**Universidad Nacional Autónoma de México**

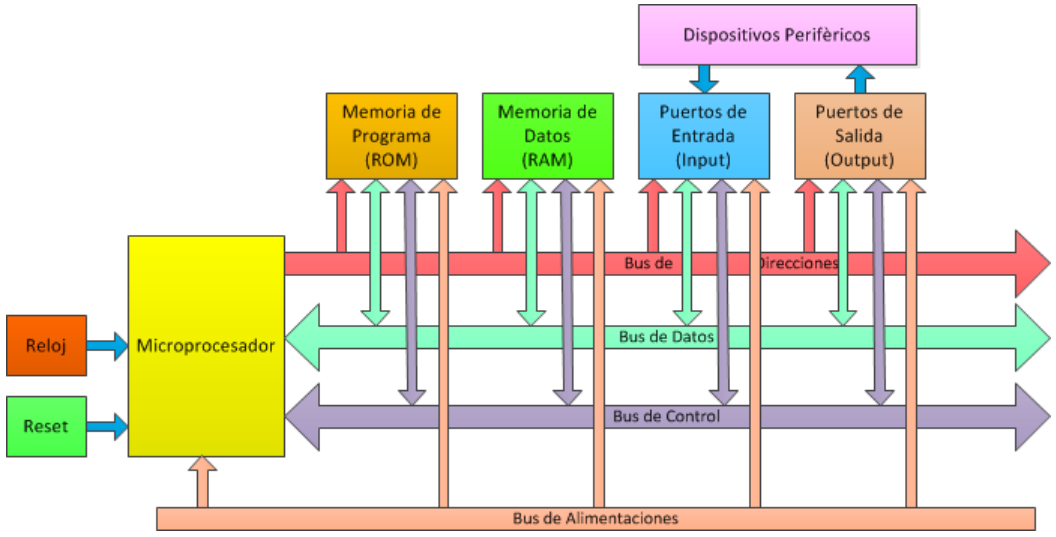
**Facultad de Estudios Superiores Cuautitlán**

**Departamento: Ingeniería**

**Sección: Electrónica**

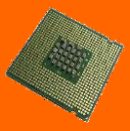
# Manual del Laboratorio de Microprocesadores

**Clave Asignatura: 0586**
**Clave Carrera: 130**

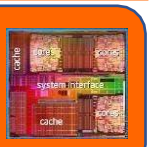



**Autor:**
**M.C Jorge Buendia Gomez**
**M.C Nidia Mendoza Andrade**
**M.C Jorge Ricardo Gersenowies Rosas**

**Fecha de Elaboración: 2000**
**Fecha de modificación: Enero 2025**



# Laboratorio de Microprocesadores
# Índice



# Laboratorio de Microprocesadores
# Prólogo


**Objetivos generales de la asignatura**

- Comprender la estructura y funcionamiento de los microprocesadores para aplicarlos en la solución de problemas de ingeniería

**Objetivos del curso experimental**

- Implementar un sistema digital que incluya un microprocesador como elemento central.
- Identificar la función que realiza cada uno de los elementos que conforman el esquema de Von Neumann: reloj, microprocesador, memoria ROM, memoria RAM, puertos de entrada, puertos de salida y dispositivos periféricos.
- Conocer la señalización característica que se presenta en los sistemas digitales con microprocesador e identificar de forma adecuada la forma en que interactúan los circuitos de apoyo que se emplean para la creación de una computadora.
- Diseñar, programar e implementar algoritmos que le permitan al sistema digital con microprocesador interactuar con los dispositivos periféricos y probar la funcionalidad de todas y cada una de las partes que conforman al esquema de Von Neumann.

**Introducción**

Actualmente, los sistemas digitales que integran microprocesadores se han expandido a la mayoría de los sistemas inteligentes que se emplean para controlar maquinaria industrial, sistemas de cómputo, teléfonos celulares, televisiones, electrodomésticos y en general todo aquel sistema que requiera capacidad de procesamiento automatizada.

Este laboratorio proporciona al alumno la teoría y los métodos para diseñar e implementar sistemas electrónicos avanzados que incluyan microprocesadores y además los dispositivos periféricos necesarios para realizar las interfaces hombre máquina que permitan controlar a este tipo de sistemas, los cuales están orientados a mejorar la forma de vida de las personas y liberarlas de actividades repetitivas.

Uno de los objetivos primordiales del Laboratorio de Microprocesadores es la implementación de un sistema de cómputo que emplee un microprocesador Z80 como elemento central, a este sistema digital se le conoce como “sistema mínimo”, el cual consta de los elementos más significativos para la construcción de una computadora.

La implementación se llevará a cabo, armando y probando las diferentes etapas que forman al sistema en cada una de las sesiones del laboratorio avanzando de forma paulatina hasta construir el proyecto en su totalidad.

El sistema mínimo por implementar se basa en el esquema de Von Neumann, el cual contiene una serie de elementos que permiten la ejecución de programas en lenguaje ensamblador y la interacción con dispositivos periféricos y con el usuario, la figura *P.1* nos muestra el esquema general de la arquitectura Von Neumann.

Esta arquitectura consta de un sistema de reloj, un microprocesador, la memoria ROM de programa, la memoria RWM (RAM) de datos, los puertos de entrada (Input), los puertos de salida (Output) y los dispositivos periféricos.

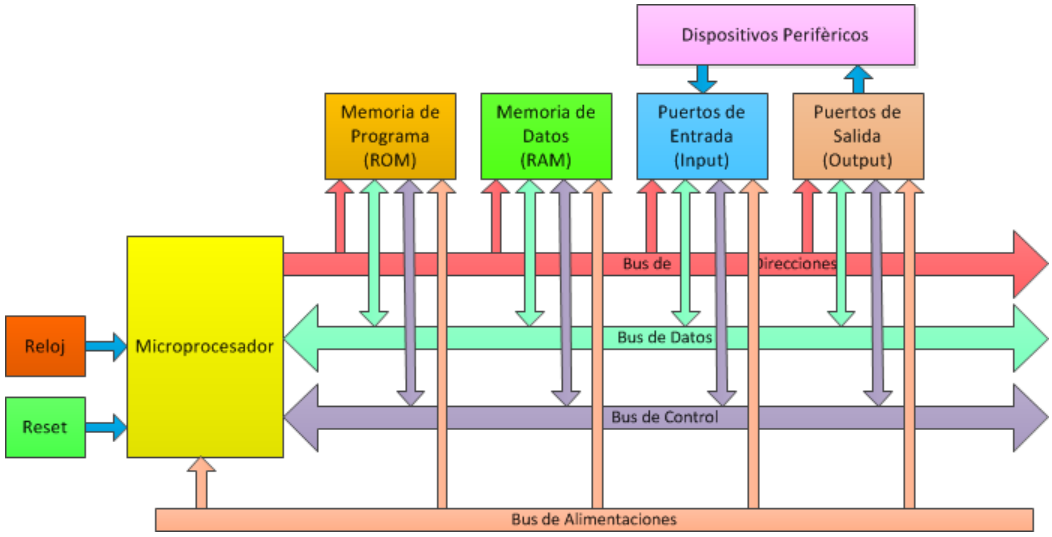



Figura P.1

En cada una de las prácticas de este laboratorio se implementará una de las partes que conforman al esquema de Von Neumann para que al final del curso se tenga un sistema de cómputo completo y funcional para la ejecución de programas en lenguaje ensamblador y para la conexión de diferentes dispositivos periféricos.

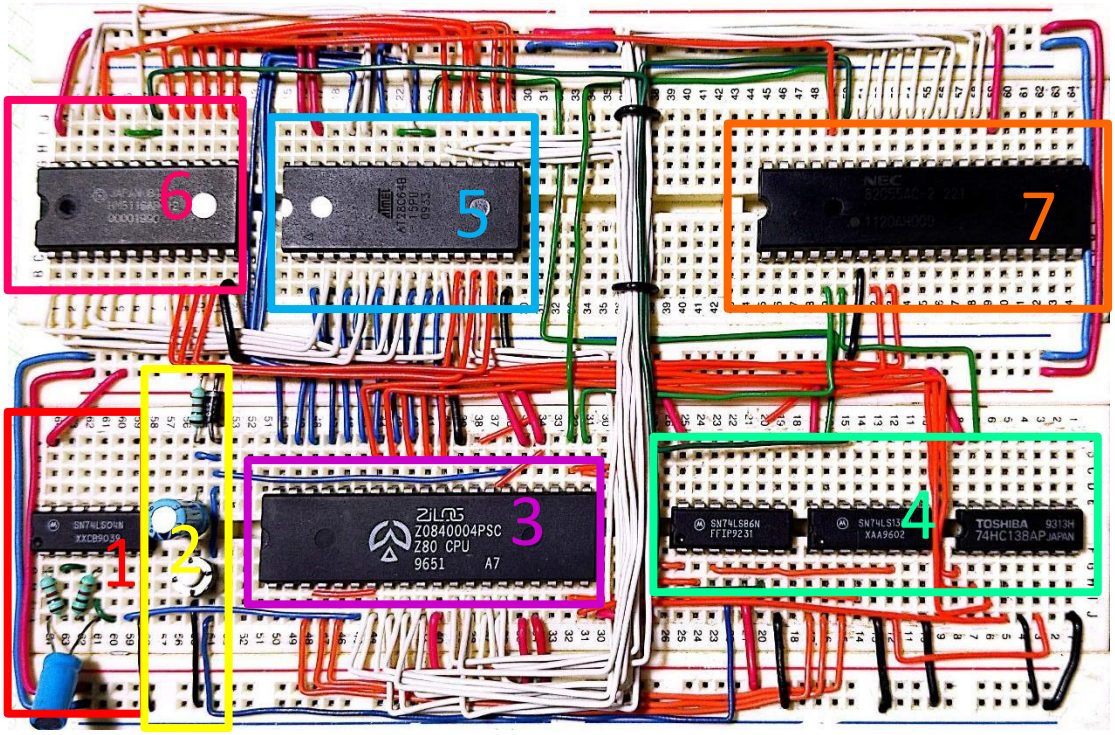



*Figura P.2 Sistema mínimo con uP Z80*

El sistema armado completo se muestra en la figura *P.2* en el cual se pueden identificar las siguientes partes:

| | # | Elemento del esquema de Von Neumann |
|---|---|---|
| | 1 | Circuito de reloj |
| | 2 | Circuito de reset |
| | 3 | Microprocesador Z80 |
| | 4 | Lógica de control y decodificación |
| | 5 | Memoria de programa (EEPROM) |
| | 6 | Memoria de datos (SRAM) |
| | 7 | Puertos de entrada / salida |

*Tabla P.1*

Cada uno de estos elementos será armado y probado para asegurar que el sistema funcione de forma adecuada.

Debido a que el objetivo es implementar el sistema completo, entonces será necesario conservar la parte del circuito que se haya armado en prácticas anteriores para que poco a poco se complete el sistema mínimo y por lo tanto el alumno deberá contar con al menos 2 tabletas de conexiones (protoboard) para armar el sistema como se muestra en la figura *P.2*.

En la implementación de este sistema deben realizarse una serie de consideraciones que de llevarse a cabo traen como consecuencia un circuito sin errores de conexión y falsos contactos facilitando las pruebas y garantizando un funcionamiento correcto.

Una de las consideraciones tiene que ver con la forma del alambrado, como se puede observar en la figura P.2, los alambres de conexión deben ser lo más corto posible entre 2 puntos del sistema y no deben elevarse sobre la tableta ni sobre los circuitos integrados puesto que esta acción provoca la interferencia por ruido.

Los circuitos deben quedar libres en su parte superior para que puedan ser extraídos en caso de un desperfecto en su funcionamiento o para poder programar a la EEPROM con los diferentes programas que se diseñarán para probar el sistema.

El grosor de los cables debe ser de calibre 24 para que se realice un buen contacto, el cable empleado en las redes de computadoras STP categoría 5 no es recomendable puesto que es muy delgado y no produce un buen contacto en la tableta de conexiones.

Para realizar la conexión, los cables deben insertarse en los orificios de la tableta de conexiones considerando que la punta descubierta del cable debe ser igual al alto de la tableta y no demasiado corta ni excesivamente larga, tal y como se muestra en la figura P.3.

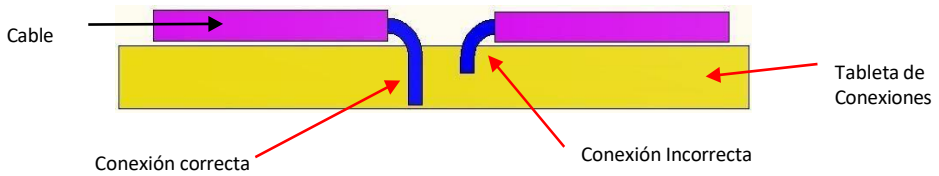



*Figura P.3*

Además, para la elaboración del sistema se deberá considerar que algunos de los circuitos empleados son de tecnología CMOS y NMOS, por lo tanto, son sensibles a descargas de voltaje por lo que hay que tener las

precauciones necesarias con respecto a las descargas electrostáticas para no causarles daño. Para realizar la manipulación de los circuitos el alumno deberá descargarse sobre algún objeto metálico y tomar los circuitos por los extremos y no por las terminales. En ambientes industriales esta operación se realiza con pulseras y tapetes de descarga antiestática.

Para la realización de este laboratorio se utilizarán herramientas adicionales a las consideradas en los laboratorios tradicionales de electrónica, tales como computadoras, software de ensamblado y ligado de programas en lenguaje ensamblador, simuladores de microprocesadores, software para programar memorias EEPROM además de la creación de una punta lógica de prueba para la comprobación del tercer estado presentado por los circuitos del proyecto.

El alumno deberá estar familiarizado con estas herramientas para poder realizar los procesos necesarios para las actividades previas y también los archivos ensamblados que se programarán en la memoria del sistema.

**Los criterios de evaluación para el laboratorio son los siguientes:**

| | | |
|---|---|---|
| C1 | Actividades previas a la práctica | 20% |
| C2 | Habilidad en el armado y funcionalidad de los sistemas | 30% |
| C3 | Habilidad para el manejo de del equipo e interpretación correcta de las lecturas | 20% |
| C4 | Reporte entregado con todos los puntos indicados | 30% |

**También será necesario incluir en cada práctica, una portada (obligatoria), que debe incluir los puntos indicados a continuación, en formato libre.**

U. N. A. M.
F. E. S. C.

Laboratorio de: _____
Grupo: _____ No. de Práctica: _____
Nombre de Práctica: _____
Profesor: _____
Alumno: _____
Fecha de realización: _____ Fecha de entrega: _____
Semestre: _____

*Figura P.4*

**Instrucciones para la elaboración del reporte**

Para la presentación del reporte se deberá cumplir con los requisitos indicados en cada una de las prácticas, incluyendo:

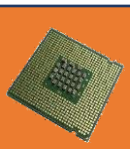
- Portada
- Introducción.
- Procedimiento experimental
- Circuito
- Tablas de datos
- Gráficas

- Comentarios
- Observaciones
- Esquemas
- Diagramas
- Cuestionario

y en general todos los elementos solicitados dentro del desarrollo de la práctica.

**Bibliografía**

1. Brey, Barry B (2001), *Los microprocesadores Intel: arquitectura, programación e interfaz de los procesadores 8086/8088, 80186/80188, 80286, 80386, 80486 Peintium, Peintium Pro y Pentuim II*, México, D.F.: Pearson Educación.
2. E. Mandado, E. Tassis (1996). *Diseño de sistemas digitales con microprocesadores*. México, D.F.: Alfaomega.
3. Medina E. A., Marrero M.M. (2008) *Diseño de sistemas electrónicos digitales con microprocesadores*. España: Universidad de las Palmas de Gran Canaria.
4. Tokheim, Roger L. (1991). *Fundamentos de los microprocesadores*. Madrid; México: McGraw-Hill.
5. José Ma. Angulo Usategui. (1990). *Microprocesadores 8086, 80286 y 80386*. Madrid: Paraninfo.
6. Khambata, A.J. (1987). *Microprocesadores-microcomputadores: Arquitectura, software y sistemas*. Barcelona; México: Gustavo Gili.
7. Ciarcia Steve. (1984). *Construya una microcomputadora basada en el Z80: Guía de diseño y funcionamiento*. Madrid, México: Byte/Mcgraw-Hill.
8. Crisp, John. (2004). *Introduction to microprocessors and microcontrollers*. (2nd). Amsterdam ; Boston : Elsevier/Newnes.



# Laboratorio de Microprocesadores

## Reglamento interno de los laboratorios de Comunicaciones, Control, Sistemas Analógicos y Sistemas Digitales.


El presente reglamento de la sección electrónica tiene por objetivo establecer los lineamientos para el uso y seguridad de laboratorios, condiciones de operación y evaluación, que deberán de conocer y aplicar, estudiantes y profesores en sus cuatro áreas: comunicaciones, control, sistemas analógicos y sistemas digitales.

1. Queda estrictamente prohibido, al interior de los laboratorios

   a. Correr, jugar, gritar o hacer cualquier otra clase de desorden.
   b. Dejar basura en las mesas de trabajo y/o pisos.
   c. Fumar, consumir alimentos y/o bebidas.
   d. Realizar o responder llamadas telefónicas y/o el envío de cualquier tipo de mensajería.
   e. La presencia de personas ajenas en los horarios de laboratorio.
   f. Dejar los bancos en desorden y/o sobre las mesas.
   g. Mover equipos o quitar accesorios de una mesa de trabajo.
   h. Usar o manipular el equipo sin la autorización del profesor.
   i. Rayar y/o sentarse en las mesas del laboratorio.
   j. Energizar algún circuito sin antes verificar que las conexiones sean las correctas (polaridad de las fuentes de voltaje, multímetros, etc.).
   k. Hacer cambios en las conexiones o desconectar el equipo estando energizado.
   l. Hacer trabajos pesados (taladrar, martillar, etc.) en las mesas de trabajo.
   m. Instalar software y/o guardar información en los equipos de cómputo de los laboratorios.
   n. El uso de cualquier aparato o dispositivo electrónico ajeno al propósito para la realización de la práctica.
   o. Impartir sesiones teóricas, su uso es exclusivo para las sesiones de laboratorio.

2. Es responsabilidad del profesor y de los estudiantes revisar las condiciones del equipo e instalaciones del laboratorio al inicio de cada práctica (encendido, dañado, sin funcionar, maltratado, etc.). El profesor deberá generar el reporte de fallas de equipo o de cualquier anomalía y entregarlo al responsable de laboratorio o al jefe de sección.

3. Los profesores deberán de cumplir con las actividades y tiempos indicados en el “cronograma de actividades de laboratorio”.

4. Es requisito indispensable para la realización de las prácticas que el estudiante:

   a. Descargue el manual completo y actualizado al semestre en curso, el cual podrá obtener en: http://olimpia.cuautitlan2.unam.mx/pagina_ingenieria

   b. Presente su circuito armado en la tableta de conexiones para poder realizar la práctica, de no ser así, tendrá una evaluación de cero en la sesión correspondiente.

   c. Realice las actividades previas indicadas en el manual de prácticas para cada sesión de laboratorio.

5. La evaluación de cada sesión debe realizarse con base en los criterios de evaluación incluidos en los manuales de prácticas de laboratorio y no podrán ser modificados. En caso contrario, el estudiante deberá reportarlo al jefe de sección.

6. La evaluación final del estudiante en los laboratorios será con base en lo siguiente:

   **A - (Aprobado); Cuando el promedio total de todas las prácticas de laboratorio sea mayor o igual a 6 siempre y cuando tengan el 90% de asistencia, el 80% de prácticas acreditadas con base en los criterios de evaluación.**

   **NA -(No Aprobado); No cumplió con los requisitos mínimos establecidos en el punto anterior.**

   **NP - (No Presentó); Cuando no asistió a ninguna sesión de laboratorio.**

7. Profesores que requieran hacer uso de las instalaciones de laboratorio para realizar trabajos o proyectos, es requisito indispensable que notifiquen por escrito al jefe de sección. Siempre y cuando no interfiera con los horarios de los laboratorios.

8. Estudiantes que requieran realizar trabajos o proyectos en las instalaciones de los laboratorios, es requisito indispensable que esté presente el profesor responsable del trabajo o proyecto. En caso contrario no podrán hacer uso de las instalaciones.

9. Correo electrónico del buzón para quejas y sugerencias para cualquier asunto relacionado con los laboratorios (sgcc.electronica.cc@gmail.com).

10. A los usuarios que, por su negligencia o descuido inexcusable, cause daños al laboratorio, materiales o equipo deberá cubrir los gastos que se generen con motivo de la reparación o reposición.

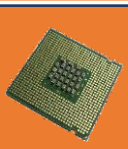
11. El incumplimiento a estas disposiciones faculta al profesor para que instruya la salida del infractor y en caso de resistencia, la suspensión de la práctica.

12. Los usuarios de laboratorio que sean sorprendidos haciendo uso indebido de equipos, sustancias, materiales, instalaciones y demás implementos, serán sancionados conforme a la legislación universitaria que le corresponda, según la gravedad de la falta cometida.

13. Los casos no previstos en el presente reglamento serán resueltos por el Jefe de Sección.

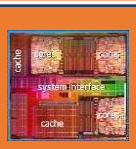
"POR MI RAZA HABLARÁ EL ESPÍRITU"

Cuautitlán Izcalli, Estado de Méx. a 18 de junio de 2024



# Laboratorio de Microprocesadores

## Lista de Materiales




| Cantidad | Material |
|---|---|
| 12 | Resistencias de 0.47KΩ a ½ W |
| 10 | Resistencias de 1KΩ a ½ W |
| 1 | Capacitor de 0.1 uF o 100 nF |
| 1 | Capacitor de 10uF |
| 1 | Capacitor de 4.7 uF |
| 1 | Capacitor de 470 uF |
| 9 | Led rojo |
| 1 | Led verde |
| 1 | Led amarillo |
| 1 | Diodo 1N4001 |
| 1 | Transistor BC547 |
| 1 | Circuito integrado 74LS00 |
| 2 | Circuito integrado 74LS04 |
| 1 | Circuito integrado 74LS125 |
| 1 | Circuito integrado 74LS86 |
| 2 | Circuito integrado 74LS138 |
| 1 | Circuito integrado PPI 8255 |
| 1 | Memoria EEPROM AT28C64 |
| 1 | Memoria SRAM 6116 |
| 1 | Cristal de cuarzo de 4 MHz |
| 1 | Microprocesador Z80 CPU |
| 1 | Display de cristal líquido LCD 16 x 2 |
| 1 | Push Button normalmente abierto |
| 2 | Bananas |
| 0.3m | Cable plano |

# Laboratorio de Microprocesadores

## Práctica 1 Herramientas de prueba para el sistema mínimo


**Tema**

1.1 Elementos de un microcomputador

**Objetivo**

- Implementar un módulo de visualización con leds para probar el estado de las entradas y las salidas de los buses del microprocesador.
- Implementar un dispositivo de prueba denominado “punta lógica” para probar los estados digitales de “0” lógico, “1” lógico y tercer estado “Z”.
- Comprobar las mediciones de tercer estado sobre un circuito 74LS125.

**Introducción**

Para realizar pruebas al sistema en una forma más eficiente es necesario utilizar herramientas de apoyo que permiten visualizar las señales de los buses del microprocesador.

En este tipo de sistemas es necesario verificar el estado de varias señales digitales de forma simultánea y debido a que los osciloscopios comunes solo tienen 2 canales es preferible utilizar puntas de prueba con LED´s o barras de LED´s, tal y como se muestra en las figuras 1.1 y 1.2.

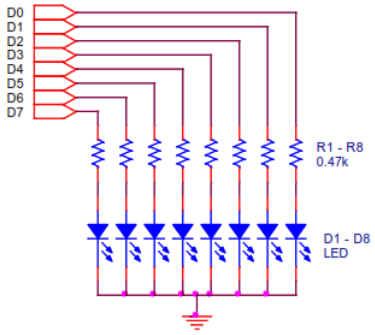



*Figura 1.1*

*Figura 1.2*

En los sistemas de microprocesadores varios de los circuitos que intervienen en su implementación son capaces de proporcionar salidas de tercer estado necesarias para dar la capacidad de conexión de 2 o más salidas en un mismo punto sin provocar un corto circuito, siempre y cuando exista un control de activación de cada una de las salidas conectadas a dicha unión.

El circuito de salida típico empleado en las compuertas TTL es el circuito de Totem Pole mostrado en la figura 1.3 para una compuerta AND, en el cual se logran los estados de "0" lógico y "1" lógico al saturar y cortar a los transistores Q3 y Q4 como se muestra en la tabla 1.1.

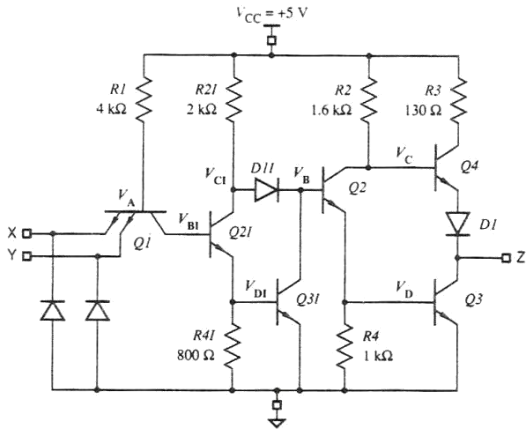



*Figura 1.3*

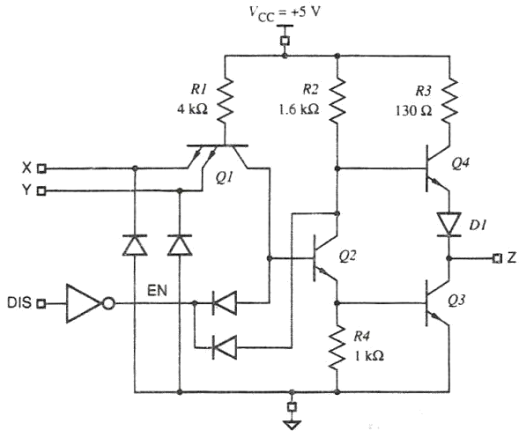



*Figura 1.4*

| X | Y | Q3 | Q4 | Salida |
|---|---|---|---|---|
| 0V | 0V | Saturación | Corte | 0V |
| 0V | 5V | Saturación | Corte | 0V |
| 5V | 0V | Saturación | Corte | 0V |
| 5V | 5V | Corte | Saturación | 5V |

*Tabla 1.1 Tabla de estados de salidas Totem Pole*

En este circuito los transistores Q3 y Q4 siempre están en estados contrarios, nunca se establecen los 2 en corte o los 2 en saturación, lo cual produce solo 2 posibles estados "0" y "1".

