2º Grado Informática
Estructura de Computadores
31 de enero de 2020

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. En la práctica "media" se pide como ejercicio previo sumar una lista de N enteros sin signo en doble precisión acumulando acarreos. Para aplicar esta técnica se sugiere utilizar la(s) instrucción(es)...

a. ADC $0, ...
b. CLTD, ADD y ADC
c. JNC y INC
d. CLTQ, CQTO y IDIV

2. En la práctica "media" se pide como ejercicio previo sumar una lista de N enteros sin signo en doble precisión extendiendo con ceros. Para aplicar esta técnica se sugiere utilizar la(s) instrucción(es)...

a. ADC $0, ...
b. CLTD, ADD y ADC
c. JNC y INC
d. CLTQ, CQTO y IDIV

3. En la práctica "media" se pide como ejercicio previo sumar una lista de N enteros en doble precisión extendiendo el signo. Para aplicar esta técnica se sugiere utilizar la(s) instrucción(es)...

a. ADC $0, ...
b. CLTD, ADD y ADC
c. JNC y INC
d. CLTQ, CQTO y IDIV

4. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de popcount5:

```
int popcount5(int* array, size_t len){
    size_t i,j,val;
    long x;
    int result=0;
    for (i=0; i<len; i++){
      x = array[i];
      val = 0;
      for (j=0; j<8; j++){
        val += x & 0x0101010101010101L;
        x >>= 1;
      }
      val += (val >> 32);
      val += (val >> 16);
      val += (val >>  8);
    result+=  val & 0xFF;
    }
    return result;
}
```

Esta función no sigue la adaptación a 32 bits recomendada en clase como versión "oficial", y conserva muchos vestigios de la versión de 64 bits, como los tipos de **array, x, val**, la máscara **0x...01L** y el desplazamiento **>>32**.

Esta función popcount5 fallaría:

a. con array={0x80000000, 0x00400000, 0x00000200, 0x00000001}
b. con array={0, -1, -2, -3}
c. con ambos ejemplos
d. con ninguno de los dos ejemplos

5. La práctica "popcount" debía calcular la suma de bits (peso Hamming) de los elementos de un array. Un estudiante entrega la siguiente versión de popcount6:

```
int pc6(unsigned* array, size_t len){
  size_t i;
  unsigned x;
```

```
  int result=0;

  const unsigned m1  = 0x55555553;
  const unsigned m2  = 0x33333333;
  const unsigned m4  = 0x0f0f0f0f;
  const unsigned m8  = 0x00ff00ff;
  const unsigned m16 = 0x0000ffff;

  for (i=0; i<len; i++){
   x = array[i];

   x = (x & m1 ) + ((x >>  1) & m1 );
   x = (x & m2 ) + ((x >>  2) & m2 );
   x = (x & m4 ) + ((x >>  4) & m4 );
   x = (x & m8 ) + ((x >>  8) & m8 );
   x = (x & m16) + ((x >> 16) & m16);

   result+= x;
  }
  return result;
}
```

Esta función sólo se diferencia de la versión "oficial" recomendada en clase en algún dígito de alguna máscara.

Esta función popcount6 fallaría:

a. con array={0x80000000, 0x00400000, 0x00000200, 0x00000001}
b. con array={0x10000000, 0x00200000, 0x00000400, 0x00000008}
c. con ambos ejemplos
d. con ninguno de los dos ejemplos

---

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

```
400746: lea    0xc(%rsp),%rsi
40074b: lea    0x1ab(%rip),%rdi
              # 4008fd <stdin_u+0x10d>
400752: mov    $0x0,%eax
400757: callq  400590 <scanf@plt>
40075c: cmpl   $0x400600,0xc(%rsp)
400764: je     40076b <main+0x8c>
400766: callq  400697 <boom>
40076b: callq  4006bb <defused>
```

...el código numérico (pin) es...

a. el entero 0x400590
b. el entero 0x400600
c. el entero almacenado a partir de 0xc(%rsp)
d. el entero alm. a partir de 0x1ab+0x400752

---

7. En la práctica “blink” de Arduino, un estudiante muestra a los profesores lo que le han enseñado en algún otro lugar: conectando un led directamente entre las patillas *Digital pin 13* (LED_BUILTIN) y GND (justo al lado) también parpadea. Cabe esperar la siguiente respuesta de los profesores:

a. Exactamente esa solución viene en el guión, no hace falta aprenderla fuera de clase
b. Nosotros conectamos los led en serie con una resistencia como dice el guión
c. Hay que revisar ese equipo en concreto, debe tener algún defecto, ese led no debe parpadear
d. Es cierto, es un error de diseño del guión, es más sencillo conectarlo así

---

8. Suponga una memoria cache con las siguientes propiedades: Tamaño: 32 KB. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Tamaño de línea 64 B. Dado el siguiente fragmento de código:

```
int x[262144], y[262144], a=0;
for (i = 0; i < 262144; i ++)
  a += x[i] * y[i];
```

¿Cuál será la tasa de fallos aproximada que se obtiene en la ejecución del bucle anterior? Suponer que el tamaño de los tipos de datos es como en x86-64, y que el compilador optimiza el acceso a las variables a/i en un registro.

a. 1/2 (un fallo por cada 2 accesos)
b. 1/4 (un fallo por cada 4 accesos)
c. 1/8 (un fallo por cada 8 accesos)
d. 1/16 (un fallo por cada 16 accesos)

