1. Considere la ALU y registros que se ilustran en la Figura 1.

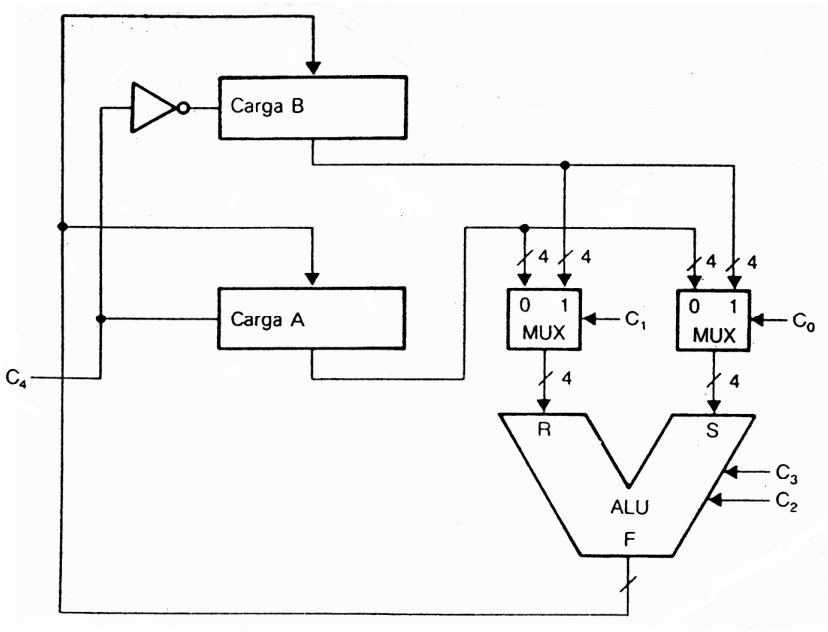


Figura 1. ALU y registros del problema 1.

La interpretación de los distintos puntos de control se resume en la Tabla 1.

Tabla 1. Interpretación de las señales de control del problema 1.

C_3C_2	F
0 0	R + S
0 1	R − S
1 0	R AND S
1 1	R XOR S

C_1C_0	Entrada R	Entrada S
0 0	A	A
0 1	A	B
1 0	B	A
1 1	B	B

C_4	Acción
0	B ← F
1	A ← F

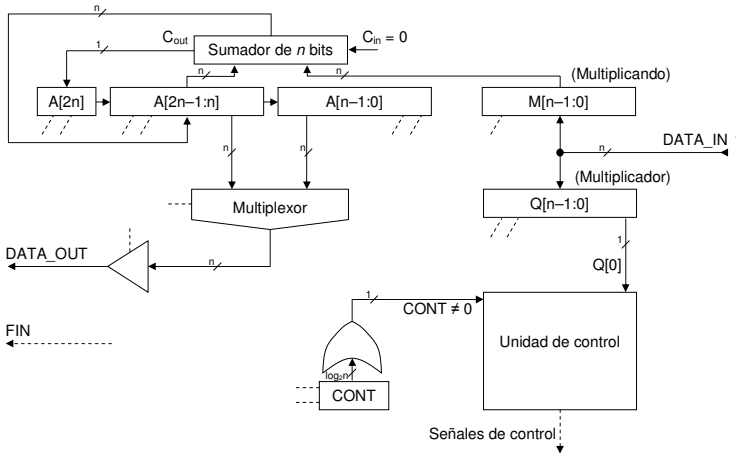
Cada palabra de control debe especificarse de acuerdo con el formato $c_4\ c_3\ c_2\ c_1\ c_0$.
Por ejemplo:

```
C4  C3  C2  C1  C0
1   0   0   0   1        ; A ← A + B
```

a) ¿Cómo se puede poner el registro A a cero? Sugiera al menos dos formas de hacerlo. No se dispone de una entrada directa para poner a cero el registro.
b) Sugiera una secuencia de control que intercambie los contenidos de los registros A y B.

2. La Figura 2 muestra la estructura y el algoritmo de un circuito multiplicador para números de *n* bits (*n* potencia de 2) sin signo. Los buses DATA_IN y DATA_OUT permiten introducir los datos y obtener los resultados, respectivamente, y CONT es un contador que se decrementa en cada paso de la multiplicación.

La señal FIN indica a los circuitos externos al multiplicador que en los dos ciclos de reloj posteriores al actual se enviará el resultado.




```
A = 0;
CONT = 0;
Q = DATA_IN;
M = DATA_IN;
TEST:  if (Q[0] == 0) goto DESP;
       A[2n–1:n] = A[2n–1:n] + M[n–1:0]; A[2n] = Cout;
DESP:  A[n–1:0] = A[n:1];
       A[2n–1:n] = A[2n:n+1];
       A[2n] = 0;
       Q[n–2:0] =  Q[n–1:1];
       CONT = CONT–1;
       if (CONT != 0) goto TEST;
       FIN = 1;
       DATA_OUT = A[2n–1:n];
       DATA_OUT = A[n–1:0];
```

Figura 2. Estructura y algoritmo del multiplicador de números binarios sin signo del problema 2.

a) Asigne en la Figura 2 un nombre a cada señal de control (líneas discontinuas), teniendo en cuenta que una misma señal (un mismo nombre) puede actuar sobre varios elementos.
b) Diseñe una unidad de control microprogramada que efectúe la multiplicación de números binarios sin signo en ese circuito.
c) Escriba el microprograma en binario en una tabla donde se indique para cada dirección de la memoria de control, los distintos campos de la microinstrucción. Debe procurarse que la multiplicación consuma el

```
{Parte declarativa}
register A[2n – 2:0], M[n – 2:0], Q[n – 2:0], CONT[p – 1:0], XS, YS, PS;
terminal DATA-IN[n – 1:0], DATA-OUT[n – 1:0];
{Parte activa}
{Esta es una versión modificada del algoritmo de multiplicación binaria de la figura      , que trabaja
 con números CD de n bits.}
while RELOJ = ACTIVO do
  if INICIO = 1 then
  begin    {Cargar los operandos de entrada y convertirlos a forma positiva, si es necesario.
  XS : = DATA-IN[n – 1], Q : = DATA-IN[n – 2:0], A : = 0, CONT : = 0;
  YS : = DATA-IN[n – 1], M : = DATA-IN[n – 2:0];
  PS : = XS ⊕ YS, if XS = 1 then Q : = Q̄ + 1;
                  if YS = 1 then M : = M̄ + 1;
  while CONT < 2n + 2 do
    begin    {Etapa principal de la multiplicación}
    if Q[0] = 1 then A[2n – 2:n – 1] : = A[2n – 3:n – 1] + M;
    A[2n – 2] : = 0, A[2n – 3:0] : =  A[2n – 2:1], Q[n – 2] : = 0, Q[n – 3:0] : =  Q[n – 2:1]
    end;
  if PS = 1 then
    begin    {Negar el bit (2n – 2)-del producto en el acumulador.}
    A[2n – 2].A[n – 2:0] : = Ā[n – 2:0] + 1;    {Almacenar en A[2n – 2] el bit de acarreo de
                                                  salida.}
    A[2n – 2].A[2n – 3:n – 1] : = Ā[2n – 3:n – 1] + A[2n – 2]
    end;
  DATA-OUT : = PS.PS.A[2n – 3:n – 2], PARAR : = 1;
  DATA-OUT : = A[n – 1:0]
  end.
```

Figura 3. Algoritmo del multiplicador en complemento a dos del problema 2.

menor número de ciclos de reloj posible.

d) Rediseñe el circuito para realizar el algoritmo de multiplicación en complemento a dos que se da en la Figura 3, enumerando las señales de control nuevas que se necesitan y sus funciones, y modificando convenientemente la unidad de control microprogramada. Escribir el microprograma necesario.

