# Procesador

Se ha diseñado e implementado manualmente un procesador con arquitectura RISC basado en la arquitectura de los procesadores DLX estudiados durante el grado en ingeniería de computadores[1][2]. Se trata de un procesador con un ancho de palabra de 32 bits y una segmentación en 5 etapas.

La implementación ha sido adaptada para poder ejecutar instrucciones del repertorio ARM. En concreto, se permite ejecutar un subconjunto del juego de instrucciones THUMB-2. Este juego de instrucciones es utilizado principalmente por los procesadores de la gama ARM CORTEX M. Otras gamas de procesadores ARM también soportan el juego de instrucciones THUMB-2 además de las instrucciones ARM.

## Estructura

* El banco de registros dispone de 16 registros (R0 .. R15) de propósito general con un tamaño de 32 bits. Estos registros se pueden utilizar para guardar datos leídos de memoria y enviar los valores a memoria. Podemos trabajar con los valores que tengan almacenados ejecutando operaciones sobre ellos. El registro R15 es accesible de forma limitada puesto que el identificador de este registro se utiliza para diferenciar unas instrucciones de otras.
* Además existe el registro del contador de programa (PC). Este registro especial almacena la dirección de memoria de la instrucción que debe ejecutarse a continuación. Se incrementa automáticamente en 4 cada ciclo. Y solo se puede alterar este mecanismo por medio de instrucciones de control.
* La memoria es accesible por palabras de 32 bits. Es decir, todo acceso a memoria carga o almacena 32 bits. Para acceder a memoria se dispone de instrucciones de lectura y escritura de una palabra con direccionamiento relativo a registro base. La dirección de acceso se calcula con un registro base al que se le suma un inmediato de 12 bits.
* Las instrucciones se componen de 32 bits con formato variable.

## Tipos de datos

Para simplificar la arquitectura del procesador, se ha limitado el tamaño de datos a palabras completas de 32 bits. Se trabaja con un bus de ancho de palabra de 32 bits donde todos los bits cargados tienen valor.

## Instrucciones

El juego de instrucciones elegido está compuesto por instrucciones de 32 bits con formato variable. El formato de las diferentes instrucciones se explicarán más adelante.

A diferencia de la arquitectura DLX y sus instrucciones sencillas, se ha utilizado un juego de instrucciones más complejo, lo que requiere una unidad de control compleja para decodificar las instrucciones.

El procesador implementado es capaz de ejecutar 3 tipos de instrucciones:

* Accesos a memoria
* Operaciones sobre registros

1. Operaciones con dos registros
2. Operaciones con un registro y un inmediato

* Operaciones de salto

Ahora se explica brevemente los diferentes tipos de instrucciones. Más adelante se expondrán las instrucciones con más detalle, explicando los campos de cada una.

### Accesos a memoria

Las instrucciones de acceso a memoria son necesarias cuando se requiere cargar (load) un dato desde la memoria al banco de registros, o almacenar (store) el valor de un registro en la memoria.

Es posible acceder a las direcciones de memoria direccionadas por byte. En esta implementación se está obligado a cargar valores de 4 bytes de tamaño (tamaño de palabra). Es recomendable utilizar direcciones de memoria que sean múltiplos de 4.

Para el cálculo de la dirección de carga o almacenamiento se ha implementado un único modo de direccionamiento. Registro base "Rn + imm12".

La dirección base se obtiene del registro Rn, y se suma un inmediato de 12 bits obtenido de la instrucción.

### Procesamiento de datos

Las instrucciones de procesamiento realizan cálculos aritméticos y lógicos. Se aplican sobre dos operandos y el resultado (si existe) se almacena en un registro.

Dependiendo de la instrucción los operandos pueden ser:

#### Operaciones con dos registros

Los datos con los que se trabaja se extraen de dos registros codificados en 4 bits.

Al utilizar el registro R15 se deben tener en cuenta ciertas restricciones. Este registro se utiliza para diferenciar ciertas operaciones de otras. Por ejemplo, si el código de operación es "0010" y el registro origen Rn es R15 ("1111") la operación ejecutada será la operación "MOVE", si el registro Rn es cualquier otro, se ejecutará una "Ó lógica" (operación or).

#### Operaciones con un registro y un inmediato

El conjunto de operaciones con inmediato se limita a cuatro operaciones. Se permite mover un inmediato de 16 bits a un registro, pudiendo elegir si los dos bytes se almacenarán en los 16 bits más significativos o en los 16 bits menos significativos. Además se permite sumar o restar un inmediato de 12 bits a un registro.

### Operaciones de control

Las operaciones de control intervienen en la ejecución normal del programa. Se utilizan para modificar el valor del registro del contador de programa.

Los procesadores ARM combinan instrucciones de 32 bits con instrucciones de 16 bits. Por ello, el inmediato con el desplazamiento es desplazado un bit hacia la izquierda. En nuestro caso nos debemos asegurar de codificar las instrucciones con un 0 en el bit menos significativo del inmediato. Con esto se evita acabar en una dirección equivocada, y leer dos mitades de dos instrucciones distintas.

Existen dos tipos de instrucciones de salto. El primero es el salto incondicional y permite sumar un entero de 24 bits al valor del contador de programa y almacenar el resultado en el mismo.

La segunda operación de control es el salto condicional. Para este tipo de salto se reduce el tamaño del inmediato a 20 bits. El tamaño del inmediato se ve afectado porque se requiere un campo extra de 4 bits para la condición de salto.

Previamente a un salto condicional se debe ejecutar una operación de comparación. Esta operación activa los flags de la unidad aritmético-lógica dependiendo del resultado de la comparación, y estos se mantienen hasta que se vuelva a ejecutar otra comparación. Los flags se comparan a la condición de salto y en caso de coincidir, se realiza el salto. Si no se ejecuta la comparación, el estado de los flags es desconocido y el procesador se comportará de manera no controlada.

## Arquitectura

### Ruta de datos

En la Ilustración 1 se muestra el diseño de la ruta de datos del procesador. La ruta de datos del procesador consta de 5 etapas. Estas etapas se explican a continuación.

D:\TFG\TFG\Memoria\Imagenes\usar\Procesador\Procesador segmentado.tif

Ilustración . Ruta de datos de procesador segmentado

1. Búsqueda de instrucción

La primera etapa es la encargada de cargar las instrucciones de memoria y transmitirlas a la siguiente etapa. De forma paralela calcula la dirección de la siguiente instrucción. Para estas tareas sus componentes son:

* El acceso a la memoria de instrucciones se realiza a través de un módulo independiente. Este módulo recibe la dirección de memoria y devuelve la instrucción a ejecutar. Actualmente este modulo es una memoria ROM. Esta memoria contiene las instrucciones del programa en código binario.
* El contador de programa es el registro que almacena la dirección de memoria donde se localiza la instrucción.
* Un sumador encargado de incrementar en 4 el contador de programa.
* Un multiplexor encargado de seleccionar la siguiente dirección. En caso de haberse ejecutado un salto, se seleccionará la dirección calculada en la etapa de ejecución. En caso contrario, la ejecución normal, la siguiente instrucción será la salida del sumador de la etapa.

1. Decodificación

En la etapa de decodificación se analiza la instrucción y se obtienen los datos necesarios para realizar las operaciones correctamente. Para decodificar las instrucciones dispone de:

* Un banco de registros que contiene los registros que almacenan los datos con los que se trabaja.
* Un circuito combinacional de extensión de signo. Este circuito obtiene el inmediato codificado dentro de la instrucción que se está ejecutando.

1. Ejecución

En la etapa de ejecución se realizan los cálculos aritméticos o lógicos sobre los datos obtenidos del banco de registro y del circuito de extensión de signo. Para realizar los cálculos incluye:

* Una Unidad Aritmético-Lógica (ALU) para las operaciones sobre los registros. Junto a la ALU aparece un multiplexor que permite seleccionar el origen de los datos.
* Un sumador para el cálculo de la dirección de salto.

1. Acceso a memoria

En la etapa de memoria se realizan intercambios de datos con la memoria principal.

* Memoria de datos. Es el módulo encargado de la interacción con los datos almacenados en memoria. Permite cargar los datos de memoria en los registros y almacenarlos después de trabajar con ellos. Actualmente esta memoria está implementada como una memoria RAM de acceso directo de un ciclo.

1. Escritura en registros

En la etapa final del procesador se escriben los resultados calculados por la ALU, o los datos cargados de memoria en el banco de registros.

* Contiene un multiplexor para seleccionar el origen de los datos que se almacenarán en el registro.

### Ruta de control

Para completar el procesador es necesario una unidad de control principal. Ésta unidad se encarga de analizar la instrucción que debe ejecutarse y prepara las señales de control para el resto de módulos. En la Ilustración 2, podemos observar el procesador con la unidad de control y las señales de control. A continuación se enumeran las señales de control y se explica su funcionalidad.