En la condición de tercer estado mostrada en la figura 1.4, los 2 transistores de la salida Totem Pole Q3 y Q4 de una compuerta NAND TTL se ponen en el estado de corte y por lo tanto presentan una impedancia alta entre la salida y tierra y entre la salida y Vcc, en otras palabras, al estar los 2 transistores en corte la terminal de salida es una terminal abierta o flotante que no tiene valor de "0" o de "1". Esta condición no puede ser interpretada por un multímetro ya que no existe un valor de voltaje que defina a este estado.

En esta práctica se implementarán 2 circuitos:

- El primero es el circuito de leds o barra de leds de la figura 1.1 que deberá implementarse en forma permanente como se muestra en la figura 1.2. Este circuito no se probará en el laboratorio, se realizará de forma externa y se utilizará a partir de la práctica 4.
- El segundo denominado "punta lógica" el cual es capaz de identificar a través del encendido de 3 leds independientes, alguno de los 3 estados posibles de un circuito que presente alta impedancia además de los dos estados lógicos "0" y "1". Debido a que el estado de alta impedancia es muy similar al estado de desconexión, entonces la punta de prueba al aire mostrará un estado similar al tercer estado.

Este circuito debe ser implementado de forma permanente para emplearlo como instrumento de prueba en todas las partes que forman al sistema de microprocesador, una posible forma de implementación es la siguiente.

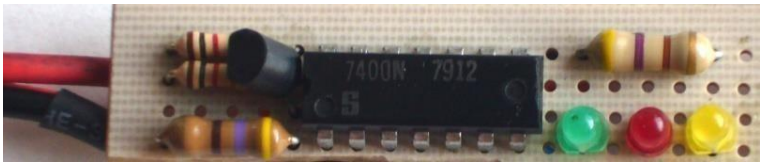



*Figura 1.5 Circuito de punta lógica*

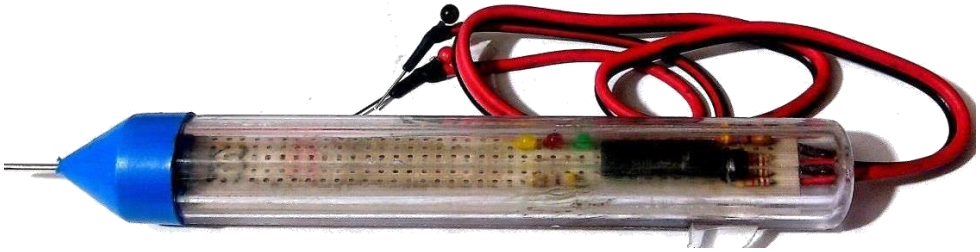


*Figura 1.6 Punta lógica con puntas de alimentación y prueba*

**Actividades Previas a la Práctica**

1. El alumno deberá realizar la lectura de la práctica de laboratorio.
2. Traer el circuito de la figura 1.7 armado en la tableta de conexiones.

**Material**

8 Diodo LED Rojo ó barra de 10 leds (deberán adicionarse 2 resistencias de 0.47 kΩ a ½ W.)
1 Diodo LED Rojo
1 Diodo LED Verde
1 Diodo LED Amarillo
12 Resistencias de 0.47kΩ a ½ W.
1 Circuito Integrado 74LS00
1 Circuito Integrado 74LS125
1 Transistor BC547
2 Bananas
Alambre de conexión
Cable plano (30 cm.)
1 Tableta de conexiones

**Equipo**

1 Multímetro

**Procedimiento Experimental**

1. Implemente el circuito de la figura 1.7.
2. Alimente el circuito con 5V, así como la compuerta 74LS00 (terminal 14 = 5V. y terminal 7 = 0V.)
3. Compruebe el comportamiento del circuito llevando la punta de prueba a los 3 estados posibles de un circuito lógico:

    - “0” lógico igual a 0V

- "1" lógico igual a 5V
- "Z" alta impedancia o tercer estado igual a punta al aire o desconectada

y proceda al llenando de la tabla 1.2.

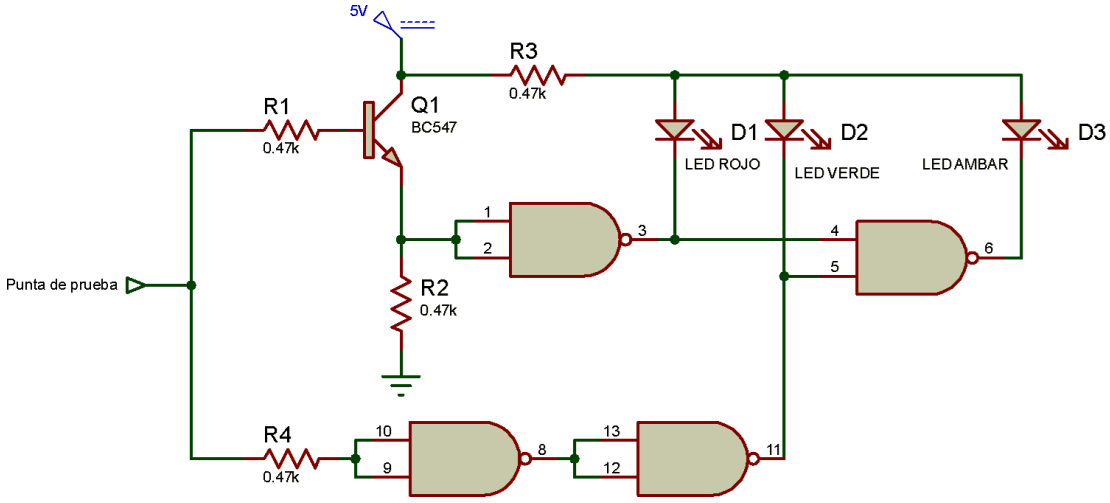



*Figura 1.7 Esquemático de punta lógica*

| Estado punta lógica | Voltaje de Entrada | Estado Led Rojo | Estado Led Verde | Estado Led Amarillo |
|---|---|---|---|---|
| 0 | 0 V | | | |
| 1 | 5 V | | | |
| Z | Desconectada | | | |

*Tabla 1.2 Tabla de valores del circuito de punta lógica*

4. Implemente el circuito de la figura 1.8 manteniendo el circuito anterior.

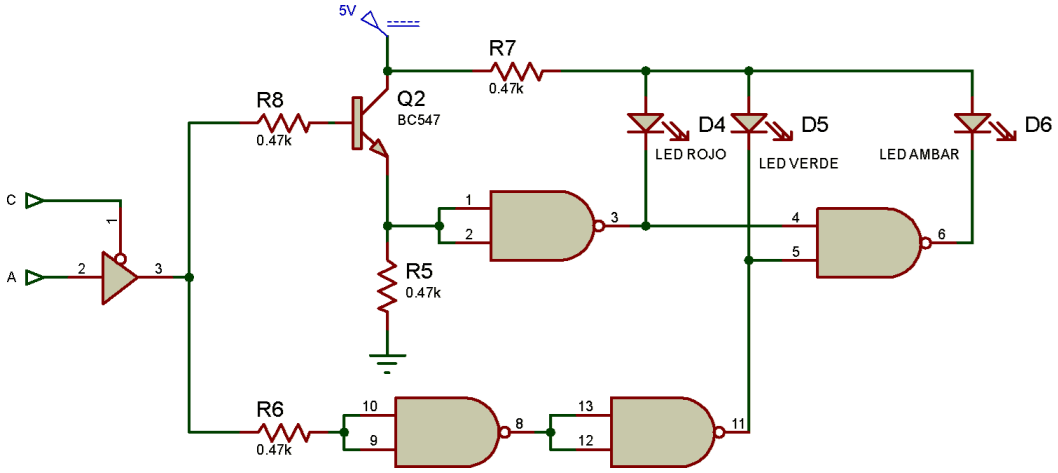
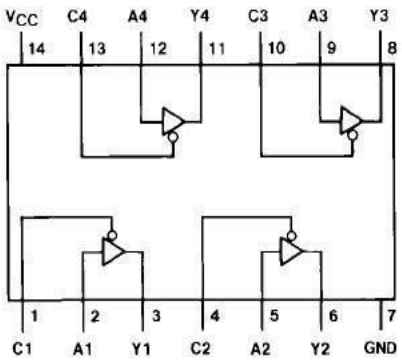



*Figura 1.8 Circuito de prueba*

5. Alimente el circuito 74LS125 con 5V en terminal 14 y 0V en la terminal 7.




Y = A

| Inputs | | Output |
|---|---|---|
| A | C | Y |
| L | L | L |
| H | L | H |
| X | H | Hi-Z |

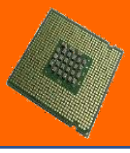
H = HIGH Logic Level
L = LOW Logic Level
X = Either LOW or HIGH Logic Level
Hi-Z = 3-STATE (Outputs are disabled)

*Figura 1.9 Hoja técnica de circuito 74LS125*

6. Compruebe la tabla de verdad del buffer mostrada en la figura 1.9, empleando para ello el circuito de prueba de tercer estado.

**Cuestionario**

1. Explique el comportamiento de la compuerta lógica de tercer estado mostrada en la figura 1.8 considerando la tabla de verdad de la figura 1.9.
2. Explique el concepto de compuertas con salida de colector abierto incluyendo el diagrama de una compuerta comercial de este tipo.
3. Explique el concepto de OR alambrada (Wired OR) e indique a través de un diagrama de conexiones la forma en que funciona.
4. Explique porque se requiere que los circuitos de memoria y microprocesadores tengan salidas de tercer estado.



# Laboratorio de Microprocesadores

## Práctica 2 Generador de reloj y autoreset para microprocesador


**Tema**

2.1.1 Sistema de reloj.
2.1.3 Características del microprocesador

**Objetivos**

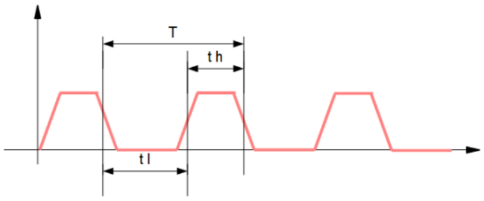
- El alumno implementará un reloj digital para alimentar al microprocesador Z80.
- Implementar un circuito de auto reset para inicializar el funcionamiento del microprocesador Z80.
- Comprobar las condiciones que presenta el µP Z80 al insertar la señal de reset.

**Introducción**

Los microprocesadores son básicamente circuitos digitales secuenciales que generan conjuntos de señales a través de autómatas finitos para que así el microprocesador pueda controlar a los dispositivos asociados que le permitan realizar sus funciones.

Debido a su naturaleza secuencial, se requiere contar con un circuito de reloj que defina la frecuencia de cambio del sistema secuencial. El reloj es el primer elemento indispensable en todos los sistemas con microprocesadores como se muestra en el esquema de Von Neumann. Este sistema es el encargado de proporcionar una señal digital con una frecuencia característica establecida por el microprocesador en cuestión.

Las señales de reloj de los microprocesadores tienen otras características importantes, tales como el ciclo de trabajo, definido como la relación entre el tiempo que la señal de reloj se mantiene en estado alto ( $th$) y el periodo de la señal.




$$Ciclo\ de\ Trabajo\ [\%] = \frac{th}{T} * 100 = \frac{th}{th + tl} * 100$$

*Figura 2.1.*

La señal digital generada por el sistema de reloj actúa como la señal de sincronía del microprocesador, el cual ejecuta las instrucciones del programa e interactúa con los elementos que conforman al sistema tales como las memorias RAM y ROM y los puertos de entrada (I) y salida (O) en función de los cambios que realiza la señal

Todas las operaciones que realiza un microprocesador para ejecutar una instrucción o para activar un circuito, se ejecutan en sincronía con los pulsos de reloj proporcionados por el oscilador digital. Es por eso por lo que todos los sistemas de microprocesador requieren de una base de tiempo sobre la cual trabajar.

El conocimiento de las señales de reloj requeridas permite elaborar diagramas de tiempo bajo los cuales opera el microprocesador, haciendo posible sincronizar las señales del microprocesador con los tiempos de acceso, retardo o de control de sus periféricos.

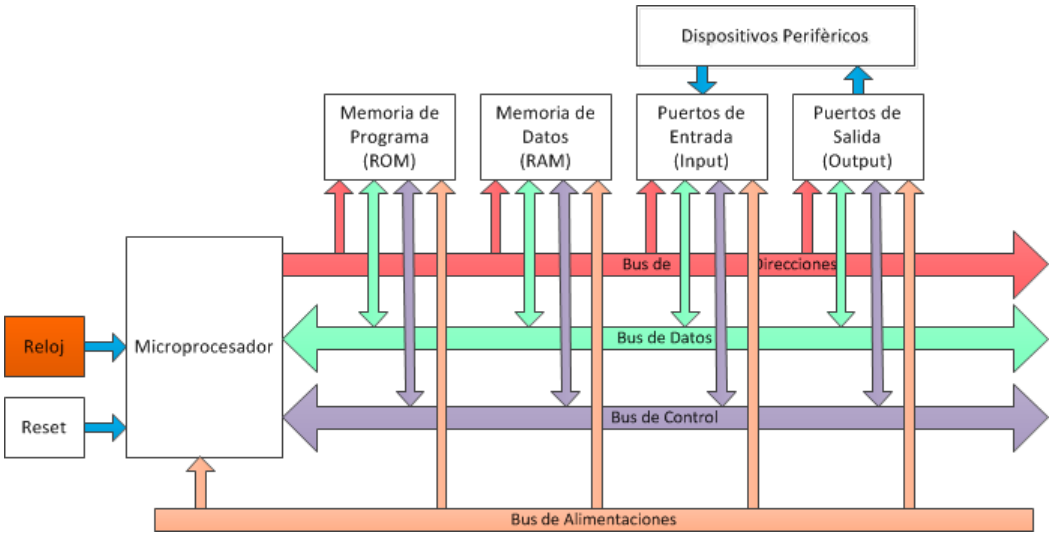



*Figura 2.2 Reloj dentro del esquema de Von Neumann.*

Una forma de generar la señal de reloj es emplear compuertas lógicas retroalimentadas, usando resistencias, capacitores y cristales de cuarzo como se muestra en la figura 2.3. Se observa que el circuito utiliza un cristal de cuarzo de 4 MHz para fijar la frecuencia de la oscilación, este tipo de circuito garantiza oscilaciones estables asociadas con la frecuencia del cristal.

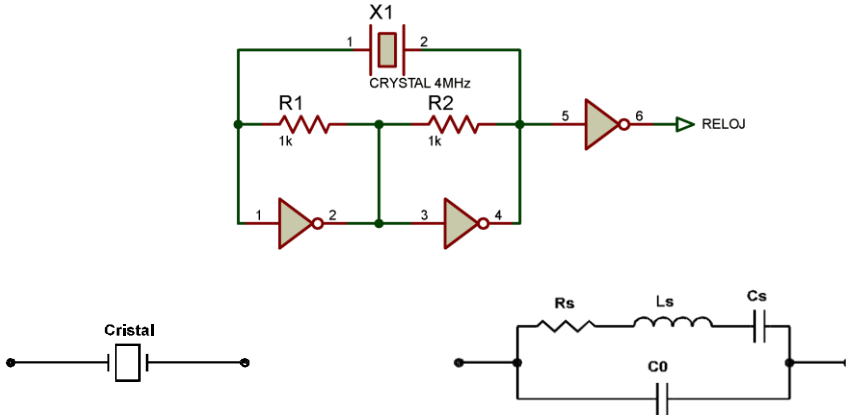



*Figura 2.3 Circuito de reloj digital y analogía de comportamiento.*

En los microprocesadores, la señal de reset se utiliza para llevar al microprocesador a un estado conocido. La señal de reset inicializa al µP Z80 de forma interna del siguiente modo:

- Limpia el Flip Flop asociado con la habilitación de las interrupciones mascarables
- Limpia los registros: contador de programa (PC), registro de interrupción (I) y refresco de memoria (R).
- Selecciona el modo 0 de interrupción mascarable.
- Deshabilita las interrupciones mascarables.

De forma externa la señal de reset se refleja en el estado de algunas de sus señales como se indica:

- El bus de direcciones y de datos se pone en alta impedancia mientras la señal de reset está activa.
- Todas las señales de salida del bus de control se mantienen en su estado inactivo.

Además, para que la inicialización sea exitosa, la señal de reset debe mantenerse activa en bajo por lo menos 3 ciclos completos de reloj.

Para la implementación de un circuito de reset, se emplea una malla RC que proporciona un "0" lógico al alimentar al sistema puesto que el capacitor está originalmente descargado y después de 5τ (constantes de tiempo), el capacitor es cargado al 99.3% del voltaje de alimentación de 5V, lo que produce que la señal de reset pase a inactiva. Para lograr la especificación de 3 ciclos completos de reloj para una aplicación correcta del reset, se requiere que el voltaje de carga del capacitor se mantenga por debajo del voltaje $V_{IL}$ considerado como "0" lógico por un tiempo lo suficientemente largo para que transcurran 3 ciclos de reloj del microprocesador tal y como se muestra en la figura 2.4

La ecuación que describe comportamiento de carga del capacitor está definida por:

$$Vc(t) = 5\,(1 - e^{-\frac{t}{\tau}})$$

y el tiempo necesario para alcanzar el valor de $V_{IL}$ = 0.8 V queda definido por:

$$t = -\,[ln\,(1 - \frac{0.8}{5})]\,\tau$$

Si se cumplen estas condiciones entonces el microprocesador se auto inicializará al alimentar al circuito completo.

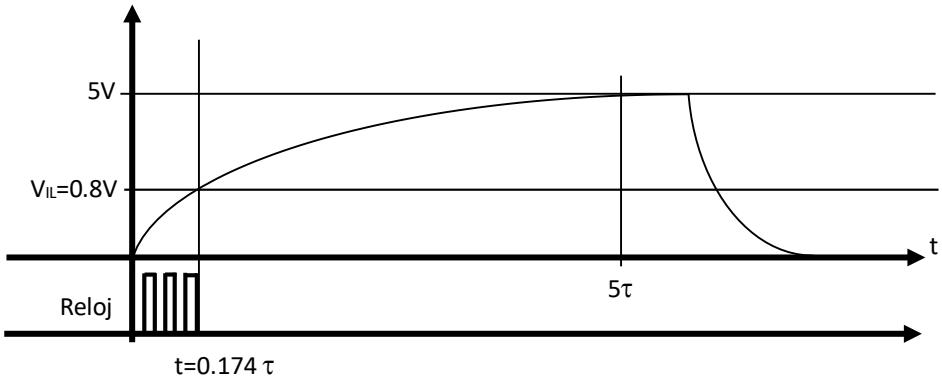
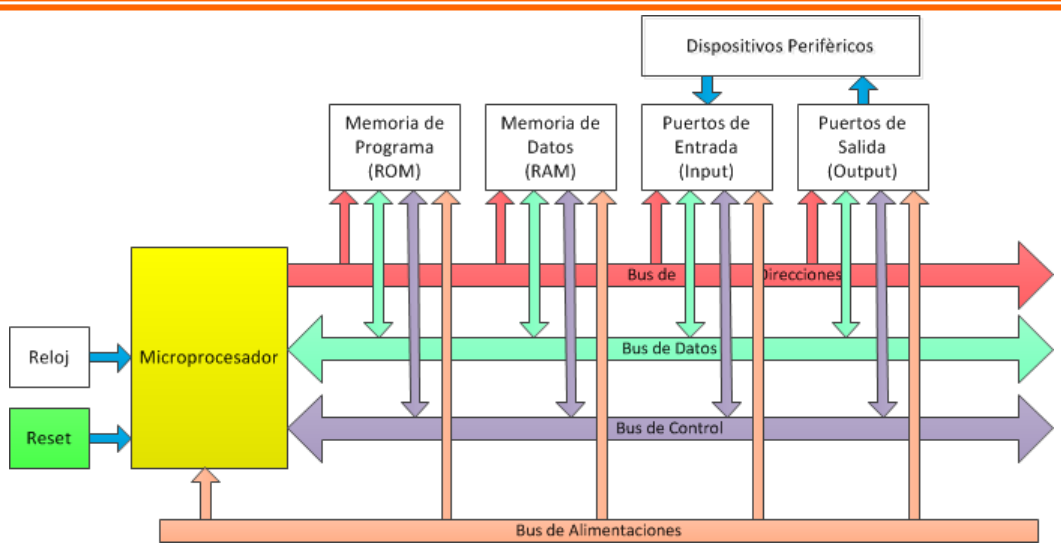



*Figura 2.4 Diagrama de tiempo para el reset del microprocesador.*

Sin embargo, al estar el microprocesador en su funcionamiento normal se requiere ocasionalmente resetear al microprocesador y eso se puede lograr introduciendo un switch en paralelo a las terminales del capacitor descargándolo de forma directa y produciendo un estado bajo durante todo el tiempo que el switch permanezca cerrado y la señal de reset regresará al estado de inactividad (5V) al cargarse nuevamente al capacitor después de desconectar el switch.

En esta práctica se integrarán el circuito de reloj y de autoreset al microprocesador de acuerdo con el esquema de Von Neumann como se muestra en la figura 2.5




.

*Figura 2.5 Microprocesador y reset en el esquema de Von Neumann.*

**Actividades Previas a la Práctica**

1. El alumno deberá realizar la lectura de la práctica de laboratorio.
2. Obtenga el valor de tiempo teórico necesario para alcanzar el valor $V_{IL}$ utilizando la fórmula que describe la carga del capacitor y la constante de tiempo de la malla RC.
1. Traer el circuito de la figura 2.6 armado en la tableta de conexiones.

**Material**

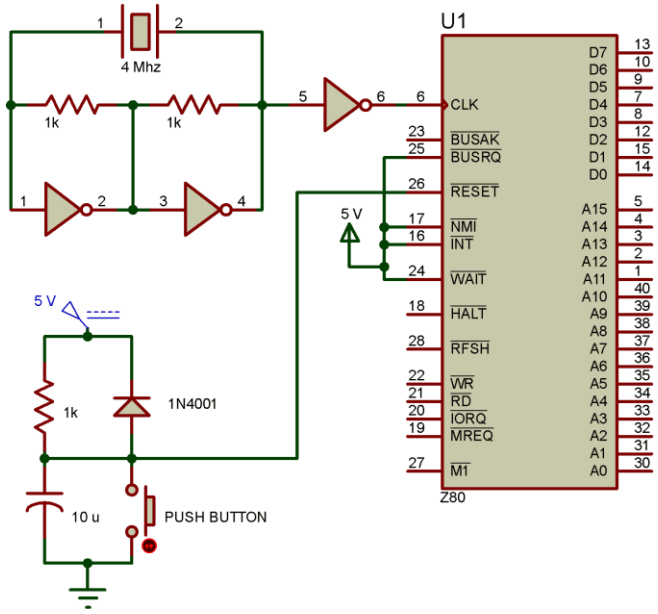
1 Capacitor de 10 uF
1 Switch Push Button normalmente abierto
1 Diodo 1N4001
1 Microprocesador Z80
3 Resistencias de 1 KΩ ½W
1 Circuito integrado 74LS04 o 7404, NO emplear 74HC04
1 Cristal de cuarzo de 4 MHz.
1 Tableta de conexiones
Alambre de conexión

**Equipo**

1 Fuente de alimentación de C.D.
1 Osciloscopio.
1 Multímetro

**Procedimiento Experimental**

1. Implemente el circuito de la figura 2.6.
2. El circuito de la figura 2.6 se utilizará como el primer elemento del sistema con microprocesador y por lo tanto deberá mantenerse armado para las pruebas posteriores.




| Circuito | Vcc | GND |
|---|---|---|
| 74LS04 | 14 | 7 |
| Z80 CPU | 11 | 29 |

*Figura 2.6 Circuito de reloj y autoreset*

3. Realice la conexión en la tableta de conexiones utilizando como referencia la fotografía de la figura 2.7

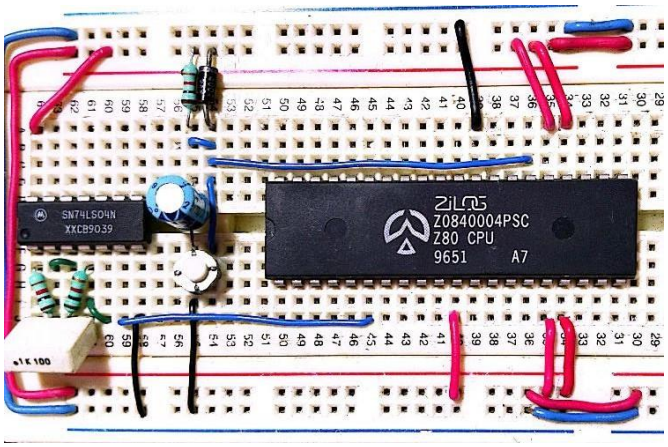



*Figura 2.7*

4. Con el osciloscopio observe la señal de salida del inversor 74LS04 (terminal 6) y mida los valores indicados en la tabla 2.1, dibuje la forma de onda.

| Frecuencia (f) | Periodo (T) | Tiempo en alto (th) | Tiempo en bajo (tl) | Ciclo de Trabajo |
|---|---|---|---|---|
| | | | | |

*Tabla 2.1*

5. Observe la señal de reset en la salida del circuito y compruebe su comportamiento al presionar y liberar el botón de reset (SW1).
6. Grafique la señal de reset indicando los parámetros de voltaje y tiempo, para realizar esta operación puede utilizar la función de Stop o captura de imagen que poseen los osciloscopios digitales.

