2º Grado Informática
Estructura de Computadores
Ensayo Enero 2016

Nombre:	
DNI:	**Grupo:**

Test de Prácticas (puntuaría sobre 4.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.
Cada respuesta vale 4/20 si es correcta, 0 si está en blanco o claramente tachada, -4/60 si es errónea.
Anotar las respuestas (a, b, c o d) en la siguiente tabla.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
a	b	b	a	b	c	d	b	b	c	b	d	b	c	b	a	c	d	a	b

1. En la convención cdecl estándar para arquitecturas x86 de 32 bits, cuál de las siguientes afirmaciones es cierta:

a. Los parámetros se pasan en pila, de derecha a izquierda; es decir, primero se pasa el último parámetro, después el penúltimo… y por fin el primero.
b. Solamente es necesario salvar el registro EAX
c. Los registros EBX, ESI y EDI son salva - invocante
d. Ninguna de las anteriores es cierta

Re:ver guión P3 pp.1-3

2. En la realización de la práctica de la bomba digital, una parte del código máquina es el siguiente:

0x080486e8 <main+120>: call 0x8048524 <strncmp>
0x080486ed <main+125>: test %eax,%eax
0x080486ef <main+127>: je 0x80486f6<main+134>
0x080486f1 <main+129>: call 0x8048604 <boom>

¿Cuál de los siguientes comandos cambiaría el salto condicional por un salto incondicional?

a. set $0x080486ef=0xeb
b. set *(char*)0x080486ef=0xeb
c. set *(char*)0x080486f6=jmp
d. set %0x080486ef=0xeb

Re:ver P4 p.4

3. Alguna de las siguientes líneas de código sirve para definir una variable entera llamada `tam` en ensamblador (as) Linux x86. ¿Cuál?

a. `var tam : integer;`
b. `tam: .int .-msg`
c. `_int tam = 0`
d. `int tam;`

Re:ver P1 tr.10. a: Pascal, c: inventado, d: lenguaje C

4. La práctica "popcount" debía calcular la suma de bits de los elementos de un array. Un estudiante entrega la siguiente versión de popcount4:

```
int popcount4(unsigned* array,
                      int len) {
  int i, j, res = 0;
  for(i = 0; i < len; ++i) {
    unsigned x = array[i];
    int n = 0;
    do {
      n += x & 0x01010101L;
      x >>= 1;
    } while(x);
    for(j = 16; j == 1; j /= 2){
      n ^= (n >>= j);
    }
```

```
    res += n & 0xff;
  }
  return res;
}
```

Esta función popcount4:

a. Produce el resultado correcto
b. Fallaría con **array={0,1,2,3}**
c. Fallaría con **array={1,2,4,8}**
d. No es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

Re:ver P3 p.17. El for interno no da ninguna iteración, aparte del for es igual que la propuesta en clase

5. La práctica "paridad" debía calcular la suma de paridades impar (XOR de todos los bits) de los elementos de un array. Un estudiante entrega la siguiente versión de parity6:

```
int parity6(unsigned * array,
                    int len) {
  int  i, result = 0;
  unsigned x;
  for (i=0; i<len; i++){
    x = array[i];
    asm("mov %[x], %%edx  \n\t"
        "shr $16,  %%edx  \n\t"
        "shr $8,   %%edx  \n\t"
        "xor %%edx,%%edx  \n\t"
        "setp %%dl        \n\t"
        "movzx %%dl, %[x] \n\t"
        : [x] "+r" (x)
        :
        : "edx"
        );
    result += x;
  }
  return result;
}
```

Esta función parity6:

a. Produce el resultado correcto
b. Fallaría con **array={0,1,2,3}**
c. Fallaría con **array={1,2,4,8}**
d. No es correcta pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

Re:ver P3 p.21. Los dos shr pierden bits 16-23 y aun peor, calculan xnor 0 = 1. Acierta por casualidad con c), falla con b)

6. Sea un computador de 32 bits con una memoria caché L1 para datos de 32 KB y líneas de 64 bytes asociativa por conjuntos de 2 vías. Dado el siguiente fragmento de código:

```
int v[262144];
for (i = 0; i < 262144; i += 2)
   v[i] = 9;
```

¿Cuál será la tasa de fallos aproximada que se obtiene en la primera ejecución del bucle anterior?

a. 0 (ningún fallo)
b. 1/2 (mitad aciertos, mitad fallos)
c. 1/8 (un fallo por cada 8 accesos)
d. 1 (todo son fallos)

Re:int=4B, líneas 64B=2^6 -> 2^4=16ints/línea. For salta de 2 en 2 -> 1 fallo por 8 accesos

