Apellidos y Nombre: .....Grupo:.....DNI: .....

Examen 4 (temas 12, 13 y 14)

- Duración: 2 horas. La solución de cada ejercicio se tiene que escribir en el espacio reservado para ello en el propio enunciado.
- No podéis utilizar calculadora, móvil, apuntes, etc. En hoja aparte se os da una "chuleta" con información útil para realizar los ejercicios.
- La solución se publicará mañana y las notas antes de una semana. La revisión será el 7 de junio a las 12:00 en el aula C6-E101.

Ejercicio 1 **(0,5 puntos)**

Completad el código SISA para que multiplique por 6 el contenido de `R0` y deje el resultado en `R0`. Se considera que `R0` contiene un número natural codificado en binario. El código calcula correctamente el resultado excepto si no se puede representar en binario con 16 bits. El código usa `R7` como registro temporal, por lo que `R7`quedará modificado.

```
[      ]  R7, [      ]
[      ]  R7, [      ]
[      ]  R7, [      ]
[                    ]
```

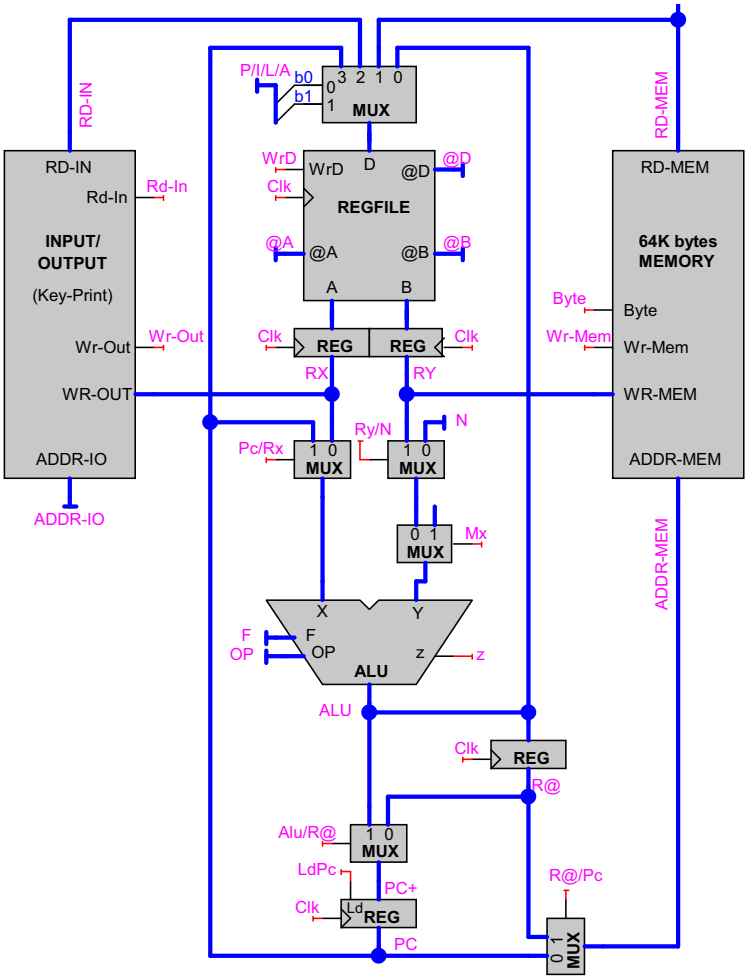
Ejercicio 2 **(3,75 puntos)**

Completad el diseño del SISC Von Neumann para que pueda ejecutar, además de las 25 instrucciones originales, las dos nuevas instrucciónes, `MUL5` y `MUL6`, con la siguiente codificación en lenguaje máquina, sintaxis ensamblador y semántica:

```
Codificación MUL5:   1110 aaa ddd xxxxxx
Sintaxis MUL5:       MUL5 Rd, Ra
Semántica MUL5:      Rd  = Ra * 5;   La multiplicación es de números naturales

Codificación MUL6:   1111 aaa ddd xxxxxx
Sintaxis MUL6:       MUL6 Rd, Ra
Semántica MUL6:      Rd  = Ra * 6;   La multiplicación es de números naturales
```

Para su implementación se debe modificar la UP (solamente añadiendo un MUX-2-1 con señal de selección Mx, como se muestra en la figura) y la UC (que debe generar la nueva señal Mx en su palabra de control y ser capaz de ejecutar el nuevo grafo de estados, con tres nuevos nodos para implementar las dos nuevas instrucciones, para lo que solamente se debe modificar la ROM_OUT y la ROM_Q+).



