Documentación de diseño RSA

Jasson Rodríguez Méndez Marco Herrera Valverde Kenneth Hernández Salazar
Edgar Chaves González

Julio 2020

1 Requerimientos de sistema

El problema consiste en la desencriptación de una imagen encriptada en formato RSA, de la cual se tiene una llave pública y una privada, este método se basa en que el cliente envía su llave pública a un servidor y solicitar la información, dicho servidor encripta la información y la envía de vuelta al cliente el cual la desencripta utilizando la llave privada.

El proyecto consiste en la la implementación de una arquitectura y microarquitectura destinada a la desencriptación de la información, en este caso una imagen de dimensiones 320x320. En cuanto a la arquitectura del ISA se requiere la elaboración de un conjunto de instrucciones que permitan la solución al problema planteado considerando modos de direccionamiento, tamaño y tipo de datos, tipo y sintaxis de instrucciones, registros y sus nombres y la codificación y descripción funcional de las instrucciones, a su vez se debe de crear un compilador que permita mover las instrucciones a un formato binario y la creación de un código que permita la desencriptación de la imagen con las instrucciones previamente mencionadas.

La microarquitectura o hardware debe de ser implementada en la tarjeta Terasic DE1-SoC-MTL2 y presentar segmentación, esta a su vez debe de permitir la visualización de la imagen mediante el uso de VGA y la ejecución de las instrucciones determinadas en la arquitectura.

2 Soluciones de problema

2.1 Procesador MIMD multi-nucleo.

Como nos dice S. Irabashetti[1] los procesadores clasificados en la taxonomía de Flynn MIMD computan diferentes instrucciones sobre diferentes conjuntos de datos, estas instrucciones normalmente trabajan de forma independiente la una de la otra.

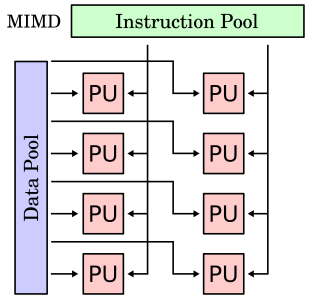

Figure 1: Imagen de la taxonomía MIMD, recuperada el 12 de julio del 2020 [2]

En el caso de esta solución se planea el cargar instrucciones en conjuntos de 4, como se ve en el diagrama 2 las unidades de ejecución están replicadas, mientras que las unidades de memoria, búsqueda, decodificación y escritura están acondicionadas a manejar 4 instrucciones diferentes. En medio de cada etapa se encuentran los diferentes registros en los cuales en cada ciclo de reloj es cargada la instrucción que se encuentra en la etapa previa.Esta solución se concentra en el tratar de ejecutar la mayoría de instrucciones posibles por ciclo de reloj e implementar paralelismo de instrucciones, de esta forma decodificando la imagen de una forma mas eficiente.

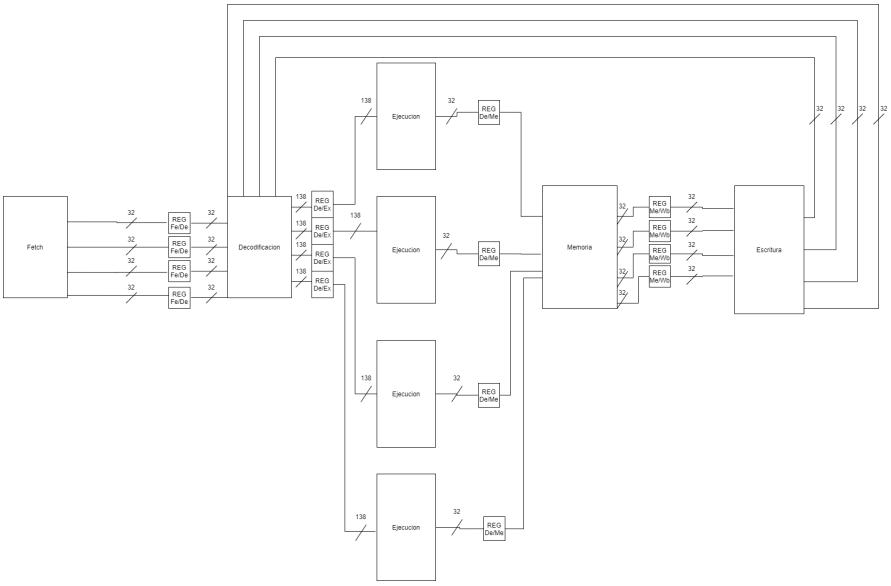

Figure 2: Diagrama de bloques del procesador MIMD multi nucleo.

