

Practica 1: Introducción a Microcontrolador & Ensamblador

Marco A. Morales Aguirre, Jose Guadalupe Romero
Edgar Granados, J. Carlos Urteaga Reyesvera
Instituto Tecnológico Autónomo de México

2017

1 Objetivo

- Programar un microcontrolador utilizando ensamblador.
- Utilizar entradas y salidas de un microcontrolador a través de ensamblador.

2 Problema

Se requiere poder utilizar operaciones de bits para cambiar y leer estados en los puertos del microcontrolador. La salida puede ser una cadena de caracteres en el monitor, se recomienda utilizar más de cuatro leds para ver cambios los cambios en el led.

3 Esbozo de la solución

Parte 1:

- Utilizando la función `asm volatile`, escribir código ensamblador para determinar si un número es par o no. Mostrar el resultado en la consola.
- Prender y apagar LEDES desde código ensamblador.
 - Utilizar los puertos DDR para asignar pines de lectura y escritura.
 - Utilizar las instrucciones `cbi` y `sbi`
- Imprimir los estados de los pines en la consola.
- Diseñar un arreglo de leds para representar un sucesión de 4 bits.

Parte 2:

- Hacer parpadear un LED utilizando código ensamblador utilizando ciclos
- Hacer parpadear un LED utilizando C a 1 Hz.
- Hacer parpadear 4 LEDES a las siguientes frecuencias: 1Hz, 2Hz, 4Hz, 8Hz.

PWM

- Realizar la siguiente prueba:

```

sbi 0x0e, 4
nop
nop
nop
nop
cbi 0x0e, 4
nop
nop
nop
nop

```

¿Qué sucede cuando se altera el número de instrucciones *nop* que hay entre *sbi* y *cbi*?

- Hacer *respirar* dos LEDs a diferentes *velocidades*
- Implementar una interrupción asociada a que un LED se prenda cuando se apriete un botón, mientras los otros LEDs siguen *respirando*.
- Modificar la función de interrupción a que el LED cambie de intensidad (entre 5 intensidades) cada vez que se apriete el botón.

4 Aspectos técnicos de la implementación

4.1 Ensamblador

El lenguaje ensamblador es un lenguaje de programación de bajo nivel que utiliza mnemónicos para representar las operaciones básicas de un microcontrolador y cualquier otro circuito integrado programable.

Aunque semejante, cada arquitectura de procesador tiene su propio lenguaje ensamblador normalmente definida por el fabricante del hardware. Debido a que en ella se definen los mnemónicos de los registros, operaciones y hasta las direcciones de memoria que son accesibles. No obstante comparten operaciones y algunas directivas similares entre ellos que facilitan el aprendizaje en general de la programación en ensamblador.

Actualmente su uso pasó de ser esencial en el desarrollo de programas para las computadoras, a ser una herramienta académica y especializada cuando se requiere utilizar los componentes de hardware directamente para hacer más eficiente la operación de un sistema.

- Mediante la sentencia `asm (/*código ensamblador*/) se pueden ejecutar sentencias de ensamblador en el código de Arduino. Recuerda el formato`
`asm(code : output operand list : input operand list : clobber list)`
- Una forma de sumar 1 es utilizando la operación `inc`, pero se puede hacer a través de constantes utilizando un registro auxiliar y las funciones `ldi` y `add`.
- Para evitar que el programa elimine valores asignados en variables de ensamblador se puede usar el comando `asm volatile (/*código ensamblador*/) para desactivar esta optimización del código C.`
- `sbi` y `cbi` son operaciones de un solo bit

```

#Prende
asm (
    "sbi %0, %1 \n": : "I" (_SFR_IO_ADDR(PORTC)), "I" (PORTC)
);
#Apaga
asm (
    "cbi %0, %1 \n": : "I" (_SFR_IO_ADDR(PORTC)), "I" (PORTC)
);

```

- `_SFR_IO_ADDR()` es una macro que permite mapear registros.
- Una forma de modificar un conjunto de bits es utilizando la operación `out`.
- Para cargar un conjunto de bits al registro se utiliza `in`.
- Interrupt Service Routine (ISR) también soporta código ensamblador, utiliza los registros TCCR y TCNT para realizar el contador.
- El tercer punto es de acuerdo a sus notas de clase, este punto se deja de tarea.

5 Referencias útiles

Arduino <https://www.arduino.cc/en/Reference/HomePage>.

Ensamblador en línea de arduino <https://web.stanford.edu/class/ee281/projects/aut2002/yingzong-mouse/media/GCCAVRInlAsmCB.pdf>.

Manipulación de puertos <https://www.arduino.cc/en/Reference/PortManipulation>

Referencia de Arduino sobre PWM <https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM> y <http://playground.arduino.cc/Main/TimerPWMCheatsheet> .

Operaciones binarias <http://playground.arduino.cc/Code/BitMath>.

