Universidad del País Vasco
Facultad de Informática

Departamento de Arquitectura y Tecnología de Computadores

Arquitectura de Computadores I

Unidad de control 1 (solución): instrucción CALL

Queremos añadir una instrucción más al conjunto de instrucciones del procesador BIRD:

```
call subrutina
```

Esta instrucción se utiliza para llamar a una subrutina. Por un lado se guarda en la pila la dirección de retorno (es decir, el valor que en ese momento tiene el PC), (para eso, primero hay que actualizar el índice de la pila, el registro `r31`, incrementando su valor en 1, para que señale la posición que queda libre); por otro lado se hace el salto a la subrutina de forma relativa a la dirección inicial del PC:

```
r31 := r31+1
MEM[r31] := PC
PC := PC_call + desplazamiento
```

Su formato es el siguiente:

Escribe la parte del microporograma necesaria para ejecutar la instrucción. Si fuera necesario hacer algún cambio en la estructura de la unidad de proceso, exprésalo claramente. En lo referente al control del resto de las instrucciones, ¿habrá algún cambio?

Solución

Por un lado, tendremos que escribir el microprograma e indicar cuáles son las señales de control que se tienen que activar en cada microinstrucción; por otro lado, habrá que indicar los cambios a realizar en la unidad de proceso.

En cuanto al microprograma, aunque existen varias opciones, nosotros proponemos la siguiente:

call1: `R_e1 := CR[r31];`

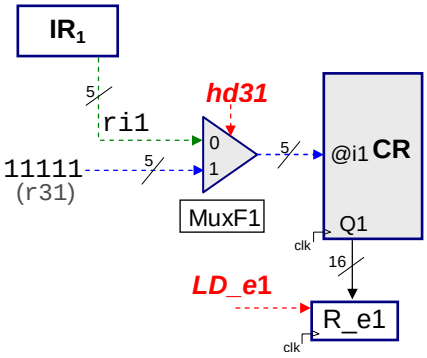
call2: `R_ual := R_e1 + 1;`

call3: `MEM[R_ual] := PC; CR[r31] := R_ual; PC := PCi + IR2; goto 0;`

Es decir, en principio, hay que obtener el valor de la cima de la pila (la dirección guardada en el registro `r31`) para decrementarlo. Por lo tanto, la siguiente microinstrucción después de la fase de descodificación, *call1* (`R_e1 := CR[r31]`), lee el valor de `r31` para copiar el valor al registro `R_e1`. Como en esta instrucción el registro `r31` se utiliza de manera implícita

(puesto que no aparece en el formato de instrucción), hay que indicarle a la entrada `@f1` del conjunto de registros que el registro que hay que leer es el 31.

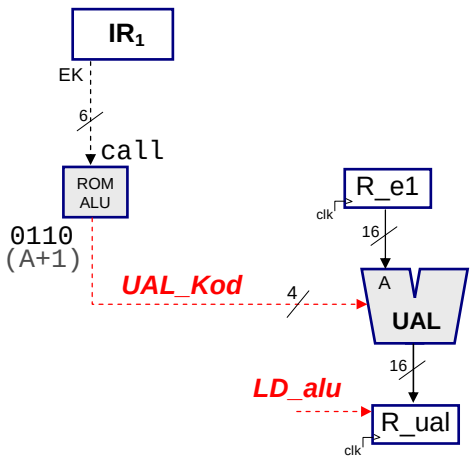
Como en la entrada `@f1` ya están conectados los 5 bits provenientes del registro `IR1`, llamados `rf1`, habrá que cambiar la unidad de proceso de la máquina BIRD para que pueda leer `r31`. ¿Qué cambio será necesario para conseguirlo? Tendremos que añadir un multiplexor (MuxF1) en la entrada `@f1` del conjunto de registros. Las entradas de ese multiplexor son de cinco bits: los bits `rf1` provenientes del registro `IR1`, o cinco unos (para indicar el valor 31 en binario, `11111`). Para seleccionar una de las entradas del multiplexor, será necesaria una señal de control nueva, pongamos que sea la señal con nombre `hd31`. Cuando se esté ejecutando la instrucción `call`, habrá que seleccionar la secuencia de cinco unos, por ejemplo esos bits pueden estar en la entrada 1 del multiplexor, y en el resto de instrucciones se elegirán los bits `rf1` provenientes de la entrada 0. Por lo tanto, para poder ejecutar la microinstrucción ***call1***, son necesarios los siguientes cambios en la unidad de proceso:



Para ejecutar la microinstrucción ***call1*** hay que activar las siguientes señales de control: `hd31` y `LD_e1`.

