INTRODUCCIÓN A LOS MICROPROCESADORES

Fundamentos de microprocesadores

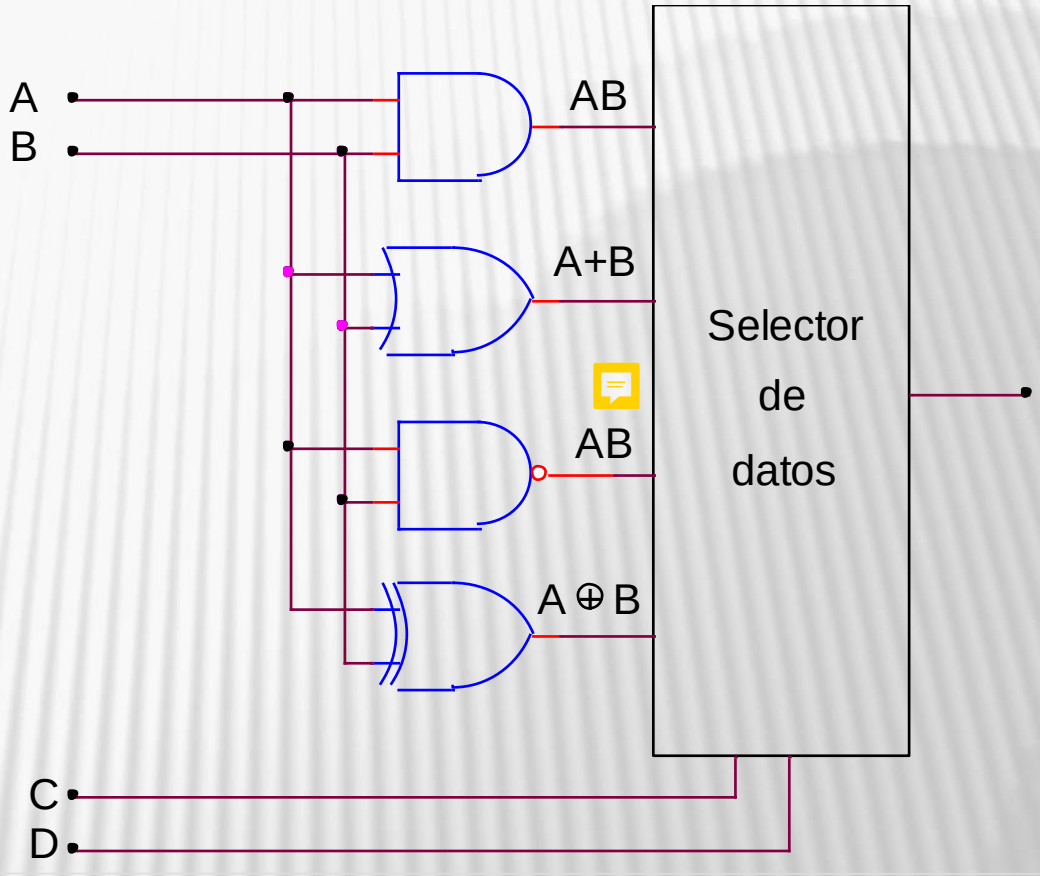
Fundamentos de microprocesadores*

- Un microprocesador es un circuito integrado que está compuesto, podríamos decir, de muchos circuitos integrados más simples. Dicho en otras palabras, un microprocesador tiene muchas ANDs, NANDs, ORs, NOTs, flip flops, contadores, comparadores, registros de corrimiento (SHR's), etc.

- Todos estos elementos están arreglados de tal forma que en un momento dado podemos usar el microprocesador para realizar operaciones distintas, mediante un *programa*.

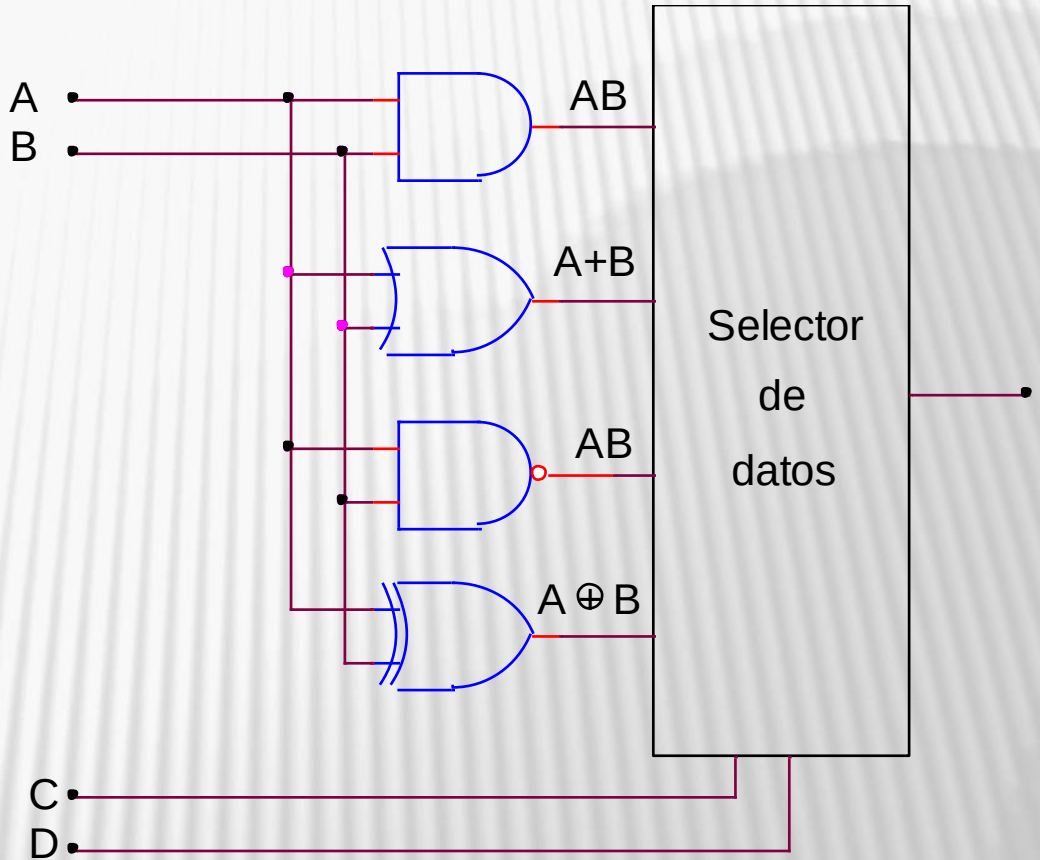
* Texto y ejemplo basado en el libro de *Introducción a los microprocesadores.* Juan Lanzagorta Ibarra. Segunda Edición, ITESO. 1982.

La figura muestra un sistema digital que sirve para realizar cuatro funciones lógicas. Se puede usar como AND, OR, NAND o como OR exclusiva (XOR). La salida dependerá directamente de las entradas A y B pero también de las señales de control, con las que se le van a dar "instrucciones" para que realice la función deseada.



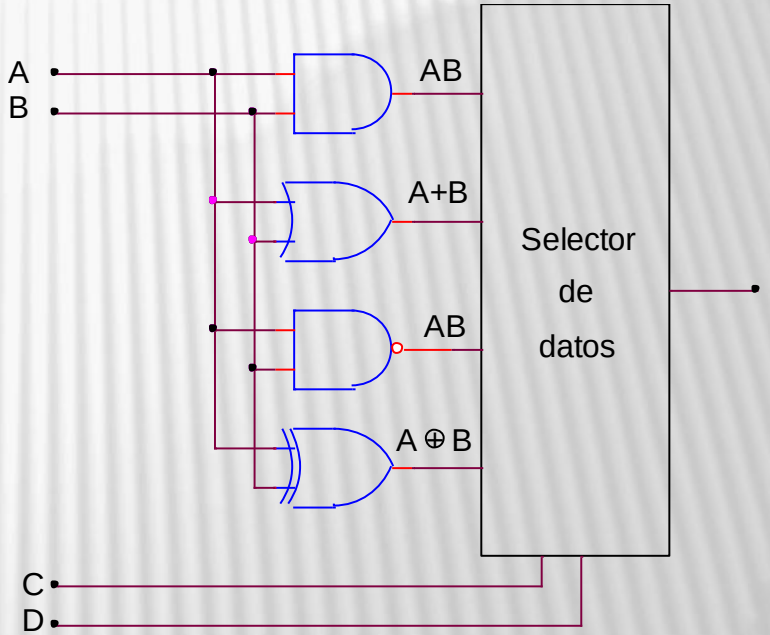
En un momento dado será equivalente a una NAND, en otro a una OR, etc. Dependiendo de la "instrucción" que se le dé, será la operación que haga.

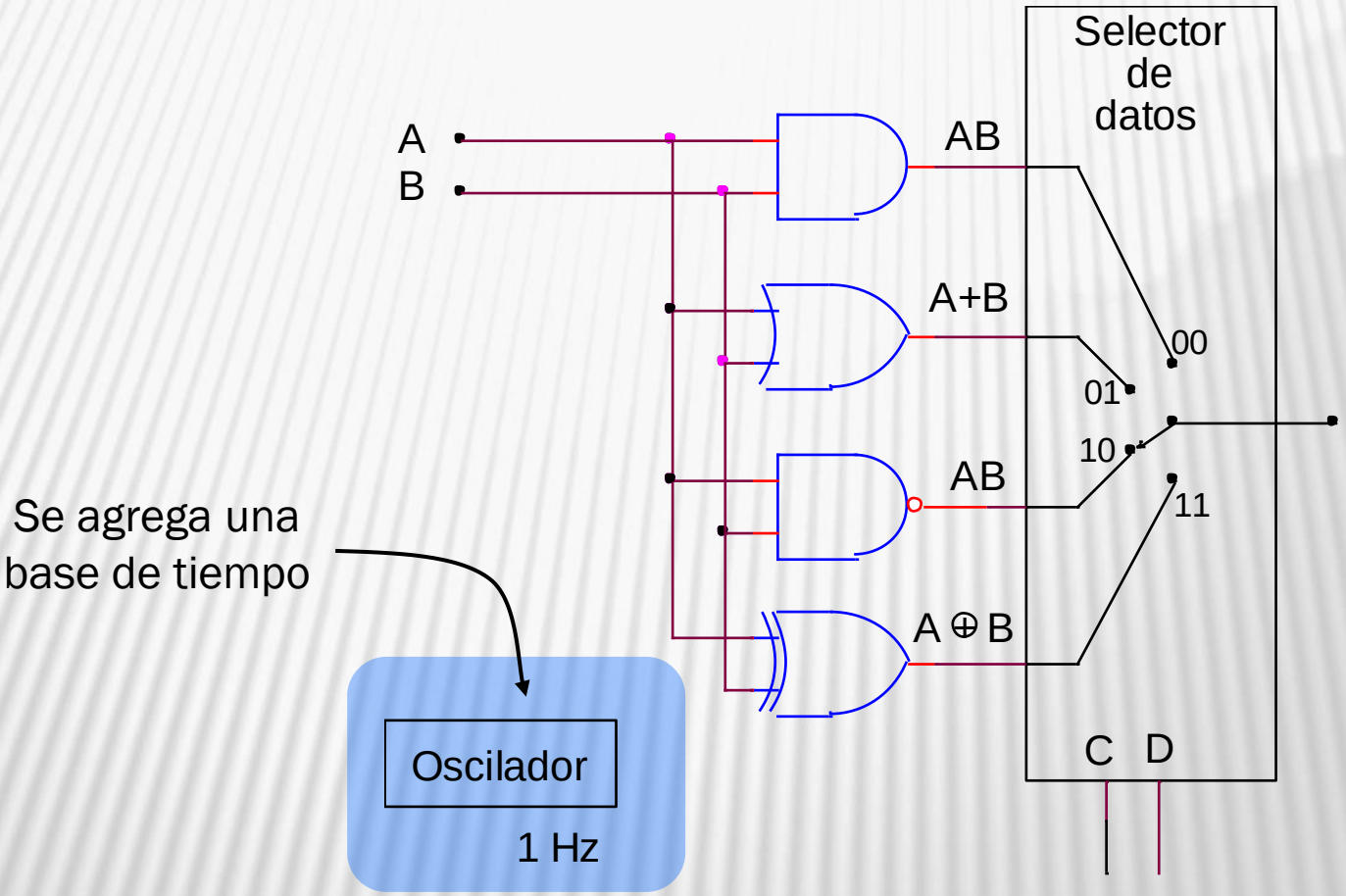
Por supuesto que pueden implementarse las cuatro funciones diferentes del circuito, pero si sólo se tiene una salida, no podrá realizarse más de una instrucción a la vez.

