บทที่ 5
การทดลอง

บทนี้กล่าวถึงการทดลองและผลที่ได้จากการทดลองเครื่องเสมือนแบบแสตก (Virtual stack machine) ที่ออกแบบขึ้น มีจุดประสงค์เพื่อทวนสอบ (Verification) เครื่องเสมือนแบบแสตก และวัดผลการทำงานในด้านต่างๆ

การทดลองทั้งหมดจะทำบนการจำลอง (Simulation) เครื่องเสมือนถูกออกแบบและพัฒนาด้วยภาษาเวอร์ริลอกในแบบจำลองระดับการถ่ายโอนเรจิสเตอร์ (RTL model) โดยใช้โปรแกรมวัดเปรียบเทียบสมรรถนะแบบจำนวนเต็มของแสตนฟอร์ด (Stanford's integer benchmark) ในการทดลองทั้งทางด้านการทวนสอบระบบ และการวัดผล

การวัดผลการทำงานจะวัดผลของเครื่องเสมือนเปรียบเทียบกับระบบอ้างอิงพื้นฐาน (Baseline system) ซึ่งเป็นระบบฝังตัวที่ใช้หน่วยประมวลผล C1 ควบคุมโดยตรง โดยผลที่สนใจประกอบด้วยอัตราการบีบอัดของโปรแกรมวัดเปรียบเทียบสมรรถนะ และสมรรถนะของระบบ

จะกล่าวถึงรายละเอียดของโปรแกรมเพื่อวัดเปรียบเทียบที่นำมาใช้ในการทดลองเป็นอันดับแรก จากนั้นจะกล่าวถึงการแปลโปรแกรมให้อยู่ในรูปแบบที่นำมาใช้ในการจำลองในหัวข้อที่ 5.2 ในหัวข้อที่ 5.3 เป็นรายละเอียดการจำลอง ซึ่งจะกล่าวถึงสภาพแวดล้อมของระบบอ้างอิงพื้นฐานและเครื่องเสมือนแบบแสตก ผลการทดลองและข้อสรุปที่ได้จากการทดลองจะกล่าวถึงในหัวข้อที่ 5.4 และ 5.5 ตามลำดับ

5.1 โปรแกรมวัดเปรียบเทียบสมรรถนะ

โปรแกรมวัดเปรียบเทียบสมรรถนะ (Benchmark programs) ที่ใช้ในงานวิจัยนี้เป็นโปรแกรมวัดเปรียบเทียบสมรรถนะแบบตัวเลขจำนวนเต็มของแสตนฟอร์ด (Integer benchmark's Stanford) [25] มีด้วยกัน 7 โปรแกรมดังแสดงรายละเอียดในตารางที่ 5.1 ทุกโปรแกรมถูกเขียนให้อยู่ในรูปแบบภาษาส้ม โดยรายละเอียดของภาษาส้มจะอยู่ในภาคผนวก ก และโปรแกรมต้นฉบับของโปรแกรมวัดเปรียบเทียบสมรรถนะทั้งหมดจะแสดงไว้ในภาคผนวก ข

ตารางที่ 5.1 รายละเอียดโปรแกรมวัดเปรียบเทียบสมรรถนะจำนวนเต็มของแสตนฟอร์ด

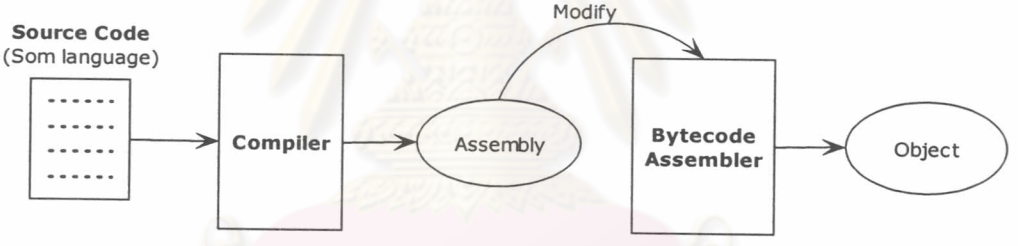
Benchmark	Description
bubble	Sort 20 numbers using bubble sort algorithm
quick	Sort 20 numbers using quick sort algorithm
hanoi	Find a solution to move 3 disks in "Tower of Hanoi"
sieve	Find all prime numbers less than 100
perm	Permute 4 digits of 0, 1, 2, 3
8-queen	Find a number of solution in "8-queen" problem
matmul	Multiply matrix 4×4

5.2 การแปลโปรแกรม

การแปลโปรแกรมคือโปรแกรมในภาษาระดับสูง (ภาษาส้ม) ให้อยู่ในรูปแบบของชุดคำสั่งรหัสไบต์ (Bytecode) และชุดคำสั่งเดิม (Native code) ของหน่วยประมวลผล C1 เพื่อนำมาใช้ในการจำลองเครื่องเสมือนแบบแสตกและระบบอ้างอิงพื้นฐานตามลำดับ

5.2.1 การแปลโปรแกรมให้เป็นชุดคำสั่งรหัสไบต์

เครื่องมือที่ใช้ในการแปลโปรแกรมจากภาษาส้มให้เป็นชุดคำสั่งรหัสไบต์ประกอบด้วยตัวแปลโปรแกรม (Compiler) และแอสเซมเบลอร์ของคำสั่งรหัสไบต์ (Bytecode assembler) ขั้นตอนทั้งหมดแสดงดังรูปที่ 5.1 โดยเริ่มจากนำโปรแกรมต้นฉบับในภาษาส้มมาแปลโดยใช้ตัวแปลโปรแกรม (Compiler) จะได้โปรแกรมในรูปแบบภาษาแอสเซมบลีของคำสั่งรหัสไบต์ (Bytecode assembly) หลังจากนั้นนำโปรแกรมดังกล่าวมาดัดแปลงในส่วนการแสดงผลลัพธ์การทำงาน และสุดท้ายนำโปรแกรมแอสเซมบลีดังกล่าวมาแปลงให้เป็นรหัสจุดหมายของคำสั่งรหัสไบต์ (Bytecode object) ด้วยแอสเซมเบลอร์ ก็จะได้โปรแกรมที่พร้อมนำไปใช้กับการจำลองเครื่องเสมือนแบบแสตก



