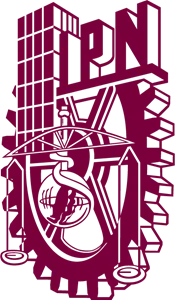
****

**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**

Practica 1.

Profesor:

Perez Perez Jose Juan

Integrantes

Jiménez Velázquez José Bryan Omar

Grupo: 7CM4 14/10/2025

Introducción

Programación en Ensamblador del Microcontrolador ATmega8535

La práctica desarrollada con el microcontrolador ATmega8535 en lenguaje ensamblador constituye un punto de convergencia entre el software y el hardware, al permitir una optimización precisa tanto del tiempo de ejecución como del uso de la memoria. Este enfoque de bajo nivel proporciona un control directo sobre los registros y periféricos del sistema, favoreciendo la comprensión detallada de su arquitectura interna y del funcionamiento de los puertos de entrada y salida.

1. Control de Puertos de Entrada/Salida (I/O)

El ATmega8535, perteneciente a la familia AVR de 8 bits, utiliza tres registros principales de control para cada puerto de E/S: DDRx, PORTx y PINx. La manipulación directa de estos registros constituye la base de la configuración del hardware a nivel de bit.

* DDRx (Data Direction Register):  
  Define la dirección de flujo de datos de cada pin del puerto *x*.
  + Escribir un ‘1’ en un bit configura el pin correspondiente como salida.
  + Escribir un ‘0’ lo configura como entrada.
  + *Ejemplo:* OUT DDRB, $FF configura todos los pines del Puerto B como salidas.
* PORTx (Port Data Register):   
  Cumple una doble función, dependiendo de la configuración del pin:
  + Si el pin es salida, escribir en PORTx establece su nivel lógico (alto o bajo).
  + Si el pin es entrada, escribir un ‘1’ en PORTx activa la resistencia de *pull-up* interna del microcontrolador.
* PINx (Port Input Pins Register): Es un registro solo de lectura, utilizado para conocer el estado lógico actual (voltaje) de los pines del puerto *x*. Permite la lectura de datos provenientes de dispositivos externos, como interruptores, botones o sensores.

2. Control de Flujo: Comparación y Salto Condicional

En lenguaje ensamblador, las estructuras de control de alto nivel (*if*, *for*, *while*) se implementan mediante comparaciones y saltos condicionales.

* Instrucción de Comparación (CP): Compara el contenido de dos registros sin alterar su valor. La instrucción CP Rd, Rr realiza internamente la operación $Rd - Rr$, afectando únicamente las banderas del Registro de Estado (SREG).
* Registro de Estado (SREG):  
  Almacena los *flags* que reflejan el resultado de la última operación aritmético-lógica.
  + El flag Z (Zero) se activa cuando el resultado de la comparación es cero, indicando que ambos operandos son iguales.
* Instrucciones de Salto Condicional:  
  Evalúan el estado de las banderas del SREG para modificar el flujo de ejecución:
  + BREQ (*Branch if Equal*): salta si el flag Z está activado (los valores comparados son iguales).
  + BRLO (*Branch if Lower*): salta si el resultado es menor en una comparación sin signo.
  + RJMP (*Relative Jump*): realiza un salto incondicional a una etiqueta dentro del mismo segmento del programa, siendo útil para bucles o repeticiones controladas.

3. Conversión de Datos Binarios a ASCII

Dado que los microcontroladores manipulan datos en formato binario o hexadecimal, y los dispositivos de salida (como pantallas o terminales) utilizan caracteres legibles, es necesario realizar una conversión a formato ASCII (American Standard Code for Information Interchange).

* Representación ASCII:  
  Cada carácter alfanumérico posee un código numérico de 7 u 8 bits.
  + Los caracteres numéricos del ‘0’ al ‘9’ se encuentran secuencialmente entre 0x30 (48) y 0x39 (57) en código ASCII.
* Conversión de Dígitos:  
  Para transformar un valor numérico (0–9) en su equivalente ASCII, basta con sumarle la constante 0x30.
  + *Ejemplo:* Si el registro R16 contiene el valor 5 (0x05), al ejecutar ADIW R16, $30 el resultado será 0x35, correspondiente al carácter ‘5’.
* Conversión de Números Multidígito: En el caso de valores mayores a 9, se requiere una división sucesiva entre 10 para extraer unidades, decenas y centenas. Posteriormente, cada dígito se convierte individualmente a su representación ASCII. Este proceso se implementa mediante bucles y restas repetidas, debido a que el conjunto de instrucciones AVR carece de una instrucción de división directa.

