Основы TASM

Доп. семинар #1

Почему Dos? Историческая справка

DOS (Disk Operating System) — это семейство операционных систем, использовавшихся на персональных компьютерах, главным образом в 1980–1990-х годах. DOS обеспечивает базовое управление файлами, запуск программ и работу с дисками.

Мы будем делать первые доп. задачи на DOS потому что:

1. Исторически более ранняя система с более простыми системными вызовами
2. В ней практически отсутствуют механизмы защиты и абстракции
3. Можно залезть внутрь системы и сломать что-то :)

Модель памяти DOS

- В Intel 8086 регистры были всего по 16 бит.
- Сколько памяти мы можем адресовать 16-битным адресом?

Модель памяти DOS

- В Intel 8086 регистры были всего по 16 бит.
- Сколько памяти мы можем адресовать 16-битным адресом?
- Всего 2^{16} бит или 32 Кбит или 4 Кбайт.

Модель памяти DOS

- В Intel 8086 регистры были всего по 16 бит.
- Сколько памяти мы можем адресовать 16-битным адресом?
- Всего 2^{16} бит или 32 Кбит или 4 Кбайт.
- Даже для тех времен это мало. Но создать бóльшие регистры нет возможности. Как решим проблему?

Модель памяти DOS

- В Intel 8086 регистры были всего по 16 бит.
- Сколько памяти мы можем адресовать 16-битным адресом?
- Всего 2^{16} бит или 32 Кбит или 4 Кбайт.
- Даже для тех времен это мало. Но создать большие регистры нет возможности. Как решим проблему?
- Давайте использовать два регистра для адресации. Один регистр адресует страницу (сегмент), а второй – только внутри сегмента.

