

## Introduction

Environment for the **S**imulation of **C**omputer **A**rchitectures for the **P**urpose of **E**ducation (E.S.C.A.P.E.) es una herramienta didáctica desarrollada por Peter Verplaetse y Jan Van Campenhout en la Universidad de Ghent, Bélgica. Esta herramienta, en contraste a otras herramientas similares, fue diseñada para ser simple e intuitiva para no abrumar al estudiante de las complejidades de modelos certeros de arquitectura de microprocesadores. Aún, E.S.C.A.P.E. tiene una versatilidad enorme en el diseño de un microprocesador simple de arquitectura Von Neumann y de un microprocesador con pipeline simple de arquitectura Harvard.

Para este primer práctico, de la serie de prácticos de E.S.C.A.P.E., van a aprender a usar la herramienta de simulación, a configurar las características internas de un microprocesador simple de arquitectura Von Neumann, a establecer el ISA del mismo, y a simular el procesador con código escrito en assembler basado en el nivel ISA establecido.

## Instalación

La herramienta es portable y no requiere instalación. Igual, es recomendable poner el archivo ejecutable en su propia carpeta porque al guardar las configuraciones del procesador diseñado, E.S.C.A.P.E. crea una serie de archivos que en futuro uno puede usar para restablecer una configuración.

## Ejecución

Al ejecutar E.S.C.A.P.E., aparece la ventana principal de la aplicación, Fig. 1. La ventana tiene tres opciones: Configure (Configurar), Microprogrammed Architecture (Arquitectura Microprogramada) y Pipelined Architecture (Arquitectura con Pipeline). Por ahora, se usarán solo las primeras dos opciones.

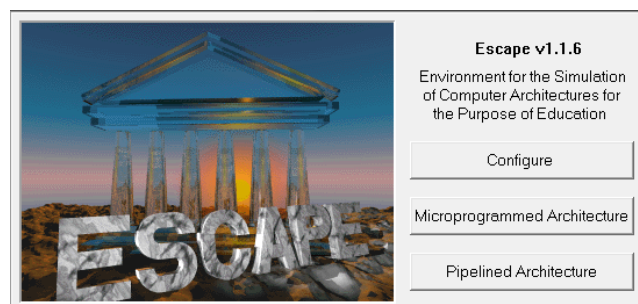


Fig. 1: Ventana principal de E.S.C.A.P.E.



ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II  
TRABAJO PRÁCTICO  
ESCAPE

## Configure (Configurar)

Aquí se encuentran las opciones generales del microprocesador separadas en cuatro pestañas:

- General Options
- Instruction Encoding
- Microprogrammed Architecture
- Pipelined Architecture.

### Pestaña: General Options

En esta pestaña, mostrada en Fig. 2, hay opciones generales sobre la operación del ALU, el comparador y la memoria del microprocesador.

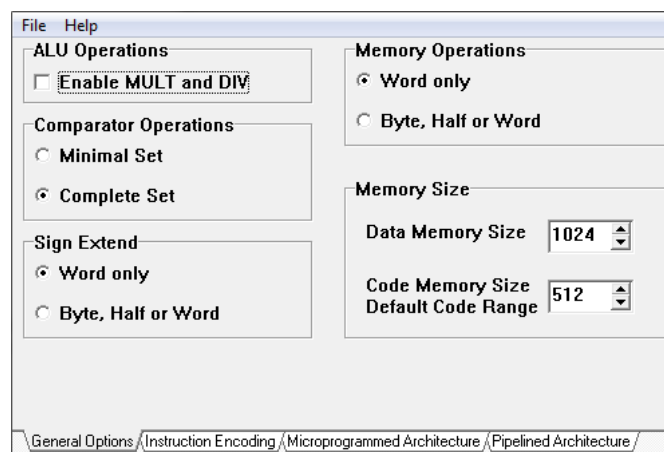


Fig. 2: Pestaña General Options dentro de la ventana de configuración.

