บทที่ 6
หน่วยความจำ

~~~~~~

หน่วยความจำ หรือเมมโมรี (Memory) เป็นส่วนประกอบหนึ่งที่อยู่บนเมนบอร์ดของคอมพิวเตอร์ที่มีความสำคัญและทำงานร่วมกับโพรเซสเซอร์อย่างใกล้ชิด หน่วยความจำทำหน้าที่เก็บโปรแกรมและข้อมูลในส่วนที่โพรเซสเซอร์เขียนและอ่านข้อมูล

หน่วยความจำของคอมพิวเตอร์จะใช้ในการเก็บคำสั่งและข้อมูลในขณะที่ระบบคอมพิวเตอร์มีการประมวลผล ซึ่งจะมีหน่วยความจำอยู่ 2 ประเภทคือ หน่วยความจำภายใน (Internal Memory) และหน่วยความจำภายนอก (External Memory)

หน่วยความจำภายในไม่ได้หมายถึงหน่วยความจำหลัก (Main Memory) เท่านั้น แต่หมายถึงหน่วยความจำที่อยู่ภายในซีพียู (Local Memory) นอกจากนี้ภายในตัวหน่วยควบคุม (CU) ซึ่งเป็นส่วนหนึ่งของซีพียู ก็ต้องการมีหน่วยความจำเป็นของตัวเองด้วย

ส่วนหน่วยความจำภายนอกหมายถึง อุปกรณ์เก็บข้อมูลอื่น ๆ ที่มักเรียกว่าเป็น Peripheral Storage Devices เช่น ดิสก์ เทป ซึ่งติดต่อกับซีพียูด้วย I/O Controller

6.1 ชนิดของหน่วยความจำ

หน่วยความจำของเครื่องคอมพิวเตอร์ใช้ในการเก็บคำสั่งในตัวโปรแกรมและข้อมูลที่ใช้ ซึ่งทั้งสองส่วนนี้จะถูกเก็บลงในหน่วยความจำในขณะที่ซีพียูทำการประมวลผล ขนาดของหน่วยความจำจะเป็นไบต์ (Byte) เช่น 32 MByte แต่ละไบต์จะมีขนาด 8 bit ซึ่งข้อมูลแต่ละบิตจะมีค่าได้ 2 อย่างคือ 0 กับ 1 ดังนั้นข้อมูลขนาด 1 ไบต์ จึงมีค่าต่างกันได้ถึง 256 ค่า ข้อมูลทุกประเภทจะถูกเก็บโดยการแปลงเป็นบิตลงในหน่วยความจำ เมื่อจะนำขึ้นมาใช้ก็แปลงกลับเป็นข้อมูลที่ต้องการ

6.1.1 คุณลักษณะทั่วไปของหน่วยความจำ

1) ตำแหน่งที่ตั้งของหน่วยความจำ

หน่วยความจำภายในซีพียูเป็นหน่วยความจำที่มีอยู่ภายในตัวซีพียู เนื่องจากการทำงานของซีพียูต้องการพื้นที่สำหรับจัดเก็บข้อมูลและคำสั่งของตนเอง ดังนั้นจึงต้องมีหน่วยความจำขนาดเล็กอยู่ภายในเรียกว่า รีจิสเตอร์ (Register) ซึ่งรีจิสเตอร์สามารถส่งผ่านข้อมูลได้เร็วมาก ถ้ายิ่งมีรีจิสเตอร์จำนวนมากเท่าใดก็จะยิ่งทำให้ไมโครโพรเซสเซอร์สามารถส่งผ่านข้อมูลได้มากขึ้นเท่านั้นในแต่ละครั้ง
~~~~~~

หน่วยความจำภายในหมายถึง หน่วยความจำหลักของคอมพิวเตอร์เรียกว่าแรม (RAM) โดยแรมจะให้ซีพียูอ่าน/เขียนข้อมูลและคำสั่ง แรมถือเป็นปัจจัยหนึ่งที่มีส่วนสำคัญทำให้ระบบคอมพิวเตอร์โดยรวมทำงานได้อย่างมีประสิทธิภาพ

หน่วยความจำภายนอก หมายถึง หน่วยความจำสำรองซึ่งจัดเก็บแยกต่างหากจากแผงวงจรหลัก ได้แก่ จานแม่เหล็ก (Magnetic Disk) แถบแม่เหล็ก (Magnetic Tape) ซึ่งการทำงานของอุปกรณ์จัดเก็บข้อมูลเหล่านี้จะถูกควบคุมโดยหน่วยควบคุมที่เรียกว่า ไอโอ (I/O Controller) อีกทอดหนึ่ง

2) ความจุ

หน่วยความจำภายในหรือหน่วยความจำหลัก จะใช้หน่วยความความจุเป็นบิต หรือไบต์ โดยปกติข้อมูลขนาด 1 ไบต์ มีขนาด 8 บิต และอาจกำหนดขนาดของหน่วยวัดเป็นอย่างอื่นอีกได้เช่น เวิร์ด โดยข้อมูลขนาด 1 เวิร์ด อาจมีจำนวน 8, 16 หรือ 32 บิต ซึ่งในขณะที่หน่วยความจำภายนอกมักใช้หน่วยวัดความจุเป็น ไบต์

3) หน่วยของการขนย้ายข้อมูล

หน่วยการขนย้ายข้อมูลหรือรับส่งข้อมูลสำหรับหน่วยความจำหลักหมายถึง จำนวนบิตที่มีการอ่าน/เขียนข้อมูลแต่ละครั้ง การขนย้ายข้อมูลแต่ละครั้งสำหรับหน่วยความจำหลักจะเท่ากับจำนวนของเส้นทางข้อมูลที่เข้าออกจากหน่วยความจำหลักซึ่งโดยปกติจะมีขนาดเท่ากับเวิร์ด ดังนั้นการขนย้ายข้อมูลจึงมีหน่วยเป็นเวิร์ด แต่ในเครื่องคอมพิวเตอร์บางเครื่องก็อาจมีความแตกต่างกันออกไป ส่วนหน่วยความจำภายนอกมีการรับ/ส่งข้อมูลจำนวนมากครั้งละหลาย ๆ คำเรียกว่า บล็อก (Blocks) การกำหนดขนาดของการอ้างถึงตำแหน่งโดยทั่ว ๆ ไปจะกำหนดตำแหน่งในหน่วยความจำได้ในระดับของแต่ละไบต์เช่น หน่วยความจำหลักมีขนาดความจุสูงสุด $2^A = N$ เมื่อ A เป็นจำนวนบิตที่ใช้กำหนดตำแหน่งในหน่วยความจำหลัก

4) วิธีการเข้าถึงข้อมูล

การเข้าถึงข้อมูลในหน่วยความจำมีหลายวิธี ได้แก่

- การเข้าถึงตามลำดับ ได้แก่ แถบแม่เหล็กหรือเทปแม่เหล็ก เป็นหน่วยความจำภายนอกที่ใช้การเข้าถึงข้อมูลแบบตามลำดับโดยการเข้าถึงทีละระเบียน ไม่สามารถกระโดดข้ามระเบียนที่อยู่ก่อนหน้านั้นได้ เวลาที่ใช้ในการเข้ามีค่าไม่แน่นอน แปรผันตามขนาดและจำนวนระเบียนข้อมูลที่อยู่ก่อนหน้านั้น

- การเข้าถึงโดยตรง ได้แก่ จานแม่เหล็ก วิธีการเข้าถึงข้อมูลแบบนี้แต่ละระเบียนข้อมูลหรือบล็อกข้อมูลจะมีตำแหน่งที่ชัดเจนบนอุปกรณ์จัดเก็บข้อมูลที่มีการใช้กลไกหัวอ่าน/เขียนร่วมกัน การเข้าถึงครั้งแรกเป็นการเข้าถึงตำแหน่งเริ่มต้นของร่อง (แทร็ก) ที่จัดเก็บข้อมูลโดยตรง และหลังจากนั้นจะเป็นการค้นหาหรือนับตามลำดับในร่องไปเรื่อย ๆ จนกว่าจะถึงระเบียนหรือบล็อกข้อมูลต่อไป ดังนั้นเวลาที่ใช้ในการเข้าถึงจึงไม่แน่นอน

- การเข้าถึงแบบสุ่ม ได้แก่ หน่วยความจำหลัก มีการกำหนดตำแหน่งของแต่ละหน่วยข้อมูลที่ชัดเจนไม่มีซ้ำกันตามวงจรที่ออกแบบ เวลาในการเข้าถึงหน่วยข้อมูลจึงเป็นอิสระจากการเข้าถึงข้อมูลก่อนหน้านั้น โดยมีค่าที่คงที่เท่าเดิมทุกครั้งที่จะมีการเข้าถึงข้อมูล

- การเข้าถึงแบบเข้าร่วม ได้แก่ หน่วยความจำแคช วิธีการเข้าถึงข้อมูลแบบนี้คล้ายกับการเข้าถึงข้อมูลแบบสุ่ม แต่การเข้าถึงแบบนี้ไม่ได้ใช้ค่าตำแหน่งของข้อมูลเหมือนกับแบบสุ่ม แต่ใช้การเปรียบเทียบบิตข้อมูลในเวิร์ดแทน

5) ประสิทธิภาพ

ผู้ใช้มักมองหน่วยความจำในเชิงราคาและประสิทธิภาพมากกว่าอย่างอื่น ซึ่งในการพิจารณาประสิทธิภาพของหน่วยความจำโดยที่จริงแล้วจะพิจารณาจากปัจจัยดังนี้

- เวลาในการเข้าถึงข้อมูล เวลาที่ใช้ในการอ่าน/เขียนข้อมูลของการเข้าถึงข้อมูลแบบสุ่ม คือ เวลาที่ใช้หลังจากมีการกำหนดตำแหน่งข้อมูลเรียบร้อยแล้ว นั่นคือเวลาที่นำข้อมูลขึ้นมาหรือเวลาที่ใช้ในการบันทึกลงหน่วยความจำจริง ๆ ส่วนในกรณีของการเข้าถึงข้อมูลแบบอื่น ๆ จะหมายถึงเวลาที่กลไกหัวอ่าน/เขียนไปชี้ตำแหน่งของข้อมูลและเตรียมพร้อมที่จะเริ่มทำการอ่าน/เขียนข้อมูล

- วัฏจักรหน่วยความจำ ในกรณีของการเข้าถึงข้อมูลแบบสุ่มนั้น เวลาในการเข้าถึงข้อมูลหมายถึง เวลาของการเข้าถึงบวกกับเวลาของส่วนเพิ่มเติม ซึ่งอาจเป็นเวลาที่รอให้มีสัญญาณใหม่ของการทำงานครั้งถัดไป

- เวลาส่งถ่ายข้อมูล หรือเป็นเวลาของการรับส่งข้อมูล เป็นเวลาในเชิงของอัตราส่วนที่ข้อมูลถูกนำออกมาหรือนำเข้าไปบันทึกในหน่วยความจำจนเสร็จสิ้น

6) ชนิดของหน่วยความจำในเชิงกายภาพ ได้แก่ หน่วยความจำประเภทสารกึ่งตัวนำ ประเภทใช้แม่เหล็ก และประเภทใช้แสง

- ประเภทสารกึ่งตัวนำ หน่วยความจำประเภทนี้ใช้สารกึ่งตัวนำในการบันทึกข้อมูล เช่น SRAM DRAM และ Cache ส่วนเทคโนโลยีการผลิตใช้วงจรรวมแบบ LSI และเทคโนโลยี VLSI

- ประเภทใช้แม่เหล็ก หน่วยความจำประเภทนี้ใช้วัสดุเคลือบสารแม่เหล็กไว้ เช่น จานแม่เหล็ก แถบแม่เหล็ก ตลับแถบแม่เหล็ก

- ประเภทใช้แสง หน่วยความจำประเภทนี้ใช้การบันทึกข้อมูลในลักษณะที่สามารถอ่านได้ด้วยแสงเลเซอร์ เช่น CD-ROM DVD

7) คุณลักษณะเชิงกายภาพ แบ่งได้เป็น

- หน่วยความจำชั่วคราว เป็นหน่วยความจำประเภทที่หากขาดกระแสไฟฟ้าหล่อเลี้ยง ข้อมูลจะถูกลบเลือนไปไม่สามารถจดจำข้อมูลได้ เช่น หน่วยความจำหลัก หรือ แรม (RAM)

- หน่วยความจำแบบถาวร สามารถเก็บรักษาข้อมูลไว้ได้เป็นระยะเวลานานมาก เช่น หน่วยความจำประเภทที่ใช้สารแม่เหล็กเคลือบผิว ได้แก่ จานแม่เหล็ก แถบแม่เหล็ก

- นอกจากนี้ยังอาจแบ่งหน่วยความจำเป็นแบบ หน่วยความจำแบบลบได้ กับหน่วยความจำแบบลบไม่ได้ หน่วยความจำแบบลบได้สามารถเขียนบันทึกข้อมูลลงไปได้หลาย ๆ ครั้ง เช่น แรม จานแม่เหล็ก แถบแม่เหล็ก ส่วนหน่วยความจำแบบลบไม่ได้นั้นเมื่อบันทึกข้อมูลลงไปแล้ว สามารถอ่านขึ้นมาได้เพียงอย่างเดียวเท่านั้น เรียกหน่วยความจำประเภทนี้ว่า รอม (ROM)

6.2 หน่วยความจำหลัก

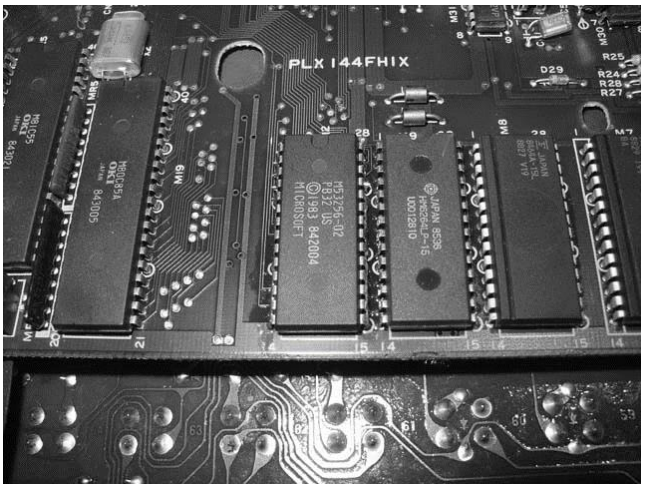
ในการทำงานของระบบคอมพิวเตอร์จำเป็นต้องพึ่งพาหน่วยความจำเป็นอย่างมาก แต่ก่อนอื่นต้องทำความรู้จักกับชนิดของหน่วยความจำหลักก่อน ซึ่งสามารถแบ่งประเภทของหน่วยความจำหลักตามลักษณะการเก็บข้อมูลได้ 2 ประเภท ดังนี้

6.2.1 หน่วยความจำชนิดรอม (Read Only Memory : ROM)

เรียกอีกอย่างหนึ่งว่า หน่วยความจำที่ข้อมูลไม่สูญหาย (Nonvolatile Memory) เป็นหน่วยความจำชนิดที่จะเก็บข้อมูลหรือโปรแกรมไว้อย่างถาวร ไม่สามารถเปลี่ยนแปลงอะไรได้ไม่ว่าจะต้องการหรือไม่ สิ่งที่เก็บไว้ประกอบด้วยข้อมูลที่จำเป็นสำหรับการเริ่มสตาร์ทเครื่องคอมพิวเตอร์ และใช้เก็บโปรแกรมไบออส (BIOS) หรือโปรแกรมของคอมพิวเตอร์ที่ฝังอยู่ในฮาร์ดแวร์ของเครื่องที่ทำหน้าที่ตรวจสอบฮาร์ดแวร์และอุปกรณ์ต่าง ๆ ของคอมพิวเตอร์ สามารถแบ่งออกได้อีกหลายชนิด ได้แก่

1) ROM (Read Only Memory)

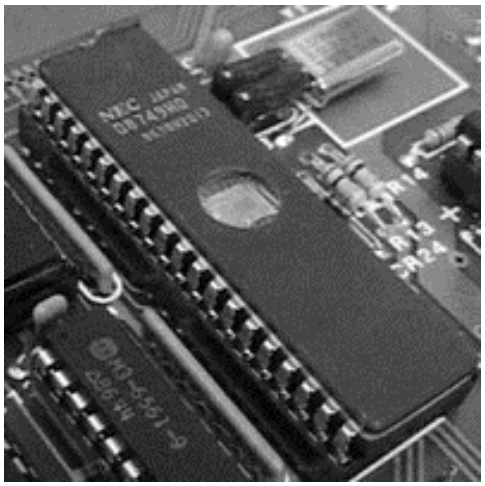
ข้อมูลทั้งหมดที่อยู่ในรอมจะถูกติดตั้งโปรแกรมมาจากโรงงานผู้ผลิต เราจะใช้รอมชนิดนี้เมื่อข้อมูลนั้นไม่มีการเปลี่ยนแปลงและมีความต้องการใช้งานเป็นจำนวนมาก ผู้ใช้ไม่สามารถเปลี่ยนแปลงข้อมูลภายในรอมได้ ซึ่งรอมจะมีการใช้เทคโนโลยีที่แตกต่างกัน เช่น Bipolar, CMOS, NMOS, PMOS



ภาพที่ 6.1 แสดง ROM (Read Only Memory)

2) PROM (Programmable Read-only Memory)

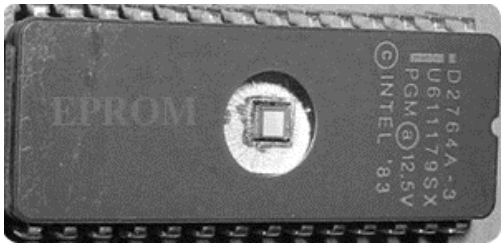
ข้อมูลที่ต้องการโปรแกรมจะถูกโปรแกรมโดยผู้ใช้เอง โดยป้อนพัลส์แรงดันสูง (High Voltage Pulse) ทำให้ Metal Stripe หรือ Poly crystalline silicon ที่อยู่ในตัว IC ขาดออกจากกัน ทำให้เกิดเป็นลอจิก 1 หรือ 0 ตามตำแหน่งที่กำหนดในหน่วยความจำนั้น ๆ เมื่อ PROM ถูกโปรแกรมแล้วข้อมูลภายในจะไม่สามารถเปลี่ยนแปลงได้อีก หน่วยความจำชนิดนี้จะใช้ในงานที่ใช้ความเร็วสูง เนื่องจากมีความเร็วสูงกว่าหน่วยความจำที่โปรแกรมได้ชนิดอื่น



ภาพที่ 6.2 แสดง PROM (Programmable Read-only Memory)

