Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Операционные системы

Студент: Немкович А.В.

ФИТ 3 курс 1 группа

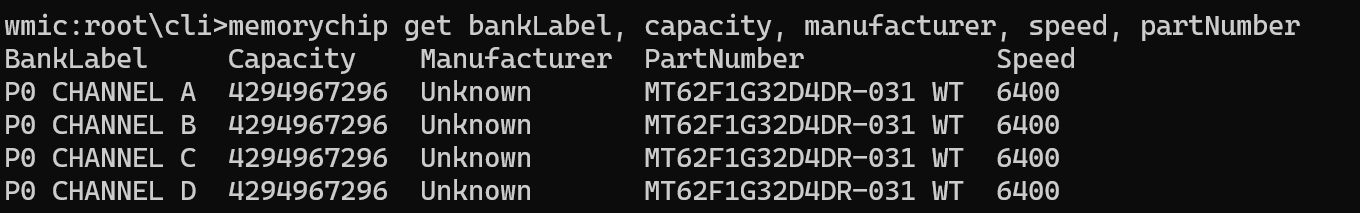
Преподаватель: Бернацкий П.В

Минск 2024

**Лабораторная работа №8**

**Задание 01. Windows**

* **Получите с помощью утилиты wmic информации об физической оперативной памяти компьютера, поясните эту информацию.**



Команда **WMIC** используется для получения сведений об оборудовании и системе, управления процессами и их компонентами, а также изменения настроек с использованием возможностей инструментария управления Windows.

Memorychip - получить информацию о модулях памяти.

BankLabel: Слот памяти на материнской плате, в котором установлен модуль.

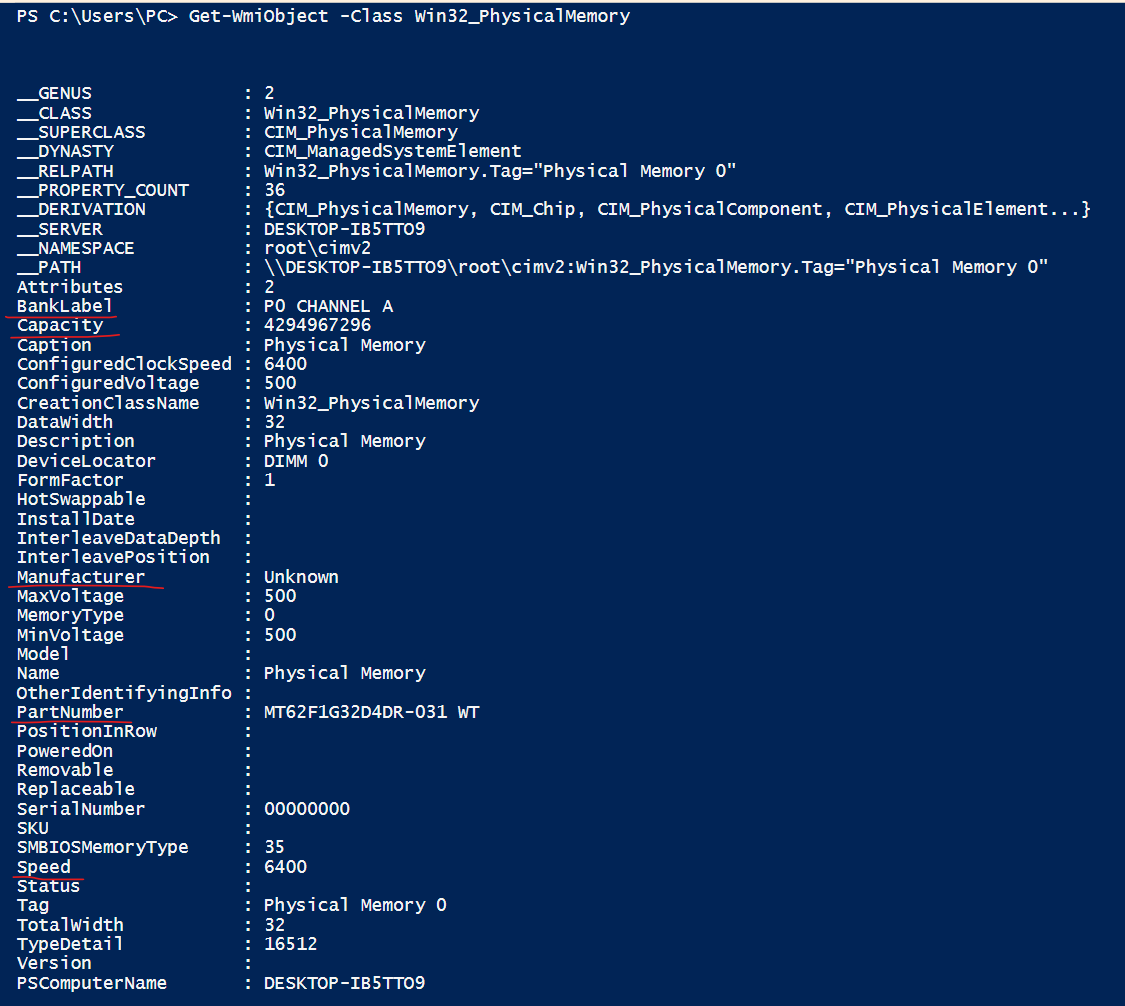
Capacity: Объём каждого модуля памяти в байтах. Значение 4294967296 соответствует 4 ГБ.

Manufacturer: Производитель модуля памяти. Производитель памяти указан как Unknown. *Это может быть связано с ограничением данных, предоставляемых BIOS/UEFI.*

PartNumber — Номер модели модуля памяти. Используется для точной идентификации конкретного типа памяти.

Speed: Тактовая частота работы памяти в мегагерцах. 4 модуля работают на частоте 6400МГц.

* **Получите с помощью утилиты powershell информации об физической оперативной памяти компьютера, поясните эту информацию.**



ConfiguredClockSpeed: Это скорость работы памяти, выраженная в мегатрансферах в секунду (MT/s). Значение 6400 MT/s указывает на DDR5 память с такой частотой.

ConfiguredVoltage: Напряжение памяти в милливольтах (мВ). Значение 500 мВ (или 0,5 В) указывает на стандартное напряжение для DDR5 памяти.

DataWidth: Ширина данных, которая указывает на количество битов, которые могут передаваться за один такт.

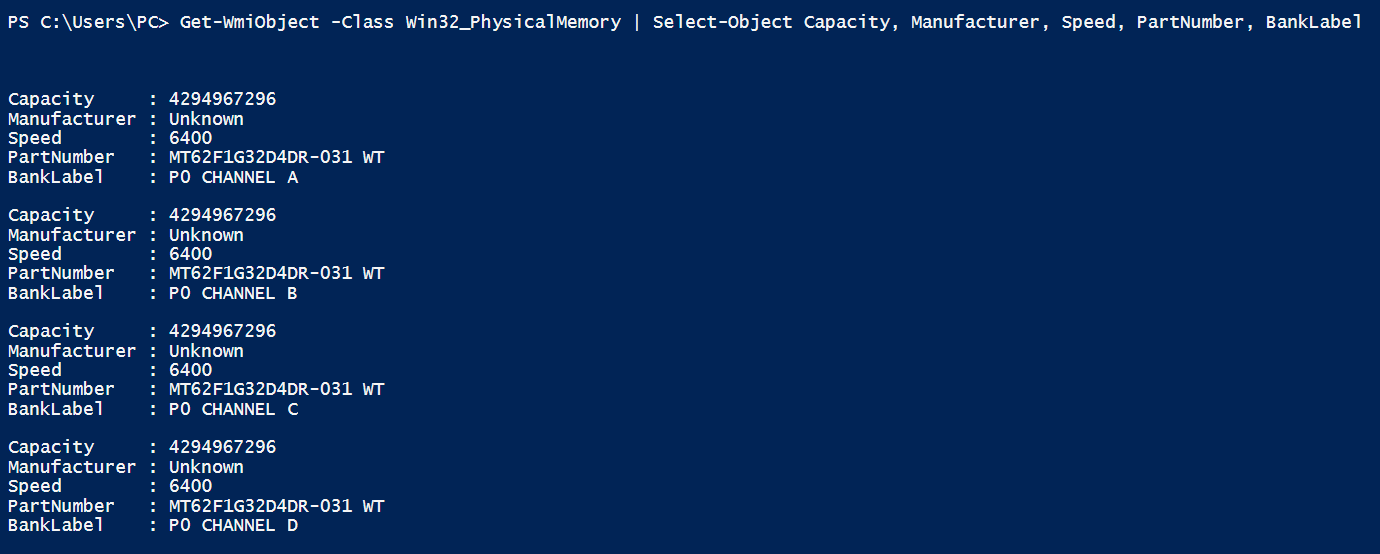
Description: Physical Memory: Это описание типа устройства. В данном случае, это физическая память (RAM).

