

### การทดลองที่ A

การทำงานของแคชชนิด Direct Mapped

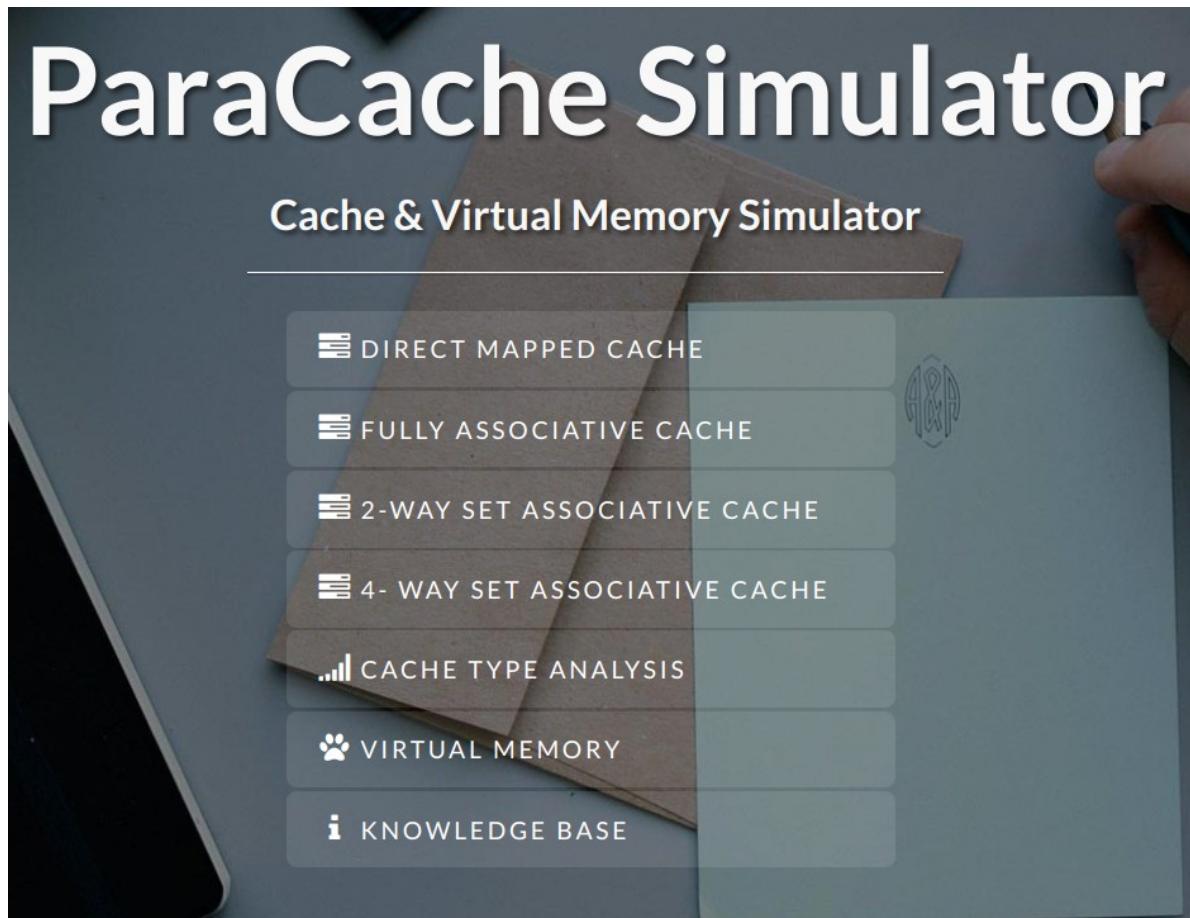
วิชา Computer Organization and Assembly Language

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใช้เว็บเบราว์เซอร์เปิดใช้งานชิมูเลเตอร์ 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**     **Write Around**

---

**Allocate**

**Cache Size (power of 2)**

**Memory Size (power of 2)**

**Offset Bits**

**Reset**    **Submit**

1 block คือ 1 word

2. กด Submit และสังเกตรายละเอียดของแคชที่อยู่ด้านขวา

### DIRECT MAPPED CACHE

<b>Instruction Breakdown</b>			<b>Memory Block</b>						
<b>TAG</b>	<b>INDEX</b>	<b>OFFSET</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>D. A W. 0</td></tr> <tr><td>B. B W. 0</td></tr> <tr><td>B. C W. 0</td></tr> <tr><td>B. D W. 0</td></tr> <tr><td>B. E W. 0</td></tr> <tr><td>B. F W. 0</td></tr> </table>	D. A W. 0	B. B W. 0	B. C W. 0	B. D W. 0	B. E W. 0	B. F W. 0
D. A W. 0									
B. B W. 0									
B. C W. 0									
B. D W. 0									
B. E W. 0									
B. F W. 0									
2 bit	2 bit	0 bit							