7. Empleando el circuito de punta lógica compruebe que mientras está presionado el botón de reset, ambos buses de direcciones y de datos permanecen en tercer estado.
8. Pruebe y anote el estado de las señales siguientes empleando el circuito de punta lógica mientras activa la señal de reset en cero lógico.

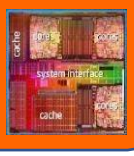
   $\overline{BUSAK}$, $\overline{HALT}$, $\overline{M1}$, $\overline{REFRESH}$, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ $y$ $\overline{WR}$
9. Debido a que este sistema aún no contiene una memoria ROM de programa, una vez que se libera el botón de reset, el microprocesador intenta leer la dirección 0000H, pero al estar las terminales del bus de datos al aire, entonces la lectura es teóricamente FFH o basura de forma real, por lo tanto, el microprocesador no puede ejecutar ningún programa lógico.
10. Aun bajo estas condiciones, es posible probar algunas de las señales de salida del bus de control cuando es liberado el switch de reset, puesto que el microprocesador genera las señales sin importar que no se tengan los circuitos de apoyo del esquema de Von Neumann.
11. Verifique el funcionamiento de las señales: $\overline{M1}$, $\overline{REFRESH}$, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ $Y$ $\overline{WR}$ empleando el osciloscopio y con la señal de reset inactiva (5V).

**Cuestionario**

1. Calcule el ciclo de trabajo (duty cycle) de las señales de reloj obtenidas, de acuerdo con lo indicado en la introducción.
2. Obtenga la hoja técnica de un resonador cerámico de 4 MHz y describa que características adicionales tiene con respecto a un cristal de cuarzo.
3. Indique el estado que presentarían los registros del microprocesador Z80 al insertar la señal de reset activa en 0 V.
4. Describa la función de las señales: $\overline{M1}$, $\overline{REFRESH}$, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ $Y$ $\overline{WR}$ del microprocesador Z80.
5. Determine cuál sería la frecuencia de reloj mínima para poder realizar correctamente el proceso de auto reset, calcúlelo en función de la ecuación de carga del capacitor.

# Laboratorio de Microprocesadores

## Práctica 3 Memoria EEPROM o memoria de programa



**Tema**

2.1.3. Memoria de programa

**Objetivos**

- El alumno realizará la conexión de una memoria EEPROM a un microprocesador Z80.
- El alumno implementará el circuito decodificador de direcciones.
- El alumno probará un programa de prueba en lenguaje ensamblador que será ejecutado en el microprocesador Z80.

**Introducción**

Los microprocesadores son circuitos de muy alta escala de integración que son capaces de ejecutar una serie de códigos binarios proporcionados en secuencia sobre sus terminales de datos y controlados por las terminales de direcciones y por las señales de control, pero dichos códigos que conforman un programa no pueden almacenarse dentro del microprocesador y por lo tanto en los sistemas de microprocesadores es necesario añadir un sistema de memoria no volátil que contenga al programa.

Esta memoria de programa es típicamente una memoria de solo lectura o memoria ROM (Read Only Memory) que es programada previamente en forma independiente del sistema de microprocesador, dentro de ella y en cada una de sus localidades se almacena un dato binario que representa el código de máquina de una instrucción en lenguaje ensamblador o parte del código de máquina de una instrucción mayor.

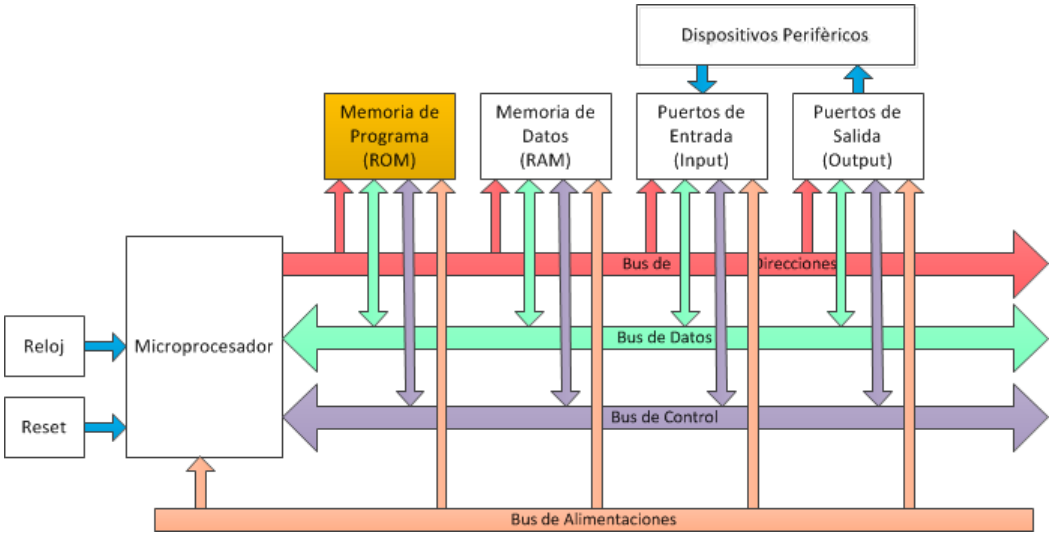



*Figura 3.1 Memoria de programa ROM en el esquema de Von Neumann*

Utilizando las señales de sus buses, los microprocesadores leen, trasladan, decodifican y ejecutan cada uno de los códigos contenidos en la memoria ROM y con estas acciones el sistema de microprocesador es capaz de realizar la ejecución lógica de un programa en lenguaje ensamblador.

En esta práctica se realizará la conexión de una memoria EEPROM AT28C64, figura 3.2, con una capacidad de (8K x 8) que se utilizará para el almacenamiento de los programas de prueba del sistema de microprocesador.

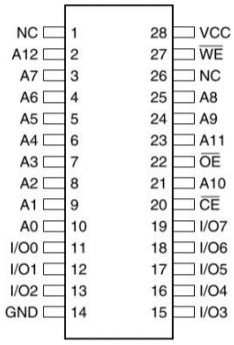



*Figura 3.2 Memoria EEPROM AT28C64*

Además, se comprobará el funcionamiento de un programa de prueba que nos permita asegurar que el sistema de microprocesador está decodificando correctamente los códigos de máquina de las instrucciones almacenadas en la EEPROM.

El programa de prueba realizará la carga del acumulador A del microprocesador Z80 con el valor de 00H en 8 bits, el registro B con el valor 01H, con esos valores cargados, se procede a sumar ambos registros dejando el valor de la suma en el acumulador, después el valor del acumulador se intentará almacenar en la localidad 1040H, posteriormente el registro B se incrementará de 1 en 1 y se realizará de forma infinita un ciclo de suma como se muestra en el diagrama de la figura 3.3

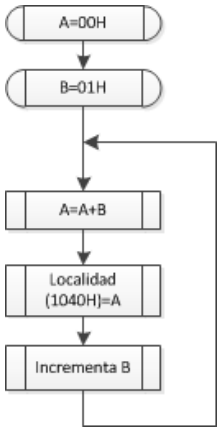



Figura 3.3

Este sistema aún no tiene memoria RAM en donde guardar los datos, así que el almacenamiento en la localidad 1040H no puede realizarse de forma real, pero como el microprocesador no tiene un sistema que verifique la existencia de la memoria RAM, de todas formas, realiza el procedimiento para escribir el dato, aunque ningún dispositivo lo reciba.

Aun así, el dato a almacenar será visible en el bus de datos en el cuarto ciclo de máquina de la ejecución de la carga en la localidad de memoria.

Para realizar la conexión de la memoria EEPROM se debe tomar cuenta que al insertar el pulso de RESET, este microprocesador inicializa el valor del contador de programa (PC) a cero y por lo tanto la memoria se conectará a partir de la dirección 0000H dentro del mapa de memoria que es la dirección en donde debe estar el código de máquina de la primera instrucción.

En la figura 3.4 se muestra el mapa de memoria del microprocesador Z80 con un tamaño de 64K x 8 y la posición que ocupará la EEPROM dentro del mapa, desde la dirección 0000H hasta la 1FFFH y la asignación de los bits de direccionamiento.

Este microprocesador emplea 16 bits para el direccionamiento de la memoria, de los cuales se emplearán los 13 bits menos significativos para selección de la localidad interna de la memoria y los 3 bits más significativos se usarán para la decodificación de la posición de la memoria EEPROM dentro del mapa de memoria, no se utilizarán bits para la selección de circuito debido a que el arreglo solo constará de 1 circuito integrado AT28C64.

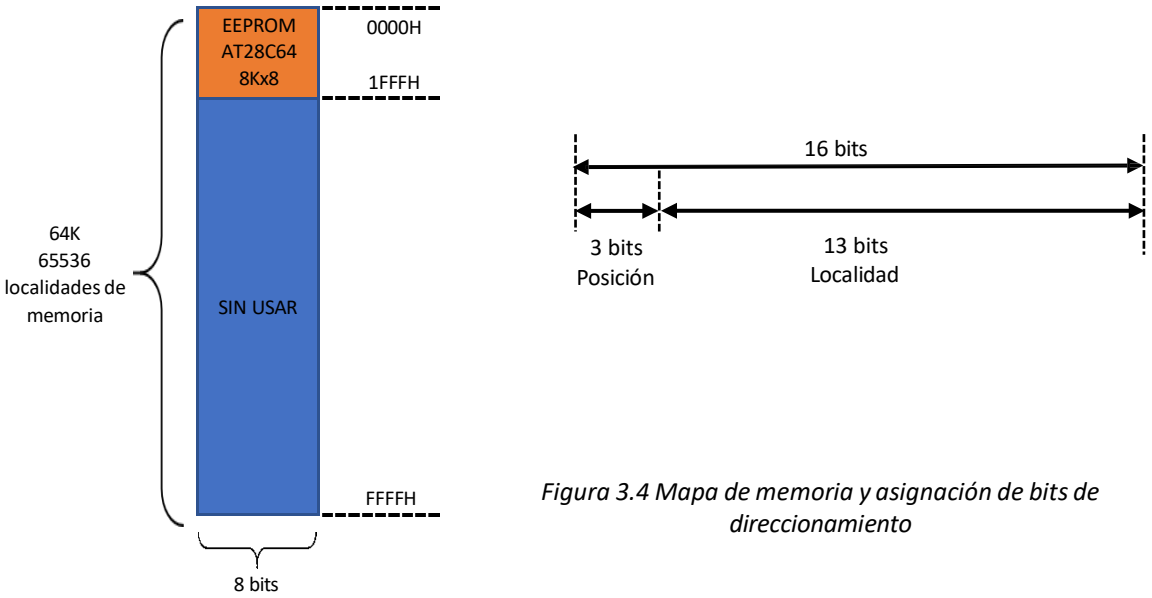



*Figura 3.4 Mapa de memoria y asignación de bits de direccionamiento*

Para la activación de la memoria en el instante correcto se deberán considerar además de los tres bits de direcciones (A15 – A13) = 000, las señales $\overline{MREQ} = 0$ $y$ $\overline{RD} = 0$, las que en conjunto generarán la señal de activación $\overline{CE} = 0$ de la memoria, esto se realizará a través de un circuito Decodificador / Demultiplexor 74LS138 y una compuerta XOR 74LS86.

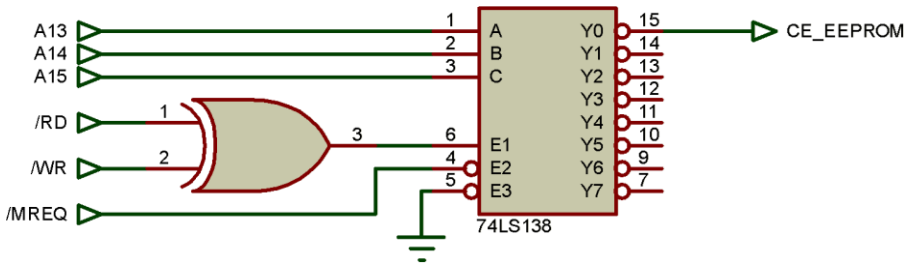



*Figura 3.5 Circuito de activación de la memoria EEPROM*

Para la realización de esta práctica se utilizará el software de ambiente integrado Z80 (**Z80 Simulator IDE**), con el cual se realizará la escritura del programa en lenguaje ensamblador.

Se ensamblará el programa para su conversión a código de máquina y se generará el archivo en formato hexadecimal (**.hex)** para poderlo descargar en el circuito de memoria AT28C64 a través del programador universal SuperPro 610P o SuperPro M.

Debido a que el programa es muy pequeño, solo se ocuparán las primeras 11 localidades de memoria y por lo tanto para simplificar las conexiones, se omitirán las líneas A11 y A12 del microprocesador y las líneas correspondientes de la memoria se conectarán a tierra, con lo cual siempre estarán fijas a “0”.

**Actividades Previas**

1. El alumno deberá realizar la lectura de la práctica de laboratorio.
2. El alumno generará un proyecto en lenguaje ensamblador siguiendo el procedimiento mostrado en los siguientes incisos empleando el software **Z80 simulador IDE** cuyo icono se muestra en la figura 3.6.


*Figura 3.6*

3. Utilizando el software Z80 Simulator IDE, seleccione la pestaña Tools/Assembler para abrir la ventana de edición según se muestra en la figura 3.7.

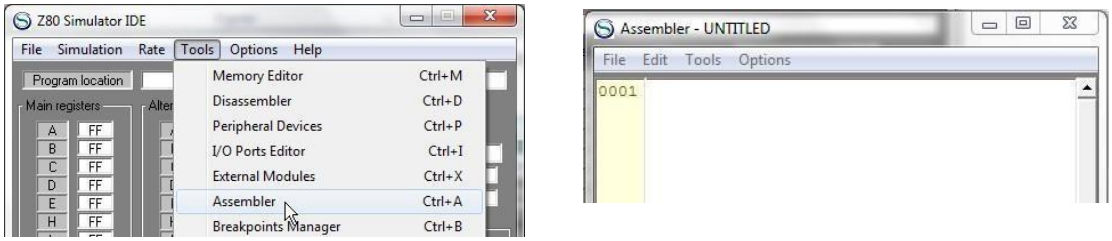



*Figura 3.7 Pantallas del software Z80 Simulator IDE.*

4. El alumno editará el programa de suma de 2 registros mostrado en la figura 3.8.
5. Salve el programa con el nombre Prac03.asm
6. Active la opción para generar el código hexadecimal del programa, figura 3.9 y proceda a ensamblarlo para obtener el código de máquina tal y como se muestra en la figura 3.10, con esta opción se generarán 3 archivos adicionales: Prac03.lst que contiene un listado de códigos e instrucciones, Prac03.obj código objeto que sirve para hacer el ensamblado y Prac03.hex que se utiliza para hacer la programación de la memoria EEPROM.

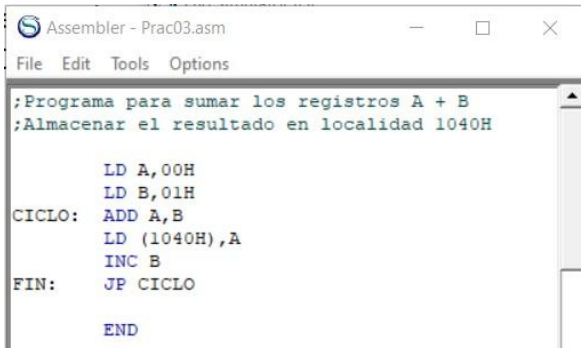



*Figura 3.8 Programa de suma de 2 registros*

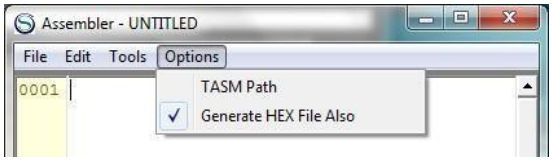



*Figura 3.9 Opción para generar el archivo .hex*

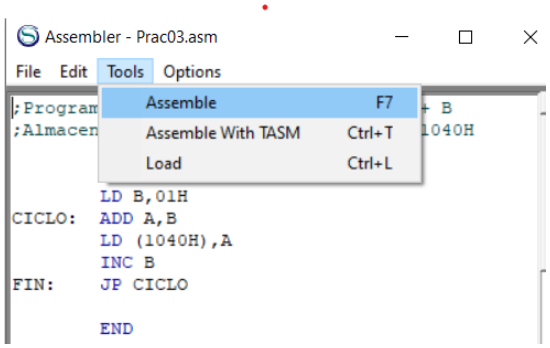



*Figura 3.10 Ensamblado del programa*

7. El código de máquina generado se muestra en la figura 3.11, indicándose en el recuadro rojo: la localidad de memoria, el código de máquina y la instrucción correspondiente.

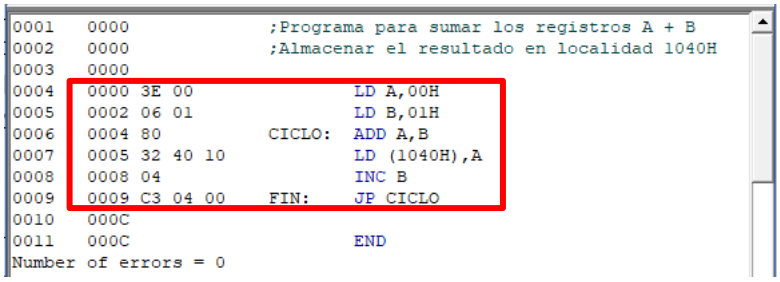



*Figura 3.11 Localidad de memoria, código de máquina e instrucción del programa*

8. El alumno programará la memoria AT28C64 con el código de máquina mostrado en la figura 3.11, considerando que en cada localidad debe grabarse un byte, utilizando el programador universal y empleando el archivo Prac03.hex que se generó en el paso anterior y se almacenó en el mismo subdirectorio.
9. Traer el circuito de la figura 3.13 armado en la tableta de conexiones utilizando de referencia la figura 3.12.

**Material**

1 Sistema mínimo con microprocesador, circuito de reloj y circuito de reset
1 Memoria AT28C64 EEPROM de 8Kx 8
1 Capacitor de 470 uF.
1 Circuito integrado 74LS86
1 Circuito integrado 74LS138
Alambres de conexión

**Equipo**

Fuente de C.D.
Osciloscopio
Multímetro
Punta lógica para prueba de tercer estado
Circuito de leds implementado en la práctica 1

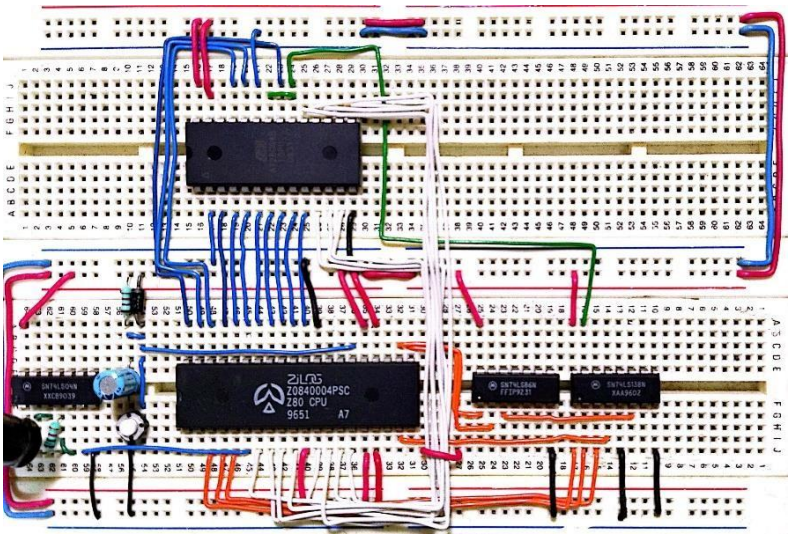


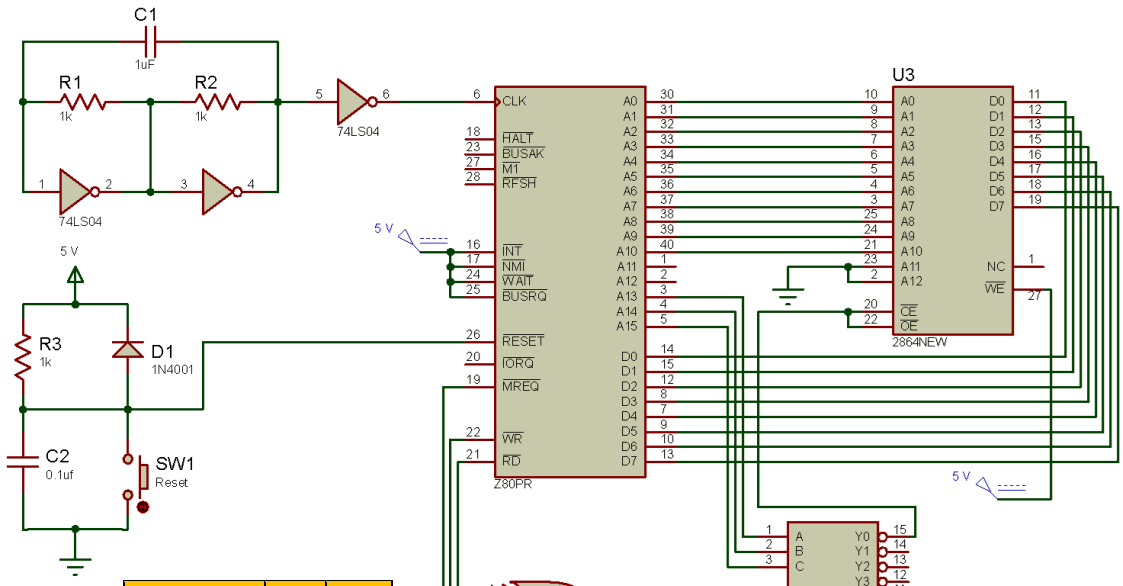

*Figura 3.12 Circuito implementado en 2 tabletas de conexiones.*

**Procedimiento experimental**

1. Implemente el circuito mostrado en la figura 3.13 considerando que ya se tienen armados los circuitos de reloj y reset y solo deberán adicionarse la memoria EEPROM AT28C64, el multiplexor 74LS138 y la compuerta 74LS86.
2. Pruebe el sistema completo para comprobar la presencia de los datos y las instrucciones en el bus de datos del microprocesador, para ello deberá conectar el circuito de leds en el bus de datos y cambiar el cristal de cuarzo del reloj por un capacitor de 470 uF, lo cual reducirá la velocidad del reloj y permitirá observar los cambios del bus de datos.
3. Genere una tabla con los valores que se presentan en el bus de datos y compruebe que son los códigos de las instrucciones del programa. Deberá notar que después del código de la instrucción LD (1040H),A con código de máquina 32, 40, 10 aparecerá el valor enviado a la memoria intercalado con los códigos de las instrucciones, el cual se irá modificando cada vez que se haga un ciclo.
4. Observe y dibuje las señales: MREQ, RD y la señal de activación de la memoria EEPROM.

**Cuestionario**

1. Calcule el tiempo de ejecución de un ciclo completo del programa de la figura 3.9 considerando un reloj de 4MHz. y la duración de cada una de las instrucciones que intervienen, especificada en las tablas de instrucciones del microprocesador Z80.
2. Diseñe un programa en lenguaje ensamblador que divida un dato de 16 bits entre otro dato de 8 bits y compruébelo empleando el simulador Z80, entregue los resultados de la simulación a su profesor de laboratorio.




| Circuito | VCC | GND |
|---|---|---|
| 74LS04 | 14 | 7 |
| Z80 CPU | 11 | 29 |
| AT28C64 | 28 | 14 |
| 74LS86 | 14 | 7 |
| 74LS138 | 16 | 8 |

*Figura 3.13*

# Laboratorio de Microprocesadores

## Práctica 4 Puertos de entrada/salida (Input/Output)


**Tema**

2.1.5. Puertos de entrada
2.1.6. Puertos de salida

**Objetivos**

- El alumno realizará la conexión de un circuito de interfaz periférica programable (PPI 8255) a un microprocesador Z80.
- El alumno ensamblará y probará un programa que genera tres patrones de 8 bits conectados en el puerto A del circuito PPI 8255, para seleccionar que patrón se mostrará, se emplean 3 bits de entrada del puerto B.

**Introducción**

Otro elemento en el esquema de Von Neumann son los puertos de entrada /salida o (input / output) los cuales le permiten al microprocesador conectarse con los dispositivos periféricos y de esta manera trasladar información hacia el microprocesador o del microprocesador hacia los dispositivos periféricos. La figura 4.1 nos muestra estos elementos.

