




Manual de Prácticas Microprocesadores

División: Ingeniería Eléctrica

Departamento: Ingeniería Electrónica

Conocimiento de la plataforma para el desarrollo de las prácticas

N.º de práctica: 02

Nombre completo del alumno		Firma
Maria Yesica Pérez Navarro		
N.º de brigada:	Fecha de elaboración 29 Agosto 2017	Grupo: 3



1. Objetivos de aprendizaje.

- El alumno conocerá el ambiente de desarrollo integrado Code Composer Studio, empleado para editar código fuente, depurarlo y programar la tarjeta de desarrollo.
- Conocerá el proceso que se lleva a cabo para programar un microprocesador.

2. Material y equipo.

- Tener instalado el IDE Code Composer Studio de Texas Instruments.
- Tarjeta de desarrollo.
- Documento *spnu118o ARM Assembly Language Tools v15.12.0.LTS User's Guide*

3. Actividad de investigación

1. ¿Qué es un Integrated Development Environment (IDE)?

IDE del acrónimo de Integrated Development Environment (Entorno de desarrollo integrado en español), es un software que contiene un “paquete” de lo que necesitamos, por lo cual se le denomina entorno, dentro de sus características esta en contener un editor de código, un compilador, un depurador, estos pueden trabajar con un lenguaje de programación específico o con más de uno.

Los IDEs nos ayudan a poder trabajar de una forma mucho más amigable y fácil, ya que por ejemplo al escribir todo nuestro código en un editor de texto y este contiene un error no sabremos en donde está dicho error.

Se dice que un buen IDE debe tener:

- Resaltado de sintaxis (Syntax highlighting).
- Completado de código (code intelligence).
- Búsqueda de código.
- Resaltado de errores y advertencias.
- Refactorización y generación de código.
- Depuración de código (Debugging).

(Ormeño, 2012, <http://ormeno-nicolas.blogspot.mx/2012/02/que-es-un-ide.html>)

2. ¿Qué es lenguaje máquina?

El lenguaje máquina es el único lenguaje que puede ejecutar una computadora, es específico en cada arquitectura, es un código que es interpretado directamente por el microprocesador, está compuesto por un conjunto de instrucciones ejecutadas en secuencia que representan acciones que la máquina podrá tomar.

El lenguaje máquina es el único que entiende directamente la computadora, utiliza el alfabeto binario que consta de los dos únicos símbolos 0 y 1, denominados bits; físicamente, se materializan con tensiones comprendidas entre 0 y 4.0 [V] y entre 4 y 5 [V], respectivamente. Para representar datos que contengan una información se utilizan una serie de unos y ceros cuyo conjunto indica dicha información.

Todo código fuente en última instancia debe llevarse a un lenguaje máquina mediante el proceso de compilación o interpretación para que la computadora pueda ejecutarlo.

(Suárez, 2014, <https://arquitecturadelcomputadorsemestre6.blogspot.mx/>)

3. ¿Qué es un editor, un compilador, un ensamblador, un ligador (linker) y un depurador?

Los compiladores son programas que *traducen* un fichero de código fuente de cualquier lenguaje al lenguaje ensamblador y lo llama, cuando sea necesario. Los más importantes son GCC (GNU Compiler Colector) para C, G++ para C++, G77 para Fortran 77 y Microsoft Visual C++, entre otros.

Los ensambladores son aquellos programas que se encargan de desestructurar el código en lenguaje ensamblador y traducirlo a lenguaje binario. Los archivos en lenguaje binario se enlazan posteriormente en un único fichero, el ejecutable. Los más importantes son *tas*, *gas*, *nasm*.

Los enlazadores son los programas que enlazan varios ficheros objeto en lenguaje binario para crear un único fichero, el ejecutable del programa. El más importante es *ld*.

Los depuradores como su nombre lo indica, sirven para corregir los errores o fallas de la programación (bugs). Se encargan de ejecutar, paso a paso un programa, alertando sobre los errores presentados y los valores de las variables, entre otros. Son particularmente útiles cuando el programa parece estar bien, pero no se obtiene el resultado esperado (se cuelga, da resultados erróneos...). El más importante es GDB. Actualmente casi todas las IDEs incluyen uno, o deberían.

Los editores de texto son tan importantes como un compilador pues es el editor de la programación; actualmente incluyen funciones específicamente dedicadas a la programación, como resaltado de sintaxis, y autoindentación, entre otras. Grandes editores de texto son GNU Emacs, Vim, Scite, Notepad++. (Wikilibros contributors, 2017, https://es.wikibooks.org/wiki/Fundamentos_de_programaci%C3%B3n/Herramientas_de_desarrollo)

4. ¿Cuáles son las características de una memoria RAM?

