Universidad de Granada
E. T. S. Ingenierías Informática y de Telecomunicación

Grado en Ingeniería Informática
ESTRUCTURA DE COMPUTADORES
Relación de problemas de
Nivel de lenguaje máquina

Departamento Arquitectura y Tecnología de Computadores

1. La memoria de un ordenador tiene capacidad para 65536 palabras de 25 bits cada una. El código de instrucción de ese ordenador se divide en cuatro partes: un código de operación, un bit de modo de direccionamiento directo/indirecto, dos bits para especificar un registro del procesador y una parte de dirección.
a) ¿Cuál es el máximo número de operaciones que se pueden codificar si cada una de ellas sólo puede ocupar una palabra?
b) ¿Cuántos registros hay en el ordenador?
c) ¿Cuántos bits tiene el PC (program counter)?

2. El direccionamiento por base ponen en juego 4 longitudes:
*L*1: número de bits del campo que especifica un registro base.
*L*2: número de bits del campo que especifica un desplazamiento.
*L*3: número de bits de un registro base.
*L*4: número de bits de una dirección de memoria.
Para un conjunto dado de valores de los registros base:
a) ¿A cuántas direcciones distintas se puede hacer referencia?
b) ¿Qué fracción de la memoria total se puede direccionar?

3. En un ordenador hipotético se puede acceder directamente a 16 MBytes. La CPU de este ordenador dispone de 16 registros base. Determine el tamaño del campo desplazamiento de modo que sea posible direccionar el 6,25% del espacio total de direcciones sin cambiar el contenido de los registros base.

4. Un procesador hipotético puede acceder a 4 GB de RAM. El procesador dispone de 16 registros base. Determine el tamaño del campo desplazamiento de modo que sea posible direccionar, mediante direccionamiento relativo a registro base, el 6,25% del espacio total de direcciones sin cambiar el contenido de los registros base en cada uno los dos siguientes supuestos:
a) Todos los registros base contienen el mismo valor.
b) Todos los registros base contienen un valor tal que las zonas que se pueden direccionar a partir de ellos son disjuntas.

5. Un microprocesador dispone de 4 registros base y puede direccionar por bytes hasta 1 MB. El único modo de direccionamiento de memoria existente es relativo a registro base. La dirección de memoria efectiva se calcula según: EA = base*16 + desplazamiento.
a) ¿Cuál es el tamaño del campo desplazamiento si sabemos que es posible direccionar como máximo el 25% del espacio total de direcciones sin cambiar el contenido de los registros base?
b) ¿Cuál es el tamaño de los registros base?

6. Dados los valores de memoria que se indican a continuación y una arquitectura de acumulador, ¿qué valores cargan en el acumulador las instrucciones 1,2, ..., 6? La palabra 20 contiene 40, la 30 contiene 50, la 40 contiene 60 y la 50 contiene 70.
1. LOAD(inmediato) 20
2. LOAD(directo) 20
3. LOAD(indirecto) 20
4. LOAD(inmediato) 30

5. LOAD(directo) 30
6. LOAD(indirecto) 30

7. Compare las máquinas de 0, 1, 2 y 3 direcciones elaborando con los repertorios de instrucciones de cada una de ellas un programa que permita realizar la operación:

X = (A+B*C) / (D-E*F)

sin cambiar el valor de A, B, C, D, E, F, almacenadas en memoria.
Los repertorios de instrucciones son:

M0:	M1:	M2:	M3:
PUSH MCARGA M	MOV X,Y (X:=Y)	MOV X,Y (X:=Y)	
POP M	ALMACENA M	ADD X,Y (X:=X+Y)	ADD X,Y,Z (X:=Y+Z)
ADD	ADD M	SUB X,Y (X:=X-Y)	SUB X,Y,Z (X:=Y-Z)
SUB	SUB M	MUL X,Y (X:=X*Y)	MUL X,Y,Z (X:=Y*Z)
MUL	MUL M	DIV X,Y (X:=X/Y)	DIV X,Y,Z (X:=Y/Z)
DIV	DIV M		

donde M es una dirección de memoria de 16 bits, X, Y, y Z son direcciones de memoria (16 bits) o de registros (4 bits). La máquina de 0 direcciones utiliza una pila y la de 1 un acumulador. Si se consideran códigos de operación de 8 bits. ¿Cuántos bits necesita cada máquina para calcular X?

8. Sea un computador de cero direcciones (pila) cuyo juego de instrucciones contiene `push`, `pop`, `add`, `sub`, `mul` y `div`.
Escriba la rutina que calcula el producto escalar de dos vectores de tres componentes (a_1, a_2, a_3) y (b_1, b_2, b_3). Las componentes de los vectores están en memoria, y el resultado debe almacenarse también en memoria.

9. Una calculadora programable con arquitectura de pila, 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

Escriba un programa que pida al usuario tres números A, B y C, y saque como resultado las dos soluciones de la ecuación de 2º grado $Ax^2+Bx+C=0$. Si no hay solución real el programa terminará sin sacar ningún resultado.

10. Cierta máquina de pila implementada en memoria dispone de las siguientes instrucciones:

FETCH X	push: *tope* ← *tope* + 1; C(*tope*) ← C(X)
STORE X	pop: C(X) ← C(*tope*); *tope* ← *tope* - 1
ADD	C(*tope* - 1) ← C(*tope*) + C(*tope* - 1); *tope* ← *tope* - 1
SUB	C(*tope* - 1) ← C(*tope*) - C(*tope* - 1); *tope* ← *tope* - 1
MPY	C(*tope* - 1) ← C(*tope*) * C(*tope* - 1); *tope* ← *tope* - 1
DIV	C(*tope* - 1) ← C(*tope*) / C(*tope* - 1); *tope* ← *tope* - 1
DUP	C(*tope* + 1) ← C(*tope*); *tope* ← *tope* + 1
EXCH	intercambiar C(*tope*) y C(*tope* - 1)

donde *tope* es la dirección de la cima de la pila y C(*posición*) es el contenido de esa posición de memoria. Escriba el programa para arquitectura de tipo pila más corto posible para evaluar

Z ← (((B*C)-A)^2)/(B*C)

11. Un ordenador con 16 registros direccionables (R0, R1, ..., R15) de 16 bits y una memoria de 64 KB tiene instrucciones cuyo tamaño es un número entero de bytes. El código de operación (que incluye el modo de direccionamiento) ocupa un byte. El destino de las instrucciones con dos operandos es el primer operando.
a) Para cada una de las instrucciones del siguiente programa, indique qué modos de direccionamiento se utilizan y dibuje un esquema (cajitas y flechas) que muestre cómo funcionan esos modos de direccionamiento (con direcciones y valores para este programa concreto y suponiendo ubicación en

memoria *little-endian*):

```
1000: SUB R1, R1
1002: AND R0, R1
1004: MOV [++R1], R0
1006: XOR 0001h[R0], R1
100A: JNZ -0Ah
100C: ADD [0002h], #0003h
```

b) ¿Cuáles son los contenidos de R0, R1, M[1], M[2] y M[3] cuando el programa finaliza para el caso de ubicación *little-endian*? ¿Y para *big-endian*?

12. Un ordenador tiene las siguientes características:

- Microprocesador con 16 registros direccionables (r0, r1, ..., r15) de 32 bits cada uno.
- Memoria de 4 GB direccionable por bytes.
- El tamaño de palabra (tanto en el microprocesador como en memoria) es 32 bits.
- Utiliza la ubicación *little-endian*.
- El tamaño de las instrucciones es un número entero de bytes.
- El código de operación de las instrucciones incluye los modos de direccionamiento y siempre ocupa un byte.
- Los desplazamientos son de 16 bits.
- El destino de las instrucciones con dos operandos es el primer operando.

a) Para cada una de las instrucciones del siguiente programa, indique qué modos de direccionamiento se utilizan y dibuje un esquema (cajitas y flechas) que muestre cómo funcionan esos modos de direccionamiento (con direcciones y valores para este programa concreto).

```
10000000:          sub r1, r1
10000002: Bucle:   and r0, r1
10000004:          mov [r1++], r0
10000006:          xor Tabla[r0], r1  ; Tabla es una etiqueta y su despl. es 0
1000000A:          jnz Bucle
1000000D:          add [00000002h], #00001000h
```

b) Muestre en una tabla cuáles son los contenidos de R0, R1, y la primera palabra de memoria (la que se encuentra almacenada a partir de la dirección 0) desde el principio hasta que el programa finaliza.

13. Un computador dispone de las 9 instrucciones adjuntas.

Para cada una de estas instrucciones y operandos indique su(s) modo(s) de direccionamiento y dibuje un esquema lo más didáctico, preciso y claro que pueda de su funcionamiento (se trata de explicar sólo con un dibujo, sin palabrería, lo que hace cada instrucción).

1.	ADD	Reg,#Dato
2.	ADD	RegDst,RegSrc
3.	LD	Reg,#Dato
4.	LD	Reg,[Dir]
5.	LD	Reg,[Reg]
6.	MOVE	RegSrc,RegDst
7.	ST	Reg,[Dir]
8.	ST	Reg,[Reg]
9.	Bcond	Desp

14. Un computador tiene 8 registros (R1, R2, ..., R8), de los cuales R1 actúa como contador de programa, y R2 como puntero de pila, y dispone solamente de las siguientes instrucciones:

ADD	Reg,#Dato
ADD	RegDst,RegSrc
LD	Reg,#Dato
LD	Reg,[Dir]
LD	Reg,[Reg]
MOVE	RegSrc,RegDst
ST	Reg,[Dir]
ST	Reg,[Reg]
Bcond	Desp

Las instrucciones que tienen direccionamiento inmediato o absoluto ocupan 2 palabras, y el resto una palabra. Los registros no se autoincrementan ni autodecrementan, excepto el contador de programa.

Programe, con este juego de instrucciones, el equivalente a las siguientes funciones:

a) PUSH Reg y POP Reg

b) ADD [Dir],[Reg++]

c) CALL [Dir] (llamada a la rutina que comienza en la dirección de memoria contenida en la dirección de memoria Dir)

El valor de los registros utilizados como almacenamiento temporal será el mismo que el que tuvieran antes de los fragmentos de programa correspondientes.

15. 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.

a) Escriba un programa en ensamblador para esta máquina que implemente la operación

$$R \leftarrow M * N$$

mediante el siguiente algoritmo en pseudocódigo:

$R \leftarrow 0$
Repetir M veces la siguiente instrucción:
$R \leftarrow R + N$

Pueden usarse posiciones temporales de memoria y etiquetas. Se puede asumir que hay una posición de memoria llamada UNO que contiene la constante 1.

b) Si le estuviera permitido implementar *una* instrucción más para este ordenador, ¿cuál escogería? Razone su respuesta escribiendo de nuevo el programa usando la nueva instrucción.

16. Considere un ordenador hipotético con arquitectura de acumulador y un repertorio de instrucciones formado por sólo dos instrucciones de *n* bits. El primer bit especifica el código de operación, y los bits restantes direccionan una de las 2^{n-1} palabras de *n* bits de la memoria principal. Las dos instrucciones son:

`SUB X` *Subtract*. Restar al acumulador el contenido de la posición X, y memorizar el resultado en la posición X y en el acumulador.

`BC X` *Branch if Carry*. Saltar si acarreo a la posición X: si hubo acarreo (adeudo) en la última operación SUB, colocar la dirección X (el valor X, no el contenido de la posición X) en el contador de programa.

Una palabra de memoria principal puede contener una instrucción o un número binario en notación de complemento a dos. Demuestre que este repertorio de instrucciones es razonablemente completo especificando en ensamblador cómo se podrían programar las siguientes operaciones:

a)	`CLR X`	*Clear*. Poner a 0 el contenido de la posición X y el acumulador.	Pista: Utilizar SUB.
b)	`LD X`	*Load*. Cargar: transferir el contenido de la posición de memoria X al acumulador.	Pista: Utilizar CLR, SUB, y una posición auxiliar.
c)	`ST X`	*Store*. Almacenar: transferir el contenido del acumulador a la posición de memoria X.	Pista: Utilizar SUB y suponer que la posición de memoria CERO contiene 0. Nota: Después de esta instrucción se podría poner una instrucción CLR CERO para dejar CERO a 0.
d)	`ADD X`	*Add*. Sumar el contenido del acumulador con el contenido de la posición X. El resultado se almacena en la posición X y en el acumulador.	Pista: Utilizar ST, CLR CERO, SUB, y una posición auxiliar.
e)	`SHL X`	*Shift Left*. Desplazar la posición X hacia la izquierda. El resultado se almacena en la posición X y en el acumulador.	Pista: Utilizar LD y ADD.
f)	`JMP X`	*Jump*. Saltar incondicionalmente: colocar la dirección X en el contador de programa.	Pista: Utilizar CLR, SUB, JC, y suponer que la posición de memoria DOSN1 contiene 2^{n-1}.
g)	`OR X`	*Or*. Operación lógica OR entre el acumulador y el contenido de la posición X. El resultado se almacena en el acumulador.	Pista: Hacer un bucle y utilizar desplazamientos y acarreos. Utilizar posiciones temporales y suponer que la posición de memoria UNO contiene 1.

			Consejo: Utilice pseudocódigo antes de empezar a escribir instrucciones.
h)	`IN X` `OUT X`	*Input*. Operación de entrada desde un puerto en el acumulador. *Output*. Operación de salida del acumulador en un puerto.	Pista: Suponer E/S mapeada en memoria.

17. 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, supuestas 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.~~

18. ***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.

19. Traduzca a ensamblador (el repertorio de instrucciones se encuentra en la Tabla 1) la siguiente función en Pascal:

```
function min(i,j,k: integer): integer;
var m: integer;
begin
     if i < j then m:=i else m:=j;
     if k<m then m:=k;
     min := m;
end;
```

El tipo `integer` ocupa una palabra de 16 bits. Cada posición direccionable de memoria es una palabra de 16 bits. Los parámetros de la función están ya almacenados en la pila, antes de hacer la llamada, en direcciones superiores al contador de programa, siguiendo el mismo orden de los parámetros de la llamada, es decir, primero i, luego j, luego k y más abajo el PC. La función almacena la variable local m en la pila, debajo del contador de programa. El resultado de la función debe quedar almacenado en el acumulador (AC). No es responsabilidad de la función quitar de la pila los parámetros de llamada.

Tabla 1. Repertorio de instrucciones máquina para el problema 19. 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

20. Traduzca a ensamblador (el repertorio de instrucciones se encuentra en la Tabla 2) la siguiente función escrita en C:

```
int min(int i, int j, int k)
{
        int m;

        if (i < j) m = i else m = j;
        if (k < m) then m = k;
        return m;
}
```

El tipo `int` ocupa una palabra de 16 bits. Cada posición direccionable de memoria es una palabra de 16 bits. Los parámetros de la función están ya almacenados en la pila, antes de hacer la llamada, en direcciones superiores al contador de programa, siguiendo el orden inverso de los parámetros de la llamada, es decir, primero k, luego j, luego i. Más abajo en la pila está almacenado el PC. La función almacena la

Tabla 2. Repertorio de instrucciones máquina para el problema 20. 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

variable local m en la pila, debajo del contador de programa. El resultado de la función debe quedar almacenado en el acumulador (AC). No es responsabilidad de la función quitar de la pila los parámetros de llamada.

21. Sea un computador de arquitectura memoria-memoria, con dos operandos explícitos en las instrucciones aritmético-lógicas (el primero es el destino). Realice una subrutina reubicable que calcule la varianza y la media de una serie unidimensional de datos. Los parámetros de entrada y salida se pasan a través de la pila del sistema. Los de entrada serán el número de datos de la serie y la dirección de comienzo de la misma. Los de salida la varianza y la media. Puede suponer que dispone de cualquiera de las instrucciones máquina habituales, siempre que respete el modelo memoria-memoria.

Ayuda: $$Varianza = \frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N}$$

22. Escriba un programa en lenguaje ensamblador para un microprocesador de 16 bits (con varios registros de uso general de 16 bits) que genere una secuencia de 32 números aleatorios con una distribución gausiana. Un método para obtener dicha secuencia se basa en el teorema del límite central según el cual un conjunto de valores medios de secuencias de números sin correlacionar tiene una distribución gausiana de valores.

Suponga que tiene una tabla de 16 Kbytes compuesta de números de 16 bits con números sin correlacionar. Tome 256 datos de la tabla y súmelos (la suma de números de 16 bits puede producir un número de más de 16 bits). Divida por 256. De esta forma obtendrá el valor medio de esos 256 números. El número obtenido tendrá un valor aleatorio que sigue una distribución gausiana. Genere así otra tabla, de 32 palabras de 16 bits (8K / 256 = 32), con los valores obtenidos a partir de la tabla original.

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

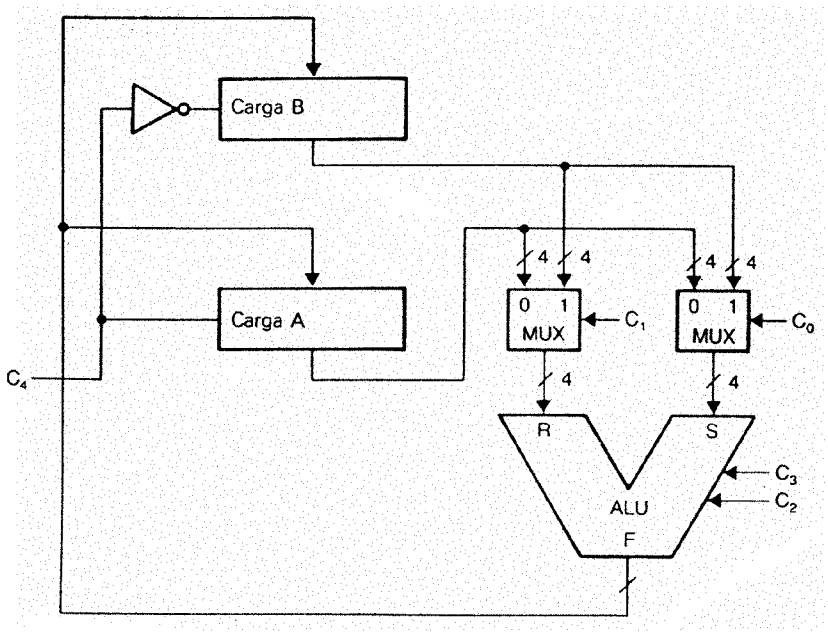


Figura 1. ALU y registros del problema 1.

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

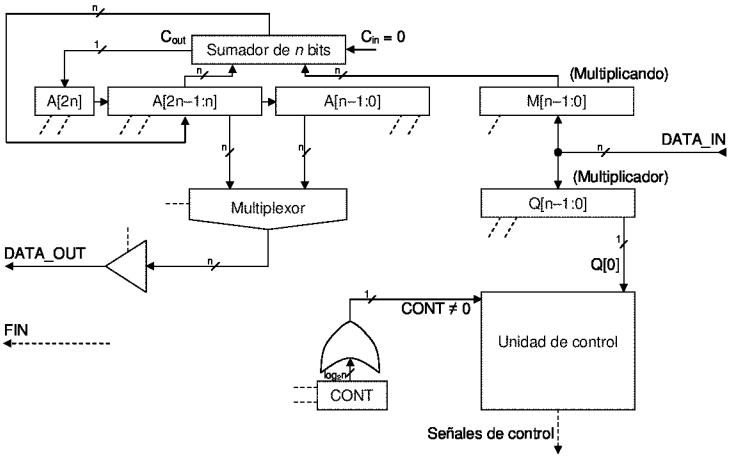
Cada palabra de control debe especificarse de acuerdo con el formato $c_4\ c_3\ c_2\ c_1\ c_0$.
Por ejemplo:

```
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 1.

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.

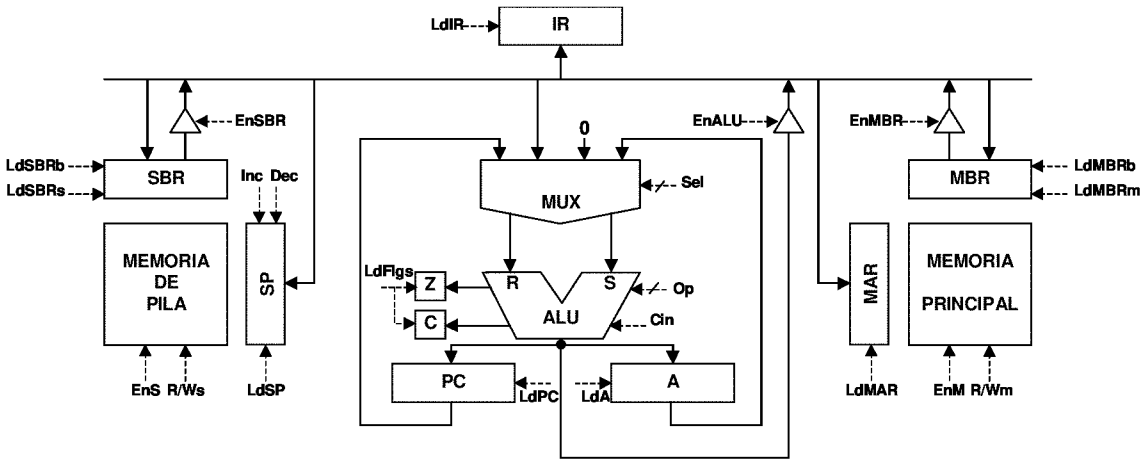


Figura 4. CPU de la máquina de pila del problema 2.

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.

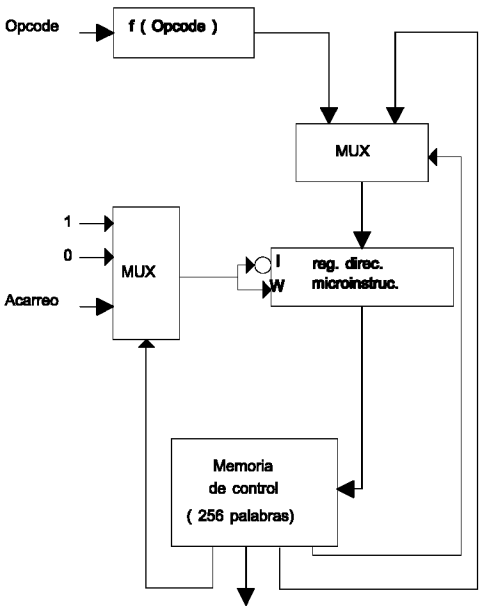


Figura 5. U.C. microprogramada del problema 5.

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:

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

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

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

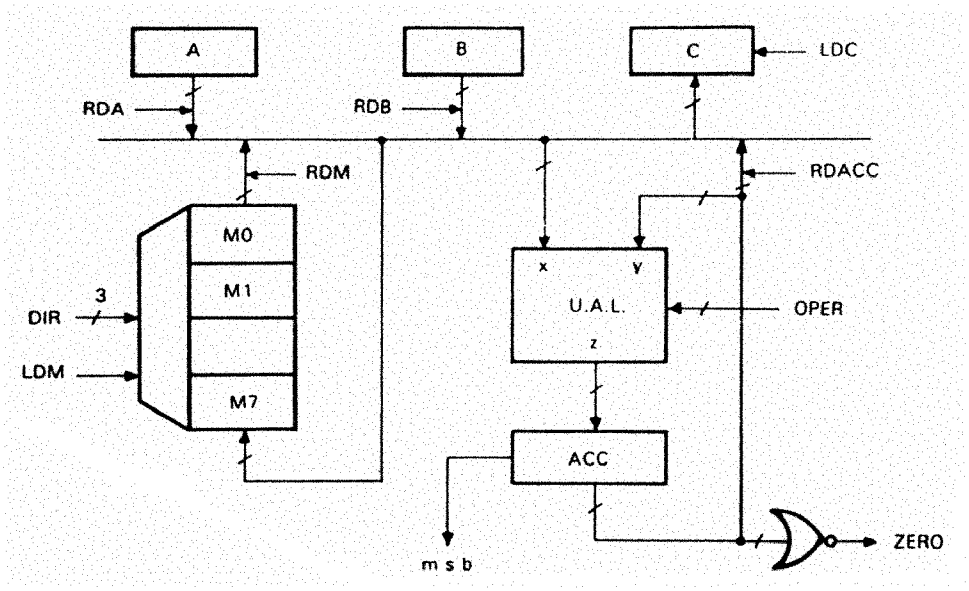


Figura 6. Unidad de procesamiento del problema 6.

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.

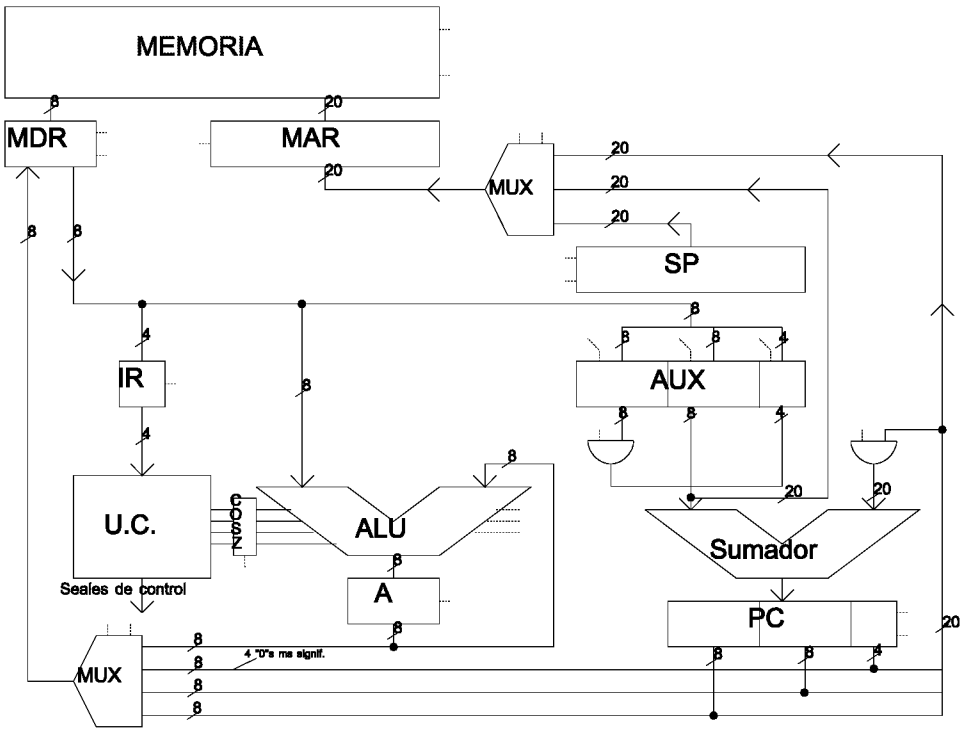


Figura 7. Camino de datos del ordenador del problema 1.

Tabla 2. Instrucciones máquina del ordenador del problema 7.

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.

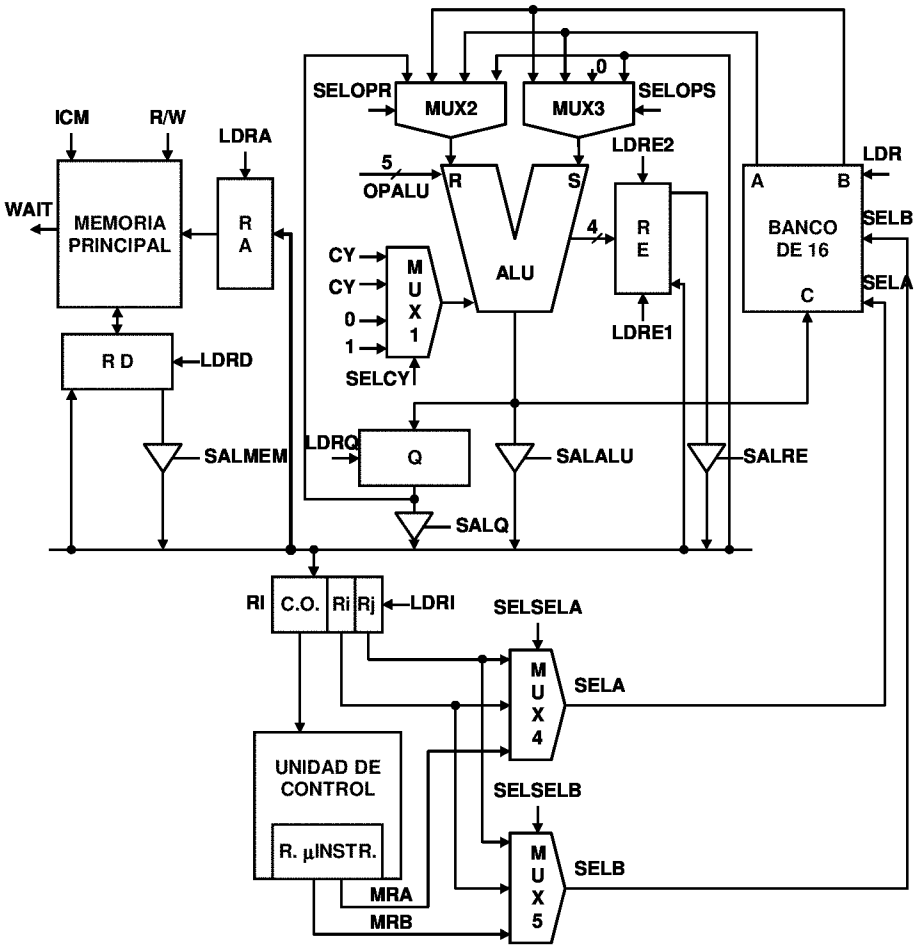


Figura 8. Esquema de bloques del camino de datos del problema 8.

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.

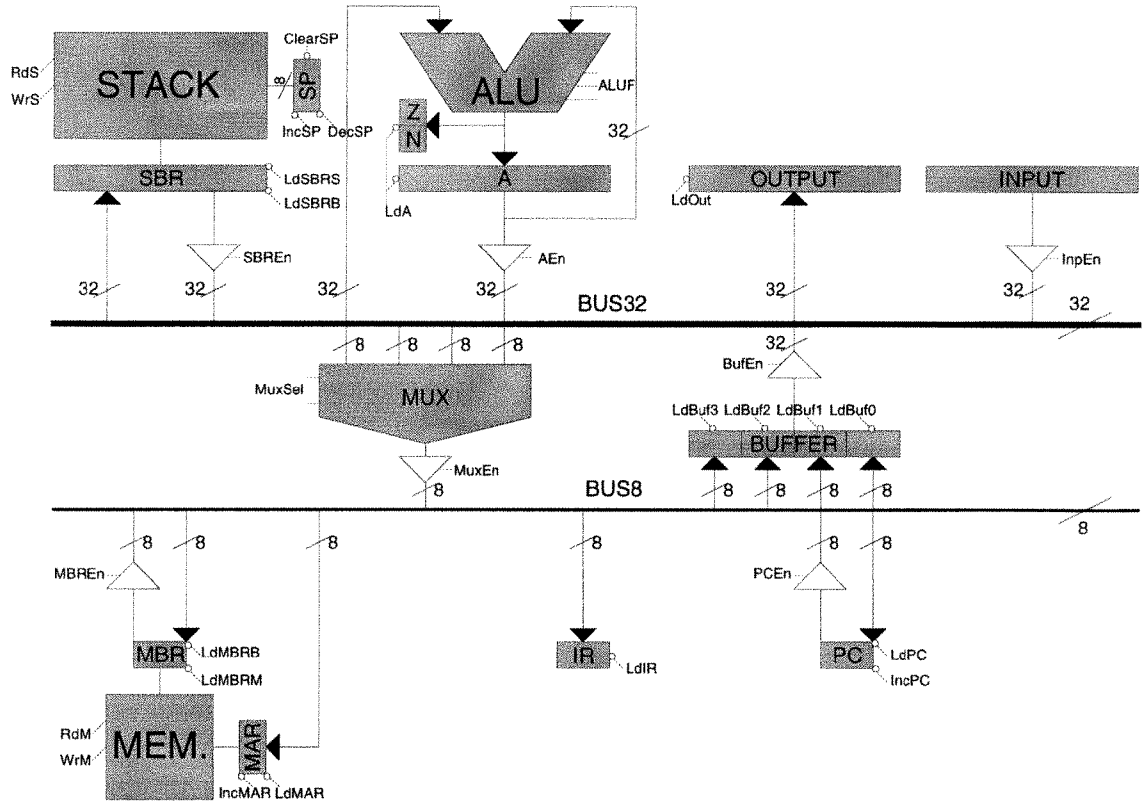


Figura 9. Camino de datos de la calculadora del problema 3.

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 2.

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.

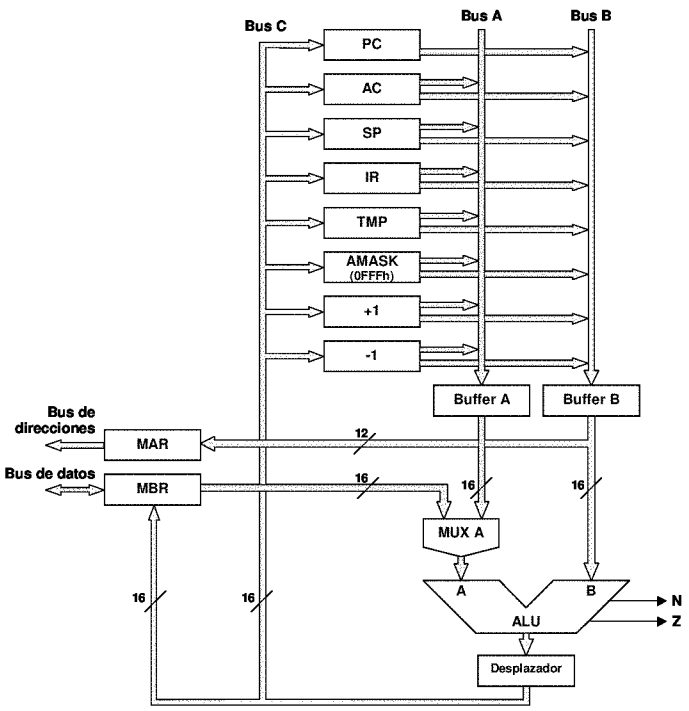


Figura 10. Camino de datos del problema 4.

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 1. 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 5. 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.

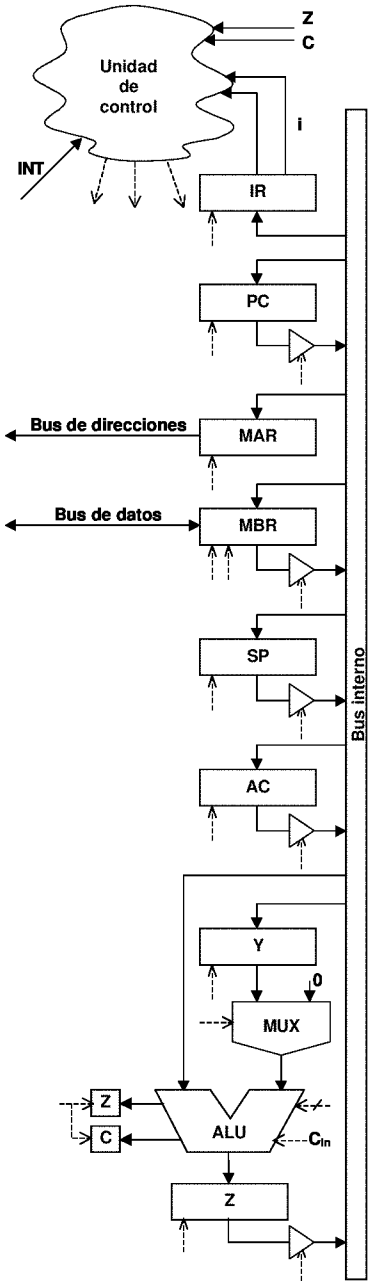


Figura 11. Camino de datos del problema 7.

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.

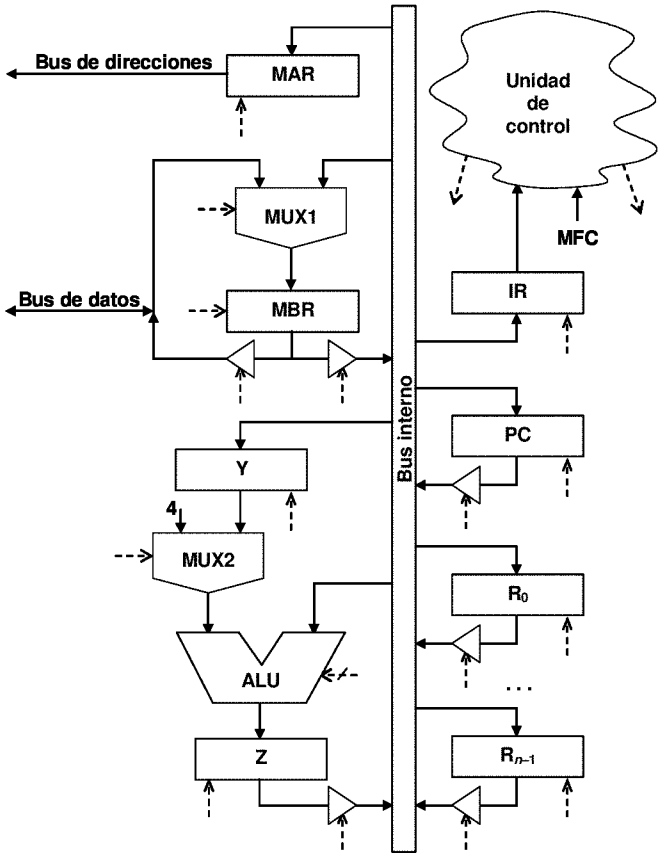


Figura 12. Camino de datos del problema 8.

Universidad de Granada
E. T. S. Ingenierías Informática y de Telecomunicación

Grado en Ingeniería Informática
ESTRUCTURA DE COMPUTADORES
Relación de problemas de
Entrada/salida y buses

Departamento Arquitectura y Tecnología de Computadores

1. Se dispone de una CPU cuyas señales se describen en la Figura 1. Como dirección de E/S se utilizan los 8 bits inferiores del bus de direcciones. Se desea conectar, mediante E/S programada, dicha CPU a un periférico de salida cuyas señales se muestran en la Figura 2 y cuyo protocolo de *handshaking* se muestra en la Figura 3.

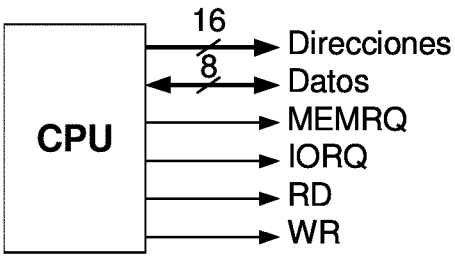


Figura 1. Señales de la CPU.

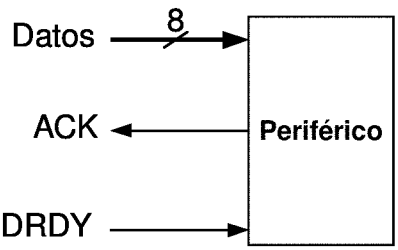


Figura 2. Señales del periférico.

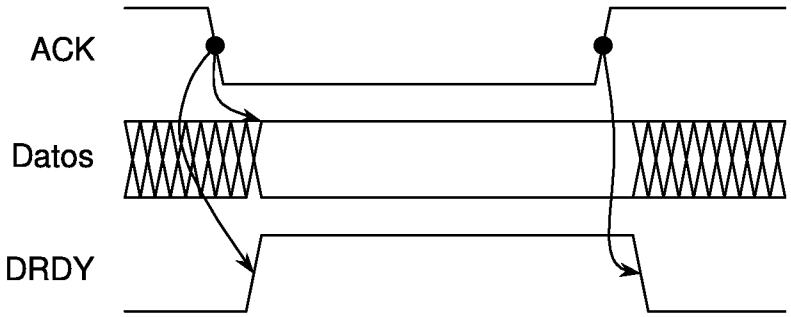


Figura 3. Protocolo de *handshaking* del periférico..

a) Diseñe una interfaz que permita la conexión entre la CPU y el periférico.
b) Escriba un programa que, de acuerdo al protocolo exigido, envíe un dato al periférico.

2. La CPU de cierto computador dispone para la E/S de las siguientes señales:

A_0-A_7 ≡ 8 líneas de dirección para seleccionar el periférico
D_0-D_7 ≡ 8 líneas de datos para leer o enviar un byte
$\overline{RD}$ ≡ 1 línea para indicar lectura (el procesador toma la información por D_0-D_7)
$\overline{WR}$ ≡ 1 línea para indicar escritura (el procesador envía la información por D_0-D_7)

La CPU dispone de las instrucciones IN *dir* y OUT *dir*, donde *dir* indica la dirección del periférico que intercambia un byte de información con un determinado registro de la CPU. Se desea emplear esta CPU para supervisar 5000 sensores de una fábrica. Cada sensor devuelve 0/1 indicando si una determinada válvula está cerrada / abierta.
a) Diseñe el sistema de direccionamiento e interfaz necesarios para poder acceder y leer los 5000 sensores.
b) Indique la secuencia de pasos necesaria para leer el estado de un sensor determinado.

3. Se dispone de una CPU, de bloques de memoria, y de periféricos, como se describe en la Figura 4, cuyas características más importantes son las siguientes:

CPU: En los accesos a memoria ($M / \overline{IO} = 1$) son válidos los 16 bits del bus de direcciones y en los accesos a E/S ($M / \overline{IO} = 0$) son válidos únicamente los 4 bits menos significativos de la dirección

(A0-A3).

Memorias: Bloques de 64 K palabras cuya activación se realiza mediante la señal $\overline{CS}$ = 0.

Periféricos: La CPU podrá leer de ellos o escribir en ellos datos de 16 bits.

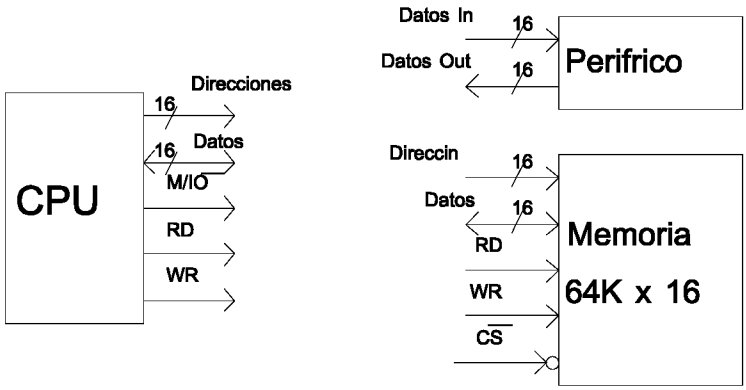


Figura 4. CPU, bloques de memoria y periféricos del problema 3.

Se desea conectar a dicha CPU 15 periféricos de los descritos, y además se desea tener una memoria de 128 K palabras dividida en dos bancos de 64 K palabras.

a) Diseñe una interfaz de E/S simple para cada periférico con dos puertos, uno de entrada y otro de salida, pero ambos con las misma dirección. Utilice un *latch* o registro *buffer* para escritura en el periférico, un registro *buffer* triestado para lectura del periférico (ambos con entradas de selección $\overline{CS}$), las señales de lectura y de escritura y una señal de selección de periférico CSi.

b) Diseñe la conexión y decodificación tanto de la E/S como de la memoria. Si faltasen líneas de direccionamiento utilice un biestable D (direccionado como puerto de salida y seleccionado para escritura a través de su entrada CK) para indicar a cuál de los dos bancos de memoria se desea realizar los siguientes accesos.

c) Piense si la solución diseñada para el apartado *b)* puede plantear problemas en la ejecución de programas (en la captación de instrucciones) al cambiar de banco de memoria para acceder a datos. Si es así soluciónelo modificando el diseño (basta añadir un par de puertas) suponiendo que la CPU activa (=1) una salida adicional FETCH siempre que realiza un acceso a memoria correspondiente a una búsqueda de instrucción, y que en el banco 0 se almacenarán datos y programas y en el banco 1 sólo datos.

d) Escriba un programa en lenguaje ensamblador para mover un bloque de 128 palabras desde la posición de memoria 00000h y siguientes hasta la 10000h y siguientes. Utilice instrucciones del siguiente repertorio:

MOV *RegDestino*,*DatoInmediato*
MOV *RegDestino*,*RegFuente*
MOV [*RegDestino*],*RegFuente*
MOV *RegDestino*,[*RegFuente*]
IN *RegDestino*,*PuertoEntradaInmediato*
OUT *PuertoSalidaInmediato*,*RegFuente*
INC *RegDestino*
DEC *RegDestino*
CMP *RegDestino*,*DatoInmediato*
CMP *RegDestino*,*RegFuente*
JMP *Etiqueta*
J*cond* *Etiqueta*

donde *cond* es cualquiera de las condiciones conocidas, los corchetes [] indican direccionamiento indirecto a memoria y las instrucciones INC, DEC y CMP afectan a los indicadores. Suponga que la CPU dispone de todos los registros que sean necesarios: R0, R1, R2, R3, R4, ...

4. Se quiere controlar mediante un microprocesador de 8 bits un gran sistema de alarma que cuenta con 100 sensores de "efecto doppler" (la señal de salida de cada uno será de 5 ó 0 voltios según detecte o no la presencia de un intruso en las inmediaciones del correspondiente sensor).

Diseñe la estructura de E/S del sistema usando circuitos integrados 74244 (Figura 5) y 74138

(Figura 6), y suponiendo E/S aislada o independiente (no mapeada en memoria. Indique las direcciones de puertos usadas.

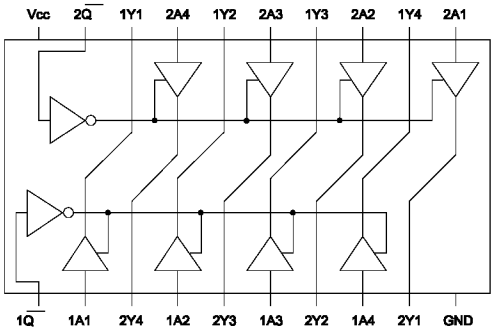
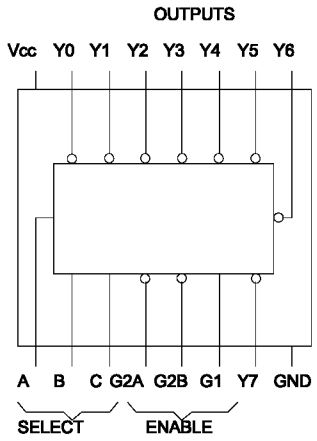


Figura 5. Buffer octal (puerto de entrada de 8 bits) 74244.



Inputs					Outputs							
Enable		Select										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	1	X	X	X	1	1	1	1	1	1	1	1
0	X	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

* G2 = G2A + G2B

Figura 6. Decodificador 3 a 8 74138.

5. Cada una de las 16 teclas del teclado de la Figura 7 consiste en un conmutador como el de la Figura 8. El comportamiento real de este circuito cuando se pulsa una tecla se muestra en la Figura 9.

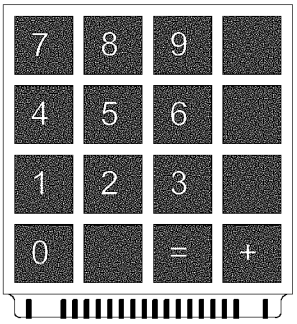


Figura 7. Teclado.

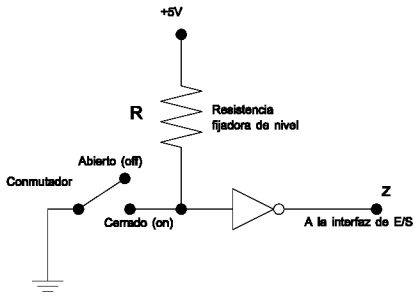


Figura 8. Conmutador.

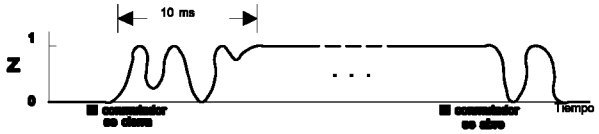


Figura 9. Comportamiento del conmutador.

a) Diseñe la interfaz necesaria para conectar el teclado a un 8086.

b) Escriba una subrutina READKEY que devuelva en el registro CL el número de tecla pulsada, de 1 a 16, 0 si no hay ninguna tecla pulsada y -1 si se ha pulsado más de una tecla. La rutina no contemplará la eliminación de rebotes.

c) Suponga que dispone de una subrutina DELAY que realiza un retardo de 15 ms. ¿Cómo podría aprovechar esa subrutina junto con la del apartado *b)* para evitar los rebotes?

6. Se desea conectar un sistema basado en 8086 (el bus del sistema se detalla en la Figura 10) al convertidor analógico / digital (A/D) de la Figura 12 mediante un interfase paralelo programable 8255 (Figura 11), que comprende tres puertos de 8 bits (A, B y C) y un registro o puerto de control.

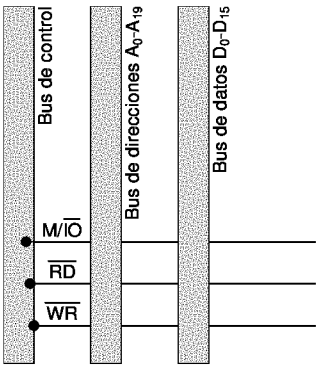


Figura 10. Señales del bus del sistema.

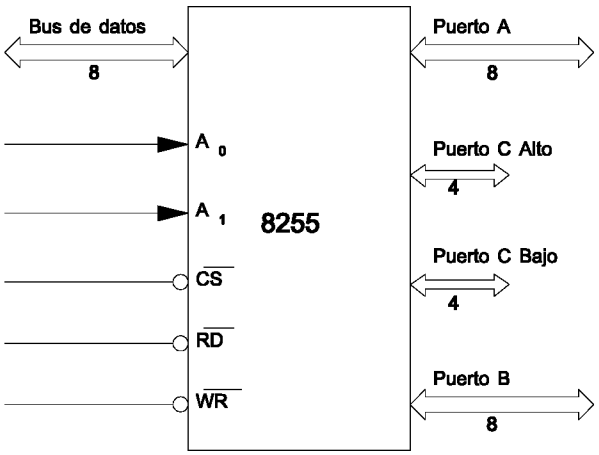


Figura 11. Interfaz paralela programable 8255.

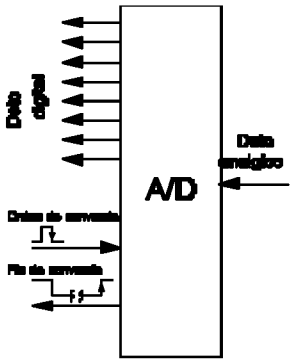


Figura 12. Convertidor A/D.

La selección de un puerto de estos cuatro se realiza mediante los bits de dirección A_0 y A_1 del 8255 (Tabla I). Se pueden seleccionar por software tres modos de funcionamiento del 8255: *modo 0* o E/S básica, *modo 1* o E/S validada, y *modo 2* o E/S validada con bus bidireccional. El modo 0 o E/S básica se usa en operaciones sencillas de E/S con control programado. En ese modo el puerto C funciona como dos puertos independientes de 4 bits, y cada puerto (Puerto A, Puerto B, Puerto C Bajo y Puerto C Alto) puede ser programado independientemente como de entrada o de salida. Cada bit de estos puertos puede ser utilizado para datos o para señales de control o de estado. La configuración de cada puerto como de salida o de entrada, así como el modo de funcionamiento de los puertos A y B puede programarse enviando la palabra de la Figura 13 al registro de control. Para escribir en el puerto C en el modo 0 se puede usar una instrucción normal de escritura en ese puerto, y si se desea escribir un solo bit sin modificar los demás se envía simplemente la palabra de la Figura 14 al registro de control.

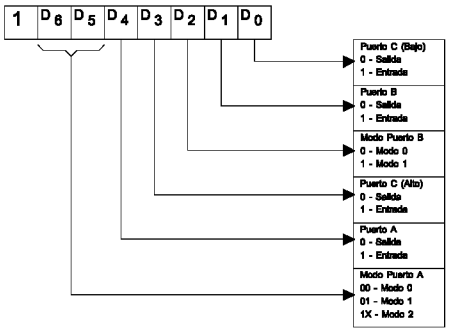


Figura 13. Palabra de control usada para programar la configuración de cada puerto.

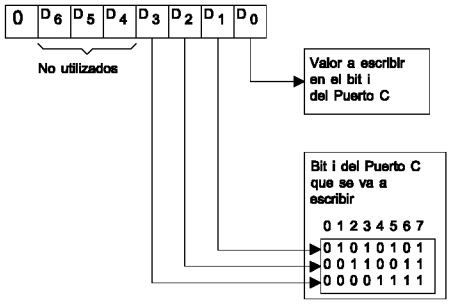


Figura 14. Palabra de control usada para escribir un bit del Puerto C.

Tabla I. Direccionamiento de puertos.

A_1	A_0	Registro direccionado
0	0	Puerto A
0	1	Puerto B
1	0	Puerto C
1	1	Registro de control

Para efectuar una conversión analógico-digital se debe enviar una orden de conversión (impulso positivo) por la línea *Orden de conversión* del convertidor. El convertidor A/D pone inmediatamente la señal de estado *Fin de conversión* a 0 y procede a realizar la conversión. Cuando la conversión finaliza el convertidor pone *Fin de conversión* a 1.

a) Diseñe un **esquema de conexión** con espacio de E/S separado del espacio de memoria de forma que las direcciones de los puertos / registros de la Tabla I sean 0, 2, 4, y 6, respectivamente, teniendo en cuenta que al 8086 pueden conectarse hasta 64 K puertos de 8 bits (los bits A_{16}-A_{19} no se usan), y que los puertos de 8 bits en direcciones pares se conectan a las líneas D_0 a D_7.

b) Dibuje el **organigrama** y escriba la correspondiente sección de un **programa en ensamblador** para leer un dato del convertidor, usando únicamente el *modo 0* del 8255.

7. Consideremos una CPU de 8 bits de ancho de palabra, que cuenta con una sola entrada de petición de interrupción **INT** y la correspondiente salida de reconocimiento de interrupción **INTA**.

Se desea dotar a la CPU de 4 niveles de petición de interrupción, en cada uno de los cuales pueden interrumpir 4 periféricos. Para ello se pide:

a) Diseñar la lógica de control de interrupciones que hay que añadir a la CPU de modo que:

- Se resuelvan las prioridades de los distintos niveles.
- Se resuelvan las prioridades de los distintos dispositivos del mismo nivel.
- Se identifique el dispositivo mediante una vector de 8 bits.

b) Describir el funcionamiento de la unidad diseñada.

8. Se dispone de una CPU (Figura 15) con las siguientes características:

- Bus de direcciones de 16 bits y bus de datos de 16 bits.
- Cuando se realizan accesos a puertos de E/S (instrucciones del tipo IN y OUT) se activa la señal IORQ, y cuando se accede a una dirección de memoria se activa la señal MEMRQ.
- La lectura se controla con la señal RD y la escritura con la señal WR.
- Hay una línea de petición de interrupción INT y una salida de reconocimiento de interrupción INTA.
- Las interrupciones son vectorizadas, con vectores de 8 bits.

a) Dibuje el interfaz de E/S que es necesario añadir a un periférico de E/S como el de la Figura 16 para conectarlo a la CPU.

b) Para un sistema con dos niveles de petición de interrupción (uno con mayor prioridad que el otro en el caso de peticiones simultáneas) con cuatro periféricos de E/S en cada nivel conectados mediante *daisy-chain*, describa las conexiones y el esquema de decodificación para los dispositivos de E/S, detallando la conexión con las líneas INT e INTA y la generación de los vectores de interrupción (pero sin detallar la implementación a nivel de puertas / biestables del *daisy-chain*).

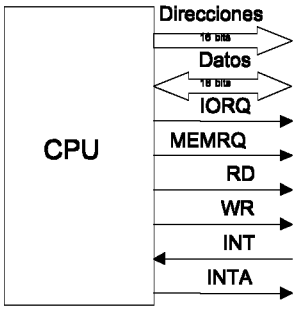


Figura 15. CPU.

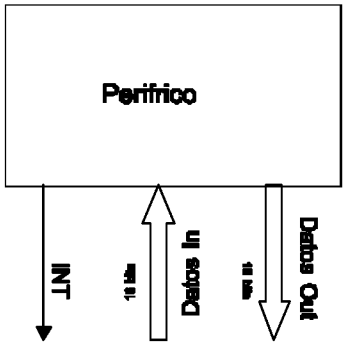


Figura 16. Periférico de E/S.

9. Un microprocesador tiene, entre otras, las siguientes características:

- Bus de datos: 8 bits, D_7-D_0.
- Bus de direcciones: 16 bits, A_{15}-A_0.
- Señales de control:
  · IORQ, indica operación de E/S (también se utiliza en el reconocimiento de interrupciones).
  · MREQ, indica operación de acceso a memoria.
  · WR, operación de escritura.
  · RD, operación de lectura.
  · F0 y F1, indican la fase de ejecución del microprocesador:

F1	F0	Fase
0	0	Ejecución
0	1	FETCH
1	0	Acceso a memoria (datos)
1	1	Reconocimiento de interrupción

  · INT3, INT2, INT1 e INT0, líneas de solicitud de interrupción.

Para la ubicación de las rutinas de interrupción se emplea vectorización. Cuando el microprocesador atiende una interrupción activa F1=1 y F0=1, y coloca en las líneas A1 y A0 del bus de

direcciones el valor binario correspondiente al nivel (0, 1, 2 y 3) de la interrupción aceptada. Seguidamente, activa IORQ para validar el contenido del bus de direcciones y leer del bus de datos un vector de 8 bits que debe suministrarle el periférico cuya solicitud de interrupción ha sido atendida.

Se desea conectar a esta CPU cuatro periféricos de sólo lectura, cada uno de los cuales suministra datos de 8 bits y cuenta con una línea de interrupción. El vector generado por cada periférico ha de ser programable desde la CPU (mediante una instrucción de salida).

Diseñe el sistema de E/S, es decir, mapa de E/S, decodificación, gestión de interrupciones. Utilice los decodificadores, registros, buffers triestado, puertas lógicas, etc. que crea convenientes.

10. Considere un paso a nivel con barrera como el de la Figura 17.

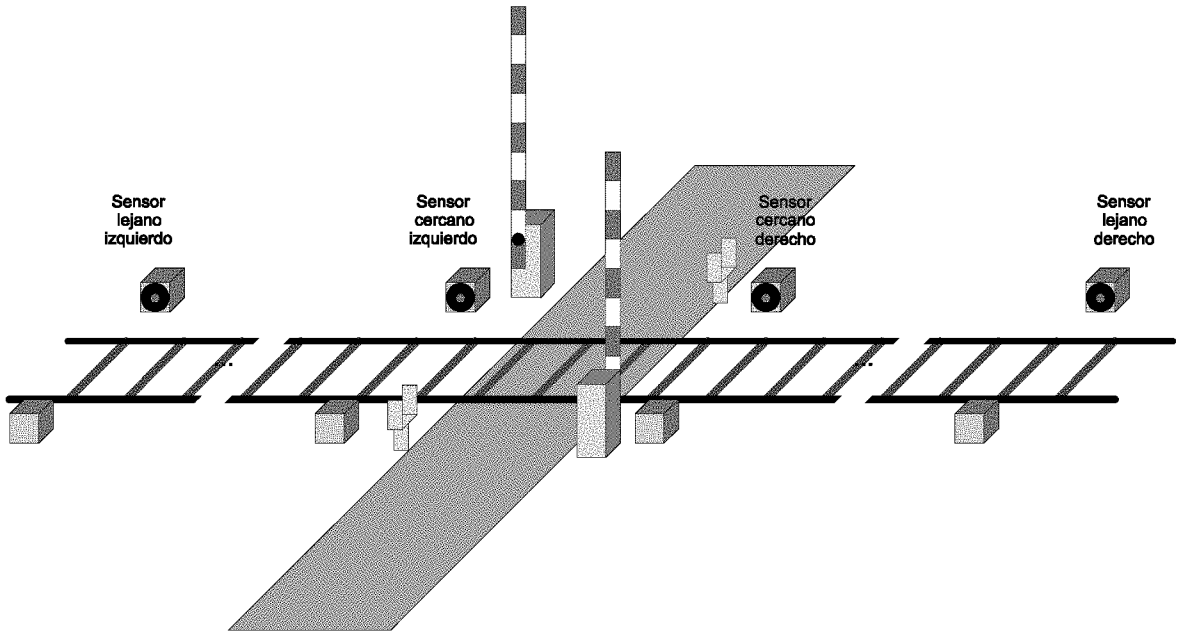


Figura 17. Paso a nivel con barrera.

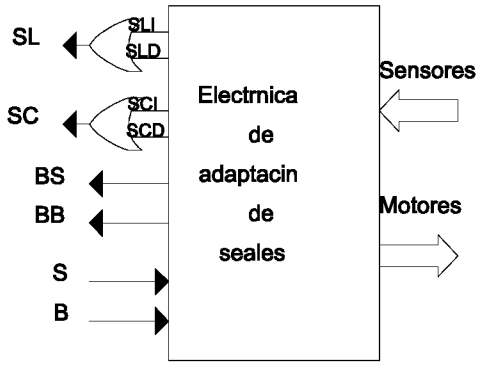


Figura 18. Circuitería de adaptación de señales.

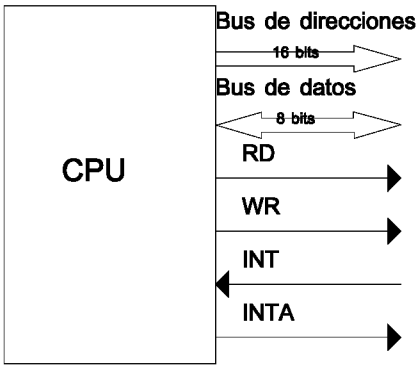


Figura 19. CPU y buses.

En la vía existen dos sensores lejanos situados a una distancia adecuada del paso a nivel, y dos sensores cercanos colocados junto al paso, para detectar el paso del tren. Las barreras disponen de sensores para determinar si se encuentran totalmente subidas y totalmente bajadas.

Para el control de las barreras se dispone de una circuitería de adaptación (Figura 18) que genera y acepta las siguientes señales:

SL (**S**ensor **L**ejano)	Se pone a 1 cuando alguno de los sensores lejanos detecta el principio del tren (siempre que el tren se acerca al paso a nivel, nunca cuando se aleja) y se vuelve a poner a 0 cuando el sensor detecta el final del tren.
SC (**S**ensor **C**ercano)	Se ponen a 1 cuando alguno de los sensores cercanos detecta el principio del tren y se vuelve a poner a 0 cuando detecta el final del tren.
BS (**B**arreras **S**ubidas)	Están a 1 siempre que las dos barreras estén completamente subidas o bajadas, respectivamente.
BB (**B**arreras **B**ajadas)	
S (**S**ubir barreras)	Mientras estén a 1 funciona el motor encargado de subir o bajar, respectivamente, las barreras. Habrá que ponerlos a 0 cuando las barreras lleguen al tope superior o inferior, respectivamente, para no forzar los motores.
B (**B**ajar barreras)	

Se desea usar la CPU de la Figura 19 para controlar el sistema, esto es, bajar las barreras cuando se acerque un tren y subirlas cuando haya rebasado completamente el paso a nivel. La CPU acepta hasta 256 interrupciones vectorizadas. Los vectores de interrupción se sitúan en la parte inferior de la memoria, en direcciones pares (cada vector contiene una dirección de 16 bits). La CPU acepta una interrupción cuando se detecta un flanco de subida en la señal INT, respondiendo con un pulso INTA, durante el cual se lee el número de vector de interrupción por el bus de datos.

a) Diseñe la lógica de decodificación e interfaz necesaria para conectar las señales anteriores a la CPU, suponiendo que se va a usar únicamente E/S programada.

b) Suponiendo que se desea dedicar la CPU también a otras tareas, modifique el diseño del apartado *a)*, de tal modo que se tengan dos fuentes de interrupción, SL y SC, la primera prioritaria sobre la segunda, que pueden interrumpir a la CPU a través de INT, cada una con un vector de interrupción diferente.

c) Realice los diagramas de flujo para cada una de las dos ISRs correspondientes al apartado *b)*, y escriba el programa en ensamblador de la primera (SL).

11. Un computador consta de una CPU y un dispositivo de E/S conectados a M.P. por medio de un bus común de una palabra. La CPU puede ejecutar, como máximo, 10 MIPS. Una instrucción necesita, por término medio, 5 ciclos máquina y durante tres de ellos utiliza el bus de memoria. Una operación de lectura o escritura en M.P. lleva un ciclo. Estime la velocidad de transferencia de datos (palabras/s) en los dos siguientes supuestos:

a) El computador tiene 5 tareas en ejecución que no realizan E/S por programa. Se arranca una nueva tarea para realizar E/S por programa. Los recursos del ordenador se reparten equitativamente entre las tareas. Cada transferencia de un dato requiere dos instrucciones.

b) Una vez arrancado el periférico, éste transfiere sus datos accediendo directamente a memoria, realizando la transferencia por bloque.

12. ***a)*** Dibuje un esquema de un sistema de E/S por DMA que conecte un microprocesador de 8 bits (bus de datos de 8 bits y bus de direcciones de 16 bits) que usa E/S con correspondencia en memoria, la memoria principal y una impresora. El sistema incluirá:

- Una interfaz paralela con un puerto de salida de un byte, que se comunica con la impresora a través de un bus de datos de 8 bits.
- Un controlador de DMA con los registros habituales.

b) Escriba en lenguaje ensamblador la rutina de inicialización del DMA para imprimir un bloque de 200 caracteres (bytes) almacenados a partir de la dirección 0x3000 de memoria.

13. Un computador de 32 bits tiene dos canales selectores y un canal multiplexor. Cada canal selector soporta dos unidades de disco y dos unidades de cinta magnética. El canal multiplexor tiene dos impresoras de líneas, dos lectoras de tarjetas y diez terminales CRT conectados a él. Supongamos que las velocidades de transferencia para estos periféricos son las siguientes:

Disco: 800 KB/s
Cinta: 200 KB/s
Impresora: 6.6 KB/s
Lectora: 1.2 KB/s
CRT: 1 KB/s
Estime la máxima velocidad de E/S conjunta de este sistema.

14. Se pretende leer y procesar bloques de *N* bytes desde un dispositivo de entrada orientado a caracteres utilizando una técnica de *doble buffer*. El dispositivo genera un breve pulso a 1 por una línea de 1 bit cuando dispone de un nuevo dato de 8 bits.

a) Diseñe la interfaz de entrada necesaria para conectar el dispositivo a una CPU con E/S mapeada en memoria, bus de direcciones de 32 bits, bus de datos de 8 bits, señal de control R/W#, y entrada de interrupción INT. Esta interfaz ha de contar adicionalmente con:

• un puerto de estado de 1 bit que proporcione a la CPU la señal de *dato listo* en el caso de que se utilice entrada programada con consulta de estado, y

• un puerto de control de 1 bit que permita habilitar la generación de interrupciones por parte del dispositivo para realizar entrada por interrupciones.

b) Escriba en lenguaje de alto nivel (se recomienda C) un programa y una *ISR* (rutina de servicio de interrupción) que permitan realizar la técnica de *doble buffer*. Para ello se ha de contar con dos buffers (0 y 1) de *N* bytes cada uno. El programa inicialmente leerá mediante E/S programada con consulta de estado el bloque 0. A continuación entrará en un bucle infinito en el que se llamará a una función dada `ProcesarBuffer(int NumBuffer)` encargada de procesar un buffer completo ya leído mientras la *ISR* va realizando la transferencia de caracteres a memoria. Debe tenerse en cuenta que el periférico puede ser más lento que `ProcesarBuffer()`, en cuyo caso no se debe entrar de nuevo en esta función hasta que no termine de llenarse el siguiente buffer. También puede preverse el caso contrario, es decir, que `ProcesarBuffer()` termine después de que haya finalizado la transferencia del siguiente buffer, en cuyo caso no se almacenarán nuevos caracteres en un buffer hasta que no termine el procesamiento de ese buffer.

15. Una planta industrial usa varios sensores para monitorizar temperatura, presión y otros factores. La salida de cada sensor consiste en un conmutador ON / OFF, y ocho de estos sensores han de ser conectados al bus de un pequeño ordenador. Diseñe una interfaz apropiada de forma que el estado de los ocho sensores pueda leerse como un único byte en la dirección $FE10_{16}$. Diseñe también la interfaz apropiada para conectar un indicador de siete segmentos (cuya entrada está codificada en 4 bits) como dispositivo de salida en la dirección $FE11_{16}$.

16. Sea el sistema de adquisición de datos analógicos de la Figura 20 compuesto por los siguientes módulos:

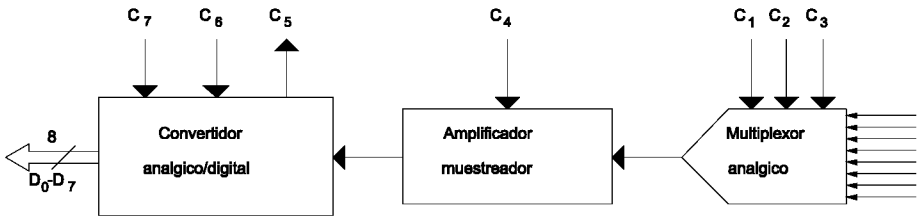


Figura 20. Sistema de adquisición del problema 16.

- Multiplexor analógico. Tiene 8 entradas y una salida analógicas. La salida está gobernada por tres señales de control (C_1, C_2 y C_3).

- Amplificador muestreador (SH). Toma una muestra de la entrada analógica y la mantiene a su salida. Se gobierna mediante una señal de control (C_4), que normalmente está a 1 y que debe pasar a 0 durante un período de 10 µs para realizar la toma de la señal, tal y como muestra la Figura 21.

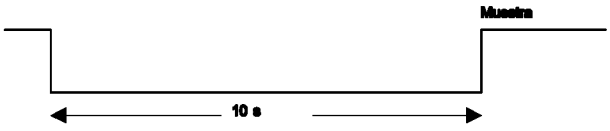


Figura 21. Señal C_4.

- Convertidor analógico digital de 8 bits. Produce una palabra de 8 bits proporcional a la entrada analógica. Se gobierna mediante la señal de control C_6, que debe permanecer a 0 durante el tiempo de la conversión, que dura de 10 a 25 µs. El convertidor contesta a la señal C_6 mediante la señal C_5, que pasa a 1 cuando el convertidor completa la conversión. La lectura del valor digital se hace mediante la señal C_7, que activa los 8 buffers triestado (D_0-D_7) del convertidor y que hace un reset de la señal C_5.

a) Diseñe un sistema de E/S basado en interrupciones vectorizadas (cada vez que se completa una conversión ha de producirse una interrupción) para gobernar este sistema de adquisición de datos, partiendo de las señales del microprocesador de la Figura 22. Incluya en el diseño la lógica necesaria para la lectura del vector de interrupción.
b) Suponiendo que todas las instrucciones tienen una duración de 1 µs, escriba la rutina de servicio de interrupción que permita leer el valor convertido (dejando el resultado en el registro R2) y prepare una nueva conversión de la entrada analógica especificada en el registro R1.

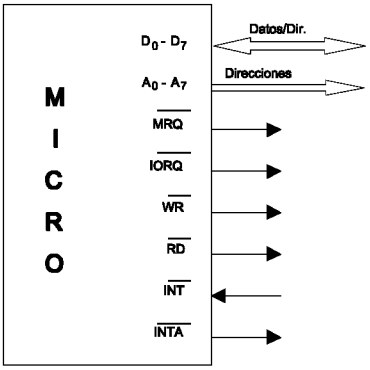


Figura 22. Microprocesador.

17. Un ordenador tiene instrucciones que requieren dos ciclos del bus, uno para captar la instrucción y otro para captar los datos. Cada ciclo del bus toma 250 ns y cada instrucción toma 500 ns (es decir, el procesamiento interno es despreciable). El ordenador tiene un disco con 16 sectores por pista, cada uno de 512 bytes. El tiempo de rotación del disco es 8,192 ms. Cuando se transfieren datos entre disco y memoria por DMA se utiliza robo de ciclo, y el disco transfiere datos a su máxima velocidad. ¿A qué porcentaje de su velocidad normal (sin transferencias con disco) se reduce la velocidad del ordenador durante la transferencia por DMA de un gran bloque de datos entre disco y memoria si cada transferencia simple por DMA tarda un ciclo del bus? Considere dos casos: transferencias de 8 bits y de 16 bits de ancho en el bus.

18. Disponemos de un microprocesador de 8 bits (bus de datos de 8 bits y bus de direcciones de 16 bits) con E/S independiente. Diseñe un sistema de E/S que permita acceder a los siguientes puertos: puerto 0x0020 de entrada y 0x0021 de salida. Utilice lógica de decodificación distribuida. No emplee decodificadores.

19. Un controlador de E/S posee un búfer para el almacenamiento temporal de los datos con una capacidad de 256 KB. En un instante determinado inicia una operación de E/S con una impresora a una velocidad de transferencia de 256 KB/s. Si el controlador de E/S recibe la información que debe enviar a la impresora a una velocidad de 1 MB/s, ¿cuánto tiempo tardará en llenarse por primera vez el búfer suponiendo que inicialmente está vacío, y que recibe y envía información simultáneamente de forma continua?

Grado en Ingeniería Informática
ESTRUCTURA DE COMPUTADORES
Relación de problemas de
Memoria

1. Suponga una jerarquía de memoria de dos niveles, M_1 y M_2, con capacidades s_1 y s_2 bits, tiempo de acceso a los circuitos de cada nivel t_1 y t_2, y coste por bit c_1 y c_2, respectivamente. Obtenga:

a) El costo por bit de toda la memoria.

b) El tiempo medio de acceso.

c) La eficacia de la jerarquía de memoria en términos de la razón de velocidad del nivel 2 respecto al 1.

2. (0,7) Considere una jerarquía de memoria de dos niveles, M_1 (más cercano al procesador) y M_2, con tiempos de acceso t_1 y t_2, costes por byte c_1 y c_2, y tamaños s_1 y s_2, respectivamente. La razón de aciertos de M_1 es $A_1 = 0{,}9$.

a) (0,1) Escriba una fórmula mostrando el coste total C_{tot} del sistema de memoria, sin tener en cuenta el coste de conexión de los dos niveles de la jerarquía.

b) (0,2) Escriba una fórmula que modele el tiempo de acceso efectivo T_{ef} del sistema de memoria, teniendo en cuenta que el acceso a M_1 y M_2 no se puede llevar a cabo simultáneamente.

c) (0,4) Suponga que $t_1 = 20$ ns, t_2 es desconocido, $s_1 = 512$ KB, s_2 es desconocido, $c_1 = 0{,}00015$ € / byte, $c_2 = 0{,}0000006$ € / byte.

i) ¿Cuántos GB de M_2 (s_2 = ?) se pueden adquirir sin exceder un presupuesto total aproximado de unos 400 €, si se desprecia el coste de la conexión de los dos niveles de la jerarquía?

ii) ¿De qué velocidad hay que comprar la memoria M_2 (t_2 = ?) para conseguir un tiempo de acceso medio $T_{ef} = 30$ ns en el sistema de memoria completo bajo la suposición de tasa de aciertos anterior?

3. Teniendo en cuenta que la gráfica de la Figura 1 pretende estar relacionada con el pseudocódigo siguiente, etiquete los ejes de coordenadas, rodee con un círculo los puntos que le parezcan relacionados, etiquételos según la nomenclatura del pseudocódigo e indicando qué tipo de localidad se para cada caso, y deduzca el valor de N. (*Conteste en este mismo folio escribiendo sobre la Figura 1*).

