

องค์ประกอบคอมพิวเตอร์และภาษาแอสเซมบลี: กรณีศึกษา Raspberry Pi

รศ.ดร.สุรินทร์ กิตติธรกุล

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

สารบัญ

- 5.1 โครงสร้างของลำดับขั้นหน่วยความจำของบอร์ด Pi3
- 5.2 หน่วยความจำเสมือน (Virtual Memory)
- 5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)
- 5.4 หน่วยความจำชนิดสแตติคแรม (Static RAM: SRAM)
- 5.5 หน่วยความจำหลักชนิดไดนามิกแรม (Dynamic RAM: DRAM)

Multi-User Multi-Programming OS

16:09:15 up 4 days, 5:30, 21 users, load average: 0.00, 0.01, 0.00							
USER	TTY	FROM	LOGIN@	IDLE	JCPU	PCPU	WHAT
pi	tty1	-	Wed10	4days	0.15s	0.12s	-bash
t6301088	pts/0	100.127.251.0	15:21	47:31	0.23s	0.10s	nano Lab7_7.s
t6301088	pts/1	100.127.251.128	15:37	35.00s	0.60s	0.16s	nano Lab7_7.s
t6301035	pts/2	100.127.251.129	15:58	7:47	0.11s	0.11s	-bash
t6301003	pts/3	100.127.251.1	15:47	27.00s	0.79s	0.79s	-bash
t6301035	pts/4	100.127.251.129	16:06	17.00s	0.10s	0.10s	-bash
pi	pts/5	100.127.251.128	16:09	2.00s	0.14s	0.05s	w
t6301088	pts/7	100.127.251.129	13:50	2:09m	0.14s	0.14s	-bash
t6301088	pts/8	100.127.251.128	14:34	1:30m	0.24s	0.11s	nano Lab7_3.s
t6301035	pts/9	100.127.251.0	14:12	1:54m	0.21s	0.11s	nano
t6301059	pts/10	100.127.251.129	14:07	1:35m	0.51s	0.36s	nano Lab7a.s
t6301088	pts/11	100.127.251.128	14:04	1:47m	0.22s	0.22s	-bash
t6301003	pts/12	100.127.251.129	14:38	1:08m	0.63s	0.63s	-bash
t6301003	pts/13	100.127.251.129	14:16	1:50m	0.19s	0.05s	vi lab7_1.s
t6301088	pts/14	100.127.251.0	14:24	1:41m	0.12s	0.12s	-bash
t6301003	pts/15	100.127.251.128	14:22	1:46m	0.14s	0.01s	vi lab7_1.s
t6301018	pts/16	100.127.251.129	14:42	1.00s	0.50s	0.25s	nano Lab7_5.s
t6301088	pts/17	100.127.251.0	14:43	1:16m	0.14s	0.14s	-bash
t6301088	pts/18	100.127.251.1	15:00	1:00m	0.12s	0.12s	-bash
t6301088	pts/19	100.127.251.128	15:15	52:02	0.30s	0.10s	nano Lab7_7.s
t6301003	pts/20	100.127.251.0	15:44	8:34	0.10s	0.10s	nano Lab7_3.s

<https://www.cyberciti.biz/faq/unix-linux-list-current-logged-in-users/>

Multi-User

Multi-Programming

OS: HTOP

pi@Pi432b: ~

```
1          0.0% Tasks: 40, 15 thr; 1 running
2  || 2.0% Load average: 0.00 0.00
3          0.0% Uptime: 3 days, 22:47:14
4          0.0%
Mem | | | | | 101M/7.71G
Swp          0K/100.0M
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
16324	pi	20	0	7960	2816	2444	R	0.7	0.0	0:00.70	htop
404	avahi	20	0	6016	3084	2628	S	0.0	0.0	1:59.54	avahi-daemon: run
16309	pi	20	0	12204	3784	2996	S	0.0	0.0	0:00.02	sshd: pi@pts/2
1	root	20	0	33952	8356	6404	S	0.0	0.1	0:33.99	/sbin/init splash
120	root	20	0	70636	40856	39668	S	0.0	0.5	0:18.45	/lib/systemd/syst
157	root	20	0	18588	3832	2940	S	0.0	0.0	0:01.39	/lib/systemd/syst
354	systemd-t	20	0	22380	5384	4732	S	0.0	0.1	0:00.01	/lib/systemd/syst
319	systemd-t	20	0	22380	5384	4732	S	0.0	0.1	0:01.52	/lib/systemd/syst
359	root	20	0	7948	2220	2040	S	0.0	0.0	0:00.86	/usr/sbin/cron -f
361	root	20	0	27656	80	0	S	0.0	0.0	0:05.16	/usr/sbin/rngd -r
362	root	20	0	27656	80	0	S	0.0	0.0	0:00.21	/usr/sbin/rngd -r
363	root	20	0	27656	80	0	S	0.0	0.0	0:00.40	/usr/sbin/rngd -r
360	root	20	0	27656	80	0	S	0.0	0.0	0:05.85	/usr/sbin/rngd -r
374	root	39	19	11768	4544	3944	S	0.0	0.1	0:00.17	/usr/sbin/alsactl
446	root	20	0	64920	10352	8736	S	0.0	0.1	0:00.03	/usr/lib/udisks2/
457	root	20	0	64920	10352	8736	S	0.0	0.1	0:01.99	/usr/lib/udisks2/
509	root	20	0	64920	10352	8736	S	0.0	0.1	0:00.00	/usr/lib/udisks2/
528	root	20	0	64920	10352	8736	S	0.0	0.1	0:00.00	/usr/lib/udisks2/
384	root	20	0	64920	10352	8736	S	0.0	0.1	0:03.71	/usr/lib/udisks2/
388	root	20	0	13196	5932	5160	S	0.0	0.1	0:12.10	/lib/systemd/syst
430	root	20	0	25916	3320	2900	S	0.0	0.0	0:01.49	/usr/sbin/rsyslog

F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 SortByF7 Nice -F8 Nice +F9 Kill F10 Quit

การเชื่อมโยงระหว่างรีจิสเตอร์ แคช หน่วยความจำสมீனและอุปกรณ์เก็บรักษาข้อมูล



Cortex®-A53

ARM CoreSight™ Multicore Debug and Trace

ARMv8-A
32b/64b CPU
NEON™
SIMD engine with
crypto ext.
Floating Point
Unit

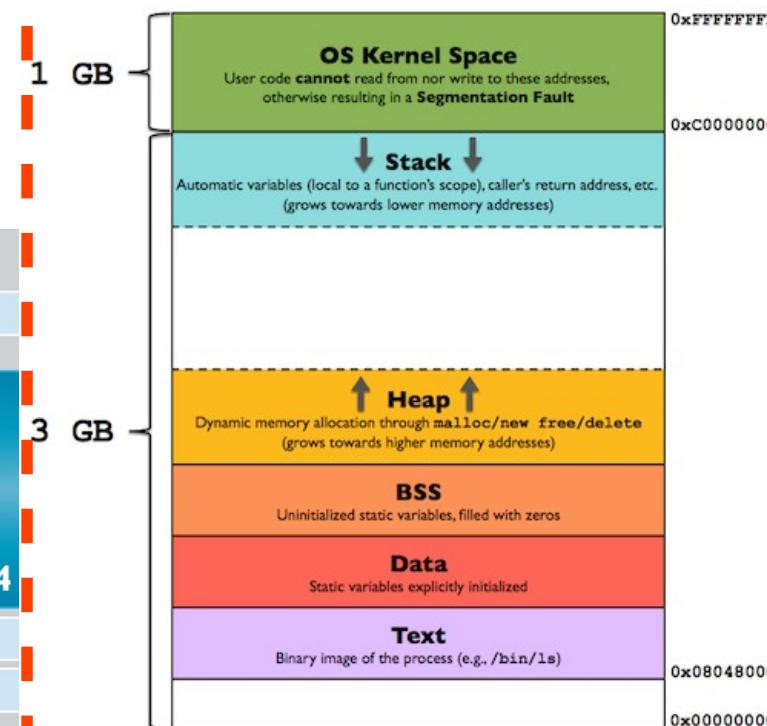
8-64k I-Cache
w/parity 8-64k D-Cache
w/ECC Core
1 2 3 4

L2 w/ECC (128kB ~ 2MB)
Configurable AMBA®4 ACE or
AMBA5 CHI Coherent Bus Interface

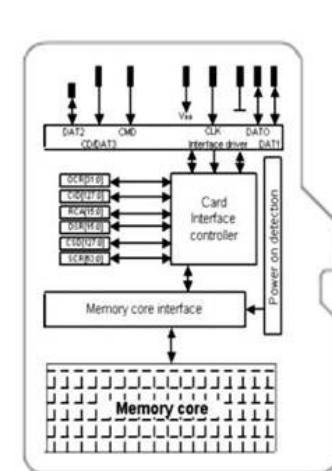
รีจิสเตอร์

แคชลำดับที่ 1 และ 2

BCM2837/2711

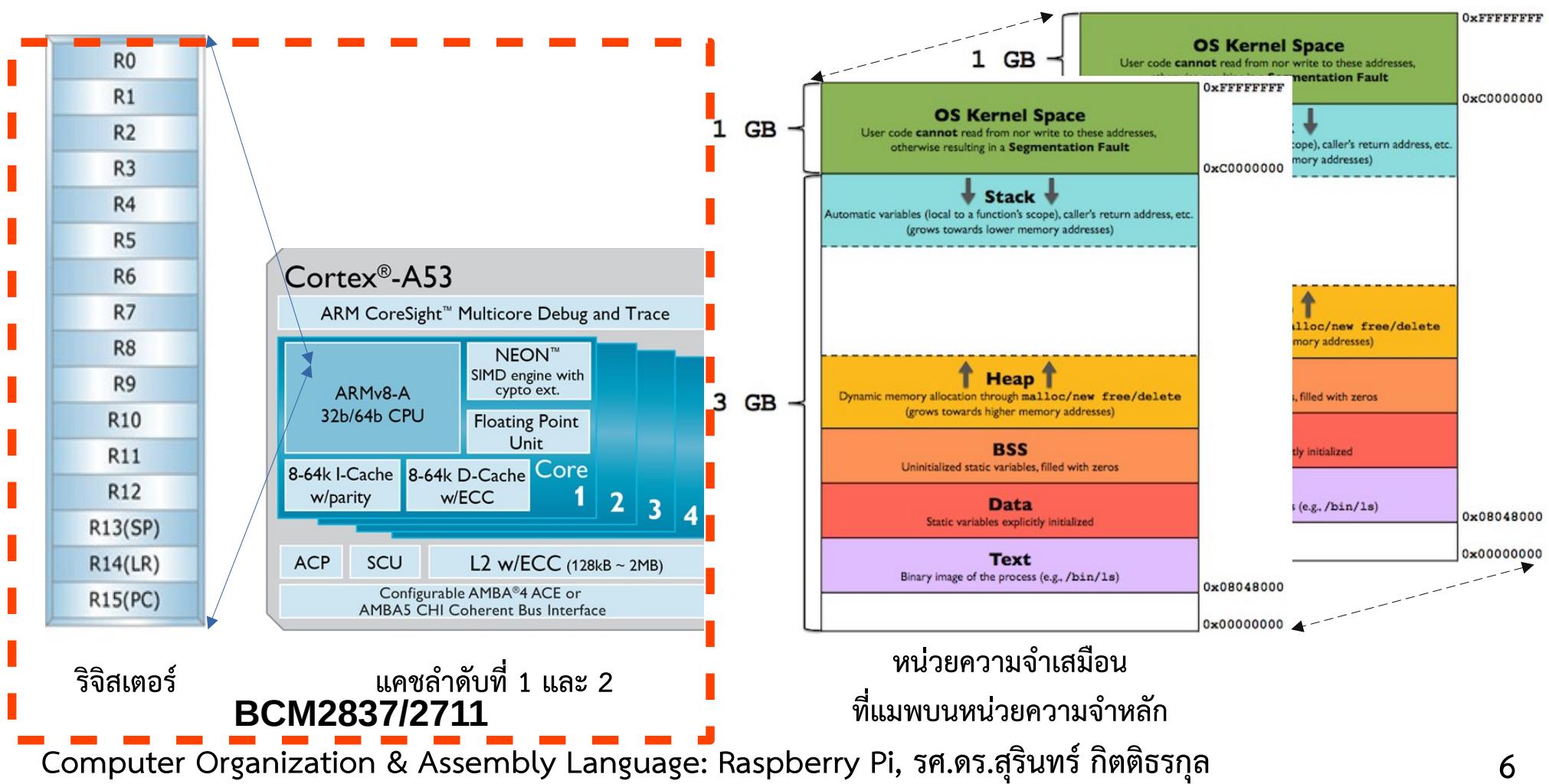


หน่วยความจำสมீன
ที่ mapping หน่วยความจำหลัก



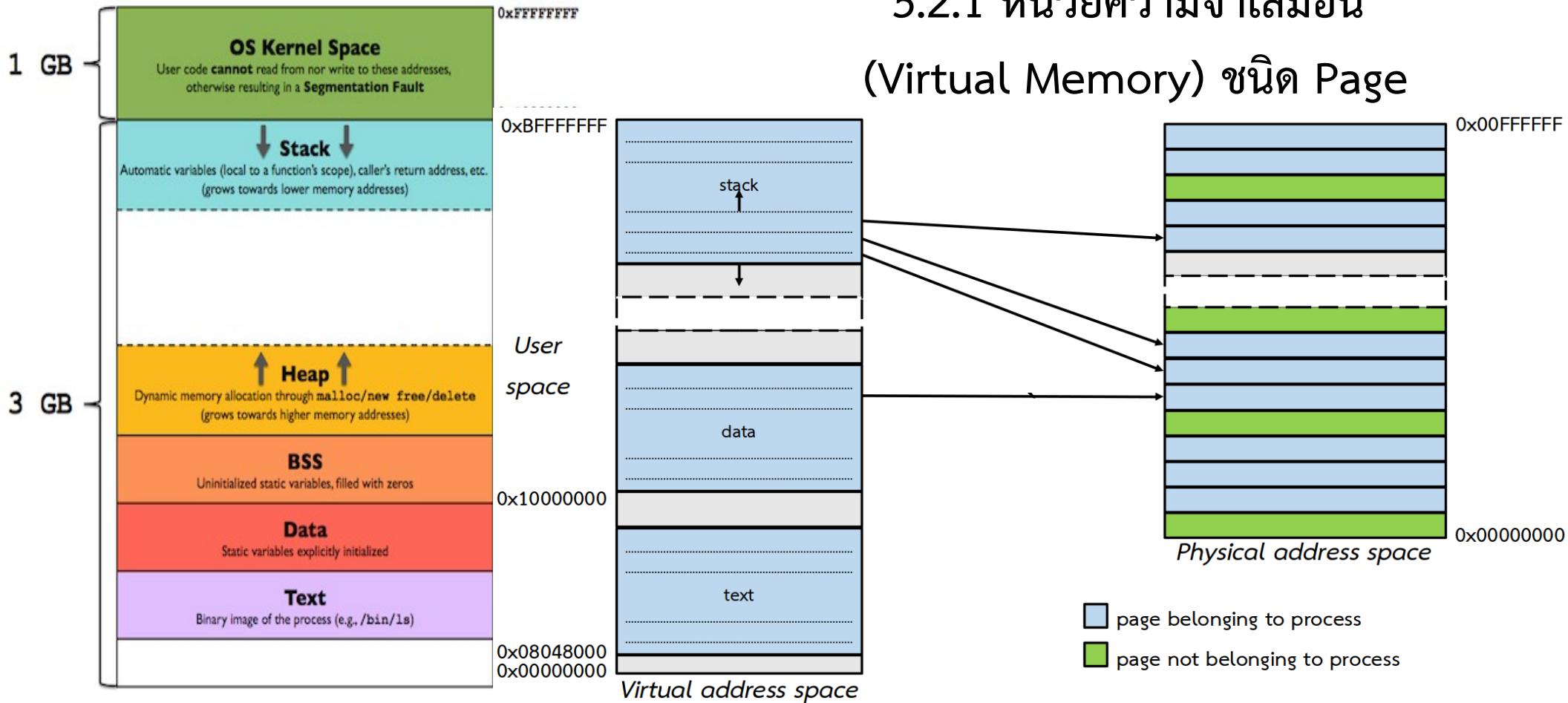
อุปกรณ์เก็บรักษาข้อมูล

การเข้มข้นของรีจิสเตอร์ แคช หน่วยความจำเสมือนของโปรเซสหลัก

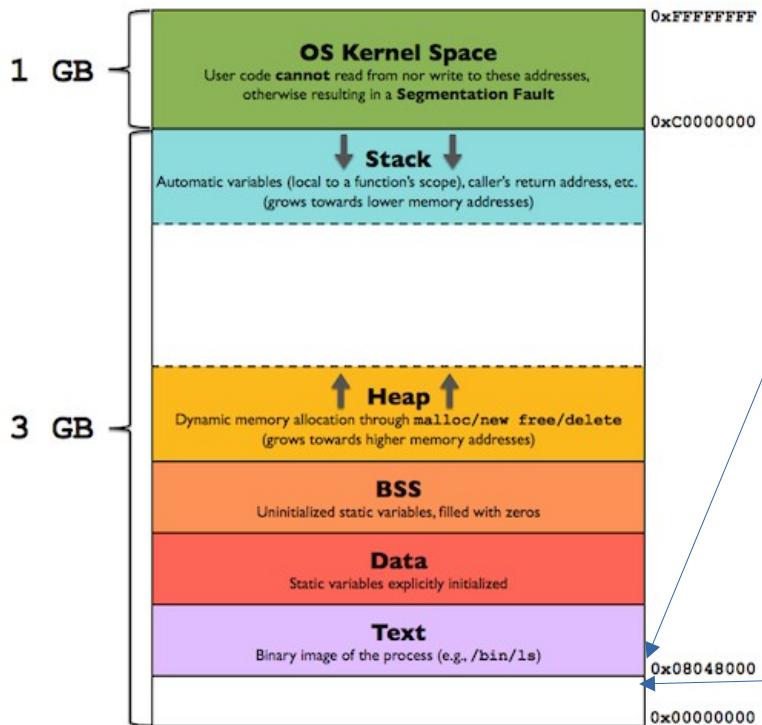


5.2.1 หน่วยความจำเสมือน

(Virtual Memory) ชนิด Page



5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page



Page 0x0804_8

- 0x0804_8000 – 0x0804_8003 คำสั่งที่ 00
- 0x0804_8004 – 0x0804_8007 คำสั่งที่ 01
- 0x0804_8008 – 0x0804_800B คำสั่งที่ 02
- ...
