

أدارة الذاكرة الأساسية

يعتبر تنظيم وقيادة الذاكرة الأساسية أو بمعنى آخر ادارة الذاكرة واحداً من أهم الأعمال التي توليها نظم التشغيل إهتماماً كبيراً باعتبار الذاكرة الأساسية المكان الوحيد الذي منه تستقى وحدة التشغيل المركزية إيعازات البرامج (*) والبيانات المرتبطة بها ، ودون الذاكرة الأساسية لا منفذ إلى وحدة التشغيل المركزية مهما ضم النظام من وسائط التخزين الثانوية رغم ما تتصف به هذه الوسائط من إمكانية تخزين أحجام هائلة من البيانات أو البرامج ، إضافة إلى رخص سعرها النسبي مقارنة بتكلفة تخزين بايت واحدة على الذاكرة الأساسية ، ورغم أن التكنولوجيا الحديثة عدلت وطورت وأفرزت ذاكرات أساسية تتصف بـكبير حيز تخزينها فلا تزال تكلفة تخزين بايت على الذاكرة الأساسية أعلى من نظيرها على الذاكرات الثانوية، وان عكس ما أسلفنا شيئاً فانه يعكس تضخم حجم البرامج الحديثة ومدى حاجة المبرمجين إلى حيز أكبر من عناصر التخزين المعنونة.

والواقع أن تنظيم الذاكرة الأساسية يعنى فى المحل الأول كيف يوزع هذا الحيز؟ هل يخصص لبرنامج واحد؟ أو يقسم بين عدة برامج تتناولها وحدة التشغيل المركزية ، وتعطى الإيحاء لكل صاحب برنامج أنه ينفرد بالنظام نتيجة سرعتها على الفصل والوصل SWITCHING بين مختلف البرامج بسرعة فائقة؟

(*) تستخدم كلمات برنامج / مهمة/ عمل تبادلياً للدلالة على نفس المفهوم.

وإذا اخترنا تقسيمها بين عدة برامج فهل تقسم الذاكرة الأساسية بين مختلف البرامج بالتساوى أم تقسم حسب حاجة كل برنامج ، وإذا قسمناها حسب حاجة كل برنامج فهل يظل التقسيم ثابتاً لا يتغير أو يتبدل أم من الممكن تغييره ديناميكياً بحيث يلائم الحيز متطلبات واحتياجات البرنامج؟ ولو تم التقسيم هل يفترض حتمية اتصال الحيز CONTIGOUS أم من الممكن إجراء التقسيم على هيئة صفحات PAGES لا تكون متلامسة فيزيائياً.

وقيادة الذاكرة تعنى فى المحل الأول [وبصرف النظر عن تنظيم الذاكرة] إستراتيجية التعامل معها بحيث تحقق أعلى درجات كفاءة الأداء.. مثلاً متى ندفع ببرنامج لم يسبق تنفيذة إلى النظام؟ هل ندفع به متى طلب النظام ذلك أم نرج به إلى النظام متى شئنا؟ وأين ندفع بهذا البرنامج فى حيز خال أو ننتظر حتى تتجمع الثقوب HOLES (والتي تسمى أحياناً فتات التوزيع Fragmentation) فى حيز أكبر صالح للاستخدام بعد عملية دمج الثقوب COMPACTION .

أسئلة كثيرة حول قيادة الذاكرة مثل هل عند طلب تنفيذ برنامج فى قائمة الإنتظار فما هو البرنامج الذى يجب تحييته من دورة التشغيل.. أهو البرنامج الأقدم من حيث التواجد فى دورة التشغيل.. أم هو البرنامج الأحدث تواجداً.. أم هو البرنامج الذى لا يمثل إيقافه عبئاً على مستخدمى النظام أو خلافاً فى أداء منظومة العمل.. أم هو البرنامج الذى يستهلك وقتاً أقصر.. أم هو البرنامج الذى يستأثر بجل وقت وحدة التشغيل المركزية.

أسئلة كثيرة ، فرضت مشاكل عديدة حيال تصميم برامج تنظيم وقيادة الذاكرة وأثر كذلك على تصميمها.. ولنا فى تاريخ أجيال الحاسبات دروس كثيرة.. فى الجيل الثانى وبدايات الجيل الثالث كانت الذاكرة تصنع من حلقات مغناطيسية وكان ثمنها مرتفعاً لذلك حدث من التفكير فى نظم تشغيل متطورة كما حددت أعداد مستخدمى منظومة الحاسب.. ثم تطورت عبر رحلة تكنولوجية طويلة وصولاً إلى

الذاكرة المخفية CACHE الأسرع آلاف المرات من ذاكرات الدوائر الإلكترونية المتكاملة متناهية الصغر مما دعا في كل مرحلة إلى تطوير نظم التشغيل تطويراً كبيراً ليوائم التطور في الكيان الآلى والكيان البرمجى.

ونخلص إلى أن ادارة الذاكرة تستهدف أربع وظائف أساسية هي:

أ - مراقبة حالة STATUS جميع المواقع المعنونة سيات كانت موزعة أو غير موزعة.

ب - تحديد سياسة توزيع الحيز على البرامج والبيانات مع تحديد الأسبقيات الملائمة للتنفيذ.

ج - تحديد أسلوب التوزيع هل هو متعدد المهام ثابت الحيز Multifixed Tasks (MFT) أو على هيئة دفعة واحدة BATCH أو بنظام الصفحات PAGING أو نظام التبديل SWAPING.

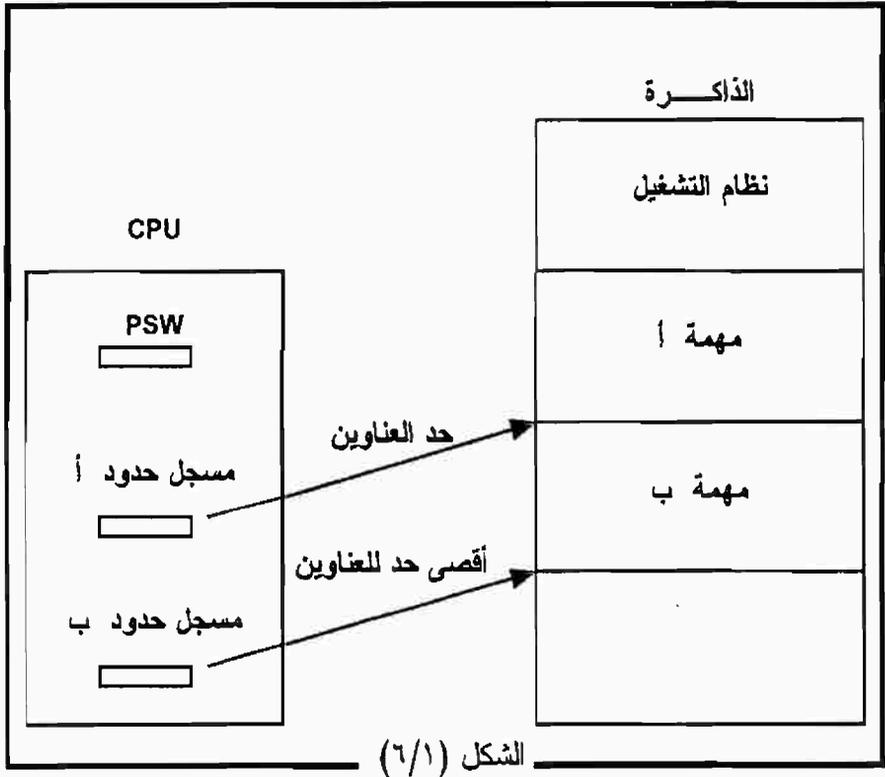
د - تحديد أسلوب التعامل مع الحيز الفائض أو الثغوب الناجمة عن إخلاء بعض المناطق الموزعة.

ويمكن دراسة ادارة الذاكرة من خلال منظورين ، الأول: التوزيع المتصل ، الثانى: التوزيع غير المتصل ، وفى كل منظور تتناول عددا من الاساليب المتعلقة بهذا المنظور وذاك المنظور الآخر ، لكن سيات كان التوزيع متصلا أو دون ذلك فان وقاية الذاكرة واعادة تسكين المهام داخل الذاكرة من ابرز الواجبات التى يوليها مدير الذاكرة عنايته واهتمامه ونجد لزاما قبل تناول اساليب التوزيع أن نتطرق بدراسة موجزة إلى امرين يتعلقان بوقاية الذاكرة واعادة تسكين المهام:

١ - وقاية الذاكرة:

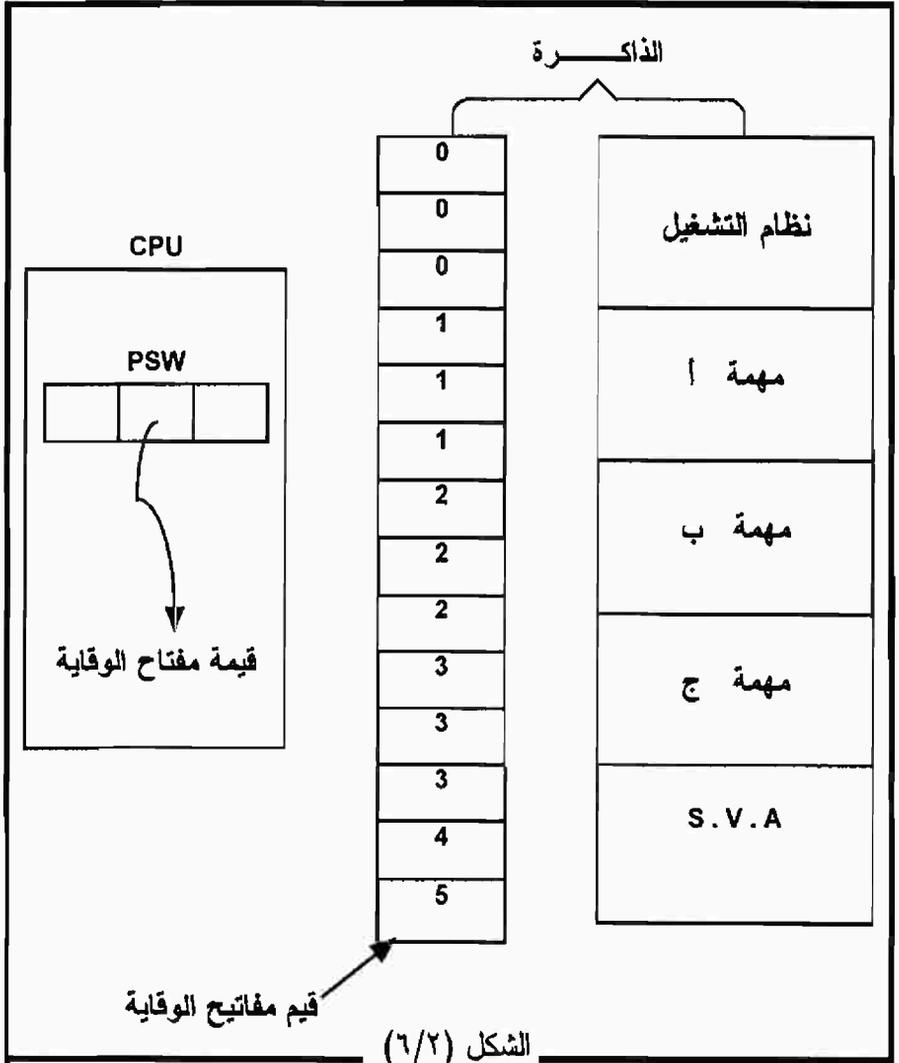
يقصد بوقاية الذاكرة سيات شغلنها مهمة واحدة أو عدة مهام أن تتحقق وقاية لكل مهمة من التداخل مع مهمة أخرى أو مع برامج نظام التشغيل ، لذا