รูปที่ 5.1 ขั้นตอนการแปลโปรแกรมให้อยู่ในรูปแบบชุดคำสั่งแบบแสตก

5.2.2 การแปลโปรแกรมให้เป็นชุดคำสั่งเดิม

สำหรับเครื่องมือที่ใช้ในการแปลโปรแกรมให้อยู่ในรูปแบบชุดคำสั่งของหน่วยประมวลผล C1 นั้นมีเพียงแอสเซมเบลอร์ของชุดคำสั่งเดิม ไม่มีตัวแปลโปรแกรมที่ใช้การแปลโปรแกรมภาษาส้มให้เป็นแอสเซมบลีหรือรหัสจุดหมายได้โดยตรง ดังนั้นจำเป็นต้องแปลจากภาษาส้มให้เป็นภาษาแอสเซมบลีของหน่วยประมวลผล C1 ด้วยมือเสียก่อน จากนั้นจึงใช้แอสเซมเบลอร์แปลงให้เป็นรหัสจุดหมายเพื่อนำไปใช้ในการจำลองการทำงานของระบบอ้างอิงพื้นฐาน รายละเอียดการแปลงโปรแกรมให้เป็นภาษาแอสเซมบลีกล่าวไว้ในภาคผนวก ค

5.3 การทดลอง

การทดลองทั้งหมดทำบนการจำลองการทำงาน (Simulation) ของเครื่องเสมือนแบบแสตกโดยมีจุดประสงค์เพื่อ*ทวนสอบระบบ* และ*วัดผลการทำงาน*ในด้านต่างๆ

ในด้านการทวนสอบระบบจะตรวจสอบความถูกต้องของการทำงาน 2 ประการได้แก่

1. ตรวจสอบผลลัพธ์ที่ได้ของโปรแกรมนั้นๆ เช่นโปรแกรม bubble สามารถเรียงลำดับข้อมูลได้อย่างถูกต้องหรือไม่
2. ตรวจสอบความถูกต้องของการทำงานในแต่ละคำสั่งว่าทำงานได้ถูกต้องตามที่ออกแบบหรือไม่ ซึ่งใช้หลักของการตรวจสอบตัวเอง (Self-checking) [26] ซึ่งเป็นการตรวจสอบผลการทำงานของวงจรที่ต้องการทดสอบกับแบบจำลองอ้างอิง (Reference model) ว่าผลการทำงานของวงจรที่ออกแบบมาทำงานถูกต้องหรือไม่ โดยแบบจำลองอ้างอิงเป็นข้อกำหนด (Specification) ของวงจรที่ต้องการทดสอบที่อยู่ในแบบจำลองเชิงพฤติกรรม (Behavioral model) ของวงจรที่นำมาทดสอบ

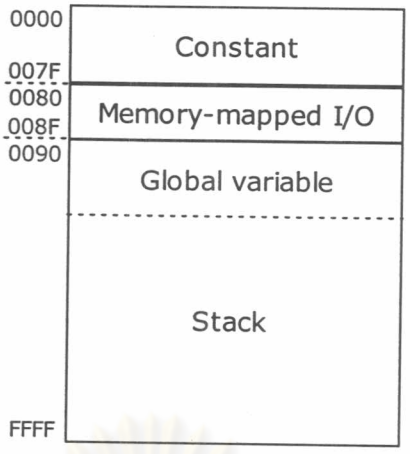
การวัดผลการทำงานใช้ผลการทำงานที่ได้จากการจำลองเครื่องเสมือนแบบแสตกมาเปรียบเทียบกับผลการทำงานที่ได้จากระบบอ้างอิงพื้นฐาน (Baseline system) โดยนำโปรแกรมวัดเปรียบเทียบสมรรถนะที่ผ่านการแปลโปรแกรมให้อยู่ในรูปแบบรหัสจุดหมายของแต่ละระบบมาใช้กับระบบการจำลองการทำงานของทั้งสองระบบ โดยมีจุดประสงค์ดังนี้

1. วัดขนาดของโปรแกรมที่อยู่ในรูปแบบของคำสั่งเดิม และคำสั่งรหัสไบต์ เพื่อนำมาหาอัตราการบีบอัดของโปรแกรมเปรียบเทียบสมรรถนะแต่ละโปรแกรม
2. นับจำนวนสัญญาณนาฬิกาที่ใช้ในการทำงานแต่ละโปรแกรม ในแต่ละระบบ
3. บันทึกจำนวนคำสั่งที่ทำงานทั้งหมด โดยถ้าเป็นระบบเครื่องเสมือนจะบันทึกทั้งจำนวนคำสั่งรหัสไบต์ และจำนวนคำสั่งเดิมที่ใช้ในการทำงานทั้งหมด แต่สำหรับระบบอ้างอิงพื้นฐานจะบันทึกเฉพาะจำนวนคำสั่งเดิมเท่านั้น

การจำลองการทำงานทั้งหมดจะทำอยู่บนโปรแกรม ModelSim XE II รุ่น 5.7g ซึ่งพัฒนาโดยบริษัท Xillinx

5.3.1 พื้นที่เลขที่อยู่ของหน่วยความจำข้อมูล

ในการจำลองการทำงานของระบบทั้งสองได้ออกแบบพื้นที่ของหน่วยความจำข้อมูล (Address space) ออกเป็น 3 ส่วนได้แก่พื้นที่เก็บค่าคงที่ (Constant segment) พื้นที่ Memory-mapped I/O และพื้นที่เก็บข้อมูลซึ่งแบ่งออกเป็นอีก 2 ส่วนย่อยใช้ร่วมกันได้แก่พื้นที่เก็บตัวแปรส่วนกลาง และพื้นที่แสตก ดังแสดงในรูปที่ 5.2



รูปที่ 5.2 โครงสร้างหน่วยความจำที่ใช้ในการจำลองการทำงาน

Memory-mapped I/O [27] เป็นหลักการในการเข้าถึงอุปกรณ์อินพุต/เอาต์พุต (I/O device) ซึ่งในการจำลองใช้ในการแสดงผลลัพธ์การทำงานผ่านการเขียนข้อมูลด้วยคำสั่งสโตร (Store) ลงในตำแหน่งเลขที่อยู่ที่กำหนดดังตารางที่ 5.2

