Parcial 1C2020
Organización de Computadoras

23/06/2020

1. Ejercicio 1

1.1. Question 1

Se proponen 3 mejoras para una nueva arquitectura con los siguientes speedups:

Speedup1 = 5
Speedup2 = 10
Speedup3 = 2

La fracción de uso de estas mejoras es de 25 %, 23 % y 30 % respectivamente. Las mejoras son mutuamente excluyentes (sólo una de ellas puede ser aplicada). Se requiere maximizar la performance: ¿cuál de ellas debe ser aplicada?

a. Ninguna de las otras opciones
b. Mejora 2
c. Mejora 3
d. Mejora 1

Solo se puede aplicar una mejora por vez,entonces

$$SU_1 = \frac{1}{1 - f_1 + \frac{f_1}{SU_1}} = \frac{1}{1 - 0,25 + \frac{0,25}{5}} = 1,25$$

$$SU_2 = \frac{1}{1 - f_2 + \frac{f_2}{SU_2}} = \frac{1}{1 - 0,23 + \frac{0,23}{10}} = 1,2610$$

$$SU_3 = \frac{1}{1 - f_3 + \frac{f_3}{SU_3}} = \frac{1}{1 - 0,3 + \frac{0,3}{2}} = 1,176$$

Si se quiere maximizar la performance, se quiere el SpeedUp máximo, entonces debe aplicarse la mejora 2.

1.2. Question 2

Indicar cuál de los siguientes grupos de instrucciones contiene una o más pseudo-instrucciones MIPS32:

Select one:

a. subu, lbu, sb, syscall
b. Ninguna de las otras opciones
c. lw, bnez, subu, sb
d. sll, srl, sub, sb

Estaba entre la a y la b.. elegí la b pero no sé.

1.3. Question 3

Seleccionar un bloque de código MIPS que implemente el comportamiento equivalente para evaluar el 'if' en el siguiente programa C:

```
static int unos[] = {1,1,1,1,1};

int main() {
    if (unos[1] == 0) {
        //if-block
    }
    //fallthrough
}
```

a. Todos los fragmento MIPS son correctos
b.

```
#..
_eval:  la t1, unos
        lw t1, 4(t1)
        bnez t1, _fallt
_true: #if-block
_fallt: #fallthrough
```

c.

```
#..
_eval:  la t0, unos
        lw t1, 1(t0)
        bnez t1, _fallt
_true: #if-block
_fallt: #fallthrough
```

d.

```
#..
_eval:  la t0, unos
        lb t1, 4(t0)
        bnez t1, _fallt
_true: #if-block
_fallt: #fallthrough
```

e. Ninguna de las opciones restantes

Opción b: unos es un array de enteros, cada entero ocupa 4 bytes. Para evaluar la condición del if hay que acceder a unos[1], como cada elemento de unos ocupa 4 bytes, entonces acceder a la posición 1 es acceder a la posición 4x1. Además, 4 bytes equivale a un word.

Y con lo mencionado quedan descartadas las opciones c y d. La c porque accede a 1(t1), o sea no estaría escalando con el tamaño de un entero; la opción d porque hace un load byte en vez de load word.

1.4. Question 4

Supongamos la siguiente declaración C:

extern int suma(int a, int b, int c, int d);

a. El caller (invocante de suma) va a grabar en el stack frame de suma() los cuatro registros 'a' usados en el pasaje de parámetros. Esto ocurrirá justo antes de invocar a esa función
b. Ninguna de las otras opciones
c. Los registros 'a' usados en el pasaje de parámetros serán grabados en el stack frame de suma() una vez invocada esa función

d. Se usan los registros a1-a4 para pasar los valores de los parámetros

---

Elegí la opción b.

Descarto opciones a y b porque en el stack frame de suma() no se graban los registros 'a' usados en el pasaje de parámetros. Los registros 'a' que se pasan por parámetros están guardados en la ABA del caller, en gral se guardan en algún registro temporal para no perderlos si la función es non-leaf. La opción d la hubiese considerado correcta si dijese a0-a3.

1.5. Question 5

Para el siguiente fragmento MIPS, la cantidad total de referencias a memoria realizada por el bloque comprendido entre el tag loop y la instrucción bnez es,

```
      .globl main
      .ent main
main:
      .set noreorder
      .cpload t9
      .set nomacro
       addiu sp, sp, -24
       sw fp, 20(sp)
       move fp, sp
      .cprestore 0

       la t0, aligned
       li t1, 100
      .align 20
loop:  lw t2, 0(t0)
       addu t0, t0, 4
       subu t1, t1, 1
       bnez t1, loop
```

a. 300
b. 200
c. 100
d. Ninguna de las otras opciones
e. 400

Cantidad total de referencias a memoria = Cantidad de referencias a datos + cantidad de referencias a instrucciones

=>1x100 + 4x100 = 500

Hay una referencia a memoria (0(t0)) que se ejecuta 100 veces (por el loop) y 4 instrucciones que también se ejecutan 100 veces.