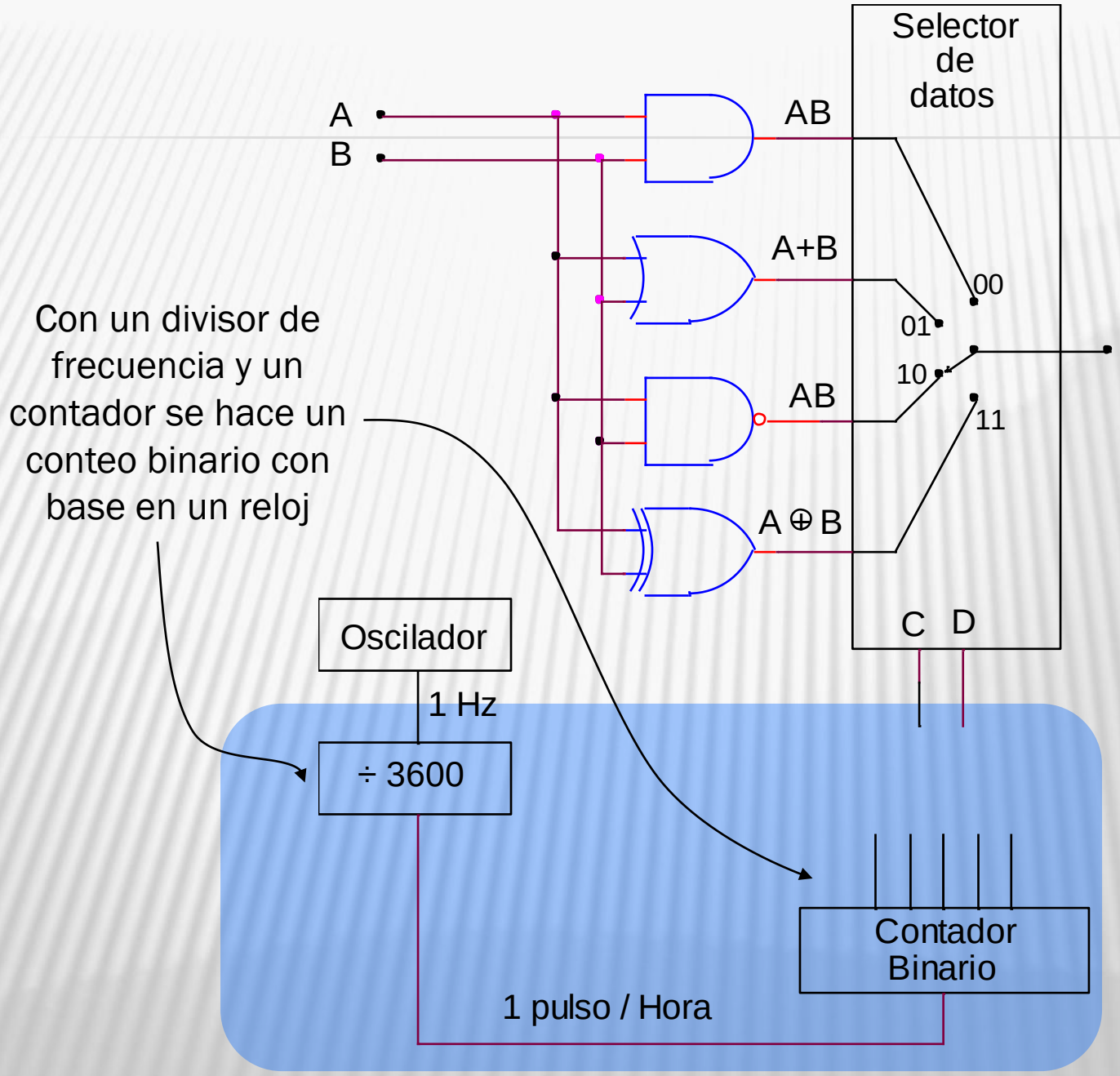


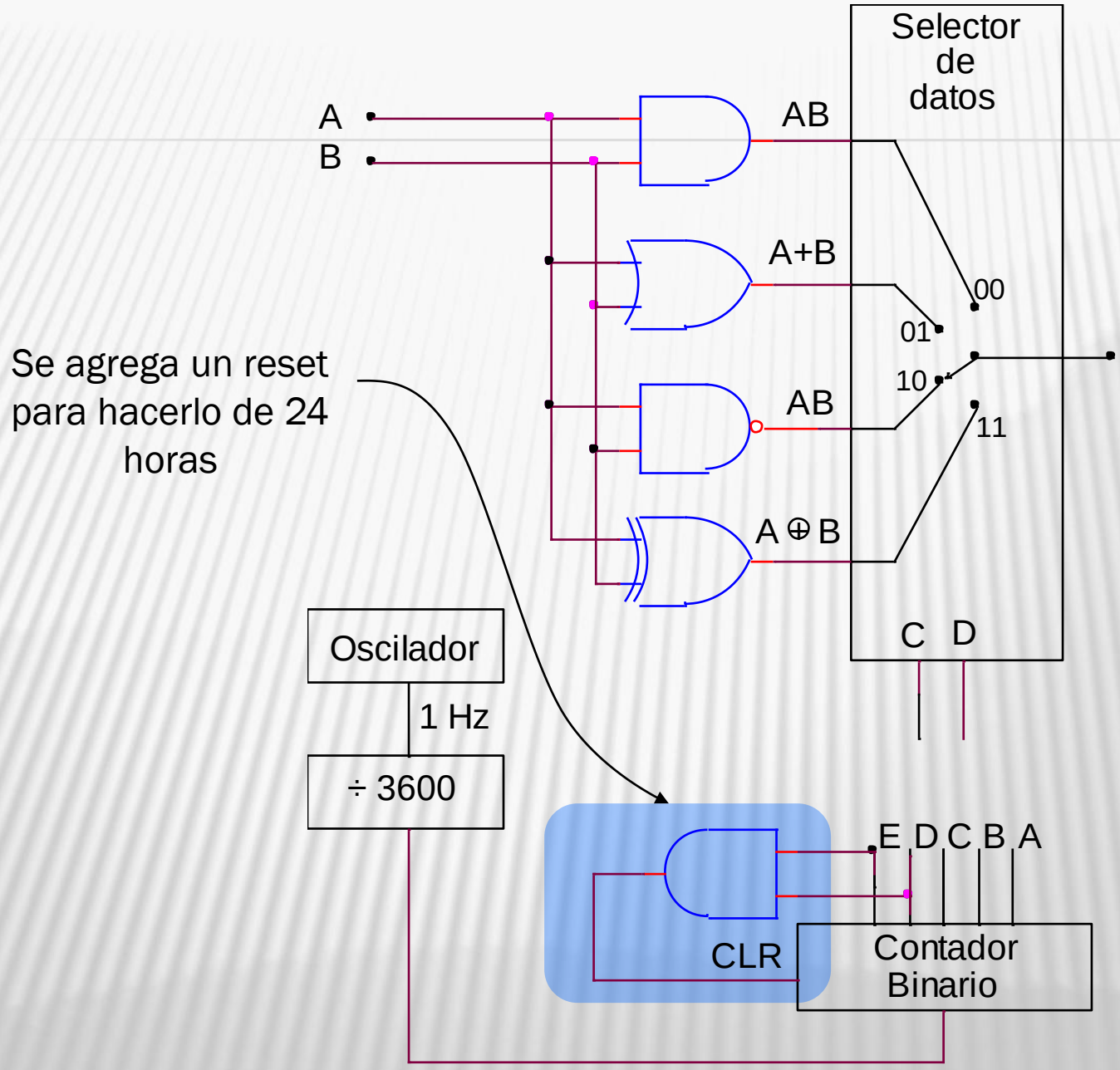
Supóngase que se usa este “procesador” para controlar una alarma que debe funcionar de la siguiente manera:

- De las 8 am a las 11 am la alarma debe sonar solo si las señales A y B están en uno (AND).
- De las 11 am a las 3 pm la alarma debe sonar si se prende cualquiera de las señales (OR).
- De las 3 pm a las 11 pm la alarma debe oírse cuando cualquiera de las dos señales este en cero (NAND).
- De las 11 pm a las 8 am la alarma debe escucharse si las dos señales son diferentes entre si (XOR).

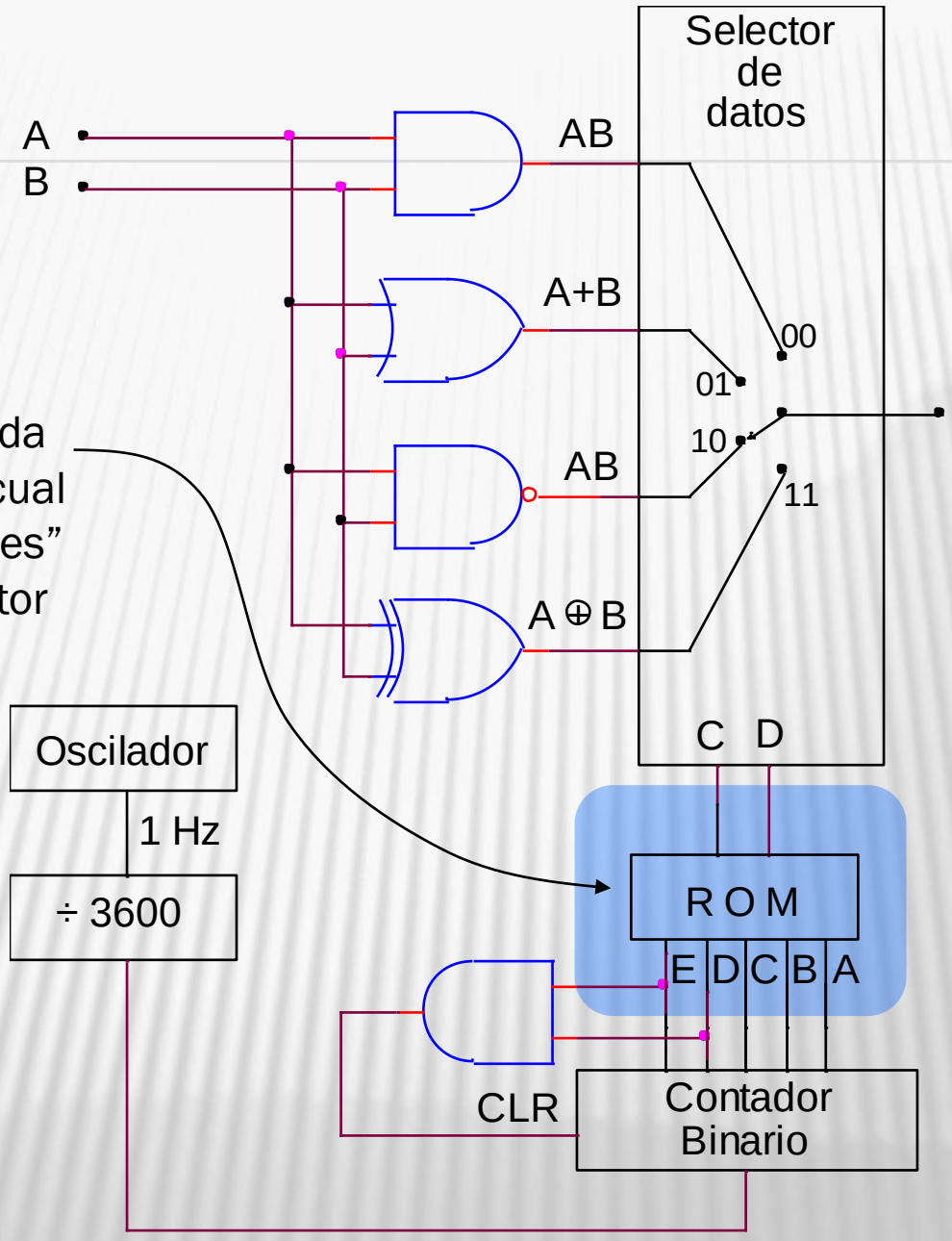



Selector
de
datos
A
B
AB
A+B
AB
A ⊕ B
00
01
10
11
C D
Se agrega una
base de tiempo
Oscilador
1 Hz