<b>Cache Table</b>					
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 โปรดสังเกตหมายเลขล็อก (B.) มีค่าเท่ากับ 0 ถึง F และหมายเลขเวิร์ด (W.) เท่ากับ 0 เมมโม

### Memory Block

D. A W. 0 1 word = 4 byte  
 B. B W. 0  
 B. C W. 0  
 B. D W. 0  
 B. E W. 0  
 B. F W. 0 1 Word

เพราะเหตุใด

B. (Block) นั้นอยู่กับ Memory Size

W. (Word) นั้นอยู่กับ Offset bits

- LDR**
3. การทดลองคำสั่ง Load Instruction ที่หมายเลขแอดเดรสที่ต้องการ หรือ ให้โปรแกรมสุ่มหมายเลขแอดเดรสให้กรอก 4 ลงในหมายเลขฐานสิบหกที่มีอยู่ในกล่องข้อความด้านขวากรอกหมายเลข 7, 7, c, 4, 0, 4, 3, 5, 5 ในกล่องข้อความดังรูป Address word

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 สมพันธ์กับ Cache Size และ Memory Size ที่กรอก

### Cache Size และ Index

### Memory Size และ Instruction Length ที่ Cache และ Memory คำนวณต่ำ Tag

4. กดปุ่ม Submit หมายเลข 4 ที่กรอก โปรดสังเกตและอ่านกล่องข้อความที่เป็นสีเข้มพู อธิบายตามความเข้าใจ

เงน 4 เป็น binary ที่ hex หนึ่งตัว tag, index, offset

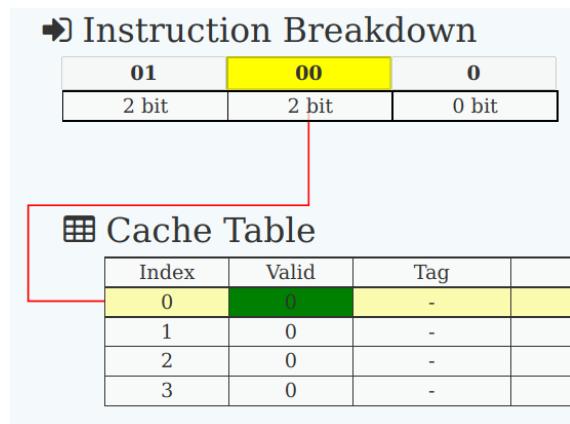
5. กดปุ่ม Next และสังเกตกล่องข้อความที่เปลี่ยนเป็นสีเหลืองว่าเกี่ยวข้องกับหมายเลขที่ Submit ไปก่อนหน้านี้อย่างไร

➔ Instruction Breakdown	■ Memory Block			
$A = 0100_2$ <p>01 tag      00 index      0 offset</p> <p>2 bit      2 bit      0 bit</p>	<b>Memory Block</b> <ul style="list-style-type: none"> <li>D. A W. 0</li> <li>B. B W. 0</li> <li>B. C W. 0</li> <li>B. D W. 0</li> <li>B. E W. 0</li> <li>B. F W. 0</li> </ul>			
■ Cache Table				
Index	Valid	Tag	Data (Hex)	Dirty Bit
0 00	0	-	0	0
1 01	0	-	0	0
2 10	0	-	0	0
3 11	0	-	0	0

