องค์ประกอบของเครื่องคอมพิวเตอร์ และภาษาแอสเซมบลี: ARM และ RaspberryPi3

บทที่ 7 อุปกรณ์เก็บรักษาข้อมูลและระบบไฟล์

รศ.ดร.สุรินทร์ กิตติธรกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

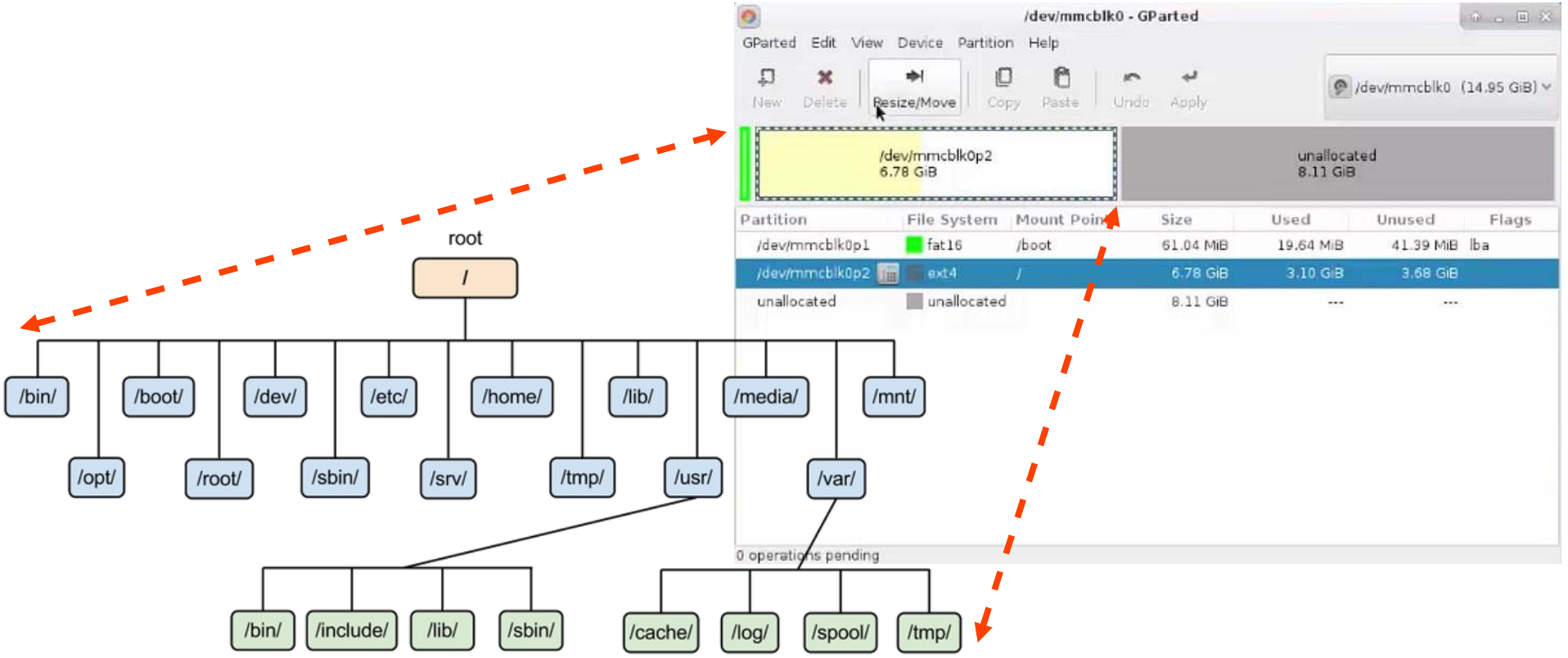
สารบัญ

7.1 ระบบไฟล์ (File System)

ไฟล์จะเก็บอยู่ในอุปกรณ์เก็บรักษาข้อมูลตามโครงสร้างหรือระบบไฟล์ (File System) ของระบบปฏิบัติการนั้นๆ ผู้ใช้และนักพัฒนาสามารถใช้งานไฟล์โดยการ สร้างไฟล์ (Create) เขียน/บันทึก/อ่านไฟล์ เปลี่ยนชื่อ (Rename) ไฟล์ ลบ (Remove/Delete) ไฟล์ ทำสำเนา (Copy) ย้ายตำแหน่ง (Move) ไฟล์ และกู้คืน (Recover) ไฟล์เมื่อเกิดปัญหา โดยไม่จำเป็นต้องรับรู้ว่าระบบไฟล์เป็นชนิดใด เนื่องจากระบบปฏิบัติการได้ซ่อนรายละเอียดไว้ เพื่อให้ผู้ใช้และนักพัฒนาโปรแกรม สามารถพัฒนาโปรแกรมได้สะดวกมากขึ้น ตำราเล่มนี้จะเน้นที่พื้นฐานของระบบไฟล์ดั้งเดิมของระบบยูนิกซ์ ซึ่งระบบอื่นๆ ได้พัฒนาต่อยอด ระบบบริหารจัดการไฟล์ที่สำคัญและเป็นที่นิยม ได้แก่

- ระบบ NTFS (File System) สำหรับระบบวินโดวส์ รายละเอียดเพิ่มเติมที่ ntfs.com
- ระบบ EXT4 สำหรับระบบลีนุกซ์ รายละเอียดเพิ่มเติมที่ wikipedia
- ระบบ Apple File System (APFS) สำหรับระบบ MAC OS รายละเอียดเพิ่มเติมที่ wikipedia

7.1 ระบบไฟล์ (File System)



7.1 ระบบไฟล์ (File System)

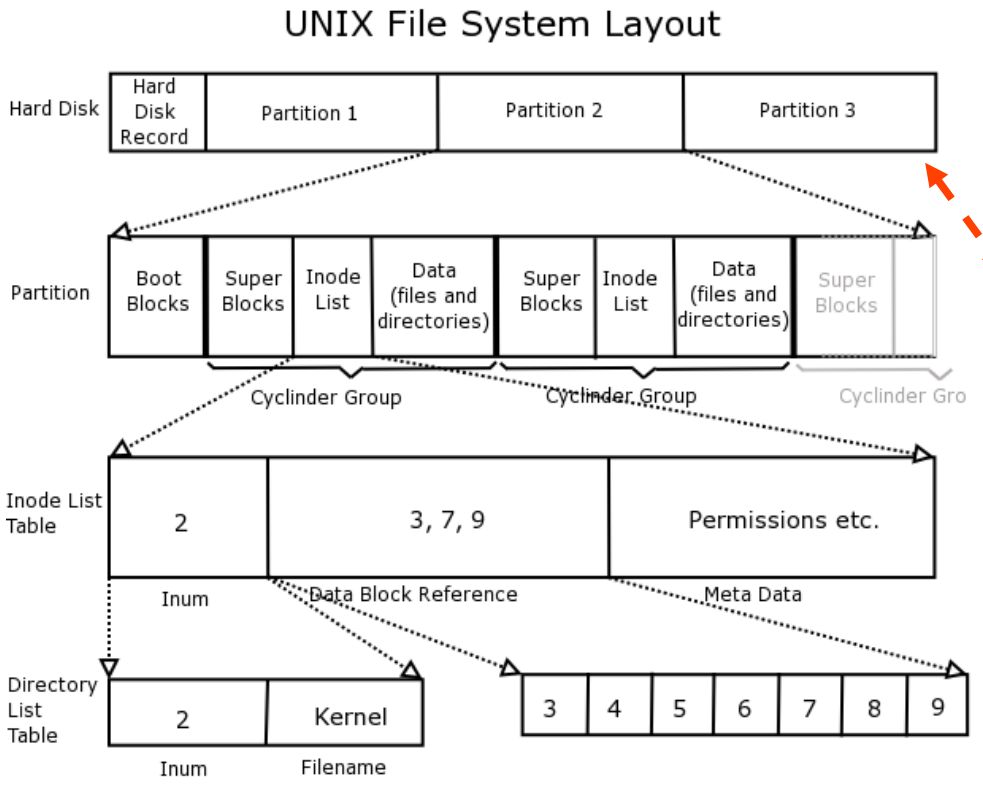
เราจะจินตนาการว่าอุปกรณ์เก็บรักษาข้อมูล เช่น การ์ดหน่วยความจำ SD โซลิดสเตทดิสค์และฮาร์ดดิสค์เหล่านี้ มีลักษณะเป็น**แถบข้อมูล**ที่มีความยาวตามขนาดความจุ และแบ่งแถบออกเป็นพื้นที่ย่อยๆ เรียกว่า **พาร์ทิชัน** (Partition) ซึ่งอุปกรณ์รักษาข้อมูล 1 ตัวสามารถแบ่งเป็นหลายพาร์ทิชัน (Partition) แต่ละพาร์ทิชันมีระบบบริหารจัดการไฟล์ (File System) ของตนเอง ตามรูปที่ 7.1

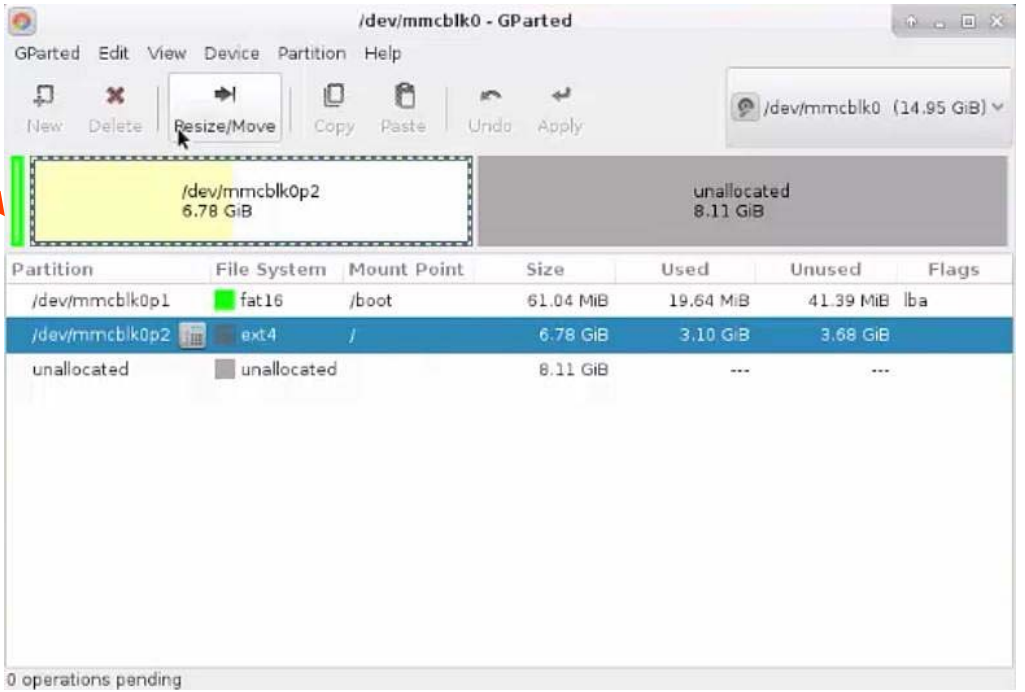
การแบ่งพาร์ทิชัน คือ การแบ่งพื้นที่แถบข้อมูลออกเป็นส่วนต่างๆ ตามที่ระบบไฟล์ออกแบบมา การแบ่งพาร์ทิชันสามารถตอบสนองความต้องการที่หลากหลาย ได้แก่ การบูตระบบปฏิบัติการได้หลากหลาย การจัดเก็บข้อมูลในบางพาร์ทิชัน เป็นต้น ผู้ใช้สามารถติดตั้งระบบปฏิบัติการในแต่ละพาร์ทิชันได้โดยอิสระจากกัน ที่มา archlinux และ datadoctor.biz โดยตำแหน่งเริ่มต้นจะเป็นพื้นที่สำหรับเก็บ **ตารางพาร์ทิชัน** (Partition Table) ประกอบด้วยรายชื่อและข้อมูลประกอบของแต่ละพาร์ทิชัน ตารางพาร์ทิชันที่สำคัญในทางปฏิบัติได้แก่

