

Unidad 2

Arquitectura de Von Neumann:

Es a partir de la misma que se promueve el tratamiento digital de la información dado que hasta 1945 dicho tratamiento era mecánico. Esto lo hace a partir de dos conceptos claves:

El programa almacenado: El mismo computador tiene un programa de instrucciones almacenado en su propia memoria. Las operaciones se ejecutaban al compás de su lectura antes, lo que supone esta arquitectura es que la memoria ahora no solamente es para almacenamiento de datos si no que también aloja las instrucciones del programa.

La ruptura de secuencia: Una toma de decisión involucraba una intervención humana. La ruptura de secuencia permita dotar a la máquina de poder decidir lógicamente en forma automática, a partir de la instrucción de salto condicional.

Un computador de arquitectura Von Neumann se lo puede considerar como un conjunto de unidades conectadas entre si, con una función determinada dentro del esquema. Consta de cinco componentes:

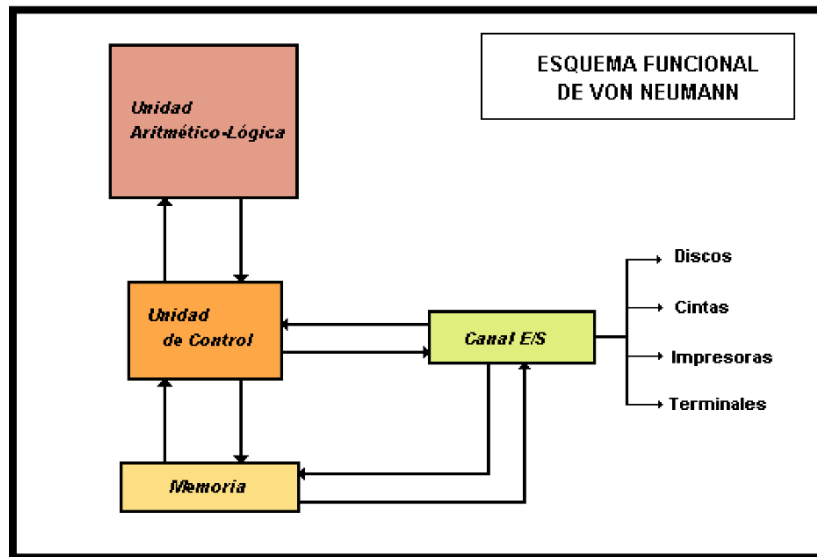
Memoria: Se divide en celdas donde el contenido de cada una es variable. Cada celda posee una identificación un número fijo. La cantidad de celdas disponibles determina la capacidad de una memoria. Aquí se alojan las instrucciones y los datos con los cuales trabaja un programa (llamados operandos).

Unidad aritmético-lógica (UAL): Realiza operaciones de tipo aritméticas (suma y resta) y lógicas (comparaciones).

Unidad de control (UC): Desde acá se controlan y gobiernan todas las operaciones (búsqueda, decodificación y ejecución de instrucciones). Extrae de la memoria central la nueva instrucción a ejecutar, la analiza y extrae los operandos implicados. Habilita el tratamiento de los datos en la UAL y de ser necesario los almacena en la memoria central.

Dispositivos de entrada/salida: gestiona la transferencia de información entre unidades periféricas y memoria central, en ambos sentidos. Advierte a a unidad de control cuando todas las informaciones han sido transferidas.

Bus de datos: Proporciona un medio de transporte de datos entre las distintas partes, no almacena solo transmite.



Abacus

Para estudiar esta máquina, es necesario definir ciertos conceptos que se encuentran en la misma:

Registro: Constituye una suerte de memoria de paso que puede almacenar una cierta cantidad de bits.

Compuerta: Circuitos electrónicos biestables unidireccionales, es decir, permiten el pasaje de información en un único sentido y admiten únicamente dos estados: abierto (1) y cerrado (0).