PLANTEAMIENTO DEL EJERCICIO A REALIZAR

Inciso A:

Desarrollar un programa en ensamblador que lea los valores de los puertos B y D, los compare, y muestre en los puertos A y C los valores según el resultado de la comparación:

* Si los valores son iguales, se muestran ambos en los puertos A y C.
* Si el valor del puerto B es mayor, se envía a PORTA el valor de B y a PORTC el de D.
* Si el valor del puerto B es menor, se invierte el orden de salida (D en PORTC y B en PORTA).

Inciso B:

El segundo ejercicio consiste en leer un valor numérico desde el Puerto B y convertirlo a su equivalente ASCII hexadecimal, mostrando el resultado en el Puerto A.  
Esto permite visualizar, por ejemplo, un número binario como el carácter correspondiente (‘0’–‘9’ o ‘A’–‘F’).

DESARROLLO

Inciso A

1. Se incluye la librería del microcontrolador con la directiva .include "m8535def.inc".
2. Se configuran los puertos A y C como salidas y los puertos B y D como entradas.
3. Se realiza la lectura de los puertos B y D.
4. Se compara el contenido de ambos registros (CP r16, r17).
5. Según el resultado:
   * Si son iguales (BREQ): se muestran ambos valores.
   * Si B < D (BRLO): se intercambian los valores en la salida.
   * Si B > D: se muestran directamente.
6. Se repite continuamente con un bucle RJMP ciclo.

Inciso B

1. Se inicializa el puerto B como entrada y el puerto A como salida.
2. Se lee el valor desde PINB.
3. Se convierte el valor leído a su código ASCII sumando el valor base $30 (correspondiente a ‘0’).
4. Si el resultado excede $3A, se suma un ajuste de 7 para obtener los caracteres ‘A’–‘F’.
5. Se muestra el valor convertido en PORTA.
6. El proceso se repite continuamente dentro del bucle main.

MATERIALES Y EQUIPO USADO

* Microcontrolador ATmega8535 o kit de desarrollo compatible (Ej. ATmega Trainer Board o STK500).
* Fuente de alimentación de 5V DC.
* Computadora con software de desarrollo AVR Studio / Atmel Studio.
* Cables de conexión (jumper o protoboard).
* Display LED o conjunto de LEDs conectados al puerto A para observar la salida.
* Interruptores (DIP-switches) conectados a los puertos B y D como entradas.
* Protoboard para el armado del circuito.
* Cables USB/Serial para programación y comunicación.

SOLUCIÓN DEL PROBLEMA

Inciso A:

El problema de comparación se resolvió utilizando las instrucciones CP, BREQ y BRLO, que permiten comparar los contenidos de dos registros y realizar saltos condicionales según el resultado (igual, menor o mayor).  
De esta manera se logró mostrar en los puertos A y C los valores correspondientes dependiendo de la relación entre las entradas B y D.

A screen shot of a computer

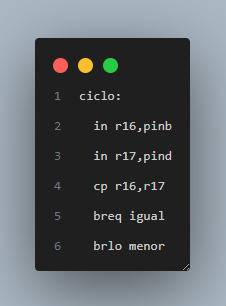
AI-generated content may be incorrect.

Incluye el fichero con las definiciones del microcontrolador (nombres de registros, constantes y direcciones para el ATmega8535). Permite usar etiquetas como DDRA, PORTB, PINB, etc.

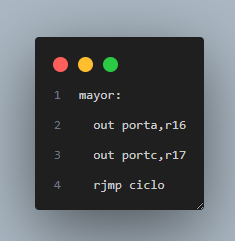
A screenshot of a computer screen

AI-generated content may be incorrect.