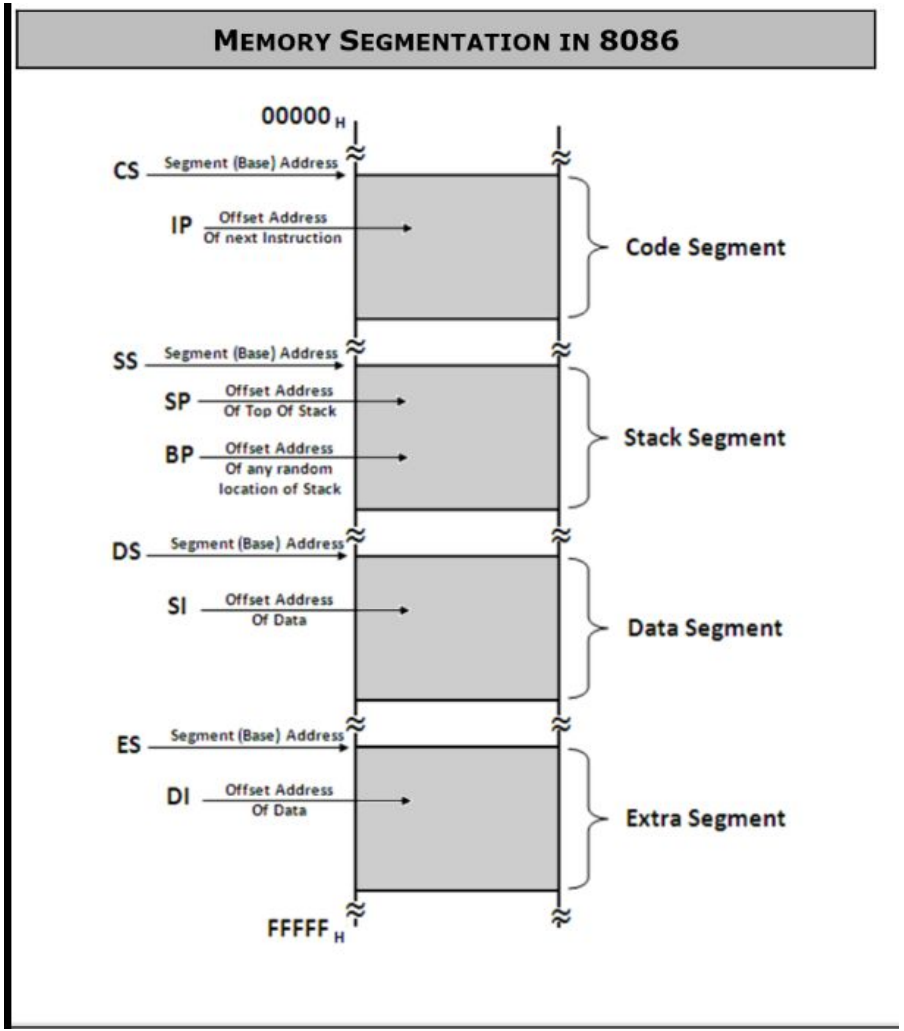
Модель памяти DOS

Адреса сегментов хранятся в специальных регистрах:

- DS – data segment
- CS – code segment
- SS – stack segment
- ES – extra segment

Физический адрес считается так:

P_ADDR = SEG * 10 + OFFSET



https://blogsystem5.substack.com/p/dos-memory-models

Видеопамять DOS

- В DOS одновременно может работать только одна программа
- Нет необходимости в виртуальной памяти
- Мы работаем напрямую с физической памятью
- Видеопамять – то что выводится на экран – тоже часть памяти
- Можем ли мы напрямую записать в видеопамять?

Видеопамять DOS

- В DOS одновременно может работать только одна программа
- Нет необходимости в виртуальной памяти
- Мы работаем напрямую с физической памятью
- Видеопамять – то что выводится на экран – тоже часть памяти
- Можем ли мы напрямую записать в видеопамять? Да!
- Но безопасно ли это?

Видеопамять DOS

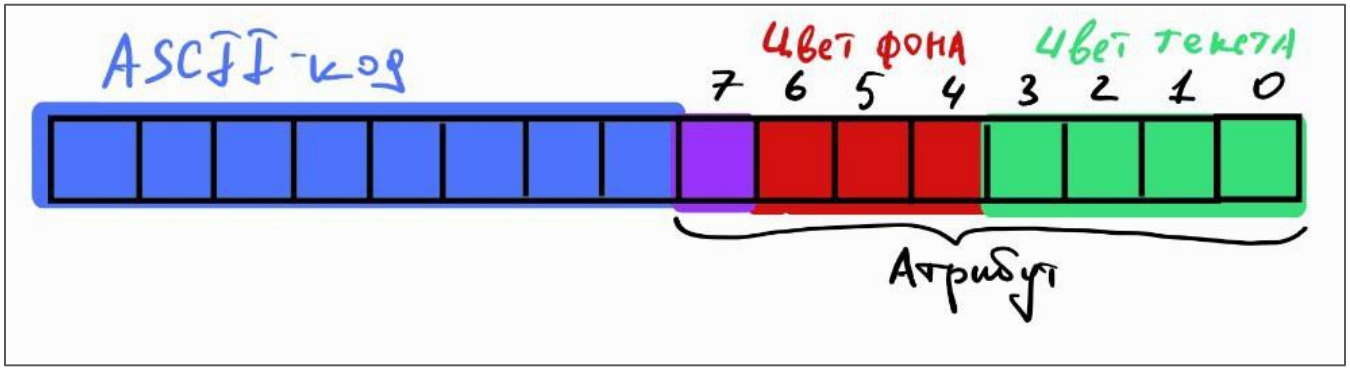
- В DOS одновременно может работать только одна программа
- Нет необходимости в виртуальной памяти
- Мы работаем напрямую с физической памятью
- Видеопамять – то что выводится на экран – тоже часть памяти
- Можем ли мы напрямую записать в видеопамять? Да!
- Но безопасно ли это? Конечно нет!
- Волнует ли нас это?

Видеопамять DOS

- В DOS одновременно может работать только одна программа
- Нет необходимости в виртуальной памяти
- Мы работаем напрямую с физической памятью
- Видеопамять – то что выводится на экран – тоже часть памяти
- Можем ли мы напрямую записать в видеопамять? Да!
- Но безопасно ли это? Конечно нет!
- Волнует ли нас это? Конечно нет!

