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

# Examen Final, Parte 2

Duración de esta parte: 2 horas. Escribid las respuestas en el enunciado y entregad todas sus hojas con apellidos y nombre en la cabecera. 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 ¿?/??/???? a las ¿?:?? en el aula C6-E101.

## Ejercicio 1 (1,5 puntos)

El programa ensamblador se ha traducido a lenguaje máquina para ser ejecutado en el SISC Von Neumann, situando la sección .data a partir de la dirección 0x80A0 de memoria y a continuación la sección .text.

Una vez ensamblado y cargado el programa en memoria:

a) ¿A qué direcciones de memoria corresponden las etiquetas, o direcciones simbólicas siguientes? (0,25 per error)

```
C=0x80A2        E=0x80AC        L2=0x80D4
```

b) ¿Cuál es la dirección de memoria y su contenido donde han quedado almacenadas las siguientes instrucciones? (0,25 per error)

```
MOVHI   R2, HI(C)    => Mem_w[0x80CE ] = 0x9580
BZ      R0, L1       => Mem_w[0x80C8 ] = 0x80FE
LD      R5, 0(R2)    => Mem_w[0x80D4 ] = 0x3540
```

```
.data
     N = 5
A:   .space 1
B:   .byte -27
     .even
C:   .word 2,-5,264,-63,23
D:   .byte 58,-64,32,0,-7
     .even
E:   .space 20
.text
L1:  IN      R0, KEY-STATUS
     BZ      R0, L1
     IN      R0, KEY-DATA
     MOVI    R2, LO(C)
     MOVHI   R2, HI(C)
     MOVI    R1, N
     MOVI    R4, 1
L2:  LD      R5, 0(R2)
     ...
```

## Ejercicio 2 (1 punto) (0,25 per fila/col amb error)

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. Suponed, para responder al apartado b, que el contenido de R4 antes de ejecutarse la instrucción BNZ R4, -7 es 0xFFFF.

| 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** | Addi | `ADDI R5, R2, -10` | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | X | 0 | x | x | x | F | F | F | 6 | 7 | 6 |
| **b** | Bnz | `BNZ R4, -7` | 1 | 0 | 0 | 1 | 1 | 1 | 0 | x | 1 | 0 | 0 | 0 | 0 | x | x | x | x | x | 0 | 0 | 0 | 0 | x | 1 | x | 0 | x | X | X | X | X | F | 9 |
| **c** | Addr | `LD R1, 28(R2)` | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | x | x | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | 0 | 0 | 1 | C | 5 | C |

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

**Ejercicio 3 (1 punto)** (0,25 per fila/col amb error)
Completad la tabla que representa en forma compacta parte del contenido de la ROM_OUT de la unidad de control del SISC Von Neumann. Poned x siempre que un bit pueda valer tanto 0 como 1.

| @ROM | Bz | Ldlr | R@/Pc | Alu/R@ | Pc/Rx | Ry/N | MxN1 | MxN0 | MxF | Mx@D1 | Mx@D0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 0 | 0 | x | x | 0 | 0 | 0 | 0 | 1 | x | x | Addr |
| 9 | 0 | x | 1 | x | x | x | x | x | x | x | x | Stb |
| 11 | 1 | x | x | 0 | 0 | x | x | x | 1 | x | x | Bz |
| 14 | 0 | x | x | x | 0 | 0 | 0 | 1 | 1 | 1 | 0 | Movhi |

**Ejercicio 4 (1,5 puntos)** (0,5 per fila correcte, sense cap error)
Indicad qué cambios hay en el estado del computador SISC Von Neumann después de ejecutar cada una de las instrucciones de la tabla suponiendo que antes de ejecutarse cada una de ellas el PC vale 0x3C18, el contenido de todos los registros es 0x9ABC y que el contenido de todas las direcciones pares de la memoria es 0xA5 y el de todas las impares es 0x00. Utiliza el mnemotécnico MEMb[...]=... y/o MEMw[...]=... para indicar los cambios en la memoria.

| Instrucción a ejecutar | Cambios en el estado del computador |
|---|---|
| `LDB R2, -1(R6)` | R2 = 0x0000, PC = 0x3C1A |
| `STB -4(R1), R2` | $Mem_b$[ 0x9AB8] = 0xBC, PC = 0x3C1A |
| `ST  1(R3), R2` | $Mem_w$[ 0x9ABC] = 0x9ABC, PC = 0x3C1A |