2.2 Procesador MIMD vectorial

Esta solución apreciada en la imagen 3 implementa segmentación con la implementación de registros en medio de cada una de sus etapas, esta microarquitectura también implementa registros vectoriales de 8 elementos y una unidad de ejecución vectorial para el rápido procesamiento de conjuntos de datos extraídos de la imagen codificada, pero solo realiza la búsqueda de una instrucción por cada ciclo de reloj, además de esto, presenta una memoria condicionada a la escritura de dichos registros vectoriales para su escritura eficiente en un ciclo, esta opción se basa en que el problema es la desencriptación de muchos datos, por lo que operar sobre estos en conjuntos ayudaría a realizar mejor la tarea.

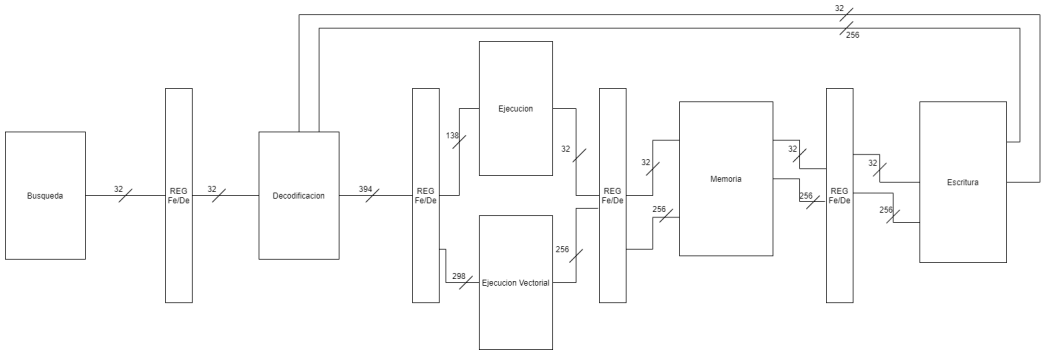

Figure 3: Diagrama de bloques del procesador MIMD vectorial.

3 Comparación

Table 1: Comparacion de características entre micro-arquitecturas.

Aspecto	**Procesador MIMD vectorial**	**Procesador MIMD multi-nucleo.**
Numero de datos procesados por instrucción	Las instrucciones vectoriales operan en 8 datos diferentes al mismo tiempo con el uso de una sola instrucción.	Por instrucción solo modifica o trabaja en un dato.
Instrucciones por ciclo	Carga de la sección de búsqueda una instrucción por ciclo.	Carga 4 instrucciones por cada ciclo de la sección de búsqueda
Acceso a memoria	Se necesita la implementación de una memoria que permita la escritura de todo un vector de datos y a la misma vez la lectura de estos.	Se necesita la implementación que permita la escritura y lectura de 4 instrucciones de forma simultanea, lo cual añade riesgos a la hora de por ejemplo dos instrucciones escriban en una sección de la memoria al mismo tiempo.
Riesgos y dependencias de control	Al estar cargando solo una instrucción de la sección de búsqueda en cada ciclo es fácilmente solucionable mediante la implementación de stalls.	Al estar cargando 4 instrucciones de forma simultanea los riesgos y dependencias se agravan ya que el control de que hacer con las instrucciones que han sido cargadas o la cantidad de stalls que hay que añadir para solucionar el riesgo aumentan conforme aumenta el paralelismo en el procesador.
Dependencias y riesgos de datos	Al estar cargando de la sección de búsqueda una instrucción por ciclo los riesgos entre estas en cada sección de la segmentación es mitigado mediante la implementación de una unidad de adelantamiento que compare los operando de cada sección, de esta forma adelantando los datos si es necesario.	Al estar ejecutando 4 instrucciones de forma paralela el mantener consistencia de los datos se complica mucho, ya que entre cada uno de los diferentes procesadores o unidades se debe de comunicar si se esta usado o escribiendo un dato que otra instrucción de forma paralela esta escribiendo o necesita para realizar operaciones, el uso de stalls seria una penalización muy grande en este caso ya que se deben de detener todos los procesadores que tengan algún riesgo.

