IIC2343 - Material de estudio I2 - 2020-1

PREGUNTA FLOAT:

I1-2014-1

Pregunta 2

Considere el estándar IEEE 754 para la representación de números de punto flotante de precisión simple (32 bits). ¿Cuántos números se pueden representar con este estándar, sin considerar ±∞ ni NaN?

Solución:

Según el estándar, podemos representar un total de 232 combinaciones posibles. De esas combinaciones, tenemos que 2 corresponden a la representación de los infinitos y $223-1$ corresponden a NaN (223 porque la diferenciación se produce con los bits del significante, considerando que alguno debe ser distinto de 0, y por eso se le resta el caso en que todos son 0's). Además, considerar que existen dos representaciones del 0, por lo que quitamos una de ellas, obteniendo aśı una cantidad de $232-2-(223-1)-1 = 4286578686$ combinaciones posibles.

Distribución Puntaje:

Combinaciones totales 20 %.

Infinitos 20 %.

NaN 20 %.

0's 20 %.

Resultado Final con Distribución de bits 20 %.

PREGUNTA FLOAT:
I1-2014-1
Pregunta 3
Escriba en formato float el número −48. Indique cómo se compone y qué significa cada una de las partes de la secuencia de bits.

Solución:
El formato float, según el estándar IEEE 754, se representa con 32 bits, divididos de la siguiente forma:

1 bit de signo

8 bits para el exponente (desplazado en +127)

23 bits para el significante normalizado

Aśı, la representación se veŕıa de la siguiente forma:

Signo Exponente Significante
1 bit 8 bits 23 bits

Por lo tanto, tenemos que:

Dado que el número es negativo, el bit de signo valdrá 1.

Tenemos que 48 = 110000b. A continuación, debemos normalizar el número binario. Tenemos que 110000b = 1,10000·25b. Por lo tanto, se tie que el significante = 10000000000000000000000.

Finalmente, debemos determinar el exponente. Del procedimiento anterior, obtuvimos que el exponente es 5. Pero es necesario desplazarlo en +127. Es decir, exp = 5 + 127 = 132 = 10000100b.

Aśı, juntando todo, tenemos que la representación seŕıa:

S Exp Significante
1 10000100 10000000000000000000000

Distribución:

Bit signo 10 %.

Significante 30 %

PREGUNTA ARQUITECTURA COMPUTADOR BÁSICO

I1-2016-1
Pregunta 3

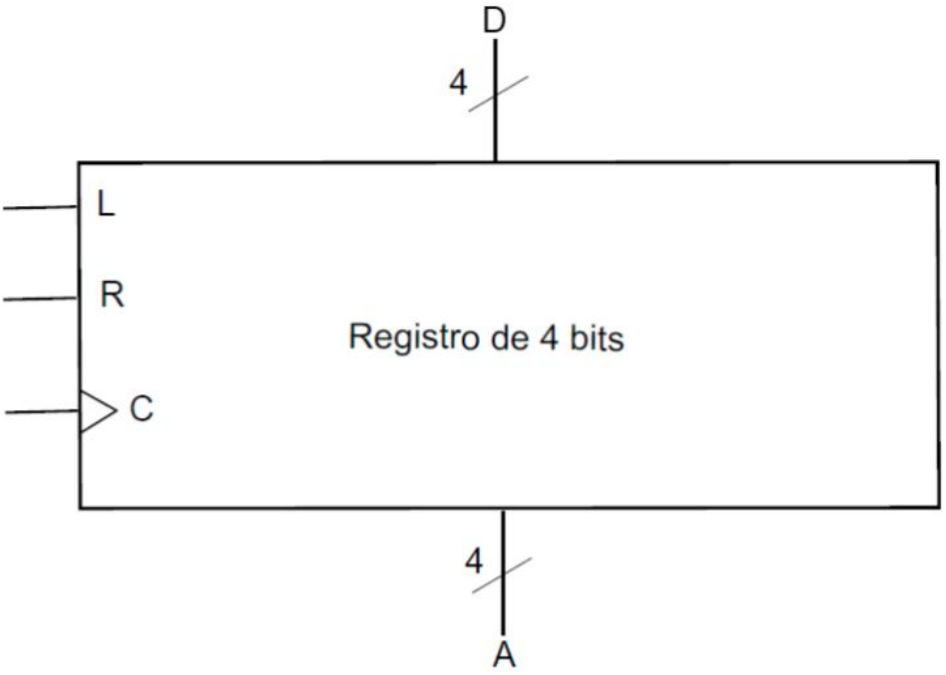
a) Una máquina RAM, es un tipo de computador en el cual se utiliza sólo la memoria RAM de datos
para almacenar los resultados de las operaciones aritméticas y lógicas. Una máquina RAM puede tener
más registros para uso interno, pero para un programador, sólo se encuentran expuestos la memoria y
los literales para realizar las operaciones. Por ejemplo, ADD A,B es sustituida por ADD (Dir1),(Dir2).
Construya una máquina RAM que posea las mismas funcionalidades que el computador básico. Especifique detalladamente el hardware y el formato de las instrucciones (señales de control, opcodes, assembly).
(3 ptos.)
Solución:

Para construir esta máquina basta con utilizar el hardware del computador básico, manteniendo además todos los opcodes, pero modificando el assembly, de tal manera que las operaciones que utilizan registros sean sustituidas por las que utilizan las direcciones de memoria. La implementación de cada una de estas instrucciones se puede realizar trivialmente utilizando una secuencia de opcodes adecuada, que implemente con instrucciones del computador básico la misma funcionalidad.

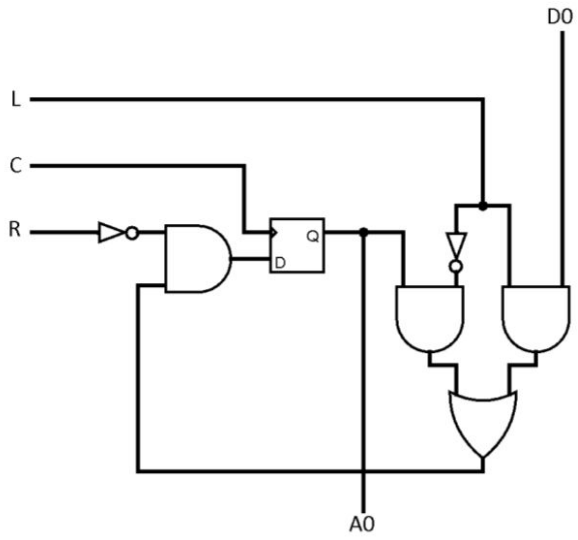
Por ejemplo, la secuencia para ADD (Dir1),(Dir2) seŕıan los opcodes correspondientes a la siguiente secuencia de instrucciones del computador básico: MOV A, (Dir1) ; MOV B, (Dir2) ; ADD A, B ; MOV (Dir1), A.

PREGUNTA CIRCUITOS SECUENCIALES
I1-2012-1
Pregunta 3

a) Implemente mediante compuertas lógicas y flip-flops tipo D, el registro de la figura, con señales de control (C), carga (Load) y reset (Reset), que funciona con flanco de subida. **(3 ptos.)**



Solución: Por motivos de espacio, se presenta la solución para el bit menos significativo. Para el resto de los bits el diagrama es el mismo.

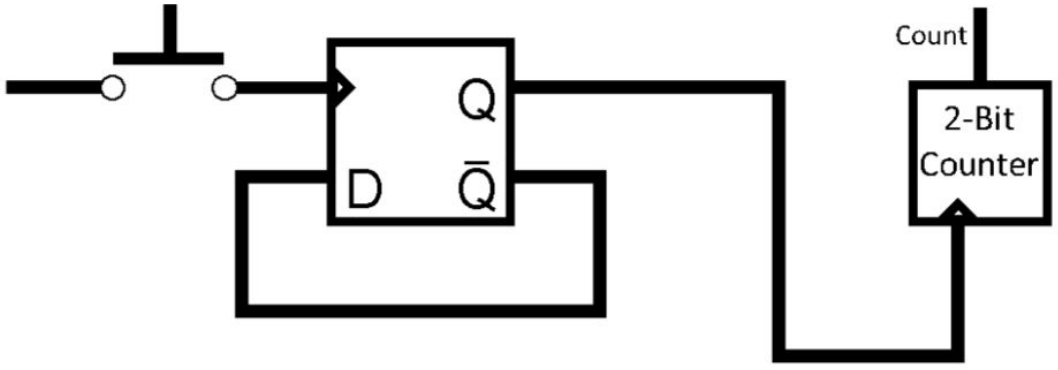


PREGUNTA CIRCUITOS SECUENCIALES
I1-2012-1

Pregunta 3

b) Diseñe, utilizando todos los elementos de circuitos lógicos vistos en clases que necesite, un contador secuencial circular ascendente de 2 bits, que se incrementa cada dos flancos de subida de la señal de control. **(3 ptos.)**

