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
18 de abril de 2008 ("repesca")

Temas teóricos

1 Imagine que tiene que dar una charla extremadamente corta sobre "El formato IEEE-754" explicándolo someramente y justificando el porqué de sus particularidades. Ponga por escrito lo que diría.

2 Un alumno de SETI se encuentra en una discoteca, y no puede evitar formarse un modelo de la secuencia de tipos de canciones que "pincha" el discjockey. Cuando lleva escuchadas 34 tiene detectado que éste sigue una secuencia cíclica que es la siguiente: HCDD seguido de R o B donde R es el doble de frecuente que B. Si en el momento de escuchar la canción 35 ésta es de tipo B ¿supone esto que recibe alguna información?¿cuánta en tal caso?. ¿Y si esto sucede en la número 31? (que sea B). Suponiendo que la secuencia no se rompe nunca ¿Qué entropía caracteriza a nuestro discjockey?

(por si sirve de contextualización, diremos que H es un Hit de actualidad, C un Clásico, D es música Disco, R un Rock suave, y B una Balada)

Hardware

3 No es infrecuente que la arquitectura interna de un procesador sea tal que sus registros sean el doble de "anchos" que su bus de datos (es el caso del 68000). Suponga que fabricamos una nueva versión de nuestro computador virtual con arquitectura interna de 32 bits manteniendo el bus de datos de 16. ¿Qué implicaciones tendría esto?. De un listado de todo aquello que considere que se ve afectado y esboce soluciones para lo que represente un cierto problema.

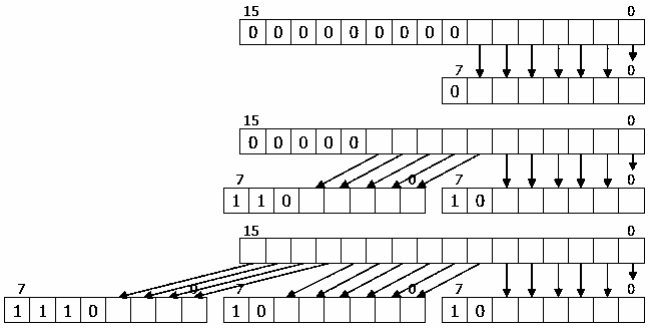
4 En el examen de enero se planteaba la siguiente situación:

> Disponemos de una vieja placa de computador con 68000 con un conector para extensión con todos los buses, y de cuatro circuitos integrados de memoria RAM que vamos a aprovechar para añadirla al sistema mediante una pequeña placa de circuito impreso. Estos circuitos son: 2 RAM de 256Kbytes cada una, estructuradas en niblas, y 2 RAM de 256Kbytes cada una, estructuradas en bytes
>
> El sistema dispone de 1MByte de ROM en las posiciones iniciales, 1MByte de RAM en las finales, y pastillas de periféricos ocupando las zonas 80XXXX y 81XXXX.
>
> Debemos "mapear" nuestras memorias del modo que nos parezca más conveniente.
>
> Dibuje cómo quedará el mapa de memoria con la inclusión que decida realizar y el circuito correspondiente.

Queremos alterar ahora ligeramente esto, de modo que tanto la ROM como los periféricos sólo sean accesibles en modo supervisor, mientras que toda la RAM sea accesible tanto en supervisor como en usuario. Dado el esquema con que se resolvió el problema anterior (solución al examen) y entendiendo cómo ha de ser la circuitería que se encuentra de entrada en la placa para "mapear" los componentes previos, explique y/o dibuje (mejor "y") lo que es preciso hacer.

Programación del 68000.

5 En un sistema basado en 68000 queremos disponer de la capacidad de manejar textos en Unicode (a 16 bits) y en UTF8. Entre otras cosas queremos poder convertir textos entre estos dos formatos en los dos sentidos. La relación entre estos dos formatos se muestra en la figura de la derecha.



Para esto habremos de disponer de dos rutinas que lleven a cabo las conversiones de un caracter "Unicode_to_UTF8" y "UTF8_to_Unicode" que pueden tener los prototipos que se muestran más abajo. Escriba una de las dos rutinas, y la correspondiente para convertir textos (acabados en un cero) basada en la primera.

```
;RUTINA UTF8_to_Unicode
;ENTRADA    A0.L        Direccion donde se encuentra el UTF8 (seran 1, 2, o 3 bytes)
;SALIDA     D0.W        Caracter Unicode
;           A0.L        Direccion del siguiente byte a los leidos

;RUTINA Unicode_to_UTF8
;ENTRADA    D0.W        Caracter Unicode
;           A1.L        Direccion para dejar el UTF8
;SALIDA     A1.L        Direccion del siguiente byte a los escritos
```

eman ta zabal zazu
universidad euskal herriko
del país vasco unibertsitatea

departamento de electricidad y electrónica
elektrika eta elektronika saila

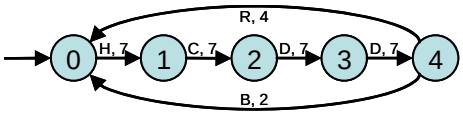
Examen de S.E.T.I.

