**TRƯỜNG ĐẠI HỌC BÁCH KHOA ĐẠI HỌC QUỐC GIA TP. HCM**

**KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**



**KIẾN TRÚC MÁY TÍNH**

**LAB 8 : BỘ NHỚ CHÍNH, BỘ NHỚ ĐỆM VÀ HIỆU NĂNG BỘ NHỚ**

**LỚP: L07 NHÓM: 4**

**HK231**

**SINH VIÊN THỰC HIỆN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **HỌ** | **TÊN** | **% Đóng góp** |
| 1 | 2211133 | Phan Ngọc | Hoà | 100 |
| 2 | 2211327 | Hồ Nguyễn Phi | Hùng | 100 |
| 3 | 2211911 | Võ Tá Bảo | Long | 100 |
| 5 | 2011867 | Lê Quang | Phục | 100 |

*TP. Hồ Chí Minh, năm 2023*

**Xác định tag, index, offset**

**Bài 1:**

Không gian địa chỉ là 32 bit

* Direct mapped:

Byte offset: 8 bits

Index: 14 bits

Tag: 32 – 14 – 8 = 10 bits

* 4-way associative:

Byte offset: 8 bits

Index: => 12 bits (Vì 4 blocks tạo thành 1 set nên có 214 blocks sẽ tạo thành 212 sets)

Tag: 32 – 8 – 12 = 12 bits

* Fully associative:

Half-word offset: 8 bits

Index: 0 bit

Tag: 32 – 8 = 24 bits

**Bài 2:**

Không gian địa chỉ là 256MB = 228, nên ta sẽ dùng thanh ghi 28 bit tính theo byte offset để lưu trữ. => half-word offset có 27 bits

* Direct mapped:

Half-word offset: 7 bits

Index: 10 bits

Tag: 27 – 10 – 7 = 10 bits

* 4-way set associative:

Half-word offset: 7 bits

Index: => 8 bits (Vì 4 blocks tạo thành 1 set nên có 210 blocks sẽ tạo thành 28 sets)

Tag: 27 – 8 – 7 = 12 bits

* Fully associative:

Half-word offset: 7 bits

Index: 0 bit

Tag: 27 – 7 = 20 bits

**Xác định HIT/MISS**

**Bài 3:**

Ta tính được có 4 bits byte-offset, 4 bits index, block size = 16 bytes

* Direct mapped: 16 sets và mỗi block 16 bytes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Address words (byte) | Address/blocksize = A | Tag = A / 16 | Index = A % 16 | H/M | Giải thích |
| 0 (0) | 0 | 0 | 0 | M | First access |
| 4 (16) | 1 | 0 | 1 | M | First access |
| 1 (4) | 0 | 0 | 0 | H |  |
| 5 (20) | 1 | 0 | 1 | H |  |
| 65 (260) | 16 | 1 | 0 | M | Khác tag |
| 1 (4) | 0 | 0 | 0 | M | Khác tag |
| 67 (268) | 16 | 1 | 0 | M | Khác tag |
| 46 (184) | 11 | 0 | 11 | M | First access |
| 1 (4) | 0 | 0 | 0 | M | Khác tag |
| 70 (280) | 17 | 1 | 1 | M | Khác tag |
| 2 (8) | 0 | 0 | 0 | H |  |
| 0 (0) | 0 | 0 | 0 | H |  |

* 4-way associative: 4 sets và mỗi block 16 bytes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Address words (byte) | Address/blocksize = A | Tag = A / 4 | Index = A % 4 | H/M | Giải thích |
| 0 (0) | 0 | 0 | 0 | M | First access |
| 4 (16) | 1 | 0 | 1 | M | First access |
| 1 (4) | 0 | 0 | 0 | H |  |
| 5 (20) | 1 | 0 | 1 | H |  |
| 65 (260) | 16 | 4 | 0 | M | First access |
| 1 (4) | 0 | 0 | 0 | H |  |
| 67 (268) | 16 | 4 | 0 | H |  |
| 46 (184) | 11 | 2 | 3 | M | First access |
| 1 (4) | 0 | 0 | 0 | H |  |
| 70 (280) | 17 | 4 | 1 | M | First access |
| 2 (8) | 0 | 0 | 0 | H |  |
| 0 (0) | 0 | 0 | 0 | H |  |

* Fully associative:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address : words (byte) | Address/blocksize = A | Tag = A | H/M | Giải thích |
| 0 (0) | 0 | 0 | M | First access |
| 4 (16) | 1 | 1 | M | First access |
| 1 (4) | 0 | 0 | H |  |
| 5 (20) | 1 | 1 | H |  |
| 65 (260) | 16 | 16 | M | First access |
| 1 (4) | 0 | 0 | H |  |
| 67 (268) | 16 | 16 | H |  |
| 46 (184) | 11 | 11 | M | First access |
| 1 (4) | 0 | 0 | H |  |
| 70 (280) | 17 | 17 | M | First access |
| 2 (8) | 0 | 0 | H |  |
| 0 (0) | 0 | 0 | H |  |

Tính thời gian truy xuất trung bình (AMAT)

**Bài 4:**

Hit time = 5 cycles, thời gian truy xuất RAM là 10ns, tần số máy tính là 2GHz.

Tần số = 2GHz -> Chu kỳ = 0.5 ns

=> Thời gian truy xuất Ram = = 20 chu kỳ = Miss penalty

AMAT = Hit time + Miss rate x Miss penalty

* Direct mapped:

AMAT (cycles) = chu kỳ

AMAT (time) = = 9.165 ns

* 4-way associative:

AMAT (cycles) = chu kỳ

AMAT (time) = = 6.665 ns

* Fully associative:

AMAT (cycles) = chu kỳ

AMAT (time) = = 6.665 ns

**Bài 5:**

HTL1 = 10 cycles, HTL2 = 15 cycles, RAM access time = 100 cycles, MRL1 = 20%. MRL2 = 10%

AMAT (cycles) = HTL1 + MRL1 x (HTL2 + MRL2 x RAM access time)

= 10 + 0.2x(15 + 0.1x100)

= 15 chu kỳ

**Tính CPI trung bình**

**Bài 6:**

I-cache MR = 5%, D-cache MR = 10%, IC = 1000, 100 lệnh load/store, Miss penalty = 100 cycles

CPI = I-cache MR x Miss penalty + LS Frequency x D-cache MR x Miss penalty

= 0.05 x 100 + 0.1 x 0.1 x 100

= 6