บทที่ 3

หลักการของพจนานุกรมเสมือน (Virtual Dictionary)

3.1 การสร้างพจนานุกรมเสมือน (Virtual Dictionary)

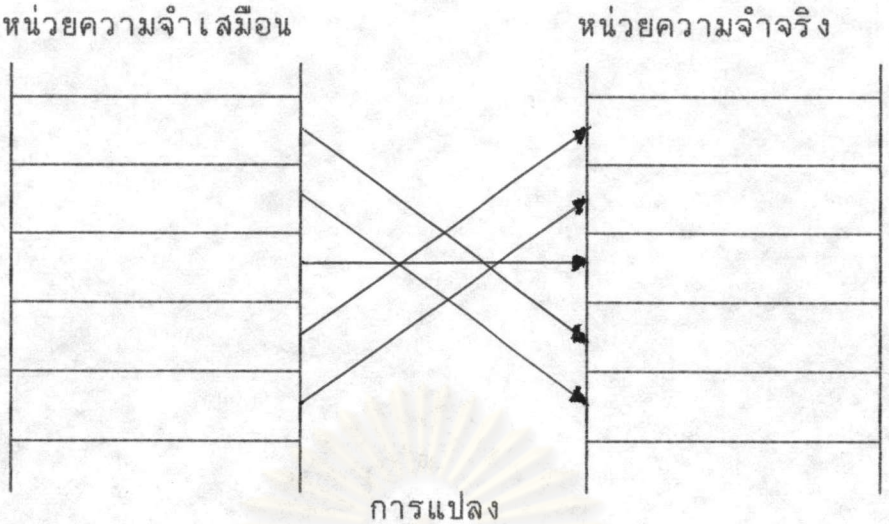
การสร้างพจนานุกรมเสมือน (virtual dictionary) มีหลักการเช่นเดียวกับหลักการสร้างหน่วยความจำเสมือน (Deitel and Harvey M. , 1990) เพียงแค่ปรับโครงสร้างให้ใช้กับระบบพจนานุกรมได้อย่างมีประสิทธิภาพ

เนื่องจากปัญหาจำนวนเนื้อที่ในหน่วยความจำหลักมีขนาดจำกัดเพื่อให้คอมพิวเตอร์สามารถทำงานกับเนื้อที่ ที่มีขนาดใหญ่กว่าหน่วยความจำหลักที่มีอยู่จริง ดังนั้นจึงได้มีการจำลองหน่วยความจำสำรอง (Secondary Storage) มาเป็นหน่วยความจำเสมือน

หน่วยความจำเสมือนถูกกำหนดให้มีลักษณะที่เหมาะสมกับการใช้งานและมีขนาดพอที่ผู้เขียนโปรแกรมจะไม่ต้องกังวลว่าข้อมูลจะใหญ่เกินไป จากนั้นเป็นหน้าที่ของผู้ออกแบบระบบที่จะกำหนดวิธีการแปลงหน่วยความจำเสมือนนี้ให้เหมาะสมกับหน่วยความจำจริงที่มีอยู่ การแยกหน่วยความจำเสมือนที่ใช้งาน ออกจากหน่วยความจำจริง ยังได้ประโยชน์อีกประการหนึ่งคือ สามารถออกแบบให้หน่วยความจำเสมือนมีขนาดใหญ่เพียงใดก็ได้

หัวใจของระบบหน่วยความจำเสมือน คือการแยก (disassociate) การอ้างอิงตำแหน่งที่ปรากฏในกระบวนการที่ดำเนินอยู่ ออกจากตำแหน่งที่มีให้อยู่ในหน่วยความจำหลัก ตำแหน่งที่ถูกอ้างอิงในกระบวนการที่ดำเนินอยู่ เรียกว่า ตำแหน่งเสมือน (virtual address) ส่วนตำแหน่งในหน่วยความจำหลักที่มีให้ไว้ในระบบ เรียกว่า ตำแหน่งจริง (real address)

แม้ว่ากระบวนการจะอ้างอิงแต่ตำแหน่งเสมือน ตามที่ปรากฏอยู่ในส่วนเนื้อความของโปรแกรม (program text) ของกระบวนการนั้น แต่กระบวนการต้องดำเนินอยู่ในระบบจริงดังนั้น ตำแหน่งเสมือนต้องถูกแปลงเป็นตำแหน่งจริงเมื่อตำแหน่งเสมือนนั้นถูกอ้างอิง ดังตัวอย่างในรูป 3.1 เป็นการแสดงความสัมพันธ์ ระหว่างตำแหน่งเสมือนกับตำแหน่งจริง นั่นคือทุกครั้งที่ตำแหน่งเสมือนถูกอ้างอิง ตำแหน่งเสมือนนั้นต้องผ่านการแปลง (mapping) ให้กลายเป็นตำแหน่งจริงเสียก่อน



รูปที่ 3.1 แสดงความสัมพันธ์ระหว่าง ตำแหน่งเสมือน และ ตำแหน่งจริง

หลักการสร้างหน่วยความจำเสมือนหน่วยความจำสำรองจะถูกแบ่งเป็นส่วนๆขนาดเท่ากันโดยภายในแต่ละส่วนเป็นเนื้อที่ที่ต่อเนื่องกันเรียกว่า เพจ (page) การประมวลผลในแต่ละครั้งมีการเรียกใช้ข้อมูลจากหน่วยความจำสำรองเพียง 2-3 เพจเท่านั้น โดยที่เพจดังกล่าวจะถูกนำมาไว้ในหน่วยความจำหลัก ณ เวลาใดเวลาหนึ่ง และมีวิธีการแปลงตำแหน่งเสมือน (Virtual Address) เป็นตำแหน่งที่ใช้งาน เรียกว่า Block Mapping ทำให้ตำแหน่งในหน่วยความจำเสมือนถูก แปลงเป็นตำแหน่งในหน่วยความจำหลัก

หลักการแปลงจากตำแหน่งเสมือนเป็นตำแหน่งที่ใช้งานจริง เมื่อพิจารณาตำแหน่งเสมือนประกอบด้วย

