Trabajo Práctico Nº 8
MICROARQUITECTURA

1. Diseñar una unidad aritmético-lógica capaz de resolver las operaciones AND, OR, NOR y XOR en operadores de 8 bits. Posee una entrada C de dos bits que permite elegir la operación: según indica la tabla adjunta. Realice su diseño en base a 8 módulos ALU de 1 bit y un multiplexor.

C	Operación
00	AND
01	OR
10	NOR
11	XOR

2 Basándose en la microarquitectura para ARC propuesta en el libro de Murdocca-Heuring, discutir las funciones e vínculos entre los siguientes bloques: “Control Branch Logic”, Control Store Address incrementer”, “Control Store Address Multiplexer”.

3. Escriba la forma binaria de las siguientes microinstrucciones, presentando para cada posición de memoria en el Control Store el contenido de los campos a cargar en el registro de microinstrucciones. Complete con 0´s los campos que no sean necesarios.

60: R[temp0] = NOR(R[0],R[temp0]); IF Z THEN GOTO 64;
61: R[rd] = INC(R[rs1]);

3. En la tabla siguiente cada una de las 3 palabras de 41 bits puede ser interpretada como una microinstruccion. Obtener la versión con mnemonicos de micro-assembler

A	AMUX	B	BMUX	C	CMUX	URW	ALU	COND	JUMP ADDR
10010	1	0000000	1	00001	0	001	1100	000	00000000000
00000	1	1000010	1	00001	0	001	1000	110	11100000001
00000	1	0000000	1	10000	1	001	1000	101	11100010010

4. Un microprograma requiere de un salto condicional basado en el bit 13 del registro de instrucciones. Se proponen dos soluciones, los respectivos segmentos de microcódigo se presentan a continuación:

(a)

1792: R[temp0] = ADD(R[rs1],R[rs2]);
IF R[IR[13]] THEN GOTO 1794;

(b)

1792: IF R[IR[13]] THEN GOTO 1794;
R[temp0] = ADD(R[rs1],R[rs2]);

Indique, justificando, si tiene algún efecto el cambio de orden de las instrucciones de micro-Assembler.
Indique cuál podría ser la razón de examinar el contenido de ese bit en particular.

5. El siguiente microcódigo corresponde a la instrucción *call* del Assembler ARC.

1280: R[15] = AND(R[pc],R[pc]);
1281: R[temp0] = ADD(R[ir],R[ir]);
1282: R[temp0] = ADD(R[temp0],R[temp0]);
1283: R[pc] = ADD(R[pc],R[temp0]);
GOTO 0

Reescribirlo de modo que se usen sólo 3 líneas de microcódigo en vez de 4. Utilizar la operación LSHIF2.

6. La instrucción ARC *subcc*, que realiza una resta, es implementada por medio del siguiente microprograma

1584: R[temp0] = SEXT13(R[ir]); / Extrae operando
IF IR[13] THEN GOTO 1586; / Es direcc. Inmediato?
1585: R[temp0] = OR(R[0],R[rs2]); / Extrae operando
1586: R[temp0] = NOR(R[temp0], R[0]); / Complemento a 1
1587: R[temp0] = INC(R[temp0]); / Complemento a 2
1588: R[rd] = ADDCC(R[rs1],R[temp0]);
GOTO 2047;

Discutir el funcionamiento de esta subrutina y además indicar:
(a) Porqué se inicia en la dirección 1584
(b) Cuántas microinstrucciones son ejecutadas para interpretar *subcc*
(c) En qué orden son ejecutadas cada una de las líneas de microcódigo, empezando por la microinstrucción en la dirección 0.

7. Traduzca a binario el segmento de microprograma del ejercicio anterior

8. El registro *%r0* puede ser diseñado por medio de buffers tri-state exclusivamente. Proponga el diagrama cicuital correspondiente.

9. Proponga un patrón de 0´s y 1´s para asignar al campo *C* del registro de micorinstrucciones si se necesita que ninguno de los registros del archivo de registros cambie su contenido.

10 Modifique el microcódigo de la implementación ARC de modo que incluya una nueva instrucción: *xorcc*. Esta realiza un OR-exclusivo de los operandos modificando el *%psr*. Esta instrucción sigue el formato aritmético de ARC y su campo op3 es 010011.

11 El microcódigo implementado propuesto en el libro de Murdocca-Heuring incluye el siguiente segmento:

2047: R[pc] = INCPC(R[pc]); GOTO 0; / Increment %pc and start over

Se producirían cambios en el microprograma en caso de que "Goto 0" fuese eliminado?

12 Una unidad aritmético-lógica es implementada por medio de "look-up tables" (LUT) con operandos de 1 bit. Entre las operaciones que esta ALU realiza está INC(A). Indique las estructuras de la LUT_0 y LUT_i con $i > 0$.

13. En algunas arquitecturas existe un hardware especial para actualizar el registro Program Counter (PC) teniendo en cuenta el hecho de que sus dos bits menos significativos están siempre en 0. Sobre la base de una arquitectura que no incluye esta solución de hardware, el siguiente microcódigo (libro de Murdocca-Heuring) incluye intencionalmente un error en la posición 12 en la forma en que el PC es actualizado. El objetivo del código presentado es implementar las instrucciones ARC de salto condicional. En ARC los saltos son medidos con desplazamientos en palabras de 4 bytes.
Identifique el error y explique como solucionarlo.

```
      / Branch instructions: ba, be, bcs, bvs, bneg
1088: GOTO 2;                          / Decoding tree for branches
   2: R[temp0] ← LSHIFT10(R[ir]);      / Sign extend the 22 LSB's of %temp0
   3: R[temp0] ← RSHIFT5(R[temp0]);    / by shifting left 10 bits, then right 10
   4: R[temp0] ← RSHIFT5(R[temp0]);    / bits. RSHIFT5 does sign extension.
   5: R[ir] ← RSHIFT5(R[ir]);          / Move COND field to IR[13] by
   6: R[ir] ← RSHIFT5(R[ir]);          / applying RSHIFT5 three times. (The
   7: R[ir] ← RSHIFT5(R[ir]);          / sign extension is inconsequential.)
   8: IF R[IR[13]] THEN GOTO 12;       / Is it ba?
      R[ir] ← ADD(R[ir],R[ir]);
   9: IF R[IR[13]] THEN GOTO 13;       / Is it not be?
      R[ir] ← ADD(R[ir],R[ir]);
  10: IF Z THEN GOTO 12;               / Execute be
      R[ir] ← ADD(R[ir],R[ir]);
  11: GOTO 2047;                       / Branch for be not taken
  12: R[pc] ← ADD(R[pc],R[temp0]);     / Branch is taken
      GOTO 0;
  13: IF R[IR[13]] THEN GOTO 16;       / Is it bcs?
      R[ir] ← ADD(R[ir],R[ir]);
  14: IF C THEN GOTO 12;               / Execute bcs
  15: GOTO 2047;                       / Branch for bcs not taken
  16: IF R[IR[13]] THEN GOTO 19;       / Is it bvs?
  17: IF N THEN GOTO 12;               / Execute bneg
  18: GOTO 2047;                       / Branch for bneg  not taken
  19: IF V THEN GOTO 12;               / Execute bvs
  20: GOTO 2047;                       / Branch for bvs not taken
2047: R[pc] ← INCPC(R[pc]); GOTO 0;    / Increment %pc and start over
```