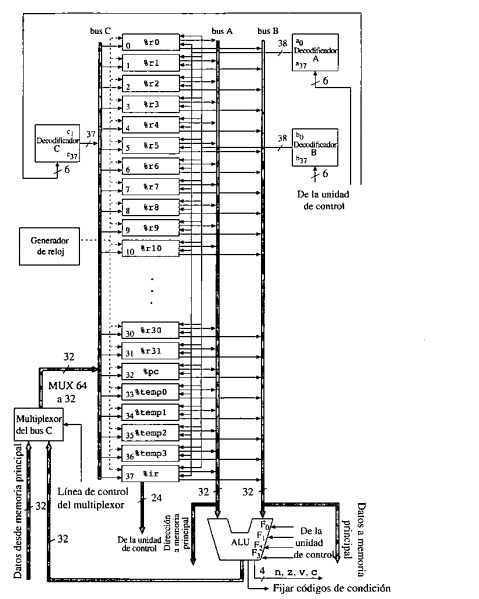
**Capítulo 6**

La microarquitectura está constituida por la unidad de control y los registros accesibles al programador, la ALU y todo otro registro adicional que la unidad de control pueda requerir.

La ejecución del ciclo de fetch (búsqueda-ejecución) es responsabilidad de la microarquitectura: busca la instrucción a ejecutar, determina que instrucción es, localiza los operandos, ejecuta la instrucción, almacena los resultados y repite esta secuencia para la próxima instrucción.

Una unidad de control microprogramada utiliza un microprograma no visible al usuario, que implementa las operaciones sobre los registros y sobre otros sectores de la máquina.

Una unidad de control cableada genera los distintos pasos requeridos para implementar una operación como estados sucesivos de una máquina de estados finitos.

En ambas variaciones, los trayectos de datos no tienen casi ninguna diferencia. Para este análisis, el trayecto de datos será el mismo.

En la figura anterior se puede ver el trayecto de datos. Contiene 32 registros accesibles por el usuario (%r0 - %r31), el contador de programa (%pc), el registro de instrucciones (%ir), la ALU, cuatro registros temporarios no accesibles por el programador (%temp0 - %temp3) y las conexiones entre estos elementos. El contador del programa apunta a la dirección a ser leída desde la memoria principal y se puede modificar únicamente a través de las instrucciones call y jmpl. Finalmente, el registro %ir contiene la instrucción en ejecución.

La ALU puede realizar una de 16 operaciones sobre los buses A y B. El resultado de 32 bits se coloca en el bus C a menos que quede bloqueada por el MUX C como consecuencia de colocar en él una palabra proveniente de la memoria.

La ALU puede implementarse de distintas formas. Para simplificar el análisis se considera, inicialmente, un enfoque basado en una tabla de búsqueda. La ALU tiene dos entradas, A y B, y una salida C, todas de 32 bits, una entrada de control F de cuatro bits, una salida de códigos de condición de cuatro bits (N, V, C, Z) y una señal (SCC (set condition codes)) que actualiza los bits del registro %psr. La ALU puede descomponerse en una cascada de 32 tablas de búsqueda que implementan cada una de las operaciones aritméticas y lógicas, y a continuación un circuito controlador de desplazamientos (barrel shifter) que implementa los desplazamientos.

El barrel shifter provoca del desplazamiento de la palabra de entrada en una cantidad de bits arbitraria (entre 0 y 31 posiciones), determinada según la entrada de control.

Los códigos de condición se implementan en forma directa. Los bits m y c se obtienen directamente de la salida c31 del barrel shifter y de la posición de arrastre que sale de la tabla de búsqueda LUT31 de la ALU respectivamente. El bit z se determina a partir de la operación NOR ejecutada sobre las salidas del circuito de desplazamiento. El bit z vale 1 solo si todas las salidas del mismo son 0 al mismo tiempo. EL bit v adopta el valor 1 si el arrastre ingresado a la posición más significativa de la palabra difiere del arrastre generado desde dicha posición, lo que se implementa con una compuerta XOR. La señal SCC es cierta cuando F3 y F2 son falsas.

