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 2004

Cuestiones teórico-prácticas. 1 punto cada una

1 En un examen de esta misma asignatura (sept 2003) se preguntaba por la demostración del problema de la parada, y en la solución dada posteriormente se decia:

> 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)

Explique esto de la aplicación del problema (k=K) a la máquina (K)... ¿Qué quiere decir exactamente? ¿Por qué es necesario?

2 Supongamos que queremos utilizar nuestro computador virtual visto en la asignatura para llevar a cabo unos procesamientos especificos que necesitan realizar de un modo muy eficaz una operación que denominaremos "espejo". Esta operación cambiará los bits del acumulador de un modo "especular", es decir el que ocupa la posición 0 con el que ocupa la posición 15, 1 con 14, 2 con 13, etc.

Como tenemos el requerimiento de realizar esta operación muy eficazmente no nos sirve llevar a cabo esta operación mediante una secuencia de instrucciones, sino que plantearemos modificar el circuito de nuestro computador para implantar la instrucción "ESP" que lleve a cabo lo indicado.

Tenemos un problema en el hecho de que hay ya 16 instrucciones codificadas con 4 bits, de modo que no disponemos de un codigo libre para nuestra nueva instrucción. Para no complicar demasiado el hardware con ampliaciones de la codificación, lo que haremos será eliminar una instrucción existente en favor de la nueva. La instrucción eliminada será la JMP, de modo que sacrificamos la posibilidad de realizar saltos incondicionales (será preciso poner a cero el acumulador y hacer JMZ) y ahora el código 001100 se corresponderá con la operación ESP y el 101100 no correspondera a una instrucción.

Explique cómo modificar el hardware para conseguir este objetivo.

3 Explique para qué sirve la capacidad de procesamiento de excepciones.

4 El modo de transferencia "natural" del 68000 con los elementos situados en su mapa de memoria es asíncrono. a) Con una sola frase: ¿Qué quiere decir exactamente "asincrono" en este caso? b) También con una sola frase: ¿Cómo "controla" que las transfrerecias se realicen correctamente?, o lo que es lo mismo... ¿Qué informaciones (señales o grupos de señales) intervienen en la transferencia?

Mapeo de memoria. 2 puntos

5 Aprovechando unos viejos circuitos integrados entre los que se encuentra un 68000 queremos construir un computador. Los circuitos disponibles que utilizaremos son:

- Un 68000
- Una RAM de 8Mbits estructurada en bytes
- Dos RAM de 4Mbits estructuradas en niblas (4 bits)
- Una EPROM de 128 KBytes estructurada en words de 16 bits
- Una 68681 (16 registros)

Por lo demás disponemos de toda clase de circuitos con puertas lógicas que podamos precisar.

Diseñe el mapa de memoria y el circuito correspondiente.

Programación del 68000. 4 puntos (0.5+ 2.5 + 1)

6 En tres ciudades españolas (Madrid, Barcelona y Bilbao) están instaladas estaciones metereológicas que recogen las precipitaciones anuales. Dicha información es representada en una estructura de datos que contiene tres elementos:

- Un espacio de 6 caracteres como descriptor de la ciudad.
- Una word donde guardar el año
- Una word para la precipitación anual (en litros por metro cuadrado).

Supondremos que en la memoria está alojado un vector de dichas estructuras. El problema a resolver es la ordenación de dicho vector en función de las precipitaciones, debiendo ordenarse de ***mayor a menor***. Para ello, se hará uso del ***Algoritmo de la Burbuja***.

A pesar de tratarse de uno de los más sencillos algoritmos de ordenación, el algoritmo de la burbuja es también uno de los menos eficientes. Su funcionamiento es bien sencillo: se trata de recorrer el vector elemento a elemento (no de dos en dos) e intercambiarlo con el siguiente en caso de que se encuentren desordenados entre sí. El proceso de recorrer el vector deberá realizarse tantas veces como sea necesario, concretamente, hasta que en un recorrido completo no se haya intercambiado elemento alguno.

