الباب السادس

٦

أدارة الذاكرة الأساسية

يعتبر تنظيم وقيادة الذاكرة الأساسية أو بمعنى آخر ادارة الذاكرة واحداً من أهم الأعمال التى توليها نظم التشغيل إهتماماً كبيراً بإعتبار الذاكرة الأساسية المكان الوحيد الذى 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 النسبى مقارنة بتكلفة تخزين بايت واحدة على الذاكرة الأساسية ، ورغم أن التكنولوجيا الحديثة عدلت وطورت وأفرزت ذاكرات أساسية تتصف بكبر حيز تخزينها فلا تزال تكلفة تخزين بايت على الذاكرة الأساسية أعلى من نظيرها على الذاكرات الثانوية، وان عكس ما أسلفنا شيئا 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 الايحاء 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 أخترنا 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 أتصال الحيز CONTIGOUS أم من الممكن إجراء التقسيم على هيئة صفحات PAGES لاتكون 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 تنفيذه إلى النظام؟ هل ندفع به متى طلب النظام ذلك أم نزج به إلى النظام متى شئنا؟ وأين ندفع بهذا البرنامج فى حيز خال أو ننتظر حتى تتجمع الثقوب HOLES (والتى تسمى أحياناً فتات التوزيع Fragmentation) فى حيز أكبر صالح للأستخدام بعد عملية دمج الثقوب COMPACTION .

أسئلة كثيرة حول قيادة الذاكرة مثل هل عند طلب تنفيذ برنامج فى قائمة الإنتظار فما هو البرنامج الذى يجب تنحيته 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 خللاً فى أداء منظومة العمل.. أم هو البرنامج الذى يستهلك وقتا 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.. ففى الجيل الثانى وبدايات الجيل الثالث كانت الذاكرة تصنع من حلقات مغناطيسية وكان ثمنها مرتفعاً لذلك حدت من التفكير فى نظم تشغيل متطورة كما حددت أعداد مستخدمى منظومة الحاسب.. ثم تطورت عبر رحلة تكنولوجية طويلة وصولاً إلى

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

ونخلص إلى أن ادارة الذاكرة تستهدف أربع وظائف أساسية هى:

أ ــ مراقبة حالة STATUS جميع المواقع المعنونة سيان كانت موزعة أو غير موزعة.

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

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

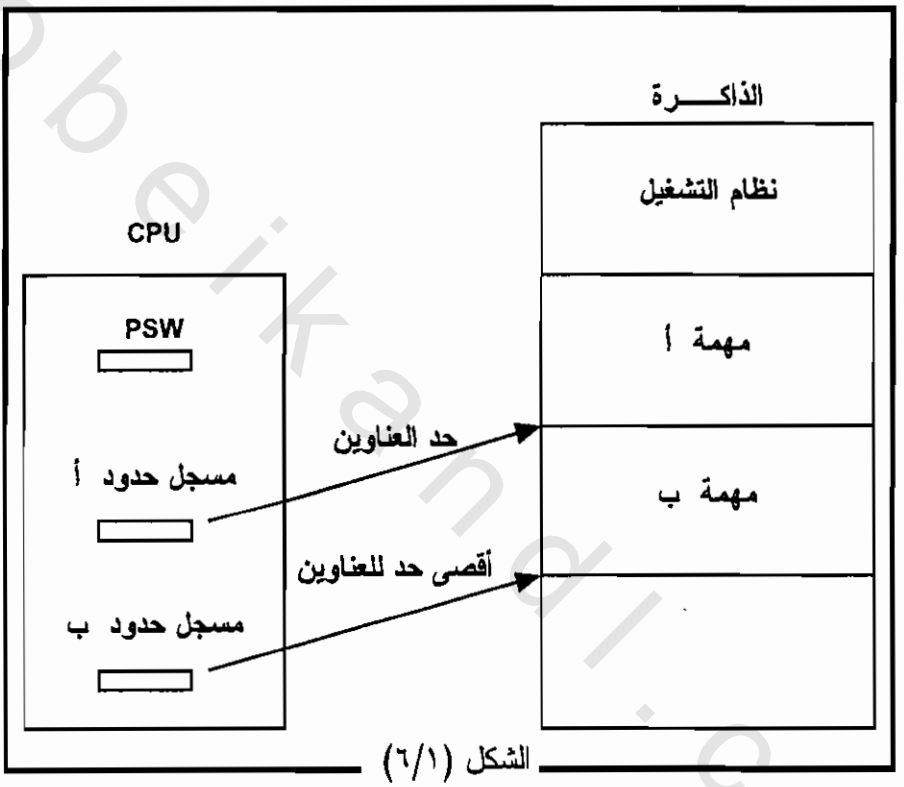
د ــ تحديد أسلوب التعامل مع الحيز الفائض أو الثقوب الناجمة عن إخلاء بعض المناطق الموزعة.

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 سيان 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 ــ وقاية الذاكرة:

يقصد بوقاية الذاكرة سيان شغلتها 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 أستخدام مفاتيح وقايـــة لكـــل مهمة Protection Key ويوضح الشكل (٦/١) أستخدام مسجلات الحدود بحيث تتم مراجعة قيم هذه المسجلات بواسطة المشغل CPU كلما نشطت المهمة إن تطابقت القيم سمح للمهمة بإستكمال المعالجة ، أما إذا تدنت القيم وحاولـــت

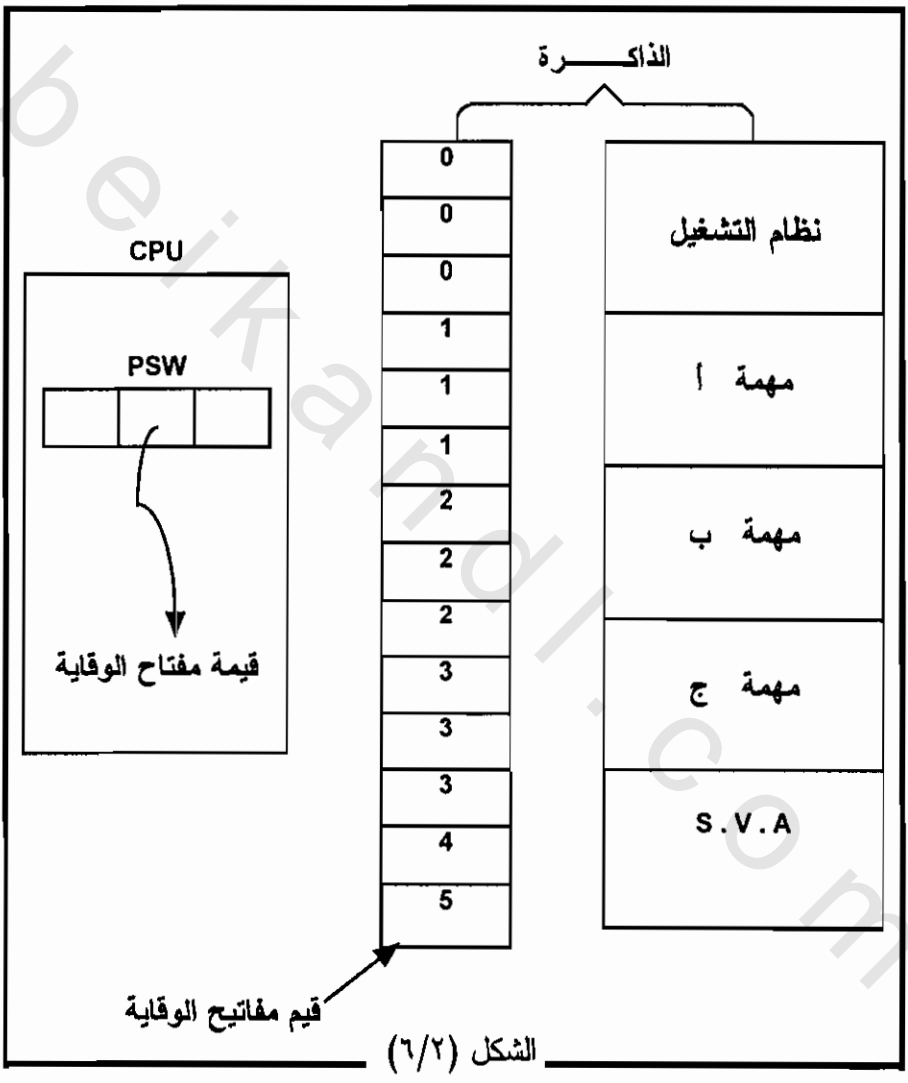


الشكل (٦/١)

مهمة اخرى اختراق الحدود تتم مقاطعة على المشغل ، كما يوضـــح الشكـــل (٦/٢) استخدام مفاتيح الوقاية التى تتطلب تقسيم الذاكرة إلى قطع متساوية الحيز ولكل قطعة مفتاح يخزن فى حيز (بايت) داخل الذاكرة مثلما تسجل قيمته فى حـــيز PSW وتتـــم

مراجعته بواسطة وحدة التشغيل المركزية ومطابقة قيمته فى PSW والذاكـــرة فـــإن تطابقا كان بها وسمح بالولوج إلى حيز المهمة.

