Instituto Politécnico Nacional

Escuela Superior de Cómputo

Alumnos:

Esquivel Pérez Jonathan Alfredo
Hernández López Ángel Zait
Salgado Gallegos Jesús
Sánchez Pizano Irving Daniel

Profesor:

Pérez Pérez José Juan

Unidad de aprendizaje:

Introducción a los microcontroladores

Práctica 8

De código C a ASM

Grupo: 3CM8

Objetivo

Escribir un código en C a ensamblador(ASM), que recorre un bit en 4 posiciones cambiando la dirección mediante interrupciones para el Microcontrolador Atmega8535.

Introducción

Código C

C es un lenguaje de programación de propósito general: originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell, como evolución del anterior lenguaje B, a su vez basado en BCPL.

Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel, ya que dispone de las estructuras típicas de los lenguajes de alto nivel, pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

Código ASM

El lenguaje ensamblador o assembler es un lenguaje de programación de bajo nivel. Consiste en un conjunto de mnemónicos que representan instrucciones básicas para los computadores, microprocesadores, microcontroladores y otros circuitos integrados programables. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura de procesador y constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador. Cada arquitectura de procesador tiene su propio lenguaje ensamblador que usualmente es definida por el fabricante de hardware, y está basada en los mnemónicos que simbolizan los pasos de procesamiento (las instrucciones), los registros del procesador, las posiciones de memoria y otras características del lenguaje. Un lenguaje ensamblador es por lo tanto específico de cierta arquitectura de computador física (o virtual). Esto está en contraste con la mayoría de los lenguajes de programación de alto nivel, que idealmente son portables.

Un programa utilitario llamado ensamblador es usado para traducir sentencias del lenguaje ensamblador al código de máquina del computador objetivo. El ensamblador realiza una traducción más o menos isomorfa (un mapeo de uno a uno) desde las sentencias mnemónicas a las instrucciones y datos de máquina. Esto está en contraste con los lenguajes de alto nivel, en los cuales una sola declaración generalmente da lugar a muchas instrucciones de máquina.

Muchos sofisticados ensambladores ofrecen mecanismos adicionales para facilitar el desarrollo del programa, controlar el proceso de ensamblaje, y la ayuda de

depuración. Particularmente, la mayoría de los ensambladores modernos incluyen una facilidad de macro (descrita más abajo), y se llaman macroensambladores.

WinAvr

WinAvr es una suite de archivos ejecutables, de código abierto para el desarrollo de aplicaciones basadas en microcontroladores AVR de ATMEL, que corre en la plataforma de Windows. Incluye el compilador GNU GCC para C y C++.

WinAVR incluye todo lo necesario para el desarrollo en microcontroladores AVR, incluyendo el compilador (avr-gcc), depurador (avr-gdb) entre otros.

WinAvr corre bajo la misma plataforma de desarrollo AVR Studio de Atmel.

Algunos de los módulos que maneja WinAvr, entre otro son:

- bucles de retardo/espera

- manejo de eeprom

- funciones matemáticas

- interrupciones y señales

- tipos estándar para enteros.

Para trabajar con WinAVR se requiere la inclusión de las librerías que se necesiten. La principal de ellas y que siempre debe incluirse es la que define el nombre de los registros de entradas y salidas: “io.h”. La cual se encuentra dentro de la subcarpeta avr. Para incluirla en nuestro proyecto, basta con hacer una inclusión de archivos “.h” tradicional de C/C++:

```
#include<avr/io.h>
```

Algunas de la s funciones básicas más usadas son:

Dentro de io.h:

_BV(x) “bit Value”, esta función sirve para “poner” un 1 en alguno de los bits de un registro.”

```
loop_until_bit_is_set(REG,b)

loop_until_bit_is_set(PIND,1)

loop_until_bit_is_clear(REG,b)
```

Dentro de “interrupt.h”:

```
sei()
```

cli()

ISR(vector){código} “rutina de servicio de interrupción”.

