COGNOMS:

NOM: DNI/NIE:

IMPORTANTE leer atentamente antes de empezar el examen: Escriba los apellidos, el nombre y el DNI/NIE antes de empezar el examen. Escriba un solo carácter por recuadro, en mayúsculas y lo más claramente posible. Es importante que no haya tachones ni borrones y que cada carácter quede enmarcado dentro de su recuadro sin llegar a tocar los bordes. Use un único cuadro en blanco para separar los apellidos y nombres compuestos si es el caso. No escriba fuera de los recuadros, todo lo que haya fuera de ellos es ignorado. La identificación del alumno se realiza de forma automática, no seguir correctamente estas instrucciones puede comportar no tener nota.

Problema 1. (5 puntos)

Un programa (P) se ejecuta en un computador cuya CPU (C1) funciona a una frecuencia de 2,4 GHz. El programa (P) contiene 50000 instrucciones estáticas, realiza $2x10^9$ operaciones de punto flotante y se ejecuta en 10 segundos en C1. La siguiente tabla muestra la distribución de instrucciones dinámicas y estáticas para el programa (P) junto con el CPI medio de cada tipo de instrucción.

	punto flotante	**enteras**	**memoria**
% instrucciones estáticas	10%	40%	50%
% instrucciones dinámicas	40%	35%	25%
CPI	4,0	2,0	6,8

a) **Calcula** el rendimiento en MFLOPS del programa (P).

b) **Calcula** el CPI del programa (P).

c) **Calcula** el numero de instrucciones dinámicas (N) del programa P.

Se ha decidido que una rutina (R), que representa el 80% del tiempo de (P), sea programada en ensamblador.

d) **Calcula** la ganancia en tiempo de ejecución (speedup) de la rutina (R) que se debería obtener respecto al compilador para que el programa (P) se ejecute 3 veces más rápido.

En nuestro computador hemos cambiado la CPU (C1) por un nuevo modelo (C2) que soporta instrucciones SIMD y tiene una cache de datos más grande, aunque funciona a menor frecuencia (2 GHz). Una vez recompilado el programa (P), para hacer uso de las nuevas instrucciones SIMD, observamos que el número de instrucciones dinámicas de punto flotante se ha reducido a la mitad (el resto no ha cambiado). Además el CPI de las instrucciones de punto flotante ha aumentado en un 25% y el de las instrucciones de memoria es de 2,8 ciclos/instrucción (el CPI de las enteras es el mismo).

e) **Calcula** el **CPI** del programa (P) con la nueva CPU (C2).

f) **Calcula** el tiempo de ejecución del programa (P) con la nueva CPU (C2).

La CPU (C1), tiene una capacidad efectiva equivalente de 8 nF (nanofaradios), y una corriente de fugas de 12 A y funciona a un voltaje de 1,25 V.

g) **Calcula** la potencia media debida a fugas, la debida a conmutación y la potencia total disipada por la CPU (C1).

En la CPU C2 el voltaje y la corriente de fugas son los mismos que en C1, pero debido al hardware adicional para ejecutar las instrucciones SIMD la capacidad efectiva equivalente es de 12 nF. Una de las métricas que se usa para comparar la eficiencia energética de las CPUs es el producto de la energía consumida para ejecutar un programa por el tiempo de ejecución de este. Esta métrica se conoce como EDP (Energy Delay Product)

h) **Calcula** la ganancia en EDP de la CPU C2 respecto C1 al ejecutar el programa P.

COGNOMS:

NOM: DNI/NIE:

Problema 2. (5 puntos)

Dado el siguiente código escrito en C, que compilamos para un sistema linux de 32 bits:

```
typedef struct {
  char a[3];
  char b;
  short c[3];
  } s1;
```

```
typedef struct {
  s1 u[10];
  char v;
  short w[2];
  } s2;
```

a) **Dibuja** cómo quedarían almacenadas en memoria las estructuras **s1** y **s2**, indicando claramente los desplazamientos respecto al inicio, el tamaño de todos los campos y el tamaño de los structs.

b) **Escribe** UNA ÚNICA INSTRUCCIÓN que permita mover **x.u[5].c[1]** al registro **%ax**, siendo **x** una variable de tipo **s2** cuya dirección está almacenada en el registro **%ecx**.
Indica claramente la expresión aritmética utilizada para el cálculo de la dirección.

Dado el siguiente código escrito en C, que compilamos para un sistema linux de 32 bits:

```
int examen(char b[2][3], char c, short d) {
  char y[2][3];
  char z;
  short w;
  int x;
  . . .
  x=examen(y,z,w);
  . . .
}
```

c) **Dibuja** el bloque de activación de la rutina examen, indicando claramente los desplazamientos respecto a **%ebp** y el tamaño de todos los campos.

d) **Traduce** a ensamblador x86 la instrucción `x=examen(y,z,w);` que se encuentra en el interior de la subrutina, usando el mínimo número de instrucciones.