**ALU Operations (Operaciones del ALU):** Sirve para habilitar o inhabilitar las operaciones multiplicar y dividir.

**Comparator Operations (Operaciones del Comparador):** el comparador puede ser configurado para poder hacer el mínimo conjunto de operaciones de comparación (igual a y menos que); o para hacer el conjunto completo de operaciones de comparación (igual a, no-igual a, menos que, más que, menos que o igual a, y más que o igual a).

**Sign Extend (Extender el Signo):** la extensión del signo (positivo/negativo) puede ser aplicado al Byte, Half (2 Bytes) o Word (4 Bytes) o solo para un Word. Esto permite expresar en complemento a dos de cualquier dato en las longitudes de 4 bits, 8 bits, y 16bit.

**ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II**  
**TRABAJO PRÁCTICO**

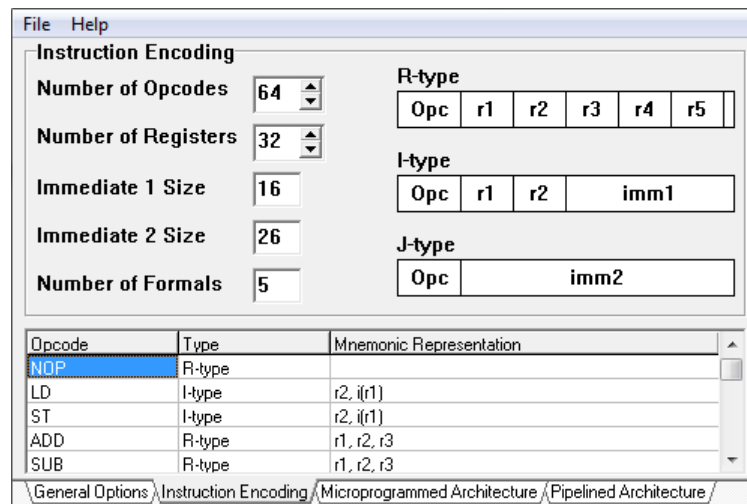
**ESCAPE**

**Memory Operations (Operaciones de Memoria):** las operaciones con la memoria pueden ser hechas con la resolución de Byte, Half o Word, o simplemente solo con resolución de Word.

**Memory Size (Tamaño de la Memoria de Código/Datos):** Tanto el tamaño de la memoria de datos (Data Memory Size) como el tamaño de la memoria de código (Default Code Range) se pueden configurar desde 64 a 32768 bytes.

## Pestaña: Instruction Encoding

En esta pestaña, Fig. 3, se puede configurar el ISA del microprocesador. Aquí se definen una variedad de opciones sobre el formato de instrucciones.



Opcode	Type	Mnemonic Representation
NOP	R-type	
LD	I-type	r2, i(r1)
ST	I-type	r2, i(r1)
ADD	R-type	r1, r2, r3
SUB	R-type	r1, r2, r3

Fig. 3: Pestaña Instruction Encoding dentro de la ventana de configuración.

**Number of Opcodes (Total de Códigos de Operación):** Permite determinar el número máximo de códigos de operación del nivel ISA entre 16, 32, 64, 128 y 256 instrucciones.

**Number of Register (Total de Registros de Propósito General):** Permite determinar el número máximo de registros de propósito general entre 4, 8, 16, 32, 64, 128, y 256.

**Immediate 1 Size (Tamaño del valor Inmediato 1):** Es el tamaño en bits del valor inmediato de las instrucciones del tipo I-type.

**Immediate 2 Size (Tamaño del valor Inmediato 2):** Es el tamaño en bits del valor inmediato de las operaciones J-type.

**Number of Formals (Total Formales Disponibles):** La máxima cantidad de operandos que se pueden usar en las operaciones R-type.



ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II  
TRABAJO PRÁCTICO

ESCAPE

**Opcode (Código de Operación):** El código de operación en forma alfanumérica y sin espacios en blanco.

**Type (Tipo de Código de Operación):** El tipo de instrucción, R-type, I-type, J-type.

**Mnemonic Representation (Representación mnemónica):** la representación textual de los operandos e inmediatos usados en la operación. Se pueden usar r1-r6 (dependiendo de la máxima cantidad configurada) y la letra 'i' y 'j' se pueden usar para representar valores inmediatos. Con respecto a los valores inmediatos, 'i' representa un valor absoluto mientras que 'j' representa un valor relativo al valor del PC (Contador de programa).

