DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (II/2025)

Guía de ejercicios: Modificación del Computador Básico

Ayudantes: Daniela Ríos (danielaarp@uc.cl), Alberto Maturana (alberto.maturana@uc.cl), José Mendoza (jfmendoza@uc.cl)

Pregunta 1: Preguntas Conceptuales

(a) ¿Cuál es la función del *clock* en un computador? ¿Que pasaría si este es removido?

> **Solución:** El *clock* permite sincronizar todos los elementos de memoria de un computador. Sin él, no existe garantía de que los resultados almacenados en los elementos de memoria, luego de cualquier operación, sean correctos.

(b) Dada la microarquitectura del computador básico, ¿es posible crear una **ISA** distinta la actual? Argumente su respuesta.

> **Solución:** Es posible crear nuevas ISAs usando un subconjunto de las instrucciones soportadas por la ISA actual. Además, es posible utilizar también un superconjunto, utilizando instrucciones que no tienen un *opcode* asignado, pero que tienen una combinación de señales de control.

(c) Explique en detalle que pasaría en el computador básico si el registro PC cambia su funcionamiento de flanco de subida a flanco de bajada.

> **Solución:** Al cambiar la sincronización, el almacenamiento en los registros o memoria, no coincidiría necesariamente con lo que la instrucción espera almacenar. Dependiendo de la velocidad de ejecución de la instrucción, algunas instrucciones funcionarán de la misma manera, mientras que otras almacenarán resultados incorrectos.

(d) Si se elimina el condition code Z del registro Status, ¿que otro posible condition code podría crearse para ser usado en conjunción con N, de manera de emular el comportamiento de Z?

Solución: Dado que se tiene el condition code N, un condition code que podría ser usado en conjunción con este sería el condition code M, que es uno cuando el resultado de la última operación de la ALU es mayor que cero. De esta manera, si N=0 y M=0, entonces el número en la ALU no es ni positivo ni negativo, por lo que es cero, lo que implicaría Z = 1. Para implementar M basta con hacer un `AND` de los 8 bits de la ALU, donde a diferencia del condition code Z, el bit más significativo llega negado al `AND`.

(e) ¿Que modificación realizaría al computador básico para reducir el número de instrucciones de los programas? Describa brevemente como lo implementaría. Este problema es abierto a su creatividad ya que existen múltiples respuestas.

Solución: Una modificación sencilla y muy efectiva es agregar a la ALU unidades de multiplicación y división. De esta manera, cada vez que se necesite realizar una de estas operaciones, se necesitará sólo una instrucción, a diferencia del caso actual, en donde se necesita realizar un *loop*.

(f) Explica el rol del multiplexor *Address*, da un ejemplo de su funcionamiento.

Solución: Su rol es poder seleccionar la dirección que se desea utilizar en la memoria de datos (tanto para lectura como escritura). Existen dos posibilidades, la dirección proveniente como literal de la memoria de instrucciones (direccionamiento directo) y la dirección obtenida desde el registro `B` (direccionamiento indirecto). Un ejemplo de esto es el contraste entre las instrucciones `MOV (Dir),A` y `MOV (B),A`. En el primer caso, sin pérdida de generalidad, la unidad de control le asignara el valor 0 a la señal S_{add0} para que el multiplexor *Address* seleccione el bus proveniente de la memoria de instrucciones (que sera la dirección de `Dir`), mientras que en el segundo asignara el valor 1 para seleccionar el valor del registro `B`. No obstante, en ambos casos el valor escrito sera el contenido en el registro `A`.

Pregunta 2: Modificación del Computador Básico (P4-I2-2011-2)

Se desea modificar el computador básico a nivel de microarquitectura e ISA. Para los siguientes puntos detalle las modificaciones que haría. Debe utilizar diagramas de componentes y conexiones y tablas de opcodes e instrucciones cuando corresponda. Las modificaciones son acumulativas.

(a) Agregar un tercer registro, que cumpla la función de acumulador de resultados y que pueda usarse para direccionamiento indirecto.

Solución: En este caso, para permitir el paso de C a la ALU, se agrega la salida de este a Mux A.

