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
20 de enero de 2003

Cuestiones teórico-prácticas. — 1 punto cada una

(Conteste a 3 de las 4 preguntas siguientes. Aquellos alumnos que han presentado un trabajo pueden contestar sólo a 2 de las 4 preguntas y deben indicar junto a su nombre el titulo del trabajo presentado.)

1 ¿Es posible que alguna de las siguientes codificaciones corresponda al resultado de la aplicación del método de Huffman?. Por cada una que descarte explique el motivo.

Símbolo	Probabilidad	A	B	C	D	E	F
s1	1/2	000	0	0	0	0	0
s2	1/4	001	01	10	10	10	100
s3	1/16	010	011	110	110	1100	101
s4	1/16	011	0111	1110	1110	1101	110
s5	1/16	100	01111	11110	1011	1110	111
s6	1/16	101	011111	111110	1101	1111	001

2 Razone cual de las siguientes afirmaciones es cierta:

1. El concepto de Maquina de Turing incluye al de Máquina Universal de Turing.
2. El concepto de Maquina Universal de Turing incluye al de Máquina de Turing
3. El concepto de Maquina Universal de Turing incluye a ciertas Máquinas de Turing (no todas).

3 Suponga que en el computador virtual visto en los temas 6 y 7 de la asignatura, el acumulador no tiene la capacidad de incremento y decremento por si mismo. Explique (no diseñe un circuito) de qué otro modo podría introducir en la máquina la capacidad para seguir disponiendo de las operaciones incremento y decremento en el modelo de programación (lo más detalladamente posible).

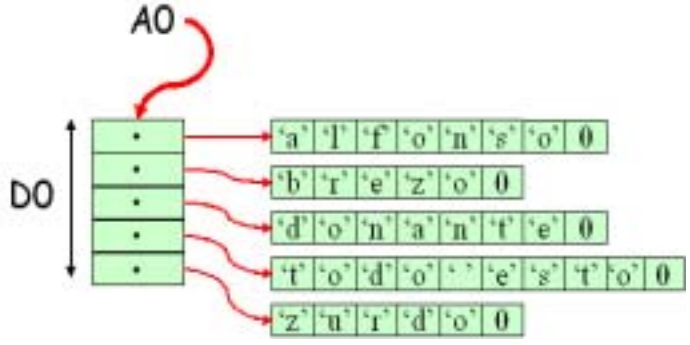
4 Explique para qué sirve, en un procesador, la capacidad de procesamiento de excepciones.

Mapeo de memoria. — 2 puntos

5 Para un sistema con microprocesador 68000 sitúe memoria RAM cubriendo toda la segunda mitad de su espacio direccionable mediante pastillas de 2Mbytes estructuradas en bytes y ponga 64Kbytes de ROM al principio del espacio direccionable utilizando las pastillas que prefiera (sin copias).

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

6 Supongamos que contamos con un vector de direcciones, y que cada dirección de 32 bits hace referencia a una cadena de caracteres. Trataremos de comprobar si dichas cadenas están alfabéticamente ordenadas de menor a mayor:



Para ello, diseñaremos las siguientes subrutinas y programa principal:

Subrutina CMPCAD

Esta subrutina deberá realizar la comparación alfabética de dos cadenas de caracteres y devolver 1, 0 ó -1, dependiendo de si la primera de ellas es mayor, igual o menor que la segunda. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:      CMPCAD
; Descripción:    comparación de 2 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:       (rellenar debidamente)
```

Subrutina COMPRUEBA

Esta subrutina deberá tomar un vector de direcciones que apuntan a cadenas de caracteres y comprobar la correcta ordenación alfabética de ellas, de menor a mayor. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:      COMPRUEBA
; Descripción:    comprueba que las cadenas están en orden alfabético ascendente
; Entrada:        A0.L  inicio del vector de direcciones
;                 D0.W  tamaño del vector
; Salida:         D0.B:  1 ⇔ correcto (orden ascendente)
;                        0 ⇔ incorrecto (orden no ascendente)
; Modifica:       (rellenar debidamente)
```

Programa principal

El programa principal deberá crear una serie de cadenas de caracteres (al menos 4) y un vector de direcciones que contenga las direcciones de las mismas, así como comprobar si las cadenas están en orden alfabético ascendente. El programa principal ***podría*** tener en concreto las siguientes partes:

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

org     $1000
; Inicialización de los parámetros de entrada antes de la llamada a subrutina
;Llamada a COMPRUEBA
;Final

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

Examen de S.E.T.I.