DeviceLocator: Указывает, в каком слоте материнской платы установлен модуль памяти. Здесь DIMM 0 указывает на первый слот для модулей памяти

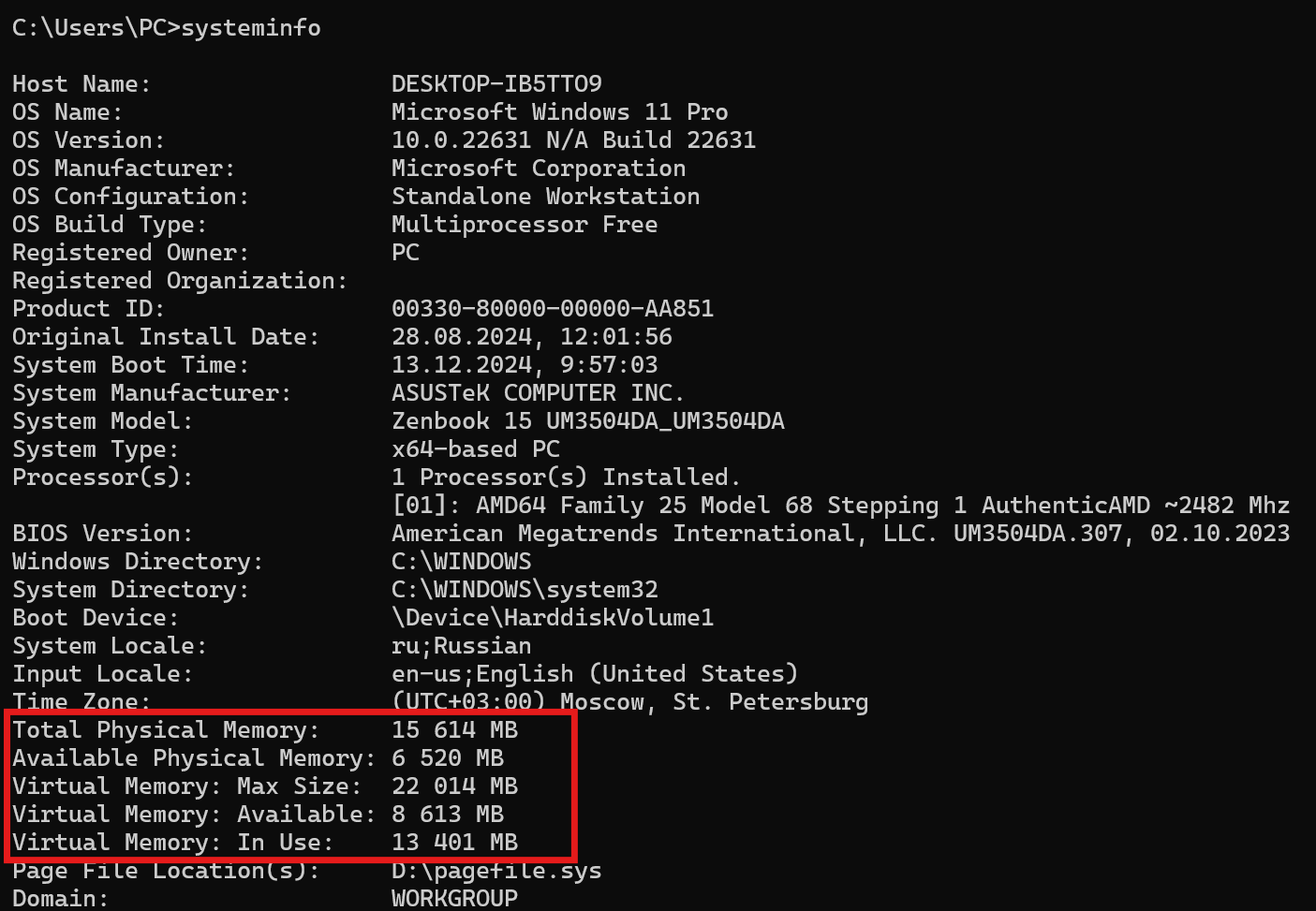
MaxVoltage: Максимальное напряжение модуля памяти в милливольтах (мВ).

MemoryType: Код 0 может означать, что тип памяти не был правильно идентифицирован или используется более старая версия данных. В некоторых случаях это значение может быть неинтерпретируемым.

MinVoltage:500: Минимальное напряжение модуля памяти в милливольтах (мВ).



* **Получите с помощью утилиты systeminfo информации об оперативной памяти компьютера, поясните эту информацию.**

  
Total Physical Memory (Общая физическая память):Это общий объем установленной оперативной памяти (RAM) в системе. Не вся установленная память доступна для использования, так как часть может быть зарезервирована для аппаратных нужд, таких как встроенная графика.

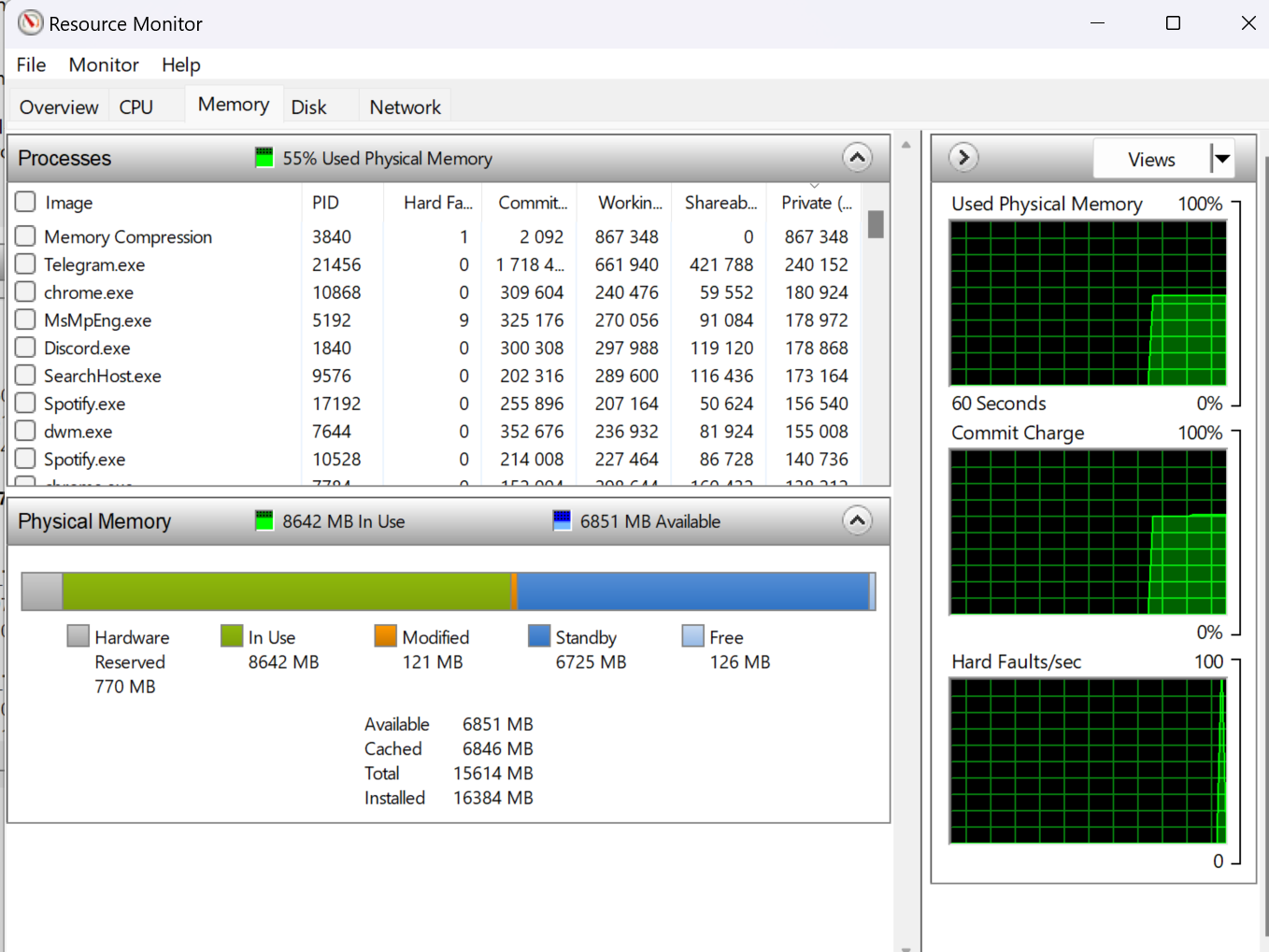
Available Physical Memory (Доступная физическая память): Это объем физической памяти, который в данный момент не используется и доступен для выполнения задач. Если эта память заканчивается, система начинает использовать файл подкачки, что может замедлить производительность.

Virtual Memory: Max Size (Максимальный размер виртуальной памяти): Виртуальная память — это комбинация физической памяти (RAM) и файла подкачки на диске. Максимальный размер виртуальной памяти указывает на суммарный объем RAM и выделенного пространства на диске для файла подкачки.

Virtual Memory: Available (Доступная виртуальная память): Это объем виртуальной памяти (RAM + файл подкачки), который в данный момент свободен и доступен для выполнения задач. Если доступной виртуальной памяти мало, это может привести к снижению производительности системы.

Virtual Memory: In Use (Используемая виртуальная память): Это объем виртуальной памяти, который в данный момент используется операционной системой и приложениями. Если этот показатель близок к максимальному размеру виртуальной памяти, система работает под высокой нагрузкой и может начать испытывать проблемы с производительностью.

* **Получите с помощью утилиты performance monitor информации об оперативной памяти компьютера, поясните эту информацию.**

****

Hardware Reserved: память, зарезервированная для использования BIOS и драйверами других периферийных устройств.

In Use: память, используемая процессами, драйверами или ОС.

Modified: память, содержимое которой необходимо записать на диск, прежде чем использовать это содержимое в других целях.

Standby: память, в которой содержатся кэшированные данные, и которая недоступна для использования.