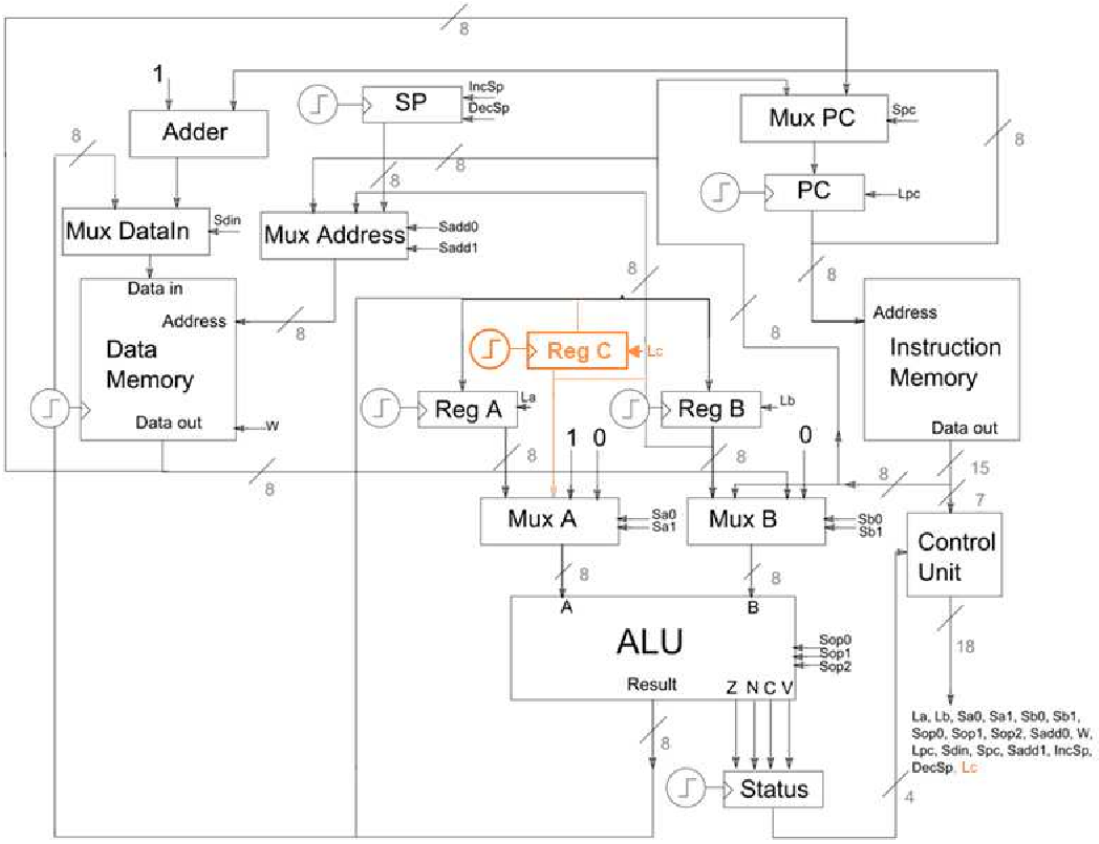

Figura 1: Diagrama Solución P2.a.

(b) Agregar las instrucciones `ADD3 reg1, reg2, reg3` y `SUB3 reg1, reg2, reg3`, que toman los valores en los registros `reg1`, `reg2` y `reg3`, los suma/resta y los almacena en `reg1`.

Solución: Se asume que el orden de los operandos de las operaciones corresponden a `A=reg1`, `B=reg2` y `C=reg3`. Además del diagrama, es necesario especificar el comportamiento de las nuevas señales de control. Sop3 es 0 cuando la operación elegida es la suma de 3 registros, mientras que es 1 cuando es la resta. Por otro lado, la señal Sa2 es 0 cuando se quiere almacenar en el registro A el resultado de la operación `SUB3/ADD3`, y es 1 cuando se quiere almacenar el resultado proveniente de la ALU. Dado que aun quedan opcodes de 7 bits disponibles, se asignan los opcodes 1001111 para `ADD3` y 1010000 para `SUB3`.

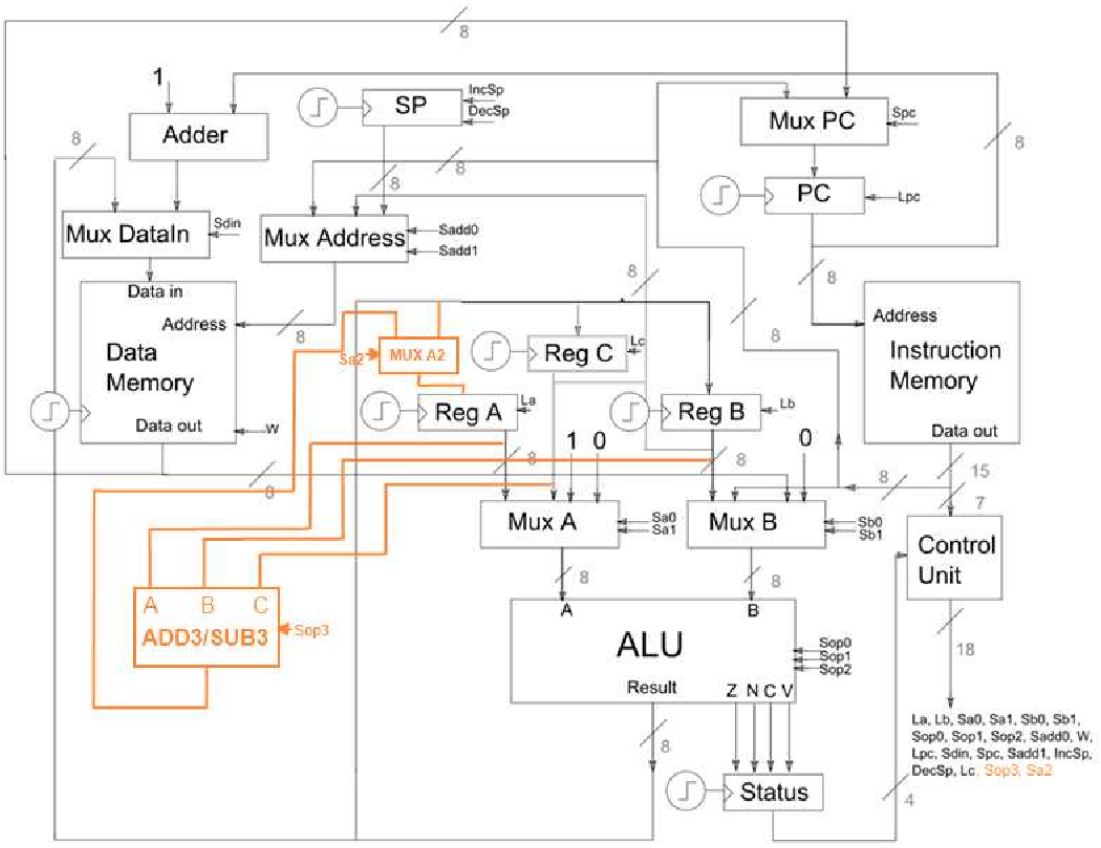

Figura 2: Diagrama Solución P2.b.

