Universidad Francisco de Vitoria
UFV Madrid

Fundamentos de la ingeniería informática

Ingeniería de sistemas industriales

Curso 2019-2020

Sistemas Programables I: Procesadores

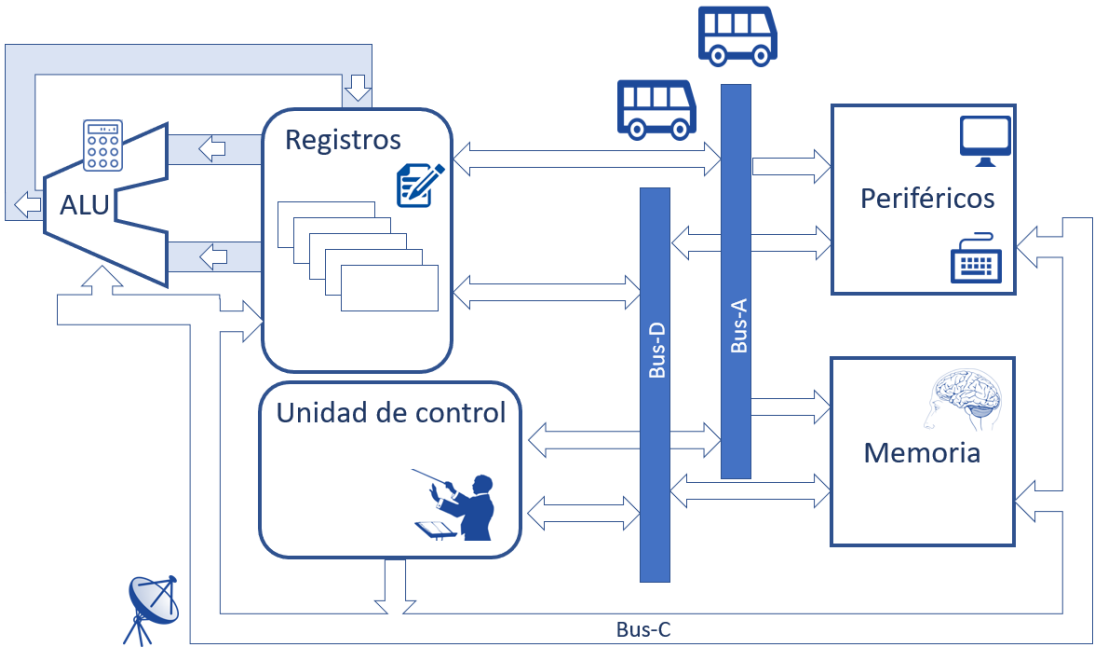
1. Introducción

Los procesadores son sistemas secuenciales capaces de realizar operaciones sencillas combinacionales sencillas de forma repetitiva. La combinación y secuenciación de estas operaciones pueden convertirse en actividades tan complejas como las vinculadas con la inteligencia artificial.

2. Arquitectura

Un procesador es una máquina electrónica compuesta por:

- Unidad de memoria (memoria principal)
- Unidad de entrada/salida (periféricos).
- Unidad aritmeticológica.
- Banco de registros
- Unidad de control



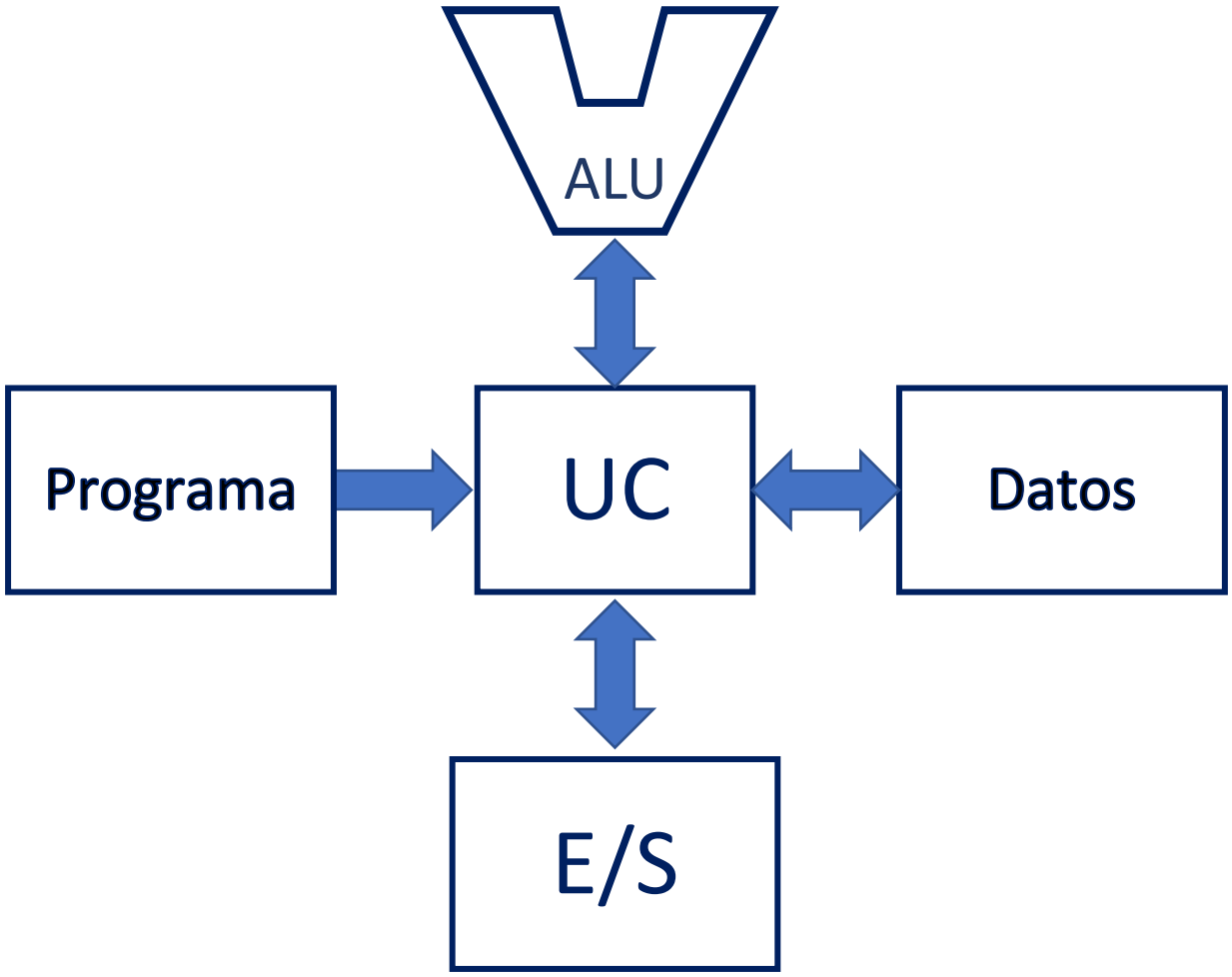
Al conjunto de Unidad Central y Unidad Aritmético Lógica se denomina C.P.U. (Central Process Unit) Unidad central de proceso.