7.1 ระบบไฟล์ (File System)

- **มาสเตอร์บูตเรคอร์ด (Master Boot Record: MBR)** ทำหน้าที่เก็บรายชื่อพาร์ทิชัน ตำแหน่งเริ่มต้นหรือหมายเลข Logical Block Address (LBA) ในฮาร์ดดิสค์ไว้ในตารางพาร์ทิชัน (Partition Table) พื้นที่ส่วนหนึ่งของมาสเตอร์บูตเรคคอร์ดทำหน้าที่เก็บตารางพาร์ทิชัน ซึ่งตั้งอยู่ในเซ็คเตอร์แรก หรือเซ็กเตอร์หมายเลข 0 ของฮาร์ดดิสค์ โดยใช้พื้นที่ 64 ไบต์ ซึ่งฮาร์ดดิสค์ 1 ตัวสามารถแบ่งพาร์ทิชันได้มากที่สุดเป็นจำนวน 4 พาร์ทิชัน รายละเอียดเพิ่มเติมที่ wikipedia
- **ตาราง GPT** ในปัจจุบันตารางพาร์ทิชันมีชื่อว่า Globally Unique Identification Partition Table หรือ **GUIDPT** หรือ **GPT** ทำหน้าที่คล้ายกับ MBR แต่รองรับจำนวนพาร์ทิชันในฮาร์ดดิสค์ได้มากกว่า และไบออสหรือเฟิร์มแวร์ของเครื่องคอมพิวเตอร์นั้นๆ ตามมาตรฐาน เรียกว่า Unified Extensible Firmware Interface: UEFI รายละเอียดเพิ่มเติมเกี่ยวกับ GPT ที่ wikipedia เพื่อให้ตาราง GPT สามารถรองรับระบบปฏิบัติชนิด 64 บิต และฮาร์ดดิสค์ที่มีความจุเพิ่มสูงขึ้นระดับเพตะไบต์ (Peta Byte) หรือ 10^{15} ไบต์ หรือประมาณ 1000 เทอราไบต์

7.1 ระบบไฟล์ (File System)



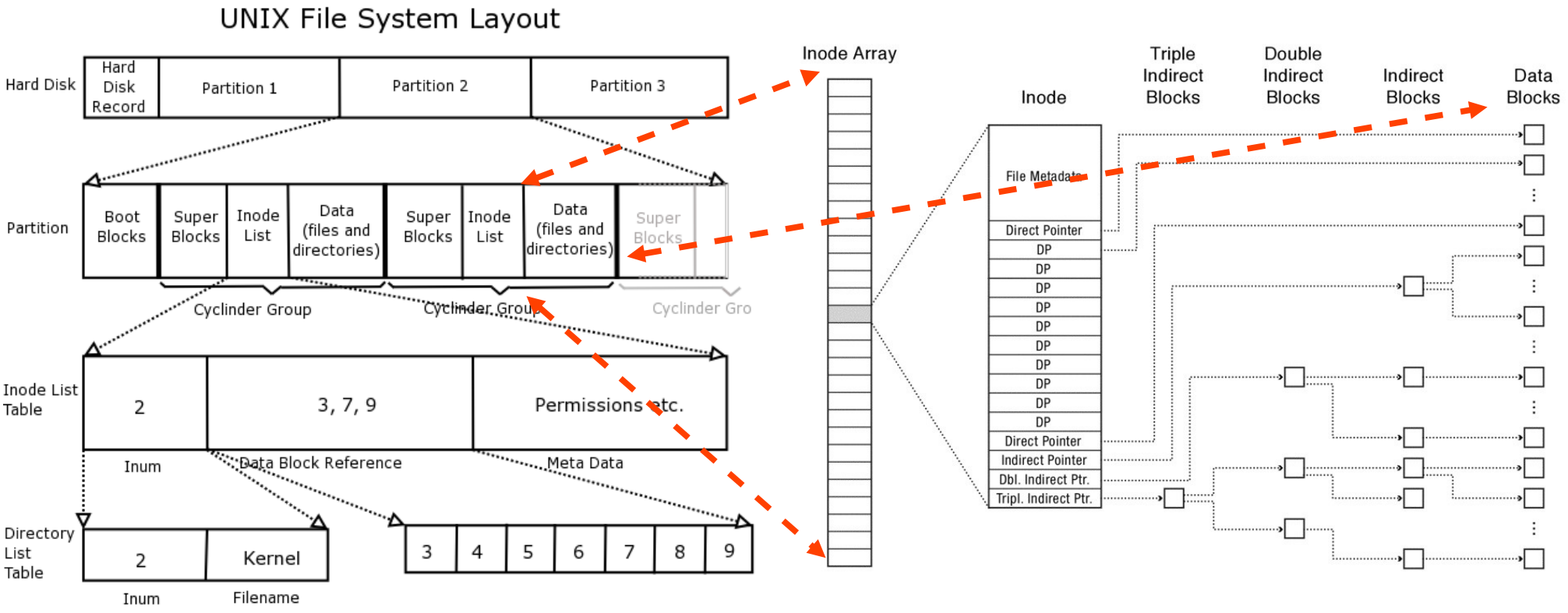


7.1 ระบบไฟล์ (File System)

โครงสร้างของระบบไฟล์ในระบบปฏิบัติการตระกูลยูนิกซ์มีโครงสร้าง และหน้าที่ของระบบไฟล์เป็นชั้นการจัดการ 3 ชั้น โดยเรียงลำดับ ดังนี้

- **ชั้นตรรกะ** (Logical Layer) เป็นชั้นบนสุดทำหน้าที่เชื่อมต่อกับซอฟต์แวร์ประยุกต์ โดยใช้คำสั่งเปิดไฟล์ ปิดไฟล์ อ่านไฟล์ เขียนไฟล์ เป็นต้น ซึ่งไฟล์ที่ซอฟต์แวร์ประยุกต์ต้องการใช้งาน โดยเนื้อหาในบทนี้จะเน้นที่การทำงานในชั้นตรรกะ และชั้นกายภาพ
- **ชั้นเสมือน** (Virtual Layer) ทำหน้าที่เชื่อมต่อระหว่างชั้นตรรกะและชั้นกายภาพ รายละเอียดขึ้นอยู่กับวิธีการพัฒนาโปรแกรมของแต่ละระบบปฏิบัติการ
- **ชั้นกายภาพ** (Physical Layer) เป็นชั้นล่างสุด ทำหน้าที่จัดการบล็อกข้อมูลบนอุปกรณ์เก็บรักษาข้อมูลแต่ละชนิด เพื่อตอบสนองต่อการร้องขอจากชั้นเสมือน ซึ่งกลไกสำคัญคือ การบริหารบัฟเฟอร์ (Buffer) ในหน่วยความจำกายภาพ การเข้าถึงหน่วยความจำกายภาพโดยตรง (DMA) การจัดวางตำแหน่งบล็อกข้อมูลบนอุปกรณ์เหล่านั้น เพื่อประสิทธิภาพการอ่านหรือเขียนดีขึ้น และ การเชื่อมต่อกับดีไวซ์ไดรเวอร์ของอุปกรณ์เก็บรักษาข้อมูล

7.1 ระบบไฟล์ (File System)



7.1 ระบบไฟล์ (File System)

- **ซูเปอร์บล็อก** (Superblock) ทำหน้าที่เก็บรายละเอียดต่างๆ ของระบบไฟล์ ขนาดของบล็อกข้อมูล รายละเอียดการใช้งานบล็อกข้อมูลต่างๆ เช่น สถานะว่างหรือใช้งานอยู่ เป็นต้น
- **ตารางไอโหนด** (Inode Table) หรืออาจเรียกว่า ไอโหนดอาร์เรย์ (Inode Array) หรือ ไอโหนดลิสต์ (Inode List) ทำหน้าที่เก็บโครงสร้างข้อมูล Inode จำนวนหนึ่งภายใต้ไซลินเดอร์กรุปนี้ รายละเอียดเพิ่มเติมในหัวข้อถัดไป
- **บล็อกข้อมูล**จำนวนหนึ่ง (Data Blocks) ทำหน้าที่บรรจุข้อมูลของไฟล์หนึ่งไฟล์ให้เรียงต่อกันไปจนครบขนาดไฟล์ โดยทั่วไปขนาดของบล็อกข้อมูลแต่ละบล็อกมีความจุเท่ากับ **4096** ไบต์ หรือ **4 KiB (kibibyte)** สำหรับระบบปฏิบัติการยูนิกซ์และอื่นๆ ทั้งนี้ขึ้นกับชนิดของเทคโนโลยีและระบบไฟล์ที่ใช้ เมื่อกำหนดขนาดความจุของแต่ละบล็อกแล้ว จำนวนบล็อกข้อมูลจะแปรผันตามขนาดความจุของไซลินเดอร์กรุปนั้นๆ

