Introducción

Las computadoras se han utilizado de forma general desde los años 50. En un principio las computadoras digitales eran sistemas grandes y costosos utilizados por el gobierno, universidades y grandes empresas.

El tamaño y forma de las computadoras digitales cambiaron gracias a la invención del circuito integrado (IC). El cual permitió tener todo un procesador en una sola pastilla denominándolo **microprocesador**. El microprocesador es un pequeño, pero extremadamente complejo dispositivo *VLSI*.

VLSI: Very Large Scale Integration

Organización de un sistema computarizado

Las bases que conforman un sistema computarizado son:

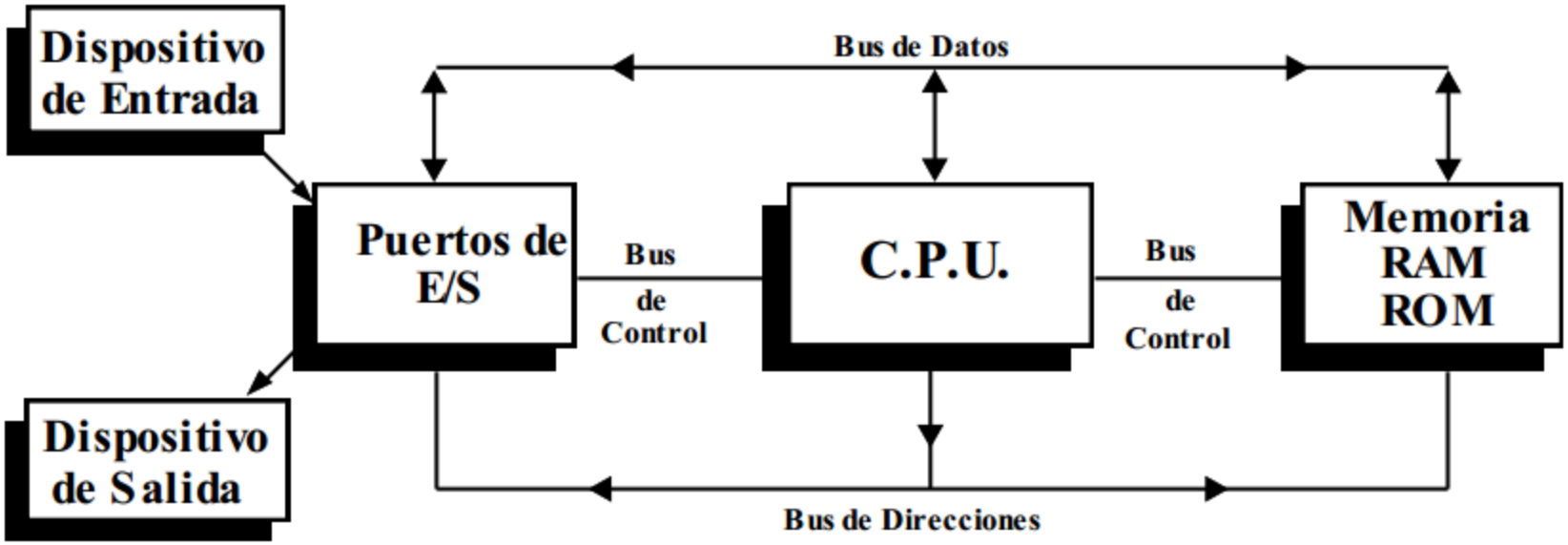
- la Unidad Central de Procesamiento (**CPU**)
- la sección de **Memoria**
- la sección de Entrada/Salida (**E/S**)

Estas tres secciones están interconectadas por tres conjuntos de líneas paralelas llamadas **buses** o ductos.

Estos buses son:

- bus de **Direcciones**
- bus de **Datos**
- bus de **Control**

Organización de un sistema computarizado



Organización de un sistema computarizado

Breve introducción a cada sección y a los buses

Organización de un sistema computarizado

- **Unidad Central de Procesamiento (CPU)**

Controla las operaciones del sistema computarizado, **trae el código binario de las instrucciones** desde la memoria, **decodifica** las instrucciones a una serie de acciones simples y **ejecuta** tales acciones.

El CPU contiene una unidad aritmética y lógica (**ALU**), la cual realiza operaciones como sumar, restar, or, and, xor, invertir etc., sobre palabras binarias, cuando las instrucciones así lo requieran.

El CPU también contiene un contador de direcciones o **Contador de Programa** el cual se utiliza para retener la dirección de la próxima instrucción ser traída desde la memoria.

Además contiene **registros de propósito general** los cuales se utilizan para almacenar temporalmente datos binarios, y una **circuitería de control** que genera las señales del bus de control.

Organización de un sistema computarizado

- **Sección de Memoria**

La sección de memoria es generalmente una mezcla de RAM y ROM.

La memoria tiene dos propósitos, el primero es almacenar el código binario de las **instrucciones** que se quieren ejecutar en el sistema.

El segundo propósito es almacenar el código binario de los **datos** con los cuales se trabajará.

Organización de un sistema computarizado

- **Memoria**

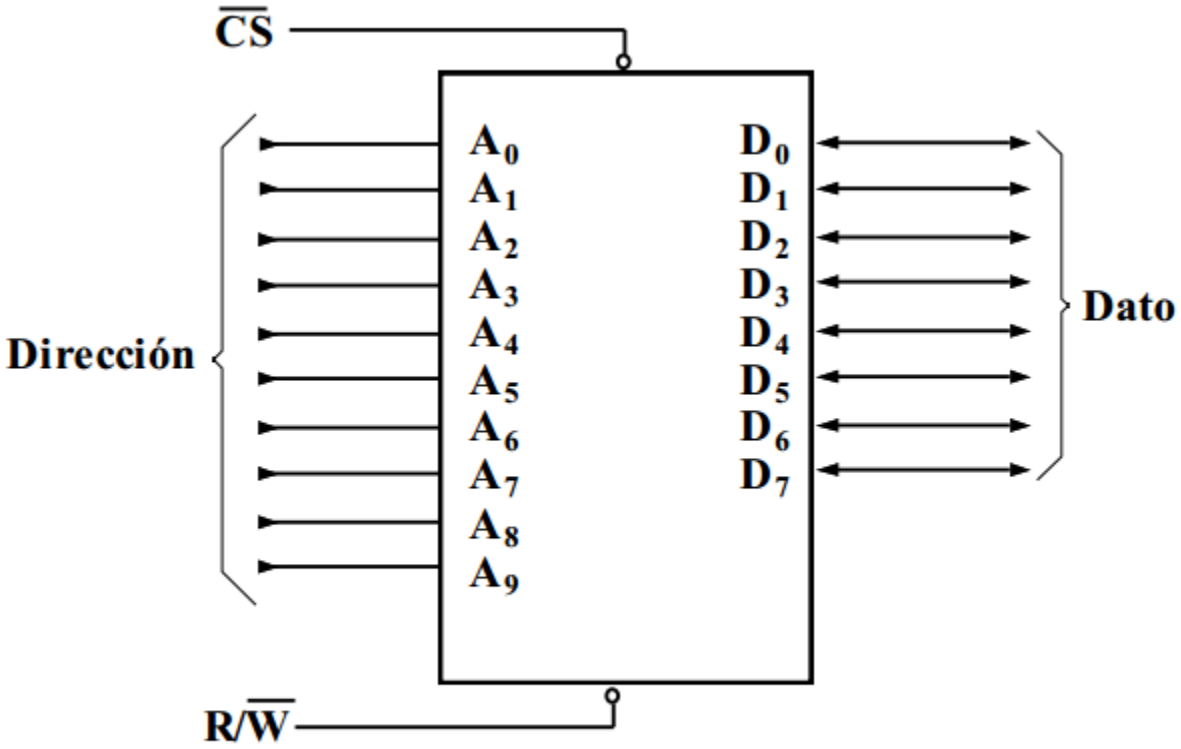


Figura 5. RAM de 1K × 8 bits o 1K bytes

Organización de un sistema computarizado

- **Sección de Entrada/Salida (E/S)**

Permite a la computadora tomar datos del mundo real o mandar datos al mundo real. Los periféricos tales como teclados, pantalla e impresoras se conectan a la sección de E/S.

Los dispositivos físicos utilizados para interfazar los buses de la computadora a sistemas externos se les denomina **puertos**.

Un puerto de entrada permite que la información de un teclado o un convertidor analógico digital (ADC) o alguna otra fuente pueda ser leído por la computadora bajo el control del CPU.

Un puerto de salida se utiliza para mandar información de la computadora a algún periférico tal como pantalla, impresora o un convertido digital analógico (DAC).

Organización de un sistema computarizado

- **Sección de Entrada/Salida (E/S)**

El microprocesador es el foco de todas las operaciones, por tanto una salida significa que el dato **fluye del** microprocesador, mientras que una entrada significa que el dato **fluye hacia** el microprocesador.

