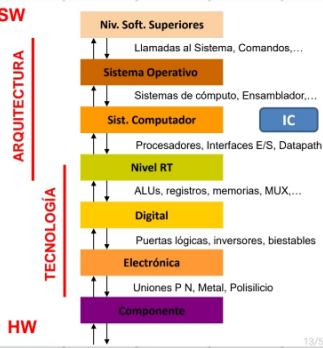


Unidad 1 IC

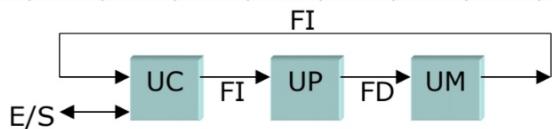
* Niveles de Abstracción de un computador:



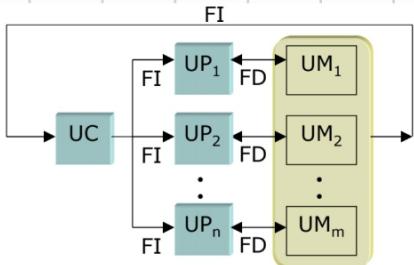
* Taxonomía de Flynn:

"PU" = Unidad de procesamiento

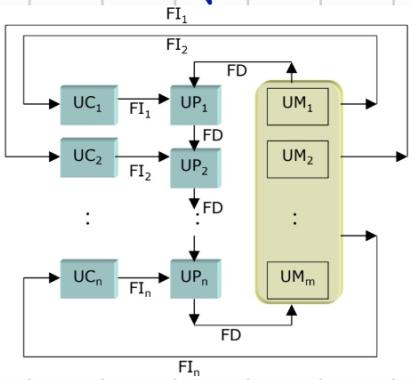
- SISD : Simple Instruction Simple Data .



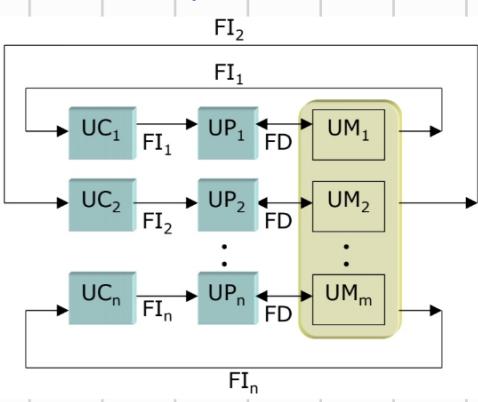
- SIMD : Simple Instruction Multiple Data .



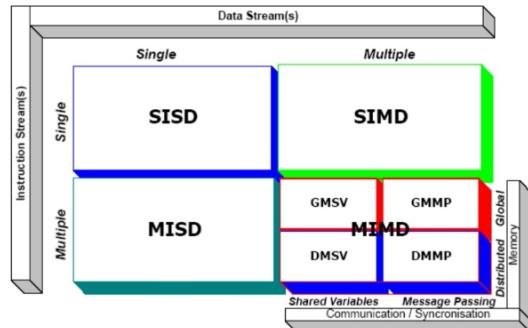
- MISD : Multiple Instruction Simple Data .



- MIMD : Multiple Instruction Multiple Data .



• Taxonomía de Flynn - Johnson



• Tipos de paralelismo:

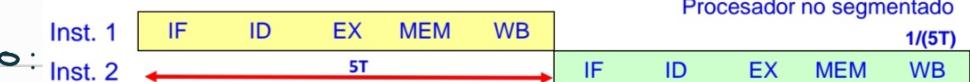
- **Paralelismo de datos**: La misma función, instrucción, etc., se ejecuta en paralelo pero en cada una de esas ejecuciones se aplica sobre un conjunto de datos distinto.

• **Paralelismo funcional**: Varias funciones, tareas, instrucciones, etc. (iguales o distintas) se ejecutan en paralelo.

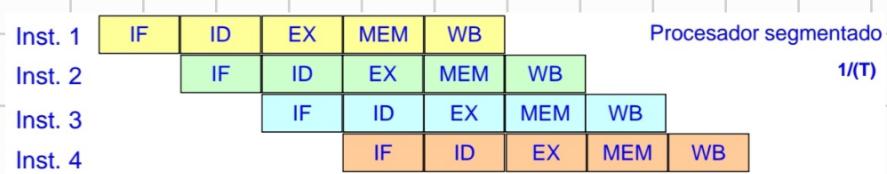
- **Nivel Instrucción**: Se ejecutan en paralelo las instrucciones de un programa.
- **Nivel de bucle o hebra**: se ejecutan en paralelo distintas iteraciones de un bucle o secuencias de instrucciones de un programa.
- **Nivel de procedimiento**: distintos procedimientos que constituyen un programa se ejecutan simultáneamente.
- **Nivel de programa**: la plataforma ejecuta en paralelo programas diferentes que pueden corresponder, o no, a una misma aplicación.

