

Contexto

Evaluación: LAB 2
Semestre: 2024-2
Curso: 1ELE01 ARQUITECTURA DE COMPUTADORAS

Pregunta 1 (5 puntos)

Utilizarás la computadora diseñada en el ejemplo de [la grabación](#) compartida el martes pasado. Esta ya se encuentra en el circuito llamado "preg1". Se pide guardar en la memoria ROM el conjunto de instrucciones que permitan guardar los cuatro primeros números primos, en orden creciente, en R3, R2, R1 y R0. No puede incrementar el tamaño de la ROM, sólo dispone de 16 espacios de memoria de 1 byte cada uno.

Al final de la ejecución de las instrucciones, los registros de esta computadora deben contener los siguientes valores:

	hex	dec
R3	02	002
R2	03	003
R1	05	005
R0	07	007

En su documento respuesta debe incluir la lista de instrucciones en formato hexadecimal ("0x"), y una breve descripción de lo que hace cada una de ellas. Luego, debe incluir una breve descripción de lo que hace todo el conjunto de instrucciones, y cómo es que este resuelve el problema planteado.

No está permitido modificar el circuito proporcionado para esta pregunta. Lo único que puede cambiar son las instrucciones.

SOLUCIÓN:

Instrucción en hexadecimal	Instrucción en binario	Descripción
0x00	0000 0000	Se niega el valor en el registro R0 y se coloca el resultado en R0. Por ende, R0 ahora contiene 0xFF.
0xcc	1100 1100	Se suma lo contenido en R0 consigo mismo y se guarda en R3. Esto se puede hacer

		colocando los bits correspondientes a A y B de la siguiente forma: dirA=00 y dirB=00. Dado que $0xFF+0xFF=0xFE$, se guardará este resultado en R3.
0xc3	1100 0011	Se niega lo contenido en R3 y se guarda en R3. Por tanto, R3 contendrá el valor de 0x01.
0xbf	1011 1111	Se suma R3 consigo mismo y se coloca en R2. En R2 se guardará el valor de 0x02.
0xaf	1010 1111	Se suma R3 y R2. El resultado de esta suma es 0x03 y se coloca en R2.
0xff	1111 1111	Suma R3 consigo mismo y se coloca en R3, por lo que contendrá el valor de 0x02
0x7e	0111 1110	Se suma R2 y R3. El resultado de esta suma es 0x05 y se coloca en R1
0x1f	0001 1111	Se suma R1 y R3. El resultado esta suma es 0x07 y se coloca en R0.

Descripción del conjunto:

El objetivo es obtener los 4 primeros números primos: 2, 3, 5 y 7. Dado que todos los valores empiezan en 0, se trata de obtener el 0x01 en alguno de los registros primero. Para ello, se puede negar el registro R0 y obtener 0xFF, cuyo valor se guarda en el mismo registro R0. Luego, al sumar R0 consigo mismo, se obtiene 0xFE ($0xFF+0xFF=0xFE$) y se guarda en el registro R3. Al negar el registro R3, se obtendrá el valor de 0x01 y, en este caso, se guarda el resultado en el registro R3. A partir de allí, se puede obtener el valor de **0x02** sumando R3 consigo mismo, el cual guardamos en otro registro. Con ambos valores, se puede obtener el valor de **0x03**. Luego, con los sumandos correctos, se pueden obtener **0x05** y **0x07**, lo cual resuelve el problema planteado.

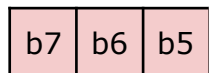
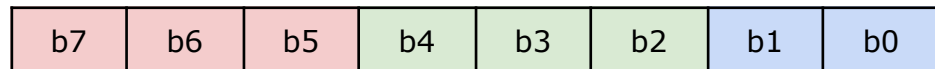
NOTA: Para sumar un registro 2 veces o consigo mismo ($R0+R0$), se puede colocar los bits correspondientes a A y B de la siguiente forma:

RES	DIRA	OPER	DIRB
11(R3)	00(R0)	11(SUMA)	00(R0)

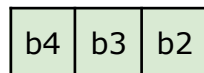
Pregunta 2 (10 puntos)

Se ha expandido la capacidad del banco de registros, ahora se tiene un banco de 8 registros de 1 byte cada uno. Además se ha cambiado la sintaxis de las instrucciones, de tal forma que ahora los bits significan los siguiente:

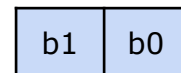
Instrucción:



Operación



Dirección del
operando A
(dirA)



Dirección del
operando B (dirB)
ó
Número "k", de dos
bits (para LOAD)

