Microarquitectura EV19
Electrónica V

Instituto Tecnológico de Buenos Aires
4 de Junio, 2019

***Abstract*—Desarrollo de un microprocesador RISC para la puesta en práctica de los conceptos básicos aprendidos en la materia.**

I. Introducción

A. Microarquitectura

El procesador EV19 es un procesador programable con estructura pipeline de 5 etapas. Cuenta con memoria de programa separada de la memoria de datos. Tiene 32 registros General Purpose (R0 a R31), 2 registros auxiliares (A32 y A33) y 1 registro Working Register dedicado a la comunicación con la memoria RAM (de xxxx posiciones de 16 bit cada una). Dicho procesador tiene un Conjunto Reducido de Instrucciones (RISC) implementado mediante una microarquitectura basada en microcódigo. Las instrucciones y el hardware han sido diseñados de modo que por cada instrucción exista una única microinstrucción. Las microinstrucciones se almacenan en una MicroInstruction ROM que es indexada con el código de operación (opcode) de la instrucción actualmente en curso. Esta MicroInstruction ROM provee la microinstrucción utilizada por los registros de microinstrucción (MIRs) del pipeline.

II. Descripción

A continuación se explica detalladamente el funcionamiento de cada módulo y qué tarea desarrolla.

A. Cartilla de instrucciones

Los opcodes utilizados son de xx bits. Las instrucciones están divididas en los siguientes grupos: instrucciones de salto, movimiento con la memoria RAM, operaciones aritméticas y lógicas y operaciones de asignación entre registros. Los opcode utilizados son los de la cartilla original, respetando la consigna otorgada.

B. Microinstrucción en ROM

El código binario de cada instrucción escrita por el programador produce una microinstrucción determinada. La misma se puede formar: a partir de la microinstrucción contenida en ROM únicamente, o complementándose parcialmente con el opcode introducido. Cada microinstrucción, a su vez, describe la palabra de control. Esta palabra se encuentro conformada por:

1) Alu Control (4 bits): le indican a la ALU qué operaciones debe realizar. Sólo interviene en operaciones lógicas y aritméticas.
2) SH (2 bit): indican al Shifter que se encuentra a la salida de la ALU cómo debe comportarse. Sólo interviene en las instrucciones de shifteo hacia la izquierda o hacia la derecha (pudiendo optar por shift hacia la derecha lógico o artimético).
3) A y B (6 bit c/u): son señales recibidas por el banco de registros para saber cuál/es de ellos debe utilizarse en determinada operación.
4) C (35 bits): Cada bit corresponde a cada uno de los registros incorporados en la microarquitectura, pudiéndose escribir el resultado de la operación en la cantidad de registros deseados. Cómo se respetó la cartilla otorgada por la cátedra esta función no es utilizada, pero se encuentra disponible en hardware.
5) T (7 bit): son señales que usa solamente el pipeline y sirven para detectar dependencias entre operaciones consecutivas para frenar el avance del programa por seguridad.
6) M (2 bit): muestran la operación efectuada en la memoria. El bit más significativo indica la escritura de memoria y el menos significativo, la lectura de memoria.
7) KMux (1 bit): señal que se utiliza en el KMux para seleccionar el operando que irá depositado en el latch A. Dicho operando puede venir de un registro del banco correspondiente o bien de la memoria del programa.
8) Dadd (6 bit): es la dirección de memoria de datos donde se efectuará una lectura o escritura.

Una vez obtenida la microinstrucción, se debe recorrer el pipeline. Cada uno de los campos detallados es utilizado únicamente por una etapa del pipeline.

C. Pipeline

El pipeline está compuesto por 5 etapas: Fetch, Decode, Operand, Execute y Retire. En la etapa Fetch, se obtiene la instrucción del programa en el Instruction Register. Durante la etapa Decode, se definen todos los campos de la microinstrucción, por este motivo, es a partir de este punto que las unidades de control, internas al pipeline, comienzan a ser funcionales. En las etapas posteriores se verifica si existen dependencias entre instrucciones sucesivas o en la cercanía, y se ejecutan las operaciones correspondientes a la ALU, el shifter, read/write y los registros utilizados por los buses.

Cada etapa está formada por flip-flops D que retienen los datos de la microinstrucción hasta pasarlos a la siguiente etapa en el siguiente pulso de clock.

Para controlar el pipeline, existen dos unidades de control. Uno de ellos, denominado UC2, se encarga de decidir si hay dependencia o no entre instrucciones sucesivas, y activa una señal HOLD, si es necesario, indicando al resto del pipeline

que debe esperar a que terminen de ejecutarse las instrucciones que están siendo procesadas antes de seguir con las siguientes. El bloque UC1 consiste en un MUX que deja pasar los datos de la microinstrucción o fuerza en la salida ceros, con el único objetivo de generar 'distancia' entre instrucciones con posible conflicto, según lo indique la señal de HOLD.

III. Conclusiones

The conclusion goes here.