

องค์ประกอบคอมพิวเตอร์และภาษาแอสเซมบลี: กรณีศึกษา Raspberry Pi

รศ.ดร.สุรินทร์ กิตติธรกุล

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

สารบัญ

- 5.1 โครงสร้างของลำดับชั้นหน่วยความจำของบอร์ด 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	0.20s	nano Lab7_7.s
t6301003	pts/20	100.127.251.0	15:41	61:05	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

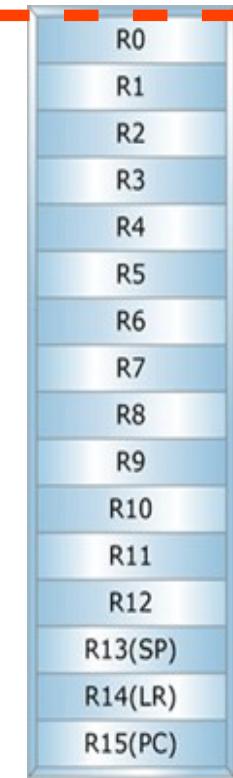
pi@Pi432b: ~

```
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

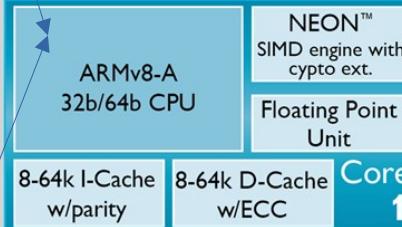
F1Help F2Setup F3Search F4Filter F5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit

การเชื่อมโยงระหว่างรีจิสเตอร์ แคช หน่วยความจำสมீனและอุปกรณ์เก็บรักษาข้อมูล



