Задание 2

Логическая организация кэш-памяти; тестовое покрытие

Для кэш-памяти данных с заданными параметрами:

1. Описать структуру адреса;
2. Построить конечный автомат Мили, описывающий набор кэш-памяти (set); выходной алфавит должен позволять различать адреса (теги) запросов;
3. Описать входную последовательность минимальной длины, приводящую к вытеснению модифицированных данных (исходное состояние: кэш пуст);
4. Рассчитать достигнутое тестовое покрытие в метрике состояний и в метрике переходов построенного конечного автомата.

Возможные события:

- Операции чтения/записи со стороны вычислительного ядра;
- Снуп-инвалидатор со стороны подсистемы памяти: при его получении кэш вычёркивает немодифицированный блок с заданным адресом либо вытесняет модифицированный блок в память.

Структура кэш-памяти (вариант "d"):

- Разрядность адреса (W_a) – 48;
- Размер кэш-памяти (S) – 2 Мбайта;
- Размер блока (B) – 64 байта;
- Ассоциативность (A) – 32;
- Тип записи – отложенная (write-back);
- Политика заведения – первый свободный блок набора;
- Промах по чтению и записи;
- Политика заведения – LRU.

1. Описать структура адреса

Для смещение внутри блока необходимо иметь

$$\log_2(B) = \log_2(64) = 6 \text{ бит.}$$

Для определения поля под "номер набора" нужно иметь

$$2^n = \frac{S}{A * B} = \frac{2 \text{ Мбайт}}{32 * 64 \text{ байт}} = \frac{2^{21}}{2^5 * 2^6} \Rightarrow n = 10 \text{ бит}$$

Оставшееся место для тега блока: $48 - 6 - 10 = 32$ бита. Структура адреса будет:

Тег	Индекс	Смещение
[48 : 16]	[15 : 6]	[5 : 0]

2. Построить конечный автомат Мили

Ограничимся набором тегов $A + 1 = 33$.

Конечный автомат Мили:

$$A_m = (S_t, X, Y, \delta : S \times X \to S, s_0 \in S, \lambda : S \times X \to Y)$$

- Набор состояний: $S_t = (fulltag_i)_{i=0}^{31}$,

$$fulltag_i = \begin{cases} (v = 0, \varnothing) \\ (v = 1, \mathrm{tag} \in \{1, \ldots, A+1\}, \mathrm{age} \in \{1, \ldots, A\}, \mathrm{modified} \in \{0, 1\}) \end{cases}$$

v – валидность, tag – один из 33 тегов, age – давность использования, $modified$ – флаг изменения.

- Начальное состояние – $s_0 = [(0, \varnothing), (0, \varnothing), ..., (0, \varnothing)]$
- Входной алфавит – $X = \{ld, st, inv\} \times \{tag_i, i = \overline{0, 32}\}$
- Выходной алфавит – $Y = \{ld, st, st + ld, \varnothing\}$
- Функции переходов – $\delta : (s, x) \to s'$
- Функции выходов – $\lambda : (s, x) \to y$

Запись

$$\delta : ((st, tag), s) \to s'$$

Попадание: $s : \exists k : v_k = 1 \land tag_k = tag$

$$s' : fulltag'_i = \begin{cases} fulltag_i & v_i = 0 \lor (v_i = 1 \land age_k < age_i) \\ (1, tag_i, age_i + 1, modified_i) & v_i = 1 \land i \neq k \land age_i < age_k \\ (1, tag, 1, 1) & i = k \end{cases}$$

$$y = \varnothing$$

Промах без вытеснения: $s : (\nexists k : v_k = 1 \land tag_k = tag) \land (\exists j : v_j = 0)$, j – минимальное из свободных (так как политика заведения – первый свободный)

$$s' : fulltag'_i = \begin{cases} fulltag_i & v_i = 0 \land i \neq j \\ (1, tag_i, age_i + 1, modified_i) & v_i = 1 \\ (1, tag, 1, 1) & v_i = 0 \land i = j \end{cases}$$

$$y = ld$$

Промах с вытеснением: $s : (\nexists k : v_k = 1 \land tag_k = tag) \land (\nexists j : v_j = 0) \land (\exists d : age_d = 32)$

$$s' : fulltag'_i = \begin{cases} (1, tag_i, age_i + 1, modified_i) & i \neq d \\ (1, tag, 1, 1) & i = d \end{cases}$$

$$y = \begin{cases} ld + st & modified_d = 1 \\ ld & modified_d = 0 \end{cases}$$