Organización de un sistema computarizado

- **Bus de Direcciones**

El bus de direcciones consiste de 16,20,24 o mas líneas de señales en paralelo. Por estas líneas el CPU envía la localidad de memoria en la cual va escribir o leer.

El número de localidades que el CPU puede direccionar o acceder se determina por el número de líneas del bus de direcciones. Si el CPU tiene N líneas de dirección entonces puede direccionar **2^N localidades**.

Cuando el CPU lee o manda datos hacia o desde un puerto, la dirección del puerto también se envía por el bus de direcciones.

Organización de un sistema computarizado

- **Bus de Datos**

El bus de datos consiste de 8,16, 32 o mas líneas de señales en paralelo, estas líneas son **bidireccionales**. Esto significa que el CPU puede leer datos por estas líneas desde la memoria o un puerto así también puede mandar datos a una localidad de memoria o a un puerto.

Muchos dispositivos en un sistema pueden tener sus salidas conectadas al bus de datos, pero solamente uno puede estar habilitado a la vez. Por lo que cualquier dispositivo conectado al bus de datos debe ser de tres estados de forma que los dispositivos que no estén en uso estén flotados.

Organización de un sistema computarizado

- **Bus de Control**

El bus de control consiste de 4 a 10 líneas de señales en paralelo. El CPU manda señales sobre el bus de control para **habilitar** las salidas de los dispositivos de memoria o puertos direccionados.

Generalmente las señales del bus de control son **leer memoria**, **escribir en memoria, leer E/S y escribir E/S**.

Por ejemplo, para leer un dato de una localidad de memoria, el CPU:

1. manda la dirección de la localidad de memoria deseada por el bus de direcciones
2. después manda la señal de lectura de memoria por el bus de control. La señal de lectura habilita al dispositivo de memoria direccionado, el cual
3. proporciona el dato en el bus de datos
4. de donde es leído por el CPU.

Organización de un sistema computarizado

- **Decodificador de Direcciones:**

El decodificador de direcciones es una parte del control lógico. Este genera señales para seleccionar dispositivos cuando una cierta dirección (o rango de direcciones) se presenta en el bus de direcciones.

El siguiente diagrama muestra un decodificador para la dirección 3000H (0011 0000 0000 0000 binario).

Organización de un sistema computarizado

- **Decodificador de Direcciones:**

La salida del decodificador es verdadera (lógica 0) solamente cuando la dirección exacta se presenta en el bus de direcciones. Esta salida puede ser utilizada para habilitar un puerto que tenga asignada la dirección 3000H.

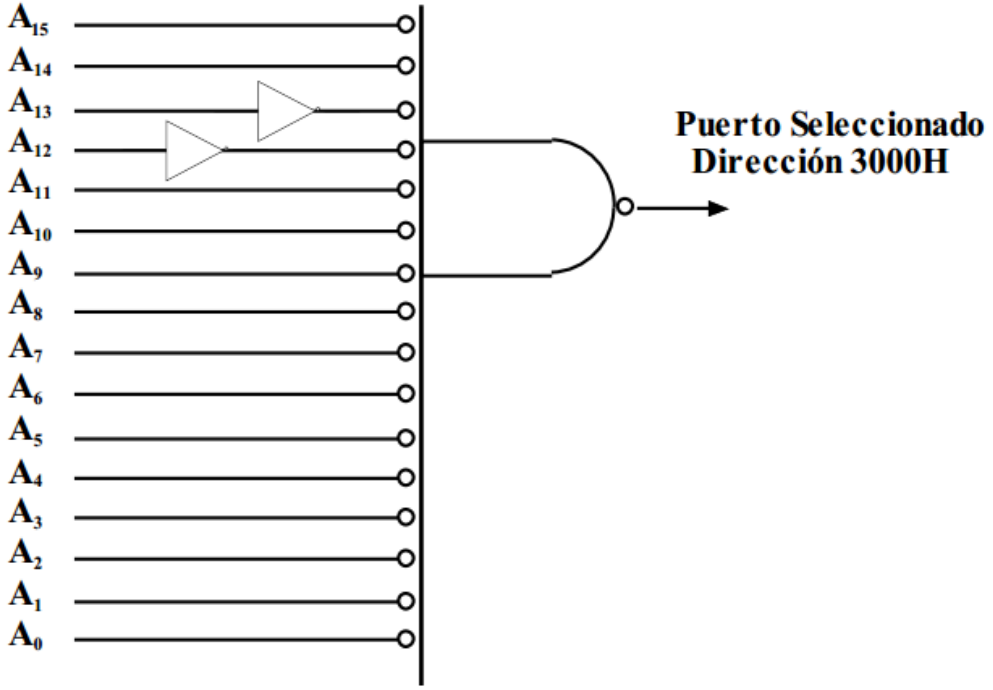


Figura 6. Decodificador de direcciones (dirección 3000H).