• Segmentación ILP:

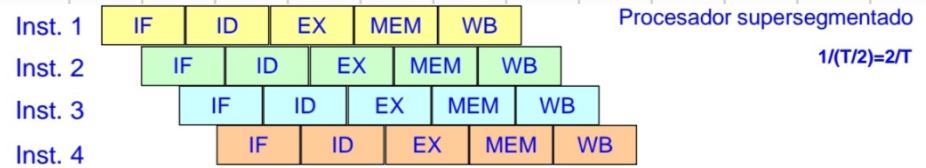
* **Procesador no segmentado**:



* **Procesador segmentado**:



* **Procesador supersegmentado**:



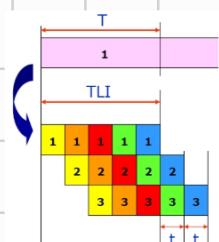
* **Ganancia**: suponemos que TLI (tiempo de latencia de inicio) = T, tiempo que tarda en ejecutarse una operación en una unidad sin segmentar.

$$TLI = K \cdot t$$

$K = \text{nº etapas del cauce}$, $t = \text{duración de cada etapa}$

$$GK = \frac{T_1}{T_K} = \frac{K \cdot n \cdot t}{K \cdot t + (n-1) \cdot t} = \frac{K \cdot n}{K + n - 1}$$

$$\lim_{n \rightarrow \infty} GK = K$$



* Ganancia Real: $G_k = \frac{T_{\text{sin-segmentar}}}{T_{\text{segmentado}}} = \frac{n \cdot T}{TLI + (n-1) \cdot t}$

$$G_{\max} = \lim_{n \rightarrow \infty} \frac{n \cdot t}{(k \cdot t) + (n-1) \cdot t} = \frac{T}{t} < k$$

$$\downarrow$$

$$TLI = k \cdot t$$

$$\downarrow$$

$$TLI = kt$$

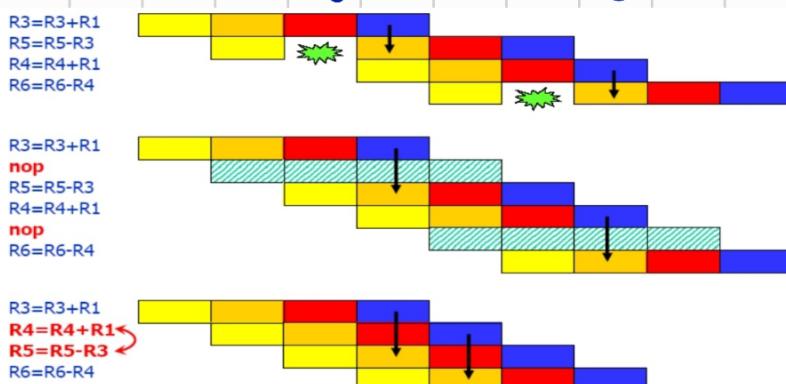
Tipos de riesgos (detección del cauce):

- Riesgos de datos: Dependencias entre operandos y resultados de instrucciones distintas
- Riesgos de control: Se originan a partir de instrucciones de salto condicional que, según su resultado, determinan la secuencia de instrucciones que hay que procesar tras ellos.
- Riesgos estructurales o colisiones: Se producen cuando instrucciones diferentes necesitan el mismo recurso al mismo tiempo

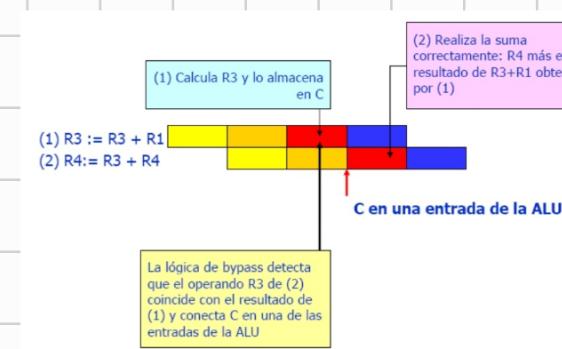
Riesgos de almacenamiento:



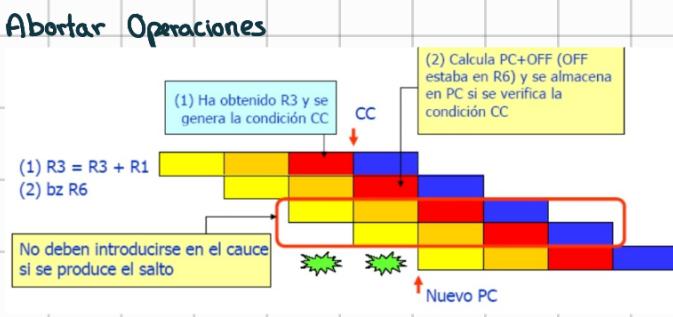
Soluciones de datos Reorganización de código:



Forwardings:



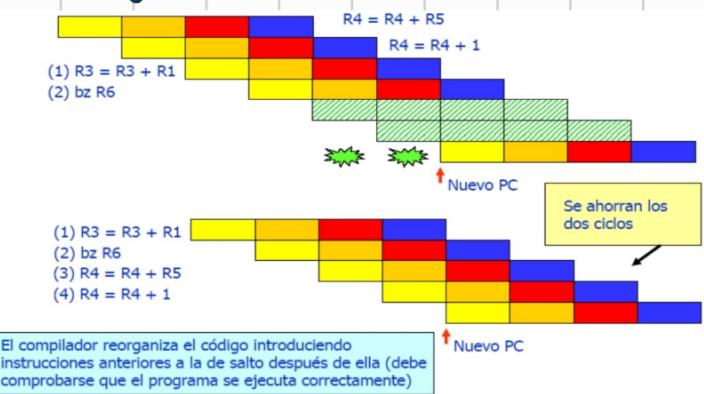
Abortar Operaciones



• Bloqueo o uso de NOP



• Delayed Branch



• Tiempo de ejecución de un programa

• Tiempo de CPU (usuario y sistema)

• Tiempo de E/S.

$$\text{Tiempo de CPU (T}_{\text{cpu}}) = \frac{\text{Ciclos_del_programa}}{\text{Frecuencia del Reloj}}$$

$$\text{Tciclo} = \frac{\text{Ciclos_del_Programa}}{\text{Frecuencia del Reloj}}$$

$$\text{Ciclos por Instrucción (CPI)} = \frac{\text{Ciclos_del_programa}}{\text{Número de Instrucciones (NI)}}$$

$$\text{T}_{\text{cpu}} = \text{NI} \cdot \text{CPI} \cdot \text{Tciclo}$$

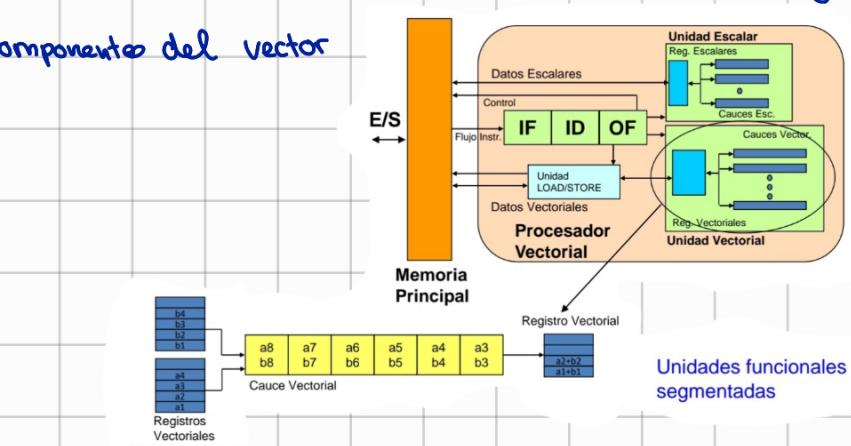
$$\text{Tciclo} = 1/F$$

$\hookrightarrow F = \text{frecuencia}$

• Arquitecturas Vectoriales: ILP y paralelismo de datos.