3) EPROM (Erasable Programmable Read-only Memory)

ข้อมูลจะถูกโปรแกรมโดยผู้ใช้โดยการให้สัญญาณที่มีแรงดันสูง (High Voltage Signal) ผ่านเข้าไปในตัว EPROM ซึ่งเป็นวิธีเดียวกับที่ใช้ใน PROM แต่ข้อมูลที่อยู่ใน EPROM สามารถเปลี่ยนแปลงได้ โดยการลบข้อมูลเดิมที่อยู่ใน EPROM ออกก่อนแล้วค่อยโปรแกรมเข้าไปใหม่ การลบข้อมูลนั้นทำได้โดยการฉายแสงอัลตราไวโอเลตเข้าไปในตัว IC โดยผ่านทางกระจกใสที่อยู่บนตัว IC เมื่อฉายแสงประมาณ 5-10 นาที ข้อมูลที่อยู่ภายในก็จะถูกลบทิ้ง ซึ่งช่วงเวลาที่ฉายแสงนี้สามารถดูได้จากข้อมูลที่กำหนด (Data sheet) มากับตัว EPROM และ EPROM มีความเหมาะสมที่จะใช้สำหรับงานของระบบที่ต้องปรับปรุงแก้ไขข้อมูลใหม่



ภาพที่ 6.3 แสดง EPROM (Erasable Programmable Read-only Memory)

4) EAROM (Electrically Alterable Read-only Memory)

EAROM หรือ EEPROM (Electrical Erasable EPROM) เป็นรอมชนิดที่มีการใช้ไฟฟ้าในการลบข้อมูลในรอมเพื่อเขียนใหม่ ซึ่งใช้เวลาน้อยกว่า EPROM

การลบข้อมูลขึ้นอยู่กับพื้นฐานของการใช้เทคโนโลยีที่แตกต่างกัน ซึ่ง EAROM ใช้พื้นฐานของเทคโนโลยีแบบ NMOS ซึ่งข้อมูลจะถูกโปรแกรมโดยผู้ใช้เหมือนใน EPROM แต่สิ่งที่แตกต่างก็คือ ข้อมูลของ EAROM สามารถลบได้โดยใช้ไฟฟ้าไม่ใช่โดยการฉายแสงแบบ EPROM

แต่โดยทั่วไปจะใช้ EPROM เพราะเราสามารถหามาใช้และทดลองได้ง่าย มีราคาถูก วงจรต่อง่ายไม่ยุ่งยาก และสามารถเปลี่ยนแปลงโปรแกรมได้ นอกจากระบบที่ทำการค้าเป็นจำนวนมากจึงจะใช้รอมประเภทโปรแกรมสำเร็จ

ภาพที่ 6.4 แสดง EAROM (Electrically Alterable Read-only Memory)

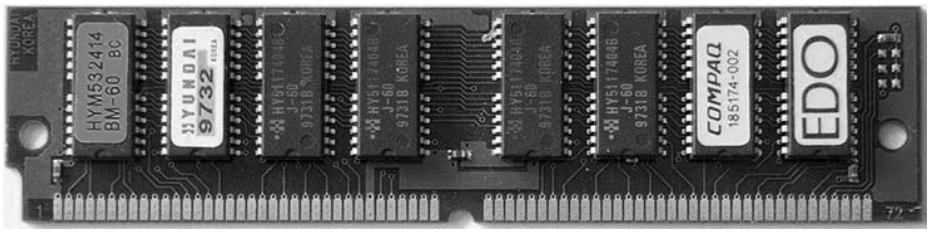
6.2.2 หน่วยความจำแรม (Random Access Memory : RAM)

เรียกอีกอย่างหนึ่งว่า หน่วยความจำที่ข้อมูลสูญหายได้ (Volatile Memory) เป็นหน่วยความจำชนิดที่ข้อมูลจะถูกลบหรือหายไปเมื่อคอมพิวเตอร์ไม่มีกระแสไฟเลี้ยง เมื่อใดก็ตามที่ไม่มีกระแสไฟมาหล่อเลี้ยงข้อมูลที่อยู่ในหน่วยความจำชนิดนี้จะสูญหายไปทันที หน่วยความจำแรมทำหน้าที่เก็บชุดคำสั่งและข้อมูลที่ระบบคอมพิวเตอร์กำลังทำงานอยู่ด้วย ไม่ว่าจะเป็นการนำเข้าข้อมูล (Input) หรือการส่งข้อมูลออก (Output)

โดยปกติหน่วยความจำหลักของระบบคอมพิวเตอร์เป็นหน่วยความจำแบบแรมซึ่งสามารถเขียนข้อมูลได้ตลอดเวลา สามารถแบ่งแรมออกเป็นชนิดต่าง ๆ ดังนี้

1) DRAM (Dynamic Random Access Memory)

DRAM จะเก็บข้อมูลในตัวเก็บประจุ (Capacitor) ซึ่งจำเป็นต้องมีการรีเฟรชเพื่อเก็บข้อมูลให้คงอยู่ โดยการรีเฟรชนี้ทำให้เกิดการหน่วงเวลาในการเข้าถึงข้อมูล และเนื่องจากที่มันต้องรีเฟรชตัวเองอยู่ตลอดเวลานั้นทำให้ได้ชื่อว่า Dynamic RAM

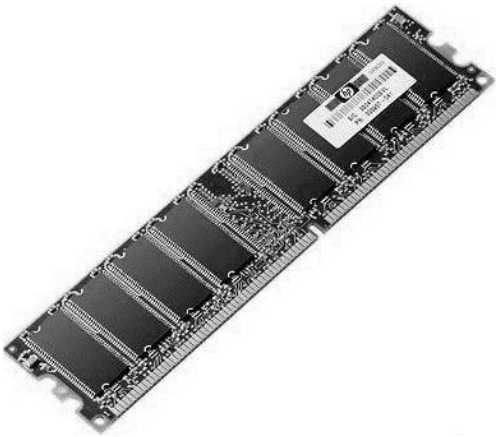


ภาพที่ 6.5 แสดง DRAM (Dynamic Random Access Memory)

2) SRAM (Static Random Access Memory)

จะแตกต่างจาก DRAM ตรงที่ DRAM ต้องทำการรีเฟรชข้อมูลอยู่ตลอดเวลา ส่วน SRAM จะเก็บข้อมูลนั้น ๆ ไว้และจะไม่ทำการรีเฟรชโดยอัตโนมัติ โดยจะทำการรีเฟรชก็ต่อเมื่อผู้ใช้สั่งรีเฟรชเท่านั้น ซึ่งข้อดีของ SRAM คือความเร็วที่เร็วกว่า DRAM ปกติมาก แต่ราคาสูงกว่ามากจึงเป็นข้อด้อยของ SRAM เป็นเหตุผลให้ผู้ผลิตไม่นำ SRAM มาใช้เป็นหน่วยความจำมาตรฐานของเครื่องคอมพิวเตอร์พีซี

การใช้งานส่วนใหญ่ของหน่วยความจำประเภทนี้จะถูกจำกัดไว้เฉพาะการเป็นหน่วยความจำแคช (Cache) ซึ่งเป็นหน่วยความจำที่มีขนาดเล็กมากเมื่อเทียบกับหน่วยความจำทั้งหมดที่ติดตั้งอยู่ในเครื่องคอมพิวเตอร์



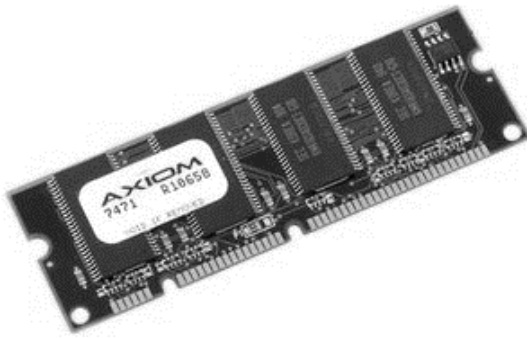
ภาพที่ 6.6 แสดง SRAM (Static Random Access Memory)

หน่วยความจำแรมนั้นได้รับการพัฒนาอย่างต่อเนื่อง ทั้งนี้เพื่อให้ทำงานทันกับความเร็วของซีพียู สามารถจำแนกประเภทของแรมที่ใช้กันตั้งแต่อดีตจนถึงปัจจุบันได้อีกดังนี้

1) FPM DRAM (Fast Page Mode DRAM)

เป็นแรมรุ่นเก่าแก่ที่ใช้เทคโนโลยีที่ถูกผลิตขึ้นมาหลายปีแล้ว พบได้ในเครื่องรุ่น 286 มีโมดูลแบบ SIMM ขนาด 30ขา และ 72ขา ซึ่งใช้กับเครื่องรุ่นเก่า ในระยะแรกแรมแบบโมดูล FPM ออกมามีขนาด 2 4 8 16 และ 32MB ความเร็ว 50 และ 70ns (Nano Second) การใช้แรมแบบนี้ต้องใช้แรมที่มีความเร็วเดียวกันเท่านั้น ไม่สามารถใช้แรมที่มีความเร็วต่างกันบนเมนบอร์ดเดียวกันได้

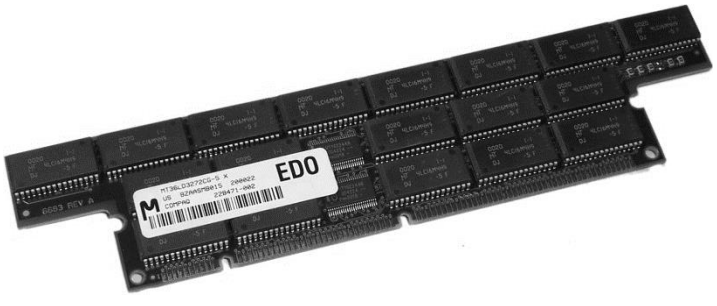
FPM นั้นก็เหมือนกับ DRAM เพียงแต่ว่ามันลดช่วงเวลาการหน่วงเวลาขณะเข้าถึงข้อมูลลง ทำให้มันมีความเร็วในการเข้าถึงข้อมูลเร็วกว่า DRAM ปกติ โดยที่สัญญาณนาฬิกาในการเข้าถึงข้อมูลเป็น 6-3-3-3 (Latency เริ่มต้นที่ 3 clock พร้อมด้วย 3 clock สำหรับการเข้าถึง page) และสำหรับระบบแบบ 32bit จะมีอัตราการส่งถ่ายข้อมูลสูงสุด 100MB ต่อวินาที ส่วนระบบแบบ 64bit จะมีอัตราการส่งถ่ายข้อมูลที่ 200MB ต่อวินาที ปัจจุบันนี้แรมชนิดนี้แทบจะหมดไปจากท้องตลาดแล้วแต่ยังอาจมีให้เห็นบ้างในบางพื้นที่



ภาพที่ 6.7 แสดง FPM DRAM (Fast Page Mode DRAM)

2) EDO RAM (Extended Data Output RAM)

ใช้ในเครื่องตั้งแต่รุ่น Pentium ขึ้นไป เป็นแรมที่มีความเร็วสูงกว่าแรมชนิด FPM โดยมีความเร็วอยู่ระหว่าง 70-50ns (ตัวเลขยิ่งน้อยยิ่งเร็ว) EDO RAM ทำงานได้ดีที่ความเร็ว 66 MHz ขนาด 32bit มีขาอยู่ 72ขา มีโมดูลแบบ SIMM นอกจากนี้ EDO RAM ถูกออกแบบให้เป็นแบบ ECC (Error Correcting Code) ซึ่งมีคุณสมบัติที่สามารถตรวจสอบข้อผิดพลาดในการทำงานของแรมได้ แต่เมื่อความเร็วในการทำงานของเครื่องคอมพิวเตอร์สูงขึ้นโดยเฉพาะเมื่อความเร็วของบัสระบบอยู่ที่ระดับสูงกว่า 66 MHz ขึ้นไป หน่วยความจำแบบ EDO จึงอาจกลายเป็นตัวหน่วงให้ความเร็วในการทำงานของคอมพิวเตอร์โดยรวมต่ำกว่าที่ควรจะเป็น เนื่องจากซีพียูต้องเสียเวลาในการรอข้อมูลจากหน่วยความจำ

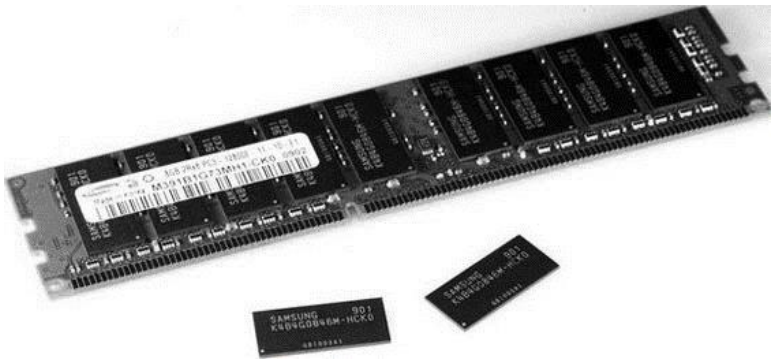


ภาพที่ 6.8 แสดง EDO RAM (Extended Data Output RAM)

3) BEDO DRAM (Burst EDO RAM)

BEDO ได้เพิ่มความสามารถขึ้นมาจาก EDO RAM เดิมคือ Burst Mode โดยหลังจากที่มันได้แอดเดรสแรกที่ต้องการแล้วมันจะทำการ generate อีก 3 แอดเดรสขึ้นมาทันทีภายใน 1 สัญญาณนาฬิกา

ดังนั้นจึงตัดช่วงเวลาในการรับแอดเดรสต่อไป เพราะฉะนั้น Timing ของมันจึงเป็น 5-1-1-1 ที่ 66 MHz แต่ BEDO ไม่เป็นที่แพร่หลายและได้รับความนิยมเพียงระยะเวลาสั้น ๆ เนื่องจากทาง Intel ตัดสินใจใช้ SRAM แทน และไม่ได้ใช้ BEDO เป็นส่วนประกอบในการพัฒนาชิปเซตของตน ทำให้บริษัทผู้ผลิตต่าง ๆ หันมาพัฒนา SRAM แทน

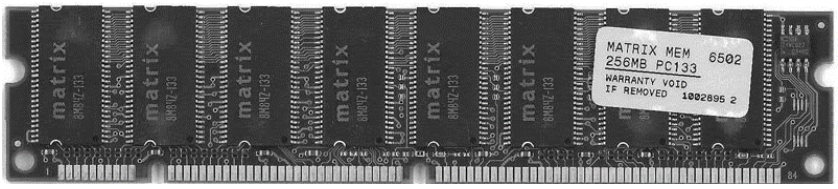


ภาพที่ 6.9 แสดง BEDO DRAM (Burst EDO RAM)

4) SDRAM (Synchronous DRAM)

จะแตกต่างจาก DRAM เดิมตรงที่มันจะทำงานสอดคล้องกับสัญญาณนาฬิกา สำหรับ DRAM เดิมจะทราบตำแหน่งที่อ่านก็ต่อเมื่อเกิดทั้ง RAS และ CAS ขึ้น แล้วจึงไปอ่านข้อมูลโดยมีช่วงเวลาในการเข้าถึงข้อมูลตามที่เรามักจะเห็นบนตัวชิปของตัวแรมเลย เช่น -50 -60 -80 โดย -50 หมายถึง ช่วงเวลาในการเข้าถึงใช้เวลา 50 นาโนวินาที แต่ SDRAM จะใช้สัญญาณนาฬิกาเป็นตัวกำหนดการทำงานโดยจะใช้ความถี่ของสัญญาณเป็นตัวระบุ

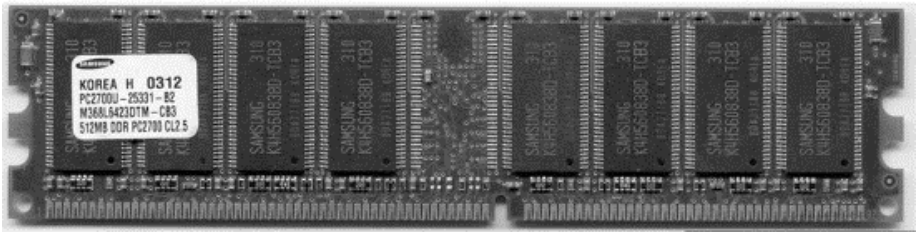
SDRAM จะทำงานตามสัญญาณนาฬิกาขาขึ้นเพื่อรองรับตำแหน่งข้อมูลที่ต้องการให้มันอ่าน หลังจากนั้นมันก็จะไปค้นหาให้และให้ผลลัพธ์ออกมาหลังจากได้รับตำแหน่งแล้วเท่ากับค่าของ CAS เช่น CAS2 คือ หลังจากรับตำแหน่งที่อ่านแล้วมันจะให้ผลลัพธ์ออกมาภายใน 2 ลูกของสัญญาณนาฬิกา ซึ่ง SDRAM จะมี Timing เป็น 5-1-1-1 ซึ่งมีความเร็วพอกันกับ BEDO RAM แต่ว่ามันสามารถทำงานได้ที่ 100 MHz หรือมากกว่า และมีอัตราการส่งถ่ายข้อมูลสูงสุดที่ 528MB ต่อวินาที



ภาพที่ 6.10 แสดง SDRAM (Synchronous DRAM)

5) DDR SDRAM (SDRAM II)

DDR SDRAM เป็นแรมที่แยกมาจาก SDRAM โดยจุดที่ต่างกันหลัก ๆ ของแรมทั้งสองชนิดนี้ คือ DDR SDRAM สามารถใช้งานได้ทั้งขาขึ้นและขาลงของสัญญาณนาฬิกาเพื่อส่งถ่ายข้อมูล ทำให้อัตราการส่งถ่ายเพิ่มขึ้นได้ถึงหนึ่งเท่าตัว โดยอัตราการส่งถ่ายข้อมูลสูงสุดอยู่ที่ 1GB ต่อวินาที



ภาพที่ 6.11 แสดง DDR SDRAM (SDRAM II)

6) RDRAM (Rambus DRAM)

ชื่อของ Rambus เป็นเครื่องหมายการค้าของบริษัท RAMBUS Inc. ซึ่งก่อตั้งมาตั้งแต่ยุค 80 แล้ว เพราะฉะนั้นชื่อนี้ก็ไม่ได้เป็นชื่อที่ใหม่อะไรนัก โดยปัจจุบันได้เอาหลักการของ Rambus มาพัฒนาใหม่โดยการลด pin รวม static buffer และทำการปรับแต่งทางอินเตอร์เฟสใหม่

