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
26 de enero de 2006

Cuestiones. 1 + 1 puntos

1 Si escuchamos una obra de música clásica que no hemos oído nunca anteriormente, prácticamente todos podemos suponer –con mayor o menor éxito en función del "entrenamiento de nuestro oído"- quién es su autor. ¿Qué reflexiones puede realizar en relación con esto referentes a lo estudiado sobre la teoría de la información? (Es decir, ¿puede hablarse de símbolos, mensajes, modelos, etc. para explicar el fenómeno comentado?).

2 Cuando Turing envió su trabajo sobre la Máquina que lleva su nombre a una revista científica para su publicación, recibió comentarios bastante negativos de uno de los revisores. En la columna de la derecha puede ver el comienzo de su carta. Comenta que Turing demuestra (con un formalismo "innecesariamente complicado") que hay números que no pueden ser computados por su máquina, y que eso sólo puede significar dos cosas: o el número es demasiado grande o simplemente la máquina esta "rota". ¿A qué cree el alumno que se refiere esa imposibilidad demostrada por Turing de computar ciertos números y qué tendría que decirle al revisor?

> "On Computable Numbers, with an Application to the Entscheidungs-problem." This is a bizarre paper. It begins by defining a computing device absolutely unlike anything I have seen, then proceeds to show—I haven't quite followed the needlessly complicated formalism—that there are numbers that it can't compute. As I see it, there are two alternatives that apply to any machine that will ever be built: Either these numbers are too big to be represented in the machine, in which case the conclusion is obvious, or they are not; in that case, a machine that can't compute them is simply broken!

Ejercicio de C. 1,25 puntos

3 El programa en C adjunto sirve para comprobar el funcionamiento de la rutina "fuente".

Analice esta rutina y explique qué hace y cómo funciona.

```
#include <stdio.h>
#include <stdlib.h>
#define NLETRAS ('Z'-'A'+1)

double rand0_1(){return rand()/(double)RAND_MAX;}

void fuente(int longitud, double *distribucion) {
int n;
srand(time(NULL));
while (longitud)
    if (distribucion[n=(int)(rand0_1()*NLETRAS)] > rand0_1() && longitud--)
        printf("%c",'A'+n);
}

int main(void){
    //una distribución de prueba (no necesita normalización)
    double distribucion[NLETRAS];
    for (int i=0;i<NLETRAS;i++) distribucion[i]= (i+1)/(i*i+2.0)+i/40.0;
    //test
    fuente(1000,distribucion); printf("\n"); return 0;
}
```

Estructura de un microprocesador. 1,25 puntos

4 ¿Qué es el prefetch? ¿Seria útil plantear la capacidad de hacer prefetch en nuestro computador virtual? ¿Qué sería necesario para implantarlo en nuestra máquina (a grandes rasgos, es decir, no es necesario remodelar los circuitos sino plantear las alteraciones necesarias para la arquitectura)?

Mapeo de memoria. 1,5 puntos

5 Diseñe el circuito necesario para que un computador con microprocesador 68000 se ajuste a los siguientes requisitos: (y muestre el mapa resultante)

a) 4 Mbytes de RAM situados a partir de 0 mediante cuatro pastillas estructuradas en niblas (4 binits).
b) 1 Mbyte de ROM mediante 2 pastillas en bytes a partir de 800000_{hex} sólo accesible en estado supervisor.
c) Un periférico con 64 registros sólo accesible en estado supervisor (sin especificación de su posición).
d) Al menos 2 Mbytes libres para futuros componentes al final del espacio direccionable.

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.

Programación del 68000. 4 puntos (1 + 2 + 1)

6 La **Criba de Eratóstenes** es un algoritmo que permite hallar todos los números primos hasta un número natural dado *N*. Se forma una tabla con todos los números naturales comprendidos entre 1 y *N* y se van tachando los números que no son primos de la siguiente manera: comenzando en el 2, cuando se encuentra un entero que no ha sido tachado, ese número es declarado primo, y se procede a tachar todos sus múltiplos. El proceso termina cuando el cuadrado del mayor numero confirmado como primo es mayor que *N*.

