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

# Examen 3 (temas 8, 9, 10 y 11)

Duración del examen: 2 horas. Escribid la solución de cada ejercicio en el espacio reservado para ello en el enunciado. No podéis utilizar calculadora, móvil, apuntes, etc. La solución se publicará en Atenea mañana por la tarde y las notas antes de una semana.

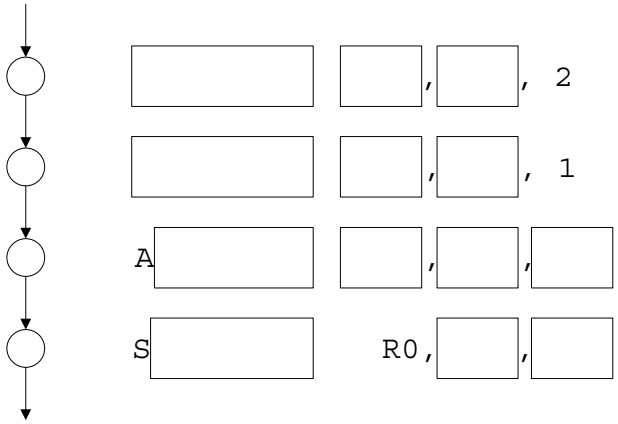
## *Ejercicio 1* (1,5 puntos)

Para ejecutar el fragmento de código `do{R2=R2*6;R0=R0-1;}while(R0>0);` donde los registros contienen números naturales, completad: (En ambos apartados se usa R7 para almacenar valores temporales y recordad que 6=4+2)

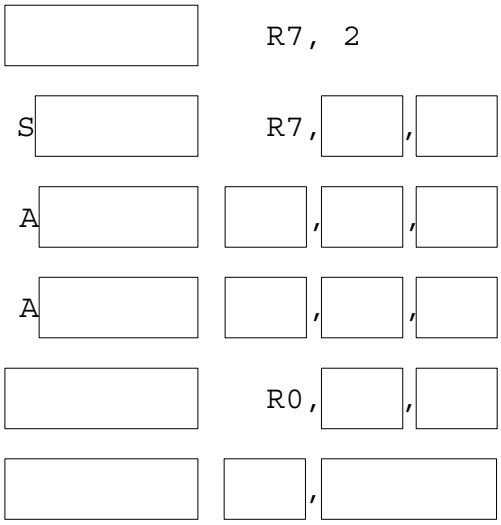
a) El grafo de estados de la UC de propósito específico para que, junto con la UPG, formen un PPE que lo ejecute. No faltan nodos, faltan arcos y etiquetas (0, 1, o x) y la palabra de control en mnemotécnicos de cada nodo.

b) El código en ensamblador SISA para ejecutarlo en el computador SISC Harvard uniciclo (UCG+UPG+$IO_{key\text{-}print}$+MEM).