Procesador segmentado Ruta de control.tif

Ilustración . Ruta de control de procesador segmentado

1. Búsqueda de instrucción

* **PCSrc:** Esta señal se calcula en la etapa de acceso a memoria. La señal PCSrc indica si se cargará un salto o se continúa la ejecución normal del programa.
* ~~Esta señal se deriva de una comparación entre las condiciones de salto de la instrucción y el valor de las ALU flags calculadas por instrucciones anteriores. Esta comparación se realiza en la etapa de memoria. La señal PCSrc selecciona el origen para la siguiente dirección que se cargará en el contador de programa.~~

1. Decodificación

* **RegWrite:** Esta señal proviene de la etapa de escritura en registros.

1. Ejecución

* **ALUSrc:** Selecciona el origen del segundo operando de entrada para la unidad aritmético-lógica.
* **ALUop:** Selecciona la operación que se aplica en la unidad aritmético-lógica.

1. Acceso a memoria
   * **MemWrite:** Indica que la instrucción debe acceder a memoria en modo escritura.
   * **MemRead:** Indica que la instrucción debe acceder a memoria en modo lectura.
   * **BranchCond:** Condición necesaria para activar la señal de control "PCSrc".
2. Escritura en registros
   * **MemtoReg:** Indica si el resultado de la instrucción tiene origen en la memoria de datos o en la unidad aritmético-lógica.
   * **RegWrite:** Indica si el resultado de la instrucción debe almacenarse en el banco de registros.

## Formato de instrucciones

El juego de instrucciones implementado es un subconjunto de las instrucciones de la arquitectura Thumb-2.

En este apartado se explican las instrucciones implementadas con sus campos. Para más información sobre el juego de instrucciones THUMB-2 se acudir al manual de referencias[3].

Organizado en 3 tipos, el juego de instrucciones se divide en instrucciones de transferencia, instrucciones de operaciones e instrucciones de control de flujo.

Las instrucciones implementadas, divididas por grupo, son:

1. Transferencia

Instrucciones de acceso a memoria, LOAD y STORE de un único dato con desplazamiento.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Formato general | 1 | 1 | 1 | 1 | 1 | 0 | 0 |  | | | | | | | | |
| Rn + imm12 |  | | | | | | | S | 1 | Size | | L | Rn | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Formato general |  | | | | | | | | | | | | | | | |
| Rn + imm12 | Rt | | | | imm12 | | | | | | | | | | | |

|  |  |
| --- | --- |
| S | Extensión de signo (S==1) o extensión con cero (S==0) |
| Size | ??? Tamaño de dato cargado |
| L | Indica si es una operación LOAD (L==1) o STORE (L==0) |
| Rn | Registro con la dirección de carga |
| Rt | Registro destino del dato |
| imm12 | Desplazamiento de dirección |

Restricciones:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instrucción | S | Size | L | Rt |
| LDR | X | 0b0X | 1 | Not R15 |
| STR | 0 | Not 0b11 | 0 | Not R15 |

Notas:

Las instrucciones LOAD y STORE se identifican por los bits [31..25] y deben ser "111\_1100".

El origen de la dirección debe ser un registro (Rn), previamente cargado con el valor dirección del dato deseado.

1. Operaciones

Las operaciones se separan según el tipo de operandos que se apliquen. El operando A siempre es obtenido de un registro. Mientras que el operando B puede ser el valor de un segundo registro o puede formar parte de la instrucción.

* 1. Con inmediato

Se han implementado dos tipos de operaciones con inmediato, operaciones de carga de inmediato y operaciones aritméticas.

Las operaciones aritméticas se componen principalmente de la operación, un registro origen, un inmediato de 12 bits y un registro destino. Las operaciones implementadas son la suma y la resta.

Las operaciones de carga de inmediato, se componen principalmente de la operación, un inmediato de 16 bits y el registro destino. Las operaciones implementadas son "move top" (carga en los 16 bits más significativos) y "move wide" (carga en los 16 bits menos significativos).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Formato general | 1 | 1 | 1 | 1 | 0 |  | | | | | | | | | | |
| Add, Subtract, plain 12-bit immediate |  | | | | | i | 1 | 0 | OP | 0 | OP2 | | Rn | | | |
| Move, plain 16-bit immediate |  | | | | | i | 1 | 0 | OP | 1 | OP2 | | imm4 | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Formato general | 0 |  | | | | | | | | | | | | | | |
| Add, Subtract, plain 12-bit immediate |  | imm3 | | | Rd | | | | imm8 | | | | | | | |
| Move, plain 16-bit immediate |  | imm3 | | | Rd | | | | imm8 | | | | | | | |

|  |  |
| --- | --- |
| OP:OP2 | Código de operación |
| Rn | Registro origen para operando 1 |
| Operaciones con inmediato de 12 bits | |
| Rd | Registro destino |
| i:imm3:imm8 | Inmediato para operaciones de 12 bits |
| Operaciones con inmediato de 16 bits | |
| imm4:i:imm3:imm8 | Inmediato para operaciones de 16 bits |

Restricciones:

No hay restricciones especiales.

Notas:

Las operaciones con inmediato se identifican por los bits [31..27] y el bit [15], estos deben tener los valores "1\_1110" y "0" respectivamente.

Tabla de operaciones con inmediatos de 12 bits

|  |  |  |  |
| --- | --- | --- | --- |
| Operación | OP | OP2 | |
| ADD | 0 | 0 | 0 |
| SUB | 1 | 1 | 0 |

Tabla de operaciones con inmediatos de 16 bits

|  |  |  |  |
| --- | --- | --- | --- |
| Operación | OP | OP2 | |
| MOVT | 1 | 0 | 0 |
| MOV | 0 | 0 | 0 |

* 1. Con registro

Las operaciones que hacen uso de dos registros son aritméticas (sumar, restar y mover), lógicas (and, or y or exclusiva) y de comparación que activan diferentes flags (Negativo, Cero). En el caso de una comparación no se modifican los registros.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Formato general | 1 | 1 | 1 |  | 1 | 0 | 1 |  | | | | | | | | |
| Data processing: Constant shift |  | | | 0 |  | | | OP | | | | S | Rn | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Formato general |  | | | | | | | | | | | | | | | |
| Data processing: Constant shift | SBZ | imm3 | | | Rd | | | | imm2 | | type | | Rm | | | |

|  |  |
| --- | --- |
| OP | Código de operación |
| S | Actualiza los flags si se cumplen las condiciones |
| Rn | Registro origen para operando 1 |
| SBZ | Debería ser cero ("Should be Zero") |
| imm3:imm2 | Desplazamiento |
| Rd | Registro destino |
| type | Tipo de desplazamiento usado |
| Rm | Registro origen para operando 2 |

Restricciones:

Existen restricciones para cada operación, se pueden consultar junto a los códigos de operación más adelante.

Notas:

Las operaciones con inmediato se identifican por los bits [31..29] y [28..25], estos deben tener los valores "111" y "101" respectivamente.

En esta implementación los campos "S", "imm3:imm2" y "type" no se utilizan.

Las instrucciones implementadas junto con sus respectivos códigos de operación son los siguientes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operación | Código | | | | Notas |
| ADD | 1 | 0 | 0 | 0 |  |
| AND | 0 | 0 | 0 | 0 |  |
| CMP | 1 | 1 | 0 | 1 | (Rd=="1111", S==1) |
| EOR | 0 | 1 | 0 | 0 |  |
| MOV | 0 | 0 | 1 | 0 | (Rn=="1111") |
| ORR | 0 | 0 | 1 | 0 |  |
| SUB | 1 | 1 | 0 | 1 |  |

1. Control de flujo

Las instrucciones de operación de flujo son aquellas que pueden alterar el contador del programa. En esta implementación se aceptan los saltos condicionales, haciendo uso de los flags activados por las operaciones de comparación, y los saltos incondicionales.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Formato general | 1 | 1 | 1 | 1 | 0 |  | | | | | | | | | | |
| Salto |  | | | | | S | offset[21:12] | | | | | | | | | |
| Salto condicional |  | | | | | S | cond | | | | offset[17:12] | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Formato general | 1 |  | | | | | | | | | | | | | | |
| Salto |  | 0 | I1 | 1 | I2 | offset[11:1] | | | | | | | | | | |
| Salto condicional |  | 0 | J1 | 0 | J2 | offset[11:1] | | | | | | | | | | |

|  |  |
| --- | --- |
| S | Extension de signo (S==1) o Extensión con cero (S==0) |
| offset | Desplazamiento del salto |
| I1, I2 | bits 23 y 22 del desplazamiento, respectivamente |
| J1, J2 | bits 19 y 18 del desplazamiento, respectivamente |
| cond | Condiciones necesarias para realizar el salto |

Restricciones:

No hay restricciones especiales.

Notas:

Las instrucciones de control de flujo se reconocen por los bits [31..27] cuando tienen el valor "1\_1110" y por el bit [15] que debe ser "1".

Los bits [1..0] del campo "cond" (bits [23..22] de la instrucción) indican los flags N y Z respectivamente.

Por completar:

-- Implementación del procesador

-Tolerancia a fallos

-- Votadores

-- Procesador Tolerante a fallos

-- Inserción de fallos

Bibliografia

[1] D. E. L. a S. Antiguas and C. De, *Manual DLX*. pp. 1–14.

[2] *Un enfoque cuantitativo*. .

[3] E. E. R.B.J. Brinkgreve, W.M. Swolfs, “ARM Thumb 2 Reference Manual,” 2011.