2.1. Memoria

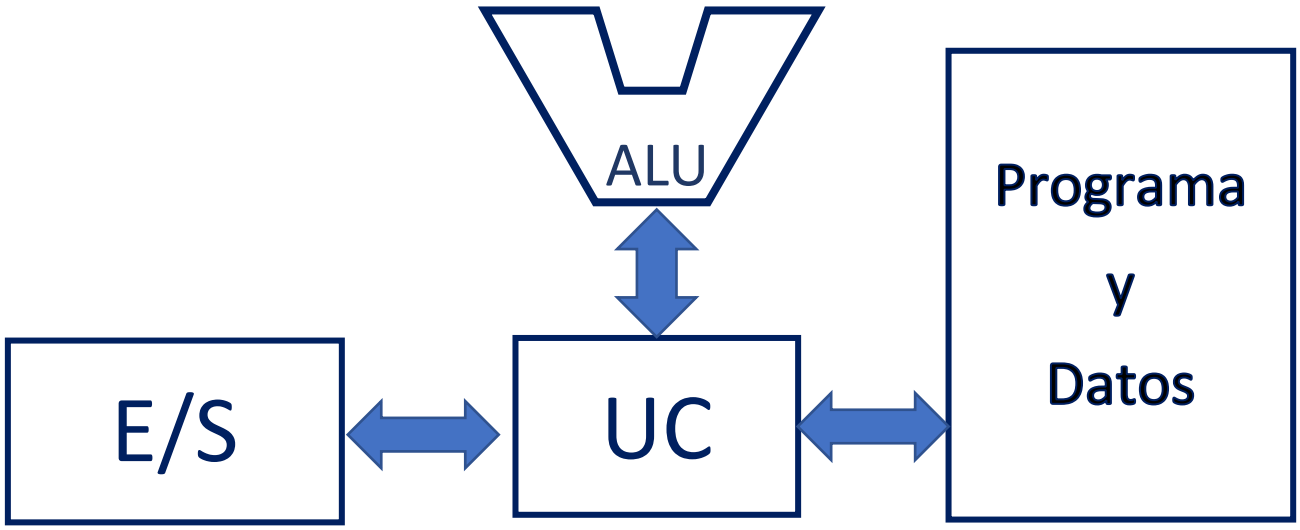
La memoria es el dispositivo encargado de almacenar información de cualquier tipo en forma digital. De forma genérica son dos los tipos de información que se puede almacenar en la memoria: programas y datos.

Un programa es una secuencia de datos binarios que interpretados como instrucciones indican a la CPU cuales son las acciones a realizar y su secuencia.

En razón de estos dos tipos de datos se han desarrollado dos tipos de arquitectura. En la arquitectura Harvard la memoria de programa es independiente de la memoria de datos.



Mientras que en la arquitectura von Neumann, el programa y los datos comparten el espacio de memoria.



Por un lado, la arquitectura Harvard no sólo el programa y los datos no comparten espacio de memoria sino que tampoco comparten el bus por el que la información circula desde o hacia la

unidad central, lo que permite que la operación de lectura de programa y la de lectura o escritura de datos se realicen simultáneamente, y al mismo tiempo permite diseñar el tamaño de la palabra de programa (tamaño de la instrucción) del modo más eficiente dado que es independiente de del tamaño de la palabra de la memoria de datos.

Por otro, la arquitectura von Neumann permite mucha más flexibilidad; el programa es posible tratarlo como una secuencia de datos y por tanto no sólo leerlo sino también escribirlo.

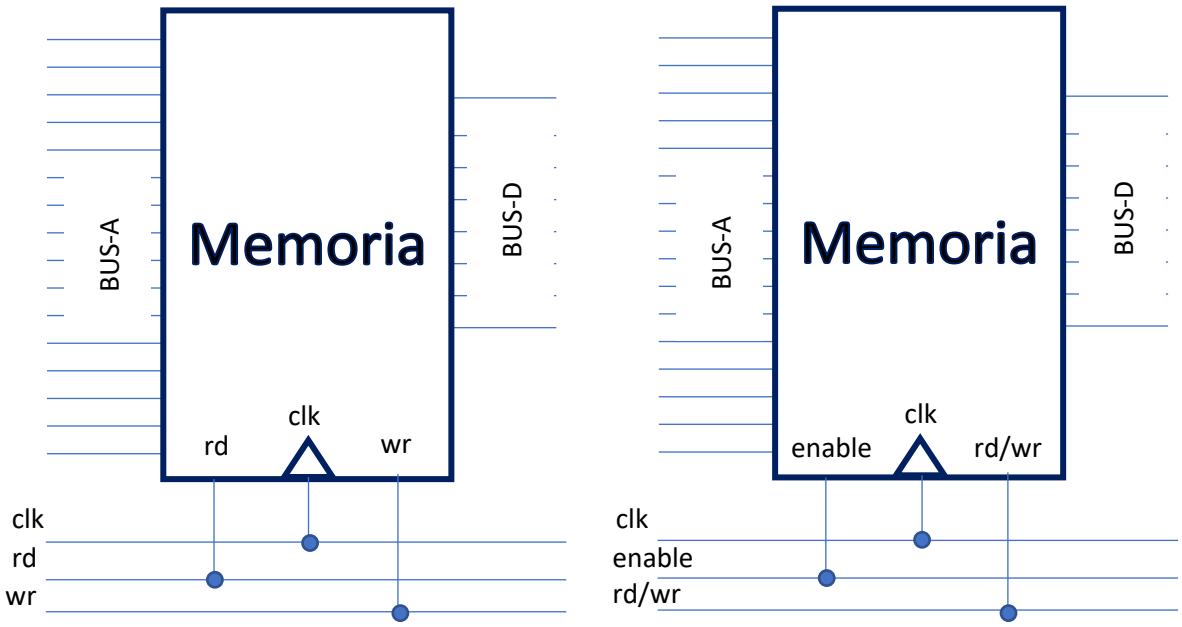
De ahí que sea la arquitectura von Neumann la preferida en procesadores de propósito general, ya que los programas deben poder ser cargados en memoria de forma ágil, cosa que no es posible en la Harvard. Simetricamente esta última es utilizada en procesadores de propósito específico donde el programa no se modifica.

Cualquiera que sea la arquitectura una CPU accede a la lectura o escritura de la memoria indicando la posición de la palabra buscada (dirección). Esta dirección se suministrada por los hilos del Bus de direcciones (BUS-A).

Los fabricantes de memorias, junto con el bus de direcciones, ofrecen líneas de control (entradas para gestionar las operaciones) .

Una opción típica son las señales : clk, rd, wr (reloj, lectura, escritura). Para que se ejecute una operación de escritura debe estar activa la señal wr en el flanco activo del reloj. Para que se ejecute una operación de lectura debe estar activa la señal rd en el flanco activo del reloj.

Otra alternativa son las señales: clk, rd/wr, enable. Con esta configuración para realizar una operación de escritura debe estar activa la señal enable e inactiva la señal rd/wr en el flanco activo del reloj. Para realizar una operación de lectura deben estar activas tanto la señal enable como la rd/wr en el flanco activo del reloj.



2.2. Entrada / salida

La unidad de entrada salida es la unidad encargada de conectar al ordenador con el mundo exterior. El manejo de cada dispositivo de entrada/salida se realiza mediante la lectura y escritura en registros específicos de cada dispositivo. Alguno de los valores que la CPU escribe esos registros son ordenes (controles) que obligan al dispositivo a realizar alguna acción, otros son los datos que se deben ser enviados. Algunos de los valores que la CPU lee en esos registros son datos que se reciben del exterior, otros son información del estado del dispositivo.

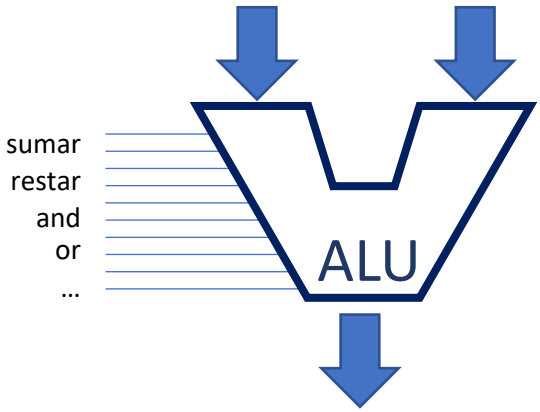
Todos los registros, pese a que funcionalmente son distintos de la memoria de datos, son idénticos desde el punto de vista operativo, por lo que algunas CPU's obligan a que el espacio de direcciones de los datos (o programa) y los registros sea compartido.

2.3. Aritmético-lógica

La Unidad Aritmético-Lógica es la circuitería encargada de realizar las operaciones matemáticas y booleanas del repertorio de instrucciones, como pueden ser en una CPU básica: Sumar, Restar, And, Or, Xor, Rotación, Desplazamiento, Negación y Complemento.

Esta unidad sólo realiza operaciones unarias o binarias, y junto al resultado activa/desactiva un conjunto de bits (flags) que señalan características interesantes del resultado, como puede ser si el resultado ha sido 0 o negativo o si se ha producido overflow o acarreo y algunos otros flags que el fabricante considera oportunos para el aprovechamiento de la arquitectura.

La UC administra la UAL mediante las múltiples señales de control (una por cada una de las operaciones básicas que ésta puede realizar). La UC coloca el o los dos datos desde donde proceda en la/s entrada/s de la UAL y activa la señal correspondiente a la operación que se quiere ejecutar; finalmente cuando la UAL ha obtenido el resultado la UC lo traslada al destino programado.



2.4. Banco de registros

Para su operativa, todas las CPU disponen de un conjunto de registros (palabras de memoria) que almacenan cierto tipo de información necesaria para la ejecución de los programas.

2.4.1. Registros de control y estado

Registros dedicados a controlar el funcionamiento o informar sobre el estado del mismo. Los más comunes son:

- Registro de instrucción (IR) donde se almacena la instrucción de programa que se va a ejecutar inmediatamente.
- Contador de programa(PC), puntero que señala a la próxima instrucción que deberá ejecutarse.
- Registro de estado (SR), donde se anotan los flags de la UAL.

2.4.2. Registros de propósito específico

Registros en los que cada uno de ellos tiene una función especial dentro de la arquitectura del ordenador. En algunos casos son registros dedicados a almacenar exclusivamente direcciones, en otros esas direcciones son todavía más específicos y se utilizan para apuntar a los operandos o a los resultados. Algunas CPUs tienen registros de índice, que no son directamente direcciones si no que se combinan con direcciones para obtener la posición del registro.

Tal vez los registros de propósito específico más populares sean el puntero de pila y el acumulador.

El puntero de pila (SP, stack pointer) es un registro de direcciones que permite organizar una pila sobre la memoria principal.

El registro acumulador (A) es un registro de datos que almacena el resultado del cálculo realizado por la UAL.

2.4.3. Registros de propósito general

Son registros que pueden utilizarse para casi cualquier funcionalidad, pueden contener una dirección, un dato, un índice, etc.

2.4.4. Tipos de CPU

Los tipos de registro que contiene una CPU la caracterizan. Pueden ser CPUs de acumulador y/o con registros de propósito general y/o específico.

2.5. Unidad de Control

La unidad central, es el director de orquesta de las diversas funcionalidades que realiza el hardware del ordenador. Consigue que las acciones individuales de cada unidad se realicen de forma sincronizada y ordenada, siguiendo la partitura que constituye el programa. El sincronismo se obtiene en base a una señal de reloj que marca los instantes en que deben suceder los cambios.

La unidad de control es un sistema secuencial que genera órdenes, a las otras unidades, mediante la activación de señales.

Existen dos tipos unidad de control: cableada y microprogramada.

Las unidades de control cableadas corresponden a un circuito secuencial que implementa una máquina de estados.

Las unidades de control microprogramadas tienen almacenadas las secuencias de ordenes en una memoria cuyas posiciones son leídas sucesivamente y con ellas activadas las señalizaciones de las otras unidades.

2.5.1. Funcionamiento

Ejecutar un programa implica ejecutar cada una de las instrucciones una a una secuencialmente, con excepción de los casos de ejecución paralela en el que varias instrucciones se ejecutan simultáneamente.

Cada instrucción, dependiendo del tipo de que se trate, implica realizar algunas de las siguientes tareas

1. Lectura de la instrucción.

   Petición a memoria de la siguiente instrucción a ejecutar, cuya dirección se encuentra en el PC, de paso se incrementa el contenido del PC para que apunte a la siguiente instrucción.

   La dirección de la siguiente instrucción a ejecutar se encuentra en el registro PC.

   Esa dirección se coloca en el bus de direcciones y se activa la operación de lectura de la memoria

   Cuando el ciclo de memoria concluye y la memoria ha depositado el dato en el BUS-D, la unidad de control guarda la instrucción en el IR.

2. Descodificación

   Se interpreta el contenido del IR: con ello se determina el tipo de instrucción y la ubicación de los operandos

3. Lectura de operandos

   En su caso se accede a memoria o registros para obtener los valores a operar.

4. Ejecución

   Se realiza la operación indicada en la instrucción

5. Escritura del resultado

   En su caso el resultado de la operación se almacena en memoria o registros.

Estas tareas suelen realizarse en etapas, donde cada etapa dura un ciclo de reloj. El número de etapas y la duración de cada ciclo son característicos de cada procesador.

