جَـامعة
المَـنارة
MANARA UNIVERSITY

OPERATING SYSTEM

Lecture Notes

Dr. Professor, J.M. Khalifeh

قسم المعلوماتية

الوحدة السادسة

النسخة العربية

Unit-6

Main Memory Management

ملخص

الغرض الرئيسي من نظام الحاسوب هو تنفيذ البرامج. وهذه البرامج يجب أن تكون، مع البيانات التي تصل إليها، موجودةً جزئيًا على الأقل في الذاكرة الرئيسية أثناء التنفيذ. ويعتبر الوصول إلى الذاكرة وإدارتها جزءاً بالغ الأهمية من عمليات الحاسوب الحديثة. يجب جلب كل تعليمة من الذاكرة قبل تنفيذها، وتتضمن معظم التعليمات استرجاع البيانات من الذاكرة أو تخزينها فيها أو كليهما.

يزيد ظهور أنظمة التشغيل متعددة المهام من تعقيد إدارة الذاكرة، فمع تبادل العمليات داخل وخارج وحدة المعالجة المركزية، يجب تبادل أكوادها وبياناتها داخل وخارج الذاكرة، كل ذلك بسرعات عالية ودون تداخل مع أي عمليات أخرى.

تحتفظ أنظمة الحاسوب الحديثة بالعديد من العمليات في الذاكرة أثناء تنفيذ النظام. توجد العديد من خوارزميات وآليات إدارة الذاكرة، تعكس مناهج مختلفة، وتختلف فعالية كل خوارزمية باختلاف الحالة. يعتمد اختيار نظام إدارة الذاكرة لنظام ما على عوامل عديدة، وعلى الأخص تصميم المكونات الفيزيائية للنظام. تتطلب معظم الخوارزميات دعمًا عتاديًا. ويزداد الأمر تعقيدا بازدياد المتطلبات المتعلقة بالذاكرة المشتركة، والذاكرة الافتراضية، وبتصنيف الذاكرة إلى للقراءة فقط مقابل للقراءة والكتابة، ومفاهيم مثل تقسيم النسخ عند الكتابة.

أهداف الوحدة

- شرح الفرق بين العنوان المنطقي والعنوان المادي، ودور وحدة إدارة الذاكرة MMU في ترجمة العناوين.
- تطبيق استراتيجيات تخصيص الذاكرة
- بيان الفرق بين التجزئة الداخلية والخارجية وبيان الفرق بين استراتيجيات "الأول، والأفضل، والأسوأ" عند تطبيق التخصيص المتجاور.
- شرح مفهوم التصفيح Paging.
- معرفة كيف يتم ترجمة العناوين المنطقية إلى عناوين مادية في نظام التصفيح paging

مدخل

تُعدّ الذاكرة كما رأينا جوهر عمل أنظمة الحاسوب الحديثة، حيث تتكون من مصفوفة كبيرة من البايتات، لكل منها عنوانها الخاص. تستدعي وحدة المعالجة المركزية CPU التعليمات من الذاكرة وفقًا لقيمة تحدد بناء على عداد البرنامج. قد تُسبب هذه التعليمات تحميلًا لعناوين إضافية.

على سبيل المثال، تبدأ دورة تنفيذ التعليمات النموذجية بجلب تعليمة من الذاكرة ويتم البدء بتنفيذها، وقد يؤدي ذلك إلى جلب مُعاملات من الذاكرة. بعد تنفيذ التعليمات على المُعاملات، قد تُخزّن النتائج في الذاكرة. لا ترى وحدة الذاكرة سوى سيل من العناوين؛ فهي لا تعرف كيفية توليدها (بواسطة عداد التعليمات، أو الفهرسة، أو العناوين الحرفية، وما إلى ذلك) أو الغرض منها (تعليمات أم بيانات). بناءً على ذلك، سنتجاهل هنا كيفية توليد البرنامج لعنوان ذاكرة. وسنهتم فقط بتسلسل عناوين الذاكرة التي يُولّدها البرنامج قيد التشغيل. سنبدأ نقاشنا بتغطية عدة قضايا ذات صلة بإدارة الذاكرة: المكونات الأساسية، وربط عناوين الذاكرة الافتراضية بالعناوين المادية الفعلية، والتمييز بين العناوين المنطقية والمادية. نختتم هذا القسم بمناقشة الربط الديناميكي والمكتبات المشتركة.

١.١ المكونات الأساسية

تجدر الإشارة إلى أنه من منظور الذاكرة، فإن جميع عمليات الوصول إليها متكافئة. ولا تعرف مكونات الذاكرة ما هو الغرض من استخدام أي جزء معين منها، ولا تهتم به. ويمكن قول هذا تقريبًا على نظام التشغيل أيضًا، وإن لم يكن بالكامل. كما لا يمكن لوحدة المعالجة المركزية الوصول إلا إلى مسجلاتها والذاكرة الرئيسية. على سبيل المثال، لا يمكنها الوصول مباشرةً إلى القرص الصلب، لذلك يجب نقل أي بيانات مخزنة هناك أولًا إلى الذاكرة الرئيسية قبل أن تتمكن من العمل عليها. (تتواصل برامج تشغيل الأجهزة مع هذه الأجهزة عبر المقاطعات وعمليات الوصول إلى "الذاكرة"، مرسلةً تعليمات قصيرة. فعلى سبيل المثال، لنقل البيانات من القرص الصلب إلى موقع محدد في الذاكرة الرئيسية. تراقب وحدة تحكم القرص الناقل بحثًا عن مثل هذه التعليمات، لتقوم بالتفاعل معها وتنقل البيانات، ثم تُعلم وحدة المعالجة المركزية بوجود البيانات المطلوبة من خلال مقاطعة أخرى. لكن وحدة المعالجة المركزية لا تحصل أبدًا على وصول مباشر إلى القرص.)

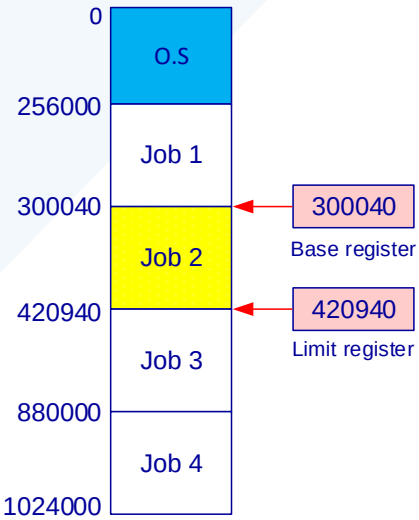


Figure 1 - A base and a limit register define a logical addresses space

عمليات الوصول إلى المسجلات registers سريعة جدًا، وقد تتمكن وحدة المعالجة المركزية من تنفيذ أكثر من تعليمة واحدة لكل دقة ساعة. بينما عمليات الوصول إلى الذاكرة الرئيسية بطيئة نسبيًا، وقد تستغرق عدة دقات ساعة

لإكمالها. هذا الأمر سيتطلب انتظارًا طويلًا من وحدة المعالجة المركزية CPU لولا وجود ذاكرة تخزين مؤقتة سريعة cacheمدمجة في معظم وحدات المعالجة المركزية الحديثة. الفكرة الأساسية لذاكرة التخزين المؤقت هي نقل أجزاء من الذاكرة دفعةً واحدة من الذاكرة الرئيسية إلى ذاكرة التخزين المؤقت، ثم الوصول إلى مواقع الذاكرة الفردية واحدًا تلو الآخر من ذاكرة التخزين المؤقت.