- 0x0804_80FC – 0x0804_80FF คำสั่งที่ 63
- 0x0804_8100 – 0x0804_8103 คำสั่งที่ 64
- 0x0804_8104 – 0x0804_8107 คำสั่งที่ 65
- 0x0804_8108 – 0x0804_810B คำสั่งที่ 66
- ...
- 0x0804_81FC – 0x0804_81FF คำสั่งที่ 127
- ...
- 0x0804_8FFC – 0x0804_8FFF คำสั่งที่ 1023

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page ของพรเซสตัวหนึ่ง

Page 0x0804_8

- 0x0804_8000 – 0x0804_8003 คำสั่งที่ 00
- 0x0804_8004 – 0x0804_8007 คำสั่งที่ 01
- 0x0804_8008 – 0x0804_800B คำสั่งที่ 02
- ...
- 0x0804_80FC – 0x0804_80FF คำสั่งที่ 63
- 0x0804_8100 – 0x0804_8103 คำสั่งที่ 64
- 0x0804_8104 – 0x0804_8107 คำสั่งที่ 65
- 0x0804_8108 – 0x0804_810B คำสั่งที่ 66
- ...
- 0x0804_81FC – 0x0804_81FF คำสั่งที่ 127
- ...
- 0x0804_8FFC – 0x0804_8FFF คำสั่งที่ 1023

Page 0x0804_9

- 0x0804_9000 – 0x0804_9003 คำสั่งที่ 1024+00
- 0x0804_9004 – 0x0804_9007 คำสั่งที่ 1024+01
- 0x0804_9008 – 0x0804_900B คำสั่งที่ 1024+02
- ...
- 0x0804_90FC – 0x0804_90FF คำสั่งที่ 1024+63
- 0x0804_9100 – 0x0804_9103 คำสั่งที่ 1024+64
- 0x0804_9104 – 0x0804_9107 คำสั่งที่ 1024+65
- 0x0804_9108 – 0x0804_910B คำสั่งที่ 1024+66
- ...
- 0x0804_91FC – 0x0804_91FF คำสั่งที่ 1024+127
- ...
- 0x0804_9FFC – 0x0804_9FFF คำสั่งที่ 1024+1023

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page ของพรเซสตัวหนึ่ง

Page 0x0804_A

- 0x0804_A000 – 0x0804_A003 คำสั่งที่ 2048+00
- 0x0804_A004 – 0x0804_A007 คำสั่งที่ 2048+01
- 0x0804_A008 – 0x0804_A00B คำสั่งที่ 2048+02
- ...
- 0x0804_A0FC – 0x0804_A0FF คำสั่งที่ 2048+63
- 0x0804_A100 – 0x0804_A103 คำสั่งที่ 2048+64
- 0x0804_A104 – 0x0804_A107 คำสั่งที่ 2048+65
- 0x0804_A108 – 0x0804_A10B คำสั่งที่ 2048+66
- ...
- 0x0804_A1FC – 0x0804_A1FF คำสั่งที่ 2048+127
- ...
- 0x0804_AFFC – 0x0804_AFFF คำสั่งที่ 2048+1023

Page 0x0804_B

- 0x0804_B000 – 0x0804_B003 คำสั่งที่ 3096+00
- 0x0804_B004 – 0x0804_B007 คำสั่งที่ 3096+01
- 0x0804_B008 – 0x0804_B00B คำสั่งที่ 3096+02
- ...
- 0x0804_B0FC – 0x0804_B0FF คำสั่งที่ 3096+63
- 0x0804_B100 – 0x0804_B103 คำสั่งที่ 3096+64
- 0x0804_B104 – 0x0804_B107 คำสั่งที่ 3096+65
- 0x0804_B108 – 0x0804_B10B คำสั่งที่ 3096+66
- ...
- 0x0804_B1FC – 0x0804_B1FF คำสั่งที่ 3096+127
- ...
- 0x0804_BFFC – 0x0804_BFFF คำสั่งที่ 3096+1023

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page ของพรเซสตัวหนึ่ง

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

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

- Virtual Page Number = 0804_8 ถึง FFFF_F
- หน่วยความจำภายในภาพ มีขนาดเท่ากับ $2^{24} = 2^2 \times 2^{20} = 16\text{MB}$
- Physical Page Num = 000 ถึง FFF
- พื้นที่เสมือนแบ่งเป็น Kernel Space และ User Space โปรแกรมจะมีเซ็กเมนท์ต่างๆ

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page ของพรเซสตัวหนึ่ง

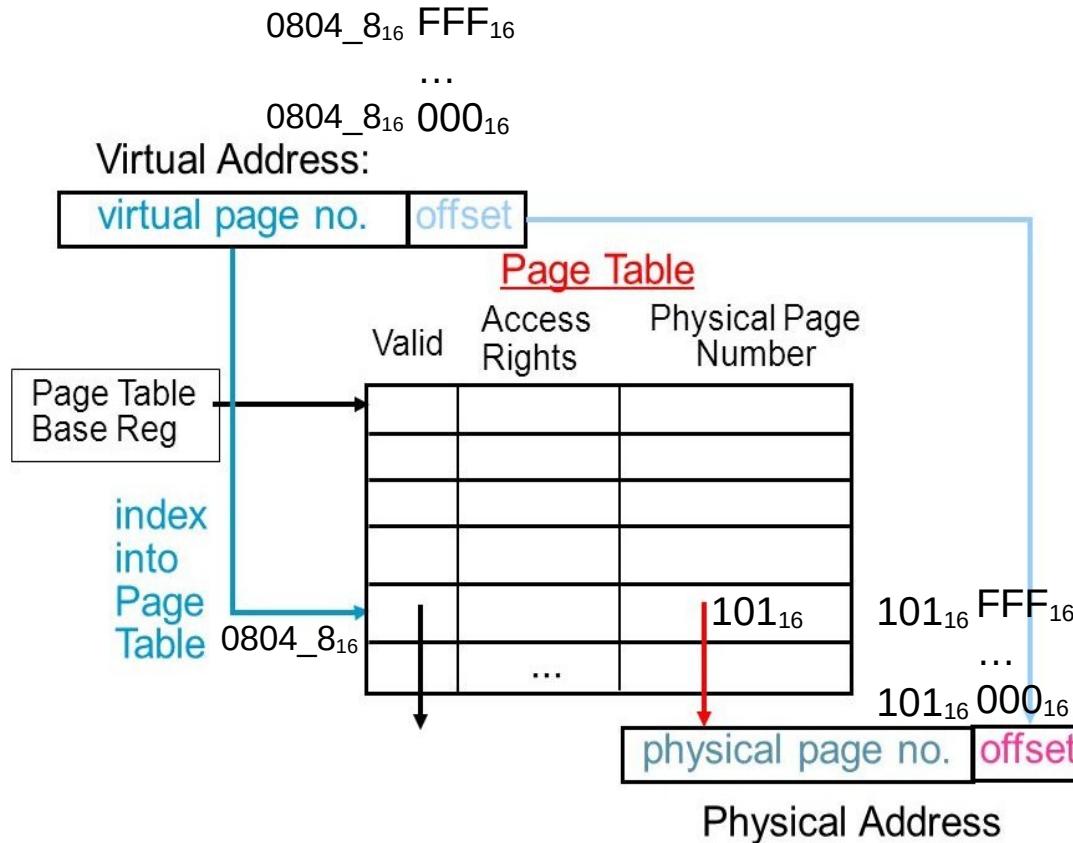
Virtual Page Num = 0x0804_8

- 0x0804_8000 – 0x0804_8003 คำสั่งที่ 00
- 0x0804_8004 – 0x0804_8007 คำสั่งที่ 01
- 0x0804_8008 – 0x0804_800B คำสั่งที่ 02
- ...
- 0x0804_80FC – 0x0804_80FF คำสั่งที่ 63
- 0x0804_8100 – 0x0804_8103 คำสั่งที่ 64
- 0x0804_8104 – 0x0804_8107 คำสั่งที่ 65
- 0x0804_8108 – 0x0804_810B คำสั่งที่ 66
- ...
- 0x0804_81FC – 0x0804_81FF คำสั่งที่ 127
- ...
- 0x0804_8FFC – 0x0804_8FFF คำสั่งที่ 1023

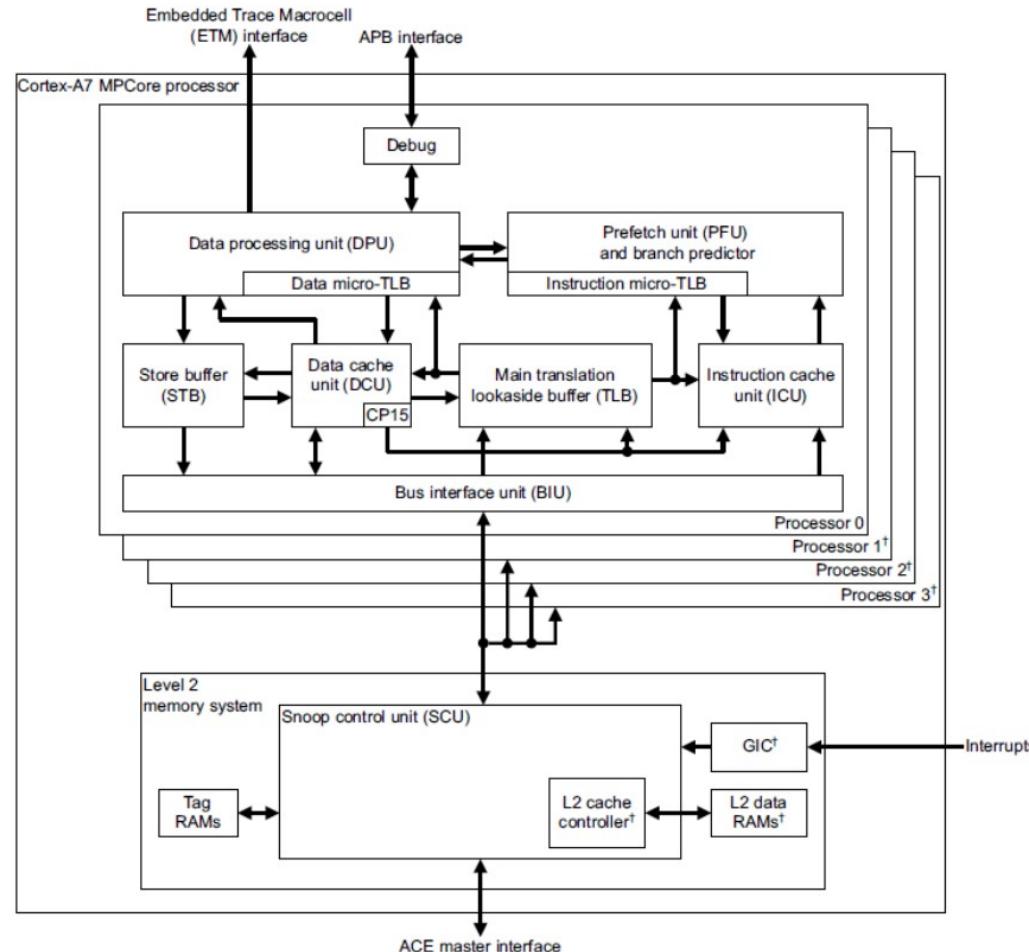
Phy Page Num = 0x101

- 0x10_1000 – 0x10_1003 คำสั่งที่ 00
- 0x10_1004 – 0x10_1007 คำสั่งที่ 01
- 0x10_1008 – 0x10_100B คำสั่งที่ 02
- ...
- 0x10_10FC – 0x10_10FF คำสั่งที่ 63
- 0x10_1100 – 0x10_1103 คำสั่งที่ 64
- 0x10_1104 – 0x10_1107 คำสั่งที่ 65
- 0x10_1108 – 0x10_110B คำสั่งที่ 66
- ...
- 0x10_11FC – 0x10_11FF คำสั่งที่ 127
- ...
