Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

ENS2025

MANUAL DE USUARIO

versión 0.93 – Abril 2026

CONTENIDO

1 INTRODUCCIÓN

ENS2025 es una aplicación que integra la función de **Ensamblador** de un subconjunto de instrucciones del estándar *IEEE 694* y la función de **Simulador**, ya que es capaz de ejecutar programas ensamblados para dicha implementación particular del estándar. Se trata de una mejora de la herramienta anterior denominada *ENS2001*, e incluye una arquitectura de máquina virtual mejorada, una implementación del lenguaje más homogénea y, principalmente, una nueva y cómoda interfaz gráfica web.

El **Ensamblador** permite cargar desde un fichero el código fuente en ensamblador que será ejecutado, informando de su corrección o de los errores que pudiera contener, según el caso, como se puede ver en la Figura 1.

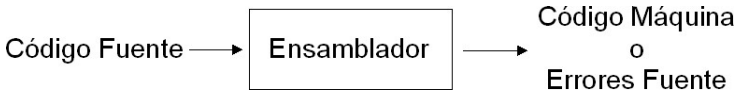


Figura 1. El Ensamblador de ENS2025

El **Simulador**, aparte de las diversas opciones de configuración y ejecución, proporciona las funciones de acceso a memoria, pila, banco de registros, código fuente y consola, tanto para consultar datos como para alterarlos manualmente. En la Figura 2 se muestra el diagrama de bloques simplificado para esta parte de la herramienta.

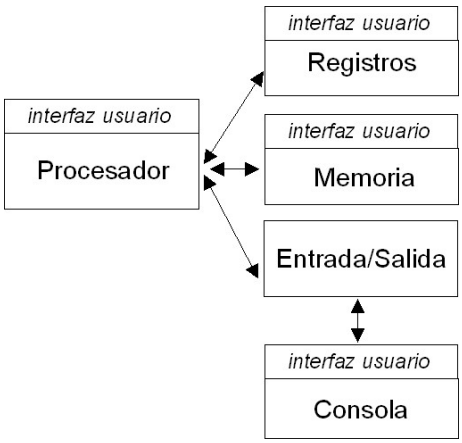


Figura 2. El Simulador ENS2025

En principio, la aplicación está dirigida a los alumnos de Traductores de Lenguajes, que deberán probar el código generado por sus respectivas prácticas, si bien también puede ser de utilidad, sobre todo didáctica, para todo aquél que esté interesado en iniciarse en el mundo de la programación en ensamblador, sin profundizar en ningún lenguaje para ningún microprocesador en concreto. En esta herramienta se hallará el complemento ideal para comprobar en un entorno simulado la ejecución de los programas que se vayan creando durante las etapas de aprendizaje.

A continuación, se describe el sistema que la herramienta simula y el lenguaje ensamblador ideado para dicho sistema. Posteriormente se ahondará en los detalles acerca de las características de la interfaz de la herramienta.

La herramienta puede utilizarse desde cualquier navegador web accediendo a la siguiente dirección:

https://dlsiis.fi.upm.es/ens2025

2 MÁQUINA VIRTUAL

La Máquina Virtual que simula la aplicación posee las siguientes características:

- **Procesador** con ancho de palabra de 16 bits.
- **Memoria** de 64 KPalabras (de 16 bits cada una). Por tanto, el direccionamiento es de 16 bits, coincidiendo con el ancho de palabra, desde la dirección 0 a la 65535 (FFFFh).
- **Banco de Registros**. Todos ellos son de 16 bits. Los registros se codifican internamente con 4 bits, valores de 0 a 15, según se muestra en la Tabla 2 y en el apartado 9.3.
  - **PC** (Contador de Programa): Indica la posición en memoria de la siguiente instrucción que se va a ejecutar.
  - **SP** (Puntero de Pila): Indica la posición de memoria donde se encuentra la cima libre de la pila.
  - **SR** (Registro de Estado): Almacena el conjunto de los biestables de estado, según se describe en la Tabla 1 y en el apartado 9.4.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	H	S	P	V	C	Z

Tabla 1. Biestables de Estado

  - **IX, IY** (Registros Índices): Se emplean para efectuar direccionamientos indexados.
  - **A** (Acumulador): Almacena el resultado de las operaciones aritméticas y lógicas de dos operandos.
  - **R0..R9** (Registros de Propósito General): Son registros cuyo uso decidirá el programador en cada momento.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC	SP	IY	IX	SR	A	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0

Tabla 2. Registros del Simulador

- **Biestables de estado:** Se actualizan después de que el procesador ha ejecutado una operación aritmética o de comparación. Su significado y condiciones de activación son los siguientes:
  - **Z (cero)**. Si el resultado de la operación es 0, se activa. En caso contrario, se desactiva.
  - **C (acarreo)**. Si el resultado excede los 16 bits de tamaño, se activa. En caso contrario, se desactiva.
  - **V (desbordamiento)**. Si el resultado de la operación excede el rango de representación de los enteros de 16 bits en complemento a 2 (desde –32768 hasta 32767), se activa. En caso contrario, se desactiva.
  - **P (paridad)**. Es el resultado de realizar un O Lógico Exclusivo con todos los bits del resultado. Así, se activa si el número de bits que valen 1 es impar y se desactiva si el número de bits que valen 1 es par.
  - **S (signo)**. Se activa cuando el resultado de la operación es negativo y se desactiva si es positivo. En otras palabras, indica el valor del bit más significativo del resultado de la operación (como la ALU (Unidad Aritmético-Lógica) opera en complemento a 2, será 0 para valores positivos y 1 para valores negativos).
  - **H (fin de ejecución)**. Se activa únicamente después de que el procesador haya ejecutado una instrucción de parada (`HALT`).
- **Entrada/Salida** por consola: Se proveen instrucciones para leer y escribir enteros, caracteres y cadenas de caracteres acabadas en el carácter nulo (‘`\0`’).
- **Unidad Aritmético-Lógica (ALU)** sobre enteros en complemento a 2 de 16 bits. El juego de instrucciones proporciona operaciones de suma, resta, multiplicación, división, módulo, cambio de signo, incremento, decremento, *or* lógico, *and* lógico, *or* exclusivo lógico y negación lógica.

Al operar en complemento a 2, los enteros negativos (en el rango –32768..–1) se transforman en los correspondientes positivos (32768..65535 respectivamente). Por lo tanto, al introducirlos, bien en el código fuente, bien durante la ejecución, serán tratados indistintamente por la herramienta, tanto en el ensamblador como durante la simulación.

- Generación de **Excepciones**: Durante la simulación se pueden producir las siguientes condiciones de excepción, que detendrán la ejecución:
  - *Instrucción no implementada*. Ocurre cuando el procesador lee un código de operación que no corresponde con ninguna instrucción de su juego de instrucciones. Esto puede ocurrir si el usuario introduce código directamente editando la memoria, si el propio programa modifica la memoria de la zona de código o si no se posiciona el contador de programa adecuadamente para la ejecución (por ejemplo, en zonas de datos o en una dirección de memoria intermedia dentro de una instrucción que ocupe varias posiciones).
  - *División por cero*. Ocurre cuando el segundo operando de una operación de división o módulo (el divisor) toma el valor cero.
  - *Sobrepasado el límite de la memoria*. Ocurre cuando, tras ejecutar una instrucción, el Contador de Programa toma un valor más alto que el límite superior de memoria. O bien cuando se está ejecutando una instrucción `INSTR` o `WRSTR` y la cadena sobrepasa dicho límite.
  - *Modo Argumento Inválido*. Ocurre cuando el procesador lee un código de un modo de direccionamiento no permitido para el código de operación al que acompaña. Esto puede ocurrir si el usuario introduce código directamente editando la memoria, si el programa modifica la región de código o si no se posiciona el contador de programa adecuadamente para la ejecución (por ejemplo, en zonas de datos o en una dirección de memoria intermedia dentro de una instrucción que ocupe varias posiciones).
  - *Registro Inválido.* Ocurre cuando el procesador lee un código de registro inválido como parámetro para una operación que pide un registro como argumento. Esto puede ocurrir si el usuario introduce código directamente editando la memoria, si el programa modifica la región de código o si no se posiciona el contador de programa adecuadamente para la ejecución (por ejemplo, en zonas de datos o en una dirección de memoria intermedia dentro de una instrucción que ocupe varias posiciones).
  - *Demasiadas Instrucciones Ejecutadas.* Ocurre cuando el procesador ejecuta un numero de instrucciones superior al límite de instrucciones permitido. Esto ocurre para evitar bucles infinitos. Por defecto, esta excepción salta cuando el número de instrucciones ejecutadas supera las 10.000.
  - *Puntero de pila invade la zona de código.* Se genera un aviso si el SP va a tomar un valor comprendido dentro de los límites del código almacenado en memoria.
  - *Ejecución detenida por el usuario.* El usuario puede detener la simulación en cualquier momento, generándose esta excepción.

3 ESTRUCTURA DEL CÓDIGO FUENTE

El código fuente se lee desde un **fichero de texto plano**. Esta es la forma común de enviar programas para ejecutar en la aplicación, aunque también dispone de un pequeño editor en el que escribir directamente el programa en ensamblador. Evidentemente, es también posible introducir el código ensamblado a mano modificando directamente las posiciones de memoria, lo cual resulta bastante más tedioso y peligroso ya que no se comprobará si se está cometiendo algún error (además no se actualizarán los límites de la zona de código). En la Figura 3 se muestra la arquitectura del módulo ensamblador. Se parte de un fichero donde está almacenado el código fuente y se genera en dos pasadas el código máquina o una lista de errores en el programa introducido.

