

องค์ประกอบของเครื่องคอมพิวเตอร์
และภาษาแอสเซมบลี:
ARM และ RaspberryPi3

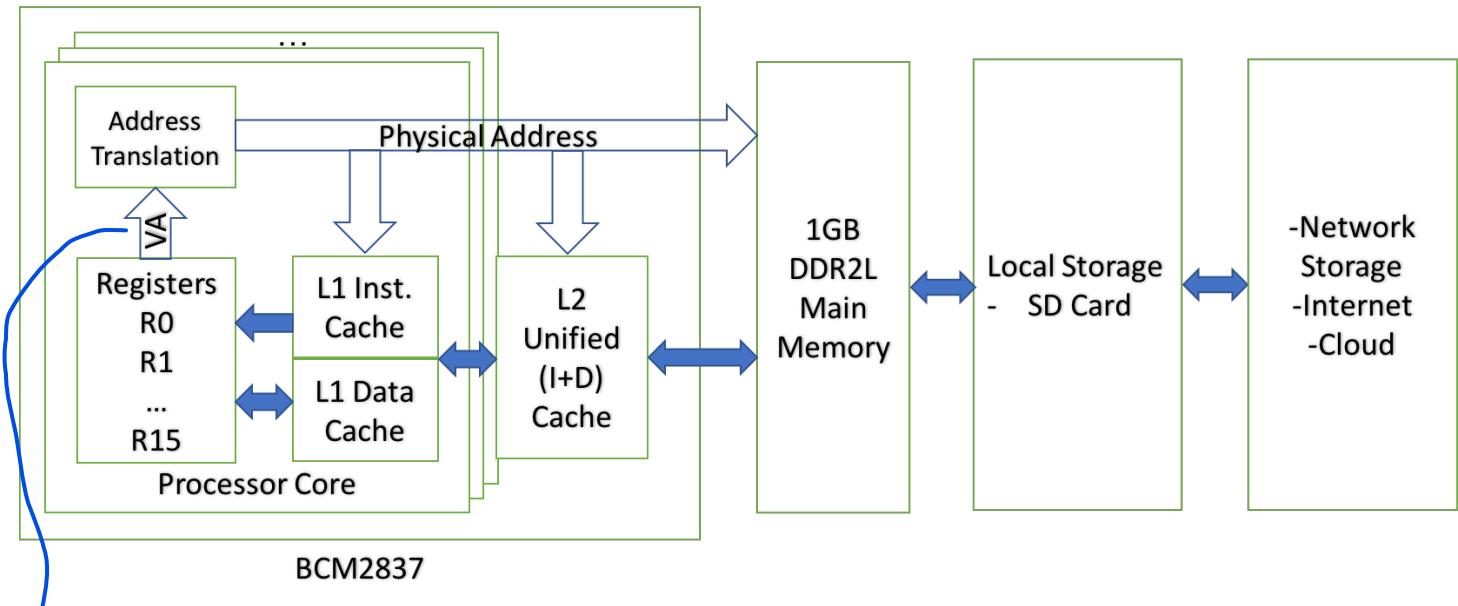
บทที่ 5 ลำดับชั้นของหน่วยความจำ

ผศ.ดร.สุรินทร์ กิตติธรกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

สารบัญ

- 5.1 โครงสร้างของลำดับชั้นหน่วยความจำของบอร์ด Pi3
- 5.2 หน่วยความจำเสมือน (Virtual Memory)
- 5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)
- 5.4 หน่วยความจำชนิด静态ติกแรม (Static RAM: SRAM)
- 5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM)

5.1 โครงสร้างของลำดับชั้นหน่วยความจำของบอร์ด Pi3

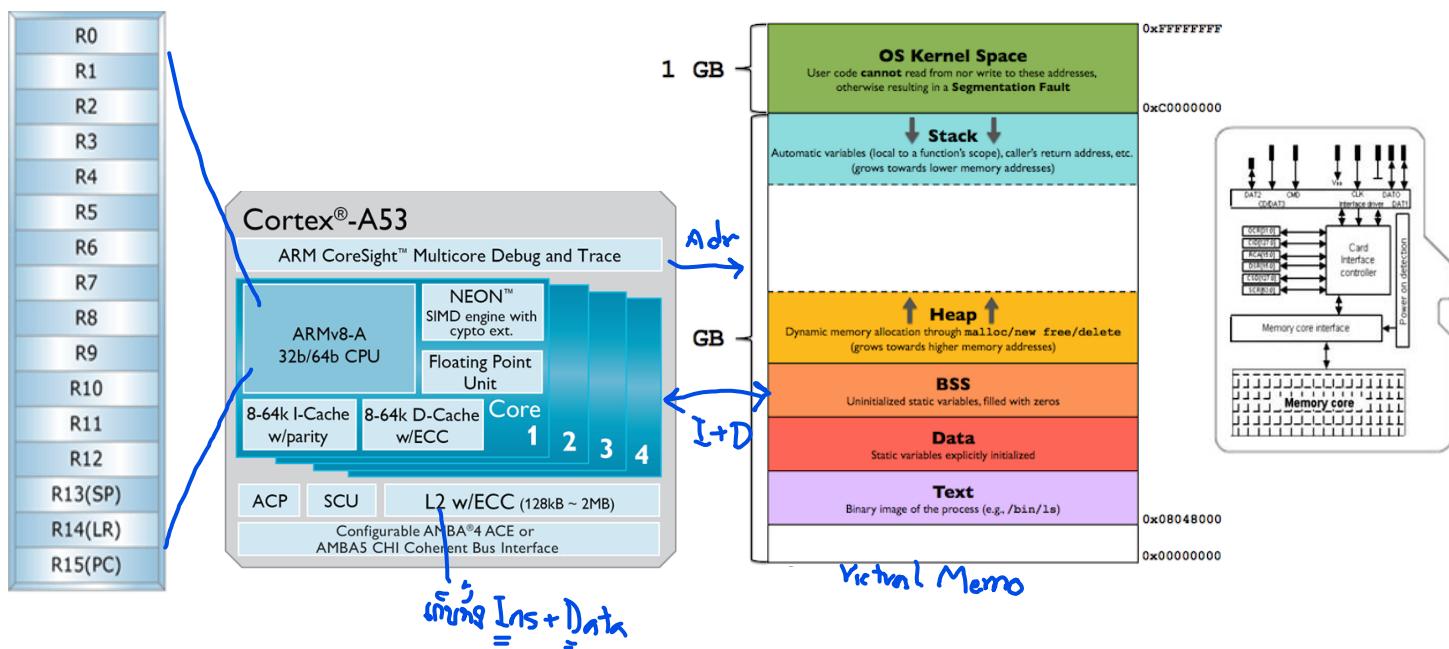


Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

3

Virtual Addr

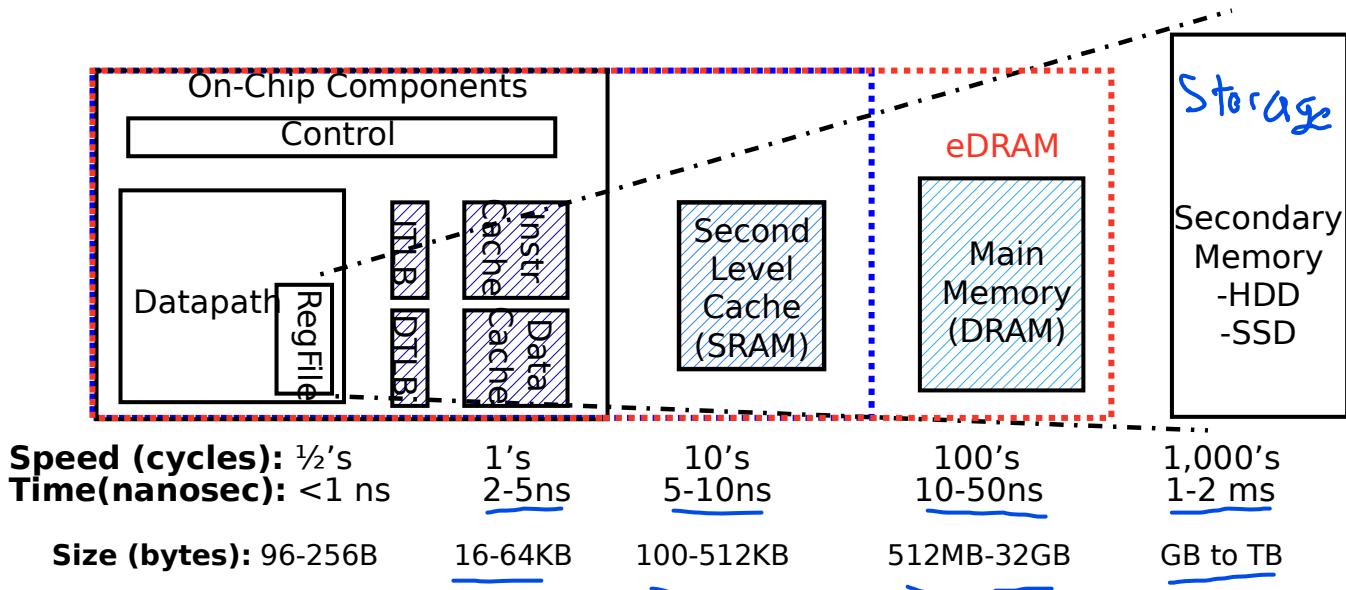
การเชื่อมโยงระหว่างรีจิสเตอร์ แคช หน่วยความจำสามมิติและสตอเรจ



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

4

5.1 โครงสร้างของลำดับชั้นหน่วยความจำของคอมพิวเตอร์



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

5

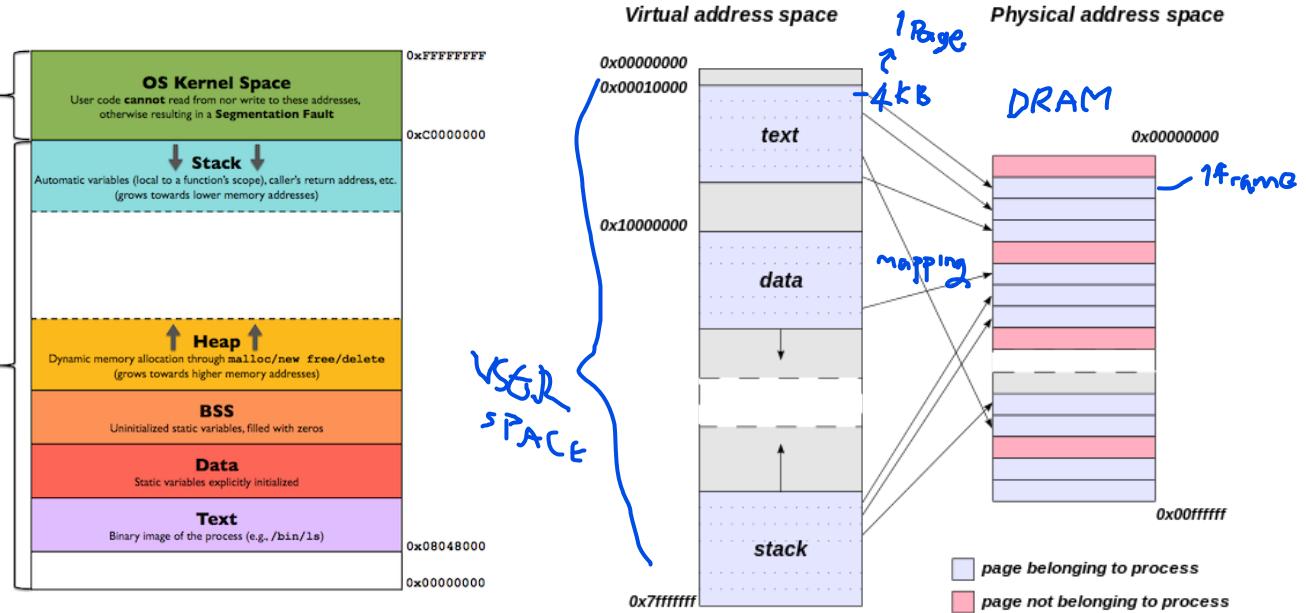
5.1 โครงสร้างของลำดับชั้นหน่วยความจำของคอมพิวเตอร์