Todos los registros están implementados con FF-D activados por flanco descendente, esto es que las salidas solo cambian con un flanco de caída del reloj. Todos los registros son de 32 bits. Para los registros %r1 a %r31, la entrada CLK se introduce en una compuerta AND junto con la línea de selección correspondiente, ci del decodificador C. Esto asegura que el registro solo cambia cuando la sección de control lo determina. Las entradas de datos al registro de toman rectamente de las líneas correspondientes desde el bus C. Las salidas de escriben sobre las líneas correspondientes de los buses A y B a través de compuertas buffer de tres estados, que están desconectadas a menos que sus entradas de habilitación tomen el valor 1. Las salidas de los buffers se transfieren hacia los buses A y B por medio de las salidas ai y bi de los decodificadores A y B respectivamente. Si estas últimas no tienen un 1, las salidas del registro se desconectan eléctricamente de ambos buses A y B gracias a los buffers inhibidos.

El registro %r0 no tienen entradas desde el bus C, ni tampoco desde el decodificador C, y por ende no requiere FF.

El registro %ir tiene salidas adicionales que corresponden a los campos rd, rs1, rs2, op, op2, op3 y bit 13 de las respectivas instrucciones. La unidad de control utiliza estas salidas en la interpretación de las instrucciones. El %pc solo puede contener valores que sean múltiplos de 4, por lo que los sus dos bits menos significativos pueden ser conectados eléctricamente a cero.

Los decodificadores A, B y C simplifican la selección de registros. Sus entradas se seis bits seleccionan un único registro para cada uno de los buses. Existen 64 posibles salidas desde los decodificadores, pero solo hay 38 registros de datos. El índice asignado a cada registro indica el valor que debe colocarse en las entradas del decodificador para seleccionar dicho registro. La salida 0 no se utiliza debido a que %r0 no puede ser escrito. Los índices mayores a 37 no corresponden a ningún registro y pueden utilizarse cuando no se requiere conectar registro alguno a los buses.

El corazón de la unidad de control es una memoria de lectura (Read Only Memory) de 2048 palabras de 41 bits que contiene los valores de todas las líneas que deben controlarse para implementar cada instrucción a nivel del usuario. Esta ROM se conoce como memoria de control y cada palabra de 41 bits es una microinstrucción. La unidad de control es la responsable de la búsqueda de las microinstrucciones y de su ejecución. Esta última se controla a través del registro de instrucciones de microprograma (MIR), del registro de estado %psr y de un mecanismo que permite determinar cuál es la siguiente microinstrucción a ejecutar, formado por la unidad de saltos de control (Control Branch Logic) y el multiplexor de direcciones de la memoria de control. No se requiere contador de programa para almacenar la dirección de la próxima instrucción del microprograma, dado que la misma se recalcula en cada ciclo de reloj.

Cuando la microarquitectura inicia la operación, se coloca la micropalabra de la dirección 0 de la memoria de control en el MIR para su ejecución. A partir de ese punto se seleccionan las micropalabras a ejecutar desde alguna de las entradas Next, Decode o Jump del multiplexor de direcciones de la memoria de control, sobre la base de los valores que adoptan el campo COND del MIR y la salida de la lógica de saltos de control.

Cada palabra de 41 bits comprende 11 campos distintos:

1. El campo A determina cual es el registro del trayecto de datos cuyo contenido debe colocarse sobre el bus A.
2. El campo MUX A selecciona si el decodificador A obtiene su entrada desde el campo A del registro MIR (MUX A = 0) o desde el campo rs1 del registro %ir (MUX A = 1).
3. Idem 1) para el bus B.
4. Idem 2) para el MUX B.
5. El campo C determina en cuál de los registros se almacenará el dato transferido a través del bus C.
6. El campo MUX C elige si la entrada el decodificador C se obtienen desde el campo C del MIR (MUX C = 0) o desde el campo rd de %ir (MUX C = 1).
7. Aclaración: si no se requiere modificar el contenido de ningún registro, se puede usar 000000.
8. RD y WR determinan respectivamente si se debe leer o escribir en memoria. Si RD = 1 se realiza una lectura. Si WR = 1 se realiza una escritura. No pueden valer 1 simultáneamente, pero si pueden valer 0 simultáneamente, en cuyo caso no se lee ni se escribe la memoria. RD controla el multiplexor de 64 a 32 bits del bus C que determina si el bus C se carga desde memoria (RD = 1) o desde la ALU (RD = 0).
9. El campo ALU determina cuál de las operaciones se ejecuta. Puede adoptar 16 valores diferentes. Esto implica que no hay forma de apagar la ALU si no se la quiere usar. En este caso debe elegirse una operación de la ALU que no presente efectos colaterales indeseados.
10. El campo COND hace que el microcontrolador rescate la micropalabra siguiente o bien desde la posición siguiente en la memoria de control, o desde la posición indicada en el campo JUMP ADDR del MIR, o bien desde los bits del código de operación almacenado en %ir. Si vale 000, no hay salto alguno y se utiliza la entrada Next del MUX de direcciones de la memoria de control. E ese caso, se transfiere al circuito que incrementa las direcciones de la memoria de control (Control Store Address Incrementer, CSAI), que incrementa en 1 la salida actual del MUX de direcciones. Si COND vale 001, 010, 011, 100 o 101 se procede a realizar un salto condicional a la posición de la memoria de control indicada en el campo JUMP ADDR, según el valor de los flags o del bit 13 del %ir. Si COND vale 110, se produce un salto incondicional. COND vale 111 cuando se está decodificando una instrucción. En ese caso, la dirección de la memoria de control que debe copiarse en el MIR no se toma ni desde la entrada Next del MUX de direcciones ni desde la entrada Jump, sino desde una combinación de 11 bits creada mediante el agregado de un 1 a la izquierda de los bits 30 y 31 de %ir y del agregado de ceros a la derecha de los bits 19-24 del %ir. El objetivo de esto es que la instrucción sea decodificada en un solo paso a través de saltos a diferentes ubicaciones definidas por los valores de los campos op, op2 y op3 de la instrucción.
11. El campo JUMP ADDR ocupa los 11 bits menos significativos del formato de la microinstrucción. Existen 211 micropalabras en la memoria de control, por lo que se requieren 11 bits de direccionamiento para poder acceder a cualquiera de sus posiciones.

Cuando se habla del microprograma almacenado en la memoria de control suele hablarse de firmware, dado que es el que establece el puente entre el hardware y el software de la máquina. En la memoria de control, cada microsentencia de almacena en forma codificada en una única micropalabra. El lenguaje del microensamblador es un lenguaje simbólico, se utiliza para la codificación del firmware y no es accesible al usuario. Una modificación en el conjunto de instrucciones del procesador involucra cambios en el firmware, en tanto que una modificación a nivel de software rescrito por el usuario no influye sobre el firmware.

Cada sentencia del microprograma esta antecedida por un número decimal que indica la dirección de la micropalabra correspondiente en la memoria de control de 2048 palabras. Se admite más de una operación por línea, siempre que todas las operaciones puedan realizarse en un único ciclo de instrucción.

Antes de iniciada la ejecución por parte del microprograma, el contador de programa se inicializó con la dirección de comienzo del programa cargado en la memoria principal. La primera tarea en la ejecución de un programa a nivel del usuario es la de cargar la instrucción a la que apunta el contador de programa, desde la memoria principal hacia el %ir. En la línea 0 del microprograma se carga el contador de programa en el bus A y se genera una operación de lectura de la memoria. La expresión “AND(R[pc], R[pc])”, simplemente realiza el producto lógico del contador de programa consigo mismo. ¿Para qué sirve? Con el objeto de colocar %pc en el bus A, hace falta elegir una operación de la ALU que utilice el bus pero que no afecte los códigos de condición. Se elige arbitrariamente esa operación, el resultado se descarta.

Una operación de lectura requiere más tiempo de ejecución que una microinstrucción. Como el tiempo de acceso a memoria principal varía, el circuito que incrementa las direcciones de la memoria de control (CSAI) no lleva a cabo el incremento de la dirección hasta que no se haya recibido una señal de reconocimiento (ACK) que indique que la memoria haca completado su operación.