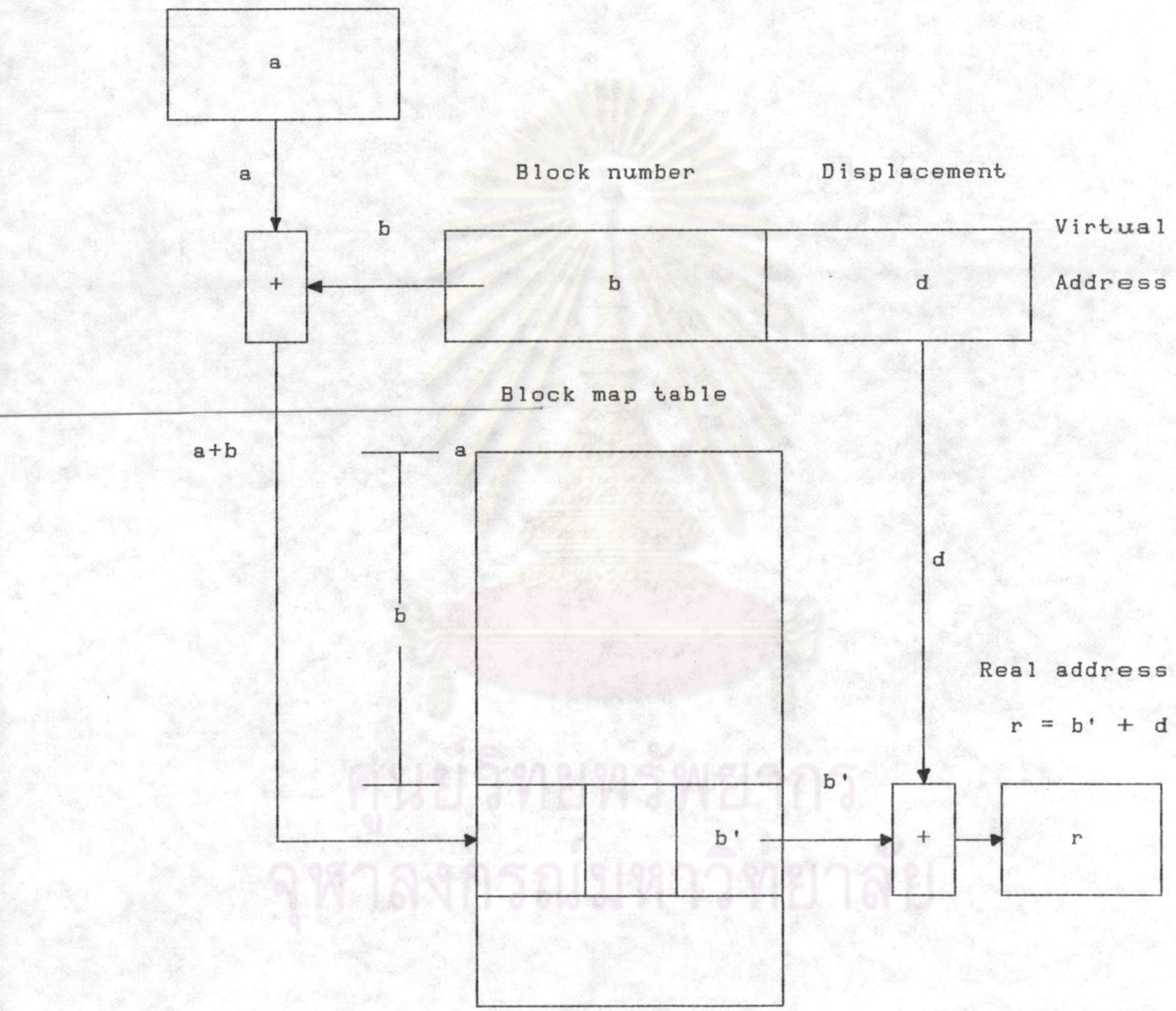
1. หมายเลขบล็อก (Block number : b)
2. ระยะ (Displacement : d)

Block number (b)	Displacement(d)

ตำแหน่งเสมือนอยู่ในรูปคู่อันดับของหมายเลขบล็อกและระยะ ซึ่งสามารถแสดงในรูปสมการ

Virtual address (v) = [b,d]

ตำแหน่งใน Block mapping แบ่งออกเป็น 2 มิติ เพื่อที่จะอ้างไปถึงตำแหน่งของข้อมูล การแปลงตำแหน่งข้อมูล จากตำแหน่งเสมือน เป็นตำแหน่งในหน่วยความจำหลักใช้งานจริง สามารถแสดงได้ดังรูปที่ 3.2



รูปที่ 3.2 การแปลงตำแหน่งข้อมูลจากตำแหน่งเสมือนเป็นตำแหน่งในหน่วยความจำหลัก

โดยที่ a เป็นตำแหน่งเริ่มต้นของ Block map table ในหน่วยความจำหลักเมื่อมีการอ้างถึงตำแหน่งใดๆ ตำแหน่งดังกล่าวจะเป็นตำแหน่งเสมือน ดังนั้นจึงจำเป็นต้องแปลงเป็นตำแหน่งที่ใช้งานจริงก่อน และถ้ามีการอ้างถึงหมายเลขบล็อก b และระยะ d

ตำแหน่งใน block map table สำหรับหมายเลขบล็อก b = a + b

b' = ตำแหน่งที่หน่วยความจำหลัก

ตำแหน่งที่ใช้งานจริง (r) = b' + d

3.2 เทคนิคการทำ Block Mapping มี 3 แบบ คือ

3.2.1. การแบ่งปล้อง (Segmentation)

3.2.2. การแบ่งเพจ (Paging)

3.2.3. การแบ่งเพจร่วมกับการแบ่งปล้อง (Combined paging segmentation)

3.2.1 การแบ่งปล้อง (Segmentation)

ระบบการแบ่งปล้อง(Segmentation) เป็นการจัดการหน่วยความจำหลักซึ่งพิจารณาถึงการใช้หน่วยความจำ ในแง่ของผู้ใช้เป็นหลัก

3.2.1.1 หลักการของการแบ่งปล้อง

เนื้อที่หน่วยความจำเสมือนจะถูกแบ่งออกเป็นปล้อง (segment) ซึ่งสมนัยกับการแบ่งงานออกเป็นหน่วยทางตรรก โดยแต่ละปล้องจะมีขนาดใหญ่เท่ากับ ขนาดของหน่วยของงานนั้นๆที่บรรจุอยู่ ปล้องหนึ่งๆอาจบรรจุโปรแกรมย่อยหนึ่งของงาน หรืออาจบรรจุข้อมูลชุดหนึ่งก็ได้

การอ้างอิงจุดใดๆในโปรแกรมหรือข้อมูล สามารถทำได้โดยอ้างถึง ชื่อของปล้องที่บรรจุจุดนั้น และตำแหน่งของจุดนั้นเทียบกับจุดเริ่มต้นของปล้อง ด้วยเหตุนี้

ปล้องต่างๆของข้อมูลไม่จำเป็นต้องอยู่เรียงต่อกันในหน่วยความจำจริง หรือหากปล้องไหนไม่ได้ใช้ ก็สามารถเก็บปล้องนั้นไว้ในหน่วยความจำรองได้ เพื่อให้ปล้องอื่นเข้าใช้ หน่วยความจำจริงได้

3.2.1.2 วิธีการของการแบ่งปล้อง

ในระบบแบ่งปล้องนี้ แต่ละตำแหน่งถูกอ้างอิงด้วย [s,d]

s คือ ชื่อ หรือ เลขหมายบ่งปล้อง

d คือ จุดที่ต้องการในปล้อง เทียบกับจุดเริ่มต้นของปล้อง หรือเรียกว่า เยื้อง (displacement or offset)

โดยมีตารางปล้องหนึ่งตารางสำหรับเก็บข้อมูลเกี่ยวกับแต่ละปล้องของงานโดยมีข้อมูลดังนี้

3.2.1.2.1 หมาย (flag) แสดงว่าปล้องนั้นอยู่ในหน่วยความจำหลักหรือไม่

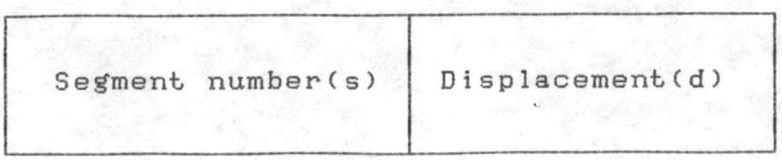
3.2.1.2.2 ตำแหน่งฐาน (base address) ของปล้อง

3.2.1.2.3 วิสัย (limit) ซึ่งแสดงถึงขนาดของปล้อง

3.2.1.2.4 บิตแสดงขอบเขตการใช้ปล้อง (protection bit)

3.2.1.2.5 ตำแหน่งในหน่วยความจำรองของปล้อง (secondary storage address)

ระบบปฏิบัติการใช้ตารางปล้องนี้เพื่อการแปลงตำแหน่งในหน่วยความจำเสมือนให้เป็นตำแหน่งในหน่วยความจำจริง ในการนี้จะมี รีจิสเตอร์ตัวหนึ่งเข้ามาช่วย เรียกว่า รีจิสเตอร์บ่งตารางปล้อง (segment table register)



