Учим kernel

Издательство programoffice

#1. Основы языка

№1. Вывод

Вывод осуществляется оператором 01BX. Синтаксис:

01BX | размер сообщения | PT «сообщение»

№2. Ввод

Ввод осуществляется оператором 01GV Синтаксис:

01GV | размер сообщения | AN

№3. Первая программа

Напишем программу, выводящую приветствие на консоль:

01BX | bits 13 | PT «HELLO? WORLD!»

01BX | bits 16 | PT «HELLO, WORLD! (:»

№4. Функции

Функции создаются при помощи оператора 01FN. Синтаксис:

01FN | 0AF | NW | функция

№5. Оформление

Если используется одни и те же операторы в столбик с одной и той же целью, то при использовании следующего отличающегося оператора можно отделить их при помощи комментария (#).

№6. Память

Для того, чтобы работать с памятью, нужно отправлять скрипты в двоичном виде (0BB), шестнадцатеричном (0BH), в десятичном (0BD) или в восьмеричном (0BE). Также необходим адрес отправки (0CA) (он может быть в десятичной системе счисления от 0 до 79). Синтаксис:

0BD | 0CA:0 | 01FN | 0AF | NW | close\_files

№7. Сбор байт с консоли

Для того, чтобы заполнить поле в сообщении, где нужно прописать количество байт, можно использовать авто-оператор 0DN. Синтаксис:

01BX | bytes 0DN | PT «HELLO WORLD! (:»

В данном примере автоматически заполняется поле, где нужно вписать количество байт.

№8. Переменные

Для создания переменной нужно дать ей тип. Надо назубок знать простейшую таблицу типов переменных:

0BB (бинарный) 🡪 002X

0BH (шестнадцатеричный) 🡪 016X

0BD (десятичный) 🡪 010X

0BE (восьмеричный) 🡪 008X

После типа следует дать название, а потом значние. Синтаксис:

002X database | 64

№9. Операторы сложения, вычитания, умножения и деления.

Если нужно сравнить адреса, количество байтов или битов, значения переменных и т.д., используется знак /, если сложить - +, вычесть - -, умножить - \*. Синтаксис:

008X window | 83

002X CPU | 27

CPU |00P| window

01BX | bytes 0DN | PT «83 + 27 =» 0XV

Оператор 0XV используется для вставки значения.

Для того, чтобы сложить, вычесть, умножить, разделить, найти остаток переменную (№1) и число (№2), нужно использовать соответствующие знаки сложения/вычитания/деления/умножения/нахождения остатка от числа.

+ - / \* %

0AP 0AM 0AD 0BM 0AR

Синтаксис:

016X hello | 29 |0AP| 3 |

№10. Перенос значений переменных

Чтобы перенести значение переменной, нужно после переменной написать оператор 00T и название переменной, в какую нужно перенести значение. Синтаксис:

016X hello | 29

008X world | 40 | 00T | hello

В данном случае значение переменной world зануляется, а значение переменной hello увеличивается на 40.

Чтобы перенести определённое значение переменной, нужно написать тоже, что выше указанно, но заменить оператор 00T на оператор 00C, после этого написать значение, которое хотим перенести, а потом уже название переменной, в которую нужно перенести. Синтаксис:

016X hello | 29

008X world | 40 | 00C | 31 | hello

№11. Условие

Чтобы выполнять, определять и т.д. нужно использовать оператор 0CN. Синтаксис:

0CN | условие

Скрипт

0ST

№12. Комментарий

Комментарий пишется при помощи оператора #. Синтаксис:

#скрипт

№13. Циклы

Для создания цикла нужен оператор 0CL. Синтаксис:

0CL | скрипт

0EX

Оператор 0EX нужен для выхода из цикла.