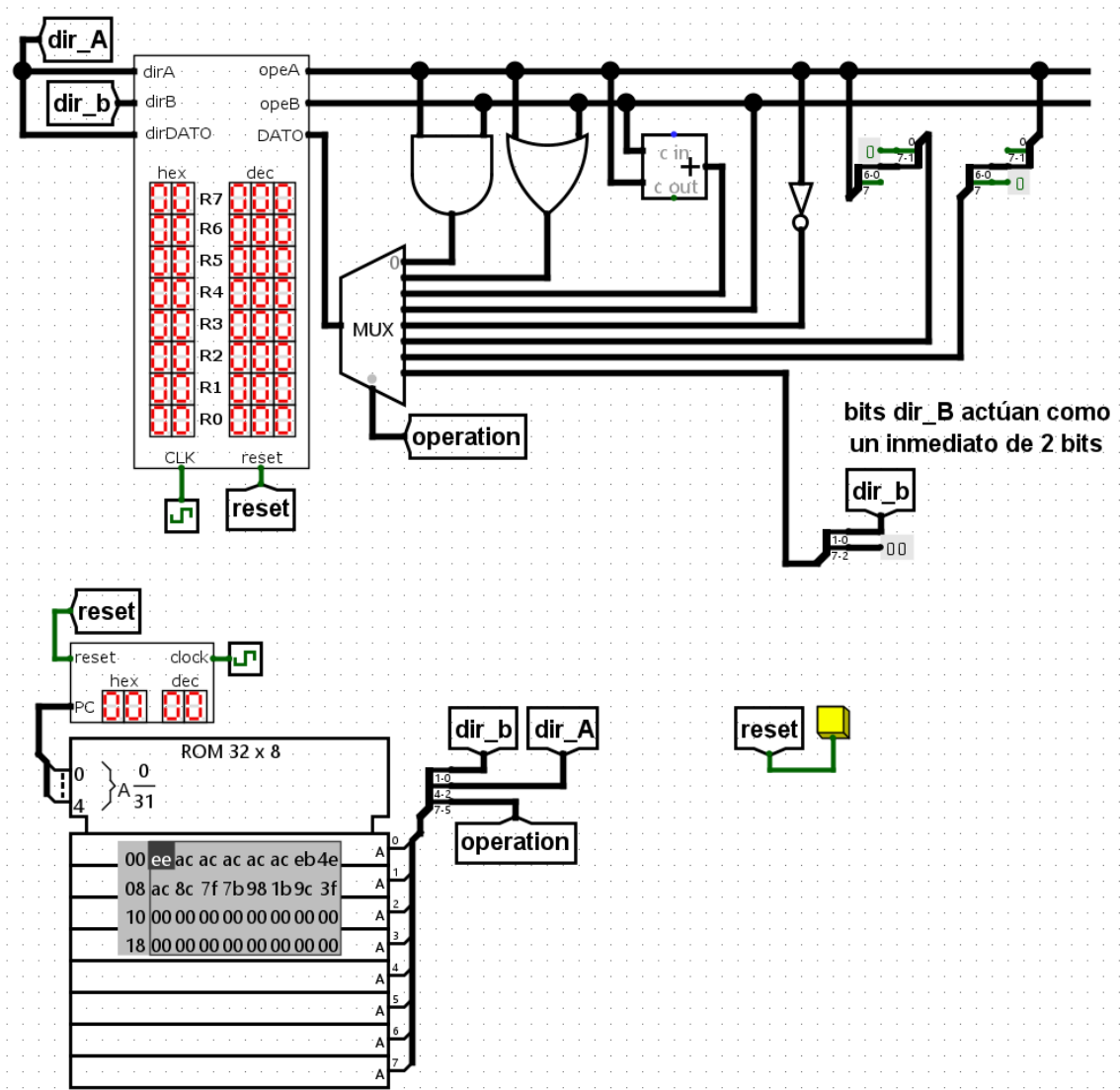
La codificación de las operaciones de esta nueva computadora es:

Operación	Código	Sintaxis	Ejemplo	Descripción del ejemplo
AND	000	AND RA, RB	AND R1, R0	$R1 \leftarrow R1 \text{ and } R0$
OR	001	OR RA, RB	OR R2, R3	$R2 \leftarrow R2 \text{ or } R3$
SUM	010	SUM RA, RB	SUM R4, R1	$R4 \leftarrow R4 + R1$
COPY	011	COPY RA, RB	COPY R6, R2	Copia el valor de R2 en R6 $R6 \leftarrow R2$
NOT	100	NOT RA	NOT R7	Complemento a 1 de R7, se guarda en el mismo R7.
LSL	101	LSL RA	LSL R4	Logical Shift to the Left. Mueve todos los bits del registro un espacio a la izquierda. Si $R4 \leftarrow 0b1001010$ Entonces ahora: $R4 \leftarrow 0b10010100$
LSR	110	LSR RA	LSR R4	Logical Shift to the Right Mueve todos los bits del registro un espacio a la derecha. Si $R4 \leftarrow 0b11001010$ Entonces ahora: $R4 \leftarrow 0b1100101$
LOAD	111	LOAD RA, k	LOAD R5, 2	Load a number into register $R5 \leftarrow 0b00000010$ Carga el número 2 en R5

Diseñe y simule las partes que faltan de esta computadora en Logisim. Incluya una explicación de lo que hace cada parte de la computadora.

