# Организация памяти мультипрограммных ЭВМ

1. Динамическое распределение и фрагментация основной памяти (ОП).
2. Сегментная организация памяти.
3. Сегментно-страничная организация памяти.
4. Концепция виртуальной памяти.

# Организация памяти мультипрограммных ЭВМ

- **Знать:** методы преобразования математического адреса в физический для сегментной, страничной и сегментно-страничной организации ОП; способы борьбы с фрагментацией ОП, методы замещения страниц в ОП; концепцию виртуальной памяти.

# Организация памяти мультипрограммных ЭВМ

**Помнить:** о "пробуксовывании" вычислительной системы при использовании виртуальной памяти.

**Уметь:** разработать аппаратно-микропрограммные средства для преобразования математического адреса в физический.

***Литература:***

- Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем. Учебник для вузов. – СПб.: Питер, 2004. – 668 с. (с. 387-412)

# *1. Динамическое распределение и фрагментация памяти*

- Под программой в ОП будем понимать весь объем информации, необходимой для выполнения программы (рисунок 1), полагая, что выполняемая программа должна полностью размещаться в ОП.


Рисунок 1 – Представление программы

# *Фрагментация памяти*

- При динамическом распределении памяти (рисунок 2) по мере завершения программ (P2, P4) на их место система динамического распределения памяти загружает с ВЗУ новые программы (P7, P8). При этом могут оставаться свободные участки памяти – фрагменты. В результате возникают ситуации, подобные той, что изображена на рисунке 2, которые приводят к фрагментации ОП.

