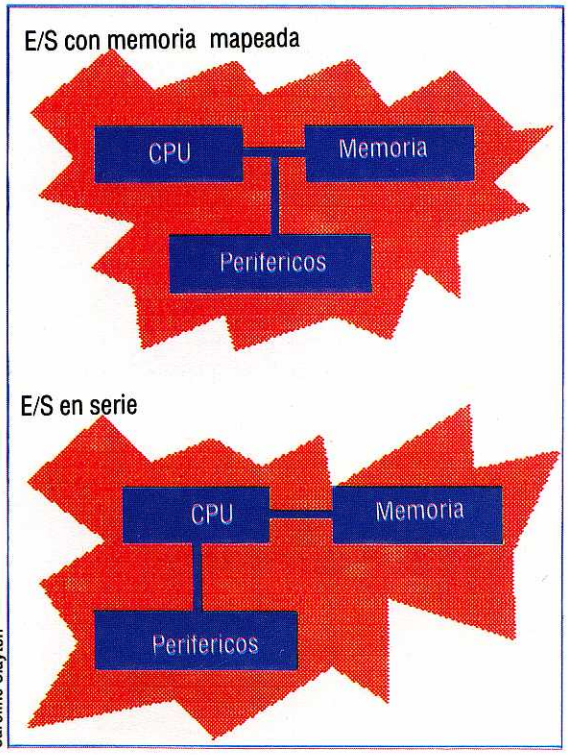
Bit... bit... bit

¿Cuál es la información necesaria para que la CPU saque el mayor partido del mapa de memoria en las comunicaciones de E/S? Veámoslo

Hay dos métodos básicos para seleccionar y comunicarse con dispositivos de E/S:

- *Memoria mapeada*: Este método conecta el bus central de comunicaciones (al que están conectadas la CPU y la memoria) con los dispositivos de E/S. Mediante la adecuada combinación electrónica, los periféricos son sencillamente agregados a este bus. Lo que significa que un periférico se convierta en memoria mapeada, dado que el procesador puede leer y escribir ahora en el dispositivo como si se tratara de una memoria normal. Este procedimiento presenta la desventaja de ralentizar la velocidad en el bus, pues el periférico entra en competencia para obtener sus servicios.
- *E/S aisladas*: En este método se dispone de un bus especial para los dispositivos de E/S, con lo que se incrementa la velocidad de transferencia de los datos respecto al procedimiento anterior. El inconveniente está en que hay que conocer las nuevas instrucciones de E/S que rigen los dispositivos.



Caroline Clayton

Líneas de los buses
Este dibujo muestra la diferencia entre una E/S en serie, que emplea un bus de periférico especial para intercambiar información, y una E/S de memoria mapeada, donde los periféricos toman los datos del mismo bus con el que se accede a la memoria. En este último caso ciertas direcciones habrán de ser reservadas para los periféricos, a ellas puede acceder el ordenador de igual modo que a la RAM o a la ROM

Llegados a este punto no puede sorprendernos saber que el 68000 opta por el método de memoria mapeada. Ya estudiamos en otro lugar la estructura del microordenador. Vimos entonces que algunos dispositivos se conectaban al mismo bus que la memoria y que la CPU, lo que equivalía a decir unas E/S mapeadas.

Veamos ahora los tres tipos principales de información que el procesador exige, y que es mapeado en la memoria.

- *Estado*: Dado que los periféricos tardarán algún tiempo para realizar la operación deseada (imprimir un carácter, leer el teclado, p. ej.), es necesaria una información de su estado para impedir que el periférico reciba órdenes antes de haber acabado su última operación. También es necesaria otra información no directamente referida a la operación pero que puede interesar, supongamos, al modo de operación o a cualquier condición de fallo como quedarse sin papel en la impresora.
- *Control*: Necesitamos un registro que nos permita configurar o activar el periférico que estamos usando (p. ej., ordenar a una unidad de disco que lea un bloque de datos, o configurar un canal de comunicación a una velocidad específica de transmisión).
- *Datos*: Necesitamos, por último, poder leer o escribir datos en el periférico y retener la información allí hasta que se concluya la operación y quede listo para recibir nueva información.

Chips de E/S

En el campo de las señales en el bus principal del ordenador, la operación detallada de los chips de interface puede resultar bastante complicada. Para evitar esta complicación (desde un punto de vista de la programación) se utilizan chips de interface de usos específicos. Motorola proporciona dos chips, uno para el control de dispositivos en serie (ACIA) y otro para el control en paralelo (PIA).

Los dispositivos en serie se conectarán al chip mediante dos hilos solamente para envío y dos para recepción de información; los bits que constituyen los bytes de datos llegan en serie uno detrás de otro. En el caso de dispositivos en paralelo los bits de cada byte van juntos y al mismo tiempo. El periférico en sí se conecta con el chip a través de un número de un byte (o incluso de una palabra) de líneas.