Cortex®-A53

ARM CoreSight™ Multicore Debug and Trace



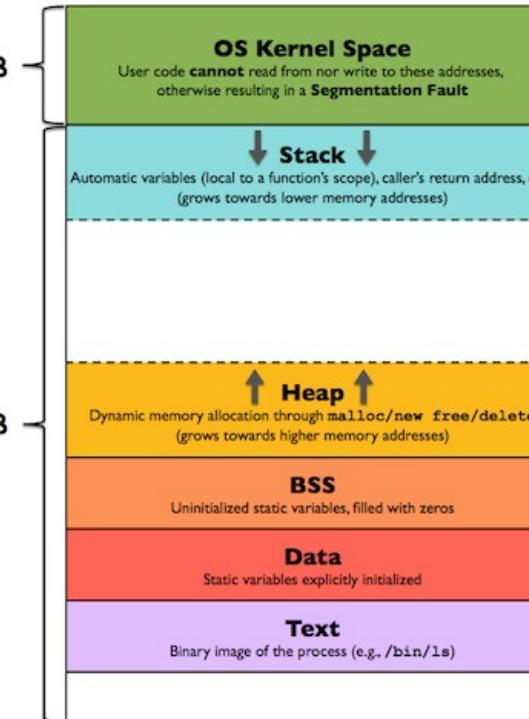
รีจิสเตอร์

แคชลำดับที่ 1 และ 2

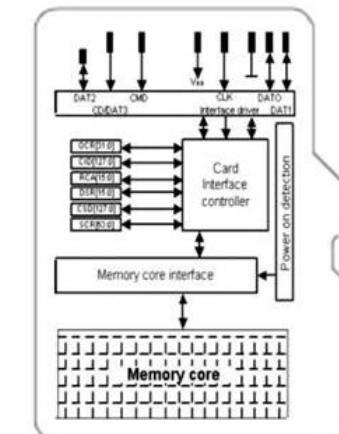
BCM2837/2711

1 GB

3 GB

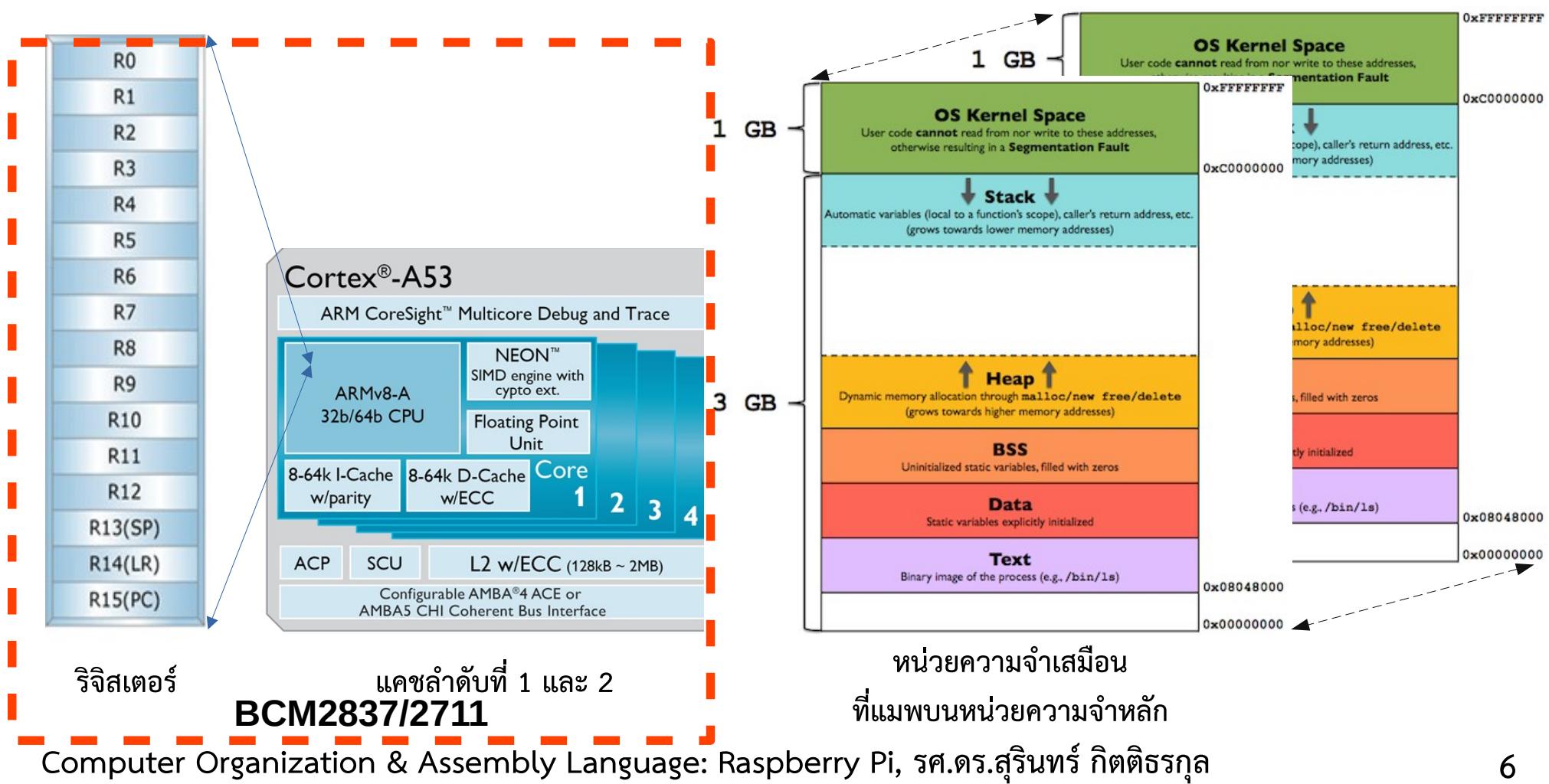


หน่วยความจำสมீன
ที่แมปบนหน่วยความจำหลัก



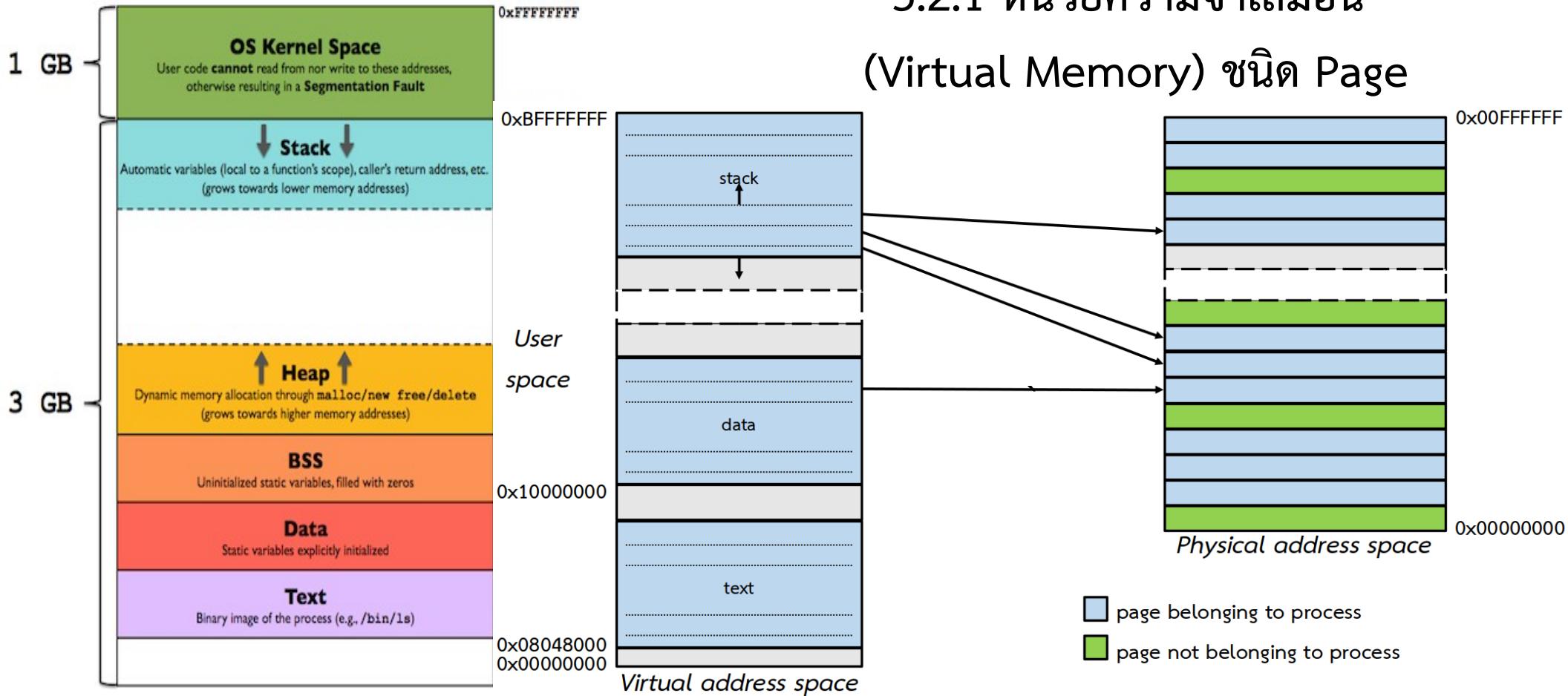
อุปกรณ์เก็บรักษาข้อมูล

การเชื่อมโยงระหว่างรีจิสเตอร์ แคช หน่วยความจำเสมือนของ.library โปรเซส

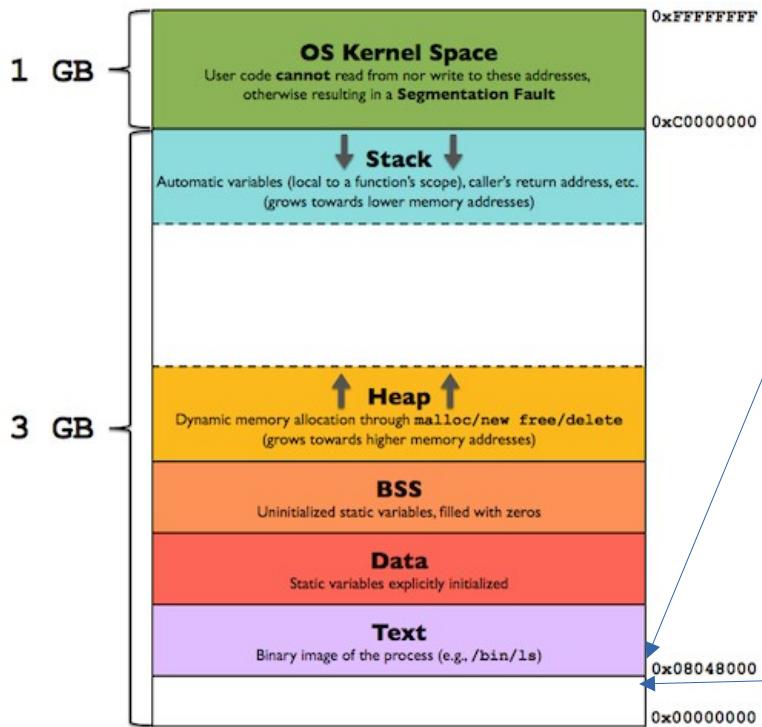


5.2.1 หน่วยความจำเสมือน

(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

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

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