ومن أجل حماية المستخدمين من إمكانية الدخول إلى المواقع التي "تنتمي" إلى العملية المعنية تحديدًا. يتم استخدام سجل base register أساسي وسجل حد limit register لكل عملية، كما هو موضح في الشكلين 1 و2. يتم التحقق من كل وصول إلى الذاكرة تقوم به عملية مستخدم عبر هذين السجلين، وإذا جرت محاولة الوصول إلى الذاكرة خارج النطاق المسموح به، فسيتم توليد خطأ. من الواضح أن نظام التشغيل يتمتع بالوصول إلى جميع مواقع الذاكرة الموجودة، لأن هذا ضروري لتبادل بيانات وشفرات المستخدمين داخل وخارج الذاكرة. يجب أن يكون واضحًا أيضًا أن تغيير محتويات السجلين الأساسي والحدي نشاط ذو امتيازاتprivileged ، أي أنه مسموح به فقط لنواة نظام التشغيل.

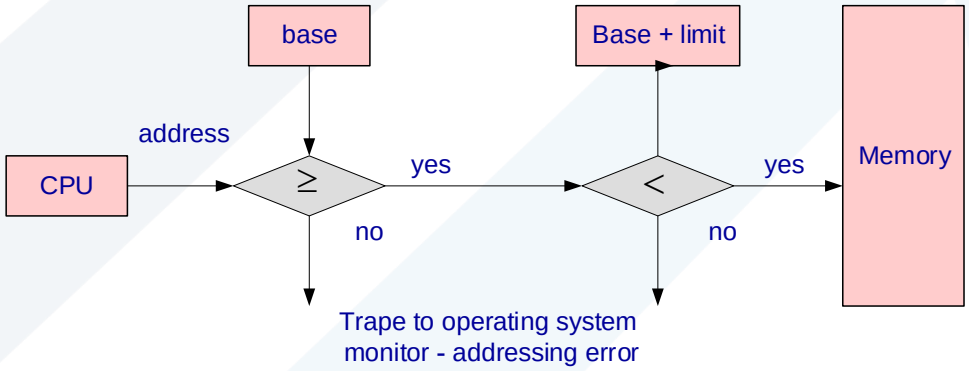
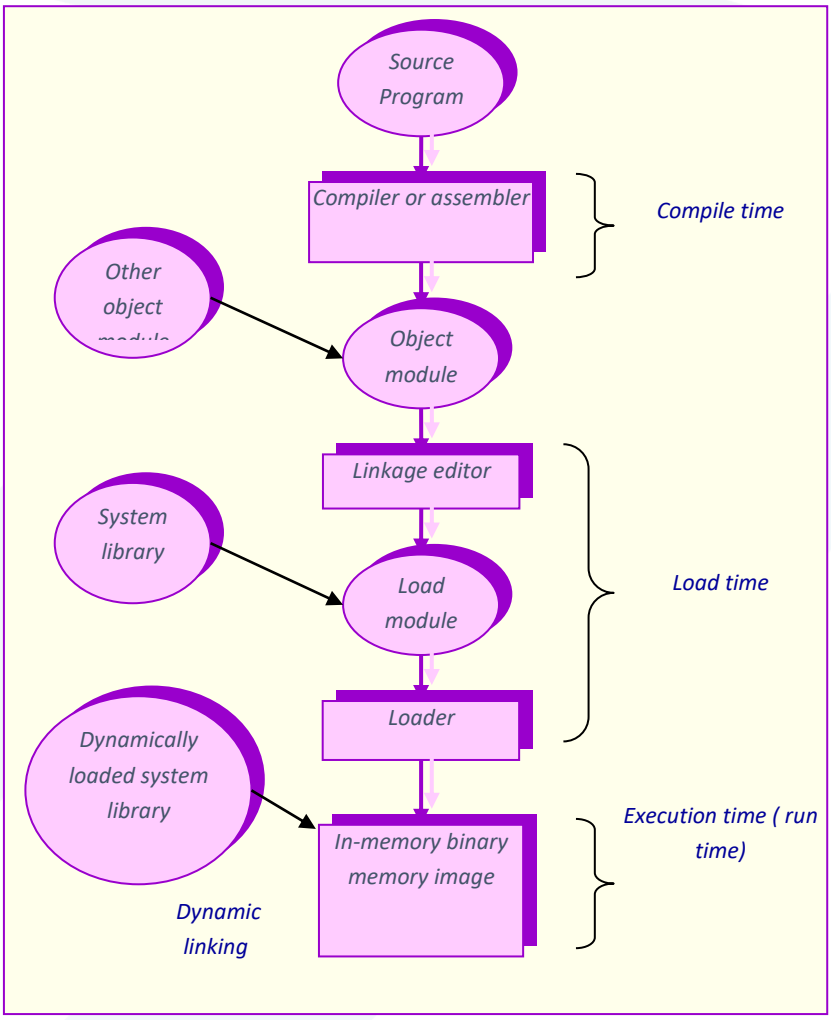


Figure 2 - Hardware address protection with base and limit registers

٢.١ ربط العناوين Address Binding

عادةً ما تُشير برامج المستخدم إلى عناوين الذاكرة بأسماء رمزية مثل "i" و "count" و "averageTemperature". ولا بد قبل بدء التنفيذ من ربط هذه الأسماء الرمزية بعناوين mapped or bound الذاكرة الفعلية، والتي تتم عادةً على في مراحل:



يوضح الشكل 3 المراحل المختلفة لعمليات الربط والوحدات المشاركة في كل مرحلة:

الشكل : سياق ربط العنوان

- **زمن الترجمة Compile Time**: إذا عُرف مكان تواجد البرنامج في الذاكرة الفعلية وقت الترجمة، فيمكن للمترجم compiler إنشاء شيفرة مُطلقة absolute code تحتوي على عناوين فيزيائية فعلية. أما إذا تغير عنوان التحميل لاحقًا، فيجب إعادة ترجمة البرنامج.
- **زمن التحميل Load Time**: إذا لم يكن موقع تحميل البرنامج معروفًا وقت الترجمة، فيجب على المحول إنشاء شيفرة قابلة لإعادة التخصيص *relocatable code* ، والتي تُشير إلى عناوين نسبة إلى بداية البرنامج. إذا تغير عنوان البداية، فيجب إعادة تحميل البرنامج وليس إعادة تحويله.

زمن التنفيذ Execution Time: إذا كان من الممكن نقل البرنامج في الذاكرة أثناء تنفيذه، فيجب تأخير الربط حتى وقت التنفيذ. يتطلب هذا أجهزة خاصة، وهي الطريقة المُطبقة في معظم أنظمة التشغيل الحديثة.

١,٣ فضاء العنوان المنطقي مقابل فضاء العنوان المادي Logical Versus Physical Address Space

العنوان الذي تُولّده وحدة المعالجة المركزية CPU هو عنوان منطقي Logical، بينما العنوان الذي تراه أجهزة الذاكرة فعليًا هو عنوان مادي physical. ومن الضروري أن تكون العناوين المرتبطة وقت التحويل أو وقت التحميل لها عناوين منطقية ومادية متطابقة معها. أما العناوين التي تُنشأ وقت التنفيذ، تكون لها عناوين منطقية ومادية مختلفة. يُعرف العنوان المنطقي أيضًا باسم العنوان الافتراضي virtual address، ويدل المصطلحان في نصنا على نفس المعنى أينما وردا.

