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
13 de septiembre de 2006

Cuestiones. 1 + 1 puntos

1 El problema de la parada: en qué consiste, su interés, y cómo se demuestra su irresolubilidad.

2 ¿Puede utilizarse una máquina de Turing para "aprender" un modelo de Markov de una secuencia de símbolos que situemos en su cinta? Razone su respuesta, y en caso afirmativo explique en qué consistiría esto, es decir cómo plantearía la solución (sin llegar a desarrollar la máquina).

Ejercicio de C. 1,25 puntos

3 Analice esta rutina y explique qué hace y cómo funciona. (La explicación no debe ser una simple "lectura" del código del estilo: "si *s es espacio, del se pone a uno", sino una explicación funcional redactada en lenguaje convencional)

```
char *pepe(char *s) {
int del=0;
char *t, *tt;
tt=t=malloc(sizeof s);

while (*s!='\0' && *s==' ') s++;

while (*s!='\0') {
   if (*s==' ') {del=1; s++; continue;}
   if (del==1) {del=0; *t++=' ';}
   *t++=*s++;
   }
t='\0';
return tt;
}
```

Estructura de un microprocesador. 1,25 puntos

4 Explique paso a paso la secuencia de movimientos de datos a través de los registros y circuitos internos de nuestro procesador virtual cuando la memoria esta inicialmente como se nmuestra en el recuadro.

```
0: 6405
1: 8004
2: 2806
3: 2C00
4: 0005
5: 0009
6: 0000
```

Mapeo de memoria. 1,5 puntos

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

a) 8 Mbytes de RAM situados a partir de 0 mediante 4 pastillas estructuradas en niblas.
b) 1 MByte de ROM en el final del espacio direccionable mediante 2 pastillas en bytes .

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. (muestre el mapa resultante)

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

6 Supongamos que deseamos crear una serie de subrutinas que sirvan para gestionar dietas alimentarias. Cada alimento se deberá codificar como un entero de 8 bits, y una dieta no será sino una secuencia de enteros. Las secuencias de alimentos deberán finalizar siempre con el entero 0, que queda reservado, y por tanto, no puede ser utilizado como código de alimento alguno. Se pide que se diseñen las siguientes subrutinas y programa principal:

Subrutina CONTIENE

Esta subrutina tomará la dirección de una dieta y el código de un alimento, y comprobará si dicho alimento está contenido en la dieta. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:      CONTIENE
; Descripción:    Comprueba si una dieta contiene un alimento.
; Entrada:        A0.L  dirección de la dieta
;                 D0.B  código del alimento
; Salida:         D1.B ( 1 si lo contiene, 0 si no)
; Modifica:       …..
```

Subrutina IGUALES

Esta subrutina tomará la dirección de dos dietas y mirará si son iguales. Téngase en cuenta que los alimentos pueden aparecer en distinto orden en ambas dietas. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:      IGUALES
; Descripción:    Comprueba si dos dietas son iguales.
; Entrada:        A0.L  dirección de una dieta
;                 A1.L  dirección de otra dieta
; Salida:         D0.B  ( 1 si son iguales, 0 si no)
; Modifica:       …..
```