RDRAM ชนิดนี้จะสามารถทำงานได้ทั้งขอบขาขึ้นและขาลงของสัญญาณนาฬิกา มี performance มากกว่าเป็นสามเท่าจาก SDRAM และมีเพียงแค่ช่องสัญญาณเดียวก็มีอัตราการส่งถ่ายข้อมูลสูงถึง 1.6GB ต่อวินาที ถึงแม้ว่าเวลาในการเข้าถึงข้อมูลแบบสุ่มของแรมชนิดนี้จะช้า แต่การเข้าถึงข้อมูลแบบต่อเนื่องจะเร็วมาก

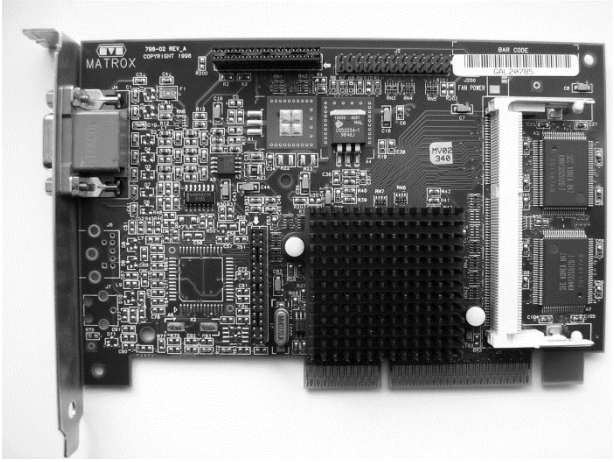
RDRAM มีการพัฒนาอินเตอร์เฟส และมี PCB (Printed Circuit Board) ที่ดี ๆ และรวมถึงคอนโทรลเลอร์ของอินเตอร์เฟสให้สามารถใช้งานได้ถึง 2 ช่องสัญญาณ และมีอัตราการส่งถ่ายข้อมูลเพิ่มเป็น 3.2GB ต่อวินาที และถ้าหากใช้ได้ถึง 4 ช่องสัญญาณก็มีอัตราการส่งถ่ายข้อมูลเพิ่มขึ้นถึง 6.4GB ต่อวินาที

ภาพที่ 6.12 แสดง RDRAM (Rambus DRAM)

7) SGRAM (Synchronous Graphic RAM)

SGRAM นี้ก็แยกออกมาจาก SDRAM เช่นเดียวกันโดยมันถูกปรับแต่งมาสำหรับงานด้านกราฟิกเป็นพิเศษ แต่โดยโครงสร้างของฮาร์ดแวร์แล้วแทบไม่มีอะไรต่างจาก SDRAM เลย ใน graphic card ที่เป็นรุ่นเดียวกันบ้างก็ใช้ SDRAM บ้างก็ใช้ SGRAM เช่น Matrox G200 แต่จุดที่แตกต่างกันคือ ฟังก์ชันที่ใช้โดย page register ซึ่ง SGRAM สามารถเขียนข้อมูลได้หลาย ๆ ตำแหน่งในหนึ่งสัญญาณนาฬิกา ทำให้ความเร็วในการแสดงผลและ clear screen ทำได้เร็วมาก และยังสามารถเขียนแค่บาง bit ในการแก้ไข word ได้โดยไม่ต้องเขียนข้อมูลใหม่ทั้งหมด โดยการใช้ bit mask ในการเลือก bit ที่จะเขียนใหม่

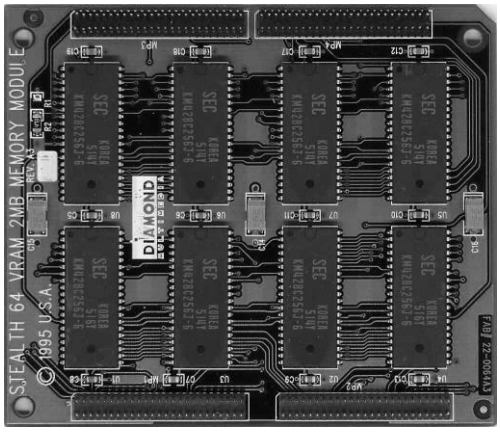
SGRAM เหมาะกับงานด้านกราฟิกมากกว่างานปกติทั่ว ๆ ไป เพราะสามารถแสดงผลได้เร็วและ clear screen ได้เร็ว จึงเหมาะกับใช้บน graphic card มากกว่าใช้บน system



ภาพที่ 6.13 แสดง SGRAM (Matrox G200)

8) Video Ram (VRAM)

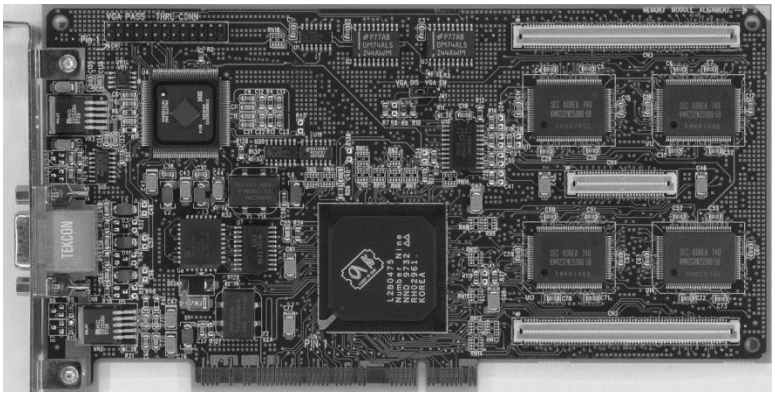
ได้รับการออกแบบให้ใช้บน Display card ซึ่ง VRAM พัฒนามาจาก DRAM แต่แตกต่างกันด้วยกลไกการทำงานบางอย่างที่เพิ่มขึ้นมาคือ VRAM มี serial port พิเศษเพิ่มขึ้นอีก 1-2 port ทำให้ผู้ใช้มองว่าเป็นแรมแบบพอร์ทคู่ (Dual port หรือ Triple port) นอกจากนี้ในส่วนของ parallel port ซึ่งเป็น standard interface จะถูกใช้ในการติดต่อกับ host processor เพื่อสั่งให้ทำการรีเฟรชภาพขึ้นมาใหม่และ serial port จะส่งข้อมูลภาพออกสู่จอภาพ



ภาพที่ 6.14 แสดง VRAM

9) Windows RAM (WRAM)

เป็นแรมที่ใช้บน graphic card ตระกูล Millennium, Millennium II และในรุ่น Number 9 Revolution IV ที่ถูกพัฒนาโดย Matrox ซึ่งคุณสมบัติโดยรวมของ WRAM ก็เหมือนกับ VRAM แต่แตกต่างกันตรงที่ WRAM สามารถรองรับ bandwidth ที่สูงกว่าและใช้ระบบ double buffer ทำให้สามารถทำงานได้เร็วกว่า VRAM



ภาพที่ 6.15 แสดง WRAM (รุ่น Number 9)

6.3 หน่วยความจำสำรอง

เป็นหน่วยความจำสำรอง หรือสื่อเก็บข้อมูลประเภทต่าง ๆ ซึ่งมีพื้นที่ในการจัดเก็บโปรแกรมและข้อมูลของคอมพิวเตอร์ เช่น แผ่นดิสก์ ฮาร์ดดิสก์ ซีดีรอม หรือ ROM/BIOS เป็นต้น ซึ่งจัดเป็นหน่วยความจำที่ใช้เก็บข้อมูลแบบถาวร (Permanent storage areas)

6.3.1 ฟลอปปี้ดิสก์ (Floppy disk or diskette)

เป็นแผ่นพลาสติกที่ทำจากวัสดุไมล่าบาง ๆ เคลือบด้วยสารอ๊อกไซด์ที่สามารถทำให้มีสภาพเป็นเหล็กได้ บรรจุอยู่ในซองที่ทำหน้าที่ในการป้องกันแผ่นพลาสติกนี้ มีการเก็บข้อมูลไว้บนผิวของแผ่นพลาสติก มักใช้ในการเก็บซอฟท์แวร์ การส่งไฟล์ หรือการเก็บสำรองข้อมูล ผิวของแผ่นดิสก์ฉาบด้วยสารแม่เหล็ก แบ่งเป็นแนววงกลมคล้ายร่องของแผ่นเสียง เรียกว่า แทร็ค (track) ซึ่งจะมีหลาย ๆ แทร็คซ้อนกันเป็นรูปวงแหวน แต่ละวงแหวนจะถูกแบ่งเป็นพื้นที่ย่อย ๆ เรียกว่า เซกเตอร์ (sector) ซึ่งข้อมูลจะถูกบันทึกลงบนเซกเตอร์เหล่านี้ จำนวนแทร็คและเซกเตอร์จะแตกต่างกันขึ้นอยู่กับวิธีการจัดเก็บข้อมูลหรือที่เรียกว่าการฟอร์แมต (format)

เราสามารถคำนวณความจุของแผ่นดิสก์ได้จากสูตร จำนวนด้าน * จำนวนแทร็ค * จำนวนเซกเตอร์ * จำนวนไบต์ในเซกเตอร์ เช่น แผ่นดิสก์ขนาด 3.5 นิ้ว ในแต่ละด้านจะแบ่งเป็น 40 แทร็ค แทร็คละ 36 เซกเตอร์ หนึ่งเซกเตอร์เก็บข้อมูลได้ 512 ไบต์ ดังนั้นแผ่นดิสก์นี้จะมีความจุ 2*4*36*512 = 1,474,560 ไบต์ หรือประมาณ 1.44 เมกะไบต์

จุดประสงค์ของการฟอร์แมต

1. เพื่อจัดเก็บพื้นที่สงวน (Reserve area) ซึ่งประกอบด้วย
   - Boot record หรือ Boot sector ซึ่งจะอยู่ที่เซกเตอร์ 0 ของดิสก์
   - FAT (File allocation table) คือ พื้นที่ที่ถัดจาก boot sector จะสงวนไว้ใช้สำหรับข้อมูลว่าไฟล์ถูกเก็บไว้ที่เซกเตอร์ใดบ้าง และไฟล์เหล่านั้นมีการเรียงลำดับกันอย่างไร
2. พื้นที่ที่เหลือสำหรับเก็บข้อมูล โดยจะแบ่งเป็นเซกเตอร์ที่เท่า ๆ กัน ซึ่งในการเก็บข้อมูลจะต้องเก็บอย่างน้อย 1 เซกเตอร์
3. กำหนดหมายเลขเซกเตอร์เป็นแบบสัมพันธ์กัน (Relative sector numbering)

6.3.2 ฮาร์ดดิสก์ (Hard Disk)

เป็นอุปกรณ์เก็บข้อมูลเก็บข้อมูลที่มีลักษณะการทำงานคล้ายกับฟลอปปี้ดิสก์ แต่สามารถบรรจุข้อมูลได้มากกว่าและมีความเร็วในการเข้าถึงข้อมูลสูงกว่า โครงสร้างภายในของฮาร์ดดิสก์ประกอบด้วย แผ่นอลูมิเนียมเคลือบด้วยสารแม่เหล็กหลายแผ่นเรียงซ้อนกันเป็นแผ่นดิสก์รวมเรียกว่า แพล็ตเตอร์ (Platters) ซึ่งแผ่นดิสก์เหล่านี้หมุนด้วยความเร็วประมาณ 3,600 รอบต่อวินาที ทำให้สามารถเก็บข้อมูลได้อย่างรวดเร็วและเก็บได้จำนวนมาก หัวอ่าน/เขียนของฮาร์ดดิสก์จะไม่สัมผัสกับแผ่นดิสก์โดยตรงเหมือนกับฟลอปปี้ดิสก์ แต่หัวอ่านจะลอยอยู่เหนือแผ่นดิสก์นั้นห่างประมาณ 4 ไมครอน

ฮาร์ดดิสก์คอนโทรลเลอร์ (Harddisk controller) เป็นอุปกรณ์ที่อยู่บนเมนบอร์ดของเครื่องคอมพิวเตอร์ ทำหน้าที่แปลงข้อมูลและควบคุมสัญญาณต่าง ๆ ให้เหมาะสมกับไดร์ฟของฮาร์ดดิสก์แต่ละชนิด และทำหน้าที่เชื่อมต่อวงจรภายในของคอมพิวเตอร์ให้เข้ากับวงจรของฮาร์ดดิสก์

ชนิดของคอนโทรลเลอร์

- ESDI (Enhanced Small Device Interface) มีอัตราการรับส่งข้อมูลอยู่ในช่วง 10-24 เมกะบิตต่อวินาที

- IDE (Intelligent Drive Electronics) เป็นฮาร์ดดิสก์ที่มีวงจรควบคุมอยู่ในตัวเอง และใช้คอนโทรลเลอร์การ์ด (Controller card) เป็นตัวเชื่อต่อกับระบบบัสของเมนบอร์ดแบบ IDE มีข้อดีคือ ความสามารถในการทำงานสูง มีความเร็วในการทำงานสูง โดยอัตราการส่งถ่ายข้อมูลมากกว่าชนิด ESDI ถึง 2 เท่า และมีราคาถูกกว่า

- SCSI (Small Computer Systems Interface) มีความเร็วในการรับส่งข้อมูลอยู่ระหว่าง 4-10 เมกะบิตต่อวินาที สามารถเชื่อมต่อกับอุปกรณ์ได้หลากหลายแบบและเชื่อมต่อได้ง่าย มีวงจรควบคุมอยู่ในตัวเองเช่นเดียวกับ IDE

ความจุของฮาร์ดดิสก์ในปัจจุบันใช้หน่วยวัดเป็นกิกะไบต์ (GB) ถึง เทราไบต์ (TB) ยิ่งฮาร์ดดิสก์ที่มีความจุมากก็สามารถเก็บข้อมูลได้มากและมีราคาสูงขึ้นตามไปด้วย ซึ่งเราสามารถคำนวณความจุของฮาร์ดดิสก์ได้จากสูตร

ความจุของฮาร์ดดิสก์ = จำนวนไซลินเดอร์ * จำนวนด้าน * จำนวนเซกเตอร์ * จำนวนไบต์ในเซกเตอร์

6.3.3 ซีดีรอม (CD-ROM : Compact Disk Read Only Memory)

เป็นอุปกรณ์ที่ใช้สำหรับเก็บข้อมูล ซึ่งโดยปกติผู้ใช้สามารถอ่านข้อมูลจากซีดีรอมได้อย่างเดียวเท่านั้น โดยแผ่นซีดีรอม 1 แผ่นสามารถทำการเขียนข้อมูลลงไปได้เพียงแค่ครั้งเดียวโดยใช้แสงเลเซอร์ความเข้มสูงทำให้ผิวหน้าของแผ่นเปลี่ยนสภาพไปถาวรตามข้อมูลที่เขียนลงไป แผ่นซีดีรอมที่ถูกเขียนข้อมูลลงไปแล้ว จะไม่สามารถเขียนข้อมูลทับลงไปได้อีกและไม่สามารถลบข้อมูลที่เขียนแล้วได้ การอ่านข้อมูลจากแผ่นซีดีรอมจะใช้แสงจากโฟโตไดโอด (Photo diode) วิ่งผ่านชั้นพลาสติกไปตามส่วนที่เป็นหลุมเป็นเนินของแผ่น เมื่อแสงกระทบกับส่วนที่เป็นหลุมแสงนั้นจะกระจายออกไปไม่มีการสะท้อนกลับมา โดยจะมีปริซึมเพื่อแยกแสงที่สะท้อนออกมาและส่งต่อไปยังหัวอ่านข้อมูล (Photo detector) เพื่อทำการตรวจจับแสง

รูปแบบการจัดเก็บข้อมูลของซีดีรอมเป็นลักษณะของแทร็คและเซกเตอร์เช่นเดียวกับดิสก์ แต่บนแผ่นซีดีรอมนั้นทุกเซกเตอร์จะมีขนาดเท่ากัน ดังนั้นจึงมีมอเตอร์ที่หมุนได้หลายความเร็วเพื่อให้อัตราในการอ่านข้อมูลที่สม่ำเสมอทุกเซกเตอร์

6.3.4 ซิปไดร์ฟ (Zip drive)

เป็นอุปกรณ์ที่สามารถเก็บข้อมูลที่มีขนาด 230 MB ลงบนแผ่นที่มีขนาด 3.5 นิ้ว ซึ่งมีทั้งแบบ internal และ external

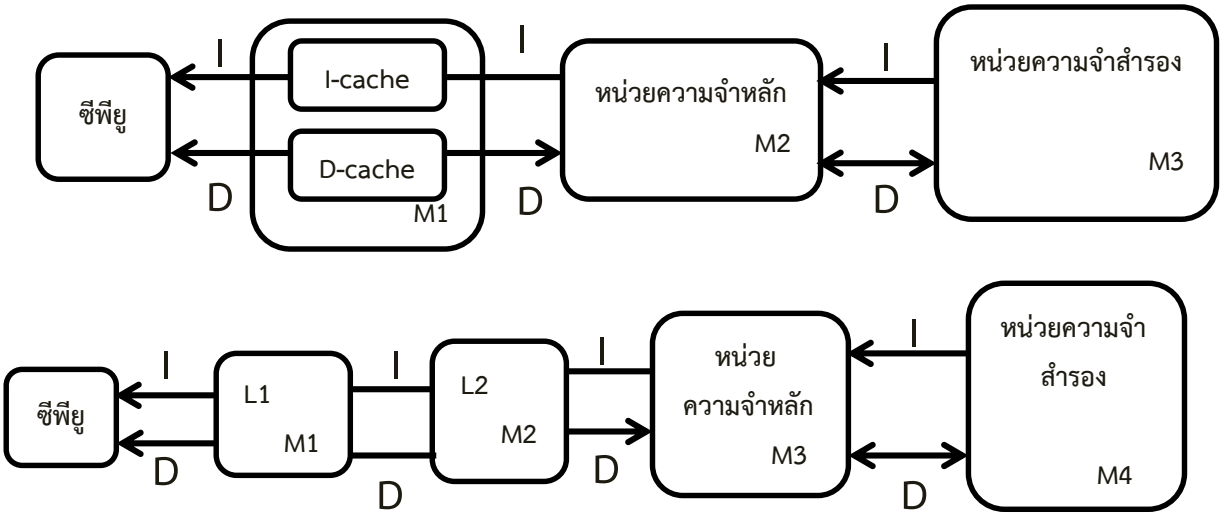
6.3.5 เทป (Tape)

เป็นอุปกรณ์การเก็บข้อมูลที่มีลักษณะการบันทึกข้อมูลแบบ sequential คือ ข้อมูลจะถูกเก็บเรียงกันไป ทำให้การเข้าถึงข้อมูลมีความล่าช้า เพราะหัวอ่านต้องเริ่มอ่านข้อมูลตั้งแต่ต้นเทปจนกว่าจะพบข้อมูลที่ต้องการ

