Bienvenido: Ingresar
location: WebHome / Hardware / ModuloEMC1

Mapeo en 32 bits

Introducción

Una característica importante en la arquitectura de cualquier microprocesador, es la cantidad de bits que puede operar en forma simultanea, es común entonces referirse a arquitectura de 8,16 32 o 64 bits indicando de esta forma la capacidad de la Unidad Aritmética Lógica (ALU por sus siglas en ingles) en procesar datos.

En los primeros microprocesadores, este tamaño definía además el bus de datos, como se vio en el capitulo anterior los 8 bits de datos o 1 byte, permite conectarse perfectamente con cualquier dispositivo de ese tamaño de bus, permitiendo una relación directa entre la dirección de memoria que pretende direccionar el micro y casillero a acceder en la misma.

Con el advenimiento de arquitecturas de mayor tamaño, aparecieron microprocesadores que no respetaban esta relación entre cantidad de bits de ALU y bus de datos.

En principio condicionados por la tecnología del momento, se debió restringir el bus de datos, así por ejemplo el microprocesador de Intel 8088 utilizado en la primera PC, a pesar de disponer de una ALU que permitía operar con números de 16 bits, su bus de dato era de tan solo 8 bits.

Esto supone un cuello de botella importante, debido a que un numero de 16 bits que era procesado en forma simultanea exigía en la memoria 2 acceso individuales.

La evolución de los microprocesadores y el problema cada vez mayor de poder acceder a la memoria en forma rápida, contribuyó al aumento en el tamaño del bus de dato, en principio equiparándolo a la ALU, para luego duplicarse con respecto a esta en los microprocesadores Pentium II y posteriores.

Acceso a Memoria

La principal ventaja de un bus de dato grande, es que cuando el micro debe leer o escribir a memoria un dato mayor a 1 byte, este lo puede hacer en menos pasos, debido a que se puede operar con 2, 4 u 8 bytes en cada transferencia desde o hacia la memoria. Permitiendo con la misma tecnología y misma velocidad de reloj multiplicar por 2, 4 u 8 la velocidad de transferencia.

Pero esta ventaja viene acompañada de una mayor complejidad, y es el echo que a pesar de contar con un micro de 32 bits de ALU, probablemente se quiera operar algunas veces con dato que son de 16 u 8 bits, se debe de alguna forma instrumentar un método adecuado para poder operar con estos tamaños de datos.

Dirección de Memoria

Algo importante a tener en cuenta, es que a pesar de disponer de un microprocesador de 32 bits de ALU, las direcciones de memoria siguen siendo de bytes por ser esta la menor unidad de datos de almacenamiento direccionable, esto es independiente del tamaño de la ALU.

Como ejemplo se puede hacer.

```
int a,b;
std::cout << "posición de a " << &a << std::endl;
std::cout << "posición de b " << &b << std::endl;
```

dará como resultado algo parecido a

```
posición de a 0xbffff064
posición de b 0xbffff068
```

La diferencia entre ambas posiciones es 4, resultado lógico para dos variables de tipo int (4 bytes), posicionadas en forma contigua.

Simplificando algunos conceptos, y suponiendo un ejemplo mas extenso, estos dos números resultantes son los que colocará el microprocesador en el bus de direcciones cada vez que necesite acceder a una de estas variables en el resto del programa.

Sin embargo, en la memoria donde se encuentran guardadas estas variables, esto no se desarrolla de la misma forma, aquí un bus de dato de 32 bits implica que cada localidad de memoria o casillero mide 32 bits, siguiendo el ejemplo ahora las dos direcciones de memoria que indican donde se encuentran guardados los dos valores, son contiguas y no están separadas 4 bytes como en el microprocesador.

Para ejemplificar este problema, suponemos dos variables contiguas de 4 bytes cada una, posicionadas en la dirección mas baja del mapa.

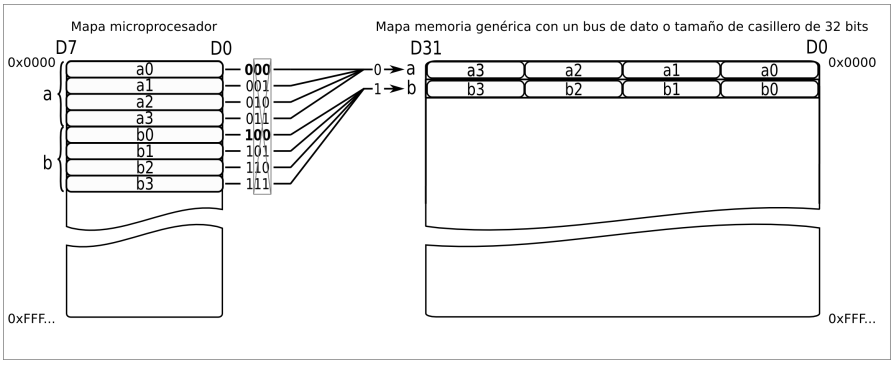
En la figura 1, se pueden observar estas variables denominadas **a** y **b**, posicionadas en el mapa del microprocesador, cada una ocupando sus 4 bytes denominados a0 ... a3 y b0 ... b4 respectivamente.

A la derecha del mapa del microprocesador se encuentran las direcciones en binario de cada uno de los bytes que constituyen las dos variables, resaltando en negrita las direcciones que corresponden además a los punteros de sendas variables.

En el mapa de la memoria, se observan que los 4 bytes correspondiente a cada variables se ubican en un solo casillero, siendo su dirección 0 para la variable a y 1 para la variable **b**.

