Universidad del País Vasco
Facultad de Informática

Departamento de Arquitectura y Tecnología de Computadores

Arquitectura de Computadores I

Unidad de control 2 (solución): instrucción RET

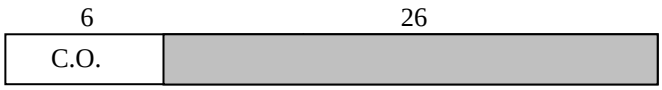
Queremos añadir una instrucción más al conjunto de instrucciones del procesador BIRD:

```
ret
```

Esta instrucción le asigna al registro `PC` el valor de la cima de la pila y decrementa el valor del registro `r31`.

```
PC := MEM[r31],
r31 := r31 – 1
```

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:

ret1: `R_e1 := CR[r31];`

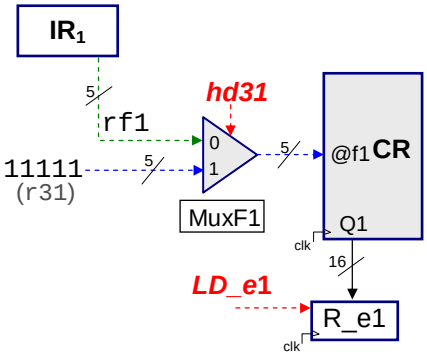
ret2: `R_ual := R_e1 – 1; PC := MEM[R_e1];`

ret3: `CR[r31] := R_ual; 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, ***ret1*** (`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

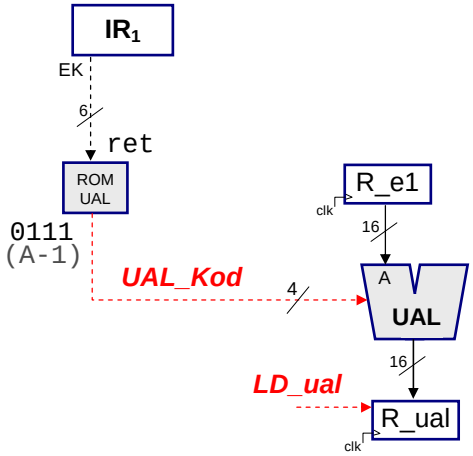
están conectados los 5 bits provenientes del registro `IR1` llamados `rf1`, habrá que cambiar la unidad de proceso de la máquina BIRD para que pueda leer `r31`.

Ahora bien, ¿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 `ret`, 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 ***ret1***, son necesarios los siguientes cambios en la unidad de proceso:



Para ejecutar la microinstrucción ***ret1*** hay que activar las siguientes señales de control: `hd31` y `LD_e1`.

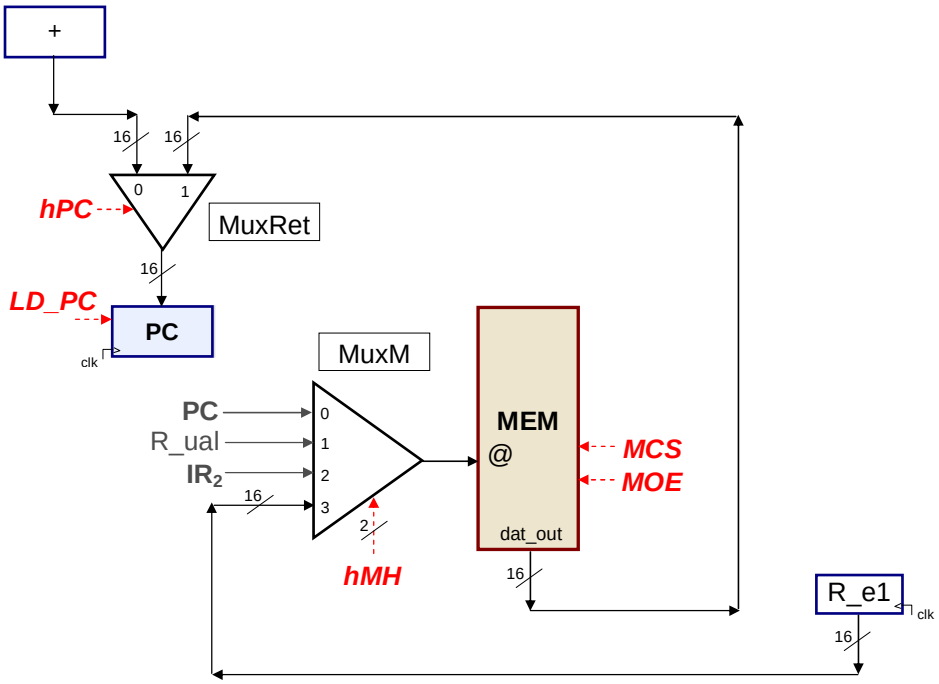
A continuación, hay que ejecutar la microinstrucción ***ret2*** (`R_ual := R_e1 - 1; PC := MEM[R_e1]`). Así, una vez copiado en el registro `R_e1` hay que decrementar el valor del registro `r31` utilizando para ello la `UAL`. A la `UAL` le llegará el código para decrementar desde la memoria llamada `ROM_UAL` (partiendo del código de operación de la instrucción `ret`) 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 ***ret2***, 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 `ret`, hay que escribir el código de operación `(0111)` correspondiente a la operación `(A-1)`. De este modo, cuando llegue la instrucción `ret`, la UAL sabrá que tiene que decrementar el valor del registro `R_e1`.