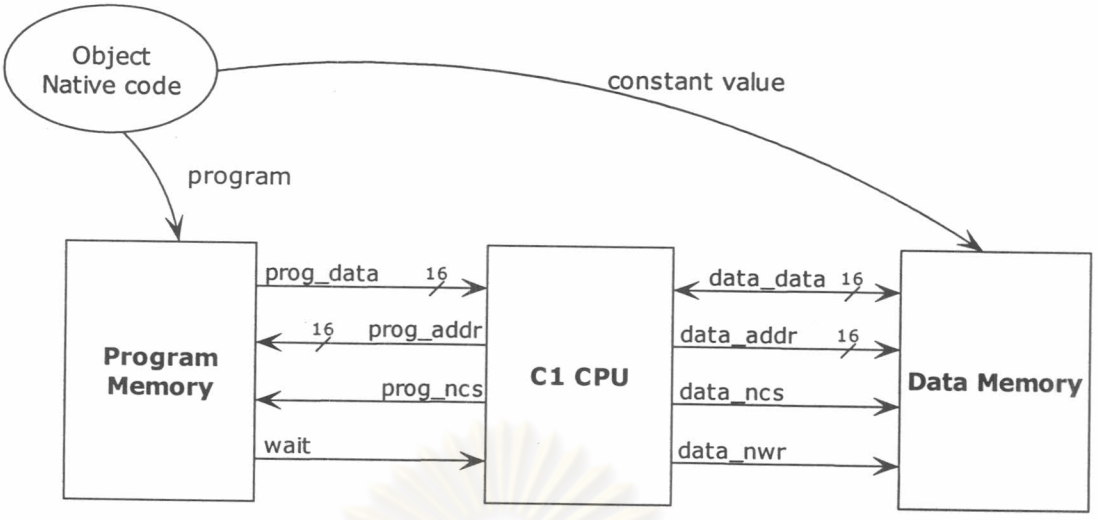
ตารางที่ 5.2 รายละเอียดการใช้ Memory-mapped I/O ในตำแหน่งต่างๆ

Address (in HEX)	Description
0080	Display data in integer format
0081	Display data in string format

พื้นที่แสตกเริ่มทำงานจากตำแหน่งสูงไปตำแหน่งต่ำ โดยเริ่มต้นที่ตำแหน่ง FFFFh ถ้ามีการดันข้อมูลลงแสตก ค่าของตัวชี้แสตกจะมีค่าลดลง และในทางกลับกันถ้ามีการดึงข้อมูลจากแสตก ค่าของตัวชี้แสตกจะมีค่าเพิ่มขึ้น ส่วนพื้นที่สำหรับตัวแปรส่วนกลางจะเริ่มต้นที่ตำแหน่ง 0090h เป็นต้นไป ซึ่งสังเกตได้ว่าพื้นที่แสตกจะลดลงจากตำแหน่งสูงไปต่ำและพื้นที่ตัวแปรส่วนกลางจะเพิ่มขึ้นจากตำแหน่งต่ำไปสูงทำให้ข้อมูลของทั้งสองส่วนไม่ซ้อนทับกัน

5.3.2 การจำลองระบบอ้างอิงพื้นฐาน

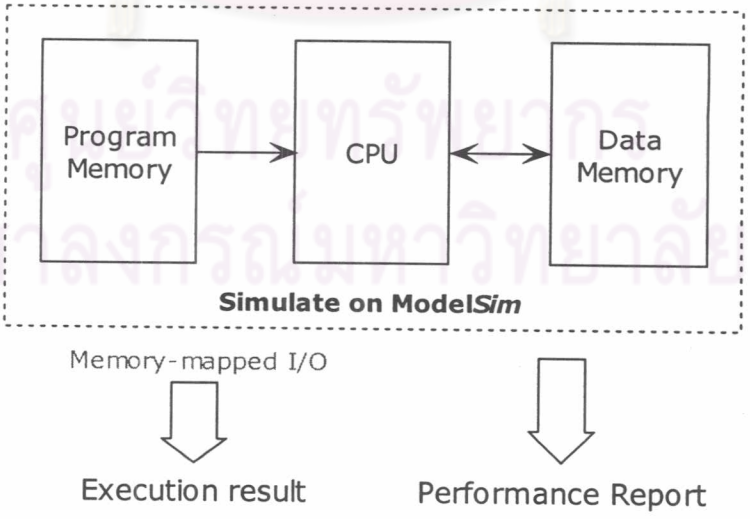
ระบบอ้างอิงพื้นฐาน (Baseline system) เป็นระบบฝังตัวที่ใช้หน่วยประมวลผล C1 ควบคุมการทำงาน ผลจากการทำงานที่ได้จะนำมาใช้ในการเปรียบเทียบกับผลการทำงานของเครื่องเสมือน



รูปที่ 5.3 โครงสร้างระบบฝังตัวธรรมดาที่ใช้ในการจำลองการทำงาน

จากรูปที่ 5.3 โครงสร้างของระบบอ้างอิงที่ใช้ในการจำลอง ประกอบด้วยหน่วยประมวลผล C1 หน่วยความจำโปรแกรม และหน่วยความจำข้อมูล โดยที่หน่วยประมวลผล C1 นั้นถูกพัฒนาในระดับถ่ายโอนเรจิสเตอร์ ส่วนหน่วยความจำทั้งสองส่วนนั้นพัฒนาด้วยแบบจำลองเชิงพฤติกรรม (Behavioral model) ซึ่งมีพฤติกรรมเหมือนหน่วยความจำจริง และพื้นที่หน่วยความจำข้อมูลมีโครงสร้างดังรูปที่ 5.2

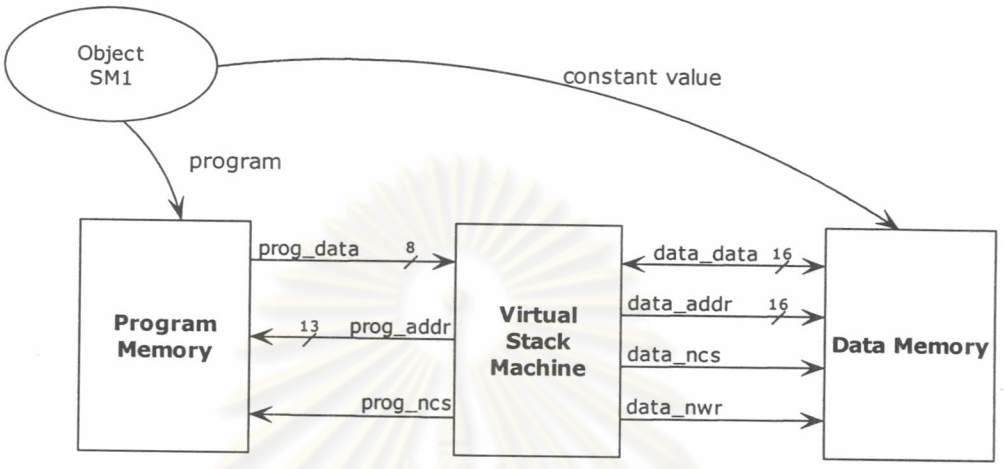
การจำลองระบบอ้างอิงพื้นฐานแสดงได้ดังรูปที่ 5.4 ผลลัพธ์ของโปรแกรมจะแสดงออกมาในการจำลองการทำงานผ่านทาง Memory-mapped I/O และภายหลังการทำงานของโปรแกรมเสร็จสิ้นระบบจำลองการทำงานจะรายงานสมรรถนะของระบบที่ใช้ในการทำงานโปรแกรมนั้นๆ ซึ่งจะแบ่งออกเป็นจำนวนสัญญาณนาฬิกาและจำนวนคำสั่งที่ถูกประมวลผลทั้งหมด