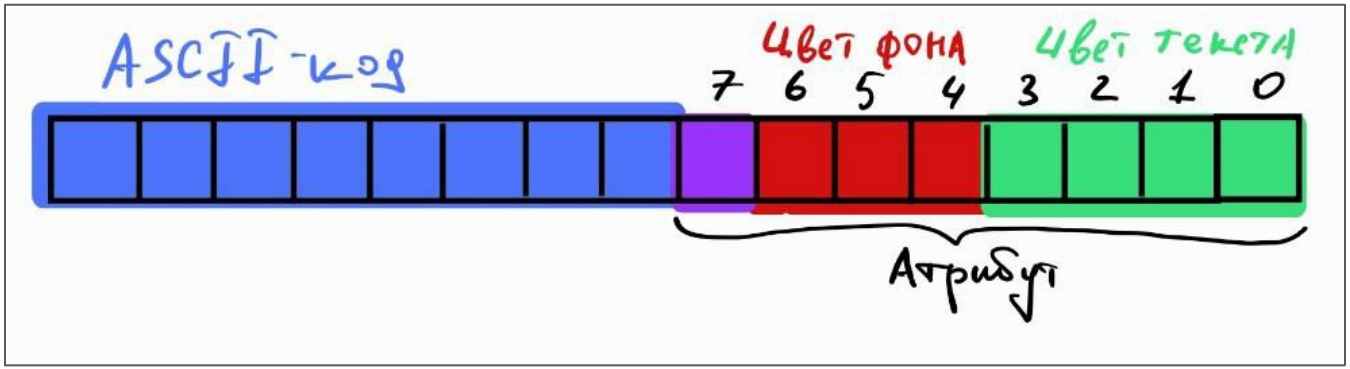
Видеопамять DOS

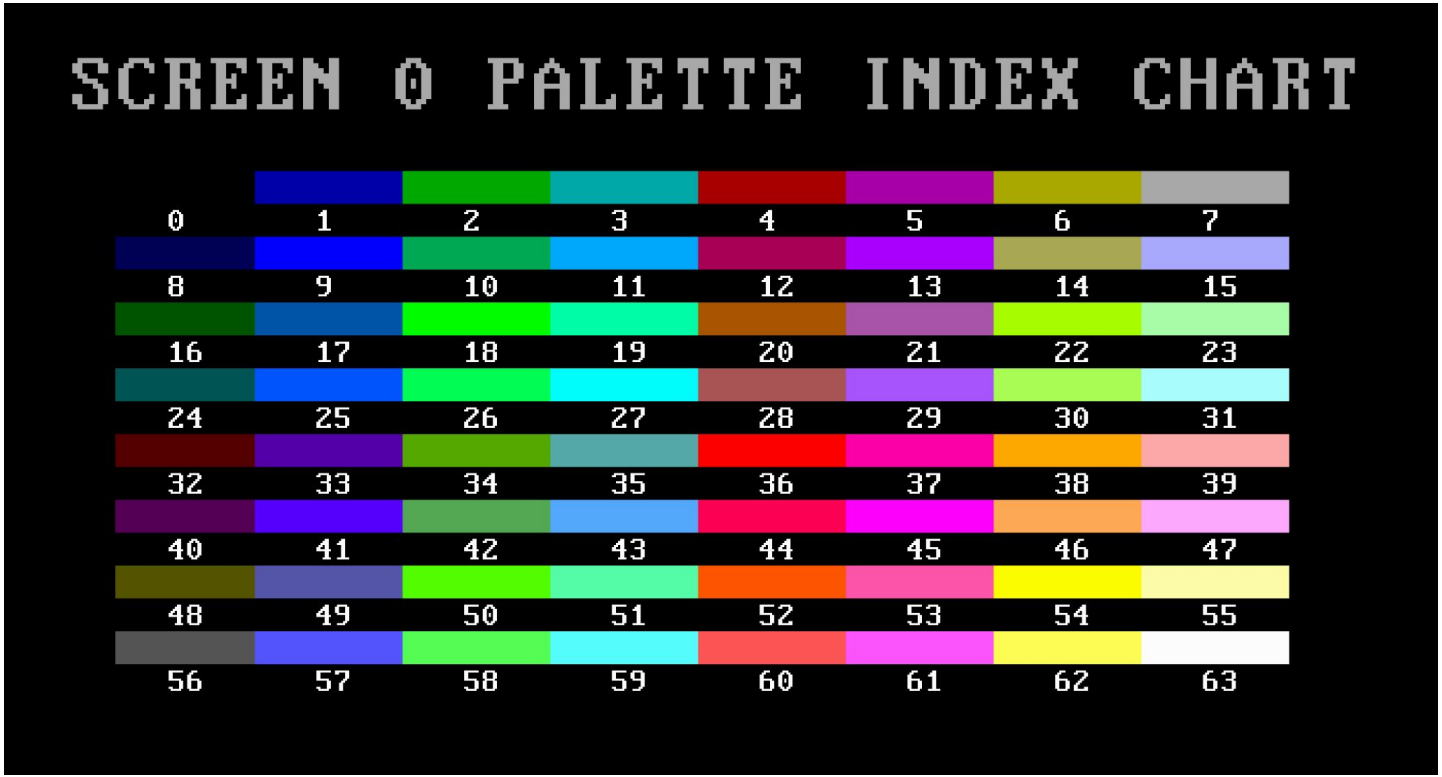
- Видеопамять DOS представлена как линеаризованный двумерный массив размером 80 x 25 – размер консоли
- Сегмент видеопамяти имеет адрес 0B800h
- Каждая ячейка кодируется двумя байтами: ASCII-код, Атрибут