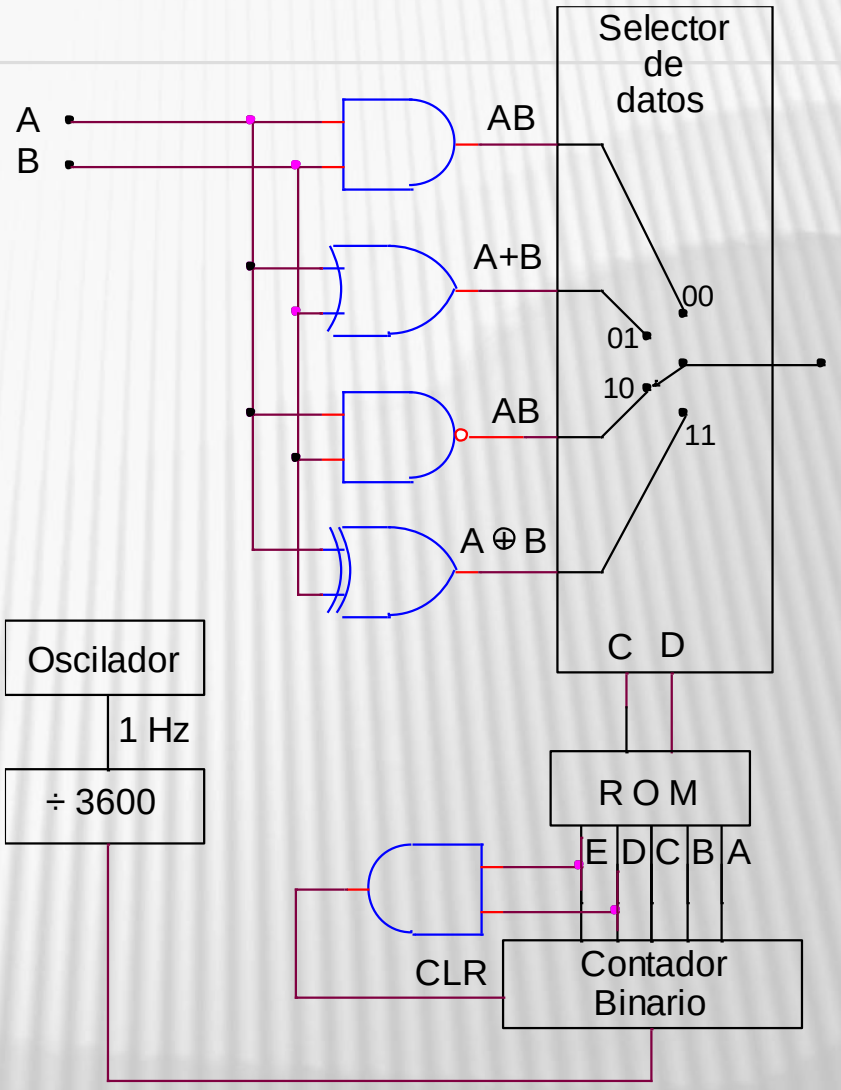

Selector de datos
A
B
AB
A+B
AB
A ⊕ B
00
01
10
11
C D
Con un divisor de frecuencia y un contador se hace un conteo binario con base en un reloj
Oscilador
1 Hz
÷ 3600
Contador Binario
1 pulso / Hora


Selector de datos
A
B
AB
A+B
AB
A ⊕ B
00
01
10
11
C D
Se agrega un reset para hacerlo de 24 horas
Oscilador
1 Hz
÷ 3600
CLR
E D C B A
Contador Binario

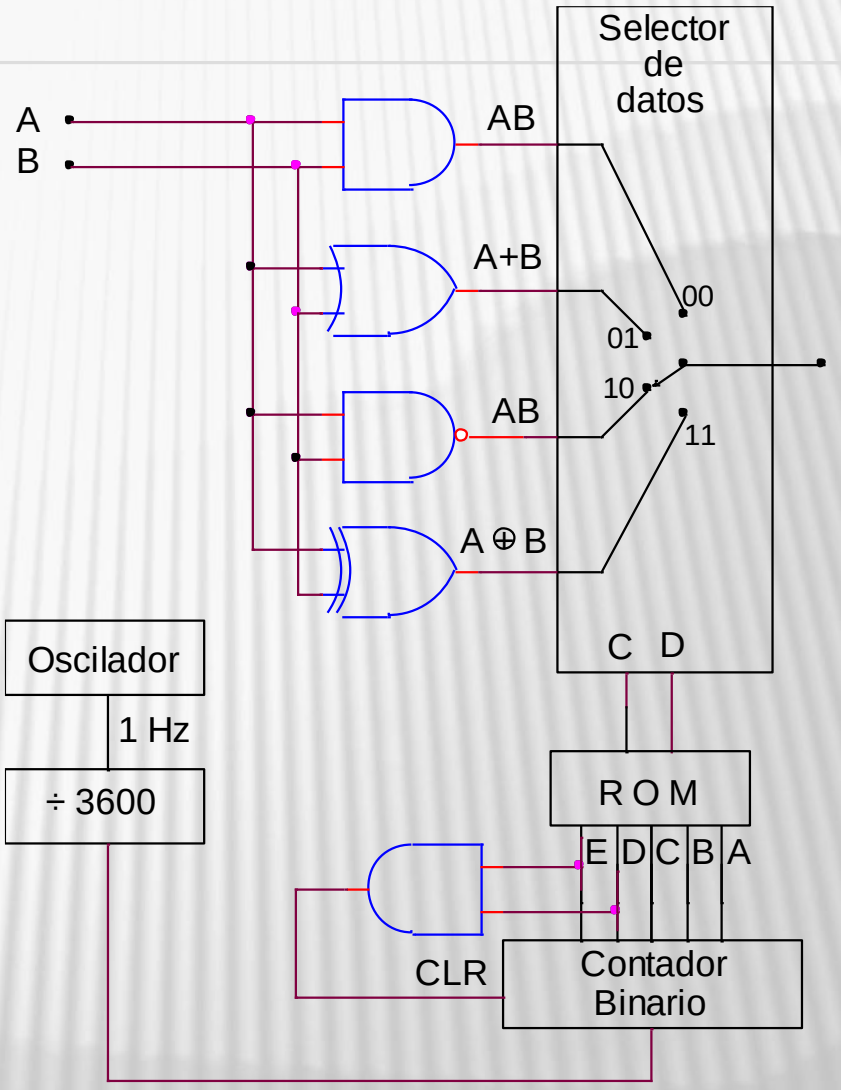
Se agrega una memoria controlada por el contador,la cual da las ”instrucciones” correctas al selector de datos



Hora	Valor del Contador (dirección)	Valor Almacenado (dato)
12:00 a.m.	00000	
01:00 a.m.	00001	
02:00 a.m.	00010	
03:00 a.m.	00011	
04:00 a.m.	00100	
05:00 a.m.	00101	
06:00 a.m.	00110	
07:00 a.m.	00111	
08:00 a.m.	01000	
09:00 a.m.	01001	
10:00 a.m.	01010	
11:00 a.m.	01011	
12:00 p.m.	01100	
01:00 p.m.	01101	
02:00 p.m.	01110	
03:00 p.m.	01111	
04:00 p.m.	10000	
05:00 p.m.	10001	
06:00 p.m.	10010	
07:00 p.m.	10011	
08:00 p.m.	10100	
09:00 p.m.	10101	
10:00 p.m.	10110	
11:00 p.m.	10111	



Hora	Valor del Contador (dirección)	Valor Almacenado (dato)
12:00 a.m.	00000	11
01:00 a.m.	00001	11
02:00 a.m.	00010	11
03:00 a.m.	00011	11
04:00 a.m.	00100	11
05:00 a.m.	00101	11
06:00 a.m.	00110	11
07:00 a.m.	00111	11
08:00 a.m.	01000	00
09:00 a.m.	01001	00
10:00 a.m.	01010	00
11:00 a.m.	01011	01
12:00 p.m.	01100	01
01:00 p.m.	01101	01
02:00 p.m.	01110	01
03:00 p.m.	01111	10
04:00 p.m.	10000	10
05:00 p.m.	10001	10
06:00 p.m.	10010	10
07:00 p.m.	10011	10
08:00 p.m.	10100	10
09:00 p.m.	10101	10
10:00 p.m.	10110	10
11:00 p.m.	10111	11

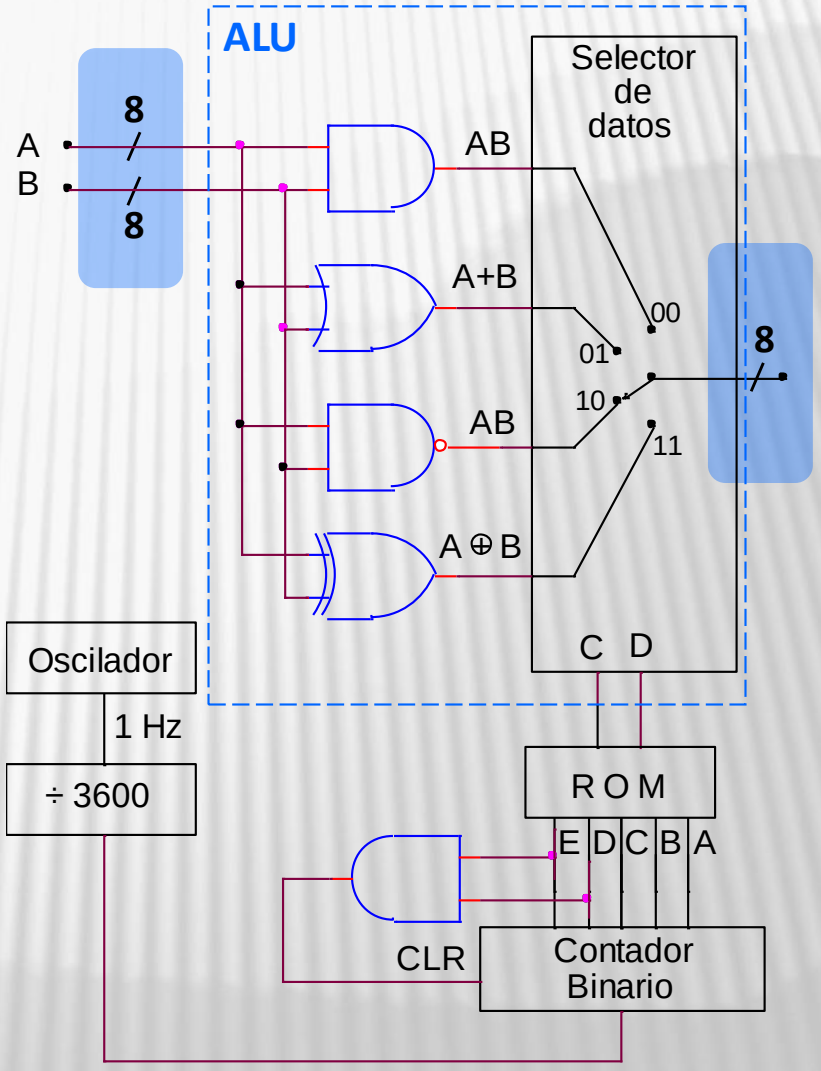


Para convertir este sistema digital en un procesador de aplicación más general, pueden hacerse algunos cambios en su arquitectura.

1. Se **m**odifica el selector de datos, para permitir datos de 8 bits.

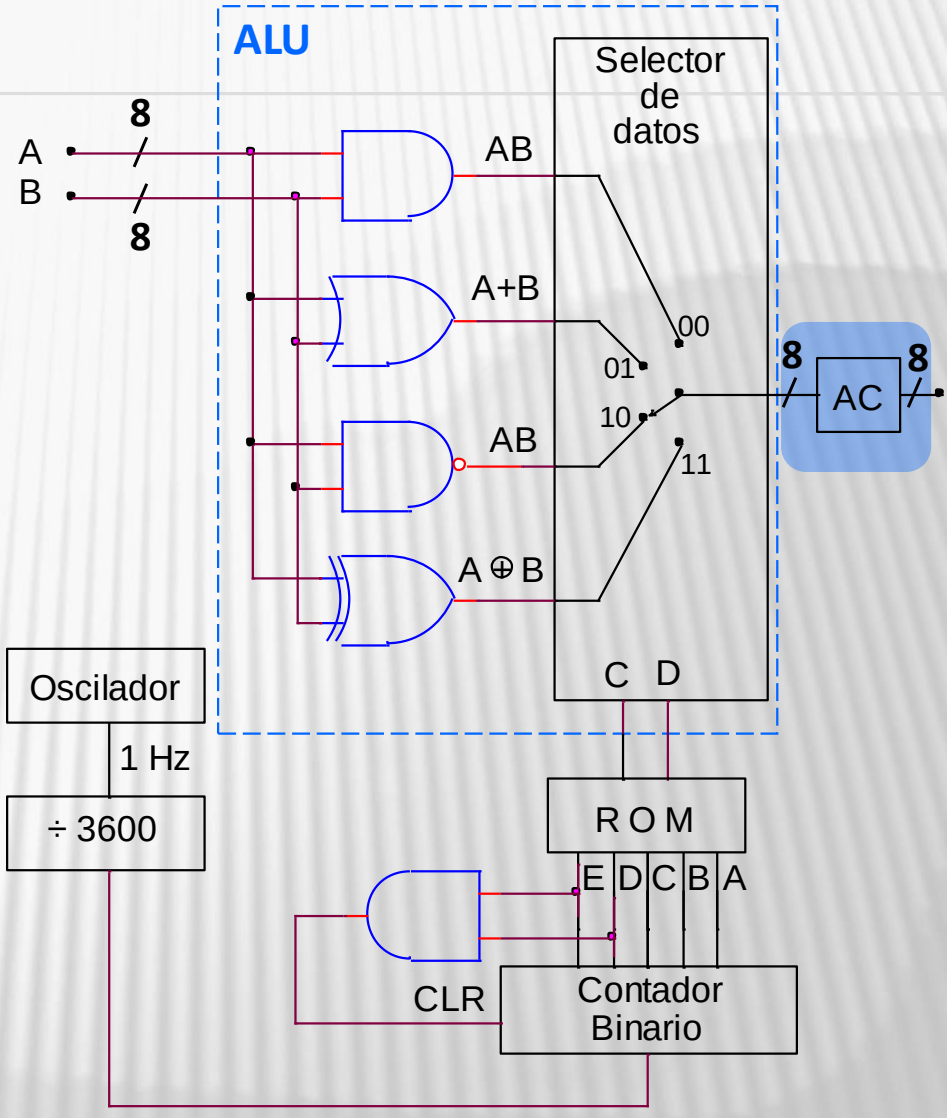
El selector se convierte en la Unidad Aritmético Lógica (ALU) cuya función es realizar las operaciones aritméticas y lógicas del procesador.

