2º Grado Informática
Estructura de Computadores
18 de enero de 2018

Nombre:	
DNI:	**Grupo:**

Test de Prácticas (4.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.
Cada respuesta vale 0.2p si es correcta, 0 si está en blanco o claramente tachada, -0.0$\widehat{6}$p si es errónea.
Anotar las respuestas (a, b, c ó 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. ¿Cuál de los siguientes fragmentos es correcto para comenzar un programa en ensamblador que conste de un solo archivo .s?

a.
```
.text:
_start:
```
b.
```
.text
.local _start
_start:
```
c.
```
.text
.start _global
_start:
```
d.
```
.text
.global _start
_start:
```

---

2. Suponga una memoria cache con las siguientes propiedades: Tamaño: 512 bytes. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Suponga que para la siguiente secuencia de direcciones enviadas a la cache: 1, 2, 4, 8, 16, 32, la tasa de acierto es 0,333. ¿Cuál es el tamaño de línea de la cache?

a. 4 bytes
b. 8 bytes
c. 16 bytes
d. 32 bytes

---

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

a. Los 6 primeros parámetros se pasan a través de registros
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

---

4. ¿Cuál es el popcount (peso Hamming, nº de bits activados) del número 0x10101010?

a. 4
b. 8
c. 16
d. 32

---

5. Compilar de fuente C a ejecutable usando sólo as y ld, sin gcc...

a. Se puede, repartiendo entre as y ld los modificadores (switches) que corresponda
b. Basta usar ld, con los modificadores de gcc que corresponda, y añadiéndole el runtime de C
c. Se puede, repartiendo modificadores entre as y ld, y añadiendo al comando ld el runtime de C
d. No se puede

---

6. La función gettimeofday() en la práctica de popcount y parity se utiliza para

a. Comparar las duraciones de las distintas soluciones del programa
b. Imprimir la fecha y hora
c. Cifrar el código en función de la hora actual
d. Cronometrar lo que tarda el usuario en pulsar una tecla

---

7. ¿Cuál de los siguientes registros tiene que ser salvaguardado (si va a modificarse) dentro de

una subrutina según la convención cdecl para IA32?

a. ECX
b. EAX
c. EBP
d. EDX

---

8. ¿Qué hace gcc -O1?

a. Compilar .s→.o (fuente ASM a objeto)
b. Compilar con optimización
c. Compilar .c→.o (fuente C a objeto)
d. Compilar .c→.s (C→ASM sin generar objeto)

---

9. Dada la siguiente definición de datos:

```
lista: .int 0x10000000, 0x50000000,
            0x10000000, 0x20000000
longlista: .int (.-lista)/4
resultado: .quad 0x123456789ABCDEF
formato: .ascii "%llu=%llx hex\n\0"
```

La instrucción para copiar la dirección de memoria donde comienza lista en el registro EBX es:

a. movl lista, %ebx
b. movl $lista, %ebx
c. movl (lista), %ebx
d. movl $lista, (%ebx)

---

10. En la práctica "media" se programa la suma de una lista de 32 enteros de 4 B para producir un resultado de 8 B, primero sin signo y luego con signo. Si la lista se rellena con el valor 0x8000 0000, ¿en qué se diferencian los resultados de ambos programas?

a. no se diferencian
b. en uno ocupa 32 bits, en otro 64 bits
c. en uno los 16 bits superiores son 0xFFFF, en el otro no
d. en uno los 16 bits inferiores son 0x0000, en el otro no

---

11. En la práctica "media" se pide sumar una lista de 32 enteros CON signo de 32bits en una plataforma de 32bits sin perder precisión, esto es, evitando desbordamiento. Un estudiante entrega un programa que se diferencia de la versión recomendada en el siguiente bucle, en particular en la instrucción adc.

```
bucle:
   mov (%ebx,%esi,4), %eax
   cltd
   add %eax, %edi
   adc %eax, %ebp
   inc %esi
   cmp %esi, %ecx
   jne bucle
```

Esta versión de la suma CON signo

a. produce siempre el resultado correcto
b. fallaría con lista: .int 1, 1, 1, 1, ...
c. fallaría con lista: .int -1,-1,-1,-1, ...
d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