تُشكّل مجموعة جميع العناوين المنطقية التي يستخدمها البرنامج فضاء العنوان المنطقي logical address space، وتُشكّل مجموعة جميع العناوين المادية المقابلة فضاء العنوان المادية physical address space.

تُدير وحدة إدارة الذاكرة memory–management unit, MMU (الشكل 4) عملية تعيين العناوين المنطقية إلى المادية وقت التشغيل. ويمكن أن تتخذ وحدة إدارة الذاكرة MMU أشكالًا متعددة. أحد أبسطها هو تعديل لبنية مسجل الأساس base–register الذي مر معنا سابقا.

يُطلق في هذا التعديل على المسجل الأساس اسم مسجل إعادة التموضع relocation register، حيث تُضاف قيمته إلى كل طلب ذاكرة على مستوى الأجهزة. لابد من ملاحظة أن برامج المستخدم لا ترى عناوين مادية أبدًا. تعمل برامج المستخدم بالكامل في مساحة العناوين المنطقية، وتُجرى أي مراجعة أو عمليات على الذاكرة باستخدام عناوين منطقية بحتة. ولا يُولَّد عنوان الذاكرة المادية إلا عند إرسال العنوان إلى شرائح الذاكرة المادية.

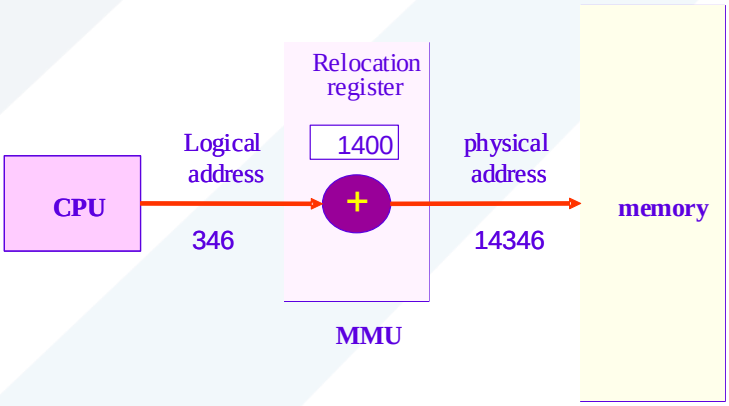


Figure 4 – Dynamic relocation using a relocation register

٤.١ التحميل الديناميكي Dynamic Loading

بدلاً من تحميل برنامج كامل في الذاكرة دفعةً واحدة، يستدعي التحميل الديناميكي كل روتين عند استدعائه. ميزته هي عدم الحاجة لتحميل الروتينات غير المستخدمة، مما يُقلل من إجمالي استخدام الذاكرة ويُسرّع من أوقات بدء تشغيل البرنامج. أما الجانب السلبي فهو التعقيد الإضافي والتكلفة الإضافية للتحقق مما إذا كان الروتين مُحمّلاً في كل مرة يُستدعى فيها، ثم تحميله إذا لم يكن مُحمّلاً بالفعل.

٥.١ الربط الديناميكي والمكتبات المشتركة Dynamic Linking and Shared Libraries

عند الاعتماد على الربط الثابت static linking ، لابد من تضمين الكود القابل للتنفيذ نسخة من وحدات المكتبة المعنية بتنفيذ بعض الأجزاء بشكل كامل في الوحدات القابلة للتنفيذ من البرنامج، مما يُهدر مساحة القرص ويقلل

من كفاءة استخدام الذاكرة الرئيسية، لأن كل برنامج يتضمن روتيناً مُعيناً من المكتبة يجب أن يكون لديه نسخة خاصة به من ذلك الروتين مُرتبطة بكوده القابل للتنفيذ.

مع الربط الديناميكي dynamic linking ، يتم ربط جزء صغير فقط من وحدات المكتبة المعنية بالوحدة القابلة للتنفيذ، يحتوي هذا الجزء على مراجع لوحدة المكتبة الفعلية المُرتبطة وقت التشغيل.

توفر هذه الطريقة مساحة على القرص، لأن روتينات المكتبة لا تحتاج إلى أن تُدرج بالكامل في الوحدات النمطية القابلة للتنفيذ، بل في الأجزاء الأولية فقط. ونشير أيضًا أنه إذا كان قسم التعليمات البرمجية في روتينات المكتبة قابلًا لإعادة الدخول reentrant ، أي أنه لا يعدل التعليمات البرمجية أثناء تشغيله، مما يجعله آمنًا لإعادة الدخول إليه ، فيمكن توفير الذاكرة الرئيسية عن طريق تحميل نسخة واحدة فقط من الروتينات المرتبطة ديناميكيًا في الذاكرة ومشاركة التعليمات البرمجية بين جميع العمليات التي تستخدمها في نفس الوقت. (ستكون لكل عملية نسختها الخاصة من قسم البيانات في الروتينات، ولكن قد تكون صغيرة نسبيًا بالنسبة لأجزاء التعليمات البرمجية.) من الواضح هنا أن نظام التشغيل يجب أن يدير الروتينات المشتركة في الذاكرة في هذه الحالة.

هناك فائدة إضافية للمكتبات المرتبطة ديناميكيًا (dynamically linked libraries (DLLs ، والمعروفة أيضًا باسم المكتبات المشتركة أو الكائنات المشتركة في أنظمة (UNIXترقيات وتحديثات سهلة. عندما يستخدم برنامج روتينًا من مكتبة قياسية ويتغير الروتين، فيجب إعادة بناء البرنامج (إعادة ربطه) لدمج التغييرات. مع ذلك، في حال استخدام مكتبات DLL، فما دامت القطعة الأصلية ثابتة، يُمكن تحديث البرنامج بمجرد تحميل إصدارات جديدة من مكتبات DLL على النظام. تُحفظ معلومات الإصدار في كلٍّ من البرنامج ومكتبات DLL، بحيث يُمكن للبرنامج تحديد إصدار مُحدد من مكتبة DLL عند الحاجة. عمليًا، في المرة الأولى التي يستدعي فيها البرنامج روتين DLL، سيتعرف القطعة الأصلية على هذه الحقيقة ويستبدل نفسه بالروتين الفعلي من مكتبة DLL. ستؤدي الاستدعاءات اللاحقة لنفس الروتين إلى الوصول إلى الروتين مباشرةً دون تكبد تكلفة الوصول إلى القطعة الأصلية. (وفقًا لنمط وكيل .(UML

٣ تخصيص الذاكرة المتجاور Contiguous Memory Allocation

إحدى طرق إدارة الذاكرة هي تحميل العمليات في مساحات متجاورة. تُخصص في البداية مساحة لنظام التشغيل، وتكون عادةً في مواقع الذاكرة الدنيا أو العليا، ثم تُخصص الذاكرة المتاحة المتبقية للعمليات حسب الحاجة. (عادةً ما يكون نظام التشغيل مُحمّلًا في الموقع الأدنى، لأن هذا هو المكان الذي توجد فيه متجهات المقاطعة interrupt vectors.

