ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине "Языки Ассемблера"

Выполнил Ковалев Даниил

СКБ171, вариант 12

МИЭМ НИУ ВШЭ

Дата: / /2020

Баллы:

Задание: даны массивы A и B из 16 байтов (слов). Если элемент массива A больше соответствующего элемента массива B (числа знаковые), то обменять эти элементы местами. Сосчитать количество обменов. В массив C поместить адреса (смещения) этих элементов.

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <string>

using BYTE = std::int8_t;
using WORD = std::int16_t;

void lab_byte(BYTE* A, BYTE* B, std::uint64_t* C, std::uint8_t& swapped_count)
{
    asm(".intel_syntax          \n\t"
        "mov rdi, rax           \n\t" // адрес A в RDI
        "mov rsi, rbx           \n\t" // адрес B в RSI
        "mov r8,  rcx           \n\t" // адрес C в R8
        "xor dl,  dl            \n\t" // обнуляем счетчик обменов
        "mov ecx, 16            \n\t"
        "start_b:               \n\t" // начало цикла
        "mov al,  [rdi]         \n\t" // байт из массива A
        "cmp al,  [rsi]         \n\t" // сравниваем с байтом из массива B
        "jle after_swap_b       \n\t" // если A[i] <= B[i], ничего делать не надо
        "mov bl,  [rsi]         \n\t" // меняем местами A[i] и B[i]
        "mov [rdi],  bl         \n\t"
        "mov [rsi],  al         \n\t"
        "inc dl                 \n\t" // увеличиваем счетчик обменов
        "mov [r8], rdi          \n\t" // текущий адрес A в массив C
        "add r8, 8              \n\t"
        "mov [r8], rsi          \n\t" // текущий адрес A в массив C
        "add r8, 8              \n\t"
        "after_swap_b:          \n\t"
        "inc rdi                \n\t"
        "inc rsi                \n\t"
        "loop start_b"
        : "=d"(swapped_count)
        : "a"(A), "b"(B), "c"(C)
        );
}

void lab_word(WORD* A, WORD* B, std::uint64_t* C, std::uint8_t& swapped_count)
{
    asm(".intel_syntax          \n\t"
        "mov rdi, rax           \n\t" // адрес A в RDI
        "mov rsi, rbx           \n\t" // адрес B в RSI
        "mov r8,  rcx           \n\t" // адрес C в R8
        "xor dl,  dl            \n\t" // обнуляем счетчик обменов
        "mov ecx, 16            \n\t"
        "start_w:               \n\t" // начало цикла
        "mov ax,  [rdi]         \n\t" // слово из массива A
        "cmp ax,  [rsi]         \n\t" // сравниваем с словом из массива B
        "jle after_swap_w       \n\t" // если A[i] <= B[i], ничего делать не надо
        "mov bx,  [rsi]         \n\t" // меняем местами A[i] и B[i]
        "mov [rdi],  bx         \n\t"
        "mov [rsi],  ax         \n\t"
        "inc dl                 \n\t" // увеличиваем счетчик обменов
        "mov [r8], rdi          \n\t" // текущий адрес A в массив C
        "add r8, 8              \n\t"
        "mov [r8], rsi          \n\t" // текущий адрес A в массив C
        "add r8, 8              \n\t"
        "after_swap_w:          \n\t"
        "add rdi, 2             \n\t"
```