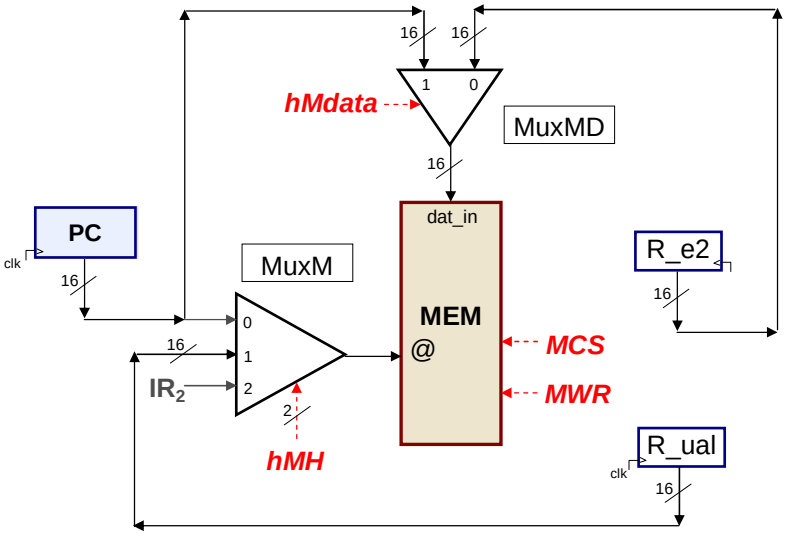
A continuación, hay que ejecutar la microinstrucción ***call2*** (`R_ual := R_e1 + 1`). Así, una vez copiado en el registro `R_e1`, hay que incrementar el valor del registro `r31` utilizando para ello la `UAL`. A la `UAL` le llegará el código para incrementar desde la memoria llamada `ROM_UAL` (partiendo del código de operación de la instrucción `call`) y el resultado se guardará en el registro `R_ual`, activando la señal de control `LD_ual`.

Por lo tanto, para poder ejecutar la microinstrucción ***call2***, en la unidad de proceso no es necesario ningún cambio, puesto que todos los componentes necesarios ya están presentes en la misma, pero dentro de la memoria `ROM_UAL`, en la posición señalada por el código de operación de la instrucción `call`, hay que escribir el código de operación `(0110)` correspondiente a la operación `(A+1)`. De este modo, cuando llegue la instrucción `call`, la `UAL` sabrá que tiene que incrementar el valor del registro `R_e1`.



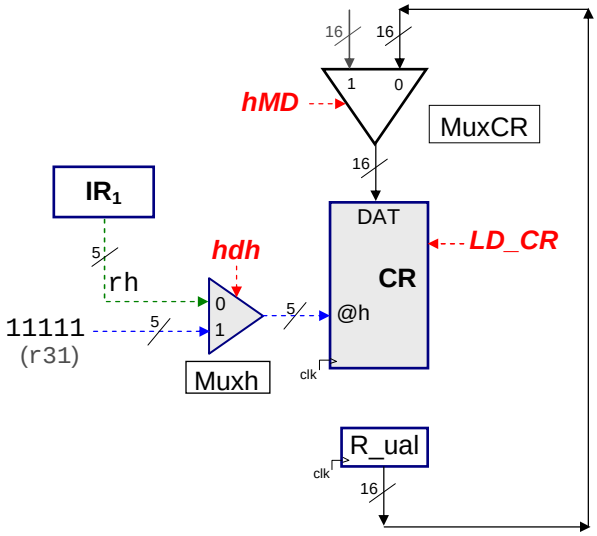
En la microinstrucción **call3** hemos puesto muchas operaciones, pero subrayamos que todas pueden ejecutarse en paralelo (aprovechando que los datos siguen caminos distintos), haciendo ahora también algún cambio en la unidad de proceso. Por un lado, hay que guardar el valor del registro `PC` en la memoria (`MEM[R_ual] := PC`); por otro lado, hay que guardar en el conjunto de registros el nuevo valor de `r31`, el calculado en el paso anterior y que se encuentra en `R_ual` (`CR[r31] := R_ual`); y, por último, hay que asignarle un nuevo valor al registro `PC` (`PC := PCi + IR2`) para llevar a cabo el salto. Haciendo todas esas operaciones en paralelo, la ejecución es más rápida. Ahora, aclararemos todo lo necesario para llevar a cabo cada una de las operaciones.