1er curso de Ingeniería Electrónica
18 de abril de 2008 ("repesca")

SOLUCIONES Y COMENTARIOS

1 *La respuesta a esta pregunta debe explicar la necesidad que mueve a disponer de un formato para números reales de tipo exponencial(es decir, hablar de "rango" y "precisión") así como de una representación tan "complicada" como el IEEE-754, ya que su formulación presenta 4 interpretaciones (es decir, cabe hablar del problema en torno al cero de la formulación genera,l así como de las necesidades de representación de infinitos y de resultados aritméticos fuera de los reales)*

2 Nuestro alumno de SETI se ha formado el modelo de la figura después de escuchar 34 canciones (los números son contadores). En el momento de comenzar a escuchar la que hace el número 35 se encontrará en el estado 4. Si la canción que escucha es de tipo B se corresponde con una observación que ha realizado anteriormente 2 veces de las 6 que se ha encontrado en dicha situación. La probabilidad asignada por su modelo era por tanto 1/3, y en consecuencia la información recibida **$\log_2 3$**.



De haber observado qua la canción número 31 (paso de estado 0 al 1) era de tipo B, la cantidad de información se considera infinita, dado que la probabilidad asignada por el modelo es nula.

La entropía de la fuente (el discjockey) es la del modelo de la figura. Como los estados de 0 a 3 presentan entropía nula y representan a 4 de cada 5 transiciones, lo que nos queda es que la entropía será 1/5 de la del estado 4. Por tanto:

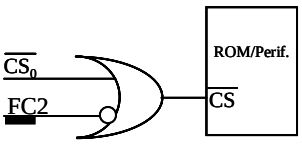
$$1/5\ [\ (2/3)\log_2(3/2) + (1/3)\log_2(3)\ ] = [\ (3\log_2 3) - 2\ ] / 15$$

3 El que la arquitectura interna sea de 32 bits supone que tanto los registros como la ALU pasen de ser de 16bits a tener un ancho de 32 bits, lo que no supone ninguna dificultad de diseño. Si este aumento del "ancho" se trasmite íntegramente al exterior, es decir, buses de direcciones y de datos de 32 bits, tampoco se da ninguna dificultad añadida para la arquitectura interna. La dificultad surge en el caso de que el bus de datos se mantenga en 16 bits como dice el enunciado (en cuanto al bus de direcciones, si no se amplia, simplemente los bits extra serán inservibles).

Para mantener el ancho del bus de datos a 16 bits y manejar datos de 32 bits, será necesario adoptar la estrategia que conocemos del microprocesador 68000: realizar dos ciclos de lectura/escritura consecutivos. En cada ciclo se actuará sobre una mitad de los registros internos, por lo que éstos han de ser seleccionables por mitades, es decir deben tener dos señales internas que los activen: una para la parte alta y otra para la parte baja. Con esto sólo quedaría modificar las secuencias de acciones asociadas a cada instrucción para que donde hay un acceso a memoria haya dos consecutivos y afectaran al registro correspondiente en cada una de las dos mitades sucesivamente.

En principio esto es todo. Otra cosa sería que añadiésemos la capacidad de trabajar con operandos del tamaño del bus (16bits) y del tamaño interno (32 bits) (W,L en el caso del 68000), lo que realmente no añade una dificultad nueva sino que simplemente exige establecer una codificación para especificarlos y ampliar la función combinacional para dar cabida a las dos variantes de ejecución.

4 En la solución dada siempre era posible acceder a todos los circuitos, y lo que ahora se pretende varía tan sólo en que se niegue el acceso en estado usuario para el caso de la ROM y los periféricos. En consecuencia bastará con tomar la señal FC2 para añadirla a la decodificación de las señales de selección de dichos circuitos de modo que sólo se activen cuando FC2=1 (estado supervisor). Bastará con alterar por tanto la señal de selección como se ve en la figura.



