Reference Sheet

Version 1

Fabián Montero Villalobos[1], José Julián Camacho Hernández[1],
José Alejandro Soto Chacón[1]
[1]Tecnológico de Costa Rica
fmonterov@estudiantec.cr, jcamacho341@estudiantec.cr, soto@estudiantec.cr
October 2023

Índice general

Capítulo 1

Introducción

1.1. Descripción general

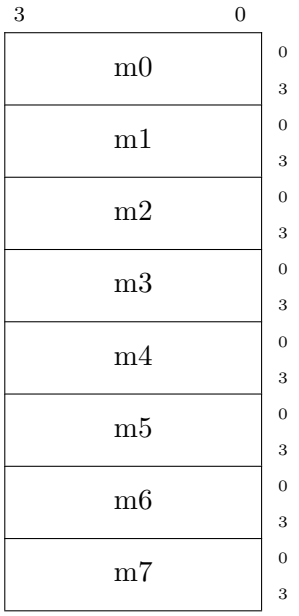
1.1.1. Registros vectoriales

Esta arquitectura cuenta con 8 registros que se pueden interpretar de dos maneras:

- Registros de tamaño 256bits, que se interpretan como matrices de 4×4
- Registros de tamaño 4×64bits, que se interpretan como cuatro vectores de tamaño 4.

Cada elemento de una matriz/vector se conoce como una *word*. Cada *word* es un número de punto flotante de 16 bits.

No existen tipos escalares o enteros.



1.1.2. Tipos de dato

1.1.2.1. Matrices

Las matrices corresponden a 1 de 8 posibles registros vectoriales. Cada uno de estos registros contiene una matriz de 4×4 elementos. Cada elemento es un número de punto flotante de 16 bits.

1.1.2.2. Vectores

Los vectores corresponden a 1 de 8 posibles registros vectoriales. Cada uno de estos registros contiene cuatro vectores de 4 elementos cada uno. Cada elemento es un número de punto flotante de 16 bits.

Las operaciones realizadas a vectores se realizan a los 4 vectores que están en el registro referenciado.

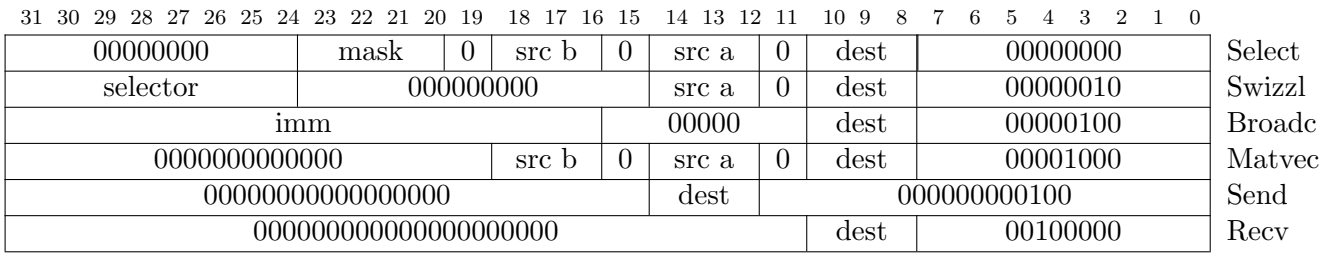
1.2. Modos de direccionamiento

Se usan registros para guardar matrices y vectores. Cada registro contiene ya sea una matriz o cuatro vectores. En estos casos, el modo de direccionamiento es de registro.

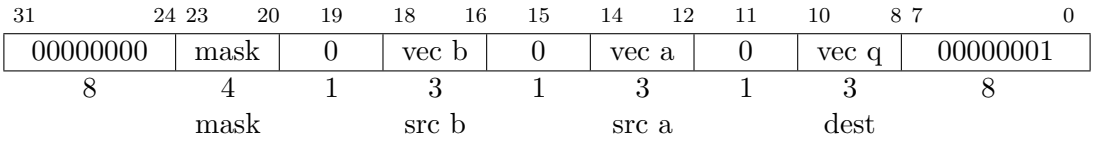
Además, también existe el modo de direccionamiento inmediato. Este modo se utiliza en instrucciones como `broadc`, en el cual se recibe un numero directamente para ser escrito en todos los vectores.

Capítulo 2

Set de instrucciones vectoriales



2.1. Select



Genera un vector de salida a partir de dos vectores. El vector de salida se construye evaluando los bits de la máscara **mask** (hay 1 por cada posición). Si el bit i-ésimo de la máscara es 1, el bit i-ésimo de la salida **vec q** viene del bit i-ésimo de **vec b**, de lo contrario, viene de **vec a**.

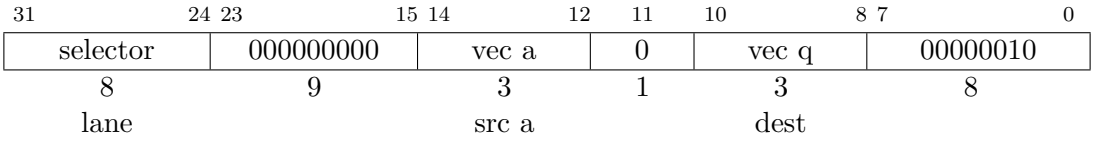
Uso:

```
select <vec q> <vec a> <vec b> <mask>
```

Ejemplo:

```
select m1, m1, m2, aabb
```

2.2. Swizzl



Reordena los elementos de un vector. El i-ésimo elemento del vector de salida **vec q** corresponde a cierto elemento del vector de entrada **vec a**. Los valores del **selector** definen el orden del vector de salida asumiendo que el orden del vector de entrada es `xyzw`.

La operación se aplica a todos los vectores del registro referenciado.

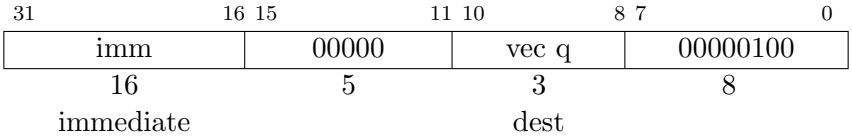
Uso:

```
select <vec q> <vec a> <selector>
```

Ejemplo:

```
swizzle m0, m1, wzyx
```

2.3. Broadc



Reemplaza el valor de todos los elementos del registro **vec q** por el valor determinado por el inmediato **imm**.

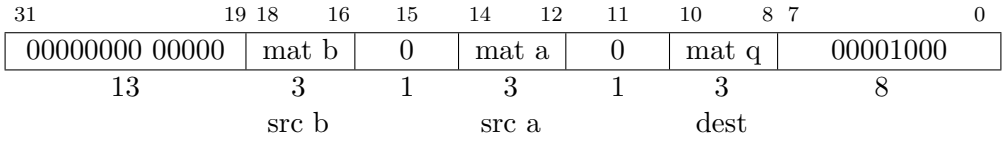
Uso:

```
select <vec q> <imm>
```

Ejemplo:

```
broadc m2, 1.0
```

2.4. Matvec



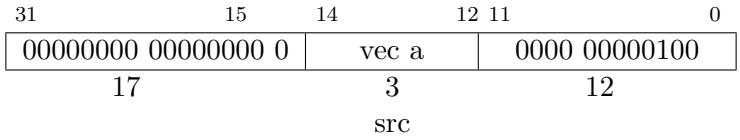
Realiza multiplicación matriciales. Recibe dos índices de correspondientes a registros vectoriales en los cuales se encuentran dos matrices que son los operandos.

Además, recibe un tercer índice en el cual guardar el resultado de la operación.

Uso:

`matvec <mat q> <mat b> <mat a>`

2.5. Send



Toma el valor del registro **vec a** y lo envía hacia el *pipeline* de rasterizado.

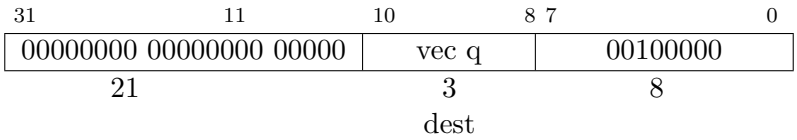
Uso:

`send <vec a>`

Ejemplo:

`send m1`

2.6. Recv



Recibe información desde la memoria *batch* y la almacena en el registro de destino **vec q**.

Uso:

```
recv <vec q>
```

Ejemplo:

```
recv m1
```