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

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

- 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

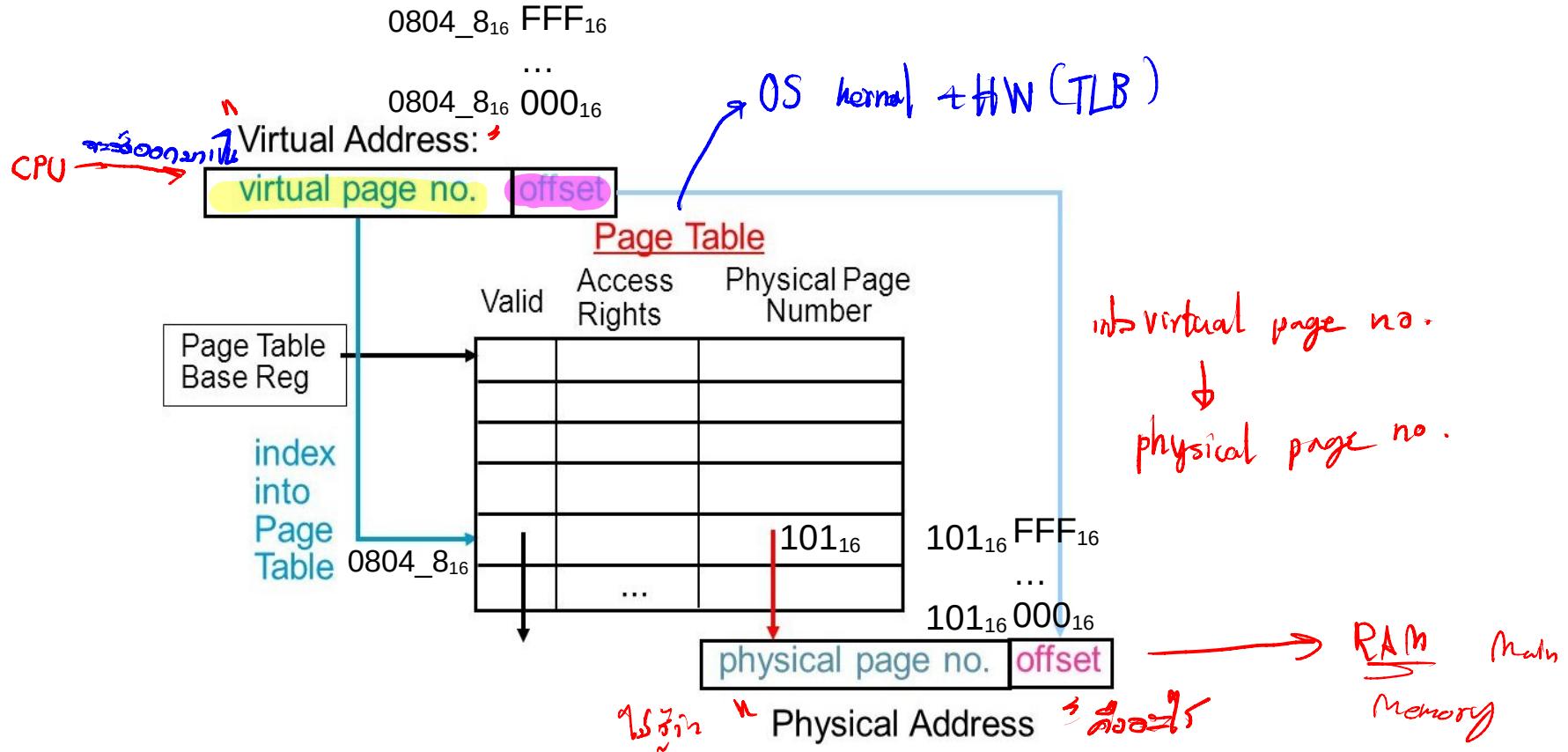
Virtual Page Num = 0x0804_8

- 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

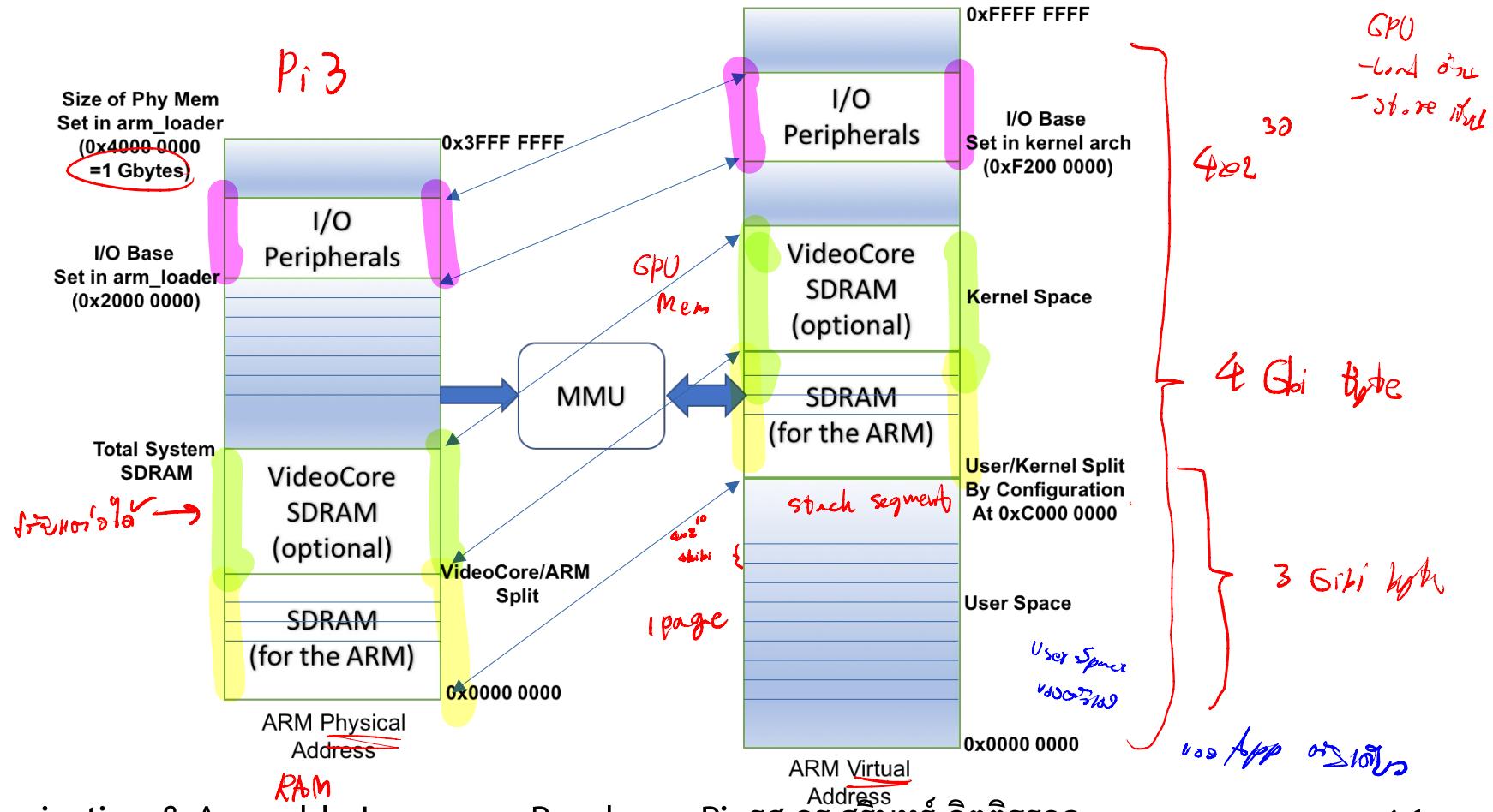
Phy Page Num = 0x101

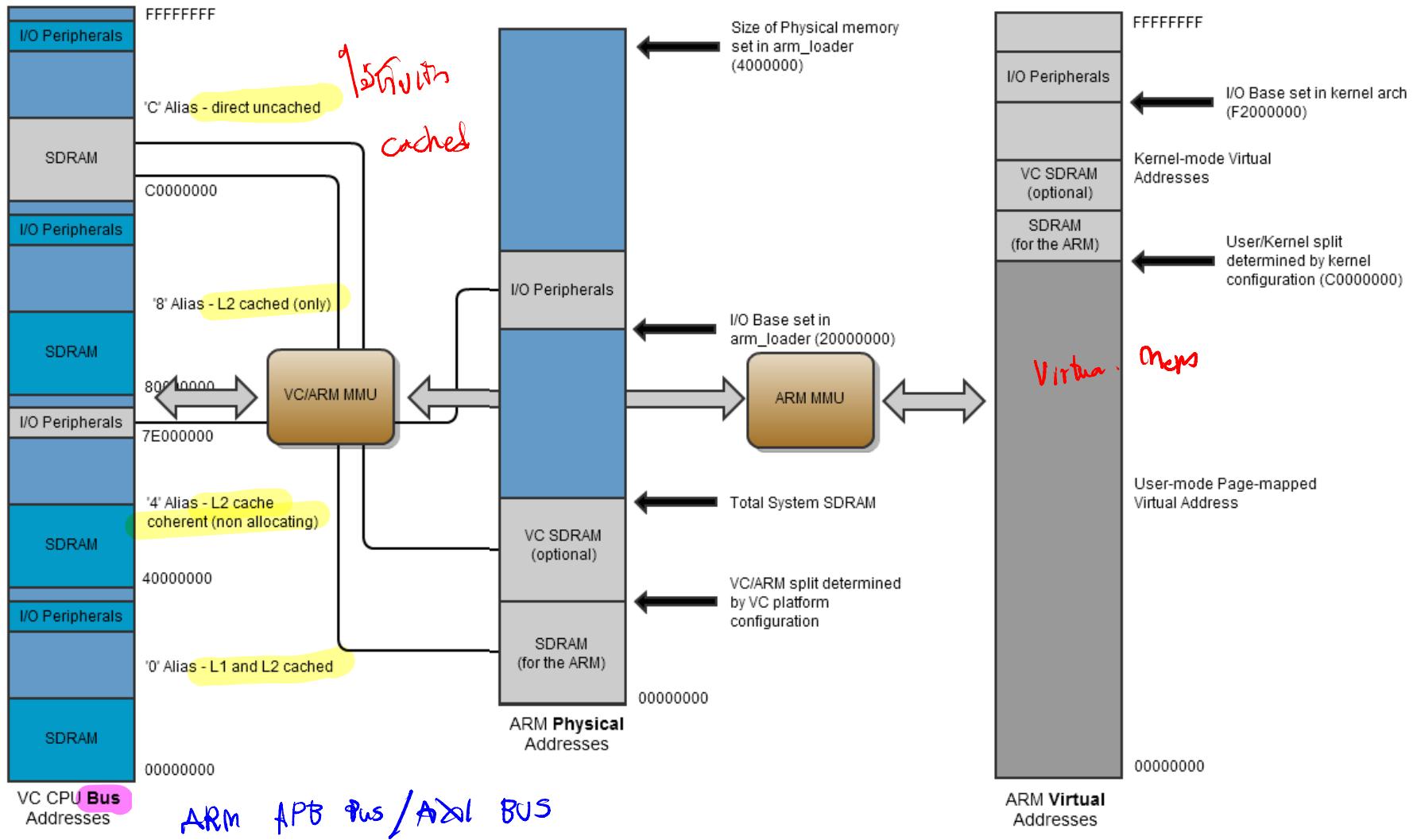
- 0x10_1000 – 0x10_1003 คำสั่งที่ 00
- 0x10_1004 – 0x10_1007 คำสั่งที่ 01
- 0x10_1008 – 0x10_100B คำสั่งที่ 02
- ...
- 0x10_10FC – 0x10_10FF คำสั่งที่ 63
- 0x10_1100 – 0x10_1103 คำสั่งที่ 64
- 0x10_1104 – 0x10_1107 คำสั่งที่ 65
- 0x10_1108 – 0x10_110B คำสั่งที่ 66