- 0x10_1FFC – 0x10_1FFF คำสั่งที่ 1023

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page ของพรเซสตัวหนึ่ง



5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page ของพรเซสตัวหนึ่ง



5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page ของพรเซสตัวหนึ่ง

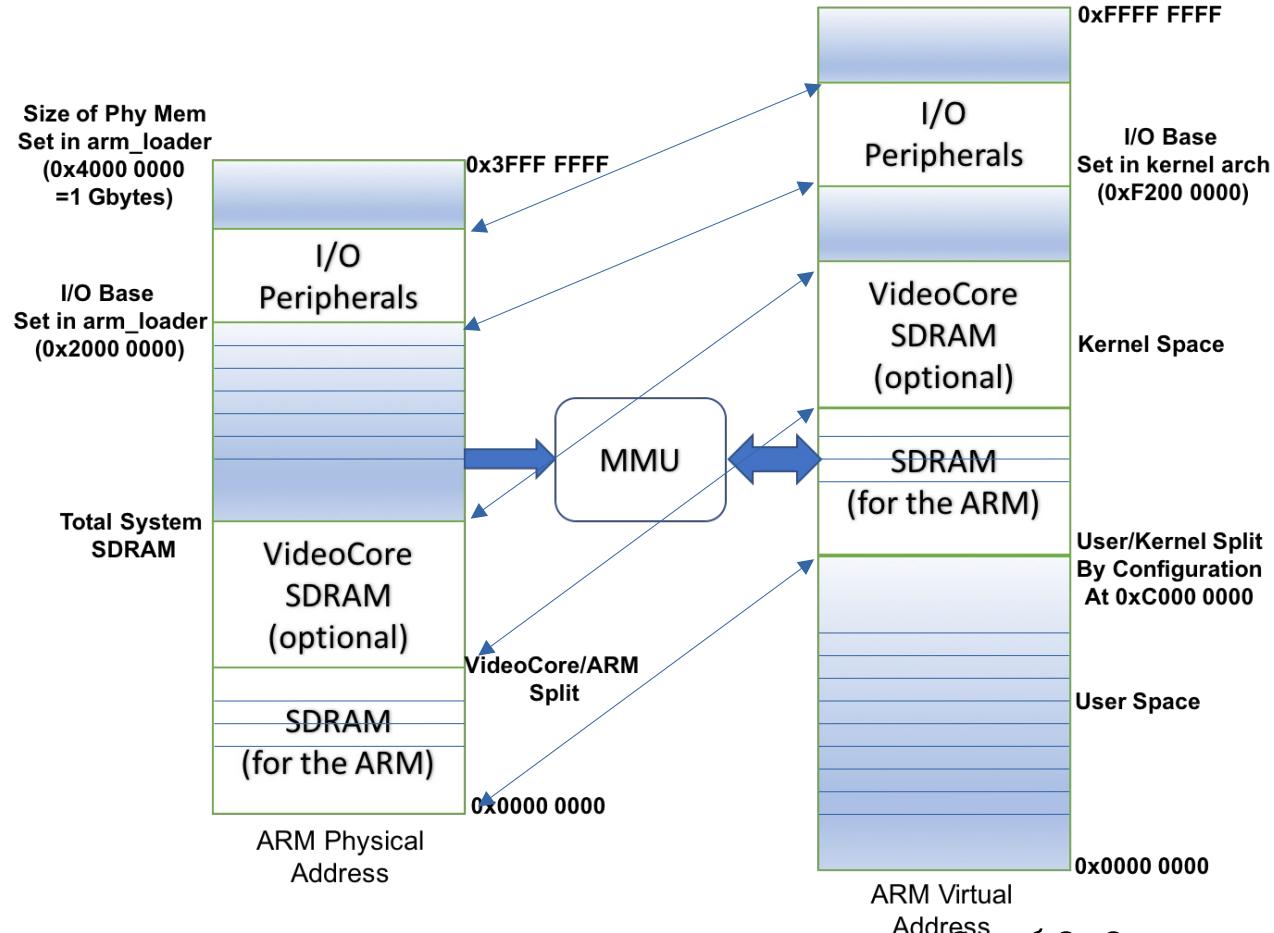
1. นำเลขเพจเวอร์ชวลใน PC (Program Counter) ไปสืบค้นหาค่าเลขเพจภายใน Instruction micro-TLB
(0.5-1 นาโนวินาที)
 - หากไม่เจอ เรียกว่า TLB มิส (Miss) วงจรจะนำหมายเลขเพจเวอร์ชวลไปสืบค้นใน Main TLB ต่อไปซึ่งจะใช้เวลาเพิ่มขึ้นอีก
 - หาก希ต ส่งค่าหมายเลขเพจภายในไปใช้งาน (2-4 นาโนวินาที)
 - หากมิส อีกรอบ จะต้องสืบค้นในเพจเทเบิลเป็นลำดับสุดท้ายซึ่งจะใช้เวลานานขึ้น
 - * หาก希ต ส่งค่าหมายเลขเพจจากเพจเทเบิลไปใช้งาน (หลักสิบนาโนวินาที)
 - * หากมิส อีกรอบ เรียกว่า เพจฟอลท์ (Page Fault) แสดงว่าเครื่องเนลยังไม่ได้อ่านคำสั่งที่ต้องการจากเพจในอุปกรณ์เก็บรักษาข้อมูลมาบรรจุ ในหน่วยความจำภายในซึ่งจะใช้เวลานานที่สุด (เกิน 100 นาโนวินาที) รายละเอียดเพิ่มเติมที่ [Tanenbaum and Bos \(2014\)](#) และ [wikipedia](#)

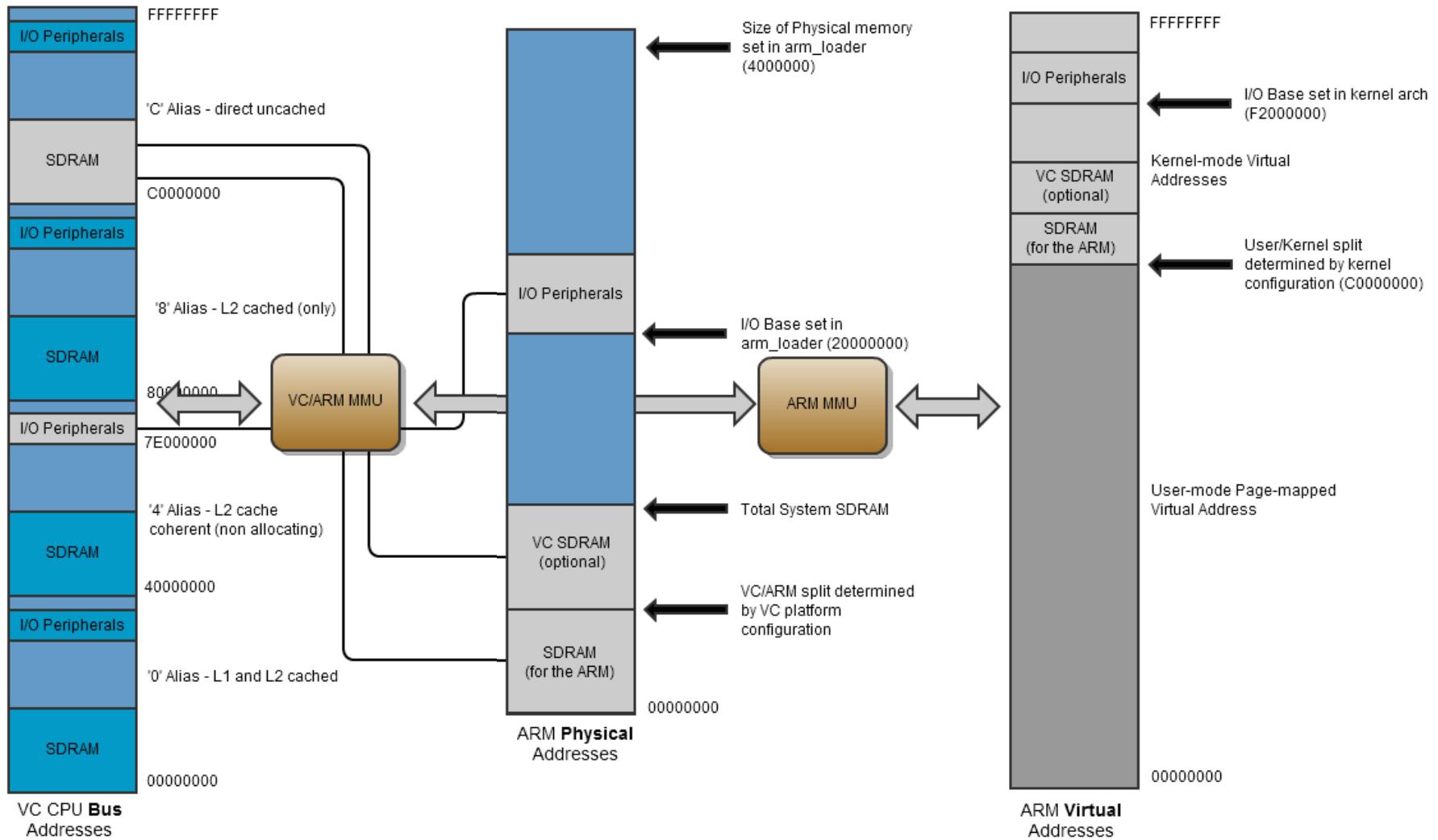
5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page ของprocessor ตัวหนึ่ง

2. นำเลขเพจ กายภาพมาเข้าม กับค่าอффเซตเพื่อนำไปสืบค้นคำสั่งใน ICU ซึ่งเทียบเท่าแแคชคำสั่ง ลำดับที่ 1

- หากเจอ เรียกว่า เกิดแแคชชิตที่ลำดับที่ 1 (L1 Cache Hit) และนำคำสั่งส่งกลับไปยังซีพีयูต่อไปซึ่งจะใช้เวลาสั้นที่สุด (0.5 - 1 นาโนวินาที)
- หากไม่เจอ เรียกว่า เกิดแแคชมิสที่ลำดับที่ 1 (L1 Cache Miss) นำแออดเดรสภายในภาพไปค้นหาคำสั่งในแแคชลำดับที่ 2
 - หากมี ที่ลำดับที่ 2 (L2 Cache Hit) และนำคำสั่งส่งกลับไปยังซีพีyu และแแคชลำดับที่ 1 ซึ่งจะใช้เวลานานขึ้น (2 - 4 นาโนวินาที)
 - หากไม่มี ที่ลำดับที่ 2 (L2 Cache Miss) วงจรจะนำแออดเดรสภายในภาพไปค้นหาคำสั่งในหน่วยความจำหลัก (SDRAM) ผ่านทางวงจรเชื่อมต่อ กับหน่วยความจำ SDRAM ซีพีyu จะต้องใช้เวลารอ (นานกว่า สิบนาโนวินาที) เพื่อที่คำสั่ง จะเดินทางมาจากหน่วยความจำภายในภาพ คำสั่งที่ได้จากแแคชลำดับต่าง ๆ และจาก SDRAM จะเป็นเลขฐานสองจำนวน 4-8 คำสั่ง คิดเป็นความยาว 128-256 บิต ซึ่งอยู่กับความกว้างของแแคชแต่ละบล็อก ซึ่งจะอธิบายในหัวข้อถัดไป

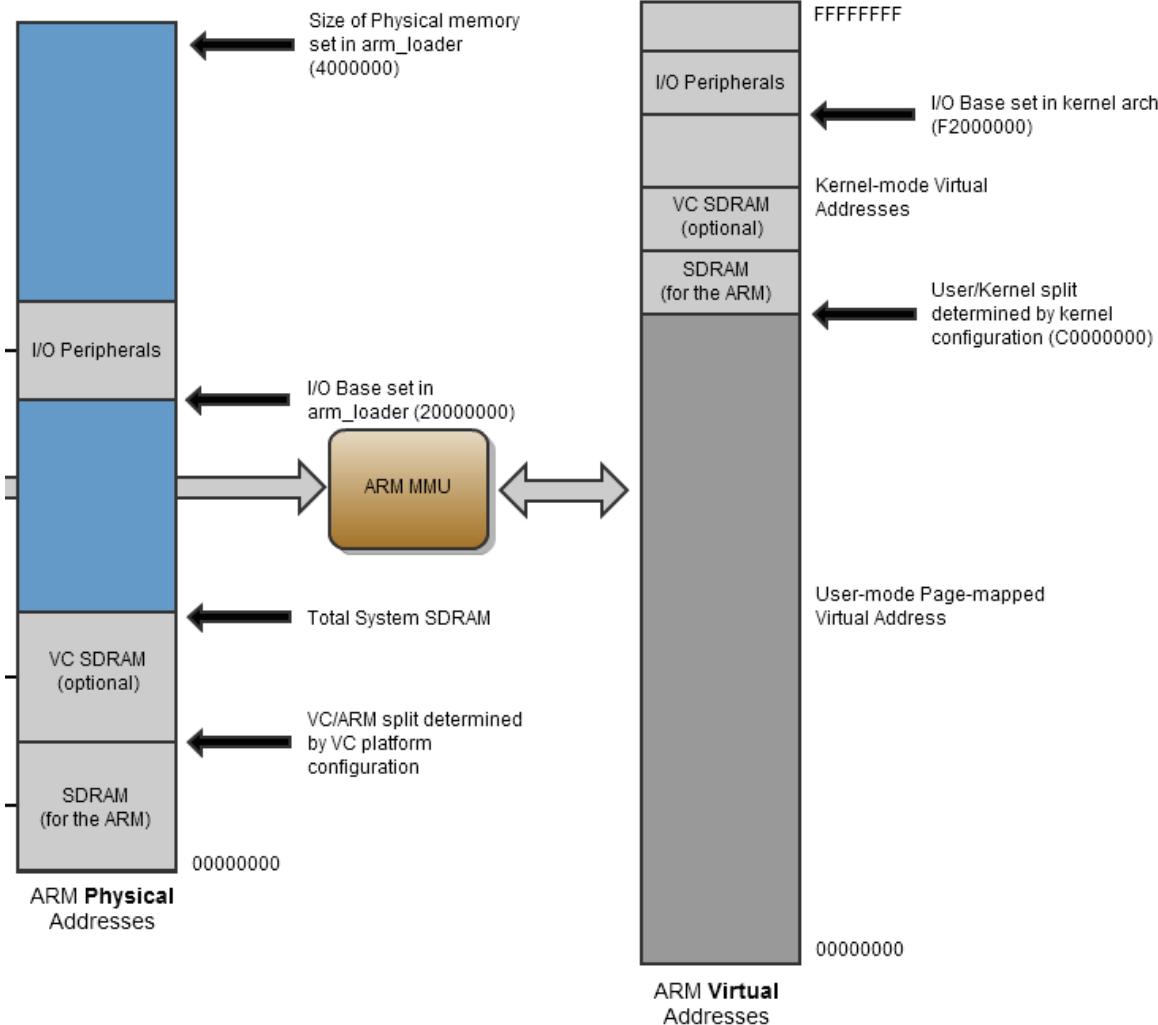
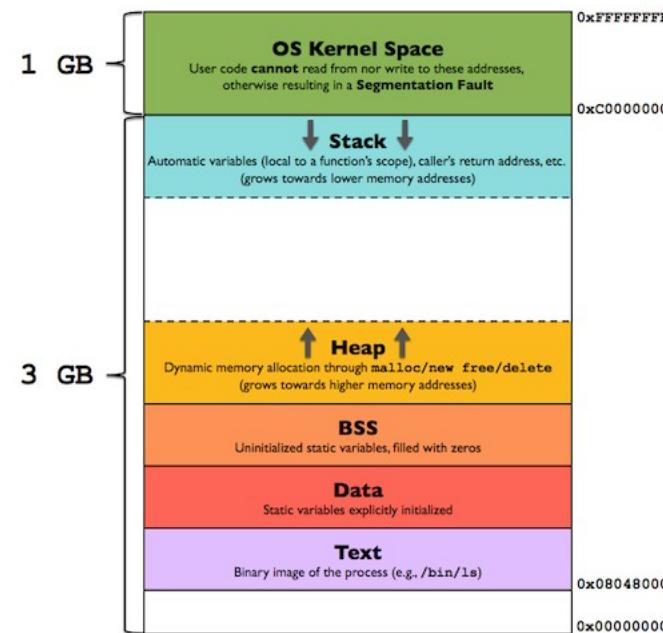
5.2.2 หน่วยความจำเสมือน (Virtual Memory) ของบอร์ด 32 bit Pi OS





5.2.2 หน่วยความจำเสมือน (Virtual Memory)

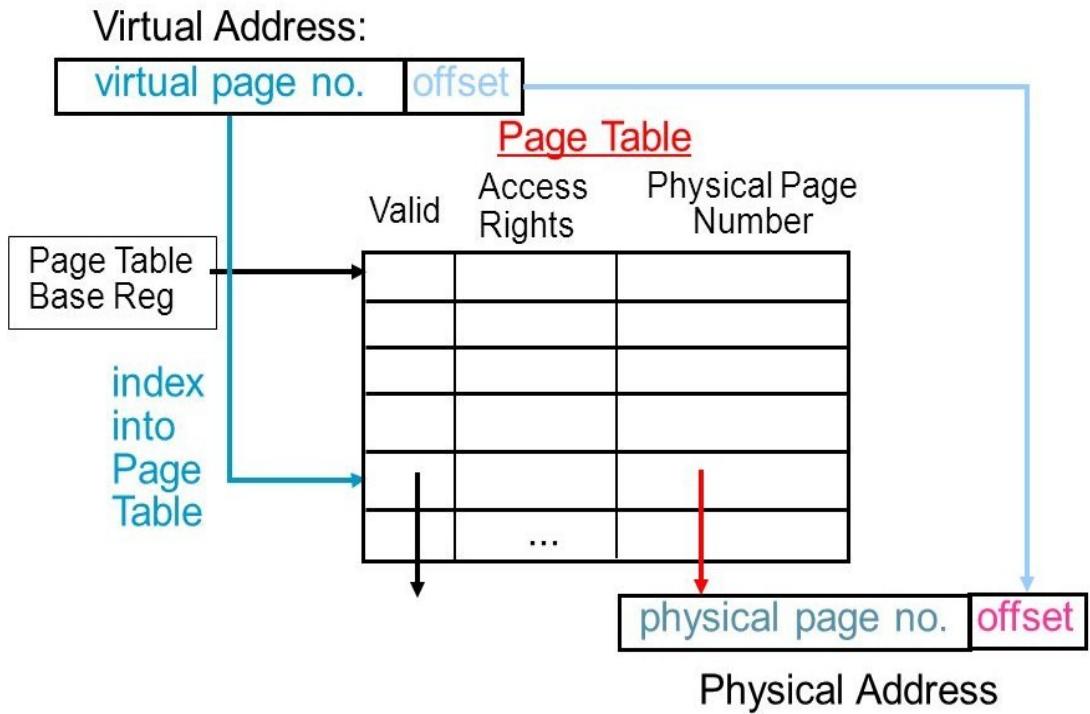
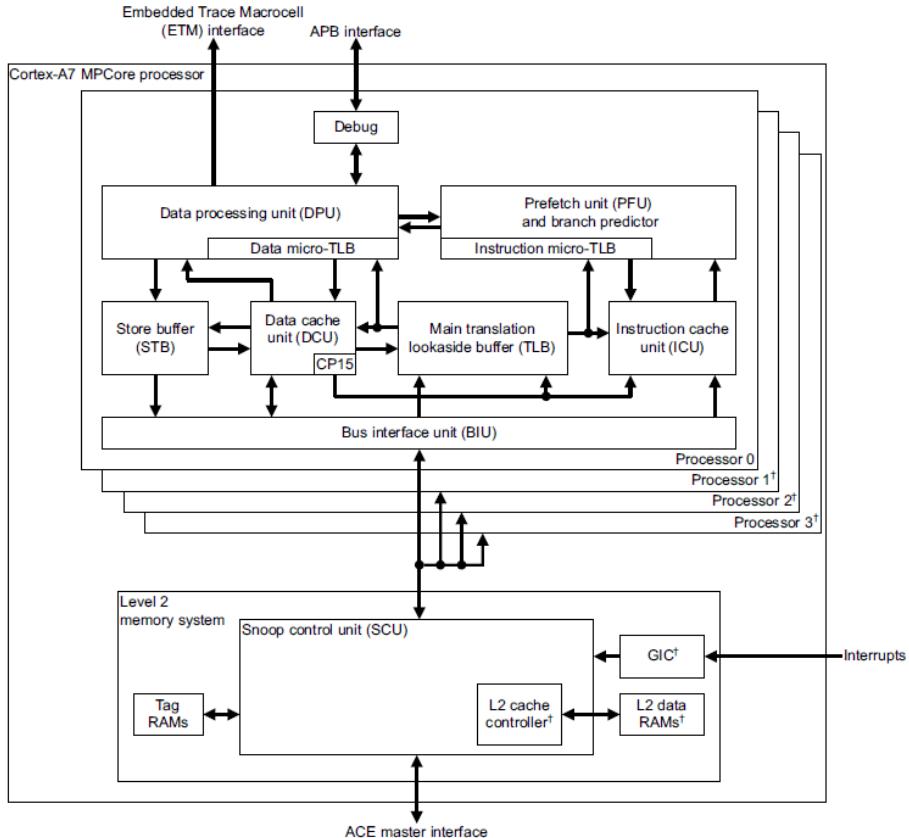
ของบอร์ด Pi3



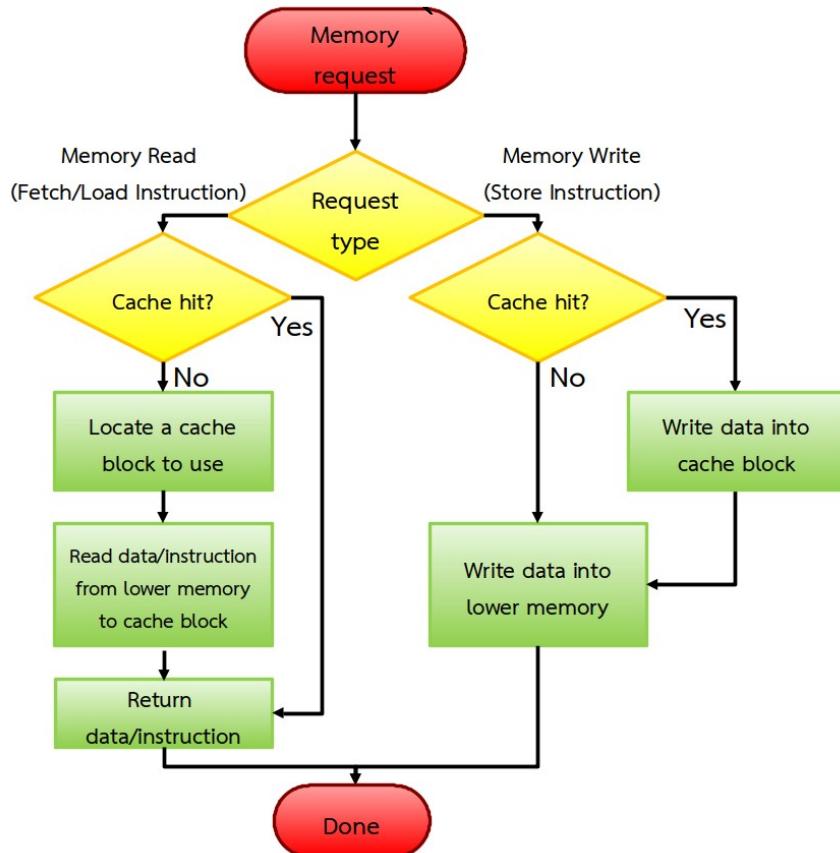
5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)

- แคชมีความจุน้อยแต่ทำงานเร็ว เพราะใช้เทคโนโลยี SRAM และสามารถบรรจุอยู่ในแผ่นซิลิกอนเดียวกับซีพียูคอร์
- แคชมีหลายชั้น ชั้นบนสุดของแต่ละคอร์ เรียกว่า ชั้น L1 ทำหน้าที่แยกจากกัน คือ
 - แคชชั้น L1 Instruction เป็นพื้นที่ของ Text Segment เก็บคำสั่ง
 - แคชชั้น L1 Data Cache เป็นพื้นที่อื่นๆ เก็บข้อมูลที่ไม่ใช่ Text Segment
 - แคชชั้น L2 จะมีความจุมากขึ้นและเก็บคำสั่งและข้อมูลจากทุกๆ เซ็กเมนท์ และคอร์ต่างๆ ใช้งานร่วมกัน
- บางพื้นที่ของหน่วยความจำเสมือนใช้งานผ่านแคช บางพื้นที่จะไม่ใช้งานผ่านแคชแม้แต่ชั้นเดียว
- แคชใช้ Phy Address ในการทำงาน เพราะจะไม่ซ้ำช้อนกับ Virtual Address

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ARM Cortex A



5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)



- Memory Request
 - Instruction Fetch
 - Data Read (Load) / Write (Store)
- Cache Hit? Miss? พิจารณาจากอะไร
 - ใช้ Phy Address ในการเข้าถึง เพราะเลขจะไม่ซ้ำเมื่อน Virtual Address
- ชนิดของแคช
- พฤติกรรมการใช้งาน
 - Instruction: Loop, Loop Body
 - Data: Loop Index, Loop Data ($a[j]$, $i++$)

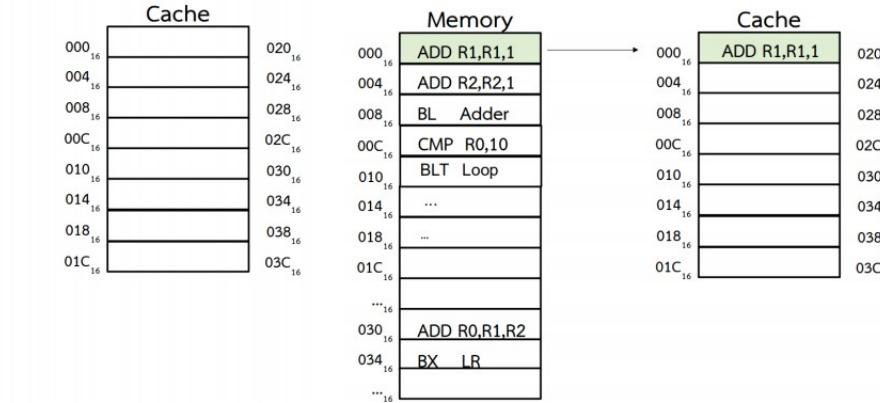
5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)

PC Offset	เลbel	คำสั่ง	คอมเมนท์
000_{16}	Loop	ADD R1,R1,1	@ R1=R1+1
004_{16}		ADD R2,R2,1	@ R2=R2+1
008_{16}		BL Adder	@ call R0 = R1+R2
$00C_{16}$		CMP R0,10	@ Compare R0 with 10
010_{16}		BLT Loop	@ if Less Than, PC = Loop
014_{16}		...	
018_{16}		...	
$01C_{16}$...	
....			
030_{16}	Adder	ADD R0, R1, R2	@ R0=R1+R2
034_{16}		BX LR	@ Return R0
....			
FFC_{16}		...	

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ชนิด Direct Map

Memory	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	CMP R0,10
010 ₁₆	BLT Loop
014 ₁₆	...
018 ₁₆	...
01C ₁₆	
...	
030 ₁₆	ADD R0,R1,R2
034 ₁₆	BX LR
...	

Cache	
000 ₁₆	
004 ₁₆	
008 ₁₆	
00C ₁₆	
010 ₁₆	
014 ₁₆	
018 ₁₆	