## Pestaña: Microprogrammed Architecture

Aquí, Fig. 4, tiene opciones específicas al microprocesador con arquitectura microprogramada. Para el propósito de lo que haremos, sólo se explicarán dos opciones bajo esta pestaña: Microcode Memory y Extra Registers.

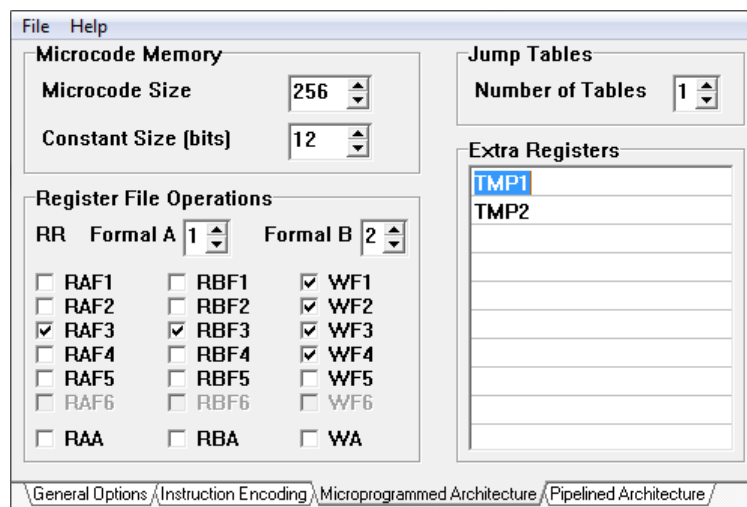


Fig. 4: Pestaña Microprogrammed Architecture dentro de la ventana de configuración.

**Microcode Memory – Microcode Size (Tamaño del Microcode):** Es el tamaño (en número de filas) del Microcode que se usará para definir el funcionamiento de los códigos de operación.

**Microcode Memory – Constant Size (Tamaño del Constante en el Microcode):** Es el tamaño en bits del constante, que es una de las columnas de la tabla del Microcode.

**Extra Registers (Extra Registros):** En esta sección se pueden nombrar extra registros nombrados, en caso de necesitar extender los registros específicos.

**ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II**  
**TRABAJO PRÁCTICO**  
 ESCAPE

## Microprogrammed Architecture (Arquitectura Microprogramada)

Esta ventana muestra los componentes del microprocesador de arquitectura Von Neumann y una serie de botones que se usan para la simulación del mismo, Fig. 5.