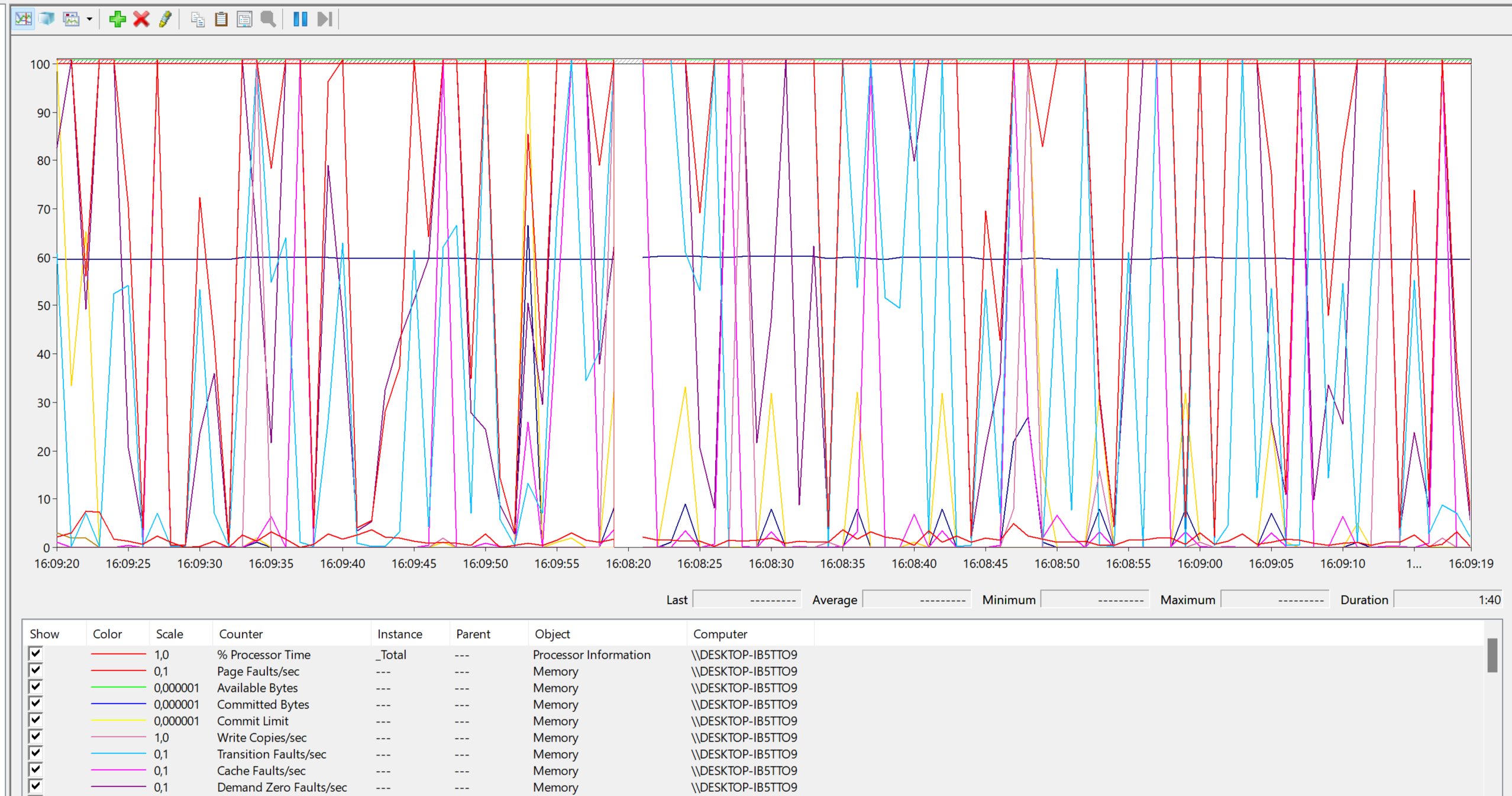
Free: память, в которой не содержатся полезные данные, и которая в первую очередь занимается процессами, драйверами и ОС в случае необходимости. В нормальных условиях её и должно быть свободно чуть-чуть, т.к. остальная доступная память должна уходить на кеширование.

Available: объём памяти (включая зарезервирванную и свободную), доступную для немедленного использования процессами, драйверами или ОС.

Cached: объём памяти (включая зарезервирванную и свободную), в которой содержатся кэшированные данные и код, предназначенные для мгновенного доступа и использования процессами, драйверами или ОС.

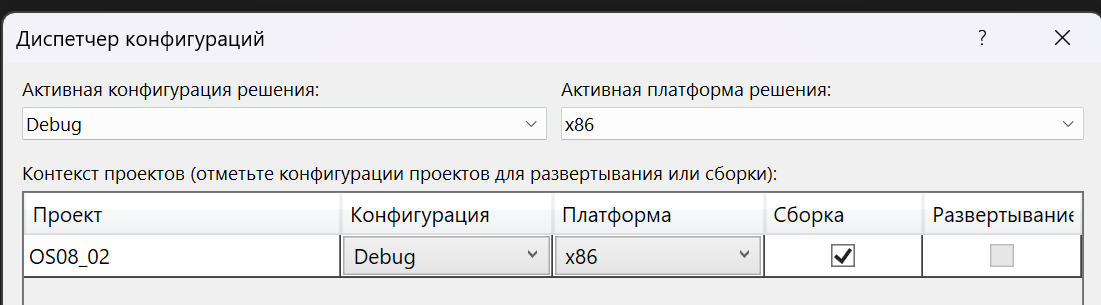
Total: объём физической памяти, доступной для использования ОС, процессами и драйверами.

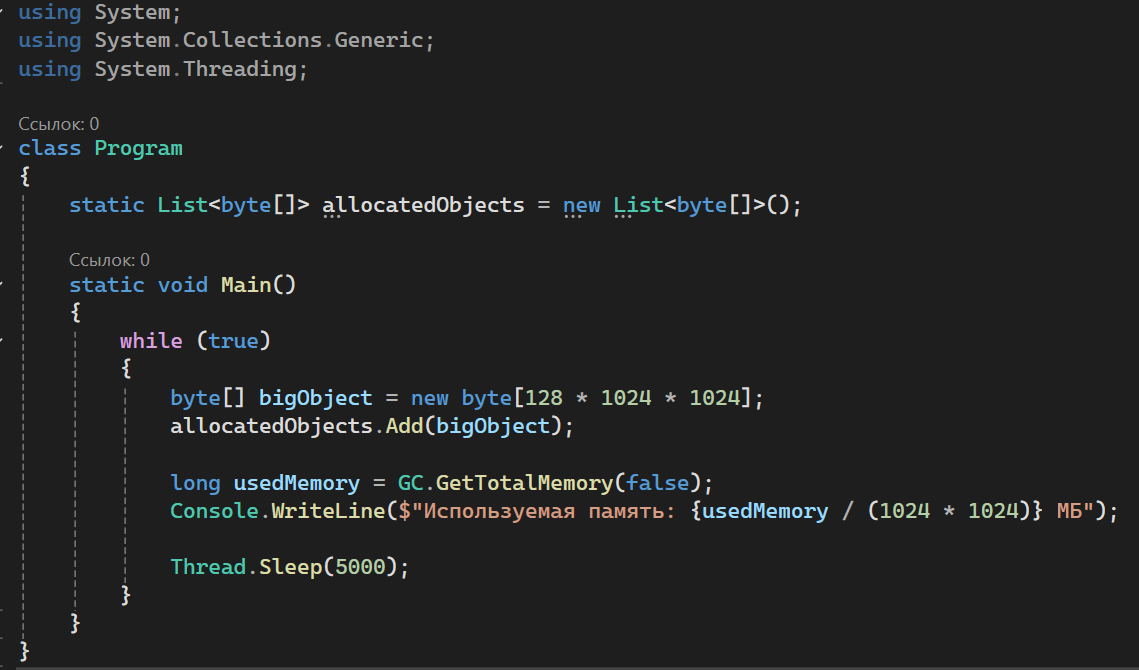
Installed: объём физической памяти, установленной на компьютере.

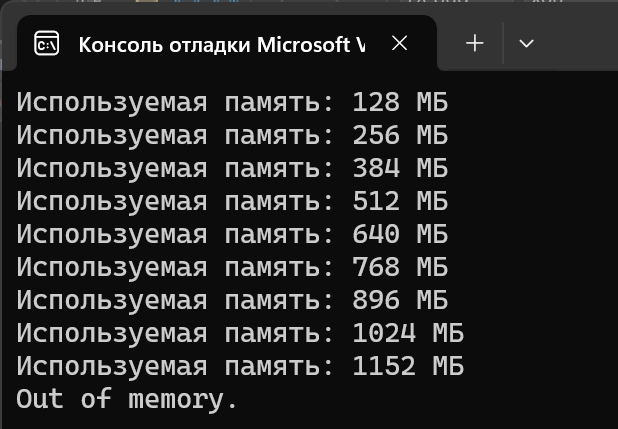
****

**Задание 02.Windows**

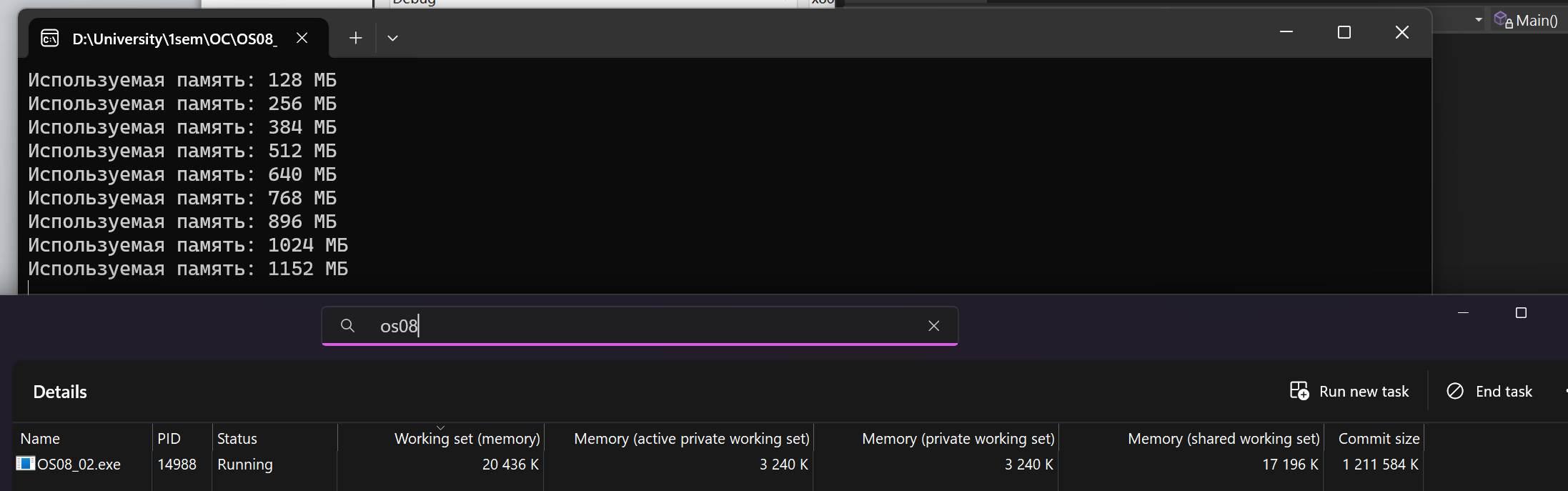
**Разработайте на языке программирования C# консольное приложение OS08\_02, которое каждые 5 секунд создает новый объект размером 128 МБ и выводит объем используемой памяти, установите Platform Target x86.**







**Понаблюдайте за выделенной памятью рабочим набором памяти приложения OS08\_02 в Диспетчере задач. Будет ли приложение работать бесконечно?**



Working Set (Memory): Объем физической памяти, который в данный момент используется процессом.