```
Para i=1...N a[i] = i * i
Para j=1...N b[j] = j * j * j
Para k=1...N c[k] = a[k] + b[k]
```

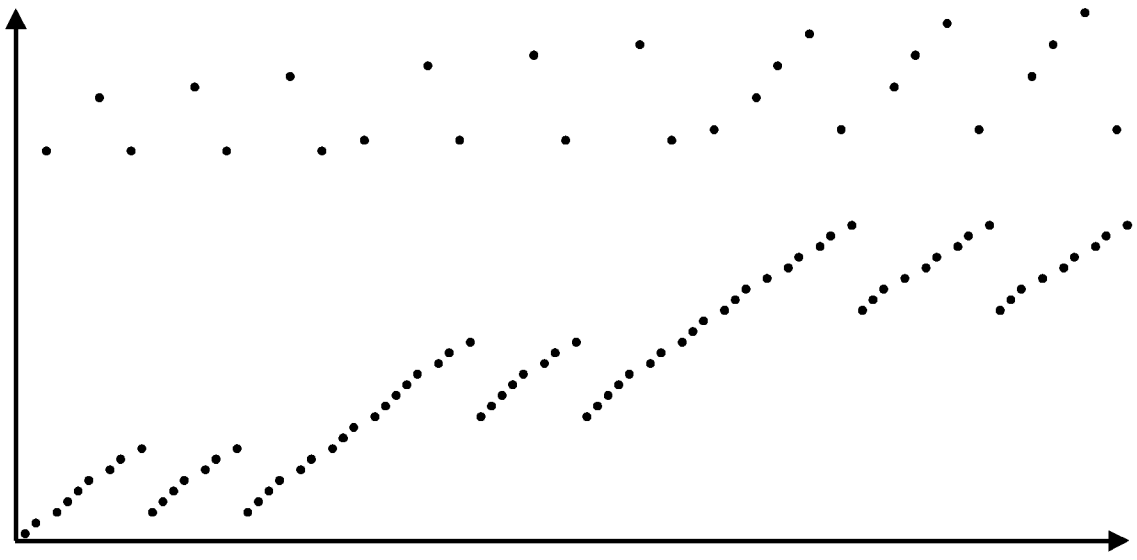


Figura 1. Gráfica del problema 3.

4. Un computador con un bus de datos de 32 bits usa circuitos de memoria RAM dinámica de 1M × 4.

a) ¿Cuál es la memoria más pequeña (en bytes) que puede tener ese computador?

b) Dibuje un esquema de la misma, detallando las líneas de direcciones y de datos.

5. Una placa madre de un IBM PC basada en el microprocesador Intel 8088 dispone de 256 KB de DRAM con 1 bit de paridad, constituida por circuitos integrados 4164 (Figura 2). Las señales que conectan esta memoria con los buses y otros circuitos (circuitería de refresco de memoria y de generación/ comprobación de paridad) se muestran en la Figura 3 (PAR_IN y PAR_OUT son las señales de paridad).

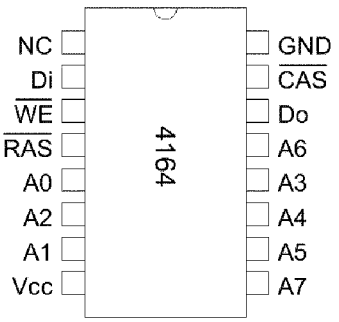


Figura 2. Circuito DRAM 4164 (Di = Data in, Do = Data out, NC = No conectada).

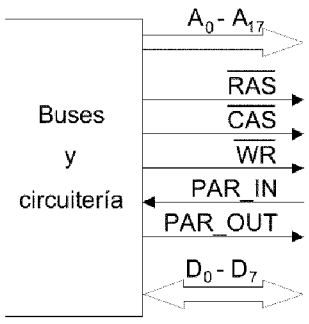


Figura 3. Señales para el sistema de memoria DRAM.

a) Dibuje el sistema de memoria y su conexionado con las señales de la Figura 3. No incluya ninguna circuitería de refresco.

b) Especifique en el dibujo del apartado (a) o en una tabla las direcciones de memoria asociadas a cada chip o grupo de chips. Indique asimismo en qué chips y en qué posición dentro de esos chips se encuentra almacenado el byte de memoria cuya dirección es 20001h.

6. Una placa madre de un PC basada en el microprocesador Intel 8088 disponía de 512 KB de DRAM más bit de paridad, constituida por circuitos integrados 41256 (Figura 4). Las señales que conectaban esta memoria con los buses y otros circuitos (circuitería de refresco de memoria y de generación/ comprobación de paridad) se muestran en la Figura 5 (PAR_IN y PAR_OUT son las señales de paridad).

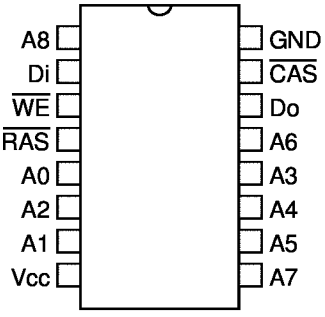


Figura 4. Circuito DRAM 41256 (Di = Data in, Do = Data out).

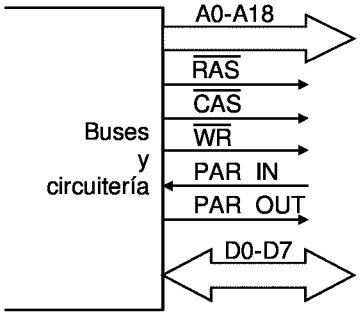


Figura 5. Señales para el sistema de memoria DRAM.

Dibuje el sistema de memoria completo (todos los chips) y su conexionado con las señales de la Figura 5. No hay que incluir la circuitería de refresco.

Especifique en el dibujo el rango de direcciones de memoria asociado a cada chip.

7. Suponga que dispone de circuitos integrados de memoria SRAM de 32K×8 (Figura 6) y desea construir con ellos una memoria de 256 K palabras de 32 bits, organizada con entrelazado de orden inferior con bancos de 64 K palabras.

a) Dibuje el esquema del sistema de memoria, detallando las líneas de dirección.

b) Indique en el dibujo en qué chips y en qué posición dentro de esos chips se encuentra almacenada la palabra cuya dirección es 20001h (el direccionamiento de memoria se realiza a nivel de palabras de 32 bits).

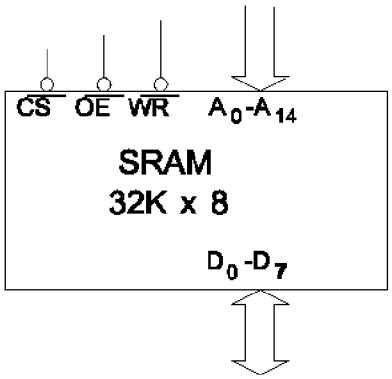


Figura 6. Circuito SRAM.

8. La placa madre de un procesador de 32 bits (486) dispone de 4 MB de memoria DRAM distribuida en módulos SIMM de 30 contactos sin paridad de 1M x 8. Cada SIMM contiene 8 chips de DRAM. Aparte de 9 contactos para alimentación, tierra y no conectados, cada SIMM tiene los siguientes contactos:

A0-A9: Entradas de dirección.
RAS#: Row Address Strobe.

CAS#: Column Address Strobe.
WE#: Write Enable.
DQ1-DQ8: Entradas/salidas de datos.

a) ¿Qué tamaño tiene cada chip de DRAM?

b) Dibuje un esquema de uno de los SIMM con las conexiones entre sus contactos y las patillas de los chips de DRAM.

c) Dibuje un esquema con todos los SIMM (sin dibujar el interior de cada uno) y su conexión con los buses de direcciones y datos del sistema.

9. Un computador dispone de 16 MB de memoria principal entrelazada de orden inferior y acceso simultáneo (Tipo S), constituida por módulos de 1 M Byte. Suponga que el procesador desea acceder a los bytes cuyas direcciones son:

AB0010h, AB0021h, 0010AAh, 2227AAAh, 0101AAh,01016Ah,010163h

a) ¿En qué módulo se encuentra cada uno de esos bytes?

b) ¿Cuántos accesos simultáneos a memoria se necesitan como mínimo?

10. Se tiene un computador con una memoria principal entrelazada con esquema de entrelazado de orden inferior con *latch* en las entradas (acceso C) de 256 K palabras dividida en 4 módulos.

a) ¿Cuáles son las cinco primeras direcciones de cada módulo?

b) Comparar la velocidad de acceso a la memoria para peticiones de acceso espaciadas uniformemente 3 y 2 direcciones empezando en la dirección 5, con la de acceso a un sólo módulo. Dibujar un esquema de tiempos.

11. En un computador que dispone de cache, las instrucciones necesitan 8.5 ciclos de reloj para ejecutarse en caso de que no haya falta. Si la tasa de faltas de la cache es del 11%, cada falta da lugar a un incremento de tiempo de 6 ciclos de reloj, y por término medio cada instrucción necesita 3 accesos a memoria. ¿Cuál debe ser el número medio de ciclos de una instrucción (CPI) en el computador sin cache para que sea beneficioso utilizarla?

12. Suponga que el tiempo de ejecución de un programa es directamente proporcional al tiempo de acceso a instrucciones, y que el acceso a una instrucción en la caché es ocho veces más rápido que el acceso a una instrucción en la memoria principal. Suponga que una instrucción pedida por el procesador se encuentra en la caché con una probabilidad de 0.9, y suponga también que si una instrucción no se encuentra en la caché primero debe ser captada desde la memoria principal a la caché y a continuación captada de la caché para ser ejecutada.

a) Calcule la eficiencia como la relación entre el tiempo de ejecución de un programa sin la caché y el tiempo de ejecución con la caché.

b) Siempre que se dobla el tamaño de la caché, la probabilidad de no encontrar en ella una instrucción buscada se reduce a la mitad. Dibuje una gráfica que represente la eficiencia, tal como se define en el apartado *a)*, en función del tamaño de la caché, indicando cual es la eficiencia mínima (tamaño 0) y máxima (tamaño $\rightarrow \infty$).

13. El tiempo de acceso a la memoria cache de un sistema es de 50 ns, y el tiempo de acceso a la memoria principal es de 500 ns. Se estima que el 80% de las peticiones de acceso a la memoria son para lectura y el resto para escritura. La razón de aciertos es de 0.9.

a) Determinar el tiempo de acceso promedio considerando sólo los ciclos de lectura.

b) Determinar el tiempo de acceso promedio considerando también los ciclos de escritura.

14. Un computador con una memoria principal de 1 M palabra dispone de una memoria cache de correspondencia directa de 4 K palabras con marcos de bloque de 16 palabras.

Suponiendo que la memoria cache se encuentra inicialmente "vacía":

a) Indique el número de faltas de cache que se producen si el procesador genera la secuencia de accesos a

memoria:

ABC13h, CDC14h, ABC1Fh, AB305h, CAC13h, CDC1Ah, CA00Fh, ABC10h

b) Para esa misma secuencia, indique los marcos de bloque de cache en los que se carga cada posición de memoria principal a la que se pretende acceder.

15. Un computador direccionable por bytes tiene una pequeña caché de datos capaz de almacenar 8 palabras de 32 bits. Cada bloque de caché consiste en una palabra de 32 bits. Al ejecutar un determinado programa, el procesador lee datos de la siguiente secuencia de direcciones (en hexadecimal):

200, 204, 208, 20C, 2F4, 2F0, 200, 204, 218, 21C, 24C, 2F4

Este patrón se repite cuatro veces en total.

a) Muestre los contenidos de la cache al final de cada pasada a ese bucle si se usa correspondencia directa. Calcule la tasa de aciertos para este ejemplo. Asuma que la caché está inicialmente vacía.

b) Repita la parte *a)* para una correspondencia completamente asociativa que usa el algoritmo de reemplazo LRU.

c) Repita la parte *a)* para una caché asociativa por conjuntos de cuatro vías.

16. Un computador tiene una cache de 128 bytes. Usa correspondencia asociativa por conjuntos, de cuatro vías, con 8 bytes en cada bloque. El tamaño de la dirección física es 32 bits, y la unidad mínima direccionable es 1 byte.

a) Dibuje un diagrama que muestre la organización de la cache e indique como se relaciona una dirección física con la cache (campos).

b) ¿En qué marcos de bloque de la cache puede residir el byte de memoria principal cuya dirección es 000010AFh?

c) Si las direcciones 000010AFh y FFFF7A*XY*h pueden asignarse simultáneamente al mismo conjunto de cache, ¿cuántos posibles valores distintos puede tener *XY* en la segunda dirección?

17. Suponga una memoria cache de 4 K palabras asociativa por conjuntos, con marcos de bloque de 64 palabras y 8 conjuntos, y una memoria principal de 8 M palabras.

a) ¿ En qué posición de cache se situaría la palabra de memoria principal cuya dirección es 3A0A39h ?

b) ¿Qué direcciones de memoria principal no pueden encontrarse en cache al mismo tiempo que esa dirección?

18. Un sistema de cómputo tiene una memoria principal de 32 K palabras de 16 bits. También tiene una memoria cache de 4 K palabras organizadas de forma asociativa por conjuntos con 4 bloques por conjunto y 64 palabras por bloque.

a) Calcúlese el número de bits de cada uno de los campos Etiqueta, Conjunto y Palabra.

b) Si el tiempo de acceso a la memoria principal es 10 veces mayor que el de la memoria cache, y la razón de aciertos de la cache es de 0.9, determinar la eficiencia de acceso (razón entre el tiempo de acceso a la memoria cache y el tiempo medio de acceso).

19. Un ordenador dispone de 32 KB de memoria principal direccionable por bytes y una memoria cache completamente asociativa de 4 KB. El tamaño del bloque de la memoria cache es de 8 palabras de 32 bits. El tiempo de acceso a la memoria principal es 10 veces mayor que el de la memoria cache.

a) ¿Cuántos comparadores hardware se necesitan?

b) ¿Cuál es el tamaño del campo identificador?

c) Si se utiliza el esquema de sustitución directa, ¿cuál sería el tamaño del campo identificador?

d) Suponer que la eficiencia de acceso se define como la razón entre el tiempo de acceso con memoria cache y el tiempo de acceso sin memoria cache. Determinar la eficiencia de acceso suponiendo que la razón de aciertos de la memoria cache es 0.9.

e) Si el tiempo de acceso a la memoria cache es de 200 ns, ¿cuál será la razón de aciertos necesaria para lograr un tiempo de acceso de 500 ns?

20. Los parámetros que definen la memoria de un computador son los siguientes:

- Tamaño de la memoria principal: 32 K palabras.
- Tamaño de la memoria cache: 4 K palabras.
- Tamaño de bloque: 64 palabras.

Determine el tamaño de cada campo de una dirección de memoria y explique brevemente cómo se obtiene, para las siguientes políticas de colocación:
a) Totalmente asociativa.
b) Por correspondencia directa.
c) Asociativa por conjuntos con 16 bloques por conjunto.

21. Los parámetros que definen la memoria de un computador son los siguientes:

- Tamaño de la memoria principal: 4 G bytes.
- Tamaño de la memoria cache: 1 MB.
- Tamaño de bloque: 256 bytes.

Determine el tamaño de cada campo de una dirección de memoria desde el punto de vista de la cache, para las siguientes políticas de colocación:
a) Totalmente asociativa.
b) Por correspondencia directa.
c) Asociativa por conjuntos con 4 bloques por conjunto.

22. Los parámetros que definen la memoria de un computador son los siguientes:

- Tamaño de la memoria principal: 4 G bytes.
- Tamaño de la memoria cache: 16 MB.
- Tamaño de bloque: 256 bytes.

Determine el tamaño de cada campo de una dirección de memoria desde el punto de vista de la cache, para las siguientes políticas de colocación:
a) Totalmente asociativa. ***b)*** Directa. ***c)*** Asociativa por conjuntos, de 8 vías. ***d)*** Por sectores con 16 bloques por sector.

23. Una memoria cache asociativa por conjuntos consiste en un total de 64 bloques divididos en conjuntos de 4 bloques. La memoria principal contiene 4096 bloques. Cada bloque contiene 128 palabras.
a) Indicar el número de bits que hay en una dirección de memoria principal.
b) Indicar el número de bits que hay en cada uno de los campos Marca, Conjunto y Palabra.

24. Un ordenador dispone de 32 K palabras de memoria principal y una memoria cache con colocación asociativa por conjuntos. El tamaño de un bloque es de 16 palabras y el campo identificador de 5 bits. Si la misma memoria cache se sustituye directamente, el campo identificador tendría una longitud de 3 bits. Determinar el número de palabras que alberga la memoria cache. Determinar el número de bloques que alberga un conjunto de la memoria cache.

25. Los parámetros que definen la memoria de un computador son los siguientes:

- Tamaño de la memoria principal : 8 K líneas
- Tamaño de la memoria cache : 512 líneas
- Tamaño de la línea : 8 palabras

Determinar el tamaño de los distintos campos de una dirección en las siguientes condiciones:
a) Colocación completamente asociativa.
b) Colocación directa.
c) Colocación asociativa por conjuntos con 16 líneas por conjunto.
d) Colocación por sectores con 16 líneas por sector.

26. Sea un sistema de memoria con cache, con las siguientes características:

Tamaño de la memoria principal:	4 GB
Tamaño de la memoria cache:	256 KB
Tamaño de palabra (anchura bus de datos):	32 bits
Correspondencia:	Asociativa por conjuntos
Nº de bloques / líneas por conjunto (nº de vías):	4
Nº de palabras por bloque / línea:	16

Dibuje un esquema detallado de la memoria cache y su conexión con la CPU para una de las dos alternativas que se indican más abajo. No incluya la MP ni la interfaz entre cache y MP (circuitería necesaria para transferir bloques entre cache y MP).

a) Puede utilizar 1024 memorias asociativas de 4 × 16, un decodificador de 10 a 1024, 1024 codificadores de 4 a 2, 1024 puertas OR de 4 entradas y 4096 circuitos SRAM de 64 × 8.

b) Puede utilizar 8 circuitos SRAM de 1K × 8, 4 comparadores de dos entradas de 16 bits y 16 circuitos SRAM de 16K × 8.

27. Un ordenador usa una pequeña cache, con correspondencia directa, entre la memoria principal y la CPU. La cache tiene 4 marcos de bloque con dos palabras de 16 bits cada uno. Cada bloque tiene asociada una etiqueta de 13 bits. Cuando ocurre una falta durante una operación de lectura, el bloque que contiene la palabra pedida se lee de la memoria principal y se envía a la cache, y su número de bloque se almacena simultáneamente en la etiqueta asociada. Después, se lleva la palabra de la cache a la CPU. Considere la siguiente subrutina de un programa cuyas instrucciones y datos tienen 16 bits:

```
loop: add R0,[R1++]      ; R0 es el destino
      dec R2
      bnz loop
      ret
```

Suponga que, antes de que se entre en el bucle, los registros R0, R1 y R2 contienen 0, 054Eh y 3, respectivamente, y la cache está vacía. También suponga que las direcciones de memoria principal 054Eh, 054Fh, 0550h y 0551h contienen los datos A03Ch, 05D9h, 10D7h y B6ECh. El bucle comienza en la posición loop = 02ECh.

a) Muestre los contenidos de la cache cada vez que ésta se modifica.

b) Suponga que el tiempo de acceso de la memoria principal es 10τ (tiempo para transferir un bloque entre memoria principal y cache) y el de la cache es τ. Escriba una expresión teórica para el tiempo de acceso promedio a la cache considerando sólo la lectura y sin tener en cuenta el reemplazo de bloques. Para el programa anterior, indique el número de accesos a memoria y la tasa de aciertos. Calcule entonces el tiempo total de ejecución estimado por la expresión teórica e indique si es el mismo que el experimental (ignore el tiempo que consume la CPU entre ciclos de memoria).

28. Dada la organización de memoria cache que se muestra en la Figura 7:

a) ¿Qué tipo de correspondencia utiliza?

b) Determine el número de palabras y de bloques que almacena la cache.

c) Nombre cada uno de los campos de una dirección de memoria principal y determine su tamaño.

d) ¿Qué son y cuál es la utilidad de los dos triángulos?

e) ¿Para qué sirve la puerta OR?

f) ¿Cómo se puede utilizar el bit LRU? ¿Por qué basta con un único bit?

g) Explique brevemente cómo se busca un dato en la cache.

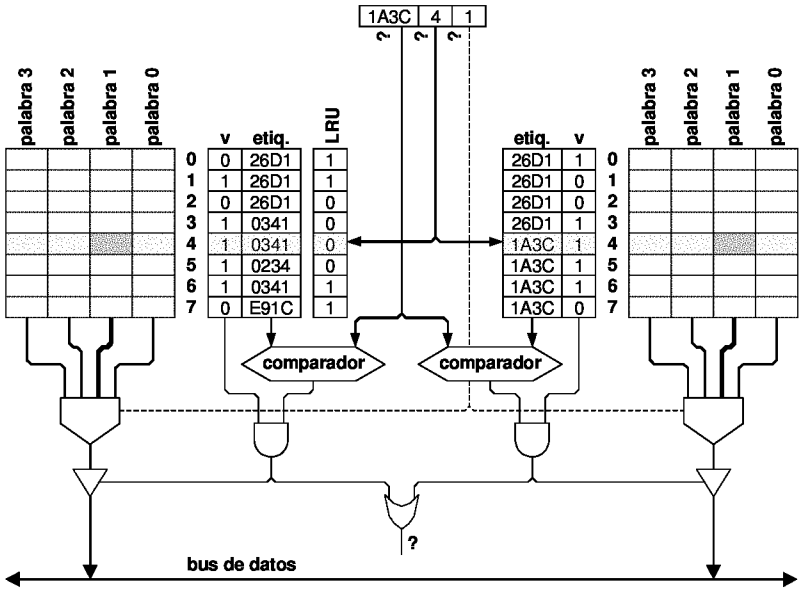


Figura 7. Organización de memoria cache del problema 28 (sólo está representada la lógica de lectura).

29. Un computador direccionable por palabras tiene una pequeña caché de datos e instrucciones capaz de almacenar 16 palabras. Cada bloque de caché consiste en dos palabras. Al ejecutar un bucle de 8 instrucciones (cada instrucción ocupa una palabra), el procesador accede a la memoria según el siguiente patrón:

Instrucción dentro del bucle	Dirección de la instrucción	Dirección del dato accedido por la instrucción
i1 (acceso a datos)	0012h	FF03h
i2 (salto a i5)	0013h	-
i5 (acceso a datos)	0016h	FF07h
i6 (salto a i9)	0017h	-
i9 (acceso a datos)	001Ah	FF0Bh
i10 (salto a i13)	001Bh	-
i13 (acceso a datos)	001Eh	FF0Fh
i14 (salto a i1)	001Fh	-

El bucle se repite 1000 veces en total.

a) Muestre el contenido de la cache al final de la última pasada a ese bucle si se usa correspondencia directa. Calcule la tasa de aciertos para este ejemplo. Asuma que la caché está inicialmente vacía.

b) Repita el apartado *a)* para una correspondencia asociativa por conjuntos de dos vías.

30. Un procesador direcciona la memoria principal por bytes (esto quiere decir que cuando se incrementa en 1 una dirección, se direcciona el objeto del siguiente byte). La lectura o escritura de una palabra de 32 bits alineada requiere un único acceso a memoria, mientras que si no está alineada se necesitan dos accesos. El procesador dispone de una pequeña caché de datos capaz de almacenar 4 palabras de 32 bits cada una. Cada bloque de caché consiste en una palabra de 32 bits. Al ejecutar un determinado programa, el procesador lee palabras de 32 bits de las siguientes direcciones (en hexadecimal):

200, 326, 204, 326

Este patrón se repite varias veces.

a) Muestre en una pequeña tabla el contenido de los 16 bytes de la cache al final de cada pasada a ese bucle si se usa correspondencia directa. Calcule la tasa de aciertos para este ejemplo a partir de la segunda iteración del bucle.

b) Repita la parte *a)* para una caché asociativa por conjuntos de dos vías, con reemplazo LRU.

c) Repita la parte *a)* para una caché totalmente asociativa.

31. Suponga que tiene que diseñar un sistema basado en el procesador TMS320C30, que puede direccionar un total de 16 M palabras de 32 bits, y dispone de una memoria cache interna para instrucciones de 64 palabras de 32 bits. Revisando la Guía del Usuario de este circuito encuentra la Figura 8 y los siguientes comentarios:

"*...La cache está dividida en dos segmentos de 32 palabras. Asociado con cada segmento hay un Registro de Dirección de Comienzo del Segmento (SSA). Por cada palabra de la cache hay un bit llamado Indicador de Presencia (P). Cuando la CPU requiere una instrucción, se chequea si la palabra está ya en la cache de instrucciones. Los 19 bits más significativos de la dirección de una instrucción se usan para seleccionar el segmento y los 5 bits menos*

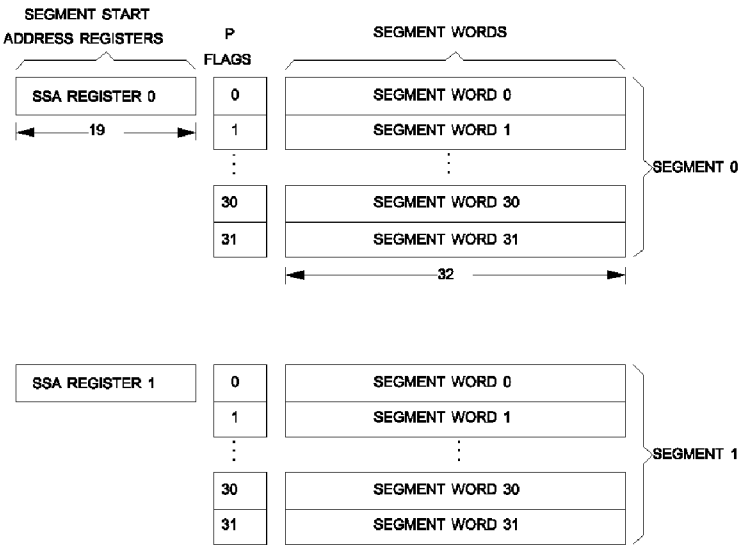


Figura 8. Arquitectura de la cache de instrucciones

SEGMENT START ADDRESS REGISTERS = Registros de dirección de comienzo de segmento, WORD = Palabra, SEGMENT WORD = Palabra dentro de segmento.

significativos definen la dirección de la palabra (instrucción) dentro del segmento. Los 19 MSBs de la instrucción se comparan con los dos registros SSA. Si hay coincidencia, se chequea el indicador P. El indicador P indica si una palabra de un segmento está ya presente en memoria (1) o no (0).
Cuando el TMS320C30 requiere una instrucción de la memoria externa dos posibles eventos pueden ocurrir:
***Acierto de cache**: La instrucción requerida se encuentra en cache y es leída de ella.*
***Falta de cache**: Hay dos tipos de faltas:*
1) Falta de palabra. El registro SSA coincide con la dirección de la instrucción, pero el indicador P no está a 1. Entonces se lee la instrucción de memoria y se copia en la cache, poniendo a 1 el indicador P.
2) Falta de segmento. Ninguno de los registros SSA coincide con la dirección. Se selecciona el segmento menos recientemente usado. Se borran los 32 indicadores P. El registro SSA del segmento seleccionado se carga con los 19 bits más significativos de la dirección requerida. La palabra de la instrucción se trae de memoria a cache, colocándose en la palabra de cache, dentro del segmento seleccionado, indicada por los 5 bits menos significativos de la dirección. El bit P de esa palabra se pone a 1..."
a) Salvo nomenclatura, esta cache usa una política de colocación estudiada en clase. ¿Cuál?.
b) Identifique cada uno de los campos con los nombres conocidos, indicando su tamaño.
(Razone las respuestas)

32. Se dispone de un computador con las siguientes características:

1. El 90% (87,5% para el apartado *b)*) de todos los accesos a memoria se encuentran en la caché interna del procesador.
2. Cada bloque de caché interna consta de dos palabras.
3. El procesador envía referencias a su caché interna a la velocidad de 10^7 palabras por segundo.
4. El 25% de las referencias de (3) son escrituras.
5. El bus del sistema puede soportar 10^7 transferencias (lecturas o escrituras) de palabras por segundo.
6. El bus del sistema lee o escribe una sola palabra cada vez (no puede leer o escribir dos palabras cada vez).

Se está considerando el añadir un periférico al bus del sistema, y se quiere saber qué ancho de banda del bus se está utilizando. Calcule el porcentaje del ancho de banda del bus utilizado en los siguientes cuatro casos:

a) La caché es de escritura directa (*write-through*) con asignación en escritura. No se escriben bloques completos de caché interna a memoria principal sino sólo palabras

b) Igual que *a)* pero escritura directa sin asignación en escritura. Este cambio provoca que la tasa de aciertos baje del 90% al 87,5%.

c) La caché es de postescritura-siempre (*write-back*). Se escriben bloques completos de caché interna a memoria principal.

d) Igual que *c)* pero postescritura-marcada. Suponga que en cualquier instante de tiempo, el 30% de los bloques de la caché interna han sido modificados.

33. Un ordenador tiene un sistema de memoria con las siguientes características:

- Memoria virtual de 16 T bytes con segmentación paginada, que permite direccionar hasta 2^{22} segmentos de 1024 páginas cada uno, siendo el tamaño de la página de 4 KB.
- Memoria principal de 1 G byte.
- Si hay un fallo de segmento, hay que llevar su tabla de páginas de disco a memoria principal. Lo mismo sucede si hay un fallo de página, es decir, hay que llevar la página de disco a memoria principal. Cualquiera de estas dos operaciones consume 20 ms.
- Memoria cache de 256 K Bytes, cuya organización es asociativa por conjuntos. Cada conjunto está formado por 4 bloques de 32 bytes cada uno. El tiempo para acceder al directorio de cache y determinar si la dirección buscada reside o no en cache es de 20 ns, y de 30 ns para leer la palabra buscada. Si hay un fallo de cache, se lleva el correspondiente bloque de memoria principal a cache (operación que consume 200 ns), y a continuación se realiza el acceso a cache (no se accede de nuevo al directorio).
- Para la traducción de direcciones virtuales a físicas se utiliza:
  · Un TLB cuyo tiempo de acceso es de 20 ns.
  · Una tabla de segmentos, que residirá en memoria principal, parte de la cual puede estar en cache.
  · Una tabla de páginas (que ocupa una página) para cada segmento, que puede residir en disco o en memoria principal. Si una tabla de páginas está en memoria principal, parte de ella puede estar

en cache.
- Las probabilidades de acierto son las siguientes:
  TLB: 95%; Memoria cache: 90%; Memoria principal: 99,996%
- Todo el direccionamiento se realiza a nivel de bytes.

a) Determine los campos (así como su longitud) de que están compuestas una dirección de memoria virtual, y una dirección de memoria principal desde el punto de vista de la cache. Indique también cuántos bits se necesitan para direccionar una página física.
b) Dibuje un organigrama (diagrama de flujo) que refleje los pasos a seguir desde que se presenta una dirección virtual al sistema de memoria hasta que la CPU obtiene la información referenciada, considerando todos los casos posibles.
c) Determine el tiempo de acceso medio en lectura en caso de acierto en la TLB.

34. Un ordenador tiene un espacio de direcciones virtual de 16 páginas pero sólo 4 marcos de página. Inicialmente, la memoria está vacía. Un programa referencia páginas virtuales en el siguiente orden:

0, 7, 2, 7, 5, 8, 9, 2, 4

¿Cuáles de esas referencias provocan fallos de página con LRU? ¿Y con FIFO?

35. Un multiprocesador tiene 6 procesadores conectados a 4 módulos de memoria mediante un único bus común con funcionamiento síncrono por ciclo partido. El tiempo de acceso a un módulo de memoria, tanto en lectura como en escritura, es de 3 μs (durante ese tiempo el módulo está ocupado y no puede ser utilizado para otro acceso). El tiempo de ciclo de cada procesador y de cada ranura del bus es de 500 ns. El número de ciclos de utilización del bus para hacer los intercambios de información entre la memoria y los procesadores es:

Operación de lectura	Envío de dirección	1 ciclo
	Lectura del dato	1 ciclo
Operación de escritura	Envío de dirección y dato	1 ciclo
	Recepción de confirmación de dato escrito correctamente	1 ciclo

Sabiendo que de cada siete operaciones en memoria cuatro son de lectura y tres de escritura, calcule el número máximo posible de operaciones en memoria por segundo.

36. Disponemos de un procesador que direcciona la memoria por bytes y dispone de las patillas de datos D0 a D15, las patillas de direcciones A0 a A15, y las señales BHE# (Bus High Enable), MRD#, MWR#, IORD# e IOWR# (el símbolo # significa señal activa a nivel bajo). Diseñe un sistema de memoria para este procesador a partir de módulos SRAM de 8 K × 8 y ROM 8 K × 4. La memoria ROM debe ocupar las direcciones 0x0000 a 0x3FFF y la SRAM 0x4000 a 0xFFFF. El procesador debe poder acceder a una palabra completa en una dirección par en un sólo ciclo del bus poniendo A0 y BHE# a 0, a un byte en una dirección par poniendo A0 a 0 y BHE# a 1, o a un byte en una dirección impar poniendo A0 a 1 y BHE# a 0.

37. Un sistema de memoria contiene una caché, una memoria principal y una memoria virtual. Cuando se accede a caché, en el 80% de los casos se produce un acierto. Cuando, debido a una falta de caché, se tiene que acceder a memoria principal, en el 99,995% de esos accesos (no de todos los accesos del procesador, sino sólo de los que producen falta en caché), el dato está en memoria principal y no hay que acceder a disco. La caché tiene un tiempo de acceso de 5 ns, el tiempo de acceso de la memoria principal es de 100 ns, y el de disco es de 10 ms. ¿Cuál es el tiempo medio de acceso de la jerarquía?

38. Diseñe un sistema de memoria para un 8086 que contenga una SRAM de 64K × 16 en las direcciones E0000 a FFFFF y una EPROM de 32K × 16 en las direcciones 00000 a 0FFFF. Para ello dispone de circuitos SRAM de 32K × 8 cuyas señales de control son WE# (Write Enable), OE# (Output Enable) y CS# (Chip Select), y de circuitos EPROM de 16K × 8 cuyas señales de control son CE# (Chip Enable) y OE# (Output Enable), además de los decodificadores y puertas lógicas que considere convenientes. Tenga en cuenta que el 8086 utiliza las señales A0 y BHE# (Byte High Enable) para activar los bytes bajo y alto, respectivamente.

39. Si se utilizan memorias DRAM con modo página rápido para implementar la memoria principal de un sistema cuya caché usa bloques de 8 palabras, ¿cuántos ciclos del tipo RAS-CAS se ahorran (o sea, cuántos pueden ser sustituidos por ciclos CAS o modo página rápido) cada vez que se copie un bloque de memoria en la caché, suponiendo que los bloques están siempre en una única fila de la DRAM y que ésta devuelve una palabra de datos en cada acceso? Suponga que el tiempo de ciclo RAS-CAS es de 120 ns y el tiempo de ciclo CAS o modo página rápido es de 40 ns. ¿qué porcentaje de tiempo se ahorra por usar modo página rápido frente a no usarlo en la copia de un bloque de memoria principal a caché?

40. El siguiente programa en C calcula los 100 primeros valores de la sucesión de Fibonacci de forma iterativa:

```
const int n = 100;
int fib[n] = {1, 1};
int main()
{
      int i;

      for (i=2; i<n; i++)
            fib[i] = fib[i-2] + fib[i-1];
}
```

Al compilar el programa en un sistema de 64 bits da lugar al código ensamblador siguiente:

```
400510: 31 d2                xor edx,edx
400512: 8b 04 95 40 09 60 00 mov eax,DWORD PTR [rdx*4+0x600940]
400519: 03 04 95 44 09 60 00 add eax,DWORD PTR [rdx*4+0x600944]
400520: 89 04 95 48 09 60 00 mov DWORD PTR [rdx*4+0x600948],eax
400527: 48 83 c2 01          add rdx,0x1
40052b: 48 83 fa 62          cmp rdx,0x62
40052f: 75 e1                jne 400512 [main+0x2]
```

Supongamos que se ejecuta en un procesador que utiliza una caché unificada para datos e instrucciones, con correspondencia directa con 4 marcos de bloque y 8 palabras por bloque. El procesador direcciona la memoria por bytes. El vector `fib` comienza en la dirección 0x600940 y sólo sus dos primeras posiciones están inicializadas a 1. `rdx` es la extensión a 64 bits del registro `edx`. Las operaciones sobre registros de 32 bits ponen a 0 los 32 bits más significativos del equivalente registro de 64 bits; así, la instrucción `xor edx,edx` pone a 0 tanto `edx` como su extensión a 64 bits `rdx`.

a) Muestre el contenido de la caché en cada instante para las 7 primeras instrucciones ejecutadas.

b) ¿Cuál es la tasa de aciertos de la caché tras la ejecución de las primeras 7 instrucciones del programa?