7.1 ระบบไฟล์ (File System): Inode

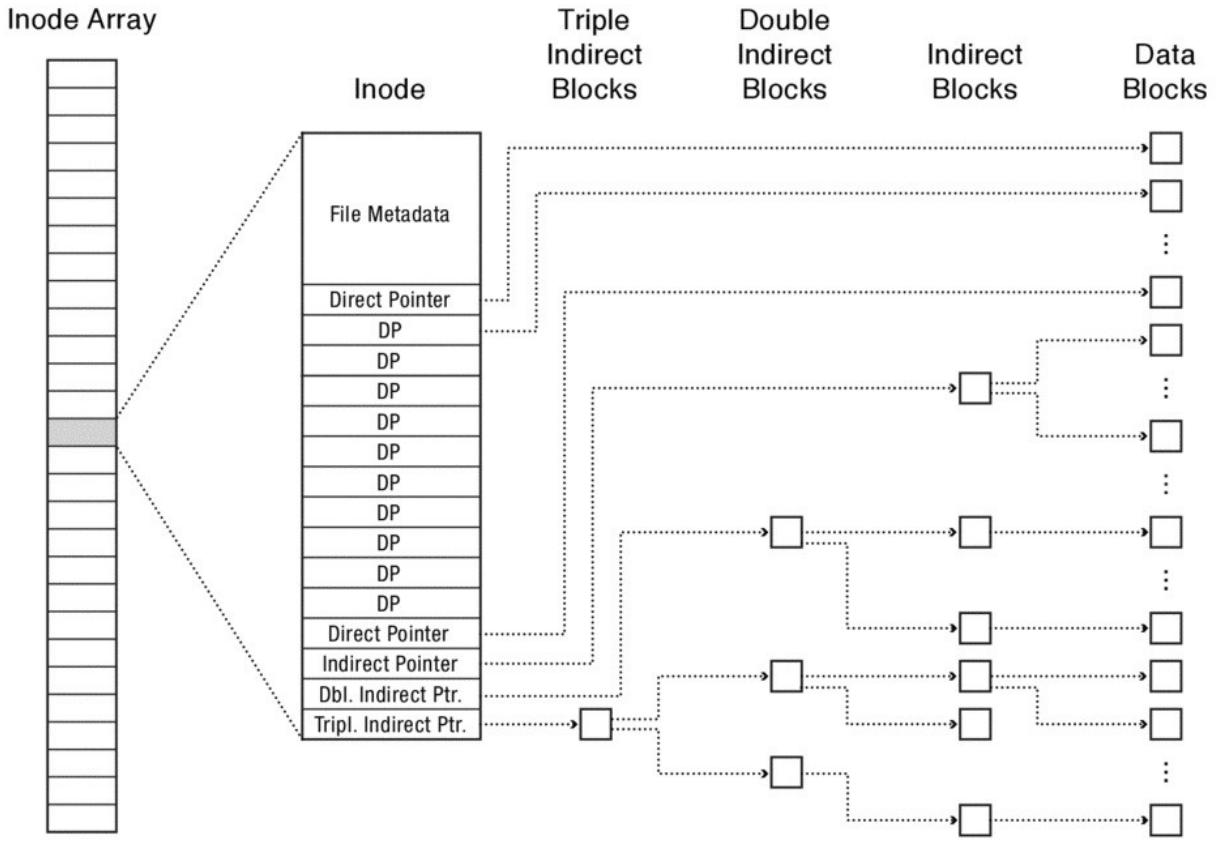
- **หมายเลขไอโหนด** (Inode Number: Inum): หมายเลขประจำตัวของไฟล์หรือไดเรกทอรีนั้นๆ เป็นเลขจำนวนเต็มชนิดไม่มีเครื่องหมายความยาว 32 บิต สำหรับระบบไฟล์ Ext4 โหมด 32 บิต ที่มา: kernel.org
- **รายละเอียดของไฟล์** (FIle Metadata) แบ่งเป็น
  - ชนิดไฟล์: **ไฟล์** (file), **ไดเรกทอรี** (directory), **ไปป์** (pipe) เป็นต้น
    * **ไฟล์** ตามที่เคยนิยามที่ 3.2.1 ในหัวข้อที่ 3.2.6 ว่าเป็นการเรียงตัวกันของตัวเลขฐานสองทีละไบต์ๆ โดยอาศัยพื้นที่จัดเก็บในอุปกรณ์รักษาข้อมูล เรียกว่า **บล็อกข้อมูล** ไฟล์เกิดจากการเรียงของบล็อกข้อมูลอย่างน้อย 1 บล็อกขึ้นไป เพื่อจัดเก็บข้อมูลหรือคำสั่งต่างๆ ลงในอุปกรณ์เก็บรักษาข้อมูล
    * **ไดเรกทอรี** หรือ โฟลเดอร์ คือ ไฟล์ชนิดหนึ่งที่เก็บหมายเลขไอโหนดที่อยู่ภายใต้โครงสร้างนี้โดยไดเรกทอรีสามารถซ้อนกันได้ เพื่อความสะดวกในการจัดหมวดหมู่ของไฟล์และไดเรกทอรี
    * **ไปป์** (pipe) ตำราเล่มนี้ไม่ครอบคลุม ผู้อ่านสามารถค้นคว้าเพิ่มเติมได้จาก Wikipedia

7.1 ระบบไฟล์ (File System): Inode

- **ลิงก์เชื่อมโยงไปยังบล็อกข้อมูล** (Links to Data Blocks) ทำหน้าที่เก็บหมายเลขบล็อกข้อมูล (บล็อกสี่เหลี่ยมจัตุรัส) ซึ่งทำหน้าที่เก็บข้อมูลจริงๆ การเชื่อมโยงบล็อกข้อมูลหลายๆ บล็อกเข้าด้วยกันเป็นไฟล์ 1 ไฟล์ หรือไดเรกทอรี 1 ไดเรกทอรี ลิงก์เชื่อมโยงแบ่งเป็น 2 ชนิดหลัก คือ

  - **ลิงก์หรือพอยน์เตอร์บล็อกข้อมูลทางตรง** (Direct Blocks หรือ Direct Pointers: DP) คือ หมายเลขบล็อกข้อมูลที่เก็บข้อมูลสำหรับไอโหนดนี้ สำหรับไฟล์ขนาดเล็ก ในรูปมีจำนวน DP เท่ากับ 12 ตำแหน่งๆ 8 ไบต์ ทำให้สามารถเก็บหมายเลขบล็อกได้สูงสุด 12 หมายเลข เพื่อรองรับไฟล์ขนาดเล็กที่มีขนาดไม่เกิน 48 KiB (kibibyte) โดยแต่ละบล็อกข้อมูลในระบบไฟล์มีขนาดเท่ากับ 4 KiB (kibibyte)
  - **ลิงก์หรือพอยน์เตอร์บล็อกข้อมูลทางอ้อม** ใช้สำหรับกรณีที่ไฟล์มีขนาดใหญ่กว่า 48 KiB (kibibyte)จนถึงหลายกิบิไบต์ (GiB) ระบบไฟล์สามารถจองจำนวนบล็อกข้อมลมากขึ้นตามขนาดไฟล์จนทำให้จำนวนพอยน์เตอร์บล็อกข้อมูลทางตรงไม่เพียงพอ ดังนั้น พอยน์เตอร์บล็อกข้อมูลทางอ้อมจะทำหน้าที่เก็บหมายเลขบล็อกข้อมูลจำนวนมากๆ เพื่อรองรับไฟล์ที่มีขนาดใหญ่ขึ้นดังกล่าว พอยน์เตอร์บล็อกข้อมูลทางอ้อมแบ่งเป็น 3 ระดับ ดังนี้

7.1 ระบบไฟล์ (File System): Inode



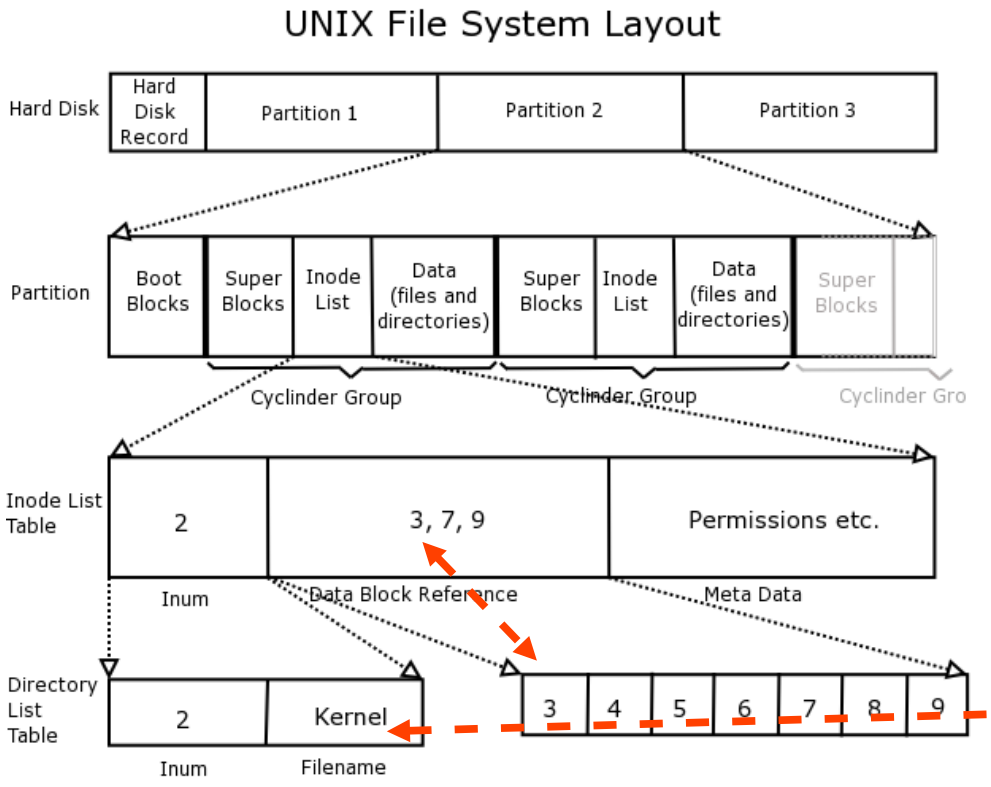
7.1 ระบบไฟล์ (File System): Inode

* **พอยน์เตอร์หนึ่งชั้น** (Indirect Pointer) เมื่อไฟล์มีขนาดใหญ่ขึ้นอีกจนเกิน 48 KiB ทำให้จำนวนพอยน์เตอร์ทางตรงไม่เพียงพอ ระบบไฟล์จะอาศัยพอยน์เตอร์หนึ่งชั้นในโครงสร้างไอโหนด เพื่อเก็บหมายเลขบล็อกพิเศษจำนวนหนึ่ง เรียกว่า **บล็อกทางอ้อมหนึ่งชั้น** (Indirect Blocks) ทำหน้าที่เก็บหมายเลขบล็อกข้อมูลเสริมจากพอยน์เตอร์บล็อกข้อมูลทางตรงที่เต็มแล้ว โปรดสังเกตบล็อกสี่เหลี่ยมจัตุรัสที่ขั้นระหว่างช่องพอยน์เตอร์หนึ่งชั้นและบล็อกข้อมูลในรูปที่ 7.2
* **พอยน์เตอร์สองชั้น** (Double Indirect Pointer) เมื่อไฟล์มีขนาดใหญ่ขึ้นอีกจนพอยน์เตอร์หนึ่งชั้นไม่เพียงพอ ระบบไฟล์จะอาศัยพอยน์เตอร์สองชั้นในโครงสร้างไอโหนด เพื่อเก็บหมายเลขบล็อกพิเศษ เรียกว่า **บล็อกทางอ้อมสองชั้น** (Double Indirect Blocks) ทำหน้าที่เก็บหมายเลขบล็อกทางอ้อมหนึ่งชั้นเสริมจากบล็อกทางอ้อมหนึ่งชั้นที่เต็มแล้ว โปรดสังเกตบล็อกสี่เหลี่ยมจัตุรัสที่ขั้นระหว่างช่องพอยน์เตอร์สองชั้นและบล็อกข้อมูล ในรูปที่ 7.2

7.1 ระบบไฟล์ (File System): Inode

* **พอยน์เตอร์สามชั้น** (Triple Indirect Pointer) เมื่อไฟล์มีขนาดใหญ่ขึ้นอีกจนพอยน์เตอร์สองชั้นไม่เพียงพอ ระบบไฟล์จะอาศัยพอยน์เตอร์สามชั้นในโครงสร้างไอโหนด เพื่อเก็บหมายเลขบล็อกพิเศษจำนวนหนึ่ง เรียกว่า **บล็อกทางอ้อมสามชั้น** (Triple Indirect Blocks) ทำหน้าที่เก็บหมายเลขบล็อกทางอ้อมสองชั้นเสริมจากบล็อกทางอ้อมสองชั้นที่เต็มแล้ว โปรดสังเกตบล็อกสี่เหลี่ยมจัตุรัสที่ขั้นระหว่างช่องพอยน์เตอร์สามชั้นและบล็อกข้อมูล ในรูปที่ 7.2

