บทที่ 2

ระบบหน่วยความจำ

2.1 ส่วนประกอบพื้นฐานของคอมพิวเตอร์

ระบบคอมพิวเตอร์ในส่วนฮาร์ดแวร์(Hardware) ประกอบด้วยส่วนประกอบสามส่วนคือ หน่วยความจำ(Memory Unit),หน่วยประมวลผลกลาง(Central Processing Unit), และหน่วยรับ/ส่งข้อมูล(Input-Output Unit)(Wear,Pinkert,Wear,and Lane,1991,95-102) (Bartee,1985,13-14) เมื่อคอมพิวเตอร์ทำงานหน่วยความจำจะเป็นที่เก็บข่าวสารต่างๆที่ต้องใช้ในการประมวลผลได้แก่ ข้อมูล(Data)และคำสั่ง(Insturction) หน่วยรับ/ส่งข้อมูลจะเป็นส่วนที่คอมพิวเตอร์ติดต่อกับภายนอก และหน่วยประมวลผลกลางจะเป็นส่วนที่ดำเนินการประมวลผลข้อมูลตามชุดของคำสั่งที่เรียกว่าโปรแกรม(Program) นอกจากหน่วยความจำจะเป็นที่เก็บข่าวสารต่างๆแล้ว ยังมีอุปกรณ์เก็บข้อมูล(Storage Device)เป็นที่เก็บข่าวสารต่างให้คอมพิวเตอร์ คอมพิวเตอร์จะติดต่อกับอุปกรณ์เก็บข้อมูลนี้ผ่านหน่วยรับ/ส่งข้อมูลที่ทำหน้าที่ควบคุมอุปกรณ์เก็บข้อมูลเฉพาะ อุปกรณ์เก็บข้อมูลนี้จะเรียกว่าหน่วยความจำรอง(Secondary Memory) ส่วนหน่วยความจำจะเรียกว่าหน่วยความจำหลัก(Main Memory)

รูปที่ 2.1 แสดงให้เห็นส่วนประกอบในคอมพิวเตอร์

ในขณะที่คอมพิวเตอร์ทำงาน จะมีการส่งผ่านข้อมูลไปมาระหว่างหน่วยประมวลผลกลาง หน่วยความจำหลักและหน่วยรับ/ส่งข้อมูล(รวมทั้งหน่วยความจำรองโดยส่งผ่านหน่วยรับ/ส่งข้อมูล) เป็นจำนวนมาก และประสิทธิภาพ(Performance)ของคอมพิวเตอร์จะดีขึ้นได้ถ้าการส่งผ่านข้อมูลเหล่านี้ดีขึ้น

2.2 หน่วยความจำ

ตามที่กล่าวมา หน่วยความจำเป็นส่วนประกอบทางฮาร์ดแวร์ของคอมพิวเตอร์ หน่วยความจำเป็นส่วนที่มีหน้าที่เก็บข่าวสารต่างๆที่ต้องใช้ในการประมวลผล หน่วยความจำในเครื่องคอมพิวเตอร์แบ่งออกได้เป็น 3 กลุ่ม(Hayes,1988,376-377)คือ

1. หน่วยความจำภายในตัวประมวลผล เป็นหน่วยความจำที่ประกอบอยู่ภายในตัวประมวลผล หน่วยความจำประเภทนี้ทำงานเร็วเท่ากับการทำงานของตัวประมวลผล แต่มีความจุข้อมูลไม่มากนัก หน่วยความจำในตัวประมวลผลได้แก่ รีจิสเตอร์ต่างๆ

2. หน่วยความจำหลัก เป็นหน่วยความจำที่อยู่ภายนอกตัวประมวลผลกลาง ทำงานได้เร็วใกล้เคียงกับการทำงานของตัวประมวลผลกลาง หน่วยความจำหลักมีความจุข้อมูลมากกว่าหน่วยความจำในตัวประมวลผล ตัวประมวลผลกลางสามารถอ่าน/เขียนข้อมูลในหน่วยความจำหลักได้โดยตรง หน่วยความจำหลักทั่วไปได้แก่ หน่วยความจำ RAM (Random Access Memory)

3. หน่วยความจำรอง เป็นหน่วยความจำที่อยู่ภายนอกตัวประมวลผลกลาง และเป็นหน่วยความจำที่มีความจุมากที่สุดในระบบคอมพิวเตอร์ ตัวประมวลผลกลางไม่สามารถเข้าถึงข้อมูลในหน่วยความจำรองได้โดยตรง ต้องเข้าถึงข้อมูลผ่านอุปกรณ์รับ/ส่งข้อมูล หน่วยความจำรองทั่วๆไปได้แก่ หน่วยความจำ DASD(Direct Access Storage Devices) เช่น ดิสค์(Disk)

ในระหว่างการทำงานของกระบวนการ(Process)หน่วยความจำทั้งสามกลุ่มถูกใช้งานในการเก็บข่าวสารต่างๆ หน่วยความจำภายในตัวประมวลผลถูกใช้เก็บข้อมูลที่เกิดจากการดำเนินการแต่ละคำสั่ง หน่วยความจำหลักใช้เก็บข่าวสารที่กระบวนการใช้ในการดำเนินการได้แก่โปร

แกรมและข้อมูลต่างๆที่กระบวนการต้องใช้ในขณะนั้น ส่วนหน่วยความรองใช้เก็บข่าวสารต่างๆที่ไม่ได้ใช้ในการดำเนินการของกระบวนการในขณะนั้น กระบวนการจะใช้ข่าวสารในหน่วยความจำรองได้ โดยดำเนินการบรรจุข่าวสารดังกล่าวลงหน่วยความจำหลักด้วยการขอบริการจากระบบปฏิบัติการ

