2º Grado Informática
Estructura de Computadores
3 Septiembre 2014

Nombre:	
DNI:	**Grupo:**

Examen de Problemas (3,0 p)

1. **Disposición de estructuras en memoria** (0.5 puntos). Considerar las siguientes declaraciones de estructuras en lenguaje C:

```
struct a {
     float* f;
     char c;
     int i;
     char z[4];
     double d;
     short s;
};
```

```
struct b {
     struct a a1;
     int j;
     struct a a2;
};
```

A. Mostrar la disposición de **`struct a`** en memoria, en una máquina Linux en modo IA32 (es decir, con direcciones de 32bits), mediante un dibujo como los estudiados en clase, donde se indiquen los desplazamientos y tamaños de cada campo. Marcar los bytes de relleno (si los hubiera) tachándolos o sombreándolos. ¿Cuántos bytes usa **`struct a`** en total en este caso?

B. Mostrar la disposición de **`struct a`** en memoria (tachando o sombreando los bytes de relleno si los hubiera), en una máquina Linux en modo x86-64 (es decir, con direcciones de 64bits). ¿Cuántos bytes usa **`struct a`** en total en este caso?

C. ¿Cuántos bytes usa **`struct b`** en una máquina Linux en modo IA32 (con direcciones de 32bits)?

D. ¿Cuántos bytes usa **`struct b`** en una máquina Linux en modo x86-64 (direcciones de 64bits)?

2. **Ensamblador IA32** (0.5 puntos). Traduzca a ensamblador de IA32 la siguiente función escrita en C:

```
int hex2bin (int c) {
   if (c >= '0' && c <= '9')
      return c - '0';
   if (c >= 'A' && c <= 'F')
      return c + 10 - 'A';
   if (c >= 'a' && c <= 'f')
      return c + 10 - 'a';
   return -1;  // Error code
}
```

Códigos de los caracteres:

- '0' → 48
- '9' → 57
- 'A' → 65
- 'Z' → 70
- 'a' → 97
- 'z' → 102

3. **Código Ensamblador x86-64**. (0.5 puntos). Para cada una de las siguientes funciones en lenguaje C (a la izquierda) indicar la letra del bloque de código ensamblador x86-64 correspondiente (a la derecha).

```
long int func1(long int a)
{
  long int i;
  if (a > 100)
    return a * 5;
  for (i = 0; i < 10; i++) {
    a = a * 5;
  }
  return a;
}
```

func1 *corresponde con asm:____*

```
long int func2(long int a)
{
  long int i = 0;
  while (i < 10) {
    a = a * 5;
    i++;
  }
  if (a > 100)
    return a;
    else return a * 5;
}
```

func2 *corresponde con asm:____*

```
long int func3(long int a)
{
  long int i = 0;
  do {
    if(a > 100)
      return a;
    a = a * 5;
    i++;
  } while(i < 10);
  return a;
}
```

func3 *corresponde con asm:____*

(A)

```
0: 48 8d 04 bf      lea (%rdi,%rdi,4),%rax
4: 48 83 ff 63      cmp $0x63,%rdi
8: 7e 16            jle 0x20
a: 48 0f af ff      imul %rdi,%rdi
e: 48 83 ff 0a      cmp $0xa,%rdi
12: b8 64 00 00 00  mov $0x64,%eax
17: ba 00 00 00 00  mov $0x0,%edx
1c: 48 0f 4c c2     cmovl %rdx,%rax
20: f3 c3           repz retq
```

(B)

```
0: b8 00 00 00 00   mov $0x0,%eax
5: 48 8d 3c bf      lea (%rdi,%rdi,4),%rdi
9: 48 83 c0 01      add $0x1,%rax
d: 48 83 f8 0a      cmp $0xa,%rax
11: 75 f2           jne 0x5
13: 48 83 ff 64     cmp $0x64,%rdi
17: 48 8d 04 bf     lea (%rdi,%rdi,4),%rax
1b: 48 0f 4e f8     cmovle %rax,%rdi
1f: 48 89 f8        mov %rdi,%rax
22: c3              retq
```

(C)

```
0: 48 89 f8         mov %rdi,%rax
3: ba 00 00 00 00   mov $0x0,%edx
8: 48 83 ff 64      cmp $0x64,%rdi
c: 7e 05            jle 0x13
e: 48 8d 04 bf      lea (%rdi,%rdi,4),%rax
12: c3              retq
13: 48 8d 04 80     lea (%rax,%rax,4),%rax
17: 48 83 c2 01     add $0x1,%rdx
1b: 48 83 fa 0a     cmp $0xa,%rdx
1f: 75 f2           jne 0x13
21: f3 c3           repz retq
```

(D)

```
0: 48 89 f8         mov %rdi,%rax
3: 48 83 ff 64      cmp $0x64,%rdi
7: 7f 1f            jg 0x28
9: 48 8d 04 bf      lea (%rdi,%rdi,4),%rax
d: ba 01 00 00 00   mov $0x1,%edx
12: eb 0e           jmp 0x22
14: 48 8d 04 80     lea (%rax,%rax,4),%rax
18: 48 83 c2 01     add $0x1,%rdx
1c: 48 83 fa 0a     cmp $0xa,%rdx
20: 74 06           je 0x28
22: 48 83 f8 64     cmp $0x64,%rax
26: 7e ec           jle 0x14
28: f3 c3           repz retq
```

4. **Unidad de Control** (0.5 puntos). Un procesador con una unidad de control micro-programada tiene una memoria de control de 640 palabras de 70 bits, de las que 280 son diferentes.

   ¿Qué ahorro en número de bits obtendríamos si usáramos nanoprogramación?

   Razone la respuesta con dos dibujos, uno sin nanoprogramación y otro con nanoprogramación.

5. **Configuración de memoria** (0.5 puntos). Considere chips de memoria RAM estática de 32K x 8 con los cuales se quiere construir una memoria direccionable por palabras de 32 bits con un tamaño total de 1 MB (256 K palabras de 32 bits).

A. Exprese como potencia de 2 el nº de posiciones de cada chip
B. Exprese como potencia de 2 el nº de bits de cada posición del chip
C. ¿Cuántas patillas de direccionamiento tiene cada chip?
D. ¿Cuántas patillas de direccionamiento tiene el sistema completo de memoria?
E. ¿Cuántas patillas de datos tiene cada chip?
F. ¿Cuántas patillas de datos tiene el sistema completo de memoria?
G. ¿Cuántos chips son necesarios?
H. ¿Cuántas entradas (bits de dirección) y salidas (CS para los chips de memoria) necesita el decodificador?

6. **Jerarquía de memoria** (0.5 puntos). Suponga los dos diseños de cache siguientes:

Parámetro	Diseño A	Diseño B
Tamaño cache	L1: 1MB	L1: 256KB L2: 768KB
Tasa de acierto	L1: 92% de todos los accesos	L1: 70% de todos los accesos L2: 85% de los accesos que han fallado en L1
Tiempo de acceso	L1: 4 ns MP: 100 ns	L1: 2 ns L2: 8 ns MP: 100 ns

A. Calcule el tiempo promedio de acceso para el diseño A.
B. Calcule el tiempo promedio de acceso para el diseño B.
C. ¿Cuál de los dos diseños es más rápido?