يستخدم لكل مهمة زوج من مسجلات الحدود أو استخدام مفاتيح وقاية لكل مهمة Protection Key ويوضح الشكل (٦/١) استخدام مسجلات الحدود بحيث تتم مراجعة قيم هذه المسجلات بواسطة المشغل CPU كلما نشطت المهمة إن تطابقت القيم سمح للمهمة بإستكمال المعالجة ، أما إذا تكدت القيم وحاولت



مهمة اخرى اختراق الحدود تتم مقاطعة على المشغل ، كما يوضح الشكل (٦/٢) استخدام مفاتيح الوقاية التي تتطلب تقسيم الذاكرة إلى قطع متساوية الحيز ولكل قطعة مفتاح يخزن في حيز (بايت) داخل الذاكرة مثلما تسجل قيمته في حيز PSW وتتم

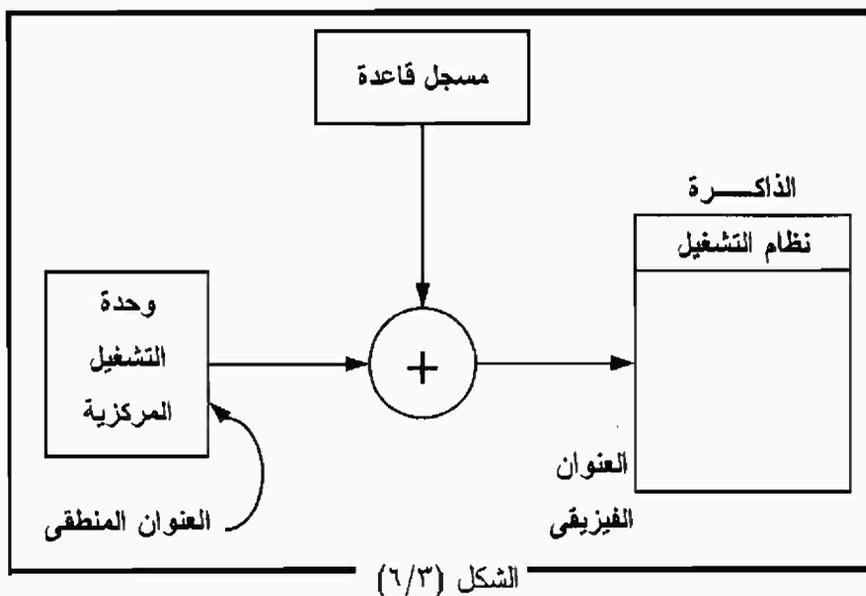
مراجعتَه بواسطة وحدة التشغيل المركزية ومطابقة قيمته في PSW والذاكرة فإن تطابقا كان بها وسمح بالولوج إلى حيز المهمة.
 وفي كلا الأسلوبين يتم حذف أو تعديل قيم المفاتيح عند إنتهاء تنفيذ المهمة وإستعادة مدير الذاكرة للمحلات الخالية.



الشكل (٦/٢)

٢ - إعادة تسكين المهام:

سبق وعرضنا إلى مشكلة تسكين البرامج في الذاكرة خاصة البرامج التسي لاتقبل التنفيذ إلا في مواقع محددة أى لاتقبل التنفيذ إلا في العناوين المطلقة مما يستدعى إعادة تسكينها ، والعملية في مفهومها العام بسيطة وإن كان تنفيذها فعلياً يتطلب كيانين ألياً ويرمجياً يضيفان قيمة مسجل قاعدة إلى العناوين لتتحول إلى عناوين فعلية أو فيزيائية فيما يوضحه الشكل (٦/٣).



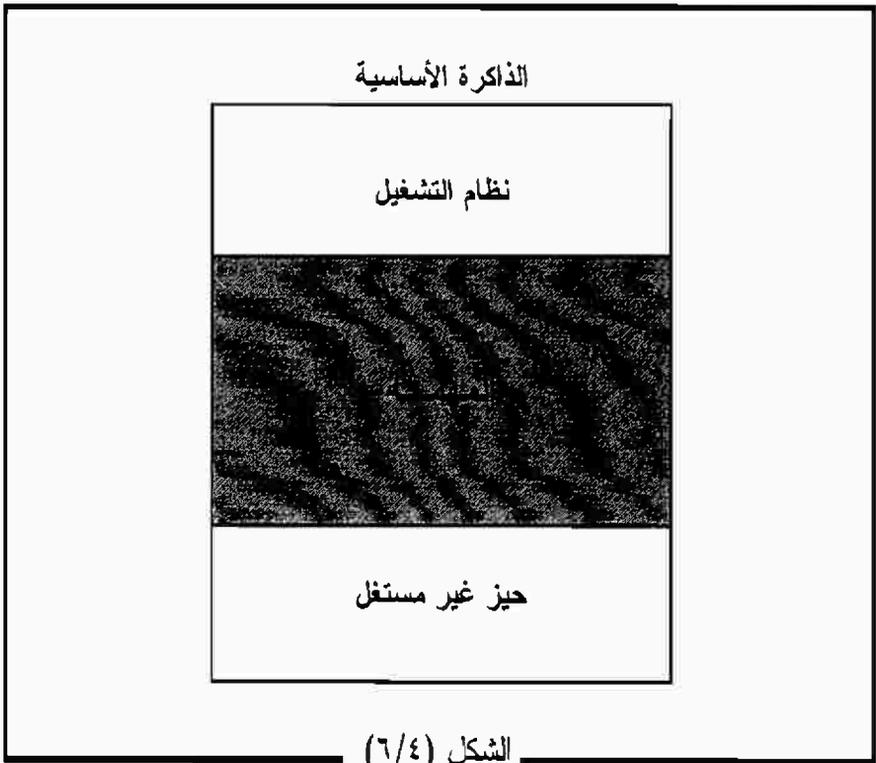
أدارة الذاكرة:

أولاً : ادارة الذاكرة وفق اسلوب الحيز المتصل لمهمة واحدة:
نظام المستخدم الواحد :

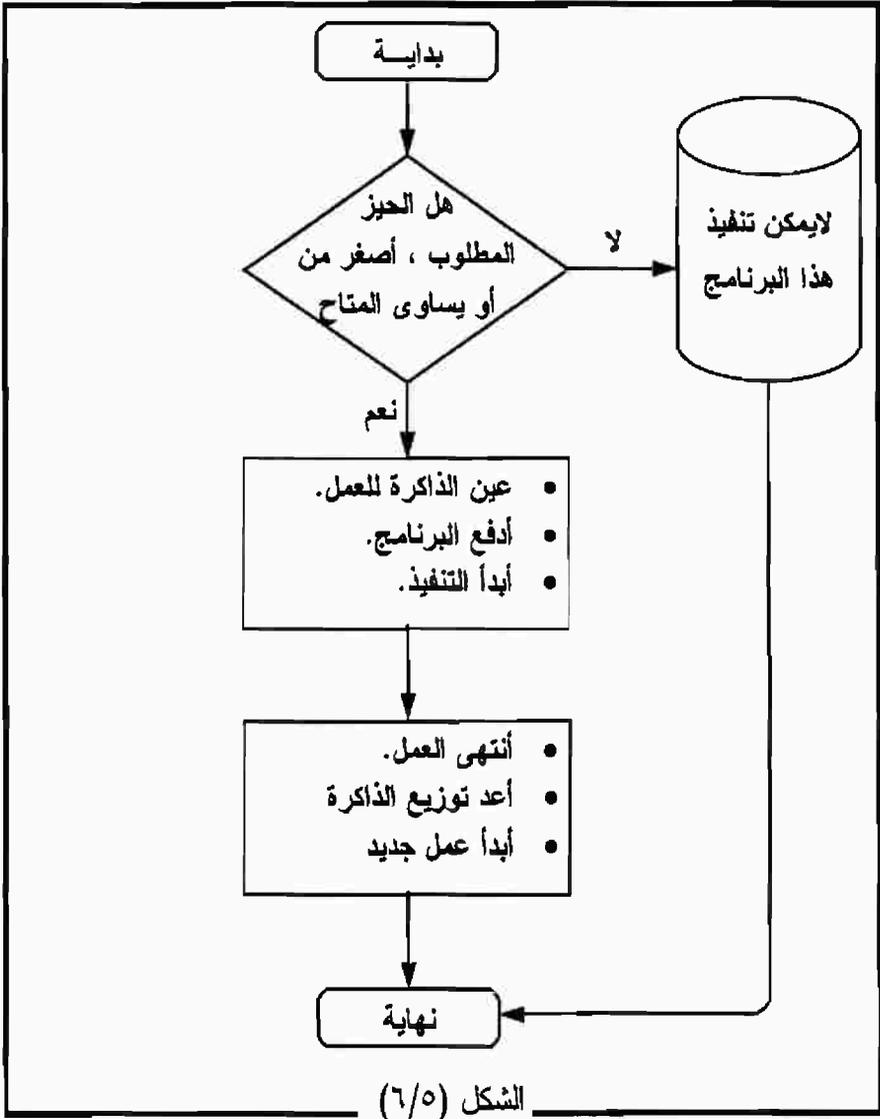
نظام الدفعة BATCH :

يعتبر هذا النظام أقدم النظم فى إدارة الذاكرة ، إذ طبق مع بدايات استخدام الحاسبات ، ومازال مستخدماً فى بعض المواقع والتطبيقات ، ويفترض هذا الأسلوب إتاحة جميع موارد النظام لمستخدم واحد بشرط ألا يزيد حيز المهمة عن الحيز المتاح فى الذاكرة.