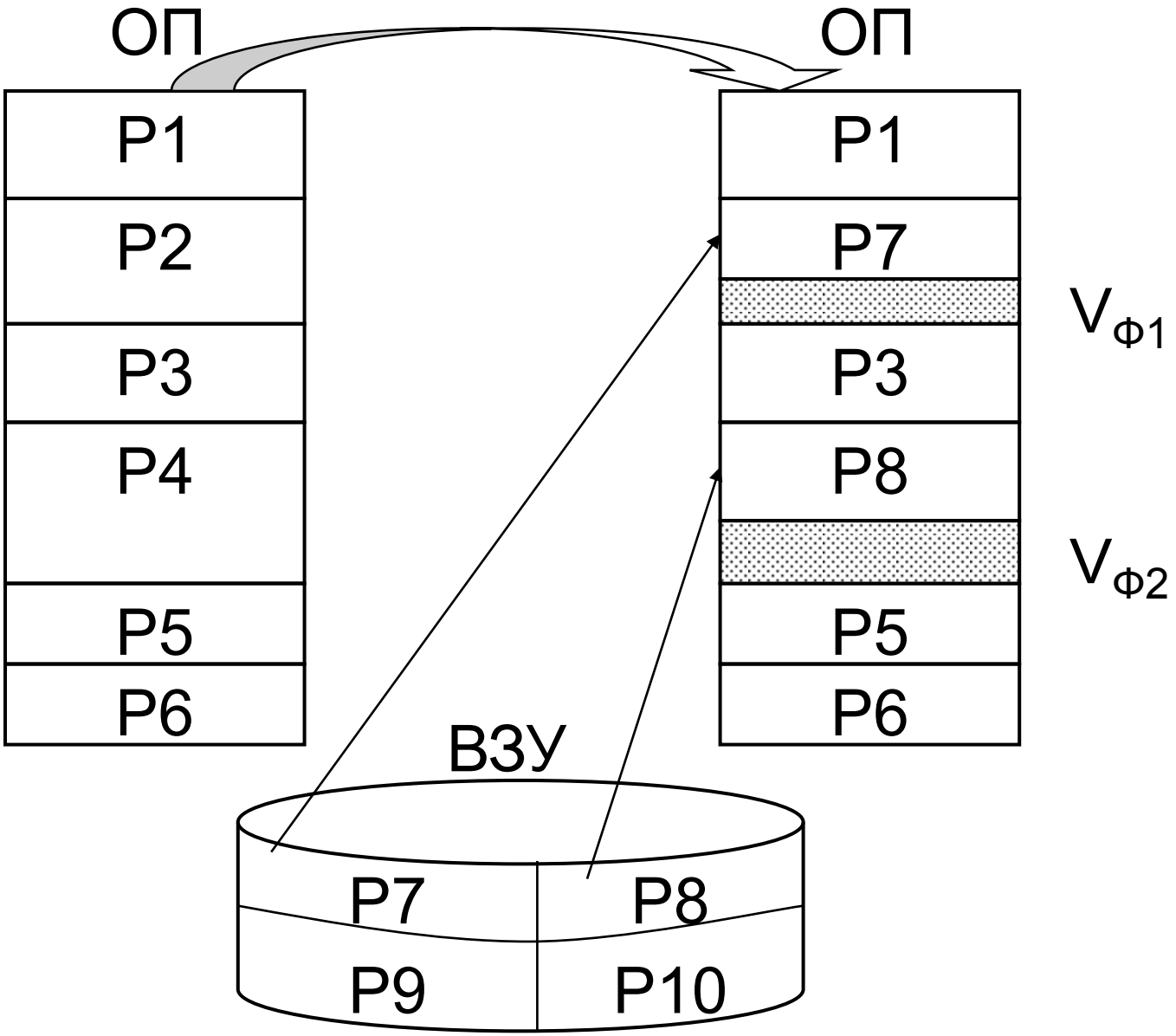


Рисунок 2 – Образование фрагментов в ОП

# *Фрагментация памяти*

- На рисунке 2 представлена ситуация, когда суммарный объем свободной памяти превышает объем памяти, необходимой любой из двух готовых к выполнению программ (V[P9] и V[P10]), находящихся в ВЗУ. Однако ни одна из них не может быть загружена в ОП т.к. ее объем оказывается больше объема любого свободного фрагмента (VФ1,VФ2):
- VФ1+VФ2>V[P9], VФ1+VФ2>V[P10];
- V[P9]>VФ1, V[P9]>VФ2;
- V[P10]>VФ1, V[10]>VФ2.

# Способы борьбы с фрагментацией ОП и увеличения коэффициента мультипрограммирования

- Использование специальных программ – сборщиков мусора.
- Сегментация программ. В этом случае снимается ограничение состоящее в том, что программа должна находится в связной (единой) области основной памяти.
- Страничная или сегментно-страничная организация памяти.

# *2. Сегментная организация памяти*

- Программы разделяются на логически завершенные части обычно различных размеров, называемые сегментами. Эти сегменты размещаются в основной памяти  не в непрерывной области, а в произвольных свободных областях (рисунок 3).

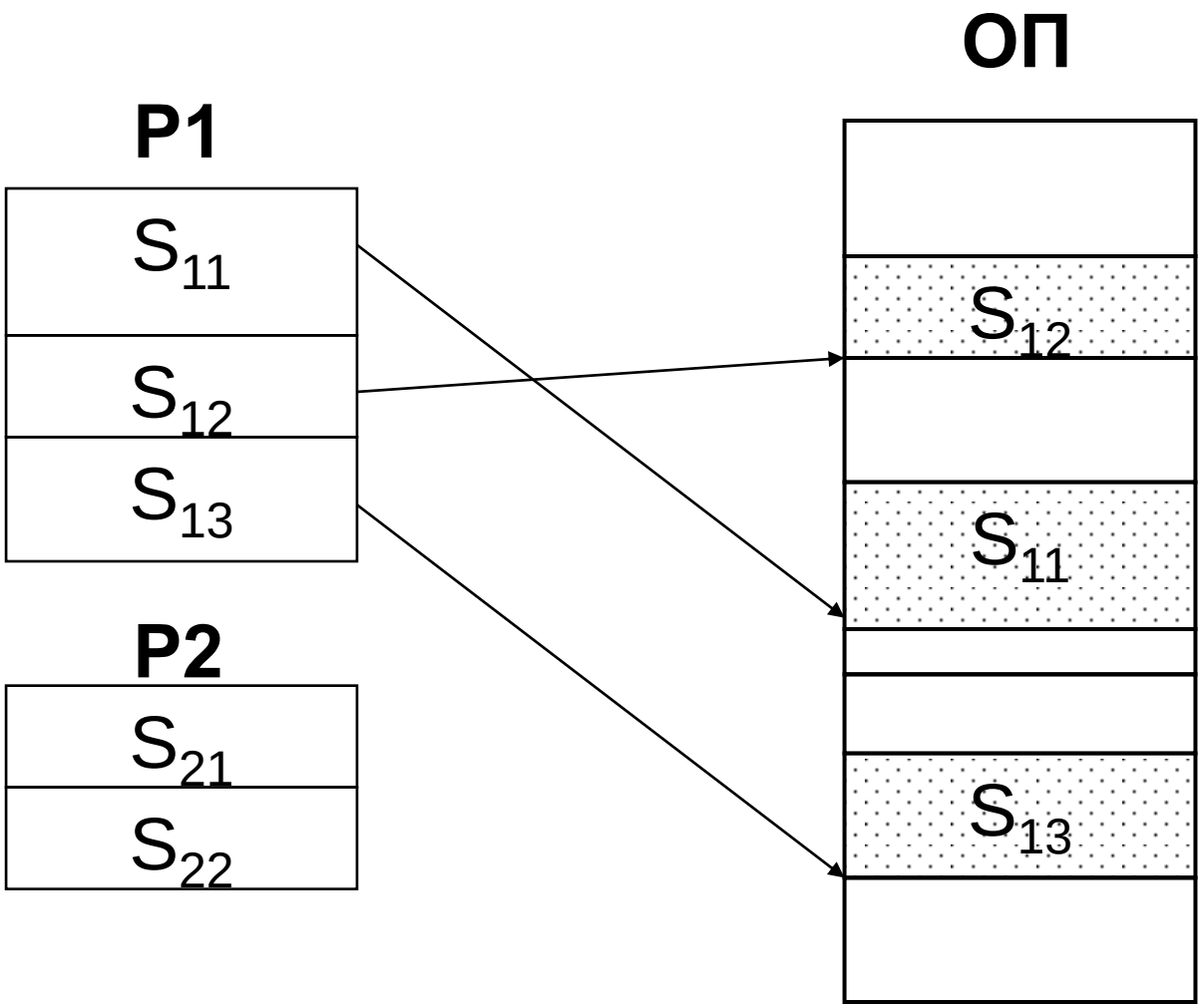


Рисунок 3 – Сегментация программ

# Сегментная организация ОП

- При сегментной организации снимается ограничение, требующее размещать в ОП всю программу в непрерывной области памяти. В результате в ОП может быть загружено больше программ, что повышает коэффициент мультипрограммирования и производительность вычислительной системы.
- Сегментная организация ОП предполагает использование логических (математических) и физических адресов.