* SER pone todos los bits del registro en 1. Equivale a ldi r16, 0xFF.
* Resultado: r16 = 0xFF (1111 1111b). Se usa como valor “máscara” o para escribir 1s a registros PORT/DDR.
* Escribe el contenido de r16 (0xFF) en el registro DDRA.
* DDRA = 0xFF -> todos los pines del puerto A se configuran como salidas.
* Escribe 0xFF en PORTB. Dado que no se configuró DDRB a salida (por defecto es entrada), escribir 1 en PORTB activa las resistencias pull-up internas de PORTB.
* Resumen: configura las entradas de B con pull-ups.
* Igual que la anterior, pero para PORTD. Activa las resistencias pull-up internas de PORTD (suponiendo DDRD = 0 por defecto).



* Etiqueta que marca el inicio del bucle. Desde aquí se repite el proceso indefinidamente.
* Lee el registro de entrada PINB (estado actual de los pines físicos del puerto B) y almacena el valor en r16.
* r16 contiene ahora el valor leído desde las entradas (por ejemplo, el estado de interruptores conectados a PBx).
* Lee PIND (estado de los pines del puerto D) y lo guarda en r17.
* Compara r16 con r17 haciendo la operación interna r16 - r17 y actualiza las banderas del estado (Z, C, N, V, S). No cambia r16 ni r17.
* Las banderas resultantes sirven para las instrucciones de salto condicional siguientes.
* BREQ salta a la etiqueta igual si la bandera Z está a 1 → significa que r16 == r17 (los dos valores son iguales).
* BRLO salta a menor si la comparación indica que r16 < r17 (comparación sin signo); en AVR BRLO equivale a salto si C = 1 tras la resta.
* Si r16 es menor que r17, se va al caso menor.



* Etiqueta mayor: aquí se ejecuta cuando r16 > r17 (caso por caída).
* out PORTA, r16 → envía el valor de r16 al puerto A (salida).
* out PORTC, r17 → envía el valor de r17 al puerto C (salida).
* rjmp ciclo → salto incondicional a ciclo (vuelve al inicio y repite).

A black square with white text and green and red circles

AI-generated content may be incorrect.

* Etiqueta igual: se ejecuta si r16 == r17.
* Hace exactamente lo mismo que mayor en este código: pone r16 en PORTA y r17 en PORTC, luego vuelve a ciclo.

A screenshot of a computer

AI-generated content may be incorrect.

* Etiqueta menor: se ejecuta si r16 < r17.
* Primero out PORTC, r17 y después out PORTA, r16, luego vuelve al inicio.

Inciso B:

El problema de conversión numérica se solucionó utilizando aritmética simple en ensamblador, sumando la constante $30 al valor leído para transformarlo en su representación ASCII. Para los valores mayores que 9, se aplicó un desplazamiento adicional de 7 para cubrir los caracteres ‘A’–‘F’.

A black rectangular with white text and colorful dots

AI-generated content may be incorrect.

Incluye el fichero con las definiciones del ATmega8535 (etiquetas para DDRA, PORTA, PINB, etc.). Permite usar nombres legibles en vez de direcciones numéricas.

A black rectangular with white text

AI-generated content may be incorrect.

DIAGRMA DEL CIRCIUTO

CAPTURA YO FOTOS DEL CIRCUITO

CONCLUSIONES INDIVIDIALES

Bibliografía Recomendada (Formato IEEE)

Para profundizar en la programación en ensamblador del ATmega8535 y comprender la arquitectura AVR, se recomienda consultar las siguientes fuentes:

Manuales y Fichas Técnicas

* [1] Atmel Corporation, *ATmega8535 Complete Datasheet*, San Jose, CA, USA: Atmel Corporation, 2005. [En línea]. Disponible en el sitio web de Microchip Technology Inc.
* [2] Atmel Corporation, *AVR Instruction Set Manual*, San Jose, CA, USA: Atmel Corporation, 2013. [En línea]. Disponible en el sitio web de Microchip Technology Inc.
* Libros de Texto
* [3] H. M. Paz, *Pazuino con el Microcontrolador AVR Mega 8535*, vol. 1. Createspace Independent Publishing Platform, 2016.
* [4] E. Paredes Martínez, *Curso Práctico para Programación de AVR*, Barcelona, España: Marcombo, 2024.
* [5] M. Predko, *Programming and Customizing the AVR Microcontroller*, New York, NY, USA: McGraw-Hill Education, 2002.
* [6] V. H. Orozco, *Microcontroladores PIC y AVR: Teoría y Práctica en Lenguaje Ensamblador y C*, México: Alfaomega Grupo Editor, 2010.