٣,١ حماية الذاكرة (تعيين وحماية الذاكرة)Memory Protection

يوفر النظام الموضح في الشكل 5 أدناه الحماية من وصول برامج المستخدم إلى مناطق لا ينبغي لها الوصول إليها، ويسمح بنقل البرامج إلى عناوين بدء مختلفة في الذاكرة حسب الحاجة، ويسمح لمساحة الذاكرة المخصصة لنظام التشغيل بالنمو أو التقلص ديناميكيًا مع تغير الاحتياجات.

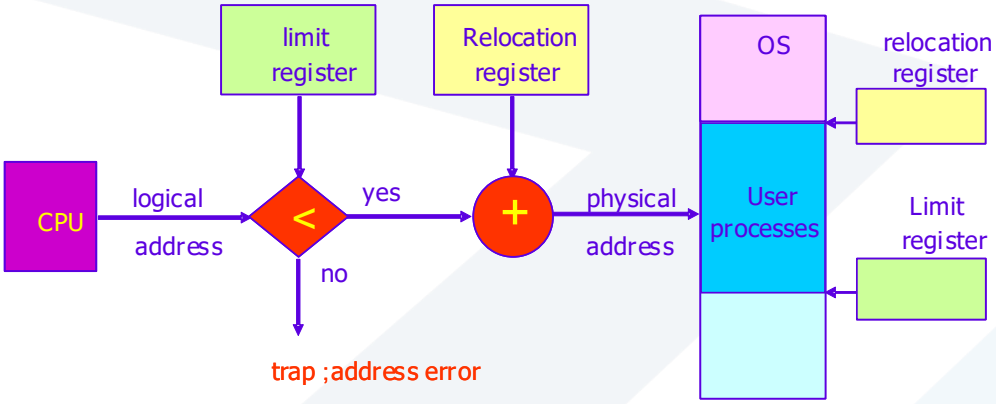
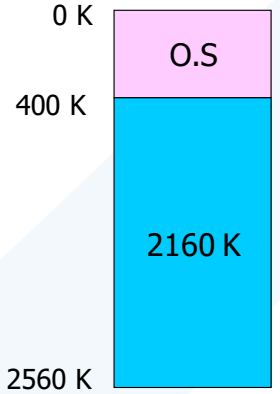


Figure 5 - Hardware support for relocation and limit registers

٣,٢ تخصيص الذاكرة Memory Allocation

إحدى طرق تخصيص الذاكرة المتجاورة هي تقسيم جميع الذاكرة المتاحة إلى أقسام متساوية الحجم، وتعيين كل عملية إلى قسمها الخاص. هذا يحد من عدد العمليات المتزامنة والحد الأقصى لحجم كل عملية، ولم يعد مستخدمًا.

ولكن الأعم هو تخصيص العمليات لأقسام ذات أحجام متغيرة variable partition في الذاكرة، حيث قد يحتوي كل قسم على عملية واحدة فقط. في نظام التقسيم المتغير هذا، يحتفظ نظام التشغيل بجدول يوضح أجزاء الذاكرة المتاحة والمشغولة. في البداية، تكون جميع الذاكرة متاحة لعمليات المستخدم، ومع استمرار تنفيذ العمليات تنشأ مساحات غير كافية لمتطلبات العمليات التي تنتظر دورها في التنفيذ وتسمى هذه المساحات بالفجوات holes وستحتوي الذاكرة على مجموعة من الفجوات بأحجام مختلفة.



Job queue		
Process	Needed memory	Burst time
P_1	600 K	10
P_2	1000 K	5
P_3	300 K	20
P_4	700 K	8
P_5	500 K	15

الشكل 6

يوضح الشكل 6 هذا النظام. في البداية، تخصص العمليات 1 و 2 و3 بمساحات متجاورة مناسبة لكل منها، وتبقى فجوة بحجم 260 kB لا تكفي لتنفيذ أي من العمليتين المتبقيتين. بعد مغادرة العملية 2، سيتم تحميل العملية 4 مكانها وسيكون هناك فجوة إضافية بحجم 300kB وسيكون في الذاكرة فجوتين لا تكفي أي منها بمفردها لتنفيذ العملية 5 ولا يمكن تخصيصهما معا لكونهما غير متجاورتين. وهذا سيضطر العملية 5 إلى الانتظار لحين انتهاء العملية الأولى مثلا لتشغل مكانها مما ينتج أيضا عن ذلك فجوة ثالثة غير متصلة مع بقية الفجوات حجمها 100kB.

إذا عند دخول العمليات إلى النظام، يأخذ نظام التشغيل في الاعتبار متطلبات الذاكرة لكل عملية ومقدار مساحة الذاكرة المتاحة لتحديد العمليات المخصصة لها. عندما تُخصص مساحة لعملية ما، تُحمل في الذاكرة، حيث يمكنها التنافس على وقت وحدة المعالجة المركزية. عند انتهاء عملية ما، تُحرر ذاكرتها، والتي قد يُوفرها نظام التشغيل لعملية أخرى. ماذا يحدث عندما لا تكون هناك ذاكرة كافية لتلبية متطلبات عملية قادمة؟ أحد الخيارات هو ببساطة رفض العملية وتقديم رسالة خطأ مناسبة. بدلاً من ذلك، يمكننا وضع هذه العمليات في قائمة انتظار. عند تحرير الذاكرة لاحقًا، يتحقق نظام التشغيل من قائمة الانتظار لتحديد ما إذا كانت ستلبي متطلبات الذاكرة لعملية الانتظار. بشكل عام، كما ذكرنا، تتكون كتل الذاكرة المتاحة من مجموعة من الفجوات بأحجام مختلفة متناثرة في جميع أنحاء الذاكرة. عندما تصل عملية ما وتحتاج إلى ذاكرة، يبحث النظام في المجموعة عن فجوة كبيرة بما يكفي لهذه العملية. إذا كانت الفجوة كبيرًا جدًا، يتم تقسيمها إلى جزئين. يُخصص جزء للعملية القادمة؛ ويُعاد الجزء الآخر إلى مجموعة الفجوات. عند انتهاء عملية ما، تُحرر كتلة الذاكرة الخاصة بها، والتي تُعاد بعد ذلك إلى مجموعة الفجوات. إذا كانت الفجوة الجديدة مجاورًة لفجوات أخرى، تُدمج هذه الفجوات المجاورة لتكوين فجوة واحدة أكبر. هذا الإجراء مثالٌ خاصٌّ على مشكلة تخصيص التخزين الديناميكي العامة dynamic storage allocation problem ، والتي تتعلق بكيفية تلبية طلب حجمه n من قائمة من الثقوب الشاغرة. هناك حلولٌ عديدة لهذه المشكلة تعتمد على واحدة من ثلاث استراتيجيات. حيث تعتبر هذه الاستراتيجيات "الأولى" و"الأفضل" و"الأسوأ" هي الأكثر استخدامًا لاختيار فجوة شاغرة من مجموعة الفجوات المتاحة.

