eman ta zabal zazu
universidad euskal herriko
del país vasco unibertsitatea

departamento de electricidad y electrónica
elektrika eta elektronika saila

Examen de S.E.T.I.

1er curso de Ingeniería Electrónica
1 de septiembre de 2003

Cuestiones teórico-prácticas. — 1 punto cada una

1 Un hábil tahúr juega a los dados con la siguiente estrategia tramposa: dispone de dos juegos de dados (un juego de dados es simplemente un par); uno de ellos esta trucado de modo que la probabilidad de sacar un doble seis es 1/3 (las demás combinaciones se reparten los 2/3 de probabilidad restantes) mientras que el otro juego es normal (todas las combinaciones equiprobables); para no levantar demasiadas sospechas con series de dobles seises poco razonables, cada vez que obtiene dicha combinación realiza las dos siguientes tiradas con el juego de dados normal, para volver a utilizar a continuación de nuevo el juego "cargado".

La obtención de resultados de cada tirada de dados puede verse como la emisión de un símbolo por una fuente, y esta fuente es conocida para nosotros ya que se ha descrito la estrategia del jugador. Dibuje la fuente de Markov correspondiente (con sus estados, transiciones y probabilidades correspondientes). ¿Cual es el orden de dicha fuente?

Notas.: Emplee dos símbolos cualesquiera para representar la combinación ganadora (doble seis) y la agrupación de todas las demás (p.ej. "0" y "1") . No se pide ningún valor de información ni entropías: sólo la representación del modelo y su orden.

2 Sin extenderse demasiado (no mas de una cara con un tamaño de letra razonable) diga lo que sea esencial respecto al "problema de la parada": en qué consiste, a qué debe su importancia, cómo se demuestra,...

3 El computador virtual visto en la asignatura no tiene capacidad interna para multiplicar. Razone qué podria hacerse para dotarle de tal capacidad, indicando todos los cambios que serian necesarios en el hardware.

Nota.: Evidentemente no hay una solución única. Cualquiera es válida si está razonablemente planteada. Posiblemente sepa el alumno que el mismo caso se da en el núcleo del procesador 68000 y la capacidad de multiplicación que éste tiene es resultado de una estrategia que, entre otras, puede ser válida también aquí (no es necesariamente la más simple de plantear).

4 Explique para qué sirve, en el procesador 68000, la capacidad de procesamiento de excepciones.

Mapeo de memoria. — 2 puntos

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

a) 4Mbytes de RAM al final del espacio de memoria mediante dos pastillas estructuradas en bytes.
b) Un periférico 68230 (32 regs.) situado a partir de 600000hex.
c) 128Kbytes de ROM a partir de la dirección 0 mediante una sola pastilla.

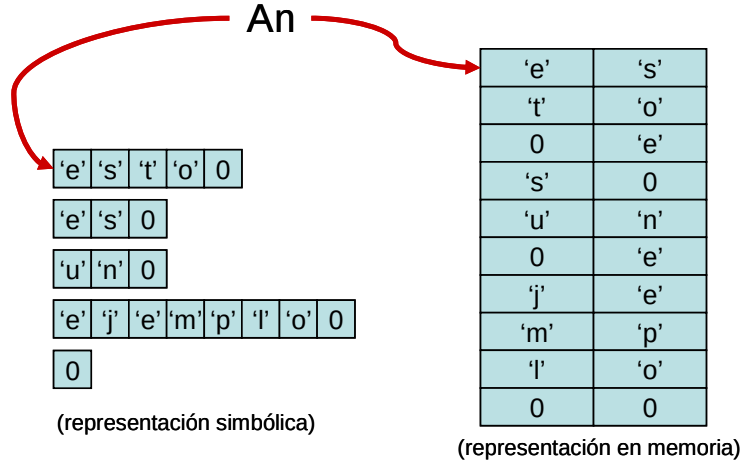
No se impone ninguna restricción sobre la posibilidad de que existan copias de las pastillas en otras posiciones del espacio direccionable aparte, claro está, de las necesarias para que los distintos circuitos no se solapen..

