



# Guía de trabajo – máquina de Brookshear

## Introducción

La maquina de Brookshear es un modelo de máquina hipotético. Dicha máquina trabaja con un set de instrucciones RISC (Reduced Instruction Set Computer) en la que dispone de un reducido rango de valores (en enteros de -128 a 127 y de punto flotante de -15,5 a 15,6) pero si de registros muchos registros.

## Set de instrucciones

A continuación, el set de instrucciones de la máquina de Brookshear

1RXY	MOV R, [XY]	carga el registro R con el contenido de la dirección XY
2RXY	MOV R, XY	carga el registro R con el número XY
3RXY	MOV [XY], R	copia el registro R en la posición de memoria XY
4ORS	MOV S, R	copia el registro R en el registro S
5RST	ADDI R, S, T	suma los registros S y T (como enteros en complemento a 2) y deja el resultado en R
6RST	ADDF R, S, T	suma los registros S y T (como números de 8 bits con punto flotante) y deja el resultado en R
7RST	OR R, S, T	realiza un OR entre S y T, y deja el resultado en R
8RST	AND R, S, T	realiza un AND entre S y T, y deja el resultado en R
9RST	XOR R, S, T	realiza un XOR entre S y T, y deja el resultado en R
AR0X	ROR R, X	rota X veces el registro R hacia la derecha
BRXY	CMP R, R0;	salta a la dirección XY si el registro R es igual al registro R0
BOXY	JMP XY	salta incondicionalmente a XY
C000	HALT	detiene la ejecución del programa

Un ejemplo: codificar un programa que cargue 07 en R1 y 03 en R2 y guarde la suma en FF

2107

2203

5312

33FF

C000

Siempre la última instrucción es C000.

Como se ve en la tabla, dicho set de instrucciones **no** implementa la resta, multiplicación o división; solo implementa la suma. De modo que, por ejemplo, para hacer 3-8 debe hacerse 3+(-8). Para estos casos veremos que es necesario aplicar máscaras con operaciones lógicas.



## Mascaras para operar números

### Convertir a negativo

Para hacer un número negativo, debe hacerse el complemento a 1, es decir un XOR con la máscara FF (en binario 1111 1111) y luego sumarle 1. Dicho esto, para tener un -6 quedaría:

```
20FF  R0 = mascara
2106  R1 = 6
2201  R2 = 1
9310  XOR entre R1 y R0 y guarda resultado en R3
5332  suma R3 y R2 y se guarda resultado en R3 (R3 = -6, que es FA)
C000  fin de programa
```

### Como saber si un número es impar

En este caso se utiliza la máscara 01 (en binario 0000 0001) y se hace un AND con el valor deseado. El ejemplo que sigue deja en R0 un 0 si es par, un 1 si es impar:

```
2101  Carga la máscara 01
2205  carga el valor a operar
8012  Ejecuta el AND entre R1 y R2 y guarda el resultado el R0
C000  Fin de programa
```

### Como saber si un número es positivo, negativo o cero

Dado que sabemos que el signo lo define el primer bit, la mascara a utilizar es 80 (en binario 1000 0000) y se aplica un AND al valor deseado seguido de un ROT. Con lo cual si quiero saber si es positivo el valor -6, quedaría:

```
2180  Carga la máscara 80
22FA  carga el valor -6 (FA en binario) a operar
8012  Ejecuta el AND entre R1 y R2 y guarda el resultado el R0
A007  Rota los bits de R0 en 7 veces
C000  Fin de programa
```

## Un ejemplo

Codificar un programa que cargue de F0 a F8 los valores del 0 al 8.

Dir	Operación	Descripción
00	2009	Carga 09 en R0
02	2101	Carga 01 en R1 (incremental del ciclo)
04	2200	Carga 00 en R2 (contador del ciclo)
06	32F0	Guarda el contenido de R2 en F0
08	1307	\
0A	5331	> Incrementa la posición de memoria en 1
0C	3307	/
0E	5221	Incrementa R2 en 1
10	B214	Si R2 es igual a R0, salta a la dirección 14
12	B006	Salta a la dirección 06
14	C000	Fin del programa



## Uso de la máquina de Brookshear en la herramienta HTML

El sitio web tiene la siguiente forma:

MEMORIA PRINCIPAL																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	20	09	21	01	22	00	32	F9	13	07	53	31	33	07	52	21
1	B2	14	B0	06	C0	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	01	02	03	04	05	06	07	08	00	00	00	00	00	00	00
REGISTROS																
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
09	01	09	F9	00	00	00	00	00	00	00	00	00	00	00	00	00
CONT. DE PROG.							0	REG. DE INSTR.							C000	
MONITOR																
HEX:	FA	DECODE					C-2:		-6	FP:		-13				
Paró la ejecución.																
LOAD				CLEAR				RUN				STEP			LIST	

En donde:

- Memoria Principal: es el espacio de memoria a utilizar, en el que se cargan las instrucciones del programa y los datos.
- CONT. DE PROG.: el contador de programa. Muestra la dirección de la próxima instrucción a ejecutar.
- REG. DE INSTR.: es el registro de instrucción, muestra la instrucción que se está ejecutando.
- Registros: son los 16 registros que posee la máquina.
- Monitor: es una sección de ayuda en la que se carga el valor en Hexa en el campo HEX y al clicar en DECODE se muestra su valor en DEC en los campos C-2 (complemento a 2) y FP (punto flotante).
- LOAD: carga el programa en formato de texto. Se recomienda usar un Notepad donde se codifica para luego copiarlo y pegarlo en el cuadro que abre LOAD.
- CLEAR: limpia la máquina, diferenciando entre la memoria y los registros.
- RUN: corre todo el programa.
- STEP: corre el programa "paso a paso"
- LIST: al cargar el programa aquí, muestra sus equivalentes en mnemónicos de una máquina CISC



Para el uso de esta herramienta, lo ideal es utilizar un editor de texto a parte (por ejemplo, Notepad++) para codificar el programa y luego cargarlo en la herramienta web con la función LOAD para probarlo.

## Ejercicios:

1. Llenar las últimas 10 direcciones de la memoria con los primeros 10 múltiplos naturales del contenido de la posición 02, si éste es natural
2. Llenar las últimas 10 direcciones de la memoria con los primeros 10 números cuadrados
3. Llenar las últimas 10 direcciones de la memoria con los primeros 10 términos de la sucesión de Fibonacci
4. Calcular el mínimo común múltiplo de los valores enteros contenidos en las posiciones de memoria 02 y 03, dejando el resultado en R5
5. Calcular el máximo común divisor de los valores enteros contenidos en las posiciones de memoria 02 y 03, dejando el resultado en R4
6. Calcular la parte entera de la raíz cuadrada del valor entero contenido en la posición de memoria 02, dejando el resultado en R2
7. Calcular el promedio de los valores enteros contenidos en las posiciones de memoria 02 y 03, dejando el resultado en R1
8. Contar la cantidad de ceros que tiene la representación binaria del contenido de la posición de memoria 02, dejando el resultado en R0
9. Si el valor entero contenido en la posición de memoria 02 corresponde al número del código ASCII de una mayúscula, calcular el número del código ASCII de la minúscula correspondiente, y viceversa, dejando el resultado en R0. Si el valor no corresponde al número del código ASCII de una letra, en R0 se debe dejar el valor original
10. Invertir el orden de las últimas 32 posiciones de memoria