Material y equipo utilizado

- Protoboard
- Dip Switch 8 entradas
- Cable y Pinzas de Corte
- 2 Push Buttons
- Resistencias de 100 Ohms
- Barra de Leds
- Microcontrolador ATMEGA8535
- Software AVR Studio

Que se realizará en la práctica

Se estudiará el código en C otorgado para esta práctica, se analizará su funcionamiento y se transcribió en lenguaje ASM para ser probado posteriormente y compararlo, teniendo esta comparación que salir idéntica.

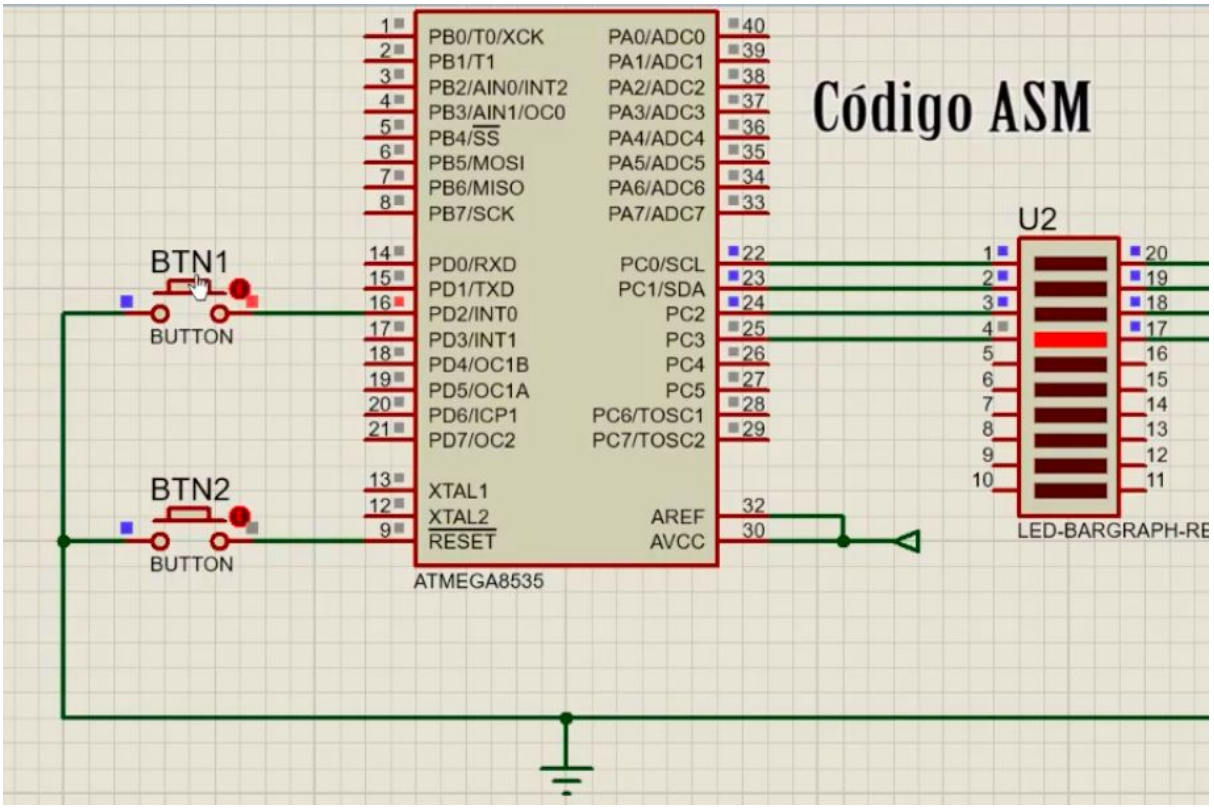
Desarrollo de la práctica

Se reescribió el código C correspondiente a la práctica a código ASM como se muestra a continuación:

```
.include"m8535def.inc"
.def aux = r16
.equ step1 = 8 ; 00001000
.equ step2 = 4 ; 00000100
.equ step3 = 2 ; 00000010
.equ step4 = 1 ; 00000001

rjmp config_io        ; El programa brinca a esta instrucción cuando ocurre
reset
rjmp secuencia2              ; Registro $001 (INT0) tiene rutina 'secuencia2
'


config_io:
       ldi aux, $0F
       out DDRC, aux  ; Activamos los puertos 'A' para la salida (solo 4
puertos)
       ldi aux, $04
       out PORTD,aux  ; Activamos los puertos 'D' para la ENTRADA
       ldi aux,$02
       out MCUCR,aux  ; Se establece INT0 para flanco de bajada
       ldi aux,$40
       out GICR,aux   ; Se habilitan INT0
       sei

main:
       rcall secuencia1      ; Ciclo infinito
       rjmp main
```

```
secuencia1:                          ; Secuencia uno
       ldi aux,step1          ; Valor de 1
       out PORTC,aux          ; Muestra a la salida
       rcall retardo          ; Retardo

       ldi aux,step2          ; Valor de 2
       out PORTC,aux          ; Muestra a la salida
       rcall retardo          ; Retardo

       ldi aux,step3          ; Valor de 4
       out PORTC,aux          ; Muestra a la salida
       rcall retardo          ; Retardo

       ldi aux,step4          ; Valor de 8
       out PORTC,aux          ; Muestra a la salida
       rcall retardo          ; Retardo
       ret

secuencia2:                          ; Secuencia 2
       ldi aux,step4          ; Valor de 8
       out PORTC,aux          ; Muestra a la salida
       rcall retardo          ; Retardo

       ldi aux,step3          ; Valor de 4
       out PORTC,aux          ; Muestra a la salida
       rcall retardo          ; Retardo

       ldi aux,step2          ; Valor de 2
       out PORTC,aux          ; Muestra a la salida
       rcall retardo          ; Retardo

       ldi aux,step1          ; Valor de 1
