Examen de "S.E.T.I."

1er curso de Ingeniería Electrónica
21 de septiembre de 2000

Cuestiones teóricas. *(1 punto cada una)*

1 El "Problema de la parada" demuestra:

A- La imposibilidad de construir una MT para resolver ciertos problemas
B- La existencia de problemas sin solución.
C- La incapacidad de la MUT para resolver ciertos problemas
D- Que a toda Máquina de Turing puede asociársele otra que actúe al contrario.

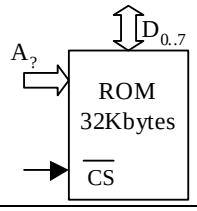
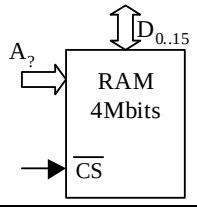
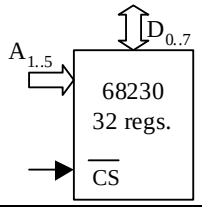
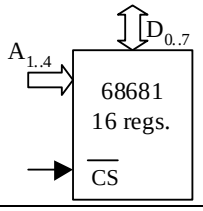
Razone si es cierto o falso cada uno de los puntos anteriores.

2 Ventajas de cada una de las representaciones numéricas para enteros estudiadas: sin signo (base 2), con signo en complemento a dos y con signo en complemento a uno.

3 Uno de los aspectos que teóricamente se contempla dentro de la arquitectura RISC consiste en primar las operaciones con operandos internos (registros del procesador) y limitar las capacidades de acceso externo (fuera del procesador, es decir a direcciones de memoria). Al proporcionar un elevado número de registros internos se considera que esta característica supone una ventaja importante sobre la aproximación CISC (con más capacidad de direccionamiento y menos registros). Explique porqué este enfoque permite confeccionar programas más rápidos (quizás un ejemplo sea aclaratorio).
(Se le ruega no explique otras características de la arquitectura RISC)

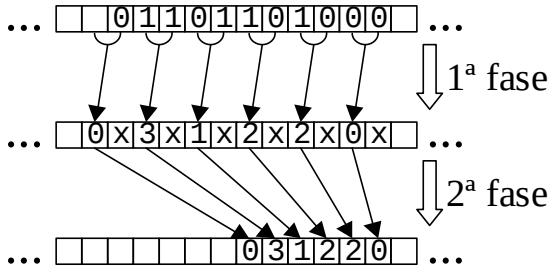
Cuestiones prácticas. *(3+1+4 puntos)*[1]

4 Diseñe el circuito necesario para establecer el mapa de memoria de una computadora con 68000 según los requerimientos siguientes:

<table>
<tr><td>A partir de 000000_{Hex}</td><td></td><td>A partir de B00000 _{Hex}</td><td></td></tr>
<tr><td> 
</td><td>128 Kbytes de memoria ROM utilizando circuitos de 32Kbytes (estructuradas en bytes 8 líneas de datos)</td><td> 
</td><td>1 Mbyte de memoria RAM utilizando circuitos de 4Mbits (estructuradas en words16 líneas de datos)</td></tr>
<tr><td>A partir de 080000 _{Hex} sin que haya copias</td><td></td><td>A continuación de la 68230 y sin copias.</td><td></td></tr>
<tr><td> 
</td><td>Un circuito 68230</td><td> 
</td><td>Un circuito 68681</td></tr>
</table>

[1] La puntuación de las preguntas hace que se pueda llegar a obtener incluso un 11. Toda calificación que pase de 10 se dejará en 10, y las demás no serán alteradas.

5 Queremos diseñar una Maquina de Turing que convierte números de cualquier longitud en base 2 a sus equivalentes en base 4. Para ello planteamos dos fases: en la primera cada par de dígitos se sustituyen por el equivalente y una "x" (véase la figura); en la segunda se compacta eliminando las "x" y juntando los dígitos. Escriba la tabla de la máquina de Turing para la primera fase indicada. -es suficiente un número de reglas del orden de una decena-



6 PROBLEMA DE PROGRAMACIÓN

Las matrices son una de las estructuras de datos muy típica en la mayoría de programas de cierta complejidad. Es por ello, que suele ser comveniente definir el tipo de dato Matriz y definir ciertas operaciones que serán utilizadas en gran número de ocasiones.