01C ₁₆	
...	

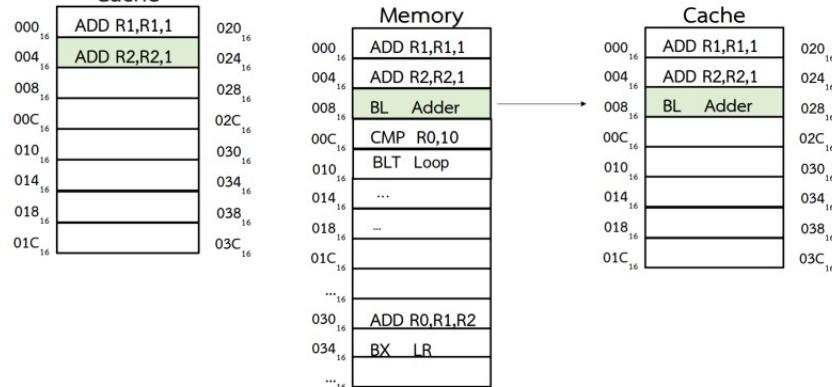


(a)

Memory	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	CMP R0,10
010 ₁₆	BLT Loop
014 ₁₆	...
018 ₁₆	...
01C ₁₆	
...	
030 ₁₆	ADD R0,R1,R2
034 ₁₆	BX LR
...	

Cache	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	
00C ₁₆	
010 ₁₆	
014 ₁₆	
018 ₁₆	
01C ₁₆	
...	

(b)



(c)

Memory	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	CMP R0,10
010 ₁₆	BLT Loop
014 ₁₆	...
018 ₁₆	...
01C ₁₆	
...	
030 ₁₆	ADD R0,R1,R2
034 ₁₆	BX LR
...	

Cache	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	
00C ₁₆	
010 ₁₆	
014 ₁₆	
018 ₁₆	
01C ₁₆	
...	

(d)

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ชนิด Direct Map

Memory
000 ₁₆ ADD R1,R1,1
004 ₁₆ ADD R2,R2,1
008 ₁₆ BL Adder
00C ₁₆ CMP R0,10
010 ₁₆ BLT Loop
014 ₁₆ ...
018 ₁₆ ...
01C ₁₆
... ₁₆
030 ₁₆ ADD R0,R1,R2
034 ₁₆ BX LR
... ₁₆

Cache
000 ₁₆ ADD R1,R1,1
004 ₁₆ ADD R2,R2,1
008 ₁₆ BL Adder
00C ₁₆
010 ₁₆ ADD R0,R1,R2
014 ₁₆
018 ₁₆
01C ₁₆
... ₁₆

Memory
020 ₁₆ ADD R1,R1,1
024 ₁₆ ADD R2,R2,1
028 ₁₆ BL Adder
02C ₁₆ CMP R0,10
030 ₁₆ BLT Loop
034 ₁₆ ...
038 ₁₆ ...
03C ₁₆
... ₁₆

Cache

(e)

(f)

Memory
000 ₁₆ ADD R1,R1,1
004 ₁₆ ADD R2,R2,1
008 ₁₆ BL Adder
00C ₁₆ CMP R0,10
010 ₁₆ BLT Loop
014 ₁₆ ...
018 ₁₆ ...
01C ₁₆
... ₁₆
030 ₁₆ ADD R0,R1,R2
034 ₁₆ BX LR
... ₁₆

Cache
000 ₁₆ ADD R1,R1,1
004 ₁₆ ADD R2,R2,1
008 ₁₆ BL Adder
00C ₁₆ CMP R0,10
010 ₁₆ ADD R0,R1,R2
014 ₁₆ BX LR
018 ₁₆
01C ₁₆
... ₁₆

Memory
020 ₁₆ ADD R1,R1,1
024 ₁₆ ADD R2,R2,1
028 ₁₆ BL Adder
02C ₁₆ CMP R0,10
030 ₁₆ BLT Loop
034 ₁₆ ...
038 ₁₆ ...
03C ₁₆
... ₁₆

Cache

(g)

(h)

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ชนิด Direct Map

ผู้อ่านจะสังเกตเห็นว่า คำสั่งจากหน่วยความจำภายในอยู่ 2 ตำแหน่งถูกแมปให้ใช้แคชบล็อกเดียวกัน เช่น แคชตำแหน่งที่ 000_{16} จะเป็นเป้าหมายของหน่วยความจำตำแหน่งที่ 000_{16} และ 020_{16} เป็นต้น ยิ่งไปกว่านั้น แคชลำดับที่ 1 ที่ใช้งานในซีพียูต่างๆ มีความจุน้อยกว่าหน่วยความจำภายในมาก เพราะหน่วยความจำภายในของบอร์ด Pi3 มีขนาดอย่างน้อย 1×2^{30} ไบท์ หรือ 1 กิกะไบท์ ในขณะที่ขนาดแคชลำดับที่ 1 มีขนาดเล็กเพียง 16×2^{10} ไบท์ หรือ 16 กิโลไบท์ คิดเป็น $2^{16} : 1$ ทำให้เกิดการแข่งขัน (Contention) เป็นแบบ Many to One ดังนั้นเมื่อแคชมีขนาดเล็ก อัตราการแข่งขัน (Contention Rate) จะยิ่งเพิ่มสูงมากขึ้น ในทางปฏิบัติ ซีพียูต้องมีแคชหลายลำดับชั้นเพิ่มขึ้นเพื่อช่วยลดอัตราการแข่งขันนี้

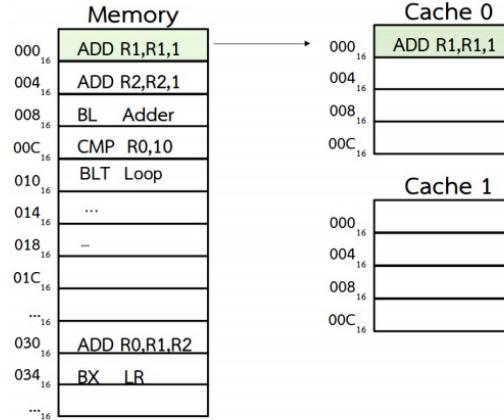
5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ชนิด 2-Way Set Associative

Memory
000 ₁₆ ADD R1,R1,1
004 ₁₆ ADD R2,R2,1
008 ₁₆ BL Adder
00C ₁₆ CMP R0,10
010 ₁₆ BLT Loop
014 ₁₆ ...
018 ₁₆ ...
01C ₁₆ ...
... ₁₆
030 ₁₆ ADD R0,R1,R2
034 ₁₆ BX LR
... ₁₆

Cache 0
000 ₁₆
004 ₁₆
008 ₁₆
00C ₁₆

Cache 1
000 ₁₆
004 ₁₆
008 ₁₆
00C ₁₆

(a)



(b)

Memory
000 ₁₆ ADD R1,R1,1
004 ₁₆ ADD R2,R2,1
008 ₁₆ BL Adder
00C ₁₆ CMP R0,10
010 ₁₆ BLT Loop
014 ₁₆ ...
018 ₁₆ ...
01C ₁₆ ...
... ₁₆
030 ₁₆ ADD R0,R1,R2
034 ₁₆ BX LR
... ₁₆

(c)

Memory
000 ₁₆ ADD R1,R1,1
004 ₁₆ ADD R2,R2,1
008 ₁₆ BL Adder
00C ₁₆ CMP R0,10
010 ₁₆ BLT Loop
014 ₁₆ ...
018 ₁₆ ...
01C ₁₆ ...
... ₁₆
030 ₁₆ ADD R0,R1,R2
034 ₁₆ BX LR
... ₁₆

(d)

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ชนิด 2-Way Set Associative

Memory	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	CMP R0,10
010 ₁₆	BLT Loop
014 ₁₆	...
018 ₁₆	...
01C ₁₆	
... ₁₆	
030 ₁₆	ADD R0,R1,R2
034 ₁₆	BX LR
... ₁₆	

Cache 0	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	
010 ₁₆	
014 ₁₆	
018 ₁₆	
01C ₁₆	
... ₁₆	
030 ₁₆	
034 ₁₆	
... ₁₆	

Cache 1

(e)

Memory	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	CMP R0,10
010 ₁₆	BLT Loop
014 ₁₆	...
018 ₁₆	...
01C ₁₆	
... ₁₆	
030 ₁₆	ADD R0,R1,R2
034 ₁₆	BX LR
... ₁₆	

Cache 0	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	
010 ₁₆	
014 ₁₆	
018 ₁₆	
01C ₁₆	
... ₁₆	
030 ₁₆	
034 ₁₆	
... ₁₆	

Cache 1

(f)

Memory	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	CMP R0,10
010 ₁₆	BLT Loop
014 ₁₆	...
018 ₁₆	...
01C ₁₆	
... ₁₆	
030 ₁₆	ADD R0,R1,R2
034 ₁₆	BX LR
... ₁₆	

Cache 0	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	CMP R0,10
010 ₁₆	
014 ₁₆	
018 ₁₆	
01C ₁₆	
... ₁₆	
030 ₁₆	
034 ₁₆	
... ₁₆	

Cache 1

(g)

Memory	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	CMP R0,10
010 ₁₆	BLT Loop
014 ₁₆	...
