DESCRIPCIÓN DE LA ARQUITECTURA DE LA MÁQUINA RUDIMENTARIA

INTRODUCCIÓN

La Máquina Rudimentaria (MR) es un procesador pedagógico. Su principal objetivo es servir como herramienta para enseñar los conceptos básicos sobre estructura y arquitectura de computadores. El diseño lógico de la MR se ha realizado utilizando conocimientos básicos sobre sistemas digitales, por lo cual puede utilizarse como ejemplo de un sistema complejo en un curso de lógica digital, permitiendo realizar una transición natural entre el estudio de los circuitos digitales y el del lenguaje máquina.

La MR ha sido diseñada en el Departament d' Arquitectura de Computadores (DAC) de la Universitat Politècnica de Catalunya (Barcelona, España).

ARQUITECTURA DE LA MÁQUINA RUDIMENTARIA: NIVEL DE LENGUAJE MÁQUINA

Introducción

La MR es un procesador RISC de propósito general. Tiene un banco de registros de 8 registros de 16 bits, numerados desde R0 a R7. El R0 es un registro especial que no se puede escribir y que siempre contiene el valor 0. Dispone de tres indicadores de condición: V (overflow), Z (cero) y N (negativo) que las instrucciones de salto usan para decidir si el salto se produce o no.

La memoria está compuesta por 256 palabras de 16 bits, y es direccionable a nivel de palabra. Los operandos de la MR son números enteros codificados en complemento a dos con 16 bits.

Las instrucciones de la MR son de longitud fija de 16 bits. El código de operación se encuentra en los dos bits de mayor peso. Existen tres tipos de instrucciones:

- Instrucciones aritmético-lógicas.
- Instrucciones de acceso a memoria.
- Instrucciones de salto.

Las instrucciones usan cuatro modos de direccionamiento:

- Modo registro.
- Modo inmediato.
- Modo relativo (o base + desplazamiento).
- Modo absoluto.

El repertorio de instrucciones de la MR se describe a continuación.

Instrucciones aritmético-lógicas

Las instrucciones aritmético-lógicas usan el modo registro y el modo inmediato.

- En el modo registro se accede a un registro del banco de registros.
- El operando inmediato (en aquellas instrucciones que lo utilizan) es un número de 5 bits codificado en complemento a dos y contenido dentro de la propia instrucción.

Todas las instrucciones aritmético-lógicas escriben su resultado en un registro (Rd) y activan convenientemente los indicadores de condición V, N y Z. Si el registro indicado es el R0 no se escribe el resultado, y solamente se activan los indicadores V, N y Z.

Se distinguen dos tipos de instrucciones aritmético-lógicas, en función de donde están sus operandos fuente: las que leen los operandos de registros y las que tienen un operando en un registro y el otro operando es inmediato. La MR dispone de seis instrucciones aritmético-lógicas:

- ADD-Suma con registros: suma (el contenido de) dos registros (Rf1 + Rf2).
- SUB-Resta con registros: resta dos registros (Rf1 - Rf2).
- ASR-Desplazamiento aritmético: desplaza 1 bit a la derecha con extensión de signo un registro (Rf >> 1). Esta instrucción tiene un solo operando fuente.
- AND- *and* lógica bit a bit de dos registros (Rf1 *and* Rf2).
- ADDI-Suma con inmediato: suma un registro y el operando inmediato (n) extendido a 16 bits (Rf + n).
- SUBI-Resta con inmediato: resta el operando inmediato (n) extendido a 16 bits de un registro (Rf - n).

Todas las instrucciones aritmético-lógicas usan el mismo código de operación (11). A continuación se codifican en binario los registros destino, fuente 1 y fuente 2. En las instrucciones registro registro no se usan los dos bits que vienen a continuación. En las instrucciones con un operando inmediato se codifica este operando usando los tres bits de Rf2 y los dos bits no usados. En el caso particular de la instrucción ASR, de sólo un operando, éste secodifica en el campo Rf2. Los tres bits más bajos de la instrucción constituyen el campo denominado OP, y codifican el tipo de operación a realizar con los datos. El campo OP (bits 2-0) permite identificar a cada instrucción según la siguiente tabla:

- ADD: 100
- SUB: 101
- ASR: 110
- AND: 111
- ADDI: 000
- SUBI: 001

Como puede observarse en la tabla anterior, el bit OP2 distingue los dos tipos de instrucciones aritmético-lógicas: registro-registro (OP2=1) y registro-inmediato (OP2=0).

Instrucciones de acceso a memoria

Las instrucciones de acceso a memoria usan el modo registro y el modo desplazamiento. La dirección de memoria a la que acceden se calcula sumando los 8 bits de menor peso de un registro del banco de registros, denominado registro índice (Ri), con una dirección codificada en la propia instrucción (dirección base).

Existen dos instrucciones de acceso a memoria

- Load: almacena en un registro (Rd) el contenido de una posición de memoria y activa los indicadores V, N y Z. Si el registro destino es R0, no se escribe. Su código de operación es 00.
- Store: almacena en una posición de memoria el contenido de un registro (Rf). Esta instrucción no altera el valor de los indicadores de condición. Su código de operación es 01.

El campo CO (código de operación) ocupa los dos bits de mayor peso. A continuación se codifica el registro destino (LOAD) o el registro fuente (STORE). Después se codifican los tres bits del registro índice y, finalmente, en los 8 bits de menor peso se codifica la dirección base.

Instrucciones de salto

Las instrucciones de salto usan el modo absoluto para indicar la dirección de salto. Se requieren 8 bits para identificar una dirección absoluta. Ninguna instrucción de salto altera el valor de los indicadores de condición. Existen seis instrucciones de salto condicional y una de salto incondicional:

- BG-Saltar si mayor: se produce el salto si N=0 y Z=0.
- BGE-Saltar si mayor o igual: se produce el salto si N=0.
- BL-Saltar si menor: se produce el salto si N<>V.
- BLE-Saltar si menor o igual: se produce el salto si N<>V ó Z=1.
- BEQ-Saltar si igual: se produce el salto si Z=1.
- BNE-Saltar si distinto: se produce el salto si Z=0.
- BR-Saltar incondicionalmente: se produce el salto siempre.

Todas las instrucciones de salto usan el código de operación (10) codificado en los bits de mayor peso. El campo COND (bits 13-11), codificado a continuación, permite identificar a cada instrucción según la siguiente tabla:

- BR: 000
- BEQ: 001
- BL: 010
- BLE: 011
- BNE: 101
- BGE: 110
- BG: 111

A continuación del campo COND se codifican tres bits a 0. Finalmente, los 8 bits más bajos de la instrucción contienen la dirección de salto.

ARQUITECTURA DE LA MÁQUINA RUDIMENTARIA: NIVEL DE LENGUAJE ENSAMBLADOR

El lenguaje ensamblador (LE) se introduce para simplificar la programación en lenguaje máquina. El LE de la MR está constituido por instrucciones, etiquetas, directivas, expresiones y macros.

Instrucciones

El LE codifica, mediante la utilización de un código mnemotécnico, cada una de las distintas instrucciones del lenguaje máquina y la forma de acceder a los operandos.

Instrucciones aritmético-lógicas

- ADD Rf1, Rf2, Rd
- SUB Rf1, Rf2, Rd
- AND Rf1, Rf2, Rd
- ASR Rf, Rd
- ADDI Rf1, #num, Rd
- SUBI Rf1, #num, Rd

Instrucciones de acceso a memoria

- LOAD dir_base(Ri), Rd
- STORE Rf, dir:base(Ri)

Instrucciones de salto

- BEQ dir_destino
- BNE dir_destino
- BG dir_destino
- BGE dir_destino
- BL dir_destino
- BLE dir_destino
- BR dir_destino

Descripción de los mnemotécnicos y de la especificación de los operandos.

- Los registros fuente (Rf, Rf1 y Rf2) contienen los operandos de la instrucción.
- El registro destino (Rd) almacena el resultado de la operación.
- La constante #num es un operando inmediato codificado en complemento a dos con cinco bits.
- La dirección dir_base es la dirección base de memoria para realizar el acceso a un dato.
- El registro índice (Ri) contiene el desplazamiento a sumar a la dirección base para obtener la dirección efectiva del operando.
- La dirección de memoria dir_destino es la dirección donde se debe saltar en caso de cumplirse la condición de un salto.

