Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores

(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores

(Licentiate Degree Program in Computer Engineering)

Curso: CE-4302 Arquitectura de Computadores II

(Course: CE-4302 Computer Architecture II)

Proyecto 2: Modelo de procesador vectorial para encriptación de imágenes

(Project 2:Vector processor model for image encryption)

Realizado por:

Made by:

Luis Alfredo Piedra Esquivel, 2013241554

Profesor:

(Professor)

Ing. M.Sc. Jeferson González Gómez

Fecha: Cartago, Mayo, 10, 2019

(Date: Cartago, May, 10, 2019)

Arquitectura del set de instrucciones - ISA

Instrucciones soportadas

A continuación se brinda una descripción detallada de las instrucciones por categoría que componen el set propuesto e implementado. Se hace también la diferenciación entre las instrucciones que trabajan con registros vectoriales y las instrucciones que utilizan registros escalares.

Instrucciones vectoriales

- Aritméticas

Vector Add: operación suma de vectores, su mnemónico es VADD. Esta instrucción trabaja entre dos registros vectoriales de 8 bytes (64 bits) o un registro vectorial y un valor inmediato de 8 bits. Dicho comportamiento se logra mediante el uso un operando flexible. Pertenece a las instrucciones de formato **F**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VADD Vd, Va, Vb (2) VADD Vd, Va, Imm	FOR i := 0 a 7 j := i * 8 (1) Vd [7+j : j] = Va [7+j : j] + Vb[7+j : j] (2) Vd [7+j : j] = Va [7+j : j] + Imm[7 : 0] ENDFOR

Vector Subtract: operación resta de vectores, su mnemónico es **VSUB.**Esta instrucción trabaja entre dos registros vectoriales de 8 bytes (64 bits) o un registro vectorial y un valor inmediato de 8 bits. Dicho comportamiento se logra mediante el uso un operando flexible. Pertenece a las instrucciones de formato **F**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VSUB Vd, Va, Vb (2) VSUB Vd, Va, Imm	FOR i := 0 a 7 j := i * 8 (1) Vd [7+j : j] = Va [7+j : j] - Vb[7+j : j] (2) Vd [7+j : j] = Va [7+j : j] - Imm[7 : 0] ENDFOR

❖ Lógicas

Vector Exclusive OR: operación lógica de OR exclusiva, su mnemónico es **VXOR.** Esta instrucción trabaja entre dos registros vectoriales de 8 bytes (64 bits) o un registro vectorial y un valor inmediato de 8 bits. Dicho comportamiento se logra mediante el uso un operando flexible. Pertenece a las instrucciones de formato **F**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VXOR Vd, Va, Vb (2) VXOR Vd, Va, Imm	(1) Vd [63 : 0] = Va [63 : 0] ⊕ Vb[63 : 0] FOR i := 0 a 7 j := i * 8 (2) Vd [7+j : j] = Va [7+j : j] ⊕ Imm[7 : 0] ENDFOR

Vector AND: operación lógica AND, su mnemónico es **VAND.** Esta instrucción trabaja entre dos registros vectoriales de 8 bytes (64 bits) o un registro vectorial y un valor inmediato de 8 bits. Dicho comportamiento se logra mediante el uso un operando flexible. Pertenece a las instrucciones de formato **F**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VAND Vd, Va, Vb (2) VAND Vd, Va, Imm	(1) Vd [63 : 0] = Va [63 : 0] • Vb[63 : 0] FOR i := 0 a 7 j := i * 8 (2) Vd [7+j : j] = Va [7+j : j] • Imm[7 : 0] ENDFOR

Vector OR: operación lógica OR, su mnemónico es **VOR.** Esta instrucción trabaja entre dos registros vectoriales de 8 bytes (64 bits) o un registro vectorial y un valor inmediato de 8 bits. Dicho comportamiento se logra mediante el uso un operando flexible. Pertenece a las instrucciones de formato **F**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VOR Vd, Va, Vb (2) VOR Vd, Va, Imm	(1) Vd [63 : 0] = Va [63 : 0] | Vb[63 : 0] FOR i := 0 a 7 j := i * 8 (2) Vd [7+j : j] = Va [7+j : j] | Imm[7 : 0] ENDFOR

❖ Desplazamientos

Vector Shift Right: operación desplazamiento derecho de bits, su mnemónico es **VSHFR.** Esta instrucción trabaja entre dos registros vectoriales de 8 bytes (64 bits) o un registro vectorial y un valor inmediato de 8 bits. Dicho comportamiento se logra mediante el uso un operando flexible. Pertenece a las instrucciones de formato **F**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VSHFR Vd, Va, Vb (2) VSHFR Vd, Va, Imm	FOR i := 0 a 7 j := i * 8 (1) Vd [7+j : j] = Va [7+j : j] >> Vb[7+j : j] (2) Vd [7+j : j] = Va [7+j : j] >> Imm[7:0] ENDFOR

Vector Shift Left: operación desplazamiento izquierdo de bits, su mnemónico es **VSHFL.** Esta instrucción trabaja entre dos registros vectoriales de 8 bytes (64 bits) o un registro vectorial y un valor inmediato de 8 bits. Dicho comportamiento se logra mediante el uso un operando flexible. Pertenece a las instrucciones de formato **F**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VSHFL Vd, Va, Vb (2) VSHFL Vd, Va, Imm	FOR i := 0 a 7 j := i * 8 (1) Vd [7+j : j] = Va [7+j : j] << Vb[7+j : j] (2) Vd [7+j : j] = Va [7+j : j] << Imm[7:0] ENDFOR

Vector Circular Shift Right: operación desplazamiento circular derecho de bits, su mnemónico es **VCSHFR.** Esta instrucción trabaja entre dos registros vectoriales de 8 bytes (64 bits) o un registro vectorial y un valor inmediato de 8 bits. Dicho comportamiento se logra mediante el uso un operando flexible. Pertenece a las instrucciones de formato **F**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VCSHFR Vd, Va, Vb (2) VCSHFR Vd, Va, Imm	FOR i := 0 a 7 j := i * 8 (1) Vd [7+j:j] = Va [7+j:j] >>> Vb[7+j : j] (2) Vd [7+j:j] = Va [7+j:j] >>> Imm[7:0] ENDFOR

Vector Circular Shift Left: operación desplazamiento circular izquierdo de bits, su mnemónico es **VCSHFL.** Esta instrucción trabaja entre dos registros vectoriales de 8 bytes (64 bits) o un registro vectorial y un valor inmediato de 8 bits. Dicho comportamiento se logra mediante el uso un operando flexible. Pertenece a las instrucciones de formato **F**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VCSHFL Vd, Va, Vb (2) VCSHFL Vd, Va, Imm	FOR i := 0 a 7 j := i * 8 (1) Vd [7+j:j] = Va [7+j:j] <<< Vb[7+j : j] (2) Vd [7+j:j] = Va [7+j:j] <<< Imm[7:0] ENDFOR

❖ Almacenamiento

Vector Store: operación de almacenamiento de vector en memoria, su mnemónico es **VSTR.** Esta instrucción trabaja entre un registro vectorial de 8 bytes (64 bits) y un registro escalar de 4 bytes(32 bits), adicionalmente se puede utilizar un inmediato de 20 bits extendido con cero. Del registro escalar se obtiene la dirección de memoria donde se almacenará el vector, si se especifica se le suma el valor inmediato al contenido del registro escalar. Pertenece a las instrucciones de formato **M**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VSTR Vd, Rs (2) VSTR Vd, Rs, Imm post-indexado (3) VSTR Vd, Imm, Rs pre-indexado	addr := Rs[31:0] ImmZE := ZeroExt(Imm[19:0]) (2) IF pre-index => addr := addr + ImmZE Mem[addr + 63 : addr] := Vd[63 : 0] (3) IF post-index => addr := addr + ImmZE

❖ Carga

Vector Load: operación de carga de vector en memoria, su mnemónico es **VLDR.** Esta instrucción trabaja entre un registro vectorial de 8 bytes (64 bits) y un registro escalar de 4 bytes(32 bits), adicionalmente se puede utilizar un valor inmediato de 20 bits. Del registro escalar se obtiene la dirección de memoria donde se cargará el vector, si se especifica se le suma el valor inmediato al contenido del registro escalar. Pertenece a las instrucciones de formato **M**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(4) VLDR Vd, Rs (5) VLDR Vd, Rs, Imm post-indexado (6) VLDR Vd, Imm, Rs pre-indexado	addr := Rs[31:0] ImmZE := ZeroExt(Imm[19:0]) (2) IF pre-index => addr := addr + ImmZE Vd[63 : 0] := Mem[addr + 63 : addr] (3) IF post-index => addr := addr + ImmZE

Vector Set: operación de copia de un valor a un registro vectorial, su mnemónico es **VSET.** Esta instrucción trabaja con dos registros vectoriales (64 bits) o un registro vectorial y un valor inmediato de 8 bits. Pertenece a las instrucciones de formato **F**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) VSET Vd, Vb (2) VSET Vd, Imm	(1) Vd [63 : 0] := Vb[63 : 0] (2) FOR i := 0 a 7 j := i * 8 Vd [7+j **:** j] = Imm [7 : 0] ENDFOR

Instrucciones escalares

❖ Aritméticas

Register Add: operación suma de registros escalares, su mnemónico es ADD. Esta instrucción trabaja entre dos registros escalares de 4 bytes (32 bits) o un registro escalar y un valor inmediato de 21 bits con extensión de cero. Pertenece a las instrucciones de formato **I**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) ADD Rd, Ra, Rb (2) ADD Rd, Ra, Imm	(1) Rd[31:0] := Ra[31:0] + Rb[31:0] (2) ImmZE[31:0] = ZeroExt(Imm[20:0]) Rd[31:0] := Ra[31:0] + ImmZE[31:0]

❖ Almacenamiento

Register Store: operación de almacenamiento de registro en memoria, su mnemónico es **STR.** Esta instrucción trabaja con dos registros escalares de 4 bytes(32 bits) o un registro escalar y un valor inmediato de 21 bits con extensión de cero. De un registro se obtiene la dirección de memoria donde se almacenará el dato contenido en el otro registro o valor inmediato. Pertenece a las instrucciones de formato **I**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) STR Rd, Rb (2) STR Rd, Imm	addr := Rd[31:0] (1) Mem[addr+31:addr] := Rb[31:0] (2) ImmZE[31:0] := ZeroExt(Imm[20:0]) Mem[addr+31:addr] := ImmZE[31:0]

- Carga

Register Load: operación de almacenamiento de registro en memoria, su mnemónico es **LDR.** Esta instrucción trabaja con dos registros escalares de 4 bytes(32 bits) o un registro escalar y un valor inmediato de 21 bits con extensión de cero. De un registro o el valor inmediato se obtiene la dirección de memoria desde donde se cargará el dato que será guardado en el otro registro. Pertenece a las instrucciones de formato **I**. Su uso y operación se describe a continuación

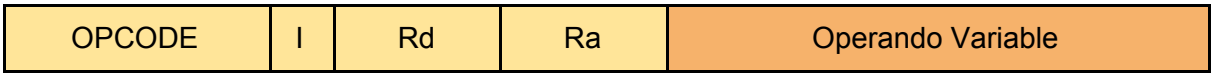
Sintaxis	Operación
(1) LDR Rd, Rb (2) LDR Rd, Imm	(1) addr := Rb[31:0] (2) addr := ZeroExt(Imm[20:0]) Rd[31:0] := Mem[addr+31:addr]

Register Set: operación de copia de un valor en un registro escalar, su mnemónico es **SET.** Esta instrucción trabaja con dos registros escalares de 4 bytes (32 bits) o un registro escalar y un valor inmediato de 21 bits con extensión de cero. Pertenece a las instrucciones de formato **I**. Su uso y operación se describe a continuación:

Sintaxis	Operación
(1) SET Rd, Rb (2) SET Rd, Imm	(1) Rd[31:0] := Rb[31:0] (2) Rd[31:0] := ZeroExt(Imm[20:0])

Codificación de instrucciones

Formato I



31 28 27 26 24 23 21 20
0

Operando variable

- Caso de uso de registro

20
0 3 2

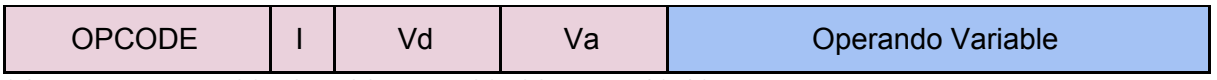
- Caso uso de valor inmediato

20
0

Formato M

31 28 27 26 23 22 3 2
0

Formato F



31 28 27 26 23 22 19 18
0

Operando variable

- Caso de uso de registro

18
0 4 3

- Caso uso de valor inmediato

El bit I

Para el formato F y el formato I se utiliza un bit para indicar si en la instrucción se utiliza o no un valor inmediato y se conoce como el bit I. El significado de sus posibles valores es:

- **I = 0:** no se utiliza un valor inmediato en la instrucción.
- **I = 1:** si se utiliza un valor inmediato en la instrucción.

El bit P

Para el formato M se utiliza un bit para indicar el modo de direccionamiento utilizado en la instrucción, se denomina a este bit como P. El significado de sus posibles valores es:

- **P = 0:** se utiliza el modo de direccionamiento pre-indexado.
- **P = 1:** se utiliza el modo de direccionamiento post-indexado.

Códigos de operación

Instrucción	Código (OPCODE)
VADD	**0000**
VSUB	**0001**
VXOR	**0010**
VAND	**0011**
VOR	**0100**
VSHFR	**0101**
VSHFL	**0110**
VCSHFR	**0111**
VCSHFL	**1000**
VSET	**1001**
VLDR	**1010**
VSTR	**1011**

ADD	**1100**
SET	**1101**
STR	**1110**
LDR	**1111**

Modos de direccionamiento

Instrucciones vectoriales

El procesador vectorial soporta el direccionamiento indexado para las instrucciones vectoriales con dos variedades:

- **Pre-indexado:** la dirección de memoria contenida en el registro escalar se le suma un valor inmediato de 20 bits con extensión de cero antes de ser utilizada para obtener o guardar el dato en memoria. Su sintaxis en ensamblador se define como:
  - VLDR Vd, Imm, Rs
  - VSTR Vd, Imm, Rs
- **Post-indexado:** la dirección de memoria contenida en el registro escalar es utilizada para obtener o guardar el dato en memoria y posteriormente se le suma un valor inmediato de 20 bits con extensión de cero. Su sintaxis en ensamblador se define como:
  - VSTR Vd, Rs, Imm
  - VSTR Vd, Rs, Imm

Instrucciones escalares

El procesador vectorial soporta el direccionamiento por registro y directo o absoluto para las instrucciones escalares donde la dirección se obtiene de un registro escalar de 32 bits o un valor inmediato de 21 bits con extensión de cero. Al utilizar el inmediato se limita el rango de direccionamiento a 2 MB, mientras que al utilizar los registros escalares se tiene acceso a todo el rango de direccionamiento. Su sintaxis en ensamblador se define como:

- LDR Rd, Rs
- LDR Rd, Imm
- STR Rd, Rs
- STR Rd, Imm

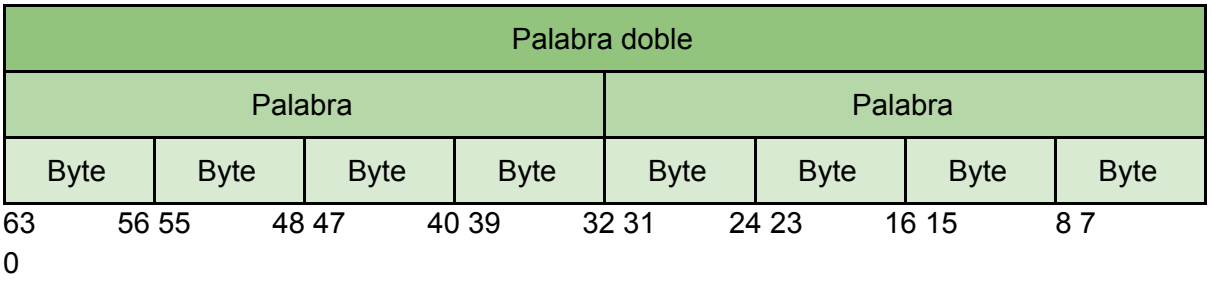
Tipos de datos

El procesador vectorial da soporte a dos tipos de datos:

- Palabra (word) de 32 bits o 4 bytes
- Palabra doble (double word) de 64 bits o 8 bytes

Se debe tener en cuenta el alineamiento:

- Para direccionar palabras se debe usar direcciones alineadas a 4 bytes.
- Para direccionar palabras dobles se debe usar direcciones alineadas a 8 bytes.



Registros

El procesador vectorial tiene un total de 24 registros de propósito general:

- 16 registros vectoriales de 64 bits
- 8 registros escalares de 32 bits

El registro escalar R7 se utiliza como el contador del programa o PC, por lo tanto se encuentra reservado.

R0	V0	V8
R1	V1	V9
R2	V2	V10
R3	V3	V11
R4	V4	V12

R5	V5	V13
R6	V6	V14
R7 (PC)	V7	V15

31 0 63 0 63 0

Formato de memoria

El formato utilizado por el procesador para almacenar y leer la información es **Big-endian**. En este formato el procesador almacena el byte más significativo de una palabra en el byte con menor número en memoria y el byte menos significativo de la palabra en el byte con mayor número en memoria. De esta manera el byte en la dirección 0 en memoria se conecta al bus de datos en las líneas 31 a la 24 y así sucesivamente.

Considerando una dirección de memoria A se tiene que:

Palabra doble en dirección A							
Palabra en dirección A				Palabra en dirección A+4			
Byte en dirección A	Byte en dirección A+1	Byte en dirección A+2	Byte en dirección A+3	Byte en dirección A+4	Byte en dirección A+5	Byte en dirección A+6	Byte en dirección A+7

63 56 55 48 47 40 39 32 31 24 23 16 15 8 7 0

Interfaz con memoria

El procesador vectorial posee una arquitectura de Harvard, por lo que se hace la separación física entre la memoria para instrucciones y la memoria para datos. Cada memoria posee su propio bus de 32 bits para garantizar el acceso simultáneo.

El pipeline de instrucciones

El procesador vectorial utiliza un pipeline para incrementar la velocidad del flujo de instrucciones. Esto permite ejecutar operaciones de manera simultánea y que el procesamiento y los sistemas de memoria operen de manera continua. Un pipeline de 5 etapas es utilizado, por lo que las instrucciones pasan por las siguientes etapas:

- **Fetch:** en esta etapa se obtiene la siguiente instrucción a ejecutar.

- **Decode:** en esta etapa se decodifica la instrucción, se realiza la lectura de los bancos de registros y se activan las señales correspondientes mediante la unidad de control.
- **Execute:** en esta etapa se realizan las operaciones y cálculos respectivos.
- **Memory:** en esta etapa se realiza la lectura o escritura en memoria si aplica.
- **Writeback:** en esta etapa se realiza la escritura en los bancos de registros si aplica.