2.3 หน่วยความจำลำดับชั้น

หน่วยความจำลำดับชั้น(Memory Hierarchy) เป็นระบบจัดหน่วยความจำที่ใช้หน่วยความจำหลายๆชนิดที่มีเวลาเข้าถึงข้อมูลและราคาต้นทุนต่อหนึ่งหน่วยความจำต่างกัน มาประกอบกันเป็นลำดับชั้นเพื่อให้เวลาเข้าถึงข้อมูลและราคาต้นทุนต่อหนึ่งหน่วยความจำเฉลี่ยของระบบหน่วยความจำเหมาะสม(Hayes,1988,404-405)(Wear...[etal.],1991,157-159)

รูปที่ 2.2 แสดงให้เห็นภาพของระบบหน่วยความจำลำดับชั้น

ในหน่วยความจำลำดับชั้น ลำดับชั้นที่ของหน่วยความจำ($M_1, M_2, \ldots, M_n$) โดย M_{i+1} เป็นลำดับชั้นที่อยู่รองจาก M_i ตัวประมวลผลกลางจะอ่าน/เขียนข้อมูลในหน่วยความจำในลำดับชั้น M_1 ได้โดยตรง หน่วยความจำ M_1 จะติดต่อโอนย้ายข้อมูลกับหน่วยความจำในลำดับชั้น M_2 และต่อๆไป หน่วยความจำหลายชนิดที่นำมาประกอบเป็นหน่วยความจำลำดับชั้นโดยที่ความจุข้อมูลของหน่วยความจำลำดับชั้น M_i น้อยกว่าความจุข้อมูลของหน่วยความจำลำดับชั้น M_{i+1} เวลาเข้าถึงข้อมูลของหน่วยความจำลำดับชั้น M_i น้อยกว่าหน่วยความจำลำดับ M_{i+1} และราคาต้นทุนต่อหนึ่งหน่วยความจำของหน่วยความจำลำดับชั้น M_i สูงหน่วยความจำลำดับ M_{i+1}

ในระหว่างการทำงานของกระบวนการ ตัวประมวลผลกลางจะเข้าถึงข้อมูลในหน่วยความจำต่อเนื่องกันไป ข้อมูลนี้กระจายไปตลอดทุกลำดับชั้นของหน่วยความจำ ถ้าข้อมูลนี้อยู่ในหน่วยความจำลำดับชั้น M_i โดยที่ $i > 1$ ข้อมูลนี้จะต้องโอนลงในหน่วยความจำในลำดับชั้น M_1 ซึ่งเป็นหน่วยความจำลำดับชั้นที่ตัวประมวลผลกลางติดต่อโดยตรงได้ ดังนั้นหน่วยความจำลำดับชั้นจะทำงานได้อย่างมีประสิทธิภาพเมื่อข้อมูลที่ตัวประมวลผลกลางต้องการควรจะอยู่ในหน่วยความจำลำดับชั้น M_1 มากที่สุดและถูกใช้บ่อยที่สุด

เพื่อให้ข้อมูลที่ตัวประมวลผลกลางต้องการอยู่ในหน่วยความจำลำดับชั้นที่ M_1 เป็นหน่วยความจำลำดับชั้นที่ตัวประมวลผลกลางติดต่อโดยตรงได้ จำเป็นต้องคาดคะเนความต้องการข้อมูลของตัวประมวลผลกลางล่วงหน้าได้เพื่อจะได้โอนย้ายข้อมูลดังกล่าวมาไว้ในหน่วยความจำลำดับชั้น M_1 ก่อนที่ตัวประมวลผลกลางต้องการ การสามารถคาดคะเนความต้องการข้อมูลของตัวประมวลกลางเพื่อให้หน่วยความจำลำดับชั้นทำงานได้สมบูรณ์ขึ้นอยู่กับคุณสมบัติของข้อมูลคุณสมบัติหนึ่งที่เรียกว่า Locality of Reference กล่าวคือ ในช่วงเวลาหนึ่งข้อมูลที่ตัวประมวลผลกลางต้องการมี Logical Address Space เกาะกลุ่มในบริเวณเดียวกัน(Hayes,1988,405-406) เหตุที่ข้อมูลดังกล่าวมี Logical Address Space เกาะกลุ่มในบริเวณเดียวกันเพราะข้อมูลที่ตัวประมวลผลกลางต้องการถูกเก็บเรียงลำดับกันในหน่วยความจำ ดังนั้นการโอนย้ายมาเป็นบล็อก(Block) แทนที่จะโอนย้ายมาเฉพาะข้อมูลหนึ่งๆ ทำให้มีข้อมูลต่อไปที่ตัวประมวลผลกลางต้องการติดมาด้วย Locality of Reference ยังเกิดจากการวน(Loop)ในโปรแกรมได้ด้วย คำสั่งเป็นกลุ่มจะถูกทำงานซ้ำๆทำให้ที่อยู่ของคำสั่งถูกอ้างอิงบ่อยครั้งเมื่อโปรแกรมวนทำงาน

