Examen Primer Parcial **ESTRUCTURA DE COMPUTADORES** Diciembre 2014

Apellidos y Nombre	DNI	Grupo
SOLUCIÓN		

1. (**2 puntos**) Se desean incluir las instrucciones `ori rt,rs,inm` y `andi rt,rs,inm` como dos instrucciones más del juego básico del MIPS visto en clase. Las instrucciones tendrán formato I según la Figura 1.
   Teniendo en cuenta que:
   - La instrucción ori hace la operación OR bit a bit entre el dato inmediato, extendiendo ceros, y el registro rs. El resultado se almacena en rt.
   - La instrucción andi hace la operación AND bit a bit entre el dato inmediato, extendiendo unos, y el registro rs. El resultado se almacena en rt. (OJO: es una instrucción con un comportamiento diferente del que tiene la del procesador MIPS original)

Para dar soporte a ambas instrucciones la ruta de datos de la Figura 2 incluye un nuevo operador multifunción que según la señal de control aplicada extiende 1´s, 0´s o el bit de signo del valor inmediato, según se indica en la Tabla 1.

OpExtension	Función
00	Extiende signo
01	Extiende 1´s
10	Extiende 0´s
11	____

Tabla 1

OpALU	Operación
000	and
001	or
010	suma
110	resta
111	

Tabla 2

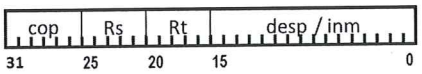

Figura 1

a) **(1 punto)** Marque sobre esta ruta de datos los caminos activos para ejecutar estas instrucciones. Considérese también la etapa de búsqueda de la instrucción y las acciones correspondientes.

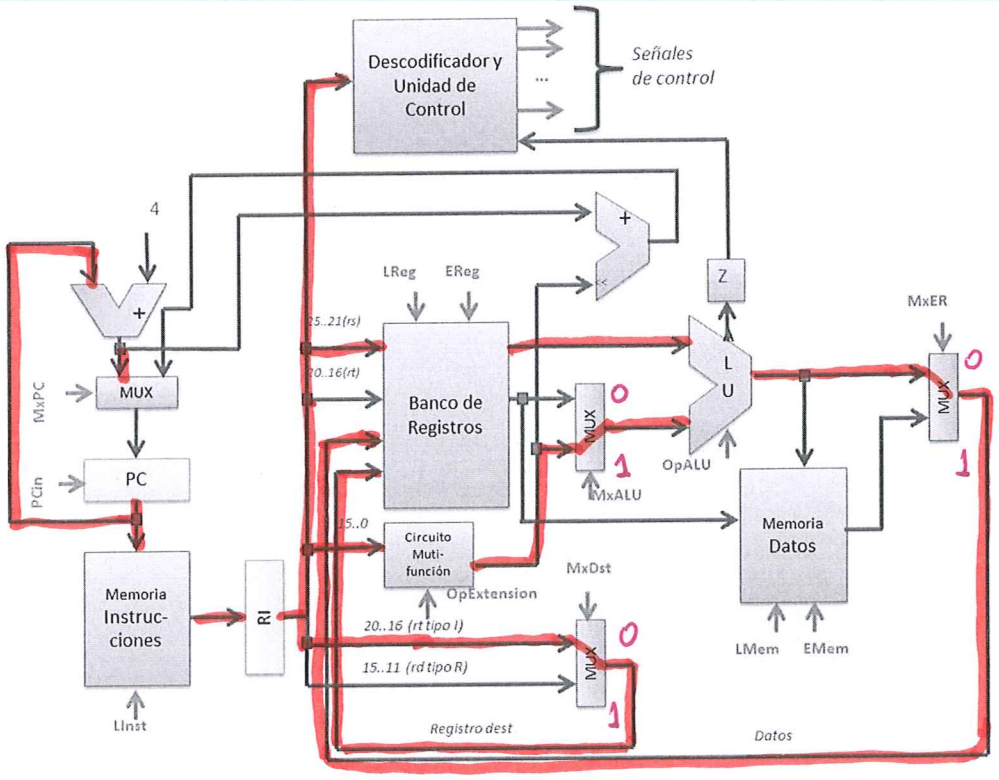

Figura 2. Ruta de los datos.

b) **(1 punto)** Complete la tabla correspondiente a las señales de control para poder ejecutar las instrucciones que se indican. Las señales de operación en la ALU se codifican según la Tabla 2.