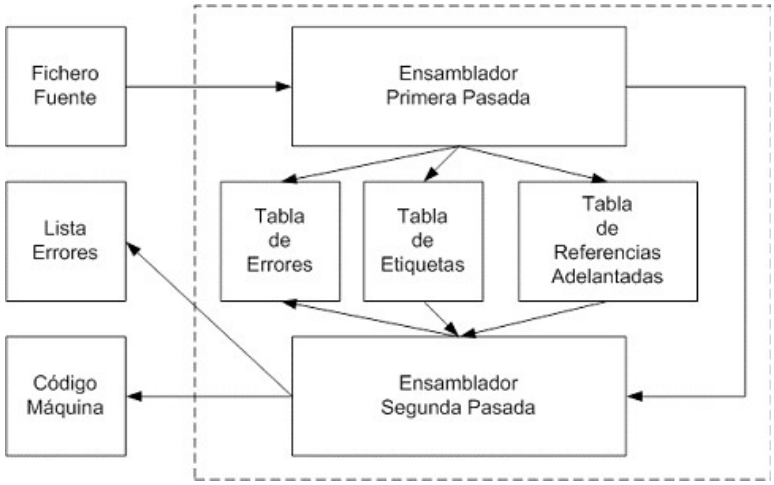


Figura 3. Arquitectura del Ensamblador

El código fuente se compone de una sucesión de líneas que obedecen al siguiente formato:

```
[Etiqueta ':'] [Instrucción] [';' Comentario]
```

No existe límite para el número de caracteres que pueda ocupar una línea (salvo las restricciones propias del sistema operativo o la memoria disponible). Son perfectamente posibles combinaciones tales como introducir líneas en blanco, comentarios tras las instrucciones, líneas únicamente con comentarios, etc.

Además, excepcionalmente, se permite introducir saltos de línea y comentarios entre etiquetas e instrucciones, en aras de aumentar la legibilidad del código. La única restricción "real" es que cada instrucción estará contenida en una única línea, ni más ni menos (es decir, una instrucción sólo ocupa una línea y en cada línea sólo hay una instrucción como mucho). Por ejemplo:

```
BZ .R3 ; saltar si el contador es cero          Correcto
; esta línea sólo tiene comentarios             Correcto
ADD #300 ; primer op , [.R0] ; segundo op       Erróneo
```

El ensamblador distingue entre mayúsculas y minúsculas a la hora de definir etiquetas. En los casos especiales de las instrucciones y los nombres de registros, no se distingue entre mayúsculas y minúsculas. Por ejemplo, las siguientes tres instrucciones son correctas:

```
NOP
nop
Nop
```

Las etiquetas deben escribirse usando solo letras, dígitos o subrayados ('_'), no pudiendo comenzar por un dígito. Si en una línea solo se coloca una etiqueta, la dirección a la que se asociará es la de la siguiente instrucción que aparezca en el código.

El formato general de las instrucciones es el siguiente:

```
Mnemónico [Operando1 [Operando2]]
```

Un mnemónico es una **palabra** o una **abreviatura** en inglés que hace referencia a la operación.

Los operandos pueden contener **enteros**, **nombres de registros** o **etiquetas**. En cualquier caso, se deben ajustar a alguno de los modos de direccionamiento que se analizan en el siguiente apartado. No todas las instrucciones admiten todos los modos de direccionamiento posibles. Las combinaciones

permitidas se verán al estudiar con detalle el juego de instrucciones y pueden consultarse en el apartado 9.5.

4 MODOS DE DIRECCIONAMIENTO

El estándar *IEEE 694* define un conjunto de modos de direccionamiento para determinar los distintos operandos o su ubicación. La máquina simulada por *ENS2025* permite **siete** de ellos. Dependiendo de cada instrucción, se pueden usar unos u otros en los operandos. Los modos de direccionamiento son los siguientes:

4.1 DIRECCIONAMIENTO INMEDIATO

El direccionamiento es inmediato cuando el operando se encuentra **contenido** en la propia instrucción. Por tanto, dadas las características de la máquina virtual, internamente serán necesarios 16 bits para representar este tipo de direccionamiento.

Para indicar este modo de direccionamiento, se empleará el carácter '**#**' (almohadilla) precediendo al operando, que en este caso indicará un valor entero de 16 bits en complemento a 2.

Se pueden introducir valores decimales con signo, en el intervalo comprendido entre –32768 y 32767, o bien sin signo en el rango 0..65535. También se permite introducir valores hexadecimales, precedidos por el prefijo '`0x`', en el rango 0..FFFFh.

Ejemplos:

- La instrucción de carga `MOVE #1000, .R1`, transfiere el valor 1000d al registro R1, como se muestra en la Figura 4.

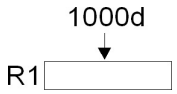


Figura 4. Direccionamiento Inmediato (ejemplo 1)

- La instrucción de suma `ADD .R3, #0x00FF` suma el contenido de R3 con el valor FFh y lo almacena en el acumulador, tal y como se aprecia en la Figura 5.

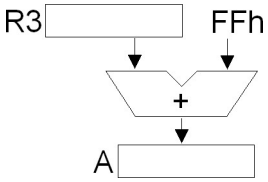


Figura 5. Direccionamiento Inmediato (ejemplo 2)

4.2 DIRECCIONAMIENTO DIRECTO A REGISTRO

El direccionamiento es directo a registro cuando expresa el **nombre del registro** donde se encuentra almacenado el objeto. Dado el banco de registros definido para la máquina virtual, internamente serán necesarios 4 bits para este tipo de direccionamiento (ya que el banco de registros contiene 16 registros en total).

Para indicar este modo de direccionamiento, se empleará el carácter '**.**' (punto) precediendo al operando, que en este caso será el nombre de un registro del banco de registros.

Por ejemplo, en la instrucción de resta `SUB .R3, #0xFF`, el primer operando emplea un direccionamiento directo a registro, ya que se está indicando el nombre del registro donde se encuentra almacenado. El segundo operando emplea direccionamiento inmediato, de la misma forma

que el ejemplo anterior (Figura 5). Dicha instrucción resta al contenido del registro R3 el valor inmediato FFh y lo almacena en el acumulador.

4.3 DIRECCIONAMIENTO DIRECTO A MEMORIA

El direccionamiento es directo a memoria cuando se expresa la **dirección absoluta de memoria** donde se encuentra almacenado el operando. Como la memoria consta de 64 KPalabras, internamente se necesitan 16 bits para este tipo de direccionamiento.

Para indicar este modo de direccionamiento, se empleará el carácter '**/**' (barra) precediendo al operando, que en este caso indicará una dirección de memoria. Como en el caso del direccionamiento inmediato, se permiten valores decimales sin signo en el rango 0..65535 o con signo, en el intervalo –32768..32767. En el caso de que se introduzcan decimales con signo, se traducirán a la dirección que se correspondería con su representación en complemento a 2 (en 16 bits); por ejemplo, `-1` se corresponde con 65535, `-2` con 65534 y así sucesivamente. También se permite introducir valores hexadecimales, precedidos por el prefijo '`0x`', en el rango 0..FFFFh. Por último, se puede utilizar el nombre de una etiqueta en vez de una dirección numérica.

Por ejemplo, para la instrucción de incremento `INC /1000`, el resultado será que el valor contenido en la posición de memoria 1000d se habrá incrementado en una unidad, tal y como se muestra en la Figura 6.

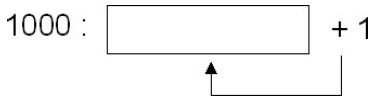


Figura 6. Direccionamiento Directo a Memoria

En el caso de haber introducido `INC /-30`, por ejemplo, el ensamblador haría la conversión pertinente, y el operando se correspondería con la celda de memoria ubicada en la dirección 65506 (que es la representación en complemento a 2 con 16 bits del valor –30). Por tanto, dicha instrucción es la misma que `INC /65506`.

4.4 DIRECCIONAMIENTO INDIRECTO

El direccionamiento es indirecto cuando se tiene la **dirección donde se encuentra el operando** (en vez del operando directamente). Dicha dirección se encontrará almacenada en un registro. Por tanto, se requerirá de dos accesos (el primero al banco de registros y el segundo a memoria principal) para recuperar el objeto. Internamente se requieren 4 bits para indicar cuál es el registro a partir del que se recupera la dirección de memoria destino.

Para indicar este modo de direccionamiento, se encierra entre corchetes '**[ ]**' el nombre del registro sobre el que se efectúa la indirección en notación de registro, esto es, precedido por el carácter '**.**' (punto).

Por ejemplo, la instrucción de decremento `DEC [.R3]` decrementará en una unidad el contenido de la posición de memoria contenida en el registro R3. Se puede ver con claridad en la Figura 7.

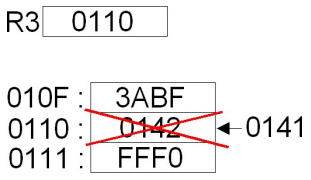


Figura 7. Direccionamiento Indirecto

4.5 DIRECCIONAMIENTO INDEXADO

El direccionamiento es indexado (o relativo a un registro) cuando la instrucción contiene un **desplazamiento** sobre una **dirección** marcada por un **puntero** (almacenado en un registro). La dirección del operando, por tanto, se calcula sumando el desplazamiento al puntero de referencia.

