COGNOMS:

NOM: DNI:

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 mayusculas 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. (2,5 puntos)

Dado el siguiente código escrito en C y compilado para x86 linux 32 bits:

```
typedef struct {
   char c;
   short s[4];
   char c2;
   int i2;
} sx;
int examen(sx *p1, sx s1, short sh){
   int i;
   char ch;
   short aux;
   int *x;
   . . .
}
```

a) **Dibuja** como quedaría almacenada la estructura **sx,** indicando claramente los desplazamientos respecto del inicio y el tamaño de todos los campos

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

c) **Traduce** a ensamblador x86 la sentencia **return(*x + s1.i2);** suponiendo que esta dentro la función examen:

d) **Traduce** a ensamblador x86 la sentencia **(*p1).s[2] = aux;** suponiendo que esta dentro la función examen:

COGNOMS:

NOM: DNI:

Problema 2. (3,5 puntos)

Se dispone de un computador C con una memoria RAM de 64 MB (que corresponde al espacio físico direccionable). El procesador genera direcciones lógicas de 32 bits y está conectado a una memoria caché de datos de primer nivel L1D con política de escritura copy back-write allocate. La memoria cache L1D tiene un tamaño de 3 KB, es 3-asociativa y el tamaño de línea es de 64 bytes. El sistema dispone también de un TLB de 4 entradas, completamente asociativo y con política de reemplazo LRU. Se accede simultáneamente a TLB y cache. El tamaño de página del sistema operativo es de 4KB.

a) Un proceso P del sistema lanza un acceso de lectura que se mapea en la dirección física 0x2345678. Este acceso produce un acierto de lectura en L1D. **Indica** claramente cuántos bits tiene cada uno de los campos tag, conjunto y byte con que se accede a la cache, y **escribe EN HEXADECIMAL** el tag, el conjunto y el byte dentro de la línea al que se accederá en memoria cache L1D.

b) **Calcula** el tamaño en **MEGABYTES** de la tabla de páginas del proceso P, teniendo en cuenta que cada entrada de la tabla, además de los bits necesarios para codificar la página física, tiene un bit de presencia y un bit de página modificada.

El proceso P contiene el siguiente código escrito en ensamblador del x86:

```
      movl $0, %ebx
      movl $0, %esi
for:
      cmpl $256*1024, %esi
      jge end

  (a) movl 0x2000(%ebx, %esi, 4), %eax
  (b) movb %bh, 0x2000(%ebx, %esi, 4)
  (c) addl %eax, 0x3000(%ebx, %esi, 4)
  (d) movw %ax, 0x5000(%ebx, %esi, 4)

      addl $1, %esi
      jmp for
end:
```

c) **Calcula** la **tasa de fallos** de la cache para el fragmento de código anterior suponiendo que inicialmente está vacía.

d) Para cada uno de los accesos indicados (etiquetas a, b, c, d), **indica** a qué página de la memoria virtual se accede en cada una de las siguientes iteraciones del bucle.

iteración	0	1*256	2*256	3*256	4*256	5*256	6*256	7*256	8*256	9*256
a										
b										
c										
d										

e) **Calcula** el **número de fallos** de TLB en el fragmento de código.

COGNOMS:

NOM: DNI:

Problema 3. (4 puntos)

Tenemos un procesador (que llamaremos Pmp) sin memoria cache, todos los accesos se realizan directamente sobre memoria principal. En este procesador hemos ejecutado una aplicación A (que usaremos a lo largo de todo el problema) y hemos obtenido los siguientes datos: tiempo de ejecución 18 s, $2x10^9$ instrucciones ejecutadas, $3x10^9$ accesos a memoria (incluye tanto los accesos a datos como los accesos a instrucciones). Sabemos además que Pmp funciona a una frecuencia de 2 GHz. Para simplificar el problema supondremos que todos los accesos son lecturas.

a) **Calcula** el CPI del procesador Pmp al ejecutar la aplicación A (CPIpmp).

Con un simulador hemos simulado una versión ideal (que llamaremos Pideal) del procesador, en donde todos los accesos a memoria tardan un ciclo. En el Pideal hemos obtenido un CPI (CPIideal) de 3 ciclos/instrucción.

b) **Calcula** el tiempo de penalización de los accesos a memoria (en ciclos) del procesador Pmp respecto el Pideal.

Para mejorar el rendimiento respecto el procesador Pmp, añadimos una cache 2-asociativa al procesador (que denominaremos procesador P2a). Para poder mantener la frecuencia de 2 GHz, los accesos a la cache 2-asociativa tardan 2 ciclos, es decir que en caso de acierto tenemos una penalización de 1 ciclo. La penalización media en caso de fallo es de 17 ciclos. La tasa de fallos de la cache 2-asociativa es del 5%.

c) **Calcula** el tiempo de ejecución de la aplicación A en el procesador P2a

Para reducir la penalización de accesos a la cache, uno de los ingenieros ha sugerido el uso de un predictor de vía. El predictor sugerido tendría una tasa de aciertos en la predicción de vía del 80% y tendría un impacto negligible tanto en área como en la frecuencia y consumo del procesador. Al procesador con cache 2-asociativa y predictor de vía lo denominaremos procesador Ppv. En caso de que el predictor acierte la vía no hay penalización respecto el procesador ideal. Si hay fallo de predictor pero acierto de cache se incurre en un ciclo de penalización (tal como ocurría con la cache 2-asociativa) ya que se necesita un ciclo adicional para acceder a la otra vía. Finalmente, si es fallo de predictor y también de cache, la penalización es de 17 ciclos (también la misma que la cache 2-asociativa).

d) **Calcula** el CPI del procesador Ppv (CPIppv).

Queremos saber la potencia media debida a conmutación de la jerarquía de memoria del procesador P. Ignoraremos por tanto la potencia disipada por fugas así como la potencia de conmutación del procesador, y también la del predictor que ya hemos comentado que tiene un impacto despreciable. Sabemos que cada vez que se accede una vía de la cache (etiquetas + datos) se consumen 5 nJ (nanojoules) y cada vez que hay un fallo de cache se consumen 30 nJ adicionales.

e) **Calcula** la energía (de conmutación) consumida por la jerarquía de memoria del procesador Ppv al ejecutar la aplicación A

f) **Calcula** la potencia media (de conmutación) consumida por la jerarquía de memoria del procesador Ppv al ejecutar la aplicación A