Subrutina UNION

Esta subrutina tomará la dirección de dos dietas y obtendrá una tercera dieta que será la unión de las dos. Un alimento no podrá aparecer repetido en la nueva dieta. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:      UNION
; Descripción:    Realiza la union de dos dietas, obteniendo una nueva dieta que contiene una única
;                 vez todos los alimentos de ambas dietas
; Entrada:        A0.L  dirección de una dieta
;                 A1.L  dirección de otra dieta
;                 A2.L  dirección donde dejar la dieta resultante
; Modifica:       …..
```

Programa principal

El programa principal deberá tomar las siguientes dos dietas: d1={PAN,LECHE,LACTEO,CAFE,ACEITE} y d2={ARROZ,ACEITE,CAFE,SAL} , mirar si son iguales y obtener la unión de ellas dos.

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
13 de septiembre de 2006

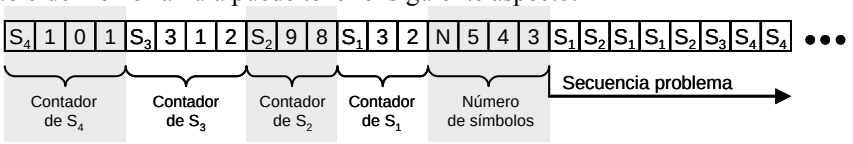
SOLUCIONES Y COMENTARIOS

1 Esta cuestión apareció también en el examen de septiembre del curso 2002-2003, de modo que esta es la misma contestación a la pregunta:

- 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)

2 La respuesta es inmediata: si. La máquina de Turing es un mecanismo capaz de resolver cualquier problema computable, y el aprendizaje de un modelo de Markov lo es. El planteamiento de cómo hacerlo también es algo sencillo de hacer, aunque la implementación sea demasiado compleja como para abordarla manualmente.

El modelo de Markov no es otra cosa que una lista valores de probabilidad asociados a símbolos (o cadenas de símbolos dependiendo del orden del modelo) y una función que determina transiciones de estado cada vez que se procesa un símbolo. Pero esta función no es necesaria cuando los estados están relacionados directamente con los símbolos vistos como es el caso que nos ocupa, y la lista de frecuencias es suficiente. De este modo el resultado de la ejecución para un modelo de memoria nula puede tener el siguiente aspecto:



Donde, por simplificar, se usa notación decimal y no se anotan las frecuencias sino una cuenta de las ocasiones en que aparece cada símbolo y la cuenta total (incluso sobraría la cuenta total para que una MT pudiera usar el modelo como generador de secuencias).

Para obtener esta información de la secuencia original, una máquina de Turing iría "mirando" cada símbolo dejando una marca para volver a por el siguiente y desplazándose a la izquierda para localizar su contador. Una vez localizado lo incrementaría en 1 cuidando de si es preciso utilizar una casilla más, caso en que procedería a desplazar toda la ristra a la izquierda antes de poner un 1 en la casilla liberada. También hay que tener en cuenta la situación en que aparezca un símbolo por primera vez, que sería preciso situarlo en el extremo izquierdo con su contador a 1.

El proceder para un modelo de orden mayor será semejante con la dificultad añadida (y nada desdeñable) de que la tabla de contadores ha de ser no para símbolos sueltos sino cadenas de una longitud dada.

3 Cabe decir antes de nada que el nombre de la rutina "pepe" es desafortunado ya que un nombre de rutina debe indicar lo mejor posible cual es su función. Evidentemente en este caso se ha puesto "pepe" intencionadamente dado que la pregunta consiste en interpretar el código para descubrir su función.

La función recibe como parámetro y devuelve como valor de retorno apuntadores a caracteres, lo que permite suponer (no asegurar por el momento) que se trata de algún tipo de acción modificadora de una cadena de caracteres.

En cuanto a variables locales vemos que disponemos de un entero (`del`) y de dos apuntadores a caracteres (`t` y `tt`) que se inician por igual apuntando a una zona de memoria que resulta de una reserva mediante la función `malloc` de la que hablaremos al final.

- El primer ciclo `while` va deslizando el apuntador `s` a la cadena de caracteres de entrada mientras los apuntados sean espacios en blanco con el cuidado necesario de que no se sobrepase el final de la cadena '\0' (ahora sí podemos asegurar que se trata de cadenas de caracteres). Es decir, en caso de que la cadena apuntada por s comience por un indeterminado número de caracteres en blanco, este ciclo los deja de lado y sitúa el apuntador en el primer carácter no blanco.
- A continuación otro ciclo `while`, que se mantiene mientras no se llegue al fin de la cadena, hace lo siguiente:
  - Si el carácter apuntado es blanco se avanza y se "marca" poniendo a 1 la variable "del". En este caso no se hace nada más dentro del ciclo dado que se encuentra la secuencia "continue".
  - En caso contrario si la marca "del" esta puesta (a 1) se elimina (`del`=0) y se escribe un blanco sobre un espacio apuntado por `t` y avanzando. El efecto combinado con lo anterior es que cualquier secuencia de uno o más caracteres blancos es sustituida por uno solo en una string que se forma mediante el apuntador `t`.
  - Se copia el carácter apuntado por `s` en el espacio apuntado por `t` avanzando ambos apuntadores. A este punto llegamos siempre que no se haya encontrado un espacio en blanco (no ha actuado la primera línea dentro del `while`), de modo que todo carácter no blanco es copiado.
- Por último se "termina" la cadena que se ha ido construyendo mediante `t` con un `'\0'`, retornándola mediante `tt`, que se ha conservado inamovible al principio de esta.

Con todo tenemos que el efecto pretendido de la rutina es proporcionar una cadena de caracteres que se basa en otra y que difiere de ella en que no aparecen caracteres blancos ni al principio ni al final y en la que internamente cualquier cadena de ellos se ha sustituido por sólo uno. (ejemplo: " `Hola  mundo` " → "`Hola mundo`")

Ahora debemos volver al `malloc`. Evidentemente, vista la función, el `malloc` debe reservar el espacio suficiente para recibir la cadena de salida, que será todo lo más el mismo que la de entrada. Es decir el parámetro debe ser la longitud de s. Aquí se ha deslizado un error… si ponemos "`sizeof s`" obtenemos el tamaño del apuntador, mientras que lo que necesitamos es el tamaño de la cadena, cosa que puede conocerse mediante la función de librería `strlen()`. Deberíamos por tanto poner `tt=t=malloc(strlen(s))`.

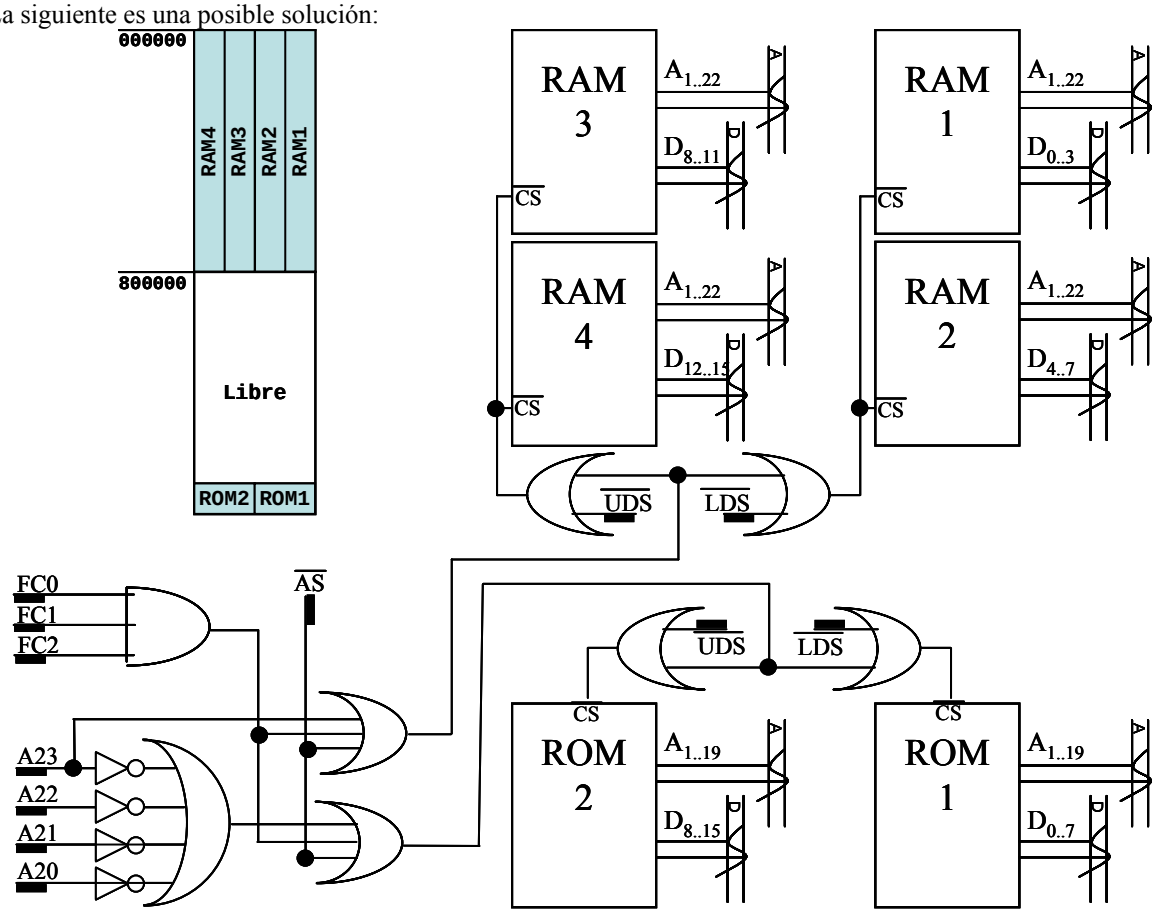
En definitiva tenemos que la rutina no es correcta. Su función es la indicada anteriormente pero debe ser corregida puesto que su inclusión en un programa puede llevar a problemas de sobreescritura de otras variables a través del apuntador `t`.

4 Operación realizada: 5+9=E

