Управление на паметта

Любомир Чорбаджиев[1]
lchorbadjiev@elsys-bg.org

[1]Технологическо училище "Електронни системи"
Технически университет, София

13 май 2009 г.

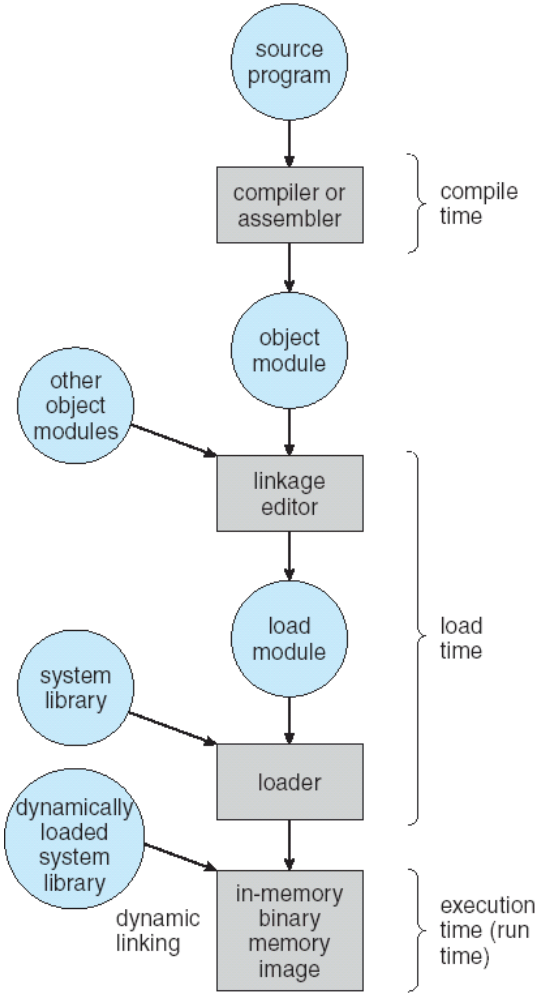
Съдържание

1. Въведение
2. Основни концепции
3. Динамично зареждане и свързване
4. Размяна (swapping)
5. Раздели в оперативната памет
6. Делене на страници
7. Сегментна организация
8. Виртуална памет

Въведение

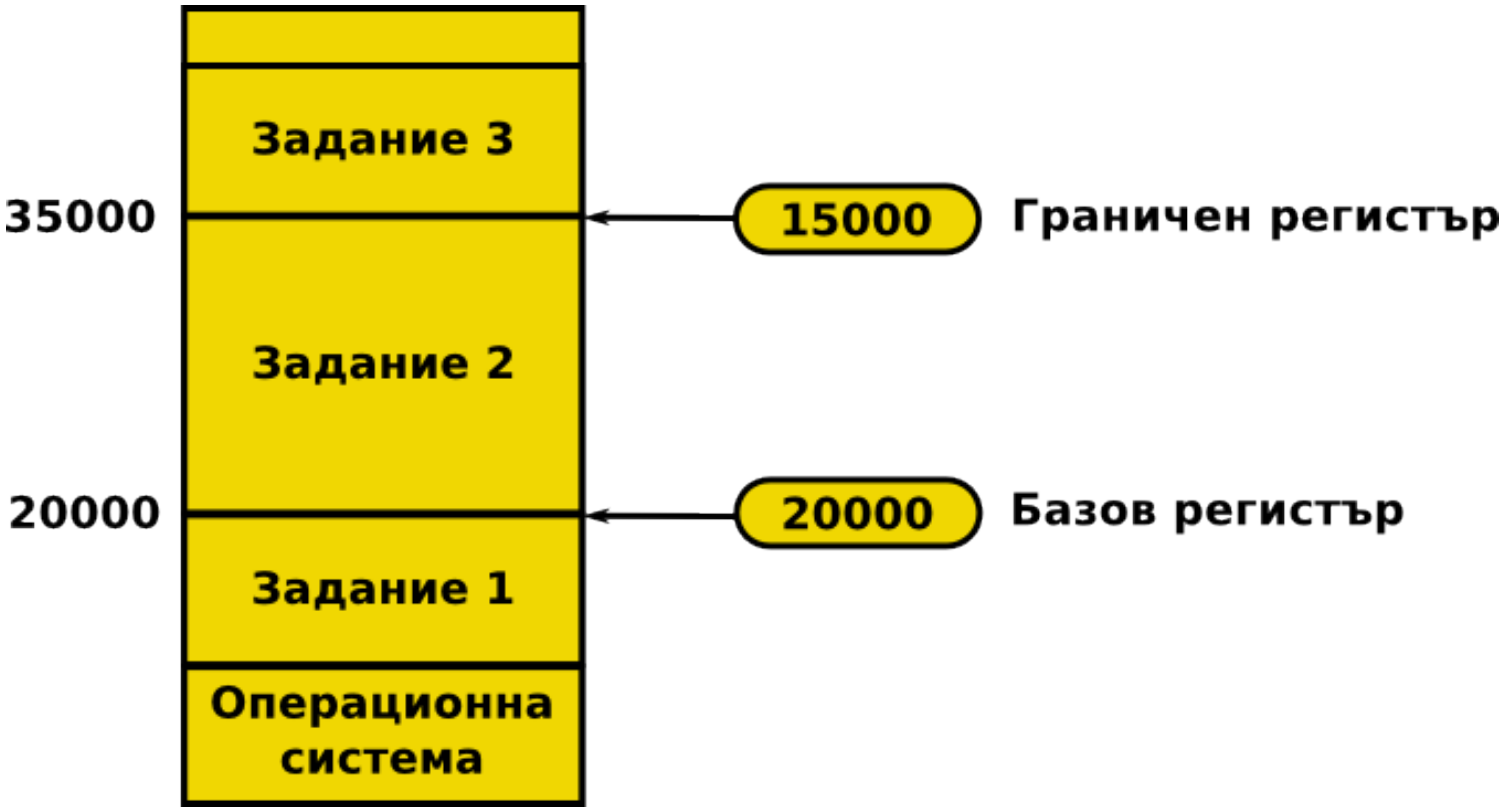
- За да бъде изпълнена една програма, първо тя трябва да бъде заредена в оперативната памет на компютърната система.
- *Входна опашка* се нарича опашка от процеси, които чакат да бъдат заредени в оперативната памет на компютъра, за да бъдат изпълнени.
- Потребителската програма преминава през няколко стъпки преди да бъде изпълнена.

Въведение

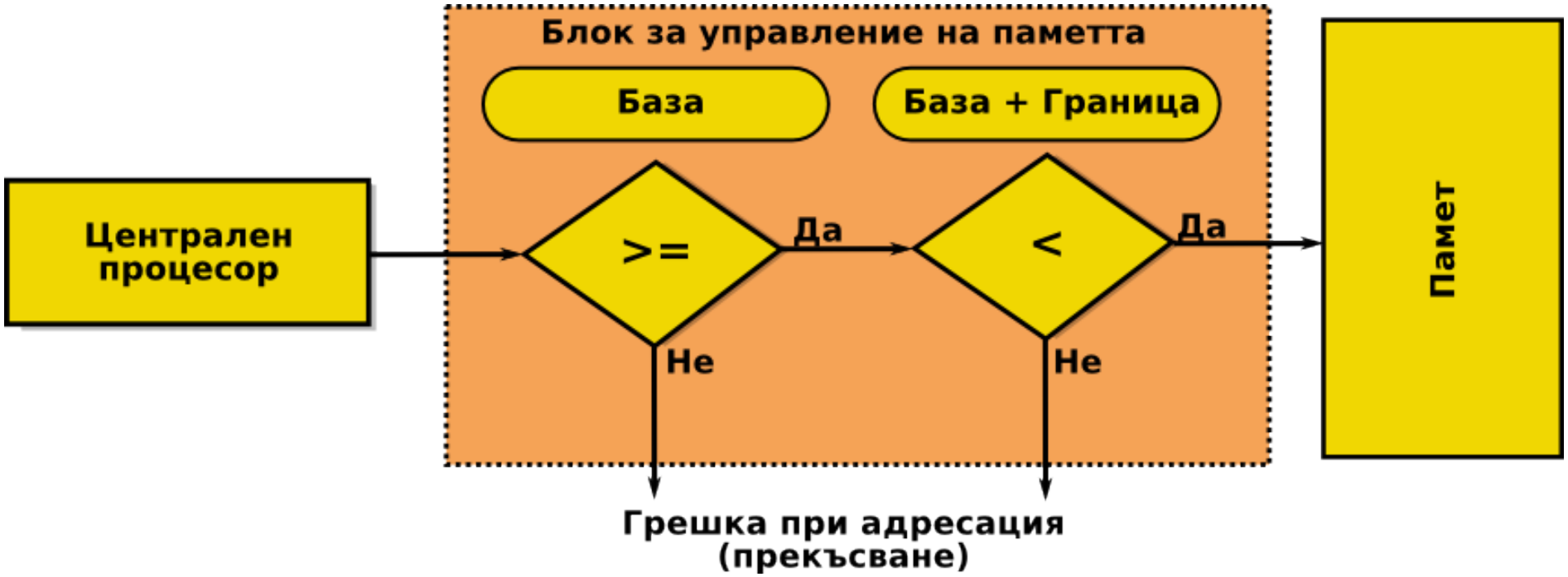


Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Базов и граничен регистър



Базов и граничен регистър



Свързване на програмата с адреси в паметта

Свързването на инструкциите и данните на програмата с адреси в оперативната памет на компютъра може да се реализира в различни моменти от жизнения цикъл на програмата.

- *По време на компилация:* Ако разположението на програмата в паметта е известно предварително, още по време на компилация на програмата, то компилатора може да генерира абсолютни адреси в паметта. Ако трябва да се промени разположението в паметта на програмата, то е необходима тя да се прекомпилира.

Свързване на програмата с адреси в паметта

- *По време на зареждане:* Когато разположението на програмата в паметта не е предварително известно, то е необходимо компилаторът да генерира *премествае* код.
- *По време на изпълнение:* Свързването на програмата с адреси в оперативната памет може да се отложи до момента на изпълнение на програмата, ако програмата (или части от нея) могат да бъдат преместени от един сегмент на оперативната памет в друг. За тази цел обикновено е необходима хардуерна поддръжка.

Логически и физически адреси

- Логически адрес се нарича адрес, генериран от централния процесор. Понякога тези адреси се наричат виртуални адреси.
- Физически адрес е адресът на клетката от оперативната памет, който се зарежда върху шината за адресиране.
- Разделянето на концепциите за логическо и физическо адресно пространство има основно значение за управлението на паметта.
- Управлението на паметта се заключава в начините, по които логическото адресно пространство се свързва с физическото адресно пространство.

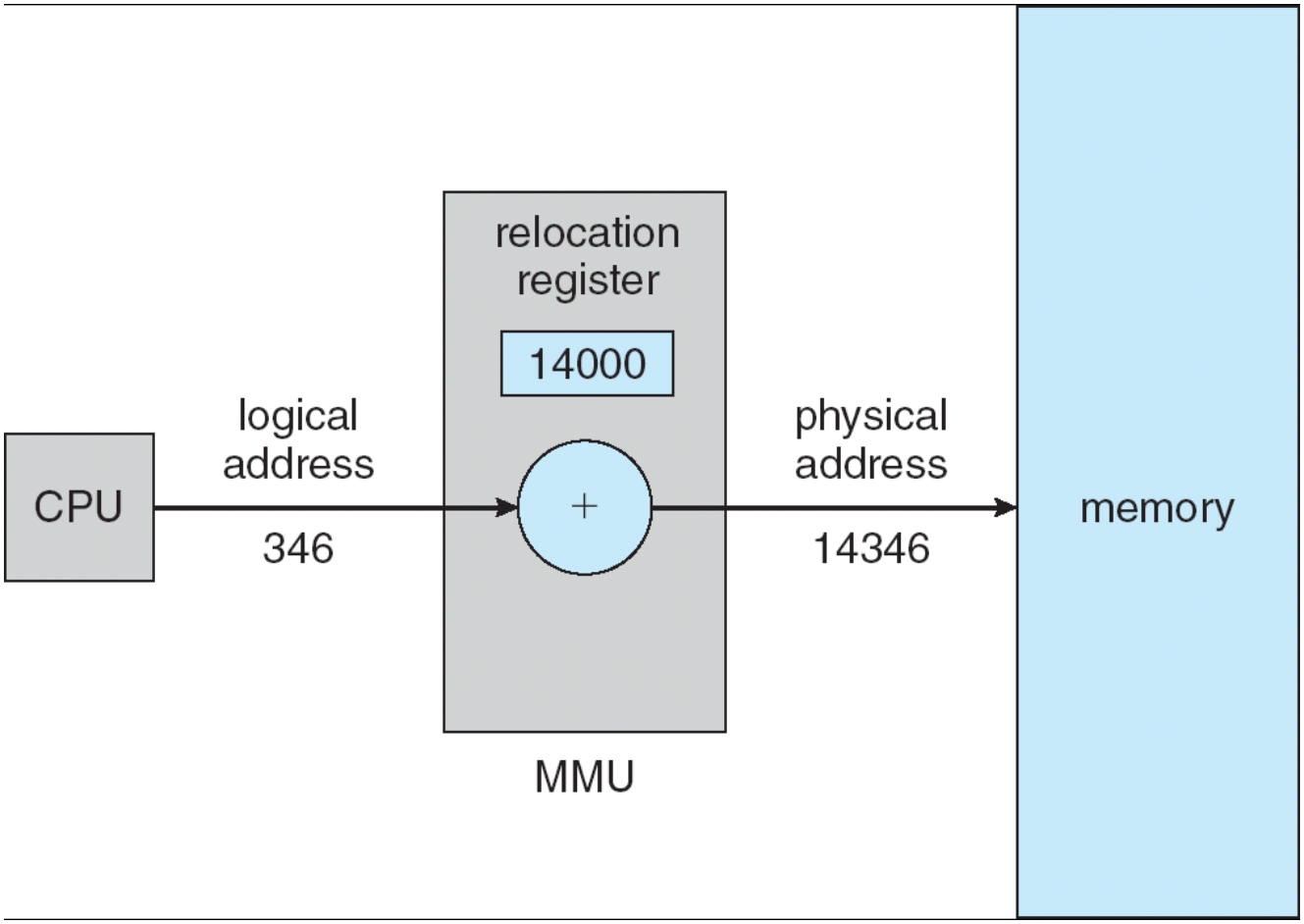
Логически и физически адреси

- В случаите на свързване на адреси при компилация и по време на зареждане на програмата, логическите и физическите адреси съвпадат.
- При използване на подхода за свързване на адресите по време на изпълнение, логическите и физическите адреси са различни.

