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
26 de enero de 2004

Cuestiones teórico-prácticas. 1 punto cada una

1 El sorteo de "el gordo de la primitiva" es un generador de símbolos que emite cada domingo. ¿Conseguiremos alguna "ventaja" para participar en dicho sorteo si construimos un modelo de Markov?. Explique cuales serán los símbolos, y cómo será el modelo de Markov que podremos formar a partir de la secuencia temporal de resultados (orden del mismo y "aspecto" gráfico).

(nota.- Cada sorteo consiste en la extracción de seis números entre el 1 y el 49. No tendremos en cuenta la existencia de un "complementario" ni de un "reintegro")

2 Supongamos que le proponen a usted formar parte de una nueva empresa que se dedique a la fabricación de software basado siempre en un núcleo de funcionamiento consistente en una Máquina de Turing específica para el problema a resolver. ¿Intentaría usted convencer a sus colegas de que dicho empeño estaría abocado al fracaso porque no es posible generar dichas máquinas con la seguridad de que resuelvan siempre cualquier problema para el que se hayan diseñado?, o, si no es exactamente esto, ¿tiene alguna implicación el "problema de la parada" que le lleve a usted a hacer desistir en su empeño a sus colegas?

3 El microprocesador virtual visto en la asignatura presenta recursos muy limitados, pero se ha presentado como suficiente para llevar a cabo cualquier tarea. Esta afirmación se ha realizado considerando al margen que dicho sistema no dispone de mecanismo de interrupciones. Si consideramos esta carencia, ¿podremos mantener la afirmación?. Dicho de otro modo: ¿Es el mecanismo de interrupciones/excepciones imprescindible para que un procesador sea "de proposito general"?. Independientemente de que sea aplicable a cualquier problema, carecer de esta capacidad, ¿qué limitaciones funcionales nos impone? (cosas concretas que no podremos hacer).

(nota.- hay términos en el enunciado que no se han definido formalmente, por lo que no se espera una respuesta determinada, sino una reflexión al respecto que sea "razonable" o "convincente" independientemente de que "defienda" una opinión o la contraria. Evidentemente no se valorara la respuesta por su extensión sino por su consistencia)

4 ¿Qué quiere decir que un microprocesador sea síncrono o asíncrono internamente? ¿Qué ventaja principal tiene uno sobre sobre otro? (no hablar de transferencias de datos síncronas/asíncronas)

Mapeo de memoria. 2 puntos

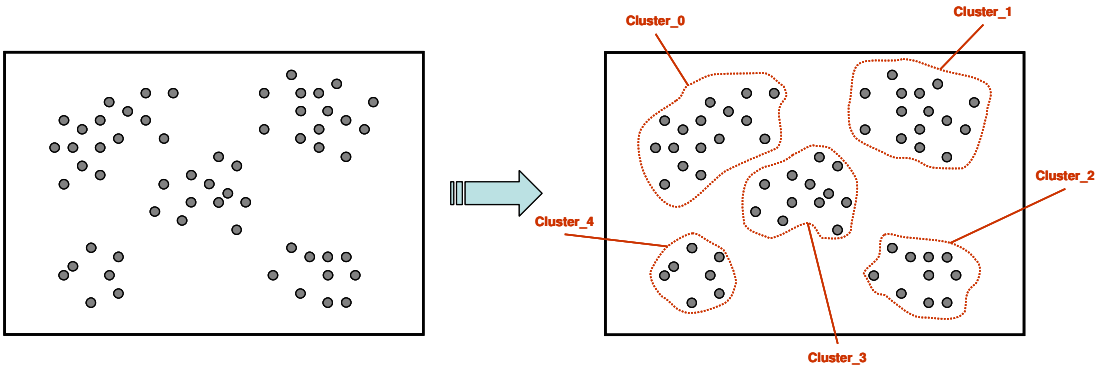
5 Diseñe el circuito necesario para que un computador con microprocesador 68000 disponga de:

a) 4Mbytes de RAM exactamente centrada en el espacio direccionable de memoria mediante pastillas de 8Mbits estructuradas en bytes.
b) 128Kbytes de ROM mediante una sola pastilla que sean accesibles como extremos del espacio direccionable (es decir, deben verse al menos 2 copias comenzando una de ellas en 000000_{Hex} y terminando otra en $FFFFFF_{Hex}$.