La memoria RAM es una memoria de acceso aleatorio o directo; es decir, el tiempo de acceso a una celda de la memoria no depende de la ubicación física de la misma (se tarda el mismo tiempo en acceder a cualquier celda dentro de la memoria).

La memoria RAM es temporal o memoria de lectura y escritura.

La memoria RAM está destinada a contener los programas cambiantes del usuario y los datos que se vayan necesitando durante la ejecución y reutilizable.

La memoria RAM es volátil, es decir si se pierde la alimentación eléctrica, la información presente en la memoria también se pierde.

5. ¿Cuáles son las características de una memoria ROM?

La característica principal de ser una memoria ROM es que es de sólo lectura.

La Memoria ROM se utiliza para la gestión del proceso de arranque, el chequeo inicial del sistema, carga del sistema operativo y diversas rutinas de control de dispositivos de entrada/salida que suelen ser las tareas encargadas a los programas grabados en la Memoria ROM. Estos programas (utilidades) forman la llamada Bios del Sistema.

La memoria ROM no es volátil.

6. ¿Qué es el mapa de memoria de un Microprocesador?

Un mapa de memoria (del inglés *memory map*) es una estructura de datos (tablas) que indica cómo está distribuida la memoria. Contiene información sobre el tamaño total de memoria y las relaciones que existen entre direcciones lógicas y físicas, además de poder proveer otros detalles específicos sobre la arquitectura del computador y es capaz de direccionar un microprocesador. Los mapas de memoria suelen ser creados usualmente por el firmware para dar información al núcleo del sistema operativo sobre cómo está distribuida la memoria.

(colaboradores de Wikipedia, 2017, https://es.wikipedia.org/w/index.php?title=Mapa_de_memoria&oldid=97746929)

7. Leer los apartados siguientes.

Capítulo 1.

- *1.1 Software Development Tools Overview*
- *1.2 Tools Descriptions*

Capítulo 2.

- *2.2 Executable Object Files*
- *2.3 Introduction to Sections*
 - 2.3.1 Special Section Name*
- *2.4 How the Assembler Handles Sections*
 - 2.4.1 Uninitialized Sections*
 - 2.4.2 Initialized Sections*
 - 2.4.7 Using Sections Directives*

Explique:

¿Qué es un archivo Objeto?

Es el archivo que genera el ensamblador al *traducir* los archivos fuente del *assembly language*.

¿Cuáles son las tres secciones principales que componen a un archivo objeto?

La sección *.text* que contiene código ejecutable, la sección *.data* que usualmente contiene información inicializada y la sección *.bss* que usualmente reserva espacio para las variables no inicializadas.

¿Qué es un símbolo en un archivo objeto? (*spnu1180* sección 2.4.1, 2.6)

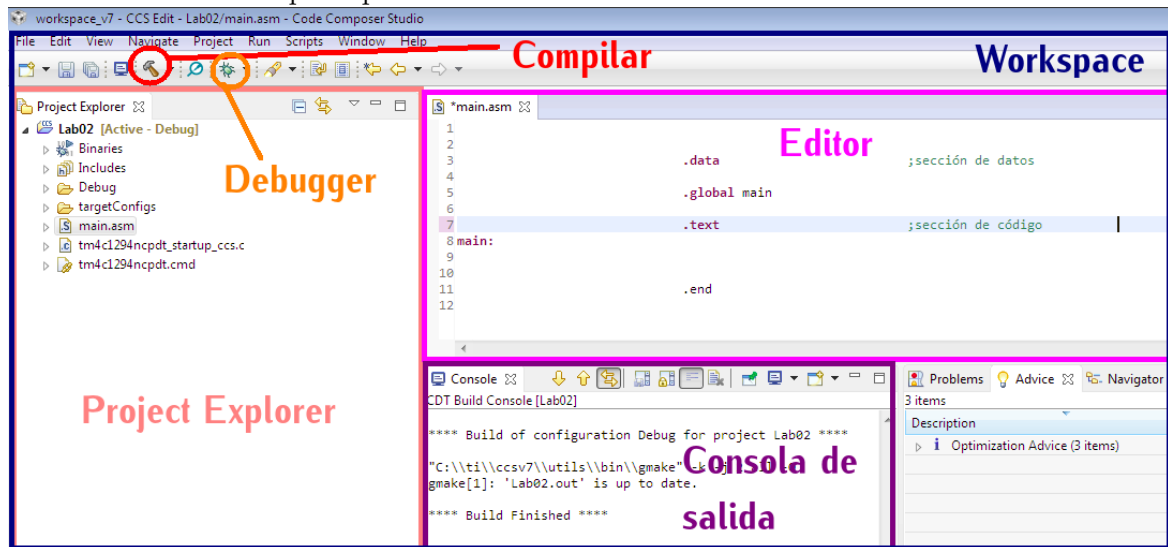
El símbolo es una representación/etiqueta de que ahí va una variable (a la cuál estás reservando espacio).

