الانتهاء من عملية التمرير وتخبر المشرف عن هذا والذي بدوره يعاود أمر برامج الاخراج بالاستمرار في عملية التمرير .
- عند وصول برنامج الاخراج الى النهاية (end) توقف عملية التمرير حيث يقوم المشرف بإيقاف برنامج الاخراج .
- يُحدَّث دور المخرجات وذلك بطرح واحد منه، وإذا أصبح الدور خالياً، تنتقل برامج الاخراج الى حالة الانتظار، وإلا تكرر الخطوات السابقة .

د - عمل برامج التسلسل :

- تقوم هذه البرامج بالاشراف على تسلسل عملية التمرير الى المنطقة R أو من المنطقة P وبهذا نمكن تحديد وظائف هذه البرامج من خلال :
- 1 - الوظيفة R (الاشراف على تسلسل عملية التمرير الى المنطقة R) .
- 2 - الوظيفة P (الاشراف على تسلسل عملية التمرير من المنطقة P) .

الوظيفة R :

- فحص دور المدخلات، فإذا كان خالياً، فإن برنامج التسلسل تنتقل الى حالة الانتظار، أما إذا لم يكن خالياً فيتم الانتقال الى الخطوات التالية :
- يتم اختيار العمل ذو الأولوية العليا .
- يطرح واحد من دور المدخلات .
- يخبر المشرف والذي بدوره يقوم بتشغيل برنامج الإدخال .
- عند انتهاء برنامج الإدخال يضيف واحد الى دور المخرجات .

الوظيفة P :

يمكن إيجاز هذه الوظيفة في :

- فحص دور المخرجات، فإذا لم يكن خالياً، يتم الانتقال الى الخطوات التالية .
- يتم اختيار عمل معين حسب أولوية معينة من المنطقة P وتبدأ برامج الاخراج عملها بتمرير المعلومات من المنطقة P الى الآلة الطابعة .
- بعد الانتهاء من عملية التمرير يقوم المشرف بإخلاء المنطقة P والخاصة بالعمل .
- تقوم برامج التسلسل بتحديث جدول المخرجات، وذلك بطرح واحد منه .
- يفحص الدور من جديد ، فإذا كان خالياً تنتقل برامج التسلسل الى حالة الانتظار، وإلا تكرر الخطوات السابقة .

الوحدة الخامسة

برمجيات إدارة المعلومات (نظام الملفات)

Information Management Programs
(File System)

- مقدمة عامة .

- وظائف نظام الملف ومكوناته .

برمجيات إدارة المعلومات (نظام الملفات)

5 - 1 مقدمة عامة :

تخزن المعلومات على وحدات التخزين الثانوية (القرص المغناطيسي مثلاً) ويتم تنظيم هذه المعلومات في ملفات يمكن الرجوع اليها عند الحاجة لتنفيذ عمليات المعالجة المختلفة عليها .

تهتم برمجيات ادارة المعلومات بالهيكل التنظيمي للمعلومات أو بمعنى آخر التركيب الفيزيائي لهذه المعلومات والمتمثل في كيفية تخزين المعلومات على الوسط الفيزيائي وكيفية الرجوع اليها من خلال تحديد العنوان الفيزيائي لها وتتصل برمجيات إدارة المعلومات ببرمجيات إدارة قاعدة البيانات (مجموعة الملفات المرتبطة معاً) والتي تهتم بالتركيب المنطقي للملفات أي أن برمجيات إدارة قاعدة البيانات تهتم بمحتوى الملفات (أو تهتم بالوحدات المنطقية المؤلفة للملفات : السجلات) .

وسوف نتناول في هذه الوحدة برمجيات إدارة المعلومات أو ما يسمى نظام الملف (File system) .

كما أشرنا سابقاً فإن نظام الملف يهتم بالبنية الفيزيائية للمعلومات ويقصد بهذا المفهوم :

- تهيئة البنية الفيزيائية وجعلها قادرة على استقبال المعلومات وذلك من خلال فحص المسارات المكونة لاسطوانة التخزين، استحداث ما يلزم من قطاعات (قطاع الاستنهاض، جداول حجز الملفات (File allocation table : FAT) ، القطاعات التي تحدد مواقع الأدلة (Directory) .

- استحداث مناطق للدليل، والدليل ما هو إلا جدول يتكون من مجموعة من

المدخل بحيث يخصص كل مدخل الملف واحد ويضم المدخل الواحد معلومات عن الملف مثل: اسم الملف ، امتداد الملف ، حجم الملف ، قيود الملف ، تاريخ انشاء الملف وغيرها من معلومات .

- استحداث الملفات

هذا وسوف نستعرض عمليات التهيئة لاحقاً عند دراستنا للحالة الواقعية .

5 - 2 وظائف نظام الملف ومكوناته

تتلخص وظائف نظام الملفات في ما يلي :

- 1 - متابعة المعلومات (الملفات) المخزنة في الذاكرة الثانوية اعتماداً على دليل الملفات وجداول توزيع الملفات (FAT) .
- 2 - تحديد سياسة متابعة الملفات على وسائط التخزين وكيفية الوصول الى الملفات (تتابعياً، باستخدام الفهرس أو عشوائياً) .
- 3 - تخزين الملفات على واسطة التخزين وترحيلها للذاكرة الرئيسية عند الحاجة لمعالجتها .
- 4 - تحديث المعلومات على واسطة التخزين والفاؤها إذا تطلب الامر. ويتكون نظام الملف من الاجزاء الرئيسية التالية :
- 1 - نظام الملف الرمزي (Symbolic file system) والذي يعمل على البحث عن اسم الملف المطلوب في منطقة الدليل .
- 2 - نظام الملف الاساسي (Basic file system) والذي يعمل على قراءة المعلومات الاساسية عن الملف من منطقة الدليل .
- 3 - نظام حماية الملف (File protection system) ويتكفل هذا النظام بمطابقة قيود الملف مع شيفرة العملية المراد تنفيذها على الملف للتأكد من أن العملية المراد تنفيذها مطابقة لمحتوى قيود الملف في دليل الملفات .
- 4 - نظام الملف المنطقي (Logical file system) والذي يتولى حساب عنوان السجل المنطقي في الملف .

5 - نظام الملف الفيزيائي (Physical file system) والذي يعمل على تحويل العنوان المنطقي للسجل في الملف الى عنوان فيزيائي للسجل على واسطة التخزين .

6 - نظام التوطين (Stratigic allocation system) ويعمل هذا النظام على قراءة السجل أو كتابته وذلك بنقل السجل من منطقة التخزين المؤقتة في الذاكرة الرئيسية الى القرص في حالة الكتابة أو نقل السجل من القرص المغناطيسي الى الذاكرة المؤقتة لمعالجته في حالة القراءة .

هذا وسوف نستعرض أهم العمليات المنفذة على الملفات وعلى الأقراص المغناطيسية عند استعراضنا لدراسة الحالة الواقعية وسوف نستعرض أيضاً كيفية صيانة دليل الملفات من استحداث وتعديل وإضافة وحذف والغاء .

وفيما يلي سوف نستعرض مفهوم بعض العمليات المنفذة على الملفات وآلية تنفيذها .

1 - فتح الملف :

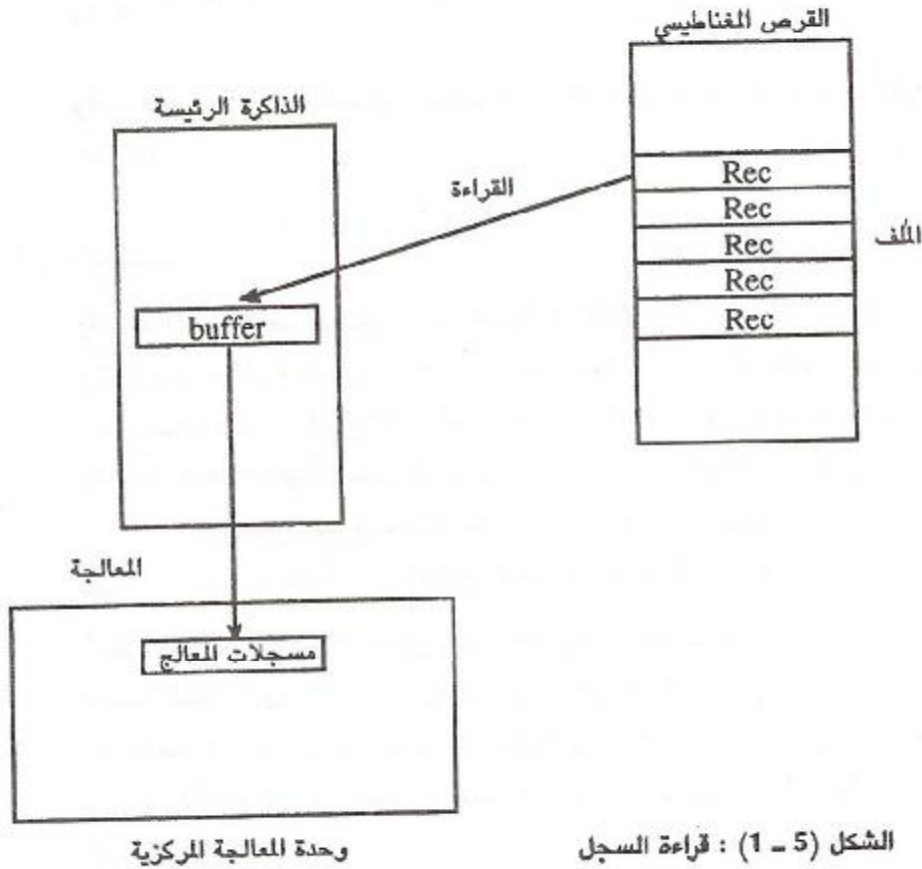
يقصد بفتح الملف تهيئة الملف للقراءة أو الكتابة أو للقراءة والكتابة معاً وتعتمد عملية الفتح على نوع الملف (مدخلات ، مخرجات ، مدخلات ومخرجات) فعند فتح الملف للقراءة (ملف المدخلات) يعمل مشرف نظام التشغيل (Supervisor) على تحويل اسم الملف الى نظام الملف والذي يتكفل بالبحث عن اسم الملف في منطقة الدليل فاذا كان الملف موجوداً يحدد موقع الملف (عنوانه) وينقل المؤشر (مؤشر الملف) الى بداية أول سجل فيه .

أما ملف المخرجات فيفتح بنفس الطريقة إلا أنه في حالة وجود الملف في منطقة الدليل فإن نظام الملف يعمل على تصفير حجم الملف (مسح المعلومات من الملف) (حتى ان احتوى على بيانات في السابق) ، أما اذا لم يكن الملف موجوداً فيتم استحداث مدخلة جديدة في الدليل يعبأ فيها المعلومات الأساسية عن الملف .

ب - عملية القراءة :

تنفذ عملية القراءة على ملف مهياً للإدخال أو للإخراج وتتم هذه العملية بقراءة وحدة فيزيائية قد تكون سجلاً أو مجموعة سجلات (كتلة) في حالة تنظيم السجلات في كتل وتتعاون مكونات نظام التشغيل في هذا كما يلي :

يخبر المشرف نظام الملف بتعليمه القراءة، ثم يعمل نظام الملف على الاتصال ببرمجيات الإدخال والإخراج وبرمجيات إدارة الذاكرة الرئيسية وذلك لانجاز عملية نقل السجل (الكتلة) من القرص الى منطقة تخزين مؤقتة (buffer) معرفة في الذاكرة الرئيسية كما هو مبين في الشكل (5 - 1) :



الشكل (5 - 1) : قراءة السجل

بعد هذا يعمل نظام الملف على نقل مؤشر الملف الى السجل التالي وبهذا يمكن معالجة السجل وذلك بنقل حقول السجل (أو جزء منها) من الذاكرة الرئيسية الى مسجلات وحدة المعالجة المركزية لتنفيذ العمليات الحسابية والمنطقية على الحقل (الحقول) وتتعاون في تحقيق مهمة المعالجة برمجيات إدارة الذاكرة وبرمجيات إدارة وحدة المعالجة المركزية .

وتجدر الإشارة ان عملية المعالجة تتم على السجلات واحد تلو الآخر (فيما لو قرأت كتلة وتواجد في منطقة التخزين المؤقتة مجموعة سجلات) .

جـ - عملية الكتابة :

تنفذ عملية الكتابة على ملف مهيا في حالة الإخراج أو في حالة الإدخال والإخراج وتتم عملية الكتابة بنفس الآلية المبينة في الشكل (5 - 1) وذلك بتغيير الاسهم وبعد الانتهاء من نقل السجل (الكتلة) من الذاكرة المؤقتة الى القرص يعمل نظام الملف على نقل مؤشر السجلات الى بداية السجل التالي وتعديل حجم الملف في منطقة الدليل .

دراسة لحالة واقعية

نظامي التشغيل DOS & UNIX

- نظام التشغيل DOS

- نظام التشغيل UNIX

نظام التشغيل DOS

ما هو Dos ؟؟

يسمح لك هذا النظام باستخدام التطبيقات والتعامل مع الملفات بسهولة، كما أنه يسمح لك بالاتصال مع الأجهزة الطرفية مثل الطابعات وسواقة الاسطوانات والسواقة الصلبة .

إستعمال اسطوانة نظام Dos (Using Dos Diskettes) :

عند استخدامك النظام لأول مرة لا بد من اختيار لوحة المفاتيح التي تستعملها. كما يلي :

ضع اسم وتاريخ الاسطوانة الجديدة بواسطة قلم لا يحوي رأساً مدبباً ثم احفظ الاسطوانة الأصلية في مكان أمين واستعمل النسخة الجديدة في أعمالك اليومية .

الاسطوانة وسواقة الاسطوانات (Diskettes Drives and Diskettes) :

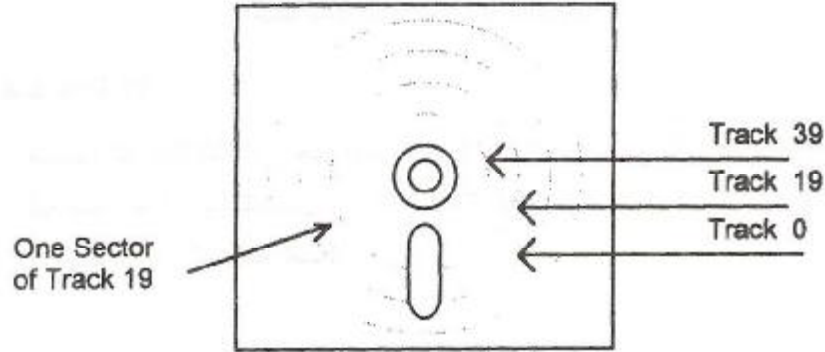
الأقسام / القطاعات الدائرية / البايت (Tracks, Bytes, and Sectors)

تتم عملية كتابة المعلومات على الاسطوانة وفق دوائر وحلقات تدعى (Track) كما ان رأس القراءة في سواقة الاسطوانة يتحرك من حلقة الى أخرى حسب دوران الاسطوانة وتتم اثناء هذه العملية قراءة البيانات وتحديد موقعها على الاسطوانة .

تستخدم التعابير Byte / Sector لتوصيف الاسطوانة بحيث ان الحلقات (Tracks) يتم تقسيمها الى قطاعات تدعى (Sector) تقاس سعة الاسطوانة بوحدة تدعى (Byte) .

ملاحظة :

إن عدد الحلقات والقطاعات والبايتات في الاسطوانة يختلف تبعاً لنوع سواقة الاسطوانات المستخدمة .



أنواع سواقات الاسطوانات (Types Of Diskettes Drives) :

بإمكان الحاسب IBM استخدام أحد أنواع السواقات التالية :

- * وجه واحد بسعة تخزين 160/180 كيلو بايت .
- * وجهين بسعة تخزين 320/360 كيلو بايت .
- * وجهين بسعة تخزين 720 كيلو بايت .
- * كثافة عالية بسعة تخزين 1.2 ميغا بايت .

أنواع الاسطوانات (Types Of Diskettes) :

بإمكانك استخدام أحد الأنواع التالية لحفظ البيانات :

- * وجه واحد بسعة تخزين 160/180 كيلو بايت .
- * وجهين بسعة تخزين 320/360 كيلو بايت .
- * وجهين بسعة تخزين 720 كيلو بايت .
- * كثافة عالية بسعة تخزين 1.2 ميغا بايت .

ان الاسطوانة ذات الوجه الواحد تحوي 40 حلقة وكل حلقة تحوي 8 - 9 قطاعات وبامكانها تخزين 160K/180K كيلو بايت (1 كيلو بايت = 1024 بايت) .

الاسطوانة ذات الوجهين تحوي أيضاً 40 حلقة و 8 - 9 قطاعات في كل حلقة وسعة تخزين 320/360 كيلو بايت (1 كيلو بايت = 1024 بايت) .

الاسطوانة ذات الوجهين بسعة تخزين 720 كيلو بايت تحوي 80 حلقة وكل حلقة تحوي 9 قطاعات .

الاسطوانة ذات الكثافة العالية هي ذات وجهين وتحوي 80 حلقة في كل وجه وكل حلقة تحوي 15 قطاع بسعة تخزين 1.2 ميغا بايت (1 ميغا بايت = 1.048.576 بايت) .

التوافق بين الاسطوانات وسواقات الاسطوانات (Dskettes and Drive Compatability) :

لا يسمح في بعض عمليات القراءة والكتابة باستعمال اسطوانة مغايرة لنوع السواقة، ويقوم القسم التالي بوصف الاسطوانات والسواقات المتوافقة .

- السواقات ذات الوجه الواحد وبسعة تخزين 160/180 كيلو بايت :

يمكن استخدام الاسطوانات ذات الوجه الواحد فقط وذات السعة 160/180

- السواقات ذات الوجهين وبسعة تخزين 320/360 كيلو بايت :

بامكانك استخدام أحد النوعين التاليين :

* 160/180 كيلو بايت لاسطوانات ذات وجه واحد .

* 320/360 كيلو بايت لاسطوانات ذات وجهين .

- السواقات ذات الوجهين بسعة تخزين 720 كيلو بايت .

يمكنك استخدام الاسطوانات ذات قدرة تخزين 720KB ثنائية الوجه .

- السواقات ذات الكثافة العالية (High Capacity Drives)

* أحادية الوجه بسعة تخزين 160/180 كيلو بايت

* ثنائية الوجه بسعة تخزين 320/360 كيلو بايت

* عالية الكثافة ثنائية الوجه

هام : إذا استخدمت أحد هذه الاسطوانات مع السواقة ذات الكثافة العالية فلن تستطيع استخدام هذه الاسطوانات مع السواقات ذات السعة 160/180 أو 320/360 . إذا كنت تملك أكثر من نوع واحد من سواقات الاسطوانات فيجب أن تأخذ بعين الاعتبار التوافق في أوامر DOS أثناء استخدام الاسطوانة. فمثلاً ان الأمر FORMAT يحوي على قسم يدعى بالتوافقية مع أمر التوصيف FORMAT . اقرأ هذا القسم قبل استخدام هذا الأمر .

تهيئة الاسطوانات (FORMAT) :

قبل استخدامك اسطوانة جديدة يجب أن تجري عملية التوصيف عليها بواسطة الأمر FORMAT . لاحظ بأن هذا الأمر يستخدم مرة واحدة فقط عند استخدامك اسطوانة جديدة وليس في كل مرة .

حماية الاسطوانات من الكتابة (Write Protected Diskettes) :

في بعض الأحيان يقوم المستخدم بالكتابة على الاسطوانة أو توصيفها دون الانتباه الى ذلك، وبالتالي من المحتمل ضياع البيانات الموجودة عليها. وهذا يتطلب في البرامج المهمة حماية الاسطوانة من الكتابة عليهما مع امكانية قراءتها. ويمكن اجراء الحماية كما يلي :

توجد لصقات مع الاسطوانة يتم وضعها على الفتحة الموجودة على طرف الاسطوانة وبذلك تتم عملية الحماية .

ملاحظة : (لا تستعمل شريط الالتصاق العادي بل اللصاقات الفضية) .

* توجد بعض الاسطوانات التي لا تحتوي على فتحة جانبية (مثل اسطوانة Dos) هذا يعني أن الاسطوانات محمية ضد الكتابة ولا يمكن الكتابة عليها .

* في الوجه الثاني لاسطوانات 720 كيلو بايت وفي الزاوية اليمنى السفلية توجد نافذة الحماية. عندما تحول القطعة البلاستيكية الى وضعية فتح النافذة فهذا يعني أن الاسطوانة محمية بينما عند اغلاق النافذة تصبح الاسطوانة جاهزة للكتابة .

الطرفيات التي يتعامل معها نظام DOS :

أسماء الطرفيات :

إن من بعض الطرفيات التي يتعامل معها الحاسب - شاشة الاظهار - السواقات المرنة - السواقات الصلبة - الطابعات ويمكن تعريف نظام التشغيل عليها عن طريق اسم كل طرف .

يبين الجدول التالي بعض هذه الطرفيات مع الاسم الخاص بكل منها .

نوع الجهاز الطرفي	الاسم الرمزي المستخدم
الطابعة	LPT 1 او PRN
الشاشة	Con (console)
سواقات الاسطوانات	A — Z الاحرف

نوع النظام المستعمل :

يمكن للحاسب استعمال السواقات المرنة أو السواقة الصلبة أو كلاهما معاً، كما يتم تعريف كل سواقة بواسطة الاحرف الأبجدية A - Z حيث يقوم نظام Dos بتسمية الاسطوانات في جهازك بدأ بالحرف A وبعد ذلك يقوم بتسمية الاسطوانة الصلبة بأحرف أخرى .

سواقة اسطوانات واحدة (One Diskettes Drive) :

مثلاً : إذا كان لديك حاسب بسواقة واحدة فإن النظام يتعرف عليه بالحرفين A و B عوضاً عن الوجود الفعلي لسواقتي الاسطوانات. وعند استعمالك لأمر ما مثل Copy مع سواقة واحدة فإن النظام يقوم بطباعة رسائل على الشاشة لتحديد الاسطوانات المطلوبة. لذا يجب عليك تبديل الاسطوانات في كل مرة يطلب الحاسب منك ذلك .

سواقتي اسطوانات (Two Diskette Drive)

إذا كان لجهازك سواقتي اسطوانات فإن الأول يتم الاشارة اليها بالحرف A والسواقة الثانية بالحرف B ، كما يمكن استعمال الاسطوانات في السواقتين معاً وفي هذه الحالة لا يتطلب الحاسب تبديل الاسطوانات .

سواقة مرنة واحدة وسواقة صلبة :

إذا كان لحاسبك سواقة مرنة وسواقة صلبة فإنه يتم تعريف السواقة المرنة بالحرفين A و B والسواقة الصلبة بالحرف C .

الاسطوانات الصلبة (Fixed disk drive) :

يجب تهيئة الاسطوانة قبل استخدام النظام في التعامل معها، ويتم ذلك عن طريق الامر (Select) .

وظائف المفاتيح :

ان لوحة المفاتيح الخاصة بالحاسب تتضمن بعض المفاتيح ذات الوظائف الخاصة وبسبب الاختلاف بين لوحات المفاتيح فسيتم شرح وظيفة كل مفتاح، كما يبين الجدول التالي أكثر المفاتيح شيوعاً. أما من أجل الحصول على كل وظيفة فإن المفاتيح الواردة في العمود الخاص بالحاسب PC يجب

أن تضغط مع بعضها في نفس الوقت فمثلاً من أجل إعادة شحن النظام يجب الضغط على (DE +ALT+CTRL) في نفس الوقت .

الوظيفة	الحاسب PC	الحاسب PCjr
إيقاف العمل	Ctrl-Break	Break ثم Fn
إيقاف الشاشة	Ctrl-Numlock	Pause ثم Fn
طباعة الشاشة	Shift-PrtSc	PrtSc ثم Fn
طباعة ما تكتبه	Ctrl-PrtSc	Echo ثم Fn
إعادة شحن النظام	Ctrl-Alt-Del	Del-Alt-Ctrl

* ادخال أمر الى الحاسب (Enter) بعد أن تكتب الأمر المطلوب اضغط المفتاح Enter لاخبار الحاسب برغبتك بتنفيذ الأمر .

* الغاء أمر يتم تنفيذه (Break) يتم بذلك إيقاف تنفيذ أمر ما .

* اصلاح خطأ كتابي (Back Space) : يتم عن طريق ذلك نقل الدالة مسافة واحدة الى الخلف لالغاء الحرف المكتوب ومن ثم إعادة الكتابة من جديد قبل الضغط على Enter .

* إيقاف الشاشة (Pause Screen) :

يتم بذلك إيقاف ظهور البيانات على الشاشة من أجل التمكن من قراءة البيانات الحالية وذلك عندما تظهر البيانات بسرعة عالية .

* طباعة الشاشة (Print Screen) :

يتم بذلك نقل ما هو مكتوب على الشاشة الى الطابعة ويجب أن تكون الطابعة جاهزة قبل استخدام هذا الأمر وتدعى هذه العملية بـ Hard Copy.

* طباعة ما تكتبه (Print Echo) :

طباعة كل سطر تكتبه كما يظهر على الشاشة، وذلك بعد الضغط على مفتاح

Enter . أما من أجل إلغاء هذا العملية فأعد ضغط المفاتيح المتحركة بهذه العملية مرة أخرى .

إعادة شحن النظام (System Reset) : -

يتم بذلك إعادة تحميل نظام التشغيل مرة أخرى .

مفاتيح التحرير الخاصة بنظام التشغيل Dos :

عندما تقوم بكتابة سطر ثم تضغط Enter فإن نظام Dos يحتفظ بنسخة عن هذا السطر في ذاكرة مؤقتة (buffer) وبالتالي يمكن طلب هذا السطر مرة أخرى دون الحاجة الى كتابته من جديد .

ان مفاتيح التحكم تستخدم لهذه الغاية ويقوم الجدول التالي بشرح مهمة كل مفتاح .