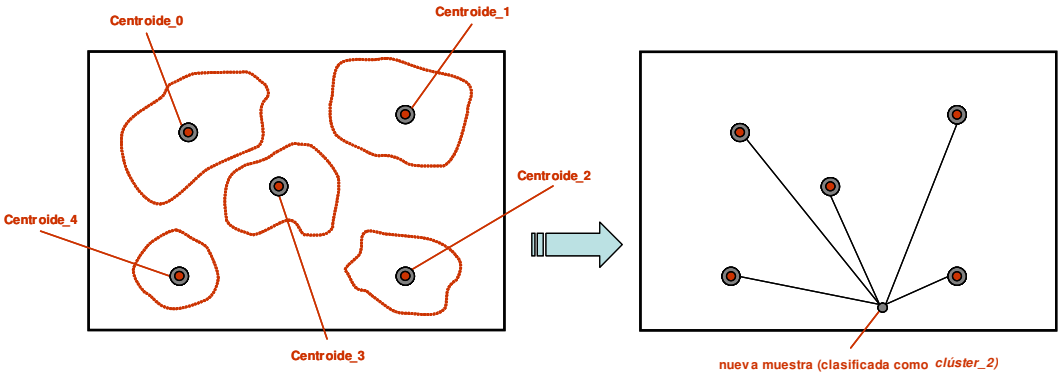
Puede permitirse que se "vean" copias adicionales, pero en todo caso debe quedar libre al menos la mitad del espacio direccionable.

Programación del 68000. 4 puntos (1 + 2 + 1)

6 El *clustering* es una técnica que permite la clasificación de muestras (sonidos, imágenes o cualquier otra información que pueda ser codificada). Básicamente trata de encontrar un número mínimo de subconjuntos que representen suficientemente la variabilidad que las muestras puedan tener:



Una vez se han obtenido dichos subconjuntos, se debe definir una función de distancia entre un cluster y una muestra cualquiera. Dicha distancia será la que se utilizará para clasificar nuevas muestras. Una de las técnicas más sencillas es la de definir un solo elemento o *centroide* que representa al clúster (típicamente será el valor medio de todas las muestras pertenecientes al clúster). En tal caso, la distancia al clúster se calcula mediante la distancia al centroide de dicho clúster. Una nueva muestra será clasificada como perteneciente a aquel cluster cuya distancia es mínima:



En este problema, supondremos que las muestras son valores unidimensionales (en el ejemplo de las figuras eran bidimensionales). Cada muestra vendrá determinada por un **número entero sin signo de 32 bits**. Supondremos que la tarea de crear los clústers ya ha sido realizada, de tal forma que existe un fichero (llamado *centroides.data*) que contiene **256** centroides y que podremos incluir en nuestro programa:

```
CENTROIDES  DC.L   125
            DC.L   368
            DC.L   8230
            DC.L   10263
            DC.L   20323
            DC.L   78450
            DC.L   763872
            DC.L   2987987
            DC.L   7238479
            …
            …
```

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

Subrutina DISTANCIA

Esta subrutina deberá calcular la distancia entre una muestra y un centroide. Ambos datos, al igual que el resultado de la subrutina, son números enteros de 32 bits sin signo. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:      DISTANCIA
; Descripción:    distancia entre una muestra y un centroide
; Entrada:        D0.L  muestra
;                 D1.L  centroide
; Salida:         D1.L : distancia
; Modifica:       (rellenar debidamente)
```

Subrutina CLASIFICA

Esta subrutina deberá tomar un vector de centroides y una muestra, devolviendo el índice del cluster al que pertenece la muestra (el índice del centroide frente al cual la distancia es mínima). Dado que contamos con 256 centroides, los indexaremos de 0 a 255, y el resultado será de tamaño byte. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:      CLASIFICA
; Descripción:    Clasifica una muestra a partir de un vector de 256 centroides.
; Entrada:        A0.L  inicio del vector de centroides
;                 D0.L  muestra a clasificar
; Salida:         D0.B : índice del centroide más cercano
; Modifica:       (rellenar debidamente)
```

Programa principal

El programa principal deberá importar (incluir) los centroides contenidos en el fichero "centroides.data", y clasificar las siguientes 5 muestras: *172,4347,3452353,4343344,54*. Tanto las muestras como el resultado de su clasificación deberán guardarse en memoria. El programa principal ***podría*** tener en concreto las siguientes partes:

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

org     $1000
; para cada una de las cinco muestras:
;   Inicialización de los parámetros de entrada antes de la llamada a subrutina
;   Llamada a CLASIFICA
;   Guardar resultado
;Final

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

eman ta zabal zazu
universidad del país vasco
euskal herriko unibertsitatea
departamento de electricidad y electrónica
elektrika eta elektronika saila

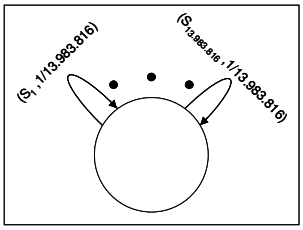
Examen de S.E.T.I.

1er curso de Ingeniería Electrónica
26 de enero de 2004

SOLUCIONES Y COMENTARIOS

1 No tiene sentido plantear tal modelo ya que la distribución de probabilidad en cada sorteo es conocida a priori como equiprobable e independiente de los resultados anteriores, es decir, el proceso es de memoria nula, y además en cada sorteo todas las combinaciones de números son equiprobables. No es posible obtener ninguna ventaja porque, aunque construyamos el modelo, éste no hará otra cosa que reflejar esa equiprobabilidad de cada símbolo en todo instante.

No es preciso realizar el cálculo para determinar las características del modelo, pero podemos ver que el número de combinaciones posibles es de 13.983.816.



$$Comb(49,6) = 49! / (6!\ (49 - 6)!) = 13.983.816$$

Este es el tamaño del vocabulario de símbolos emitidos por nuestro hipotético modelo. La probabilidad de que se dé una de estas combinaciones en concreto (cualquiera de ellas) será de 1/13.983.816.

Algunos alumnos (pocos) han mencionado una posible (¿cuan probable?) circunstancia que es expresada con claridad en la siguiente frase (sic.): "Aunque todas las bolas del sorteo son iguales, no son perfectamente idénticas en sus características físicas como tamaño, volumen, densidad, masa, centro de inercia... y por tanto siempre tendrán distintas probabilidades de aparición en el sorteo". Aunque la premisa es cierta sin duda, la conclusión no resulta evidente, y en todo caso no se trataba de entrar a valorar esta circunstancia.

Para ver qué hay de cierto en ello sería necesario establecer el modelo a partir de la secuencia de resultados, y dada la amplitud del vocabulario frente a la "velocidad" de emisión de símbolos sería imposible en la práctica llegar a obtener un modelo razonable.

2 No cabe considerar el problema de la parada en el planteamiento propuesto. Dicho problema no es otra cosa que una demostración de la existencia de problemas si solución. Evidentemente si un problema no tiene solución no hay mecanismo alguno capaz de resolverlo (si bien es cierto que esto se basa en la conjetura que establece la equivalencia entre existencia de solución y computabilidad así como que la MT cubre todo el espacio de "lo computable", en todo caso nada hace sospechar que esta conjetura sea falsa)

La empresa en cuestión, como cualquier otra, pretenderá construir sistemas que solucionen problemas resolubles, y en ese sentido la Máquina de Turing es un sistema capaz de llevar a cabo la tarea al ser capaz de resolver cualquier problema computable (es decir, con solución).

Por tanto no procedería hacer desistir en la empresa por el motivo apuntado en el enunciado de la cuestión. No obstante, es cierto que puede argumentarse que la eficacia computacional de la Máquina de Turing frente a otros mecanismos más elaborados juega en su contra, así como la dificultad del diseño de las reglas frente a estos otros mecanismos que disponen de potentes técnicas de diseño.