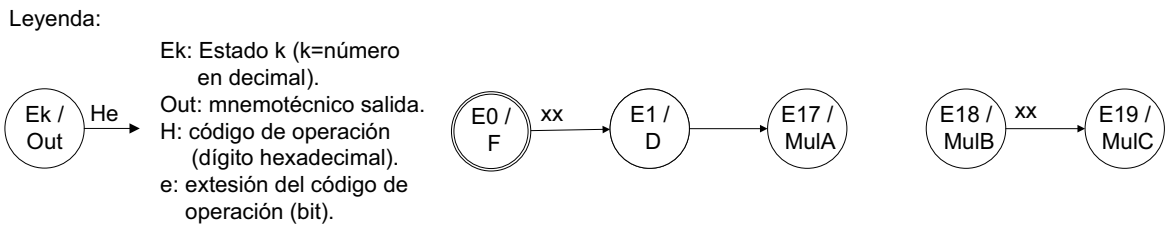
Se pide:

a) Dibujad, sobre la figura de la UP, la conexión que falta: el bus de la entrada 1 del nuevo MUX-2-1 con señal de selección Mx. Si lo deseáis podéis utilizar conexión por nombre, como permite LogicWorks. (**0,25 puntos**)

b) Escribid la acción (o acciones en paralelo) que se realiza en el computador para cada uno de los tres nodos involucrados en la ejecución de las dos nuevas instrucciones (MulA, MulB y MulC), además de para los estados F y D. Para especificar las acciones en la siguiente tabla debéis usar el mismo lenguaje de transferencia de registros que en la documentación u otro que se entienda sin ambigüedad. (**1,25 puntos**)

Número	Mnem.	Acciones
E0	F	
E1	D	
E17	MulA	
E18	MulB	
E19	MulC	

c) Completad el fragmento del grafo de estados del circuito secuencial de la unidad de control de la figura. Se da la leyenda del grafo y todos los nodos necesarios para ejecutar las dos nuevas instrucciones, pero faltan arcos y etiquetas. Dibujad todos los arcos que faltan y todas las etiquetas. Os pedimos que no dibujéis ningún otro nodo. (**0,25 puntos**)



d) Indicad la dirección o las direcciones (en binario con x cuando sea posible para referirnos a más de una dirección) de la memoria ROM_Q+ y su contenido (en hexadecimal) para implementar correctamente el arco o los arcos que salen de cada uno de los nodos E1 y E17. Para cada arco indicad el nodo origen y el destino. (**0,5 puntos**)

e) Completad (poniendo 0, 1 o x en cada bit) la nueva columna Mx y las 5 filas F, D, MulA, MulB y MulC de la tabla que especifica el contenido de la ROM_OUT para que se ejecuten correctamente las dos nuevas instrucciones y las 25 originales, poniendo el máximo número de x posibles. La dirección 0 de la ROM corresponde al estado E0 (F), la 1 al E1 (D), la dirección 17 al estado E17 (MulA), etc. El resto de filas en blanco de la siguiente tabla son para completar en el ejercicio siguiente. (**1,5 puntos**)

Apellidos y Nombre: .....Grupo:.....DNI: .....

@ROM	Mx	Bnz Bz	WrMem RdIn WrOut	WrD	Ldlr	Byte	R@/Pc Alu/R@ Pc/Rx Ry/N	P/I/L/A1 P/I/L/A0	OP1 OP0	MxN1 MxN0	MxF	F2 F1 F0	Mx@D1 Mx@D0	Node
0														F
1														D
2														Al
3														Cmp
4														Addi
5														Addr
6														Ld
7														St
8														Ldb
9														Stb
10														Jalr
11														Bz
12														Bnz
13														Movi
14														Movhi
15														In
16														Out
17														MulA
18														MulB
19														MulC
20..31														Nop

Ejercicio 3 (1 punto)

Aprovechando la tabla del ejercicio anterior del contenido de la ROM_OUT del SISC Von Neumann, completad también las 4 filas Addi. Stb, Bnz y Movhi (que también se encuentran con fondo blanco, para ser rellenadas).

Ejercicio 4 (1,5 puntos)

Cada uno de los apartados pregunta sobre un ciclo concreto de la ejecución de una instrucción en el SISC Von Neumann. **Escribid el valor de los bits de la palabra de control** que genera el bloque SISC Von Neuman CONTROL UNIT durante el ciclo a que hace referencia cada apartado. Poned x siempre que no se pueda saber el valor de un bit (ya que no sabemos cómo se han implementado las x en la ROM_OUT). La segunda y tercera columna definen la situación en la que se encuentra la Unidad de Control (UC) para cada apartado/fila: nodo/estado de la UC en ese ciclo e instrucción (en ensamblador) que está almacenada en el IR en ese ciclo. Podéis ver el grafo de estados de Moore de la UC en el anexo. En el ciclo en que se ejecuta cada apartado el contenido de todos los registros, `Ri`, es cero.

