Unidad I - Introducción al Ensamblador

Msc. Lic. Víctor Rodríguez Estévez

August 19, 2025

El Problema - Porque el ensamblador?

Bienvenida

- En este curso: **Programación en bajo nivel**
- Enlazar las destrezas sobre **programación**...
- ... y los conocimientos sobre **la arquitectura del computador**.

El Problema - Porque el ensamblador?

Bienvenida

- Cual es la **dinámica de la arquitectura del procesador** para ejecutar los programas que implementan los **algoritmos diseñados**?
- Como podemos **diseñar nuestros algoritmos** para optimizar el **uso del procesador**?

El Problema - El modelo

Bienvenida

- Acudir a **la dinámica del procesador**
- Realizar **representaciones formales de las estructuras de datos**
- **Manipular** el sistema, acceso a registros, memoria, dispositivos E/S, coprocesador matemático.
- Esto es: **"programar en bajo nivel"**.

El Problema - El modelo

Que vamos a hacer hoy?

- **En esta unidad:**
  - **Presentamos algunos aportes de aprender a programar en bajo nivel.**
  - **Se describirá la dinámica del procesador para la ejecución de los programas.**
  - **Recordaremos además conceptos básicos de lógica, y arquitectura de computadoras.**

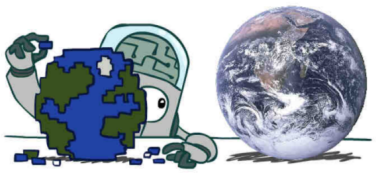
Problema - El modelo

Para que aprender a programar en bajo nivel?

Más allá de su valor histórico, aporta una serie de beneficios fundamentales:

- Comprender a fondo el **funcionamiento de un computador**
- Escribir **código más eficiente** y aprovechar al máximo los recursos del hardware
- Desarrollar habilidades de resolución de problemas
- Fomentar la creatividad e innovación

Problema - El modelo



Para que aprender a programar en bajo nivel?

En relación a las competencias específicas:

- **Análisis de problemas informáticos**: El ensamblador ayuda a **descomponer problemas complejos en tareas más simples** y a entender cómo se pueden implementar estas tareas a nivel de máquina.
- **Identificación de entidades y roles**: Al programar en ensamblador, los estudiantes aprenden a **identificar las diferentes partes de un sistema informático** y cómo interactúan entre sí.

Problema - El modelo

Para que aprender a programar en bajo nivel?

En relación a las competencias específicas:

- **Formalización de problemas**: La necesidad de **traducir algoritmos a código de máquina** fomenta un **pensamiento estructurado** y la capacidad de formalizar problemas de manera precisa.
- **Evaluación de alternativas**: Al comparar diferentes implementaciones de un mismo algoritmo en ensamblador, los estudiantes aprenden a **evaluar la eficiencia y a elegir la mejor solución**.
- **Innovación**: Fomenta la creatividad y la búsqueda de soluciones innovadoras a problemas complejos.

El Problema - El modelo

Situación: Control Remoto de una Puerta

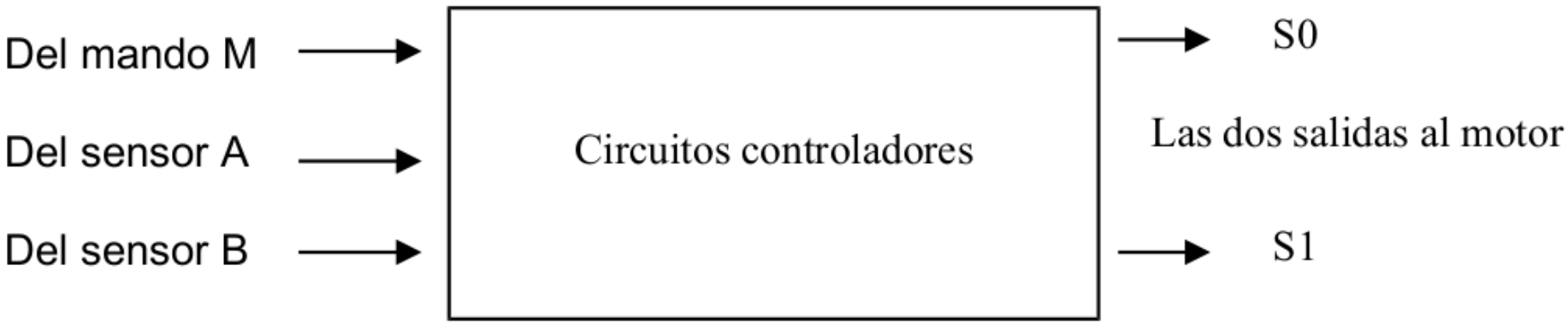
- **Convertir señales de control en señales de acción**
- **Señales de Control**:
  - **Mando de control remoto (M)**: Indica si se presiona cerrar puerta (0) o abrir perta(1).
  - **Sensor de la puerta (B)**: Indica si la perta esta cerrada (0) o esta abierta (1).