Supongamos que tenemos una tabla que comienza con una Word que nos informa del número de casillas que le siguen. Cada casilla es de tamaño Byte, y hace referencia a los números de 1 a N (la primera casilla a 1, la siguiente a 2, etc). Para marcar si un número es primo o no, guardaremos un $FF (número primo) o un 0 (número no primo).

Para determinar los números primos entre 1 y 20 haríamos lo siguiente:

1. Crear la tabla y suponer que todos son primos (inicializar a $FF):

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
20	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**	**FF**

2. El siguiente primo a 1 es 2. Todos sus múltiplos no son primos y por tanto son marcados:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
20	**FF**	**FF**	**FF**	**00**	**FF**	**00**	**FF**	**00**	**FF**	**00**	**FF**	**00**	**FF**	**00**	**FF**	**00**	**FF**	**00**	**FF**	**00**

3. El siguiente primo a 2 es 3. Todos sus múltiplos no son primos y por tanto son marcados:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
20	**FF**	**FF**	**FF**	**00**	**FF**	**00**	**FF**	**00**	**00**	**00**	**FF**	**00**	**FF**	**00**	**00**	**00**	**FF**	**00**	**FF**	**00**

4. El siguiente primo a 3 es 5, pero $5^2 > 20$, así que concluimos. Todos los primos están a $FF

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
20	**FF**	**FF**	**FF**	**00**	**FF**	**00**	**FF**	**00**	**00**	**00**	**FF**	**00**	**FF**	**00**	**00**	**00**	**FF**	**00**	**FF**	**00**

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

Subrutina TACHA

Esta subrutina tomará la dirección de la tabla y un número entero, y tachará (marcará con $00) todos sus múltiplos. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:      TACHA
; Descripción:    Tacha en una tabla los múltiplos de un numero dado.
; Entrada:        A0.L  dirección de la tabla
;                 D0.W  numero
; Modifica:       CCR
```

Subrutina CRIBA

Esta subrutina tomará la dirección de la tabla, y tachará (marcará con $00) todos los números no primos. La subrutina ***podría*** tener en concreto la siguiente descripción:

```
; Subrutina:      CRIBA
; Descripción:    Realiza la Criba de Eratóstenes sobre una tabla.
; Entrada:        A0.L  dirección de la tabla
; Modifica:       CCR
```

Programa principal

El programa principal deberá realizar la criba de Eratóstenes para calcular los números primos del 1 al 1000.

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
26 de enero de 2006

SOLUCIONES Y COMENTARIOS

1 La música es fácilmente entendible como una secuencia discreta de sonidos ya que, aunque sea una señal continua, nuestro oído tiene una resolución temporal máxima (la del muestreo doble a la máxima frecuencia audible). Por otro lado, a diferencia de lo que ocurre con el ruido en general, la música sigue unos determinados patrones que nos resultan más o menos agradables, es decir una determinada muestra de la secuencia no es independiente de las anteriores. Por tanto, para cada valor de una muestra (un símbolo) existe una distribución de probabilidades condicionadas a las muestras anteriores que puede adquirirse y almacenarse a modo de modelo estimado de la fuente que las produce (un modelo Markoviano del orden que sea necesario). Si hacemos esta estimación separadamente para secuencias musicales de diferentes compositores, tendremos un modelo para cada uno de ellos, y dada una secuencia nueva (una obra desconocida) podremos evaluar cuál de entre todas las fuentes estimadas sería la que produciría dicha secuencia con más probabilidad.

Esto es básicamente lo que sucede en el cerebro humano: se asocia a cada autor un determinado modelo. Parece evidente que no será un modelo de toda posible combinatoria de secuencias, sino más bien sólo de aquellas más probables. Además es razonable pensar que en el caso del cerebro humano el modelo no toma como símbolos las muestras de que hablábamos inicialmente (amplitudes de la señal a intervalos fijos), sino otra información relativa a frecuencias, dinámicas, timbres, etc., lo cual no afecta básicamente al razonamiento hecho anteriormente (nuestros símbolos serán algo así como la codificación denominada MIDI -Musical Instrument Digital Interface- utilizada actualmente para la generación de música sintética).

En el caso de la identificación del autor de una obra, sería casi suficiente codificar la secuencia de notas musicales con sus duraciones, prescindiendo de otras informaciones, de manera que independientemente del timbre del instrumento utilizado o de las particularidades de una interpretación, estaríamos comparando la partitura de la nueva obra con modelos internos generados con partituras de autores conocidos.

2 La imposibilidad de computar ciertos números, evidentemente, no puede tratarse de otra cosa que de la imposibilidad de resolver ciertos problemas: aquellos que no tienen solución. Tanto los desarrollos y demostraciones de Turing como la demostración del teorema de Gödel se basan en estrategias de codificación, de modo que a cada problema (o a cada Máquina de Turing) se le puede asociar un número. Al parecer el revisor desconocía el teorema de Gödel -manteniendose en la idea quimérica de Hilbert de la existencia de una base axiomática capaz de resolver cualquier problema- o bien simplemente no se tomo el trabajo de entender la exposición de Turing en la medida necesaria para relacionarlo con dicho teorema. (Hay una tercera posibilidad, quizás más plausible a juzgar por sus comentarios, que consiste en que el revisor fuese un completo ignorante de cualquier fundamento teórico y entendiera la computación como un mero ejercicio mecánico).

3 La rutina "**`fuente`**" es una fuente de emisión de secuencias de símbolos -codificados como letras en el rango A..Z- con una longitud que le es dada como parámetro y una distribución de probabilidad que también recibe como parámetro. El programa principal esta generando una distribución de probabilidad concreta, y con ella solicita a la rutina "**`fuente`**" la generación de una secuencia de 1000 símbolos.

El funcionamiento de la rutina es el siguiente:

- Comienza por inicializar el generador de números aleatorios dándole como "semilla" el valor devuelto por "**`time(NULL)`**", que es el número de milisegundos transcurridos desde una fecha fija hasta el instante de su ejecución, y continúa con un ciclo (un **`while`**) que terminará cuando la variable "**`longitud`**" se haga 0 (es decir cuando se interprete como "**`False`**"). Dentro de este ciclo se emiten los símbolos de acuerdo con una determinada condición y cada vez que se emite un símbolo, se descuenta en una unidad la variable "**`longitud`**" de manera que actúa de contador (descendente) siendo válida como variable de control de permanencia en el ciclo.

- La condición para que un símbolo sea emitido refleja la necesidad de que la distribución de probabilidad final se ajuste a la dada por la matriz "distribución". Para ello se obtiene una letra al azar (**`n=(int)(rand0_1()*NLETRAS)`**) y se emite si su probabilidad asociada (**`distribucion[n]`**) es mayor que un valor aleatorio entre 0 y 1 (de este modo la probabilidad de que el símbolo se emita es directamente proporcional a su probabilidad en el array. En caso de que esto suceda, se evalúa la segunda parte de la expresión condicional que en realidad no hace otra cosa que descontar 1 a la variable de control "**`longitud`**" (es decir no supone una condición añadida).

4 El prefetch es la capacidad de recoger el código de una operación antes de que haya acabado la ejecución de la anterior. Esto supone una ventaja en el caso de que se aprovechen aquellos tiempos en que la ejecución de una instrucción no realice accesos al exterior del procesador para llevar a cabo este fetch "adelantado". Es decir, esto supone una capacidad de "paralelizar" la ejecución interna de las instrucciones con el fecth de instrucciones en el exterior.

En el caso de nuestro ordenador virtual no tiene sentido establecer el prefetch (cuyo objetivo es evidentemente un aumento en la velocidad de cómputo) siempre y cuando no contemos con que el simulador tenga la posibilidad de ejecutar tareas en paralelo sobre un ordenador multiprocesador o con procesador multinúcleo, y aún así todo, en tal caso cabría plantearse la paralelización general del software quedando en entredicho que la paralelización del prefetch supusiera una ventaja.

En todo caso, el plantearlo a nivel de arquitectura implicaría –a grandes rasgos- la necesidad de un registro interno donde mantener los códigos de operación en espera de su turno de ejecución (llamémosle **`IR0`**), y un cambio de la función combinacional de modo que el fetch se reduzca al paso de **`IR0`** a **`IR`**, y los tres pasos del prefetch (**`PC→ADDR, mem→IR0, PC++`**) se ejecuten en paralelo a los pasos de ejecución de instrucciones.

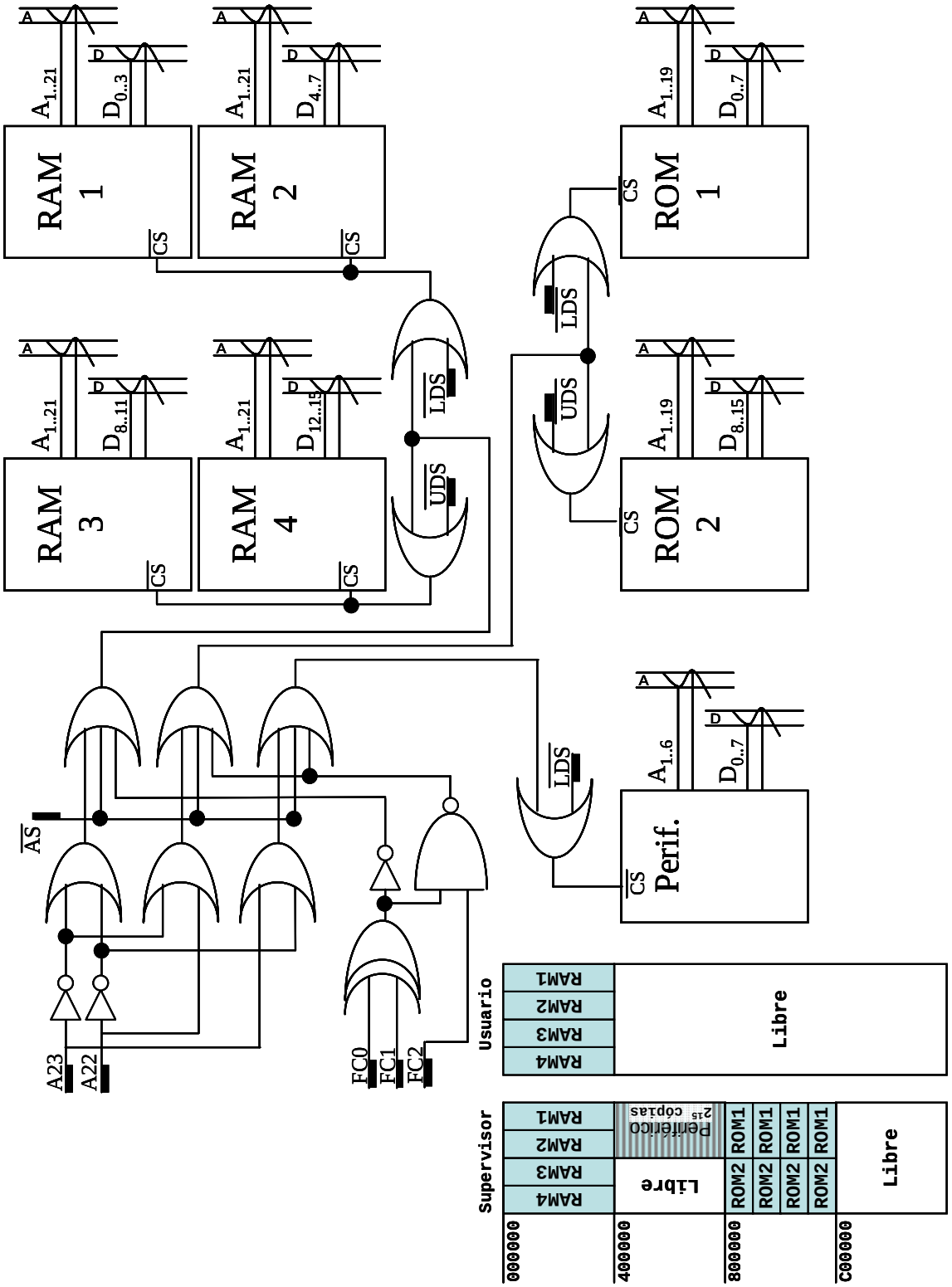
En una solución "correcta" debería detectarse de algún modo cuando el acceso a memoria es posible para aprovechar estos ciclos en la ejecución del prefetch. Una solución mucho menos "elegante" pero más simple adaptada a nuestro computador, puede ser aplicada particularizando los cambios a aplicar a cada instrucción. Como ejemplo puede verse acontinuación lo que podríamos hacer para la instrucción ADD (se ha incluido la paralelización de los pasos 2 y 3 del (pre)fetch que ya era posible anteriormente.)

Instrucción ADD		
Ejecución		Prefetch + fetch
Cargar ADDR desde el IR	1	
Cargar X desde el AC y Cargar Y desde memoria	2	
Seleccionar ADD, Cargar AC desde la ALU, Cargar Flag desde la ALU	3	Cargar ADDR desde el PC
Seleccionar ADD	4	Cargar IR0 desde memoria e Incrementar el PC
Poner COUNT a cero	5	(fetch)Cargar IR desde IR0

Es posible que en algunas ejecuciones de instrucción no se pueda de ejecutar en paralelo el prefetch al no haber pasos suficientes sin acceso a memoria. La solución más simple a este problema será incluir ciclos nulos en la ejecución de modo que generamos artificialmente el "tiempo" necesario para el prefetch.

Naturalmente una solución completa pasa por reconstruir la función combinacional una vez resueltas las secuencias de todas las ejecuciones, para lo que habremos de poner el cuidado suficiente de modo que todo quede correcto (p.ej. en las instrucciones de salto prescindir del prefetch).

5 La siguiente es una posible solución:



6