اسم المفتاح	الوظيفة
Ins	السماح بإضافة احرف ضمن جملة ما
Del	إلغاء حرف واحد الى يمين الدالة
Esc >	نقل السطر الحالي مع إبقاء الذاكرة المؤقتة كما هي .
F1 او >-	نقل حرف واحد من الذاكرة المؤقتة الى الشاشة .
F2	نقل كافة الاحرف من الذاكرة المؤقتة الى الشاشة وإلغاء حرف معين .
F3	نقل كافة الاحرف من الذاكرة المؤقتة الى الشاشة
F4	إلغاء كافة الاحرف وإلغاء حرف معين (F4 يعاكس عمل F2)
F5	يعتبر السطر الذي حررته بأنه الموجود في الذاكرة المؤقتة

استعمال مفاتيح التحرير في نظام Dos :

لنفرض أنك كتبت سطر معين . ثم ضغطت على مفتاح Enter

ان الذاكرة المؤقتة تحتوي الآن على السطر 1 dir file إذا ضغطت F1 فيتم طباعة الحرف التالي d على الشاشة وهكذا، فكل مرة تضغط F1 يتم طباعة الحرف التالي فاذا ضغطت F1 تسعة مرات فيتم طباعة : 1 dir file . ثم اضغط الآن المفتاح Enter . إذا ضغطت مفتاح F2 ثم كتبت الحرف 1 فيتم عرض ما يلي :

لاحظ بأنه تمت طباعة كافة الأحرف ولغاية الحرف 1 . إذا ضغطت المفتاح F3 فسيتم طباعة ما يلي Die file ، إذا ضغطت المفتاح F4 ثم الحرف r فان الذاكرة المؤقتة ستحتوي على السطر r وبالضغط على المفتاح F3 لعرض السطر يتم طباعة ما يلي : r file .

ما هو الملف :

يحتوي الملف على مجموعة من المعلومات والبيانات تم وضعها على الاسطوانة المرنة أو الصلبة . بإمكانك مقارنة ملفين موجودين على اسطوانتين أو اسطوانة واحدة، أي أن الاسطوانة المرنة والصلبة يمكن اعتبارهما كخزانة يتم وضع المصنفات فيها وكل مصنف يدعى بالملف. إن لكل ملف اسم مشابه للاسم الموجود على المصنف وعندما ترغب في استدعاء هذا الملف فيكفي تحديد اسم هذا الملف وبالتالي إمكانية التعامل مع البيانات في هذا الملف .

كيفية تسمية الملفات :

بإمكانك الاحتفاظ بمجموعات مختلفة من الملفات على نفس الاسطوانة. ولكن يجب أن يكون لكل ملف اسم مستقل عن الآخر بيد أنه يمكنك استعمال نفس الاسم على اسطوانتين مختلفتين .

ان اسم الملف يتألف من الاسم الأصلي Filename يمكن وضع اسم فرعي (بشكل اختياري) Extension فمثلاً :



إذا استعملت الأمر Dir فسيتم طباعة هذا الملف على الشاشة ، لاحظ بأن الفاصل (.) لا يظهر على الشاشة ولكنك يجب أن تستعمل هذا الفاصل عند الملف المطلوب .

اسماء الملفات (Files Names) :

إن اسم الملف حسب النظام يتألف من حرف واحد ولغاية ثمانية أحرف وعند كتابتك لاسم الملف تأكد من عدم استخدامك لرموز غير مسموح بها وفيما يلي هذه الرموز غير المسموحة : " , ; = + < > : [] / ' .

أسماء الملفات الفرعية (Extension) :

يمكن استعمال اسم يدعى Extension مع اسم الملف الأصلي (بشكل اختياري) وهذا الاسم يجب أن لا يزيد عن ثلاثة أحرف ويجب وضعه مباشرة بعد الاسم الأصلي وأن الرموز غير المسموحة السابقة هي نفسها غير المسموحة في الأسماء الفرعية .

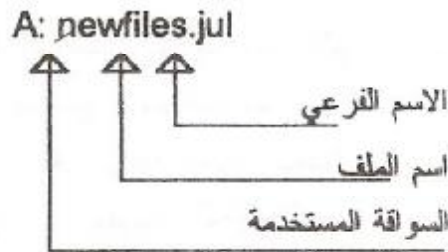
هام : إذا كان اسم الملف يحوي على اسم فرعي فيجب عليك استخدام الاسم الفرعي في كل مرة تطلب فيها هذا الملف .

إن الاسم الجيد للملف يساعدك على تذكر ماهية المعلومات الموجودة على

الملف كون هذا الملف يحوي برنامجاً أو بيانات . فمثلاً ان الاسم ADDRLIST.BAS هو اسم جيد لبرنامج بلغة البيسك ويدل على وظيفته والتي هي طباعة عناوين معينة .

مواصفات الملف (Files Specifications) :

من أجل ايجاد الملف المطلوب وبالإضافة الى اسم الملف الأصلي فإنه يجب تحديد سواقة الاسطوانات المطلوب البحث فيها وبالتالي تحديد - اسم السواقة - اسم الملف - الاسم الفرعي مثلاً .



حيث ان اسم السواقة مع الرمز (:) يدعيان drive Specifier ويعرف اسم السواقة نظام Dos على السواقة المستخدمة ويجب دوماً استعمال الرمز (:) بعد اسم السواقة. كما يجب وضع اسم الملف والاسم الفرعي مباشرة بعد الرمز (:) .

ملاحظة : لا تضع أي فراغ بين الأسماء الثلاثة .

هام : إذا كان لديك اطلاع أو معرفة بنظام Dos فما عليك إلا استخدام الأمر SELECT حسب نوع النظام الذي تستخدمه، من حيث عدد الاسطوانات المرنّة والصلبة .

اما إذا كان نظام الـ Dos جديداً فيجب أن تتبع ما يلي :

* كيفية وضع الاسطوانات .

* استخدام الأمر SELECT من أجل :

- تحديد لوحة المفاتيح والتاريخ والتوقيت المستخدمين .
- نسخ اسطوانة اضافة من اسطوانة النظام .
- نقل ملفات Dos الى الاسطوانة الصلبة لأول مرة .
- * معرفة كيفية استخدام النظام كل يوم .
- * معرفة كيفية تحديد سواقة الاسطوانات المستخدمة .

تحميل نظام التشغيل Dos :

ان عملية تحميل نظام التشغيل تعني نقل أوامر نظام التشغيل من اسطوانة النظام الى ذاكرة الحاسب. كما يخبرك نظام Dos بجاهزيته عندما يطلب منك ادخال الوقت والتاريخ وبعد ذلك يصبح النظام جاهزاً لتلقي الاوامر بالقيام بعمل ما يريده المشغل. وعند القيام بتحميل نظام Dos لأول مرة يجب اتباع الارشادات الواردة في تعليمة SELECT .

كيفية وضع الاسطوانات (Inserting Diskettes)

قبل أن تبدأ بنظام Dos لا بد أن تعرف كيف تضع الاسطوانات في سواقة الاسطوانات ويجب اتباع الطريقة التي تتوافق مع نوع الاسطوانات المستخدمة :

وضع اسطوانات 180/160 و 360/320 وذات الكثافة العالية :

- 1 - أخرج الاسطوانة من الغلاف الورقي .
- 2 - افتح باب سواقة الاسطوانات .
- 3 - ضع الاسطوانات بلطف داخل المجرى المخصص لها بحيث تكون اللصاقة التي تحمل اسم الاسطوانة نحو الأعلى .
- 4 - تأكد من صيغة الاسطوانة ثم أغلق الباب .

وضع الاسطوانة ذات سعة 720 كيلو بايت :

- 1 - ضع الاسطوانة في المجرى المخصص لها بحيث يكون الوجه الذي يحوي على الاسم نحو الأعلى .
- 2 - اضغط بلطف على القسم المتبقي من الاسطوانة وادفعها نحو الداخل لحين سماعك (صوت الغلاق) وبالتالي وصلت الاسطوانة الى موقعها المحدد .
- 3 - من أجل إزالة الاسطوانة اضغط على المفتاح الصغير الموجود على القسم الأيمن السفلي لسوافة الاسطوانات .

استخدام نظام Dos لأول مرة :

إذا كنت تستخدم النظام لأول مرة فيجب عليك اتباع التعليمات الواردة في الأمر SELECT من أجل اخبار نظام Dos عن نوع لوحة المفاتيح المستخدمة والتاريخ والوقت إن الرمز الدال على لوحة المفاتيح يخبر نظام Dos عن شكل توزيع الأحرف على اللوحة فمثلاً من أجل استخدام لوحة مفاتيح أمريكية قم بوضع الرمز الخاص بالولايات المتحدة . إن رمز الدولة يخبر الحاسب عن التاريخ والوقت المستخدم وكذلك نوع العملة المستخدمة ورمزها ورمز الفاصلة العشرية فمثلاً اختيارك لأمريكا يؤدي الى الأمور التالية :

* شكل التاريخ يكون كما يلي :

MM/DD/YY أي سنة / يوم / شهر .

* شكل التوقيت يكون كما يلي :

HH:MM:SS أي ثانية : دقيقة : ساعة .

* شكل ورمز العملة المستخدمة \$ دولار .

* الفاصلة العشرية المستعملة هي (.) .

استخدام أحد الاختيارات في الأمر SELECT لاختبار الحاسب عن لوحة المفاتيح مع العلم انه في البداية يجب أن تختار الرقم الخاص بالدولة من

الجدول المخصص لذلك وبعد أن تنتهي من الأمر SELECT يجب عليك نسخ اسطوانة نظام وهذا ما يقوم به الأمر SELECT بشكل آلي لتحصل على اسطوانة تحوي الرمز الخاص باللوحة المستخدمة .

إذا كان لديك اسطوانة صلبة وعند انتهاءك من عملية SELECT يجب نسخ اسطوانة النظام الى الاسطوانة الصلبة والتي تحوي على الرموز المختارة، ويمكنك بعد ذلك تحميل نظام التشغيل من الاسطوانة الصلبة مباشرة .

ما يلزم لتنفيذ الأمر SELECT :

* اسطوانة نظام التشغيل Dos .

* اسطوانة فارغة أخرى اذا كان لجهازك نظام اسطوانات مرنة فقط .

الاسطوانة التي تحوي على الرمز الخاص بلوحة المفاتيح المستخدمة (إذا كان الرمز الخاص موجوداً على اسطوانة Dos) .

هام : يجب ألا تكون الاسطوانة الفارغة 160/180 كيلو بايت ذات وجه واحد. وبعد أن عرفت الرمز الخاص بلوحة المفاتيح اختر الأمر SELECT الذي يتوافق مع الحاسب الذي تعمل به (سواقة مرنة واحدة - سواقتين مرتنتين - سواقة صلبة) وذلك من لائحة الخيارات .

الأمر SELECT في جهاز ذي سواقتين :

أمر لوحة المفاتيح موجود على اسطوانة Dos

اتباع الخطوات التالية لاختيار رمز لوحة المفاتيح حسب الدول السابقة :

1. ضع اسطوانة النظام Dos في السواقة A

2. اضغط المفاتيح التالية معاً (CTRL-ALT-DEL) من أجل تحميل النظام وتجاهل ادخال الوقت والتاريخ وذلك بالضغط على Enter مرتين

3. بعد اشارة الجاهزية للنظام (>A) اكتب ما يلي : Select XXX YY

4. اضغط Enter فتظهر الرسالة التالية :

SELECT is used to install Dos the first time .

SELECT erases everything on the specified target and
then installs Dos .

Do you want to continue (Y/N) ? Y

5. في حال وجود معلومات سابقة على الاسطوانة فلا تتابع واضغط الحرف N من
أجل انتهاء تعليمة SELECT

6. إذا أردت المتابعة بالأمر SELECT أي تشكيل نسخة عمل للنظام Dos فاضغط
Enter .

7. ستري الآن الرسالة التالية :

Insert new diskette for drive B :

and strike ENTER when ready

8. ضع الاسطوانة الفارغة في السواعة B ثم اضغط Enter سيبدأ نظام Dos
بتوصيف الاسطوانة وتظهر على الشاشة عدد الدورات التي يقوم بها
الحاسب وتستمر هذه العملية لحين الانتهاء من التوصيف ثم تظهر الرسالة
التالية :

Format Complete

System transferred

ويظهر أيضاً سعة التخزين المتاحة ثم الرسالة التالية :

Format another (Y/N) ?

9. اضغط الحرف N

10. اضغط Enter

عندما ترى الرسالة التالية فإن نظام Dos يقوم بقراءة كافة الملفات الموجودة على اسطوانة النظام Dos

Reading source file (s) ...

وتظهر على الشاشة أسماء الملفات التي يتم نسخها من الاسطوانة الأصلية الى الاسطوانة الجديدة .

عند ظهور اشارة الجاهزية >A فهذا يدل على ان الأمر SELECT قد انتهى. إذا كنت تتعامل مع نظام Dos لأول مرة فعليك اتباع ما يلي :

1 - ضع اسطوانة Dos في السواقة A .

2 - اضغط (Ctrl- Alt -Del) من أجل اعادة شحن النظام وتجاهل ادخال التاريخ والوقت وذلك بالضغط على Enter مرتين .

3 - بعد ظهور اشارة الجاهزية A اكتب ما يلي : select xxx yy

ان رمز المفاتيح يجب أن يكون موجوداً على اسطوانة أخرى غير اسطوانة النظام .

4 - اضغط Enter فتلاحظ ظهور الرسالة التالية :

SELECT is used to install DOS the first time .

SELECT erases everything on the specified target and then installs DOS .

Do you want to continue (Y/N) ? Y

5 - في حال وجود معلومات سابقة على الاسطوانة فلا تتابع واضغط N لانتهاء العمل

6 - إذا رغبت في المتابعة بالأمر SELECT (أي تشكيل ملف عمل لنظام dos) . اضغط Enter فترى الرسالة التالية :

IS KEY BYY . COM on another diskette (Y/n) ?

تذكر بأنه عوضاً عن YY فسوف تحصل على رمز لوحة المفاتيح الذي
كتبته اثناء تنفيذ الأمر SELECT في بداية العملية .

7 - اضغط Y .

8 - سترى الآن ظهور الرسالة التالية :

insert new diskette for drive B :

and strike Enter when ready

9 - ضع الاسطوانة الفارغة في السواقة B ثم اضغط Enter .

سيبدأ الحاسب بتوصيف الاسطوانة الجديدة مع طباعة عدد الدورات التي
يقوم بها وتستمر هذه العملية لحين التوصيف. وتظهر عندها الرسالة
التالية :

Format complete

system transferred

كذلك تظهر سعة التخزين المتاحة على هذه الاسطوانة ثم الرسالة التالية :

Format another (Y / N) ?

10 - اكتب N

11 - اضغط Enter

تظهر الرسالة التالية على الشاشة :

Insert KEYBYY .com diskette in drive a:

strike any key when ready

12 - أخرج اسطوانة النظام DOS من السواقة A وضع مكانها الاسطوانة التي
تحتوي على الرمز الخاص بلوحة المفاتيح المستخدمة ثم اضغط على أي مفتاح
لنقل الرمز الخاص الى الاسطوانة الجديدة في المحرك B ثم تظهر الرسالة
التالية :

INSERT DOS DISKETTE IN DRIVE A:

STRIKE ANY KEY WHEN READY

13 - أخرج الاسطوانة من السواقة A وضع مكانها اسطوانة نظام DOS ثم اضغط على أي مفتاح. لتلاحظ بعدها ظهور الرسالة التالية :

Reading source file (s) ...

يتم قراءة الملفات من اسطوانة النظام ونقلها الى الاسطوانة الجديدة مع طباعة أسماء الملفات التي يتم نقلها ويعني ظهور إشارة الجاهزية >A انتهاء الأمر SELECT .

استخدام الأمر SELECT مع السواقة الصلبة :

قبل أن تبدأ باستخدام السواقة الصلبة عليك تشكيل مكان خاص عليها لتتعامل مع نظام DOS والخطوات اللازمة لذلك هي :

* استخدام الأمر FDISK لتشكيل مكان لـ DOS

* استخدام الأمر SELECT لتعريف لوحة المفاتيح (وهذا الأمر يتعلق بتوصيف الاسطوانة الصلبة ونقل كافة ملفات النظام إليها) .

* استخدام الأمر COPY لنقل الملفات من اسطوانة SUPPLEMENTAL الى السواقة الصلبة .

استخدام الأمر FDISK لتشكيل مكان لعمل نظام DOS :

لا يمكنك استعمال السواقة الصلبة قبل تهيئتها بواسطة برنامج خاص هو FDISK ، وإذا حاولت استخدام السواقة الصلبة قبل ذلك فتظهر الرسالة التالية :

Invalid drive specification

ملاحظة : يمكنك استخدام الأمر FDISK من أجل أمور أخرى مثل (إلغاء

مكان عمل DISK تغيير موقع العمل - طباعة معلومات حول هذا المكان) .
أما اذا كنت ترغب في تهيئة السواقة فقط دون تفاصيل اضافية فاتبع
الخطوات التالية للبدء في برنامج fdisk :

1 - ضع اسطوانة النظام في السواقة A وبعد اشارة الجاهزية >A اكتب :

A>fdisk

2 - اضغط Enter .

ان لائحة الخيارات الاساسية تظهر على الشاشة ب 4 او 5 خيارات حسب
عدد السواقات الصلبة المستخدمة : هذه الخيارات هي :

IBM Personal Computer

Fixed Disk setup program version 3.20

(C) Copyright IBM Corp. 1983 .1986

FDISK option

current fixed disk drive : 1

choose one of the following :

1. Create DOS Partition
2. Change Active Partiton
- 3- Delete DOS partition
- 4- Dispaly Partition date
- 5- Select Next Fixed Disk Drive

Enter Choice : [1]

Press ESC to return to dos

- حيث : 1 - تشكيل مكان لعمل DOS .
2 - تغيير مكان عمل DOS .
3 - الغاء مكان عمل DOS .
4 - طباعة معلومات حول مكان عمل DOS .
5 - اختيار السواقة الصلبة التالية .
3. اختر الرقم 1 من أجل تشكيل مكان لـ DOS وذلك بالضغط على Enter ، فتظهر الرسالة التالية :

Create DOS Partition

current fixed disk drive : 1

Do you wish to use the entire fixed disk

for DOS (Y/N)? (y)

4. اضغط Y ثم Enter فترى ظهور الرسالة التالية :

System will now restart

Insert DOS diskette in drive A:

Press any key when ready

يجب أن تكون اسطوانة نظام DOS موجودة في السواقة A.

5. اضغط أي مفتاح .

6. ادخل الوقت والتاريخ .

لقد شكلت الآن مكان على الاسطوانة الصلبة من أجل نظام DOS ، ويجب استعمال الأمر SELECT من أجل تحديد رمز لوحة المفاتيح المستخدمة ثم نسخ كافة ملفات النظام الى الاسطوانة الصلبة .

استخدام الأمر SELECT :

تحذير : أثناء تنفيذ الأمر SELECT فإنه يتم توصيف الاسطوانة الصلبة فإذا كانت لديك أية بيانات على الاسطوانة الصلبة فإنها ستزول .

أمر لوحة المفاتيح على الاسطوانة DOS

اكتب هذه الرموز مع الأمر بالشكل التالي :

1. بعد ظهور A> اكتب :

select c: xxx yy

2. اضغط Enter فتظهر الرسالة التالية :

SELECT is used to install DOS the first
time .SELECT erases everything on the
specified target and then installs DOS
DO you want to continue (Y/N) ? Y

3. في حال وجود معلومات على الاسطوانة الصلبة فلا تتابع واكتب N لانتهاء العمل
بالأمر SELECT

4. إذا أردت المتابعة أي تهيئة DOS لأول مرة :

فاضغط Enter

1. ضع اسطوانة النظام Dos في السواقة A (Diskcopy هو أمر خارجي)

2. تأكد من ظهور إشارة الجاهزية A>

3. اكتب : Diskcopy a: b:

4. اضغط Enter فتظهر الرسالة التالية :

Insert SOURCE diskette in drive A:

press any key when ready ...

5. أخرج اسطوانة النظام من السواقة A

6. ضع الاسطوانة الأصلية في السواقة A

7. اضغط أي مفتاح .

اثناء عملية القراءة يتم اضاءة الضوء وبعد الانتهاء تظهر الرسالة التالية :

Insert TARGET diskette in drive A:

press any key when ready ...

8. أخرج الاسطوانة الأصلية من السواقة A

9. ضع الاسطوانة النسخة في السواقة A

10. اضغط أي مفتاح .

ملاحظة : قد تحتاج الى تكرار عملية تبديل الاسطوانة تبعاً لحجم الذاكرة المستخدمة في الجهاز حيث يخبرك الحاسب ما يجب فعله. استمر في التبديل حتى ظهور الرسالة التالية :

Copy another diskette (Y/N) ?

11. اكتب احدى التعليمات التالية :

* N من أجل انتهاء العمل بالأمر Diskcopy حيث يطلب منك وضع اسطوانة تحوي الملف Command. Com وعندها يجب عليك وضع اسطوانة Dos ، ثم اضغط أي مفتاح فتظهر اشارة الجاهزية A> وتكون عملية النسخ قد انتهت.

* Y من أجل تكرار عملية النسخ لاسطوانة أخرى ، عليك في هذه الحالة اتباع الارشادات بدءاً من الخطوة 6 .

الأمر Diskcopy مع سواقتي اسطوانات :

من أجل استخدام هذا الأمر يجب عليك تذكر الاسطوانة الأصلية والاسطوانة النسخة .

1. ضع اسطوانة النظام Dos في السواعة A (Diskcopy هو امر خارجي) .

2. تأكد من ظهور اشارة الجاهزية A>

3. اكتب : Diskcopy a: b:

4. اضغط Enter فتظهر الرسالة التالية :

Insert SOURCE diskette in drive A:

Insert TARGET diskette in drive B:

press any key when ready ...

5. اخرج اسطوانة النظام dos من السواعة A

6. ضع الاسطوانة الاصلية في السواعة A

7. ضع الاسطوانة النسخة في السواعة B

8. اضغط أي مفتاح .

يتم إضاءة الضوء أثناء عملية نقل الملفات من اسطوانة الى أخرى ثم تظهر
الرسالة التالية :

Copy another diskette (Y/N) ?

9. اكتب :

* N لانتهاء العمل بالأمر Diskcopy فتظهر بعد ذلك اشارة الجاهزية A>

وتكون عملية النسخ قد تمت ثم اخرج الاسطوانة التي تم النسخ عليها .

* Y لنسخ اسطوانة أخرى في هذه الحالة يجب عليك اتباع الارشادات بدءاً من
الخطوة 6 .

مقارنة الاسطوانات (Comparing Diskettes) :

استخدم الأمر Diskcomp لمقارنة المعلومات الموجودة على اسطوانة مع

المعلومات الموجودة على اسطوانة أخرى. إن استخدام الأمر Diskcomp عادة

يتم بعد استعمال الأمر Diskcopy للتأكد من صحة عملية النسخ ومطابقة البيانات بين الاسطوانتين .

ماذا تحتاج لمقارنة اسطوانتين :

* اسطوانة نظام Dos

* الاسطوانة الأصلية التي تود مقارنتها معها .

* الاسطوانة النسخة التي تود مقارنة محتوياتها .

توافقية الأمر Diskcomp :

يستخدم هذا الأمر مع توافقية محددة بين الاسطوانات وسواقات الاسطوانات .

الأمر Diskcomp مع سواقة اسطوانات واحدة :

لاستخدام هذا الأمر تذكر بأن الاسطوانة الأصلية هي الاسطوانة الأولى والاسطوانة النسخة هي الاسطوانة الثانية :

1. ضع اسطوانة النظام في السواقة A

2. تأكد من ظهور اشارة الجاهزية A>

3. اكتب : Diskcomp a: b:

4. اضغط Enter فتظهر الرسالة التالية :

Insert FIRST diskette in drive A:

press any key when ready ...

5. أخرج اسطوانة النظام من السواقة A

6. ضع الاسطوانة الأصلية في السواقة A

7. اضغط أي مفتاح .

يضيء الضوء وتبدء عملية قراءة الملفات ثم تظهر الرسالة التالية :

Insert SECOND diskette in drive A:

press any key when ready ...

8. أخرج الاسطوانة الاصلية من السواقة A

9. ضع الاسطوانة النسخة في السواقة A

10. اضغط أي مفتاح .

ملاحظة : قد تحتاج الى تكرار عملية تبديل الاسطوانات، وذلك تبعاً لحجم الذاكرة المستخدمة في الجهاز استمر في التبديل حتى ظهور الرسالة التالية :

Compare OK

Compare more diskette (Y/N) ?

ملاحظة : في حال عدم المطابقة بين الاسطوانة الاصلية والنسخة فاعد عملية النسخ بواسطة الأمر Diskcopy ثم عملية المقارنة Diskcompe ..

11. اكتب :

* N لانهاء العمل بالأمر Diskcomp فتظهر اشارة الجاهزية >A وتكون قد تمت عملية المقارنة بنجاح .

* Y لمقارنة اسطوانات أخرى، اتبع الارشادات مرة أخرى من الخطوة 6 .

ملاحظة : إذا حصلت على الرسالة التالية :

Insert COMMAND.COM diskette in drive X:

and strik any key when ready ...

قبل ظهور اشارة الجاهزية >A فضع اسطوانة النظام Dos في السواقة A ثم اضغط أي مفتاح. عندها تظهر اشارة الجاهزية .

الامر Diskcomp مع سواقتي اسطوانات :

لاستخدام هذا الامر تذكر بأن الاسطوانة الاصلية هي الاسطوانة الاولى
والاسطوانة النسخة هي الاسطوانة الثانية :

1. ضع اسطوانة النظام Dos في السواعة A (Diskcomp هو امر خارجي) .

2. تأكد من ظهور اشارة الجاهزية A>

3. اكتب : Diskcomp a: b:

4. اضغط Enter فتظهر الرسالة التالية :

Insert First diskette in drive A:

Insert SECOND diskette in drive B:

press any key when ready ...

5. اخرج اسطوانة النظام من السواعة A

6. ضع الاسطوانة الاصلية في السواعة A

7. ضع الاسطوانة النسخة في السواعة B

8. اضغط أي مفتاح .

يضيء الضوء الأحمر اثناء عملية المقارنة وبعد ذلك تظهر الرسالة التالية :

Compare Ok

Compare more diskettes (Y/N) ?

ملاحظة : في حال عدم التطابق بين الاسطوانتين أعد عملية النسخ بواسطة
الامر Diskcopy ثم عملية المقارنة مرة أخرى Diskcomp .

9. اكتب :

* N لانتهاء العمل بالمقارنة فتظهر عندها اشارة الجاهزية A> وتكون عملية
المقارنة قد تمت بنجاح .

* Y لمقارنة اسطوانات أخرى أعد الارشادات من الخطوة 6 .