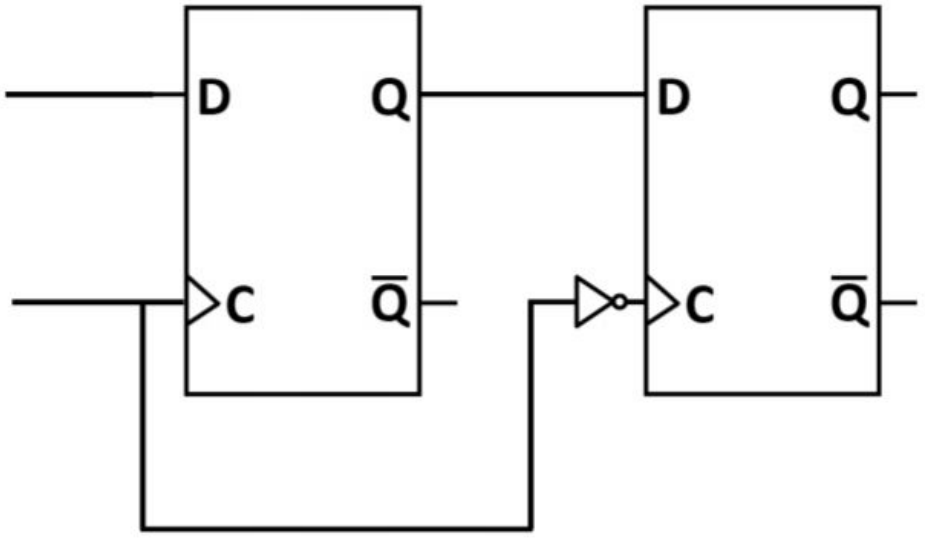
Solución:
La siguiente solución asume que el Flip-Flop parte con un 1 almacenado:



I1-2012-2
Pregunta 2

a) Implemente mediante compuertas lógicas, elementos de control y latches, un flip-flop tipo D que funcione con flanco de bajada. (3 ptos.)

Solución:



PREGUNTA ARQUITECTURA COMPUTADOR BÁSICO
2019-2

Pregunta 4 (2 ptos.)

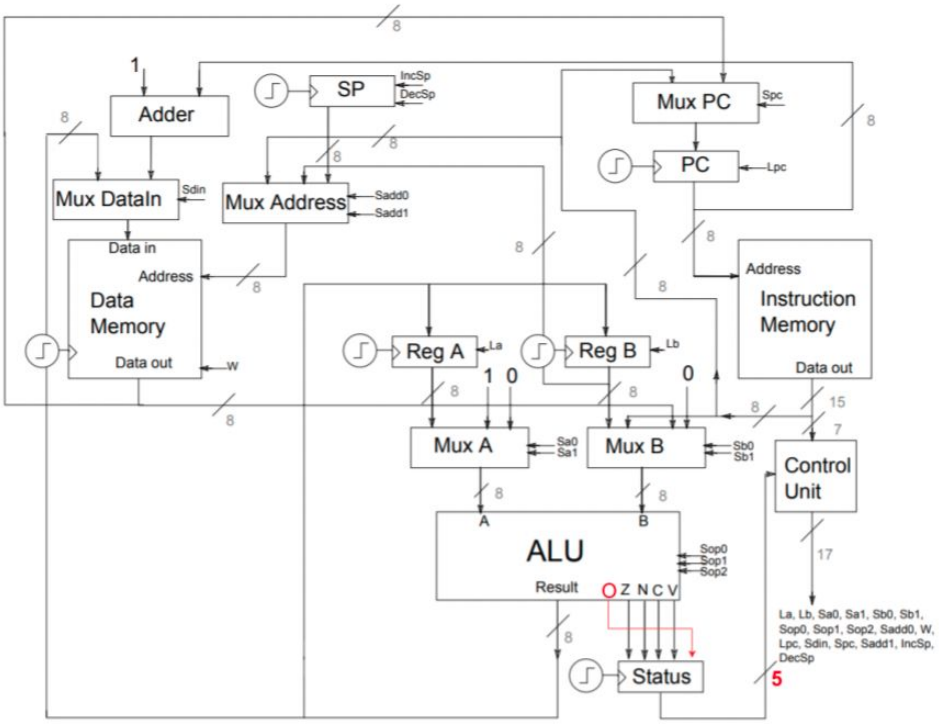
Se requiere agregar las siguientes instrucciones al computador básico. Para cada caso indique si es posible agregarlas sin modificar el hardware del computador, en cuyo caso debe especificar las señales de control que se deben habilitar para ejecutar la instrucción. En caso de que se requiera modificar el computador, realice los cambios en el diagrama adjunto, especifique las nuevas señales de control agregadas (si corresponde) e indique que señales de control se deben habilitar para ejecutar la instrucción.

