Лабораторная работа 1

Калашников Михаил, Б03-205

Вставки и переходы

1. Воспользуемся ассемблерной вставкой и сгенерируем листинги.

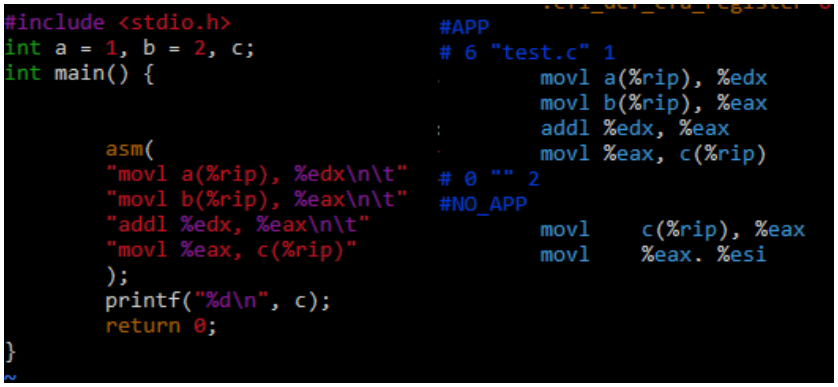

Рис. 1: К пункту 1

2-4. Вставим в код условный опретор и получим в листинге метку, команду перехода и оператор сравнения. Также воссоздадим условный оператор с помощью листинга, пользуясь метками куст и лампа (не спрашивай).

5. Создадим статический одномерный массив. В объявлении выделится 4 * 10 (размер массива) байт. Обращение происходит с помощью команды lea. Инициализация двумерных массивов аналогична.

6. Очевидно, безусловные команды перехода (jmp) осуществляют безусловный переход, а условные (e.g., je) – при удовлетворении некоего условия (предыдущей команды cmp). Проверить их работу мы сможем в следующем пункте.

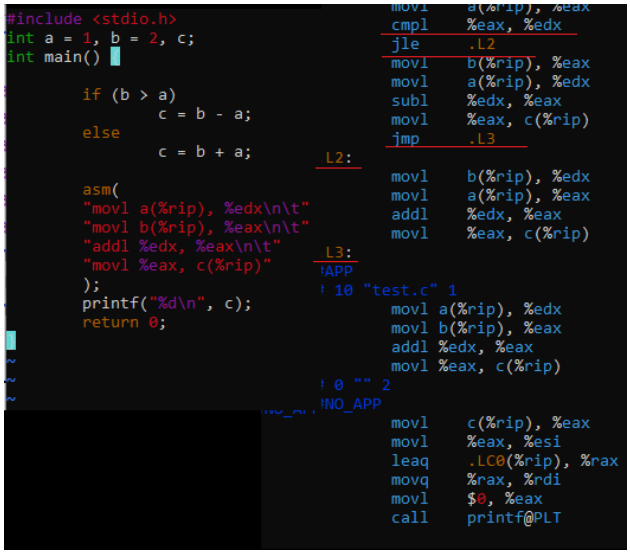

Рис. 2: К пунктам 2-4

7. Условный оператор уже был организован в предыдущих пунктах. Реализуем цикл for. Он должен десять раз инкрементировать переменную 1. Он действительно работает как и представлено на картинке. Остальные виды циклов реализуются очень похожим простым образом.

8. Команда loop сильно упрощает реализацию цикла. Ниже представлен тот же цикл for из прошлого пункта, но с использованием оператора loop.

9. На картинке представлена реализация поиска максимума из двух чисел.

10. На картинке представлена реализация поиска суммы элементов массива.

11. Теперь реализуем сортировку пузырьком. Вставка приложена к отчету в отдельном файле, графики представлены на картинке. Дефолтная сортировка пузырьком с флагами -O1 и -O2 оказывается быстрее сортировки, написанной целиком на ассемблере.

```c
#include <stdio.h>
int a = 1, b = 2, c;
int main() {

        asm(
        "movl a(%rip), %edx\n\t"
        "movl b(%rip), %eax\n\t"
        "cmpl %eax, %edx\n\t"
        "jle  bush\n\t"
        "subl %edx, %eax\n\t"
        "movl %eax, c(%rip)\n\t"
        "jmp  lamp\n"
        "bush:\n\t"
        "addl %edx, %eax\n\t"
        "movl %eax, c(%rip)\n"
        "lamp:\n\t"
        );

        printf("%d\n", c);
        return 0;
}
```