Para indicar este modo de direccionamiento, se empleará el carácter '**#**' (almohadilla), que en este caso indicará un desplazamiento (entero de 16 bits en complemento a 2, esto es, valores comprendidos entre –32768 y 32767) y, a continuación, en notación de indirección '**[ ]**' (entre corchetes), el nombre del registro. El valor del desplazamiento se puede introducir, como cualquier otro entero, bien en base decimal, bien en base hexadecimal, siguiendo las mismas consideraciones que en los casos anteriores de direccionamiento inmediato y directo a memoria.

ENS2025 permite el uso del direccionamiento indexado con los registros índice IX e IY y con el puntero de pila SP. Los desplazamientos están restringidos a enteros de 16 bits en complemento a 2. A efectos prácticos, se considerarán indexado a IX, indexado a IY e indexado a SP como modos de direccionamiento distintos. En el caso del direccionamiento indexado a SP, la codificación interna del modo de direccionamiento es la misma que la del direccionamiento relativo a PC, y el modo de direccionamiento empleado depende de la instrucción en la que se utilice.

Ejemplos:

- En la instrucción de carga `MOVE #6[.IX], .R1`, se carga en el registro R1 el contenido de la posición de memoria a la que hace referencia el índice IX más 6 posiciones (Figura 8).

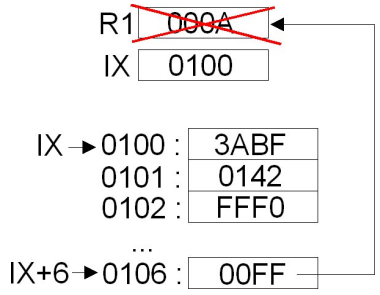


Figura 8. Direccionamiento Indexado

4.6 DIRECCIONAMIENTO RELATIVO A CONTADOR DE PROGRAMA

Se trata de un caso particular de direccionamiento indexado. El registro **puntero** es el **contador de programa (PC)**, y se indica el desplazamiento, que como en el caso general, se trata de un valor entero de 16 bits en complemento a 2.

Así, la dirección de la próxima instrucción que se va a ejecutar, se obtiene sumando el desplazamiento al valor del registro PC. En la práctica, el desplazamiento indica una cantidad entera que se va a sumar al contador de programa para reubicarlo antes de la ejecución de la siguiente instrucción.

Para indicar este modo de direccionamiento, se empleará el carácter '**$**' (símbolo de dólar) precediendo al operando, que se trata del desplazamiento. El formato del desplazamiento es idéntico al caso general (enteros comprendidos entre –32768 y 32767).

Por ejemplo, en la instrucción de salto incondicional `BR $3` de la Figura 9, la siguiente instrucción que ejecutará el procesador se encuentra en la posición de memoria a la que apunte el contador de programa más 3 posiciones. Hay que recordar que el contador de programa apunta siempre a la siguiente instrucción de la que se está ejecutando.

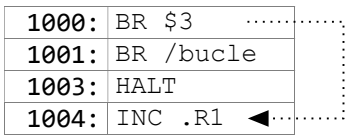


Figura 9. Direccionamiento Relativo a PC

Por tanto, en la Figura 9, la siguiente instrucción que se ejecutaría sería `INC .R1` (y no `HALT` como se podría pensar), ya que en el momento de ejecutar `BR $3`, PC vale 1001, así que PC = 1001+3.

En las instrucciones de salto es muy frecuente el empleo de etiquetas, tanto si el direccionamiento es directo a memoria como relativo a contador de programa. No obstante, en el caso de direccionamiento relativo, la etiqueta no se traduce por su valor, si no que se calcula el desplazamiento entre la posición a la que apuntaría PC en el caso de seguir el flujo de ejecución y la posición que ocupa la etiqueta. Por ello, si se introduce en el código fuente un direccionamiento relativo a contador de programa hacia una etiqueta que está alejada más de 32768 posiciones de memoria (32767 hacia adelante, 32768 hacia atrás, cantidades representables con 16 bits), el ensamblador devolverá un error en tiempo de ensamblado, ya que no puede calcular un desplazamiento válido.

Cambiando el caso anterior, empleando etiquetas en vez de números enteros, quedaría representando en la Figura 10. La etiqueta `Seguir` tomaría el valor 1004 y el valor del desplazamiento seguiría siendo 3.

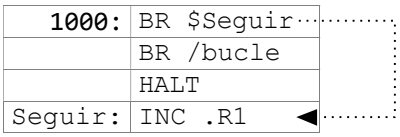


Figura 10. Direccionamiento Relativo a PC con etiquetas

El código interno del modo de direccionamiento relativo a PC es el mismo que el del modo de direccionamiento indexado a SP. El modo de direccionamiento empleado a la hora de decodificar la instrucción depende de qué instrucción esté empleando el modo de direccionamiento, de tal manera que las instrucciones de salto y llamada utilizan un direccionamiento relativo a PC mientras que las demás instrucciones realizarían un direccionamiento indexado a SP.

5 JUEGO DE INSTRUCCIONES

A continuación, se enumeran todas las instrucciones que componen el juego de instrucciones, ordenadas por código de operación. En el apartado 9.5 se puede consultar la relación completa de instrucciones y los modos de direccionamiento que soportan. Este juego de instrucciones está basado en el juego de instrucciones de un micro-procesador real.

5.1 INSTRUCCIÓN NULA Y DE PARADA

Instrucción	NOP
Descripción	Instrucción de no operación
Formato	NOP
Código de Operación	0
Número de Operandos	0
Modos de Direccionamiento	N/A
Comportamiento	No hace nada. No modifica los biestables de estado.

Instrucción	HALT
Descripción	Detiene la ejecución de la máquina
Formato	HALT
Código de Operación	1
Número de Operandos	0
Modos de Direccionamiento	N/A
Comportamiento	Activa el biestable H de fin de programa y detiene la ejecución.

5.2 TRANSFERENCIA DE DATOS

Instrucción	MOVE
Descripción	Copia el operando origen en el operando destino
Formato	MOVE op1, op2
Código de Operación	2
Número de Operandos	2
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado op2: registro, memoria, indirecto, indexado
Comportamiento	Lee el contenido del operando 1 (origen) y lo escribe en el operando 2 (destino). No modifica los biestables de estado.

Instrucción	PUSH
Descripción	Pone en la pila un valor
Formato	PUSH op1
Código de Operación	3
Número de Operandos	1
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Depende del modo de funcionamiento de la pila: • Con crecimiento hacia direcciones descendentes de memoria, almacena el contenido del operando 1 en la dirección apuntada por SP y decrementa el valor del puntero de pila. • Con crecimiento hacia direcciones ascendentes de memoria, incrementa el valor del puntero de pila y almacena el contenido del operando 1 en la dirección apuntada por SP. No modifica los biestables de estado.

Instrucción	POP
Descripción	Saca de la pila un valor
Formato	POP op1
Código de Operación	4
Número de Operandos	1
Modos de Direccionamiento	op1: registro, memoria, indirecto, indexado
Comportamiento	Depende del modo de funcionamiento de la pila: • Con crecimiento hacia direcciones descendentes de memoria, incrementa el valor del puntero de pila y almacena en el operando 1 el valor contenido en la dirección de memoria apuntada por SP. • Con crecimiento hacia direcciones crecientes de memoria, almacena en el operando 1 el valor contenido en la dirección de memoria apuntada por SP y decrementa el valor del puntero de pila. No modifica los biestables de estado.

5.3 ARITMÉTICAS

- De dos operandos

Instrucción	ADD
Descripción	Suma números enteros
Formato	ADD op1, op2
Código de Operación	5
Número de Operandos	2
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado op2: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Suma el contenido del operando 1 y del operando 2, y almacena el resultado en el registro acumulador (A). Modifica los biestables de estado.

Instrucción	SUB
Descripción	Resta números enteros
Formato	SUB op1, op2
Código de Operación	6
Número de Operandos	2
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado op2: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Resta el contenido del operando 1 menos el contenido del operando 2 (*op1 – op2*), y almacena el resultado en el registro acumulador (A). Modifica los biestables de estado.

Instrucción	MUL
Descripción	Producto de números enteros
Formato	MUL op1, op2
Código de Operación	7
Número de Operandos	2
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado op2: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Multiplica el contenido del operando 1 y del operando 2, y almacena el resultado en el registro acumulador (A). Modifica los biestables de estado.

Instrucción	DIV
Descripción	Cociente de la división de números enteros
Formato	DIV op1, op2
Código de Operación	8
Número de Operandos	2
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado op2: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Divide el contenido del operando 1 entre el contenido del operando 2, y almacena el cociente de la operación en el registro acumulador (A). Si el operando 2 contiene el valor 0, se genera una excepción de tipo "división por cero". Si la división no devuelve un valor entero exacto, el resultado se trunca (*2 / 3 = 0*). Modifica los biestables de estado.

Instrucción	`MOD`
Descripción	Resto de la división de números enteros
Formato	`MOD op1, op2`
Código de Operación	9
Número de Operandos	2
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado op2: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Divide el contenido del operando 1 entre el contenido del operando 2, y almacena el resto de la operación en el registro acumulador. Si el operando 2 contiene el valor 0, se genera una excepción de tipo "división por cero". Modifica los biestables de estado.

- De un operando

Instrucción	`INC`
Descripción	Incremento unitario
Formato	`INC op1`
Código de Operación	10
Número de Operandos	1
Modos de Direccionamiento	op1: registro, memoria, indirecto, indexado
Comportamiento	Incrementa el contenido del operando en una unidad. Modifica los biestables de estado.

Instrucción	`DEC`
Descripción	Decremento unitario
Formato	`DEC op1`
Código de Operación	11
Número de Operandos	1
Modos de Direccionamiento	op1: registro, memoria, indirecto, indexado
Comportamiento	Decrementa el contenido del operando en una unidad. Modifica los biestables de estado.