¿Qué es una directiva (de preprocesamiento)?

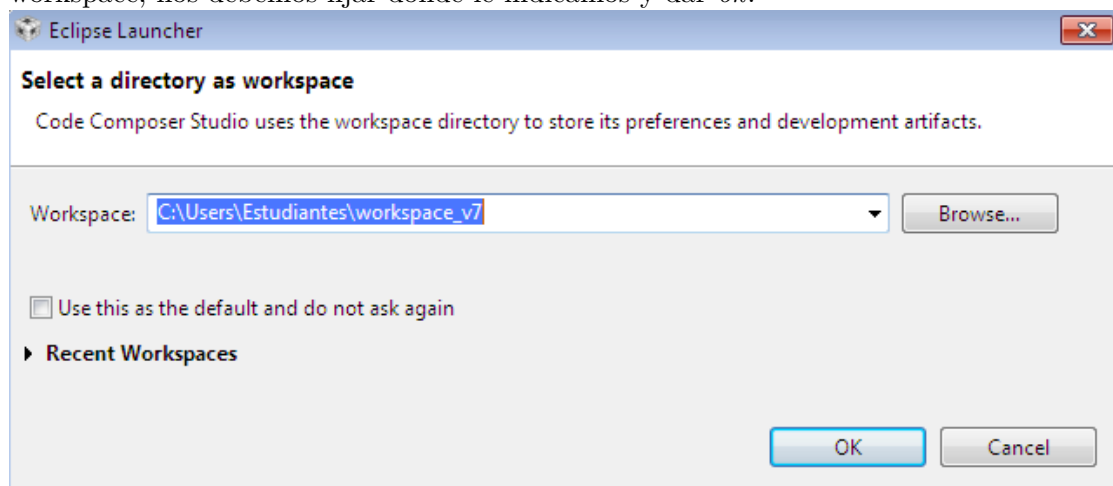
Son instrucciones para el compilador/interprete y se ejecutan antes del código, no son del lenguaje de programación.

4. Desarrollo

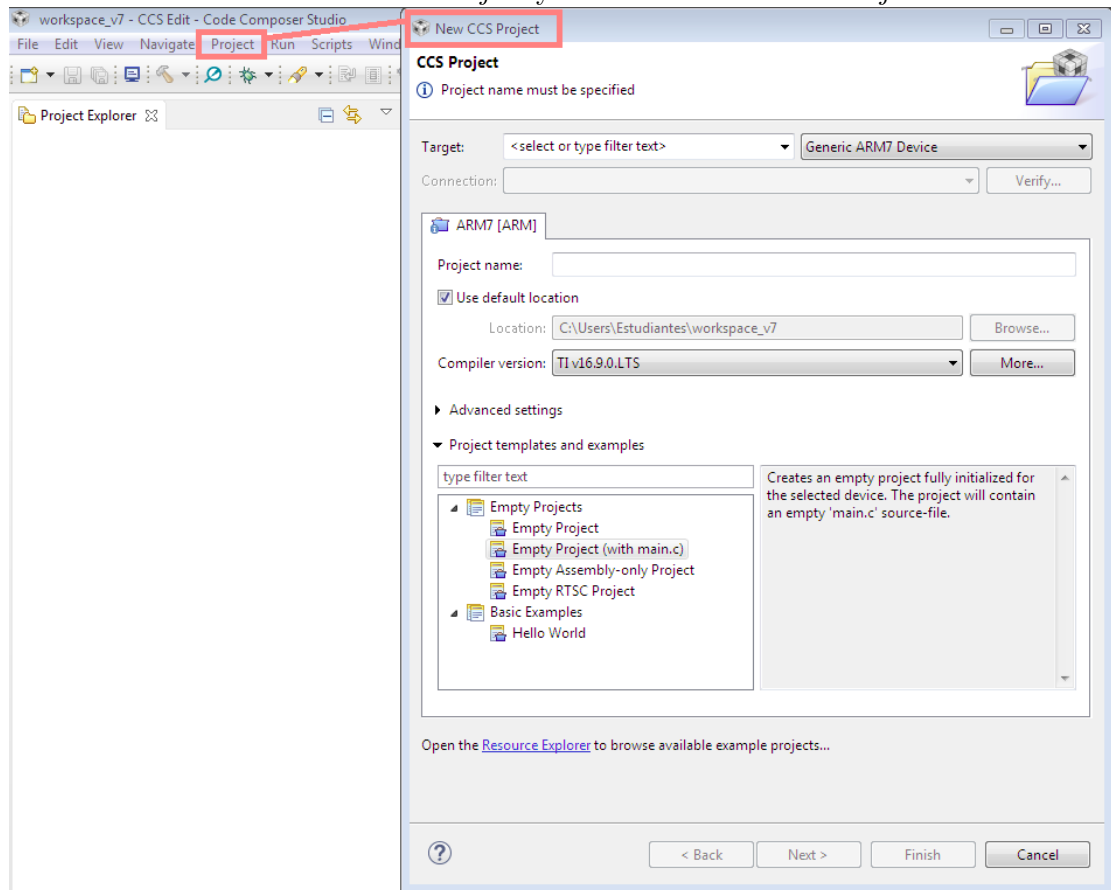
1. Conocer las secciones principales del ambiente de desarrollo.