3. La Figura 4 representa la CPU de una máquina de pila para la que se desea diseñar una unidad de control microprogramada con control residual para el control de las operaciones de la ALU.

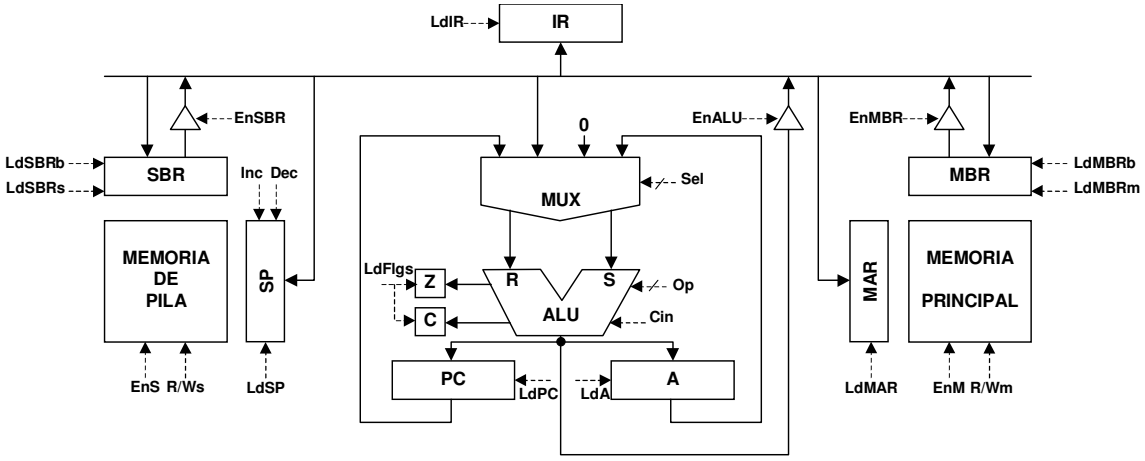


Figura 4. CPU de la máquina de pila del problema 3.

Esta CPU consta de los siguientes elementos:

- Una ALU capaz de realizar las operaciones de SUMA, RESTA, AND, OR, INVERSIÓN DE SIGNO y NEGACIÓN
- Un registro contador de programa (PC) y un registro acumulador (A).
- Un registro de intercambio de datos con memoria principal (MBR).
- Un registro de direcciones de memoria principal (MAR).
- Un registro de instrucción (IR).
- Una memoria donde se almacenan datos e instrucciones.
- Un registro SP que apunta a la cima de la pila, capaz de ser incrementado, decrementado y cargado desde el bus.
- Un registro SBR que contiene la palabra leída de la pila o la palabra que se quiere escribir en ella.
- Un bus de conexiones entre los componentes de la CPU.

La CPU debe ser capaz de ejecutar el conjunto de instrucciones siguiente:

- de una palabra:
  ADD, SUB, AND, OR, NEG, NOT sobre datos de la pila
- de dos palabras:
  PUSH X, POP X
  LDSP X; carga SP con el contenido de X
  donde X es una dirección de memoria principal

a) Diseñe el formato de microinstrucción para controlar esta CPU, codificando campos de señales de control (microprogramación vertical), teniendo en cuenta que se utilizara control residual para las operaciones de la ALU.

b) Diseñe la unidad de control microprogramada.

c) Escriba el microprograma indicando las microoperaciones que se realizan en cada microinstrucción.

4. Se está diseñando un computador y se quiere estudiar la posibilidad de dotarle de una unidad de control microprogramada basada en memoria de control horizontal, o bien, en memoria de control vertical. El secuenciador de microprograma que se va a utilizar proporciona una dirección de 12 bits.

Los respectivos formatos de microinstrucciones son:

- Memoria de control horizontal:

A1	A2	A3	B1	B2	B3	C1	C2	C3	C4	D1	D2	D3	E1	...	E8

donde Ai, Bj, Ck, Dm y En son señales de control.

- Memoria de control vertical:

sel-A	x	sel-C	sel-B o sel-D	sel-E
campo1	campo2	campo3	campo4	campo5

donde el campo4 se decodifica indirectamente (como sel-B o sel-D) a través del campo2.
a) Compare el tamaño de la memoria de control para ambas alternativas.
b) Compare el tiempo de activación de las señales de control para los dos tipos de memoria de control.

5. Una microarquitectura tiene microinstrucciones de tamaño igual a 100 bits. Los microprogramas almacenados requieren 1800 posiciones de la memoria de control. El objetivo es reducir la cantidad de memoria de control utilizando una de las tres siguientes técnicas:
a) Una posibilidad es usar control residual, teniendo en cuenta que existen 50 microprogramas, de 9 microinstrucciones cada uno, que se diferencian entre sí sólo en el contenido de un mismo campo de la microinstrucción número 5.
b) Otra técnica es usar una nanomemoria, teniendo en cuenta que hay 300 microinstrucciones distintas.
Obtenga el tamaño de la memoria de control en bits para cada una de las dos técnicas y explique cómo se obtiene dicho resultado en cada caso.

6. Una unidad de control microprogramada convencional incluye 2048 palabras de 117 bits. Existen 512 microinstrucciones diferentes. Calcule el ahorro de memoria que se conseguiría empleando una nanomemoria. Calcule los tamaños de la memoria de control original y de la nanomemoria y micromemoria.

7. Un computador tiene una memoria de control de 16K palabras de 250 bits cada una, emplea códigos de operación de 8 bits y dispone de 4 biestables de condición. Se quiere rediseñar la memoria de control utilizando nanoprogramación. Para ello se ha hecho un estudio del número de microinstrucciones diferentes, obteniéndose un total de 447.

Realice el esquema completo de dicha unidad de control, teniendo en cuenta que se deben poder realizar saltos, tanto condicionales como incondicionales, a nivel de microprograma.

8. La Figura 5 corresponde a la unidad de control microprogramada de un pequeño computador que consta de los siguientes elementos:

- Tres registros de propósito general.
- Un registro contador de programa.
- Una ALU que puede realizar 4 operaciones.
- Un registro de estado de la ALU de un bit que contiene el bit de acarreo.