Bus: es posible discernir en tres tipos:

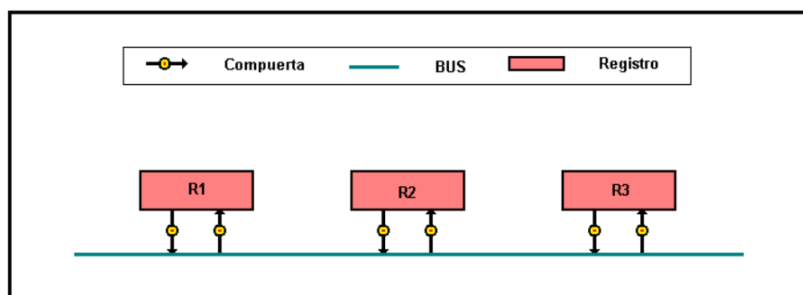
Bus de datos: mueve la información de componentes internos y externos del hardware.

Bus de direcciones: ubica los datos en memoria teniendo relación directa con los procesos del CPU.

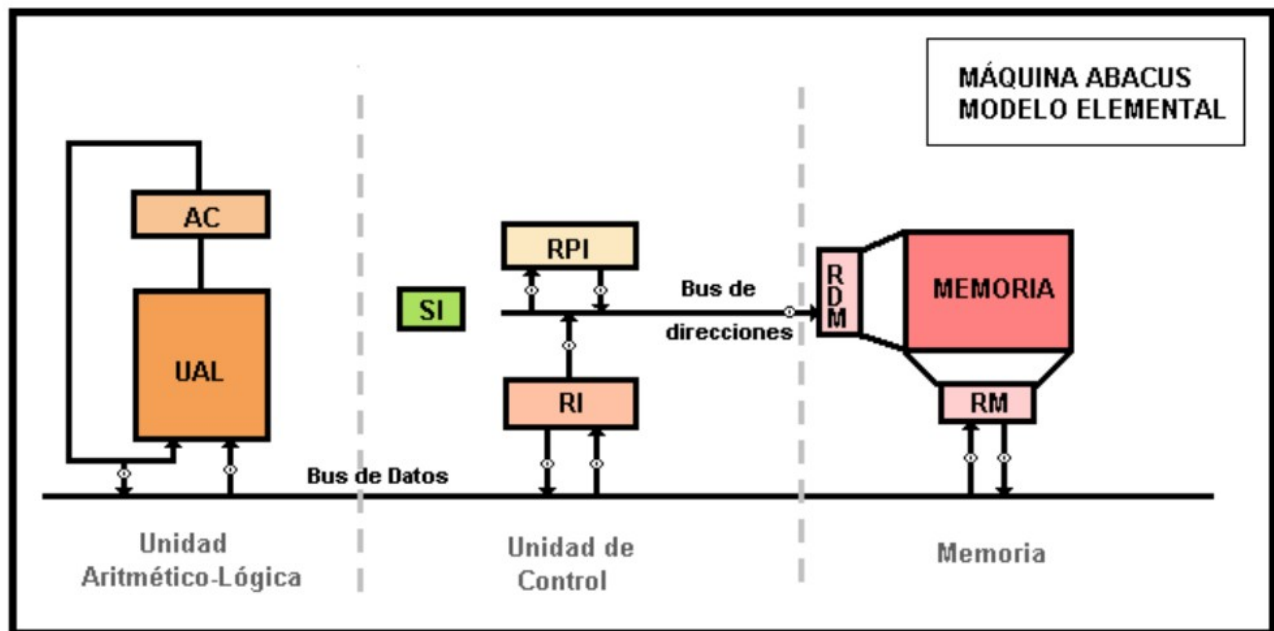
Bus de control: marca estado de una instrucción dada a la PC.

La transferencia de datos puede darse entre componentes de un mismo ordenador o entre varios. Es por esto que existen buses internos y externos.

A partir de compuertas, registros y buses es que se translada la información:



No pueden dos compuertas estar abiertas simultáneamente, tal que la compuerta que manda información al bus es única.



Registros

AC: acumulador

SI: secuenciador de instrucciones

RI: registro de instrucción

RM: registro de memoria

RI: registro de instrucción

RPI: registro de próxima instrucción

RDM: registro de direcciones de memoria

Abacus es una máquina en una sola dirección, la **UAL** (unidad aritmético lógica) posee un registro particular llamado acumulador (**AC**). Este suele albergar primer operando o bien un resultado. Las instrucciones entonces se ejecutan en una sola dirección, la del segundo operando.

