ESTRUCTURA DE COMPUTADORS (ETSINF)

Curs 2015/2016

Pràctica 11

EL SISTEMA DE MEMÒRIA CAU EN EL MIPS R2000
LA CAU DE DADES

Introducció

En esta pràctica es contínua el treball començat en la pràctica anterior amb la memòria cau del MIPS R2000. Incidix en el funcionament de la memòria cau de dades i la seua influència en el temps d'execució dels programes i s'utilitza com a punt de partida el mateix programa de la pràctica anterior. La ferramenta de treball continua sent el simulador del processador MIPS R2000 denominat PCSpim-Cache.

Objectius

- Determinar com ajuda la memòria cau per a reduir el temps d'accés a la informació (dades).
- Conéixer com la memòria cau interpreta les adreces emeses pel processador.
- Analitzar com influïx l'organització de la memòria cau en la taxa d'encerts.

Material

El material es pot obtindre de la carpeta de recursos de PoliformaT.

- Simulador PCSpim-Cache del MIPS R2000.
- Arxius font (`programa.s`).

Programa de treball: producte d'un vector per una constant

El *programa original* és el mateix de l'anterior pràctica de la memòria cau.

```
                ######################################################
                # Segment de dades
                ######################################################

                .data 0x10000000
A:              .word 0,1,2,3,4,5,6,7   # Vector A
                .data 0x10001000
B:              .space 32               # Vector B (resultat)
                .data 0x1000A030
k:              .word 7                 # Constant escalar
dim:            .word 8                 # Dimensió dels vectors
```

```
                ########################################################
                # Segment de codi
                ########################################################

                .text 0x00400000
                .globl __start

__start:        la $a0, A               # $a0 = adreça de A
                la $a1, B               # $a1 = adreça de B
                la $a2, k               # $a1 = adreça de k
                la $a3, dim             # $a2 = adreça dimensió
                jal sax                 # Crida a subrutina

                ########################################################
                # Fi d'execució per mitjà de crida al sistema
                ########################################################

                addi $v0, $zero, 10     # Codi per a exit
                syscall                 # Fi de l'execució

                ########################################################
                # Subrutina que calcula Y <- k*X
                # $a0 = Adreça inici vector X
                # $a1 = Adreça inici vector Y
                # $a2 = Adreça constant escalar k
                # $a3 = Adreça dimensió dels vectors
                ########################################################

sax:            lw $a2, 0($a2)          # $a3 = constant k
                lw $a3, 0($a3)          # $a3 = dimensió
bucle:          lw $t0, 0($a0)          # Lectura de X[i] en $t0
                mult $a2, $t0           # Efectua k*X[i]
                mflo $t0                # $t0 <- k*X[i] (HI val 0)
                sw $t0, 0($a1)          # Escriptura de Y[i]
                addi $a0, $a0, 4        # Adreça de X[i+1]
                addi $a1, $a1, 4        # Adreça de Y[i+1]
                addi $a3, $a3, -1       # Disminució número elements
                bgtz $a3, bucle         # Bota si queden elements
                jr $ra                  # Retorn de subrutina

                .end
```

Memòria cau de dades

Considerarem ara l'existència d'una memòria cau de dades amb les següents característiques:

Paràmetre	Valor
Capacitat	256 bytes
Correspondència	Directa
Bloc o línia	16 bytes
Política d'escriptura	Directa (*write throught*) amb ubicació (*write allocate*)

La correspondència és directa, com en el cas anterior (pràctica anterior) en el que hem analitzat el comportament de la memòria cau de codi. En este apartat hem de fixar-nos en els accessos que es fan en el programa per mitjà de les instruccions de *load* i *store*. En este cas, hi ha una lectura (instrucció `lw`) per a la constant `k` i una altra per a la grandària dels vectors `dim`, així com una lectura (`lw`) i una escriptura (`sw`) per cada element dels vectors. **De moment no cal que utilitzes el simulador.**

1. ►Tenint en compte les característiques anteriors, indica quantes línies hi ha en la memòria cau i quantes línies ocupa cada vector.

2. ►Indica quina serà la interpretació que esta memòria cau farà de les adreces que reba (camps d'etiqueta, línia i desplaçament).

Si et fixes en la declaració de les variables del programa i de la seua ubicació en el segment de dades per mitjà de la directiva `.data`, podras comprovar que, d'acord amb la interpretació de les adreces, les variables `k` i `dim` s'ubiquen en la línia número 3.

3. ►Indica en quines línies de la cau s'ubiquen els vectors `A` i `B`.

Tin en compte que, si ambdós vectors compartixen les mateixes línies de cau i l'accés als mateixos es realitza de forma consecutiva, com ocorre en el nostre programa, la memòria cau no ens servix per a reduir el temps d'execució. Comprovarem açò per mitjà del simulador.

4. ►Carrega el *programa original* en el simulador configurat amb les característiques anteriors per a la memòria cau de dades i executa'l per mitjà de l'opció F10 (pas a pas) a fi d'analitzar el seu comportament. Assegura't de que els vectors s'emmagatzemen en les línies previstes de la cau de dades i després completa la taula següent:

Accessos al segment de dades	
Encerts	
Fallades	
Taxa d'encerts (H)	

5. ►Quin ha sigut l'accés que ha provocat l'únic encert en la memòria cau de dades?