ملاحظة : إذا حصلت على الرسالة التالية :

Insert COMMAND.COM diskette in drive X:

and strik any key when ready ...

وذلك قبل ظهور اشارة الجاهزية >A هنا يجب أن تضع اسطوانة النظام
Dos في السواقة A ثم اضغط أي مفتاح . عندما تظهر اشارة الجاهزية >A.

نسخ ملف (Copying a File) :

استخدم الأمر Copy عندما ترغب في نسخ ملف واحد وليس اسطوانة
بالكامل .

ماذا تحتاج لنسخ ملف :

* الاسطوانة التي تحوي على الملف المطلوب نسخه .

* الاسطوانة التي سيتم نسخ الملف عليها .

إذا كان نظام Dos قد جرى شحنة سابقاً فانك لن تحتاج الى اسطوانة
النظام لأن الأمر Copy هو أمر داخلي ويمكنك أيضاً اختيار اسم الملف
المطلوب حفظه على الاسطوانة الجديدة .

استخدام الأمر Copy مع وجود سواقة اسطوانات واحدة :

من السهل معرفة الاسطوانات الاصلية والاسطوانات النسخة، حيث B يرمز
للنسخة Backup وذلك عند ظهور رسالة تطلب وضع اسطوانة النسخة :

copy a:filename.ext b:

1. تأكد من ظهور اشارة الجاهزية >A

2. ضع الاسطوانة في السواقة A

3. اكتب : COPY A: FILENAME.EXT B:

حيث FILENAME هو اسم الملف المطلوب .

4. اضغط ENTER فتظهر الرسالة التالية :

Insert diskette for drive b:
and strike any when ready

5. اخرج الاسطوانة الاصلية من السواعة A

6. ضع الاسطوانة النسخة في السواعة A

7. اضغط اي مفتاح .

ملاحظة : قد تحتاج الى تكرار عملية تبديل الاسطوانات وذلك تبعاً لحجم
الذاكرة المستعملة في الجهاز. استمر في هذه العملية حتى ظهور الرسالة :

1 File (s) copied

A>

8. اخرج الاسطوانة النسخة .

استخدم الامر Copy مع سواقتي اسطوانات :

من السهل معرفة أي من الاسطوانتين الاصل وأي منهما النسخة إذا اعتبرت
أن B يرمز الى النسخة Backup وذلك عند ظهور رسالة تطلب وضع
اسطوانة النسخة :

1. تأكد من ظهور اشارة الجاهزية A

2. ضع الاسطوانة الصلبة في السواعة A

3. ضع الاسطوانة النسخة في السواعة B

4. اكتب : copy a:filename.ext b:

ضع عوضاً عن filename.ext اسم الملف المطلوب نسخه .

5. اضغط Enter :

يضيء الضوء لفترة أثناء عملية النسخ وحين الانتهاء تظهر الرسالة :

1 File (s) copied

A>

6. أخرج الاسطوانتين وضع لصاقة على اسطوانة النسخة بالمحتويات والتاريخ .

مقارنة ملف (Comparing file) :

استخدم الأمر COMP لمقارنة ملفين وفحص فيما اذا كانا متطابقين. وعادة تجري هذه العملية بعد استخدام الأمر Copy في نسخ ملف ما .

ماذا تحتاج لمقارنة ملفين :

1. اسطوانة نظام Dos

2. الاسطوانتين التي تحوي على الملفين المطلوب مقارنتهما .

استخدم الأمر Comp مع سواقة اسطوانات واحدة :

1. ضع اسطوانة النظام Dos في السواقة A (Comp هو أمر خارجي)

2. تأكد من ظهور اشارة الجاهزية A>

3. اكتب : Comp

4. اضغط Enter فتظهر الرسالة التالية :

Enter primary file name

5. أخرج اسطوانة نظام Dos في السواقة A

6. ضع الاسطوانة الأصلية في السواقة A

7. اكتب : a:filename.ext

(استبدل filename.ext باسمك الملف المطلوب)

8. اضغط Enter فتظهر الرسالة التالية :

Enter 2nd file name or drive id

9. اكتب : b: filename.ext

(استبدل filename.ext باسم الملف الثاني)

10. اضغط Enter فتظهر الرسالة التالية :

Insert diskette for drive B :

and strike any key wgen ready

11. أخرج الاسطوانة الأصلية من السواعة A

12. ضع الاسطوانة النسخة في السواعة A

13. اضغط Enter

14. اكتب :

* N إذا رغبت في انتهاء المقارنة وبعد ذلك تظهر اشارة الجاهزية >A وتمت عملية المقارنة بنجاح .

* Y إذا رغبت في مقارنة ملفات أخرى عندها يجب متابعة الارشادات السابقة بدءاً من الخطوة 6 .

ملاحظة : في حال ظهور الرسالة التالية :

Insert COMMAND.COM diskette in drive x :

and strike any key when ready

وذلك قبل ظهور اشارة الجاهزية >A . ضع اسطوانة نظام Dos في السواعة A ثم اضغط أي مفتاح فتظهر اشارة مرة أخرى .

استخدام الأمر Comp مع سواقتي اسطوانات :

1. ضع اسطوانة النظام Dos في السواقة A (Comp هو أمر خارجي)

2. تأكد من ظهور اشارة الجاهزية A>

3. اكتب : Comp

4. اضغط Enter فتظهر الرسالة التالية :

Enter primary file name

5. أخرج اسطوانة النظام من السواقة A

6. ضع الاسطوانة الاصلية في السواقة A

7. ضع الاسطوانة النسخة في السواقة B

8. اكتب : filename.ext

. (استبدل filename.ext باسم الملف المطلوب)

9. اضغط Enter .

10. اكتب : filename.ext

(استبدل filename.ext باسم الملف المطلوب)

11. اضغط Enter

يضيء الضوء الأحمر فترة من الزمن ، وحين انتهاء المقارنة تظهر الرسالة التالية :

File compare ok

Compare more file (Y/N) ?

12. اكتب :

* N من أجل انتهاء عملية المقارنة فتظهر اشارة الجاهزية A> . وتكون قد تمت عملية المقارنة بنجاح .

* Y من أجل مقارنة ملفات أخرى عد الى الخطوة 6 .

ملاحظة : في حال ظهور الرسالة التالية :

Insert COMMAND.COM diskette in drive x :

and strike any key when ready

وذلك قبل ظهور اشارة الجاهزية A> . فضع اسطوانة نظام Dos في السواقة A ثم اضغط أي مفتاح فتعود الاشارة للظهور .

فحص محتويات الاسطوانة (Dir) :

استخدم لهذا الغاية الأمر DIR لمعرفة أسماء الملفات الموجودة على الاسطوانة كما يلي :

1. تأكد من ظهور اشارة الجاهزية A>

2. ضع الاسطوانة المطلوب فحص محتوياتها في السواقة A

3. اكتب : dir a:

(من أجل فحص محتويات السواقة "C" اكتب : dir c:)

4. اضغط Enter

5. راقب الشاشة، حيث أن أول رسالة تظهر هي اسم الاسطوانة (في حال وجوده) ثم أسماء الملفات الموجودة في هذه الاسطوانة، وأسماء الفهارس المختلفة .

فحص محتويات الملف (Displaying the file) :

يستخدم لهذه الغاية الأمر Type حيث يسمح لك بفحص ما يحتويه ملف ما من بيانات وطباعتها على الشاشة. حيث أن ملفات النصوص تطبع على الشاشة بشكل واضح ومقروء بينما الملفات الأخرى مثل ملفات البرامج وما شابهها تطبع بشكل رموز غريبة والسبب أنها تحوي على أحرف لا تعود الى الأحرف الأبجدية .

ماذا تحتاج لفحص محتويات الملف :

- * الاسطوانة التي تحوي على الملف المطلوب فحصه .
- * معرفة الاسم الصحيح للملف المطلوب فحصه (استعمل dir مرة أخرى) .
- (إذا كان نظام Dos قد شحن سابقاً فلن تحتاج لاسطوانة النظام) .

الأمر type :

1. تأكد من ظهور اشارة الجاهزية >A
2. ضع الاسطوانة التي تحوي الملف المطلوب في السواقة A .
3. اكتب : type a:filename.ext

استبدل filename.ext باسم الملف المطلوب

ومن أجل السواقة B اكتب type b:filename.ext

ومن أجل السواقة C اكتب type c:filename.ext

4. اضغط Enter

(ان محتويات الملف تبدأ الآن بالظهور على الشاشة)

تذكر : إذا كانت المعلومات تظهر بسرعة على الشاشة فانه يمكنك استخدام وظيفة الايقاف المؤقت (pause sc-) أو طباعة البيانات على الطابعة عن طريق وظيفة printer echo .

5. بعد انتهاء عرض المحتويات تعود اشارة الجاهزية >A للظهور .

تغيير اسم الملف :

يستخدم لهذه الغاية الأمر RENAME حيث يسمح لك بتغيير اسم الملف الأصلي أو الاسم الفرعي أو كلاهما معاً. إن الأمر RENAME مفيد فيما اذا رغبت بالاشارة الى الملف عن طريق اسم آخر .

ماذا يلزم لتبديل اسم الملف :

- * الاسطوانة التي تحتوي الملف المطلوب تغيير اسمه .
 - * تحتاج أيضاً الى الاسم الصحيح لهذا الملف (تذكر بأنه يمكن استعمال الأمر Dir لفحص الاسم الكامل لهذا الملف) .
- إذا كان نظام Dos قد شحن سابقاً فلن تحتاج الى اسطوانة النظام حيث أن الأمر RENAME هو أمر داخلي .

الأمر RENAME :

1. تأكد من ظهور اشارة الجاهزية A>
2. ضع الاسطوانة التي تحوي الملف المطلوب تبديل اسمه في السواقة A .
3. اكتب :

`Rename a:filename.ext filename.ext`

استبدل filename.ext الاول بالاسم الحالي الملف و filename.ext الثانية بالاسم الجديد للملف .

ومن أجل تحديد تبديل الاسم في السواقة B اكتب :

`Rename b:filename.ext filename.ext`

ومن أجل تحديد الاسم في السواقة C اكتب :

`Rename c:filename.ext filename.ext`

اضغط Enter

بعد إتمام تبديل الاسم تظهر اشارة الجاهزية A> على الشاشة .

إزالة ملف من الاسطوانة :

تستخدم لهذه الغاية التعليمة ERASE من أجل إزالة الملفات التي لا تريدها

من الاسطوانة. وبالتالي افساح المجال لمزيد من الملفات الأخرى. وتستفيد أيضاً من عدم وجود ارباك في استخدام نسخة قديمة من برنامج ما أو ملف ما .

ملذا تحتاج لإزالة ملف :

• الاسطوانة التي تحوى على الملف المطلوب ازالته .

• تحتاج الى الاسم الصحيح والكامل لهذا الملف (استخدم الأمر DIR إذا أردت) إذا كان نظام Dos قد شحّن سابقاً فلا حاجة لاسطوانة النظام حيث ان الأمر داخلي .

أمر الإزالة Erase أو Del :

11. تأكد من ظهور اشارة الجاهزية >A

2. ضع الاسطوانة التي تحتوي الملف المطلوب في السواعة A

3. اكتب :

`erase filename.ext`

استبدل filename.ext باسم الملف المطلوب .

ومن أجل السواعة B

اكتب : `erase b:filename.ext`

ومن أجل السواعة C

اكتب : `erase c:filename.ext`

4. اضغط Enter :

بعد إزالة الملف يطبع الحاسب اشارة الجاهزية >A .

بعض النصائح المفيدة :

- * قم بعملية نسخ اسطوانتك بشكل دائم .
- * للتأكد ان الأمر يعمل بشكل جيد يجب عليك :
- فحص كتابة الأمر .
- وجود الاسطوانة الصحيحة في السواعة .
- تحديد الملفات المطلوبة، أي التأكد من السواعة التي افترضت .
- لا تنس الإشارة (:) .
- تأكد من كتابتك لاسم الملف بشكل صحيح .
- باستخدام الاسم الفرعي (في حال برامج البيسك فعليك استخدام الاسم الفرعي BAS) .
- * اطبع بشكل دائم أسماء الملفات المستخدمة على الطابعة واحتفظ بهذه النسخة مع الاسطوانة .
- * جميع الأوامر (باستثناء Diskcomp & Diskcopy) التي تتعامل مع الملفات ستعمل مع السواقات المرننة والصلبة .
- * ان الوقت والتاريخ الذي يظهر بجانب اسم الملف اثناء استخدام الأمر Dir يشير الى آخر مرة تم تعديل هذا الملف ، وهذا التاريخ والوقت لا يتغير باستخدام الأمرين Diskcopy و copy .

تشكيل الفهارس الفرعية (Making subdirectones) :

لنفترض أن لديك أساسيتان من الملفات الخاصة بالبيانات والتقارير وترغب بتشكيل فهرسين فرعيين ضمن الفهرس الرئيسي، يدعى أحدهما MDMSOS ويدعى آخر REPORTS . ويحوي الأول على ملفات تم تشكيلها بواسطة معالج الكلمات والثاني يحوي على الملفات اللازمة للبرامج التطبيقية .

ملاحظة : سنستعمل هذه الفهارس الفرعية كامثلة فقط حيث أنه يمكنك تسمية الفهرس الفرعي كما نشاء مع الانتباه الى شروط التسمية ، ومن أجل تشكيل ملف memos إتبع ما يلي :

1. قم بشحن نظام Dos من الاسطوانة الصلبة .
2. استخدم امر Dos وهو MKDIR (او اختصار MD) من أجل انشاء فهرس فرعي mwmos بالشكل : md memos .

3. اضغط Enter

عند ظهور اشارة الجاهزية فهذا يدل على ان الفهرس memos قد تشكل على الاسطوانة الصلبة .

4. استخدم نفس الأمر لتشكيل الفهرس الفرعي reports بالشكل .

5. اضغط Enter

عند ظهور اشارة الجاهزية فهذا يدل على ان الفهرس الفرعي قد تشكل .
أصبح لديك الآن فهرسين فرعيين. يمكنك تشكيل فهرس فرعية حسب ما ترغب ويحدد عددها بالمساحة المتوفرة على الاسطوانة الصلبة، كما يمكنك تشكيل الفهارس الفرعية ضمن بعضها البعض فمثلاً يمكنك تشكيل فهرس فرعي يدعى Letters وهذا الفهرس يوجد ضمن فهرس فرعي آخر يدعى memos . الأمر اللازم لفعل مثل هذا الشيء هو اضغط Enter .

نسخ البرامج الوظيفية الى الفهارس الفرعية :

يمكن حفظ الملفات الخاصة بمعالجة الكلمات في الفهرس الفرعي memos ومن أجل هذا اتبع ما يلي :

ملاحظة : إذا لم يكن لديك ملفات معالجة فاستخدم أي ملفات تريد نقلها الى أحد الفهارس الفرعية .

- 1 - ضع الاسطوانة الخاصة بملفات معالجة الكلمات في السواقة A

2 - قم بنسخ كافة الملفات الموجودة على الاسطوانة الى الفهرس الفرعي الموجود على السواقة الصلبة بالشكل التالي :

```
copy a:*.c:\memos
```

3 - اضغط Enter

إن الأمر *.c يعني أنك ترغب بنقل كافة الملفات وليس ملفاً واحداً .

4 - عند ظهور اشارة الجاهزية اخرج اسطوانة الملفات من السواقة A .

5 - ضع الاسطوانة التي تحوي على التقارير والبيانات (أو أي ملفات أخرى ترغب بنقلها الى الفهرس الفرعي) في السواقة A .

6 - قم بنسخ كافة الملفات الى الفهرس الفرعي بالشكل التالي :

```
copy a:*.c:\reports
```

7 - اضغط Enter

ان كل من الفهرسين الفرعيين يحتوي الآن على الملفات المحددة، كما يمكنك حفظ الملفات المشككة عن طريق هذه البرامج في هذه الفهارس الفرعية اذا رغبت .

نسخ ملف واحد الى الفهرس الفرعي :

لنفترض أن لديك ملفاً واحداً ترغب بنقله الى الفهرس الفرعي. وفي هذه الحالة يجب استخدام الطريقة السابقة مع استبدال *.c باسم الملف المطلوب. مثلاً :

1. لنسخ ملف يدعى Newfile الى الفهرس Reports اكتب :

```
copy a:newfile c:\reports
```

2. اضغط Enter

لقد قمت الآن بنسخ الملف Newfile الى الفهرس الفرعي reports .

إزالة ملف من الفهرس الفرعي :

قد ترغب في بعض الأحيان بإزالة الملفات التي لا تحتاج اليها، مثلاً لنحاول الآن إزالة Newfile من الفهرس reports :

1. ضع اسطوانة النظام Dos في السواعة A (Diskcopy هو أمر خارجي)

2. تأكد من ظهور اشارة الجاهزية A>

3. اكتب : Diskcopy a: b:

4. اضغط Enter فتظهر الرسالة التالية :

Insert SOURCE diskette in drive A:

press any key when ready ...

5. أخرج اسطوانة النظام من السواعة A

6. ضع الاسطوانة الأصلية في السواعة A

7. اضغط أي مفتاح .

أثناء عملية القراءة يتم اضاءة الضوء وبعد الانتهاء تظهر الرسالة التالية :

Insert TARGET diskette in drive A:

press any key when ready ...

8. أخرج الاسطوانة الأصلية من السواعة A

9. ضع الاسطوانة النسخة في السواعة A

10. اضغط أي مفتاح .

ملاحظة : قد تحتاج الى تكرار عملية تبديل الاسطوانة تبعاً لحجم الذاكرة المستخدمة في الجهاز حيث يخبرك الحاسب ما يجب فعله. استمر في التبديل حتى ظهور الرسالة التالية :

Copy another diskette (Y/N) ?

11. اكتب احدى التعليمات التالية :

* N من أجل انتهاء العمل بالأمر Diskcopy حيث يطلب منك وضع اسطوانة تحوي الملف Command. Com وعندها يجب عليك وضع اسطوانة Dos ، ثم اضغط أي مفتاح فتظهر اشارة الجاهزية >A وتكون عملية النسخ قد انتهت .

* Y من أجل تكرار عملية النسخ لاسطوانة أخرى ، عليك في هذه الحالة اتباع الارشادات بدءاً من الخطوة 6 .

الأمر Diskcopy مع سواقتي اسطوانات :

من أجل استخدام هذا الأمر يجب عليك تذكر الاسطوانة الاصلية والاسطوانة النسخة .

1. ضع اسطوانة النظام في السواقة A (Diskcopy هو أمر خارجي) .

2. تأكد من ظهور إشارة الجاهزية >A

3. اكتب : Diskcopy a: b:

4. اضغط Enter فتظهر الرسالة التالية :

Insert SOURCE diskette in drive A:

Insert TARGET diskette in drive B:

press any key when ready ...

5. أخرج اسطوانة النظام من السواقة A

6. ضع الاسطوانة الاصلية في السواقة A

7. ضع الاسطوانة النسخة في السواقة B

8. اضغط أي مفتاح .

يتم اضاءة الضوء اثناء عملية نقل الملفات من اسطوانة الى أخرى تظهر الرسالة التالية :

Copy another diskette (Y/N) ?

9. اكتب :

* N لانتهاء العمل بالأمر Diskcopy فتظهر بعد ذلك اشارة الجاهزية A> وتكون عملية النسخ قد تمت ثم اخرج الاسطوانة التي تم النسخ عليها .

* Y لنسخ اسطوانة أخرى في هذه الحالة يجب عليك اتباع الارشادات بدءاً من الخطوة 6 .

مقارنة الاسطوانات (Comparing Diskettes) :

استخدم الأمر Diskcomp لمقارنة المعلومات الموجودة على اسطوانة مع المعلومات الموجودة على اسطوانة أخرى. ان استخدام الأمر Diskcomp عادة يتم بعد استعمال الأمر Diskcopy للتأكد من صحة عملية النسخ ومطابقة البيانات بين الاسطوانتين .

ماذا تحتاج لمقارنة اسطوانتين :

* اسطوانة نظام Dos

* الاسطوانة الاصلية التي تود المقارنة معها .

* الاسطوانة النسخة التي تود مقارنة محتوياتها .

توافقية الأمر Diskcomp :

يستخدم هذا الأمر مع توافقية محددة بين الاسطوانات وسواقات الاسطوانات .

الأمر Diskcomp مع سواقة اسطوانات واحدة :

لاستخدام هذا الأمر تذكر بأن الاسطوانة الاصلية هي الاسطوانة الاولى
والاسطوانة النسخة هي الاسطوانة الثانية :

1. ضع اسطوانة النظام في السواقة A

2. تأكد من ظهور اشارة الجاهزية >A

3. اكتب : Diskcomp a: b:

4. اضغط Enter فتظهر الرسالة التالية :

Insert FIRST diskette in drive A:

press any key when ready ...

5. أخرج اسطوانة النظام Dos من السواقة A

6. ضع الاسطوانة الاصلية في السواقة A

7. اضغط أي مفتاح .

يضيء الضوء وتبدأ عملية قراءة الملفات ثم تظهر الرسالة التالية :

Insert SECOND diskette in drive A:

press any key when ready ...

8. أخرج الاسطوانة الاصلية من السواقة A

9. ضع الاسطوانة النسخة في السواقة A

10. اضغط أي مفتاح .

ملاحظة : قد تحتاج الى تمرار عملية تبديل الاسطوانات، وذلك تبعاً لحجم
الذاكرة المستخدمة في الجهاز استمر في التبديل حتى ظهور الرسالة التالية :

Compare OK

Compare more diskette (Y/N) ?

ملاحظة : في حال عدم المطابقة بين الاسطوانة الاصلية والنسخة فاعد عملية النسخ بواسطة الامر Diskcopy ثم عملية المقارنة Diskcomp .
11. اكتب :

* N لانتهاء العمل بالامر Diskcomp فتظهر اشارة الجاهزية >A وتكون قد تمت عملية المقارنة بنجاح .

* Y لمقارنة اسطوانات أخرى، اتبع الارشادات مرة أخرى من الخطوة 6 .
ملاحظة : إذا حصلت على الرسالة التالية :

Insert COMMAND.COM diskette in drive X:

and strik any key when ready ...

قبل ظهور اشارة الجاهزية >A فضع اسطوانة النظام Dos في السواقة A ثم اضغط أي مفتاح. عندها تظهر اشارة الجاهزية .

الامر Diskcomp مع سواقتي اسطوانات :

لاستخدام هذا الامر تذكر بان الاسطوانة الاصلية هي الاسطوانة الاولى والاسطوانة النسخة هي الاسطوانة الثانية :

1. ضع اسطوانة النظام Dos في السواقة A (Diskcomp هو أمر خارجي)

2. تأكد من ظهور اشارة الجاهزية >A

3. اكتب : Diskcomp a: b:

4. اضغط Enter فتظهر الرسالة التالية :

Insert FIRST diskette in drive A:

Insert SECOND diskette in drive B:

press any key when ready ...

5. أخرج اسطوانة النظام Dos من السواقة A

6. ضع الاسطوانة الاصلية في السواعة A

7. ضع الاسطوانة النسخة في السواعة B

8. اضغط أي مفتاح .

يضيء الضوء الاحمر وأثناء عملية المقارنة وبعد ذلك تظهر الرسالة التالية :

Compare OK

Compare more diskette (Y/N) ?

ملاحظة : في حال عدم المطابقة بين الاسطوانتين أعد عملية النسخ بواسطة الأمر Diskcopy ثم عملية المقارنة مرة أخرى Diskcomp .

9. اكتب :

* N لانتهاء العمل بالمقارنة فتظهر اشارة الجاهزية >A وتكون عملية عملية المقارنة قد تمت بنجاح .

* Y لمقارنة اسطوانات أخرى أعد الارشادات من الخطوة 6 .

ملاحظة : إذا حصلت على الرسالة التالية :

Insert COMMAND.COM diskette in drive X:

and strik any key when ready ...

وذلك قبل ظهور إشارة الجاهزية >A هذا فيجب أن تضع اسطوانة النظام Dos في السواعة A ثم اضغط أي مفتاح . عندها تظهر إشارة الجاهزية >A .

نسخ ملف (Coping aFile) :

استخدم الأمر Copy عندما ترغب في نسخ ملف واحد وليس اسطوانة بالكامل .

ماذا تحتاج لنسخ ملف :

* الاسطوانة التي تحوي على الملف المطلوب نسخه .

* الاسطوانة التي سيتم نسخ الملف عليها .

إذا كان نظام Dos قد جرى شحنته سابقاً فانك لن تحتاج الى اسطوانة النظام لأن الأمر Copy هو امر داخلي ويمكنك أيضاً اختيار اسم الملف المطلوب حفظه على الاسطوانة الجديدة .

استخدام الأمر Copy مع وجود سواقة اسطوانات واحدة :

من السهل معرفة الاسطوانات الأصلية والاسطوانات النسخة، حيث B يرمز للنسخة Backup وذلك عند ظهور رسالة تطلب وضع اسطوانة النسخة :

copy a:filename.ext b:

1. تأكد من ظهور اشارة الجاهزية A>

2. ضع الاسطوانة الصلبة في السواقة A

3. اكتب : Copy a: Filename.ext b:

حيث Filename هو اسم الملف المطلوب .

4. اضغط Enter فتظهر الرسالة التالية :

Insert diskette for drive B:

and strike any when ready

5. أخرج الاسطوانة الأصلية من السواقة A

6. ضع الاسطوانة النسخة في السواقة

7. اضغط أي مفتاح .

ملاحظة : قد تحتاج الى تكرار عملية تبديل الاسطوانات وذلك تبعاً لحجم

الذاكرة المستعملة في الجهاز. استمر في هذه العملية حتى ظهور الرسالة :

1 File(s) copied

A>

8. أخرج الاسطوانة النسخة .

استخدام الأمر copy مع سواقتي اسطوانات :

من السهل معرفة أي من الاسطوانتين الأصل وأي منهما النسخة إذا اعتبرت أن يرمز إلى النسخة وذلك عند ظهور رسالة تطلب وضع اسطوانة النسخة :

1. تأكد من ظهور اشارة الجاهزية A>

2. ضع الاسطوانة الصلبة في السواقة A

3. ضع الاسطوانة النسخة في السواقة

4. اكتب : copy a:filename.ext b:

ضع عوضاً عن filename.ext اسم الملف المطلوب نسخه .

