บทที่ 7

การศึกษาและวิเคราะห์สมรรถนะเมื่อเพิ่มหน่วยความจำแคช

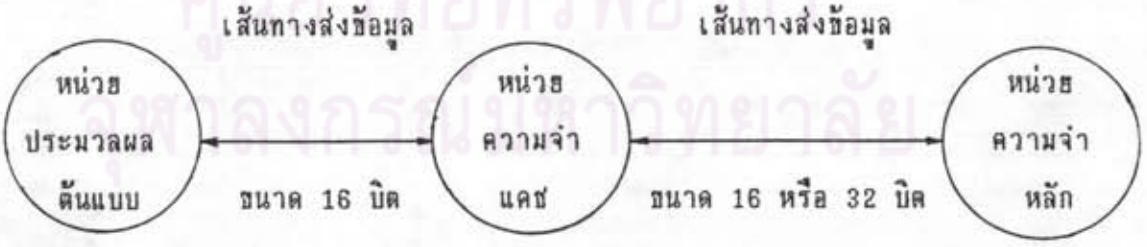
วิธีหนึ่งในการเพิ่มประสิทธิภาพของระบบคอมพิวเตอร์คือการเพิ่มความเร็วในการปฏิบัติการคำสั่งหรือมีการปรับปรุงกลไกการทำงานของหน่วยประมวลผล อย่างไรก็ตามการปรับปรุงดังกล่าวอาจไม่ทำให้สมรรถนะของระบบดีขึ้นจริง เมื่อหน่วยประมวลผลยังต้องรอการทำงานของหน่วยความจำ เพื่ออ่านหรือบันทึกข้อมูล ดังนั้นเวลาในการปฏิบัติการคำสั่งของหน่วยประมวลผลจึงถูกจำกัดด้วยเวลาที่ใช้ในการเข้าถึงข้อมูลในหน่วยความจำ ข้อจำกัดนี้เป็นปัญหาสำคัญ ซึ่งถ้าความเร็วในการทำงานของหน่วยความจำหลักไม่ได้รับการปรับปรุงให้ดีขึ้นแล้ว ประสิทธิภาพของระบบก็ไม่สามารถเพิ่มขึ้นได้อย่างเต็มที่ ดังนั้นถ้าสามารถปรับปรุงการทำงานของหน่วยความจำหลักให้รวดเร็วทันความต้องการของหน่วยประมวลผล จะส่งผลให้ประสิทธิภาพของระบบดีขึ้น

ในปัจจุบันผู้ออกแบบได้ให้ความสำคัญกับการเพิ่มประสิทธิภาพการทำงานของระบบ โดยการเพิ่มหน่วยความจำแคชภายในระบบ เพื่อเก็บสำเนาข้อมูลบางส่วนในหน่วยความจำหลัก ซึ่งเวลาที่ใช้ในการเข้าถึงข้อมูลที่อยู่ในหน่วยความจำแคชน้อยกว่าเมื่ออยู่ในหน่วยความจำหลัก ดังนั้นเมื่อหน่วยประมวลผลต้องการใช้ข้อมูลในหน่วยความจำหลัก ก็สามารถเรียกใช้ได้จากหน่วยความจำแคชโดยไม่ต้องรอข้อมูลจากหน่วยความจำหลัก สิ่งหนึ่งที่ควรพิจารณาในการออกแบบหน่วยความจำแคชให้เหมาะสมกับการทำงานของระบบได้แก่ อัตราส่วนในการพบข้อมูล (hit ratio) ซึ่งค่าอัตราส่วนนี้จะขึ้นกับขนาดหน่วยความจำแคช จำนวนไบต์ที่ถ่ายเทระหว่างหน่วยความจำหลักกับหน่วยความจำแคชในแต่ละรอบการทำงานของหน่วยความจำ โครงสร้างของหน่วยความจำแคช และพฤติกรรมของโปรแกรม (Williams,1985 ; Smith,1982)

ในที่นี้เราจะศึกษาและวิเคราะห์สมรรถนะของหน่วยประมวลผลต้นแบบเมื่อกำหนดให้มีหน่วยความจำแคชโดยพิจารณาเป็น 2 แนวทางคือ แปรขนาดหน่วยความจำแคช โดยศึกษาข้อมูลจากการวัดเปรียบเทียบสมรรถนะของหน่วยประมวลผลต้นแบบด้วยค่าสมรรถนะสัมพัทธ์ ซึ่งได้จากเวลาที่ใช้ในการประมวลผลคำสั่งในโปรแกรมโดยตัวแบบจำลองกรณีไม่มีหน่วยความจำแคชเปรียบเทียบกับเวลาที่ใช้ในการประมวลผลคำสั่งในโปรแกรมกรณีมีหน่วยความจำแคช และอีก

แนวทางหนึ่งคือ แปรขนาดเส้นทางส่งผ่านข้อมูล โดยเปรียบเทียบเวลาที่ตัวแบบจำลองใช้ในการปฏิบัติการกรณีไม่มีหน่วยความจำแคช กับเวลาที่ตัวแบบจำลองใช้ในการปฏิบัติการกรณีมีหน่วยความจำแคช โดยให้มีการแปรขนาดเส้นทางส่งข้อมูลคือขนาด 16 และ 32 บิต รวมทั้งได้ทำการศึกษาอัตราส่วนการพลาด (miss ratio) เพื่อศึกษาประสิทธิภาพการใช้งานหน่วยความจำแคช

ตัวแบบจำลองการทำงานของหน่วยประมวลผลต้นแบบ เมื่อกำหนดมีหน่วยความจำแคช ได้แสดงดังรูปที่ 7.1 โดยมีการพัฒนากรรมวิธีหน่วยควบคุมความจำแคช ซึ่งดูแลการทำงานเมื่อระบบต้องการข้อมูลในหน่วยความจำหลัก โดยทั่วไปโครงสร้างหรือการจัดองค์กรภายในหน่วยความจำแคชมี 3 รูปแบบคือ แบบ direct mapping แบบ fully associative และแบบ set associative สำหรับงานวิจัยนี้ได้เลือกการจัดองค์กรภายในหน่วยความจำแคชเป็นแบบ direct mapping ซึ่งเป็นวิธีที่หน่วยประมวลผลขนาดเล็กเลือกใช้เพราะหน่วยความจำแคชแบบนี้มีราคาถูกและการทำงานไม่ซับซ้อน และวิธีการปรับปรุงข้อมูลในหน่วยความจำหลักก็มีหลายวิธี เช่นวิธีการ write back ซึ่งเป็นวิธีการปรับปรุงข้อมูลก็ต่อเมื่อต้องมีการเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำแคชกับหน่วยความจำหลัก โดยทำการบันทึกข้อมูลจากหน่วยความจำแคชลงในตำแหน่งหน่วยความจำหลักที่อ้างอิงตรงกับตำแหน่งหน่วยความจำแคชนั้นก่อนการรับข้อมูลใหม่เข้ามาในหน่วยความจำแคช แต่ในงานวิจัยนี้กำหนดใช้วิธีการปรับปรุงข้อมูลโดยเมื่อมีการบันทึกข้อมูลใดๆลงในหน่วยความจำแคชก็จะมีการบันทึกในหน่วยความจำหลักด้วย วิธีการนี้เรียกว่า write through สำหรับรายละเอียดต่างๆที่เกี่ยวกับหน่วยความจำแคชสามารถศึกษาได้จาก (Hwang and Briggs,1988) (Smith,1982) และ (Stallings,1987)