2.6. Señales

Las señales de orden que emite la unidad de control pueden clasificarse en dos tipos:

- Señales de flanco, aquellas que son activas solamente en el cambio de estado. En estos apuntes el grafismo que asociaremos a esta señal será una flecha de trazo discontinuo. Pueden ser activas en flanco de subida (etiqueta de la señal sin negar), o pueden serlo en el flanco de bajada (etiqueta de la señal negada).

RESo ⇢ *(wr* ⇢

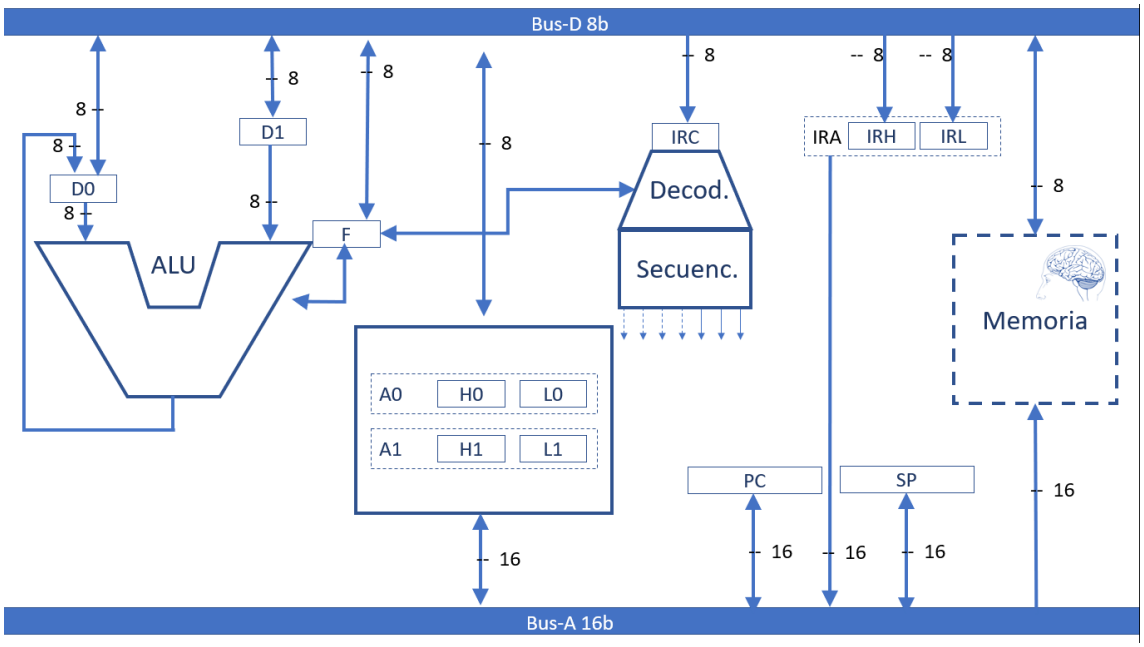
- Señales de nivel, son activas durante todo el periodo en que permanecen en el nivel activo. El grafismo que utilizaremos es una flecha de trazo continuo. Pueden ser activas a nivel alto (etiqueta de la señal sin negar) o a nivel bajo (etiqueta de la señal negada).

D0o → */sng* →

3. CH-2020

El CH-2020 es un procesador virtual educativo desarrollado en la UFV que nos permitirá estudiar de forma práctica y concreta el funcionamiento de un ordenador básico

3.1. Arquitectura del CH-2020



La imagen anterior presenta el esquema de la arquitectura del CH-2020.

Se trata de un procesador de 8 bits, lo que implica que es ese el tamaño de palabra básico que utiliza, por ello también lo es el tamaño del bus de datos.

Es una CPU de acumulador, en este caso dicha función la realiza el registro D0 que funciona como acumulador y primer operando de la UAL. Como segundo operando se utiliza el registro D1. Ambos registros son de 8b.

Dispone de un registro de estado (F) también de 8 bits, pero en el que sólo son operativos tres.

El registro IRC (instruction register code, 8b) es la primera de los tres bytes que constituyen el registro de instrucción, dado que una de ellas puede tener 8, 16 o 24bits de longitud.

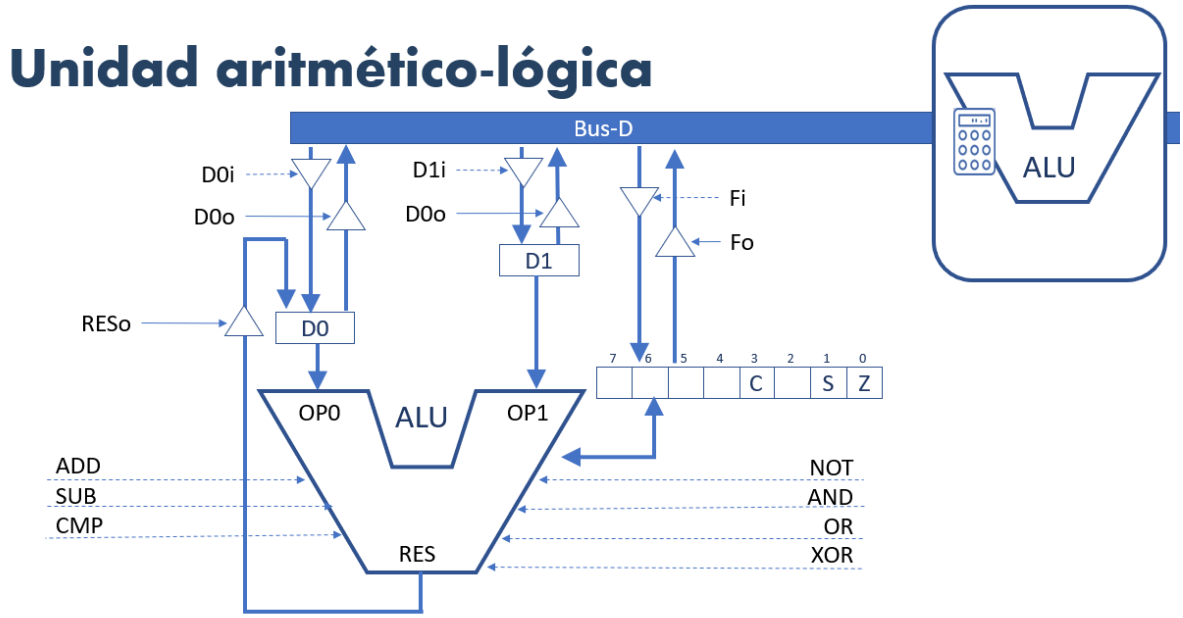
El bus de direcciones tiene 16b, lo que implica que el espacio de direccionamiento es de $0000 a $FFFF (64KB), es una arquitectura von Neumann, por lo que ese espacio de direccionamiento es compartido por el programa y los datos. Como es lógico, también tendrán 16b los registros de direcciones que son cinco: Program Counter (PC), Stack Pointer (SP) (registros específicos de control), A0 y A1 registros de direcciones de propósito general y el IRA (Instruction register address) que almacena cuando es necesario la dirección absoluta contenida en la instrucción.

Los registros A0, A1 e IRA están formados por dos bytes cada uno A0H (A0 High) A0L (A0 Low), A1H, A1L, IRH, IRL, que pueden ser leídos y escritos de forma independiente.

Los registros PC, SP, A0, A1 son contadores lo que les permite ser incrementados y decrementados directamente.

3.2. UAL CH-2020

La unidad UAL es muy sencilla que permite un repertorio reducido de operaciones binarias y unarias de datos de 8b.



Las operaciones que puede son:

- ADD que suma los dos operandos (OP0 y OP1) de su entrada y muestra el resultado en la salida (RES).
- SUB que resta el segundo operando(OP1) de su entrada al primero (OP0) y muestra el resultado en la salida.
- CMP que resta el segundo operando de su entrada al primero pero NO muestra el resultado en la salida
- NOT que obtiene el CA1 del operando 1 y muestra el resultado en la salida.
- AND realiza la operación AND bit a bit de los operandos de entrada y muestra el resultado en la salida.
- OR realiza la operación OR bit a bit de los operandos de entrada y muestra el resultado en la salida.

- XOR realiza la operación XOR bit a bit de los operandos de entrada y muestra el resultado en la salida.

Todas ellas ordenadas por señales de flanco homónimas.

Todas las operaciones que realiza la UAL modifican, en función del resultado los flags del registro F

- Z = 1 cuando el resultado ha sido 0.
- C = 1 cuando se ha producido acarreo.
- S = 1 cuando el resultado ha sido negativo.

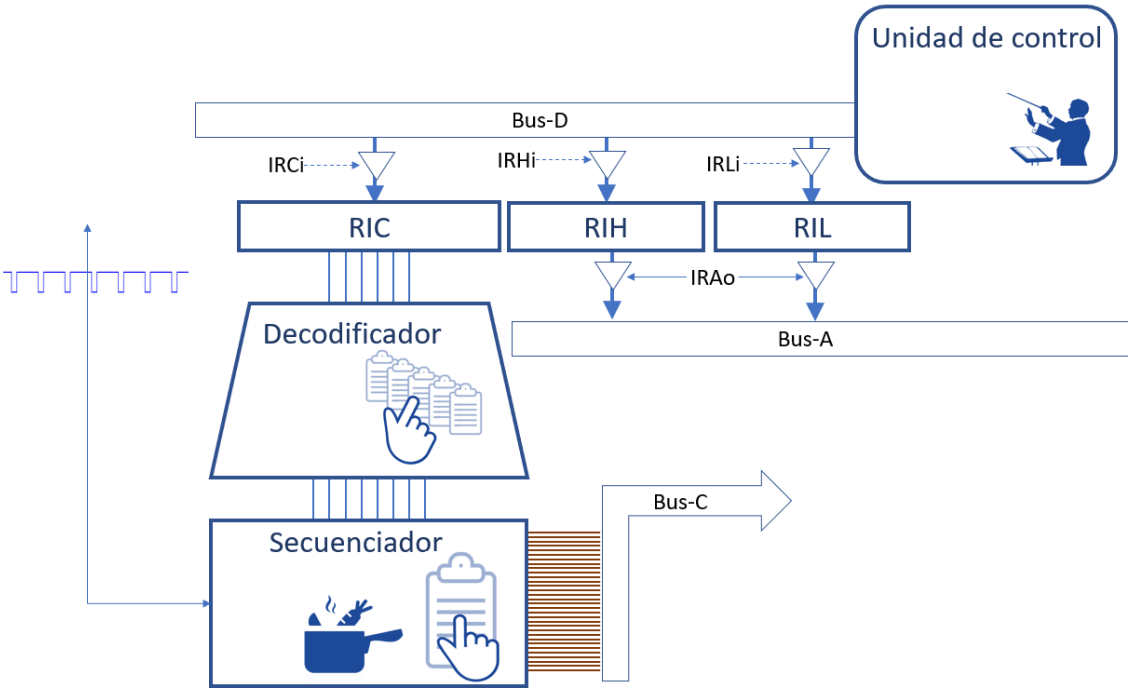
3.3. Registros de datos

El registro D0 alimenta directamente el OP0 mientras que el D1 lo hace sobre el OP1.

El control de escritura en los registros se realiza mediante señales de flanco D0i, D1i, Fi, RESo.

El control de lectura de los registros se realiza mediante señales de nivel D0o, D1o, Fo.

3.4. Unidad de control



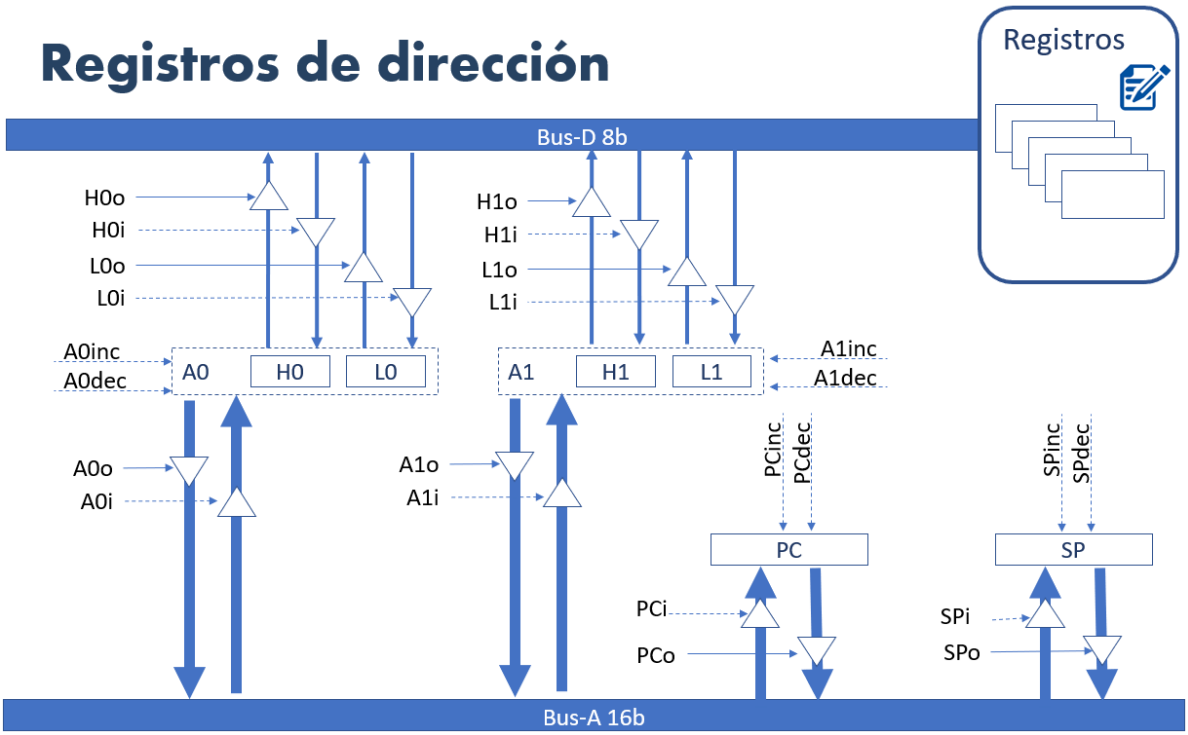
La unidad de control está formada por el decodificador alimentado por el registro RIC, y el secuenciador que descarga sobre el bus de control BUS-C

3.5. Registro de instrucción

El registro de instrucción está formado por tres bytes el primero RIC contiene el código de operación, mientras que los otros dos (RIH, RIL: juntos RIA) contienen, cuando procede la dirección en instrucciones con modos de direccionamiento absoluto.

Al igual que para los registros de datos la operación de escritura esta controlada por señales de flanco (IRCi, IRHi, IRLi) para la carga de los registros desde el bus de datos, mientras que las de lectura por una señale de nivel (IRAo) para la descarga de los registros en el bus de direcciones.

3.6. Registros de instrucción



Como se ha dicho previamente, los bytes (H0, L0, H1, L1) de los registros A0 y A1 pueden ser manejados independientemente, por lo que, al igual que los registros de datos, son controladas por las señales de flanco y nivel equivalentes.

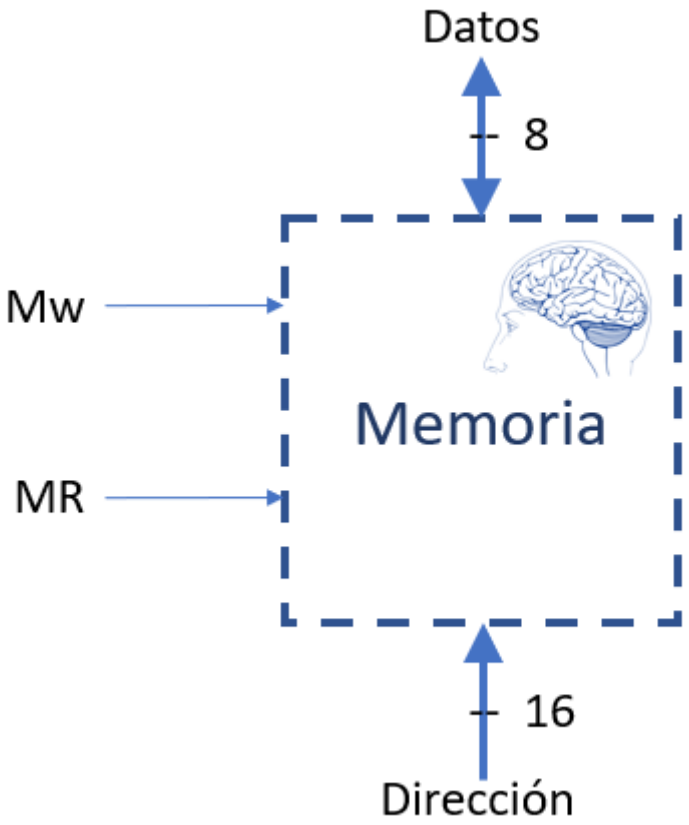
Por su parte tanto A0 y A1 como PC y SP se manejan en bloque de 16b con las señales de entrada y salida correspondientes, pero también disponen de señales para el autoincremento y el auto decremento.

3.7. Memoria

La UC gestiona lectura de la memoria principal mediante la activación de las señale MR (memory read) cuando la dirección está presente en sus entradas, o con la activación de la señal y MW (memory write) cuando el dato y la dirección están presentes en las entradas.

Los ciclos de lectura y escritura de/en la memoria duran tres ciclos de reloj.

Los datos se almacenan en el formato Big endian.



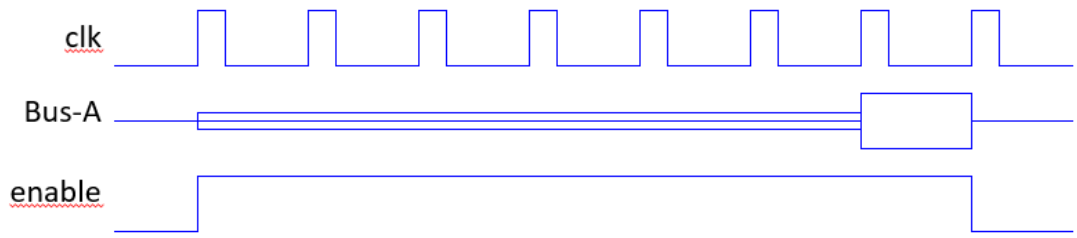
3.8. Tiempo de estabilización

Los datos de los buses A y D no adquieren su estado estable inmediatamente. Necesitan cierto tiempo, por corto que sea, para asegurar que todos los hilos implicados contienen el valor correcto, hecho que afecta a la sincronización de las distintas unidades.

Esta es la causa de que las señales de salida, las de descarga en los buses sean señales de nivel, en espera de que alcancen la estabilidad, mientras que las de entrada, las de carga desde los buses, sean de flanco.

En un cronograma indicaremos este hecho con una barra triple como se muestra en la figura para el bus A.

El tiempo de estabilización de las señales en los buses es 1 ciclo de reloj.



Por supuesto cada acción de entrada ha de realizarse posteriormente al momento en que la estabilidad está asegurada y antes de que se los datos desaparezcan del bus.