Cada registro está controlado por dos señales: una de activación y otra de lectura o escritura del registro. El registro contador tiene la capacidad de autoincremento.

Establezca el formato de las microinstrucciones en un diseño horizontal, incluyendo el campo de secuenciamiento y la lógica de secuenciamiento.

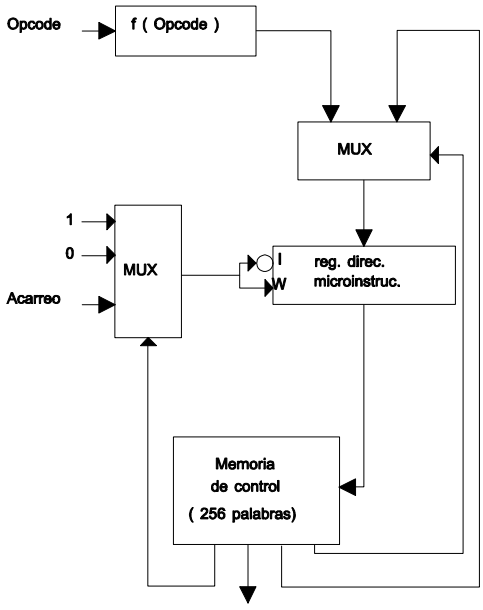


Figura 5. **U.C. microprogramada del problema 8.**

9. La unidad de procesamiento de la Figura 6 es capaz de realizar, en un ciclo de reloj, cada una de las operaciones siguientes:

ACC := F(ACC,Rj), Ri:=A, Ri:=B, Ri:=ACC, C:=Rj

donde Rj puede ser uno de los registros:

M0, M1, ..., M7, A y B,

y Ri uno de los registros:

M0, M1, ..., M7 y C.

Los registros son de 32 bits. La unidad aritmético lógica, controlada por la señal OPER, puede ejecutar las siguientes operaciones:

$$z := x \cdot y$$
$$z := x + y$$
$$z := x + 1$$
$$z := x - 1$$
$$z := 1$$
$$z := 0$$
$$z := x$$
$$z := \text{SHIFT } (x)$$

Genere los programas de control que permiten realizar las operaciones siguientes:

$$C := A + B$$
$$C := A \cdot B$$
$$C := B^A$$

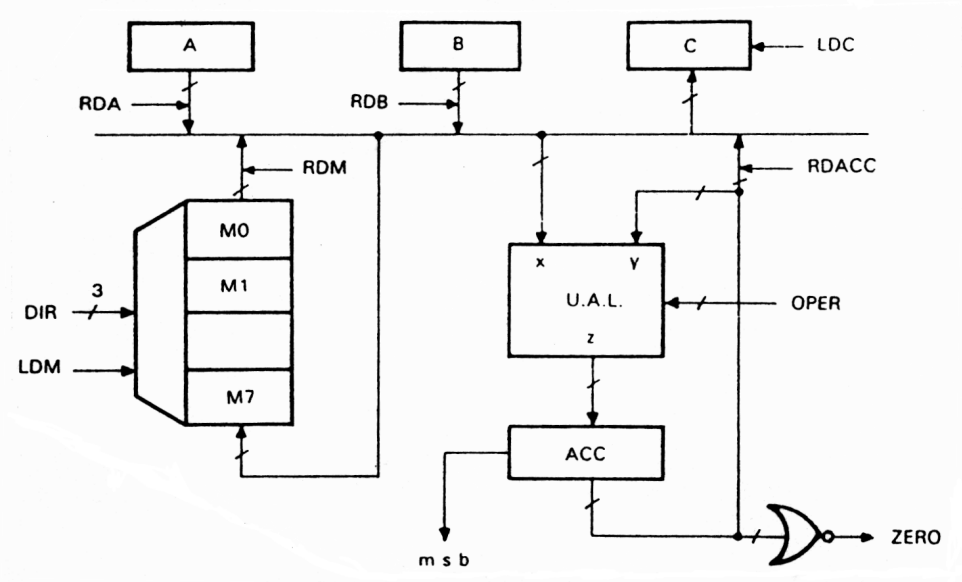


Figura 6. Unidad de procesamiento del problema 9.

10. La Figura 7 corresponde al camino de datos de un ordenador de 8 bits con arquitectura basada en acumulador que usa el modelo *little-endian*. La arquitectura está compuesta, entre otros, por:

- Una memoria principal accesible por bytes.
- Una ALU de 8 bits capaz de realizar las siguientes operaciones:
  - *salida_alu* = *entrada_izqda*
  - *salida_alu* = NOT *entrada_drcha*
  - *salida_alu* = *entrada_drcha* SHR 1
  - *salida_alu* = *entrada_drcha* AND *entrada_izqda*
  - *salida_alu* = *entrada_drcha* + *entrada_izqda*
  - *salida_alu* = *entrada_drcha* - *entrada_izqda*
- Un registro de dirección de memoria (MAR) de 20 bits.
- Un registro de datos de memoria (MDR) de 8 bits.
- Un registro de instrucción (IR) de 4 bits, que se carga con MDR[7:4] y almacena el código de operación de la instrucción en ejecución.
- Un registro de indicadores de 4 bits (C = Acarreo, O = Desbordamiento, S = Signo, Z = Cero).
- Un registro acumulador (A) de 8 bits.
- Un registro contador de programa (PC) de 20 bits.

• Un registro puntero de pila (SP) de 20 bits que se inicializa a 111...111 automáticamente al encenderse o reiniciar el ordenador. La pila está implementada en memoria principal y crece hacia direcciones inferiores.
• Un registro auxiliar (AUX) de 20 bits. AUX[19:12] se puede cargar con MDR[7:0], AUX[11:4] con MDR[7:0], y AUX[3:0] con MDR[3:0].
• Un sumador de 20 bits.

