Universidad Autónoma de Baja California
FAC. DE CS. QUIM. E INGENIERIA
INGENIERIA EN COMPUTACION

PRACTICA 8b

Laboratorio de: Microprocesadores y microcontradores

Equipo:

López Madrigal Leonardo

Maestro:
García López Jesús Adán

Tijuana, B. C. 27 Abril, 2017

Programación del uC en lenguaje C y comunicación serie.

Objetivo: Mediante esta práctica el alumno aprenderá el uso básico de la programación en lenguaje C con las herramientas AVR Studio y WinAVR. Para ello el alumno

implementará los procedimientos comunes para inicializar y operar el puerto serie del microcontrolador.

Equipo:

- Computadora Personal.
- Módulo T-Juino

Teoría:

- Programación en lenguaje C en microcontroladores.
- Manejo del Periférico de Comunicación Serie 0 (UART0) del microcontrolador ATmega1280/2560.

Teoría:

- Programación en lenguaje C en microcontroladores.

C es un lenguaje bastante conciso y en ocasiones desconcertante. Considerado ampliamente como un lenguaje de alto nivel, posee muchas características importantes, tales como: programación estructurada, un método definido para llamada a funciones y para paso de parámetros, potentes estructuras de control, etc. Sin embargo gran parte de la potencia de C reside en su habilidad para combinar comandos simples de bajo nivel, en complicadas funciones de alto nivel, y en permitir el acceso a los bytes y words del procesador. En cierto modo, C puede considerarse como una clase de lenguaje ensamblador universal. La mayor parte de los programadores familiarizados con C, lo han utilizado para programar grandes máquinas que corren Unix, MS-DOS, e incluso Windows (programación de drivers). En estas máquinas el tamaño del programa no es importante, y el interface con el mundo real se realiza a través de llamadas a funciones o mediante interrupciones DOS. Así el programador en C sólo debe preocuparse en la manipulación de variables, cadenas, matrices, etc. Con los modernos microcontroladores de 8 bits, la situación es algo distinta. Tomando como ejemplo el 8051, el tamaño total del programa debe ser inferior a los 4 u 8K (dependiendo del tamaño de la EEPROM), y debe usarse menos de 128 o 256 bytes de RAM. Idealmente, los dispositivos reales y los registros de funciones especiales deben ser direccionados desde C. Las interrupciones, que requieren vectores en direcciones absolutas también deben ser atendidas desde C.

Además, se debe tener un cuidado especial con las rutinas de ubicación de datos para evitar la sobre escritura de datos existentes. Uno de los fundamentos de C es que los parámetros (variables de entrada) se pasan a las funciones (subrutinas) en la pila, y los resultados se devuelven también en la pila. Así las funciones pueden ser llamadas desde las interrupciones y desde el programa principal sin temor a que las variables locales sean sobre escritas. Una seria restricción de la familia 8051 es la carencia de una verdadera pila.

En un procesador como el 8086, el apuntador de la pila tiene al menos 16 bits. Además del apuntador de pila, hay otros registros que pueden actuar como apuntadores a datos en la pila, tal como el BP (Base Pointer). En C, la habilidad para acceder a los datos en la pila es crucial. Como ya ha sido indicado, la familia 8051 está dotada de una pila que realmente sólo es capaz de manejar direcciones de retorno. Con 256 bytes disponibles, como máximo, para la pila no se pueden pasar muchos parámetros y realizar llamadas a muchas funciones. De todo ello, puede pensarse que la implementación de un lenguaje que como C haga un uso intensivo de la pila, es imposible en un 8051. Hasta hace poco así ha sido. El 8051, hace tiempo que dispone de compiladores C, que en su mayor parte han sido adaptados de micros más potentes, tal como el 68000. Por ello la aproximación al problema de la pila se ha realizado creando pilas artificiales por software. Típicamente se ha apartado un área de RAM externa para que funcione como una pila, con la ayuda de rutinas que manejan la pila cada vez que se realizan llamadas a funciones. Este método funciona y proporciona capacidad de repetir variables locales a distintos niveles sin sobre escritura, pero a costa de hacer los programas muy lentos. Por lo tanto, con la familia 8051, la programación en lenguaje ensamblador ha sido la única alternativa real para el desarrollo de pequeños sistemas en los que el tiempo es un factor crítico.

- Manejo del Periférico de Comunicación Serie 0 (UART0) del microcontrolador ATmega1280/2560.

Una interrupción (del inglés interrupt request, en español «petición de interrupción») es una señal recibida por el procesador de una computadora, para indicarle que debe «interrumpir» el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situación.

Una interrupción es una suspensión temporal de la ejecución de un proceso, para pasar a ejecutar una subrutina de servicio de interrupción, la cual, por lo general, no forma parte del programa, sino que pertenece al sistema operativo o al BIOS. Una vez finalizada dicha subrutina, se reanuda la ejecución del programa.

Las interrupciones son generadas por los dispositivos periféricos habilitando una señal del CPU (llamada IRQ del inglés "interrupt request") para solicitar atención del mismo. Por ejemplo. cuando un disco duro completa una lectura solicita atención al igual que cada vez que se presiona una tecla o se mueve el ratón.

La primera técnica que se empleó para esto fue el polling, que consistía en que el propio procesador se encargara de sondear los dispositivos periféricos cada cierto tiempo para averiguar si tenía pendiente alguna comunicación para él. Este método presentaba el inconveniente de ser muy ineficiente, ya que el procesador consumía constantemente tiempo y recursos en realizar estas instrucciones de sondeo.

El mecanismo de interrupciones fue la solución que permitió al procesador desentenderse de esta problemática, y delegar en el dispositivo periférico la responsabilidad de comunicarse con él cuando lo necesitara. El procesador, en este caso, no sondea a ningún dispositivo, sino que queda a la espera de que estos le avisen (le "interrumpan") cuando tengan algo que comunicarle (ya sea un evento, una transferencia de información, una condición de error, etc.).

Interrupciones del ATmega1280/2560

Manipulación global de la bandera de interrupción

El indicador de interrupción global se mantiene en el bit I del registro de estado (SREG).

Manejar las interrupciones con frecuencia requiere atención con respecto al acceso atómico a los objetos que podrían ser alterados por el código que se ejecuta dentro de un contexto de interrupción, vea <util / atomic.h>.

Con frecuencia, las interrupciones se desactivan durante periodos de tiempo para realizar ciertas operaciones sin ser perturbadas; Vea Problemas con el código de reordenación de las cosas que se deben tener en cuenta con respecto a las optimizaciones del compilador.

```
#define sei()
#define cli()
```

Macros usando interrupciones

```
#define ISR(vector, attributes)
#define SIGNAL(vector)
#define EMPTY_INTERRUPT(vector)
#define ISR_ALIAS(vector, target_vector)
#define reti()
#define BADISR_vect
```

ISR atributos

```
#define ISR_BLOCK
#define ISR_NOBLOCK
#define ISR_NAKED
#define ISR_ALIASOF(target_vector)
```

Conclusión:

López Madrigal Leonardo

En la práctica 8b hicimos varias funciones de getchar , putchar e ISR las cuales son interrupciones que transmitían los datos que tenía la cola de datos, en getchar cheque si la cola estaba en condiciones para recibir datos y putchar solo para transmitir o mostrar en pantalla los datos de la cola.

Bibliografía o referencias:

- **http://www.atmel.com/Images/Atmel-0856-AVR-Instruction-Set-Manual.pdf**