

การทดลองที่ A

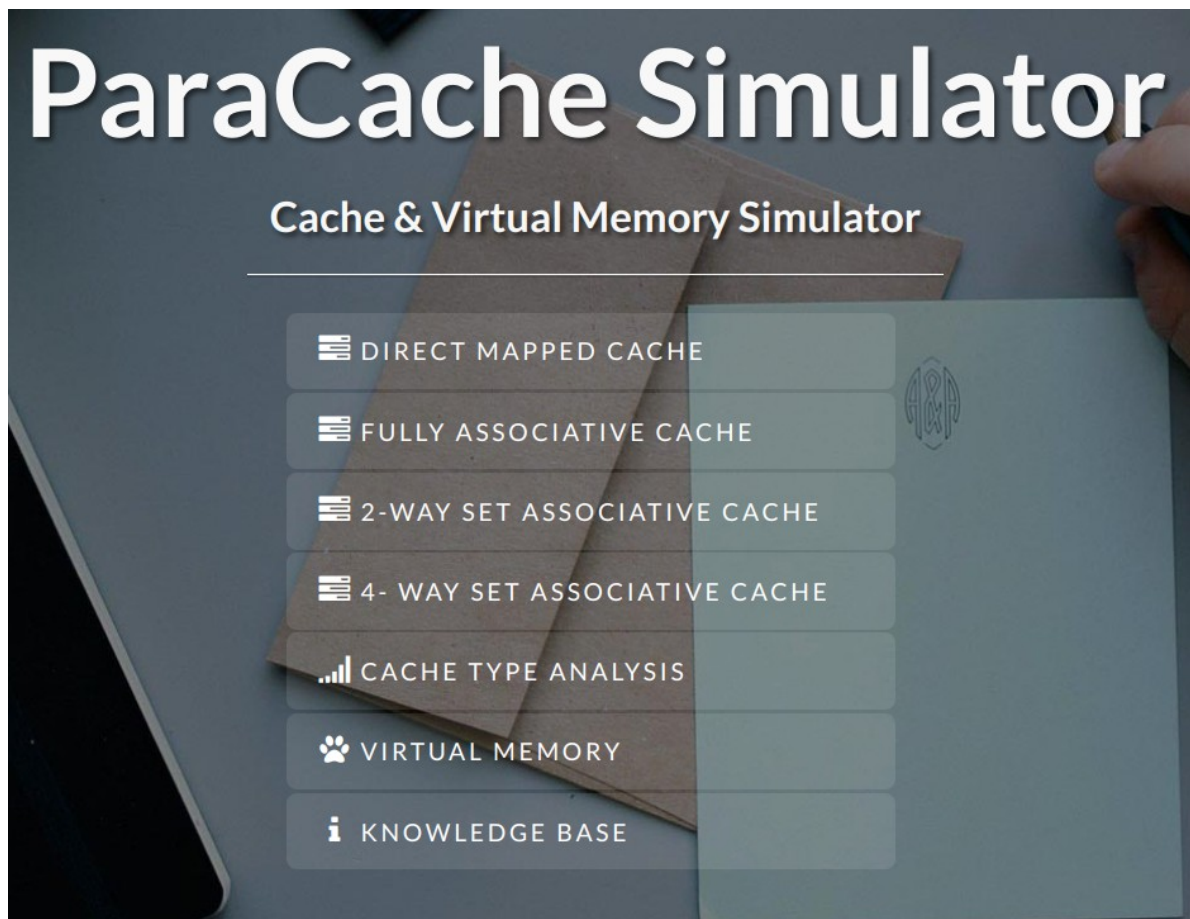
การทำงานของแคชชนิด Direct Mapped

วิชา Computer Organization and Assembly Language

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใช้เว็บเบราว์เซอร์เปิดใช้งานซิมูเลเตอร์ ชื่อ Para Cache

