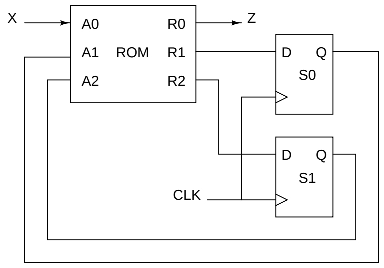
Organización del Computador
Práctico 6

B. Gonzalez Kriegel - N. Wolovick

6 de junio de 2000

1. Se implementa una FSM con una ROM como *lookup-table* y dos FFD.



Locación			Valor		
A_0	A_1	A_2	R_0	R_1	R_2
0	0	0	0	0	1
0	0	1	1	1	0
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	0

Construya la tabla y el diagrama de transición de estados para esta máquina de estados finitos.

2. Diseñe una RAM de 8 palabras $\times$ 32 bits (8×32) usando RAMs de 8×8.

3. Con memorias de 2^n palabras y 8 bits por palabra muestre:
   (a) Como construir una memora de $2^n \times 4p$.
   (b) Como construir una memora de $2^{n+2} \times p$.

4. Diseñe el circuito de un decodificador árbol 4-a-16 sin tener en cuenta restricciones de *fan-in* y *fan-out*.

5. Una memoria caché de mapeo directo consiste en 128 ranuras, cada una conteniendo un bloque de 16 palabras. La memoria contiene 256K[1] palabras. El tiempo de acceso a la caché es de 10ns y el tiempo necesario para llenar una ranura de la caché desde la memoria principal es de 200ns.
   (a) Muestre como esta estructura de caché divide las direcciones de memoria.
   (b) Calcule el porcentaje de aciertos de un programa que itera 10 veces sobre las direcciones de memoria de la 10 a la 200.
   (c) Compute el tiempo de acceso efectivo a memoria para este programa teniendo en cuenta que no se usa *load-through*.

[1] Cuando hablamos de memorias 1K=1024

6. Una caché de mapeo completamente asociativo tiene 16 ranuras para bloques de 8 palabras. La memoria principal es de 2^{16} palabras y la cache está vacía inicialmente. El tiempo de acceso a la caché es de 40ns y el tiempo requerido para transferir las 8 palabras desde la memoria principal a la ranura es de 1μs.
   (a) Calcule el tamaño de los campos *tag* y *word*.
   (b) Compute el porcentaje de aciertos para un programa que ejecuta la secuencia de instrucciones $20 \ldots 45(28 \ldots 45)^4$.
   (c) Calcule el tiempo de acceso efectivo a memoria, suponiendo nuevamente que no se utiliza *load-through*.
7. Explique como el caché de mapeo directo y el caché completamente asociativo pueden ser vistos como una particularización de un mapeo asociativo de conjunto de N elementos.
8. Dada una jerarquía de memoria con caché de mapeo directo de 2^{14} ranuras de 2^5 palabras y memoria principal de 2^{32} palabras.
   (a) ¿Cuán alejados deben estar los accesos a memoria para que todas causen una falla en la caché?
   (b) Usando el ejercicio (8a) calcule la taza de aciertos y el tiempo de acceso efectivo para un programa con $T_{miss} = 1000ns$ y $T_{hit} = 10ns$. Asuma que se opera utilizando *load-through*.
9. A primera vista, la presencia o ausencia de memoria caché no parece tener nada que ver con la decisión de ubicar registros de entrada/salida en el espacio de direcciones de memoria. Piense en los problemas potenciales que esto podría traer.
10. Una computadora tiene 16 páginas de espacio de memoria virtual, pero sólo 4 marcos de página físicos. Tomando inicialmente la memoria vacía, un programa hace referencia a las páginas $(0, 7, 2, 7, 5, 8, 9, 2, 4)$.