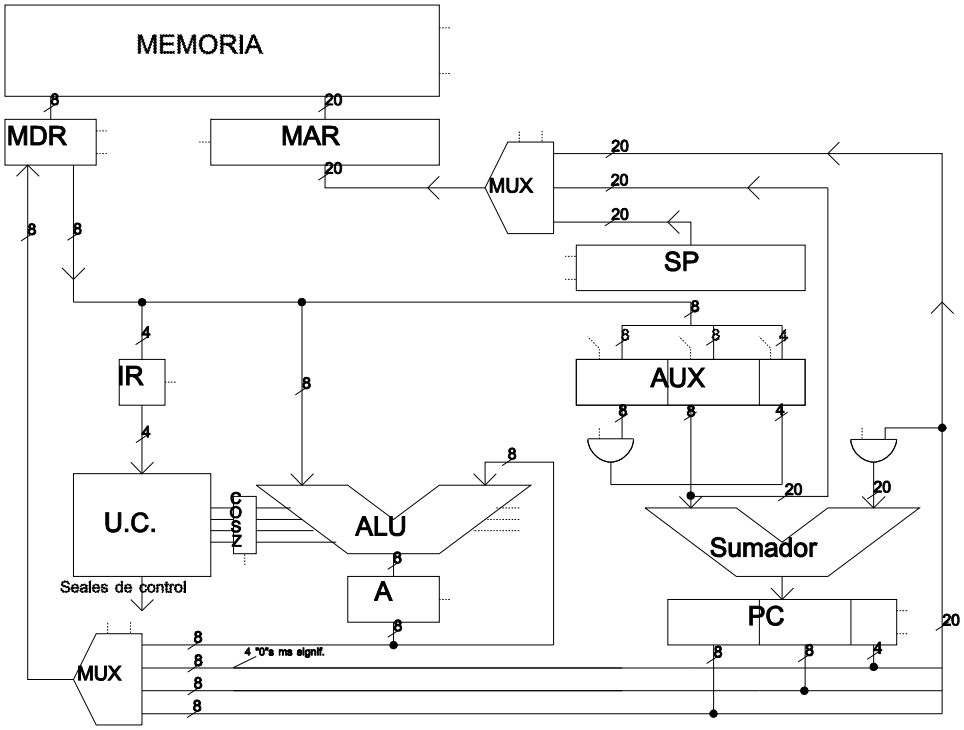


Figura 7. Camino de datos del ordenador del problema 10.

Tabla 2. Instrucciones máquina del ordenador del problema 10.

CODOP	Instrucción	Formato
0000	SHR A	1 byte: cccc---- (Los 4 bits menos significativos no se utilizan)
0001	PUSH A	
0010	POP A	
0011	RET	
0100	JC X	2 bytes: ccccxxxx xxxxxxxx (X es un desplazamiento de 12 bits relativo a PC)
0101	JO X	
0110	JS X	
0111	JZ X	
1000	LOAD A,M[X]	3 bytes: ccccxxxx xxxxxxxx xxxxxxxx (X es una dirección absoluta de memoria principal, de 20 bits) (ADD, SUB, NAND y CMP afectan a los indicadores de estado)
1001	STORE M[X],A	
1010	ADD A,M[X]	
1011	SUB A,M[X]	
1100	NAND A,M[X]	
1101	CMP A,M[X]	
1110	JMP X	
1111	CALL X	

• Una unidad de control (U.C.).

Este ordenador debe ser capaz de ejecutar las instrucciones de la Tabla 2, que pueden ocupar uno, dos o tres bytes.

a) ¿Cuál es el tamaño de memoria principal que se puede direccionar?

b) Asigne una función a cada una de las señales de control (indicadas en la figura mediante una línea discontinua).

c) Diseñe la unidad de control microprogramada con un formato de microinstrucción que incluya los tres campos siguientes:

- Tipo de salto (incrementar contador de microprograma, cargarlo con f(IR), saltar si se cumple la condición indicada por IR[1:0], saltar si IR[2] = 1, saltar si IR[3] = 1 y saltar incondicionalmente).
- Señales de control.
- Dirección de salto (microinstrucción siguiente).

d) Escriba en lenguaje de alto nivel (como el visto para la máquina de Tanenbaum) el microprograma completo de la unidad de control.

e) ¿Cuál es el tamaño de la memoria de control? ¿Cuál sería utilizando nanoprogramación?

11. En la Figura 8 se muestra el esquema de bloques del camino de datos de un ordenador cuya unidad de control microprogramada estamos diseñando.

Las señales de control que dirigen su funcionamiento son las que se relacionan a continuación:

ICM: Inicio del ciclo de memoria.

R/W: Señal de lectura(1)/escritura(0).

LDRA: Carga del registro de dirección.

LDRD: Carga del registro de datos.

SALMEM: Pone la salida de la memoria en el bus.

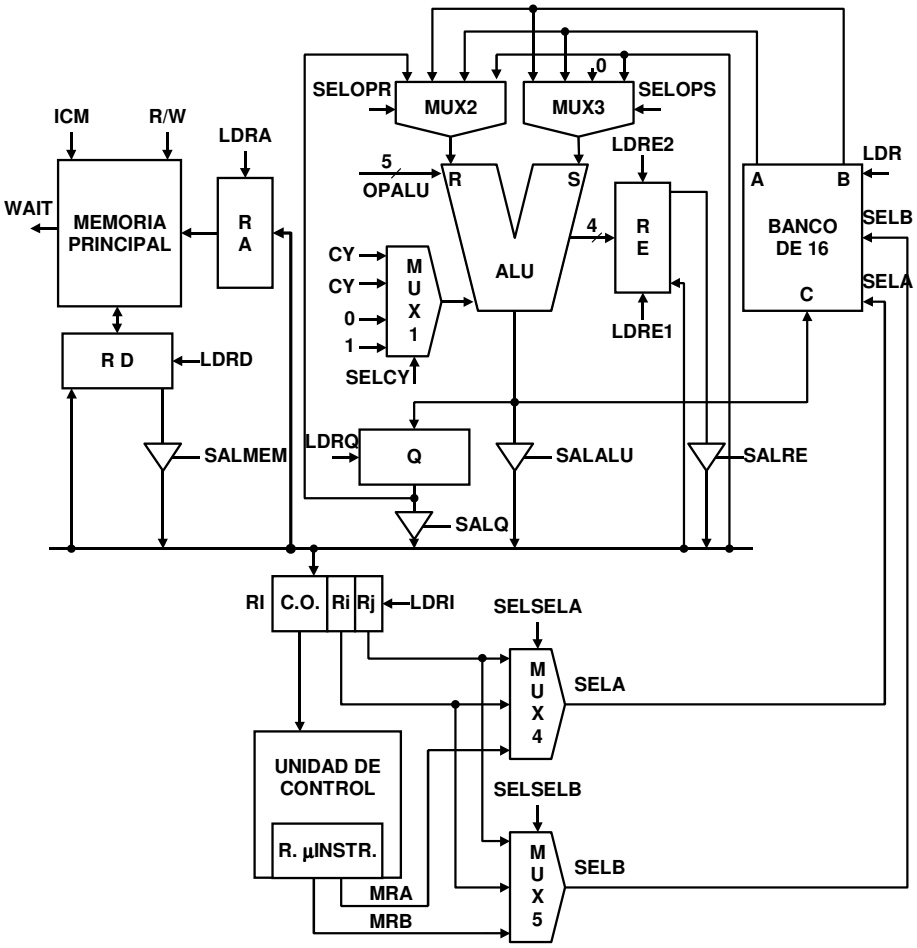


Figura 8. Esquema de bloques del camino de datos del problema 11.