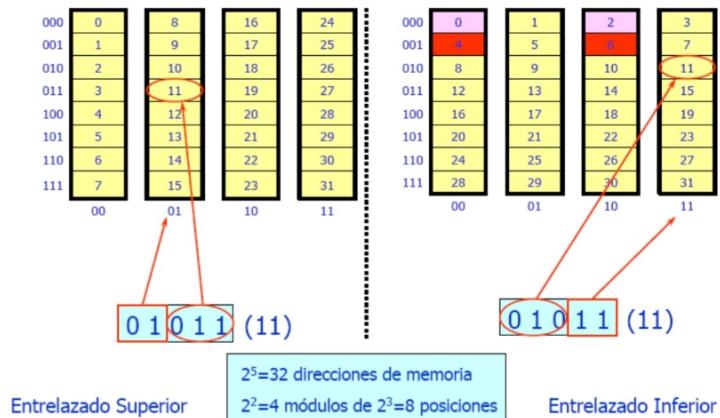
El procesamiento de Instrucciones está segmentado y se utilizan múltiples unidades funcionales

Paralelismo de datos: Cada instrucción Vectorial codifica una operación sobre todos los Componentes del vector

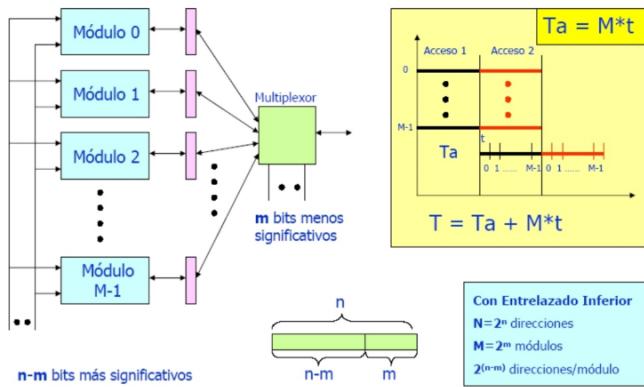


- Arquitectura Orientada al procesamiento de vectores, repertorio de instrucciones especializado.
- Características: Cálculo de los componentes del vector de forma independiente, cada operación vectorial codifica gran cantidad de cálculos, se optimiza el uso de memoria.

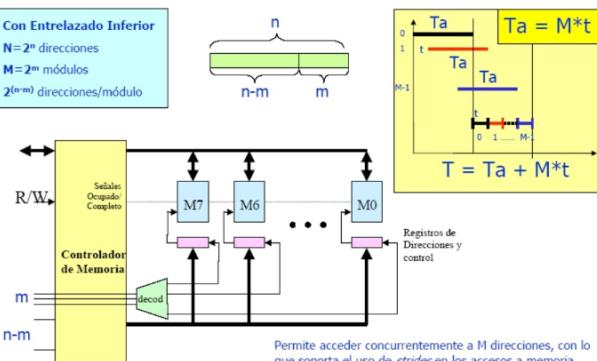
* Entrelazado de memoria:



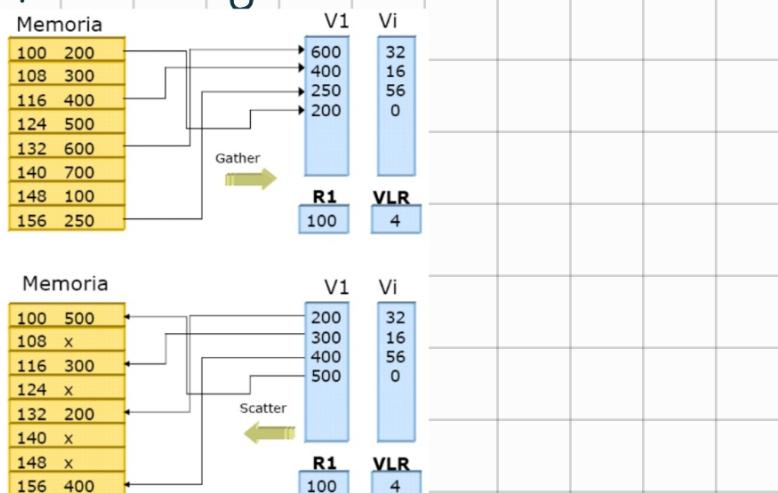
* Acceso a memoria simultánea o tipo S



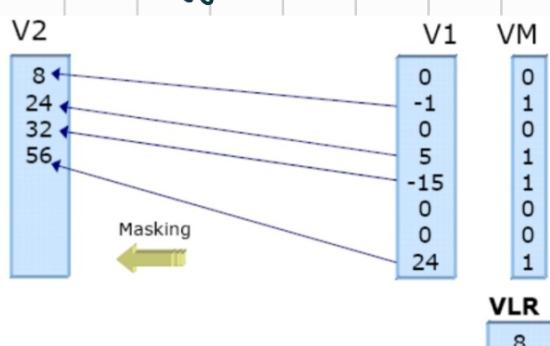
* Acceso a memoria Concurrente o Tipo C



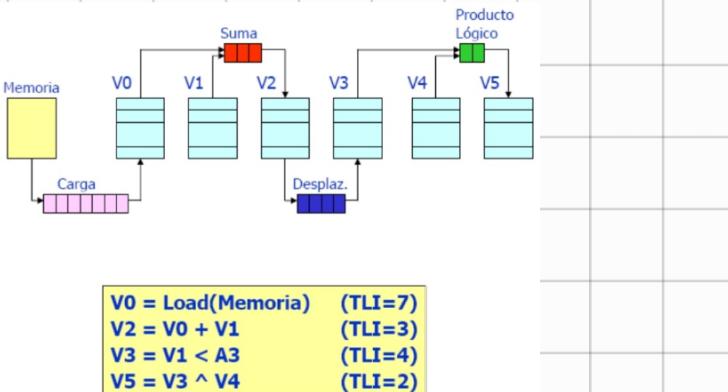
* Operaciones gather - scatter:



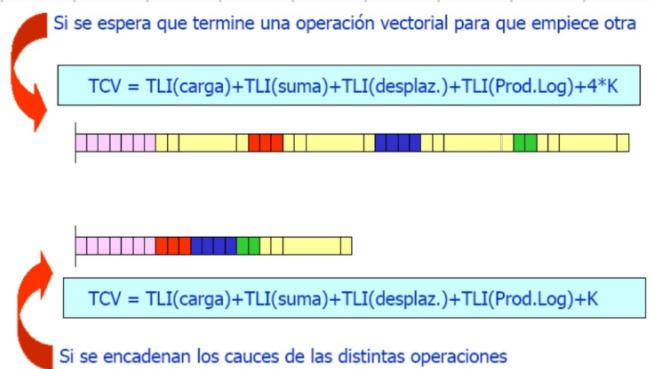
* Enmascaramiento (gestión de matrices dispersas):



*Rendimiento: encadenamiento de cauce



*Rendimiento: encadenamiento de cauce



Ejercicios

Ejercicio 1.

Se require sumar 2 vectores de 10 elementos, ordenados de forma consecutiva desde la dirección: 01C2 HEX.

$$A[10] + B[10].$$

Cuántos accesos se debe hacer? Cuándo duraría?

- Acceso de tipo S.
- 8 Bloques.
- Entrelazado INF = [POS][MOD]
- Datos empiezan en: 01C2 HEX.

Pasos para resolver el ejercicio

Paso 1: Pasar de Hex a Bin la dirección de inicio

$$0 = 0000, \quad 1 = 0001, \quad C = 1100, \quad Z = 0010$$

$$\text{Dirección de inicio} = \underline{\underline{0000\ 0001\ 1100\ 0010}}$$

Paso 2: ¿Cuántos bloques hay? tenemos 8 bloques ¿Cuántos bits necesitamos para direccionar 8 módulos? $\log_2(8) = 3$

* El entrelazado al ser inferior, significa que escogeremos los últimos 3 bits para direccionar [0:10] y el resto [000 0000 11-1000] para especificar una posición dentro de cada módulo.

Paso 3: Módulo 010 = N°z (de 0 a 2)