    Indique que referencias causan un fallo de página para la política de reemplazo
   (a) LRU.
   (b) FIFO.
11. Un sistema de memoria virtual tiene un tamaño de página de 1024 palabras, 8 páginas virtuales, 4 físicas y usa la política de reemplazo LRU. Dado el siguiente estado de la tabla de páginas

Pag #	Present Bit	Disk Addr	PagFrame Field
0	0	01001011100	xx
1	0	10100100111	xx
2	1	01001000110	00
3	0	00101010000	xx
4	1	00000111100	01
5	0	11001100100	xx
6	1	11101000111	11
7	0	10101000000	xx

   (a) ¿Cuál es la dirección física de la dirección virtual 4096?
   (b) ¿Y para la dirección virtual 1024?
   (c) Se referencia a una posición de memoria en la página 0 de la memoria virtual, produciéndose una falla de página, ¿Qué marco de página físico será utilizado para almacenar esta página virtual?
12. Un programa dado tiene N accesos a memoria en una arquitectura con caché y memoria virtual paginada. Se producen M fallas en la caché y F fallas de página en la memoria virtual. Dados los tiempos T_1:acierto caché, T_2:acierto memoria principal y T_3:tiempo para cargar una página de memoria desde disco, calcular:

(a) Tasa de aciertos de la caché.

(b) Tasa de aciertos de la memoria principal, donde un acierto es cuando una referencia a memoria no genera una falla de página.

(c) Tiempo de acceso efectivo del sistema completo de memoria.

13. Calcule el almacenamiento necesario para la tabla de páginas de un sistema de memoria virtual de 2^{32}bytes, con 2^{12} bytes por página y 8 bytes por entrada de la tabla de páginas.

14. Compute la cantidad de entradas de compuerta para un decodificador de una RAM de 64×1, utilizando decodificadores 2D y 2-1/2D. Suponga *fan-in/fan-out* ilimitado y para ambos utilize decodificadores ordinarios de 2 niveles.

    En el caso del esquema 2-1/2D, suponga que la memoria sólo es leída, evitando el uso del DEMUX para las columnas.

15. * Intente mostrar como hacer *benchmarks tramposos*. Se tienen 2 máquinas con el mismo procesador y memoria principal, pero organizaciones de memoria caché diferentes. Asuma que el *miss time* es 10 veces el tiempo de un acierto para ambas máquinas. Asuma también que escribir una palabra de 32 bits toma 5 veces más que que un acierto de caché (para una caché *write-through*) y que escribir un bloque entero de 32 bytes toma 10 veces más que un acierto de caché (para el esquema *write-back*). El caché está unificado, es decir contiene instrucciones y datos.

    Caché A 128 conjuntos, 2 elementos por conjunto, bloques de 32 bytes, con *write-through* y *no-write-allocate*.

    Caché B 256 conjuntos, 1 elemento por conjunto, bloques de 32 bytes, con *write-back* y *write-allocate*.

    (a) Describa un programa en ensamblador que haga correr la máquina A lo más rápido posible respecto a la máquina B.

    (b) Idem pero con la máquina B ganándole a la A.

16. * En este ejercicio se correrá un programa para evaluar el comportamiento de un sistema de memoria. Un programa en C para UNIX que se encuentra para bajar en la página de la materia, nos permite hacer estas mediciones (`stride.c`).

    En la primera parte este programa se presenta una función para obtener una medición precisa del tiempo. En la segunda parte, un arreglo se lee y escribe en rangos crecientes de longitud 2^n y por cada uno de estos rangos se hacen accesos de a saltos (*strides*) cada vez más grandes, desde 2^0 hasta 2^{n-1}. El código se repite varias veces a fin de obtener mayor precisión en la medición.

    La tercera parte repite exactamente el mismo lazo pero no efectua operaciones con la memoria para poder medir y luego restar el tiempo de *overhead* que tiene la repetición. La parte final imprime los tiempos de acceso de la memoria para cada tamaño y salto.

    Resulta importante asegurar que el compilador hace caso al modificador `register`, a fin de no interferir la memoria con lecto-escrituras de las variables que controlan los lazos.

    (a) Grafique los resultados experimentales con el eje X indicando saltos de memoria y el eje Y los tiempo de acceso y una serie por cada tamaño de de caché. Use escala logarítmica para el eje X.

    (b) ¿Cuántos niveles de caché hay en la jerarquía de memoria?

    (c) ¿Cuál es el tamaño de la cache L1? ¿Cuál es su asociatividad?

    (d) ¿Cuál es el tamaño de la cache L2 (si existe)? ¿Cuál es su asociatividad?

    (e) ¿Hay algo más que usted haya descubierto acerca de la jerarquía de memoria a partir de estos experimentos?