Operaciones que realiza la UAL:

- CARGA (cargar un contenido en el AC)
- ALMACENAMIENTO (enviar datos a memoria por el bus)
- SUMA (sumar un dato a lo que haya en el AC)
- LOGICAS (OR, AND, XOR)

Es importante recordar que en una máquina Abacus **todas las operaciones** se realizan “contra” el acumulador y el resultado siempre queda almacenado en él.

En la **UC** (unidad de control) extraemos y analizamos instrucciones de la memoria central, a partir de dos registros:

RPI: Contiene la **dirección** de la próxima instrucción a ejecutar, se comunica con la memoria y el RI a partir del bus de direcciones. A medida que se ejecutan las direcciones, este registro aumenta en una unidad, al menos que se produzca una ruptura de secuencia.

RI: Contiene la **instrucción** extraída de la memoria y se guarda en dos partes: El código de operación y la dirección del operando. Se comunica con la memoria a partir del bus de datos.

El **SI** (secuenciador de instrucciones) se encuentra interconectado con cada componente y es el encargado de administrar la apertura y cierre de compuertas junto al correcto pasaje de la información.

La **Memoria** intercambia información con el resto del computador, ya sea para acceder a los datos o almacenar información dentro a partir de dos registros:

RDM: Contiene la dirección de la celda de memoria en la cual se escribiera o leera la información.

RM: Contiene el dato que debe ser leído o escrito en la memoria.

Registro de instrucción en Abacus

Como vimos, las instrucciones son unidireccionales y constan de dos partes: la operación y el operando. Es decir, lo que queremos hacer con un dato y el dato en sí. Para realizar una suma por ejemplo hay que cargar un operando contra el acumulador, luego sumar el segundo operando para finalmente almacenar en una celda de memoria el resultado.

CARGAR	200	Se suman los contenidos de las celdas de memoria cuyas direcciones son 200 y 206 El resultado se almacena en la celda de memoria de direccion 200
SUMAR	206	
ALMACENAR	200	

Desarrollo de una instrucción

Las instrucciones en Abacus se descomponen en dos fases

Fase de búsqueda: La instrucción inicialmente se encuentra cargada en memoria y la búsqueda de la misma inicia a partir de tener en el RPI la dirección de dicha instrucción. Del RPI se transfiere el contenido al RDM, quien realiza una orden de lectura contra la memoria del contenido alojado en dicha dirección. Una vez que obtengo el contenido, este pasa al RM para luego a partir de una orden la UC se transfiere el contenido al RI. Una vez que los circuitos especializados analizan el código de la operación de la instrucción, se vuelve al RPI y se ejecuta la siguiente, para repetir el ciclo:

RDM ← (RPI)
RM ← ((RDM))
RI ← RM
RPI ← (RPI) + 1

Para buscar un operando, una vez que la UC analiza el código de operación, el RI manda al RDM la orden de operación de lectura. El RDM busca dicha operación y la aloja en el RM.

Ejecución: Cada ejecución implica movimiento de datos de forma secuencial. Cada fase depende del tipo de tarea que se quiera realizar:

- **Suma:** se debe sumar el contenido del RM al contenido del acumulador

RDM ← (Op)
RM ← ((RDM))
AC ← AC + (RM)

- **Carga:** se debe almacenar en el acumulador un dato contenido en memoria

RDM ← (Op)
RM ← ((RDM))
AC ← (RM)

- **Almacenamiento:** se debe “guardar” en memoria el contenido del acumulador

RDM ← (Op)
 RM ← (AC)
 (RDM) ← (RM)

- **Bifurcación:** se debe “saltar” a la dirección indicada en la instrucción. La dirección de bifurcación debe ser transferida al RPI (para buscar la próxima instrucción a ejecutar).

RPI ← (Op)

El código de instrucción se guarda en formato hexadecimal. Las celdas en abacus son de 16 bitsm donde las direcciones ocupan 12 bits y el contenido 4bits.