Etiquetas

Las etiquetas son un mecanismo para establecer referencias a instrucciones y datos sin necesidad de conocer su dirección exacta en memoria. Una etiqueta se define al principio de una línea mediante un identificador seguido por el símbolo ":". Los símbolos definidos como etiquetas pueden ser utilizados para especificar direcciones en instrucciones de acceso a memoria o de salto. El proceso de ensamblaje encargado de la traducción a lenguaje máquina calcula la dirección de memoria exacta correspondiente a cada etiqueta, y la almacena en una *tabla de símbolos* que es utilizada para la traducción de cada instrucción.

Directivas

Las directivas son indicaciones que se introducen en un programa para controlar el proceso de ensamblaje. Suelen empezar con un punto. Existen dos tipos básicos de directivas en el LE de la MR:

- Directivas de definición de variables y constantes: se utilizan para reservar posiciones de memoria o inicializarlas con valores concretos. También permiten la asignación de valores a símbolos.
  - *.dw n1, n2, ..., nN*: inicializa posiciones de memoria consecutivas con los valores indicados (n1, n2, etc).
  - *.rw n*: reserva n palabras consecutivas de memoria sin asignarles valor inicial (la implementación del ensamblador incluido en este entorno de simulación asigna un 0 a estas posiciones).
  - *Identificador = n*: establece una equivalencia entre un identificador y un valor numérico. La declaración de esta directiva sólo afecta a la tabla de símbolos, y no ocupa por tanto memoria del programa.
- Directivas de control de ejecución: Permiten conocer la dirección de la primera y la última instrucción a ejecutar.
  - *.begin identificador*: indica la dirección de la primera instrucción a ejecutar en un programa.
  - *.end*: indica la finalización de un programa.

Cualquier programa en lenguaje ensamblador debe tener una directiva *.begin* y al menos una directiva **.end**.

Expresiones aritméticas

El lenguaje ensamblador permite el uso de expresiones aritméticas en lugar de valores numéricos. Las expresiones aritméticas son evaluadas durante el proceso de ensamblaje. Una expresión es:

- un valor numérico,
- una etiqueta,
- la suma (+), la resta (-), el producto (*) o la división (/) de dos expresiones.

Para realizar la evaluación de las expresiones se utiliza el orden de precedencia clásico de los operadores. No se permiten paréntesis.

Macros

Una *macro* o *pseudoinstrucción* es un módulo de instrucciones parametrizable que puede ser referenciado desde cualquier punto del programa. Para definir una macro se dispone de dos directivas especiales, entre las cuales debe escribirse el código (cuerpo) de la macro:

.def nombre {lista de argumentos}
cuerpo de la macro
.enddef

La directiva *.def* señala el principio de la definición de la macro y le da nombre. La directiva *.enddef* indica el final de la macro. Los argumentos pueden ser de tres tipos, y van precedidos de los símbolos $d, $i o $, según el espacio de direcciones en el que está almacenado el dato al que hacen referencia:

- $dn: el operando n debe ser interpretado como una dirección de memoria,
- $in: el operando n debe ser interpretado como una constante inmediata, y
- $n: el operando n está en el registro Rn de la UP.

ARQUITECTURA DE LA MÁQUINA RUDIMENTARIA: NIVEL DE IMPLEMENTACIÓN

La Máquina Rudimentaria (MR) posee una arquitectura de tipo Von Neumann. Por tanto, la MR ejecuta programas guardados en la memoria que también contiene los datos requeridos por los programas. La forma normal de introducir datos y programas en la memoria en un computador von Neumann es haciendo uso de la unidad de Entrada/Salida (E/S). Dado que la MR no dispone de esta unidad, se asumirá que los datos y las instrucciones se encuentran en una dirección de memoria determinada, sin preocuparse de como han sido almacenados en ella.

Describiremos a continuación las características generales de cada una de las unidades de la MR, según el modelo de arquitectura von Neumann:

- la memoria,
- la Unidad Central de Proceso o procesador y
- los Buses de interconexión.

La Memoria

La memoria de la MR está organizada en 256 posiciones de 16 bits cada una. Esta memoria recibe los datos a través del bus de entrada Min y los envía al procesador a través del bus Mout. Ambos buses son de 16 bits. Para acceder a una posición de memoria es preciso poner su dirección en el bus de direcciones M@. Este bus es de 8 bits. La señal L’/E indica a la memoria si debe escribir en (1) o leer de (0) la dirección presente en el bus de direcciones.

La MR trabaja con datos numéricos enteros de 16 bits, codificados en complemento a 2. Las instrucciones de la MR se codifican en 16 bits, de modo que cada posición de memoria puede contener indistintamente una instrucción o un dato.

La Unidad Central de Proceso

La Unidad Central de Proceso (CPU) de la MR esta constituida por dos elementos:

- Unidad de Proceso: es la encargada de realizar operaciones sobre los datos almacenados en la memoria.
- Unidad de Control: es la encargada de gobernar el correcto funcionamiento de los circuitos lógicos que componen la Unidad de Proceso.

Unidad de Proceso (UP)

La UP es capaz de ejecutar las instrucciones descritas en el nivel de lenguaje máquina. Ha sido diseñada con el doble objetivo de ser eficiente y de fácil comprensión.

La UP tiene los siguientes elementos, gestionados por la UC:

- Un *Registro de Instrucciones* (IR) de 16 bits, encargado de almacenar la instrucción de lenguaje máquina que se esta ejecutando en cada momento.
- Un *Contador de Programa* (PC) de 8 bits, encargado de almacenar la dirección en memoria de la siguiente instrucción a ejecutar (implementando así el secuenciamiento implícito).
- Un *Banco de Registros* de 8 registros de 16 bits numerados de R0 a R7, encargado de almacenar datos. Todos los registros son visibles para el programador de lenguaje máquina. El registro R0 es un registro especial, que contiene siempre la constante 0 (que no puede ser moficada).
- Una *Unidad Aritmético-Lógica* para realizar las operaciones numéricas requeridas por las instrucciones. Además, esta unidad se encarga de evaluar si el resultado de la última operación realizada ha sido cero, negativo o se ha producido overflow. El resultado de esta evaluación se guarda en 3 registros de un bit, denominados *flags de condición* N (negativo), Z (cero) y V (overflow).

En la UP se distinguen dos circuitos sencillos que realizan funciones específicas y tres grandes bloques:

- Evaluación de la condición: es un circuito combinacional que recibe como entrada el valor actual de los indicadores de condición V, N y Z y los tres bits que codifican el campo COND de una instrucción de salto. Envía un bit a la UC que indica si el salto debe o no producirse.
- Extensión de signo (EXT): realiza la extensión de signo a 16 bits del operando inmediato de 5 bits.
- Bloque de gestión del banco de registros: El banco de registros contiene 8 registros de 16 bits (R0 siempre vale cero). Tiene un puerto de salida y un puerto de entrada, y permite lectura y escritura simultánea. La señal de escritura (ERd) está gobernada por la UC. La codificación del registro a escribir se lee directamente de los bits 13-11 de la instrucción (registro destino de las instrucciones aritmético-lógicas y *Load*). El registro a leer se selecciona entre los campos que codifican el registro fuente en las instrucciones aritmético-lógicas (bits 10-8 y 7-5), el registro índice en las instrucciones de acceso a memoria (bits 10-8) o el registro fuente en las instrucciones *Store* (bits 13-11). La selección se realiza mediante el multiplexor SELREG, cuyos bits de control (CRf) son gestionados por la UC.

- Bloque de cálculo aritmético-lógico: Este bloque incluye la Unidad Aritmético-Lógica (UAL), en la que se realizan las operaciones requeridas por las instrucciones y se calcula el valor de los indicadores de condición V, N y Z. La UAL también puede dejar pasar el operando B para activar los indicadores de condición durante la ejecución de las instrucciones *Load*. La señal OPERAR de la UAL es gestionada por la UC, e indica si debe realizarse una operación (OPERAR=1) o bien dejar pasar el operando B (OPERAR=0). La UAL ha sido diseñada de tal modo que los bits 1-0 que codifican el campo OP en las instrucciones aritmético-lógicas coincidan con la codificación de la operación a realizar, por lo que estos bits del IR pueden conectarse directamente a los dos bits de control de menor peso de la UAL.
  El operando A de la UAL siempre proviene del banco de registros. El multiplexor SELDAT permite seleccionar el operando B entre el banco de registros y el operando inmediato para las instrucciones aritméticas, o de la memoria para las instrucciones *Load*.
- Bloque de cálculo de direcciones: En este bloque se realiza el cálculo de todas las direcciones de memoria a las que se accede durante la ejecución de un programa. Las direcciones se almacenan en el PC o en el R@, y la UC selecciona el registro adecuado mediante el multiplexor SELDIR. Los accesos a memoria pueden realizarse por tres causas distintas:
  - Secuenciamiento implícito: La dirección de la siguiente instrucción a ejecutar se lee del PC. Incluye el autoincremento del PC.
  - Ejecución de una instrucción de acceso a memoria: La dirección de acceso está en el registro R@, y se calcula sumando la dirección base (bits 7-0 de la instrucción) con los 8 bits de menor peso del registro índice. Cuando se realiza el cálculo de la dirección, la UC selecciona la entrada 1 del multiplexor SELREG para leer el registro índice.
  - Ejecución de una instrucción de salto: La dirección de acceso está en el registro R@. Para almacenar en R@ los bits 7-0 de la instrucción (dirección de salto) se suman éstos con 0. Para ello, la UC selecciona en el multiplexor SELREG la entrada 1. Esta entrada selecciona los bits 10-8 de la instrucción, que en las instrucciones de salto valen “000” , y por tanto se selecciona el registro R0 del banco de registros, poniendo un 0 en la entrada del sumador.

<u>Unidad de control (UC)</u>

El funcionamiento de la UP esta gobernado por la UC. La UC es un sistema secuencial que determina el orden en que deben actuar los distintos elementos en la UP para completar la ejecución de cada una de las instrucciones. La UC gestiona la carga de los distintos registros (Ld_RA, Ld_IR, Ld_PC, Ld_R@, Ld_RV, Ld_RZ y Ld_RN) y del banco de registros (ERd), la memoria (L’/E) la UAL (OPERAR) y los distintos multiplexores (PC’/@ y CRf). El control de estos bloques se realiza según el código de operación de la instrucción en ejecución (CO) y la evaluación de la condición de salto (Cond).

La UC se comunica con la Unidad de Proceso mediante un conjunto de señales de control. Su función principal es controlar el secuenciamiento de las operaciones realizadas en la Unidad de Proceso para que las instrucciones del programa se ejecuten correctamente. Este circuito recibe como entradas 3 bits de la Unidad de Proceso y envía 12 bits de salida para controlar los módulos que la forman (registros, ALU, multiplexores, etc.).

La UC se especifica como una máquina de estados finita usando el modelo de *Moore*. En este modelo, las salidas están asociadas a los estados y las transiciones entre estados dependen de las entradas del sistema. Las operaciones a realizar en cada estado se especifican en una tabla de salidas.

Los Buses

Los buses de la MR establecen la comunicación entre la CPU y memoria. En la MR, los buses tienen las siguientes características.

- El bus de datos está dividido en dos:
  - Un bus de 16 bits para transportar los datos de la CPU a la memoria (Min).
  - Un bus de 16 bits para transportar los datos de la memoria (Mout) a la CPU.
- El bus de direcciones es de 8 bits y va sólo de la CPU a la memoria (la MR no dispone de unidad de E/S).
- El bus de control tiene únicamente la señal que indica a la memoria si la operación que se realiza es de lectura o de escritura. En los ciclos de escritura no está definido el contenido de la salida Mout.

¿CÓMO ES UN PROGRAMA ESCRITO EN LENGUAJE ENSAMBLADOR DE LA MR?