SELOPR: Selección de la entrada R de la ALU.
SELOPS: Selección de la entrada S de la ALU.
OPALU: Selección de la operación a realizar por la ALU.
SELCY: Selección del acarreo de entrada a la ALU.
LDRQ: Carga del registro Q.
SALQ: Pone la salida del registro Q en el bus.
SALALU: Pone la salida de la ALU en el bus.
LDRE1: Carga del registro de estado desde el bus.
LDRE2: Carga del registro de estado con los indicadores Z, C, P y S, según el resultado de la operación realizada por la ALU.
SALRE: Pone la salida del registro de estado en el bus.
SELA: Selecciona un registro del banco por la puerta A.
SELB: Selecciona un registro del banco por la puerta B.
LDR: Carga del valor que aparece a la entrada C del banco en el registro seleccionado por la puerta A.
LDRI: Carga del registro de instrucción desde el bus.
SELSELA: Selección de campo de direccionamiento SELA del banco de registros.
SELSELB: Selección de campo de direccionamiento SELA del banco de registros.
MRA y MRB son dos campos de 4 bits de la microinstrucción que permiten seleccionar registros desde microinstrucción.
WAIT: Esta señal indica (=1) a la unidad de control que la operación con memoria aún no ha finalizado.

El computador debe disponer, entre otras, de la instrucción de una palabra ADD [++Ri],[Rj] cuya acción consiste en incrementar en 1 el registro Ri y sumar el valor almacenado en la posición de memoria Ri (valor ya incrementado de Ri) al valor almacenado en la posición Rj, almacenando el resultado en esta última posición de memoria.

Describa en lenguaje natural el microprograma completo correspondiente a esta instrucción indicando las señales de control que se deben activar en cada microinstrucción.

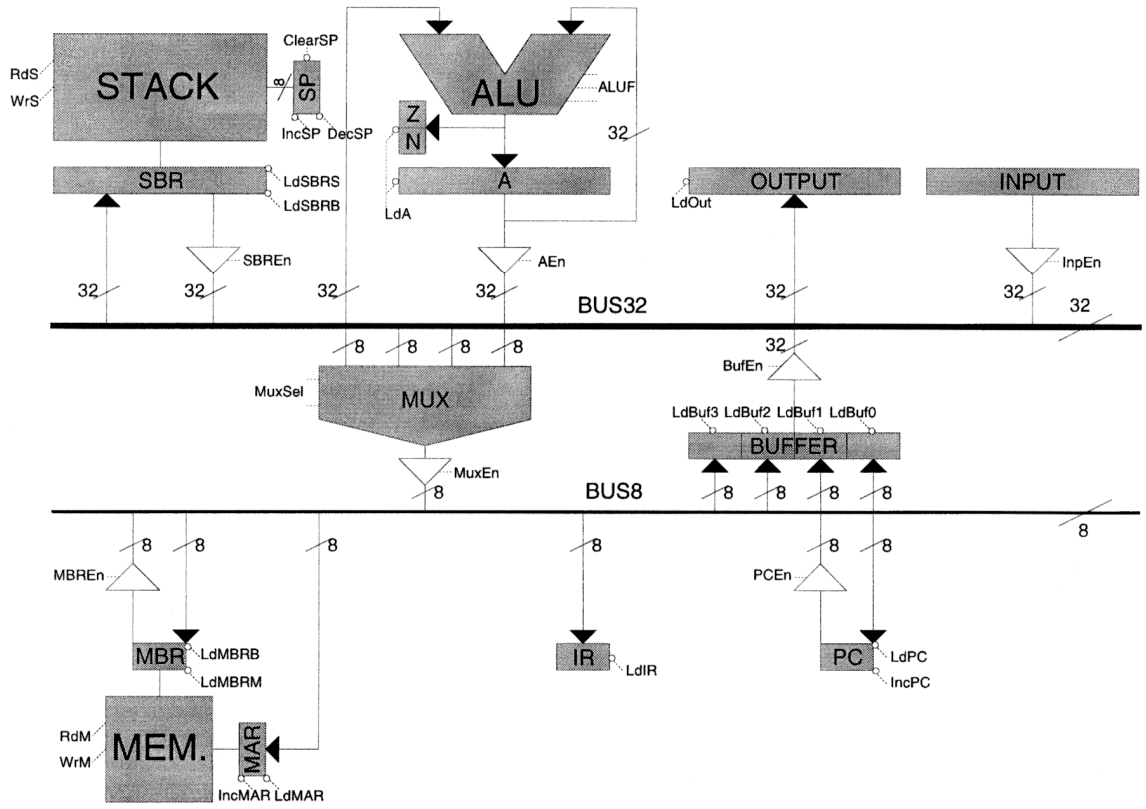


Figura 9. Camino de datos de la calculadora del problema 12.

12. La calculadora programable cuyo camino de datos se muestra en la Figura 9 tiene una memoria de 256 bytes para datos de 32 bits e instrucciones, y una pila de 256 palabras de 32 bits para almacenar resultados intermedios. Dispone de las siguientes instrucciones de dos bytes:
PUSH X, **POP X**, **IN X**, **OUT X**, **JMP X**, **JZ X** y **JN X**
(donde X es una dirección de memoria) y de las siguientes instrucciones de un byte:
ADD, **SUB**, **MUL**, **DIV** y **SQRT.**

a) Diseñe el formato de microinstrucción horizontal y la unidad de control microprogramada teniendo en cuenta las siguientes restricciones:

- El contador de microprograma no tendrá capacidad de autoincremento pero sí se dispondrá de un circuito combinacional incrementador.
- Existirán cuatro tipos de salto, codificados según la Tabla 3.
- El campo de dirección siguiente (DIR) estará solapado con el campo de señales de control, de manera que una microinstrucción será o bien de salto, o bien de control sobre el camino de datos.
- Las instrucciones JZ X y JN X tienen la misma codificación excepto el bit 0.

b) Escriba en un lenguaje simbólico cada una de las microinstrucciones de la fase de captación del código de operación, y de la fase de captación de la dirección X y ejecución correspondientes a las instrucciones **PUSH X**, **JN X** y **ADD**. Tenga en cuenta que en la primera mitad de cada ciclo de reloj hay tiempo suficiente para realizar las operaciones de lectura, escritura, acceso a buses, y operaciones con la ALU, y que todas las señales que actúan sobre los registros se activan en la segunda mitad del ciclo de reloj, cuando las entradas a esos registros son estables.

Tabla 3. Tipos de salto para la U.C. de la calculadora del problema 12.

Codificación	Tipo de salto	
00	μPC←μPC+1	
01	μPC←μPC+1	si COND=0
	μPC←DIR	si COND=1
10	μPC←DIR	
11	μPC←f(IR)	

(COND = Z ó N según sea el bit 0 de IR)

13. Dado el repertorio de instrucciones en ensamblador siguientes:

```
LOAD A,d      ; Cargar en el acumulador el contenido de la posición de memoria d
LOAD A,[d]    ; Cargar en el acumulador el contenido de la posición de memoria
              almacenada en la posición de memoria d
SUB A,d       ; Restar del acumulador el contenido de la posición de memoria d
SUB A,[d]     ; Restar del acumulador el contenido de la posición de memoria
              almacenada en la posición de memoria d
STORE d,A     ; Almacenar en la posición de memoria d el contenido del acumulador
STORE [d],A   ; Almacenar en la posición de memoria almacenada en la posición de
              memoria d el contenido del acumulador
JZ d          ; Saltar a la posición de memoria d si el resultado de la última operación
              aritmética es cero
JZ [d]        ; Saltar a la posición de memoria almacenada en la posición de memoria d
              si el resultado de la última operación aritmética es cero
JC d          ; Saltar a la posición de memoria d si el resultado de la última operación
              aritmética provocó acarreo
JC [d]        ; Saltar a la posición de memoria almacenada en la posición de memoria d
              si el resultado de la última operación aritmética provocó acarreo
JMP d         ; Saltar a la posición de memoria d
JMP [d] ; Saltar a la posición de memoria almacenada en la posición de memoria d
```