**Ejercicio 5 (5 puntos)**
Vamos a diseñar el SISC2020 Von Neumann para que pueda ejecutar código SISA2020. El SISA2020 es igual al SISA excepto que:

1. El estado del computador del SISA2020 es el mismo que el del SISA más un nuevo registro especial de un bit (biestable) que denominamos Fz (Flag z). El Fz solamente lo escriben todas las instrucciones de tipo Aritmético-Lógico, AL, y Comparación, CMP, que además de incrementar en dos el PC (como todas las instrucciones) y escribir el registro destino (Rd) con el resultado de la operación, modifican Fz como se indica a continuación (la siguiente sentencia hay que añadirla a la semántica original de las instrucciones AL y CMP en el SISA2020):

if (Rd == 0) Fz = 1; else Fz = 0;

El SISA2020, además de las 25 instrucciones originales (incluida la modificación anterior en las AL y CMP) tiene nuevas instrucciones aritmético-lógicas y de comparación Condicionales, que denominamos ALc y CMPc. Estas nuevas instrucciones condicionales hacen lo mismo que las originales SISA pero solo escriben el resultado en Rd si Fz vale 0. Las instrucciones ALc y CMPc no escriben Fz, solo lo leen para decidir si se escribe Rd. La definición de las nuevas instrucciones es:

**Codificación**: cccc aaa bbb ddd fff (excepto la NOT que en bbb tiene xxx)
Siendo cccc igual a 1110 para las instrucciones ALc y 1111 para las CMPc. La codificación del campo fff para las instrucciones ALc y CMPc es la misma que para las AL y CMP, respectivamente.

**Sintaxis**: MNEMOc Rd, Ra, Rb
siendo MNEMO cualquiera de los mnemotécnicos de las operaciones AL o CMP originales. Ejemplo de instrucción ALc: SUBc Rd, Ra, Rb. Ejemplo de CMPc: CMPLEUc Rd, Ra, Rb.

**Semántica**: if (Fz == 0) Rd = Ra op Rb;
siendo op la operación AL o CMP que corresponde al MNEMO (de la instrucción en ensamblador) o al código de operación y campo fff (de la instrucción en lenguaje máquina).

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

En general, el SISA2020 es más eficiente que el SISA original ya que se puede reducir, en algunos casos, el número de instrucciones BZ y BNZ utilizadas. Las primeras dos preguntas de este ejercicio tratan este tema:

a) Completad los código SISA original y SISA2020 para que cada uno de ellos realice la siguiente funcionalidad (La comparación es de números naturales. Usad R7 como registro temporal si es necesario): (0,75 puntos, 0,25 por error)

if (R5 >= R6) R0 = R0 + R1;

```
SISA:   CMPLEU  R7, R6, R5          SISA2018:  CMPLEU  R7, R6, R5
        BZ      R7, 1                          ADDc    R0, R0, R1
        ADD     R0, R0, R1
```

b) ¿Cuanto vale x para que sea cierta la siguiente afirmación? "El código SISA del apartado anterior ejecutado en el SISC Von Neumann es un x% más lento que el código SISA2020 ejecutado en el SISA2020 Von Neumann". Suponed que los dos computadores tienen el mismo tiempo de ciclo, que es Tc=2000 u.t., que al ejecutarse el código R5 es mayor o igual que R6 y que en el SISC2020 las instrucciones tardan el mismo número de ciclos en ejecutarse que en el SISC (las ALc y CMPc tardan lo mismo que las AL y CMP). (0,25 puntos, BINARI)

x = 33,3 %

A continuación, vamos a modificar el diseño del SISC Von Neumann para que pueda ejecutar el lenguaje máquina SISA2020 y convertirlo así en el computador SISC2020 Von Neumann. Para ello, solamente se requieren hacer los siguientes cambios en la Unidad de Control (UC) del SISC Von Neumann:

1. Se añade un biestable D activado por flanco ascendente de la señal de reloj, que implementa el registro especial de estado, de 1 bit, Fz (la salida de este biestable es Fz). Este biestable lo podríamos disponer en la UP, pero lo dispondremos en la UC, ya que esto es indiferente.
2. La ROM_OUT deberá generar dos nuevas señales de salida, denominadas Cond y LdFz, para poder implementar las diferencias entre SISA y SISA2020. La señal Cond valdrá 1 en el ultimo ciclo de ejecución de cualquiera de las nuevas instrucciones Condicionales y 0 en cualquier otro caso. La señal LdFz sirve, como se indica en el punto 4, para gestionar la carga de Fz. La ROM_Q+ también deberá cambiar para adaptarse a los nuevos nodos.
3. Se cambia el nombre de la señal de salida de la ROM_OUT, WrD, que pasará a llamarse WrD1, para diferenciarla de la señal WrD que sale de la Lógica de Control para ir a la UP como parte de la palabra de control, ya que ahora no siempre WrD (de la palabra de control) será igual a WrD1 (de la salida de la ROM_OUT del SISC2020). No importará el valor que tenga WrD1 en el ultimo ciclo de ejecución de cualquiera de las nuevas instrucciones Condicionales, por lo que en estos casos se especifica como x.
4. Se añade un pequeño circuito secuencial en la UC, que denominamos FzC (Fz Control), que contiene al biestable Fz. La salida del circuito FzC es la salida Fz del biestable. Al circuito le entran, además de la señal Clk, las señales LdFz y z, para gestionar su funcionamiento: if (LdFz(c) == 1) Fz(c+1) = z(c) else Fz(c+1) = Fz(c); para cualquier ciclo c = 0, 1, 2... siendo (para cualquier señal binaria x) x(c) el valor de la señal x en el ciclo c. Los nombres de las entradas y salidas indican su conexionado en la UC.

**Examen Final IC** **17 Enero de 2020** **IC-19-20-Q1**

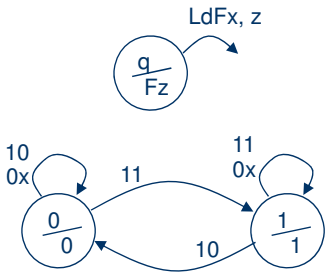
Apellidos y Nombre: ……Grupo: …… DNI: ……..

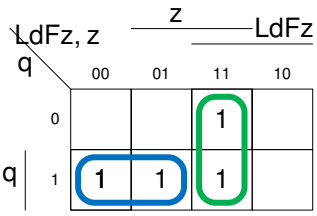
5. Se añade un pequeño circuito combinacional en la UC, denominado CIC (Conditional Instrucction Control), que genera la señal WrD, que forma parte de la palabra de control del SISC2020, en función de sus señales de entrada: Cond, Fz y WrD1.

Responded a las siguientes preguntas sobre el diseño del SISC2020:

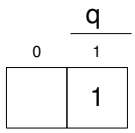
c) Dibujad el grafo de estados del circuito lógico secuencial FzC, sin olvidar la leyenda del grafo. Escribid la expresión lógica en suma de productos mínima (por Karnaugh) del estado siguiente y de la salida del circuito (Dibujando primero los dos mapas de Karnaugh con las agrupaciones de unos). Dibujad el esquema lógico del circuito secuencial implementando los circuitos combinacionales del estado siguiente y de la salida en suma de productos mínima (que son el resultado de la minimización por Karnaugth anterior) usando puertas Not, And y Or. (1 punto) (0,25 por parte)

Leyenda y Grafo | Mapa y expresión Q+ | Mapa y expresión Salida | Esquema lógico del circuito