7. Utilizando la sentencia asm, las denominadas restricciones que se indican al final de dicha sentencia, involucran a:

a. Solamente las entradas
b. Solamente las salidas
c. Solamente los sobrescritos
d. Ninguna de las anteriores es cierta

Re:ver P3 pp.9-11, involucran todo ello

8. El servidor de SWAD tiene dos procesadores Xeon E5540 con 4 núcleos cada uno. Cada procesador tiene 4 caches L1 de instrucciones de 32 KB, 4 caches L1 de datos de 32 KB, 4 caches unificadas L2 de 256 KB y una cache unificada L3 de 8MB. Suponga que un proceso swad, que se ejecuta en un núcleo, tiene que ordenar un vector de estudiantes accediendo repetidamente a sus elementos. Cada elemento es una estructura de datos de un estudiante y tiene un tamaño de 4KB. Si representamos en una gráfica las prestaciones en función del número de estudiantes a ordenar, ¿para qué límites teóricos en el número de estudiantes se

observarán saltos en las prestaciones debidos a accesos a la jerarquía de memoria?

a. 4 / 32 / 512 estudiantes
b. 8 / 64 / 2048 estudiantes
c. 16 / 32 / 64 estudiantes
d. 32 / 256 / 8192 estudiantes

Re: estudiante=4K=2^{12} -> L1=32K=2^{15}=8 estud. L2=256K=2^{18}=64 estud. L3=8M=2^{23}=2048 estud.

9. En la práctica de la cache, el código de line.cc incluye la sentencia

```
for (unsigned long long line=1;
     line<=LINE; line<<=1) { … }
```

¿Qué objetivo tiene la expresión line<<=1?

a. salir del bucle si el tamaño de línea se volviera menor o igual que 1 para algún elemento del vector
b. duplicar el tamaño del salto en los accesos al vector respecto a la iteración anterior
c. volver al principio del vector cuando el índice exceda la longitud del vector
d. sacar un uno (1) por el stream line

Re: line<<=1 significa line=line*2

10. En un sistema Linux x86-64, ¿cuál de las siguientes variables ocupa más bytes en memoria?

a. char a[7]
b. short b[3]
c. int *c
d. float d

Re: ver Tema2.1 tr.42, puntero 64b=8B, a) 7B, b) 6B, d) 4B

11. En una bomba como las estudiadas en prácticas, del tipo…

```
0x0804873f <main+207>: call   0x8048504 <scanf>
0x08048744 <main+212>: mov    0x24(%esp),%edx
0x08048748 <main+216>: mov    0x804a044,%eax
0x0804874d <main+221>: cmp    %eax,%edx
0x0804874f <main+223>: je  0x8048756<main+230>
0x08048751 <main+225>: call   0x8048604 <boom>
0x08048756 <main+230>:  ...
```

la contraseña es...

a. el entero 0x804a044
b. el entero almacenado a partir de la posición de memoria 0x804a044
c. el string almacenado a partir de la posición de memoria 0x24(%esp)
d. ninguna de las anteriores

Re: ver P4 p.3 antepenúltimo ítem, p5 penúltimo item

12. En una bomba como las estudiadas en prácticas, del tipo…

```
0x080486e8 <main+120>: call 0x8048524 <strncmp>
0x080486ed <main+125>: test   %eax,%eax
0x080486ef <main+127>: je    0x80486f6 <main+134>
0x080486f1 <main+129>: call  0x8048604 <boom>
0x080486f6 <main+134>: ...
```

la contraseña es...

a. el valor que tenga %eax
b. el string almacenado a partir de donde apunta %eax
c. el entero almacenado a partir de donde apunta %eax
d. ninguna de las anteriores

Re: ver P4 p.3 item 7, p.5 item 4. Se tenía que haber conseguido que strncmp devuelva 0, habría que ver qué argumentos se le pasan

13. En un procesador de la familia 80x86 una variable de 32 bits, entera con signo, almacenada a partir de la dirección n contiene: 0xFF en la dirección n, 0xFF en la dirección n+1, 0xFF en la dirección n+2 y 0xF0 en la dirección n+3. ¿Cuánto vale dicha variable?

a. -16
b. -251658241
c. 16
d. 4294967280

Re: ver T1 tr.17. Debe ser negativo grande (little-endian), el único negativo grande es b)

14. En la práctica "suma" se pide sumar una lista de 32 enteros SIN signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando acarreos. ¿Cuál es el mínimo valor entero que repetido en toda la lista causaría acarreo con 32bits (sin signo)?

a. **0xfc00 0000**
b. **0xfbff ffff**
c. **0x0800 0000**
d. **0x07ff ffff**

Re:ver P2 pp.9-10,19 Tabla 12. $32=2^5$ así que el MSB en c) llega justo a salirse