Virtual address

v = [s,d]

ตำแหน่งเสมือนในระบบแบ่งปล้อง จะเป็นแบบคู่ลำดับ [s,d]

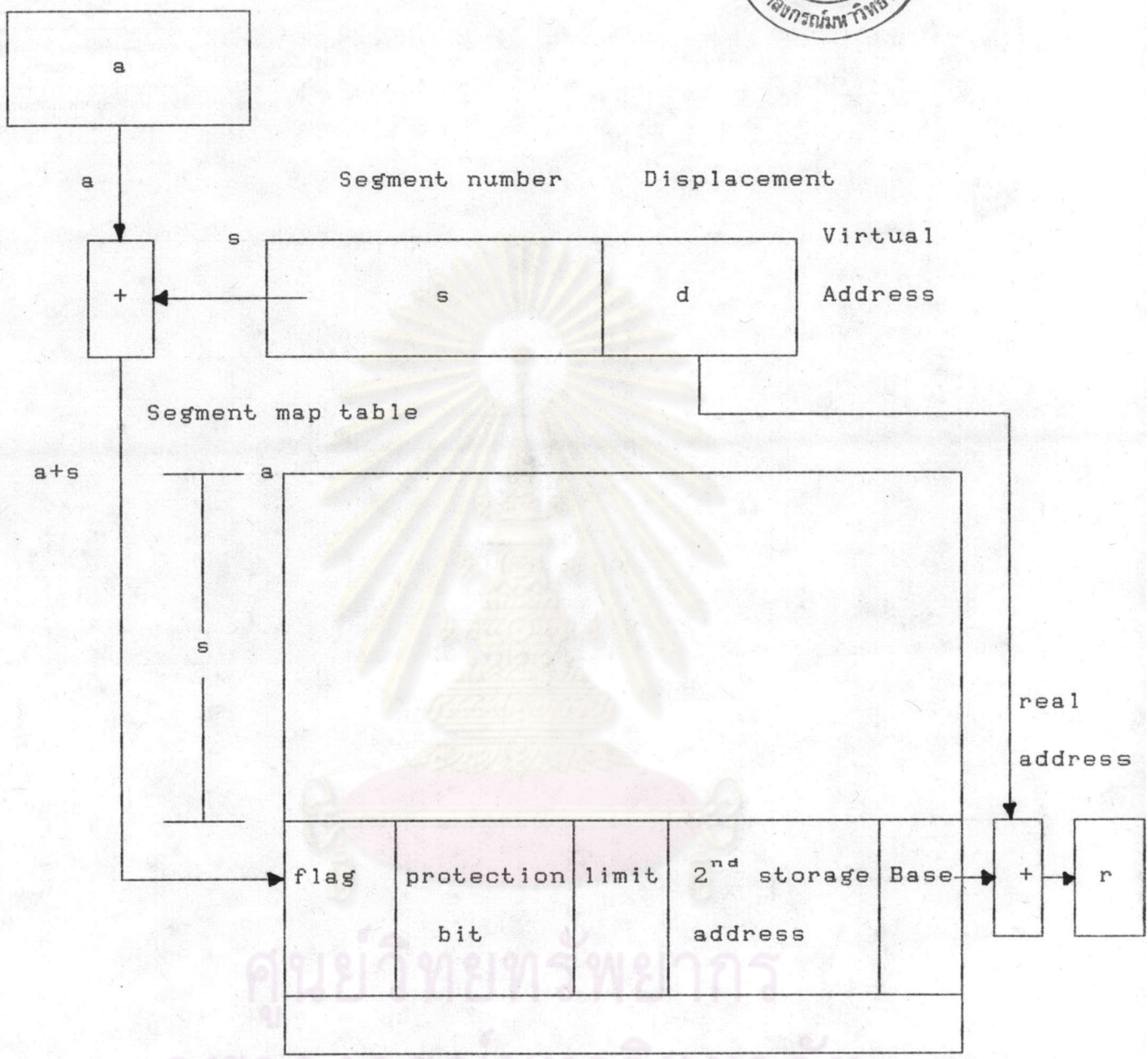
โดยที่ s คือ หมายเลขปล้อง ในหน่วยความจำเสมือน

d คือ ระยะในปล้อง s

ในการอ้างอิงจุด [s,d] ระบบจัดการหน่วยความจำหลักจะอาศัยค่าในรีจิสเตอร์บ่งตารางปล้อง (segment table register) เพื่อมุ่งไปยังตำแหน่งในหน่วยความจำหลักซึ่งบรรจุตารางปล้องของงานนี้ไว้ เมื่อพิจารณารายการที่ s ในตารางปล้องนั้นก็ทราบตำแหน่งฐานของปล้องได้ และสารสนเทศที่ต้องการ จะอยู่ที่ตำแหน่งเยื้อง d จากฐานของปล้องนั้น หลักการแปลงนี้แสดงไว้ในรูปที่ 3.3

เนื่องจากขนาดของปล้องนั้นไม่แน่นอน จึงจำเป็นต้องมีวิสัยเพื่อป้องกันมิให้โปรแกรมอ้างอิงตำแหน่งที่อยู่นอกขอบเขตของปล้อง และจะเห็นได้ว่าการอ้างอิงตำแหน่งใดๆในหน่วยความจำเสมือน ต้องทำการอ้างอิงหน่วยความจำจริง 2 ครั้ง คืออ้างอิงตารางปล้องแล้วจึงอ้างอิงสารสนเทศที่ต้องการอีกทีหนึ่ง

สำหรับ หมาย (flag) ในรายการในตารางปล้องนั้น จะถูกตรวจสอบก่อนที่จะมีการแปลงใดๆ เพื่อดูว่าปล้องนั้นๆ อยู่ในหน่วยความจำหลักหรือไม่ ถ้าไม่มีอยู่เรียกว่าเกิดขาดปล้อง(segment fault) และระบบปฏิบัติการต้องนำปล้องที่ขาดไปนั้นเข้ามาจาก หน่วยความจำรอง ซึ่งในการนำปล้องเข้ามานี้ ถ้าไม่มีเนื้อที่หน่วยความจำหลักเหลือมากพอ ระบบอาจต้องทำการย้ายปล้องที่ไม่ได้ใช้งานออกไปจากหน่วยความจำหลัก



รูปที่ 3.3 แสดงวิธีการของการแบ่งปล้อง

โดยที่ a เป็นตำแหน่งเริ่มต้นของ Segment map table ในหน่วยความจำหลักเมื่อมีการอ้างถึงตำแหน่งใดๆ ตำแหน่งดังกล่าวจะเป็นตำแหน่งเสมือน ดังนั้นจึงจำเป็นต้องแปลงเป็นตำแหน่งที่ใช้งานจริงก่อน และถ้ามีการอ้างถึงหมายเลขปล้อง s และระยะ d

ตำแหน่งใน segment map table สำหรับหมายเลขปล้อง

s' = a + s

โดย s' = ตำแหน่งที่หน่วยความจำหลัก

ตำแหน่งที่ใช้งานจริง (r) = s' + d

3.2.1.3 ข้อดีและข้อเสียของระบบแบ่งปล้อง

ข้อดี

การแบ่งปล้องนั้นเป็นไปตามการแบ่งทางตรรกของงาน ดังนั้นปล้องจึงถือได้ว่าเป็นหน่วยของการเชื่อมประกอบส่วนต่างๆ ของงานเข้าด้วยกันที่เรียกว่าการเชื่อมโยง (linking) ดังนั้นการเชื่อมโยงจึงสามารถทำได้โดยง่าย โดยกำหนดโปรแกรมผล เช่น โปรแกรมย่อยสำหรับหาค่า sine ให้เป็นปล้องหนึ่ง เนื่องจากการอ้างอิงตำแหน่งใดๆในโปรแกรมผลนี้ เป็นการอ้างอิงแบบโยกย้ายได้ทั้งหมด จึงเป็นการง่ายสำหรับโปรแกรมระบบในการเชื่อมหน่วยต่างๆ เหล่านี้เข้าด้วยกัน

