

Juego de instrucciones del Simulador de Von Neumann

Nomenclatura utilizada:

Rd: Tres bits que codifican el registro destino en una operación.

Rs: Tres bits que codifican el registro fuente de una operación.

Rs1: Tres bits que codifican el registro fuente de una operación.

Rs2: Tres bits que codifican el registro fuente de una operación.

Ri: Tres bits que codifican el registro índice para direccionamiento indirecto.

Rd/s: Tres bits que codifican un registro que es a la vez fuente y destino de una operación.

Rx: Tres bits que codifican el registro que contiene la dirección de destino para saltos indirectos.

Inm_8: Valor numérico de 8 bits.

Dec: Valor decimal positivo o negativo

Hexa: Valor hexadecimal de 2 dígitos

Mnemónico	Código	Operación	Descripción
<i>Instrucciones de movimiento</i>			
NOP	00000 000000000000	-	Instrucción nula
MOV Rd, Rs	00001 Rd Rs 00000	$Rd = Rs$	Transferencia entre registros.
MOV Rd, [Ri]	00010 Rd Ri 00000	$Rd = [Ri]$	Copiar el contenido de la posición de memoria cuya dirección está en Ri en Rd.
MOV [Ri], Rs	00011 Ri Rs 00000	$[Ri] = Rs$	Copiar el contenido del registro Rs en la posición de memoria cuya dirección está en Ri.
MOVL Rd, Hexa	00100 Rd Inm_8	$Rdbajo = Inm_8$	Copiar en los 8 bits menos significativo de Rd el dato codificado en los 8 bits del campo Inm_8.
MOVH Rd, Hexa	00101 Rd Inm_8	$Rdalto = Inm_8$	Copiar en los 8 bits más significativo de Rd el dato codificado en los 8 bits del campo Inm_8.
<i>Instrucciones Aritmético-Lógicas de tres operandos</i>			
ADD Rd, Rs1, Rs2	01000 Rd Rs1 Rs2 00	$Rd = Rs1 + Rs2$	Suma el contenido de los registros Rs1 y Rs2 y almacena el resultado en Rd.
SUB Rd, Rs1, Rs2	01001 Rd Rs1 Rs2 00	$Rd = Rs1 - Rs2$	Resta el contenido del registro Rs2 al registro Rs1 y almacena el resultado en Rd.
OR Rd, Rs1, Rs2	01010 Rd Rs1 Rs2 00	$Rd = Rs1 \text{ OR } Rs2$	Realiza la operación OR con el contenido de los registros Rs1 y Rs2 y almacena el resultado en Rd.
AND Rd, Rs1, Rs2	01011 Rd Rs1 Rs2 00	$Rd = Rs1 \text{ AND } Rs2$	Realiza la operación AND con el contenido de los registros Rs1 y Rs2 y almacena el resultado en Rd.
XOR Rd, Rs1, Rs2	01100 Rd Rs1 Rs2 00	$Rd = Rs1 \text{ XOR } Rs2$	Realiza la operación XOR con el contenido de los registros Rs1 y Rs2 y almacena el resultado en Rd.

Mnemónico	Código	Operación	Descripción
Instrucciones Aritmético-Lógicas de 2 operandos			
COMP Rs1, Rs2	01101Rs1 Rs2 000 00	Rs1 - Rs2	Resta el contenido del registro Rs2 al registro Rs1.
Instrucciones Aritmético-Lógicas de 1 operando			
NOT Rd/s	10000 Rd/s 00000000	$Rd/s = \sim Rd/s$	Realiza la operación lógica NOT con los bits del registro Rd/s.
INC Rd/s	10001 Rd/s 00000000	$Rd/s = Rd/s + 1$	Incrementa el contenido del registro Rd/s en una unidad.
DEC Rd/s	10010 Rd/s 00000000	$Rd/s = Rd/s - 1$	Decrementa el contenido del registro Rd/s en una unidad.
NEG Rd/s	10011 Rd/s 00000000	$Rd/s = \sim Rd/s + 1$	Cambia de signo (complemento a 2) el contenido del registro Rd/s.
Instrucciones de control de flujo – Salto incondicional			
JMP Dec	11000 000 Inm_8	$PC = PC + Ex_16(Inm_8)$	Realiza un salto relativo.
JMP Rx	11001 Rx 00000000	$PC = Rx$	Realiza un salto indirecto absoluto a la posición de memoria contenida en el registro Rx.
Instrucciones de control de flujo – Salto condicional			
BRC Dec	11110 000 Inm_8	Si la condición es cierta: $PC = PC + Ex_16(Inm_8)$	Salta si CF = 1
BRNC Dec	11110 001 Inm_8		Salta si CF = 0
BRO Dec	11110 010 Inm_8		Salta si OF = 1
BRNO Dec	11110 011 Inm_8		Salta si OF = 0
BRZ Dec	11110 100 Inm_8		Salta si ZF = 1
BRNZ Dec	11110 101 Inm_8		Salta si ZF = 0
BRS Dec	11110 110 Inm_8		Salta si SF = 1
BRNS Dec	11110 111 Inm_8		Salta si SF = 0