5. اضغط Enter :

يضيء الضوء لفترة اثناء عملية النسخ وحين الانتهاء تظهر الرسالة :

1 File(s) copied

A>

6. أخرج الاسطوانتين وضع لصاقة على اسطوانة النسخة بالمحتويات والتاريخ .

مقارنة ملف (Comparing a file) :

استخدم الأمر COMP مقارنة ملفين وفحص فيما اذا كانا متطابقين. وعادة تجري هذه العملية بعد استخدام الأمر Copy في نسخ ملف ما .

ماذا تحتاج لمقارنة ملفين :

1. اسطوانة نظام Dos .
2. الاسطوانتين التي تحوي على الملفين المطلوب مقارنتهما .

استخدام الأمر Comp مع سواقة اسطوانات واحدة :

1. ضع اسطوانة النظام Dos في السواقة A (Comp هو أمر خارجي)
2. تأكد من ظهور اشارة الجاهزية >A
3. اكتب : Comp
4. اضغط Enter فتظهر الرسالة التالية :

Enter primary file name

5. أخرج اسطوانة نظام Dos في السواقة A
6. ضع الاسطوانة الأصلية في السواقة A
7. اكتب : a:filename.ext
- (استبدل filename.ext باسم الملف المطلوب)
8. اضغط فتظهر الرسالة التالية :

Enter 2nd file name or drive id

9. اكتب : b: filename.ext
- (استبدل filename.ext باسم الملف الثاني)
10. اضغط Enter فتظهر الرسالة التالية :

Insert diskette for drive B:

and strike any key when ready

11. أخرج الاسطوانة الأصلية من السواقة A

12. ضع الاسطوانة النسخة في السواعة A

13. اضغط Enter

14. اكتب :

* N إذا رغبت في انتهاء المقارنة وبعد ذلك تظهر اشارة الجاهزية >A وتم عملية المقارنة بنجاح .

* Y إذا رغبت في مقارنة ملفات أخرى عندها يجب متابعة الارشادات السابقة بدءاً من الخطوة 6 .

ملاحظة : في حال ظهور الرسالة التالية :

Insert COMMAND.COMP diskette in drive x:

and strike any key when ready

وذلك قبل ظهور إشارة الجاهزية >A . ضع اسطوانة نظام Dos في السواعة A ثم اضغط أيفتاح فتظهر الاشارة مرة أخرى .

استخدام الأمر Comp مع سواقتي اسطوانات :

1. ضع اسطوانة النظام Dos في السواعة A (Comp هو أمر خارجي)

2. تأكد من ظهور اشارة الجاهزية >A

3. اكتب : Comp

4. اضغط Enter فتظهر الرسالة التالية :

Enter orimary file name

5. أخرج اسطوانة نظام Dos في السواعة A

6. ضع الاسطوانة الاصلية في السواعة A

7. ضع الاسطوانة النسخة في السواعة B

8. اكتب : filename.ext

(استبدل filename.ext باسم الملف المطلوب)

9. اضغط Enter .

10. اكتب : filename.ext

(استبدل filename.ext باسم الملف المطلوب)

11. اضغط Enter

يضيء الضوء الأحمر من الزمن، وحين انتهاء المقارنة تظهر الرسالة التالية :

Files compare ok

Compare more file (Y/N) ?

12. اكتب :

* N من أجل انتهاء عملية المقارنة فتظهر عندها اشارة الجاهزية A> . وتكون قد تمت عملية المقارنة بنجاح .

* Y من أجل مقارنة ملفات أخرى عد الى الخطوة 6 .

ملاحظة : في حال ظهور الرسالة التالية :

Insert COMMAND.COMP diskette in drive x:

and strike any key when ready

وذلك قبل ظهور اشارة الجاهزية A> . فضع اسطوانة نظام Dos في السواقة A ثم اضغط أي مفتاح فتعود الاشارة للظهور .

فحص محتويات الاسطوانة (Dir) :

استخدم لهذه الغاية الأمر DIR لمعرفة أسماء الملفات الموجودة على الاسطوانة كما يلي :

(من أجل فحص محتويات السواقة "C" اكتب : dir c)

4. اضغط Enter

5. راقب الشاشة، حيث أن أول رسالة تظهر هي اسم الاسطوانة (في حال وجوده) ثم أسماء الملفات الموجودة في هذه الاسطوانة، وأسماء الفهارس المختلفة .

فحص محتويات الملف (Displaying the file) :

يستخدم لهذه الغاية الأمر Type حيث يسمح لك بفحص ما يحتويه ملف ما من بيانات وطباعتها على الشاشة. حيث أن ملفات النصوص على الشاشة بشكل واضح ومقروء بينما الملفات الأخرى مثل ملفات البرامة وما شابهها تطبع بشكل رموز غريبة والسبب أنها تحوي على أحرف لا تعود إلى الأحرف الأبجدية .

ماذا تحتاج لفحص محتويات ملف :

* الاسطوانة التي تحوي على الملف المطلوب فحصه .

* معرفة الاسم الصحيح للملف المطلوب فحصه (استعمل dir مرة أخرى) .

(إذا كان نظام Dos قد شُحن سابقاً فلن تحتاج لاسطوانة النظام) .

الأمر type :

1. تأكد من ظهور إشارة الجاهزية >A

2. ضع الاسطوانة التي تحوي الملف المطلوب في السواقة A .

3. اكتب : type a:filename.ext

استبدل filename.ext باسم الملف المطلوب

ومن أجل السواعة B اكتب type b:filename.ext

ومن أجل السواعة C اكتب type c:filename.ext

4. اضغط Enter

(ان محتويات الملف تبدأ الآن بالظهور على الشاشة)

تذكر : إذا كانت المعلومات تظهر بسرعة على الشاشة فإنه يمكنك استخدام وظيفة الايقاف المؤقت (-sc pause) أو طباعة البيانات على الطابعة عن طريق وظيفة orinter echo .

5. بعد انتهاء عرض المحتويات تعود اشارة الجاهزية >A للظهور .

تغيير اسم الملف :

يستخدم لهذه الغاية الأمر RENAME حيث يسمح لك بتغيير اسم الملف الأصلي أو الاسم الفرعي أو كلاهما معاً. إن الأمر RENAME مفيد فيما إذا رغبت بالإشارة إلى الملف عن طريق اسم آخر .

ماذا يلزم لتغيير اسم الملف :

* الاسطوانة التي تحتوي الملف المطلوب تغيير اسمه .

* تحتاج أيضاً إلى الاسم الصحيح لهذا الملف (تذكر بأنه يمكن استعمال الأمر Dir لفحص الاسم الكامل لهذا الملف) .

إذا كان نظام Dos قد شحن سابقاً فلن تحتاج إلى اسطوانة النظام حيث إن الأمر RENAME هو أمر داخلي .

الأمر RENAME :

1. تأكد من ظهور اشارة الجاهزية >A

2. ضع الاسطوانة التي تحوي الملف المطلوب تغيير اسمه في السواعة A .

3. اكتب :

`Rename a:filename.ext filename.ext`

استبدل filename.ext الأول بالاسم الحالي للملف و filename.ext الثانية بالاسم الجديد للملف .

ومن أجل تحديد تبديل الاسم في السوارة B اكتب :

`Rename b:filename.ext filename.ext`

ومن أجل تحديد الاسم في السوارة C اكتب :

`Rename c:filename.ext filename.ext`

اضغط Enter

بعد إتمام تبديل الاسم تظهر اشارة الجاهزية >A على الشاشة .

إزالة ملف من الاسطوانة :

تستخدم لهذه الغاية التعليمة ERASE من أجل إزالة الملفات التي لا تريدها من الاسطوانة. وبالتالي افساح المجال لمزيد من الملفات الأخرى. وتستفيد أيضاً من عدم وجود ارباك في استخدام نسخة قديمة من برنامج ما أو ملف ما .

ماذا تحتاج لازالة ملف :

* الاسطوانة التي تحوي على الملف المطلوب ازالته .

* تحتاج الى الاسم الصحيح والكامل لهذا الملف (استخدم الأمر Dir إذا أردت)

إذا كان نظام Dos قد شحن سابقاً فلا حاجة لاسطوانة النظام حيث ان الأمر الداخلي .

أمر إزالة Erase أو Del :

1. تأكد من ظهور اشارة الجاهزية >A
2. ضع الاسطوانة التي تحوي الملف المطلوب تبديل اسمه في السواقة A .
3. اكتب :

erase filename.ext

استبدل filename.ext باسم الملف المطلوب .

ومن أجل السواقة B

اكتب : erase b:filename.ext

ومن أجل السواقة C

اكتب : erase c:filename.ext

4. اضغط Enter :

بعد إزالة الملف يطبع الحاسب إشارة الجاهزية >A .

بعض النصائح المفيدة :

- * قم بعملية نسخ اسطوانتك بشكل دائم .
- * للتأكد ان الأمر يعمل بشكل جيد يجب عليك :
- فحص كتابة الأمر .
- وجود الاسطوانة الصحيحة في السواقة .
- تحديد الملفات المطلوبة، أي التأكد من السواقة التي افترضت .
- لا تنس الاشارة (:) .
- تأكد من كتابتك لاسم الملف بشكل صحيح .
- باستخدام الاسم الفرعي (في حال برامج البيسك فعليك استخدا
الفرعي BAS) :

* اطبع بشكل دائم أسماء الملفات المستخدمة على الطابعة واحتفظ بهذه النسخة مع الاسطوانة .

* جميع الأوامر (باستثناء Diskcomp & Diskcopy) التي تتعامل مع الملفات ستعمل مع السواقات المرنة والصلبة .

* ان الوقت والتاريخ الذي يظهر بجانب اسم الملف أثناء استخدام الأمر Dir يشير الى آخر مرة تم تعديل هذا الملف، وهذا التاريخ والوقت لا يتغير باستخدام الأمرين Diskcopy و Copy .

تشكيل الفهارس الفرعية (Making subdirectones) :

لنفترض أن لديك أساسيتان من الملفات الخاصة بالبيانات والتقارير وترغب بتشكيل فهرسين فرعيين ضمن الفهرس الرئيسي، يدعى أحدهما MDMSOS ويدعى آخر REPORTS . ويحوي الأول على ملفات تم تشكيلها بواسطة معالج الكلمات والثاني يحوي على الملفات اللازمة للبرامج التطبيقية .

ملاحظة : سنستعمل هذه الفهارس الفرعية كأمثلة فقط حيث أنه يمكنك تسمية الفهرس الفرعي كما تشاء مع الانتباه الى شروط التسمية، ومن أجل تشكيل ملف memos اتبع ما يلي :

1. قم بشحن نظام Dos من الاسطوانة الصلبة .
2. استخدم الأمر Dos وهو MKDIR (أو اختصار MD) من أجل انشاء فهرس فرعي memos بالشكل : md memose .

3. اضغط Enter

عند ظهور اشارة الجاهزية فهذا يدل على ان الفهرس memos قد تشكل على الاسطوانة الصلبة .

4. استخدم نفس الأمر لتشكيل الفهرس الفرعي reports بالشكل .

5. اضغط Enter

عند ظهور اشارة الجاهزية فهذا يدل على ان الفهرس الفرعي قد تشكل .
أصبح لديك الآن فهرسين فرعيين. يمكنك تشكيل فهرس فرعية حسب ما
ترغب ويحدد عددها بالمساحة المتوفرة على الاسطوانة الصلبة، كما يمكنك
تشكيل الفهارس الفرعية ضمن بعضها البعض فمثلاً يمكنك تشكيل فهرس
فرعي يدعى Letters وهذا الفهرس يوجد ضمن فهرس فرعي آخر يدعى
Memos . الامر اللازم لفعل مثل هذا الشيء هو اضغط Enter .

نسخ البرامج الوظيفية الى الفهارس الفرعية :

يمكن حفظ الملفات الخاصة بمعالجة الكلمات في الفهرس الفرعي memos
ومن أجل هذا اتبع ما يلي :
ملاحظة : إذا لم يكن لديك ملفات معالجة فاستخدم أي ملفات تريد نقلها
الى أحد الفهارس الفرعية .

1 - ضع الاسطوانة الخاصة بملفات معالجة الكلمات في السواقة A

2 - قم بنسخ كافة الملفات الموجودة على الاسطوانة الى الفهرس الفرعي الموجود
على السواقة الصلبة بالشكل التالي :

copy a:*. *c:\memos

3. اضغط Enter

ان الامر *.* يعني انك ترغب بنقل كافة الملفات وليس ملفاً واحداً .

4 - عند ظهور اشارة الجاهزية اخرج اسطوانة الملفات من السواقة A .

5 - ضع الاسطوانة التي تحوي على التقارير والبيانات (أو أي ملفات أخرى ترغب
بنقلها الى الفهرس الفرعي) في السواقة A .

6 - قم بنسخ كافة الملفات الى الفهرس الفرعي بالشكل التالي :

copy a:*. *c:\reports

7 - اضغط Enter

ان كل من الفهرسين الفرعيين يحتوي الآن على الملفات المحددة، كما يمكنك حفظ الملفات المشككة عن طريق هذه البرامج في هذه الفهارس الفرعية اذا رغبت .

نسخ ملف واحد الى الفهرس الفرعي :

لنفترض ان لديك ملفاً واحداً ترغب بنقله الى الفهرس الفرعي. وفي هذه الحالة يجب استخدام الطريقة السابقة مع استبدال *. * باسم الملف المطلوب مثلاً :

1. لنسخ ملف يدعى Newfile الى الفهرس Reports اكتب :

copy a:newfile c:\reports

2. اضغط Enter

لقد قمت الآن بنسخ الملف newfile الى الفهرس الفرعي reports .

إزالة ملف من الفهرس الفرعي :

قد ترغب في بعض الأحيان بإزالة الملفات التي لا تحتاج اليها، مثلاً لنحاول الآن إزالة newfile من الفهرس reports :

1. اكتب : erase c:\reports\newfile

2. اضغط Enter

يجب أن تكتب أولاً اسم الفهرس الفرعي ثم الرمز (\) ثم الاسم الكامل للملف المطلوب إزالته .

الانتقال من وإلى الفهارس الفرعية :

كما ذكرنا سابقاً عندما تبدأ من نظام Dos ستكون مباشرة في الفهرس الرئيسي أي الفهرس الحالي الذي تعمل فيه سواء ذكرت اسمه أم لم تذكره،

وهكذا فإن نظام Dos يقوم بتذكر هذا الفهرس دوماً ما لم تحدد له غيره .
على الرغم من أنك قد شكلت فهرسين فرعيين، ولكنك لم تأمر الحاسب
بالتحول الى احدهما حتى الآن. وكما ترى فإن الأبواب الخاصة بهذه الغرف
لا تزال مغلقة مما يشير على أنك لا زلت في الفهرس الرئيسي. يجب عليك
الآن أن تتعرف على طريقة الانتقال من وإلى هذه الفهارس الفرعية للوصول
الى الملفات والبرامج المطلوبة، لذلك يجب أن تستخدم الأمر CHDIR للانتقال
الى الفهرس المطلوب .

لنفترض بأنك ترغب باستعراض الملفات الموجودة في الفهرس الفرعي Re-
ports ، وفي هذه الحالة يجب الانتقال الى الفهرس الفرعي كما يلي :

1. استخدم الأمر (CHDIR) أو اختصار CD واكتب :

cd\reports

2. اضغط Enter

ان الباب الخاص بغرفة Reports قد فتح الآن مشيراً الى أنك الآن في الفهرس
الفرعي Reports .

3. من أجل استعراض الملفات الموجودة في الفهرس Reports اكتب : Dir

4. اضغط Enter : فتظهر على الشاشة لائحة بأسماء الملفات الموجودة .

من أجل الانتقال الى الفهرس الفرعي memos وجعله الفهرس الحالي اكتب :

cd\memos

ثم اضغط Enter .

ان الباب الخاص بغرفة Memos قد فتح الآن مشيراً الى أنك في الفهرس
الفرعي Memos .

فحص الفهرس المستخدم :

إذا لم تكن متأكداً في أي فهرس فرعي تعمل فبإمكانك طلب اسم هذا

الفهرس، لا تستعمل الرمز (\) في الأمر واكتب : cd ثم اضغط Enter فترى الآن أن الفهرس المستخدم هو Memos وتظهر النتيجة التالية على الشاشة :
C:\MEMOS

العودة الى الفهرس الرئيسي :

للعودة الى الفهرس الرئيسي من الفهرس الفرعي اكتب : cd\ ثم اضغط Enter .

عرض محتويات الفهرس الفرعي :

من أجل عرض كافة محتويات الفهرس الفرعي :

* استخدم cd (تغير الفهرس) أو (تغير الاتجاه) ثم الأمر Dir .

المثال التالي يوضح العمل بالطريقة الأولى، أي إذا كنت في الفهرس الرئيسي ورغبت في عرض محتويات الفهرس الفرعي Reports فقم بما يلي :

1 - انقل رأس القارئ الى الفهرس Reports كما في الشكل

cd/Reports

2 - اضغط Enter .

3 - استخدم الأمر Dir لعرض الملفات على الشاشة واكتب : Dir

4 - اضغط Enter فتتم عملية عرض كافة الملفات على الشاشة .

عرض ملف واحد من الفهرس الفرعي :

لعرض ملف واحد معروف الاسم، يمكن استخدام الأمر Dir كما يجب اخبار نظام Dos باسم الفهرس المطلوب البحث فيه بالاضافة الى اسم الملف. لنفترض بأن اسم الملف Myfile وإن الفهرس هو Reports من أجل عرض هذا الملف اكتب :

ثم اضغط Enter .

ملاحظة : اذا لم يكن لديك ملف اسمه myfile (وهذا ممكن) استخدم الطريقة السابقة مع اسم الملف الخاص بك .

إزالة فهرس فرعي :

استخدم الأمر RMDIR لإزالة الفهرس الفرعي تذكر الأمور التالية :

* لا يمكن إزالة الفهرس الفرعي ما لم يكن فارغاً أي يحوي فقط على الإدخالات الخاصة (.) و (..) وعند عرضك لمحتويات هذا الملف لن ترى سوى هذه المدخلات الخاصة .

* يمكن إزالة فهرس فرعي واحد في كل مرة .

* لا يمكن إزالة فهرسك الحالي الموجود فيه الفهرس الرئيسي. لنفترض بانك ترغب في إزالة الفهرس Reports من الاسطوانة الصلبة، لذلك اتبع الخطوات التالية :

1. قم بتغير الفهرس الى الفهرس الفرعي Reports واكتب :

cd \Reports

2. اضغط Enter .

3. قم بإزالة كافة الملفات الموجودة في هذا الفهرس .

هام : لا تقم بإزالة كافة الملفات بهذه العملية ما لم تكن متأكداً انك في الفهرس الفرعي المطلوب إزالته. لانك في هذه الحالة ستزيل ملفات أخرى بشكل غير مقصود .

اكتب : erase *.*

4. اضغط Enter .

أجب بالحرف Y على السؤال ? Are you sure .

5. عد الى الفهرس الرئيسي حيث انه لا يمكنك ازالة الفهرس الفرعي وأنت موجود فيه .

اكتب : cd/

6. اضغط Enter .

7. من أجل ازالة الفهرس Reports استخدم الامر RMDIR (RD)

اكتب :

Rd\reports

8. اضغط Enter . ان ظهور اشارة الجاهزية يدل على ان الفهرس Reports قد أزيل من الاسطوانة الاصلية. وإذا حاولت الانتقال الى الفهرس Reports فستحصل على الرسالة التالية :

Invalid directory

طباعة أسماء الفهارس الفرعية الموجودة على الاسطوانة الصلبة :

بإمكانك طباعة أسماء الفهارس الفرعية باستخدام الامر TREE كما انه بإمكانك طباعة أسماء الملفات في كل فهرس اذا استخدمت الأمر الفرعي /f بالشكل :

tree\ ثم اضغط Enter

من أجل رؤية أسماء الفهارس الفرعي فقط دون الملفات اكتب :

A>tree A:/f

RAM MANAGEMENT

أحد المهام الرئيسية لنظام التشغيل هو إدارة الذاكرة العشوائية Random Access Memory (RAM) والتي تجمع فيها مكونات النظام (مشغلات الأجهزة، برامج الذاكرة المبيتة TSR والتطبيقات) ، ولذلك يقوم نظام التشغيل DOS بتنظيم الذاكرة الرئيسية للحيلولة دون تداخل تلك البرامج .

عند بداية تصنيع أجهزة الحاسوب في بداية الثمانينات كان الحجم الموضوع آنذاك للذاكرة الرئيسية أكثر من كاف لأنه لم يكن يتوقع أحد أن يستخدم ذاكرة تصل الى 640 كيلو بايت ولذلك فإن حجم الذاكرة المستخدمة بدأ بـ 46 كيلو بايت ثم ازداد حتى 128K و 256K وبمرور الوقت ازداد الطلب على الذاكرة الرئيسية حتى وصل الآن الى 640 كيلو بايت كحد أدنى .

وما دمنا وصلنا الى عصر أجهزة البنتيوم والواجهات الرسومية للمستخدم Graphical User Interface وأنظمة التشغيل متعددة الوظائف Multitasking Operating System والتي لا تكفيها الـ 640 كيلو بايت فإن السؤال الذي يطرح نفسه هو ... لماذا لا نزيد حجم الذاكرة بزيادة رقاقات الذاكرة الى نظام الحاسوب ؟؟

وللإجابة على هذا التساؤل نقول انه لا يمكن ذلك لعدة أسباب، السبب الأول يتمثل في أن نظم المعالجات 8086 قد حددت بحجم 1 ميجابايت من الذاكرة ولا يمكن تغيير الأساس الذي بني عليه المعالج بسهولة، أما السبب الثاني فهو التوافقية (Compatiplity) ، لأنه من المفروض منطقياً على برنامج يعمل في جهاز بنتيوم أن يعمل بنفس الكفاءة على أجهزة XT (وان اختلفت السرعة) ، وهذا يتطلب توافق مع مستويات الذاكرة كلا الجهازين ومن هنا جاءت التوافقية أحد شروط تطوير إدارة الذاكرة في الأجيال المختلفة من أعمار الحاسوب .

في هذا البند سوف نقوم بالتعرف على أهم مستويات الذاكرة الرئيسية

وتقسيماتها التي تمكن نظام التشغيل من التعامل معها، نبدأ أولاً بمناطق الذاكرة ومستوياتها .

مستويات الذاكرة : -

يوصف نظام التشغيل (دوس) DOS بأنه نظام يعمل في الصيغة الحقيقية Real Mode والتي تعني أنه لا يستطيع استخدام الذاكرة التي تزيد عن 1 ميجابايت، وهذا الحجم يتجزأ الى عدة مستويات هي كالتالي : -

الأول : منطقة الذاكرة السفلية Lower Memory Area وتسمى أحياناً Conventional Memory وتحمل هذه الذاكرة أول 640 كيلوبايت من الذاكرة حيث يستطيع نظام دوس التعامل معها مباشرة دون الحاجة الى أي برنامج اضافي كما يحتاج الى هذه المنطقة معظم البرامج التي تعمل في بنية DOS مثل برامج إدارة البايوس PCBIOS ، الدوس، جداول الدوس Dos Table الاعتراضات Interrupts والبرامج المبينة في الذاكرة TSR وغيرها وما تبقى يمنح للبرامج التقليدية والتطبيقية والتي تسمى Transient Program Area (TPR) .

الثاني : الذاكرة العلوية Upper Memory Area : وتبلغ سعتها 384 كيلوبايت وتستخدم عادة لعتاد الكمبيوتر Computer Hardware مثل موائم العرض Display Adapters مشغلات الأجهزة الطرفية (والتي تسمى البرامج القائدة للأجهزة الطرفية) Device Drivers .

أما الاسم الاصطلاحي Extended Memory أو الذاكرة الممددة XMS فيطلق على مساحة الذاكرة فيما وراء الـ 1 ميجابايت وهي تحتاج الى برنامج خاص لإدارتها، ويحتاج النظام WINDOWS والبرمجيات التي تعتمد عليه الى هذا النوع من الذاكرة، يطلق على أول 64 كيلوبايت من الذاكرة الممددة XMS اسم الذاكرة العالية (HMA) HIGH MEMORY والتي يتم شحن جزء من نظام التشغيل DOS فيها مما يزيد من مساحة الذاكرة في منطقة الذاكرة السفلية - التقليدية Conventional Memory والمخصصة للبرامج

التطبيقية TPA ومع ذلك نبقى محددين بقيمة 640 كيلوبايت كحد أقصى للذاكرة (حتى هذه المرحلة) .

أما الذاكرة الموسعة Expanded Memory فهي ذاكرة يمكن إضافتها على بطاقة خاصة ويمكن استعمالها من قبل تطبيقات DOS إضافة إلى الذاكرة التقليدية والعوية، وتحتاج مثل هذه الذاكرة إلى برنامج إدارة خاص، وتستطيع البرامج استخدام هذه الذاكرة ضمن وحدات مساحة الوحدة منها 46 كيلوبايت في الوقت الواحد وذلك عن طريق منطقة في الذاكرة العوية UPPER MEMORY تسمى إطار الصفحة EMS Page Frame وبسبب محدودية مقدار الذاكرة التي يمكن الوصول إليها في وقت واحد فإن استخدام الذاكرة الموسعة Expanded بطيء مقارنة مع الذاكرة الممددة Extended .

ولكن كيف يمكن التغلب على حاجز 640 كيلوبايت في الأجهزة الحديثة والتي تزود بحجم 2 ميجابايت من الذاكرة كحد أدنى !!!

كما ذكرنا فإن مشكلة توسعة الذاكرة ظهرت عند تخطي برامج المستخدم والبرامج التطبيقية TPA الحجم 640 كيلوبايت وحل هذه المعضلة استخدمت عدة طرق منها أولاً استخدام الذاكرة الثانوية (عادة القرص الصلب HDD) كذاكرة رئيسية ولكن المشكلة في هذه الطريقة هي محدودية السرعة لوحدات التخزين الثانوية نسبة إلى الذاكرة الرئيسية RAM حيث تكون السرعة بطيئة نسبياً. وقد استبعدت هذه الطريقة لتوفير الوقت اللازم لمعالجة البرامج وذلك انطلاقاً من القاعدة التي تقول ان الوقت هو المال "TIME IS MONEY" .