ข้อเสีย

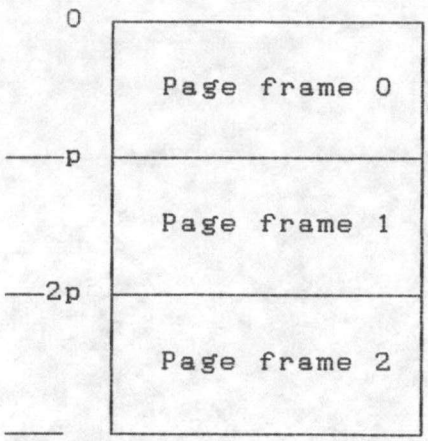
การที่ปล้องต่างๆมีขนาดไม่เท่ากัน และแต่ละปล้องต้องการเนื้อที่ที่ต่อเนื่องกัน ในหน่วยความจำจริง ทำให้เกิดปัญหาการจัดสรร ในลักษณะการจัดสรรแบบแปร ทำให้เกิดสภาพหน่วยความจำหลักขาดวิ่น

3.2.2 การแบ่งเพจ (Paging)

วิธีการจัดหน่วยความจำแบบแบ่งเพจนี้ เป็นวิธีการลดความซับซ้อนของการแปลงระหว่างหน่วยความจำเสมือน กับหน่วยความจำหลัก

3.2.2.1 หลักการของการแบ่งเพจ

วิธีการแบ่งเพจนี้หน่วยความจำเสมือนจะถูกแบ่งออกเป็นส่วนๆมีขนาดเท่ากันเรียกว่าเพจ(page) ส่วนหน่วยความจำจริงก็ถูกแบ่งออกเป็นส่วนๆ มีขนาดเท่ากับเพจเช่นเดียวกัน เรียกว่าเพจเฟรม (page frame) ดังนั้นโปรแกรมและข้อมูลที่บรรจุอยู่ในหน่วยความจำเสมือน จะถูกแบ่งออกเป็นเพจด้วย ดังนั้นสารสนเทศแต่ละเพจจะบรรจุลงใน เพจเฟรมได้พอดีเมื่อมีการเรียกใช้เพจที่ไม่ได้อยู่ในหน่วยความจำจะมีการนำเพจที่ถูกเรียกนั้นเข้ามาไว้ใน เพจเฟรม (page frames)



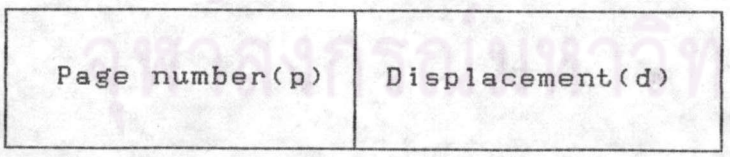
Page number	Page size	Range of real storage addresses
0	p	0 -> p-1
1	p	p -> 2p-1
2	p	2p -> 3p-1
.	.	.
.	.	.
.	.	.

รูปที่ 3.4 หลักการของการแบ่งเพจ

เมื่อมีการอ้างถึงตำแหน่งเสมือน กลไกการทำ Page Mapping จะมองหาตำแหน่ง ใน Page Map Table ซึ่งเป็นตัวกำหนดว่ามีเพจใดอยู่ในหน่วยความจำหลักและอยู่ที่ตำแหน่งใด ถ้าไม่อยู่สามารถค้นหาได้จากตำแหน่งใดในหน่วยความจำสำรอง

3.2.2.2 วิธีการของการแบ่งเพจ

การอ้างอิงในระบบแบ่งเพจ กระทำคล้ายคลึงกับการอ้างอิงในระบบแบ่งปล้อง คือใช้คู่ลำดับ [p,d] โดย



Virtual address
v = [p,d]

ตำแหน่งเสมือนในระบบแบ่งเพจ จะเป็นแบบคู่ลำดับ [p,d]

โดยที่ p คือ หมายเลขเพจ ในหน่วยความจำเสมือน

d คือ ระยะในเพจ p

และทำนองเดียวกัน จะมีตารางเพจประจำสำหรับแต่ละงาน โดยแต่ละรายการในตาราง เป็นสารสนเทศเกี่ยวกับเพจนั้นๆ สารสนเทศในแต่ละรายการมีดังต่อไปนี้

3.2.2.2.1 หมาย (flag) แสดงว่า เพจนั้นอยู่ในหน่วยความจำหลักหรือไม่

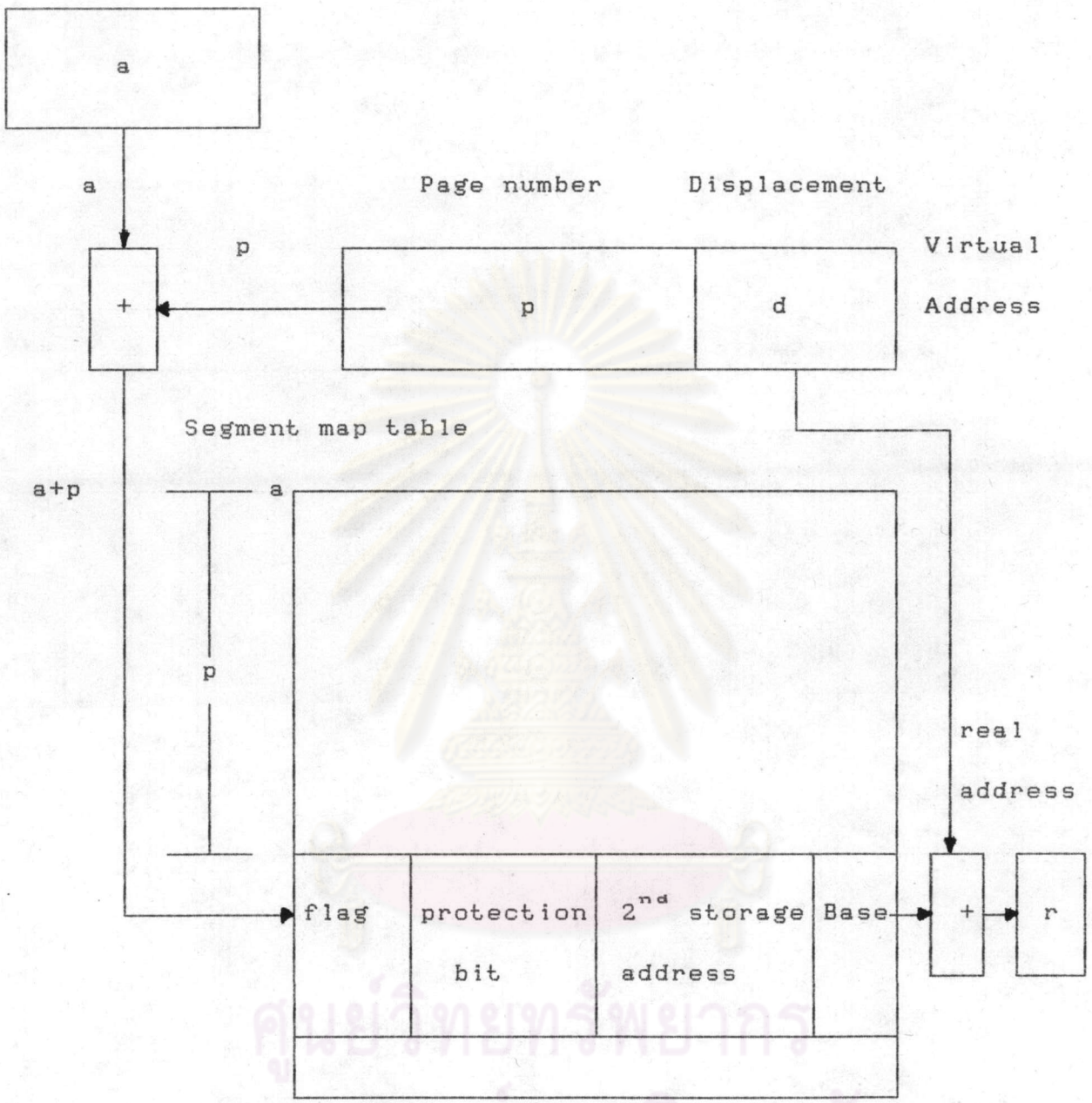
3.2.2.2.2 ตำแหน่งฐาน (base address) คือจุดเริ่มต้นของเพจในหน่วยความจำหลัก