# Логические и физические адреса

- Логические адреса, используемые в программе с номером NPR, указываются в кодах команд. Логический адрес состоит из номера сегмента NS и адреса объекта в сегменте D.
- Физический адрес вычисляется как сумма базы B (начального адреса выбранного сегмента в ОП) и адреса объекта в сегменте D: AF=B+D,
- где база B является функцией номера программы NPR и номера сегмента NS: $B=\varphi(NPR,NS)$.
- Функция $\varphi$ обычно вычисляется табличным способом.

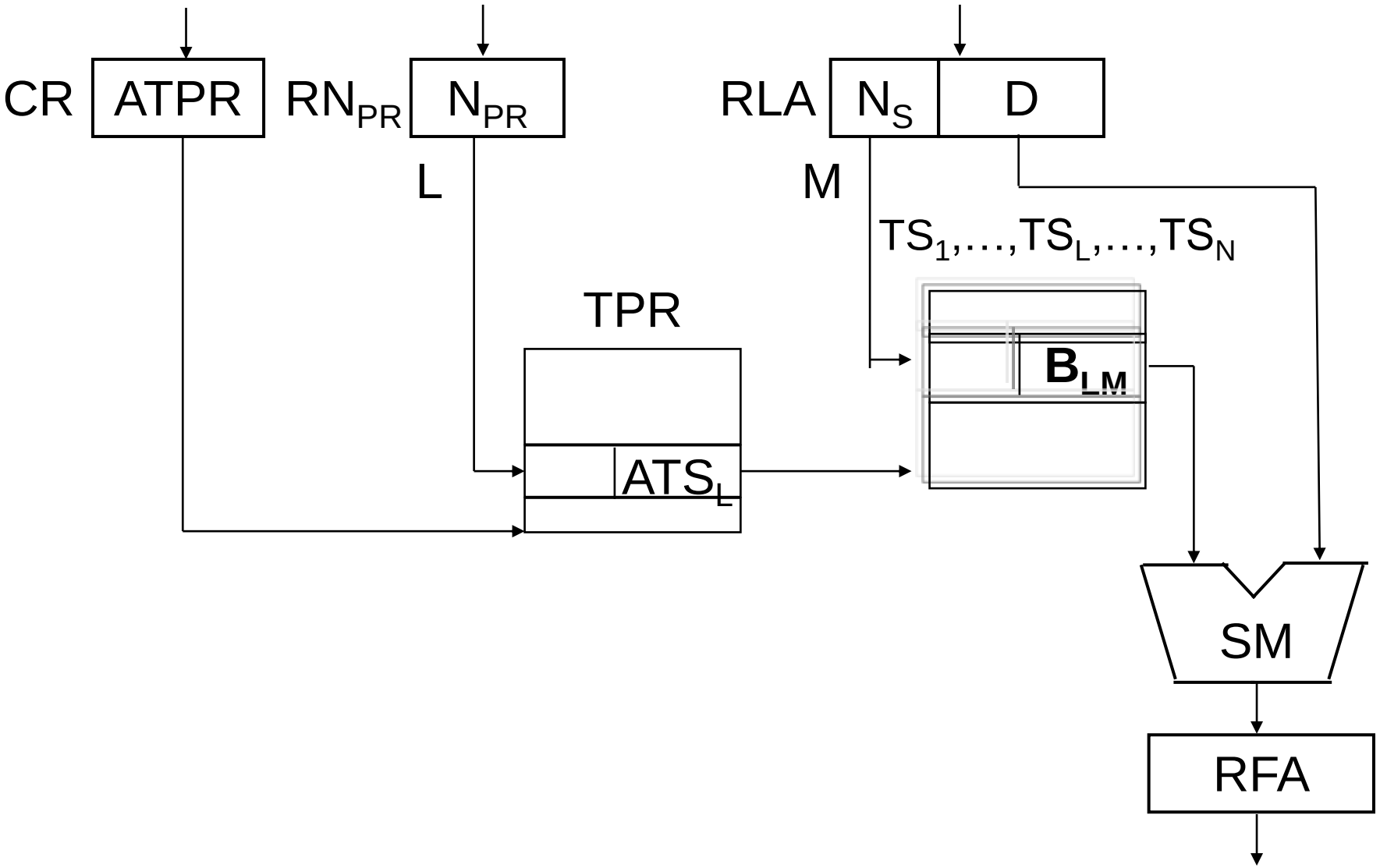


Рисунок 4 – Преобразование логического адреса в физический

# Преобразование адреса

CR – управляющий регистр;

ATPR – начальный адрес таблицы программ;

TPR – таблица программ;

$RN_{PR}$ – регистр номера выполняемой программы;

$N_{PR}$ – номер программы (в примере $N_{PR}$=L);

RLA – регистр логического адреса (в примере $N_S$=M);

$ATS_L$ – адрес таблицы сегментов L-й программы;

$TS_1,\ldots,TS_L,\ldots,TS_N$ – таблицы сегментов программ;

$B_{LM}$ – базовый адрес M-го сегмента, в L-й программе;

KR – кэш-регистр для быстрого преобразования логического адреса в физический;

RFA – регистр физического адреса;

LS – логическая схема.