<https://www3.ntu.edu.sg/home/smitha/ParaCache/Paracache/start.html>

เอกสารอธิบาย

<https://www3.ntu.edu.sg/home/smitha/ParaCache/Paracache/kb.pdf>

ทำการทดลอง ตามขั้นตอนต่อไปนี้

1. การทดลอง Direct Mapped Cache

กดเมนู เลือก Direct Mapped Cache ตั้งขนาดและ Write Policy ของแคช ดังรูป

ParaCache

Write Policies

☒ **Write Back** ☐ **Write Through**
☒ **Write On Allocate** ☐ **Write Around**

Cache Size (power of 2)
Memory Size (power of 2)
Offset Bits

2. กด Submit แล้วสังเกตรายละเอียดของแคชที่อยู่ด้านขวา

DIRECT MAPPED CACHE

➡ Instruction Breakdown

TAG	INDEX	OFFSET
2 bit	2 bit	0 bit

☐ Memory Block

B. A W. 0
B. B W. 0
B. C W. 0
B. D W. 0
B. E W. 0
B. F W. 0

☐ Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0

เลื่อนหน้าต่างลงไปด้านล่างสุดของ Memory Block โปรดสังเกตหมายเลขบล็อก (B.) มีค่าเท่ากับ 0 ถึง F และหมายเลขเวิร์ด (W.) เท่ากับ 0 เสมอ

☐ Memory Block

B. A W. 0
B. B W. 0
B. C W. 0
B. D W. 0
B. E W. 0
B. F W. 0

เพราะเหตุใด

เนื่องจาก B.(Block) ขึ้นอยู่กับ Memory Size และ W. (word) ขึ้นอยู่กับ Offset bits

3. การทดลองคำสั่ง Load Instruction ที่หมายเลขแอดเดรสที่ต้องการ หรือ ให้โปรแกรมสุ่มหมายเลขแอดเดรสให้
 กรอก 4 ลงในหมายเลขฐานสิบหกที่มีอยู่ในกล่องข้อความด้านขวา
 กรอกหมายเลข 7, 7, c, 4, 0, 4, 3, 5, 5 ในกล่องข้อความดังรูป

Instruction

Load

(in hex)#

4

7,7,c,4,0,4,3,5,5

Gen. Random

Submit

Information

Offset = 0 bits

Index bits = $\log_2(4/1) = 2$ bits

Instruction Length = $\log_2(16) = 4$ bits

Tag = 4 bits - 0 bits - 2 bits = 2

Next

Fast Forward

อธิบาย information ในรูปว่า Tag, Index และ Offset สัมพันธ์กับ Cache Size และ Memory Size ที่กรอก

Cache Size มีผลต่อ index

Memory size มีผลต่อ Instruction Length ซึ่งจะนำ cache และ memory มาคำนวณหา Tag

4. กดปุ่ม Submit หมายเลข 4 ที่กรอก โปรแกรมสังเกตและอ่านกล่องข้อความที่เป็นสีชมพู อธิบายตามความเข้าใจ

หมายเลข 4 จะถูกเปลี่ยนจาก Hex เป็น Binary แล้วแบ่งใน tag, index, offset

5. กดปุ่ม Next และสังเกตกล่องข้อความที่เปลี่ยนเป็นสีเหลืองว่าเกี่ยวข้องกับหมายเลขที่ Submit ไปก่อนหน้านี้อย่างไร

➡ Instruction Breakdown

01	00	0
2 bit	2 bit	0 bit

☐ Memory Block

B. A W. 0
B. B W. 0
B. C W. 0
B. D W. 0
B. E W. 0
B. F W. 0

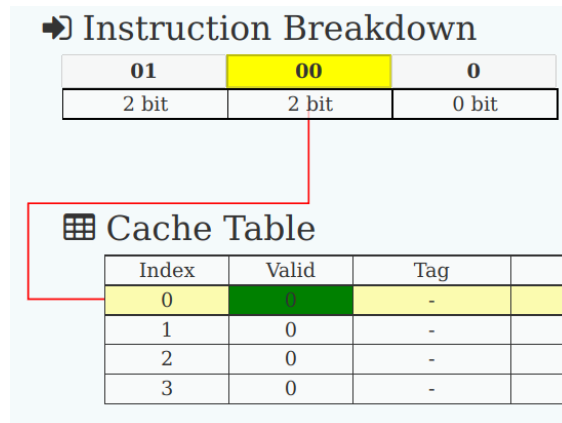
☐ Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0