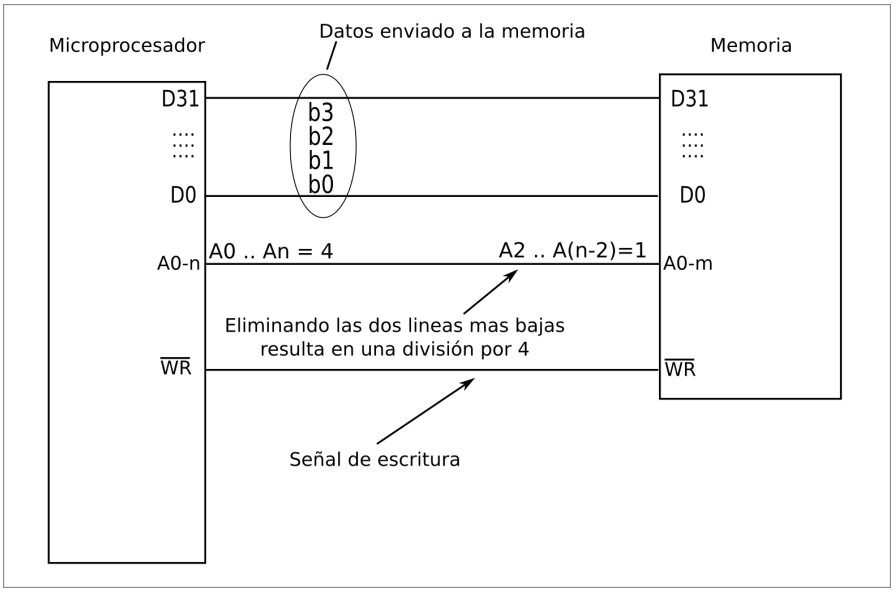
Se debe instrumentar entonces entre los dos dispositivos, una interfase que modifique las direcciones que salen del micro y entran a la memoria.

Como se muestra en la misma figura, eliminar los dos bits menos significativos, (dividir por 4) la dirección enviada por el micro, permite obtener de manera sencilla la dirección del casillero en la memoria en donde se encuentra cada variable o cualquiera de sus bytes.



- Figura 1

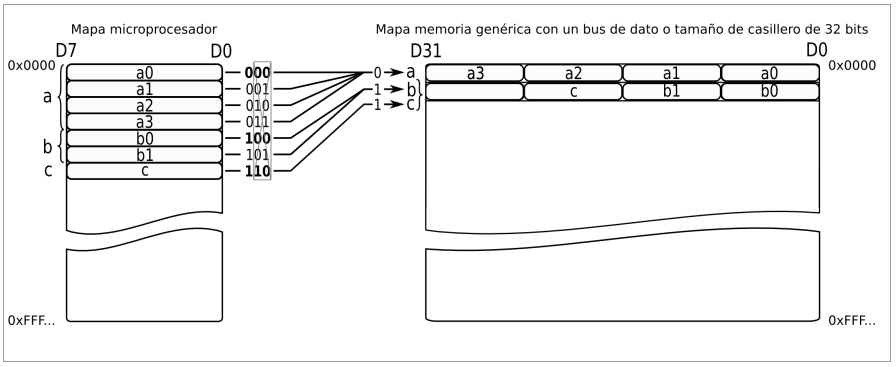
En la figura 2, se detalla el estado de cada bus en el proceso de actualización de la variable **b** ubicada en la posición 4 dentro del mapa del microprocesador



- Figura 2

El problema ahora, es poder operar con variables que resulten de menor tamaño que el ancho del bus de dato.

Modificando el ejemplo anterior ahora la variable **b** es de solo 2 bytes mientras que se suma una nueva variable **c** de un solo byte, como se puede ver en la figura 3, ahora 2 variables comparten el mismo casillero de memoria (casillero 1).



- Figura 3

En estos microprocesadores, se implementan lineas extras que permiten notificar a la memoria cual de los bytes del casillero seleccionado debe modificar, estás lineas (una por cada byte que posea el bus de dato) copia a la señal de WR pero ahora solo en los bytes que deban actualizarse.

Por otro lado, el microprocesador divide a su memoria en grupos de 4 bytes (para el caso de bus de datos de 32 bits), donde cada byte que envía o lee del bus de dato, lo hace relativo al múltiplo de 4 cercano mas bajo, esto se puede ver en la siguiente tabla, donde muestra los valores útiles en el bus de dato que está enviando el microprocesador y los valore de los demás buses, incluido las 4 señales nuevas que denominaremos BLS y que permiten guardar solo los bytes útiles en la memoria.

acción	salida bus de dato	salida bus de direcciones	BLS3	BLS2	BLS1	BLS0	bus de direcciones memoria
escribir **a**	a3a2a1a0	0	0	0	0	0	0
escribir **b**	xxxxb1b0	4	1	1	0	0	1

escribir **c**	xxc0xxxx	6	1	0	1	1	1

Vemos en el ejemplo el caso de la variable **c** que es el mas interesante, si bien en el bus de direcciones envía el 6 (posición de la variable), en el bus de dato desplaza al dato al 3 byte activando también la escritura al BLS2, este desplazamiento es el calculado en base al múltiplo de 4 cercano mas bajo (el cual es 4), permitiendo ubicar el byte en la posición correcta para ser guardada en el mismo casillero en la memoria de la variable **b** pero en el 3er byte del mismo.

UntitledWiki: WebHome/Hardware/ModuloEMC1 (última edición 2013-10-27 16:12:17 efectuada por GuillermoSteiner)