รูปที่ 5.4 แผนผังการตรวจสอบในการจำลองการทำงานของระบบอ้างอิงพื้นฐาน

5.3.3 การจำลองการทำงานของระบบเครื่องเสมือน

โครงสร้างของระบบเครื่องเสมือนที่ใช้ในการจำลองการทำงานเป็นไปดังรูปที่ 5.5 ซึ่งจะคล้ายกับการจำลองการทำงานของระบบฝังตัวพื้นฐาน เพียงแต่ขนาดของหน่วยความจำโปรแกรมนั้นมีความกว้างเพียง 8 บิตและมีจำนวน 8 กิโลไบต์



รูปที่ 5.5 โครงสร้างระบบเครื่องเสมือนที่ใช้ในการจำลองการทำงาน

การทวนสอบระบบแบ่งออกเป็น 2 การทดสอบได้แก่การตรวจสอบผลลัพธ์ของแต่ละโปรแกรมที่แสดงผ่าน Memory-mapped I/O และการตรวจสอบความถูกต้องในการประมวลผลคำสั่งแต่ละคำสั่ง

การตรวจสอบความถูกต้องของการประมวลผลแต่ละคำสั่งใช้หลักการตรวจสอบตัวเองโดยสร้างแบบจำลองอ้างอิง (Reference model) เพื่อใช้ในการอ้างอิงการทำงานขึ้น 2 แบบจำลองได้แก่แบบจำลองคอมพิวเตอร์แบบแสตก (Stack machine reference model) และแบบจำลองระบบเครื่องเสมือนแบบแสตก (Virtual stack machine reference model) ซึ่งประกอบไปด้วยแบบจำลองวงจรแปลคำสั่ง และแบบจำลองหน่วยประมวลผล C1 แบบจำลองทั้งหมดถูกสร้างเพื่ออธิบายคุณลักษณะของระบบที่ต้องการทดสอบ ดังนี้

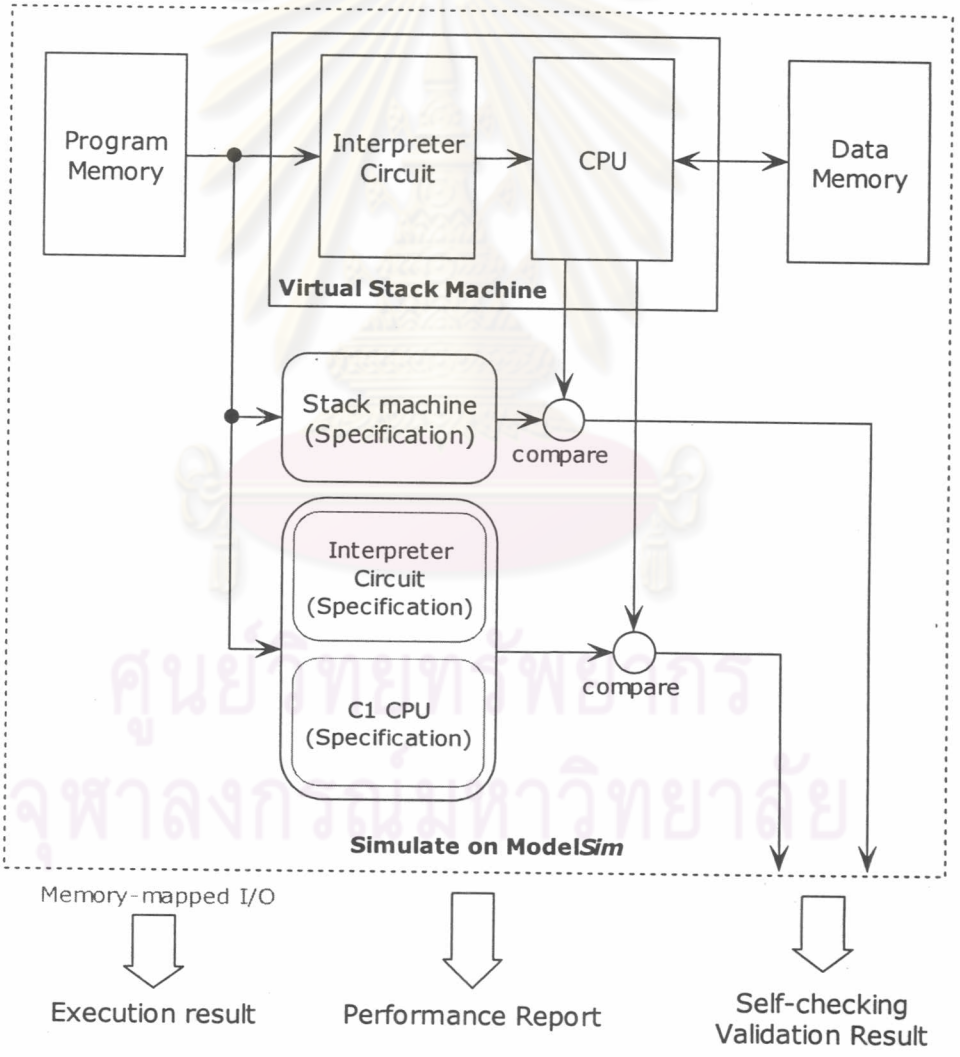
1. แบบจำลองคอมพิวเตอร์แบบแสตกเป็นแบบจำลองที่อธิบายข้อกำหนด (Specification) การทำงานของคอมพิวเตอร์แบบแสตกที่ใช้ชุดคำสั่งแบบแสตก SM1 ในการประมวลผล
2. แบบจำลองเครื่องเสมือนแบบแสตกเป็นแบบจำลองที่อธิบายข้อกำหนดของเครื่องเสมือนแบบแสตก โดยประกอบไปด้วยแบบจำลองย่อยอีก 2 แบบจำลองได้แก่