- Cada línea del programa debe contener una sola instrucción
- Todos los programas deben tener una línea con la directiva '.BEGIN <etiq>'; la etiqueta <etiq> corresponde a la dirección de la primera instrucción que comenzará a ejecutar el simulador.
- Todo programa debe finalizar con una directiva '.END' (puede haber más de una de estas directivas). Esta directiva es en realidad una instrucción HALT camuflada, que le indica al simulador que el programa ha terminado. Como es una instrucción, ocupa memoria, y por eso debe ser la última instrucción del programa. Si no fuese así, es preciso considerar que ocupa 16 bits en la memoria de cara a tener en cuenta posibles traducciones a lenguaje máquina del programa y a conocer la ubicación física de cada instrucción.
- Las etiquetas se definen como <etiq:>, pudiendo estar la instrucción a la que referencian en la misma línea o en una nueva (a continuación, con tantos espacios en blanco o saltos de línea como sean necesarios). Una etiqueta no puede comenzar por el carácter '_'.
- Una etiqueta en el código se puede usar para referenciar la posición de memoria en la que está una instrucción o un dato (la escrita inmediatamente a continuación).
- Se pueden definir constantes que no ocupan memoria, pero que sí aparecen en la tabla de símbolos: por ejemplo, B=13.
- Se pueden poner comentarios. Un comentario es cualquier cosa que va después de un ';' y llega hasta el final de la línea.
- Existen las directivas '.RW' y '.DW', que permiten reservar espacio de memoria para variables o reservar espacio asignando valores respectivamente. En la presente versión del compilador se soporta que los números incluidos en estas directivas

puedan ser definidos, usando la notación explicada entre paréntesis para cada ejemplo, como decimales (242), binarios (1010b) o hexadecimales (0xA001).

- Existe la directiva '.ORG *dir'*. Esta directiva hace que la instrucción que se encuentra inmediatamente a continuación se traduzca como si se almacenase en la dirección de memoria *dir*. El compilador traducirá las siguientes instrucciones a continuación de ésta. Esta directiva es útil si quiere dividirse la memoria en varias zonas (por ejemplo, una zona de datos y una zona de instrucciones).
- Una macro es un grupo de instrucciones asociado con un nombre y con algunos posibles parámetros. Su homónimo en un lenguaje de alto nivel sería una función declarada como *inline*. Por tanto, para ejecutar una macro no se inserta una instrucción de llamada (como se haría con una subrutina), si no que se ejecuta el código de la macro allí donde se requiere, substituyendo adecuadamente los parámetros.
- Las macros se pueden definir en un fichero aparte, que se selecciona en el menú *Compilador -> Fichero de Macros por defecto*, y que por defecto es el fichero "macros.mr". El fichero de macros contiene una especificación de cada una de las macros. No obstante, las macros pueden ser también definidas al comienzo del fichero '*.mr'* que contiene el programa.
- Es conveniente tener todos los ficheros que intervendrán en la compilación en el mismo directorio.
- Para especificar una macro en el fichero, el formato es el siguiente:
  .DEF <nombre macro> <lista parámetros>
  <lista instrucciones de la macro, usando si es necesario los parámetros>
  .ENDDEF
- Para llamar a una macro desde un programa sólo hay que escribir su nombre y los parámetros que pueda necesitar.

CAMBIOS CON RESPECTO A VERSIONES ANTERIORES

En la máquina se ha incluido un nuevo flag: el bit de overflow V almacenado en el registro RV. Este bit se pone a 1 cuando el resultado de la última operación aritmética realizada produce desbordamiento, considerando los datos como enteros de 16 bits representados en complemento a dos. La inclusión de este bit ha sido necesaria para garantizar que los programas se ejecutan correctamente. En versiones anteriores las instrucciones de salto BG, BGE, BL y BLE podían provocar saltos erróneos si en la operación anterior se había producido desbordamiento. Teniendo en cuenta la simplicidad de la MR, hemos preferido añadir este bit y modificar las instrucciones de salto antes mencionadas para que salten teniéndolo en cuenta, de forma que ninguno de los programas que puedan llegar a ejecutarse lo haga de forma errónea. Las instrucciones de salto antes mencionadas funcionan de la siguiente forma:

- BG: salta si Z=0 o N=0
- BGE: salta si N=0
- BL: salta si N<>V
- BLE: salta si Z=1 o N<>V