ومن وجهة النظر المنطقية فإنه يتم تقسيم الذاكرة إلى ثلاثة أجزاء ، قسم يخصص لبرامج نظام التشغيل ، والقسم التالى للمهمة أو برنامج التطبيق ، وعادة يفيض الحيز المتاح عن الحيز المشغول ويعتبر هذا الحيز هو القسم الثالث فيما يوضحه الشكل (٦/٤) ، وهو حيز غير مرتبط بأى برنامج أو مهمة JOB.



والواقع أن نظام "الدفع" بسيط في منطقة ويحقق الوظائف الأربع الأساسية لإدارة الذاكرة ، كذلك لا يتطلب أى دعم إضافي على الهيكل الآلى أو الهيكل البرمجى ، ويوضح الشكل (٦/٥) منطق أسلوب الحيز المتصل:



مميزات الحيز المتصل لمستخدم واحد:

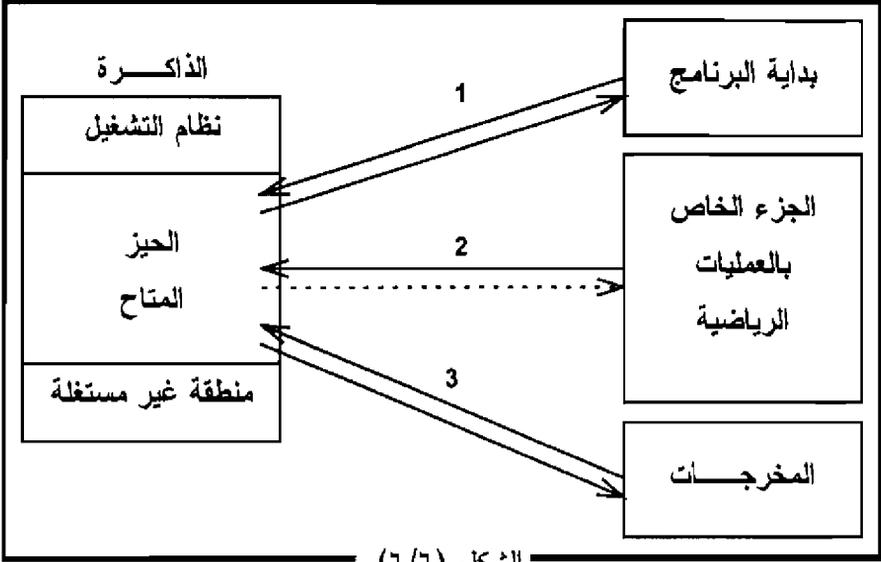
- أ - بسيط والتعامل معه بسيط.
- ب - رخيص.
- ج - لا يحتاج دعماً فنياً أو آلياً خاصاً.
- د - أسلوب وقاية البرنامج من نظام التشغيل بسيط.
- هـ - ترصص المهام على هيئة قطار إنتظار Stream of Jobs ويطبق عليها سياسة من يأتي أولاً يخدم (يشغل) أولاً.

عيوب أسلوب الحيز المتصل:

- أ - عدم أستغلال حيز الذاكرة الأستغلال الأمثل وكذلك عدم استغلال قدرة وحدة التشغيل المركزية الأستغلال الأوفق حيث تتوقف عن أداء دورها وتتفرغ لمراقبة معدات I/O البطيئة.
- ب - طول مدة إنتظار المهام الأخرى حيث عليها البقاء راكدة حتى يأتي عليها الدور.
- ج - أسلوب غير مرن حيث يعجز عن معالجة مهمة يزيد الحيز المطلوب لها عن الحيز المتاح.

وهذا العيب أمكن التغلب عليه بأسلوب الإحلال OVERLAY ، بأن يقسم البرنامج إلى أجزاء مستقلة حيزها أصغر قليلاً من الحيز المتاح وكل منها يؤدي وظيفة منطقية متكاملة مثل CALCULATION- OUTPUT - INITIALIZATION بحيث يدفع كل جزء تلو الآخر إلى الذاكرة الأساسية فيما يوضحه الشكل (٦/٦).

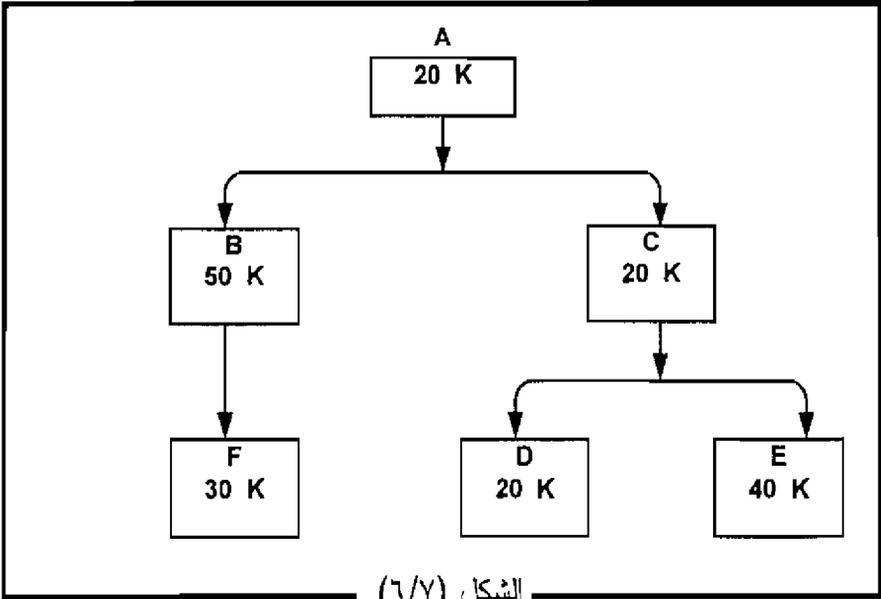
وما يعيب هذا الأسلوب أنه يتطلب مهارة عالية في صياغة منطق البرنامج ، كما يتطلب نظام تشغيل ملائم.



الشكل (٦/٦)

مثال:

برنامج على النحو الموضح بالشكل (٦/٧) ، كيف يتم تنفيذه ؟



الشكل (٦/٧)

الحل:

الكتلة (A) تستدعى (B) أو (C) فقط ، بينما (B) تستدعى فقط (F) فى حين تستدعى (C) الكتلة (D) أو (E) ، وكنتيجة لهذا المخطط أن (B) لا تستدعى (C) والعكس صحيح ، وبالتالي فإنه يستحيل تواجد (B) أو (C) سوياً فى الذاكرة الأساسية لذا يستخدم برنامج مشرف إجلاالى OVERLAY SUPERVISOR على أن ينص المبرمج صراحة على منطق الإتصالات بين كتل البرنامج ، للفصل بين هذه الخطوات.

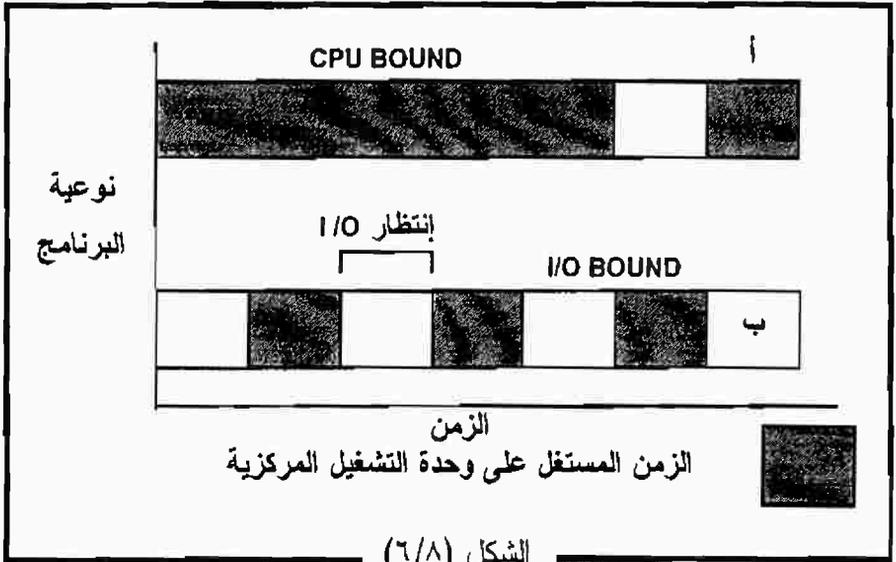
تأثير طبيعة البرامج على أسلوب الحيز المتصل:

تعتبر البرامج ذات الطبيعة العلمية أصلح أنواع البرامج لنظام الدفعة حيث تستغل موارد وحدة التشغيل المركزية الأستغلال الأمثل.

ويوضح الشكل (٦/٨) مقارنة بين برنامجين احدهما:

(أ) برنامج تطبيقات علمية والآخر.

(ب) برنامج يرتكن على المدخلات والمخرجات.



الشكل (٦/٨)

أسلوب الحيز المتصل لعدد من

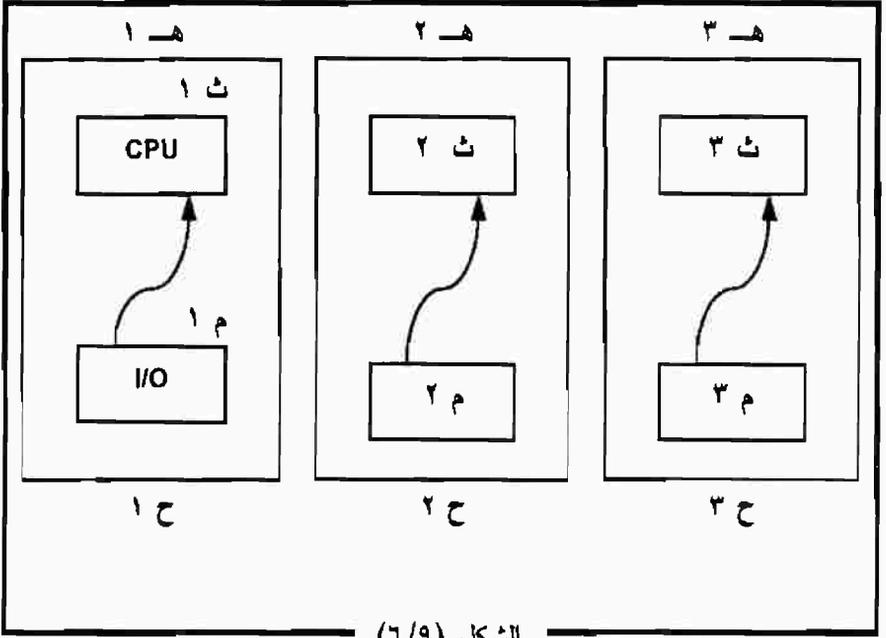
المهام MULTI PROGRAMING :

يعتبر هذا الأسلوب علاجاً للقصور الواضح في أسلوب المهمة الواحدة فسي استغلال الموارد خاصة في البرامج التي تعتمد على المدخلات والمخرجات ، حيث يهدر البرنامج وقت وحدة التشغيل المركزية إهداراً فادحاً مما دعا إلى التفكير في دفع عدة برامج في نفس حيز الذاكرة الأساسية ، وتتولى وحدة التشغيل المركزية الانتقال بينها من خلال سرعة الفصل والاتصال SWITCHING وتوضح الفقرة التالية مدى الارتقاء بكفاءة النظام من حساب زمن الإنتظار الذي يواجهه كل برنامج.