(c) Soportar direccionamiento indirecto con registro base y registro índice.

Solución: Para seleccionar como entrada de Mux Address el resultado de la suma de `A` y `B`, se utiliza el valor 1 tanto para Sadd0 y Sadd1. La instrucción de transferencia que hace uso de este direccionamiento será `MOV A, (A+B)`, que tendrá el opcode 1010001 y la única diferencia con las señales de control de `MOV A, (B)` son los valores de Sadd0 y Sadd1.

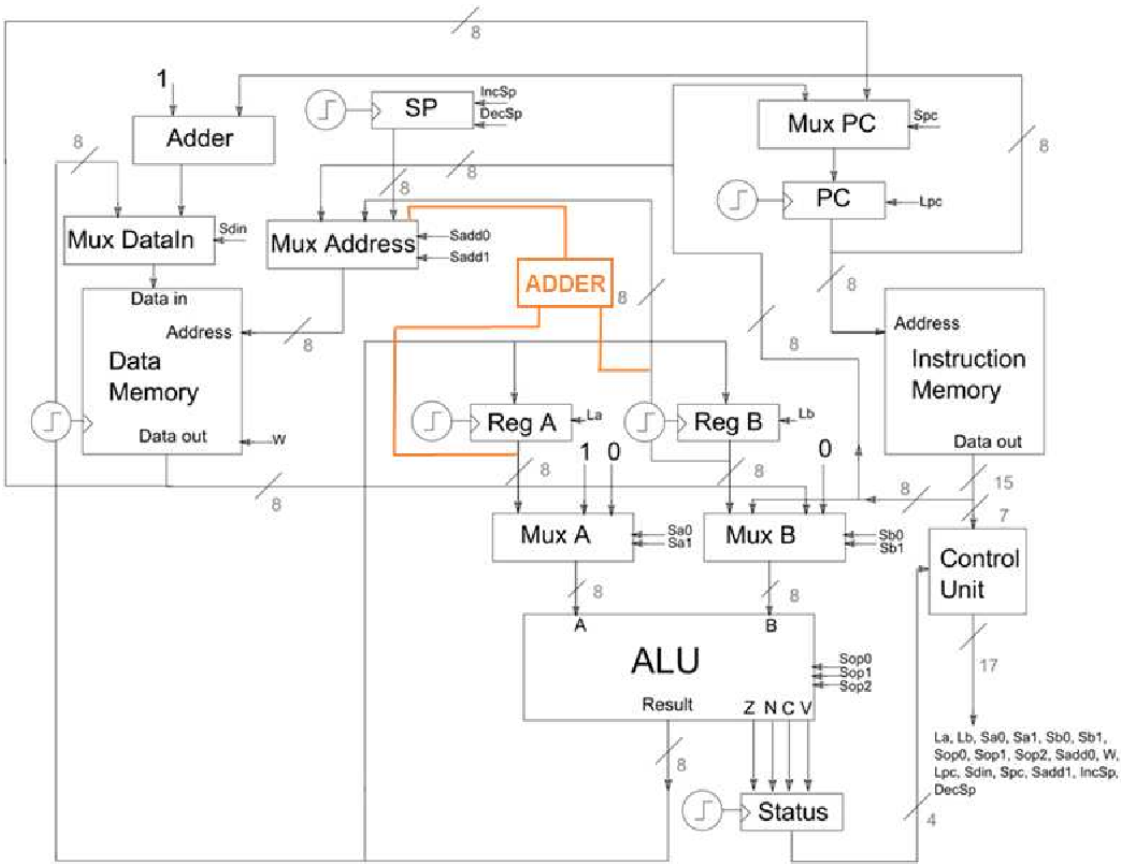

Figura 3: Diagrama Solución P2.c.

Pregunta 3: Modificación del Computador Básico (P3-I2-2012-1)

Se desea modificar el computador básico a nivel de microarquitectura e ISA. Para los siguientes puntos detalle las modificaciones que haría. Debe utilizar diagramas de componentes y conexiones y tablas de opcodes e instrucciones cuando corresponda.

(a) Permitir al computador la ejecución de dos operaciones aritméticas-lógicas iguales, pero con distintos argumentos, de manera simultánea, i.e., el proceso debe tomar sólo un ciclo del *clock*. Por simplicidad, se espera que el resultado de la segunda operación se guarde en el registro B.

Solución: La modificación presentada a continuación agrega 8 nuevas instrucciones al computador, donde cada una de estas corresponde a la versión paralela de las operaciones aritmético-lógicas. Estas instrucciones toman los valores de los registros A y B, los modifican de acuerdo a la operación y al literal ingresado como parámetro, y luego los almacenan nuevamente en los registros A y B. Por ejemplo, la nueva operación `ADDP 5`,

es análogo a ejecutar `ADD A, 5` y `ADD B, 5`. Desde el punto de la microarquitectura, se agrega una ALU adicional, cuyo output se conecta al registro B, por medio del nuevo MUX B2. Además, se agrega la nueva señal *Sb2*, que controla al MUX B2.
La siguiente figura muestra las modificaciones hechas a la microarquitectura:

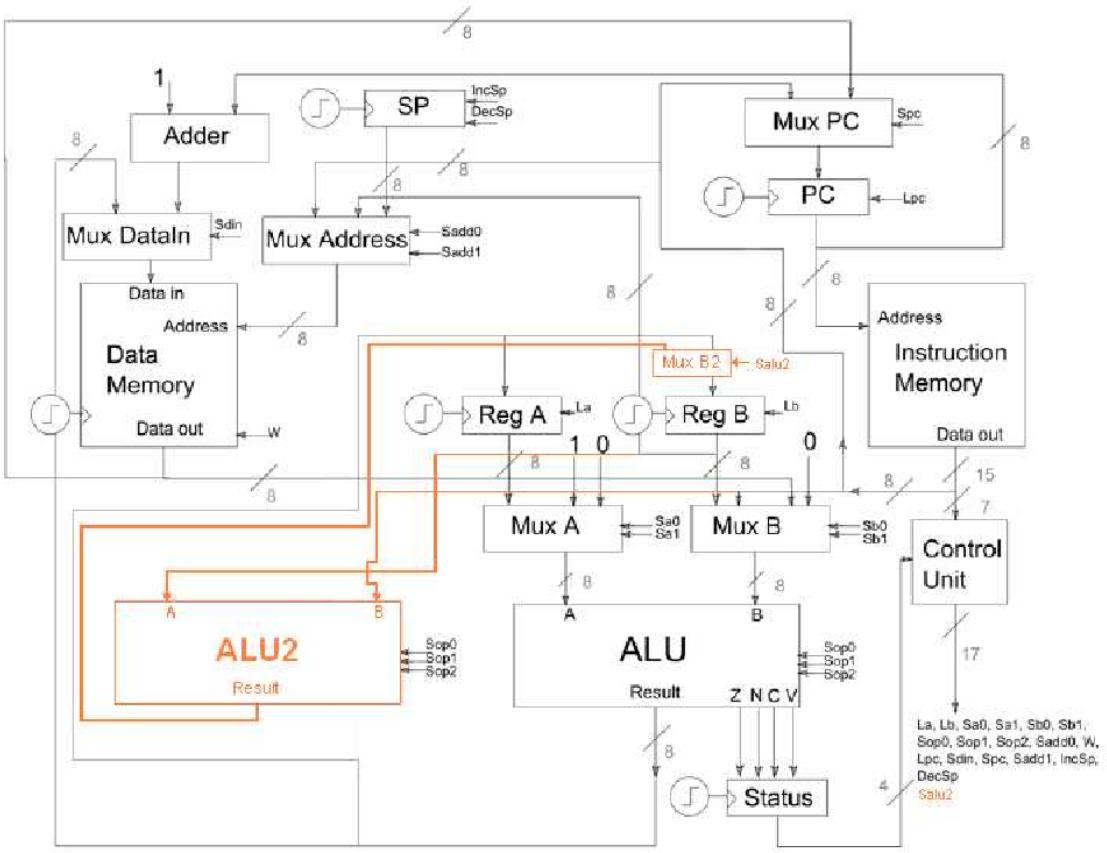

Figura 4: Diagrama Solución P3.a.

La siguiente tabla presenta los opcodes y señales de control de las nuevas operaciones, excluyendo las sieguiente señales, cuyo valor indicado a continuación se repita de la misma manera en todas las operaciones: Sadd=-, Sdin=-, Spc=-, W=0, IncSp=0 y DecSp=0.

Instr.	Op.	Opcode	Lpc	La	Lb	Salu2	Sa	Sb	Sop
ADDP	Lit	1001110	0	1	1	ALU2	A	LIT	ADD
SUBP	Lit	1001111	0	1	1	ALU2	A	LIT	SUB
ANDP	Lit	1010000	0	1	1	ALU2	A	LIT	AND
ORP	Lit	1010001	0	1	1	ALU2	A	LIT	OR
NOTP	Lit	1010010	0	1	1	ALU2	A	-	NOT
XORP	Lit	1010011	0	1	1	ALU2	A	LIT	XOR
SHLP	Lit	1010100	0	1	1	ALU2	A	-	SHL
SHRP	Lit	1010101	0	1	1	ALU2	A	-	SHR

(b) Permitir la autoprogramabilidad manteniendo memorias de datos e instrucciones separadas.

Solución: La modificación realizada cambia la memoria ROM de instrucciones por una memoria RAM, para que pueda ser escrita. Además se añade un mux (Mux Address 2) para elegir el origen de la dirección que ingresa a la memoria, que puede ser el PC o dirección la dirección obtenida de Mux Address. Para asegurar que la instrucción siguiente a la de escritura se ejecute correctamente, se agrega una nueva señal al PC, HaltPC, que evita que este se incremente por una iteración. También se agrega la instrucción NOP, que provoca que el computador no haga nada por un ciclo y se agrega un enabler (NOPEn) a la salida de la memoria de instrucciones, que al ser activado (W1=1) evita que pase la instrucción actual y genera el opcode de la nueva instrucción NOP, mientras que si no es activado (W=0), deja pasar el opcode ingresado. Finalmente, los datos de entrada para esta memoria se obtienen desde la ALU.

