УНИВЕРСИТЕТ ИТМО
КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Отчёт по лабораторной работе №2

Выполнил Ощепков Артём,
группа P3202
Преподаватель Жирков И.О.

Санкт-Петербург 2017

Задание

Используя макрос, создать статический LinkedList, инициализируемый на стадии препроцессинга. Использовать его в качестве словаря. Реализовать функцию find_word, предоставляющую данные по заданному ключу.

The macro will take two arguments:

- Dictionary key (inside quotes).
- Assembly label name. Keys can contain spaces and other characters, which are not allowed in label names.

Each entry should start with a pointer to the next entry, then hold a key as a null-terminated string. The content is then directly described by a programmer.

Текст программы

colon.inc:

```
; This macro accepts two args:
; 1) A key string. (you do not need to write \0 symbol)
; 2) A current element label
%macro colon 2

%ifndef prev        ; Automatically refering to the previous element.
%define prev 0
%endif

%2:                 ; The label
dq   prev           ;  A pointer to the next element
db %1, 0            ; Key string
%define prev %2
%endmacro
```

dict.asm:

```
; Args:
; 1, rdi) A pointer to a null terminated key string
; 2, rsi) A pointer to the last word in the dicntionary
find_word:
        push rdi            ; string_equals spoils these registers
        push rsi            ;
        lea  rsi, [rsi + 8] ; We now that the key locates after 8 bytes
                            ; List structure:
                            ; 1) Pointer to the next node (64 bit)
                            ; 2) Key-\0 string
                            ; 3) Date-\0 string

        call string_equals
        pop rsi
        pop rdi

        test rax, rax       ; 1 if equals, 0 otherwise.
        jnz  .found
        mov  rsi, [rsi]     ; Now rsi is the pointer to the next node
        test rsi, rsi       ; If the reference is null, we've gone over
                            ; All the nodes
        jnz find_word       ; Repeat if there is the next node
        xor eax, eax        ; Return 0 if the word have not been found
        ret
.found:
        mov rax, rsi        ; If success, return the record's addr
        ret
```

Выводы

Я познакомился с концепцией make-file'ов. Это — удобный и полезный инструмент. Научился разделять программы, написанные на языке ассемблера, на модули. Глубже проник в процесс сборки программы и мир препроцессинга. Думаю, если бы я сделал эту лабу, будучи в 10 классе, потратил бы тогда на подключение какой-то библиотеки гораздо меньше времени.