Видеопамять DOS

- С ASCII-кодом все понятно
- Биты атрибута:
  - 7 – моргание / интенсивность
  - 6-4 – цвет фона
  - 3-0 – цвет текста




SCREEN 0 PALETTE INDEX CHART
0 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63

Путь к простейшей программе

Узнав все про память, мы (почти) готовы написать программу.

Выведем на экран Hello world.

Эта сложнейшая задача ставит перед нами два фундаментальных вопроса:

1. Как выводить на экран
2. Как завершать программу

Путь к простейшей программе. Как выводить на экран

Узнав про видеопамять, мы знаем как выводить на экран. Этого уже достаточно. Но это не всегда удобно:

- мы можем перетереть уже отрисованные части экрана
- нам нужно самим центрировать текст
- придется значительно менять программу при изменении текста

Путь к простейшей программе. Как выводить на экран

Dos решает все эти проблемы, предоставляя нам функции вывода на экран.

Чтоб это сделать, нужно:

1. остановить нашу программу, сохранить все ее данные (значения регистров, памяти)
2. переместиться внутрь кода Dos, исполнить функцию вывода
3. переместиться обратно в нашу программу, восстановить все ее данные

Путь к простейшей программе. Как выводить на экран

К счастью, такой механизм уже реализован и называется **прерывания.**

Прерывание – это механизм, с помощью которого процессор может временно остановить текущую программу и передать управление специальной функции (обработчику прерывания)

Путь к простейшей программе. Как выводить на экран

Прерывание – это механизм, с помощью которого процессор может временно остановить текущую программу и передать управление специальной функции (обработчику прерывания)

Прерывания используются для:

- обращения к аппаратуре (клавиатура, диск, таймер)
- вызова системных функций
- обработки ошибок.

