

Примерный план проблемной лекции

ЛамПанель, или школьный ассемблер.

Время – от 2 до 4 часов, из них лекция - 2 часа.

Учебные цели занятия: познакомиться с обучающей программой ЛамПанель для языка Ассемблер. По образцу, в 3 этапа, начать писать программный код, получить положительный опыт от общения и написания несложных программ в ЛамПанель.

Форма проведения занятия: проблемная лекция.

Учебные задания:

- Этап первый.
 - 1) Пошагово разобрать у доски или на проекторе приведенный пример.
 - 2) Нарисовать в тетради или в EXCEL свой пиксельный рисунок 8 на 16 клеток.
 - 3) При помощи таблицы перевода перевести из бинарной системы исчисления каждую строку в шестнадцатеричную систему исчисления.
 - 4) При помощи таблицы перевода перевести из бинарной системы исчисления каждую строку в шестнадцатеричную систему исчисления. Вспомнить таблицу перевода систем исчисления.
- Этап второй
 - 1) Прочитать теоретическую справку или мануал ЛамПанель.
 - 2) Подставить полученные шестнадцатеричные коды в пример, запустить, получить рисунок в лампочках портов, исправить ошибки.
- Этап третий
 - 1) Прочитать теоретическую справку или мануал ЛамПанель.
 - 2) При помощи сдвигов, перестановок портов с использованием любого регистра сделать анимацию. Одновременное циклическое движение, на выбор: снизу вверх или сверху вниз; слева направо или справа налево.

Инструкции по проведению и ходу занятия: В начале занятия обосновываются цель и задачи проблемной лекции, определяется проблема для обсуждения. Преподаватель излагает материал, задает вопросы студентам, посвященные проблематике лекции, затем начинается дискуссия по поставленной проблеме; вторая половина занятия - групповая практика на ПК.

Методические рекомендации: соблюдайте последовательность действий; не перескакивайте на следующий этап, не пройдя предыдущий; обратите внимание - есть секретный компактный способ решения поставленной задачи через системные процедуры (system [16 ричный код процедуры]), об этом не подсказывайте учащимся, это награда за внимательное прочтение методички ЛамПанели.

Вопрос для самоконтроля:

Учим команды: mov,in, out, jmp, :metko,system,ror,rol,shr,shl и так далее... - все команды, что

встречаются в программном коде.

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 несуществующей 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 невероятно 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 паттернов проектирования. Хотя в литературе и повторяют заученную фразу: "Только паттерны и никаких антипаттернов", - 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 простотой и универсальностью инструментарий. Можно сказать, что налицо неканоническое применение антипатерна "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!!!

Практическое задание №3. Мой первый рисунок в ЛамПанель.

Основная проблема юного ассемблериста - это вывод на экран состояний регистров.

Чтобы это "провернуть", требуется средний уровень владения языком (не меньше).

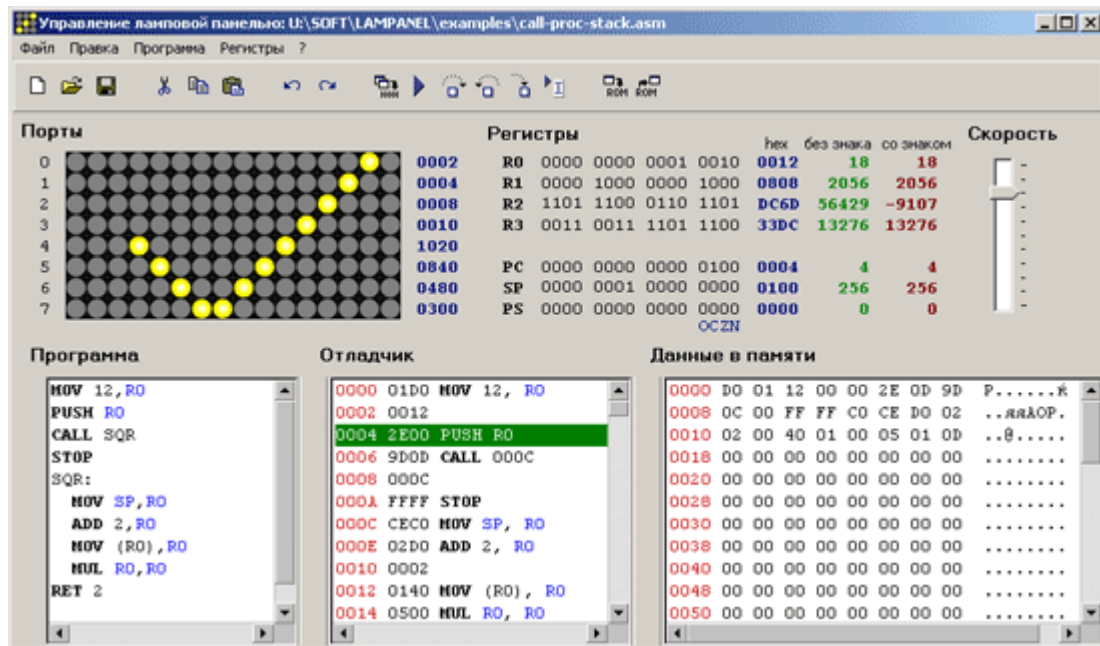
Это первая из преград, которая преодолевается нахождением адекватной среды разработки.