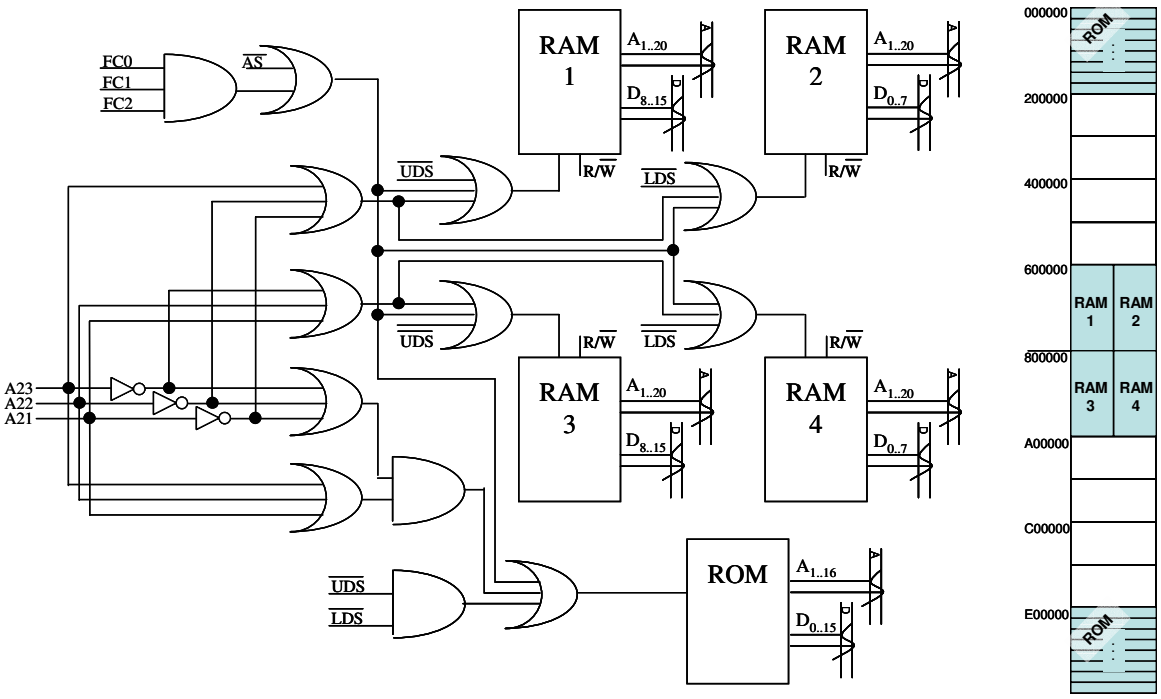
3 No disponer de interrupciones (excepciones en general) implica que la programación del sistema lo es para una sola tarea, lo que no supone una limitación respecto a la generalidad de su capacidad de resolver problemas. Como referencia podemos fijarnos en que el modelo computacional más simple, y capaz de resolver cualquier problema con solución -la Máquina de Turing- no incluye ningún mecanismo semejante a las excepciones.

No obstante, la existencia de un mecanismo de excepciones permite programar más de una tarea con una cierta independencia entre ellas (siempre habrá que tener en cuenta algunas consideraciones), que incluso puede ser "simulado" -con extrema dificultad- en un sistema sin dicho mecanismo. El problema principal para dicha "simulación" consiste en que en la programación de todas y cada una de las unidades de código que se escriban será preciso ceder el procesador voluntariamente a una rutina que se encargue de gestionar la multitarea. Dicha "simulación" tendrá en todo caso desventajas adicionales sobre el mecanismo de excepciones como son la carga computacional que supone atender a tareas que no precisan procesador (p.ej. periféricos sin actividad) o reacciones más retardadas frente a situaciones a atender. Obsérvese que estas desventajas sólo se refieren en definitiva al tiempo, de modo que en el marco teórico de la computabilidad no tienen ningún significado: basta con que el procesador sea "suficientemente rápido".

4 Esta cuestión se mencionó de modo muy somero en el tema sobre arquitecturas alternativas, por lo que sólo se ha pretendido una respuesta muy básica centrada en la idea de que el retardo presentado por el circuito más lento de un procesador síncrono impone el desaprovechamiento de tiempo al sincronizar la atención a la salida de otros más rápidos con una señal de reloj. En la respuesta dada a continuación se introducen algunos detalles añadidos para reforzar el concepto.