حساب زمن الانتظار:

تفرض أن هناك ثلاثة مهام سوف تنفذ على الحاسب هي: [هـ١] ، [هـ٢] ، [هـ٣] على التوالي ، وأن [هـ١] يستهلك [ث١] من إجمالي الزمن المتاح على وحدة التشغيل المركزية ، وأن [هـ٢] يستهلك [ث٢] ، وأن [هـ٣] يستهلك [ث٣] وأن البرامج الثلاثة سوف تشغل المساحات التالية من الذاكرة الأساسية على النحو [ح١] ، [ح٢] ، [ح٣] وأن تنفيذ I/O للبرامج الثلاثة يستهلك [م١] ، [م٢] ، [م٣] على التوالي ونقارن بين تطبيق أسلوب نظام الدفعة وبين أسلوب البرمجة المتعددة فيما يوضحه الشكل (٦/٩) ويلخصه الجدول التالي:

المهام	الحيز	CPU	I/O
هـ ١	ح ١	ث ١	م ١
هـ ٢	ح ٢	ث ٢	م ٢
هـ ٣	ح ٣	ث ٣	م ٣



الشكل (٦/٩)

بالتالى يكون الحيز المتاح : ح ١ + ح ٢ + ح ٣.

فى حالة نظام المهمة الواحدة:

- بفرض أن تم تنفيذ أحد المهام ولتكن [١ هـ] فقط.
- بالتالى فإن الحيز غير المستغل فى الذاكرة الأساسية هو ح ٢ + ح ٣.
- أيضاً فإن نسبة إنتظار وحدة التشغيل المركزية حتى تنتهى عمليات المدخلات والمخرجات :

$$\frac{1م}{1م + 1ث} =$$

* إذا النسبة المئوية للإنتظار [عدم أداء عمل] لوحدة التشغيل المركزية :

$$100 \times \frac{م}{م + ث} =$$

في حالة استخدام أسلوب تعددية البرمجة:

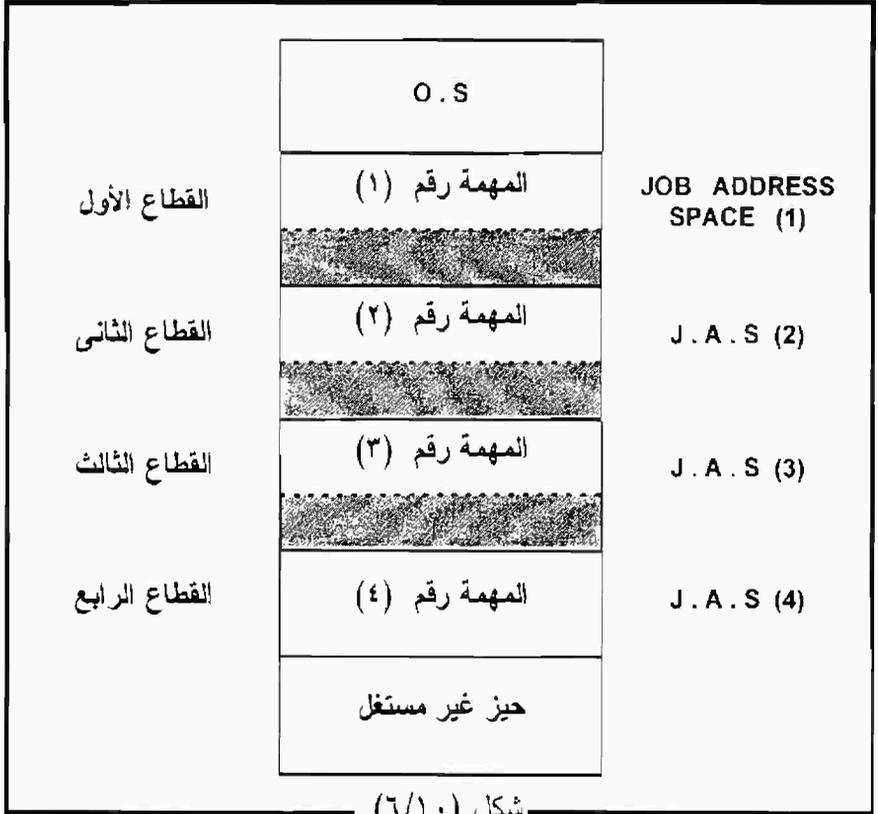
بفرض أنه تم تحميل الذاكرة الأساسية بالمهام الثلاث دفعة واحدة مع الفصل الفيزيقي والمنطقي بينها وبين نظام التشغيل ، وكذلك الفصل بين المهام الثلاث.

- إذا فإن نسبة إشغال الذاكرة الأساسية = 100%.
- حيث أن النظام يعطى أمراً بتنفيذ [هـ1] فإنه فور إنتهاء [ث1] لن تنتظر وحدة التشغيل المركزية إنتهاء [م1] بل توجه إلى المهمة [هـ2] حيث تنفذ عملية حسابية تستغرق [ث2] وحدة زمن وعند الفراغ منها سوف توجه وحدة التشغيل المركزية إلى [هـ3] حيث تنفذ عملية حسابية تستغرق [ث3] وعندما تنفذ [م3] سوف توجه نحو [ث1].
- بفرض أن [م1] أصغر زمنا من [م2] + [م3]،
- بالتالي فإن زمن الإنتظار WAIT في الحالتين: نجد أن (ث2 + ث3) - م1 أقل من زمن انتظار الدفعة الواحدة ، وترتيباً على ذلك فإن أسلوب تعددية البرمجة يحقق للنظام كفاءة أعلى من أسلوب الدفعة الواحدة.
- في الحالتين السابقتين تم تطبيق القانون التالي:

$$TOTAL WAIT = I/O WAIT TIME + CPU TIME$$

تقسيم الذاكرة في أسلوب الحيز المتصل:

لا يتطلب أسلوب تعددية البرمجة نظام تشغيل على درجة كبيرة من الكفاءة لكن يشترط تقسيم الذاكرة إلى جملة قطاعات Partitions بحيث يشحن كل قطاع ببرامج مهمة واحدة كما في الشكل (٦/١٠).



وهذا التقسيم يفضى إلى نوعين من أساليب تقسيم الذاكرة هما:

أولاً : التقسيم الثابت أو التقسيم الإستاتيكي:

وهو تقسيم حيز الذاكرة إلى قطاعات ثابتة محددة على أن يخصص لكل

برنامج القطاع المناسب له ، ونلمح هذا التقسيم في نظام التشغيل :

[OS / MFT] Operating System Multi Fixed Tasks

وأبرز عيوب هذا التقسيم عدم كفاءة أستغلال النظام "وحدة الذاكرة الأساسية" في الحالات التالية:

- أ - وجود قطاع خال من العمل لعدم وجود برامج تنتظر التنفيذ.
 ب - وجود برامج مؤجلة لعدم وجود القطاع ذي الحيز المناسب.
 ج - تفتت الحيز المتاح الى ثقب صغيرة Holes لا قيمة فعلية لها من حيث توافق الحيز مع متطلبات البرنامج المنتظرة.
 ويوضح الجدول التالي مضمون تفتت الذاكرة ، الذي يوضح أن أكثر من

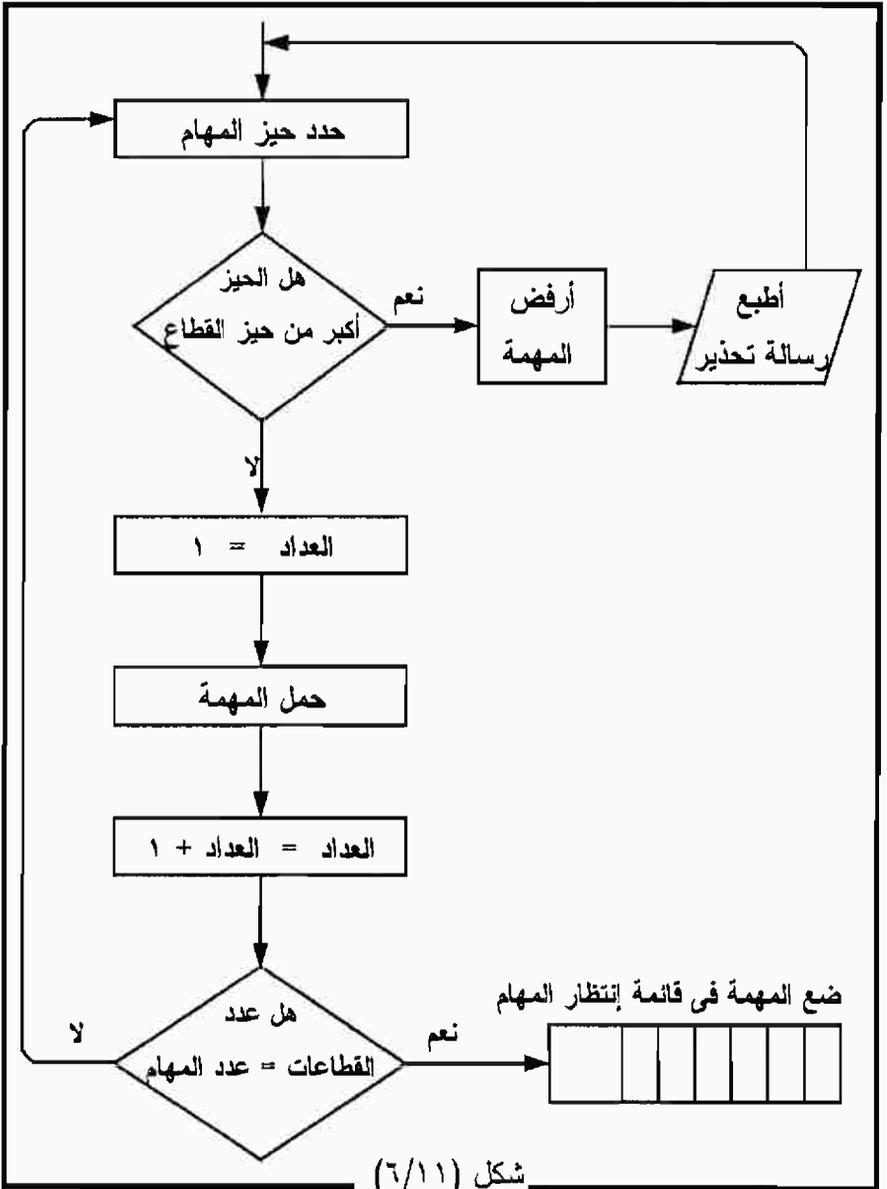
الحيز المتاح بوحدة كيلو بايت			رقم القطاع
حيز مفقود	حيز المهمة	حيز القطاع	
٧	١	٨	١
٢٣	٩	٣٢	٢
٢٣	٩	٣٢	٣
٨٨	٣٢	١٢٠	٤
٣٩٩	١٢١	٥٢٠	٥

٧٥% من جملة حيز مساحة الذاكرة الأساسية غير مستغل.