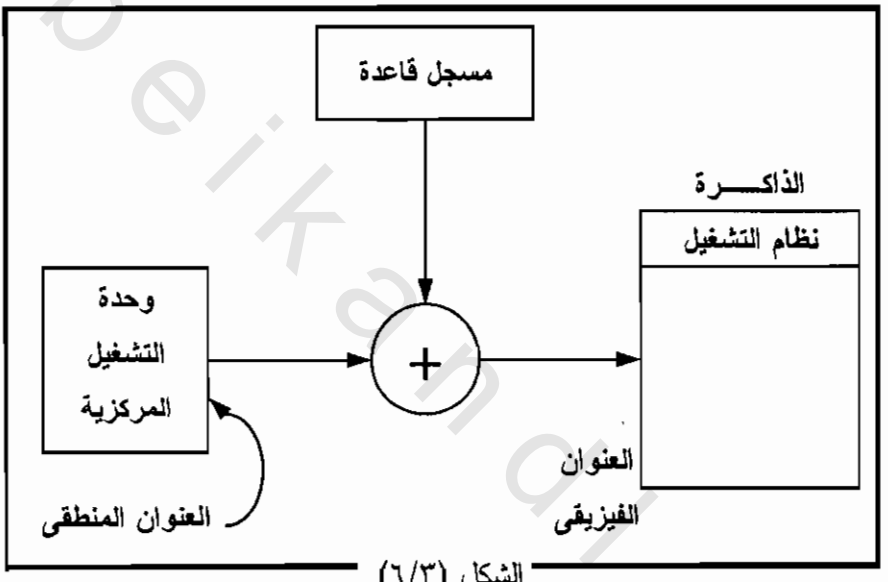
وفى كلا الأسلوبين يتم حذف أو تعديل قيم المفاتيح عند إنتهاء تنفيذ المهمـــة وإستعادة مدير الذاكرة للمحلات الخالية.



الشكل (٦/٢)

٢ ــ إعادة تسكين المهام:

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



الشكل (٦/٣)

أدارة الذاكرة:

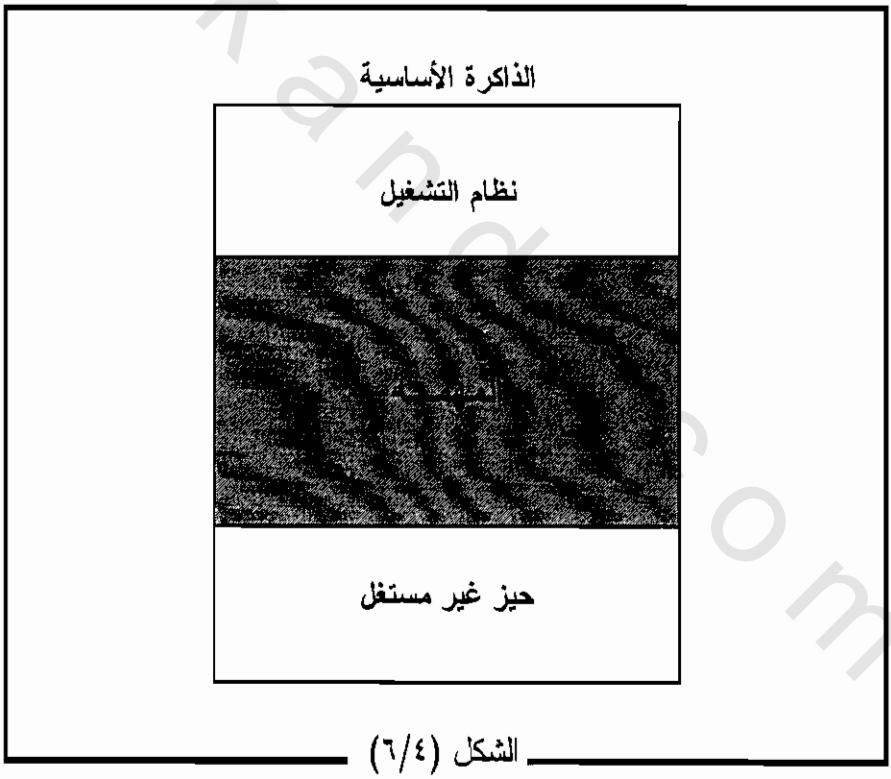
أولا : ادارة الذاكرة وفق اسلوب الحيز المتصل لمهمة واحدة:

نظام المستخدم الواحد :

نظام الدفعة BATCH :

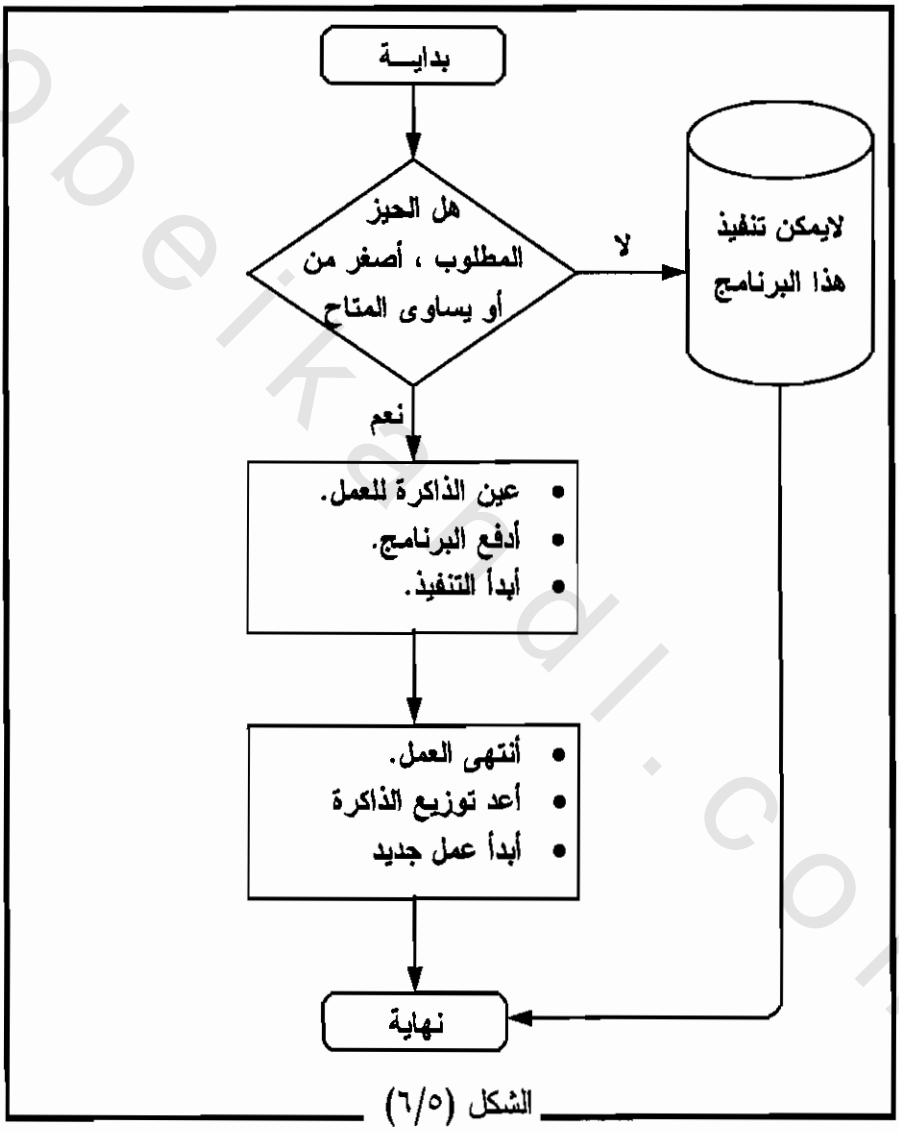
يعتبر هذا النظام أقدم النظم فى أدارة الذاكرة ، إذ طبق مع بدايات أستخدام الحاسبات ، ومازال مستخدما فى بعض المواقع والتطبيقات ، ويفترض هذا الأسلوب إتاحة جميع موارد النظام لمستخدم واحد بشرط ألا يزيد حيز المهمة عن الحيز المتاح فى الذاكرة.

ومن وجهة النظر المنطقية فأنه 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.



الشكل (٦/٤)

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



الشكل (٦/٥)

مميزات الحيز المتصل لمستخدم واحد:

أ ــ بسيط والتعامل معه بسيط.

ب ــ رخيـــــــص.

ج ــ لا يحتاج دعماً فنياً أو آلياً خاصاً.

د ــ أسلوب وقاية البرنامج من نظام التشغيل بسيط.

هـ ــ ترص المهام على هيئة قطار إنتظار Stream of Jops ويطبق عليها سياسة من يأتى أولاً يخدم (يشغل) أولاً.

عيوب أسلوب الحيز المتصل:

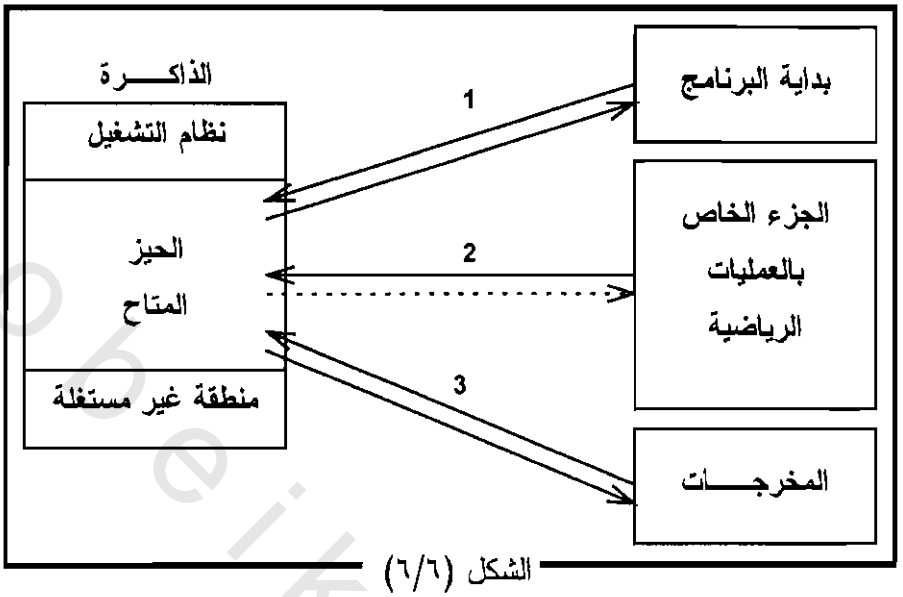
أ ــ عدم أستغلال حيز الذاكرة الأستغلال الأمثل وكذلك عدم استغلال قدرة وحدة التشغيل المركزية الأستغلال الأوفق حيـث تتوقـف عـن أداء دورها وتتفرغ لمراقبة معدات I/O البطيئة.