هناك نهج بديل يتمثل في الاحتفاظ بقائمة من كتل الذاكرة غير المستخدمة (الفارغة) (الفجوات)، وإيجاد فجوة بحجم مناسب كلما دعت الحاجة إلى تحميل عملية في الذاكرة. هناك العديد من الاستراتيجيات المختلفة لإيجاد "أفضل" تخصيص للذاكرة للعمليات، بما في ذلك الاستراتيجيات الثلاث الأكثر شيوعًا:

1. الملاءمة الأولى - First fit وتعتمد على البحث في قائمة الفجوات حتى يتم إيجاد فجوة كبيرة بما يكفي لتلبية الطلب، ثم يخصص جزءًا من تلك الفجوة لتلك العملية. أي جزء من الفجوة غير المطلوبة للطلب يبقى في القائمة الحرة كفجوة أصغر. يمكن للطلبات اللاحقة أن تبدأ البحث إما من بداية القائمة أو من النقطة التي انتهى عندها هذا البحث.
2. أفضل ملاءمة - Best fit حيث يخصص أصغر فجوة كبيرة بما يكفي لتلبية الطلب. هذا يوفر فجوات كبيرة لطلبات العمليات الأخرى التي قد تحتاجها لاحقًا، ولكن الأجزاء غير المستخدمة الناتجة من الفجوات قد تكون صغيرة جدًا بحيث لا تُجدي نفعًا، وبالتالي ستُهدر. يمكن أن يُسرّع الحفاظ على ترتيب القائمة الفارغة عملية العثور على الفجوات المناسبة.
3. أسوأ ملاءمة - Worst fit حيث يخصص أكبر فجوة متاحة، مما يزيد من احتمالية استخدام الجزء المتبقي لتلبية الطلبات المستقبلية.

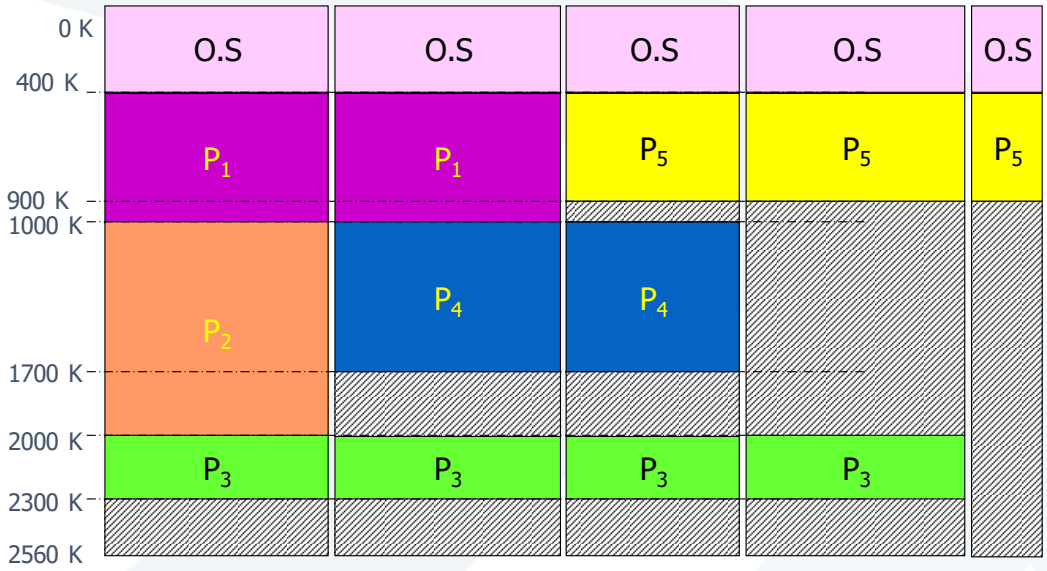
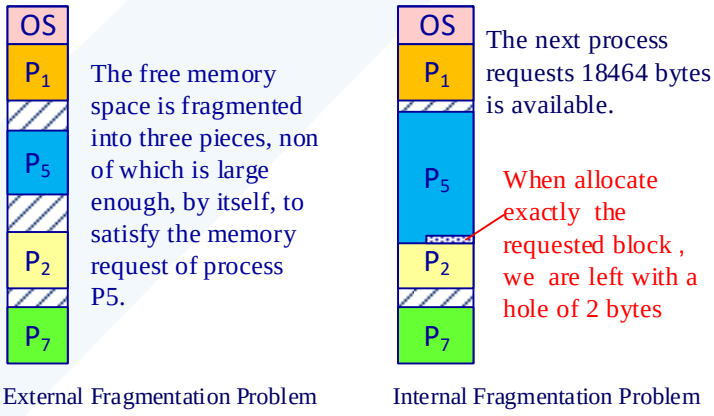


Figure 7 Variable partition.

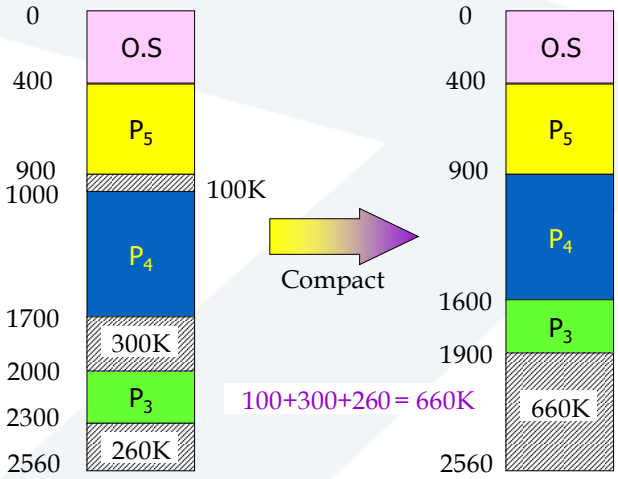
تُظهر المحاكاة أن أول ملاءمة أو أفضل ملاءمة أفضل من أسوأ ملاءمة من حيث الوقت واستخدام مساحة التخزين. الملاءمة الأولى وأفضل ملاءمة متساوية تقريبًا من حيث استخدام مساحة التخزين، ولكن الملاءمة الأولى أسرع.

3.3. التجزئة Fragmentation

تعاني جميع استراتيجيات تخصيص الذاكرة من التجزئة الخارجية external fragmentation ، على الرغم من أن الملاءمة الأولى وأفضل ملاءمة تواجه مشاكل أكثر من أسوأ ملاءمة. التجزئة الخارجية تعني أن الذاكرة المتاحة مُقسّمة إلى أجزاء صغيرة كثيرة، لا يكفي أي منها لتلبية متطلبات الذاكرة التالية، على الرغم من أن المجموع الكلي قد يكون كذلك.



الشكل 8: التجزئة الداخلية والتجزئة الخارجية

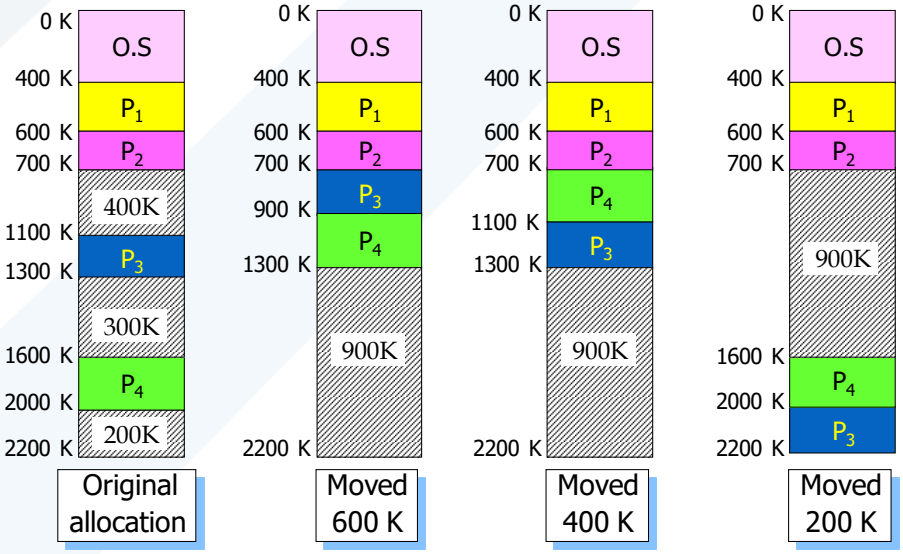


الشكل 9: دمج الفراغات

قد يختلف مقدار الذاكرة المفقودة بسبب التجزئة باختلاف الخوارزمية وأنماط الاستخدام وبعض قرارات التصميم، مثل تحديد نهاية الفجوة التي يجب تخصيصها والنهاية التي يجب حفظها في القائمة المجانية.