5 A continuación se muestran soluciones par alas dos opciones que se planteaban a elegir.

OPCION 1: "UTF8_to_Unicode" y "STR_ UTF8_to_Unicode"

```
;RUTINA UTF8_to_Unicode
;ENTRADA A0.L  Direccion donde se encuentra el UTF8 (seran 1, 2, o 3 bytes)
;SALIDA  D0.W  caracter Unicode
;        A0.L  Direccion del siguiente byte a los leidos
;OBSERVACIONES: No es robusta, porque no tiene en cuenta que no todo codigo vale.

UTF8_to_Unicode:
       MOVE.W    D1,-(SP)
       CLR.W     D0
       MOVE.B    (A0)+,D0        ;Cogemos 1 byte
       BTST      #7,D0           ;y si el bit 7 es 1 hay más bytes
       BNE.S     DosOTres
       ; -------------------------- CASO 1 byte
Fin1:  MOVE.W    (SP)+,D1
       RTS

DosOTres:
       BTST      #5,D0           ;Si el bit 5 es 1
       BNE.S     Son3Bytes       ;son tres bytes

       ; -------------------------- CASO 2 bytes
       AND.B     #%00011111,D0   ;nos quedamos con los 5 bits del primero
       LSL.W     #6,D0           ;los ponemos en su sitio
ByteBajo:
       MOVE.B    (A0)+,D1        ;cogemos el siguiente byte
       AND.B     #%00111111,D1   ;nos quedamos con sus 7 bits
       OR.B      D1,D0           ;y lo mezclamos con lo anterior
       BRA.S     FIN1
Son3Bytes:
       ; --------------------------CASO 3 bytes
       AND.B     #%00001111,D0   ;nos quedamos con los 4 bits del primero
       LSL.W     #8,D0           ;que empujamos arriba
       MOVE.B    (A0)+,D0        ;cogemos el segundo byte
       LSL.B     #2,D0           ;y lo empujamos para juntarlo con lo anterior
       LSL.W     #4,D0           ;y ahora ponemos todo esto en su sitio
       BRA.S     ByteBajo        ;para terminar con el byte bajo como en el caso anterior



;RUTINA STR_UTF8_to_Unicode
;ENTRADA A0.L  Direccion donde se encuentra el UTF8 (acabado con un byte a cero)
;        A1.L  Direccion donde dejar el UNICODE (acabado con una word a cero)
;OBSERVACIONES: No es robusta, en el sentido en que UTF8_to_Unicode no lo es.

STR_UTF8_to_Unicode
       MOVEM.W   D0/A0-A1,-(SP)
CICLO1:
       TST.B     (A0)
       BEQ.S     FINstr1
       BSR.S     UTF8_to_Unicode
       MOVE.W    D0,(A1)+
       BRA.S     CICLO1
FINstr1:
       MOVE.W    #0,(A1)+
       MOVEM.W   (SP)+,D0/A0-A1
       RTS
```

OPCION 2: "Unicode_to_UTF8" y "STR_ Unicode_to_UTF8"

```
;RUTINA Unicode_to_UTF8
;ENTRADA   D0.W        caracter Unicode
;          A1.L        Direccion para dejar el UTF8
;SALIDA    A1.L        Direccion del siguiente byte a los escritos

Unicode_to_UTF8:
       MOVE.W    D1,-(SP)
       ;Comprueba si va a 1 byte o mas
       MOVE.W    D0,D1
       AND.W     #$FF80,D1
       BNE.S     MasDe1
       ; -------------------------- CASO a 1 byte
       MOVE.B    D0,(A1)+        ; El unico byte va tal cual
       BRA.S     FIN2

       ;Comprueba si va a 2 bytes o mas
MasDe1:
       MOVE.W    D0,D1
       AND.W     #$F800,D1
       BNE.S     MasDe2
       ; -------------------------- CASO a 2 bytes
       MOVE.W    D0,D1           ; El byte mas alto
       LSR.W     #6,D1           ; lleva los bits eliminando los 6 mas bajos
       OR.B      #%11000000,D1   ; con esta mascara
       MOVE.B    D1,(A1)+
       BRA.S     ByteBajo2       ; el byte mas bajo es igual al caso de 3 bytes

MasDe2:
       ; -------------------------- CASO a 3 bytes
       MOVE.W    D0,D1           ; El byte mas alto
       ROL.W     #4,D1           ;lleva los 4 bits altos abajo
       OR.B      #%11100000,D1   ;con esta mascara
       MOVE.B    D1,(A1)+        ;
       MOVE.W    D0,D1           ; El byte central
       LSR.W     #6,D1           ;lleva los bits eliminando los 6 mas bajos
       AND.B     #%00111111,D1   ;y los que de arriba ya guardados
       OR.B      #%10000000,D1   ;con esta mascara
       MOVE.B    D1,(A1)+
ByteBajo2:
       MOVE.W    D0,D1           ; El byte mas bajo
       AND.B     #%00111111,D1   ;lleva solo los 6 bits mas bajos
       OR.B      #%10000000,D1   ;con esta mascara
       MOVE.B    D1,(A1)+

FIN2:
       MOVE.W    (SP)+,D1
       RTS


;RUTINA STR_Unicode_to_UTF8
;ENTRADA A0.L  Direccion donde se encuentra el UNICODE (acabado con una word a cero)
;        A1.L  Direccion donde dejar el UTF8 (acabado con un byte a cero)
;OBSERVACIONES: No es robusta, en el sentido en que UTF8_to_Unicode no lo es.

STR_Unicode_to_UTF8:
           MOVEM.W     D0/A0-A1,-(SP)
CICLO2:    MOVE.W       (A0)+,D0
           BEQ.S       FINstr2
           BSR.S       Unicode_to_UTF8
           BRA.S       CICLO2
FINstr2:   MOVE.B      #0,(A1)+
           MOVEM.W     (SP)+,D0/A0-A1
           RTS
```