2º Grado Informática
Estructura de Computadores
28 Enero 2013

Nombre:	
DNI:	**Grupo:**

Examen Test (3.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.
Cada respuesta vale 3/30 si es correcta, 0 si está en blanco o claramente tachada, -1/30 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	21	22	23	24	25	26	27	28	29	30

1. En una CPU de 32bits con memoria de bytes, el problema es que...

a. No tiene sentido, un registro no cabría en memoria
b. No hay problema, cuando se salva un registro a memoria se escribe en la posición deseada
c. Hay que usar 4 instrucciones de lectura (o escritura) para leer (o escribir) un registro completo
d. Hay que respetar el ordenamiento de bytes y reglas de alineamiento con que se diseñó la CPU

2. En una memoria de bytes que contuviera a partir de la posición 0 los valores 1,0,0,0,0xFE,0xFF,0xFF,0xFF, se puede decir que...

a. Hay una palabra de 16bit big-endian con valor 1 en la posición 0
b. Hay una palabra de 16bit little-endian con valor 254 en la posición 3
c. Hay una palabra de 32bit little-endian con valor -1 en la posición 4
d. Todas las respuestas anteriores son incorrectas

3. ¿Cuál de las siguientes instrucciones es errónea? (sale mensaje de error al intentar ensamblar):

a. movw %dx, (%eax)
b. pushb $0xFF
c. movswl (%eax), %edx
d. movzbl %dl, %eax

4. La instrucción xor $3, %eax tiene como resultado:

a. Poner a 0 los últimos 3 bits del registro EAX
b. Cambiar 0<->1 (complemento a 1 de) los últimos 2 bits del registro EAX
c. Poner a 1 el último bit del registro EAX
d. Ninguno de los anteriores resultados

5. De entre las siguientes construcciones de flujo de control en lenguaje C, la que se traduce más directamente a lenguaje ensamblador es...

a. El bucle for
b. El bucle while
c. El bucle do-while
d. La selección switch-case

6. GCC/Linux IA32 resuelve el ajuste de marco de pila mediante las instrucciones:

a. movl %esp, %ebp; popl %esp

b. movl %ebp, %esp; popl %ebp
c. pushl %esp; movl %ebp, %esp
d. pushl %ebp; movl %esp, %ebp

7. Respecto a registros salva-invocante y salva-invocado en GCC/Linux IA32, ¿cuál de éstos es de distinto tipo que el resto?

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

8. La convención de llamada Linux/GCC x86-32 considera, respecto a convenios de uso de registros:

a. 3 registros salva-invocante, 3 registros salva-invocado, y 2 especiales
b. 8 registros salva-invocante, 6 registros salva-invocado, y 2 especiales
c. Algunos registros salva-invocante, otros salva-invocado, uno especial
d. Algunos registros para pasar argumentos, otros salva-invocante, otros salva-invocado, dos especiales

9. El marco de pila en x86-64 Linux...

a. no existe, porque RBP no es registro especial en x86-64
b. sólo se crea para funciones que invocan anidadamente a otra función (procedimientos padre, no hojas)
c. se crea para funciones en las que GCC no puede evitar que RBP baje más, como por ejemplo: que haya demasiadas variables locales (y no quepan en registros), o que haya que salvar algún registro salva-invocado
d. se crea para funciones en las que GCC no puede evitar que RSP baje más, como por ejemplo: que haya que calcular la dirección de una variable local, o pasar más de 6 argumentos a otra función

10. En un sistema de 32bits, ¿cuál de las siguientes expresiones C es equivalente a la expresión (x[2] + 4)[3]? Asumir que x se ha declarado como int **x. Recordar que C usa aritmética de punteros. Notar que muchos de los paréntesis no son necesarios, sólo se han añadido para evitar confusiones por precedencia de operadores

a. *((*(x + 8)) + 28)
b. *((*(x + 2)) + 7)
c. (**(x + 2) + 7)
d. *(((*x) + 2) + 7)