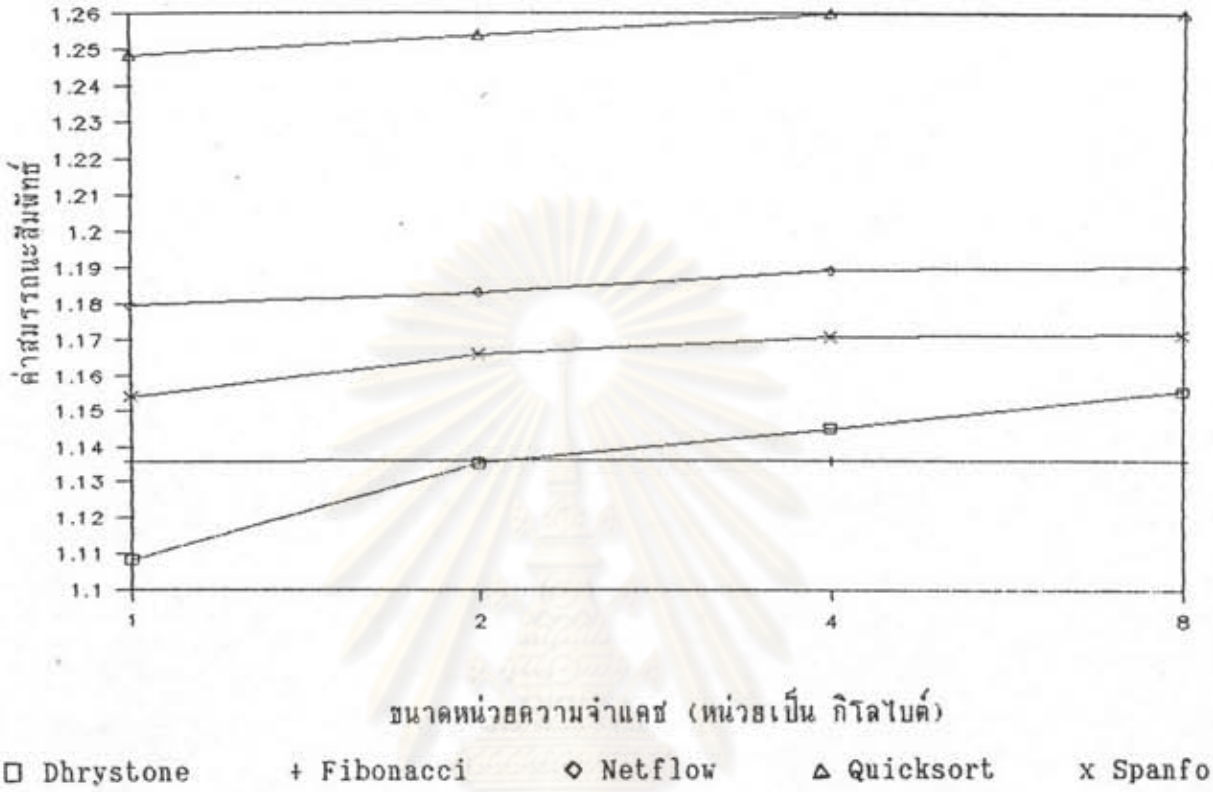


รูปที่ 7.1 แบบจำลองหน่วยประมวลผลต้นแบบเมื่อกำหนดมีหน่วยความจำแคช

7.1 กรณีศึกษาเมื่อเพิ่มหน่วยความจำแคชขนาดต่างกัน

ในการเข้าถึงข้อมูลในหน่วยความจำหลักจะกระทำโดยหน่วยเชื่อมต่อบัสซึ่งต้องใช้เวลา 4 รอบสัญญาณนาฬิกา ดังนั้นเมื่อมีการอ่านหรือบันทึกข้อมูลแต่ละครั้ง หน่วยปฏิบัติการก็ต้องรอให้หน่วยเชื่อมต่อบัสส่งข้อมูลมาให้เป็นเวลาอย่างน้อย 4 รอบสัญญาณนาฬิกาด้วย และถ้าการอ่านแล้วบันทึกข้อมูลในตำแหน่งเดียวกัน หน่วยเชื่อมต่อบัสก็ต้องใช้เวลาในการเข้าถึงข้อมูลทั้งหมด 8 รอบสัญญาณนาฬิกา แต่เมื่อมีหน่วยความจำแคชในระบบ จะทำให้ประสิทธิภาพการทำงานของระบบดีขึ้น เพราะเวลาที่ใช้ในการเข้าถึงข้อมูลในหน่วยความจำแคชใช้เพียง 1 รอบสัญญาณนาฬิกา ดังนั้นเมื่อหน่วยเชื่อมต่อบัสต้องการข้อมูลสำหรับการดำเนินงาน จะดึงข้อมูลได้จากหน่วยความจำแคชนั่นคือหน่วยเชื่อมต่อบัสใช้เวลาในการอ่านแล้วบันทึกข้อมูลในตำแหน่งเดิมทั้งหมด 6 รอบสัญญาณนาฬิกา โดยที่หน่วยเชื่อมต่อบัสใช้เวลาในการอ่านข้อมูลในหน่วยความจำแคช 1 รอบสัญญาณนาฬิกา ใช้เวลาในการบันทึกข้อมูลในหน่วยความจำแคช 1 รอบสัญญาณนาฬิกา และเนื่องจากกำหนดใช้วิธีการปรับปรุงข้อมูลแบบ write through ซึ่งต้องทำการบันทึกข้อมูลที่แก้ไขลงในหน่วยความจำหลักด้วย จึงต้องใช้เวลาในการเข้าถึงข้อมูลในหน่วยความจำหลักเพิ่มอีก 4 รอบสัญญาณนาฬิกา เห็นได้ว่าหน่วยปฏิบัติการเสียเวลารอน้อยลง ทำให้ปฏิบัติการคำสั่งได้เร็วขึ้น

การกำหนดให้มีหน่วยความจำแคชในระบบเป็นวิธีการหนึ่งที่ช่วยเพิ่มประสิทธิภาพการทำงานให้แก่หน่วยประมวลผล และถ้าขนาดหน่วยความจำแคชยิ่งเพิ่มขึ้นซึ่งทำให้มีข้อมูลเก็บในหน่วยความจำแคชได้มากขึ้น น่าจะเป็นผลให้อัตราส่วนการพบข้อมูลที่ต้องการสูงขึ้น ซึ่งทำให้สมรรถนะของระบบดีขึ้น สำหรับงานวิจัยนี้ได้ทำการวัดเปรียบเทียบสมรรถนะของหน่วยประมวลผลต้นแบบเมื่อแปรขนาดของหน่วยความจำแคช โดยศึกษาจากเวลาที่ใช้ในการปฏิบัติการโปรแกรมกรณีไม่มีหน่วยความจำแคช เปรียบเทียบกับเวลาที่ใช้ในการปฏิบัติการโปรแกรมกรณีหน่วยความจำแคชมีขนาด 1, 2 , 4 และ 8 กิโลไบต์ โดยกำหนดขนาดเส้นทางส่งข้อมูลระหว่างหน่วยความจำแคชและหน่วยความจำหลัก มีขนาด 16 บิต ซึ่งได้แสดงผลการเปรียบเทียบเป็นรูปกราฟ ดังรูปที่ 7.2



รูปที่ 7.2 แสดงค่าสมรรถนะสัมพัทธ์ในแต่ละโปรแกรมเมื่อแปรขนาดหน่วยความจำแคช

จากรูปที่ 7.2 สังเกตได้ว่าการกำหนดให้มีหน่วยความจำแคชในระบบให้ผลดีต่อการทำงานของระบบในทุกกรณี ทั้งนี้เพราะหน่วยเชื่อมต่อบัสเข้าถึงข้อมูลในหน่วยความจำแคชได้เร็วกว่าในหน่วยความจำหลัก และเมื่อพิจารณาแต่ละโปรแกรม พบว่ามีลักษณะที่ช่วยส่งเสริมให้มีการใช้งานหน่วยความจำแคชได้อย่างดีคือคำสั่งที่เข้าปฏิบัติการในลำดับต่อไป เป็นคำสั่งที่ดึงเข้ามาในหน่วยความจำแคชก่อนหน้าแล้ว เช่นโปรแกรม Fibonacci และ Quicksort ที่มีการทำงานภายในโปรแกรมแบบเวียนเกิดซึ่งเป็นการทำงานซ้ำๆโดยเรียกตัวเอง นั่นคือใช้กลุ่มคำสั่งเดิมในการทำงานเพื่อหาผลลัพธ์ถัดไป และอีกลักษณะหนึ่งคือตำแหน่งการอ้างอิงข้อมูลในครั้งต่อไปใกล้เคียงกับตำแหน่งที่อ้างอิงในขณะนี้ เช่นข้อมูลในโปรแกรม Dhrystone เป็นกลุ่มตัวอักษรที่เรียงติดต่อกัน การใช้โครงสร้างข้อมูลแบบแถวลำดับในโปรแกรม Netflow Quicksort และ Spanfo ซึ่งทำให้หน่วยความจำแคชมีข้อมูลที่จะใช้ต่อไปโดยไม่ต้องเสียเวลาไปติดต่อกับหน่วยความจำหลักทุกครั้ง และเมื่อมีการเพิ่มขนาดหน่วยความจำแคชให้มากขึ้นในทุกโปรแกรม พบว่าค่า