**a) (0,75 puntos)**

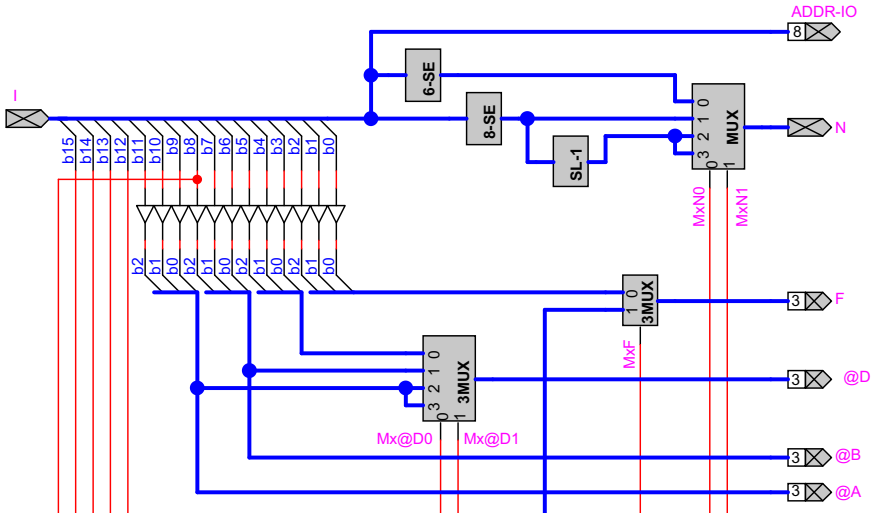



**b) (0,75 puntos)**




## *Ejercicio 2* (2 puntos)

Escribid el contenido de las filas que se muestran de la tabla de la ROM del bloque ROM-CTRL-LOGIC del computador SISC Harvard uniciclo (UCG+UPG+$IO_{key\text{-}print}$+MEM). La figura muestra parte de la lógica de control. Las señales de la palabra de control que no se muestran salen directamente de la ROM (excepto TknBr que se forma así: TknBr=Bz·z+Bnz·!z).



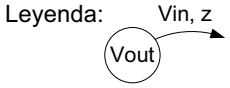

| Dirección | | | | | Contenido | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I<15> | I<14> | I<13> | I<12> | I<8> | Bnz | Bz | Wr-Mem | Rdln | WrOut | WrD | Byte | Rb/N | -/i/l/a1 | -/i/l/a0 | OP1 | OP0 | MxN1 | MxN0 | MxF | F2 | F1 | F0 | Mx@D1 | Mx@D0 | |
| 0 | 0 | 0 | 1 | x | | | | | | | | | | | | | | | | | | | | | CMP |
| 0 | 0 | 1 | 1 | x | | | | | | | | | | | | | | | | | | | | | LD |
| 0 | 1 | 1 | 0 | x | | | | | | | | | | | | | | | | | | | | | STB |
| 1 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | BZ |
| 1 | 0 | 0 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | MOVHI |
| 1 | 0 | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | OUT |
| 1 | 0 | 1 | 1 | x | | | | | | | | | | | | | | | | | | | | | (NOP) |

## *Ejercicio 3* (3 puntos)

**a) (2 puntos)** Completad el grafo de estados de la Unidad de Control Específica (UCE) de un PPE que junto a la UPG (sin espacio de E/S: con acciones `IN Rd` y `OUT Rb`) calcula **indefinidamente** cuántos de los N datos entrados son pares y cuántos impares. N siempre es mayor que 0. El PPE tiene dos señales de validación de un bit, una de entrada, Vin, y otra de salida, Vout. Si en el ciclo c la señal Vin vale 1 al ciclo siguiente en la entrada RD-IN se encuentra el número de datos a entrar, N. A los 3 ciclos (en el ciclo c+3) en RD-IN se encuentra el primer dato de la secuencia de N datos y, si N es mayor que 1, el resto de datos llegan a razón de uno cada 4 ciclos: en c+7 el segundo, en c+11 el tercero... Cuando el cálculo ha terminado, se saca por WR-OUT el número de datos pares a la vez que se pone Vout a 1. Al ciclo siguiente, se saca el número de datos impares y Vout a 0. La señal Vin se ignora desde el ciclo siguiente en el que Vin toma el valor 1 hasta el ciclo siguiente a que Vout toma el valor 1, ambos incluidos. Se calcula primero el número total de datos pares (sobre `R0`) y al final el de datos impares (sobre `R2`). En `R1` se carga N y se actualiza con el número de datos que faltan por procesar (incrementar `R0` o no). `R3` almacena el dato a procesar. No se puede usar ningún otro registro. En el grafo no faltan nodos, falta el valor de la salida Vout en el interior de algunos nodos, faltan arcos y etiquetas y falta la palabra de control en mnemotécnicos de cada nodo. Utilizar el mnemotécnico `NOP` para no hacer nada durante un ciclo si esto ayuda a la sincronización de la entrada/salida de datos. El esquema a bloques de la UPG se encuentra al final de la siguiente página.