Блок за управление на паметта (MMU)

- Блок за управление на паметта се нарича част от хардуера (централния процесор), който се използва за изграждане на съответствие между логическите и физическите адреси.
- Основен елемент на блока за управление на паметта е регистърът за преместване (базов регистър) (relocation register, base register), който се добавя към всеки генериран от централния процесор логически адрес в момента, в който централния процесор се обръща към паметта.
- Потребителската програма работи само с логически адреси — тя никога не използва физически адреси.

Блок за управление на паметта (MMU)



Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

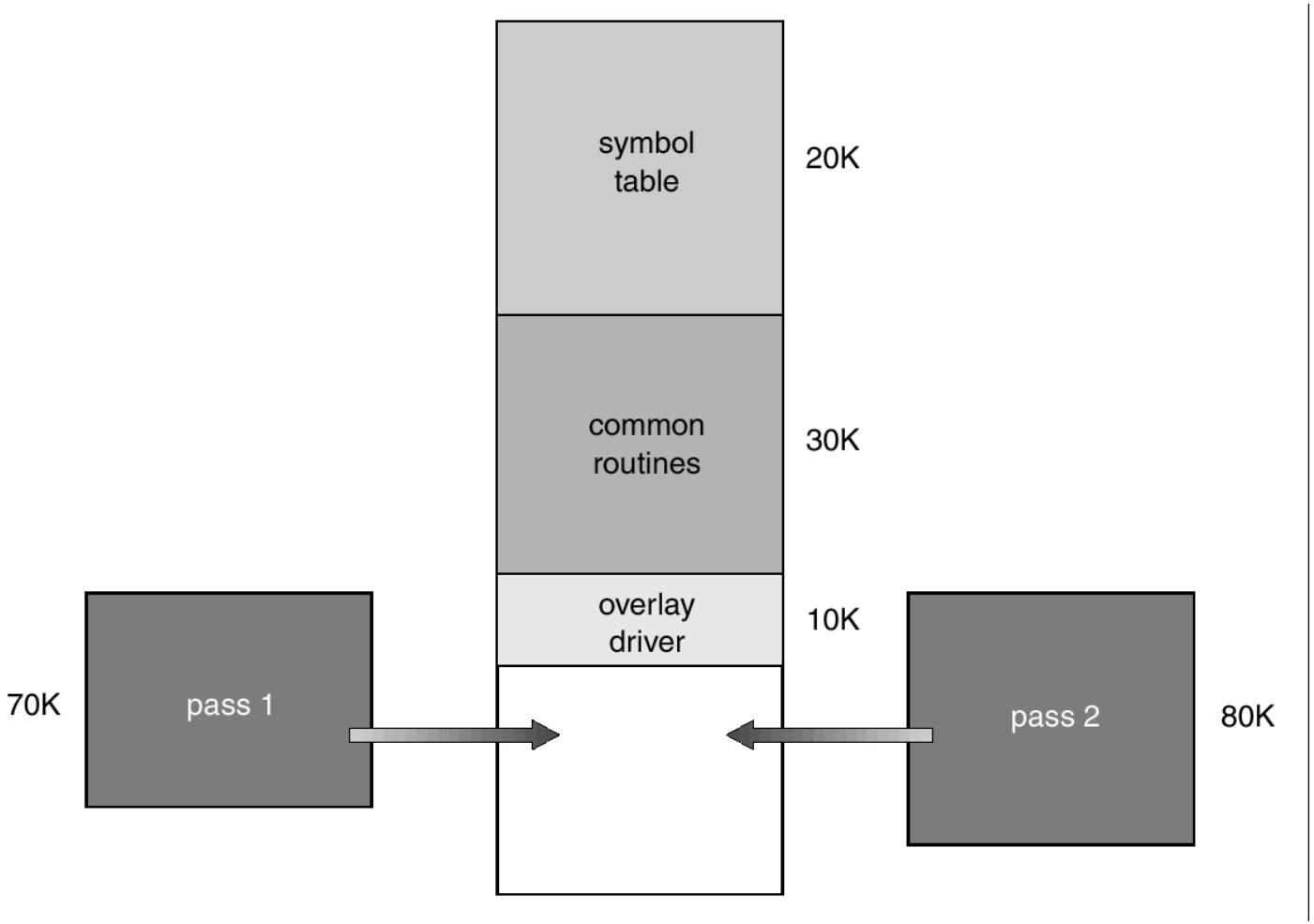
Динамично зареждане

- Зареждането в паметта на функциите и подпрограмите се отлага до момента, в който те действително са необходими (мързеливо зареждане).
- Този подход позволява по-добро използване на паметта – функциите и подпрограмите които не се използват, не се зареждат в оперативната памет. Когато големи участъци от кода са предназначени за обработка на рядко случващи се ситуации, динамичното зареждане води до значително по-добро използване на паметта.
- Динамичното зареждане не изисква специална поддръжка от операционната система. То може да бъде реализирано изцяло в потребителската програма.

Динамично свързване

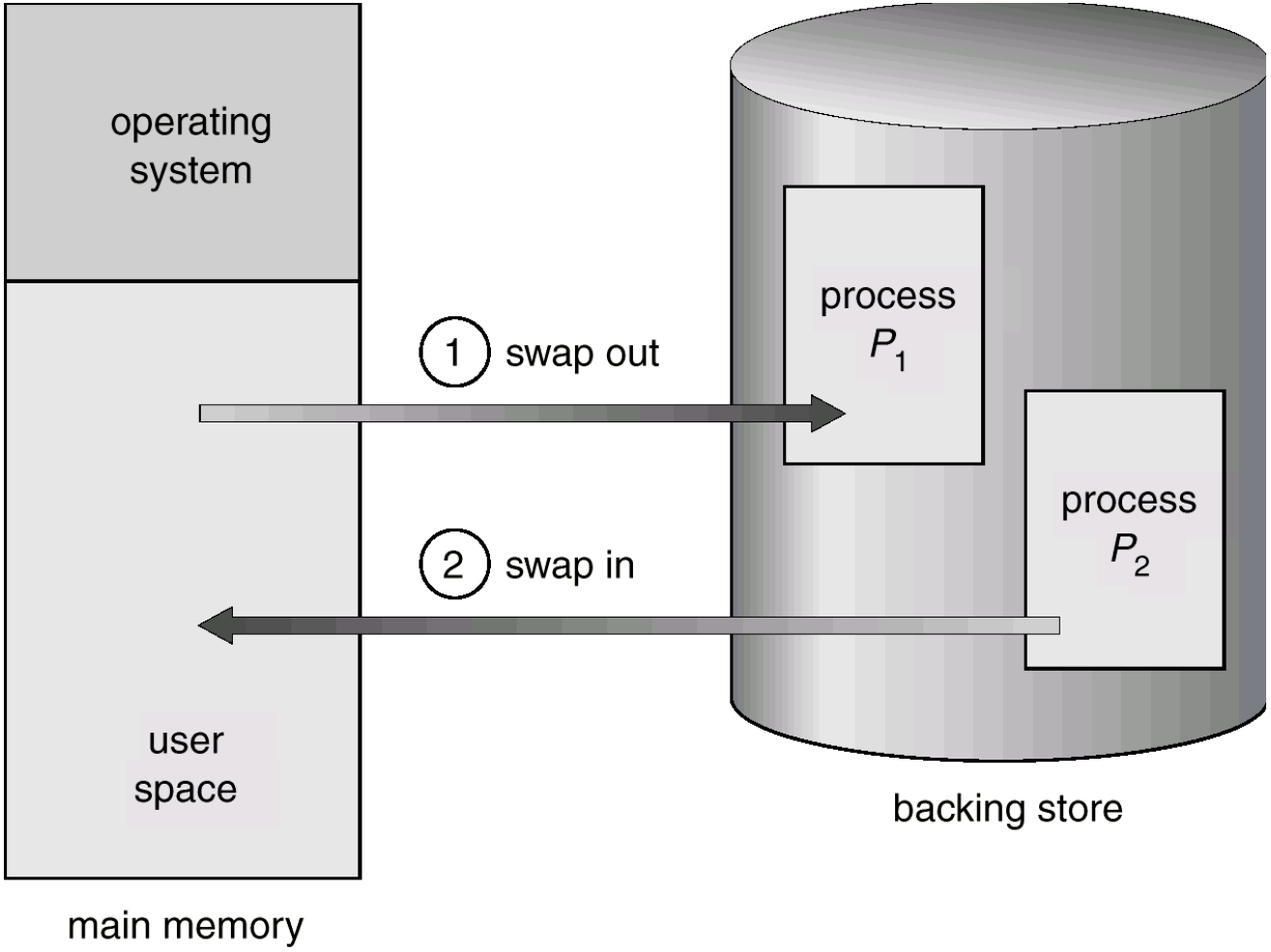
- Динамичното свързване е особено полезно при реализиране на библиотеки.
- В кода на приложението се интегрират малки участъци код, наречени *стъбове (stub)*, чиято задача е да намерят правилната, заредена вече в оперативната памет функция.
- Когато стъбът намери необходимата функция, той се заменя с адреса на тази функция и я извиква.
- Операционната система трябва да провери дали съответната функция е в адресното пространство на процеса, който я извиква.

Препокриване (Overlay)



Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Размяна (swapping)

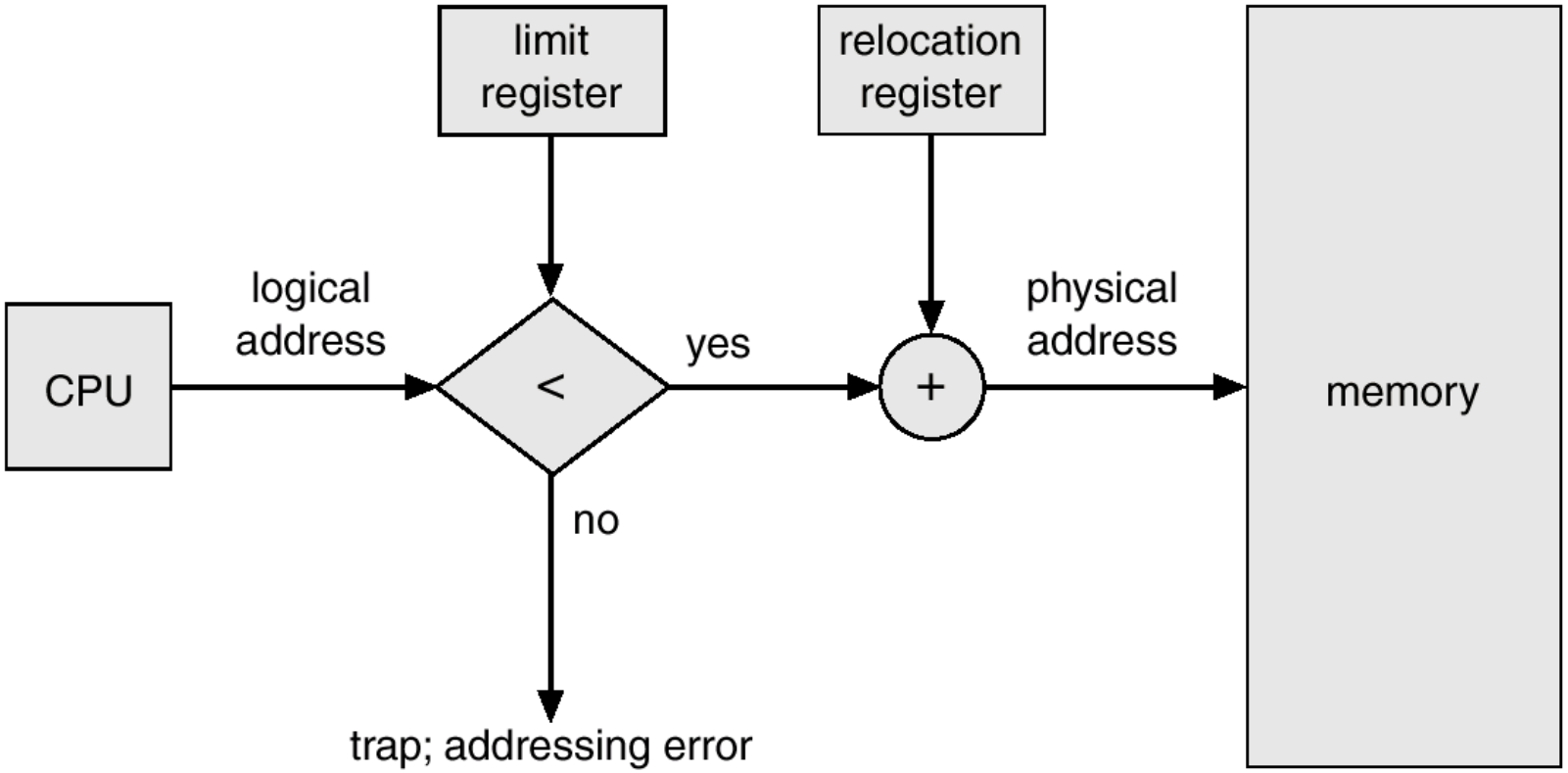


Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Фиксирани раздели

- Оперативната памет се разделя на определен брой и с определен размер парчета – раздели.
- В първите операционни системи често се използва във варианта два фиксирани раздела – един за резидентния монитор (операционна система) и втори раздел за потребителското задание.
- Този подход може да се комбинира със свързването на физическите адреси по време на компилация. По време на компилация се определя в кой точно раздел ще работи дадена програма и компилаторът и свързващият редактор генерират абсолютни адреси.
- По-гъвкаво е да се използва преместваем обектен код – свързването на физическите адреси да става по време на зареждане на програмата, в зависимост от това в кой раздел се зарежда тя.

