ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ

นางสาวกมลลักษณ์ สุขเสน

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2556

WEB BASED CLIENT-SIDE COMPILER

Miss Kamonluk Suksen

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2013

หัวข้อวิทยานิพนธ์	ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ
โดย	นางสาวกมลลักษณ์ สุขเสน
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร.บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา)

.....กรรมการภายนอกมหาวิทยาลัย

(ดร.เฉลิมทรัพย์ สังขวิจิตร)

กมลลักษณ์ สุขเสน : ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ. (WEB BASED CLIENT-SIDE COMPILER) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ศ. ดร.ประภาส จงสถิตย์วัฒนา, 82 หน้า.

การพัฒนาโปรแกรมบนเว็บได้รับความนิยมมากขึ้นเนื่องจากโปรแกรมเมอร์สามารถพัฒนาโปรแกรมในสภาพแวดล้อมที่ไม่จำเป็นต้องติดตั้งโปรแกรมใดๆนอกเหนือไปจากเว็บเบราว์เซอร์ซึ่งเป็นโปรแกรมพื้นฐานของคอมพิวเตอร์ โน้ตบุ๊ค และอุปกรณ์เสริมอาทิ สมาร์ทโฟนและแท็บเล็ต เหตุผลดังกล่าวทำให้ช่วยลดเวลาในการพัฒนาโปรแกรมและการทำงานร่วมกับผู้อื่นเป็นไปอย่างสะดวก นอกจากนี้การพัฒนาโปรแกรมบนเว็บเบราว์เซอร์ยังช่วยให้โปรแกรมเมอร์สามารถพัฒนาโปรแกรมได้ทุกที่ทุกเวลา อย่างไรก็ตามงานวิจัยที่เกี่ยวกับการพัฒนาโปรแกรมบนเว็บส่วนมากรองรับการทำงานบนเครื่องผู้ให้บริการ ซึ่งจำเป็นต้องมีผู้ดูแลระบบและยังมีปัญหาเรื่องความเป็นส่วนตัวและความปลอดภัยของข้อมูล สำหรับงานวิจัยนี้ เรานำเสนอตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการเพื่อสนับสนุนการเรียนรู้การเขียนโปรแกรม ตัวแปลภาษาดังกล่าวเป็นตัวแปลภาษาที่ถูกพัฒนาโดยภาษาจาวาสคริปต์ซึ่งเป็นภาษามาตรฐานของทุกๆเว็บเบราว์เซอร์ทำให้ตัวแปลภาษาสามารถรองรับการทำงานบนเว็บเบราว์เซอร์ได้

วัตถุประสงค์ของงานวิจัยคือการนำเสนอวิธีการสำหรับพัฒนาโปรแกรมที่ทำงานบนเครื่องผู้รับบริการ และเป็นทางเลือกใหม่สำหรับนักเรียนในการเรียนรู้การเขียนโปรแกรมและการสร้างโปรเจกต์บนเว็บเบราว์เซอร์ การเขียนโปรแกรมบนเว็บเบราว์เซอร์จะเป็นประโยชน์อย่างมากสำหรับนักเรียนด้วยเหตุผลสำคัญสี่ประการ คือ นักเรียนสามารถเขียนโปรแกรมได้ทุกที่ทุกเวลา, สามารถทำงานบนระบบปฏิบัติการใดๆได้, ไม่จำเป็นต้องเปิดโปรแกรมจำนวนมากพร้อมกัน และสามารถทดสอบโปรแกรมได้อย่างรวดเร็วโดยปราศจากปัญหาเรื่องความเป็นส่วนตัวและความปลอดภัยของข้อมูลในโปรแกรม ด้วยเหตุผลเหล่านี้ทำให้ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการเหมาะอย่างยิ่งสำหรับสนับสนุนการเรียนรู้รายบุคคลในการเขียนโปรแกรม

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2556

5670104021 : MAJOR COMPUTER ENGINEERING
KEYWORDS: WEB BASED / COMPILER / INDIVIDUALIZED LEARNING

KAMONLUK SUKSEN: WEB BASED CLIENT-SIDE COMPILER. ADVISOR: PROF. PRABHAS CHONGSTITVATANA, Ph.D., 82 pp.

The program development on web browsers has become more popular since it allows programmers to experience writing code in an environment where they can see an output of their programs without needing any setup beyond a web browser. The advantages that mentioned above will often reduce development time and make collaboration easier. Furthermore, Web based programming allows programmers to develop programs anytime and anywhere. However, all researchers have developed ideas and tools for supporting online programming work only at the server side. So, it requires some level of administration to set up system and there are problems with privacy and security. In this paper, we present a web based client-side compiler for supporting individualized learning. The compiler is written in JavaScript language since it is present in essentially all web browsers.

The paper aims to offer a method for compiling programs on the client-side web browser and an option for students to learn programming and build their projects on the web browser. The web based programming is very useful for students in four aspects. Students can code anywhere with anyone. They can use any operating system and they don’t need to have many programs opened. Moreover, the program testing can be done faster without privacy and security problems. These reasons make this system ideal for supporting individualized learning in programming.

Department: Computer Engineering Student's Signature .....
Field of Study: Computer Engineering Advisor's Signature .....
Academic Year: 2013

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี ด้วยความช่วยเหลือของศาสตราจารย์ ดร. ประภาส จงสถิตย์วัฒนา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งท่านได้ให้คำแนะนำและข้อคิดเห็นต่างๆ อันเป็นประโยชน์อย่างยิ่งในการทำวิจัย อีกทั้งยังช่วยแก้ปัญหาต่างๆ ที่เกิดขึ้นระหว่างการดำเนินงานอีกด้วย ขอกราบขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณคณะกรรมการสอบวิทยานิพนธ์ทุกท่านเป็นอย่างสูง ได้แก่ผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ และ ดร.เฉลิมทรัพย์ สังขวิจิตร ที่สละเวลาอันมีค่ามาชี้ให้เห็นถึงข้อบกพร่อง พร้อมทั้งให้ข้อคิดและคำแนะนำอันเป็นประโยชน์อย่างยิ่งต่องานวิจัย

ขอขอบคุณพี่ๆทุกคนในห้องปฏิบัติการ ISL (Intelligent System Laboratory) ที่ให้คำแนะนำช่วยแก้ปัญหาต่างๆที่เกี่ยวข้องกับงานวิจัยนี้และปัญหาอื่นๆ รวมถึงให้กำลังใจตลอดมา

ขอขอบคุณเพื่อนๆในภาควิชาคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่สละเวลารับฟังปัญหาต่างๆ พร้อมทั้งแนะนำแนวคิดอันเป็นประโยชน์ต่องานวิจัยนี้

ขอขอบคุณเจ้าหน้าที่ประจำภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่านที่มีส่วนช่วยเหลือทำให้วิทยานิพนธ์ฉบับนี้สำเร็จเรียบร้อยลงด้วยดีทุกประการ

สุดท้ายนี้ ผู้วิจัยขอขอบพระคุณบิดามารดา และครอบครัว ซึ่งเปิดโอกาสให้ได้รับการศึกษาเล่าเรียน ตลอดจนคอยช่วยเหลือและให้กำลังใจผู้วิจัยเสมอมาจนสำเร็จการศึกษา

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การเขียนโปรแกรมบนเว็บได้รับความนิยมเป็นอย่างมากในปัจจุบัน [1] และมีงานวิจัยจำนวนมากที่ทำการวิจัยเกี่ยวกับการเขียนโปรแกรมบนเว็บ เช่น การเขียนโปรแกรมภาษาซีและภาษาซีพลัสพลัส [2] และการเขียนโปรแกรมภาษาจาวา [3] เป็นต้น นอกจากนี้ยังมีเว็บไซต์จำนวนมากที่ให้บริการเกี่ยวกับการเขียนโปรแกรมบนเว็บแบบออนไลน์ เช่น http://compileonline.com http://codepad.org http://c9.io และ http://shiftedit.net เป็นต้น ซึ่งความสะดวกของการเขียนโปรแกรมบนเว็บ คือ ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้ทุกที่ทุกเวลาเพียงแค่มีอุปกรณ์ที่รองรับการใช้งานเว็บเบราว์เซอร์ที่สามารถเชื่อมต่อกับอินเตอร์เน็ตได้ ได้แก่ คอมพิวเตอร์ โน้ตบุ๊ค สมาร์ทโฟน หรือแท็บเล็ต และผู้เขียนโปรแกรมสามารถใช้งานได้กับทุกระบบปฏิบัติการ เช่น ระบบปฏิบัติการวินโดวส์ ระบบปฏิบัติการลินุกซ์ และระบบปฏิบัติการแมค เป็นต้น ซึ่งระบบปฏิบัติการเหล่านี้รองรับการใช้งานเว็บเบราว์เซอร์และติดตั้งมาพร้อมกับระบบปฏิบัติการ ผู้เขียนโปรแกรมจึงไม่จำเป็นต้องติดตั้งเพิ่มเติม นอกจากนี้ผู้เขียนโปรแกรมยังสามารถทำการทดสอบโปรแกรมได้อย่างรวดเร็ว โดยการเปิดใช้งานเว็บเบราว์เซอร์เพื่อเขียนโปรแกรม ซึ่งเว็บเบราว์เซอร์ใช้เวลาในการเปิดใช้งานน้อยกว่าโปรแกรมอื่นๆ และไม่ต้องเสียเวลาในการติดตั้งโปรแกรมสำหรับการเขียนโปรแกรมเพิ่มเติม ด้วยความสะดวกเหล่านี้จึงทำให้การเขียนโปรแกรมบนเว็บได้กลายมาเป็นคู่แข่งที่สำคัญของการเขียนโปรแกรมบนคอมพิวเตอร์ในแง่ของคุณภาพและความสะดวกในการใช้งาน

ที่ผ่านมามีงานวิจัยจำนวนมากที่เกี่ยวกับการเขียนโปรแกรมบนเว็บ อย่างไรก็ตามงานวิจัยเหล่านั้นมีการพัฒนาแนวคิดและเครื่องมือเพื่อสนับสนุนการเขียนโปรแกรมบนเว็บแบบออนไลน์ ทำให้กระบวนการแปลภาษาของโปรแกรมจากภาษาระดับสูงไปเป็นภาษาเครื่องเพื่อแสดงผลลัพธ์ของโปรแกรมสามารถทำงานได้เฉพาะฝั่งผู้ให้บริการเท่านั้น อันก่อให้เกิดความซับซ้อนในการดูแลระบบ เพราะจำเป็นต้องมีผู้เชี่ยวชาญทางด้านคอมพิวเตอร์เพื่อดูแลระบบ นอกจากนี้ระบบยังมีความเป็นส่วนตัวและความปลอดภัยในการทำงานที่ต่ำ เนื่องจากโค้ดของผู้เขียนโปรแกรมอาจถูกคัดลอกข้อมูลระหว่างการส่งโค้ดเพื่อไปประมวลผลยังฝั่งผู้ให้บริการ และผลลัพธ์ของการประมวลผลที่ถูกส่งจากฝั่งผู้ให้บริการกลับมายังฝั่งผู้รับบริการอาจไม่ใช่โค้ดเดิมของผู้เขียนโปรแกรม

ในงานวิจัยนี้จึงสนใจศึกษาวิธีในการเพิ่มประสิทธิภาพของความเป็นส่วนตัวและความปลอดภัยในการเขียนโปรแกรมบนเว็บ วิธีนั้นคือการพัฒนาตัวแปลภาษาบนเว็บที่สามารถทำงานบนเครื่องผู้รับบริการโดยไม่จำเป็นต้องใช้บริการเครื่องผู้ให้บริการ และได้ผลลัพธ์ของการแปลภาษาเป็นภาษาแอสเซมบลี ตัวแปลภาษาดังกล่าวทำให้ผู้เขียนโปรแกรมสามารถทำการประมวลผลโปรแกรมได้ทันทีที่เครื่องของตน โค้ดของผู้เขียนโปรแกรมจึงไม่เสี่ยงกับการถูกคัดลอก และสามารถมั่นใจได้ว่าผลลัพธ์ของโปรแกรมเป็นผลลัพธ์ของการประมวลผลโค้ดเดิม นอกจากนี้เวลาที่ใช้ในการประมวลผลโปรแกรมน้อยลงเนื่องจากไม่เสียเวลาในการส่งโค้ดไปประมวลผลยังเครื่องผู้ให้บริการ และไม่มี

ค่าใช้จ่ายในการดูแลระบบของเครื่องผู้ให้บริการ โดยการนำเสนอวิธีดังกล่าวจะนำมาประยุกต์ใช้จริงกับการพัฒนาตัวแปลภาษาของภาษาอาร์แซดซึ่งเป็นภาษาขนาดเล็กและไม่ซับซ้อนมาก ดังนั้นจึงเหมาะกับการนำมาเป็นต้นแบบในการพัฒนาตัวแปลภาษาบนเว็บให้สามารถทำงานบนเครื่องผู้รับบริการได้ โดยตัวแปลภาษาดังกล่าวเป็นส่วนสำคัญในระบบ RZ-WEB ที่พัฒนาขึ้นสำหรับงานวิจัยนี้ พร้อมทั้งมีการวิเคราะห์ประสิทธิภาพของตัวแปลภาษาในแง่ของความถูกต้องและเวลาที่ใช้ในการทำงานจนได้ผลลัพธ์เป็นภาษาแอสเซมบลีเปรียบเทียบกับตัวแปลภาษาดั้งเดิมของภาษาอาร์แซดที่ทำงานบนระบบปฏิบัติการดอส

การพัฒนาตัวแปลภาษาบนเว็บที่สามารถทำงานบนเครื่องผู้รับบริการถูกพัฒนาโดยภาษาจาวาสคริปต์ซึ่งเป็นภาษามาตรฐานของทุกๆเว็บเบราว์เซอร์ในปัจจุบัน เนื่องจากทุกเว็บเบราว์เซอร์มีจาวาสคริปต์เอนจินซึ่งเป็นตัวแปลภาษาที่ทำหน้าที่แปลภาษาจาวาสคริปต์และประมวลผลโดยทันที ทำให้ในปัจจุบันภาษาจาวาสคริปต์สามารถรันได้กับทุกแพลตฟอร์มที่รองรับเว็บเบราว์เซอร์ ตั้งแต่เว็บเบราว์เซอร์บนคอมพิวเตอร์ตั้งโต๊ะตลอดจนเว็บเบราว์เซอร์บนแท็บเล็ตและสมาร์ทโฟน ดังนั้นตัวแปลภาษาบนเว็บในงานวิจัยนี้จึงสามารถทำงานได้เป็นอย่างดีกับอุปกรณ์ดังกล่าว โดยมีความเป็นส่วนตัวและความปลอดภัยในการเขียนโปรแกรมบนเว็บที่สูง

1.2 วัตถุประสงค์ของการวิจัย

1. นำเสนอวิธีการพัฒนาโปรแกรมสำหรับการแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ
2. พัฒนาเครื่องมือสำหรับเขียนโปรแกรมบนเว็บเพื่อสนับการเรียนรู้รายบุคคลในการเขียนโปรแกรม

1.3 ขอบเขตของการวิจัย

1. ระบบ RZ-WEB รองรับเฉพาะภาษาอาร์แซดเวอร์ชั่น 3.0
2. ผลลัพธ์ของระบบ RZ-WEB เป็นภาษาแอสเซมบลี
3. ขนาดสูงสุดของโค้ดนำเข้าคือ 40,000 ไบท์
4. การใช้หน่วยความจำสามารถใช้ได้ภายในขอบเขตที่เว็บเบราว์เซอร์มี

1.4 ขั้นตอนการวิจัย

1. ศึกษางานวิจัยที่เกี่ยวข้องการการเขียนโปรแกรมบนเว็บ
2. ศึกษาโครงสร้างของตัวแปลภาษา
3. ศึกษาทฤษฎีการพัฒนาตัวแปลภาษา
4. ศึกษาโครงสร้างของภาษาอาร์แซด
5. ออกแบบตัววิเคราะห์ศัพท์
6. ออกแบบตารางสัญลักษณ์
7. ออกแบบรายงานความผิดพลาด
8. ออกแบบตัววิเคราะห์เชิงวากยสัมพันธ์
9. ออกแบบตัววิเคราะห์ความหมาย
10. ออกแบบตัวก่อกำเนิดรหัส
11. พัฒนาโปรแกรม
12. ทดสอบและประเมินความถูกต้อง
13. แก้ไขข้อผิดพลาด
14. สรุปผลการวิจัยและเรียบเรียงวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการสามารถสนับสนุนการเรียนรู้รายบุคคลในการเขียนโปรแกรมภาษาอาร์แซดโดยมีความเป็นส่วนตัวและความปลอดภัยในการเขียนโปรแกรมบนเว็บสูง
2. การพัฒนาตัวแปลภาษาอาร์แซดด้วยภาษาจาวาสคริปต์จะเป็นต้นแบบในการประยุกต์ใช้การพัฒนาตัวแปลภาษาอื่นๆให้สามารถทำงานบนเว็บได้

1.6 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตอบรับให้ตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง “Program Development Tools: Debugging by Reverse Computing” โดย กมลลักษณ์ สุขเสน และ ประภาส จงสถิตย์วัฒนา ในงานประชุมวิชาการ “International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC) 2014” ณ โรงแรมภูเก็ต เกรซแลนด์ รีสอร์ท แอนด์ สปา จังหวัดภูเก็ต ในระหว่างวันที่ 1-4 กรกฎาคม 2557

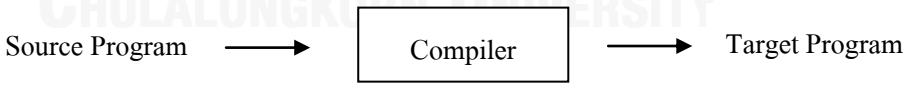
บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้เป็นการนำเสนอความรู้เบื้องต้นและทฤษฎีที่นำมาประยุกต์ใช้ในการพัฒนาตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ เริ่มต้นด้วยการอธิบายถึงความหมาย โครงสร้างของตัวแปลภาษา ประกอบด้วยตัววิเคราะห์ศัพท์ ตัววิเคราะห์เชิงวากยสัมพันธ์ ตัววิเคราะห์ความหมาย ตัวทำรหัสให้ได้ประสิทธิภาพระดับสูง และตัวก่อกำเนิดรหัส โดยมีส่วนที่เสริมการทำงานของแต่ละขั้นตอนในโครงสร้างหลักคือตารางสัญลักษณ์และรายงานความผิดพลาด จากนั้นอธิบายกระบวนการแปลภาษาของตัวแปลภาษา โดยกล่าวถึงกระบวนการทำงานของแต่ละขั้นตอนในโครงสร้างหลัก และนำเสนอทฤษฎีเกี่ยวกับโครงสร้างของภาษาอาร์แซดซึ่งเป็นภาษาต้นแบบที่นำมาประยุกต์ใช้ในงานวิจัยนี้ โดยแบ่งเป็นส่วนไวยากรณ์ของภาษาอาร์แซดและตัวอย่างโปรแกรม เพื่อเป็นพื้นฐานในการทำความเข้าใจเนื้อหาที่นำเสนอในบทถัดๆไป พร้อมทั้งนำเสนองานวิจัยที่เกี่ยวข้องกับการเขียนโปรแกรมบนเว็บ

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 ความหมายของตัวแปลภาษา

ตัวแปลภาษา (compiler) คือโปรแกรมคอมพิวเตอร์ที่ทำหน้าที่แปลภาษาจากภาษาระดับสูง เช่นภาษาซี ภาษาจาวา หรือภาษาปาสคาล เป็นต้น ไปเป็นภาษาเครื่องเพื่อให้เครื่องคอมพิวเตอร์สามารถเข้าใจคำสั่งได้ ภาษาที่ถูกแปลเรียกว่า ภาษาต้นฉบับ (source language) ส่วนภาษาที่ได้จากการแปล เรียกว่า ภาษาเป้าหมาย (target language) และเรียกโปรแกรมที่เขียนด้วยภาษาต้นฉบับว่าโปรแกรมต้นฉบับ (source program) ส่วนโปรแกรมที่เขียนด้วยภาษาเป้าหมายเรียกว่าโปรแกรมเป้าหมาย (target program) โดยสามารถดูแผนผังเบื้องต้นของกระบวนการทำงานของตัวแปลภาษาดังภาพที่ 2-1

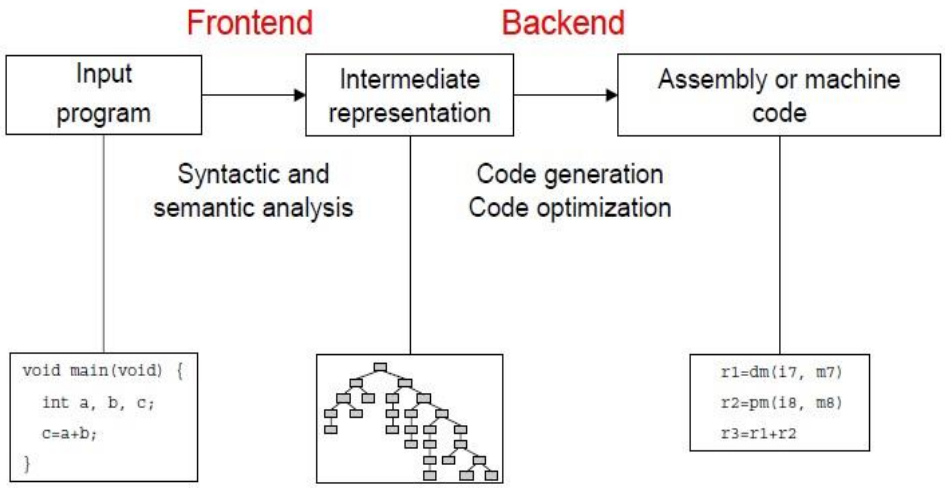


ภาพที่ 2-1 แผนผังเบื้องต้นของตัวแปลภาษา

ตัวแปลภาษาเป็นโปรแกรมค่อนข้างใหญ่และสลับซับซ้อน จึงจำเป็นต้องแบ่งออกเป็นขั้นตอนย่อยๆ โดยที่แต่ละขั้นตอนทำหน้าที่อย่างชัดเจน ซึ่งตัวแปลภาษาอาจจะมีหลายขั้นตอนแตกต่างกัน แต่ขั้นตอนที่สำคัญซึ่งตัวแปลภาษาส่วนใหญ่มี ดังแสดงในภาพที่ 2-3 คือ ตัววิเคราะห์ศัพท์ (lexical analyzer) ตัววิเคราะห์เชิงวากยสัมพันธ์ (syntax analyzer) ตัววิเคราะห์ความหมาย (semantic analyzer) ตัวทำรหัสให้ได้ประสิทธิภาพระดับสูง (high-level optimizations) และตัวก่อกำเนิดรหัส (code generator) [4] โดย

ส่วนเสริมการทำงานของขั้นตอนหลัก คือ ตารางสัญลักษณ์ (symbol table) จะทำหน้าที่เก็บข้อมูลต่างๆ เช่น ชื่อของตัวระบุ (identifier) และข้อมูลที่เกี่ยวข้องกับตัวระบุ เป็นต้น และส่วนรายงานความผิดพลาด (error handler) จะทำหน้าที่รายงานความผิดพลาดที่เกิดขึ้นให้ทราบ

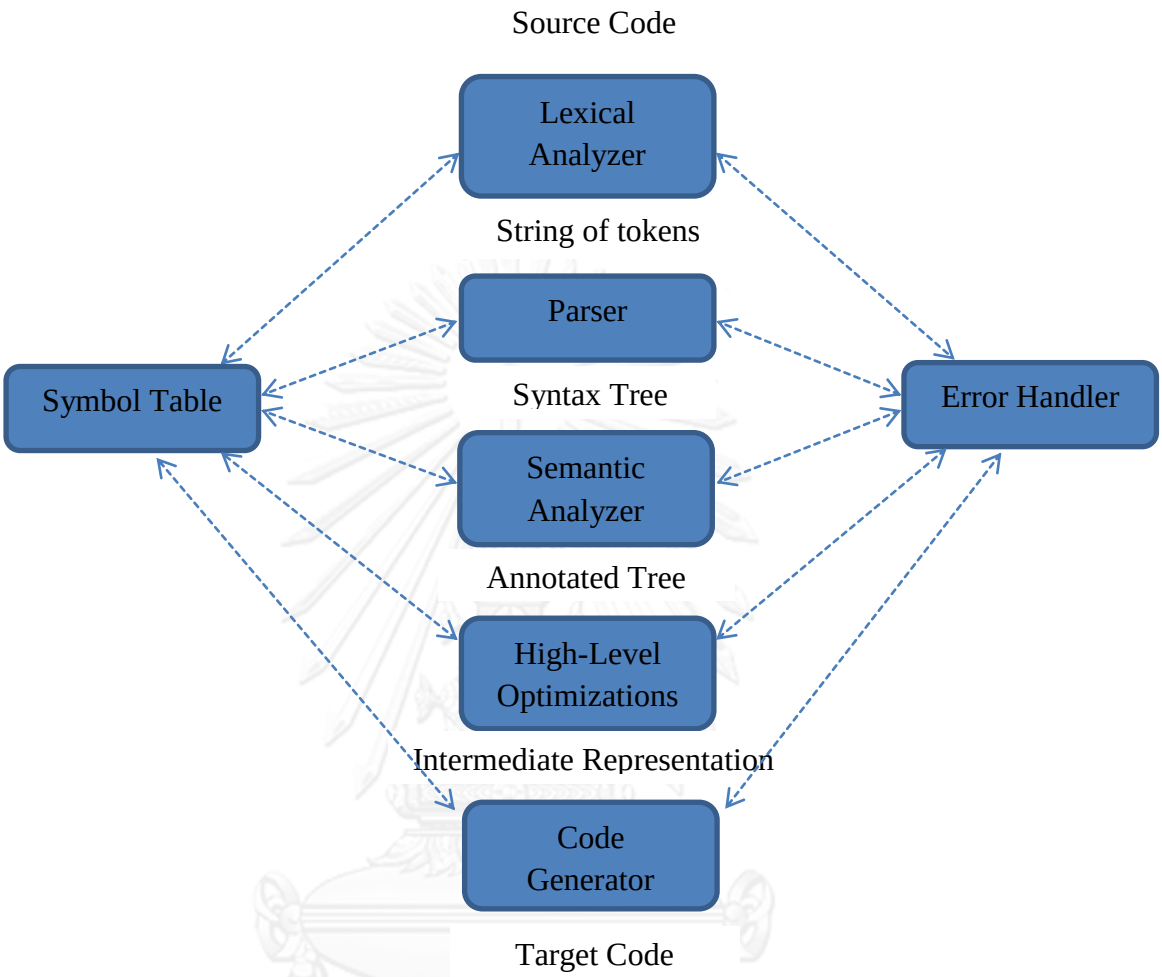
2.1.2 โครงสร้างของตัวแปลภาษา



ภาพที่ 2-2 โครงสร้างของตัวแปลภาษา

โครงสร้างของตัวแปลภาษาแบ่งเป็นสองส่วนสำคัญคือส่วนข้างหน้า (frontend) และส่วน ข้างหลัง (backend) [5] ส่วนข้างหน้าเป็นส่วนที่เกี่ยวข้องกับโครงสร้างภาษา ในส่วนนี้จะมีกระบวนการปลีกย่อยอีก เช่น ตัววิเคราะห์ศัพท์ (lexical analyzer) ตัววิเคราะห์เชิงวากยสัมพันธ์ (parser) ตัววิเคราะห์ความหมาย (semantic analyzer) หน้าที่หลักๆของส่วนนี้คือทำหน้าที่วิเคราะห์และตรวจสอบว่าโค้ดที่เขียนมานั้นมีความถูกต้องตามข้อบังคับต่างๆของภาษาหรือไม่ ซึ่งถ้าหากโค้ดที่เขียนมีความถูกต้องเรียบร้อยแล้ว ตัวแปลภาษาจะสร้างรหัสระหว่างกลาง (intermediate code หรือ intermediate representation) ซึ่งเป็นชุดคำสั่งระดับล่าง ที่เกือบจะเท่ารหัสเครื่อง (machine code) จากนั้นจึงส่งต่อไปยังส่วนสุดท้ายคือส่วนข้างหลัง ในส่วนนี้จะมีกระบวนการปลีกย่อยคือ ตัวก่อกำเนิดรหัส (code generator) ซึ่งจะทำหน้าที่แปลงรหัสระหว่างกลางให้กลายเป็นภาษาเครื่องหรือภาษาเป้าหมายที่พร้อมจะทำงาน

2.1.3 กระบวนการแปลภาษาของตัวแปลภาษา



ภาพที่ 2-3 กระบวนการแปลภาษาของตัวแปลภาษา

1. ตัววิเคราะห์ศัพท์ (lexical analyzer)

ขั้นตอนนี้ทำหน้าที่อ่านโปรแกรมต้นฉบับและทำการแยกโปรแกรมต้นฉบับออกเป็นหน่วยย่อยๆที่เรียกว่า โทเค็น (token) โดยแต่ละภาษาจะมีโทเค็นไม่เหมือนกัน แต่ส่วนใหญ่จะคล้ายกัน เช่น โทเค็นตัวระบุ โทเค็นตัวดำเนินการทางคณิตศาสตร์ (+, -, *, /) เป็นต้น ตัวอย่างโปรแกรมภาษาอาร์แซดในภาพที่ 2-4

```
main ( ) {
      a[index] = 3 + 7 ;
}
```

ภาพที่ 2-4 ตัวอย่างโปรแกรมภาษาอาร์แซด

จากตัวอย่างโปรแกรมในภาพข้างบน เมื่อไม่นับช่องว่างและการขึ้นบรรทัดใหม่จะประกอบด้วยอักขระทั้งหมด 21 ตัว แต่เมื่อแยกออกเป็นโทเค็นจะมีทั้งหมด 11 โทเค็น ดังนี้

main	identifier
(	left parentheses
)	right parentheses
{	left brace
a	identifier
[	left bracket
index	identifier
]	right bracket
=	assignment
3	number
+	plus sign
7	number
;	semicolon
}	right brace

โดยแต่ละโทเค็นอาจประกอบด้วยอักขระมากกว่า 1 ตัวเช่น โทเค็นตัวระบุเป็นต้น สำหรับอักขระที่ไม่อยู่ในเซตของอักขระของภาษาต้นฉบับ จะถูกตรวจจับได้และรายงานความผิดพลาดให้ผู้เขียนโปรแกรมทราบ

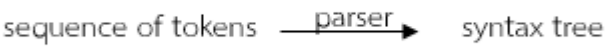
2. ตัววิเคราะห์เชิงวากยสัมพันธ์ (parser)

ขั้นตอนนี้ทำหน้าที่ตรวจสอบความถูกต้องของภาษาในส่วนของวากยสัมพันธ์ โดยใช้ไวยากรณ์ (grammar) ในการตรวจสอบวากยสัมพันธ์ของภาษา ซึ่งตรวจสอบเฉพาะการจัดเรียงลำดับของโทเค็นว่าถูกต้องหรือไม่ เช่น โปรแกรมภาษาอาร์แซดในภาพที่ 2-5 เขียนผิดวากยสัมพันธ์ เพราะหลังเลข 7 ไม่มี ; เป็นต้น

```
main ( ) {
      a[index] = 3 + 7
}
```

ภาพที่ 2-5 ตัวอย่างโปรแกรมภาษาอาร์แซดที่ผิด

ผลลัพธ์ที่ได้จากขั้นตอนนี้คือต้นไม้วิเคราะห์กระจาย (parse tree หรือ syntax tree) สามารถแสดงแผนผังเบื้องต้นของการทำงานในขั้นตอนนี้ดังแสดงในภาพที่ 2-6



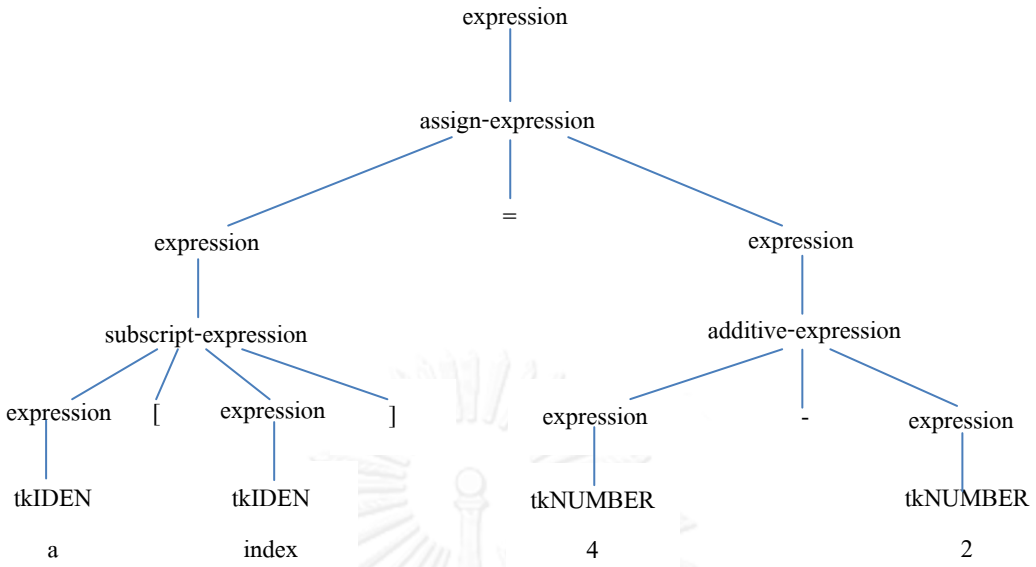
ภาพที่ 2-6 แผนผังเบื้องต้นการทำงานของตัววิเคราะห์เชิงวากยสัมพันธ์

ตัวอย่างโปรแกรมในภาพที่ 2-7

```
a[index] = 4 - 2;
```

ภาพที่ 2-7 ตัวอย่างโปรแกรมภาษาอาร์แซด

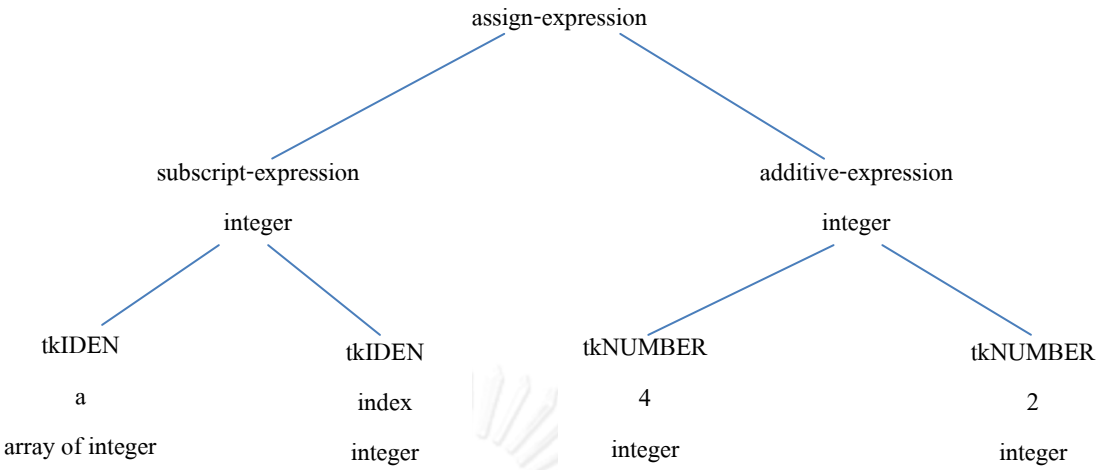
เมื่อผ่านขั้นตอนตัววิเคราะห์เชิงวากยสัมพันธ์จะได้ผลลัพธ์เป็นต้นไม้วิเคราะห์กระจายดังในภาพที่ 2-8



ภาพที่ 2-8 ต้นไม้วิเคราะห์กระจายของโปรแกรมในภาพที่ 2-7

3. ตัววิเคราะห์ความหมาย (semantic analyzer)

ขั้นตอนนี้ทำหน้าที่ตรวจสอบความหมายของภาษา โดยมีการบันทึกข้อมูลต่างๆไว้ในตารางสัญลักษณ์ เช่น ข้อมูลเกี่ยวกับประเภทของตัวระบุ ได้แก่ ชื่อค่าคงที่ ชื่อตัวแปร ชื่อฟังก์ชัน และชนิดของข้อมูล (data type) เช่น ตัวแปรประเภท สเกล่าร์ ตัวแปรประเภทอาร์เรย์ จำนวนเต็ม เป็นต้น เพื่อให้โปรแกรมสามารถตรวจสอบได้ว่าการใช้ตัวระบุต่างๆใช้ได้ถูกต้องตามข้อกำหนดของภาษาหรือไม่ ซึ่งตัววิเคราะห์ความหมายจะทำการบันทึกชนิดของข้อมูลแต่ละข้อมูลกำกับไว้ที่ ซินเท็กทรี (syntax tree) และตรวจสอบการกำหนดข้อมูลว่าถูกต้องตามชนิดของข้อมูลหรือไม่ ถ้าการระบุข้อมูลไม่ตรงตามชนิดของข้อมูลจะรายงานความผิดพลาดที่เกิดขึ้น จากโปรแกรมภาษาอาร์แซดในภาพที่ 2-7 ซินเท็กทรีที่ผ่านตัววิเคราะห์ความหมายแล้วจะมีชนิดของข้อมูลกำกับอยู่ดังภาพที่ 2-9 เรียกว่าแอนโนเทดทรี (annotated tree)



ภาพที่ 2-9 ต้นไม้วิเคราะห์ที่มีการกำกับชนิดของข้อมูล (Annotated tree)

4. ตัวทำรหัสให้ได้ประสิทธิภาพระดับสูง (high-level optimizations)

การทำรหัสให้ได้ประสิทธิภาพสูงมีหลายวิธี โดยจะกล่าวถึงบางวิธีดังนี้

1. การขจัดนิพจน์ย่อยซ้ำซ้อน (common subexpression) เช่น ส่วนของโปรแกรมต่อไปนี้

```
A = Y - 2;
B = 3 * (Y - 2) - (Y - 2);
```

สามารถทำให้ได้ผลดีที่สุด ดังนี้

```
A = Y - 2;
B = 2 * A - A;
```

2. การทำรอบซ้ำๆให้ได้ผลดีที่สุด (loop optimization) เช่น โค้ดเทียม (pseudo code) ของโปรแกรมต่อไปนี้

```
for i=1 to n Begin
        for j=1 to m Begin
                X[i][j] = X[i][j] + Y[i][j]
                Z = i + j
        End
End
```

สามารถทำให้ได้ผลดีที่สุด ดังนี้

```
for i=1 to n Begin
        for j=1 to m Begin
                X[i][j] = X[i][j] + Y[i][j]
        End
End
Z = n + m
```

เพราะ Z ไม่ได้ถูกใช้ในลูปซ้ำๆ จึงสามารถนำออกมานอกลูปได้ โดยที่ค่าของZ หลังออกมาจากนอกลูปมีค่าเท่ากับ n + m ทำให้สามารถลดการคำนวณ Z = i + j ได้ถึง n * m ครั้ง และใช้เวลาคำนวณ Z = n + m เพียงครั้งเดียว

3. การใช้คำสั่งที่เร็วกว่า เช่น แทน Y * 2 ด้วย Y + Y เพราะการบวกจะทำได้เร็วกว่าการคูณ

5. ตัวก่อกำเนิดรหัส (code generator)

ตัวก่อกำเนิดรหัสทำหน้าที่นำรหัสระหว่างกลาง (intermediate code หรือ intermediate representation) มาผลิตภาษาเป้าหมายตามที่ต้องการ โดยตัวแปลภาษาบางตัวอาจมีขั้นตอนตัวปรับปรุงรหัสให้ได้ผลดีที่สุด เป็นขั้นตอนถัดจากตัววิเคราะห์ความหมาย และมาขั้นตอนตัวก่อกำเนิดรหัสเป็นขั้นตอนสุดท้าย โดยหลังจากขั้นตอนตัววิเคราะห์ความหมาย และขั้นตอนตัวปรับปรุงรหัสให้ได้ผลดีที่สุดจะได้ผลลัพธ์เป็นแอนโนเทดทรี ซึ่งเป็นรหัสระหว่างกลางนั่นเอง ข้อดีของการมีรหัสระหว่างกลางคือเมื่อจำเป็นต้องผลิตภาษาเป้าหมายใหม่ก็เพียงแต่เขียนขั้นตอนที่เปลี่ยนจากรหัสระหว่างกลางไปเป็นภาษาเป้าหมายใหม่ โดยไม่จำเป็น ต้องเขียนโค้ดขึ้นมาใหม่ แต่ข้อเสียคือ ภาษาเป้าหมายที่ผลิตจากรหัสระหว่างกลาง อาจจะมีประสิทธิภาพน้อยกว่าการผลิตภาษาเป้าหมายโดยตรงโดยไม่ใช้รหัสระหว่างกลาง

2.1.4 โครงสร้างของภาษาอาร์แซด

1. เป็นภาษาที่คล้ายกับส่วนเล็กๆของภาษาซี
2. มีชนิดข้อมูลชนิดเดียว คือ จำนวนเต็ม (integer)
3. ตัวแปรเฉพาะที่ (local variable) ไม่จำเป็นต้องประกาศตัวแปร
4. ตัวแปรส่วนกลาง (global variable) จำเป็นต้องประกาศตัวแปรก่อนใช้
5. ตัวแปรส่วนกลางสามารถเป็นได้ทั้งจำนวนสเกล่าร์และจำนวนอาร์เรย์
6. ตัวแปรเฉพาะที่เป็นได้เฉพาะจำนวนสเกล่าร์เท่านั้น
7. จำนวนอาร์เรย์ต้องมีการระบุขนาดของอารเรย์เสมอ
8. จำนวนอาร์เรย์เป็นได้เฉพาะอาร์เรย์ 1 มิติ
9. คำสงวน (reserved words) ได้แก่ if, else, while, return, print.
10. ตัวดำเนินการ (operators) ได้แก่ +, -, *, /, ==, !=, <, <=, >, >=, !, &&, ||, * (dereference), และ & (address)

ตัวอย่างโปรแกรมภาษาอาร์แซด

```
array[12], M;
start(){
 in = 0;
 while ( in < M ){
  array[in] = in;
  in = in + 1;
 }
}

main(){
 M = 12;
 start();
 min = array[0];
 in = 1;
 while( in < M ){
  if(min > array[in]) min = array[in];
  in = in + 1;
 }
 print("the min value is ",min);
}
```

ภาพที่ 2-10 ตัวอย่างโปรแกรมหาจำนวนที่น้อยที่สุด

จากโปรแกรมภาษาอาร์แซดดังภาพที่ 2-10 เป็นโปรแกรมหาจำนวนที่น้อยที่สุดและแสดงค่าออกมา โดยฟังก์ชัน start คือฟังก์ชันที่ทำการใส่ค่าเข้าไปในอาร์เรย์ทั้งหมด 12 ช่อง โดยอาร์เรย์ช่องแรกคือช่องที่ศูนย์ใส่ค่า 0 และอาร์เรย์ช่องถัดไปใส่ค่า 1, 2, 3, ไปเรื่อยๆ ตามลำดับจนถึงอาร์เรย์ช่องสุดท้ายคือช่อง 11 จะใส่ค่า 11 และกำหนดค่าน้อยที่สุด (min) เป็นค่าอาร์เรย์ช่องที่ 0 หลังจากนั้นจะมีการเปรียบเทียบค่าอาร์เรย์ในแต่และช่องกับค่าน้อยที่สุด โดยใช้วิธีการวนลูปเพื่อเปรียบเทียบค่าไปเรื่อยๆ ถ้าค่าในอาร์เรย์น้อยกว่าค่าที่น้อยที่สุดก็จะกำหนดให้ค่าที่น้อยที่สุดเป็นค่าเดียวกับค่าอาร์เรย์ ณ ช่องนั้นๆ และเมื่อวนลูปจนครบแล้ว จะทำการแสดงค่าที่น้อยที่สุดออกมา

2.2 งานวิจัยที่เกี่ยวข้อง

บทนี้จะกล่าวถึงงานวิจัยที่เกี่ยวข้องกับการเขียนโปรแกรมบนเว็บ จุดประสงค์ของการเขียนโปรแกรมบนเว็บในปัจจุบันมีจุดประสงค์ 3 ประการได้แก่

1. สำหรับภายในมหาวิทยาลัยหรือสถาบันการศึกษา เพื่อให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้อย่างสะดวก และอาจารย์ผู้สอนสามารถติดตามการเขียนโปรแกรมของผู้เขียนโปรแกรมแต่ละคนได้
2. เพื่อให้ผู้เขียนโปรแกรมสามารถฝึกทักษะการเขียนโปรแกรมได้ด้วยตนเองโดยไม่จำเป็นต้องติดตั้งโปรแกรมใดๆ
3. เพื่อให้การสอบเขียนโปรแกรมภายในมหาวิทยาลัยหรือสถาบันการศึกษาเป็นไปอย่างสะดวกและเหมาะสม

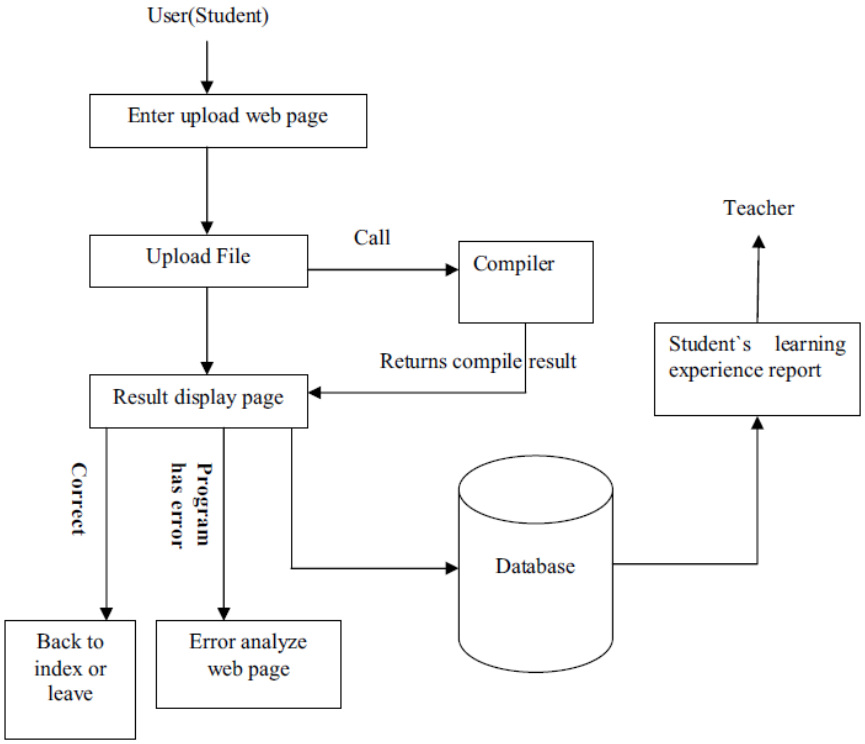
2.2.1 สภาพแวดล้อมการเรียนการสอนแบบออนไลน์เพื่อสนับสนุนการเรียนรู้รายบุคคลในการเขียนโปรแกรมภาษาจาวา

ผู้วิจัยได้พัฒนาตัวแปลภาษาของภาษาจาวาแบบออนไลน์ โดยจุดประสงค์ของการพัฒนาเครื่องมือนี้ คือ เปิดโอกาสให้นักเรียนสามารถเรียนรู้การเขียนโปรแกรมได้ทุกที่ทุกเวลา นอกจากนี้ระบบการเรียนการสอนแบบออนไลน์ยังมีตัวติดตามที่คอยติดตามและจัดการเกี่ยวกับการเรียนเขียนโปรแกรมของนักเรียน [3]

เนื่องจากโปรแกรมภาษาจาวานั้นเป็นการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming) และการรันโปรแกรมภาษาจาวาจำเป็นต้องรันบนจาวาเวอชวลเมชชีน (Java Virtual machine หรือ JVM) ซึ่งก็คือโปรแกรมที่ใช้อ่านคำสั่งจาวาไบท์โค้ด (Java bite code) และจะดำเนินการตามที่อ่านได้ ถึงแม้ว่าจาวาไบท์โค้ดจะไม่ใช่ภาษาของเครื่องที่ซีพียูสามารถอ่านได้ แต่มันถือได้ว่าเป็นภาษาเครื่องสำหรับจาวาเวอชวลเมชชีน นั่นคือจาวาเวอชวลเมชชีนคือโปรแกรมที่จำลองเครื่องคอมพิวเตอร์ที่มีภาษาเครื่องเป็นจาวาไบท์โค้ดนั่นเอง ด้วยเหตุผลดังกล่าวจึงทำให้การรันโปรแกรมภาษาจาวาสามารถทำได้บนหลายแพลตฟอร์ม และในมหาวิทยาลัยของผู้วิจัยมีการเปิดสอนรายวิชาการเขียนโปรแกรมเชิงวัตถุโดยเลือกภาษาจาวามาเป็นภาษาที่ใช้ในการเรียนการสอนในรายวิชานี้

เพื่อให้นักเรียนสามารถเรียนการเขียนโปรแกรมภาษาจาวาได้ทุกที่ทุกเวลา ระบบที่ผู้วิจัยพัฒนาจึงมีการเรียกใช้งานตัวแปลภาษาจาวาแบบออนไลน์ เพื่อให้สามารถทำการประมวลผลโค้ดจาวาของนักเรียนได้ ผลลัพธ์ที่ได้จากการประมวลผลจะถูกเขียนลงในฐานข้อมูลการเรียนรู้ หลังจากนั้นข้อผิดพลาดต่างๆของการเขียนโค้ดจาวาจะถูกส่งกลับไปยังนักเรียนเพื่อรายงานความผิดพลาดให้ทราบ ดังนั้นอาจารย์สามารถที่จะวิเคราะห์ข้อมูลต่างๆได้จากฐานข้อมูลการเรียนรู้เพื่อพัฒนาการเรียนการสอนของพวกเขาต่อไป นอกจากนี้ระบบยังมีตัวติดตาม (active agent) เพื่อติดตามและจัดการเกี่ยวกับการเรียนรู้ของนักเรียน

ระบบดังกล่าวใช้จาวาเซิร์ฟเล็ต (Java Servlet) จาวาเซิร์ฟเวอร์เพจ(JavaServer Pages) และจาวาดาต้าเบสคอนเน็กติวิตี้ (Java Database Connectivity หรือ JDBC) เพื่อเข้าถึงข้อมูลในฐานข้อมูลมายเอสคิวแอล (MySQL) ของฝั่งผู้ให้บริการ (server) โดยสามารถแสดงกลไกการทำงานของระบบดังกล่าว ดังภาพที่ 2-11



ภาพที่ 2-11 กลไกการทำงานของระบบตัวแปลภาษาจาวาแบบออนไลน์

2.2.2 สภาพแวดล้อมการเรียนรู้สำหรับการเขียนโปรแกรมผ่านเว็บเพื่อสนับสนุนการพัฒนาองค์ความรู้

การเรียนรู้การเขียนโปรแกรมแบบดั้งเดิม เช่น ในชั้นเรียนคอมพิวเตอร์ มันไม่ง่ายที่จะส่งเสริมการพัฒนาองค์ความรู้ของนักเรียนในการเขียนโปรแกรม ถ้าไม่มีกิจกรรมการเรียนรู้และเครื่องมือที่ช่วยในการเรียนการสอนเข้ามาเป็นองค์ประกอบเสริม งานวิจัยชิ้นนี้จึงพยายามสร้างเครื่องมือที่ช่วยพัฒนาความรู้ในการเขียนโปรแกรมโดยเครื่องมือนี้จะเป็นเครื่องมือที่ช่วยในการเขียนโปรแกรมผ่านเว็บ ซึ่งสิ่งสำคัญที่ต้องนำมาพิจารณาสำหรับการสร้างเครื่องมือนี้คือการคำนึงถึงความซับซ้อนของฝั่งผู้รับบริการและฝั่งผู้ให้บริการ (client-server) และฐานข้อมูลที่จำเป็นต้องใช้ [6]

ผู้วิจัยได้พยายามออกแบบชุดของกิจกรรมการเรียนเขียนโปรแกรมผ่านเว็บที่จะเข้ามาช่วยเสริมการพัฒนาองค์ความรู้ในการเขียนโปรแกรมของนักเรียนผ่านเว็บ และนำมาพัฒนาระบบที่ชื่อระบบดับเบิ้ลยูพีเอเอส (WPAS หรือ Web-based Programming Assisted System) ระบบนี้ถูกพัฒนาเพื่อเข้ามาสนับสนุนกิจกรรมการเรียนเขียนโปรแกรม โดยระบบนี้มีเครื่องมือหลัก 3 เครื่องมือให้บริการ ได้แก่

1. เครื่องมือช่วยในการเขียนโปรแกรมแบบออนไลน์ (Online coding tool)

ในส่วนนี้ประกอบด้วยเครื่องมือ program gap filling เครื่องมือนี้ช่วยเพิ่มความสะดวกในการเขียนโปรแกรมโดยมีการเขียนโค้ดบางส่วนในโปรแกรมให้แล้ว และมีส่วนช่องว่างที่นักเรียนจำเป็นต้องเขียนเพิ่มเติมเอง เพื่อให้โปรแกรมสมบูรณ์ และมีเครื่องมือ program debugging เป็นเครื่องมือที่ช่วยสำหรับการหาส่วนที่เป็นข้อผิดพลาดในการเขียนโปรแกรม และสุดท้ายเครื่องมือ coding to solve problem เมื่อเกิดข้อผิดพลาดในโค้ดขึ้น นักเรียนสามารถแก้ไขโค้ดแบบออนไลน์ได้ทันที

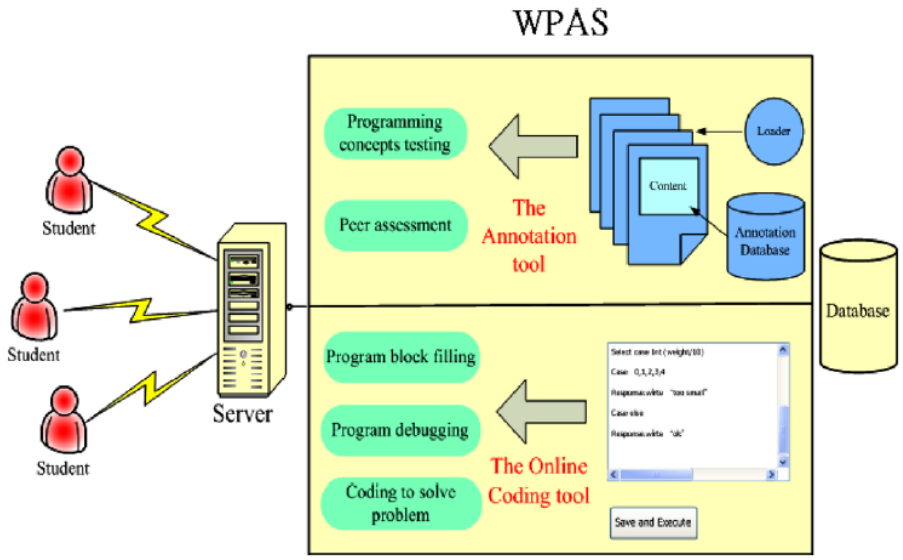
2. เครื่องมือเกี่ยวกับคำอธิบายประกอบสำหรับการประเมินโปรแกรม (Annotation tool for assessing the programs)

นักเรียนได้รับอนุญาตให้ทำคำอธิบายประกอบในแต่ละบรรทัดของโปรแกรมที่ถูกอัพโหลดโดยนักเรียนคนอื่นได้ อันจะทำให้เกิดการแลกเปลี่ยนความรู้กันระหว่างนักเรียนผ่านการเขียนความเห็นและข้อแนะนำ โดยมีเครื่องมือ Programming concept testing และเครื่องมือ peer assessment ช่วยในการทำงานดังกล่าว

3. เครื่องมือสำหรับช่วยหาคำสำคัญในโค้ด (Searching for key words in source code)

นักเรียนสามารถหาคำสำคัญต่างๆในโปรแกรมได้อย่างง่ายดายโดยใช้เครื่องมือนี้

สามารถแสดงสถาปัตยกรรมของระบบ WPAS ดังภาพที่ 2-12



ภาพที่ 2-12 สถาปัตยกรรมของระบบ WPAS

2.2.3 ตัวแปลภาษาแบบออนไลน์ของภาษาซีและซีพลัสพลัสโดยใช้ระบบคลาวด์

ผู้วิจัยได้นำการประมวลผลแบบคลาวด์ (cloud computing) [2] เข้ามาช่วยในการลดปัญหา portability กล่าวคือ โปรแกรมคอมพิวเตอร์สามารถทำงานในระบบปฏิบัติการอื่นได้ โดยไม่จำเป็นต้องมีการสร้างโปรแกรมใหม่ทั้งหมด เนื่องจากโปรแกรมบางโปรแกรมเมื่อเคลื่อนย้ายไปที่ระบบปฏิบัติการอื่นอาจจำเป็นต้องติดตั้งสภาพแวดล้อม (runtime) เพื่อให้โปรแกรมสามารถทำงานได้ หรืออาจจำเป็นต้องเขียนโปรแกรมขึ้นมาใหม่ทั้งหมด นอกจากนี้ยังช่วยลดปัญหาพื้นที่ในการจัดเก็บข้อมูลเกี่ยวกับผลลัพธ์ของการประมวลผลโปรแกรมหรือข้อผิดพลาดที่อาจเกิดขึ้น

ผู้วิจัยได้พัฒนาตัวแปลภาษาแบบออนไลน์ของภาษาซีพลัสพลัสและภาษาซีโดยใช้บริการการประมวลผลของคลาวด์ โดยฟังก์ชันหลักที่มีในสถาปัตยกรรมของระบบนี้ได้แก่

1. ฟังก์ชันคอมไพล์ (Compile option)

ฟังก์ชันนี้เริ่มทำงานโดยนำโค้ดในกล่องข้อความส่งไปยังฝั่งผู้ให้บริการเพื่อทำการประมวลผล และฝั่งผู้ให้บริการจะมีการอิมพอร์ตแพคเกจของตัวแปลภาษา (compiler)

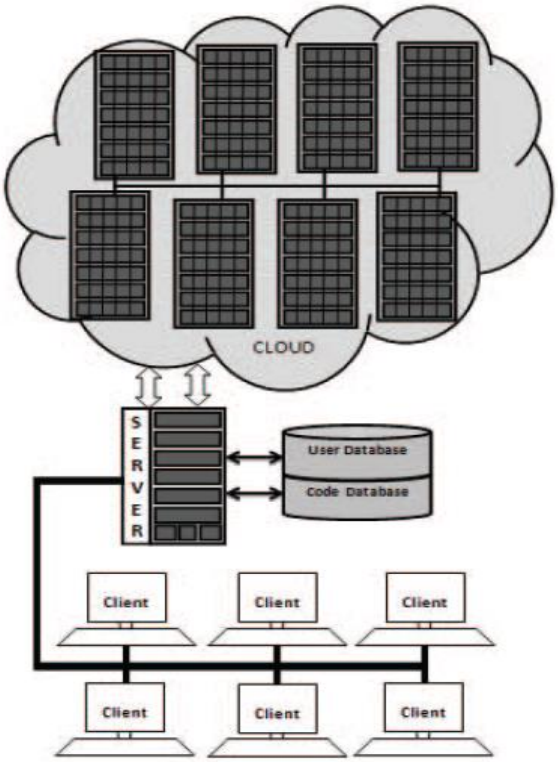
2. ฟังก์ชันเอ็กซิคิวต์ (Execute option)

มีการเชื่อมโยงของไฟล์ทั้งหมดที่ปฏิบัติการซึ่งปรากฏในโฟลเดอร์ของนักเรียน และเคยประมวลผลมาแล้วอย่างน้อยหนึ่งครั้งโดยไม่มีข้อผิดพลาดใดๆเลย

3. ฟังก์ชันเริ่มทดสอบ (Start test option)

ตราบใดที่ยังไม่มีการคลิกปุ่ม start test นักเรียนจะยังไม่สามารถเริ่มทำการเขียนโปรแกรมได้

ระบบนี้ใช้สถาปัตยกรรมแบบสองชั้น (dual-layered architecture) โดยชั้นล่างประกอบด้วย เครื่องผู้รับบริการ ส่วนชั้นบนประกอบด้วยเครื่องผู้ให้บริการ ได้แก่ A web framework และ visual studio 2010 ทำหน้าที่จัดการการทำงานของสคริปต์และการประมวลผลโค้ด นอกจากนี้ IIS server ทำหน้าที่จัดการการร้องขอใช้บริการของเครื่องผู้รับบริการ ฐานข้อมูลสำหรับเก็บข้องมูลของเครื่องผู้รับบริการ และ cloud hard disk เป็นทรัพยากรที่ใช้ร่วมกัน สามารถแสดงสถาปัตยกรรมของระบบนี้ได้ดังภาพที่ 2-13

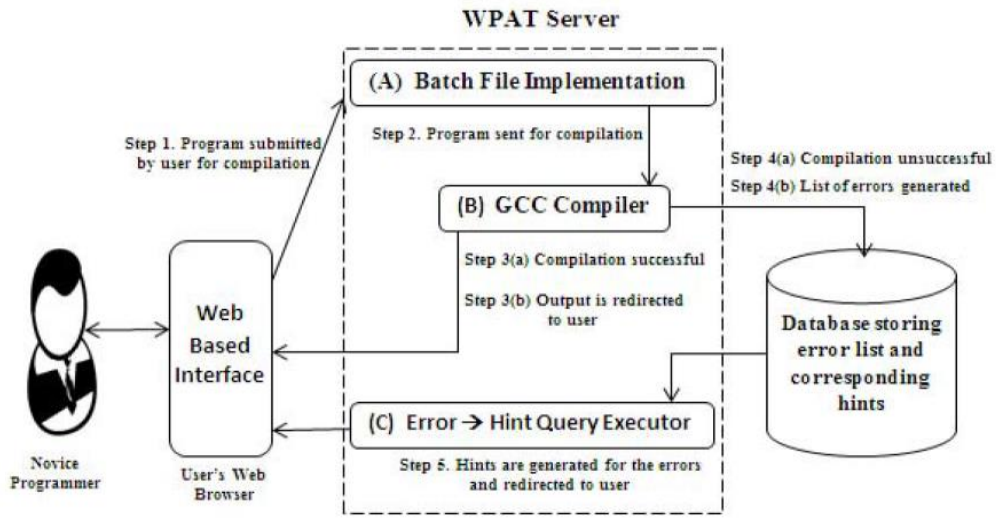


ภาพที่ 2-13 สถาปัตยกรรมของระบบตัวแปลภาษาซีและซีพลัสพลัสแบบออนไลน์

2.2.4 เครื่องมือช่วยเหลือในการเขียนโปรแกรมผ่านเว็บสำหรับผู้ริเริ่มเขียนโปรแกรม

งานวิจัยชิ้นนี้ผู้วิจัยได้กล่าวถึงความท้าทายในวงการศึกษาคอมพิวเตอร์คือ การสอนเขียนโปรแกรมให้กับนักเรียนที่เริ่มเขียนโปรแกรม [7] เนื่องจากนักเรียนที่เริ่มเขียนโปรแกรมใหม่ๆพบว่ามันยากต่อการทำความเข้าใจข้อความรายงานความผิดพลาดในการเขียนโปรแกรมของตัวแปลภาษา ผู้วิจัยจึงพัฒนาเครื่องมือต้นแบบที่เข้ามาช่วยผู้ริเริ่มเขียนโปรแกรมให้สามารถทำการแก้ไขโปรแกรมได้ด้วยตนเอง โดยเครื่องมือนี้เป็นเครื่องมือช่วยเหลือในการเขียนโปรแกรมภาษาซีผ่านเว็บ ด้วยประสิทธิภาพของเครื่องมือนี้จะช่วยให้ผู้เริ่มเขียนโปรแกรมสามารถทำความเข้าใจข้อความรายงานความผิดพลาดในการเขียนโปรแกรมจากตัวแปลภาษาและสามารถทำการแก้ไขโค้ดส่วนที่ผิดพลาดนั้นได้ด้วยตนเอง