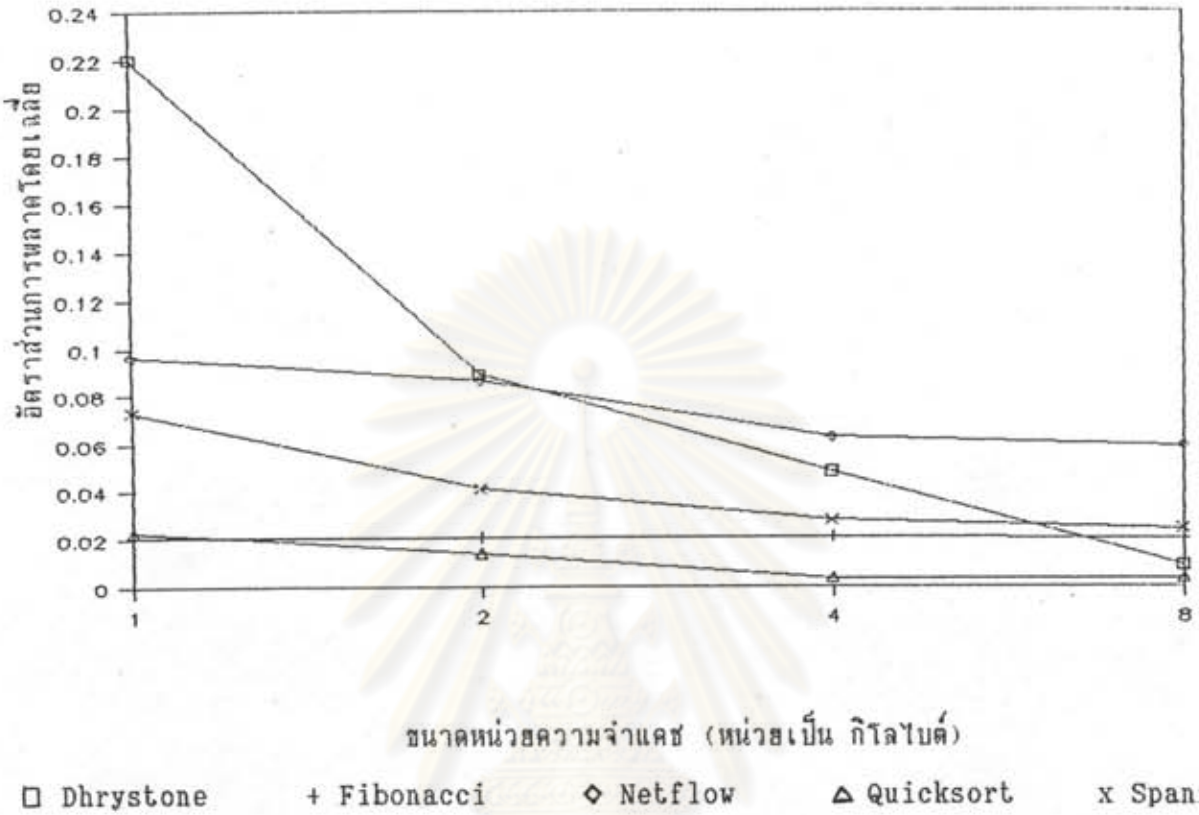
สมรรถนะสัมพัทธ์มีค่าสูงขึ้นเช่นกัน แต่อัตราการเปลี่ยนแปลงนั้นค่อนข้างน้อยโดยเฉพาะเมื่อเพิ่มขนาดหน่วยความจำแคชจาก 4 กิโลไบต์เป็น 8 กิโลไบต์ ทั้งนี้อาจเป็นเพราะขนาดโปรแกรมที่ใช้ในการทดสอบอยู่ในช่วง 1.7 - 5.5 กิโลไบต์ ซึ่งเมื่อกำหนดขนาดหน่วยความจำแคชให้เพียงพอต่อการเก็บข้อมูลสำหรับใช้งานโดยไม่ต้องยุ่งเกี่ยวกับหน่วยความจำหลักแล้ว การเพิ่มขนาดหน่วยความจำแคชมากขึ้นก็ไม่ส่งผลให้ค่าสมรรถนะสัมพัทธ์ดีขึ้นตามด้วย

เมื่อพิจารณาแต่ละโปรแกรม จะเห็นว่า Dhrystone ให้การเปลี่ยนแปลงที่ดีขึ้นอย่างชัดเจน น่าจะเป็นเพราะลักษณะการทำงานภายในโปรแกรมที่สังเคราะห์ขึ้นนี้ไม่เฉพาะเจาะจงที่ช่วงคำสั่งใดเป็นพิเศษ และมีข้อมูลแบบแถวลำดับจำนวนหนึ่ง จึงทำให้การเพิ่มหน่วยความจำแคชมีผลต่อการทำงานของโปรแกรม Dhrystone ได้เป็นอย่างดี สำหรับ Fibonacci เราพบว่าการกำหนดให้มีหน่วยความจำแคชเพียง 1 กิโลไบต์ ก็เพียงพอต่อการทำงานแล้ว เพราะเมื่อเพิ่มขนาดหน่วยความจำแคชมากขึ้น ก็ไม่ทำให้ค่าสมรรถนะสัมพัทธ์สูงขึ้นกว่าเดิมมากนัก อาจวิเคราะห์ได้ว่าโปรแกรมมีขนาดเพียง 1.7 กิโลไบต์ และมีการทำงานแบบเวียนเกิดซึ่งจำนวนไบต์คำสั่งในช่วงการทำงานนี้มีประมาณ 60 ไบต์ ดังนั้นคำสั่งที่จะถูกกระทำการน่าจะมีอยู่ในหน่วยความจำแคชแล้ว และสำหรับ Netflow Quicksort และ Spanfo ให้การเปลี่ยนแปลงที่ดีขึ้น เมื่อเพิ่มหน่วยความจำแคชเพราะโปรแกรมมีขนาดใหญ่ และภายในโปรแกรมมีข้อมูลแบบแถวลำดับอยู่เป็นจำนวนมาก

อีกแง่มุมหนึ่งที่ใช้ในการพิจารณาถึงการเพิ่มหน่วยความจำแคชแล้วทำให้ค่าสมรรถนะสัมพัทธ์ดีขึ้นคือ ค่าอัตราส่วนการพลาดเมื่อกำหนดให้มีหน่วยความจำแคชขนาดต่างๆกัน จากการทำงานของหน่วยเชื่อมต่อบัส เมื่อหน่วยเชื่อมต่อบัสต้องการข้อมูลสำหรับการดำเนินงาน ก็จะมีการค้นหาข้อมูลที่ต้องการนั้นในหน่วยความจำแคชก่อน ซึ่งผลที่ได้คือพบหรือไม่พบอย่างใดอย่างหนึ่ง ถ้าหน่วยเชื่อมต่อบัสค้นหาข้อมูลในหน่วยความจำแคชแล้วพบข้อมูลที่ต้องการทุกครั้ง แสดงว่าได้ใช้งานหน่วยความจำแคชอย่างมีประสิทธิภาพ และทำให้สมรรถนะการทำงานของหน่วยประมวลผลดีขึ้นด้วย ในการแสดงถึงการใช้งานหน่วยความจำแคชได้อย่างมีประสิทธิภาพ เราสามารถพิจารณาด้วยอัตราส่วนการพบข้อมูลในหน่วยความจำแคช ซึ่งได้จากจำนวนครั้งที่พบข้อมูลในหน่วยความจำแคชทันที หารด้วยจำนวนครั้งในการเข้าถึงหน่วยความจำแคช สมมติให้แทนอัตราส่วนการพบข้อมูลโดยทันทีนี้ด้วย h ดังนั้นถ้าค่า h มีค่ามากแสดงว่ามีการใช้งานหน่วยความจำแคชอย่างมีประสิทธิภาพมาก เพราะนั่นหมายถึงข้อมูลที่เก็บอยู่ในหน่วยความจำแคชเป็นข้อมูลที่ตรงความต้องการของหน่วยเชื่อมต่อบัส ดังนั้นเมื่อหน่วยเชื่อมต่อบัสต้องการข้อมูลก็สามารถพบได้ทันทีในหน่วยความจำแคช ทำให้หน่วยเชื่อมต่อบัสไม่ต้องรอข้อมูลมาจากหน่วยความจำหลักจึงเป็นผลให้หน่วยประมวลผลทำงานได้เร็วขึ้นหรือเราอาจพิจารณาการใช้งานหน่วยความจำแคชได้อย่าง

มีประสิทธิภาพด้วยอัตราส่วนการพลาด ซึ่งได้จากจำนวนครั้งที่ไม่พบข้อมูลที่ต้องการในหน่วยความจำแคชได้โดยทันที หารด้วยจำนวนครั้งในการเข้าถึงหน่วยความจำแคชหรือเท่ากับ 1 - h นั่นเอง ดังนั้นอัตราส่วนการพลาดนี้จึงควรมีค่าต่ำสุดในการแสดงประสิทธิผลเมื่อมีการใช้งานหน่วยความจำแคช

