1g) разрядность адреса – 50; размер кэш-памяти – 32 Кбайт; размер блока – 64 Б; ассоциативность – 4;
тип записи – сквозная (write-through); политика заведения – первый свободный блок набора, промах по чтению;
политика вытеснения – MRR (Most recently read).

Структура адреса

Адрес делится на:
- Смещение (offset): 6 бит (для указания внутри блока).
- Индекс: $log2(2^7) = 7$бит для выбора набора.
- Тег: 50 - 7 - 6 = 37бит для идентификации уникального блока.

Входная последовательность с явными адресами (не приводящее к вытеснению)

Для примера возьмем следующие 50-битные адреса:

- Addr1=0x0000000001\text{Addr1} = 0x0000000001,
- Addr2=0x0000000041\text{Addr2} = 0x0000000041,
- Addr3=0x0000000081\text{Addr3} = 0x0000000081,
- Addr4=0x00000000C1\text{Addr4} = 0x00000000C1,
- Addr5=0x0000000101\text{Addr5} = 0x0000000101.

Адреса выбраны таким образом, чтобы различались теги, но попадали в один и тот же набор (с одинаковым индексом).

Входная последовательность минимальной длины, приводящая к вытеснению модифицированных данных (исходное состояние: кэш пуст)

Для того чтобы произошло вымещение из кэша нам необходимо подать 6 разных запросов с 0 индексом и 0 смещением, отличие у запросов будет только в графе: Тег.

Таким образом для достижения условия вымещения данных из допустим 1-го set кэша, нам необходимо подать запросы (Запросы записаны в 16-ти ричной системе счисления):

0x0000 load 0

0x2000 load 2^{13}

0x4000 load 2^{14}

0x6000 load $3 * 2^{13}$

0x6000 store $3 * 2^{13}$

0x8000 load $4 * 2^{13} = 2^{15}$

Таким образом 6-й запрос приведет к вымещению блока с адресом из set, благодаря тому что политика вытеснения MRR подразумевает выгрузку последнего прочитанного блока.

Конечный автомат Мили

Определим множество состояний конечного автомата:

Ассоциативность A = 4

Размер блока B = 2^b = 64, b = 6

Политика заведения: по чтению, первый свободный блок набора

Политика вытеснения: MRR (Most recently read), в первую очередь вытесняет последний прочитанный блок

Входной алфавит:

X = {ld, st, inv} x {tagi, i∈{0,4}}, где

ld – чтение;

st – запись;

inv – операция инвалидации (сброс значимости для не модифицированного блока, или вытеснение модифицированного блока)

Выходной алфавит:

Y = {ld, ld+st, st, ∅}

Состояние:

S = $(fulltag_i)_{i=0}^{n}$

При этом n = 3, так как ассоциативность равна 4.

Полный тег:

$$\text{fulltag} = \begin{cases} (v = 0, \emptyset) \\ (v = 1, tag, age \in \{1,4\}, \text{modified} \in \{0,1\}) \end{cases}$$

Для описания поведения 1 набора достаточно A+1 различных тегов:

tag ∈ {tagi, i∈{0,4}}, таким образом имеем 5 необходимых тегов

Начальное состояние:

S0 = (∅, ∅, ∅, ∅)

3.1 Чтение с промахом (без вытеснения):

S (ld, tag)

S: ∃k: v_k = 1 ∧ tag_k = tag

∃j: v_j = 0 ∧ ∀$_i$ < j v_i =1

S`=fulltag$_i$ = $\begin{cases} (v_i, tag_i, age = 1, mod = 0),\ i = j \\ (v_i, tag, age + +),\ i \neq j\ \wedge\ v_i = 1\ \wedge\ age_i < age_j \\ (fulltag_i),\ i \neq j\ \wedge\ v_i = 0 \end{cases}$

Y = ld

3.2 Чтение с промахом (с вытеснением):

S (ld, tag)

S: ~~∃~~k: v_k = 1 ∧ tag_k = tag

~~∃~~j: v_j = 0

∃ m: age_m = 1 (Так как у нас политика вытеснения MRR – необходимо вытеснить блок, который читался последним).

S` = fulltag$_i$ $\begin{cases} (1, tag, age = 1, mod = 0)\ i = m \\ (v_i, tag_i, age_i, mod_i)\ i \neq m \end{cases}$

Y = $\begin{cases} (ld),\ modified = 0 \\ (ld + st),\ modified = 1 \end{cases}$

3.3 Чтение с попаданием:

S: (st, tag)

S: $\exists$ k: $v_k = 1 \wedge tag_k = tag$

S`=fulltag$_i$ = $\begin{cases} (v_i, tag_i, age = 1),\ i = k \\ (v_i, tag_i, age_i + +),\ i \neq k \wedge v_i = 1 \wedge age_i < age_k \\ (fulltag_i),\ i \neq k \wedge v_i = 0 \end{cases}$

Y = $\varnothing$

3.4 Запись с попаданием:

S (st, tag)

S: $\exists$ k: $v_k = 1 \wedge tag_k = tag$

S` = fulltag`$_i$ $\begin{cases} (fulltag_i),\ v_i = 0 \\ (1,\ tag_i, age_i, modified_i),\ v_i = 1 \wedge \mathrm{i} \neq \mathrm{k} \\ (1, tag, age_i, modified` = 1),\ i = k \end{cases}$

Y = $\varnothing$

3.5 Инвалидация с попаданием:

S (inv, tag)

S: $\exists$ k: $v_k = 1 \wedge tag_k = tag$

S` =fulltag$_i$ $\begin{cases} (v_i),\ v_i = 0 \wedge i = k \wedge mod_i = 0 \\ (fulltag_i),\ i = k, mod_i = 1 \end{cases}$

При этом операция инвалидации в нашем случае может произойти вычеркивание немодифицированного блока памяти или выгрузка модифицированного блока, отсюда выходные состояния:

Y = $\begin{cases} \emptyset,\ mod_k = 0 \\ st,\ mod_k = 1 \end{cases}$

3.6 Инвалидация с промахом:

S (inv, tag)

S: $\nexists$ k: $v_k = 1 \wedge tag_k = tag$

S`=S

Y = $\varnothing$

Расчёт тестового покрытия

S, X

F:X*S > S`

Так как в качестве входного алфавита мы имеем:

Fulltag = {(v, tag, age), (v, tag, age), (v, tag, age), (v, tag, age)}

Для v имеет 2 состояния

Для tag имеем 5 состояний

Для age имеем 4 состояния

Получаем

$2^4 * 5^5 * C_4^5$