---

## 1. Расчет размера блока памяти в `StartAnotherProgram`

### Код в `StartAnotherProgram`

Рассмотрим фрагмент кода, связанный с выделением памяти:

```asm

StartAnotherProgram proc

mov ax, 4a00h

mov bx, ((csize/16) + 17) + ((dsize/16) + 17) + 1

int 21h

...

```

### Что происходит?

- \*\*Цель\*\*: Процедура использует функцию DOS `4Ah` (`mov ax, 4a00h`), чтобы изменить размер блока памяти, выделенного для текущей программы, и освободить лишнюю память перед запуском новой программы (`test.exe`).

- \*\*Почему это нужно?\*\* В DOS память ограничена, и перед запуском новой программы важно убедиться, что текущая программа занимает только необходимый объем памяти, чтобы оставить достаточно места для новой программы.

- \*\*Регистр BX\*\*: В `BX` передается размер блока памяти (в параграфах), который программа хочет сохранить для себя. Один параграф = 16 байт.

### Расчет размера в `BX`

Разберем выражение:

```asm

mov bx, ((csize/16) + 17) + ((dsize/16) + 17) + 1

```

#### Компоненты:

1. \*\*`csize`\*\*:

- Это размер секции кода, определенный в конце программы:

```asm

csize = $ - start

```

- `$` — текущий адрес в секции кода, `start` — начало секции кода. Разница дает размер кода в байтах.

- Пример: Если секция кода занимает 512 байт, то `csize = 512`.

2. \*\*`dsize`\*\*:

- Это размер секции данных, определенный как:

```asm

dsize = $ - msg1

```

- `$` — текущий адрес в секции данных, `msg1` — начало секции данных. Разница дает размер данных в байтах.

- Пример: Если секция данных занимает 256 байт, то `dsize = 256`.

3. \*\*Деление на 16 (`csize/16`, `dsize/16`)\*\*:

- В DOS память выделяется параграфами (1 параграф = 16 байт). Поэтому размер кода и данных делится на 16, чтобы перевести байты в параграфы.

- Пример:

- Для `csize = 512`: `512 / 16 = 32` параграфа.

- Для `dsize = 256`: `256 / 16 = 16` параграфов.

4. \*\*Добавление 17 параграфов\*\*:

- К каждому из размеров (`csize/16` и `dsize/16`) добавляется 17 параграфов. Это дополнительный "буфер" памяти, который включает:

- \*\*PSP (Program Segment Prefix)\*\*: DOS выделяет 256 байт (16 параграфов) для PSP каждой программы. PSP содержит командную строку, указатели на среду и другую системную информацию.

- \*\*Дополнительный параграф\*\*: 1 параграф (16 байт) добавляется для учета возможных накладных расходов или выравнивания.

- Итог: `+17` учитывает PSP (16 параграфов) и небольшой запас (1 параграф).

5. \*\*Добавление 1 параграфа\*\*:

- В конце добавляется еще 1 параграф (`+1`) для дополнительной безопасности, чтобы гарантировать, что выделенной памяти достаточно для всех нужд программы, включая стек и возможные временные данные.

#### Пример расчета

Допустим:

- `csize = 512` байт (размер кода).

- `dsize = 256` байт (размер данных).

Шаги:

1. Код: `csize/16 = 512/16 = 32` параграфа.

2. Данные: `dsize/16 = 256/16 = 16` параграфов.

3. Добавляем 17 параграфов для кода: `32 + 17 = 49`.

4. Добавляем 17 параграфов для данных: `16 + 17 = 33`.

5. Суммируем и добавляем 1: `49 + 33 + 1 = 83` параграфа.

Итог: `BX = 83` (83 параграфа, или 83 \* 16 = 1328 байт).

### Зачем это нужно?

- Функция `4Ah` освобождает память, которая не нужна текущей программе, чтобы новая программа (`test.exe`) могла загрузиться и работать.

- Если не освободить лишнюю память, может не хватить места для `test.exe`, и запуск завершится ошибкой.

### Что будет, если расчет неверный?

- \*\*Слишком маленький размер\*\*: Программа может обрезать собственные данные или код, что приведет к сбою.

- \*\*Слишком большой размер\*\*: Программа займет больше памяти, чем нужно, и может не оставить достаточно места для `test.exe`, вызвав ошибку при запуске.

---

## 2. Что такое EPB (Environment Parameter Block)?

### Простое объяснение

EPB (Environment Parameter Block) — это структура данных, которую DOS использует для передачи информации при запуске новой программы. Простыми словами, это "набор инструкций" для DOS, который говорит:

- Какую программу запускать.

- Какие параметры передать этой программе.