Memory (Active Private Working Set): Объем физической памяти, используемой процессом, который не может быть использован другими процессами (за исключением приостановленных процессов UWP).

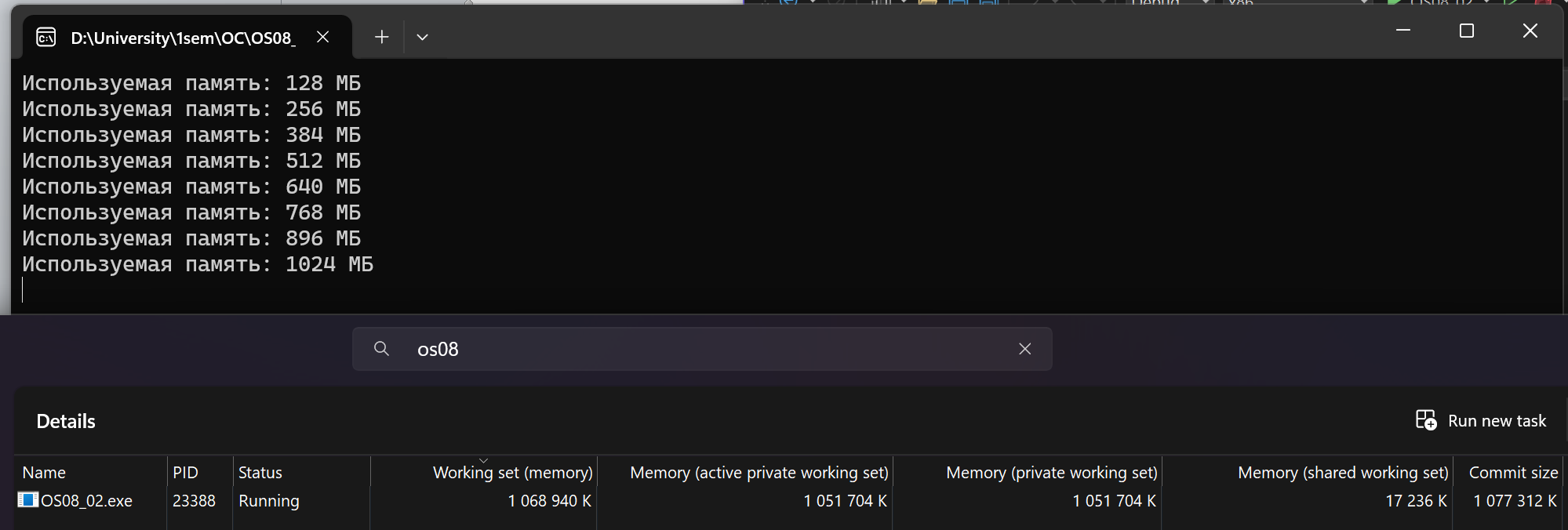
Memory (Private Working Set): Объем физической памяти, используемой процессом, который не может быть использован другими процессами.

Memory (Shared Working Set): Объем физической памяти, используемой процессом, который может быть разделен с другими процессам

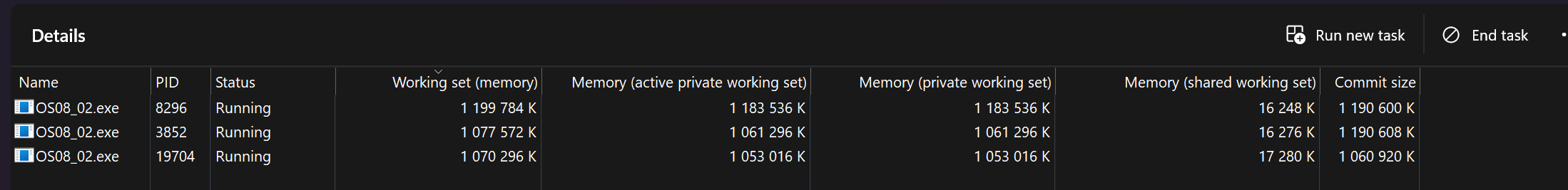
(Commit Size): Это объем памяти, зарезервированной для процесса, включая физическую память и файл подкачки.

**Добавьте в приложение OS08\_02 код, который заполняет произвольными значениями выделенную память, лучше всего в отдельном потоке. Изменилось ли соотношение выделенной памяти и рабочего набора?**



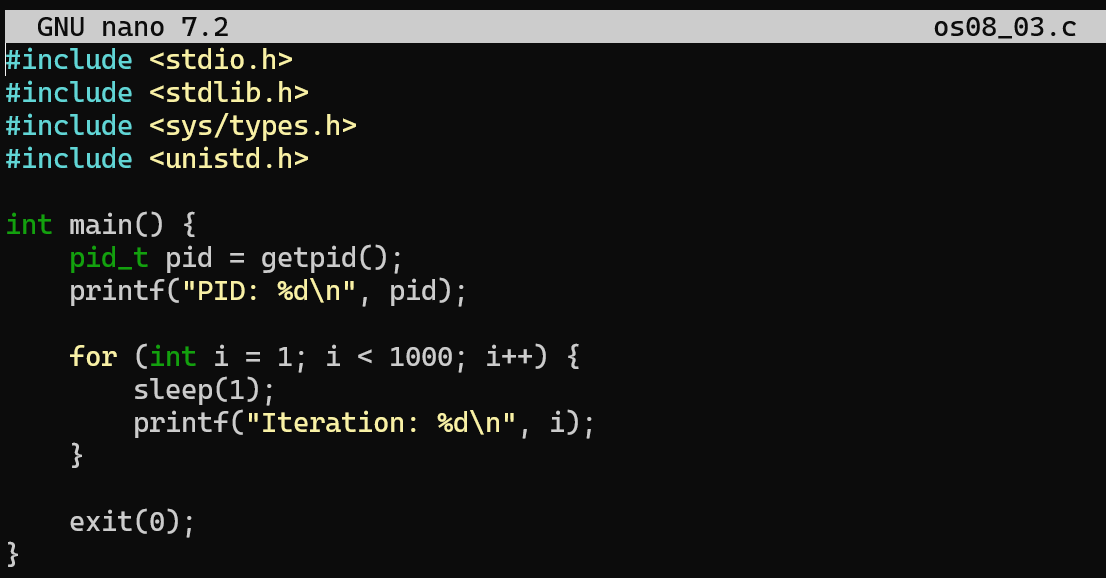


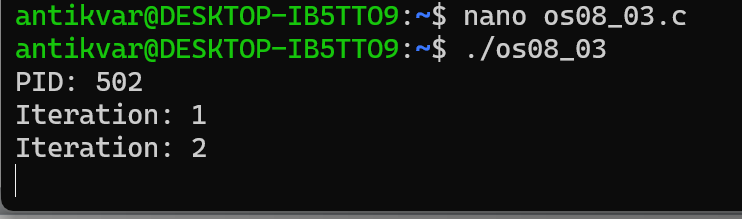
**Запустите несколько экземпляров приложения OS08\_02.Сравните частный и общий рабочие наборы.**

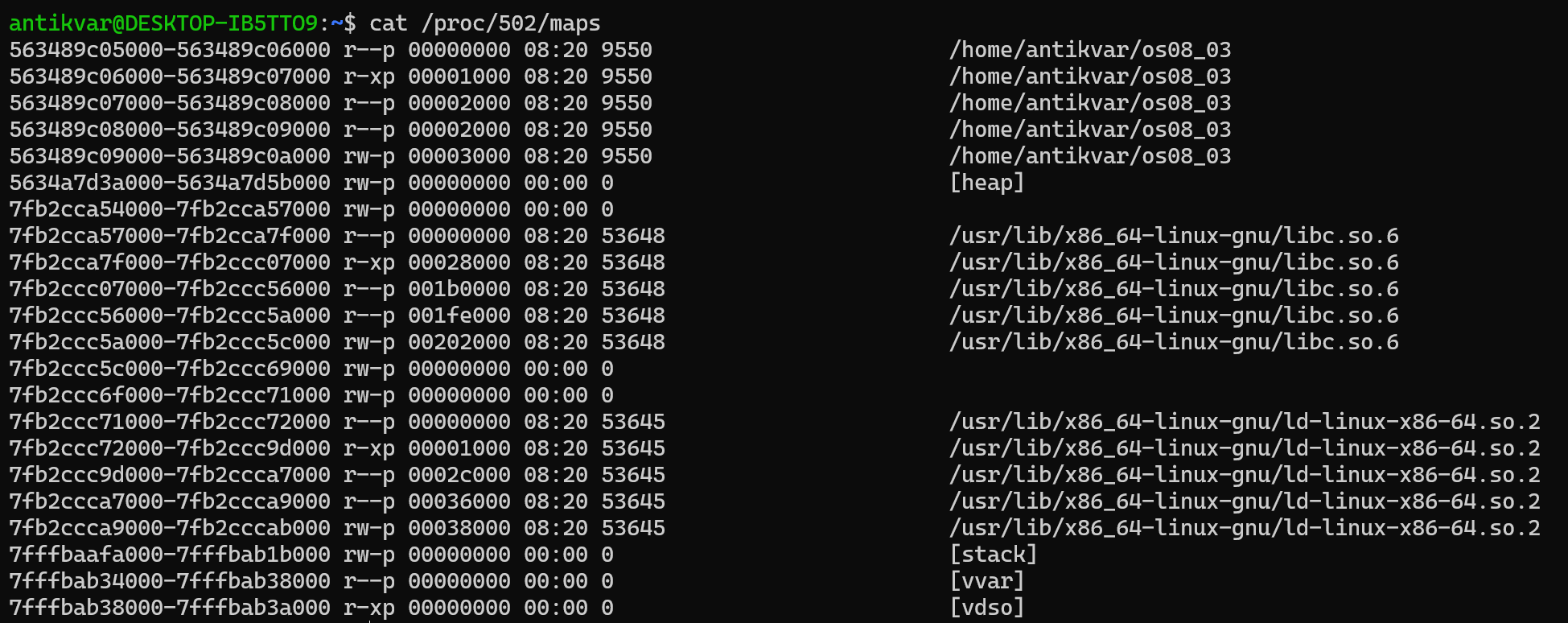


**Задание 03.Linux**

**Разработайте консольное приложение OS08\_03, выполняющее длинный цикл.**





**Продемонстрируйте с помощью файловой системы /proc структуру адресного пространства.** 

Содержимое файла /proc/<PID>/maps, который отображает информацию о виртуальных страницах памяти, используемых процессом.