อธิบายความสัมพันธ์ระหว่าง Instruction Breakdown 01 00 และหมายเลข 4

0100 คือ เลขฐานสอง ซึ่งในเลขฐาน 10 จะได้เท่ากับ 4 ซึ่งแบ่งออกเป็น tag = 10 และ index = 00

6. กดปุ่ม Next และสังเกตกล่องข้อความที่เปลี่ยนเป็นสีเขียว

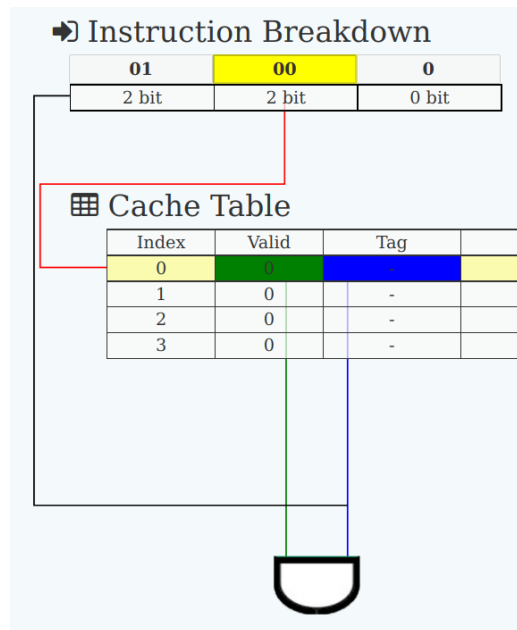


อธิบายรูปนี้ และบิต Valid จึงเป็น 0

ไม่มี value เก็บไว้ใน Cache

7. กดปุ่ม Next และสังเกตกล่องข้อความที่เปลี่ยนเป็นน้ำเงิน และ AND เกตว่าทำกระบวนการอะไรกัน

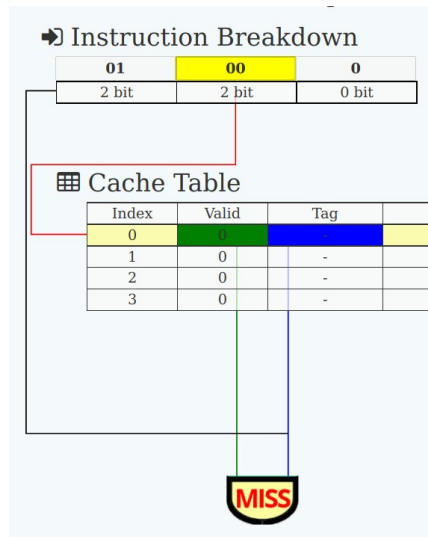
ทำการตรวจสอบ 2 bit แรก



อธิบายว่า Tag จึงมีสัญลักษณ์ '-'

ไม่มี value เก็บไว้ใน Cache

8. กดปุ่ม Next เพื่อดำเนินการต่อ โปรดสังเกตข้อความบน AND เกต



กดปุ่ม Next เพื่อดำเนินการต่อ โปรดสังเกต Cache Table ว่ามีการเปลี่ยนแปลงอย่างไร

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	1	01	BLOCK 4 WORD 0 - 0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0

อธิบาย Valid Tag และ Data (Hex) จึงเปลี่ยนเป็นรูปนี้

Valid Tag บ่งบอกว่า Index ที่ 0 มีค่า value เก็บอยู่แล้ว, Data เป็นการเก็บค่ามาจาก Block ที่ 4 จาก Main memory