Una de las ventajas del empleo de estos chips de interface está en que el programa cobra así un alto grado de flexibilidad en el establecimiento de detalles relativos a la configuración del hardware. No obstante, esto significa que el número de bits en la palabra de control, especialmente, puede ser bastante grande. Por ejemplo, una asignación de bit de control ACIA es la siguiente:

Bits 0 y 1	frecuencia de división del reloj e inicialización (velocidad bit serial)
Bits 2 al 4	formato de carácter en serie (del tipo paridad o no, número de bits de stop)
Bits 5 y 6	bits de control del transmisor (como la activación de interrupciones)
Bit 7	bit de control del receptor se pone a uno para activar interrupciones

El orden en que se establecen estos bits debe primeramente asegurar que tenga lugar una comunicación aceptable entre el ordenador y el periférico. Si esto falla se pueden obtener datos sin sentido en el dispositivo receptor.

Se precisan dos instrucciones para una E/S en serie concreta. Una de ellas configura el ACIA para el periférico serial. Por ejemplo:

```
MOVE.B   #$3,ACIACON   inicializa el hardware
MOVE.B   #$15,ACIACON  configura el hardware
```

En este caso, ACIACON ha de estar previamente definido.

Tendremos igualmente bits de estado que se leerán para una correcta transferencia de datos. Por ejemplo:

Bit 0	registro receptor datos preparado (un nuevo carácter está disponible)
Bit 1	registro transmisión vacío (se puede enviar un nuevo carácter)
Bits 2 y 3	señales control modem (CTS y DCD)
Bits 4 a 6	indicaciones de error sobre los datos recibidos (p.e., error de paridad)
Bit 7	petición de interrupción

Las interrupciones serán estudiadas en el próximo capítulo, pero los modems caen fuera del ámbito de esta serie.

Dado que sólo leemos un registro de estado y escribimos en un registro de control, podríamos usar sólo una dirección para lograr el mapeo de la memoria. Esto es igualmente aplicable al registro de datos, ya que sólo leemos de un registro receptor de datos y escribimos en un registro transmisor. Como ejemplo supongamos que queremos entrar un carácter,

```
INCH BTST     #0,ACIASTAT ¿hay carácter
                          disponible?
     BEQ      INCH        si no es así, seguir
                          probando
     MOVE.B   ACIADATA,D0 coloca el carácter
                          como parámetro
     RTS                  retorno subrutina
```

En este caso, esperamos hasta que el bit 0 se active en el registro de control antes de leer un registro de datos. ACIASTAT es igual que ACIACON en este ejemplo.

De modo similar podemos escribir una subrutina para salida de caracteres:

```
OUTCH BTST     #1ACIASTAT  espera hasta
                           que el reg. de
                           datos esté
                           vacío
      BEQ      OUTCH       si no es así,
                           bifurcar y
                           volver a
                           comparar
      MOVE.B   D0,ACIADATA salida carácter

      RTS
```

Si es necesario repetir cualquier carácter recibido llamaríamos simplemente a una subrutina después de la otra:

```
ECHO  JSR      INCH        lee un carácter
      JSR      OUTCH       salida carácter
      BRA      ECHO        iteración
```

Es preciso subrayar aquí que la ejecución de una E/S de esta manera es poco eficaz dado que el ordenador trabaja a la velocidad de la transmisión de los caracteres. Aun así no es útil para ilustrar el mecanismo básico de una entrada/salida programada. En el próximo capítulo veremos cómo contribuyen las interrupciones a solventar este problema y cómo se ejecutan las transferencias de datos en E/S en paralelo.

Resumen de instrucciones del 68000

Antes de pasar a examinar las interrupciones en el 68000, vamos a proporcionar un resumen de todas las instrucciones del 68000 que hemos estudiado. La primera lista es el grupo que sirve para copiar datos y gira en torno a la instrucción MOVE:

Instrucción	Operación
MOVE	Mueve la fuente al destino
MOVEM	Mueve los registros de dirección y datos a y de la memoria (útil en entradas y salidas de subrutinas)
MOVEA	Mueve el contenido de la dirección efectiva al registro de direcciones
LEA	Mueve la dirección fuente al registro de direcciones
MOVEQ	Mueve con rapidez pequeñas constantes al registro de datos
PEA	Coloca la dirección efectiva en la pila
SWAP	Intercambia palabras superiores e inferiores en un registro de datos
EXG	Intercambia palabras largas entre determinados registros
LINK/UNLK	Empleado para establecer parámetros para llamadas a subrutinas (especializadas)