Sobreescritura de memoria

Así como guardo valores directamente en las celdas, puedo guardar direcciones de otro lugar de memoria, es decir tengo puntero hacía otro lugar. Para poder acceder al valor tengo que pensar en escribir una celda con la instrucción pero en tiempo de ejecución, no en tiempo de codificación.

El contenido en sí, la dirección se encuentra escrita en 16 bits, con los primeros 4 bits en 0 ante la ausencia de una operación. Es decir, solamente tendría la dirección cargada como contenido. Tendría que poder modificar mi primer nipple con el operador que desee. Para ello, debería tener una celda donde guarde previamente, la instrucción pero en ausencia de una dirección (auxiliar de carga o de otra instrucción). De esta manera al sumar ambos números hexadecimales, podría obtener tanto mi operación como mi operando. Una vez que ya tengo la suma en el acumulador, lo guardo en una celda de memoria, dentro de la secuencia de ejecución.

free* Abacus y Abacus Avanzado 6/9/22

Ejercicio 2: Sumar A+B.
 La dirección de A está almacenada en 200
 La dirección de B está almacenada en 201
 Dejar el resultado de A+B en celda 202
 Punto de carga: celda 300

	Contenido	Comentarios	Ayuda
200	050A		
201	050B		
202			
2FE	3000		
2FF	1000		
300	1200	(AC)=050A	
301	32FF	(AC)=150A	
302	2306		
303	1201	(AC)=050B	
304	32FE	(AC)=350B	
305	2307		
306	CCCC	(AC)=<A>	
307	AAAA	(AC)=<A>+	
308	2202		
309	F000		
30A			

La dirección de A está almacenada en 200.
 La dirección de B está almacenada en 201.
 Dejar el resultado de A+B en la celda cuya
 dirección se encuentra almacenada en 202.
 Punto de carga: celda 300.

	Contenido	Comentarios	Ayuda
200	050A		
201	050B		
202	050C		
2FE	3000		
2FF	1000		
300	1200	(AC)=050A	
301	32FF	(AC)=150A	
302	230A		
303	1201	(AC)=050B	
304	32FE	(AC)=350B	
305	230B		
306	1202	(AC)=050C	
307	32FF	(AC)=150C	
308	32FF	(AC)=250C	
309	230C		
30A	1111	(AC)=<A>	
30B	3333	(AC)=<A>+	
30C	2222	(50C)=<A>+	
30D	F000		

Vectores

Un vector es una serie de elementos que se almacenan en memoria. Cuantas celdas de memoria ocupa depende de la lógica de nuestro programa. El vector trivial es que cada elemento ocupe una celda, pero no siempre es así un elemento podría ocupar más de una celda.

Dado un vector de números positivos, sumar sus elementos.

En la celda 200 se encuentra la dirección de inicio del vector.

En la celda 201 dejar el resultado de la sumatoria.

El fin del vector está indicado con el elemento -1.

Punto de carga: celda 300.

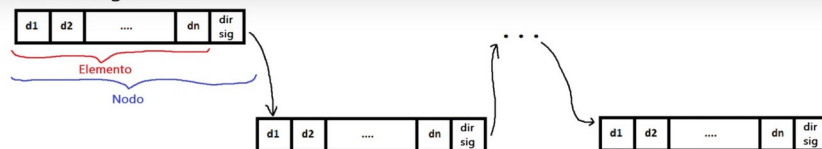


202		
2FD		
2FE	0001	
2FF	1000	
300	0000	(AC)=0000
301	2201	Inicializo sumatoria
302	1200	(AC)=050A/050B
303	32FF	(AC)=150A/150B
304	2305	
305	C000	(AC)=x1/x2
306	050C	
307	3201	(AC)=Sumat + x1
308	2201	
309	1200	(AC)=050A
30A	32FE	(AC)=050B
30B	2200	(200)=050B
30C	0000	
30D	7302	
30E	F000	
30F		

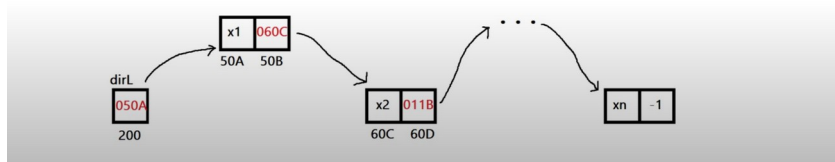
Listas

Las listas no necesariamente están en secuencia todos los elementos. Cada elemento es un nodo, el cual se define como una cantidad de celdas continuas con datos:

Estructura general:



Ejemplo concreto:



La última celda del nodo suele tener la dirección del nodo siguiente.