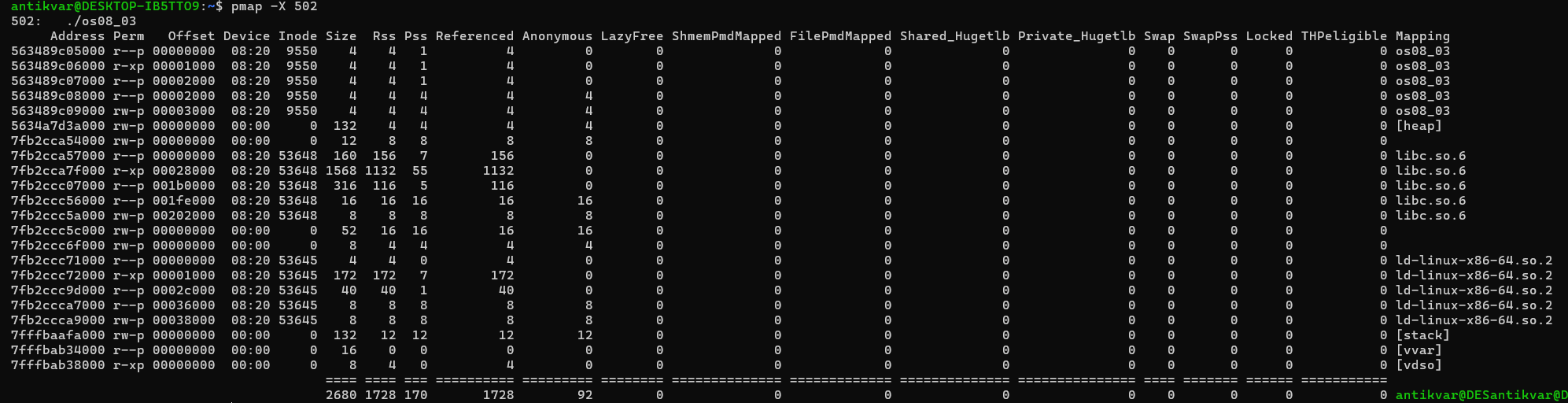
Формат строки /proc/[PID]/maps:

<адреса> <права> <смещение> <устройство> <inode> <имя файла или описание>

* Адресный диапазон в памяти (начальный и конечный адрес)
* Права доступа (read, write, execute, private – привязка к файлу)
* Смещение в файле (число байт от начала файла, с которого начинается область, отображенная в память)
* Устройство к которому привязана эта память. Если устройство отображает 00:00, это означает, что область памяти не связана с физическим устройством (например, анонимная память, куча, стек или разделяемая память). 08: Старший номер устройства. Может указывать на определенное устройство (например, жесткий диск). 20: Младший номер устройства. Может указывать на конкретный раздел диска или область.
* Inode в Linux - это структура данных, которая хранит информацию о файле или каталоге, такую как его владелец, права доступа, дата и время создания и изменения, размер и расположение на жестком диске. 0 означает, что область памяти не связана с файлом (например, это стек или куча).
* Имя файла. Указывает, с каким файлом или объектом связана эта область памяти.

**Продемонстрируйте с помощью pmap структуру адресного пространства**.

Она выводит информацию о том, как память процесса распределена, включая информацию о сегментах памяти (например, текстовые, данные, стек и т. д.), а также о том, какие области памяти являются доступными или защищенными.



Address — адрес начала сегмента памяти. Это показывает, где в виртуальном адресном пространстве процесса начинается данный сегмент.

Perm — права доступа к сегменту памяти (например, r-xp — чтение, исполнение, доступ на чтение и исполнение).

Offset — смещение в файле или устройстве для этого сегмента.

Device — идентификатор устройства, с которым ассоциирован сегмент.

Size — размер сегмента в байтах.

Rss — размер сегмента, который реально загружен в физическую память.

Pss — пропорциональная доля использования физической памяти процессом, если сегмент разделяется с другими процессами.

Referenced — количество страниц, которые были использованы.

Anonymous — количество анонимных страниц, которые не связаны с файлами.

LazyFree — количество страниц, которые еще не освобождены, но отмечены для освобождения.

SharedClean — количество чистых (не измененных) страниц, которые разделяются с другими процессами.

SharedDirty — количество измененных страниц, которые разделяются с другими процессами.

PrivateClean — количество чистых страниц, которые не разделяются.

PrivateDirty — количество измененных страниц, которые не разделяются.

FilePmdMapped — количество страниц, отображенных из файлов.

SharedHugeTlB — количество страниц, используемых для больших страниц, которые могут быть разделены между процессами.

PrivateHugeTlB — количество больших страниц, которые не могут быть разделены.

Swap — количество страниц, выгруженных в swap (виртуальная память на диске).

Pss — пропорциональный размер используемой памяти.

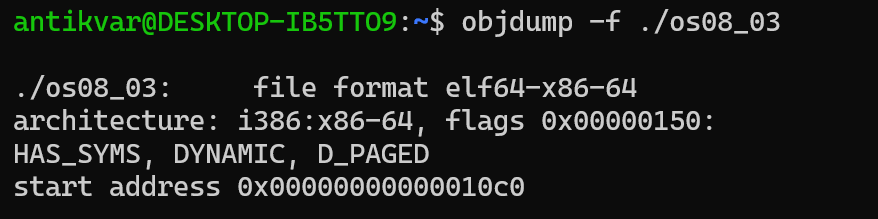
Locked — количество страниц, которые были заблокированы в памяти и не могут быть выгружены в swap.

Mapping — путь к файлу или описание сегмен

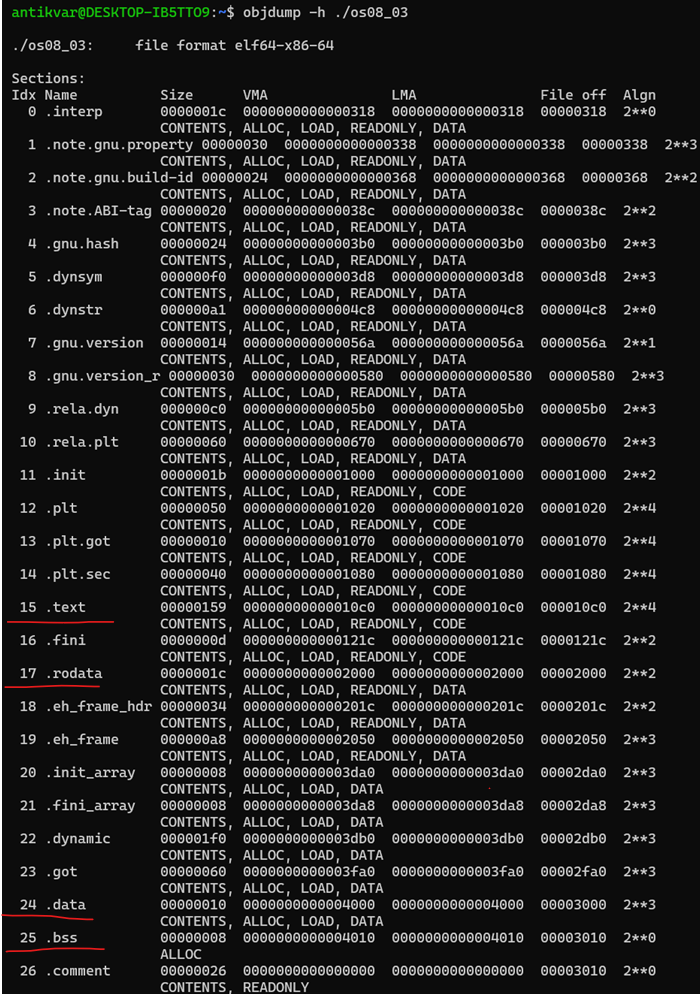
**Определите с помощью утилиты objdump адрес загрузки mainмодуля, секций с кодом, данными, неинициализированными глобальными переменными.**

-f : отобразить содержимое общего заголовка файла

-h: отобразить содержимое заголовков разделов.



ELF (Executable and Linkable Format) — это стандартный формат файлов, используемый в операционных системах на основе Unix, для хранения исполняемых файлов, объектных файлов, общих библиотек и дампов памяти.



.text — секция с кодом программы.

.data — секция с инициализированными данными.

.bss — секция с неинициализированными данными (глобальные переменные, которые не были инициализированы в исходном коде).

.rodata — секция с только для чтения данными (например, строки).

**Структура**:

Idx Name - Имя секции

Size - Размер секции в байтах. Показывает, сколько данных или кода содержится в секции.