018 ₁₆	...
01C ₁₆	
... ₁₆	
030 ₁₆	ADD R0,R1,R2
034 ₁₆	BX LR
... ₁₆	

Cache 0	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	CMP R0,10
010 ₁₆	BLT Loop
014 ₁₆	
018 ₁₆	
01C ₁₆	
... ₁₆	
030 ₁₆	
034 ₁₆	
... ₁₆	

Cache 1

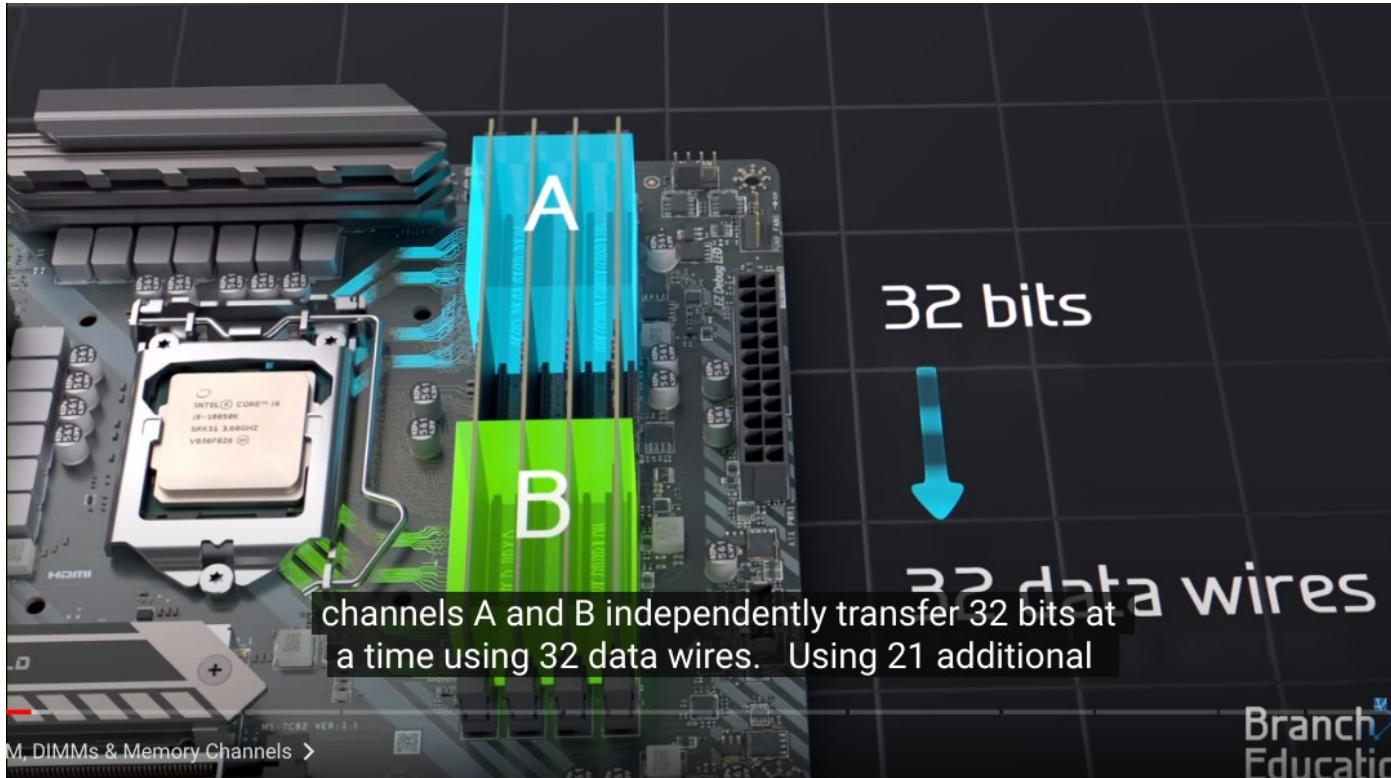
(h)

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ชนิด 2-Way Set Associative

ผู้อ่านจะสังเกตเห็นว่า แคชชนิด SA มีการทำงานคล้ายกับแคชชนิด DM แต่มีจำนวน 2 เวյ์ นั่นคือ แคชหมายเลขบล็อกเดียวกันมี 2 ทางเลือก คือ Cache 0 และ Cache 1 ขึ้นอยู่กับว่า เว yi ได้ว่าง หากไม่ว่างทั้งคู่ แคชจะตัดสินใจให้เขียนทับ เรียกว่า Cache Replacement Algorithm หากใช้อัลกอริธึมการตัดสินใจที่เรียกว่า Least Recently Used แคชจะเขียนทับตำแหน่งในเวย์ที่ไม่ได้ใช้งานล่าสุด จึงเห็นได้ว่า แคชชนิดนี้เป็นการขยายการทำงานของแคชชนิด DM ให้มีความยืดหยุ่นมากขึ้น ปัจจุบัน ซีพียูนิยมใช้แคชชนิด SA ขนาด $N = 8-16$ เวย์เพื่อประสิทธิภาพที่ดีกว่า

หลักการของแคชมีบทบาทต่อประสิทธิภาพของระบบโดยองค์รวม แคชชนิด N-way Set Associative ได้รับความนิยม เนื่องจากมีโครงสร้างเหมือนกับแคชชนิด Direct Mapped และมีความสามารถคล้ายแคชชนิด Fully Associative มีการนำหลักการ Principle of Locality มาประยุกต์ใช้กับหน่วยความจำประเภทต่างๆ และหลากหลาย รวมถึงหลักการหน่วยความจำเสมือน หน่วยความจำเสมือนอาศัยการทำงานร่วมกันระหว่าง ฮาร์ดแวร์และระบบปฏิบัติการ เพื่อเพิ่มลำดับชั้นในการบริหารจัดการหน่วยความจำทุกลำดับชั้นดังได้กล่าวไปแล้ว

SRAM vs. DRAM

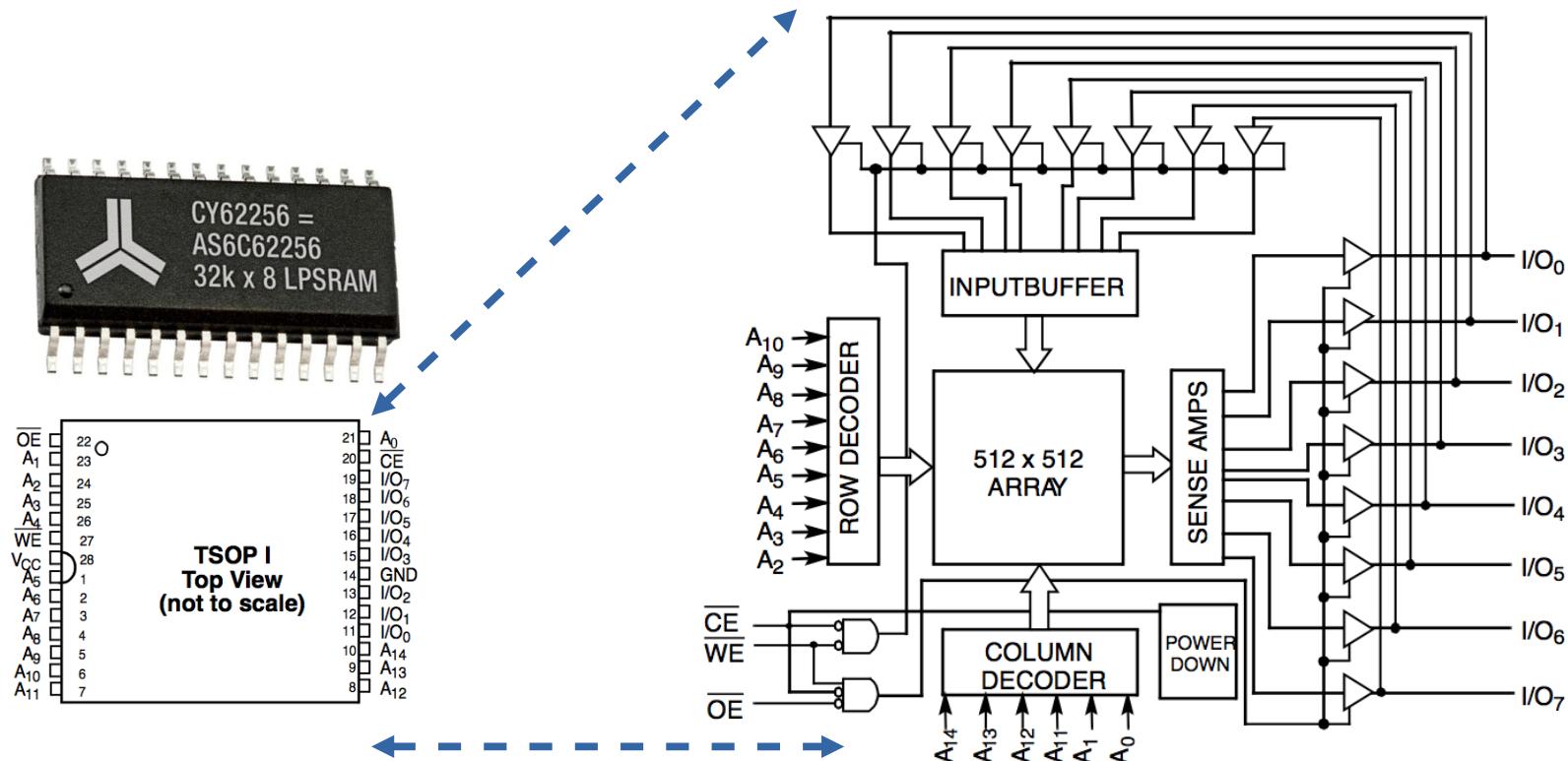


<https://www.youtube.com/watch?v=7J7X7aZvMXQ>

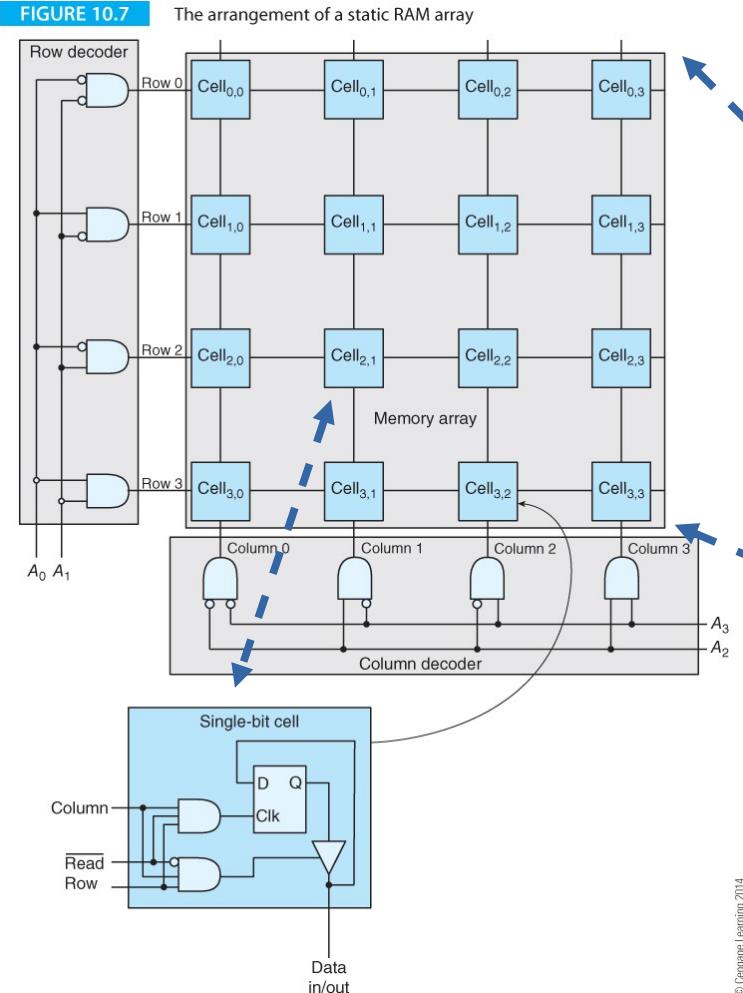
5.4 หน่วยความจำชนิดสแตติคแรม (Static RAM: SRAM)

- รีจิสเตอร์ R0-R15 แคชลำดับที่ 1 และ 2 ในชิป BCM2837 สร้างจากหน่วยความจำสแตติคแรมซึ่งมีโครงสร้างที่ไม่ซับซ้อนและสามารถออกแบบให้ผลิตพร้อมกับวงจรทرانซิสเตอร์ในซีพียู นอกจากนี้ หน่วยความจำสแตติคแรมนี้ยังนิยมใช้งานเป็นหน่วยความจำหลักภายในชิปไมโครคอนโทรลเลอร์ ที่ต้องการสมรรถนะตั้งถึงปานกลางโดยมีความจุหลายขนาด ตั้งแต่ 16 กิกะไบต์ ถึงหลายเมบิไบต์
- หน่วยความจำสแตติคแรม (Static RAM: SRAM) หมายเลข CY62256 เป็นกรนีศึกษาชิป CY62256 ใช้เทคโนโลยีการผลิตชนิด CMOS ในปี ค.ศ. 2002
 - } ใช้กับแหล่งจ่ายไฟตั้งแต่ 4.5 - 5.5 โวลท์
 - } รองรับการทำงานความเร็วสูง เนื่องจากใช้เวลาเข้าถึงน้อยเท่ากับ 55 นาโนวินาที
 - } ชิปบริโภคกำลังไฟน้อยโดยมีค่าสูงสุดเพียง 275 มิลลิวัตต์ระหว่างปฏิบัติงาน
 - } ชิปบริโภคกำลังไฟน้อยโดยมีค่าสูงสุดเพียง 28 มิลลิวัตต์ระหว่างไม่ทำงาน (Stand by)

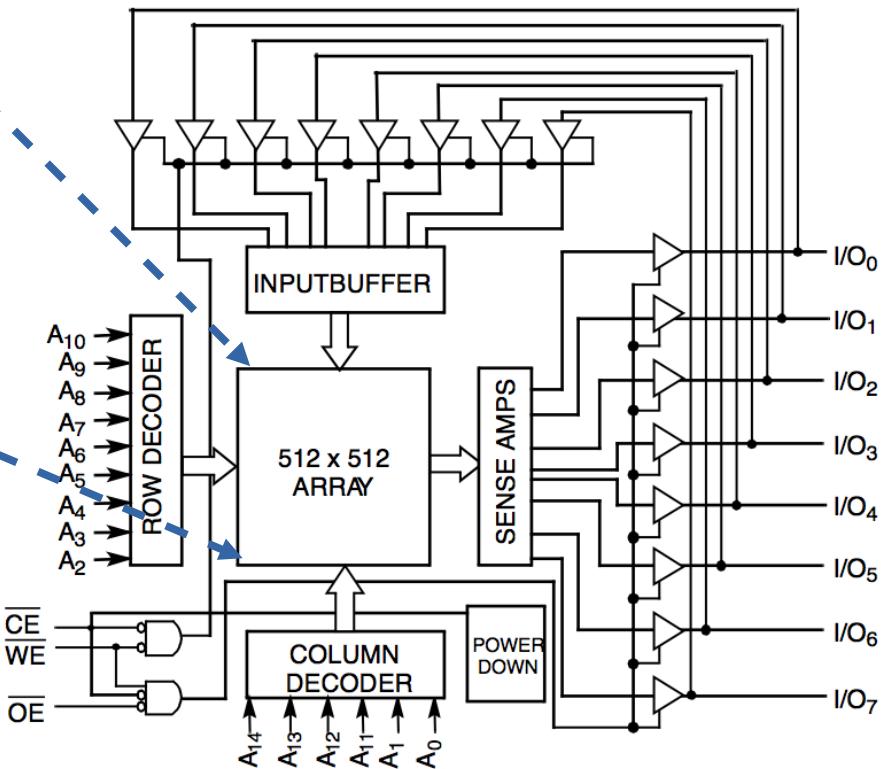
5.4 หน่วยความจำชนิดสแตติคแรม (Static RAM: SRAM)



5.4 หน่วยความจำชนิดสแตติคเรเม (Static RAM: SRAM)



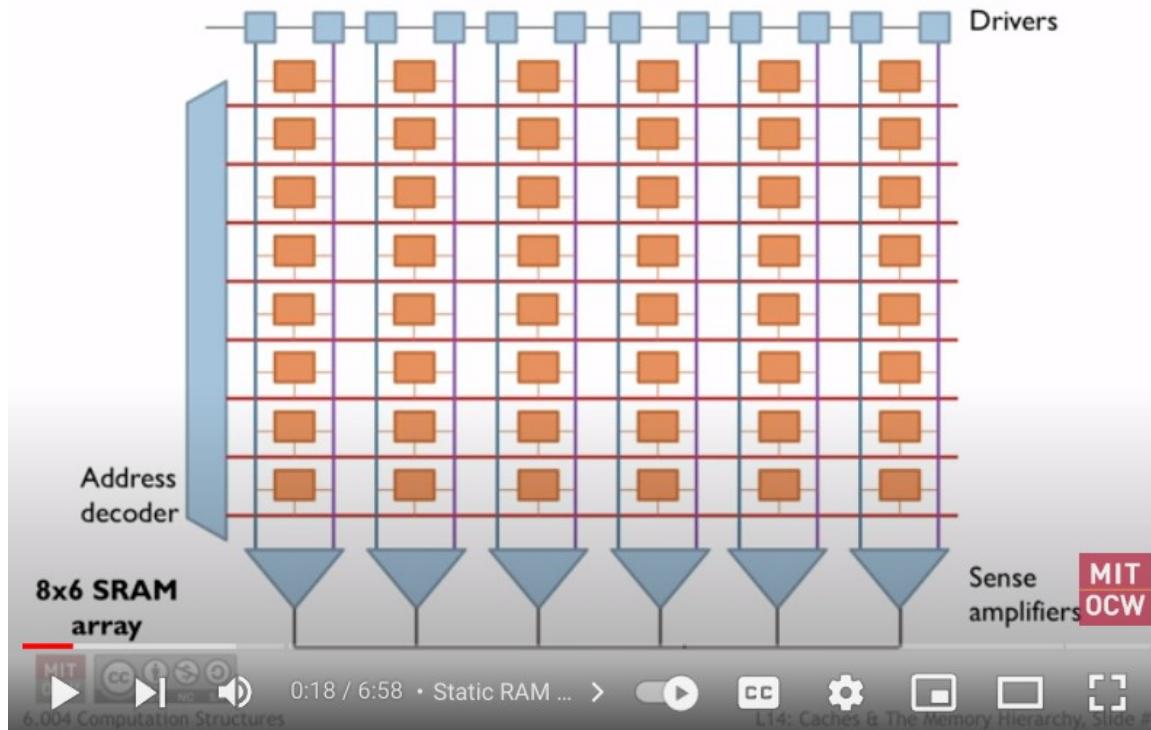
© Cengage Learning 2014



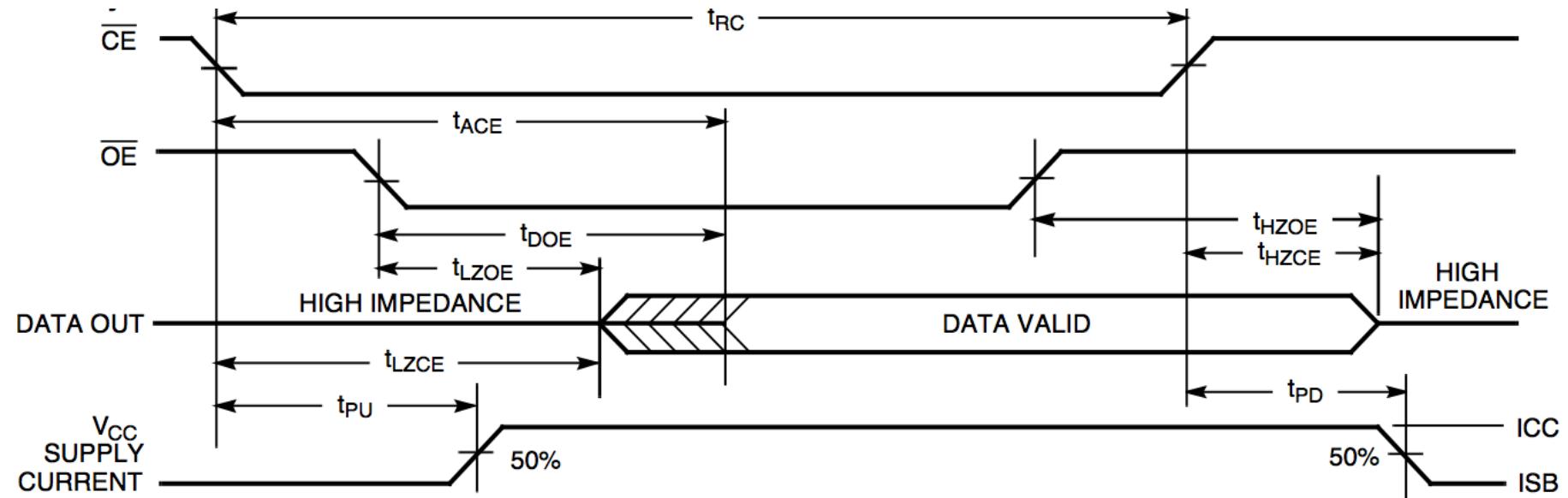
5.4 หน่วยความจำชนิด静态 RAM (Static RAM: SRAM): Read Cycle

https://www.youtube.com/watch?v=GBL28_Tw6UQ

Static RAM (SRAM)



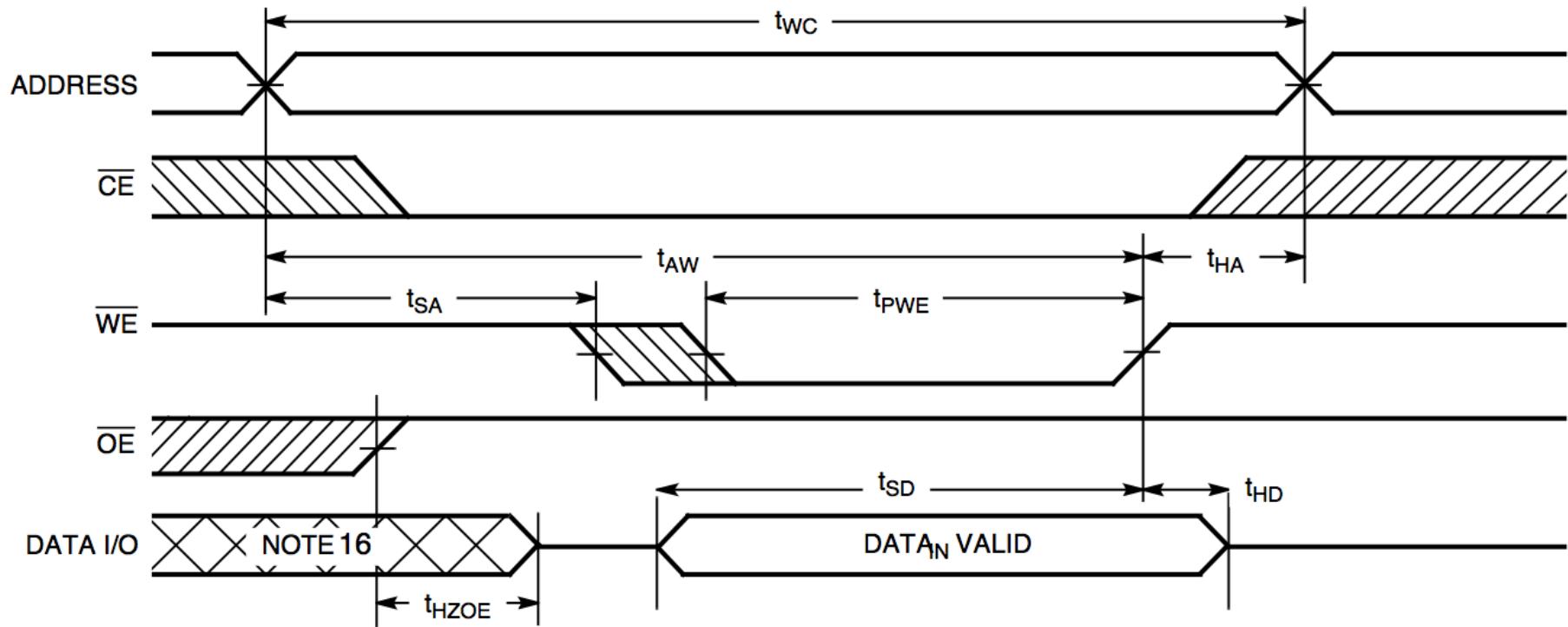
5.4 หน่วยความจำชั้นนิดสแตติกแรม (Static RAM: SRAM): Read Cycle



5.4 หน่วยความจำชนิด静态 RAM (Static RAM: SRAM): Read Cycle