```
0: 6405 LOD-C  5   Carga el valor 5 en el acumulador
1: 8004 ADD-I  4   Suma lo apuntado por 4, osea 9 (5+9=14)
2: 2806 STO    6   Almacena en 6 el resultado (000E)
3: 2C00 HLT
4: 0005
5: 0009
6: 0000
```

	Cnt.	Operación
LOD-C 5	1	PC → ADDR {0}
	2	memoria(ADDR{0}) → IR {6405}
	3	PC+1 → PC {1}
	4	IR→ AC {0005}
	5	reset count
ADD-I 4	1	PC → ADDR {1}
	2	memoria(ADDR{1}) → IR {8004}
	3	PC+1 → PC {2}
	4	IR→ ADDR {0004}
	5	memoria(ADDR{4}) → Y {0005}
	6	Y→ ADDR {0005}
	7	AC→X{005}; memoria(ADDR{5}) → Y {0009}
	8	ALU(ADD,X{0005},Y{0009})→AC{000E}
	9	ALU(ADD)
	10	reset count
STO 6	1	PC → ADDR {2}
	2	memoria(ADDR{2}) → IR {2806}
	3	PC+1 → PC {3}
	4	IR→ ADDR {0006}
	5	AC→memoria(ADDR{6}) {000E}
	6	reset count
HLT	1	PC → ADDR {3}
	2	memoria(ADDR{3}) → IR {2C00}
	3	PC+1 → PC {1}
	4	Parar el reloj

5 La siguiente es una posible solución:



6