3.2.2.2.4 บิตแสดงขอบเขตการใช้เพจ (protection bit)

3.2.2.2.5 ตำแหน่งในหน่วยความจำรอง (secondary storage address)

นอกจากนี้ยังมี รีจิสเตอร์พิเศษ เรียกว่า รีจิสเตอร์บ่งตารางเพจ (page table register) ซึ่งบรรจุตำแหน่งเริ่มต้นของตารางเพจสำหรับกระบวนการที่กำลังดำเนินอยู่ในขณะนั้น

การแปลงตำแหน่งในหน่วยความจำเสมือน ให้เป็นตำแหน่งในหน่วยความจำจริงในระบบแบ่งเพจ เป็นไปตามรูปที่ 3.5



รูปที่ 3.5 แสดงวิธีการของการแบ่งเพจ

โดยที่ aเป็นตำแหน่งเริ่มต้นของ Page map table ในหน่วยความจำหลัก เมื่อมีการอ้างถึงตำแหน่งใดๆ ตำแหน่งดังกล่าวจะเป็นตำแหน่งเสมือน ดังนั้นจึงจำเป็นต้องแปลงเป็นตำแหน่งที่ใช้งานจริงก่อน และถ้ามีการอ้างถึงหมายเลขเพจ p และระยะ d

ตำแหน่งใน page map table สำหรับหมายเลขปล้อง p' = a + p

โดยที่ p' = ตำแหน่งที่หน่วยความจำหลัก

ตำแหน่งที่ใช้งานจริง (r) = p' + d

3.2.2.3 ข้อดีและข้อเสียของระบบแบ่งเพจ

ข้อดี

ข้อดีของการแบ่งเพจคล้ายคลึงกับการแบ่งปล้อง เช่นการทับซ้อนโดยอัตโนมัติ รวมถึงการแปลงตำแหน่งเสมือนไปเป็นตำแหน่งจริงโดยระบบทำให้เอง และการมีหน่วยความจำเสมือนที่ใหญ่กว่าหน่วยความจำหลักได้

นอกจากนี้การที่แต่ละเพจเท่ากันหมดดังนั้นในทางกายภาพ เพจใดๆจะอยู่ในเพจเฟรมใดๆก็ได้ ทำให้ลดความซับซ้อนในการบรรจุเพจใดๆลงในหน่วย - ความจำหลัก และตัดปัญหาการขาดวิ่นของหน่วยความจำหลัก

ข้อเสีย

ข้อเสียของการแบ่งเพจคือ เป็นลักษณะการแบ่งด้วยระบบอัตโนมัติ ซึ่งไม่เป็นไปตามตรรกการแบ่งหน่วยของโปรแกรมและข้อมูล ทำให้การเชื่อมโยงส่วนต่างๆของงานเดียวกัน หรือระหว่างหลายงานซับซ้อนขึ้น

3.2.3 การแบ่งเพจร่วมกับการแบ่งปล้อง (Combined Paging Segmentation)

เป็นการผนวกวิธีการแบ่งปล้องและแบ่งเพจเข้าด้วยกัน ทำได้ 2 ลักษณะคือ

3.2.3.1 การผนวกแบบแบ่งปล้องเป็นแผ่น(page segmentation)

วิธีการนี้พยายามแก้ไขลักษณะ 2 ประการที่ไม่ดีของการแบ่งปล้องคือ

1 การที่ปล้องมีขนาดไม่เท่ากัน ทำให้ลำบากต่อการข้อมูลลงในหน่วยความจำหลัก

2 กรณีที่ข้อมูลส่วนมากในปล้องไม่ได้ใช้ จึงไม่จำเป็นต้องนำเข้ามาหมดทั้งปล้อง นำเฉพาะส่วนที่ต้องการใช้เข้ามาเท่านั้น

ด้วยเหตุนี้การแก้ปัญหาจะทำได้โดยแบ่งปล้องออกเป็นเพจที่มีขนาดเท่ากัน

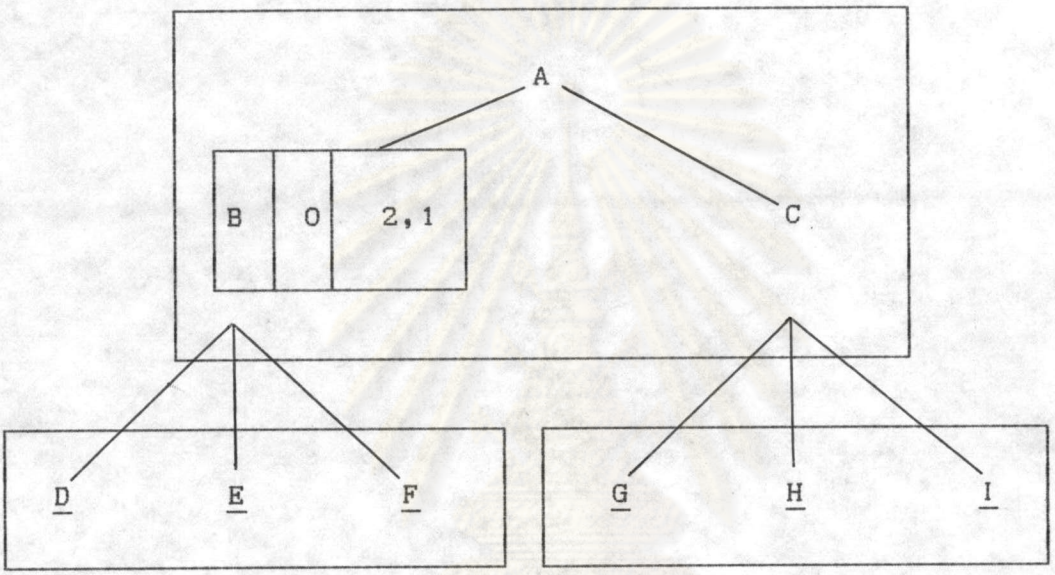
3.2.3.2 การผนวกแบบแบ่งปล้องตารางแผ่น(segmented paging)

ในระบบแบ่งเพจที่มีขนาดของหน่วยความจำเสมือนใหญ่มากๆ ขนาดของตารางเพจจะใหญ่ตามไปด้วย เนื่องจากหน่วยความจำเสมือนที่มีขนาดใหญ่มากๆ