Los microprocesadores que utilizamos hoy en día son síncronos en su funcionamiento interno. Esto quiere decir que utilizan una señal periódica (que llamamos "reloj" (clock)) para marcar los momentos en que se llevan a cabo ciertas acciones. Estas acciones son básicamente copias de unos registros en otros reorganizando su estado o bien situando estos datos a la entrada de circuitos que los elaboraran de algún modo (sumadores, desplazamientos, etc.) y recogiéndolos de su salida, como hemos visto en la asignatura con el procesador virtual. Los circuitos que elaboran los datos necesitan un determinado tiempo para llevar a cabo su acción debido a que están formados por dispositivos que requieren un tiempo para conmutar (transistores). Cada uno de estos circuitos tiene una configuración diferente dando lugar a distintos "tiempos de reacción" que vienen dados por el número máximo de elementos en serie que presenten. En consecuencia, en un sistema síncrono, el circuito que necesite el máximo tiempo para presentar una salida estable será el que marque el mínimo periodo (la máxima frecuencia) para la señal que determina los instantes de actuación. Esto tiene como consecuencia que las reacciones más rápidas de otros circuitos son desaprovechadas al esperarse a tomar sus salidas en sincronización con el "reloj". Para no desaprovechar esta circunstancia se plantean como una arquitectura alternativa los procesadores asíncronos, que no siguen el comportamiento pautado por una señal de reloj sino que establecen otros mecanismos más localizados en cada circuito. Esto supone una mayor complejidad en el diseño pero proporciona varias ventajas además del aumento en la velocidad de procesamiento (p.ej. requieren menos potencia, generan menos interferencias electromagnéticas, etc). No obstante, la complejidad de diseño que presentan y el cambio que suponen frente a lo que actualmente se utiliza, esta haciendo que este tipo de procesadores no pasen de ser, hoy por hoy, otra cosa que prototipos en los laboratorios de los diversos fabricantes (IBM, INTEL, SUN,...), si bien es cierto que muchos procesadores actuales (sin ir más lejos el Pentium IV) incluyen algunos componentes que se relacionan asíncronamente, lo que indica que es posible que estas técnicas ganen terreno pero no se produzca una sustitución drástica de una tecnología por otra sino más bien un "desplazamiento" paulatino.

5



6

```
; Subrutina:    DISTANCIA
; Descripción:  distancia entre una muestra y un centroide
; Entrada:      D0.L  muestra (número de 32 bits Sin Signo)
;               D1.L  centroide (número de 32 bits Sin Signo)
; Salida:       D1.L : distancia (número de 32 bits Sin Signo)
; Modifica:     CCR

DISTANCIA MOVE.L  D0, -(A7)
IF        CMP.L   D0, D1
          BHS.S   CONT
ELSE      EXG     D0, D1
CONT      SUB.L   D0, D1
          MOVE.L  (A7)+, D0
          RTS
```

```
; Subrutina:    CLASIFICA
; Descripción:  Clasifica una muestra a partir de un vector de 256 centroides.
; Entrada:      A0.L  inicio del vector de centroides
;               D0.L  muestra a clasificar
; Salida:       D0.B : índice del centroide más cercano
; Modifica:     CCR

CLASIFICA MOVEM.L  D1-D4, -(A7)
          ADDA.L   #1024, A0          ; Nos colocamos más allá del final del vector
          MOVE.W   #256, D2           ; D2 hará las funciones de índice
          MOVE.L   #$FFFFFFFF, D4     ; D4 hará las funciones de distancia mínima
          BRA.S    FOR                ; for (D2=255 ; D2 != -1 ; D2-- )  {
DO        MOVE.L   -(A0), D1          ;   Cargamos un centroide
          BSR.S    DISTANCIA          ;   Calculamos la distancia entre la muestra y el centroide.
IF        CMP.L    D4, D1             ;   Si la distancia es menor que la mínima distancia
          BHS.S    FOR                ;   hasta el momento,
ELSE      MOVE.L   D1, D4             ;   actualizamos la distancia mínima
          MOVE.B   D2, D3             ;   y guardamos el índice del centroide
FOR       DBRA     D2, DO             ; }
          MOVE.B   D3, D0             ; En D3 ha quedado el índice den centroide de dist. min.
          MOVEM.L  (A7)+, D1-D4       ; A0 no es modificado…
          RTS
```

```
; PROGRAMA PRINCIPAL

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

                org         $1000

INICIO          LEA         MU_ORIG, A1
                LEA         MU_CLAS, A2
                MOVE.W      NUM, D1
                LEA         CENTROIDES, A0
                BRA.S       FOR
DO              MOVE.L      (A1)+, D0
                BSR.S       CLASIFICA
                MOVE.B      D0, (A2)+
FOR             DBRA        D1, DO
                SERMON      FPROG

                org         $2000
                include     centroides.data

NUM             DC.W        5
MU_ORIG         DC.L        172
                DC.L        4347
                DC.L        3452353
                DC.L        4343344
                DC.L        54
MU_CLAS         DS.B        5
```