DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Apunte de direccionamiento

Segundo Cuatrimestre 2022

Organización del Computador I
DC - UBA

Direcciones

¿Qué significa lo siguiente?

> Esta arquitectura tiene direcciones de 16 bits, direcciona de a cuatro bytes y opera con palabras de 64 bits.

¿Qué significa **direccionar**, y en consecuencia, qué es una **dirección**?

Memoria principal

Para hablar de direcciones hay que hablar primero de lo que solemos llamar memoria principal, que se puede entender como:

- El dispositivo que almacena temporariamente y que permite acceder a la información con la que opera el procesador (capacidad de lecto-escritura)
- El espacio contiguo y lineal de información sobre el cual opera nuestro módelo de cómputo

Las dos cosas están vinculadas, porque el espacio lineal modela a la información disponible en el dispositivo físico. ¿Qué significa **contiguo y lineal**?
Que interpretamos a la memoria como una secuencia de números ordenados, uno junto al otro.

Espacio de memoria

El espacio de memoria es la unión de la información disponible para el procesador junto con la forma de accederla.
En particular, si consideramos que la memoria está ordenada de forma contigua podemos indicar la ubicación de un dato dentro de la misma por **la posición que ocupa en una secuencia ordenada de tamaño fijo (dirección)**. Por ejemplo, si interpretamos a la memoria como una secuencia de datos de 1 byte, podemos indicar la ubicación del tercer byte sólo por su posición 0x03 (recuerden que 0x es el sufijo para la notación hexadecimal).

Dirección	0x00	0x01	0x02	0x03	...	0xFE	0xFF
Dato	0xC0	0x31	0xA5	0x44	...	0x6E	0xE3

Espacio de memoria

Simplemente le dimos una interpretación a un espacio lineal para poder acceder a una porción del mismo e interpretarlo como un dato de un tamaño dado. Pasamos de la secuencia:

Dato	0xC0	0x31	0xA5	0x44	...	0x6E	0xE3

A la secuencia con direcciones:

Dirección	0x00	0x01	0x02	0x03	...	0xFE	0xFF
Dato	0xC0	0x31	0xA5	0x44	...	0x6E	0xE3

Tamaño de dato

Como en tantas otros mecanismos dentro de nuestra arquitectura la interpretación del tipo de dato viene dado por el uso. Imaginen que queremos interpretar nuestros datos como **enteros de dos bytes**, el espacio de nuestra información es:

Dato	0x31C0	0x44A5	...	0xE36E

Pero en la memoria se encuentra de esta forma:

Dirección	0x00	0x01	0x02	0x03	...	0xFE	0xFF
Dato	0xC0	0x31	0xA5	0x44	...	0x6E	0xE3

Resumen

Hasta ahora tenemos:

- Una memoria ordenada de forma lineal y contigua **M**
- Un espacio de direcciones en base a un tamaño de dato **t**
- Una interpretación de nuestra memoria como una secuencia de datos de tamaño **T**

Imaginen que queremos acceder a dos datos distintos en nuestra memoria para sumarlos. Primero debemos copiar el dato en los registros de nuestra ALU. **Los registros son de dos bytes y la memoria está ordenada en posiciones de datos de 1 byte**. ¿Cómo accedo al segundo dato y al dato que está en la posición número 127?

Accediendo al dato

¿Cómo accedo al segundo dato y al dato que está en la posición número 127?

Hay una aritmética implícita en el acceso a un dato de memoria, tanto en el modelo matemático como en el dispositivo físico. Veamos dos ejemplos para acceder a un **dato de dos bytes** según cómo se definen las direcciones, **direccionamiento a x bytes** indica cuánta información hay entre dos posiciones contiguas de memoria:

direccionando de a 1 byte	0x00 apunta al primer byte
	0x02 al tercero
direccionando de a dos bytes	0x00 apunta al primer byte
	0x02 apunta al sexto

Accediendo al dato

¿Cómo accedo al segundo dato y al dato que está en la posición número 127?

En general, si quiero acceder a un dato de tamaño **T** en la posición **i** de una memoria **M** con direcciones de tamaño t, la aritmética del acceso sería:

$$M_T[i] \equiv M[i * (T/t)]$$

Básicamente hay que escalar el índice del dato que queremos recuperar por la relación entre el tamaño del mismo y el tamaño de direccionamiento T/t.

Accediendo al dato

¿Cómo accedo al segundo dato y al dato que está en la posición número 127?

Entonces, el segundo dato de 2 bytes (**T**) en una memoria que direcciona de 1 byte (**t**) va a estar en $M[1 * (2/1)] = M[2](M[0x02])$ (recordemos que el primer índice de la memoria es el 0x00), y el dato en la posición 127 en $M[126 * (2/1)] = M[254](M[0xFE])$.

Dirección	0x00	0x01	**0x02**	0x03	...	**0xFE**	0xFF
Dato	0xC0	0x31	0xA5	0x44	...	0x6E	0xE3

Copiando al dato

¿Pero qué sucede cuando el dato están en varias posiciones de memoria? Bueno, aquí entra en juego lo que se suele definir como **palabra de una arquitectura dada**, que es el tamaño en bits tanto de los registros como del bus. Entonces, si la arquitectura trabaja con palabras de dos bytes, aunque la memoria esté definida en posiciones que apuntan de a un byte la transferencia va a funcionar bien, porque pueden imaginar que están definiendo la dirección desde la cual se copia el dato completo. En el contexto de Orga 1 el tamaño de palabra va a funcionar, por lo general como el **T** de nuestra aritmética.

Dirección	0x00	0x01	**0x02**	0x03	...	**0xFE**	0xFF
Dato	0xC0	0x31	0xA5	0x44	...	0x6E	0xE3

Memoria direccionable

Hay una pregunta interesante que invierte la forma en la que venimos pensando el problema.

> ¿Utilizando direcciones de tamaño **A** que apuntan a datos de tamaño **t**, a cuánta información puedo apuntar?

Por ejemplo, si **el tamaño de la dirección en sí** (no del dato al que apunta) es de 32 bits y apunta a datos de 2 bytes, puedo tener 2^{32} direcciones, cada una apuntando a una porción distinta dentro de mi memoria. Entonces decimos que puedo direccionar $2^{32} * 2$ bytes.

En general, la fórmula sería $2^A * t$, por eso el tamaño de la dirección en sí y del dato apuntado determina la cantidad máxima de información a la que puedo hacer referencia.

Conclusión

¿Qué significa lo siguiente?

> Esta arquitectura tiene direcciones de 16 bits, direcciona de a cuatro bytes y opera con palabras de 64 bits.

Que cada dirección de mi memoria contigua apunta a una porción distinta de cuatro bytes y que cada operación de transferencia entre registros va a copiar 64 bits (8 bytes), que para conseguir un dato de 64 bits voy a calcular $M[i * (64/32)]$ y que la cantidad de memoria direccionable es $2^{16} * 32 = 256Kb$