Чтение

$$\delta : ((ld, tag), s) \to s'$$

Попадание: $s : \exists k : v_k = 1 \wedge tag_k = tag$

$$s' : fulltag_i' = \begin{cases} fulltag_i & v_i = 0 \vee (v_i = 1 \wedge age_i > age_k) \\ (1, tag_i, age_i + 1, modified_i) & v_i = 1 \wedge age_i < age_k \\ (1, tag, 1, modified_i) & i = k \end{cases}$$

$$y = \varnothing$$

Промах без вытеснения: $s : (\nexists k : v_k = 1 \wedge tag_k = tag) \wedge (\exists j : v_j = 0)$

$$s' : fulltag_i' = \begin{cases} fulltag_i & v_i = 0 \wedge i \neq j \\ (1, tag_i, age_i + 1, modified_i) & v_i = 1 \\ (1, tag, 1, 0) & v_i = 0 \wedge i = j \end{cases}$$

$$y = ld$$

Промах с вытеснением: $s : (\nexists k : v_k = 1 \wedge tag_k = tag) \wedge (\nexists j : v_j = 0) \wedge (\exists d : age_d = 32)$

$$s' : fulltag_i' = \begin{cases} (1, tag_i, age_i + 1, modified_i) & i \neq d \\ (1, tag, 1, 0) & i = d \end{cases}$$

$$y = \begin{cases} ld + st & modified_d = 1 \\ ld & modified_d = 0 \end{cases}$$

Инвалидация

$$\delta : ((inv, tag), s) \to s'$$

Попадание: $s : \exists k : v_k = 1 \wedge tag_k = tag$

$$s' : fulltag_i' = \begin{cases} fulltag_i & v_i = 0 \vee (v_i = 1 \wedge age_i < age_k) \\ (1, tag_i, age_i - 1, modified_i) & v_i = 1 \wedge age_i > age_k \\ (0, \varnothing) & i = k \end{cases}$$

$$y = \begin{cases} st & modified_k = 1 \\ \varnothing & modified_k = 0 \end{cases}$$

Промах: $s : \nexists k : v_k = 1 \wedge tag_k = tag$

$$s' = s$$

$$y = \varnothing$$

3. Пример – вытеснение модифицированных данных

Исходное состояние – кэш пуст:

$$s_0 = [(0, \varnothing), ..., (0, \varnothing)]$$

Политика вытеснения LRU – это значит вытеснению будет подлежать самый "старый" блок. Этот же блок должен быть (а) модифицированным и (б) должен (желательно) быть вытеснен первым. Но так как изменение ведёт к обнулению возраста среди прочих блоков, то этот блок должен быть изменён до появления других блоков, приводящих к его вытеснению. А значит, что этот блок, который мы обязаны сразу же модифицировать, должен быть первым занесён в кэш:

$$\begin{array}{lr} st,\ 2^{16}*(0) & \Leftarrow \text{помещение первого блока} \\ st,\ 2^{16}*(1) & \\ st,\ 2^{16}*(2) & \\ \vdots & \\ st,\ 2^{16}*(30) & \\ st,\ 2^{16}*(31) & \\ st,\ 2^{16}*(32) & \Leftarrow \text{вытеснение первого блока} \end{array}$$

4. Тестовое покрытие

- **Метрика состояний**

Пусть будет $k \in \{0, 1, \ldots, 32\}$ валидных блоков.

Для k валидных блоков будет $\mathbf{C}_A^k$:

$$\mathbf{C}_A^k = \frac{A!}{k!(A-k)!}$$

Каждый из k валидных блоков может быть заполнен $tag \in \{1, \cdots, 33\}$, но без повторений. То есть для первого из валидных блоков (если такой есть) будет 33 варианта тега, для второго (если есть) будет 32 и так далее. Итогово, если k валидных, то комбинаций с тегами будет:

$$\mathbf{A}_{A+1}^k = \frac{(A+1)!}{(A+1-k)!}$$

Каждый из этих k валидных блоков имеет возраст $age \in \{1, \cdots, k\}$ – вариантов распределения возрастов среди этих блоков:

$$k!$$

Также, любой валидный блок может быть как модифицирован, так и немодифицирован:

$$2^k$$

Итого состояний:

$$|S| = \sum_{k=0}^{32} \mathbf{C}_A^k \mathbf{A}_{A+1}^k k! 2^k = \sum_{k=0}^{32} \frac{A!}{k!(A-k)!} \frac{(A+1)!}{(A+1-k)!} k! 2^k = \sum_{k=0}^{32} \frac{A!}{(A-k)!} \frac{(A+1)!}{(A+1-k)!} 2^k$$

$$|S| \approx 6.1 * 10^{107}$$

точное значение: 610658749490633920905628536908573609957420537965769233311973133492019439684354198481936553510387047429721121

В примере было пройдено 33 состояний, тестовое покрытие:

$$\frac{33}{|S|} \approx 5.4 * 10^{-107}$$

- **Метрика переходов**

  На каждый $fulltag$ в любом состоянии может произойти 3 различных операций с 33 различными тегами, тогда всего переходов будет: $3*33*6.1*10^{107} \approx 1.6*10^{110}$. С учётом того что переходов было 32, то тестовое покрытие будет:

$$\frac{32}{3*33*|S|} \approx 2*10^{-109}$$