Cognoms: . . . . Nom: . . . .

2on Control Arquitectura de Computadors **Curs 2013-2014 Q1**

Problema 1. (3 puntos)

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

```
typedef struct {
  char a;
  short int b;
  char c;
} s1;
```

```
typedef struct {
  char e;
  s1 f[10];
  int g;
  } s2;
```

```
int F(s1 *uno, char c);
int examen(int *x, char d[10], s2 dos){
  int i;
  char h[10];
  int j;
  ...
}
```

a) **Dibuja** como 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) **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.

c) **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 :

```
i = 0;
```

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

```
return (j);
```

e) **Traduce** la siguiente sentencia a ensamblador del x86, suponiendo que se encuentra dentro de la rutina **examen**:

```
*x = F(&(dos.f[0]),d[4]);
```

Cognoms: . . . . . Nom: . . . . . .

2on Control Arquitectura de Computadors **Curs 2013-2014 Q1**

Problema 2. (4 puntos)

Tenemos un microcontrolador con un camino de datos de 8 bits, por lo que todos los accesos a memoria son a byte. Este microcontrolador tiene un mecanismo de memoria virtual con páginas de 4 Kbytes, direcciones lógicas de 20 bits y direcciones físicas de 16 bits. Por simplicidad ignoraremos los accesos a instrucciones. Para acelerar la traducción tiene un TLB de datos totalmente asociativo de cuatro entradas con algoritmo de reemplazo LRU. También dispone de una cache de datos que se accede en serie con el TLB (con direcciones físicas). Esta cache es de mapeo directo, tiene un tamaño de 256 bytes y bloques (lineas) de 16 bytes. La cache de datos sigue una política de escritura inmediata (Write Through) sin asignación en caso de fallo (Write NO Allocate).

a) **Dibuja** una dirección lógica con los campos de bits relevantes para la traducción de direcciones, indicando claramente el nombre y tamaño de cada uno de ellos. **Dibuja** también una dirección física con los campos de bits relevantes para el acceso a cache, indicando claramente el nombre y tamaño de cada uno de ellos.

b) **Describe** el funcionamiento de las políticas de escritura de lacache en caso de **escritura-acierto** y de **escritura-fallo**:

El tiempo de ciclo del procesador es de 1 ns, un acceso a TLB necesita 0,6 ns y uno a cache 0,7 ns (la mayor parte del cual se dedica al acceso paralelo a etiquetas y datos).

c) **¿Cuantos** ciclos tardaría un acceso que acierte tanto en TLB como en cache con la implementación de traducción en serie descrita del conjunto TLB-Cache? ¿Y con una implementación de traducción en paralelo?

d) Si pudiésemos implementar una cache más grande (manteniendo el resto de parámetros) ¿Cual podría ser el tamaño máximo de cache que podríamos implementar con la traducción paralela? Justifícalo brevemente.

Sabemos que tanto el TLB como la cache están vacios y que las 6 primeras entradas de la tabla de páginas tienen el siguiente contenido.

VPN	PPN	P	M
00	A	1	0
01	B	1	0
02	0	1	0
03	5	1	0
04	4	1	0
05	3	1	0

e) **Rellena** la siguiente tabla a partir de la secuencia de referencias a memoria dada:

@lógica	L/E	VPN	Desp	PPN	@física	TLB	BMP	TAG	BMC	Cache	LMP	EMP
00000	L											
01111	L											
02212	L											
00003	E											
03324	L											
05525	E											
00016	L											
01117	E											
05528	L											
02219	L											
0000A	L											

@lógica: Dirección lógica (en hexadecimal) generada por el procesador.
L/E: El acceso es lectura (L) o escritura (E).
VPN: Virtual Page Number (página lógica) (en hexadecimal).
Desp: desplazamiento dentro de la página (en hexadecimal).
PPN: Physical Page Number (página física) (en hexadecimal).
@física: Dirección física (en hexadecimal)
TLB: Acierto de TLB (A) o fallo de TLB (F).
BMP: Bloque (linea) de memoria principal que se accede (en hexadecimal).
TAG: Etiqueta del acceso a cache (en hexadecimal).
BMC: Bloque (linea) de cache que se accede (en hexadecimal).
Cache: Acierto de Cache (A) o fallo de Cache (F).
LMP: Numero de bytes leídos de memoria principal (vacío si no se lee).
EMP: Indica numero de bytes escritos en memoria principal (vacío si no se escribe).

Cognoms: . . . . . Nom: . . . . . .

2on Control Arquitectura de Computadors **Curs 2013-2014 Q1**

Problema 3. (3 puntos)

Se quiere diseñar la memoria cache de datos de primer nivel para un procesador de muy bajo consumo de tipo Load/ Store que consume 0,05 W. Se barajan dos alternativas:

- (1) write through y write NO allocate.
- (2) copy back y write allocate

Se dispone de un programa X que ejecuta 8 x 10^9 instrucciones dinámicas, de las cuales el 25% son instrucciones de acceso a memoria (una instrucción puede hace un acceso a memoria como máximo), el 30% son instrucciones que realizan una operación en coma flotante, el 35% son instrucciones aritmético lógicas y el 10% restante son instrucciones de salto. Se han obtenido por simulación las siguientes medidas para el programa X:

- porcentaje de escrituras (sobre el total de accesos): 15%
- porcentaje de bloques modificados: 10%
- tasa de aciertos caso (1): 0,9
- tasa de aciertos caso (2): 0,8
- Tiempo de ejecución caso (1): 100 segundos
- Tiempo de ejecución caso (2): 120 segundos

En ambos casos, la memoria cache es de mapeo directo y se leen etiquetas y datos en paralelo. En caso de fallo, el bloque de MP se escribe en la MC y posteriormente el dato se envía a la CPU desde la MC. La escritura es segmentada. El tiempo de acceso (Tsa) a memoria cache (MC) es de 10 ns. 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 100 ns. La energía consumida al realizar un acceso a MC (memoria de datos) es de 3 nJ si es un acceso a bloque y 1 nJ si es a palabra (byte, word o longword). El consumo de la memoria de etiquetas de la MC es despreciable. La energía consumida al realizar un acceso a MP, tanto si es a palabra como si es a bloque, es de 5 nJ. La corriente de fugas de toda la jerarquía de memoria también es despreciable.

a) **Calculad** el tiempo empleado en realizar 1000 accesos consecutivos en el caso (1)..

b) **Calculad** el tiempo empleado en realizar 1000 accesos consecutivos en el caso (2)..

c) **Calculad** la energía consumida por la jerarquía de memoria al realizar 1000 accesos para el caso (1).

d) **Calculad** la eficiencia energética del sistema procesador-memoria (MFLOPS/w) al ejecutar TODO EL PROGRAMA en el caso (1)..