จุดมุ่งหมายในการออกแบบหน่วยความจำลำดับชั้น เพื่อให้ได้หน่วยความจำลำดับชั้นทำงานได้เร็วใกล้เคียงกับหน่วยความจำที่ทำงานได้เร็วที่สุดในลำดับชั้น M_1 และราคาต้นทุนต่อหนึ่งหน่วยความจำ ใกล้เคียงหน่วยความจำที่มีราคาต้นทุนต่อหนึ่งหน่วยความจำที่ถูกที่สุดในลำดับชั้น M_n (Hayes,1988,406-407) ในหน่วยความจำลำดับชั้นที่มีเพียง 2 ลำดับชั้น M_1 และ M_2 ถ้าให้ S_1 และ C_1 เป็นความจุข้อมูลและราคาต้นทุนต่อหนึ่งหน่วยความจำของหน่วยความจำลำดับชั้น M_1 ให้ S_2 และ C_2 เป็นความจุข้อมูลและราคาต้นทุนต่อหนึ่งหน่วยความจำของหน่วยความจำลำดับชั้น M_2 ราคาต้นทุนต่อหนึ่งหน่วยความจำเฉลี่ยของหน่วยความจำลำดับชั้นคือ

$$C = \frac{C_1S_1 + C_2S_2}{S_1 + S_2} \qquad (2.1)$$

การออกแบบหน่วยความจำลำดับชั้นให้ราคาต้นทุนต่อหนึ่งหน่วยความจำเฉลี่ย C ของหน่วยความจำลำดับชั้นใกล้เคียงกับราคาต้นทุนต่อหนึ่งหน่วยความจำ C_2 ความจุของหน่วยความจำ S_1 ต้องมีขนาดเล็กเมื่อเทียบกับความจุของหน่วยความจำ S_2 และความจุของหน่วยความจำลำดับชั้นที่ตัวประมวลผลกลางเข้าถึงได้เท่ากับความจุของหน่วยความจำ S_2

ค่า Hit Ratio คือความน่าจะเป็นที่ข่าวสารที่ตัวประมวลผลกลางต้องการอยู่ในหน่วยความจำลำดับชั้นที่ M_1 (Hayes,1988,407-410)ค่า Hit Ratio ได้จากการนับในการทดสอบการทำงานด้วยโปรแกรมทดสอบหรือจากการจำลองแบบ(Simulation)ถ้าให้ N_1 และ N_2 เป็นจำนวนครั้งที่ตัวประมวลผลอ้างอิงข่าวสารในหน่วยความจำลำดับชั้น M_1 และ M_2 ค่า Hit Ratio จะได้จาก

$$H = \frac{N_1}{N_1 + N_2} \qquad (2.2)$$

ค่า Hit Ratio นี้ขึ้นอยู่กับโปรแกรมที่ทำงานอย่างมาก ถ้าให้ t_{m1} และ t_{m2} เป็นเวลาเข้าถึงของหน่วยความจำลำดับชั้น M_1 และ M_2 จากตัวประมวลผลกลาง เวลาเข้าถึงข้อมูลของหน่วยความจำลำดับชั้นเฉลี่ย t_m เมื่อตัวประมวลผลกลางต้องการข้อมูลคือ

$$t_m = Ht_{m1} + (1 - H)t_{m2} \qquad (2.3)$$

ในหน่วยความจำลำดับชั้นแบบสองลำดับชั้นทั่วๆไปมีหน่วยความจำในลำดับชั้น M_1 เป็นหน่วยความจำ RAM และมีหน่วยความจำในลำดับชั้น M_2 เป็นหน่วยความจำ DASD เมื่อข่าวสารที่ตัวประมวลผลต้องการไม่อยู่ในหน่วยความจำลำดับชั้น M_1 เป็นผลให้มีการโอนย้ายบล็อกของข่าวสารที่ตัวประมวลผลต้องการจากหน่วยความจำลำดับชั้น M_2 ไว้ในหน่วยความจำลำดับชั้น M_1 เวลาที่ใช้ในการโอนย้ายบล็อกของข่าวสารเรียกว่า "Block-Transfer Time" ถ้า t_b เป็นเวลาที่ใช้ในการโอนย้ายบล็อกของข่าวสารดังกล่าว เวลาเข้าถึงของหน่วยความจำลำดับชั้น M_2 จากตัวประมวลผล $t_{m2} = t_b + t_{m1}$ ดังนั้นเวลาเข้าถึงของหน่วยความจำลำดับชั้น t_m เป็น

$$t_m = t_{m1} + (1-H)t_b \qquad (2.4)$$

การโอนย้ายบล็อกของข่าวสารจากหน่วยความจำลำดับชั้น M_2 ที่เป็นหน่วยความจำแบบ DASDs เป็นการทำงานของอุปกรณ์รับ/ส่งข้อมูล เวลาที่ใช้ในการโอนย้ายบล็อกของข่าวสาร (t_b) โดยทั่วๆไป มีค่ามากกว่าเวลาเข้าถึงของหน่วยความจำลำดับชั้น M_1 (t_{m1}) เป็นหน่วยความจำ RAM ดังนั้น $t_{m2} >> t_{m1}$ ถ้าให้ $r = t_{m2}/t_{m1}$ เป็นอัตราส่วนระหว่างเวลาเข้าถึงของหน่วยความจำทั้ง 2 ลำดับชั้น ให้ $e = t_{m1}/t_m$ เรียกว่า "Access Efficiency" ของหน่วยความจำลำดับชั้น จะได้

$$e = \frac{1}{r + (1-r)H} \qquad (2.5)$$

2.4 การจัดสรรหน่วยความจำ

การทำงานของกระบวนการ จำเป็นต้องใช้หน่วยความจำเพื่อเก็บข่าวสารได้แก่ข้อมูลและคำสั่งต่างๆ ตำแหน่งที่อยู่ทางตรรก(Loigcal Address)หรือตำแหน่งที่อยู่ที่อ้างอิงในโปรแกรมจะถูกเปลี่ยนเป็นตำแหน่งที่อยู่จริง(Real Address) ซึ่งเป็นตำแหน่งที่อยู่ในหน่วยความจำหลัก การกำหนดตำแหน่งที่อยู่จริงให้กระบวนการแทนตำแหน่งที่อยู่ทางตรรกเรียกว่า การจัดสรรหน่วยความจำ(Memory Allocation)(Hayes,1988,417-419) การกำหนดตำแหน่งที่อยู่ในหน่วยความจำจริงอาจทำในขั้นตอนใดขั้นตอนหนึ่งหรือหลายๆขั้นตอนดังต่อไปนี้