Фиксирани раздели



Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Фрагментация

- Наличието на неизползвани участъци от паметта се нарича *фрагментация*
- Една от основните причини за фрагментация на оперативната памет при фиксираните раздели е, че част от раздела не се използва, тъй като размерът на раздела е фиксирана и не може да се променя.

Фрагментация

- Фрагментацията на оперативната памет се разделя на два вида:
  - *Външна фрагментация* – съществува свободна оперативна памет, но тя е разделена на малки дупки между разделите, които се използват от работещите процеси.
  - *Вътрешна фрагментация* – заделената памет е по-голяма от паметта, която реално се използва от процеса; тази памет е вътрешна за раздела, но не се използва от процеса.

Променливи раздели

- *Дупка* – свободен участък от оперативната памет.
- Първоначално цялата оперативна памет (преди да се заредят процеси) може да се разглежда като една "дупка".
- Когато един процес трябва да бъде зареден в оперативната памет, то за него се заделя памет от някоя от дупките, които са достатъчно големи за да поберат процеса.
- Операционната система трябва да съхранява информацията за заетите раздели и за свободните раздели (дупки) в оперативната памет.
- Този подход също страда от фрагментация.

Уплътняване

- Възможен подход за справяне с фрагментацията при променливите раздели е *уплътняването* на оперативната памет.
- Идеята е всички процеси да се преместят в оперативната памет така, че да остане само една дупка.
- Този подход има ред недостатъци:
  - Местенето на процеси обаче, е времеемко.
  - Докато се местят процесите трябва да бъдат спрени.
- Поради тези недостатъци уплътняването на оперативната памет се използва рядко.

Стратегии за избиране на раздел

Нека си представим, че са стартира нов процес, който има нужда от N байта оперативна памет. Възможните стратегии за избор на подходящ раздел от паметта се следните:

- *Първият подходящ:* избира се първата дупка в паметта, която е достатъчно голяма.
- *Най-подходящият:* избира се най-малката дупка в паметта, която е достатъчно голяма.
- *Най-неподходящият:* избира се най-голямата дупка в оперативната памет.

Моделиране на тези стратегии показва, че стратегиите "първият подходящ" и "най-подходящият" имат по-добра производителност от гледна точка на бързодействие и използваемост на оперативната памет. Това са най-често използваните стратегии при работа с раздели.

Делене на страници

- Логическото и физическото адресно пространство на процеса не са непрекъснати (един раздел), а са разделени на парчета.
- Физическото адресно пространство е разделено на фиксирани по размер блоково, които се наричат *кадри (frames)*. Типично като размер на кадъра се избира число, степен на 2 – да речем 512, 1024,...
- Логическото адресно пространство се разделя на блокове със същия размер (като размера на кадъра), които се наричат *страници (pages)*.
- Операционната система трябва да съхранява информация за всички свободни кадри.

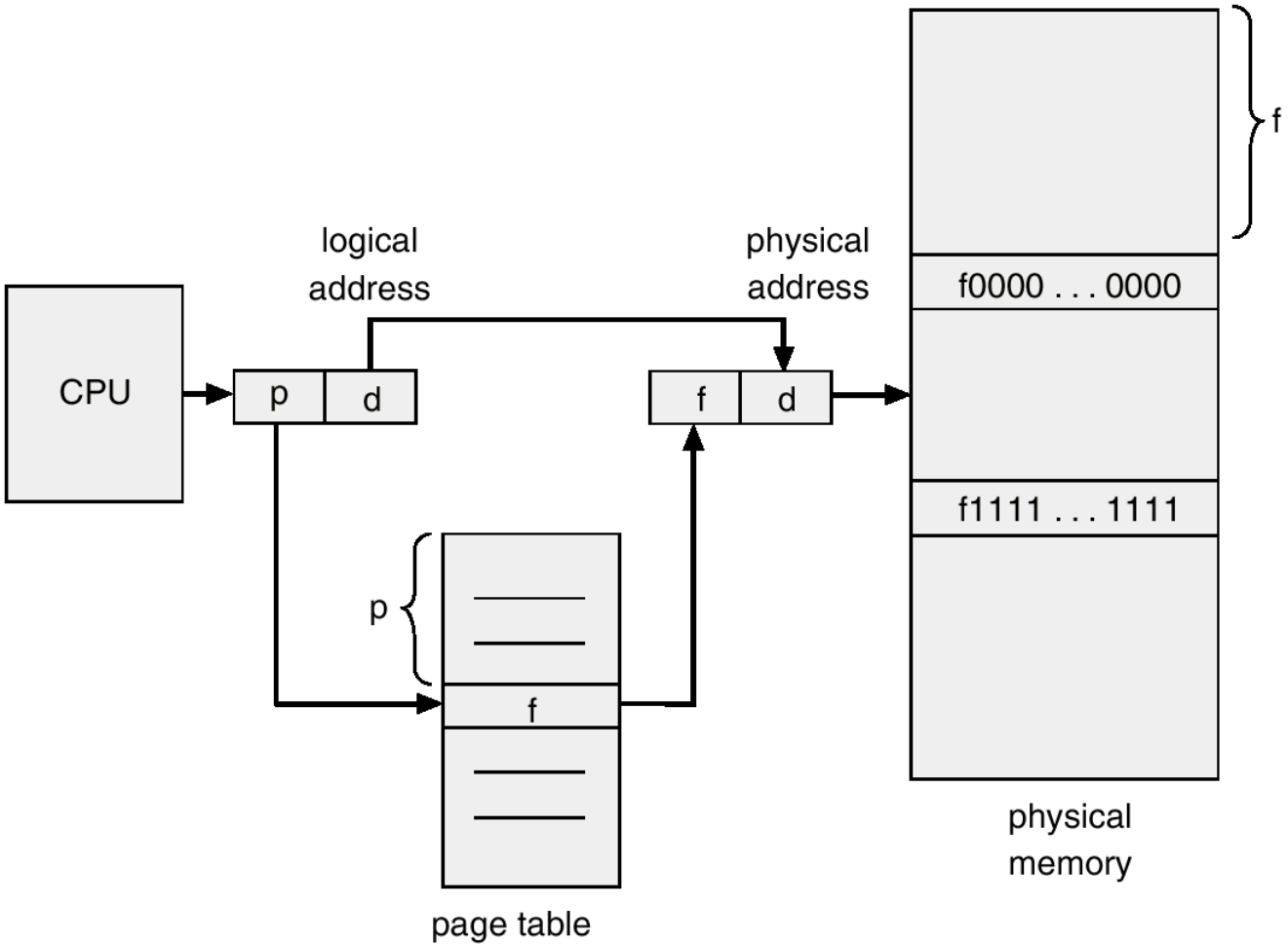
Делене на страници

- За да се зареди една програма с размер N страници в оперативната памет е необходимо операционната система да намери съответния брой свободни кадри, да зареди програмата и да инициализира таблицата за преобразуване на логическите във физически адреси.
- Деленето на страници води до увеличаване на вътрешната фрагментация.

Преобразуване на адреси

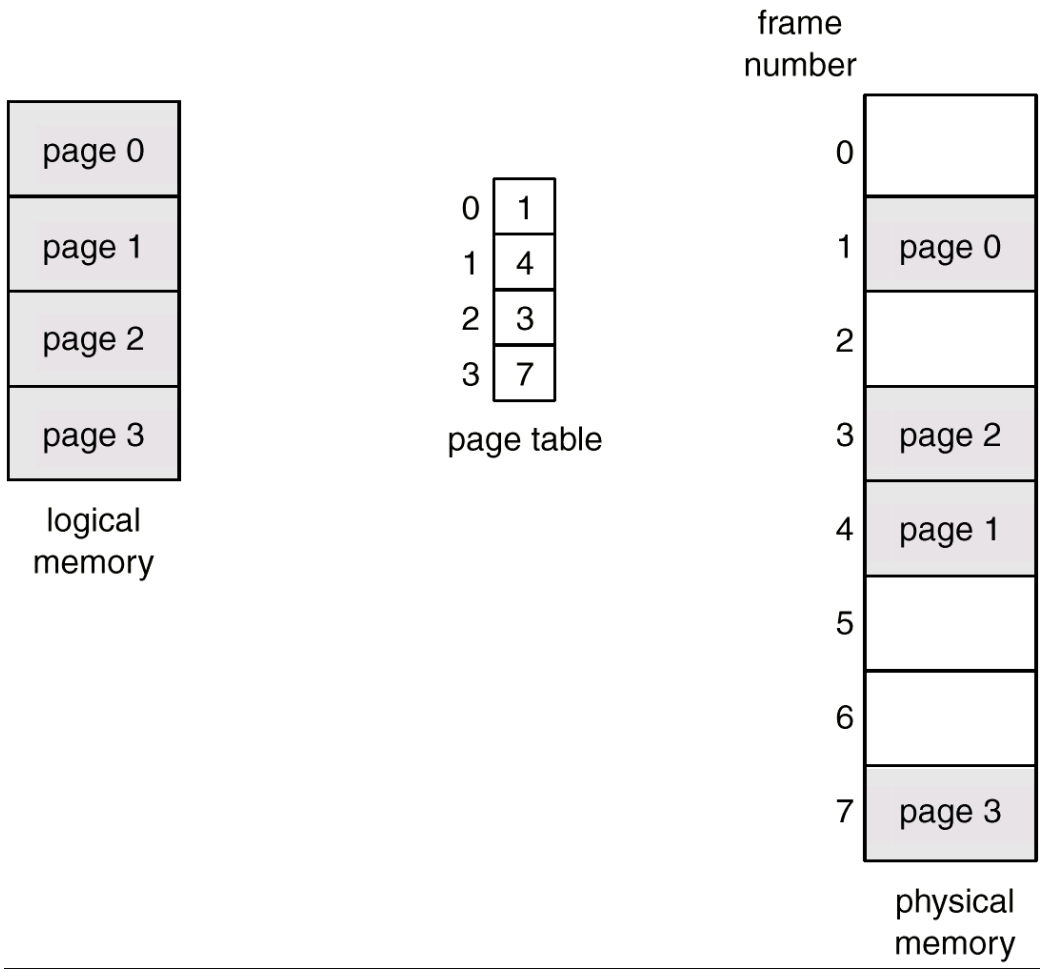
- Всеки логически адрес генериран от централния процесор се разделя на две: *номер на страница (p)* и *отместване в страницата (d)*.
- Номерът на страницата (p) се използва като индекс в *преобразуващата таблица (таблицата на страниците, page table)*, където се съхранява базовия адрес на кадъра, съответстващ на дадената страница.
- Отместването в страницата (d) се добавя към базовия адрес на кадъра за да се получи физическия в оперативната памет.