11. Justo antes de que una instrucción máquina escriba un resultado en memoria:

a. en IR está el resultado y en MBR la dirección donde se almacenará
b. en IR está el resultado y en MAR la dirección donde se almacenará
c. en MAR está el resultado y en MBR la dirección donde se almacenará
d. en MBR está el resultado y en MAR la dirección donde se almacenará

12. En una arquitectura RISC típica:

a. la UC es más compleja que en una arquitectura CISC
b. la programación resulta mucho más simple que en una arquitectura CISC
c. se usan pocas instrucciones de las disponibles en el conjunto de instrucciones
d. suele usarse segmentación

13. ¿Qué circuito suele utilizarse para traducir el código de operación de una instrucción máquina a dirección de comienzo en la memoria de control del microprograma correspondiente?

a. Una memoria
b. Un multiplexor
c. Un contador
d. Un demultiplexor

14. ¿Cómo actúa el indicador de signo?

a. Se pone a 1 cuando el resultado es negativo
b. Se pone a 1 cuando el resultado es distinto de cero
c. Se pone a 0 cuando el resultado es negativo

d. Se pone a 1 cuando el resultado es positivo

15. ¿Cuál de las siguientes afirmaciones es cierta?

a. La E/S en memoria emplea la patilla IO/M#
b. En E/S independiente, las instrucciones de acceso a memoria se emplean tanto para memoria como para E/S
c. La E/S independiente facilita la protección
d. La E/S en memoria es mucho más rápida que la E/S independiente

16. La técnica de sondeo, escrutinio o "polling"...

a. Se utiliza para identificar la fuente de una interrupción
b. No permite establecer un mecanismo de asignación de prioridades a los distintos dispositivos
c. En caso de utilizarse, es necesario emplear varias líneas para que los dispositivos soliciten una interrupción
d. Es incompatible con el daisy-chain

17. Señale cuál de las siguientes opciones es una técnica habitual para llevar a cabo la transferencia de datos entre el computador y los dispositivos de E/S externos:

a. E/S por nivel
b. E/S por flanco
c. Acceso directo a memoria (DMA)
d. Acceso indirecto a memoria (IMA)

18. ¿Cuál de las siguientes afirmaciones acerca del concepto de interrupción es cierta?

a. Es una bifurcación normalmente externa al programa en ejecución
b. Su objetivo es incrementar el ancho de banda con el dispositivo
c. Solicita que el procesador se aisle de los buses
d. Permite realizar transferencias sin el control de un programa

19. La primera instrucción ensamblador de una subrutina compilada con gcc en Linux/x86 cdecl suele ser:

a. mov %esp, %ebp
b. push %ebp
c. push %ebx
d. pop %ebx

20. En cdecl/x86, ¿cuál de los siguientes registros tiene que ser guardado por la función llamada si es alterado por ésta?

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

21. Se desea construir una memoria de SRAM de tamaño 3G X 8 a partir de elementos de memoria SRAM más pequeños. Cuál de las siguientes soluciones sería correcta:

a. 256 chips de 16Mx 1 bits
b. 16 chips de 512 M x 2 bits
c. 12 chips de 512M x 4 bits
d. Ninguna de las anteriores es correcta

22. Un procesador de 1GHz tarda 4ns en realizar 4 instrucciones sin realizar segmentación de cauce. Cuanto tardaría en realizar 9 instrucciones un procesador con segmentación de cauce de 4 etapas si no existiera ningún retraso en ninguna de las instrucciones.

a. 2 ns
b. 3 ns
c. 4.5 ns
d. 9 ns

23. ¿Cuántas patillas de dirección tiene una memoria DRAM de 1G palabra, siendo la longitud de palabra de 16 bits?

a. 20
b. 16
c. 30
d. 15

24. ¿Cuál de las siguientes afirmaciones es cierta?

a. La memoria SRAM es más lenta que la DRAM
b. La lectura en la memoria SRAM es destructiva
c. La memoria DRAM es más cara que la SRAM
d. Ninguna de las anteriores