ب ــ طول مدة إنتظار المهام الأخرى حيث عليها البقاء راكدة حتى يأتى عليها الدور.

ج ــ أسلوب غير مرن حيث يعجز عن معالجة مهمة يزيد الحيز المطلوب لها عن الحيز المتاح.

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

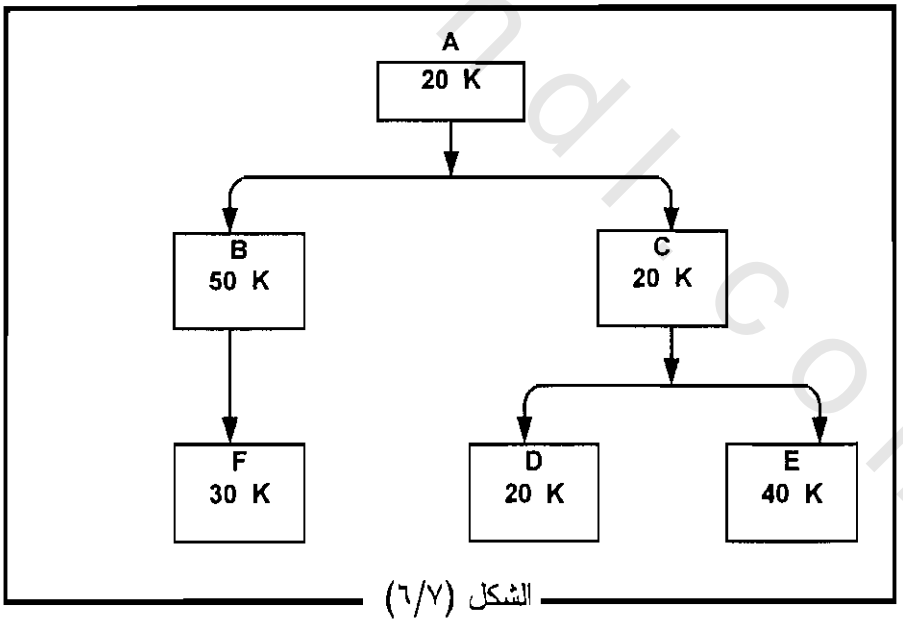
وما يعيب هذا الأسلوب أنه يتطلب مهارة عالية فى صياغة منطق البرنامج ، كما يتطلب نظام تشغيل ملائم.



الشكل (٦/٦)

مثـــال:

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



الشكل (٦/٧)

الحـــل:

الكتلة (A) تستدعى (B) أو(C) فقط ، بينما (B) تستدعى فقط (F) فـــى حيـــن تستدعى (C) الكتلة (D) أو (E) ، وكنتيجة لـــهذا المخطـــط أن (B) لا تســـتدعى (C) والعكس صحيح ، وبالتالى فأنه يستحيل تواجد (B) أو (C) سوياً فى الذاكرة الأساسية لذا يستخدم برنامج مشرف إحلالـــى OVERLAY SUPERVISOR علـــى أن ينـــص المبرمج صراحة على منطق الإتصالات بين كتل البرنـــامج ، للفصـــل بيـــن هـــذه الخطوات.

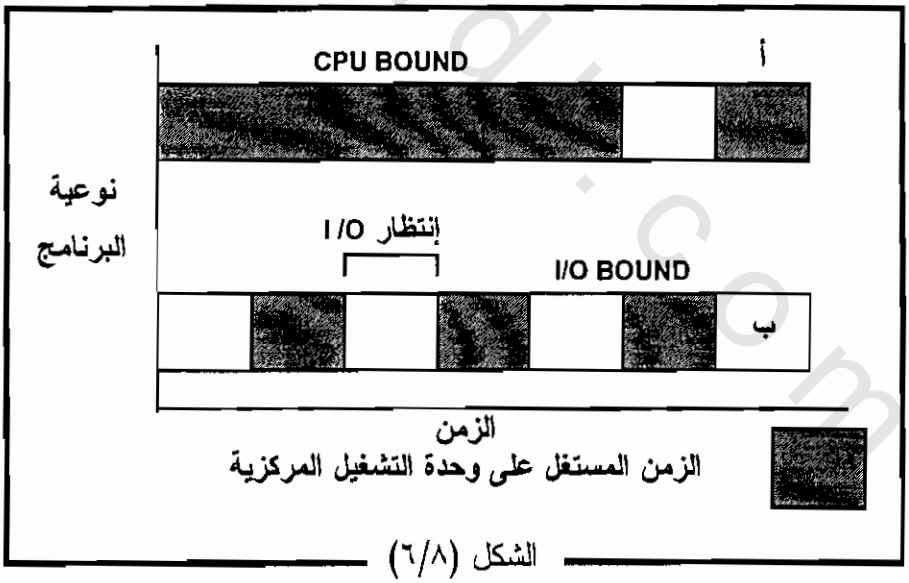
تأثير طبيعة البرامج على أسلوب الحيز المتصل:

تعتبر البرامج ذات الطبيعة العلمية أصلح أنواع البرامج لنظام الدفعة حيـــث 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)

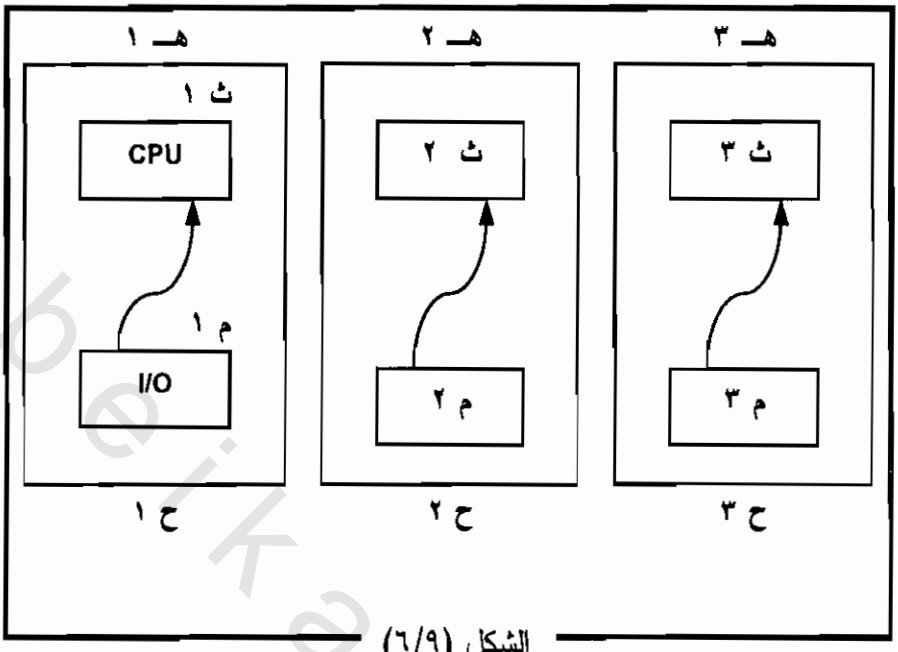
أسلوب الحيز المتصل لعدد من المهام MULTI PROGRAMING:

يعتبر هذا الأسلوب علاجاً للقصور الواضح فى أسلوب المهمة الواحدة فى أستغلال الموارد خاصة فى البرامج التى تعتمد على المدخلات والمخرجات ، حيث يهدر البرنامج وقت وحدة التشغيل المركزية إهداراً فادحاً مما دعا إلى التفكير فى دفع عدة برامج فى نفس حيز الذاكرة الأساسية ، وتتولى وحدة التشغيل المركزية الأنتقال بينها من خلال سرعة الفصل والأتصال SWITCHING وتوضح الفقرة التالية مدى الأرتقاء بكفاءة النظام من حساب زمن الإنتظار الذى يواجهه كل برنامج.

حساب زمن الانتظار:

تفرض أن هناك ثلاثة مهام سوف تنفذ على الحاسب هى: [هـ١] ، [هـ٢] ، [هـ٣] على التوالى ، وأن [هـ١] يستهلك [ث١] من إجمالى 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 التوالى ونقارن بين تطبيق أسلوب نظام الدفعة وبين أسلوب البرمجة المتعددة فيما يوضحه الشكل (٦/٩) ويلخصه الجدول التالى:

المهام	الحيز	CPU	I/O
هـ ١	ح ١	ث ١	م ١
هـ ٢	ح ٢	ث ٢	م ٢
هـ ٣	ح٣	ث ٣	م ٣



الشكل (٦/٩)

بالتالى يكون الحيز المتاح : ح١ + ح٢ + ح٣.