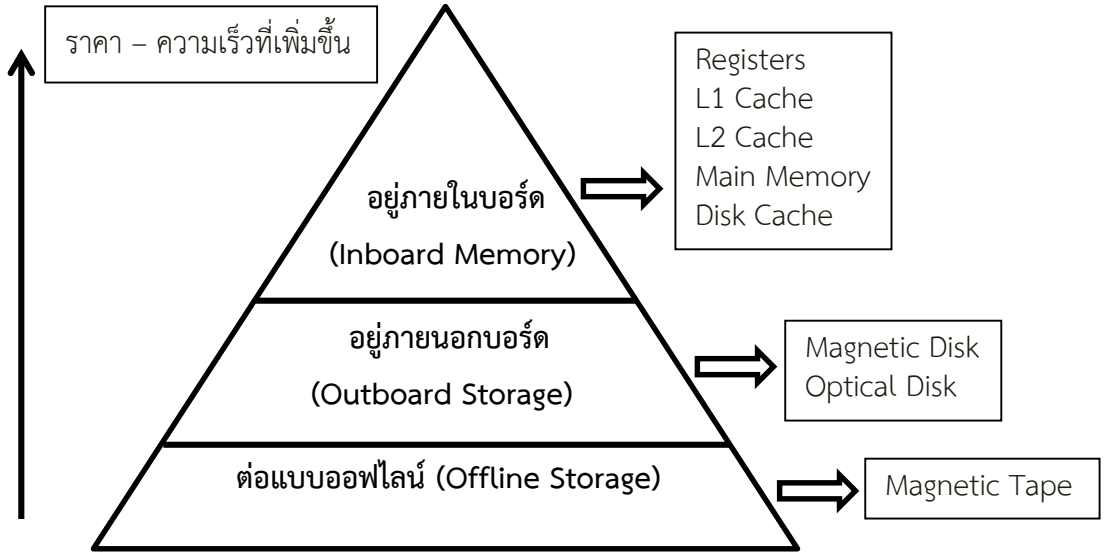
6.4 ลำดับชั้นของหน่วยความจำ

สามารถแบ่งลำดับชั้นของหน่วยความจำออกได้หลายระดับ ดังภาพ

ภาพที่ 6.16 แสดงลำดับชั้นของหน่วยความจำแบบ 2ระดับ คือ หน่วยความจำหลัก และหน่วยความจำสำรอง



ภาพที่ 6.17 แสดงลำดับชั้นของหน่วยความจำแบบ 3 ระดับ และ 4 ระดับ



ภาพที่ 6.18 แสดงรายละเอียดของชื่อที่ใช้เรียกหน่วยความจำแต่ละแบบ จัดกลุ่มตามที่ตั้งและลักษณะเฉพาะ

สถาปัตยกรรมของคอมพิวเตอร์โดยปกติแล้วจะจัดให้หน่วยความจำที่มีความเร็วสูงไว้ข้างบนใกล้ ๆ กับซีพียู โดยหน่วยความจำนี้ไม่จำเป็นต้องมีขนาดใหญ่มาก แต่ต้องมีความเร็วในการเข้าถึงข้อมูลสูง เนื่องจากใช้อัตราความเร็วในการเข้าถึงข้อมูลที่ความเร็วเดียวกับของซีพียู (clock rate) เมื่อมีการเรียกใช้คำสั่ง (working set) ชุดหนึ่ง คำสั่งเหล่านี้จะถูกนำไปไว้ในหน่วยความจำที่มีความเร็วสูงที่อยู่ติดกับซีพียู รวมถึงหน่วยความจำที่อยู่ใกล้กันถัดไปด้วย โดยอาศัยหลักการของ Locality of reference คือ ไม่ว่าโปรแกรมเมอร์จะเขียนโปรแกรมด้วยภาษาอะไรก็ตาม โปรแกรมเหล่านี้จะมีคุณสมบัติคือ เมื่อมีการเรียกใช้คำสั่งใดในโปรแกรมแล้วการเรียกคำสั่งนั้นซ้ำอีกจะเกิดขึ้นเสมอ ดังนั้นเมื่อมีการเรียกใช้คำสั่งนั้นซ้ำอีกครั้งซีพียูจะสามารถหาคำสั่งเหล่านี้ได้จากหน่วยความจำที่มีความเร็วสูงที่อยู่ใกล้กันนั่นเอง โดยไม่ต้องไม่ค้นหาที่หน่วยความจำหลักหรือแรมซึ่งมีความเร็วในการเข้าถึงข้อมูลที่ช้ากว่า

หน่วยความจำที่มีขนาดเล็กที่อยู่ใกล้กับซีพียูและมีความเร็วในการเข้าถึงข้อมูลสูงนี้เราเรียกว่าหน่วยความจำแคช (Cache memory) ซึ่งแคชที่อยู่ติดกับตัวซีพียูเราเรียกว่า แคช1 (cache level1 : L1) ส่วนแคชที่อยู่นอกซีพียูเรียกว่า แคช2 (cache level2 : L2)

ตามปกติแล้ว cache L1 จะมีราคาแพง มีขนาดเล็ก ส่วนแคชเลเวลถัดไปจะมีขนาดใหญ่ขึ้นเรื่อย ๆ หากซีพียูค้นหาข้อมูลใน cache L1 ไม่เจอก็จะไปค้นหาในเลเวลลำดับถัดไป แล้วจึงย้ายข้อมูล (transfer) ขึ้นไปตามลำดับขั้นของหน่วยความจำ การทรานสเฟอร์นั้นจะกระทำกับข้อมูลแบบเป็นชุดเรียกว่า บล็อก (block) และเนื่องจากหน่วยความจำในเลเวลแรก ๆ มีขนาดเล็กจึงต้องมีการสลับกันเข้า-ออกของบล็อกหรือข้อมูล ข้อมูลใดที่ถูกโหลดเข้ามาในแคชครั้งแรกและไม่มีการถูกเรียกใช้เลยก็จะถูก rolled out ออกไปจากแคชเพื่อให้แคชมีพื้นที่เหลือสำหรับเก็บข้อมูลและคำสั่งที่ซีพียูต้องการใช้

ปัจจัยที่มีผลทำให้หน่วยความจำมีประสิทธิภาพและทำงานได้รวดเร็ว

1. ความเป็น locality ของโปรแกรมเช่น โปรแกรมนั้นเขียนในลักษณะเป็น structure programming คือ มีการรีเคอร์ซีฟหรือมีการวนลูปการทำงาน ทำให้ใช้คำสั่งเดิม ๆ อยู่ตลอดเวลา จึงไม่ต้องมีการสลับเข้า-ออกของคำสั่งที่อยู่ในหน่วยความจำ
2. ความเร็วในการเข้าถึง (access time) ของหน่วยความจำแต่ละเลเวล
3. ขนาดความจุของหน่วยความจำในแต่ละเลเวล
4. ขนาดของบล็อกที่ใช้ทรานสเฟอร์ในแต่ละเลเวล
5. อัลกอริธึมหรือยุทธวิธีที่ใช้กำหนดตำแหน่ง หรือการแทนที่ในหน่วยความจำในแต่ละเลเวล

วิธีเพิ่มประสิทธิภาพในการทำงานของหน่วยความจำ

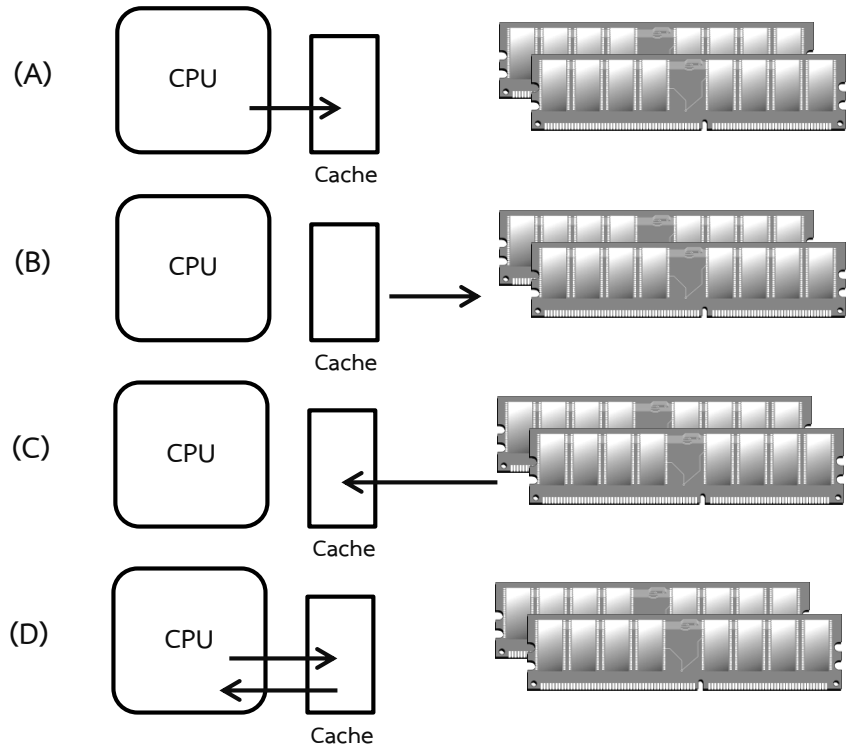
1. ลด access time ในหน่วยความจำ โดยการพัฒนาเทคโนโลยีของ SRAM และ DRAM ให้มี access time เร็วขึ้น
2. ทำหน่วยความจำให้เป็นแบบ interleaved โดยการแบ่งหน่วยความจำออกเป็นหลาย ๆ โมดูลที่สามารถเข้าถึงข้อมูลได้พร้อมกัน เป็นการทำงานแบบขนานซึ่งแต่ละโมดูลจะทำงานได้อย่างอิสระ ซึ่งจะทำให้ในหนึ่งรอบของการเข้าถึงข้อมูล (1 cycle access) สามารถทรานสเฟอร์ข้อมูลได้ทีละหลายเวิร์ด งานเหล่านี้เป็นงานที่ต้องใช้คอมพิวเตอร์ขนาดใหญ่ เช่น เมนเฟรม หรือ ซูเปอร์คอมพิวเตอร์ เพื่อให้สามารถให้บริการงานหลาย ๆ อย่างได้พร้อม ๆ กัน
3. Cache memory คือ การเพิ่มหน่วยความจำที่มีความเร็วสูงขึ้นมา ซึ่งอยู่ระหว่างซีพียูกับหน่วยความจำหลัก
4. Associative memory คือ การเพิ่มหน่วยความจำพิเศษ เช่น ฮาร์ดแวร์ ซึ่งช่วยในการค้นหาข้อมูลได้เร็วขึ้น จากเดิมที่มีการค้นหาข้อมูลทีละลำดับในแอดเดรสก็จะใช้ content เป็นคีย์ในการค้นหา โดยการค้นหาข้อมูลพร้อมกันทุกแอดเดรส

6.5 หน่วยความจำแคช

เนื่องจากเทคโนโลยีในการผลิตวงจรรวม (Integrated Circuit) มีการพัฒนาไปอย่างรวดเร็ว ทำให้ซีพียูที่ถูกผลิตออกมามีความเร็วสูงขึ้นเรื่อย ๆ จึงทำให้เกิดปัญหาด้านความเร็วที่แตกต่างกันระหว่างซีพียูกับแรม ซึ่งเป็นเหตุให้ซีพียูมีประสิทธิภาพในการทำงานที่ต่ำลง เนื่องจากไม่สามารถทำงานได้เต็มประสิทธิภาพของมัน ทำให้เกิดแนวความคิดในการสร้างแคช (Cache) ขึ้นไว้ภายในซีพียู ซึ่งแคชก็คือหน่วยความจำขนาดเล็กที่มีความเร็วและประสิทธิภาพในการทำงานสูง

หน้าที่ของแคชคือ จดจำคำสั่งและผลลัพธ์ที่ถูกเรียกใช้บ่อย ๆ เพื่อไว้ใช้ในการประมวลผลในครั้งต่อ ๆ ไป เพื่อที่ซีพียูไม่ต้องเสียเวลารอเรียกคำสั่งนั้น ๆ อีก ถ้าแคชมีขนาดใหญ่ก็สามารถจดจำคำสั่งได้มาก แต่แคช

นั้นไม่สามารถเก็บข้อมูลไว้ทั้งหมดได้เนื่องจากมีขนาดที่เล็ก ดังนั้นภายในแคชจึงเก็บเฉพาะข้อมูลและคำสั่งสำคัญ ๆ ที่ถูกเรียกใช้บ่อย ๆ เมื่อซีพียูต้องการข้อมูลนั้นก็สามารถเรียกใช้จากแคชได้ทันที แคชมีความเร็วในการทำงานสูงกว่าแรมมากและถูกสร้างให้อยู่ใกล้กับหน่วยประมวลผล จึงทำให้คอมพิวเตอร์สามารถประมวลผลได้เร็วขึ้น

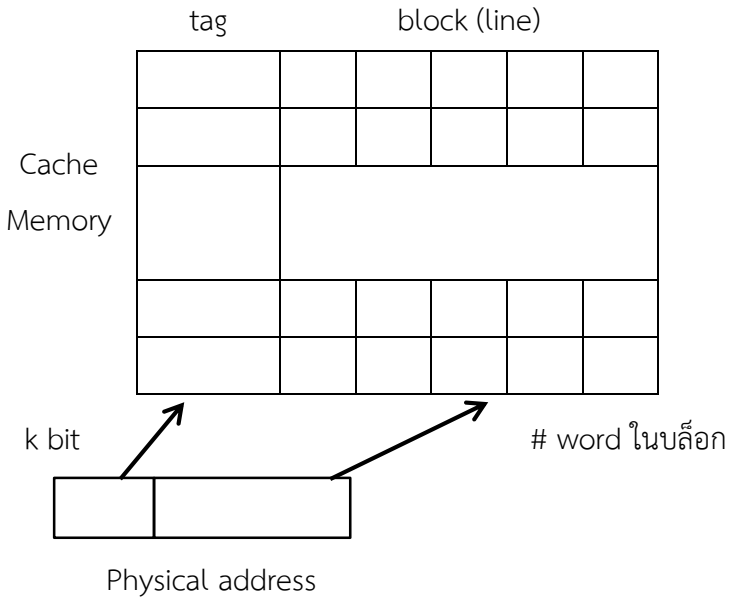


ภาพที่ 6.19 แสดงขั้นตอนการอ่านข้อมูลหรือชุดคำสั่งของซีพียู

จากภาพ 6.19 (A) ซีพียูต้องการอ่านข้อมูลหรือชุดคำสั่งจากหน่วยความจำ ซึ่งข้อมูลที่ซีพียูต้องการไม่ได้อยู่ที่แคช (B) เมื่อข้อมูลหรือชุดคำสั่งไม่ได้อยู่ที่แคช ซีพียูก็ต้องไปอ่านข้อมูลหรือชุดคำสั่งนั้นจากหน่วยความจำหลัก (RAM) (C) หน่วยความจำหลักส่งข้อมูลหรือชุดคำสั่งที่ซีพียูต้องการไปให้แคช (D) แคชส่งข้อมูลหรือชุดคำสั่งที่ได้รับจากหน่วยความจำหลักไปให้ซีพียู

แคชเป็นหน่วยความจำที่มีความเร็วสูง ติดตั้งอยู่ระหว่างซีพียูกับหน่วยความจำหลัก ตามปกติแล้วโปรแกรมที่เขียนตามหลักการ locality จะมีการเรียกใช้คำสั่งที่ซ้ำ ๆ กันบ่อย ๆ ดังนั้นถ้าเราเอาคำสั่งที่อยู่ในหน่วยความจำหลักมาไว้ที่หน่วยความจำที่อยู่ใกล้กับซีพียูและมีความเร็วในการเข้าถึงข้อมูลสูง ก็จะทำให้โปรแกรมนั้นสามารถทำงานได้เร็วขึ้น เพราะไม่ต้องไปค้นหาคำสั่งที่อยู่ในหน่วยความจำหลักทุกครั้งที่ต้องการ ซึ่งประสิทธิภาพของแคชจะขึ้นอยู่กับอัตราการค้นเจอข้อมูลที่อยู่ในแคช (Hit ratio) โดยปกติอยู่ระหว่าง 95-99%

หน่วยความจำแคชจะถูกแบ่งออกเป็นบล็อกหรือเป็นไลน์ แต่ละบล็อกจะมี tag ที่บอกถึงแอดเดรสเริ่มต้นของบล็อกที่อยู่ในหน่วยความจำหลัก โดยตามปกติแล้ว k bit แรกของแอดเดรสกายภาพ (physical address) จะถูกกำหนดให้เป็น tag ดังภาพ



ภาพที่ 6.20 แสดงลักษณะทางกายภาพของหน่วยความจำแคช

จากภาพ 6.20 k bit แรกของ physical address จะถูกเปรียบเทียบกับ tag ที่อยู่ในหน่วยความจำแคช ถ้าตรงกันหรือ match กัน เรียกว่า cache hit ส่วนที่เหลือของ physical address ก็จะไปเลือกข้อความที่ต้องการมาจาก line ของ tag นั้น แต่กรณีที่ไม่ match เรียกว่า cache miss ซึ่งบล็อกที่ถูกเรียกนั้นจะต้องถูกย้ายจากหน่วยความจำหลักเข้าไปไว้ในแคช ในขณะเดียวกันบล็อกนั้นก็จะถูกย้าย (transfer) ไปที่ซีพียูเพื่อ access ด้วย

6.5.1 หน่วยความจำที่ใช้กับแคช

แคชคือหน่วยความจำขนาดเล็ก ที่มีความเร็วในการทำงานสูง อยู่ใกล้กับซีพียู เก็บข้อมูลหรือคำสั่งที่ถูกซีพียูเรียกใช้หรือเรียกใช้บ่อย ๆ แคชจะทำงานได้เต็มประสิทธิภาพเมื่อใช้แรมแบบ SRAM (Static Ram) ซึ่งข้อมูลหรือคำสั่งถูกดึงไปใช้งานได้เร็วกว่าการดึงข้อมูลจากหน่วยความจำหลัก (Main Memory) ที่ใช้ DRAM (Dynamic Ram) หลายเท่าตัว

6.5.2 ระดับของแคช (Cache Level)

แคชมีหลายรูปแบบด้วยกัน สามารถแบ่งได้เป็น 3 ระดับดังนี้

1. แคชระดับที่ 1 (L1 Cache หรือ Cache level 1)