1. กำหนดโดยโปรแกรมเมอร์ในขณะเขียนโปรแกรม
2. กำหนดโดยตัวแปลภาษา (Compiler) ในระหว่างการแปลโปรแกรม
3. กำหนดโดยตัวบรรจุ (Loader) ในระหว่างบรรจุโปรแกรมลงหน่วยความจำก่อนที่กระบวนการจะเริ่มทำงาน
4. กำหนดโดยโปรแกรมปฏิบัติการระหว่างที่กระบวนการกำลังทำงาน

การจัดสรรหน่วยความจำอาจเป็นการจัดสรรหน่วยความจำแบบตายตัวที่กำหนดตำแหน่งที่อยู่จริงให้กับกระบวนการในโปรแกรมตายตัว หรืออาจเป็นการจัดสรรหน่วยความจำที่กำหนดตำแหน่งที่อยู่ให้กับกระบวนการขณะดำเนินการ การจัดสรรหน่วยความจำแบบแรกเรียกว่าการจัดสรรหน่วยความจำแบบสถิต(Static Memory Allocation) และการจัดสรรหน่วยความจำแบบหลังเรียกว่าการจัดสรรหน่วยความจำแบบพลวัต(Dynamic Memory Allocation)

การจัดสรรแบบพลวัตยังแบ่งเป็นแบบ Preemptive และ Nonpreemtive ในการจัดสรรแบบ Preemptive หน่วยความจำหลักที่กำหนดให้กับกระบวนการที่กำลังดำเนินการอยู่จะไม่เปลี่ยนแปลงจนกว่ากระบวนการดำเนินการไปจนสิ้นสุด การจัดสรรแบบ Nonpreemptive หน่วยความจำหลักที่กำหนดให้กระบวนการขณะดำเนินการอยู่ อาจเปลี่ยนแปลงได้เพื่อให้มีที่ว่างในหน่วยความจำพอสำหรับจัดสรรให้กับกระบวนการอื่นๆดำเนินการได้ การเปลี่ยนแปลงอาจเป็นการย้ายข่าวสารของโปรแกรมจากตำแหน่งที่อยู่หนึ่งไปตำแหน่งที่อยู่อื่นเรียกว่า Compaction หรือเป็นการย้ายข่าวสารไปเก็บไว้ในหน่วยความจำสำรองเรียกว่า Replacement หรือ Swapping

2.5 หน่วยความจำเสมือน

หน่วยความจำเสมือน(Virtual Memory)เป็นหน่วยความจำลำดับชั้นที่มีลำดับชั้นอย่างน้อยสองชั้น(Hayes,1988,404) หน่วยความจำเสมือนส่วนใหญ่เป็นหน่วยความจำลำดับชั้นสองลำดับชั้น มีหน่วยความจำหลักเป็นหน่วยความจำในลำดับชั้น M_1 มีหน่วยความจำรองเป็นหน่วยความจำในลำดับชั้น M_2 หน่วยความจำเสมือนมีจุดประสงค์เพื่อ

ก. ช่วยให้โปรแกรมเมอร์มีอิสสระจากการจัดสรรหน่วยความจำ

ข. ช่วยให้โปรแกรมเป็นอิสสระจากรูปลักษณ์(Configuration)และความจุของหน่วยความจำในระบบ

ค. ช่วยให้เกิดการเข้าถึงข้อมูลในหน่วยความจำหลักอยู่ในระดับสูง

ง. ช่วยให้ที่ว่างในหน่วยความจำหลักถูกใช้งานได้อย่างมีประสิทธิภาพ

การทำงานของหน่วยความจำเสมือนจะถูกควบคุมโดยโปรแกรมปฏิบัติการ วิธีการของหน่วยความจำเสมือนคือการแยกการอ้างอิงตำแหน่งที่อยู่จากกระบวนการที่กำลังดำเนินการอยู่ออกจากตำแหน่งที่อยู่ในหน่วยความจำหลัก ตำแหน่งที่อยู่ที่ถูกอ้างอิงจากกระบวนการเรียกว่าตำแหน่งที่อยู่เสมือน(Virtual Address) ตำแหน่งที่อยู่ในหน่วยความจำหลักเรียกว่าตำแหน่งที่อยู่จริง (Real Address)

การเข้าถึงข้อมูลในหน่วยความจำเสมือนนั้น หน่วยความจำเสมือนจะมีกลไกการแปลงตำแหน่งที่อยู่ จากตำแหน่งที่อยู่เสมือนที่ได้จากตัวประมวลผลกลาง มาเป็นตำแหน่งที่อยู่จริงอยู่ในหน่วยความจำหลัก เรียกว่าการแปลงตำแหน่งที่อยู่(Address Mapping) สามารถเขียนเป็นฟังก์ชันนามธรรม(abstractly) f:L->P โดยที่ L เป็นเซ็ตของตำแหน่งที่อยู่เสมือน และ P เป็นเซ็ตของตำแหน่งที่อยู่จริง