Прерывания бывают:

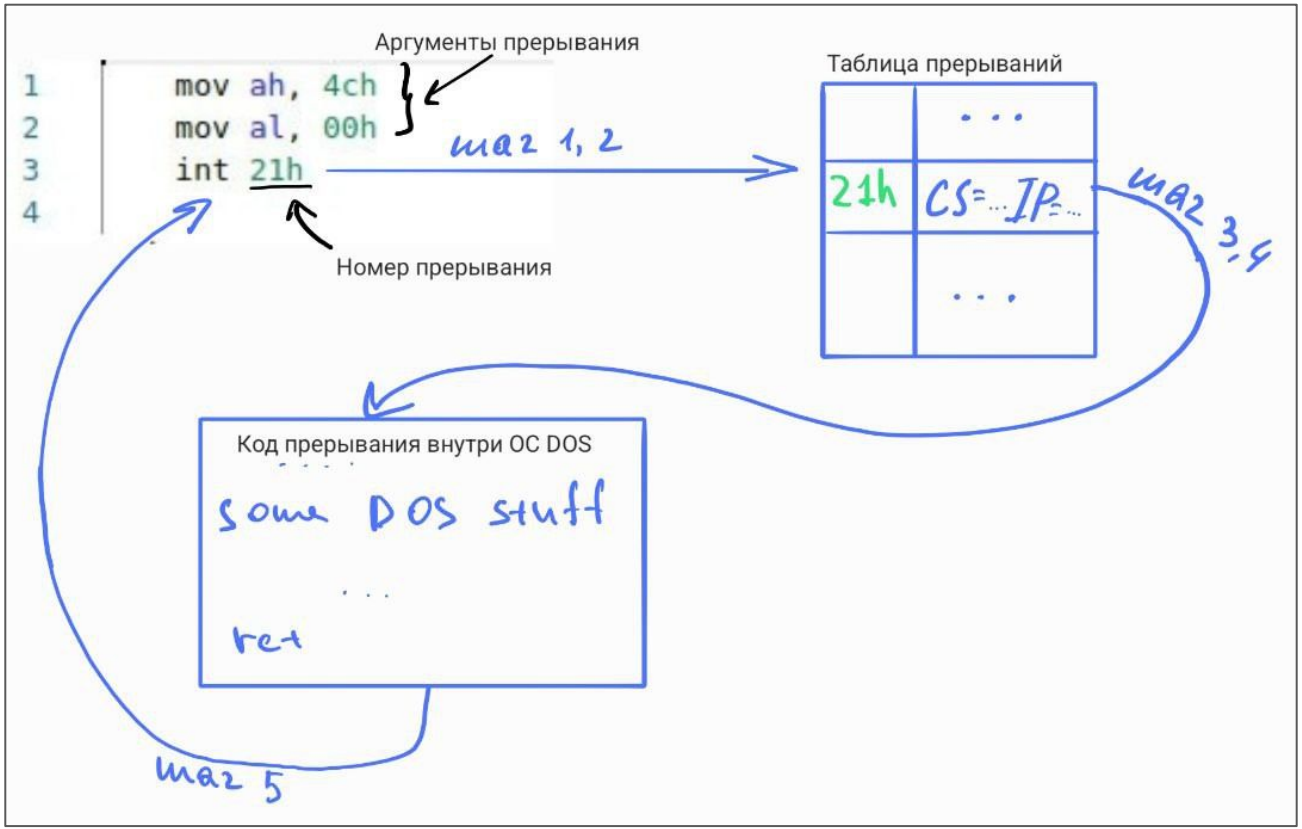
- Аппаратные (hardware): от клавиатуры, таймера, контроллеров.
- Программные (software): вызываются командой int (от слова interrupt)

Путь к простейшей программе. Как выводить на экран

Работает прерывание так:

1. CPU останавливает программу, сохраняет регистры в стек
2. CPU идет по адресу 0000:0000, там лежит таблица прерываний
3. Элемент таблицы содержит адрес функции, которую нужно вызывать (значение CS, IP)
4. CPU выставляет CS и IP из таблицы, начинает исполнять код
5. После исполнения кода, CS, IP выставляются обратно в пользовательскую программу. Продолжается исполнение пользовательской программы

Путь к простейшей программе. Как выводить на экран



Путь к простейшей программе. Как выводить на экран

- Прерываний очень много. Однажды мы испортим таблицу прерывания, записав в нее свои функции, но до этого еще далеко
- Все DOS-функции (системные вызовы) находятся внутри прерывания номер 21h. Пока будем работать только с ним.
- Это прерывание работает так:
  - В регистр AH кладем номер DOS-функции
  - В остальные регистры другие аргументы для DOS-функции

Путь к простейшей программе. Как выводить на экран

Внутри функции 21h все выглядит примерно так.