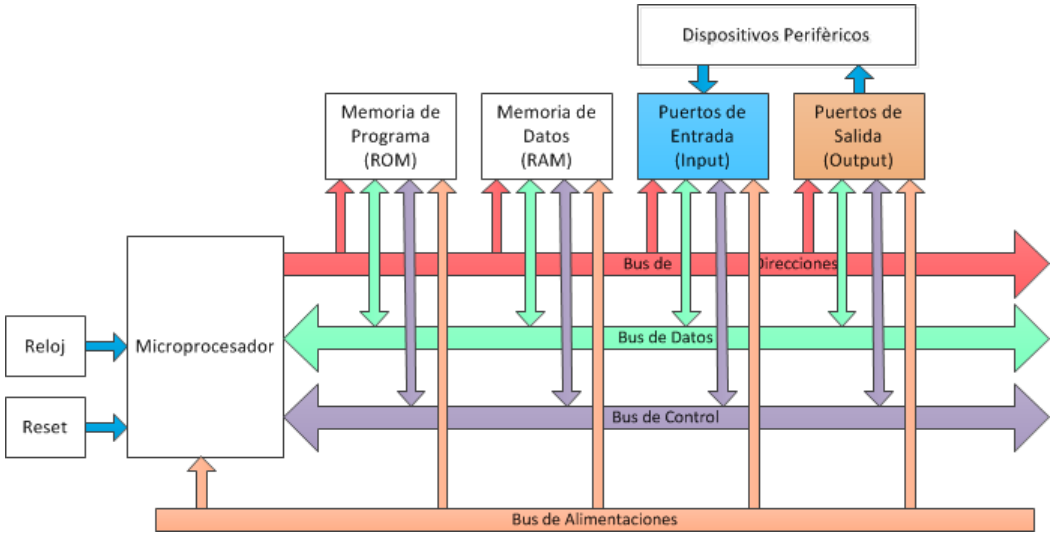



*Figura 4.1 Puertos de entrada /salida (I / O) en el esquema de Von Neumann*

Los puertos pueden ser implementados con registros de flip- flops o con circuitos integrados programables que contienen todos los elementos necesarios para interconectar de forma correcta a los dispositivos periféricos, ahorrando espacio y complejidad a las conexiones.

En esta práctica emplearemos el circuito integrado 8255 que contiene 3 puertos programables de 8 bits cada uno, denominados puerto A, puerto B y puerto C y un puerto de control de 8 bits con el cual se configura el funcionamiento del dispositivo.

El dispositivo periférico empleado en esta práctica es el integrado PPI 8255 como el mostrado en la figura 4.2

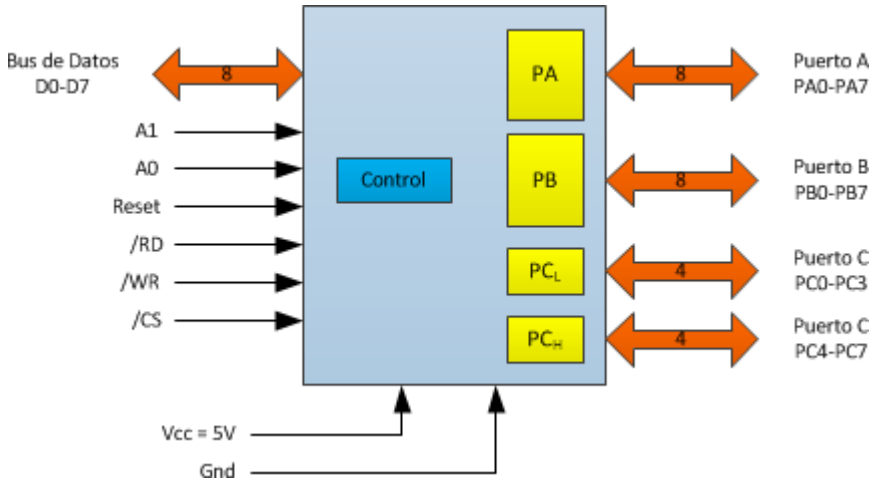



*Figura 4.2 Circuito PPI 8255*

Este circuito pertenece a la familia de circuitos desarrollados por INTEL para el soporte de sus microprocesadores 80XXX y por lo tanto ya incluye las señales para el control de lectura y escritura, así como para realizar una inicialización externa y solo requieren la activación correcta para las direcciones de los 3 puertos de usuario y el puerto de control. La figura 4.3 muestra la asignación de terminales del PPI 8255.

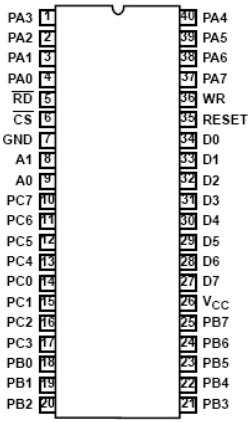



*Figura 4.3 Asignación de terminales del PPI 8255*

Este dispositivo contiene 4 puertos; 3 de usuario y uno de control y por lo tanto solo necesita de 2 líneas de direcciones (A1 y A0) para definir el puerto al que deberá tener acceso según la tabla 4.1.

| A1 | A0 | Puerto | Dirección |
|---|---|---|---|
| 0 | 0 | Puerto A | 00H |
| 0 | 1 | Puerto B | 01H |
| 1 | 0 | Puerto C | 02H |
| 1 | 1 | Puerto de Control | 03H |

*Tabla 4.1 Asignación de puertos*

En la figura 4.4 se muestra el mapa de puertos correspondiente al sistema de microprocesador, donde se observa que la posición del circuito PPI 8255 se ha seleccionado en las primeras cuatro posiciones del mapa de puertos (puertos 00H, 01H, 02H y 03H) puesto que no hay ningún otro puerto conectado, no se producirán conflictos.

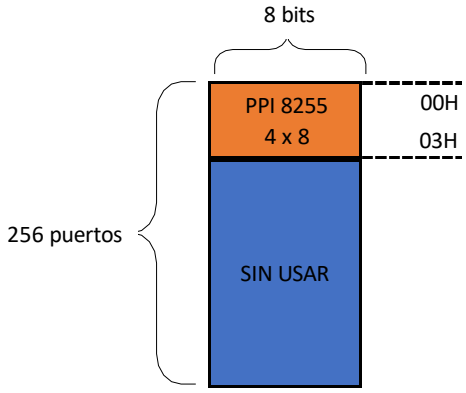



*Figura 4.4 Mapa de Puertos*

Para realizar la conexión se deberá implementar una función que active la señal de CE del PPI y además genere la señal de RESET del circuito de puertos como se muestra en la figura 5.5.

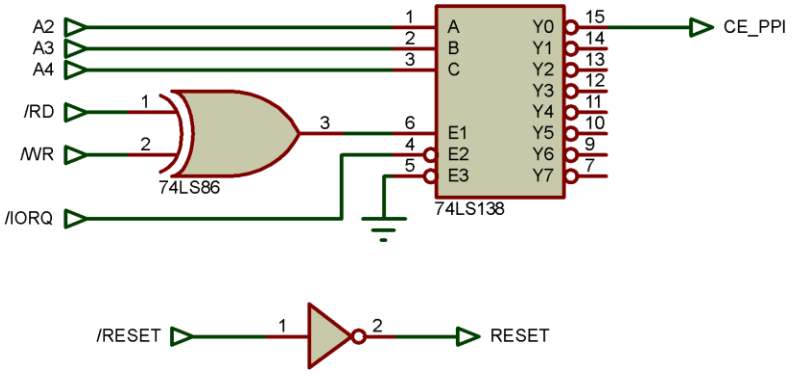



*Figura 4.5 Circuitos decodificador de direcciones para el PPI e inversor para la señal de reset.*

Para realizar la prueba de funcionamiento de todo el sistema se utilizará un programa que generará tres patrones de 8 bits, los cuales se seleccionaran a través de 3 bits (PB2, PB1 y PB0) del puerto B configurado como puerto de entrada. Los códigos que hay que insertar son los mostrados en la tabla 4.2.

*Tabla 4.2*

| PB2 | PB1 | PB0 | Patron |
|---|---|---|---|
| 1 | 0 | 0 | Patrón 1 |
| 0 | 1 | 0 | Patrón 2 |
| 0 | 0 | 1 | Patrón 3 |

- Patron 1: Encendido permanente de los 8 bits del puerto A insertando el código 100 en el puerto B de entrada.
- Patron 2: Encendido y apagado de los 8 bits del puerto A en forma intermitente insertando el código 010 en el puerto B de entrada.
- Patron 3: Corrimiento a la derecha de 2 bits del puerto A insertando el código 001 en el puerto B de entrada.de acuerdo a la figura 4.6.

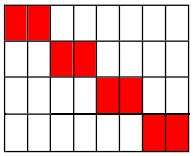
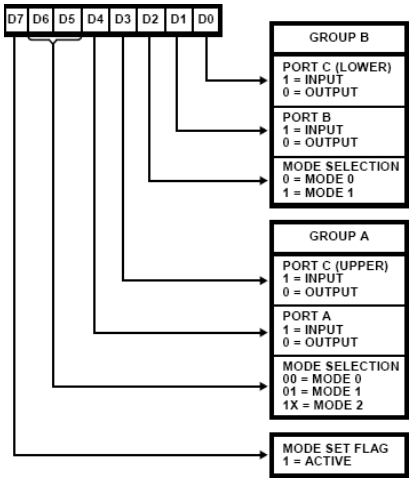


*Figura 4.6 Patrón 3 de encendido de los leds*

Para reducir la velocidad de encendido de los leds, se deberá sustituir el cristal de cuarzo del oscilador de señal cuadrada por uno de 4.7 uF.

Este programa no permite la utilización de subrutinas debido a que no se cuenta aún con memoria SRAM y no se puede establecer la localidad donde se localizará la pila y por lo tanto el programa repite varias veces la misma serie de instrucciones de consumo de tiempo debido a que el programa debe ser solo secuencial.

Para configurar el PPI 8255 con el puerto A de salida en modo 0, el puerto B de entrada en modo 0 y el puerto C de salida en modo 0, se deberá escribir la palabra de control (82H) de 8 bits en el puerto 03H de acuerdo con la siguiente asignación mostrada en la figura 4.7.




| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

*Figura 4.7 Asignación de bits para la configuración del PPI 8255*

**Actividades Previas**

1. El alumno deberá realizar la lectura de la práctica de laboratorio.
2. El alumno programará la memoria AT28C64 con el código de máquina generado por el programa mostrado en la figura 4.9, considerando que en cada localidad debe grabarse un byte.
3. Traer el circuito armado en la tableta de conexiones.

**Material**

1 Sistema mínimo con Microprocesador Z80, circuito de reloj, circuito de reset, memoria EEPROM AT28C64.
1 Circuito Integrado PPI 8255.
1 1 Circuito integrado 74LS138
Capacitor de 4.7 µF.
Alambres de conexión

**Equipo**

Fuente de C.D.
Osciloscopio
Multímetro
Punta lógica para prueba de tercer estado
Circuito de Leds de prueba

**Procedimiento Experimental**

1. Implemente el circuito mostrado en la figura 4.10 considerando que ya se tiene armado el sistema mínimo de microprocesador con la memoria EEPROM y solo deberá adicionarse el circuito integrado PPI 8255.

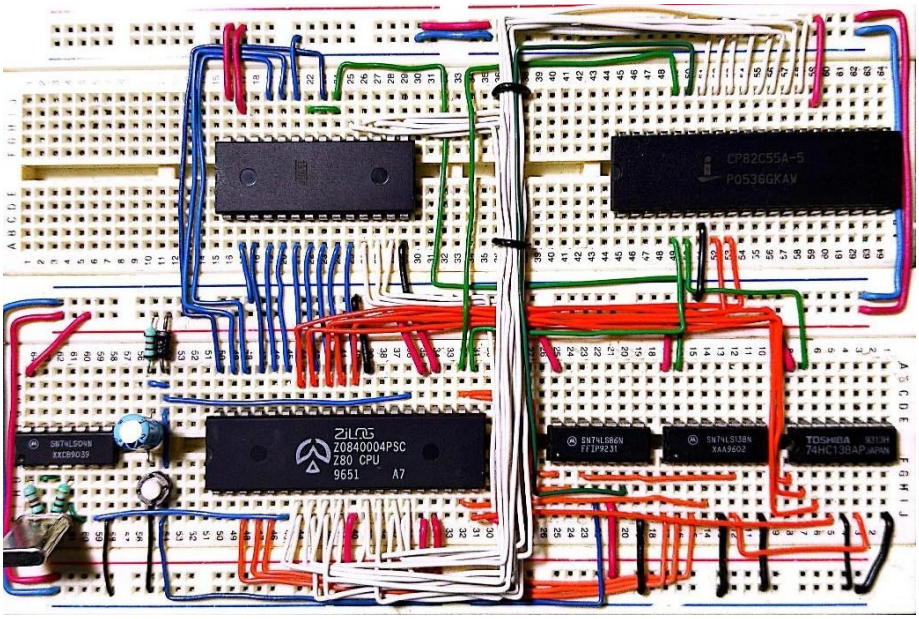



*Figura 4.8 Sistema mínimo con puertos de entrada – salida*

2. Edite el programa en lenguaje ensamblador que se muestra en la figura 4.9 y obtenga el código de máquina.

```
;Practica 4 Microprocesadores
;Programa para generar 3 patrones de bits

INICIO: LD      A,82H           ;Configurar el PPI, PA de salida
        OUT     (03H),A         ;PB de entrada y PC de salida modo 0

CHECAR: IN      A,(01H)         ;Seleccionar el patron a exhibir
        AND     07H
        CP      04H
        JR      Z,ENCENDER
        IN      A,(01H)
        AND     07H
        CP      02H
        JR      Z,BLINK
        IN      A,(01H)
        AND     07H
        CP      01H
        JR      Z,MOVER
        LD      A,00H
        OUT     (00H),A
        JP      CHECAR
```

*Figura 4.9 Programa para los patrones en el puerto A. (Parte 1).*

```
ENCENDER:
        LD      A,0FFH          ;Encender los leds permanentemente
        OUT     (00H),A
        JP      CHECAR

BLINK:  LD      A,0FFH          ;Parpadeo de leds
        OUT     (00H),A

        LD      DE,0002H        ;Consumo de tiempo
        LD      BC,0002H
CICLO1: LD      HL,0FFFFH
CICLO2: ADD     HL,BC
        JP      C,CICLO2
        EX      DE,HL
        ADD     HL,BC
        EX      DE,HL
        JP      C,CICLO1

        LD      A,00H
        OUT     (00H),A

        LD      DE,0002H
        LD      BC,0002H
CICLO3: LD      HL,0FFFFH
CICLO4: ADD     HL,BC
        JP      C,CICLO4
        EX      DE,HL
        ADD     HL,BC
        EX      DE,HL
        JP      C,CICLO3

        JP      CHECAR
MOVER:  LD      A,0C0H          ;Corrimiento de leds
        OUT     (00H),A

        LD      DE,0002H
        LD      BC,0002H
CICLO5: LD      HL,0FFFFH
CICLO6: ADD     HL,BC
        JP      C,CICLO6
        EX      DE,HL
        ADD     HL,BC
        EX      DE,HL
        JP      C,CICLO5

        LD      A,30H
        OUT     (00H),A

        LD      DE,0002H
        LD      BC,0002H
CICLO7: LD      HL,0FFFFH
CICLO8: ADD     HL,BC
        JP      C,CICLO8
        EX      DE,HL
        ADD     HL,BC
        EX      DE,HL
        JP      C,CICLO7

        LD      A,0CH
        OUT     (00H),A

        LD      DE,0002H
        LD      BC,0002H
CICLO9: LD      HL,0FFFFH
CICLO10: ADD    HL,BC
        JP      C,CICLO10
        EX      DE,HL
        ADD     HL,BC
        EX      DE,HL
        JP      C,CICLO9

        LD      A,03H
        OUT     (00H),A

        LD      DE,0002H
        LD      BC,0002H
CICLO11: LD     HL,0FFFFH
CICLO12: ADD    HL,BC
        JP      C,CICLO12
        EX      DE,HL
        ADD     HL,BC
        EX      DE,HL
        JP      C,CICLO11

        JP      CHECAR

        END
```

*Figura 4.9 Programa para los patrones en el puerto A. (Parte 2).*

| Circuito | VCC | GND |
|---|---|---|
| 74LS04 | 14 | 7 |
| Z80 CPU | 11 | 29 |
| AT28C64 | 28 | 14 |
| PPI 8255 | 26 | 7 |
| 74LS86 | 14 | 7 |
| 74LS138 | 16 | 8 |

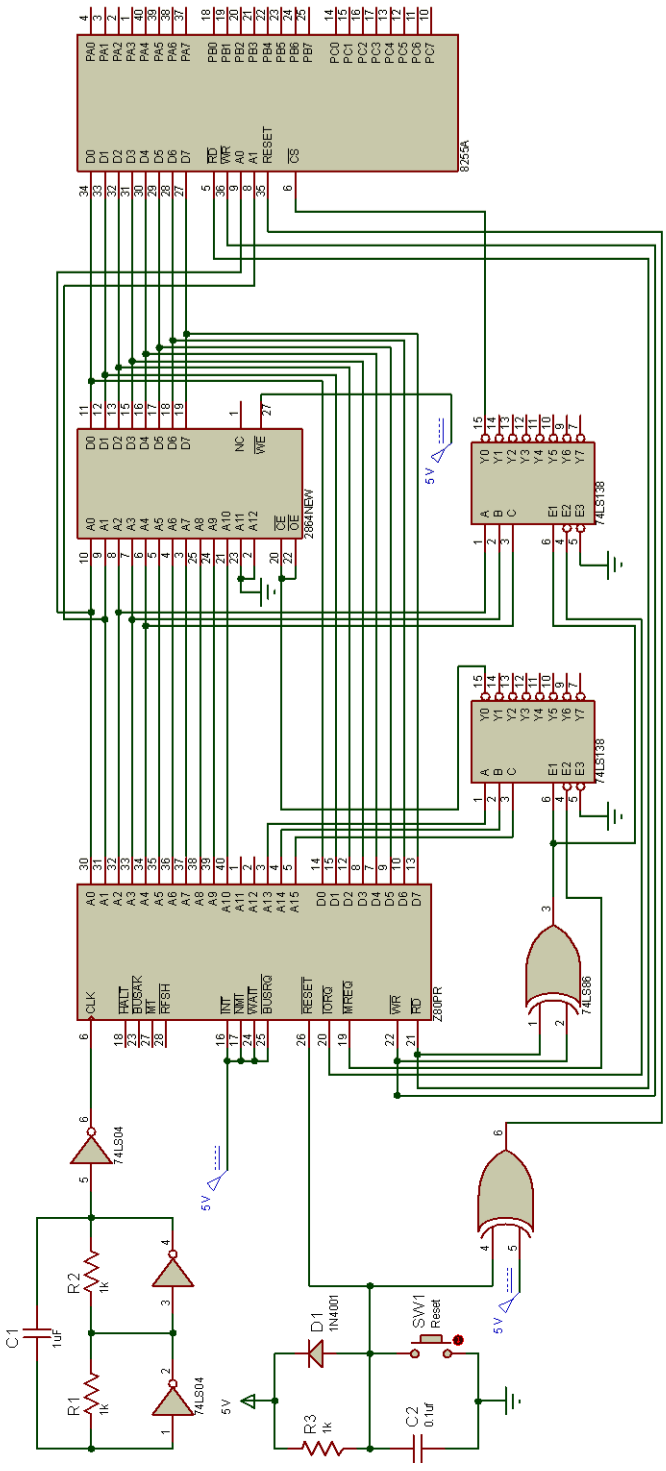



*Figura 4.10 Sistema mínimo con puertos de entrada y salida.*

3. Utilice la figura 4.8 para guiarse en la implementación física.
4. Utilice el capacitor de 4.7 µF en sustitución del cristal de cuarzo para proporcionar un reloj de baja velocidad.

5. Programe la memoria EEPROM en el programador universal con el archivo Prac04.hex.
6. Conecte el circuito probador de 8 leds en el puerto A del PPI 8255.
7. Inserte las señales que controlan la selección del patrón a mostrar en el puerto B con los bits PB2, PB1 y PB0 y compruebe el funcionamiento del sistema.

**Cuestionario**

1. Mida el periodo de reloj considerando el capacitor de 4.7 uF.
2. Calcule el tiempo requerido por el programa para ejecutar una vez el patrón 3 completo, tomando en cuenta el número de ciclos de reloj necesarios para cada una de las instrucciones.
3. Determine la palabra de control necesaria para configurar el PPI con puerto A bidireccional en modo 2 , puerto B de salida en modo 1 y los bits restantes del puerto C como entrada.
4. Explique el funcionamiento del programa empleado en este sistema.

# Laboratorio de Microprocesadores

## Práctica 5 Memoria SRAM o memoria de datos


**Tema**

5.4. Arreglos de memoria SRAM

**Objetivos**

- El alumno realizará la conexión de un circuito de memoria SRAM 6116 al sistema de microprocesador Z80
- El alumno creará un programa que utilice subrutinas y el almacenamiento de datos en la memoria SRAM.

**Introducción**

El siguiente elemento en el esquema de Von Neumann es la memoria SRAM o memoria de datos, la cual se emplea dentro del sistema mínimo para poder almacenar datos binarios que pueden ser empleados por el procesador para realizar sus operaciones o simplemente como almacenamiento de información.

Esta memoria debe realizar las funciones de lectura y escritura, es decir que debe ser memoria RWM (Read Write Memory) aunque típicamente se le conoce como memoria RAM, aun cuando el concepto RAM se refiere al método de acceso y no al tipo de operaciones posibles, la figura 5.1 nos muestra este elemento.

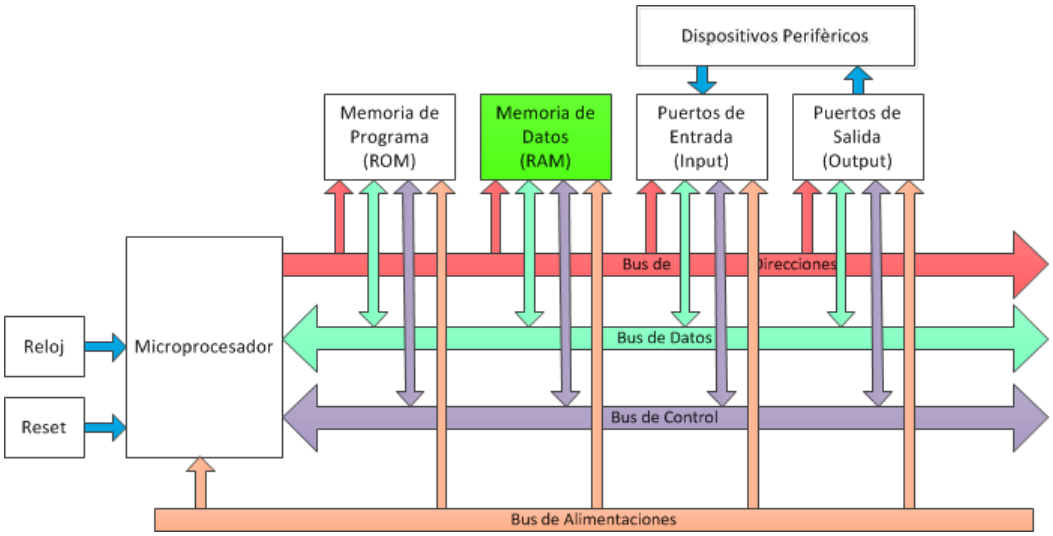



*Figura 5.1 Memoria RAM en el esquema de Von Neumann*

La memoria de datos se empleará para el almacenamiento de la información del usuario (variables, tablas, caracteres, etc.) y para el establecimiento de la pila o Stack, necesaria para la ejecución de las subrutinas que se puedan incluir en los programas en lenguaje ensamblador.

La SRAM tiene las características de ser una memoria volátil y estática y por lo tanto mantiene su información mientras el circuito esté alimentado eléctricamente sin requerir de refresco de memoria. Para la implementación de este sistema emplearemos memoria SRAM debido a que no requiere del circuito de refresco y por lo tanto el diseño es más simple.

Se utilizará una memoria SRAM 6116 con capacidad de 2k x 8 como la que se muestra en la figura 5.2.

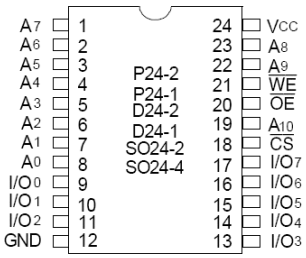



*Figura 5.2 Memoria SRAM 6116 de 2k x 8*

Esta conexión requiere la modificación de las funciones de Boole establecidas dentro del decodificador de direcciones para incluir la característica de activación de la memoria SRAM.

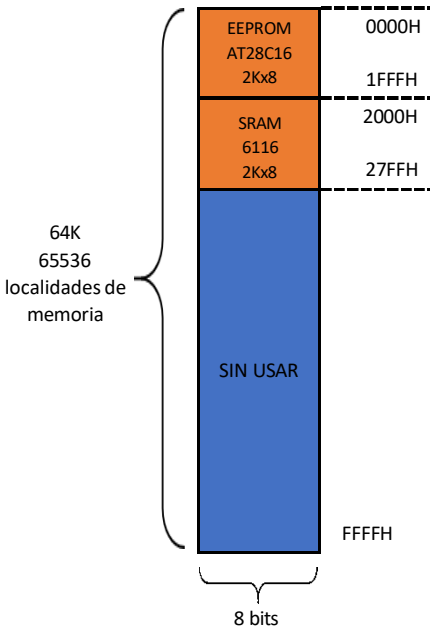



*Figura 5.3 Posición de la memoria SRAM dentro del mapa de memoria*

La dirección de memoria donde se localizará la SRAM debe seleccionarse por encima del espacio ocupado por la memoria EEPROM que fue conectada a partir de la localidad 0000H y hasta la dirección 1FFFH (8K x 8), la ubicación de la memoria SRAM se establecerá a partir de la dirección 2000H. En la figura 5.3 se muestra el mapa de memoria.

La señal de activación de la memoria SRAM se hará a través de un multiplexor decodificador 74LS138 y una compuerta 74LS86, la figura 6.4 muestra el circuito para la activación de la memoria.

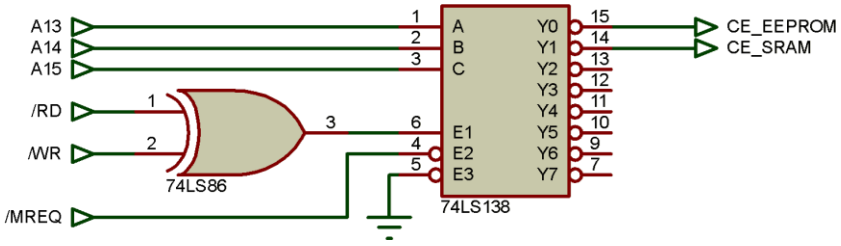



*Figura 5.4 Activación de la memoria EEPROM y SRAM*

Para realizar la prueba de funcionamiento de todo el sistema se utilizará un programa que generará un registro de corrimiento de 8 bits en el puerto A con espaciamiento de RETARDO2 entre cambio y cambio y una duración de encendido de cada led de RETARDO1 y que invierte el sentido del corrimiento de forma automática.

