PEC - Ampliación

Unidad SIMD y sistema operativo multiproceso

Oussama Chbaly, Marc Solé,
Andrea Querol, Pablo Vizcaino

10 de Mayo de 2019

Índice

1. SIMD

1.1. Estructura

Hemos decidido crear una unidad separada para las operaciones SIMD con la intención de crear una unidad que podría aplicarse a diferentes procesadores realizando cambios mínimos en estos. Por lo tanto, los registros SIMD estarán en un banco de registros dentro de esta unidad y no en el banco de registros convencional del procesador.

Nuestra intención es que dichos registros sean de 128 bits (como ocho registros generales de nuestro procesador) y tengamos un total de 8 registros SIMD. De todas formas, tal vez tenemos que reducir el tamaño de los registros SIMD para lograr cargar/ almacenar los registros de/ a memoria en un único ciclo del procesador.

La unidad también contará con una ALU que operará con los registros SIMD, tratando todos los datos en paralelo.

Las operaciones que hará la ALU se describen en la siguiente sección.

1.2. Operaciones

1. **ADDS:** Suma el valor de dos registros SIMD y guarda el resultado en otro.
2. **SUBS:** Resta el valor de dos registros SIMD y guarda el resultado en otro.
3. **ANDS:** Hace la and lógica de dos registros SIMD y guarda el resultado en otro.
4. **ORS:** Hace la or lógica de dos registros SIMD y guarda el resultado en otro.
5. **XORS:** Hace la xor lógica de dos registros SIMD y guarda el resultado en otro.
6. **NOTS:** Hace la not lógica de un registro SIMD y guarda el resultado en otro.
7. **MOVRS:** Mueve un registro general a uno de los ocho conjuntos de 16 bits que forman un registro SIMD.
8. **MOVSR:** Mueve un conjunto de 16 bits de un registro SIMD a un registro general.

1.3. Acceso a memoria - Load y Stores

Debido a que la memoria es tanto más rápida que el procesador, intentaremos hacer los 8 accesos necesarios para llenar un registro SIMD en un único ciclo del procesador.

1. **LDS:** Carga en el registro SIMD el contenido de memoria de la dirección indicada alineada a 2, transfiere tantos bits como el tamaño del registro.

2. **STS:** Almacena el contenido del registro SIMD en la dirección de memoria indicada alineada a 2, transfiere tantos bits como el tamaño del registro.

1.4. Codificación

Hemos decidido utilizar los códigos de operación de las instrucciones de coma flotante, ya que no están implementadas en nuestro procesador. También mantendremos una distribución similar de los bits de la instrucción a las ya implementadas, para facilitar la decodificación.

15 14 13 12	11 10 9	8 7 6	5 4 3	2 1 0	Operación	Instrucción
1001	RSd	RSa	f	RSb	Arit-Log y mov SIMD	ADDS, SUBS, ANDS, ORS, XORS, NOTS, MOVRS, MOVSR
1011	RSd	RSa	n		Load SIMD	LDS
1100	RSb	RSa	n		Store SIMD	STS

1.5. Juegos de prueba

Para poder testear el correcto funcionamiento de nuestra unidad SIMD realizaremos juegos de pruebas. Estas pruebas consistirán en un programa en C basado en bucles *for* que operen sobre diferentes vectores

Habrá un bucle para cada instrucción a probar. En el caso de suma, resta, and, or, xor y not, los bucles iterarán sobre vectores. Por lo tanto, estos vectores estarán almacenados en memoria y se probará que las instrucciones load y store funcionan. Para comprobar las instrucciones mov, no se iterará sobre un vector, sino sobre variables locales, de manera que estarán almacenadas sobre registros generales.

2. Sistema Operativo

El sistema operativo debe ser capaz de ejecutar dos tareas concurrentemente. Esto implica, que ha de cambiar de contexto entre ellas.

Para ello tendremos que tener un *quantum* fijado para que los procesos sólo se ejecuten durante este tiempo, y, al agotarse, se produzca el cambio de contexto. Para mantener un contador de este *quantum* usaremos la interrupción de *timer*.

Por lo tanto deberemos tener una rutina para tratar las diferentes interrupciones y derivarlas al tratamiento correspondiente. Esto debe hacerse para las excepciones implementadas.

Así pues el SO deberá guardar el contexto antes de entrar al sistema y restaurarlo al salir, y a la vez realizar un cambio de contexto cuando sea necesario.

3. Panificación

Como realizaremos una ampliación de los procesadores elaborados en la asignatura durante el curso, consideraremos que la fecha inicial de este proyecto es el primer día de clase posterior a la entrega del procesador. Así pues, tendremos que realizar las tareas entre el 20 de mayo y el 20 de junio, fecha de la entrega final.

Para facilitar el avance del proyecto dividiremos la ampliación de la unidad para operaciones SIMD en diferentes tareas. Para aquellas en las que sea necesario, se harán juegos de pruebas para validar su correcta implementación. También se realizará una breve documentación para facilitar el redactado de la memoria final.