Instrucción	`NEG`
Descripción	Cambio de signo
Formato	`NEG op1`
Código de Operación	12
Número de Operandos	1
Modos de Direccionamiento	op1: registro, memoria, indirecto, indexado
Comportamiento	Modifica el operando cambiando su signo. Modifica los biestables de estado.

5.4 COMPARACIONES

Instrucción	`CMP`
Descripción	Comparación entre enteros
Formato	`CMP op1, op2`
Código de Operación	13
Número de Operandos	2
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado op2: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Compara los operandos, realizando la resta del contenido del operando 1 menos el contenido del operando 2 (pero no almacena el resultado de la operación en ningún sitio). Modifica los biestables de estado en función del resultado.

5.5 LÓGICAS

- De dos operandos

Instrucción	AND
Descripción	Y Lógico
Formato	`AND op1, op2`
Código de Operación	14
Número de Operandos	2
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado op2: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Efectúa la operación 'y lógico' bit a bit entre el contenido del operando 1 y el operando 2, y almacena el resultado en el registro acumulador (A). La tabla de verdad de la operación con cada bit es: **op1** **op2** **op1 AND op2** 0 0 0 0 1 0 1 0 0 1 1 1 No modifica los biestables de estado.

Instrucción	OR
Descripción	O Lógico
Formato	`OR op1, op2`
Código de Operación	15
Número de Operandos	2
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado op2: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Efectúa la operación 'o lógico' bit a bit entre el contenido del operando 1 y el operando 2, y almacena el resultado en el registro acumulador (A). La tabla de verdad de la operación con cada bit es: **op1** **op2** **op1 OR op2** 0 0 0 0 1 1 1 0 1 1 1 1 No modifica los biestables de estado.

Instrucción	XOR
Descripción	O Exclusivo Lógico
Formato	`XOR op1, op2`
Código de Operación	16
Número de Operandos	2
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado op2: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Efectúa la operación 'o exclusivo lógico' bit a bit entre el contenido del operando 1 y el operando 2, y almacena el resultado en el registro acumulador (A). La tabla de verdad de la operación con cada bit es: **op1** **op2** **op1 XOR op2** 0 0 0 0 1 1 1 0 1 1 1 0 No modifica los biestables de estado.

- De un operando

Instrucción	NOT
Descripción	Negación Lógica
Formato	NOT op1
Código de Operación	17
Número de Operandos	1
Modos de Direccionamiento	op1: registro, memoria, indirecto, indexado
Comportamiento	Efectúa la operación 'negación lógica' bit a bit en el operando 1. La tabla de verdad de la operación con cada bit es: **op1** | **NOT op1** 0 | 1 1 | 0 Hay que destacar que la operación se realiza bit a bit, no con el valor completo. De esta forma, un NOT aplicado al número 1d (0000 0000 0000 0001b), devolverá -2d (1111 1111 1111 1110b). No modifica los biestables de estado.

5.6 BIFURCACIONES

Todas las instrucciones de bifurcación producen un salto a la posición de memoria indicada por el operando, dependiendo de la instrucción y el contenido de los biestables de estado.

Instrucción	BR
Descripción	Bifurcación incondicional
Formato	BR op1
Código de Operación	18
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1. Carga el PC con el valor contenido en el operando 1.

Instrucción	BZ
Descripción	Bifurcación si resultado igual cero
Formato	BZ op1
Código de Operación	19
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa resultó ser cero. Si el biestable Z está activo (Z=1), carga el PC con el valor contenido del operando 1.

Instrucción	BNZ
Descripción	Bifurcación si resultado distinto de cero
Formato	BNZ op1
Código de Operación	20
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa resultó ser distinta a cero. Si el biestable Z está inactivo (Z=0), carga el PC con el valor contenido en el operando 1.

Instrucción	BP
Descripción	Bifurcación si resultado positivo
Formato	BP op1
Código de Operación	21
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa resultó ser un valor positivo. Si el biestable S está inactivo (S=0), carga el PC con el valor contenido en el operando 1.

Instrucción	BNP
Descripción	Bifurcación si resultado no positivo
Formato	BNP op1
Código de Operación	22
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa resultó ser un valor no positivo. Si los biestables S o Z están activos (S=1 o Z=1), carga el PC con el valor contenido en el operando 1.

Instrucción	BN
Descripción	Bifurcación si resultado negativo
Formato	BN op1
Código de Operación	23
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa resultó ser un valor negativo. Si el biestable S está activo (S=1), carga el PC con el valor contenido en el operando 1.

Instrucción	BNN
Descripción	Bifurcación si resultado no negativo
Formato	BNN op1
Código de Operación	24
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa resultó ser un valor no negativo. Si el biestable S está inactivo (S=0) o el biestable Z está activo (Z=1), carga el PC con el valor contenido en el operando 1.

Instrucción	BV
Descripción	Bifurcación si hay desbordamiento
Formato	BV op1
Código de Operación	25
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa provocó un desbordamiento. Si el biestable V está activo (V=1), carga el PC con el valor contenido en el operando 1.

Instrucción	BNV
Descripción	Bifurcación si no hay desbordamiento
Formato	BNV op1
Código de Operación	26
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa no provocó un desbordamiento. Si el biestable V está inactivo (V=0), carga el PC con el valor contenido en el operando 1.

Instrucción	BC
Descripción	Bifurcación si hay acarreo
Formato	BC op1
Código de Operación	27
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa provocó acarreo. Si el biestable C está activo (C=1), carga el PC con el valor contenido en el operando 1.

Instrucción	BNC
Descripción	Bifurcación si no hay acarreo
Formato	BNC op1
Código de Operación	28
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa no provocó acarreo. Si el biestable C está inactivo (C=0), carga el PC con el valor contenido en el operando 1.

Instrucción	BE
Descripción	Bifurcación si el resultado tiene paridad par
Formato	BE op1
Código de Operación	29
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa tuvo un resultado con paridad par. Si el biestable P está inactivo (P=0), carga el PC con el valor contenido en el operando 1.

Instrucción	BO
Descripción	Bifurcación si el resultado tiene paridad impar
Formato	BO op1
Código de Operación	30
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Salta la ejecución a la dirección indicada en el operando 1 si una operación previa tuvo un resultado con paridad impar. Si el biestable P está activo (P=1), carga el PC con el valor contenido en el operando 1.

5.7 CONTROL DE SUBRUTINAS

Instrucción	`CALL`
Descripción	Llamada a subrutina
Formato	`CALL op1`
Código de Operación	31
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, relativo a contador de programa, indirecto
Comportamiento	Almacena en la pila el valor del contador de programa, actualiza el SP y salta a la dirección de destino indicada por el operando 1. Carga el PC con el valor contenido en el operando 1.

Instrucción	`RET`
Descripción	Retorno de subrutina
Formato	`RET`
Código de Operación	32
Número de Operandos	0
Modos de Direccionamiento	N/A
Comportamiento	Rescata de la pila el contenido del contador de programa, eliminándolo de la pila y actualizando el SP. Carga el PC con el valor contenido en la cima de la pila.

5.8 ENTRADA/SALIDA

Instrucción	`INCHAR`
Descripción	Lee un carácter
Formato	`INCHAR op1`
Código de Operación	33
Número de Operandos	1
Modos de Direccionamiento	op1: registro, memoria, indirecto, indexado
Comportamiento	Lee un carácter introducido por el teclado en la consola y lo codifica en *ASCII*, dejando a cero los 8 bits superiores de la palabra de 16 bits. Almacena el carácter leído en el operando 1.

Instrucción	`ININT`
Descripción	Lee un entero
Formato	`ININT op1`
Código de Operación	34
Número de Operandos	1
Modos de Direccionamiento	op1: registro, memoria, indirecto, indexado
Comportamiento	Lee un entero introducido por el teclado en la consola. Si la entrada no puede ser convertida en un número entero (porque se sale de rango o no es un número), la instrucción supondrá que se ha leído un cero. Los enteros se pueden introducir en formato decimal o hexadecimal (precedidos por '`0x`').

Instrucción	`INSTR`
Descripción	Lee una cadena
Formato	`INSTR op1`
Código de Operación	35
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, indirecto, indexado
Comportamiento	Lee una cadena de caracteres introducida por el teclado en la consola y la almacena en posiciones consecutivas de memoria a partir de la dirección indicada por el operando 1. La cadena se finalizará automáticamente con el carácter '`\0`'. No se comprueba previamente si hay espacio para almacenar la cadena, por lo que se puede producir una excepción si se sobrepasa el límite superior de la memoria.

Instrucción	`WRCHAR`
Descripción	Escribe un carácter
Formato	`WRCHAR op1`
Código de Operación	36
Número de Operandos	1
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Escribe en la consola el carácter cuyo valor *ASCII* se corresponde con los 8 bits inferiores del valor del operando 1.

Instrucción	`WRINT`
Descripción	Escribe un entero
Formato	`WRINT op1`
Código de Operación	37
Número de Operandos	1
Modos de Direccionamiento	op1: inmediato, registro, memoria, indirecto, indexado
Comportamiento	Escribe en la consola el valor del operando 1. El número se mostrará en formato decimal con signo si es necesario (-32768 a 32767).

Instrucción	`WRSTR`
Descripción	Escribir una cadena
Formato	`WRSTR op1`
Código de Operación	38
Número de Operandos	1
Modos de Direccionamiento	op1: memoria, indirecto, indexado
Comportamiento	Escribe en la consola una cadena de caracteres, que estará almacenada en memoria a partir de la dirección indicada por el operando 1. Se escribirán caracteres hasta que se llegue al carácter '`\0`'. Por tanto, puede ocurrir que durante este proceso se genere una excepción si no se encuentra el carácter de fin de cadena antes de llegar al límite superior de la memoria.