فى حالة نظام المهمة الواحدة:

* بفرض أن تم تنفيذ أحد المهام ولتكن [هـ١] فقط.
* بالتالى فإن الحيز غير المستغل فى الذاكرة الأساسية هو ح٢ + ح٣.
* أيضاً فإن نسبة إنتظار وحدة التشغيل المركزية حتى تنتهى عمليات المدخلات والمخرجات :

$$= \frac{م١}{م١ + ث١}$$

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

$$= \frac{\text{م١}}{\text{م١ + ث١}} \times ١٠٠$$

فى حالة أستخدام أسلوب تعددية البرمجة:

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

* إذا فإن نسبة إشغال الذاكرة الأساسية = ١٠٠%.
* حيث أن النظام يعطى أمراً بتنفيذ [هـ١] فإنه فور إنتهاء [ث١] لن تنتظر وحدة التشغيل المركزية انتهاء [م١] بل توجه إلى المهمة [هـ٢] حيث تنفذ عملية حسابية تستغرق [ث٢] وحدة زمن وعند الفراغ منها سوف توجه وحدة التشغيل المركزية إلى [هـ٣] حيث تنفذ عملية حسابية تستغرق [ث٣] وعندما تنفذ [م٣] سوف توجه نحو [ث١].
* بفرض أن [م١] أصغر زمنا من [م٢] + [م٣]،
* بالتالى فإن زمن الإنتظار WAIT فى الحالتين:

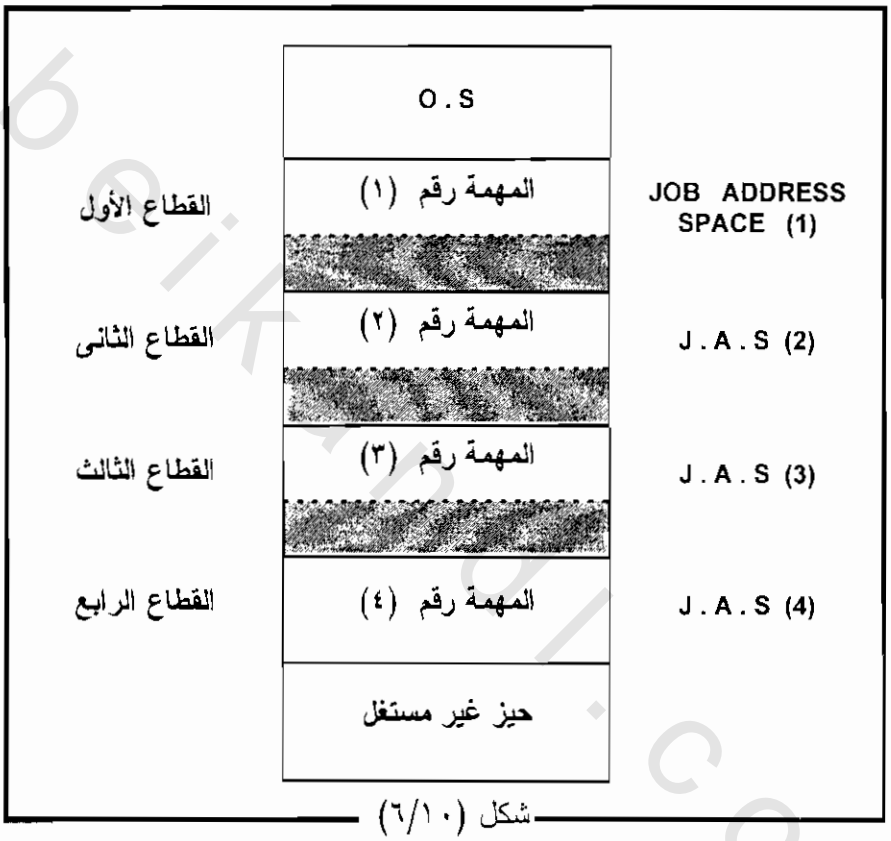
نجد أن (ث٢ + ث٣) – م١ أقل من زمن انتظار الدفعة الواحدة ، وترتيباً على ذلك فإن أسلوب تعددية البرمجة يحقق للنظام كفاءة أعلى من أسلوب الدفعة الواحدة.

* فى الحالتين السابقتين تم تطبيق القانون التالى:

TOTAL WAIT = I/O WAIT TIME + CPU TIME

تقسيم الذاكرة فى 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 فى الشكل (٦/١٠).



شكل (٦/١٠)

وهذا التقسيم يفضى إلى نوعين من أساليب تقسيم الذاكرة هما:

أولا : التقسيم الثابت أو التقسيم الإستاتيكى:

وهو تقسيم حيز الذاكرة الى قطاعات ثابتة محددة على أن يخصص لكل برنامج القطاع المناسب له ، ونلمح هذا التقسيم فى نظام التشغيل :

[OS / MFT] Operating System Multi Fixed Tasks

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

أ ــ وجود قطاع خال من العمل لعدم وجود برامج تنتظر التنفيذ.

ب ــ وجود برامج مؤجلة لعدم وجود القطاع ذى الحيز المناسب.

ج ــ تفتيت الحيز المتاح الى ثقوب صغيرة Holes لا قيمة فعلية لها من حيث توافق الحيز مع متطلبات البرامج المنتظرة.

ويوضح الجدول التالى مضمون تفتيت الذاكرة ، الذى يوضح أن أكثر من

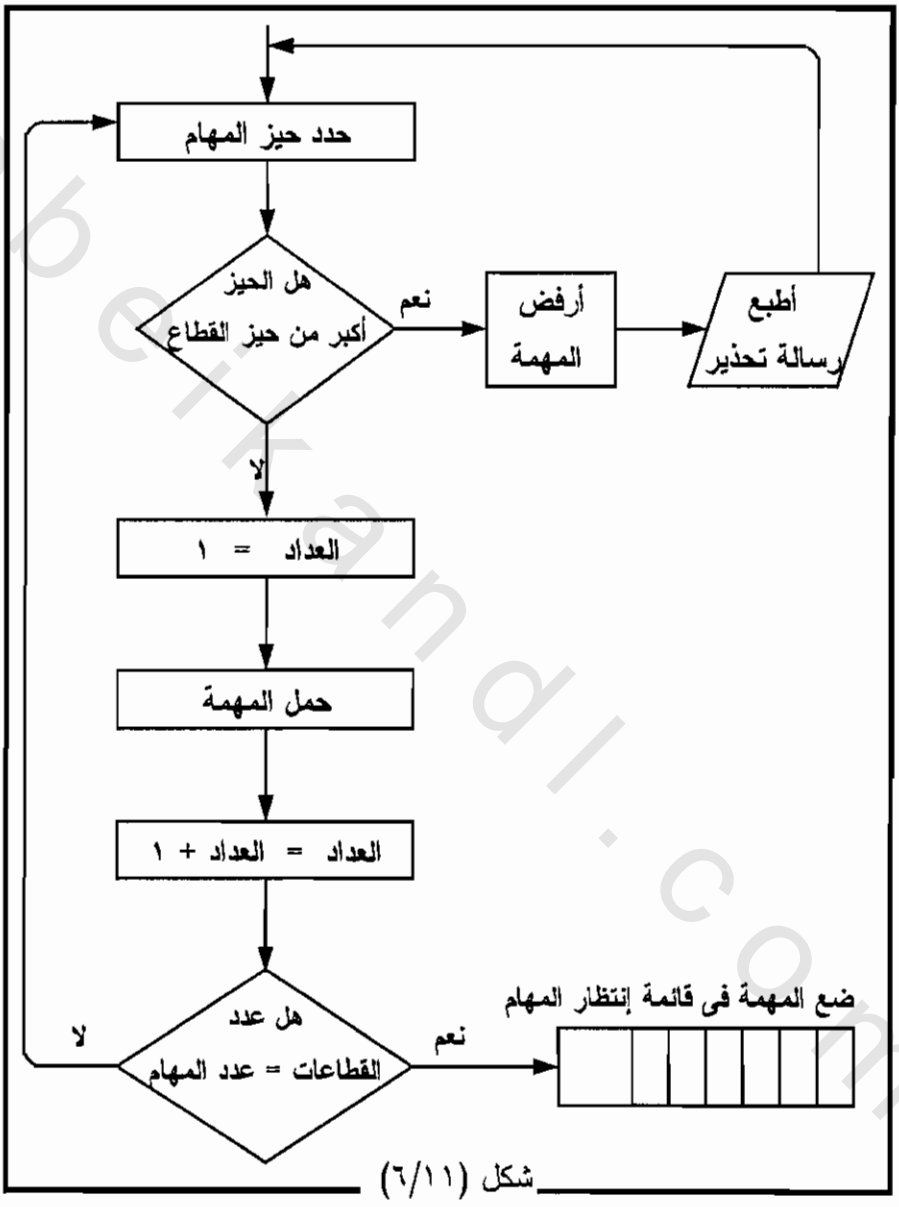
رقم القطاع	الحيز المتاح بوحدة كيلو بايت		
	حيز القطاع	حيز المهمة	حيز مفقود
١	٨	١	٧
٢	٣٢	٩	٢٣
٣	٣٢	٩	٢٣
٤	١٢٠	٣٢	٨٨
٥	٥٢٠	١٢١	٣٩٩

٧٥% من جملة حيز مساحة الذاكرة الأساسية غير مستغل.

سياسات أستخدام الحيز المتصل فى إطار تعددية البرمجة:

ثبت أن أستخدام تعددية البرمجة ينتج عنه تفتيت يحتاج الأستفادة بها إجراء عملية جمع هذه الثقوب على هيئة حيز واحد بإجراء عملية التضاغط Compaction وهى عملية لها مشاكلها وآثارها السلبية على كفاءة المنظومة ، منها أن تحديد الحيز مسبقا يعنى تحديد درجة التعددية البرمجية المتاحة ، وهذا لا ينفى 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 فى إطار سياسة الحيز المتصل مع تعددية البرمجة وفــق المنطق الموضح فى الشكل (٦/١١).



شكل (٦/١١)

ويتم تنسيق المهام وفق سياسات محددة هى:

أ ــ أول حيز يمكنه قبول المهمة FIRST FIT .