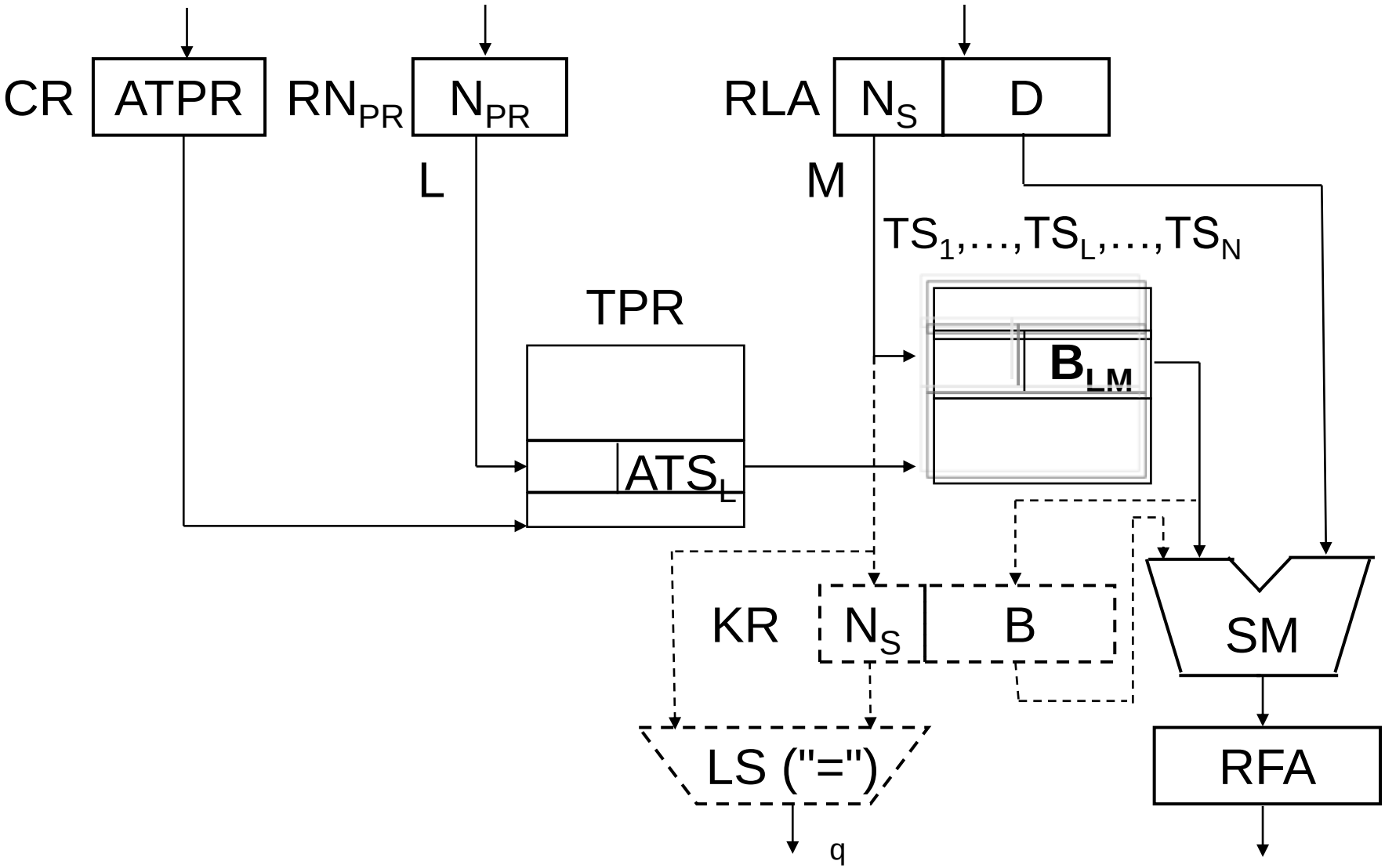


Рисунок 5 – Преобразование адреса
с использованием КЭШ-регистров

# Использование КЭШ-регистров

- Таблицы программ и сегментов обычно хранятся в ОП, поэтому табличное преобразование логического адреса в физический требует двух дополнительных обращений к памяти при выборке команды или операнда.
- Поскольку большая часть обращении к ОП идет в пределах одного и того же сегмента, то введение кэш-регистра (кэш-регистров) дает резкое сокращение числа обращений к ОП.

