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. (3,5 puntos)

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

```
int examen(char *e[], char d[8], char c, char b, short a){
  char i[10];
  short j;
  int k;
  int *p;
  ...
}
```

a) **Dibuja** el bloque de activación de la función **examen**, indicando claramente el **tamaño de cada campo** y los **desplazamientos** relativos al registro EBP necesarios para acceder a los parámetros y a las variables locales.

b) **Escribe** en ensamblador del x86 el código de inicio de la subrutina **examen** (sabemos que **la subrutina examen utiliza todos los registros)** desde el principio de esta hasta la primera sentencia en C de la rutina (incluida) que es :

```
k = 0;
```

c) **Escribe** en ensamblador del x86 el código hasta el final de la subrutina **examen** (teniendo en cuenta lo hecho en el apartado b) desde la última sentencia en C de la rutina (incluida) que es:

```
return (*p);
```

d) Indica el rango de los operandos naturales y enteros que pueden ser almacenados en el registro %bx

COGNOMS:

NOM: DNI/NIE:

Problema 2. (3,5 puntos)

a) Dibuja los cronogramas que representan las operaciones a realizar en CPU, memoria cache y memoria principal en caso de acierto y fallo, tanto de lectura como de escritura, para una sistema de memoria *write through* y *write NO allocate* con un *tsa*=1 ciclo, Tiempo de leer/escribir un bloque de MP=7 ciclos, y Tiempo de escribir una palabra en MP=4 ciclos.

Se quiere diseñar una memoria cache de datos con políticas de escritura *write through* y *write NO allocate*:

Se han obtenido por simulación las siguientes medidas para un determinado programa:

- porcentaje de escrituras (sobre el total de accesos): 20%
- tasa de aciertos: 0,9

La memoria cache es de mapeo directo y se leen etiquetas y datos en paralelo. En caso de fallo de lectura, el bloque de MP se escribe en la MC y posteriormente el dato se envía a la CPU desde la MC. El tiempo de acceso (Tsa) a memoria cache (MC) es de 10 ns tanto para lectura como escritura. El tiempo de acceso a memoria principal (MP) para escribir una palabra es de 80 ns. Para leer o escribir un bloque en la MP se emplean 110 ns.

b) **Calcula** el tiempo empleado en realizar 1000 accesos consecutivos.

COGNOMS:

NOM: DNI/NIE:

Problema 3. (3 puntos)

En un procesador con direcciones de 32 bits, el camino crítico, y por tanto el tiempo de ciclo, está limitado por la memoria cache de datos. El tiempo y la energía, según el uso que se realice de los componentes de la memoria (en caso de que se usen), se desglosa de la siguiente forma:

Componente	Tiempo	Energía
Memoria de etiquetas	0,30 ns	4 nJ/acceso
Comparación de etiquetas y (en caso necesario) **selección de vía**	0,20 ns	0,50 nJ/acceso
Memoria de datos y selección de byte de la linea	0,50 ns	24 nJ/acceso
Mux de vía de datos: selecciona el dato de la vía correspondiente (cuando sea necesario)	0,10 ns	1 nJ/acceso
Registro de desacoplo (cuando sea necesario)	0,05 ns	0,05 nJ/acceso

Queremos analizar 2 configuraciones para la cache de datos (C1 y C2), ambas con 16KB de capacidad y lineas de 8 bytes:

- C1: Cache de mapeo directo con acceso PARALELO a etiquetas y datos.
- C2: Cache asociativa por conjuntos de dos vías, con acceso SECUENCIAL SEGMENTADO en 2 etapas (el tiempo de servicio en caso de acierto de la cache son 2 ciclos de procesador).

a) **Calcula** el tiempo de ciclo de la cache de datos y el tiempo total de un acceso para las dos configuraciones del procesador con las caches C1 y C2, usando la distribución de los componentes por etapas más eficiente desde el punto de vista del tiempo de ciclo en la configuración C2.

b) **Calcula** la frecuencia de reloj para las diferentes versiones del procesador con las caches C1 y C2.

Un programa P que ejecuta $2{,}5x10^9$ instrucciones tiene un 50% de instrucciones aritméticas, un 20% de instrucciones de salto y un 30% de instrucciones de acceso a memoria (Load/Store). Las instrucciones aritméticas tardan 4 ciclos, las de salto 3 y las de memoria 5 ciclos + los ciclos del acceso a la cache. Sabemos que el programa P tiene un 10% de fallos con la cache de datos C1 y un 6% con la C2. Además, el tiempo de penalización medio por fallo en ambos casos es de 60 ciclos.

c) **Calcula** el CPI del programa P para los procesadores con C1 y C2.

d) Calcula la potencia media consumida por la cache C1 ejecutando el programa P suponiendo que en un acceso con fallo la cache consume lo mismo que en un acceso con acierto (aunque no tarden el mismo tiempo).

Se hace una implementación multibanco (con 4 bancos) de la cache C2 para reducir el consumo de un acceso que llamaremos C2.b. Para ello dividimos la memoria de etiquetas en 4 bancos donde cada banco consume 1nJ por acceso y la memoria de datos en otros 4 bancos donde cada banco consume 6nJ por acceso.

e) **Calcula** la energía (dinámica) consumida por un acceso a la cache C2.b (con la implementación multibanco).