Para poder ajustar la duración de los cambios será necesario cambiar el capacitor del circuito de reloj, por un capacitor de 0.1uF.

Este programa permite la utilización de subrutinas debido a la conexión de la memoria SRAM y sobre este circuito ya se puede implementar la pila (stack), el registro apuntador de pila (SP) se establecerá a partir de la localidad 27FFH que es la última posición de la memoria SRAM.

Para configurar el PPI 8255 con todos sus puertos de salida en modo 0 se deberá escribir la palabra de control (80H) de 8 bits en el puerto 03H.

**Actividades Previas**

1. El alumno deberá realizar la lectura de la práctica de laboratorio.
2. El alumno editará el programa de la figura 5.5 en lenguaje ensamblador y obtendrá el código de máquina.

Assembler - Prac05.asm

File Edit Tools Options

```
INICIO: LD      SP,27FFH        ;Inicializa la dirección de la pila
SIGUE:  LD      A,80H           ;Palabra de control del PPI
        OUT     (03H),A         ;Configura el puerto de control

        LD      E,08H
        LD      A,01H           ;Enciende el bit 0 del acumulador
        LD      HL,2000H        ;Inicializa la localidad de la variable
        LD      (HL),A          ;Guarda el ultimo valor del registro
CICLO:  LD      A,(HL)          ;Recupera el valor desde la SRAM
        OUT     (00H),A         ;Saca el valor por el puerto
        CALL    RETARDO1        ;Consume RETARDO1
        LD      A,00H           ;Apaga todos los leds del puerto
        OUT     (00H),A
        CALL    RETARDO2        ;Consume RETARDO2
        RLC     (HL)            ;Rota el registro a la izquierda
        DEC     E
        JP      NZ,CICLO

        LD      E,08H
        LD      A,080H          ;Enciende el bit 8 del acumulador
        LD      HL,2000H        ;Inicializa la localidad de la variable
        LD      (HL),A          ;Guarda el ultimo valor del registro
CICLO2: LD      A,(HL)          ;Recupera el valor desde la SRAM
        OUT     (00H),A         ;Saca el valor por el puerto
        CALL    RETARDO1        ;Consume RETARDO1
        LD      A,00H           ;Apaga todos los leds del puerto
        OUT     (00H),A
        CALL    RETARDO2        ;Consume RETARDO2
        RRC     (HL)            ;Rota el registro a la derecha
        DEC     E
        JP      NZ,CICLO2
        JP      SIGUE
RETARDO1: LD    B,05H           ;Realiza el consumo de tiempo a traves
TIEMPOA: LD     D,020H          ;de un decremento anidado de 2 registros
TIEMPOB: DEC    D               ;B=05 y D=20
        JP      NZ,TIEMPOB
        DEC     B
        JP      NZ,TIEMPOA
        RET

RETARDO2: LD    C,04H           ;Se repite 5 veces (05H) la subrutina
SEG2:   CALL    RETARDO1        ;RETARDO1
        DEC     C
        JP      NZ,SEG2
        RET

        END
```

*Figura 5.5 Programa de registro de corrimiento*

3. El alumno programará la memoria AT28C64 con el código de máquina generado por el programa en lenguaje ensamblador mostrado en la figura 5.5.
4. Traer el circuito armado en la tableta de conexiones.

**Material**

Sistema mínimo con Microprocesador, circuito de reloj, circuito de reset, memoria EEPROM AT28C64, puertos, PPI 8255, sistema de decodificación.
1 Circuito Integrado SRAM 6116.
1 Capacitor de 0.1 uF
Alambres de conexión.

**Equipo**

Fuente de C.D.
Osciloscopio
Multímetro
Punta lógica para prueba de tercer estado
Circuito de Leds de prueba

**Procedimiento Experimental**

1. Implemente el circuito mostrado en la figura 5.7 considerando que ya se tiene armado el sistema mínimo de microprocesador incluyendo la memoria EEPROM y el PPI 8255 y solo deberá adicionarse el circuito integrado 6116 que proporciona la memoria SRAM.
2. Utilice la figura 5.6 para guiarse en la implementación física.

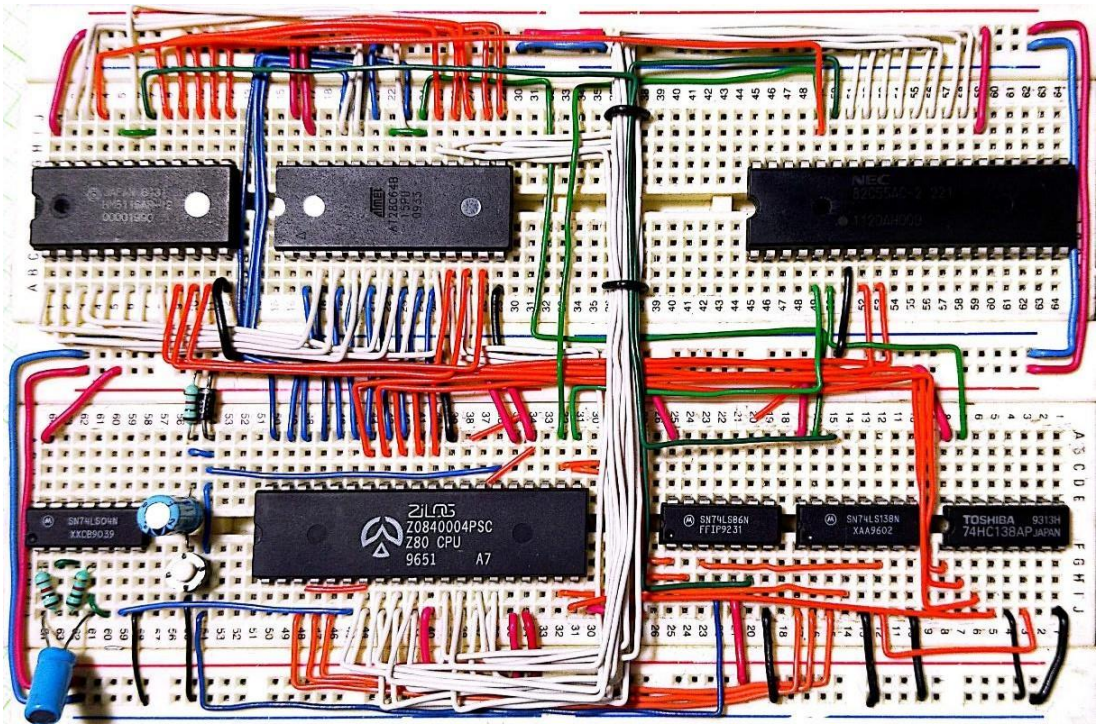


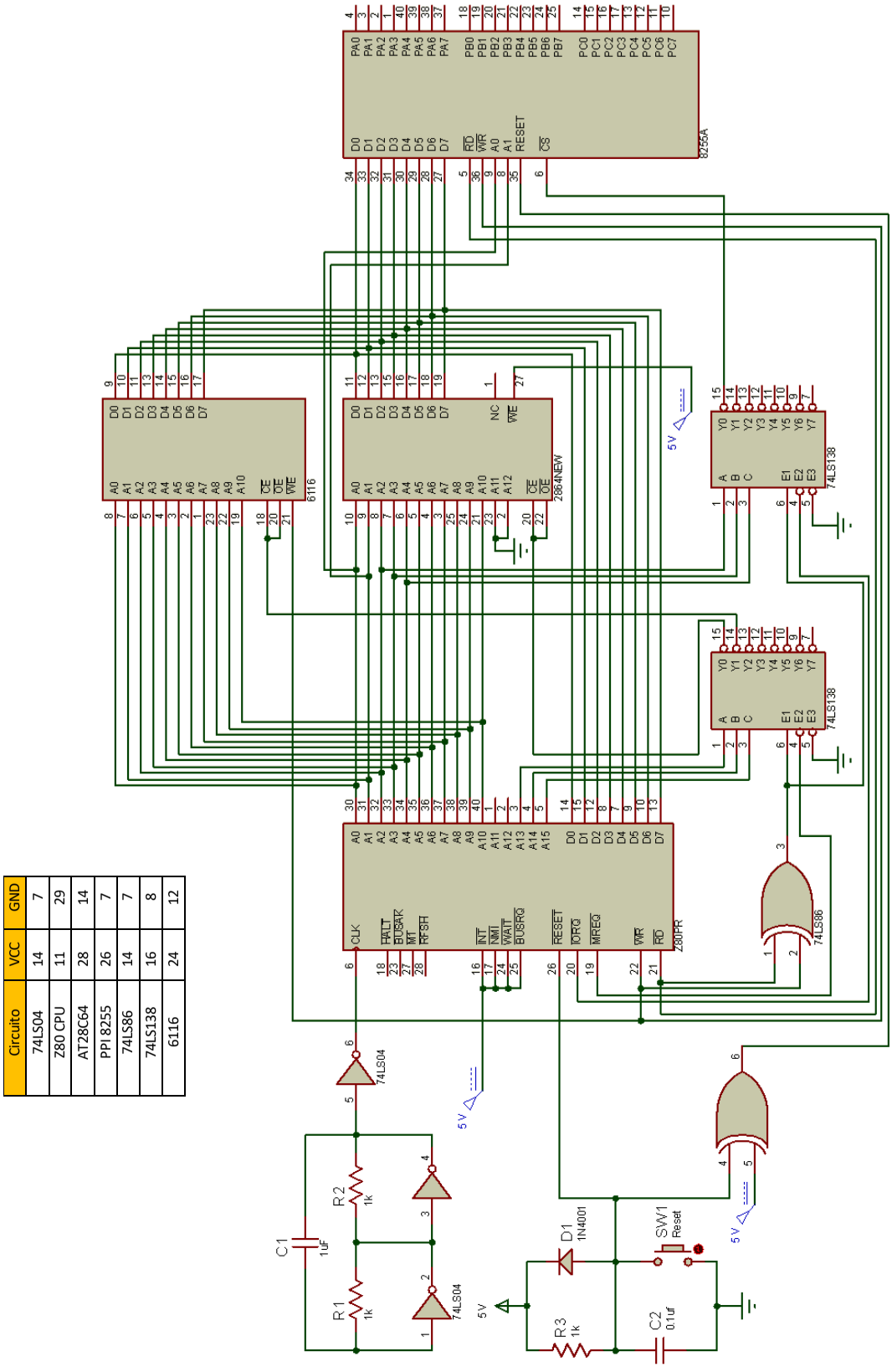

*Figura 5.6 Implementación de memoria SRAM.*

3. Compruebe el funcionamiento del sistema conectando 8 leds en las terminales del puerto A.

4. Mida con el osciloscopio alguna de las salidas del puerto A y anote el tiempo de encendido y el tiempo de apagado del led.

**Cuestionario**

1. Explique porque no es posible llamar una y otra vez en forma recursiva a las subrutinas con la estructura de la pila de este microprocesador.
2. Explique porque se establece la dirección inicial de la pila en la dirección final de la SRAM.
3. Justifique la expresión de Boole empleada para la activación de la SRAM.
4. Explique el concepto LIFO empleado en la pila del microprocesador Z80.
5. Determine el tiempo de encendido y apagado del led en forma teórica considerando el periodo de reloj aplicado y los ciclos T necesarios para realizar el RETARDO1 y el RETARDO2 .




| Circuito | VCC | GND |
|---|---|---|
| 74LS04 | 14 | 7 |
| Z80 CPU | 11 | 29 |
| AT28C64 | 28 | 14 |
| PPI 8255 | 26 | 7 |
| 74LS86 | 14 | 7 |
| 74LS138 | 16 | 8 |
| 6116 | 24 | 12 |

*Figura 5.7 Diagrama de conexiones del sistema mínimo*

# Laboratorio de Microprocesadores

## Práctica 6 Conexión de dispositivo periférico de entrada (Teclado)

**Tema**

6.0 Interfase con puertos de entrada – salida paralelos y serie

**Objetivos**

- El alumno realizará la conexión de un teclado matricial telefónico de 12 teclas al sistema de microprocesador.
- El alumno creará y probará un programa en lenguaje ensamblador para obtener el código binario de la tecla presionada.

**Introducción**

El circuito desarrollado hasta la práctica 5 ya conforma el esquema de Von Neumann con todos los elementos necesarios para la creación de una computadora, pero aún no proporciona interacción adecuada con el usuario externo. Para que el sistema tenga mayor utilidad es necesario conectarle dispositivos periféricos de entrada y salida para facilitar la interacción con la electrónica del microprocesador tal como se muestra en la figura 6.1.

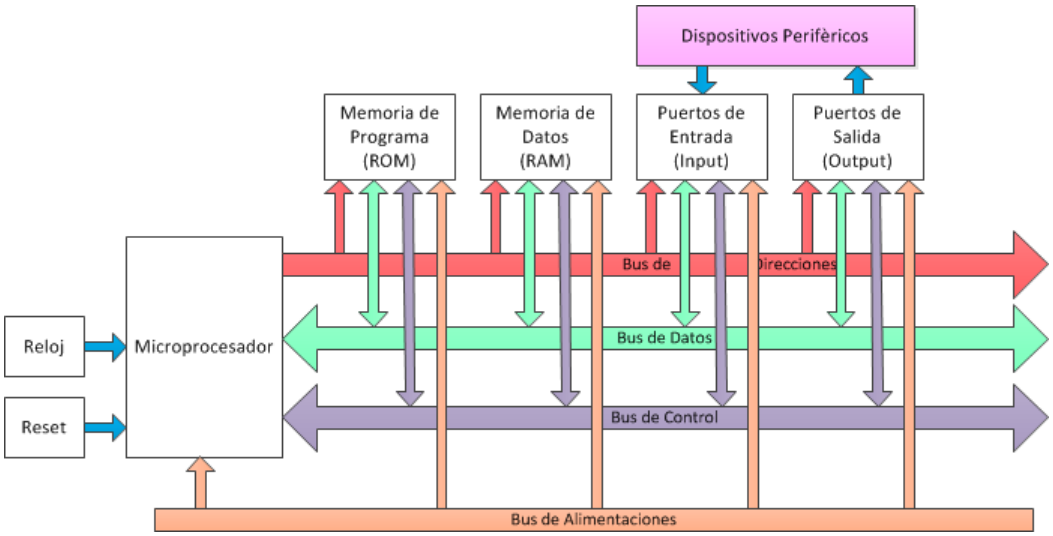



*Figura 6.1 Dispositivo periférico de entrada en Esquema de Von Neumann*

Los dispositivos periféricos de entrada se emplean para que el usuario externo pueda interactuar con el sistema de microprocesador y facilitar la inserción de datos externos, entre estos dispositivos se encuentran los teclados, los dispositivos apuntadores como Mouse, Track Ball, Touch Screen, tarjetas de red, audio o video, discos duros, CD, o DVD y en general cualquier dispositivo que permita enviar datos hacia el microprocesador.

En esta práctica se propone adicionar un teclado matricial de 12 teclas como el mostrado en la figura 6.2, que nos permitirá insertar datos numéricos o aún alfanuméricos si seleccionamos otro tipo de teclado.


*Figura 6.2 Teclado telefónico matricial de 12 teclas*

Este teclado cuenta con 9 terminales como se observa en la figura 6.2, cuatro dedicadas a los renglones (gris, violeta, azul y amarillo), 3 dedicados a las columnas (naranja, rojo y verde), una tierra (café) y un cable que no tiene conexión (negro).

Para utilizarlo en el sistema de microprocesador se realizará la conexión mostrada en la figura 6.3, la cual proporciona un cero lógico en el renglón y en la columna donde se localice la tecla presionada.

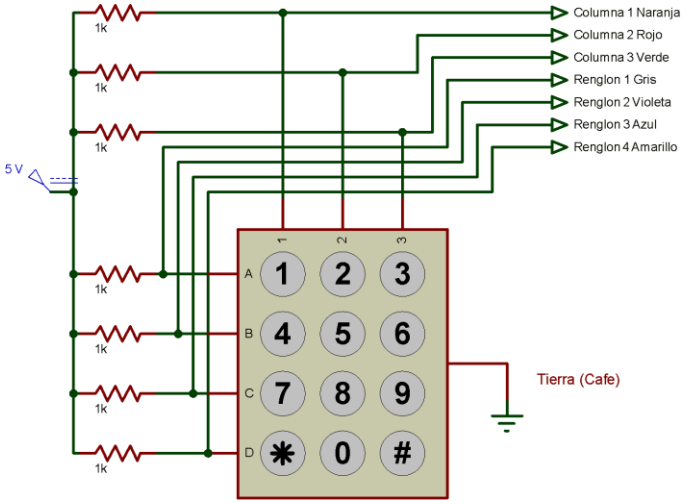



*Figura 6.3 Conexión de teclado telefónico*

Las conexiones hacia el sistema de microprocesador se indican en la tabla 6.1.

| Teclado | Color | Puerto C | PIN PPI |
|---|---|---|---|
| Columna 1 | Naranja | C6 | 11 |
| Columna 2 | Rojo | C5 | 12 |
| Columna 3 | Verde | C4 | 13 |
| Renglón 1 | Gris | C3 | 17 |
| Renglón 2 | Violeta | C2 | 16 |
| Renglón 3 | Azul | C1 | 15 |
| Renglón 4 | Amarillo | C0 | 14 |
| Tierra | Café | - | Tierra |

*Tabla 6.1 Asignación de terminales del PPI*

Los códigos generados por el teclado en sus siete terminales se insertarán en los 7 bits menos significativos del puerto C del PPI (C6 a C0) y el bit C7 permanece sin conexión. Por lo que los códigos se codifican de la manera indicada en la tabla 6.2.

| Puerto C<br>C7 C6 C5 C4 C3 C2 C1 C0 | Valor hexadecimal | Tecla |
|---|---|---|
| X 0 1 1 0 1 1 1 | 37 | 1 |
| X 1 0 1 0 1 1 1 | 57 | 2 |
| X 1 1 0 0 1 1 1 | 67 | 3 |
| X 0 1 1 1 0 1 1 | 3B | 4 |
| X 1 0 1 1 0 1 1 | 5B | 5 |
| X 1 1 0 1 0 1 1 | 6B | 6 |
| X 0 1 1 1 1 0 1 | 3D | 7 |
| X 1 0 1 1 1 0 1 | 5D | 8 |
| X 1 1 0 1 1 0 1 | 6D | 9 |
| X 0 1 1 1 1 1 0 | 3E | * |
| X 1 0 1 1 1 1 0 | 5E | 0 |
| X 1 1 0 1 1 1 0 | 6E | # |

*Tabla 6.2 Codificación de teclas*

El valor de la tecla correspondiente se mostrará en los 4 bits menos significativos del puerto A.

Considerar que los teclados de membrana no cuentan con una terminal a tierra y por lo tanto no es posible conectarlos empleando el método indicado en esta práctica.

**Actividades Previas**

1. El alumno deberá realizar la lectura de la práctica de laboratorio.
2. El alumno programará la memoria AT28C64 con el código de máquina generado por el programa en lenguaje ensamblador mostrado en la figura 6.5.
3. Traer el circuito armado en la tableta de conexiones.

**Material**

1 Sistema mínimo con microprocesador Z80, circuito de reloj, circuito de reset, memoria EEPROM AT28C16, memoria SRAM 6116 y circuito de puertos PPI 8255.
7 Resistencias de 1kΩ a ½ W.

**Equipo**

Teclado telefónico matricial de 12 teclas (proporcionado por el laboratorio)
Fuente de C.D.
Osciloscopio
Multímetro
Punta lógica para prueba de tercer estado
Circuito de Leds de prueba

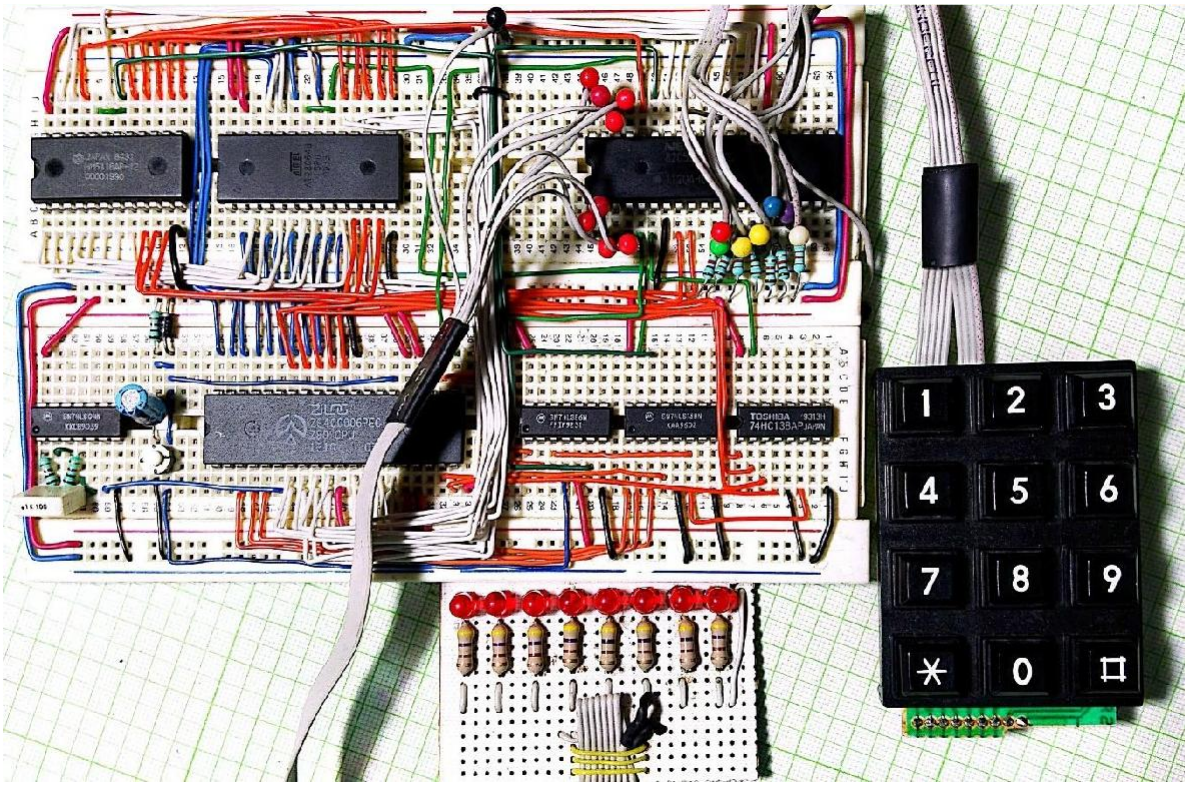



*Figura 6.4 Sistema mínimo con decodificador de teclado*

**Procedimiento Experimental**

1. Implemente el circuito mostrado en la figura 6.6 considerando que ya se tiene armado el sistema mínimo de Von Neumann y solo deberán adicionarse las resistencias de 1k del teclado y las conexiones del puerto C y la salida de leds en el puerto A.
2. Utilice la figura 6.4 para establecer la posición de los circuitos y sus conexiones.
3. Edite el programa en lenguaje ensamblador de la figura 6.5 y obtenga el código de máquina. Este programa decodificará el número de tecla presionado y lo mostrará en los 4 bits menos significativos del puerto A (00H).