9. กดปุ่ม Next เพื่อดำเนินการต่อ โปรดสังเกตข้อมูลสถิติสี่เหลี่ยมด้านล่าง

Statistics	
Hit Rate :	0%
Miss Rate :	100%
List of Previous Instructions :	
• Load 4 [Miss]	

อธิบายข้อมูลที่ได้

ค่าใน cache ไม่ตรงกับ request เลขหรือ miss ทั้งหมด
หลังจากนี้จะเก็บค่า value ใน cache

10. โปรดสังเกตหมายเลขแอดเดรสถัดไปจะย้ายมาในกล่องข้อความด้านขวาบนของรูปนี้ กดปุ่ม Submit

11. กดปุ่ม Fast Forward เพื่อเร่งการทำงานของคำสั่งให้รวดเร็วขึ้น โปรดสังเกตการเปลี่ยนแปลงใน Cache Table และ Statistics หลังจากนั้น

01	11	0
2 bit	2 bit	0 bit

Memory Block

Cache Table

Index	Valid	Tag	Data (Hex)
0	1	01	BLOCK 4 WORD 0 - 0
1	0	-	0
2	0	-	0
3	1	01	BLOCK 7 WORD 0 - 0

Statistics

Hit Rate : 0%

Miss Rate : 100%

List of Previous Instructions :

- Load 4 [Miss]
- Load 7 [Miss]

12. กด Submit และ Fast Forward เรื่อยๆ จนไม่เหลือหมายเลขแอดเดรส โปรดสังเกตการเปลี่ยนแปลงใน Statistics หลังจากนั้น

Instruction

Load
▼
(in hex)#

List of next 10 Instructions

Gen. Random
Submit

Information

The cycle has been completed.
Please submit another instructions

Next
Fast Forward

Statistics

Hit Rate : 20%

Miss Rate : 80%

List of Previous Instructions :

- Load 4 [Miss]
- Load 7 [Miss]
- Load 7 [Hit]
- Load C [Miss]
- Load 4 [Miss]
- Load 0 [Miss]
- Load 4 [Miss]
- Load 3 [Miss]
- Load 5 [Miss]
- Load 5 [Hit]

อธิบายข้อมูลที่ได้ว่า Hit Rate และ Miss Rate คำนวณอย่างไร

Hit rate คือ $\frac{\text{จำนวนครั้งที่ Hit}}{\text{จำนวนข้อมูล request ทั้งหมด}}$, Miss Rate คือ $\frac{\text{จำนวนครั้งที่ Miss}}{\text{จำนวนข้อมูล request ทั้งหมด}} = 1 - \text{Hit rate}$

นักศึกษาควรจะได้ผลการทดลองใน Cache Table ตรงกับรูปนี้

Cache Table

Index	Valid	Tag	Data (Hex)
0	1	01	BLOCK 4 WORD 0 - 0
1	1	01	BLOCK 5 WORD 0 - 0
2	0	-	0
3	1	00	BLOCK 3 WORD 0 - 0