Puede añadir más elementos al circuito, pero no puede modificar los bloques brindados inicialmente en la [plantilla de esta pregunta](#).





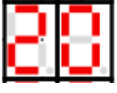
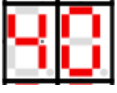
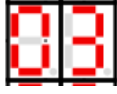


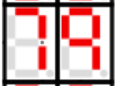



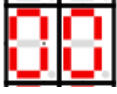
Este es el circuito que implementa todas las instrucciones solicitadas





Las instrucciones guardadas en ROM sirven para poner a prueba la computadora y todas sus instrucciones.

Las instrucciones son:

INSTRUCCIÓN	Lo que sucede luego de ejecutar esta instrucción		Lo que estoy haciendo
-------------	--	--	-----------------------

LOAD	R3, 0b10	$R3 \leftarrow 0x02$ $R3 \leftarrow 0b00000010$	 R3	<p>Estas primeras 10 instrucciones tienen un único objetivo:</p> <p>Guardar el número 0x79 en el registro R3</p> <p>La única instrucción que nos permite guardar números (inmediatos) en los registros es LOAD. Pero en nuestra computadora, la instrucción LOAD está limitada a cargar números de dos bits, entonces sólo podemos cargar 0,1,2 o 3 en cualquier registro. Si quiero guardar el número 0x79 en un registro, tengo que encontrar una forma de hacerlo combinando otras instrucciones.</p> <p>Seguro hay otras formas de lograr esto, con otras combinaciones de instrucciones. Sería interesante probar a ver podemos lograr lo mismo con menos de 10 instrucciones</p>
LSL	R3	$R3 \leftarrow 0x04$ $R3 \leftarrow 0b00000100$	 R3	
LSL	R3	$R3 \leftarrow 0x08$ $R3 \leftarrow 0b00001000$	 R3	
LSL	R3	$R3 \leftarrow 0x10$ $R3 \leftarrow 0b00010000$	 R3	
LSL	R3	$R3 \leftarrow 0x20$ $R3 \leftarrow 0b00100000$	 R3	
LSL	R3	$R3 \leftarrow 0x40$ $R3 \leftarrow 0b01000000$	 R3	
LOAD	R2, 0b11	$R2 \leftarrow 0x03$ $R2 \leftarrow 0b00000011$	 R2	
SUM	R3, R2	$R3 \leftarrow 0x43$ $R3 \leftarrow 0b01000011$	 R3	
LSL	R3	$R3 \leftarrow 0x86$ $R3 \leftarrow 0b10000110$	 R3	
NOT	R3	$R3 \leftarrow 0x79$ $R3 \leftarrow 0b01111001$	 R3	
COPY	R7, R3	$R7 \leftarrow 0x79$ $R7 \leftarrow 0b01111001$	 R7	<p>Estas instrucciones buscan poner a prueba las instrucciones COPY, NOT, AND y OR.</p> <p>Se está copiando el valor de R3 en R6 y R7. Ahora tendré el número 0x79 guardado en R6 y R7.</p> <p>Para poner a prueba la instrucción AND y OR, estoy invirtiendo todos los bits de R6 y R7 con la instrucción NOT.</p> <p>Luego, estoy haciendo una operación AND del registro R6 con R3 y una operación OR del registro R7 con R3.</p> <p>R6 y R7 tienen un número en el cual los bits son la versión invertida del número que está en R3.</p>
COPY	R6, R3	$R6 \leftarrow 0x79$ $R6 \leftarrow 0b01111001$	 R6	
NOT	R6	$R6 \leftarrow 0b10000110$	 R6	
AND	R6, R3	$R6 \leftarrow 0b00000000$	 R6	

NOT	R7	$R7 \leftarrow 0b10000110$	 R7	Si todo está bien, entonces el resultado del AND debería ser 0x00, y el resultado del OR debería ser 0xFF. Esto se debe a que el AND de un número con su versión negada siempre me dará puros cero, mientras que el OR de un número con su versión negada siempre me dará puros unos.
OR	R7, R3	$R7 \leftarrow 0b11111111$	 R7	

Aquí hay un [archivo txt](#) que pueden cargar en la ROM para ponerla a prueba