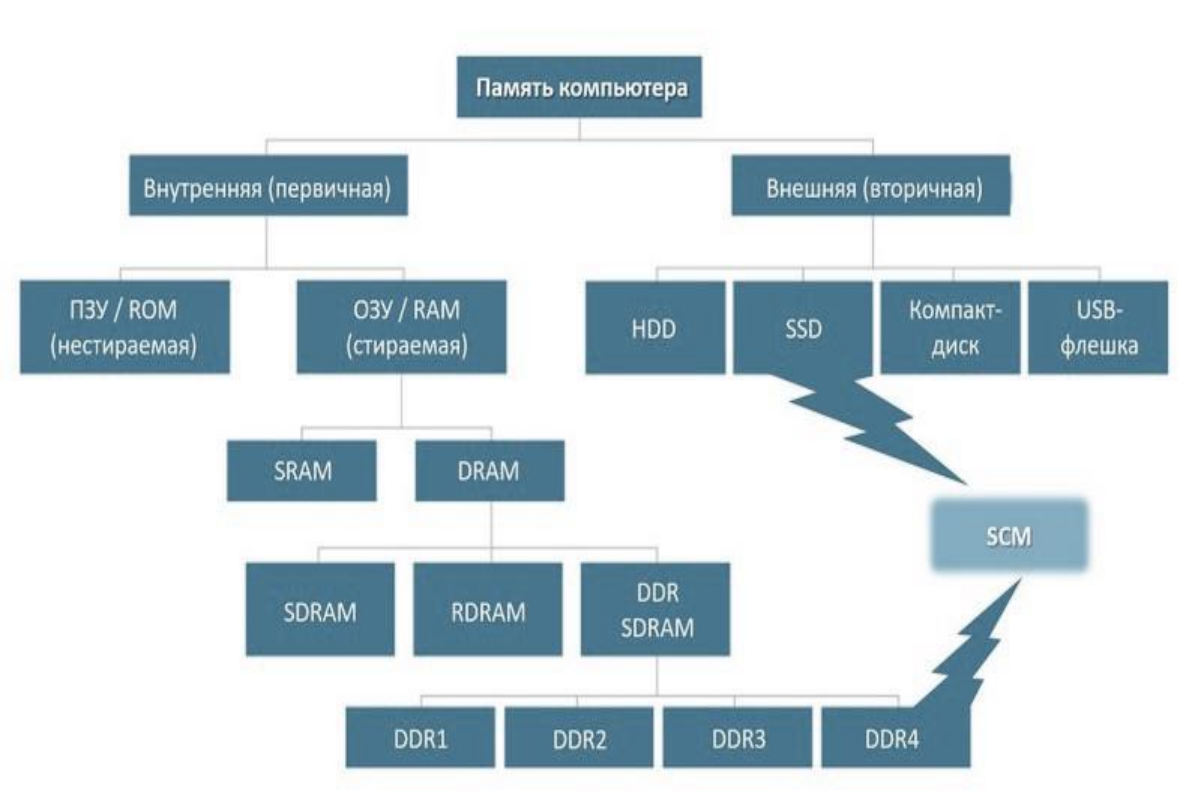
# *«Алгоритм управления памятью близнецы + SLAB»*

Подготовила:

*Омельян Екатерина*

*Б9121-09.03.03Пикд(3)*

# Что такое память и какой она бывает?




***Выделение/распределение памяти***

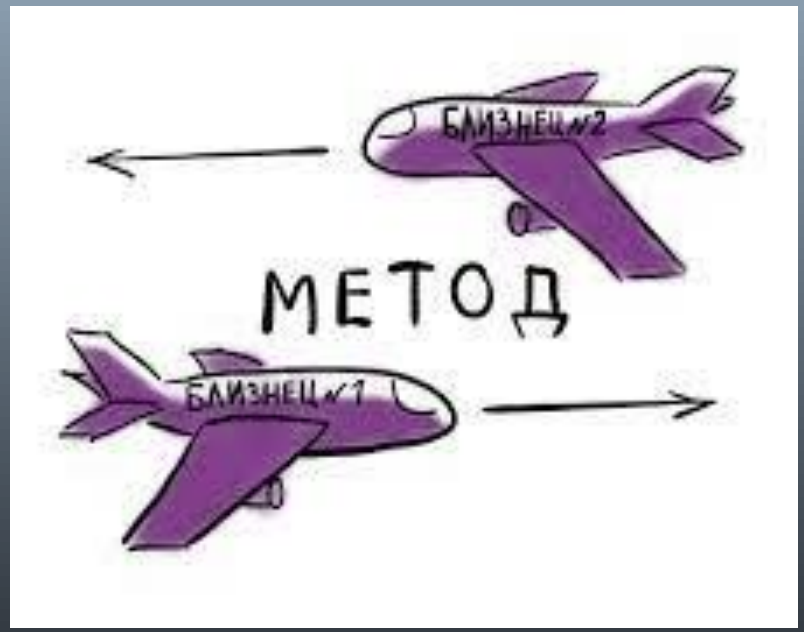
Random Access Memory - RAM устроена довольно сложно, она иерархична (многоэтажна). ОП можно разделить на несколько типов. Это разделение обусловлено историческими причинами.