เป็นแคชที่สร้างลงบนชิพของซีพียู หรือเรียกอีกชื่อหนึ่งว่า internal cache แคชระดับที่ 1 จะทำงานที่ความเร็วเดียวกับซีพียู L1 Cache มีขนาดความจุไม่ใหญ่มากนัก เช่น สำหรับชิพ 486 มีขนาด 8 KB สำหรับ Pentium และ Pentium Pro มีขนาด 16 KB ส่วน Pentium MMX และ Pentium II มีขนาด 32 KB แต่ในปัจจุบันสำหรับซีพียูรุ่นใหม่ ๆ ความจุของแคชก็มีขนาดใหญ่ขึ้นตามไปด้วย

2. แคชระดับที่ 2 (L2 Cache หรือ Cache level 2)

แคชระดับนี้มีอยู่ 2 แบบคือ แบบที่อยู่บนซีพียู (Internal Cache) และแบบที่อยู่ภายนอกซีพียู (External Cache)

แคชแบบที่อยู่ภายนอกซีพียูจะแบ่งเป็นอีก 2 แบบคือ แบบที่ติดตั้งอยู่บนแผ่นวงจรเดียวกับซีพียู เช่น แบบที่อยู่บนซีพียู Pentium กับ Pentium III รุ่นแรก ๆ (ลักษณะเป็นการ์ดใส่อยู่ในตลับ) ซึ่งเรียกว่า “แคชระดับที่ 2” ส่วนที่ติดตั้งอยู่บนเมนบอร์ดเรียกว่า “แคชระดับที่ 3”

3. แคชระดับที่ 3 (L3 Cache หรือ Cache level 3)

เป็นแคชที่ออกแบบมาเฉพาะซีพียูของค่าย AMD รุ่น K6-III ซึ่งเป็นแคชแบบที่ติดตั้งมาบนเมนบอร์ด เนื่องจากซีพียูรุ่น K6-III มี L2 Cache ติดตั้งอยู่บนซีพียูแล้ว และเมนบอร์ดที่ใช้กับ K6-III ก็เป็นเมนบอร์ดรุ่นที่มีการใส่แคชมาให้แล้วบนเมนบอร์ด

6.5.3 การทำงานของหน่วยความจำแคช

1. ขั้นตอนการอ่าน (Read)

Cache Hit	พบข้อมูลที่ต้องการใช้งานในแคช
Cache Miss	ไม่พบข้อมูลที่ต้องการใช้งานในแคช

ดังนั้น

Cache Hit	ส่งข้อมูลไปยังโพรเซสเซอร์
Cache Miss	แคชจะถามหาข้อมูลจากหน่วยความจำระดับถัดไป จากนั้นให้ทำการคัดลอกข้อมูลบางส่วนที่ต้องการออกมาและส่งต่อไปยังหน่วยความจำระดับถัดขึ้นมาเพื่อเตรียมส่งต่อให้โพรเซสเซอร์

2. ขั้นตอนการบันทึก (Write)

Cache Hit	พบข้อมูลที่ต้องการบันทึกหรืออัพเดตในแคช
Cache Miss	ไม่พบข้อมูลที่ต้องการบันทึกหรืออัพเดตในแคช

ดังนั้น

Cache Hit	บันทึกหรือเขียนข้อมูลลงในแคช
Cache Miss	แคชจะถามหาข้อมูลจากหน่วยความจำระดับถัดไป จากนั้นให้ทำการคัดลอกข้อมูลบางส่วนที่ต้องการออกมาแล้วบันทึกลงในแคช

โดยการบันทึกอาจใช้วิธี

Write-Through เป็นการบันทึกโดยพิจารณาว่าหากข้อมูลหรือคำสั่งใดที่ถูกเปลี่ยนแปลงค่า จะต้องทำการเปลี่ยนแปลงค่าที่หน่วยความจำหลักทุกครั้ง

Write-Back เป็นการบันทึกข้อมูลหรือคำสั่งที่ต้องการเปลี่ยนแปลงค่าในแคชก่อน ต่อมาเมื่อข้อมูลหรือคำสั่งนั้นถูกนำออกจากแคช จึงค่อยบันทึกข้อมูลหรือคำสั่งนั้นลงในหน่วยความจำหลัก

การบันทึกข้อมูลหรือคำสั่งด้วยวิธี Write-Through ทำให้เกิดการหน่วงเวลาขึ้นเนื่องจากต้องรอคอยการบันทึกข้อมูลหรือคำสั่งลงในหน่วยความจำหลักด้วย ซึ่งจะเกิดความล่าช้าและจะล่าช้ามากขึ้นหากมีการบันทึกข้อมูลหรือคำสั่งอยู่บ่อยครั้ง

ส่วนการบันทึกข้อมูลหรือคำสั่งด้วยวิธี Write-Back นั้นจะบันทึกข้อมูลหรือคำสั่งที่มีการเปลี่ยนแปลงเฉพาะในหน่วยความจำแคชก่อน ต่อมาเมื่อข้อมูลหรือคำสั่งนั้น ๆ ถูกขับออกจากแคช จึงค่อยนำข้อมูลหรือคำสั่งเหล่านั้นบันทึกลงในหน่วยความจำหลักต่อไป ซึ่งมีความรวดเร็วกว่าการบันทึกข้อมูลหรือคำสั่งด้วยวิธี Write-Through

6.5.4 วิธีการอัพเดตหน่วยความจำหลัก

1. Copy back / Write back

เมื่อมีการอัพเดตข้อมูลจากซีพียูจะอัพเดตเฉพาะในแคชเท่านั้นจนกระทั่งมีการ replace time เกิดขึ้นคือ เมื่อมีการ load line ใหม่จากหน่วยความจำหลักมาทับใน line เดิมในแคชจึงจะ write line ที่ถูก copy นั้นกลับไปที่หน่วยความจำหลักก่อนแล้วจึงทำการ replace line

วิธีนี้มีข้อดีคือ ใช้เวลาในการเข้าถึงข้อมูลน้อย ไม่ต้องมีการอัพเดตในหน่วยความจำหลักทุกครั้งที่มีการแก้ไขข้อมูล

2. Write through

มีการอัพเดตข้อมูลในหน่วยความจำหลักทุกครั้งที่มีการอัพเดตในแคช ทำให้มีการเข้าถึงหน่วยความจำบ่อยครั้ง แต่ข้อมูลในแคชและหน่วยความจำหลักตรงกัน

วิธีนี้มีปัญหาคือ กรณีที่เป็นระบบมัลติโพรเซสเซอร์ เนื่องจากโพรเซสเซอร์แต่ละตัวจะมีแคชของตัวเอง เมื่อเกิดการอัพเดตแคชของแต่ละโพรเซสเซอร์ขึ้นก็จะมีการอัพเดตที่หน่วยความจำหลักด้วย ดังนั้นปัญหาคือ ข้อมูลที่อัพเดตของแคชหนึ่งอาจจะเป็นข้อมูลที่ไม่ได้อัพเดตจากอีกแคชหนึ่งก็ได้ เรียกว่า การเกิด consistency ระหว่างแคชกับหน่วยความจำหลัก ซึ่งแก้ปัญหาโดยใช้ฮาร์ดแวร์ดังนี้

- ใช้การสื่อสาร (broadcast) บอกกับแคชตัวอื่นว่าจะมีการอัพเดต line ในหน่วยความจำหลัก ดังนั้น ถ้ามี line อยู่ในแคชตัวใดก็ให้อัพเดต line นั้นด้วย
- ใช้การ snooping คือ การให้แคชทุกตัวมีฮาร์ดแวร์สำหรับคอยตรวจสอบการเปลี่ยนแปลงบนบัส ถ้ามีการอัพเดต line ในหน่วยความจำหลักก็จะมีการตรวจสอบว่ามี line อยู่ในแคชตัวใดบ้างแล้วให้อัพเดตแคชที่มี line เหล่านั้นอยู่

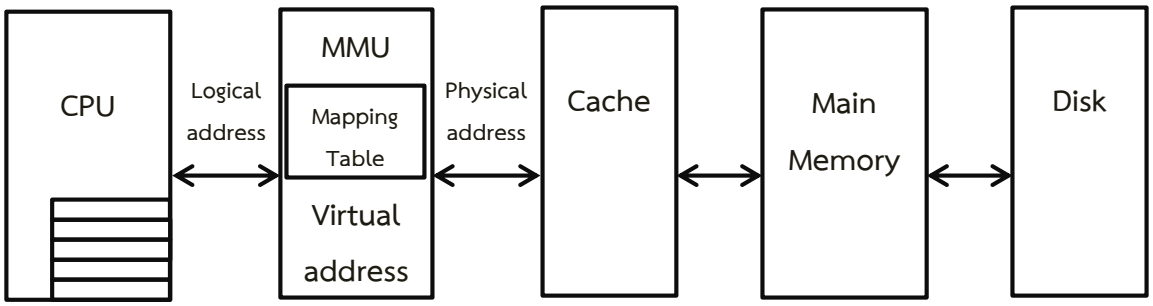
6.5.5 สิ่งที่ต้องคำนึงถึงในการออกแบบ cache memory

1. Address mapping ระหว่างหน่วยความจำกับแคช คือ การกำหนดว่าจะใช้ข้อมูลจำนวนกี่บิตในการที่จะ match กับ tag ในแคชเพื่อให้เกิดประสิทธิภาพมากที่สุด ส่วนกรณีที่แคชเต็มจะเกิด cache miss จะต้องมีการทรานสเฟอร์ข้อมูลจากหน่วยความจำมาไว้ที่แคชโดยตรวจสอบว่าจะแทนที่ข้อมูลตำแหน่งใด หรือจะ roll out ข้อมูลที่ตำแหน่งใดออกไป

2. เมื่อมีการอัพเดตข้อมูลที่แคชแล้ว ควรอัพเดตข้อมูลที่อยู่ในหน่วยความจำหลักไปพร้อมกันด้วย เพราะ line ที่อยู่ในแคชเป็น image หรือเป็นคู่แฝดกับ line ที่อยู่ในหน่วยความจำหลัก

6.6 หน่วยความจำเสมือน

หน่วยความจำเสมือน (Virtual Memory) เป็นการแบ่งเนื้อที่ส่วนหนึ่งของฮาร์ดดิสก์มาทำหน้าที่เสมือนเป็นหน่วยความจำแรมของเครื่อง เพื่อสามารถรองรับงานต่าง ๆ ที่ใช้เนื้อที่ในการเก็บข้อมูลมากเกินกว่าที่หน่วยความจำแรมจะรับได้ ตัวอย่างเช่น ผู้ใช้ได้เปิดงานที่ 1 และงานที่ 2 ไว้ในหน่วยความจำแรม และต้องการจะเปิดงานที่ 3 ขึ้นมาใช้ แต่ไม่สามารถบันทึกลงในหน่วยความจำแรมได้ การทำงานของหน่วยความจำเสมือน คือ จะดึงงานเก่าที่สุดออกมาสร้างเป็นไฟล์ชั่วคราวเพื่อ swap ลงในฮาร์ดดิสก์ในส่วนของพื้นที่ที่จองไว้สำหรับทำหน่วยความจำเสมือน ซึ่งทำให้หน่วยความจำแรมสามารถมีเนื้อที่ว่างเพื่อนำงานที่ 3 มาไว้ในหน่วยความจำแรมได้



ภาพที่ 6.21 แสดงลำดับชั้นของหน่วยความจำของคอมพิวเตอร์

6.6.1 การใช้งานหน่วยความจำเสมือน

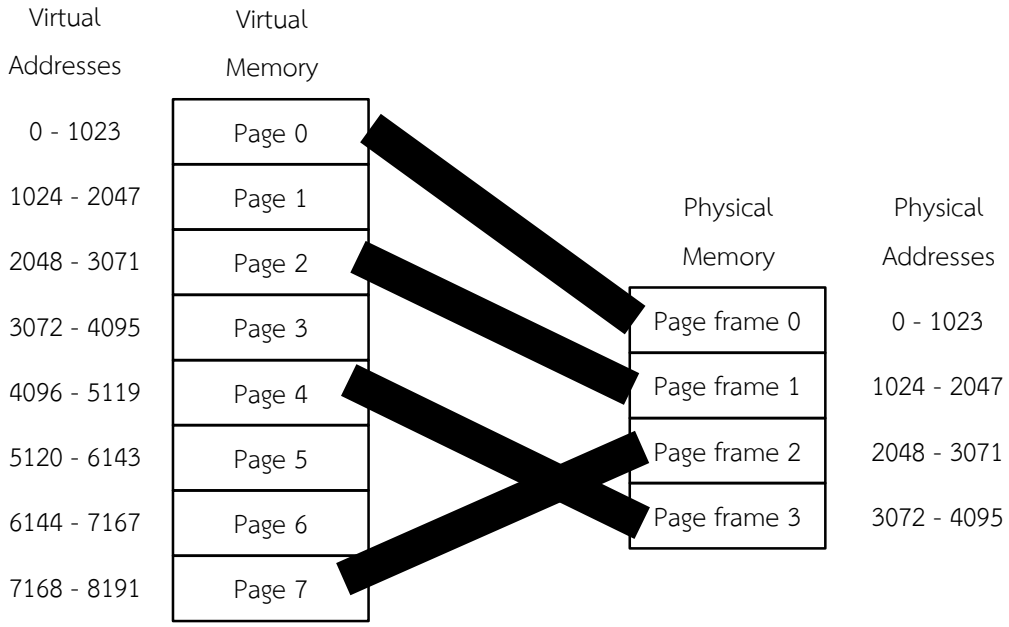
ในยุคแรก ๆ หน่วยความจำของคอมพิวเตอร์มีขนาดเล็กและราคาแพง การเขียนโปรแกรมขนาดใหญ่จึงต้องใช้หน่วยความจำให้เต็มประสิทธิภาพมากที่สุด โดยการใช้หน่วยความจำสำรองมาช่วย โปรแกรมเมอร์จะต้องแบ่งโปรแกรมออกเป็นโปรแกรมย่อย ๆ เรียกว่า โอเวอร์เลย์ (Overlay) โดยที่แต่ละโปรแกรมย่อยจะต้องมีขนาดที่พอดีกับหน่วยความจำ การทำงานของโปรแกรมนั้นจะเริ่มจากโปรแกรมส่วนแรกเริ่มทำงานก่อนสักครู่หนึ่ง เมื่องานส่วนแรกเสร็จก็จะเรียกโอเวอร์เลย์ส่วนที่สองเข้ามายังหน่วยความจำเดิม

และทำงานต่อไปจนเสร็จแล้วจึงดึงโอเวอร์เลย์ส่วนถัดไปขึ้นมาทำงาน การทำงานจะวนแบบนี้ไปเรื่อย ๆ โดยที่โปรแกรมเมอร์ต้องมีหน้าที่แบ่งโปรแกรม แล้วเลือกว่าโปรแกรมโอเวอร์เลย์แต่ละส่วนจะถูกเก็บไว้ที่หน่วยความจำส่วนใด

มีการใช้เทคนิคโอเวอร์เลย์กันอยู่หลายปี ต่อมาได้มีผู้นำเสนอวิธีการทำโอเวอร์เลย์แบบอัตโนมัติ โดยให้คอมพิวเตอร์ทำแทนโปรแกรมเมอร์ เทคนิคนี้เรียกว่า หน่วยความจำเสมือน (Virtual Memory) ซึ่งมีมาตั้งแต่ปี ค.ศ.1970 ซึ่งเป็นเทคนิคที่อยู่ในระบบคอมพิวเตอร์ทั่ว ๆ ไป (น.ท.ไพศาล, Processor Pipeline และ Superscalar, แคช (Cache) และหน่วยความจำเสมือน (Virtual Memory), 2547, หน้า 29)

1. การแบ่งออกเป็นหน้า (Paging)

เป็นวิธีที่จัดการหน่วยความจำแบบอัตโนมัติทำหน้าที่โดยระบบปฏิบัติการ วิธีการทำงานเริ่มต้นจากมีการแบ่งพื้นที่และแอดเดรสของหน่วยความจำเสมือนออกเป็นบล็อกเท่า ๆ กันเรียกว่า หน้า (Page) ขนาดของหน้าโดยปกติมีขนาดเช่น 2^{10} = 1024 ไบต์ ลักษณะของการแบ่งเป็นหน้านี้ช่วยทำให้ดูเหมือนว่าหน่วยความจำหลักมีขนาดใหญ่ขึ้น เนื่องจากมีการเชื่อมโยงหน่วยความจำหลักเข้ากับหน่วยความจำเสมือนบางส่วน ซึ่งโดยปกติก็จะเก็บไว้ในฮาร์ดดิสก์



ภาพที่ 6.22 แสดงการเชื่อมโยง (การ map) ระหว่างหน่วยความจำหลักและหน่วยความจำเสมือน

การใช้งานหน่วยความจำเสมือนนั้นจะต้องมีการจัดการเรื่องของการอ้างถึงหน่วยความจำเสมือนในส่วนอื่น ๆ ที่ไม่ได้ใช้งานหรือไม่ได้ถูก map เข้ากับหน่วยความจำหลัก

กระบวนการที่เกิดขึ้นเมื่อมีการอ้างอิงถึงหน่วยความจำเสมือนที่ไม่ได้อยู่ในหน่วยความจำหลัก เรียกว่า การผิดหน้า (Page-fault) เป็นดังนี้

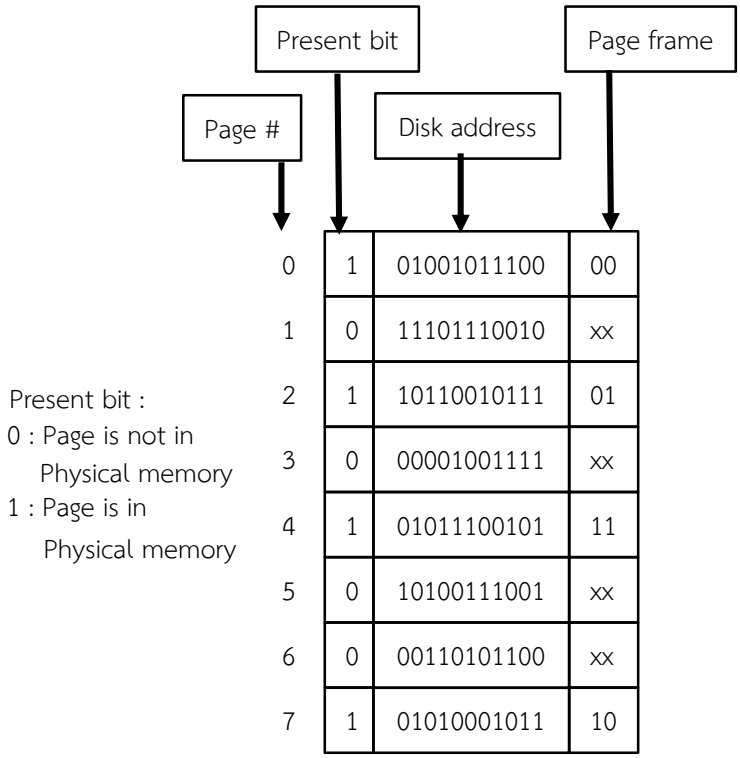
- ถ้าเฟรมของหน้าใดหน้าหนึ่งถูกกำหนดว่าจะมีการเขียนโปรแกรมหรือข้อมูลทับ เนื้อหาที่อยู่ในเฟรมของหน้านั้นก็จะถูกเขียนลงในหน่วยความจำสำรอง (กรณีที่มีการแก้ไขเปลี่ยนแปลงข้อมูล) เพื่อให้ค่าดังกล่าวถูกบันทึกก่อนที่เฟรมของหน้านั้นจะถูกเขียนข้อมูลทับ
- หน้าของข้อมูลที่เราต้องการเข้าถึงและเก็บไว้ในหน่วยความจำสำรองจะถูกเขียนลงในหน่วยความจำหลัก (ในเฟรมหน้าที่เราได้กำหนดไว้ในข้อที่ 1)
- ตารางหน้า (Page Table) จะถูกอัพเดตการเชื่อมโยงข้อมูลระหว่างหน่วยความจำเสมือนเข้ากับหน่วยความจำหลักใหม่
- ทำงานอื่น ๆ ต่อไป