تحدث التجزئة الداخلية Internal fragmentation أيضًا في جميع استراتيجيات تخصيص الذاكرة. ويحدث ذلك لأن الذاكرة تُخصص في كتل ذات حجم ثابت، بينما نادرًا ما تكون الذاكرة الفعلية المطلوبة بهذا الحجم بالضبط.

لاحظ أن التأثير نفسه يحدث مع محركات الأقراص الصلبة، وأن الأجهزة الحديثة تُوفر لنا محركات أقراص وذاكرة أكبر حجمًا على حساب أحجام كتل أكبر باستمرار، مما يُترجم إلى فقدان المزيد من الذاكرة بسبب التجزئة الداخلية.



الشكل 10: تجميع الفراغات

تستخدم بعض الأنظمة كتلًا متغيرة الحجم لتقليل الخسائر الناتجة عن التجزئة الداخلية.

إذا كانت البرامج في الذاكرة قابلة للنقل (باستخدام ربط العناوين وقت التنفيذ)، فيمكن الحد من مشكلة التجزئة الخارجية عبر الضغط compaction ، أي نقل جميع العمليات إلى أحد طرفي الذاكرة الفعلية. يقتصر هذا على تحديث سجل التحويل لكل عملية، حيث يتم تنفيذ جميع الأعمال الداخلية باستخدام عناوين منطقية.

وهناك حل آخر، كما سنرى في الأقسام القادمة، وهو السماح للعمليات باستخدام كتل غير متجاورة من الذاكرة الفعلية، مع سجل نقل منفصل لكل كتلة.

٤ التجزئة Segmentation

٤,١ الطريقة الأساسية

لا يعتقد معظم المستخدمين (المبرمجين) أن برامجهم موجودة في مساحة عنوان متصلة واحدة. ويميلون إلى اعتبار ذاكرتهم عبارة عن مقاطع متعددة، كل منها مخصص لاستخدام معين، مثل الكود، والبيانات، والمكدس، والكومة، إلخ. تدعم تجزئة الذاكرة هذا المنظور من خلال توفير عناوين برقم مقطع (مُطابق لعنوان قاعدة المقطع) ورقم إزاحة من بداية ذلك المقطع. على سبيل المثال، قد يُنشئ محول لغة C خمسة مقاطع لكود المستخدم، وكود للمكتبة، وآخر للمتغيرات العامة (الثابتة)، وكذلك للمكدس، والكومة، كما هو موضح في الشكل 11:

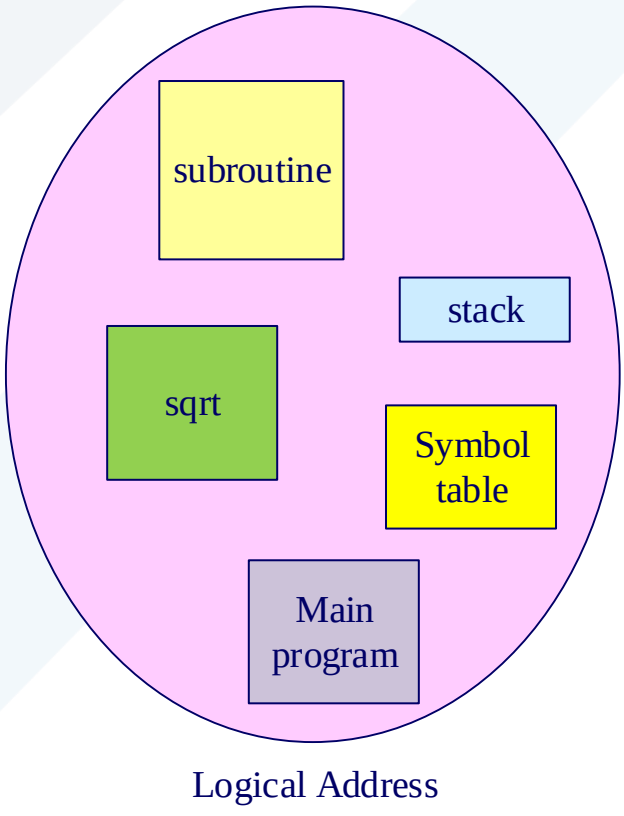


Figure 11 P:rogrammer's view of a program.

٤,٢ تجهيزات التجزئة Segmentation Hardware

يُطابق جدول المقطع **segment table** عناوين إزاحة المقطع مع العناوين الفعلية، ويتحقق في الوقت نفسه من العناوين غير الصالحة، باستخدام نظام مشابه لجداول الصفحات وسجلات قاعدة النقل التي تمت مناقشتها سابقًا. يُحفظ كل جزء في ذاكرة متجاورة وقد يكون بأحجام مختلفة، ولكن يمكن أيضًا دمج التجزئة مع التصفيح Paging كما سنرى لاحقًا.)

5/ التصفيح Paging

• التصفيح (من كلمة صفحة) هو نظام لإدارة الذاكرة يسمح للعمليات بأن تكون الذاكرة الفعلية متقطعة، ويزيل مشاكل التجزئة الخارجية من خلال تخصيص الذاكرة في كتل متساوية الحجم تُعرف بالصفحات. حيث يُزيل التصفيح معظم مشاكل الطرق الأخرى التي ناقشناها سابقًا، وهو أسلوب إدارة الذاكرة السائد المستخدم حاليًا.

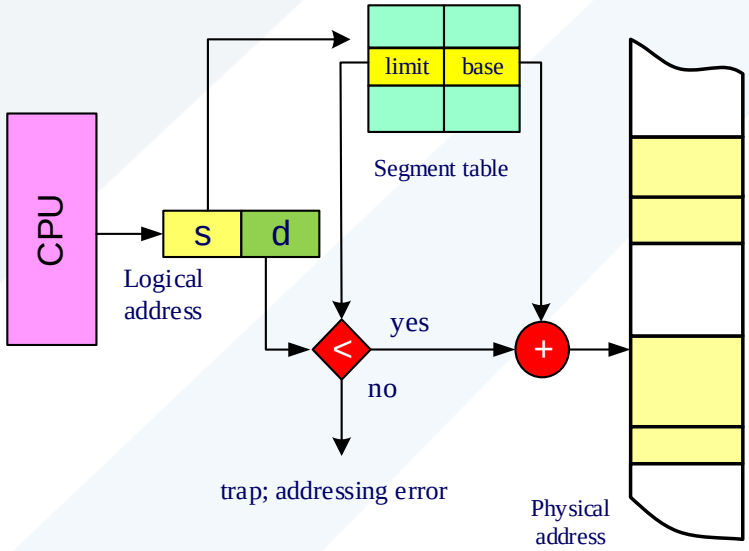


Figure 12 – Segmentation hardware

5.1 الطريقة الأساسية

الفكرة الأساسية وراء التصفيح هي تقسيم الذاكرة الفعلية إلى عدد من الكتل متساوية الحجم تُسمى الإطارات Frames، وتقسيم مساحة الذاكرة المنطقية للبرنامج إلى كتل من نفس الحجم تُسمى الصفحات pages .