โดยปกติใช้เพียงส่วนน้อยเท่านั้น เนื้อที่ส่วนใหญ่จะว่าง รายการสำหรับของบริเวณที่ว่างนั้นจะไม่มีความจำเป็น จึงสามารถตัดแบ่งตารางแผ่นออกเป็นส่วน หรือเป็นปล้อง โดยเก็บเฉพาะปล้องที่บรรจุรายการเพจที่ไม่ว่างเท่านั้น

ในวิทยานิพนธ์ฉบับนี้จะใช้วิธีการแบ่งเพจมาใช้ในการแบ่งหน่วยความจำหลักออกเป็นส่วนๆดังตัวอย่างต่อไปนี้

สมมุติว่าพจนานุกรมประกอบด้วยทรายดังรูป



จากรูปข้างต้นนี้สามารถอธิบายโครงสร้างของพจนานุกรมโดยพิจารณาจากโหนด B ดังนี้

ค่าที่เก็บ	ความหมาย
B	ตัวอักษรในพจนานุกรม
O	ตัวบ่งชี้ว่าไม่เป็นอักษรตัวสุดท้าย
2,1	ตัวอักษรถัดไปอยู่ในเพจที่ 2 โหนดที่ 1

3.3 วิธีการแทนที่

ในระบบการแบ่งปล้อง การบรรจุปล้องลงในหน่วยความจำหลักจำเป็นต้องหาบริเวณที่ว่างให้เพียงพอก่อน หากเมื่อกระชับหน่วยความจำหลักแล้วยังมีที่ไม่พอก็จะต้องดูว่าปล้องใดที่สามารถนำออกจากหน่วยความจำหลักได้ ในระบบการแบ่งเพจก็เช่นเดียวกัน หากเลือกเพจที่จะถูกแทนที่ไม่ดีเพจนั้นอาจถูกเรียกใช้ในอนาคตอันใกล้ ทำให้เกิดการขาดเพจขึ้นอีกโดยไม่จำเป็น

วิธีการในการเลือกเพจที่ไม่ใช้งานออกจากหน่วยความจำหลัก ซึ่งมีหลายวิธีดังนี้

1. การแทนที่ซึ่งได้ผลดีที่สุด(Optimal Page Replacement)
2. การแทนที่แบบไม่ได้ใช้นำออกก่อน(Not-Recently-Used Page Replacement)
3. การแทนที่แบบเข้าก่อนออกก่อน(First-In First-Out Page Replacement)
4. การแทนที่แบบไม่ได้ใช้นานที่สุด(Least Recently Used Page Replacement)
5. การจำลองวิธีการแทนที่แบบไม่ได้ใช้นานที่สุดโดยใช้ซอฟต์แวร์(Simulating LRU in Software)

1. การแทนที่ซึ่งได้ผลดีที่สุด(Optimal Page Replacement)

วิธีการนี้ใช้หลักการ ให้แทนที่เพจซึ่งจะไม่ได้ใช้นานที่สุด วิธีการนี้ให้ปริมาณการขาดเพจต่ำที่สุดเปรียบเทียบกับวิธีการอื่นๆ โดยการมองไปในอนาคตว่าเพจต่างๆ ในหน่วยความจำหลักขณะนั้นเพจใดจะถูกอ้างอิงช้าที่สุด ก็เลือกเอาเพจนั้นออก แล้วนำเพจใหม่เข้าไปแทนที่(เพจใหม่ที่เกิดจากการขากเพจเพราะถูกอ้างอิงแล้วไม่อยู่ในหน่วยความจำหลัก)

ข้อดีและข้อเสียของการแทนที่ซึ่งได้ผลดีที่สุด

ข้อดี

เป็นวิธีการที่ให้ผลดีที่สุดหากพิจารณาในหัวข้อถัดไปวิธีอื่นๆจะไม่สามารถทำได้ดีกว่านี้เลย

ข้อเสีย

ระบบไม่สามารถทราบได้ว่า ในอนาคตจะมีการอ้างอิงในลักษณะใดบ้าง ดังนั้นระบบการแทนที่แบบนี้จึงใช้เป็นบรรทัดฐานสำหรับการเปรียบเทียบระดับความสามารถของวิธีการแทนที่แบบต่างๆ

2. การแทนที่แบบไม่ได้ใช้นำออกก่อน(Not-Recently-Used Page Replacement)

เป็นวิธีที่ใช้ฮาร์ดแวร์เข้าช่วยในการเก็บสถิติการใช้งานของเพจใดๆโดยใช้

1. บิตอ้างอิง(Referenced bit) จะถูกกำหนดค่าโดยฮาร์ดแวร์เมื่อมีการอ่านหรือเขียนเพจ

2. บิตแก้ไข(Modified bit) จะถูกกำหนดค่าโดยฮาร์ดแวร์เมื่อมีการเขียนเพจ

วิธีนี้จะมีการแก้ไขบิตอ้างอิงและบิตแก้ไขทุกครั้งที่มีการอ้างอิงถึงเมื่อเกิดการขาดเพจขึ้นการเลือกเพจใดๆออกจะทำโดยแบ่งชนิดของเพจออกเป็น 4 กลุ่มโดยแบ่งจากค่าบิตอ้างอิงและบิตแก้ไขดังนี้

กลุ่มที่ 0 : ไม่ถูกอ้างอิง,ไม่ถูกแก้ไข

กลุ่มที่ 1 : ไม่ถูกอ้างอิง,ถูกแก้ไข

กลุ่มที่ 2 : ถูกอ้างอิง,ไม่ถูกแก้ไข

กลุ่มที่ 3 : ถูกอ้างอิง,ถูกแก้ไข

การดึงเพจใดๆออกจะดึงกลุ่มที่เลขน้อยที่สุดออกก่อน โดยในกลุ่มเดียวกันการดึงออกจะเป็นแบบสุ่ม

ข้อดีและข้อเสียของการแทนที่แบบไม่ได้ใช้นำออกก่อน

ข้อดี

1. ง่ายต่อการเข้าใจ

2. สามารถนำมาใช้ได้จริงอย่างมีประสิทธิภาพ

ข้อเสีย

ค่าใช้จ่ายในการนำมาใช้สูงเพราะต้องใช้ฮาร์ดแวร์เข้ามาช่วย

3. การแทนที่แบบเข้าก่อนออกก่อน(First-In First-Out Page Replacement)

การแทนที่แบบเข้าก่อนออกก่อนใช้หลักการให้แทนที่แผ่นซึ่งเก่าแก่ที่สุด นั่นคือให้แทนที่ แผ่นที่เข้ามาอยู่ในหน่วยความจำหลักนานที่สุด

ข้อดีและข้อเสียของการแทนที่แบบเข้าก่อนออกก่อน

ข้อดี

วิธีการนี้สามารถที่จะเข้าใจได้ง่ายและปฏิบัติได้ง่าย

ข้อเสีย

แม้ว่าวิธีการแทนที่แบบเข้าก่อนออกก่อนนี้จะเข้าใจง่ายและปฏิบัติได้ง่าย แต่ผลกระทบต่อระบบจะขึ้นกับประเภทของแผ่น เช่นหากเป็นแผ่นซึ่งเกี่ยวข้องกับการกำหนดสภาวะเริ่มต้นของงานวิธีการนี้ก็เหมาะที่จะใช้ แต่ถ้าเป็นข้อมูลซึ่งใช้อยู่ตลอดช่วงอายุของงาน จะถูกนำออกจากหน่วยความจำหลักทั้งๆที่ยังเป็นที่ต้องการอยู่

4. การแทนที่แบบไม่ได้ใช้นานที่สุด(Least Recently Used Page Replacement)