25. La ganancia en velocidad de un cauce de K etapas de igual duración ejecutando un programa de N instrucciones es:

a. S = KN/(K-N+1)
b. S = NKT/(N-K+1)T
c. S = KN/(K+N-1)
d. S = NT/(N+K-1)T

26. ¿A qué tipo de localidad de memoria hace referencia la siguiente afirmación: “si se referencia un elemento, tenderá a volver a ser referenciado pronto"?

a. Localidad espacial
b. Localidad lógica
c. Localidad temporal
d. Ninguna de las respuestas anteriores es correcta

27. ¿Cuáles de las siguientes direcciones de memoria podrían estar simultáneamente en una memoria caché con correspondencia directa de 256 palabras con 16 palabras por bloque?

a. 0000 y FF0F
b. ABAB y ABAC
c. 08E3 y 74E1
d. Ninguna de las combinaciones anteriores

28. En una memoria DRAM que permite el acceso en modo página se accede a la palabra 0x1234. Si emplea páginas de 256 palabras, ¿Cuál será la menor dirección a la que podremos acceder rápidamente?

a. 0x1000
b. 0x1200
c. 0x1230
d. Otra

29. ¿Cuál de las siguientes afirmaciones es cierta?

a. Al realizar la segmentación de cauce aumenta en general el tiempo necesario para la ejecución de un programa
b. Debido a que pueden existir dependencia de datos, los resultados de un programa pueden ser diferentes a si el programa se ejecutara sin segmentación
c. La segmentación de cauce disminuye el número de instrucciones necesarias para la ejecución de un programa
d. Ninguna de las combinaciones anteriores

30. En un procesador con segmentación de cauce, aumentar el número de etapas (p.ej. de 2 a 4, o de 4 a 8), tiene en general como consecuencia:

a. Un incremento de las prestaciones
b. Un mayor retraso en la ejecución de los programas debido al incremento del número de etapas
c. Una disminución en la posible dependencia de datos
d. Una disminución de la máxima frecuencia de reloj a la que puede operar el cauce

2º Grado Informática
Estructura de Computadores
28 Enero 2013

Nombre:	
DNI:	**Grupo:**

Examen de Problemas (3.0p)

1. **Bucles while** (0.5 puntos). Una función, fun_a, tiene la siguiente estructura general:

```
int fun_a(unsigned x) {
    int val = 0;
    while ( ________ ) {
        ________;
    }
    return ________;
}
```

El compilador GCC genera el siguiente código ensamblador:

```
# x en %ebp+8
   movl   8(%ebp), %edx
   movl   $0, %eax
   testl  %edx, %edx
   je     .L7
.L10:
   xorl   %edx, %eax
   shrl   %edx         # Desplazar a derecha 1
   jne    .L10
.L7:
   andl   $1, %eax
```

Analizar el funcionamiento de este código y responder a las siguientes preguntas:

A. Usar la versión en código ensamblador para rellenar las partes que faltan del código C.

B. Describir en castellano qué calcula esta función.

2. **Representación y acceso a estructuras** (0.5 puntos). Para cada una de las siguientes declaraciones de estructuras, determinar el desplazamiento de cada campo, el tamaño total de la estructura, y sus requisitos de alineamiento bajo Linux/IA32.

A. `struct P1 { int i; char c; int j; char d; };`

B. `struct P2 { int i; char c; char d; int j; };`

C. `struct P3 { short w[3]; char c[3]; };`

D. `struct P4 { short w[3]; char *c[3]; };`

E. `struct P5 { struct P1 a[2]; struct P2 *p };`

Responder en cada apartado con una tabla con el siguiente formato:

nombre campo1	**n.campo2**	**…**	**…**	**total**	**alineamiento**
desplaz. campo1	d.campo2	…	…	tamaño	requisitos

3. **Unidad de control** (0.3 puntos). Un computador usa el formato vertical de codificación de instrucciones para parte de las señales de control y el formato horizontal para k señales de control. El formato vertical posee n campos codificados de m bits cada uno. ¿Cuál es el máximo número de señales de control que pueden usarse en este computador, teniendo en cuenta que cada campo codificado indica una señal que estará activa?