سياسات استخدام الحيز المتصل في إطار تعددية البرمجة:

ثبت أن استخدام تعددية البرمجة ينتج عنه تفتت يحتاج الأستفادة بها إجراء عملية جمع هذه الثقب على هيئة حيز واحد بإجراء عملية التضاعط Compaction وهي عملية لها مشاكلها وأثارها السلبية على كفاءة المنظومة ، منها أن تحديد الحيز مسبقا يعني تحديد درجة التعددية البرمجية المتاحة ، وهذا لا ينفي أنه يمكن إعادة تقسيم الذاكرة قبل بدء التشغيل لكنه إجراء غير متاح متى بدأ تشغيل المهام اللهم إلا إذا أوقف عمل المنظومة تماماً.

ويتم إختيار المهام في إطار سياسة الحيز المتصل مع تعددية البرمجة وفق المنطق الموضح في الشكل (٦/١١).



شكل (٦/١١)

ويتم تنسيق المهام وفق سياسات محددة هي:

- أ — أول حيز يمكنه قبول المهمة
- ب — أفضل حيز يناسب المهمة
- ج — أى حيز متاح
- أ . FIRST FIT
- ب . BEST FIT
- ج . WORST FIT

وتعتمد هذه السياسات على تقليل الجهد الذى سوف يبذله نظام التشغيل فى متابعة التفتيت [حتى لو كان حيزها مجرد بايت واحدة] حيث يعنى نظام التشغيل بإعداد قائمة للمواقع ، يرصد فيها حالتها سيات كانت مشغولة أو غير مشغولة. والواقع أن مدير الذاكرة ضمن إطار برامج نظام التشغيل يمارس مهامه فى متابعة الذاكرة مثلما يتابع أمين مكتبة يتردد عليها قراء ومستعيرون كثيرون منهم من يعيد كتباً ومنهم من يستعير كتباً أخرى ، لذا يعتمد أمين المكتبة إلى إعداد قائمة بالأرفف المشغولة والأماكن الخالية التى يمكنه إستغلالها ، فكل كتاب يعار يترك خلفه محلاً خالياً وكل كتاب يعاد يتطلب حيزاً يناسب ضخامة وعدد صفحاته ، فإذا كان أمين المكتبة من النوع الذى يدع الأمور تجرى على أعنتها فإنه سوف يحشر كتاب صغير فى موقع خال قد لا يتناسب مع حجمه (أكبر من اللازم) مما يفسد الموقع الحالى بفتات حيز لا قيمة لها ولا فائدة ترجى منها بينما لو كان أمين المكتبة من النوع الحريص والحصيف فإنه سوف يجمع حيزين أو ثلاثة ليلائم الحيز المطلوب لكتاب ضخم ، للأسف مدير الذاكرة أدق وأفضل فى تأدية مهامه من عشرات من البشر .

والسؤال الآن ما هى السياسة الأفضل؟ تتوقف الاجابة على عدة اعتبارات:

أ — الرغبة فى تحديد الحيز المناسب.

ب — أو الرغبة فى تسكين المهمة بسرعة.

فإذا كانت الرغبة هى الأولى فإن سياسة BEST FIT هى الأفضل والأنسب

أما إذا كانت الرغبة هى تحقيق السرعة فإن أفضل سياسة هى FIRST FIT ، أما إذا

كان الأمر لا هذا ولا ذاك فلا مفر من تطبيق سياسة لا سياسة النى تعنى WORST . FIT

مثال:

يوضح الجدول التالى قائمة المهام وزمن تشغيلها على المنظومة كما يوضح الشكل الحيز المتاح فى الذاكرة الموزعة توزيعاً ثابتاً والمطلوب توزيع المهام وفق السياسات المختلفة على ما هى عليه.

المهمة	الحيز المطلوب	زمن المعالجة
P1	٦٠٠	١٠
P2	١٠٠٠	٥
P3	٣٠٠	٢٠
p4	٧٠٠	٨
p5	٥٠٠	١٥

دقيقة

كيلو بايت

الحل:

١ - التوزيع الإستانىكى لقطاعات الذاكرة.

الذاكرة

نظام التشغيل
القطاع أ ٦٠٠ كيلو بايت
ب ١٠٠
ج ٥٦٠

نظام التشغيل
P 1
P 2
P 3

٢ - تم شحن المهام P1, P2 وفق سياسة BEST FIT أما المهمة P3 فقد شحنت وفق سياسة FIRST FIT ولذلك أحيطت بدائرة بما معناه أن هناك حيز غير مستغل.

نظام التشغيل
(P 1)
(P 2)
(P 3)

٣ - بعد خمس دقائق :

أ - خرجت المهمة P2 حيث تم تنفيذها.
ب - دخلت مكانها في القطاع ب المهمة P4 وفق سياسة WORST FIT ولذلك أحيطت بدائرة.

نظام التشغيل
(P 5)
(P 4)
(P 3)

٤ - بعد خمس دقائق أخرى :

أ - خرجت المهمة P1 لإنهاء تنفيذها وأحل محلها P5 حيث هي المهمة الباقية والتمشية مع الحيز.
ب - تسبب كل هذه المهام تفتتت في القطاعات.
ج - إدخال P5 تم وفق سياسة FIRST FIT.

نظام التشغيل
قطاع خال
قطاع خال
(P 3)

٥ - بعد خمس دقائق أخرى :
 أى بعد ١٥ دقيقة من
 بدء التشغيل يصبح شكل
 الذاكرة على النحو:

ثانياً: التوزيع الديناميكي للذاكرة :

فى أسلوب الحيز المتصل يقصد بالتقسيم الديناميكي للذاكرة الأساسية إجراء عملية تحديد وإنشاء القطاعات أثناء تنفيذ المهام حتى يتلائم حيز الذاكرة مع حيز المهمة ، ويعتمد التوزيع الديناميكي للذاكرة احياناً على طبيعة لغة البرمجة المستخدمة

، ففى لغة البرمجة LP-١ [لغة البرمجة رقم ١ وهى لغة خاصة أبتكرتها شركة IBM وتجمع بين خصائص لغتى الكوبول والفورتران] يستخدم الأمر ALLOCATE لإضافة حيز جديد من الذاكرة الى حيز القطاع ، كما أن استخدام الأمر FREE يخلى القطاع تماماً من المهمة.

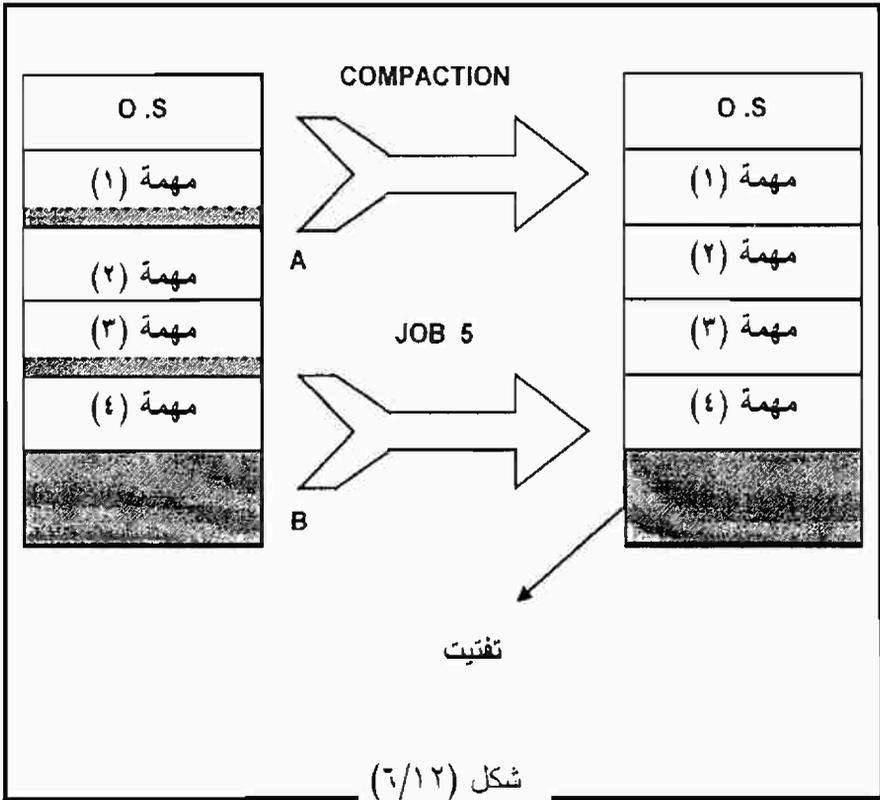
ويتطلب التوزيع الديناميكي للذاكرة وجود بيانات كافية على هيئة جداول توصف المناطق المخصصة لمهام والمناطق الحرة المتاح استخدامها.

ولعل أبرز أنظمة التشغيل التى تلبى حاجات التوزيع الديناميكي هو نظام تشغيل IBM المعروف بأسم OS/MTV .

ويعيب هذا الأسلوب احتمال حدوث تفتيت محدود في الذاكرة رغم أنه يمكن استغلاله في تنفيذ مهمة منتظرة بشرط توافق حيز المهمة مع حيز الثقب أو يتم دمج بعض هذه الثقوب لتعطي حيزاً أكبر مناسباً لعمل منتظر ويطلق على هذا الإجراء عدة مسميات أبرزها:

- * STORAGE COMPACTION.
- * BUAPING THE STORAGE.
- * GARBAGE COLLECTION.