GCC-AVR Inline Assembler Cookbook <https://web.stanford.edu/class/ee281/projects/aut2002/yingzong-mouse/media/GCCAVRInlAsmCB.pdf>

ATMEL ATmega Datasheet <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-Datasheet.pdf>

6 Mapeo de pines del ATmega2560-Arduino

Cada tipo de Arduino tiene su configuración, utiliza los puertos digitales para realizar las tareas. Tip: <https://www.arduino.cc/en/Hacking/PinMapping2560>

Pin Number	Pin Name	Mapped Pin Name
1	PG5 (OC0B)	Digital pin 4 (PWM)
2	PE0 (RXD0/PCINT8)	Digital pin 0 (RX0)
3	PE1 (TXD0)	Digital pin 1 (TX0)
4	PE2 (XCK0/AIN0)	
5	PE3 (OC3A/AIN1)	Digital pin 5 (PWM)
6	PE4 (OC3B/INT4)	Digital pin 2 (PWM)
7	PE5 (OC3C/INT5)	Digital pin 3 (PWM)
8	PE6 (T3/INT6)	
9	PE7 (CLK0/ICP3/INT7)	
10	VCC	VCC
11	GND	GND
12	PH0 (RXD2)	Digital pin 17 (RX2)
13	PH1 (TXD2)	Digital pin 16 (TX2)
14	PH2 (XCK2)	
15	PH3 (OC4A)	Digital pin 6 (PWM)
16	PH4 (OC4B)	Digital pin 7 (PWM)
17	PH5 (OC4C)	Digital pin 8 (PWM)
18	PH6 (OC2B)	Digital pin 9 (PWM)
19	PB0 (SS/PCINT0)	Digital pin 53 (SS)
20	PB1 (SCK/PCINT1)	Digital pin 52 (SCK)
21	PB2 (MOSI/PCINT2)	Digital pin 51 (MOSI)
22	PB3 (MISO/PCINT3)	Digital pin 50 (MISO)
23	PB4 (OC2A/PCINT4)	Digital pin 10 (PWM)
24	PB5 (OC1A/PCINT5)	Digital pin 11 (PWM)
25	PB6 (OC1B/PCINT6)	Digital pin 12 (PWM)
26	PB7 (OC0A/OC1C/PCINT7)	Digital pin 13 (PWM)
27	PH7 (T4)	
28	PG3 (TOSC2)	
29	PG4 (TOSC1)	
30	RESET	RESET
31	VCC	VCC

Pin Number	Pin Name	Mapped Pin Name
32	GND	GND
33	XTAL2	XTAL2
34	XTAL1	XTAL1
35	PL0 (ICP4)	Digital pin 49
36	PL1 (ICP5)	Digital pin 48
37	PL2 (T5)	Digital pin 47
38	PL3 (OC5A)	Digital pin 46 (PWM)
39	PL4 (OC5B)	Digital pin 45 (PWM)
40	PL5 (OC5C)	Digital pin 44 (PWM)
41	PL6	Digital pin 43
42	PL7	Digital pin 42
43	PD0 (SCL/INT0)	Digital pin 21 (SCL)
44	PD1 (SDA/INT1)	Digital pin 20 (SDA)
45	PD2 (RXDI/INT2)	Digital pin 19 (RX1)
46	PD3 (TXD1/INT3)	Digital pin 18 (TX1)
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin 38
51	PG0 (WR)	Digital pin 41
52	PG1 (RD)	Digital pin 40
53	PC0 (A8)	Digital pin 37
54	PC1 (A9)	Digital pin 36
55	PC2 (A10)	Digital pin 35
56	PC3 (A11)	Digital pin 34
57	PC4 (A12)	Digital pin 33
58	PC5 (A13)	Digital pin 32
59	PC6 (A14)	Digital pin 31
60	PC7 (A15)	Digital pin 30
61	VCC	VCC
62	GND	GND
63	PJ0 (RXD3/PCINT9)	Digital pin 15 (RX3)
64	PJ1 (TXD3/PCINT10)	Digital pin 14 (TX3)
65	PJ2 (XCK3/PCINT11)	
66	PJ3 (PCINT12)	
67	PJ4 (PCINT13)	
68	PJ5 (PCINT14)	
69	PJ6 (PCINT 15)	
70	PG2 (ALE)	Digital pin 39
71	PA7 (AD7)	Digital pin 29
72	PA6 (AD6)	Digital pin 28
73	PA5 (AD5)	Digital pin 27
74	PA4 (AD4)	Digital pin 26
75	PA3 (AD3)	Digital pin 25
76	PA2 (AD2)	Digital pin 24
77	PA1 (AD1)	Digital pin 23
78	PA0 (AD0)	Digital pin 22
79	PJ7	
80	VCC	VCC
81	GND	GND
82	PK7 (ADC15/PCINT23)	Analog pin 15

Pin Number	Pin Name	Mapped Pin Name
83	PK6 (ADC14/PCINT22)	Analog pin 14
84	PK5 (ADC13/PCINT21)	Analog pin 13
85	PK4 (ADC12/PCINT20)	Analog pin 12
86	PK3 (ADC11/PCINT19)	Analog pin 11
87	PK2 (ADC10/PCINT18)	Analog pin 10
88	PK1 (ADC9/PCINT17)	Analog pin 9
89	PK0 (ADC8/PCINT16)	Analog pin 8
90	PF7 (ADC7)	Analog pin 7
91	PF6 (ADC6)	Analog pin 6
92	PF5 (ADC5/TMS)	Analog pin 5
93	PF4 (ADC4/TMK)	Analog pin 4
94	PF3 (ADC3)	Analog pin 3
95	PF2 (ADC2)	Analog pin 2
96	PF1 (ADC1)	Analog pin 1
97	PF0 (ADC0)	Analog pin 0
98	AREF	Analog Reference
99	GND	GND
100	AVCC	VCC