2. Ejercicio 2

2.1. Question 1

Suponga que tiene el siguiente cache (2WSA de 8 conjuntos con líneas de 4 words) con los datos ya cargados. Suponiendo que el cache es write through-no allocate, seleccione la respuesta correcta.

	V	D	Tag	Data 0	Data 1	Data 2	Data 3
set 0	1	0	0x12	0x0A	0x1A	0x2A	0x3A
set 1	1	0	0x12	0x4B	0x5B	0x6B	0x7B
set 2	1	0	0x12	0x3C	0x2C	0x1C	0x0C
set 3	1	0	0x12	0x7D	0x6D	0x5D	0x4D
set 4	1	1	0x67	0x33	0x23	0x13	0x03
set 5	1	1	0x67	0x44	0x34	0x24	0x14
set 6	0	0	0x34	0x55	0x65	0x75	0x85
set 7	1	0	0x58	0x66	0x76	0x86	0x96

V	D	Tag	Data 0	Data 1	Data 2	Data 3
1	0	0x27	0x80	0x81	0x82	0x83
1	1	0x27	0xB4	0xB5	0xB6	0xB7
1	0	0x90	0xC3	0xC2	0xC1	0xC0
1	0	0x90	0xD3	0xD4	0xD5	0xD6
1	0	0x11	0x89	0x88	0x87	0x86
1	0	0xA0	0x92	0x93	0x94	0x95
1	1	0x37	0xF5	0xF6	0xF7	0xF8
1	1	0x21	0xA7	0xA8	0xA9	0xAA

1. Un acceso de lectura a la dirección 0x1A60 resulta en hit
2. Un acceso de escritura a la dirección 0x140 no resulta en un reemplazo de la vía marcada como LRU en el índice correspondiente
3. Ninguna es correcta
4. Un acceso de lectura a la dirección 0x310 resulta en hit
5. Un acceso de lectura a la dirección 0x910 resulta en miss

Líneas de 4 words ⇒ como cada word ocupa 4 bytes, habrá 16 bytes por línea/bloque ⇒ se necesitan 4 bits de offset.

8 conjuntos ⇒ 3 bits de indice

⇒ $32 - 3 - 4 = 25$ bits de tag

1. Lectura 0x1A60 = 0001 1010 0110 0000 ⇒ idx = 6, tag = 0011 0100 = 0x24 ⇒ MISS
2. Escritura 0x140 = 0001 0100 0000 ⇒ idx = 4, tag = 0x02 ⇒ MISS pero como es write no allocate no traigo bloque a la caché, entonces no hay reemplazo de la vía LRU [elegí esta respuesta]
4. Lectura 0x310 = 0011 0001 0000 ⇒ idx = 1, tag = 0x06 ⇒ MISS
5. Lectura 0x910 = 1001 0001 0000 ⇒ idx = 1, tag = 0x12 ⇒ HIT

2.2. Question 2

Sea un cache de un procesador MIPS32 Cuya arquitectura es FA, con tamaño de bloque de 4 words y capacidad de 1 megabyte. Indique la respuesta correcta.

1. Los 4 bits menos significativos son utilizados como bits de offset
2. Los 2 bits más significativos son utilizados como bits de offset
3. Los 4 bits más significativos son utilizados como bits de offset
4. Los 2 bits menos significativos son utilizados como bits de offset
5. Ninguna es correcta

Opción 1

2.3. Question 3

Sea un cache de un procesador MIPS32 Cuya arquitectura es FA, con tamaño de bloque de 4 words y capacidad de 1 megabyte. Indique la respuesta correcta.

1. Ninguna es correcta
2. Se utilizan 15 bits de índice
3. Se utilizan 16 bits de índice
4. Se utilizan 18 bits de índice
5. No se utilizan bits de índice
6. Se utilizan 17 bits de índice

Caché FA es como un solo conjunto, no se necesitan bits de indice.

