Universidad Autónoma de Baja California

Facultad de ciencias químicas e ingeniería

Materia.

Microprocesadores y Microcontroladores.

Maestro.

Garcia Lopez Jesus Adan.

Alumno

Gonzalez Cardiel Luis Enrique

Matricula:

1217258

Grupo:

561

Trabajo:

Practica No.2

31/08/2018

Práctica No. 2

Introducción al intérprete 80X86 sobre la plataforma T-Juino

Objetivo: El alumno se familiarizará con el intérprete 80X86 (modo 16bits) sobre la plataforma T-Juino. Esto con el fin de desarrollar programas en lenguaje C para dicha plataforma.

Material:
- Computadora Personal (PC)
- Programa Editor de texto (ASCII), TASM y TLINK
- Tarjeta T-Juino (con intérprete 80x86)
- Manejador FTDI instalado

Equipo:
- Computadora Personal
- Programa emulador de terminal

Teoría: - Lenguaje C, Lenguaje Ensamblador y combinación de éstos en sistemas embebidos.

Desarrollo:

1) Basándose en listado 1 crear in archivo texto llamado **pra2.**

Listado 1 *

```
void putchar( char dato );
void puts( char *str );
char getch( void );
void printdec(unsigned char dato );

char msg[]="Hola UABC\r\n";

unsigned char i=0;

int main ( void ){

    while(1){

        printdec(i++);
        puts( msg );
        getch();

    }
    return 0;
}


void puts ( char *str )
{
    while( *str )
        putchar( *str++ );
}
```

```
void putchar ( char dato )
{
    asm mov dl,dato asm mov ah,2 asm int
    21h
}
char getch( void )
{
    char dato;

    asm mov ah,8 asm int  21h asm mov
    dato,al

    return dato;
}
void printdec ( unsigned char dato )
{

    putchar( dato/100 + 0x30 ); dato%=100;
    putchar( dato/10 + 0x30 ); putchar(
    dato%10 + 0x30 );

}
```

2) Descargue los programas **tcc.exe**, **tasm.exe, tlink.exe y mkbintj.exe** de moodle (curso Microprocesadores y Microcontroladores en http://uabc-lan.net) y deposítelos en un directorio exclusivo de trabajo (por ejemplo. C:\uPuC\Prac2).

3) Compile el programa **prac2.c** mediante la línea de comando:

```
C:\uPuC\Prac2>tcc –c pra2.c
```

Esto generará el archivo **prac2.obj**

4) Encadena el archivo **obj** para generar programa **.exe** mediante la línea de comando:

```
C:\uPuC\Prac2>tlink initj prac2, prac2
```

Esto generará el archivo ejecutable **prac2.exe**

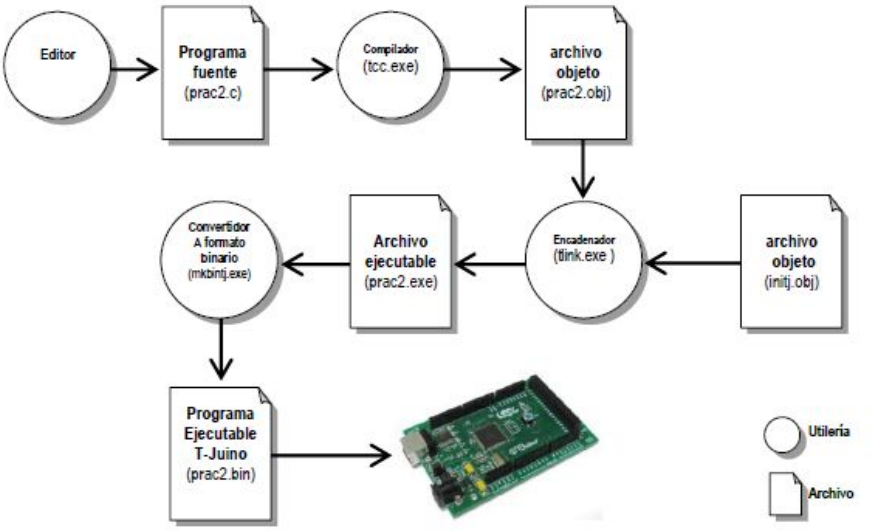

Figura 1. Proceso de generación de archivo ejecutable para T-Juino.