1. **(0,5 ptos.)** Agregue la instrucción `JPO Dir` la cual salta a la instrucción asociada al label `Dir` si el resultado de la instrucción anterior fue impar.

   `RESPUESTA:`

   - No es posible agregar esta instrucción sin modificar el hardware del computador. Esto debido a que los saltos condicionales existentes en el computador básico ocupan las señales: `Z` (resultado de la ALU es cero), `N` (el bit más significativo del resultado de la ALU, indicando que este es negativo cuando es uno), `C` (pérdida de un bit) y `V` (overflow). Ninguna de esta señales, ni mezcla de ellas nos indica que el resultado es impar, por lo que es necesario agregar hardware. **(+ 0,1 pts.)**

- Modificación: **(+ 0,2 pts.)**



- Especificación de señales agregadas: **(+ 0,1 pts.)**
  Se agrega la señal condicional `O`, la cuál será el bit menos significativo del resultado de la ALU, indicado que es impar cuando es uno.
- Indicación de señales de control a ejecutar: **(+ 0,1 pts.)**
  NOTA: Toda señal de cargado que no esta en la tabla (como La, Lb, W, etc) son cero.

Instrucción	Condición	Lpc	Spc0
JPO	O=1	1	LIT

4. **(0,5 ptos.)** Agregue la instrucción `MOV B,((Dir))` la cual guarda en el registro B el valor en memoria en la posición indicada por la posición memoria asociada al label `Dir`. (`B = MEM[MEM[Lit]]`)

   `RESPUESTA:`

   - Si es posible agregar esta instrucción sin modificar el hardware del computador, en especifico esta instrucción se puede realizar ejecutando dos ciclos. Primero ejecutando las mismas señales de la instrucción `MOV B, (Dir)`, y finalmente en el siguiente ciclo ejecuta las mismas señales de la instrucción `MOV B, (B)`. **(+ 0,2 pts.)**
   - Indicación de señales de control a ejecutar: **(+ 0,3 pts.)**
     NOTA: Toda señal de cargado que no esta en la tabla (como La, Lpc, W, etc) son cero.

Instrucción	Lb	Sadd0,1	Sop0,1,2	Sb0,1	Sa0,1
MOV	1	LIT	ADD	Dout	ZERO
B,((Dir))	1	B	ADD	Dout	ZERO

2013-2

Pregunta 2 (2,0 ptos)

Pregunta 2.1 (1,0 ptos)

Para un determinado programa la *secuencia efectiva* de instrucciones corresponde a la secuencia de todas las instrucciones que se ejecutaron desde que comenzó hasta que terminó el programa, incluyendo las instrucciones que se repitieron por saltos o llamados a subrutinas.

A modo de ejemplo, para el siguiente programa:

```
CODE:
            MOV A, 0     //Instruccion 0
            MOV B, 2     //Instruccion 1
    label1: CMP A,B      //Instruccion 2
            JGE end      //Instruccion 3
            ADD A,1      //Instruccion 4
            JMP label1   //Instruccion 5
    end:
```

La secuencia efectiva de instrucciones es:

```
        MOV A, 0     //Instruccion 0
        MOV B, 2     //Instruccion 1
        CMP A,B      //Instruccion 2
        JGE end      //Instruccion 3
        ADD A,1      //Instruccion 4
        JMP label1   //Instruccion 5
        CMP A,B      //Instruccion 2
        JGE end      //Instruccion 3
        ADD A,1      //Instruccion 4
        JMP label1   //Instruccion 5
        CMP A,B      //Instruccion 2
        JGE end      //Instruccion 3
```

