Introducción al simulador

Departamento de Automática

Universidad de Alcalá

/gso>

Índice

Descripción general

Sitio web

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

Proyecto github (código fuente)

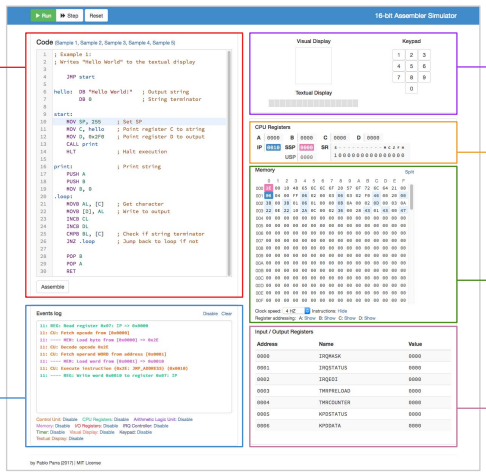
https://github.com/parraman/asm-simulator

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

Descripción general

Editor de código

Visor de eventos

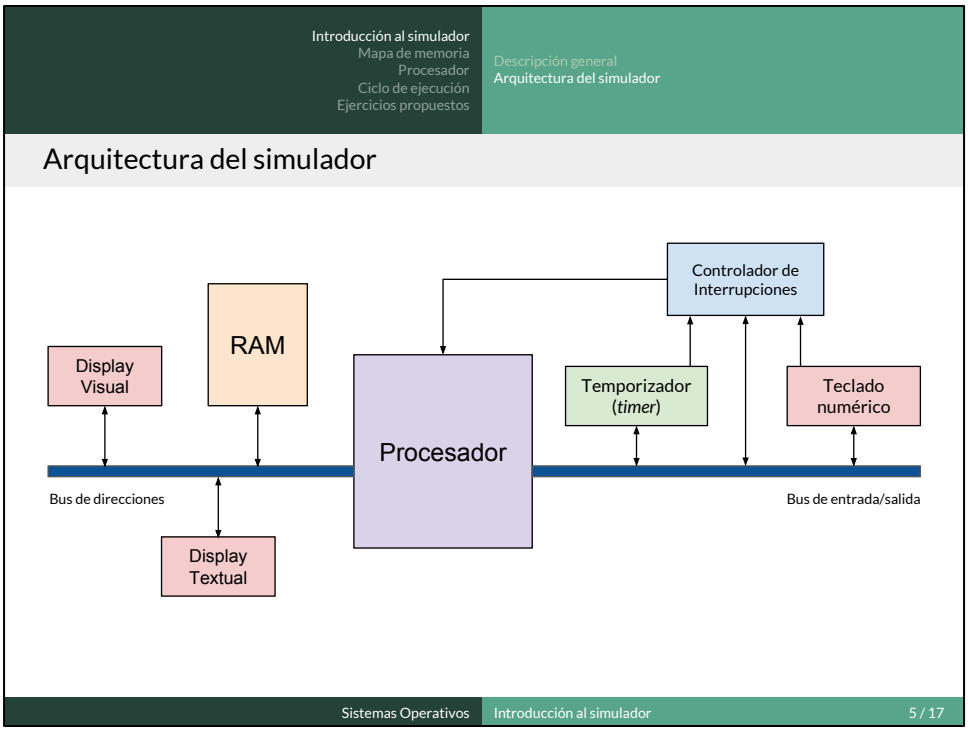


Dispositivos E/S

CPU (Registros)

Mapa de memoria

Mapa de E/S



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

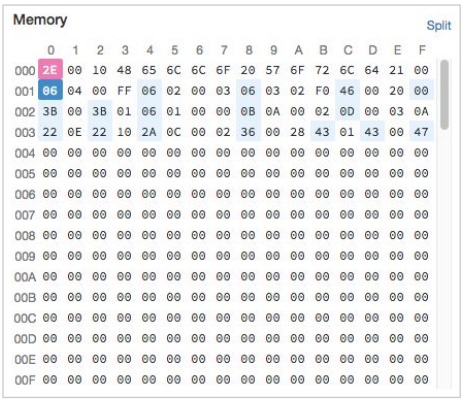
Mapa de memoria

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



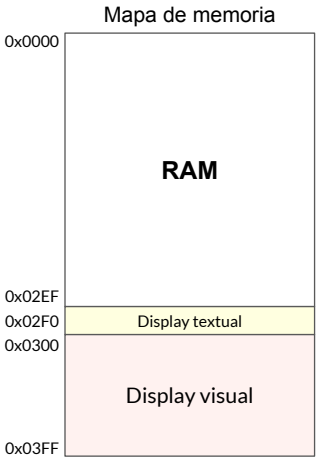
Ejemplos: edición manual de una o varias celdas

