DCC

DEPARTAMENTO DE CIENCIA
DE LA COMPUTACIÓN

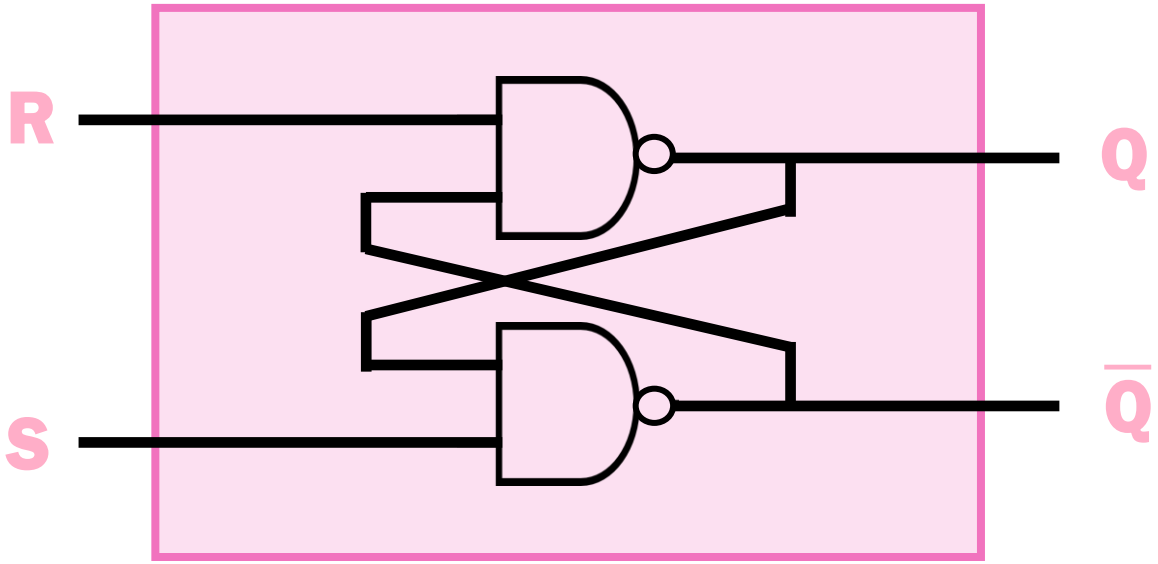
IIC2343 – Arquitectura de Computadores

Clase 5 – Procesador

Susana Figueroa

Latch RS

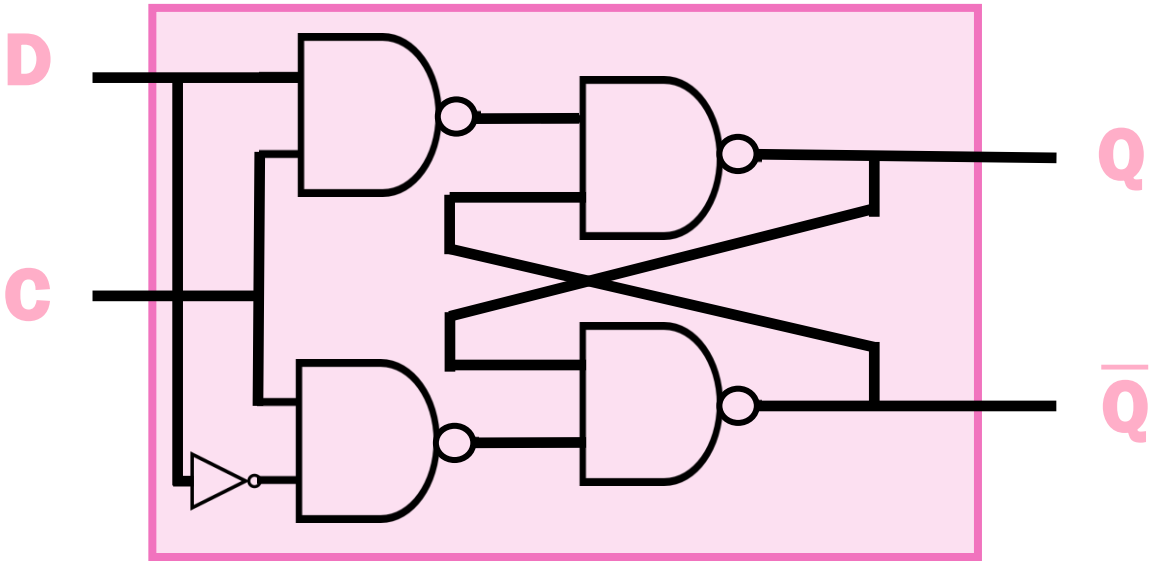
- Entonces este latch nos permite setear (Q=1) y resetear (Q=0) el valor de la memoria.
- En el caso de que R=S=1, Q se mantiene.



R	S	Q_t
0	0	-
0	1	1
1	0	0
1	1	Q_{t-1}

Latch D

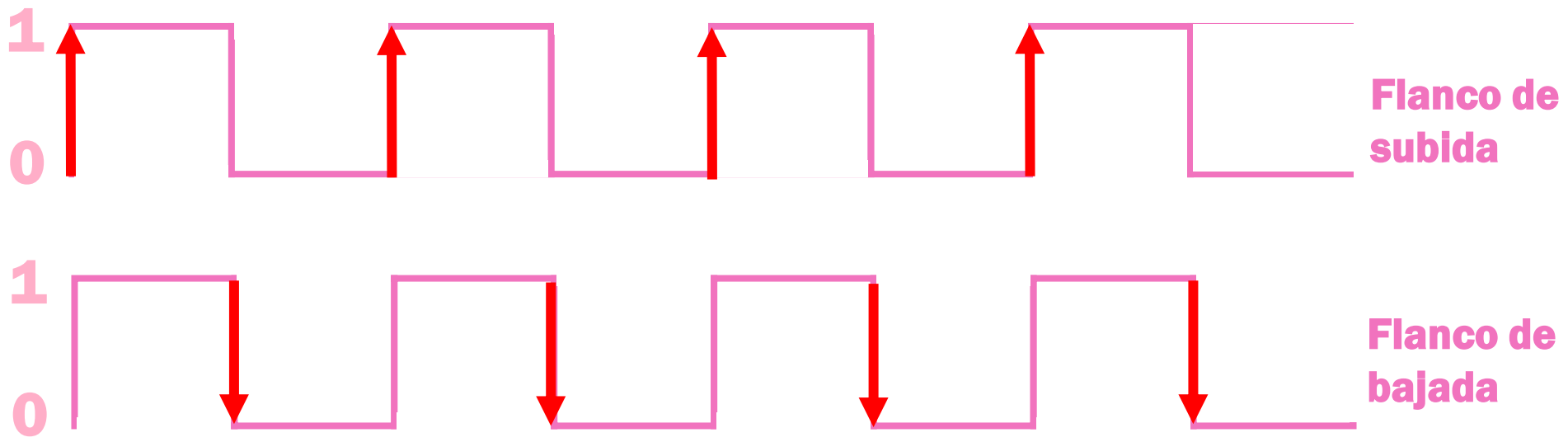
- Este latch se construye a partir de un latch RS, pero nos quita el estado invalido.
- C es una señal de control que habilita (o deshabilita) la carga de datos
- Ahora, cada vez que C=0, el estado anterior se conserva (independiente del valor de D).
- Cuando C=1, el valor que toma Q es el valor de D (Data).



C	D	Q_t
0	0	Q_{t-1}
0	1	Q_{t-1}
1	0	0
1	1	1

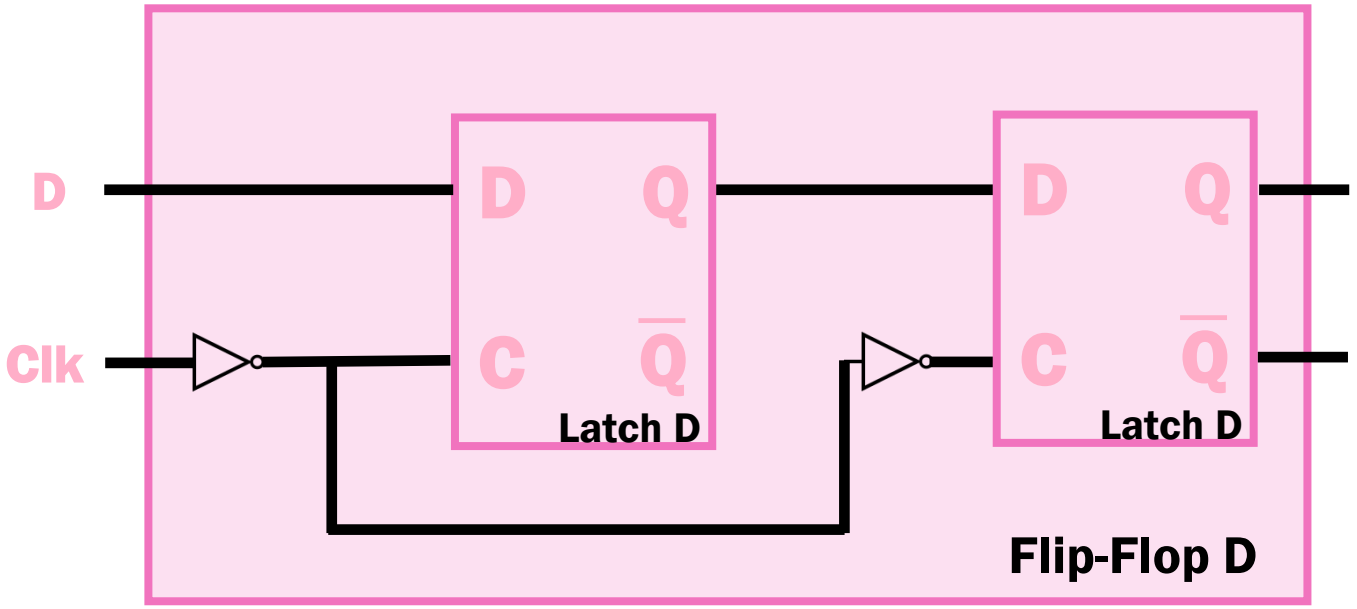
Clock

- El ***clock*** corresponde a una señal que oscila entre 0 a 1 en tiempo constante.
- Nos permitirá sincronizar los circuitos para que todos se encuentren en el estado esperado, en el mismo momento.
- Cada vez que el clock cambie de 0 a 1, se le llamará **flanco de subida** (el que utilizaremos).
- Cada vez que el clock cambie de 1 a 0, se le llamará **flanco de bajada**.