Por otro lado, en la microinstrucción ***ret2***, también hemos puesto la actualización del registro `PC` (`PC := MEM[R_e1]`), aunque también era posible hacerlo en la microinstrucción ***ret3***. Viendo las conexiones de la unidad de proceso, está claro que esa actualización no es posible sin previos cambios, es decir, en el `PC` no es posible cargar un dato leído de la memoria. Por eso hay que extender el camino de la salida `dat_out` de la memoria al registro `PC`. Para poder hacer eso, pondremos un multiplexor en la entrada de `PC`, conectando con la entrada 0 el valor proveniente del sumador y con la entrada 1 el proveniente de `dat_out`. Para hacer la selección en ese multiplexor, habrá que añadir una nueva señal de control, `hPC`.

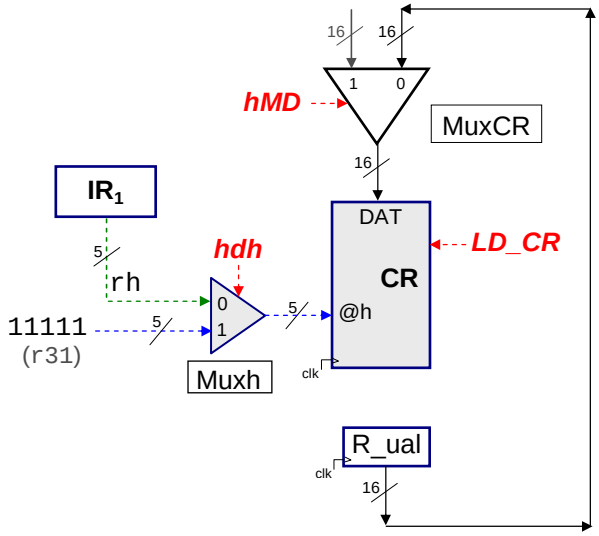
Aún así, para cargar ese valor en `PC`, hay que leerlo en la dirección indicada por el registro `R_e1`. Para ello, hay que conectar la salida de `R_e1` con la entrada `@` de la memoria. En esa entrada se encuentra el multiplexor `MuxM`, con las entradas 0, 1 y 2 utilizadas y la entrada 3 libre. Por lo tanto, podemos insertar en la entrada 3 el valor que viene de `R_e1`, para saber dónde se encuentra el valor a copiar en `PC`. Además, hay que activar las señales de control `MCS` y `MOE` de la memoria para poder leerlo.

Resumiendo, para poder ejecutar la microinstrucción ***ret2***, hay que activar las siguientes señales de control: `LD_ual`, `hMH` (hay que asignarle el valor '`11`'), `MCS`, `MOE`, `hPC` y `LD_PC`. Y hay que realizar los siguientes cambios en la unidad de proceso:



En la microinstrucción ***ret3***, 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:



En cuanto a la secuenciación, después de la microinstrucción ***ret1*** siempre hay que ejecutar la microinstrucción ***ret2***, y, después de la microinstrucción ***ret2***, siempre la microinstrucción ***ret3***. 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 que poner, puesto que nunca va a realizarse ningún salto. En el caso de la microinstrucción ***ret3***, 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.

A continuación se muestra cómo quedan las microinstrucciones en binario.

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	hPC	hdh
ret1	00	0	xxxxx	0	0	0	xx	0	0	0	x	x	1	0	x	0	0	0	0	1	x	x
ret2	00	0	xxxxx	1	1	0	11	0	0	0	x	x	0	0	x	1	1	0	0	x	1	x
ret3	11	0	00000	0	0	1	xx	0	0	1	0	x	0	0	x	0	0	0	x	x	x	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.