Ejercicio 4: Dada una lista (L) de nros positivos, sumar sus elementos.

En la celda 200 se encuentra la dir de inicio de la lista

En la celda 201 dejar el resultado de la sumatoria

El fin de L esta indicado con el valor -1 en el último nodo en la celda que apunta al siguiente

Punto de carga: celda 300

200		
201		
202		
...		
2FD		
2FE	0001	
2FF	1000	
300	0000	
301	2201	
302	1200	(AC)=050A/060C
303	8311	
304	32FF	(AC)=150A/160C
305	2306	
306	CCCC	(AC)=X1
307	3201	(AC)=sumat + X1
308	2201	
309	1200	(AC)=050A
30A	32FE	(AC)=050B
30B	32FF	(AC)=150B
30C	230D	
30D	CCCC	(AC)=060C
30E	2200	
30F	0000	
310	7302	
311	F000	
312		

Ejercicio de parcial

Se tiene una lista (L) cuya dirección de inicio se encuentra almacenada en la celda 200(16). Esta lista representa los cursos de un instituto. Cada nodo de la lista (L) está formado por 3 celdas contiguas en memoria: * La primera contiene un número de curso. * La segunda contiene la cantidad de inscriptos al curso. * La última contiene la dirección del siguiente nodo de la lista. El final de la lista (L) se indica con un valor -1 en la última celda del último nodo.

Por otro lado se tiene un vector (V) que comienza en la celda 100(16) que contiene las novedades semanales de bajas del curso. El vector tiene dos celdas contiguas en memoria: * La primera celda contiene un número de curso. * La segunda celda contiene la cantidad de bajas del curso. El final del vector se representa con un -1 en la primera celda.

Se pide realizar un programa ABACUS con punto de carga en la celda 300(16) que recorra el vector (V) y actualice la cantidad de inscriptos al curso, teniendo en cuenta que tanto la lista (L) como el vector (V) están ordenados por número de curso y que no hay ningún curso en el vector (V) que no esté en la lista.

Para que un número sea negativo hay que hacerle not y sumarle 1.

Superabacus

U2 – Máquina Elemental (SuperAbacus)

- Características principales:
 - Registros generales (datos o direcciones)
 - No posee RPI. R0 contiene dirección de próxima instrucción. Incremento vía sumador.
 - Máquina de dos direcciones (dos operandos)
 - UAL permite hacer cálculos con direcciones y datos

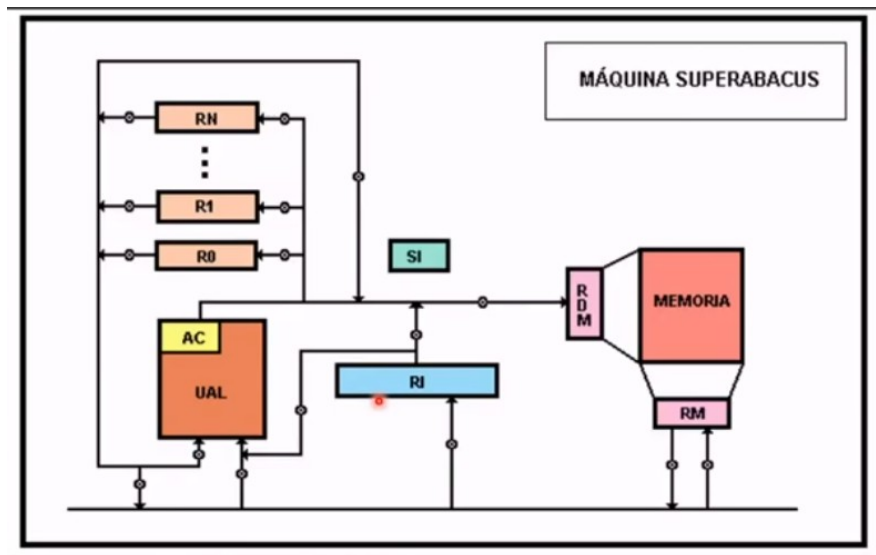
Los registros generales en superabacus tienen la particularidad de que además de ser portadores de información de tránsito, también es posible acceder como programadores a esta información. Es decir, no necesariamente hay que hacerles una carga. Se puede operar sobre los mismos registros.