6 PSEUDOINSTRUCCIONES DEL ENSAMBLADOR

Las pseudoinstrucciones (también conocidas como macroinstrucciones) son órdenes dirigidas al programa ensamblador. Sirven para indicarle cómo se quiere que realice la traducción del programa, pero no forman parte del programa en sí. Sin embargo, algunas pseudoinstrucciones sirven para reservar posiciones de memoria, destinadas a los datos.

Algunas de ellas van seguidas de una *expresión*. En este contexto, se entiende que una expresión consta de enteros (en base decimal o hexadecimal) combinados con los operadores aritméticos

clásicos: suma, resta, producto, división, módulo y paréntesis, y con las reglas de precedencia de operadores habituales. En el caso de la división, el resultado se trunca (3/2=1). Por ejemplo:

```
(3+7*(16-1)/(4-1))%27
```
(el resultado es 11)

6.1 ORG EXPRESIÓN

Ensambla el código que venga a continuación a partir de la posición de memoria resultado de evaluar la expresión. Se pueden usar pseudoinstrucciones `ORG` a lo largo del código fuente todas las veces que se precise. Sin embargo, si no se emplean con orden, se pueden obtener resultados inesperados en el código máquina. Por ejemplo, en este fragmento de código:

```
            ORG 0
bloque_1:   MOVE .R2, .R1
            ...
            ...
            ORG 0
bloque_2:   PUSH #0xFF
```

Se lee la pseudoinstrucción `ORG 0`, por tanto, la etiqueta `bloque_1` tomará el valor 0 y se ensamblarán las siguientes instrucciones a partir de dicha dirección. Luego se vuelve a leer otra pseudoinstrucción `ORG 0`, y nuevamente, la etiqueta `bloque_2` tomará el valor 0 y se ensamblarán las siguientes instrucciones a partir de dicha dirección, sobrescribiendo a las del primer bloque que se ensambló.

Si no se incluye ninguna pseudoinstrucción `ORG` en el código fuente, se comenzará a ensamblar a partir de la posición de memoria 0.

6.2 EQU EXPRESIÓN

Permite establecer una equivalencia entre una etiqueta y un valor, es decir, dar un valor numérico a una etiqueta, y empleándola de la siguiente forma:

`etiqueta: EQU` *`expresión`* ; etiqueta=expresión

Por tanto, cada vez que aparezca la etiqueta como operando en una instrucción, será sustituida por el valor que ha tomado en la pseudoinstrucción `EQU`. Si no se indica la etiqueta, la expresión se calcula, pero su valor se pierde.

6.3 END

Marca el final del código. Después de la pseudoinstrucción `END`, el ensamblador da por finalizado el proceso de traducción del código fuente, ignorando cualquier contenido posterior.

6.4 RES EXPRESIÓN

Reserva tantas posiciones de memoria como indica el resultado de evaluar la expresión. Para marcar el inicio de esta zona de memoria, se emplea una etiqueta de la siguiente forma:

`etiqueta: RES` *`expresión`* ; etiqueta toma el valor de la primera dirección reservada

Por tanto, cada vez que aparezca la etiqueta como operando en una instrucción, será sustituida por el valor que ha tomado en la pseudoinstrucción `RES`. Si no se indica ninguna etiqueta, la zona de memoria se reservará igualmente, pero se pierde la referencia a su dirección de comienzo.

6.5 DATA A, B, C...

Define un conjunto de datos en memoria. Los datos a, b, c... pueden ser enteros (en decimal o hexadecimal) o cadenas de caracteres delimitadas entre comillas dobles. Se puede indicar solamente un dato o varios separándolos por comas. Para marcar el inicio de la zona de datos se usa una etiqueta de la siguiente forma:

`etiqueta: DATA a, b, c... ;` etiqueta toma el valor de la dirección de *a*

Por tanto, cada vez que aparezca la etiqueta como operando en una instrucción, será sustituida por el valor que ha tomado en la pseudoinstrucción `DATA`. Si no se indica ninguna etiqueta, los datos se almacenarán en memoria igualmente, pero se pierde la referencia al primero de ellos.

Las cadenas de caracteres definidas en esta pseudoinstrucción pueden contener el carácter especial '`\n`' (fin de línea), así como cualquiera de los caracteres imprimibles.

Un posible ejemplo de uso de esta pseudoinstrucción sería:

```
ORG 0
DATA "texto\n", 33
END
```

Este programa fuente produce el resultado que se observa en la Figura 11.

0000:	't'
0001:	'e'
0003:	'x'
0004:	't'
0005:	'o'
0006:	'\n'
0007:	'\0'
0008:	33d

Figura 11. Resultado de la pseudoinstrucción DATA

7 DETECCIÓN DE ERRORES EN EL CÓDIGO FUENTE

Tras el proceso de ensamblado, si todo ha ido bien, la herramienta habrá ubicado en memoria el código máquina correspondiente al `programa` fuente introducido. Sin embargo, si había errores en el programa fuente, se comunicarán al usuario. *ENS2025* presenta una lista de errores con este formato:

`Error en línea m: Descripción del error`

Donde *m* es el número de línea donde se encontró el error. Obviamente, si hay errores no se genera código máquina (y no se altera el contenido de la memoria que existiera antes del ensamblado).

Adicionalmente, si se intenta ejecutar código máquina incorrecto (códigos de operación incorrectos o modos de direccionamiento incorrectos), también se producirá un error durante la ejecución, deteniéndose la ejecución inmediatamente.

8 INTERFAZ GRÁFICA

La interfaz gráfica se muestra en la Figura 12. La plataforma es compatible con las versiones actuales de cualquiera de los navegadores principales (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari…). En esta sección, se explica cómo interactuar con la interfaz y la función de cada elemento del sistema.

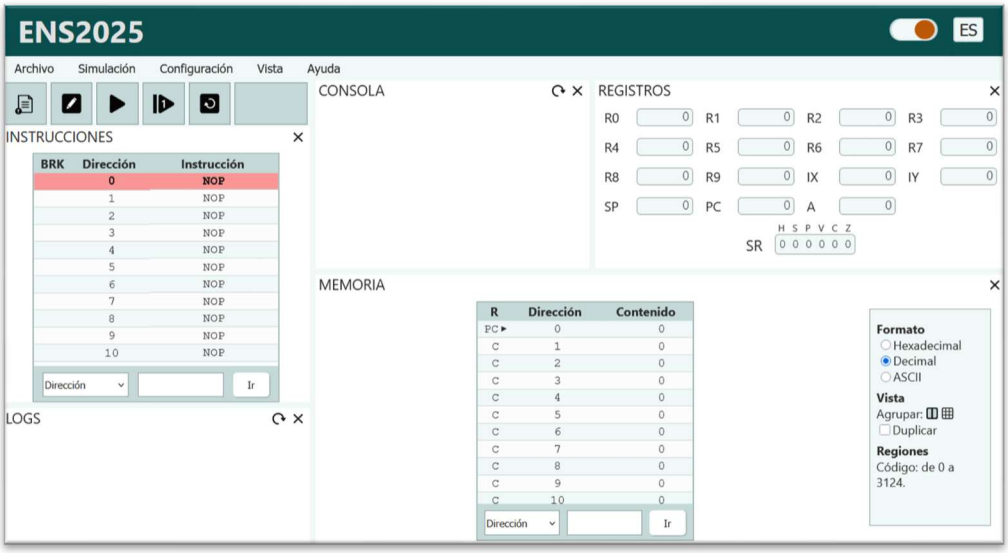


Figura 12. Interfaz gráfica del ENS2025

En la interfaz gráfica se pueden distinguir varias secciones con funcionalidades diferentes:

- **Encabezado**. Muestra el título de la página a la izquierda y los botones de selección de tema e idioma a la derecha.
- **Menú de la aplicación**. A través de este menú se pueden realizar la mayoría de las acciones del sistema.
- **Barra de botones**. Contiene botones más directos para la realización de las acciones más comunes.
- **Ventanas** de la aplicación:
  - *Instrucciones*. Muestra las instrucciones desensambladas del programa actual. Esta ventana se puede alternar con la ventana del editor.
  - *Editor*. Permite observar y editar el código fuente del programa actual.
  - *Logs*. Comunica al usuario confirmación de las acciones realizadas, avisos y errores del programa.
  - *Consola*. Muestra la salida del programa ejecutado y permite introducir una entrada cuando el programa en ejecución lo requiera.
  - *Registros*. Permite observar y editar el valor de los registros y biestables de la máquina simulada.
  - *Memoria*. Permite observar y editar el contenido de la memoria de la máquina simulada.

A continuación, se explica en más detalle los procedimientos disponibles en cada sección.

8.1 Selección de Tema e Idioma

Los botones situados en el encabezado (Figura 13) permiten seleccionar el tema visual y el idioma de la aplicación. Estos también se pueden modificar a través del menú principal.

Figura 13. Botones de selección de tema e idioma

El **botón de tema** (izquierda) permite alternar entre el modo claro y oscuro. Su apariencia se modifica dependiendo del modo elegido. Durante el modo claro, se muestra como se ve en la Figura 14.

Figura 14: Botón de tema con interfaz clara

Si se elige el modo oscuro, el botón adquiere la apariencia mostrada en la Figura 15.

Figura 15: Botón de tema con interfaz oscura

El tema inicial de la interfaz es el claro, pero depende de las preferencias configuradas previamente en la máquina del usuario.