ب ــ أفضل حيز يناسب المهمة BEST FIT .

ج ــ أى حيز متاح 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 سيان 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 يعمد 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 الافضل؟ تتوقف الاجابة على عدة أعتبارات:

أ ــ الرغبة فى تحديد الحيز المناسب.

ب ــ 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 التى تعنى WORST FIT .

مثـــال:

يوضح الجدول التالى قائمة المهام وزمن تشغيلها على المنظومة كما يوضح الشكل الحيز المتاح فى الذاكرة الموزعة توزيعاً ثابتاً والمطلوب توزيع المهام وفـــق 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
	كيلو بايت	دقيقة

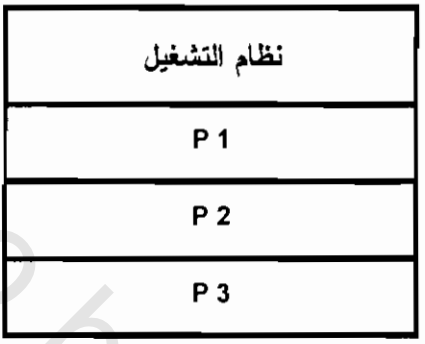
الحـــل:

١ ــ التوزيع الإستاتيكى لقطاعات الذاكرة.

الذاكـــــرة

نظام التشغيـــل
القطاع أ ٦٠٠ كيلو بايت
ب ١٠٠
ج ٥٦٠

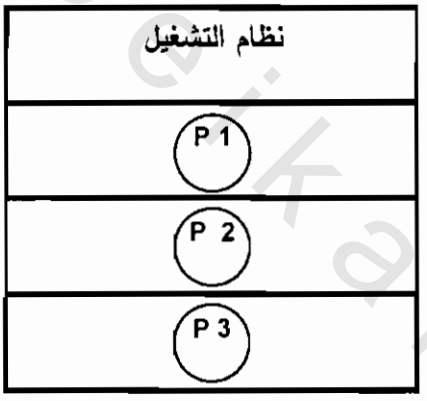
٢ ــ تم شحن المهام P1 , P2 وفق سياسة BEST FIT أما المهمة P3 فقد شحنت وفق سياسة FIRST FIT ولذلك أحيطت بدائرة بما معناه أن هناك حيز غير مستغل.



٣ ــ بعد خمس دقائق :

أ ــ خرجت المهمة P2 حيث تم تنفيذها.

ب ــ دخلت مكانها فى القطاع ب المهمة P4 وفق سياسة WORST FIT ولذلك أحيطت بدائرة.

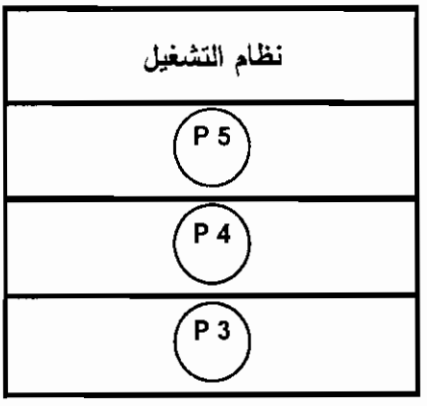


٤ ــ بعد خمس دقائق أخرى :

أ ــ خرجت المهمة P1 لإنتهاء تنفيذها وأحل محلها P5 حيث هى المهمة الباقية والمتمشية مع الحيز.

ب ــ تسبب كل هذه المهام تفتيت فى القطاعات.

ج ــ إدخال P5 تم وفق سياسة FIRST FIT.



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

ثانيا: التوزيع الديناميكى للذاكرة :

فى أسلوب الحيز المتصل يقصد بالتقسيم الديناميكى للذاكرة الأساسية إجراء عملية تحديد وإنشاء القطاعات أثناء تنفيذ المهام حتى يتلائم حيز الذاكرة مـــع حـيز المهمة ، ويعتمد التوزيع الديناميكى للذاكرة احيانـــا علـــى طبيعـــة لغـــة البرمجـــة المستخدمة

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

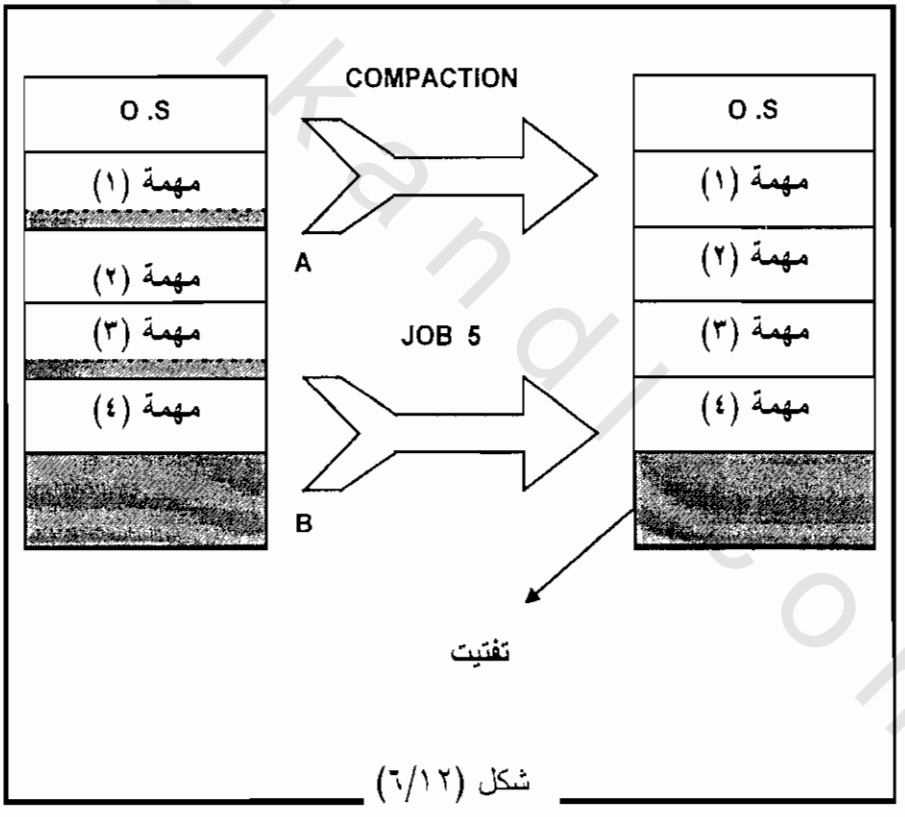
ويتطلب التوزيع الديناميكى للذاكرة وجود بيانات كافية على هيئـــة جـــداول توصف المناطق المخصصة لمهام والمناطق الحرة المتاح أستخدامها.

ولعل أبرز أنظمة التشغيل التى تلبى حاجات التوزيع الديناميكى هـــو نظـــام تشغيل IBM المعروف بأسم OS/ MTV .

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

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

فيما يوضحه الشكل (٦/١٢):

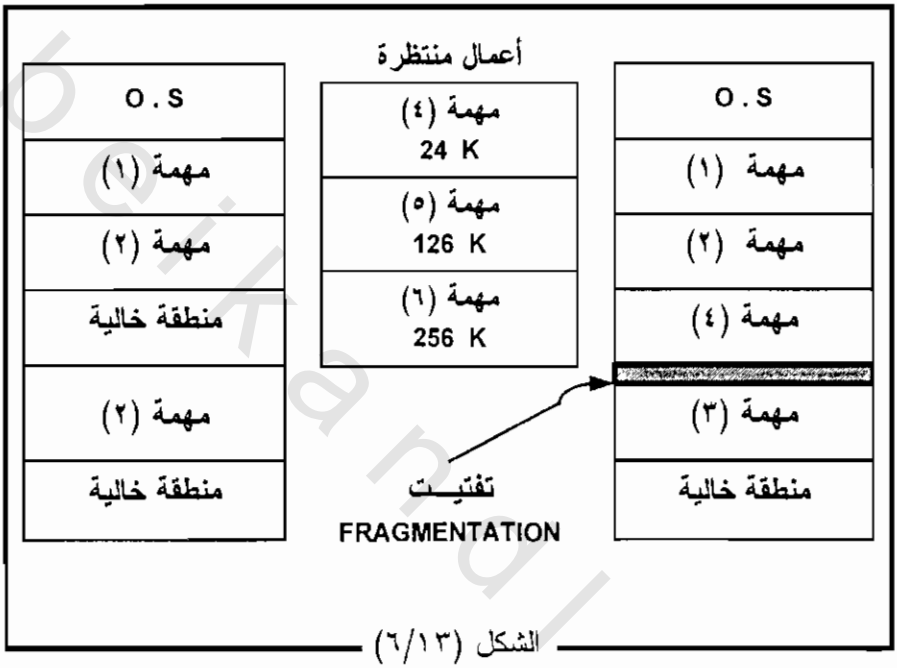


شكل (٦/١٢)

وتطلق المسميات التالية على التوزيع الديناميكى للذاكرة:

* Dynamic Storage Allocation.
* Controlled Storage.

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



الشكل (٦/١٣)

ودمج الثقوب عملية بسيطة من وجهة النظر المنطقية لكنها لاتضمن نفس أداء البرامج فى المواقع الجديدة لعدة أسباب أهمها حساسية بعض المواقع مثل:

أ ــ مسجلات القاعدة.

ب ــ المواقع المحددة للعنونة.

ج ــ قوائم المؤثرات.

د ــ تركيب البيانات.