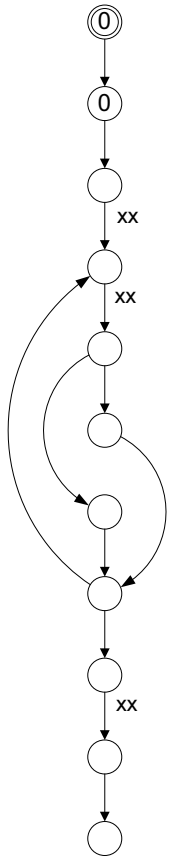
**b) (1 punto)** Completad el fragmento de código ensamblador SISA para que el SISC Harvard uniciclo (ver esquema en pag.4) calcule, en `R0`, cuántos de los N datos entrados por el teclado (con efecto lateral) son pares y, en `R2`, cuántos impares. N siempre será mayor que 0. Se hace el mismo uso de los registros que en el apartado a, pero ahora se usará `R7` para almacenar valores temporales. En este apartado no se hace el calculo de N datos indefinidamente, sino solo una vez y no se imprimen los resultados, sino que quedan en `R0` y `R2`. Para la instrucción `IN` usad las direcciones simbólicas de los puertos `KEY-STATUS` y `KEY-DATA`.

**Respuesta apartado a)**




MNEMOTÉCNICOS de la PALABRA de CONTROL

Grafo:




```
[                                ]
[                                ]
[                                ]
[                                ]
A [                              ]
ADDI  R0, R0, 1
[                                ]
S [                              ]
S [                              ]
O [                              ]
[                                ]
```

**Respuesta apartado b)**

```
[                                ]
[      ] R7, [                   ]
[      ] R7, [                   ]
[      ] R1, [                   ]
[      ] R2, [                   ]
[      ] R7, [                   ]
[      ] R7, [                   ]
[      ] R3, [                   ]
[      ] R7, [                   ]
[      ] R7, [                   ]
ADDI   R0, R0, 1
[                                ]
B [                              ]
[                                ]
```

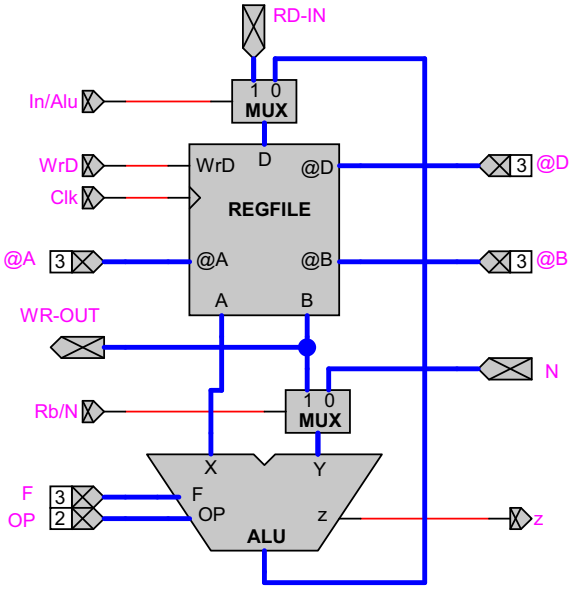
## *Ejercicio 4* (3,5 puntos)

Completad la tabla en la que cada fila es un apartado diferente que contiene 4 columnas para una misma instrucción: 1) Instrucción en ensamblador SISA, 2) Instrucción en lenguaje máquina (LM) en hexadecimal, 3) algunos bits de la palabra de control del SISC Harvard uniciclo (UCG+UPG+$IO_{key\text{-}print}$+MEM), ver pag. 4, (poned x siempre que no se sepa el valor del bit al no saber cómo se han implementado las x en la ROM de la Lógica de Control) y 4) estado del computador después de ejecutar la instrucción suponiendo que el estado antes de su ejecución es (El símbolo % denota la operación módulo): `PC=0x03DE`; `Ri=2*i` para `i=0`,... `7`; $MEM_W$`[@]=(@+2)%(2^16)` para `@=0`, 2, 4, 6... `(2^16)-2` y `IN[p]=OUT[p]=p%2` para `p=0`,... `255`. Escribid solo el contenido, en hexadecimal, de los registros (incluido el `PC`), palabras de la memoria de datos, $MEM_W$ (si se modifica un byte debe indicarse el valor de la palabra a la que pertenece) y puertos de entrada salida, `IN[p]` y `OUT[p]`, que se modifican al ejecutar cada instrucción).