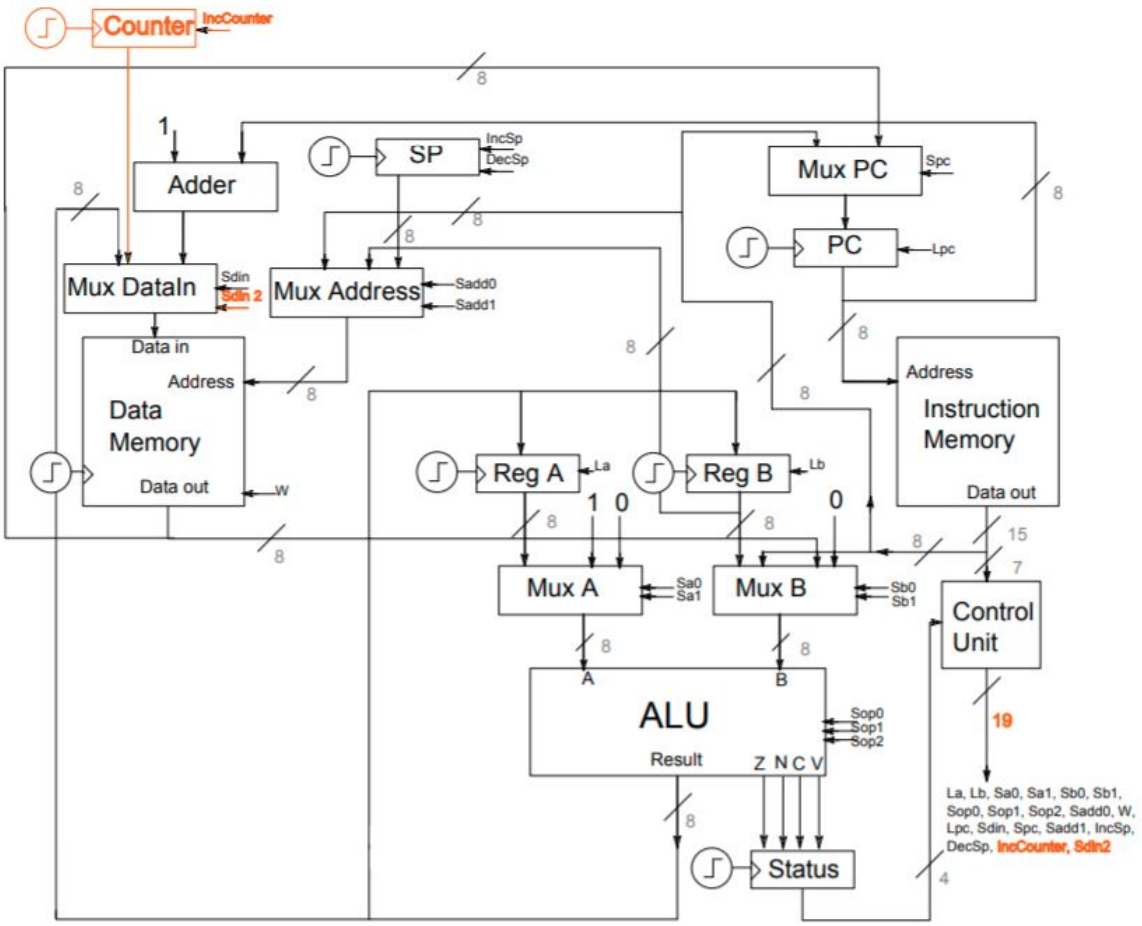
Realice las modificaciones necesarias al computador básico de manera que se pueda implementar la instrucción `NUM_INSTRUCTIONS (var1)` la cual almacena en la variable `var1` la cantidad de instrucciones que efectivamente se ejecutaron hasta que se llamó a esta instrucción (no incluyendo a esta instrucción).

A modo de ejemplo, para el programa anterior, si se llama a esta instrucción luego del label `end` se debería almacenar el valor 12 en la variable `var1` dado que se ejecutaron 12 instrucciones.

Importante 1: Para esta pregunta puede asumir que todas las instrucciones se ejecutan en un ciclo del clock.

Importante 2: No puede agregar memorias para resolver esta pregunta.

Solución:



- Se agrega un contador al computador con una señal de control IncCounter, que hace que el contador se increment si está habilitada.
- Se aumenta el tamaño del Mux de data in de la memoria de datos, y se agrega la salida del contador como entrada posible de este Mux.
- A todas las instrucciones se les agrega la señal de control IncCounter con valor 1, para que se incremente el contador luego de cada llamado, exceptuando a la nueva instrucción `NUM_INSTRUCTIONS (var1)` la cual no activa esta señal.
- Las señales de control relevantes de la nueva instrucción `NUM_INSTRUCTIONS (var1)` son:
  - La señal de selección del MUX Data In se elige para que se escoja el valor del contador como entrada a la memoria (Sdin = 10 por ejemplo).
  - Como dirección para la memoria de datos se elige Saddress = Literal.
  - La señal IncCounter se setea en 0.
  - Todas las otras señales de Load o Incremento/Decremento se setean en 0.