Organización de un sistema computarizado

- **Decodificadores de Dirección Para Varios Dispositivos**

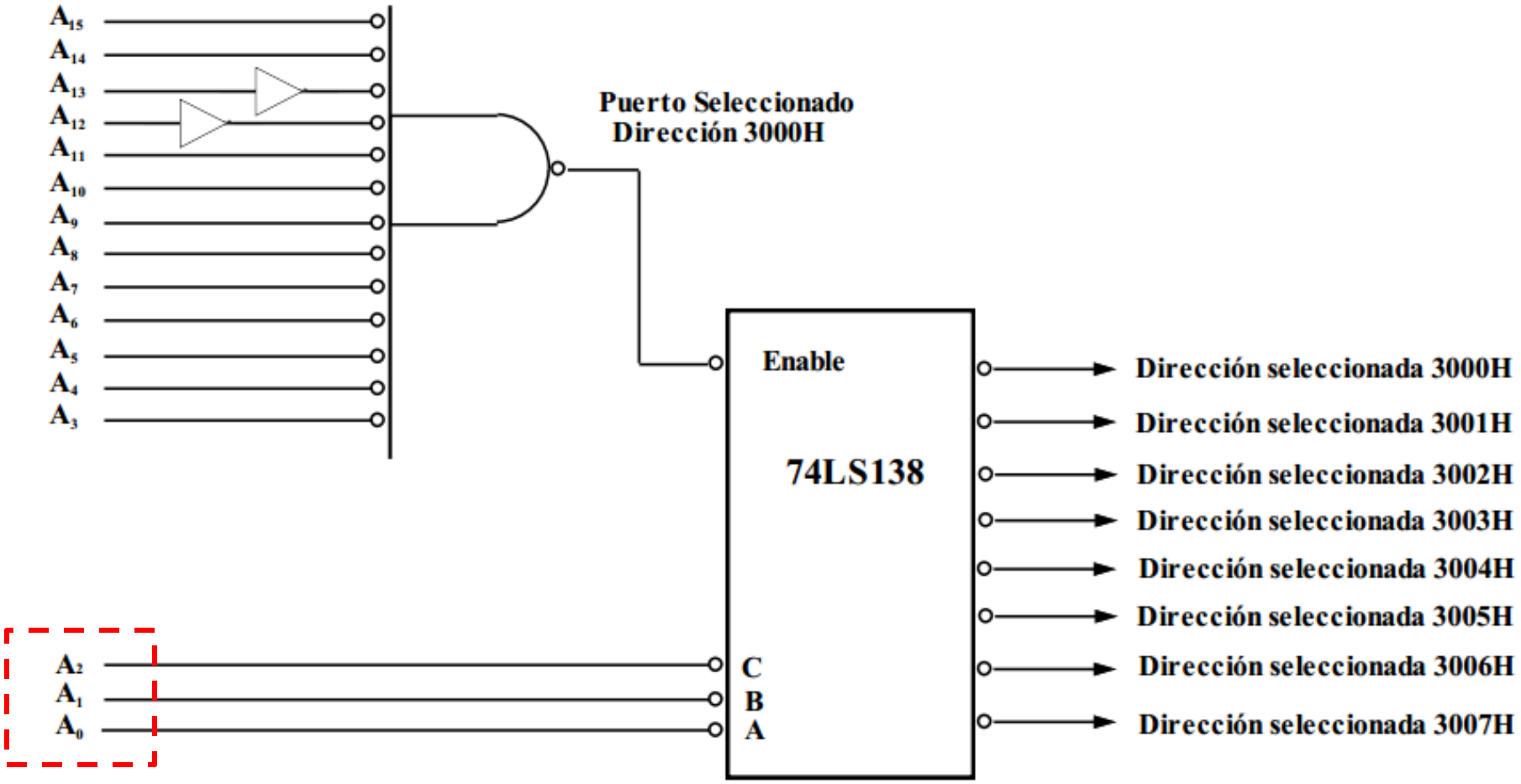


Figura 9. Decodificador múltiple utilizando un IC 74LS138

Organización de un sistema computarizado

- **Puertos de Salida**

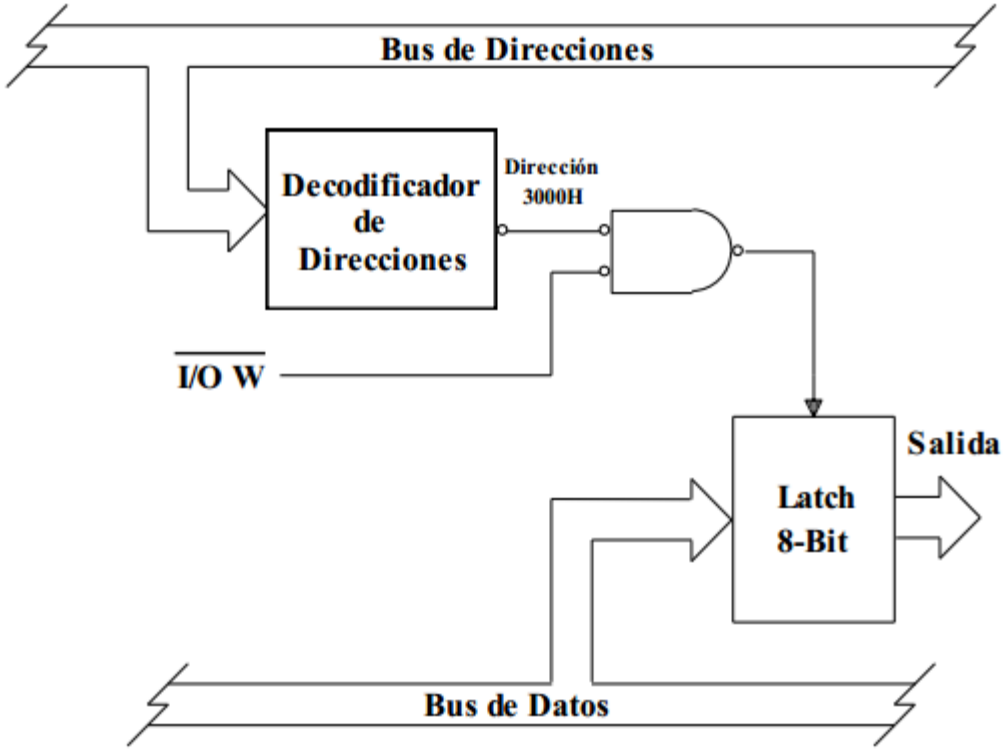


Figura 7. Puerto de salida decodificado en la dirección 3000H.

Organización de un sistema computarizado

- **Puertos de Entrada**

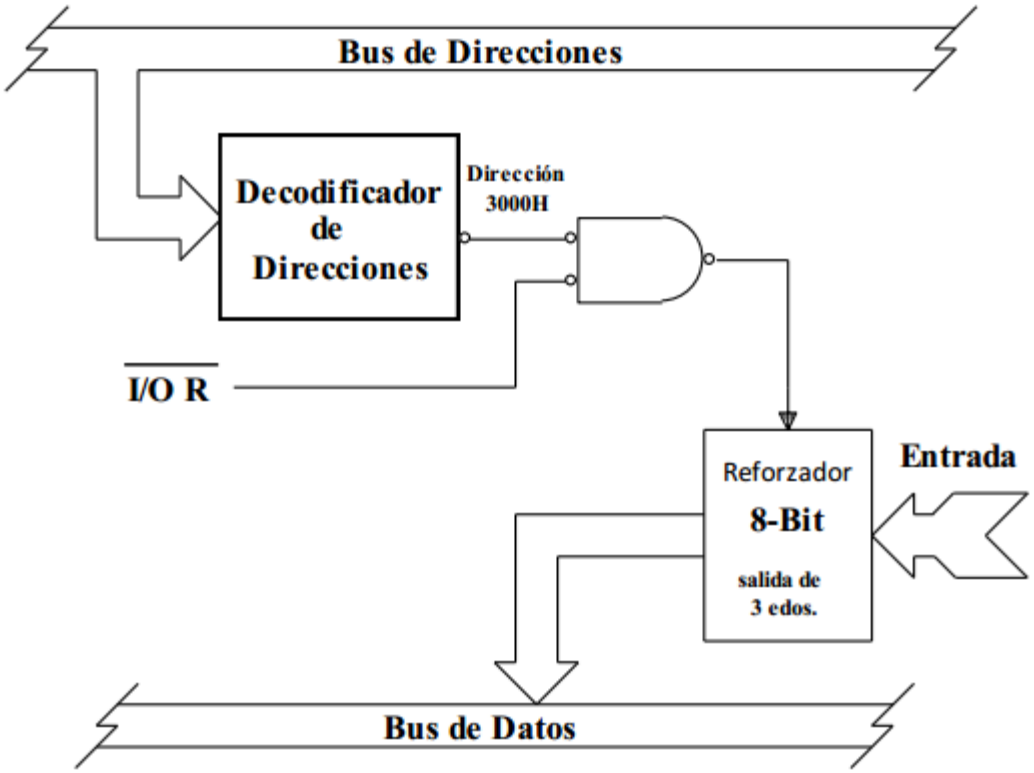


Figura 8. Puerto de entrada decodificado en la dirección 3000H.

Organización simplificada del CPU

Organización simplificada del CPU

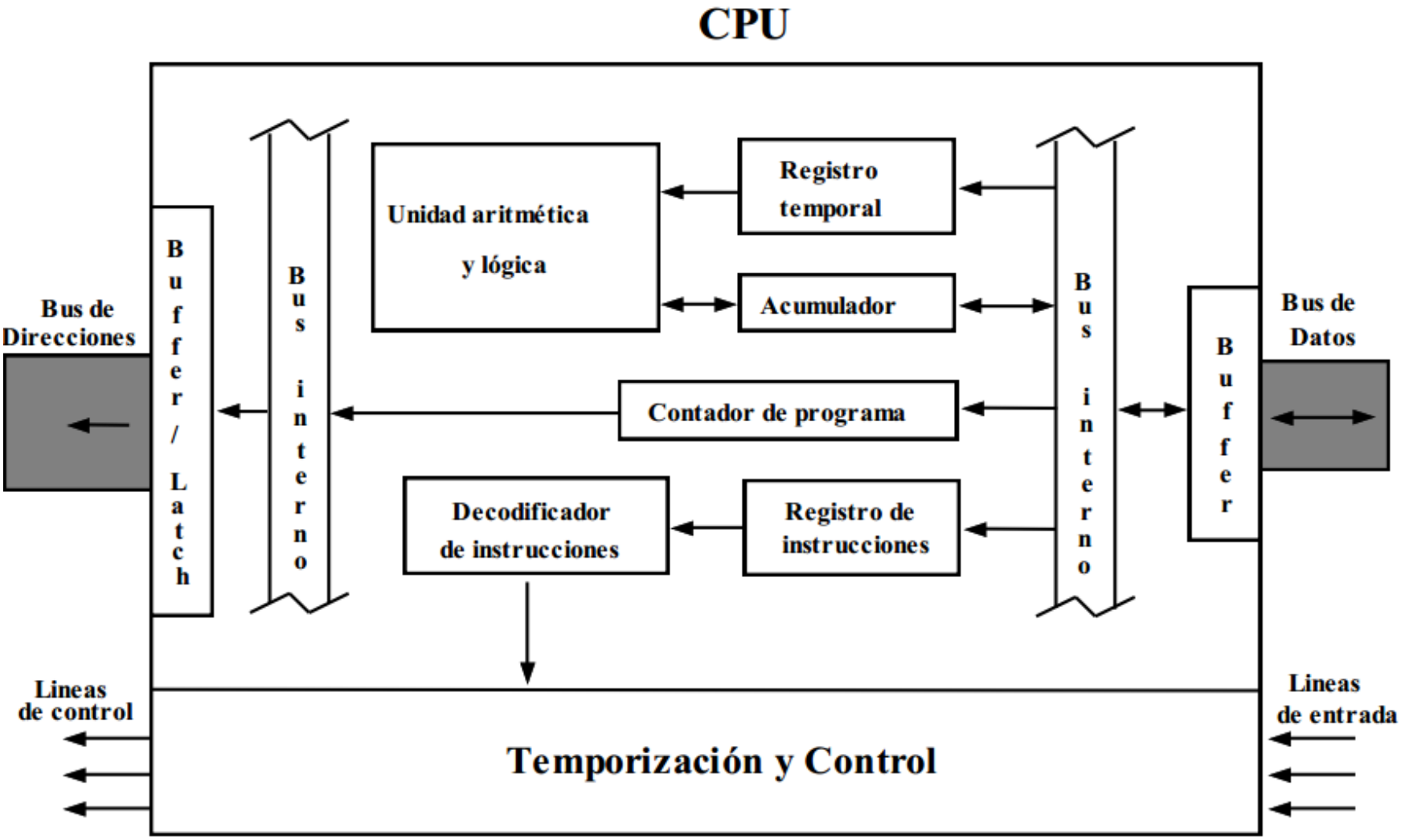
Las principales funciones del CPU son:

- Seleccionar, decodificar y ejecutar instrucciones de programa en el orden adecuado.
- Transferir datos hacia y desde la memoria, y hacia y desde las secciones de E/S.
- Responder a interrupciones externas.
- Proporcionar las señales de control y de tiempo necesarias para la totalidad del sistema.