№14. Обработка ошибок

Например, чтобы подключиться к сети нужен пароль. Здесь полезна **обработка ошибок**. Она делается при помощи оператора err. Синтаксис:

Скрипт

0ER | 0

Скрипт

№15. Выделение и освобождение памяти

Чтобы работать с памятью, необходимо вызубрить таблицу:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|гигабайт | мегабайт | килобайт | байт|

| | | | | | | | |

| \/ | \/ | \/ | \/ |

|001G | 001M | 001K | 001B|

|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_|

|гигабит | мегабит | килобит | бит |

| | | | | | | | |

| \/ | \/ | \/ | \/ |

| 000G | 000M | 000K | 000B|

|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_|

Для выделения памяти под текст, файл и т.д. можно использовать оператор 0L1T. Синтаксис:

0L1T | память

Например:

| 0BD | 0CA:21| 01LT | 24001G #выделяем память 24 гигабайта

Для освобождения памяти, нужен оператор 01FP. Синтаксис:

| 0BD | 0CA:31| 01FP | 63001G #выделяем память 63 гигабайта.

#2. Программирование на самом низком уровне

№16. Команды напрямую к процессору

Чтобы научиться программировать на самом низком уровне (напрямую писать команды процессору), нужно наизусть знать ещё одну таблицу (таблица в файле CPU Microsoft Excel.).

В общем, ничего сложного нет, также можно создавать функции, переменные и т.д., но нельзя использовать циклы.

№17. Функции на самом низком уровне

Для создания функций на самом низком уровне нужно использовать тот же самый синтаксис, как обычно, но вместо оператора fun нужно использовать оператор 0NF. Пример:

0BX | 0CA:43 | 0NF | 0AF | set\_temperature\_on\_the\_CPU

№18. Параметры

Если, например, нужно ввести параметры ввода/вывода, функций и др., нужно использовать оператор 0PA. Синтаксис:

0BX | 0CA:6 | 0NF | 0AF | SET\_THE\_TEMPERATURE\_ON\_THE\_CPU | 0PA | TEMPERATURE:3

№19. Строковые значения

Если нам нужно указать строковое значение переменной, можно использовать оператор 0VS. Синтаксис:

002X hello | 0CA:32 | 0VS | temperature\_CPU

Здесь у нас значение бинарной переменной hello, которая находится в ячейке ОЗУ с номером 32, равно значению температуры процессора.

№20. Регистры процессора

Нужно выучить таблицу регистров, чтобы дальше работать с регистрами:

XCD 🡪 2-битный регистр

URT 🡪 4-битный регистр

ABX 🡪 8-битный регистр

COH 🡪 16-битный регистр

DKH 🡪 32-битный регистр

RYB 🡪 64-битный регистр

PLS 🡪 128-битный регистр

IVD 🡪 256-битный регистр для кэша

То есть, можно использовать для временной или маловажной информации не память ОЗУ, а регистры. Стоит отметить, что можно задавать параметры регистрам, например, запрограммировать счётчик, очищающий конкретный регистр через определённое количество времени. Обозначает регистр оператор 0RY.

№21. Битовые операции

Если нам нужно, например, работать только с определённым количеством битов в ОЗУ или регистрами, нужен оператор 0TB. Синтаксис:

0RY | XCD | 0TB | 58 bytes | 0PA | back\_off\_32\_bytes

№22. Нибблы и работа с ними

Можно использовать различные выше упомянутые единицы измерения памяти, но нужно использовать оператор 0HB, чтобы работать с нибблами. Нибблы – это полубайты.

№23. Перенос из одной ячейки ОЗУ в другую

Для переноса информации из одной ячейки в другую, нужен оператор 0MC. Синтаксис:

0BX | 0CA:18 | 0TB | 351 bytes | 0PA | back\_off\_46\_bytes | 0MC | 0BX | 0CA:79

№24. Вставки ассемблерного кода

Для вставки ассемблерного код нужен оператор 0AS. Синтаксис:

0AS:

Ассемблерный\_код

0SS

№25. Буферная память

Можно передавать информацию через буферную память при помощи оператора 0FM. Синтаксис:

0BX | 0CA:18 | 0FM | 0BX | 0CA:79