Assembler - Prac06.asm

File Edit Tools Options

```
        ORG     0000H           ;Define la direccion de inicio

INICIO: LD      SP,27FFH        ;Inicia la localidad de la pila
        LD      A,089H          ;PA = Salida, PB = Salida y PC = Entrada
        OUT     (03H),A         ;Envia al puerto de control
        LD      A,00H           ;Limpia la localidad 2000H
        LD      (2000H),A
        CALL    CHECA           ;Checa el codigo generado por el teclado
CICLO:  LD      A,(2000H)       ;Carga en el acumulador el ultimo dato almacenado
        OUT     (00H),A         ;Saca el dato al puerto A
        CALL    CHECA
        JP      CICLO           ;Repite el ciclo de forma infinita

CHECA:  IN      A,(02H)         ;Inserta el dato del teclado desde el puerto C
        AND     7FH
        CP      7FH             ;Limpia los 7 bits
        RET     Z               ;Si no hay tecla presionada deja el dato anterior
        LD      H,00H
        LD      L,A
        LD      A,(HL)          ;Accede a la localida sobre la tabla
        LD      (2000H),A       ;Guarda el dato en la memoria
        RET
```

*Figura 6.5 Programa para control de teclado (Parte 1)*

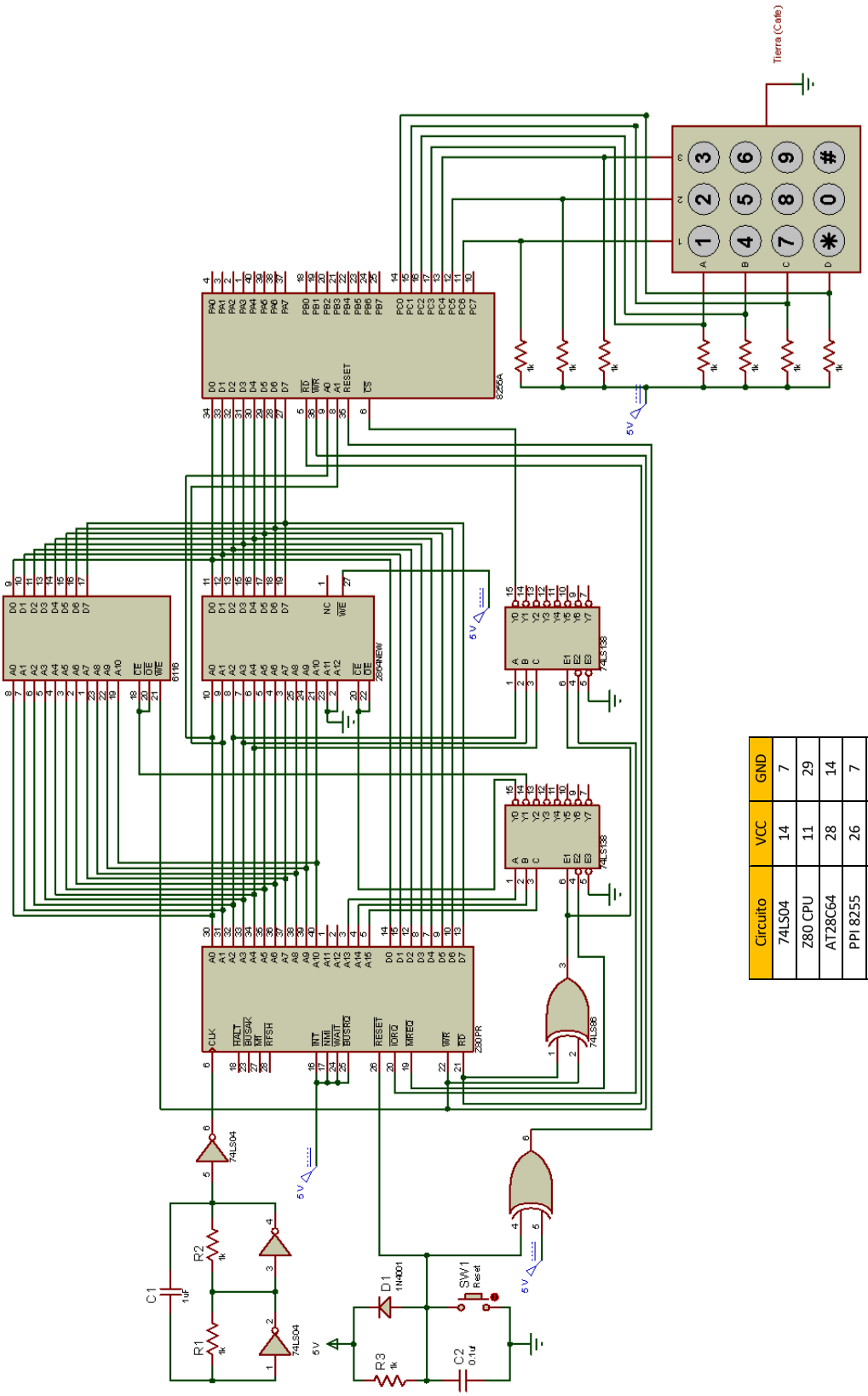
```
        ORG     37H
        DB      01H             ;Tabla de asignacion de valores para cada tecla ORG
        ORG     57H
        DB      02H
        ORG     67H
        DB      03H
        ORG     3BH
        DB      04H
        ORG     5BH
        DB      05H
        ORG     6BH
        DB      06H
        ORG     3DH
        DB      07H
        ORG     5DH
        DB      08H
        ORG     6DH
        DB      09H
        ORG     3EH
        DB      0AH
        ORG     5EH
        DB      00H
        ORG     6EH
        DB      0BH
        END
```

*Figura 6.5 Programa para control de teclado (Parte 2)*

4. Programe la memoria EEPROM AT28C64 en el programador universal con el archivo Prac06.hex.
5. Compruebe que el código binario mostrado en las terminales de los 4 bits menos significativos del puerto A corresponden con la tecla presionada.

**Cuestionario**

1. Justifique el valor del byte de configuración enviado al PPI en su puerto 03H.
2. Investigue algún circuito integrado que se utilice para la decodificación de teclado y explique su funcionamiento incluyendo algún diagrama que defina su comportamiento.




| Circuito | VCC | GND |
|---|---|---|
| 74LS04 | 14 | 7 |
| Z80 CPU | 11 | 29 |
| AT28C64 | 28 | 14 |
| PPI 8255 | 26 | 7 |
| SRAM 6116 | 24 | 12 |
| 74LS86 | 14 | 7 |
| 74LS138 | 16 | 8 |

*Figura 6.6 Sistema mínimo con teclado*

# Laboratorio de Microprocesadores

## Práctica 7 Conexión de dispositivo periférico de salida display LCD

**Tema**

7.6. Programación para control de dispositivos periféricos.

**Objetivos**

- El alumno realizará la conexión de una pantalla LCD de 16 caracteres por 2 líneas al sistema de microprocesador.
- El alumno creará y probará un programa en lenguaje ensamblador para desplegar y controlar un mensaje sobre la pantalla.

**Introducción**

A través de esta práctica el alumno le proporcionará al sistema mínimo la capacidad de desplegar mensajes hacia el usuario e incrementará la interacción entre el microprocesador y el mundo exterior.

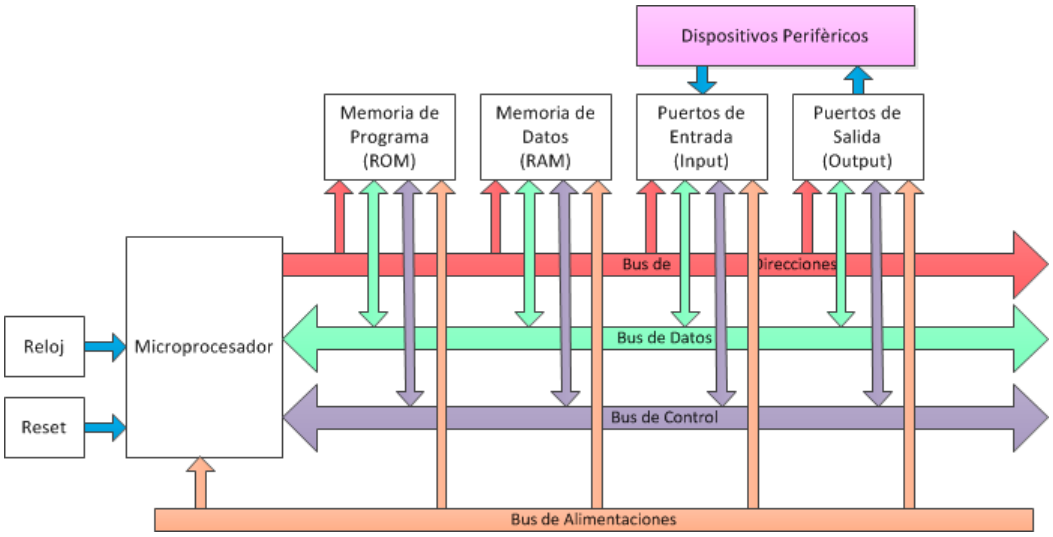



*Figura 7.1 Dispositivo periférico de salida en esquema de Von Neumann*

Los dispositivos periféricos de salida se emplean para que el microprocesador pueda enviar información hacia el usuario externo. Entre los dispositivos que se emplean como salidas están: los monitores, las impresoras, los discos duros, los CD, los DVD, las tarjetas de audio, video o redes y en general cualquier dispositivo que pueda ser utilizado para visualizar o manipular la información del sistema mínimo.

Es por eso por lo que en esta práctica se propone adicionar una pantalla inteligente de cristal líquido (LCD) de 16 caracteres x 2 líneas que recibe códigos en caracteres ASCII y que convierte el código a un mapa de pixeles que se puede desplegar sobre cada uno de los 16 caracteres de la pantalla o almacenarlo en la memoria interna del display (DDRAM).

*Figura 7.2 Pantalla LCD de 16 x 2*

Esta pantalla puede recibir ya sea comandos de configuración para controlar el funcionamiento de la pantalla o datos de 8 bits que representan los caracteres ASCII a desplegar. Estos comandos se muestran en la tabla 7.1.

| Instrucciones | RS | R/W | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Descripción | Tiempo de Ejecución |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Limpia display | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Limpia el display completamente y pone la dirección 0 en el contador de dirección | 1.28 ms |
| Regreso al origen | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | - | Pone a 0 el contador de la DDRAM. La DDRAM mantiene su contenido sin cambio | 1.28 ms |
| Modo de Inserción de datos | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | I/D | S | Establece la dirección de movimiento y especifica si se corre el cursor o el display | 31 us |
| Control del display | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B | Enciende o apaga el display, el cursor y el parpadeo | 31 us |
| Corrimiento del display o del cursor | 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | - | - | Mueve el cursor o el display a la izquierda o a la derecha sin cambios en la DDRAM | 31 us |
| Formato de datos | 0 | 0 | 0 | 0 | 1 | DL | N | F | - | - | Establece la longitud de los datos, el número de líneas y el formato del carácter | 31 us |
| Dirección de la CGRAM | 0 | 0 | 0 | 1 | AGC | AGC | AGC | AGC | AGC | AGC | Establece la dirección de la memoria de usuario CGRAM | 31 us |
| Dirección de la DDRAM | 0 | 0 | 1 | ADD | ADD | ADD | ADD | ADD | ADD | ADD | Establece la dirección de la memoria de datos DDRAM | 31 us |
| Bandera de ocupado | 0 | 1 | BF | AC | AC | AC | AC | AC | AC | AC | Lee la bandera de ocupado y la dirección del contador | 31 us |
| Escribir un dato en la memoria | 1 | 0 | Dato a cargar en la memoria | | | | | | | | Escribe un dato dentro de la DDRAM o de la CGRAM | 31 us |
| Leer un dato de la memoria | 1 | 1 | Dato leído de la memoria | | | | | | | | Lee un dato de la DDRAM o de la CGRAM | 31 us |

*Tabla 7.1 Comandos del display LCD 16 x 2*

Para la conexión de esta pantalla LCD se utilizarán 2 puertos del circuito PPI 8255:

- El puerto A de 8 bits para enviar al display el código ASCII a mostrar en la pantalla o el comando de configuración del display.
- 2 bits del puerto B para enviar las señales de control, habilitación del display (E) y selección de comando o dato (RS), la señal R/W permanecerá todo el tiempo a cero lógico.

Este display inteligente tiene 16 terminales como se muestra en la figura 7.3, las cuales se utilizan para controlar al dispositivo y para enviar la información a desplegar.

En la tabla 7.2 se menciona la función de cada una de las terminales del display.


*Figura 7.3 Asignación de las terminales del display*

| Terminal | Señal | Descripción |
|---|---|---|
| 1 | Vss | Tierra |
| 2 | Vcc | + 5 V.c.d. |
| 3 | Vee | Voltaje de control de contraste<br>0 V. = Contraste mínimo |
| 4 | RS | Señal de Comando RS =0 ó Dato RS = 1 |
| 5 | R/W | Señal de Lectura R/W = 1 ó Escritura R/W =0 |
| 6 | E | Señal de habilitación del display E = 1 |
| 7 - 14 | D0 – D7 | Bus de datos |
| 15 | LD+ | Positivo del led de iluminación trasera |
| 16 | LD- | Negativo del led de iluminación trasera |

*Tabla 7.2 Asignación de terminales del display*

Debido a que el display se empleará siempre en modo de escritura, el bit de R/W se conectará directamente a tierra. La asignación de los bits de los puertos se hará como se muestra en la tabla 7.3.

| Terminal | Señal | Puerto - bit |
|---|---|---|
| 4 | RS | PB1 |
| 5 | R/W | Tierra |
| 6 | E | PB0 |
| 7 | D0 | PA0 |
| 8 | D1 | PA1 |
| 9 | D2 | PA2 |
| 10 | D3 | PA3 |
| 11 | D4 | PA4 |
| 12 | D5 | PA5 |
| 13 | D6 | PA6 |
| 14 | D7 | PA7 |

*Tabla 7.3 Asignación de terminales del display a los puertos del PPI*

**Actividades Previas**

1. El alumno deberá realizar la lectura de la práctica de laboratorio.
2. El alumno programará la memoria AT28C64 con el código de máquina generado a partir del programa en lenguaje ensamblador mostrado en la figura 7.4.
3. Modifique la programación de la memoria para que la fecha del semestre se adicione al final de la palabra ITSE (2025-1).
4. Modifique el programa original para hacer que a través del teclado y la tecla de asterisco (*) el sistema modifique el sentido de desplazamiento del letrero, ya sea hacia la derecha o hacia la izquierda.
5. Traer el circuito armado

**Material**

1 Sistema mínimo con microprocesador Z80, circuito de reloj, circuito de reset, memoria EEPROM AT28C64 y circuito de puertos PPI 8255.
1 Display LCD 16 x 2

**Equipo**

Fuente de C.D.
Osciloscopio
Multímetro
Punta lógica para prueba de tercer estado
Circuito de Leds de prueba

**Procedimiento Experimental**

1. Implemente el circuito mostrado en la figura 7.7 considerando que ya se tiene armado el sistema mínimo de Von Neumann y el teclado y solo deberá adicionarse la pantalla LCD con sus conexiones de alimentaciones, bus de datos y señales de control.
2. Edite el programa en lenguaje ensamblador de la figura 7.4 (parte 1 y parte2), obtenga el código de máquina.
3. Programe la memoria EEPROM.

Assembler - Prac07.asm

File Edit Tools Options

```
        ORG     0000h           ;Origen del reset
        JP      0100H           ;Direccion del programa principal
;*************************
        ORG     0100H           ;Programa principal
INICIO: LD      SP,27FFH        ;Direccion de la pila
        LD      A,089H          ;PA = Salida, PB = Salida, PC = Entrada
        OUT     (03H),A         ;Configura el PPI
;*************************
CONFIG: LD      HL,0200H        ;Posicion de la tabla de comandos
OTROCOM: LD     A,(HL)          ;Recupera valor de comando
        CP      '$'             ;Checa si es el fin de la tabla
        JP      Z,LINEA1        ;Salta a ecribir en la primera linea
        OUT     (00H),A         ;Escribe el comando en el puerto A
        CALL    COMANDO         ;Ejecuta la subrutina para envio al LCD
        INC     HL              ;Incrementa el indice de la tabla
        JP      OTROCOM         ;Ejecuta el siguiente comando
;*************************
LINEA1: LD      HL,0250H        ;Posicion del primer mensaje
        CALL    OTRALET         ;Checa si hay mas letras
LINEA2: LD      A,0C0H          ;Cambia a segunda linea del display
        OUT     (00H),A         ;Envia el comando
        CALL    COMANDO         ;Ejecuta la subrutina para envio al LCD
;*************************
        LD      HL,0260H        ;Posicion del segundo mensaje
        CALL    OTRALET         ;Checa si hay mas letras
;*************************
FIN:    LD      A,18H           ;Mueve el letrero hacia la izquierda
        OUT     (00H),A         ;Envia el comando al LCD
        CALL    COMANDO         ;Ejecuta la subrutina para envio al LCD
        JP      FIN             ;Realiza un ciclo infinito
;*************************
OTRALET: LD     A,(HL)          ;Lee el dato de la memoria
        CP      '$'             ;Checa si es el final de la cadena
        RET     Z               ; Si es el final termina la subrutina
        OUT     (00H),A         ;Envia el comando hacia el LCD
        CALL    DATO            ;Ejecuta la subrutina de envio de caracter
        INC     HL              ;Incrementa el indice de la tabla
        JP      OTRALET         ;Repite el proceso letra por letra
;*************************
COMANDO: LD     A,00H           ;Activa RS=0 y E=0
        OUT     (01H),A
        LD      A,01H           ;Activa RS=0 y E=1
        OUT     (01H),A
        LD      A,00H           ;Activa RS=0 y E=0
        OUT     (01H),A
        CALL    TIEMPO          ;Consume 10 ms de tiempo
        RET
;*************************
DATO:   LD      A,02H           ;Activa RS=1 y E=0
        OUT     (01H),A
        LD      A,03H           ;Activa RS=1 y E=1
        OUT     (01H),A
        LD      A,02H           ;Activa RS=1 y E=0
        OUT     (01H),A
        CALL    TIEMPO          ;Consume 10 ms de tiempo
        RET
;*************************
```

*Figura 7.4 Programa para control y despliegue de mensaje ( Parte 1)*

```
TIEMPO: LD      A,05H           ;Decrementa 2 registros en
CICLO2: LD      B,05H           ;forma recursiva
CICLO:  DEC     B               ;Decrementa 255 veces el
        JP      NZ,CICLO        ;numero 255
        DEC     A
        JP      NZ,CICLO2
        RET
;*************************
        ORG     0200H           ;Direccion de tabla de comandos
        DB      01H             ;Limpia el display
        DB      02H             ;Regreso al origen
        DB      06H             ;Insercion de datos con incremento
                                ;y display fijo
        DB      0FH             ;Display,cursor y parpadeo encendido
        DB      38H             ;8 bits, 2 lineas y caracteres de 5x7
        DB      080H            ;Primer caracter de linea 1
        DB      '$'             ;Indicador de fin de tabla
;*************************
        ORG     0250H           ;Tabla de caracteres de linea 1
        DB      "Laboratorio de$"
        ORG     0260H           ;Tabla de caracteres de linea 2
        DB      "Microprocesadore"
        ORG     0270H           ;continuacion de linea 2
        DB      "s ITSE$"
        END
```

*Figura 7.4 Programa para control y despliegue de mensaje ( Parte 2)*

4. Este programa escribirá el mensaje “Laboratorio de” en la primera línea y “Microprocesadores ITSE” en la segunda línea, considere que el mensaje de la segunda línea es más grande que el tamaño del display que es de 16 caracteres y por lo tanto eso indica que los caracteres restantes se escriben en la memoria del display y no se pueden ver de forma directa tal y como se muestra en las figuras 7.5 y 7.6.

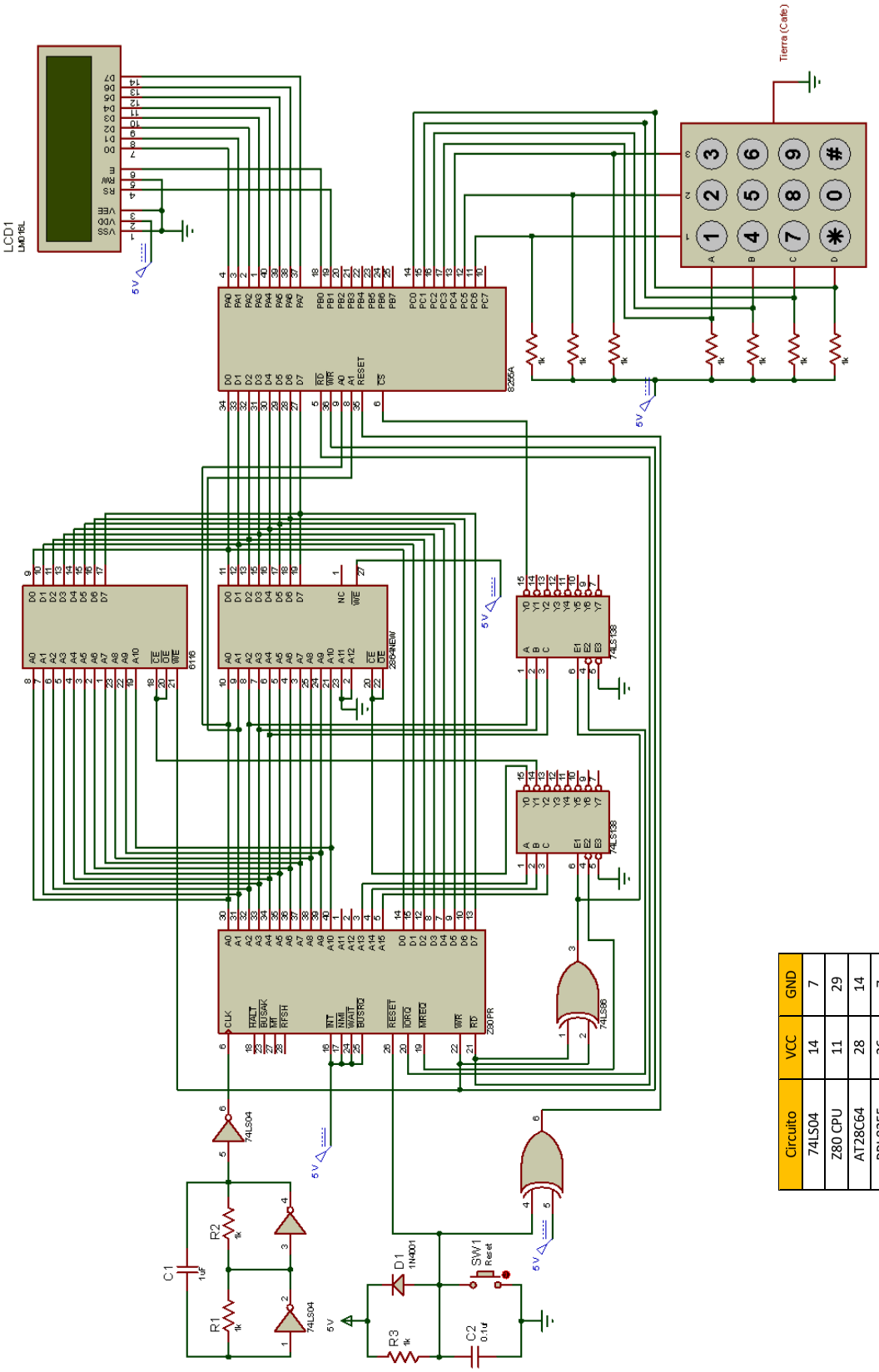

*Figura 7.5 Mensaje Parte 1*


*Figura 7.6 Mensaje Parte 2*

5. Compruebe que se despliega correctamente el mensaje en el display de acuerdo a la figura 7.5 y que después se desplaza el mensaje sobre el display como en la figura 7.6.

**Cuestionario**

1. Describa el procedimiento para programar el display usando datos de 4 bits en lugar de datos de 8 bits.
2. Porque es necesario correr una subrutina de consumo de tiempo después del envío de cada comando o dato.
3. ¿Existe alguna otra forma de checar que el display está libre para ejecutar el siguiente comando?




| Circuito | VCC | GND |
|---|---|---|
| 74LS04 | 14 | 7 |
| Z80 CPU | 11 | 29 |
| AT28C64 | 28 | 14 |
| PPI 8255 | 26 | 7 |
| SRAM 6116 | 24 | 12 |
| 74LS86 | 14 | 7 |
| 74LS138 | 16 | 8 |

*Figura 7.7 Sistema Mínimo con Display LCD*

# Laboratorio de Microprocesadores

## Práctica 8 Entorno de Desarrollo


**Tema**

8.5. Tecnologías de última generación en microprocesadores.

**Objetivos**

- Instalar el entorno de desarrollo que nos permita escribir código, depurar programas y programar el procesador; tanto en lenguajes ensamblador como en C y C++.
- Escribir un programa en ensamblador para probar el entorno.
- Probar el programa, mediante la simulación del procesador.

**Introducción**

Un microprocesador (o procesador) es esencialmente una CPU (Central Processing Unit). La unidad central de procesamiento es el componente funcional principal de una computadora; está compuesta por la unidad de control, registros de propósito general y específico, canales de intercomunicación (buses) y una unidad encargada de las operaciones aritméticas y lógicas (ALU ).

Desde un punto de vista didáctico, el diseño de sistemas basados en microprocesadores puede ser abordado utilizando microcontroladores, pues internamente un microcontrolador cuenta con una Unidad Central de Procesamiento (CPU o microprocesador). A su vez, de esta manera se puede acceder a herramientas hardware y software de bajo costo, indispensables en la etapa iterativa del diseño.
La arquitectura de un microprocesador se refiere a diversas metodologías de diseño y organización de los componentes de la CPU. Otro factor importante es la arquitectura del conjunto de instrucciones (ISA) que el procesador puede realizar. Las arquitecturas de computadora de conjunto de instrucciones reducidas (RISC ) y computadora de conjunto de instrucciones complejas (CISC ), proporcionan varios métodos para el procesamiento de datos, ofreciendo diferentes niveles de rendimiento, confiabilidad y velocidad dependiendo de la aplicación.

Los microprocesadores varían en potencia, rendimiento, metodologías de arquitectura, tamaño, consumo de energía y muchas otras variables. Los microprocesadores de uso general son habituales en computadoras personales y dispositivos móviles, mientras que las unidades especializadas de alto rendimiento se diseñan para tareas exigentes. Los siguientes son algunos de los principales tipos de microprocesadores:

- Microprocesadores de uso general,
- Microcontroladores,
- Procesadores de señales digitales (DSP),
- Unidades de procesamiento de gráficos (GPU),
- Procesadores de red,
- Coprocesadores.

Las dos empresas más importantes en este campo son Intel y AMD (Advanced Micro Devices). Cada una utiliza un tipo diferente de arquitectura de conjunto de instrucciones (ISA). Los procesadores Intel utilizan una arquitectura de conjunto de instrucciones complejo (CISC ), mientras que los procesadores AMD siguen una arquitectura de conjunto de instrucciones reducido (RISC ).

Otro jugador importante es arm (Advanced RISC Machine). Aunque no fabrica equipos, sí alquila sus diseños de procesadores de alta gama y otras tecnologías patentadas a otras empresas que sí fabrican equipos. Apple, por ejemplo, ya no utiliza chips Intel en sus computadoras, sino que fabrica sus propios procesadores personalizados basados en diseños arm.
Es una familia de arquitecturas de conjunto de instrucciones (ISA) RISC para procesadores, diseñada por Arm Holdings, ampliamente utilizada por su arquitectura eficiente. Tiene cuatro líneas de unidades centrales de procesamiento:

- Cortex (procesadores de 32/64-bit),
- Neoverse (procesadores de 64-bit para cómputo de alto rendimiento),
- GPUs (procesadores en tiempo real),
- NPUs (Microcontroladores).