    2.1 แบบจำลองวงจรแปลคำสั่ง

    2.2 แบบจำลองหน่วยประมวลผล C1

จากรูปที่ 5.6 เห็นได้ว่าการจำลองการทำงานจะเปรียบเทียบผลที่ได้จากการประมวลผลของแต่ละคำสั่งรหัสไบต์และคำสั่งเดิมกับผลอ้างอิงที่ได้จากแบบจำลองอ้างอิงเชิงพฤติกรรมของเครื่องแสตก และวงจรอ้างอิงเชิงพฤติกรรมของเครื่องเสมือนแบบแสตกตามลำดับ ผลการเปรียบเทียบที่ได้จะแสดงหลังจากการทำงานของโปรแกรมเสร็จสิ้น ถ้าพบข้อผิดพลาดที่การประมวลผลคำสั่งใด ระบบจำลองการทำงานจะแสดงข้อผิดพลาดออกมา เพื่อนำมาพิจารณาแก้ไขความถูกต้องของวงจรและนำมาจำลองการทำงานใหม่อีกครั้งหนึ่ง

นอกจากนั้นระบบการจำลองการทำงานยังแสดงผลการทำงานของโปรแกรมผ่านทาง Memory-mapped I/O และภายหลังการทำงานของโปรแกรมเสร็จสิ้นระบบจำลองการทำงานจะรายงานจำนวนสัญญาณนาฬิกา จำนวนคำสั่งรหัสไบต์และคำสั่งเดิมที่ถูกประมวลผลในการทำงานแต่ละโปรแกรม



รูปที่ 5.6 แผนผังการตรวจสอบในการจำลองการทำงานของระบบฝังตัวที่ใช้เครื่องเสมือน

5.4 ผลการทดลอง

หัวข้อนี้แสดงผลการทดลองที่ได้จากโปรแกรมวัดเปรียบเทียบสมรถนะ โดยกล่าวถึงอัตราการบีบอัดที่แสดงถึงประสิทธิภาพของวิธีที่ใช้ในการลดขนาดของโปรแกรมที่ใช้ในวิทยานิพนธ์นี้เป็นอันดับแรก หลังจากนั้นกล่าวถึงสมรรถนะการทำงานของระบบเครื่องเสมือนเปรียบเทียบกับการทำงานของระบบอ้างอิงพื้นฐาน และรายละเอียดการสังเคราะห์วงจร (Synthesis) สุดท้ายเป็นการสรุปและวิจารณ์ผลที่ได้จากการทดลอง

5.4.1 อัตราการบีบอัด

อัตราการบีบอัดของโปรแกรมคำนวณได้ตามสมการที่ 2 โดยจะเป็นอัตราส่วนระหว่างขนาดของโปรแกรมในรูปแบบคำสั่งรหัสไบต์กับรูปแบบคำสั่งเดิม ซึ่งค่าอัตราการบีบอัดทั้งหมดแสดงดังตารางที่ 5.3 โดยมีค่าอยู่ระหว่าง 0.48 ถึง 0.79 คิดเป็นค่าเฉลี่ยเท่ากับ 0.63

ตารางที่ 5.3 อัตราบีบอัดของโปรแกรม

Benchmark	Native Code Size (Byte)	Bytecode Size (Byte)	Compression ratio
bubble	418	261	0.62
quick	570	347	0.61
hanoi	456	221	0.48
sieve	532	337	0.63
perm	432	268	0.62
8-queen	510	402	0.79
matmul	792	509	0.64

5.4.2 สมรรถนะการทำงาน

จำนวนสัญญาณนาฬิกาและจำนวนคำสั่งที่ใช้ในการทำงานของแต่ละโปรแกรมแสดงไว้ในตารางที่ 5.4 สมรรถนะการทำงานของระบบเครื่องเสมือนคำนวณจากอัตราส่วนระหว่างจำนวนสัญญาณนาฬิกาที่ใช้ในการทำงานของเครื่องเสมือนเทียบกับระบบอ้างอิงพื้นฐาน พบว่าเครื่องเสมือนทำงานช้ากว่าระบบอ้างอิงพื้นฐาน 2.12 ถึง 4.44 เท่า หรือคิดเป็นค่าเฉลี่ยเท่ากับ 3.10 เท่า

ตารางที่ 5.4 สมรรถภาพของระบบเครื่องเสมือนเทียบกับระบบอ้างอิงพื้นฐาน

Inst และ ByteInst คือจำนวนคำสั่งเดิมและคำสั่งรหัสไบต์ที่ถูกประมวลผลตามลำดับ และ Ratio คือ อัตราส่วนระหว่างจำนวนสัญญาณนาฬิกาในการทำงานของเครื่องเสมือนเทียบกับระบบอ้างอิงพื้นฐาน

Benchmark	Baseline		Virtual Stack Machine			Ratio
	Inst	Clock	ByteInst	Inst	Clock	
bubble	21,379	92,876	14,618	44,282	256,705	2.76
quick	6,877	29,598	5,202	16,473	94,210	3.18
hanoi	1,121	4,831	537	1,797	10,265	2.12
sieve	7,213	30,877	5,475	17,881	101,445	3.29
perm	14,083	60,760	10,103	32,931	187,130	3.08
8-queen	823,984	3,640,877	923,202	2,807,950	16,156,197	4.44
matmul	25,252	108,262	15,770	53,737	302,287	2.79

5.4.3 รายละเอียดการสังเคราะห์วงจร

ผลการนำวงจรแปลคำสั่งไปสังเคราะห์วงจรโดยใช้โปรแกรม Xilinx WebPack รุ่น 6.3i บนอุปกรณ์เอฟพีจีเอรุ่น Spartan 3 XC3S200 ได้รายละเอียดของวงจรดังตารางที่ 5.5

