P22021sol.pdf

Olmar_eps

Arquitectura de Ordenadores

3º Grado en Ingeniería Informática

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Arquitectura de Ordenadores – Ejercicios Tema 3

Ejercicio 1 y 2. Una CPU con arquitectura Harvard y tamaño de instrucciones y palabra de 16 bits, dispone de un sistema de memoria virtual de 16 MBytes, y una memoria real de 1 MByte. El sistema de memoria es paginado con tamaño de página de 4 kBytes, siendo el tamaño del descriptor de 2 bytes. En la parte alta se encuentra el marco de página, en la baja los bits de control y relleno (padding)

Cuenta, además, con una caché L1 de Instrucciones virtual, asociativa de 4 vías, con capacidad para guardar un total de 64 bloques de 16 bytes cada uno. Y una caché L1 de datos virtual de 4 vías, mismo tamaño de bloques, pero del doble de tamaño. Ambas caches utilizan LRU para su reemplazo.

En la MMU se dispone de un TLB completamente asociativo con 8 entradas. Se muestran algunas de las entradas de la caché, del TLB y de la memoria real. En la memoria caché, los asteriscos (* * *) significan que las entradas no son relevantes para el problema (en la figura se representan el bit de validez, LRU, suciedad y otros de control en el campo "ctr" de 8 bits). En los bloques de datos de la caché, el byte más a la izquierda se corresponde con la dirección más baja, y el más a la derecha, con la dirección más alta. En caso de actualización de la caché, suponer que el algoritmo LRU que no se muestra el contenido especifico en la figura, indica que debe modificarse la vía 1. En las figuras se muestra **parcialmente** el contenido del TLB, de la cache de Instrucciones, la de datos y de memoria y no aparecen los bits de control.
Los tiempos de acceso son los siguientes: Acierto en TLB = 1 ciclo. Fallo TLB, acceso tablas y actualización de TLB = 200 ciclos. Fallo de página: 15000 ciclos. Tiempo de acceso a la Cache=1 ciclos. Tiempo de acceso a un dato en memoria = 60 ciclos. Tiempo de actualizar un Bloque de 16 bytes = 100 ciclos

Cache L1 Instrucciones

	VIA 1			VIA 2	VIA 3	VIA 4
	Tag	ctr	Bloque			
0	AA42	XX	6642 - - - D3F5	* * *	* * *	* * *
--	- - -	XX	- - - - - -	* * *	* * *	* * *
--	- - -	XX	- - - - - -	* * *	* * *	* * *
5	4F21	XX	AAFC - - - 004C	* * *	* * *	* * *
--	- - - - -	XX	- - - - - -	* * *	* * *	* * *
--	- - - - -	XX	- - - - - -	* * *	* * *	* * *
C	55FE	XX	FF44 - - - 3322	* * *	* * *	* * *
--	- - - -	XX	- - - - - -	* * *	* * *	* * *
E	AABB	XX	0102 - - - AFDD	* * *	* * *	* * *
F	0109	XX	F58B - - - 541C	* * *	* * *	* * *

TLB	
NPV	Descrip
010	CC XX
4F2	F0 XX
324	D5 XX
55F	35 XX
AA4	C7 XX
F50	50 XX
AAA	10 XX
100	2C XX

Cache L1 de Datos

	VIA 1	VIA 2	VIA 3	VIA 4
0	* * *	* * *	* * *	* * *
--	* * *	* * *	* * *	* * *
--	* * *	* * *	* * *	* * *
--	* * *	* * *	* * *	* * *
--	* * *	* * *	* * *	* * *
--	* * *	* * *	* * *	* * *
--	* * *	* * *	* * *	* * *
--	* * *	* * *	* * *	* * *
--	* * *	* * *	* * *	* * *
1F	* * *	* * *	* * *	* * *

MEMORIA

DIRECC.	DATO	DIRECC.	DATO	DIRECC.	DATO	DIRECC.	DATO	DIRECC.	DATO
- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -
109E0	C4	35EC0	FF	50D80	44	CC9E0	66	F0150	AA
109E1	2F	35EC1	44	50D81	5C	CC9E1	42	F0151	FC
- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -
109EE	3F	35ECE	33	50D9E	32	CC9EE	D3	F015E	00
109EF	50	35ECF	22	50D9F	F4	CC9EF	F5	F015F	4C
- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -
2FFFE	66	3FFFE	77	5FFFE	88	CFFFE	99	FFFFE	CC
2FFFF	66	3FFFF	77	5FFFF	88	CFFFF	99	FFFFF	CC
- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -	- - - -

Notas: Se indican con "X" los dígitos hexadecimal no relevantes para la solución.
NPV = Número dePagina Vitual (etiqueta/tag)

Arquitectura de Ordenadores – Ejercicios Tema 3

Ejercicio 1:

a) ¿Cuántos comparadores y de que tamaño se utilizan en las caches?