~~***a)*** Escriba en ensamblador el programa para realizar la ordenación de menor a mayor de una lista de N bytes sin signo almacenados en memoria, supuestos los siguientes contenidos de memoria ya almacenados:~~

~~M[0] = 1~~
~~M[1] = N~~
~~M[2] = Posición de memoria inmediatamente siguiente al último elemento de la lista~~

Utilice el algoritmo siguiente:

```
for i:=N-1 downto 1 do
        for j:=i-1 downto 0 do
                if LISTA(i) < LISTA(j) then
                        begin
                                Temp := LISTA(i);
                                LISTA(i) := LISTA(j);
                                LISTA(j) := Temp;
                        end;
```

b) Realice una codificación del repertorio de instrucciones, teniendo en cuenta que hay 256 posiciones de memoria, cada una de un byte, y que sea lo más sencilla posible para el desarrollo de los siguientes apartados, aunque desperdicie espacio en memoria.
c) Diseñe el camino de datos (datapath) del ordenador que contenga sólo ese repertorio de instrucciones.
d) Diseñe la estructura de la unidad de control.
e) Escriba, en un lenguaje de alto nivel, el contenido de la memoria de control.

14.

a) Diseñe una mínima exo-arquitectura de registros de uso general del tipo *load-store*, con un registro destino y dos registros fuente, que permita implementar el programa en C que se muestra a continuación. Se trata de decidir los registros, describir los modos de direccionamiento (con dibujos), y concebir un mínimo repertorio de instrucciones en lenguaje máquina (menos de 8). El programa ordena de menor a mayor una lista de N bytes sin signo almacenados en memoria.

```
for (i=N-1; i>=1; i--)
    for (j=i-1; j>=0; j--)
        if (LISTA[i] < LISTA[j])
        {
            temp = LISTA[i];
            LISTA[i] = LISTA[j];
            LISTA[j] = temp;
        }
```

b) Escriba en ensamblador el programa completo utilizando esas instrucciones. Suponga que las constantes que necesite se encuentran ya almacenadas en posiciones de memoria.
c) Diseñe una micro-arquitectura para los apartados anterior; es decir, dibuje el camino de datos. Puede inspirarse en la arquitectura de Tanenbaum.
d) Diseñe la unidad de control microprogramada, y escriba el microprograma completo.

15.

Se quiere diseñar un computador microprogramado de 32 bits de ancho de palabra, con 29 bits de direccionamiento de memoria, que tenga el juego de instrucciones siguiente:

ADD	**A, [X]**	
SUB	**A, [X]**	
MOVE	**A, [X]**	
MOVE	**[X], A**	
JMP	**[X]**	(salto a la posición de memoria X)
JZ	**[X]**	(salto a la posición de memoria X si el indicador de cero es 1)

Diseñe el formato de instrucción, así como la ruta de datos y la unidad de control del computador, estableciendo la conexión de los diversos elementos: ALU, memoria, unidad de control, etc., con sus señales de control.

16.

La Figura 10 representa el camino de datos de una CPU de 16 bits, con arquitectura de acumulador, para la que se desea diseñar una unidad de control microprogramada. Esta CPU consta de los siguientes elementos:

- 8 registros de 16 bits: contador de programa (PC), acumulador (AC), puntero de pila (SP), registro de instrucción (IR), registro para uso auxiliar (TMP), registro máscara de direcciones cuyo valor es 0FFFh (AMASK) y dos registros cuyo valor es 1 y -1.
- Un registro de direcciones de memoria principal (MAR) de 12 bits.
- Un registro de intercambio de datos de 16 bits con memoria principal (MBR).
- Una ALU de 16 bits con dos entradas A y B capaz de realizar 4 funciones: A+B, A AND B, A y $\overline{A}$.

- Un desplazador capaz de desplazar un bit a la izquierda o a la derecha la salida de la ALU.
- Buses de 16 y 12 bits entre los componentes de la CPU.
- Dos registros *buffer* de 16 bits para los buses A y B.
- Un multiplexor para seleccionar la entrada A de la ALU.

Un ciclo básico de la unidad de control consiste en una secuencia de 4 subciclos controlada por un reloj de 4 fases:

1. Se lee de la memoria de control la siguiente microinstrucción a ejecutar y se carga en el registro de microinstrucción.
2. Los contenidos de uno o dos registros pasan a los buses A y B y se cargan en los buffers A y B.
3. Las entradas de la ALU están estabilizadas. Se da tiempo a la ALU y al desplazador para que produzcan una salida estable y se carga el MAR si es necesario.
4. La salida del desplazador está estabilizada. Se almacena el bus C en un registro si es necesario y/o en el MBR si es necesario. La elección de la siguiente microinstrucción (y la carga del contador de microprograma) se realiza en este subciclo a partir de N y Z (que ya son válidos), de los campos *tipo de salto* y *microdirección* del registro de microinstrucción.

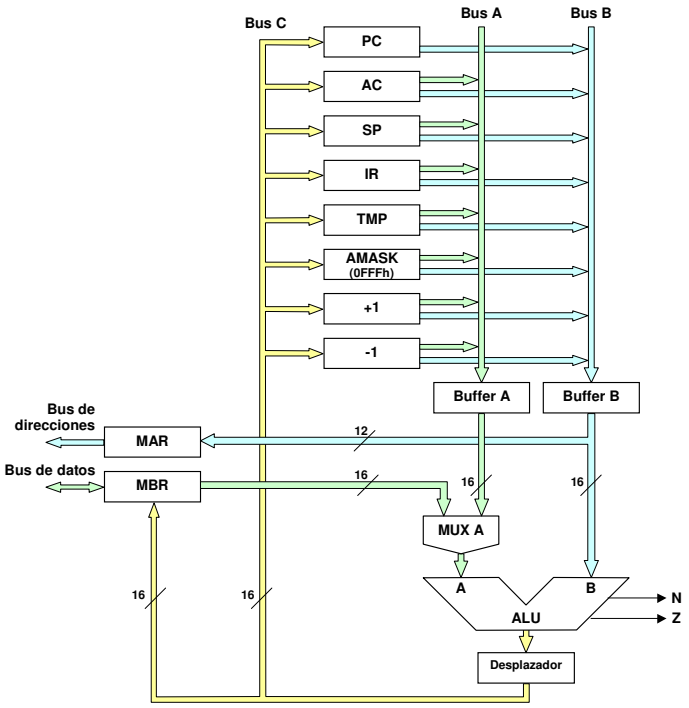


Figura 10. Camino de datos del problema 16.