13. กรอกหมายเลขบล็อกที่แคชมีอยู่ เพื่อจงใจให้เกิด แคชฮิต ดังรูป

Instruction

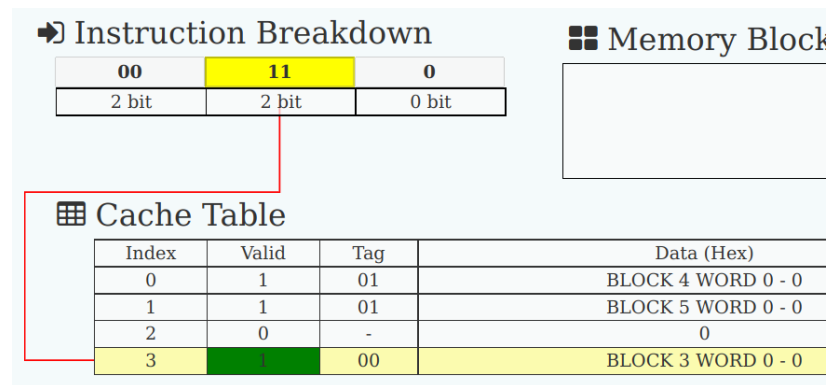
Load (in hex)#

3, 4, 5

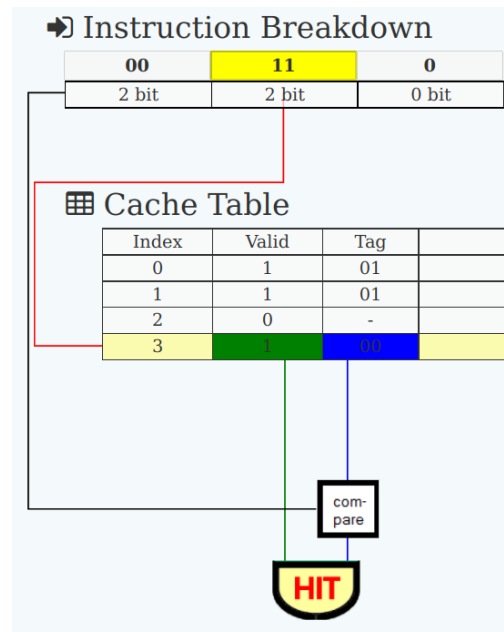
Gen. Random

Submit

กด Submit และ Next จนได้เหตุการณ์นี้



โปรดสังเกตคอลัมน์ Valid และ Tag ว่าตรงกันหรือไม่ **ตรงกัน**

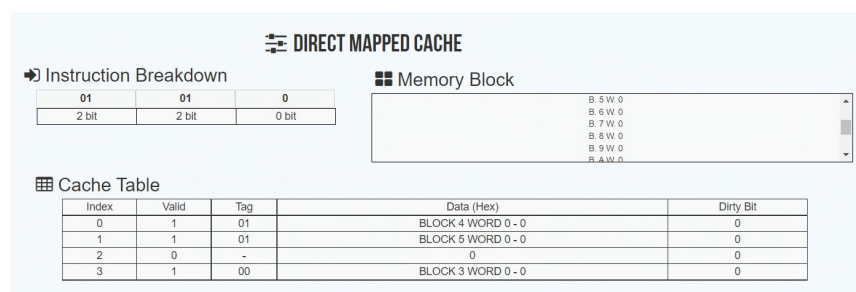


14. กดปุ่ม Submit หมายเลขถัดไปจนหมด และแนบบรูปตาราง Statistics ว่ามีการเปลี่ยนแปลงหรือไม่ อย่างไร

Statistics	
Hit Rate :	38%
Miss Rate :	62%
List of Previous Instructions :	
• Load 4 [Miss]	
• Load 7 [Miss]	
• Load 7 [Hit]	
• Load C [Miss]	
• Load 4 [Miss]	
• Load 0 [Miss]	
• Load 4 [Miss]	
• Load 3 [Miss]	
• Load 5 [Miss]	
• Load 5 [Hit]	
• Load 3 [Hit]	
• Load 4 [Hit]	
• Load 5 [Hit]	