อธิบายความสัมพันธ์ระหว่าง Instruction Breakdown 01 00 และหมายเลข 4

$(0100)_2 \Rightarrow (4)_{16}$  หนึ่งปุ่ม tag=10 หนึ่งปุ่ม index=00

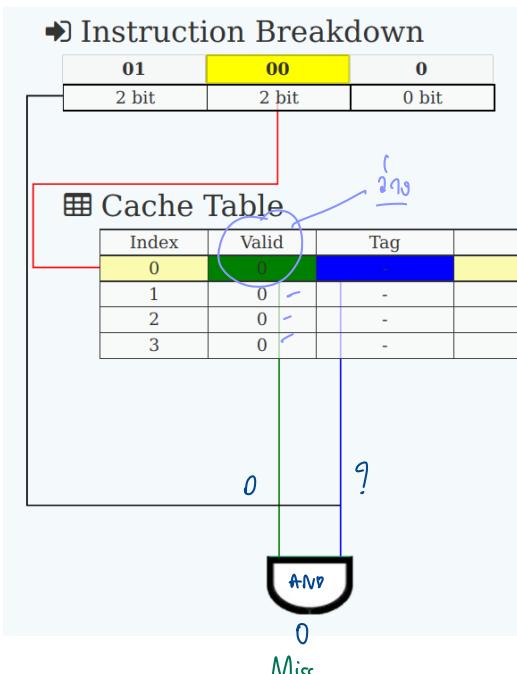
6. กดปุ่ม Next และสังเกตกล่องข้อความที่เปลี่ยนเป็นสีเขียว



อธิบายรูปนี้ และบิต Valid จึงเป็น 0

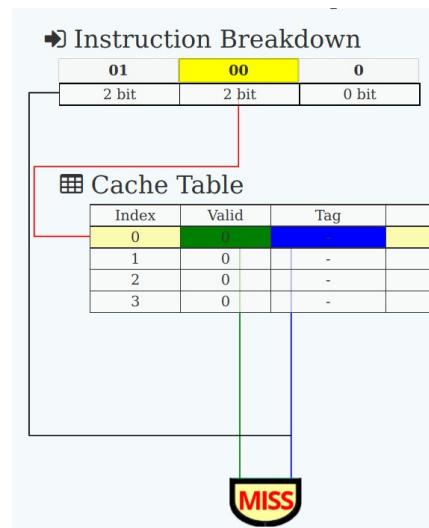
ไม่มี value เก็บใน Cache

7. กดปุ่ม Next และสังเกตกล่องข้อความที่เปลี่ยนเป็นน้ำเงิน และ AND เกตว่าทำกระบวนการอะไรกัน

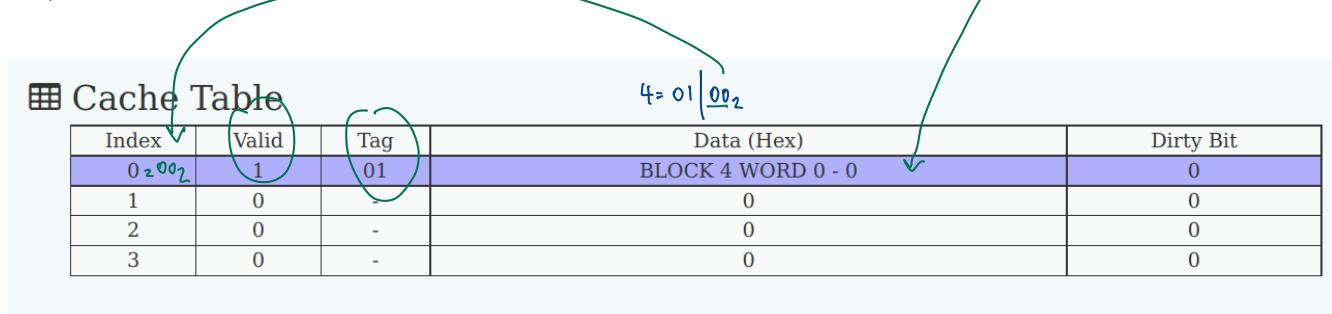


หา value เก็บใน Cache

8. กดปุ่ม Next ต่อเพื่อดำเนินการต่อ โปรดสังเกตข้อความบน AND เกต



กดปุ่ม Next เพื่อดำเนินการต่อ โปรดสังเกต Cache Table ว่ามีการเปลี่ยนแปลงอย่างไร



อธิบายบิต Valid Tag และ Data (Hex) จึงเปลี่ยนเป็นรูปนี้

Valid Tag ของว่า index ที่ 0 มี value เก็บอยู่แล้ว, Data เก็บค่าจาก Block ที่ 4 ใน main memory

9. กดปุ่ม Next เพื่อดำเนินการต่อ โปรดสังเกตข้อมูลสถิติสีเหลืองด้านล่าง

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

อธิบายข้อมูลที่ได้

ค่าใน Cache จะมีตัวไนน์ตรงกับที่ request เดียวแต่ที่ต้อง หรือ miss ห้ามขาด

หมายเหตุ value ที่ Cache

10. โปรดสังเกตหมายเลขแอดเดรสสดที่เปลี่ยนมาในกล่องข้อความด้านขวาบนของรูปนี้ กดปุ่ม Submit  $7 = 0111_2$  [index  $1_2 = 3$ ]
11. กดปุ่ม Fast Forward เพื่อเร่งการทำงานของคำสั่งให้รวดเร็วขึ้น โปรดสังเกตการเปลี่ยนแปลงใน Cache Table และ Statistics หลังจากนั้น