7.1 ระบบไฟล์ (File System): Inode

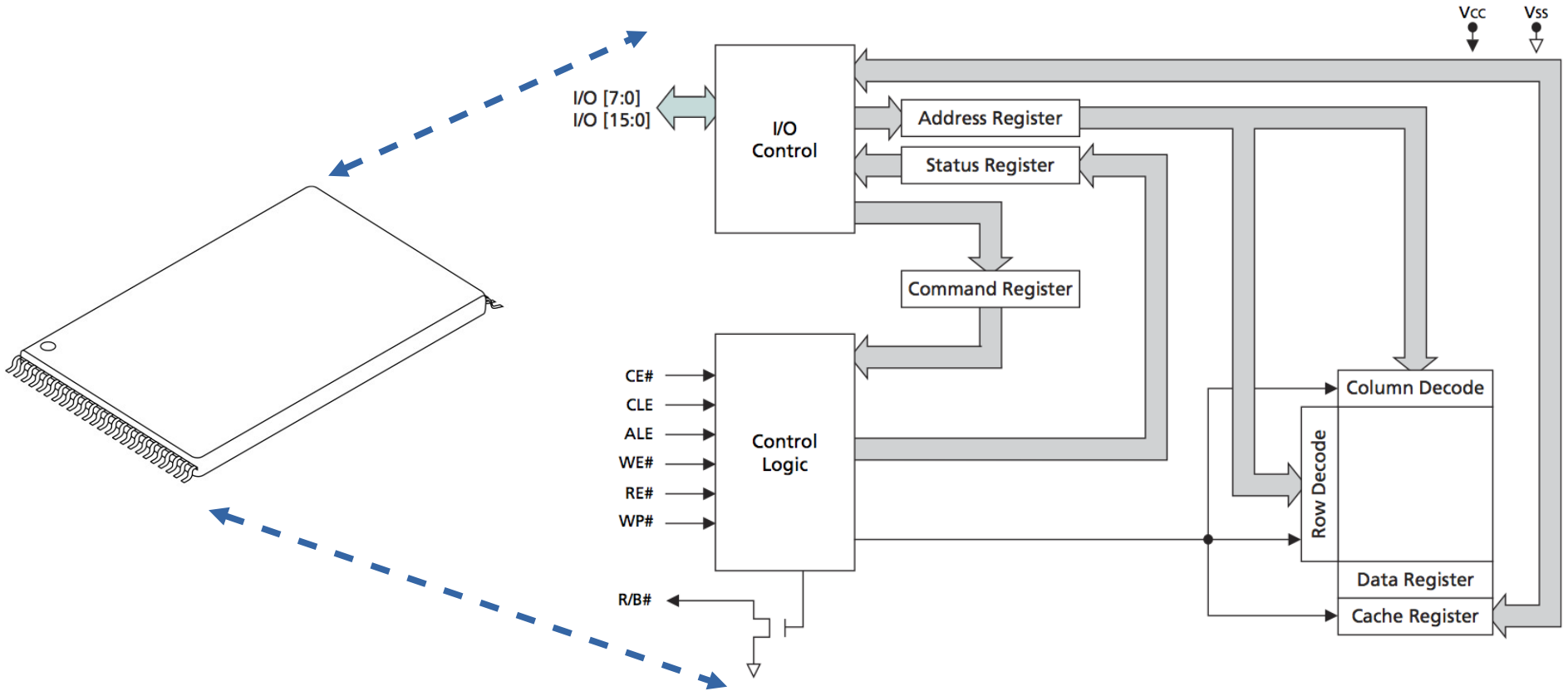


- ไฟล์ชื่อ Kernel ในไดเรคทอรีรูท (/ หรือ root directory)
- มีไอโหนดหมายเลข (Inum = 2) ซึ่งข้อมูลจะบันทึกอยู่ในบล็อกข้อมูลหมายเลข 3, 7, 9
- มีรายละเอียดสิทธิ์ต่างๆ ตามมา เรียกรวมๆ ว่า MetaData
- ส่วนชื่อไฟล์ (Filename) Kernel จะเก็บบันทึกในตารางไดเรคทอรีอ้างอิงโดยใช้ Inum = 2 เป็นหมายเลขอ้างอิง

7.2 ชิปหน่วยความจำแฟลช (Flash Memory Chip)

- หน่วยความจำแฟลช เดิมเรียกว่า แฟลชรอม (Flash ROM)
- คำว่า ROM ย่อมาจากคำว่า Read-Only Memory
- วัตถุประสงค์เดิมของแฟลชรอม ทำหน้าที่เป็นหน่วยความจำที่สร้างจากเทคโนโลยีสารกึ่งตัวนำ โดยมีคุณสมบัติใช้เก็บคำสั่งและข้อมูลเพื่อการอ่านเป็นหลัก
- นักพัฒนาประยุกต์ใช้เก็บคำสั่งประจำเครื่อง หรือ เฟิร์มแวร์ (Firmware) เป็นหลัก
- ด้วยความนิยมในเทคโนโลยีชนิดนี้ หน่วยความจำแฟลชจึงถูกพัฒนาอย่างต่อเนื่องจนสามารถเขียนหรือแก้ไขข้อมูลภายในแฟลชรอมได้รวดเร็วขึ้นจนกลายเป็นอุปกรณ์เก็บรักษาข้อมูล

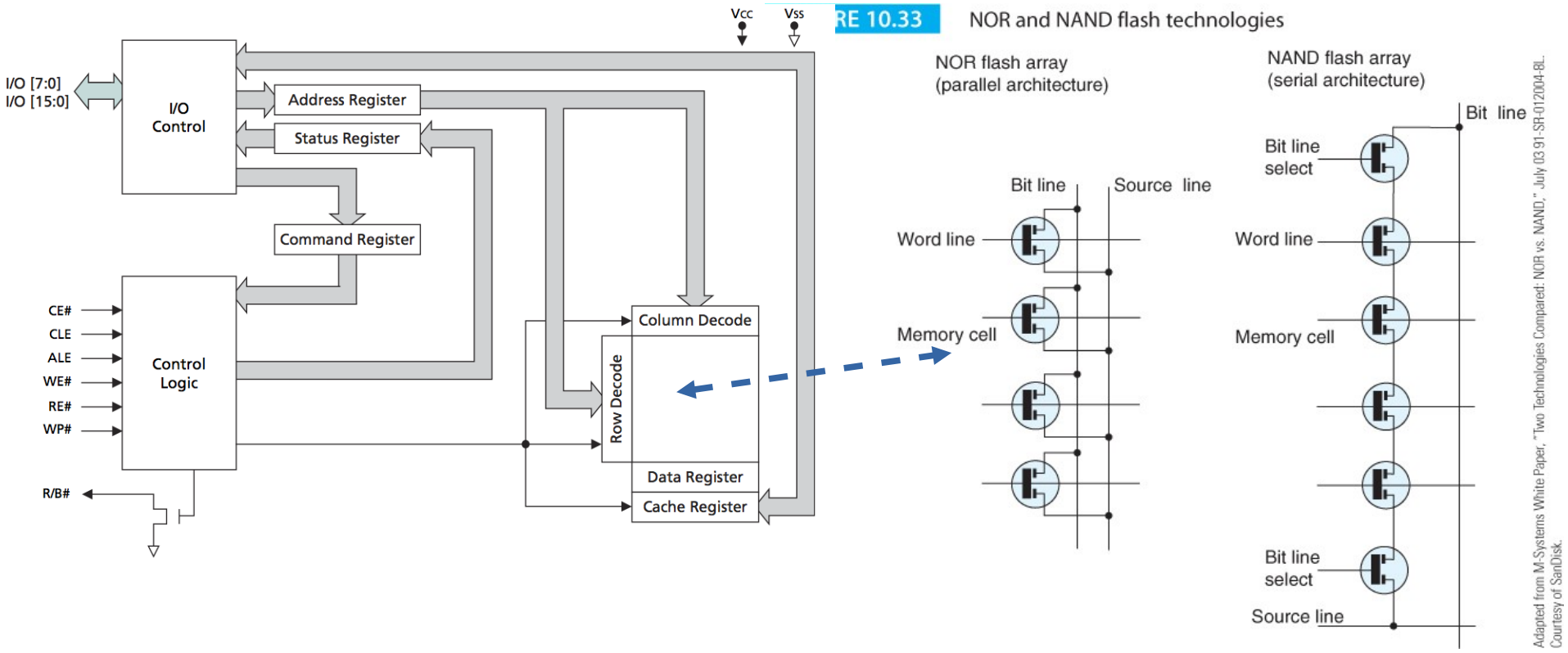
7.2 ชิปหน่วยความจำแฟลช (Flash Memory Chip)



7.2 ชิปหน่วยความจำแฟลช (Flash Memory Chip)

- **ประสิทธิภาพการอ่านข้อมูล**ขึ้นอยู่กับตำแหน่งและรูปแบบการอ่านและรูปแบบการเรียงตัวของข้อมูล เช่น การอ่านข้อมูลที่**เรียงตัวต่อเนื่อง** (Sequential Address) จะใช้**เวลาเข้าถึง** (Access Time) สั้นมากเพียง 30 นาโนวินาที การอ่านข้อมูลที่ไม่เรียงตัวต่อเนื่องและ**สุ่มตำแหน่ง** (Random Address) จะใช้เวลาเข้าถึงนานขึ้นเป็น 25 ไมโครวินาที จะเห็นได้ว่าใช้เวลาเพิ่มขึ้นเกือบ 1000 เท่า เทียบกับการอ่านข้อมูลแบบเรียงตัวต่อเนื่อง
- **ประสิทธิภาพการเขียนข้อมูล**มีลักษณะเช่นเดียวกับการอ่านข้อมูล โดย การเขียนข้อมูลต้องเขียนครั้งละเพจ (Page Program) หรือ 2048 ไบต์ 'ซึ่งจะใช้เวลาเข้าถึงยาวนานกว่าการอ่านข้อมูลเป็น 300 ไมโครวินาที ใช้เวลาเพิ่มขึ้นเป็น 10 เท่า เทียบกับการอ่านข้อมูลแบบสุ่ม
- **ประสิทธิภาพการลบข้อมูล** (Erase) จะใช้เวลาเพิ่มขึ้นสูงมากถึง 2 มิลลิวินาที จะเห็นได้ว่าใช้เวลาเพิ่มขึ้นเกือบ 100 เท่าเทียบกับการเขียนข้อมูลจำนวนหนึ่งเพจ