- ...
- 0x10_11FC – 0x10_11FF คำสั่งที่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

5.2.1 หน่วยความจำเสมือน (Virtual Memory) ชนิด Page: Page Table



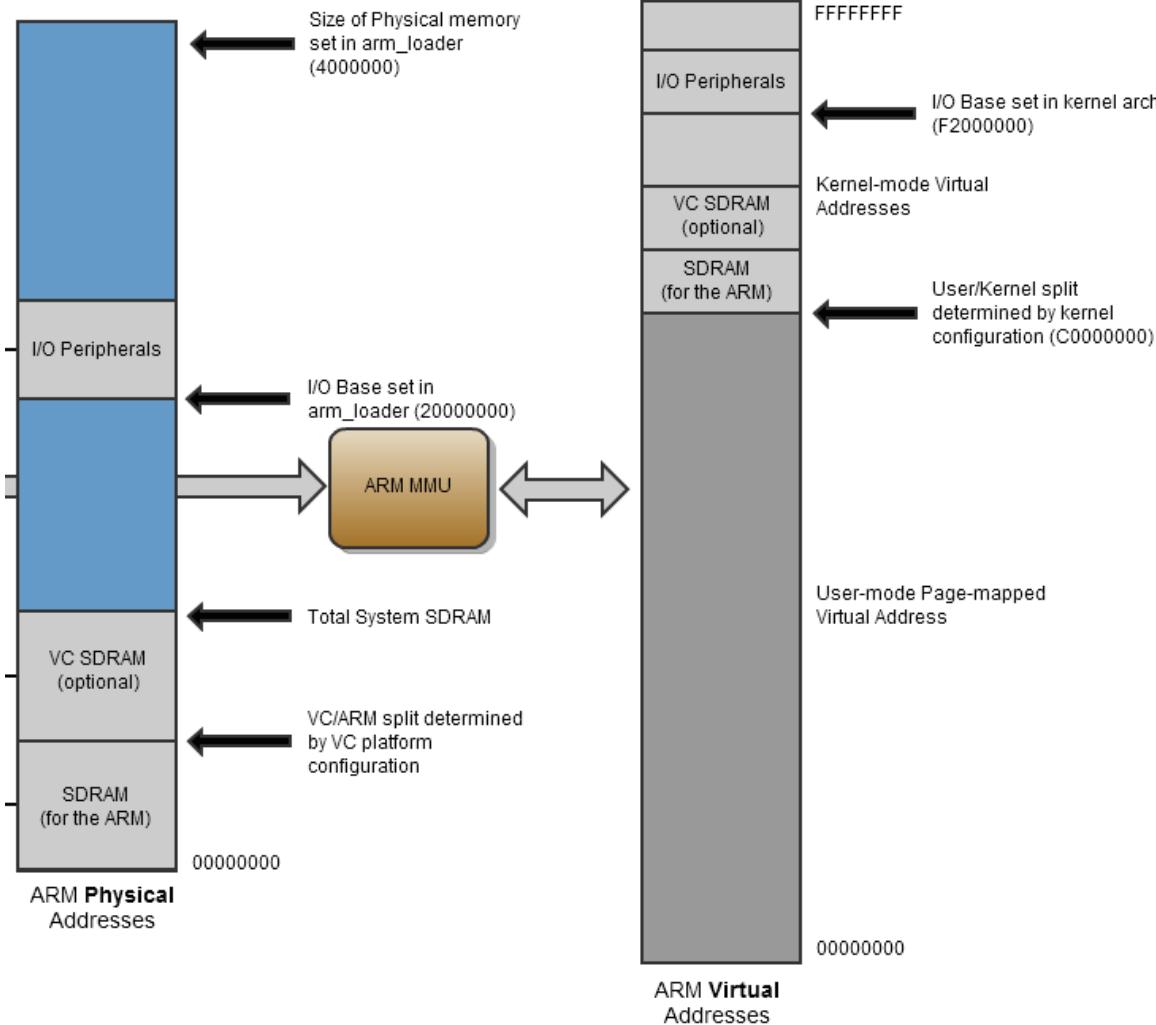
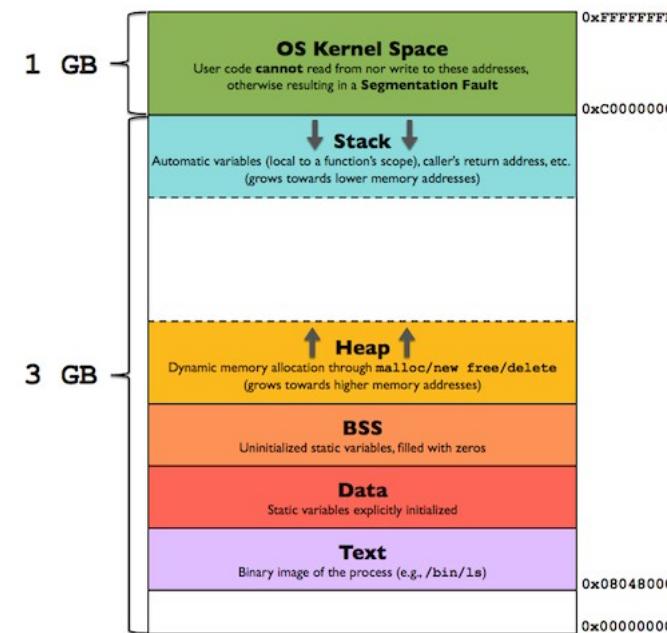
5.2.2 หน่วยความจำเสมือน (Virtual Memory) ของบอร์ด 32 bit Pi OS





5.2.2 หน่วยความจำเสมือน (Virtual Memory)

ของบอร์ด Pi3



5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)

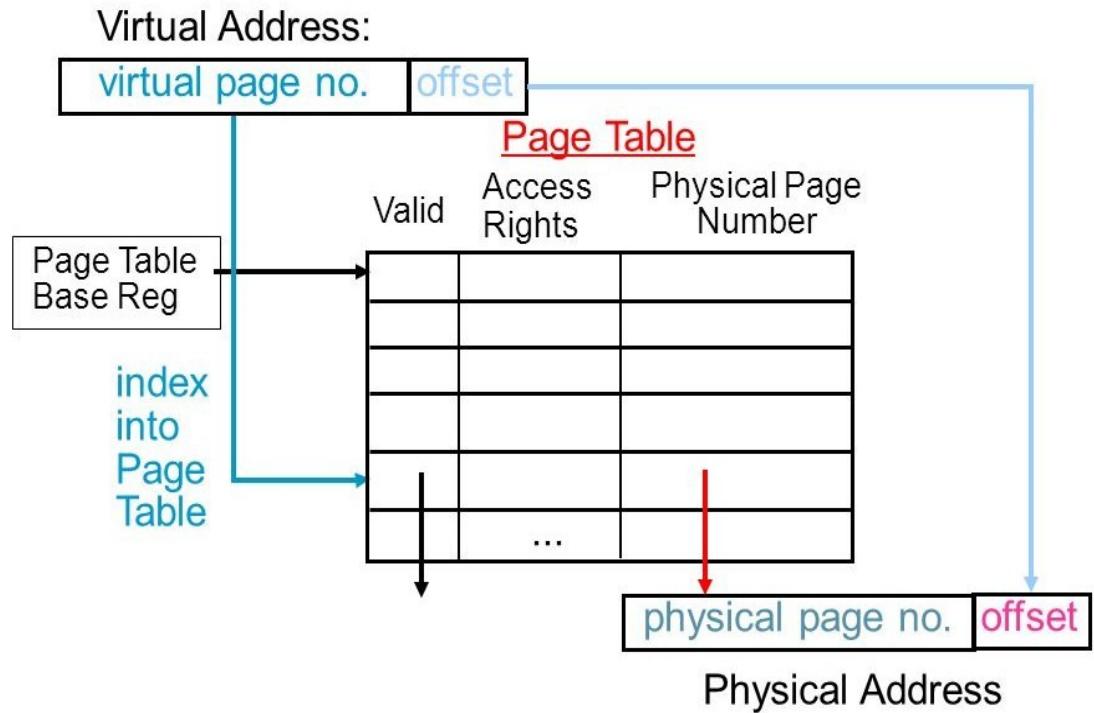
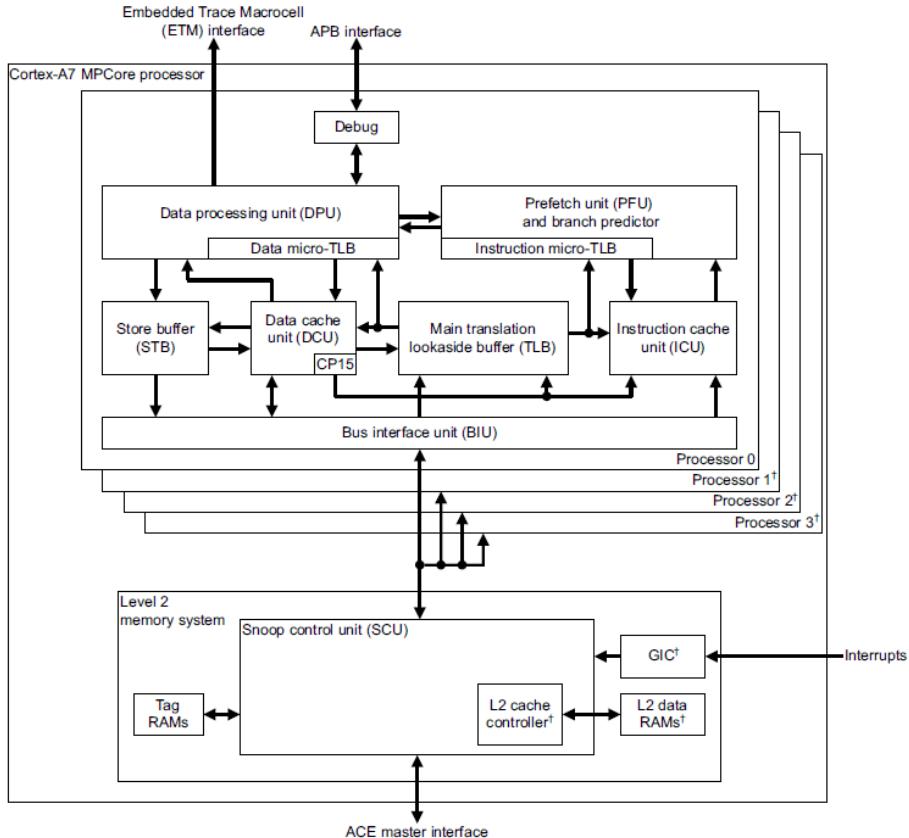
cacheอยู่ใน chip เท่ากับ CPU

(อยู่ใน chip)

Static RAM & Dynamic RAM

- แคช่มีความจุน้อยแต่ทำงานเร็ว เพราะใช้เทคโนโลยี SRAM และสามารถบรรจุอยู่ในแผ่นซิลิกอนเดียวกับซีพียูคอร์
- แคช่มีหลายชั้น ชั้นบนสุดของแต่ละคอร์ เรียกว่า ชั้น L1 ทำงานที่แยกจากกัน คือ
 - แคชชั้น L1 Instruction เป็นพื้นที่ของ Text Segment เก็บคำสั่ง วย
 - แคชชั้น L1 Data Cache เป็นพื้นที่อื่นๆ เก็บข้อมูลที่ไม่ใช่ Text Segment วย
- แคชชั้น L2 จะมีความจุมากขึ้นและเก็บคำสั่งและข้อมูลจากทุกๆ เช็คเม้นท์ และคอร์ต่างๆ ใช้งานร่วมกัน
- บางพื้นที่ของหน่วยความจำเสมือนใช้งานผ่านแคช บางพื้นที่จะไม่ใช้งานผ่านแคชแม้แต่ชั้นเดียว
- แคชใช้ Phy Address ในการทำงาน เพราะจะไม่ซ้ำซ้อนกับ Virtual Address

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ARM Cortex A



5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ARM Cortex A

- virtual page no. *Fault* physical page no.