El Problema - El modelo

Situación: Puerta con control remoto

- **Convertir señales de control en señales de acción**
- **Señales de acción**:
  - 01 = mover la puerta para abrirla
  - 10 = mover la puerta para cerrarla
  - 00 = detenerse

El Problema - El modelo



El Problema - Recordando Lógica

Pregunta

- Para diseñar los circuitos construye una tabla con todas las posibilidades de entradas y sus respectivas salidas.
- Plantea la expresión para las señales de salida.

El Problema - Recordando definiciones - Derivada

Entrada			Salida		Descripción
H	L	M	S1	S2	
0	0	0	0	0	Se ordena cerrar la puerta, y la puerta esta cerrada, por tanto no debe hacerse nada
0	0	1	0	0	
0	1	0	1	0	Se ordena cerrar la puerta, y la puerta no esta cerrada, por tanto debe moverse hasta cerrarse
0	1	1	1	0	
1	0	0	0	0	Se ordena abrir la puerta, y la puerta esta abierta, por tanto no debe hacerse nada
1	0	1	0	0	
1	1	0	0	1	Se ordena abrir la puerta, y la puerta no esta abierta, por tanto debe moverse hasta abrirse
1	1	1	0	1	

El Problema

Codificación

$S0$ es 1 solo cuando $M = 0$ y $A = 1$,sin importar el valor de B

- $S0 = (\overline{M}) \wedge A$

$S1$ es 1 solo cuando $M = 1$ y $B = 1$, por lo que:

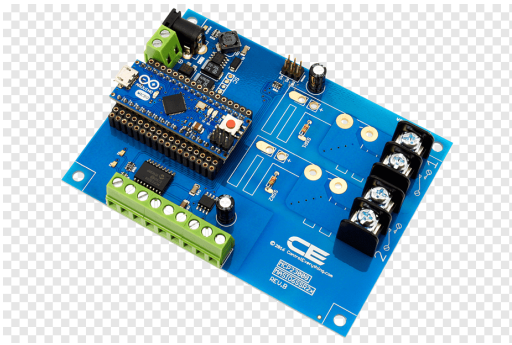
- $S1 = M \wedge B$

Problema

Propósito específico

- Este tipo de dispositivos tiene **propósito especifico**.
- Es posible construir dispositivos de **propósito general**?
- Básicamente **procesadores** , capaces de resolver una gran cantidad de problemas.
- La clave...capacidad de recibir **programas**.

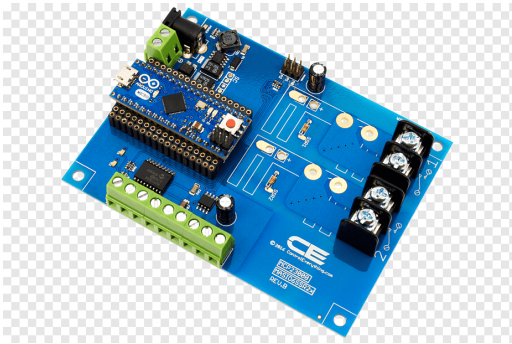
Problema - El modelo



Propósito General

1. **leer y almacenar** la señal M en un **elemento de memoria**
2. **obtener la negación** del elemento de memoria
3. **leer y almacenar** la señal A en otro elemento de memoria
4. procesar ambos elementos de memoria por un circuito Y
5. entregar el resultado al motor
6. volver al 1

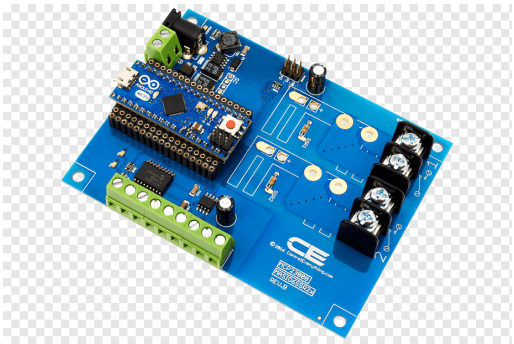
El Problema - El modelo



Pregunta

¿Con que debería contar el procesador para ejecutar el programa?