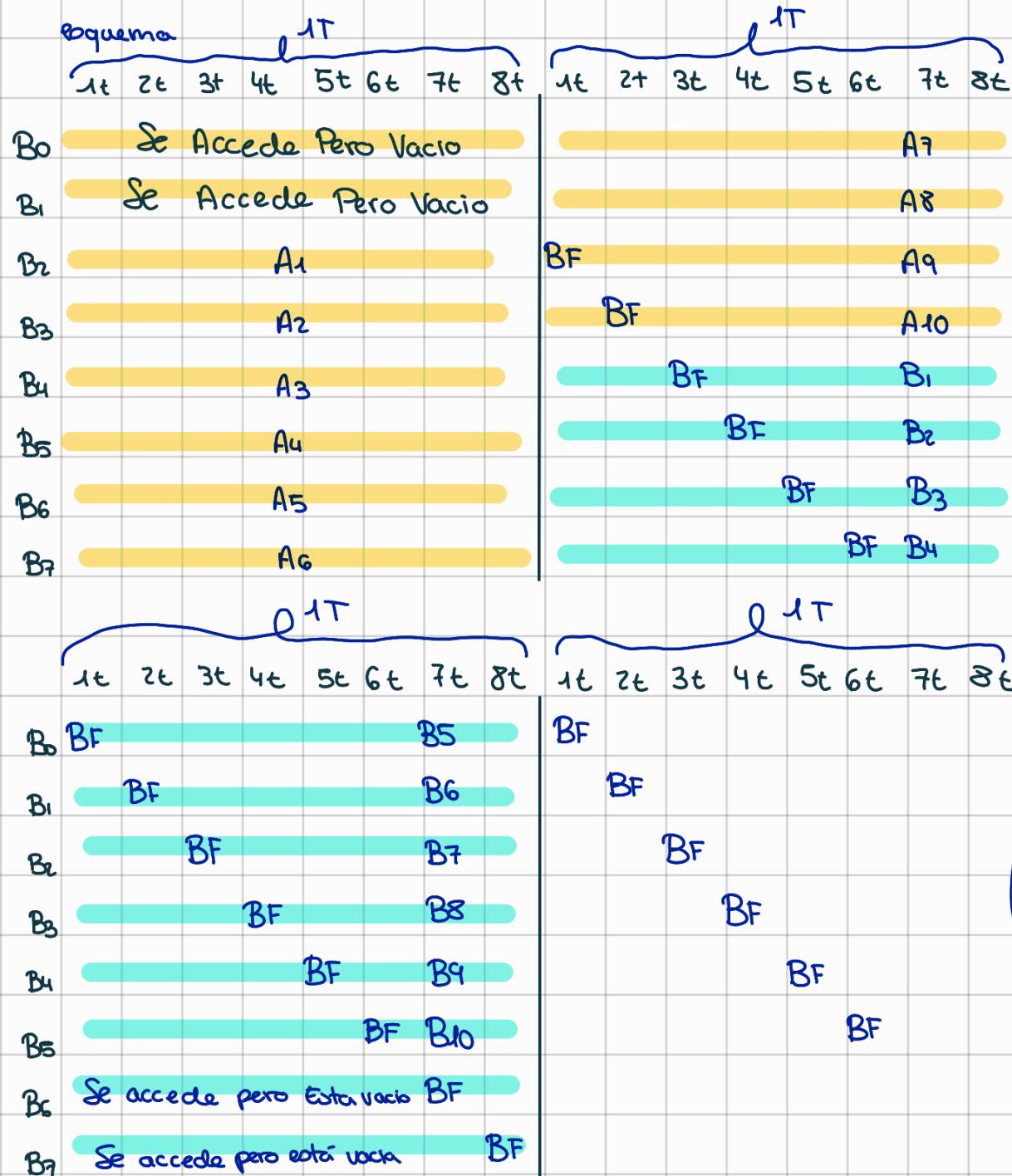
Sacamos como quedarían los bloques:

$$A[10] + B[10].$$

	B0	B1	B2	B3	B4	B5	B6	B7	→ Bloques
P1	X	X	A1	A2	A3	A4	A5	A6	
P2	A7	A8	A9	A10	B1	B2	B3	B4	
P3	B5	B6	B7	B8	B9	B10	X	X	

Tenemos 8 accesos a memoria de 8 bloques antes de utilizar Buffer

Paso 4: Accedemos de manera simultanea como nos dice el ejercicio, accedemos a todos los bloques a la vez, iterando posición a posición y hacemos un esquema



Esto es el eco de las posiciones que deberia de leer el buffer pero no lee

Creo que el Buffer accede 20 veces por la cantidad de posiciones de memoria que deberia de leer