Programación del 68000. — 4 puntos (1.25 + 1.25 + 1.5)

6 Una forma sencilla de reconocer a qué idioma pertenece un texto es mirar si sus palabras aparecen en un diccionario de dicho idioma. Sin embargo, por un lado no podemos exigir que todas las palabras aparezcan en el diccionario (siempre nos encontraremos con palabras que pertenecen a la lengua

y no aparecen en él), y por otro, una palabra puede pertenecer a más de un idioma. Es por ello que una solución adecuada sería la de contar cuantas palabras del texto aparecen en el diccionario: el idioma cuyo dicconario contenga más apariciones de palabras del texto será, presumiblemente, la lengua en la que esté escrito.

En nuestro caso concreto, supondremos que tanto un texto como un diccionario viene dado por una secuencia de ***cadenas de caracteres*** que concluye con una cadena vacía:



Supondremos además que existen (y por tanto NO DEBEMOS CREARLAS) dos rutinas que nos serán útiles:

```
; Subrutina:      STRLEN
; Descripción:    calcula la longitud de una cadena de caracteres
; Entrada:        A0.L  dirección de la cadena
; Salida:         D0.L  longitud de la cadena
; Modifica:       sólo CCR y D0

; Subrutina:      STRCMP
; Descripción:    compara dos cadenas
; Entrada:        A0.L  dirección de la primera cadena
;                 A1.L  dirección de la segunda cadena
; Salida:         D0.B:   1 ⇔ cad1 > cad2
;                         0 ⇔ cad1 = cad2
;                         -1 ⇔ cad1 < cad2
; Modifica:       sólo CCR y D0
```

Se pide que se diseñen las siguientes subrutinas y programa principal:

Subrutina BUSCACAD

Esta subrutina deberá tomar la dirección de una palabra y un diccionario, y descubrir si dicha pala está contenida en el diccionario. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:      BUSCACAD
; Descripción:    Busca una palabra en un diccionario
; Entrada:        A0.L  dirección de la palabra
;                 A1.L  dirección de la primera palabra del diccionario
; Salida:         D0.B      1 ⇔ el diccionario contiene la palabra
```

```
;                       0 ⇔ el diccionario no contiene la palabra
; Modifica:     (rellenar debidamente)
```

Subrutina CUENTACAD

Esta subrutina deberá tomar la dirección de un texto y un diccionario, y realizar la cuenta de apariciones de las palabras del texto en el diccionario. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:    CUENTACAD
; Descripción:  Cuenta cuantas palabras del texto están contenidas
;               en el diccionario
; Entrada:      A0.L  dirección de la primera palabra del texto
;               A1.L  dirección de la primera palabra del diccionario
;               A2.L  dirección del contador
; Salida:
; Modifica:     (rellenar debidamente)
```

Programa principal

El programa principal deberá definir un texto que contenga al menos 5 palabras, así como tres diccionarios (sactellano, euskera e ingles) que contengan al menos otros cuatro terminos cada uno. Además, deberá realizar la cuenta de apariciones de las palabras del texto en cada idioma y guardarlos en tres contadores diferentes. El programa principal ***podría*** tener en concreto las siguientes partes:

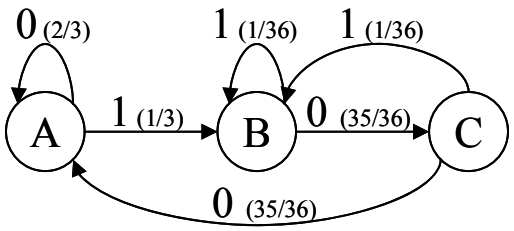
```
cpu     68000
include /usr/local/68k/sermones.inc

org     $1000
; Para cada uno de los diferentes diccionarios:
; Inicialización de los parámetros de entrada antes de la llamada a subrutina
; Llamada a CUENTACAD
; Final

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

Examen de S.E.T.I.

1er curso de Ingeniería Electrónica
1 de septiembre de 2003

SOLUCIONES Y COMENTARIOS

1 En la figura puede verse el modelo correspondiente al enunciado. En cada tirada de un par de dados, puede obtenerse un doble seis, que aquí se representa con el 1, o cualquiera otra configuración que se representa con el 0.



0 = 6/6, 1=el resto de combinaciones

El estado A corresponde al estado en que el tahúr utiliza sus dados "cargados", de modo que la probabilidad de obtener un doble seis (1) es de 1/3 y le hace salir de ese estado para pasar a otro en que usará dados "legales"(B). Si por el contrario no obtiene un doble seis (0), continuará usando sus dados trucados (A).
Una vez que obtiene un doble seis, la estrategia consiste en hacer al menos dos tiradas con dados "legales", por lo que el estado (B) recoge la situación en que se acaba de obtener un doble seis (todas las flechas etiquetadas como 1 llegan a a él), A partir de este estado se obliga a que al menos se realicen dos tiradas sin trampa gracias a la existencia del el estado (C) que permite llegar al estado en que volverá a hacer trampas (A) en dos pasos.

2

- El "problema de la parada" consiste en definir una máquina de Turing que tomando como entrada dos secuencias que representen a una maquina cualquiera de Turing y a un problema cualquiera (es decir, para toda máquina y para todo problema) sea capaz de vaticinar si la ejecución de dicha máquina con dicho problema llegará a término o por el contrario su ejecución será infinita en el tiempo.
- La importancia de este "problema" radica en que es un ejemplo de problema sin solución y a que la demostración de este extremo es muy sencilla.
- Esta demostración se realiza por "reducción al absurdo" planteando la existencia de dicha máquina (H) y construyendo otra (K) para la cual se comprueba que (H) es incapaz de predecir correctamente el comportamiento, llegando a un absurdo que invalida la única hipótesis de que existe una máquina (H) con la capacidad especificada.
  La construcción de la máquina (K) es extremadamente sencilla ya que consiste en incluir en su especificación a la máquina (H) y, dado que (K) puede "conocer" el vaticinio de (H), añadir lo necesario para llevarle la contraria. De este modo la aplicación del problema (k≡K) a la máquina (K) incluye como una fase intermedia el análisis por (H) del comportamiento de la máquina (K) con el problema (k), es decir, justo lo que se esta ejecutando, y por construcción la ejecución será contraria a la determinación de (H).
  (nota.- La representación gráfica y concreta de la construcción de esta máquina puede encontrarla el alumno en las copias de las proyecciones realizadas en el aula)

3 En principio pueden ocurrirse dos vías para dotar a nuestro computador de la capacidad de multiplicar: añadir hardware específico a la ALU, o añadir una pequeña zona de memoria interna conteniendo una rutina de multiplicación basada en las capacidades hardware ya disponibles. Ninguna de las dos es trivial en cuanto al número de cambios a realizar, aunque la primera (la que consiste en puramente hardware) puede ser más clara. La segunda es la implantada en algunos procesadores como es