1er curso de Ingeniería Electrónica
20 de enero de 2003

SOLUCIONES Y COMENTARIOS

1 Se trata de determinar la posibilidad de que los códigos presentados se correspondan con el resultado de aplicar el método de Huffman sin llegar a ejecutar dicho método. Por tanto la respuesta se basará en consideraciones que puedan hacerse de modo sencillo y "a ojo" conociendo el tipo de resultado que se obtendría.

El código A no presenta distintas longitudes para símbolos con distintas frecuencias de aparición, cuando evidentemente sería posible ya que sin perdida de ninguna característica podrían sustituirse los códigos 100 y 101 por 10 y 11 bajando inmediatamente la longitud media, y por tanto no puede ser resultado del método mencionado. (Una reasignación para que estos dos códigos se correspondiesen con s1 y s2 bajaría aún más la longitud media)

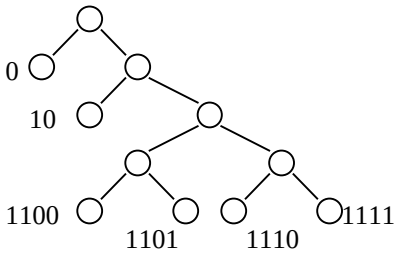
El código B no es inmediato (unos códigos son prefijos de otros), lo que no sucede con el resultado de Huffman, por lo que queda inmediatamente descartado.

El código C también debe descartarse por una característica que podríamos haber considerado también con el B: según el método de Huffman los dos símbolos con mínima frecuencia comparten un prefijo común y se diferencian en el último binit, con lo que al menos dos de los cuatro últimos símbolos deberían tener códigos de igual longitud.

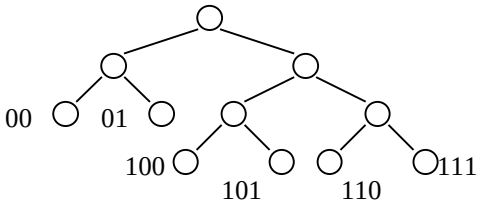
Los códigos D y F son no unívocos, por lo que evidentemente no pueden ser resultado del método. (p.ej: -D 10/110 ó 101/10- -F 0/0/100 ó 001/0/0-)

Nos queda por considerar el código E. En principio parece razonable pensar que puede ser resultado de Huffman ya que asocia los códigos más largos a los símbolos menos probables (lo hace incluso de modo homogéneo aunque esta no es una consecuencia necesaria de la aplicación de Huffman al depender del número de símbolos con idéntica probabilidad y el número de configuraciones binarias disponibles con una longitud dada). En cuanto a los símbolos más cortos también parecen razonablemente asociados a códigos.

Lo anterior puede ser suficiente como respuesta a la pregunta, de todos modos consideremos algo más a fondo la posibilidad de que E sea ciertamente una codificación de Huffman: si planteamos el árbol de códigos y buscamos una distribución que presente una longitud aparente más corta podemos hacerlo "sacrificando" 1 binit a incluir en el código de s0 a cambio de eliminar 1 binit de cada uno de los cuatro códigos más largos (véase la figura). Aunque esta situación puede ser la variación más razonable vemos inmediatamente que es perjudicial ya que en 4 de cada 16 situaciones (1/4) "ahorraremos" 1 binit mientras que en una de cada 2 lo "gastaremos" de más. Este razonamiento, aunque no sea del todo riguroso, nos permite decir que el código E debe de ser un código de Huffman (puede comprobarse por construcción que ciertamente lo es).



Estructura del código E



Estructura de la variación más próxima

2 El concepto de Máquina de Turing se representa o asocia a un mecanismo de evolución en el tiempo basado en una serie de elementos: dos estructuras de datos (la cinta infinita y la tabla de reglas), una variable de estado, un alfabeto de símbolos y un conjunto de estados. En el caso de la Maquina Universal de Turing, todo lo anterior es cierto pero han de añadirse restricciones al contenido de las estructuras: la codificación de una parte de la cinta debe corresponderse con una representación de una

Máquina de Turing y su propia tabla de reglas debe tener de un determinado contenido que permite basarse en la mencionada representación para emular a dicha máquina. Tenemos por tanto que es un concepto más restrictivo y la respuesta correcta será la A.

Visto de otro modo: si asociamos el concepto de Máquina de Turing con el conjunto de todas las posibles máquinas de Turing formulables y, del mismo modo, el concepto de Máquina Universal de Turing con el conjunto de todas las Máquinas Universales de Turing formulables, claramente este último es un subconjunto del primero, lo que nos lleva de nuevo a considerar la A como la respuesta correcta. (*independientemente de que los propuestos "conjuntos de máquinas formulables" sean finitos o no, lo que es otra cuestión en la que no entramos).