4 comparadores de 16 bit en la cache L1 de instrucciones
4 comparadores de 15 bit en la cache L1 de datos

b) ¿Cuántos comparadores y de que tamaño se utilizan en el TLB?

8 comparadores de 12 bit

c) Suponga que la cache de L1 de datos utiliza política de funcionamiento de posescritura ¿En qué se diferenciaría de la cache de Instrucciones? ¿Tendría sentido tener un cache de instrucciones con posescritura?

En los bits de control estará definido el bit de modificación.
Un cache de instrucciones no se escribe y por tanto no tiene sentido bit de modificación.

d) La CPU solicita una lectura de una instrucción en la dirección 0x4F2150 ¿Cuál es el resultado que se devuelve a la CPU si utiliza notación "little endian" (byte de menor peso en dirección más baja)? Justifique la respuesta.

Acceso a cache virtual L1 instrucciones con TAG 4F21 indice 5 Acierto
Instrucción FCAA

e) La CPU solicita una lectura de una instrucción en la dirección 0x0109EE¿Cuál es el resultado que se devuelve a la CPU si utiliza notación "little endian" (byte de menor peso en dirección más baja)? Justifique la respuesta.

Acceso a cache virtual L1 instrucciones con TAG 0109 indice E Fallo
Acceso al TLB con TAG 010 acierto DR CC9EE
Instrucción F5D3

f) ¿Se modifica el contenido de la cache tras las lecturas de los apartados anteriores? Actualice sobre la figura los valores que se modifican.

Si, se actualiza en la vía 1 la entrada de índice E con TAG 0109 y el bloque correspondiente.

El inicio de la tabla de páginas de un solo nivel y guardada en memoria principal se indica en el registro CR dentro de la MMU. Suponga que el valor de este registro es CR =0x10000. En los descriptores los bits de control están en la parte menos significativa, se utiliza representación “little endian” (el byte de mayor peso corresponde con la dirección de memoria mayor) y que la CPU accede a la DV 0x4F7D80 para leer una palabra.

g) ¿Qué DR corresponde a la DV? Justifique la respuesta

Inicio de la tabla de páginas en dirección de memoria 0x10000.
DV= 0x4F7D80 , NPV = 0x4F7 que se usa como índice para acceder a la tabla (teniendo en cuenta el tamaño del descriptor)
Se accede a la dirección= 0x10000+(0x4F7 x 2) = 0x109EE
Se lee el descriptor 0x503F. Marco 0x50 y junto al offset 0xD80 se genera la DR= 0x50D80.

Se accede a la dirección de memoria 0x50D80 para leer la palabra 0x5C44.

En todo el proceso se accede a las direcciones de memoria 0x109EE y 0x50D80

h) ¿Cuántos accesos a memoria han sido necesarios el proceso de traducción de DV a DR?

Uno para la traducción

(y otro para leer la palabra que busca la CPU)

i) ¿Qué valor obtendrá la CPU?

Se accede a la dirección de memoria 0x50D80 para leer la palabra 0x5C44.

j) ¿Qué direcciones de memoria han sido accedidas en todo el proceso hasta que la CPU obtiene la palabra?

En todo el proceso se accede a las direcciones de memoria 0x109EE y 0x50D80

Arquitectura de Ordenadores – Ejercicios Tema 3

Ejercicio 2

a) Con los tiempos de acceso provistos:
Tiempo de acceso al TLB =1 ciclo
Tiempo de acceso a la Cache=1 ciclos
Tiempo de acceso a un dato en memoria = 60 ciclos.
Tiempo de actualizar un Bloque de 16 bytes = 100 ciclos
Tiempo de acceso a tabla de página (200 ciclos)

Calcule el tiempo la cantidad de ciclos que ha transcurrido cuando la CPU ha requerido las instrucciones de las direcciones siguiente direcciones: 0x4F2150, y 0x4F7D80

0x4F2150, acierto en la cache (1c).
Tiempo de acceso: 1 ciclos.

0x4F7D80, falla en cache (1c), falla en el TLB (1c), accede a tabla de página (200 ciclos). Actualización de Caché (100 ciclos).
Tiempo de acceso: 1 + 1 + 200 + 100 = 302 ciclos
Si se considera que el acceso a caché y TLB se puede solapar. 301.

b) Si las instrucciones de load y store representan el 10% de las instrucciones y la tasa de fallos para L1 de instrucciones es del 2% y la de datos del 4%. ¿cual es el tiempo medio de acceso a memoria? Suponga siempre acierto en TLB y que se solapa el acceso a TLB y Caché.

$T_{medio} = T_{cache}Inst + F_{Ins}*TBloq + InstDatos * (T_{cache}Datos + F_{dato}*TBloq)$
$= 1 + 2\%*100 + 0.1*(1 + 4\%*100) = 3 + 0.5 = 3.5$ ciclos

c) Si para el escenario anterior, se agrega una caché L2 unificada, que en caso de fallo de L1 devuelve el bloque a L1 en 20 ciclos si está presente en L2. Y en caso de no estar en L2, actualizar el bloque desde memoria requiere 100 ciclos. La probabilidad que el dato NO esté presente en L2 ante una petición de las L1 es del 5% ¿Cual es el tiempo medio de acceso a memoria?

