Московский физико-технический институт

Ассемблер, лабораторная работа №0

# Арифметика и основы работы с листингами

выполнил студент группы Б03-205

Овсянников Андрей

Долгопрудный, 2023 г.

1. Поставлена wsl.

2.

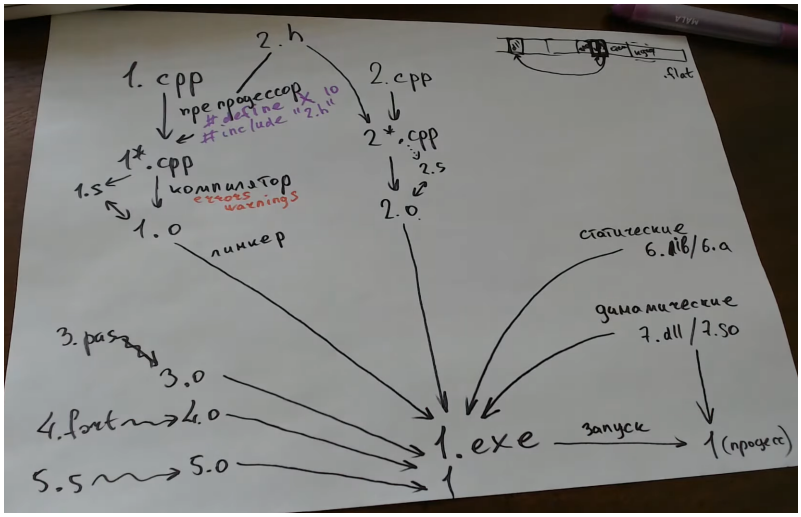



Рис. 1: Схема сборки

3. Сгенерируем ассемблерные листинги программы, просто выводящей хелло ворлд на g++ и gcc. Видим различие в строке вывода.

```
        .file   "1.c"
        .text
        .section        .rodata
.LC0:
        .string "Hello, World!"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    .LC0(%rip), %rax
        movq    %rax, %rdi
        movl    $0, %eax
        call    printf@PLT
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
```

gcc

```
        .text
        .globl  main
        .type   main, @function
ain:
LFB1731:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    .LC0(%rip), %rax
        movq    %rax, %rsi
        leaq    _ZSt4cout(%rip), %rax
        movq    %rax, %rdi
        call    _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@PLT
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
LFE1731:
```

g++

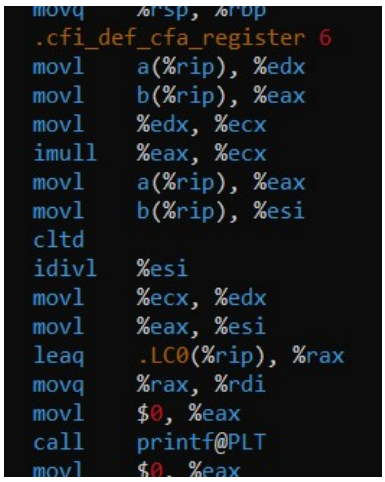
4. Напишем так:

```
#include <stdio.h>

int a = 6, b = 3;

int main() {
        printf("/, *: %d, %d", a / b, a * b);
        return 0;
}
```

Сгенерируем листинг:




Вывод такой программы будет:

```
>> /, *: 2, 18
```

Теперь изменим ассемблерный файл так:

```
        movl    a(%rip), %edx
        movl    b(%rip), %eax
.inj    movl    $2, %eax
        movl    %edx, %ecx
        imull   %eax, %ecx
        movl    a(%rip), %eax
        movl    b(%rip), %esi
        cltd
        idivl   %esi
        movl    %ecx, %edx
        movl    %eax, %esi
        leaq    .LC0(%rip), %rax
        movq    %rax, %rdi
        movl    $0, %eax
        call    printf@PLT
```

Результат станет:

```
>> /, *: 2, 12
```

Вполне ожидаемо.

5. Сравнили листинги программы, просто выводящей привет мир, в пунктах до этого.
6. - '-' - div;
   - '+' - add;
   - присвоение - mov;
7. Глобальные переменные определяются в начале ассемблерного кода так:

```
        .globl  a
        .data
        .align 4
        .type   a, @object
        .size   a, 4
a:
        .long   6
        .globl  b
        .align 4
        .type   b, @object
```

```
        .size   b, 4
b:
        .long   3
```

8.

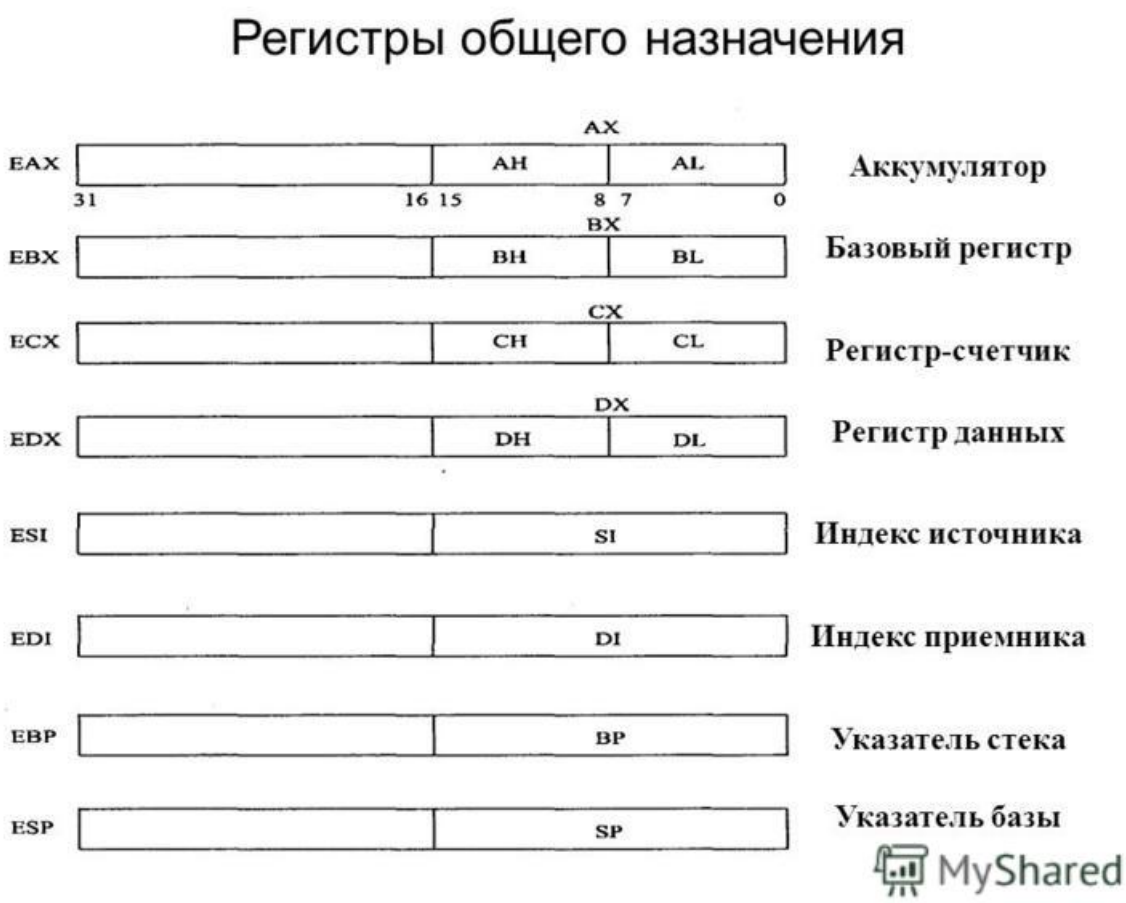



Рис. 2: Регистры общего назначения

9. Приведем информацию с очень удобной документации(я это проверил в коде):

**Команды DIV и IDIV**

| | |
|---|---|
| Синтаксис: | DIV op1<br>IDIV op1 |
| Операнды: | op1 - r/m8, r/m16, r/m32 |
| Назначение: | Деление беззнакового числа (DIV) и числа со знаком (IDIV) |
| Процессор: | 8086+ |
| Флаги: | Флаги OF, SF, ZF, AF, PF и CF не определены |
| Комментарий: | Если размерность операнда составляет 8 бит, то команда производит целочисленное деление содержимого регистра AX на значение операнда и помещает результат деления в регистр AL, а остаток - в регистр AH.<br>Если операнд - 16-битное слово, команда производит целочисленное деление содержимого пары регистров DX:AX на значение операнда и помещает результат деления в регистр AX, а остаток - в регистр DX.<br>Если операнд - двойное слово, команда производит целочисленное деление содержимого пары регистров EDX:EAX на значение операнда и помещает результат деления в регистр EAX, а остаток - в регистр EDX. |
| Ограничения: | Если результат деления не помещается в регистр-приемник (такое может произойти при делении больших чисел на маленькие), происходит вызов прерывания 0. То же самое происходит и при попытке поделить число на ноль. |
| Примеры: | mov ax,2343h<br>mov bl,355<br>div bl |

Рис. 3: div_idiv

**Команда IMUL (форма 1)**

| | |
|---|---|
| Синтаксис: | IMUL op1 |
| Операнды: | op1 - r/m8, r/m16, r/m32 |
| Назначение: | Умножение со знаком |
| Процессор: | 8086+ |
| Флаги: | Флаги CF и OF сбрасываются в 0, если размер результата умножения получился меньше или равным размеру заданного операнда. В противном случае эти флаги становятся равными 1.<br>Значения флагов SF, ZF, AF и PF не определены. |
| Комментарий: | Если размерность операнда составляет 8 бит, то команда IMUL производит умножение содержимого регистра AL на значение операнда и помещает результат в регистр AX.<br>Если операнд - 16-битное слово, команда IMUL производит умножение содержимого регистра AX на значение операнда и помещает результат в пару регистров DX:AX.<br>Если операнд - двойное слово, команда IMUL производит умножение содержимого регистра EAX на значение операнда и помещает результат в пару регистров EDX:EAX. |
| Ограничения: | Нет |
| Примеры: | mov al,5<br>mov bl,3<br>imul bl ;AX=000Fh |

Рис. 4: imul 1

**Команда IMUL (форма 2)**

| | |
|---|---|
| Синтаксис: | `IMUL op1,op2,op3` |
| Операнды: | op1 - r16, r32<br>op2 - r/m16, r/m32<br>op3 - i8, i16, i32 |
| Назначение: | Умножение со знаком |
| Процессор: | 80186+ |
| Флаги: | Флаги CF и OF устанавливается в 1, если результат умножения не вместился в регистр назначения (первый операнд). В противном случае эти флаги сбрасываются в 0.<br>Значения флагов SF, ZF, AF и PF не определены. |
| Комментарий: | Команда выполняет умножение второго операнда на третий операнд и помещает результат в первый операнд. |
| Ограничения: | Разрядность операндов должна совпадать. Исключением является использование в качестве второго операнда непосредственного 8-битного значения. |
| Примеры: | `imul dx,cx,-45`<br>`imul ax,[bx],3` |

Рис. 5: imul 2

**Команда IMUL (форма 3)**

| | |
|---|---|
| Синтаксис: | `IMUL op1,op2` |
| Операнды: | op1 - r16, r32<br>op2 - r/m16, r/m32, i8, i16, i32 |
| Назначение: | Умножение со знаком |
| Процессор: | 80186+ |
| Флаги: | Флаги CF и OF устанавливается в 1, если результат умножения не вместился в регистр назначения (первый операнд). В противном случае эти флаги сбрасываются в 0.<br>Значения флагов SF, ZF, AF и PF не определены. |
| Комментарий: | Команда выполняет умножение первого операнда на второй операнд и помещает результат в первый операнд. |
| Ограничения: | Разрядность операндов должна совпадать. Исключением является использование в качестве второго операнда непосредственного 8-битного значения. |
| Примеры: | `imul dx,cx`<br>`imul ebx,334` |

Рис. 6: imul 3