รูปกราฟต่อไปนี้ แสดงถึงอัตราส่วนการพลาดโดยเฉลี่ยที่เกิดขึ้นจากแต่ละโปรแกรมโดยกำหนดให้มีหน่วยความจำแคชขนาดต่างๆกัน ซึ่งผลที่ได้พบว่าเมื่อหน่วยความจำแคชเพิ่มขึ้น อัตราส่วนการพลาดในทุกโปรแกรมมีค่าลดลงอย่างเห็นได้ชัด ยกเว้น Fibonacci และเมื่อพิจารณาผลการทดสอบโปรแกรม Dhrystone พบว่าการเพิ่มขนาดหน่วยความจำแคชจาก 1 กิโลไบต์เป็น 2 4 และ 8 กิโลไบต์ ทำให้อัตราส่วนการพลาดโดยเฉลี่ยมีค่าลดลงร้อยละ 13 ร้อยละ 17 และร้อยละ 21 ตามลำดับ แต่ในการทดสอบโปรแกรม Fibonacci พบว่าแม้จะเพิ่มขนาดหน่วยความจำแคชมากขึ้น แต่การเปลี่ยนแปลงของอัตราส่วนการพลาดโดยเฉลี่ยมีเพียงร้อยละ 0.02 หรือกล่าวได้ว่าการเพิ่มขนาดหน่วยความจำแคชยังคงให้ค่าอัตราส่วนนี้เท่ากันกับเมื่อหน่วยความจำแคชมีขนาด 1 กิโลไบต์ จากการทดสอบโปรแกรม Netflow พบว่าเมื่อเพิ่มขนาดหน่วยความจำแคชเป็น 2 4 และ 8 กิโลไบต์ ทำให้ค่าอัตราส่วนการพลาดโดยเฉลี่ยมีการเปลี่ยนแปลงไปจากเมื่อกำหนดขนาดหน่วยความจำแคชเป็น 1 กิโลไบต์ คือลดลงร้อยละ 1 ร้อยละ 3 และร้อยละ 4 ตามลำดับ สำหรับผลการทดสอบโปรแกรม Quicksort พบว่าการเพิ่มขนาดหน่วยความจำแคชเป็น 2 และ 4 กิโลไบต์ ทำให้ค่าอัตราส่วนการพลาดโดยเฉลี่ยลดลงร้อยละ 1 ร้อยละ 2 ตามลำดับ และเมื่อเพิ่มขนาดหน่วยความจำแคชเป็น 8 กิโลไบต์ มีผลให้อัตราส่วนการพลาดนี้มีการเปลี่ยนแปลงน้อยมาก โดยให้ค่าอัตราส่วนการพลาดโดยเฉลี่ยใกล้เคียงกับกรณีขนาดหน่วยความจำแคชเท่ากับ 4 กิโลไบต์ และสุดท้ายเมื่อทดสอบโปรแกรม Spanfo พบว่าเมื่อเพิ่มขนาดหน่วยความจำแคชเป็น 2 และ 4 กิโลไบต์ ทำให้ค่าอัตราส่วนการพลาดโดยเฉลี่ยลดลงร้อยละ 3 และร้อยละ 5 ตามลำดับ และเมื่อเพิ่มขนาดหน่วยความจำแคชขึ้นอีก ก็ทำให้อัตราส่วนการพลาดมีการเปลี่ยนแปลงไม่มากคือมีค่าใกล้เคียงกับเมื่อขนาดหน่วยความจำแคชเป็น 4 กิโลไบต์



รูป 7.3 แสดงอัตราส่วนการพลาดโดยเฉลี่ยในแต่ละโปรแกรม เมื่อแปรขนาดหน่วยความจำแคช

ผลจากการพิจารณาค่าสมรรถนะสัมพัทธ์และอัตราส่วนการพลาดในกรณีศึกษานี้พอจะกล่าวได้ว่า การเพิ่มขนาดหน่วยความจำแคชเกินกว่า 4 กิโลไบต์อาจไม่จำเป็นสำหรับโปรแกรมชุดที่เลือกใช้ในการทดสอบ เพราะขนาดหน่วยความจำแคช 8 กิโลไบต์ ไม่ได้ทำให้ค่าสมรรถนะสัมพัทธ์สูงขึ้น ซึ่งแสดงว่าการทำงานของหน่วยประมวลผลไม่ดีขึ้นและยังมีผลต่อการลดลงของอัตราส่วนการพลาดน้อยลง นั่นคือไม่ทำให้ประสิทธิผลจากการใช้งานหน่วยความจำแคชเพิ่มขึ้น

7.2 กรณีศึกษาเมื่อกำหนดมีหน่วยความจำแคชและแปรขนาดเส้นทางส่งข้อมูล

เนื่องจากการถ่ายเทข้อมูลระหว่างหน่วยความจำแคชและหน่วยความจำหลัก กระทำผ่านเส้นทางส่งข้อมูล ดังนั้นถ้าในแต่ละรอบการทำงานของหน่วยความจำหลักสามารถส่งผ่านข้อมูลได้มากขึ้น น่าจะทำให้ประสิทธิภาพของระบบดีขึ้นด้วย เช่นการกำหนดขนาดเส้นทางส่งผ่านข้อมูล 16 บิต ทำให้สามารถเข้าถึงข้อมูลได้ครั้งละ 2 ไบต์ต่อ 1 รอบการทำงานของหน่วยความจำหลัก ถ้าต้องการข้อมูล 4 ไบต์ ต้องเข้าถึงหน่วยความจำหลัก 2 รอบ ในขณะที่ถ้าเรากำหนดขนาดเส้นทางส่งผ่านข้อมูล 32 บิต ทำให้ 1 รอบการทำงานของหน่วยความจำ สามารถเข้าถึงข้อมูลได้พร้อมกัน 4 ไบต์ จะเห็นว่าการเพิ่มขนาดเส้นทางส่งผ่านข้อมูลช่วยลดจำนวนครั้งที่ต้องเข้าถึงหน่วยความจำหลัก ซึ่งเป็นผลให้หน่วยประมวลผลใช้เวลาในการเข้าถึงหน่วยความจำหลักน้อยลง และจากข้อสันนิษฐานที่ว่ามีการอ้างถึงหน่วยความจำหลักตามลำดับที่ต่อเนื่องกันด้วยเปอร์เซนต์สูง ดังนั้นถ้าจำนวนไบต์ข้อมูลที่ส่งมาจากหน่วยความจำหลักแต่ละครั้งมีมาก โอกาสที่จะพบข้อมูลในหน่วยความจำแคชโดยทันทีก็มีมากขึ้น นั่นคือการเพิ่มขนาดเส้นทางส่งข้อมูล ส่งผลดีต่อลักษณะการทำงานที่มีการอ้างอิงบางตำแหน่งซึ่งใกล้เคียงกับตำแหน่งที่อ้างอิงก่อนหน้านี้

ในการศึกษาเพื่อวัดเปรียบเทียบสมรรถนะของหน่วยประมวลผลต้นแบบ เมื่อแปรขนาดของเส้นทางส่งผ่านข้อมูลระหว่างหน่วยความจำแคชและหน่วยความจำหลัก ศึกษาจากเวลาที่ใช้ในการปฏิบัติการโปรแกรมกรณีไม่มีหน่วยความจำแคช เปรียบเทียบกับเวลาที่ใช้ในการปฏิบัติการโปรแกรมกรณีมีหน่วยความจำแคชขนาดต่างๆกัน โดยแปรเส้นทางส่งผ่านข้อมูลขนาด 16 และ 32 บิต ดังแสดงในตารางที่ 7.1 และยังได้แสดงอัตราส่วนการพลาดโดยเฉลี่ยในการใช้งานหน่วยความจำแคชของแต่ละโปรแกรมที่ใช้ทดสอบ ดังแสดงในตาราง 7.2 พร้อมกันนี้ก็จะแสดงให้เห็นอัตราส่วนการพลาดในแต่ละช่วงการทำงานภายในโปรแกรม ดังแสดงในรูปกราฟที่ 7.4 - 7.8

โปรแกรมที่ใช้ในการทดสอบ	ขนาดเส้นทางส่งข้อมูล	ขนาดหน่วยความจำแคช (หน่วยเป็น กิโลไบต์)			
		1	2	4	8
Dhrystone	16 บิต	1.108481	1.135443	1.145231	1.155691
	32 บิต	1.089757	1.101215	1.101828	1.111135
Fibonacci	16 บิต	1.135794	1.135997	1.136026	1.136055
	32 บิต	1.077858	1.078024	1.078051	1.078079
Netflow	16 บิต	1.179498	1.183010	1.189043	1.189888
	32 บิต	1.115970	1.118904	1.123538	1.124034
Quicksort	16 บิต	1.248277	1.253863	1.259831	1.259833
	32 บิต	1.122547	1.125715	1.129141	1.129142
Spanfo	16 บิต	1.154142	1.165834	1.170642	1.171278
	32 บิต	1.106615	1.115572	1.118917	1.119526

ตารางที่ 7.1 แสดงค่าสมรรถนะสัมพัทธ์กับกรณีไม่มีหน่วยความจำแคชเมื่อแปรขนาดเส้นทางส่งข้อมูลขนาด 16 บิตและ 32 บิต