4. **Unidad de control** (0.2 puntos). Un procesador con una unidad de control microprogramada tiene una memoria de control de 340 palabras de 16 bits, de las que 180 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. **Entrada/Salida** (0.5 puntos). Un computador con 13 líneas de direcciones tiene una memoria de M palabras y utiliza una E/S localizada al final del mapa de memoria (no independiente). Si se supone que cada uno de los periféricos que puede conectarse ocupa 4 direcciones y que el número máximo de periféricos de estas características que se conecta es de 210. ¿Cuál es el tamaño de la memoria del computador? Dibuje un mapa del espacio de direccionamiento indicando en hexadecimal la primera y la última dirección de memoria y la primera y la última dirección de E/S.

6. **Diseño del sistema de memoria** (0.5 puntos). Diseñe un sistema de memoria para una CPU de 16 bits a partir de módulos SRAM 4Kx8 y ROM 4Kx4. La memoria ROM debe ocupar las direcciones 0x0000 a 0x1FFF y la SRAM 0xC000 a 0xEFFF.

7. **Memoria cache** (0.5 puntos). Los parámetros que definen la memoria de un computador son los siguientes:

   • Tamaño de la memoria principal: 1M palabras.
   • Tamaño de la memoria cache: 8K palabras.
   • Tamaño de bloque: 32 palabras.

   Determinar el tamaño de los distintos campos de una dirección en las siguientes condiciones:

   a) Correspondencia directa.

   b) Correspondencia completamente asociativa.

   c) Correspondencia asociativa por conjuntos con 4 bloques por conjunto.

   d) Correspondencia por sectores con 4 bloques por sector.

2º Grado Informática
Estructura de Computadores
28 Enero 2013

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. ¿Cuál de los siguientes es el orden correcto en el ciclo de compilación de un programa en lenguaje C? (el fichero sin extensión es un ejecutable):

a. fich.c -> fich.o -> fich.s -> fich
b. fich -> fich.s -> fich.o -> fich.c
c. fich.c -> fich.s -> fich -> fich.o
d. fich.c -> fich.s -> fich.o -> fich

2. ¿Qué hace gcc -O?

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

3. ¿Qué modificador (switch) de gcc hace falta para compilar .c->.s (de fuente C a fuente ASM)?

a. Eso no se puede hacer con gcc
b. gcc -c
c. gcc -s
d. gcc -S

4. ¿Qué modificador (switch) de ld hace falta para enlazar una aplicación de 32bits en un sistema de 64bits en el que se ha instalado también el compilador de 32bits?

a. -m32
b. -32
c. -m elf_i386
d. No hace falta modificador, ld lo deduce del tipo de objeto a enlazar

5. Compilar .c->exe (de fuente C a ejecutable) usando sólo as y ld, sin gcc...

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

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

a. En lenguaje C, al llamar a una subrutina o función se introducen los parámetros en la pila y después se realiza una llamada a la subrutina
b. 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
c. 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

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

---

7. El primer parámetro de printf:

a. es un entero
b. es un número en coma flotante
c. es un puntero
d. puede ser de cualquier tipo

---

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

a. NOT
b. AND
c. OR
d. XOR

---

9. 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

---

10. 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. la salida de la función se fuerza a que esté en la variable result

---

11. En una bomba como las estudiadas en la práctica 4, 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

---

12. En una bomba como las estudiadas en la práctica 4, del tipo…

```
0x08048705 <main+149>: call   0x80484c4
                                <gettimeofday>
...
0x08048718 <main+168>: cmp    $0x5,%eax
0x0804871b <main+171>: jle 0x8048722 <main+178>
0x0804871d <main+173>: call   0x8048604 <boom>
0x08048722 <main+178>: ...
```

ejecutada paso a paso con el depurador ddd, interesaría...