Organización simplificada del CPU

La mayoría de los CPU de los microprocesadores contienen como mínimo los elementos mostrados en la siguiente figura:

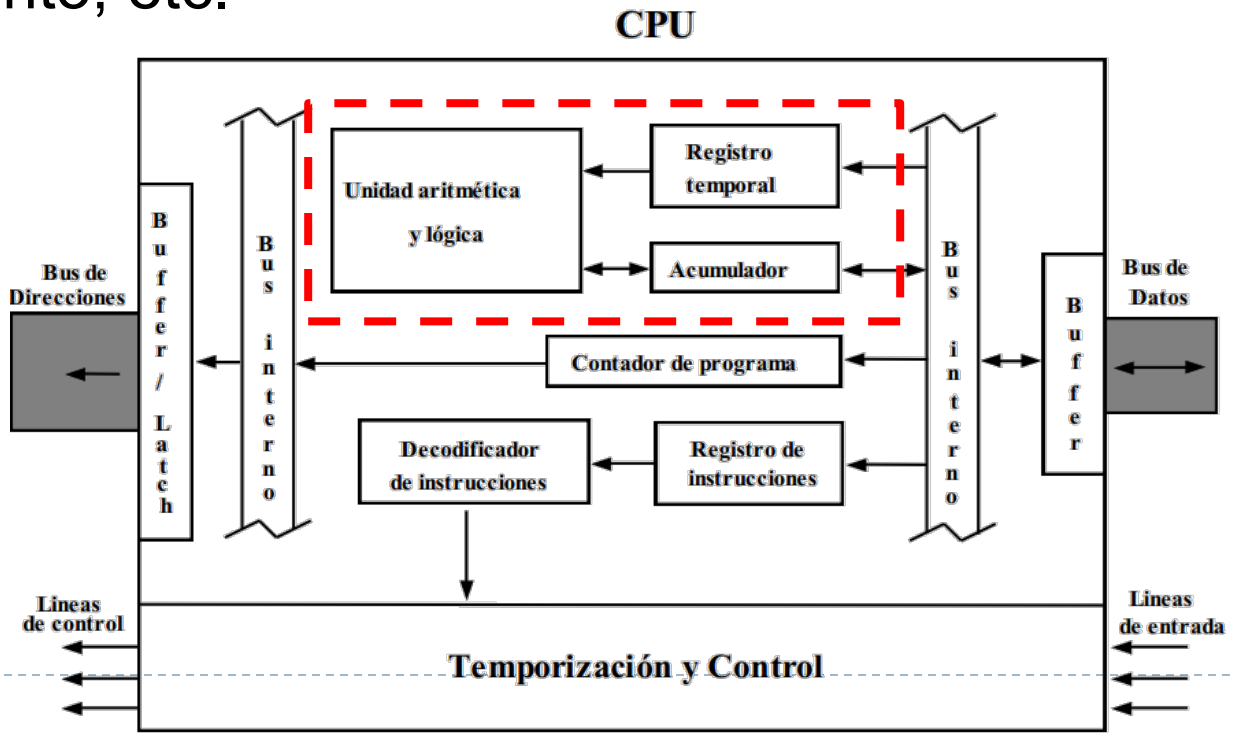
Organización simplificada del CPU



Organización simplificada del CPU

Unidad de Aritmética y Lógica (ALU)

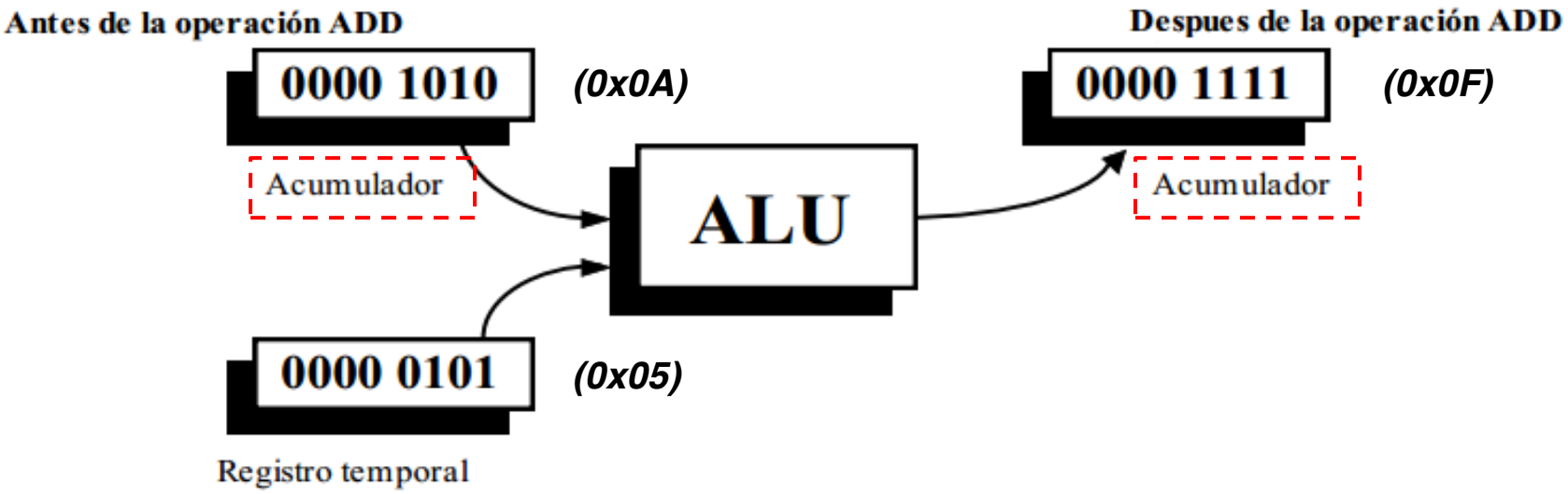
Realiza operaciones tales como suma, corrimiento circular, comparación, incrementar, decrementar, negar, AND, OR, XOR, complemento, etc.



Organización simplificada del CPU

Unidad de Aritmética y Lógica (ALU)

Ejemplo del procedimiento de suma por medio de la ALU:



a) Ejecución de la instrucción de sumar en la ALU

Organización simplificada del CPU

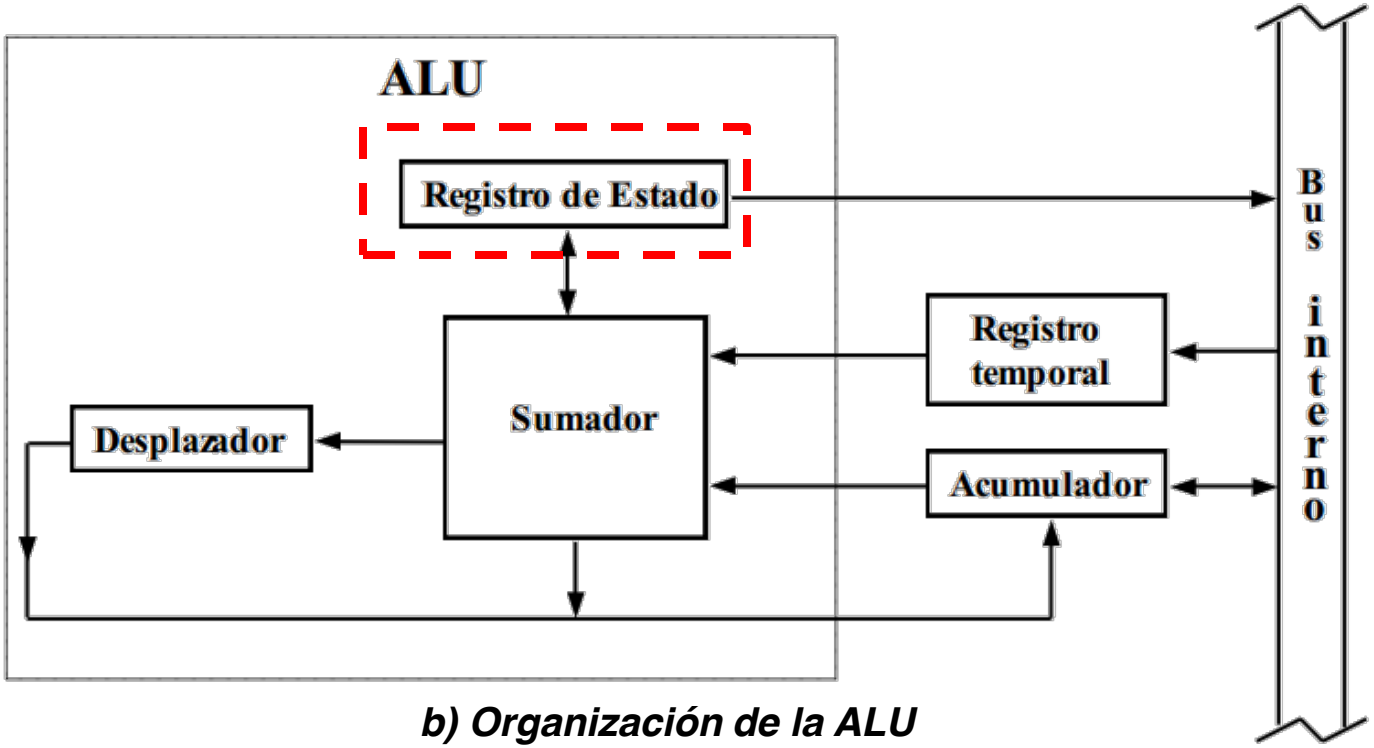
Unidad de Aritmética y Lógica (ALU)

Como se ve en la *figura a)*, el contenido del acumulador (0x0A) se suma al contenido del registro temporal (0x05) luego el resultado (0x0F) se coloca de regreso en el **acumulador**.