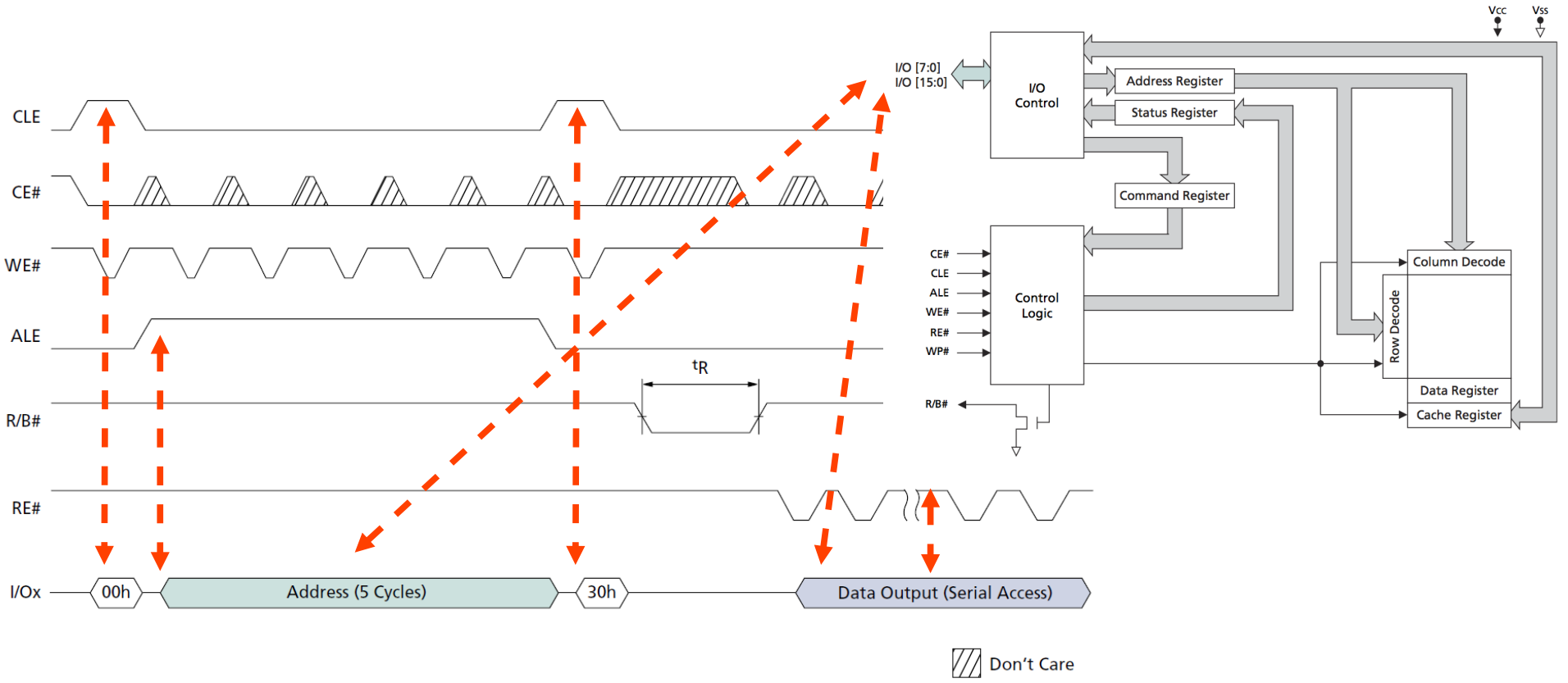
7.2 ชิปหน่วยความจำแฟลช (Flash Memory Chip)



7.2 ชิปหน่วยความจำแฟลช (Flash Memory Chip)

- โครงสร้างภายในชิปแฟลช NAND มีลักษณะคล้ายกับหน่วยความจำ SRAM และ DRAM คือมี อะเรย์ของเซลหน่วยความจำเป็นหลัก และ
- วงจรควบคุมการอ่านหรือเขียนข้อมูลลงไปในอะเรย์นี้ จะเข้าถึงโดยใช้ แอดเดรสแถว (เลขแถว) และ แอดเดรสคอลัมน์ (เลขคอลัมน์)
- เพื่อบ่งชี้ตำแหน่งเซลนั้นๆ ในอะเรย์ แต่การทำงานของหน่วยความจำแฟลชมีความซับซ้อนกว่า

7.2 ชิปหน่วยความจำแฟลช (Flash Memory Chip): Read



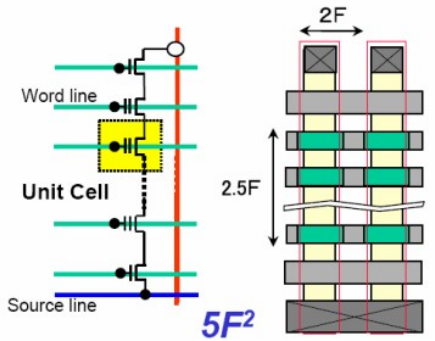
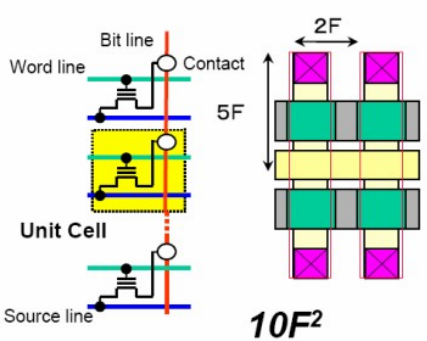
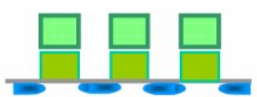
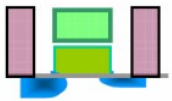
7.2 ชิปหน่วยความจำแฟลช (Flash Memory Chip): Read

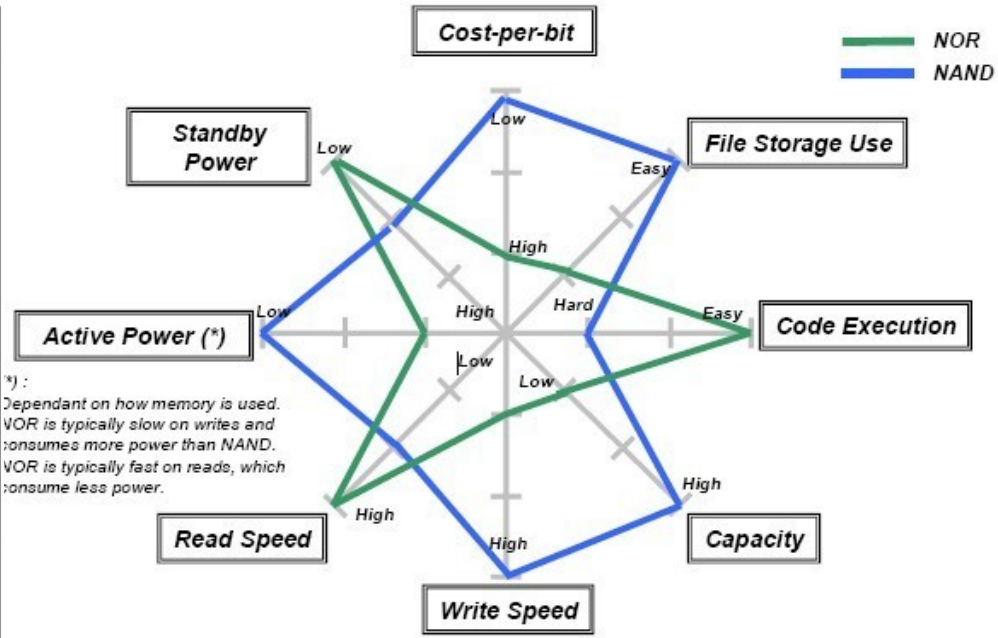
- สัญญาณ CLE เปลี่ยนจาก 0 เป็น 1 เพื่ออ่านค่าคำสั่งที่ปรากฏบนบัส I/O
- สัญญาณ CE# เปลี่ยนจาก 1 เป็น 0 เพื่อสั่งชิปให้เริ่มต้นทำงาน
- สัญญาณ WE# จะมีการเปลี่ยนแปลงจาก 1 เป็น 0 สลับไปมา เพื่อเขียนคำสั่งและแอดเดรสทางขา I/Ox ไปเก็บพักในรีจิสเตอร์ต่างๆ
- สัญญาณ ALE จะเปลี่ยนแปลงจาก 0 เป็น 1 ในระหว่างที่บัส I/O รับแอดเดรสเป็นระยะเวลา 5 คาบเวลา (Cycles)
- สัญญาณ R/B# จะเปลี่ยนแปลงจาก 1 เป็น 0 ระยะเวลาหนึ่ง t_R เพื่อบ่งบอกว่าชิป มีสถานะยุ่ง (Busy) แล้วกลับไปเป็น 1 เพื่อบ่งบอกว่าชิปพร้อม

7.2 ชิปหน่วยความจำแฟลช (Flash Memory Chip): Read

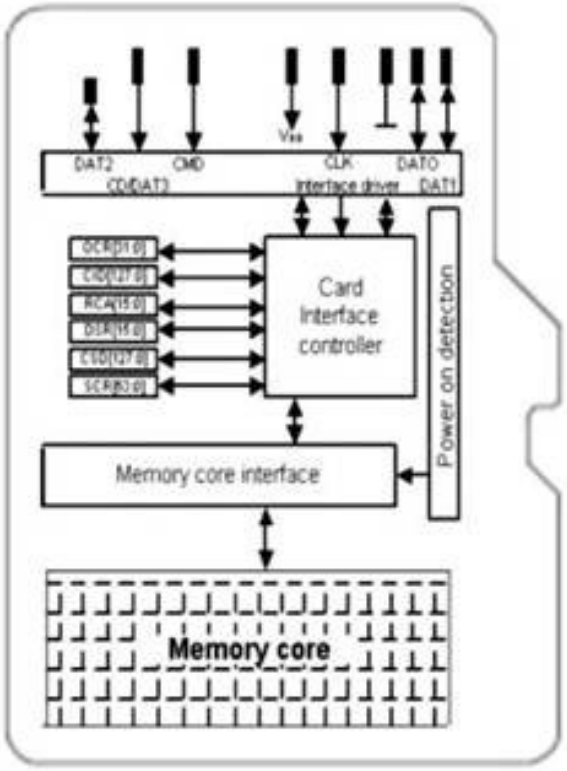
- สัญญาณ RE# จะเปลี่ยนแปลงจาก 1 เป็น 0 สลับไปมา เมื่อตรวจจับว่าสัญญาณ R/B# เปลี่ยนจาก 0 เป็น 1 เพื่ออ่านข้อมูลทางขา I/Ox โดยใช้ขอบขาขึ้นหรือขอบขาลงของสัญญาณ RE#
- สัญญาณ I/O[0:15] ทั้งหมด 16 ขา จะเปลี่ยนแปลงตามลำดับดังนี้
  - คำสั่ง 00h จำนวน 1 คาบเวลา - แอดเดรส จำนวน 5 คาบเวลา - คำสั่ง 30h จำนวน 1 คาบเวลา
- ข้อมูล จำนวนหลายคาบเวลาตามขนาดของเพจข้อมูลโดยอ่านข้อมูลเรียงตามลำดับหมายเลขแอดเดรส (Serial Access)

7.2 ชิปหน่วยความจำแฟลช (Flash Memory Chip)

	NAND	NOR
Cell Array & Size		
Cross-section		
Features	Small Cell Size, High Density Low Power & Good Endurance ➔ *Mass Storage*	Large Cell Current, Fast Random Access ➔ *Code Storage*



7.3 การ์ดหน่วยความจำ SD (Secure Digital)