```
60          "add rsi, 2             \n\t"
61          "loop start_w"
62          : "=d"(swapped_count)
63          : "a"(A), "b"(B), "c"(C)
64          );
65  }
66
67  template<typename T>
68  void print_all(const std::vector<T>& A, const std::vector<T>& B)
69  {
70      std::cout << "A: {";
71      for (T elem : A)
72          std::cout << std::dec << std::setw(6) << static_cast<int>(elem) << ' ';
73      std::cout << "}\nB: {";
74      for (T elem : B)
75          std::cout << std::dec << std::setw(6) << static_cast<int>(elem) << ' ';
76      std::cout << "}\n";
77  }
78
79  template<typename T>
80  void print_all(const std::vector<T>& A, const std::vector<T>& B, const std::vector<std::
        uint64_t> C, std::uint8_t swapped_count)
81  {
82      print_all(A, B);
83      std::cout << "\nC: {\n";
84      for (std::uint8_t i = 0; i < swapped_count * 2; i += 2)
85          std::cout << "      " << std::dec << std::setw(2)
86                    << (C[i] - reinterpret_cast<std::uint64_t>(A.data())) / sizeof(T) << ' '
87                    << std::hex << C[i] << ' ' << C[i + 1] << '\n';
88      std::cout << "   }\nSwapped number: " << std::dec << static_cast<int>(swapped_count)
            << '\n';
89  }
90
91  int main()
92  {
93      std::cout << "i: {";
94      for (int i = 0; i < 16; ++i)
95          std::cout << std::setw(6) << i << ' ';
96      std::vector<std::uint64_t> C(32);
97      std::uint8_t swapped_count;
98      std::cout << "}\n";
99      {
100         std::cout << "\n" + std::string(55, ' ') + "BYTE\n";
101         std::vector<BYTE> A = {  105,   70,   90,   82, -114,   45,   59,   82,  -13,
                35, -123,  -70,   57,  -98,   81,  -32 };
102         std::vector<BYTE> B = {  -20,  -55,   32,  -10,   39,  -57,   27,   69,  -29,  -
                81,  123,   34, -118,  -96,   44, -110 };
103         std::cout << "Before:\n";
104         print_all(A, B);
105         lab_byte(A.data(), B.data(), C.data(), swapped_count);
106         std::cout << "\nAfter:\n";
107         print_all(A, B, C, swapped_count);
108     }
109     {
110         std::cout << "\n" + std::string(55, ' ') + "WORD\n";
111         std::vector<WORD> A = {   4996, -28469,  12215, -14279, -22535, -16077, -16993, -
                26883,    510,   4745,  -4753,   3704, -30365,  30061,  31930,  22819 };
112         std::vector<WORD> B = {  25757, -20781,  -4543,  31013, -32147,  -2348,  22367,
                27115,  30568,   6958,  25501,  23278, -25810,  22947,   4479,   1446 };
113         std::cout << "Before:\n";
114         print_all(A, B);
115         lab_word(A.data(), B.data(), C.data(), swapped_count);
116         std::cout << "\nAfter:\n";
117         print_all(A, B, C, swapped_count);
118     }
119 }
```

Результат работы программы:

```
i: {  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15 }

BYTE
Before:
A: {  105  70  90 -114  45  59  82 -13  35 -123 -70  57 -98  81 -32 }
B: {  -20 -55  32 -10  39 -57  27 -29 -81  123  34 -118 -96  44 -110 }

After:
A: {  -20 -55  32 -10 -114 -57  27  69 -29 -123 -70 -118 -98  44 -110 }
B: {  105  70  90  82  39  45  59  82 -13  123  34  57 -96  81 -32 }

C: {
    0 60200000010 60200000030
    1 60200000011 60200000031
    2 60200000012 60200000032
    3 60200000013 60200000033
    5 60200000015 60200000035
    6 60200000016 60200000036
    7 60200000017 60200000037
    8 60200000018 60200000038
    9 60200000019 60200000039
   12 6020000001c 6020000003c
   14 6020000001e 6020000003e
   15 6020000001f 6020000003f
}
Swapped number: 12

WORD
Before:
A: {  4996 -28469  12215 -14279 -22535 -16077 -16993 -26883   510  4745 -4753  3704 -30365  30061  31930  22819 }
B: {  25757 -20781  -4543  31013 -32147  -2348  22367  27115  30568  6958  25501  23278 -25810  22947  4479  1446 }

After:
A: {  4996 -28469  -4543 -14279 -32147 -16077 -16993 -26883   510  4745 -4753  3704 -30365  22947  4479  1446 }
B: {  25757 -20781  12215  31013 -22535  -2348  22367  27115  30568  6958  25501  23278 -25810  30061  31930  22819 }

C: {
    2 60300000014 60300000044
    4 60300000018 60300000048
   13 6030000002a 6030000005a
   14 6030000002c 6030000005c
   15 6030000002e 6030000005e
}
Swapped number: 5
```