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

Subrutina INTERCAMBIA

Esta subrutina deberá tomar la dirección de dos registros meteorológicos e intercambiarlos entre sí. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina: INTERCAMBIA
; Descripción:      Intercambia dos registros meteorológicos
; Entrada:   A1.L  Dirección del primer registro
;            A2.L  Dirección del segundo registro
; Salida:    ninguna
; Modifica:  Ambos registros
```

Subrutina BURBUJA

Esta subrutina deberá tomar la dirección inicial del vector, su longitud y ordenar los elementos (registros) por precipitaciones anuales, de mayor a menor. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina: BURBUJA
; Descripción:      Ordena un vector de registros meteorológicos.
; Entrada:   A0.L  Dirección del vector
;            D0.W  longitud del vector
; Salida:    ninguna
; Modifica:  Contenido del vector
```

Programa principal

El programa principal deberá crear un vector que contenga los siguientes datos (un registro por cada ciudad y año) y ordenarlo:

Ciudad	Precipitaciones (l/m^2)	
	2002	2003
Madrid	354	432
Barcelona	751	740
Bilbao	887	846

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 BURBUJA
;   Final

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

eman ta zabal zazu
universidad euskal herriko
del país vasco unibertsitatea

departamento de electricidad y electrónica
elektrika eta elektronika saila

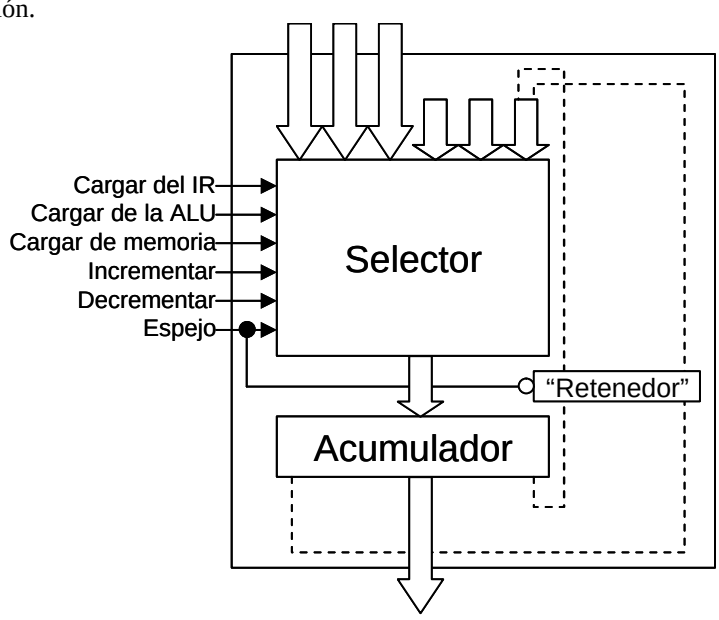
Examen de S.E.T.I.

1er curso de Ingeniería Electrónica
1 de septiembre de 2004

SOLUCIONES Y COMENTARIOS

1 Una vez que construimos la máquina K, para demostrar que no existe H debemos estudiar lo que sucede cuando ejecutamos K con su propia descripción en la cinta, es decir la aplicación a un problema genérico k (K,k) se concreta en la aplicación con su propia descripción (K,K). Una vez pasada la primera fase, en la cinta encontramos dos copias de K, que es lo que H analiza. Es decir, H analiza exactamente el comportamiento de la ejecución que se esta llevando a cabo (K,K). En consecuencia, el diseño de la tercera fase nos asegura que el comportamiento de (K,K) es el contrario del vaticinado por H para (K,K) invalidando la hipótesis sobre la capacidad de H.

2 Podemos plantear varias formas de llevar a cabo esta operación. La más "evidente" consistiría en añadirla a las realizadas por la ALU, pero también puede plantearse como una capacidad específica del Acumulador (al igual que lo es, por ejemplo, el incremento o el decremento). Esta segunda opción será con mucho más eficaz que la primera ya que bastará con un solo ciclo interno para la fase de ejecución. Si planteamos la solución mediante la ALU, será preciso hacer transferencias entre el acumulador y el registro de entrada a la ALU por lo que resulta más costoso. Por tanto planteamos aquí la segunda opción.



