С чего начать?

Большой проблемой является поиск актуальной литературы на тему "Программирование на Ассемблере".
Для решения этого вопроса рекомендую посетить сайты, названные ниже.
Изучать будем FASM.

Как писать на Ассемблере в 2018 году?

В этой статье приводится соблазнительно большой список литературы, статья может претендовать на краткий обзор вопроса.
Вдобавок, на основании этой статьи можно сказать, что приличных пакетов под Ассемблер осталось "раз два и обчелся"...
Среди них:

- **MASM** - много версий, документация запутана, требует установки,нужна лицензия... Нередко можно встретить источник литературы по уже несуществующией версии, так что: "Удачи!"...
- **TASM** - когда-то был в приличном состоянии. Но сейчас ... он не обновлялся со времен 32 битного Borland, а это примерно 2008 год.
- Следующий в списке - **FASM** - когда-то был темной лошадкой. На сегодняшний момент заслуженно теснит MASM. FASM распространяется свободно, не требует установки, и, что удивительно, по сей день активно обновляется. К тому же он неимоверно складно и просто синтаксически устроен. При скачивании с официального сайта идет вместе с примерами. Есть актуальный, недавно обновленный, программистский мануал. Всего этого по-прежнему недостаточно для начала работы. Потому попробуем создать начальное ускорение.

Поэтому предложим свой взгляд на ситуацию...

- Имеет смысл посетить сайт FasmWorld. Он посвящен тренировке навыков программирования на FASM под DosBox. Там высказана мысль: "Прежде чем браться за Windows, можно поиграть с DosBox". Из этого этапа изучения Ассемблера получается неплохой экскурс в историю. Тем не менее, материал хоть и обладает редким качеством достоверности изложения, на практике половине учащихся трудно и почти невозможно изучить его в том варианте, в котором он предложен на сайте. Большие объемы теории, резко "сваливающиеся на голову", демотивируют. По этой причине будем FasmWorld использовать как дополнительный материал высокого качества.
- На одном из форумов был задан вопрос:" А есть ли руссифицированный мануал программиста под FASM?" Ответ был таков:"Нет, но подожди, сейчас перегоню через онлайн переводчик..." После этого энтузиасты, с различной степенью отклонения от исходного текста, с применением личной интерпретации, с минимальным объяснением - как этим пользоваться, стали выкладывать в Интернет свои варианты переводов... Я

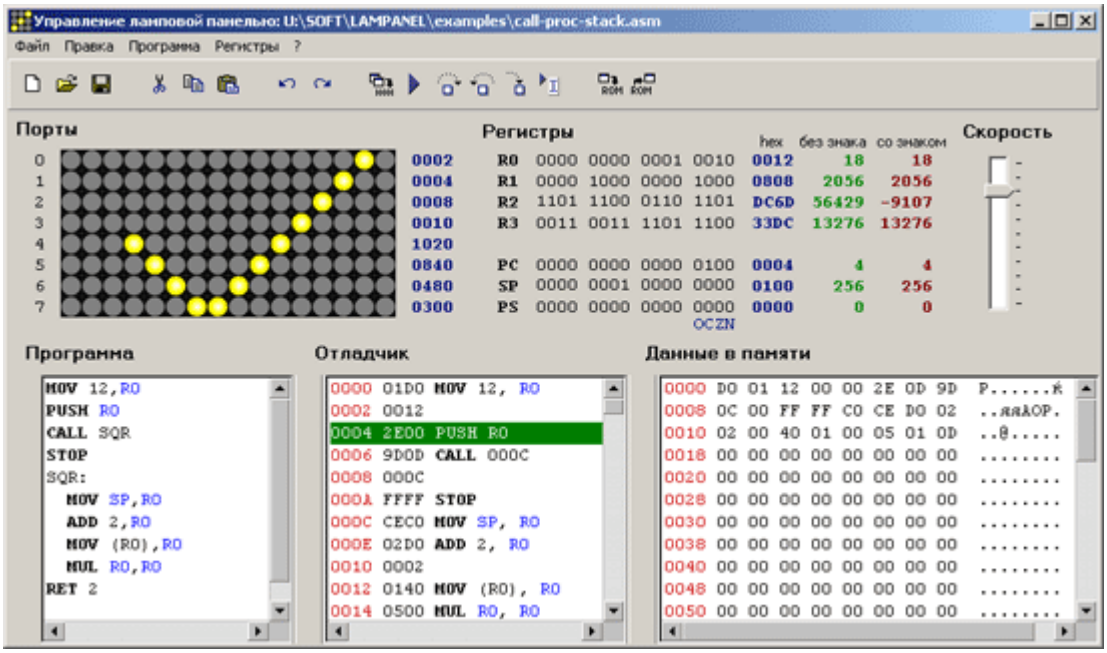
насчитал 2,3 или 4 таких варианта. Каждый из них не охватывает общую картину, но тем не менее хочется предложить один из наиболее приличных вариантов. Мануал программера.flat assembler 1.71

