Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería

MICROCONTROLADORES

Practica No. 1

Introducción

Docente: Castro Gonzalez Ricardo

Alumno: Gómez Cárdenas Emmanuel Alberto

Matricula: 01261509

Objetivo:

El alumno se familiarizará con algunos usos típicos del Lenguaje C en sistemas embebidos.

Material:

- Computadora Personal

Teoría:

- **Lenguaje C y su uso en sistemas embebidos.**

El lenguaje C es un lenguaje eficiente que permite un control a nivel de hardware, ofrece portabilidad y la capacidad de trabajar con recursos limitados, gracias a esto es una elección muy popular para el desarrollo de sistemas embebidos.

- **Compilador Cruzado (Cross-Compiler)**

Un cross-compiler (compilador cruzado) es una herramienta de desarrollo de software que permite compilar código fuente en un lenguaje de programación en una plataforma de hardware diferente de aquella en la que se está ejecutando el compilador. En otras palabras, un cross-compiler genera código ejecutable que se ejecutará en una arquitectura de CPU y un sistema operativo distintos de los del sistema en el que se encuentra el compilador.

Este tipo de compilador es especialmente útil en el desarrollo de software para sistemas embebidos, donde los recursos pueden ser limitados y las arquitecturas de hardware pueden variar ampliamente. En lugar de compilar el código directamente en la plataforma de destino (lo que puede ser ineficiente o incluso imposible en algunos casos), se utiliza un cross-compiler para generar código optimizado que se ejecutará en la plataforma objetivo.

Desarrollo:

Investigar:

- Tamaño de las primitivas y uso de **inttypes.h**.

La biblioteca **inttypes.h** fue creada con el objetivo de proveer un set de tipos las cuales tienen una definición constante en todas las máquinas o sistemas operativos, el mismo nombre indica el número de bits y si es signado o no.

- Representación de literales (1, 0x1, 0b1, 0)

En lenguaje C existen dos tipos de representación de literales:

- **Prefijos:** Indican la base en que se debe leer
  - **Base Decimal:** Empieza con un dígito decimal distion de cero, seguido de cualquier dígito decimal.
  - **Base Octal:** Un **0** seguido de cualquier dígito octal (0-7).
  - **Base Hexadecimal: 0x** o **0X** seguido de cualquier dígito hexadecimal (0-F/f).
  - **Base Binaria: 0b** o **0B** seguido de cualquier dígito binario (0-1).

- **Sufijo:** Indican el tipo en que se leerá.
    - **Int:** No se requiere sufijo.
    - **Unsigned int: "**U" o "u" al final de una constante entera.
    - **Long int:** l o L al final de una constante entera.
    - **Unsigned long int:** "ul" o "UL" al final de una constante entera.
    - **Long long int:** "ll" o "LL" al final de una constante entera.
    - **Unsigned long long int:** "ull" o "ULL" al final de una constante entera.

• Operadores Bitwise ({&, |, ~, ^, >>, << } vs {&&, ||, !})

- **Bit a bit:**
    - **AND:** operación and, 100 & 101 == 100
    - **OR:** operación or, 100 | 101 == 101
    - **NEG:** operación negación, 100 == 011
    - **XOR:** operación xor, 100 ^ 101 == 100
    - **AND:** operación and, 100 & 101 == 100
    - **SHIFTS:** operación desplazamiento, 100 >> 1 == 010 ó 100 << 1 == 000
- **Lógicos:**
    - **AND:** and lógica, true && false == false
    - **OR:** or lógica, true || false == false
    - **NOT:** negación lógica, !true == false
- • **Apuntadores** (char *x, int (*cb)(void))
    - Un puntero es una variable que almacena la dirección de memoria de un objeto. Los punteros se usan ampliamente en C y C++ para tres propósitos principales: para asignar nuevos objetos en el montón, para pasar funciones a otras funciones para iterar sobre elementos en matrices u otras estructuras de datos.
      Puede ser utilizado para apuntar hacia una variable char *x o para apuntar a una función int (*cb)(void).

• **Estructuras** (struct, union, bitfield)

- **Struct:** Un tipo de estructura es un tipo compuesto definido por el usuario. Se compone de campos o de miembros que pueden tener diferentes tipos.
- **Union:** es un tipo definido por el usuario en el que todos los miembros comparten la misma ubicación de memoria. Esta definición significa que, en un momento dado, una union no puede contener más de un objeto de su lista de miembros. También significa que, independientemente de cuántos miembros tiene una union, en todo momento usa únicamente la memoria suficiente para almacenar al miembro más grande.
- **Bitfield:** una estructura también puede ser de un número específico de bits, llamado "campo de bits". Su longitud se separa del declarador del nombre del campo mediante dos puntos. Un campo de bits se interpreta como un tipo integral.

- Palabras claves (keywords: const, static, volatile) tienen un significado especial para el compilador

    - **Const:** Especifíca que un objeto o variable es constante
    - **Static:** Una variable estática specifíca que una variable o función mantiene sus valores aún despues de que la función ha terminado.
    - **Volatile:** Es utilizada para indicar al compilador que la variable sea omitida de cualquier optimización

- Condicionales del pre-compilador (#)

Las directivas del precompilador le indican al preprocesador ejecutar ciertas acciones al código antes de la compilación como lo son #IF-#ELIF-#ELSE-#ENDIF entre otras.

- Casteo (type casting)

El operador de casteo **"()"** provee un método para realizar una conversión de un tipo de objeto en una situación específica.

- git (github)

Git es un sistema de control de versiones que permite a los desarrolladores a mantener un record de los camvios hechos en el código a traves del tiempo. Además de que lo hace mas sencillo de colaborar con otros, revertir a versiones anteriores y mantener un historial de los cambios.

Conclusiones y comentarios:

En esta práctica, se enfatizó el uso del lenguaje C en los sistemas embebidos, además de servir como una revisión del lenguaje que cubrió temas como literales en diversas bases, operadores bitwise y lógicos, apuntadores, estructuras y palabras clave.

Bibliografía:

https://pubs.opengroup.org/onlinepubs/009695399/basedefs/inttypes.h.html

https://www.ibm.com/docs/es/aix/7.3?topic=files-inttypesh-file

https://barcelonageeks.com/tipos-de-literales-en-c-c-con-ejemplos/

https://ortizol.blogspot.com/2013/07/operadores-bitwise-bit-bit-en-c.html

https://learn.microsoft.com/es-es/cpp/c-language