Definiremos el tipo de datos “Matriz de Words” como un vector de words. El tamaño de la matriz vendrá dado en las dos primeras words, que contendrán el número de filas y el número de columnas respectivamente. Los valores de la matriz vendrán dados por filas. Por lo tanto, una matriz podrá tener la siguiente representación en memoria:

$$\begin{pmatrix} 0010_H & 2020_H & 104E_H & 00A0_H \\ A47D_H & 0022_H & 1234_H & 0023_H \\ 1110_H & AA00_H & FFFF_H & 0152_H \end{pmatrix} \quad \Rightarrow \quad \Rightarrow$$

3
4
0010
2020
104E
00A0
A47D
0022
1234
0023
1110
AA00
FFFF
0152

Supuesta la definición de “Cadena de Caracteres”, deberán implementarse diversas rutinas de manipulación de matrices y un programa principal que haga uso de ellas, **siguiento en lo posible** el siguiente esquema:

Subrutina MATRIX_OFFSET

Esta subrutina deberá tomar una matriz (su dirección), los índices (i, j) y devolver el número de bytes que dista dicho elemento desde la posición inicial de la matriz (su dirección). Esta subrutina servira para localizar los elementos de la matriz, ya que el elemento (i,j) de una matriz M se deberá encontrar en la posición M+MATRIX_OFFSET(M,i,j). La subrutina pudiera tener en concreto la siguiente descripción:

; Subrutina: MATRIX_OFFSET
; Descripción: Obtiene la distancia en bytes entre la dirección de la matriz y un elemento (i, j)
; Entrada: A0.L dirección de la matriz
; D0.**W** índice i (número de fila)
; D1.**W** índice j (número de columna)
; Salida: D2.**L** offset
; Modifica: .... (especificar)

NOTA: en caso de hacer uso de una multiplicación, **se deberá razonar** la elección de con/sin signo.

Subrutina MATRIX_ADD

Esta subrutina deberá tomar dos matrices, y sumarlas, guardando el resultado en la segunda matriz.. La subrutina pudiera tener en concreto la siguiente descripción:

; Subrutina: MATRIX_ADD
; Descripción: Suma dos matrices.
; Entradas: A0.L dirección de la primera matriz.
; A1.L dirección de la segunda matriz.
; Salida: Ninguna (guarda el resultado en la segunda matriz).
; Modifica: ... (especificar)

NOTA1: La subrutina es resolvible mediante **un único ciclo FOR**
NOTA2: Ojo a los **direcionamientos no permitidos** en la suma.

Subrutina MATRIX_TRANS

Esta subrutina deberá tomar una matriz origen, una destino y transponerla. El resultado deberá guardarse en la matriz destino. La subrutina pudiera tener en concreto la siguiente descripción:

; Subrutina: MATRIX_TRANS
; Descripción: Transpone una matriz
; Entradas: A0.L dirección de la matriz a transponer
; A1.L dirección de la matriz destino
; Salida: Ninguna (modifica la propia matriz).
; Modifica: ... (especificar)

NOTA: Ya que $m[i,j]=m^T[j,i]$,una solución sencilla al problema, haciendo uso de la rutina MATRIX_OFFSET, pudiera tener la siguiente estructura:

```
mT.filas=m.columnas
mT.columnas=m.filas
Para i=[m.filas-1.. 0]
        Para j=[m.columnas-1 .. 0]
                *(mT+matrix_offset(mT,j,i)]=*(m+matrix_offset(m,i,j))
```

Programa principal

El programa principal deberá realizar la siguiente operación de matrices:

$$C = (A+B)^T \text{ , siendo } A = \begin{pmatrix} 0012 & 1024 & AF34 & 1020 & 0023 \\ A346 & 04FF & DDDD & 1234 & 4563 \\ 23FF & FF23 & 0303 & 0001 & 234A \end{pmatrix} B = \begin{pmatrix} 12DD & 1212 & A3E3 \\ FFFF & 1234 & 0034 \\ 4545 & 00EE & ABAB \\ 2323 & 3748 & 9832 \\ 1000 & 2346 & 6523 \end{pmatrix}$$

El programa principal pudiera tener en concreto las siguientes partes:

INCLUDE /usr/68k/semones.i

SECTION 0
Inicialización de los parámetros antes de la llamada a subrutina
Llamada a MATRIX_ADD
Inicialización de los parámetros antes de la llamada a subrutina
Llamada a MATRIX_TRANS
Final

SECTION 1
Declaración de las matrices A y B
Reserva de espacio para la matriz C

SOLUCIONES

MT.

inicial	0	x	izq.	0
inicial	1	x	izq.	1
0	0	0	izq.	inicial
0	1	2	izq.	inicial
1	0	1	izq.	inicial
1	1	3	izq.	inicial
inicial	#	#	dcha.	halt
0	#	0	dcha.	halt
1	#	1	dcha.	halt

Comienza en el "lsb". En el estado inicial se encuentra sobre el bit menos significativo del par a convertir, por lo que lo borra con una "x" y pasa a uno de dos estados en función de lo que había (0 o 1). En cada uno de esos estados comprueba la cinta y tiene la información suficiente para hacer el cambio de base (estado 0 carácter 0 => 00=0 ; estado 0 carácter 1 => 10=2 ; estado 1 carácter 0 => 01=1 ; estado 01 carácter 1 => 11=3). En caso de encontrar espacio en blanco, su valor es cero y actúa igual además de terminar la ejecución. También en el estado inicial termina la ejecución si se encuentra un blanco.