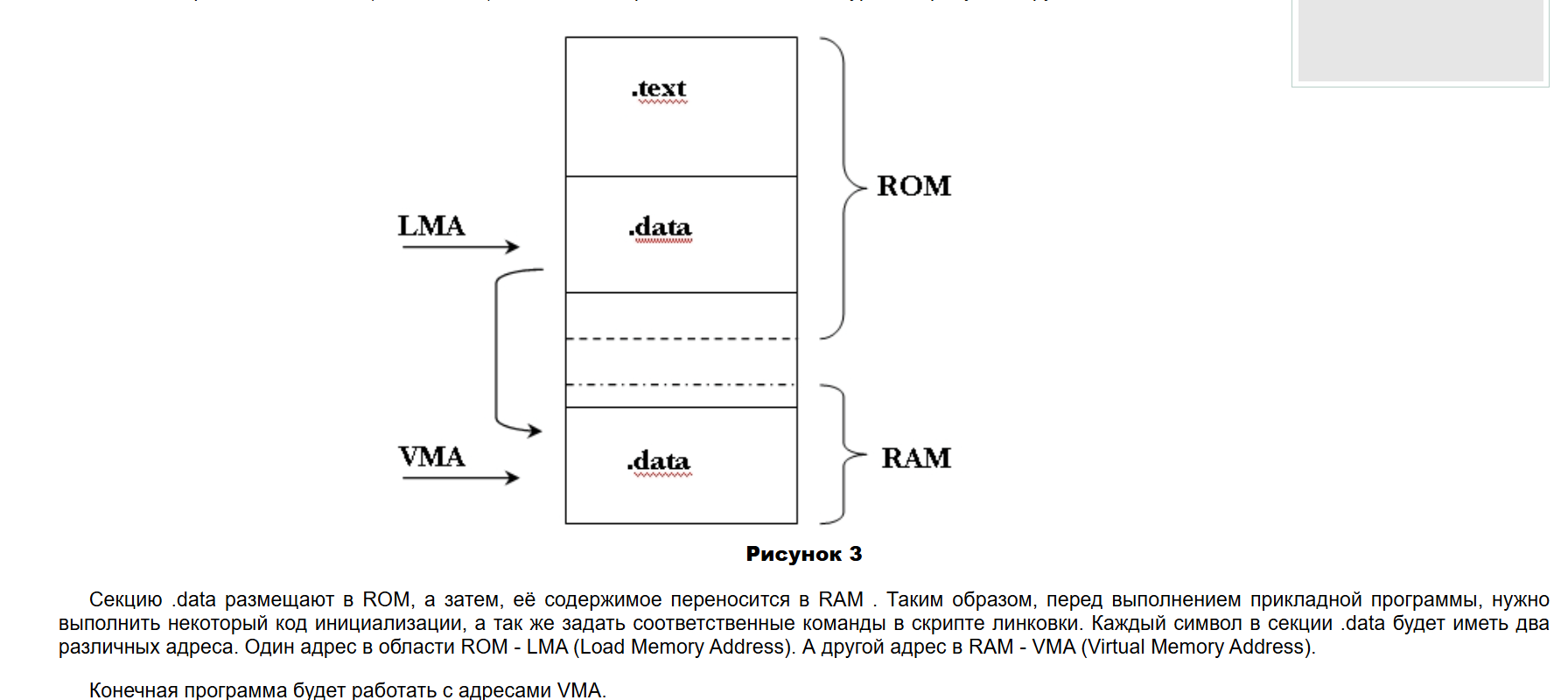
VMA (Virtual Memory Address) - Виртуальный адрес, по которому секция будет загружена в оперативную память. Этот адрес используется процессором во время выполнения программы.

Load Memory Address - Адрес загрузки секции. Адрес, по которому секция изначально загружается из ELF-файла. Обычно совпадает с VMA, но может отличаться для секций, которые перемещаются в процессе выполнения.

File off (File Offset) - Смещение секции в файле относительно начала ELF-файла. Показывает, где физически находятся данные этой секции в файле.

Algn (Alignment): Выравнивание секции в памяти. Значение 2\*\*n означает, что адрес секции выравнен по границе в 2^n байт. Например, 2\*\*4 (16 байт) означает, что секция начинается с адреса, кратного 16.

Flags: Права доступа и характеристики секции (например, READONLY, CODE, DATA).



**Задание 04.**

Ответьте на вопрос, соответствующий вашему варианту (первая буква фамилии А-И — вариант 1, К-О — вариант 2, П-Я — вариант 3).

Вариант 2. Укажите содержимое байта по виртуальному адресу 0x0167015F, если регистр CR3 содержит значение 0x13907000.

**Запишите виртуальный адрес в двоичном виде**:

|  |  |
| --- | --- |
| 0 | = 0000 |

|  |  |
| --- | --- |
| 1 | = 0001 |

|  |  |
| --- | --- |
| 6 | = 0110 |

|  |  |
| --- | --- |
| 7 | = 0111 |

|  |  |
| --- | --- |
| 5 | = 0101 |

|  |  |
| --- | --- |
| F | = 1111 |

00000 00101 10011 10000 00010 10111 11

**Индекс каталога страниц:**

PDI = старшие 10 бит виртуального адреса.

0000000101

В16ричн это 0x5

**Индекс таблицы страниц:**

PTI = следующие 10 бит виртуального адреса.

1001110000

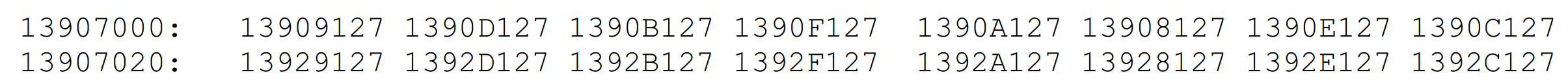
В 16ричн это 270

Адрес записи каталога таблиц (PDE)

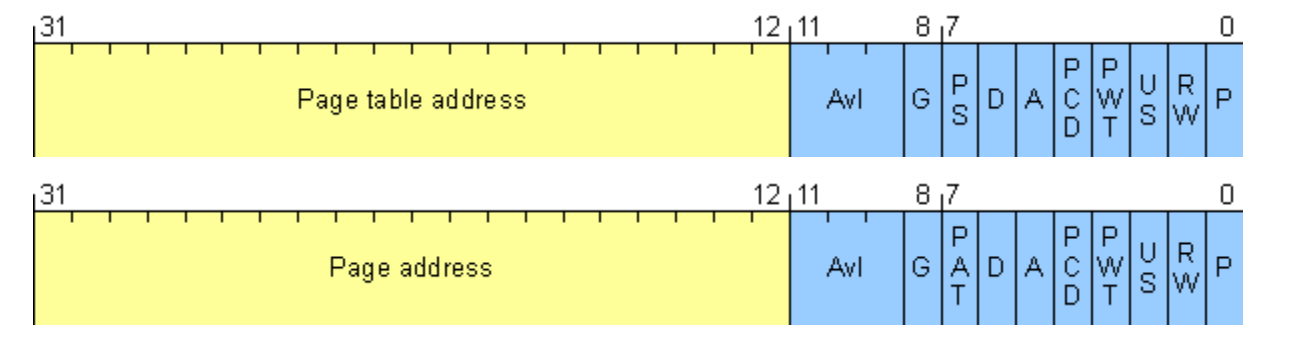
PDE=CR3+(Индекс каталога страниц×4)

0х5\*4 = 5 \* 4 = 20 = 0х14

PDE= 0x13907000+0x14=0x**13907014**



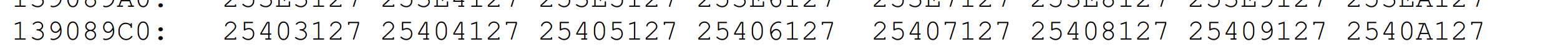
13908127 (индекс каталога страниц = 5) – PDE.



Поскольку нам надо использовать старшие 20 бит, мы будем в вычислениях использовать **13908000**.

PTE=PDE+(Индекс таблицы страниц×4)

PTE = 13908000+ (0x270\*4) =13908000 + 0x9C0 = **0x139089C0**



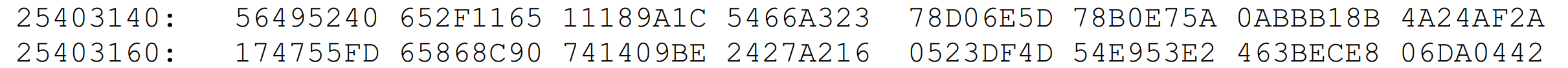
25403127 – первое значение

**25403000**

25403000 + offset = 25403000 + 15f = **2540315f**

Базовый адрес физической страницы памяти: 0x25403000

Искомый физический адрес: 0x **2540315f**



Смещение внутри строки:

Смещение=0x2540315F−0x25403140=0x1F = 32

140 — это начало второго блока в дампе

Индекс двойного слова:

0x1F÷4=7(остаток 3) – 7ое слово в строке 0x25403140 - 0x4A24AF2A

0 - 2A

1 – AF

2 – 24

3 - 4A

**Требуемые данные: 0x4A**

**Задание 05 Ответьте на следующие вопросы**

**Поясните понятие «виртуальная память».**

Виртуальная память — метод управления памятью компьютера, позволяющий выполнять программы, требующие больше оперативной памяти, чем имеется в компьютере, путём автоматического перемещения частей программы между основной памятью и вторичным хранилищем (например, жёстким диском).

Файл подкачки – это специальный файл, размещающийся на одном или нескольких дисках компьютера и предназначенный для (временного) хранения страниц виртуальной памяти, который располагается в корне жесткого диска с установленной системой и называется pagefile.sys.

**Поясните понятие «свопинг».**

Это механизм виртуальной памяти, при котором отдельные фрагменты памяти (обычно неактивные) перемещаются из ОЗУ во вторичное хранилище (жёсткий диск или другой внешний накопитель, такой как флеш-память), освобождая ОЗУ для загрузки других активных фрагментов памяти.

Основное отличие этого механизма от страничного заключается в том, что процессы перемещаются между ОЗУ и жестким диском целиком, поэтому иногда некоторые процессы могут полностью отсутствовать в ОЗУ.

**Поясните понятие «страничная память».**

В основе виртуальной памяти лежит идея, что у каждой программы имеется собственное адресное пространство, которое разбивается на участки, называемые страницами. Каждая страница представляет собой непрерывный диапазон адресов. Эти страницы отображаются на физическую память.

1.Виртуальный адрес: Адрес, который используется инструкциями программы и хранится в указателях, называется виртуальным адресом.

2.Физический адрес: Адрес, который непосредственно используется аппаратурой для доступа к данным в ОЗУ, называется физическим адресом. Процессор выставляет его на шину памяти.

**Поясните понятие MMU.**

Блок управления памятью (MMU) — это аппаратный компонент компьютера, который обрабатывает все операции с памятью и кэшированием, связанные с процессором. Другими словами, MMU отвечает за все аспекты управления памятью. Обычно он интегрирован в процессор

Работа MMU делится на три основные категории:

* Аппаратное управление памятью контролирует и регулирует использование процессором оперативной памяти и кэш-памяти.
* Управление памятью ОС обеспечивает наличие достаточных ресурсов памяти для объектов и структур данных каждой запущенной программы.
* Управление памятью приложений выделяет необходимую память каждой отдельной программе, а затем повторно использует освобожденное пространство памяти после завершения операции.

**Поясните понятие TLB.**

Буфер ассоциативной трансляции (англ. translation lookaside buffer, TLB) — специализированный кэш центрального процессора, используемый для ускорения трансляции адреса виртуальной памяти в адрес физической памяти.