El cambio a introducir en el Acumulador consiste simplemente en permitir que se cargue de una entrada más y ser esta la formada por líneas procedentes del mismo Acumulador con la reordenación planteada. En realidad esto conlleva la necesidad de situar un "retenedor" intermedio (circuito "latch", p.ej. basado en "flip-flops" de tipo D), de modo que el acumulador no actúe a la vez de entrada y salida de señales (véase la figura). La nueva línea de carga que hemos denominado "espejo" en la figura, ha de venir dada por el circuito combinacional que gestiona el conjunto de líneas de control. Este circuito habrá de activar la señal "espejo" cuando se entre en la fase de ejecución con el código de instrucción correspondiente a ESP (que antes era JMP), por lo tanto el paso 4 de JMP ya no es tal (Cargar PC desde IR) sino que ahora hablaremos del paso 4 de ESP que será "activar" la señal "espejo" en el Acumulador. Además será preciso eliminar del circuito combinacional todo lo correspondiente a la instrucción JMP-I.

ESP (001100)	~~**JMP (001100)**~~	~~**JMP-I (101100)**~~
4. Activar "Espejo" en Acumulador	~~4. Cargar PC desde el IR~~	~~4. Cargar ADDR desde el IR~~
5. Poner COUNT a cero	~~5. Poner COUNT a cero~~	~~5. Cargar PC desde memoria~~
		~~6. Poner COUNT a cero~~

En caso de optar por incluir la instrucción en la ALU, no es necesario un retenedor ya que tal función esta llevada a cabo por el registro X a su entrada. En este caso la operación se realiza mediante el cableado correspondiente dentro de la ALU (como sucede con SHR y SHL) y la secuencia a generar por el circuito combinacional será:

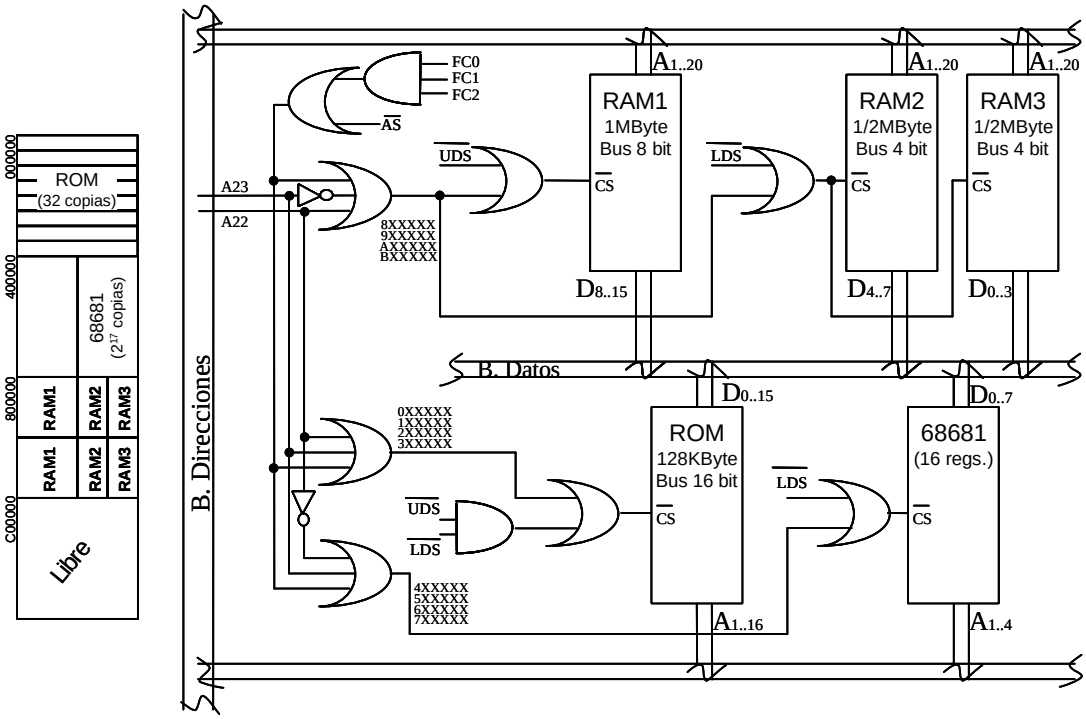
ESP (001100)

4. Cargar X desde el Acumulador
5. Seleccionar "Espejo" y cargar el Acumulador desde la ALU
6. Seleccionar "Espejo"
7. Poner COUNT a cero

3 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**.