2. ตารางหน้า (Page Table)

ตารางหน้าจะมีจำนวนเรคอร์ดเท่ากับจำนวนหน้าที่มีในหน่วยความจำเสมือนและฟิลด์ต่าง ๆ ที่สำคัญดังนี้

- Present bit มีขนาดเท่ากับ 1 และเป็นตัวแสดงว่าหน้าดังกล่าวอยู่ในหน่วยความจำหลักหรือไม่ ถ้ามีการกำหนดค่าเป็น 1 แสดงว่ามีอยู่ในหน่วยความจำหลัก ส่วน 0 แสดงว่า ไม่มีอยู่ในหน่วยความจำหลัก ถ้ามีการพยายามเข้าถึงข้อมูลของหน้าที่บิตนี้กำหนดเป็น 0 แล้วก็จะทำให้เกิดปัญหาการผิดหน้า (Page fault) ได้
- Disk address เป็นตัวชี้ (pointer) ที่ชี้ไปยังตำแหน่งที่หน้าดังกล่าวถูกเก็บไว้ในดิสก์ ซึ่งโดยปกติแล้วระบบปฏิบัติการจะทำหน้าที่จัดการเกี่ยวกับการเข้าถึงดิสก์ ดังนั้น ตารางหน้าจึงต้องการการดูแลแอดเดรสของดิสก์ที่ระบบปฏิบัติการได้กำหนดให้กับบล็อกข้อมูลต่าง ๆ เมื่อระบบเริ่มต้นทำงาน และแอดเดรสของดิสก์นี้โดยปกติจะไม่ค่อยมีการเปลี่ยนแปลงในช่วงที่มีการทำงานหรือการคำนวณอยู่แล้ว
- Page frame เป็นฟิลด์ที่มีความสำคัญมากที่สุด เพราะฟิลด์นี้มีหน้าที่กำหนดว่าหน้าเฟรมนี้จะไปอยู่กับเฟรมเลขที่เท่าใดในหน่วยความจำหลัก และสำหรับหน้าที่ไม่ได้อยู่ในหน่วยความจำหลัก ฟิลด์นี้จะเก็บค่าที่ไม่สื่อความหมายที่เขียนเป็นค่า XX

การโหลดโปรแกรมเข้าไปใช้งานในหน่วยความจำนั้นบางครั้งก็ใช้เวลานาน และบางครั้งโปรแกรมอาจจะยังไม่เคยถูกเรียกใช้งานทั้งหมดก็ได้ ทำให้เวลาที่ต้องการโหลดโปรแกรมจากดิสก์เข้าสู่หน่วยความจำลดลงได้โดยการโหลดข้อมูลเฉพาะส่วนที่ต้องการในโปรแกรมนั้น สำหรับช่วงระยะเวลาหนึ่งที่ทำงานเท่านั้น วิธีการโหลดหน้าที่ต้องการ (demand page) จะไม่ทำการโหลดหน้าเข้าสู่หน่วยความจำหลักจนกระทั่งมีการผิดหน้า (page fault) เกิดขึ้น หลังจากที่รันโปรแกรมไปได้สักระยะหนึ่งจะมีเพียงหน้าที่กำลังถูกใช้งานเท่านั้นที่จะอยู่ในหน่วยความจำหลัก

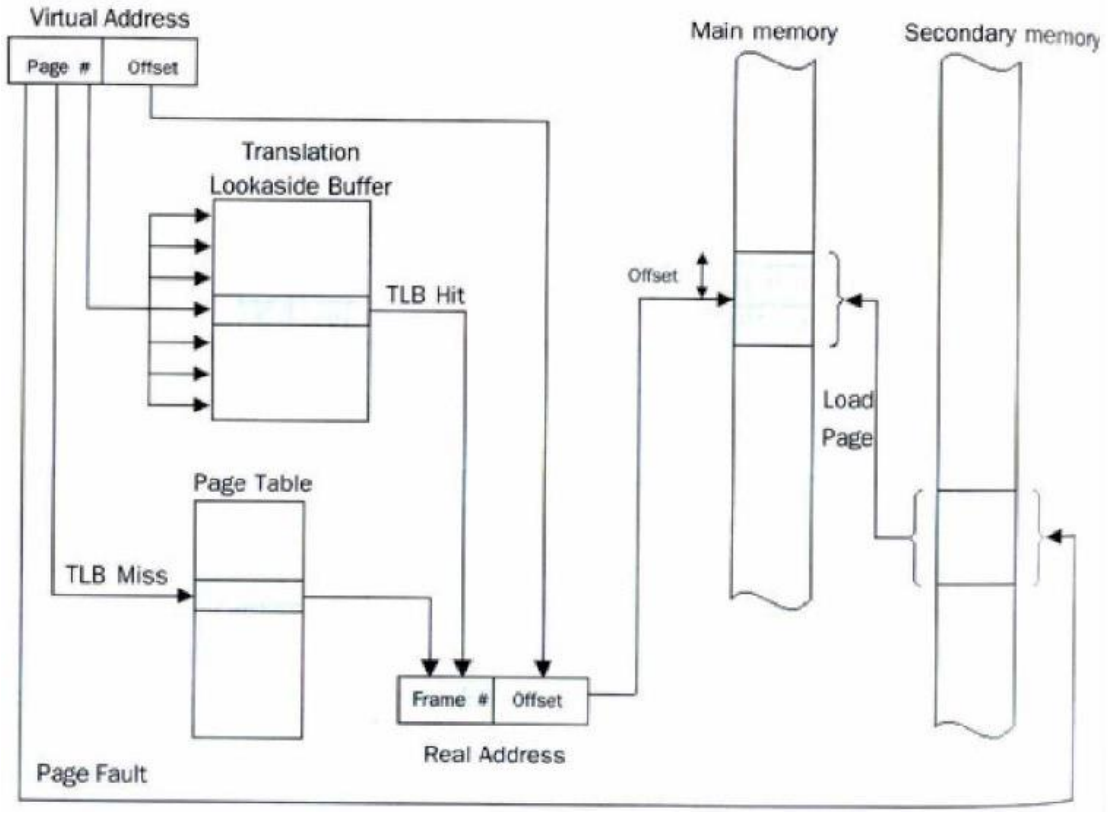


0	1	01001011100	00
1	0	11101110010	xx
2	1	10110010111	01
3	0	00001001111	xx
4	1	01011100101	11
5	0	10100111001	xx
6	0	00110101100	xx
7	1	01010001011	10

ภาพที่ 6.23 แสดงตารางหน้า (Page Table) ของหน่วยความจำเสมือน

3. บัฟเฟอร์ค้นหาที่อยู่ (Translation Look-aside Buffer : TLB)

ทุก ๆ ครั้งที่มีการอ้างถึงหน่วยความจำเสมือน หน่วยความจำหลักจะถูกเรียกใช้ 2 ครั้งด้วยกัน คือ มีการดึงแถวของตารางหน้า และการดึงข้อมูลที่ต้องการออกมาจากหน่วยความจำหลัก ดังนั้น การทำงานของหน่วยความจำเสมือนจะใช้เวลาในการเข้าถึงหน่วยความจำเป็นสองเท่า วิธีการป้องกันปัญหาที่เกิดขึ้นนี้ทำได้โดยใช้บัฟเฟอร์หรือหน่วยความจำพิเศษในการเก็บแถวของตารางหน้าที่เรียกว่า บัฟเฟอร์ค้นหาที่อยู่ (TLB) ซึ่งบัฟเฟอร์นี้จะทำหน้าที่เก็บแถวของตารางหน้าที่เพิ่งถูกเรียกใช้งานเสร็จแล้ว การทำงานของบัฟเฟอร์แสดงดังภาพที่ 6.24 (น.ท.ไพศาล, Processor Pipeline และ Superscalar, แคช (Cache) และหน่วยความจำเสมือน (Virtual Memory), 2547, หน้า 32)



ภาพที่ 6.24 แสดงบัฟเฟอร์ค้นหาที่อยู่ (Translation Look-aside Buffer : TLB)
ที่มา (น.ท.ไพศาล โมลิสกุลมงคล และคณะ สถาปัตยกรรมคอมพิวเตอร์, 2547)

เมื่อมีการให้ค่าแอดเดรสเสมือนกับระบบซีพียูจะทำการค้นหาในตารางหน้าของบัฟเฟอร์ค้นหาที่อยู่ก่อน ถ้าพบแถวที่ต้องการในตารางหน้า (TLB hit) ก็สามารถเก็บเลขที่เฟรมและแปลงเป็นแอดเดรสจริงได้เลย แต่ถ้าไม่พบแถวที่ต้องการ (TLB miss) ซีพียูจะต้องใช้เลขที่หน้าไปเปรียบเทียบกับอินเด็กซ์ของตารางหน้าเพื่อหาแถวที่ต้องการ และซีพียูจะทำหน้าที่เพิ่มตารางหน้าของบัฟเฟอร์ค้นหาที่อยู่ให้มีแถวของตารางหน้าใหม่นี้ด้วย

เนื่องจากตารางหน้าของบัฟเฟอร์ค้นหาที่อยู่จะเก็บข้อมูลของบางแถวจากตารางหน้าเท่านั้น ทำให้เราไม่สารถเรียงลำดับแถวของตารางหน้าของบัฟเฟอร์ค้นหาที่อยู่ด้วยเลขที่หน้าได้ ดังนั้นค่าของฟิลด์ในแถวของตารางหน้าของบัฟเฟอร์ค้นหาที่อยู่จะต้องมีเลขที่หน้ากำกับด้วย และการค้นหาเลขที่หน้าในตารางหน้าของบัฟเฟอร์ค้นหาที่อยู่ของซีพียูนั้นเป็นการเชื่อมโยงข้อมูลแบบสัมพันธ์ (Associative mapping)

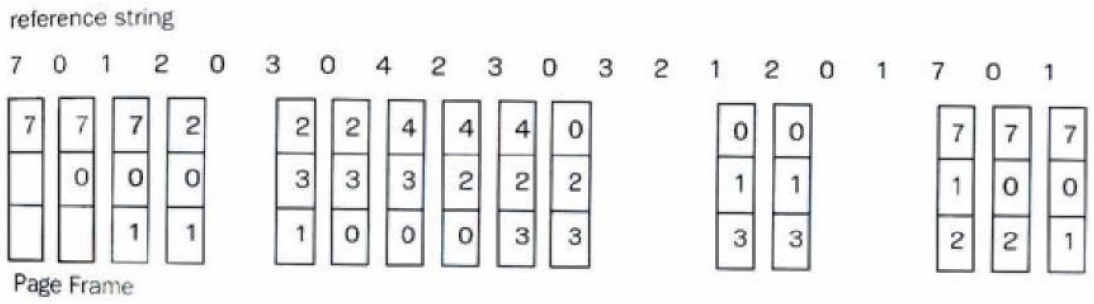
4. การสับเปลี่ยนหน้า (Page Replacement Algorithms)

เป็นการเลือกหน้าในหน่วยความจำที่จะถูกแทนที่ด้วยหน้าใหม่ที่จะเพิ่มเข้ามา มี 6 วิธี ดังนี้ (น.ท.ไพศาล, Processor Pipeline และ Superscalar, แคช (Cache) และหน่วยความจำเสมือน (Virtual Memory), 2547, หน้า 33-39)

4.1 วิธีสับเปลี่ยนแบบมาก่อน-ออกก่อน (First-in First-out Algorithm)

เป็นวิธีการสับเปลี่ยนหน้าที่ง่ายที่สุด วิธีการคือ ใช้เวลาที่หน้านั้น ๆ ถูกนำเข้ามาในหน่วยความจำหลักเป็นเกณฑ์ในการตัดสินใจ เมื่อต้องการเลือกหน้าใดออกก็ให้เลือกจากหน้าที่เข้ามานานที่สุด

ตัวอย่าง สมมติว่าในระบบมี 3 เฟรม และว่างอยู่ ดังภาพ

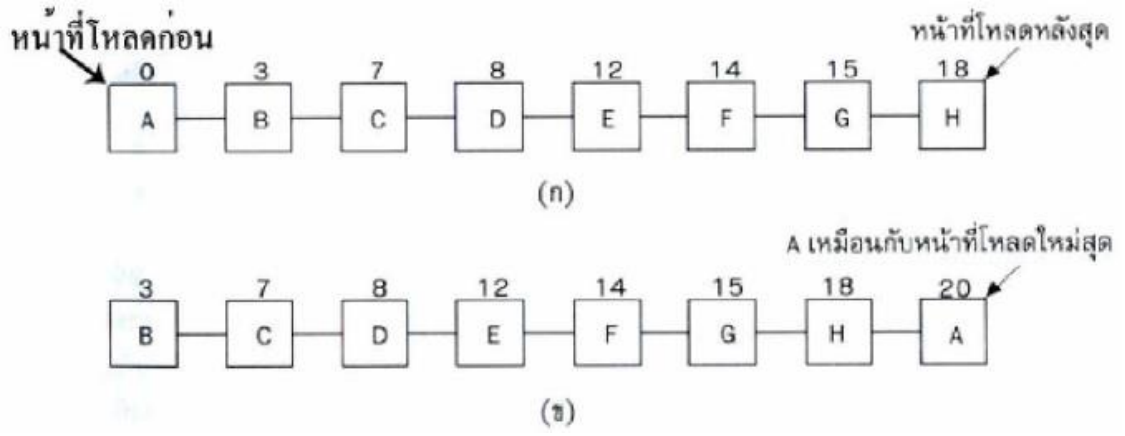


- จากแถวของหน้าที่เข้ามาในเฟรมคือ 7,0,1 ซึ่งจะเกิดการผิดหน้า (Page-fault) เนื่องจากหน้าที่ 3 ยังไม่ได้อยู่ในเฟรม จึงมีการนำหน้าที่ต้องการเข้ามาในหน่วยความจำ
- เรียกหน้าต่อไปคือ หน้า 2 จะถูกสับเปลี่ยนเข้ามาแทนหน้า 7 เพราะหน้า 7 เข้ามาก่อนเป็นหน้าแรก
- เรียกครั้งต่อไปเป็นหน้า 0 แต่หน้า 0 อยู่ในหน่วยความจำอยู่แล้ว จึงไม่เกิดการผิดหน้าและไม่เกิดการสับเปลี่ยนหน้าด้วย
- เรียกครั้งต่อไปหน้า 3 จะถูกสับเปลี่ยนเข้ามาแทนหน้า 0
- และต่อไปหน้า 0 จะถูกสับเปลี่ยนเข้ามาแทนหน้า 1 เป็นเช่นนี้ต่อไปเรื่อย ๆ
- จากภาพ ทุก ๆ ครั้งที่เกิดการผิดหน้า จะเห็นได้ว่าหน้าใดที่จะถูกสับเปลี่ยนเข้ามาในเฟรม ซึ่งการผิดหน้าที่เกิดในตัวอย่างนี้มีทั้งหมด 15 ครั้ง

วิธีสับเปลี่ยนแบบมาก่อน-ออกก่อนนี้เป็นวิธีที่เข้าใจง่ายและเขียนโปรแกรมได้ง่าย แต่อาจไม่มีประสิทธิภาพเท่าใดเพราะหน้าที่ถูกสับเปลี่ยนออกไปอาจเป็นส่วนเริ่มต้นของโปรแกรมซึ่งถูกเรียกใช้ในตอนต้น ๆ และไม่มีความต้องการเรียกใช้งานอีกต่อไป หรือหน้าที่ถูกสับเปลี่ยนออกไปอาจเก็บค่าตัวแปรหลักของโปรแกรมซึ่งถูกเรียกใช้บ่อยครั้ง โดยถูกกำหนดเป็นค่าเริ่มต้นในส่วนแรก ๆ ของโปรแกรม ให้สังเกตว่าถึงแม้หน้าที่กำลังใช้งานจะถูกสับเปลี่ยนออกไปแต่การทำงานของโพรเซสก็ยังคงถูกต้องเหมือนเดิม เพราะหลังจากที่นำหน้าที่กำลังใช้งานออกไปเพื่อใส่หน้าใหม่ การผิดหน้าก็จะเกิดขึ้นเกือบจะทันทีเพื่อนำหน้าที่กำลังถูกใช้งานนั้นกลับคืนมา และหน้าบางหน้าในหน่วยความจำก็จะถูกสับเปลี่ยนออกไปแทน ดังนั้นการสับเปลี่ยนหน้าที่ไม่มีประสิทธิภาพจะเพิ่มอัตราการผิดหน้าได้ และส่งผลให้โพรเซสทำงานช้าลง แต่ไม่ทำให้ระบบทำงานผิดพลาด

4.2 วิธีสับเปลี่ยนแบบให้โอกาสครั้งที่สอง (Second Change Page Replacement Algorithm)

ปรับปรุงประสิทธิภาพจากวิธีสับเปลี่ยนแบบมาก่อน-ออกก่อน โดยการป้องกันการเปลี่ยนหน้าที่ถูกเรียกใช้งานบ่อยออกไป ซึ่งสามารถทำได้โดยการตรวจสอบที่บิต Referenced (R) ของหน้าที่เข้ามานานที่สุด ถ้าบิต R มีค่าเป็น 0 ก็แสดงว่าหน้านั้นเก่าและไม่ได้ถูกเรียกใช้งานเลย ระบบก็สามารถทำการสับเปลี่ยนได้ทันที แต่ถ้าบิต R มีค่าเท่ากับ 1 ก็ให้กำหนดให้บิต R นั้นเป็น 0 และนำหน้านั้นกลับไปเข้าแถวใหม่อีกครั้ง พร้อมกับทำการเปลี่ยนแปลงเวลาของหน้านั้นใหม่ให้เหมือนกับว่าหน้านั้นเพิ่งเข้ามาในหน่วยความจำ จากนั้นก็ทำการค้นหาหน้าที่จะถูกสับเปลี่ยนต่อไป ดังตัวอย่าง