Instrucción	Form	EReg	OpALU	OpExtension	LMem	EMem	MxPC	MxALU	MxDst	MxER
`sub rd, rs, rt`	R	1	110	xx	0	0	0	0	1	0
`lw rt, desp(rs)`	I	1	010	00	1	0	0	1	0	1
`beq rs, rt, etiq`	I	0	110	00	0	0	2	0	x	x
`ori rt, rs, inm`	I	1	001	10	0	0	0	1	0	0
`andi rt, rs, inm`	I	1	000	01	0	0	0	1	0	0

2. **(0.5 puntos)** El procesador monociclo de la pregunta 1 tiene una frecuencia de reloj de 8 MHz. En él se ejecuta el siguiente fragmento de código del MIPS R2000

```
(1)          ori $8, $0, 20
(2)  do:     lw    $7, 0($4)
(3)          add   $6, $6, $7
(4)          addi  $4, $4,4
(5)          addi  $8, $8, -1
(6)          bgtz  $8, do
```

JUSTIFIQUE LAS RESPUESTAS, NO PONGA SOLO LA SOLUCIÓN.

Especifique:

a) (0.25 puntos) Tiempo necesario para ejecutar dicho código

$\tau = 1/F = 125\,ns$

$T = (5\,inst * 20 + 1) \cdot \tau = 12{,}6\,\mu s$

b) (0.25 puntos) Productividad de este procesador

$P = \frac{n}{T} = \frac{101}{12{,}6} \simeq 8\,MIPS$

3. **(0.75 puntos)** El procesador anterior se rediseña segmentándolo en 5 etapas (LI, DI, EX, M, ER). La duración de las etapas con acceso a memoria es de 35 ns y las de decodificación, ejecución y escritura en registros 20 ns. Asuma que los registros de segmentación introducen un retardo de 5 ns. Indique, justificando en todos los casos el resultado indicado:

a) **(0,25 puntos)** Frecuencia de reloj del procesador segmentado

$\tau = \tau_{max} + t_{reg} = 35 + 5 = 40\,ns$

$f = 1/\tau = 25\,MHz$

b) **(0,25 puntos)** Productividad máxima que puede llegar a ofrecer

$P_{(n \to \infty)} = 25\,MIPS$ — 1 instrucción por ciclo

c) **(0,25 puntos)** Aceleración máxima respecto del procesador original monociclo

$S_{n \to \infty} = \frac{T_{no\,seg}}{T_{seg}} = \frac{125}{40} = 3{,}125$

4. **(1.5 puntos)** Sobre el procesador segmentado se ejecuta el mismo código anterior. Este procesador tiene latencia de salto 2 y los conflictos de control se resuelven mediante la inserción de instrucciones NOP. Para la resolución de los conflictos debidos a dependencias de datos, el procesador puede hacer uso de técnicas de anticipación implementando los cortocircuitos vistos en clase (MaEX, ERaEX, ERaM). En caso de no poder resolver enteramente el conflicto mediante estas técnicas el compilador insertará instrucciones NOP.

   a) (**0.5 puntos**) Indique utilizando la tabla los **conflictos en los datos** que se producen así como la solución más eficiente para los mismos. (En caso de insertar alguna NOP mantenga la numeración original de las instrucciones tal y como indica el enunciado del ejercicio 2.