Este módulo se encuentra dentro de todos los microprocesadores. Sin embargo dependiendo del procesador, la complejidad de la ALU es diferente.



2. Se agrega un registro a la salida de la ALU cuya función es retener el resultado de la ALU.

Este registro tiene un nombre casi estandarizado en los procesadores: El acumulador, ACC o AC.



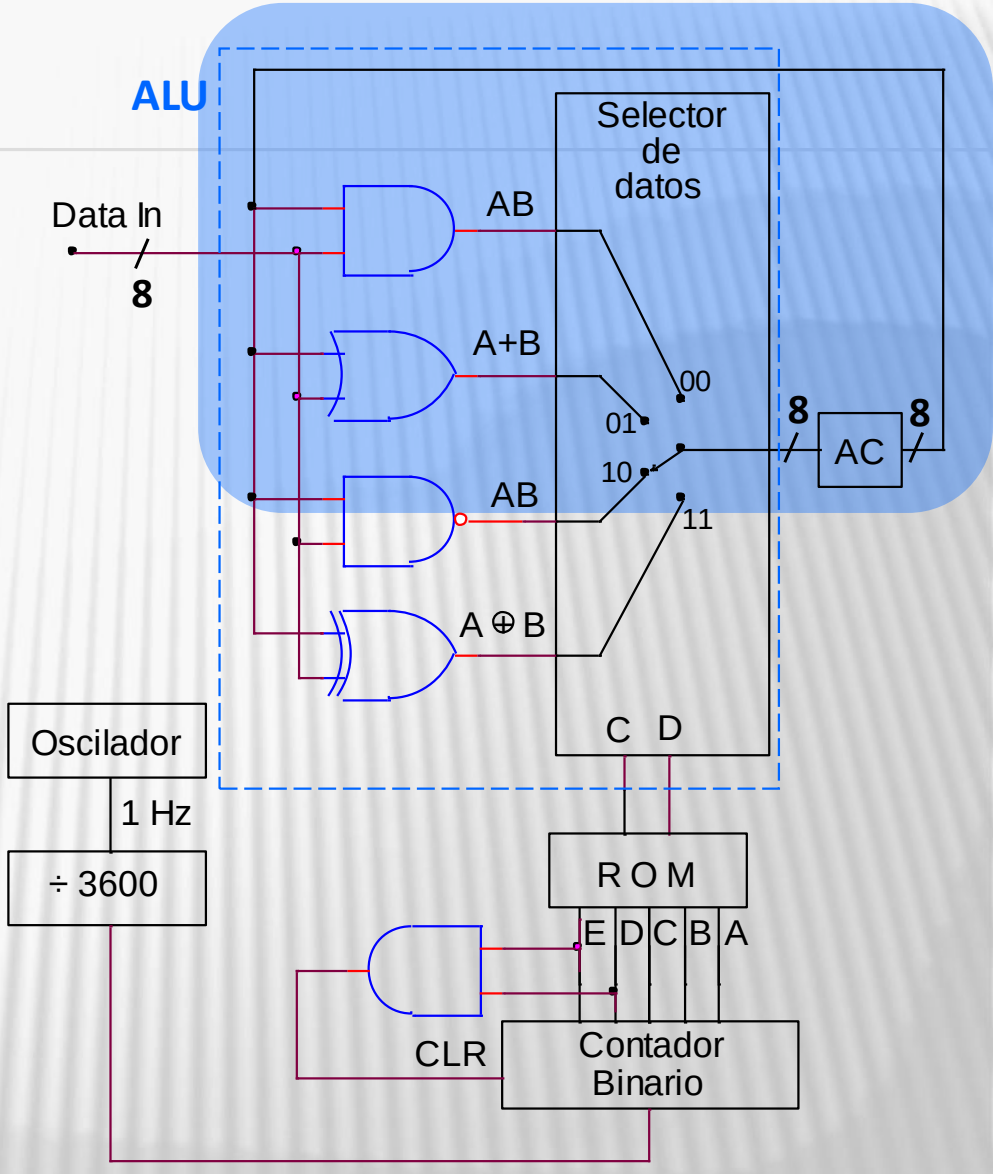
3. Se conecta la salida de AC a una entrada de la ALU.

Esta conexión de retroalimentación es muy común en un procesador. Permite operar sobre un dato de entrada y el resultado anterior.

Nótese que el resultado vuelve a quedar en el acumulador.

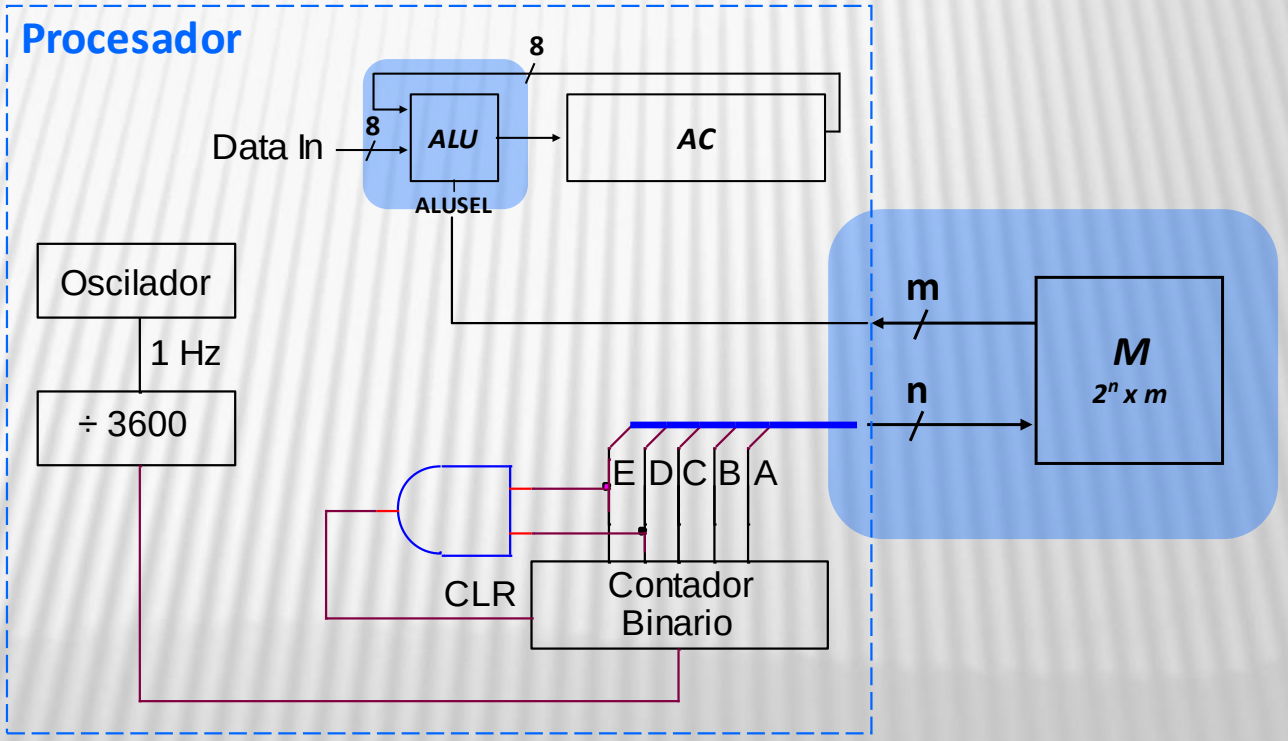
Por ejemplo, esto es útil en operaciones del tipo:

```
int a=5;
int b=15;
a=a+b;          //a=5+15=20
```



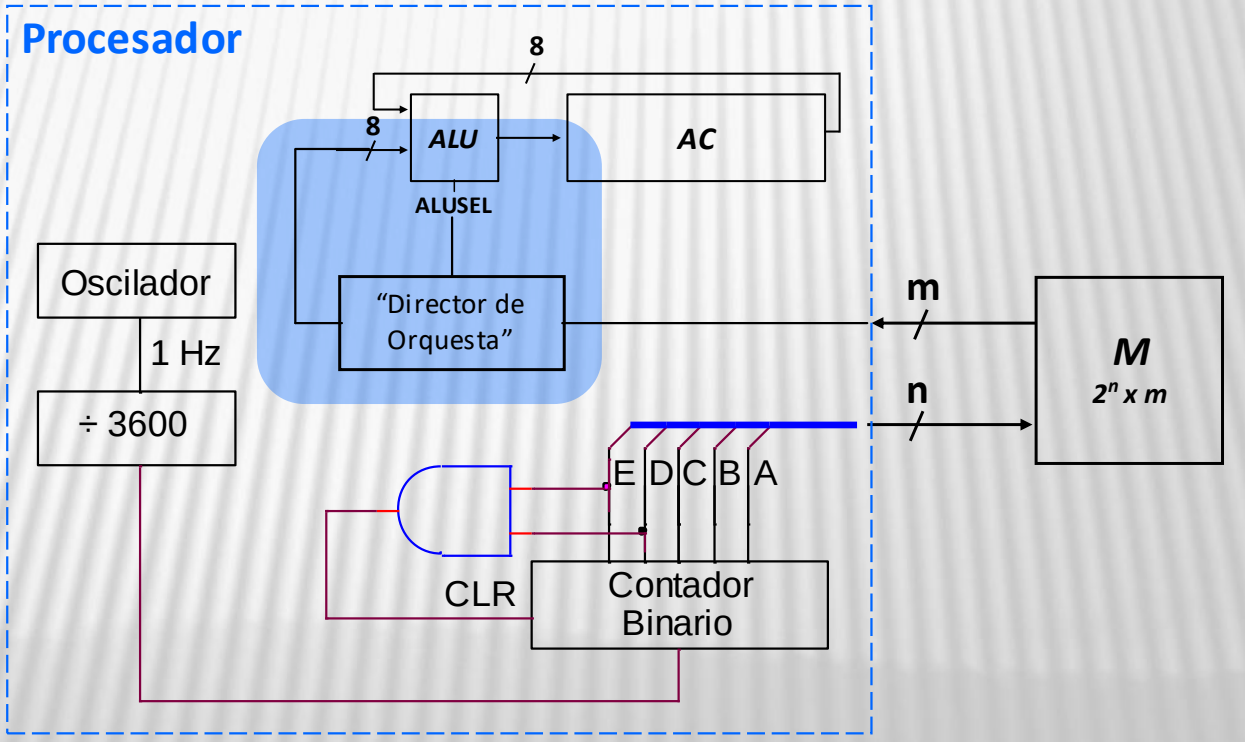
4. Se usa una memoria externa, de modo que esta pueda ser de mayor capacidad. La capacidad de memoria interna que puede incluirse en el procesador es limitada. Esto permite grabar programas más grandes y guardar mayor cantidad de datos .

Además, hacemos caja negra a la ALU y la señal de control la cambiamos por un nombre más estándar, en este caso ALUSEL.