La siguiente figura muestra las modificaciones hechas a la microarquitectura:

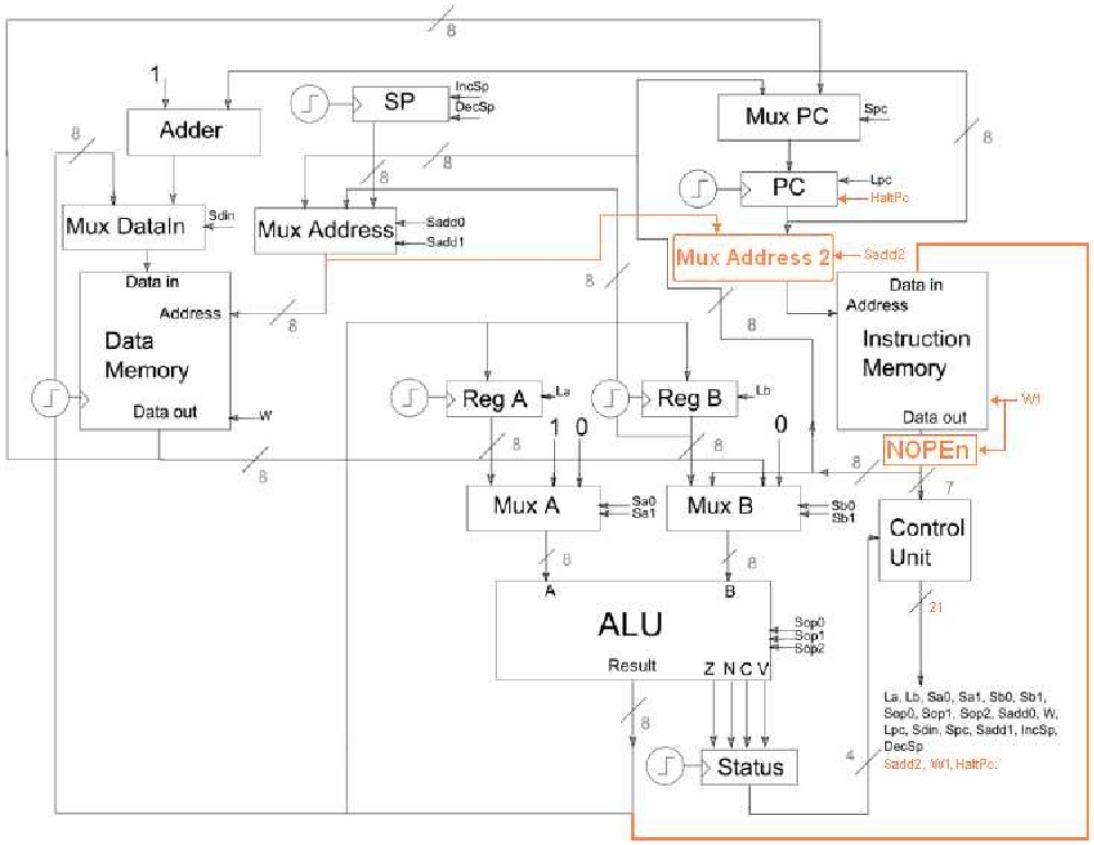

Figura 5: Diagrama Solución P3.b.

Desde el punto de vista de la ISA, además de NOP, se agregan las instrucciones MOV_DM (Dir), A y MOV_DM (B), A, que escriben el valor almacenado en el registro A en la memoria de datos. Cabe destacar que estas instrucciones toman 2 ciclos, una para escribir el dato en memoria y la siguiente ejecutando NOP La siguiente tabla presenta los opcodes y señales de control de las nuevas instrucciones, incluyendo las nuevas señales Sadd2,

HaltPc y W1:

Instr.	Op.	Opcode	Lpc	La	Lb	Sa	Sb	Sop	Sadd	Sdin	Spc	W
NOP	-	1001110	0	0	0	-	-	-	-	-	-	0
MOV_DM	(B), A	1001111	0	0	0	A	0	ADD	B	-	-	0
MOV_DM	(Dir), A	1010000	0	0	0	A	0	ADD	Lit	-	-	0

Instr.	Op.	IncSp	DecSp	Sadd2	HaltPc	W1
NOP	-	0	0	-	1	0
MOV_DM	(B), A	0	0	MuxAddr	1	1
MOV_DM	(Dir), A	0	0	MuxAddr	1	1

Pregunta 4: Modificación del Computador Básico (P2.1-I1-2013-2)