```
org    $1000

; PROGRAMA PRINCIPAL
lea      DIETA1,a0
lea      DIETA2,a1
bsr.s    IGUALES
move.b   d0,RESULT
lea      DIETA3,a2
bsr.s    UNION
trap     #15
dc.w     0

; Subrutina:    CONTIENE
; Descripción:  Comprueba si una dieta contiene un alimento
; Entrada:      A0.L  dirección de la dieta
;               D0.B  alimento
; Salida:       D1.B  (1 si contiene, 0 si no contiene)
; Modifica:     CCR

CONTIENE
move.l   a0,-(a7)
move.b   #1,d1
bra.s    Awhile
Ado      cmp.b   (a0)+,d0
beq.s    Afinal
Awhile   tst.b   (a0)
bne.s    Ado
clr.b    d1
Afinal   movem.l (a7)+,a0
rts
```

```
; Subrutina:    IGUAL
; Descripción:  Comprueba si dos dietas son iguales
; Entrada:      A0.L  dirección de una dieta
;               A1.L  dirección de otra dieta
; Salida:       D0.B  (1 si son iguales 0 si no lo son
; Modifica:     CCR

IGUALES
        movem.l a0/a1,-(a7)
        bra.s   Bwhile   ; si la primera cadena
Bdo     bsr.s   contiene ; no contiene todos los
        tst.b   d1       ; alimentos de la segunda,
        beq.s   Bmal     ; entonces son diferentes
Bwhile  move.b  (a1)+,d0
        bne.s   Bdo
        move.l  4(a7),a1 ; recuperamos A1 (no lo sacamos de la pila)
        bra.s   Bwhile2  ; si ambas dietas
Bdo2    tst.b   (a0)+    ; no tienen la misma
        beq.s   Bmal     ; longitud,
Bwhile2 tst.b   (a1)+    ; entonces son diferentes
        bne.s   Bdo2
        tst.b   (a0)
        bne.s   Bmal
Bbien   move.b  #1,d0
        bra.s   Bfin
Bmal    clr.b   d0
Bfin    movem.l (a7)+,a0/a1
        rts


; Subrutina:    UNION
; Descripción:  Realiza la union de dos dietas, obteniendo
;               una nueva dieta que contiene una unica vez
;               todos los alimentos de ambas dietas
; Entrada:      A0.L  dirección de una dieta
;               A1.L  dirección de otra dieta
;               A2.L  dirección donde dejar la dieta resultante
; Modifica:     CCR

UNION   movem.l a0/a1/a2/d0/d1,-(a7)
        bra.s   Cwhile1         ; copiamos la primera
Cdo1    move.b  (a0)+,(a2)+     ;  cadena completa
Cwhile1 tst.b   (a0)
        bne.s   Cdo1
        movea.l 8(a7),a0 ; recuperamos A0 (no lo sacamos de la pila)
        bra.s   Cwhile2  ; añadimos solo los
Cdo2    bsr.s   CONTIENE ; elementos de la segunda
Cif     tst.b   d1       ; que no esten en la
        bne.s   Cwhile2  ; primera
Cthen   move.b  d0,(a2)+
Cwhile2 move.b  (a1)+,d0
        bne.s   Cdo2
        clr.b   (a2)
        movem.l (a7)+,a0/a1/a2/d0/d1
        rts
```

```
org     $2000

PAN     equ     1
LECHE   equ     2
AZUCAR  equ     3
CAFE    equ     4
ARROZ   equ     5
ACEITE  equ     6
LACTEO  equ     7
SAL     equ     8

DIETA1  dc.b    PAN,LECHE,LACTEO,CAFE,ACEITE,0
DIETA2  dc.b    ARROZ,ACEITE,CAFE,SAL,0
DIETA3  ds.b    10
RESULT ds.b     1
```