- В тот момент, когда программки в стиле "Привет, мир" остались позади, возникает вопрос: "Как со всем этим обращаться? Как не тратить время на часто повторяющиеся проблемы?" Именно в этот момент в дело вступают макросы. Руководство по препроцессору FASM. В программировании 2000 годов в литературе часто встречается антипатерн "GoldenHammer" или "золотой молоток", что означает решение всех проблем одним способом. Антипатерны являются обратной стороной медали патернов проектирования. Хоть в литературе и повторяют заученную фразу:"Только патерны и никаких антипатернов", - разумное применение любых приемов ещё никому не вредило! В варианте FASM, судя по всему, пошли по пути максимального упрощения, и очень талантливо решили почти все проблемы при помощи макросов.
  Да, FASM - только компилятор, не литнкер. Не объединяет заранее скомпилированные файлы. Зачем такие сложности!Проблема решается через препроцессинг, макросы и подшивание *.Inc файлов в один на этапе перед подачей на компиляцию.
  Получился поражающий простой и универсальностью инструментарий. Можно сказать, что налицо неканоническое применение антипарена "золотой молоток".
  Если освоение макросов удастся, то оно превратит процесс написания программных кодов Ассемблера из решения ребуса в увлекательное программирование на близком аналоге императивного языка высокого уровня.
  Вы удивитесь, узнав, что так было не всегда. Копнув чуть глубже Интернет-форумы, можно узнать, что MASM пестрит разношерстным синтаксисом в различных версиях на тему линкинга, предкомпиляции и сегментации памяти программы.
  Возможно, под FASM встречается куда больше примеров работающего программного кода, чем под MASM.
  В этом плане на экзамене легко узнать, кто из студентов отлынивал от работы. Некоторые граждане находят программный код, тасуют его, не глядя, и пытаются сдавать... При этом очень сильно удивляются на то, что он ещё должен компилироваться без ошибок и работать.
  Дело доходит до того, что сразу видно - какие блоки размером примерно в 10-15 строк взяты из FASM, а какие из MASM. **Делаем вывод. Программный код ничего не стоит, если он не компилируется и не запускается!!!**
  Именно этому вопросу мы и собираемся уделить особое внимание!!!

Начинаем писать

Основная проблема юного ассемблериста - это вывод на экран состояний регистров.
Чтобы это "провернуть", требуется средний уровень владения языком (не меньше).
Это первая из преград, которая преодолевается нахождением адекватной среды разработки.
EMU8086 - хотелось бы сказать, но за прошедшее десятилетие среда стала платной.
Free Pascal 3.0/Lazarus/Delphi - хотелось бы сказать, но среда не является мейнстримом дня сегодняшнего.

Visual Studio 2019/C++ - тоже обладает своими неочевидными с первого взгляда недостатками, но пользоваться мы ей будем, хоть и не с самого начала...
А потому, встречайте: ЛамПанель.



Сразу скажу, ЛамПанель ни на что не способна, кроме визуализации... Что на определенном этапе себя оправдывает.
Есть русскоязычный хелп на 2 страницы.
Есть *.PDF на 14 страниц.
Есть все, что требуется для изучения.

- Задание. Часть первая.

Нарисовать авторскую картинку.

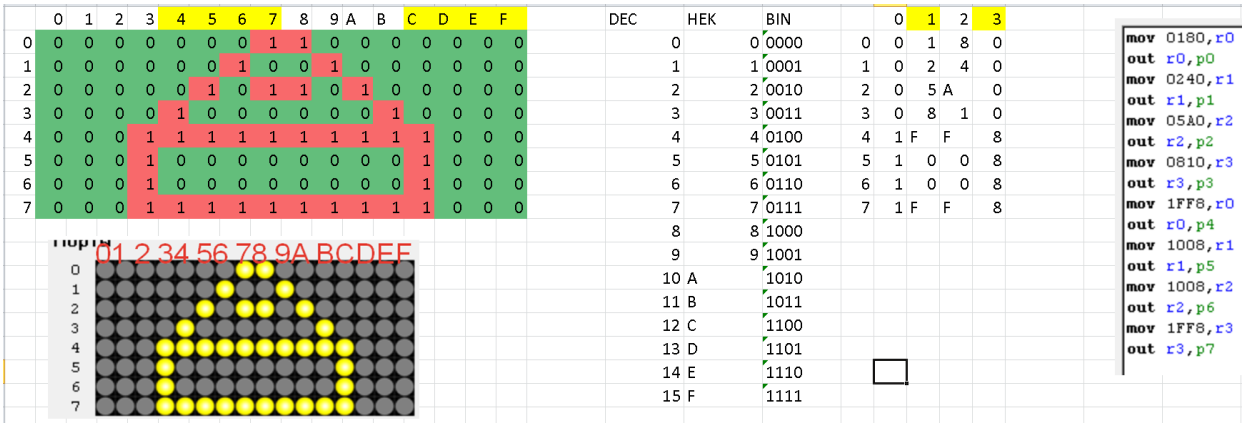
- Задание. Часть вторая.