2. Crear un proyecto nuevo para el μC TM4C1294NCDPT para escribir un programa en lenguaje ensamblador (*assembly*).
 - Al presionar el icono del programa se abre una ventana que nos pregunta la ruta para el workspace, nos debemos fijar donde le indicamos y dar *ok*:



- En el menu de archivo buscamos *Project* y le indicamos un *New Project*:



- En el menu de *New Project* llenamos los campos, tal como se observa:

New CCS Project

CCS Project

Create a new CCS Project.

Nuestra tarjeta

Target: Tiva C Series Tiva TM4C1294NCPDT

Connection: Stellaris In-Circuit Debug Interface Verify...

Cortex M [ARM]

Project name: Lab02

☒ Use default location

Location: C:\Users\Estudiantes\workspace_v7\Lab02 Browse...

Compiler version: TI v16.9.0.LTS More...

Advanced settings

Project templates and examples

type filter text

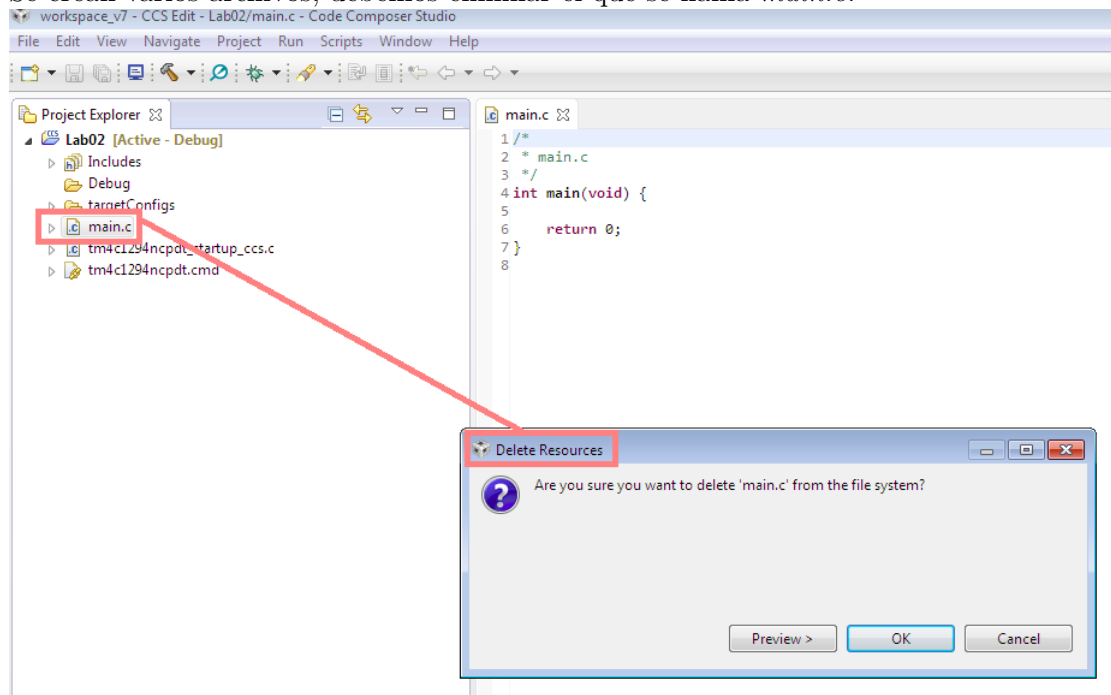
- Empty Projects
 - Empty Project
 - Empty Project (with main.c)**
 - Empty Assembly-only Project
 - Empty RTSC Project
- Basic Examples
 - Hello World

Creates an empty project fully initialized for the selected device. The project will contain an empty 'main.c' source-file.