# Особенности представления логического адреса

- При большом числе сегментов номер сегмента NS занимает значительное число разрядов в коде команды (рисунок 6а), поэтому вместо него можно указывать номер специального регистра (NRNS), в котором хранится номер сегмента (рисунок 6б).
- Номер сегмента, хранящийся в регистре, является индексом в соответствующей таблице сегментов. В этом случае логический адрес определяется следующим образом: (RNS)$\leftrightarrow$D. Загрузка регистров выполняется специальными командами.

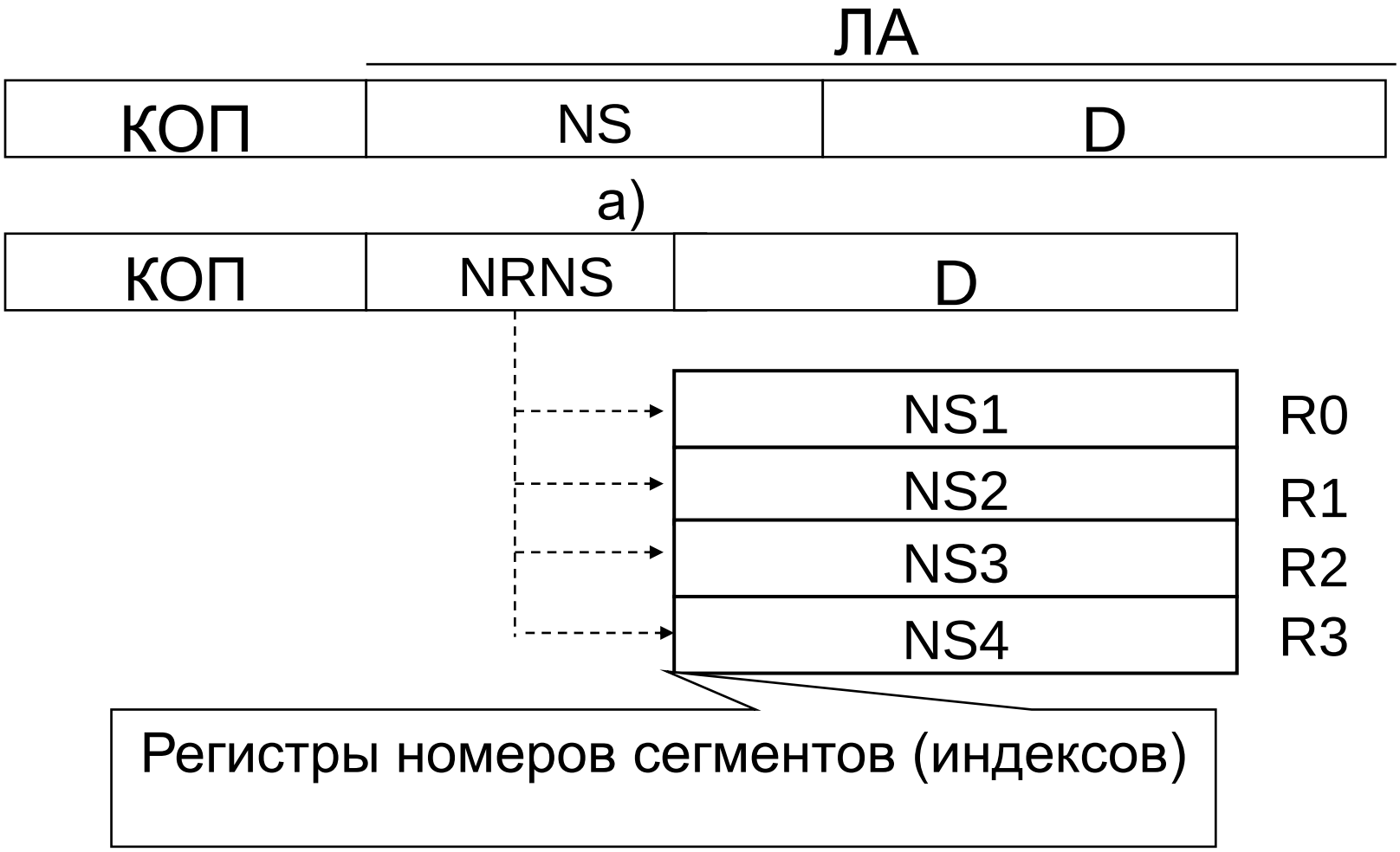


Рисунок 6 – Использование номеров регистров

# Использование неявной адресации

- Для выбора регистров номеров сегментов может быть использована неявная адресация. Все сегменты можно разделить на несколько типов: кода, стека, данных и др. В этом случае регистры часто называют селекторными (рисунок 7).
- Кэш-регистры, связанные с регистрами селекторов, называются дескрипторными, т.к. в них хранятся дескрипторы. Дескриптор соответствует строке таблицы сегментов и описывает представляемый им сегмент. Кроме базы (начального адреса сегмента), он содержит длину сегмента и другие атрибуты.

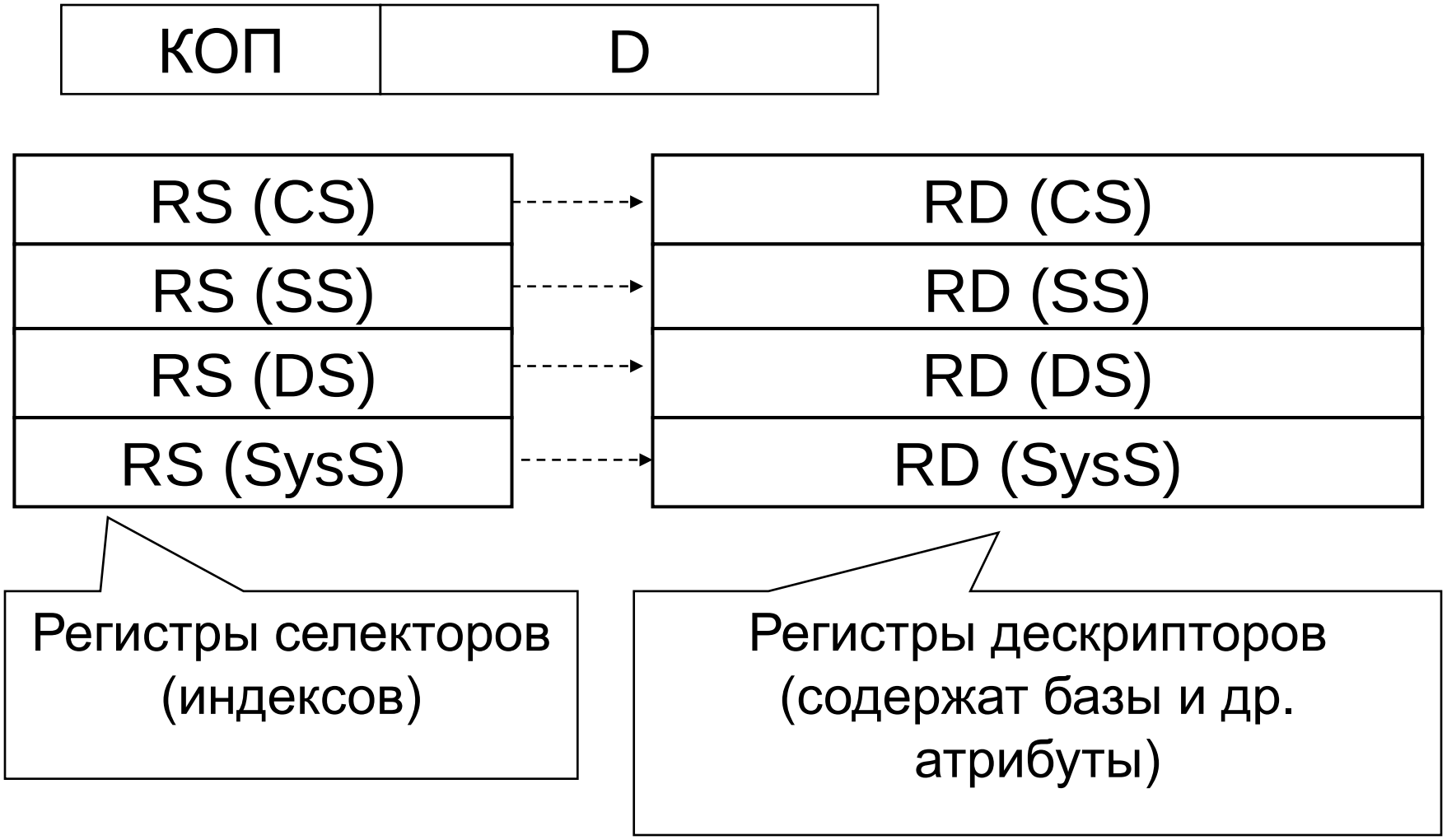


Рисунок 7 – Использование селекторных регистров

# *3. Сегментно-страничная организация памяти*

- Сегменты имеют относительно большой объем и переменную длину. В тоже время для обмена с ВЗУ используются блоки постоянного объема (кратного $2^N$), называемые страницами. Естественно разделить сегменты на страницы.
- В этом случае на уровне страниц проблема фрагментации ОП снимается. А уменьшение гранулярности представления программы в ОП открывает дополнительные возможности по увеличению коэффициента мультипрограммирования.

# Логический и физический адрес

- При использовании сегментно-страничной организации памяти логический адрес состоит из номера сегмента NS, номера математической страницы NP и адреса объекта на странице d.
- Физический адрес вычисляется путем конкатенации суммы базы B (начального адреса выбранного сегмента в ОП) и номера физической страницы NF с адресом объекта на странице D:

$AF=(B+NF)\leftrightarrow D$,

где $B=\varphi(NPR,NS)$; $NF=\psi(NPR,NS.NP)$.

- Функции $\varphi$ и $\psi$ обычно вычисляется табличным способом.

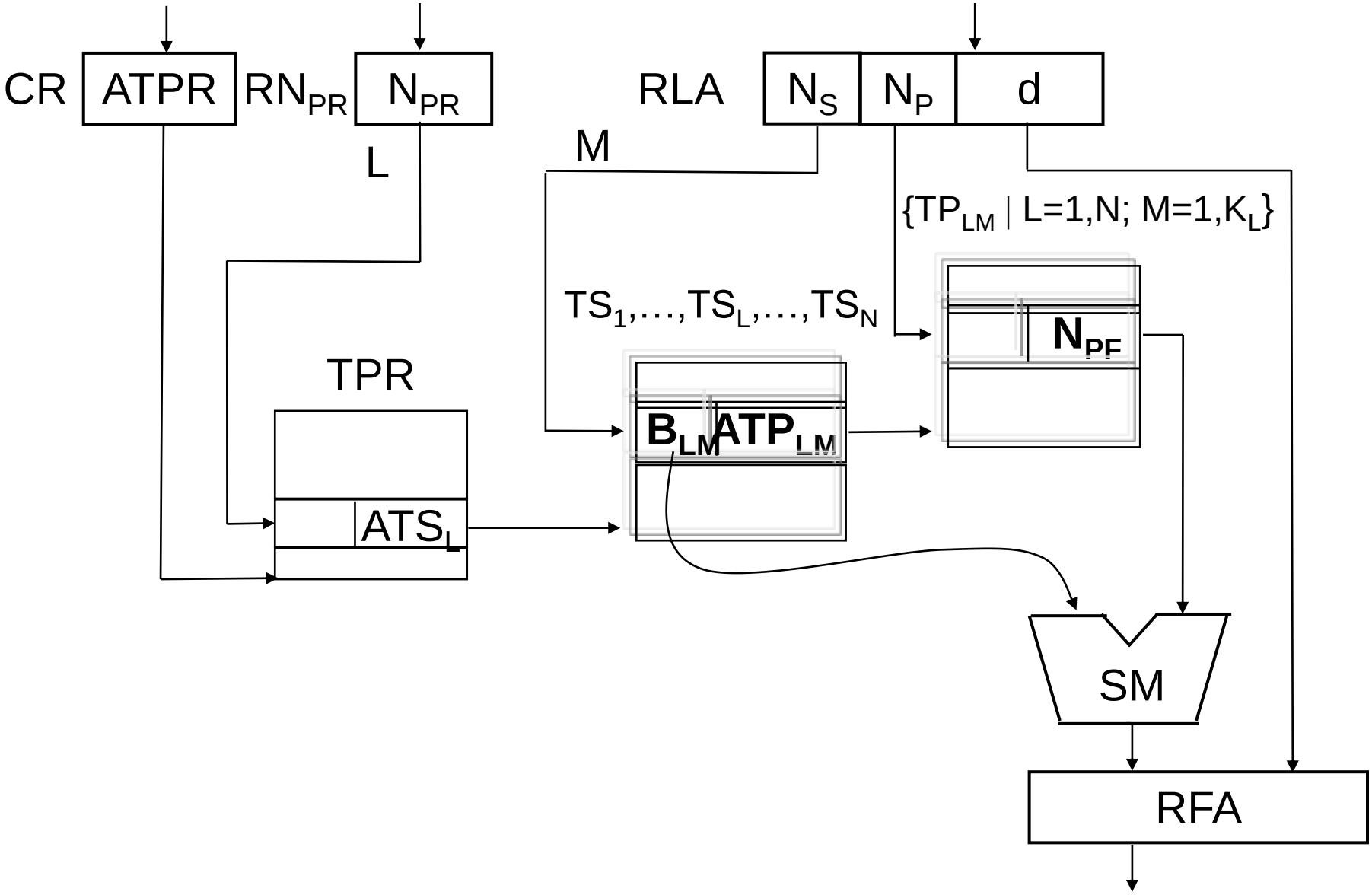


Рисунок 8 – Сегментно-страничная адресация

# Преобразование адреса (обозначения)

CR – управляющий регистр;
ATPR – начальный адрес таблицы программ;
TPR – таблица программ;
RNPR – регистр номера выполняемой программы;
NPR – номер программы (в примере NPR=L);
RLA – регистр логического адреса (в примере NS=M).
ATSL – адрес таблицы сегментов L-й программы;
TS1,…,TSL,…,TSN – таблицы сегментов программ;
BLM – базовый адрес;
ATPLM адрес таблицы страниц M-го сегмента, в L-й программе;
{TPLM | L=1,N; M=1,KL} – множество таблиц страниц.

# Использование кэш-регистров

- Для ускорения преобразования адреса здесь также используются кэш-регистры (быстрая кэш-память), хранящие ранее считанные базу и номер физической страницы.
- Это может быть отдельный блок, часто называемый буфером ассоциативной трансляции (TLB).

# Виды организации памяти

- Сегментная.
- Страничная (при большом числе страниц применяется двухуровневая страничная организация памяти, использующая каталоги страниц и таблицы страниц).
- Сегментно-страничная, когда сегменты делятся на страницы.
- Сегментная и страничная, когда эти два вида организации памяти применяются одновременно и независимо друг от друга.

# *4. Концепция виртуальной памяти*

- Каждому пользователю предоставляется виртуальная память с непрерывным адресным пространством большого (потенциально неограниченного) размера.
- Виртуальная память отображается операционной системой на физическую ОП и внешнюю память. Потенциальная неограниченность виртуальной памяти обеспечивается возможностью загрузки недостающих частей виртуальной памяти в ОП из ВЗУ и постоянного увеличения числа накопителей ВЗУ.
- В ОП программы загружаются из ВЗУ не полностью, а частями по мере возникновения необходимости. Таким образом, снимается еще одно ограничение, заключающееся в том, что выполняемая программа должна целиком находится в ОП. В результате повышается коэффициент мультипрограммирования.

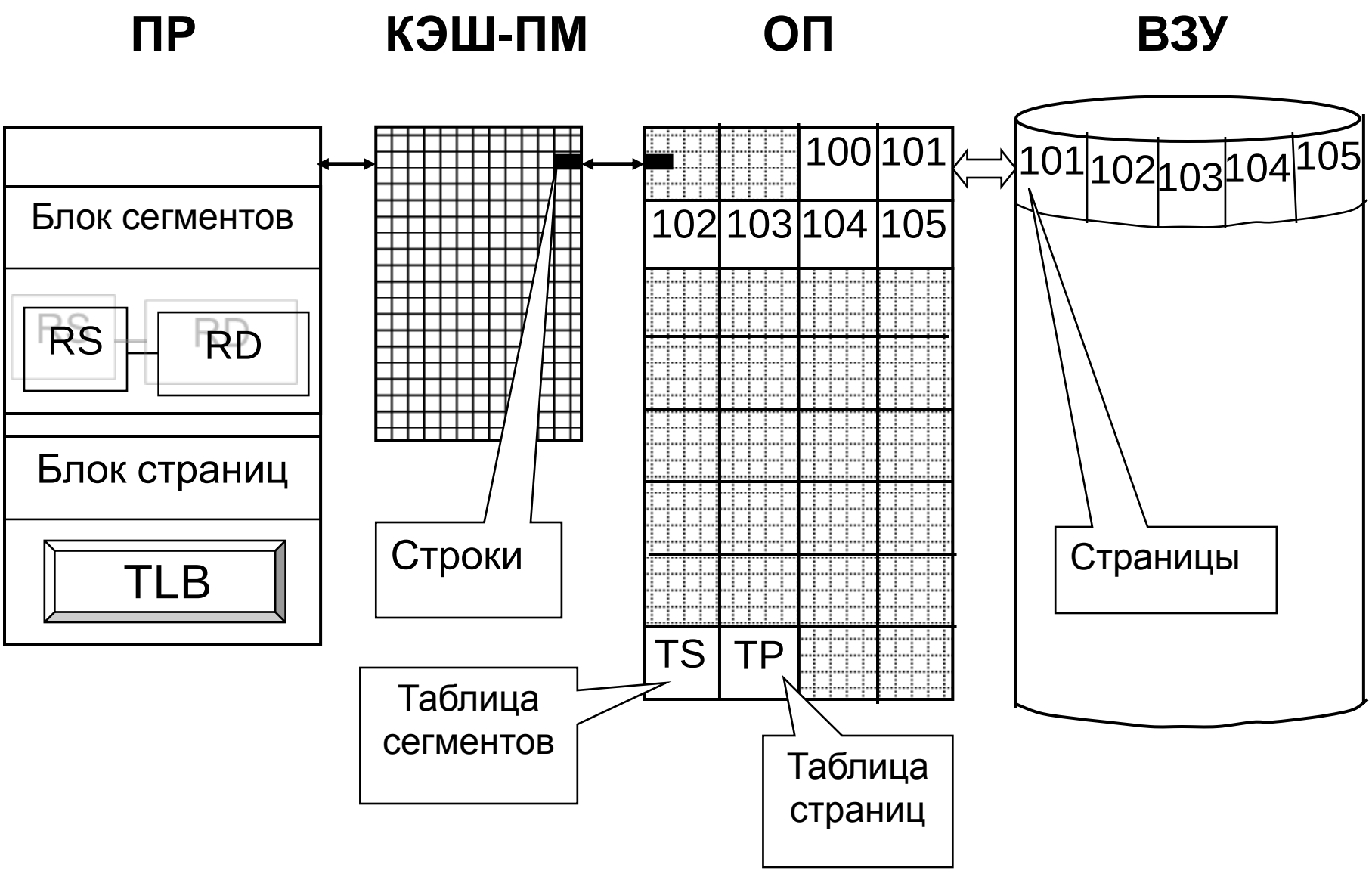


Рисунок 9 – Отображение виртуальной памяти на иерархическую память ВС

# ОП как кэш-память для внешней памяти

- Оперативную память ВС можно рассматривать как кэш-память для внешней памяти. При этом роль строи кэш-памяти здесь играют страничные кадры, которыми производится обмен данными между ОП и ВЗУ.
- Здесь также возникает проблема замещения страниц в полностью заполненной ОП. Для решения данной проблемы применяются методы аналогичные тем, что используются при замещении строк кэш-памяти.

# «Пробуксовывание» ВС

- При работе ВС с виртуальной памятью возможно возникновение, так называемого «пробуксовывания» ВС, которое проявляется в том, что значительная часть времени тратится не на вычисления, а на пересылку страничных кадров из ВЗУ и обратно.
- Причинами «пробуксовывания» могут быть особенности программ, требующих частой смены страниц, а также несовершенство алгоритмов замещения страниц.