$T_{medio} = T_{cache}Inst + F_{Ins}*TL2 + InstDatos * (T_{cache}Datos + F_{dato}*TL2)$
$T_{L2} = T_{acierto}\_L2 + FL2 * TMem = 20$ ciclos $+ 5\%*100$ciclos $= 25$ ciclos

$T_{medio} = 1 + 2\%*25 + 0.1*(1 + 4\%*25) = 1.5 + 0.2 = 1.7$ ciclos

Arquitectura de Ordenadores – Ejercicios Tema 3

Ejercicio 3.- A un sistema procesador con bus de direcciones de 16 bits, tamaño de palabra de 2 bytes y formato de instrucciones de 16 bits, se le desea dotar de una pequeña unidad caché de correspondencia directa para instrucciones, con 16 bloques en total y 8 bytes por bloque. El programa que se va a ejecutar sigue el siguiente patrón:

- Las instrucciones en las direcciones de memoria de 0 a 62, pertenecen a un primer bucle que se ejecuta 2 veces.
- A continuación, las instrucciones en las direcciones de memoria de 64 a 130, que pertenecen a un segundo bucle, se ejecutan 10 veces.

Responda razonadamente a las siguientes preguntas:

a) Indicar como se decodifica la dirección para el acceso a caché.

ETIQUETA	INDICE	B/B
9	4	3

b) Calcular la tasa de fallos.

Una cache de 16 bloques * 8 bytes/bloque almacena 128 bytes → 2 bytes por instrucción → 64 instrucciones consecutivas

Las instrucciones de las primeras 62 direcciones se corresponden con 32 instrucciones que la primera mitad de la caché. Dado que tenemos 8 bytes por bloque, se almacenan 4 instrucciones por bloque, de modo que por cada bloque se produce un fallo y tres aciertos. Se solicitan 32 instrucciones -> 8 fallos 24 aciertos. En la segunda iteración del bucle se producen 0 fallos porque todas las instrucciones están en la caché. Se han solicitado 2 * 32 = 64 instrucciones = (# accesos a la caché).

Las instrucciones de las direcciones 64 a 126 se corresponden con otras 32 instrucciones. En este caso se localizan en la segunda mitad de la caché. Se solicitan 32 instrucciones → 8 fallos 24 aciertos. Las instrucciones 128 y 130 respectivamente pertenecen al mismo bloque, y sobrescriben el índice 0, por lo que se ven afectadas las primeras 4 instrucciones del primer bucle, pero no afecta a este segundo bucle. Se solicitan 2 instrucciones → 1 fallo 1 acierto. Como se ejecuta el bucle 10 veces, el resto de las iteraciones no hay fallos, ya que las instrucciones están alojadas en la segunda mitad de la caché, además del índice 0. Se han solicitado 34 * 10 = 340 instrucciones (# accesos a la cache).

Por tanto, el total de fallos es 8 + 8 + 1 = 17. 17 fallos/404 accesos = 0,0420 → 4,2%

Se quiere añadir soporte para memoria virtual, por lo que se propone incluir una MMU con TLB y un sistema paginado de tres niveles, donde el primer nivel se implementa usando una memoria de sustitución directa dentro de la MMU. El tamaño de página es de 128 bytes para permitir acceso simultáneo al TLB y la caché (caché de direcciones reales). El direccionamiento virtual debe permitir trabajar con 2 MB de memoria, aunque solo disponga de 64 KB de memoria física. El TLB debe ser completamente asociativo con 32 entradas.
Indicar justificando la respuesta:

c) Indicar como se decodifica la dirección para acceder al TLB.

La dirección virtual es de 21 bits. Si el offset ocupa 7 bits, el resto de la dirección se usa como etiqueta dentro del TLB al ser completamente asociativo.

ETIQUETA	OFFSET
14	7

d) Indicar como se decodifica la dirección para acceder a la tabla de páginas. El tamaño de descriptor es de 2 bytes.

La dirección real tiene 16 bits. Si el offset es de 7, el MPR es de 9. Por tanto, el número de bytes mínimo que debe ocupar el descriptor es 2 bytes (16 bits).

Con esta información la decodificación de la dirección virtual para el acceso a la tabla de páginas es:

N1	N2	N3	OFFSET
2	6	6	7

e) El tamaño de la MMU.

La MMU contiene al TLB y la memoria de sustitución directa. Por tanto, el tamaño de la MMU es el tamaño del TLB + tamaño de N1.
Tamaño del TLB = 32 entradas * (14 bits etiqueta + 16 bits descriptor) = 960 bits
Tamaño de N1 = 2^2 posiciones * 16 bits descriptor = 64 bits
Total = 1024 bits = 128 bytes.