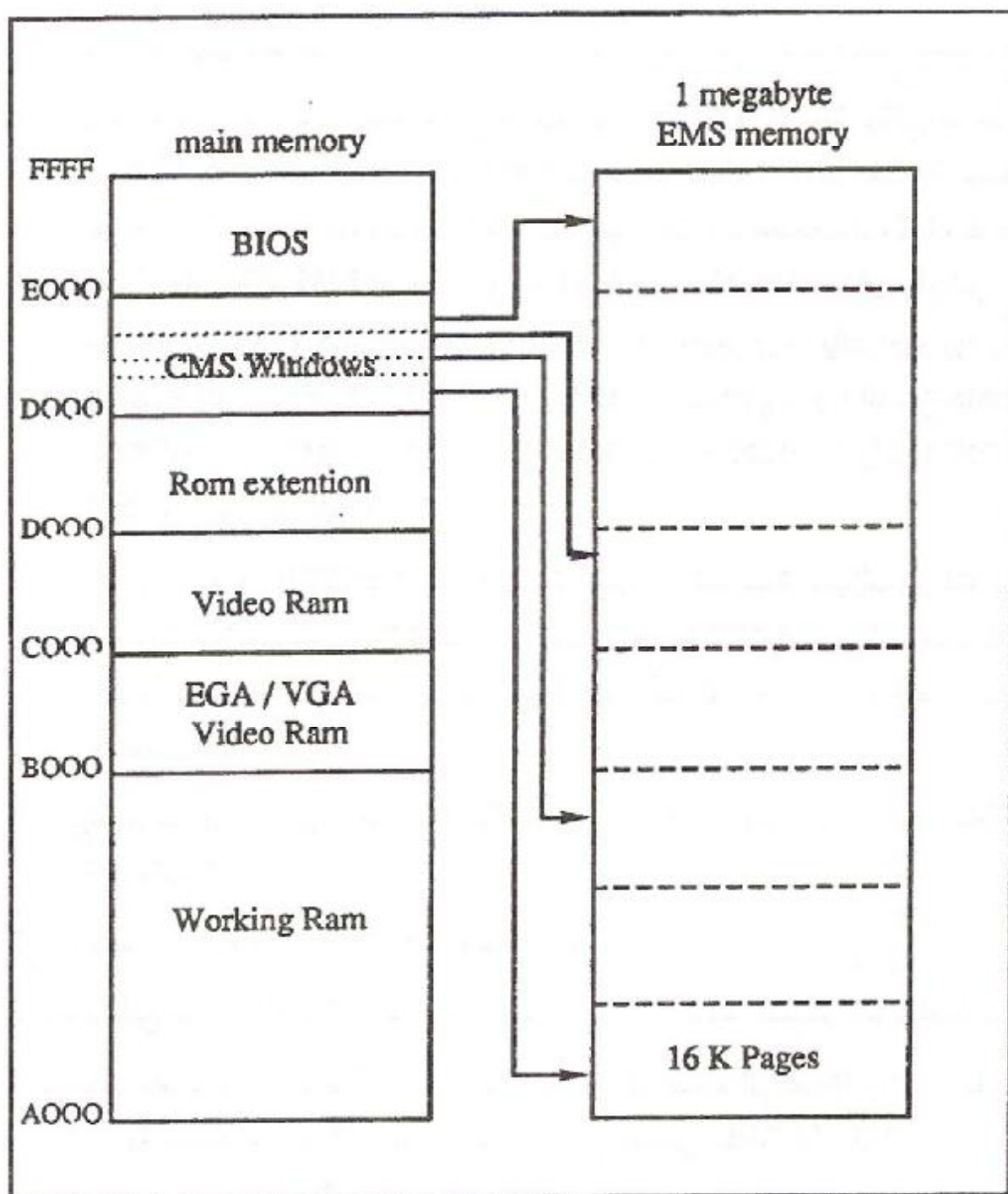
الطرق الأخرى التي استخدمت هي استخدام الذاكرة الموسعة Expanded والذاكرة الممددة Extended واللذان سنتعرض لهما بشيء من التفصيل .

الذاكرة الموسعة : (EMS) EXPANDED MEMORY

الذاكرة الموسعة هي ذاكرة اضافية استخدمت بداية في أجهزة الحاسوب PC XT / ، حيث كانت معالجات 8088 محددة بـ 640 كيلوبايت. وبسبب الحاجة الى زيادة الذاكرة للعمل في بيئة الدوس والتي تصلح لأجهزة PCs, ATs قامت شركات لوتس LOTUS وانتل INTEL ومايكروسوفت MICROSOFT بالتعاون معاً وتشكيل نظام معياري يدعى LIM والتي تمثل اختصاراً للأحرف الأولى لتلك الشركات الشهيرة في عالم الحاسوب، هذا النظام المعياري سمح باضافة 8 ميجابايت من الذاكرة على كرت توسعة خاص بحيث يتم استخدام 64 كيلوبايت في الرقعة الواحد وذلك بوضعه في المنطقة العلوية من الذاكرة والموجودة في الميجابايت الأولى حيث تسمى هذه المنطقة بإطار الصفحة Page Frame ، وتكون التوسعة باستخدام خدعة يتم فيها انشاء نافذة عناوين تقوم بالتأشير على أماكن تواجد المعلومات (الذاكرة) والتي تكون عادة خارج عنوان الواحد ميجابايت المستخدم أي انها عناوين صفحات منطقية وهذا يمكن النظام من استخدام ذاكرة ذات سعة تصل الى عدة ميجابايت والشكل (1) يوضح المعنى .

حيث نلاحظ أن 64 كيلوبايت والتي تسمى إطار الصفحة Page Frame (تخطط - تؤشر) الى أربعة صفحات خارج حدود الواحد ميجابايت ولذلك تسمى هذه الصفحات «الصفحات المنطقية Logical Pages» بينما تسمى الصفحات الموجودة في منطقة الذاكرة العلوية بـ «الصفحات الفيزيائية Physical Pages» .

تقسيم هذه المنطقة الى أربعة صفحات مستقلة يوفر ميزة في هذه الذاكرة وهي سهولة تشغيل أربعة برامج والانتقال بينها دون أي تأخير في الوقت. ويجدر الإشارة الى أن العلاقة بين الصفحات الفيزيائية والمنطقية هي علاقة تأشير (تخطيط) من الصفحات الفيزيائية الى المنطقية وبمعنى آخر فانه عند استخدام المعلومات في الصفحات المنطقية (خارج حدود الواحد ميجابايت) يتم نقل المؤشر اليها عن طريق الصفحات الفيزيائية، ولا يتم نقل المعلومات



الشكل (1) : تنظيم الذاكرة وإدارتها

من الصفحات المنطقية الى الفيزيائية كما يحدث في الذاكرة الممددة Extended وكما سنرى بعد قليل .

يتم استخدام البرمجيات التي تقوم بإدارة الذاكرة الموسعة والتي تسمى Expanded Memory Manager (Emm) حيث تعمل كبرامج معيارية بينية تسمح بالوصول الى كروت التوسعة على اختلاف مصنعها، وكما ذكرنا أعلاه فان نظام LIM المعياري (والذي أنشأته الشركات انتل - لوتس - وميكروسوفت) كان الركيزة الأساسية لتطوير برمجيات تقوم بإدارة الذاكرة الموسعة والتي اطلق عليها EMS حيث انتج منها النسخ EMS VERSION 1, EMS VERSION 3, EMS VERSION 3.2 واخيراً EMS VERSION 4 في عام 1987 .

يمتاز نظام EMS VERSION 4 لإدارة ذاكرة EMS بعدة مميزات جعلته في مرتبة متقدمة بين الأنظمة التي ظهرت لإدارة الذاكرة حيث استخدم 58 اقتراحاً ووظيفة لدعم عتاد Hardware الكمبيوتر كما انه ضم العديد من السمات منها :-

1 - يدعم حجم ذاكرة تصل الى 32 ميغابايت مقارنة مع 8 ميغابايت EMS VERSION 3.2 .

2 - يستطيع انشاء نوافذ EMS في أي مكان من الذاكرة .

3 - يستطيع نوافذ EMS أن تكون بأي حجم وليست مقيدة بالحجم 64 كيلوبايت.

4 - يمكن حماية نوافذ EMS في حالة حدوث خطأ في النظام أو Reset وذلك بزيادة سرعة الحفظ من الذاكرة الى القرص (والتي تسمى RAM Disk) .

ويمكن استخدام الاعتراض 67H لاستخدام الذاكرة الموسعة والحصول على معلومات عن نسخة EMS الموجودة على جهاز الحاسوب، الجدول التالي يبين الاعتراض 67H والاقترانات التابعة له .

67H INTERRUPT

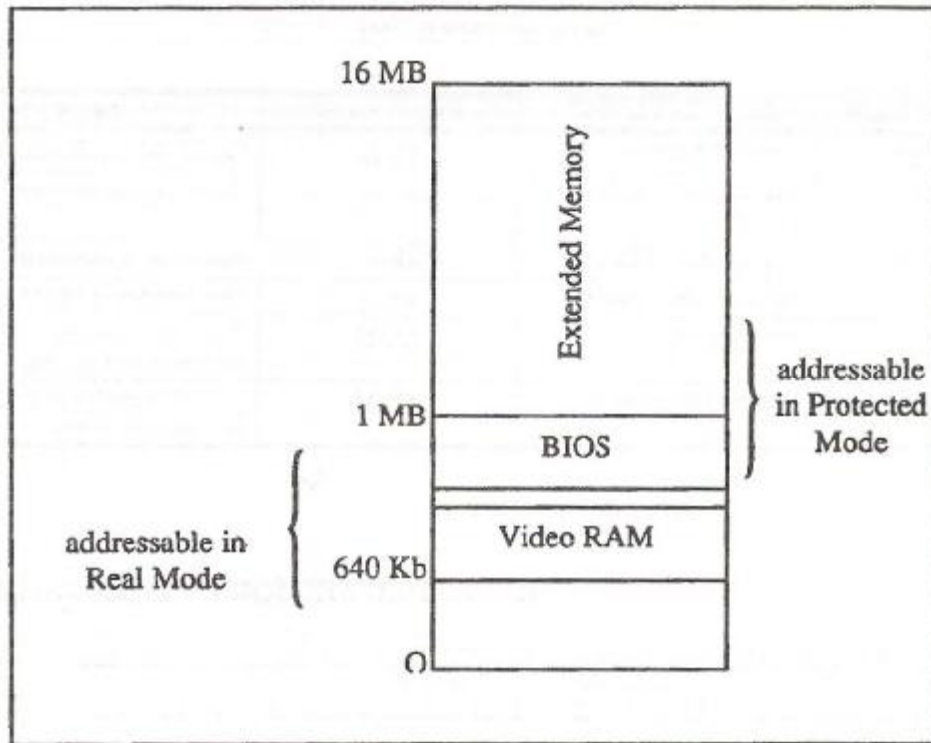
FUNCTION	TASK	FUNCTION	TASK
40 H	Get EMM status	46 H	Get EMM version number
41 H	Get segment address of page frame	47 H	Save current mapping
42 H	Get number of pages	48 H	Reset saved mapping
43 H	Allocate EMS pages	49 H	Get number of EMM handles
44 H	Set mapping	4A H	Get the number of pages allocated to a handle
45 H	Release EMS pages	4B H	Get all handles and numbers of allocated pages

الذاكرة الممتدة : EXTENDED MEMORY

كما ذكرنا مسبقاً فإن الذاكرة الممتدة Extended Memory هي الذاكرة الموجودة بعد الواحد ميجابايت الأولى والتي لا تستطيع الوصول إليها إلا النظم التي تعمل في الشكل المحمي Protected Mode وذلك فإن الانظمة ذات المعالجات 8086 و 8088 لا تستطيع عنوان الذاكرة فوق الواحد ميجابايت .

ونستطيع (تشغيل) الشكل المحمي Protected Mode وذلك بتغيير أحد الخانات في مسجل الحالة وبالتالي فإنه يمكن استخدام (عنوان) ذاكرة تصل الى 16 ميجابايت ولكن احتمال حدوث أخطاء في النظام وارد بسبب الميكانيكية المختلفة التي تستخدمها المعالجات 80286 في العنوان في حالة Protected Mode .

يتم استخدام بعض اقترانات البايوس وما يسمى مشغلات الذاكرة الممتدة XMS Drivers وذلك للوصول الى هذا النوع من الذاكرة مع تفضيل XMS Drivers على BIOS Function لأن XMS DRIVERS يستطيع اشراك أكثر من برنامج الذاكرة الممتدة بدلاً من استخدامها جميعاً لبرنامج واحد. والشكل (2) يبين منطقة الذاكرة الممتدة Extended Memory .



الشكل (2) : الذاكرة الممتدة

تتواجد الذاكرة الممتدة في كل جهاز حاسوب يحتوي على 1 ميجابايت من الذاكرة فاكثر ، لأن الحجم الفعلي المستخدم هي 640 كيلوبايت أو 512 كيلوبايت، وما تبقى يمكن اعتباره ذاكرة ممتدة، وعند استخدام البايوس في معالجة هذه الذاكرة نجد أن الاقتران 88H من الاعتراض 15H يقوم بإعلام المستخدم عن حجم الذاكرة الممتدة التي يستخدمها النظام بالكيلوبايت، ويمكن الوصول إلى الذاكرة الممتدة بتحريك Block عن طريق الاقتران 87H ويتم النقل من المنطقة تحت الواحد ميجابايت إلى الذاكرة الممتدة وبالعكس . وبشكل عام فإن الاعتراض 15H يستخدم لمعالجة الذاكرة الممتدة ، ولكن يجب مراعاة أن استخدام الاقترانات السابقة يجب أن يتم عند عمل النظام في حالة Protected Mode فقط .

استخدام XMS المعيارية : -

وبخلاف EMS التي احتوت على برامج معيارية لتوفير البنية مع الذاكرة الموسعة Memory Expanded فان الذاكرة الممتدة Extended Memory الموجودة منذ سنين طويلة لم يتوفر لها مثل هذه البرمجيات الا في وقت متأخر حيث قامت شركات انتل ولوتس ومايكروسوفت و AST بتطوير نظام مواصفات معياري للذاكرة الممتدة - Ectended Memory Specificalion (XMS) ولذلك فان XMS قامت بتوفير البنية اللازمة لاستخدام الذاكرة الممتدة وبعض مناطق الذاكرة مثل :

1 - منطقة الذاكرة العالية High Memory Area HMA وهي اول 64 كيلوبايت بعد الميجابايت الاولى والتي تمتد من 1024K الى 1088K .

2 - اربع حزم من الذاكرة الممتدة Extended Memory Block (EMB) والتي تبدأ من 1088K .

3 - حزم الذاكرة العلوية Upper Memory Block (UMB) والتي تكون في المنطقة ما بين 640K و 1024K .

من أشهر المشغلات القائمة لـ XMS هو برنامج HIMEM.SYS والذي أنتجته مجموعة مايكروسوفت وتضعه حالياً في النسخ الحديثة من اصدار DOS وويندوز حيث يتم استدعاؤه عند البدء (BOOTUP) بوضعه في . CONFIG.SYS.

ومن الأمثلة على الأنظمة التي تعتمد XMS هو QEMM و EMM386.EXE والتي تقوم بالتعامل مع الذاكرة الممتدة كذاكرة عليا Upper Memory Blocks .

ويمكن استخدام اقترانات البايوس عند التعامل مع XMS ، فالاعتراض 2FH يستخدم لهذا الهدف الجدول التالي يبين الاقترانات المتوفرة لاستخدام XMS .

2F INTERRUPT

<i>FUNCTION</i>	<i>TASK</i>	<i>FUNCTION</i>	<i>TASK</i>
00 H	Determine XMS version number	09 H	Allocate Extended Memory Block (EMB)
01 H	Allocate High Memory Area (HMA)	0A H	Free allocated Extended Memory Block (EMB)
02 H	Free High Memory Area (HMA)	0B H	Move Extended Memory Block (EMB)
03 H	Globally enable address line A20	0C H	Lock Extended Memory Block (EMB)
04 H	Globally disable address line A20	0D H	Unlock Extended Memory lock (EMB)
05 H	locally enable address line A20	0E H	Get EMB handle information
06 H	locally disable address line A20	0F H	Resize Extended Memory Block (EMB)
07 H	Query status of address line A20	10 H	Allocate Upper Memory Block (UMB)
08 H	Query free extended memory	11 H	Free allocated Upper Memory Block (UMB)

نظام التشغيل

UNIX

يعتبر نظام التشغيل UNIX من أنظمة التشغيل متعددة البرامج ويتيح لك هذا النظام تطبيق الفعاليات والمهام التالية :

أولاً : الدخول والخروج من النظام وبين الجدول (1) التعليمات المستخدمة للدخول والخروج .

الجدول (1) : تعليمات الدخول والخروج

To Do Thos ...	Type This ...
Log in	(after logging:) <i>your-username</i> (after password:) <i>your-passwoed</i> (after TERM=) <i>your-term-type</i>
Enter a command	(type the command after the prompt and press Return)
Find out your shell name	echo \$SHELL
Set or chanhe password	passwd
Log out	exit

ثانياً : التعامل مع الملفات من استحداث الملف، تسمية الملف استعراض الملف، طباعة الملف، تغيير اسم الملف، نسخ الملف، حذف الملف وغيرها من مهام ويبين الجدول (2) التعليمات المستخدمة في معالجة الملفات .

الجدول (2) : تعليمات معالجة الملفات

To Do Thos ...	Type This ...
Creat a file	<code>cat > filename</code>
Terminate keyboard input for cat	CTRL - D
List visible files in current directory	<code>ls</code>
List visible and invisible files in current directory	<code>ls - a</code>
View a file	<code>more filename</code>
Print a file	<code>lp myfile</code>
Revame ("move") a file	<code>mv fromfile tofile</code>
Copy a file	<code>cp fromfile tofile</code>
Delet (remove) a file	<code>rm filename</code>
Find out access permissions	<code>ll filename</code>

ثالثاً : تنظيم الملفات في الأدلة وذلك بتنفيذ مجموعة من التعليمات مرتبطة بالأدلة مثل : تحديد مستوى الدليل في شجرة الأدلة، تحديد موقع المستخدم، عرض محتويات الدليل، استحداث الدليل ، نقل الملفات بين الأدلة ، حذف الدليل ويبين الجدول (3) أهم تعليمات معالجة الأدلة .

الجدول (3) : تعليمات الدليل

To Do Thos ...	Type This ...
List files: show directories with "/"	<code>lsf</code>
Gange directory	<code>cd directory-path</code>
Change to home directory	<code>cd</code>
Display working directory	<code>pwd</code>
Remove an (empty) dorectory	<code>remdir directory-name</code>
Display oermissions for a directory	<code>ll -d directory-name</code>

رابعاً : تعليمات الإدخال والإخراج وتشمل نقل المعلومات بين الملفات والبرامج، كتابة المخرجات في الملف، استخدام الملف للقراءة، استخدام مخرجات احدى التعليمات كمدخلات لتعليمة أخرى ويبين الجدول (4) بعضاً من هذه التعليمات .

الجدول (4) : تعليمات الإدخال والإخراج

To Do Thos ...	Type This ...
Redirect (save) standard output to a file	<i>command > outfile</i>
Append standard output on a file	<i>command >> outfile</i>
Redirect input from a file to a command	<i>command < infile</i>
Redirect both standard input and putput to a a file	<i>command < in file > outfile</i>
Connect (:pipe" between) tow process	<i>command1 command2</i>
Simulataneously file command output and send to another command ("tee")	<i>command1 tee file command2</i>

خامساً : عمليات الكتابة والتحرير باستخدام محرر الشاشة وذلك لتنفيذ كافة عمليات التحرير اللازمة لنص معين ويبين الجدول (5) أهم هذه التعليمات .

الجدول (5) : تعليمات المحرر

To Do Thos ...	Type This ...
Enter vi	vi filename
Make sure you're in command mode	Press ESC
Insert text before the cursor position	Press i
Insert text after the cursor position	Press a
Find out what file you are in	Press CTRL - g
Show whether you are in command or insert mode	: set shoemode
Write (save) thw current file.	: w
Write the current file to <i>file-name</i>	: W <i>file-name</i>
Overwrite the contents of <i>file-name</i> with the current file.	: w! <i>file-name</i>
Write lines x through y of the current file to <i>file-name</i> .	: x.y w <i>file-name</i> where x.y represents special line numbers or place markers.
Copy a line into a buffer	yy
Put copied or deleted text line after cursor line	p
Read in another file after cursor line	: r filename Return
Find <i>text</i>	/ <i>text</i>
Run an HP-UX command while in vi	: ! <i>command-name</i>
Print the current file	: ! lp %
Print a file	: ! lp filename
write the current file and quot (exit) vi.	: wq
Quit vi <i>without</i> saving any changes.	: q!
Join lines of text	J at end of line
Set line length (wrapmarging)	: set wrapmargin=n

سادساً : ارسال واستقبال البريد وتشمل هذه المهمة مجموعة من الفعاليات أهمها:
 قراءة البريد، ارسال البريد حذف البريد، حفظ البريد وبيّن الجدول (6) أهم
 العمليات المنفّذة على البريد .

الجدول (6) : تعليمات البريد

To Do Thos ...	Use this elm commnd ...
Delete the message marked for delete without quitting elm. ("Resync")	\$
Get help on elm commands.	?
Allows you to send a command to the shell without leaving elm.	!
Set up mail aliases.	a
Change the mailbox.	c
Mark messages for deletion.	d
Forward the current message to another user.	f
Send a group reply to all recipients of the original message.	g
Move the message pointer to the next message (below).	j
Move the message pointer to the previous message (above).	k
Send mail to a specified user or users.	m
Allows you to alter the setting of different mail parameters. including the sorting method for messages, the destination of printed messages, the type of menus displayed, and so on.	o
Print messages. (You can change the destination of printed messages using the o command listed above) .	p
Quit elm with the option of changing the contents of the mailbox.	q
Reply to the author of the current message.	r
Save a message to a file.	s
Exit elm without any changes.	x.

سابعاً : البحث والفرز وتضم هذه المهمة مجموعة من الفعاليات أهمها : البحث عن ملف، البحث عن نص، ترتيب الملف ويبين الجدول (7) أهم هذه العمليات .

الجدول (7) : تعليمات البحث والفرز

To Do Thos ...	Type This ...
Sort the contents of a file and display the results to standard output	<code>sort options filename</code>
Find all filename in the current directory and subdirectories beginning with "x"	<code>find . -name "x*" - print</code>
Find the word "Jim" in any file in the current sirectory	<code>grep Jim *</code>
Find the words "Joe Smith" in any file in the current directory	<code>grep 'Joe Smith *</code>

أما خيارات الترتيب فهي مبينة في الجدول (8)

الجدول (8) : خيارات الترتيب

Sort option ...	Does This ...
-b	(disregard leading blanks) Blanks, which can be either spaces or tabs. delimit sortable fields. Wothout the -b option. sort considers all blanks preceeding a field to be part of that field, and will use them in sort comparisons.
-d	(dictionary) disregards all characters that are not alphanumeric or blanks. In particular. this option disregards control characters or punctuation.
-f	(fp1d) doesn't consider any differences between upper and lower case.
-n	(numerical) sorts in numerical order. Plus and minus signs are considered as plus and minus signs, and dots (.) are considered as decimal points.
-r	(reverse) reverse sort order (i.e.. z-a).
-u	(Unique) removes duplicate lines fro a sorted file.

ثامناً : استخدام بيئة Shell ويبين الجدول (9) أهم العمليات المنفذة في بيئة Shell.

الجدول (9) : تعليمات بيئة Shell

To Do This ...	Type This ...
Determine what shell you're in	echo \$SHELL
Temporarily change to Bourne Shell	sh
Temporarily change to Korn Shell	ksh
Temporarily change to C Shell (see Appendix)	csh
Permanently change to another shell	chsh <i>username shell-path-name</i> (then log out and log in again)
Set command-line editor	set -o <i>editor-name</i>
Edit your command line (once editor is set)	Press [ESC] ; use vi commands to move cursor and enter text
Recall a previous command line	In vi mode, press [ESC] ; press k (backwards) or j (forward) to move through command history file
Execute a previous command line	Press [Return] when desired command line is displayed
Set a variable value	VARIABLE-NAME= <i>variable-value</i>
Enter a subshell	sh
Display PATH setting	echo \$PATH
Set terminal parameters	test <i>options term-type</i>

تاسعاً : التعامل مع بيئة الشبكة وبيّن الجدول (10) أهم التعليمات المستخدمة في بيئة الشبكة .

الجدول (10) : تعليمات بيئة الشبكة

To Do Thos ...	Type This ...
Invoke ftp and connect to <i>remote-hots</i>	ftp <i>remote-host</i>
Copy files from <i>remote-hots</i> to current directory, in ftp	get <i>remote-file</i>
Copy files from your local current directory to the current directory on <i>remote-host</i> in ftp	put <i>local-file</i>
List the contents of the current remote directory	ls
Exit ftp	bye
<i>Also see Table 11-1 for File-Manipulation Commands on ftp</i>	
Copy a file from your <i>local-file</i> to a file on a remote host, using rcp	rcp <i>local-file</i> <i>remote-hostname:remote-file</i>
Copy a file from a remote host to your local sirectory, using rcp	rcp <i>remote-hostname:remote-file</i> <i>local-file</i>
Copy a directory structure from your local system to a remote host	rcp -r <i>local-dir</i> <i>remote-hostname:remote-dir</i>
Copy a directory structure from a <i>remote system</i> to your local system	rep -r <i>remote-hostname:remote-dir</i> <i>local-dir</i>
Log in on a remote system	rlogin <i>remote-hostname</i>
Exit rlogin	exit

عاشراً : الأمن والحماية ويبين الجدول (11) أهم العمليات المنفذة .

الجدول (11) : الأمن والحماية

To Do This ...	Type This ...
Display file permissions	II
Numerically change file or directory permissions	<code>chmod number name</code>
Symbolically add or subtract file or directory Permissions	<code>chmod class±permissions name</code>
Symbolically change (absolutely) file or directory Permissions	<code>chmod class=permissions name</code>
Find out permissions mask setting	<code>umask</code>
Numerically change permissions mask setting	<code>umask mask-number</code>

تطبيق عملي

تطبيق عملي

في هذا التطبيق سوف نستعرض كيفية استخدام لغة باسكال ولغة اسمبلي لبناء برنامج. تطبيقي يستطيع تنفيذ بعض مهام نظام التشغيل DOS وقد اخترنا :

1 - التعامل مع الأدلة : انشاء دليل، تغيير الدليل، حذف الدليل .