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

La secuencia efectiva de instrucciones es:

```
MOV A, 0    //Instrucción 0
MOV B, 2    //Instrucción 1
CMP A,B     //Instrucción 2
JGE end     //Instrucción 3
ADD A,1     //Instrucción 4
JMP label1  //Instrucción 5
CMP A,B     //Instrucción 2
JGE end     //Instrucción 3
ADD A,1     //Instrucción 4
JMP label1  //Instrucción 5
CMP A,B     //Instrucción 2
JGE end     //Instrucción 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: Una posible solución es la siguiente:

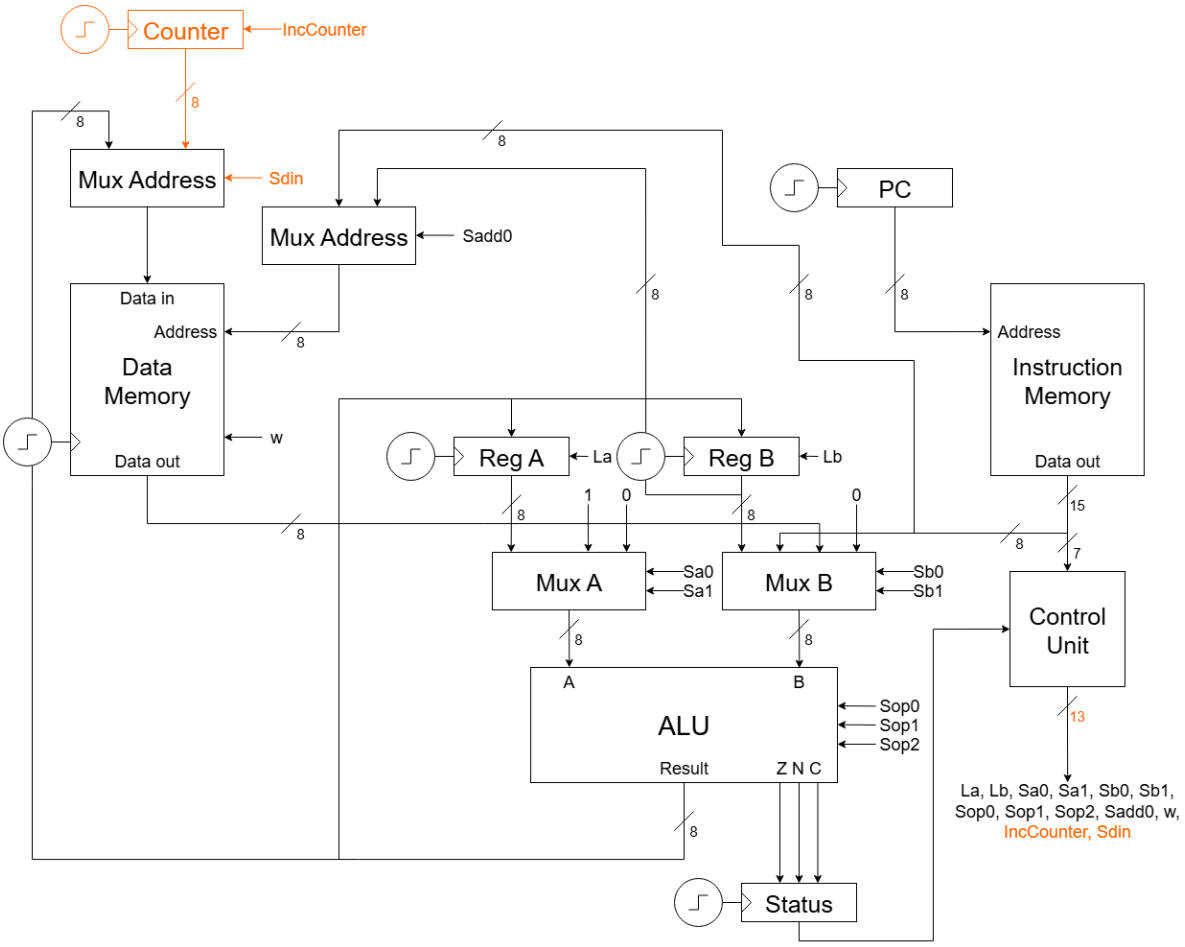

Figura 6: Diagrama Solución P4.

- Se agrega un contador al computador con una señal de control IncCounter, que hace que el contador se incrementa si está habilitada.
- Se aumenta el tamaño del Mux de data in en la memoria de datos, y 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 eincremente 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.

Pregunta 5: Modificación del Computador Básico (P3.b-I1-2016-1)

Modifique el computador básico para dar soporte a la instrucción CLEAR, que setea en 0 el valor de todos los registros (excepto PC y STATUS) y el de todas las palabras de la memoria de datos. Indique el(los) opcode(s) y señal(es) de control de la nueva instrucción.