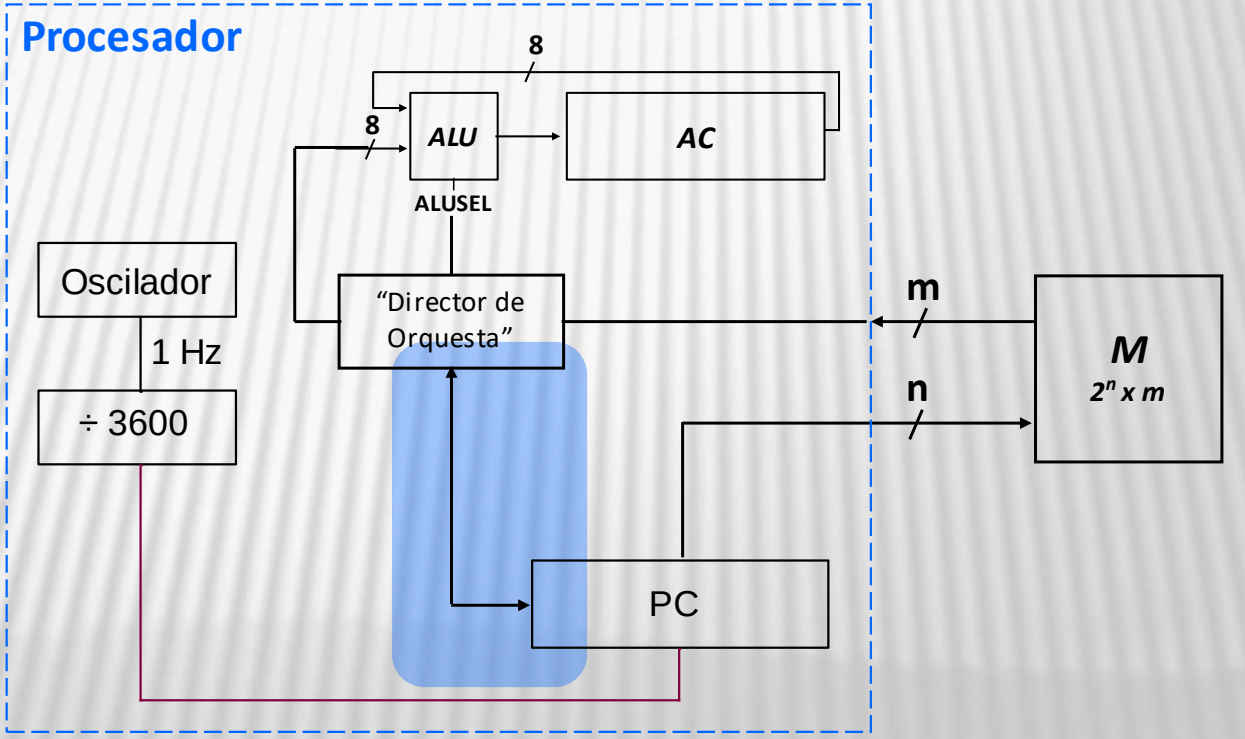
5. Se proporciona el dato de entrada desde la memoria. Esto permite que el programa trabaje sobre operandos guardados en la misma.

Así mismo, se hace necesario agregar un "director de orquesta" que se encargue de decidir qué parte del dato obtenido de memoria es instrucción y qué parte es dato.



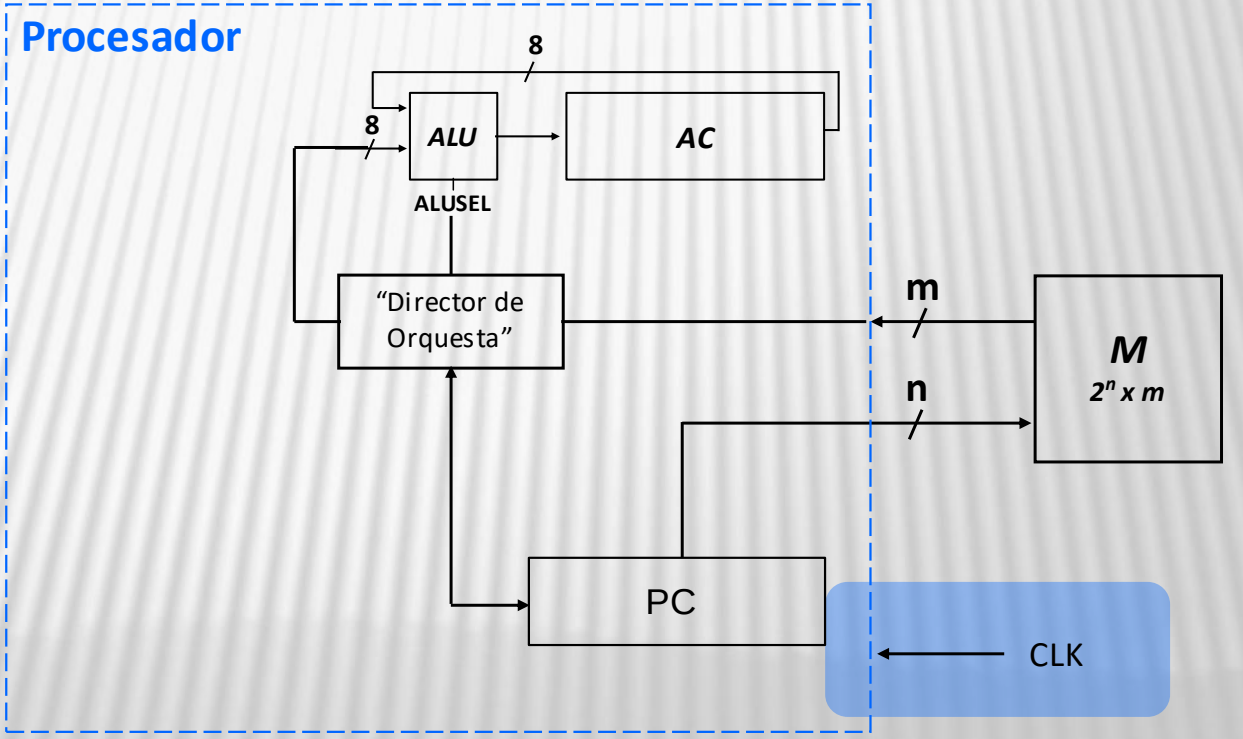
6. Se le otorga al “director de orquesta” el control del contador. El director de orquesta puede incrementar el valor, reiniciarlo o cargar cualquier otro valor. Esto significa que habrá instrucciones con la capacidad de modificar el valor del contador.

El contador toma el nombre de contador de programa o **PC**, ya que precisamente indica cuál instrucción es la que se “adquiere” desde memoria.



7. Finalmente, se agrega un fuente de reloj más acorde para realizar operaciones eficientes, es decir, de más alta frecuencia.

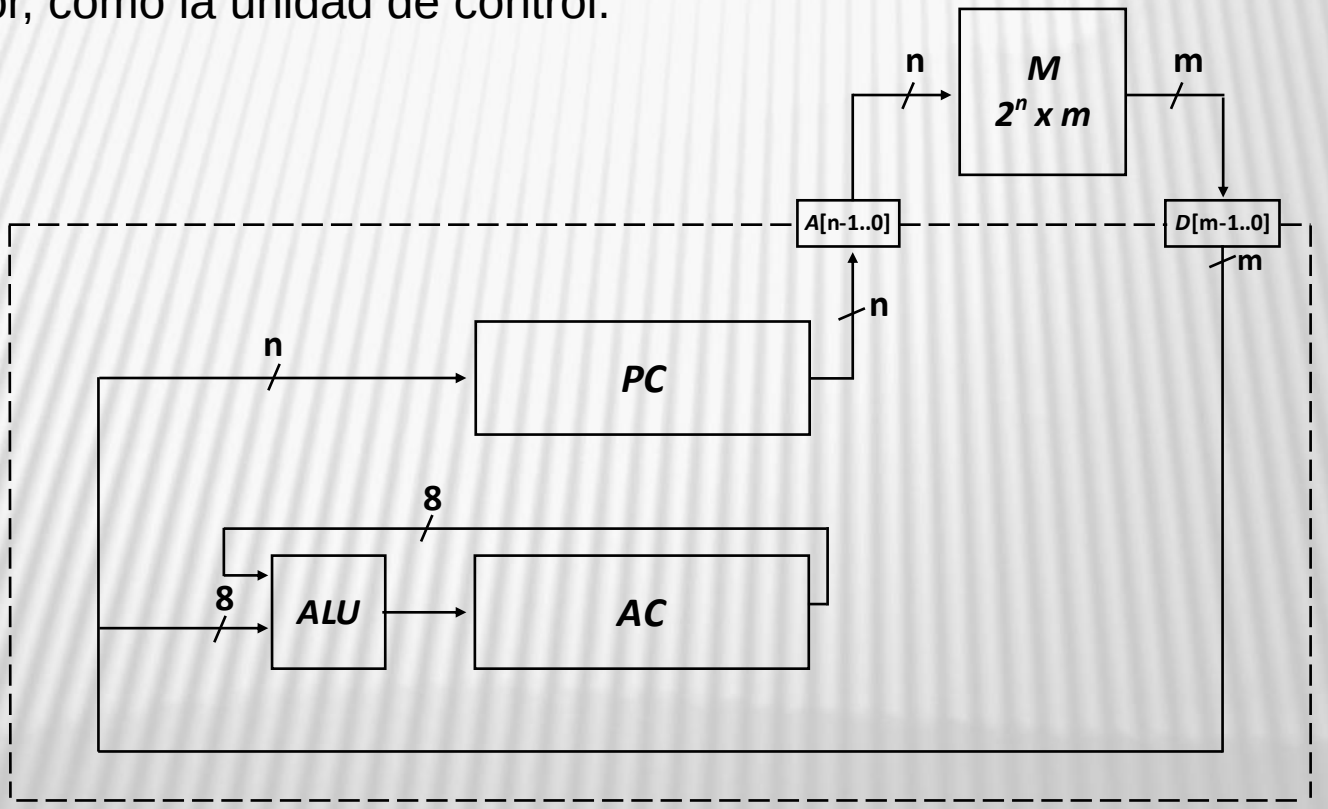
La frecuencia máxima de operación estará determinada por la tecnología de construcción del procesador, por ejemplo tamaño y tipo de transistores.



Reacomodando el diagrama del procesador se obtiene un diagrama como el mostrado en esta diapositiva.

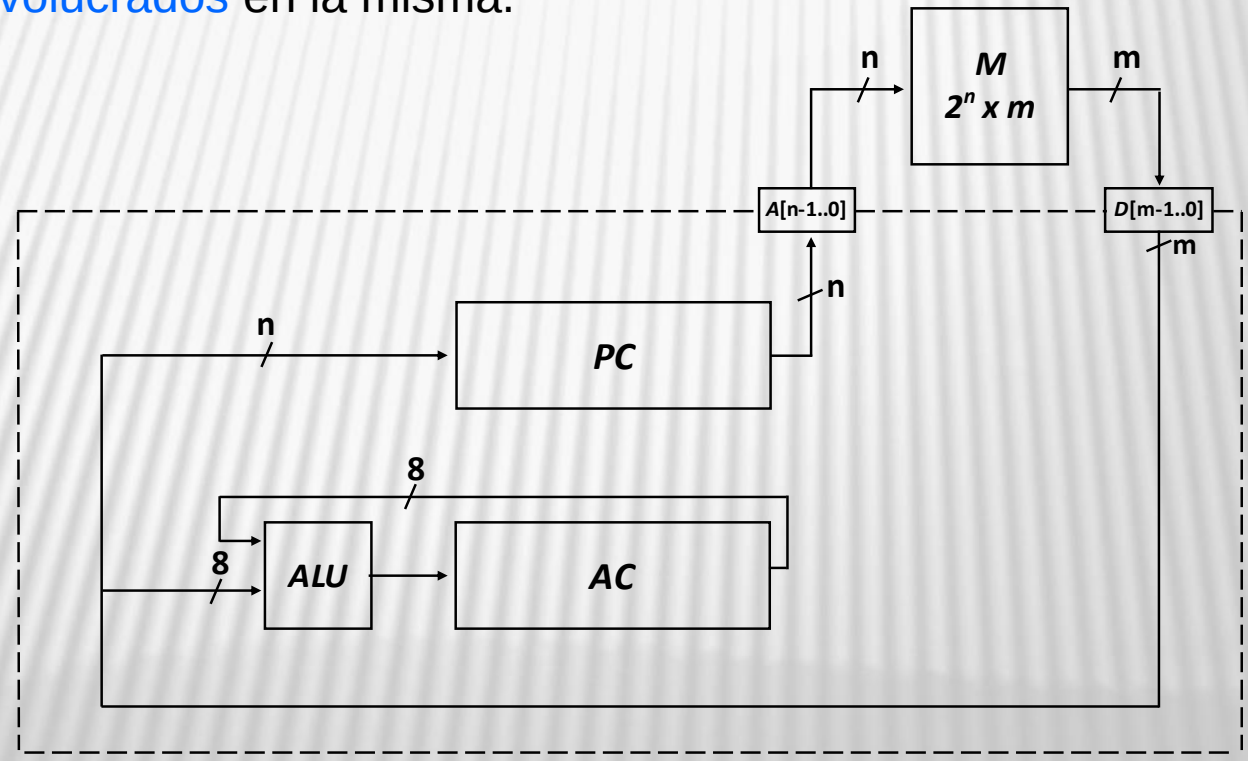
Este diagrama solo muestra el modelo del programador omitiendo al “director de orquesta”, cuyo nombre formal es unidad de control.

El modelo del programador solo muestra los módulos importantes para el programador y oculta todo lo relacionado con el funcionamiento interno del procesador, como la unidad de control.