a. ejecutar hasta jle, ajustar %eax a 6, y continuar ejecutando paso a paso
b. ejecutar hasta jle, ajustar %eax a 4, y continuar ejecutando paso a paso
c. cambiar jle por jmp usando ddd o un editor hex, salvar el programa y reiniciar la depuración con el nuevo ejecutable
d. Ninguna de las opciones anteriores es de interés (bien porque no se pueda hacer eso o porque no sirva para evitar la bomba)

---

13. En una bomba como las estudiadas en la práctica 4, 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

---

14. Respecto a las bombas estudiadas en la práctica 4, ¿en cuál de los siguientes tipos de bomba sería más difícil descubrir la contraseña? Se distingue entre strings definidos en el código fuente de la bomba, y strings solicitados al usuario mediante scanf(). Por "cifrar" podemos entender la cifra del César, por ejemplo. "Invertir" es darle la vuelta al string de manera que la primera letra se convierta en la última y viceversa.

a. 1 string del usuario se cifra, y se compara con el string del fuente
b. 2 strings del fuente se invierten, se concatenan, se cifra el resultado, y se compara con el string del usuario
c. 2 strings del fuente se concatenan, se invierte el resultado, se cifra, y se compara con el string del usuario
d. Las 2 (o 3) opciones más difíciles son de la misma dificultad, así que no se puede marcar ninguna como la más difícil

---

15. Respecto a las bombas estudiadas en la práctica 4, ¿en cuál de los siguientes tipos de bomba sería más difícil descubrir la(s) contraseña(s)? Se distingue entre enteros definidos en el código fuente de la bomba, y enteros solicitados al usuario mediante scanf(). Por "procesar" podemos entender calcular el n-ésimo elemento de la serie de Fibonacci, por ejemplo.

a. 1 entero del fuente se procesa, y se compara con el entero del usuario
b. 2 enteros del usuario se suman, se procesa la suma, y se compara con el entero del fuente
c. 2 enteros del usuario se procesan, se suman los resultados, y se compara con el entero del fuente
d. Las 2 (o 3) opciones más difíciles son de la misma dificultad, así que no se puede marcar ninguna como la más difícil

---

16. Respecto al procesador que denominamos ssrDLX (procesador DLX sin segmentar), ¿cuál de las siguientes afirmaciones es correcta?

a. Todas las instrucciones deben realizar las cuatro etapas del cauce
b. Las etapas ID y MEM duran un 80% del tiempo de ciclo de reloj
c. Una operación de suma de enteros (operandos y resultado en el banco de registros) necesitará 3.8 ciclos en el procesador sin segmentar
d. ld f2,a en el procesador ssrDLX requiere 4.6 ciclos

---

17. Dado un programa en un procesador DLX, se puede aplicar la técnica que recibe el nombre de desenrollado de bucle, la cual repercute en:

a. realizar los cálculos de varias iteraciones en diferentes subrutinas
b. reducir el número de instrucciones de salto que se tienen que ejecutar
c. aumentar el número de instrucciones en el código para aumentar la probabilidad de que existan riesgos de control
d. reorganización de las instrucciones para reducir el efecto de las dependencias entre ellas

¿Cuál de las anteriores afirmaciones es cierta?

---

18. La directiva .text en el simulador DLX con la dirección 256, (.text 256), indica en un programa que:

a. la variable text tiene el valor 256
b. La primera instrucción del programa se ubicará en la posición 0x100
c. Existe una etiqueta de salto denominada .text en la posición 256
d. Todas las anteriores afirmación son incorrectas

---

19. ¿Cómo se almacenaría como palabra de 32 bits el número -128 en un sistema que utilice el criterio del extremo menor (“little endian")?

a. posición 0: FF pos.1:FF pos.2: FF pos.3: 00
b. 0:00 1:FF 2:FF 3:FF
c. 0:00 1:01 2: 00 3:80
d. Ninguna de las anteriores