1. นำเลขเพจเมื่อใน PC (Program Counter) ไปสืบค้นหาค่าเลขเพจภายใน Instruction micro-

TLB

- หากเจอ เรียกว่า TLB ฮิท (Hit) จะใช้เวลาส่งค่าหมายเลขเพจภายใน instruction ไปใช้งานสั้นๆ (0.5-1 นาโนวินาที)
fast (less than 1ns)
- หากไม่เจอ เรียกว่า TLB มิส (Miss) วงจรจะนำหมายเลขเพจเมื่อใน Main TLB ต่อไปซึ่งจะใช้เวลาเพิ่มขึ้นอีก
 - หากฮิท ส่งค่าหมายเลขเพจภายใน instruction ไปใช้งาน (2-4 นาโนวินาที)
 - หากมิสอีกรอบ จะต้องสืบค้นใน **ตารางเพจ** เป็นลำดับสุดท้ายซึ่งจะใช้เวลานานขึ้น *long storage*
copy \rightarrow DRAM
long time ELF

400 ms
550 ms

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ARM Cortex A

L1 - I - cache

2. นำเลขเพจภายในเข้ามายกับค่าอффเซทเพื่อนำไปสืบค้นคำสั่งใน L1 Cache ซึ่งเทียบเท่าแคชคำสั่งลำดับที่

1

physical Address \Rightarrow L1 Cache

- หากเจอ เรียกว่า เกิดแคชฮิทที่ลำดับที่ 1 (L1 Cache Hit) และนำคำสั่งส่งกลับไปยังซีพียูต่อไปซึ่งจะใช้เวลาสั้นที่สุด (0.5 - 1 นาโนวินาที)
- หากไม่เจอ เรียกว่า เกิดแคชมิสที่ลำดับที่ 1 (L1 Cache Miss) นำแอดเดรสภายในมาสู่แคชลำดับที่ 2
 - หากฮิท ที่ลำดับที่ 2 (L2 Cache Hit) และนำคำสั่งส่งกลับไปยังซีพียูและแคชลำดับที่ 1 ซึ่งจะใช้เวลานานขึ้น (2 - 4 นาโนวินาที)
 - หากมิส ที่ลำดับที่ 2 (L2 Cache Miss) จะจจะนำแอดเดรสภายในไปค้นหาคำสั่งในหน่วยความจำหลัก (DRAM) ผ่านทางวงจรเข้ามต์อกกับหน่วยความจำ DRAM ซีพียูจะต้องใช้เวลารอ (เกินสิบนาโนวินาที) เพื่อที่คำสั่งจะเดินทางมาจากหน่วยความจำภายในไปคำสั่งที่ได้จากแคชลำดับต่างๆ และจาก SDRAM จะเป็นเลขฐานสองจำนวน 4-8 คำสั่ง คิดเป็นความยาว 128-256 บิต ขึ้นอยู่กับความกว้างของแคชแต่ละบล็อก ซึ่งจะอธิบายในหัวข้อถัดไป

Access หน่วย

test segment

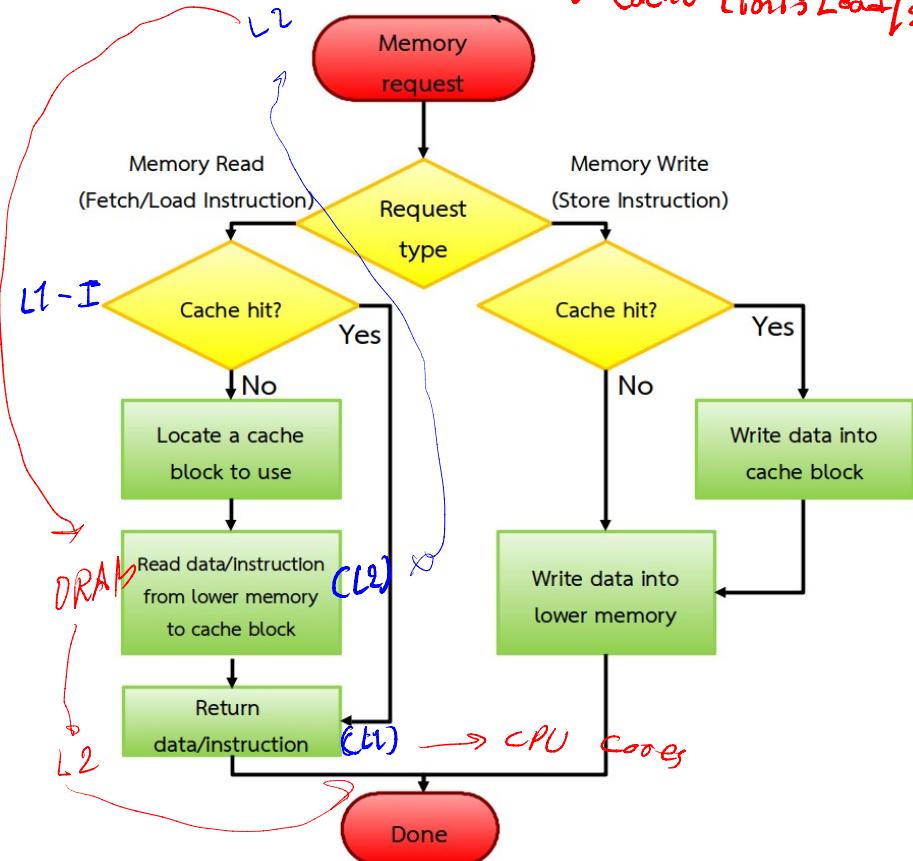
⇒ A30

Data

segment อย่าง

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)

D-Cache (with Load/Store tag)



- 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 ($a[j]$, $i++$)

Phy Address

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache)

Phy Page No. | Offset

1 page 1000

40 96 byte

12
--
1

ที่ 1000 000 000 offset

ที่ 1000 3 000

PC	Offset	เลขบล.	คำสั่ง	คอมเม้นท์
	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		...	