Solución: La solución mas simple y directa es agregar a los registros A y B y a la memoria de datos, una señal CLR, que setea en 0 el valor almacenado, tal como el registro visto en cátedra. Esta señal se agrega al conjunto de señales de control generada por la unidad de control, y se seteará en 1 solo cuando la unidad de control reciba el opcode de la instrucción CLEAR (1001110):

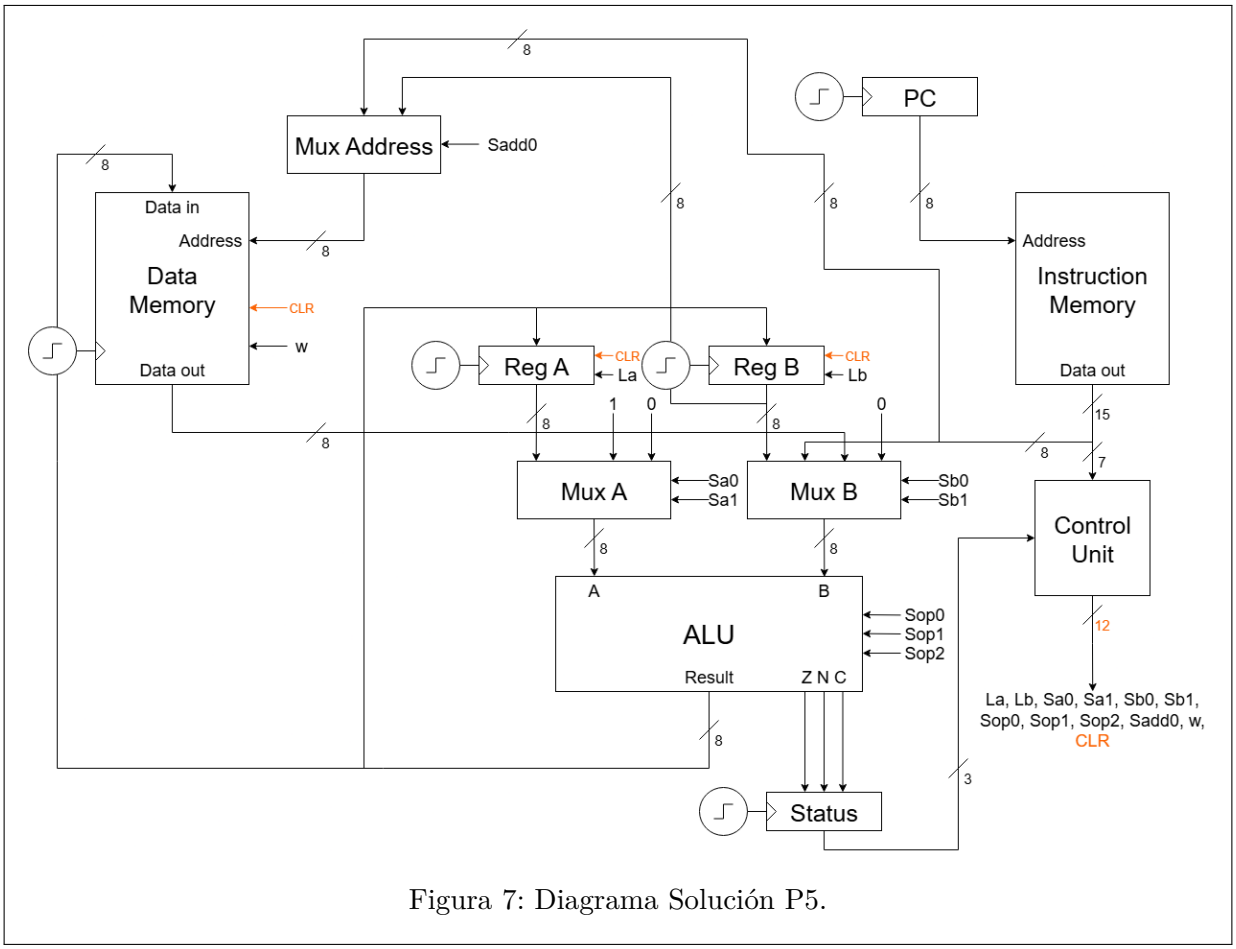

Figura 7: Diagrama Solución P5.

Pregunta 6: Modificación del Computador Básico (P4-I1-2022-2)

Para continuar con esta línea de investigación, se requiere agregar las siguientes instrucciones al computador básico. Para cada una indique si es posible agregarla sin modificar el *hardware* del computador. En caso de que se requiera modificar el computador, realice los cambios en el diagrama adjunto. Si corresponde, especifique las nuevas señales de control agregadas e indique el valor que deben tomar cada una de las señales de control para ejecutar cada instrucción.

Considerando un nuevo registro C de uso general:

(a) Agregue la instrucción `ADD A,C`, la cual permite guardar en el registro A el valor de la suma del registro A y el registro C.

Solución:

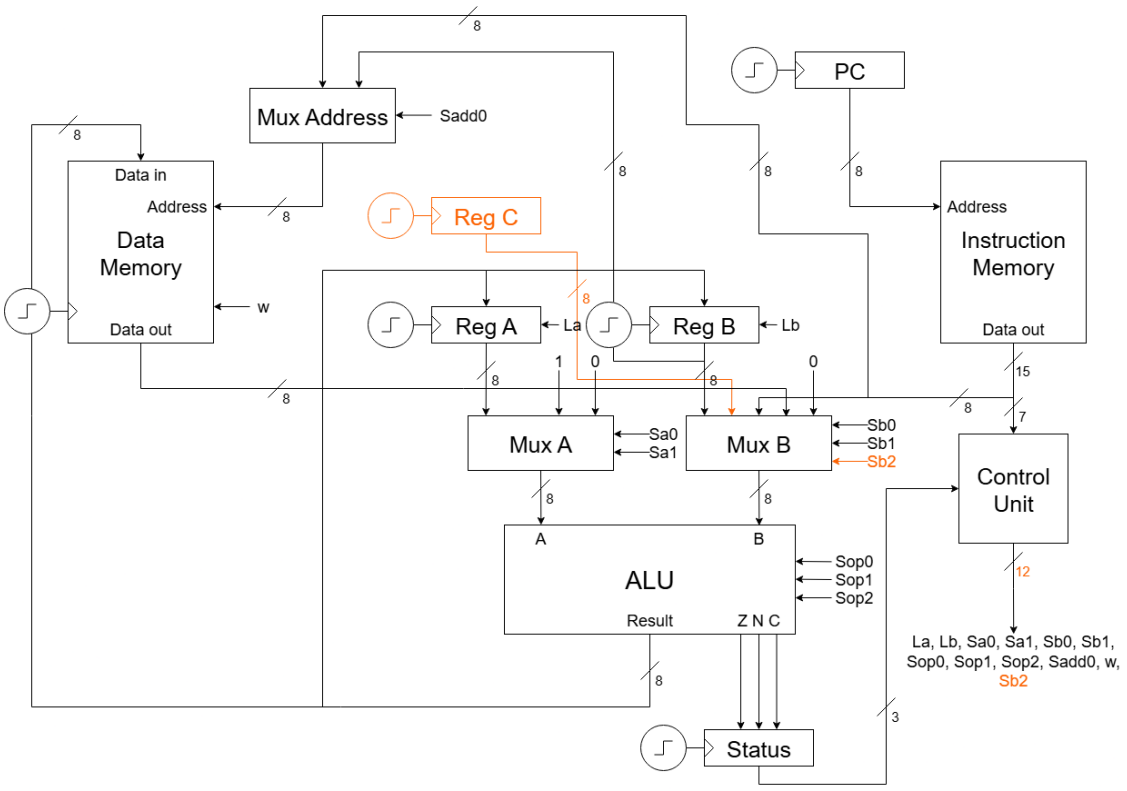

Figura 8: Diagrama Solución P6.a.

Para operar entre el nuevo registro C y el registro A, debo poder seleccionar el registro C en el Mux B, como a este muxer ya no le quedan entradas debo agrandar su selector agregando la señal `Sb2`. Con eso, para la instrucción `ADD A,C` las señales de control son:

Instr.	Op.	La	Lb	Sa0,1	Sb0,1,2	Sop0,1,2	Sadd0	W