	Registro	Número de instrucción en que se escribe	Número de instrucción en que se lee	Solución
Riesgo 1	$7	2	3	NOP + ERaEX
Riesgo 2	$8	5	6	MaEX
Riesgo 3				
...				

   b) (**1 punto**) Indique para dicho código y especifique cómo obtiene el resultado:

Número total de Instrucciones ejecutadas (I)	1 + (5 + 1NOP + 2NOP) * 20 = 161
Número de ciclos totales de ejecución (T)	= K+n-1 = 5+161-1 = 165
CPI	1

5. **(0.75 puntos)** Se decide supersegmentar el procesador anterior en 7 etapas descomponiendo las etapas de memoria en dos subetapas de 18 ns de duración cada una (el resto de etapas mantienen sus retardos). Indique para este nuevo diseño del procesador (justificando las respuestas en todos los apartados):

   a) (**0,25 puntos**) Productividad máxima que puede llegar a ofrecer

$\tau_{nuevo} = \tau_{max} + t_{reg} = 20 + 5 = 25\ us$

$P(n \to \infty) = 40\ MIPS$ (1 instrucción por ciclo)

   b) (**0,5 puntos**) Aceleración máxima respecto a los dos diseños anteriores y aceleración ideal

Aceleración respecto del procesador monociclo: $S = \frac{T_{no\ seq}}{T_{super}} = \frac{125}{25} = 5$

Aceleración respecto del procesador segmentado en 5 etapas

$S = \frac{T_{seq}}{T_{super}} = \frac{40}{25} = 1,6$

Aceleración ideal

$S_{ideal} \to K = 7$

6. **(1.25 puntos)** La figura 3 muestra la conexión genérica de sumadores con propagación de acarreo (CPA) para formar un sumador con selección de acarreo (CSA). Se desea diseñar uno para sumar números enteros de 32 bits. Para ello se dispone de CPA´s de 4 bits y CPA´s de 8 bits. Suponiendo que estos CPA´s no son compatibles entre ellos, indique en base al tiempo de respuesta y productividad qué diseño es el más eficiente. Considere que el retado de todas las puertas es el mismo e igual a 1 ns y que el multiplexor tiene 3 niveles de puertas.

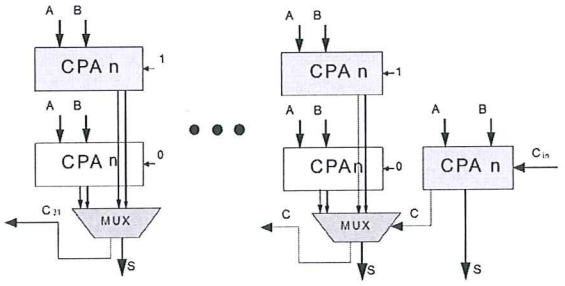

Figura 3

Los CPA se encuentran implementados con puertas lógicas, siendo necesario atravesar 3 niveles de puerta para la obtención del correspondiente bit de suma (S) y 2 para el de acarreo (C).

CSA basado en 15 CPA´s de 4 bits	$T_{respuesta}$ = 29T	Productividad = 33,48 MOPS
CSA basado en 7 CPA´s de 8 bits	$T_{respuesta}$ = 25T	Productividad = 40 MOPS

JUSTIFIQUE LOS CÁLCULOS:

Caso CPA 4 bits S = 9T C = 8T Este C selecciona la salida del primer MPX, que saca la S en 12T y el C = 11T. Ahora las señales de C se van propagando por todos los MPX añadiendo 3T por cada uno

Caso CPA 8 bits S = 17T C = 16T. El acarreo selecciona la salida del 1er MPX que saca la suma en 20T y el acarreo en 19T. Este es el que se propaga en los 2 siguientes MPX añadiendo 3T por cada paso.

7. **(1.25 puntos)** Considere los siguientes números enteros de 6 bits expresados en complemento a dos:

M = 1 0 1 1 0 0 y Q = 0 1 1 0 0 1 Ca2M = 010100

a) Realice "a mano" la multiplicación de M*Q tanto mediante el algoritmo de Booth como por recodificación por parejas. En ambos casos muestre claramente cuál es la recodificación correspondiente del multiplicador.