El **botón de idioma** (derecha) permite alternar entre el idioma español o inglés. El cambio de idioma afecta a todos los elementos de la interfaz: el menú, los botones, el título de las secciones e incluso la descripción interna de los elementos, por lo que también afecta a los usuarios que utilicen productos de apoyo. Por defecto, la aplicación está en español. Este botón también cambia de apariencia según la opción seleccionada, mostrando el texto '`ES`' si la interfaz en español y '`EN`' si está en inglés. En este manual se explica la interfaz de la aplicación utilizando el idioma español.

8.2 MENÚ DE LA APLICACIÓN

El menú de la aplicación (Figura 16) divide las funcionalidades en cinco submenús (Archivo, Simulación, Configuración, Vista y Ayuda).

Figura 16. Menú de la aplicación

8.2.1 Menú 'Archivo'

El menú 'Archivo' ofrece las siguientes funciones (Figura 17):

- **Crear archivo nuevo**. Abre un archivo vacío. Se reinicia el valor de los registros y la memoria.
- **Abrir**. Permite subir un archivo del ordenador. El contenido del archivo es ensamblado y preparado para la simulación. Además, el código fuente es cargado al editor.
- **Descargar archivo**. Descarga en un nuevo archivo del ordenador el contenido actual del programa.
- **Abrir ejemplo**. Al seleccionar esta opción, se muestra un desplegable de opciones con los nombres de programas de ejemplo del ensamblador. Tras elegir uno, se abre y se ensambla ese ejemplo.

Figura 17. Menú 'Archivo'

Si se selecciona cualquiera de las opciones que suponen la apertura de un nuevo archivo (Crear archivo nuevo, Abrir, Abrir ejemplo) cuando ya había un archivo abierto y editado, se muestra el aviso de la Figura 18 antes de realizar la acción correspondiente. Este aviso presenta tres opciones:

- 'Descargar y continuar'. Descarga el archivo modificado antes de abrir el siguiente.
- 'Descartar cambios y continuar'. Descarta los cambios del archivo modificado y abre el siguiente.

- ‘Cancelar’. Cierra este menú y no realiza ninguna acción.

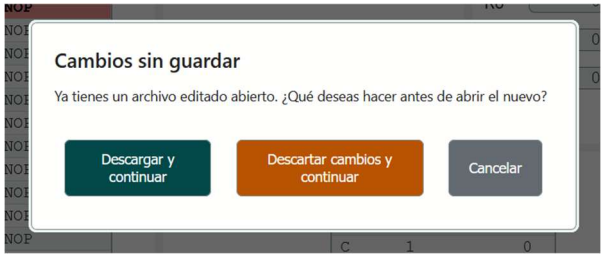


Figura 18. Aviso de conflicto de archivos

8.2.2 Menú ‘Simulación’

El menú ‘Simulación’ (Figura 19) ofrece las siguientes funciones:

- **Ensamblar programa**. Reensambla el programa actual.
- **Ejecutar programa**. Ejecuta el programa hasta el final o hasta el siguiente punto de ruptura. Una vez detenida la ejecución, se actualiza la memoria, los registros y la consola y se muestran en la ventana de *logs* los mensajes relacionados con esa ejecución. Si, durante la ejecución, el programa requiere de entrada del usuario, se interrumpe la ejecución y se muestra la ventana de entrada, hasta que el usuario introduzca una entrada y continúe la ejecución. Por otro lado, si se ha modificado el programa desde el editor, éste se reensambla antes de ejecutar, a menos que la ejecución ya esté en marcha (se haya ejecutado ya al menos una instrucción y aún no se haya alcanzado el fin del programa), en cuyo caso se debe reiniciar la simulación si se desea reensamblar.
- **Ejecutar una instrucción**. Ejecuta la siguiente instrucción. El comportamiento es homólogo a la acción de ‘Ejecutar programa’, pero solamente ejecuta una instrucción cada vez.
- **Reiniciar simulación**. Reinicia la memoria y los registros al estado que tenían tras el ensamblado del programa. Es necesario reiniciar la simulación tras cada ejecución para volver a ejecutar.
- **Abrir o cerrar editor**. Alterna la vista entre las ventanas de ‘Instrucciones’ y ‘Editor’ de ensamblador.

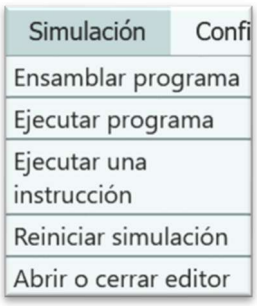


Figura 19. Menú ‘Simulación’

8.2.3 Menú ‘Configuración’

El menú ‘Configuración’ (Figura 20) permite configurar la interfaz a través de las siguientes opciones:

- **Formato de las instrucciones**. Se puede elegir entre hexadecimal y decimal. Esta configuración afecta a las columnas de ‘Dirección’ en las tablas de instrucciones y memoria. El valor por defecto es decimal.
- **Formato del contenido**. Se puede elegir entre hexadecimal, decimal y *ASCII*. Esta configuración afecta al contenido de los registros, la columna de ‘Contenido’ en la tabla de memoria y a los

números y direcciones de las instrucciones desensambladas. Sin embargo, la opción de *ASCII* sólo es visible en la tabla de memoria, mientras que, si está seleccionada, el resto de las regiones afectadas toman el valor del formato de las instrucciones. El valor por defecto es decimal.

- **Dirección de la pila**. Se puede elegir entre ascendente o descendente. Si es ascendente, la pila crece hacia direcciones mayores, mientras que si es descendente crece hacia direcciones menores. El valor por defecto es descendente.
- **Tema**. Se puede elegir entre tema claro y tema oscuro (ver apartado 8.1).
- **Idioma**. Se puede elegir entre español e inglés (ver apartado 8.1).

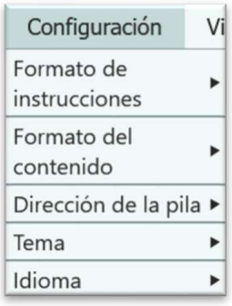


Figura 20. Menú 'Configuración'

Para cada opción, aparece un desplegable con sus valores posibles, y se marca con un punto la opción actualmente seleccionada, al igual que la opción 'Decimal' en la Figura 21.

Figura 21. Valores del submenú 'Formato del contenido' en el menú 'Configuración'

Durante la ejecución de un programa no se puede modificar la dirección de la pila. Esta opción se mostrará desactivada y sin posibilidad de abrir su menú lateral, como en la Figura 22. Para poder modificar esta opción, hay que reiniciar la simulación.

Figura 22. Menú 'Configuración' con el submenú 'Dirección de la pila' desactivado

8.2.4 Menú 'Vista'

El menú 'Vista' (Figura 23) contiene las siguientes opciones:

- **Mostrar ventana**. Permite abrir una ventana cerrada o cerrar una ventana abierta. Las ventanas abiertas se marcan con un *tick* (✓). La opción 'Instrucciones' representa tanto la ventana de instrucciones como la ventana del editor.
- **Duplicar tabla de memoria**. Muestra dos tablas de memoria independientes en la ventana de memoria (**¡Error! No se encuentra el origen de la referencia.**).

- **Expandir columnas de la tabla de memoria**. Alterna entre dos formatos de tabla: el valor de una dirección por cada fila o tantos valores de direcciones por fila como quepan cómodamente en la ventana.

Figura 23. Menú 'Vista' con las opciones de 'Mostrar ventana' desplegadas

8.2.5 Menú 'Ayuda'

El menú 'Ayuda' (Figura 24) solo contiene una opción, **Abrir manual de usuario**, que descarga este documento.

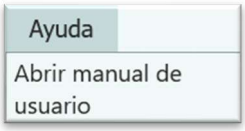


Figura 24. Menú 'Ayuda'

8.2.6 Barra de botones

La barra de botones (Figura 25) contiene botones directos a las acciones más comunes de la interfaz.

Figura 25. Barra de botones

De principio a fin, se muestran los siguientes botones:

- **Abrir**. Equivalente a la opción de 'Abrir' del menú 'Archivo'.
- **Editar**. Alterna entre la ventana del editor y de instrucciones, equivalente a la opción 'Abrir o cerrar editor' del menú 'Simulación'.
- **Ejecutar programa**. Equivalente a la opción de 'Ejecutar programa' del menú 'Simulación'.
- **Ejecutar una instrucción**. Equivalente a la opción de 'Ejecutar una instrucción' del menú 'Simulación'.
- **Reiniciar simulación**. Equivalente a la opción de 'Reiniciar simulación' del menú 'Simulación'.

8.3 VENTANAS DE LA APLICACIÓN

Todas las ventanas se pueden cerrar seleccionando el aspa situada en la esquina superior derecha. Se pueden volver a abrir a través del menú 'Vista'. Se explican brevemente a continuación las distintas ventanas disponibles.

8.3.1 Ventana de instrucciones

La ventana de instrucciones (Figura 26) muestra la tabla de desensamblado de la memoria. Esta tabla tiene tres columnas: la columna de puntos de ruptura (‘BRK’), la de direcciones y la de las instrucciones desensambladas. Al no conocer el comienzo de las instrucciones, es posible que las primeras instrucciones desensambladas muestren datos incoherentes, al igual que el desensamblado de las secciones de datos. Si se encuentra un valor de memoria que no es una instrucción válida, en esa dirección aparece el texto ‘**INSTRUCCIÓN NO IMPLEMENTADA**’.

Figura 26. Ventana de instrucciones

Para navegar por la tabla se puede subir y bajar con la barra de desplazamiento que aparece en el lateral, con las teclas de los cursores o a través de la opción de navegación inferior, que se explica con más detalle en el apartado 8.3.7.

