Problema

En un procesador vectorial con las siguientes características:

- Registros con una longitud vectorial máxima de 64 elementos.
- Una unidad de suma vectorial con tiempo de arranque de 6 ciclos.
- Una unidad de multiplicación con tiempo de arranque de 7 ciclos.
- Una unidad de carga/almacenamiento con tiempo de arranque de 12 ciclos.
- Tbase de 0 ciclos y Tbucle de 15 ciclos.

se pretende ejecutar el siguiente bucle:

```
for (i=0; i<64; i++)
     for (j=0; j<64; j++)
          Y[j]:= a*X[i,j]+Y[j];
     end for;
end for;
```

a) Escriba la secuencia de código correcta para el bucle interior en instrucciones vectoriales. Considere que la dirección en memoria del vector `X[i,-]` se encuentra en `Rx`, la del vector `Y` en `Ry` y el valor del escalar `a` se encuentra en `Ra`.

b) Estime el rendimiento del bucle completo en un procesador vectorial calculando T_{64*64} en ciclos de reloj considerando la posibilidad de encadenamiento entre las unidades funcionales y que solo se emite un nuevo convoy cuando las instrucciones del convoy previo han concluido. Suponga que hay un gasto de T_{bucle} tanto en cada iteración del bucle externo como del interno.

c) ¿Qué limita el rendimiento del bucle completo?

d) Reescriba el código vectorial para reducir la limitación del rendimiento; muestre el bucle interior resultante en instrucciones vectoriales.

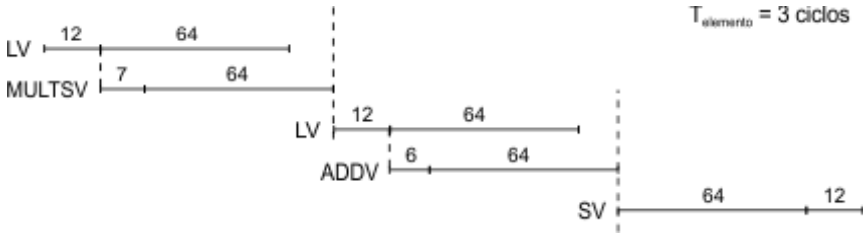
Solución

a) El código vectorial para el bucle interior del enunciado es el siguiente:

```
LV        V1,Rx              // Carga del vector X[i,-]
MULTSV    V2,V1,Fa           // aux:=a*X[i,-];
LV        V3,Ry              // Carga del vector Y
ADDV      V4,V2,V3           // Y:=a*X[i,]+Y;
SV        Ry,V4              // Almacenamiento de Y
```

b) Dado que ahora es posible encadenar los resultados de las unidades, la organización del código vectorial en convoyes quedaría de la siguiente forma:

```
Convoy 1:  LV        V1,Rx
           MULTSV    V2,V1,Fa
Convoy 2:  LV        V3,Ry
           ADDV      V4,V2,V3
Convoy 3:  SV        Ry,V4
```



El T*elemento* es de 3 ciclos dado que se tienen tres convoyes. El T*arranque* total se obtiene de sumar los tiempos de arranque visibles de las unidades funcionales. Si se analiza el esquema previo se tiene

T*arranque* = T*arranque* LV + T*arranque* MULTV + T*arranque* LV + T*arranque* ADDV + T*arranque* SV

T*arranque* = (12 + 7 + 12 + 6 + 12) ciclos = 49 ciclos

Con estos valores y dado que el tiempo base es 0, la expresión del tiempo total de ejecución del bucle interior queda

$$T_{64} = \left\lceil \frac{64}{64} \right\rceil \cdot (15 + 49) + 3 \cdot 64 = 256 \text{ ciclos}$$

Este resultado representa el tiempo en ciclos para ejecutar el bucle interno. El tiempo total puede calcularse a partir del dato anterior

$$T_{64*64} = i \cdot (T_{bucle} + T_{64}) = 64 \cdot (15 + 256) = 17344 \text{ ciclos}$$

donde *i* es el número de iteraciones del bucle exterior multiplicado por el tiempo de ejecución del bucle interior al que hay que añadir los sobrecostes.

c) La limitación radica en que en cada para iteración del bucle interior es necesario cargar el vector `Y` en un registro vectorial, y posteriormente, almacenar el resultado de las operaciones de nuevo en `Y`.

Para mejorar el rendimiento se podría cargar y almacenar `Y` antes y después del bucle interior de forma que el bucle interior quede reducido únicamente a 3 instrucciones vectoriales (la carga de `X[i,-]`, la multiplicación por el escalar y la posterior suma de `Y`).

d) Poniendo en práctica la mejora comentada en el apartado anterior tendremos el siguiente código:

```
          LV        V3,Ry          // Carga de Y
bucle_i:
bucle_j:  LV        V1,Rx          // Carga de X[i,-]
          MULTSV    V2,V1,Fa       // Y:=a*X[i,-]
          ADDV      V4,V2,V3       // Y:=a*X[i,-]+Y

          // aquí se coloca el código escalar para
          // gestionar el bucle externo. Básicamente
          // incrementar i y saltar a bucle_i si no se ha
          // completado

          SV        Ry,V4          // Almacenamiento de Y
```