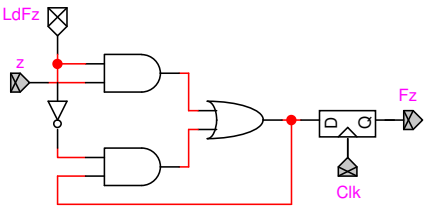



**q+ = LdFz·z + q·!LdFz**




**Fz = q**

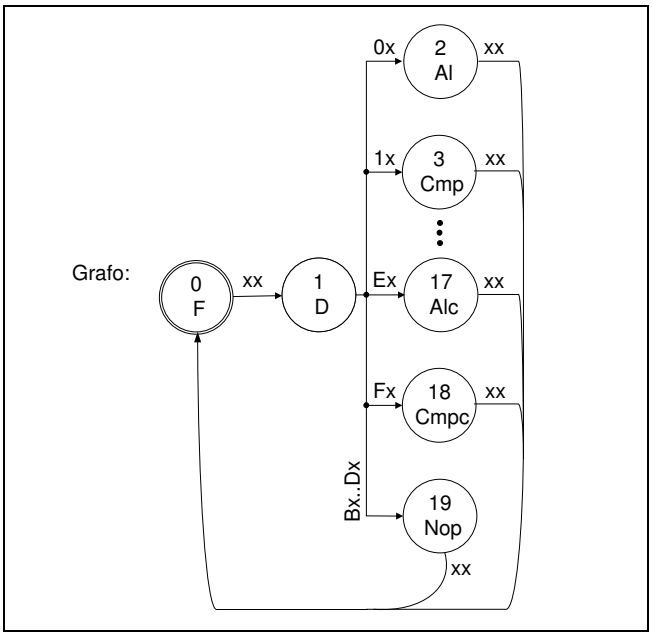



d) Escribid la expresión lógica en suma de minterms de la salida del circuito combinacional CIC (WrD) ordenando las entradas así: Cond, Fz, WrD1. (0,5 puntos) (BINARI)

WrD(Cond, Fz, WrD1) = **!Cond·!Fz·WrD1 + !Cond·Fz·WrD1 + Cond·!Fz·!WrD1 + Cond·!Fz·WrD1**

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

e) Completad el fragmento del grafo de la UC necesario para ejecutar las instrucciones AL y CMP originales, las nuevas ALc y CMPc del SISA2020 y todas las que tienen un código ilegal (que se ejecutan en Nop). Se dan todos los nodos necesarios y su codificación, pero solo un arco con su etiqueta. Dibujad todos los arcos que faltan con sus etiquetas. (1 punto) (0,25 per node erroni)




f) Completad (poniendo 0, 1 o x en cada bit) las casillas vacías de las tablas que especifican parte del contenido de la ROM_OUT para que se ejecuten correctamente todas las instrucciones SISA2020, poniendo el máximo número de x posibles. Damos la señal Cond totalmente completada y la señal WrD1 parcialmente completada, de acuerdo con la definición que hemos dado de ellas anteriormente. La dirección 0 de la ROM_OUT corresponde al estado 0 (F), la 1 al 1 (D), la 2 y 3 a las instrucciones aritméticológicas y de comparación (que han modificado su semántica) y las direcciones 17 y 18 a los dos nuevos estados Alc y Cmpc. En la segunda tabla, se representan las dos columnas de la ROM_OUT para las nuevas salidas Cond y LdFz, que se han dibujado en horizontal para ocupar menos espacio (En la fila de cabecera se indica la dirección de la ROM_OUT y el mnemotécnico de salida. Se han sombreado las casillas que no tenéis que completar por haberlo hecho en la primera tabla). (1 punto, 0,25 per columna/fila amb error)

| **@ROM** | **Cond** | **LdFz** | Bnz | Bz | WrMem | RdIn | WrOut | **WrD1** | LdIr | 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 | **Mnemo** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | x | x | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | x | x | F |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | 1 | 0 | x | x | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | x | x | D |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | x | x | x | x | 0 | 1 | 0 | 0 | 0 | 0 | x | x | 0 | x | x | x | 0 | 0 | Al |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | x | x | x | x | 0 | 1 | 0 | 0 | 0 | 1 | x | x | 0 | x | x | x | 0 | 0 | Cmp |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| 17 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | 0 | 1 | 0 | 0 | 0 | 0 | x | x | 0 | x | x | x | 0 | 0 | Alc |
| 18 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | 0 | 1 | 0 | 0 | 0 | 1 | x | x | 0 | x | x | x | 0 | 0 | Cmpc |

| | 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 Alc | 18 Cmpc | 19 Nop |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cond | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| LdFz | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |

g) Indicad la dirección o las direcciones (en binario, usando x para representar varias direcciones cuyo contenido de la ROM es el mismo) de la memoria ROM_Q+ y su contenido o sus contenidos (en Hexa) para implementar correctamente el arco del nodo/estado 1 al 18 y todos los arcos que salen del nodo 18. (0,5 puntos, 0,25 por error)

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

Arco del nodo 1 al 18: Direcciones binario = 000011111x, Contenido Hexa = 0x12
Arcos del nodo 18 al 0: Direcciones binario = 10010xxxxx, contenido Hexa = 0x00