Рис. 3: К пунктам 2-4

```c
#include <stdio.h>

int a = 6;
int c[10] = {0};

int main() {

        printf("%d\n", c[a]);

        return 0;
}
```

```
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    a(%rip), %eax
        cltq
        leaq    0(,%rax,4), %rdx
        leaq    c(%rip), %rax
        movl    (%rdx,%rax), %eax
```

Рис. 4: К пункту 5

```c
#include <stdio.h>

int a = 1;

int main() {

        printf("%d\n", a);

        return 0;
```

```
        movl    $0, %edx
body:
        addl    $1, %eax
        addl    $1, %edx
        cmpl    $10, %edx
        jl      body
```

```
layth
a=11
```

Рис. 5: К пункту 7

```
        movl    $10, %ecx
body:
        addl    $1, %eax
        loop    body
```

Рис. 6: К пункту 8

```
#include <stdio.h>

int a = 10, b = 2, c;

int main() {

        asm(
                "movl a(%rip), %edx\n\t"
                "movl b(%rip), %eax\n\t"
                "cmpl %edx, %eax\n\t"
                "jl   mark\n\t"
                "movl %eax, c(%rip)\n\t"
                "jmp  continue\n"
                "mark:\n\t"
                "movl %edx, c(%rip)\n"
                "continue:\n\t"
        );

        printf("Max is equal to %d\n", c);


        return 0;
}
```

Рис. 7: К пункту 9

```
int a[] = { 1, 3, 5, 4 };
int n = 4, c = 0;

int main() {

        asm(
                "movq n(%rip), %rcx\n"
                "cycle:\n\t"
                "movq %rcx, %rdx\n\t"
                "subq $1, %rdx\n\t"
                "leaq a(%rip), %rax\n\t"
                "movq (%rax, %rdx, 4), %rbx\n\t"
                "addq %rbx, c(%rip)\n\t"
                "loop cycle"
        );
```

Рис. 8: К пункту 10

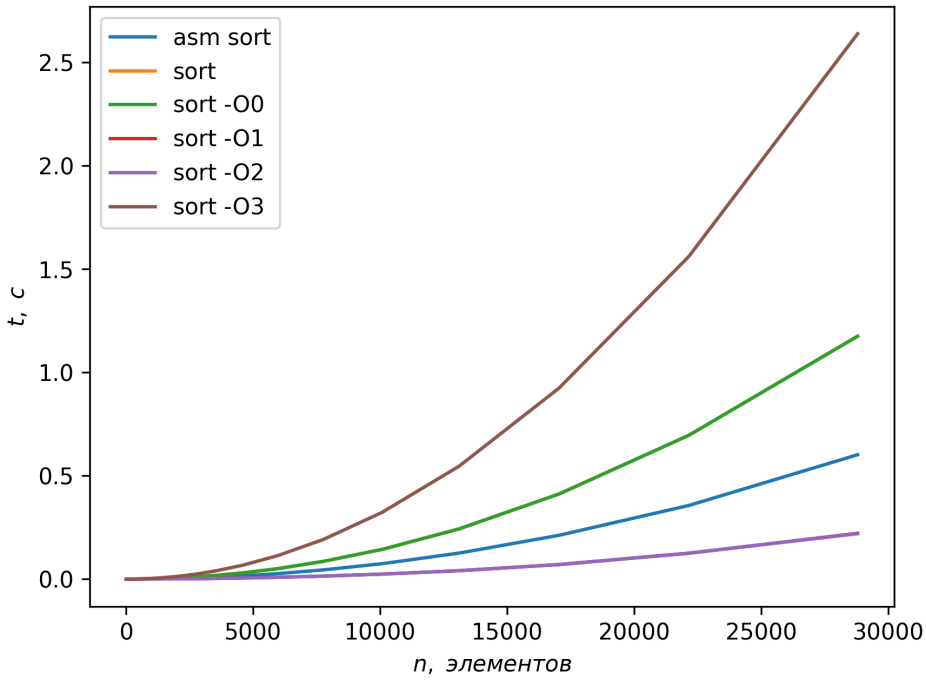

Рис. 9: К пункту 11