Сделать анимацию. Циклический сдвиг снизу вверх, слева направо.

Студентам можно сказать, что есть:

- 4 регистра, r0,r1,r2,r3;
- 8 портов для работы с внешними устройствами (имитация), p0,p1,p2,p3,p4,p5,p6,p7;
- in/out - пересылка машинных слов между регистрами и портами;
- mov - команда пересылки между регистрами и оперативной памятью (об этом подробнее позже);
- есть системные процедуры, которые можно посмотреть как "программма посмотреть ПЗУ".

Подробнее о том, как сделать это...



Создаем файл Excel, делаем эскиз рисунка из нулей и единиц, переводим при помощи таблиц в 16-ричный формат (слева направо). Осталось подставить в программный код.

Пример программного кода

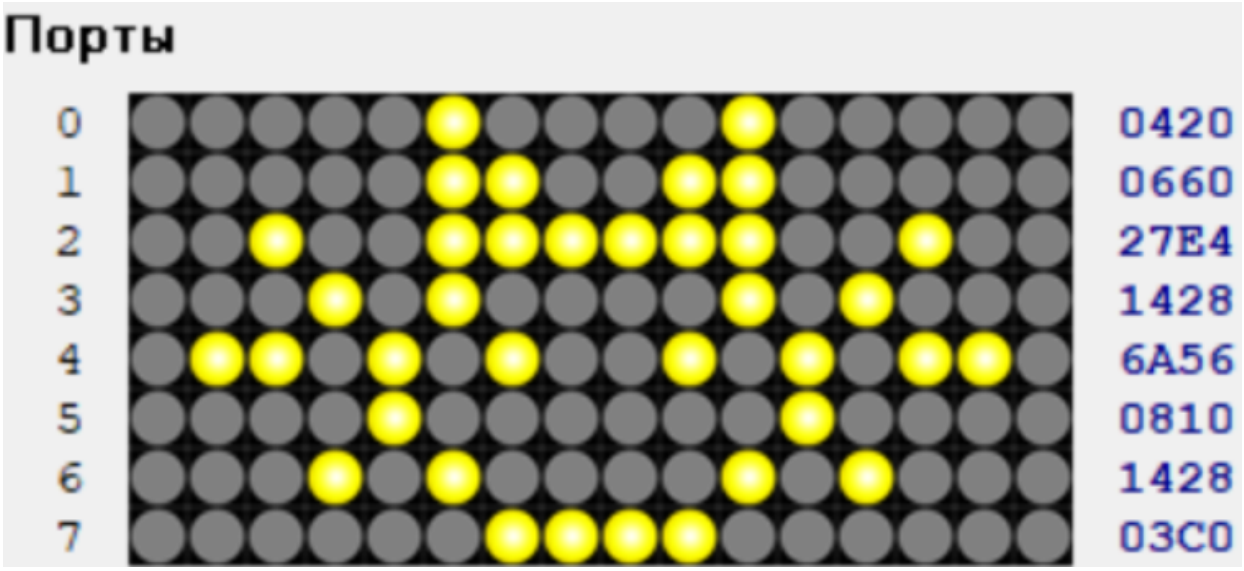
```
mov 0180,r0
out r0,p0
mov 0240,r1
out r1,p1
mov 05A0,r2
out r2,p2
mov 0810,r3
out r3,p3
mov 1FF8,r0
out r0,p4
mov 1008,r1
out r1,p5
mov 1008,r2
out r2,p6
mov 1FF8,r3
out r3,p7

m:
;дВИГАЕМ КРЫШУ
IN P0,R0
ROR 1,r0
OUT R0,P0
;
IN P1,R0
ROR 1,r0
OUT R0,P1
;
IN P2,R0
ROR 1,r0
OUT R0,P2
```

```
;
IN P3,R0
ROR 1,r0
OUT R0,P3
;
IN P4,R0
ROR 1,r0
OUT R0,P4
;
IN P5,R0
ROR 1,r0
OUT R0,P5
;
IN P6,R0
ROR 1,r0
OUT R0,P6
;
IN P7,R0
ROR 1,r0
OUT R0,P7
;
JMP m

stop
```

Кто-то из студентов подходит к вопросу творчески.



Если возникают трудности, то задание может исполняться парами студентов.
ЛамПанель также может работать с оперативной памятью, но оставим подобные вопросы до более интересного языка.
Задания хватит на несколько часов.

В дальнейшем работает правило: если что-то не получается, то можно вернуться назад, к предыдущей среде разработки, что-либо отработать там и идти дальше.

In [ ]: