eman ta zabal zazu
universidad del país vasco
euskal herriko unibertsitatea

departamento de electricidad y electrónica
elektrika eta elektronika saila

Examen de S.E.T.I.

1er curso de Ingeniería Electrónica
18 de septiembre de 2002

Cuestiones teórico-prácticas. 1 punto cada una

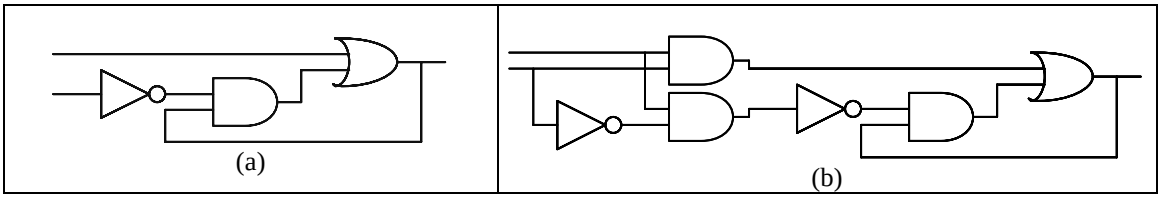
1 Tenemos un fichero que almacena la información enviada por una estación meteorológica clasificando el tiempo de cada día según cuatro categorías: 1-sol, 2-nublado, 3-sirimiri, 4-tormenta. Para un mes concreto el fichero muestra lo siguiente: 31323323342321334233122324323 3.

a) ¿En qué consistiría considerar la extensión de orden 2 de esta fuente (la estación meteorológica) y cómo podría quedar la información del mes mostrado?
b) ¿Se ha reducido la necesidad de capacidad de almacenamiento de datos?
c) ¿Se ha reducido de este modo la entropía?

2 Supongamos que construimos una Máquina de Turing capaz de llevar a cabo la tarea expuesta en el ejercicio anterior. ¿Podremos asegurar de antemano que siempre terminará su ejecución?. En caso afirmativo ¿no contradice esto lo que se conoce como "problema de la parada"?. En caso contrario demuestre (o al menos explique) porqué no es posible.

3 Se han estudiado los formatos más habituales de representación numérica en las máquinas computadoras. Se ha visto que ya al nivel más bajo (en el hardware) se hace una distinción entre números enteros y números reales. Esta distinción de formatos ha de tener, evidentemente, su razón de ser en ciertas ventajas/desventajas para cada caso. Argumente las razones que considere determinantes a la vista de su conocimiento de los formatos.

4 En la figura (a) puede verse un circuito con capacidad de almacenamiento de información. En la figura (b) el mismo circuito es ampliado de modo que se mantiene su función cambiando su modo de operación. ¿En qué consiste este cambio de modo de operación y cual es su ventaja frente al primero?



Mapeo de memoria. 2 puntos

5 Diseñe el circuito necesario para que un computador con microprocesador 68000 disponga de:

a) 4Mbytes de RAM situados a partir de 0 mediante dos pastillas estructuradas en bytes.
b) Un periférico 68230 (32 regs.) situado a partir del punto central del espacio direccionable.
c) 128Kbytes de ROM en el final del espacio direccionable mediante una sola pastilla.

No se impone ninguna restricción sobre la posibilidad de que existan copias de las pastillas en otras posiciones del espacio direccionable aparte, claro está, de las necesarias para que los distintos circuitos no se solapen..

Programación del 68000. 4 puntos (1,5 + 1,5 + 1)

6 Escríbanse las siguientes subrutinas y programa principal:

Subrutina MCD

Esta subrutina deberá tomar dos números naturales (enteros y positivos) y obtener el máximo común divisor de ambos. La subrutina podría tener en concreto la siguiente descripción:

```
; Subrutina:     MCD
; Descripción:   Obtiene el máximo común divisor de dos números naturales
; Entrada:       D0.W  D1.W
; Salida:        D0.W
; Modifica:      … (Lo menos posible)
```

Nota: El máximo común divisor de dos números naturales puede obtenerse mediante el ***Algoritmo de Euclides***:

```
MCD (a,b)
INICIO
  SI   b > a   INTERCAMBIAR(a,b)
  MIENTRAS   b != 0
               aux  ← a
               a  ← b
               b  ← ( aux  %  b )     #  resto de la división
  FMIENTRAS
  MCD  ← a
FIN
```

Subrutina INVSTRING

Esta subrutina deberá tomar la dirección de una cadena de caracteres y devolverla invertida. La subrutina podría tener en concreto la siguiente descripción:

```
; Subrutina:     INVSTRING
; Descripción:   Invierte una cadena
; Entrada:       A0.L  (cadena original)
;                A1.L  (cadena destino)
; Modifica:      ... (Lo menos posible)
```

Programa principal

El programa principal deberá obtener el máximo común divisor de 3245 y 4526, y obtener la cadena inversa a partir de la cadena “Esta es la cadena a invertir”.

El programa principal podría tener en concreto las siguientes partes:

```
cpu       68000
include /usr/local/68k/sermones.inc

org       $1000
; PARA CADA SUBRUTINA:
;    Inicialización de los parámetros de entrada antes de la llamada a subrutina
;    Llamada a SUBRUTINA
; Final

org       $2000
; Declaración de todas las variables/estructuras.
```

Examen de S.E.T.I.

1er curso de Ingeniería Electrónica
18 de septiembre de 2002

SOLUCIONES y COMENTARIOS

nota: Esta pregunta ha sido eliminada de la evaluación del ejercicio ya que no se ha interpretado correctamente la intención. Se pretendía que se hicieran consideraciones al respecto de la longitud de los códigos y de la entropía, y no que se entrara en largos cálculos que no tienen sentido en el planteamiento de este ejercicio.

a) La extensión de orden dos no es otra cosa que la agrupación de símbolos en pares. Para ello se establece un nuevo alfabeto donde cada símbolo representa una secuencia de dos originales. En nuestro caso, y por ejemplo, podemos tomar la secuencia 11 (dos dias seguidos con sol) y codificarla como "a". Siguiendo de este modo estableceremos el siguiente código:

a(11), b(12), c(13), d(14), e(21), f(22), g(23), h(24), i(31), j(32), k(33), l(34), m(41), n(42), o(43), p(44)

Mediante este código la secuencia de un mes indicada resulta: **`ijkglgeknkbghjk`**

b) La necesidad de almacenamiento se ha de medir en binits. Si utilizamos un código binario de tipo bloque para codificar los símbolos originales necesitaremos 2 binits/símbolo (4 posibilidades), mientras que para el segundo alfabeto serán necesarios 4 binits/símbolo (16 posibilidades). Esto unido a las longitudes del mensaje en cada caso nos lleva a una misma cantidad de binits para el mensaje (2*30=4*15).

Otra cosa sería si conociesemos la naturaleza de la fuente (los valores de sus parámetros probabilísticos) y utilizaramos un código optimizado. En este caso sería posible que la extensión de la fuente redujera la necesidad de codificación, como indica el teorema de Shannon, pero el mensaje disponible en el enunciado es insuficiente para caracterizar dicha fuente ya que la distribución de probabilidades será distinta para otro mes del año.
En caso de que se de por buena la distribución de probabilidad observada, se da la casualidad de que la extensión de orden 2 no reduce la necesidad de almacenamiento, y habrá que ir a ordenes superiores para obtener dicho resultado (la entropía de la [supuesta] fuente sería 1.6855[(1)] y la longitud media del código compacto de 1.7 binits por símbolo, quedando lugar para una pequeña reducción por extensión. En el caso de la extensión de orden 2 se obtiene un código compacto de 3,4 binits/símbolo que supone la misma necesidad de almacenamiento)

Símbolo	frecuencia	código
1	1/10	000
2	3/10	01
3	5/10	1
4	1/10	001

$H = 2\ (1/10)\ \log_2(10) + (3/10)\ \log_2(10/3) + (5/10)\ \log_2(10/5) = 1.6855$

$L = 3 * (2/10) + 2 * (3/10) + 1 * (5/10) = 1.7$ binits / símbolo

30 símbolos * 1.7 binits / símbolo = 51 binits

Símb.	Frec.	Código	Símb.	Frec.	Código	Símb.	Frec.	Código	Símb.	Frec.	Código
a	1/100	110000	e	3/100	00001	i	5/100	1110	m	1/100	110010
b	3/100	00000	f	9/100	0001	j	15/100	011	n	3/100	01001
c	5/100	1101	g	15/100	001	k	25/100	10	o	5/100	0101
d	1/100	110001	h	3/100	01000	l	5/100	1111	p	1/100	110011

$L = 6\ (4/100) + 5\ (12/100) + 4\ (29/10) + 3\ (30/100) + 2\ (25/100) = (24+60+116+90+50)/100 = 3{,}4$ binits / símbolo

15 símbolos * 3.4 binits / símbolo = 51 binits

[1] Esto supone considerar la independencia de los símbolos (fuente de memoria nula). En caso de no serlo, habría de modelizarse con un orden suficiente para obtener la entropía, lo que no tiene que ver con la extensión de la fuente, como parece que confunden algunos alumnos.

c) La entropía es una característica de la fuente, por lo que no tiene sentido hablar de cambio de la misma por el mero hecho de hacer cambios de codificación.

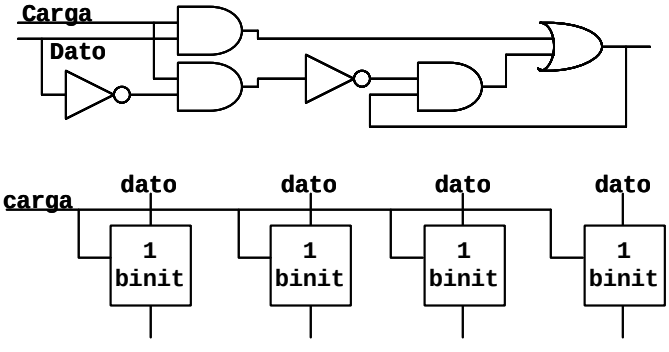
No se trata de considerar una fuente distinta que emite símbolos con información sobre dos días para estimar sus probabilidades (que no coincidirán con las de la estensión). En ese caso se calcula una mayor entropía, como es razonable esperar ya que ésta es la "información media por símbolo" y la segunda fuente emite símbolos "cargados" con [más o menos] el doble de información.

2 Se tratará de una máquina muy simple en su funcionamento, y será trivial demostrar que siempre parará. Esto no contradice el problema de la parada, ya que este "problema" consiste en saber si **cualquier Máquina de Turing** enfrentada a **cualquier secuencia** en su cinta termina la ejecución o no. Este es el problema que se demuestra irresoluble. Es decir, no existe un "mecanismo" general para toda máquina y secuencia al que yo le de mi máquina y secuencia y responda con su vaticinio sobre la parada; pero puede perfectamente existir un mecanismo particular (por ejemplo exclusivamente para mi máquina) que indique si una ejecución termina.

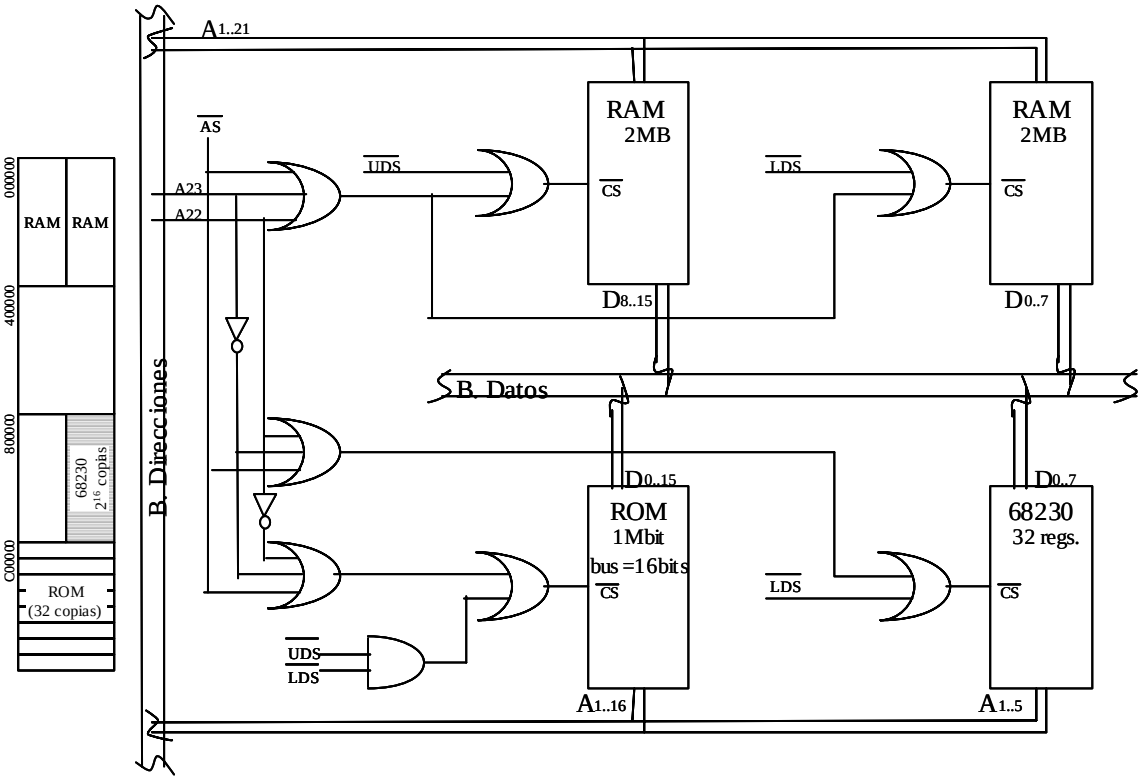
Visto de otro modo, basta considerar que nuestro problema tiene solución, es decir, es computable y por tanto existen Máquinas de Turing que lo resuelven. Esto implica evidentemente que una vez resuelta su actuación sobre la cinta terminan la ejecución.

3 En principio, y desde un punto de vista formal (matemático), los números reales incluyen a los naturales y por tanto una representación de los mismos podría entenderse como suficiente. El hecho de que la representación de datos en las máquinas computadoras sea necesariamente discreta y finita supone que esto no sea así. La aplicación de computadores a ámbitos muy dispares en cuanto a requerimientos numéricos (p.ej. desde la física sub-atómica a la astrofísica) lleva a plantear un formato para los reales capaz de representar valores absolutos muy grandes y resoluciones (distancia entre dos valores) muy pequeñas, pero es imposible (e ineficaz) dedicar un número elevado de binits a disponer de una representación para todo el rango con la resolución máxima. De este modo se tienen representaciones (como la estudiada en la asignatura) que presentan resolución variable, desde muy alta en las proximidades de cero hasta muy baja en las proximidades de los extremos del rango cubierto. En consecuencia esta representación muestra dos inconvenientes para el trabajo con enteros: por un lado, para valores pequeños, se despreciaría la mayor parte de la capacidad de representación, y por otro, para valores grandes, la representación real es incluso incapaz de dar cabida a números enteros al tener una resolución más "gruesa" que la unidad. Es por ello que los números enteros necesitan de una representación propia.

4 El circuito mostrado es capaz de almacenar un binit, y en su versión primera cada línea de entrada establece el estado a uno o a cero mediante un pulso. Si queremos almacenar cantidades de memoria mayores y estructurarlas por bloques (bytes o words de otro tamaño), sería muy contraproducente mantener ese esquema de control. Lo razonable es separar el dato a almacenar (cero o uno) del hecho de almacenarlo en un momento dado (indicándolo con un pulso) de modo que cada celda de información reciba su dato particular y el pulso se comparta entre todas las celdas que formen un mismo bloque. Esta separación de funciones es lo que se consigue con la segunda estructura.



5 Los requerimientos establecidos en este ejercicio coinciden con los dispositivos y su disposición que se mostraban en el examen ejercicio 5 de la convocatoria de junio. Por tanto la solución a este ejercicio es la misma que la vista en aquella ocasión.



Programación del 68000.	4 puntos (1,5 + 1,5 + 1)

6

Subrutina MCD

```
; Subrutina:    MCD
; Descripción:  Obtiene el máximo común divisor de dos números naturales
; Entrada:      D0.W  D1.W
; Salida:       D0.W
; Modifica:     CCR, ( D0, al ser salida)
; Dos posibles soluciones:
```

```
MCD     MOVEM.L D1/D2,-(A7)
        CMP.W   D0,D1
        BLS.S   CONT
        EXG     D0,D1
CONT    BRA.S   WHILE
DO      CLR.L   D2
        MOVE.W  D0,D2
        MOVE.W  D1,D0
        DIVU    D1,D2
        SWAP    D2
        MOVE.W  D2,D1
WHILE   TST.w   D1
```

```
MCD     MOVE.L  D1,-(A7)
        CMP.W   D0,D1
        BLS.S   CONT
        EXG     D0,D1
CONT    SWAP    D0          ; nos aseguramos de que
        CLR.W   D0          ; la parte alta esté en
        SWAP    D0          ; blanco
        BRA.S   WHILE
DO      DIVU    D1,D0       ; a ← (a%b)y(a/b)
        CLR.W   D0          ; borramos la parte baja
        SWAP    D0          ; nos quedamos con el resto
        EXG     D0,D1       ; intercambiamos (a%b)←→b
```

```
        BNE.S   DO              WHILE TST.w   D1
        MOVEM.L (a7)+,D1/D2           BNE.S   DO
        RTS                           MOVE.L  (A7)+,D1
                                      RTS
```

Subrutina INVSTRING

Esta subrutina deberá tomar la dirección de una cadena de caracteres y devolverla invertida. La subrutina podría tener en concreto la siguiente descripción:

```
; Subrutina:     INVSTRING
; Descripción:   Invierte una cadena
; Entrada:       A0.L  (cadena original)
;                A1.L  (cadena destino)
; Modifica:      CCR
```

```
INVSTRING  MOVEM.L D0/A0/A1,-(A7)
           CLR.W   D0
           BRA.S   DO_WHILE
DO_1       ADDQ.W  #1,D0
WHILE      TST.B   (A0)+
           BNE.S   DO_1
           SUBQ.L  #1,A0     ;
           BRA.S   FOR
DO_2       MOVE.B  -(A0),(A1)+
FOR        DBRA    D0,DO
           CLR.B   (A1)
           MOVEM.L (A7)+,A0/A1/D0
           RTS
```

Programa principal

```
        cpu     68000
        include /usr/local/68k/sermones.inc

        org     $1000

        MOVE.W  X1,D0
        MOVE.W  X2,D1
        BSR.S   MCD
        MOVE.W  D0,Y

        LEA     CAD1,A0
        LEA     CAD2,A1
        BSR.S   INVSTRING

        SERMON  FPROG

        org     $2000

X1      DC.W    3245
X2      DC.W    4526
Y       DS.W    1
```

```
CAD1    DC.B    “Esta es la cadena a invertir”
CAD2    DS.B    100
```