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

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

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

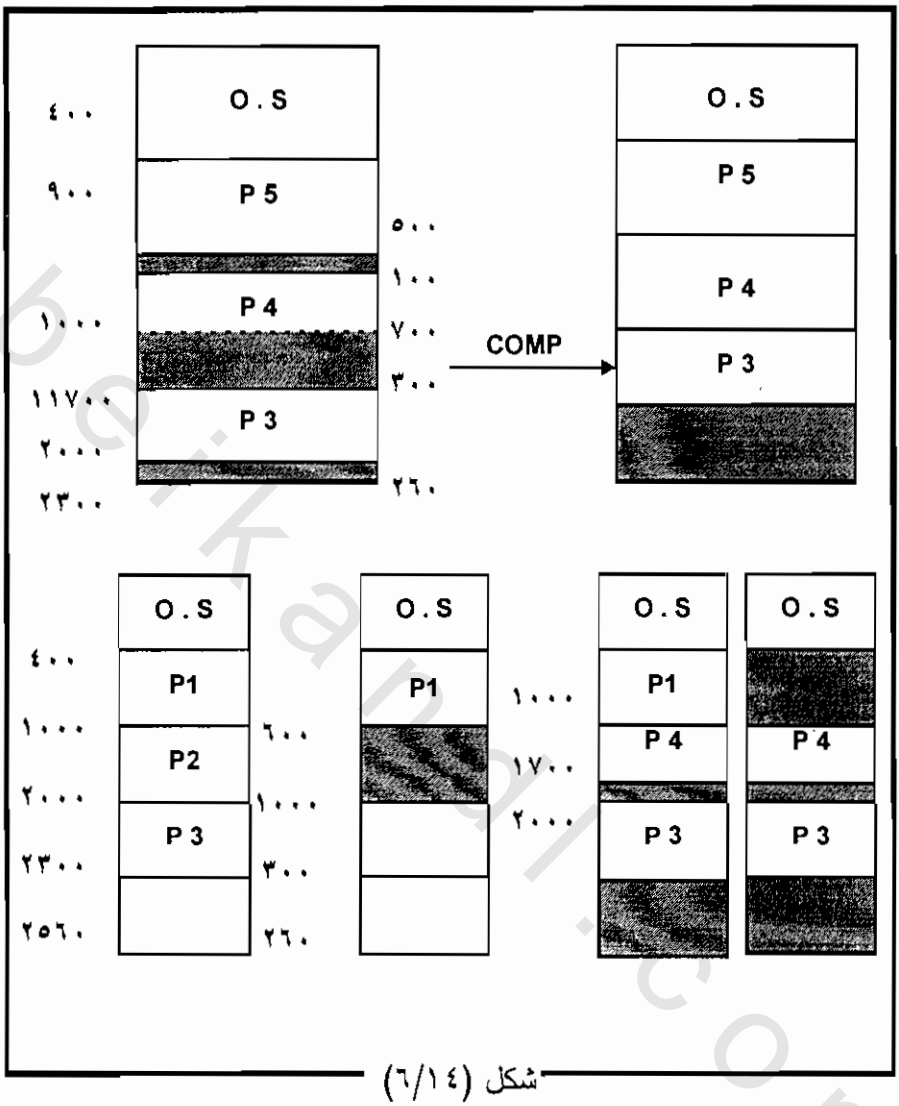
ج ــ تتطلب المواقع عنونة جديدة مما يؤثر على كفاءة الأداء.

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 COMPACTION محفوفة بالمخاطر وتحتم إعادة تشغيل هذه البرامج مرة أخرى.

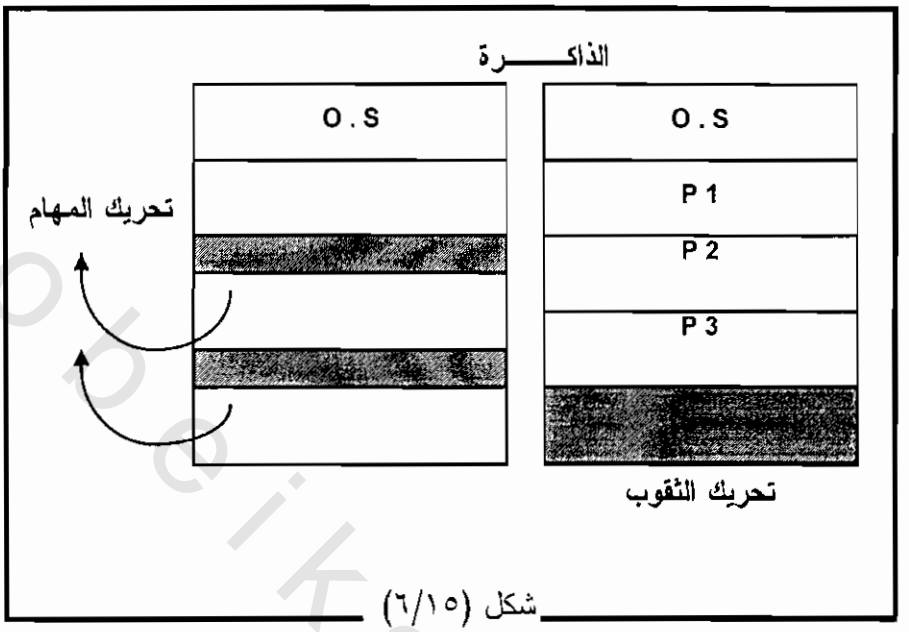
معنى هذا أن التضاغط ليس متاحاً دائماً ويوضح الشكل (٦/١٤) مثال عن ما عرضنا من مصاعب:

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



شكل (٦/١٤)

وأبسط صور التضاغط هى تحريك البرامج والمهام إلى الطـــرف الأعلـــى للذاكرة بينما تتحرك الثقوب إلى الناحية الأخرى من الذاكرة كما فى الشكل (٦/١٥):



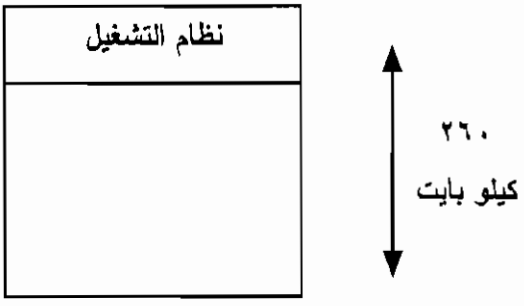
شكل (٦/١٥)

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

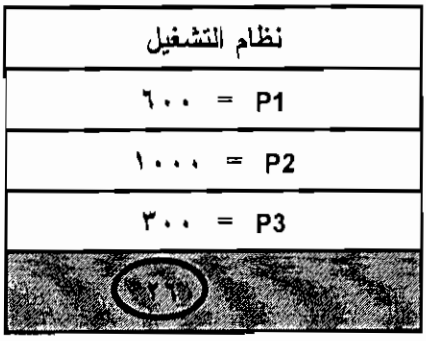
مثـــال:

عالج المثال السابق باستخدام إمكانية إجراء تضاغط .

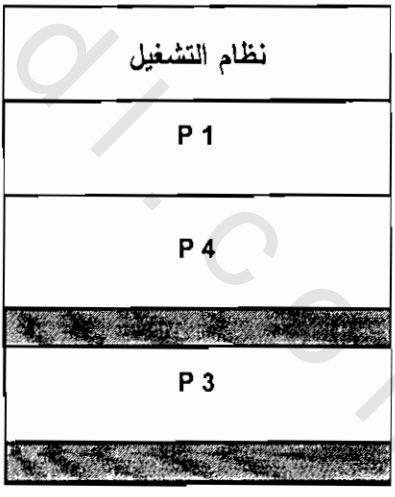
الحل:

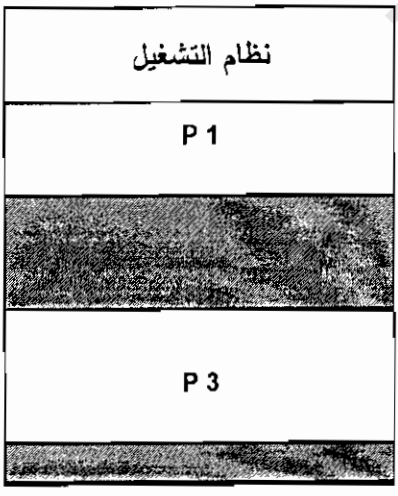


١ ــ حدد الحيز الذى يصلح لثلاث مهام هى P1 , P2 , P3 على النحو الموضح فى الشكل ونلحظ بقاء حيز غير مستخدم سعة ٢٦٠ كيلو بايت لا يصلح لأى مـــن [P4] أو [P5] .

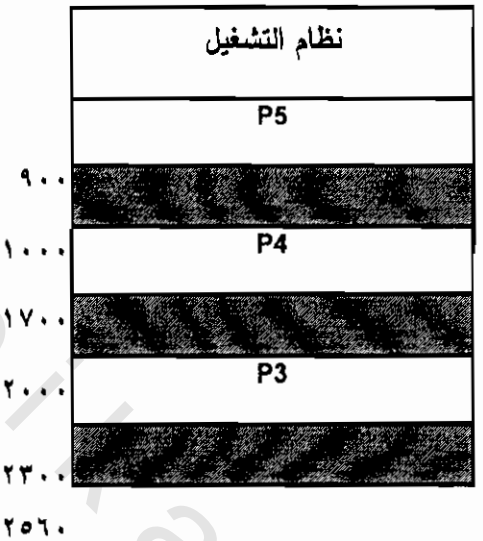


٢ ــ بعد خمس دقائق تنهى المهمة [P2] ويصبح شكل التوزيع داخل الذاكرة على النحو:

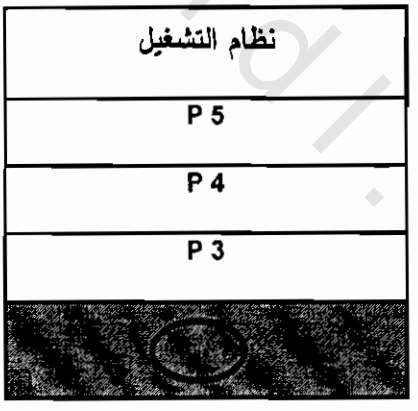




٣ ــ بعد فترة تنهى [P1] تخلى من الذاكرة وتستقبل المهمة [P5] .