Characteristic	DRAM	Static RAM	Flash memory
Static	No	Yes	Yes
Volatile	Yes	Yes	No
Typical size	> 256 Mbits	> 64 Mbits	> 256 Mbits
Organization	4 bits x 64 M	8 bits x 8 M	8 bits x 32 M
Access time	10 ns	< 2 ns	40 ns
Application	Main Memory	Cache Memory	BIOS, Storage

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

6

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

7

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page

Page 0x0000_0

- 0x0000_0000 – 0x0000_0003 คำสั่งที่ 00
- 0x0000_0004 – 0x0000_0007 คำสั่งที่ 01
- 0x0000_0008 – 0x0000_000B คำสั่งที่ 02
- ...
- 0x0000_00FC – 0x0000_00FF คำสั่งที่ 63
- 0x0000_0100 – 0x0000_0103 คำสั่งที่ 64
- 0x0000_0104 – 0x0000_0107 คำสั่งที่ 65
- 0x0000_0108 – 0x0000_010B คำสั่งที่ 66
- ...
- 0x0000_01FC – 0x0000_01FF คำสั่งที่ 127
- ...
- 0x0000_0FFC – 0x0000_0FFF คำสั่งที่ 1023

Page 0x0000_1

- 0x0000_1000 – 0x0000_1003 คำสั่งที่ 1024+00
- 0x0000_1004 – 0x0000_1007 คำสั่งที่ 1024+01
- 0x0000_1008 – 0x0000_100B คำสั่งที่ 1024+02
- ...
- 0x0000_10FC – 0x0000_10FF คำสั่งที่ 1024+63
- 0x0000_1100 – 0x0000_1103 คำสั่งที่ 1024+64
- 0x0000_1104 – 0x0000_1107 คำสั่งที่ 1024+65
- 0x0000_1108 – 0x0000_110B คำสั่งที่ 1024+66
- ...
- 0x0000_11FC – 0x0000_11FF คำสั่งที่ 1024+127
- ...
- 0x0000_1FFC – 0x0000_1FFF คำสั่งที่ 1024+1023

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

8

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page

Page 0x0000_2

- 0x0000_2000 – 0x0000_2003 คำสั่งที่ 2048+00
- 0x0000_2004 – 0x0000_2007 คำสั่งที่ 2048+01
- 0x0000_2008 – 0x0000_200B คำสั่งที่ 2048+02
- ...
- 0x0000_20FC – 0x0000_20FF คำสั่งที่ 2048+63
- 0x0000_2100 – 0x0000_2103 คำสั่งที่ 2048+64
- 0x0000_2104 – 0x0000_2107 คำสั่งที่ 2048+65
- 0x0000_2108 – 0x0000_210B คำสั่งที่ 2048+66
- ...
- 0x0000_21FC – 0x0000_21FF คำสั่งที่ 2048+127
- ...
- 0x0000_2FFC – 0x0000_2FFF คำสั่งที่ 2048+1023

Page 0x0000_3

- 0x0000_3000 – 0x0000_3003 คำสั่งที่ 3096+00
- 0x0000_3004 – 0x0000_3007 คำสั่งที่ 3096+01
- 0x0000_3008 – 0x0000_300B คำสั่งที่ 3096+02
- ...
- 0x0000_30FC – 0x0000_30FF คำสั่งที่ 3096+63
- 0x0000_3100 – 0x0000_3103 คำสั่งที่ 3096+64
- 0x0000_3104 – 0x0000_3107 คำสั่งที่ 3096+65
- 0x0000_3108 – 0x0000_310B คำสั่งที่ 3096+66
- ...
- 0x0000_31FC – 0x0000_31FF คำสั่งที่ 3096+127
- ...
- 0x0000_3FFC – 0x0000_3FFF คำสั่งที่ 3096+1023

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page

- OS ให้พื้นที่ทุกๆ โปรแกรมที่กำลังรันเป็นขนาดเท่ากัน ตามจำนวนบิตของคำสั่ง
- Page ละ $1024 \times 4 = 4096$ ไบต์ Address = 0x000 ถึง 0xFFFF

ในรูปที่ 5.x.x หน่วยความจำเสมือนมีขนาดเท่ากับ $2^{32} = 2^2 \times 2^{30} = 4GB$

$$5 \times 4 = 20 \text{ bit}$$

- Page #0000_0 ถึง #FFFF_F
- หน่วยความจำภายในภาพ มีขนาดเท่ากับ $2^{24} = 2^2 \times 2^{20} = 16MB$
- Frame #0000_0 ถึง #00FF_F
- พื้นที่เสมือนแบ่งเป็น Kernel Space และ User Space โปรแกรมจะมีเซ็กเมนท์ต่างๆ

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page

Page 0x0000_0

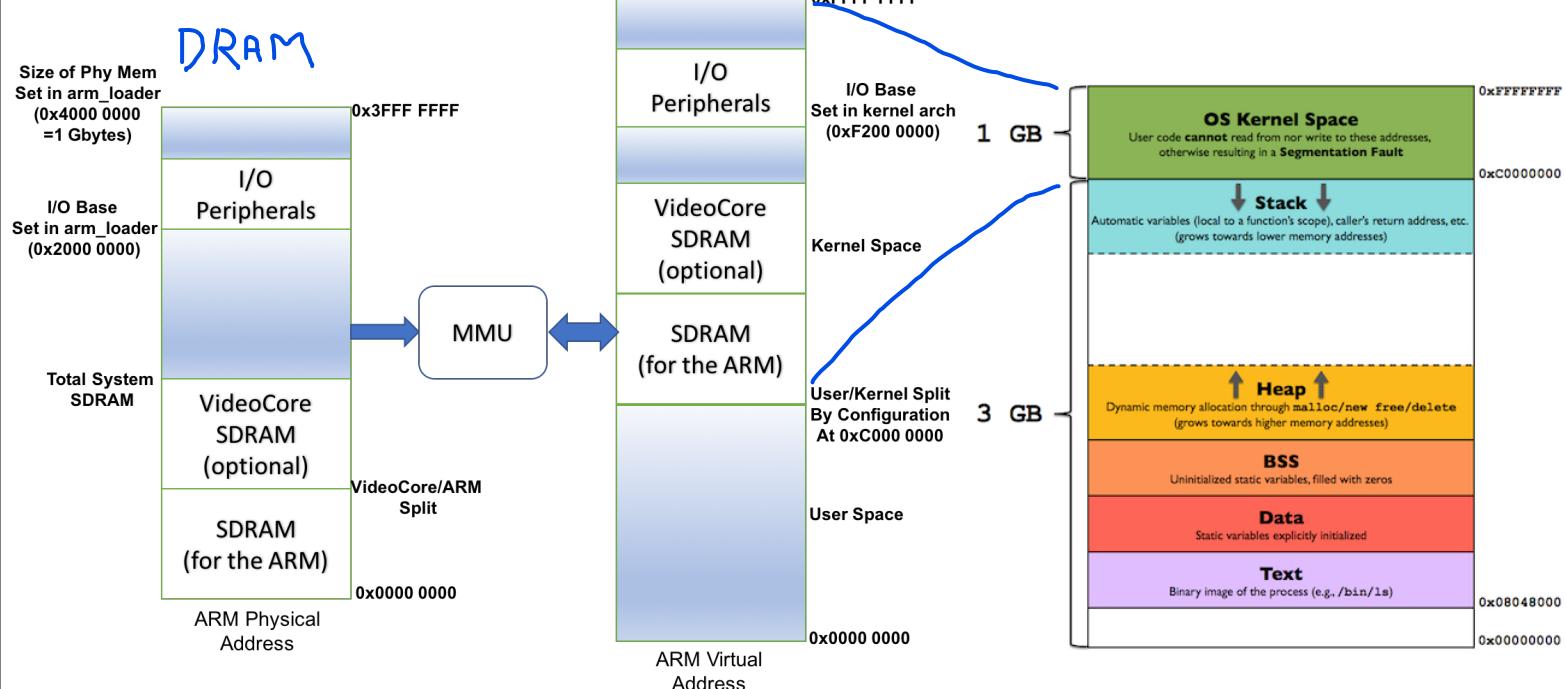
- 0x0000_0000 – 0x0000_0003 คำสั่งที่ 00
- 0x0000_0004 – 0x0000_0007 คำสั่งที่ 01
- 0x0000_0008 – 0x0000_000B คำสั่งที่ 02
- ...
- 0x0000_00FC – 0x0000_00FF คำสั่งที่ 63
- 0x0000_0100 – 0x0000_0103 คำสั่งที่ 64
- 0x0000_0104 – 0x0000_0107 คำสั่งที่ 65
- 0x0000_0108 – 0x0000_010B คำสั่งที่ 66
- ...
- 0x0000_01FC – 0x0000_01FF คำสั่งที่ 127
- ...
- 0x0000_0FFC – 0x0000_0FFF คำสั่งที่ 1023

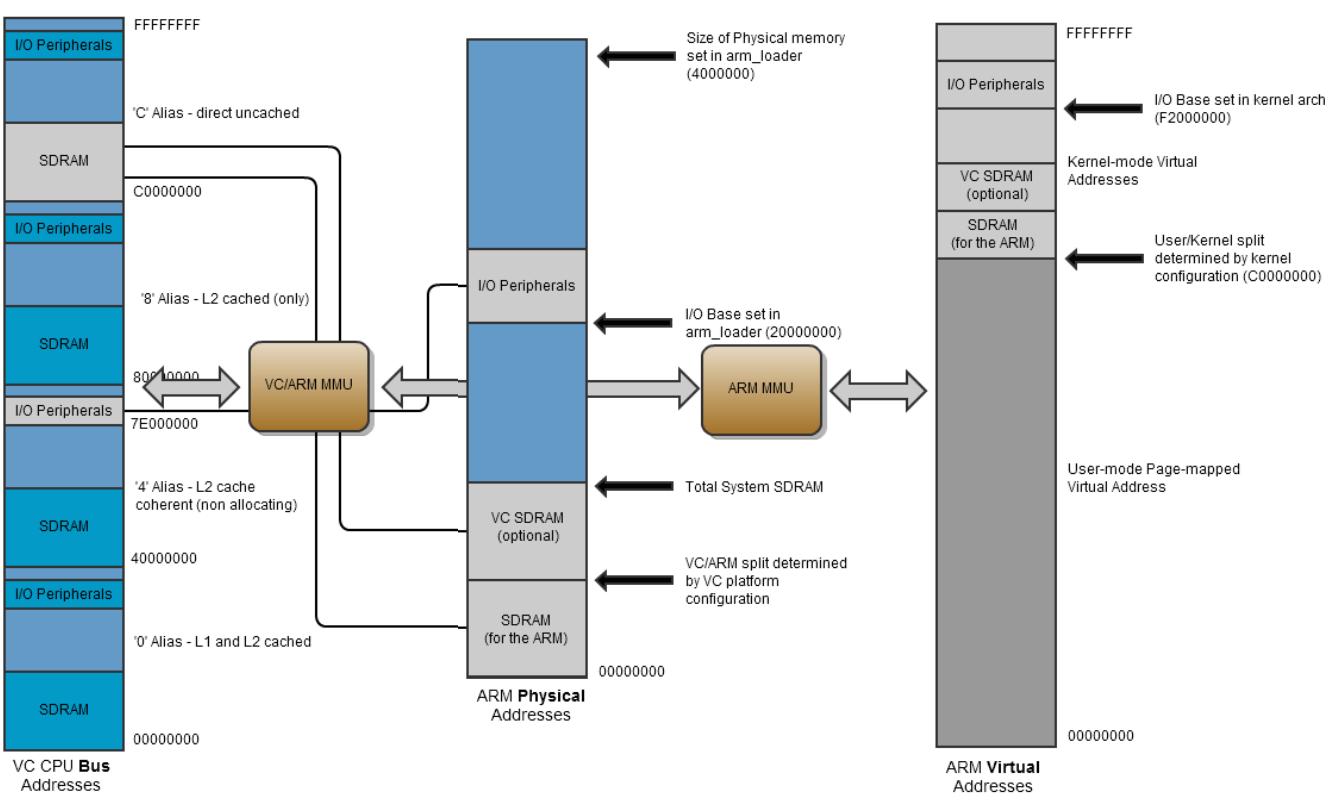
A2v Translation

Frame 0x0010_1

- 0x0010_1000 – 0x0010_1003 คำสั่งที่ 00
- 0x0010_1004 – 0x0010_1007 คำสั่งที่ 01
- 0x0010_1008 – 0x0010_100B คำสั่งที่ 02
- ...
- 0x0010_10FC – 0x0010_10FF คำสั่งที่ 63
- 0x0010_1100 – 0x0010_1103 คำสั่งที่ 64
- 0x0010_1104 – 0x0010_1107 คำสั่งที่ 65
- 0x0010_1108 – 0x0010_110B คำสั่งที่ 66
- ...
- 0x0010_11FC – 0x0010_11FF คำสั่งที่ 127
- ...
- 0x0010_1FFC – 0x0010_1FFF คำสั่งที่ 1023