- Где находится окружение (environment) программы.

- Какие дополнительные настройки использовать.

В вашей программе EPB нужен, чтобы корректно запустить `test.exe` с параметрами, которые вы сформировали (например, `hello world` из файла).

### Структура EPB в `lab4.asm`

Ваша программа определяет EPB следующим образом:

```asm

EPB dw 0 ; Сегмент окружения (environment segment)

cmd\_off dw offset parameters ; Смещение строки параметров

cmd\_seg dw ? ; Сегмент строки параметров (устанавливается позже)

fcb1 dd ? ; Первый FCB (File Control Block), не используется

fcb2 dd ? ; Второй FCB, не используется

```

#### Компоненты EPB:

1. \*\*Сегмент окружения (`dw 0`)\*\*:

- Указывает, где находится блок окружения (набор переменных, таких как `PATH`, `COMSPEC` и т.д.).

- В вашем случае установлено `0`, что означает использование того же окружения, что и у текущей программы (`lab4.exe`).

2. \*\*Смещение строки параметров (`cmd\_off dw offset parameters`)\*\*:

- Указывает адрес строки параметров, которую вы хотите передать новой программе.

- В вашем случае это `parameters`, где хранится строка, например, `hello world`.

3. \*\*Сегмент строки параметров (`cmd\_seg dw ?`)\*\*:

- Указывает сегмент памяти, где находится строка параметров.

- В процедуре `Main` вы устанавливаете его равным сегменту данных (`mov cmd\_seg, ax`).

- Это нужно, чтобы DOS знал, где искать `parameters`.

4. \*\*FCB (File Control Blocks, `fcb1 dd ?`, `fcb2 dd ?`)\*\*:

- FCB — это устаревшая структура для работы с файлами в DOS.

- В вашей программе они не используются (зарезервированы как заглушки), так как современные программы (включая вашу) используют файловые дескрипторы (`OpenFile`, `ReadFile`).

#### Размер EPB

Размер EPB определяется как:

```asm

EPB\_len dw $ - EPB

```

- `$` — текущий адрес, `EPB` — начало структуры. Разница дает размер в байтах (в данном случае 10 байт: 2 + 2 + 2 + 4 + 4).

### Зачем EPB в вашей программе?

- EPB используется в функции DOS `4Bh` (`mov ax, 4b00h`), которая запускает `test.exe`.

- Без EPB DOS не знал бы:

- Какие параметры передать `test.exe` (например, `hello world`).

- Как настроить окружение для новой программы.

- В вашем случае EPB передает строку параметров из `parameters`, сформированную из содержимого файла (например, `input.txt`).

### Как это работает?

1. В `StartAnotherProgram` вы вызываете:

```asm

mov ax, 4b00h

lea dx, program ; Указываем имя программы ("test.exe")

lea bx, EPB ; Указываем адрес EPB

int 21h

```

2. DOS читает EPB и:

- Загружает `test.exe` в память.

- Передает строку параметров из `parameters` (например, `hello world`) в командную строку новой программы (по адресу `ES:80h` в `test.exe`).

- Устанавливает окружение (в данном случае то же, что у `lab4.exe`).

### Пример

Допустим:

- В `parameters` хранится строка `hello world` (сформирована из файла `input.txt`).

- `cmd\_off` указывает на `parameters`, а `cmd\_seg` — на сегмент данных.

Когда `test.exe` запускается:

- DOS копирует `hello world` в командную строку `test.exe` (по адресу `ES:80h`).

- `test.asm` читает эту строку и выводит `hello world` на экран.

---

## Итог

### Расчет размера блока памяти

- В `StartAnotherProgram` размер памяти рассчитывается как:

```asm

((csize/16) + 17) + ((dsize/16) + 17) + 1

```

- `csize` и `dsize` — размеры кода и данных в байтах.

- Деление на 16 переводит байты в параграфы.

- `+17` добавляет 16 параграфов для PSP и 1 параграф для запаса.

- `+1` — дополнительный параграф для безопасности.

- Пример: Для `csize = 512` и `dsize = 256` выделяется 83 параграфа (1328 байт).

### EPB

- \*\*Что это?\*\* Структура, которая говорит DOS, как запустить новую программу, включая параметры и окружение.

- \*\*Зачем в вашей программе?\*\* Чтобы передать параметры (например, `hello world`) из `lab4.asm` в `test.exe` и настроить запуск.

- \*\*Как работает?\*\* Указывает DOS, где взять строку параметров (`parameters`) и какое окружение использовать (в вашем случае — то же, что у `lab4.exe`).

Если остались вопросы или нужно что-то уточнить (например, конкретный пример с числами или другой аспект), дайте знать!