Flip-Flop D

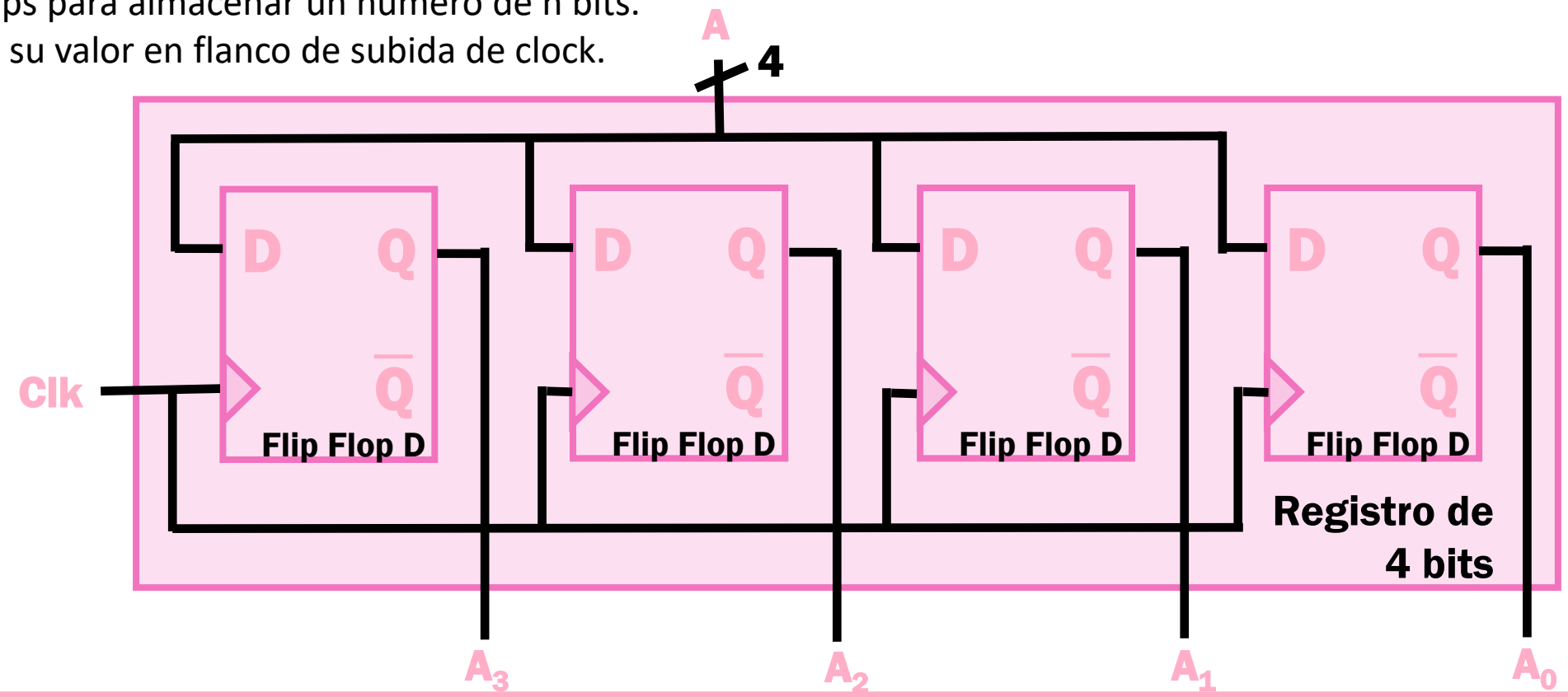
- Los ***flip-flops*** a diferencia de los latches, actualizan su valor solamente en flanco de subida de su señal de control (generalmente un clock).



Clk	D	Q_t
0/1/↓	X	Q_{t-1}
↑	0	0
↑	1	1

Registros

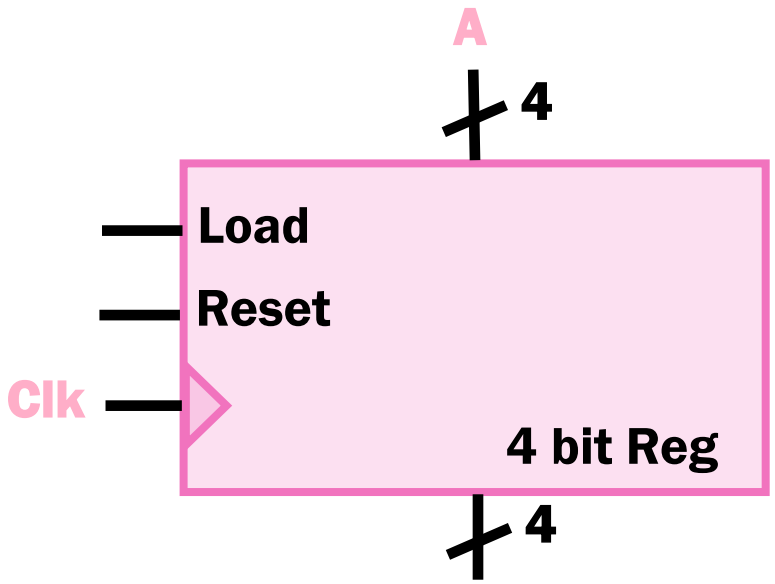
- El registro va a ser la componente que nos permita guardar números de n bits.
- Necesitamos n flip-flops para almacenar un número de n bits.
- Este registro actualiza su valor en flanco de subida de clock.



Registros

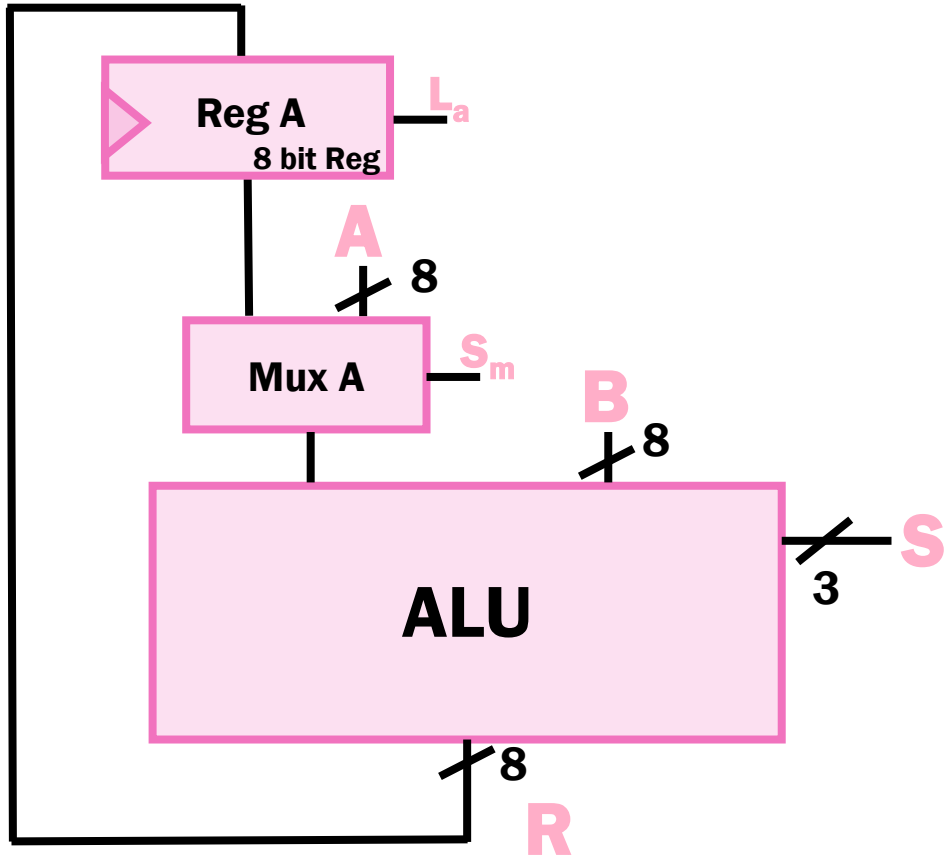
- Vamos a agregar dos señales de control:
  - Si **Load=1** y hay un **flanco de subida** de clock. Entonces el registro guarda el valor A. Para cualquier otro caso, el registro mantiene el valor anterior.
  - Si **Reset=1**. En algunas implementaciones también debe haber flanco de subida de clock. El valor del registro vuelve a su valor por defecto, 0.

Clk	Load	Reset	A'
0/1/↓	X	0	A'_{t-1}
↑	0	0	A'_{t-1}
↑	1	0	A
X	X	1	0



ALU

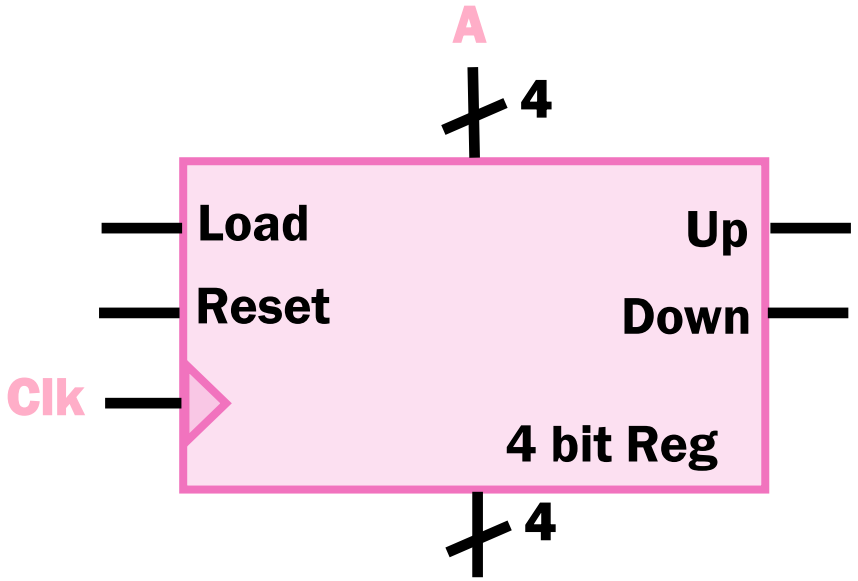
- Agregamos un registro A de 8 bits.
  - Su entrada va a ser el resultado de la ALU (**R**).
  - Su salida va conectada a un multiplexor (**Mux A**), que decide si entra a la ALU el valor almacenado en el registro (**Reg A**) o un input **A**.
  - Y va a tener una señal de control de carga $\mathbf{L_a}$. Cuando esta señal valga 1, al haber un flanco de subida de clock, el registro almacenará el valor que se encuentre en **R** en ese instante. Cuando esta señal es 0, el valor anterior se preserva.