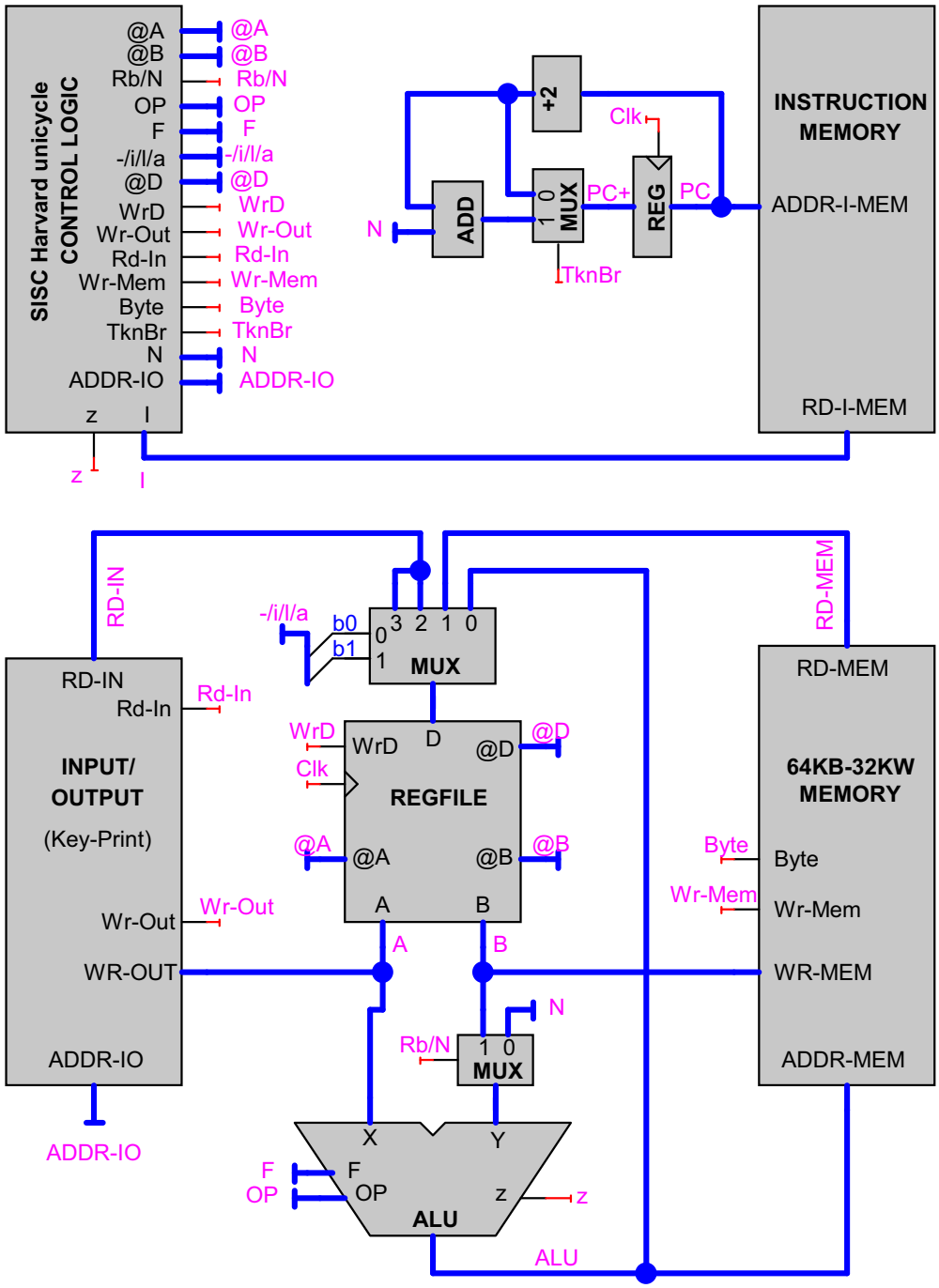
| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | Name | Mnemonic |
|---|---|---|
| 0 0 0 0 a a a b b b d d d f f f | Logic and Aritmetic Operations | `AND, OR, XOR, NOT, ADD, SUB, SHA, SHL` |
| 0 0 0 1 a a a b b b d d d f f f | Compare Signed and Unsigned | `CMPLT, CMPLE, -,CMPEQ, CMPLTU,CMPLEU, -, -` |
| 0 0 1 0 a a a d d d n n n n n n | Add Immediate | `ADDI` |
| 0 0 1 1 a a a d d d n n n n n n | Load | `LD` |
| 0 1 0 0 a a a b b b n n n n n n | Store | `ST` |
| 0 1 0 1 a a a d d d n n n n n n | Load Byte | `LDB` |
| 0 1 1 0 a a a b b b n n n n n n | Store Byte | `STB` |
| 0 1 1 1 | | Branch future extension |
| 1 0 0 0 a a a 0 n n n n n n n n | Branch on Zero | `BZ` |
| 1 0 0 0 a a a 1 n n n n n n n n | Branch on Not Zero | `BNZ` |
| 1 0 0 1 d d d 0 n n n n n n n n | Move Immediate | `MOVI` |
| 1 0 0 1 a a a d d d 1 n n n n n n n n | Move Immediate High | `MOVHI` |
| 1 0 1 0 d d d 0 n n n n n n n n | Input | `IN` |
| 1 0 1 0 a a a 1 n n n n n n n n | Output | `OUT` |
| 1 0 1 1 x x x x x x x x x x x x | | Future extensions |
| 1 1 x x | | |

