Práctica 2. Lenguaje Ensamblador, Interrupciones y Temporizadores

Emiliano Sotomayor González
155902
uboat46@gmail.com

Jaime Limón Samperio
162395
jalisa2808@gmail.com

***Abstract*—Se explora el uso de interrupciones a través de temporizadore haciendo uso del lenguaje ensamblador. De igual forma, se analiza la diferencia entre una implementación con lenguajes de alto nivel y lenguaje ensamblador.**

I. Introducción

Temporizadores son una de las funciones mas usadas en microcontroladores y por esta razon es importante conocer como se pueden implimentar de una manera optimizada. La mayoria de compiladores para microcontroladores traen funciones para hacer temporizaciones por delay, pero estos interrunpen la ejecucion principal del programa y no son muy practicos.

Casi todos los proyectos electrónicos con microcontrolador hacen uso de los temporizadores para su funcionamiento. Una de las técnicas más comunes, es usar un Timer físico (TIM) del microcontrolador para generar una interrupción a cada milisegundo y a partir de esta interrupción, incrementar o decrementar contadores para generar los tiempos de los temporizadores. Este método tiene un inconveniente cuando el proyecto usa muchos temporizadores, pues la rutina de interrupción puede usar mucho tiempo de la CPU. Otra técnica es usar temporizadores por delay, pero estos tienen el inconveniente de no poder estar leyendo las entradas del microcontrolador mientras se esté ejecutando el delay.

II. Marco Teórico

AVR utiliza una arquitectura de Harvard que es una arquitectura con memorias y buses separados para programas y datos. Las instrucciones en la memoria del programa se ejecutan con una canalización de un solo nivel. Esto significa que mientras se ejecuta una instrucción, la siguiente instrucción se obtiene previamente de la memoria del programa. Este concepto permite ejecutar la mayoría de las instrucciones en cada ciclo de reloj. Los lenguajes de ensamblaje son diferentes para cada procesador. Algunos ejemplos de lenguajes de ensamblaje están a continuación.

- ARM
- MIPS
- x86

III. Desarrollo

Se deberán de seguir las siguientes instrucciones:

- Analizar el conjunto reducido de instrucciones del microcontrolador ATMEGA y reportarbrevemente el tipo (categoria) de instrucciones que se pueden identificar
- Realizar un programa en Arduino que haga parpadear 4 diodos LED a diferentes frecuencias en 0.25, 0.5, 1 y 2 Hz respectivamente
- Implemetar la tarea anterior en código ensamblador y reportar diferencias
- Generar una rutina de tiempo de 1 segundo mediante instrucciones de ensamblador aplicadas directamente a registros y localidades de memoria
- Mediante dos juegos de diodos LED rojos, amarillos y verdes, simular en la protoboard los semaforos de un cruce peatonal en el que cada luz permanece encendida durante 6, 1 y 5 segundos respectivamente. El codigo debe ser implementado en lenguaje ensamblador, haciendo llamados a una unica rutina de tiempo.
- Crear un programa en Arduino que haga parpadear un LED a una frecuencia de 1 Hz.
- Implementar una rutina que incremente la variable de un contador cada vez que se hace una pulsacion en un push-button externo. Esta tarea no debe interferir con los tiempos de oscilacion del primer LED
- Sustituir la funcion del push-button con un interruptor optico.

IV. Resultados

Las ventajas y desventajas observadas en el uso de ensamblador se exponen a continuación.

- El rendimiento y la precisión del código del lenguaje ensamblador son mejores que los de alto nivel.
- El código del lenguaje ensamblador es más difícil de entender y depurar que un alto nivel.
- Una o dos declaraciones de lenguaje de alto nivel se expanden en muchos códigos de lenguaje ensamblador.
- Trabajar con bits es más fácil en lenguaje ensamblador.
- El código ejecutable del lenguaje de alto nivel es más grande que el código del lenguaje ensamblador, por lo que su ejecución toma más tiempo.
- Debido al largo código ejecutable, los programas de alto nivel son menos eficientes que los programas en lenguaje ensamblador.

978-1-7281-4387-3/19/$31.00 © 2019 IEEE

V. Conclusiones

Como se observa en la sección de resultados, existen ventajas y desventajas al momento de utilizar lenguaje ensamblador. El poder de procesamiento actual permite que el tiempo de compilación y ejecución de un lengiuaje de alto nivel sea muy rápido.

La manipulación de microcontroladores a bajo nivel es en ocasiones más eficiente aunque puede ser más complicado su manejo y requiere tener conocimineto del harware en cuestion.

VI. Repositorio

https://github.com/uboat46/principios-de-mecatronica.git

References

[1] "Linux man pages" Internet: "https://linux.die.net/man/"
[2] "Arduino reference" Internet: "https://www.arduino.cc/reference/en/"
[3] "GIT" Internet: "https://git-scm.com/"