Apartado	Nodo/Estado (Mnemo Salida)	Instrucción en IR (en ensamblador)	Palabra de Control: @A	@B	Pc/Rx	Ry/N	OP	F	P/I/L/A	@D	WrD	Wr-Out	Rd-In	Wr-Mem	Ldlr	LdPc	Byte	Alu/R@	R@/Pc	N (hexa)	ADDR-IO (hexa)
a	D	`LDB R2, 0(R0)`																			
b	Movhi	`MOVHI R4,lo(-9)`																			
c	Addr	`ST -4(R2),R4`																			

Ejercicio 5 **(3,25 puntos)**

a) Completad el código SISA para que obtenga la representación en binario de un número natural codificado con 5 dígitos en el sistema convencional en base 6.

Los 5 dígitos del número en base 6 están almacenados en memoria a partir de la dirección simbólica `B6`, comenzando por el dígito de más peso. Cada dígito, D_i, con $0 \le D_i \le 5$, se encuentra codificado en binario ocupando un byte en memoria. Después de la ejecución del programa, el valor del número en base 6 se almacena codificado en binario (sistema convencional en base 2) en la palabra (16 bits) con dirección simbólica `B2`.

Para obtener la representación en binario del número a partir de los n dígitos que lo representan en base 6 se efectúan los cálculos en binario de la siguiente fórmula. El código usa la expresión sin sumatorio operando de izquierda a derecha (del dígito de más peso, D_4, al de menos peso, D_0).

$$\sum_{i=0}^{4} D_i 6^i = (((((0)6 + D_4)6 + D_3)6 + D_2)6 + D_1)6 + D_0$$

En el código, `R0` hace de acumulador y se inicializa a 0. Cada iteración del bucle calcula: `R0` = `R0` × 6 + D_k desde `k=4` hasta `k=0`. Así, la primera vez que se entra en el bucle `R0` vale 0 y D_k es D_4. Para multiplicar por 6 tal vez se pueda usar el mismo código que en el *Ejercicio 1* (eso depende de cómo se haya escrito ese código) **(1 punto)**

```
.data
        .even
B6:     .byte 0,1,0,2,3
        .even
B2:     .space 2, 0
.text
Begin:  MOVI    R0, 0
        MOVI    R2, lo(B6)
        [      ] R2, [      ]
        MOVI    R5, 5
 Loop:  [      ] R7, [      ]
        [      ] R7, [      ]
        [      ] R7, [      ]
        [                    ]
        [      ] R7,  0(R2)
        [      ] R0,  R0, R7
        A[     ] R2, [      ]
        A[     ] R5, [      ]
        B[     ] R5, [      ]
Save:   ST      [           ]
.end
```

b) El programa se ha traducido a lenguaje máquina situando la sección `.text` a partir de la dirección 0x08F8 de memoria y a continuación la sección `.data`. Una vez cargado el programa en memoria y ejecutado ¿Cuál es el contenido de las siguientes palabras de memoria? **(1 punto)**

MEMw[0x08FA] = 0x	MEMw[0x0910] = 0x
MEMw[0x0916] = 0x	MEMw[0x091A] = 0x

c) Indica claramente qué cambios hay que realizar a partir del código del apartado a) para obtener un código que, usando las nuevas instrucciones definidas en el *Ejercicio 2*, realice la misma funcionalidad que el código del apartado a). Indica qué instrucciones hay que eliminar y cuales hay que añadir y donde. (**0,5 puntos**)

d) ¿Cuanto tarda en ejecutarse el código del apartado a) en el computador SISC Von Neumann suponiendo un tiempo de ciclo de 1.000 ut? ¿Cuánto tarda en ejecutarse el código resultante del apartado c) en el SISC Von Neumann ampliado con las nuevas instrucciones del *Ejercicio 2* suponiendo que el tiempo de ciclo no ha variado y que la instrucción `MUL5` se ejecuta en 4 ciclos y la `MUL6` lo hace en 5? ¿Cuánto vale x para que sea cierta la siguiente afirmación? "El código del apartado a) con los cambios del apartado c), ejecutándose en el computador ampliado, tarda un x% menos que el código del apartado a) ejecutándose en el computador original". (**0,75 puntos**)

Tejecución (Código a) = ... u.t. Tejecución(Código c) = ... u.t. x= .....%