<b>01</b>	<b>11</b>	<b>0</b>
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

(in hex) #

List of next 10 Instructions

Information

The cycle has been completed.  
Please submit another instructions

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 จำนวนครั้งที่ Hit / จำนวนทั้งหมด request ทั้งหมด คิดเป็น  $2/10 = 0.2 \text{ หรือ } 20\%$

Miss rate คือ จำนวนครั้งที่ Miss / จำนวนทั้งหมด request ทั้งหมด =  $1 - \text{Hit rate}$

นักศึกษาควรจะได้ผลการทดลองใน Cache Table ตรงกับรูปนี้

$$\begin{aligned} &= 1 - 0.2 \\ &= 0.8 \text{ หรือ } 80\% \end{aligned}$$

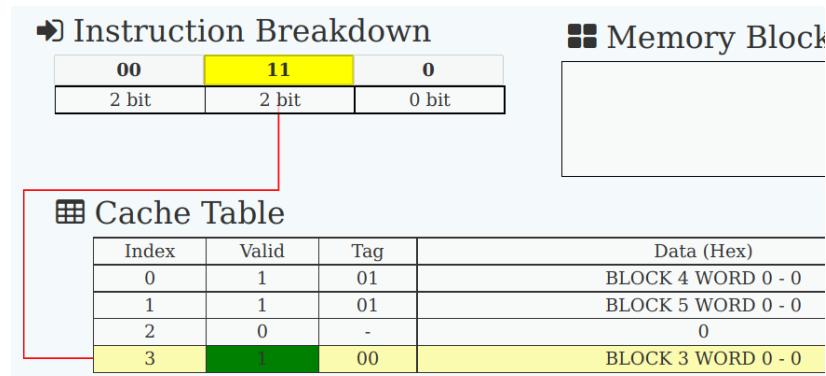
## Cache Table

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

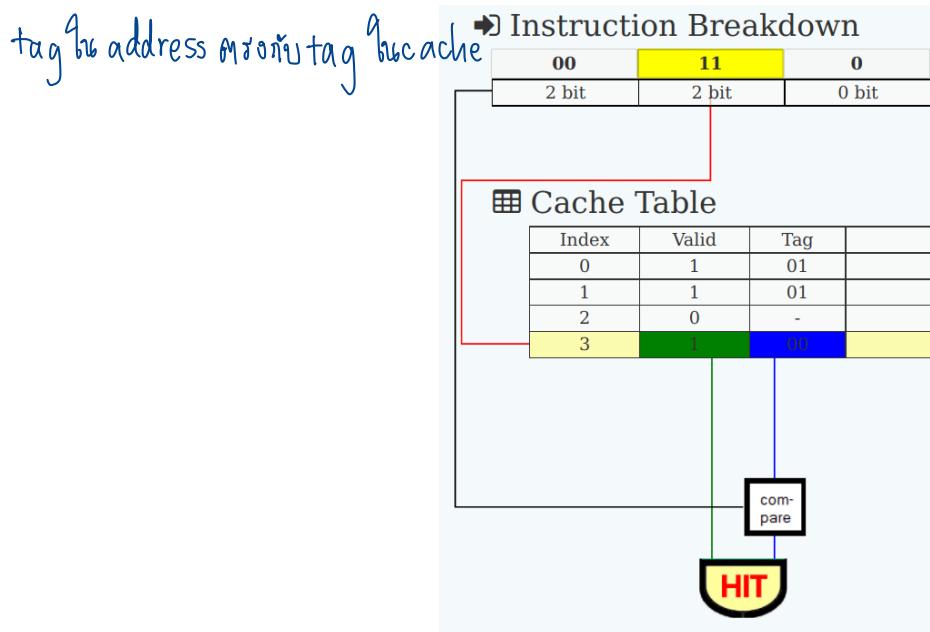
13. กรอกหมายเลขล็อกที่แคชมีอยู่ เพื่อจะได้เกิด แคชชิต ดังรูป

Instruction
Load <input type="button" value="▼"/> (in hex) # <input type="text"/>
3, 4, 5
<input type="button" value="Gen. Random"/> <input type="button" value="Submit"/>