2 - التعامل مع الملفات :

تغيير اسم الملف

حذف الملف

نسخ الملف

استعراض محتويات الملف (Dump)

البحث عن ملف

3 - التعامل مع الوقت (Time)

4 - التعامل مع التاريخ (Date)

5 - عرض قيود الملف (ATTRIB)

وفيما يلي البرنامج والذي ينفذ هذه الفعاليات :

```

PROGRAM SUL;
USES CRT, DOS, GRAPH;
LABEL L1;

TYPE
    day=(SUN,MON,TUE,WED,THU,FRI,SAT);
    RR= ARRAY[1..5] OF POINTTYPE;

VAR
    dta:array [1..60] of byte;
    NUM:ARRAY [1..4] OF BYTE;
    size :real;
    sts,filename,c_dir,c_path,drive,path,
    file_1,file_2,S,D,buf1 :string;
    f_ound,dd,mm,yy1,yy2,drive_no,DY,
    DRV_NO,e_f,ws1 :byte;
    file_1_handel,file_2_handel,yy,
    buf_handel :word;
    A,I,J,M1,M11,X,Y,ERR,B,err1,err2,C1,
    aa,att,NUM2,xx,tty :INTEGER;
    GD, GM,fr,fs,fh,fc,hor,min,sec,hsc,
    FILES_num,XF1,YF1,XF :INTEGER;
    TR,TR1,TR2,BT,CN,BT1,BT2,
    BT22 :RR ;
    C,CH :CHAR;
    FLAG :BOOLEAN;
    {*****}
PROCEDURE FIRST_MENU;
VAR
    A1:REAL;
    XF,YF:REAL;
    del_ay:integer;
BEGIN
    del_ay:=200;
    TR2[1].X:=320; TR2[1].Y:=240;
    TR2[2].X:=323; TR2[2].Y:=240;
    TR2[3].X:=323; TR2[3].Y:=243;
    TR2[4].X:=320; TR2[4].Y:=243;
    TR2[5].X:=320; TR2[5].Y:=240;

    SETFILLSTYLE(6,5);
    A1:=2;
    FOR J:=1 TO 10 DO
    BEGIN
        SETFILLSTYLE(6,J);
        XF:=(TR2[3].X-TR2[1].X)/2+TR2[1].X;
        YF:=(TR2[3].Y-TR2[1].Y)/2+TR2[1].Y;
        FOR I:=1 TO 5 DO
        BEGIN
            TR2[I].X:=ROUND(XF+(TR2[I].X-XF)*A1);
            TR2[I].Y:=ROUND(YF+(TR2[I].Y-YF)*1.7);
        END;
        SETCOLOR(14);
        DRAWPOLY(5,TR2);
        FILLPOLY(5,TR2);
        DELAY(del_ay);
    END;
    SETFILLSTYLE(6,16);
    FILLPOLY(5,TR2);
    TR2[1].X:=0; TR2[1].Y:=0;
    TR2[2].X:=630; TR2[2].Y:=0;
    TR2[3].X:=630; TR2[3].Y:=470;

```

✓

```

TR2[5].X=0;      TR2[5].Y:=0;
SETFILLSTYLE(6,5);
DRAWPOLY(5,TR2);
FILLPOLY(5,TR2);
DELAY(del_ay);
SETCOLOR(14);
SETTEXTSTYLE(7,HORIZDIR,10);
OUTTEXTXY(50,40,'P');
DELAY(500);
OUTTEXTXY(130,40,'R');
DELAY(500);
OUTTEXTXY(210,40,'O');
DELAY(500);
OUTTEXTXY(290,40,'J');
DELAY(500);
OUTTEXTXY(370,40,'E');
DELAY(500);
OUTTEXTXY(450,40,'C');
DELAY(500);
OUTTEXTXY(530,40,'T');
DELAY(500);
SETTEXTSTYLE(1,HORIZDIR,12);
OUTTEXTXY(250,170,'O');
DELAY(500);
OUTTEXTXY(350,170,'F');
DELAY(500);
SETTEXTSTYLE(4,HORIZDIR,9);
OUTTEXTXY(40,300,'S');
DELAY(500);
OUTTEXTXY(110,300,'U');
DELAY(500);
OUTTEXTXY(180,300,'L');
DELAY(580);
OUTTEXTXY(250,300,'/');
DELAY(500);
OUTTEXTXY(320,300,'O');
DELAY(500);
OUTTEXTXY(390,300,'S');
DELAY(500);
SETTEXTSTYLE(0,HORIZDIR,10);
OUTTEXTXY(460,310,'');
DELAY(500);
OUTTEXTXY(530,310,'');
DELAY(500);
SETCOLOR(4);
SETTEXTSTYLE(DEFAULTFONT,HORIZDIR,4);
OUTTEXTXY(100,410,'Presta Any Key');
SETTEXTSTYLE(6,1,20);
OUTTEXTXY(20,30,'made by Suleiman Khalaf');
SETTEXTSTYLE(3,0,3);
OUTTEXTXY(220,30,'made by Suleiman Khalaf');
    REPEAT
    UNTIL (KEYPRESSED) ;
END;
PROCEDURE get_dir;

procedure getdir;assembler;
asm
    push ax
    push bx
    push cx
    push si
    mov ah,$47

```



```

mov  dl,0
lea  si,ds:c_path
int  $21
pop  si
pop  cx
pop  bx
pop  ax
end;
{*****}
BEGIN
  c_dir:='';
  getdir;
  I:=0;
  WHILE ORD (c_path[I])<>0 DO
    BEGIN
      c_dir:=c_dir+ c_path[I];
      I:=I+1;
    END;
  END;

PROCEDURE get_drive;
PROCEDURE g_d;assembler;
asm
  push ax
  mov  ah,$19
  int  $21
  mov  byte ptr ds:drive_no,al
  pop  ax
END;
BEGIN
  g_d;
  drive:=chr (65+drive_no);
END;

PROCEDURE get_path;
BEGIN
  path:='';
  get_drive;
  get_dir;
  path:=drive+'\'+c_dir;
END;

PROCEDURE CONV (N,k:INTEGER );
VAR NUM1:INTEGER;
BEGIN
  j:=1;
  NUM1:=0;
  for i:=1 to 4 do num[i]:=$20;
  FOR I:=4 DOWNTO 4-k DO
    BEGIN
      num1:=n mod 10;
      num1:=num1*$30;
      num[i]:=num1;
      n:=n div 10;
    end;
  END;

PROCEDURE WW;
BEGIN
  CN[1].X:=-20;      CN[1].Y:=40;
  CN[2].X:=620;     CN[2].Y:=-40;

```

```

CN[4].X:=-20;          CN[4].Y:=310;
CN[5].X:=-20;          CN[5].Y:=40;
{-----}
SETFILLSTYLE(3,4);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);
CN[1].X:=-25;          CN[1].Y:=-45;
CN[2].X:=-615;         CN[2].Y:=-45;
CN[3].X:=-615;         CN[3].Y:=-305;
CN[4].X:=-25;          CN[4].Y:=-305;
CN[5].X:=-25;          CN[5].Y:=-45;
{-----}
SETFILLSTYLE(6,5);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);

```

END;

PROCEDURE WE;
BEGIN

```

CN[1].X:=-20;          CN[1].Y:=320;
CN[2].X:=-620;         CN[2].Y:=320;
CN[3].X:=-620;         CN[3].Y:=478;
CN[4].X:=-20;          CN[4].Y:=478;
CN[5].X:=-20;          CN[5].Y:=320;
{-----}
SETFILLSTYLE(1,4);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);

CN[1].X:=-25;          CN[1].Y:=325;
CN[2].X:=-615;         CN[2].Y:=325;
CN[3].X:=-615;         CN[3].Y:=473;
CN[4].X:=-25;          CN[4].Y:=473;
CN[5].X:=-25;          CN[5].Y:=325;
{-----}
SETFILLSTYLE(2,5);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);

```

END;

PROCEDURE WE2;
BEGIN

```

CN[1].X:=-20;          CN[1].Y:=320;
CN[2].X:=-620;         CN[2].Y:=320;
CN[3].X:=-620;         CN[3].Y:=478;
CN[4].X:=-20;          CN[4].Y:=478;
CN[5].X:=-20;          CN[5].Y:=320;
{-----}
SETFILLSTYLE(1,4);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);
CN[1].X:=-25;          CN[1].Y:=325;
CN[2].X:=-615;         CN[2].Y:=325;
CN[3].X:=-615;         CN[3].Y:=473;
CN[4].X:=-25;          CN[4].Y:=473;

```

```

=====
SETFILLSTYLE(2,5);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);
SETFILLSTYLE(6,5);
XF1:=80;
FOR J:=1 TO 3DO
  BEGIN
    BAR(XF1,380,XF1+150,410);
    XF1:=XF1+180;
  END;
SETCOLOR(3);
OUTTEXTXY(80,380,'(R,r)Read only');
OUTTEXTXY(270,380,'(H,h)Hidden ');
OUTTEXTXY(450,380,'(S,s)System ');
END;

PROCEDURE WE1;
BEGIN
  CN[1].X:=20;      CN[1].Y:=320;
  CN[2].X:=620;     CN[2].Y:=320;
  CN[3].X:=620;     CN[3].Y:=478;
  CN[4].X:=20;      CN[4].Y:=478;
  CN[5].X:=20;      CN[5].Y:=320;
  {=====}
  SETFILLSTYLE(1,4);
  FILLPOLY(4,CN);
  SETCOLOR(14);
  DRAWPOLY(4,CN);
  CN[1].X:=25;      CN[1].Y:=325;
  CN[2].X:=615;     CN[2].Y:=325;
  CN[3].X:=615;     CN[3].Y:=473;
  CN[4].X:=25;      CN[4].Y:=473;
  CN[5].X:=25;      CN[5].Y:=325;
  {=====}
  SETFILLSTYLE(2,5);
  FILLPOLY(4,CN);
  SETCOLOR(14);
  DRAWPOLY(4,CN);
  SETFILLSTYLE(6,5);
  YF1:=360;
  FOR I:=1 TO 2 DO
    BEGIN
      XF1:=80;
      FOR J:=1 TO 3DO
        BEGIN
          BAR(XF1,YF1,XF1+150,YF1+30);
          XF1:=XF1+160;
        END;
      YF1:=YF1+40;
    END;
  BAR(400,435,550,465);
  SETCOLOR(3);
  OUTTEXTXY(105,365,'Read only');
  OUTTEXTXY(265,365,'Hidden ');
  OUTTEXTXY(425,365,'System ');
  OUTTEXTXY(425,405,'Changed ');
  OUTTEXTXY(90,405,'Volume name ');
  OUTTEXTXY(250,405,'Subdirectory ');
  OUTTEXTXY(420,435,'Normal file ');

```

```

PROCEDURE W2;-
BEGIN
  BT[1].X:=20;          BT[1].Y:=10;
  BT[2].X:=330;         BT[2].Y:=10;
  BT[3].X:=330;         BT[3].Y:=50;
  BT[4].X:=20;          BT[4].Y:=50;
  BT[5].X:=20;          BT[5].Y:=10;
  SETFILLSTYLE(1,5);
  FILLPOLY(5,BT);
  SETCOLOR(14);
  DRAWPOLY(5,BT);
  FOR I:=1 TO 7 DO
  BEGIN
    FOR J:=1 TO 5 DO
      BT[J].Y:=BT[J].Y+60;
      FILLPOLY(5,BT);
      SETCOLOR(14);

      DRAWPOLY(5,BT);

    END;
    SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR, 2);
    OUTTEXTXY(30,20,'DIRICTORY');
    OUTTEXTXY(30,22,' ');
    OUTTEXTXY(30,80,'FILE TOOLS');
    OUTTEXTXY(50,82,' ');
    OUTTEXTXY(35,140,'FIND FILE');
    OUTTEXTXY(35,142,' ');
    OUTTEXTXY(35,200,'TIME [ SET / GET ]');
    OUTTEXTXY(35,202,' ');
    OUTTEXTXY(35,260,'DATE [ SET / GET ]');
    OUTTEXTXY(35,262,' ');
    OUTTEXTXY(35,320,'ATTRIB [ SET/GET ]');
    OUTTEXTXY(35,322,' ');
    OUTTEXTXY(35,380,'ABOUT SUL/OS');
    OUTTEXTXY(45,382,' ');
    OUTTEXTXY(35,440,'QUIT');
    OUTTEXTXY(35,442,' ');
  {REPEAT
    UNTIL KEYPRESSED;}
  END; {PROCEDURE}

```

```

PROCEDURE SECOND_MENU;
BEGIN
  TR[1].X:=340;          TR[1].Y:=60;
  TR[2].X:=628;          TR[2].Y:=60;
  TR[3].X:=628;          TR[3].Y:=468;
  TR[4].X:=340;          TR[4].Y:=468;
  TR[5].X:=340;          TR[5].Y:=60;
  {-----}
  TR1[1].X:=343;          TR1[1].Y:=63;
  TR1[2].X:=625;          TR1[2].Y:=63;
  TR1[3].X:=625;          TR1[3].Y:=465;
  TR1[4].X:=343;          TR1[4].Y:=465;
  TR1[5].X:=343;          TR1[5].Y:=63;
  {-----}
  TR2[1].X:=0;            TR2[1].Y:=0;
  TR2[2].X:=635;          TR2[2].Y:=0;
  TR2[3].X:=635;          TR2[3].Y:=475;
  TR2[4].X:=0;            TR2[4].Y:=475;

```



```

SETCOLOR(5);
SETFILLSTYLE(1,1);
FILLPOLY(5,TR2);
SETCOLOR(14);
DRAWPOLY(5,TR2);
(-----)
SETCOLOR(14);
DRAWPOLY(5,TR1);
DRAWPOLY(5,TR);
SETFILLSTYLE(2,3);
`bar(340,10,628,50);

SETTEXTSTYLE(1,0,10);
OUTTEXTXY(350,50,'S');
DELAY(50);
OUTTEXTXY(380,100,'U');
DELAY(50);
OUTTEXTXY(410,150,'L');
DELAY(50);
OUTTEXTXY(440,200,'/');
DELAY(50);
OUTTEXTXY(470,250,'O');
DELAY(50);
OUTTEXTXY(500,300,'S');
DELAY(50);
OUTTEXTXY(470,050,'*');
DELAY(50);
OUTTEXTXY(380,300,'*');
DELAY(50);
OUTTEXTXY(550,040,'*');
DELAY(50);
OUTTEXTXY(550,120,'*');
DELAY(50);
get_path;
SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,2);
OUTTEXTXY(345,15,' ');
XF1:=345;
I:=0;
WHILE ORD (path[I])<>0 DO BEGIN
  OUTTEXTXY(XF1,15,path[I]);
  XF1:=XF1+11;
  I:=I+1;
END;
W2;

END; {PROCEDURE}
PROCEDURE W22;
BEGIN
  BT22[1].X:=200;          BT22[1].Y:=60;
  BT22[2].X:=400;          BT22[2].Y:=60;
  BT22[3].X:=400;          BT22[3].Y:=100;
  BT22[4].X:=200;          BT22[4].Y:=100;
  BT22[5].X:=200;          BT22[5].Y:=60;
  SETFILLSTYLE(1,5);
  FILLPOLY(5,BT22);
  SETCOLOR(14);
  DRAWPOLY(5,BT22);
  FOR I:=1 TO 3 DO
    BEGIN
      FOR J:=1 TO 5 DO
        BT22[J].Y:=BT22[J].Y+50;
        FILLPOLY(5,BT22);
        SETCOLOR(14);
    
```

```

DRAWPOLY(5,BT22);

END;
SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,2);
OUTTEXTXY(205,65,'MAKE DIR');
OUTTEXTXY(205,67,' ');
OUTTEXTXY(205,115,'CHANGE DIR');
OUTTEXTXY(205,117,' ');
OUTTEXTXY(205,165,'REMOVE DIR');
OUTTEXTXY(205,167,' ');
OUTTEXTXY(205,215,'ESCAPE');
OUTTEXTXY(205,217,' ');
REPEAT
UNTIL KEYPRESSED;
END; {PROCEDURE}

PROCEDURE W33(SD:INTEGER;F:INTEGER);
BEGIN
  SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,2);
  IF SD<>60 THEN FOR I:-1 TO 5 DO
    BT2[I].Y:=BT2[I].Y+SD;
    SETFILLSTYLE(1,F);
    FILLPOLY(5,BT2);
    SETCOLOR(14);
    DRAWPOLY(5,BT2);
    SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,2);
    CASE BT2[I].Y OF
      60:begin OUTTEXTXY(205,65,'MAKE DIR');
        OUTTEXTXY(205,67,' ');
        end;
      110: begin
        OUTTEXTXY(205,115,'CHANGE DIR');
        OUTTEXTXY(205,117,' ');
        end;
      160: begin OUTTEXTXY(205,165,'REMOVE DIR');
        OUTTEXTXY(205,167,' ');
        end;
      210: begin OUTTEXTXY(205,215,'ESCAPE');
        OUTTEXTXY(205,217,' ');
        end;
    end;
  end;

END;
END; {PROCEDURE}
PROCEDURE W23;
BEGIN
  BT22[1].X:=200;          BT22[1].Y:=40;
  BT22[2].X:=400;          BT22[2].Y:=40;
  BT22[3].X:=400;          BT22[3].Y:=80;
  BT22[4].X:=200;          BT22[4].Y:=80;
  BT22[5].X:=200;          BT22[5].Y:=40;
  SETFILLSTYLE(1,5);
  FILLPOLY(5,BT22);
  SETCOLOR(14);
  DRAWPOLY(5,BT22);
  FOR I:-1 TO 4 DO
    BEGIN
      FOR J:-1 TO 5 DO
        BT22[J].Y:=BT22[J].Y+50;
        FILLPOLY(5,BT22);
        SETCOLOR(14);

        DRAWPOLY(5,BT22);

```

```

END;
SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR, 2);
OUTTEXTXY(205, 45, 'RENAME FILE');
OUTTEXTXY(205, 47, ' ');
OUTTEXTXY(205, 95, 'DELETE FILE');
OUTTEXTXY(205, 97, ' ');
OUTTEXTXY(205, 145, 'COPY FILE');
OUTTEXTXY(205, 147, ' ');
OUTTEXTXY(205, 195, 'DUMP FILE');
OUTTEXTXY(235, 197, ' ');
OUTTEXTXY(205, 245, 'ESCAPE');
OUTTEXTXY(205, 247, ' ');
REPEAT
  UNTIL KEYPRESSED;
END; {PROCEDURE}
PROCEDURE W44;
BEGIN
  BT22[1].X:=200;          BT22[1].Y:=60;
  BT22[2].X:=400;          BT22[2].Y:=60;
  BT22[3].X:=400;          BT22[3].Y:=100;
  BT22[4].X:=200;          BT22[4].Y:=100;
  BT22[5].X:=200;          BT22[5].Y:=60;
  SETFILLSTYLE(1, 5);
  FILLPOLY(5, BT22);
  SETCOLOR(14);
  DRAWPOLY(5, BT22);
  FOR I:=1 TO 2 DO
    BEGIN
      FOR J:=1 TO 5 DO
        BT22[J].Y:=BT22[J].Y+50;
        FILLPOLY(5, BT22);
        SETCOLOR(14);

        DRAWPOLY(5, BT22);

      END;
    SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR, 2);
    OUTTEXTXY(205, 65, 'SET');
    OUTTEXTXY(205, 67, ' ');
    OUTTEXTXY(205, 115, 'GET');
    OUTTEXTXY(205, 117, ' ');
    OUTTEXTXY(205, 165, 'ESCAPE');

    REPEAT
      UNTIL KEYPRESSED;
    END; {PROCEDURE}

  PROCEDURE W34(SD: INTEGER; F: INTEGER);
  BEGIN
    SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR, 2);
    IF SD<>40 THEN FOR I:=1 TO 5 DO
      BT2[I].Y:=BT2[I].Y+SD;
      SETFILLSTYLE(1, F);
      FILLPOLY(5, BT2);
      SETCOLOR(14);
      DRAWPOLY(5, BT2);
      SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR, 2);
    CASE BT2[1].Y OF
      40:begin OUTTEXTXY(205, 45, 'RENAME FILE');
              OUTTEXTXY(205, 47, ' ');
              end;
      90:begin OUTTEXTXY(205, 95, 'DELETE FILE');

```

```

end;
140:begin OUTTEXTXY(205,145,'COPY FILE');
      OUTTEXTXY(205,147,'_');
end;
190:begin OUTTEXTXY(205,195,'DUMP FILE');
      OUTTEXTXY(235,197,'_');
end;
240:begin OUTTEXTXY(205,245,'ESCAPE');
      OUTTEXTXY(205,247,'_');
end;
END;
END; {PROCEDURE}
PROCEDURE W(SD:INTEGER;F:INTEGER);
BEGIN
  SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,2);
  IF SD<>60 THEN FOR I:=1 TO 5 DO
    BT2[I].Y:=BT2[I].Y+SD;
    SETFILLSTYLE(1,F);
    FILLPOLY(5,BT2);
    SETCOLOR(14);
    DRAWPOLY(5,BT2);
  SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,2);
  CASE BT2[1].Y OF
60:begin OUTTEXTXY(205,65,'SET');
      OUTTEXTXY(205,67,'_');
end;
110:begin OUTTEXTXY(205,115,'GET');
      OUTTEXTXY(205,117,'_');
end;
160:begin OUTTEXTXY(205,165,'ESCAPE');
      OUTTEXTXY(205,167,'_');
end;
END;
END; {PROCEDURE}

PROCEDURE W3(SD:INTEGER;F:INTEGER);
BEGIN
  SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,2);
  IF SD<>10 THEN FOR I:=1 TO 5 DO
    BT[I].Y:=BT[I].Y+SD;
    SETFILLSTYLE(1,F);
    FILLPOLY(5,BT);
    SETCOLOR(14);
    DRAWPOLY(5,BT);
  CASE BT[1].Y OF
10: begin OUTTEXTXY(30,20,'DIRICTORY');
      OUTTEXTXY(30,22,'_');
end;
70: begin OUTTEXTXY(30,80,'FILE TOOLS');
      OUTTEXTXY(50,82,'_');
end;
130: begin OUTTEXTXY(35,140,'FIND FILE');
      OUTTEXTXY(35,142,'_');
end;
190: begin OUTTEXTXY(35,200,'TIME [ SET / GET ]');
      OUTTEXTXY(35,202,'_');
end;
250: begin OUTTEXTXY(35,260,'DATE [ SET / GET ]');
      OUTTEXTXY(35,262,'_');
end;
310: begin OUTTEXTXY(35,320,'ATTRIB [ SET/GET ]');
      OUTTEXTXY(35,322,'_');
end;

```



```

OUTTEXTXY(45,382,'_');
end;
430: begin OUTTEXTXY(35,440,'QUIT');
OUTTEXTXY(35,442,'_');
END; end;
END; {PROCEDURE}

PROCEDURE CHECKHOT(C1:CHAR); FORWARD;
PROCEDURE CHECK(M1:INTEGER); FORWARD;
PROCEDURE CC ;
BEGIN
  BT[1].X:=20; BT[1].Y:=10;
  BT[2].X:=330; BT[2].Y:=10;
  BT[3].X:=330; BT[3].Y:=50;
  BT[4].X:=20; BT[4].Y:=50;
  BT[5].X:=20; BT[5].Y:=10;
  W3(10,9);
  M1:=10;
REPEAT
  if tty=1 then begin
    tty:=0;
    c:='L';
    CHECKHOT (C);
  END ELSE begin C:=READKEY;CHECKHOT(C);END;
CASE ORD(C) OF
  0: BEGIN
    C:=READKEY;
CASE ORD(C) OF
  72:BEGIN IF M1>10 THEN W3(10,5);
    IF M1>10 THEN BEGIN M1:=M1-60;
      W3(-60,9);END;
    END;
  80:BEGIN IF M1<430 THEN W3(10,5);
    IF M1<430 THEN BEGIN M1:=M1+60;W3(60,9);END;
    END;
    END;
    END;
    END;
13:CHECK(M1);
END; {CASE}
UNTIL (C='Q');
END; {PROCEDURE}
PROCEDURE CHECK1(M11:INTEGER); FORWARD;
PROCEDURE CHECK1HOT(C11:CHAR); FORWARD;
PROCEDURE CC1;
BEGIN
  BT2[1].X:=200; BT2[1].Y:=60;
  BT2[2].X:=400; BT2[2].Y:=60;
  BT2[3].X:=400; BT2[3].Y:=100;
  BT2[4].X:=200; BT2[4].Y:=100;
  BT2[5].X:=200; BT2[5].Y:=60;
  W33(60,9);
  M11:=60;
REPEAT
  C:=READKEY;CHECK1HOT(C);
CASE ORD(C) OF
  0: BEGIN
    C:=READKEY;
CASE ORD(C) OF
  72:BEGIN IF M11>60 THEN W33(60,5);
    IF M11>60 THEN BEGIN M11:=M11-50;
      W33(-50,9);END;
    END;
    END;
    END;
    END;

```

```

        IF M11<210 THEN BEGIN M11:=-M11+50;W33(50,9);END;
                                END;
                                END;
                                END;
13:CHECK1(M11);
                                END;    {CASE}
UNTIL (C='Q');
END; {PROCEDURE}

PROCEDURE CHECK2(M11:INTEGER);FORWARD;
PROCEDURE CHECK2HOT(C11:CHAR);FORWARD;
PROCEDURE CC2;
BEGIN
    BT2[1].X:=-200;          BT2[1].Y:=40;
    BT2[2].X:=-400;          BT2[2].Y:=40;
    BT2[3].X:=-400;          BT2[3].Y:=80;
    BT2[4].X:=-200;          BT2[4].Y:=80;
    BT2[5].X:=-200;          BT2[5].Y:=40;
    W34(40,9);
    M11:=-40;
REPEAT
C:=-READKEY;CHECK2HOT(C);
CASE ORD(C) OF
    0: BEGIN
        C:=-READKEY;
        CASE ORD(C) OF
            72:BEGIN IF M11>40 THEN W34(40,5);
                IF M11>40 THEN BEGIN M11:=-M11-50;
                    W34(-50,9);END;
                END;
            80:BEGIN IF M11<240 THEN W34(40,5);
                IF M11<240 THEN BEGIN M11:=-M11+50;W34(50,9);END;
                END;
            END;
            END;
            END;
13:CHECK2(M11);
                                END;    {CASE}
UNTIL (C='Q');
END; {PROCEDURE}

PROCEDURE CHECK3(M11:INTEGER);FORWARD;
PROCEDURE CHECK4(M11:INTEGER);FORWARD;
PROCEDURE CHECK5(M11:INTEGER);FORWARD;
PROCEDURE CHECK3HOT(C11:CHAR);FORWARD;
PROCEDURE CHECK4HOT(C11:CHAR);FORWARD;
PROCEDURE CHECK5HOT(C11:CHAR);FORWARD;

PROCEDURE CC3(N:INTEGER);
BEGIN
    BT2[1].X:=-200;          BT2[1].Y:=60;
    BT2[2].X:=-400;          BT2[2].Y:=60;
    BT2[3].X:=-400;          BT2[3].Y:=100;
    BT2[4].X:=-200;          BT2[4].Y:=100;
    BT2[5].X:=-200;          BT2[5].Y:=60;
    W(60,9);
    M11:=-60;
REPEAT
C:=-READKEY;
CASE N OF
    1:CHECK3HOT(C);
    2:CHECK4HOT(C);
    3:CHECK5HOT(C);
    END;    {CASE}