- t_{ACE} เรียกว่า เวลาเข้าถึงข้อมูล หรือ Access Time โดยเริ่มนับจากเมื่อขาสัญญาณ \overline{CE} เปลี่ยนเป็น 0 จนข้อมูลที่ถูกต้องปรากฏ
- t_{RC} เรียกว่า คาบเวลาที่สั้นที่สุดในการอ่านข้อมูลจาก static RAM (Read Cycle Time) อย่างต่อเนื่อง โดย $t_{RC} > t_{ACE}$
- t_{HZOE} เรียกว่า เวลาที่ข้อมูลบนบัสข้อมูลยังถูกต้อง (Valid) เมื่อขาสัญญาณ \overline{OE} เปลี่ยนเป็น 1 แล้ว
- t_{HZCE} เรียกว่า เวลาที่ข้อมูลบนบัสข้อมูลยังถูกต้อง (Valid) เมื่อขาสัญญาณ \overline{CE} เปลี่ยนเป็น 1 แล้ว

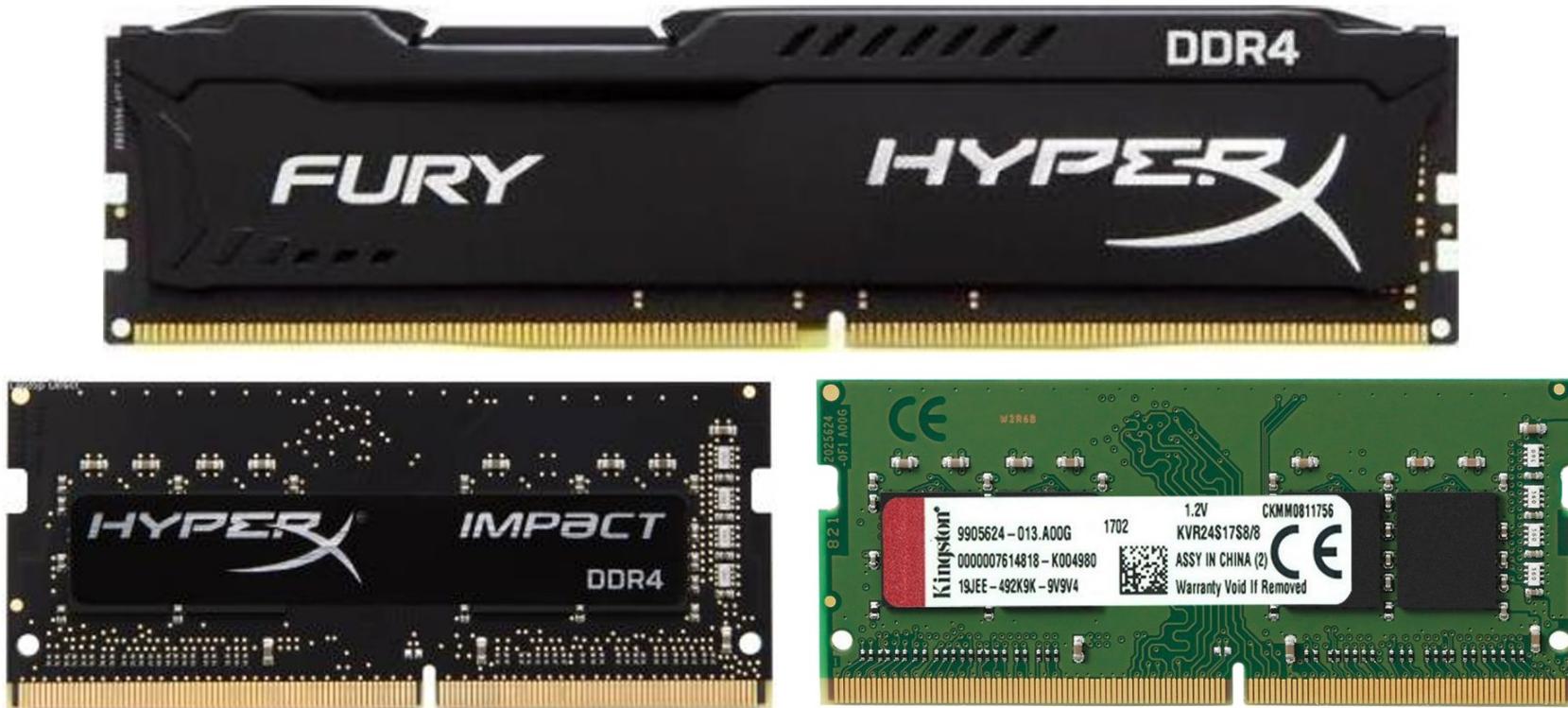
5.4 หน่วยความจำชนิดสแตติคแรม (Static RAM: SRAM): Write Cycle



5.4 หน่วยความจำชนิดสแตติคแรม (Static RAM: SRAM): Write Cycle

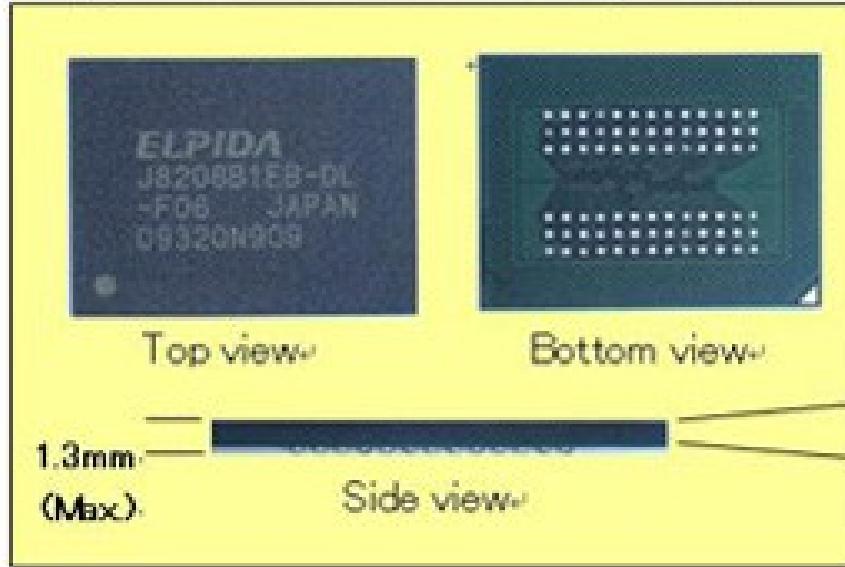
- t_{SD} เรียกว่า เวลาที่จะตรึงข้อมูลไว้ (Data Stable Time) ก่อน Write Enable (\overline{WE}) เปลี่ยนเป็น 1
- t_{HD} เรียกว่า เวลาที่จะตรึงข้อมูลไว้ (Data Hold Time) เมื่อ Write Enable (\overline{WE}) = 1 แล้ว
- t_{WC} เรียกว่า คาบเวลาที่สั้นที่สุดในการเขียนข้อมูลในสแตติคแรม (Write Cycle Time) อย่างต่อเนื่อง
- t_{AW} เรียกว่า เวลาเข้าถึงสำหรับการเขียน (Access Write Time)

5.5 หน่วยความจำหลักนิดไดนามิกแรม (Synchronous Dynamic RAM: DRAM): DDR4 SDRAM 16GiB for PC & 4GiB & 8GiB for Notebook

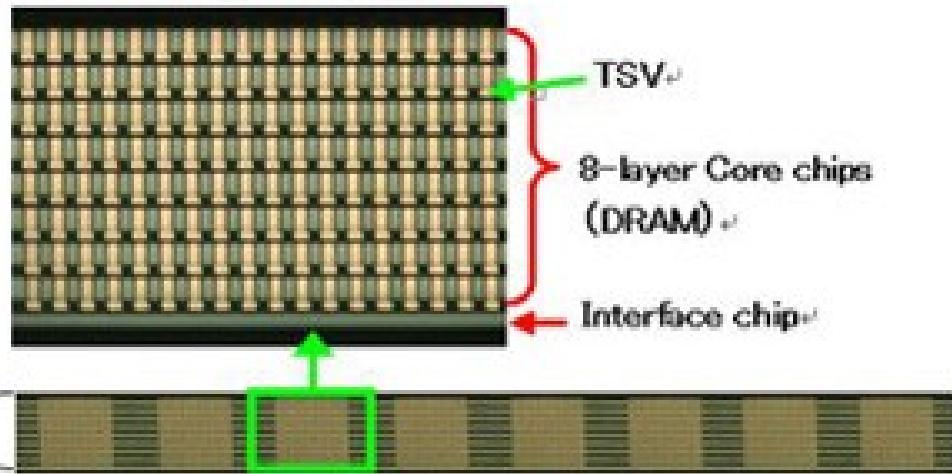


5.5 หน่วยความจำหลักชนิดซิงโครนัสไนดามิคแรม¹ (Synchronous Dynamic RAM: SDRAM)

8-Gigabit TSV DRAM Package Outline

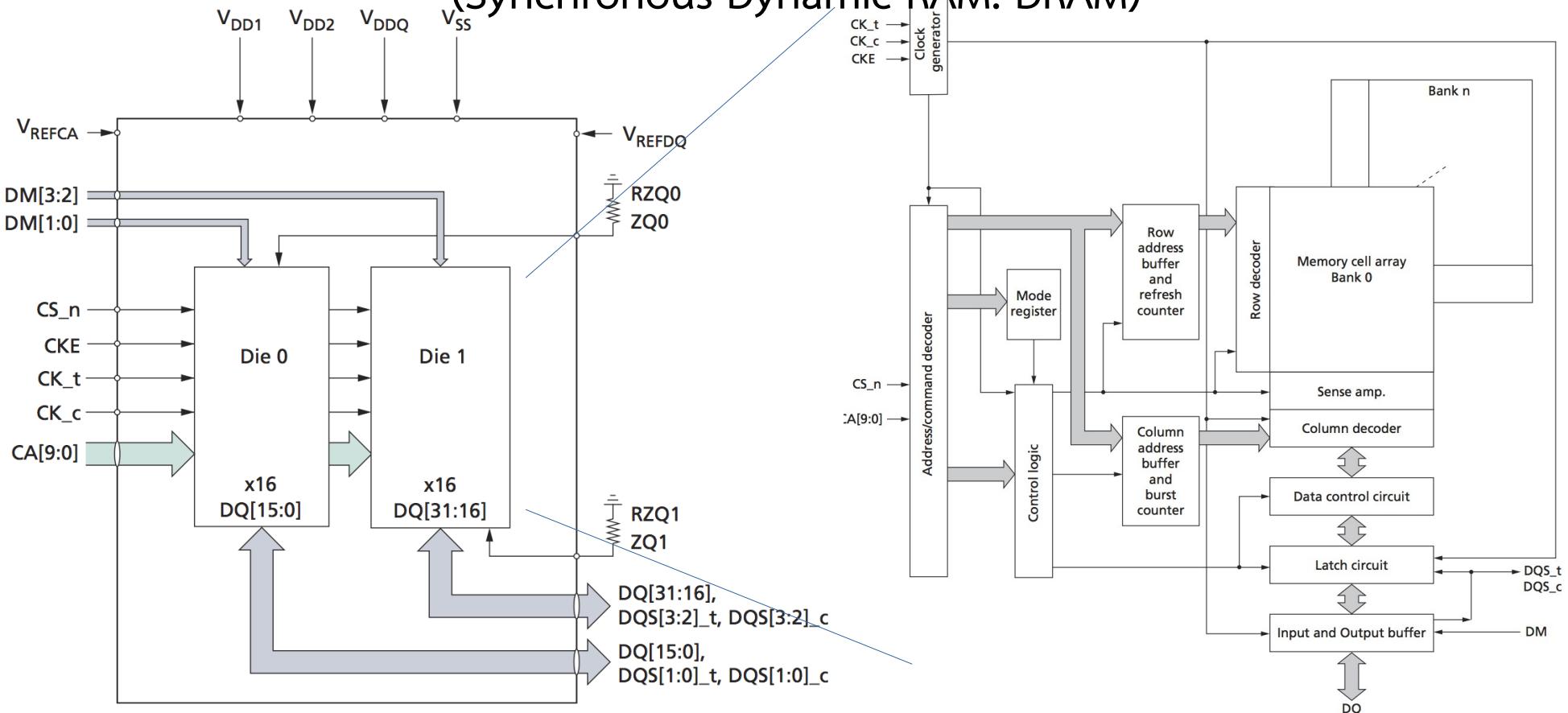


Cross-section



5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม

(Synchronous Dynamic RAM: DRAM)



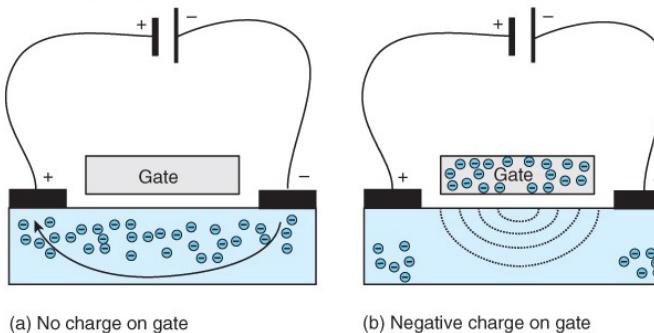
5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม (Synchronous Dynamic RAM: DRAM)

- 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 ต่อ 1 ชิพ 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 แต่ละแบงค์ประกอบด้วยวงจรถอดรหัสแอดเดรส (Address Decoder) ในแนวนอน (Row Decoder) และแนวตั้ง (Column Decoder)

5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม

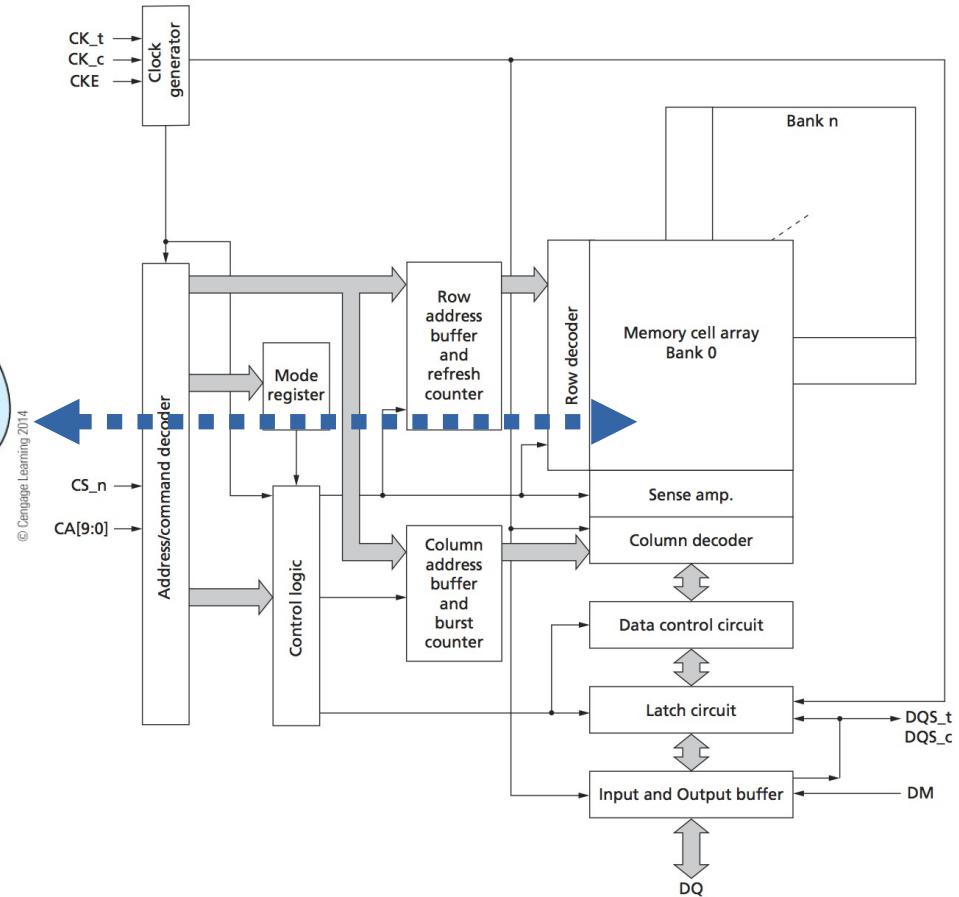
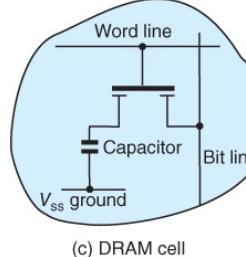
(Synchronous Dynamic RAM: DRAM): Memory Cell

FIGURE 10.16 The effect of a charged gate on electron flow



(a) No charge on gate

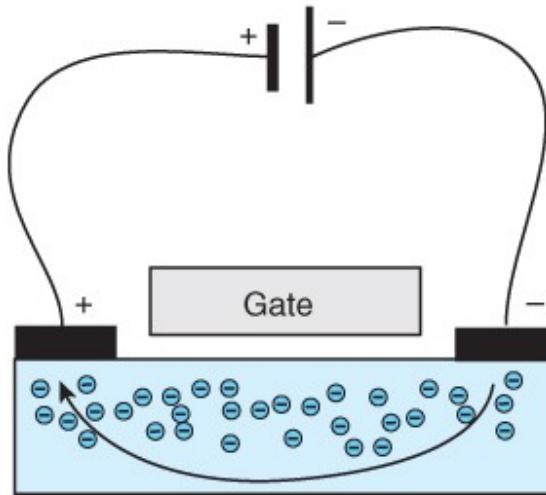
(b) Negative charge on gate



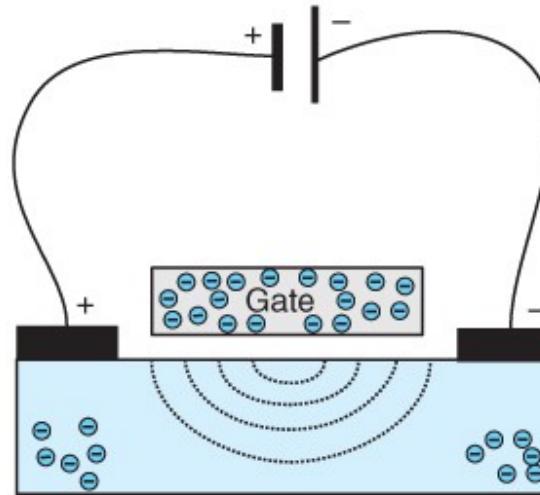
5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม^{(Synchronous Dynamic RAM: DRAM): Memory Cell}

FIGURE 10.16

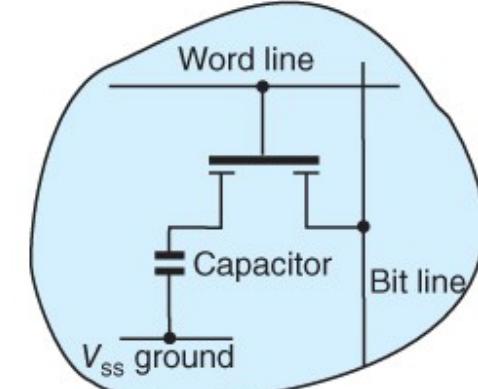
The effect of a charged gate on electron flow



(a) No charge on gate



(b) Negative charge on gate



(c) DRAM cell

© Cengage Learning 2014

5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม^(Synchronous Dynamic RAM: SDRAM)

แบ่งที่ 0 ถึง 7 แต่ละแบงค์ประกอบด้วยวงจรอドร์หัสแอดเดรส (Address Decoder) ในแนวอน (Row Decoder) และแนวตั้ง (Column Decoder) ภายในชิปประกอบด้วยขาสัญญาณต่างๆ เรียงตามลำดับความสำคัญ ดังนี้

- CS_n หรือ (Chip Select Not) หรือ \overline{CS} ใช้เปิด/ปิดการทำงานของชิป เพื่อช่วยประหยัดพลังงาน