2.4. Question 4

Considere un procesador MIPS32 con un cache L1D 4WSA con 8 conjuntos y tamaño de bloque de 4 word. Cada línea incluye los bits valid (V) y dirty (D), el cual es utilizado para implementar una estrategia write-back. La política de reemplazo es LRU

¿Qué capacidad tiene dicho cache expresado en bytes?

a. 512 bytes
b. 1024 bytes
c. 32 Bytes
d. 16 bytes
e. 256 bytes
f. 8 bytes
g. 64 bytes
h. 128 bytes

No estoy segura, planteé:

Se necesitan 4 bits de offset para direccionar 16 bytes por bloque, y 3 bits de indice para direccionar 8 conjuntos; el resto es tag. En definitiva, cada bloque va a tener 1 bit de validez V, 1 bit dirty D, el tag y los datos. Entonces por línea/bloque necesitaría (V+D+tag+offset) bits = 1+1+25+4 = 31, con padding redondeo a 32 bits. Como es 4WSA, voy a tener esa cantidad de bits en 4 sets =>4x32. Además, son 8 conjuntos =>4x32x8 = 1024 bytes.

2.5. Question 5

En un arquitectura MIPS32 se corre el siguiente código:

```
int a[1000];
size_t i, j;
for(i=0; i<1000; i++)
    for(j=0; j<1000; j++)
        a[i]=a[i]+1;
```

Para una cache L1D de 1 KB, 8 words de 32 bits por línea, write back y fully associative, calcule el miss rate (expresado en porcentaje).

1. 0.00625
2. 0.0125
3. 0.05
4. 0.025

---

8 words por línea equivalen a 32 bytes por línea, entonces se necesitan 5 bits de offset.

Como la capacidad es de 1024 bytes, queda $\frac{1024bytes}{32bytes/linea} = 32lineas.$

Es caché de datos así que ignoro los fetches, además notar que hay 2 accesos a a[i], uno en la suma y otro para la asignación.

Cada a[i] ocupa 1 word (porque array de enteros), entonces en cada linea de la caché van a entrar 8 elementos a[i].

0	a[0] a[1] ... a[7]
1	a[8] ... a[15]
..	...
31	a[248] ... a[255]

La función primero hace 2000 accesos (1000 por cada a[i]) al mismo elemento (porque el for de adentro es sobre j). Entonces, sólo va a haber 1 miss compulsivo y después va a ser siempre hit (1 M, 1999 H). Después incrementa el valor de i y vuelve a hacer 2000 accesos, pero ahora van a ser hit los 2000 accesos porque en el ciclo anterior ya traje a[i] a la caché junto a los siguientes 7 elementos. O sea que va a haber 1 miss cada (2000+7x2000) accesos. $\Rightarrow \frac{1}{2000+7x2000} = 6,25x10^{-5}$ que equivale al 0,00625 %

3. Ejercico 3

3.1. Question 1

Se tiene una computadora con soporte de memoria virtual que corre un sistema operativo multiproceso. Las direcciones virtuales y físicas son de 12 bits. La MMU mapea las direcciones virtuales a físicas con tablas de páginas lineales, y las páginas son de 256 bytes. Las tablas de páginas para los procesos A y B se muestran a continuación:

Proceso A

VPN	V	PPN
0x7	1	0xe
0x6	1	0x1
0x5	1	0x0
0x4	1	0x4
0x3	1	0xf
0x2	1	0x3
0x1	1	0x8
0x0	1	0xa

Proceso B

VPN	V	PPN
0x7	1	0x5
0x6	1	0x0
0x5	1	0x3
0x4	1	0x9
0x3	1	0xd
0x2	1	0x2
0x1	1	0x7
0x0	1	0xf

Indicar cuál de las siguientes regiones del espacio de direcciones del proceso A comparte memoria física con el proceso B:

a. 0x2f0 a 0x310
b. 0x000 a 0x0ff
c. 0x4a0 a 0x4f0
d. 0x6f0 a 0x7ff
e. 0x100 0x1f0

No me salió :(

3.2. Question 2

Se tiene un procesador de 32 bits con una MMU que maneja tablas de páginas jerárquicas de 2 niveles, con páginas de 1024 bytes, tablas L1 de 2048 bytes, tablas L2 de 4096 bytes y PTEs de 8 bytes. Indicar cuál de las siguientes respuestas describe correctamente los campos de la virtual address:

a. L1: 8 bits, L2: 9 bits, Offset: 11 bits
b. L1: 8 bits, L2: 9 bits, Offset: 10 bits
c. L1: 10 bits, L2: 11 bits, Offset: 10 bits
d. L1: 9 bits, L2: 10 bits, Offset: 10 bits
e. L1: 9 bits, L2: 10 bits, Offset: 11 bits
f. L1: 10 bits, L2: 11 bits, Offset: 11 bits

[No estoy segura]

La cantidad de bits del offset se saca del tamaño de las páginas. Como son páginas de 1024 bytes, se precisan 10 bits de offset. L1 ocupa 2048 bytes, y como son PTEs de 8 bytes queda $\frac{2048}{8} = 256bytes$ para los cuales se precisan 8 bits. L2 ocupa 4096 bytes, y como son PTEs de 8 bytes queda $\frac{4096}{8} = 512bytes$ para los cuales se precisan 9 bits.

