Laboratori Sessió 07: Repàs de Memòries Cache (I)

Objectiu de la Sessió

L'objectiu de la sessió és recordar conceptes de memòria cache vistos a EC I. Per fer-ho, programareu un simulador de cache bàsic que simuli una cache de lectura.

Característiques de la cache

En aquesta sessió farem servir una memòria cache amb les següents característiques:

- Les adreces són de 32 bits (per simplificar assumirem que tots els accessos són a byte)
- La cache és de només lectura (assumirem que tots els accessos són lectures)
- Algorisme d'emplaçament : Emplaçament directe
- Mida Cache: 4 Kbytes
- Mida Línia: 32 bytes

Presa de contacte amb l'entorn del simulador

El simulador es composa de 3 fitxers: CacheSim.o, CacheSim.h i MiSimulador.c

El programa principal i alguns components del simulador ja estan programats i és troben al fitxer CacheSim.o. Aquest fitxer s'encarrega de generar seqüències de test, d'imprimir els resultats de la simulació per pantalla amb un format agradable i de comprovar el correcte funcionament del vostre simulador.

Abans de que comenceu a programar el simulador, és interessant fer algunes proves amb aquest entorn. Per començar, compileu el simulador (MiSimulador.c no funciona correctament, però compila)

```
$> gcc CacheSim.o MiSimulador.c –o sim
```

El programa disposa de 3 tests:

- Test 0: Genera la seqüència de 20 referències de la taula del treball previ.
- Test 1: Genera accessos seqüencials a un vector d'enters (1000 referències)
- Test 2: Genera els accessos d'un producte de matrius de 25x25 (62500 referències)

Per passar qualsevol dels tests, només cal posar el nº de test com a paràmetre del simulador. Per exemple, per passar el test 0 faríem:

```
$> sim 0
Test 0 FAIL :-(
$>
```

Evidentment el test ha fallat, ja que encara no hem programat el simulador. En cas de que el simulador falli, ens interessarà veure que esta passant. Per això podem utilitzar la opció "v" (de verbose) al simulador (la v cal que aparegui com el primer paràmetre):

```
$> sim v 0
  eca130 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  eca131 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ec2172 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  eca133 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ec3175 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ec3175 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ecb136 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  eca137 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ec2178 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ecb139 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
10eca230 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  eca131 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ec2172 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
10eca233 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ec3175 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ec3175 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ecb136 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
10eca237 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
```

```
  ec2278 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
  ecb139 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
Test 0 FAIL :-(
```

Aquesta opció ens donarà una sortida semblant a l'anterior. Com podem veure, les columnes es corresponen bàsicament a la taula de l'exercici previ. D'aquesta manera, comparant la sortida i la taula podem veure on és el problema. Donat que en els tests 1 i 2 el nº de referències és molt alt, us recomanem que no els proveu fins que us funcioni perfectament el test 0. Amb la opció v, els tests 1 i 2 es paren tan bon punt apareix el primer error per tal de facilitar la seva identificació.

Programació del mòdul "MiSimulador.c"

Per programar el vostre simulador de cache heu de programar 3 seccions del fitxer "MiSimulador.c":

Estructures globals

En aquesta secció heu de declarar les estructures de dades globals necessàries per mantenir l'estat de la cache. Es necessari que siguin globals, ja que la part principal del simulador es la rutina "reference" que s'executa un cop per referència i, com ja sabeu molt be del tema de subrutines, el seu estat desapareix un cop executada.

Inicialització de la cache

La rutina "init_cache" és cridada abans de passar cada test per inicialitzar les estructures de dades globals necessàries. L'objectiu és deixar la cache en un estat inicial correcte (cache buida).

Simulació de referències

La simulació de les referències cal fer-la a la rutina "reference". Aquesta rutina és cridada un cop per cada referència a simular. Només cal que genereu el valor correcte de les 7 variables locals que ja teniu declarades a l'inici de la rutina, i que és corresponen bàsicament columnes del treball previ (excepte el boolea replacement, que no era necessari al treball prèvi).

```
void reference (unsigned int address)
{
     unsigned int byte;
     unsigned int linea_mp;
     unsigned int linea_mc;
     unsigned int tag;
     unsigned int miss;        // booleà que ens indica si és miss
    unsigned int replacement; // booleà que indica si es reemplaça una línia vàlida
     unsigned int tag_out;     // TAG de la línia reemplaçada
```

En altres paraules, el que heu de fer es implementar l'algorisme que, de forma intuïtiva, heu fet servir manualment per emplenar la taula prèvia.

Després del vostre codi, la rutina acaba amb una crida a la rutina "test_and_print" per comprovar si els valors de les variables son correctes i treure'ls per pantalla en cas de tenir la opció "v" activada.

Resultats de la sessió

Abans d'acabar la sessió heu d'haver programat el simulador de cache de lectura i comprovat el seu funcionament correcte.

Notes

Al fitxer "`CacheSim.h`" hi ha declarades les constants "`true`" i "`false`", que podeu fer servir per assignar a les variables booleanes.

Els fitxers per aquesta sessió els podeu trobar a la pàgina web de la assignatura: "`Programas.Sesion07.tar.gz`".

Treball previ a la sesió

Empleneu la taula de sota, indicant per cada referència de la seqüència de referències la informació següent:

- el byte de la línia que és accedit (**byte**),
- la línia de memòria principal (**línia MP**),
- la línia de memòria cache on es mapejarà la referència (**línia MC**),
- la etiqueta (**TAG**) que es guardarà d'aquesta referència,
- si el accés és **HIT** o **MISS**,
- i en cas que es reemplaci una línia vàlida, el TAG de la línia reemplaçada (**TAG out**).

adreça	byte	línia MP	línia MC	TAG	HIT/MISS	TAG out
00eca130						
00eca131						
00ec2172						
00eca133						
00ec3175						
00ec3175						
00ecb136						
00eca137						
00ec2178						
00ecb139						
10eca230						
00eca131						
00ec2172						
10eca233						
00ec3175						
00ec3175						
00ecb136						
10eca237						
00ec2278						
00ecb139						

Nota: Perquè la taula us sigui d'utilitat cal emplenar les columnes en hexa.