Comentario: *se repite en numerosas respuestas la frase "la MUT es una generalización de la MT". Esto no es cierto, sino lo contrario: "las MUT son MT particulares: las que resuelven un determinado problema como es la emulación de una MT". El hecho de que solucionen el problema de la "emulación de una MT cualquiera" no hace que sean una generalización.*

3 En primer lugar se nos puede ocurrir incluir unos circuitos específicos para realizar el incremento y decremento conectados directamente al Acumulador en su entrada y salida para realimentar este en función de una señal, pero esto no es otra cosa que incluir como algo externo al Acumulador lo que tiene nuestra máquina virtual considerado como parte de su estructura (del Acumulador) y sobre lo cual el enunciado plantea su eliminación. Por tanto no es una solución al problema.

La solución habría de pasar necesariamente por basarnos en la ALU para llevar a cabo esta tarea de incremento y decremento. En principio tendríamos dos opciones: o bien nos valemos de los circuitos de suma y resta aportando de algún modo un valor unidad al registro Y a la vez que llevamos el Acumulador al X, o bien construimos dentro de la ALU dos circuitos específicos de incremento y decremento que reciban entrada del registro X cargado a partir del Acumulador.

La segunda opción implica mayor complejidad para la ALU mientras que la primera no requiere más que el mecanismo para aportar el valor 1 al registro Y, y ello sin suponer una desventaja en cuanto a tiempos de ejecución, porque esta carga de Y puede ser paralela a la de X.

Nos quedamos por tanto con la opción de carga de un 1 en Y para la utilización del sumador y el restador, lo que nos lleva a necesitar una señal que realice esta carga y a alterar la función combinatoria que da las señales de control para que los pasos correspondientes a la ejecución de estas instrucciones sean los siguientes (INS será ADD o SUB):

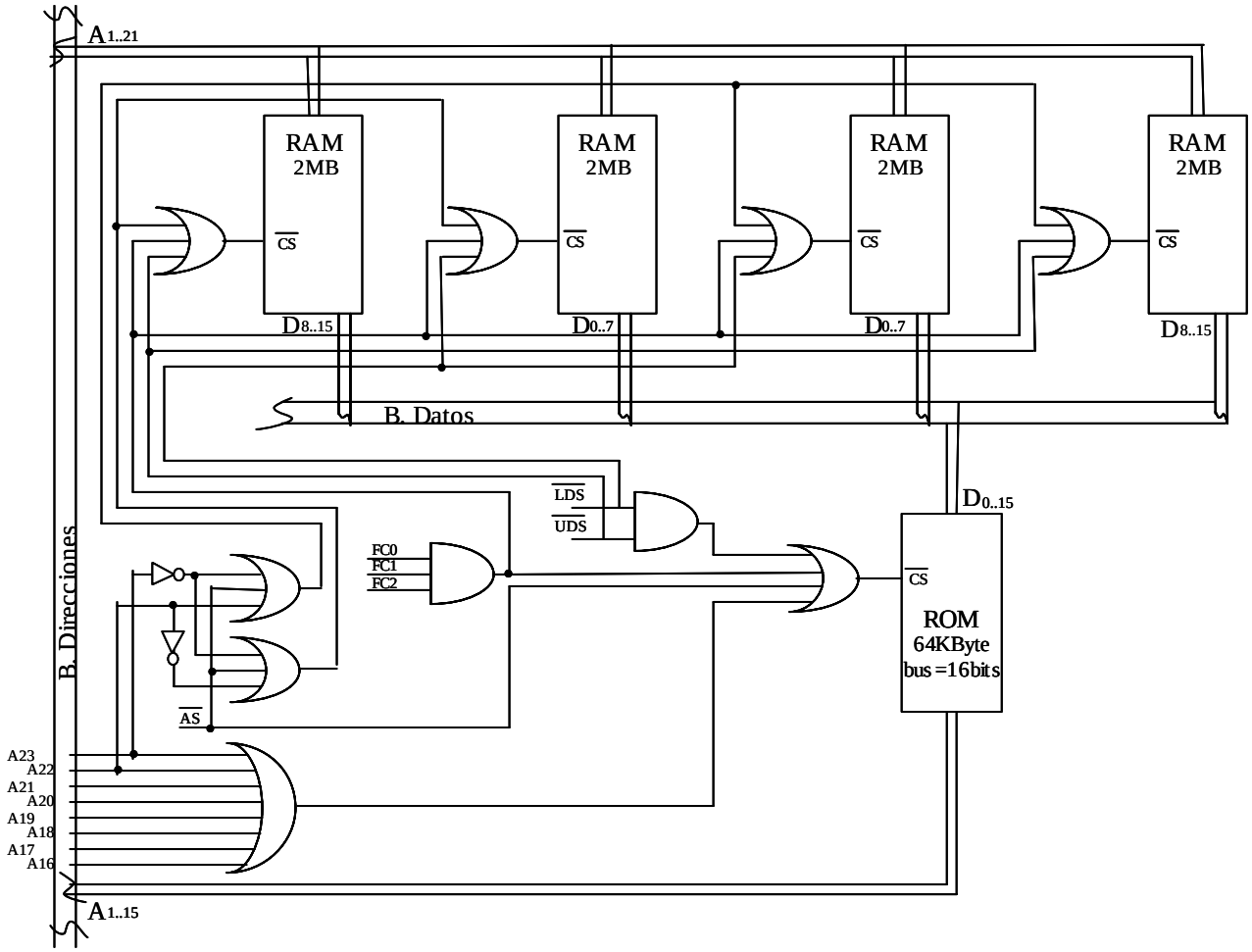
4. Cargar X desde el AC y Cargar Y con 1
5. Seleccionar INS, Cargar AC desde la ALU
6. Seleccionar INS.
7. Poner COUNT a cero