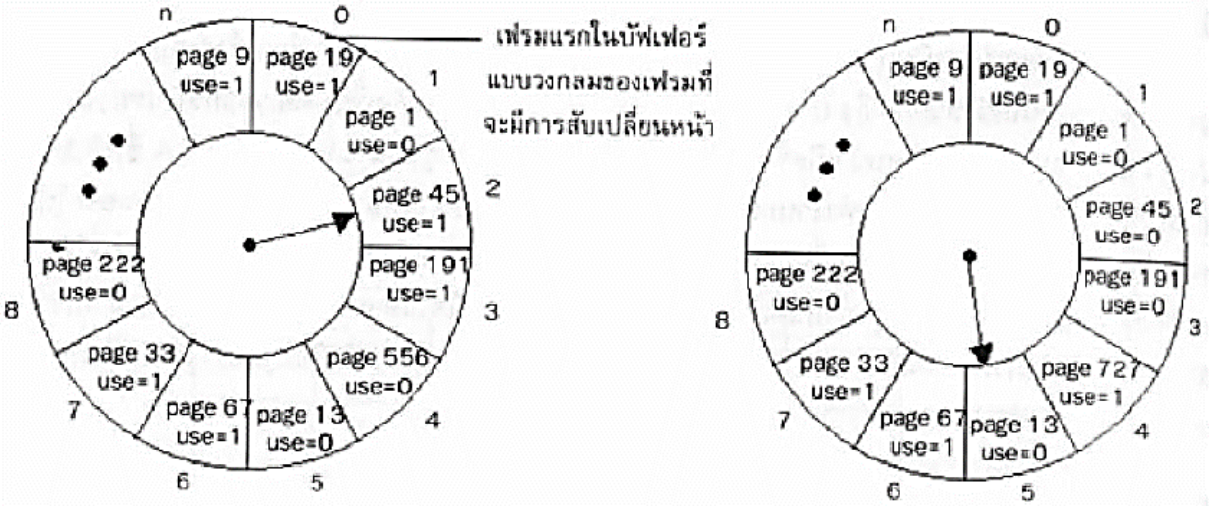


จากภาพ จะเห็นว่า หน้า A ถึงหน้า H จะถูกเก็บไว้ในลิงค์ลิสต์และถูกจัดเรียงโดยเวลาที่หน้าเหล่านี้เข้ามาในหน่วยความจำ สมมติว่าการผิดหน้าเกิดขึ้นเมื่อเวลาเท่ากับ 20 หน้าที่อยู่มานานที่สุดคือหน้า A ซึ่งเข้ามาตั้งแต่เวลาเท่ากับ 0 (เมื่อโพรเซสแรกเริ่มทำงาน) ถ้าบิต R ในหน้า A มีค่าเป็น 0 หน้า A ก็จะถูกสับเปลี่ยนออกไป (ถ้ามีการเปลี่ยนแปลง ก็อาจถูกเขียนลงดิสก์ แต่ถ้าไม่มีอะไรเปลี่ยนแปลงก็ถูกทิ้งไปเฉย ๆ ซึ่งสามารถดูได้จากบิต Modified (M)) แต่ถ้าบิต R มีค่าเป็น 1 หน้า A ก็จะถูกนำไปใส่ในตอนท้ายของแถว และเวลาที่เข้ามาในลิสต์ก็จะถูกเปลี่ยนเป็นเวลาปัจจุบันคือ 20 ในขณะเดียวกันบิต R จะถูกลบเป็น 0 และหน้าที่จะถูกตรวจสอบเพื่อสับเปลี่ยนออกต่อไปคือหน้า B

วิธีสับเปลี่ยนแบบให้โอกาสครั้งที่สองเป็นวิธีการที่สมเหตุสมผลพอสมควร แต่ก็ยังไม่มีประสิทธิภาพมากนัก เพราะการที่ระบบย้ายหน้าไปมาในลิสต์นั้น ทำให้เกิดการเสียเวลาไปพอสมควร

4.3 วิธีสับเปลี่ยนแบบวงรอบนาฬิกา (Clock Page Replacement Algorithm)

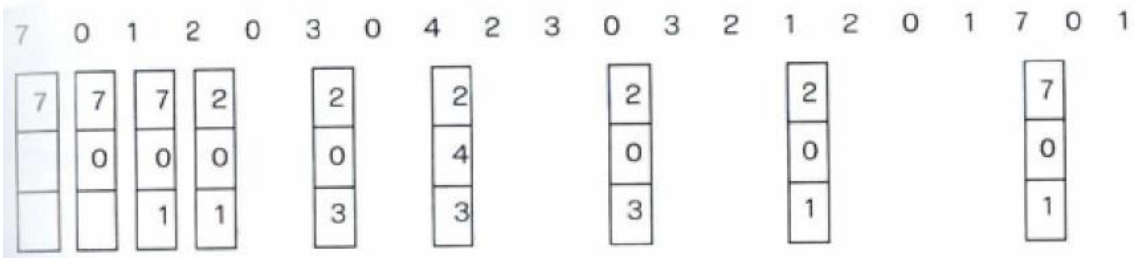
เมื่อมีการผิดหน้าเกิดขึ้น หน้าที่มีเข็มนาฬิกาชี้อยู่จะถูกตรวจสอบ ถ้าบิต R มีค่าเป็น 0 หน้านั้นก็จะถูกสับเปลี่ยนออกไป และหน้าใหม่ก็จะถูกใส่เข้ามาในตำแหน่งเดิมพร้อมกันนั้นเข็มนาฬิกาก็จะเลื่อนไปข้างหน้า 1 ตำแหน่ง แต่ถ้าบิต R ถูกกำหนดเป็น 1 ก็ให้ลบค่าของบิตนั้นเป็น 0 และเลื่อนเข็มไปหน้าถัดไป วิธีการนี้จะถูกทำซ้ำจนกว่าจะได้หน้าที่มีบิต R เป็น 0 ดังภาพที่ 6.25



ภาพที่ 6.25 แสดง (ก) สถานะของบัฟเฟอร์ก่อนที่จะมีการสับเปลี่ยนหน้า (ข) สถานะของบัฟเฟอร์หลังจากสับเปลี่ยนหน้าต่อไป

4.4 วิธีสับเปลี่ยนแบบที่ดีที่สุด (Option Page Replacement Algorithm : OPT)

วิธีสับเปลี่ยนแบบที่ดีที่สุดทำให้เกิดอัตราการผิดหน้าน้อยที่สุด มีวิธีการคือ ให้เลือกสับเปลี่ยนหน้าที่ไม่ถูกเรียกใช้งาน และมีระยะเวลารอการใช้งานนานที่สุด ตัวอย่าง



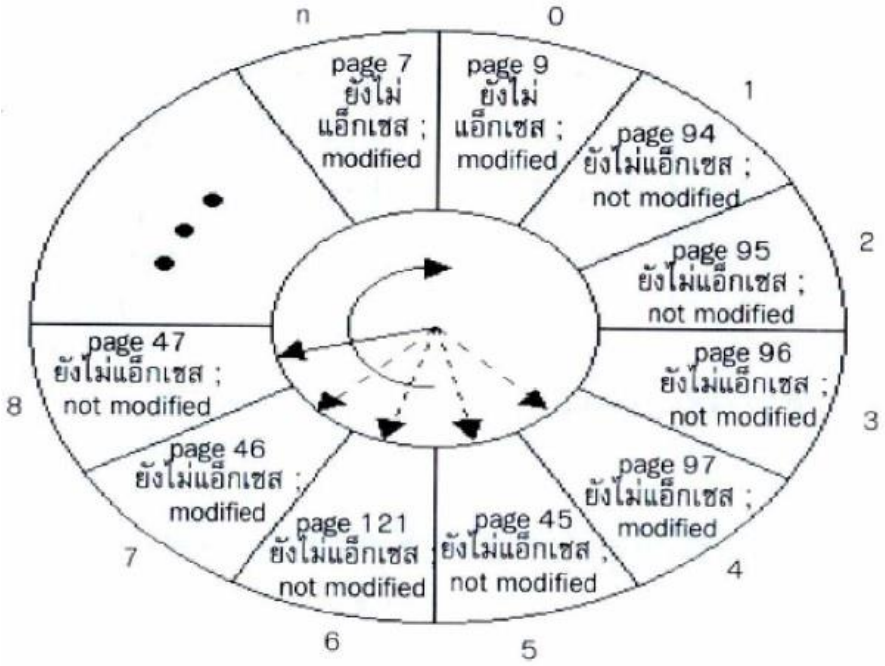
- การเรียกสามหน้าแรกเข้ามาใช้งาน ทำให้เกิดการผิดหน้าสามครั้ง
- การเรียกหน้า 2 เข้ามา จะสับเปลี่ยนหน้า 7 ออกไป เพราะหน้า 7 จะถูกเรียกใช้งานอีกครั้งเป็นลำดับที่ 18 ส่วนหน้า 0 จะถูกเรียกใช้งานอีกเป็นลำดับที่ 5 และหน้า 1 จะถูกเรียกใช้งานอีกเป็นลำดับที่ 14
- การเรียกหน้า 3 เข้ามา จะสับเปลี่ยนหน้า 1 ออก เพราะหน้า 1 เป็นหน้าสุดท้ายที่จะถูกเรียกใช้งาน ส่วนหน้า 0 จะถูกเรียกใช้งานอีกเป็นลำดับที่ 1 หน้า 2 จะถูกเรียกใช้งานอีกเป็นลำดับที่ 3 (เทียบจากตำแหน่งที่หน้า 3 กำลังเข้าใช้งาน)

จากตัวอย่างจะเห็นได้ว่า วิธีสับเปลี่ยนแบบที่ดีที่สุด จะทำให้เกิดการผิดหน้า 9 ครั้งซึ่งดีกว่าวิธีสับเปลี่ยนแบบมาก่อน-ออกก่อนมาก (มีการผิดหน้า 15 ครั้ง) ถ้าเราไม่คิดการผิดหน้า 3 ครั้งแรก (ซึ่งจะต้องเกิดขึ้นอย่างแน่นอนไม่ว่าจะใช้วิธีสับเปลี่ยนแบบใด) จะเห็นได้ว่าวิธีสับเปลี่ยนแบบที่ดีที่สุดดีกว่าวิธี

สับเปลี่ยนแบบมาก่อน-ออกก่อนถึง 2 เท่า แต่น่าเสียดายที่วิธีสับเปลี่ยนแบบที่ดีที่สุดสร้างเพื่อใช้งานจริงได้ยาก เนื่องจากเราต้องรู้ว่า จะมีการเรียกหน้าใด เมื่อไร ดังนั้นวิธีสับเปลี่ยนแบบที่ดีที่สุดจะมีไว้เพื่อใช้ในการเปรียบเทียบเท่านั้นถ้าเรามีวิธีสับเปลี่ยนหรือมีอัลกอริธึมใหม่เราก็อาจนำมาใช้เปรียบเทียบกับวิธีสับเปลี่ยนแบบที่ดีที่สุดเพื่อจะได้ทราบว่าวิธีสับเปลี่ยนแบบใหม่มีประสิทธิภาพใกล้เคียงวิธีสับเปลี่ยนแบบที่ดีที่สุดเท่าใด

4.5 วิธีสับเปลี่ยนแบบที่ไม่ได้ใช้งานออกก่อน (Not Recently Used : NRU)

เป็นการนำวิธีสับเปลี่ยนแบบวงรอบนาฬิกามาเพิ่มประสิทธิภาพในการทำงานโดยการพิจารณาบิตที่สำคัญในตารางหน้าเพิ่มอีก 1 บิต เนื่องจากระบบสามารถทำการเก็บข้อมูลสถิติว่าหน้าใดกำลังถูกใช้หรือหน้าใดที่ไม่ได้ใช้งานโดยดูจากบิตในแถวของตารางหน้าที่แสดงสถานการณ์ทำงานของแต่ละหน้า ซึ่งบิต R จะถูกกำหนดเป็น 1 เมื่อหน้านั้นถูกเรียกใช้งาน และบิต M ถูกกำหนดเป็น 1 เมื่อใดบิตนั้นก็จะมีค่าเท่ากันจนกว่าระบบปฏิบัติการจะกำหนดค่ากลับเป็น 0 โดยใช้ซอฟต์แวร์



ภาพที่ 6.26 แสดงวิธีสับเปลี่ยนแบบที่ไม่ได้ใช้งานออกก่อน (Not Recently Used : NRU)

จากภาพที่ 6.26 ทั้งบิต R และบิต M สามารถนำมาใช้ในการสร้างวิธีการสับเปลี่ยนหน้าแบบใหม่ได้โดยเมื่อโพรเซสหนึ่ง ๆ เริ่มทำงานระบบปฏิบัติการจะกำหนดให้ทั้งบิต R และ M ในทุก ๆ หน้าเป็น 0 เมื่อครบวงรอบของระยะเวลาหนึ่ง ๆ เช่น จากการแทรกของอินเทอร์รัพท์ (clock interrupt) บิต R จะถูกกำหนดค่าเป็น 0 เพื่อเป็นตัวแยกหน้าที่ไม่ได้ถูกเรียกใช้งานออกจากหน้าอื่น ๆ เมื่อมีการผิดหน้าเกิดขึ้นระบบปฏิบัติการจะทำการตรวจสอบทุกหน้าและแบ่งหน้าเหล่านั้นออกเป็น 4 กลุ่ม ขึ้นอยู่กับค่าของบิต R และบิต M ดังนี้

กลุ่มที่ 0 : ไม่ถูกเรียกใช้งาน และไม่มีการเปลี่ยนแปลงค่า

กลุ่มที่ 1 : ไม่ถูกเรียกใช้งาน แต่มีการเปลี่ยนแปลงค่า

กลุ่มที่ 2 : ถูกเรียกใช้งาน แต่ไม่มีการเปลี่ยนแปลงค่า

กลุ่มที่ 3 : ถูกเรียกใช้งาน และมีการเปลี่ยนแปลงค่า

ถึงแม้จะดูเหมือนว่าไม่มีทางเป็นไปได้ที่จะมีหน้าอยู่ในกลุ่มที่ 1 แต่อาจเกิดขึ้นได้ในกรณีที่หน้าในกลุ่มที่ 3 นั้นถูกระบบปฏิบัติการกำหนดค่าบิต R เป็น 0 แต่ไม่ได้กำหนดค่าบิต M เป็น 0 ด้วยเมื่อครบวงรอบของระยะเวลาหนึ่ง ๆ เนื่องจากเราต้องการรู้ว่าหน้านั้นจะต้องถูกเขียนกลับลงไปในดิสก์ด้วยหรือไม่ จึงทำการเคลียร์บิต R แต่ไม่ต้องทำอะไรกับบิต M และทำให้หน้านั้นอยู่ในกลุ่มที่ 1

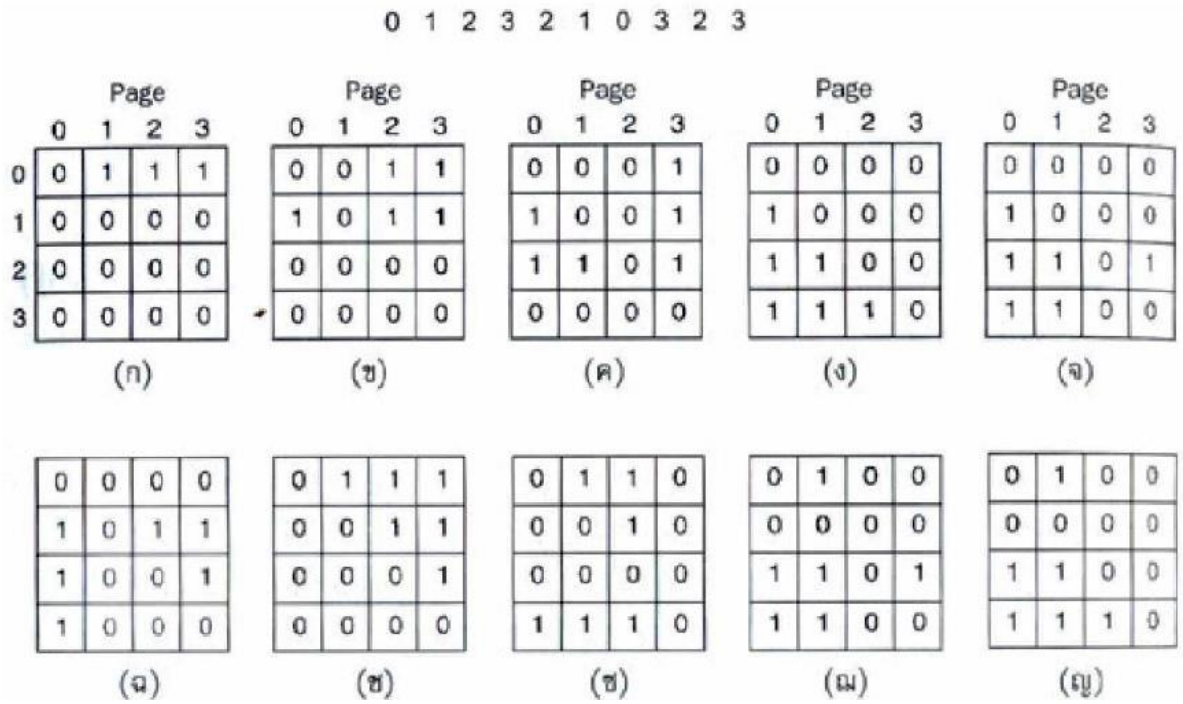
วิธีสับเปลี่ยนแบบที่ไม่ได้ใช้งานออกก่อนนี้จะสุ่มเอาหน้าออกจากกลุ่มลำดับต่ำสุดที่มีหน้าของโพรเซสอยู่ (เช่น กลุ่มที่ 1 มีลำดับต่ำกว่ากลุ่มที่ 2 และกลุ่มที่ 3) และวิธีการนี้ยังพยายามนำเอาหน้าที่ถูกเปลี่ยนแปลงแต่ไม่ได้ถูกเรียกใช้งานหรืออ้างอิงถึงออกทุกครั้งเมื่อครบวงรอบนาฬิกา (ประมาณ 20 msec) มากกว่าหน้าที่ไม่มีการเปลี่ยนแปลงแต่ถูกเรียกใช้งานบ่อย ข้อดีของวิธีนี้คือ ทำความเข้าใจง่าย นำมาใช้งานง่าย และมีประสิทธิภาพค่อนข้างดี นอกจากนี้ยังใช้ในระบบปฏิบัติการของเครื่องแมคอินทอช

