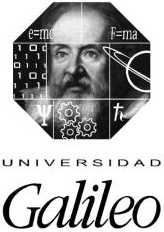
|  |  |
| --- | --- |
| Universidad Galileo | Guatemala 09 de Agosto del 2021 |
| Facultad: FISICC | **Alumno: Oscar Barrios** |
| Curso: Arquitectura del Computador | **Carnet: 09000120** |
| Sección: A | **Hora de Laboratorio: 19:00 a 19:49** |
| Auxiliar: Jabes Guerra | **Día de Laboratorio: Miércoles** |



L

**Laboratorio # 01 Keil MDK 5**

# Objetivos:

Aprender el procedimiento necesario para desarrollar una aplicación utilizando el IDE Keil uVision5 con el lenguaje de programación C.

# Resumen:

## Materiales y Equipo: • PC con Windows 10 y Keil uVision5

Primero necesitaremos instalar Keil uVision5 en la computadora, luego de instalado necesitaremos crear un nuevo proyecto para el microcontrolador ARM Cortex M4, a este proyecto le deberemos incluir los ambientes de CMSIS Core, STDOUT Compile y el Startup del device. Una vez creado el proyecto debemos crear los archivos necesarios para poder desarrollar los métodos de bitSet, bitClear, bitToggle y stringLength.

# Teoría:

**Keil uVision5**

Ambiente de desarrollo de software para microcontroladores basados en Arm, el cual incluye todos los componentes necesarios para crear, construir y depurar aplicaciones integradas.

**Microprocesadores**

Es el circuito integrado central de un ordenador, este suele ser el mas complejo dentro del sistema informático.

**Microcontroladores**

Es u circuito integrado programable, este es capaz de ejecutar las instrucciones grabadas dentro de su memoria. Está compuesto por varios bloques funcionales los cuales cumplen tareas específicas.

**ARM Cortex-M**

Es un grupo de procesadores RISC ARM de 32 bits licenciados por Arm Holdings. Estos núcleos han sido optimizados para circuitos integrados de bajo costo y eficiencia energética.

**Lenguaje ensamblador**

Conocido como *assembly languaje*, es un lenguaje de programación de muy bajo nivel, este consiste en un conjunto de instrucciones básicas para los computadores, microprocesadores y microcontroladores.

**Lenguaje C**

Es un lenguaje de programación de propósito general basado en el lenguaje B, este es un lenguaje orientado a la implementación de sistemas operativos.

# Datos Prácticos:

**bitSet**

Texto

Descripción generada automáticamente con confianza media

**bitClear**

Texto

Descripción generada automáticamente con confianza media

**bitToggle**

Texto

Descripción generada automáticamente

**stringLength**

Texto

Descripción generada automáticamente

# Cálculos Teóricos:

**bitSet**

* Encabezado



* Implementación

Gráfico

Descripción generada automáticamente con confianza baja

Para este método hacemos un corrimiento a la izquierda de n cantidad que se indique en la variable **bit**, para luego hacer un **OR** a nivel de bit por lo que si el bit de **ptr** en la posición de bit es 0 pasara a ser 1 y si el mismo ya estaba en 1 se quedara en 1.

Imagen que contiene Forma

Descripción generada automáticamente

**bitClear**

* Encabezado



* Implementación

Imagen que contiene Gráfico

Descripción generada automáticamente

Para este método hacemos un corrimiento a la izquierda de n cantidad que se indique en la variable **bit**, luego de hacer este corrimiento negamos el resultado, para luego hacer un **AND** a nivel de bit por lo que si el bit de **ptr** en la posición de bit es 1 pasara a ser 0 y si el mismo ya estaba en 0 se quedara en 0.

Un dibujo de una persona

Descripción generada automáticamente con confianza baja

**bitToggle**

* Encabezado



* Implementación

Imagen que contiene Gráfico

Descripción generada automáticamente

Para este método hacemos un corrimiento a la izquierda de n cantidad que se indique en la variable **bit**, para luego hacer un **XOR** a nivel de bit por lo que si el bit de **ptr** en la posición de bit es 0 pasara a ser 1 y si el bit es 1 pasara a ser 0.

Imagen que contiene Forma

Descripción generada automáticamente

**stringLength**

* Encabezado



* Implementación

Texto

Descripción generada automáticamente

Para este método se utilizó la función **strlen** la cual ya nos retorna la cantidad de caracteres contenida en la variable **str**.

# Conclusiones:

Por el momento hemos aprendido a instalar Keil, importar ciertos componentes que nos ayudaran a programar los microprocesadores y sobre como funcionan los punteros, es importante notar que en este caso debemos utilizar punteros para que podamos modificar el valor de la variable ya que si no C creara una copia de esta dentro del método impídenos así modificar el valor de la variable pasada al método. Adicional a esto hemos aprendido a realizar operaciones a nivel de bit llamadas *bitwise operations* las cuales son importantes ya que nos ayudan a enmascarar los bits de las variables.

# Bibliografía:

Keil uVision5: <https://www2.keil.com/mdk5>

Microprocesadores: <https://es.wikipedia.org/wiki/Microprocesador>

Microcontrolador: <https://es.wikipedia.org/wiki/Microcontrolador>

ARM Cortex-M: <https://en.wikipedia.org/wiki/ARM_Cortex-M>

Lenguaje Ensamblador: <https://es.wikipedia.org/wiki/Lenguaje_ensamblador>

Lenguaje C: <https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)>