กด Submit และ Next จะได้เหตุการณ์นี้



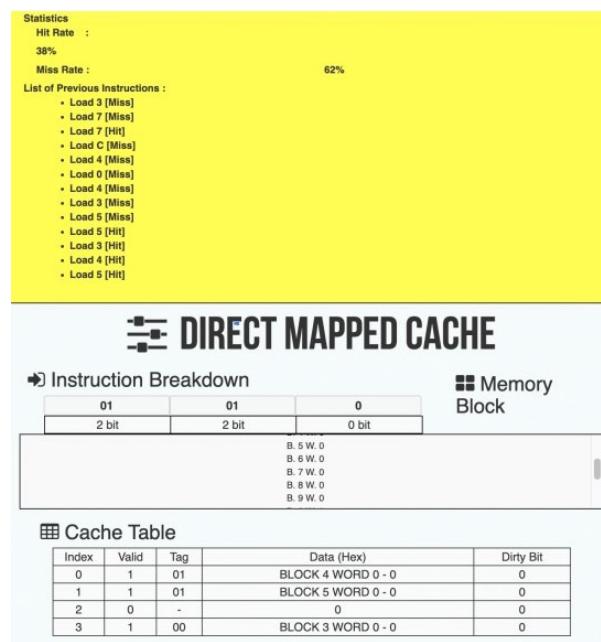
โปรดสังเกตคอลัมน์ Valid และ Tag ว่าตรงกันหรือไม่



14. กดปุ่ม Submit หมายเลขอัตโนมัติไปจนหมด และแนบรูปตาราง Statistics ว่ามีการเปลี่ยนแปลงหรือไม่ อย่างไร

Hit rate ณ 20%  $\Rightarrow$  38%

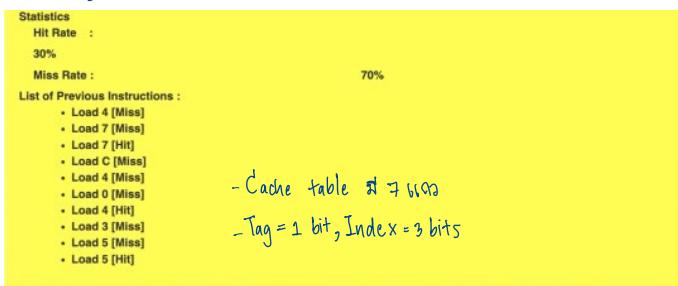
Miss rate ณ 80%  $\Rightarrow$  62%



### กิจกรรมท้ายการทดลอง

1. 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
2. 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
3. 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
4. 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

### 1.1.)



### DIRECT MAPPED CACHE

#### Instruction Breakdown

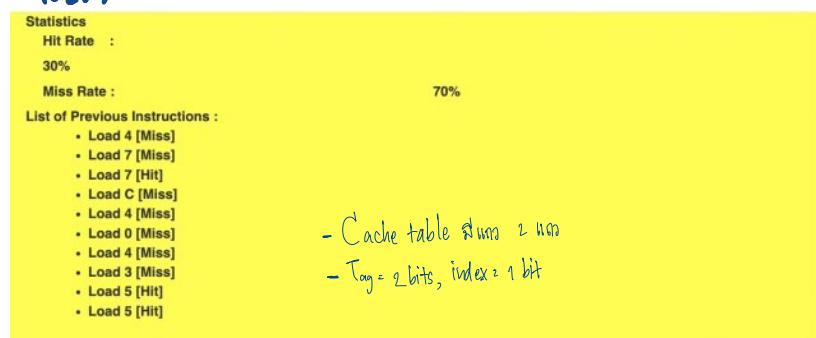
Tag	Index	0	Memory Block
1 bit	3 bit	0 bit	

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	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

### 1.2.)



### DIRECT MAPPED CACHE

#### Instruction Breakdown

01	0	1	Memory Block
2 bit	1 bit	1 bit	
B. 2 W. 0	B. 3 W. 1	B. 4 W. 1	
B. 3 W. 0	B. 4 W. 0	B. 5 W. 1	
B. 4 W. 0	B. 5 W. 0	B. 6 W. 1	
B. 5 W. 0	B. 6 W. 0	B. 7 W. 1	
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

หากคราวเปรียบเทียบชุด 1.1 และ ชุด 1.2 มีข้อแตกต่างที่ กรณีที่ Cache และ Memory Size ให้ฟังก์ชันที่ใช้ในการคำนวณ Cache table ของชุด 1.1 มีผลลัพธ์ต่างๆ กัน และเก็บข้อมูลได้มากกว่า ชุด 1.2