Existen muchas revisiones de arquitectura arm diferentes (ARMv6, ATMv6-M, ARMv7-M, ARMv7-A, ARMv8-M, etc.) y muchas arquitecturas de núcleo (Cortex-M3, Cortex-M1, Cortex-M0, Cortex-M4, Cortex-M0+, etc.).
La arquitectura sigue evolucionando de tal manera que, por ejemplo, la versión siete (ARMv7) define tres perfiles de arquitectura:

- ARM Cortex serie Ax.
- ARM Cortex serie Rx.
- ARM Cortex serie Mx,

Existen muchas empresas que son titulares de licencias ARM, entre las más conocidas están: Samsung, Apple Inc., Microchip, Broadcom, STMicroelectronics, LG, MediaTek, Qualcomm, Texas Instruments. Las empresas fabrican chips de acuerdo con alguna revisión y núcleo de arquitectura.
Por ejemplo, STMicroelectronics [1] tiene una línea de procesadores Cortex-M para aplicaciones de alto desempeño, los más usados, bajo consumo de energía, aplicaciones inalámbricas y otros (figura 8.1).


*Figura 8.1: Procesadores Arm Cortex-M de STMicroelectronics.*

Considerando el costo, el tamaño, las características y la disponibilidad, para este manual de prácticas se ha elegido el STM32F103c81, que es un procesador basado en el núcleo Cortex-M3 diseñado sobre la arquitectura ARMv7-M y una arquitectura clásica Harvard.

Además de la elección del hardware para la creación de sistemas integrados, se requiere del proceso de desarrollo de firmware2, que implica una serie de pasos, herramientas y tecnologías para desarrollar productos desde su inicio hasta la producción. El propósito de una cadena de herramientas de software es tenerlas vinculadas y optimizadas, de tal manera que la salida generada por una herramienta es utilizada por la siguiente herramienta como entrada. Las cadenas de herramientas básicas suelen incluir: ensamblador, enlazador, depurador, compilador, bibliotecas. Estas herramientas pueden estar incluidas en un solo entorno de desarrollo o ser ejecutadas de manera individual por el desarrollador. Si bien hay una variedad de entornos de desarrollo integrados (IDE) disponibles para STM32, los más utilizados son STM32CubeIDE de STMicroelectronics, EWARM de IAR y µVision (MDK) de Keil. Este último será utilizado durante el desarrollo para estas prácticas.

*Debido a que existen clones, verificar que el chip tenga el logo de ST .*
*Es el software que tiene directa interacción con el hardware, que reside en la memoria no volátil.*

**Actividades Previas**

1. Leer la explicación introductoria al contenido y alcance de esta práctica.
2. Leer detenidamente y comprender cada una de las secciones de esta práctica.
3. Instalar el entorno de desarrollo siguiendo las indicaciones descritas en este enlace. Y, una vez completada la instalación, activar la licencia siguiendo estas otras indicaciones.

**Material**

- No se requiere.

**Equipo**

- Computadora portátil o de escritorio, con sistema operativo Windows 8 (o mayor).

**Procedimiento experimental**

**Crear el proyecto**

1. Abrir el entorno de desarrollo Keil µVision.
2. En la barra de herramientas, seleccionar Project y luego µVision Project. Crear la carpeta, escribir el nombre del proyecto y salvar (figura 8.2).

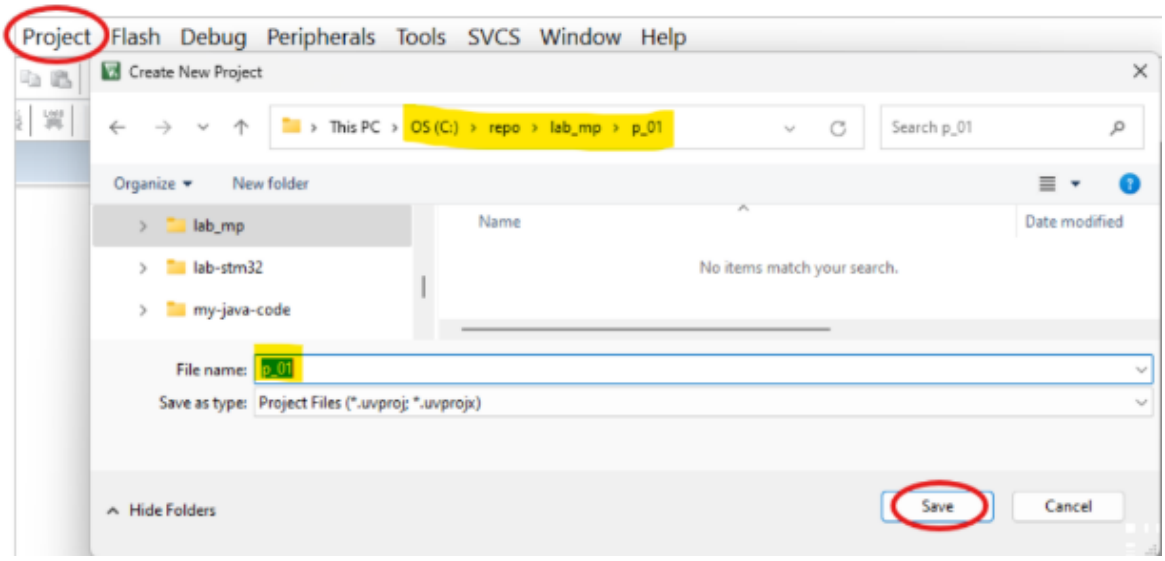



*Figura 8.2: Crear carpeta y nombre para el proyecto*

3. A continuación, aparece otra ventana para localizar el dispositivo con el que se trabajará, seleccionar **stm32f103c8** para esta práctica (ver figura 8.3). Luego oprimir el botón OK.

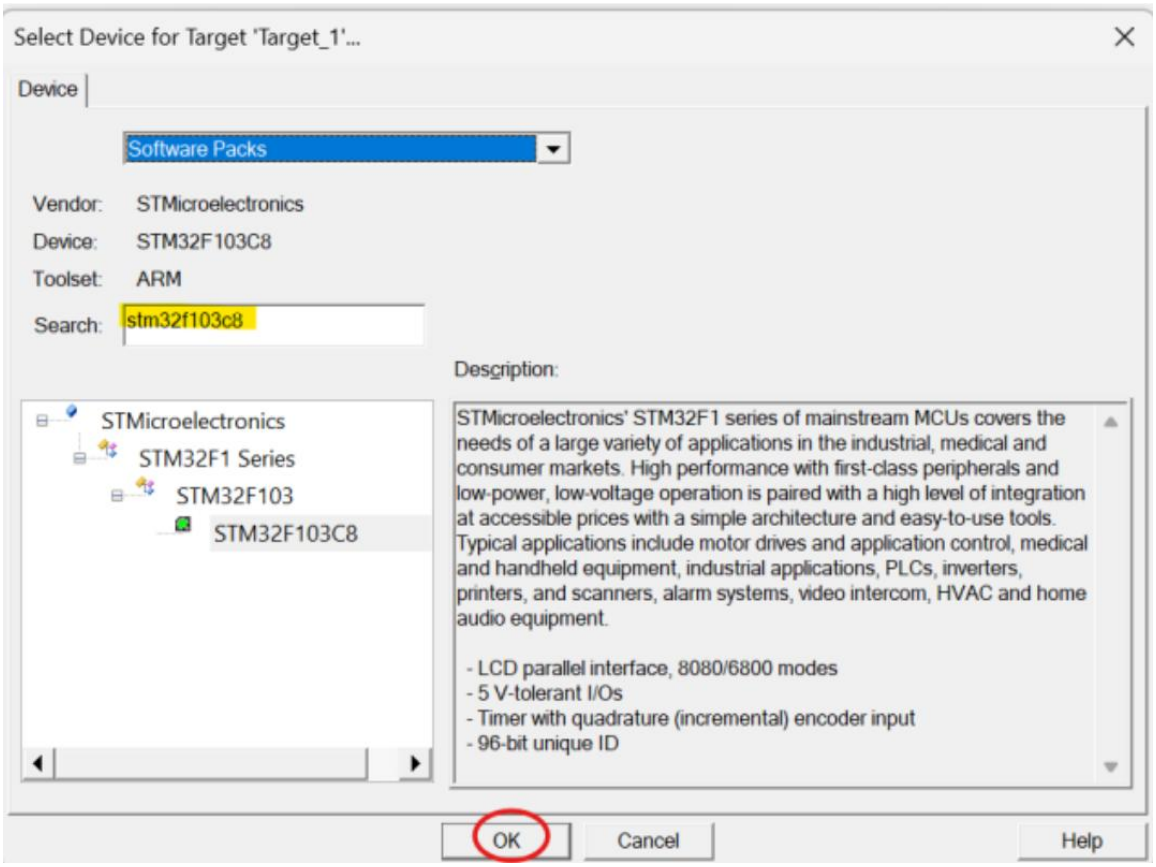



*Figura 8.3: Seleccionar el procesador.*

4. Aparece una nueva ventana para seleccionar las interfaces estándar de software para el procesador (CMSIS por sus siglas en inglés). Seleccionar las básicas, como indica la figura 8.4 y oprimir OK.

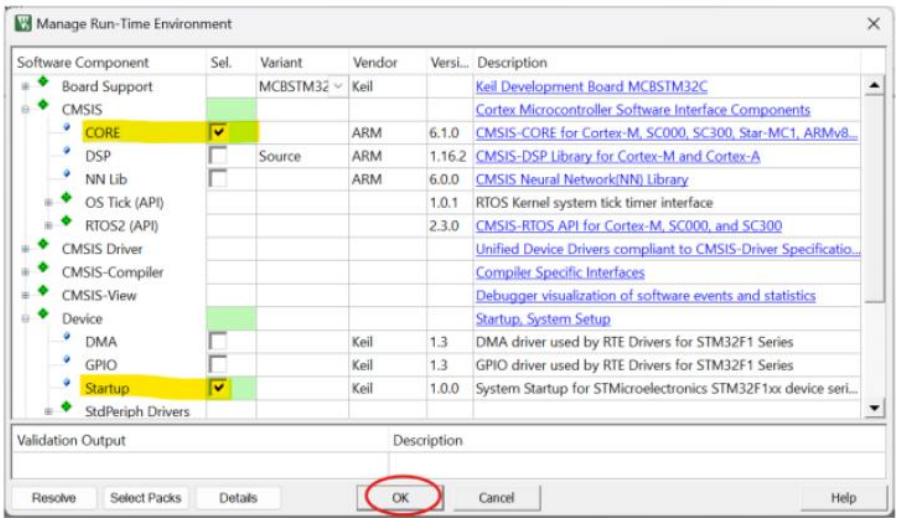



*Figura 8.4: Incluir interfaces de software.*

*Al incluir las interfaces de software (CMSIS), se añade código que se utiliza para inicializar (startup) el procesador (establecer una pila para las operaciones push/pop, establecer un vector de excepciones, relojes y demás) e inicializar el entorno de ejecución de Ç". Aun si se utiliza lenguaje ensamblador, el llamado a estos programas se hace a través de la función main.*

**Crear un archivo de código**

1. En la sección del proyecto (ver figura 8.5), seleccionar Source Group 1, botón derecho y Add New Item.

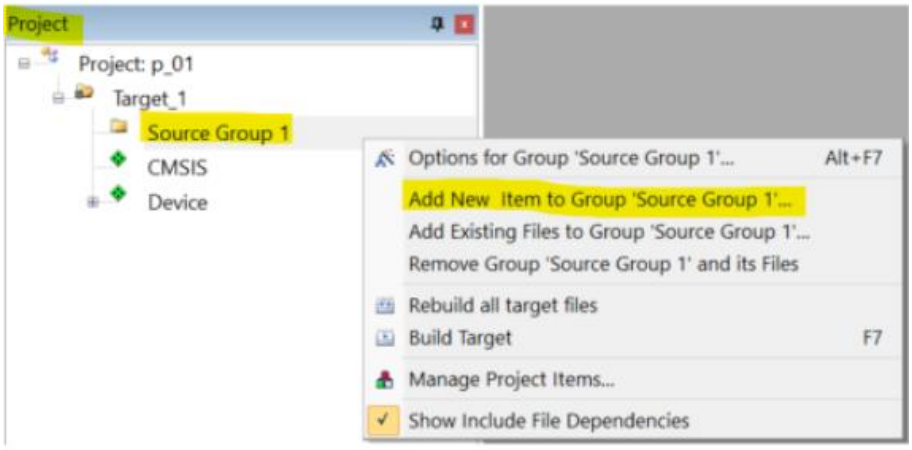



*Figura 8.5: Añadir un nuevo archivo.*

2. Aparece una nueva ventana (figura 8.6) para seleccionar el tipo de archivo fuente (ensamblador para esta práctica), la carpeta donde se alojará y asignar el nombre. Posteriormente, oprimir añadir (Add).

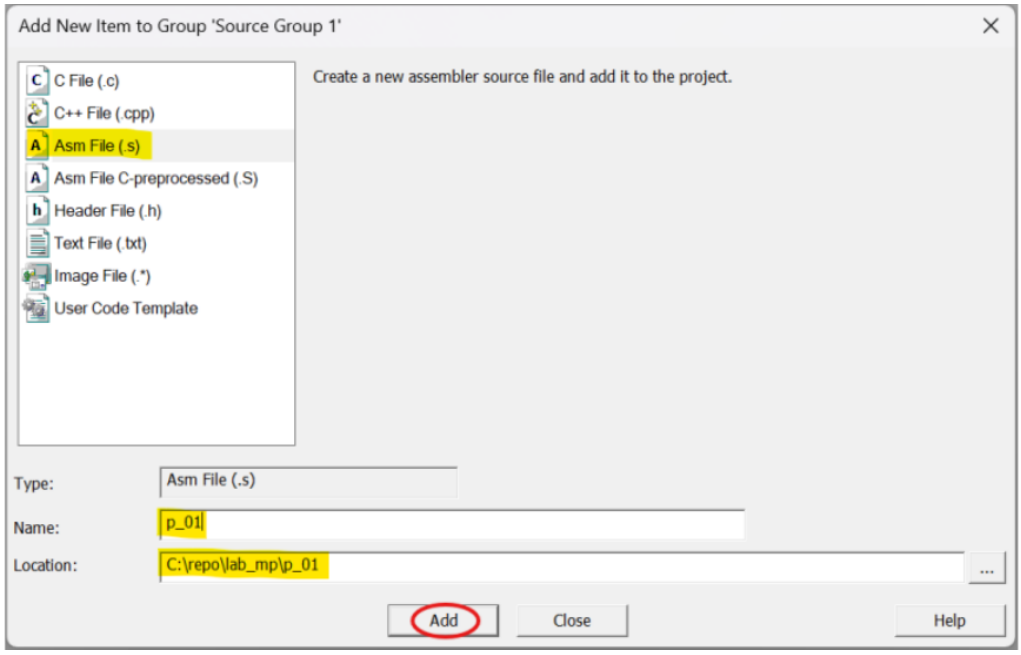



Figura 8.6: Crear el archivo fuente.

3. En la figura 8.7, se muestra un código escrito en lenguaje ensamblador. Analícelo y compréndalo; para luego escribirlo en el archivo recién creado. Tome en cuenta las sangrías después de la columna 1 (también llamadas tabulación o indentación).

p_01.s

```
1  ;directivas de ensamblador
2  ;-------------------------
3              area constantes, data, readonly     ;area de código de solo lectura
4  RCC_APB2ENR equ 0x40021018                      ;registro reloj de puertos
5  GPIOC_CRL   equ 0x40011000                      ;registro para configuración de puerto C (parte baja)
6  GPIOC_CRH   equ 0x40011004                      ;registro para configuración de puerto C (parte alta)
7  GPIOC_ODR   equ 0x4001100C
8
9  ;area de programa
10             area p_01, code, readonly   ; area de código de solo lectura
11             export __main               ; se exporta a startup_stm32f10x_md.s
12 ;inicio del programa
13 ;--------------------
14 __main
15     ldr r0, =0x00000010     ;r0 valor para habilitar reloj de puerto C (bit4=1)
16     ldr r1, =RCC_APB2ENR    ;r1 apunta al registro de reloj de puertos
17     str r0, [r1]            ;almacena '1' en el bit 4 del registro
18
19     ldr r0,=0x44444444      ;r0 valor para reset
20     ldr r1,=GPIOC_CRL
21     str r0,[r1]             ;reset al puerto
22
23     ldr r0,=0x44344444      ;r0 valor para reset
24     ldr r1,=GPIOC_CRH
25     str r0,[r1]             ;reset al puerto
26 ciclo
27     ldr r0,=0x00002000      ;r0 valor para reset
28     ldr r1,=GPIOC_ODR
29     str r0,[r1]             ;reset al puerto
30
31     ldr r0,=0x00000000      ;r0 valor para reset
32     ldr r1,=GPIOC_ODR
33     str r0,[r1]             ;reset al puerto
34
35     b ciclo     ; ciclo infinito
36 ;fin del programa
```

*Figura 8.7: Código en lenguaje ensamblador.*

**Configurar el simulador**

1. Seleccionar el icono (Options for Target, ver figura 8.8) para configurar la herramienta de depuración.

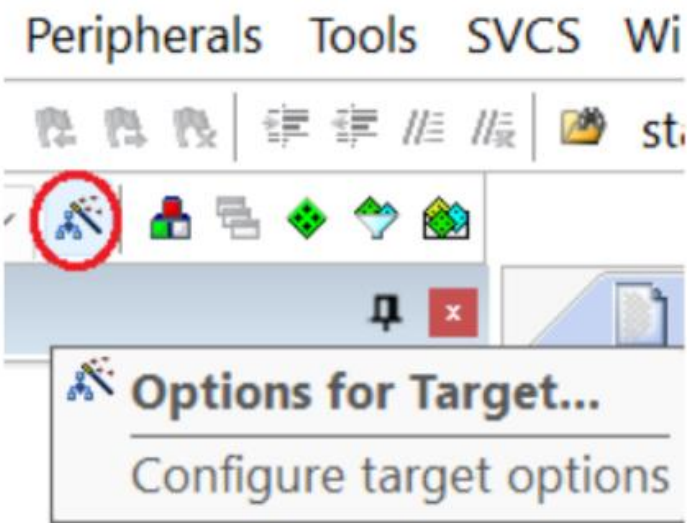



*Figura 8.8: Configuración de herramientas software.*

2. Aparece una nueva ventana. En esta, seleccionar la pestaña Debug (8.9) y activar la opción para usar el simulador (Use Simulator ). Además, si se desea una vista estilizada de las ventanas de simulación, se pueden modificar las opciones Dialog DLL y Parameter. Oprimir OK.

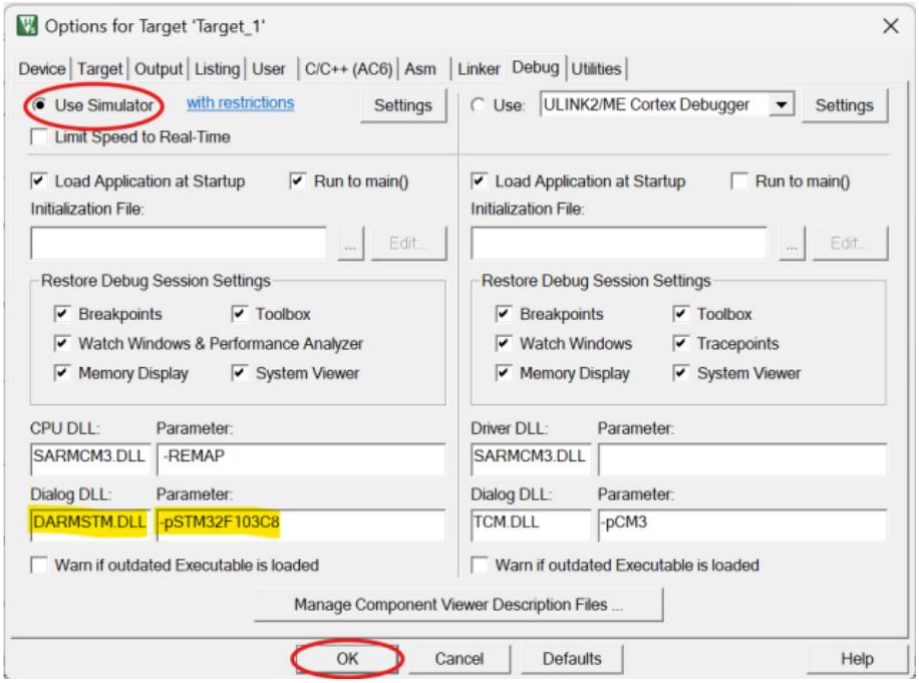



*Figura 8.9: Configuración de la herramienta de depuración.*

**Ejecutar el programa**

1. Seleccionar el icono (figura 8.10) para ensamblar el proyecto. Lleva a cabo las funciones del preprocesador, compilador, ensamblador y enlace (linker ). Verificar que no haya errores.

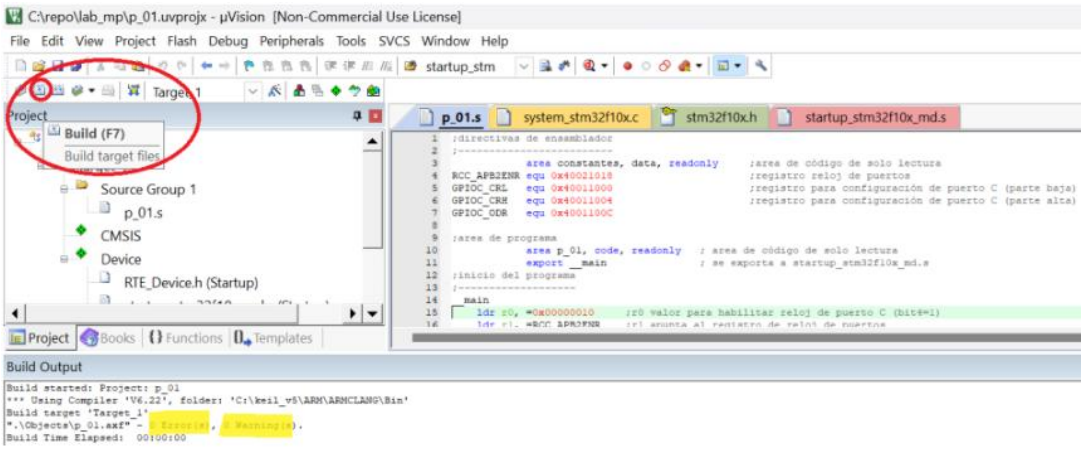



*Figura 8.10: Construir el proyecto.*

2. Iniciar la sesión para depurar (marcado con el número 1 en la figura 8.11).

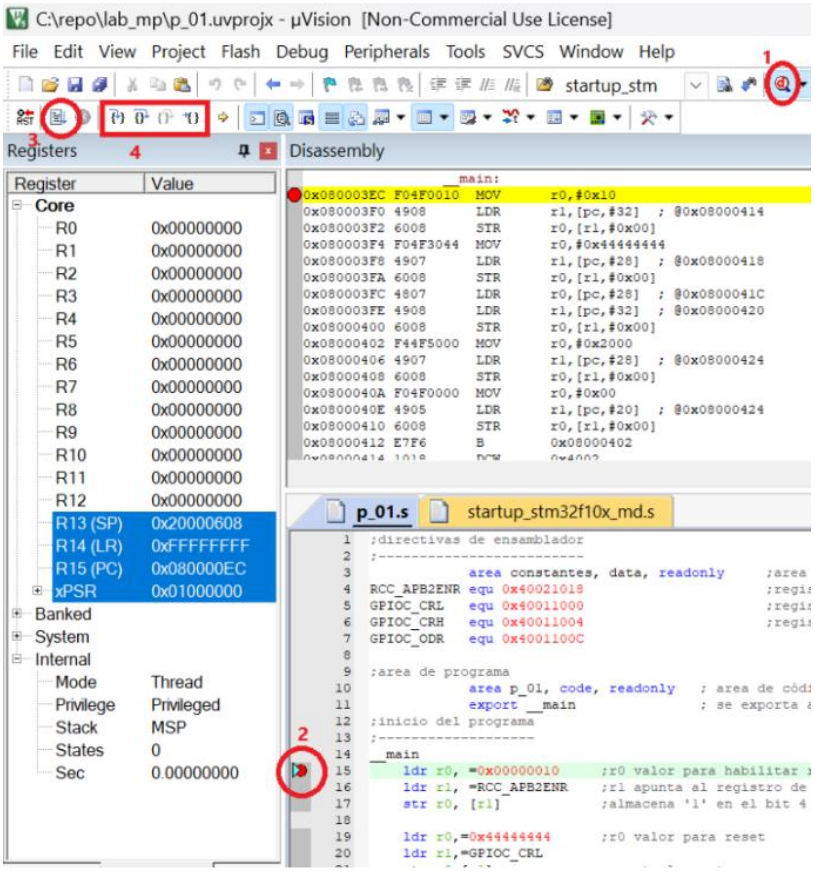



*Figura 8.11: Ejecutar, simular y depurar.*

3. Colocar un punto de prueba (marcado con el número 2 en la figura 8.11).
4. Iniciar la ejecución del programa (marcado con el número 3 en la figura 8.11). Notar que el programa se ejecuta hasta detenerse en el punto de prueba.
5. A partir del punto de prueba (breakpoint), se pueden utilizar otras opciones (marcadas con el número 4 en la figura 8.11) para ejecutar cada instrucción, ejecutar toda una función o salir de ella.
6. Para terminar la sesión, oprimir el icono marcado con el número 1 en la figura 8.11

**Cuestionario**

1. Explique sus resultados y observaciones al realizar la simulación de este programa.
2. Explique la importancia del proceso de depuración en el diseño de sistemas basados en procesadores.
3. Investigue otras herramientas, software o hardware, que nos ayudan en el proceso de diseño de sistemas basados en procesadores.