Serán útiles para alojar punteros hacia algún lugar en memoria.

No existe un RPI en específico, pero aún así los registros tienen esta capacidad. Los registros están enumerados y el 0 se utiliza para esto (la arquitectura ARM es muy cercana a esto). El circuito de suma de la máquina se encarga de cambiar a la próxima instrucción, es decir no se cambia a la siguiente instrucción.

Voy a tener la capacidad de una misma instrucción de máquina cargar dos operandos a la vez. Ya no hay que hacer pivoteos en el acumulador.

La UAL ahora no solamente puede hacer cálculos con datos, si no también con direcciones.

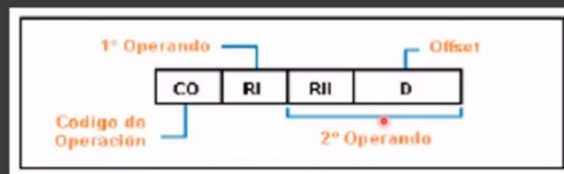


Los registros se encuentran físicamente dentro de la UAL. Por dentro estos cambian como se componen.

U2 – Máquina Elemental (SuperAbacus)

• Formato de instrucciones:

- Código de operación
- Primer operando (R I)
- Segundo operando (R II – D)

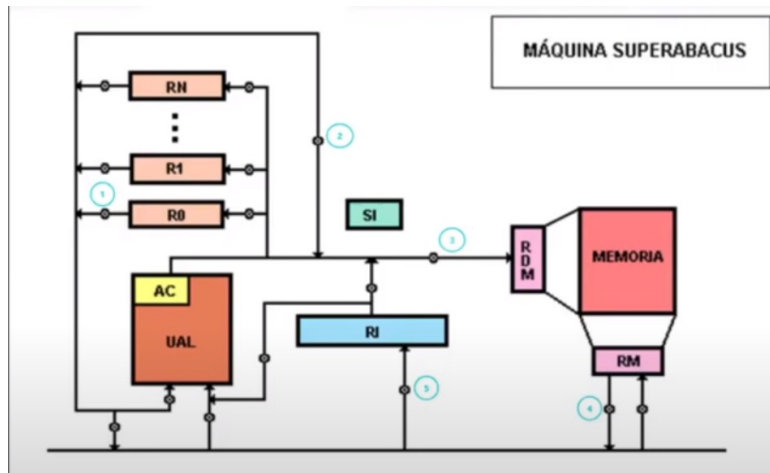


El código de operación se mantiene, no se define cual es el tamaño que alojará (en abacus eran 4 bits). Tengo un campo para el primer operando, el cual aloja el número de registro al que va a acceder. El segundo campo le da más versatilidad gracias al offset.