---

12. En la práctica "media" un estudiante usa el siguiente bucle para acumular la suma en EBP:EDI antes de calcular la media y el resto

```
bucle:
   mov (%ebx,%esi,4), %eax
   cltd
   add %eax, %edi
   adc %edx, %ebp
   jnc nocarry
   inc %edx
nocarry:
   inc %esi
   cmp  %esi,%ecx
   jne bucle
```

Este código es una mezcla de las soluciones recomendadas para suma sin signo y para suma con signo. Estando bien programado todo lo demás, este código

a. produce siempre el resultado correcto
b. fallaría con lista: .int 0,1,2,3
c. fallaría con lista: .int -1,-2,-4,-8
d. no siempre produce el resultado correcto, pero el error no se manifiesta en los ejemplos propuestos, o se manifiesta en ambos

---

13. ¿Cuál es el popcount (peso Hamming, nº de bits activados) de una lista de N números inicializada con los valores 0..N-1?

a. (N-1)*N/2
b. N*(N+1)/2
c. si N es par, N*(N/2)
d. si N es potencia de 2, log_2(N)*N/2

---

14. ¿Cuál es la paridad (XOR "lateral" de todos los bits) del número 199?

a. 0
b. 1
c. 2
d. 4

---

15. Comparando los popcounts (pop(199) vs. pop(99)) y paridades (par(199) vs. par(99)) de los números 199 y 99, se verifica que

a. pop(199) > pop(99)
b. par(199) < par(99)
c. pop(199) < par(99)
d. par(199) > pop(99)

---

16. ¿Cuál es la suma de paridades (suma de los XOR "laterales" de los bits de cada número) de una lista de N números inicializada con los valores 0..N-1?

a. (N-1)*N/2
b. N*(N+1)/2
c. si N es par, N/2
d. si N es potencia de 2, (log_2(N)*N)/2

---

17. En la práctica "popcount/paridad", para cronometrar sistemáticamente las diversas versiones necesitamos una función crono() a la que se le pueda pasar como argumento cuál versión queremos cronometrar. En lenguaje C esto se puede hacer con punteros a funciones. Sabiendo que todas las versiones devuelven un valor entero, el prototipo de la función crono() debería ser:

a. void crono(int * func (), char* msg);
b. void crono(int (* func)(), char* msg);
c. void crono((int *)func (), char* msg);
d. void crono(int * func , char* msg);

---

18. Para corregir la práctica "bomba digital", un profesor dispone de 26 ejecutables (y la lista de claves correspondientes) para asignar en el Laboratorio una bomba distinta a cada estudiante. Cuando un estudiante diga que la ha resuelto el profesor exigirá ver al estudiante ejecutando la bomba y tecleando la contraseña y el pin correctos para comprobar que no explota, para así anotarla como resuelta y que le puntúe al estudiante.

Un estudiante (usando ordenador del Laboratorio con Ubuntu 10.04) dice que ha resuelto su bomba, y cuando el profesor pide que se tecleen las claves, el ddd se "bloquea" y le empieza a "parpadear" al estudiante. Para poder puntuar, lo más recomendable para el estudiante sería…

a. teclear las claves (contraseña y pin), aunque ddd esté "bloqueado" y "parpadeando"
b. probar en orden los remedios básicos: pulsar <Ctrl>-C varias veces, pulsar <Enter> repetidamente, comprobar que está seleccionada "Machine Code Window", teclear "info line main", y si todo falla, ejecutar "`rm -rf ~/ .ddd`"
c. matar la ventana ddd, abrir un terminal con un shell, ejecutar la bomba desde línea de comandos y teclear las claves
d. reinstalar un paquete ddd más actualizado usando "sudo apt-get install"

---

19. Suponer el mismo contexto de la pregunta anterior, donde el profesor tiene una lista de las 26+26 claves (contraseñas y pines)

Un estudiante dice que ha resuelto su bomba, y cuando el profesor pide que se tecleen las claves, la contraseña coincide con la que tiene anotada el profesor, pero el pin no, y de todas formas la bomba no explota. Debería hacerse lo siguiente:

a. la bomba tiene que puntuarle al estudiante porque no ha explotado
b. el profesor puede pedirle que vuelva a descargar la bomba original e intente repetir con ese ejecutable las claves que acaba de teclear
c. la bomba debe marcarse como inválida y no hay que hacer más comprobaciones
d. no puede suceder lo que dice el enunciado, y en ningún caso el profesor tiene derecho a hacer comprobaciones adicionales como pedir que se vuelva a descargar la bomba

---

20. 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