Organización simplificada del CPU

Unidad de Aritmética y Lógica (ALU)

Ejemplo del procedimiento de suma por medio de la ALU:



b) Organización de la ALU

Organización simplificada del CPU

Unidad de Aritmética y Lógica (ALU)

Registro de estado:

El registro de estado o también conocido como las **banderas**, incluyen indicadores de si el resultado de la operación es cero, o si es un valor negativo, si hay un acarreo, etc.

Las banderas sirven para la toma de decisiones cuando se utilizan instrucciones subsecuentes de bifurcación (tales como instrucciones condicionales o ciclos).

Organización simplificada del CPU

Sección de Control de Tiempo

Cada instrucción de un programa puede dividirse en las etapas de **seleccionar**, **decodificar** y **ejecutar**.

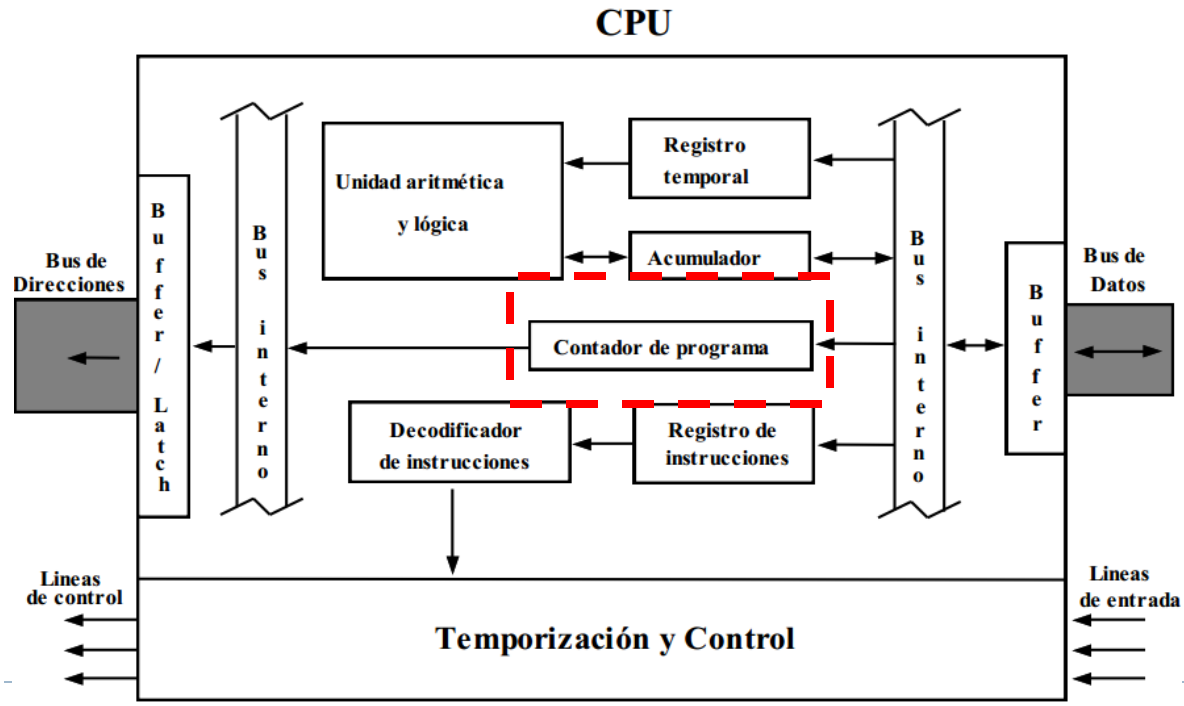
A su vez cada una de estas etapas puede ser subdivididas en una serie de pequeños pasos conocidos como **MICROPROGRAMAS**.

El microprograma de cada instrucción reside en la sección de **decodificación de instrucciones** y es ejecutado por la sección de control y de tiempo del CPU.

Organización simplificada del CPU

Contador de Programa

Es un registro de 16 bits que contiene la dirección de la siguiente instrucción que será seleccionada de la memoria.



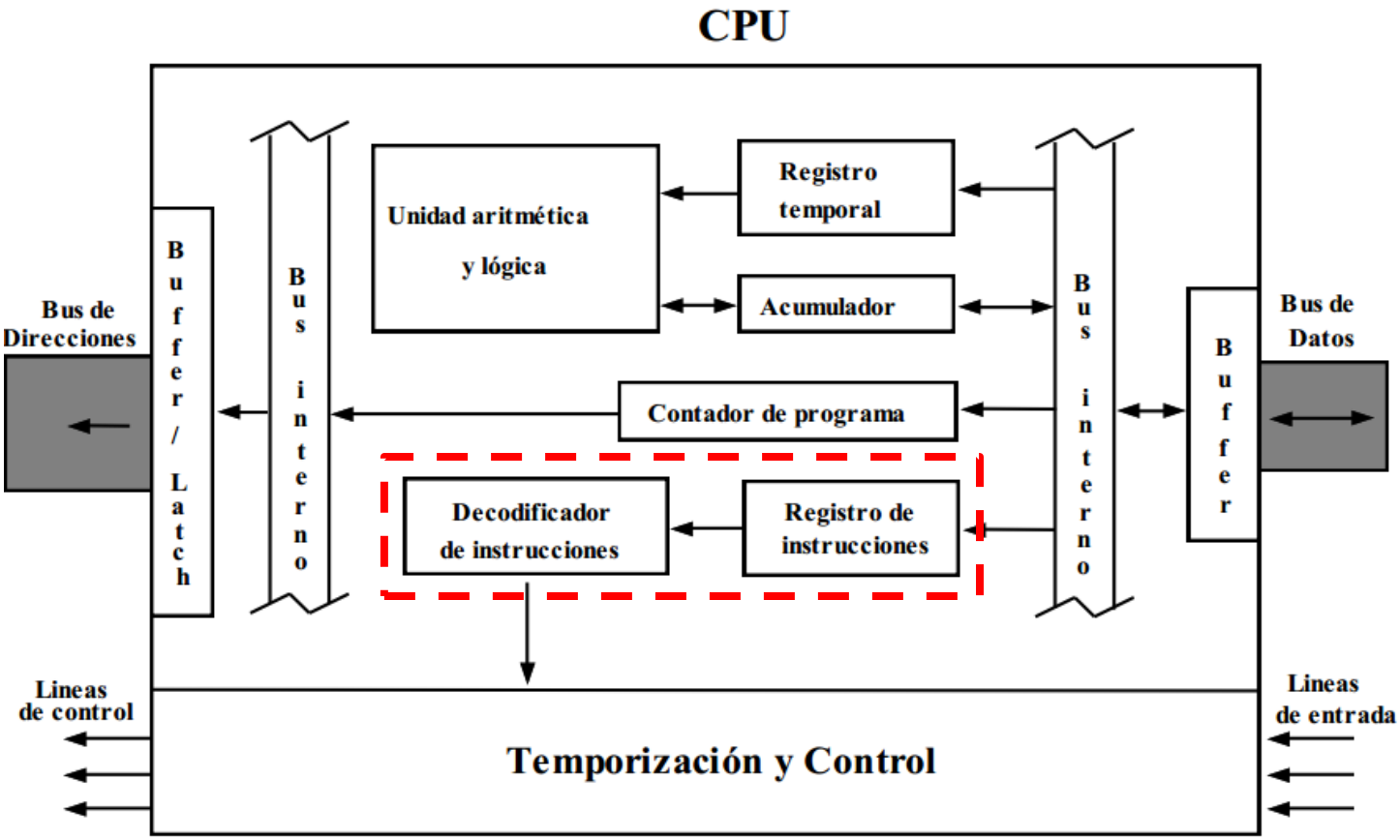
Organización simplificada del CPU

Las instrucciones están compuestas por un **código de operación (*opcode*)** y, en algunos casos, también por datos o registros sobre los cuales va a trabajar.