4 a) no existe una señal "de sincronización" (el "reloj" en los sistemas sincronos) respecto a la cual actúen de un modo convenido tanto el 68000 como el elemento con quien realiza éste la transferencia.
b) El 68000 dispone las señales para indicar el inicio del ciclo de transferencia y cuando el elemento implicado lleva acabo su función (tomar o dejar un dato) avisa al 68000 con una señal (DTACK) de modo que este puede cerrar el ciclo.

5 Una posible solución es la mostrada en la figura de abajo. Se consideran cuatro segmentos del espacio direccionable mediante las líneas A_{23} y A_{22}. En el primero se sitúa la ROM, que tiene 16 líneas de direcciones y por tanto quedan sin usar 5 (A_{17},..,A_{21}) resultando 32 (2^5) imágenes de la misma. En el segundo se sitúa el periférico que tiene 4 líneas de dirección por lo que quedan libres 17 dando lugar a 2^{17} copias. Por último, la RAM con sus 20 líneas de direcciones en todos los casos (1MB=2^{20} bytes, 1/2MB=2^{20} niblas) ocupa por completo el tercer cuarto del espacio direccionable con dos imágenes (libre la línea A_{21}).



6

```
; Subrutina:      INTERCAMBIA
; Descripción:    Intercambia dos registros meteorológicos
; Entrada:        A1.L  Dirección del primer registro
;                 A2.L  Dirección del segundo registro
; Salida:         ninguna
; Modifica:       Ambos registros

INTERCAMBIA MOVEM.L   D0/A1/A2,-(A7)
            MOVE.L    (A2),D0          ; En total debemos intercambiar
            MOVE.L    (A1),(A2)+       ; 10 bytes = 4 (L) + 4 (L) + 2 (W)
            MOVE.L    D0,(A1)+
            MOVE.L    (A2),D0
            MOVE.L    (A1),(A2)+
            MOVE.L    D0,(A1)+
            MOVE.W    (A2),D0
            MOVE.W    (A1),(A2)
            MOVE.W    D0,(A1)
            MOVEM.L   (A7)+,D0/A1/A2
            RTS


; Subrutina:    BURBUJA
; Descripción:  Ordena un vector de registros meteorológicos.
; Entrada:      A0.L  Dirección del vector
;               D0.W  longitud del vector
; Salida:       ninguna
; Modifica:     Contenido del vector

BURBUJA   MOVEM.L   D0-D3/A1/A2,-(A7)
          SUBQ      #1,D0            ; (D0-1) comparaciones en cada recorrido del vector
DO_W      CLR.B     D1               ; ciclo DO-WHILE, en D1 marcamos los intercambios
          MOVE.W    DO,D2            ;   D2←D0  es el índice del ciclo for
          BRA.S     FOR              ;   ciclo FOR → se recorre el vector
          MOVEA.L   A0,A1            ;       A1←A0 recuperamos el inicio del vector
DO_F      MOVE.W    8(A1),D3         ;       cargamos el año y lo comparamos -
          CMP.W     D3,18(A1)        ;       con el año del elemento adyacente
IF        BLO.S     SIGUE            ;       si el orden es correcto, sigue mirando, sino,
ELSE      MOVEQ     #1,D1            ;           marcar el intercambio
          LEA       10(A1),A2        ;           cargar en A1 dirección del siguiente registro
          BSR.S     INTERCAMBIA      ;           intercambiar los registros
SIGUE     ADDQ.L    #10,A1           ;       pasar al siguiente registro
FOR       DBRA      D2,DO_F          ;   ( D2 iteraciones )
WHILE     TST.B     D1               ; mientras  (D1 ? 0)
          BNE.S     DO_W
          MOVEM.L   +(A7), D0-D3/A1/A2
          RTS

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

```
            org     $1000
            LEA     VECTOR,A0
            MOVE.W  LONG,D0
            BSR.S   BURBUJA
            SERMON  FPROG

            org     $2000
LONG        DC.W    6
VECTOR      DC.B    'Madrid'
            DC.W    2002
            DC.W    354
            DC.B    'Madrid'
            DC.W    2003
            DC.W    432
            DC.B    'Barcel'
            DC.W    2002
            DC.W    751
            DC.B    'Barcel'
            DC.W    2003
            DC.W    740
            DC.B    'Bilbao'
            DC.W    2002
            DC.W    887
            DC.B    'Bilbao'
            DC.W    2003
            DC.W    846
```