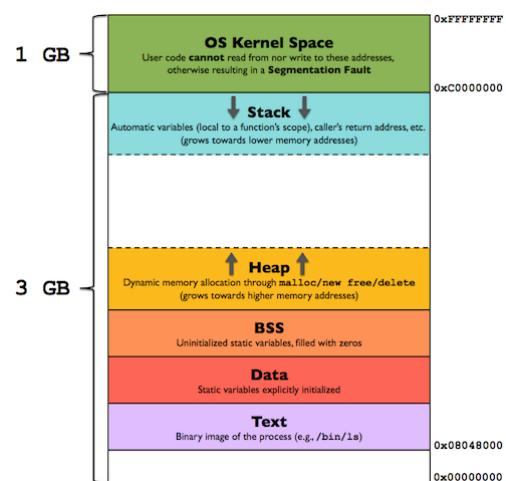
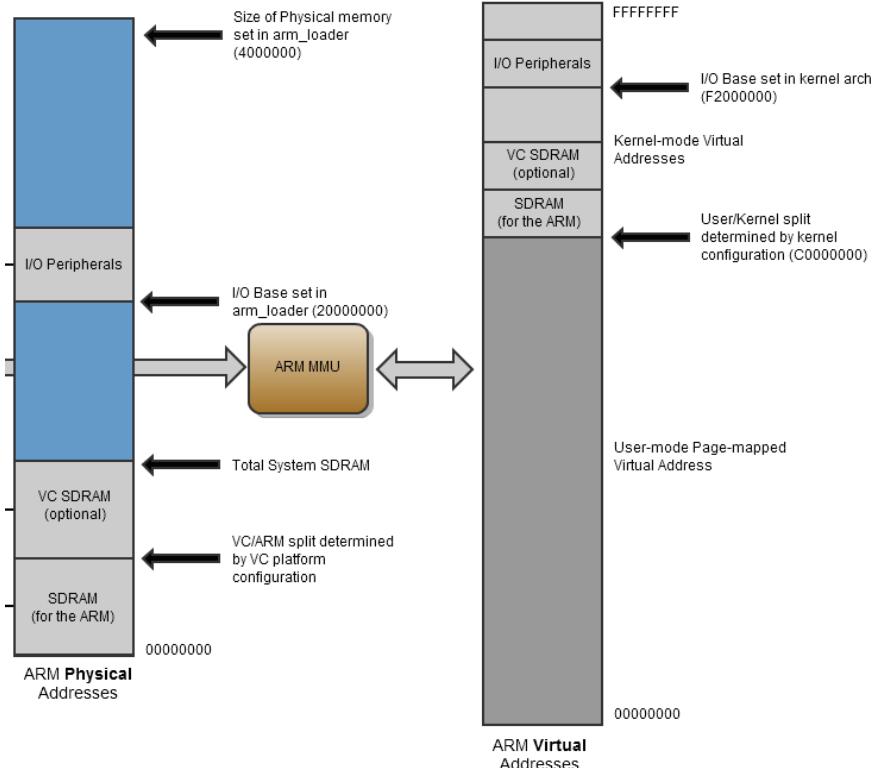
5.2.2 หน่วยความจำเสมือน (Virtual Memory) ของบอร์ด Pi3





5.2.2 หน่วยความจำเสมือน (Virtual Memory)

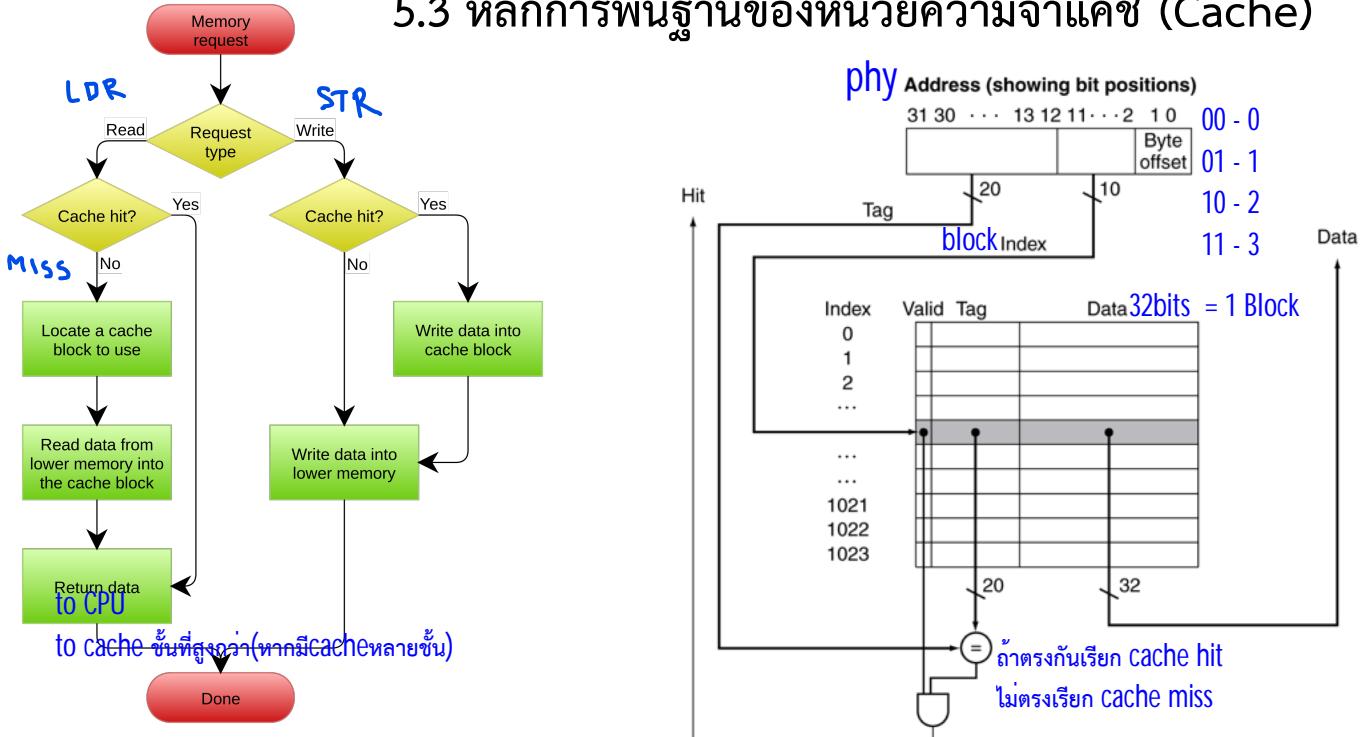
ของบอร์ด Pi3



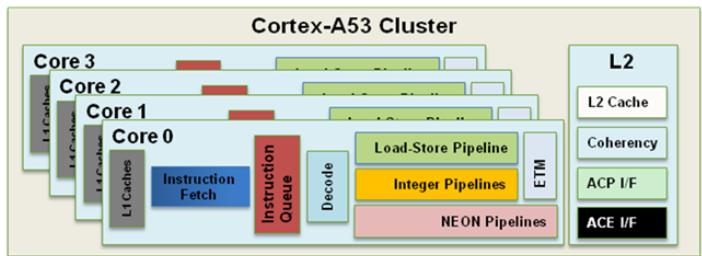
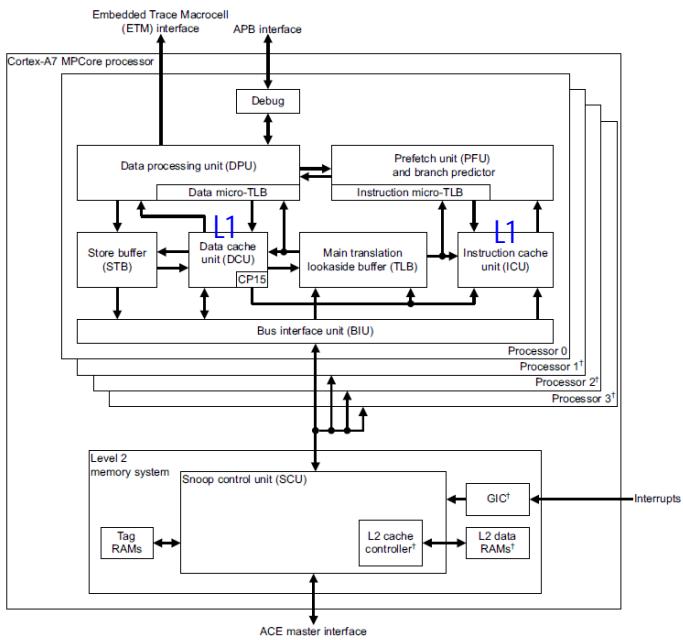
5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)

- แคชมีความจุน้อยแต่ทำงานเร็ว เพราะใช้เทคโนโลยี SRAM และสามารถบรรจุอยู่ในแผ่นซิลิกอนเดียวกับซีพียูคอร์
- แคชมีหลายชั้น ชั้นบนสุดของแต่ละคอร์ เรียกว่า ชั้น L1 ทำงานที่แยกจากกัน คือ
 - แคชชั้น L1 Instruction เป็นพื้นที่ของ Text Segment เก็บคำสั่ง
 - แคชชั้น L1 Data Cache เป็นพื้นที่อื่นๆ เก็บข้อมูลที่ไม่ใช่ Text Segment
- แคชชั้น L2 จะมีความจุมากขึ้นและเก็บคำสั่งและข้อมูลจากทุกๆ เมนูและคอร์ต่างๆ ใช้งานร่วมกัน
- บางพื้นที่ของหน่วยความจำเปลี่ยนใช้งานผ่านแคช บางพื้นที่จะไม่ใช้งานผ่านแคชแม้แต่ชั้นเดียว
- แคชใช้ Phy Address ในการทำงาน เพราะจะไม่ซ้ำซ้อนกับ Virtual Address

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)