Unidad de control	La unidad de control tendrá que manejar el uso de los diferentes registros vectoriales y la unidad de ejecución vectorial, la cual no difiere mucho de la unidad de ejecución normal, por lo que la lógica de esta no se ve muy afectada.	Se tendría que implementar una por cada núcleo y mantener una comunicación entre estas para mantener consistencia entre las instrucciones, por lo cual la lógica que esta contiene se ve muy afectada.
Uso de recursos	Este procesador utilizara mas de recursos que uno segmentado estándar ya que debe de acondicionarse al manejo de los registros vectoriales, los cuales son de aproximadamente 256 bits, además de que la unidad de ejecución vectorial ejecutara operaciones entre 2 de estos registros, lo cual nos lleva a operaciones de 256x256 bits, además de que en las unidades de escritura y memoria el manejo de estos registros conllevan la utilización de mas recursos.	Al tener acondicionadas todas las unidades básicas como lo son la de decodificación, memoria y ejecución acondicionadas para manejar 4 instrucciones de forma simultanea esto implicaría al solo tomar en cuenta ese factor el uso de aproximadamente 4 veces mas recursos que uno segmentado estándar, pero también esta el factor de que el manejo de riesgos debido a su lógica conllevaría al incremento de recursos usados.

4 Elección

Se eligió la micro-arquitectura del procesador MIMD vectorial. Este procesador es mas simple a la hora de manejo de riesgos como lo son los adelantamientos de datos o la consistencia de estos, ya que se esta trabajando sobre un solo flujo de instrucciones que pasan por cada etapa de la segmentación, contrario a como se haría con la otra arquitectura, en la cual se tienen que manejar entre cada una de las 4 instrucciones paralelas, lo que nos llevaría a tener una unidad de control o de manejos de riesgos que puede llegar a tener una lógica mas complicada de lo deseado.

```
ADD R1,R2,R3 -Inst1
SUB R2,R1,R3 -Inst2 RAW y WAR
MOV R3,4 - Inst3 WAR
SUB R2,R3,R4 - Inst4 WAW con Inst2
BEQ ETIQUETA -Inst5 Riesgo de control
ADD R4,R6,R7 -Inst6
```

Tomando el código anterior con algunas dependencias y riesgos, se puede ver que en caso de tomar la opción descartada el manejo de estos se puede llegar a complicar, se podría tomar inicialmente la ruta simple y usar stalls, el detalle es que esto romperia el propósito de la micro-arquitectura ya que estaríamos llenando 3 núcleos con stalls y ejecutando solo en uno. Suponiendo que carguemos las primeras 4 instrucciones, podemos ver la cantidad de dependencias que se tienen, el detectar cada una de estas y evitar el riesgo estructural conllevaría una lógica de control y detección muy complicada debido al paralelismo de las instrucciones.Por otro lado en la microarquitectura MIMD vectorial debido al flujo de instrucciones y su ejecución en orden se pierden los riesgos WAR y WAW, el riesgo que persiste es el RAW debido a la ejecución en orden previamente mencionada, pero esta es fácilmente resuelta mediante el uso de una unidad de adelantamiento, el que sea vectorial no llegaría a afectar esta unidad mucho debido a que lo que se compara para saber si algún registro esta siendo usado es su código, lo único que aumentaría son las cantidades de buses utilizados para adelantar los valores de los registros vectoriales.
El factor en cuanto al manejo de riesgos de control que nos llevo también a esta elección es que el uso de Stalls(mediante compilador) no penaliza tanto al procesador vectorial a como lo haría con el multi nucleo, ya que para el vectorial solo se necesitan 2 stalls como mínimo para tomar la decisión de salto, en cambio en el multi nucleo el incluir dichos stalls por software conlleva una lógica mas complicada a la hora de compilación debido a que este carga 4 instrucciones por ciclo.
Junto que a esta elección utiliza menos recursos también llega el determinante mas importante, su capacidad de trabajar en 8 diferentes datos con tan solo el uso de una instrucción. La mayoría del trabajo que realizara este procesador es la

ejecución de diferentes instrucciones relacionadas con operaciones sobre los datos de la imagen, por lo que el procesar conjuntos de datos de una forma mas rápida es una prioridad, ya que en el caso de que no fuera vectorial se harían por ejemplo 8 sumas diferentes, lo cual significa mas lineas de código y ciclos para la desencriptación.

References

[1] P. Irabashetti,"Parallel processing in processor organization ", January 2014

[2] PNGEGG,2020, https://www.pngegg.com/en/png-djkol