TLB используется всеми современными процессорами с поддержкой страничной организации памяти. TLB содержит фиксированный набор записей (от 8 до 4096) и является ассоциативной памятью. Каждая запись содержит соответствие адреса страницы виртуальной памяти адресу физической памяти. Если адрес отсутствует в TLB, процессор обходит таблицы страниц и сохраняет полученный адрес в TLB, что занимает в 10—60 раз больше времени, чем получение адреса из записи, уже закэшированной TLB.

**Какая информация содержится в строке таблицы страниц**

Типичная запись таблицы страниц:

* Бит P (Present) определяет наличие данной страницы или таблицы страниц в физической памяти.
* Бит RW (Read/Write) определяет, можно ли в эту страницу что-то писать (1 — можно, 0 — нельзя).
* Бит US (User/Supervisor) разрешает коду с CPL=3 (код пользователя) обращаться к этой странице (при US=1).
* Бит PWT (Page write through) — запрещение кэша записи
* Бит PCD (Page cache disable) — запрещение кэширования этой страницы.
* Бит A (Accessed). Аналогично биту A в дескрипторе сегмента, этот бит никак не влияет на работу со страницей. Он просто устанавливается процессором при первом же обращении к этой странице (чтение, запись, выполнение).
* Бит D (Dirty, букв. грязный) используется для отслеживания изменений страницы. Аналогично биту A, устанавливается процессором, но только при записи на эту страницу.
* Бит PS (Page size, только каталог страниц) определяет размер страницы. Если сброшен, то этот элемент указывает на таблицу страниц с размером страниц 4 КБайт.
* Бит PAT (Page attribute table, только таблица страниц). — Начиная с Pentium 3 может использоваться (совместно с PCD и PWT и MSR IA32\_CR\_PAT) для определения политики кеширования страницы (UC, WC, WT, WP, WB, UC-) наряду с MTRR
* Бит G (Global). Если этот бит установлен, то адрес страницы (или таблицы страниц) никогда не удаляется из TLB кэша.

**Поясните принцип применения хэш-таблиц.**

Альтернатива постоянно растущим уровням иерархии страничной адресации называется инвертированная таблица страниц. В данной конструкции имеется одна запись для каждого страничного блока в оперативной памяти, а не одна запись на каждую страницу в виртуальном адресном пространстве.

Применение хэш-таблиц в контексте инвертированных таблиц страниц позволяет эффективно сопоставлять виртуальные адреса с физическими при минимальных затратах на хранение и поиск.

Основные принципы применения хэш-таблиц

1. Виртуальные адреса проходят через хэш-функцию, которая вычисляет индекс строки в хэш-таблице. Эта операция выполняется быстро и занимает фиксированное время.
2. Если несколько виртуальных страниц дают одинаковое хэш-значение (коллизия), они объединяются в связный список (цепочку) или хранятся в другой структуре данных, связанной с этим хэш-значением.
3. Если размер хэш-таблицы соответствует количеству физических страниц, средняя длина цепочки равна единице, что делает поиск практически мгновенным.
4. Когда происходит запрос на преобразование виртуального адреса, сначала выполняется хэширование. Затем хэш-таблица просматривается для поиска записи с нужным виртуальным адресом. Как только запись найдена, из нее извлекается номер физического страничного блока.
5. После успешного поиска в хэш-таблице пара (виртуальный адрес, физический адрес) добавляется в TLB. Это минимизирует дальнейшие обращения к хэш-таблице для тех же виртуальных адресов.

**Поясните применение «инвертированной таблицы физических» страничной памяти.**

Инвертированная таблица физических страниц (Inverted Page Table, IPT) — это структура данных, используемая в операционных системах для управления виртуальной памятью, которая помогает отслеживать соответствие между виртуальными страницами и физическими страницами памяти.

Инвертированная таблица использует один глобальный индекс для всей системы, который отображает физические страницы на виртуальные.

Каждая запись в инвертированной таблице содержит:

* Номер физической страницы.
* Соответствующий виртуальный адрес, который указывает, какой процесс и какая виртуальная страница использует эту физическую страницу.

Таким образом, инвертированная таблица имеет только одну запись для каждой физической страницы, а не для каждой виртуальной страницы, что позволяет значительно снизить требования к памяти.

**Windows: перечислите области адресного пространства (от младших к старшим адресам) и поясните их назначения.**

1. Секция данных, специфичная для потока (Thread Local Storage, TLS) Хранит данные, уникальные для каждого потока. Эти данные изолированы между потоками и используются для переменных, специфичных для текущего потока.

2. Стек потока. Область памяти, выделенная для хранения данных вызовов функций, локальных переменных и параметров.

3. Секция кода приложения.Содержит исполняемые инструкции программы. Здесь располагается код, который выполняется процессором.

4. Секция .xdata. Используется для обработки исключений. Содержит указатели на обработчики исключений (try-catch), фильтры и блоки finally.

5. Секция указателей на конструкторы и деструкторы статических объектов. Содержит адреса функций для инициализации (конструкторов) и очистки (деструкторов) глобальных и статических объектов.

6. Секция перемещаемых записей. Хранит данные, которые могут перемещаться или изменяться во время выполнения программы.

7. Секция инициализированных данных приложения. Хранят глобальные и статические переменные с заданными начальными значениями.

8. Секция неинициализированных данных приложения (.bss). Хранят глобальные и статические переменные, которые не были инициализированы. Эти переменные инициализируются нулями перед запуском программы.

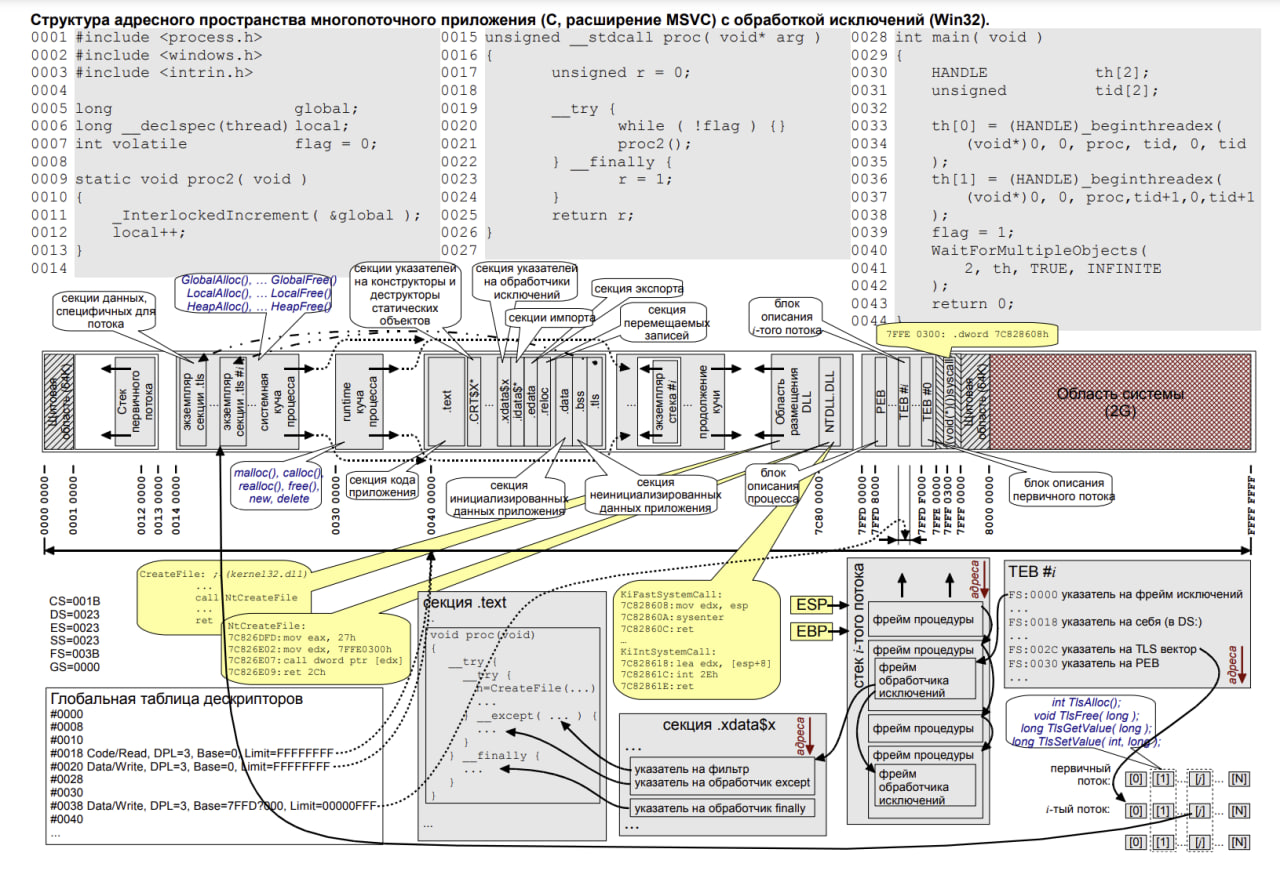
9. Секция связывания процедур. Хранит таблицы связывания, которые используются для вызовов функций (например, динамически связываемых функций из DLL).

10. Секция данных, специфичных для первичного потока. Данные, используемые исключительно основным (первичным) потоком программы.

11. Блок описания i-того потока. Хранит данные, относящиеся к конкретному потоку, такие как его стек, обработчики исключений и другая информация.

12. Блок описания первичного потока. Содержит информацию о первичном (главном) потоке программы, аналогично блоку описания i-того потока.

13. Общая куча процесса. Используется для динамического выделения памяти с помощью функций malloc, calloc, HeapAlloc или new. Хранит данные, которые живут до явного освобождения.

14. Системная область (Ring 0). Находится в старших адресах и зарезервирована для ядра операционной системы. Содержит данные ядра, драйверов и системных служб. Приложения из пользовательского пространства не имеют прямого доступа к этой области.

**Windows: какой стандартный начальный размер области heap?**

По умолчанию – 1MB, из них 4K сразу забирает процесс.

**Windows: какой стандартный размер области памяти stack?**

Стек — это сегмент памяти, в котором данные, такие как локальные переменные и вызовы функций, добавляются и/или удаляются по принципу «последним пришел — первым вышел» (LIFO).