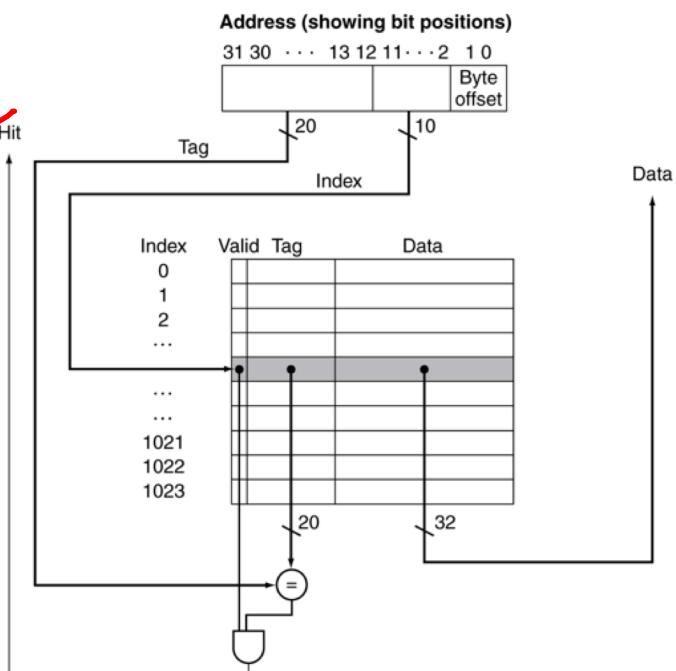
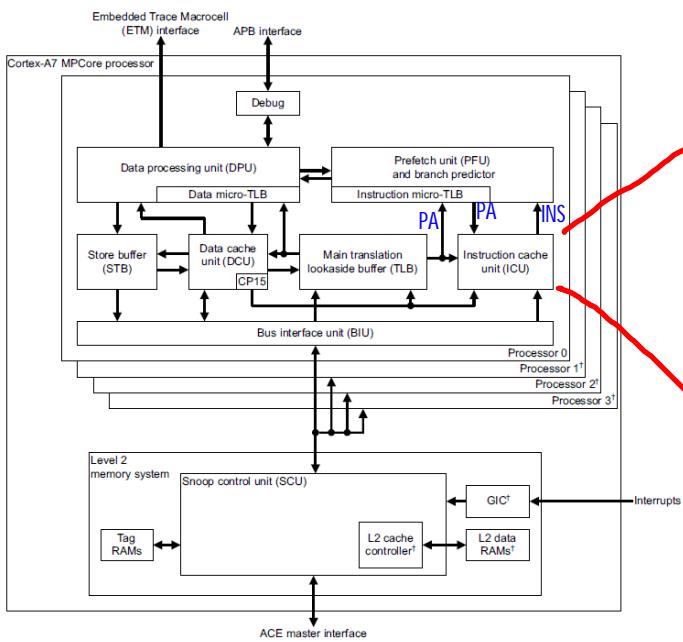
5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ARM Cortex A7 & A53



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

17

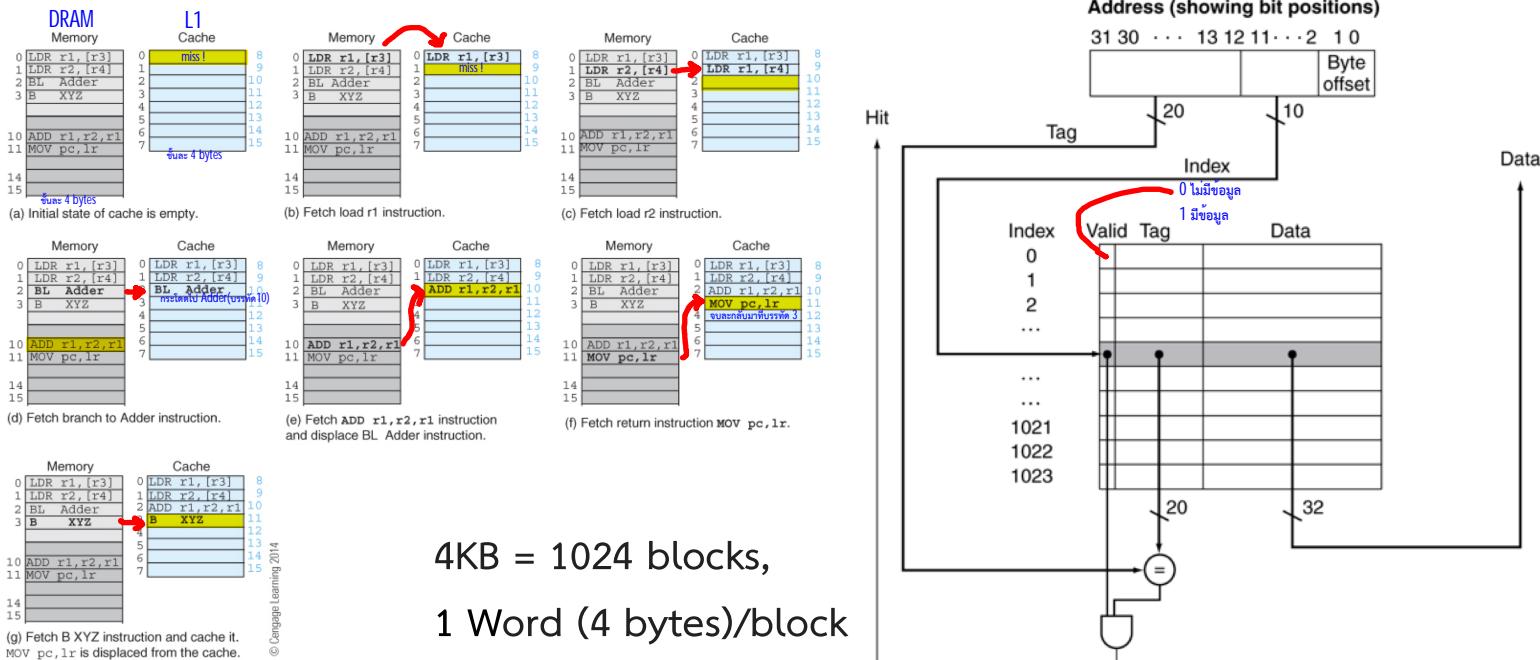
5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ARM Cortex A7 & A53