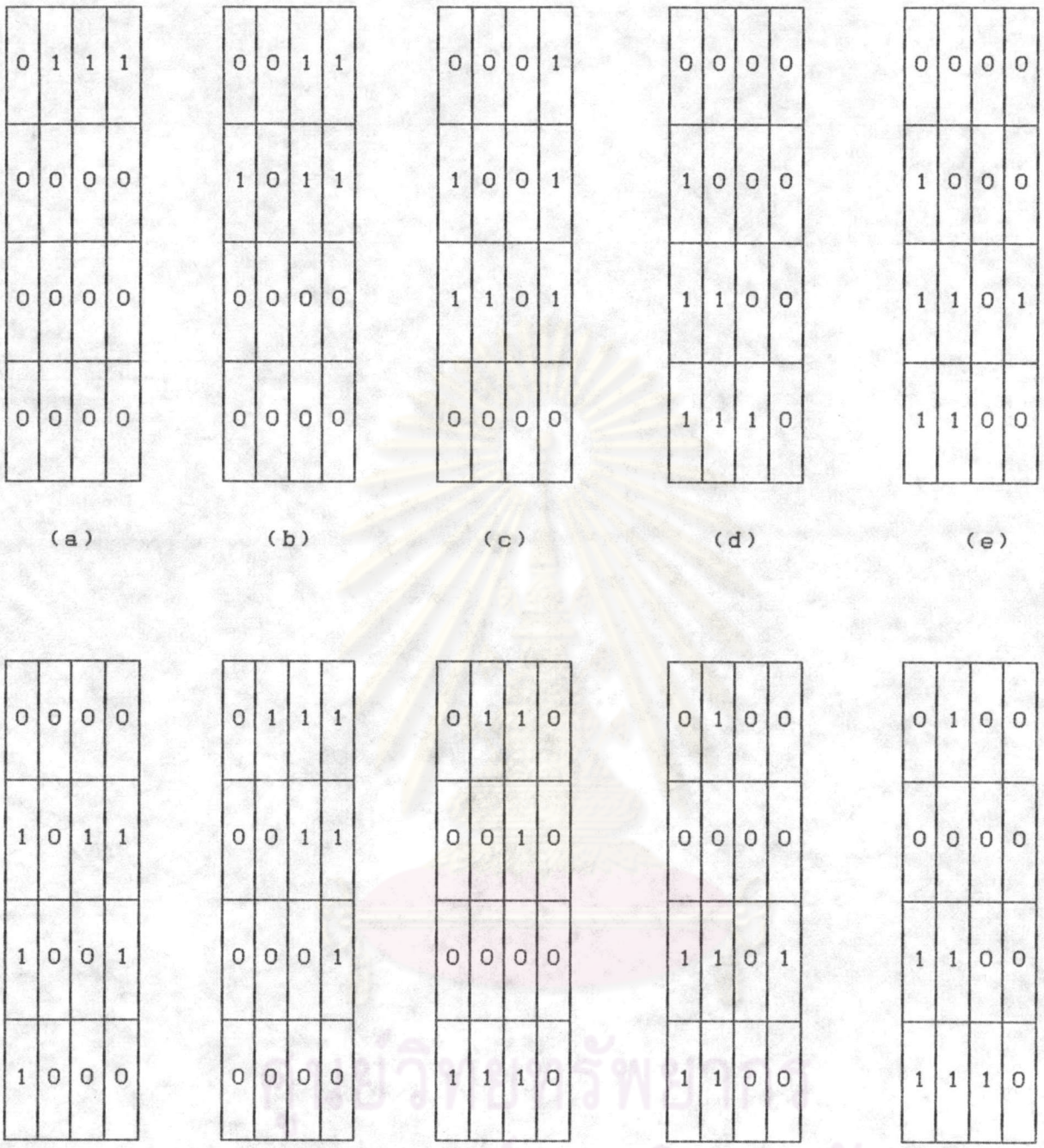
วิธีการเลือกเพจแบบ LRU (Tanenbaum and Andrew S. , 1987) พิจารณาจากการใช้งานว่าเพจใดมีการใช้งานมากในช่วงเวลาหนึ่ง คาดว่าเพจนั้นก็จะยังคงถูกใช้งานต่อไป และเพจที่ไม่ถูกเรียกใช้ ยังคงไม่ถูกเรียกใช้ต่อไป แม้ว่าหลักการ LRU เป็นทฤษฎีที่สามารถนำมาใช้จริงได้ แต่ในทางปฏิบัติราคาค่อนข้างแพง การใช้งานต้องมีตัวเชื่อมโยงระหว่างทุกเพจในหน่วยความจำหลักโดยเพจที่ถูกเรียกใช้บ่อยอยู่ที่ front ส่วนเพจที่ใช้น้อยกว่าอยู่ที่ rear การเลือกเพจออกจากหน่วย - ความจำหลักจะเลือกเพจที่ปรากฏที่ส่วน rear ซึ่งวิธีการดังกล่าวนี้ มีความยุ่งยากอยู่ที่จะมีการแก้ไขข้อมูลตัวเชื่อมโยงทุกครั้งที่มีการอ้างถึงเพจในหน่วยความจำหลัก ทำให้เสียเวลาในการประมวลผล นอกจากนี้ยังต้องใช้ฮาร์ดแวร์เพื่อช่วยในการดำเนินงานด้วย

วิธีการ (Algorithm) ที่จะกล่าวถึงต่อไปนี้เป็นการนำฮาร์ดแวร์มาช่วย โดยใช้ฮาร์ดแวร์ที่มีตัวนับขนาด 64 บิท (64 bit counter : C) เมื่อมีการอ้างถึงเพจแต่ละครั้งตัวนับจะเพิ่มค่าขึ้นอัตโนมัติ นอกจากนี้ในตารางเพจ (page table) จะประกอบด้วยเขตข้อมูลซึ่งมีขนาดใหญ่พอที่จะบรรจุค่าตัวนับ (C)

หลังจากที่มีการอ้างถึงเพจใดๆในแต่ละครั้ง ค่าของตัวนับจะถูกบันทึกลง ในตารางเพจสำหรับเพจนั้นๆ และเมื่อมีการอ้างถึงเพจที่ไม่ได้อยู่ในหน่วยความจำหลัก จะมีการตรวจสอบในตารางเพจว่า เพจใดมีค่าตัวนับต่ำสุดแสดงว่าเพจดังกล่าวไม่ได้ถูกเรียกใช้เป็นเวลานานที่สุด จะดึงเอาเพจนี้ออกจากหน่วยความจำหลัก แล้วนำเพจที่ถูกอ้างถึงที่อยู่ใน หน่วยความจำสำรองเข้ามาแทนที่

วิธีการที่ 2 ที่ใช้ฮาร์ดแวร์เข้ามาช่วยใช้กับคอมพิวเตอร์ที่มี n เพจเฟรม โดย LRU ฮาร์ดแวร์ จะประกอบด้วยเมตริกซ์ขนาด n*n บิท ซึ่งมีการกำหนดค่าเริ่มต้นในเมตริกซ์เป็น 0 เมื่อมีการอ้างถึงเพจเฟรมที่ k ขั้นตอนแรก ฮาร์ดแวร์ จะกำหนดค่าสำหรับทุกบิทในแถวที่ k ให้ มีค่าเป็น 1 และกำหนดค่าสำหรับทุกบิทในคอลัมน์ที่ k ให้มีค่าเป็น 0 เมื่อมีการอ้างถึงเพจที่ไม่อยู่ในหน่วยความจำหลัก ก็จะไปดูว่าค่าในแถวใดมีค่าน้อยที่สุด เพจนั้นจะเป็นเพจที่ใช้น้อยที่สุดและจะถูกดึงออก ซึ่งจะแสดงให้เห็นดังตัวอย่างต่อไปนี้