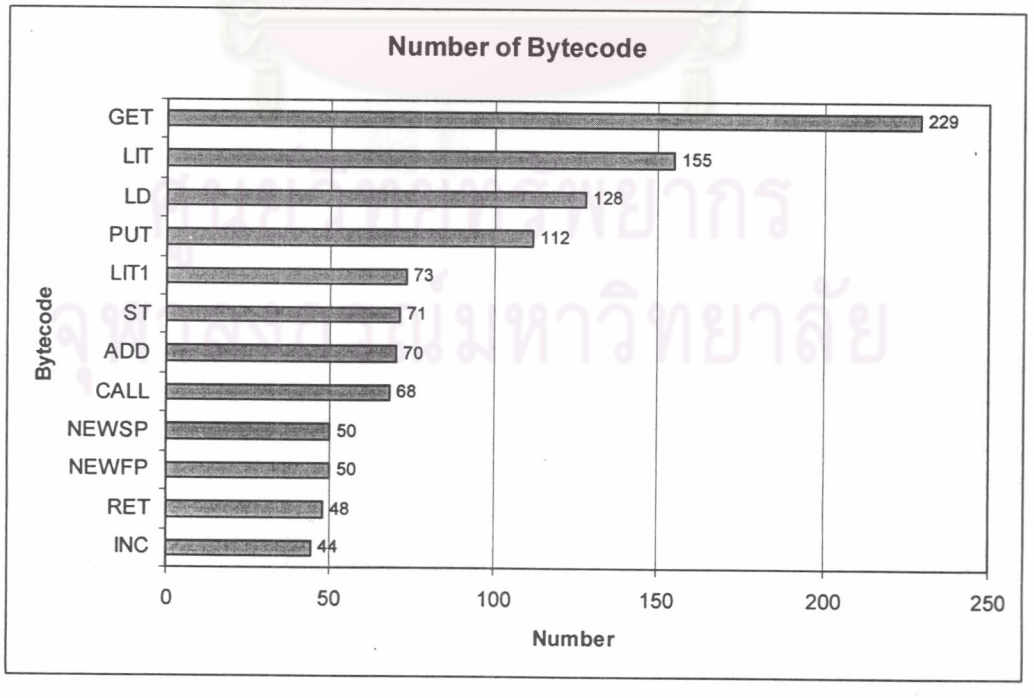
ตารางที่ 5.5 รายละเอียดของวงจรที่สังเคราะห์ได้

Device	Equivalent gates (#gates)	Maximum frequency (MHz)
Interpreter Circuit	2,175 gates	178.4MHz
C1 Processor	4,672 gates	72.7MHz
Total(Interpreter+C1)	6,625 gates	72.7MHz

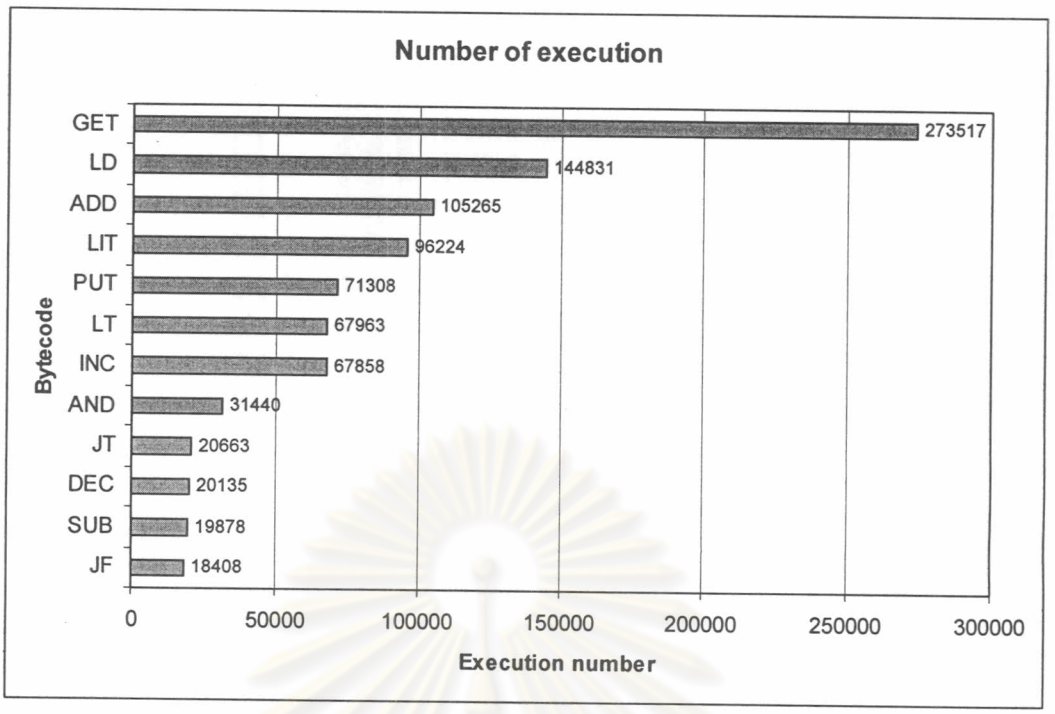
5.4.4 การปรับปรุงชุดคำสั่งรหัสไบต์

การปรับปรุงชุดคำสั่งรหัสไบต์มีจุดประสงค์หลักเพื่อลดขนาดโปรแกรมที่อยู่ในรูปแบบชุดคำสั่งรหัสไบต์ให้เล็กลง โดยมีแนวความคิดจากการพิจารณาหาคำสั่งหรือลำดับคำสั่งรหัสไบต์ที่ใช้บ่อย แล้วแทนคำสั่งหรือลำดับนั้นๆ ด้วยคำสั่งใหม่ที่มีขนาดเล็กกว่าเดิม ทำให้ขนาดของโปรแกรมโดยรวมมีขนาดลดลง

กราฟในรูปที่ 5.7 และ รูปที่ 5.8 แสดงจำนวนคำสั่งรหัสไบต์ที่พบในโปรแกรม และที่ถูกประมวลผลในการจำลองการทำงานตามลำดับ ซึ่งจะเห็นว่าจำนวนคำสั่งรหัสไบต์ที่พบมากที่สุด 4 อันดับแรกคือคำสั่ง GET, LIT, LD และ PUT ส่วนคำสั่งที่ถูกประมวลผลมากที่สุด 4 อันดับแรกคือ GET, LD, ADD และ LIT ดังนั้นถ้าต้องการลดขนาดของโปรแกรมหาคำสั่งอื่นๆ ที่มีขนาดเล็กกว่ามาใช้แทนคำสั่งที่ใช้บ่อยดังกล่าว