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

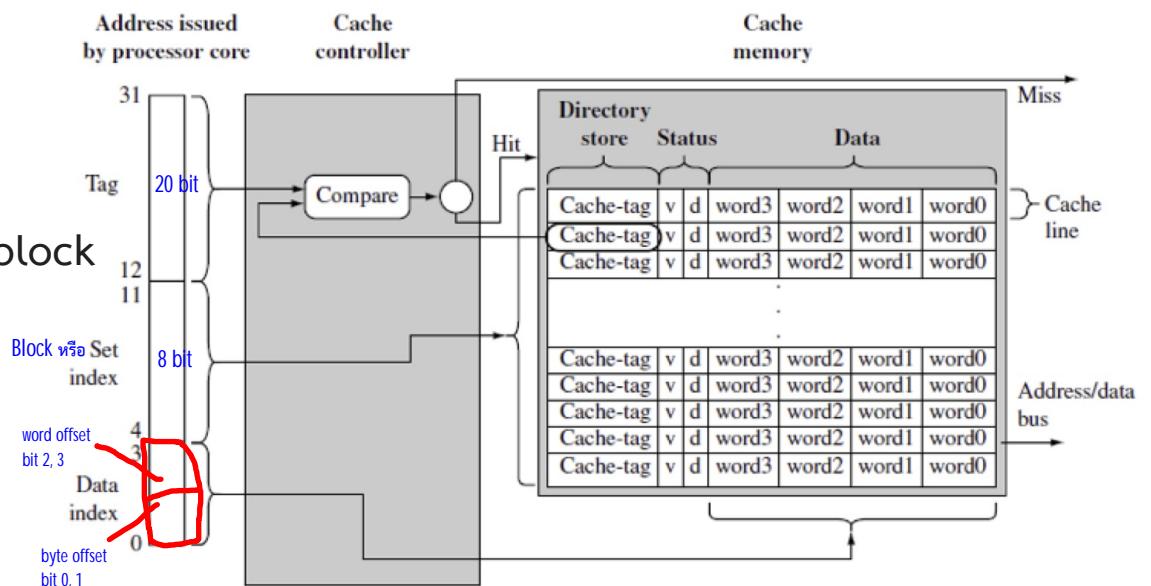
18

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache) ชนิด Direct Map

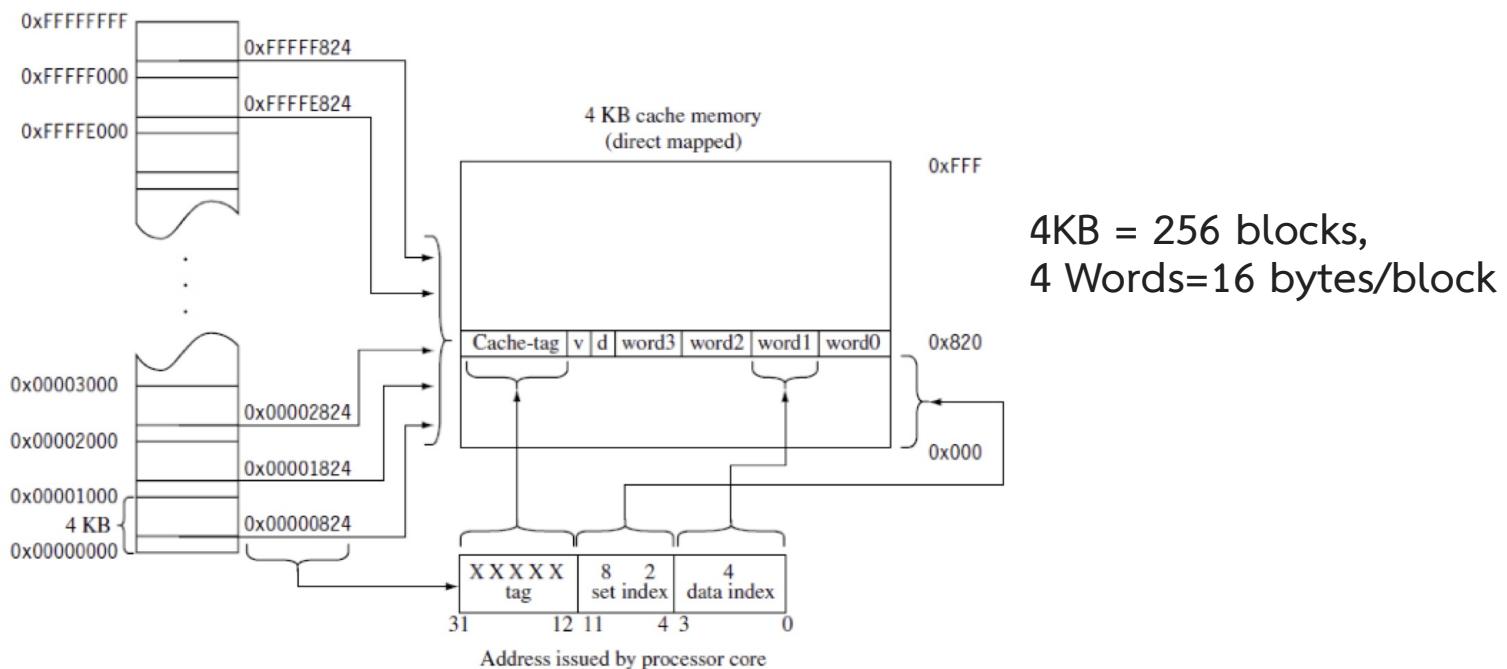


5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache) ชนิด Direct Map

4KB = 256 blocks,
4 Words=16 bytes/block



5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache) ชนิด Direct Map

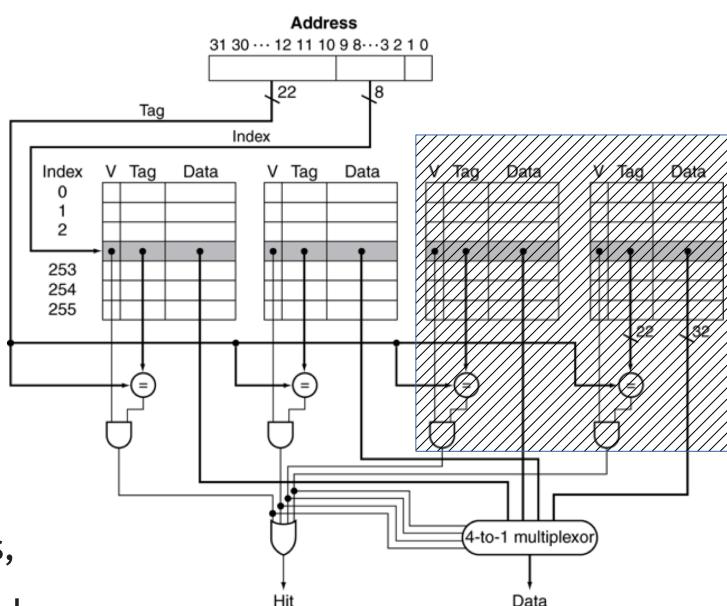
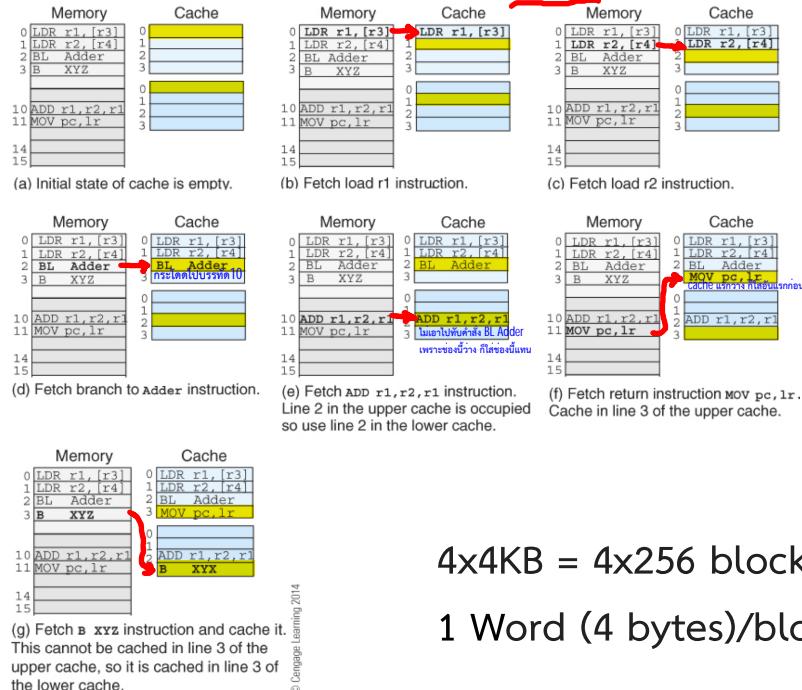


Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

21

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)

ชนิด 2-Way และ 4-Way Set Associative

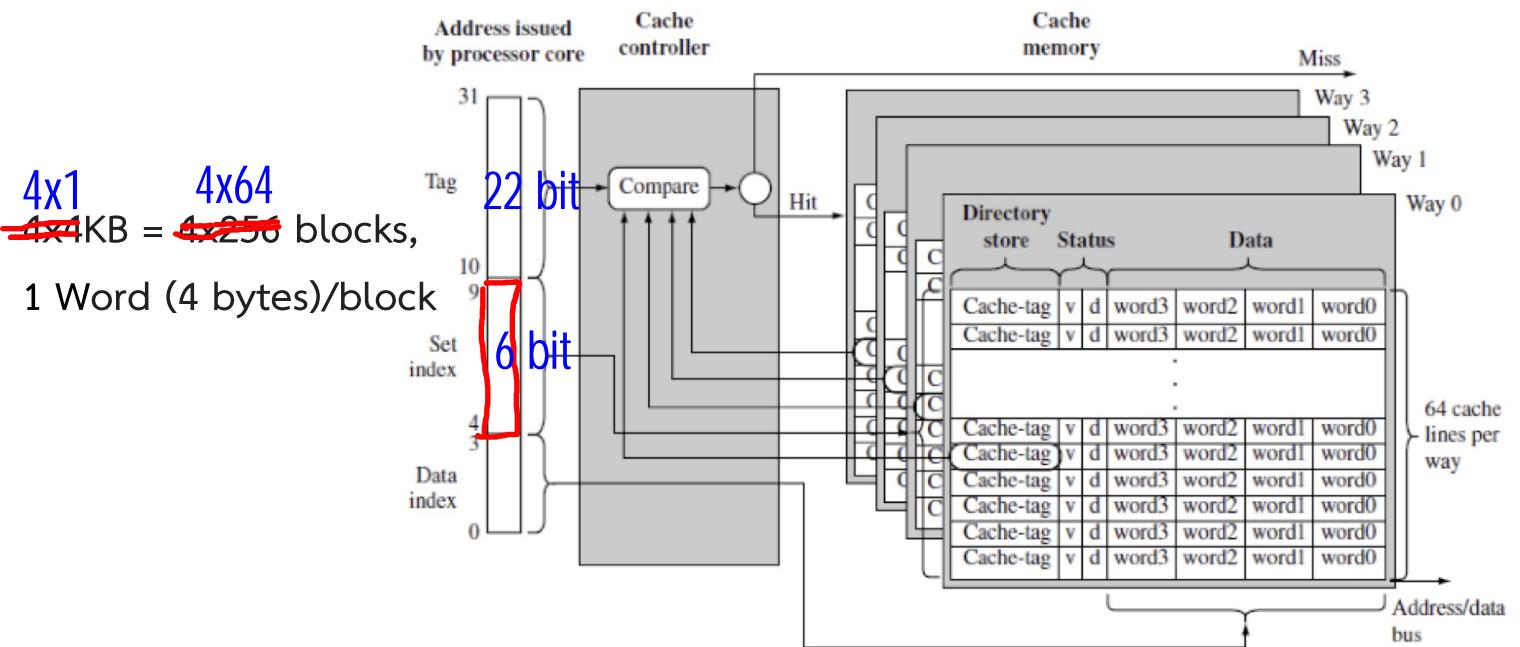


Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

22

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)

ชนิด 2-Way และ 4-Way Set Associative

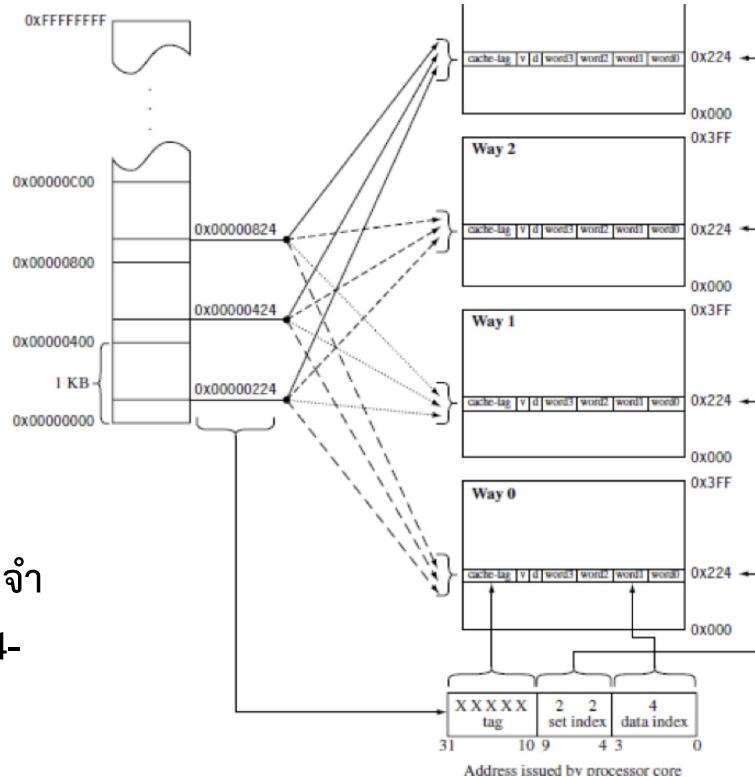


Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

23

4x4KB = 4x256 blocks,
1 Word (4 bytes)/block

5.3 หลักการพื้นฐานของหน่วยความจำ
แคช (Cache) ชนิด 2-Way และ 4-
Way Set Associative



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

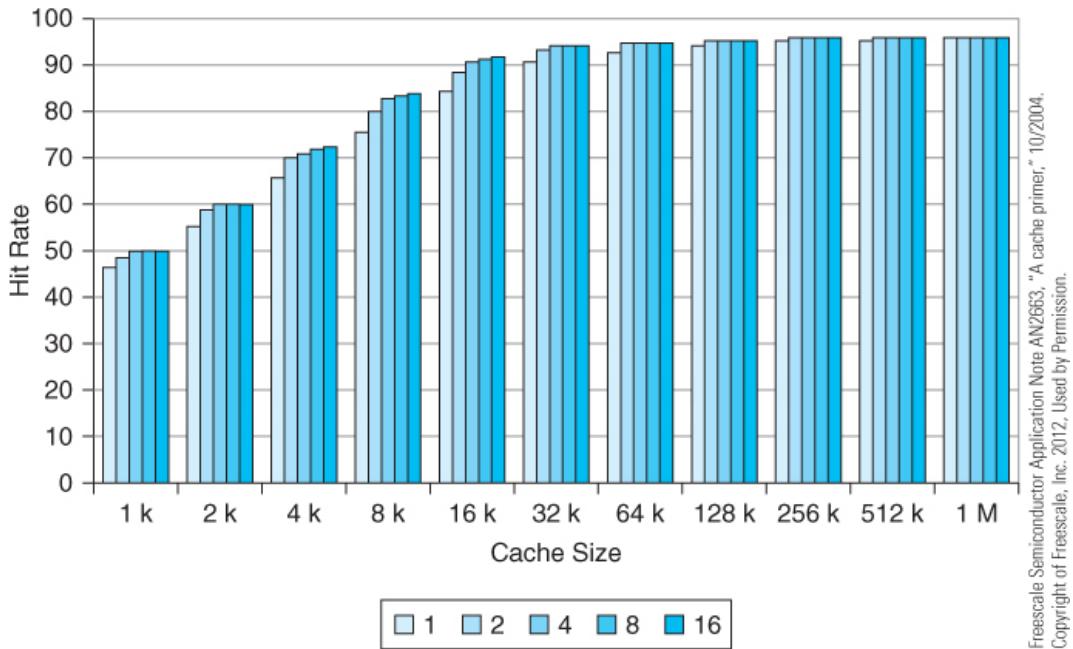
24

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)

ชนิด 2-Way และ 4-Way Set Associative

Set-associativity and cache size

FIGURE 9.17

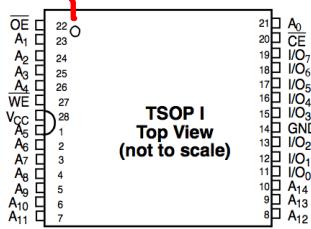


Freescale Semiconductor Application Note AN2663, "A cache primer," 10/2004.
Copyright of Freescale, Inc. 2012. Used by Permission.