- CKE (Clock Enable) เมื่อสัญญาณ $\overline{CS}=0$ เพื่อให้ชิปทำงาน หลังจากนั้น สัญญาณ CKE ใช้สำหรับเปิด/ปิดการทำงานของคล็อกที่จ่ายให้กับชิป DRAM นี้ ซึ่งมีความสามารถควบคุมสัญญาณ CKE=0 เพื่อพักการใช้งาน DRAM ชั่วคราวเพื่อช่วยประหยัดพลังงาน
- CK (Clock) และ CK# (Clock Not) คือ สัญญาณคล็อกสองสัญญาณที่มีเฟส (Phase) หรือขั้วตรงข้ามกัน เรียกว่า คู่ดิฟเฟอเรนเชียล (Differential Pair) หน่วยเป็นเมกะเฮิทซ์ สัญญาณคล็อกความถี่สูงสุด 400 เมกะเฮิทซ์

ชิป DDR4 ล่าสุดสำหรับคอมพิวเตอร์ตั้งโต๊ะความถี่สูงสุด มากกว่า 3,000 เมกะเฮิทซ์ และมีแนวโน้มเพิ่มขึ้นตามเทคโนโลยีการผลิตที่พัฒนาอย่างต่อเนื่อง ผู้อ่านสามารถค้นคว้าเพิ่มเติมได้ที่ [wikipedia](#)

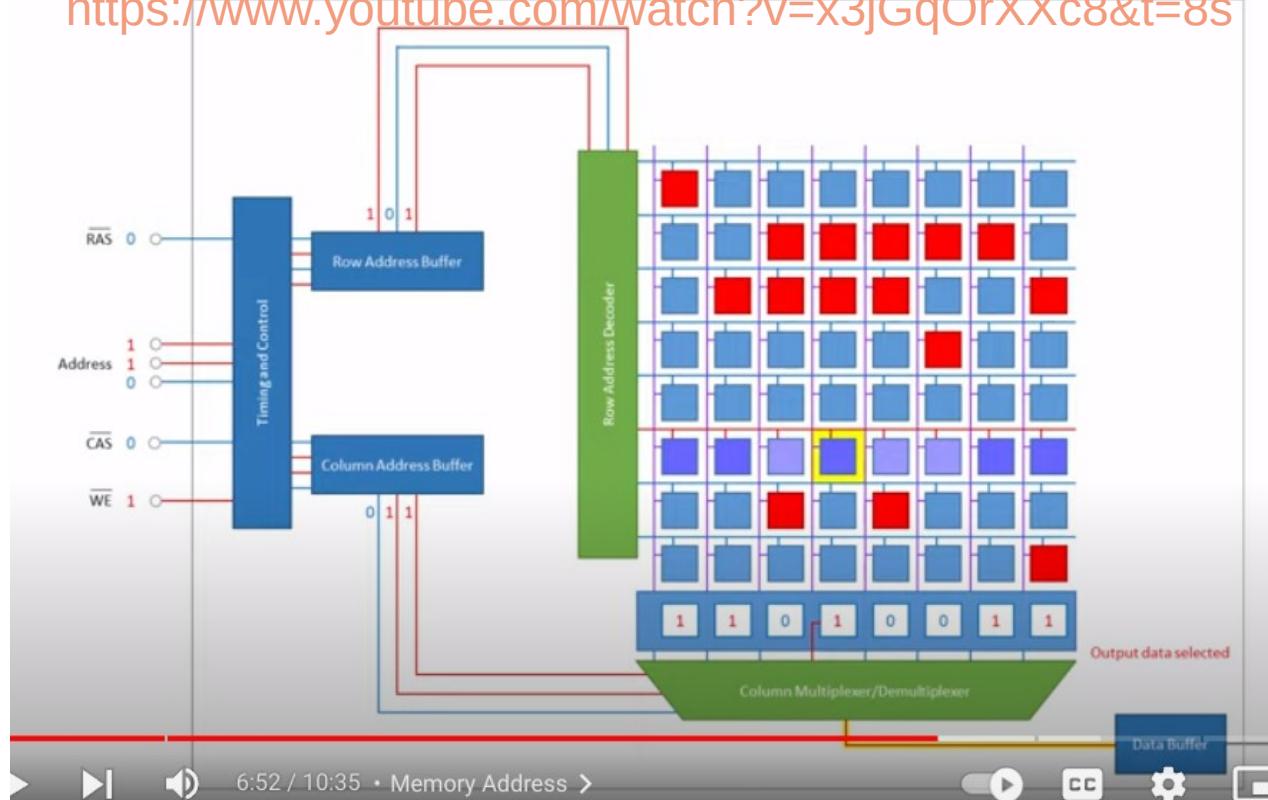
5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม (Synchronous Dynamic RAM: SDRAM)

- Command/Address CA[0:9] ขนาด 10 บิต ใช้มัลติเพล็กซ์ (Multiplex) สัญญาณคำสั่ง (Command) และแอดเดรส (Address) เพื่อรับคำสั่ง (Command) และสัญญาณแอดเดรส (Address) ต่างๆ ห่วงเวลาเดียวกัน
 - ชีพียุจะส่งคำสั่ง (Command) ต่างๆ ดังนี้ Activate, Burst Read, Burst Write, Refresh, Power Down, Precharge และ Burst Terminate เป็นต้น เพื่อกำหนดให้การทำงานของหน่วยความจำ DRAM ได้แก่ Power Up, Deep Power Down, Active, Idle, Reading, Writing, Precharging, Refreshing เป็นต้น
 - สัญญาณแอดเดรสแถว (Row Address) จำนวน 14 บิต และแอดเดรสคอลัมน์ (Column Address) จำนวน 11 บิต จะพากเก็บในวงจรบัฟเฟอร์ เพื่อป้อนให้กับวงจรดอร์หัส (Decoder) คล้ายกับการทำงานของหน่วยความจำ SRAM ในหัวข้อที่ [5.13 การรับสัญญาณแอดเดรสเกิดขึ้น ณ ขอบขาขึ้นและขอบขาลงของแต่ละสัญญาณคล็อก](#)

5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม

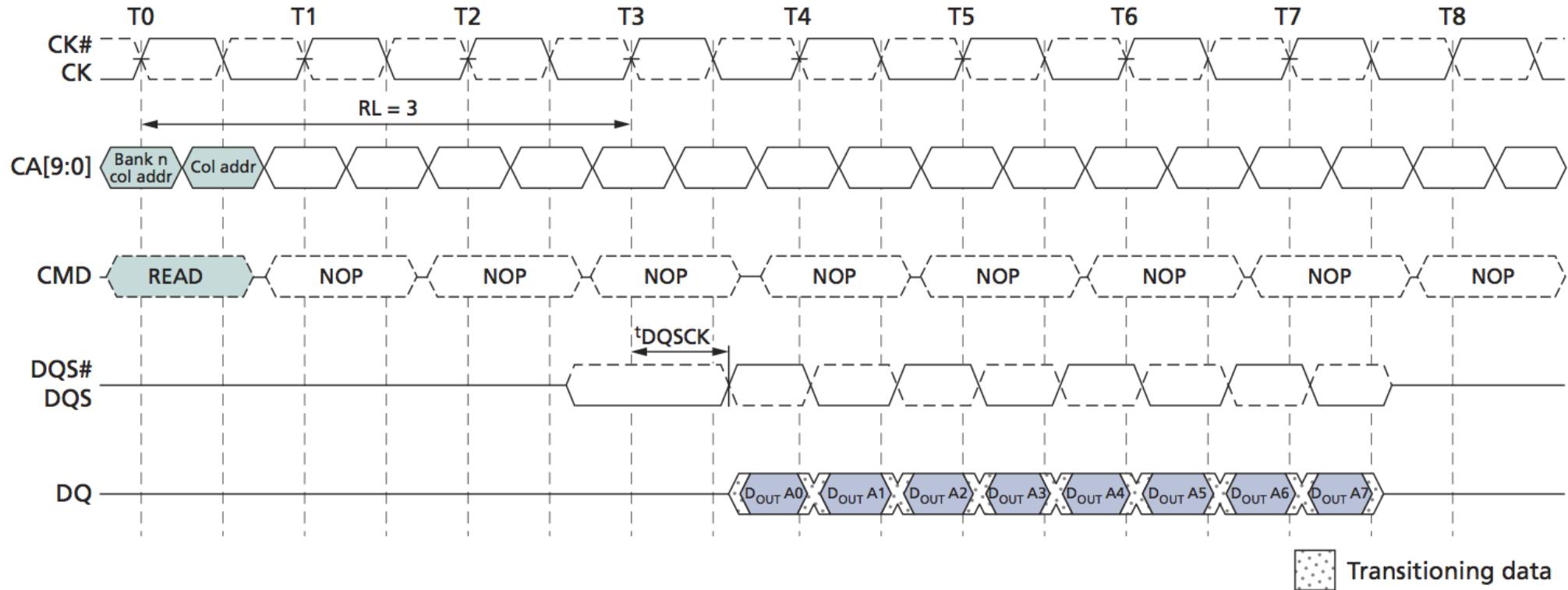
(Synchronous Dynamic RAM: SDRAM): Read

<https://www.youtube.com/watch?v=x3jGqOrXXc8&t=8s>



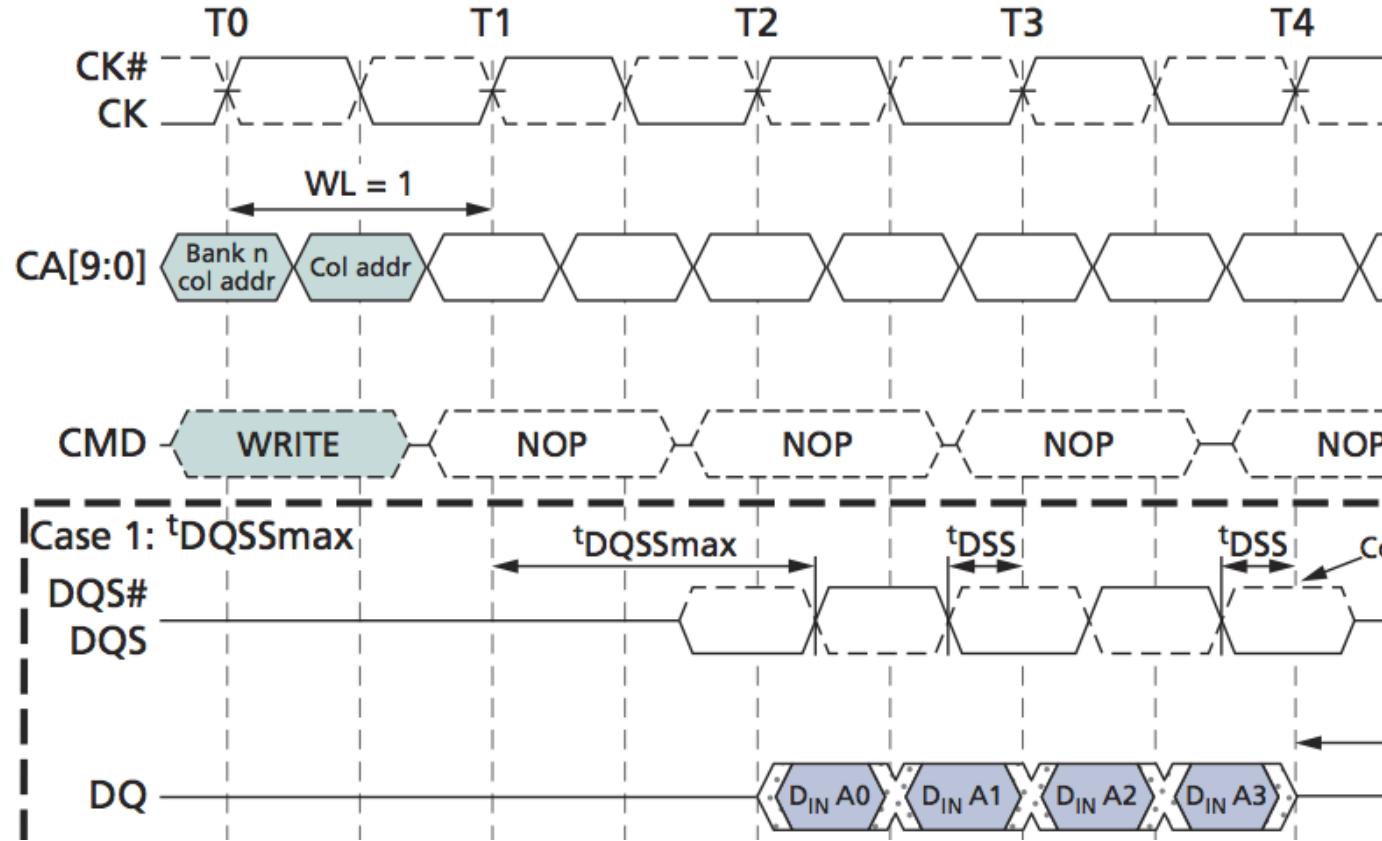
5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม

(Synchronous Dynamic RAM: SDRAM): Read



5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม

(Synchronous Dynamic RAM: SDRAM): Write



5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม^(Synchronous Dynamic RAM: SDRAM)

- ผู้ผลิตเครื่องคอมพิวเตอร์โน้ตบุ๊ค โทรศัพท์เคลื่อนที่สมาร์ทโฟน และอุปกรณ์พกพาต่างนิยมออกแบบติดตั้งชิพหน่วยความจำ DRAM บนเมนบอร์ด (Main Board) เช่นเดียวกับบอร์ด Pi3 เพื่อลดขนาดและปริมาตรของเครื่องให้มีขนาดเท่ากับบอร์ดเครดิต
- ข่าวการผลิต DRAM ยังไม่สามารถรวมกับข่าวการผลิตไมโครโปรเซสเซอร์ได้ ผู้ผลิตจึงจำเป็นต้องผลิตชิพ DRAM แยกต่างหาก
- ความถี่ของคลีอก ความจุ (กิบีไบท์) ต่อชิพเพิ่มสูงขึ้นเรื่อยๆ การระบายน้ำร้อนออกจากชิพ DRAM จึงมีความยากและท้าทายเพิ่มขึ้น

5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม (Synchronous Dynamic RAM: SDRAM): Refresh

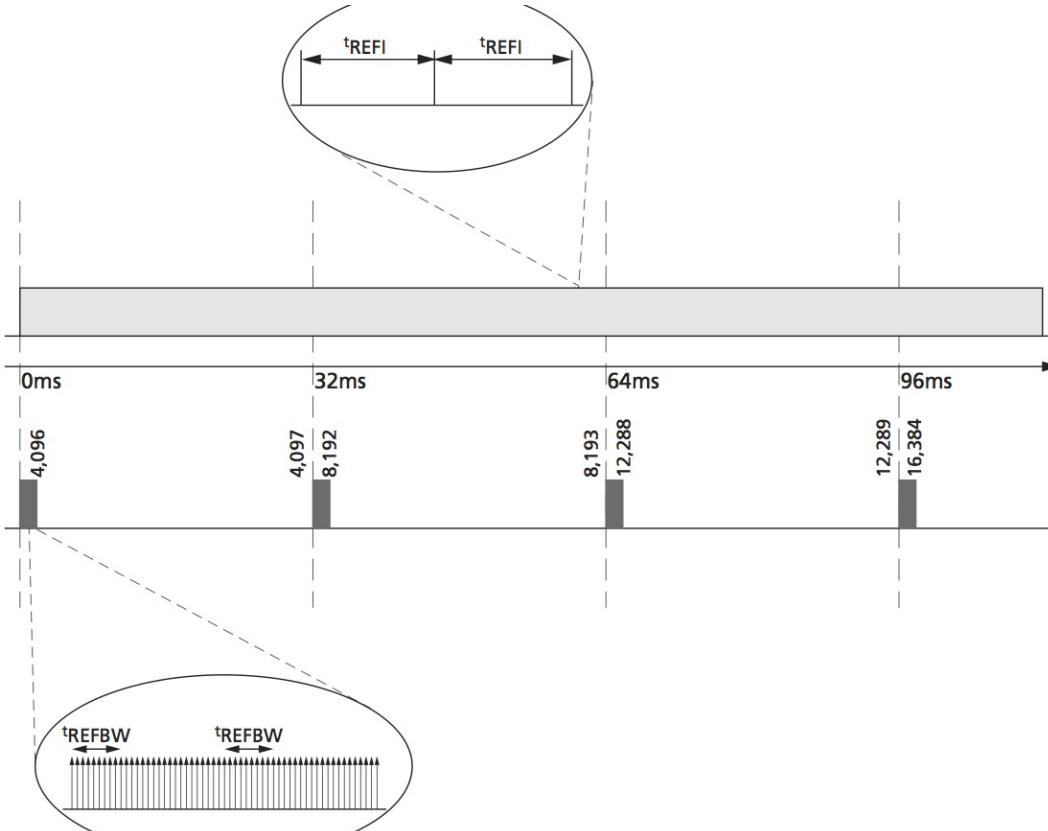