ADD	A, C	1	0	A	C	ADD	-	0

(b) Agregue la instrucción `ADD C, Lit`, la cual permite guardar en el registro C el valor de la suma del registro C y el literal.

Solución:

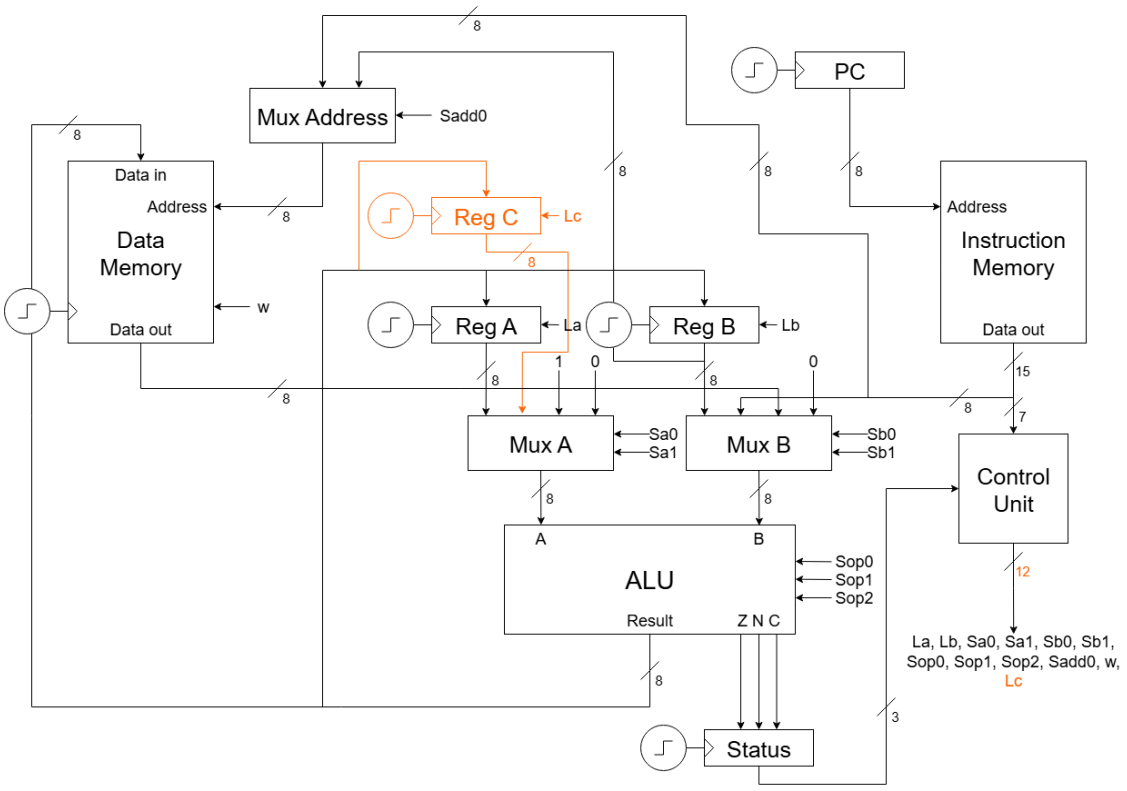

Figura 9: Diagrama Solución P6.b.

Para operar entre el nuevo registro C y el literal, debo poder seleccionar el registro C en el Mux A, pero en este caso no necesito agrandar el selector. Para poder guardar el resultado en el nuevo registro C debo conectar la salida de la ALU a la entrada de este y agregar la señal de carga `Lc`. Con eso, para la instrucción `ADD A,C` las señales de control son:

Instr.	Op.	La	Lb	Lc	Sa0,1	Sb0,1	Sop0,1,2	Sadd0	W
ADD	C, Lit	0	0	1	C	Lit	ADD	-	0

(c) Agregue la instrucción `NEG B,B`, la cual calcula el complemento de dos del registro `B` y lo almacena en el registro `B`.

Solución:

Para esta instrucción no necesito hacer cambios en el diagrama, se puede realizar restándole `B` a 0. Las señales de control son:

Instr.	Op.	La	Lb	Sa0,1	Sb0,1	Sop0,1,2	Sadd0	W
NEG	B, B	0	1	ZERO	B	SUB	-	0

(d) Agregue la instrucción `NEG A,A`, la cual calcula el complemento de dos del registro `A` y lo

almacena en el registro `A`.

Solución:

Para esta instrucción no necesito hacer cambios en el diagrama, se puede realizar con dos instrucciones: `NOT A,A` y `ADD A,1`. Las señales de control son:

Instr.	Op.	La	Lb	Sa0,1	Sb0,1	Sop0,1,2	Sadd0	W
NOT	A, A	1	0	A	-	NOT	-	0
ADD	A, Lit	1	0	A	Lit	ADD	-	0