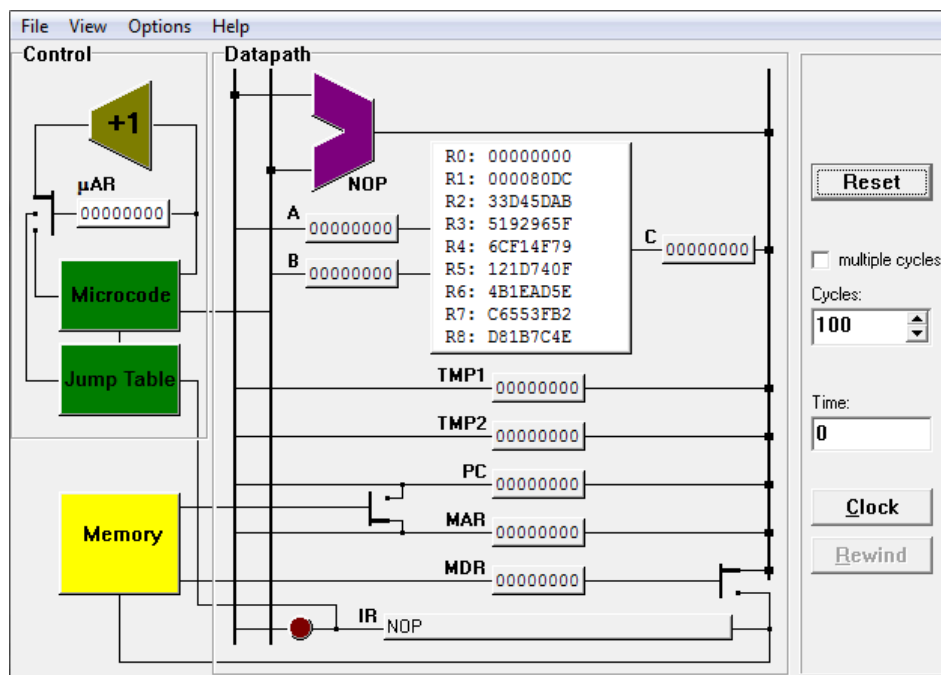


Fig. 5: Ventana principal del procesador con Microprogrammed Architecture.

La microarquitectura está dividida en tres partes: Control, Datapath y Memory. En la parte marcada Control, se puede ver tres componentes que manejan el comportamiento de la unidad de control cuando llega una instrucción al IR (Registro de instrucciones). El componente Jump Table toma parte en la decodificación de la instrucción e indica dónde en el Microcode tiene que buscar la instrucción. El Microcode contiene las microinstrucciones que gobiernan el funcionamiento del Datapath. Finalmente, el uAR contiene la dirección (fila) actual del Microcode.

En el Datapath, se puede ver el ALU, registros internos A, B, C, PC, MAR, MDR e IR, el archivo de registros de propósito general según fueron definidos en la ventana de la configuración, y cualquier otro registro extra definido en la configuración. Esta parte de la microarquitectura responde según la microinstrucción ejecutada.

En la parte marcada Memory se ve el bloque de memoria que se comporta como una caja negra que toma 'x' ciclos en completar un acceso según definido en la configuración.



**ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II**  
**TRABAJO PRÁCTICO**

**ESCAPE**

En la derecha, hay unos botones y cajas que el usuario usa para controlar la simulación. El botón Reset vuelve la simulación a su estado principal; El botón Clock adelanta la simulación un ciclo o si la caja multiple cycles está habilitada, adelanta la simulación la cantidad de ciclos que definidos hay. El botón Rewind hace retroceder la simulación de la misma manera que el botón Clock adelanta la simulación. Por último, la Time muestra el ciclo actual siendo ejecutado.

## Microcode

Como se mencionó anteriormente, el microprocesador se comporta según como están definidas las micro instrucciones en el Microcode. Para acceder al Microcode busque la opción Microcode... dentro del menú View en la parte superior de la ventana. Alternativamente puede presionar F7. La pestaña del Microcode se puede ver en Fig. 6.

File Edit View Assemble Help													
uAR	Label	ALU	S1	S2	Dest	ExtIR	Const	JCond	Adr	Mem	MAdr	MDest	Reqs
0000	Fetch							Mbusy	Fetch	RW	PC	IR	
0001		ADD	PC	Const	PC		4	Jump1					RR
0002	ADD	ADD	A	B	C								
0003	WF3							True	Fetch				WF3
0004													
0005													
0006													
0007													
0008													
0009													
000A													
000B													
000C													
000D													
000E													
000F													

Dropdown Mode Overwrite

Microcode / Jump Tables

Fig. 6: Pestaña del Microcode.

El Microcode está organizado en forma de una tabla donde cada fila representa una microinstrucción (de no ser confundida con la totalidad de la instrucción que representa el código operacional). Aquí también se puede ver una serie de columnas que definen el comportamiento de la misma. Cada microinstrucción se ejecuta en un ciclo.

**uAR:** Marca la fila o dirección de la microinstrucción dentro del Microcode.

**Label (Etiqueta):** Label ayuda al Jump Table a encontrar la fila correcta que tienen que ejecutar, según su configuración. Es una simplificación del programa, en la realidad la Jump Table decodifica la instrucción del IR para saber directamente cuál fila del microcode tiene que ejecutar.



**ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II**  
**TRABAJO PRÁCTICO**

**ESCAPE**

**ALU::** Las funciones del ALU. Los posibles valores son: Empty – ninguna operación, ADD, SUB, RSUB, AND, OR, XOR, SLL, SRL, SRA, S1, S2, S2S1. Si MUL y DIV están habilitados en la configuración, también aparecerán aquí.

**S1:** La fuente del bus S1. Los posibles valores son: Empty – ninguna fuente, A, Const, PC, MAR, MDR, IR y cualquier registro extra definido en la configuración. Todas las fuentes son registros con la excepción de Const e IR. Const es un constante definido en la columna Const del Microcode e IR es el valor inmediato de la actual instrucción en cuestión (si la instrucción es de tipo I-type o J-type).

**S2:** La fuente del bus S2. Similar a S1 con la diferencia de los posible valores que son: Empty – ninguna fuente, B, Const, PC, MAR, MDR, IR y cualquier registro extra definido en la configuración.

**Dest:** La destinación del ALU. Los posibles valores son: Empty – ninguna destinación, C, PC, MAR, MDR y cualquier registro extra definido en la configuración.

**ExtIR:** Es el tamaño al que el valor inmediato del IR debe ser extendido. Según la configuración, los posibles valores son: Empty – nada, Byte, Half o Word, o simplemente Empty – nada o Word. Sí Empty es seleccionado, el valor no se extiende y el signo es ignorado.

**Const:** Un valor constante de 'x' bits según definido en la configuración. El valor puede ser en base decimal o hexadecimal. Si se usa base hexadecimal, el número debe ser prefijo por 0x.

**JCond:** Aquí se le indica al Microcode que microinstrucción debe ejecutar próximamente. Los posibles valores son:

Valor	Resultado
Empty – nada	$uAR = uAR + 1$
True	$uAR = \text{Adr}$
EQ	$uAR = \text{Adr}$ , cuando el resultado del ALU es = 0
NE	$uAR = \text{Adr}$ , cuando el resultado del ALU es $\neq 0$
LT	$uAR = \text{Adr}$ , cuando el resultado del ALU es $< 0$
GT	$uAR = \text{Adr}$ , cuando el resultado del ALU es $> 0$
LE	$uAR = \text{Adr}$ , cuando el resultado del ALU es $\leq 0$
GE	$uAR = \text{Adr}$ , cuando el resultado del ALU es $\geq 0$
MBusy	$uAR = \text{Adr}$ , cuando la memoria está en uso
Jump1	$uAR =$ la dirección marcada por la Jump Table

**Adr:** la etiqueta donde saltara el Microcode cuando la condición del JCond es verdad.

**Mem:** La funcionalidad del bloque de memoria. Según la configuración, los posibles valores son: Empty – ninguna operación, RB, RH, RW, WB, WH, o WW, o Empty – ninguna





ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II  
TRABAJO PRÁCTICO

ESCAPE

operación, RW o WW. La primera letra indica si es una operación de lectura (R) o escritura (W), y la segunda letra indica la resolución de acceso, Byte (B), Half (H), Word (W).

**MAdr:** La fuente de la dirección en las operaciones de lectura o escritura. Los posibles valores son: Empty – por defecto usa MAR, MAR o PC.

**MDest:** La destinación de las operaciones de lectura o escritura. Los posibles valores son: Empty – por defecto usa IR, MDR o IR.

**Regs:** Define la funcionalidad del archivo de registro. Los posibles valores son:

Valor	Función
RR	Leer formales a registros A y B
RAFn	Leer formal n a registro A
RBFn	Leer formal n a registro B
WFn	Escribir registro C a formal n
RAAn	Leer actual n a registro A
RBAAn	Leer actual n a registro B
WAn	Escribir registro C a actual n

## Jump Table

La pestaña de Jump Table muestra una tabla donde la primera columna es el código operacional definido en la configuración y la segunda columna es la etiqueta donde se debe saltar a, en el Microcode, para llevar a cabo esa instrucción. Esta pestaña se puede ver en Fig. 7.