       out PORTC,aux          ; Muestra a la salida
       rcall retardo          ; Retardo
       reti

retardo:
;    delay loop generator
;     250000 cycles:
; delaying 249999 cycles:
          ldi  R17, $A7
WGLOOP0:  ldi  R18, $02
WGLOOP1:  ldi  R19, $F8
WGLOOP2:  dec  R19
          brne WGLOOP2
          dec  R18
          brne WGLOOP1
          dec  R17
          brne WGLOOP0
; ------------------------------
; delaying 1 cycle:
          nop
; ==============================
       ret
```

Diagrama Eléctrico



Link de la simulación:

https://drive.google.com/file/d/1taT0dS9sOR4hyLgj-P_RjxmmnGrv0Qew/view?usp=sharing

Conclusiones

Esquivel Pérez Jonathan Alfredo:

Al realizar esta práctica pudimos notar las diferencias de utilizar código escrito en C como en un ASM. La dificultad bajó considerablemente al utilizar el primero debido a su amplio conocimiento y practicidad. Mientras que ambos muestran sus ventajas y desventajas, el uso de cada lenguaje dependerá de la magnitud del proyecto, complejidad o sencillez del mismo.

Hernández López Ángel Zait:

En esta práctica se pudo ver que no solamente se puede programar en C; notamos también, que existen métodos y funciones especiales para poder activar ciertas cosas, como en esta práctica, que fue la activación de una interrupción externa. Una vez aprendido el lenguaje ensamblador y tener fundamentos de lenguaje C, es muy fácil pasar un código de uno a otro. Una gran diferencia es el archivo de salida, que sirve para cargarlo al microcontrolador, ya que cuando se programa en C, el archivo es pesado a comparación al que se crea con lenguaje máquina.

Salgado Gallegos Jesús

En esta práctica se pudo analizar la diferencia en el código que se tiene entre un lenguaje C y un ASM, cada uno tiene sus ventajas y desventajas, como la portabilidad, estructura, abstracción, etc. Y cabe destacar que para un proyecto de microcontroladores, el saber cuál utilizar depende enteramente de las herramientas del programador y del proyecto a considerar, así como también el gusto y la mejor solución a desarrollar.

Sánchez Pizano Irving Daniel:

En esta práctica se pudo observar las diferentes maneras de desarrollar un proyecto para un microcontrolador, ya que en este caso el objetivo es hacer el mismo código pero en dos lenguajes diferentes, como lo es ensamblador y lenguaje C.

Aunque cada uno tiene sus pro y sus contras, pero al final todo dependerá del tamaño del proyecto y sobre todo del programador.