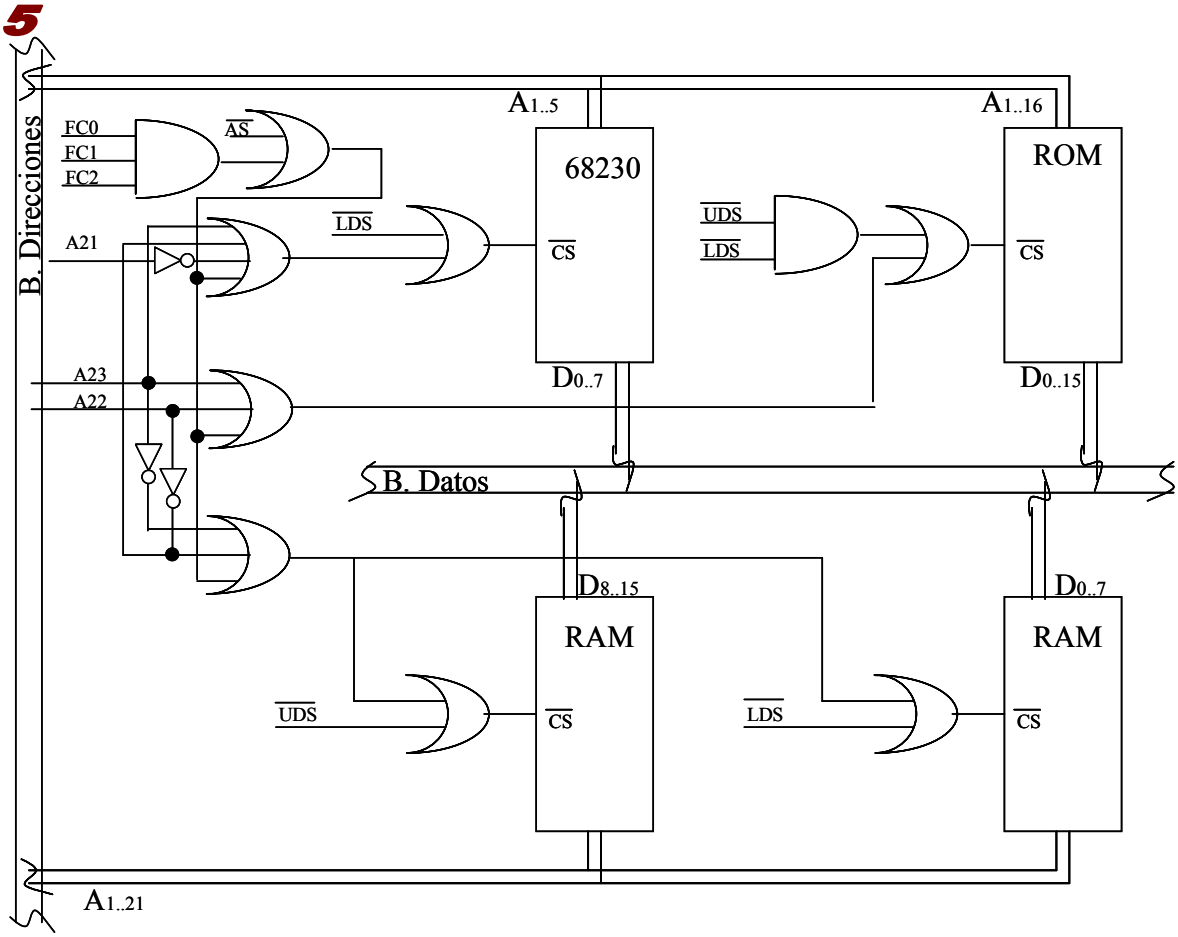
el caso del 68000. Se denomina a este tipo de estrategia (meter programas en el hardware) inclusión de "microcódigo".

Para la primera opción sería necesario simplemente diseñar el circuito multiplicador e incluirlo en la ALU; dotar a esta de una señal más para activar esta nueva operación; y establecer un código de operación y su secuencia de pasos, lo que influye en el diseño de la función combinacional de señales.
Todo esto no tiene excesiva dificultad, y frente a la segunda opción sólo tiene como aspecto negativo el hecho de que un circuito multiplicador requiere mucho hardware frente a lo que se esta utilizando en nuestra máquina virtual: sería un tanto desproporcionado en este sentido.

La segunda opción (el microcódigo) podría llevarse a cabo incluyendo la cantidad de memoria necesaria para alojar una rutina de multiplicación dentro del procesador y hacer que esta memoria fuese accesible por el contador de programa, de modo que al encontrar la operación de multiplicar el contador de programa pase a esta posición interna y se ejecute la rutina. También será necesario almacenar la dirección de la siguiente instrucción para poder ejecutarla una vez ejecutado el microcódigo.
(nota.- puede ser interesante detallar un poco más estas opciones gráficamente, pero en todo caso no se pedía una solución en la que se llegara hasta el último detalle)

4 En el procesador 68000, como en cualquier otro, el procesamiento de excepciones sirve para permitir **interrumpir y suspender temporalmente la ejecución que el procesador este llevando a cabo** en un momento dado, de un modo (en términos generales) asíncrono, es decir, en momentos no previstos de antemano, **para pasar a realizar otra tarea** (normalmente distinta). Este mecanismo puede ser provocado por hardware o por software, permitiendo implementar con eficacia multitud de técnicas de gran utilidad en un sistema basado en microprocesador como pueden ser **la multitarea, los servicios de sistema operativo, la depuración de código instrucción por instrucción, la simulación de instrucciones, el control de situaciones de error de diversos tipos, la atención a dispositivos externos, etc**.

5



El esquema mostrado es una solución al enunciado planteado en la que, tanto para la pastilla 68230 como para la ROM se producen numerosas copias.

Para la RAM se exige que A23 y A22 estén a 1 con lo que quedan al final del espacio direccionable sin réplica alguna ya que su tamaño es de 2Mb cada una y ello requiere 21 lineas de direccionamiento.

La ROM se sitúa al comienzo del espacio direccionable exigiendo que A23 y A22 estén a 0. Como su tamaño es de 128 Kb, y se estructura en 64 Kwords tiene 16 líneas de dirección que se conectarán a A1..A16 y quedarán sin determinar las que van de A17 a A21, es decir 5 líneas. Por tanto se producirán 32 copias de la ROM.

Finalmente, la 68230 tiene tan solo 5 líneas de dirección y determinamos su posición con las tres líneas más altas A23-A22-A21 exigiendo una configuración 0-1-1 (segmentos $6XXXXX_{hex}$ y $7XXXXX_{hex}$). El número de líneas sin fijar es de 15 y en consecuencia se encontrarán 2^{15} copias de este circuito.

6