Opcode	Jump Table 1
NOP	Fetch
LD	LD
ST	ST
ADD	ADD
SUB	SUB
MUL	MUL
DIV	DIV
AND	AND
OR	OR
XOR	XOR
SLL	SLL
SRL	SRL
SRA	SRA
ADDI	ADDI
SUBI	SUBI
MULI	MULI
DIVI	DIVI

Microcode / Jump Tables

Fig. 7: Pestaña de la Jump Table.



**ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II**  
**TRABAJO PRÁCTICO**  
 ESCAPE

## Código Assembler

Para simular el diseño implementado del microprocesador, se puede escribir un programa en código assembler. Para acceder al editor de código, busque la opción Instruction Memory... dentro del menú View en la parte superior de la ventana. Alternativamente puede presionar F5. Esta ventana se puede ver en Fig. 8.

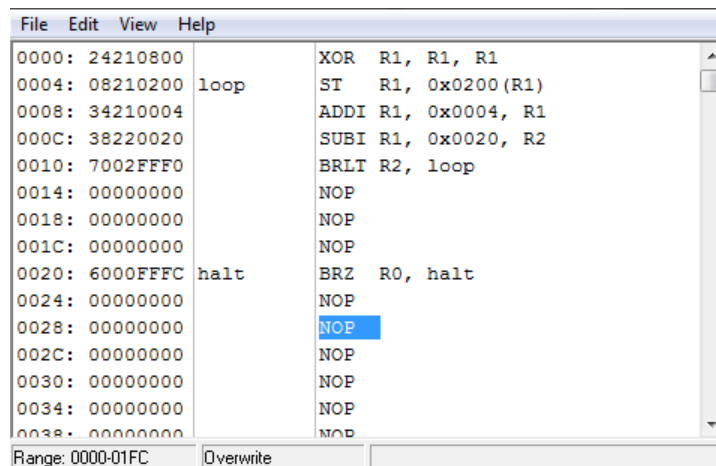


Fig. 8: Ventana del código assembly.

En la ventana de código verá tres columnas. La primera muestra dirección en memoria donde está guardada la instrucción y el código máquina de la instrucción. La segunda columna contiene una etiqueta opcional (útil para el uso de ramificaciones/saltos), y finalmente la tercera columna contiene el código. Solo se puede editar la segunda y tercera columna.

## Bloque de Memoria

Para ver que está guardado en el bloque de memoria, tanto código como data, busque la opción Data Memory... dentro del menú View en la parte superior de la ventana. Alternativamente puede presionar F6. Esta ventana se puede ver en Fig. 9.

**ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II**  
**TRABAJO PRÁCTICO**  
 ESCAPE

File	Edit	View	Help
000:	24210800	08210200	34210004 38220020 7002FFF0 00000000
018:	00000000	00000000	6000FFFC 00000000 00000000 00000000
030:	00000000	00000000	00000000 00000000 00000000 00000000
048:	00000000	00000000	00000000 00000000 00000000 00000000
060:	00000000	00000000	00000000 00000000 00000000 00000000
078:	00000000	00000000	00000000 00000000 00000000 00000000
090:	00000000	00000000	00000000 00000000 00000000 00000000
0A8:	00000000	00000000	00000000 00000000 00000000 00000000
0C0:	00000000	00000000	00000000 00000000 00000000 00000000
0D8:	00000000	00000000	00000000 00000000 00000000 00000000
0F0:	00000000	00000000	00000000 00000000 00000000 00000000
108:	00000000	00000000	00000000 00000000 00000000 00000000
120:	00000000	00000000	00000000 00000000 00000000 00000000
138:	00000000	00000000	00000000 00000000 00000000 00000000

Fig. 9: El bloque de memoria.

## Puntos de Interrupción

El usuario puede crear puntos de interrupción en el código según los estados de los diferentes registros del microprocesador. Para acceder a los puntos de interrupción, busque Breakpoints... dentro del menú View en la parte superior de la ventana. Alternativamente puede presionar F8. Esta ventana se puede ver en Fig. 10.