Para colocar o eliminar un punto de ruptura, se debe seleccionar la casilla de la columna ‘BRK’ de la instrucción deseada. Los puntos de ruptura se indican con un punto de color, como el que se muestra en la Figura 27.

Figura 27. Instrucción con punto de ruptura

Además, la fila de la instrucción a la que apunta el PC aparece resaltada con un fondo de distinto color y con el texto en negrita.

8.3.2 Editor

La ventana del editor (Figura 28**¡Error! No se encuentra el origen de la referencia.**) muestra el código fuente del programa ensamblador (que puede haber sido subido a la aplicación o editado directamente). Este código se puede modificar y será reensamblado automáticamente al ejecutar.

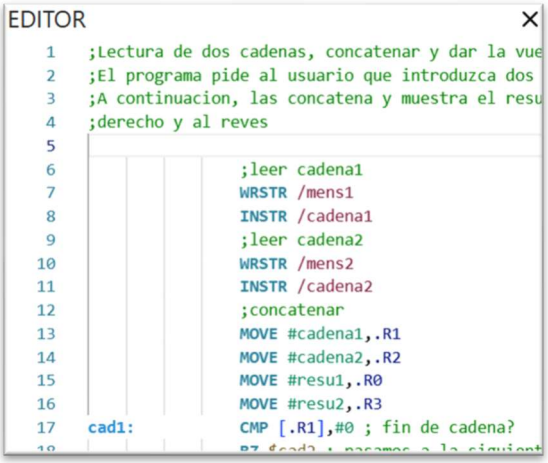


Figura 28. Ventana del editor

El editor ofrece muchas acciones habituales mediante atajos de teclado para agilizar la edición del código. Estos atajos son equivalentes a los atajos por defecto en el editor de código Visual Studio Code. Las acciones también se pueden realizar manualmente a través del menú flotante (Figura 29) que aparece al hacer clic derecho sobre el editor y seleccionar 'Paleta de comandos' o pulsar 'F1' (Figura 30).

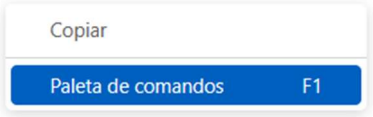


Figura 29. Menú flotante del editor

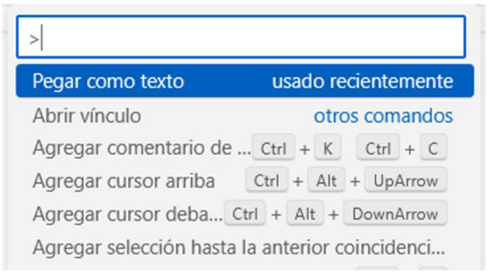


Figura 30. Paleta de comandos

8.3.3 Ventana de *logs*

La ventana de *logs* (Figura 31) muestra tres tipos de mensajes:

- Mensajes de información: comienzan por la palabra '`INFO`' y tienen color negro; confirman la correcta realización de las acciones del usuario.
- Mensajes de aviso: comienzan por la palabra '`AVISO`' y tienen color naranja; notifican situaciones que, sin ser errores críticos, requieren atención o podrían derivar en problemas con el programa o las acciones del usuario.
- Mensajes de error: comienzan por la palabra '`ERROR`' y tienen color rojo; comunican problemas del sistema, acciones que no se han podido realizar, excepciones de la máquina y errores de ensamblado.

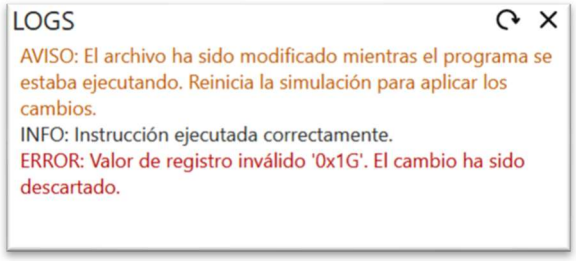


Figura 31. Ventana de logs

Para limpiar la ventana de *logs* se puede activar el botón ↷ de la esquina superior derecha.

8.3.4 Consola

La ventana de la consola (Figura 32) permite comunicarse con el módulo de entrada y salida de la máquina simulada. Tras cada ejecución, el contenido de las instrucciones de escritura aparecerá en la consola. Una línea de guiones (‘`----------`’) separa la salida de cada ejecución del programa. Para limpiar la salida de la consola se puede activar el botón ↷ de la esquina superior derecha.

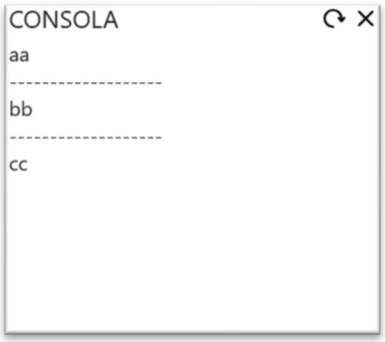


Figura 32. Ventana de la consola

Cuando el programa solicita una entrada, en la ventana de la consola aparece una sección que indica el tipo de entrada a introducir (cadena, carácter o número), como muestra la Figura 33. Se debe escribir la entrada deseada en la caja dedicada para ello y pulsar la tecla ‘Enter’ o seleccionar el botón ‘`Introducir`’. El programa quedará parado hasta que se introduzca la entrada y retomará la ejecución automáticamente después de haber sido introducida.

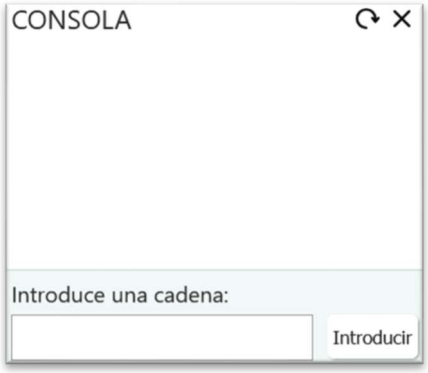


Figura 33. Ventana de la consola con solicitud de entrada

8.3.5 Ventana de registros

La ventana de registros (Figura 34) permite consultar y editar el contenido de los registros y biestables de la máquina simulada.

Para editar un registro, se debe acceder a él, introducir su nuevo valor y salir del registro o pulsar 'Enter' para confirmar el cambio. Independientemente del formato seleccionado en ese momento, los registros aceptan números enteros decimales (sin prefijo) y hexadecimales (con el prefijo '`0x`') dentro del rango aceptable. Los biestables también son editables de la misma manera y aceptan los valores '`0`' o '`1`'.

Figura 34. Ventana de registros

8.3.6 Ventana de memoria

La ventana de memoria (Figura 35) está formada por la tabla de memoria y un menú lateral con opciones relevantes para esta ventana.

Figura 35. Ventana de memoria

La tabla de memoria tiene tres columnas: `R` (región), `Dirección` y `Contenido`. La región indica con una `C` aquellas direcciones que pertenezcan a la región de código y una `D` aquellas direcciones que pertenezcan a la zona de datos establecida en el programa fuente. Además, contiene indicadores que apuntan a la dirección contenida en los registros IX, IY, PC o SP. El contenido de cada dirección es editable activando la celda correspondiente, introduciendo el valor nuevo y saliendo de la celda o pulsando 'Enter' para confirmar el cambio. Si el formato seleccionado es hexadecimal o decimal, las celdas aceptan números enteros decimales (sin prefijo) y hexadecimales (con el prefijo '`0x`') dentro del rango aceptable.

Si el formato seleccionado es *ASCII*, si el contenido de una dirección coincide con el código *ASCII* de un carácter imprimible, se mostrará ese carácter, y en caso contrario se mostrará la cadena '`**`', que indica que es un carácter no representable. Al editar una celda en este formato, se puede introducir únicamente un carácter imprimible, y adquirirá el valor de su código *ASCII* correspondiente.

Para navegar por la tabla se puede subir y bajar con la barra de desplazamiento que aparece en el lateral al entrar en la tabla, con los cursores o a través del menú de navegación inferior, que se explica con más detalle en el apartado 8.3.7.

Respecto al pequeño menú lateral de esta ventana, está dividido en tres secciones:

- **Formato**. Permite cambiar el formato globalmente entre hexadecimal, decimal y ASCII. Cambia tanto el formato de las instrucciones como el del contenido, a menos que se escoja la opción ASCII, en cuyo caso sólo se modifica el formato del contenido.
- **Vista**. Contiene dos opciones de configuración:
  - **Agrupar**. Alterna entre dos formatos de tabla: una dirección por cada fila o tantas direcciones por fila como quepan cómodamente en la ventana (Figura 36). Equivale a la opción 'Expandir columnas de la tabla de memoria' del menú 'Vista'.
  - **Duplicar**. Muestra dos tablas de memoria independientes en lugar de una (Figura 37). Las tablas pueden mostrar regiones diferentes, pero los cambios de la memoria se sincronizarán instantáneamente en ambas tablas. Equivalente a la opción 'Duplicar tabla de memoria' del menú 'Vista'.
- **Regiones**. Muestra la dirección inicial y final de la región de código y de los datos establecidos estáticamente por el fichero fuente.