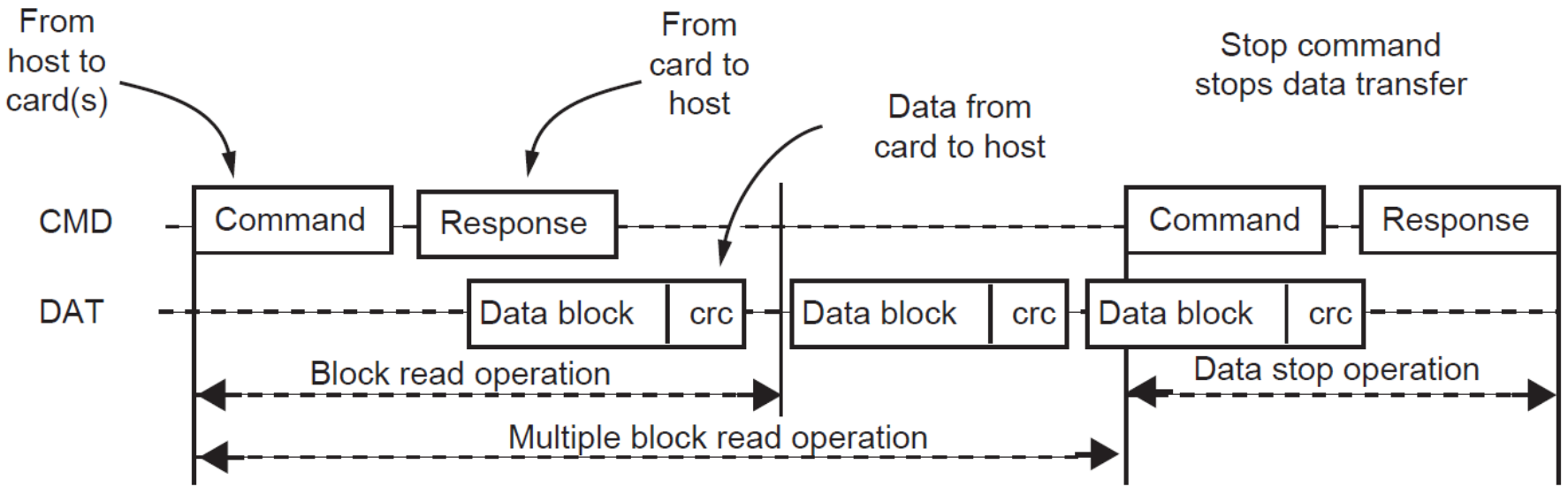
ขา	ชื่อ	วัตถุประสงค์
1	DAT2	Data บิทที่ 2
2	DAT3/CD	Data บิทที่ 3 หรือ Card Detect
3	CMD	ขารับคำสั่ง (Command)
4	VDD	ขั้วบวกแหล่งจ่ายไฟ
5	CLK	สัญญาณคล็อก
6	VSS	ขากราวด์แหล่งจ่ายไฟ
7	DAT0	Data บิทที่ 0
8	DAT1	Data บิทที่ 1

- CMD17: Read Single Block
- CMD18: Read Multiple Block
- CMD24: Write Single Block
- CMD25: Write Multiple Block
- CMD32: Erase Block Start
- CMD33: Erase Block End
- CMD38: Erase

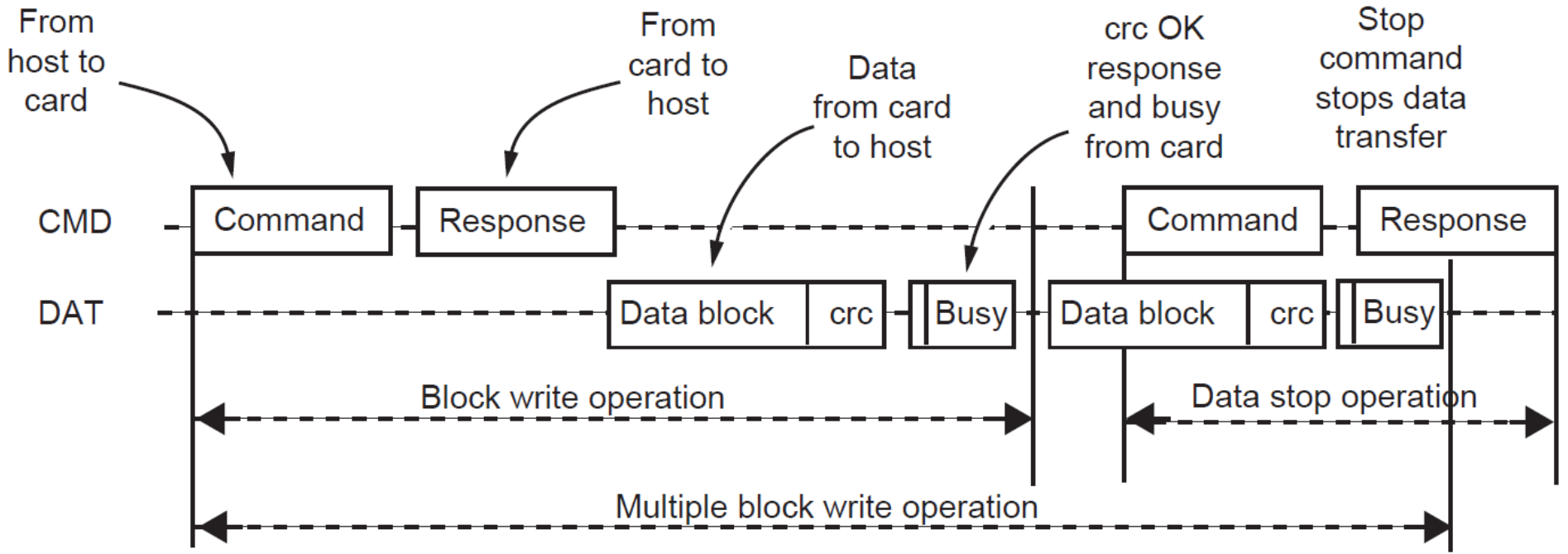
7.3 การ์ดหน่วยความจำ SD (Secure Digital)

- อะเรย์เซลหน่วยความจำชนิดแฟลช ตามขนาดของความจุที่ต้องการใช้
- Memory Core Interface เพื่อเชื่อมต่อกับหน่วยความจำ
- วงจรควบคุม Card Interface Controller เพื่อเชื่อมต่อกับโฮสท์ ทำงานร่วมกับรีจิสเตอร์ต่างๆ เหล่านี้
  - รีจิสเตอร์ CID[27:0] (Card Identification) ขนาด 28 บิต เพื่อเก็บหมายเลขประจำตัวการ์ด
  - รีจิสเตอร์ OCR[31:0] (Operation Condition Register) ขนาด 32 บิต เพื่อเก็บสถานะการทำงานของการ์ด
  - รีจิสเตอร์ RCA[15:0] (Relative Card Address) ขนาด 32 บิต เพื่อเก็บหมายเลขแอดเดรสของการ์ด ซึ่งจะเปลี่ยนแปลงระหว่างที่ถอดเข้าออกจากระบบ
  - รีจิสเตอร์ CSD[27:0] (Card Specific Data) ขนาด 28 บิต เพื่อเก็บสถานะจำเพาะของการ์ด
  - รีจิสเตอร์ SCR[63:0] (SD Configuration Register) ขนาด 64 บิต เพื่อเก็บการตั้งค่าพิเศษประจำตัวการ์ด

7.3 การ์ดหน่วยความจำ SD (Secure Digital): Read



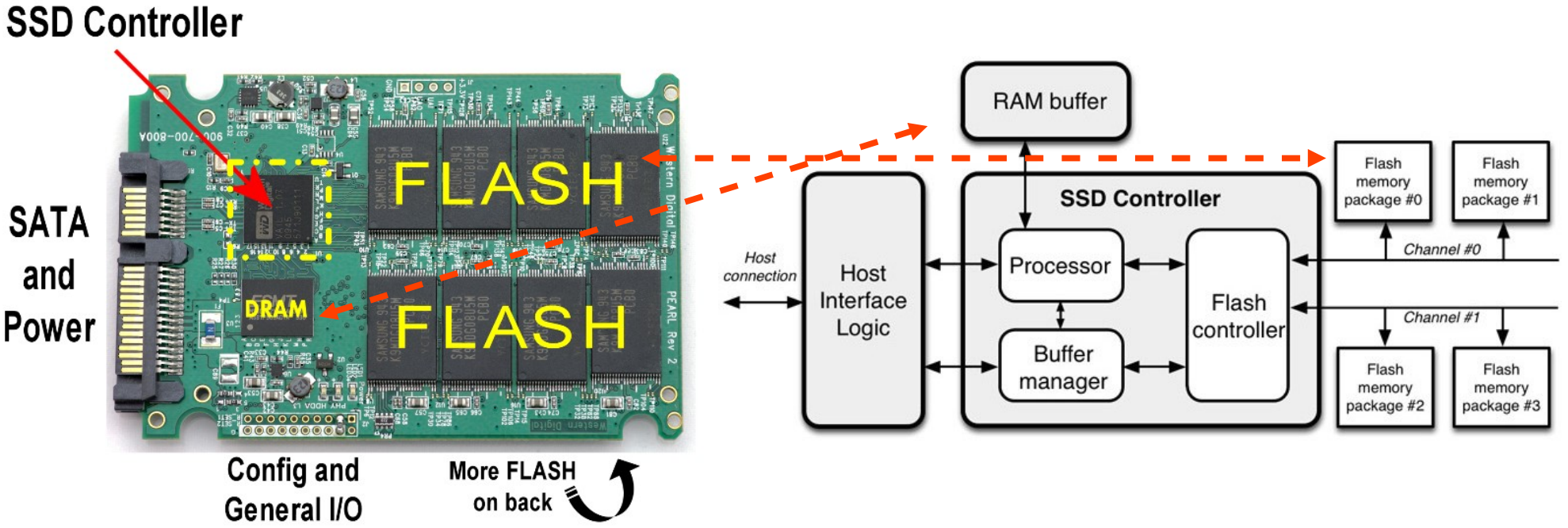
7.3 การ์ดหน่วยความจำ SD (Secure Digital): Write



7.4 โซลิดสเตทไดรฟ์ (Solid-State Drive: SSD)

- ความจุของ SSD มีแนวโน้มเพิ่มสูงขึ้น ทำให้ SSD มีขนาดเริ่มต้นตั้งแต่ 120-128 กิกะไบต์ขึ้นไป
- แนวโน้มของ SSD ต้นทุน/ความจุจะถูกลงเรื่อยๆ จนใกล้เคียงฮาร์ดดิสค์ในอนาคต โดยองค์ประกอบหลักที่สำคัญ คือ หน่วยความจำแฟลช NAND ที่มีความจุต่อชิปเพิ่มสูงขึ้นไปอีก
- ใช้หน่วยความจำ DRAM เพื่อทำหน้าที่เป็นแคช หรือ บัฟเฟอร์ เพื่อเพิ่มประสิทธิภาพการทำงานให้รวดเร็วขึ้น
- SSD ใช้ชื่อที่ใกล้เคียงกับฮาร์ดดิสค์ไดรว์ เพื่อความหมายที่ใกล้เคียงกัน
- ชิปหน่วยความจำแฟลช NAND ความจุสูงเรียงตัวต่อเนื่องกันจนได้ความจุมากพอ
- การจัดเรียงทั้งด้านบน (Channel 0) และด้านล่าง (Channel 1) ของแผ่นวงจรพิมพ์หลักของหน่วยความจำแฟลช และอาศัยชิพหน่วยความจำไดนามิคแรมทำหน้าที่เป็นแคช

7.4 โซลิดสเตทไดรฟ์ (Solid-State Drive: SSD)