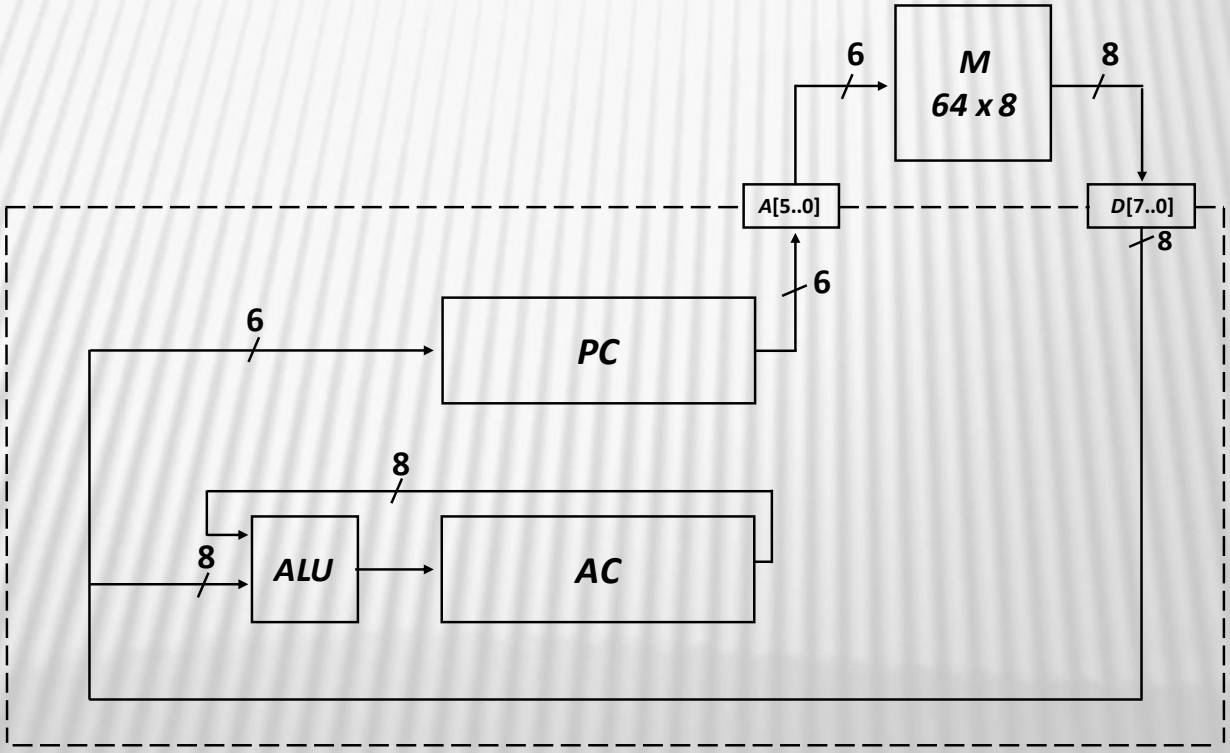
La unidad de control sigue siendo la responsable de “dirigir” el funcionamiento del procesador solo que no es relevante para entender las operaciones que el mismo hace.

Ejemplo. Suponiendo una instrucción que suma el contenido del acumulador con un dato de memoria, para el programador no es relevante saber cómo se desarrolla detalladamente cada paso de la instrucción. Para él solo es relevante saber los operandos y el resultado de la instrucción, así como los registros involucrados en la misma.



Finalmente, para detallar más el modelo de este procesador teórico, supongamos que:

1. El PC es de 6 bits. Esto implica que la memoria que se usará es de 64 localidades, dado que se necesitan 6 bits para direccionar esa cantidad de localidades.
2. El AC es de 8 bits. Esto se había planteado ya previamente. Este número es ya un estándar para la base del tamaño de datos en las computadoras actuales. Esto implica que la memoria tiene localidades de 8 bits.



Basado en la arquitectura anterior, supongamos ahora que el nuevo procesador que pueda ejecutar la siguientes instrucciones:

Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ^ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1

M[AAAAAA] significa el dato almacenado en la dirección AAAAAA de memoria. Para ejemplo, considérese la siguiente instrucción (mnemónico y argumento):

```
ADD 25
```

Esta instrucción implica tomar el valor almacenado en el acumulador y sumarle el valor almacenado en la dirección 25 de memoria. El resultado se guardará de regreso en el acumulador.

La instrucción previa se muestra usando mnemónicos, es decir letras que ayudan a recordar qué instrucción es y que posteriormente se codificarán para que las entienda el procesador. En este caso, de acuerdo a la columna 2, la instrucción previa se codifica en binario como sigue:

```
ADD 25 -> 00011001
```

La segunda instrucción funciona de la misma manera, solo que en este caso la operación realizada es una AND lógica (bit con bit).

Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1

En el caso de la instrucción JMP se tiene algo más elaborado. Esta operación no afecta el contenido del acumulador, sino el del contador de programa. Haciendo un equivalente con el procesador del primer ejemplo, se trata de cambiar el valor del contador que funcionaba como reloj. Cambiar el valor del PC implica ejecutar instrucciones desde otra dirección de memoria. Esta operación se conoce como salto. Así:

```
JMP 17
```

cambiará el valor del PC a 17, de modo que la siguiente instrucción que se ejecute sea la contenida en la localidad de memoria 17. La codificación en binario es:

```
JMP 17 -> 10010001
```

Finalmente, la instrucción INC es la más sencilla de todas.

Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ^ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1

Esta instrucción modifica el valor del acumulador, solo que no requiere un valor de memoria. Simplemente le suma 1 al valor del acumulador. Así, suponiendo que el acumulador vale 15:

```
INC
```

el acumulador guardará el valor 16 después de ejecutar la instrucción. La codificación en binario es:

```
INC -> 11XXXXXX
```

donde X es cualquier número, ya que en esta instrucción no es requerido el campo de la dirección.

Del set de instrucciones de este nuevo procesador teórico podemos ver algunas cosas interesantes:

1. Hay un campo de **código de operación** en cada instrucción. Este es el que permite diferenciar una instrucción de otra.

2. La mayoría de las instrucciones tienen un **campo de dirección**. El campo de dirección contiene tal como su nombre lo dice una dirección que indica dónde se encuentra el dato con el que se hará la operación.

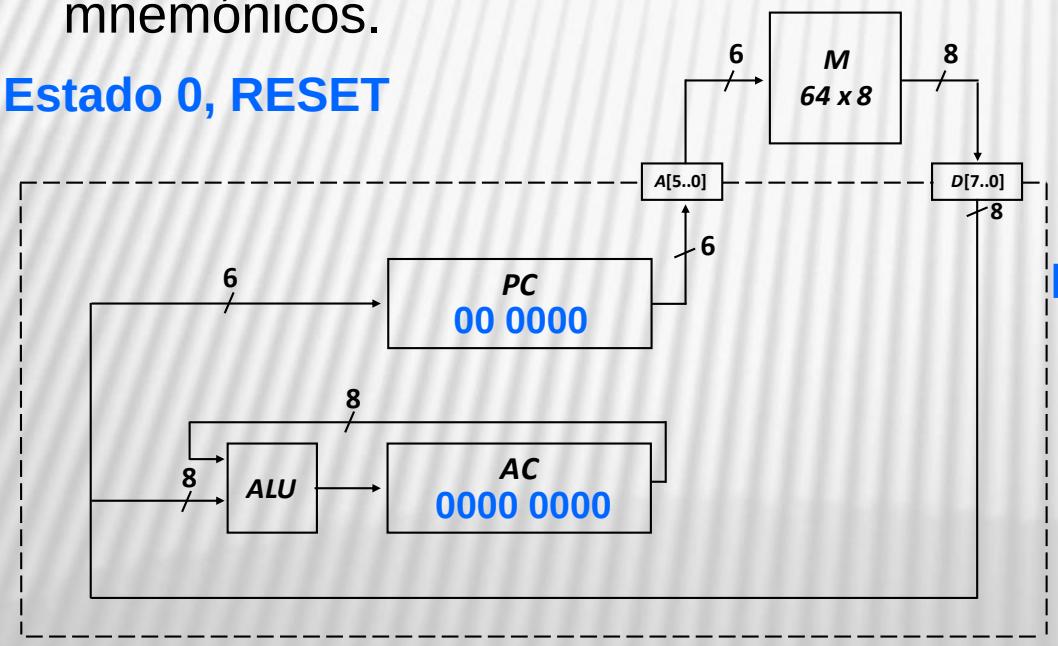
Algo importante a considerar es que, mientras no haya una instrucción JMP, el valor del PC se autoincrementa, es decir, el procesador ejecuta las instrucciones desde memoria en secuencia. Con base en resultados previos, el procesador puede saltar a otras direcciones en el programa y ejecutar otros segmentos (secciones) de un mismo programa.

Al ser el microprocesador una máquina de estados finitos, se requiere un RESET para establecer el valor del PC en un valor conocido que corresponde con el inicio del programa (que con mucha frecuencia es la dirección 0).

Ejemplo de ejecución de un programa paso a paso para el nuevo procesador teórico

Consideremos las condiciones iniciales mostradas en los registros de la figura. La tabla de la derecha muestra los datos guardados en las primeras localidades de la memoria. Nótese que en la misma memoria hay datos e instrucciones, es decir, arquitectura Von Newman. La columna central muestra los datos en binario mientras que la columna de la derecha muestra el equivalente con mnemónicos.

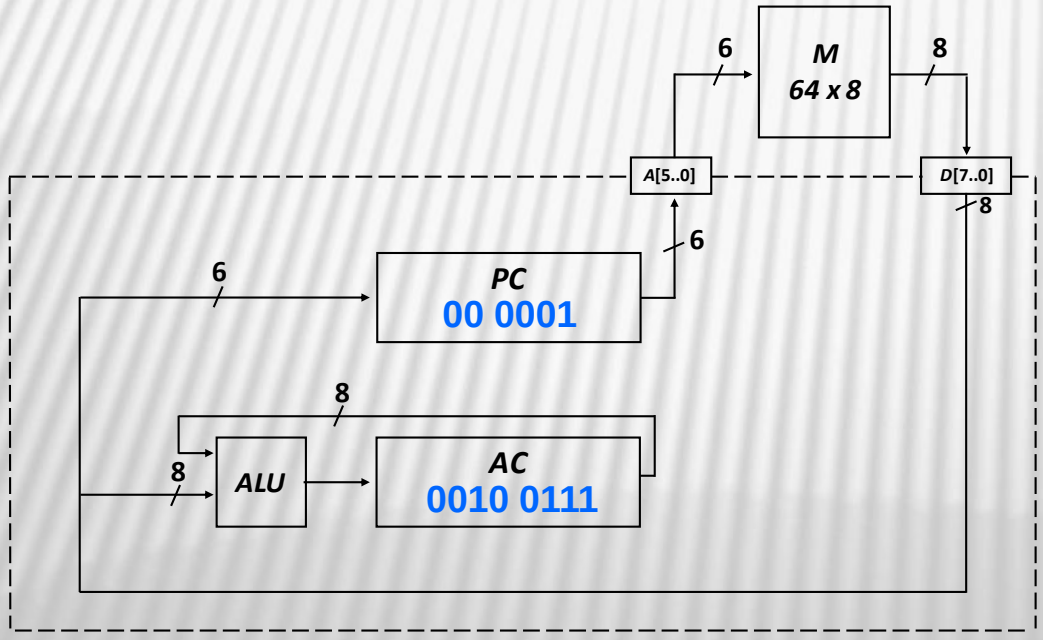
Estado 0, RESET



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

Estado 1, después de ejecutar la instrucción en la dirección 0

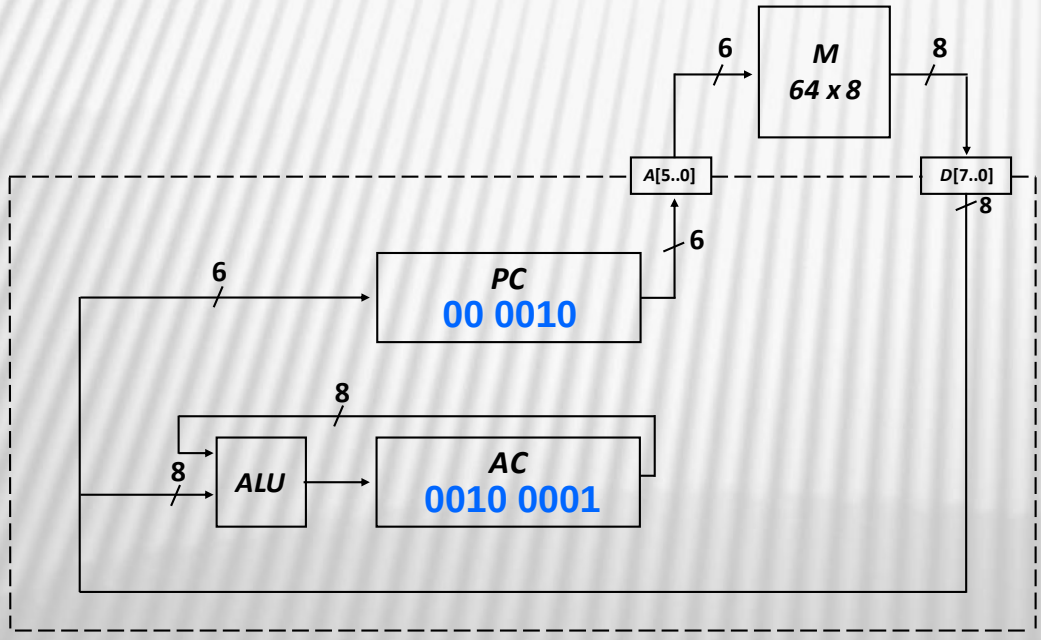
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H
5	0011 1001	39H

