



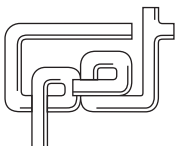
E. T. S. d'Enginyeria de Telecomunicació  
Barcelona

# ARQUITECTURA DE COMPUTADORS I SISTEMES OPERATIUS 1 ( ARISO 1 )

*Col·lecció de Problemes*

M. Jiménez  
B. Otero  
E. Salami  
G. Utrera  
M. A. Villanueva

Departament d'Arquitectura de Computadors



Consulta les novetats al llistat de publicacions a: <http://cpet.upc.edu>

Ref. 12402

# **Col·lecció de Problemes**

## **Arquitectura de Computadors i Sistemes Operatius I**



**Departament d'Arquitectura de Computadors**

M. Jiménez, B. Otero, E. Salamí, G. Utrera, M. A. Villanueva

Curs 2007-2008 (Q1)



# Problemas básicos de ensamblador IA32

## Problema 1

Obtenir els codis de condició (Carry, Zero, Overflow, i Negatiu) i el resultat de les següents operacions amb aritmètica binària c.a.2 amb 8 bits:

```
0x3E-0xA1; 0xA1+0x3E;
0xF6-0xF6; 0xFA-0x11;
17+21; 15+(-18); -12+(-20);
118-47; (-118)+47; (-110)-(-40);
```

## Problema 2

Codifiqueu en llenguatge ensamblador IA32 les següents sentències de control de fluxe expressades en llenguatge de programació d'alt nivell C:

- ```
if (a >= b) {
    ...
}
else {
    ...
}
```
- ```
for (i = 0; i < x; i++) {
    ...
}
```
- ```
while (a == b) {
    ...
}
```

## Problema 3

Escribir un programa en ensamblador del IA32 que cuente el número de bits que valen 1 en el registro EAX. El resultado hay que dejarlo en el registro EBX.

## Problema 4

Escribir un programa en ensamblador del IA32 que calcule la suma de todos los elementos de un vector de N enteros. la dirección del vector está almacenada en el registro EBX. El valor de N está almacenado en el registro ECX. El resultado se guardará en el registro EAX.

## Problema 5

Escribir un programa en ensamblador IA32 que analice un vector de 32 enteros (`v[32]`) y genere una tira de bits (en EAX) en la que el bit  $n$  se codifique con un '1' si el elemento  $n$ -esimo de  $v$  es negativo y con un '0' si es positivo.

```
v[31]v[30]v[29]v[28]v[27]v[26]v[25]v[24]v[23]v[22]v[21]v[20]v[19]v[18]v[17]v[16]
-23  -99 -4   26  18  -78  -97   7  -54  89  178  -1   0  -56  34   2
  1   1   1   0   0   1   1   0   1   0   0   1   0   1   0   0
v[15]v[14]v[13]v[12]v[11]v[10]v[9]  v[8]  v[7]  v[6]  v[5]  v[4]  v[3]  v[2]  v[1]  v[0]
-12  989  -45  -268 178  -78  -97  -679 -54  89  78  -1   0  -56  99  -2
  1   0   1   1   0   1   1   1   1   0   0   1   0   1   0   1
```

## Problema 6

Escribir un programa en ensamblador IA32 que convierta una cadena de caracteres numéricos a un número natural. La cadena de caracteres finaliza con el carácter blanco y está almacenada en posiciones consecutivas de memoria a partir de la dirección simbólica INPUT. El resultado debe quedar almacenado en la dirección simbólica OUTPUT.

## Problema 7

Escriure un programa ensamblador IA32 que trobi els valors màxim i mínim d'un vector de números enters emmagatzemats a memòria a partir de l'adreça simbòlica VECTOR i en posicions consecutives de memòria. L'adreça simbòlica N emmagatzema el nombre d'elements del vector i les adreces simbòliques MAX i MIN guardaran respectivament els valors màxim i mínim trobats pel programa.

## Problema 8

Traducir literalmente el siguiente código escrito en C a ensamblador IA32:

```
int a[50], b[50], sum;

sum=0;
for (i=0; i<50; i++)
    sum = sum + a[i]*b[i];
```

## Problema 9

Traducir literalmente el siguiente código escrito en C a ensamblador IA2 (suponer que las variables i y j se encuentran almacenadas en los registros EDI y ESI, respectivamente):

```
int C[100][50], D[100][50];

C[i][j] = D[i][3] + D[7][j];
```

## Problema 10

Traducir literalmente el siguiente código escrito en C a ensamblador IA32 (suponer que las variables i y j se encuentran almacenadas en los registros EDI y ESI, respectivamente):

```
int A[100][75], B[75][50];
int k, sum;

sum=0;
for (k=0; k<75; k++)
    sum = sum + A[i][k] * B[k][j];
```

## Problema 11

Traducir literalmente el siguiente código escrito en C a ensamblador IA32:

```
int C[100][50], D[100][50];
int i, j;

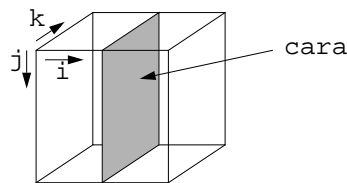
for (i=0; i<100; i++)
    for (j=0; j<50; j++)
        C[i][j] = D[i][j];
```

### Problema 12

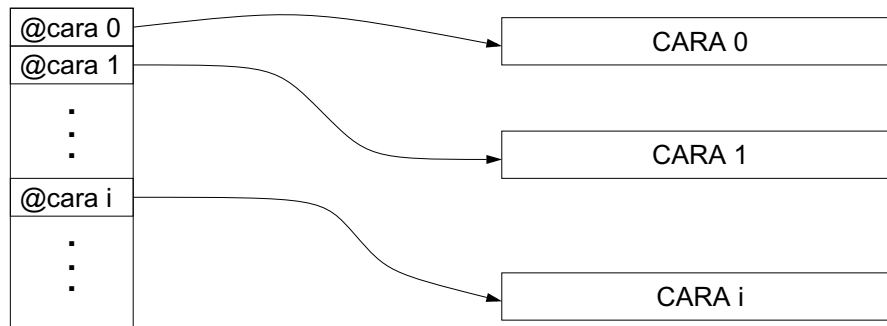
Se dispone de una matriz de  $N \times N$  enteros almacenada por filas a partir de la dirección simbólica A. Escribid las instrucciones necesarias, en lenguaje ensamblador IA32, para copiar, a partir de la dirección simbólica B, los  $N$  enteros de la diagonal de la matriz A.

### Problema 13

Considerar una matriz A de tres dimensiones definida como: `int A[N][N][N];`



La matriz se almacena en memoria por caras en posiciones no consecutivas de memoria. Cada cara se almacena por filas en posiciones consecutivas de memoria tal y como muestra la figura:



Determinar como se puede calcular la dirección del elemento  $A[i][j][k]$  de la matriz.

### Problema 14

Traducir literalmente el siguiente código escrito en C, que realiza la suma de dos matrices, a ensamblador IA32:

```
int A[10][20], B[10][20], C[10][20];
int i;
int *pA, *pB, *pC;

pA=A; pB=B; pC=C;
for (i=0; i<200; i++)
{
    *pC= *pA + *pB;
    pA++; pB++; pC++;
}
```

### Problema 15

Donada la siguiente declaración de variables:

```
typedef struct{
    char a;
    int b[10];
}elem;
elem s[100];
```

Es demana que determineu com es calcula l'adreça de memòria on s'emmagatzemarà la informació:

```
s[s[i].b[j]].a
```

Escriure la seqüència d'instruccions en IA32 per codificar la sentència:

```
x = s[s[i].b[j]].a
```

Considereu que:

- Els elements de l'estructura de dades  $s$  estan emmagatzemats en posicions consecutives de memòria a partir de l'adreça simbòlica  $s$ .
- Un enter ocupa 4 bytes i un caràcter ocupa 1 byte.

## Problemas de subrutinas

### Problema 16

Dadas las siguientes funciones:

```
int intsqr2 ( int i )
{
    return(i * i)
}

int intsqrN( int i, int n )
{
    int k, par, impar, res;

    res = 1 ;
    par = n / 2 ;
    for (k=1; k<=par; k++)
        res = res * intsqr2( i ) ;
    impar = n % 2; /* operació modul */
    if (impar != 0)
        res = res * i ;
    return(res);
}
```

- a) Traducid el programa a lenguaje ensamblador IA32.
- b) Dibujad la situación de la pila durante la ejecución de la sentencia:

```
    return(i * i)
```

### Problema 17

Traducid la siguiente función recursiva a ensamblador IA32:

```
int factorial (int x)
{
    if (x==0 || x==1) return (1);
    else return (x*factorial (x-1));
}
```

### Problema 18

Traduïu literalment aquesta funció escrita en C a llenguatge ensamblador IA3 seguint el model de programació explicat a classe. Dibuixeu també el bloc d'activació que hi ha durant l'execució de la subrutina. Suposeu que la funció `primer` ja la teniu programada.

```
void procediment(int VEC[10], int n, int *suma)
{
    int i, locvec[10];

    for (i = 0; i < n; i++)
    {
        locvec[i] = VEC[i] * 3;
        if (i == 1)
            *suma = *suma + primer(n);
        else
            *suma = *suma + locvec[i];
    }
}
```

## Problema 19

Traduïu a llenguatge ensamblador IA32 el següent algorisme recursiu que multiplica dos números. (Recordeu que els operadors << i >> signifiquen “desplaçament de bits”, que l'operador & significa “AND bit a bit”, i que el prefix 0x indica “constant hexadecimal”).

```
int mult(int a, int b)
{
    int aux;

    if ( b == 0 )
        return 0;
    else {
        aux = mult(a, b>>1) << 1;
        if ( b & 0x0001 != 0 )
            return (aux + a);
        else
            return aux;
    }
}
```

## Problema 20

Traducid la siguiente función recursiva a ensamblador IA32:

```
int A (int i, int *k)
{
    int b;
    if (i <= 0)
        return(0);
    else
    {
        b = *k - 1;
        return(A(A(i-2,&b),&i)+1);
    }
}
```

## Problema 21

Donada la següent funció escrita en C:

```
int calcula (int MAT[10][10], int m, int n)
{
    int i, suma, fila;

    suma = 0;
    fila = 0;
    for (i = m; i < n; i ++ )
        suma = suma + normalitza(MAT[fila][i], &fila);
    return(suma + 1);
}
```

Es demana:

- Dibuixar el bloc d'activació de la funció.
- Traduir **literalment** la funció anterior a llenguatge ensamblador IA32, **seguint el model de programació explicat a classe**. Supposeu que la funció `normalitza` ja la teniu programada.

Notes:

- Recordeu que les matrius en C sempre es passen per referència, i que s'emmagatzemen per files.

## Problema 22

Traduïu literalment aquesta funció escrita en C a llenguatge ensamblador IA32 seguint el model de programació explicat a classe. S'ha de suposar que la funció `modifica` ja està programada, per tant, només la heu de cridar. Dibuixeu també el bloc d'activació que hi ha durant l'execució de la subrutina.

```
int funcio(int MAT[10][10], int n)
{
    int i, nlocal, total;

    total = 1;
    nlocal = modifica(n);
    if (nlocal == 0)
        return (nlocal);
    else
        for (i = 0; i < nlocal; i++)
            total = total * MAT[i][i];
    return (total);
}
```

## Problema 23

Dado el siguiente procedimiento escrito en C:

```
void examen(int a, int b[100], int *c)
{
    int d[100];
    int aux;
    ...
    examen(0, d, &aux);    /* 1 */
    ...
    for (aux=0; aux<100; aux++) b[aux] = d[aux];    /* 2 */
    b[39] = 17;    /* 2' */
    ...
    examen(a,b,c);    /* 3 */
    ...
}
```

Se pide:

- Dibujad la pila (bloque de activación).
- Traducid a lenguaje ensamblador IA32 las sentencias {1}, {2 y 2'} y {3}.

Información adicional:

- Todos los parámetros se pasan en la pila de derecha a izquierda.
- Los enteros ocupan 4 bytes.

## Problema 24

Dado el siguiente fragmento de código escrito en C:

```
int F1 (int p1, int p2, int *p3);

void examen (int matriz[50][75], int *a, int b, int c)
{
    int local;

    ...
}
```

Se pide:

- Dibujad el bloque de activación del procedimiento "examen", indicando claramente a que distancia del registro EBP se encuentran todos los parámetros y variables locales.

b) Traducid a lenguaje ensamblador IA32 la siguiente sentencia, suponiendo que se encuentra dentro del procedimiento examen:

```
local = *a + F1(*a, matriz[c][b], &local);
```

c) Programad el siguiente código en lenguaje ensamblador IA32:

```
acum = 0;
for(j=0;j<50;j++)
    acum =acum+matriz[j][3];
```

suponiendo las siguientes declaraciones:

```
int matriz[50][75];
int acum, j;
```

## Problema 25

Donat el següent codi escrit en C:

```
#define DIM 100

unsigned int producte ( int vector[]; int N )
{
    unsigned int tmp, i;

    tmp = vector[0];
    for ( i = 0; i < N; i ++ )
        tmp = tmp * vector[i];
    return (tmp);
}

void main()
{
    unsigned int vector[DIM], resultat;
    ...
    resultat = producte ( vector, DIM );
    ...
}
```

Es demana:

- Dibuixar el bloc d'activació de la funció **producte** considerant que el pas de paràmetres de la funció és per la pila, i el retorn del resultat per registre (EAX).
- Traduir la funció a llenguatge ensamblador IA32. Afegir el codi necessari per controlar problemes d'**overflow** a l'hora de fer la multiplicació.

En cas que es detecti **overflow** es cridarà a una subrutina que s'encarregarà de tractar l'error. La capçalera d'aquesta subrutina és la següent:

```
void ERROR (int num_error);
```

En aquest cas concret, se li ha de passar per paràmetre un valor enter igual a 5. Aquesta subrutina que ens proporciona el sistema **no** s'ha d'implementar. Considereu que la rutina **ERROR** avorta el programa (no retorna al programa que s'estava executant) i retorna al Sistema Operatiu.

Notes:

- El tipus **unsigned int** es correspon a un nombre natural (sense signe) de 32 bits.
- La gestió de les variables locals és per la pila.

## Problema 26

Traducid el siguiente procedimiento en C a lenguaje ensamblador IA32:

```
typedef struct{
    char descrip [256];
```

```

    int ref;
    int precio;
}t_product;

void ponelemento(t_produc t[100], int *k)
{
    t_product e;
    int i;

    for (i=0; i<256; i++)
        e.descripcion[i] = t[*k].descripcion[i] ;
    e.ref = t[*k].ref ;
    e.precio = t[*k].precio ;
    (*k) = (*k) + 1;
}

```

## Problema 27

Donada la següent definició de variables en C:

```

typedef struct{
    int i1;
    int i2[2];
    char c1;
    char c2;
} T1;
typedef struct{
    int i1;
    T1 taula[10];
} T2;
T1 v1;
T2 v2;

```

I el següent codi:

```

int S(T1 a, T2 b, int i)
{
    T1 c;
    b.taula[a.i2[i]].i1 = FUN(a,c);
    return(c.i1);
}

```

Resoleu l'exercici seguint els passos següents:

- Dibuixeu v1 i v2 a memòria principal.
- Dibuixeu el bloc d'activació de la funció S i el de la funció FUN.
- Escriviu la fórmula per calcular l'adreça de b.taula[a.i2[i]].i1 des de la rutina S.
- Traduiu literalment la funció S, suposant que FUN és una funció ja implementada.
- La funció int FUN(T1 x, T1 y) copia tots els camps del paràmetre x sobre el paràmetre y, i retorna x.i1+x.i2[0]+x.i2[1]. Programeu aquesta funció directament en ensamblador IA32.

## Problema 28

Donat el següent programa en C:

```

typedef struct{
    char a,b;
    int c[10];
    int d;
}estruc;

estruc A;

```

```

int F(int i,estruc B,estruc *C)
{
    int j;
    if (i==0) j=32;
    else j=-32;
    C->c[i] += j; /* Equival a: (*C).c[i]=(*C).c[i]+j */
    return(B.c[i]);
}

void M()
{
    unsigned int p,q;
    estruc D;
    A.c[4]=0;
    p=0;
    q=0;
    while (p <= q)
    {
        q=F(p,A,&D);
        p++;
    }
    p=F(9,D,&A);
}

```

Es demana:

- Traduíu la declaració següent: estruc A;
- Traduíu la sentència següent:

```

        if (i==0) j=32;
        else j=-32;

```
- Traduíu la sentència següent:

```

        while (p <= q)
        {
            /* ... */;
            p++;
        }

```
- Traduíu la sentència següent: A.c[4]=0;
- Traduíu la sentència següent: return(B.c[i]);
- Traduíu la sentència següent: C->c[i] += j;
- Traduíu la sentència següent: p=F(9,D,&A);
- Traduíu la sentència següent: q=F(p,A,&D);
- Escriviu el codi de la funció F, abreujada com segueix:

```

int F(int i, estruc B, estruc *C)
{
    int j;
    /* ... */
}

```

## Problema 29

Dado el siguiente código escrito en C:

```

typedef struct {
    char b[10];
    int a;
}Telem;

void insertar_en_fila ( Telem v[50], int pos, Telem *el )
{
    Telem t;
    int i;
    t = v[49];
    for (i=48; i>=pos; i--) v[i+1] = v[i];
    v[pos] = *el;
}

```

```

    *el = t;
}

int insertar_en_matriz ( Telem m[50][50], int fila, int col, Telem el )
{
    int i;
    insertar_en_fila(&m[fila][0], col, &el);
    for (i=fila-1; i>=0; i--) insertar_en_fila(&m[i][0], 0, &el);
    if (el.a > 0)
    {
        for (i=0; i<50; i++) m[i][i].a = m[i][i].a * 5;
        return (1);
    }
    else return (0);
}

```

Se pide que contestéis a las siguientes preguntas:

- Dibujad el bloque de activación de las funciones `insertar_en_fila` e `insertar_en_matriz`, indicando claramente los desplazamientos relativos al BP necesarios para acceder a los parámetros y las variables locales.
- Traducid las siguientes sentencias de la función `insertar_en_fila` a lenguaje ensamblador IA32 (notad que las asignaciones copian estructuras completas):

```

b1)    *el = t;
b2)    for (i=48; i>=pos; i--) v[i+1] = v[i];

```

- Traducid las siguientes sentencias de la función `insertar_en_matriz` a lenguaje ensamblador IA32:

```

c1)    insertar_en_fila(&m[fila][0], col, &el);
c2)    if (el.a > 0)
        {
            . . .
            return (1);
        }
        else return (0);
c3)    for (i=0; i<50; i++) m[i][i].a = m[i][i].a * 5;

```

- Supongamos que los registros AL y CL contienen la siguiente información en binario:

```
AL = 00000010B    CL = 11111111B
```

d1) ¿Cuál/es de las siguientes instrucciones efectúan el salto si son ejecutadas después de la instrucción `cmpb AL, CL`?

```

a) JA etiq      c) JG etiq      e) JZ etiq
b) JNAE etiq    d) JLE etiq     f) JMP etiq

```

d2) ¿Cómo quedan los registros AH, AL y CL después de ejecutar la instrucción `MUL CL`?

d3) ¿Cómo quedan los registros AH, AL y CL después de ejecutar la instrucción `IMUL CL`?

### Problema 30

Disponemos de los siguientes tipos de datos definidos en lenguaje C:

```

typedef struct {
    int a;
    int b;
    int c;
} trio;
typedef struct {
    int n;
    trio vec[20];
    char text[20];
} datos;

```

a) A partir de los tipos de datos anteriores, define las siguientes variables globales.

```
datos origen, destino;
int res, i;
```

b) Traduce de C a lenguaje ensamblador IA32 la siguiente porción de código que forma parte del programa principal.

```
...
if (origen.vec[0].c == destino.vec[0].c)
    res = copia(&destino, origen.vec[0]);
else {
    for (i = 0; i < origen.n; i++)
        destino.text[i] = origen.text[i];
}
...
```

c) Codifica en ensamblador IA32 la siguiente función:

```
int copia (datos *d, char t){
    int i = 0;
    while(d->text[i] != '.'){
        d->vec[i] = t;
        i++;
    }
    return (d->n);
}
```

## Problemas de Entrada/Salida

### Problema 31

Disponemos de un computador con un controlador de terminal cuyas características se describen a continuación.

Un controlador de terminal es un dispositivo que gestiona simultáneamente un teclado y una pantalla. Este controlador dispone de los siguientes registros de E/S no mapeados en memoria:

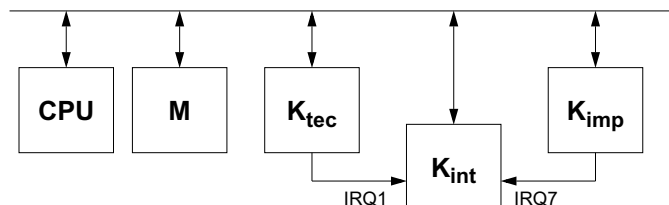
- 0x4A (lectura). Registro de datos del teclado. En este registro se almacena el código ASCII del último carácter teclado.
- 0x4B (lectura). Este registro permite averiguar el estado del controlador:
  - bit 0: cuando este bit vale 1, indica que hay un dato pendiente de leer en el registro de datos de teclado.
  - bit 1: cuando este bit vale 1, indica que podemos mandar un dato a la pantalla.
- 0x4C (escritura). Registro de datos de la pantalla. En este registro se escribe el código ASCII del carácter que queremos escribir en pantalla.
- 0x4D (escritura). Registro de control. Este registro sirve para dar la orden de escribir en pantalla, poniendo un 1 en el bit 3. El resto de bits no se utilizan.

Se pide:

Escribid un programa en C que haga el ECO de todos los caracteres tecleados. Es decir, leer los caracteres que entran por el teclado y escribirlos en la pantalla.

### Problema 32

Tenemos un sistema basado como el de la figura, y con los periféricos descritos en clase. Se quiere realizar un programa que imprima todos los caracteres tecleados.



Se pide:

- a) Escribid una versión en que la sincronización con los controladores de periférico se realice por encuesta.
- b) Escribid una versión en que la sincronización con el Ktec se realice por encuesta y con el Kimp por interrupciones.
- c) Escribid una versión en que la sincronización con el Ktec se realice por interrupciones y con el Kimp por encuesta.
- d) Escribid una versión en que la sincronización con todos los controladores se realice por interrupciones.

### Problema 33

Debe realizarse un programa que escriba en pantalla, cada 5 minutos, la frase: "HAN PASADO 5 MINUTOS". Suponed que existe una rutina que permite escribir frases por pantalla, y que el periférico reloj genera  $N=10$  interrupciones por segundo.

### Problema 34

Construir un programa que cuente el número de caracteres válidos tecleados durante 5 minutos. Debe tenerse en cuenta que el carácter especial DEL indica que el último carácter tecleado no es válido. La sincronización entre el procesador y el controlador de teclado debe realizarse por interrupción. El periférico reloj genera  $N=10$  interrupciones por segundo.

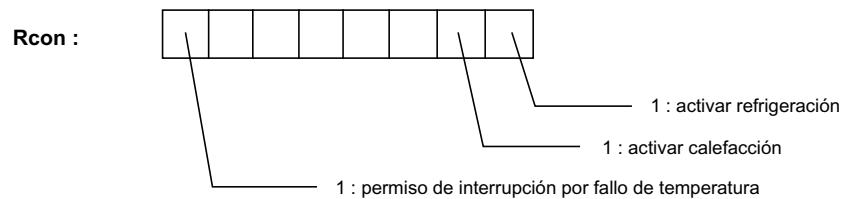
### Problema 35

El magnate griego Kikus Millonettis a sus 70 años de edad se ha convertido en un apasionado ecologista, y de esta forma ha procedido a instalar un invernadero en el jardín de su casa. Dicho invernadero requiere que la temperatura  $T$  se mantenga en un rango determinado

$$T_{\min} < T < T_{\max}.$$

El control de dicho invernadero se va a realizar mediante un computador personal que el Sr. Millonettis tiene en la mesa de su despacho y que habitualmente utiliza para redactar informes y efectuar cálculos financieros. Para ello se coloca en el invernadero un sistema de aire acondicionado que posee un controlador  $K_{\text{invernadero}}$  con los siguientes registros accesibles a nivel lenguaje máquina y mapeados en el espacio de direcciones de E/S:

- **Rtemp**: indica la temperatura ambiente;
- **Rcon**: registro de control general con la información siguiente:



Dicho controlador  $K_{\text{invernadero}}$  mide de forma automática la temperatura  $T$  depositando el valor correspondiente en el registro **Rtemp**. Así mismo también compara el valor de la temperatura medida con los valores límite prefijados  $T_{\min}$  y  $T_{\max}$ . En el momento en que la temperatura quede fuera de rango, el controlador producirá una interrupción (siempre y cuando el correspondiente bit del registro de control lo permita).

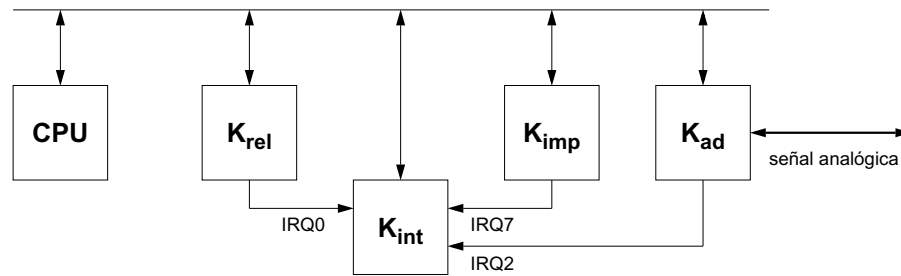
Tras detectar la existencia de anomalía, el computador deberá tomar las medidas pertinentes para que el ambiente del invernadero vuelva a su normalidad (aún y cuando el Sr. Millonetti esté trabajando con el mismo). Esto se efectuará mediante la activación del sistema correspondiente y la posterior lectura *a intervalos de 5 minutos* de la temperatura. Dicha lectura se hace a intervalos dado que la temperatura presenta una gran inercia a variaciones instantáneas (esto implica que por ejemplo, no pasaremos de anomalía por temperatura baja a anomalía por temperatura excesiva).

Si al cabo de 60 minutos el ambiente del invernadero no hubiese vuelto a la normalidad, el computador activará el altavoz para indicar lo anómalo de la situación en el invernadero (llamada a una rutina **beep()**).

Escribir en alto nivel las rutinas que consideréis necesarias para que este sistema funcione tal y como acabamos de describir.

### Problema 36

Con un PC se quiere realizar el siguiente esquema:



Donde el K<sub>ad</sub> es un conversor analógico-digital, es decir, un dispositivo capaz de convertir el valor de la tensión eléctrica que se aplica a su entrada en un valor binario equivalente. Este valor binario es accesible a nivel de lenguaje máquina accediendo al registro de datos Rdat<sub>ad</sub> (8 bits). El funcionamiento del K<sub>ad</sub>, se controla mediante tres registros adicionales:

- Rfrec<sub>ad</sub>. Registro de escritura de 8 bits en donde se indica el número de conversiones por segundo que se quieren realizar (entre 0 y 255 conversiones).
- Rnum<sub>ad</sub>. Registro de escritura de 8 bits en donde se indica el número total de conversiones que se quieren realizar (entre 0 y 255 conversiones) desde que se permite su funcionamiento. Finalizadas todas la conversiones este registro vale cero.
- Rcon<sub>ad</sub>. Registro de escritura de 8 bits del que sólo nos interesa el bit de menor peso. Cuando activamos este bit a 1, el Kad realiza las conversiones de forma automática en función de los parámetros indicado en los dos registros anteriores.

El K<sub>ad</sub> genera una interrupción por la IRQ2 cada vez que tiene una nueva conversión en el registro Rdat<sub>ad</sub>, indicando que ya se puede leer. Si no se lee una determinada conversión, ésta será eliminada por la siguiente.

Debe construirse un programa que realice la siguiente tarea:

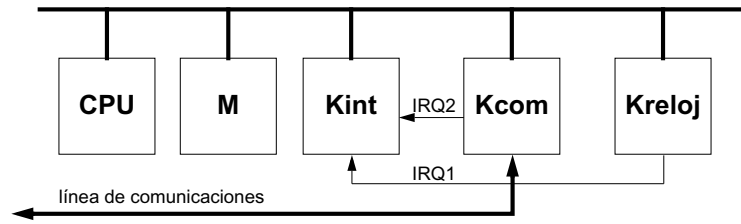
- Cada 5 minutos se ha de activar el K<sub>ad</sub> para realizar N conversiones a una frecuencia F. N y F son constantes menores que 256.
- Simultáneamente se deberán imprimir las conversiones con la menor pérdida de tiempo posible. La sincronización entre el procesador y la impresora ha de hacerse por interrupciones. El tiempo entre dos conversiones consecutivas puede ser mayor, menor o igual que el tiempo necesario para imprimir el valor de la conversión.

Existe una rutina llamada BIN\_ASCII que acepta como parámetro un número binario de 8 bits y produce como resultado los tres códigos ASCII correspondientes a los tres dígitos del número:

```
void bin_ascii (binario, car1, car2, car3)
char binario;
char *car1, *car2, *car3;
```

### Problema 37

Disponemos de un computador con la siguiente estructura:



Kcom es un controlador de comunicaciones capaz de recibir y transmitir caracteres a través de una línea de comunicaciones. Los registros visibles a nivel de lenguaje máquina son:

- *KcomDAT* (8 bits), contiene el código del carácter a transmitir o del carácter recibido por la línea de comunicaciones.
- *KcomEST* (8 bits), El bit 0 indica el motivo por el cual se ha interrumpido a la CPU: (0), ha finalizado la transmisión de un carácter; (1), se ha recibido un carácter por la línea de comunicaciones.
- *KcomCNTRL* (8 bits), para dar la orden de transmisión de un carácter debe generarse un flanco ascendente en el bit 0 (escribir un 0 y luego un 1). El resto de bits de este registro se utilizan para otras tareas no especificadas, en consecuencia no deben modificarse.

Para transmitir un carácter por la línea de comunicaciones se escribirá el código del carácter a transmitir en el registro *KcomDAT* y se dará la orden de transmisión en el bit 0 del registro *KcomCNTRL*. Al finalizar la transmisión de este dato se producirá una interrupción por la línea IRQ2. Si Kcom recibe un carácter por la línea de comunicaciones, entonces almacena el dato en *KcomDAT* y genera una interrupción por la línea IRQ2. El Kreloj y el Kint funcionan tal y como se ha descrito en clase.

A través de la línea de comunicación se desea enviar mensajes con el siguiente formato:



IM es un carácter que identifica el inicio del mensaje. TEXTO es un número variable de caracteres que configuran el cuerpo del mensaje. FM es un carácter que identifica el final del mensaje. Los caracteres IM y FM nunca aparecen en el cuerpo del mensaje.

Para dar por buena la transmisión de un mensaje, debe recibirse por la línea de comunicaciones un carácter REC de reconocimiento de mensaje, que confirma la correcta recepción del mensaje al otro extremo de la línea de comunicaciones. Si este carácter no es recibido antes de 5 segundos desde que se envió el carácter FM, dicho mensaje debe ser retransmitido. Después de 5 retransmisiones infructuosas del mismo mensaje se dará por averiada la línea de comunicaciones y se acabará la aplicación llamando a la rutina "abort()". Esta rutina puede llamarse desde cualquier punto de la aplicación (main, rutinas de interrupción, etc.). Puede suponerse que durante la transmisión de un mensaje no se producirá la recepción de ningún carácter por la línea de comunicaciones.

Existe una función `char *obtener_mensaje()` que devuelve la dirección donde se encuentra el siguiente mensaje a transmitir (incluyendo los caracteres IM y FM). Esta rutina devuelve resultado inmediatamente.

Se pide la descripción en alto nivel (pseudo código o C) del programa principal y las rutinas de atención al reloj y a Kcom de forma que se realice la transmisión de N mensajes. El programa acabará después de la transmisión correcta de los N mensajes o bien en el caso de que se dé por averiada la línea de comunicación.

### Problema 38

Es tracta d'automatitzar la nova muntanya russa del parc d'atraccions del Tibidabo. El controlador de cua detecta l'arribada de *cada* passatger a l'entrada de l'atracció. Si hi ha espai a la cua, l'hem d'admetre obrint la porta d'entrada a la cua i, si no hi ha espai, s'ha d'encendre un rètol indicador que il·lumina el missatge "Cua plena. Vingui més tard, si us plau". El nombre màxim de passatgers que caben a la cua és MAX\_CUA. A cada viatge, la nova atracció admet fins a CAPACITAT passatgers ( $MAX\_CUA > CAPACITAT$ ). Per iniciar un nou viatge (admetre passatgers al recinte interior de l'atracció, esperar que pugin al vagó i engegar) cal esperar a que acabi el viatge anterior (el viatge acaba quan han sortit tots els passatgers anteriors del recinte interior), i cal que hi hagi prou passatgers a la cua per omplir completament el vagó. Tanmateix, si passen 5 minuts des del final del viatge anterior, començarem el nou viatge amb els passatgers que hi hagi (encara que el vagó vagi buit).

El controlador de rellotge és com el descrit a classe (18 tics per segon, ID\_REL), i els registres dels controladors de cua i vagó, mapejats a l'espai d'E/S, són els següents:

- Rest\_cua (8 bits). El bit zero es posa a 1 cada vegada que arriba un passatger a l'entrada de la cua, i es desactiva automàticament quan escrivim un valor al registre de control, Rcont\_cua. Valor inicial, 0.
- Rcont\_cua (8 bits):
  - Bit 0: Permís d'interrupció. Si el valor d'aquest bit és 1, el controlador genera una interrupció (ID\_CUA) cada vegada que arriba un passatger. La sol·licitud d'interrupció es desactiva automàticament quan escrivim un valor al registre de control, Rcont\_cua. Valor inicial, indeterminat.
  - Bit 1: Obrir porta. Un 1 en aquest bit fa obrir la porta que admet el nou passatger a la cua. La porta es tanca automàticament darrera del passatger, i el bit es desactiva.
  - Bit 2: Cua plena. Un 1 en aquest registre fa encendre el rètol indicador de cua plena. El rètol s'apaga automàticament al cap d'uns instants, i el bit es desactiva.
- Rest\_vago (8 bits). El bit zero es posa a 1 quan el vagó està apunt per rebre passatgers (quan han sortit tots els passatgers del viatge anterior), i es posa automàticament a 0 quan donem l'ordre d'iniciar un nou viatge (bit 1 d'Rcont\_vago). Valor inicial, 1.
- Rcont\_vago (8 bits):
  - Bit 0: Permís d'interrupció. Si el valor d'aquest bit és 1, el controlador genera una interrupció (ID\_VAGO) cada vegada que acaba un viatge (quan surten tots els passatgers del recinte). La sol·licitud d'interrupció es desactiva automàticament al moment d'activar-se la rai. Valor inicial, indeterminat.
  - Bit 1: Ordre d'iniciar un viatge. Un 1 en aquest bit fa obrir la porta del recinte interior que, de manera automàtica, admet fins a un màxim de CAPACITAT\_VAGO passatgers. També de manera automàtica, el vagó engega quan tots els passatgers s'hi han assegut i s'han posat el cinturó. El bit es desactiva automàticament quan acaba el viatge.

ES DEMANA:

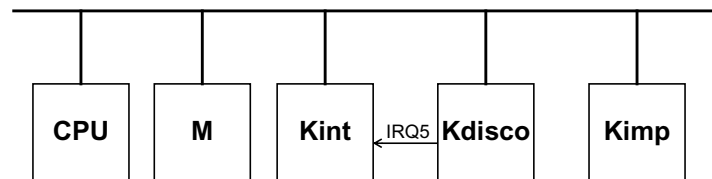
Escriviu un programa complet (el main i les *rais*) de manera que l'atracció funcioni de la forma que acabem de descriure, amb tots els controladors treballant per interrupció.

### Problema 39

Se dispone de un computador con el procesador, memoria y controladores descritos en clase. Se pide diseñar un programa en alto nivel para imprimir el contenido del sector 0 de todas las pistas de la 1 a la 15 (ambas inclusive). La transferencia de información entre el disco y la memoria se realizará por DMA. La sincronización entre el procesador y el controlador de disco se realizará por interrupciones. La sincronización entre el procesador y el controlador de impresora se realizará mediante encuesta. Puede suponerse que, para el almacenamiento de datos en memoria existe un BUFFER de tamaño ilimitado.

### Problema 40

Disponemos un computador con la siguiente estructura:



Donde los controladores son similares a los utilizados en las clases de aplicación y su descripción es la siguiente:

- Controlador de Impresora (Kimp). Esta impresora sólo puede trabajar por encuesta y tiene dos registros de E/S:
  - RDATOS. Registro de escritura. Ha de contener el código ASCII del carácter que se desea imprimir.
  - RESTCONT. Registro de lectura / escritura. Cuando el bit 0 está a 1, nos indica que la impresora está ocupada. Cuando escribimos un 1 en el bit 1, se manda a imprimir el carácter almacenado previamente en el registro RDATOS.
- Controlador de disco (Kdisco). Este dispositivo trabaja siempre por DMA. La cantidad de información que se transfiere es un sector de disco, cada sector ocupa 512 bytes. Al acabar una transferencia, el controlador provoca una interrupción por la IRQ5. Los registros de E/S del controlador son los siguientes:
  - CONTROL. Registro de escritura. En el bit 1 se indica con 1 que queremos realizar una escritura en disco, y con un 0 se indica que queremos realizar una lectura de disco. El bit 2 sirve para dar la orden de inicio de transferencia. Este bit vale inicialmente 0. Al activarlo a 1, se inicia la transferencia. Una vez acabada la transferencia, el controlador lo vuelve a dejar a cero.
  - PISTA. Registro de escritura. Indica la pista involucrada en la transferencia. Nuestro disco tiene 1024 pistas.
  - SECTOR. Registro de escritura. Indica el sector involucrado en la transferencia. Cada pista tiene 64 sectores.
  - DIRECCION. Registro de escritura. Indica la dirección de memoria en dónde se escriben (en caso de lectura de disco) o desde dónde se leen (en caso de escritura en el disco) los datos involucrados en la transferencia.
- Controlador de interrupciones (Kint), funciona igual que el visto en clase.

Se pide:

Diseñar una aplicación que lea un fichero de texto almacenado en las pistas 34 y 35 (el fichero ocupa 128 sectores) del disco y que lo mande a imprimir.

Información adicional:

- Se valorará la eficiencia de la solución propuesta, así como el correcto uso de los registros de E/S.
- El fichero no cabe completo en memoria. En concreto sólo hay espacio para almacenar 16 sectores en el vector “buffer”, definido en C de la siguiente forma: `char buffer[16*512];`
- No se pueden hacer suposiciones respecto a la velocidad de los periféricos.

### Problema 41

Disponemos de un computador con un controlador de disco cuyas características se describen a continuación.

El disco que utilizamos tiene 1024\*1024 sectores, cada uno de 4 Kbytes. El controlador de disco dispone de los siguientes registros de E/S no mapeados en memoria:

- 0x90 (lectura - escritura). Este registro permite averiguar el estado del disco y enviarle las órdenes de funcionamiento. La descripción de este registro es la siguiente:
  - bits 1-0: operación a realizar, 00 leer sector, 01 escribir sector, 10 formatear sector.
  - bit 2: sincronización, 0 encuesta, 1 interrupciones.
  - bit 3: estado del disco, 0 disponible, 1 ocupado.
  - bit 4: inicio de transferencia, 1 orden de inicio (una vez iniciada la transferencia este bit se pone a 0).
  - resto de bits: reservados, no se pueden modificar.
- 0x91 (escritura). En este registro se indica el sector que queremos transferir. A partir de esta información el controlador de disco calcula la cara, pista y sector físicos.
- 0x92 (escritura). En este registro se indica la dirección del bloque de datos a transferir.

El funcionamiento del disco es el siguiente. Una vez programados los registros del controlador, se realiza la transferencia del sector indicado por DMA. Si la sincronización se realiza por interrupciones, al acabar la transferencia el controlador interrumpe a la CPU por la IRQ5.

Se pide: Escribid un programa en C que escriba un fichero en el disco a partir del sector 34567. Este fichero tiene un tamaño de M bytes y está almacenado a partir de la dirección D. La sincronización ha de realizarse por interrupciones.

### Problema 42

En la cadena de muntatge de la empresa Chupetin<sup>TM</sup> es disposa d'un sistema computador que té com a objectiu controlar la qualitat dels productes que van sortint. Aquest sistema a més de CPU, memòria i rellotge, porta incorporat un dispositiu testejaador dels acabats dels productes i una impressora làser. El controlador del dispositiu testejaador, està format per tres registres: un de dades, un de control i un d'estat. Per fer-lo treballar, inicialment s'ha d'especificar en el registre de control què és el que volem fer (lectura o escriptura) i com ho volem fer (per enquesta o per interrupció). En quant al controlador d'impressora làser, també està format per tres registres: un de dades, un de control i un d'estat. El funcionament d'aquests registres és similar al dels registres del testejaador.

El que fa el sistema és anar llegint dades procedents del testejaador, i cada interval determinat de temps, les va imprimint a la impressora làser. El programa que gestiona aquest procés és el següent:

```

PP
Inicialitzar Vector d'Interrupcions
TESTCONT = lectures per enquesta
IMPCONT = escriure per interrupció
i = 0
tics = 0
temps = fals
final = fals
mentre no final fer
    repetir
        llegir TESTEST
        fins que TESTEST = dada rebuda
        i = i + 1
        informacio[i] = TESTDAT
        si temps llavors
            temps = fals
            total = i
            i = 0
            j = 0
            IMPDAT = informacio[j]
        fsi
    fmentre

RAI rellotge
tics = tics + 1
si tics = X_SEGONS llavors
    tics = 0
    temps = cert
fsi
EOI
IRET

RAI impressora
si j < total llavors
    j = j + 1
    IMPDAT = informacio[j]
fsi
EOI
IRET

```

El problema és que l'operari d'aquest ordinador, avorrit per la monotonia del seu treball, ha pensat en la possibilitat de poder jugar a marcians mentre la màquina està prenent mostres de la qualitat dels productes. Per poder fer això, s'ha de modificar el codi d'aquest programa amb l'objectiu de deixar el programa principal lliure i així poder fer altres tasques. Per això, l'operari ha avisat al seu nebot, un futur enginyer de telecomunicacions, i li ha proposat una sèrie de modificacions al programa:

- a) Suposant que el testejador pot treballar per interrupció, canviar el PP i escriure la RAI del testejador de forma que el sistema es comporti igual.
- b) Si a més, ara suposem que la transferència d'informació cap a l'impressora làser pot ser mitjançant el mecanisme d'accés directe a memòria (DMA), quins registres addicionals haurà de tenir el controlador d'impressora làser? Com quedarà ara el PP i la RAI de la impressora.

NOTA: Per simplificar aquest apartat, es pot suposar que el número de bytes a transferir cada X segons és suficientment petit per poder ser especificat en el registre de tamany de la transferència.

- c) Modificar la rutina que sigui necessària de forma que el PP quedi completament lliure de forma que l'operari ja podrà jugar a marcians.

### Problema 43

Un ordinador disposa d'una pantalla gràfica mapejada a memòria, que ocupa 2k bytes a partir de l'adreça `pantalla`. I d'un disc, com el descrit al final de la col·lecció de problemes, amb sectors de tamany `TAMSEC=512`. El disc funciona per accés directe a memòria, amb interrupcions després de la transferència de cada sector. Anomenarem "fitxer 1" als sectors 3..6 de la pista 7 de la cara 0 del disc. Es demana:

- a) Escriviu un programa complert (programa principal i `rai_disc`) per copiar el contingut de la pantalla en el "fitxer 1".
- b) El teclat d'aquest ordinador genera una interrupció cada vegada que es polsa una tecla, i per desactivar la sol·licitud n'hi ha prou amb llegir el registre de dades del controlador de teclat (`Rdad_tec`). Anomenarem `ESC` al codi que genera el controlador quan polsem la tecla "ESCAPE". I anomenarem "fitxer 2" als sectors 0..3 de la pista 2 de la cara 1 del disc. El "fitxer 2" es pressuposa inicialitzat. Escriviu un programa complert (programa principal, `rai_disc`, i `rai_teclat`) que, en polsar la tecla "ESCAPE", salvi el contingut de la pantalla en el "fitxer 1" i bolqui en pantalla el contingut del "fitxer 2".
- c) Descriviu (NO programeu) quines modificacions caldria fer al programa i rutines de l'apartat anterior si "fitxer 1" i "fitxer 2" fossin el mateix (és a dir, en polsar la tecla `ESCAPE` s'ha d'INTERCANVIAR els continguts de la pantalla i del fitxer sense perdre informació). L'operació s'ha de fer en el mínim de temps possible. Discutiu també si cal algun "buffer" adicional, i de quin tamany.

### Problema 44

L'empresa Virtual Quality, líder en el disseny i distribució d'aplicacions per el control de qualitat de diferents productes, ha dissenyat un nou dispositiu testejadore per a l'extracció de mostres de qualitat en una cadena de muntatge. Aquest dispositiu està connectat a un sistema computador basat en l'arquitectura IA32, tal i com l'explicat a classe.

Aquest dispositiu, mitjançant els seus diversos sensors, va prenent mostres continuadament. Cada cop que pren una nova mostra, guarda el resultat (un caràcter de 8 bits) en el seu registre de dades (que es troba a l'adreça `Rtest` de l'espai d'entrada/sortida) i envia una interrupció al processador per la línia `IRQ6`.

Es demana que programeu una aplicació que vagi llegint les mostres que arriben del testejadore. Quan la mostra llegida coincideix amb el caràcter '#', s'enviaran a imprimir totes les mostres rebudes desde l'última transferència fins al caràcter '#'. Si passen T segons sense rebre el caràcter '#', es guardaran els últims caràcters rebuts a disc a partir de la pista P i el sector S (i aquests no s'imprimiran). L'aplicació ha d'acabar a la tercera vegada que han passat els T segons de forma consecutiva, però sense fer aquesta tercera transferència.

Programeu només les rutines d'atenció a les interrupcions. La impressora i el disc són com els explicats a classe, i els dos dispositius treballen per interrupció.

Comentaris:

- Supposeu que disposeu d'un buffer de tamany il·limitat, que comença a partir de l'adreça de memòria `buffer`.
- Podria ser que al rebre el caràcter '#' encara no s'hagués acabat d'imprimir l'anterior bloc de mostres, però podeu suposar que en T segons s'ha acabat qualsevol transferència.
- Mentre es realitza qualsevol transferència podem seguir rebent altres caràcters.

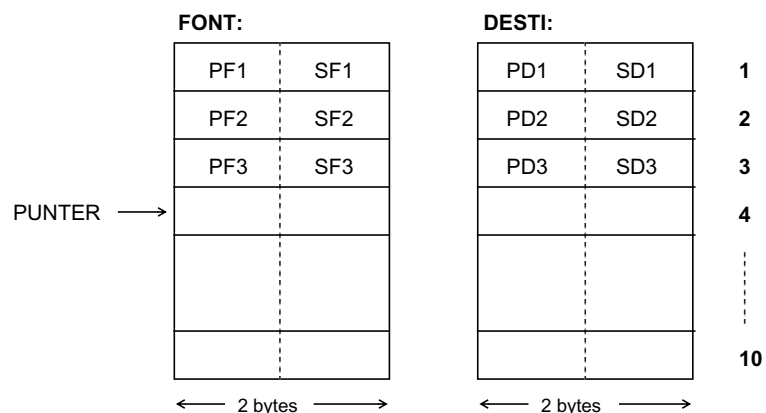
## Problema 45

L'aplicació que volem programar consisteix en realitzar la còpia de 10 fitxers emmagatzemats en uns sectors determinats d'un disc sobre uns altres sectors del mateix disc. Tant la lectura com l'escriptura en el disc es realitzen per DMA. A continuació s'explica amb més detall les característiques de l'aplicació:

- Existeixen dues taules a memòria, a partir de les adreces simbòliques FONT i DESTI. La taula FONT emmagatzema la pista i sector on es troba cadascun dels 10 fitxers a llegir. La taula DESTI emmagatzema la pista i sector on s'ha de copiar cadascun dels 10 fitxers. La figura mostra aquestes estructures de dades. Cada fitxer ocupa S sectors.
- Existeix una subrutina anomenada CAPTURA que escriu, en funció de l'informació introduïda per l'usuari, els valors adients a les dues taules anteriors. Existeix a més una variable global anomenada PUNTER que apunta a la primera posició lliure de les taules. Aquesta subrutina ja està programada i funciona correctament.

Així per exemple, la tercera vegada que cridem a CAPTURA se'ns ompliran les entrades PF3, SF3, PD3 i SD3. En retornar, la variable PUNTER valdrà 4, tal i com indica la figura.

- Es desconeix el temps que pot trigar el procés de captura, és a dir, aquest temps pot ser més petit, més gran, o igual que el temps emprat en el procés de còpia d'un fitxer.
- La lectura de cadascun dels fitxers es realitzarà a partir de l'adreça simbòlica BUFFER. La mida del BUFFER és suficientment gran com per emmagatzemar qualsevol dels 10 fitxers, **però no tots simultàniament**.
- Considereu el disc té 1 cara, P pistes per cara, i S sectors per pista.



Es demana:

- a) Descriure com pot executar-se aquesta aplicació per tal de minimitzar el temps d'execució (processos de CAPTURA i còpia de fitxers es facin de forma concurrent). Indiqueu quines variables de sincronització s'utilitzen i quina és la seva funció.
- b) Escriviu el programa principal i la rutina de servei al DMA en el llenguatge C.

## Problema 46

La multinacional catalana Xapusses Aeroespacials XAPALS, ha pensat en el disseny d'un modern sistema computador per a la recollida i processament d'informació procedent del satèl·lit TasPassat. Sembla ser que la part hardware del disseny (controladors) ja està resolta, però els falta la part software.

L'invent està format per un dispositiu receptor que llegeix informació del satèl·lit, i la converteix a digital. Per una altra banda, hi ha un altre dispositiu que s'encarrega d'agafar aquesta informació, processar-la i emmagatzemar-la degudament. Finalment, també hi tenim un rellotge.

Cada un dels dos dispositius disposa del seu propi controlador, el qual està format per un interfacie amb el dispositiu, i un interfacie amb el processador (CPU) del sistema. La descripció detallada del interfacie amb el processador de cada un dels dos dispositius és la següent:

- **Lector de Satèl·lit**, treballa sempre per **interrupció**, del qual ens interessen 2 registres:
  - SATDAT - És el registre de dades del controlador del lector d'informació del satèl·lit. En aquest registre hi ha la informació digitalitzada rebuda del satèl·lit. El registre és de 8 bits.
  - SATCONT - És el registre de control. Si en el bit 0 d'aquest registre (de 8 bits) hi ha un 1, aleshores el dispositiu està en estat de treball. Altrament, el dispositiu no fa res (està desactivat). Cada cop que el controlador rep una nova dada, aleshores envia una interrupció al processador.
- **Processador d'Informació**, treballa per **interrupció** i la transferència d'informació cap a aquest dispositiu és mitjançant el mecanisme d'**accés directe a memòria** (DMA). Aquest dispositiu està format per 3 registres:
  - PROADR - En aquest registre de 8 bits, hi posarem l'adreça de memòria a partir de la qual ha d'efectuar la transferència d'informació.
  - PROLONG - En aquest registre de 8 bits hi posarem el número de bytes que s'han de transferir. Com que només disposem de 8 bits, el tamany màxim que pot tenir cada transferència és de 256 bytes.
  - PROCONT - És el registre de control. El segon bit indica si la transferència que volem fer és una lectura (bit 2 = 0) o una escriptura (bit 2 = 1). En el moment en que posem el tercer bit a 1, començarà la transferència d'informació, sigui una lectura o una escriptura. En el moment que acabi la transferència, enviarà una interrupció a la CPU.

El programa haurà d'anar llegint dades del lector d'informació de satèl·lit i guardar-les en un buffer de memòria que suposarem que té un tamany suficientment gran (o sigui, un array que ens haguem declarat en el programa principal). Aquesta informació la haurem de transmetre al processador d'informació cada T segons. Heu de tenir en compte que podem haver llegit més de 256 bytes en T segons. Això vol dir que potser no en tenim prou amb una sola transferència. Es pot considerar que la transferència d'informació cap al processador d'informació és més ràpida que la transferència procedent del lector de satèl·lit (és a dir, no haurem llegit cap nou dada del satèl·lit abans de finalitzar la transferència cap al processador d'informació). El programa ha d'acabar quan hagin passat 5 minuts sense que haguem rebut cap nova dada del lector de satèl·lit.

Es demana el disseny en alt nivell del programa que ha de gestionar aquest sistema. Això implica que s'ha de dissenyar el programa principal, i les respectives rutines d'atenció a les interrupcions.

## Problema 47

L'empresa (...) en la seva divisió de control de qualitat, disposa d'un ordinador personal basat en l'arquitectura IA32, que s'utilitza per executar programes d'ús general. S'ha pensat en la possibilitat d'incrementar la seva funcionalitat i utilitzar-lo com a eina de control de qualitat dels productes de l'empresa, mantenint la possibilitat de seguir utilitzant l'ordinador com fins ara.

Així doncs, s'hi connecta un nou dispositiu que mostreja constantment certs aspectes dels productes i deixa el codi resultant en un registre que es troba a l'adreça simbòlica R\_DATPROD de l'espai d'entrada/sortida. Aquesta informació, que ocupa 8 bits, s'ha d'anar llegint cada T interrupcions de rellotge (tics), i guardant a un buffer que, a efectes d'implementació, podem suposar que és de tamany il.limitat.

- a) Es demana que programeu les rutines que calguin per tal d'aconseguir el control desitjat d'aquest nou dispositiu.

Un cop connectat el dispositiu mostrejador, es desitja analitzar la informació obtinguda. Per això, connectem ara una impressora, en la qual s'ha d'anar imprimint les dades que es van llegint.

- b) Modifiqueu i programeu les rutines que calguin per que s'imprimeixi la informació tal i com es demana. No es pot fer cap suposició sobre la velocitat de la impressora.

Finalment, es pretén també que tota la informació llegida es guardi a partir de la pista P i el sector S del disc del sistema. El disc treballa sempre per accés directe a memòria (DMA), i cada accés és de NUMBYTES bytes. Per a simplificar el problema, podeu suposar que una cara del disc té un número il.limitat de pistes, i que cada pista té NUMSEC sectors.

- c) Modifiqueu i programeu les rutines que calguin per aconseguir el funcionament demanat. Com abans, no es pot fer cap tipus de suposició en quant a la velocitat del disc.

Notes:

- Supposeu que el vector d'interrupcions i tots els dispositius ja s'han inicialitzat en el programa principal, que no es demana.
- Els dispositius impressora, rellotge i disc funcionen exactament com han estat explicats a classe.
- La programació de les rutines es realitzarà en llenguatge C.

## Problema 48

Es disposa d'una "llista" definida com:

```
typedef struct {
    int sector;
    int pista;
    int cara;
    int num_sectors;
} dades_fitxer;

typedef struct {
    int actual;
    int lliure;
    dades_fitxer taula[IL.LIMITADA];
} llista;
```

Cada entrada de la "taula" indica les dades d'inici d'un fitxer, i el nombre de sectors (consecu-

tius) que el componen. El camp “actual” de la “llista” indica en tot moment la posició de la “taula” que conté les dades del proper fitxer que volem llegir de disc, i el camp “lliure” indica la primera posició lliure de la taula.

- a) Dissenyu un programa complet (principal i rutines d’atenció a interrupcions) per llegir de disc i imprimir les dades de tots els fitxers (ascii) especificats a la “llista” des de la posició “actual” (inclosa) fins a la posició “lliure” (exclosa). Supposeu que la variable “llista” està correctament inicialitzada.

Utilitzeu un buffer de tamany tant gran com calgui, i els controladors de disc i impressora habituals (impressora de caràcters per interrupció, i disc per DMA i interrupció).

- b) Es disposa d’un perifèric d’entrada de dades K\_sensor que respon al següent model: Cada cop que detecta un event, genera un senyal d’interrupció; el seu registre de dades R\_dad\_sensor identifica l’event que ha provocat la interrupció. Supposeu que es disposa de la següent funció (ja programada):

```
void dir_quin_fitxer(INT id_event, DADES_FITXER *dades);
```

que determina quin fitxer s’ha d’imprimir cas de produir-se l’event “id\_event”.

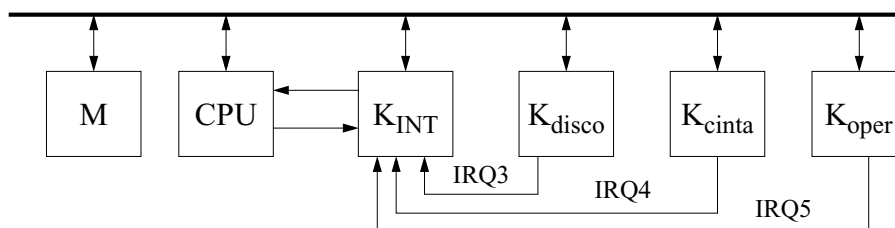
Es demana escriure la rutina d’atenció a les interrupcions del K\_sensor a fi i efecte de que, cada vegada que es produeixi un event, s’actualitzi la “llista” de fitxers a imprimir amb les dades corresponents al nou event.

- c) Feu el disseny complet (programa principal i rutines d’atenció a les interrupcions) per coordinar el funcionament del sistema sencer. El sistema recull events del K\_sensor i imprimeix, tant aviat com possible, els fitxers corresponents. El programa no ha de finalitzar mai (cicle infinit). Recordeu que la “llista” de fitxers a imprimir té capacitat il.limitada.

## Problema 49

Una aplicació típica en un centre de càlculo es realitzar backups periòdics de la informació almacenada en disco a dispositius de almacenamiento permanente. Un backup de disco consiste en realitzar una copia de la informació almacenada en el disco sobre una serie de cintas (p.e.).

Supongamos que disponemos de un computador con la siguiente configuración:



La descripció de los dispositivos sería la siguiente:

- Kdisco. Este controlador maneja un disco duro con las siguientes características: 8192 pistas, 512 sectores por pista y 4096 bytes por sector (tanto las pistas como los sectores están numerados a partir del cero hasta N-1). Este controlador, que realiza la transferencia de información mediante DMA, sólo permite leer/escribir sector a sector. Los registros de los que dispone este controlador son los siguientes:
  - Rdisco\_pista (escritura), se indica el número de pista involucrado en la transferencia.
  - Rdisco\_sector (escritura), se indica el número de sector involucrado en la

transferencia.

- Rdisco\_direccion (escritura), se indica la posición de memoria involucrada en la transferencia.
- Rdisco\_control (escritura), sólo se utilizan los dos bits de menor peso. En el bit 0 se indica si la operación es de lectura (0) o escritura (1). En el bit 1 se indica la orden de inicio (1).

Al acabar la transferencia de un sector, el controlador interrumpe a la CPU a través de la línea IRQ3.

- Kcinta. Este controlador maneja una cinta magnética capaz de almacenar 1 Gbyte de información (1024 x 1024 x 1024 bytes). Este controlador realiza transferencia de información también por DMA y permite transferir bloques de información de 4096 bytes. Los registros de que dispone este controlador son los siguientes:
  - Rcinta\_direccion (escritura), se indica la posición de memoria involucrada en la transferencia.
  - Rcinta\_control (escritura), de este registro se utilizan los bits 6 y 7, el resto de bits son utilizados por otros controladores y no pueden ser alterados. En el bit 7 se indica si la operación es una lectura (1) o una escritura (0); en el bit 6 se indica la orden de inicio de la operación (1).

Al acabar la transferencia del bloque de información (de 4096 bytes) se interrumpe a la CPU a través de la línea IRQ4.

Nótese que para realizar un backup completo del disco serán necesarias 16 cintas.

- Koper. Este controlador nos servirá para avisar al operador que ha de cambiar la cinta y cuándo ha acabado el backup. Además nos avisará cuando el operador ha montado una nueva cinta. Este dispositivo sólo dispone de un registro (Roper\_control), con la siguiente funcionalidad. Si escribimos un 1 en el bit 0 avisa al operador mediante una señal acústica y luminosa que ha de cambiar la cinta. Si escribimos un 1 en el bit 1 avisa al operador que el backup ha terminado. Estos dos bits están inicialmente a cero y pasan a cero inmediatamente después de que el controlador ha realizado la orden que se ha pedido. Este mismo controlador interrumpirá a la CPU a través de la IRQ5 cuando el operador haya montado una nueva cinta.
- Kint. El controlador de interrupciones funciona como el explicado en clase.

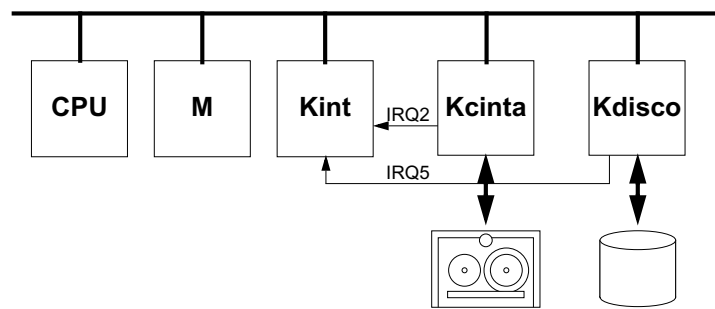
Se pide que diseñéis un programa capaz de realizar el backup del disco duro sobre la unidad de cinta, teniendo en cuenta las siguientes consideraciones:

- Disponemos de un buffer de memoria de tamaño 512 \* 1024 bytes.
- Se necesitan 16 cintas para realizar el backup completo.
- Supondremos que cuando empieza la aplicación la primera cinta ya está montada.
- Al acabar el backup la aplicación ha de avisar al operador.
- No es necesario programar los vectores de interrupción.

## Problema 50

Las cintas magnéticas son dispositivos de almacenamiento de alta capacidad y bajo coste. Su único problema es que son de acceso secuencial (como la cinta de un cassette). Las cintas, por su bajo coste y alta capacidad de almacenamiento, se utilizan para hacer copias de seguridad de la información almacenada en los discos duros.

Disponemos de un computador con la siguiente estructura:



La descripción de los dispositivos es la siguiente:

- **Kdisco.** Este controlador maneja un disco duro con las siguientes características: 8192 pistas, 512 sectores por pista y 4096 bytes por sector (tanto las pistas como los sectores están numerados a partir del cero). Este controlador, que realiza la transferencia de información mediante DMA, sólo permite leer/escribir sector a sector. Los registros de los que dispone este controlador, mapeados en el espacio de direcciones de E/S, son los siguientes:

- Rdisco\_pista (escritura), se indica el número de pista involucrado en la transferencia.
- Rdisco\_sector (escritura), se indica el número de sector involucrado en la transferencia.
- Rdisco\_direccion (escritura), se indica la posición inicial del bloque de información involucrado en la transferencia.
- Rdisco\_control (escritura), sólo se utilizan los dos bits de menor peso. En el bit 0 se indica si la operación es de lectura (0) o escritura (1). En el bit 1 se indica la orden de inicio (1).

Al acabar la transferencia de un sector, el controlador de disco interrumpe a la CPU a través de la línea IRQ5 (ID\_DISC = 13).

- **Kcinta.** Este controlador es capaz de leer/escribir cintas magnéticas, y permite transferir bloques de información de tamaño variable (hasta un máximo de 64 Kbytes). Esto permite leer/escribir ficheros menores de 64 Kbytes en una sola transferencia.

Los registros de E/S de los que dispone este controlador, mapeados en el espacio de direcciones de E/S, son los siguientes:

- ESTADO (lectura), sólo nos interesa el bit 5, que cuando vale 1, nos indica que se ha acabado la cinta.
- CONTROL (escritura), sólo se utilizan los bits 2 y 3. En el bit 3 se indica si la operación es de lectura (0) o escritura (1). En el bit 2 se indica la orden de inicio (1).
- DIRECCION (escritura), se indica la posición inicial del bloque de información involucrado en la transferencia.
- TAMAÑO (lectura - escritura), el significado de este registro depende del tipo de operación a realizar. Cuando se escribe un bloque en la cinta, en este registro hay que indicar el tamaño del bloque a transferir. Cuando se lee un bloque, al acabar la transferencia, el contenido de este registro indica la longitud en bytes del bloque leído.

Al acabar la transferència de un bloque, el controlador de cinta interrompe a la CPU a través de la línia IRQ2 ( $ID\_CINTA = 10$ ).

- **Kint.** El controlador de interrupciones funciona como el explicado en clase.

Se pide:

- a) Escribid un programa en C que lea el contenido completo de la cinta y lo escriba en el disco duro a partir de la pista 25, en pistas y sectores consecutivos. Con las siguientes suposiciones:
  - El contenido de la cinta cabe en el disco duro.
  - Se dispone de un buffer en memoria de tamaño ilimitado.
  - La cinta está rebobinada y dispuesta para leer el primer bloque de información.
- b) Indicad los cambios que serían necesarios realizar en el programa anterior, en el caso de tener un buffer de memoria de tamaño limitado (p.e. 256 Kbytes). Se pide el esquema en pseudo código.

### Problema 51

Volem fer el programa que controli una màquina de discs per a un local públic. El sistema funciona de la següent manera: un usuari introdueix una moneda a la màquina i selecciona la cançó que vol que es reproduïxi. Si la màquina està tocant una cançó, encua la petició i la reproduïx quan hagi acabat amb les peticions anteriors. Si la màquina està lliure, reproduïx directament la cançó demanada. Totes les cançons estan emmagatzemades en un disc de gran capacitat.

El dispositiu que controla el moneder i la selecció de les cançons se sincronitza per interrupcions. Genera una interrupció un cop l'usuari ha introduït la moneda i seleccionat una cançó. Aquesta interrupció pot arribar en qualsevol moment. Aquest controlador a partir de la selecció feta calcula a quina pista i sector es troba l'inici de la cançó, quants sectors de disc ocupa, i quina és la duració de la cançó en segons. Per recuperar aquesta informació, el controlador disposa dels següents registres (TOTS NOMÉS DE LECTURA):

- $R_{pista\_mon}$  (8 bits) i  $R_{sector\_mon}$  (8 bits): indiquen la pista i el sector en els que hi ha el primer sector de la cançó.
- $R_{numsects\_mon}$  (8 bits): indica el nombre de sectors que ocupa la cançó.
- $R_{duracio\_mon}$  (16 bits): indica la duració en segons de la cançó.

El sistema ha d'anar portant les cançons a memòria per tal que es reproduïxin. Cada cançó es porta a memòria a partir de l'adreça *INICI\_CÀRREGA*, a on hi ha suficient espai per a deixar-hi una cançó sencera. Un cop tenim la cançó (completa) a memòria s'ha de posar a 1 la variable global *tocala\_sam*. Quan s'hagi acabat de reproduir la cançó, aquesta variable s'ha de posar a 0. Mentre s'està reproduint una cançó, s'ha de mantenir tota a memòria, per tant el sistema no porta a memòria la següent cançó a tocar fins que hagi acabat de tocar-se l'anterior.

El programa ha de controlar quan s'acaba una cançó. Quan el programa reconeix que una cançó ha acabat de tocar-se, mira si n'hi ha més d'encuades. Si és així, s'inicia el procés de reproducció de la següent. Altrament, s'espera a que hi hagi una altra petició. En cas que no hi hagi cap petició a la cua i arribi una petició, aquesta es processa directament.

El sistema té un rellotge que interromp N vegades per segon. La transferència del disc a memòria es fa per DMA i la unitat de transferència és un sector de disc. El disc té NPISTES, cadascuna d'elles amb NSECTORS i cada sector ocupa NUMBYTES. El controlador de disc i DMA té els següents registres:

- $R_{pista\_DMA}$  (8 bits) i  $R_{sector\_DMA}$  (8 bits): indiquen la pista i el sector de disc a on es troba la informació a transferir.
- $R_{adr\_DMA}$  (16 bits): indica adreça de memòria de la que s'ha de llegir o escriure.
- $R_{control\_DMA}$  (8 bits): el bit 0 indica com es fa la sincronització (0 per interrupcions, 1 per enquesta). El bit 1 indica de quin tipus d'operació es tracta (0 lectura de disc, 1 escriptura). El bit 2 s'ha de posar a 1 per tal que s'iniciï una transferència.
- $R_{estat\_DMA}$  (8 bits): el bit 0 indica la disponibilitat del controlador (0 disponible, 1 no disponible). El bit 1 indica com ha finalitzat una transferència (1 correctament, 0 altrament).

Les estructures de dades que permeten gestionar quines cançons s'han de tocar són les següents:

- *ncançons*: variable global que indica quantes cançons estan pendents de ser reproduïdes.
- *taula\_peticions*: taula que conté la descripció de cada cançó pendent de reproduir (per cada cançó es guarda la pista i el sector inicial, nombre de sectors i duració). Suposarem que aquesta taula és de mida infinita.

Es demana:

- a) Donat el següent programa principal, programeu les rutines de servei a l'interrupció que cregueu necessàries per a que el sistema funcioni correctament (podeu afegir les variables que vulgueu):

```
typedef struct{
    int pista, sector;
    int numsects;
    int duracio;
}taula;

taula taula_peticions[MAX_CANÇONS];
int tocala_sam = 0;
int ncançons = 0;
int nticks = 0;
/* Aquí podeu afegir les vostres variables */

void main()
{
    void *adr_rel;
    void *adr_mon;
    void *adr_DMA;

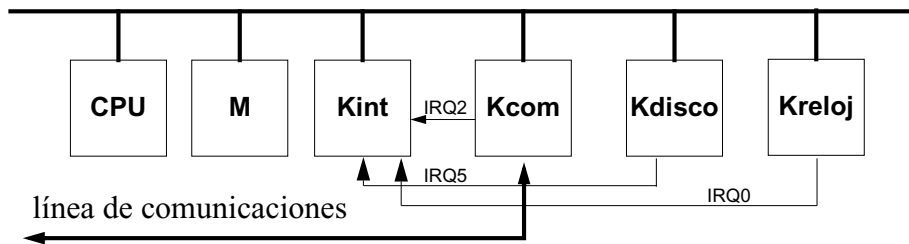
    adr_rel = getvect(8);
    adr_mon = getvect(10);
    adr_DMA = getvect(11);
    setvect(8, (void interrupt (*)())rellotge);
    setvect(10, (void interrupt (*)())moneder);
    setvect(11, (void interrupt (*)())DMA);

    while(1);
    /* El programa només acaba quan es para la màquina. */
}
```

- b) Per a què serveix el vector d'interrupcions? (Contesteu breument - com a molt 5 línies).
- c) Per què no podem passar paràmetres a una rutina de servei a una interrupció? (Contesteu breument - com a molt 5 línies).