6. ►En este cas, raona si el nombre tan escàs d'encerts obtingut es deu a la localitat de les referències del programa o bé cal atribuir-ho a la relació entre la configuració de la cau de dades i la ubicació d'estos últims en memòria principal.

En els apartats següents tractarem d'augmentar la taxa d'encerts en els accessos al segment de dades evitant que els dos vectors que maneja el programa s'ubiquen en les mateixes línies de memòria cau. En resum, experimentarem amb les accions següents:

1. Canviar la política d'escriptura de la cau per a evitar que hi haja ubicació en cas de fallada.
2. Canviar la ubicació dels vectors en el segment de dades per mitjà de les directives del programa `.data`.
3. Canviar la correspondència de la memòria cau perquè hi haja una major asociatividad. En este cas provarem amb una política associativa per conjunts i amb una política totalment associativa.

Primera alternativa: canvi en la política d'escriptura

El paràmetre responsable de què el vector `B` es porte a la memòria cau de dades és la utilització d'una política d'ubicació en escriptura quan es produïx una fallada. En conseqüència, per a evitar que el vector `B` entre en conflicte en la cau amb el vector `A` cal aplicar una política de no ubicació en escriptura.

7. ►Configura la memòria cau de dades amb una política d'escriptura directa (*write through*) sense ubicació (*no write allocate*). Executa ara el *programa original* per mitjà de l'opció F10 (pas a pas) a fi d'analitzar el seu comportament. Completa la taula següent:

Accessos al segment de dades	
Encerts	
Fallades	
Taxa d'encerts (H)	

Si examinem l'execució del programa veurem que hi ha una fallada per cada accés al vector `B`, dos fallades en l'accés al vector `A` (lectura dels elements `A[0]` i `A[4]`) i una fallada en la lectura de la variable `k`.

Ara hauràs pogut comprovar que el vector `B` no arriba a desplaçar al vector `A` de la memòria cau i, per tant, no hi ha fallades degut a la col·lisió entre els blocs d'ambdós vectors. No obstant això, encara es continuen produint fallades perquè amb la política de no ubicació hem impedit que el vector `B` arribe a la memòria cau. De fet, mai podrà ubicar-se en esta memòria perquè este vector només s'escriu però mai es llig.

Segona alternativa: canvi en la ubicació dels vectors

Una manera diferent de millorar la taxa d'encerts mantenint *la política d'ubicació en escriptura* consistix a canviar la ubicació dels vectors en el segment de dades. En efecte, ja sabem que les directives `.data` del programa original prèvies a la declaració d'ambdós vectors fan que compartisquen la mateixa línia de cau. En hi ha prou, per tant, de canviar l'adreça d'inici d'un d'estos vectors a fi de fer que la correspondència directa els situe en línies diferents.

8. ►Canvia la directiva `.data 0x10001000` de manera que el vector `B` s'ubique en les línies 4 i 5 de la memòria cau. D'esta manera no hi haurà col·lisió amb les línies 0 i 1 (ubicació del vector `A`) ni amb la línia 3 (ubicació de la constant `k` i de `dim`).

9. ►Executa el programa amb el canvi anterior (recorda que ha de mantindre la política d'escriptura amb **ubicació**) i complete la taula següent:

Accessos al segment de dades	
Encerts	
Fallades	
Taxa d'encerts (H)	

Tercera alternativa: augment de l'asociatividad de la cau

En este últim apartat canviarem la configuració de la memòria cau amb els paràmetres següents:

Paràmetre	Valor
Capacitat	256 bytes
Correspondència	Associativa per conjunts
Nombre de vies	2
Bloc o línia	16 bytes
Política d'escriptura	Directa amb ubicació
Política de reemplaçament	LRU

En esta nova situació hem augmentat l'asociatividad de la cau: un bloc provinent de la memòria principal pot triar entre les dos línies del conjunt que li correspon. Recorda que estem modificant la configuració de la memòria cau de dades per a executar el *programa original*. **De moment no cal que utilitzes el simulador**.

10. ►Indica quina serà la interpretació que esta memòria cau farà de les adreces que reba (camps d'etiqueta, conjunt i desplaçament).

11. ►Indica en quins conjunts s'ubicaran ara els vectors `A` i `B` del *programa original*.

En vista dels valors anteriors, observem que els blocs dels vectors `A` i `B` s'ubiquen sobre els mateixos conjunts de la memòria cau. Recordem que la correspondència directa empleada fins ara ubicaba els blocs en les mateixes línies de memòria, originant reemplaçaments durant l'execució del programa. Estem ara en la mateixa situació? Què és el que ha canviat?

12. ►Carrega en el simulador i executa el *programa original* amb la nova configuració de la cau i completa la taula següent:

Accessos al segment de dades	
Encerts	
Fallades	
Taxa d'encerts (H)	

Com s'ha pogut apreciar durant l'execució del programa, en este cas no es produïx cap reemplaçament. Totes les fallades són originades pel primer accés al bloc seleccionat. Així doncs, en este cas s'ha produït una fallada en l'accés al bloc que conté les variables `k` i `dim`, dos fallades en l'accés al vector `A` i altres dos fallades en l'accés al vector `B`.