Лабораторная работа №1

Игнатьев П. А.
Б03-205

1. Программа с ассемблерной вставкой

```c
#include <stdio.h>
int x = 3, y = 2, z;
int main(){

    asm ("movl x(%rip), %eax\n"
        "movl y(%rip), %edx\n"
        "imull %eax, %edx\n"
        "movl %edx, z(%rip)\n");
    printf("%d",z);
return 0;
}
```

В ассемблерной вставке выполняется:
1) перенос значений переменных в нужные регистры
2) Происходит умножение двух регистров
3) Присвоение результата умножения переменной z

В ассемблерном листинге код выглядит следующим образом:

```
#APP
# 5 "asm_attempt.c" 1
    movl x(%rip), %eax
movl y(%rip), %edx
imull %eax, %edx
movl %edx, z(%rip)
```

#APP – (imho) означает ассемблерную вставку из самого кода

Далее после этого идёт #NO_APP и остальной код, не написанный “программистом”.

ASSembler’ные метки и переход к ним (наиболее встречаемые, если ты не “жёсткий прогер”):
Для начала, что такое **cmp**?
cmp – сравнение (O_0O)

jmp – переходит на метку вне зависимости от условий
je - Jump if Equal
jne - Jump if Not Equal
jg - Jump if Greater or Equal
jz - Jump if Zero
jnz - Jump if Not Zero

Ассемблерная вставка с метками:

```
#include <stdio.h>
int x = 6, y = 5, z;
int main(){

    asm (
        "movl x(%rip), %eax\n"
        "movl y(%rip), %edx\n"
        "cmp %eax, %edx\n"
        "jg .L2\n"
        "movl x(%rip), %edx\n"
        "movl y(%rip), %eax\n"
        "subl   %edx, %eax\n"
        "movl   %eax, z(%rip)\n"
        "jmp .L3\n"
        ".L2:\n"
        "   movl    x(%rip), %edx\n"
        "   movl    y(%rip), %eax\n"
        "   imull %eax, %edx\n"
        "   movl    %edx, z(%rip)\n"
        ".L3:\n"
        "   nop");
    printf("%d",z);
return 0;
}
```

РАБОТАЕТ!!!
Если x < y: y – x;
иначе : y * x;

cmp – проверка
потом если %eax > %edx переходим на .L2
если нет, то идём дальше пока не встречаем .L3
в .L3 nop – значит нет оператора, т.е. за этим последует завершение программы

КРЧ массивы:

```
#include <stdio.h>

// int x = 2, y = 4, z;
int a[3] = {1, 2, 3};

int main(){
    return 0;
}
```

В ассемблерном листинге (очевидно там всё):

```
    .globl  a
    .data
    .align 8
    .type   a, @object
    .size   a, 12
a:
    .long   1
    .long   2
    .long   3
    .text
```

Вывод элемента массива:

```c
#include <stdio.h>

// int x = 2, y = 4, z;
int a[3] = {1, 2, 3};

int main(){
    printf("%d", a[1]);
    return 0;
}
```

В листинге:

```
movl    4+a(%rip), %eax
```

Означает начало a (a.begin()) плюс 4 байта == (a.begin() + 4) or (a[1])
Вроде легко.

Команды перехода условные и безусловные:

Писалось об этом выше (jmp, ig, je and etc.)