View		Help	
Organisational Registers			
<input type="checkbox"/>	uAR	=	0x00000000
<input type="checkbox"/>	A	=	0x00000000
<input type="checkbox"/>	B	=	0x00000000
<input type="checkbox"/>	C	=	0x00000000
<input type="checkbox"/>	PC	=	0x00000000
Register File Registers			
<input type="checkbox"/>	R 1	=	0x00000000
<input type="checkbox"/>	R 2	=	0x00000000
<input type="checkbox"/>	R 3	=	0x00000000
<input type="checkbox"/>	R 4	=	0x00000000
<input type="checkbox"/>	R 5	=	0x00000000

Fig. 10: Ventana de puntos de interrupción.

Los puntos de interrupción son útiles para poder pasar rápido la simulación y detenerla en algún punto importante del código.



**ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II**  
**TRABAJO PRÁCTICO**

ESCAPE

## Ejercicio

Para este práctico, van a empezar a diseñar una ISA y después la van a simular. Las características son las siguientes:

- 32 códigos de operación.
- 8 registros generales.
- 1 registro específico llamado SWAPTMP.
- Los Códigos de operación tienen un máximo de 3 operandos.
- Memoria de 1 KB dividida en 512B para código y 512B para data.
- Un Microcode de tamaño 64 con el constante de 16 bits.
- Las 16 operaciones posibles incluyen:
  - **Operaciones de sistema:** no-operación.
  - **Operaciones aritméticas:** sumar, restar, restar en reversa.
  - **Operaciones lógicas:** or, and, xor.
  - **Operaciones de memoria:** leer y escribir.
  - **Operaciones de saltos:** condicional y absoluta.
  - **Operaciones extras:** intercambio de datos entre dos registros (swap).

El código que tienen que simular es:

1. Usar operación lógica XOR para resetear registro R1 y R2
2. Sumar el valor 5 a R1.
3. Sumar el valor 10 a R2.
4. Sumar R1 y R2 y guardar el resultado en R2.
5. Intercambiar el valor entre R1 y R2.
6. Restar el valor en R2 al valor en R1 usando la operación de resta en reversa.
7. Restar el valor 3 a R1.
8. Usar operación lógica AND entre R1 y el número binario 00000100.
9. Usar operación lógica OR entre R1 y el número binario 00000010.
10. Usar salto condicional, continuamente reste 1 al R1 hasta que tenga 0.
11. Guardar el valor de R2 a la dirección de memoria 0x204.
12. Leer el valor en memoria en la dirección 0x204 al registro R3.
13. Usando saltos absolutos, impedir que el código siga progresando.

**Bonus:** Implementar algún algoritmo simple, como alguno para ordenar números de menor a mayor. (1 punto a su nota final)



**ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORAS II**  
**TRABAJO PRÁCTICO**  
 ESCAPE

Código	Descripción
NOP	Ninguna operación
ADD	Sumar dos operandos y guardar el resultado en un tercer operando.
SUB	Restar dos operandos y guardar el resultado en un tercer operando.
SUBR	Restar dos operandos (en forma inversa) y guardar el resultado en un tercer operando.
ADDI	Sumar un valor constante a un operando y guardar el resultado en un segundo operando.
SUBI	Restar un valor constante de un operando y guardar el resultado en un segundo operando.
OR	Hacer la operación lógica OR entre dos operandos y guardar el resultado en un tercer operando
AND	Hacer la operación lógica AND entre dos operandos y guardar el resultado en un tercer operando
XOR	Hacer la operación lógica XOR entre dos operandos y guardar el resultado en un tercer operando
ORI	Hacer la operación lógica OR entre un operando y un valor constante, y guardar el resultado en un segundo operando
ANDI	Hacer la operación lógica AND entre un operando y un valor constante, y guardar el resultado en un segundo operando
LD	Leer un valor de memoria a un registro
ST	Escribir un valor de registro a memoria
GOTO	Saltar a una dirección en memoria o una etiqueta
BRNZ	Salta a una dirección en memoria o una etiqueta si el resultado no es cero
SWAP	Intercambiar valores entre dos operandos.