Algoritmo de Booth (0. 5 puntos)	**Recodificación por parejas** (0. 5 puntos)
Recodificación del Multiplicador Q' = +1 0 -1 0 +1 -1	Recodificación del Multiplicador Q'' = +2 -2 1
Multiplicación: 101100 x +1 0 -1 0 +1 -1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0	Multiplicación: 101100 x +2 -2 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0

b) En caso de utilizar para la multiplicación un circuito secuencial como el visto en clase, indique: (0. 25 puntos)

	Algoritmo de Booth	Alg. Recodificación por parejas
Número de iteraciones necesarias para realizar el producto	6	3
Valor del registro HI en la inicialización	000000	000000
Valor de registro LO en la inicialización	011001	01100 1
Valor del registro HI al final de la operación	111000	111 00 0
Valor del registro LO al final de la operación	001100	0011 0 0
Valor del bit de signo después de la operación (N)	1	1

8. (**1 punto**) Escriba el código de una subrutina **conv-temp** que convierte una temperatura expresada en grados Fahrenheit a su correspondiente valor en grados Celsius. Para ello se utilizará la fórmula:

$$T_{Celsius} = \frac{5.0}{9.0} * (T_F - 32.0)$$

La temperatura en grados Fahrenheit se pasa como parámetro a la rutina en el registro $f12 y el resultado se devuelve en $f0. Puede utilizar pseudointrucciones `li.s` para inicializar con valores reales. Utilice datos en simple precisión.

```
conv-temp : li.s $f4, 5.0
            li.s $f6, 9.0
            div.s $f0, $f4, $f6   # $f0 = 5.0/9.0
            li.s $f4, 32.0
            sub.s $f4, $f12, $f4  # $f4 = TF - 32.0
            mul.s $f0, $f0, $f4   # $f0 = TC
            jr $ra
```

9. (**1 punto**) Complete el circuito de la figura 5 en las zonas punteadas para que sea capaz de dar soporte a la instrucción de coprocesador en coma flotante del MIPS `cvt.d.s fd,fs`. El circuito no considerará los casos especiales. Justifique la respuesta

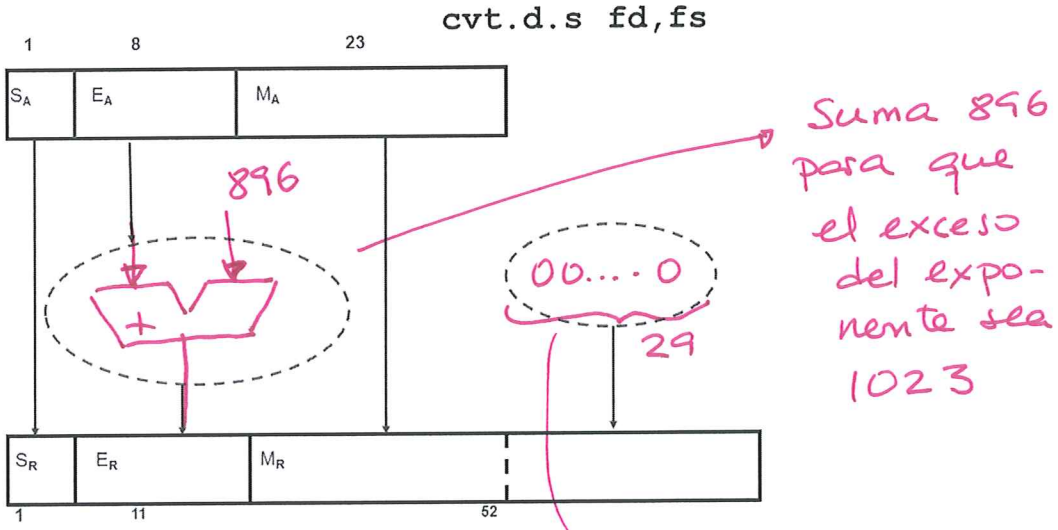

Figura 5