En cuanto a la primera operación (`MEM[R_ual] := PC`), en la unidad de proceso original no hay forma de guardar directamente en la memoria el valor del registro `PC`, puesto que el único valor que se puede escribir en la memoria es el que proviene del registro `R_e2`. Por lo tanto es necesario abrir un camino desde el registro `PC` hasta la entrada `dat_in` de la memoria. Para ello, pondremos un multiplexor, MuxMD, en la entrada `dat_in`, y las entradas de 16 bits del multiplexor serán el valor proveniente del registro `R_e2` y el valor que guarda el registro `PC`. Aquí también necesitaremos una nueva señal de control para decidir cuál es la entrada que se va a seleccionar. A la nueva señal de control le pondremos el nombre `hMdata`, y por ejemplo, cuando tenga el valor 0 se seleccionará lo que venga del registro `R_e2` y cuando tenga el valor 1 se seleccionará lo que venga del registro `PC`. Por lo tanto en esta microinstrucción hay que darle el valor 1 a la señal `hMdata`, y para poder efectuar la escritura en memoria también hay que activar las señales `MCS` eta `MWR`. En cambio, en lo que se refiere a la dirección de memoria que hay que utilizar para realizar la escritura, no hay ningún problema, porque en la unidad de proceso original ya existe el camino: siendo hMH = '01' se toma como dirección el contenido del registro `R_ual`. Así, una vez realizados los cambios, la unidad de proceso queda de la siguiente manera:



En cuanto a la segunda operación (`EM[r31] := R_ual`), como en esta instrucción `r31` se utiliza de forma implícita, para guardar un valor en el registro `r31` del conjunto de registros, al igual que en la lectura, habrá que indicar en la entrada `@h` el valor 31, además de los 5 bits que vienen de `IR1`. Aquí también, tendremos que añadir un multiplexor con dos entradas (por un lado la secuencia `rh` proveniente de `IR1`, y por otro lado, la secuencia `11111`) y con una nueva señal de control (pongamos una llamada `hdh`). Tal y como hemos hecho en casos anteriores, la entrada 0 del multiplexor será la que acoja el valor que ya existe en la unidad de

proceso, y la entrada 1 acogerá el valor que se ha tenido que añadir (la secuencia `11111` en este caso). Entonces, para poder guardar en `r31` el valor que hay en el registro `R_ual`, como el enlace que va desde el registro `R_ual` a la entrada de datos del conjunto de registros ya está hecho en la unidad de proceso inicial, la señal `hMD` tiene que tomar el valor cero, y únicamente hay que activar las señales `LD_EM` y `hdh`. Después del cambio, también esto se ha añadido a la unidad de proceso:



El último cambio que hay que realizar en la microinstrucción ***call3*** es el que le corresponde al registro `PC` (`PC := PCi + IR2`). Como hay que realizar un salto relativo, todo lo necesario para ello ya estaba anteriormente en la unidad de proceso, por lo que activando la señal `hD` se selecciona `PCi` en `MuxPC`, y en `MuxSum` el desplazamiento que hay en `IR2`. Estos dos valores se suman en el sumador que hay a la entrada del registro `PC`, y al activar la señal `LD_PC` el valor obtenido se guarda en el registro `PC`. Por lo tanto, y en resumen, las señales de control que hay que activar en la microinstrucción ***call3*** son, `hMdata`, `MCS`, `MWR`, `hdh`, `LD_EM`, `hD` y `LD_PC`, y el valor de `hMH` tiene que ser `01`.

En cuanto a la secuenciación, después de la microinstrucción ***call1*** siempre hay que ejecutar la microinstrucción ***call2***, y, después de la microinstrucción ***call2***, siempre la microinstrucción ***call3***. Por lo tanto, en esos dos casos, el código cualificador es el `00` (el correspondiente a la constante cero, puesto que nunca ha de realizar un salto), el bit de descodificación es `0`, y en los cinco bits de la dirección de salto nos da igual qué poner, puesto que nunca va a realizarse ningún salto. En el caso de la microinstrucción ***call3***, en cambio, puesto que después siempre va a ejecutarse la microinstrucción `0`, (`goto 0`), el código cualificador es el `11` (la constante 1, porque siempre hay que realizar un salto), el bit de descodificación será el 0, y los cinco bits de la dirección de salto serán ceros, para poder ir a la primera fase de búsqueda de la siguiente instrucción.

estado	CC1CC0	Desc	S4S3S2S1S0	mcs	moe	mwr	hMH	ld_IR1	ld_IR2	ld_CR	hMD	hH	ld_e1	ld_e2	hB	ld_ual	ld_PC	ld_PCi	hD	hd31	hMdata	hdh
call	00	0	xxxxx	0	0	0	xx	0	0	0	x	x	1	0	x	0	0	0	0	1	x	x
call	00	0	xxxxx	0	0	0	xx	0	0	0	x	x	0	0	x	1	0	0	0	x	x	x
call	11	0	00000	1	0	1	01	0	0	1	0	x	0	0	x	0	1	0	1	x	1	1

Está claro que, como hemos tenido que insertar nuevas señales de control, debido a los cambios que se han tenido que hacer, la longitud de las microinstrucciones ha aumentado. En la ejecución del resto de las instrucciones, las nuevas señales de control estarán desactivadas, excepto en los casos que ya se han comentado durante el ejercicio.