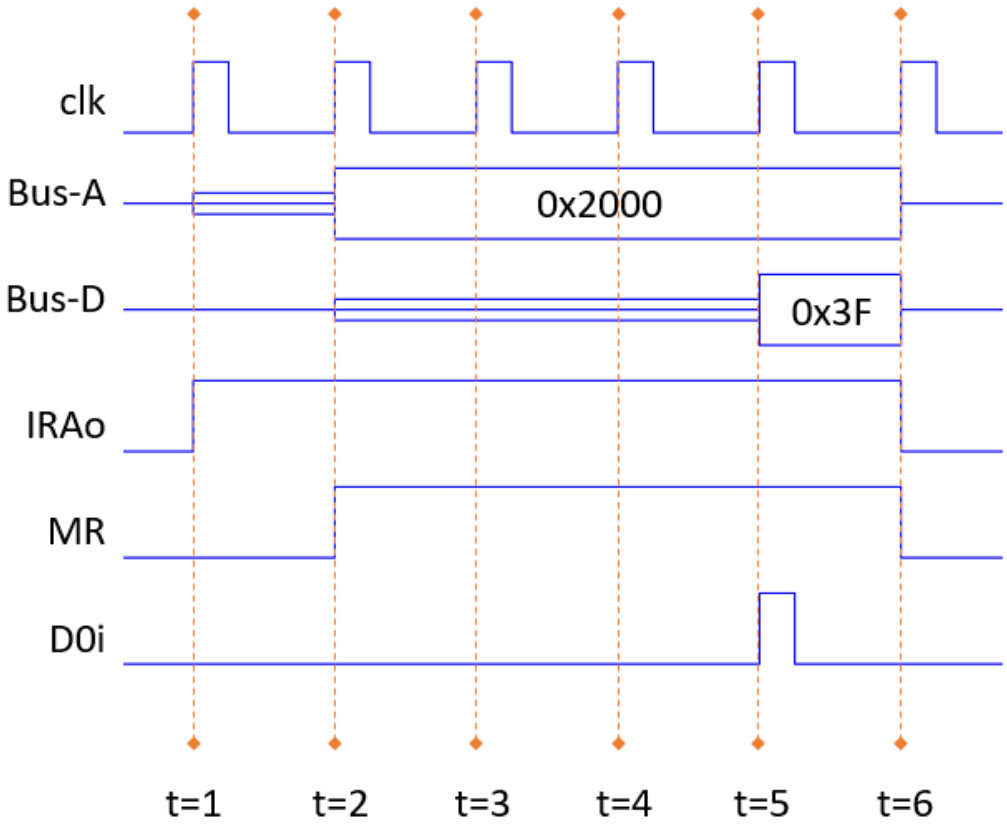
3.9. Cronogramas

Los cronogramas permiten describir cual es la secuencia de las acciones al ejecutar una instrucción o parte de ella y como se sincronizan las señales de control y los datos en los buses.

Veamos algunos ejemplos de cronogramas del CH-2020

3.9.1. Lectura de memoria

Se va ha ejecutar la instrucción **MOV 0x2000,D0** que debe copiar un byte que se encuentra en la posición 0x2000 al registro D0. Suponemos que la lectura de la instrucción ya se ha hecho y que en el RIC se encuentra el código de operación y en RIA la dirección 0x2000



En t=1, con el primer flanco de reloj se activa IRAo con lo que el contenido del registro IRA se descarga en el BUS-A.

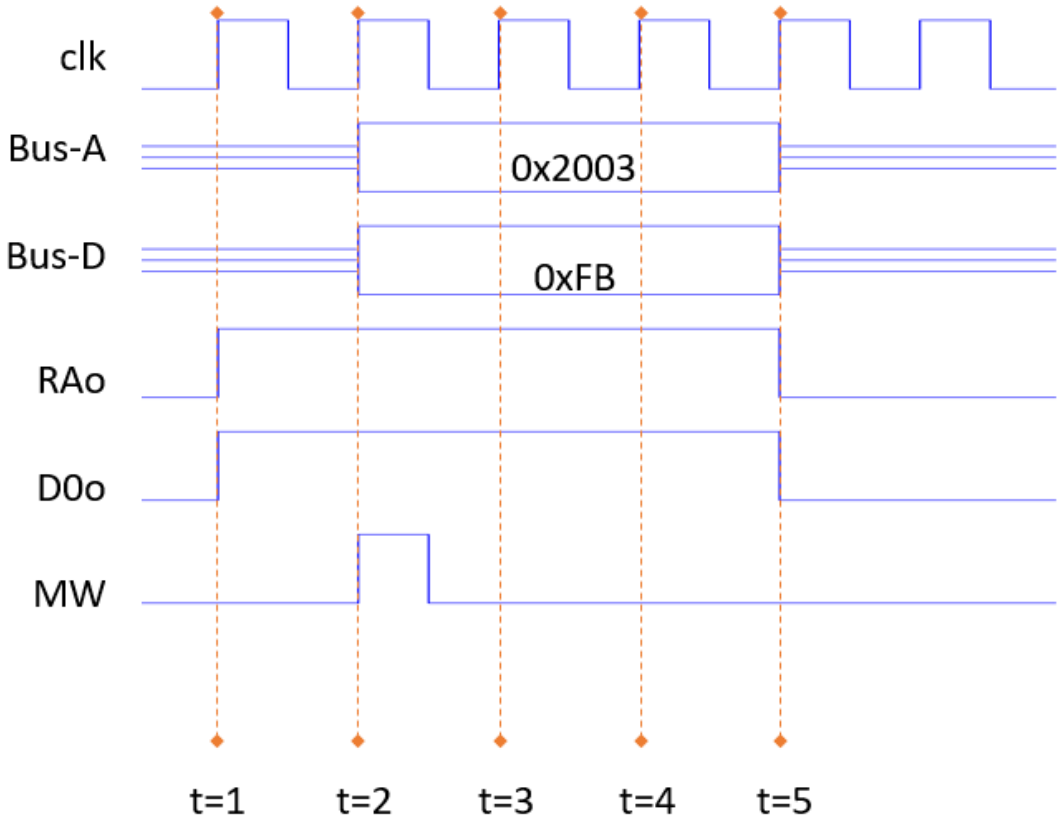
En t=2 la dirección 0x2000 es estable y la UC activa la señal MR que provoca que la memoria inicie su ciclo de lectura., al finalizar el cual el dato está estable en el BUS-D.

En t=5, al finalizar el ciclo de lectura, el dato está estable en el BUS-D yla UC activa DOi y el contenido del BUS-D pasa al registro D0.

En t=6, la UC puede desactivar las señales de nivel que permanecen activas MR e IRAo.

3.9.2. Escritura en memoria

De forma dual a la operación anterior, vamos a ejecutar el final de la instrucción **MOV D0,0x2003** que copia el contenido del registro D0 al byte 0x2003 de la memoria.



En t=1 la UC activa las señales RAo y D0o que provoca que los registros RA y D0 vuelquen su contenido en el BUS-A y BUS-D respectivamente.

en t=2 los datos están estables en el bus y la UC activa la señal MW que arranca el ciclo de escritura en la memoria.

