

# Introducción al simulador

*Departamento de Automática*

## Índice

- **Introducción al simulador**
- **Mapa de memoria**
  - Proyecciones de dispositivos de entrada/salida
- **Procesador**
  - Registros
  - Instrucciones
- **Ciclo de ejecución**
- **Ejercicios propuestos**

## Descripción general

### Sitio web

<https://parraman.github.io/asm-simulator/>

### Proyecto github (código fuente)

<https://github.com/parraman/asm-simulator>

- Simulador de una arquitectura de 16 bits
  - Angular 2+ (Typescript)
  - Bootstrap 3
  - CodeMirror

## Descripción general

Editor de código

The screenshot shows the 16-bit Assembler Simulator interface with several key components:

- Code Editor:** Contains assembly code for a program that prints "Hello World!".
- Visual Display:** Shows a 3x3 keypad and a 7x3 numeric display.
- CPU Registers:** Shows registers A, B, C, D, E, H, S, and PC with their current values.
- Memory:** A grid showing memory addresses from 0000 to 00FF and their contents.
- Event Log:** A log of system events such as instruction execution and register updates.

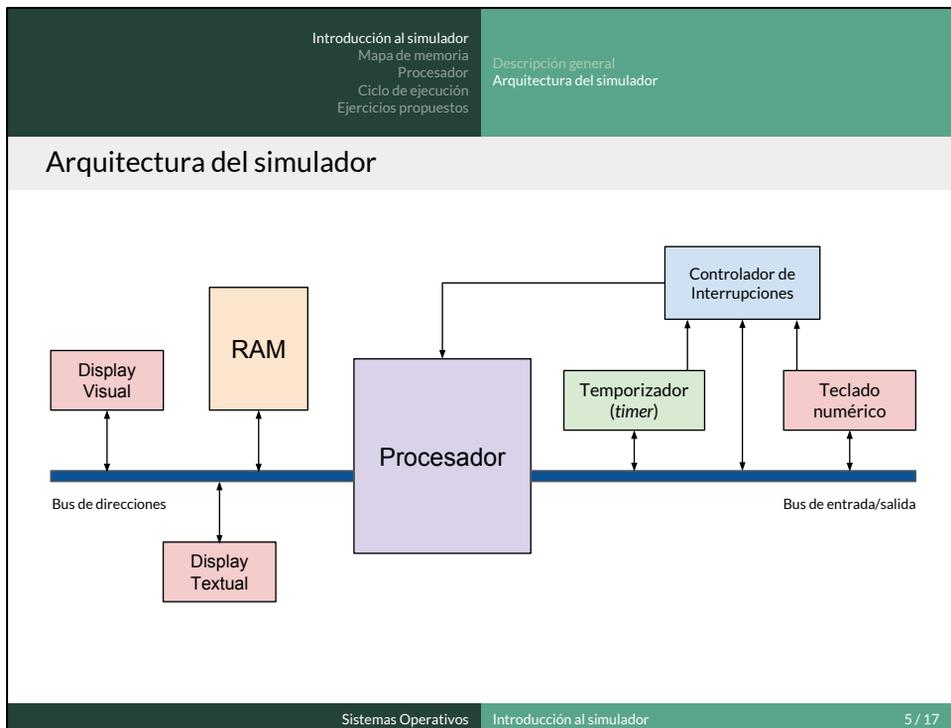
Dispositivos E/S

CPU (Registros)

Mapa de memoria

Mapa de E/S

Visor de eventos



Descripción básica de los componentes básicos del sistema simulado:

- **Procesador**
- **Memoria:** espacio de direcciones de 16 bits. Solo se usan los primeros 1024 bytes
- **Controlador de interrupciones:** centraliza las distintas fuentes de interrupción. Se explica en la siguiente práctica
- **Timer:** permite programar interrupciones (alarma). Se explica en la siguiente práctica
- **Dispositivos de E/S:** display visual, display textual y teclado numérico

## Mapa de memoria

Descripción general

- Mapa de memoria de 1024 bytes
- Permite la edición manual de celdas:

(Ctrl + click)  
ó  
(Cmd + click)

- Los valores de las celdas tienen formato **hexadecimal**

Memory																Split	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
000	2E	00	10	48	65	6C	6C	6F	20	57	6F	72	6C	64	21	00	
001	96	04	00	FF	06	02	00	03	06	03	02	F0	46	00	20	00	
002	3B	00	3B	01	06	01	00	00	0B	0A	00	02	0D	00	03	0A	
003	22	0E	22	10	2A	0C	00	02	36	00	28	43	01	43	00	47	
004	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
005	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
006	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
007	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
008	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
009	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

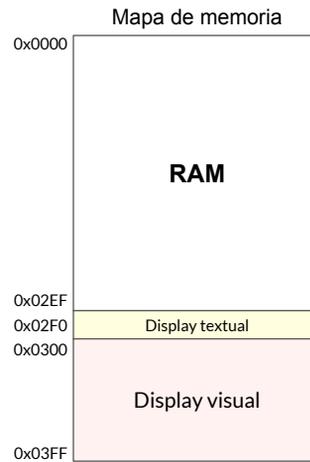
Ejemplos: edición manual de una o varias celdas

## Mapa de memoria

Proyecciones de dispositivos de E/S

El mapa de memoria está dividido en tres zonas:

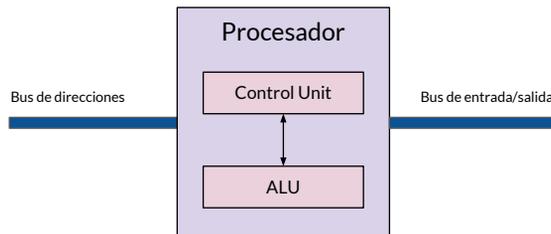
- **Zona de RAM:** memoria de lectura/escritura donde almacenar datos y código
- **Zona del display textual:** cada celda se corresponde con una letra. El valor será el carácter ASCII a mostrar
- **Zona del display visual:** cada celda se corresponde con un píxel. El valor será el color del píxel



Ejemplos: edición manual de una o varias celdas de las proyecciones del display textual y del display visual

## Procesador (CPU)

Descripción general



El procesador consta de:

- **Unidad de control (Control Unit):** se encarga de obtener de memoria, interpretar y ejecutar las instrucciones
- **Unidad Aritmético-Lógica (ALU):** realiza funciones aritméticas y lógicas a nivel de bit

## Procesador (CPU)

### Registros

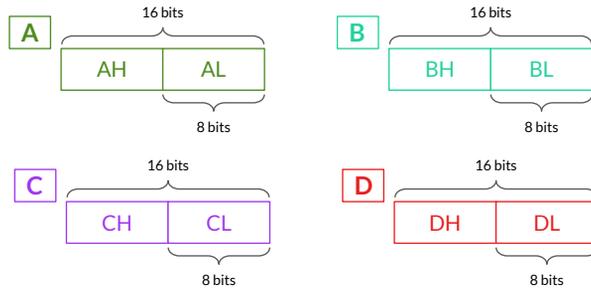
CPU Registers			
A	0000	B	0000
C	0000	D	0000
IP	0010	SSP	0000
SR	S - - - - - M C Z F H		
USP	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		

El procesador incluye tres tipos de registros:

- Registros de propósito general
- Registros de direcciones
- Registro de Estado (*Status Register*)

## Procesador (CPU)

### Registros de propósito general

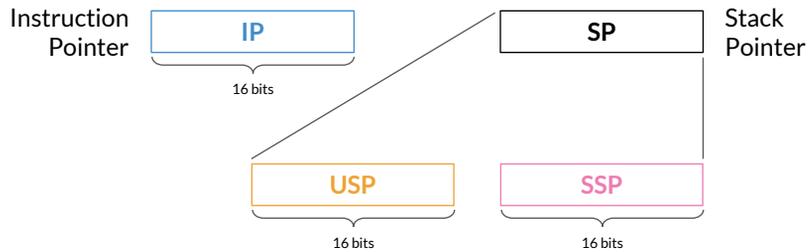


El procesador incluye cuatro registros de propósito general:

- Tienen un tamaño de 16 bits
- Pueden ser accedidos en modo palabra o modo byte

## Procesador (CPU)

Registros de direcciones



El procesador mantiene dos punteros de pila:

- Pila de modo supervisor (SSP)
- Pila de modo usuario (USP)

La selección de pila se hace de forma automática. Si la CPU está en modo supervisor, el registro SP es el SSP. Si está en modo usuario, el registro es el USP. Todas las instrucciones que direccionan el registro de pila de forma explícita, siempre se refieren al SP genérico. Ese SP es, por tanto, como un “alias” del verdadero registro puntero de pila.

Introducción al simulador  
Mapa de memoria  
**Procesador**  
Ciclo de ejecución  
Ejercicios propuestos

Descripción general  
**Registros**  
Juego de instrucciones

## Procesador (CPU)

Registro de estado

Status Register	s	M	C	Z	F	H
	1	0	0	0	0	0
	16 bits					

El registro de estado define los siguientes flags:

- **S:** Supervisor Mode
- **M:** Interrupt Mask
- **C:** Carry flag
- **Z:** Zero flag
- **F:** Fault mode
- **H:** Halt mode

Sistemas Operativos
Introducción al simulador
12 / 17

- **Supervisor Mode Flag.** Activado si la CPU está en modo supervisor y desactivado si está en modo usuario. La CPU arranca en modo supervisor. Los modos de operación se explican más en detalle prácticas posteriores.
- **Interrupt Mask Flag.** Activado si la CPU permite interrupciones y desactivado si las interrupciones están deshabilitadas. Se explica más en detalle en prácticas posteriores.
- **Carry Flag.** Se activa si la última operación aritmético-lógica ha producido acarreo. Por ejemplo, si la suma de dos enteros produce un desbordamiento o si la resta de dos enteros produce un número negativo.
- **Zero flag.** Se activa si la última operación aritmético-lógica ha dado un resultado de cero. Si el resultado último fue distinto de cero, el bit estará desactivado.
- **Fault mode flag.** Activado si la CPU se encuentra en modo fallo (error irrecuperable que requiere un reinicio).
- **Halt mode flag.** Activado si la CPU se encuentra en modo bloqueo. En este modo, la CPU sigue activa pero deja de ejecutar instrucciones hasta que se produce una interrupción. Se explica más en detalle en prácticas posteriores.

## Juego de instrucciones

### Formato de instrucción



- Una instrucción está formada por:
  - Un código de operación (*opcode*) de 8 bits
  - Ninguno, uno o dos operandos de 8 o 16 bits
- El código de operación determina el tipo de instrucción a ejecutar y los operandos que puedan necesitarse
- Cada instrucción está identificada por un mnemónico (e.g. MOV, ADD, etc)

Un mismo mnemónico (e.g. MOV) se puede utilizar para identificar una o varias instrucciones dependiendo de los distintos tipos de operandos. En ese caso el procesador define un opcode distinto para cada uno de los distintos conjuntos de operandos.

Nombre	Descripción	Tamaño
BYTE	Valor inmediato de 8 bits	1 byte
WORD	Valor inmediato de 16 bits	2 bytes
ADDRESS	Dirección de 16 bits	2 bytes
REGISTER_8BITS	Registro de 8 bits	1 byte
REGISTER_16BITS	Registro de 16 bits	1 byte
REGADDRESS	Direccionamiento por registro + offset	2 bytes

**REGISTER\_16BITS:** este operando codifica el número de referencia o índice de uno de los registros de 16 bits que implementa la CPU. Los índices de cada registro son los siguientes:

- A - Registro de propósito general A:** 0 (0x00)
- B - Registro de propósito general B:** 1 (0x01)
- C - Registro de propósito general C:** 2 (0x02)
- D - Registro de propósito general D:** 3 (0x03)
- SP - Registro puntero de pila:** 4 (0x04)

**REGISTER\_8BITS:** este operando codifica el número de referencia o índice de uno de los registros de 8 bits que implementa la CPU. Los índices de cada registro son los siguientes:

- AH - MSB del Registro A:** 9 (0x09)
- AL - LSB del Registro A:** 10 (0x0A)
- BH - MSB del Registro B:** 11 (0x0B)
- BL - LSB del Registro B:** 12 (0x0C)
- CH - MSB del Registro C:** 13 (0x0D)
- CL - LSB del Registro C:** 14 (0x0E)
- DH - MSB del Registro D:** 15 (0x0F)
- DL - LSB del Registro D:** 16 (0x10)

**REGADDRESS:** este operando codifica con 1 byte el número de referencia de un

registro de 16 bits y, con el otro byte, el offset a añadir al valor del parámetro para formar la dirección efectiva. El offset se codifica empleando complemento a dos [-128, 127].

## Juego de instrucciones

### Ejemplos

<b>opcode</b>	<b>Mnem.</b>	<b>Operando 1</b>	<b>Operando 2</b>	<b>Descripción</b>
6 (0x06)	MOV	REGISTER_16BITS	WORD	Mover un valor inmediato de 16 bits en un registro de 16 bits
33 (0x21)	INC	REGISTER_16BITS	-	Incrementar en una unidad el valor de un registro de 16 bits
5 (0x05)	MOV	ADDRESS	REGISTER_16BITS	Mover el valor de un registro de 16 bits a una dirección

Documentación sobre el juego de instrucciones

<http://asm-simulator.readthedocs.io/en/latest/instruction-set.html>

Codificar las instrucciones eligiendo un registro, un valor y una dirección de 16 bits. Editar las celdas a partir de la 0x0000 de forma manual y ejecutar las instrucciones paso a paso.

## Ciclo de ejecución de una instrucción

1. Carga del código de operación apuntado por el registro IP
  2. Decodificación del código de operación
  3. Carga de los operandos si los hubiere
  4. Resolución de direccionamiento por registro si lo hubiere
  5. Ejecución de la instrucción
  6. Actualización del registro IP
- El **log de eventos** muestra de forma detallada los pasos del ciclo

Ejemplo con el log de eventos de la ejecución de las instrucciones de la transparencia anterior.

Comentar paso a paso el Ejemplo 1 (Sample 1) del simulador. El código de este ejemplo escribe en el display textual la cadena de caracteres "Hello world!".

- Directivas de ensamblador (etiquetas y DB)
- Concepto de ensamblado (traducción a lenguaje máquina).
- Consecuencias del RESET (siempre comienza la ejecución por IP = 0)
- Ejecución paso a paso de CALL // RET. Ver que la dirección de retorno se almacena en la pila.
- Comprobar la modificación de los flags del registro de estado cuando se ejecuta la instrucción de comparación CMPB. Ver que la instrucción de salto condicional comprueba el valor del registro de estado para decidir si salta o no salta.

## Ejercicios propuestos

Implementar un programa que imprima en el display textual la cadena "Hello world!" pero pasando a mayúsculas las ocurrencias de la letra "o":

- El programa debe comprobar si la letra que se va a imprimir es una "o". En caso afirmativo, debe **pasarla a mayúscula antes de imprimirla**
- La distancia en ASCII entre una letra mayúscula y su correspondiente minúscula es de 0x20 posiciones

En una primera aproximación al ejercicio, modificar la cadena original por una cadena en minúsculas (e.g. "hello world") que se transforme a mayúsculas a medida que se imprimen los caracteres.

Proponer que revisen por su cuenta el código del Ejemplo 2 (Sample 2).