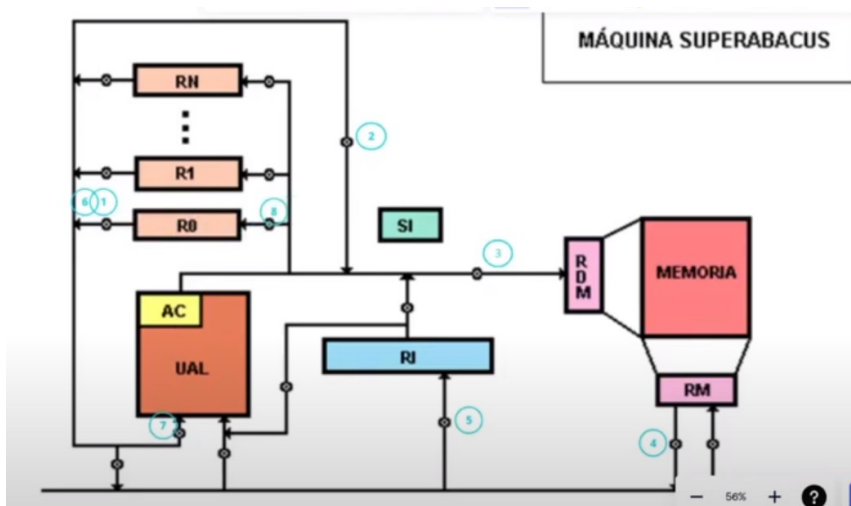
El acumulador cumple la función de alojar momentaneamente el resultado de haber procesado algún dato en la UAL, para luego alojarlo en algún registro.

Fase de búsqueda en Superabacus

Tal que el R0 cumple la función de registro de la próxima instrucción, la fase de búsqueda inicia aquí. Esta celda tiene en que dirección de memoria se encuentra la instrucción de máquina que quiero que ejecute el computador. Por ende del R0, voy al RDM. En el RDM se hace la operación de lectura y la instrucción va al RI. Del RI la instrucción viaja al RM. Del RM la instrucción viaja al AC.



Tras haber llegado a esta fase para ir a la próxima instrucción, al R0 se le suma 1. Para ello, este valor viaja a la UAL, vuelve al acumulador y de ahí vuelve al R0.

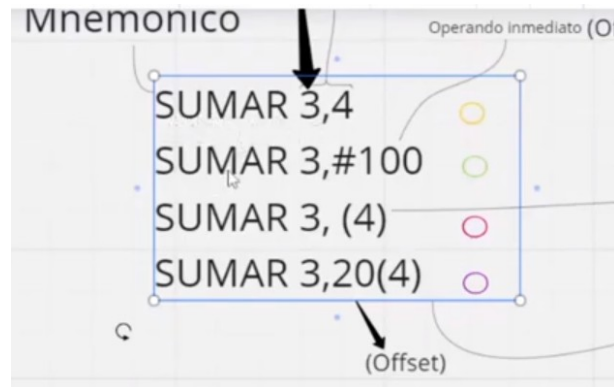


Fase de búsqueda:

(R0) --> RDM
((RDM)) --> RM
(RM) --> RI
(R0) --> AC
(AC) + 1 --> AC
(AC) --> R0

Operación de suma en superabacus

Se le puede especificar dos operandos, pero el segundo operando puede tener distinto origen. Podes ir a buscarlo a la memoria de la máquina, como algo dentro de la propia instrucción como un operando inmediato, etc.

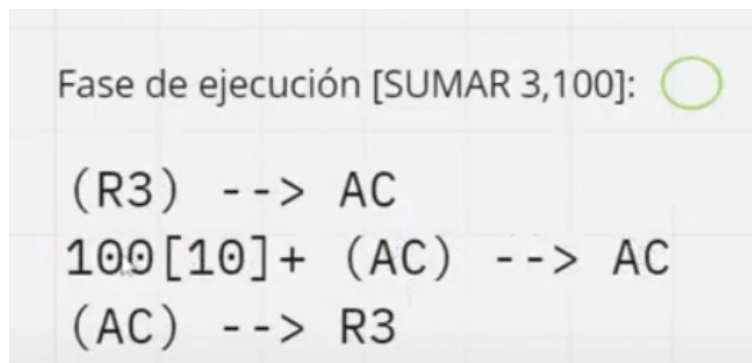


El mnemonico es una abreviación útil para el programador para saber que está haciendo. El catálogo mas usual de instrucciones son de entre 30 y 50 instrucciones de máquina. Como es tedioso acordarse 50 códigos, esto es más fácil.

La primer suma va y busca en el registro general de la máquina, el registro 3 y 4 y los suma. El resultado suele terminar en el primer registro.