```
org         $1000

; PROGRAMA PRINCIPAL (escrito para el IDE68K)

            lea     TABLA,a0
            bsr.s   INITAB
            bsr.s   CRIBA

            trap    #15
            dc.w    0


; Subrutina: INITAB
; Descripcion:  Inicializa
;               la tabla rellenandola de $FF-s
; Entrada:    A0.L  dirección de la tabla
; Modifica:   CCR

INITAB      movem.l   a0/d0,-(a7)
            move.w    (a0)+,d0
            bra.s     for3
do3         move.b    #$FF,(a0)+
for3        dbra      d0,do3
            movem.l   (a7)+,a0/d0
            RTS

; Subrutina: TACHA
; Descripción: Tacha en una tabla los
;              multiplos de un numero dado.
; Entrada:    A0.L  dirección de la tabla
;             D0.W  numero
; Modifica:   CCR

TACHA       move.w    d1,-(a7)
            move.w    d0,d1
            add.w     d0,d1
            bra.s     while1
do1         clr.b     1(a0,d1.w)
            add.w     d0,d1
while1      cmp.w     (a0),d1
            bls.s     do1
            move.w    (a7)+,d1
            RTS
```

```
; Subrutina: CRIBA
; Descripción: realiza la criba de
;              Eratostenes sobre una tabla.
; Entrada:    A0.L  dirección de la tabla
; Modifica:   CCR

CRIBA       movem.l   d0/d1,-(a7)
            moveq     #2,d0
            bra.s     while2
do2         bsr.s     TACHA
busca       addq.w    #1,d0
            tst.b     1(a0,d0.w)
            beq.s     busca
while2      move.w    d0,d1
            mulu      d1,d1
            cmp.w     (a0),d1
            bls.s     do2
            movem.l   (a7)+,d0/d1
            RTS



org         $2000

TABLA       dc.w    1000
            ds.b    1000
```