---

20. En un procesador de la familia 80x86 las posiciones de memoria que representan una variable entera contiene los bytes: F0 FF FF FF. ¿Cuánto vale dicha variable?

a. -16
b. 4043309055
c. 16
d. 4294967280

2º Grado Informática
Estructura de Computadores
28 Enero 2013

Examen Test (3.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.
Cada respuesta vale 3/30 si es correcta, 0 si está en blanco o claramente tachada, -1/30 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	21	22	23	24	25	26	27	28	29	30
d	d	b	b	c	d	a	a	d	b	d	d	a	a	c	a	c	a	b	b	c	b	d	d	c	c	b	b	d	a

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, -1.33…/20 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
d	a	d	c	a	c	c	d	d	c	b	c	d	a	c	d	b	b	d	a

Examen de Problemas (3.0p)

1. **Bucles while** (0.5 puntos).

A.

```
int fun_a(unsigned x) {
  int val = 0;
  while (x) {
    val ^= x;
    x >>= 1;
  }
  return val & 0x1;
}
```

B. Calcula la paridad de x (devuelve 1 si hay nº impar de 1s, 0 si par)

2. **Representación y acceso a estructuras** (0.5 puntos).

A. `struct P1 { int i; char c; int j; char d; };`

i	c	j	d	total	alineamiento
0	4	8	12	16	4

B. `struct P2 { int i; char c; char d; int j; };`

i	c	d	j	total	alineamiento
0	4	5	8	12	4

C. `struct P3 { short w[3]; char c[3]; };`

w	c	total	alineamiento
0	6	10	2

D. `struct P4 { short w[3]; char *c[3]; };`

w	c	total	alineamiento
0	8	20	4

E. `struct P5 { struct P1 a[2]; struct P2 *p };`

a	p	total	alineamiento
0	32	36	4

3. **Unidad de control** (0.3 puntos).
La UC puede controlar simultáneamente **n+k** señales de control,
aunque la unidad de procesamiento tiene un total de $\mathbf{n*2^m + k}$ señales de control
(si todos los 2^m códigos de cada uno de los n campos codificados son válidos)
Ver dibujo adjunto

4. **Unidad de control** (0.2 puntos).
Ahorro ninguno, se gastan 160 bits más.
340x16 < (340x8 + 180x16) = (170 + 180) x 16 = 350 x 16
340x16 < 350x16, se gastan 10x16 bits más
Ver dibujo adjunto

5. **Entrada/Salida** (0.5 puntos).
210perif x 4puertos = 840 puertos = 0x348
Si empiezan desde el final, 0x2000 - 0x0348 = 0x1cb8
E/S ocuparía 0x1cb8 … 0x1fff
Si pregunta M máxima, 0x0000 … 0x1cb7 = 8K – 840 = 7352 palabras
Ver dibujo adjunto

6. **Diseño del sistema de memoria** (0.5 puntos).
ROM 0x0000-0x1fff = 2^{13}=8K pal = 8Kx16. En módulos de 4Kx4 x (2x4) -> 8 módulos
SRAM 0xC000-0xefff = 3x2^{12}=12K pal = 12Kx16. En m.de 4Kx8 x (3x2) -> 6 módulos
Todos los módulos conectados a A11…A0, decodificación con A15…A12 (1^{er} dígito hex)
Ver dibujo adjunto

7. **Memoria cache** (0.5 puntos).

A.	Directa:	20 = 7 + 8 + 5.	etiqueta+marco+palabra.	2^{13}pal/cach / 2^5pal/bloq = 2^8bloq/cache
B.	Asociativa:	20 = 15 + 5.	etiqueta +palabra.	
C.	Asociativa Conjuntos 4vías:	20 = 9 + 6 + 5.	etiqueta+conjnt+palabra.	2^8bloq/cach / 2^2blq/conj = 2^6conj/cache
D.	Sectores de 4palabras:	20 =13+ 2 + 5.	etiqueta+bloque+palabra.	4 = 2^2bloq/sector

Ver dibujo adjunto