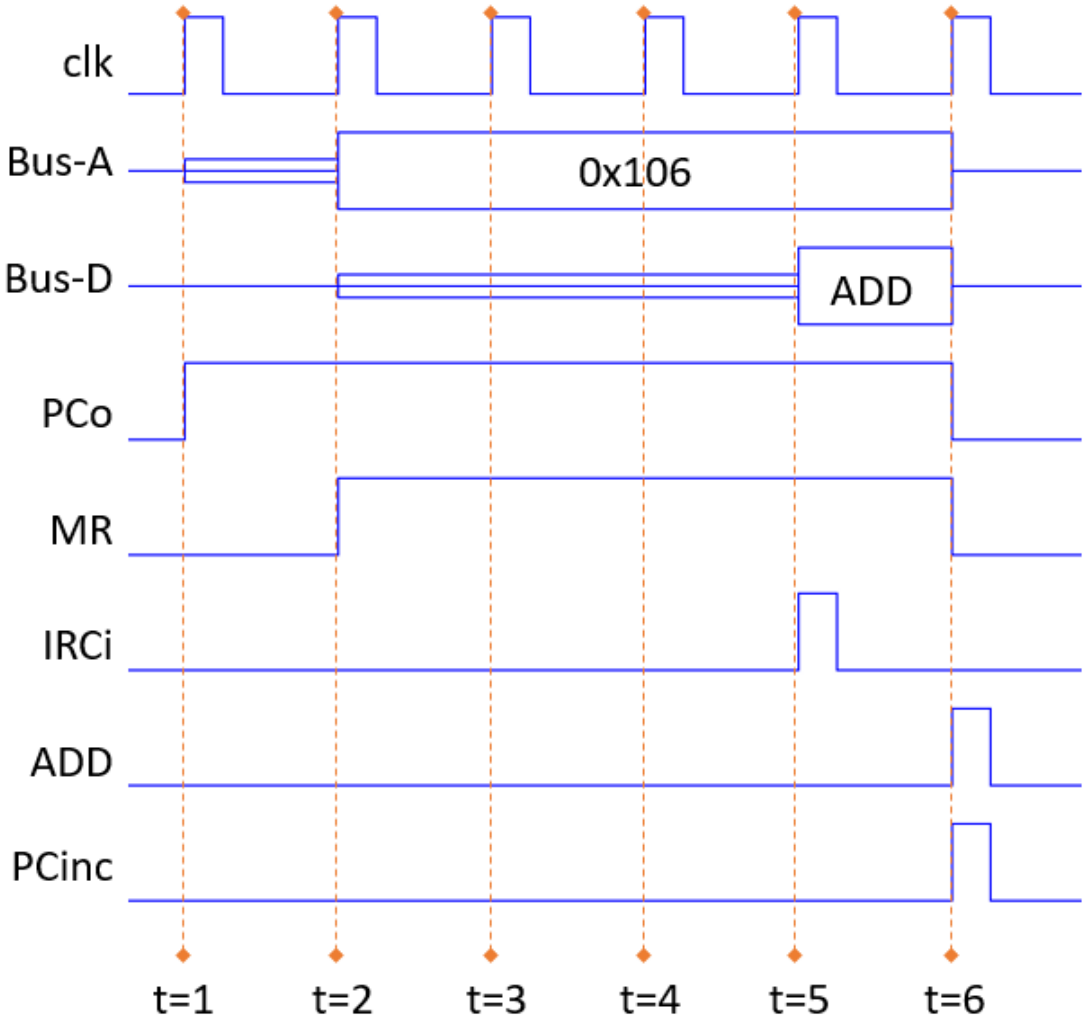
en t=5 el ciclo de lectura ha concluido y la UC desactiva las señales de nivel Rao y DOo.

3.9.3. Una instrucción completa

Vamos a ejecutar una instrucción completa, una muy simple, para poder ver como es la fase de lectura de la instrucción que se ha de realizar siempre antes de la ejecución.

La instrucción a ejecutar es **ADD** esta instrucción se encuentra en la dirección 0x106 de la memoria con lo que, para decirlo todo, podríamos escribir **0x106 ADD**, no habíamos entrado antes en este detalle (que es obligatorio) para no complicar el discurso, pero ahora es necesario.

Suponemos, pues, que el PC contiene la dirección 0x106 que es la siguiente a ejecutar.



Como se sabe, aunque antes supuesto que ya se había hecho, toda ejecución comienza por la lectura de la instrucción por ello:

En t=1 la UC activa PCo lo que provoca la descarga del contenido del PC (0x106) en el bus de direcciones,

En t=2 la dirección está estable en el bus y la UC activa el MR que inicia el ciclo de lectura.

En t=5 el ciclo de lectura ha concluido y el dato está estable en el bus, en este caso el código de operación del ADD, por ello la UC activa IRCi para que dicho código entre en el registro de instrucción donde se decodifica, lo que lleva a que…

En t=6 la UC activa la señal ADD de para que la UAL realice la suma, al mismo tiempo que desactiva las señales de nivel MR y PCo, y también aprovecha para activar PCinc, de modo que el contador se incremente y quede apuntando a la siguiente instrucción (0x107 en este caso).

3.9.4. Una instrucción más compleja

Como en **MOV D0,0x2003** la etapa de lectura de la instrucción es más compleja pues necesita acceder al código de operación (MOV) y a los dos bytes que constituyen la dirección (0x20 y 0x03).

Si la instrucción se encuentra en la dirección 0x100 y ss, **0x100 MOV D0,0x2003** se ejecutaría en las siguientes etapas

- Leer el COP ((PC)) ; ((PC)) =contenido de la dirección de memoria apuntada por PC.
- Incrementar el PC ;0x101
- Interpretar el COP ;MOV DO,ABSOLUTO
- Leer el dato en ((PC))
- Incrementar el PC ;0x102
- Copiar el dato en IRH
- Leer el dato en ((PC))
- Incrementar el PC ;0x103
- Copiar el dato en IRL
- Escribir DO en (IRA) ; Escribir el DO en la dirección apuntada por IRA

Como ejercicio desarrolla el cronograma de ejecución de esta instrucción.

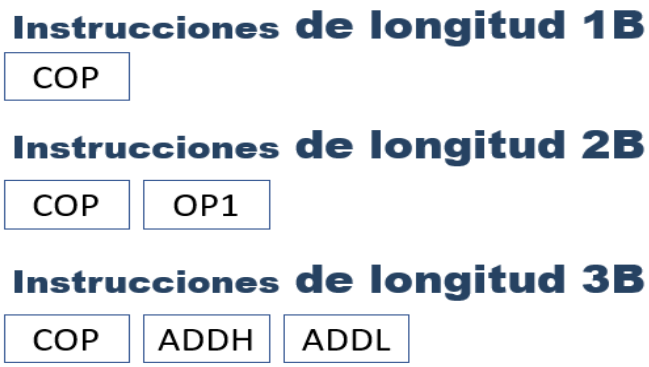
3.10. Repertorio de instrucciones

El CH-2020 tiene un repertorio de instrucciones no muy grande

Las instrucciones de 1 byte no tienen argumentos explícitos, aunque si implícitos. Ejemplos son las operaciones de la UAL como ADD, SUB etc.

En las instrucciones de 2 bytes el segundo corresponde a un dato literal de un byte. Ejemplos son las operaciones de direccionamiento inmediato como algunos MOV.

En las instrucciones de 3 bytes el segundo y tercero forman una dirección (big endian). Ejemplos son las instrucciones de salto absoluto.



En ocasiones en el COP se hace referencia a registros cuyas codificaciones se realizan según las siguientes tablas:

Los registros de datos se codifican con tres bits.

Reg.	Código		
D0	0	0	0
D1	0	0	1
-	0	1	0
F	0	1	1
L0	1	0	0
H0	1	0	1
L1	1	1	0
H1	1	1	1

Los registros de direcciones se codifican con dos bits

Reg.	Código	
A0	0	0
A1	0	1
SP	1	0
PC	1	1

En el documento anexo (CH-2020. repertorio de instrucciones) podrás encontrar una tabla con el repertorio de instrucciones completo y otra con el mapa de códigos de operación.

En el otro documento anexo (CH-2020) podrás encontrar la descripción detallada de cada una de las instrucciones: