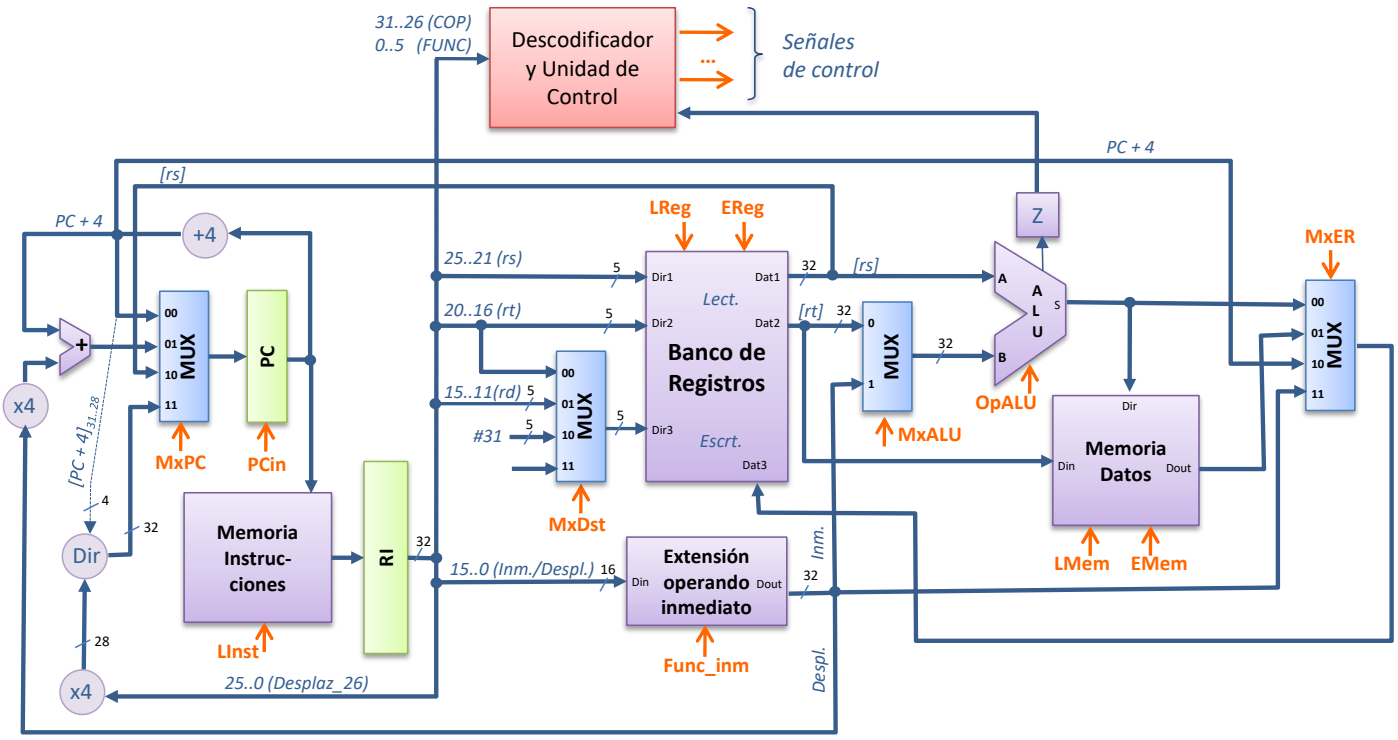
Recuperación Primer Parcial **ESTRUCTURA DE COMPUTADORES** **Enero 2017**

Apellidos y Nombre **DNI** **Grupo**

1 (1.5 puntos) Sea la ruta de datos MONOCICLO del procesador MIPS R2000 que se muestra en la figura adjunta:



Se trata de una ruta ampliada para permitir la ejecución de más instrucciones. Obsérvese que los multiplexores MxPC, MxER y MxDst son ahora de cuatro entradas, por lo que sus señales de control serán de dos bits. También se ha añadido funcionalidad a la ALU y al circuito de extensión del operando inmediato, de acuerdo con las siguientes tablas:

OpALU	Operación
000	S = A **and** B
001	S = A **or** B
010	S = A **xor** B
011	S = A nor B
100	S = A + B
101	S = A - B
110	Si A < B --> S = 1 sino S = 0
111	S = A (identidad)

Func_inm	Operación
00	Extender signo
01	Extender ceros
10	Poner en parte alta y ceros en parte baja
11	No usado

a) (0.5 puntos) Rellene la tabla siguiente con las señales de control requeridas para la ejecución de las instrucciones básicas indicadas:

Instrucción	Form	EReg	OpALU	Func_inm	LMem	EMem	MxPC	MxALU	MxDst	MxER
`sub rd, rs, rt`	R	1	101	X	0	0	00	0	01	00
`xori rt, rs, inm`	I	1	010	01	0	0	00	1	00	00
`lw rt, desp(rs)`	I	1	100	00	1	0	00	1	00	01
`sw rt, desp(rs)`	I	0	100	00	0	1	00	1	X	X
`beq rs, rt, etiq`	I	0	101	00	0	0	0Z	0	X	X

b) (1.0 punto) Indique las señales de control requeridas para la ejecución de las instrucciones siguientes:

lui rt, Inm_16 #, 'Load Upper Inmediate' rt = Inmed_16 || 0x0000
jr rs # PC ←[rs], Salta a la instrucción apuntada por el contenido del registro rs
jal rs # Salto a la subrutina apuntada por el valor del registro rs
$31 ← [PC+4] PC ←[rs]
slt rd, rs, rt # 'Set on Less Than' Si [rs] < [rt] entonces rd = 1 sino rd = 0

Instrucción	Form	EReg	OpALU	Func_inm	LMem	EMem	MxPC	MxALU	MxDst	MxER
lui rt, Inm_16	I	1	X	10	0	0	00	X	00	11
jr rs	R	0	X	X	0	0	10	X	X	X
jal rs	R	1	X	X	0	0	10	X	10	10
slt rd, rs, rt	R	1	110	X	0	0	00	0	01	00

2 (1.5 puntos) En la ruta de datos anterior, asúmase que las operaciones en memoria conllevan 30 ns, leer o escribir en el banco de registros 15 ns y operar en la ALU 35 ns. El resto de retardos es despreciable. Indíquese, **justificando SIEMPRE la respuesta**:

a) (0.3 puntos) La máxima frecuencia de reloj a la que puede trabajar este procesador monociclo.

La instrucción más larga es la lw que hace M+R+ALU+M+R = 30+15+35+30+15 = 125ns
La frecuencia será 1/125ns = 8 MHz

Para aumentar la productividad este procesador se segmenta en las 5 etapas vistas en clase (LI, DI, EX, M, ER). Asumiendo que el retardo de los registros de segmentación es de 5ns, se pide:

b) (0.3 puntos) Frecuencia de reloj del procesador segmentado.

Las etapa más lenta es la EX, que tardan 35ns, por lo que el ciclo de reloj será 35ns + el tiempo del registro de segmentación que es 5ns, total 40ns, y la frecuencia 1/40ns = 25 MHz

c) (0.4 puntos) Productividad máxima del procesador segmentado y Aceleración máxima respecto del procesador original monociclo

En el caso óptimo se ejecuta una instrucción por ciclo de reloj, por lo que la productividad máxima será de 25 MIPS.

Speedup (∞) = Tmonociclo / Tsegmentado = 125 / 40 = 3,125 (para n→infinito)

(El speedup ideal de un procesador es el número de etapas, por tanto 5.)

d) (0.5 puntos) Suponga que se decide supersegmentar el procesador, dividiendo las etapas LI y M en dos sub-etapas de 15ns y la etapa EX en tres sub-etapas de 12ns. Los registros de etapa siguen siendo de 5ns de retardo y 10ns la lectura/escritura del banco de registros. El resultado es un procesador segmentado de 9 etapas. (LI1, LI2, DI, EX1, EX2, EX3, M1, M2, ER). ¿qué aceleración máxima se obtendrá respecto al segmentado de 5 etapas?

Al dividir las etapas más lentas, la nueva etapa más lenta es la de 15ns, sumando el tiempo del registro de segmentación sería un tiempo de ciclo de reloj de 15+5=20ns

Speedup (∞) = T_{5_etapas} / T_{9_etapas} = 40 / 20 = 2 (para n→infinito)

3 (1.5 puntos) En el procesador segmentado de 5 etapas del ejercicio anterior se va a ejecutar el siguiente fragmento de código en ensamblador del MIPS R2000.

```
(1)              lui $t0, 0x1000
(2)              lw $t2, 0($t0)
(3)              beqz $t2, no
(4)              addi $t2, $t2, 1
(5)              j Escr
(6)      no:     li $t2, 0xFFFF
(7)      Escr:   sw $t2, 16($t0)
```

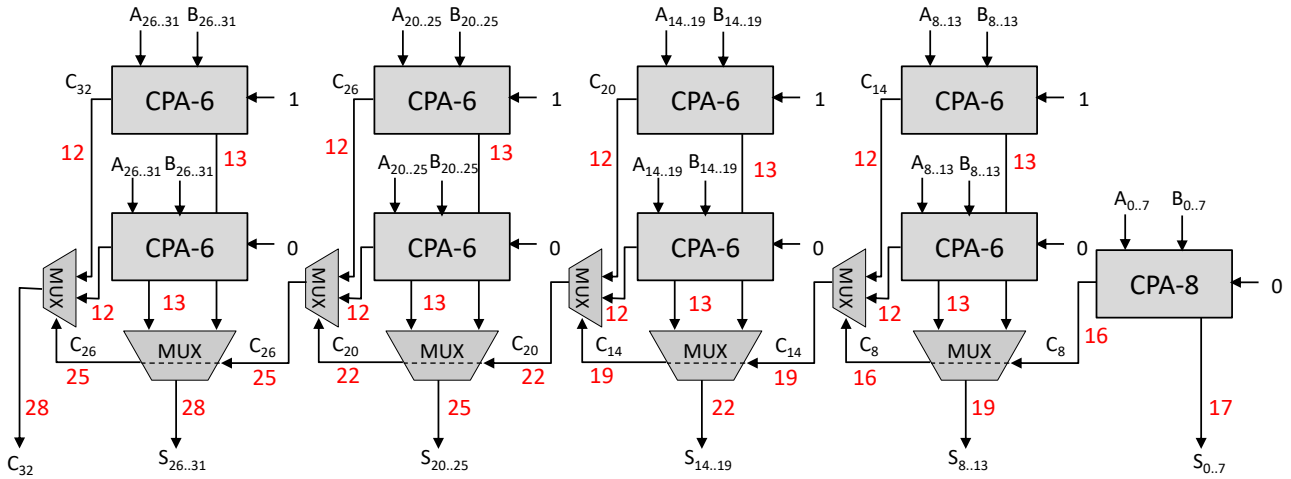
a) (1.0 punto) Asuma que los conflictos por dependencias de datos se solucionan mediante la inserción de ciclos de parada. Los riesgos de control debidos a las instrucciones de salto condicional e incondicional se resuelven también con ciclos de parada. Teniendo en cuenta que para este procesador la latencia de salto es 2 y que el valor de la variable que hay en la dirección 0x10000000 tiene el valor de 50, complete el diagrama instrucciones/ciclo hasta la ejecución de la instrucción (7) inclusive.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
(1) lui	LI	DI	EX	M	ER	$t0													
(2) lw		LI	*DI*	*DI*	DI	EX	M	ER	$t2										
(3) beqz			LI	LI	LI	*DI*	*DI*	DI	EX	M	ER								
(4) addi						LI	LI	*LI*	*LI*	LI	DI	EX	M	ER					
(5) j Escr											LI	DI	EX	M	ER				
(7) sw												LI	LI	LI	DI	EX	M	ER	

b) (0.5 puntos) Indique para dicho código:

Número de ciclos de parada (P)	4 datos + 4 control
Número de ciclos totales de ejecución (C)	18
CPI (indique las operaciones realizadas para el cálculo)	(C-4) / I = (18-4)/6 = 2,333

4 (1.5 puntos) El circuito de la figura siguiente es un sumador para enteros de 32 bits con la técnica CSA (Carry Select Adder) y formado por sumadores de tipo CPA (Carry Propaged Adder) de 8 y 6 bits. Estos sumadores están a su vez implementados por sumadores completos FA (full adder) que generan la suma y el acarreo a través de sus funciones lógicas tal y como se ha visto en clase.



Si cada puerta lógica introduce un retardo de 1 ns y los multiplexores tienen 3 ns de retardo, indíquese:

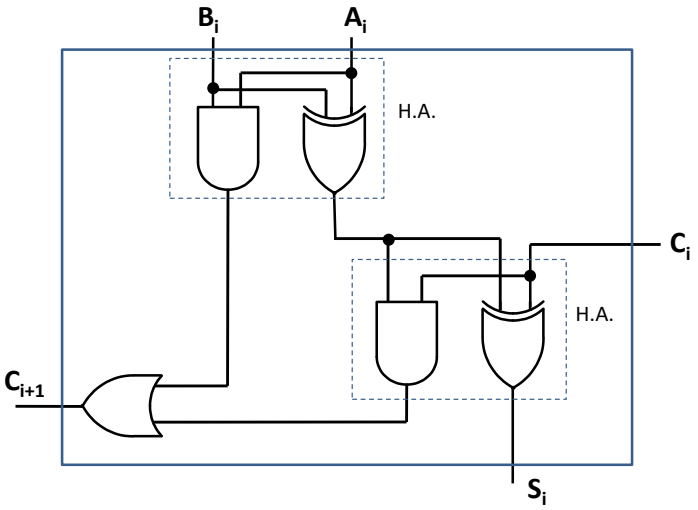
a) (0.5 puntos) El tiempo de respuesta del circuito completo.

Para el FA -> Tcarry = 2ns Tsuma = 3ns
Tiempo de respuesta = 28ns

b) (0.5 puntos) Su productividad en MOPS.

Productividad = 1 op / 28ns = 35,71 MOPS

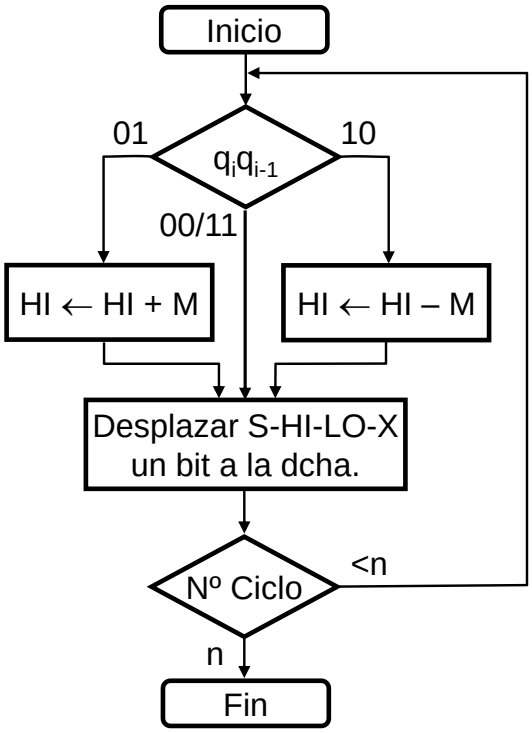
La figura siguiente muestra otra opción para implementar los sumadores completos de un bit (FA), a partir de 2 semisumadores (H.A. Half Adder). En este caso TODAS las puertas tienen un retardo de 1ns. Supóngase ahora que los CPAs que conforman el circuito CSA anterior se implementan con esta nueva opción.

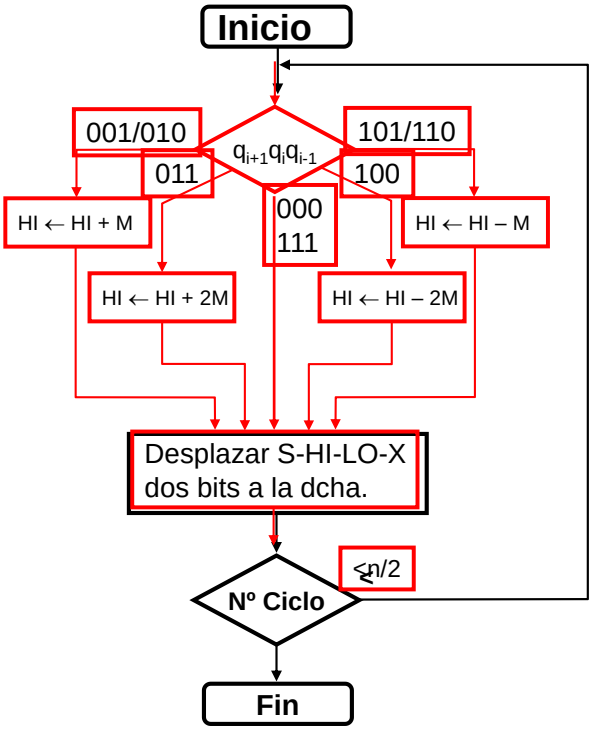


c) (0.5 puntos) ¿Supondrá esta nueva opción un ahorro de tiempo? Razónese la respuesta.

No porque los acarreos estarán disponibles 1T más tarde y el retardo global del CSA será mayor.
Para el FA -> Tcarry = 3ns Tsuma = 2ns

5 (1 punto) El siguiente organigrama (izquierdo) se corresponde con las acciones a realizar por el circuito de control de un multiplicador secuencial que aplica la técnica de Booth. Completa el organigrama de la derecha para que permita al control multiplicar mediante el algoritmo de recodificación por parejas de bits.





6 (1 punto) Para la siguiente declaración de variables en el MIPS R2000 indique cuál/les de los siguientes fragmentos de código no es correcto y porqué.

```
        .data 0x10000000
A:          .word 4
Valor1:     .float 3.756
Valor2:     .double 12.789
Valor3:     .double 1.25
```

a)

```
la $t0, Valor1
lwc1 $f2, 0($t0)
li $t1, 0x7F800000
and $t2, $t1, $f2
```

No es correcto. No se puede hacer la operación AND con registros de coma flotante

b)

```
lw $t2, A
mtc1 $t2, $f2
cvt.s.w $f2, $f2
la $t0, valor1
lwc1 $f1, 0($t0)
add.s $f1, $f1, $f2
```

Es correcto.

c)

```
la $t0, Valor2
lwc1 $f2, 0($t0)
la $t0, Valor3
lwc1 $f4, 0($t0)
mult.d $f6, $f2, $f4
```

No es correcto. Sólo hemos leído de memoria las partes bajas de Valor2 y Valor3, por lo que no se puede hacer la suma en doble precisión.

d)

```
la $t0, A
lw $t2, 0($t0)
cvt.d.w $f2, $t2
```

No es correcto. La instrucción cvt.d.w requiere registros del coprocesador de coma flotante.

7 (1 punto) El cronograma de la figura corresponde a una operación de lectura de un bloque de 4 palabras de un chip de memoria SDRAM que funciona a 200 MHz y tiene un ancho de palabra de 32 bits.

Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Orden	ACT	NOP	NOP	READ	NOP	NOP	NOP	PRE	NOP	NOP				
Dirección	FILA			COL										
Datos							W0	W1	W2	W3				

Resuelva las siguientes cuestiones:

a) (0.25 p) Complete el cronograma indicando el envío de las direcciones de fila y columna en el ciclo correspondiente así como la orden de precarga.

b) (0.25 p) ¿Cuál es la Latencia de CAS de esta memoria en ciclos de reloj? ¿Cuál es el **tiempo de acceso**, expresado en nanosegundos?

CAS Latency = 3 ciclos
Tacceso = 6 x Tclk = 6 x 5 = 30ns

c) (0.25 p)¿Cuál es su **ancho de banda?**

B = f x w (simple velocidad) = 200 x 4 = 800 MBps

d) (0.25 p)¿Cómo se vería afectado el tiempo de acceso y el ancho de banda en el caso de que el fabricante sacase una versión DDR de esta misma memoria (manteniendo sus mismos parámetros temporales internos)?

Tacceso igual, B se duplicaría

8 (1 punto) Un antiguo PC de los expuestos en el Museo de Informática tiene un procesador 80286 cuyo ancho de palabra es de 16 bits y utiliza direcciones de memoria de 24 bits.
Este computador disponía de un módulo M1 de 2 MB de memoria RAM disponible desde la dirección 0x000000 y de un módulo M2 de 8MB al final de espacio de direccionamiento.
Conteste las siguientes cuestiones:

a) (0.25 p) ¿Cuánta memoria podía direccionar este procesador? ¿ Cuantas líneas de selección de byte (BE*) tiene ¿

Memoria direccionable = 16 MB Líneas de selección de byte = BE1* y BE0*

b) (0.25 p) ¿Cuál es la última dirección correspondiente al módulo 1?

0x1FFFFF

c) (0.25 p) ¿Cuál es la primera dirección correspondiente al módulo 2?

0x800000

d) (0.25 p) Exprese la función de selección (utilizando lógica negativa) para el módulo 2

$\overline{SEL_{M2}} = /A_{23}$