5) Convertir archivo **prac2.exe** a formato binario para la placa T-Juino haciendo uso de la herramienta **mkbintj.exe**

```
C:\uPuC\Prac2>mkbintj prac2
```

Esto generará el archivo **prac2.bin** el cual puede cargarse y ejecutarse en T-Juino.

6) Verifique el funcionamiento del programa cargándolo y ejecutándolo en T-Juino.

- **Implementar las siguientes funciones:**

  a) `char` **`getchar`**`(void)`**:** Función que retorna el carácter capturado del teclado, en ASM fuera de línea.
  b) `void` **`putchar`**`(char data)`**:** Función que manda un carácter a la pantalla, en ASM fuera de línea.
  c) `void` **`gets`**`(char *str)`: Función que retorna una cadena haciendo uso de *getchar*, la cadena se retorna en el apuntador *str*.
  d) `void` **`puts`**`(char *str)`**:** Función que imprime una cadena mediante *putchar*.
  e) `void` **`itoa`**`(char* str, unsigned int number, unsigned char base)` **:** Función que convierte un número de 16 bits a su representación alfanumérica en la base dada, y la retorna en el apuntador *str*.
  f) `unsigned int` **`atoi`**`(char *str)` **:** Función que convierte una cadena numérica (de base 10) y retorna su valor numérico en 16 bits.

- **Con éstas funciones implementar un programa que captura un número, lo imprime en hexadecimal y binario, y vuelve a solicitar un número, repitiendo esto infinitamente.**

- **Realizar los ejercicios que están relacionados con la Práctica.**

Teoría:

El lenguaje C es sin duda el más apropiado para la programación de sistemas, pudiendo sustituir al ensamblador en muchos casos. Sin embargo, hay ocasiones en que es necesario acceder a un nivel más bajo por razones de operatividad e incluso de necesidad (programas residentes que economicen memoria, algoritmos rápidos para operaciones críticas, etc.). Es entonces cuando resulta evidente la necesidad de poder emplear el ensamblador y el C a la vez.

Para comprender este capítulo, basta tener unos conocimientos razonables de C estándar. Aquí se explicarán las funciones de librería necesarias para acceder al más bajo nivel, así como la manera de integrar el ensamblador y el C.

Los sistemas embebidos se pueden programar directamente en el lenguaje ensamblador del microcontrolador o microprocesador incorporado sobre el mismo, o también, utilizando los compiladores específicos, pueden utilizarse lenguajes como C o C++; en algunos casos, cuando el tiempo de respuesta de la aplicación.

A veces es bueno contar con capacidad de procesamiento suficiente como para poder: - Usar lenguajes de programación de alto nivel. - Usar sistemas operativos.
Cuando las rutinas a incluir son excesivamente largas, resulta más conveniente escribirlas como ficheros independientes y ensamblarlas por separado, incluyéndolas en un fichero de proyecto (*.PRJ) seleccionable en los menús del compilador.

Para escribir este tipo de rutinas hay que respetar las mismas definiciones de segmentos que realiza el compilador. Hoy en día existe algo más de flexibilidad; sin embargo, aquí se expone el método general para mezclar código de ensamblador con C.

Conclusiones y Comentarios.

La combinación de C y ensamblador es una muy buena combinación para los microcontroladores ya que esto simplificaria demasiado el código en ensamblador. Se me hizo interesante la parte del ASM fuera dentro de línea ya que no la conocía, desconocía esa forma de programar aunque no la aplicamos para esta práctica fue bueno el saber que existe esa manera de programar dentro del código de C en asm.