Contador

- Modificando el registro se puede convertir en un contador.
- Se agregan dos señales extra: Up y Down.
  - Up: indica si hay que incrementar en uno el valor almacenado.
  - Down: indica si hay que decrementar en uno el valor almacenado.

Ya sabemos que todas estas componentes funcionan en flanco de subida de clock, desde ahora lo vamos a “obviar”.

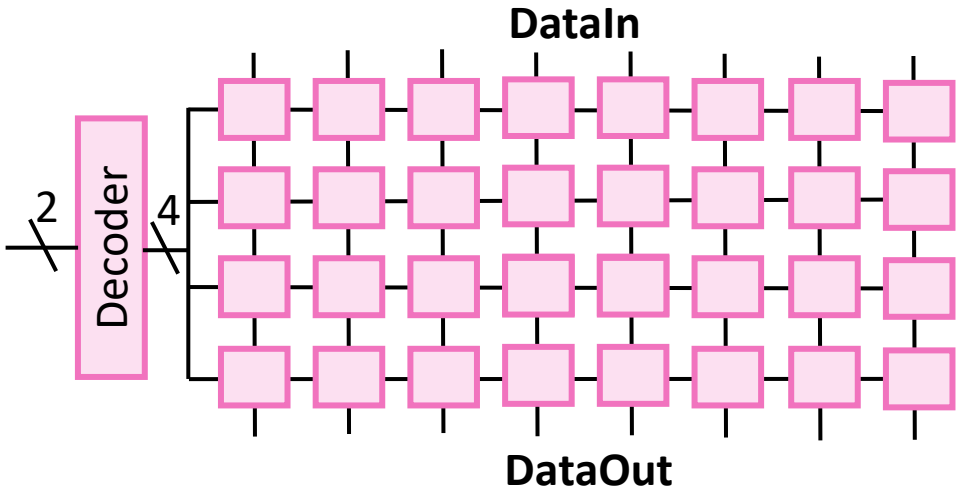


Memorias

- Con los registros logramos tener una unidad básica de memoria. Nos permiten almacenar una “palabra” de información y consultarla en el futuro.
- Pero queremos tener más memoria que tan solo un registro en nuestro computador, pero tener varios registros individuales es muy costoso… cuál es la solución intermedia a la que podemos llegar?

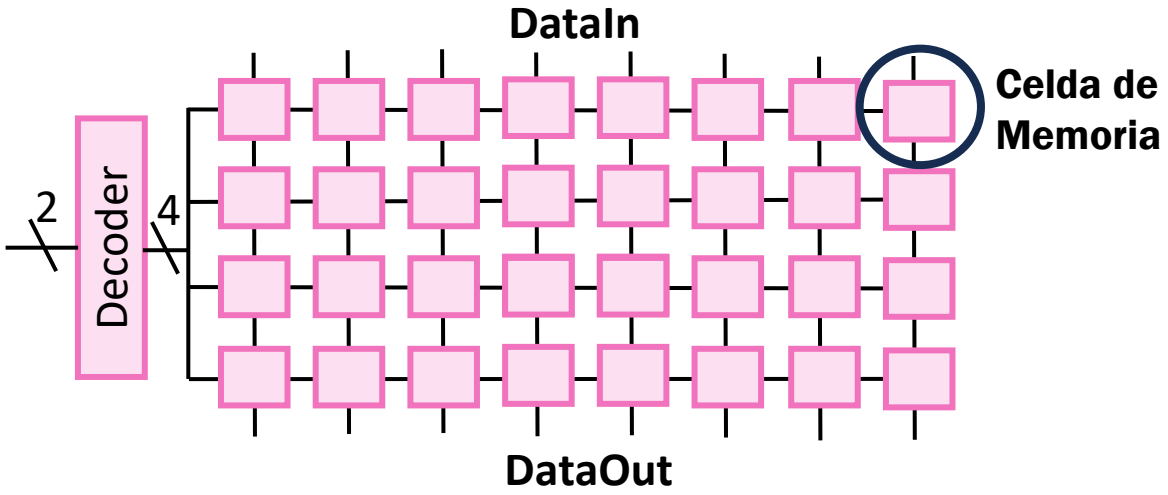
Memorias

- Con los registros logramos tener una unidad básica de memoria. Nos permiten almacenar una “palabra” de información y consultarla en el futuro.
- Pero queremos tener más memoria que tan solo un registro en nuestro computador, pero tener varios registros individuales es muy costoso… cuál es la solución intermedia a la que podemos llegar? En vez de tener registros “por separado” podemos hacer una grilla con ellos, de esta manera nos ahorramos cableo y *hardware.*
- Podemos pesar en que cada fila de la grilla corresponde a un registro (realmente no es así de simple, pero el detalle es innecesariamente muy complicado para el curso).



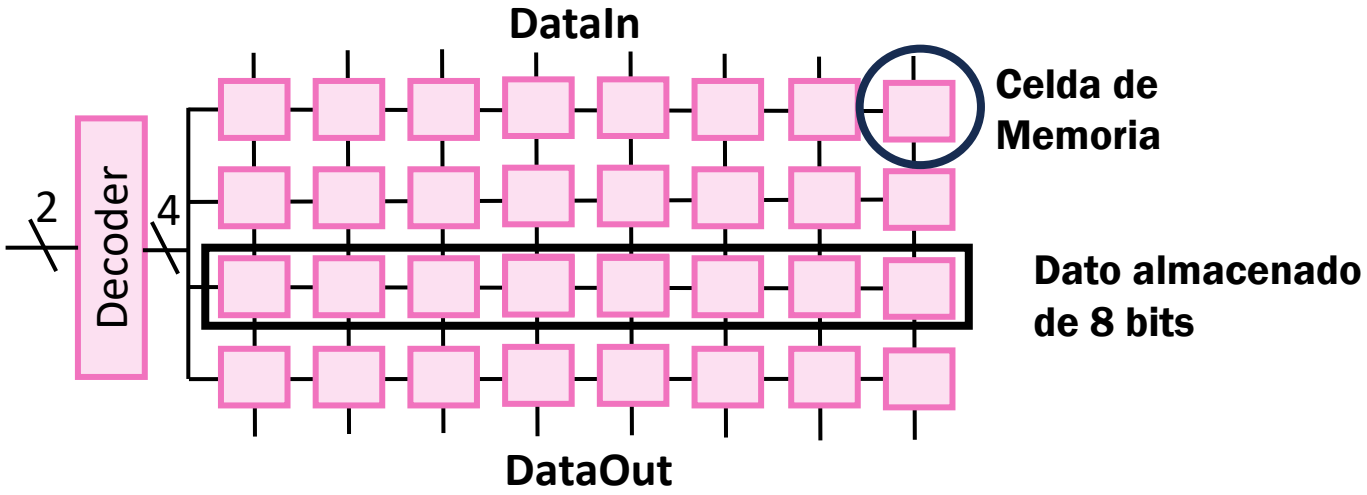
Memorias

- Con los registros logramos tener una unidad básica de memoria. Nos permiten almacenar una “palabra” de información y consultarla en el futuro.
- Pero queremos tener más memoria que tan solo un registro en nuestro computador, pero tener varios registros individuales es muy costoso… cuál es la solución intermedia a la que podemos llegar? En vez de tener registros “por separado” podemos hacer una grilla con ellos, de esta manera nos ahorramos cableo y *hardware.*
- Podemos pesar en que cada fila de la grilla corresponde a un registro (realmente no es así de simple, pero el detalle es innecesariamente muy complicado para el curso).



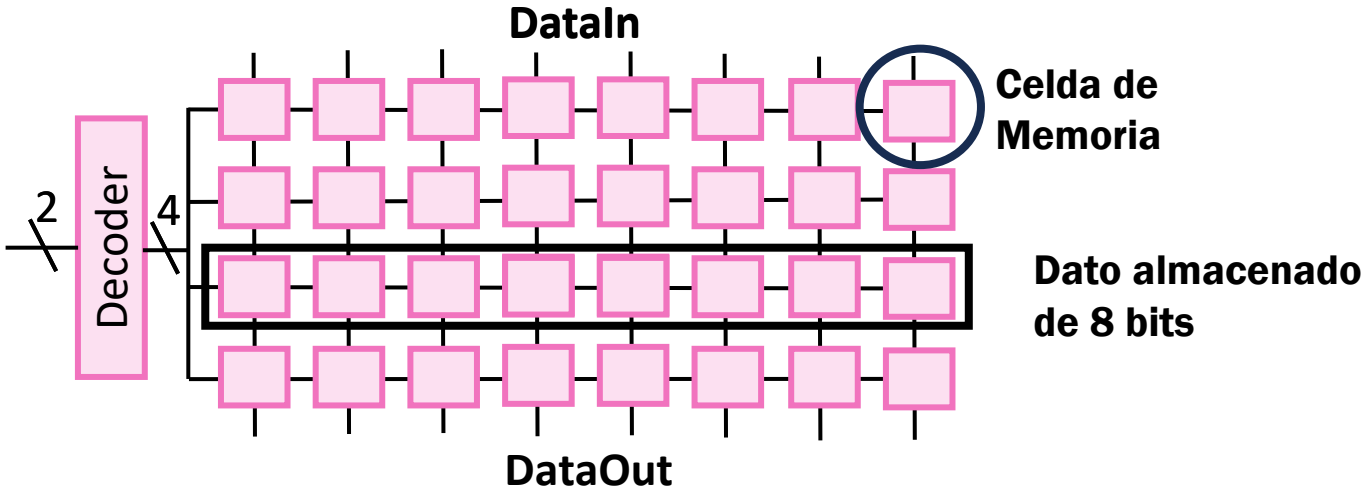
Memorias

- Con los registros logramos tener una unidad básica de memoria. Nos permiten almacenar una “palabra” de información y consultarla en el futuro.
- Pero queremos tener más memoria que tan solo un registro en nuestro computador, pero tener varios registros individuales es muy costoso… cuál es la solución intermedia a la que podemos llegar? En vez de tener registros “por separado” podemos hacer una grilla con ellos, de esta manera nos ahorramos cableo y *hardware.*
- Podemos pesar en que cada fila de la grilla corresponde a un registro (realmente no es así de simple, pero el detalle es innecesariamente muy complicado para el curso).



Memorias

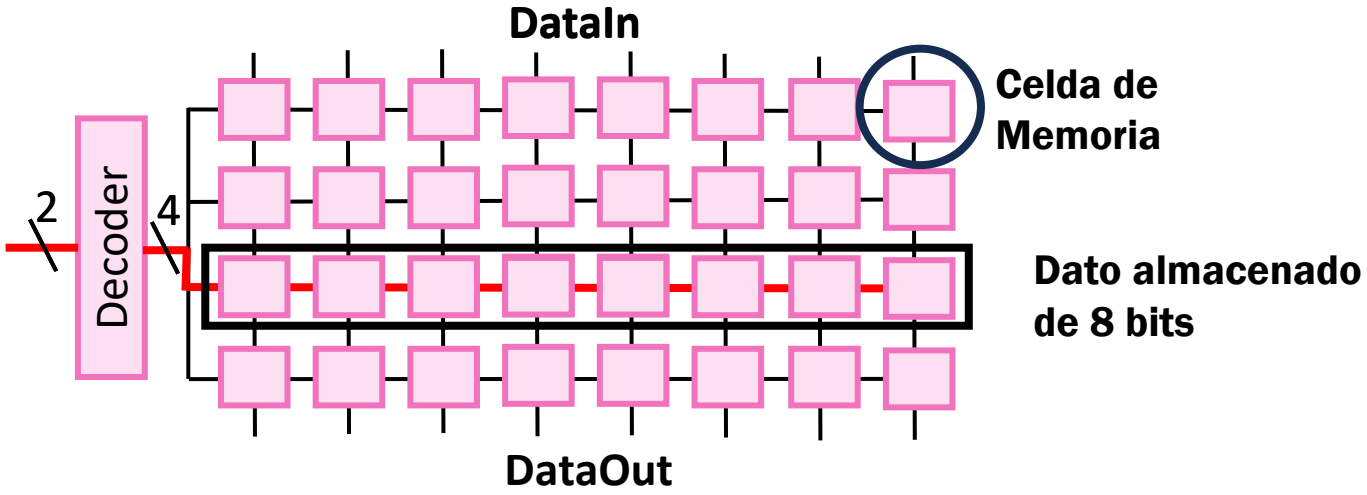
- Con los registros logramos tener una unidad básica de memoria. Nos permiten almacenar una “palabra” de información y consultarla en el futuro.
- Pero queremos tener más memoria que tan solo un registro en nuestro computador, pero tener varios registros individuales es muy costoso… cuál es la solución intermedia a la que podemos llegar? En vez de tener registros “por separado” podemos hacer una grilla con ellos, de esta manera nos ahorramos cableo y *hardware.*
- Podemos pesar en que cada fila de la grilla corresponde a un registro (realmente no es así de simple, pero el detalle es innecesariamente muy complicado para el curso).



En este caso vamos a poder almacenar 4 palabras de 1 byte. Cómo le indicamos a la grilla que palabra (o fila) es la que queremos leer/escribir?

Memorias

- Con los registros logramos tener una unidad básica de memoria. Nos permiten almacenar una “palabra” de información y consultarla en el futuro.
- Pero queremos tener más memoria que tan solo un registro en nuestro computador, pero tener varios registros individuales es muy costoso… cuál es la solución intermedia a la que podemos llegar? En vez de tener registros “por separado” podemos hacer una grilla con ellos, de esta manera nos ahorramos cableo y *hardware.*
- Podemos pesar en que cada fila de la grilla corresponde a un registro (realmente no es así de simple, pero el detalle es innecesariamente muy complicado para el curso).



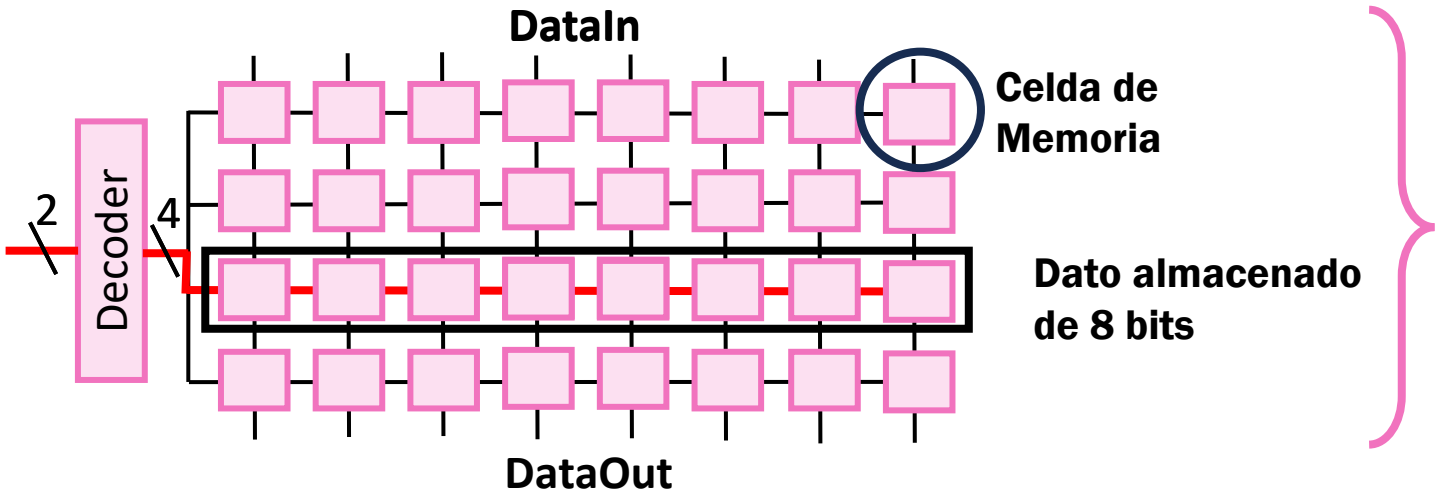
En este caso vamos a poder almacenar 4 palabras de 1 byte. Cómo le indicamos a la grilla que palabra (o fila) es la que queremos leer/escribir?

Para esto está el **decoder** (revisar circuito clase 2). A esta componente le vamos a indicar el número de la fila en la que está el dato que queremos y encenderá esa línea.

Memorias

- Al número de la línea en la que se encuentra el dato se le conocerá como **dirección de memoria**.
- Al proceso de acceso al valor a través de una dirección de memoria se le llamara **direccionamiento**. Este direccionamiento puede ser tanto para escritura como lectura.

Esta grilla es mejor representarla como una tabla.

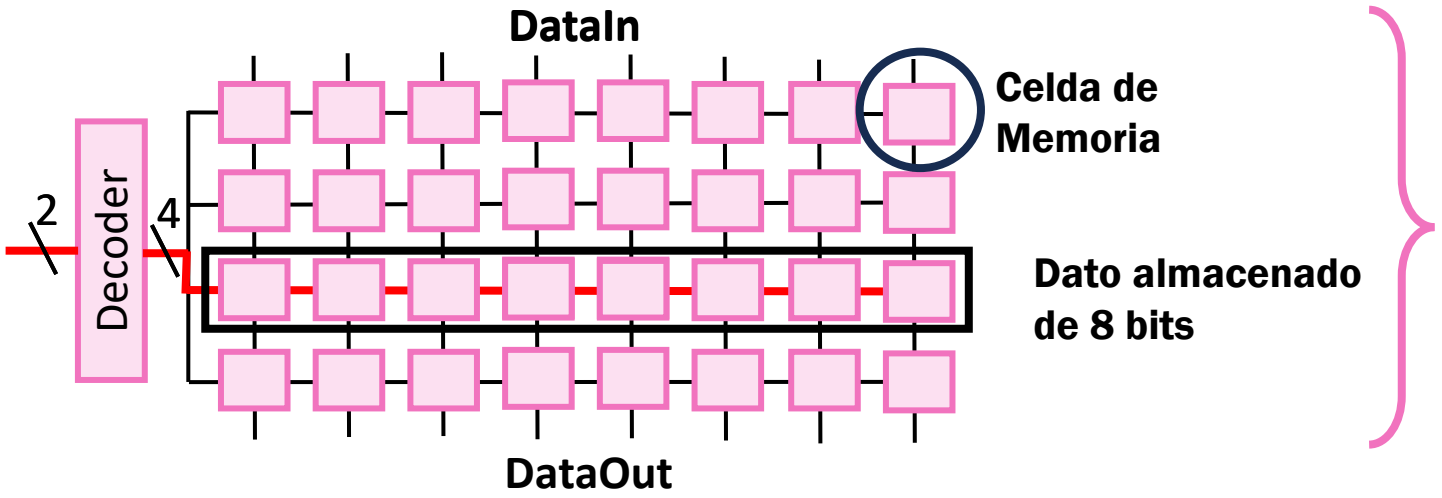


Dirección	Dato
00	Palabra0
01	Palabra1
10	Palabra2
11	Palabra3

Memorias

- Al número de la línea en la que se encuentra el dato se le conocerá como **dirección de memoria**.
- Al proceso de acceso al valor a través de una dirección de memoria se le llamara **direccionamiento**. Este direccionamiento puede ser tanto para escritura como lectura.

Esta grilla es mejor representarla como una tabla.



Ojo… Las direcciones no están almacenadas en memoria, solo los datos.

Dirección	Dato
00	Palabra0
01	Palabra1
10	Palabra2
11	Palabra3

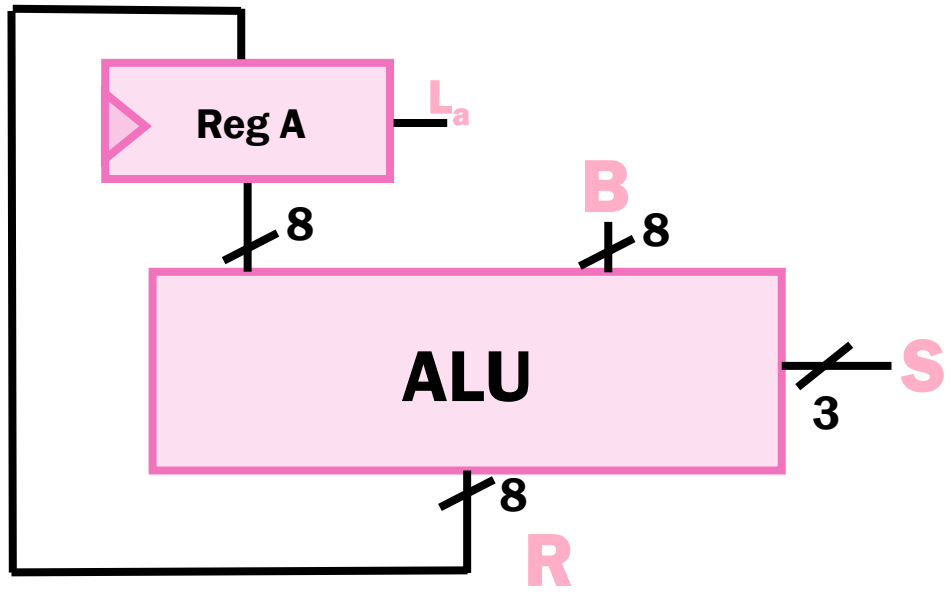
Memorias

- **Random Access Memory**: La memoria RAM consiste en una memoria que permite **leer** y **escribir** datos. Es de carácter **volátil**, por lo que si se corta su suministro eléctrico los datos se pierden. Pueden ser de tipo estático o dinámico. Las **SRAM**s (*Static* RAM) usan flip-flops como celda de memoria, mientras que las **DRAM**s (*Dinamic* RAM) usan un transistor con un condensador.
- **Read-Only Memory**: La ROM solo permite **leer** datos, estos no pueden ser modificados por nosotros. Este tipo de memoria es **no volátil**.
- **Hard Drive Disk**: Conocido por nosotros como disco duro. La forma en la que almacena los datos es a través de discos metálicos con un recubrimiento magnético. Los bits están codificados según la dirección de su campo magnético. Es de bajo costo, pero es bastante lento dado que para extraer los datos los discos deben girar hasta la posición en la que se encuentra el dato y después de esto leer/escribir.
- **Solid State Disk**: Mejor conocidas como discos sólidos, son un tipo de memoria flash. Usan descargas de eléctricas para impedir o admitir el paso de electrones a través del transistor.

A lo largo del curso, y el proyecto se usará principalmente dos tipos de memorias: **RAM** y **ROM**.

Construyendo el computador...

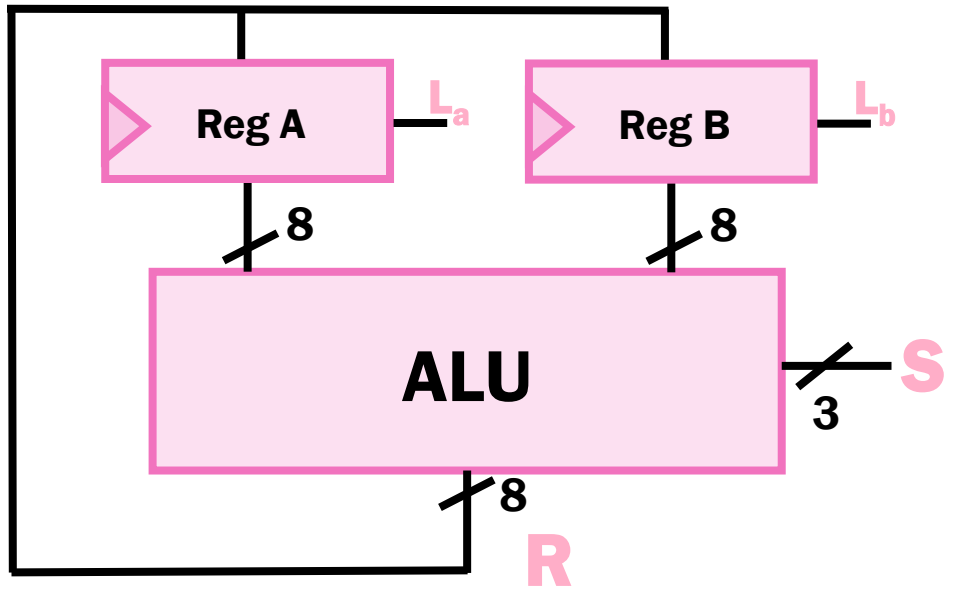
- Ahora que tenemos memorias, podemos empezar a construir nuestro computador básico.
- Partamos recordando lo que teníamos anteriormente: una ALU y un registro para A (sacamos el multiplexor porque vamos a poder inicializar el registro A con el valor que queramos).
- Ahora queremos agregar un registro para B.



Construyendo el computador...

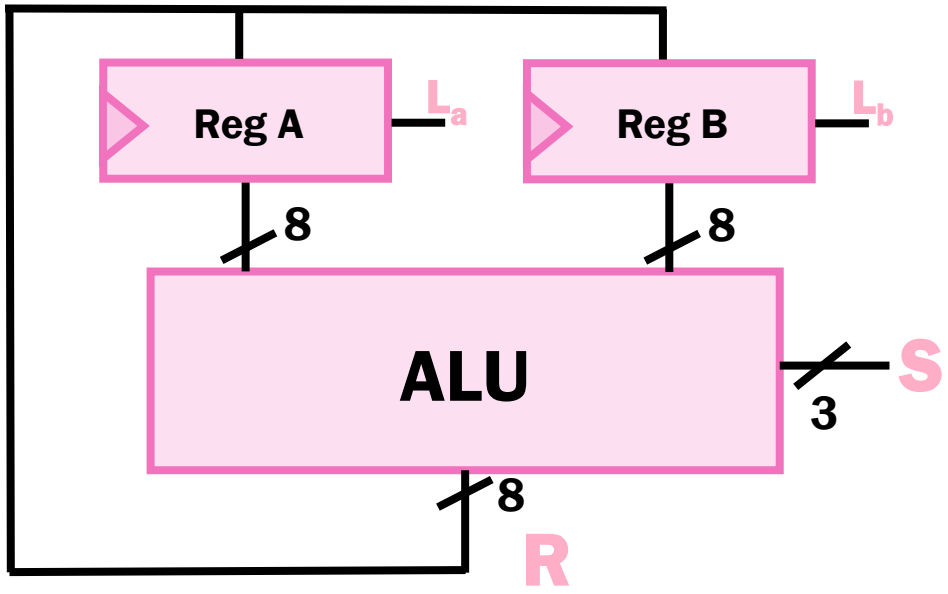
- Ahora que tenemos memorias, podemos empezar a construir nuestro computador básico.
- Partamos recordando lo que teníamos anteriormente: una ALU y un registro para A (sacamos el multiplexor porque vamos a poder inicializar el registro A con el valor que queramos).
- Ahora queremos agregar un registro para B. También tiene su señal de control L_b que permite cargar el registro.

- El resultado de la ALU ahora se puede guardar tanto en Reg A como en Reg B.



Construyendo el computador...

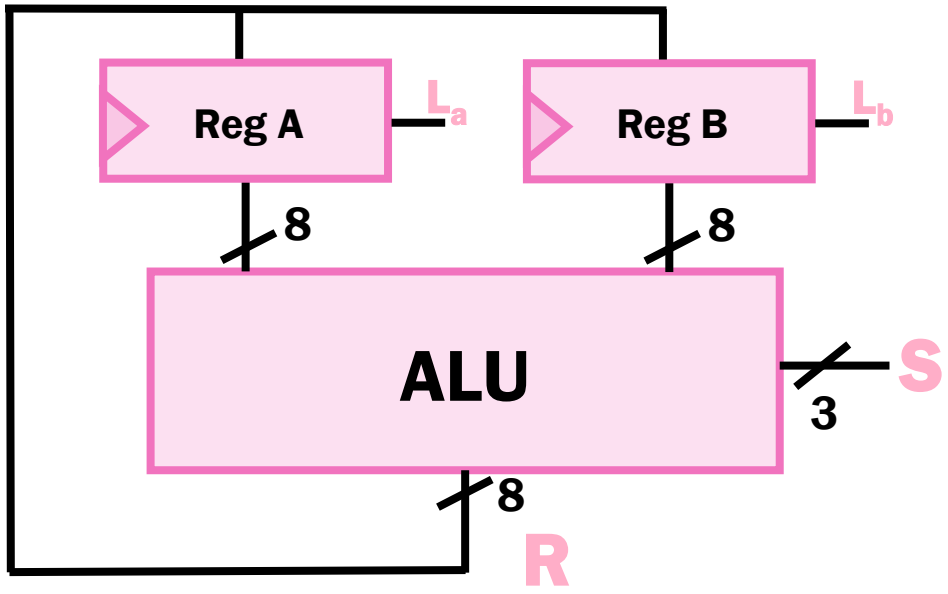
- Definamos ahora a través de una tabla las operaciones que puede realizar nuestro precario computador y los valores que deben de tomar las señales de control.



L_a	L_b	S_2	S_1	S_0	Operación
1	0	0	0	0	A = A + B
0	1	0	0	0	B = A + B
1	0	0	0	1	A = A – B
0	1	0	0	1	B = A – B
1	0	0	1	0	A = A and B
0	1	0	1	0	B = A and B
1	0	0	1	1	A = A or B
0	1	0	1	1	B = A or B
1	0	1	0	0	A = not A
0	1	1	0	0	B = not A
1	0	1	0	1	A = A xor B
0	1	1	0	1	B = A xor B
1	0	1	1	0	A = SHL A
0	1	1	1	0	B = SHL A
1	0	1	1	1	A = SHR A
0	1	1	1	1	B = SHR A

Construyendo el computador...

- Definamos ahora a través de una tabla las operaciones que puede realizar nuestro precario computador y los valores que deben de tomar las señales de control.



Cada palabra de control (secuencia de señales de control) representa una **instrucción**.

Una secuencia de instrucciones es un **programa**

L_a	L_b	S_2	S_1	S_0	Operación
1	0	0	0	0	A = A + B
0	1	0	0	0	B = A + B
1	0	0	0	1	A = A – B
0	1	0	0	1	B = A – B
1	0	0	1	0	A = A and B
0	1	0	1	0	B = A and B
1	0	0	1	1	A = A or B
0	1	0	1	1	B = A or B
1	0	1	0	0	A = not A
0	1	1	0	0	B = not A
1	0	1	0	1	A = A xor B
0	1	1	0	1	B = A xor B
1	0	1	1	0	A = SHL A
0	1	1	1	0	B = SHL A
1	0	1	1	1	A = SHR A
0	1	1	1	1	B = SHR A

Construyendo el computador...

L_a	L_b	S_2	S_1	S_0	A	B	Operación
0	0	0	0	0	1	2	-
1	0	0	0	0			
1	0	0	0	1			
0	1	0	1	1			
1	0	0	1	0			
0	1	1	1	0			
1	0	0	0	0			

- Supongamos que tenemos el siguiente programa. Cuál van a ser los valores de los registros A y B en cada paso? A qué operación corresponde cada instrucción?

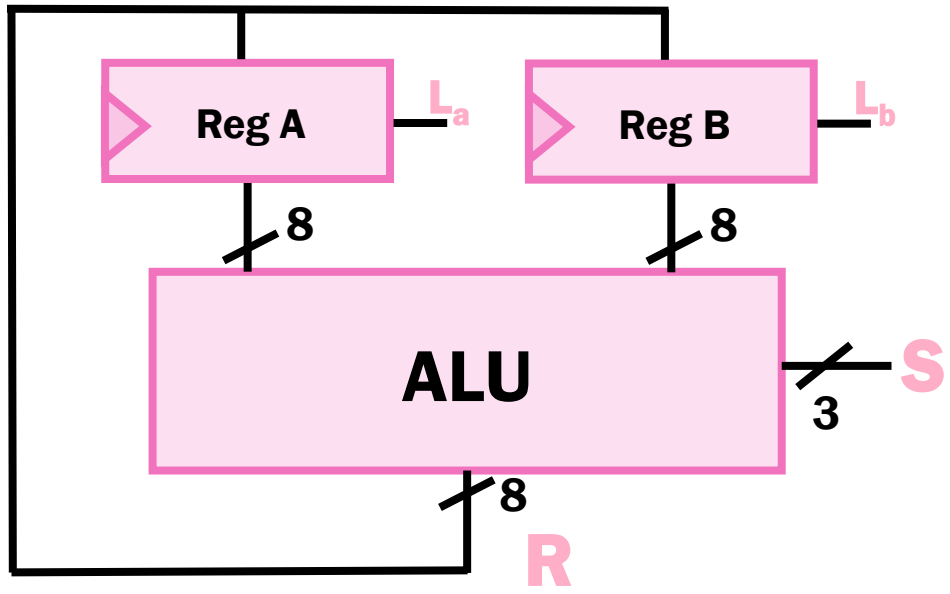
Construyendo el computador...

L_a	L_b	S_2	S_1	S_0	A	B	Operación
0	0	0	0	0	1	2	-
1	0	0	0	0	3	2	A = A + B
1	0	0	0	1	1	2	A = A - B
0	1	0	1	1	1	3	B = A or B
1	0	0	1	0	1	3	A = A and B
0	1	1	1	0	1	2	B = SHL A
1	0	0	0	0	3	2	A = A + B

- Supongamos que tenemos el siguiente programa. Cuál van a ser los valores de los registros A y B en cada paso? A qué operación corresponde cada instrucción?

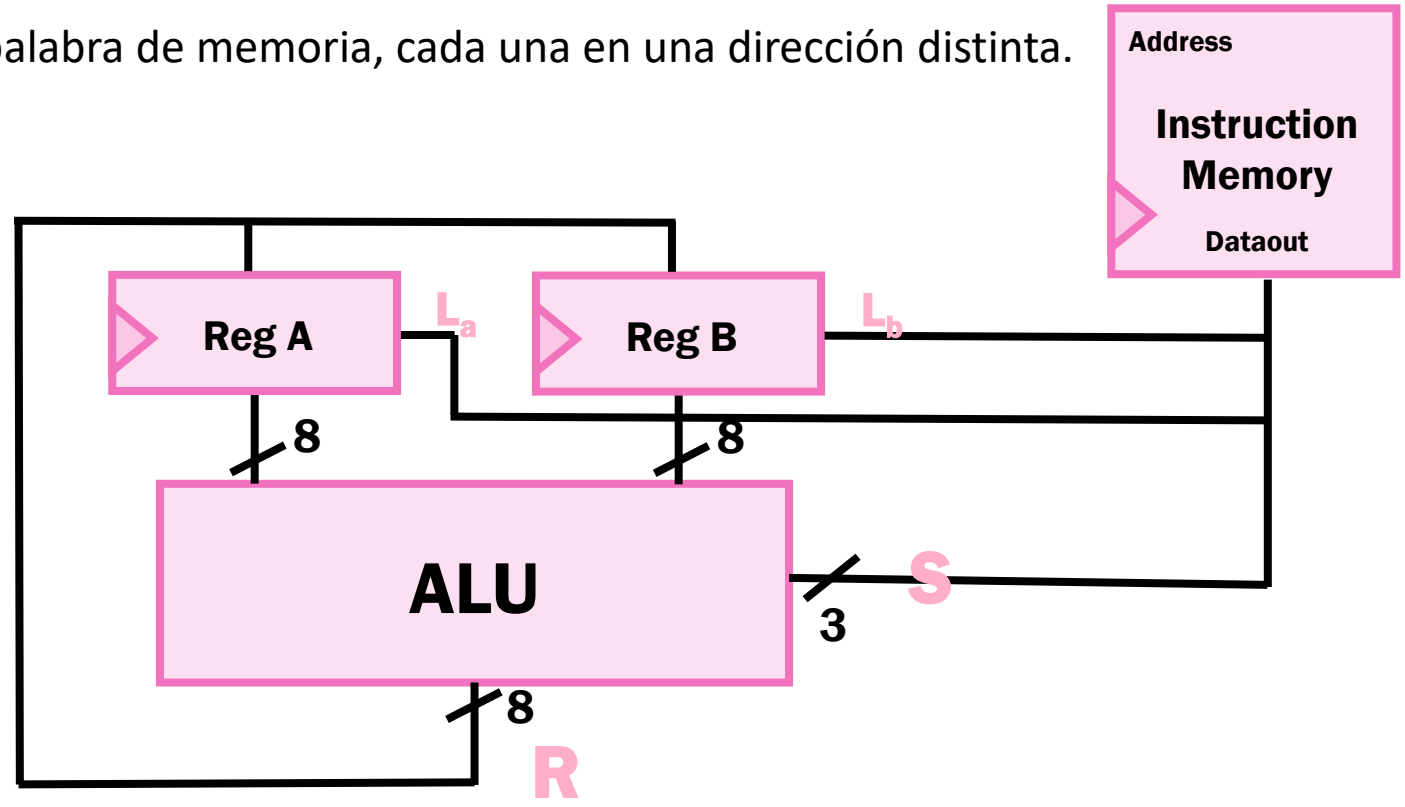
Construyendo el computador...

- Para que nuestro computador sea programable debe ser capaz de ejecutar una secuencia de instrucciones (como el programa de la diapo anterior).
- Pero cómo vamos a pasarle las instrucciones al computador y en dónde las vamos a guardar?



Construyendo el computador... Memoria de instrucciones

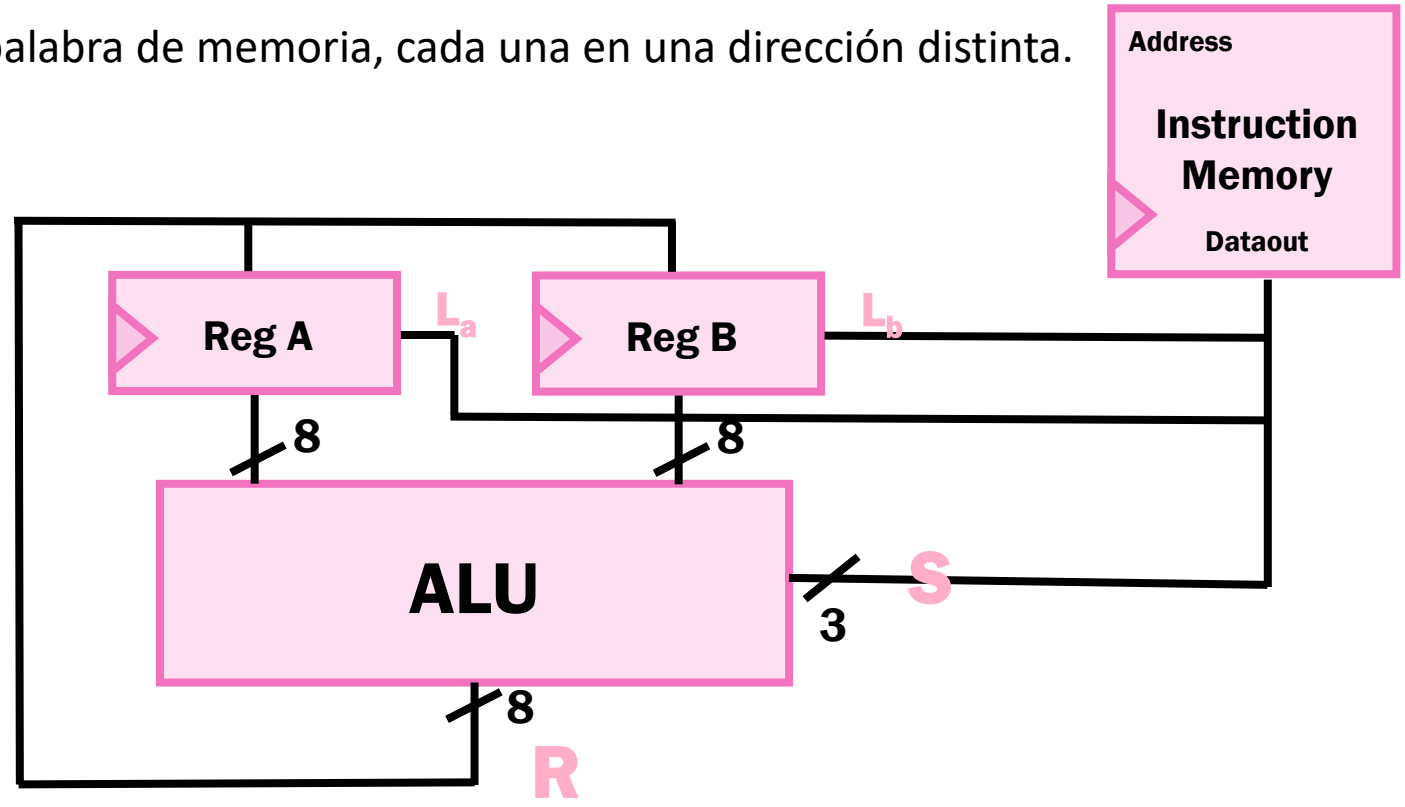
- Como las instrucciones del programa no se modificarán, podemos almacenarlas todas en una ROM llamada ***instruction memory***.
- Cada instrucción será guardada como una palabra de memoria, cada una en una dirección distinta.



Construyendo el computador... Memoria de instrucciones

- Como las instrucciones del programa no se modificarán, podemos almacenarlas todas en una ROM llamada ***instruction memory***.
- Cada instrucción será guardada como una palabra de memoria, cada una en una dirección distinta.

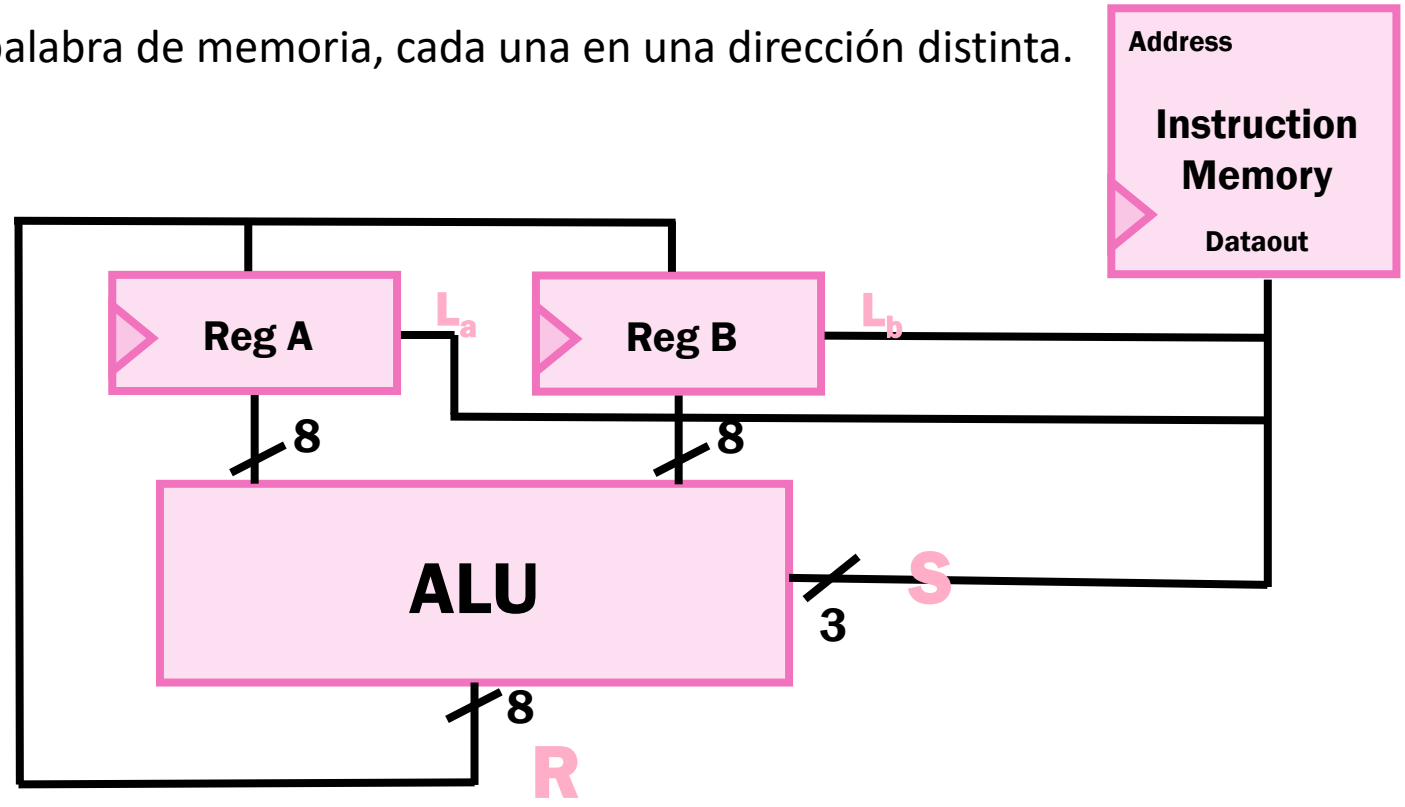
Como hacemos para "ejecutar" una instrucción tras otra?



Construyendo el computador... Memoria de instrucciones

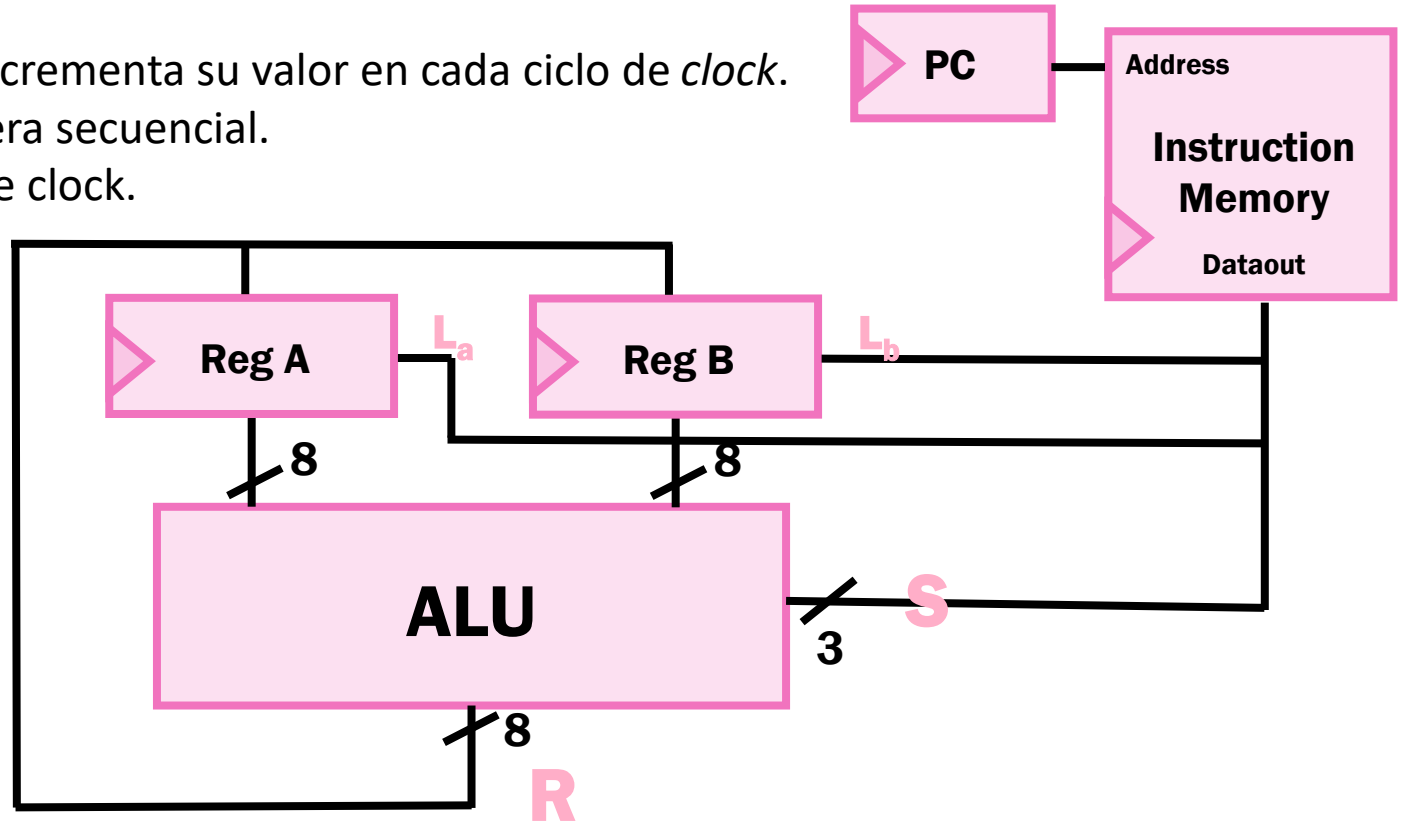
- Como las instrucciones del programa no se modificarán, podemos almacenarlas todas en una ROM llamada ***instruction memory***.
- Cada instrucción será guardada como una palabra de memoria, cada una en una dirección distinta.

Como hacemos para "ejecutar" una instrucción tras otra? Vamos a tener un registro contador conectado al *address* de la memoria de instrucciones.
La dirección a la que apunta el contador será la que de valor a las señales de control.



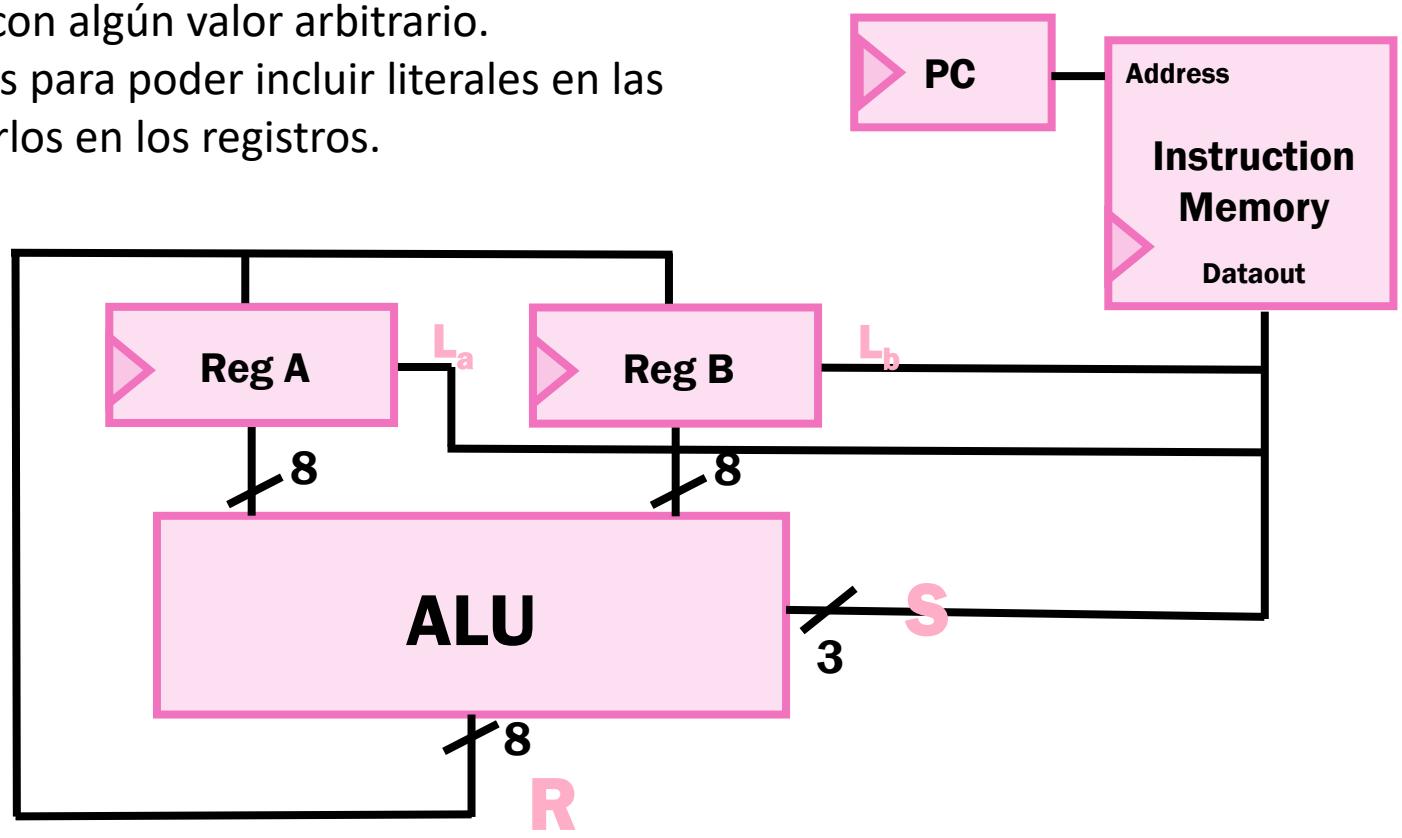
Construyendo el computador... Program counter