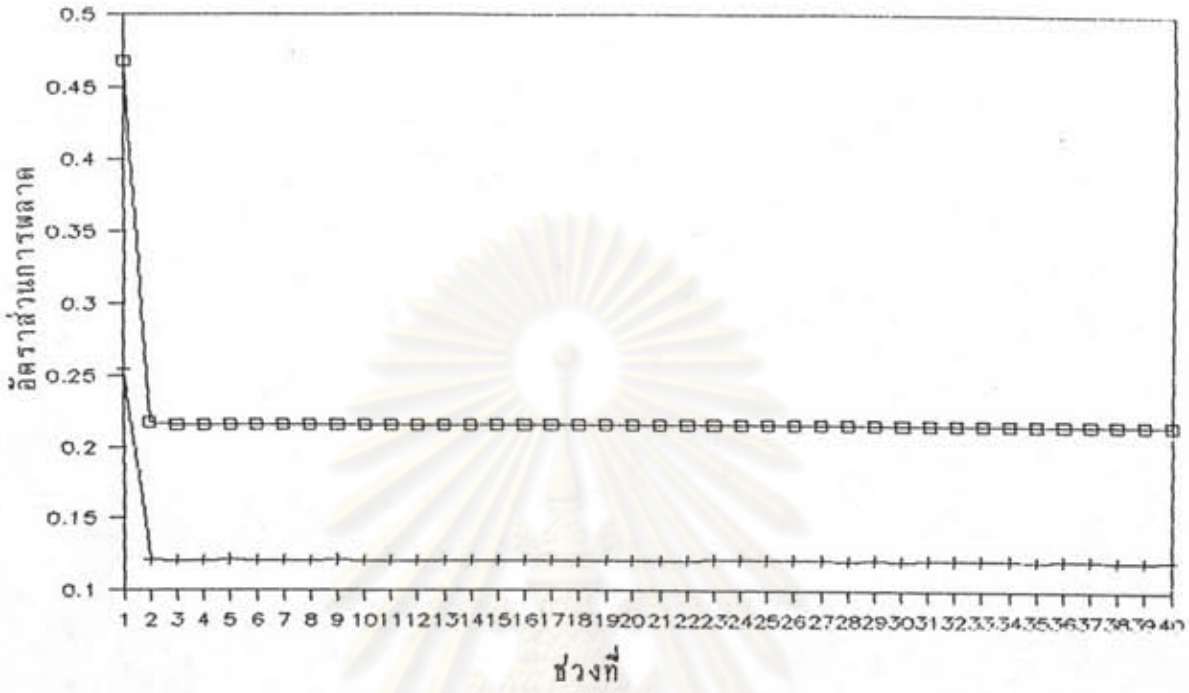
โปรแกรมที่ใช้ในการทดสอบ	ขนาดเส้นทางส่งข้อมูล	ขนาดหน่วยความจำแคช (หน่วยเป็น กิโลไบต์)			
		1	2	4	8
Dhrystone	16 บิต	0.220169	0.089033	0.048112	0.008988
	32 บิต	0.123875	0.050256	0.026606	0.004693
Fibonacci	16 บิต	0.021062	0.020803	0.020766	0.020359
	32 บิต	0.011394	0.011135	0.011098	0.010839
Netflow	16 บิต	0.096287	0.086426	0.062466	0.058794
	32 บิต	0.065348	0.056397	0.034661	0.031665
Quicksort	16 บิต	0.022882	0.014105	0.003778	0.003755
	32 บิต	0.014837	0.008841	0.001936	0.001917
Spanfo	16 บิต	0.073283	0.041306	0.027963	0.024436
	32 บิต	0.055306	0.027156	0.015839	0.012357

ตารางที่ 7.2 แสดงอัตราส่วนการพลาดโดยเฉลี่ยในแต่ละโปรแกรมเมื่อแปรขนาดเส้นทางส่งข้อมูลขนาด 16 บิตและ 32 บิต

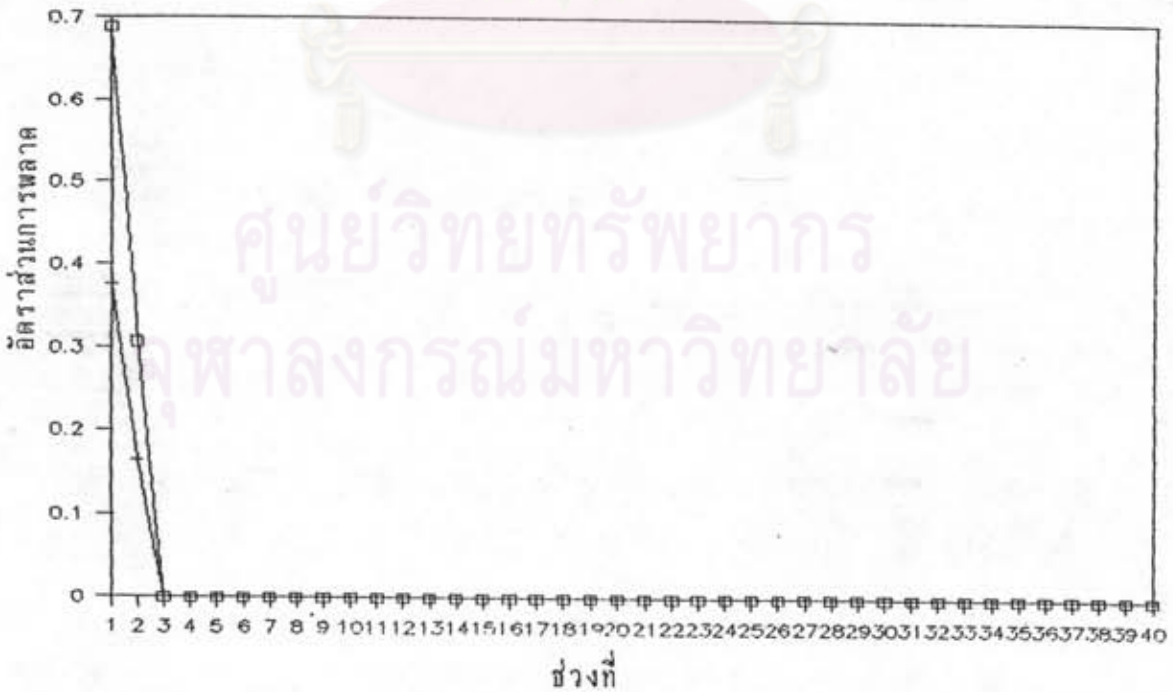
จากข้อมูลที่แสดงในตารางที่ 7.1 นี้ อาจสรุปได้ว่าการเพิ่มขนาดเส้นทางส่งข้อมูลจาก 16 บิตเป็น 32 บิต ให้ผลดีต่อการทำงานของระบบ เพราะการเพิ่มขนาดเส้นทางส่งข้อมูลช่วยให้มีจำนวนไบต์ข้อมูลถ่ายเทระหว่างหน่วยความจำแคชและหน่วยความจำหลักได้มากขึ้น ถ้าในการอ้างอิงข้อมูลครั้งต่อไปต่อเนื่องจากตำแหน่งของข้อมูลที่ได้มีการอ้างถึงก่อนแล้ว การเพิ่มขนาดเส้นทางส่งข้อมูลจะเป็นผลให้มีข้อมูลสำหรับกระทำการอยู่ในหน่วยความจำแคชแล้ว ดังนั้นจำนวนครั้งที่ต้องเข้าถึงข้อมูลในหน่วยความจำหลักจึงลดลง และทำให้หน่วยประมวลผลได้รับข้อมูลที่ต้องการโดยเสียเวลารอน้อยลงเพราะเข้าถึงข้อมูลในหน่วยความจำแคชได้ทันที

ข้อมูลในตารางที่ 7.2 ได้แสดงให้เห็นว่าการเพิ่มขนาดเส้นทางส่งข้อมูลจาก 16 บิต เป็น 32 บิต มีผลต่อค่าอัตราส่วนการพลาดโดยเฉลี่ยสำหรับทุกโปรแกรม คือทำให้อัตราส่วนการพลาดลดลงประมาณ 1.7 เท่า ซึ่งการที่หน่วยเชื่อมต่อบัสสามารถเข้าถึงข้อมูลในหน่วยความจำหลักได้มากขึ้นกว่าเดิมถึง 2 เท่า จึงทำให้หน่วยความจำแคชเก็บข้อมูลได้เพิ่มขึ้นจาก 2 ไบต์เป็น 4 ไบต์ นั่นคือมีข้อมูลที่อยู่ในตำแหน่งถัดไปจากตำแหน่งที่ต้องการขณะนั้นเข้าไปในหน่วยความจำแคชมากขึ้น ดังนั้นถ้าภายในโปรแกรมมีการปฏิบัติการชุดคำสั่งหรือเรียกใช้ข้อมูลตามลำดับต่อเนื่องกันจะช่วยให้อัตราส่วนการพลาดโดยเฉลี่ยมีค่าลดลง แต่จากการศึกษาวิเคราะห์ชุดคำสั่งในบทที่ 5 พบว่าทุกโปรแกรมมีการใช้งานประเภทคำสั่งย้ายการควบคุม นั่นคือมีการกระโดดข้ามคำสั่งในบางครั้งซึ่งทำให้ลำดับคำสั่งที่ปฏิบัติการไม่มีความต่อเนื่องกัน จึงทำให้อัตราส่วนการพลาดโดยเฉลี่ยลดลงเพียง 1.7 เท่า

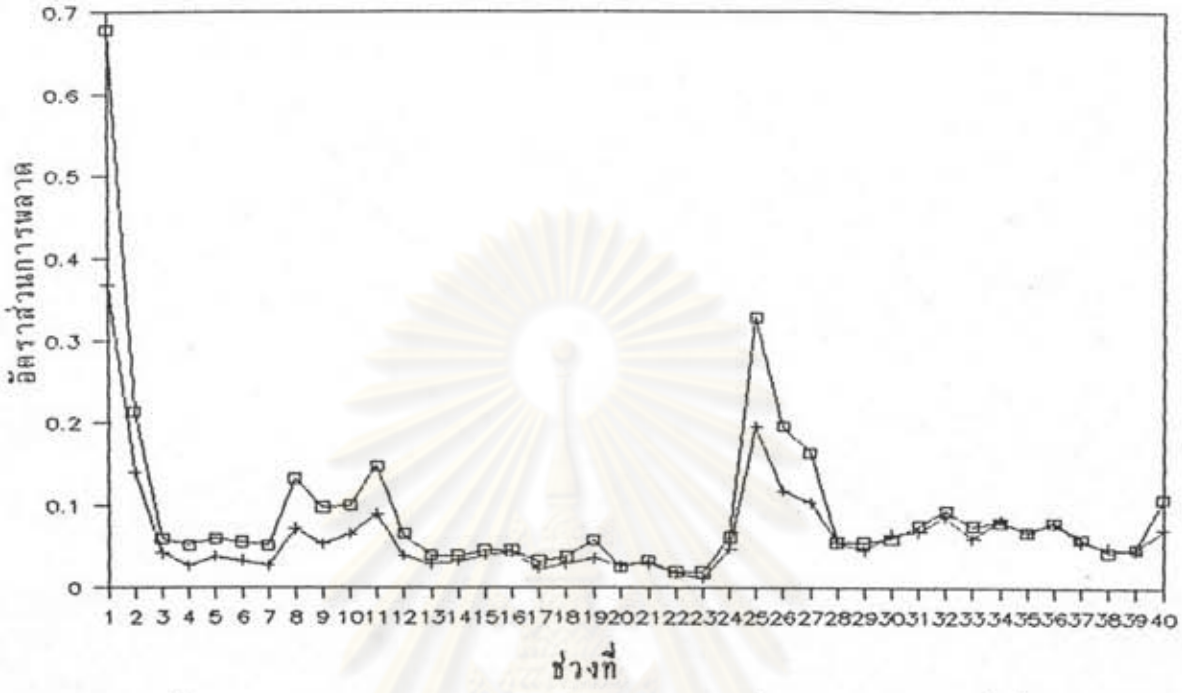
สำหรับการลดลงของอัตราส่วนการพลาดภายในแต่ละโปรแกรมจะมีความแตกต่างกันไปตามลักษณะการทำงานของแต่ละโปรแกรมโดยสามารถพิจารณาได้จากรูปกราฟที่ 7.4 - 7.8 ซึ่งแบ่งแสดงค่าอัตราส่วนนี้เป็น 40 ช่วงของจำนวนครั้งที่มีการอ้างอิงหน่วยความจำแคชในแนวนอนของกราฟ พร้อมกันนี้ได้แปรขนาดเส้นทางส่งข้อมูลในการแสดงค่าอัตราส่วนการพลาดเพื่อศึกษาเปรียบเทียบผลกระทบที่มีต่อกัน ซึ่งขนาดหน่วยความจำแคชที่กำหนดใช้ในการศึกษา คือ 1 กิโลไบต์ และในแต่ละรูปกราฟจะแทนขนาดเส้นทางส่งข้อมูล 16 บิตและ 32 บิตด้วยสัญลักษณ์ □ และ + ตามลำดับ โดยแต่ละโปรแกรมกำหนดจำนวนครั้งที่มีการอ้างอิงหน่วยความจำแคชในแต่ละช่วงดังนี้ โปรแกรม Dhrystone และ Spanfo กำหนดให้จำนวนครั้งการอ้างอิงหน่วยความจำแคชแต่ละช่วงเท่ากับ 2,500 ครั้ง โปรแกรม Fibonacci และ Netflow กำหนดให้จำนวนครั้งการอ้างอิงหน่วยความจำแคชแต่ละช่วงเท่ากับ 500 ครั้ง และโปรแกรม Quicksort กำหนดให้จำนวนครั้งการอ้างอิงหน่วยความจำแคชแต่ละช่วงเท่ากับ 5,000 ครั้ง



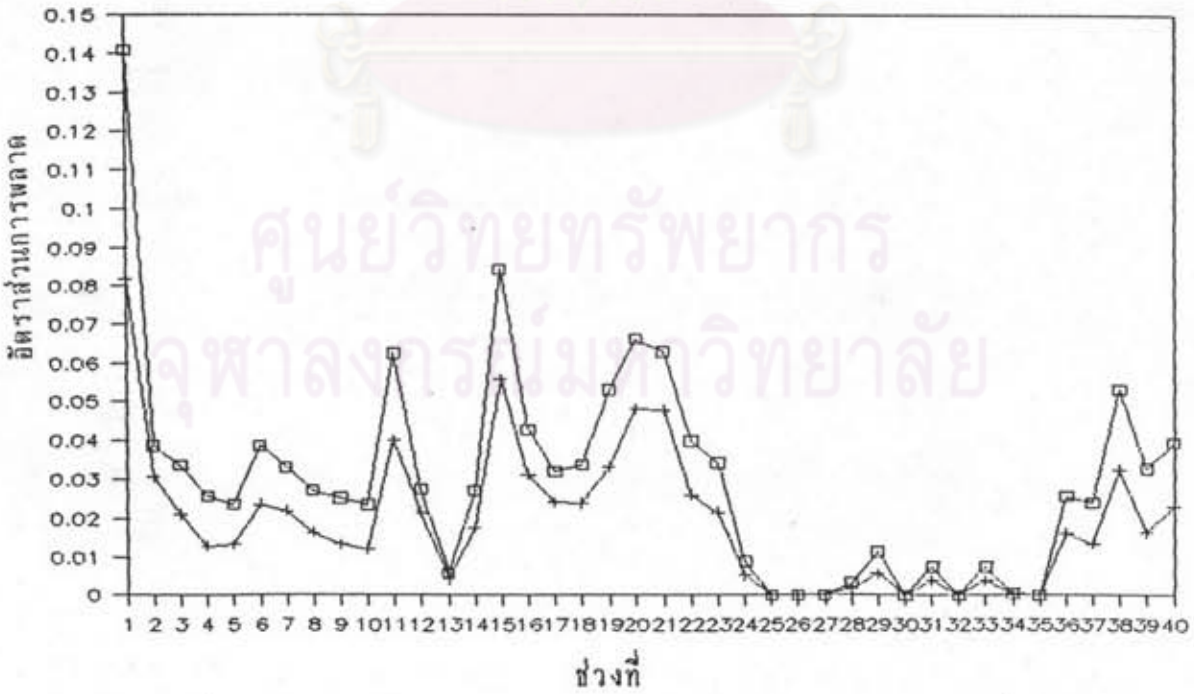
รูปที่ 7.4 แสดงอัตราส่วนการพลาดในแต่ละช่วงซึ่งแบ่งตามจำนวนครั้งที่มีการอ้างอิงหน่วยความจำแคชภายในโปรแกรม Dhrystone



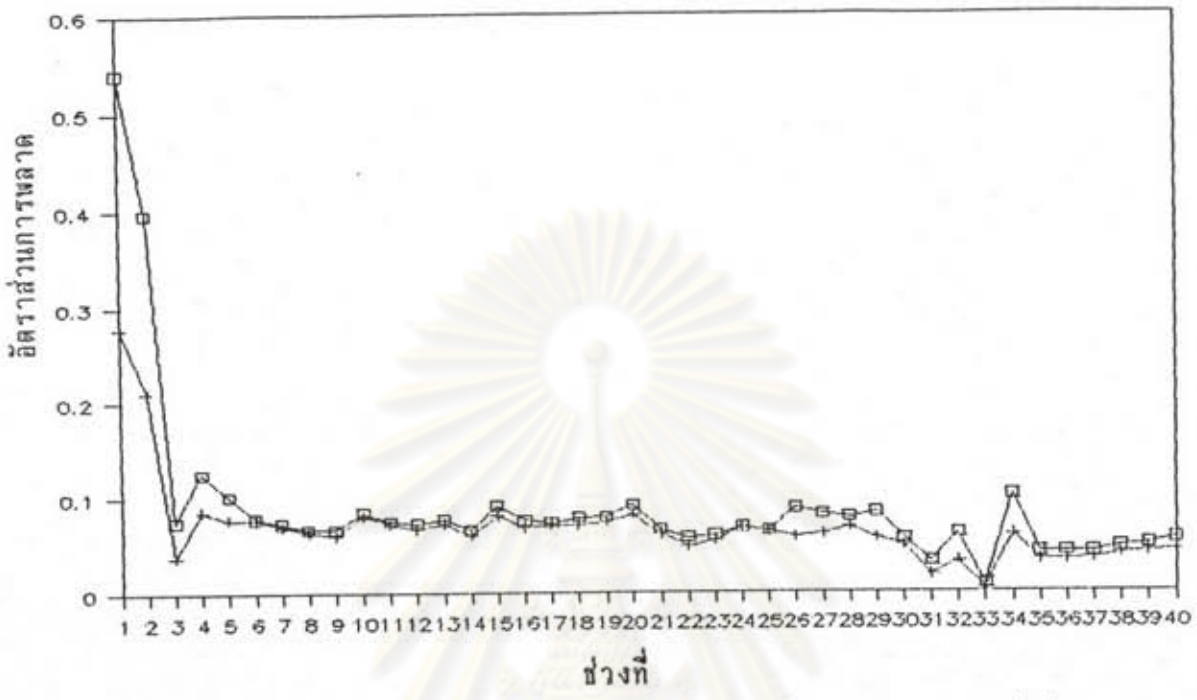
รูปที่ 7.5 แสดงอัตราส่วนการพลาดในแต่ละช่วงซึ่งแบ่งตามจำนวนครั้งที่มีการอ้างอิงหน่วยความจำแคชภายในโปรแกรม Fibonacci



รูปที่ 7.6 แสดงอัตราส่วนการพลาดในแต่ละช่วงซึ่งแบ่งตามจำนวนครั้งที่มีการอ้างอิงหน่วยความจำแคชภายในโปรแกรม Netflow



รูปที่ 7.7 แสดงอัตราส่วนการพลาดในแต่ละช่วงซึ่งแบ่งตามจำนวนครั้งที่มีการอ้างอิงหน่วยความจำแคชภายในโปรแกรม Quicksort



รูปที่ 7.8 แสดงอัตราส่วนการพลาดในแต่ละช่วงซึ่งแบ่งตามจำนวนครั้งที่มีการอ้างอิงหน่วยความจำแคชภายในโปรแกรม Spanfo

จากกราฟทั้ง 5 รูปซึ่งแสดงอัตราส่วนการพลาดในการทดสอบแต่ละโปรแกรม พบว่าการเพิ่มขนาดเส้นทางส่งข้อมูลจาก 16 บิตเป็น 32 บิตในทุกโปรแกรมยกเว้น Fibonacci จะมีผลให้อัตราส่วนการพลาดลดลง โดยอาจเนื่องจากเมื่อขนาดเส้นทางส่งข้อมูลเพิ่ม ทำให้จำนวนไบต์ข้อมูลที่ส่งผ่านบัสมายังหน่วยความจำแคชมีมากขึ้น ดังนั้นโอกาสที่จะพบข้อมูลที่ต้องใช้งานในลำดับต่อไปในหน่วยความจำแคชก็มีมากขึ้น เป็นผลให้ประสิทธิภาพการทำงานของหน่วยประมวลผลดีขึ้นด้วย แต่ในบางช่วงพบว่า ขนาดเส้นทางส่งข้อมูลทั้ง 2 ขนาดนั้นให้อัตราส่วนการพลาดเท่ากัน น่าจะเป็นผลจากการลักษณะการทำงานและโครงสร้างข้อมูลที่ใช้ภายในโปรแกรม

เมื่อพิจารณาผลจากการทดสอบโปรแกรม Dhrystone (ดูรูปที่ 7.4) พบว่าเมื่อเพิ่มขนาดเส้นทางส่งข้อมูล ทำให้อัตราส่วนการพลาดลดลง น่าจะเป็นเพราะภายในโปรแกรมมีการใช้งานข้อมูลชนิดตัวอักษรซึ่งมีโครงสร้างข้อมูลเป็นแบบแถวลำดับอย่างต่อเนื่องกัน และมีลำดับการปฏิบัติการคำสั่งต่อเนื่องกันโดยไม่เน้นเฉพาะไปที่ลักษณะการทำงานแบบเวียนเกิด ดังนั้นการเพิ่มขนาดเส้นทางส่งข้อมูลจึงเป็นผลให้อัตราส่วนการพลาดลดลงอย่างเห็นได้ชัด

แต่ในการทดสอบโปรแกรม Fibonacci (ดูรูปที่ 7.5) พบว่าอัตราส่วนการพลาดเป็นศูนย์ หลังจากมีการปฏิบัติการคำสั่งไปช่วงหนึ่ง นั่นคือทุกครั้งที่ต้องการข้อมูลจะพบในหน่วยความจำ

แคชได้ทันที อาจเป็นเพราะขนาดโปรแกรมมีขนาดเล็ก เป็นผลให้ข้อมูลที่ใช้ในการปฏิบัติการอยู่ในหน่วยความจำแคชได้ทั้งหมด และพบว่าการเพิ่มขนาดเส้นทางส่งข้อมูลมากขึ้นยังคงให้อัตราส่วนการพลาดในรูปแบบเดียวกันกับเมื่อขนาดเส้นทางส่งข้อมูลเท่ากับ 16 บิตซึ่งให้ค่าอัตราส่วนการพลาดต่ำที่สุดแล้ว

เมื่อทดสอบโปรแกรม Netflow (ดูรูปที่ 7.6) พบว่ามีบางช่วงการให้อัตราส่วนการพลาดสูงและให้อัตราส่วนนี้คงที่ในช่วงระยะหนึ่ง อาจเป็นเพราะโปรแกรมมีการเปลี่ยนตำแหน่งข้อมูลที่จะใช้ในการทำงาน จึงต้องถ่ายเทข้อมูลส่วนที่ต้องการใช้เข้ามาในหน่วยความจำแคชก่อน ซึ่งจุดนี้อาจทำให้เกิดอัตราส่วนการพลาดสูง ลักษณะดังกล่าวเรียกว่า การเปลี่ยน locality จากนั้นก็ใช้ข้อมูลจากหน่วยความจำแคชปฏิบัติการ และอาจต้องใช้ข้อมูลบางส่วนที่ไม่อยู่ในหน่วยความจำแคชด้วย จึงทำให้อัตราส่วนการพลาดลดลง แต่ยังคงมีการถ่ายเทข้อมูลระหว่างหน่วยความจำหลักและหน่วยความจำแคช เพราะโปรแกรมนี้มีขนาดใหญ่กว่าขนาดหน่วยความจำแคชประมาณ 4 เท่า ดังนั้นแต่ละตำแหน่งของหน่วยความจำแคชสามารถอ้างอิงได้ด้วยตำแหน่งจากหน่วยความจำหลัก 4 ตำแหน่ง และเมื่อเพิ่มขนาดเส้นทางส่งข้อมูลมากขึ้นพบว่าให้อัตราส่วนการพลาดที่ใกล้เคียงกัน แสดงว่าข้อมูลที่เข้ามานั้นไม่ได้ถูกใช้งานทั้งหมด

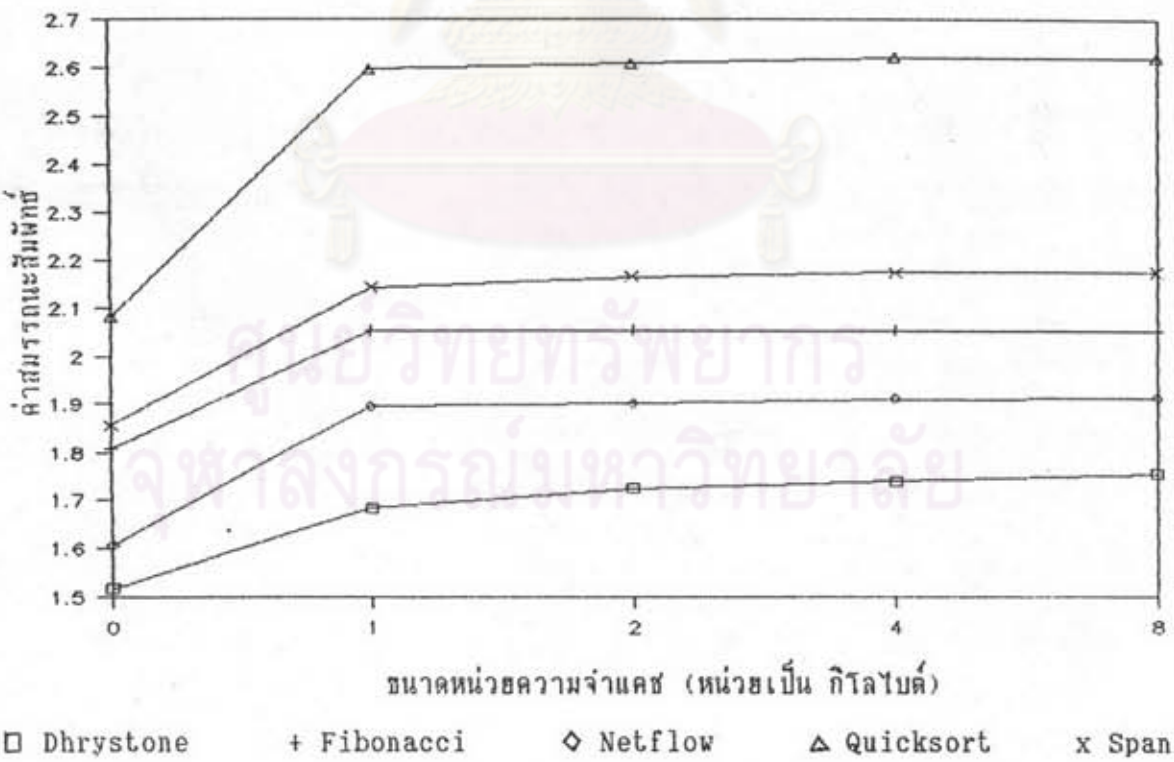
สำหรับการทดสอบโปรแกรม Quicksort (ดูรูปที่ 7.7) พบว่าการเพิ่มขนาดเส้นทางที่มีการถ่ายเทข้อมูลระหว่างหน่วยความจำแคชและหน่วยความจำเส้นทางส่งข้อมูลจาก 16 บิตเป็น 32 บิต ทำให้เกิดอัตราส่วนการพลาดลดลงโดยที่อัตราส่วนการพลาดไม่คงที่ ทั้งนี้อาจมาจากลักษณะการทำงานแบบเวียนเกิดจึงทำให้ต้องมีการดำเนินการกับข้อมูลอย่างไม่ต่อเนื่องเพราะต้องมีการกระโดดข้ามคำสั่ง และเนื่องจากโปรแกรมมีขนาดใหญ่เป็น 4 เท่าของขนาดหน่วยความจำแคช จึงยังคงมีการถ่ายเทข้อมูลระหว่างหน่วยความจำแคชและหน่วยความจำหลัก

และสุดท้ายเมื่อทดสอบโปรแกรม Spanfo (ดูรูปที่ 7.8) พบว่าการเพิ่มขนาดเส้นทางส่งข้อมูลจาก 16 บิตเป็น 32 บิต ให้อัตราส่วนการพลาดใกล้เคียงกัน แสดงว่าข้อมูลที่มีการถ่ายเทไปในแต่ละครั้ง หน่วยเชื่อมต่อบัสไม่ได้นำไปใช้งานทั้งหมด และเนื่องจากโปรแกรมมีขนาดใหญ่เป็น 5.5 เท่าของขนาดหน่วยความจำแคช จึงยังคงมีการถ่ายเทข้อมูลระหว่างหน่วยความจำแคชและหน่วยความจำหลัก

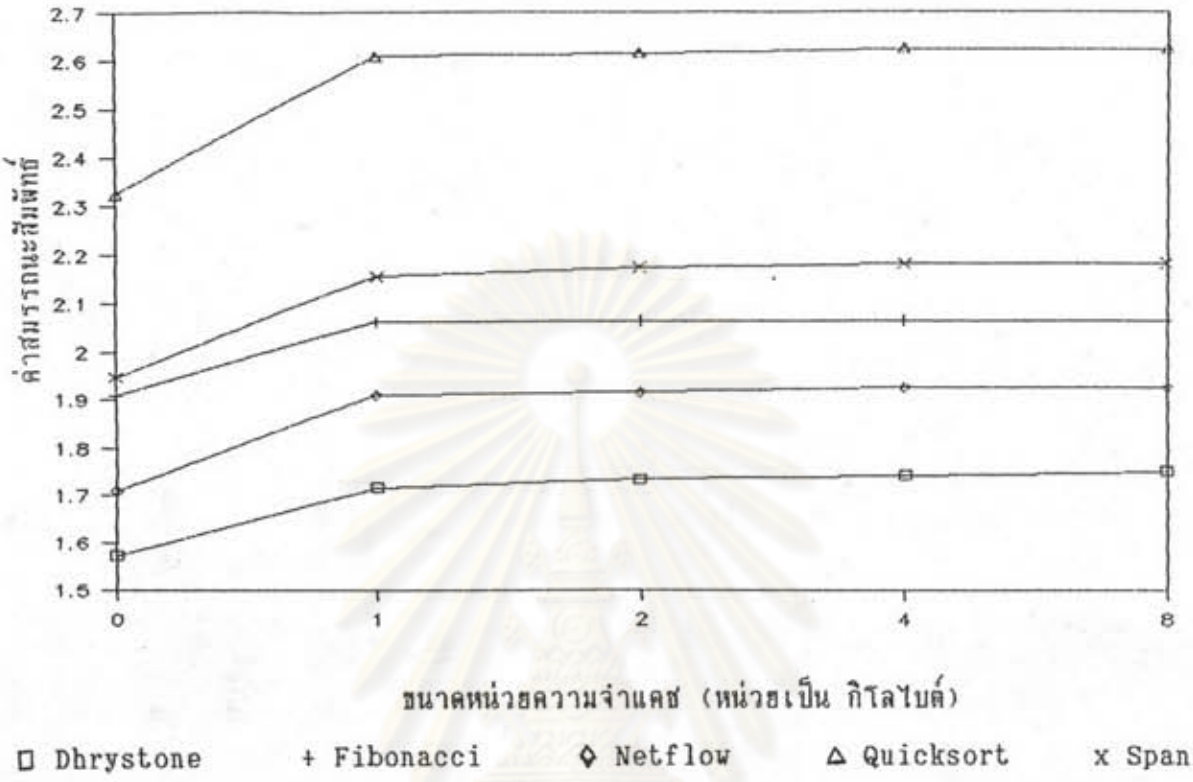
7.3 สรุป

จากผลการศึกษาที่ได้ในข้างต้นนี้ สรุปได้ว่าการกำหนดให้มีหน่วยความจำแคช ส่งผลให้หน่วยประมวลผลทำงานได้ดีขึ้น โดยที่พบว่าหน่วยความจำแคชที่ทำให้ค่าสมรรถนะสัมพัทธ์มีการเปลี่ยนแปลงดีขึ้นสำหรับการทดสอบด้วยโปรแกรมที่เลือกใช้ศึกษาในงานวิจัยนี้ คือหน่วยความจำแคชที่มีขนาดไม่เกิน 4 กิโลไบต์และการเพิ่มขนาดเส้นทางส่งข้อมูลจาก 16 บิตเป็น 32 บิต มีผลให้ค่าสมรรถนะสัมพัทธ์ของหน่วยประมวลผลดีขึ้น และส่งผลให้อัตราส่วนการพลาดโดยเฉลี่ยมีค่าลดลง 1.7 เท่า

เมื่อศึกษาภาพรวมทั้งหมดในการเปรียบเทียบค่าสมรรถนะสัมพัทธ์ ซึ่งได้จากเวลาที่แบบจำลองใช้ในการปฏิบัติการโปรแกรมกรณีไม่มีคิวคำสั่งและไม่มีหน่วยความจำแคช กับกรณีที่มีการปรับเปลี่ยนส่วนประกอบในระบบคือการกำหนดให้มีคิวคำสั่งขนาด 6 ไบต์ การกำหนดให้มีหน่วยความจำแคชขนาดต่างๆ และการเพิ่มขนาดเส้นทางส่งข้อมูล ดังแสดงในรูปที่ 7.9 และรูปที่ 7.10



รูปที่ 7.9 แสดงค่าสมรรถนะสัมพัทธ์กับกรณีไม่มีทั้งคิวคำสั่งและหน่วยความจำแคช โดยกำหนดขนาดเส้นทางส่งข้อมูลเท่ากับ 16 บิต



รูปที่ 7.10 แสดงค่าสมรรถนะสัมพัทธ์กับกรณีไม่มีทั้งคิวคำสั่งและหน่วยความจำแคช โดยกำหนดขนาดเส้นทางส่งข้อมูลเท่ากับ 32 บิต

จากรูปทั้ง 2 นี้กล่าวได้ว่าถึงแม้การเพิ่มคิวคำสั่งในระบบ จะทำให้ค่าสมรรถนะสัมพัทธ์ดีขึ้นแล้ว แต่อาจเพิ่มประสิทธิภาพการทำงานของหน่วยประมวลผลได้อีก โดยการเพิ่มหน่วยความจำแคชในระบบ ซึ่งขนาดหน่วยความจำแคชที่เหมาะสมต่อสมรรถนะการทำงานภายในระบบ คือ 1 กิโลไบต์ เพราะการเพิ่มหน่วยความจำแคชมากกว่านี้ ไม่ทำให้ค่าสมรรถนะเปลี่ยนแปลงเพิ่มขึ้นมากนัก