```

```

0:BEGIN
  C:=READKEY;
  CASE ORD(C) OF
    72:BEGIN
      IF M11>60 THEN W(60,5);
      IF M11>60 THEN
        BEGIN
          M11:=M11-50;
          W(-50,9);
        END;
      END;
    80:BEGIN
      IF M11<160 THEN W(60,5);
      IF M11<160 THEN
        BEGIN
          M11:=M11+50;
          W(50,9);
        END;
      END;
    END; {CASE}
  END;
13:CASE N OF
  1:CHECK3(M11);
  2:CHECK4(M11);
  3:CHECK5(M11);
  END; {CASE}
  END; {CASE}
UNTIL (C='Q');
END; {PROCEDURE}
{*****}
PROCEDURE DIRICTORY ;
BEGIN
  BT1[1].X:=180;          BT1[1].Y:=50;
  BT1[2].X:=430;          BT1[2].Y:=50;
  BT1[3].X:=430;          BT1[3].Y:=300;
  BT1[4].X:=180;          BT1[4].Y:=300;
  BT1[5].X:=180;          BT1[5].Y:=50;
  SETFILLSTYLE(1,4);
  FILLPOLY(5,BT1);
  SETCOLOR(14);
  DRAWPOLY(5,BT1);
  SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,4);
  OUTTEXTXY(220,255,'DIRICTORY');
  W22;
  CC1;
  SECOND_MENU;
  CC;
END; {PROCEDURE}
{*****}
PROCEDURE FILE_TOOLS;
BEGIN
  BT1[1].X:=180;          BT1[1].Y:=30;
  BT1[2].X:=430;          BT1[2].Y:=30;
  BT1[3].X:=430;          BT1[3].Y:=315;
  BT1[4].X:=180;          BT1[4].Y:=315;
  BT1[5].X:=180;          BT1[5].Y:=30;
  SETFILLSTYLE(1,4);
  FILLPOLY(5,BT1);
  SETCOLOR(14);
  DRAWPOLY(5,BT1);

  SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,4);
  OUTTEXTXY(220,280,'FILE TOOLS');

```

```

CC2;

SECOND_MENU;
CC;
END; {PROCEDURE}

PROCEDURE TIME;
BEGIN
  BT1[1].X:=180;          BT1[1].Y:=50;
  BT1[2].X:=430;          BT1[2].Y:=50;
  BT1[3].X:=430;          BT1[3].Y:=300;
  BT1[4].X:=180;          BT1[4].Y:=300;
  BT1[5].X:=180;          BT1[5].Y:=50;
  SETFILLSTYLE(1,4);
  FILLPOLY(5,BT1);
  SETCOLOR(14);
  DRAWPOLY(5,BT1);
  SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,5);
  OUTTEXTXY(235,230,'TIME');
  W44;
  CC3(1);

  SECOND_MENU;
  CC;
  END; {PROCEDURE}

PROCEDURE DATE;
BEGIN
  BT1[1].X:=180;          BT1[1].Y:=50;
  BT1[2].X:=430;          BT1[2].Y:=50;
  BT1[3].X:=430;          BT1[3].Y:=300;
  BT1[4].X:=180;          BT1[4].Y:=300;
  BT1[5].X:=180;          BT1[5].Y:=50;
  SETFILLSTYLE(1,4);
  FILLPOLY(5,BT1);
  SETCOLOR(14);
  DRAWPOLY(5,BT1);

  SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,5);
  OUTTEXTXY(235,230,'DATE');
  W44;
  CC3(2);

  SECOND_MENU;
  CC;
  END; {PROCEDURE}

PROCEDURE ATTRIB;
BEGIN
  BT1[1].X:=180;          BT1[1].Y:=50;
  BT1[2].X:=430;          BT1[2].Y:=50;
  BT1[3].X:=430;          BT1[3].Y:=300;
  BT1[4].X:=180;          BT1[4].Y:=300;
  BT1[5].X:=180;          BT1[5].Y:=50;
  SETFILLSTYLE(1,4);
  FILLPOLY(5,BT1);
  SETCOLOR(14);
  DRAWPOLY(5,BT1);

```



```

SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,5);
OUTTEXTXY(235,230,'ATTRIB');
W44;
CC3(3);

SECOND_MENU;
CC;
END; {PROCEDURE}
{*****}

PROCEDURE FIND_FILE;
VAR
  QF,WF:INTEGER;

procedure set_dta;assembler;
asm
  push ax
  push bx
  push cx
  push dx
  mov ah,$1a
  mov dx,offset ds:dta;
  int $21
  pop dx
  pop cx
  pop bx
  pop ax
end;

procedure find_first;assembler;
asm
  push ax
  push bx
  push cx
  push dx
  mov cx,$ff
  mov ah,$4e
  mov dx,offset ds:filename;
  inc dx
  int $21
  push ax
  mov ax,0
  adc al,0
  mov byte ptr ds:f_ound,al
  pop ax
  pop dx
  pop cx
  pop bx
  pop ax
end;

procedure find_next;assembler;
asm
  push ax
  push bx
  push cx
  push dx
  mov cx,$4f
  mov ah,$4f
  int $21
  push ax
  mov ax,0
  adc al,0
  mov byte ptr ds:f_ound,al

```

```

    pop  dx
    pop  cx
    pop  bx
    pop  ax
end;

begin

WE:
  CN[1].X:=295;      CN[1].Y:=330;
  CN[2].X:=610;      CN[2].Y:=330;
  CN[3].X:=610;      CN[3].Y:=355;
  CN[4].X:=295;      CN[4].Y:=355;
  CN[5].X:=295;      CN[5].Y:=330;
  {-----}
  SETFILLSTYLE(1,0);
  FILLPOLY(4,CN);
  SETCOLOR(14);
  DRAWPOLY(4,CN);

FILES_num:=0;
SETTEXTSTYLE(1,0,2);
OUTTEXTXY(30,328,'PLEASE ENTER FILE NAME: ');
GOTOXY(39,22);
READLN (filename);
for i:=1 to 60 do dta[i]:=$20;
set dta;
find first;
XF1:=30;YF1:=60;xx:=160;
WW;
while f_ound =0 do
begin
  XF:=XF1;
  FILES_num:=FILES_num+1;
  c1:=30;
  i:=31;
  SETTEXTSTYLE(2,0,5);
  while c1<> 0 do begin
    c1:=dta [i];
    OUTTEXTXY(XF,YF1,chr (dta [i]));
    XF:=XF+11;
    i:=i+1;
  end;
  j:=dta[21];j:=-j and $10 ;
  if j>0 then OUTTEXTXY(xx,YF1,'[d]'{chr (dta [i])});
  find next;
  YF1:=YF1+15;
  IF YF1=300 THEN BEGIN
    YF1:=60;
    XF1:=XF1+150;
    xx:=xx+150;
    if xx>600 then xx:=xx-20;
    IF XF1>=500 THEN BEGIN
      READLN;
      FOR QF:=4 TO 77 DO
      FOR WF:=15 TO 22 DO
      BEGIN
        OUTTEXTXY(QF,WF,' ');
        YF1:=60;XF1:=30;
      END;WW;
    END;
  END;
END;

```

```

SETTEXTSTYLE(1,0,2);
OUTTEXTXY(30,400,'Number of files found= ');
if FILES_num<=9 then conv(FILES_num,0)
else if (99>FILES_num) and (FILES_num>9)
then conv(FILES_num,1)
else if (999>FILES_num) and (FILES_num>99)
then conv(FILES_num,2)
else if FILES_num>999
then conv(FILES_num,3);
xf1:=270;
for i:=1 to 4 do
OUTTEXTXY(xf1+10*i,400,CHR(num[i]));

SETFILLSTYLE(1,blue);
bar(30,435,380,465);
OUTTEXTXY(35,440,'press any key to continue...');
c:=READkey;
for i:=1 to 60 do dta[i]:=$20;
FOR QF:=1 TO LENGTH (FILENAME) DO FILENAME[QF]:=' ';
FILENAME [0]:=CHR (0);

SECOND_MENU;CC;
end;

{*****}
PROCEDURE RENAME_FILE;

procedure ren;assembler;
asm
    push ax
    push dx
    push di
    mov ah,$56
    lea dx,ds:s
    lea di,es:d
    inc dx
    inc di
    int $21
    mov word ptr ds:err,ax
    mov al,0
    adc al,0
    mov byte ptr ds:b,al
    pop di
    pop dx
    pop ax
end;
begin
    WE;

    CN[1].X:=275;      CN[1].Y:=330;
    CN[2].X:=610;      CN[2].Y:=330;
    CN[3].X:=610;      CN[3].Y:=355;
    CN[4].X:=275;      CN[4].Y:=355;
    CN[5].X:=275;      CN[5].Y:=330;
    {-----}
    SETFILLSTYLE(1,0);
    FILLPOLY(4,CN);
    SETCOLOR(14);
    DRAWPOLY(4,CN);
    OUTTEXTXY(30,328,'ENTER OLD FILE NAME : ');
    gotoxy (37,22);
    readln (s);
    s:=s+chr (0);

```

```

        CN[2].X:=610;          CN[2].Y:=360;;
        CN[3].X:=610;          CN[3].Y:=385;
        CN[4].X:=275;          CN[4].Y:=385;
        CN[5].X:=275;          CN[5].Y:=360;
    {-----}

        SETFILLSTYLE(1,0);
        FILLPOLY(4,CN);
        SETCOLOR(14);
        DRAWPOLY(4,CN);
        OUTTEXTXY(30,363,'ENTER NEW FILE NAME : ');
        gotoxy (37,24);
        readln (D);
        D:=D+chr (0) *
        ren;
        if b=1 then
            case err of
2:BEGIN OUTTEXTXY(30,368,S);
        OUTTEXTXY(230,400,'is not found!!!') END;
3:OUTTEXTXY(230,400,'Path is not found');
5:OUTTEXTXY(230,400,'Access denide');
11:OUTTEXTXY(230,400,'Not the same device');
        end
        ELSE OUTTEXTXY(230,400,'file has been renamed');
        SETFILLSTYLE(1,blue);
        bar(30,435,380,465);
        OUTTEXTXY(35,440,'press any key to ccontinue...');
        c:=READkey;
        WE;

END; {PROCEDURE}

{*****}
PROCEDURE DUMP_FILE;
PROCEDURE open_file; assembler;
asm
    push ax
    push bx
    push cx
    push dx
    mov  ah,$3d
    mov  al,$42
    lea  dx,word ptr ds:filename
    inc  dx
    int  $21
    push ax
    mov  ax,0
    adc  al,0
    mov  byte ptr ds:b,al
    pop  ax
    mov  word ptr ds:err,ax
    mov  word ptr ds:handel,ax
    pop  dx
    pop  cx
    pop  bx
    pop  ax
end;

procedure close_file; assembler;
asm
    push ax
    push bx

```



```

push dx
mov ah,$3e
mov bx,word ptr ds:handel
int $21
pop dx
pop cx
pop bx
pop ax
end;
procedure read_file; assembler;
label ll,o_k,_eof;
asm
push ax
push bx
push cx
push dx
mov ah,$3f
mov bx,word ptr ds:handel
mov cx,1
lea dx,word ptr ds:buf1
inc dx
int $21
jc ll
cmp ax,0
je _eof
jmp o_k
ll:
mov word ptr ds:err,ax
jmp o_k
_eof:
mov ah,1
mov byte ptr ds:e_f,ah
o_k:
pop dx
pop cx
pop bx
pop ax
end;
begin
wz;
CN[1].X:=295; CN[1].Y:=330;
CN[2].X:=610; CN[2].Y:=330;
CN[3].X:=610; CN[3].Y:=355;
CN[4].X:=295; CN[4].Y:=355;
CN[5].X:=295; CN[5].Y:=330;
{-----}
SETFILLSTYLE(1,0);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);

SETTEXTSTYLE(1,0,2);
OUTTEXTXY(30,328,'PLEASE ENTER FILE NAME: ');
GOTOXY(39,22);
READLN (filename);
filename:=filename+chr (0);
e_f:=0;

open_file;
IF B= 1 THEN BEGIN we;
IF ERR = 2 THEN

```

```

OUTTEXTXY(230,400,'is not found !!!') END
else
IF ERR = 3 THEN
OUTTEXTXY(30,358, 'Access denied') else
BEGIN
OUTTEXTXY(230,400,'can not be opened ');
OUTTEXTXY(30,358,filename); END;
delay(2000);
END
ELSE BEGIN

xf1:=35;yf1:=45;
xx:=420;
settextstyle (2,0,6);
ww;

repeat
    buf1:='';
    s:='';
    read_file;
    c:=buf1[1];
    a:=ord (c);
    ws1:=a div 16;
    b:=a mod 16;
    if ws1>9 then s:=s+chr (ws1-9+$40) else
    s:=s+chr ($30+ws1);
    if b>9 then s:=s+chr (b-9+$40) else
    s:=s+chr ($30+b);

    outtextxy (xf1,yf1,s);
    if ord (c)>=32 then outtextxy (xx,yf1,c) else
    outtextxy (xx,yf1,'. ');
    xf1:=xf1+25;
    xx:=xx+12;
    if xf1>=400 then begin
        yf1:=yf1+15;
        xx:=420;
        xf1:=35;
        if yf1>=290 then
        begin c:=readkey;
            if c<>' ' then e_f:=1;
            yf1:=45;
            ww;
        end;
    end;
    { end;}
until e_f=1;
end;
close_file;
tty:=1;c:='Q';
END;{PROCEDURE}
{*****}

PROCEDURE DELETE_FILE;
PROCEDURE DELETE;ASSEMBLER;
ASM
    PUSH AX
    PUSH DX
    MOV AH,$41

```

```

INC DX
INT $21
PUSH AX
MOV AL,0
ADC AL,0
MOV BYTE PTR DS:B,AL;
POP AX
MOV WORD PTR DS:ERR,AX
POP DX
POP AX
END;
BEGIN
WE;

CN[1].X=-225;      CN[1].Y=-330;
CN[2].X=-610;      CN[2].Y=-330;
CN[3].X=-610;      CN[3].Y=-355;
CN[4].X=-225;      CN[4].Y=-355;
CN[5].X=-225;      CN[5].Y=-330;
{-----}
SETFILLSTYLE(1,0);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);

OUTTEXTXY(30,328,'ENTER FILE NAME : ');
gotoxy (30,22);
readln (s);
s:=s+chr (0);
DELETE;
IF B= 1 THEN BEGIN
  IF ERR = 2 THEN
    BEGIN OUTTEXTXY(30,358,S);
    OUTTEXTXY(230,400,'is not found!!!')
    END else
  IF ERR = 3 THEN
    OUTTEXTXY(30,358, 'Access denied') else
    BEGIN OUTTEXTXY(230,400,'can not be deleted ');
    OUTTEXTXY(30,358,s);      END
    END
  ELSE BEGIN OUTTEXTXY(30,358,S);
  OUTTEXTXY(230,400,'has been deleted');
  END;

SETFILLSTYLE(1,blue);
bar(30,435,380,465);
OUTTEXTXY(35,440,'press any key to ccontinue...');
c:=READkey;
WE;

END;
{*****}

PROCEDURE COPY_FILE;
procedure _copy; assembler;
label again,_end,11,12,13,exit;
asm

      push    ax
      push    bx
      push    dx

```

```

push    si
push    di

mov     ah,$3d
mov     al,2
mov     cx,0
lea     dx,ds:file_1
inc     dx
int     $21
push    ax
mov     ax,0
adc     ax,0
cmp     ax,0
jnz     l3
pop     ax
mov     ds:file_1_handel,ax
jmp     l2
13:pop  ax
mov     ds:err1,ax
12:mov  ah,$3c
mov     cx,0
lea     dx,ds:file_2
inc     dx
int     $21
push    ax
mov     ax,0
adc     ax,0
cmp     ax,0
jnz     l1
pop     ax
mov     ds:file_2_handel,ax
jmp     again
11:pop  ax
mov     ds:err2,ax

jmp     exit
again:  cmp     ds:err1,0
jne     exit

mov     ah,$3f
mov     bx,ds:file_1_handel
mov     cx,1
lea     dx,ds:buf
int     $21
cmp     ax,0
je      _end

mov     ah,$40
mov     bx,ds:file_2_handel
mov     cx,1
lea     dx,ds:buf
int     $21
jmp     again

_end:   mov     ah,$3e
mov     bx,ds:file_1_handel
int     $21

mov     ah,$3e
mov     bx,ds:file_2_handel

```



```

exit:  pop      di
       pop      si
       pop      cx
       pop      dx
       pop      bx
       pop      ax
       end;

BEGIN
  WE;

  CN[1].X:=275;      CN[1].Y:=330;
  CN[2].X:=610;      CN[2].Y:=330;
  CN[3].X:=610;      CN[3].Y:=355;
  CN[4].X:=275;      CN[4].Y:=355;
  CN[5].X:=275;      CN[5].Y:=330;
  {-----}
  SETFILLSTYLE(1,0);
  FILLPOLY(4,CN);
  SETCOLOR(14);
  DRAWPOLY(4,CN);

  OUTTEXTXY(30,328,'NAME OF SOURCE FILE :');
  gotoxy (37,22);
  READLN (file_1);
  file_1:=file_1+CHR (0);
  CN[1].X:=275;      CN[1].Y:=360;
  CN[2].X:=610;      CN[2].Y:=360;
  CN[3].X:=610;      CN[3].Y:=385;
  CN[4].X:=275;      CN[4].Y:=385;
  CN[5].X:=275;      CN[5].Y:=360;
  {-----}

  SETFILLSTYLE(1,0);
  FILLPOLY(4,CN);
  SETCOLOR(14);
  DRAWPOLY(4,CN);
  OUTTEXTXY(30,363,'NAME OF DESTIN. FILE :');
  gotoxy (37,24);
  READLN (file_2);
  file_2:=file_2+CHR (0);
  ERR1:=0; ERR2:=0;
  COPY;
  IF (ERR1<>0) and (err2=0) THEN BEGIN
    WE;
    if err1=2 then
      OUTTEXTXY(230,360,'source path not found');
    if err1=4 then
      OUTTEXTXY(230,360,'access denied');
    end
  else
    IF (ERR1=0) and (err2<>0) THEN
      BEGIN
        WE;
        if err2=2 then
          OUTTEXTXY(230,360,'destination path not found');
        if err2=4 then OUTTEXTXY(230,360,'access denied');
        end
      else IF (ERR1<>0) and (err2<>0) THEN BEGIN
        WE;
        if err1=2 then
          OUTTEXTXY(180,360,'source path not found');
        if err1=4 then

```

```

if err2=2 then
  OUTTEXTXY(150,400,'destination path not found');
if err2=4 then
  OUTTEXTXY(150,400,'destination access denied');
  end
else IF (ERR1=0) and (err2=0) THEN
  OUTTEXTXY(230,400,'file has been copied');
  SETFILLSTYLE(1,blue);
  bar(30,435,380,465);
  OUTTEXTXY(35,440,'press any key to ccntinue...');
  c:=READkey;
  WE;

END; {PROCEDURE}
{*****}

```

```

PROCEDURE MAKE_DIR;

```

```

  procedure m_dir; assembler;
  asm
    push ax
    push dx
    mov ah,$39
    lea dx,ds:s;
    inc dx
    int $21
    mov word ptr ds:err,ax
    mov al,0
    adc al,0
    mov byte ptr ds:b,al
    pop dx
    pop ax
  end;
  BEGIN
  WE;

  CN[1].X:=300;      CN[1].Y:=330;
  CN[2].X:=610;      CN[2].Y:=330;
  CN[3].X:=610;      CN[3].Y:=355;
  CN[4].X:=300;      CN[4].Y:=355;
  CN[5].X:=300;      CN[5].Y:=330;
  {-----}
  SETFILLSTYLE(1,0);
  FILLPOLY(4,CN);
  SETCOLOR(14);
  DRAWPOLY(4,CN);
  OUTTEXTXY(30,330,'ENTER PATH & DIR NAME :') ;

  gotoxy (40,22);
  READLN (S);
  s:=s+chr (0);
  m_dir;
  if b=1 then
    begin
  if err=3 then
    OUTTEXTXY(30,360,'path is not found')
  else
    if err=5 then
      OUTTEXTXY(30,360,'access denied');
    end
  ELSE OUTTEXTXY(30,360,' [OK] ');
  SETFILLSTYLE(1,blue);
  bar(30,435,380,465);

```

```

C-READKEY;
WE;

END; {PROCEDURE}

PROCEDURE CHANGE DIR;
procedure set_drive; assembler;
asm
    pop     ax
end;
procedure ch_dir; assembler;
asm
    push    ax
    push    dx
    mov     ah,$3b
    lea     dx,ds:s
    INC     DX
    int     $21
    mov     al,0
    adc     al,0
    mov     byte ptr ds:b,al
    pop     dx
    pop     ax
end;
BEGIN
    WE;

    CN[1].X:=300;      CN[1].Y:=330;
    CN[2].X:=610;      CN[2].Y:=330;
    CN[3].X:=610;      CN[3].Y:=355;
    CN[4].X:=300;      CN[4].Y:=355;
    CN[5].X:=300;      CN[5].Y:=330;
    {=====}
    SETFILLSTYLE(1,0);
    FILLPOLY(4,CN);
    SETCOLOR(14);
    DRAWPOLY(4,CN);
    OUTTEXTXY(30,330,'ENTER PATH & DIR NAME :') ;
    for i:=0 to 60 do s[i]:=chr($20);
    s[0]:=chr(0);
    s:=chr(0);
    gotoxy (40,22);
    READLN (S);
    s:=s+chr(0);
    if s[2]=':' then
        case 's[1] of
            'A','a':drv_no:=0;
            'B','b':drv_no:=1;
            'C','c':drv_no:=2;
        END;
    SET_DRIVE;
    ch_dir;
    if b=1 then
        OUTTEXTXY(30,360,'path is not found')
    ELSE OUTTEXTXY(30,360,' [OK] ');
    SETFILLSTYLE(2,3);
    bar(340,10,628,50);

```

```

path[0]:=chr(0);
path:=chr(0);
get_path;
SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR, 2);

XF1:=345;
I:=0;
WHILE ORD (path[I])<>0 DO BEGIN
OUTTEXTXY(XF1,15,path[I]);
XF1:=XF1+11;
I:=I+1;
END;
path[0]:=chr(0);
s[0]:=chr(0);
SETFILLSTYLE(1,blue);
bar(30,435,380,465);
OUTTEXTXY(35,440,'press any key to ccntinue...');
c:=READkey;
WE;

END;
{*****}

PROCEDURE REMOVE_DIR;
procedure d_dir; assembler;
asm
    push ax
    push dx
    mov ah,$3a
    lea dx,ds:s;
    inc dx
    int $21
    mov word ptr ds:err,ax
    mov al,0
    adc al,0
    mov byte ptr ds:b,al
    pop dx
    pop ax
end;
begin
    WE;

    CN[1].X:=300;      CN[1].Y:=330;
    CN[2].X:=610;      CN[2].Y:=330;
    CN[3].X:=610;      CN[3].Y:=355;
    CN[4].X:=300;      CN[4].Y:=355;
    CN[5].X:=300;      CN[5].Y:=330;
    {=====}
    SETFILLSTYLE(1,0);
    FILLPOLY(4,CN);
    SETCOLOR(14);
    DRAWPOLY(4,CN);

    OUTTEXTXY(30,330,'ENTER PATH & DIR NAME :') ;

    gotoxy (40,22);
    READLN (S);
    s:=s+chr (0);
    D_dir;
    if b=1 then
        begin
            if err=3 then

```



```

else
if err=5 then
OUTTEXTXY(30,360,'access denied');
if err=16 then
OUTTEXTXY(30,360,'can not delete the woking directory');
end
ELSE OUTTEXTXY(30,360,' [OK] ');
SETFILLSTYLE(1,blue);
bar(30,435,380,465);
OUTTEXTXY(35,440,'press any key to ccntinue...');
c:=READkey;
WE;

END;{PROCEDURE}
{*****}
PROCEDURE SET TIME;
procedure settm;assembler;
asm
push ax
push cx
push dx
mov ah,$2d
mov ch,byte ptr ds:hor
mov cl,byte ptr ds:min
mov dh,byte ptr ds:sec
mov dl,byte ptr ds:hsc
int $21
mov byte ptr ds:b,al
pop dx
pop cx
pop ax
end;
BEGIN
we;

CN[1].X:=320;      CN[1].Y:=380;
CN[2].X:=610;      CN[2].Y:=380;
CN[3].X:=610;      CN[3].Y:=405;
CN[4].X:=320;      CN[4].Y:=405;
CN[5].X:=320;      CN[5].Y:=380;
{=====}
SETFILLSTYLE(1,0);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);

OUTTEXTXY(40,380,'ENTET NEW SYSTEM TIME :');
OUTTEXTXY(353,380,' : : : p');
HOR:=0;MIN:=0;SEC:=0;HSC:=0;I:=0;J:=10;XF1:=350;
C:=READKEY;
WHILE (I<>8) AND (ORD(C)<>' ') DO
BEGIN
OUTTEXTXY(XF1,380,C);
IF I<2 THEN BEGIN
NUM2:=0;
NUM2:=ORD(C)-$30;
NUM2:=NUM2*J;
HOR:=HOR+NUM2;
END;
IF (I<4) AND (I>=2) THEN BEGIN
NUM2:=0;
NUM2:=ORD(C)-$30;