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 "программа посмотреть ПЗУ".

Подробнее о том, как сделать это...

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		DEC	HEK	BIN		0	1	2	3		
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0		0	0	0000		0	0	1	8	0	mov 0180,r0
1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0		1	1	0001		1	0	2	4	0	out r0,p0
2	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0		2	2	0010		2	0	5	A	0	mov 0240,r1
3	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0		3	3	0011		3	0	8	1	0	out r1,p1
4	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0		4	4	0100		4	1	F	F	8	mov 05A0,r2
5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0		5	5	0101		5	1	0	0	8	out r2,p2
6	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0		6	6	0110		6	1	0	0	8	mov 0810,r3
7	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0		7	7	0111		7	1	F	F	8	out r3,p3
																			8	8	1000						mov 1FF8,r0
																			9	9	1001						out r0,p4
																			10	A	1010						mov 1008,r1
																			11	B	1011						out r1,p5
																			12	C	1100						mov 1008,r2
																			13	D	1101						out r2,p6
																			14	E	1110						mov 1FF8,r3
																			15	F	1111						out r3,p7

01 2 34 56 78 9A BCDEF

0

1

2

3

4

5

6

7

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

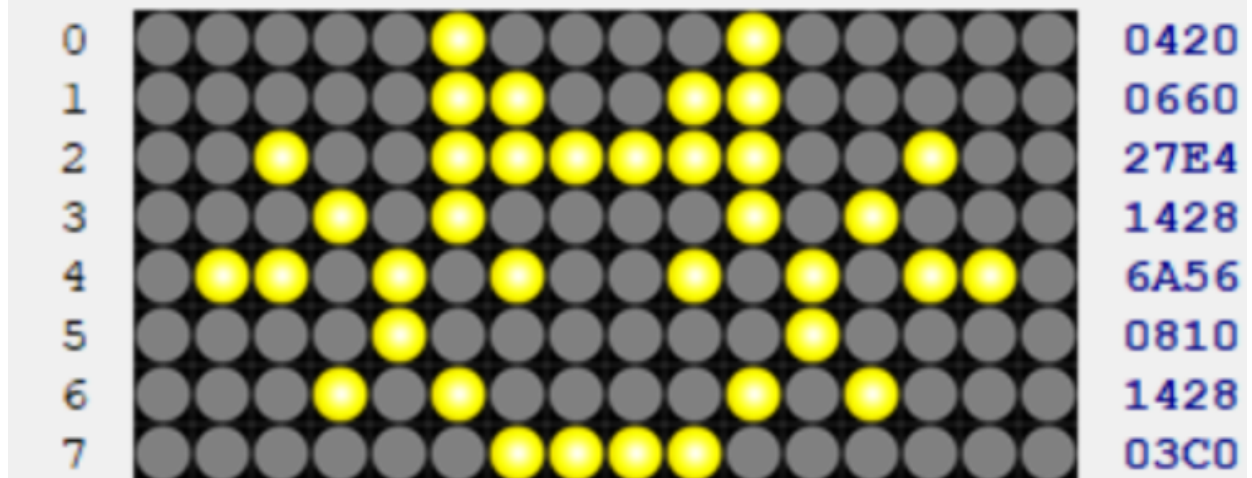
Творческая составляющая

Приветствуется реализация своей оригинальной картинки.

Кто-то из студентов подходит к вопросу творчески. Вот, например, котик, при циклическом сдвиге которого на отдельном кадре появляются "рожки". Это особенность картинки. Список

возможных картинок ограничен только фантазией.

Порты



Если возникают трудности, то задание может выполняться 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.

Распространенные ошибки.

In []:

- Попытка "закодировать" рисунок методом перебора 16-ричных кодов - превращает работу из одночасовой в 3 часовую.
- Попытка запуска программы из архива, не распаковывая. Как следствие ошибки при выполнении системных процедур нет возможности загрузки русскоязычной справки.
- Попытка использовать несуществующие регистры, их только 4: r0-r3. Вызывать r4 уже бесполезно. # Хорошая практика
- Группировка однотипно используемых команд за счет использования большего числа регистров. В теории 64 битные системы должны приводить к параллельному исполнению независимых ветвей кода. К тому же, работа с мелкими блоками команд затрудняет понимание кода.

```
mov 0180,r0  
mov 0240,r1  
mov 05A0,r2  
mov 0810,r3
```

```
out r0,p0  
out r1,p1  
out r2,p2  
out r3,p3
```

```
mov 1FF8,r0  
mov 1008,r1  
mov 1008,r2  
mov 1FF8,r3
```

```
out r0,p4
```

```
out r1,p5
```

```
out r2,p6
```

```
out r3,p7
```

</div>

- Группировка однотипно используемых команд за счет использования большего числа регистров. Работа с мелкими блоками команд затрудняет понимание.