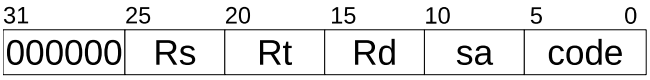
1 **(1.5 puntos)** Se ha ampliado la ruta de datos estudiada en clase para dar soporte a las tres instrucciones de desplazamiento clásicas: a la izquierda («), lógico a la derecha ($»_L$) y aritmético a la derecha ($»_A$). El valor a desplazar siempre está en *Rt*. Para cada operación hay dos instrucciones: una de desplazamiento constante, donde el número de bits a desplazar (*sa*, de *shift amount*, 0≤*sa*≤31) está codificado en cinco bits en la misma instrucción, y la de desplazamiento variable, que obtiene el número de bits a desplazarde los cinco bits inferiores del registro Rs. En total, son seis instrucciones.

instrucción	operación
sll rd,rt,sa	rd = rt « sa
srl rd,rt,sa	rd = rt $»_L$ sa
sra rd,rt,sa	rd = rt $»_A$ sa

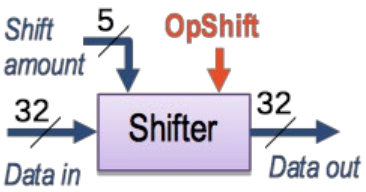
instrucción	operación
sllv rd,rt,rs	rd = rt « rs
srlv rd,rt,rs	rd = rt $»_L$ rs
srav rd,rt,rs	rd = rt $»_A$ rs

Nótese que en los desplazamientos constantes la ruta de datos ha de ignorar el contenido de *Rs*. La codificación de las instrucciones es esta:



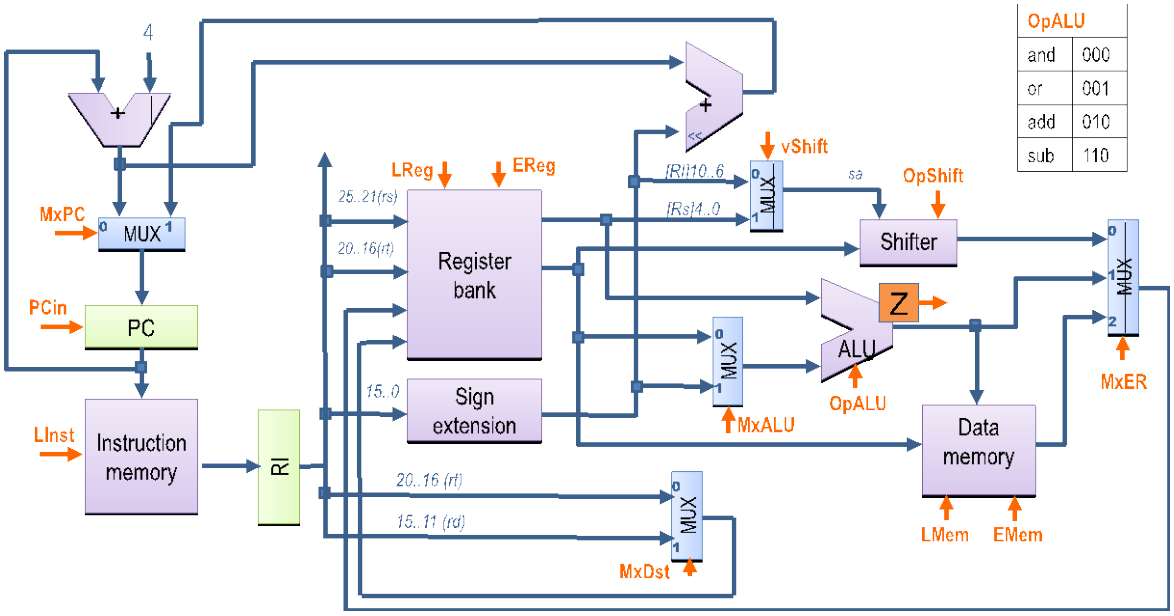
code					
0	2	3	4	6	7
sll	srl	sra	sllv	srlv	srav

Para ello, la ruta de datos tiene un componente nuevo, el *shifter*, que puede hacer tres operaciones:



OpShift	Operación
00	«, desplazamiento izquierda
10	$»_L$, desplazamiento lógico derecha
11	$»_A$, desplazamiento aritmético derecha

Un nuevo multiplexor permite seleccionar el origen del valor de *shift amount* mediante la señal de control *vShift* (0 = desplazamiento constante, 1 = desplazamiento variable). El multiplexor de escritura de registros se ha ampliado para que tenga tres entradas de datos; ahora, la señal de control *MxER* tendrá dos bits. La ruta de datos resultante es esta:



Complete la tabla de señales de control para cada instrucción:

Instrucción	EReg	vShift	OpShift	MxDst	MxALU	OpALU	LMem	EMem	MxER	MxPC
sll										
sllv										
lw										
sw										
add										
addi										
beq										

2 **(1.5 puntos)** Denominamos PROCESADOR 1 al procesador monociclo de la pregunta anterior en el que las operaciones en memoria conllevan 20 ns, leer y escribir en el banco de registros 6 ns y operar en la ALU o en el *Shifter* 10 ns. Suponga que el resto de elementos tienen un retardo despreciable. Complete la Tabla 1 que aparece al final de la pregunta con los datos demandados sobre este procesador.

Para aumentar la productividad de este procesador se procede a segmentarlo en las 5 etapas vistas en clase (LI, DI, EX, M, ER). Sea éste el PROCESADOR 2. Asumiendo que el retardo de los registros de segmentación es de 2ns, muestre en la misma tabla los datos asociados a este procesador.

En un tercer intento de mejorar las prestaciones, se diseña el denominado PROCESADOR 3, que consiste en supersegmentar el procesador original en 7 etapas subdividiendo en dos etapas de igual duración las etapas de memoria. Los registros de segmentación son iguales. Especifique en la Tabla 1 los valores demandados.

Por último, se considera el PROCESADOR 4 que consiste en hacer el PROCESADOR 2 superescalar de dos vías. Indique en la Tabla 1 los valores requeridos para este último caso.

Para comparar las prestaciones de los procesadores entre sí, se recurre a calcular la correspondiente Aceleración. Rellene la Tabla 2, con las aceleraciones demandadas.

Incluya el cálculo correspondiente y las unidades.

TABLA 1 **(0,75 puntos)**

	Tiempo mínimo de ciclo	Frecuencia de Reloj	Productividad máxima
PROCESADOR 1			
PROCESADOR 2			
PROCESADOR 3			
PROCESADOR 4			

TABLA 2 **(0,75 puntos)**

	ACELERACION MAXIMA
PROCESADOR 2 respecto de PROCESADOR 1	
PROCESADOR 3 respecto de PROCESADOR 2	
PROCESADOR 4 respecto de PROCESADOR 2	
PROCESADOR 4 respecto de PROCESADOR 3	

3 **(1 punto)** En el procesador segmentado con 5 etapas del ejercicio anterior (PROCESADOR 2) se va a ejecutar el siguiente fragmento de código en ensamblador del MIPS R2000. Asuma que se utiliza inserción de ciclos de parada tanto para solucionar los conflictos por dependencias de datos como los riesgos de control. La latencia de salto para este procesador es de 1 ciclo.

```
(1)             addi $t1, $zero, 10
(2)     bucle:  lw $t2, 0($t0)
(3)             addi $t0, $t0, -4
(4)             or $t2, $t2, $t3
(5)             addi $t2, $t2, -100
(6)             addi $t1, $t1, -1
(7)             sw $t2, 0($t4)
(8)             bne $t1, $zero, bucle
(9)             addi $t3, $t3, 1000
```

a) **(0.25 puntos)** Identifique los conflictos por dependencia de datos que se producen en dicho código. Rellene para ello la tabla siguiente utilizando tantas filas como necesite:

	Registro	**Número de instrucción en que se escribe**	**Número de instrucción en que se lee**	**Ciclos de parada a insertar**
Riesgo 1				
Riesgo 2				
Riesgo 3				
Riesgo 4				

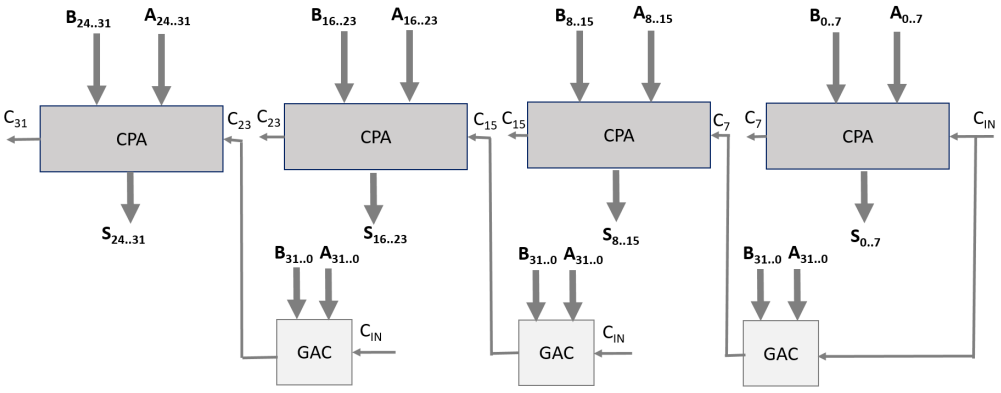
b) (0.5 puntos) Complete el diagrama instrucciones tiempo para este fragmento de código reflejando lo que ocurre en la **<u>última pasada del bucle</u>**

Instrucción/ciclo	x	x+1	x+2	x+3	x+4	x+5	x+6	x+7	x+8	x+9	x+10	x+11	x+12	x+13	x+14	x+15	x+16	x+17	x+18	x+19
`lw`																				
`addi`																				
`or`																				
`addi`																				
`addi`																				
`sw`																				
`bne`																				
`addi`																				

c) (0.25 puntos) Indique para dicho código, **justificando siempre los valores**:

Número total de instrucciones ejecutadas (I)	
Número de ciclos totales de parada (P)	
Número de ciclos totales de ejecución (T)	
CPI (indique las operaciones realizadas para el cálculo)	

4 **(1 punto)** En la figura se muestra un sumador para dos números enteros con signo (A y B) de 32 bits, basado en 4 sumadores CPA de 8 bits y 3 circuitos GAC (Generadores de Acarreo Anticipado). Los CPA están basados en sumadores completos FA como los estudiados en clase. Los GAC son capaces de generar en paralelo las señales de acarreo C_7, C_{15} y C_{23} en función de los valores de entrada A y B y el acarreo entrante C_{IN}, empleando para ello 5ns en todos los casos. Teniendo en cuenta que el retardo de cada puerta lógica es 1 ns, responda a las cuestiones que se plantean a continuación.



a) **(0.5 puntos)** Tiempo necesario para obtener la suma de A y B y Productividad del circuito

b) **(0.25 puntos)** ¿Cómo puede incluirse la detección del desbordamiento en la suma? ¿Cuánto tiempo conllevaría la suma con detección del desbordamiento?

c) **(0.25 puntos)** ¿Cuál es la ganancia de velocidad (aceleración) de este sumador respecto de un CPA para 32 bits?

5 (**1 punto**) En el diseño de la ALU de cierto procesador dotado de registros de 64 bits se plantea el operador de multiplicación de enteros con signo. Hay dos alternativas para este operador:

- Operador secuencial basado en el algoritmo de Booth simple. El sumador/restador tiene un retardo de 5 ns, el desplazamiento de HI-LO (independientemente del número de bits a desplazar) requiere 0.5 ns y el resto de retardos es despreciable.
- Operador secuencial basado en el algoritmo de Booth con recodificación por parejas. El sumador/restador tiene un retardo de 7 ns, el desplazamiento de S-HI-LO (independientemente del número de bits a desplazar) requiere 0.5 ns y el resto de retardos es despreciable.

Calcula, para ambos operadores, la máxima frecuencia a la que puede iterar el autómata y la productividad resultante. En ambos casos el operador necesita un ciclo completo de reloj para inicializar los registros.

	Algoritmo de Booth simple	Algoritmo de Booth con recodificación por parejas
frecuencia máxima de reloj (MHz)		
número de ciclos para operar		
tiempo total de operación (ns)		
productividad del operador (MOPS)		

6 **(1 punto)** Escriba una función *areaT* en lenguaje ensamblador del MIPS R2000 que calcule el área de un triángulo ($A = \frac{b*a}{2}$). La base *b* del triángulo y la altura *a* son dos variables reales que se pasan a la función en los registros $f10 y $f12 respectivamente. Él área se devuelve en el registro $f0. Considere la siguiente declaración de los datos.

```
        .text 0x00400000
__start:
……
areaT :
```

7 (**3 puntos**) Un procesador MIPS R2000 tiene instalados ocho módulos idénticos de 128 MB que llamaremos DRAM0 a DRAM7. DRAM0 está ubicado en la dirección 0x40000000, DRAM1 a continuación de este y así hasta DRAM7

a) **(0.5 puntos)** ¿Cuántas palabras contiene cada módulo? Especifique: número y nombres de las líneas que se usan para selección de palabra (A_x...A_y); número y nombres de líneas de habilitación de byte; y número y nombres de líneas de datos.

b) **(0.25 puntos)** ¿Cuál es la función de selección (a nivel bajo) del módulo DRAM0?

c) **(0.25 puntos)** ¿Cuál es la dirección inicial del módulo DRAM7?

d) **(0.25 puntos)** Si hiciera falta ubicar un nuevo módulo DRAM8 de 256 MB en las direcciones más altas del mapa de memoria, ¿cuál sería su dirección inicial?

e) **(0.25 puntos)** Si el módulo DRAM0 está formado por una fila de ocho chips de memoria, ¿cuál es la organización de cada chip? Exprésela en la forma "Nxw", con los prefijos habituales. ¿Cuántas líneas de selección de palabra tendrá el chip?

f) **(0.5 puntos)** Los chips de memoria de DRAM8 tienen organización 64Mx8 bits. Cada chip contiene cuatro bancos de memoria. Si cada banco contiene 2^{13}=8192 filas, ¿cuál será la capacidad de una fila en KB?

g) **(0.5 puntos)** ¿Cuál será el ancho de banda de cada chip de DRAM8? Suponga que se trata de chips DDR trabajando a 800 MHz. ¿Cuál sería el ancho de banda del módulo?

h) **(0.25 puntos)** Para el chip del apartado anterior, el fabricante proporciona una latencia de CAS (CL) de 13 ciclos. ¿Cuál es el tiempo de acceso en ns de una lectura cuando afecta a una fila abierta?

i) **(0.25 puntos)** Para el mismo chip, según el fabricante, el mínimo tiempo t_{RCD} es de 22 ns. ¿Cuántos ciclos de reloj habrán de separar, como mínimo, las órdenes ACT y LECTURA?