15. En la práctica "suma" se pide sumar una lista de 32 enteros CON signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando desbordamiento. ¿Cuál es el valor negativo más pequeño (en valor absoluto) que repetido en toda la lista causaría desbordamiento con 32bits (en complemento a 2)?

a. **0xfc00 0000**
b. **0xfbff ffff**
c. **0xf800 0000**
d. **0xf800 0001**

Re:ver P2 pp.10-11,19 Tabla 13. $32=2^5$ así que a) llega justo a convertirse en 0x80000000 sin overflow. b) sería el primer valor en salirse. c) y d) son más gordos, se salen antes

16. ¿Qué valor contendrá edx tras ejecutar las siguientes instrucciones?

```
xor %eax, %eax
sub $1, %eax
cltd
idiv %eax
```

a. 0
b. 1
c. -1
d. no puede saberse con los datos del enunciado

Re:ver P2 p.16 Tabla 5. EDX:EAX=-1, EAX=-1, cociente 1, resto 0 en EDX

17. En la práctica "paridad" se pide calcular la suma de paridades de una lista de enteros sin signo. Suponer que un estudiante entrega la siguiente versión

```
int paridad5(unsigned* array,
             int len) {
  int i, k, result = 0;
  unsigned x;
  for (i = 0; i < len; i++) {
    x = array[i];
    for (k = 16; k == 1; k /= 2)
      x ^= x >> k;
    result += (x & 0x01);
  }
  return result;
}
```

Esta función:

a. es correcta
b. falla para **array={0,1,2,3}**
c. falla para **array={1,2,3,4}**
d. no se puede marcar una y sólo una de las opciones anteriores

Re:ver P3 p.21. El for interno no da ninguna iteración, lo que se calcula es por tanto el popcount de los bits menos significativos. La paridad de los ejemplos b),c) debería dar 2,3, se calcula 2,2, coincide b) por casualidad.

18. Abajo se ofrece el listado de una función para multiplicar matrices C = A x B.

```
void mult_matr(   float A[N][N],
   float B[N][N], float C[N][N]){
   /* Se asume valor inicial C = {0,0...} */
int i,j,k;
for (i=0; i<N; i++)
 for (j=0; j<N; j++)
  for (k=0; k<N; k++)
   C[i][j] += A[i][k] * B[k][j];
}
```

Suponer que:

• El computador tiene una cache de datos de 8 MB, 16-vías, líneas de 64 bytes.
• N es grande, una fila o columna no cabe completa en cache.
• El tamaño de los tipos de datos es como en IA32.
• El compilador optimiza el acceso a C[i][j] en un registro.

Aproximadamente, ¿qué tasa de fallos se podría esperar de esta función para valores grandes de N?

a. 1/16
b. 1/8
c. 1/4
d. 1/2

Re: ver T2.4 tr.24-28, se almacena en row-major. En el bucle interno A[i][k] va accediendo a enteros contiguos, B[k][j] no. Línea $64B=2^6B=2^4$ints. Casi siempre (15 de cada 16 veces) A[i][k] es acierto y B[k][j] es fallo (tasa ½, si C[i][j] está optimizado). Tendrían que ser líneas de 2ints=8B para llegar a tasa de aciertos ¼. Basta que N sea grande como para que una fila no quepa en una línea, para que B[k][j] sea siempre fallo y sea imposible que tasa fallos < ½.

19. Con los mismos supuestos, imaginar que se modifica la última sentencia (el cuerpo anidado) por esta otra

```
C[i][j] += A[i][k] * B[j][k];
```

de manera que se calcule C = A x B' (A por traspuesta de B). Aproximadamente, ¿qué tasa de fallos se podría esperar de esta nueva función para valores grandes de N?

a. 1/16
b. 1/8
c. 1/4
d. 1/2

Re: ahora 1 de cada 16 veces ambos A[i][k] y B[j][k] son fallos, las restantes 15 veces son aciertos, tasa fallos = 2/32 = 1/16

20. Suponer una memoria cache con las siguientes propiedades: Tamaño: 512 bytes. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Suponer que para la siguiente secuencia de direcciones enviadas a la cache: 0, 2, 4, 8, 16, 32, la tasa de acierto es 0.33. ¿Cuál es el tamaño de bloque de la cache?

a. 4 bytes
b. 8 bytes
c. 16 bytes
d. Ninguno de los anteriores

Re: ver T6, tr. 24-25. 6 accesos, A=1/3 -> 2 aciertos. Acceso a 0 debe ser fallo, aciertos son 2,4 (más cercanos a 0), resto deben ser fallos (8,16,32) -> tamaño de línea 8B (suponiendo memoria de bytes)