Usando ensambla.exe

Para poder cargar las instrucciones en el simulador utilizaremos el programa ensambla.exe. El mismo es una aplicación de consola, lo que significa que tendremos que utilizarla de una forma particular.

Primero debemos escribir el código en cualquier editor de texto plano (bloc de notas por ejemplo) y guardarlo como “.ens” en la misma carpeta donde tenemos el ensambla.exe. La estructura de este archivo es la siguiente:

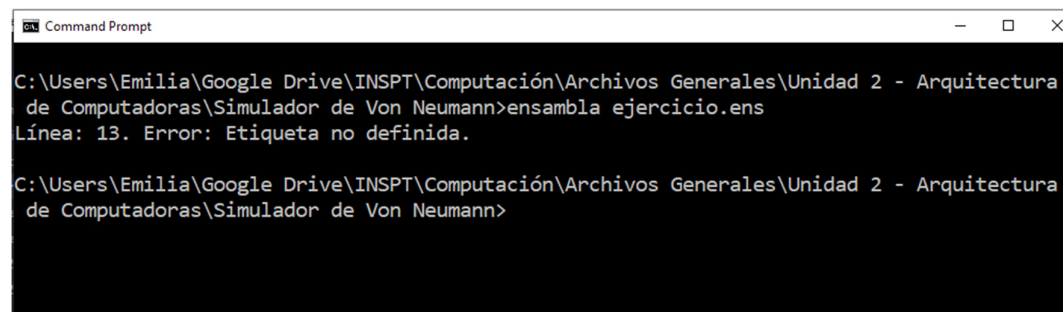
```
ORIGEN <dirección de memoria codificada en hexadecimal>

.CODIGO
    <instrucción 1>
    <instrucción 2>
    <instrucción 3>
    <...>
    <instrucción N>
FIN
```

Todo lo que está en negrita debe ser reemplazado por lo que corresponda. La instrucción ORIGEN le indicará al ensambla.exe a partir de qué dirección de memoria queremos que se disponga la instrucción 1 de nuestro código en la memoria del simulador.

Una vez grabado el archivo “.ens” abrir la carpeta donde se lo guardó junto con el ensambla.exe y arrastrar el “.ens” encima de ensambla.exe. Si no hubieron errores al escribir el código en el archivo “.ens” se va a generar un nuevo archivo, con el mismo nombre que el “.ens” pero con la extensión “.eje”. Este archivo lo podemos cargar en el simulador yendo a Archivo -> Abrir.

Si al arrastrar el archivo no vemos que se genere el “.eje” seguramente estamos incurriendo en algún error en nuestro código. Para poder ver cual es, deberemos abrir la consola de Windows (cmd.exe: “Command Prompt” o “Símbolo de sistema” dependiendo del idioma del Windows). Una vez abierta navegamos hasta la carpeta donde tenemos nuestro “.ens” y ensambla.exe. Una vez allí ejecutamos “ensambla <nombre>.ens” y debería darnos un mensaje de error. Ejemplo:



```
Command Prompt

C:\Users\Emilia\Google Drive\INSPT\Computación\Archivos Generales\Unidad 2 - Arquitectura de Computadoras\Simulador de Von Neumann>ensambla ejercicio.ens
Línea: 13. Error: Etiqueta no definida.

C:\Users\Emilia\Google Drive\INSPT\Computación\Archivos Generales\Unidad 2 - Arquitectura de Computadoras\Simulador de Von Neumann>
```