2º Grado Informática
Estructura de Computadores
14 Febrero 2014

Nombre:	
DNI:	**Grupo:**

Examen de Prácticas (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

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

2. En la convención cdecl estándar para arquitecturas x86 de 32 bits, el resultado de una función se devuelve usualmente en el registro:

a. EBX
b. EBP
c. EAX
d. ESI

3. 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

4. Suponga la siguiente sentencia asm en un programa:

```
asm(" add (%[a],%[i],4),%r"
:[r] "+r" (result)
:[i] "r" (i),
 [a] "r" (array)  );
```

¿Cuál de las siguientes afirmaciones es correcta?

a. r es una posición de memoria de entrada/salida
b. a es una posición de memoria de entrada
c. i es un registro de entrada
d. el código de retorno de la función asm se fuerza a que esté en la variable result

5. 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 las siguientes afirmaciones 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

6. El punto de entrada de un programa ensamblador en gcc/as Linux x86 se llama

a. main
b. begin
c. _start
d. _init

7. Alguna de las siguientes líneas de código sirve para definir una variable entera llamada tam en gcc/as Linux x86. ¿Cuál?

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

8. ¿Qué hace gcc –O1?

a. Compilar .c->.o (fuente C a objeto)
b. Compilar .s->.o (fuente ASM a objeto)
c. Ambas (a) y (b), según la extensión de los ficheros que se usen como argumentos
d. Compilar con optimización

9. ¿Cuál de las siguientes afirmaciones es falsa respecto al lenguaje C?

a. Antes de volver de la rutina llamada, el programa en C se encarga de quitar de la pila los parámetros de llamada realizando varios pop
b. Al llamar a una subrutina o función se introducen los parámetros en la pila y después se realiza una llamada a la subrutina
c. Los parámetros se introducen en la pila en el orden inverso a como aparecen en la llamada de C, es decir, empezando por el último y acabando por el primero
d. Pasar a una función un puntero a una variable se traduce en introducir en la pila el valor de la dirección de memoria donde está almacenada la variable

10. Para averiguar la paridad de un número se puede usar la operación:

a. AND
b. NAND
c. XOR
d. OR

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>:  ...
```

…el código numérico (pin) es...

a. el entero 0x804a044
b. el entero cuya dirección está almacenada en la posición de memoria 0x804a044
c. el entero almacenado a partir de la posición de memoria 0x24(%esp)
d. el entero almacenado a partir de la posición de memoria 0x804a044

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 0x80486f6
c. el entero almacenado a partir de donde apunta %eax
d. ninguna de las anteriores

13. 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

---

14. 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

---

15. ¿Cuál de los siguientes registros tiene que ser salvaguardado (si va a modificarse) dentro de una subrutina según la convención cdecl para IA32?

a. eax
b. ebx
c. ecx
d. edx

---

16. 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)

---

17. 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

18. En la práctica de cache hemos hecho una gráfica con el código size.cc ¿Qué forma tiene la gráfica que se debe obtener?

a. Forma de U (o V) con un tramo descendente y otro ascendente
b. Forma de U (o V) invertida, con un tramo ascendente y otro descendente
c. Forma de /, una gráfica siempre creciente y sin escalones
d. Una escalera con varios tramos horizontales

19. 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

20. En la práctica de la cache, el código de size.cc accede al vector saltando de 64 en 64. ¿Por qué?

a. Porque cada elemento del vector ocupa 64 bytes
b. Para recorrer el vector más rápidamente
c. Porque el tamaño de cache L1 de todos los procesadores actuales es de 64KB
d. Para anular los aciertos por localidad espacial, esto es, que sólo pueda haber aciertos por localidad temporal