```

```

MIN:=MIN+NUM2;
END;
IF (I<6) AND (I>=4) THEN BEGIN
    NUM2:=0;
    NUM2:=ORD(C)-$30;
    NUM2:=NUM2*J;
    SEC:=SEC+NUM2;
    END;
IF (I<8) AND (I>=6) THEN BEGIN
    NUM2:=0;
    NUM2:=ORD(C)-$30;
    NUM2:=NUM2*J;
    HSC:=HSC+NUM2;
    END;
XF1:=XF1+10;
IF I MOD 2<>0 THEN BEGIN
    J:=10;XF1:=XF1+10;END ELSE J:=1;
I:=I+1;
C:=READKEY
END;

SETTM;
IF B=0 THEN
OUTTEXTXY(250,430,'time is set now');
else
    OUTTEXTXY(250,430,'Incorrect Time');
delay(2000);
SETFILLSTYLE(1,blue);
bar(30,435,380,465);
OUTTEXTXY(35,440,'press any key to ccntinue...');
c:=READkey;
WE;

END; {PROCEDURE}
{*****}
PROCEDURE GET_DATE;

procedure getdt; assembler;
asm
    push ax
    push cx
    push dx
    mov ah,$2a
    int $21
    mov byte ptr ds:dy,al
    mov word ptr ds:yy,cx
    mov byte ptr ds:mm,dh
    mov byte ptr ds:dd,dl
    pop dx
    pop cx
    pop ax
end;
BEGIN
we;
OUTTEXTXY(40,350,'Today is :');
CASE DY OF
    0: OUTTEXTXY(150,350,'SUN');
    1: OUTTEXTXY(150,350,'MON');
    2: OUTTEXTXY(150,350,'TUE');
    3: OUTTEXTXY(150,350,'WED');
    4: OUTTEXTXY(150,350,'THU');
    5: OUTTEXTXY(150,350,'FRI');

```

```

END;
OUTTEXTXY(40,380,'The current system date is :');
GETdt; xf1:=350;
CONV(dd,1);FOR I:=1 TO 4 DO
OUTTEXTXY(XF1+10*I,380,CHR(NUM[I]));
OUTTEXTXY(405,380,'/'); xf1:=390;
CONV(mm,1);FOR I:=1 TO 4 DO
OUTTEXTXY(XF1+10*I,380,CHR(NUM[I]));
OUTTEXTXY(440,380,'/');xf1:=445;
CONV(yy,3);FOR I:=1 TO 4 DO
OUTTEXTXY(XF1+10*I,380,CHR(NUM[I]));
xf1:=480;
SETFILLSTYLE(1,blue);
bar(30,435,380,465);
OUTTEXTXY(35,440,'press any key to ccontinue...');
c:=READkey;
WE;

```

```

END; {PROCEDURE}

```

```

{*****}

```

```

PROCEDURE SET DATE;
procedure setdt;assembler;
asm

```

```

push ax
push cx
push dx
mov ah,$2b
mov cx,word ptr ds:yy
mov dh,byte ptr ds:mm
mov dl,byte ptr ds:dd
int $21
mov byte ptr ds:b,al
pop dx
pop cx
pop ax

```

```

end;
BEGIN
we;

```

```

CN[1].X:=320;      CN[1].Y:=380;
CN[2].X:=610;      CN[2].Y:=380;
CN[3].X:=610;      CN[3].Y:=405;
CN[4].X:=320;      CN[4].Y:=405;
CN[5].X:=320;      CN[5].Y:=380;

```

```

{-----}
SETFILLSTYLE(1,0);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);

```

```

OUTTEXTXY(40,380,'ENTET NEW SYSTEM DATE :');
OUTTEXTXY(353,380,' : : ');
YY:=0;MM:=0;DD:=0;I:=0;J:=10;
XF1:=350;yy1:=0;yy2:=0;
C:=READKEY;
WHILE (I<>8) AND (ORD(C)<>13) DO
BEGIN
OUTTEXTXY(XF1,380,C);
IF I<2 THEN BEGIN
NUM2:=0;
NUM2:=ORD(C)-$30;

```

```

DD:=DD+NUM2;
END;
IF (I<4) AND (I>=2) THEN BEGIN
    NUM2:=0;
    NUM2:=ORD(C)-$30;
    NUM2:=NUM2*J;
    MM:=MM+NUM2;
    END;
IF (I<6) AND (I>=4) THEN BEGIN
    NUM2:=0;
    NUM2:=ORD(C)-$30;
    NUM2:=NUM2*J;
    YY1:=YY1+NUM2;
    END;
IF (I<8) AND (I>=6) THEN BEGIN ..
    NUM2:=0;
    NUM2:=ORD(C)-$30;
    NUM2:=NUM2*J;
    YY2:=YY2+NUM2;
    END;
XF1:=-XF1+10;
IF I MOD 2<>0 THEN BEGIN
    J:=10;if i<>5 then XF1:=XF1+10;END
    ELSE J:=1;
    I:=I+1;
YY:=YY1*100 +YY2;
C:=READKEY
END;

SETdt;
IF B=0 THEN
    OUTTEXTXY(250,430,'date is set now') else
    OUTTEXTXY(250,430,'Incorrect date ');
delay(2000);
SETFILLSTYLE(1,blue);
bar(30,435,380,465);
OUTTEXTXY(35,440,'press any key to ccntinue...');
c:=READkey;
WE;

END;{PROCEDURE}
{*****}
PROCEDURE GET_TIME;
procedure gettm;assembler;
asm
    push ax
    push cx
    push dx
    mov ah,$2c
    int $21
    mov byte ptr ds:hor,ch
    mov byte ptr ds:min,cl
    mov byte ptr ds:sec,dh
    mov byte ptr ds:hsc,dl
    pop dx
    pop cx
    pop ax
end;
BEGIN
we;
OUTTEXTXY(40,380,'The current system time is :');
GETTM; xf1:=350;

```



```

OUTTEXTXY(Xr+10*i,380,CHR(NUM[I]));
OUTTEXTXY(405,380,':'); xfl:=385;
CONV(MIN,1);FOR I:=1 TO 4 DO
OUTTEXTXY(XF1+10*i,380,CHR(NUM[I]));
OUTTEXTXY(445,380,':');xfl:=430;
CONV(SEC,1);FOR I:=1 TO 4 DO
OUTTEXTXY(XF1+10*i,380,CHR(NUM[I]));xfl:=480;
CONV(HSC,1);FOR I:=1 TO 4 DO
OUTTEXTXY(XF1+10*i,380,CHR(NUM[I]));
OUTTEXTXY(530,380,'p');
SETFILLSTYLE(1,blue);
bar(30,435,380,465);
OUTTEXTXY(35,440,'press any key to continue...');
c:=READkey;
we;

END;{PROCEDURE}
{*****}
PROCEDURE SET_ATTRIB;
PROCEDURE SETAT;ASSEMBLER;
ASM
PUSH AX
PUSH DX
MOV AH,$43
MOV AL,1
LEA DX,DS:S
INC DX
MOV CX,WORD PTR DS:ATT
INT $21
PUSH AX
MOV AL,0
ADC AL,0
MOV BYTE PTR DS:B,AL;
POP AX
MOV WORD PTR DS:ERR,AX
POP DX
POP AX
END;
BEGIN
we2;

CN[1].X:=210; CN[1].Y:=330;
CN[2].X:=610; CN[2].Y:=330;
CN[3].X:=610; CN[3].Y:=352;
CN[4].X:=210; CN[4].Y:=352;
CN[5].X:=210; CN[5].Y:=330;
{-----}
SETFILLSTYLE(1,0);
FILLPOLY(4,CN);
SETCOLOR(14);
DRAWPOLY(4,CN);

OUTTEXTXY(25,330,'ENTER FILE NAME: ');
gotoxy(28,22);
READLN(S);
S:=S+CHR(0);
i:=0;att:=$20;
fr:=1;fh:=1;fs:=1;fc:=1;
OUTTEXTXY(80,440,'Enter .r. for read .h. for hide .s. for system ');
repeat
c:=readkey;
case c of
'R':if fr=1 then

```

```

        OUTTEXTXY(80,380,'(R,r)READ ONLY');end;
        'r':if fr=1 then
        begin att:=att+1;fr:=0;
        OUTTEXTXY(80,380,'(R,r)Read only');end;
        'H':if fh=1 then
        begin att:=att+2;fh:=0;
        OUTTEXTXY(270,380,'(H,h)Hidden ');end;
        'h':if fh=1 then
        begin att:=att+2;fh:=0;
        OUTTEXTXY(270,380,'(H,h)Hidden ');end;
        'S':if fs=1 then
        begin att:=att+4;fs:=0;
        OUTTEXTXY(450,380,'(S,s)System ');end;
        's':if fs=1 then
        begin att:=att+4;fs:=0;
        OUTTEXTXY(450,380,'(S,s)System ');end;
    end;
    i:=i+1;

until (i=4) or (ord(c)=13);
setat;
IF B= 1 THEN BEGIN
    IF ERR = 1 THEN
    BEGIN WE;OUTTEXTXY(30,350,'ERROR!!!');
    OUTTEXTXY(35,400,'Uknown functfon code');
    end
    ELSE IF ERR = 2 THEN
    BEGIN WE;OUTTEXTXY(30,350,'ERROR!!!');
    OUTTEXTXY(30,400,'file not found!!!');
    end
    else IF ERR = 3 THEN
    BEGIN WE;OUTTEXTXY(30,350,'ERROR!!!');
    OUTTEXTXY(30,400,'Path is not found!!!');
    end
    else IF ERR = 3 THEN
    BEGIN WE;OUTTEXTXY(30,350,'ERROR!!!');
    OUTTEXTXY(30,400,'attribute cannot be changed!!!');
    END;

    END;
SETFILLSTYLE(1,blue);
bar(30,435,380,465);
OUTTEXTXY(35,440,'press any key to ccntinue...');
c:=READkey;
WE;

END;{PROCEDURE}
{*****}
PROCEDURE GET_ATTRIB;

PROCEDURE GETAT;ASSEMBLER;
ASM
    PUSH AX
    PUSH DX
    MOV AH,$43
    MOV AL,0
    LEA DX,DS:S
    INC DX
    INT $21
    PUSH AX
    MOV AL,0
    ADC AL,0
    MOV BYTE PTR DS:B,AL;

```

```

MOV WORD PTR DS:ERR,AX
MOV WORD PTR DS:ATT,CX
POP DX
POP AX
END;
BEGIN
we1;

CN[1].X:=210;      CN[1].Y:=330;
CN[2].X:=610;      CN[2].Y:=330;
CN[3].X:=610;      CN[3].Y:=352;
CN[4].X:=210;      CN[4].Y:=352;
CN[5].X:=210;      CN[5].Y:=330;
{-----}
  SETFILLSTYLE(1,0);
  FILLPOLY(4,CN);
  SETCOLOR(14);
  DRAWPOLY(4,CN);

OUTTEXTXY(25,330,'ENTER FILE NAME: ');
gotoxy (28,22);
READLN (S);
S:=S+CHR (0);
GETAT;

IF B= 1 THEN BEGIN
  IF ERR = 1 THEN
  BEGIN WE;OUTTEXTXY(30,350,'ERROR!!!');
  OUTTEXTXY(35,400,'Unknown function code');
  end
  ELSE IF ERR = 2 THEN
  BEGIN WE;OUTTEXTXY(30,350,'ERROR!!!');
  OUTTEXTXY(30,400,'file not found!!!');
  end
  else
  IF ERR = 3 THEN
  BEGIN WE;OUTTEXTXY(30,350,'ERROR!!!');
  OUTTEXTXY(30,400,'Path is not found!!!');end
  else BEGIN
    WE;OUTTEXTXY(30,350,'ERROR!!!');END;
  END
  ELSE
  if att= $20 then
  OUTTEXTXY(420,435,'Normal file') else
  begin
    aa:=att;
    aa:= aa and 1;
    if aa = 1 then
    OUTTEXTXY(105,365,'Read only');
    aa:=att;
    aa:=aa and 2;
    if aa=2 then
    OUTTEXTXY(265,365,'Hidden  ');
    aa:=att;
    aa:=aa and 4;
    if aa=4 then
    OUTTEXTXY(425,365,'System  ');
    aa:=att;
    aa:=aa and 8;
    if aa=8 then
    OUTTEXTXY(90,405,'Volume name ');
    aa:=att;
  end

```

```

aa:=aa and 16;
if aa=16 then
OUTTEXTXY(250,405,'Subdirectory ');
aa:=att;
aa:=aa and 32;
if aa=32 then
OUTTEXTXY(425,405,'Changed ');

aa:=att;
end;
SETFILLSTYLE(1,blue);
bar(30,435,380,465);
OUTTEXTXY(35,440,'press any key to continue...');
c:=READKEY;
WE;

END; {PROCEDURE}
{*****}
PROCEDURE ABOUT;
BEGIN
WW;
yf1:=60;
OUTTEXTXY(30,YF1,'      This is a very easy program to use < DOS > ');
OUTTEXTXY(28,YF1+30,'commands .The user may use both the keyboard arrow');
OUTTEXTXY(28,YF1+60,'keys and the hot keys indicated for each command. ');
OUTTEXTXY(28,YF1+90,'      This program is done by Suleiman Khalaf and ');
OUTTEXTXY(28,YF1+120,'presented to Dr. Ziad Al-Qadi .');
settextstyle(7,0,4);

setcolor(black);
outtextxy(45,250,'HOPE YOU ENJOY USING IT ! ');

c:=readkey ;
SECOND MENU; CC;
END; {PROCEDURE}
{*****}
PROCEDURE CHECK(M1:INTEGER);
BEGIN
CASE M1 OF
10:DIRICTORY;
70:FILE_TOOLS;
130:FIND_FILE;
190:TIME;
250:DATE;
310:ATTRIB;
370:ABOUT;
430:C:='Q'
END;
END; {PROCEDURE}
PROCEDURE CHECKhot(c1:char);
BEGIN
CASE c1 OF
'd','D':DIRICTORY;
'l','L':FILE_TOOLS;
'f','F':FIND_FILE;
't','T':TIME;
'd','d':DATE;
'a','A':ATTRIB;
'b','B':ABOUT;
'Q','q':C:='Q'
END;
END; {PROCEDURE}
PROCEDURE CHECK1(M11:INTEGER);

```



```

CASE M11 OF
  60:MAKE_DIR;
  110:CHANGE_DIR;
  160:REMOVE_DIR;
  210:C:='Q';
END;
END; {PROCEDURE}
PROCEDURE CHECK1HOT (C11:CHAR);
BEGIN
  CASE C11 OF
    'M','m':MAKE_DIR;
    'C','c':CHANGE_DIR;
    'R','r':REMOVE_DIR;
    'E','e':C:='Q';
  END;
END; {PROCEDURE}

PROCEDURE CHECK2 (M11:INTEGER);
BEGIN
  CASE M11 OF
    40:RENAME_FILE;
    90:DELETE_FILE;
    140:COPY_FILE;
    190:dump_FILE;
    240:C:='Q';
  END;
END; {PROCEDURE}
PROCEDURE CHECK2HOT (C11:CHAR);
BEGIN
  CASE C11 OF
    'R','r':RENAME_FILE;
    'D','d':DELETE_FILE;
    'C','c':COPY_FILE;
    'M','m':dump_FILE;
    'E','e':C:='Q';
  END;
END; {PROCEDURE}

PROCEDURE CHECK3 (M11:INTEGER);
BEGIN
  CASE M11 OF
    50:SET_TIME;
    110:GET_TIME;
    160:C:='Q';
  END;
END; {PROCEDURE}
PROCEDURE CHECK3HOT (C11:CHAR);
BEGIN
  CASE C11 OF
    'S','s':SET_TIME;
    'G','g':GET_TIME;
    'E','e':C:='Q';
  END;
END; {PROCEDURE}

PROCEDURE CHECK4 (M11:INTEGER);
BEGIN
  CASE M11 OF
    60:SET_DATE;
    110:GET_DATE;
    160:C:='Q';
  END;

```

```

PROCEDURE CHECK4HOT(C11:CHAR);
BEGIN
    CASE C11 OF
        'S','s':SET_DATE;
        'G','g':GET_DATE;
        'E','e':C:='Q';
    END;
END; {PROCEDURE}

PROCEDURE CHECK5(M11:INTEGER);
BEGIN
    CASE M11 OF
        60:SET_ATTRIB;
        110:GET_ATTRIB;
        160:C:='Q';
    END;
END; {PROCEDURE}

PROCEDURE CHECK5HOT(C11:CHAR);
BEGIN
    CASE C11 OF
        'S','s':SET_ATTRIB;
        'G','g':GET_ATTRIB;
        'E','e':C:='Q';
    END;
END; {PROCEDURE}

BEGIN.    { MAIN PROGRAM }
    GD := DETECT;
    INITGRAPH(GD, GM, 'a:\');
    IF GRAPHRESULT <> GROK THEN
        HALT(1);
        FIRST_MENU;

        SECOND_MENU;
        CC;
        CLOSEGRAPH;
END.    {MAIN}

```

```

USES
  Crt, Mouse, Graph, Dos;
type
  fi_le = record
    name:string;
    size:longint;
    attr:byte;
  end;
VAR
  xx,yy,i_dir,i_file,i:INTEGER;
  s_size,ff,filename,drive,path,sdir,s,spath,s1:STRING;
  f_found,drive_no,attrib:byte;
  low_size,high_size:word;
  directoreis:array [1..50] of string;
  files:array [1..50] of fi_le;
  dta:array [1..60] of byte;
PROCEDURE Set_Graphics_Mode;
VAR
  gm,gd:integer;
  xmg,ymg:integer;
BEGIN
  gd:=detect;
  initgraph (gd,gm,'');
  if graphresult<>grok then halt (t);
end;

PROCEDURE Main_Menu;
VAR
  XMG,YMG:integer;
BEGIN
  set_graphics mode;
  SetColor (yellow);
  SetLineStyle (0,$3,1);
  Rectangle (2,2,635,470);
  Rectangle (5,5,632,467);
  SetLineStyle (0,$c3,3);
  setcolor (green);
  Rectangle (25,25,612,447);
  SetLineStyle (0,$c3,1);
  Line (25,55,612,55);
  Line (25,59,612,59);
  SetColor (3);
  SetTextStyle (7,0,3);
  OutTextXY (265,24,'MJ/TOOLS');
  SetTextStyle (1,0,1);
  SetColor (9);
  OutTextXY (35,28,'12/2/1996 MON');
  OutTextXY (500,28,'12:30 pm');
  SetColor (green);
  SetLineStyle (0,$c3,3);
  Line (25,89,612,89);
  SetFillStyle (11,11);
  bar (27,61,610,87);
  SetColor (15);
  SetTextStyle (0,0,2);
  xmg:=44;ymg:=67;
  OutTextXY (xmg,ymg,'FILE');
  OutTextXY (xmg+90,ymg,'DIRECTORY');
  OutTextXY (xmg+260,ymg,'SYSTEM');
  OutTextXY (xmg+385,ymg,'ABOUT');
  OutTextXY (xmg+485,ymg,'EXIT');
  SetColor (green);
  SetLineStyle (0,$c3,1);

```

```

    setcolor(green);
    Line (25,122,612,122);
    SetLineStyle (0,$c3,3);
    Line (145,122,145,447);
    SetColor (15);
    SetTextStyle (2,0,6);
    OutTextXY (44,123,'Directory');
    SetLineStyle (0,$c3,1);
    SetColor (green);
    Line (25,145,450,145);
    SetColor (15);
    OutTextXY (184,123,'File');
    OutTextXY (284,123,'Size');
    OutTextXY (380,123,'Attrib');
    SetColor (green);
    SetLineStyle (0,$c3,3);
    Line (450,123,450,447);
END;
procedure get_dir;
procedure getdir; assembler;
asm
    push ax
    push bx
    push cx
    push si
    mov ah,$47
    mov dl,0
    lea si,ds:spath
    int $21
    pop si
    pop cx
    pop bx
    pop ax
end;
begin
    sdir:='';
    getdir;
    I:=0;
    WHILE ORD (spath[I])<>0 DO BEGIN
        sdir:=sdir+ spath[I];
        I:=I+1;
    END;
end;
procedure get_drive;
procedure gd; assembler;
asm
    push ax
    mov ah,$19
    int $21
    mov byte ptr ds:drive_no,al
    pop ax
end;
begin
    gd;
    drive:=chr (65+drive_no);
end;

procedure get_path;
begin
    path:='';
    get_drive;
    get_dir;
    path:=drive+':\'+sdir;

```



```

PROCEDURE FIND;
VAR
c:byte;
procedure set_dta;assembler;
asm
    push ax
    push bx
    push cx
    push dx
    mov ah,$1a
    mov dx,offset ds:dta;
    int $21
    pop dx
    pop cx
    pop bx
    pop ax
end;

procedure find_first;assembler;
asm
    push ax
    push bx
    push cx
    push dx
    mov cx,$ff
    mov ah,$4e
    mov dx,offset ds:filename;
    inc dx
    int $21
    push ax
    mov ax,0
    adc al,0
    mov byte ptr ds:f_ound,al
    pop ax
    pop dx
    pop cx
    pop bx
    pop ax
end;
procedure find_next;assembler;
asm
    push ax
    push bx
    push cx
    push dx
    mov ah,$4f
    int $21
    push ax
    mov ax,0
    adc al,0
    mov byte ptr ds:f_ound,al
    pop ax
    pop dx
    pop cx
    pop bx
    pop ax
end;

begin
filename:='*. *';
set_dta;
find_first;

```

```

i_file:=1;
{get directoreis}
while (f_ound =0) and (i_dir<=18) do begin
    attrib:=dta [22];ff:='';
    c:=32;
    i:=31;
    while c<> 0 do begin
        c:=dta [i];
        ff:=ff+chr(dta [i]);
        i:=i+1;
    end;

    if attrib=16 then begin
        directoreis [i_dir]:=ff;
        i_dir:=i_dir+1;
    end;
    find_next;
end;

filename:='*. *';
set_dta;
find first;
i_file:=1;

{get files}

while (f_ound =0) and (i_file<=18) do begin
    attrib:=dta [22];ff:='';
    c:=32;
    i:=31;
    while c<> 0 do begin
        c:=dta [i];
        ff:=ff+chr(dta [i]);
        i:=i+1;
    end;

    if attrib<>16 then
    with files[i_file] do begin
        name:=ff;
        attr:=attrib;
        size:=dta[27]+dta[28]*256+dta[29]*65536+
        dta[30]*1677216;
        if size<0 then size:=65536+size;
        i_file:=i_file+1;
    end;

    find_next
end;
end;
BEGIN          { MAIN PROGRAM }
    Main Menu;
    Reset_Mouse;
    get_path;
    SetColor (15);
    SetTextStyle (2,0,5);
    OutTextXY (44,95,path);
    find;
    yy:=140;xx:=44;
    for i:=1 to i_dir-1 do begin
        outtextxy (xx,yy,directoreis[i]);
        yy:=yy+16;
    end;
    yy:=150;xx:=154;

```

```

        outtextxy(xx,yy,files[i].name);
        str (files[i].size,s_size);
        outtextxy (xx+130,yy,s_size);
        yy:=yy+16;
        end;

repeat
    display_c;
    getxy_button;
    if (mbu=1) then
        BEGIN
            if (ymp>=62) AND (ymp<=84) AND (xmp>=34) AND (xmp<=113) then
                BEGIN
                    SetColor (15);
                    OutTextXY (200,200,'FILE');
                    Delay (1000);
                    SetColor (0);
                    OutTextXY (200,200,'FILE');
                    END;
                IF (ymp>=62) AND (ymp<=84) AND (xmp>=123) AND (xmp<=284) THEN
                    BEGIN
                        SetColor (15);
                        OutTextXY (200,200,'DIR');
                        Delay (1000);
                        SetColor (0);
                        OutTextXY (200,200,'DIR');
                        END;
                    IF (ymp>=62) AND (ymp<=84) AND (xmp>=294) AND (xmp<=410) THEN
                        BEGIN
                            SetColor (15);
                            OutTextXY (200,200,'SYSTEM');
                            Delay (1000);
                            SetColor (0);
                            OutTextXY (200,200,'SYSTEM');
                            END;
                        IF (ymp>=62) AND (ymp<=84) AND (xmp>=420) AND (xmp<=512) THEN
                            BEGIN
                                SetColor (15);
                                OutTextXY (200,200,'ABOUT');
                                Delay (1000);
                                SetColor (0);
                                OutTextXY (200,200,'ABOUT');
                                END;
                            IF (ymp>=62) AND (ymp<=84) AND (xmp>=516) AND (xmp<=595) THEN
                                BEGIN
                                    Closegraph;
                                    Sound(450);
                                    Delay(450);
                                    Nosound;
                                    Exit;
                                END;
                            END;
                            OutTextXY (10,10,s);
                        UNTIL (mbu=3) OR (keypressed);
                    CloseGraph;
                END.

```

المراجع

- 1- BERZISS, A. T (1975) "DATA STRUCTURES THEORY AND PRACTICE" , 2nd Edn, Academic Press, LONDON .
- 2- CHARLES H. Roth, Jr (1985) , "FUNDAMENTALS OF LOGICAL DESIGN" 3rd Edn, West Publishing Company, NEW YORK .
- 3- CHARLES S. PARKER (1984) "UNDERSTANDING COMPUTERS AND DATA PROCESSING TODAY AND TOMORROW WITH BASIC" 1st Edn, Hol, Rinehart Winston, NEW YORK .
- 4- MADNICK/ DONOVAN (1978) "OPERATING SYSTEMS" Int - ST. Edn MCGRAW HILL INTERNATIONAL Book Company .
- 5- Page, E. Sand Wilson, L. B. (1973) "INFORMATION REPRESENTATION AND MANIPULARION IN A COMPUTER" Cambridge U.P. , Cambridge .
- 6- Peter Bishop (1983) "CONPREHENSIVE COMPUTER STUDIES" Edward Arnold, LONDON .
- 7- DOS - Manuals .
- 8- UNIX - Manuals .

9 - د. زياد القاضي ، مسعود نصرو ، نظم التشغيل ، دار المستقبل ، عمان ،
الأردن ، 1988 .

أنظمة التشغيل
OPERATING SYSTEMS



دار
المسيرة

للنشر والتوزيع والطباعة

www.massira.jo

ISBN 9957-06-050-3



9 789957 060503

Yaman