3.3. Question 3

Se tiene un procesador con 64 KB de memoria física, espacio de direcciones virtuales por proceso de 32KB de tamaño, y páginas de 4KB. La MMU traduce mediante tabla de páginas lineal con PTEs de 4 bytes, que poseen un bit de validez (bit 31), y el resto de los bits son la PPN.

Lo que sigue es un volcado de memoria en hexa del comienzo de la tabla de páginas:

Volcado de Memoria

Dirección	Contenido
...	...
0x1101c	0x00000002
0x11018	0x8000000f
0x11014	0x00000003
0x11010	0x80000005
0x1100c	0x80000002
0x11008	0x8000000d
0x11004	0x80000003
0x11000	0x80000006
...	...

Asumiendo que el root pointer contiene el valor 0x11000, indicar cuál de las siguientes es la dirección física para la dirección virtual 0x00002ff0:

a. Traducción inválida
b. 0x03ff0
c. 0x0dff0
d. 0x002f0
e. 0x01ff0
f. 0x00ff0

---

64 KB de memoria física =>16 bits de PA
32 KB por proceso =>15 bits de VA
Páginas de 4 KB =>12 bits de offset
VA = 0x00002ff0 =>VPN = 2
El volcado de memoria arranca en 0x11000, a partir de ahi busco con la VPN teniendo en cuenta que las PTEs son de 4 bytes.

$$rootptr + 4 \times 0x02 = 0x11000 + 0x08 = 0x11008$$

$\Rightarrow PTE = 0x8000000d$ de donde el primer bit es el de validez V=1 y PPN = 0x0d
PA = PPN + offset = 0xdff0

3.4. Question 4

Se tiene un procesador con direcciones virtuales y físicas de 16 bits, páginas de 256 bytes y TLB unificada 2WSA de 4 conjuntos.

El contenido de la TLB es el siguiente:

	W0			W1		
index	V	Tag	PPN	V	Tag	PPN
0	1	0xf	0x3a	1	0x0	0x0f
1	1	0x2	0x10	1	0xc	0xf1
2	1	0x5	0x40	1	0x7	0x00
3	1	0x3	0x5f	1	0x1	0x07

Indicar para la dirección virtual 0x0f00 cuál sería la dirección física correspondiente:

a. 0xf100
b. 0x5f00
c. 0x0f00
d. 0x0000
e. 0x3a00

f. TLB miss, por lo cual no puedo conocer la PA

Páginas de 256 bytes =>8 bits de offset
VA = 0x0f00 =>VPN = 0x0f
Busco en la TLB tag = 0x0f =>HIT en idx = 0, V = 1 =>PPN = 0x3a
PA = PPN + offset = 0x3a00

3.5. Question 5

Se tiene un procesador con un cache L1 direct mapped con 16 bloques de 16 bytes cada uno, con índice virtual y tag físico. Su MMU maneja páginas de 64 bytes. Indicar cuál de los siguientes pares de direcciones virtuales podrían mapear a distintos bloques de cache y a la misma dirección física, es decir, la misma dirección física podría residir en distintos bloques del cache al mismo tiempo (aliasing).

a. 0x360 y 0x340
b. 0xfff y 0xeff
c. 0x200 y 0x100
d. Ninguna opción es correcta
e. 0x44f y 0x34f
f. 0x34c y 0x48c

[No estoy segura]
16 bytes por bloque =>4 bits de offset caché 16 bloques DM =>4 bits de índice Páginas de 64 bytes =>6 bits offset VA

Tengo que buscar direcciones virtuales que podrían mapear a distintos bloque de caché (o sea que tengan distinto índice porque es mapeo directo) y a la misma dirección física (o sea misma VPN). Así que descarto las que tienen mismo indice (eso se ve rápido comparando el segundo byte de la VA porque es virtualmente indexado) =>descarto opciones b,c y e.

f. 0x34c = 0011 0100 1100 =>VPN = 1101
0x48c = 0100 1000 1100 =>VPN= 1001
Tienen distintos VPNs =>distintos PPNs ?? No.. [ver Q.1]

a. 0x360 = 0011 0110 0000 =>VPN = 1101
Además, idx = 6
0x340 = 0011 0100 0000 =>VPN= 1101, idx = 4
Mismo VPN, distinto índice [es la opción que elegí]