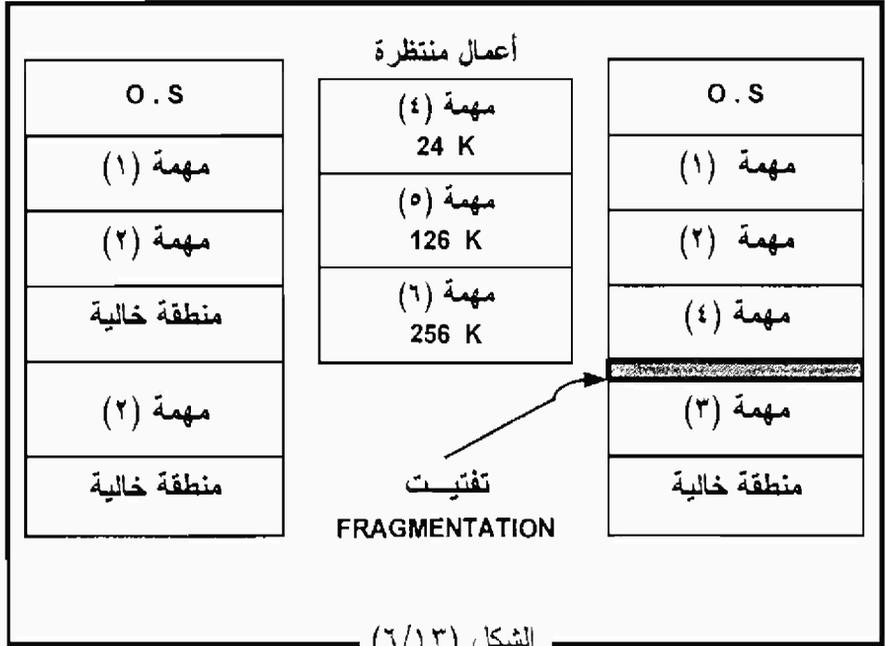
فيما يوضحه الشكل (٦/١٢):



وتطلق المسميات التالية على التوزيع الديناميكي للذاكرة:

- * Dynamic Storage Allocation.
- * Controlled Storage.

ويوضح الشكل (٦/١٣) أسلوب التعامل مع التقسيم الديناميكي:



ودمج الثقوب عملية بسيطة من وجهة النظر المنطقية لكنها لاتضمن نفس

أداء البرامج في المواقع الجديدة لعدة أسباب أهمها حساسية بعض المواقع مثل:

أ - مسجلات القاعدة.

ب - المواقع المحددة للعنونة.

ج - قوائم المؤثرات.

د - تركيب البيانات.

ورغم توفير التضاضط COMPACTON لحيز آخر للإستغلال إلا أنها تسبب عدة أمور تؤثر سلباً على أداء النظام مثل:

أ - تستقطع جزء من وقت التشغيل الفعلى كان يمكن إستغلاله فى تنفيذ عمل أو عدة أعمال.

ب - يتوقف النظام عن تأدية أى عمل آخر سوى إجراء دمج الثقوب مما يؤثر على كفاءة إتاحة النظام للمستخدمين.

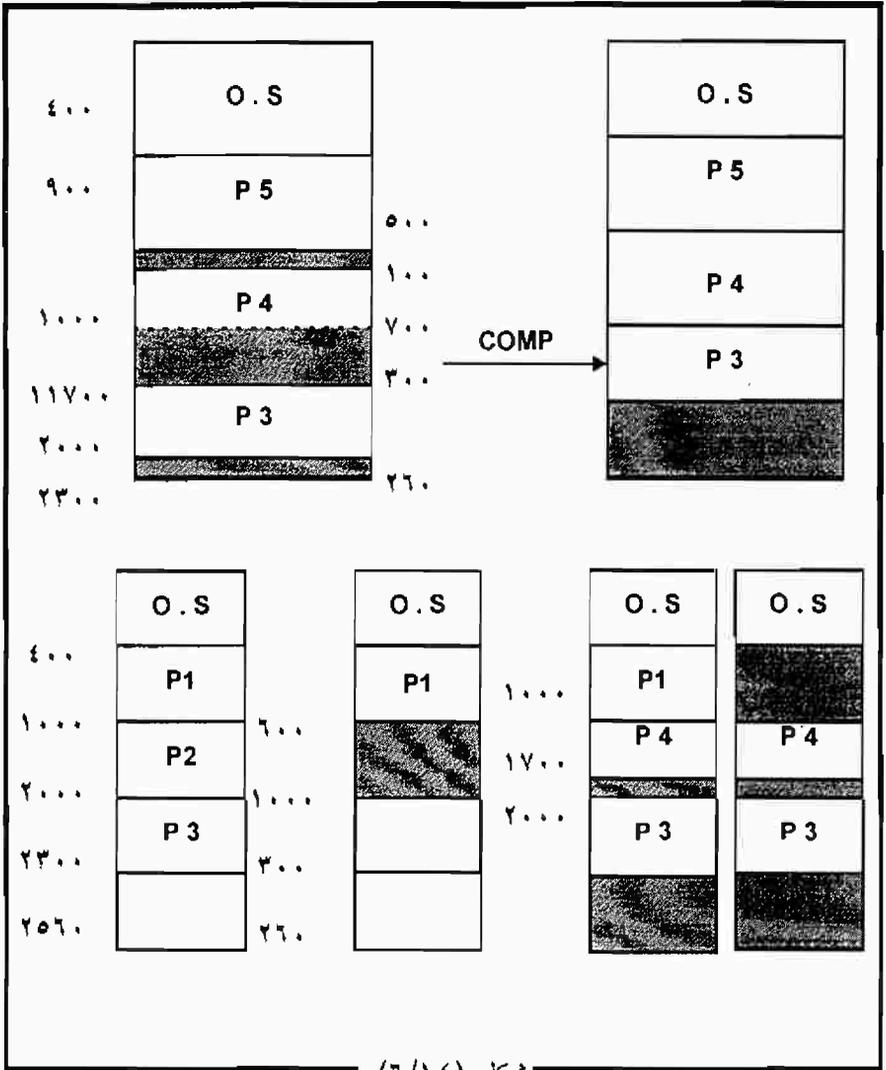
ج - تتطلب المواقع عنونة جديدة مما يؤثر على كفاءة الأداء.

وهذه العيوب والمخاطر لا علاج فنى لها سوى إعادة التشغيل من بداية البرامج مما يسبب ضياع وقت وحدة التشغيل المركزية.

كما أن البرامج التى تفضى إلى نتائج غير عكوسة IRREVERSIBLE لايمكن إتخاذ هذا الإجراء حيالها مما يجعل عملية COMPACTON محفوفة بالمخاطر وتحتم إعادة تشغيل هذه البرامج مرة أخرى.

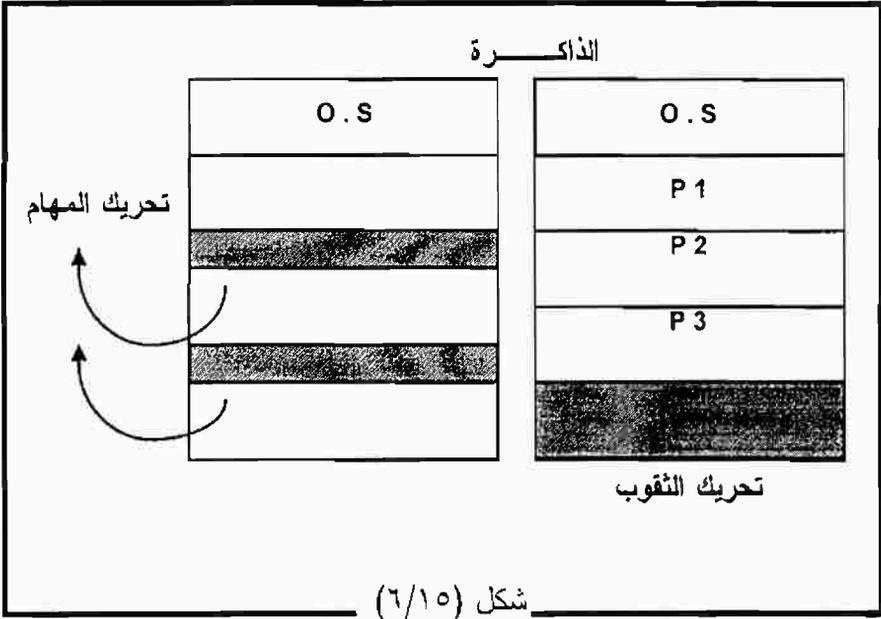
معنى هذا أن التضاضط ليس متاحاً دائماً ويوضح الشكل (٦/١٤) مثال عن ما عرضنا من مصاعب:

يستحيل إجراء عملية التضاضط إذا كانت المهام يتم تحميلها خلال مراحل الترجمة لان التضاضط يعنى تغيير عناوين [P4] [P3] وهى عملية مكلفة وحساسة ويتم التضاضط فى حالة التسكين الديناميكى ويتم وقت التنفيذ.



شكل (٦/١٤)

وأبسط صور التضاضغط هي تحريك البرامج والمهام إلى الطرف الأعلى للذاكرة بينما تتحرك الثقوب إلى الناحية الأخرى من الذاكرة كما في الشكل (٦/١٥):

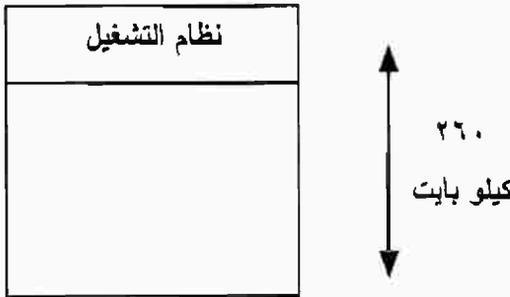


وهذه العيوب لا علاج فني لها سوى اعادة التشغيل من بداية البرنامج مما يسبب إهدار وقت المنظومة ، إضافة إلى احتمال أن البرامج ذاتها لايمكن تنفيذها إلا بعد إعادة تسكينها.

مثال:

عالج المثال السابق باستخدام إمكانية إجراء تضاعف .

الحل:



١ - حدد الحيز الذى يصلح لثلاث مهام هي P1 , P2 , P3 على النحو الموضح فى الشكل ونلاحظ بقاء حيز غير مستخدم سعة ٢٦٠ كيلو بايت لا يصلح لأى من [P4] أو [P5] .