5.4 หน่วยความจำชนิด静态ติกแรม (Static RAM: SRAM)

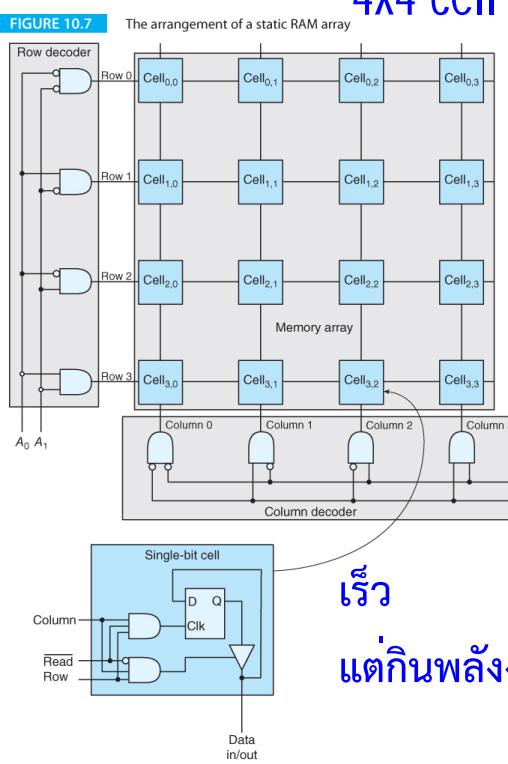
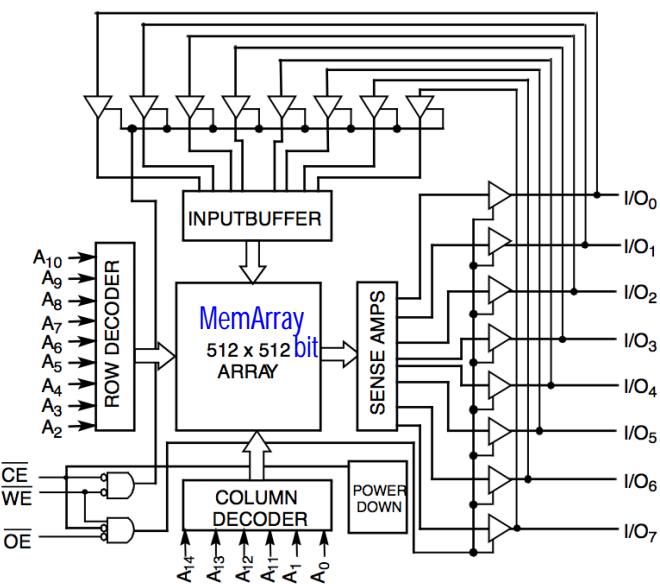
- แคชลำดับที่ 1 และ 2 ในชิป BCM2837 สร้างจากหน่วยความจำ static SRAM ซึ่งมีโครงสร้างที่ไม่ซับซ้อนและสามารถออกแบบให้ผลิตพร้อมกับวงจรทรานซิสเตอร์ในชิปปิคью นอกจากนี้ หน่วยความจำ static SRAM นี้ยังนิยมใช้งานเป็นหน่วยความจำหลักภายในชิปไมโครคอนโทรลเลอร์ ที่ต้องการสมรรถนะต่อถึงปานกลางโดยมีความจุหลายขนาด ตั้งแต่ 16 กิโลไบต์ ถึงหลายเมกะไบต์
- หน่วยความจำ static SRAM (Static RAM: SRAM) หมายเลข CY62256 เป็นกรณีศึกษาชิป CY62256 ใช้เทคโนโลยีการผลิตชนิด CMOS ในปี ค.ศ. 2002
 - ใช้กับแหล่งจ่ายไฟตั้งแต่ 4.5 - 5.5 โวลต์
 - รองรับการทำงานความเร็วสูง เนื่องจากใช้เวลาเข้าถึงน้อยเท่ากับ 55 นาโนวินาที
 - ชิปบริโภคกำลังไฟน้อยโดยมีค่าสูงสุดเพียง 275 มิลลิวัตต์ระหว่างปฏิบัติงาน
 - ชิปบริโภคกำลังไฟน้อยโดยมีค่าสูงสุดเพียง 28 มิลลิวัตต์ระหว่างไม่ทำงาน (Stand by)

5.4 หน่วยความจำชนิดสแตติคแรม (Static RAM: SRAM)

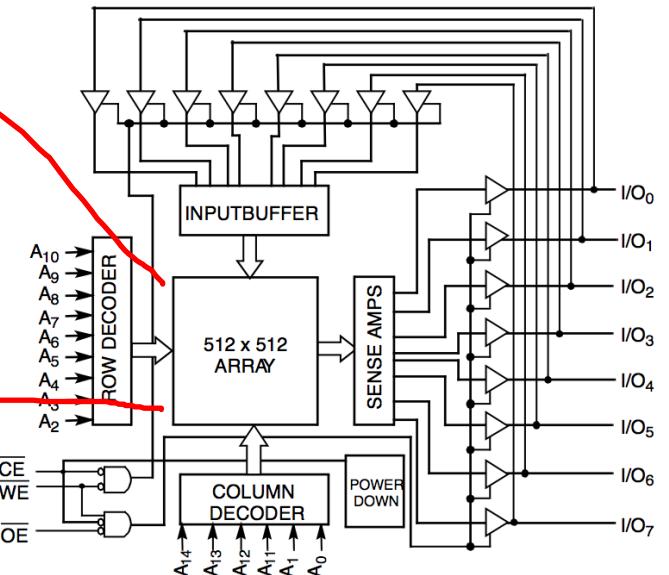


32 kB

ใช้เปิดปิด
(chip enable)

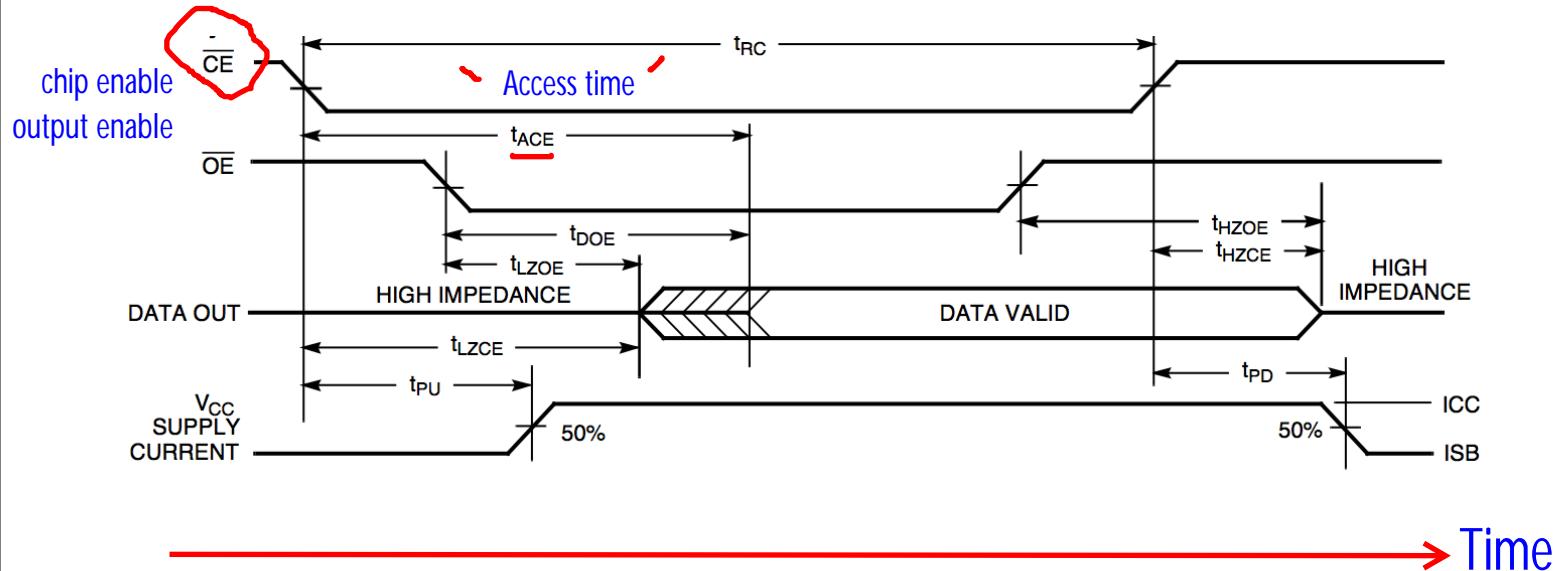


5.4 หน่วยความจำชนิดสแตติคแรม (Static RAM: SRAM)



5.4 หน่วยความจำชนิดสแตติคแรม (Static RAM: SRAM):

Read Cycle

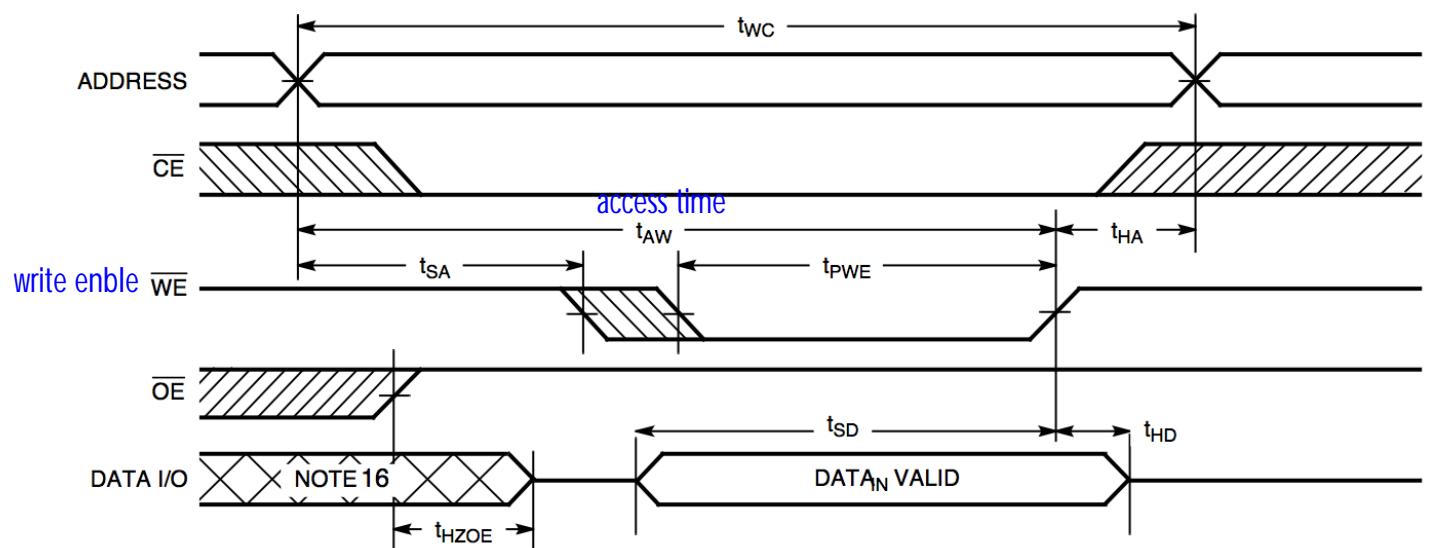


Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

29

5.4 หน่วยความจำชนิดสแตติคแรม (Static RAM: SRAM):

Write Cycle



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

30

5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM)

ความจุสูง

- ผู้ผลิตเครื่องคอมพิวเตอร์โน้ตบุ๊ก โทรศัพท์เคลื่อนที่สมาร์ทโฟน และอุปกรณ์พกพา ต่างนิยมออกแบบติดตั้งซิพหน่วยความจำ DRAM บนเมนบอร์ด (Main Board) เช่นเดียวกับบอร์ด Pi3 เพื่อลดขนาดและปริมาณของเครื่องให้มีขนาดเท่ากับบอร์ดเครดิต
- ข่าวการผลิต DRAM ยังไม่สามารถรวมกับข่าวการผลิตไมโครโปรเซสเซอร์ได้ ผู้ผลิตจึงจำเป็นต้องผลิตซิพ DRAM แยกต่างหาก

5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM): DDR2 SDRAM for PC



5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM):