สมมุติว่ามีเพจเฟรม 4 เฟรม และมีการอ้างถึงเพจต่างๆ ในหน่วยความจำหลักตามลำดับดังนี้เพจ 0 1 2 3 2 1 0 3 2 3 ในการอ้างถึงเพจ 0 ค่าในตารางเพจจะมีค่าดังรูปที่ 3.6 (a) ในการอ้างครั้งต่อไปก็จะเป็นรูปที่ 3.6 (b) และต่อไปเรื่อยๆ จนครบ 10 ครั้ง



รูปที่ 3.6 แสดงการแทนที่โดยใช้หลักการของ LRU

5. การจำลองวิธีการแทนที่แบบไม่ได้ใช้นานที่สุดโดยใช้ซอฟต์แวร์ (Simulation LRU in Software)

เนื่องจากวิธีการข้างต้นที่กล่าวมาแล้วจำเป็นจะต้องใช้ฮาร์ดแวร์ช่วย ถ้าไม่มีฮาร์ดแวร์ สามารถใช้ซอฟต์แวร์เข้าช่วยโดยวิธี Not Frequently Used Algorithm (NFU) วิธีการนี้ต้องการตัวนับซอฟต์แวร์(software counter) สำหรับแต่ละเพจ โดยกำหนดค่าเริ่มต้นของตัวนับให้เป็น 0 ในทุกครั้งที่เกิดสัญญาณขัดจังหวะ (clock interrupt) จะมีการกำหนดค่าบิทอ้างอิง(Reference bit : R bit) และทำการเพิ่มค่าค่าตัวนับ โดยนำค่า R bit มาบวก ซึ่ง R bitมี ค่าได้ 2 ค่า คือ 0 และ 1 โดยที่

0 คือ เพจ นั้นไม่ได้ถูกเรียกใช้

1 คือ เพจ นั้นกำลังถูกเรียกใช้

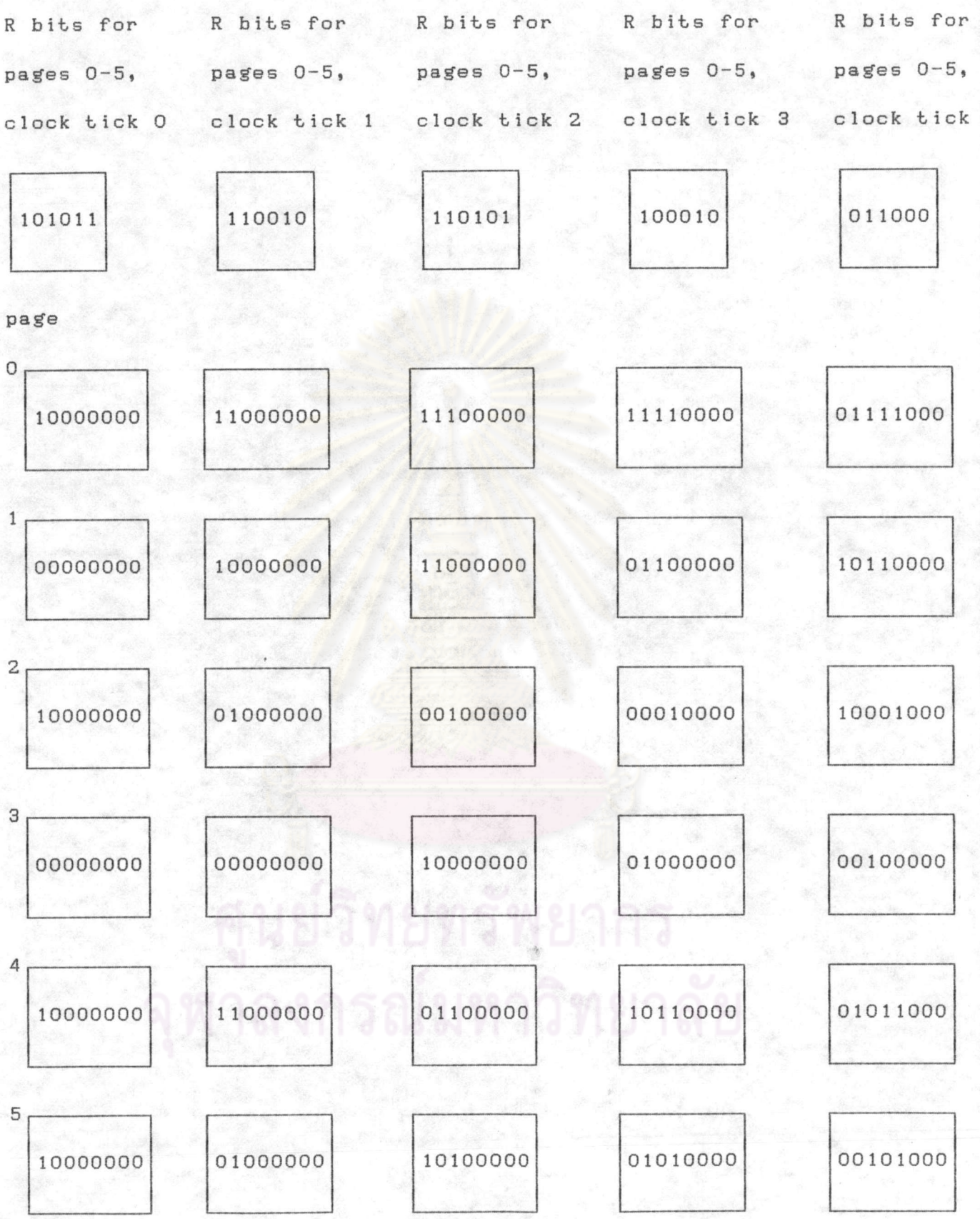
ตัวนับจะเป็นตัวชี้ว่าเพจใด ถูกเรียกใช้บ่อยแค่ไหน เมื่อมีการเรียกใช้เพจ ที่ไม่ได้อยู่ในหน่วยความจำหลัก ก็จะเลือกเพจ ที่มีค่าตัวนับ น้อยที่สุดออกไป แต่อย่างไรก็ตามวิธีการ NFU มีข้อเสียคือ ไม่มีการลบค่าตัวนับออกจนกว่าจะจบการใช้งาน ปัญหานี้สามารถแก้ไขโดย

1. ก่อนที่จะบวกค่า R bit กับ ตัวนับ จะเลื่อนบิทของตัวนับไปทางขวา 1 บิท

2. นำ R bit บวกเข้าทางซ้ายสุดของ ตัวนับ

วิธีการข้างต้นมีชื่อว่า aging จะแสดงตัวอย่างการใช้งานดังต่อไปนี้

สมมุติว่ามีเพจเฟรม 5 เฟรม หลังจากมีสัญญาณขัดจังหวะครั้งแรก ค่า R บิทของเพจต่างๆมีค่าดังนี้ 1,0,1,0,1, และ 1 (เพจ 0,2,4,5 ถูกเรียกใช้งาน)แสดงได้ดังรูปที่ 3.7 (a) รูปที่ 3.7 (b),(c), ... แสดงถึงสัญญานขัดจังหวะครั้งต่อไป



รูปที่ 3.7 แสดงการแทนที่โดยใช้หลักการของ aging

ในกรณีนี้การเลือกเพจออกจากหน่วยความจำหลัก จะเลือกเพจที่มีค่าตัวนับน้อยที่สุด