٤ ــ بإجراء التضاغط :



تقسيم القطاع الواحد:

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

ويؤدى تقسيم القطاع الواحد إلى بعض المزايا هى:

أ ــ تعددية البرمجة على مستوى القطاع.

ب ــ لا يحتاج هذا الأسلوب إلى معدات خاصة.

العيوب :

أ ــ يعانى من نفس مشكلة تفتيت الذاكرة.

ب ــ حتى فى حالة عدم وجود فتات فان الحيز الخال لا يمكنه تشكيل قطاع مستقل.

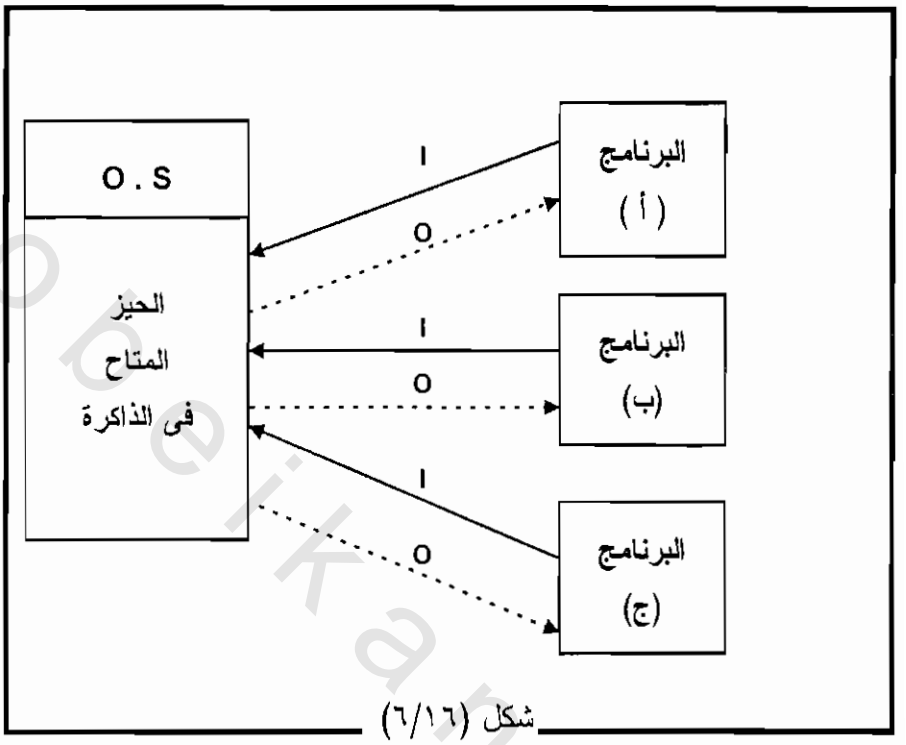
ج ــ يحتاج نظام تشغيل معقد.

الحيز المتصل والتبديل SWAPPING :

على مستوى المهام:

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

والإجابة فى أقتراح 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):



شكل (٦/١٦)

١ ــ أعتبارات التبديـــــل:

أ ــ يستمر تشغيل البرنامج ويخلى من الذاكرة الأساسية فى الحالات التالية:

(١) إذا طلب عملية إدخال / إخراج.

(٢) إذا تجاوز الوقت المخصص له دون I/O.

(٣) إذا حدثت ظروف قاهرة أثناء التشغيل.

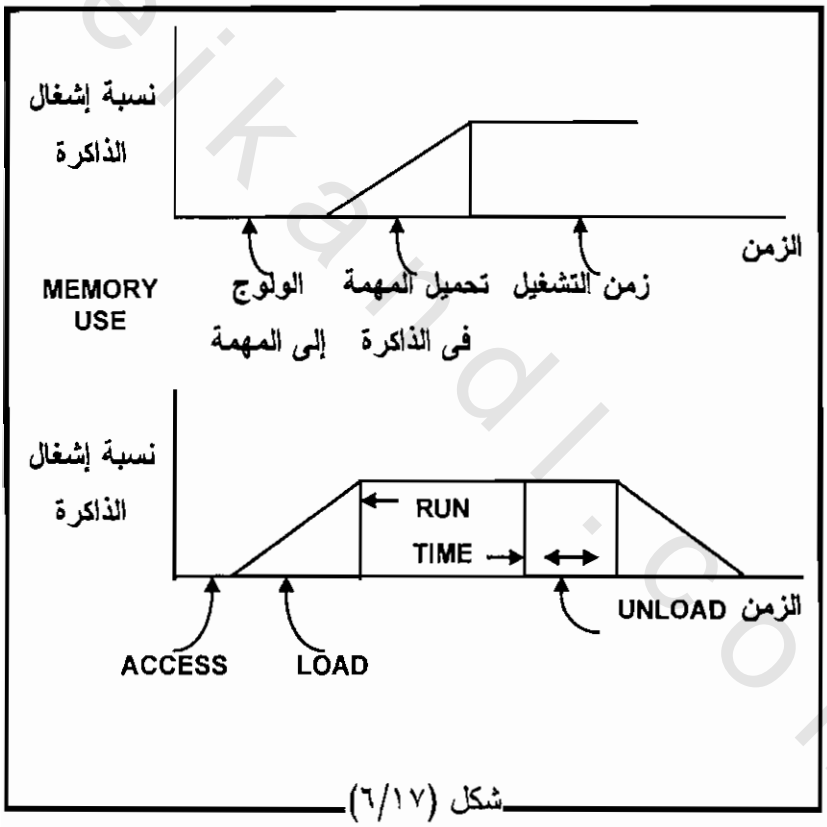
ب ــ تطبع نسخة من البرنامج على وسائط التخزين الثانوية.

ج ــ يماثل أسلوب التبديل أسلوب المشاركة فى الوقت على مستوى وحـــدة التشغيل المركزية.

د ــ تطور هذا الأسلوب أفضى إلى إمكانية تواجد أكثر من برنامج فى حيز منطقة التبديل مما تطلب إدخال تعديلات ضخمة على نظم التشغيل.

٢ ــ المنحنى الزمنى للتبديل:

يوضحه الشكل (٦/١٧) ومنه يتضح أن التبديل يتطلب فترة زمنية للوصـــول إلى المهمة ، بعدها تتم قراءتها وتحميلها فى الذاكرة الأساسية ممـــا يســتغرق زمناً آخر يضاف إلى إجمالى الوقت.



شكل (٦/١٧)

بعدها يأتى وقت التشغيل ، ويضاف وقت آخر لإزاحة المهمة إلـــى وســائط التخزين.

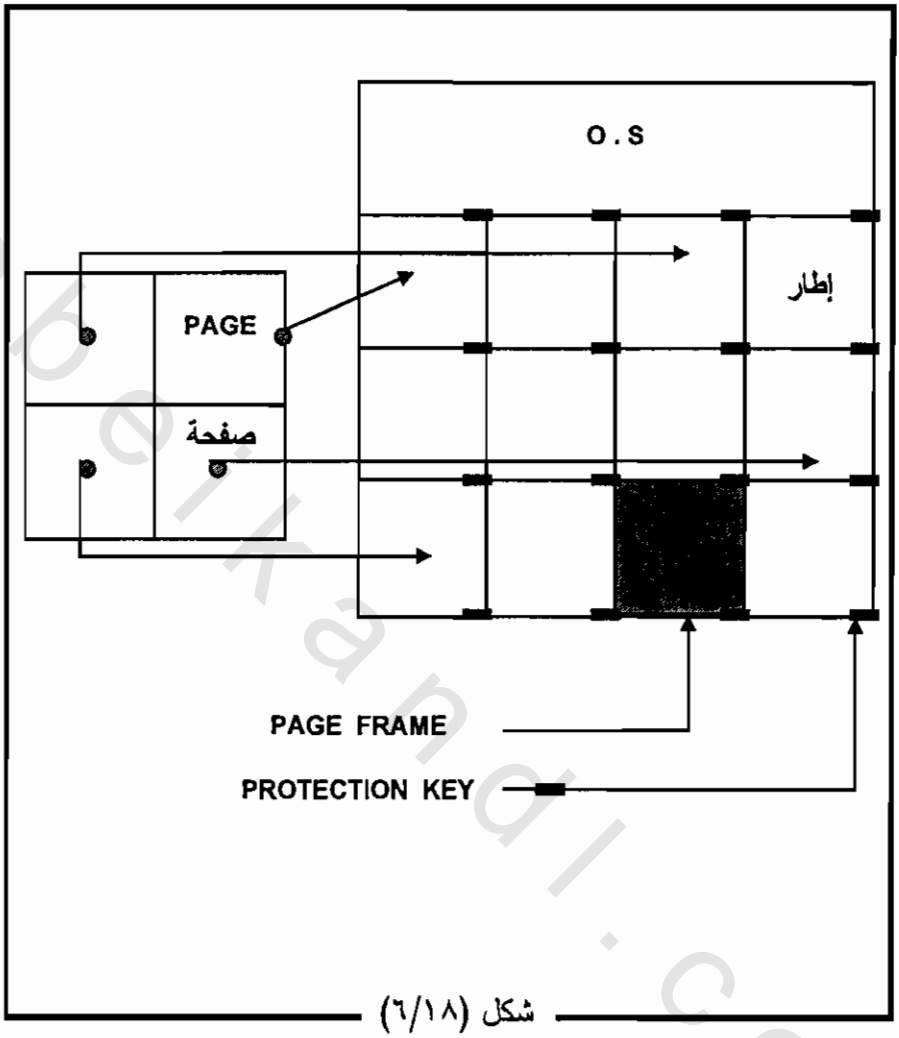
ثانيا: أساليب الحيز غير المتصل:

التصفح PAGING – فكرة مبسطة:

يعتبر أسلوب التصفح حلاً جيداً لمشكلة تفتيت الذاكــرة ومشــاكل اللاحقــة للتضاغط COMPLICATION ، وفى هذا الأسلوب تقسم الذاكرة إلى عدد كبــير مــن حيز ثابت صغير نسبياً يتراوح بين ٠،٥ كيلو بايت وحتى أربعة كيلو بايت ويســمى 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 أستخدام عدد كبير من مســجلات الوقايــة ، وقــد تكــون المسجلات جزءاً من الكيان الآلى أو أجزاء من الذاكرة وهذا يتوقــف علــى ســعة الذاكرة الأساسية.