Преобразуване на адреси



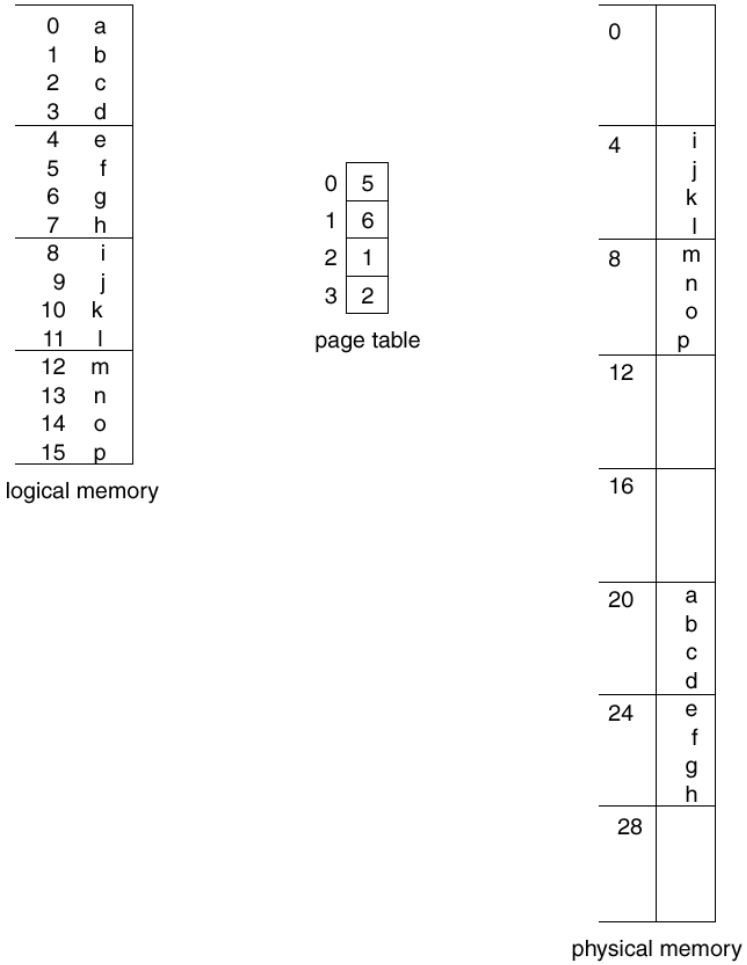
Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Страници



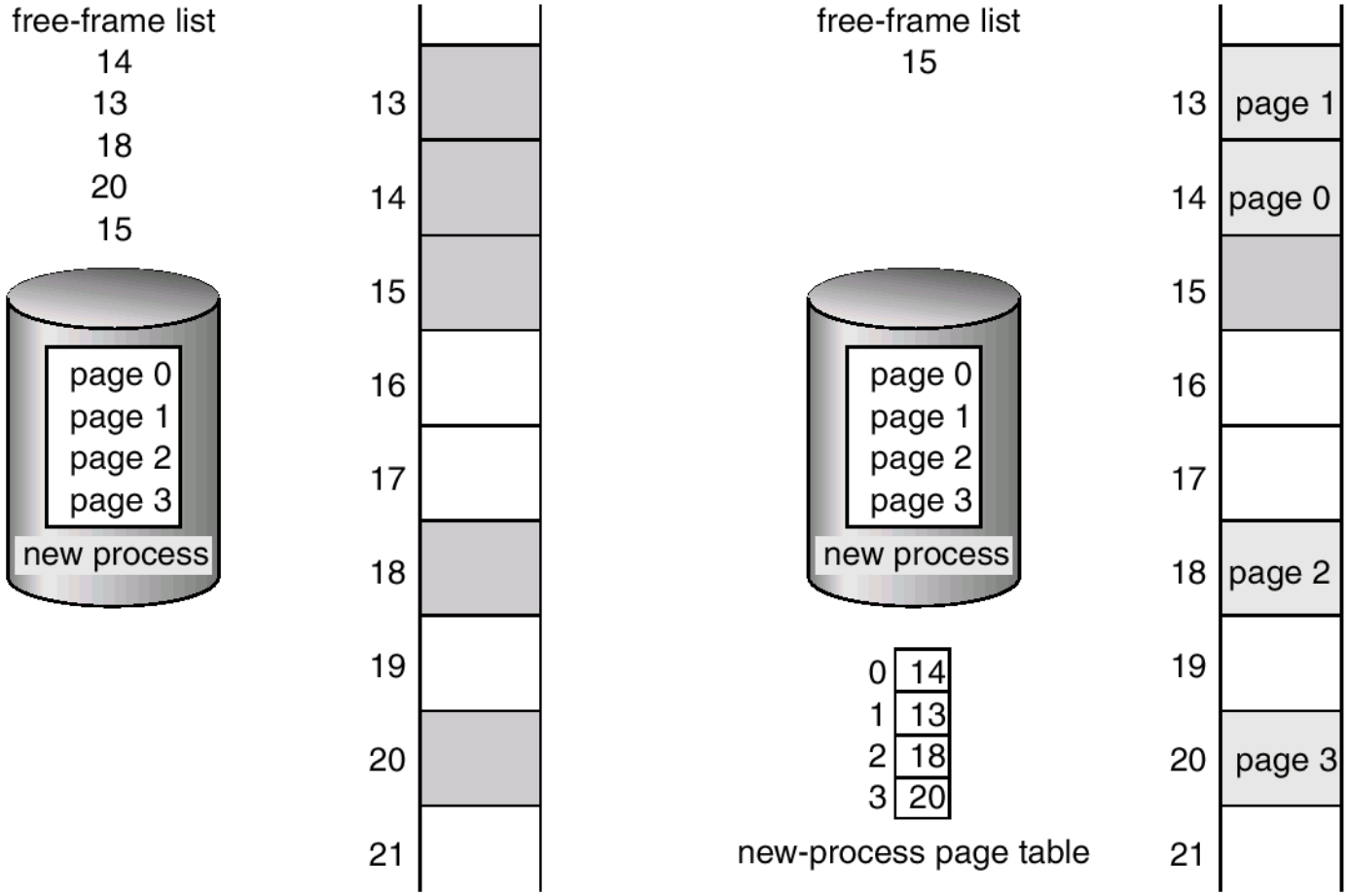
Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Страници



Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Свободни кадри

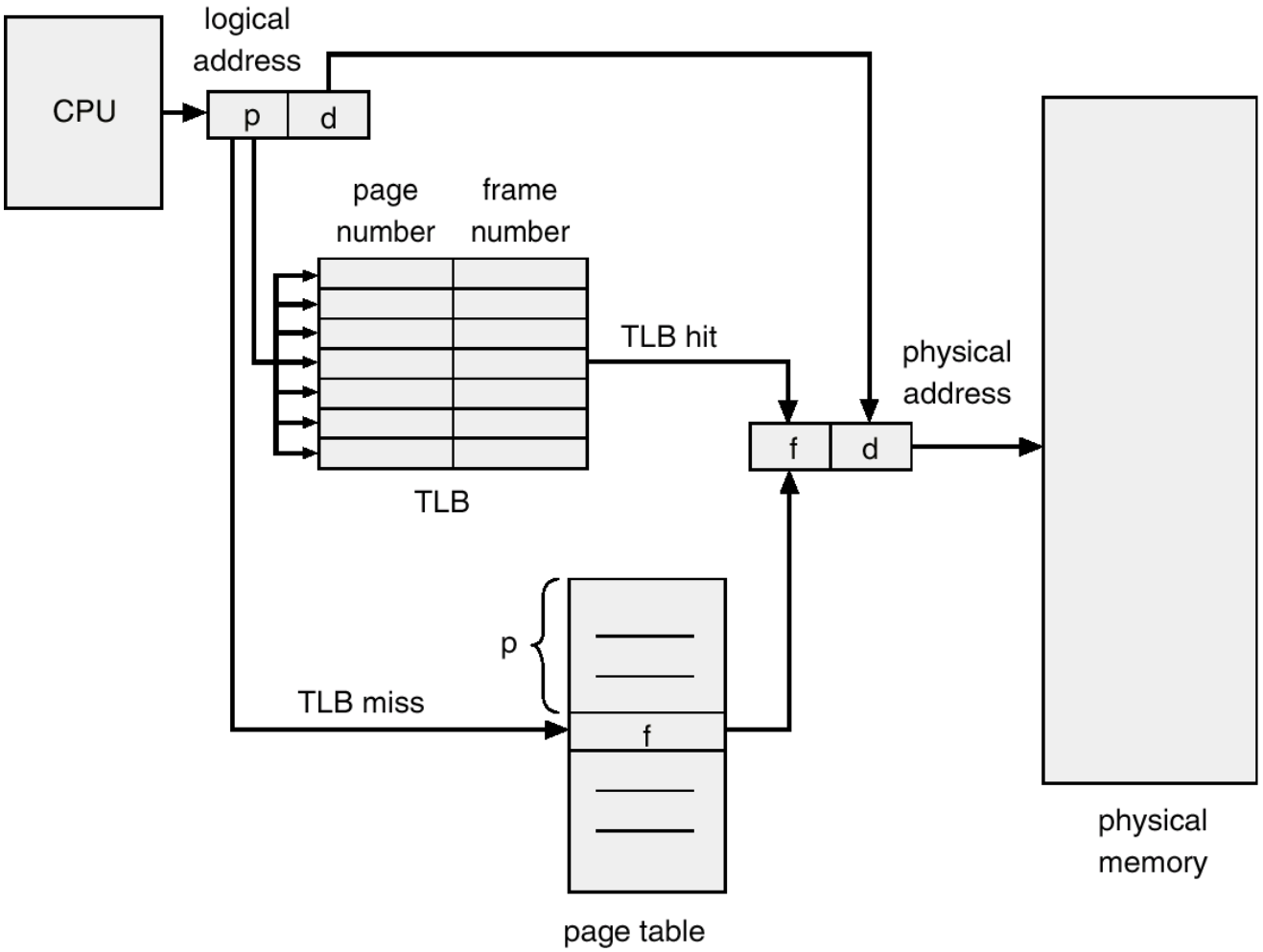


Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Преобразуваща таблица

- Преобразуващата таблица (page table) типично се съхранява в оперативната памет.
- При такъв подход достъпът до данните и инструкциите изисква две обръщания към паметта – веднъж към преобразуващата таблица за да се формира физическия адрес и веднъж към съответния адрес в оперативната памет.
- Този проблем може да бъде решен като се използва специализиран бърз хардуерен кеш, който е организиран като асоциативна памет (associative memory, translation look-aside buffer (TLB)).

Преобразуваща таблица

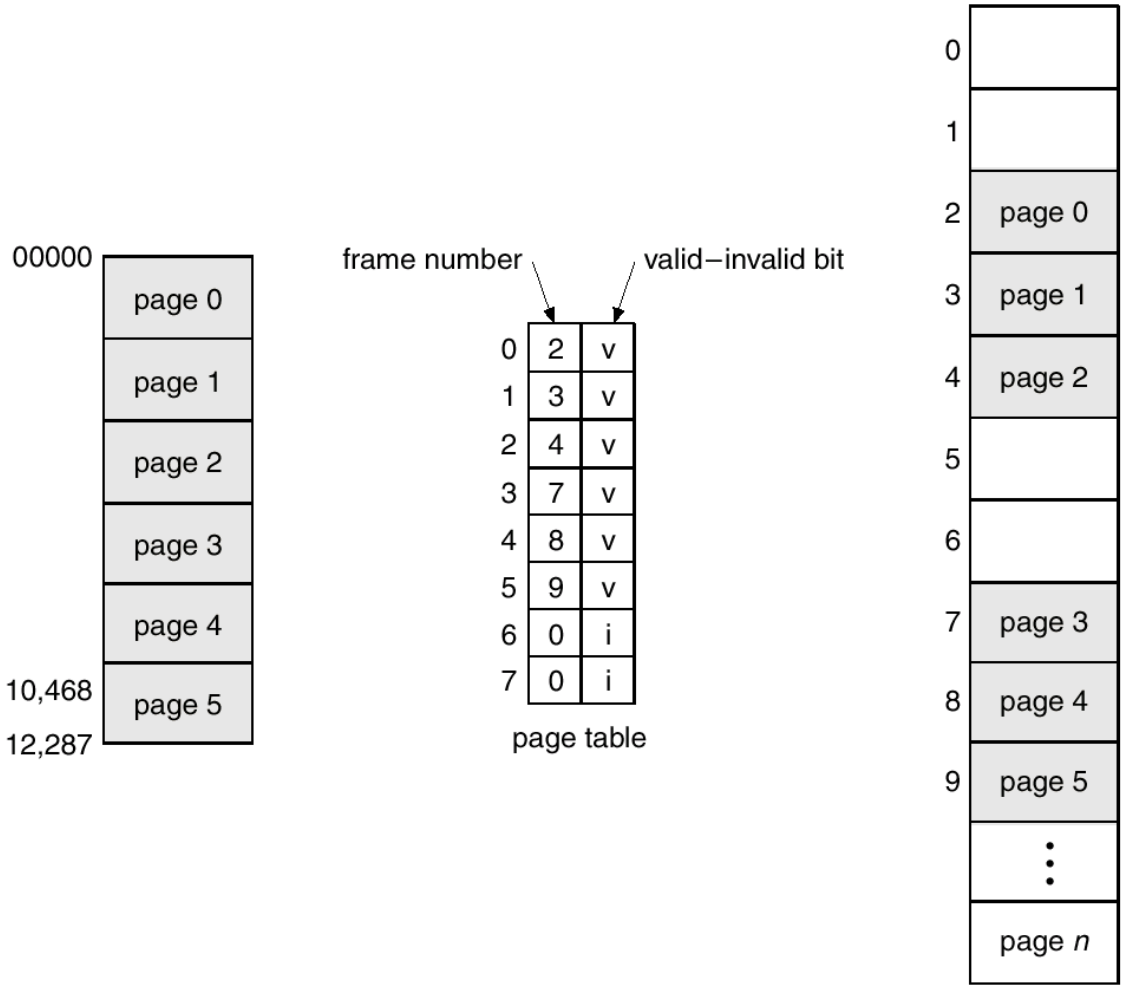


Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Защита на паметта

- Защитата на паметта при деленето на страници се реализира като с всеки кадър се асоциира допълнителен бит (или битове).
- Най-простата схема асоциира с всяка страница в таблицата за преобразуване допълнителен бит за валидност на страницата (valid/invalid bit).
- Когато битът е в състояние "валиден (valid)", то това означава че страницата е в логическото адресно пространство на процеса и следователно може да бъде използвана (адресирана).
- Когато битът е в състояние "невалиден (invalid)", то това означава, че страницата не е в логическото адресно пространство.