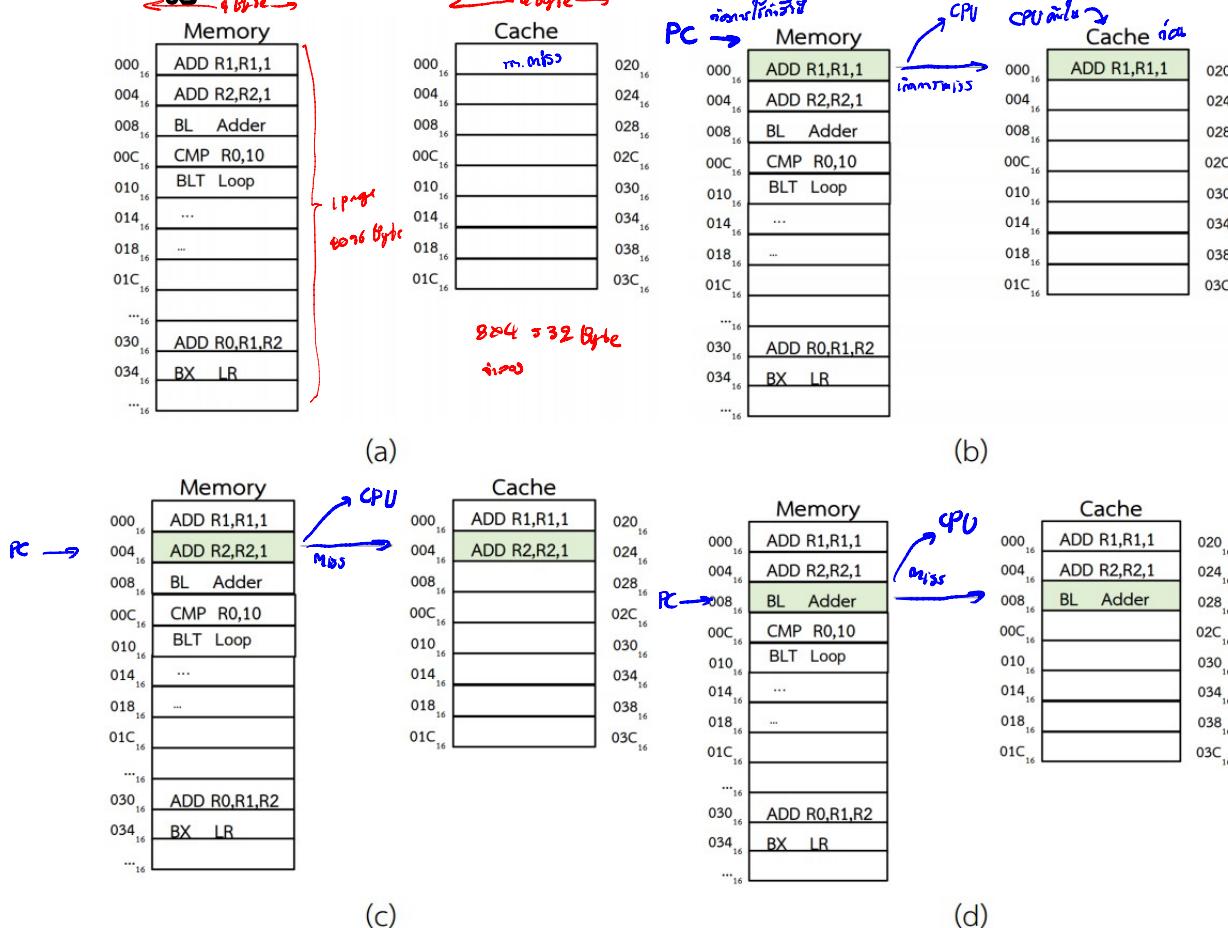
(Direct Map)

DM (Direct Map)
Associativity
Cache 8x

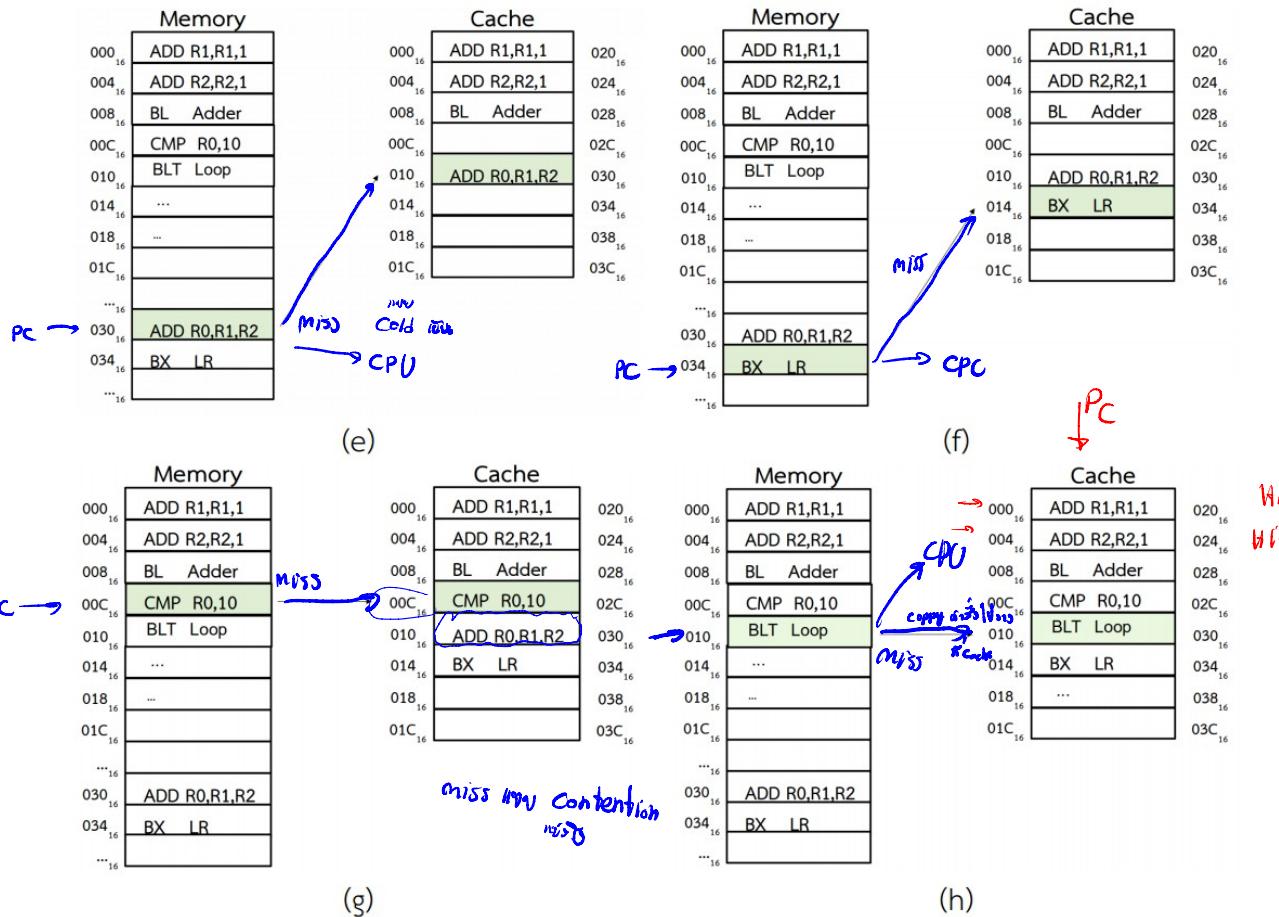
Set Associative

TLB → Fully
Associative

5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ชนิด Direct Map



5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ชนิด Direct Map

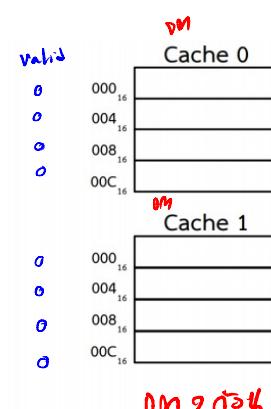


5.3 หลักการพื้นฐานของหน่วยความจำแคช (Cache): ชนิด Direct Map

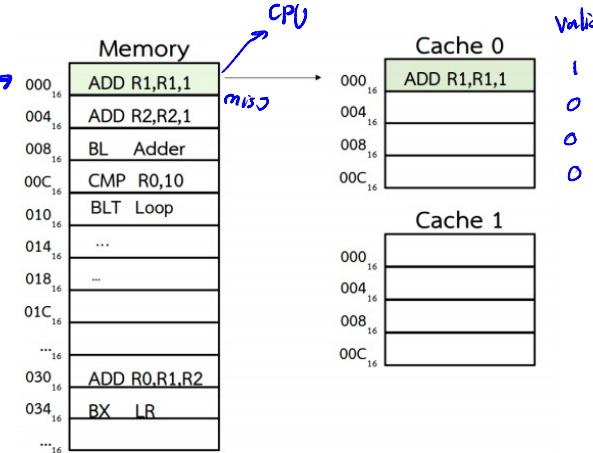
ผู้อ่านจะสังเกตเห็นว่า คำสั่งจากหน่วยความจำภายในอยู่ 2 ตำแหน่งถูกแมปให้ใช้แคชบล็อกเดียวกัน เช่น แคชตำแหน่งที่ 000_{16} จะเป็นเป้าหมายของหน่วยความจำตำแหน่งที่ 000_{16} และ 020_{16} เป็นต้น ยิ่งไปกว่านั้น แคชลำดับที่ 1 ที่ใช้งานในซีพียูต่างๆ มีความจุน้อยกว่าหน่วยความจำภายในมาก เพราะหน่วยความจำภายในของบอร์ด 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

