

2CDes1

Ejercicio 1

Considere para un procesador de arquitectura LEGv8 (Address bus de 64 bits) una CACHE L1 exclusiva para INSTRUCCIONES de 2K bytes de datos, de correspondencia DIRECTA, de 2 palabras (cada palabra es 1 instrucción de 32 bits) por bloque. La siguiente lista corresponde a DIRECCIONES de acceso a la memoria de instrucciones (PC) para la ejecución de un lazo (solo se muestran 32 bits de los 64 bits para abreviar):

- 1> 0x00...00 00 02 F4 <loop: asm_instruction>
- 2> 0x00...00 00 02 F8
- 3> 0x00...00 00 02 FC
- 4> 0x00...00 00 1A FC
- 5> 0x00...00 00 1B 00 <cbz loop>

a) Calcular el formato de memoria principal para la CACHE dada y completar la siguiente tabla en formato HEXADECIMAL para 2 iteraciones del loop:

b) Suponiendo que el procesador opera con el pipeline en régimen a 1GHz, el tiempo de acceso a CACHE es de 1 ciclo y el Miss penalty es de 50 ciclos, calcular el tiempo total de ejecución (en seg) para las condiciones del punto a).

Tiempo de ejecución: 0,00000021 seg.
 21 nseg.

c) Ahora considere que puede cambiar el tamaño del bloque y la CACHE pasa a ser asociativa por conjunto de N-vías. Determine el tamaño del bloque y el número de vías ÓPTIMOS para minimizar la ocurrencia de MISS (se considera el número óptimo aquel arriba del cual no se generan beneficios adicionales).

Tamaño de bloque: bytes.

Número de vías: vías.

d) Calcular el tiempo de ejecución de 2 iteraciones del lazo para las condiciones del punto c)

Tiempo total de ejecución: seg.

Acc.	Hit / Miss	Tag (54) bits	Index (9) bits	Word (4) bits
1>	Miss	0x0	0x17A	0x0
2>	Miss	0x0	0x17C	0x0
3>	Miss	0x0	0x17E	0x0
4>	Hit	0x6	0x17E	0x0
5>	Miss	0x6	0x180	0x0
1>	Hit	0x6	0x17A	0x0
2>	Hit	0x6	0x17C	0x0
3>	Hit	0x6	0x17E	0x0
4>	Hit	0x6	0x17E	0x0
5>	Hit	0x6	0x180	0x0

Ejercicio 2

Dado un procesador de arquitectura LEGv8 2-issue, que predice los saltos perfectamente (de modo que los hazard de control son manejados por hardware), con una modificación que permite ejecutar dos instrucciones aritméticas/lógicas juntas (es decir, una instrucción

Parcial 2 - Arquitectura de computadoras 2023

- a) Completar la tabla en hexadecimal en las posiciones de la PHT del predictor global que se modifican en dos iteraciones del salto de la posición 0xD, considerando que está completamente inicializada en cero y que $X0=10$ y el $GR=0x34$ antes de comenzar a ejecutarse primer instrucción del código.
- b) Considerando que inicialmente $X0=0$ y que existe un patrón repetitivo (cada dos iteraciones del código aparece un número par, y cada 4 aparece un número múltiplo de 4). ¿Qué cantidad de bits debería tener como **mínimo** el GR para almacenar el patrón del salto de la posición 0x5? ¿Y para el salto de la posición 0xA?

0x5> _____

0xA> _____

Ejercicio 5

Completar el siguiente kernel de openCL que multiplique una matriz cuadrada de $N \times N$ elementos por una constante alpha.

```
__kernel void parallelMMult(  
  const int N, const float alpha,  
  __global float *a_g) {  
  int i = N; _____  
  int j = N; _____  
  float tmp = 0.0f;  
  _____  
  _____  
  _____  
}
```

El kernel debe corresponderse con el siguiente código de host:

```
platform_list = cl.get_platforms()  
devices = platform_list[0].get_devices(device_type = cl.device_type.GPU)  
context = cl.Context(devices=devices)  
queue = cl.CommandQueue(context)  
a_np = np.arange(N*N).astype(np.float32)  
a_g = cl.Buffer(context, cl.mem_flags.READ_WRITE |  
  cl.mem_flags.COPY_HOST_PTR, hostbuf = a_np)  
program = cl.Program(context, KernelSource).build()  
kernel = program.parallelMMult  
kernel.set_scalar_arg_dtypes([np.int32, np.float32, None])  
kernel.set_args(N, alpha, a_g)  
globalRange = (N,N)  
localRange = (N,N)  
ev = cl.enqueue_nd_range_kernel(queue, kernel, globalRange, localRange)  
cl.enqueue_copy(queue, a_np, a_g)
```