El repertorio de instrucciones máquina de esta CPU se describe en la Tabla 4.

a) Diseñe el formato de microinstrucción para controlar esta CPU.

b) Diseñe la unidad de control microprogramada.

Tabla 4. Repertorio de instrucciones máquina para el problema 16. En las instrucciones locales, X es un número de 12 bits en complemento a 2 (de -2048 a 2047); en las restantes X es un número positivo de 12 bits (0 a 4095).

Codificación	Ensamblador	Instrucción	Significado
0000xxxxxxxxxxxx	LODD X	Carga directa	AC := M[X]
0001xxxxxxxxxxxx	STOD X	Almacenamiento directo	M[X] := AC
0010xxxxxxxxxxxx	ADDD X	Suma directa	AC := AC + M[X]
0011xxxxxxxxxxxx	SUBD X	Resta indirecta	AC := AC - M[X]
0100xxxxxxxxxxxx	JPOS X	Salto si positivo	if AC ≥ 0 then PC := X
0101xxxxxxxxxxxx	JZER X	Salto si cero	if AC = 0 then PC := X
0110xxxxxxxxxxxx	JUMP X	Salto incondicional	PC := X
0111xxxxxxxxxxxx	LOCO X	Carga de constante	AC := X (0 ≤ X ≤ 4095)
1000xxxxxxxxxxxx	LODL X	Carga local	AC := M[SP + X]
1001xxxxxxxxxxxx	STOL X	Almacenamiento local	M[SP + X] := AC
1010xxxxxxxxxxxx	ADDL X	Suma local	AC := AC + M[SP + X]
1011xxxxxxxxxxxx	SUBL X	Resta local	AC := AC - M[SP + X]
1100xxxxxxxxxxxx	JNEG X	Salto si negativo	if AC < 0 then PC := X
1101xxxxxxxxxxxx	JNZE X	Salto si no cero	if AC ≠ 0 then PC := X
1110xxxxxxxxxxxx	CALL X	Llamada a subrutina	SP := SP - 1; M[SP] := PC; PC := X
1111000000000000	PUSH	PUSH del acumulador en la pila	SP := SP - 1; M[SP] := AC
1111010000000000	POP	POP de la pila en el acumulador	AC := M[SP]; SP := SP + 1
1111100000000000	RETN	Retorno de subrutina	PC := M[SP]; SP := SP + 1
1111110000000000	SWAP	Intercambio de AC y SP	TMP := AC; AC := SP; SP := TMP

c) Escriba el microprograma. En cada línea deberá aparecer una microinstrucción, con las microórdenes separadas por punto y coma. Use la siguiente notación de alto nivel para las microórdenes:

- *Funciones de la ALU*: *Reg* **:=** *Reg* **+** *Reg*; *Reg* **:=** *Reg* **AND** *Reg*; *Reg* **:=** *Reg*; *Reg* **:=** $\overline{Reg}$
- *Desplazamientos*: *Reg* **:=** $\overleftarrow{Expr}$; *Reg* **:=** $\overrightarrow{Expr}$
- *Examen de un registro sin almacenarlo*: ALU **:=** *Expr*
- Lectura y escritura en memoria: MBR **:=** M[MAR]; M[MAR] **:=** MBR
- *Saltos incondicionales*: **goto** *microdirección*
- *Saltos condicionales*: **if** N **then goto** *microdirección*; **if** Z **then goto** *microdirección*
- *Salto en función del código de operación*: **goto f**(IR)

17. Una CPU, capaz de ejecutar el repertorio de instrucciones de la Tabla 5, consta de los siguientes elementos:

- 8 registros de 16 bits: contador de programa (PC), acumulador (AC), puntero de pila (SP), registro de instrucción (IR), registro para uso auxiliar (TMP), registro máscara de direcciones cuyo valor es 0FFFh (AMASK) y dos registros cuyo valor es 1 y -1.
- Dos buses (bus A y bus B, cada uno de 16 bits) conectados a la salida de los 8 registros y también a dos registros buffers (buffer A y buffer B).
- Un bus C de 16 bits conectado a la entrada de los 8 registros.
- Un registro de direcciones de memoria principal (MAR) de 12 bits. Su entrada proviene del buffer B.
- Un registro de intercambio de datos de 16 bits con memoria principal (MBR).
- Una ALU de 16 bits con dos entradas L y R capaz de realizar 4 funciones: A+B, A AND B, A y $\overline{A}$. La entrada L puede provenir del Buffer A o de MBR, gracias a un multiplexor externo a la ALU. La entrada R proviene del Buffer B. La ALU dispone de dos salidas de negativo (N) y cero (Z).
- Un desplazador capaz de desplazar un bit a la izquierda o a la derecha la salida de la ALU. También puede dejar pasar el dato inalterado. Su salida se conecta al registro MBR y al bus C.
- Buses de 16 y 12 bits entre los componentes de la CPU.

Un ciclo de la unidad de control consiste en una secuencia de 4 subciclos controlada por un reloj de 4 fases:

1. Se lee de la memoria de control la siguiente μinstrucción a ejecutar y se carga en el registro de μinstrucción.
2. Los contenidos de uno o dos registros pasan a los buses A y B y se cargan en los buffers A y B.
3. Las entradas de la ALU están estabilizadas. Se da tiempo a la ALU y al desplazador para que produzcan una salida estable y se carga el MAR si es necesario.
4. La salida del desplazador está estabilizada. Se almacena en un registro a través del bus C si es necesario, y/o en el MBR si es necesario. La elección de la siguiente μinstrucción (y la carga del contador de μprograma) se realiza en este subciclo a partir de N y Z (que ya son válidos), de los campos *tipo de*

Tabla 5. Repertorio de instrucciones máquina para el problema 17. En las instrucciones locales, X es un núm. de 12 bits en complemento a 2 (de -2048 a 2047); en las restantes X es un núm. positivo de 12 bits (0 a 4095).

Codificación	Ensamblador	Instrucción	Significado
0000xxxxxxxxxxxx	JPOS X	Salto si positivo	if AC ≥ 0 then PC := X
0001xxxxxxxxxxxx	JZER X	Salto si cero	if AC = 0 then PC := X
0010xxxxxxxxxxxx	JUMP X	Salto incondicional	PC := X
0011xxxxxxxxxxxx	LOCO X	Carga de constante	AC := X (0 ≤ X ≤ 4095)
0100xxxxxxxxxxxx	LODL X	Carga local	AC := M[SP + X]
0101xxxxxxxxxxxx	STOL X	Almacenamiento local	M[SP + X] := AC
0110xxxxxxxxxxxx	ADDL X	Suma local	AC := AC + M[SP + X]
0111xxxxxxxxxxxx	SUBL X	Resta local	AC := AC - M[SP + X]
1000xxxxxxxxxxxx	JNEG X	Salto si negativo	if AC < 0 then PC := X
1001xxxxxxxxxxxx	JNZE X	Salto si no cero	if AC ≠ 0 then PC := X
1010xxxxxxxxxxxx	CALL X	Llamada a subrutina	SP := SP - 1; M[SP] := PC; PC := X
1011----------	PUSH	PUSH del acumulador en la pila	SP := SP - 1; M[SP] := AC
1100----------	POP	POP de la pila en el acumulador	AC := M[SP]; SP := SP + 1
1101----------	RETN	Retorno de subrutina	PC := M[SP]; SP := SP + 1

salto y *µdirección* del registro de µinstrucción.

a) Dibuje el camino de datos a partir de la descripción previa.

b) Diseñe la unidad de control microprogramada.