- El *program counter* (PC) es un registro contador que apunta a la siguiente dirección de la instrucción guardada en la ROM.
- La señal de *up* debe estar siempre activa, incrementa su valor en cada ciclo de *clock*.
- Nos permite ejecutar un programa de manera secuencial.
- Cada instrucción se va a demorar un ciclo de clock.



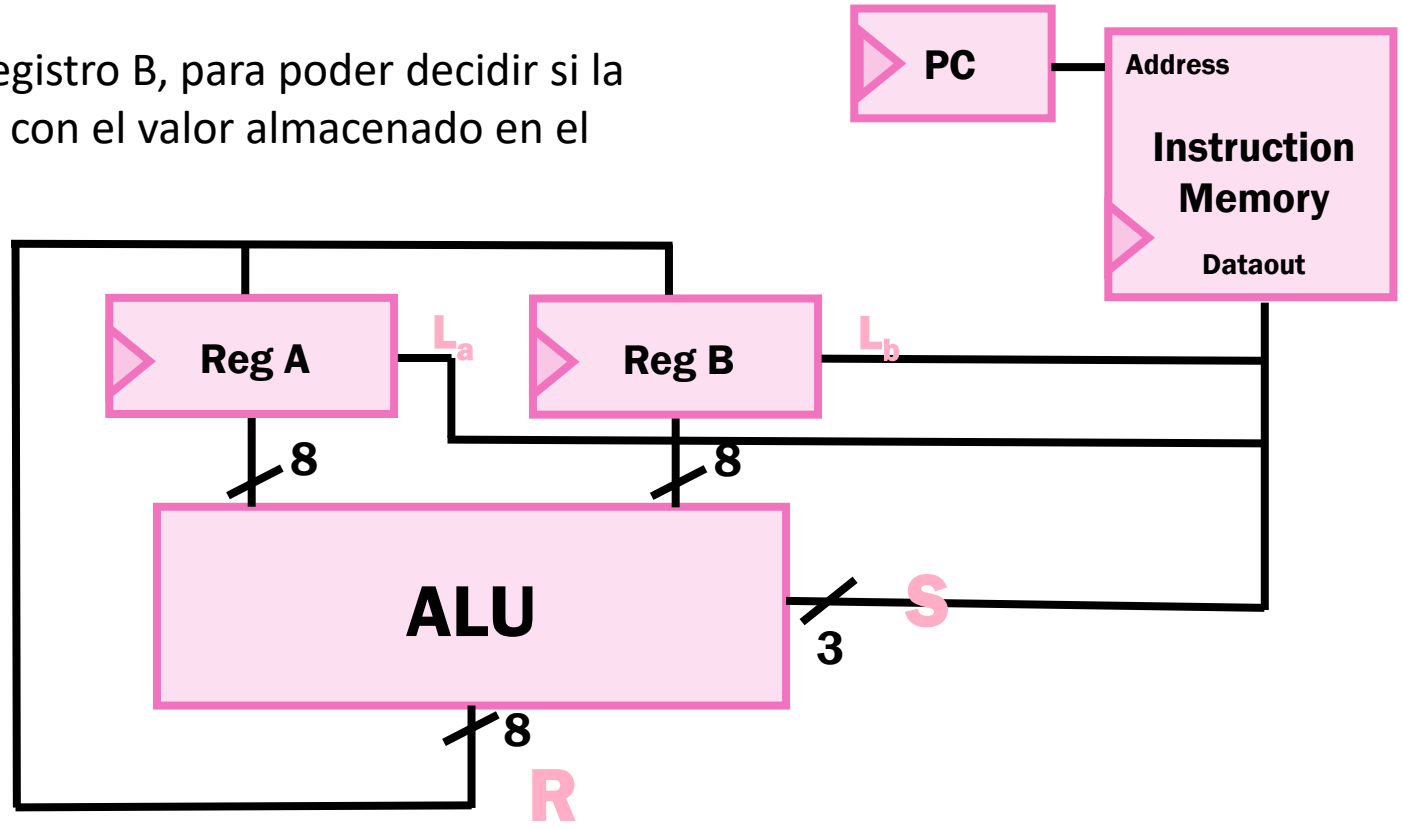
Construyendo el computador...

- Los programas que podemos hacer hasta ahora son muy limitados.
- Todavía ni podemos inicializar los registros con algún valor arbitrario.
- La extensión que queremos realizar ahora es para poder incluir literales en las instrucciones, para operar con ellos o cargarlos en los registros.



Construyendo el computador... Literales

- Los literales van a venir “dentro” de la instrucción y de alguna forma debemos pasárselos a la ALU.
- Vamos a agregar un Mux entre la ALU y el registro B, para poder decidir si la operación que se realice en la ALU debe ser con el valor almacenado en el registro B o un literal.

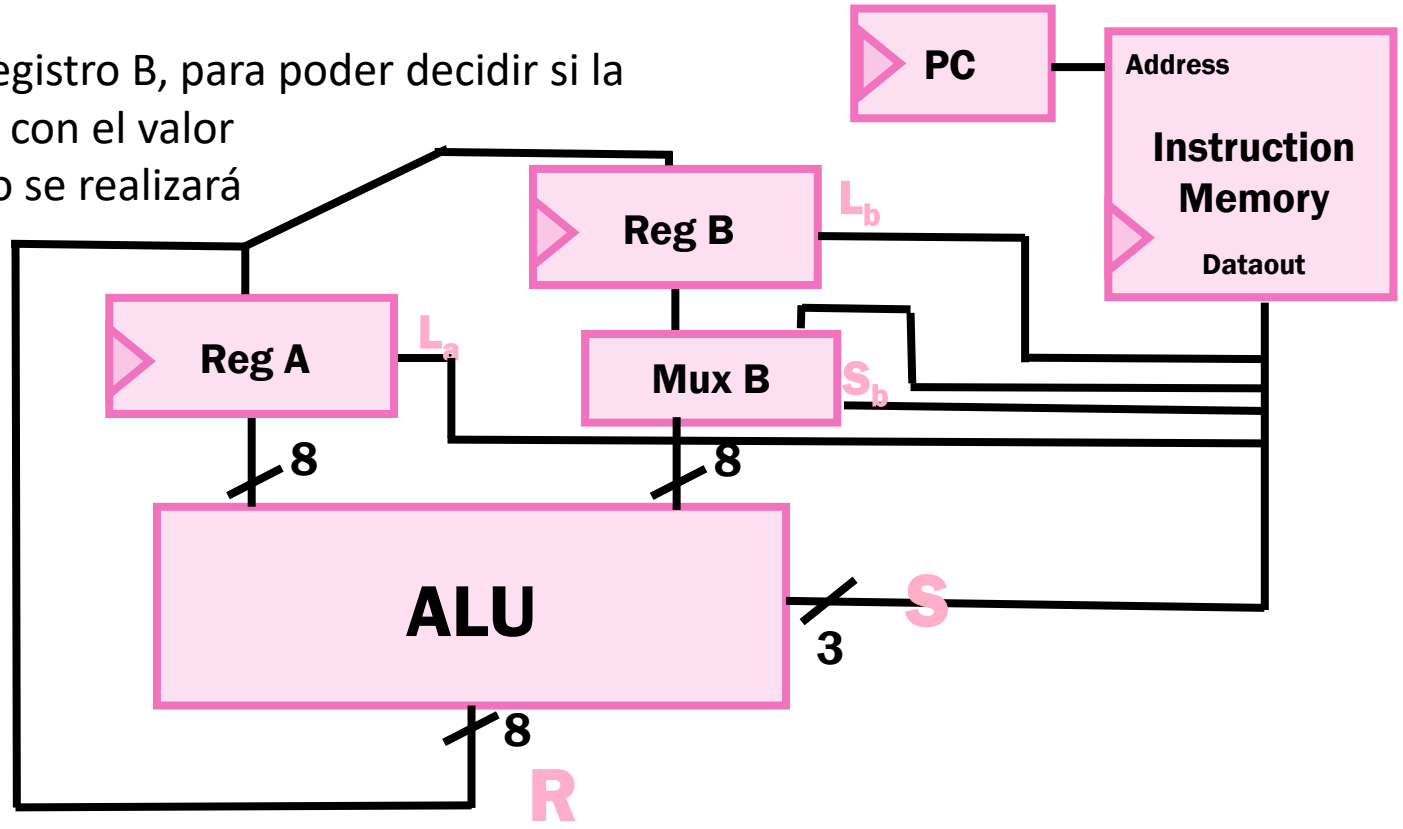


Construyendo el computador... Literales

- Los literales van a venir “dentro” de la instrucción y de alguna forma debemos pasárselos a la ALU.
- Vamos a agregar un Mux entre la ALU y el registro B, para poder decidir si la operación que se realice en la ALU debe ser con el valor almacenado en el registro B o un literal, esto se realizará a través de la señal de control del multiplexor S_b.

Ahora se agregan instrucciones tipo:

- A = A + 5
- B = A and 8



DCC

DEPARTAMENTO DE CIENCIA
DE LA COMPUTACIÓN

IIC2343 – Arquitectura de Computadores

Clase 5 – Procesador

Susana Figueroa