การรีเฟรช (Refresh) คือ การอ่านข้อมูลที่อยู่ในบิทเซลล์ต่างๆ แล้วเขียนซ้ำที่บิทเซลล์เดิม เพื่อป้องกันไม่ให้ประจุที่เก็บอยู่ในบิทเซลล์ต่างๆ รั่วไหลหายไป เนื่องจากเซลล์ต่างๆ ที่ใช้เก็บข้อมูล ทำหน้าที่มี ('1') หรือไม่มี ('0') ประจุไฟฟ้า สำหรับบิทเซลล์ที่มีประจุฯ เหล่านี้อาจรั่วไหลหายไปเมื่อเวลาผ่านไปไม่กี่มิลลิวินาที เมื่อจำนวนประจุลดลง ทำให้การแยกแยะระหว่างบิทเซลล์ที่มีและไม่มีประจุยกขึ้น และอาจทำให้ตัวความไม่ถูกต้องและเกิดข้อผิดพลาดในการอ่านข้อมูล

วงจรควบคุมจะส่งคำสั่งรีเฟรชทุกๆ 32 มิลลิวินาที หากมีการรีเฟรชดำเนินการอยู่ การอ่านหรือเขียนหน่วยความจำจะต้องหยุดรอ เพื่อให้ขบวนการรีเฟรชนั้นเสร็จสิ้น ในทำงานองเดียวกัน หากมีการอ่านหรือเขียนข้อมูลจริงอยู่ การรีเฟรชจะต้องหยุดรอ ก่อน เพื่อให้ขบวนการอ่านหรือเขียนนั้นเสร็จสิ้น

หน่วยความจำ静态แรมไม่ต้องมีการรีเฟรชข้อมูล เนื่องจากการจัดเก็บข้อมูลใช้วิธีการเก็บข้อมูลที่แตกต่างกับหน่วยความจำ DRAM ทำให้ SRAM มีสมรรถนะและประสิทธิภาพสูงกว่า แต่ต้องใช้จำนวนทรานซิสเตอร์ต่อบิทเซลล์มากกว่า จึงทำให้ใช้พื้นที่บนแผ่นซิลิคอนต่อความจุข้อมูล 1 บิทใหญ่กว่าเซ็นเซอร์ ผู้อ่านสามารถค้นคว้าเพิ่มเติมได้ที่ [wikipedia](#)

5.5 หน่วยความจำหลักชนิดซิงโครนัสไดนามิกแรม

(Synchronous Dynamic RAM: SDRAM): Refresh



สรุปท้ายบท

หน่วยความจำลำดับชั้นอาศัยเทคโนโลยีหน่วยความจำหลายชนิด หลายขนาดความจุเข้าด้วยกัน ยกตัวอย่าง เช่น

- เทคโนโลยี SRAM เป็นหน่วยความจำขนาดเล็กแต่เวลาเข้าถึงสั้น นำมาใช้งานเป็นรีจิสเตอร์ และ แคช ลำดับต่างๆ
- เทคโนโลยี SDRAM เป็นหน่วยความจำความจุมากกว่าแต่เวลาเข้าถึงนานกว่า SRAM นำมาใช้งานเป็น หน่วยความจำหลัก
- เทคโนโลยีหน่วยความจำแฟลช เป็นหน่วยความจำความจุมากกว่าแต่เวลาเข้าถึงนานกว่า SDRAM นำมา ใช้งานเป็นอุปกรณ์เก็บรักษาข้อมูล รายละเอียดเพิ่มเติมในบทที่ 7

เพื่อให้คอมพิวเตอร์มีความจุเพียงพอและตอบสนองต่อความต้องการใช้งานระบบโดยเฉลี่ยได้รวดเร็วขึ้น โดย การผ่านจุดเด่นของหน่วยความจำแต่ละชนิดเข้าด้วยกัน และช่วยประหยัดต้นทุนของระบบ

References

- https://www.researchgate.net/figure/Block-Diagram-of-Micro-SD-card_fig6_306236972
- <https://gabrieletolomei.wordpress.com/miscellanea/operating-systems/in-memory-layout/>
- <https://freedompenguin.com/articles/how-to/learning-the-linux-file-system>
- <https://www.techpowerup.com/174709/arm-launches-cortex-a50-series-the-worlds-most-energy-efficient-64-bit-processors>
- https://www.researchgate.net/figure/NVIDIA-Tegra-2-mobile-processor-11_fig1_221634532
- Harris, D. and S. Harris (2013). Digital Design and Computer Architecture (1st ed.). USA: Morgan Kauffman Publishing.
- <https://learn.adafruit.com/resizing-raspberry-pi-boot-partition/edit-partitions>

References

- https://en.wikipedia.org/wiki/Human-computer_interaction
- <https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/programmer-s-guide-for-armv8-a>
- https://xdevs.com/article/rpi3_oc/
- https://www.gsmarena.com/a_look_inside_the_new_proprietary_apple_a6_chipset-news-4859.php
- https://www.slideshare.net/kleinerperkins/2012-kpcb-internet-trends-yarend-update/25-Global_Smartphone_Tablet_Shipments_Exceeded
- <https://www.aliexpress.com/item/32329091078.html>
- <https://www.raspberrypi.org/forums/viewtopic.php?t=63750>
- <https://www.youtube.com/watch?v=2ciyXehUK-U>