## Problema 52

Disponemos de un computador con la siguiente estructura:



Kcom es un controlador de comunicaciones capaz de recibir y transmitir caracteres a través de una línea de comunicaciones. Los registros de este controlador, visibles a nivel Lenguaje Máquina y mapeados en el espacio de direcciones de E/S son:

- *KcomDAT* (8 bits), contiene el código del carácter a transmitir o del carácter recibido por la línea de comunicaciones.
- *KcomEST* (8 bits), El bit 0 indica el motivo por el cual se ha interrumpido a la CPU: (0), ha finalizado la transmisión de un carácter; (1), se ha recibido un carácter por la línea de comunicaciones.
- *KcomCNTRL* (8 bits), para dar la orden de transmisión de un carácter debe generarse un flanco ascendente en el bit 0 (escribir un 0 y luego un 1). El resto de bits de este registro se utilizan para otras tareas no especificadas, en consecuencia no deben modificarse.

Para transmitir un carácter por la línea de comunicaciones se escribirá el código del carácter a transmitir en el registro *KcomDAT* y se dará la orden de transmisión en el bit 0 del registro *KcomCNTRL*. Al finalizar la transmisión de este dato se producirá una interrupción por la línea IRQ2.

Si Kcom recibe un carácter por la línea de comunicaciones, entonces almacena el dato en *KcomDAT* y genera una interrupción por la línea IRQ2.

Kdisco es un controlador de disco duro. Este disco duro tiene 4 caras (0-3), 256 pistas por cara (0-255), 256 sectores por pista (0..255) y cada sector puede almacenar 1024 bytes. Los registros de este controlador, visibles a nivel Lenguaje Máquina y mapeados en el espacio de direcciones de E/S son:

- *KdiscoCARA* (8 bits), *KdiscoPISTA* (8 bits), *KdiscoSECTOR* (8 bits) en donde ha de especificarse la dirección (cara-pista-sector) del sector a transferir.
- *KdiscoDIR* (32 bits), en este registro se indica la dirección del bloque de datos a transferir.
- *KdiscoCNTRL* (8 bits), para leer un sector hay que poner un 1 en el bit 0, para escribir un sector hay que escribir un 1 en el bit 2. La orden de inicio de transferencia se indica poniendo un 1 en el bit 7. El resto de bits de este registro se utilizan para otras tareas no especificadas, en consecuencia no deben modificarse.

Este disco duro funciona por DMA. La transferencia de información es siempre sector a sector. Para leer un sector hay que indicar, en los registros correspondientes, la cara-pista-sector en que se encuentra, la dirección de memoria en dónde se dejará el sector leído, y la correspondiente orden en el registro de control. Cuando se acabe la transferencia del sector, el controlador generará una interrupción por la línea IRQ5. La escritura es similar.

El Kreloj genera N interrupciones por segundo a través de la línea IRQ0.

El Kint funciona tal y como se ha descrito en clase.

**Se pide que escribáis un programa (programa principal y las rutinas de atención al Krel, Kcom y Kdisco) que durante 1 hora lea sectores consecutivos del disco, a partir de la cara 2, pista 9, sector 0 y los transmita por la línea de comunicaciones.**

Notas:

- No hay que programar los vectores de interrupción.
- Disponemos de un buffer de 32 Kbytes para almacenar la información que leemos del disco: `char buffer[32*1024];`

### Problema 53

Es vol programar una aplicació que permeti encriptar un fitxer de dades guardat en el disc. L'aplicació llegirà de disc els sectors que formen el fitxer, els anirà encriptant i els tornarà a escriure a disc en un altre fitxer diferent.

Amb aquest objectiu s'ha programat la rutina `encriptar`, que té la següent interfície:

```
int encriptar (char *adr);
```

Aquesta rutina es capaç d'encriptar TAMSEC bytes guardats des de l'adreça "adr" i deixar-los (un cop encriptats) a la mateixa zona de memòria. La informació ocupa el mateix espai abans i després del procés d'encriptació. La duració de la rutina `encriptar` pot ser menor, igual o superior al temps de lectura o escriptura d'un sector de disc.

En el cas que el procés d'encriptar un determinat sector duri més de 10 segons, s'abortarà l'encriptació del sector i el sector en qüestió no s'encriptarà. Per abortar la rutina "encriptar" es suficient amb

posar a 1 la variable global "abortar\_encriptacio". En aquest darrer cas, el bloc de memòria que es volia encriptar no es veu modificat (queda igual que abans de cridar a la rutina "encriptar"). Si s'aborta, la rutina "encriptar" retorna un 0, altrament retorna un 1.

Es demana:

Programar les rutines de servei (a les interrupcions de rellotge i disc) i el programa principal de manera que es realitzi el procés d'encriptació del fitxer en el mínim de temps possible. Es disposa d'un *buffer* amb capacitat de 4 sectors (`char buffer[4*TAMSEC];`) per aquest propòsit. Si en un moment donat del programa es pot fer una lectura i una escriptura a disc, es donarà prioritat a la lectura. Suposarem que el fitxer a encriptar es troba emmagatzemat a la pista 13, sectors 3 al 18 i que el fitxer encriptat s'ha d'escriure a la pista 33, sectors 1 al 16. Considerarem que les transferències a disc sempre es realitzen sense errors.

# Problemas de microprogramación

## Problema 58

Define el valor de las diferentes señales de la unidad de control, explicada en clase, para una correcta ejecución de la instrucción:

CMP R0, [R2+R3]

Si alguna de las señales no aparece en las diferentes fases de ejecución de la instrucción, supondremos que toma su valor por defecto.

### Fetch:

toA←1 // toB←PC // Amux←0 // ALU←suma // loadMAR←1 // fromC←PC // writeReg←1 // loadPSW←\_\_\_ ///  
f1: RD←1 // AS←\_\_\_ //  $\overline{M}/P$ ←\_\_\_ //  $\overline{L}/E$ ←\_\_\_ // si ready = 0, volver a f1 ///  
Amux←1 // ALU←A // fromC←IR // writeReg←1 // loadPSW←\_\_\_ ///

### Busqueda de operandos:

toA←\_\_\_ // toB←\_\_\_ // Amux←\_\_\_ // ALU←\_\_\_ // fromC←\_\_\_ // writeReg←\_\_\_ // loadPSW←\_\_\_ ///  
toB←\_\_\_ // loadMAR←\_\_\_ ///  
f2: RD←\_\_\_ // AS←\_\_\_ //  $\overline{M}/P$ ←\_\_\_ //  $\overline{L}/E$ ←\_\_\_ // si ready = 0, volver a f2 ///

### Ejecución:

\_\_\_\_\_←\_\_\_ // \_\_\_\_\_←\_\_\_ // \_\_\_\_\_←\_\_\_ // \_\_\_\_\_←\_\_\_ // ir a Fetch ///

## Problema 59

Disponemos de la siguiente instrucción de lenguaje máquina:

INC [R1+]

que, después de hacer el correspondiente incremento, incrementa también el registro utilizado en la indirección. Tened en cuenta que la instrucción de incremento modifica la palabra de estado (PSW). Se pide la especificación del microprograma que la implementa, teniendo en cuenta que el *Data Path* y la Unidad de Control son los explicados en clase. Suponed que las fases de fetch y decodificación ya han sido realizadas.

## Problema 60

Microprogrameu amb el menor nombre de cicles possible la següent instrucció:

ADD [R1], [R2+R3]

en la qual es suma al contingut de l'adreça de memòria [R1] el contingut de [R2+R3]. Tingueu en compte que la instrucció ADD modifica la paraula d'estat.

## Problema 61

Microprogrameu amb el menor nombre de cicles possible la següent instrucció:

ADD [R3+R5], [R4+]

en la que es suma a [R3+R5] el contingut de [R4] i, a més, s'incrementa el valor del registre R4. Tingueu en compte que la instrucció ADD modifica la paraula d'estat.

## Problema 62

Escribiu el microcodi corresponent a l'execució de dues instruccions de resta definides com:

- a) SUB R1, [R2]  
R1 ← M[R2] - R1
- b) RSUB R1, [R2]  
R1 ← R1 - M[R2]

### Problema 63

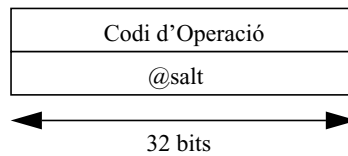
Donada la següent seqüència de senyals de control del processador explicat a classe, es demana que digueu quina instrucció del nivell llenguatge màquina està interpretant (només està escrita la fase d'execució):

```
toB <- SP // loadMAR <- 1 ///
x1: AS <- 1 // RD <- 1 // M/P <- 0 // L/E <- 0 // si READY =0 llavors anar a x1 ///
    AMUX <- 1 // ALU <- A // fromC <- T1 // writeReg <- 1 ///
    toA <- 1 // toB <- SP // ALU <- SUMA // AMUX <- 0 // writeReg <- 1 // fromC <- SP ///
    toB <- R1 // toA <- T1 // loadMAR <- 1 // ALU <- A // AMUX <- 0 // loadMDR <- 1 ///
x2: AS <- 1 // WR <- 1 // M/P <- 0 // L/E <- 1 // si READY =0 llavors anar a x2 ///
```

Es podria haver fet d'alguna altra manera més eficient? En cas afirmatiu, escriviu la seqüència optimitzada.

### Problema 64

Microprogrameu la fase d'execució d'una instrucció de salt incondicional amb el següent format d'instrucció:



### Problema 65

Escriviu el microcodi complet de la instrucció

```
CALL 1500
```

suposant que les fases de “fetch” i decodificació ja s’han realitzat. Teniu en compte que la instrucció ocupa 2 posicions de memòria (una conté el codi CALL i l’altre el valor 1500), i que tant les posicions de memòria com els registres del processador són de 32 bits.