Mapa de memoria

Proyecciones de dispositivos de E/S

El mapa de memoria está dividido en tres zonas:

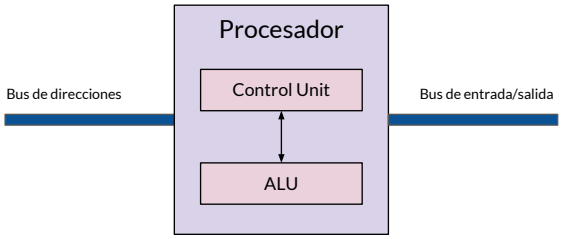
- Zona de RAM: memoria de lectura/escritura donde almacenar datos y código
- Zona del display textual: cada celda se corresponde con una letra. El valor será el carácter ASCII a mostrar
- Zona del display visual: cada celda se corresponde con un píxel. El valor será el color del píxel



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)

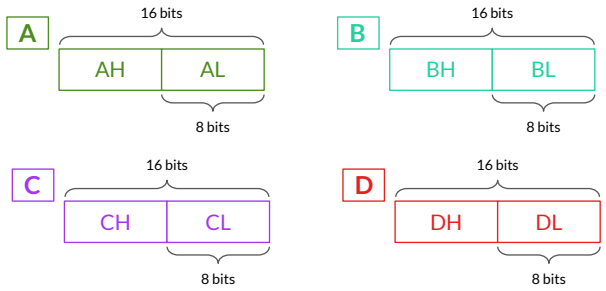
Registros

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

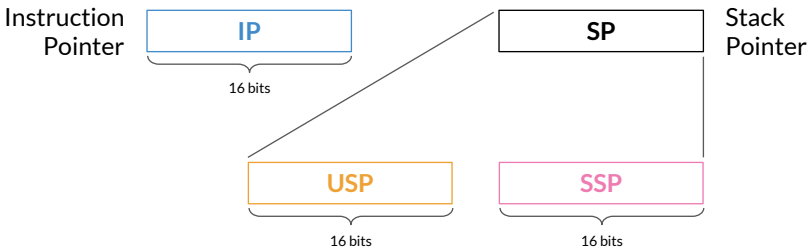


El procesador incluye cuatro registros de propósito general:

- 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:

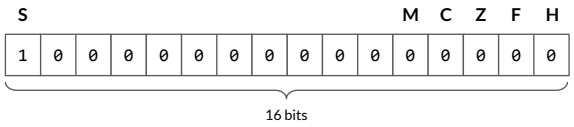
- Pila de modo supervisor (SSP)
- Pila de modo usuario (USP)

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.

Procesador (CPU)