Estado 2, después de ejecutar la instrucción en la dirección 1

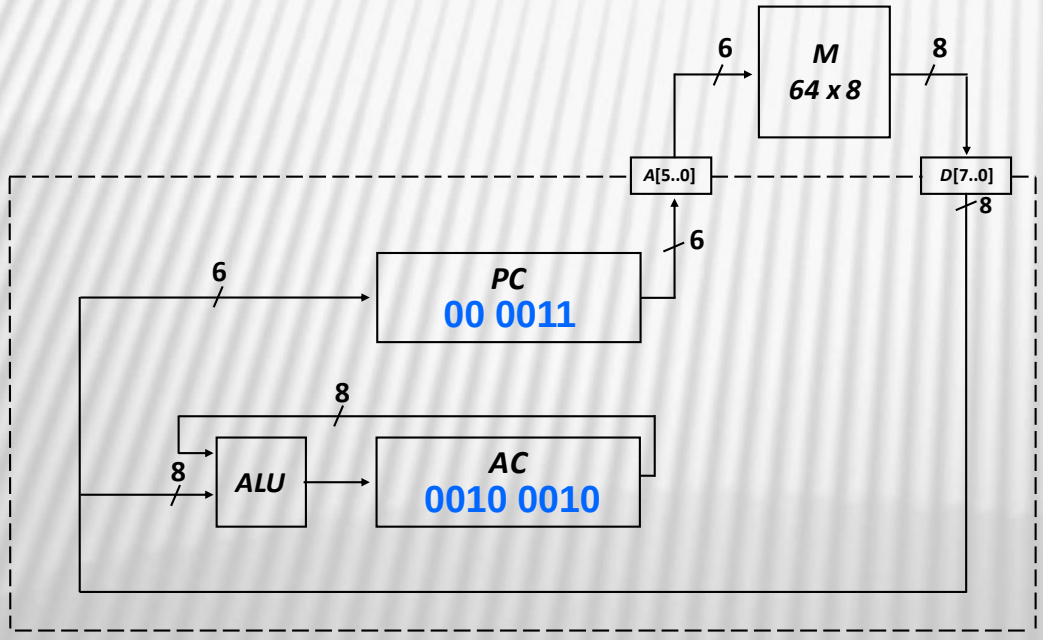
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H
5	0011 1001	39H

Estado 3, después de ejecutar la instrucción en la dirección 2

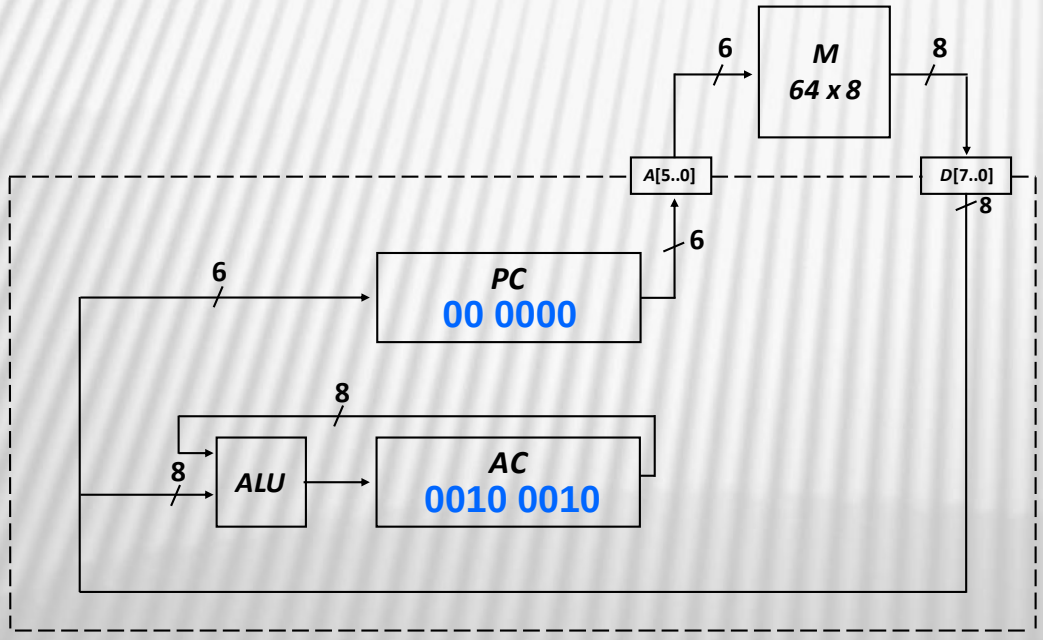
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

Estado 4, después de ejecutar la instrucción en la dirección 3

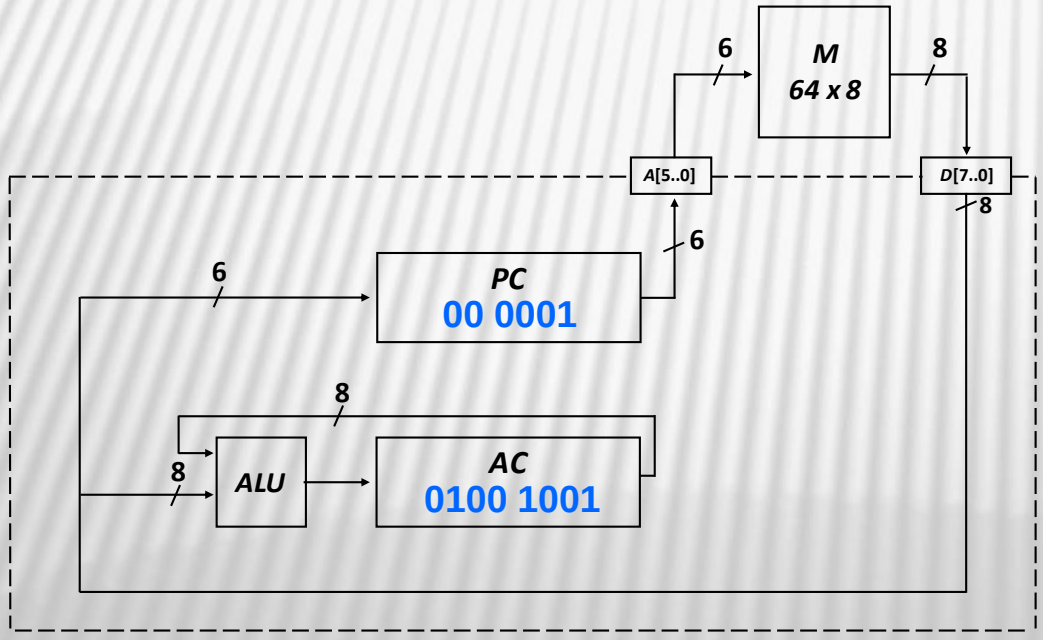
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

Estado 5, después de ejecutar la instrucción en la dirección 0

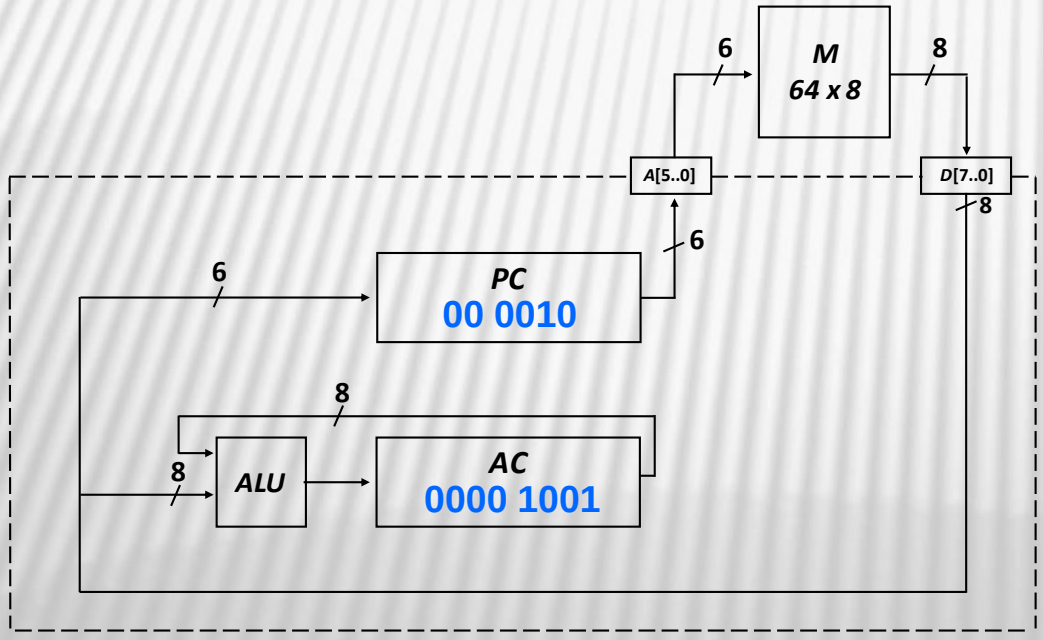
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

Estado 6, después de ejecutar la instrucción en la dirección 1

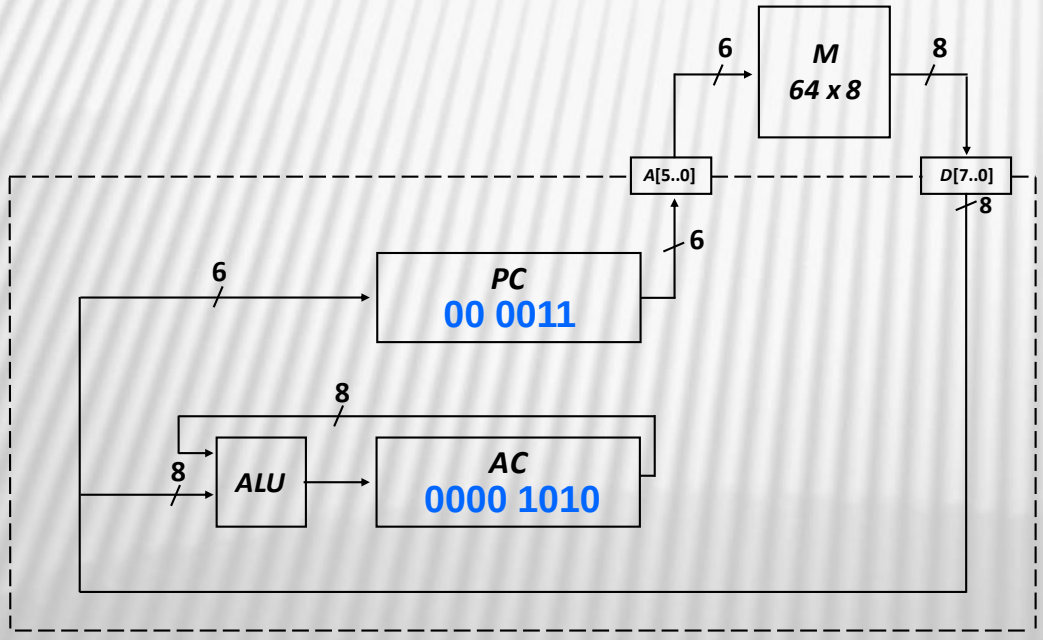
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

