INSTITUTO POLITÉCNICO NACIONAL

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

Arquitectura de Computadoras

PRÁCTICA A

19 de octubre de 2017

Alvarado Balbuena Jorge Anselmo
Avalos Vizuett Julio Cesar
Rocha Díaz Brandon

Objetivos

- Complementar el análisis teórico visto en clase con herramientas de programación.
- Comprender el lenguaje ensamblador a través del manejo ATMEGA328P.
- Aprender a configurar un AVR en lenguaje ensamblador.

Introducción

Los AVR son una familia de microcontroladores RISC de Atmel. La arquitectura de los AVR fue concebida por dos estudiantes en el Norwegian Institute of Technology, y posteriormente desarrollada en Atmel Norway, la empresa subsidiaria de Atmel, fundada por los dos arquitectos del chip.

El AVR fue diseñado desde un comienzo para la ejecución eficiente de código C compilado. Por lo tanto, algunas instrucciones tales como 'suma inmediata' ('add immediate' en inglés) faltan, ya que la instrucción 'resta inmediata' ('substract immediate' en inglés) con el complemento dos puede ser usada como alternativa.

El set de instrucciones de los AVR es más regular que el de la mayor al de los microcontroladores de 8-bit (por ejemplo, los PIC). Sin embargo, no es completamente ortogonal: Los registros punteros X, Y y Z tienen capacidades de direccionamiento diferentes entre sí. Los registros 0 al 15 tienen diferentes capacidades de direccionamiento que los registros 16 al 31. Los registros de I/O 0 al 31 tienen distintas características que las posiciones 32 al 63.

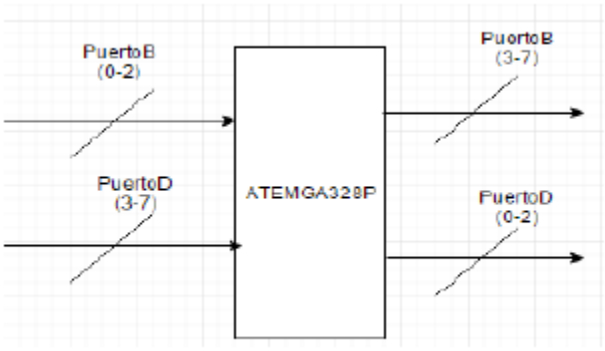
Desarrollo

Ejercicio 1

Se configura los primeros 3 pines del puerto B como entrada y los restantes del dicho puerto como salida. Para el puerto D se configura los 3 primeros pines como salida y los restantes como salida. La información que se obtiene de los pines de entrada se pasara a los pines de salida.

Código

```
.org 0x000 jmp config
config: ldi r16, 0b11110000
        out ddrb, r16
        ldi r16, 0b00001111
        out ddrd, r16
loop:   in r17, pinb
        out portd, r17
        in r17, pind
        out portb, r17
        jmp loop
```



Ejercicio 2

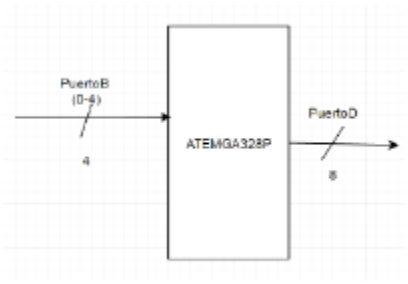
Se desarrollará un sumador entre los nibbles bajos de los puertos B y D, los pines que se sumaran y se mostraran el resultado en los primeros 5 pines del puerto C. En la parte alta de nibbles del puerto B y D mostrar la información de su nibble bajo.

Código

```
.org 0x000 rjmp config
config: ldi r20, 0xf0
        out ddrb, r20
        out ddrd, r20
        ser r20
        out ddrc, r20
loop:    in r16, pinb
        mov r17, r16
        swap r16
        out portb, r16
        in r16, pind
        mov r18, r16
        swap r16
        out portd, r16
        add r17, r18
        out portc, r17
        rjmp loop
```

Ejercicio 3

Se desarrollará 4 resultados diferentes que se mostraran en el puerto D, donde dependiendo que pin este encendido de los primeros 4 pines del puerto B se mostrara un resultado asignado.




```
.org 0x000 jmp config
config:         ldi r16, 0b00001111
                out ddrb, r16
                ldi r16, 0b11111111
                out ddrd, r16
                ldi r18, 0b00000001
                ldi r19, 0b00000010
                ldi r20, 0b00000100
                ldi r21, 0b00001000
                ldi r22, 0b00000000
loop:         in r16, pinb
                cp r16, r22
                breq cero
                cp r16, r18
                breq uno
                cp r16, r19
                breq dos
                cp r16, r20
                jmp tres
                cp r16, r21
                jmp cuatro
cero:          ldi r17, 0b00000000
                out portd, r17
                jmp loop
uno:          ldi r17, 0b11001100
                out portd, r17
                jmp loop
dos:          ldi r17, 0b00110011
                out portd, r17
                jmp loop
tres:          ldi r17, 0b11110000
                out portd, r17
                jmp loop
cuatro:         ldi r17, 0b00001111
                out portd, r17
                jmp loop
```

Conclusiones

Alvarado Balbuena Jorge Anselmo

En esta práctica comenzamos a conocer los principios de programación en ensamblador. Utilizamos la asignación de valores a los registros del AVR. Una pequeña vista a las rutinas y comparadores entre valores. También nos familiarizamos con el entorno de programación, en este caso Atmel Studio y con el programador Usb Asp que nos permite cargar el código generado por nuestro Ide al AVR.

Avalos Vizuett Julio Cesar

Por medio del lenguaje de programación ensamblador nos permite tener otro medio conocido para programar microcontroladores, en este caso al ser un AVR del tipo ATMEGA328P, tiene ciertas características y ciertos comportamientos, que tenemos tener en cuenta al intentar programarlo, ya que cada AVR comercial que existe actualmente posee diferentes puertos, aunque los más usados son puerto A, B, C, D, pero existen unos donde poseen más o menos, no podemos exigir más a un AVR de lo que puede.

Rocha Díaz Brandon

El Atmega328 AVR 8-bit es un Circuito integrado de alto rendimiento que está basado un microcontrolador RISC, es el más utilizado en múltiples proyectos y sistemas autónomos donde un micro controlador simple, de bajo consumo, bajo costo es requerido. El saber programarlo en lenguaje ensamblador de acuerdo a las necesidades es esencial y se logró realizar considerando el AVR utilizado con base en sus especificaciones encontradas en su respectiva datasheet.