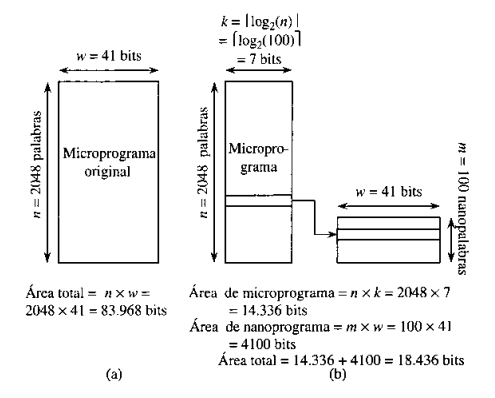
El flujo de control dentro del microprograma se transfiere a la sentencia cuyo número de identificación sea el siguiente en sentido creciente, salvo que se detecte una operación de GOTO o de DECODE. En tal caso, en el ciclo siguiente se carga el MIR con la micropalabra 1 (línea 1). Ahora que la instrucción se encuentra en el registro de instrucción, el paso siguiente es la decodificación de los campos correspondientes al código de operación. Esta operación se realiza ejecutando un salto hacia el microcódigo, de 256 posibilidades diferentes, según lo indica la palabra clave DECODE en la línea 1 del microprograma. La dirección de salto, de 11 bits, se construye como ya se mencionó. Luego de decodificar los campos del código de operación, la ejecución del microcódigo continúa en función de cuál de las 15 instrucciones de ARC está siendo interpretada.

Existen una cantidad de direcciones de DECODE que no deberían aparecer nunca. En previsión de esto, se deberá colocar en esas direcciones una rutina capaz de lidiar con la instrucción ilegal.

Instrucciones como SETHI, Branch y Call no tienen campo op3. Por esto, podría haber muchos códigos DECODE para estas instrucciones, por lo que será necesario duplicar el código para todas las posiciones de memoria. Como solución alternativa, se puede modificar el decodificador de la memoria de control de modo tal que todos los formatos de salto apunten a la misma posición.

Un microprograma escrito en lenguaje microensamblador debe traducirse al código objeto binario antes de ser almacenado en la memoria de control. Cada línea del microprograma de ARC corresponde exactamente a una palabra de la memoria de control, y no existen en el programa referencias hacia adelante sin identificar. Es por eso que el microprograma ARC se puede ensamblar línea por línea en un único paso.

Si la memoria de control es ancha y tiene una gran reiteración de las mismas palabras, puede ahorrarse espacio de memoria de microprograma colocando una copia de cada palabra de microcódigo en un elemento de nanoalmacenamiento, usando la memoria de microporgrama como índice a la memoria de nanocódigo.

Originalmente, la memoria de control consta de 2048 palabras, cada una de 41 bits, dando como resultado una capacidad de 83968 bits. Ahora bien, suponiendo que hay una cantidad de patrones binarios que se repiten, utilizando nanocódigo se lograría ahorrar espacio. Las micropalabras únicas forman un nanoprograma, que se almacena en una memoria de lectura. El microprograma accede ahora en forma indexada al nanocódigo y tiene la misma cantidad de palabras independientemente de si de utiliza el nanocódigo o no, pero cuando se lo utiliza, en la memoria de control se almacenan punteros a aquel en lugar de palabras de 41 bits. Esto representa un ahorro importante en la superficie con relación al planteo microcodificado original. Para un valor pequeño de m y un valor grande de n, siendo m la longitud del nanoprograma, se puede imaginar un gran ahorro de memoria. Por el otro lado, ahora se debe acceder al microcódigo y luego al nanocódigo, por lo que la máquina funcionará más lentamente.

La solución cableada implica el uso de FF y compuertas lógicas en lugar de usar un elemento de almacenamiento y un mecanismo de selección de micropalabras. Los pasos de un microprograma se reemplazan por los estados de una máquina de estados finitos. El diseño de la sección de control se realiza definiendo transiciones entre estados y lineas de control. Y el diseño de la sección de datos de realiza produciendo salidas para cada estado.

Solución cableada vs microprogramada: en la cableada solo se tienen 21 FF contra 83968 FF en la microprogramada, aunque es cierto que el uso de una memoria de lectura ocuparía menos espacio por permitir la utilización de elementos de almacenamiento más pequeños que un FF. La cantidad de lógica combinatoria adicional es comparable. La solución cableada es más veloz con respecto a la ejecución de las instrucciones de ARC, en especial en la decodifaciones de las intrucciones con formato de salto, pero se hace más dificil su modificación una vez inciada su fabricación.