Figura 36. Ventana de la memoria con la tabla de memoria extendida

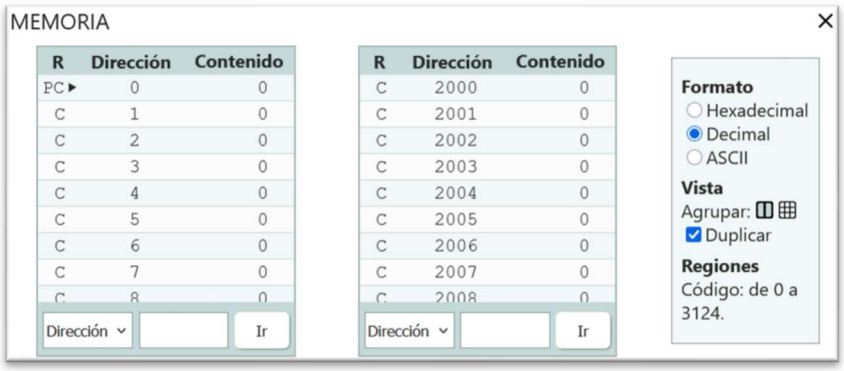


Figura 37. Ventana de memoria con tablas de memoria duplicadas

8.3.7 Opción de navegación de tablas

La opción de navegación de tablas (Figura 38) se encuentra situada en la parte inferior de las tablas de instrucciones y de memoria y permite mostrar directamente una dirección a partir de su número, del valor de un registro índice o del valor de una etiqueta.

Figura 38. Opción de navegación de tablas

Esta opción de navegación consta de dos campos interactivos. El primero (Figura 39) permite seleccionar el Modo de navegación, el cual condiciona la función del segundo campo según la opción elegida:

- Dirección: Permite introducir manualmente una dirección en el segundo campo. Se admiten formatos decimal o hexadecimal (usando el prefijo '`0x`').
- Registro: Navega hacia la dirección almacenada en un registro específico. Al seleccionar este modo, el segundo campo se convierte en un desplegable con cuatro registros (se deberá seleccionar uno de ellos): IX, IY, SP o PC.
- Etiqueta: Permite saltar a una dirección asociada a un nombre simbólico. Se debe seleccionar la etiqueta correspondiente en el desplegable del segundo campo.

Una vez definida la dirección que se desea consultar, se debe pulsar la tecla 'Enter' o el botón 'Ir'.

Figura 39. Primer campo de la opción de navegación de tablas

8.3.8 Navegación por teclado

Aunque la interfaz sigue los patrones de navegación estándar, se han implementado atajos específicos para agilizar el desplazamiento entre los bloques principales a través del teclado.

En las secciones de Editor, Registros, Tabla de Memoria y Tabla de Instrucciones, se ha modificado el comportamiento de la tecla 'Tab' para evitar recorridos innecesarios (como saltar celda por celda en la memoria o insertar tabulaciones en el editor). El funcionamiento es el siguiente:

- **Navegación Rápida** (Saltar secciones): Al pulsar 'Tab', el foco se situará sobre la sección completa. Si se pulsa 'Tab' nuevamente, saltará directamente al siguiente bloque de la interfaz, ignorando el contenido interno de la sección actual. Con 'Shift + Tab' la navegación se produce en el sentido inverso. En función del elemento actual, se podrá interactuar con las teclas 'Enter' (para activar acciones) o las teclas de los cursores (para cambiar de tamaño las ventanas).
- **Navegación Detallada** (Acceder al contenido): Para interactuar con los elementos internos de una sección (un registro específico, una línea de código o una dirección de memoria), hay que situar el foco en la sección deseada y pulsar 'Enter' para entrar. A partir de este momento, se podrá usar 'Tab' y 'Shift + Tab' para navegar por su interior. Para salir del modo detallado y volver a la navegación general, se debe pulsar la tecla 'Esc'. El foco regresará al contenedor principal

9 TABLAS RESUMEN

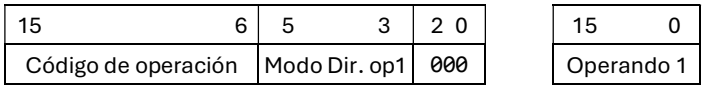
9.1 RESUMEN DE FORMATOS DE INSTRUCCIÓN

Las siguientes tablas explican la codificación interna de cada instrucción, dependiendo del tipo de instrucción y en función de sus posibles operandos.

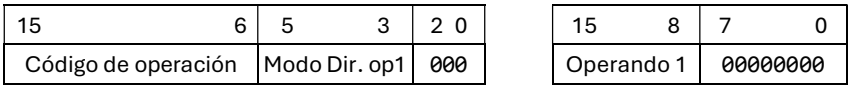
- Instrucción sin operandos:

15 6	5 3	2 0
Código de operación	000	000

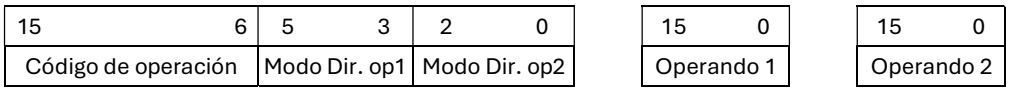
- Instrucción con un operando inmediato, directo a memoria o indexado:



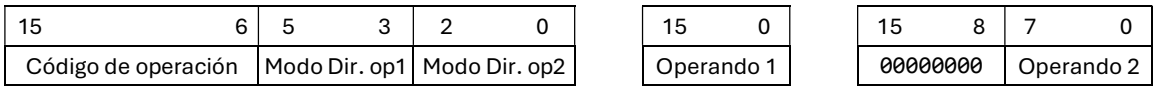
- Instrucción con un operando ni inmediato, directo a memoria o indexado:



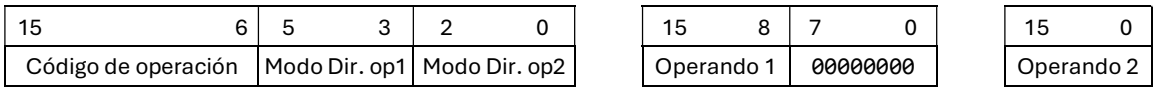
- Instrucción con dos operandos, ambos inmediatos, directos a memoria o indexado:



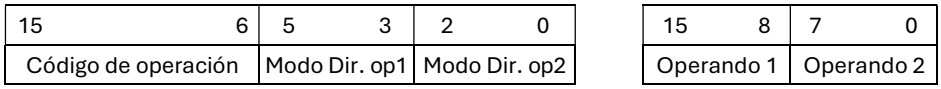
- Instrucción con dos operandos, el primero inmediato, directo a memoria o indexado y el segundo no.



- Instrucción con dos operandos, el segundo inmediato, directo a memoria o indexado y el primero no.



- Instrucción con dos operandos, ninguno de los cuales es inmediato, directo a memoria o indexado.



Nota: los operandos que usen sólo 4 bits rellenarán los bits superiores con ceros.

9.2 MODOS DE DIRECCIONAMIENTO, ANCHO Y CODIFICACIÓN

Esta tabla indica lo que ocupa internamente cada modo de direccionamiento y su codificación.

Modo	Ancho	Codificación
Sin Operando	N/A	000 (0)
Inmediato	16 bits	001 (1)
Registro	4 bits	010 (2)
Memoria	16 bits	011 (3)
Indirecto	4 bits	100 (4)
Indexado a IX	16 bits	101 (5)
Indexado a IY	16 bits	110 (6)
Relativo a PC o indexado a SP	16 bits	111 (7)

9.3 BANCO DE REGISTROS Y CODIFICACIÓN

Esta tabla indica la codificación en una palabra del banco de registros.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC	SP	IY	IX	SR	A	R9	R8	R7	R6	R5	R4	R3	R2	R1	R0

9.4 POSICIÓN DE LOS BIESTABLES DE ESTADO DENTRO DEL REGISTRO DE ESTADO

Esta tabla indica la codificación de los biestables en el registro SR.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	H	S	P	V	C	Z

9.5 INSTRUCCIONES Y DIRECCIONAMIENTOS PERMITIDOS

Esta tabla indica muestra las distintas instrucciones del lenguaje y los direccionamientos permitidos para los operandos de dichas instrucciones.

Mnemónico	Código Oper.	Modo Dir. Op1						Modo Dir. Op2					
		#	.	/	[ ]	#[ ]	$	#	.	/	[ ]	#[ ]	$
NOP	0												
HALT	1												
MOVE	2	✓	✓	✓	✓	✓			✓	✓	✓	✓	
PUSH	3	✓	✓	✓	✓	✓							
POP	4		✓	✓	✓	✓							
ADD	5	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
SUB	6	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
MUL	7	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
DIV	8	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
MOD	9	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
INC	10		✓	✓	✓	✓							
DEC	11		✓	✓	✓	✓							
NEG	12		✓	✓	✓	✓							
CMP	13	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
AND	14	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
OR	15	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
XOR	16	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
NOT	17		✓	✓	✓	✓							
BR	18			✓	✓		✓						
BZ	19			✓	✓		✓						
BNZ	20			✓	✓		✓						
BP	21			✓	✓		✓						
BNP	22			✓	✓		✓						
BN	23			✓	✓		✓						
BNN	24			✓	✓		✓						
BV	25			✓	✓		✓						
BNV	26			✓	✓		✓						
BC	27			✓	✓		✓						
BNC	28			✓	✓		✓						
BE	29			✓	✓		✓						
BO	30			✓	✓		✓						
CALL	31			✓	✓		✓						
RET	32												

Mnemónico	Código Oper.	Modo Dir. Op1						Modo Dir. Op2					
		#	.	/	[ ]	#[ ]	$	#	.	/	[ ]	#[ ]	$
INCHAR	33		✓	✓	✓	✓							
ININT	34		✓	✓	✓	✓							
INSTR	35			✓	✓	✓							
WRCHAR	36	✓	✓	✓	✓	✓							
WRINT	37	✓	✓	✓	✓	✓							
WRSTR	38			✓	✓	✓							