A la memoria se accede 3 veces

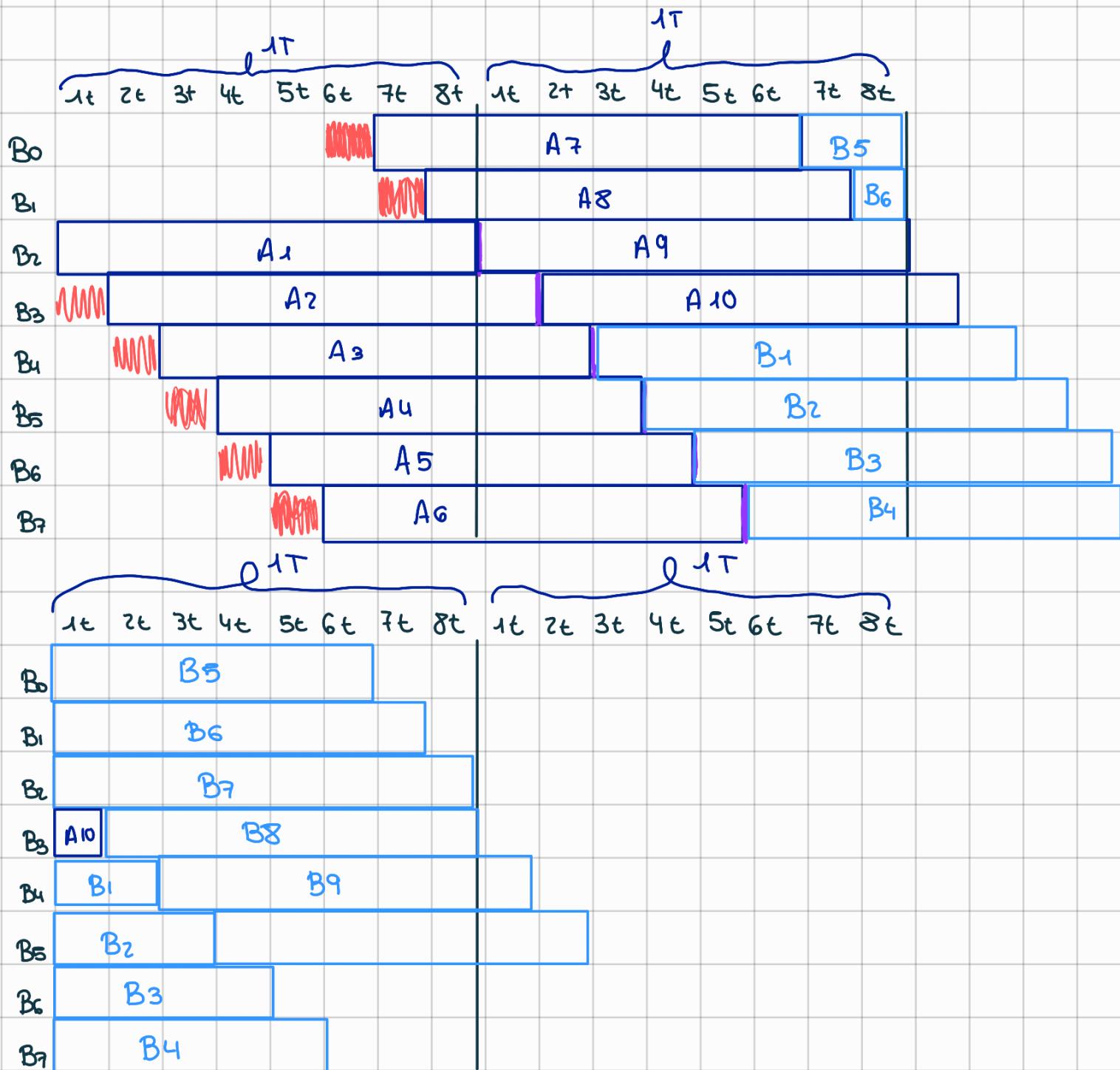
Paso 5: ¿Cuantos accesos se debe hacer? Cuanto duraria?

Se ha accedido 3 veces a memoria = 3T Lo coloreado

Duración 3T + 6t

Es el tiempo que tarda el buffer en finalizar

Tipo C de acceso concurrente



Se ha accedido 3 veces a memoria 3T y 3t Duración 3T+3t

Si los elementos de los vectores se almacenan con stride = 3 cual sería el tiempo con Tipo C?

	B ₀	B ₁	B ₂	B ₃	B ₄	B ₅	B ₆	B ₇
P ₁		A ₁		A ₂				
P ₂	A ₃		A ₄		A ₅			
P ₃		A ₆		A ₇		A ₈		
P ₄		A ₉			A ₁₀			
P ₅	B ₁		B ₂		B ₃			
P ₆		B ₄		B ₅		B ₆		
P ₇		B ₇		B ₈				
P ₈	B ₉		B ₁₀					