---

9. En el programa line.cc de la práctica "cache", se accede al vector saltando...

a. de byte en byte
b. de 64 en 64 bytes
c. de 1 KB en 1 KB
d. de line en line bytes, donde line barre los valores desde 1 hasta 1K en un bucle for

---

10. En la práctica de cache hemos hecho una gráfica con el código line.cc ¿Qué forma tiene la gráfica que se puede obtener?

a. Forma de U (o V), con un tramo descendente y otro ascendente
b. Forma de /, una gráfica siempre creciente
c. Forma de media U seguida de \, es decir, un tramo descendente suave, un pequeño tramo horizontal, y un tramo descendente lineal
d. Una escalera con varios tramos horizontales

---

11. ¿Cuál de las siguientes instrucciones es incorrecta?

a. shr $1,%rdx
b. shr %rdx
c. shr %cl,%rdx

d. shr %rcx,%rdx

12. En la práctica "media" se pide sumar una lista de 16 enteros CON signo de 32 bits en una plataforma de 32 bits sin perder precisión, esto es, evitando overflow. ¿Cuál es el mayor valor negativo (menor en valor absoluto) que repetido en toda la lista de 16 enteros causaría overflow con 32bits?

PISTA: Sumar un número 16 veces == multiplicarlo por 16 == desplazarlo 4 bits a la izquierda

a. 0xffff ffff
b. 0xfc00 0000
c. 0xf7ff ffff
d. 0xf000 0000

13. La función siguiente, que aparece en el esqueleto suma_09 presentado en la práctica "popcount"

```
int suma3(int* array, int len){
 asm("mov  $0, %%eax                \n"
"     mov  $0, %%rdx                \n"
"bucle:                             \n"
"     add  (%%rdi,%%rdx,4), %%eax   \n"
"     inc   %%rdx                   \n"
"     cmp   %%rdx,%%rsi             \n"
"     jne    bucle                    "
     :                    // output
     :                    // input
     : "cc",              // clobber
       "rax","rdx"
    );
}
```

no incluye la instrucción **ret** dentro de la sentencia **asm** porque:

a. porque la función no devuelve nada
b. porque suma3 es un puntero a función
c. porque la incluye automáticamente el compilador
d. porque suma3 no se llamará con call sino con jmp

14. Respecto a la inicialización de los elementos del array en el esqueleto suma_09 de la práctica "popcount":

```
#define SIZE (1<<16)
for (i=0; i<SIZE; i++)
    lista[i]=i;
```

a. Cualquiera de las tres funciones suma del programa suma_09_Casm irán más rápido con esta inicialización que con números aleatorios.
b. Las tres funciones suma del programa suma_09_Casm irán igual de rápido con esta inicialización que con números aleatorios.
c. popcount1 (bucle interno for) irá más rápido con esta inicialización que con números aleatorios
d. popcount2 (bucle interno while) irá más lento con esta inicialización que con números aleatorios.

15. Un estudiante entrega la solución de una bomba y en ella aparece el comando de gdb "**p(char*)$rdi**". Este comando:

a. muestra únicamente el contenido de %dil
b. es erróneo sintácticamente
c. muestra el valor de la dirección de memoria contenida en %rdi y el string almacenado a partir de esa dirección de memoria
d. muestra el contenido de %rdi interpretado como un string de 8 caracteres

16. En la realización de la práctica de la bomba digital compilada en 32 bits, una parte del código máquina es el siguiente:

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

¿Cuál de los siguientes comandos de gdb 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

17. ¿Cuál de las siguientes patillas (pin) no se menciona (no se pide usar/no se debe usar) en la 1ª práctica del Theremín de luz de Arduino? (Proyecto p06 / sólo fotocélula y zumbador)

a. Analog In A0
b. AREF
c. ledPin = 13
d. GND

18. En la práctica del Theremín de luz de Arduino se pide combinar el código del Proyecto p06 (sólo fotocélula y zumbador) con el de la Lección 26 (fotocélula y leds). Para conseguir que se enciendan y apaguen correctamente todos los led, lo mejor sería:

a. Reutilizar la variable pitch calculada mediante map() como la nueva numLEDSLit

b. Fusionar las variables sensorValue y reading de forma que sólo se llame una vez a analogRead()
c. Calcular numLEDSLit con otra función map() de 0 a 8, en lugar de barrer de 50 a 4000
d. Copiar uno y otro código en las secciones correspondientes, no se puede fusionar nada

---

19. 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 64 KB.
d. Para anular los aciertos por localidad espacial, es decir, para que sólo pueda haber aciertos por localidad temporal.

---

20. Un servidor tiene dos procesadores Intel Xeon E5-2620 v3@ 2.40GHz (2,4 Ghz, 6 núcleos, 12 hebras, 15MiB de caché L3, reloj DDR4 a 1866 MHz). Suponga que un proceso, que se ejecuta en un único núcleo, tiene que ordenar un vector de datos de usuarios accediendo repetidamente a sus elementos. Cada elemento es una estructura de datos y tiene un tamaño de 4 KB. Según http://www.cpu-world.com/ los tamaños de cache son:

Cache:	L1 data	L1 instr.	L2	L3
Size:	6 x 32KB	6 x 32KB	6 x 256KB	15MB

Si representamos en una gráfica las prestaciones en función del número de usuarios a ordenar, ¿para qué límites teóricos en el número de usuarios se observarán saltos en las prestaciones debidos a accesos a la jerarquía de memoria?

a. 4 / 32 / 512 usuarios
b. 8 / 64 / 3840 usuarios
c. 48 / 384 / 23040 usuarios
d. 32 / 256 / 8192 usuarios