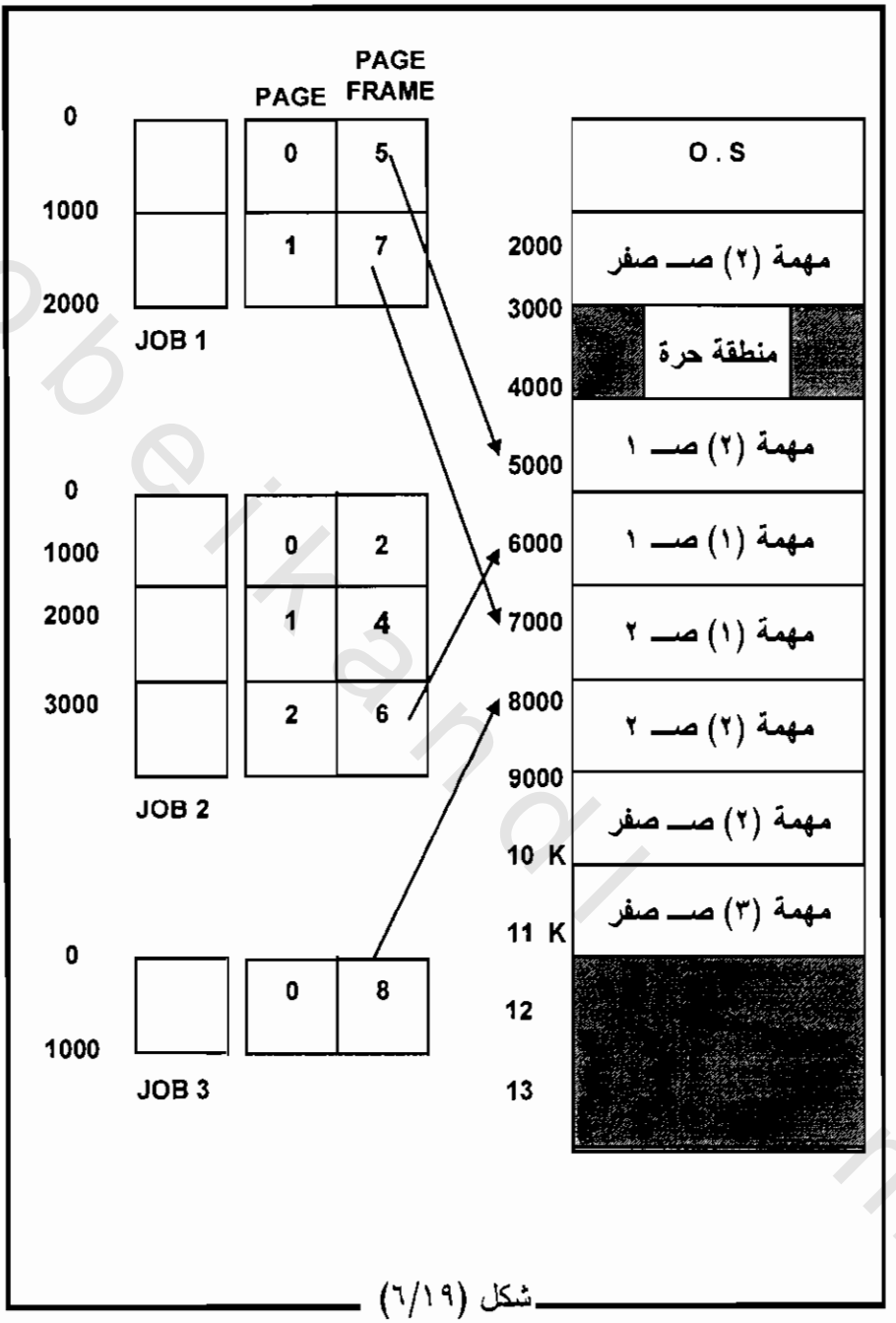
ويوضح الشكل (٦/١٨) المفهوم الأساسى للتصفح والذى يتطلــب عنونــة الصفحات وعنونة أطر الصفحات المناظرة وخلق جداول التطابق بين هذه وتلــك ، ورغم التكلفة الإضافية للنظام إلا أنه أسلوب أثبت كفاءة جيدة فى معالجة المشــاكل التى سبق لنا التعرض لها فى إدارة الذاكرة وأبرزها مشكلة التفتيت وإشغــال مديــر الذاكرة فى متابعة مواقع خالية.



شكل (٦/١٨)

مثـــال:

المطلوب توزيع المهام الثلاث الموضحة بـــالشكل (٦/١٩) علـــى الذاكـــرة الأساسية وفق أسلوب التصفح ، مع أستخدام القطاعات:


PAGE
PAGE FRAME
0
1000
2000
JOB 1
0 5
1 7
0
1000
2000
3000
JOB 2
0 2
1 4
2 6
0
1000
JOB 3
0 8
O . S
2000
مهمة (٢) صـ صفر
3000
منطقة حرة
4000
مهمة (٢) صـ ١
5000
مهمة (١) صـ ١
6000
مهمة (١) صـ ٢
7000
مهمة (٢) صـ ٢
8000
مهمة (٢) صـ صفر
9000
مهمة (٣) صـ صفر
10 K
11 K
12
13

شكل (٦/١٩)

ويوضح 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.

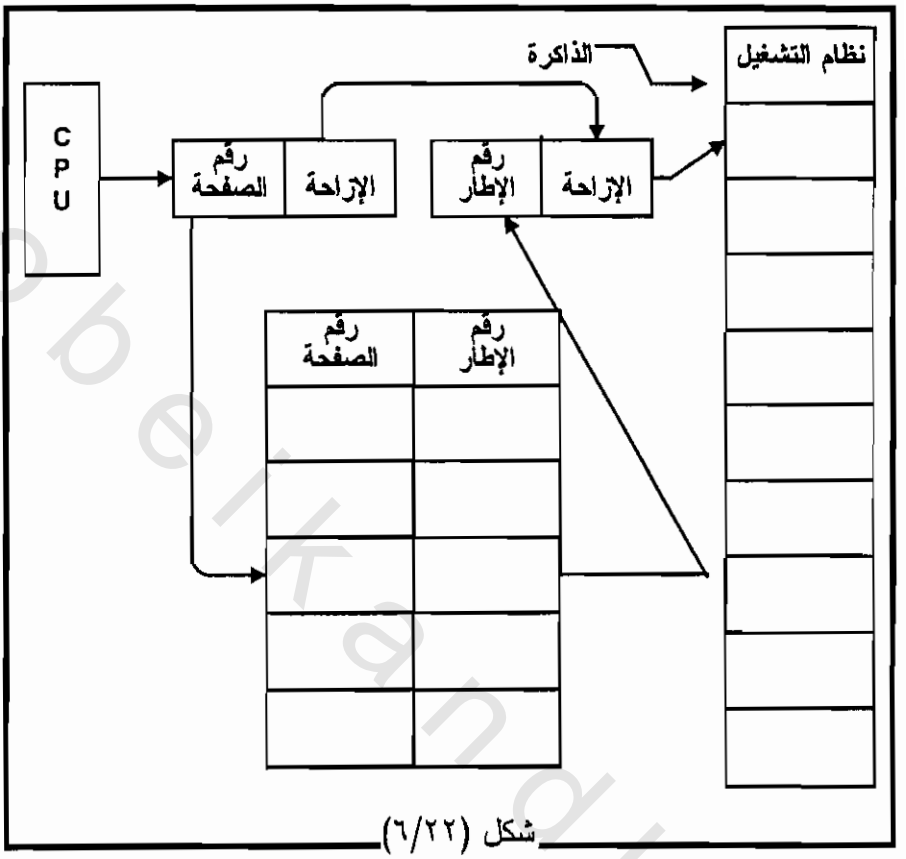
فإذا كان هناك عمل أو مهمة منتظرة ولتكن المهمة (٤) وتحتاج إلى ٥ كيلو بايت فإنه يمكن توزيع صفحاتها بين العنوان ٣٠٠٠ والعنوان ١٠٠٠٠/١١٠٠٠/١٢٠٠٠/١٣٠٠٠.

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

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

الدعم الآلى للتصفح:

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



شكل (٦/٢٢)

وبذلك يتخلق العنوان الفيزيائى ــ الواقعى ــ فى الذاكرة به لحيـــن وضـــع قيمه فى مسجل العناوين MAR ليستدعى الايعاز كما سبق وعرضنا فى الباب الثانى ، وبذا تناظر عملية التصفح وادارتها عملية إعادة تسكين المهام فى الذاكرة.

*