Однако, в отличие от массива, вы получаете доступ (чтение или запись) к данным только на «верхней части» стека. Чтение из стека называется «извлечение» (pop), а запись в стек — «вталкивание» (push). Стек также известен как очередь LIFO (последним пришел — первым ушел), поскольку значения извлекаются из стека в порядке, обратном тому, в котором они помещаются в него. Извлеченные данные исчезают из стека.

**Windows: что такое «рабочее множество»? поясните принцип управления рабочим множеством с помощью OS API.**

Набор страниц, который процесс использует в данный момент, известен как рабочий набор.

Если в памяти находится весь рабочий набор, процесс будет работать, не вызывая многочисленных ошибок отсутствия страниц, пока не перейдет к другой фазе выполнения.

Если объем доступной памяти слишком мал для размещения всего рабочего набора, процесс вызовет множество ошибок отсутствия страниц и будет работать медленно, поскольку выполнение команды занимает всего несколько наносекунд, а считывание страницы с диска — обычно 10 мс.

Чтобы получить текущий размер рабочего набора указанного процесса, используйте функцию GetProcessMemoryInfo. Чтобы получить или изменить минимальный и максимальный размер рабочего набора, используйте функции GetProcessWorkingSetSizeEx и SetProcessWorkingSetSizeEx.

Процесс очищает рабочий набор, вызывая функцию EmptyWorkingSet.

**Windows: что такое «heap»? Что такое «heap процесса»? Что такое «пользовательская heap»? Поясните принцип устройства heap.**

Куча — это сегмент памяти, для которого не устанавливается постоянный размер перед компиляцией и который может динамически управляться программистом. Размер кучи приложения определяется физическими ограничениями оперативной памяти и обычно намного больше размера стека.

Heap процесса — это область динамической памяти, выделяемая для процесса в его виртуальном адресном пространстве, которая используется для хранения объектов и данных с динамическим временем жизни (например, через вызовы malloc или new). Эта память управляется самим процессом с использованием предоставленных ОС механизмов.

Пользовательская heap:

* Создается программистом вручную (например, через HeapCreate в Windows).
* Позволяет организовать кастомное управление памятью (например, для потоков, пулов).
* HeapCreate создает отдельный heap, которым можно управлять через функции HeapAlloc, HeapFree и т.д.

Heap организован как область памяти, состоящая из нескольких блоков:

* Свободные блоки (Free Blocks): Непрерывные участки памяти, доступные для выделения.
* Занятые блоки (Allocated Blocks): Участки памяти, выделенные под данные программы.

Функция, подобная malloc(), выполняет примерно следующие действия:

* просматривает список занятых/свободных областей памяти, размещённых в куче, в поисках свободной области подходящего размера;
* в случае нехватки свободной памяти может запросить дополнительную память у ОС;
* добавляет найденную область в список занятых областей (или помечает область как занятую);
* возвращает указатель на начало области памяти;

Функция, подобная free(), выполняет примерно следующие действия:

* просматривает список занятых/свободных областей памяти, размещённых в куче, в поисках указанной области;
* удаляет из списка занятых областей памяти найденную область;
* добавляет найденную область в список свободных областей (или помечает область как свободную).

**Linux: перечислите области адресного пространства (от младших к старшим адресам) и поясните их назначения.**

1. Секция определения символов. Содержит таблицы символов и метаданные, которые используются для разрешения ссылок на функции и переменные во время компиляции или загрузки программы.

2. TLS-данные (Thread Local Storage). Данные, уникальные для каждого потока. Используются для переменных, которые должны быть изолированы между потоками. В Linux обращение к этим данным осуществляется с помощью смещений через специальные регистры.

3. Структуры, описывающие обработку исключений С++ (семантика Java). Хранят информацию, необходимую для обработки исключений, такие как указатели на обработчики try-catch и finally.

4. Секция указателей на конструкторы и деструкторы статических объектов. Содержит указатели на функции, отвечающие за инициализацию и разрушение глобальных и статических объектов.

5. Секция перемещаемых записей. Хранит данные, которые могут быть изменены или перемещены во время выполнения программы.

6. Блок описания i-того потока. Содержит информацию о текущем потоке, такую как его стек, идентификатор и данные, специфичные для потока.

7. Секции данных, специфичных для первичного потока. Данные, которые используются исключительно основным потоком программы.

8. Блок описания первичного потока. Содержит данные, аналогичные описанию других потоков, но относящиеся к главному (первичному) потоку.

9. Секция связывания процедур. Хранит таблицы, обеспечивающие динамическое связывание процедур (например, функций из динамических библиотек, таких как .so файлы).

10. Секция .text. Содержит исполняемый код программы, который выполняется процессором.

11. Секция .gcc\_except\_table. Таблицы, необходимые для обработки исключений. Хранят информацию о защищённых блоках кода и обработчиках ошибок.

12. Секция инициализированных данных приложения (.data). Хранит глобальные и статические переменные, которым заданы начальные значения.

13. Секция неинициализированных данных приложения (.bss). Хранит глобальные и статические переменные, которые не были инициализированы. Эти переменные автоматически инициализируются нулями.

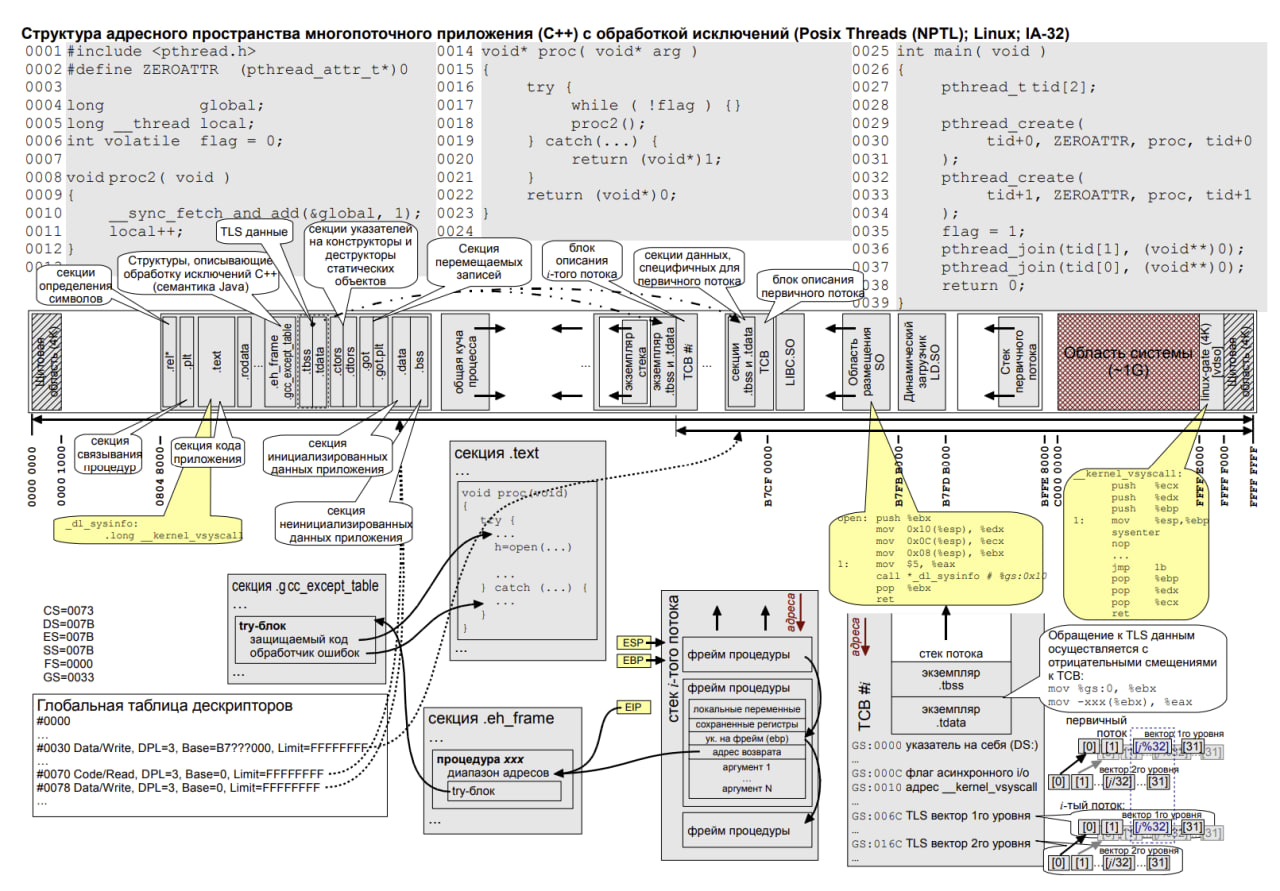
14. Общая куча процесса. Используется для динамического выделения памяти с помощью функций вроде malloc или new. Данные здесь сохраняются до явного освобождения.

15. Стек потока. Хранит данные вызовов функций, локальные переменные и параметры. Включает:

* Фреймы процедур: Данные текущих функций, такие как адрес возврата и параметры вызова.
* Указатели на исключения: Хранят информацию о текущем обработчике исключений.

16. Область системных вызовов (syscall) Зарезервирована для взаимодействия с ядром. Здесь обрабатываются системные вызовы (syscall), включая обращение к ресурсам и взаимодействие с оборудованием.

17. Область системы (1 ГБ) Находится в старших адресах и зарезервирована для ядра операционной системы. Здесь хранятся данные ядра, драйверов и системных служб. Приложения из пользовательского пространства не имеют прямого доступа к этой области.



**Linux: в какой части адресного пространства выделяется памяти с помощью функций malloc, calloc?**

В Linux память, выделяемая с помощью функций malloc и calloc, выделяется в куче процесса (Heap), которая находится в средней части виртуального адресного пространства между сегментом данных и стеком.

* Малые и средние запросы памяти (malloc, calloc) выделяются в куче.
* Большие запросы памяти обрабатываются через mmap в области отображений.

Для крупных блоков памяти (обычно больше определенного порога, например, 128 KB) malloc может использовать системный вызов mmap.

Память, выделенная через mmap, размещается в области анонимных отображений (в пространстве mmap), которая располагается между кучей и стеком.