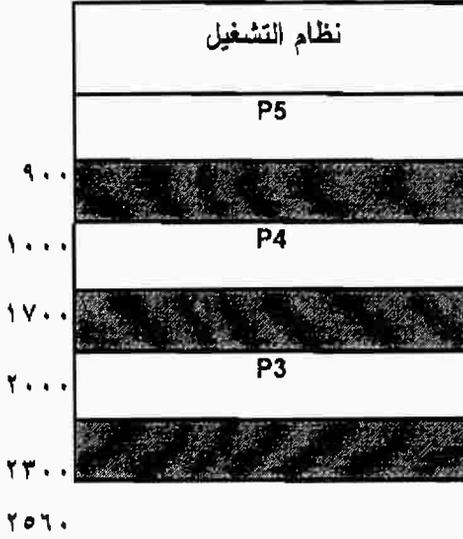
نظام التشغيل
٦٠٠ = P1
١٠٠٠ = P2
٣٠٠ = P3
(٢٦٠)

٢ - بعد خمس دقائق تنهى المهمة [P2] ويصبح شكل التوزيع داخل الذاكرة على النحو:

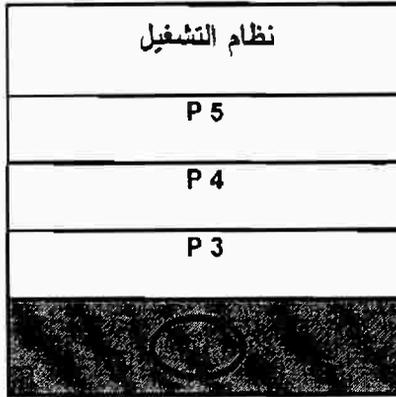
نظام التشغيل
P 1
P 3

نظام التشغيل
P 1
P 4
P 3

٣ - بعد فترة تنهى [P1] تخلى من الذاكرة وتستقبل المهمة [P5].



٤ - بإجراء التضاضط :



تقسيم القطاع الواحد:

يقصد بتقسيم القطاع الواحد امكان تنفيذ برامج صغيرة في قطاعات متعددة تتصف بالصغر ومنفصلة عن بعضها البعض ، فإذا كان البرنامج [المهمة] يحتاج مئة كيلو بايت فانه يمكن تشغيله في قطاع مستقل بذات الحيز أو خمسة قطاعات صغيرة كل منها بحيز ٢٠ كيلو بايت.

ويؤدي تقسيم القطاع الواحد إلى بعض المزايا هي:

أ — تعددية البرمجة على مستوى القطاع.

ب — لا يحتاج هذا الأسلوب إلى معدات خاصة.

العيوب :

أ — يعاني من نفس مشكلة تفتيت الذاكرة.

ب — حتى في حالة عدم وجود فتات فان الحيز الخال لا يمكنه تشكيل قطاع مستقل.

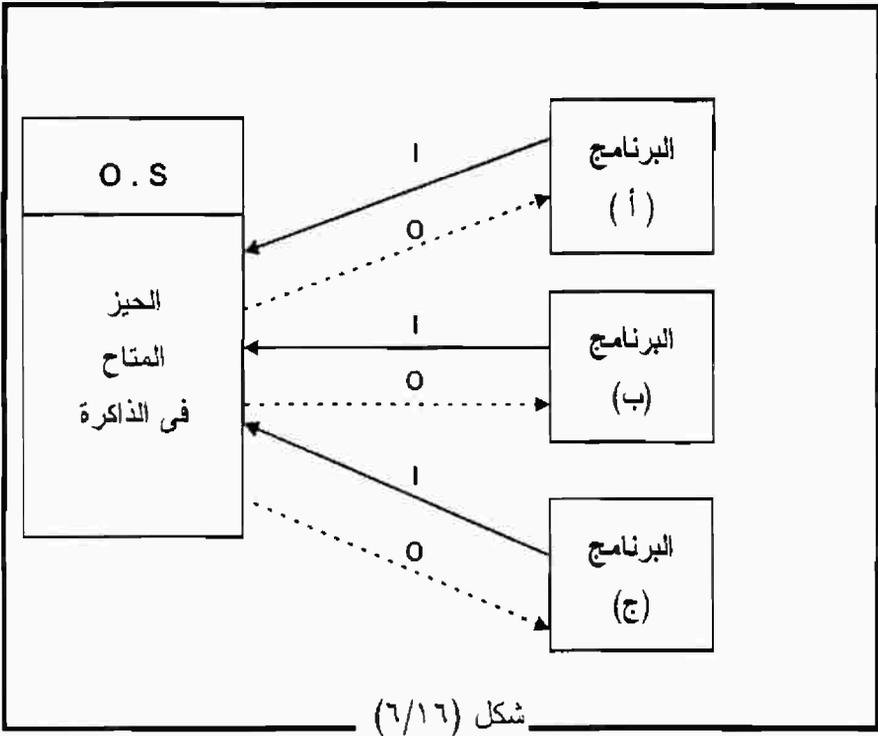
ج — يحتاج نظام تشغيل معقد.

الحيز المتصل والتبديل SWAPPING :

على مستوى المهام:

في كل ما سبق ذكره نلاحظ أن البرنامج الجارى تنفيذه يبقى فى الذاكرة الأساسية طوال فترة تشغيله وحتى يستكمل ، والسؤال : ولماذا لا يتم الجمع بين مزايا أسلوب الدفعة ومزايا أسلوب تعددية البرمجة؟

والإجابة فى اقتراح أسلوب تعددية البرامج بالتبديل ، والفكرة ببساطة أن برنامج واحد يشغل حيز الذاكرة الأساسية كله مرة واحدة ويتم تشغيله حتى يعترض التنفيذ صعوبة تمنع أستمراره ، وفى هذه الحالة يتم إخلاء الذاكرة الأساسية وتوجيه كل إمكانيات النظام صوب برنامج آخر وثالث وهكذا...فيما يوضحه الشكل (٦/١٦):



١ - أعتبارات التبديل:

أ - يستمر تشغيل البرنامج ويخلى من الذاكرة الأساسية في الحالات التالية:

- (١) إذا طلب عملية إدخال / إخراج.
- (٢) إذا تجاوز الوقت المخصص له دون I/O.
- (٣) إذا حدثت ظروف قاهرة أثناء التشغيل.

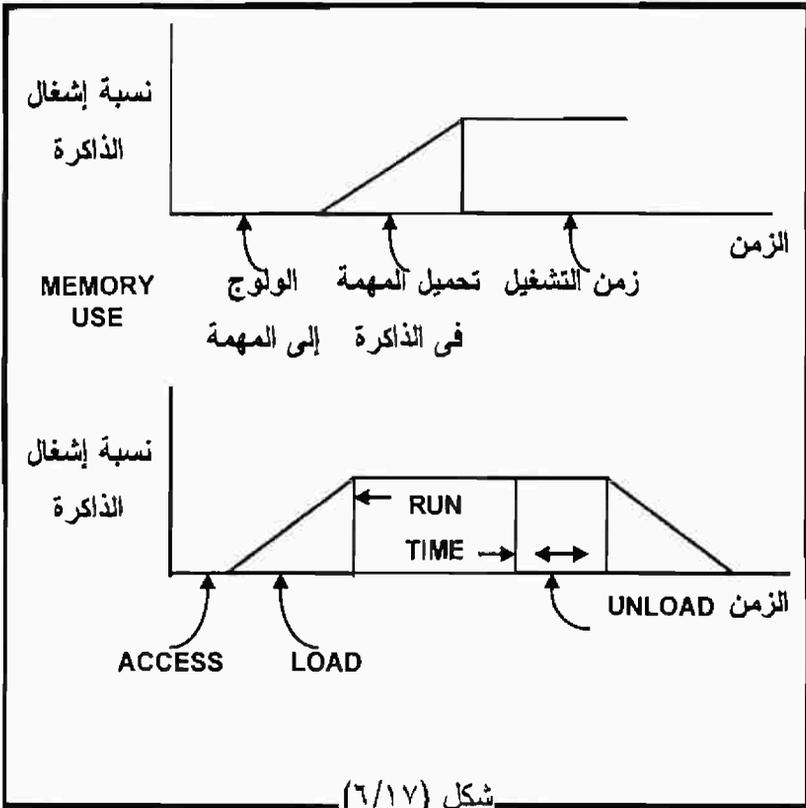
ب - تطبع نسخة من البرنامج على وسائط التخزين الثانوية.

ج - يماثل أسلوب التبديل أسلوب المشاركة في الوقت على مستوى وحدة التشغيل المركزية.

د - تطور هذا الأسلوب أفضل إلى إمكانية تواجد أكثر من برنامج في حيز منطقة التبديل مما يتطلب إدخال تعديلات ضخمة على نظم التشغيل.

٢ - المنحنى الزمني للتبديل:

يوضحه الشكل (٦/١٧) ومنه يتضح أن التبديل يتطلب فترة زمنية للوصول إلى المهمة ، بعدها تتم قراءتها وتحميلها في الذاكرة الأساسية مما يستغرق زمناً آخر يضاف إلى إجمالي الوقت.



بعدها يأتي وقت التشغيل ، ويضاف وقت آخر لإزاحة المهمة إلى وسائط التخزين.

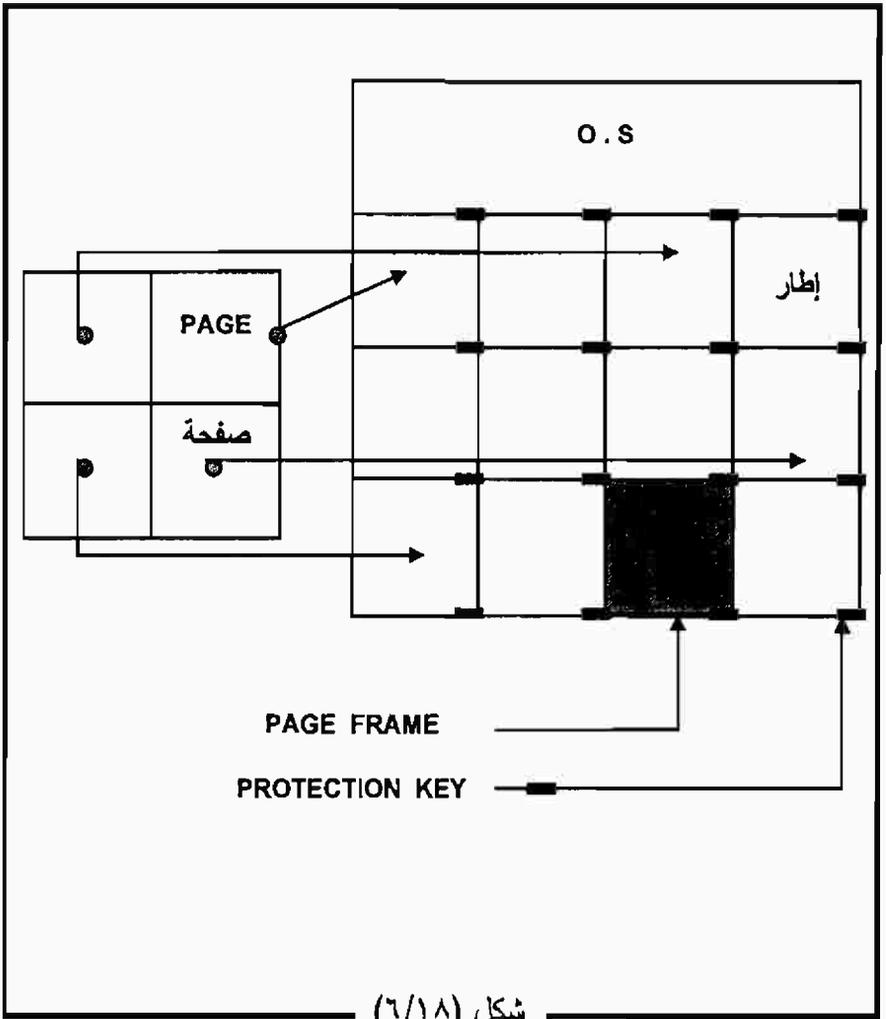
ثانياً: أساليب الحيز غير المتصل:

التصفح PAGING - فكرة مبسطة:

يعتبر أسلوب التصفح حلاً جيداً لمشكلة تفتيت الذاكرة ومشاكل اللاحقة للتضاغط COMPLICATION ، وفي هذا الأسلوب تقسم الذاكرة إلى عدد كبير من حيز ثابت صغير نسبياً يتراوح بين ٥،٠ كيلو بايت وحتى أربعة كيلو بايت ويسمى كل حيز إطار FRAME ، كما تقسم البرامج والبيانات على الأقراص إلى كتل مساوية لحيز الاطار وتسمى صفحة PAGE .