ค่า Block Hit Ratio เป็นค่าที่แสดงให้เห็นถึง การทำงาน(Performance)ของหน่วยความจำเสมือน ค่า Block Hit Ratio(H^*) หาได้โดยการนับการอ้างอิงตามชุดตำแหน่งที่อยู่บล็อกถ้าให้ N_1^* และ N_2^* เป็นจำนวนครั้งที่อ้างอิงตำแหน่งที่อยู่บล็อกในหน่วยความจำหลัก ซึ่ง

เป็นหน่วยความจำลำดับชั้น M_1 และจำนวนครั้งที่อ้างอิงตำแหน่งที่อยู่บล็อกในหน่วยความจำรองซึ่งเป็นหน่วยความจำลำดับชั้น M_2 ค่า Block Hit Ratio H^* จะมีค่าเป็น

$$H^* = \frac{N_1^*}{N_1^* + N_2^*} \qquad (2.6)$$

ถ้าให้ n^* จำนวนค่าเฉลี่ยที่ถูกอ้างอิงในแต่ละบล็อก ค่า Hit Ratio(H) คือ

$$H = 1 - \frac{1 - H^*}{n^*} \qquad (2.7)$$

2.6 หน่วยความจำแคช

หน่วยความจำหลักโดยทั่วไปเป็นหน่วยความจำ RAM แบบ MOS มีเวลาเข้าถึงค่อนข้างมากเมื่อต้องทำงานติดต่อกับตัวประมวลผลข้อมูลโดยตรงทำให้ตัวประมวลผลข้อมูลต้องหยุดรอหน่วยความจำหลัก เมื่อเปลี่ยนเป็นหน่วยความจำ RAM แบบ Bipolar ที่มีเวลาเข้าถึงน้อยกว่าหน่วยความจำ RAM แบบ Bipolar จะมีขนาดใหญ่และราคาสูงกว่า ทางหนึ่งที่เป็นไปได้คือเพิ่มหน่วยความจำลำดับชั้นขั้นอีกชั้นหนึ่งระหว่างหน่วยความจำหลักกับตัวประมวลผลกลางเรียกว่าหน่วยความจำแคช (Cache Memory)(Hayes,1988,442)

หน่วยความจำแคชเป็นหน่วยความจำ RAM แบบ Bipolar, HMOS หรือ ECL มีเวลาเข้าถึงน้อยกว่าหน่วยความจำหลักที่เป็นหน่วยความจำ RAM แบบ MOS หน้าที่หลักของหน่วยความจำแคชคือ เก็บข้อมูลที่ตัวประมวลผลกลางใช้บ่อยๆไว้เพื่อให้ตัวประมวลผลกลางอ่านเขียนข้อมูลได้รวดเร็ว

ในหน่วยความจำแคชจะมีตัวควบคุมแคชคอยตรวจสอบว่าข้อมูลที่ตัวประมวลผลกลางต้องการมีอยู่ในหน่วยความจำแคชหรือไม่ ถ้ามีอยู่ในหน่วยความจำแคช ตัวควบคุมแคชจะส่งข้อมูลนั้นให้ตัวประมวลผลกลาง แต่ถ้าข้อมูลที่ตัวประมวลผลกลางต้องการไม่มีอยู่ในหน่วยความจำแคช ตัวควบคุมแคชจะติดต่อกับหน่วยความจำหลัก เพื่ออ่านข้อมูลในหน่วยความจำหลักที่ตัวประมวลผลกลางต้องการมาไว้ที่หน่วยความจำแคช และส่งข้อมูลให้ตัวประมวลผลกลาง

2.7 บัฟเฟอร์แคช

ในระบบหน่วยความจำ หน่วยความจำรองโดยทั่วไปเป็นหน่วยความจำ DASDs ที่หน่วยประมวลผลกลางไม่สามารถเข้าถึงข้อมูลได้ตรง เมื่อกระบวนการต้องการข้อมูลกระบวนการต้องโอนข้อมูลจากหน่วยความจำรองมาไว้ในหน่วยความจำหลักก่อนแล้วจึงอ่านข้อมูลไปจากหน่วยความจำหลัก การโอนข้อมูลจากหน่วยความจำรองมาไว้ในหน่วยความจำหลักจะโอนมาเป็นบล็อก แต่ละบล็อกมีขนาดเท่ากัน หน่วยความจำหลักที่มารองรับข้อมูลจากหน่วยความจำรองเรียกว่าบัฟเฟอร์ (Buffer) และขั้นตอนการเข้าถึงข้อมูลดังกล่าวเป็นขั้นตอนในระบบปฏิบัติการที่ให้บริการต่อกระบวนการ

เนื่องจากหน่วยความจำหลักเป็นหน่วยความจำ RAM มีเวลาเข้าถึงข้อมูลน้อยกว่าหน่วยความจำรองซึ่งเป็นหน่วยความจำ DASD ดังนั้นการเข้าถึงข้อมูลที่มีอยู่ในบัฟเฟอร์แล้วจะเร็วกว่าการเข้าถึงข้อมูลที่ไม่อยู่ในบัฟเฟอร์ และถ้าเพิ่มจำนวนบัฟเฟอร์ขึ้น โอกาสที่ข้อมูลที่ต้องการพบในบัฟเฟอร์มีมากขึ้น จำนวนครั้งที่ต้องเข้าถึงข้อมูลในหน่วยความจำรองลดลงทำให้เวลาในการเข้าถึงข้อมูลเฉลี่ยลดลง วิธีการใช้บัฟเฟอร์จำนวนมากมาช่วยในการโอนย้ายข้อมูลเพื่อลดจำนวนครั้งในการเข้าถึงข้อมูลในหน่วยความจำรอง เรียกว่าบัฟเฟอร์แคช(Buffer Cache)(Wilkinson, 1991,94-95) สำหรับเครื่องไมโครคอมพิวเตอร์ หน่วยความจำรองโดยทั่วไปเป็น Movable Arm Disk ดังนั้นบางครั้งบัฟเฟอร์แคชในไมโครคอมพิวเตอร์จะเรียกว่าดิสค์แคช(Disk Cache)