Paralelamente se desarrollará el sistema operativo, que debe ser capaz de hacer cambios de contexto entre dos procesos. Esta parte también será debidamente testeada mediante juegos de pruebas y documentada en la memoria final.

Para dividir el trabajo, el grupo formado por Oussama y Marc se encargará de realizar la ampliación con las instrucciones SIMD, mientras que Andrea y Pablo programarán el sistema operativo. Sin embargo estas asignaciones pueden variar durante el desarrollo del proyecto en función de las situaciones que se produzcan.

A continuación se muestra un resumen de la planificación en forma de diagrama de Gantt (Figura 1).

3.1. Diagrama de Gantt

Nom	Durada	Inici	Fi	Predecessors
Instrucciones SIMD	30dies	19/05/2019	17/06/2019	
Banco de registros	2dies	19/05/2019	20/05/2019	
Diseño del banco de registros SIMD	8hores	19/05/2019	19/05/2019	
Conexiones del módulo	4hores	20/05/2019	20/05/2019	3
Documentación	4hores	20/05/2019	20/05/2019	4
Instrucciones MOVRS i MOVSR	7dies	21/05/2019	27/05/2019	2
Modificaciones en la unidad de control	4dies	21/05/2019	24/05/2019	
Juegos de pruebas	1dia	25/05/2019	25/05/2019	7
Documentación	2dies	26/05/2019	27/05/2019	8
Operaciones SIMD	9dies	28/05/2019	05/06/2019	6
Diseño de la ALU del módulo SIMD	3dies	28/05/2019	30/05/2019	
Modificaciones en la unidad de control	2dies	31/05/2019	01/06/2019	11
Juegos de pruebas	2dies	02/06/2019	03/06/2019	12
Documentación	2dies	04/06/2019	05/06/2019	13
Acceso a memoria	12dies	06/06/2019	17/06/2019	10
Controlador de memoria para SIMD	5dies	06/06/2019	10/06/2019	
Instrucciones LDS y STS	3dies	11/06/2019	13/06/2019	16
Juegos de pruebas	2dies	14/06/2019	15/06/2019	17
Documentación	2dies	16/06/2019	17/06/2019	18
Sistema Operativo	29dies	20/05/2019	17/06/2019	
Preparación del entorno	4dies	20/05/2019	23/05/2019	
Programación del SO	20dies	24/05/2019	12/06/2019	21
Juegos de prueba	2dies	13/06/2019	14/06/2019	22
Documentación	3dies	15/06/2019	17/06/2019	23
Redacción de la memoria final	3dies	18/06/2019	20/06/2019	1,20

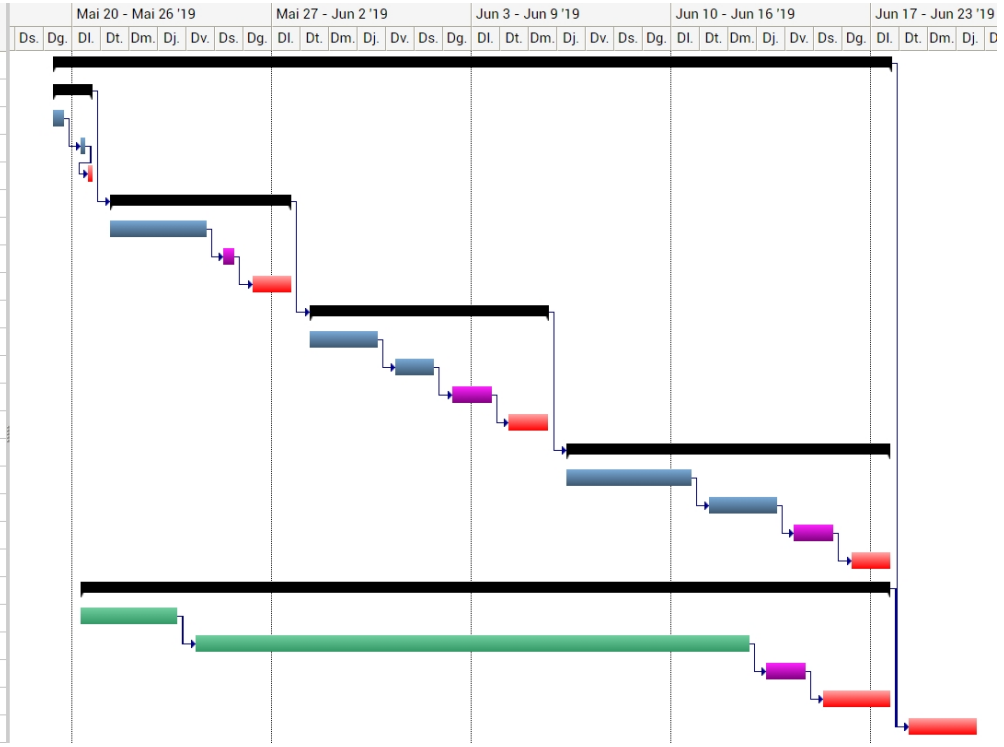

Figura 1: Diagrama de Gantt