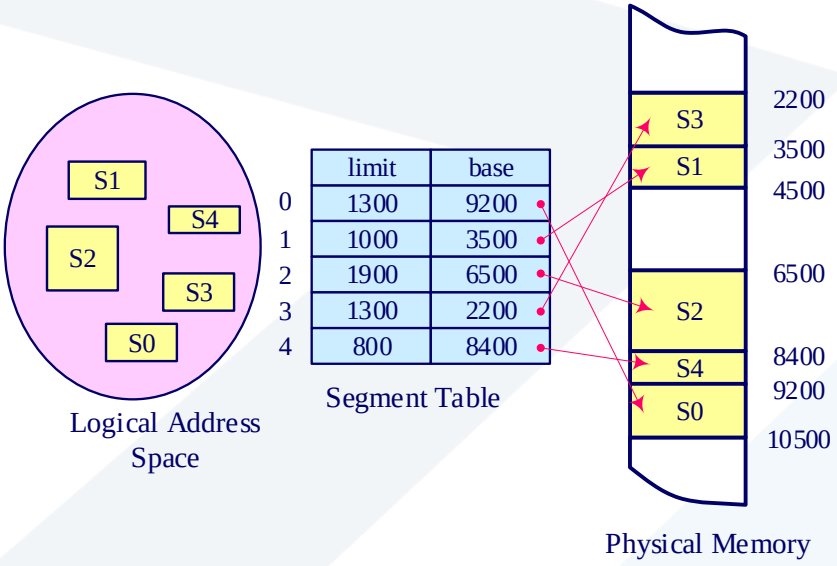


Figure 13 - Example of segmentation

بعد هذا التقسيم يمكن وضع أي صفحة (من أي عملية) في أي إطار متاح. ولمعرفة موقع أي صفحة وفي أي إطار وضعت، يُستخدم جدول الصفحات page table . في المثال التالي، على سبيل المثال، الصفحة 2 من الذاكرة المنطقية للبرنامج مُخزنة حاليًا في الإطار 3 من الذاكرة الفعلية:

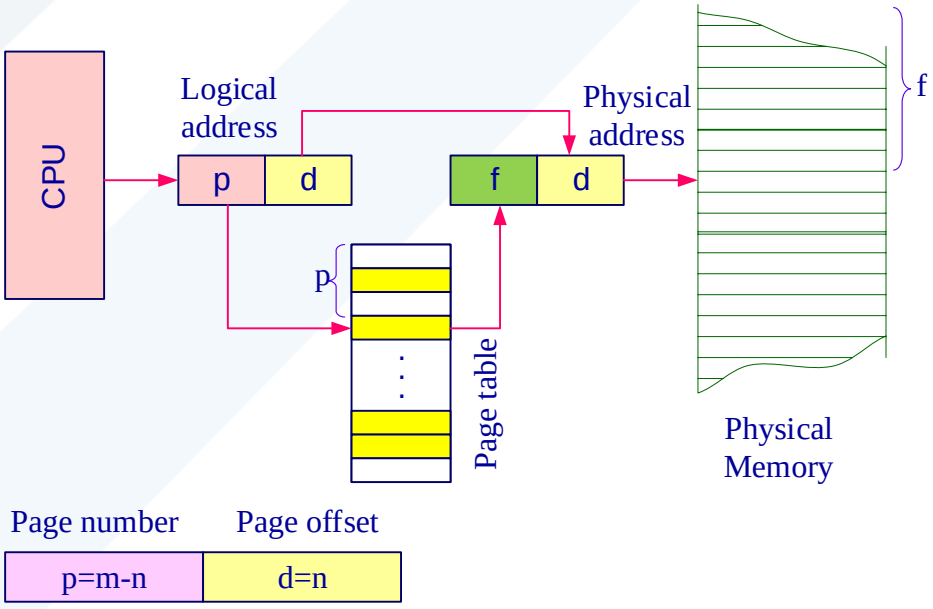


Figure 14 - Paging hardware

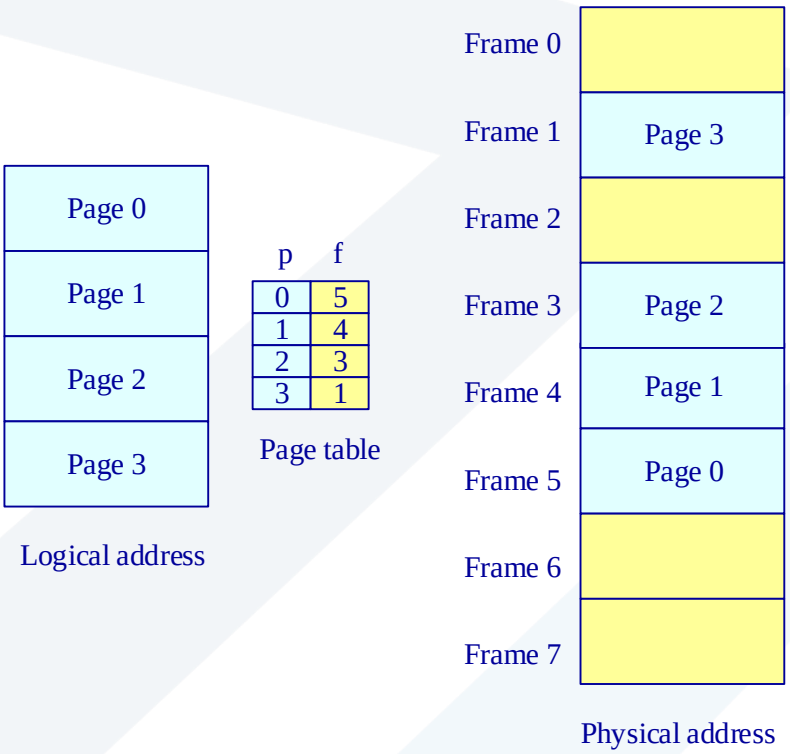


Figure 15 - Paging model of logical and physical memory

يتكون العنوان المنطقي من جزأين: رقم الصفحة التي يوجد فيها العنوان page number ، وإزاحة من بداية تلك الصفحة page offset. (يحدد عدد البتات في رقم الصفحة عدد الصفحات التي يمكن لعملية واحدة معالجتها. يحدد عدد البتات في الإزاحة الحد الأقصى لحجم كل صفحة، ويجب أن يتوافق مع حجم إطار النظام).

يربط جدول الصفحات رقم الصفحة برقم إطار، لإنتاج عنوان فعلي يتكون أيضًا من جزأين: رقم الإطار frame number والإزاحة داخل ذلك الإطار frame offset. يحدد عدد البتات في رقم الإطار عدد الإطارات التي يمكن للنظام معالجتها، ويحدد عدد البتات في الإزاحة حجم كل إطار.