Al traer una instrucción de memoria, ésta es colocada en el **registro de instrucción** por parte de la sección de control del CPU.

El código de operación es entonces interpretado por el **decodificador de instrucciones,** luego éste dirige la sección de control y de tiempo que el microprograma va a seguir para ejecutar la instrucción específica.

Organización simplificada del CPU



El CPU paso a paso

Ejecución de una instrucción por el CPU

Programa

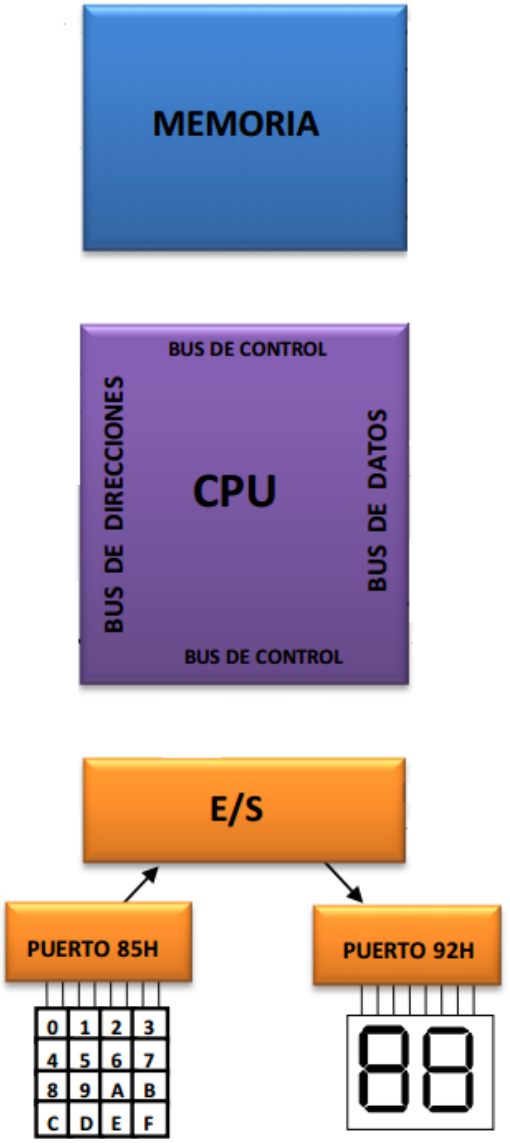
1. Leer un dato del puerto **85H** (Teclado)
2. Sumar **07H** a ese dato
3. Sacar el resultado por el puerto **92H** (7-Seg)

MEMORIA

Dirección	Contenido	
00h	30h	INPORT A, 85h
01h	85h	
02h	80h	ADD A, 07h
03h	07h	
04h	32h	OUTPORT 92h, A
05h	92h	

PC → Contador de Programa (Program Counter)

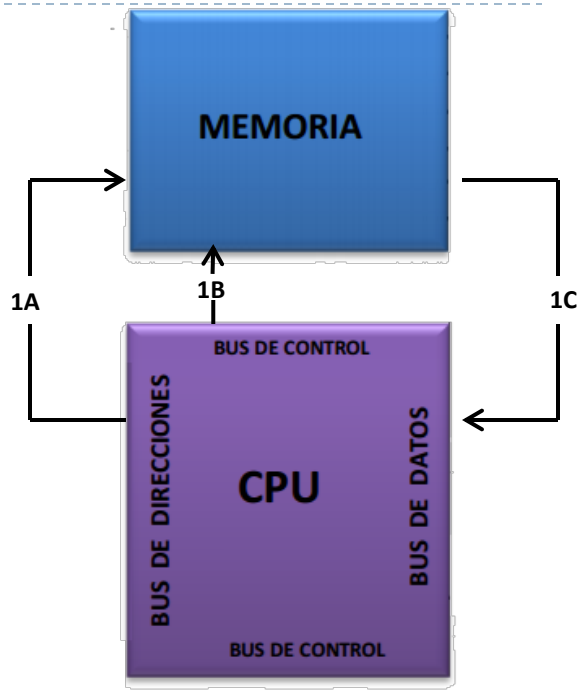
Código de Operación de la instrucción



Ejecución de una instrucción por el CPU (**Obtener** instrucción)

MEMORIA

	Dirección	Contenido	
PC →	00h	30h	INPORT A, 85h
PC →	01h	85h	
	02h	80h	ADD A, 07h
	03h	07h	
	04h	32h	OUTPORT 92h, A
	05h	92h	



1A : El CPU pone el contenido de PC (0x00) en el bus de Direcciones.

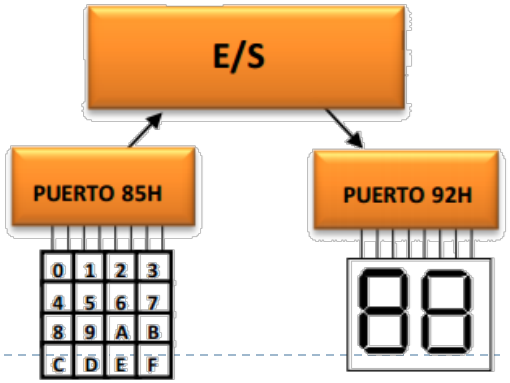
1B : El CPU pone en el bus de Control la señal de lectura.

1C : La memoria pone el dato almacenado en esa dirección (0x30) en el bus de Datos.

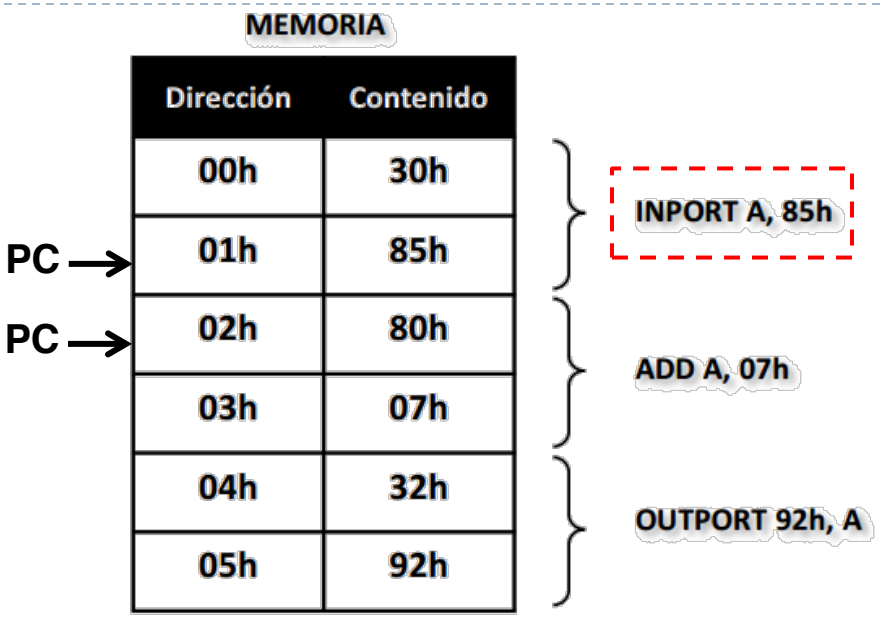
El CPU toma el dato colocándolo en el registro de instrucción.

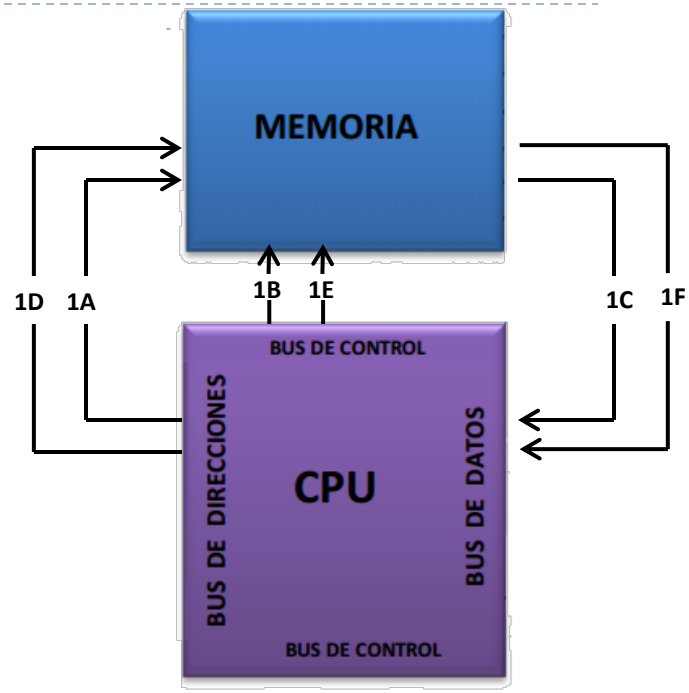
Se incrementa el valor de PC.