| | 1) Ensamblador | 2) LM (Hexa) | 3) Bits Pal. Control -/i/l/a | | WrD | Byte | TknBr | N (hexa) | 4) Estado después de su ejecución |
|---|---|---|---|---|---|---|---|---|---|
| **a)** | `STB -5(R4), R0` | | | | | | | | |
| **b)** | `BZ R0, -7` | | | | | | | | |
| **c)** | | `596E` | | | | | | | |
| **d)** | | `9DB6` | | | | | | | |

## Esquema a bloques de la UPG sin espacio de Entrada/salida



## Estructura a bloques del SISC Harvard uniciclo (UCG+UPG+IO$_{key-print}$+MEM)




### Funcionalidad de la ALU

| F | | | OP | | | |
|---|---|---|---|---|---|---|
| $b_2$ | $b_1$ | $b_0$ | 11 | 10 | 01 | 00 |
| 0 | 0 | 0 | --- | X | CMPLT (X, Y) | AND (X, Y) |
| 0 | 0 | 1 | --- | Y | CMPLE (X, Y) | OR (X, Y) |
| 0 | 1 | 0 | --- | MOVHI(X, Y) | --- | XOR(X, Y) |
| 0 | 1 | 1 | --- | --- | CMPEQ (X, Y) | NOT (X) |
| 1 | 0 | 0 | --- | --- | CMPLTU (X, Y) | ADD (X, Y) |
| 1 | 0 | 1 | --- | --- | CMPLEU (X, Y) | SUB (X, Y) |
| 1 | 1 | 0 | --- | --- | --- | SHA(X, Y) |
| 1 | 1 | 1 | --- | --- | --- | SHL(X, Y) |