Este conjunto de instrucciones proporciona algunas facilidades bastante amplias, pero hay que estar atento en el empleo de instrucciones tan exóticas como LINK/UNLK. Veamos ahora las instrucciones encargadas de la aritmética con números enteros:

Instrucción	Operación
ADD	Suma fuente a destino
ADDA	Suma registro direcciones fuente a destino
ADDI	Suma datos inmediatos a destino de datos alterable
ADDQ	Suma rápida inmediata para constantes pequeñas (del 1 al 8)
ADDX	Suma con arrastre
SUB(A/I/Q/X)	Resta fuente de destino en cualquiera de las variantes de direccionamiento admitidas por ADD
CMP(A/I)	Compara fuente con destino y establece códigos de condición
CMPM	Compara posiciones de memoria y punteros de posincremento

NEG(X)	Hace negativo el registro de datos de operando
EXT	Amplía el signo en el registro de datos de operando
MULU	Multiplica datos sin signo en el registro de datos por dirección efectiva y pone el resultado en el registro de datos
MULS	Como MULU, para datos con signo
DIVU(S)	Divide el registro destino de datos por el operando fuente. Cociente y resto quedan en el reg. destino
TST	Activa los códigos de condición según sea el operando
TAS	Comprueba y activa el bit más significativo del operando
CLR	Limpia la dirección efectiva

Un buen racimo de instrucciones, ¡pero es necesario tener cuidado al seleccionar la instrucción adecuada para los modos de direccionamiento del operando! Sigue ahora una lista de instrucciones relativas a la aritmética en BCD:

Instrucción	Operación
ABCD	Suma operandos BCD colocando el resultado en el destino
SBCD	Resta los operandos
NBCD	Pone negativo el operando BCD

Es de notar que no existe ninguna instrucción BCD para multiplicar. Sin embargo, las instrucciones lógicas que ahora siguen colman estas necesidades:

Instrucción	Operación
AND	Opera con AND el fuente y el destino, siendo uno de ellos al menos un registro de datos
ANDI	AND lógico con datos inmediatos como fuente
OR	Opera con OR de modo semejante a la AND anterior
ORI	OR lógico con datos inmediatos
EOR	OR exclusivo
EORI	OR exclusivo con datos inmediatos
NOT	Inversión lógica del operando

Es el turno de las instrucciones con comprobación y manipulación de bits:

GRUPO MANIPULADOR DE BITS

Instrucción	Operación
BTST	Comprueba el bit especificado en el operando fuente
BSET	Comprueba y activa el operando destino
BCLR	Comprueba y limpia
BCHG	Comprueba y cambia el operando

GRUPO PARA DESPLAZAMIENTO Y ROTACION

Instrucción	Operación
ASI	Desplazamiento aritmético a izq..
ASR	Desplazamiento aritmético a der.
LSL	Desplazamiento lógico a izq.
LSR	Desplazamiento lógico a der.
ROL	Rotación a izq. del operando, activando adecuadamente el arrastre
ROR	Rotación a der. del operando, activando el bit de arrastre

Por último revisaremos las instrucciones de subrutinas y control de programas:

INSTRUCCIONES DE CONTROL DE PROGRAMAS

Instrucción	Operación
BRA	Bifurca siempre (forma eficaz de bifurcación incondicional)
JMP	Salta siempre (útil cuando se desea hacer algún cálculo en la dirección del salto)
Bcc	Bifurcación condicionada según el código de condición 'cc' que se comprueba. Es decir:

Para operandos con signo:

GT	mayor que
LT	menor que
GE	mayor o igual que
LE	menor o igual que
VS	desbordamiento activado
VC	desbordamiento limpio

Para operandos sin signo:

EQ	igual a
NE	no igual a
MI	menos
PI	más
HI	superior a
LS	inferior o igual que
CS	arrastre activado
CC	arrastre limpio

DBcc	Contador de bucle en decremento y bifurcación según condición de 'cc', como antes. ¡Recordar que la instrucción tiene un sentido diferente!

INSTRUCCIONES DE CONTROL DE SUBRUTINAS

Instrucción	Operación
JSR	Salto a subrutina
BSR	Forma eficiente de JSR
RTS	Retorno de subrutina

Este resumen no incluye todas las instrucciones del 68000. Se ha omitido deliberadamente un grupo especial relativo al control del sistema que emplea instrucciones privilegiadas. Estas instrucciones son empleadas en general por los ingenieros de sistemas operativos, por lo que están fuera de la finalidad de este estudio. Las instrucciones se refieren a operaciones sobre el registro de estado y los punteros de la pila, y las instrucciones de "interrupciones software" llamadas *trampas*. Naturalmente, pueden verse en detalle en el manual del usuario del 68000