El CPU decodifica la instrucción, determinando que es una instrucción de E/S y que requiere leer otro dato de memoria.



Ejecución de una instrucción por el CPU (**Obtener** instrucción)





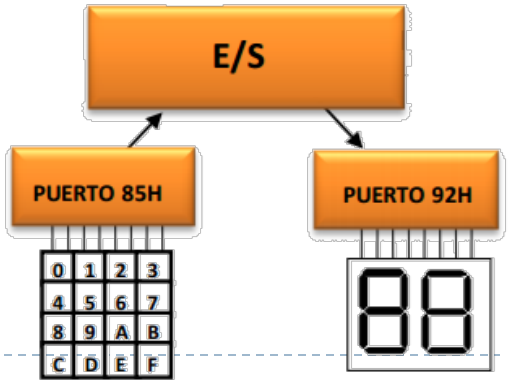
1D : El CPU pone el contenido de PC (0x01) en el bus de Direcciones.

1E : El CPU pone en el bus de Control la señal de lectura.

1F : La memoria pone el dato almacenado en esa dirección (0x85) en el bus de Datos. El CPU toma el dato.

Se incrementa el valor de PC.

La instrucción esta completa.

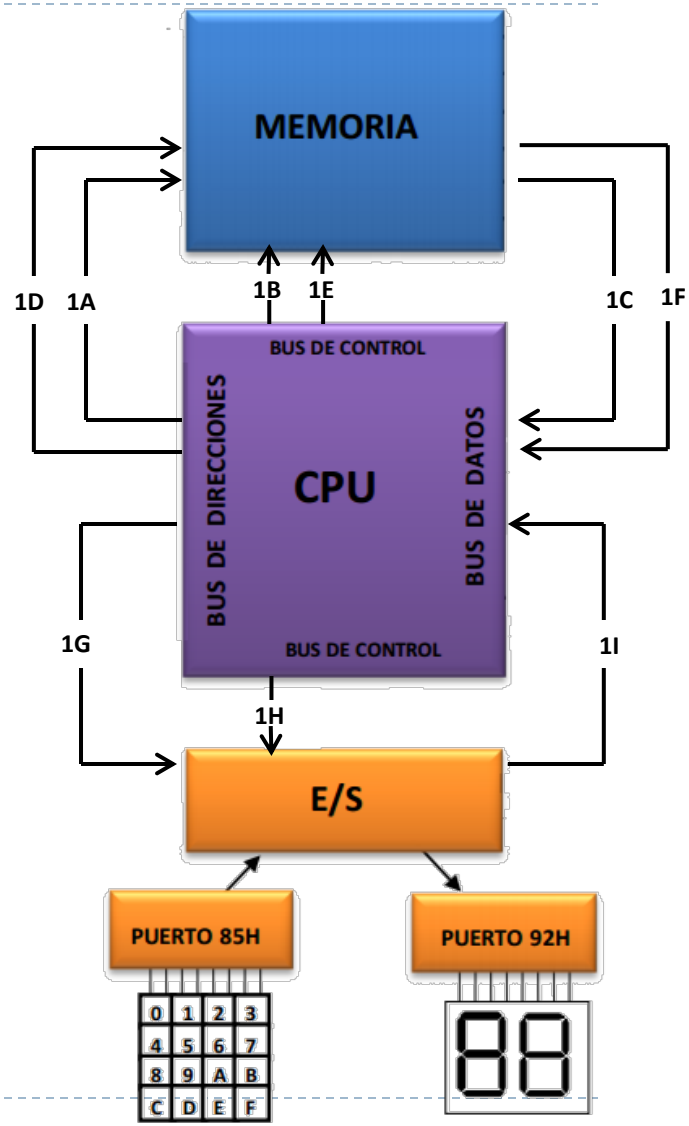


Ejecución de una instrucción por el CPU (**Ejecución** de la instrucción)

MEMORIA

Dirección	Contenido	
00h	30h	INPORT A, 85h
01h	85h	
02h	80h	ADD A, 07h
03h	07h	
04h	32h	OUTPORT 92h, A
05h	92h	

PC →



Ejecución de la instrucción:

1G : El CPU pone en el bus de Direcciones la dirección del puerto de E/S (0x85).

1H : El CPU pone en el bus de Control la señal de lectura.

1I : Se coloca el dato del puerto en el Bus de Datos.

El CPU toma el dato, colocándolo en el registro A

Se ha terminado de ejecutar la instrucción.

Implementación de Programas en Microprocesadores y Microcontroladores

Un **traductor** es un programa que convierte un programa escrito en un lenguaje a otro lenguaje distinto.

Al lenguaje en que esta escrito el programa original se le denomina **lenguaje fuente**, y al lenguaje al que se traduce se le conoce como **lenguaje objeto**.

Si se contara con un procesador que ejecutara directamente los programas escritos en lenguaje fuente, no sería necesaria la traducción.

Los programas tanto el fuente como el objetos son equivalentes desde el punto de vista funcional, es decir, producen los mismos resultados.

Los traductores se pueden dividir en dos grupos según la relación que existen entre el lenguaje fuente y el objeto:

Ensamblador → El lenguaje fuente es una representación simbólica de un lenguaje numérico (el **Lenguaje Máquina**). Al lenguaje fuente se le conoce como **Lenguaje Ensamblador**.

Compilador → El lenguaje fuente es un lenguaje mucho mas complejo que el lenguaje objeto, es decir una instrucción en el lenguaje fuente es traducida a varias instrucciones de lenguaje objeto.

Una razón para utilizar el lenguaje ensamblador en lugar de programar en lenguaje máquina, es que es mucho mas fácil programar en lenguaje ensamblador.

Es mucho mas fácil programar utilizando mnemónicos para instrucciones que usar directamente números.

Ejemplo:

Sumarle un 5 al acumulador:

En lenguaje ensamblador: ADD AX,5

En lenguaje máquina: 0x5005

Lenguajes de alto nivel

El concepto de lenguaje de alto nivel se define como un lenguaje orientado a la solución de problemas, además están diseñados para permitir al programador concentrarse en la lógica del problema a resolver, liberándolo de la necesidad de un conocimiento profundo de lenguaje máquina de la computadora.

El termino alto nivel significa que están orientados hacia la gente, en contraste con el lenguaje ensamblador de bajo nivel el cual esta orientado hacia la máquina.

Lenguajes de alto nivel

Características que identifican un lenguaje de alto nivel:

- **Utiliza una notación especial orientada al problema a resolver:**

Lenguaje de alto nivel:

```
R=X*Y - Z
```

Lenguaje ensamblador:

```
MOV AL,[X]
MOV CL,[Y]
MUL CL
SUB AX,[Z]
MOV [R],AX
```

Lenguajes de alto nivel

- **No se requiere conocimiento del código máquina.**
- **Facilidad para procesar el programa en otras computadoras (transportabilidad)**
- **Expansión de las instrucciones**

Lenguajes de alto nivel

Ventajas de los lenguajes de alto nivel:

- Facilidad de aprendizaje.
- Facilidad de utilizarlo.
- Facilidad para documentación y el mantenimiento.
- Facilidad de depuración.
- Facilidad para transportarlo a otra computadora o microcontrolador.
- Reducción del tiempo total para el desarrollo de programas.

Lenguajes de alto nivel

Desventajas de los lenguajes de alto nivel:

- Tiempo necesario para la compilación.
- No siempre se tiene el programa objeto en forma óptima.