18. Suponga un ordenador que tenga sólo las tres instrucciones de una dirección siguientes:

SUB X, que resta al acumulador A el contenido de la posición de memoria X.

STORE X, que almacena el contenido de A en la posición de memoria X.

JMPNEG X, que salta a la posición X si el contenido de A es negativo.

Diseñe:

a) El camino de datos de este ordenador.

b) Su unidad de control, incluyendo el µprograma en pseudocódigo.

19. La Figura 11 muestra el camino de datos de un procesador cuya unidad de control hemos de diseñar. El repertorio de instrucciones incluye entre otras: cargar AC desde memoria, almacenar AC en memoria, saltar incondicionalmente, saltar si acarreo, saltar si cero, mover AC a Y, efectuar operaciones aritmético lógicas, etc. El tamaño de cada instrucción es de una palabra de memoria. Cada instrucción que opera con una dirección de memoria puede tener direccionamiento directo o absoluto a memoria (bit i=0) o indirecto a través de memoria (bit i=1). Las instrucciones que no operan con memoria siempre tienen el bit i=0.

a) Diseñe la unidad de control microprogramada.

b) Escriba las partes del microprograma correspondientes a:

- *Interrupción*. En cada ciclo, antes de la fase de captación, se comprueba si la señal INT está a uno; si es así, se guarda PC y se salta a la dirección de la rutina de interrupción, que estará disponible en el bus de datos.
- *Captación de instrucción*.
- *Direccionamiento indirecto*. En caso de que la instrucción utilice direccionamiento indirecto, ha de captarse la dirección de memoria del dato, contenida en la posición de memoria que se indica en la instrucción.

No escriba la parte correspondiente a la ejecución de cada instrucción.

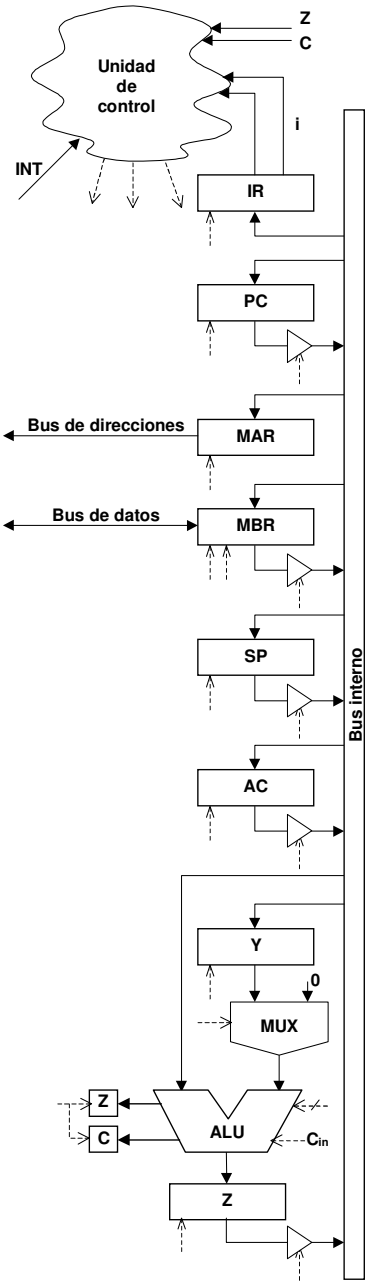


Figura 11. Camino de datos del problema 19.

20. La Figura 12 muestra el camino de datos de un procesador de 32 bits que direcciona la memoria por bytes y en el que cada instrucción y cada dato ocupa una palabra completa (4 bytes). El multiplexor MUX2 selecciona o bien la salida del registro Y o un valor constante igual a 4 utilizado para incrementar el contenido del contador de programa. Las operaciones de lectura o escritura en memoria pueden consumir un número de ciclos de reloj indeterminado. Para acomodar la variabilidad en el tiempo de respuesta, el procesador tendrá que esperar hasta recibir de la memoria la señal MFC (*Memory Function Completed*).

Se desean implementar tres instrucciones de suma:

Instrucción	*Direccionamiento del operando fuente*	*Formato*
`addr Rdst,Rsrc`	registro	una palabra
`addi Rdst,[Rsrc]`	indirecto a memoria a través de registro	una palabra
`addx Rdst,[Rsrc+desp]`	indexado	dos palabras, la segunda contiene el desplazamiento

Escriba (en lenguaje de transferencia de registros o de alto nivel) un microprograma que incluya la fase de captación de instrucción y la fase de ejecución de cada una de esas tres instrucciones.

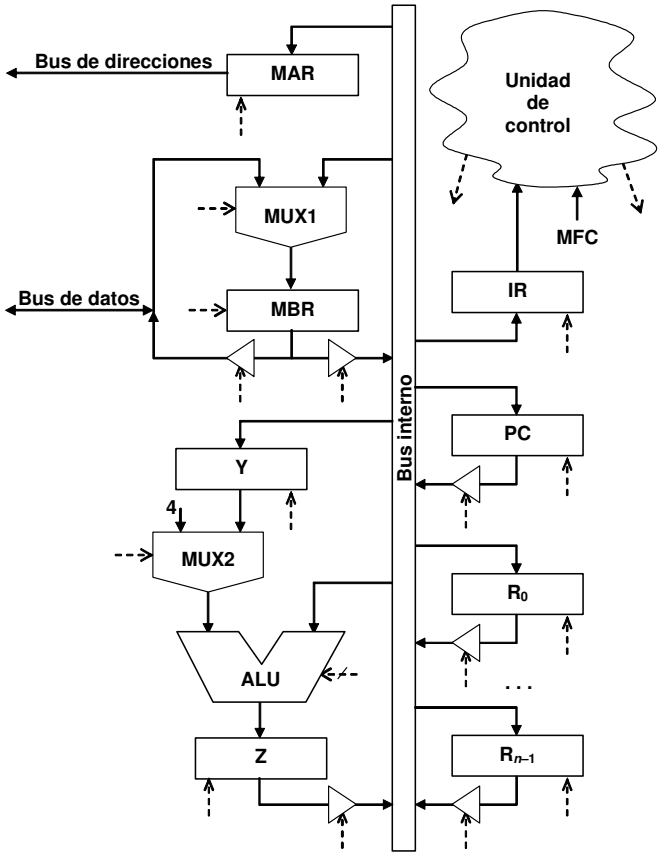


Figura 12. Camino de datos del problema 20.