บัฟเฟอร์แคชจะมีโปรแกรมควบคุมแคชคอยควบคุมการทำงาน ถ้ามีการอ่านข้อมูลจากหน่วยความจำรอง โปรแกรมควบคุมแคชจะค้นหาข้อมูลที่ต้องการในบัฟเฟอร์ก่อน หากมีบล็อกใดที่มีข้อมูลที่ต้องการ โปรแกรมควบคุมแคชจะส่งข้อมูลนั้นให้ หากไม่มีข้อมูลที่ต้องการ โปรแกรมควบคุมแคชจะอ่านข้อมูลจากหน่วยความจำรองมาเก็บในบัฟเฟอร์บล็อกและส่งข้อมูลนั้นให้

เมื่อมีการเขียนข้อมูลลงในหน่วยความจำรอง โปรแกรมควบคุมแคชจัดการให้เราได้ 2 วิธีใหญ่ๆคือ(วสิน ทรัพย์เพิ่ม,2532,61-62)

1. Write-through เมื่อมีการเขียนข้อมูลลงหน่วยความจำรอง โปรแกรมควบคุมแคชจะเขียนลงในบัฟเฟอร์แคชและในหน่วยความจำรองเลยทำให้ข้อมูลทั้ง 2 ถูกต้องตรงกัน วิธีนี้มีข้อดีคือ ข้อมูลสุดท้ายที่บันทึกลงหน่วยความจำรองจะไม่สูญหาย บัฟเฟอร์แคชแบบนี้เมื่อเอาบล็อกใหม่จากหน่วยความจำรองเข้ามาในหน่วยความจำหลักจะทับบล็อกเก่าได้เลย แต่การเขียนข้อมูลลงหน่วยความจำรองไม่เร็วขึ้น

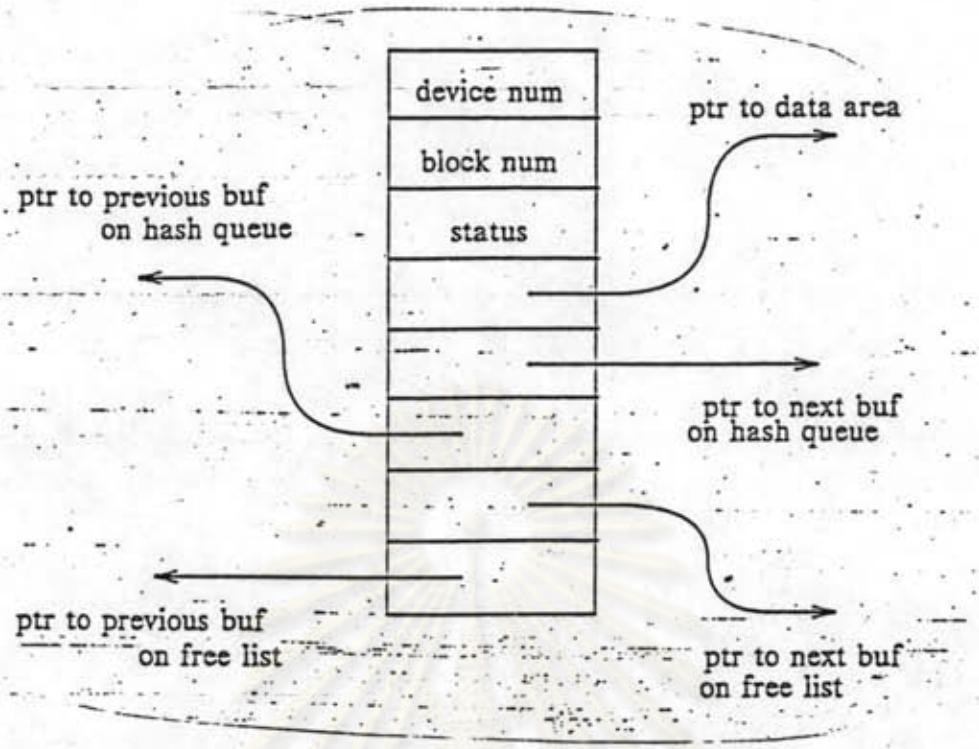
2. Write Cache เมื่อมีการเขียนข้อมูลลงหน่วยความจำรอง โปรแกรมควบคุมแคชจะเขียนลงเฉพาะในบัฟเฟอร์แคชเท่านั้น ไม่เขียนข้อมูลลงหน่วยความจำรองจนกว่าบัฟเฟอร์บล็อกนั้นถูกคัดเลือกให้เก็บข้อมูลบล็อกใหม่ ทำให้ต้องเขียนข้อมูลที่มีอยู่เดิมในหน่วยความจำรองก่อนถูกข้อมูลบล็อกใหม่เข้ามาแทนที่ บัฟเฟอร์แคชแบบนี้ทำให้การเขียนข้อมูลเร็วขึ้นได้เช่นเดียวกับการอ่านข้อมูล แต่ถ้าเกิดเหตุขัดข้องขึ้นข้อมูลที่ยังคงอยู่ในบัฟเฟอร์จะสูญหาย

