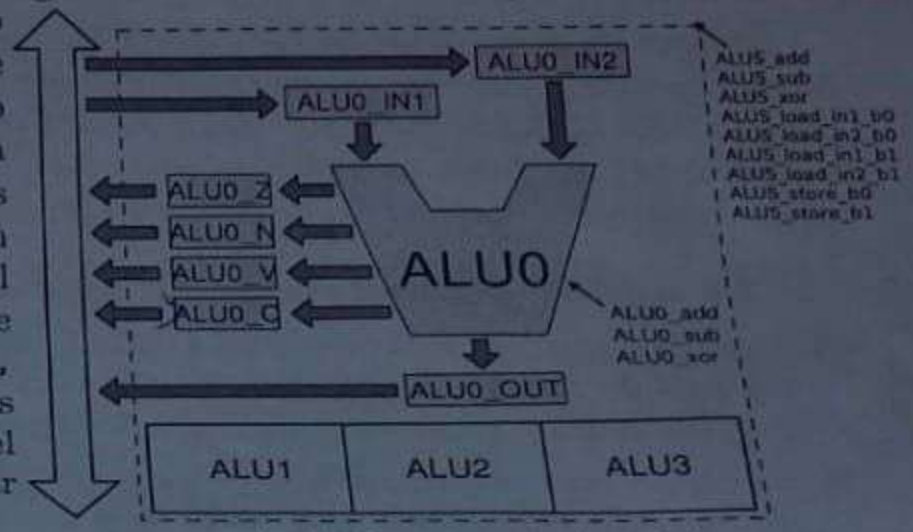
Ejercicio 1 Microprogramación

CORRECTOR: MAXI

Se extiende una arquitectura (palabras, registros y direcciones de 16 bits y direccionamiento a palabra) con operaciones SIMD[1]. Se cuenta con 4 ALUs (ALU_0 a ALU_3) que se utilizarán en paralelo sobre 4 registros destino y 4 registros origen. Se denomina a los registros `R0..R3` como el **Banco 0** (`B0`) y a los registros `R4..R7` como el **Banco 1** (`B1`). Cualquier instrucción de dos parámetros puede tomar como parámetro destino y/o origen un **banco**. En ese caso, la operación trabaja en paralelo sobre los registros de el/los banco/s la vez. Las $ALU_{1...3}$ tienen registros y señales análogas a las de ALU_0. El componente conformado por las 4 ALUs tiene señales para operar en modo SIMD: `ALUS_add`, `ALUS_sub`, `ALUS_xor`. También tiene señales para cargar todo el contenido de un banco el registro *IN1* o *IN2* de cada ALU, y para guardar la salida de todas las ALUs en un banco.



4 ALUs conectadas al bus.

El procesador resuelve las siguientes instrucciones y modos de direccionamiento:

Instrucciones	Modos de direccionamiento
De dos parámetros:	Directo a registro: `Rx`
`ADD`, `SUB`, `XOR`, `MOV`.	Inmediato: `imm16` (en complemento a dos)
De un parámetro:	Indirecto a registro: `[Rx]`
`JMP`, `JZ`, `JC`, `JV`, `JN`	Indirecto inmediato: `[imm16]`
	Directo a Banco: `Bx`

Todas las combinaciones entre modos de direccionamiento son válidas salvo (a) entre dos inmediatos o (b) cuando el origen es un Banco y el destino es un registro. La siguiente tabla da ejemplos de operaciones SIMD:

Instrucción	Efecto
`ADD B1, 0x0002`	$R_i \leftarrow R_i$ + 2 para $i = 4 \ldots 7$
`SUB B0, B1`	$R_i \leftarrow R_i$ - R_{i+4} para $i = 0 \ldots 3$
`MOV B1, B0`	$R_{i+4} \leftarrow R_i$ para $i = 0 \ldots 3$
`MOV [0xAAAA], B0`	`[0xAAAA + i]` $\leftarrow R_i$ para $i = 0 \ldots 3$.
`MOV B1, [R0]`	$R_{i+4} \leftarrow$ `[R0 + i]` para $i = 0 \ldots 3$.

1. Proponer una codificación variable de una o dos palabras para este set de instrucciones.
2. Completar el *datapath* e indicar el tamaño y conexiones de cada registro.
3. Explicar cómo se implementa el fetch de instrucciones sobre el *datapath* propuesto.
4. Codificar las instrucciones de ejemplo indicadas arriba e indicar su secuencia de microinstrucciones.

Ejercicio 2 Memoria Caché

Sea una máquina ORGA1 (16 bits, direccionable a palabra, 128 KB de memoria principal) con una memoria caché de mapeo directo de 128 bytes (sin contar el almacenamiento para los tags) **solo para datos**, y líneas de 8 bytes.

[1] *Single Instruction, Multiple Data*, Es decir, instrucciones que se aplican sobre un conjunto de múltiples datos.

1. Realizar un seguimiento de la caché al invocar a la función `func`. Calcular el *hit-rate* correspondiente y describir el estado completo de la caché luego de cada acceso. Indicar si hubo accesos desalineados.

2. Sabiendo que la función anterior va a ser invocada sucesivas veces de manera consecutiva, y sin modificar el tamaño de la caché, proponer una configuración (indicando tipo, política de reemplazo si corresponde y tamaño de línea) que maximice el *hit-rate* obtenido. Indicar cómo queda el estado de la caché bajo esta nueva configuración. Justificar.

```
func:   MOV R1, 0x1003   ; base
        MOV R2, 0x0800   ; incr
        MOV R3, 0        ; res
inicio: MOV R0, 8        ; iter
ciclo:  CMP R0, 0
        JE fin
        ADD R3, [R1]
        XOR R3, [R1+0x1000]
        ADD R1, R2
        ADD R2, R2   ; R2=R2<<1
        SUB R0, 1
        JMP ciclo
fin:    RET
```

Ejercicio 3 Entrada/Salida

Se tiene un procesador ORGA1 al que se le agregó un cooler para evitar que sobrecaliente cuando está en funcionamiento. Dicho cooler hace mucho ruido, por lo que tiene un control de velocidad (registro `VEL_COOLER`, lectoescritura) que le permite funcionar más silenciosamente cuando el procesador está frío. Además, tiene un sensor de temperatura accesible a través del registro `TEMP`, de solo lectura, y un dispositivo de apagado (registro `APAGADO`, de solo escritura).

1. Asignar valores para cada registro dentro de las direcciones de E/S de ORGA1.
2. Implementar en assembler de ORGA1 una rutina que utilice *polling* para sensar la temperatura de la CPU, y que cambie la velocidad del cooler siguiendo estas relaciones entre la temperatura t y la velocidad v: $(t < 50) \Rightarrow v = 1; (50 \leq t < 70) \Rightarrow v = 2; (t \geq 70) \Rightarrow v = 3$
3. Se reemplaza el procesador por un ORGA1i, modificando el comportamiento del sensor de temperatura para que por cada variación genere una interrupción[2], indicada como un valor en complemento a 2 de 16 bits. Por ejemplo, si la temperatura cayó 3 grados, el registro `TEMP` ahora indicará `0xFFFD`. La calibración de la medición de temperatura se hace automáticamente cuando la computadora se enciende; el sensor escribe por DMA el valor absoluto correspondiente en la dirección `0x1000`. Modificar la rutina anterior para que actualice apropiadamente la velocidad del cooler utilizando esta nueva funcionalidad[3].
4. Los coolers utilizados ocasionalmente se traban y dejan de girar. Para evitar que la CPU se queme, se decide implementar un mecanismo en el cooler para que notifique por medio de una interrupción esta situación, de manera que la CPU pueda apagarse automáticamente.

   a) Diseñar un mecanismo para que ORGA1i soporte interrupciones de ambos dispositivos, dándole mayor prioridad a la del cooler. Identificar componentes, conexiones y registros nuevos necesarios (según corresponda) para dicho mecanismo funcione, indicando cómo se lo configuraría desde la CPU.

   b) Escribir la rutina de atención para el cooler de modo que, utilizando el mecanismo anterior, apague la máquina escribiendo el valor `0xFFFF` en el registro `APAGADO`.

Ejercicio 4 Buses

Diseñar un bus sincrónico para un sistema con 3 dispositivos $D_0 \dots D_2$ con acceso compartido al bus. En cualquier momento un dispositivo puede pedir acceso al bus. El protocolo debe cumplir que (a) un dispositivo debe liberar el bus luego de usarlo por 4 ciclos consecutivos si hay pedidos pendientes, esperando 1 ciclo para volver a pedir acceso, y (b) si D_i libera el bus y hay otros dos esperando para usarlo, el que obtiene el acceso es $D_{(i+1 \mod 3)}$ (cola circular 0-1-2-0-1-2...). Luego, dar y explicar los siguientes diagramas de tiempos:

1. D_0 comienza una transacción que toma 6 ciclos; en el ciclo 3 D_1 pide usar el bus por 4 ciclos.
2. D_1 comienza una transacción de 10 ciclos. En el ciclo 2, D_0 y D_2 piden al mismo tiempo acceso al bus, para hacer transacciones de 4 y 6 ciclos de duración respectivamente.
3. Si en este bus un dispositivo necesita ocupar el bus por 15 ciclos, ¿cuál es el máximo de ciclos que puede tomarle hacer la transacción completa? Justifique detalladamente.

[2] Suponer que el manejo de interrupciones no hace que se pierdan eventos.
[3] La velocidad del cooler puede no corresponderse con la esperada al momento de prender la CPU.