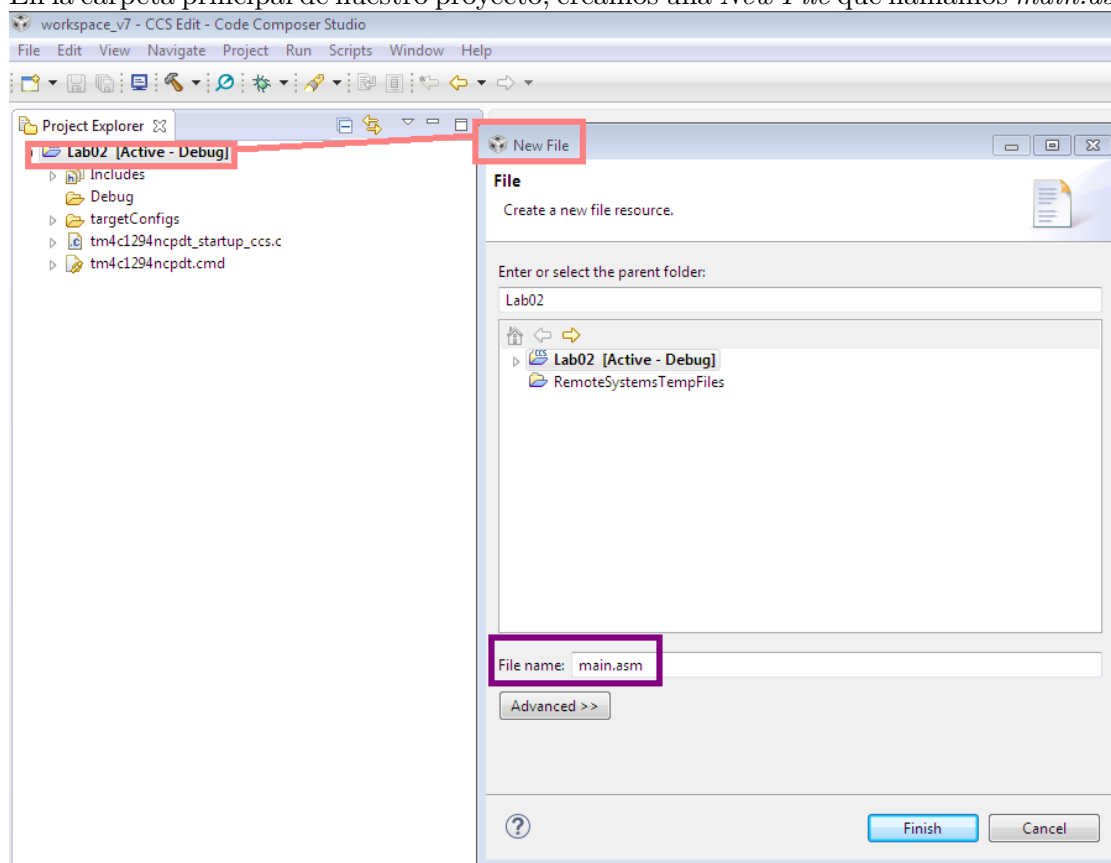
Open the [Resource Explorer](#) to browse available example projects...

? < Back Next > Finish Cancel

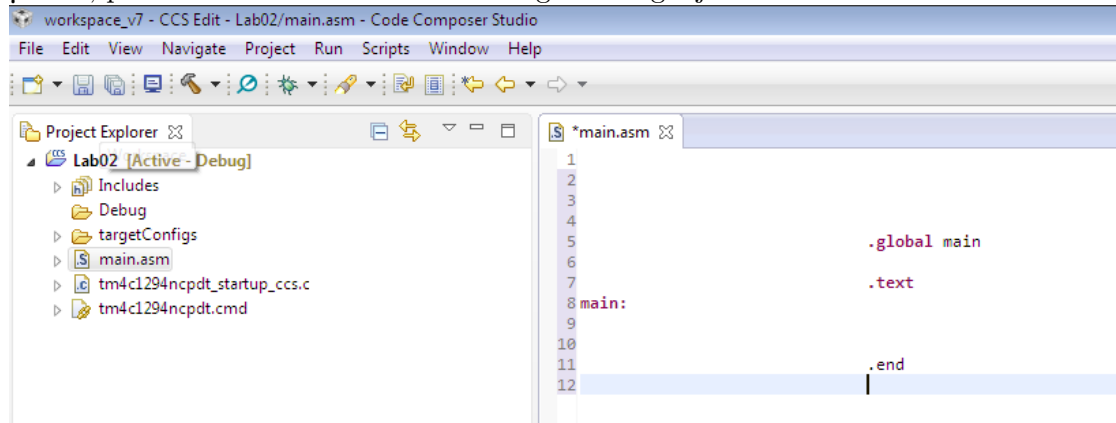
- Se crean varios archivos, debemos eliminar el que se llama *main.c*:



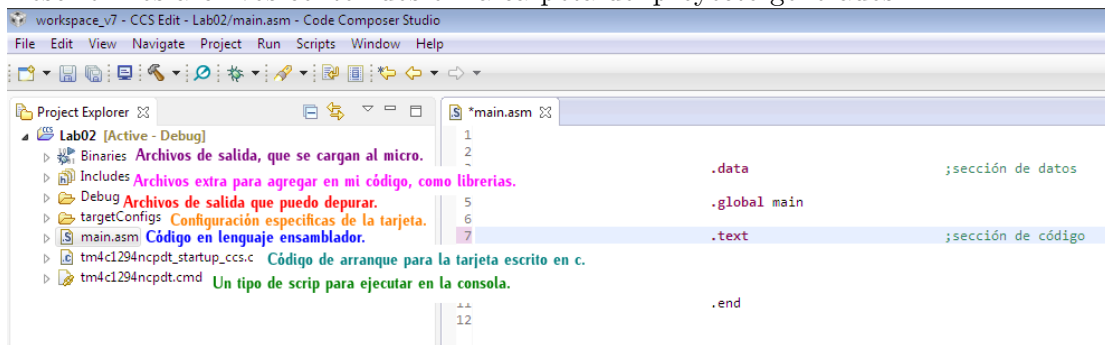
- En la carpeta principal de nuestro proyecto, creamos una *New File* que llamamos *main.asm*:



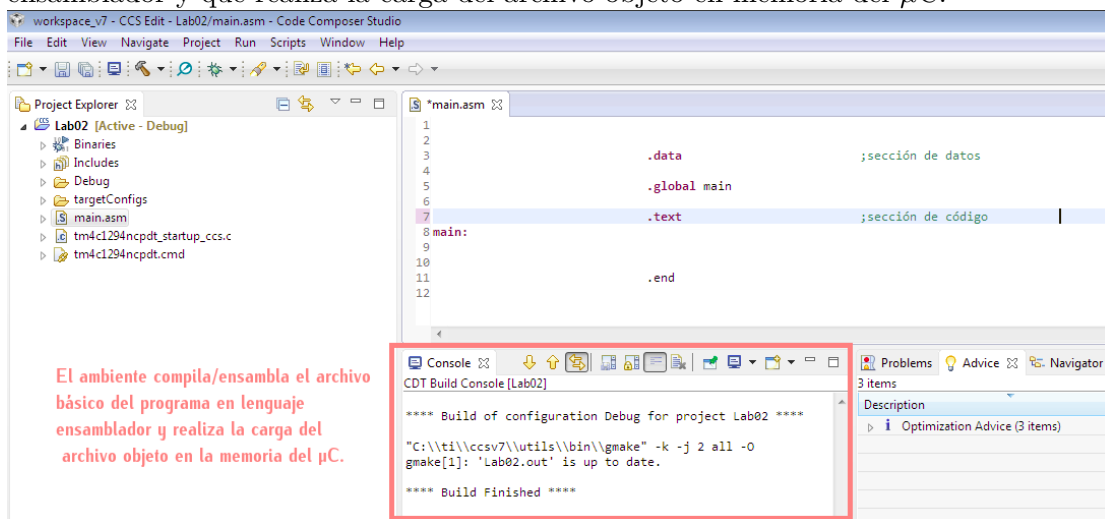
- ¡Listo!, podemos comenzar a escribir código en lenguaje ensamblador:



3. Describir los archivos contenidos en la carpeta del proyecto generados.

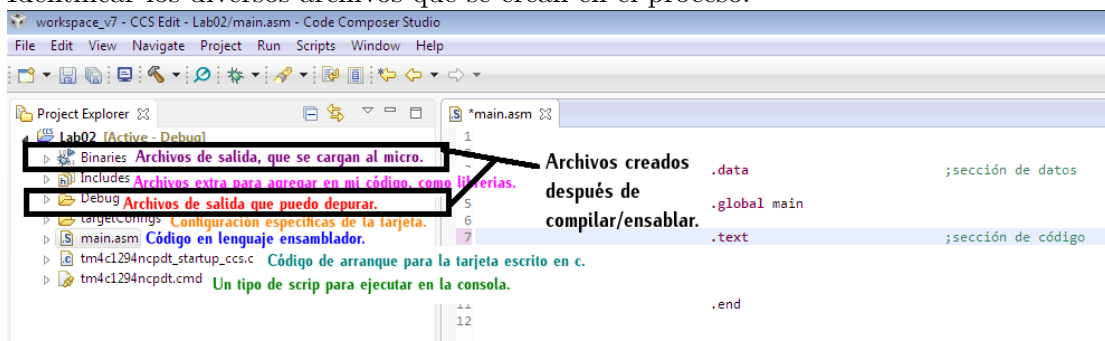


4. Comprobar que el ambiente compila/ensambla el archivo básico del programa en lenguaje ensamblador y que realiza la carga del archivo objeto en memoria del μC .

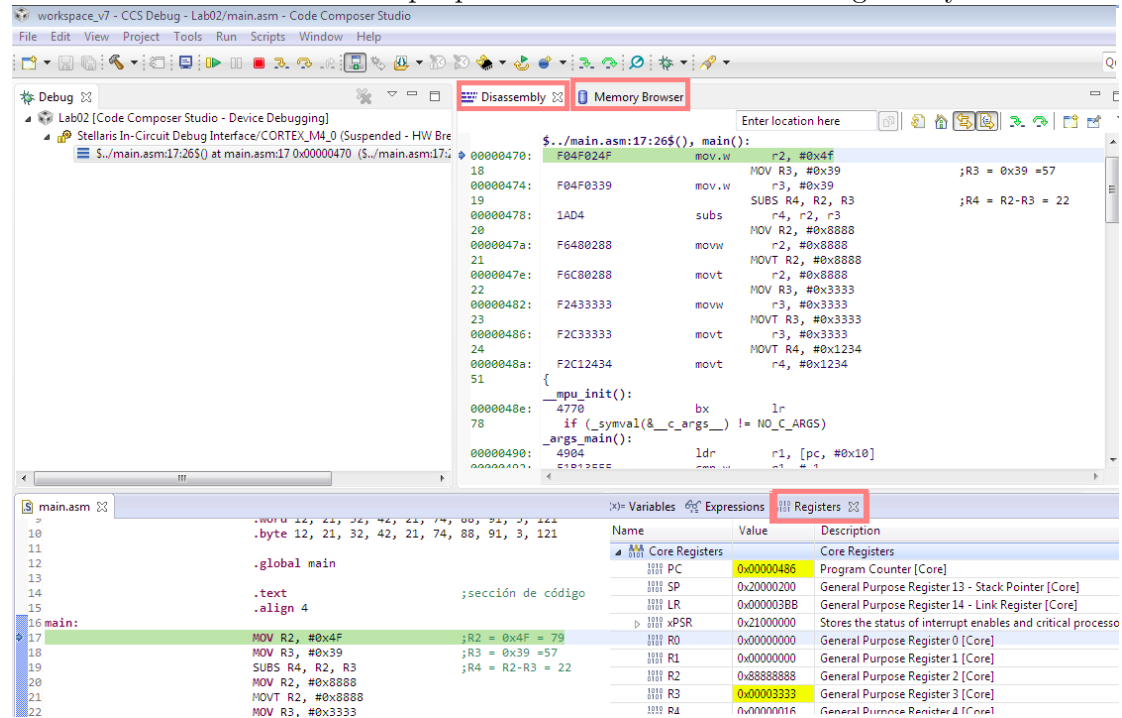


El ambiente compila/ensambla el archivo básico del programa en lenguaje ensamblador y realiza la carga del archivo objeto en la memoria del μC .