Registro de estado

Status Register



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

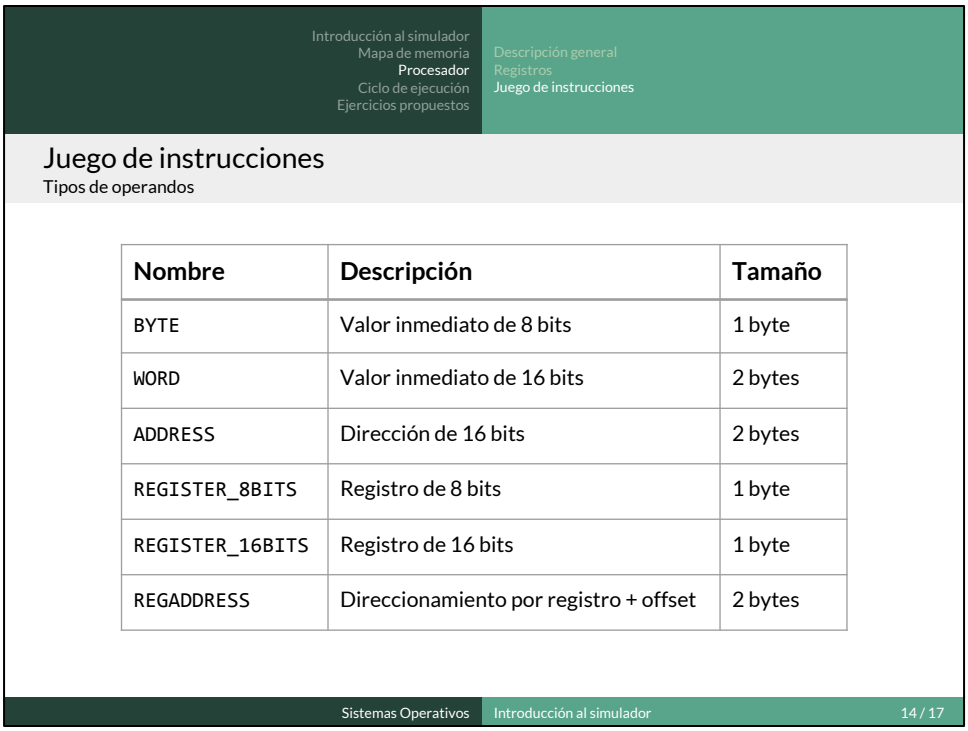
- **Supervisor Mode Flag**. Activado si la CPU está en modo supervisor y desactivado si está en modo usuario. La CPU arranca en modo supervisor. Los modos de operación se explican más en detalle prácticas posteriores.
- **Interrupt Mask Flag**. Activado si la CPU permite interrupciones y desactivado 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 activada 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
  - Niguno, 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.



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 complemente a dos [-128, 127].

Juego de instrucciones

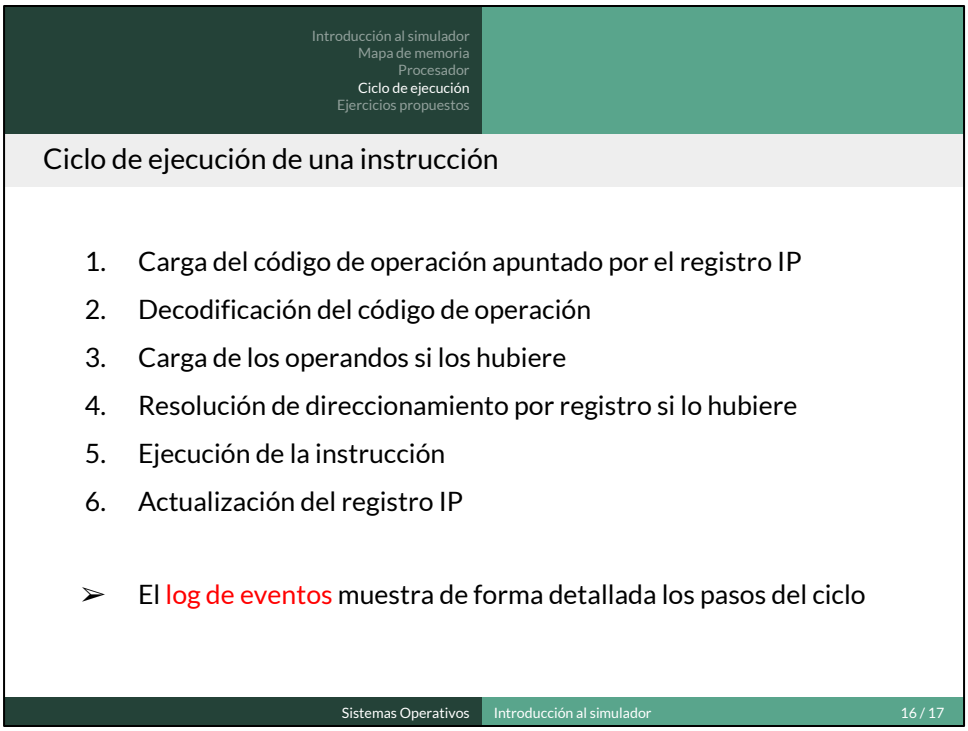
Ejemplos

opcode	Mnem.	Operando 1	Operando 2	Descripción
`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.



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).