يتم تحديد أرقام الصفحات وأرقام الإطارات وأحجام الإطارات حسب البنية، ولكنها عادةً ما تكون قوى للرقم 2، مما يسمح بتقسيم العناوين عند عدد معين من البتات. على سبيل المثال، إذا كان حجم العنوان المنطقي m^2 وحجم الصفحة n^2، فإن البتات عالية الترتيب m−n للعنوان المنطقي تُشير إلى رقم الصفحة، بينما تُمثل البتات n المتبقية الإزاحة. لاحظ أيضًا أنه ليس بالضرورة أن يكون عدد البتات في رقم الصفحة وعدد البتات في رقم الإطار متطابقين. يُحدد الأول نطاق عنوان مساحة العنوان المنطقي، بينما يتعلق الثاني بمساحة العنوان المادية.

انظر إلى المثال المصغر التالي، حيث تحتوي عملية على 16 بايت من الذاكرة المنطقية، تخصص في صفحات 4 بايت إلى 32 بايت من الذاكرة المادية. (من المفترض أن بعض العمليات الأخرى ستستهلك الـ 16 بايت المتبقية من الذاكرة الفعلية.)

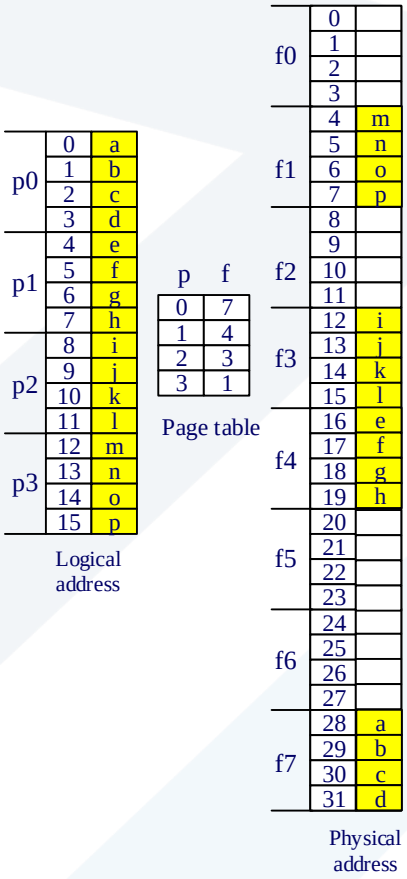


Figure 16 - Paging example for a 32-byte memory with 4-byte pages

لاحظ أن التصفيح يشبه وجود جدول لمسجلات نقل البيانات، مسجل واحد لكل صفحة من الذاكرة المنطقية. لا يوجد تجزئة خارجية مع التصفيح. يتم استخدام جميع كتل الذاكرة الفعلية، ولا توجد فجوات بينها، ولا توجد مشاكل في إيجاد الحجم المناسب لكل جزء من الذاكرة. مع ذلك، هناك تجزئة داخلية. إذ يتم تخصيص الذاكرة في أجزاء بحجم صفحة، وفي المتوسط، تكون الصفحة الأخيرة ممتلئة بنصفها فقط، مما يهدر نصف صفحة من الذاكرة لكل عملية في المتوسط. (وربما أكثر، إذا احتفظت العمليات بشفراتها وبياناتها في صفحات منفصلة.) تستهلك أحجام الصفحات الأكبر المزيد من الذاكرة، ولكنها أكثر كفاءة من حيث التكلفة. وقد اتجهت الاتجاهات الحديثة إلى زيادة أحجام الصفحات، حتى أن بعض الأنظمة تعتمد استخدام صفحات متعددة الأحجام لمحاولة تحقيق أقصى استفادة من كلا الجانبين.

عادةً ما تكون مدخلات جدول الصفحات (أرقام الإطارات) أرقامًا بحجم 32 بت، مما يسمح بالوصول إلى 32^2 إطار صفحة فعلي. إذا كان حجم كل إطار 4 كيلوبايت (2^12)، فهذا يعني 32 + 12 = 44 بت من مساحة العنوان الفعلية و 16 تيرابايت من الذاكرة الفعلية القابلة للعنونة).

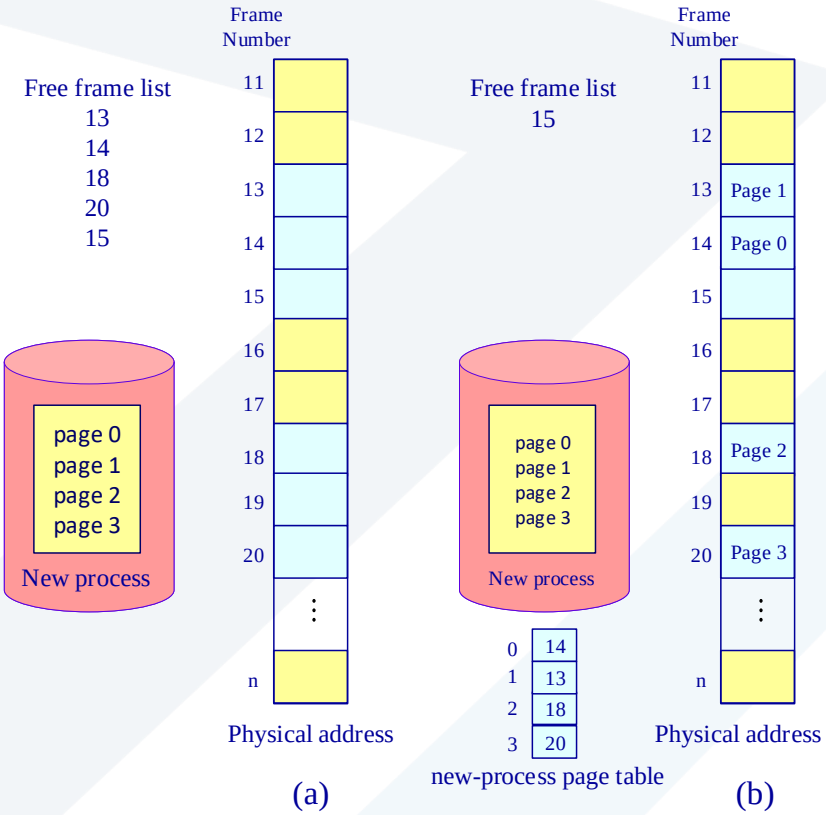


Figure 17 - Free frames (a) before allocation and (b) after allocation

عندما تطلب عملية ذاكرة (مثلاً، عند تحميل شيفرتها من القرص)، تُخصص إطارات حرة من قائمة إطارات حرة، وتُدرج في جدول صفحات تلك العملية.

تُمنع العمليات من الوصول إلى ذاكرة أي شخص آخر لأن جميع طلبات الذاكرة الخاصة بها تُربط عبر جدول صفحاتها. لا توجد طريقة لها لتوليد عنوان يُربط بمساحة ذاكرة أي عملية أخرى.

يجب على نظام التشغيل تتبع جدول صفحات كل عملية على حدة، وتحديثه كلما تم نقل صفحات العملية من الذاكرة وإليها، وتطبيق جدول الصفحات الصحيح عند معالجة استدعاءات النظام لعملية معينة. كل هذا يزيد من التكلفة الإجمالية عند تبديل العمليات داخل وحدة المعالجة المركزية وخارجها. (يجب تحديث جدول الصفحات النشط حاليًا ليعكس العملية الجارية حاليًا).