PERO... fijémonos que estos pasos no son otra cosa que los de las instrucciones ADD-C o SUB-C eliminando la salida al Flag y poniendo un 1 donde aquellas tienen IR. Quiere esto decir que salvando la cuestión de la actualización del Flag estamos generando una instrucción que no se diferencia en nada de las "ADD-C 1" y "SUB-C 1", por lo que cabria decir que no tiene mucho sentido plantear ninguna alteración de la máquina por el solo hecho de disponer de una suma y resta de una unidad sin alteración del Flag.

Como conclusión podemos decir que no merece la pena plantear cambios a la máquina en caso de no disponer de incremento y decremento directamente sobre el Acumulador, puesto que la salida lógica es utilizar la suma y resta con la unidad (evidentemente es menos eficaz que INC y DEC).

4 La idea básica consiste en 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 siguiente esquema muestra la solución al problema planteado. Se incluye la señal combinada de FC0,FC1 y FC2 que será necesaria en un caso real para excluir los ciclos de lectura de vectores de interrupción, aunque no se exige como parte de la solución al problema al pedirse sólo la combinatoria necesaria para establecer el mapa de memoria.



6

```
; Subrutina:    CMPCAD
; Descripción:  comparación de 2 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:     CCR

CMPCAD  MOVEM.L  A0/A1 , -(A7)

        BRA.S    WHILE
DO      CMPM.B   (A1)+ , (A0)+
        BHI.S    MAYOR
        BLO.S    MENOR
WHILE   TST.B    (A0)
        BNE.S    DO

        TST.B    (A1)
        BNE.S    MENOR
IGUAL   CLR.B    D0
        BRA.S    FIN
MAYOR   MOVEQ    #1 , D0
        BRA.S    FIN
MENOR   MOVEQ    #-1 , D0
FIN     MOVEM.L  (A7)+ , A0/A1
        RTS
```

```
; Subrutina:    COMPRU
; Descripción:  comprueba que las cadenas
;               están en orden alfabético ascendente
; Entrada:      A0.L  inicio del vector de direcciones
;               D0.W  tamaño del vector
; Salida:       D0.B:  1 ⇔ correcto (orden ascendente)
;                      0 ⇔ incorrecto
; Modifica:     CCR

COMPRU  MOVEM.L  D1/A0/A1/A2 , -(A7)
        MOVEA.L  A0 , A2
        MOVE.W   D0 , D1
        SUBQ.W   #1 , D1

        BRA.S    FOR
DO      MOVEA.L  (A2)+ , A1
        MOVEA.L  (A2) , A0
        BSR.S    CMPCAD
        BLT.S    MAL
FOR     DBRA     D1 , DO

        MOVEQ    #1 , D0
        BRA.S    FIN
MAL     CLR.B    D0
FIN     MOVEM.L  (A7)+ , D1/A0/A1/A2
        RTS
```

```
; PROGRAMA PRINCIPAL
; → Toma la tabla creada y deja el resultado en D0

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

        ORG      $1000

INICIO  LEA      TABLA , A0
        MOVE.W   LNG , D0
        BSR.S    COMPRU
FIN     SERMON   FPROG

        ORG      $2000

CAD1    DC.B     "La primera cadena"
CAD2    DC.B     "La segunda cadena"
CAD3    DC.B     "La tercera cadena"
CAD4    DC.B     "Con esta terminamos"

LNG     DC.W     4
TABLA   DC.L     CAD1
        DC.L     CAD2
        DC.L     CAD3
        DC.L     CAD4
```