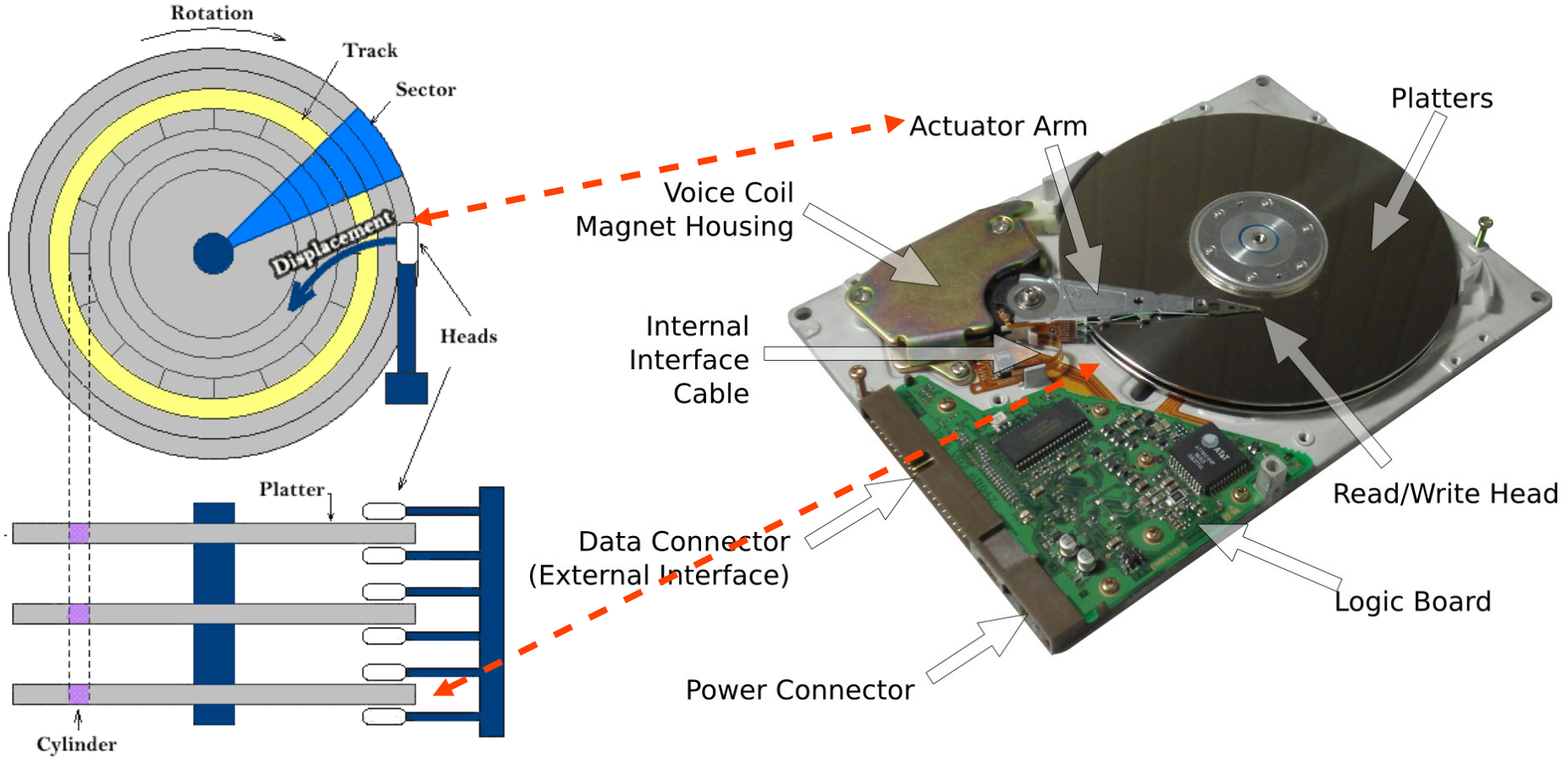
7.4 โซลิดสเตทไดรฟ์ (Solid-State Drive: SSD)

- **โพรเซสเซอร์** (Processor) สำหรับรันคำสั่งในเฟิร์มแวร์สำหรับควบคุมการทำงานภายในและเชื่อมต่อกับคอมพิวเตอร์โฮสท์ ดังนั้น ผู้อ่านควรตรวจสอบผู้ผลิตเป็นระยะๆ ว่ามีการอัพเดทเฟิร์มแวร์ของ SSD รุ่นที่ใช้หรือไม่
- **แฟลชคอนโทรลเลอร์** (Flash Controller) สำหรับควบคุมการอ่าน/เขียนข้อมูลหน่วยความจำแฟลช คล้ายกับการทำงานของหน่วยความจำ SD ซึ่งกล่าวในหัวข้อที่ 7.2
- **บัฟเฟอร์** (Buffer) และ **การบริหารจัดการบัฟเฟอร์** (Buffer Management) อาศัยหน่วยความจำ DRAM เป็นบัฟเฟอร์เพื่อพักเก็บข้อมูลชั่วคราวระหว่างที่เชื่อมต่อกับคอมพิวเตอร์โฮสท์ ทำให้การอ่าน/เขียนมีระยะเวลาเข้าถึงเฉลี่ย ดังนั้น การใช้หน่วยความจำ DRAM ให้เต็มประสิทธิภาพ และคุ้มค่า จึงต้องมีการบริหารจัดการบัฟเฟอร์ ทำให้ต้นทุนการผลิตต่ำลง

7.5 ฮาร์ดดิสก์ไดรฟ์ (Hard Disk Drive: HDD)

- อุปกรณ์เก็บรักษาข้อมูลที่ได้รับความนิยมจากในอดีต และพัฒนามาเป็นเวลายาวนาน
- แผ่นจานแม่เหล็กหมุนมีเส้นผ่าศูนย์กลางสองขนาดที่นิยมผลิต คือ 2.5 นิ้ว และ 3.5 นิ้ว หมุนด้วยความเร็วสูงตั้งแต่ 5,400 ถึง 10,000 รอบต่อนาที
- จุดเด่นของหน่วยความจำชนิดสารแม่เหล็ก คือ ความจุข้อมูลที่มากกว่า เริ่มต้นที่หลายร้อยกิกะไบต์จนถึงหลายเทอราไบต์ (Tera Byte) โดย 1 เทอราไบต์ ประมาณเท่ากับ 1000 กิกะไบต์
- ราคาต่อความจุต่ำลง ต้นทุนโดยรวมจึงถูกลง มีอายุการใช้งานที่ยาวนานพอสมควร

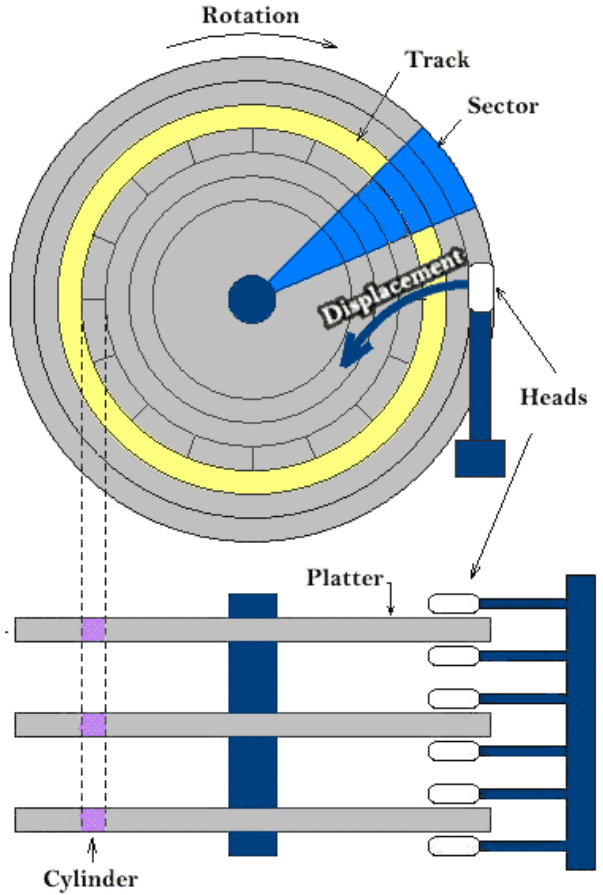
7.5 ฮาร์ดดิสก์ไดรฟ์ (Hard Disk Drive: HDD)

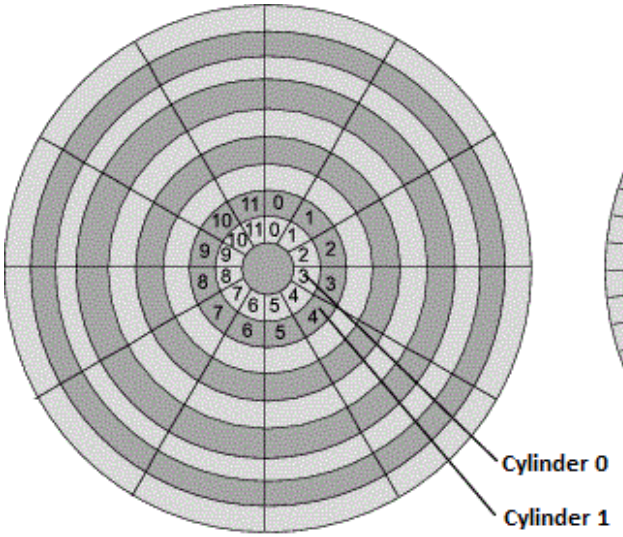


7.5 ฮาร์ดดิสก์ไดรฟ์ (Hard Disk Drive: HDD)

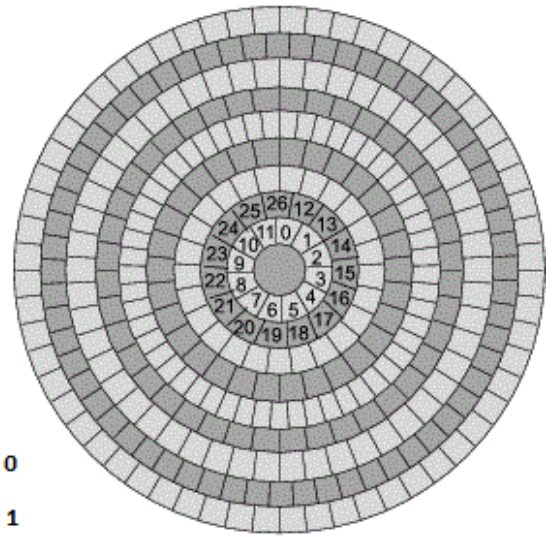
- แผ่นจานแม่เหล็กแบ่งเป็นหลายๆ แทร็ก หรือ วงรอบ แต่ละแทร็คจะมีจำนวนเซ็กเตอร์เท่าๆ กัน
- พื้นที่หนึ่งเซ็กเตอร์ มีความจุ 512 ไบต์เสมอ โดยจะแบ่งพื้นที่ในแนวรัศมีจากจุดศูนย์กลางมายังขอบจาน
- ไซลินเดอร์ (Cylinder) คือ แทร็กที่อยู่บนจานแม่เหล็กต่างๆ และอยู่ห่างจากจุดศูนย์กลางเท่ากัน เรียงตัวกันเป็นทรงกระบอก โดยจะนับจากไซลินเดอร์หรือแทร็กหมายเลข 0 ซึ่งอยู่ขอบนอกสุดของจานแม่เหล็ก โดยจะนับจากขอบนอกสุดเข้าสู่จุดศูนย์กลาง
- บล็อก หรือ คลัสเตอร์ประกอบด้วย เซ็กเตอร์ที่ต่อเนื่องกัน จำนวน 2^n เซ็กเตอร์เสมอ เช่น 2 4 8 .. เซ็กเตอร์ ในแทร็กเดียวกัน โดยระบบปฏิบัติการจะกำหนดจำนวนเซ็กเตอร์ที่ต้องการ ยกตัวอย่างเช่น คลัสเตอร์ขนาด 4096 ไบต์ต้องการพื้นที่จำนวน 8 เซ็กเตอร์
- หมายเลขบล็อก LBA: Logical Block Addressing คือ การตั้งหมายเลขบล็อกที่กล่าวมาก่อนหน้านี้ให้เรียงตัวตามหมายเลขไซลินเดอร์ หมายเลขหัวอ่าน และหมายเลขแทร็ก ทำให้ง่ายต่อการบริหารจัดการ และเป็นพื้นฐานของระบบบริหารจัดการไฟล์

7.5 ฮาร์ดดิสก์ไดรฟ์ (Hard Disk Drive: HDD)