Estado 7, después de ejecutar la instrucción en la dirección 2

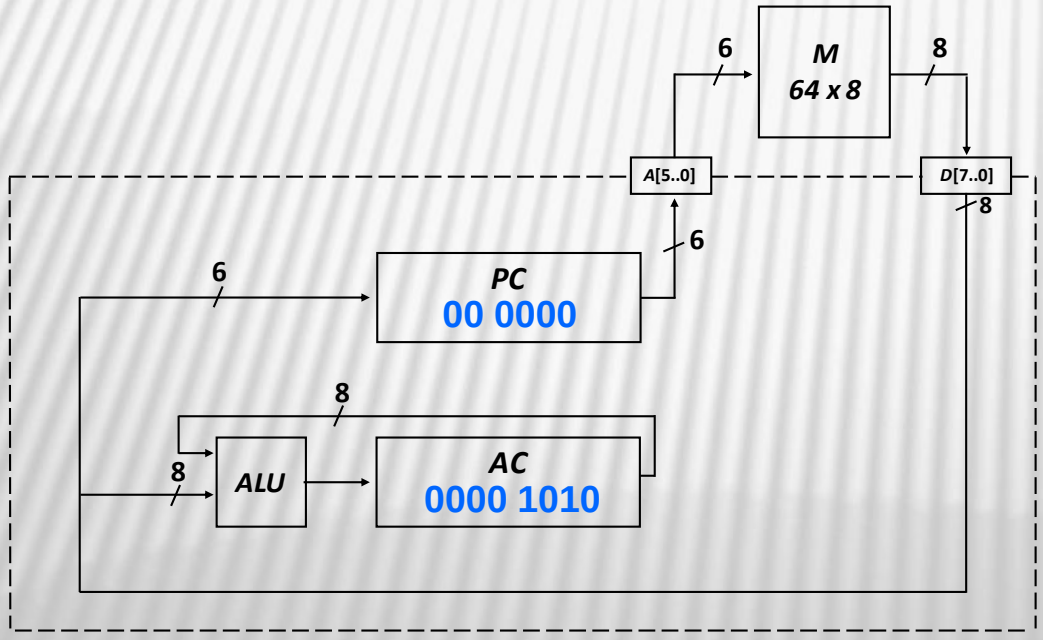
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

Estado 8, después de ejecutar la instrucción en la dirección 3

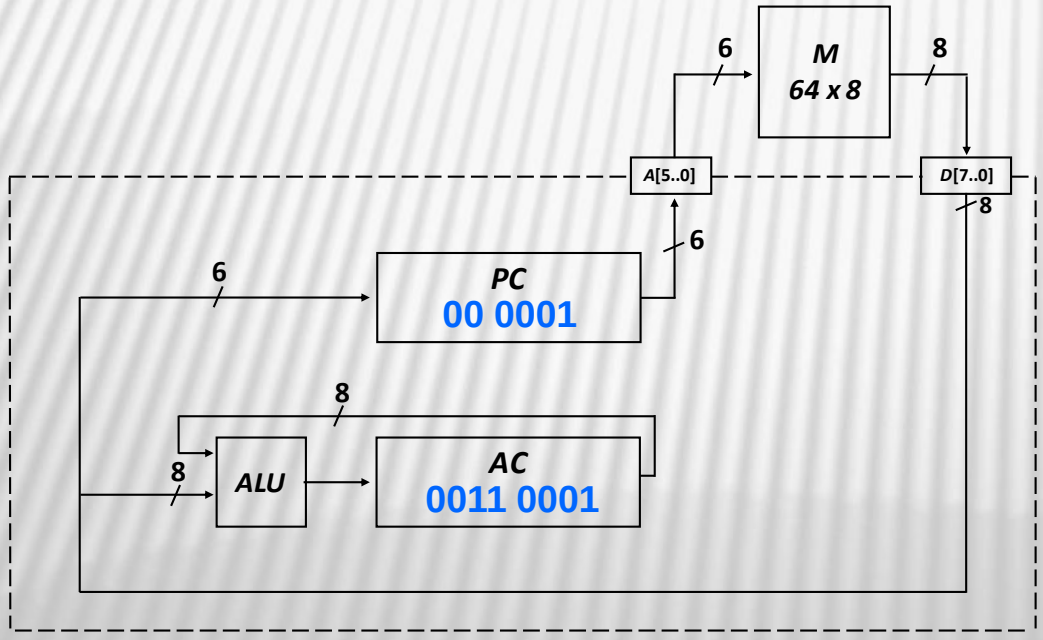
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

Estado 9, después de ejecutar la instrucción en la dirección 0

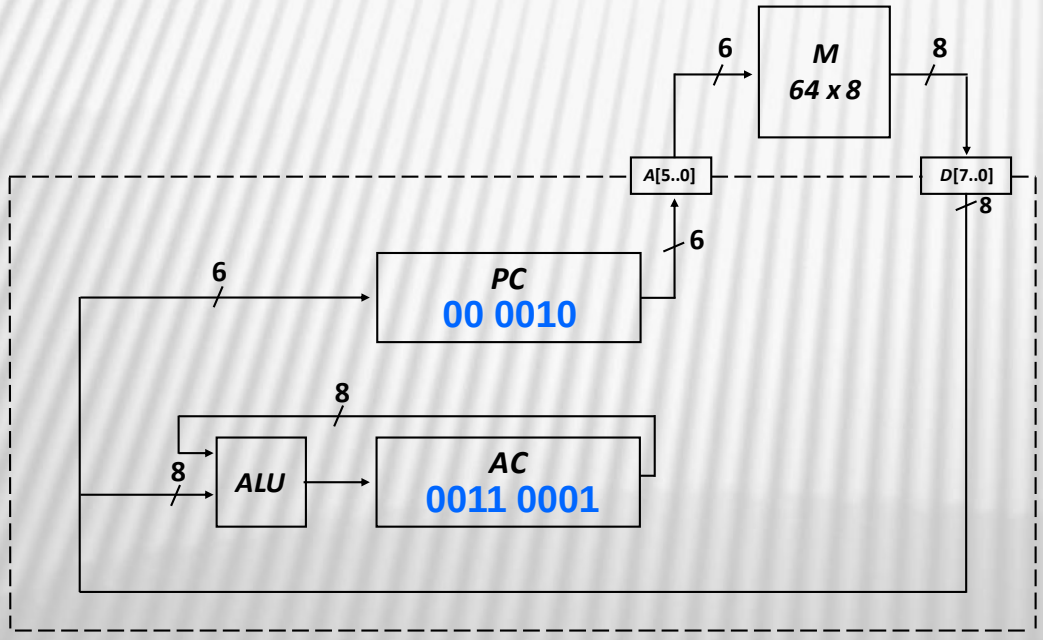
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

Estado 10, después de ejecutar la instrucción en la dirección 1

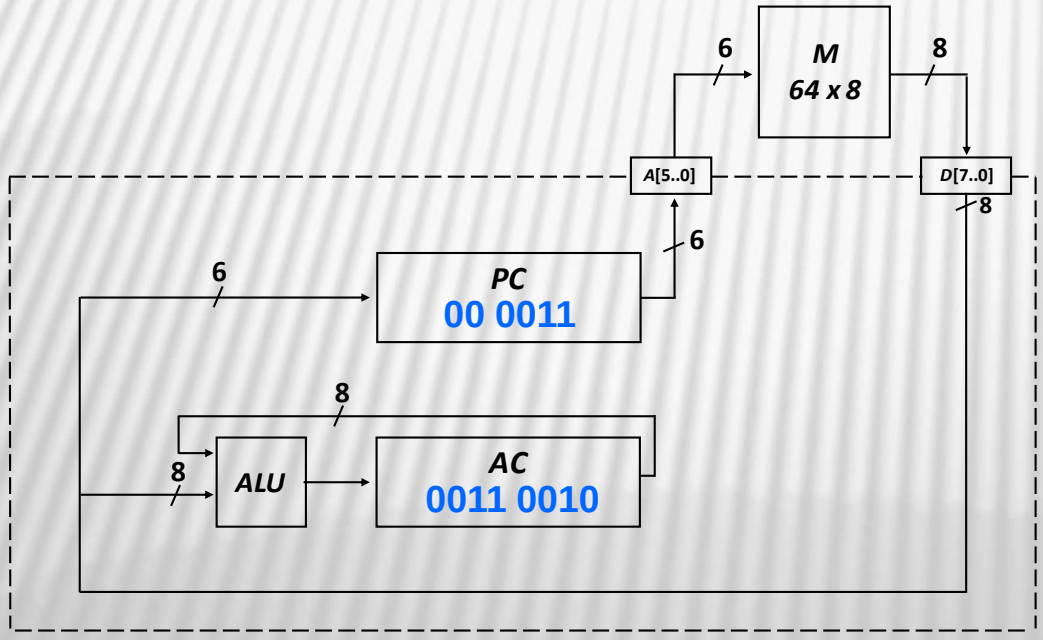
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

Estado 11, después de ejecutar la instrucción en la dirección 2

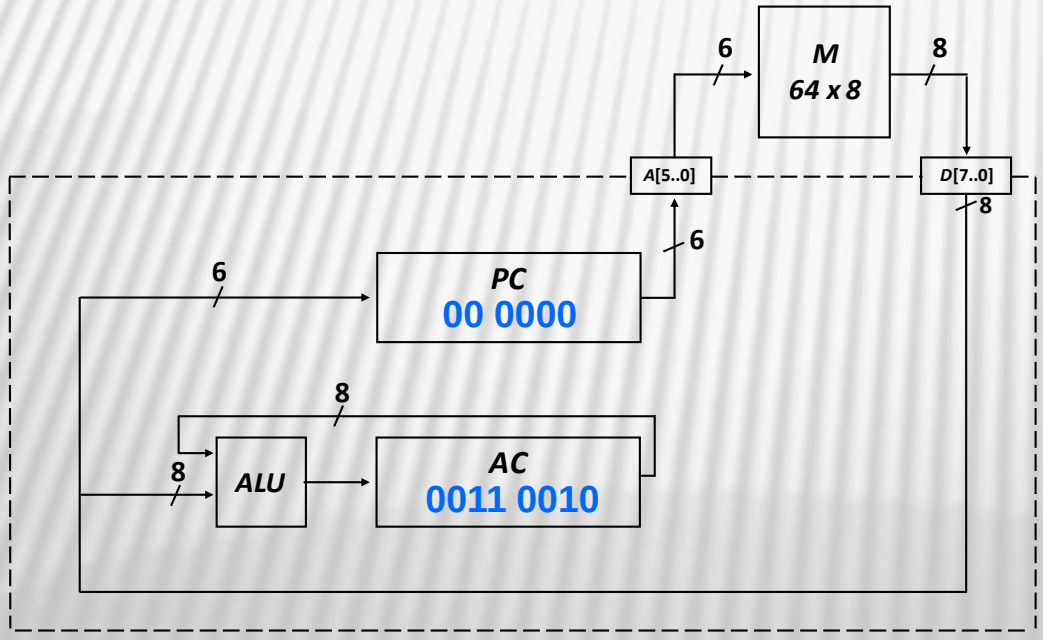
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

Estado 12, después de ejecutar la instrucción en la dirección 3

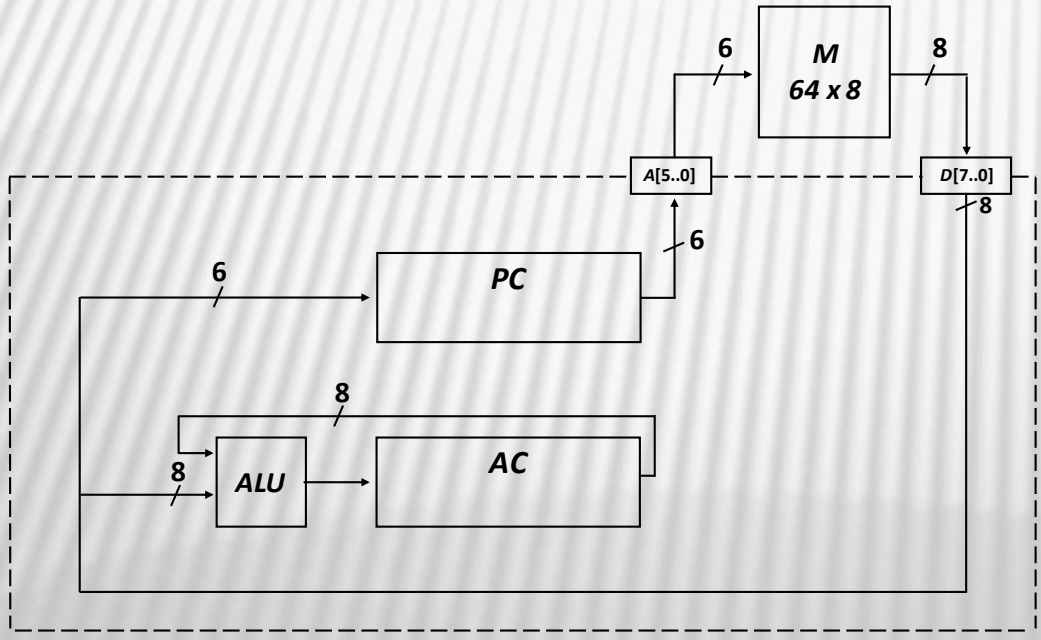
Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57

¿Puedes dar los valores para el acumulador para los estados 13, 14, 15 y 16?

Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ∧ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1



Dirección	Dato (Binario)	Codificación (Mnemónicos o número)
0	0000 0100	ADD 4
1	0100 0101	AND 5
2	1100 0000	INC
3	1000 0000	JMP 0
4	0010 0111	27H ó 39
5	0011 1001	39H ó 57