```
; Subrutina:      BUSCACAD
; Descripción:    Busca una palabra en un diccionario
; Entrada:        A0.L  dirección de la palabra
;                 A1.L  dirección de la primera palabra del diccionario
; Salida:         D0.W:   1 ⇔ el diccionario contiene la palabra
;                         0 ⇔ el diccionario no contiene la palabra
; Modifica:       CCR, D0

BUSCACAD  MOVEM.L  A1, -(A7)
          BRA.S    WHILE          ; mientras no lleguemos al final del diccionario…
DO        BSR.S    CMPSTR         ; si encontramos la palabra terminamos
          TST.B    D0
          BEQ.S    EXISTE
WHILE_DO  TST.B    (A1)+          ; pasamos a la siguiente palabra
          BNE.S    WHILE_DO       ; del diccionario
WHILE     TST.B    (A1)
          BNE.S    DO
          CLR.W    D0
          BRA.S    FIN
EXISTE    MOVEQ    #1, D0
FIN       MOVEM.L  (A7)+, A1
          RTS


; Subrutina:      CUENTACAD
; Descripción:    Cuenta cuantas palabras del texto están contenidas
;                 en el diccionario
; Entrada:        A0.L  dirección de la primera palabra del texto
;                 A1.L  dirección de la primera palabra del diccionario
;                 A2.L  dirección del contador
; Salida:
; Modifica:       CCR, el contenido del contador

CUENTACAD MOVEM.L  D0/A0 , -(A7)
          CLR.W    (A2)
          BRA.S    WHILE          ; mientras no lleguemos al final del diccionario…
DO        BSR.S    BUSCACAD       ; si la palabra está en el diccionario
          ADD.W    D0, (A2)       ; el contador se incrementará
          BSR.S    STRLEN         ; pasamos a la siguiente palabra
          LEA      1(A0,D0.L), A0 ; del texto
WHILE     TST.B    (A0)
          BNE.S    DO
FIN       MOVEM.L  (A7)+ , D0/A0
          RTS
```

```
; PROGRAMA PRINCIPAL
; → Toma el texto y guarda la cuenta de
;     cada diccionario

                CPU         68000
                INCLUDE     /usr/local/68k/sermones.inc

                ORG         $1000

INICIO          LEA         TEXTO , A0
                LEA         CAST, A1
                LEA         N_CAST, A2
                BSR.S       CUENTACAD
                LEA         EUSK, A1
                LEA         N_EUSK, A1
                BSR.S       CUENTACAD
                LEA         INGL, A1
                LEA         N_INGL, A1
                BSR.S       CUENTACAD
FIN             SERMON      FPROG



                ORG         $2000

TEXTO           DC.B        “Este”
                DC.B        “es”
                DC.B        “el”
                DC.B        “texto”
                DC.B        “que”
                DC.B        “sera”
                DC.B        “analizado”
                DC.B        0

CAST            DC.B        “a”
                DC.B        “el”
                DC.B        “que”
                DC.B        “no”
                DC.B        0
EUSK            DC.B        “da”
                DC.B        “izan”
                DC.B        “ez”
                DC.B        “eta”
                DC.B        0
INGL            DC.B        “a”
                DC.B        “of”
                DC.B        “the”
                DC.B        “no”
                DC.B        0

N_CAST          DC.W        0
N_EUSK          DC.W        0
N_INGL          DC.W        0
```