Преобразуваща таблица

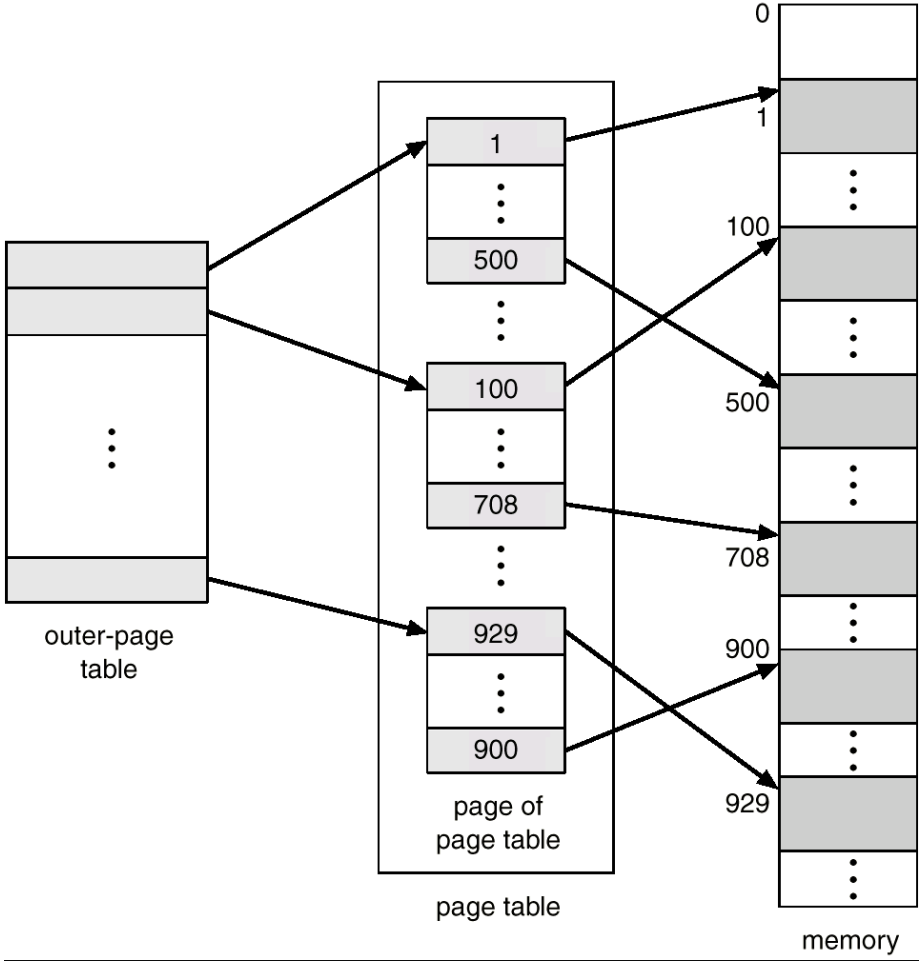


Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Видове преобразуващи таблици

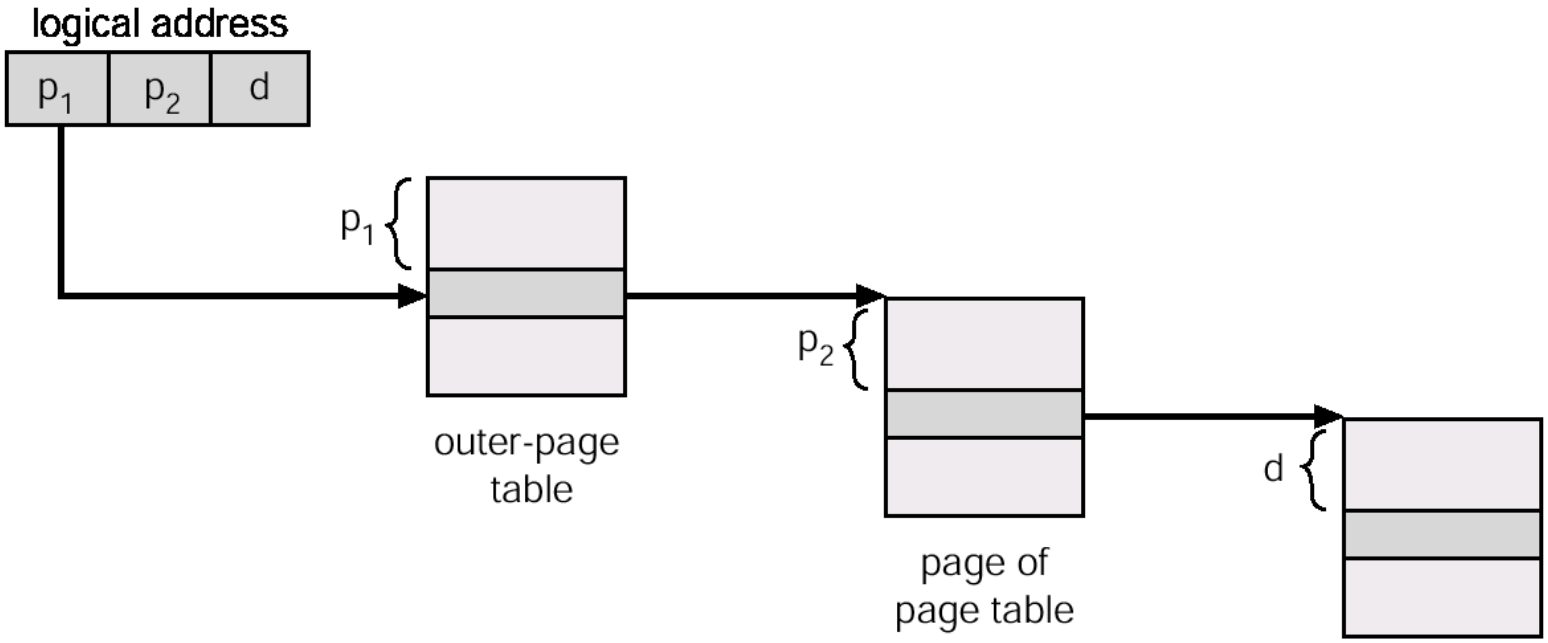
- Йерархични преобразуващи таблици.
- Преобразуващи хеш-таблици.
- Обърнати преобразуващи таблици.

Йерархична преобразуваща таблица с две нива



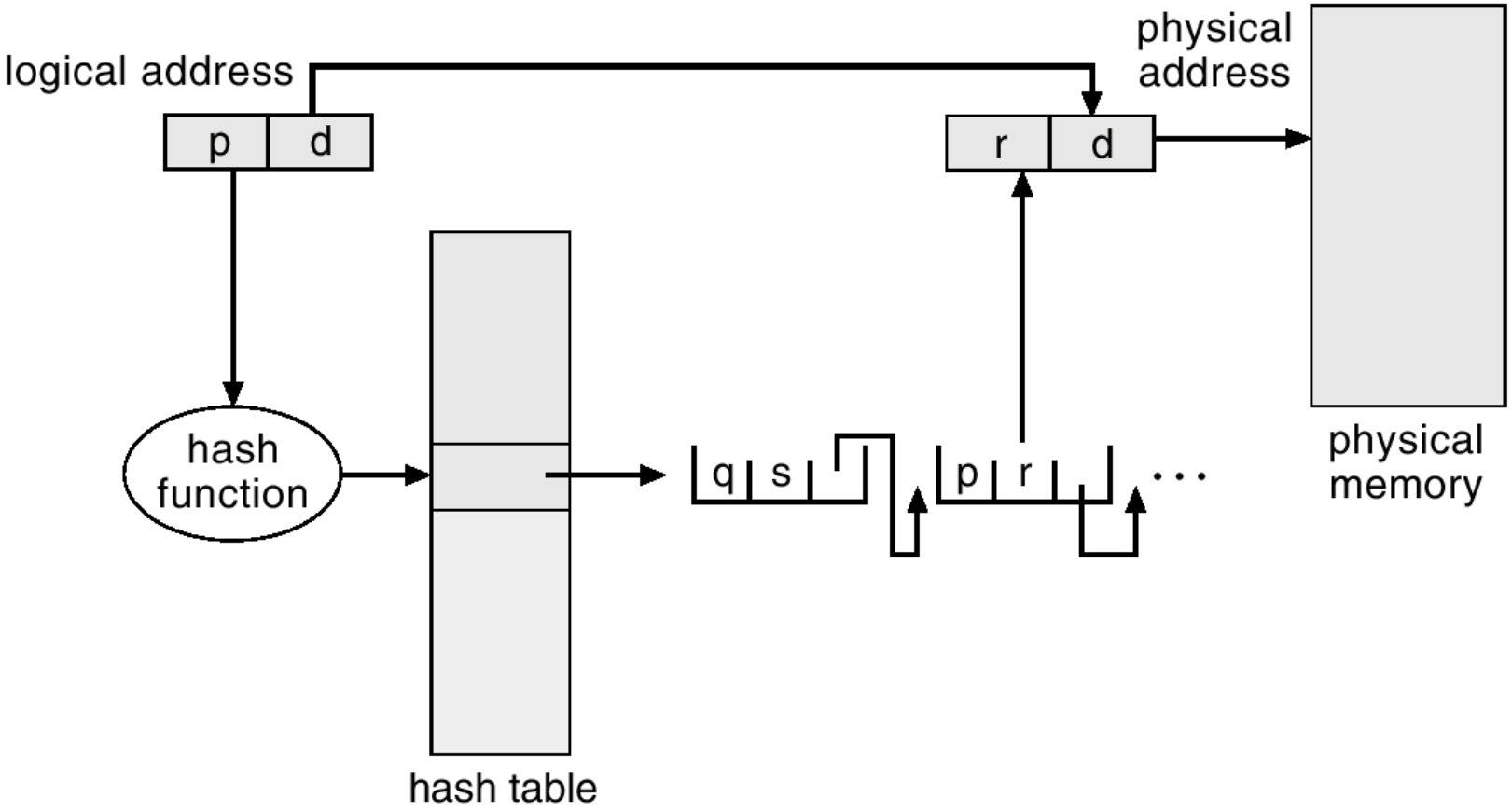
Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Йерархична преобразуваща таблица с две нива



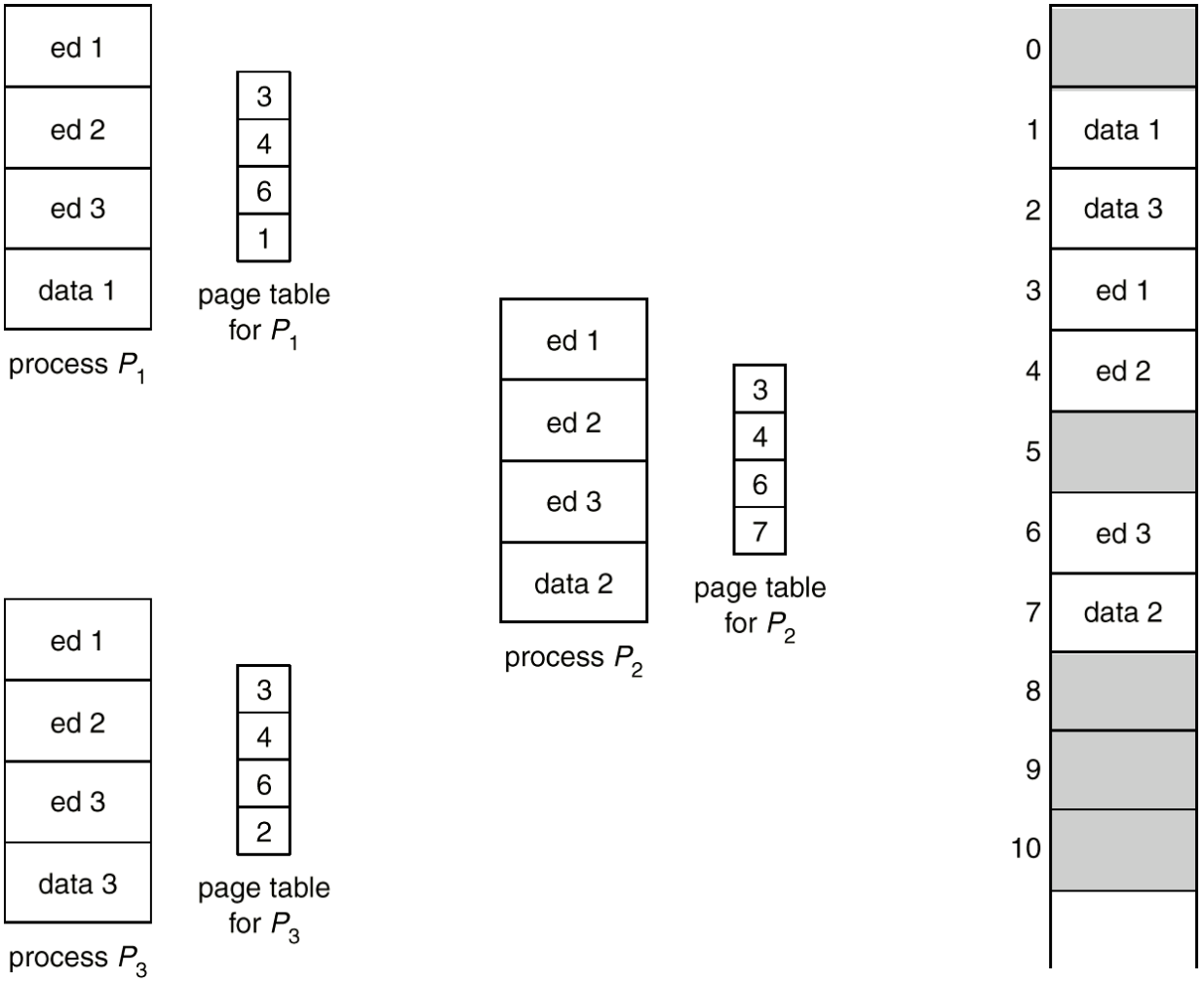
Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Преобразуваща хеш-таблица



Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Общи страници

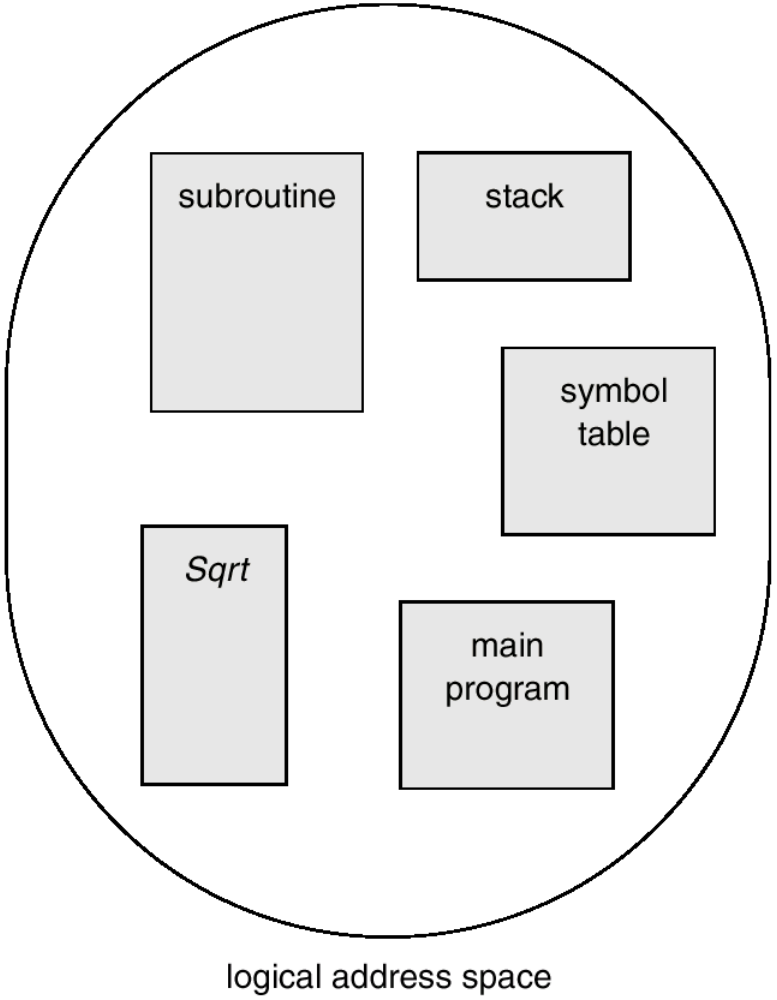


Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Сегментна организация

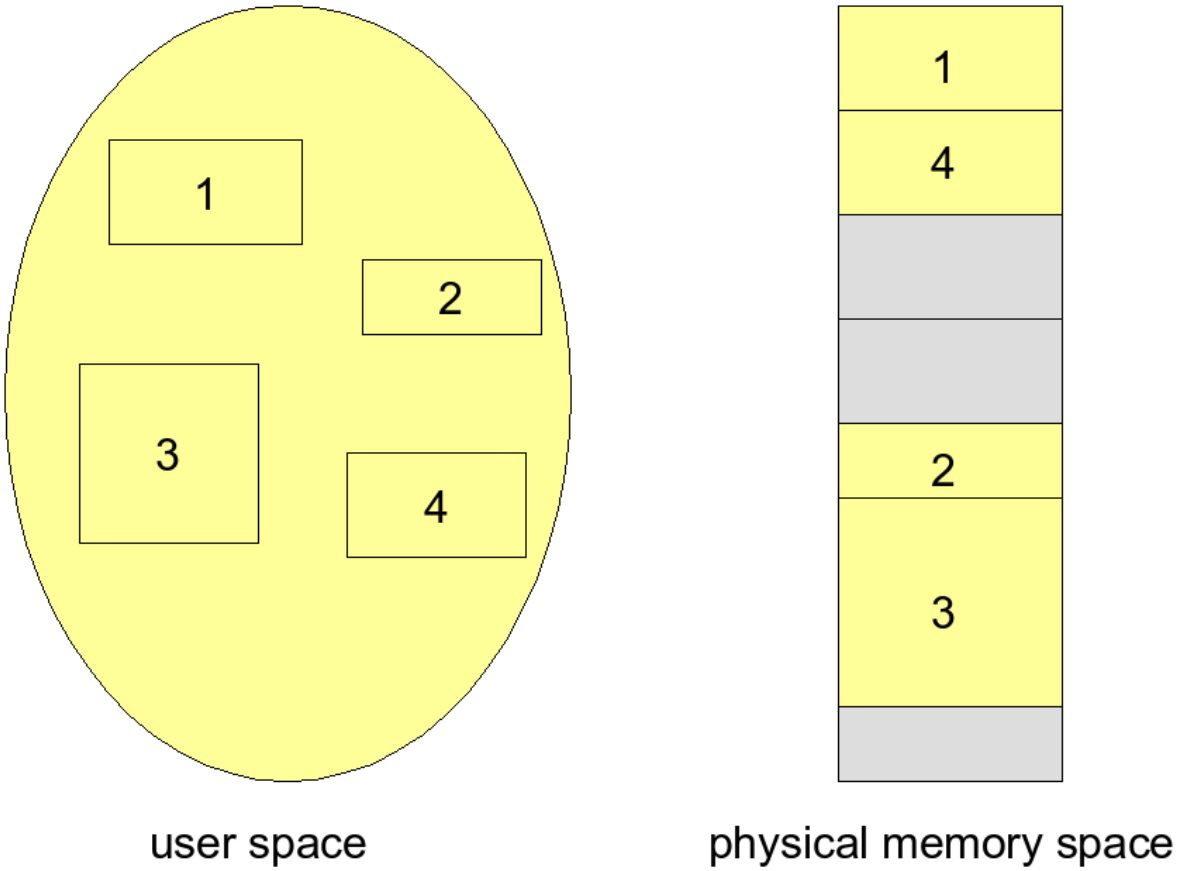
- Сегментната организация на оперативната памет позволява програмата да бъде разположена в няколко несъседни участъка от паметта.
- Сегментната организация е схема за управление на паметта, която поддържа гледната точка на потребителя върху организацията на оперативната памет.
- Програмата е разделена на съвкупност от сегменти. Всеки сегмент е логическа част от програмата – например функция, метод, обект, локални променливи, глобални променливи, масив и т.н.
- Това, че програмата може да бъде разделена на сегменти, позволява да се намали външната фрагментация.

Сегментна организация



Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Сегментна организация



Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Сегментна организация

- Логическият адрес при сегментната организация се състои от двойката $< s, d >$, където s е номер на сегмент, а d е отместване в рамките на сегмента.
- За изобразяване на логическите адреси във физически се използва *таблица на сегментите*.
- Всеки елемент на таблицата на сегментите съдържа *базов адрес* и *граница*. Базовият адрес е началния адрес във физическата памет, от който е разположен сегмента. Границата съдържа размера на сегмента.

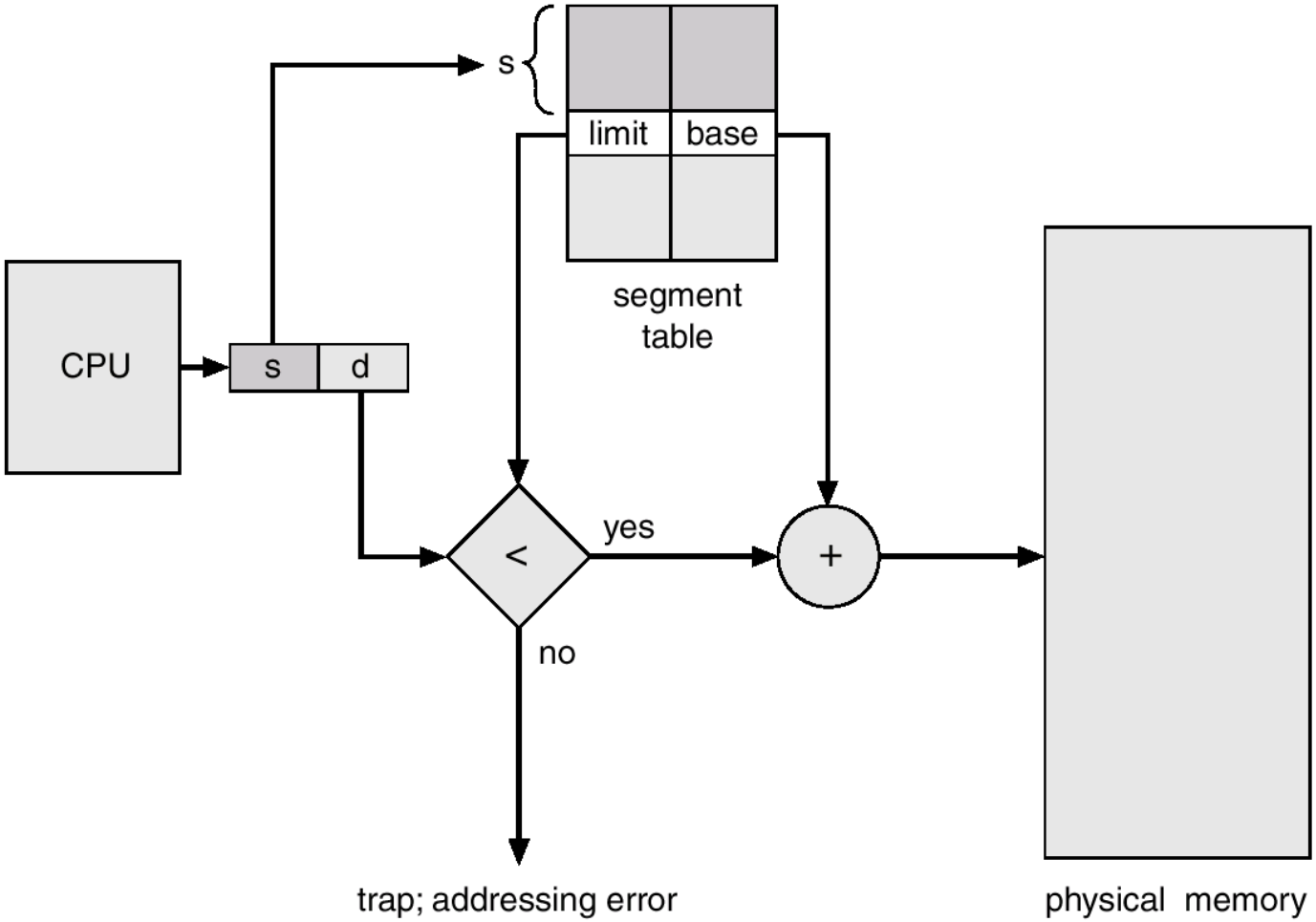
Сегментна организация

- Таблицата на сегментите е разположена в оперативната памет.
- Регистърът за начален адрес на таблицата на сегментите (segment-table base register - STBR) сочи към началото на таблицата на сегментите.
- Регистърът за размера на таблицата на сегментите (segment-table length register - STLR) съдържа броя на сегментите, които се използват от процеса. Номерът на сегмент, генериран от процесора е валиден, ако е по-малък от STLR.