Существует четыре типа памяти:

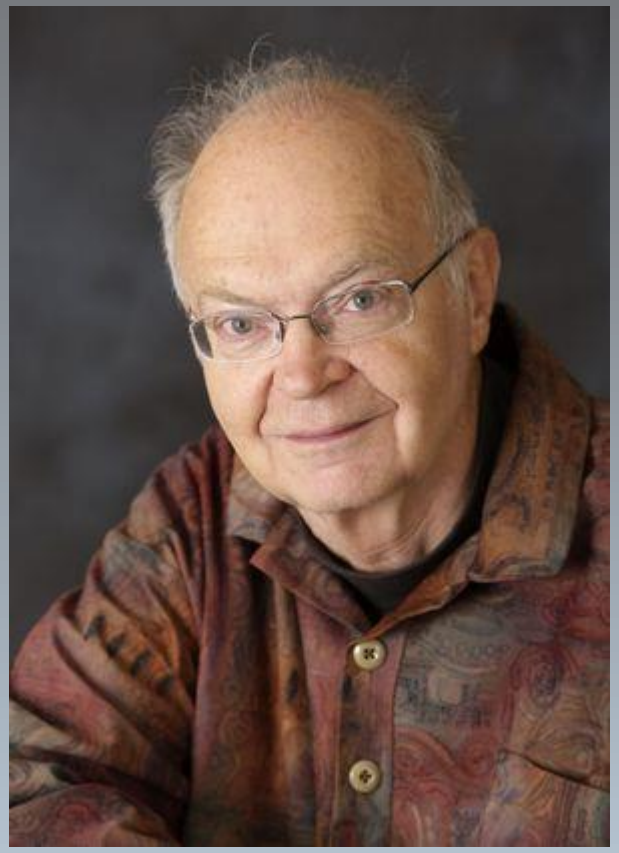
- Стандартная (традиционная область памяти)
- Верхняя (верхний блок памяти(области) )
- Дополнительная (расширенная спецификация памяти);
- Расширенная (расширенная спецификация памяти)

# *Метод близнецов*
(*Buddy system*)

Это - схема выделения памяти, сочетающей в себе возможность слияния буферов и распределитель по степени числа 21 [13]. В основе метода лежит создание буферов малого размера путем деления пополам больших буферов и слияния смежных буферов по мере возможности. При разделении буфера на два каждая половина называется близнецом (buddy) второй.



# Авторы и история алгоритма



*По словам Дональда Кнута , система близнецов была изобретена в 1963 году Гарри Марковицем и впервые описана Кеннетом К. Ноултоном (опубликовано в 1965 году).*

*Выделение памяти (Buddy) относительно легко реализовать. Он поддерживает ограниченное, но эффективное разделение и объединение блоков памяти.*

Дональд Эрвин Кнут — американский учёный в области информатики.

Гарри Макс Марко́виц — американский экономист, профессор финансов в Школе менеджмента Ради Калифорнийского университета в Сан-Диего, лауреат премии Джон фон Неймана и Нобелевской премии в области экономических наук.