Само прерывание имеет номер 21h.

В регистре AH передается номер DOS-функции:

например номер 09h – функция печати строки

а номер 4Ch – функция завершения программы

```
1   void interrupt_21H() {
2       switch (AH) {
3           case 00h:
4               ...
5           ...
6           case 09h:
7               PRINT_STRING(STRING=DS:DX);
8
9           case 4ch:
10              EXIT(RET_CODE=AL);
11      }
12  }
13
14
```

Путь к простейшей программе. Код

После небольшого введения, мы готовы написать нашу первую бесполезную программу

Разберем ее построчно

```
1   .model tiny
2   .code
3   org 100h
4
5   start:
6    mov ah, 9
7    mov dx, offset msg
8    int 21h
9    mov ax, 4c00h
10   int 21h
11
12
13  .data
14  msg     db "Hello world$"
15  end start
16
17
```

Путь к простейшей программе. Код

.model tiny – директива ассемблера, которая говорит, что вся наша программа ляжет в один сегмент. Подробнее на доп. семинаре про COM-файлы

```
1   .model tiny
2   .code
3   org 100h
4
5   start:
6    mov ah, 9
7    mov dx, offset msg
8    int 21h
9    mov ax, 4c00h
10   int 21h
11
12
13  .data
14  msg     db "Hello world$"
15  end start
16
17
```

Путь к простейшей программе. Код

.org 100h – точка начала исполнения программы.

При старте программы регистр оффсета IP = 100h. Память 0-99h занята аргументами командной строки и другой служебной информацией

```
1   .model tiny
2   .code
3   org 100h
4
5   start:
6    mov ah, 9
7    mov dx, offset msg
8    int 21h
9    mov ax, 4c00h
10   int 21h
11
12
13  .data
14  msg     db "Hello world$"
15  end start
16
17
```

Путь к простейшей программе. Код

.code – аналог .text из NASM

.data – аналог .data из NASM

start – точка входа,
расположена по адресу 100h

```
1   .model tiny
2   .code
3   org 100h
4
5   start:
6    mov ah, 9
7    mov dx, offset msg
8    int 21h
9    mov ax, 4c00h
10   int 21h
11
12
13  .data
14  msg     db "Hello world$"
15  end start
16
17
```

Путь к простейшей программе. Код

Строки 6 – 8:

1. в AH кладем номер DOS-функции печати строки – 9
2. В DX кладем указатель на строку
3. вызываем прерывание 21h

```
1   .model tiny
2   .code
3   org 100h
4
5   start:
6    mov ah, 9
7    mov dx, offset msg
8    int 21h
9    mov ax, 4c00h
10   int 21h
11
12
13  .data
14  msg     db "Hello world$"
15  end start
16
17
```

Путь к простейшей программе. Код

Строки 9 – 10:

1. в AH кладем номер DOS-функции завершения программы – 4C
2. В AL кладем код возврата – 00
3. То есть в AX кладем 4c00
4. вызываем прерывание 21h

```
1   .model tiny
2   .code
3   org 100h
4
5   start:
6    mov ah, 9
7    mov dx, offset msg
8    int 21h
9    mov ax, 4c00h
10   int 21h
11
12
13  .data
14  msg     db "Hello world$"
15  end start
16
17
```

Практика!

Настроенная для вас сборка DOSBOX:

1. Разархивируйте файл в корень: /home/user/
2. Зайдите в /home/user/.dosbox, откройте файл dosbox-0.74-3.conf
3. В самом низу найдите строки:

   mount d: /home/danny/programmings/DosPrograms

   mount c: /home/danny/.dosbox/

4. В первой пропишите путь к вашей рабочей папке с файлами .ASM
5. Во второй – пропишите путь к папке .dosbox с конфигом