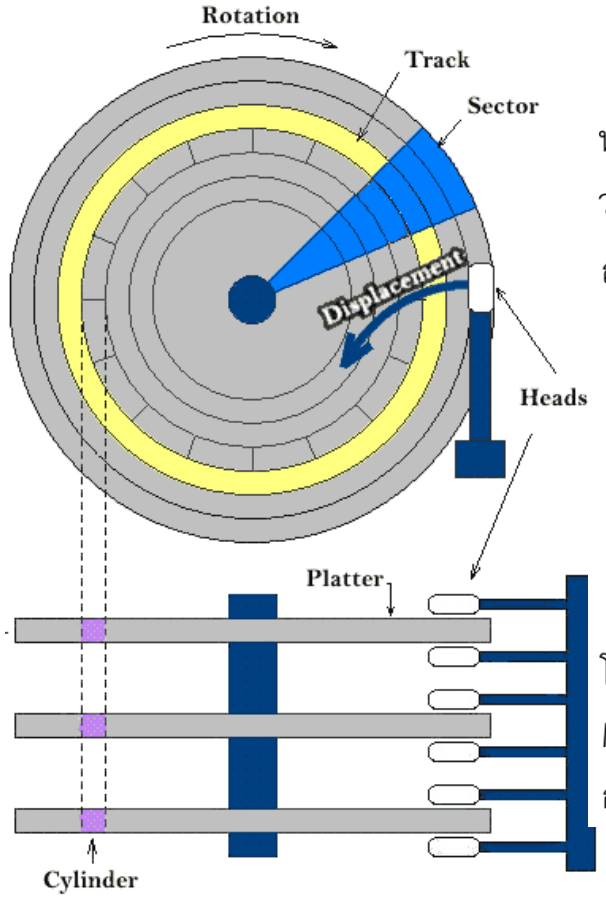


CHS Adressing



LBA Adressing

7.5 ฮาร์ดดิสก์ไดรฟ์ (Hard Disk Drive: HDD)



การวัดประสิทธิภาพของฮาร์ดดิสค์ อาศัยการวัด**เวลาการเข้าถึง** (T_{acc}, Access Time/Latency) หน่วยเป็น มิลลิวินาที ประกอบด้วย ช่วงเวลาการรอเฉลี่ยให้ตำแหน่งที่ต้องการอ่านหมุนมายังหัวอ่าน เรียกว่า **Rotation Latency** T_{rotate} ช่วงเวลาการขยับหัวอ่านมายังแทร็คที่ต้องการ T_{head} และช่วงเวลาการถ่ายโอนข้อมูล (T_{trans}, Transfer Time)

$$T_{acc} = T_{rotate} + T_{head} + T_{transf} \tag{7.1}$$

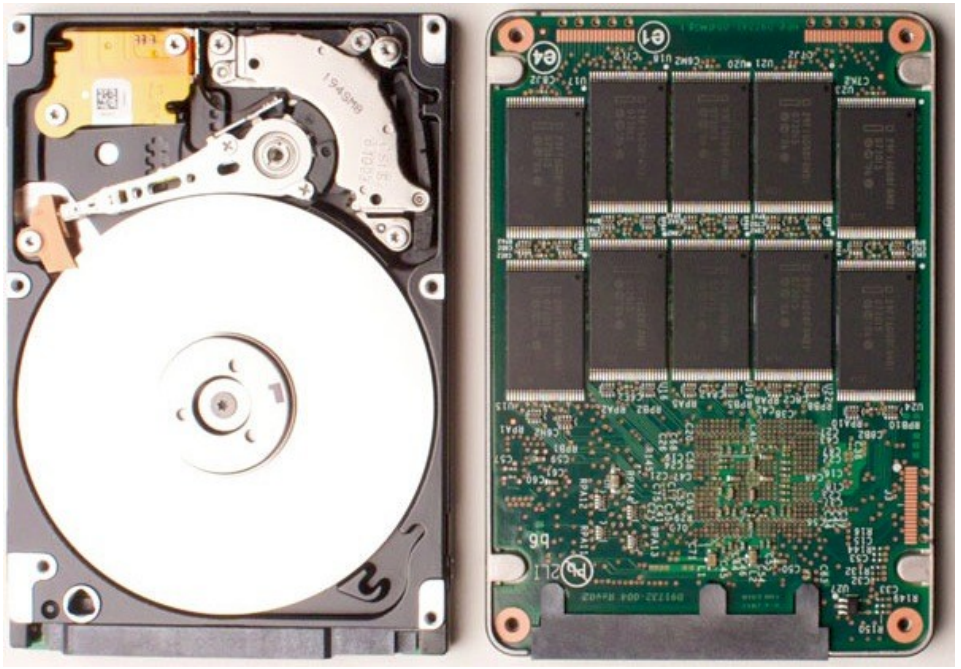
ดังนั้น เวลาการเข้าถึงจึงมีความสัมพันธ์กับความเร็วรอบในการหมุนของจานแม่เหล็ก

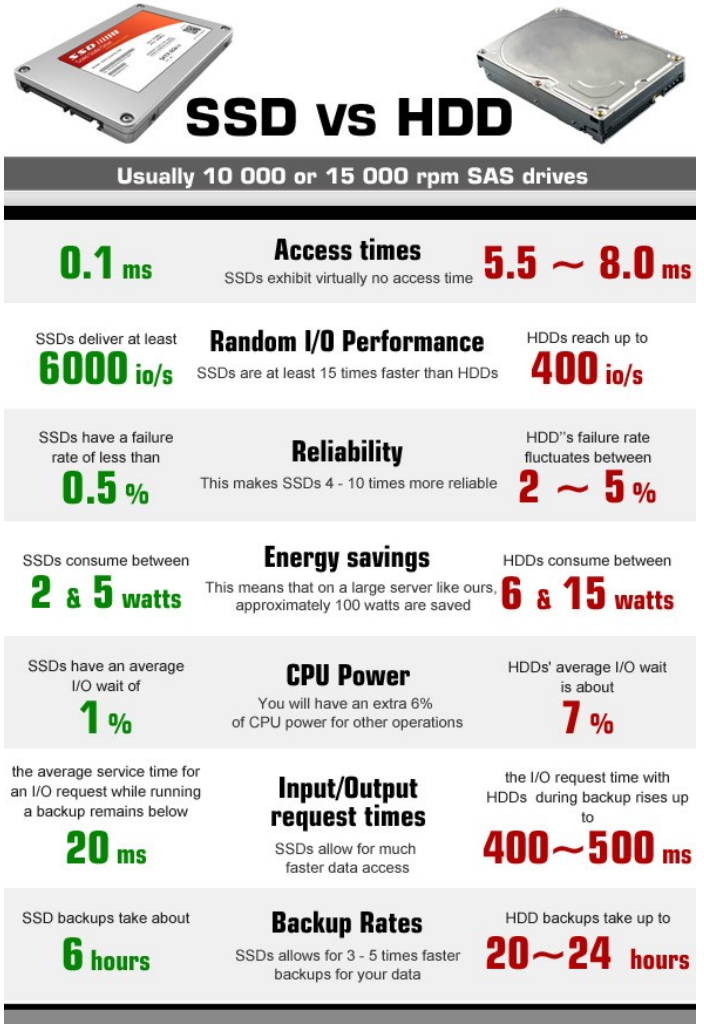
$$T_{rotate} = \frac{0.5}{V_{rotate}} \tag{7.2}$$

โดยแปรผกผันกับ V_{rotate} หรือ ความเร็วรอบในการหมุนของจาน หน่วยเป็นรอบต่อวินาที (Round per Minute: RPM) และตำแหน่งของแทร็คที่ต้องขยับหัวอ่านไป ประสิทธิภาพการอ่านและการเขียน จะขึ้นกับตำแหน่งของข้อมูล เช่นเดียวกับการอ่านและเขียนข้อมูลของหน่วยความจำแฟลช ประสิทธิภาพการ

7.5 ฮาร์ดดิสก์ไดรฟ์

(Hard Disk Drive: HDD)





สรุปท้ายบท

ตารางที่ 7.3: การเปรียบเทียบประสิทธิภาพของอุปกรณ์เก็บรักษาข้อมูลชนิดต่างๆ

อุปกรณ์	แฟลช NAND	SSD	HDD
ผู้ผลิต	Micron	Micron	Western Digital
หมายเลขโมเดล	MT29F2G08AABWP	MTFDDAK120MAV	5K1000
ที่มา:	micron.com	micron.com	hgst.com
ปี ค.ศ.	2004	2013	2016
ความจุ	25 MiB	120 KiB	1000 กิกะไบต์ (GB)
การอ่าน: เวลาเข้าถึง			
- $T_{acc,avg}$	30 นาโนวินาที	160 ไมโครวินาที	5.5 มิลลิวินาที
- $T_{acc,max}$	25 ไมโครวินาที	5 มิลลิวินาที	-
การเขียน: เวลาเข้าถึง			
- $T_{acc,avg}$	300 ไมโครวินาที	40 ไมโครวินาที	5.5 มิลลิวินาที
- $T_{acc,max}$	2 มิลลิวินาที	25 มิลลิวินาที	-
โวลเตจสูงสุด	4.6 โวลต์	5.0 โวลต์	5.0 โวลต์
กำลังไฟสูงสุด	23 มิลลิวัตต์	150 มิลลิวัตต์	1.6 วัตต์

สรุปท้ายบท

- ระบบไฟล์และอุปกรณ์เก็บรักษาข้อมูลจะต้องประสานงานกัน เพื่อให้เคอร์เนล ผู้ใช้งานและแอพพลิเคชันอื่นๆ สามารถบริหารจัดการไฟล์โปรแกรม ไฟล์ข้อมูลต่างๆ ได้อย่างถูกต้อง และมีประสิทธิภาพสอดคล้องกับภารกิจของระบบคอมพิวเตอร์
- ขนาดหรือความจุของบล็อกข้อมูลในอุปกรณ์เก็บรักษาข้อมูล เช่น หน่วยความจำแฟลชร็อม จะมีขนาดเท่ากับ 2^{11} หรือ 2048 ไบท์ ฮาร์ดดิสค์ จะมีขนาดเท่ากับ 2^{9} หรือ 512 ไบท์ สอดคล้องกับ ขนาดของบล็อกข้อมูลในระบบบริหารจัดการไฟล์ และ
- ขนาดของเพจข้อมูลในหน่วยความจำสเมือน ซึ่งจะมีขนาดเป็นจำนวนเท่าของสองเสมอ และสำหรับลีนุกซ์มีขนาดเท่ากับ 4096 หรือ 2^{12} ไบท์
- อุปกรณ์เก็บรักษาข้อมูลจะเชื่อมต่อกับหน่วยความจำผ่านวงจรด้านอินพุท/เอาท์พุท โดยใช้กลไก Direct Memory Access และ กลไกการทำ Interrupt ในชั้น Physical ร่วมกับกลไก Memory Mapped File ในระดับสูงขึ้น

References

- Stewart Weiss, Chapter 3 File Systems and the File Hierarchy, UNIX Lecture Notes, http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/unix_lecture_notes/chapter_03.pdf
- https://www.acerspace.com/ssd-solid-state-drive/
- 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
- Thomas Anderson and Michael Dahlin, Operating Systems: Principles and Practice 2nd Edition, Recursive Books, 2014
- https://learn.adafruit.com/resizing-raspberry-pi-boot-partition/edit-partitions

References

- Micron Technology, I. (2004). Nand flash memory: Mt29f2g08aabwp/mt29f2g16aabwp/mt29f4g08babwp/mt29f4g16babwp/mt29f8g08fabwp.
- https://www.raspberrypi.org/forums/viewtopic.php?t=63750
- 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
- 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://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
- 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