∴ ค่า Hit rate เปลี่ยนแปลงจาก 20 % เป็น 38 %

ค่า Miss rate เปลี่ยนแปลงจาก 80 % เป็น 62 %



กิจกรรมท้ายการทดลอง

- ศึกษาการทำงานของ Load Instruction เช่นเดิม
 - ตั้งขนาดของแคชเท่ากับ 8 และ Memory Size เท่ากับ 16 OFFSET = 0 บิต แล้วเปรียบเทียบ
 - ตั้งขนาดของแคชเท่ากับ 4 และ Memory Size เท่ากับ 16 OFFSET = 1 บิต แล้วเปรียบเทียบ
- ศึกษาการทำงานของ Load Instruction เช่นเดิม แต่ตั้ง Write Policy เป็น Write Through และ Write Around
 - ตั้งขนาดของแคชเท่ากับ 4 และ Memory Size เท่ากับ 16 OFFSET = 0 บิต
 - ตั้งขนาดของแคชเท่ากับ 8 และ Memory Size เท่ากับ 16 OFFSET = 0 บิต
 - ตั้งขนาดของแคชเท่ากับ 4 และ Memory Size เท่ากับ 16 OFFSET = 1 บิต
- เปลี่ยน Instruction เป็น Store เพื่อศึกษาการทำงานของ Dirty Bit โดยตั้ง Write Policy เป็น Write Back และ Write on Allocate
 - ตั้งขนาดของแคชเท่ากับ 4 และ Memory Size เท่ากับ 16 OFFSET = 0 บิต
 - ตั้งขนาดของแคชเท่ากับ 8 และ Memory Size เท่ากับ 16 OFFSET = 0 บิต
 - ตั้งขนาดของแคชเท่ากับ 4 และ Memory Size เท่ากับ 16 OFFSET = 1 บิต
- เปลี่ยน Instruction เป็น Store เพื่อศึกษาการทำงานของ Dirty Bit โดยตั้ง Write Policy เป็น Write Through และ Write Around
 - ตั้งขนาดของแคชเท่ากับ 4 และ Memory Size เท่ากับ 16 OFFSET = 0 บิต
 - ตั้งขนาดของแคชเท่ากับ 8 และ Memory Size เท่ากับ 16 OFFSET = 0 บิต
 - ตั้งขนาดของแคชเท่ากับ 4 และ Memory Size เท่ากับ 16 OFFSET = 1 บิต

① ส่วนที่มรทั้ง ๘ Cache และ Memory Size จึง Cache Table มีนกาสักลงและเก็บ cache ๓๓๓๓๓๓ #

1.1)

Statistics

Hit Rate : 30%

Miss Rate : 70%

List of Previous Instructions :

- Load 4 [Miss]
- Load 7 [Miss]
- Load 7 [Hit]
- Load C [Miss]
- Load 4 [Miss]
- Load 0 [Miss]
- Load 4 [Hit]
- Load 3 [Miss]
- Load 5 [Miss]
- Load 5 [Hit]

1.2)

Statistics

Hit Rate : 30%

Miss Rate : 70%

List of Previous Instructions :

- Load 4 [Miss]
- Load 7 [Miss]
- Load 7 [Hit]
- Load C [Miss]
- Load 4 [Miss]
- Load 0 [Miss]
- Load 4 [Miss]
- Load 3 [Miss]
- Load 5 [Hit]
- Load 5 [Hit]

DIRECT MAPPED CACHE

Instruction Breakdown

0	101	0
1 bit	3 bit	0 bit

Memory Block

B 5 W 0
B 6 W 0
B 7 W 0
B 8 W 0
B 9 W 0
B A W 0

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	1	0	BLOCK 0 WORD 0 - 0	0
1	0	-	0	0
2	0	-	0	0
3	1	0	BLOCK 3 WORD 0 - 0	0
4	1	0	BLOCK 4 WORD 0 - 0	0
5	1	0	BLOCK 5 WORD 0 - 0	0
6	0	-	0	0
7	1	0	BLOCK 7 WORD 0 - 0	0

DIRECT MAPPED CACHE

Instruction Breakdown

01	0	1
2 bit	1 bit	1 bit

Memory Block

B 2 W 0
B 3 W 0
B 4 W 0
B 5 W 0
B 6 W 0
B 7 W 0

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	1	01	BLOCK 2 WORD 0 - 1	0
1	1	00	BLOCK 1 WORD 0 - 1	0