?????????? ? ????????

При стандартной компиляции проекта создается полноценный ELF файл, происходит выравнивание по границам страниц памяти. При использовании указателя global main подключается стартовый код стандартной библиотеки C.

Точки входа.

Точка входа определяется значением global <что-то> Варианты точек входа.

Тип программы	Рекомендуемая точка входа	Компиляция
Самостоятельная Linux	_start	ld
C использованием libc	main	gcc
GUI Windows	WinMain	Visual Studio
DLL Windows	DllMain	Visual Studio
Ядро ОС	kmain	специальный линкер
Пользовательская	любое имя	ld -e имя

Варианты компиляции

Команды консоли

```
nasm -f elf64 hello.asm -o hello.o
ld -o hello hello.o
```

В случае использования точки входа _start

make файл

```
hello: hello.o
	gcc -o hello hello.o -no-pie
hello.o: hello.asm
	nasm -f elf64 -g -F dwarf hello.asm -l hello.lst
```

Компилирование происходит командой make,

Процедура поиска библиотек

Директивой extern printf говорится компилятору: "я знаю где эта функция, делай все остальное".

```
nm -D /lib/x86_64-linux-gnu/libc.so.6 # просмотр списка функций в библиотеке
# Какие библиотеки использует программа
ldd program

# Посмотреть неразрешенные символы в объектном файле
nm -u program.o

# Посмотреть символы в исполняемом файле
nm program | grep printf
```

Объединение нескольких файлов.

Вариант 1 - директива include. Она фактически вставляет текст одного файла в другой файл. Например есть файл sum.asm который мы будем включать в файл hello.asm.

```
section .text
; Функция возвращает сумму чисел
; Принимает два параметра:
; rdi - первое число
; rsi - второе число
; Результат функции возвращается через регистр rax
sum:
    mov rax, rdi      ; результат в rax
    add rax, rsi
    ret
```

Тогда hello.asm

```
global _start

section .text
%INCLUDE "sum.asm"
_start:
    mov rdi, 33
    mov rsi, 44

    call sum
    mov rdi, rax         ; для проверки результата помещаем сумму из RAX в RDI
    mov rax, 60
    syscall
```

Вариант 2. Раздельная компиляция. В этом случае доступные извне метки (функции, данные) объявляются с помощью директивы global. Файл sum.asm:

```
global sum  ; делаем функцию sum доступной извне

section .text
; Функция возвращает сумму чисел
; Принимает два параметра:
; rdi - первое число
; rsi - второе число
; Результат функции возвращается через регистр rax
sum:
    mov rax, rdi      ; результат в rax
    add rax, rsi
    ret
```

Файл hello.asm

```
global _start

extern sum      ; функция sum расположена где-то во вне

section .text
_start:
    mov rdi, 33
    mov rsi, 44

    call sum
    mov rdi, rax         ; для проверки результата помещаем сумму из RAX в RDI
    mov rax, 60
    syscall
```

Сначала скомпилируем файл sum.asm:

```
nasm -f elf64 sum.asm -o sum.o
```

Затем скомпилируем файл hello.asm:

```
nasm -f elf64 hello.asm -o hello.o
```

В итоге у нас получится два разных объектных файла - sum.o и hello.o. Скомпонуем их в один исполняемый файл:

```
ld hello.o sum.o -o hello
```

Revision #5
Created 6 November 2025 13:52:01 by Admin
Updated 10 November 2025 17:09:51 by Admin