2. Этапы получения исполняемого файла из исходного кода. Опции компилятора и компоновщика. (ОК)
План ответа:
 препроцессирование;
 компиляция;
 ассемблирование;
 компоновка;
 «стандартная» (POSIX) строка запуска компилятора;
 ключи компилятора и компоновщика: -std, -Wall, -Werror, -pedantic, -c, -o, -E, -S

Препроцессирование
Препроцессирование – замена всех директив препроцессора на команды языка Си. Также он выполняет
Непосредственное объявление всех функций в теле вызывающего модуля;
вырезание комментариев;
текстовые замены (директива define);
включение файлов (директива include).
Выполнение условной компиляции и прагматических выражений
Файл, получаемый в результате работы препроцессора, называется единицей трансляции.
cpp –o hello.i hello.c

Компиляция – преобразование кода на языке Си в код на языке ассемблера.
Компилятор выполняет трансляцию программы, написанной на Си, на язык ассемблера.
В процессе преобразования проверяется наличие синтаксических и семантических ошибок в коде.
Язык ассемблера - система обозначений, используемая для представления в удобочитаемой форме программ, записанных в машинном коде.
c99 –S –masm=intel hello.i

Ассемблирование в объектный файл
Ассемблер выполняет перевод программы на языке ассемблера в исполнимый машинный код.
В результате работы ассемблера получается объектный файл: блоки машинного кода и данных, с неопределенными адресами ссылок на данные и процедуры в других объектных модулях, а также список своих процедур и данных.
as –o hello.o hello.s

Компоновка
Компоновщик принимает на вход один или несколько объектных файлов и собирает по ним исполнимый файл.
Компоновщик может извлекать объектные файлы из специальных коллекций, называемых библиотеками.
ld –o hello.exe hello.o ...библиотеки

// 1. Препроцессор gcc –E main.c > main.i
// 2. Трансляция на язык ассемблера gcc –S main.i
// 3. Ассемблирование gcc –c main.s
// 4. Компоновка gcc –o main.exe main.o
// Вместо 1-3 можно написать gcc –c main.c
// Вместо 1-4 можно написать gcc –o main.exe main.c

gcc [опции] [выходной_файл] файл_1 [файл_2]

-std – определяет стандарт языка Си, согласно которому требуется компилировать код. По умолчанию стоит стандарт c89.
-Wall – заставляет компилятор учитывать все ошибки и предупреждения, в том числе не влияющие на работоспособность программы, такие как:
Неинициализированные переменные
Неиспользуемые переменные (инициализируются, но не используются)
Слежение за указателями (проверка ключевого слова const)
-Werror – все предупреждения считать ошибками, компиляцию завершить по сценарию компиляции кода с ошибками.
-pedantic – усиленная проверка на грамотность кода, выдача замечаний о возможных предупреждениях и ошибках выполнения программы.
-c (--compile) – только компиляция, без сборки приложения
-o filename – сохранение результатов в файл с именем filename.

-ggdb – подготовка к сканированию с помощью Dr.Memory.
-g[level] (--debug)
-O компилятор пробует уменьшить размер кода и время исполнения.
        `-O0' Не оптимизировать.(оптимизирует время компиляции, используется по у
молчанию)
        `-O1 or -O' оптимизация размера кода и времени исполнения
        `-O2' включает все необязательные оптимизации кроме раскрутки циклов и по
дстановки функций.
        `-O3' включает все оптимизации, определяемые `-O2', а также включает опци
ю `inline-functions'.

gcc –std=c99 –Wall –Werror -pedantic –o main.exe main.c