DDR2 SDRAM for Notebook
TECMIYO

2GB DDR2 667MHz



LAPTOP RAM/SODIMM Voltage 1.8v

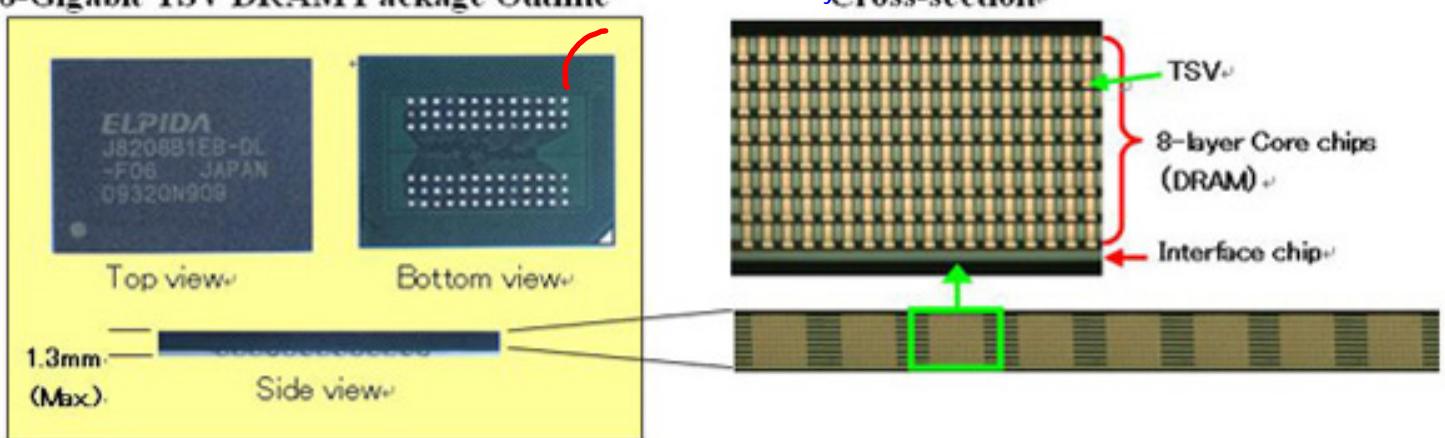
5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM)

ขาเป็นทรงกลม

8-Gigabit TSV DRAM Package Outline

BGA - Ball Grid Array

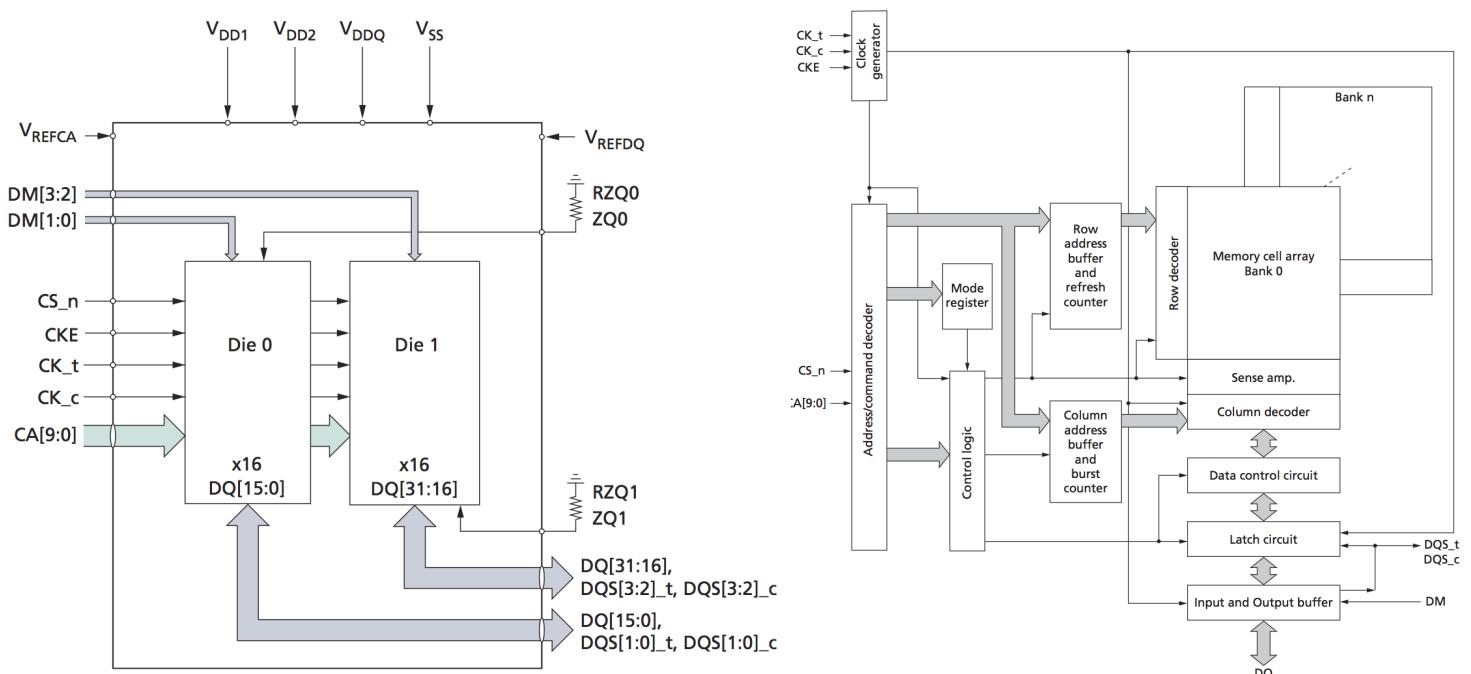
Cross-section



5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM)

- โครงสร้างของหน่วยความจำชนิด DDR2 จำนวน 2 ดาย (die) แต่ละดายประกอบด้วย แผงอะเรย์ DRAM จำนวน 8 ชั้น หรือ 8 แบงค์ (Bank) ชั้นละ 32 เมกะเซลล์ $\times 16$ บิต คิดเป็น $2 \times 8 \times 32$ เมกะเซลล์ $\times 16$ บิต \times ต่อชิพ หรือ $2^1 \times 2^3 \times 2^5 \times 2^{20} \times 2^4 = 2^{33} = 2^3 \times 2^{30} = 8 \text{ Gbits} = 1 \text{ GByte}$
- แบงค์ที่ 0 ถึง 7 แต่ละแบงค์ประกอบด้วยวงจรต่อรหัสแอดเดรส (Address Decoder) ในแนวนอน (Row Decoder) และแนวตั้ง (Column Decoder)

5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM)



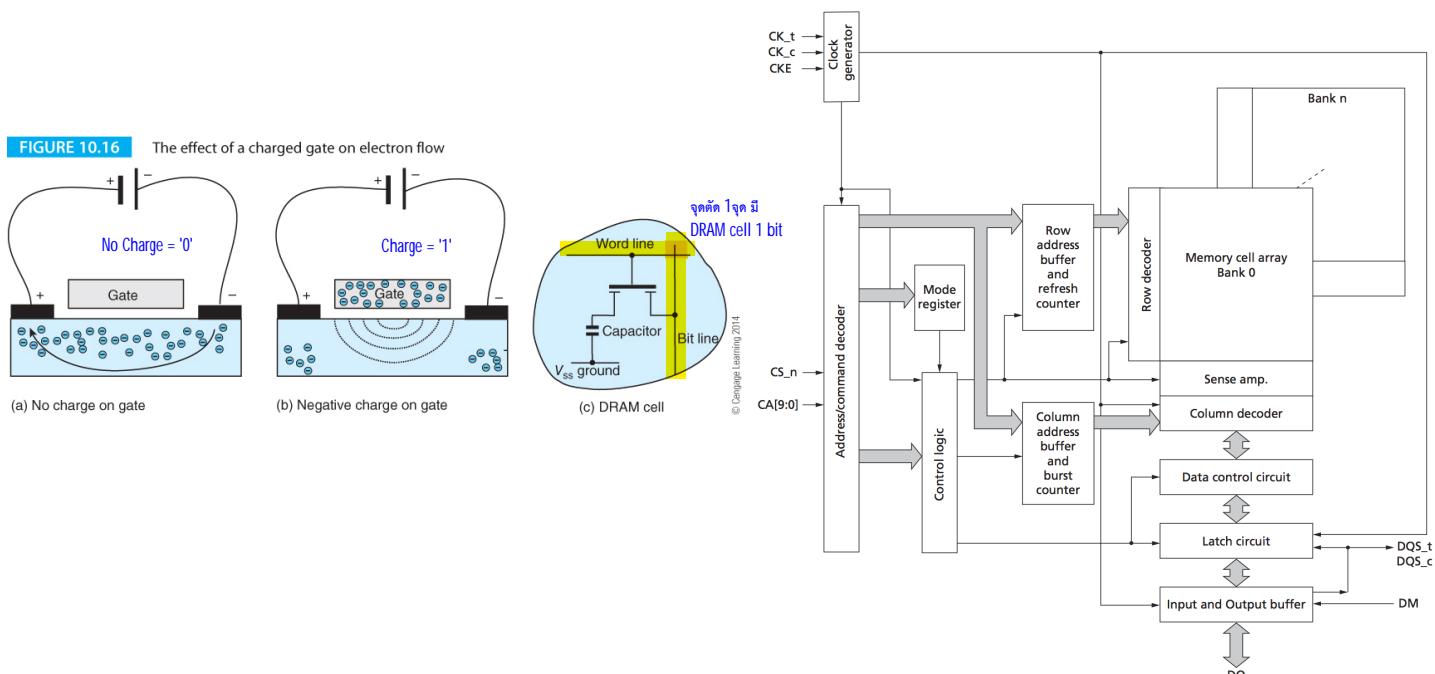
5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM)

- ภายในชิปประกอบด้วยขาสัญญาณต่างๆ เรียงตามลำดับความสำคัญ ดังนี้
 - Chip Select Not ใช้เปิด/ปิดการทำงานของชิป เพื่อช่วยประหยัดพลังงาน
 - Clock Enable ซึ่งสามารถควบคุมสัญญาณ $CKE=0$ เพื่อพักรการใช้งาน DRAM ชั่วคราวเพื่อช่วยประหยัดพลังงาน
 - CK (Clock) และ CK# (Clock Not) สัญญาณคลื่นความถี่สูงสุด 400 เมกะเฮิร์ตซ์
 - คอมมานด์แอดเดรส CA[0:9] ขนาด 10 บิต ใช้สำหรับรับสัญญาณแอดเดรสและคำสั่ง (Command) โดยการมัลติเพล็กซ์ (Multiplex)
 - ดาต้าสโตรบ DQS[0:3] ขนาด 4 บิต ใช้สำหรับควบคุมการอ่านและการเขียนข้อมูล
 - ดาต้าบัส DQ[0:31] จำนวน 32 บิต หรือ 4 ไบท์ DDR2 นี้ สามารถกำหนดขนาดข้อมูลจากการอ่านเขียนต่อเนื่อง (Burst length) เป็น 4, 8 และ 16 ตำแหน่งต่อเนื่องกัน
 - ดาตามาร์ก} DM (Data Mask) [0:3] ขนาด 4 บิต ใช้สำหรับมาสก์ (Mask) หรือปิด เพื่อควบคุมการเขียนข้อมูลแต่ละไบท์ โดย DM[0] ควบคุมไบท์ที่ 0, DM[1] ควบคุมไบท์ที่ 1 ตามลำดับ

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

37

5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM)