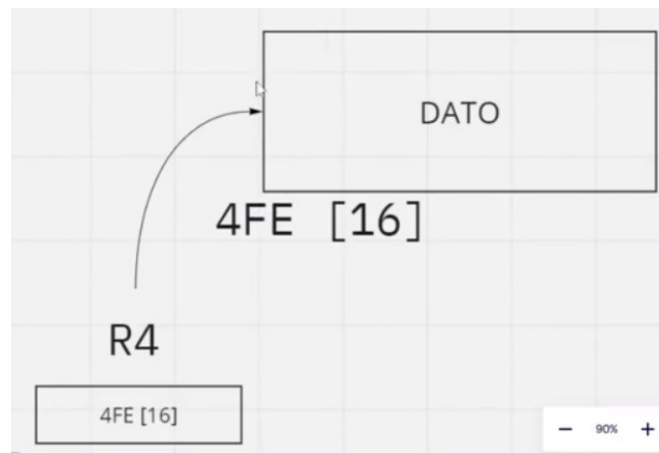
La suma en fase de ejecución para este primer ejemplo, copia el contenido del registro 3 al acumulador. Luego lleva al acumulador el registro 4 y lo suma. Luego se vuelve a llevar al registro 3.

En el segundo ejemplo, voy a poner un **operando inmediato (#)**. El número se escribe en base 10, en principio. Se acumula el registro 3 pasando por la UAL. Por la compuerta que conecta el RI con la UAL, se pasa el operando inmediato.



En el tercer ejemplo, se propone un acceso a la memoria de la máquina, asumiendo que en el registro se

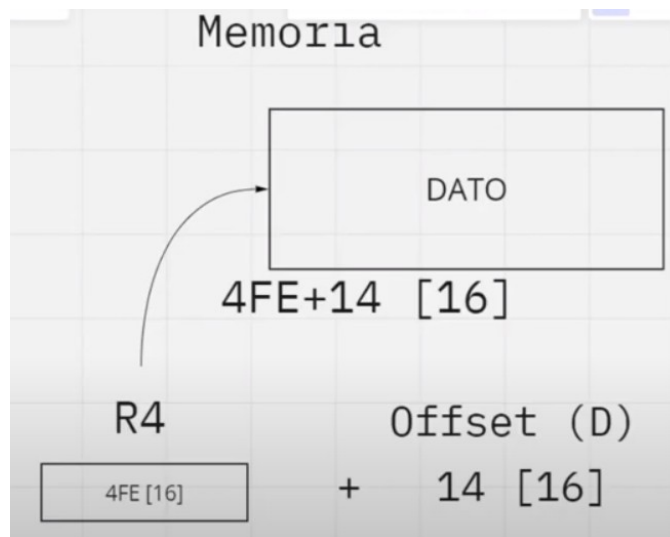
guarda la dirección en memoria de dicho valor, es decir, el registro es un puntero a esta dirección de memoria.



Por ende, para hacer esta suma primero se guarda el valor del registro 3 en el AC, luego el registro 4 manda la dirección al RDM, para que se ejecute la operación de lectura en la memoria y baje por el RDM hacia el AC y se sume. Tras esto, la suma se almacena en el registro 3.


Finalmente, el último ejemplo muestra como es posible combinar datos en el 2do operando, a partir de un número pasado de forma inmediata y un número alojado en memoria cuya dirección se encuentra apuntada en un registro. Lo que se pasa de forma inmediata es cuanto me tengo que trasladar para acceder en la memoria

SUMAR 3,20(4) Dirección del 2do operando = (R4) + 20 [10]



En este caso, no se arranca la ejecución por el registro 3, si no por calcular la dirección del segundo operando. Entonces sumo el contenido del registro 4 y el offset, abriendo compuertas correspondientes.

Tras tener la dirección de memoria en el AC, lo mando al RDM, busco dato, sale por el RM de vuelta a la UAL. Finalmente, cargo el contenido del registro 3 a la UAL, se le suma a lo del acumulador y finalmente vuelve al registro 3.

Fase de ejecución [SUMAR 3,20(4)]: 

(R4) --> AC

20[10] + (AC) --> AC

(AC) --> RDM

((RDM)) --> RM

(RM) --> AC

(R3) + (AC) --> AC

(AC) --> R3