# Пример реализации

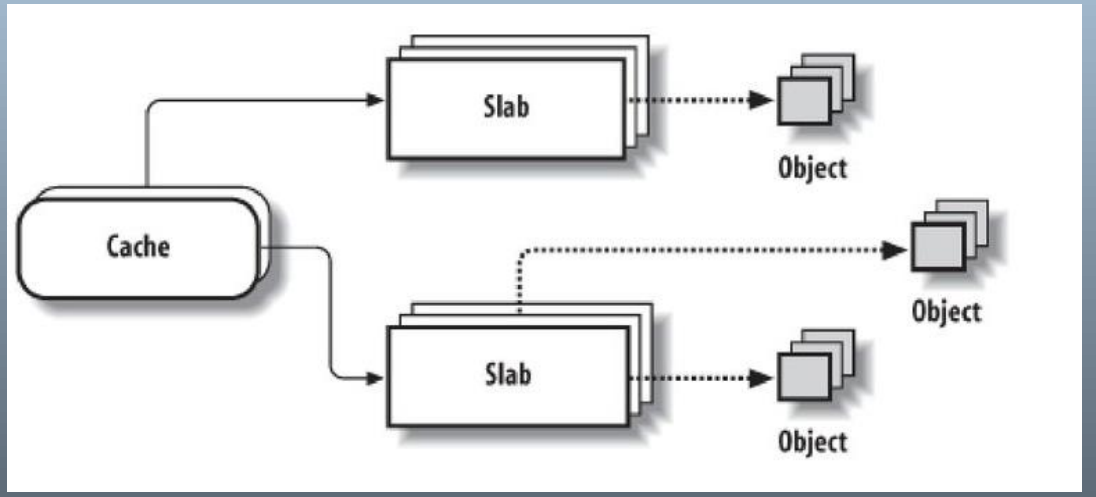
| Step | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K | 64 K |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $2^4$ | | | | | | | | | | | | | | | |
| 2.1 | $2^3$ | | | | | | | | $2^3$ | | | | | | | |
| 2.2 | $2^2$ | | | | $2^2$ | | | | $2^3$ | | | | | | | |
| 2.3 | $2^1$ | | $2^1$ | | $2^2$ | | | | $2^3$ | | | | | | | |
| 2.4 | $2^0$ | $2^0$ | $2^1$ | | $2^2$ | | | | $2^3$ | | | | | | | |
| 2.5 | A: $2^0$ | $2^0$ | $2^1$ | | $2^2$ | | | | $2^3$ | | | | | | | |
| 3 | A: $2^0$ | $2^0$ | B: $2^1$ | | $2^2$ | | | | $2^3$ | | | | | | | |
| 4 | A: $2^0$ | C: $2^0$ | B: $2^1$ | | $2^2$ | | | | $2^3$ | | | | | | | |
| 5.1 | A: $2^0$ | C: $2^0$ | B: $2^1$ | | $2^1$ | | $2^1$ | | $2^3$ | | | | | | | |
| 5.2 | A: $2^0$ | C: $2^0$ | B: $2^1$ | | D: $2^1$ | | $2^1$ | | $2^3$ | | | | | | | |
| 6 | A: $2^0$ | C: $2^0$ | $2^1$ | | D: $2^1$ | | $2^1$ | | $2^3$ | | | | | | | |
| 7.1 | A: $2^0$ | C: $2^0$ | $2^1$ | | $2^1$ | | $2^1$ | | $2^3$ | | | | | | | |
| 7.2 | A: $2^0$ | C: $2^0$ | $2^1$ | | $2^2$ | | | | $2^3$ | | | | | | | |
| 8 | $2^0$ | C: $2^0$ | $2^1$ | | $2^2$ | | | | $2^3$ | | | | | | | |
| 9.1 | $2^0$ | $2^0$ | $2^1$ | | $2^2$ | | | | $2^3$ | | | | | | | |
| 9.2 | $2^1$ | | $2^1$ | | $2^2$ | | | | $2^3$ | | | | | | | |
| 9.3 | $2^2$ | | | | $2^2$ | | | | $2^3$ | | | | | | | |
| 9.4 | $2^3$ | | | | | | | | $2^3$ | | | | | | | |
| 9.5 | $2^4$ | | | | | | | | | | | | | | | |

Идея этого алгоритма состоит в том, что организуются списки свободных блоков отдельно для каждого размера $2^k$, $0 <= k <= m$.

Когда один блок расщепляется на два (каждый из которых равен половине первоначального), эти два блока называются ***близнецами***. Позднее, когда оба близнеца освобождаются, они опять объединяются в один блок.

# *Распределение Slab*

Это - механизм управления памятью, предназначенный для более эффективного распределения памяти и устранения значительной фрагментации. Основой этого алгоритма является сохранение выделенной памяти, содержащей объект определённого типа, и повторное использование этой памяти при следующем выделении для объекта того же типа.



# Пример реализации

```
Sample Run 3
-------------
Internal Fragmentation (KB)= 2 -----% 0.03125

# of Allocation Requests Denied = 0

External Fragmentation = 0

Sample Run 4
------------
Internal Fragmentation (KB)= 16 -----% 0.25

# of Allocation Requests Denied = 6422

External Fragmentation = 0
```

- Для реализации системы напарников использовалась древовидная структура данных.

-Прежде всего создайте один узел дерева размером 64 КБ, а затем войдите в цикл, который будет выполняться 10 100 раз.

- Запустите случайное выделение и освобождение в этом цикле, вызвав соответствующие функции на основе остатка функции rand.

-Когда когда-либо вызывается функция распределения, генерируется процесс случайного размера, и вычисляется ближайшее значение в степени 2, что помогает при необходимости разбить дерево.

# *Анализ алгоритма* (*Buddy system*)

- Объем пространства, потерянного в результате внутренней фрагментации, зависит от размера процесса.

-Если система-партнер загружается процессом большего размера, то пространство, потраченное впустую в результате внутренней фрагментации, больше, потому что этим процессам выделяются более крупные блоки.

-Кроме того, количество отклоненных запросов на выделение больше, но обычно оно зависит от того, как выделения и отмены распределения накладываются друг на друга.

-Внешняя фрагментация в системе друзей всегда будет нулевой.

-Когда процесс удаляется, возникают дополнительные затраты на проверку друзей до корневого узла для слияния, что делает этот метод медленнее.

На этом всё, спасибо
за внимание