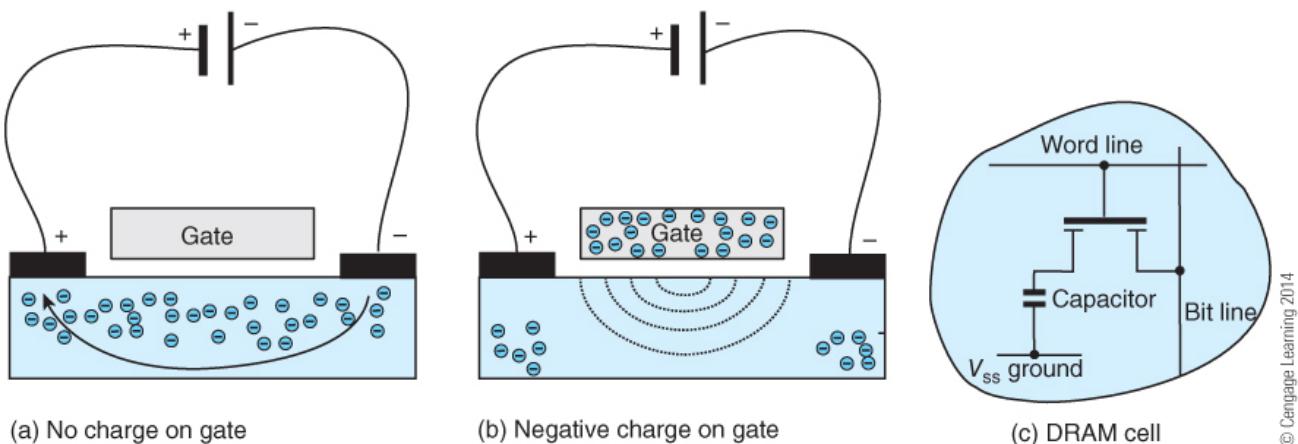


Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

38

5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM): DRAM Cell

FIGURE 10.16 The effect of a charged gate on electron flow



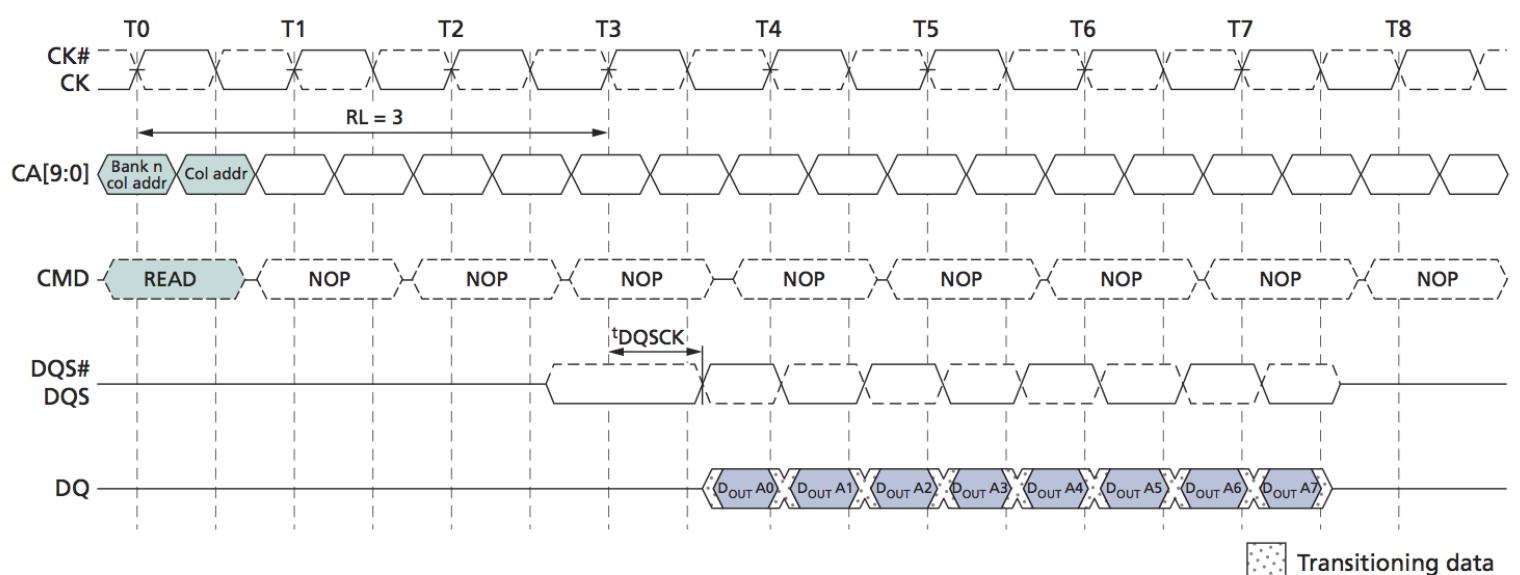
© Cengage Learning 2014

- เก็บข้อมูล: 1/0 = มีประจุ/ไม่มีประจุ
- ต้องรีเฟรช (Refresh) หรือประจุไฟใหม่ เพื่อการ charge/no charge อาจทำประจุรั่วหายไป จนอ่านข้อมูลผิดเพี้ยนได้

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

39

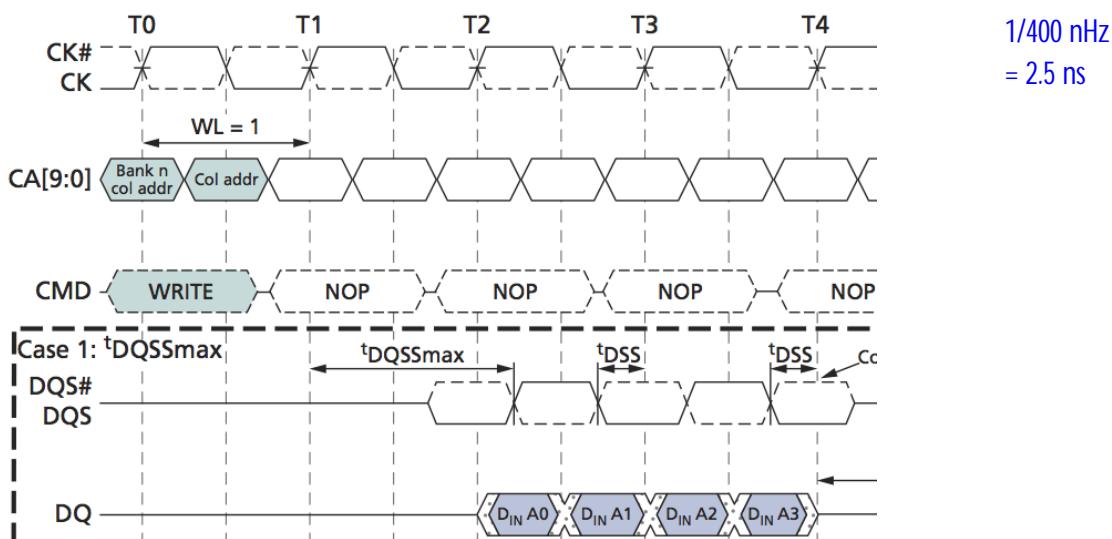
5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM): Read Cycle



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

40

5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM): Write Cycle



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

41

5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM): Refresh

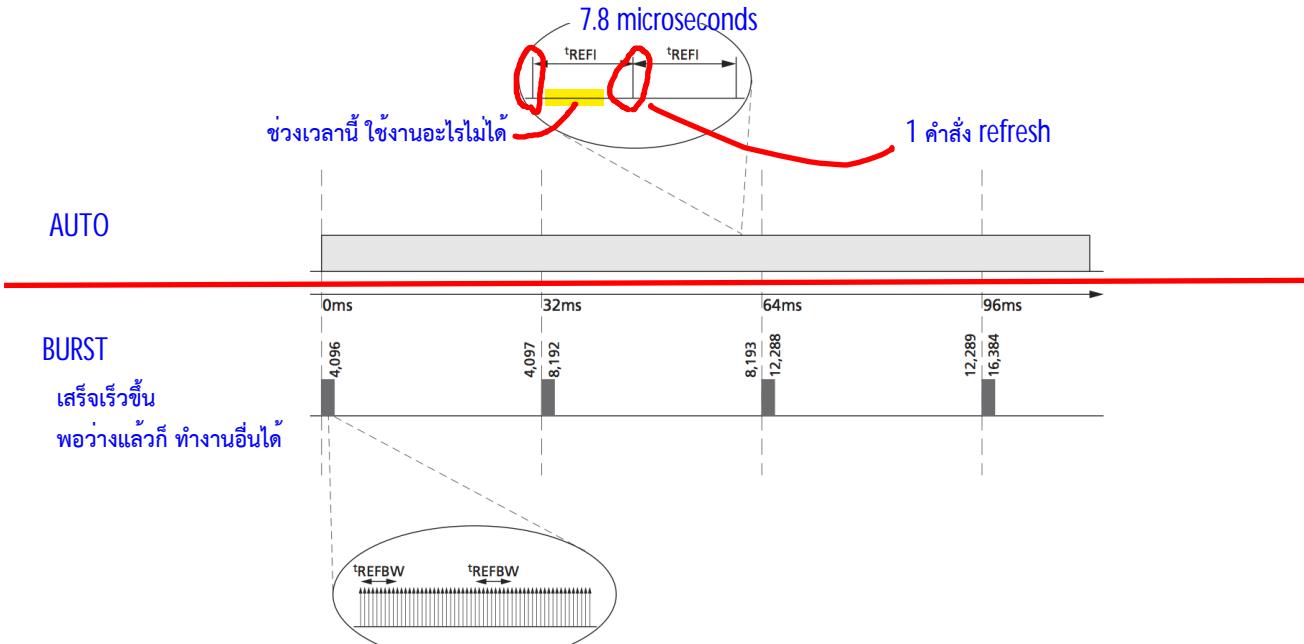
- การรีเฟรช (Refresh) คือ การอ่านข้อมูลที่อยู่ในบิทเซลล์ต่างๆ แล้วเขียนช้าลงไป เพื่อป้องกันไม่ให้ประจุที่เก็บอยู่ในบิทเซลล์ต่างๆ รั่วไหลหายไป เนื่องจากเซลล์ต่างๆ ที่ใช้เก็บข้อมูล ทำหน้าที่เก็บ ('1') หรือไม่เก็บ ('0') ประจุไฟฟ้า เมื่อเวลาผ่านไปไม่กี่มิลลิวินาที จึงเกิดการรั่วไหลของประจุเหล่านี้ เมื่อจำนวนประจุลดลง การแยกแยะระหว่างบิทเซลล์ที่มีและไม่มีประจุจึงยากขึ้น ซึ่งอาจทำให้การอ่านเกิดความผิดพลาด
- แกนนอนในรูป คือ แกนเวลา การรีเฟรชทุกๆ 32 มิลลิวินาที โดยส่งคำสั่ง Refresh 4096 คำสั่ง เป็นระยะเวลา $t_{REFW} = 32$ มิลลิวินาที แต่ละคำสั่งจะห่างกันด้วยระยะเวลา Refresh Interval $t_{REFI} = 7.8$ ไมโครวินาที หนึ่งคำสั่งจะทำการ Refresh เป็นชุดๆ แต่ละชุดด้วยระยะเวลา Refresh Burst Window $t_{REFBW} = 4.16$ ไมโครวินาที โดยแต่ละชุด จะทำการรีเฟรชทีละแบบค์ตัวๆ ระยะเวลา 60 นาโนวินาที

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

42

5.5 หน่วยความจำหลักชนิดไดนามิกแรม

(Dynamic RAM: DRAM): Refresh



Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

43

สรุปท้ายบท

ไม่มี refresh

มี refresh

- หน่วยความจำลำดับชั้นอาทิตย์การบริหารรวมกันของหน่วยความจำหลักและหลักขนาดเข้าด้วยกัน เนื่องจากมีเทคโนโลยีที่แตกต่างกัน หน่วยความจำขนาดเล็กความเร็วสูง เช่น SRAM นำมาใช้งานเป็นรีจิสเตอร์ และ แคช ทำหน้าที่เป็นตัวแทนของหน่วยความจำที่มีความจุมากกว่าแต่ความเร็วต่ำกว่า เช่น DRAM นำมาใช้งานเป็นหน่วยความจำหลัก และใช้เทคโนโลยีหน่วยความจำแฟลช และ อาร์ดดิสก์ เป็นอุปกรณ์เก็บรักษาข้อมูล เพื่อให้คอมพิวเตอร์มีความจุเพียงพอและตอบสนองต่อความต้องการใช้งานระบบโดยเฉลี่ยได้รวดเร็วขึ้น โดยการผสานจุดเด่นของหน่วยความจำแต่ละชนิดเข้าด้วยกัน และช่วยประยุกต์ทันทุนของระบบ

ไม่ต้องมีไฟเลี้ยง

Computer Organization & ARM Assembly Language: RaspberryPi3, Surin K., KMITL

44