Сегментна организация: защита на паметта

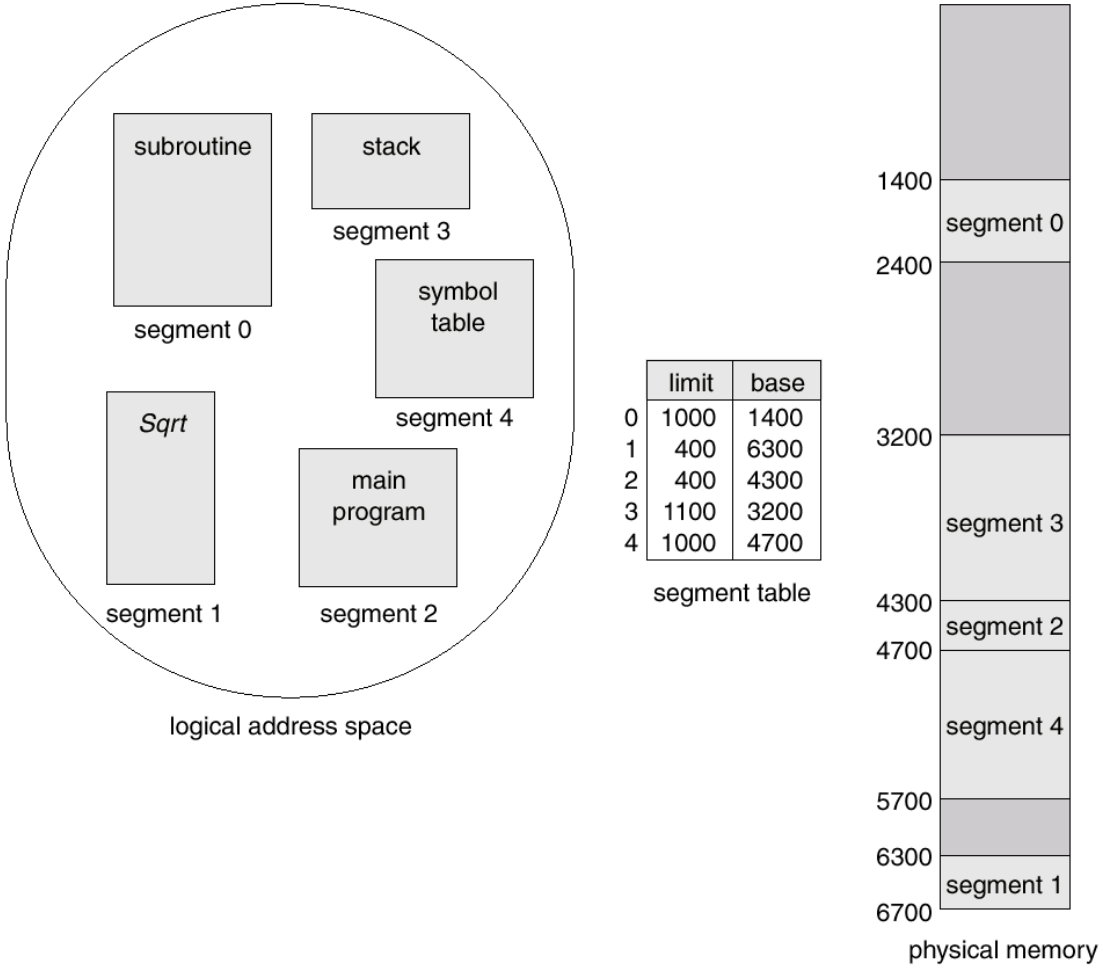
- Сегментите отразяват логическите части на програмата – код на програмата, програмен стек, данни и т.н.
- Към всеки елемент от таблицата на сегментите е асоциран набор от допълнителни битове, предназначени за контрол на достъпа до съответния сегмент.
- С всеки сегмент се свързват определени права за достъп – права за четене, права за писане и права за изпълнение.
- Например: на сегмент, който съдържа изпълнимия код на програмата, се назначават права само за изпълнение. Четенето и писането в този сегмент се забраняват.

Сегментна организация



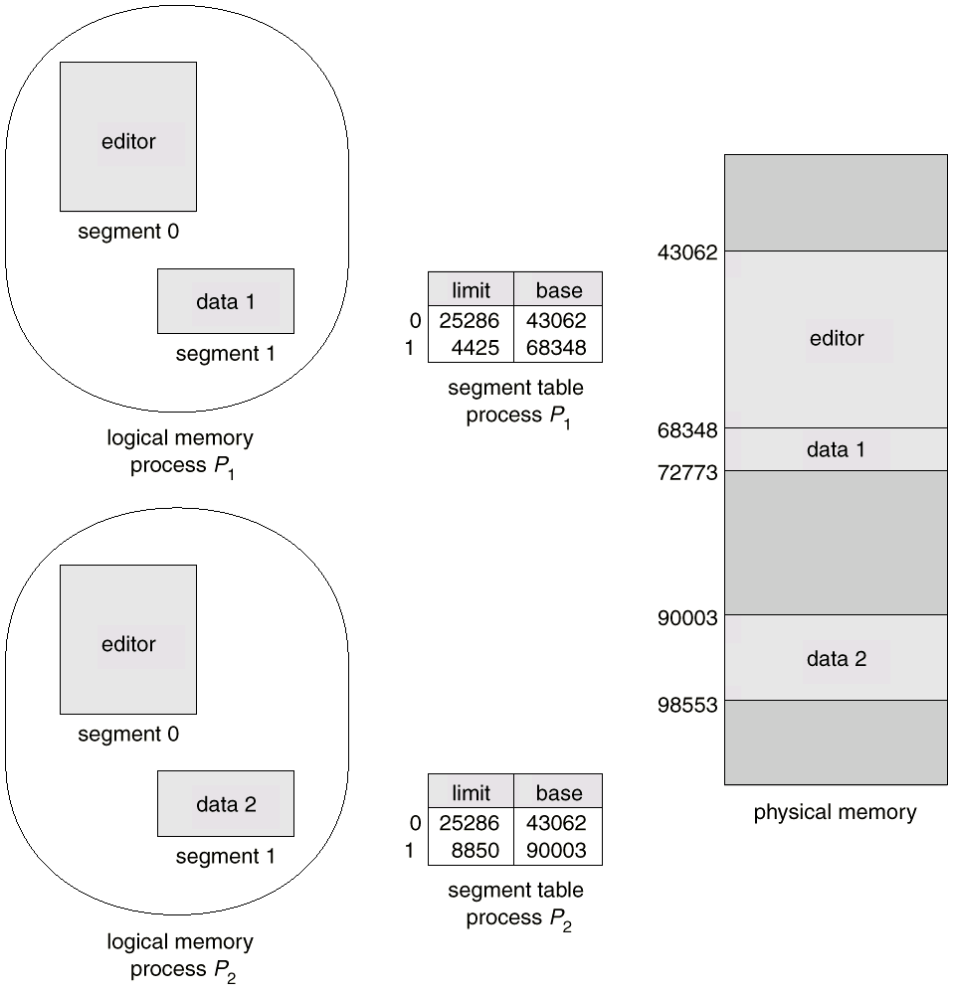
Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Сегментна организация



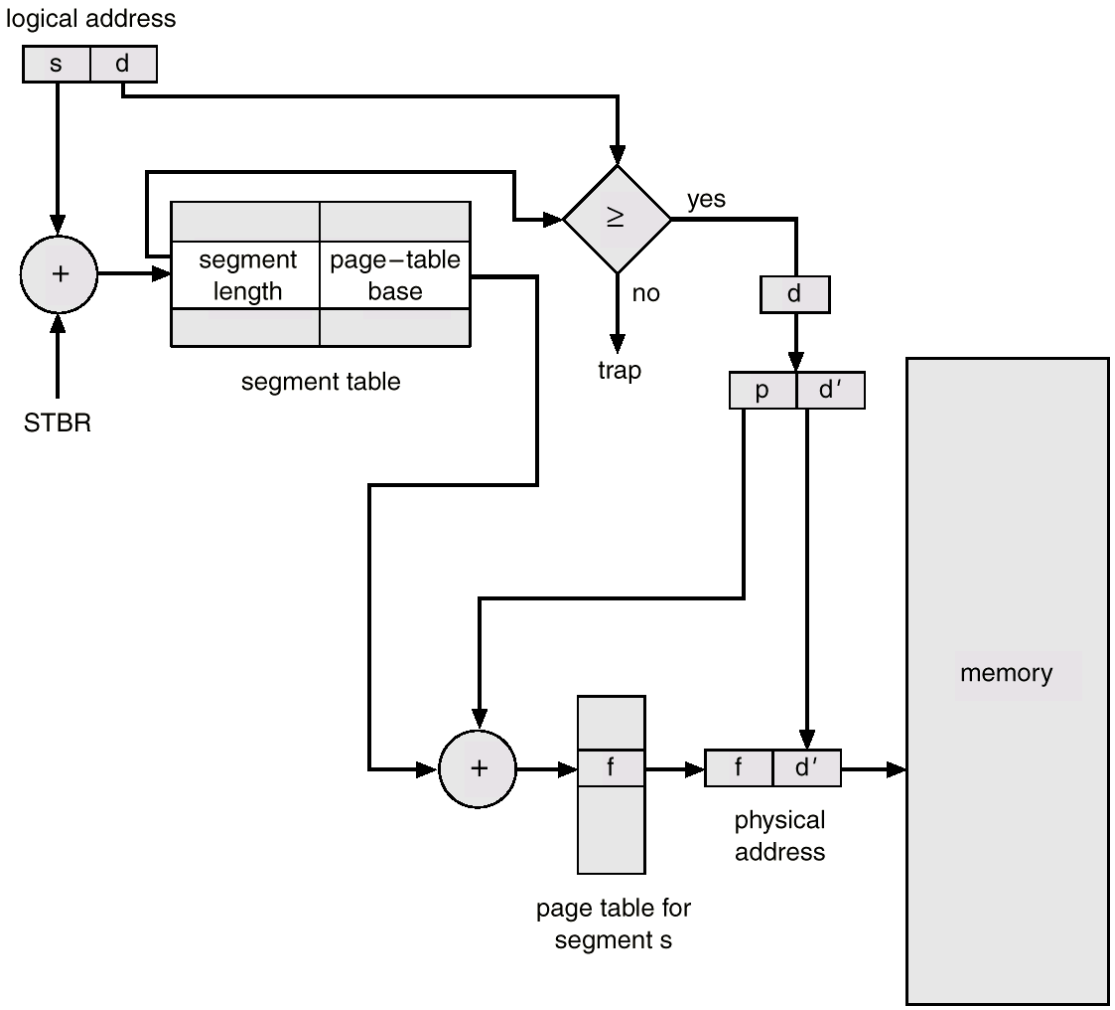
Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Сегментна организация: общи сегменти



Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Смесена организация: странично-сегментна

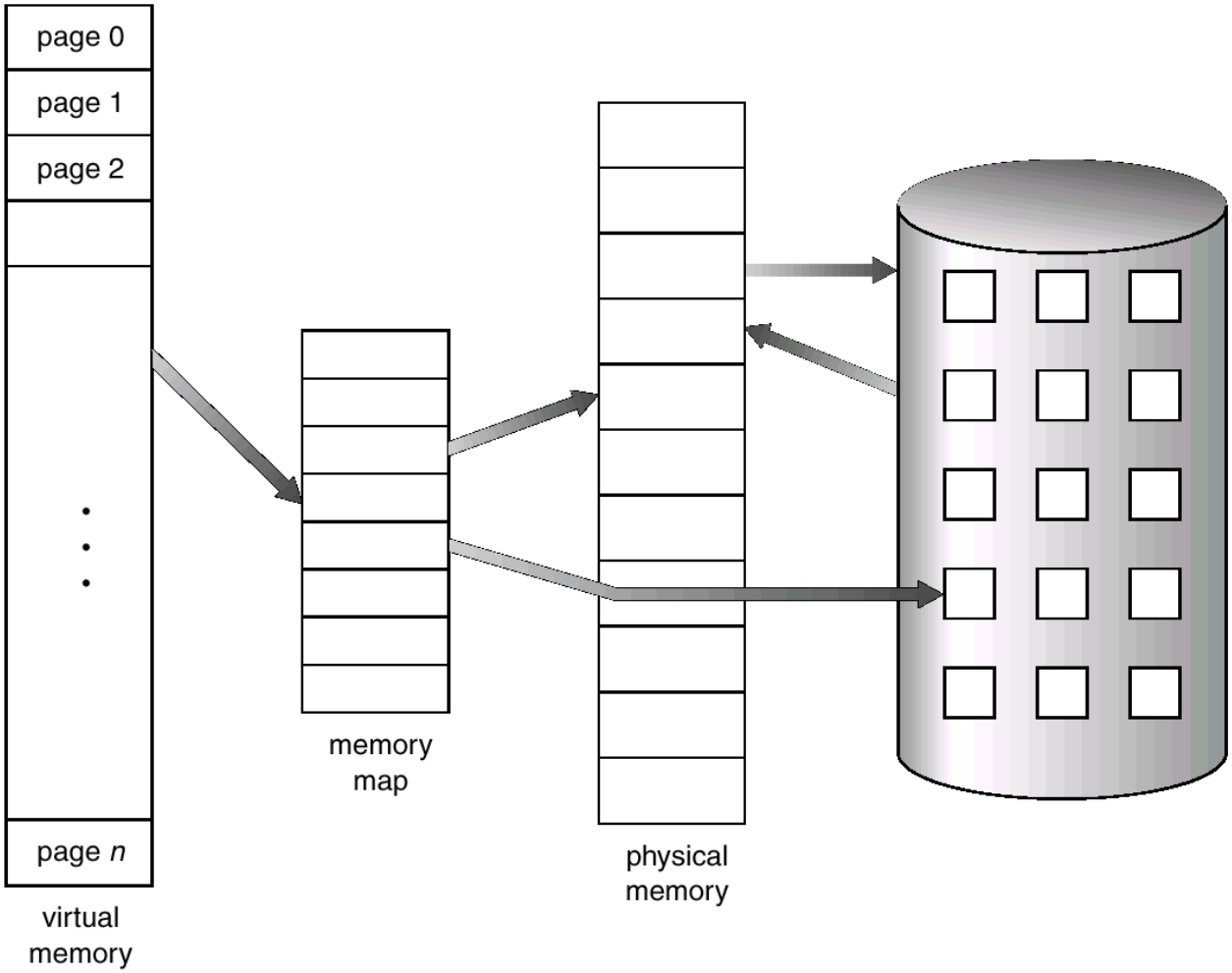


Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Виртуална памет

- *Виртуална памет* се нарича концепцията за разделяне на логическото адресно пространство от физическото адресно пространство.
- При изпълнение на дадена програма е възможно в даден момент в оперативната памет да се намира само част от програмата. Това позволява логическото адресно пространство да е по-голямо от физическото.
- Виртуалната памет позволява част от адресното пространство да е общо за няколко процеса – общи страници, общи сегменти.