5. Identificar los diversos archivos que se crean en el proceso.



6. Conocer las diversas ventanas que presentan información de los registros y memoria del μC .

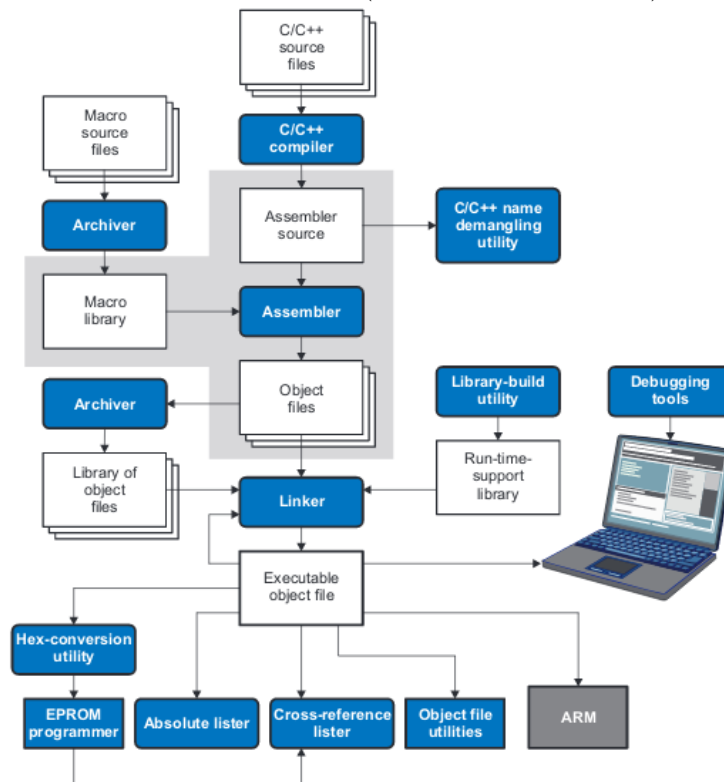


5. Cuestionario

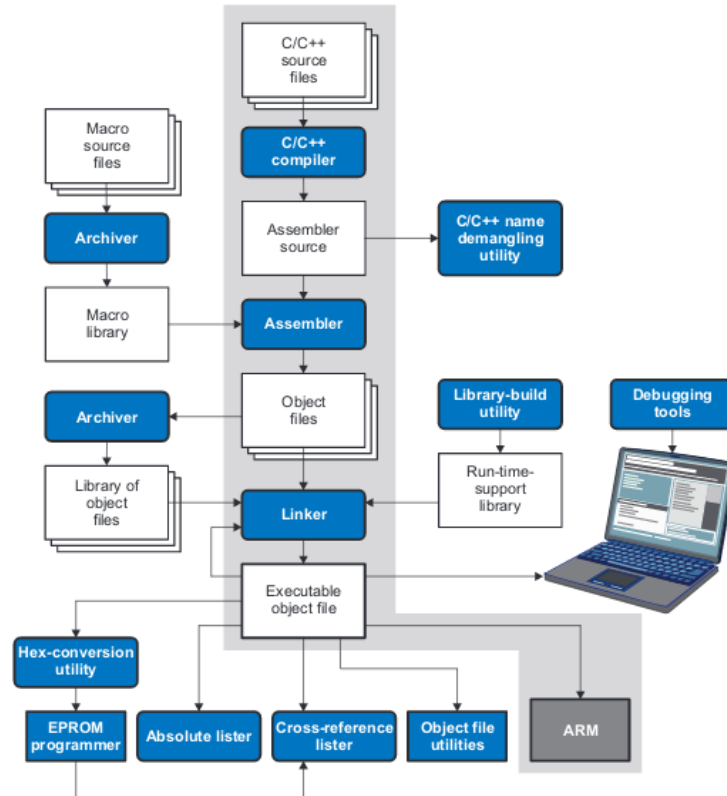
Describe los pasos que siguió para crear un proyecto nuevo en lenguaje ensamblador.

Están descritas en el punto dos del Desarrollo.

¿Cuál es el proceso que se sigue para la creación de un archivo objeto (ejecutable), para un μC solamente en lenguaje Assembly? (spnu118o, Figura 4-1)



¿Cuál es el proceso que se sigue para la creación de un archivo objeto (ejecutable), para un μC en un lenguaje de alto nivel como C/C++? (spnu118o, Figura 1.1)



¿Para qué sirven las directivas siguientes?: .data, .text, .bss

La directiva .text indica que inicia el código ejecutable, la sección .data indica contiene información inicializada y la directiva .bss indica cuanto espacio debe reservar para las variables no inicializadas.

¿Qué pasará si coloca la sección del programa principal en la sección .bss?

El programa no se ejecutaría.

6. Conclusiones.

Con la realización de esta práctica me familiarize aún más con el IDE Code Composer Studio y aprendí todos los pasos que se deben llevar a cabo para crear un proyecto en lenguaje ensamblador. Me di cuenta además, de todo el proceso que se debe llevar para programar el microcontrolador (el IDE, tiene un editor, donde escribir el código, un ensamblador, un ligador, un depurador), y comprendí porque cuando creamos el proyecto, éste en el *project explorer* tiene varios archivos, unos son para el ensamblador y que el archivo que me da el ligador es un **executable object file** (en *binaries*) que se puede cargar en el microprocesador.

Referencias

- [1] ARM Assembly Language Tools v15.12.0.LTS - User's Guide. Literature Number: SPNU118O January 2016. Disponible en <http://downloads.ti.com/docs/esd/SPNU118/index.html>