# Laboratorio de Microprocesadores

## Práctica 9 Arquitectura del procesador


**Tema**

8.5. Tecnologías de última generación en microprocesadores.

**Objetivos**

- Mediante la escritura de código en lenguaje ensamblador, identificar los elementos de la unidad de procesamiento central (CPU ).
- Probar el programa mediante el uso de un emulador.

**Introducción**

La unidad central de procesamiento de un ARM es el componente funcional principal. Como se muestra en la figura 2.12, se pueden seleccionar dos registros de origen utilizando los buses A y B. Los datos del bus B se enrutan a través de un registro de desplazamiento (shifter ) antes de llegar a la unidad aritmética-lógica (ALU ). Los datos del bus A van directamente a la ALU, pero además los buses A y B pueden proporcionar datos para el multiplicador(multiplier ). Los datos que vienen de la memoria (memory) o de los dispositivos de entrada y/o salida (I/O) se introducen directamente en el bus de la ALU, se pueden almacenar en uno de los registros de propósito general, se pueden procesar por la ALU o se pueden tomar directamente del bus B para escribirlos en la memoria o en los dispositivos de E/S. La CPU utiliza el registro de dirección siempre que necesita leer o escribir en la memoria o en los dispositivos de E/S, cada vez que se obtiene una instrucción y en todas las operaciones de carga y almacenamiento. El registro de dirección se puede cargar desde el contador de programa (obteniendo la siguiente instrucción), desde la ALU (en modos de direccionamiento) y se puede incrementar y almacenar nuevamente (para los modos de direccionamiento con apuntadores y contador de programa).

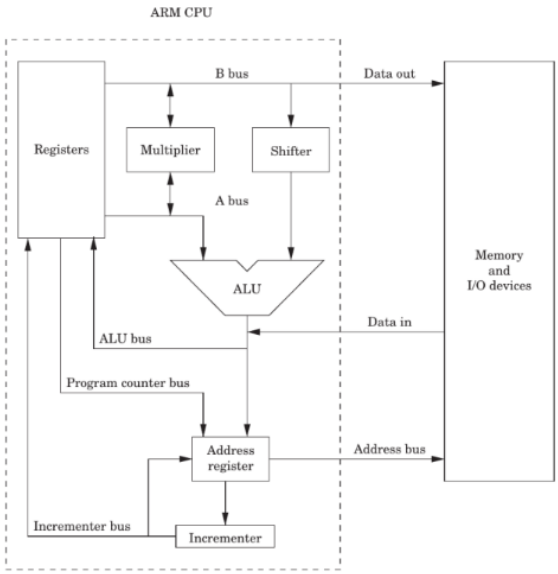



*Figura 9.1: ARM CPU.[3]*

El procesador cuenta con los siguientes registros de 32 bits (ver figura 2.13):
Los registros r0 a r12 son de propósito general. Los registros r0 a r7 son accesibles por todas las instrucciones, mientras que los registros r8 a r12 solo pueden ser accedidos por las instrucciones de 32 bits.
Registros para funciones especiales:
El registro r13 es utilizado como apuntador a la pila (Stack Pointer ).
El registro de enlace r14 (Link Register ) recibe la dirección del contador de programa cuando se ejecutan instrucciones de bifurcación con enlace (Branch and link).
El Contador de programa r15 (Program Counter ). Es utilizado por la CPU para apuntar a la dirección de la siguiente instrucción a ejecutar. El contador de programa de 32 bits puede acceder a un máximo de 4 GBytes (232) de código.

Registro de estados xPSR (Program Status Register ). El estado del procesador a nivel del sistema se divide en tres categorías:

- Aplicación, contiene el estado de banderas N (Ngative), Z (Zero), C (Carry) y V (Overflow).
- Interrupciones, contiene el número de la rutina de atención a la interrupción (ISR).
- Ejecución, contiene dos campos: uno para la continuidad o interrupción de la instrucción (ICI) y otro para la instrucción IF-THEN.

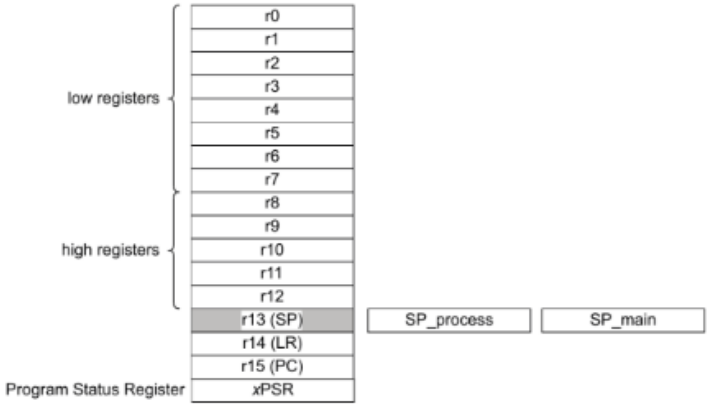



Figura 9.2: Registros del procesador.[4]

El lenguaje ensamblador es un lenguaje de bajo nivel con el que se puede acceder directamente a la estructura interna de la CPU. Las distintas formas en que se especifican los operandos en una instrucción se denominan modos de direccionamiento. En esta práctica se escribirá un programa aplicando algunos de los modos de direccionamiento clásicos como: de registro, inmediato y de registro indirecto.

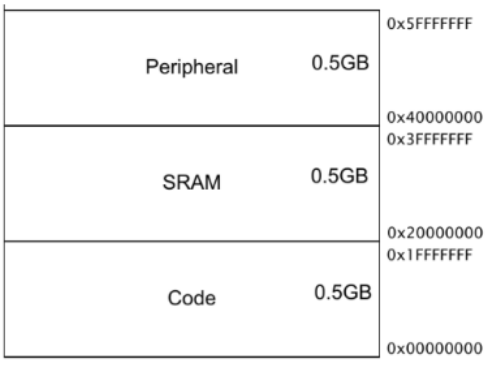



*Figura 9.3: Mapa de memoria del procesador.[4]*

Sin embargo, el programa que será ejecutado por la CPU debe ser alojado en una memoria externa a la CPU (ver la figura 2.12), para que de ahí vaya tomando las instrucciones que conformarán una tarea específica. Cuando se diseña un sistema de microcomputador, a cada recurso externo, como las memorias, puertos de interfaz y demás (que serán abordados en prácticas posteriores) se les asigna una dirección única (ver el mapa de memoria 2.14), mediante la cual se accede a ese recurso en particular. El procesador STM32F103C8 cuenta con 64Kbytes para el código e inicia en la dirección 0x00000000.
La placa de desarrollo STM32F103C8T6 Blue Pill utiliza un procesador STM32F103C8 con núcleo Cortex-M3 sobre la arquitectura ARMv7-M.

**Actividades Previas**

1. Leer detenidamente y comprender la sección introductoria a la práctica.
2. Comprender los objetivos y el procedimiento experimental.
3. Haber realizado satisfactoriamente la práctica anterior.

**Material**

- Placa de desarrollo STM32F103C8T6 Blue Pill.
- Programador, emulador ST-LINK V2.
- cables hembra-hembra tipo Dupont.

**Equipo**

- Computadora portátil o de escritorio, con sistema operativo Windows 8(o mayor).

**Procedimiento Experimental**

Estructura de un programa
1. Crear un nuevo proyecto, siguiendo el procedimiento experimental de la práctica anterior.
2. En el archivo para el código, escribir, con comentarios, la estructura que tendrá el programa:

```
;area de constantes y variables
;------------------------------

;area de programa
;----------------

;inicio del programa
;-------------------

;fin del programa
;----------------
```

3. Escribir las directivas del ensamblador, para definir las áreas de memoria (area), exportar el programa (export) e indicar el final del programa (end). En este enlace se encuentra la descripción detallada de todas las directivas soportadas por el ensamblador.

4. Escribir la etiqueta (main), que indica el inicio del programa.

*En ensamblador, los comentarios inician con ";" (signo de punto y coma), no son interpretados por la CPU. Constituyen una buena práctica de programación y son de gran ayuda para el programador.*

*Son instrucciones para el proceso del ensamblador, no las ejecuta la CPU.*

*Las etiquetas son muy útiles y versátiles en el ensamblador, dan nombre a las ubicaciones de memoria que serán descifradas posteriormente por el ensamblador o el enlazador.*

```
;area de constantes y variables
;------------------------------
    area constantes, data, readonly  ;area de código de solo lectura

;area de programa
;----------------
    area p_02, code, readonly  ; area de código de solo lectura
 export __main    ; se exporta a startup_stm32f10x_md.s

;inicio del programa
;-------------------
__main

;fin del programa
;----------------
 end
```

**Preparar el emulador y la placa de desarrollo**

1. Interconectar la placa de desarrollo con el emulador, como muestra la figura 2.15.

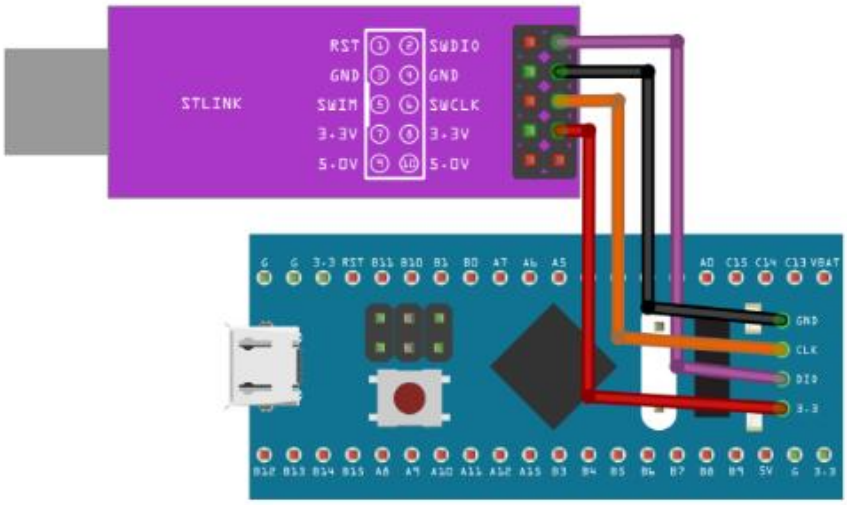



*Figura 9.4: Conexión de tarjeta a emulador.*

2. Conectar el emulador al puerto USB de la computadora.
3. Abrir el entorno de desarrollo Keil µVision.
4. Seleccionar Options for Target, luego Debug y configurar como indica la figura 2.16.

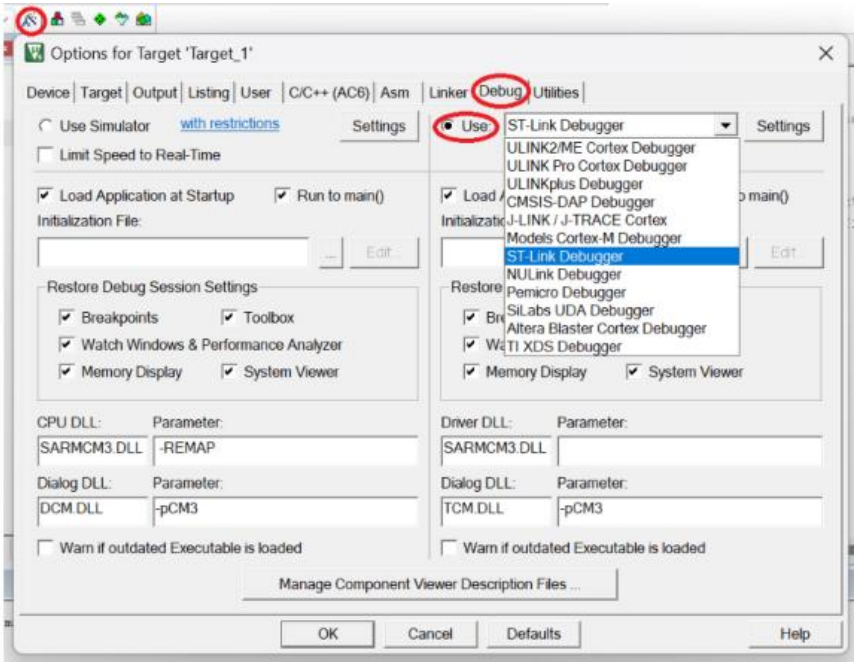



*Figura 9.5: Configurar el emulador.*

**Instrucciones en ensamblador**

1. Escribir las instrucciones de ensamblador usando los modos de direccionamiento inmediato, de registro y registro indirecto. Para el modo de registro indirecto, declarar una constante que contenga una dirección cualquiera (ver figura 2.14). Además, escribir las instrucciones para que el programa se ejecute y permanezca en un ciclo infinito.

```
;area de constantes y variables
;------------------------
   area constantes, data, readonly  ;area de código de solo lectura
DIR_SRAM  equ 0x20000000
;area de programa
;----------------
   area p_02, code, readonly  ; area de código de solo lectura
   export __main    ; se exporta a startup_stm32f10x_md.s
;inicio del programa
;-------------------
__main
 ;direccionamiento inmediato
 mov r0,#0x69  ;mover 1 byte
 mov r1,#0x1234  ;mover 2 bytes(word)
 movw r2,#0x6655  ;otra manera de mover 2 bytes
 ldr r3,=0x12345678 ;mover 4bytes(dword)
 ;direccionamiento de registro
 mov r4,r3   ;copia el valor de r3 en r4
 ;direccionamiento de registro indirecto
 ldr r5,=DIR_SRAM ;cargar la dirección de la RAM en r5
 str r4,[r5]   ;almacenar el valor que contiene r4
      ;en la dirección a donde apunta r5

 ciclo
  b ciclo    ;Ciclo infinito
 ;fin del programa
 ;----------------
  end
```

*Los programas para sistemas de procesadores y microprocesadores, deberán ejecutarse siempre que esté encendido el sistema.*

2. Construir el proyecto (ver figura 1.10), abrir el emulador e ir ejecutando las instrucciones, siguiendo el proceso descrito en la figura 1.11. Cuando utiliza por primera vez el emulador (st-link), aparecerá un mensaje (figura 2.17) con opción para actualizar el software. Por lo tanto, se procede a actualizarlo.

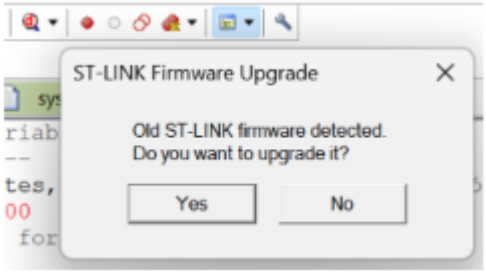



*Figura 9.6: Actualizar software del emulador.*

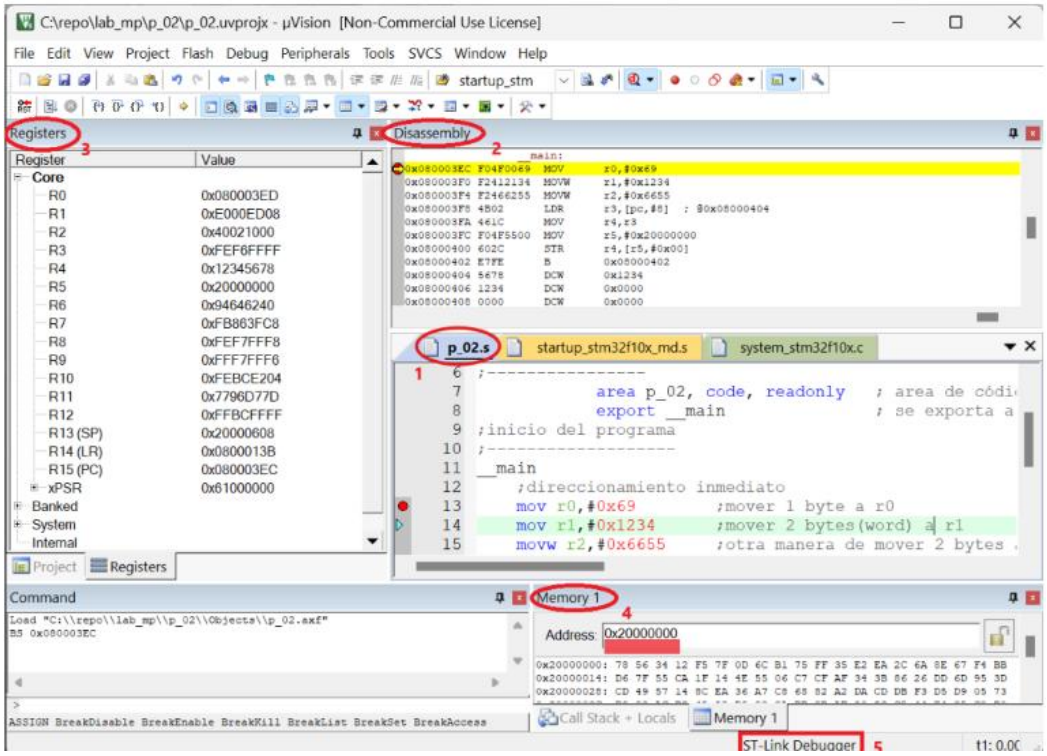



*Figura 9.7: Emulador.*

*El proceso de emulación es similar a la simulación; sin embargo, la gran diferencia es que ahora se está interactuando con el procesador a través del st-link.*

3. Colocar un punto de prueba al inicio del programa y ejecutar hasta ese punto (como en el procedimiento experimental de la práctica anterior). En la figura 2.18 se identifica lo siguiente:

- El programa escrito (código fuente).
- La ejecución paso a paso del programa y el área de memoria del programa.
- Los registros de propósito general (r0 a r12), el apuntador a la pila (SP), registro de enlace (LR), contador de programa (PC) y registro de estados del programa (xPSR).

- Ventana para mostrar el contenido de la memoria, de acuerdo con la dirección introducida.
- Indica que está conectado el emulador.

4. Ensamblar y ejecutar el programa en el emulador. Observe y haga sus anotaciones respecto al comportamiento de los recursos internos del procesador.

5. Agregar código para que la ALU realice una simple operación aritmética:

```
;registro de estados
 ldr r0,=0xFFFFFFFF ;carga el número máximo en r0
 mov r2,#0x55  ;carga un valor cualquiera en r2
 mov r3,#0x11  ;carga un valor cualquiera en r3
 adds r5,r0,#1  ;al valor de r0 suma 1 y resultado en r5
 adc r6,r2,r3  ;r6=r2+r3+C(acarreo)
```

6. Ensamblar y ejecutar en el emulador. En la figura 2.19 se observan algunas recomendaciones:

1. Añadir un punto de prueba.

2. Antes de ejecutar la instrucción ADDS, poner en "0"(punto 3) las banderas.

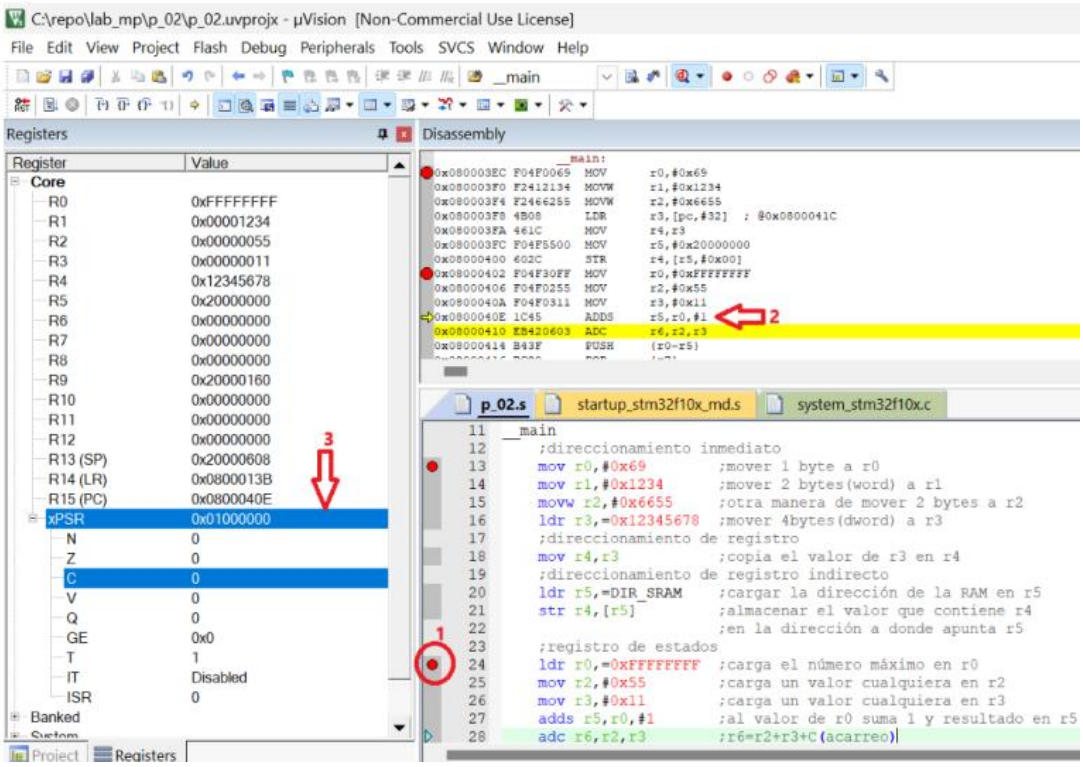



*Figura 9.8: Emulador.*

*La descripción detallada de esta y todas las instrucciones del procesador, se encuentran en el manual de programación.*

7. Continuar ejecutando el programa y haga sus anotaciones respecto al comportamiento de los bits del registro de estados.
8. Agregar código para observar el comportamiento de la pila y apuntador (SP):

```
;registro apuntador a la pila
 push {r0,r1-r5}  ;guarda el contenido de r0 y r1 a r5 en
       ;la pila de datos a partir de la dirección
       ;a donde apunta SP
 pop {r7}    ;Extrae un dato de la pila y lo deposita en r7
```

9. Ensamblar y ejecutar en el emulador. En la figura 2.19 se observan algunas recomendaciones:

1. Antes de ejecutar la instrucción PUSH, observe y anote la dirección que tiene el registro SP (punto 2).

***Escribir la dirección 0x200005f0***.

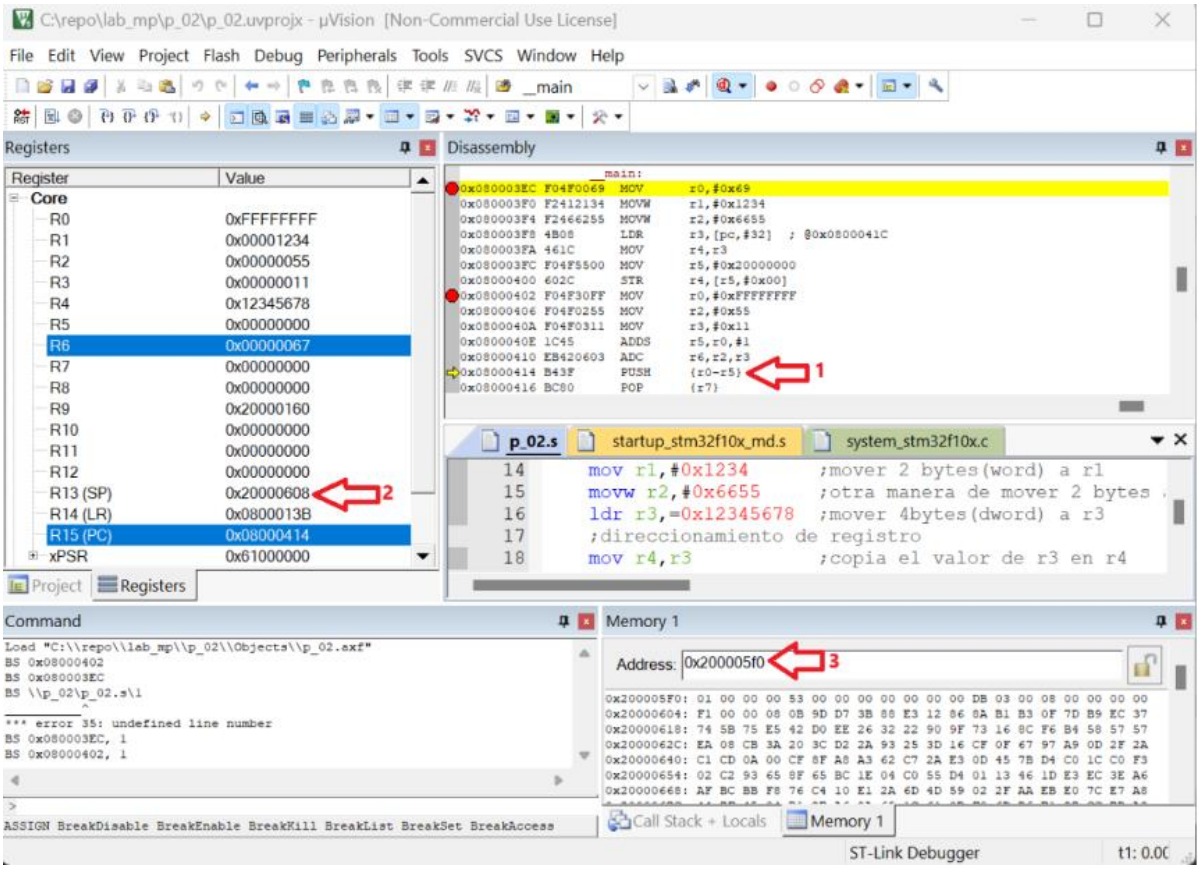



*Figura 9.9: Emulador.*

10. *Continuar ejecutando el programa y haga sus anotaciones respecto al comportamiento del contenido en la dirección de memoria seleccionada.*

**Cuestionario**

1. Para cada paso del procedimiento experimental, explique sus resultados y observaciones al utilizar el emulador como herramienta para depurar programas.

2. Escribir, ensamblar, emular y depurar un programa en lenguaje ensamblador para el procesador ARM Cortex-M3 que escriba 100 números a partir de la dirección de memoria 0x20000000.