หากเปรียบบัฟเฟอร์แคชกับหน่วยความจำแคชที่ได้กล่าวมาก่อน บัฟเฟอร์แคชมีวิธีการเหมือนหน่วยความจำแคชตรงที่ ใช้หน่วยความจำที่มีความเร็วในการทำงานสูงกว่ามาเก็บข้อมูลบางส่วนจากหน่วยความจำที่มีความเร็วในการทำงานต่ำไว้ ช่วยให้จำนวนครั้งในการเข้าถึงข้อมูลในหน่วยความจำที่มีความเร็วต่ำน้อยครั้งลง ในหน่วยความจำแคช หน่วยความจำที่มีความเร็วในการทำงานสูงคือ หน่วยความจำ RAM แบบ Bipolar ที่ลำดับชั้นของหน่วยความจำระหว่างตัวประมวลผลกลางกับหน่วยความจำหลัก ส่วนหน่วยความจำที่มีความเร็วในการทำงานต่ำคือ หน่วยความจำหลักซึ่งเป็นหน่วยความจำ RAM แบบ MOS ในบัฟเฟอร์แคช หน่วยความจำที่มีความเร็วในการทำงานสูงคือหน่วยความจำหลักซึ่งเป็นหน่วยความจำ RAM ส่วนหน่วยความจำที่มีความเร็วต่ำคือหน่วยความจำรองซึ่งเป็นหน่วยความจำ DASD ข้อมูลที่เก็บไว้ทั้งในหน่วยความจำแคชและในบัฟเฟอร์แคชเป็นบล็อกของข้อมูล ทั้งหน่วยความจำแคชและบัฟเฟอร์แคชจะมีส่วนหนึ่งที่ทำหน้าที่จัดการการใช้บล็อกข้อมูล เนื่องจากหน่วยความจำแคชเป็นหน่วยความจำที่ต้องทำงานร่วมกับตัวประมวลผลกลาง ส่วนที่ทำหน้าที่การจัดการการใช้บล็อกข้อมูลจะเป็น HardWare มีวิธีการจัดการที่ไม่ซับซ้อน สำหรับบัฟเฟอร์แคช ส่วนที่ทำหน้าที่จัดการการใช้บล็อกข้อมูลจะเป็นรูทีน(Routine) ซึ่งสามารถออกแบบให้มีการจัดการที่ซับซ้อนกว่าได้

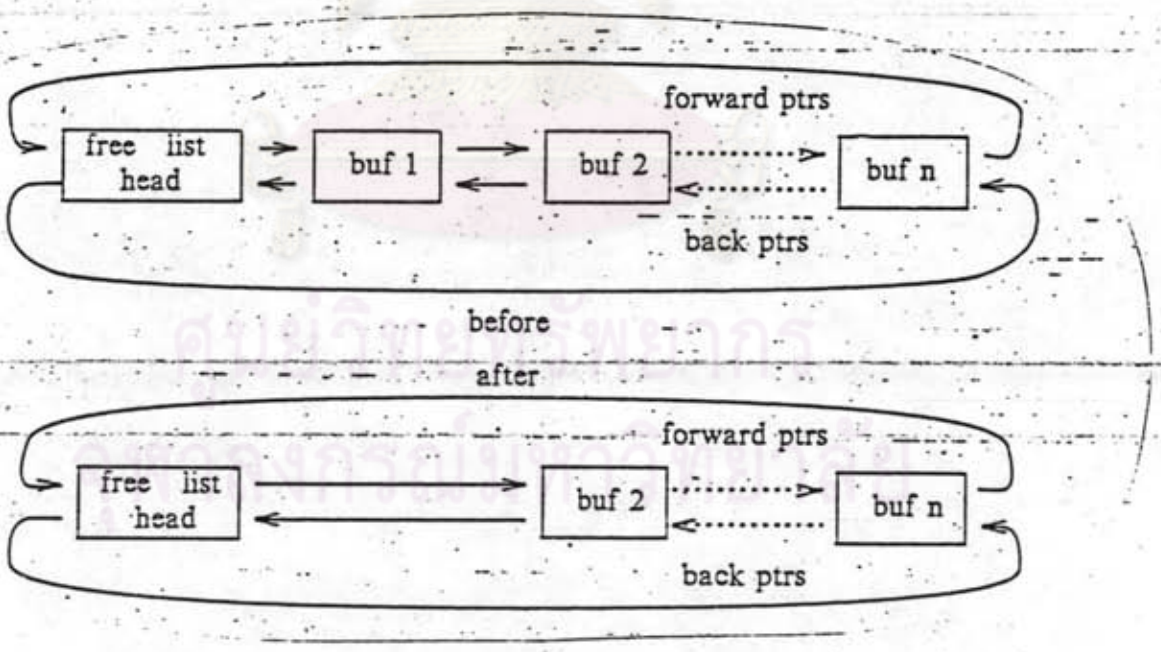
เมื่อเปรียบเทียบการทำงานระหว่างบัฟเฟอร์แคชกับหน่วยความจำเสมือน บัฟเฟอร์แคชเป็นหน่วยความจำลำดับชั้น เช่นเดียวกับหน่วยความจำเสมือน มีส่วนจัดการบัฟเฟอร์เป็นรูทีนเช่นเดียวกันกับส่วนจัดการการใช้หน่วยความจำในหน่วยความจำเสมือน แต่เนื่องจากบัฟเฟอร์แคชใช้วิธีการเก็บข้อมูลแบบบล็อกคงที่ทำให้รูทีนที่เป็นส่วนจัดการบัฟเฟอร์มีความซับซ้อนน้อยกว่า ข้อแตกต่างระหว่างหน่วยความจำเสมือนกับบัฟเฟอร์แคชคือ จุดมุ่งหมายของหน่วยความจำเสมือนเพื่อขยาย Logical Address Space ส่วนจุดมุ่งหมายของบัฟเฟอร์แคชเพื่อให้การเข้าถึงข้อมูลในหน่วยความจำรองของกระบวนการดีขึ้น