4.6 วิธีสับเปลี่ยนแบบใช้งานน้อยที่สุดออกก่อน (Least Recently Used : LRU)

วิธีนี้จะมีการบันทึกเวลาที่แต่ละหน้าถูกอ้างอิงครั้งล่าสุดไว้ เมื่อต้องการเลือกหน้าเพื่อสับเปลี่ยนออกก็จะเลือกหน้าที่ไม่ได้ถูกใช้งานมาเป็นเวลานานที่สุด (หน้าที่มีตัวเลขเวลาน้อยที่สุด) วิธีนี้เป็นการมองย้อนเวลากลับไปไม่มองในอนาคต ความเป็นจริงดูเหมือนใช้งานง่ายในทางทฤษฎี แต่ในทางปฏิบัติใช้งานยากเนื่องจากวิธีสับเปลี่ยนแบบใช้งานน้อยที่สุดออกก่อนนั้นระบบต้องสร้างลิงค์ลิสต์ไว้ในทุก ๆ หน้าของหน่วยความจำที่มีหน้าที่ใช้งานน้อยอยู่ที่หัวแถวส่วนหน้าที่ใช้งานบ่อยจะอยู่ท้ายสุด ส่วนความยากของวิธีนี้คือลิสต์จะต้องทำการปรับเปลี่ยนทุกครั้งที่มีการอ้างอิงถึงหรือมีการเรียกใช้งานในหน่วยความจำ ซึ่งการค้นหาในลิสต์ การลบ หรือการย้ายหน้าไปมานั้นเป็นการทำงานที่ซีพียูต้องเสียเวลาในการทำงานไปทั้งสิ้น

วิธีสับเปลี่ยนแบบใช้งานน้อยที่สุดออกก่อนอาจใช้ฮาร์ดแวร์พิเศษเป็นทางเลือกในการทำงาน โดยการใช้ฮาร์ดแวร์ที่มีตัวนับหรือ Counter (C) แบบ 64 บิต ที่จะทำการเพิ่มค่าทุกครั้งที่มีการเรียกใช้ นอกจากนี้ในตารางหน้าจะต้องมีฟิลด์ขนาดใหญ่ที่สามารถใส่ค่าของตัวนับนี้ได้ หลังจากที่มีการอ้างอิงหน่วยความจำในแต่ละครั้งค่า C ก็จะถูกบันทึกลงในตารางหน้าของหน้านั้น ๆ เมื่อมีการผิดหน้า ระบบปฏิบัติการจะต้องทำการตรวจสอบทุกค่าของ C ในตารางหน้า เพื่อค้นหาค่า C ที่มีค่าน้อยที่สุด ซึ่งแสดงว่าหน้านั้นก็คือหน้าที่ถูกเรียกใช้งานน้อยที่สุดนั่นเอง (น.ท.ไพศาล, Processor Pipeline และ Superscalar, แคช (Cache) และหน่วยความจำเสมือน (Virtual Memory), 2547, หน้า 38-39)

ตัวอย่างของการใช้วิธีสับเปลี่ยนแบบใช้งานน้อยที่สุดออกก่อนด้วยฮาร์ดแวร์ดังที่กล่าวตอนต้น โดยสมมติให้ในระบบมี n เพจเฟรม ซึ่งทำให้ฮาร์ดแวร์นั้นต้องมีเมทริกซ์ n*n บิต ซึ่งมีค่าเริ่มต้นเป็น 0 ทั้งหมด เมื่อใดก็ตามที่เพจเฟรม k ถูกอ้างอิงถึงฮาร์ดแวร์นั้นก็จะกำหนดให้ทุกบิตในแถว k เป็น 1 และกำหนดให้ทุกบิตในคอลัมน์ k เป็น 0 เมื่อมีการตรวจสอบแถวที่มีค่าของบิตรวมกันน้อยที่สุดคือ แถวที่ถูกเรียกใช้งานน้อยที่สุดดังภาพที่ 6.27

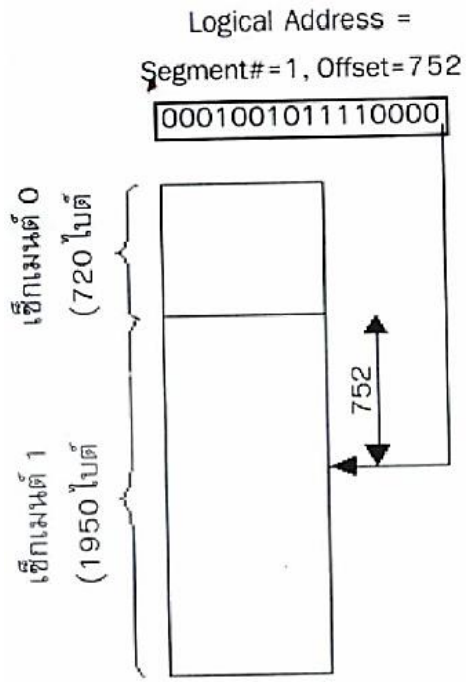


ภาพที่ 6.27 แสดงการใช้วิธีสับเปลี่ยนแบบใช้งานน้อยที่สุดออกก่อนด้วยฮาร์ดแวร์ โดยให้ระบบมี 4 เพจเฟรม

จากภาพที่ 6.27 หลังจากที่หน้า 0 ถูกอ้างอิงถึง จะได้ภาพ (ก) หลังจากที่หน้า 1 ถูกอ้างอิงถึงจะได้เมทริกซ์ดังภาพ (ข) หลังจากที่ทุกหน้าทำงานเสร็จจะได้เมทริกซ์ดังภาพ (ญ) ซึ่งมีค่าของแถวที่ 1 รวมกันน้อยที่สุด ดังนั้นเมื่อเกิดการผิดหน้าเกิดขึ้น หน้าที่อยู่ในเฟรมที่ 1 ก็จะถูกเลือกออกเพราะไม่ได้ถูกเรียกใช้งานมาเป็นเวลานานที่สุด

5. การแบ่งเป็นเซกเมนต์ (Segmentation)

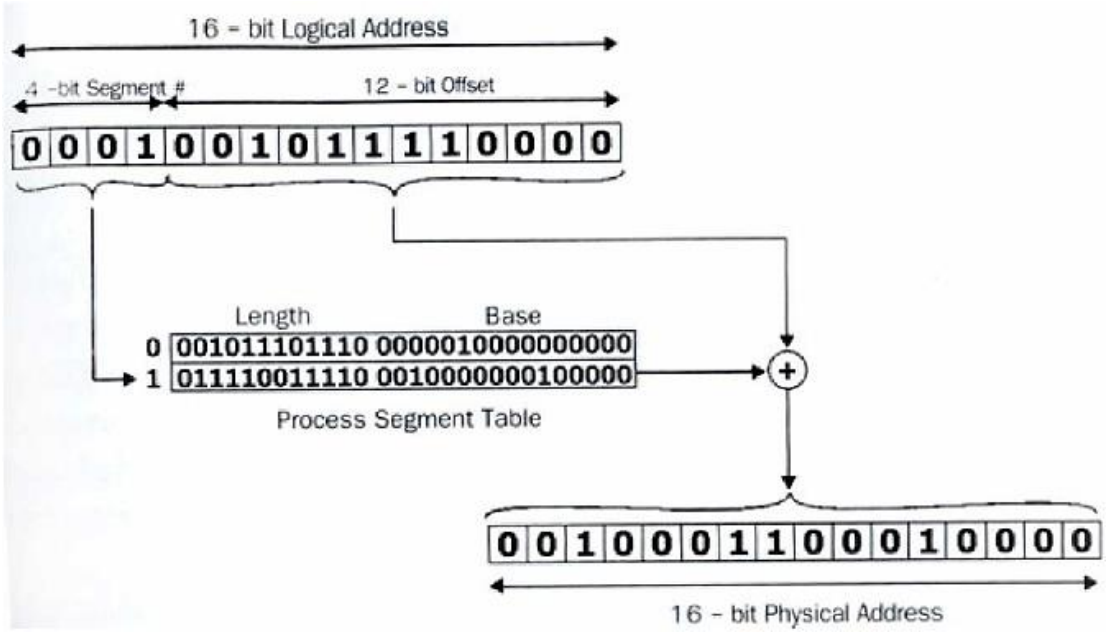
เป็นวิธีการที่มีลักษณะการทำงานคล้ายกับการแบ่งหน้าของโพรเซสแต่วิธีการนี้จะแบ่งโพรเซสออกเป็นส่วน ๆ โดยแต่ละส่วนไม่จำเป็นต้องมีความยาวเท่ากันดังภาพที่ 6.28



ภาพที่ 6.28 แสดงการแบ่งเป็นเซกเมนต์

เนื่องจากการแบ่งเป็นเซกเมนต์ทำให้เกิดชิ้นส่วนที่ไม่เท่ากัน ซึ่งเหมือนกับการแบ่งหน่วยความจำแบบเปลี่ยนแปลงขนาดได้ (Dynamic partitioning) แต่ข้อแตกต่างกันคือ โปรแกรมหนึ่ง ๆ อาจจะมีหลายส่วน (หรือหลายเซกเมนต์) ที่สามารถใช้งานในหลาย ๆ พาร์ทิชันได้ และพาร์ทิชันเหล่านี้ก็ไม่จำเป็นต้องติดกันด้วย การแปลงเป็นเซกเมนต์นั้นจะช่วยลดปัญหาการสูญเสียพื้นที่ภายใน (Inter Fragmentation) ได้ด้วย

ในขณะที่การแบ่งเป็นหน้านั้นโปรแกรมเมอร์จะไม่สามารถมองเห็นและมีส่วนร่วมได้ แต่ในขณะที่การแบ่งเป็นเซกเมนต์นั้นโปรแกรมเมอร์จะสามารถมองเห็นและมีส่วนร่วมในการจัดการกับโปรแกรมและข้อมูลของตนได้ โดยปกติโปรแกรมเมอร์หรือคอมไพเลอร์จะทำการแบ่งโปรแกรมและข้อมูลออกเป็นส่วน ๆ เอง เช่น การใช้โมดูล (Module) ผลจากการแบ่งหน่วยความจำออกเป็นส่วน ๆ ที่ไม่เท่ากันจะทำให้ความสัมพันธ์ระหว่างแอดเดรสทางตรรกะกับแอดเดรสจริงเป็นความสัมพันธ์ที่ค่อนข้างซับซ้อน โดยอาจจะใช้ตารางของเซกเมนต์เก็บรายละเอียดของแต่ละโพรเซสและแอดเดรสเริ่มต้นของที่ว่างในหน่วยความจำ ซึ่งนอกจากจะเก็บจุดเริ่มต้นของเซกเมนต์ภายในแถวของตารางแล้วจะต้องเก็บค่าความยาวของเซกเมนต์นั้น ๆ ด้วยเพื่อเป็นการยืนยันว่าระบบจะไม่ใช้แอดเดรสที่ไม่ถูกต้อง ดังตัวอย่าง



จากภาพ เราจะได้แอดเดรสทางตรรกะเป็น 0001001011110000 ซึ่งอยู่ในเซกเมนต์เลขที่ 1 และมีระยะห่างจากขอบเซกเมนต์เท่ากับ 752 สมมติว่าส่วนนี้อยู่ในหน่วยความจำหลักเริ่มต้นจากแอดเดรสจริง 0010000000100000 ดังนั้นแอดเดรสจริงของเซกเมนต์นี้จะเป็น 0010000000100000 + 001011110000 = 0010001100010000

6.7 สรุป

หน่วยความจำของคอมพิวเตอร์จะใช้ในการเก็บคำสั่งและข้อมูลในขณะที่ระบบคอมพิวเตอร์มีการประมวลผล ซึ่งจะมีหน่วยความจำอยู่ 2 ประเภทคือ หน่วยความจำภายใน (Internal Memory) และหน่วยความจำภายนอก (External Memory) หน่วยความจำภายในหมายถึง หน่วยความจำที่อยู่ภายในซีพียู (Local Memory) นอกจากนี้ภายในตัวหน่วยควบคุม (CU) ซึ่งเป็นส่วนหนึ่งของซีพียู ก็ต้องการมีหน่วยความจำเป็นของตัวเองด้วย ส่วนหน่วยความจำภายนอกหมายถึง อุปกรณ์เก็บข้อมูลอื่น ๆ ที่มักเรียกว่าเป็น Peripheral Storage Devices เช่น ดิสก์ เทป ซึ่งติดต่อกับซีพียูด้วย I/O Controller

หน่วยความจำหลัก แบ่งเป็น 2 ประเภทคือ หน่วยความจำชนิดรอม (Read Only Memory : ROM) เป็นหน่วยความจำชนิดที่จะเก็บข้อมูลหรือโปรแกรมไว้อย่างถาวร ไม่สามารถเปลี่ยนแปลงอะไรได้ไม่ว่าจะต้องการหรือไม่ ส่วนอีกประเภทหนึ่งคือ หน่วยความจำแรม (Random Access Memory : RAM) เป็นหน่วยความจำชนิดที่ข้อมูลจะถูกลบหรือหายไปเมื่อคอมพิวเตอร์ไม่มีกระแสไฟเลี้ยง

หน่วยความจำสำรอง หรือสื่อเก็บข้อมูลประเภทต่าง ๆ ซึ่งมีพื้นที่ในการจัดเก็บโปรแกรมและข้อมูลของคอมพิวเตอร์ เช่น แผ่นดิสก์ ฮาร์ดดิสก์ ซีดีรอม หรือ ROM/BIOS เป็นต้น ซึ่งจัดเป็นหน่วยความจำที่ใช้เก็บข้อมูลแบบถาวร (Permanent storage areas)

สถาปัตยกรรมของคอมพิวเตอร์โดยปกติแล้วจะจัดให้หน่วยความจำที่มีความเร็วสูงไว้ข้างบนใกล้ ๆ กับซีพียู โดยหน่วยความจำนี้ไม่จำเป็นต้องมีขนาดใหญ่มาก แต่ต้องมีความเร็วในการเข้าถึงข้อมูลสูง เนื่องจากใช้อัตราความเร็วในการเข้าถึงข้อมูลที่ความเร็วเดียวกับของซีพียู (clock rate)

ปัจจัยที่มีผลทำให้หน่วยความจำมีประสิทธิภาพและทำงานได้รวดเร็ว ได้แก่ ความเป็น locality ของโปรแกรม ความเร็วในการเข้าถึง (access time) ของหน่วยความจำแต่ละเลเวล ขนาดความจุของหน่วยความจำในแต่ละเลเวล ขนาดของบล็อกที่ใช้ทรานสเฟอร์ในแต่ละเลเวล และอัลกอริธึมหรือยุทธวิธีที่ใช้กำหนดตำแหน่ง หรือการแทนที่ในหน่วยความจำในแต่ละเลเวล

หน้าที่ของแคชคือ จดจำคำสั่งและผลลัพธ์ที่ถูกเรียกใช้บ่อย ๆ เพื่อไว้ใช้ในการประมวลผลในครั้งต่อ ๆ ไป เพื่อที่ซีพียูไม่ต้องเสียเวลารอเรียกคำสั่งนั้น ๆ อีก ถ้าแคชมีขนาดใหญ่ก็สามารถจดจำคำสั่งได้มาก แต่แคชนั้นไม่สามารถเก็บข้อมูลไว้ทั้งหมดได้เนื่องจากมีขนาดที่เล็ก ดังนั้นภายในแคชจึงเก็บเฉพาะข้อมูลและคำสั่งสำคัญ ๆ ที่ถูกเรียกใช้บ่อย ๆ เมื่อซีพียูต้องการข้อมูลนั้นก็สามารถเรียกใช้จากแคชได้ทันที แคชมีความเร็วในการทำงานสูงกว่าแรมมากและถูกสร้างให้อยู่ใกล้กับหน่วยประมวลผล จึงทำให้คอมพิวเตอร์สามารถประมวลผลได้เร็วขึ้น

หน่วยความจำเสมือนจะถูกแบ่งออกเป็นหน้า ๆ ซึ่งมีขนาดใหญ่กว่าบล็อกข้อมูลในแคชที่มีขนาดเพียงไม่กี่เวิร์ด แต่การคัดลอกบล็อกข้อมูลที่ใช้งานบ่อย ๆ นั้นจะถูกเก็บไว้ในแคช เมื่อมีการอ้างถึงข้อมูลที่อยู่ในแคชและหน่วยความจำเสมือนนั้นระบบจะทำการค้นหาแอดเดรสที่อ้างอิงในแคชก่อน ถ้าพบข้อมูลก็จะทำงานตามคำสั่งทันที แต่ถ้าไม่พบข้อมูลระบบจะทำการค้นหาข้อมูลนั้นในหน่วยความจำหลักเมื่อพบก็จะทำการคัดลอกเข้าสู่แคชและจึงส่งข้อมูลนี้ไปให้ซีพียูต่อไป แต่ถ้าข้อมูลนั้นไม่ได้อยู่ในหน่วยความจำหลักอีก หน้าของข้อมูลที่พบในหน่วยความจำเสมือนก็จะถูกสลับจากที่อยู่ในดิสก์เข้าสู่หน่วยความจำหลักแล้วส่งต่อไปยังแคชและซีพียู

คำถามทบทวนประจำบท

~~~~~~

คำสั่ง จงตอบคำถามต่อไปนี้ โดยอธิบายให้เข้าใจ

1. หน่วยความจำของคอมพิวเตอร์ที่ใช้ในการเก็บคำสั่งและข้อมูลในขณะที่ระบบคอมพิวเตอร์มีการประมวลผลมีกี่ประเภท อะไรบ้าง
2. หน่วยความจำหลักมีกี่ประเภท อะไรบ้าง และแต่ละประเภทมีคุณสมบัติหรือลักษณะที่สำคัญอย่างไร
3. หน่วยความจำสำรองมีคุณสมบัติหรือลักษณะที่สำคัญอย่างไร จงยกตัวอย่างมา 2 ชนิด พร้อมอธิบายคุณสมบัติหรือลักษณะที่สำคัญ
4. จงแสดงลำดับชั้นของหน่วยความจำแบบ 3 ระดับ พร้อมอธิบายให้เข้าใจ
5. จงอธิบายหน้าที่ และลักษณะที่สำคัญของหน่วยความจำแคช
6. จงอธิบายหน้าที่ และลักษณะที่สำคัญของหน่วยความจำเสมือน
~~~~~~