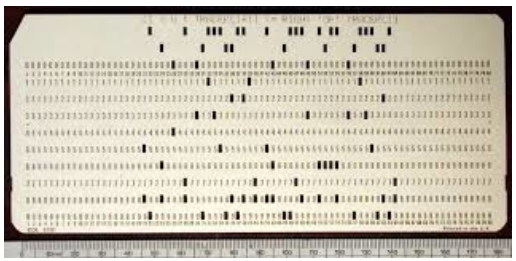
El Problema - El modelo



Crecimiento de Población

- **Unidad Aritmética - Lógica**, dotada con los usuales Y,O y NO y también circuitos mas complejos, como sumadores que pueden operar sobre solo un par de bits u otros mas complejos que pueden operar sobre bytes.
- **Registros**, memorias que pueden contener resultados intermedios para entregar a los dispositivos procesadores o recibir de ellos.

El Problema - El modelo

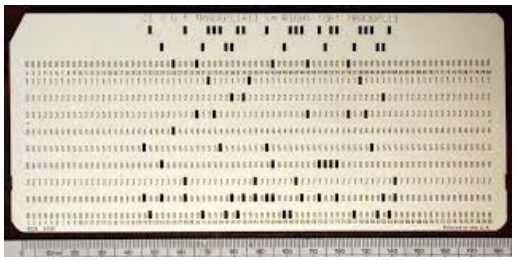


Tarjeta perforada

Formato físico

- Una tarjeta perforada típica tenía dimensiones de 7.375 x 3.25 pulgadas (el estándar IBM).
- Estaba dividida en filas y columnas. La más común tenía 80 columnas y 12 filas.
- Cada columna **representaba un carácter** (letra, número o símbolo).

El Problema - El modelo

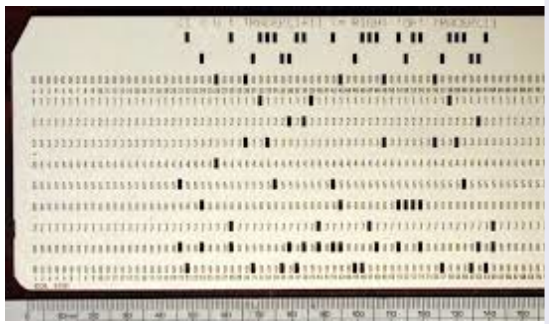


Tarjeta perforada

Codificación de datos

- Los datos se codificaban perforando agujeros en posiciones específicas de cada columna.
- Las instrucciones de igual manera recibían una codificación.
- Apartado también sobre la dirección de dispositivos, así como un controlador de entrada salida.

El Problema - El modelo



Tarjeta perforada

Lectura de la tarjeta

- Las tarjetas se introducían en una lectora de tarjetas perforadas.
- Estas máquinas detectaban los agujeros usando contactos eléctricos o sensores ópticos.
- Cada línea leída se convertía en una secuencia digital que la computadora procesaba.

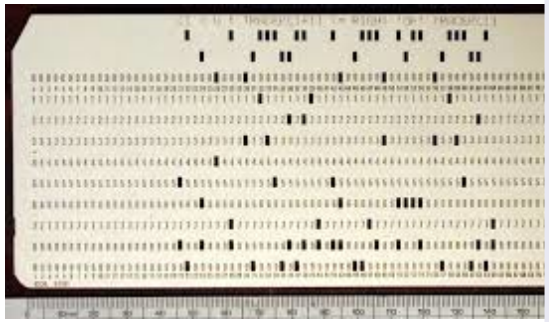
El Problema - El modelo

Pregunta

POdrías codificar :

- *INICIO* $\rightarrow$ 01
- *LEER* $\rightarrow$ 02
- *SUMAR* $\rightarrow$ 03
- *IMPRIMIR* $\rightarrow$ 04
- *FIN* $\rightarrow$ 99

El Problema - El modelo



Tarjeta perforada

Lectura de la tarjeta

- Escribir directamente en lenguaje máquina (binario o hexadecimal) era propenso a errores y difícil de leer y mantener. Idea: usar mnemónicos más comprensibles en lugar de cadenas numéricas, facilitando el desarrollo y la depuración del código..
- Organizar código que se ejecute instrucción a instrucción usando el harware directamente

El Objeto

Vistas las anteriores situaciones, establecemos el **objeto de estudio** en este curso como:

El lenguaje Ensamblador

El Objeto

Qué es el lenguaje ensamblador?

- Traduce directamente las instrucciones del procesador en una forma legible para los humanos.
- Es un puente entre el código máquina (instrucciones binarias ejecutadas por la CPU) y los lenguajes de alto nivel (como C, Java o Python).
- Usa **mnemónicos** en lugar de instrucciones.
- Es **específico** para cada **arquitectura**.
- Permite un control total del hardware, accediendo a **registros, memoria y periféricos.**
- Es esencial para la optimización de código y seguridad informática

El Objeto

Para que estudiar assembler?

- Mayor comprensión del **hardware y software**
- **Optimización** de rendimiento en aplicaciones críticas (embebidos, simulaciones, videojuegos.
- **Seguridad informática** y explotación de vulnerabilidades
- Hacking ético, **análisis de malware** y explotación de vulnerabilidades.
- Desarrollo de **sistemas operativos** y **compiladores**
- Reversing y **análisis de software**

El Objetivo

Vistas las anteriores situaciones y el Objeto de estudio en este curso, establecemos el **Objetivo** como:

Plantear soluciones a problemas
a nivel de hardware y software
mediante el uso de

PROGRAMACIÓN EN ENSAMBLADOR.

El Contenido

Acudimos a la siguiente teoría sobre nuestro objeto de estudio para llegar a nuestro objetivo y asi resolver el problema.

UT1: INTRODUCCIÓN AL LENGUAJE ENSAMBLADOR

- Circuitos lógicos
- Arquitectura del Computador
- El lenguaje máquina
- Sistemas Operativos
- Ensambladores Depuradores
- Lenguajes de Programación
- Compiladores

El Contenido

UT2: PROGRAMACIÓN BÁSICA EN ENSAMBLADOR

- Registros
- Modos de direccionamiento
- Instrucciones Básicas
- Bifurcaciones
- Ciclos
- Rutinas
- Interrupciones

Contenido

UT3: PROGRAMACIÓN EN WINDOWS

- Introducción a la programación en Windows
- Manejadores
- Ventanas
- Entrada teclado /mouse
- Cajas de dialogo
- Procesos, Programación Multihilos
- Objeto Evento
- Librerías de Enlace Dinámico (DLL)
- Imágenes, texto archivos.
- Aplicaciones

Contenido

UT4: El coprocesador Matematico

- Introducción y Fundamentos
- Representación de Datos en Coma Flotante
- El Entorno de la FPU (x87)
- Gestión de la Pila y Transferencia de Datos
- Programación de la FPU x87
- Funciones y Operaciones Avanzadas
- Aritmética Vectorial (Packed)
- Instrucciones SIMD Modernas (SSE, AVX)
- Integración y Aplicaciones Prácticas
- Graficación

El Contenido

- **Iteración 1**: 18 de Septiembre al 15 de Septiembre:

Como?

Unidad Temática	1	2	3	4	5	6	7
UT1	■	■	■	■	■	■	■
UT2	■	■	■	■	■	■	■
Examen	■	■	■	■	■	■	■

El Contenido

- **Iteración 2** : 16 de Septiembre al 27 de Octumbre:

Como?

Unidad Temática	1	2	3	4	5	6	7
UT3	■	■	■	■	■	■	■
UT4	■	■	■	■	■	■	■
Examen	■	■	■	■	■	■	■

El Contenido

- **Iteración 3**: 28 de octubre al 15 de Diciembre.

Como?

Unidad Temática	1	2	3	4	5	6
UT4	■	■	■	■	■	■
UT5	■	■	■	■	■	■
Examen	■	■	■	■	■	■

La Forma

Espacio y Tiempo donde se realizará el proceso.

Donde y cuando?

- **Tiempo**: 20 semanas (17 de Febrero al 30 de Junio)
- **Aula**: 6 horas académicas a la semana.
- **Virtual**: Lecturas y Diapositivas 6 horas a la semana.
- **Producción**: Resolución de ejercicios : 10 hora semanales.

El Método

Como utilizamos el contenido para llegar al objetivo?

Como?

- Aprendizaje basado en problemas (Casos de estudio).
- Exposición y resolución de ejercicios colaborativos.
- Implementación de los programas en emuladores y en entornos
- Modelar, resolver, programar y experimentar!!!!

La Evaluación

En que medida nos acercamos al objetivo?

Como?

- Pruebas Diagnosticas.
- **Pruebas**: Resolución prácticas.
- **Experimentación** - Desarrollo de Aplicación y pruebas.
- **Tres iteraciones**: Tres Pruebas Objetivas.

La Evaluación

En que medida nos acercamos al objetivo?

Como?

- **Primera Iteración**: Semana 7
- **Segunda Iteración** : Semana 15
- **Tercera Iteración**: Semana 21

Bibliografía

Bibliografía

- Assembly Language For X6 processor 6 Edition, Kip R. Irvine
- Proffesiional Assembly Language - Richard Blum
- NASM — The Netwide Assembler
- PC Assembly Language , Paul A. Carter