รูปที่ 5.7 กราฟแสดงจำนวนคำสั่งที่ใช้มากที่สุด 12 คำสั่ง



รูปที่ 5.8 กราฟแสดงจำนวนคำสั่งที่ประมวลผลมากที่สุด 12 คำสั่ง

ในงานวิจัยนี้ได้เพิ่มคำสั่งรหัสไบต์อีก 8 คำสั่งโดยพิจารณาถึงคำสั่งที่ใช้มากที่สุด 4 คำสั่งแรก โดยรายละเอียดของคำสั่งรหัสไบต์ใหม่นี้แสดงไว้ในตารางที่ 5.6

ตารางที่ 5.6 รายละเอียดคำสั่งรหัสไบต์ใหม่ที่เพิ่มเข้าไปในชุดคำสั่งรหัสไบต์

Mnemonic	Operand	Description	Format
GET1	-	Get local number 1	No operand
GET2	-	Get local number 2	No operand
GET3	-	Get local number 3	No operand
PUT1	-	Put local number 1	No operand
PUT2	-	Put local number 2	No operand
PUT3	-	Put local number 3	No operand
LDA	Address	Load data from address	2-Byte Operand
INCL	Local number	Increment local variable	1-Byte Operand

คำสั่งแรกที่ถูกพิจารณาได้แก่คำสั่ง GET และ PUT เนื่องจากคำสั่ง GET เป็นคำสั่งที่พบมากที่สุดในโปรแกรม และคำสั่ง PUT เป็นคำสั่งที่น่าจะใช้ควบคู่กับคำสั่ง GET และพบมากที่สุดเป็นอันดับ 4 โดยคำสั่งทั้งสองมีขนาดของคำสั่งคำสั่งละ 2 ไบต์ ดังนั้นในการทดลองจึงได้เพิ่มคำสั่ง GET1, GET2, GET3, PUT1, PUT2 และ PUT3 เข้าไปใช้แทนคำสั่ง GET และ PUT ที่มีตัวถูกดำเนินการ 1 ถึง 3

คำสั่งที่พิจารณาต่อมาคือคำสั่ง LIT และ LD ที่เป็นคำสั่งที่พบมากเป็นอันดับสองและสามตามลำดับ จากการสังเกตพบว่าคำสั่งทั้งสองนั้นมักจะถูกใช้ควบคู่กันในการโหลดข้อมูลที่เป็นตัวแปรส่วนกลาง ดังนั้นในการทดลองจึงใช้คำสั่ง LDA แทนการใช้ลำดับ LIT และ LD

สุดท้ายพิจารณาถึงการทำงานวนซ้ำ ซึ่งจำเป็นต้องมีการเพิ่มค่าให้กับตัวแปรเฉพาะที่ที่ทำหน้าที่ในการนับรอบการทำงานทุกๆ รอบการทำงาน ซึ่งเดิมจะต้องใช้คำสั่ง GET, INC และ PUT ตามลำดับ ดังนั้นจึงได้เพิ่มคำสั่ง INCL เข้าไปใช้แทนลำดับดังกล่าว

เมื่อเพิ่มคำสั่งใหม่และปรับปรุงโปรแกรมให้ใช้คำสั่งใหม่แทนคำสั่งรหัสไบต์แบบเดิมแล้ว บันทึกขนาดและอัตราการบีบอัดของโปรแกรมที่ใช้คำสั่งรหัสไบต์ที่ได้รับปรับปรุงนำมาแสดงในตารางที่ 5.7 นอกจากนั้นยังทดลองนำเอาโปรแกรมดังกล่าวไปใช้ในระบบเครื่องเสมือนเพื่อวัดสมรรถนะการทำงานของระบบ ซึ่งแสดงดังตารางที่ 5.8

ตารางที่ 5.7 อัตราการบีบอัดของโปรแกรมในชุดคำสั่งรหัสไบต์ที่ถูกปรับปรุง

Native, SM1 และ Ext-SM1 คือขนาดของโปรแกรมในรูปชุดคำสั่งเดิม คำสั่งรหัสไบต์ และคำสั่งรหัสไบต์ที่มีการเพิ่มคำสั่งใหม่เข้าไป ส่วน Com.ratio คือค่าอัตราการบีบอัด ซึ่งวัดออกมาสองค่าได้แก่อัตราการบีบอัดของโปรแกรมในรูปคำสั่งรหัสไบต์ที่ปรับปรุงเทียบกับคำสั่งรหัสไบต์แบบปกติและรูปชุดคำสั่งเดิม

Benchmark	Native (byte)	SM1 (byte)	Ext-SM1 (byte)	Com.ratio (Ext/SM1)	Com.ratio (Native/Ext)
bubble	418	261	208	0.80	0.50
quick	570	347	277	0.80	0.49
hanoi	456	221	190	0.86	0.42
sieve	532	337	272	0.81	0.51
perm	432	268	227	0.85	0.53
queen	510	402	337	0.84	0.66
matmul	792	509	432	0.85	0.55
bubble	418	261	208	0.80	0.50

ตารางที่ 5.8 สมรรถนะเปรียบเทียบระหว่างชุดคำสั่งรหัสไบต์ที่ปรับปรุงเทียบกับคำสั่งเดิม

SM1 และ Extended SM1 คือจำนวนสัญญาณนาฬิกาที่ใช้ในการทำงานของโปรแกรมในระบบเครื่องเสมือนธรรมดาและเครื่องเสมือนที่เพิ่มคำสั่งรหัสไบต์พิเศษเข้าไป และ Ratio คืออัตราส่วนระหว่างสัญญาณนาฬิกาของ SM1 กับ Extended SM1

Benchmark	SM1 (clock)	Extended SM1 (clock)	Ratio
bubble	256,705	242,675	0.95
quick	94,210	89,817	0.95
hanoi	10,265	10,045	0.98
sieve	101,445	95,941	0.95
perm	187,130	178,540	0.95
8-queen	16,156,197	15,515,145	0.96
matmul	302,287	270,549	0.90

