Парсинг программы

// Объявили сегмент StKSeg, который будет выравниваться по началу параграфа (адрес кратен 16), иметь тип стек (что говорит о том, что он будет использован для стека), и иметь класс STACK (просто имя)

```
StkSeg SEGMENT PARA STACK 'STACK'
```

```
    // Объявили выделение памяти размером 200 * 1байт, с неопределенным заполнением
    DB 200h DUP (?)
```

// закрыли сегмент

```
StkSeg ENDS
```

// Объявили сегмент данных (выравнивание по 2 байта, тип не определен)

```
DataS SEGMENT WORD 'DATA'
```

// Объявляем переменную, размер 3 + длина строки

```
HelloMessage DB 13          // поместить курсор в начало строки
             DB 10          // перевести курсор на новую строку
             DB 'Hello, world!'
             DB '$'

DataS ENDS
```

// Объявили сегмент кода (выравнивание по 2 байта, типа не определен)

```
Code SEGMENT WORD 'CODE'
```

```
    // Сделали CS регистром по умолчанию для сегмента кода, и DS для данных
    ASSUME CS:Code, DS:DataS
```

// Определили метку (она будет началом программы)

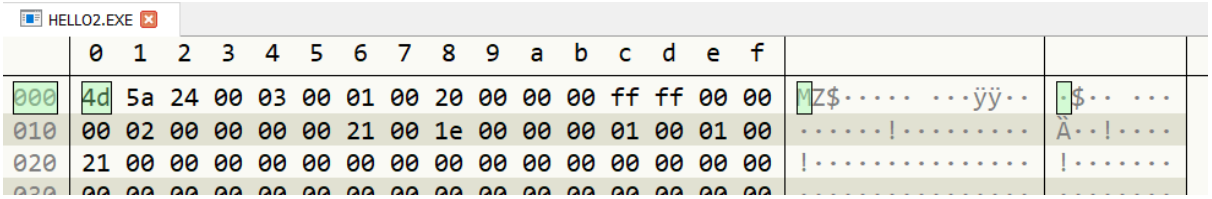
```
DispMsg:
    mov AX, DataS                   // загрузка в AX адреса сегмента данных
    mov DS, AX
    mov DX,OFFSET HelloMessage      // Положили в DS адрес строки
    mov AH,9                        // Вызов функции вывода на консоль
    int 21h                         // Прерывание для вывода
    mov AH,7                        // команда считывания символа без эха
    int 21h                         // Считали
    mov AH,4Ch                      // Завершились
    int 21h

Code ENDS
  END DispMsg
```

Варианты сборки программы:

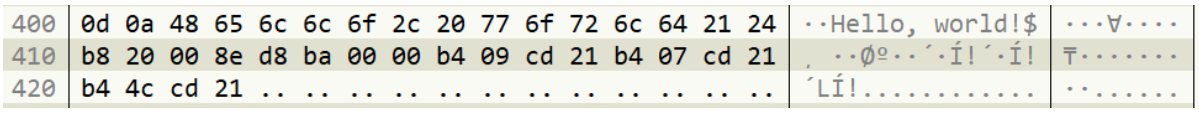
- Просто ml.exe main.asm => получим exe
- ml.exe /c main.asm + link.exe main.obj,main.com,,,nul – отдельно транслирование и линковка (получим com)

Анализ полученного exe-файла:

1. Сначала идет заголовок, который начинается с MZ (стандартный формат 16-битных файлов с расширением exe в dos). Значения дальше видимо нужно расшифровывать по таблицам (причем брать значения нужно по два, как это сделано в MZ = 4d + 5a).



2. Сама программа (“загружаемая часть”)
Видим значение строки, которое мы хотим вывести и некоторые машинные команды.



Доп

Дописать код программы так, чтобы строка выводилась три раза.

```
StkSeg SEGMENT PARA STACK 'STACK'
    DB 200h DUP (?)
StkSeg ENDS
;
DataS SEGMENT WORD 'DATA'
HelloMessage DB 13
             DB 10
             DB 'He world!'
             DB '$'
DataS ENDS
;
Code SEGMENT WORD 'CODE'
    ASSUME CS:Code, DS:DataS

mov cx, 3
DispMsg:
    mov AX,DataS
    mov DS,AX
    mov DX,OFFSET HelloMessage
    mov AH,9
    int 21h

    loop DispMsg
    mov AH,4Ch
    int 21h

Code ENDS
     END
```

Добавила mov + loop. Убрала после end имя метки (так как в этом случае mov cx не выполнялось и процесс становился бесконечным)

¯_(ツ)_/¯

Проанализировать изменение регистров в отладчике

TO DO

Примечание

Таблицы для расшифровки заголовков:
http://mzc.narod.ru/Creating/Step008.htm
https://wiki.osdev.org/MZ

00h	word	Сигнатура	Магическая сигнатура DOS-файла - два символа "MZ"
02h	word	Extra bytes	Количество байт на последней странице файла
04h	word	Pages	Количество страниц в файле
06h	word	Relocation items	Количество релокейшенов
08h	word	Header size	Размер заголовка в параграфах
0Ah	word	Minimum allocation	Мин. выделение памяти в параграфах
0Ch	word	Maximum allocation	Макс. выделение памяти в параграфах
14h	word	Initial IP	Начальное значение регистра IP
16h	word	Initial CS	Начальное (относительное) значение регистра CS
18h	word	Relocation table	Адрес на релокейшены и программу-заглушку
1Ah	word	Overlay	Количество оверлеев
1Ch	word	Overlay information	Зарезервировано
24h	word	OEMIdentifier	Для OEMInfo
26h	word	OEMInfo	Информация о программе
28h	word	Res1[10]	Зарезервировано
3Ch	dword	PEHeaderAddr	Адрес в файле заголовка PE