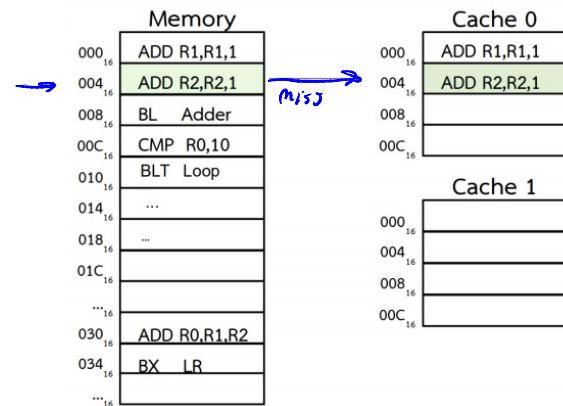
Memory	
000 ₁₆	ADD R1,R1,1
004 ₁₆	ADD R2,R2,1
008 ₁₆	BL Adder
00C ₁₆	CMP R0,10
010 ₁₆	BLT Loop
014 ₁₆	...
018 ₁₆	...
01C ₁₆	
...	
030 ₁₆	ADD R0,R1,R2
034 ₁₆	BX LR
...	



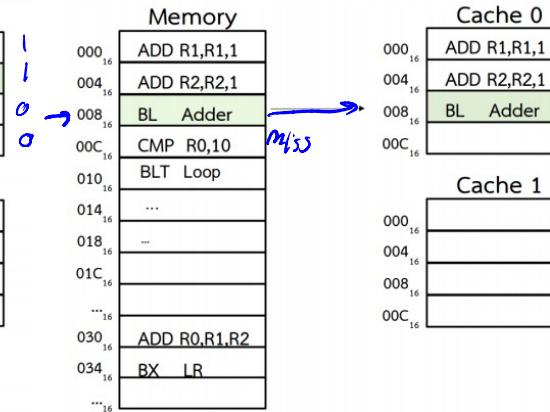
(a)



(b)

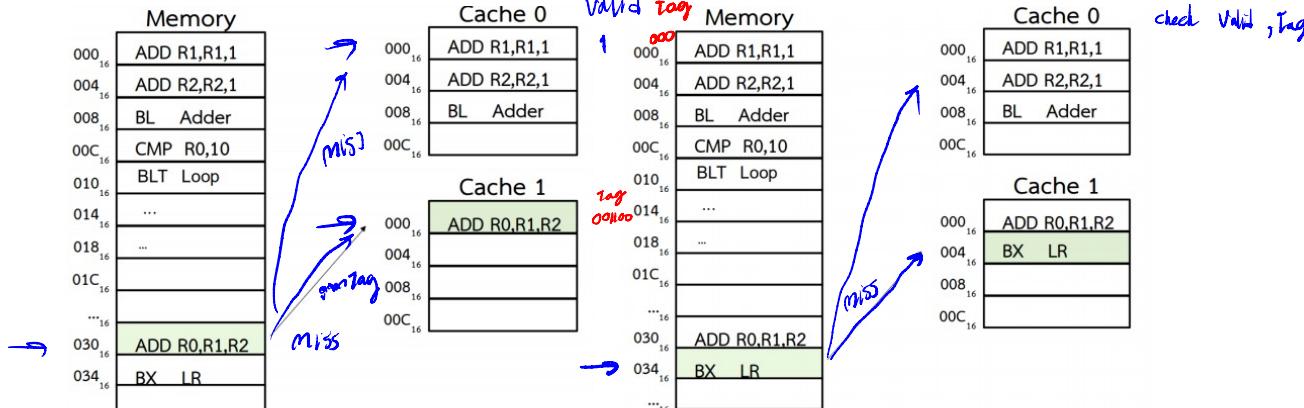


(c)



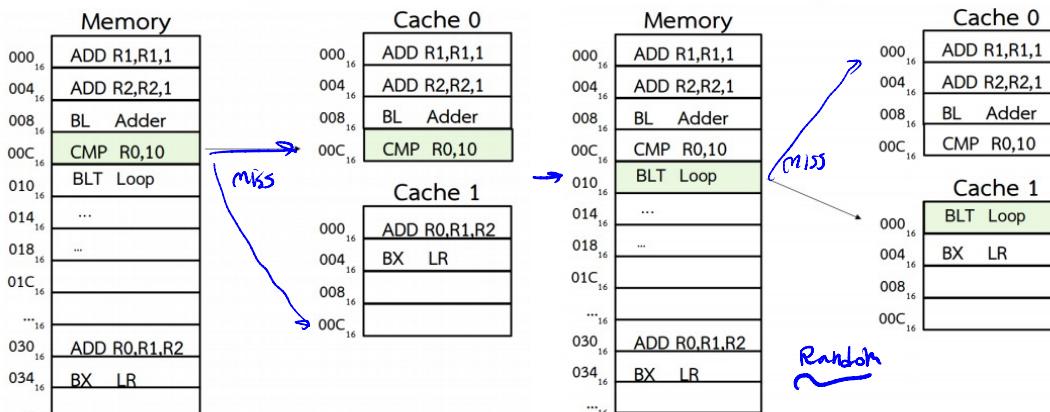
(d)

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



(e)

(f)



(g)

(h)

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

ตัวอย่าง

ผู้อ่านจะสังเกตเห็นว่า แคชชนิด SA มีการทำงานคล้ายกับแคชชนิด DM แต่มีจำนวน 2 เวյ์ นั่นคือ แคชหมายเลขล็อกเดียวกันมี 2 ทางเลือก คือ Cache 0 และ Cache 1 ขึ้นอยู่กับว่าเรียกว่า หากไม่ว่าught ทั้งคู่ 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