2.8 บัฟเฟอร์แคชในระบบปฏิบัติการ UNIX

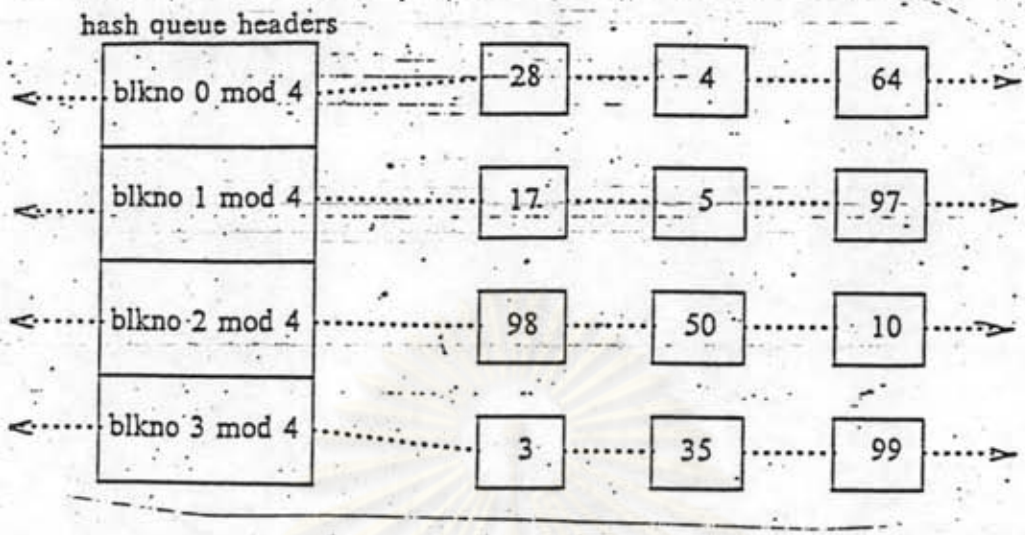
บัฟเฟอร์แคชในระบบปฏิบัติการ UNIX (Andleigh,1990,137-140)(Leffler... [etal.],1989)(Bach,1986,38-58) ช่วยให้จำนวนครั้งในการเข้าถึงดิสค์ลดลง บัฟเฟอร์ที่ใช้เก็บข้อมูลจากดิสค์จะจัดโครงสร้างข้อมูลแบบบัฟเฟอร์พูล(Buffer Pool) บัฟเฟอร์แต่ละบล็อกจะมีขนาดเท่ากับบล็อกของข้อมูลในดิสค์ มีบัฟเฟอร์เฮด(Buffer Head)เป็นที่เก็บรายละเอียดของข้อมูลได้แก่ เลขที่อุปกรณ์(Device Number), เลขที่บล็อก(Block Nummber), และตัวชี้(Pointer)ที่ชี้ไปที่บัฟเฟอร์ บัฟเฟอร์เฮดยังมีตัวชี้อีก 4 ชุด ตัวชี้ 2 ชุดมาจัดเป็นลิสค์เชื่อมโยง(Linked List)สำหรับเชื่อมโยงบัฟเฟอร์ที่ไม่ได้ถูกโปรแกรมใดใช้งาน ลิสค์เชื่อมโยงนี้เรียกว่า Free List กล่าวคือระบบปฏิบัติการ UNIX เป็นระบบปฏิบัติการแบบ Time Sharing เมื่อโปรแกรมใดต้องการข้อมูลในบัฟเฟอร์บล็อก ระบบปฏิบัติการจะนำบัฟเฟอร์บล็อกนั้นออกจาก Free List เพื่อไม่ให้โปรแกรมอื่นนำบัฟเฟอร์บล็อกนั้นไปใช้ ระบบปฏิบัติการจะนำบัฟเฟอร์บล็อกนั้นมาคืนไว้ที่หางของ Free List เสมอเมื่อโปรแกรมไม่ต้องการใช้บัฟเฟอร์นั้นอีก ดังนั้นส่วนหัวของ Free List จะเป็นบัฟเฟอร์บล็อกที่ไม่ได้ใช้งานมานานที่สุดเพราะไม่มีโอกาสได้ย้ายมาที่หางของลิสค์เลย ส่วนตัวชี้ในบัฟเฟอร์เฮดอีก 2 ชุดถูกใช้จัดเป็นลิสเชื่อมโยงเรียกว่า Hash Queue เพื่อใช้ในการค้นหาข้อมูลที่โปรแกรมต้องการในบัฟเฟอร์บล็อก



รูปที่ 2.3 บัฟเฟอร์เฮดของบัฟเฟอร์แคชในระบบปฏิบัติการ UNIX



รูปที่ 2.4 สภาพก่อนและหลังจากการนำบัฟเฟอร์บล็อก 1 ออกจาก Free List



รูปที่ 2.5 Hash Queues ที่ใช้ค้นหาข้อมูลในบัฟเฟอร์บล็อก