وحتى يتولى الكيان الآلى نقل كل صفحة إلى إطارها الصحيح تحت تعددية البرمجة فإن ذلك يتطلب استخدام عدد كبير من مسجلات الوقاية ، وقد تكون المسجلات جزءاً من الكيان الآلى أو أجزاء من الذاكرة وهذا يتوقف على سعة الذاكرة الأساسية.

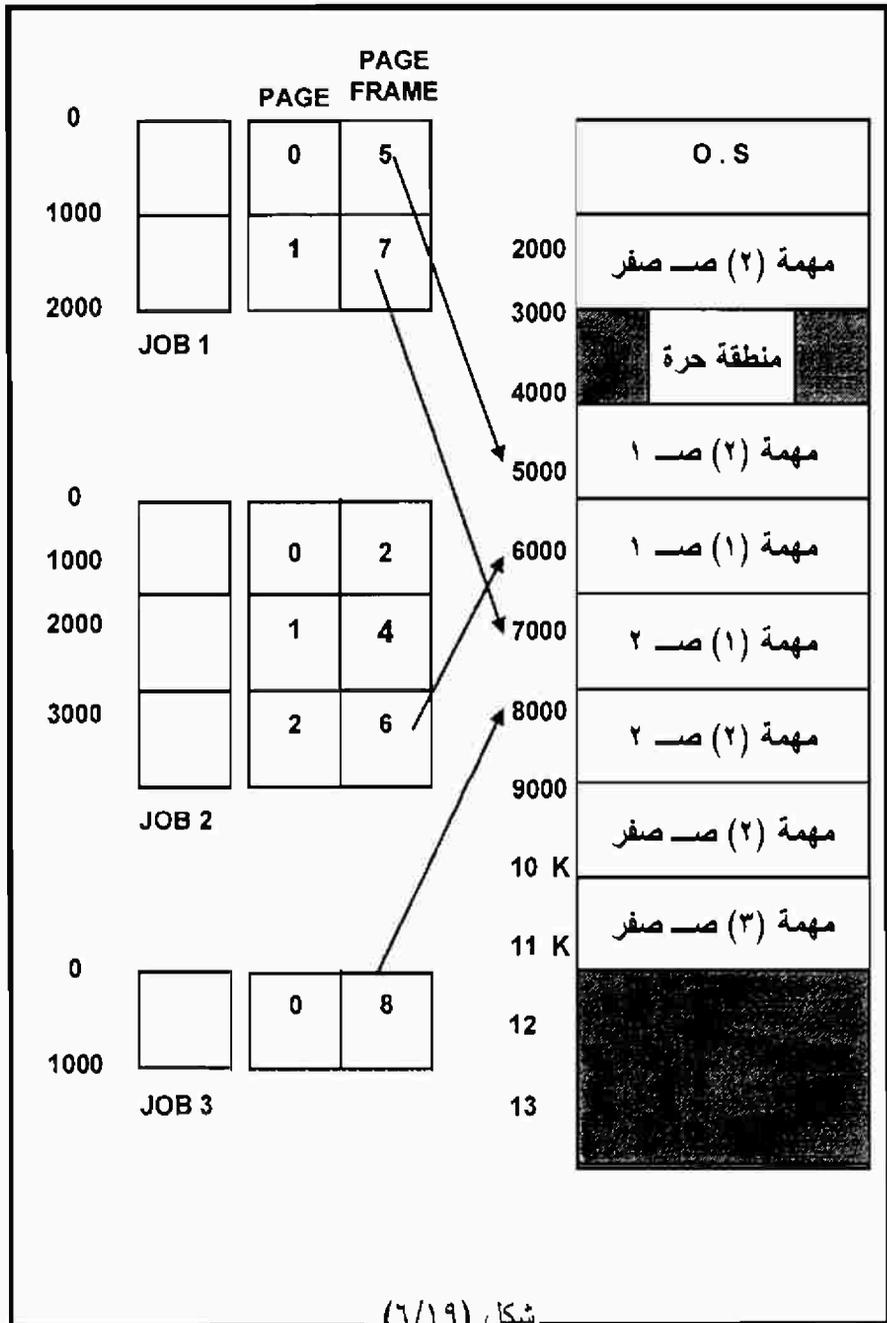
ويوضح الشكل (٦/١٨) المفهوم الأساسى للتصفح الذى يتطلب عنونة الصفحات وعنونة أطر الصفحات المناظرة وخلق جداول التطابق بين هذه وتلك ، ورغم التكلفة الإضافية للنظام إلا أنه أسلوب أثبت كفاءة جيدة فى معالجة المشاكل التى سبق لنا التعرض لها فى إدارة الذاكرة وأبرزها مشكلة التفتيت وإشغال مدير الذاكرة فى متابعة مواقع خالية.



شكل (٦/١٨)

مثال:

المطلوب توزيع المهام الثلاث الموضحة بالشكل (٦/١٩) على الذاكرة الأساسية وفق أسلوب التصفح ، مع استخدام القطاعات:



شكل (٦/١٩)

وبوضح هذا الشكل كيفية إيجاد التطابق MAPPING بين عناوين الصفحات وعناوين أطر الصفحات عن طريق الجداول ، وبذلك أمكن حل مشكلة التفتيت دون توزيع الصفحات أو القطاعات.

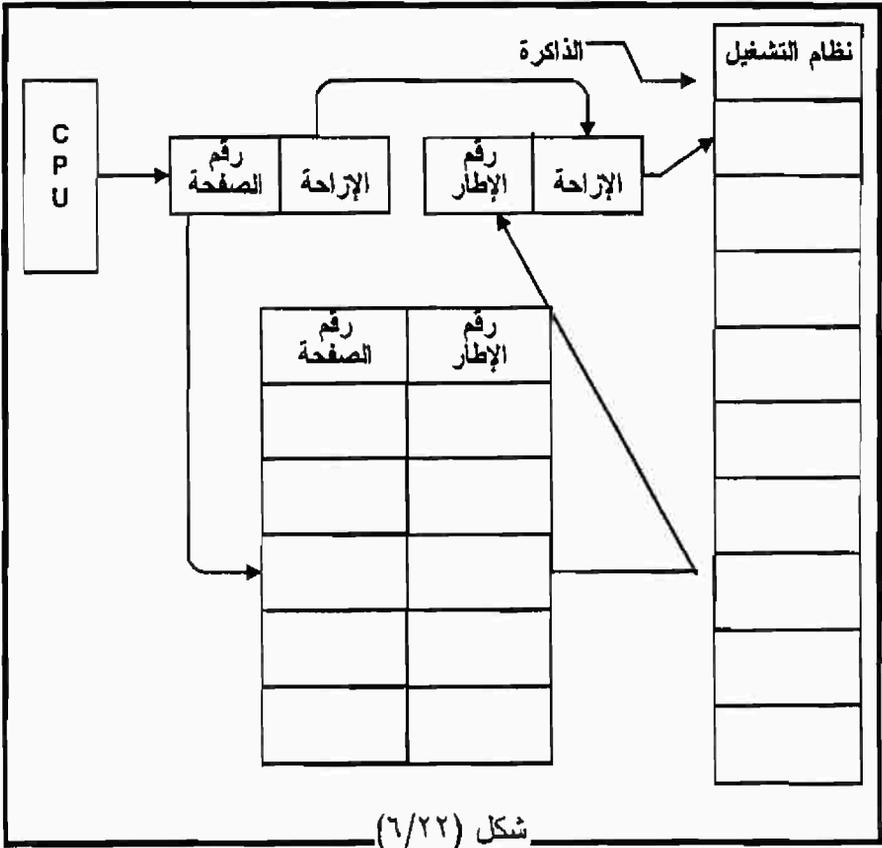
فإذا كان هناك عمل أو مهمة منتظرة ولتكن المهمة (٤) وتحتاج إلى ٥ كيلو بايت فإنه يمكن توزيع صفحاتها بين العنوان ٣٠٠٠ والعنوان ١٣٠٠٠/١٢٠٠٠/١١٠٠٠/١٠٠٠٠.

حيث يتم تخليق جداول التطابق آلياً حيث تحفظ في الذاكرة الأساسية وتحفظ إلى جوارها CSW، CAW، PSW لكل صفحة حيث تتغير هذه الكلمات آلياً عندما يقوم بالفصل والاتصال بين الصفحات حيث تجمع هذه الكلمات لكل برنامج في جداول خاصة.

PAGE	STATUS	PAGE FRAME NO.
0	Y	5
N	N	لم يحدد بعد

الدعم الآلي للتصفح:

يوضح الشكل (٦/٢٠) الدعم الآلي المطلوب ، وعندما تستدعي وحدة التشغيل أى إيعاز فإنها تخلق له العنوان المنطقي الذى يشمل رقم الصفحة والإزاحة ، ويستخدم رقم الصفحة فى تحديد رقم الإطار المقابل وتبقى الإزاحة داخل إطار كما هى بالنسبة للصفحة .



وبذلك يتخلق العنوان الفيزيائي - الواقعي - في الذاكرة به لحين وضع قيمه في مسجل العناوين MAR ليستدعى الابعاز كما سبق وعرضنا في الباب الثاني ، وبذا تتناظر عملية التصفح وادارتها عملية إعادة تسكين المهام في الذاكرة.

*