จากตารางที่ 5.7 พบว่าอัตราการบีบอัดของโปรแกรมที่อยู่ในรูปแบบชุดคำสั่งรหัสไบต์ที่ได้รับการปรับปรุงเทียบกับชุดคำสั่งรหัสไบต์แบบเดิมมีค่าอยู่ระหว่าง 0.80 ถึง 0.86 หรือมีค่าเฉลี่ยเท่ากับ 0.83 และเมื่อเปรียบเทียบกับโปรแกรมที่อยู่ในรูปแบบชุดคำสั่งเดิมพบว่าได้อัตราการบีบอัด 0.42 ถึง 0.66 คิดเป็นค่าเฉลี่ยเท่ากับ 0.52 นอกจากนั้นระบบที่ได้รับการปรับปรุงให้รองรับ

ชุดคำสั่งรหัสไบต์ที่ได้รับการปรับปรุงทำงานได้เร็วขึ้น 2% ถึง 10% ของการทำงานของระบบเครื่องเสมือนปกติ คิดเป็นค่าเฉลี่ยเท่ากับ 5%

5.4.5 การทดลองการทำงานของชุดคำสั่งรหัสไบต์ด้วยชุดคำสั่งอื่น ๆ

เพื่อเป็นการยืนยันว่าคำสั่งรหัสไบต์สามารถอธิบายการทำงานได้ด้วยชุดคำสั่งของหน่วยประมวลผลแบบเรจิสเตอร์อื่นๆ ได้ ในวิทยานิพนธ์จึงนำเสนอการอธิบายคำสั่งรหัสไบต์ด้วยลำดับชุดคำสั่งของหน่วยประมวลผลอีกสองหน่วยประมวลผลได้แก่ หน่วยประมวลผล Q-Chip และหน่วยประมวลผล 8086/88

หน่วยประมวลผล Q-Chip [6] ถูกพัฒนาขึ้นเพื่อใช้ในการควบคุมการทำงานของเครื่องรับโทรทัศน์ และหน่วยประมวลผล 8086/88 ก็เป็นหน่วยประมวลผลที่ได้รับความนิยมในอดีต สาเหตุที่เลือกหน่วยประมวลผลสองตัวนี้มาเนื่องจากเป็นหน่วยประมวลผลขนาด 16 บิตซึ่งสามารถรองรับการทำงานของชุดคำสั่งรหัสไบต์ที่ประมวลผลข้อมูลได้ 16 บิตเช่นเดียวกัน

ลำดับชุดคำสั่งของหน่วยประมวลผลทั้งสองที่อธิบายการทำงานของคำสั่งรหัสไบต์แสดงไว้ในตารางตารางที่ ง.3 และตารางที่ ง.5 ในภาคผนวก ง

5.5 สรุปผลการทดลอง

จากการทดลอง ผลการทวนสอบ (Verification) ระบบเครื่องเสมือนแบบแสตกสรุปได้ว่าระบบสามารถทำงานกับคำสั่งรหัสไบต์ได้ตามที่ออกแบบไว้ โดยการทำงานด้วยระบบเครื่องเสมือนแบบแสตกให้อัตราการบีบอัดเฉลี่ยของโปรแกรมวัดเปรียบเทียบสมรรถนะแบบจำนวนเต็มของแสตนฟอร์ดเท่ากับ 0.63 แต่การทำงานของเครื่องเสมือนแบบแสตกช้าลงประมาณ 3 เท่าของระบบฝังตัวที่ให้หน่วยประมวลผลควบคุมโดยตรง

เมื่อเปรียบเทียบกับวิธีที่ใช้การลดขนาดโปรแกรมอื่นๆ แล้วพบว่าการแปลโปรแกรมให้อยู่ในรูปแบบของคำสั่งรหัสไบต์ให้อัตราการบีบอัดอยู่ในเกณฑ์ดี ส่วนสมรรถนะในการทำงานนั้นเมื่อเปรียบเทียบกับระบบที่ใช้วิธีการแปลโปรแกรมด้วยกัน [9, 10] พบว่าการแปลโปรแกรมด้วยวงจรแปลคำสั่งทำงานได้เร็วกว่าการใช้ซอฟต์แวร์แปลคำสั่ง (Software interpreter)

ส่วนเรื่องขนาดของวงจรคลายคำสั่ง (Decompressor) นั้นไม่มีงานวิจัยใดรายงานถึงขนาดของวงจรดังกล่าว จึงไม่สามารถเปรียบเทียบได้ว่าขนาดของวงจรแปลคำสั่งที่ออกแบบขึ้นใหญ่หรือเล็กกว่าวิธีการอื่นหรือไม่ แต่ถ้าพิจารณาจากขนาดของวงจรแปลคำสั่งดังกล่าวพบว่ามีขนาดที่เล็กพอสมควรเมื่อเปรียบเทียบกับขนาดของหน่วยความจำโปรแกรมที่ลดลงไปได้

ข้อเสียของงายวิจัยนี้อยู่ที่การแปลโปรแกรมเปรียบเทียบสมรรถนะให้อยู่ในรูปแบบของคำสั่งเดิม เนื่องจากไม่ได้ใช้วิธีที่เป็นมาตรฐาน อีกทั้งยังไม่ได้ใช้โปรแกรมแปลภาษาในการแปลโปรแกรมโดยตรง อาจทำให้เกิดคำสั่งที่ไม่จำเป็นในโปรแกรมที่แปลได้ ซึ่งส่งผลให้อัตราการบีบอัดสูงเกินควร แต่ไม่น่าสูงเกิน 5% ถึง 15%

แต่ถ้ามองในมุมกลับกันงานวิจัยนี้ก็สามารถเพิ่มอัตราการบีบอัดของโปรแกรมด้วยการเพิ่มคำสั่งรหัสไบต์ที่มีขนาดเล็กเข้าไปใช้แทนคำสั่งรหัสไบต์เดิมหรือลำดับของคำสั่งรหัสไบต์ที่มีการใช้บ่อย ซึ่งจะส่งผลให้โปรแกรมที่ได้มีขนาดเล็กลงไปอีก ดังในการทดลองนั้นพบว่าสามารถลดขนาดของโปรแกรมลงไปได้อีกประมาณ 15% ถึง 20%

นอกจากนั้นโปรแกรมเปรียบเทียบสมรรถนะที่ใช้ยังเป็นโปรแกรมขนาดเล็กและไม่เกี่ยวข้องกับระบบฝังตัว แต่เนื่องจากในการทดลองไม่มีโปรแกรมแปลภาษาสำหรับแปลให้อยู่ในรูปแบบชุดคำสั่งของหน่วยประมวลผล C1 จึงไม่สามารถทดลองกับโปรแกรมขนาดใหญ่ได้