Lenguaje ensamblador y Lenguajes de alto nivel

Subrutinas y Macros

Lenguaje ensamblador y Lenguajes de alto nivel

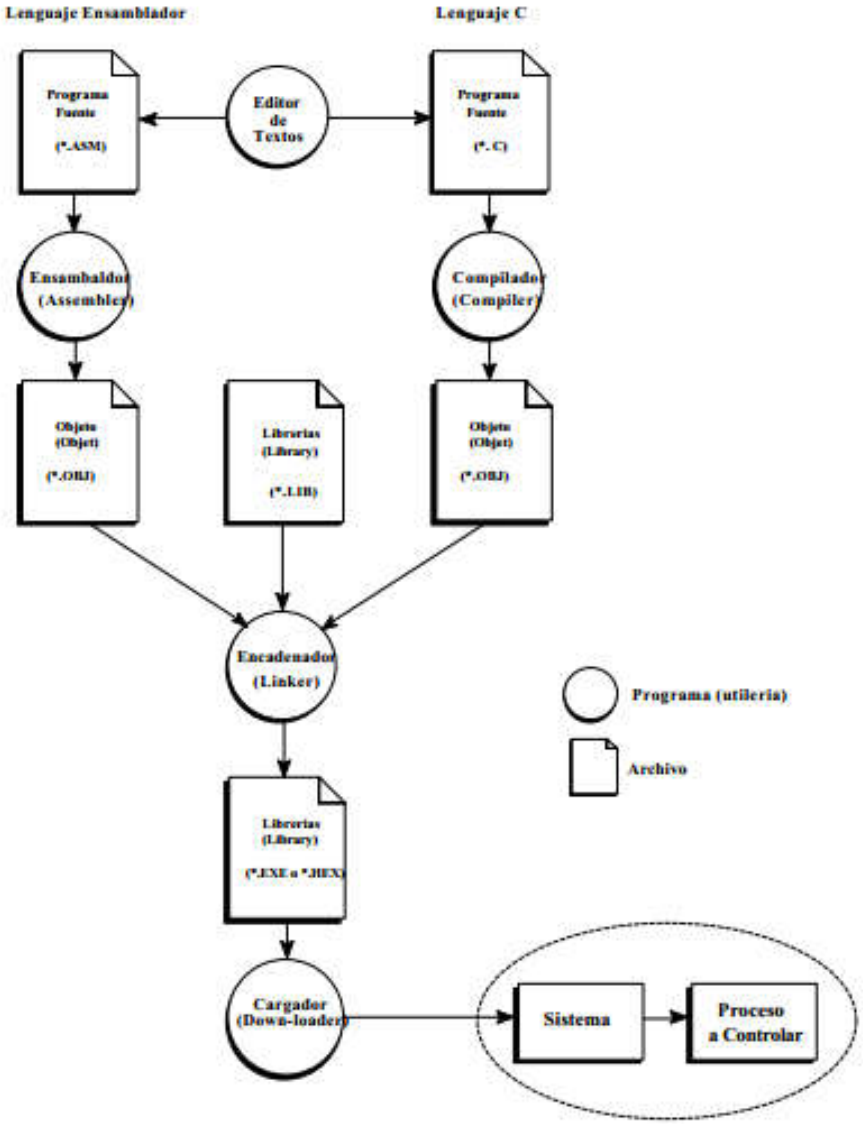


Figura 2.7 Diagramas de bloques de la conversión de programas de un código fuente a un programa ejecutable por el microcontrolador.

Números de punto flotante

Para representar números reales, muchos formatos de punto flotante emplean notación científica y usan un cierto número de bits para representar una ***mantisa*** y un número pequeño de bits para representar el ***exponente***. Esto resulta en que los números flotantes pueden representar a un número con solo una cantidad específica de dígitos significativos.

Notación científica:

$$n = f \times 10^{e}$$

f es la fracción o mantisa

e es el exponente

Ejemplos:

$$3.14 = 0.314 \times 10^{1} = 3.14 \times 10^{0}$$
$$0.000001 = 0.1 \times 10^{-5} = 1.0 \times 10^{-6}$$
$$1941 = 0.1941 \times 10^{4} = 1.941 \times 10^{3}$$

Estándar IEEE 754

En 1985 se estableció el estándar IEEE 754 para la implementación de números flotantes.

El estándar define tres formatos:

- Precisión sencilla
- Precisión doble
- Precisión extendida

Estándar IEEE 754

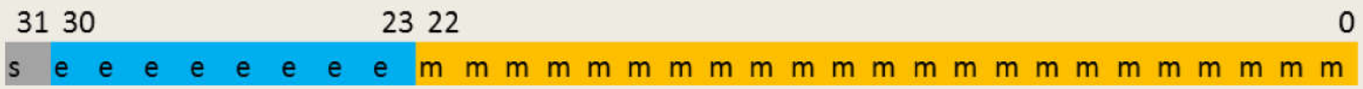
En este estándar la **fracción** consiste en un bit 1 y un punto binario implícito (*1.*), y 23 o 52 bits arbitrarios.

- Si todos los bits de la fracción son ceros, la fracción tiene el valor numérico de 1.0;
- Si todos son uno, la fracción es numéricamente un poco menos que 2.0.

A fin de evitar confusiones con una fracción convencional, se le denomina ***significando*** en vez de fracción o mantisa.

Precisión sencilla

Tiene un tamaño de dato de **32 bits** y su organización es la siguiente:



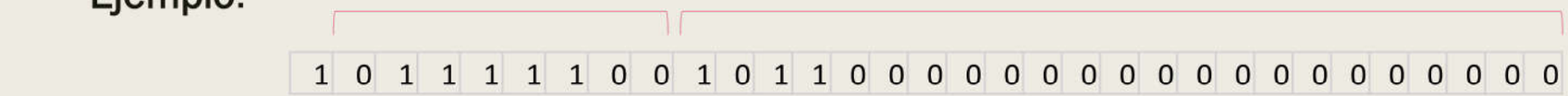
Donde:

- 1 bit corresponde al **signo.** El 0 es positivo y 1 es negativo.
- 8 bits corresponden al **exponente** el cual usa exceso 127.
- 23 bits corresponden al **significando.**

Intervalo decimal que puede representar: aprox. 10^{-38} a 10^{38}

Precisión sencilla

Ejemplo:



Signo = 1 = **número negativo**

Exponente: 01111100_2 = 124 – 127 = **-3**

Significando: $10110000000000000000000_2$

Conversión a decimal:

1.1011_2 (El *1.* es implícito en el estándar)

$1.1011_2 = (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) + (1 \times 2^{-4}) = 1.6875_{10}$

$1.6875_{10} \times 2^{-3} = 0.2109375$

Valor: - 0.2109375

Precisión doble

Tiene un tamaño de dato de **64 bits** y su organización es la siguiente:

63 62 … 52 51 … 0

s e e e e e e e e e e e m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m m

Donde:

- 1 bit corresponde al **signo.** El 0 es positivo y 1 es negativo.
- 11 bits corresponden al **exponente** el cual usa exceso 1023.
- 52 bits corresponden al **significando.**

Intervalo decimal que puede representar: aprox. 10^{-308} a 10^{308}

Precisión extendida

Su implementación es dependiente de la aplicación.

Acotaciones por parte de IEEE 754:

- **Precisión extendida sencilla**:

  Bits de exponente: ≥ 11

  Número de bits significativos (precisión): ≥ 32

- **Precisión extendida doble**:

  Bits de exponente: ≥ 15

  Número de bits significativos (precisión): ≥ 64

Suma y resta en punto flotante

Para sumar y restar dos números en punto flotante:

- Convertir los operandos a notación científica, representando explícitamente el 1 oculto.
- Ajustar los operandos de forma que ambos tengan el mismo exponente.
- Sumar o restar las mantisas de los operandos.
- Ajustar el resultado de forma que esté ***normalizado***, esto es, que se encuentre de la forma $(1.fracción)_2$
- Convertir el valor a formato de punto flotante, remover el 1 implícito del significando.

Suma y resta en punto flotante

Ejemplo:

a = 0.25 b = 100 c = a + b

Representación en punto flotante:

a = 0 01111101 00000000000000000000000 $\rightarrow 1.0 \times 2^{-2}$

b = 0 10000101 10010000000000000000000 $\rightarrow 1.5625 \times 2^{6}$

Ajuste de los exponentes para que sean iguales:

a = 0 10000101 0.00000001000000000000000 $\rightarrow 0.00390625 \times 2^{6}$ No normalizado

b = 0 10000101 10010000000000000000000 $\rightarrow 1.5625 \times 2^{6}$

Suma y resta en punto flotante

Suma de las mantisas:

	0.00000001000000000000000	(a)
+	1.10010000000000000000000	(b)
	1.10010001000000000000000	(c)

En decimal es equivalente a hacer:
$0.00390625 + 1.5625 = 1.56640625$

Suma y resta en punto flotante

Normalización del resultado:

1.10010001000000000000000

El resultado ya se encuentra normalizado, es decir, se encuentra de la forma $(1.fracción)_2$.

Si este no fuera el caso, para normalizarlo hay que realizar corrimientos en la mantisa de forma que haya un único 1 antes del punto, y realizar los correspondientes incrementos o decrementos al exponente.

Suma y resta en punto flotante

Conversión a formato de punto flotante:

c = 0 10000101 10010001000000000000000

Se removió el 1 que se encuentra antes del punto, el cual quedará implícito.

En decimal el valor corresponde a:

$$c = 1.56640625 \times 2^6 = 100.25$$

Algoritmo para multiplicación en punto flotante

Para multiplicar dos números en punto flotante:

- Multiplicar los significandos (incluir el *1.* implícito en ambos).
- Sumar los exponentes (después de restarles a cada uno el exceso-127 o exceso-1023).
- El signo se obtiene de aplicar un xor entre los signos de los operandos.
- Convertir el valor a formato de punto flotante, remover el 1 implícito del significando, sumar el exceso al exponente.

Algoritmo para división en punto flotante

Para dividir dos números en punto flotante:

- Dividir los significandos (incluir el *1.* implícito en ambos).
- Restar los exponentes (después de restarles a cada uno el exceso-127 o exceso-1023). La resta corresponde a

  exponente del dividendo - exponente del divisor

- El signo se obtiene de aplicar un xor entre los signos de los operandos.
- Convertir el valor a formato de punto flotante, remover el 1 implícito del significando, sumar el exceso al exponente.