Виртуална памет

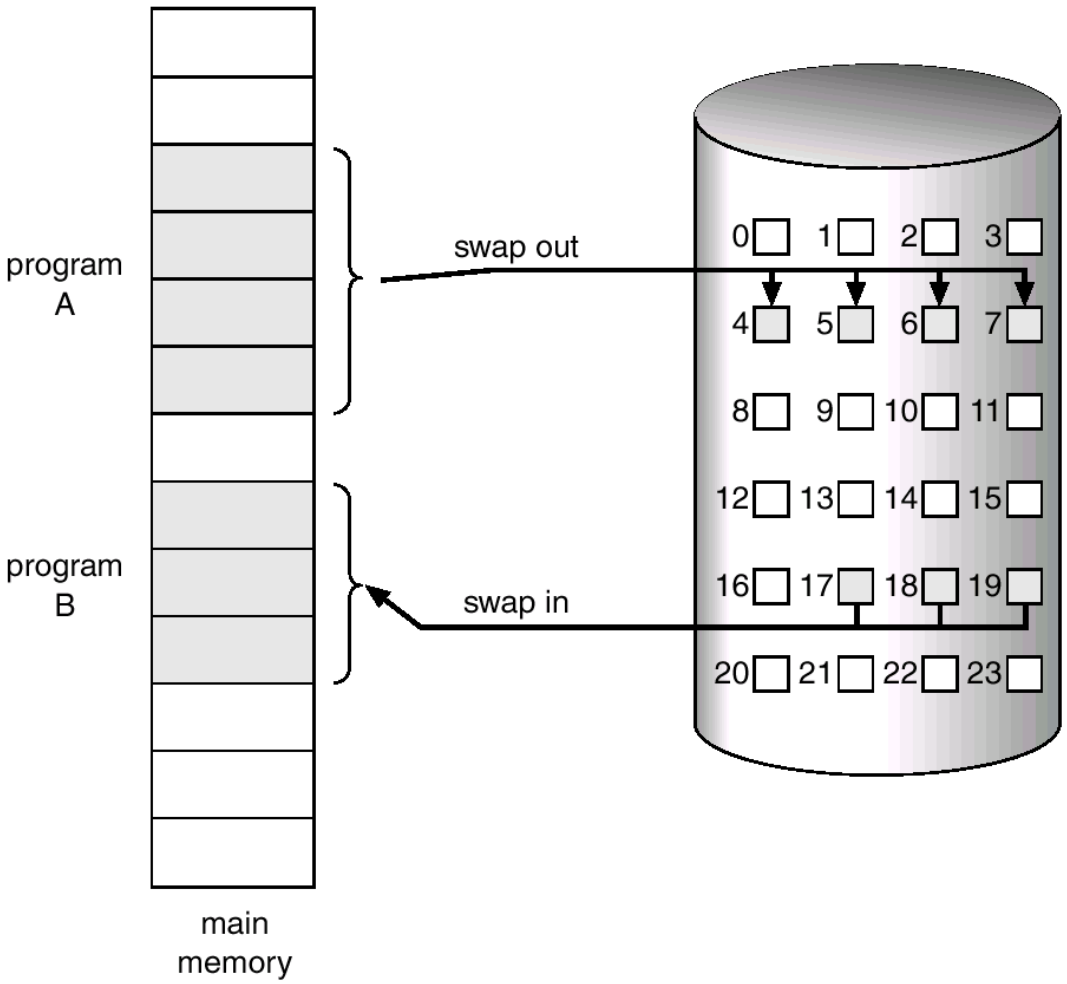


Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Зареждане на страница при поискване

- Дадена страница се зарежда в оперативната памет, само в случай, че действително има нужда от нея.
- Този подход спестява входно/изходни операции при зареждане на програмата, нужна е по-малко оперативна памет, системата реагира по-бързо на заявките на потребителя.
- Когато дадена страница е необходима, то се генерира обръщане към нея. Когато адреса е невалиден, то това води до генериране на прекъсване. Ако страницата не е в оперативната памет, то страницата първо се зарежда.

Виртуална памет

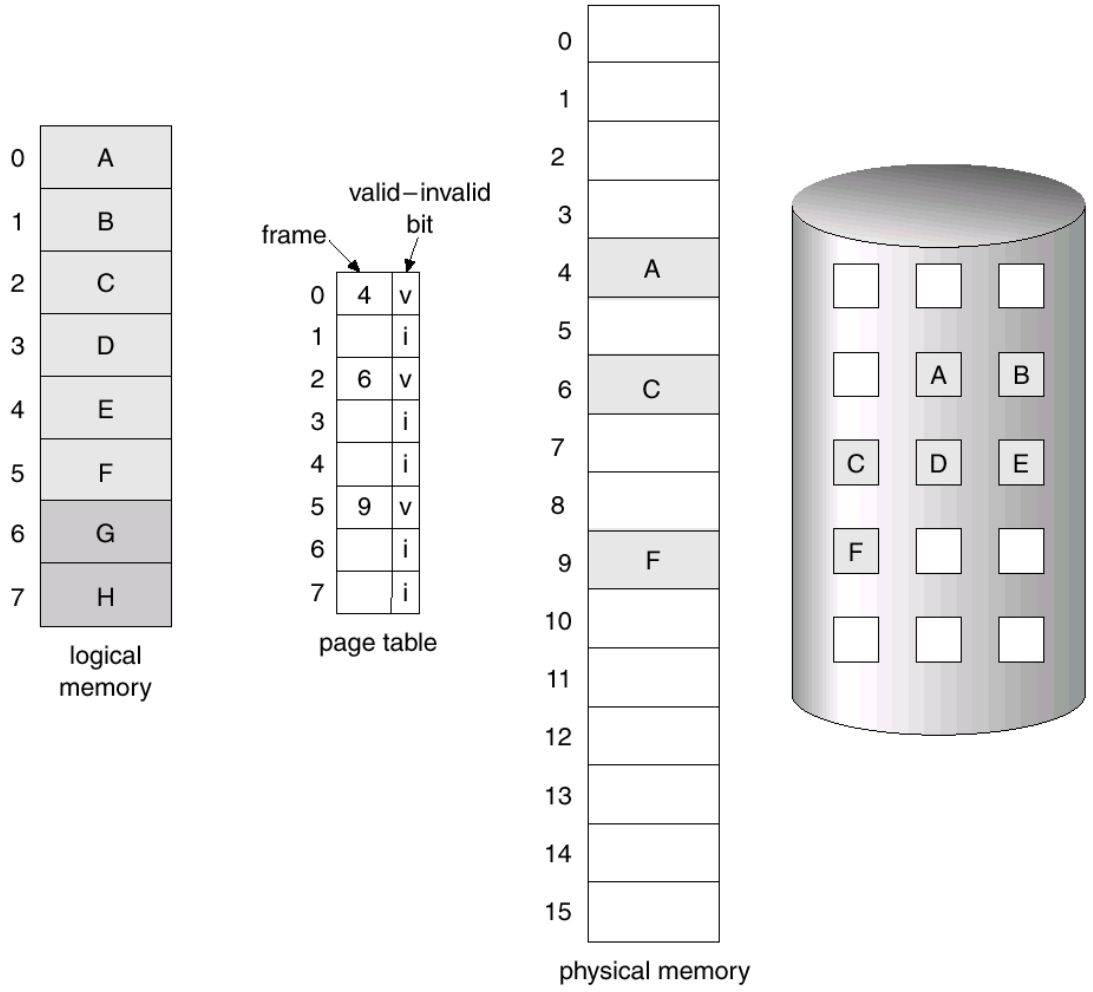


Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Бит за валидност/невалидност на страницата

- С всеки елемент на таблицата за преобразуване на адресите се асоциира бит за валидност/невалидност на страницата.
- Първоначално за всеки елемент на таблицата за преобразуване на адресите, асоциирания бит за валидност/невалидност се установява в стойност “невалиден”.
- При преобразуването на адреси, когато бита за валидност/невалидност е със стойност “невалиден” се генерира прекъсване “грешна страница”.

Виртуална памет

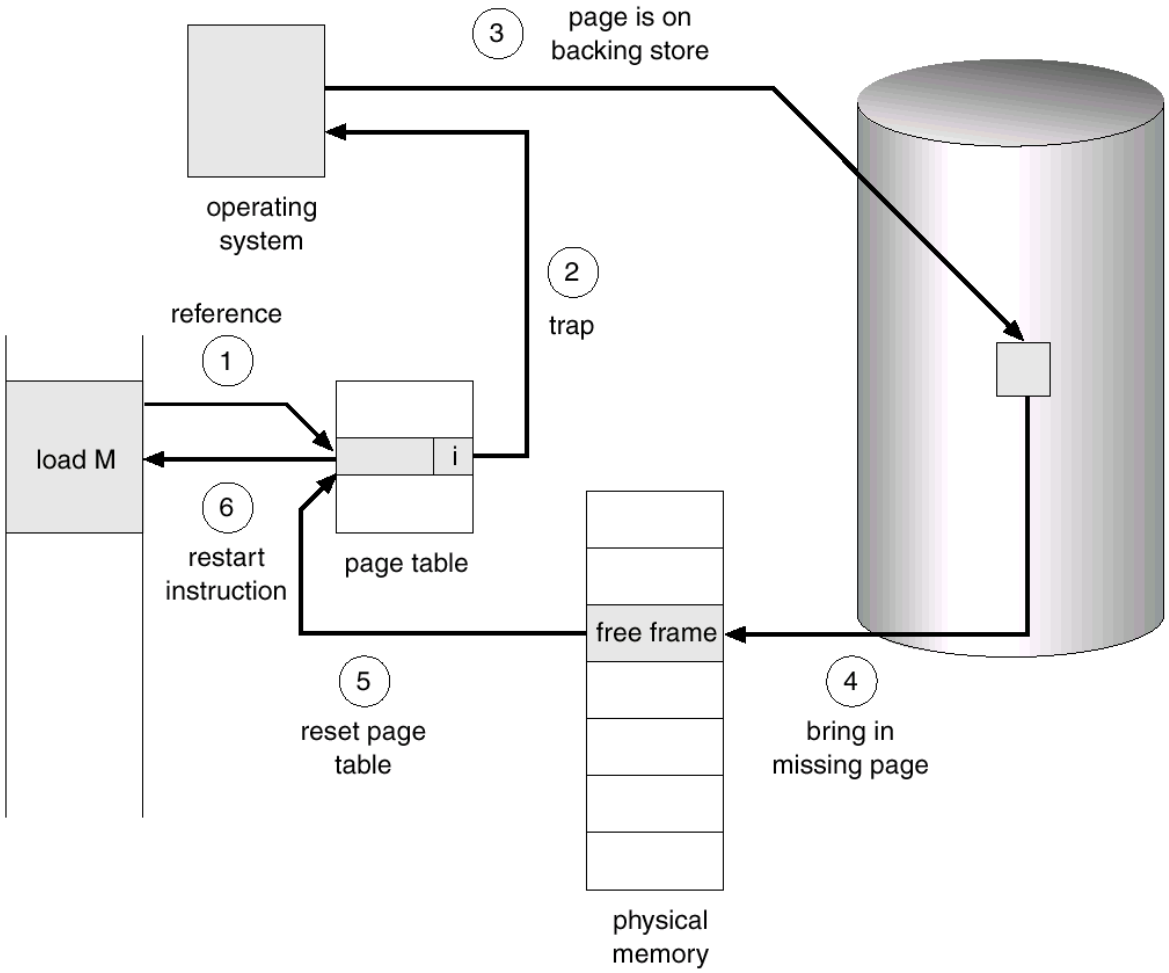


Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*

Прекъсване “грешна страница (page fault)”

- При първото обръщане към дадена страница винаги се генерира прекъсване “грешна страница”.
- Операционната система трябва да различи двете възможни причини за генерираното прекъсване – невалиден адрес или страницата не е в оперативната памет. Ако адреса е невалиден, то процеса се спира.
- Ако причината за прекъсването е, че страницата не е в оперативната памет, то операционната система трябва да намери празен кадър, да зареди страницата в този кадър, да обнови таблицата за преобразуване на адреси и да изпълни повторно прекъсната инструкция.

Прекъсване “грешна страница (page fault)”



Фигура: Silberschatz, Gavin, Gagne: *Operating Systems Concepts*