ระบบที่ผู้วิจัยพัฒนานี้ชื่อ WPAT Server ประกอบด้วย GUI editor ตัวแปลภาษา (GCC compiler) และฐานข้อมูลส่วนกลาง (central database) โดยฐานข้อมูลจะเก็บข้อมูลรายงานความผิดพลาดของโปรแกรมจากตัวแปลภาษา และการแก้ไขที่เกี่ยวข้องและคำแนะนำ หลังจากมีข้อความผิดพลาดจากตัวแปลภาษาที่เกิดขึ้นหลังการประมวลผล ข้อความนั้นจะถูกส่งไปยังฐานข้อมูล และการแก้ไขที่เกี่ยวข้องจะถูกนำเสนอไปยังผู้เขียนโปรแกรมแทนข้อความผิดพลาดจากตัวแปลภาษาโดยตรง สถาปัตยกรรมของระบบนี้แสดงดังภาพที่ 2-14



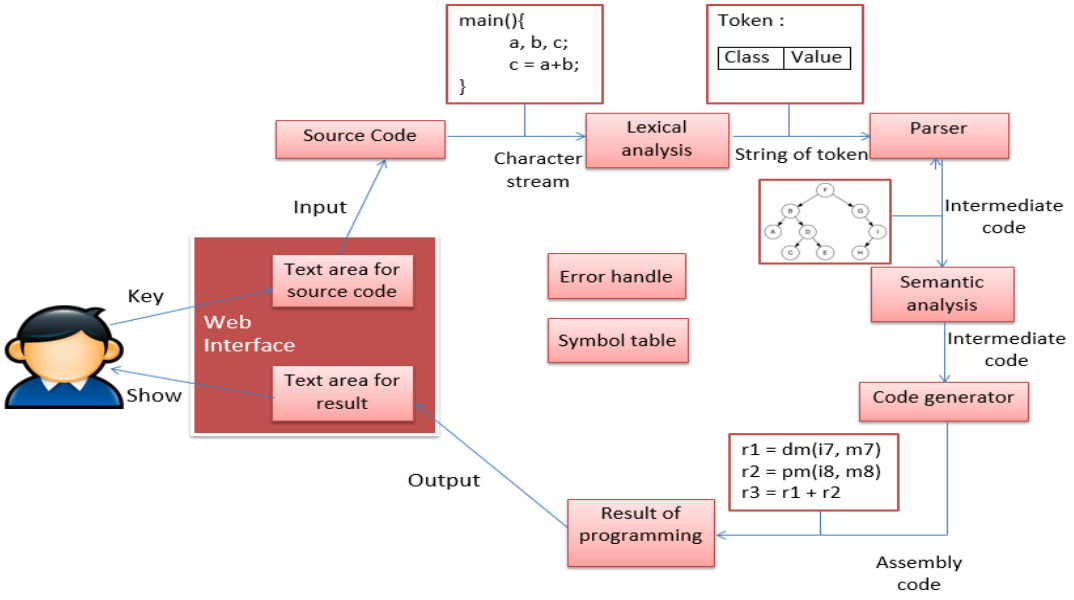
ภาพที่ 2-14 สถาปัตยกรรมของระบบ WPAT Server

บทที่ 3
การออกแบบตัวแปลภาษาอาร์แซด

ตัวแปลภาษาที่ออกแบบในงานวิจัยนี้ ได้พัฒนามาจากตัวแปลภาษาอาร์แซด [8] ซึ่งได้ออกแบบเพื่อใช้ในการเรียนการสอนเกี่ยวกับการเขียนโปรแกรม และศึกษาสถาปัตยกรรมของคอมพิวเตอร์ โดยเป็นภาษาขนาดเล็กที่สามารถนำมาใช้เพื่อแสดงให้เห็นโครงสร้างภายในทั้งหมดในส่วนการทำงานของระบบคอมพิวเตอร์ เช่น การแปลโปรแกรม (compilation) การสร้างรหัส (code generation) และโปรแกรมจำลองคอมพิวเตอร์ที่ใช้ในการสร้างสถานการณ์จำลอง (ISA simulation) นอกเหนือจากนี้นักเรียนยังสามารถทดลองเล่นกับระบบได้ โดยภาษาอาร์แซดมีลักษณะคล้ายกับภาษาซี

การเขียนโปรแกรมภาษาอาร์แซดแบบดั้งเดิมนั้นจำเป็นต้องมีโปรแกรมสำหรับเขียนโปรแกรมภาษาอาร์แซด (Text Editor) และทำการแปลภาษาและประมวลผลบนระบบปฏิบัติการดอส (Disk operating system) เพื่อเพิ่มความสะดวกในการเรียนรู้การเขียนโปรแกรมภาษาอาร์แซดมากขึ้น งานวิจัยนี้จึงออกแบบและพัฒนาระบบ RZ-WEB ซึ่งประกอบด้วยตัวแปลภาษาอาร์แซดที่สามารถทำงานได้บนเว็บเบราว์เซอร์อันเป็นโปรแกรมพื้นฐานของทุกๆระบบปฏิบัติการและไม่จำเป็นต้องอาศัยการทำงานของเครื่องผู้ให้บริการ โดยตัวแปลภาษานี้สามารถทำงานบนเครื่องผู้รับบริการได้โดยตรง ซึ่งภาษาที่ใช้ในการพัฒนาตัวแปลภาษาคือภาษาจาวาสคริปต์อันเป็นภาษามาตรฐานของเว็บเบราว์เซอร์

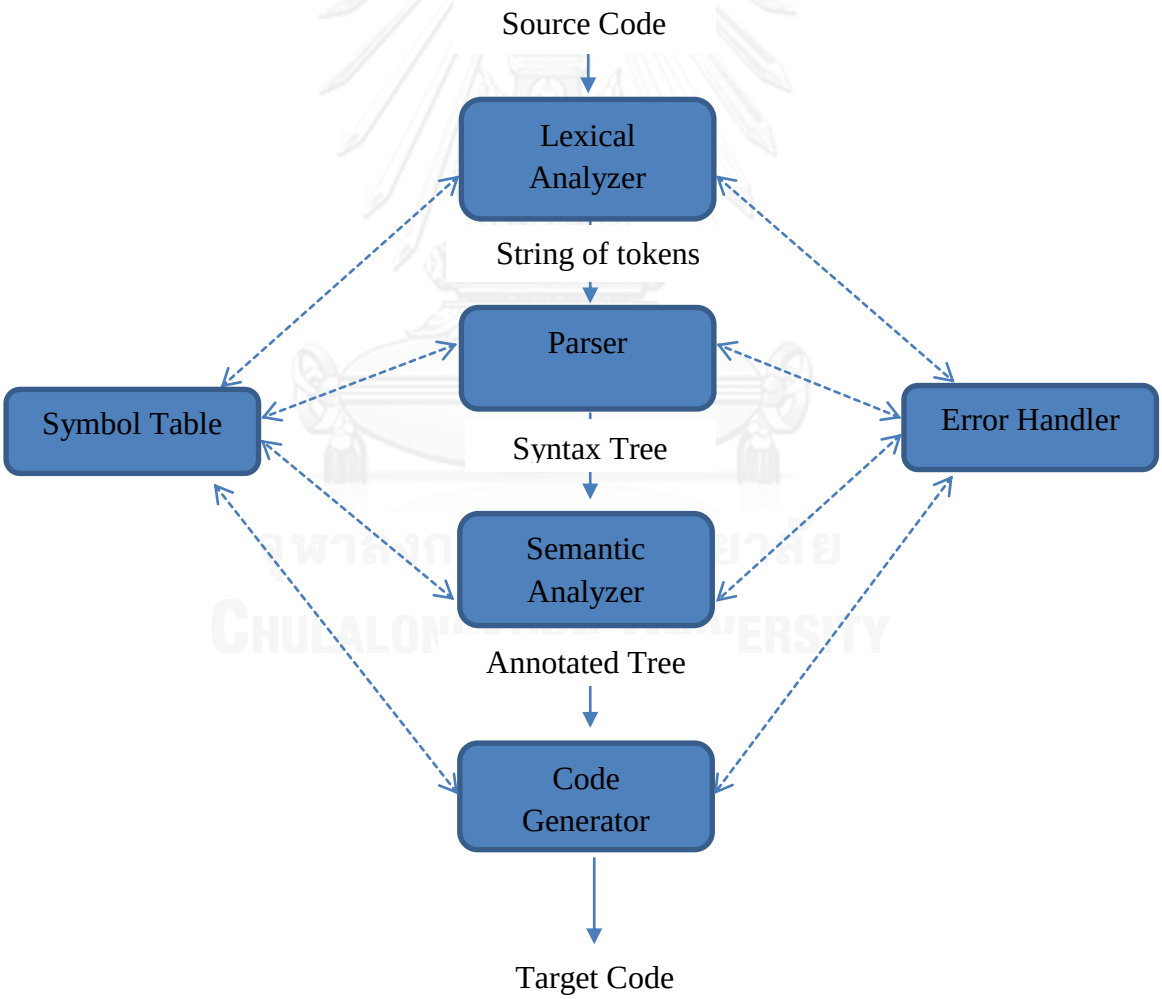
สถาปัตยกรรมของระบบ RZ-WEB ดังภาพที่ 3-1



ภาพที่ 3-1 สถาปัตยกรรมของระบบ RZ-WEB

กลไกการทำงานของระบบ RZ-WEB เริ่มจากผู้เขียนโปรแกรมทำการเขียนโปรแกรมต้นฉบับลงไปในส่วนของ Text area สำหรับการเขียนโปรแกรม และหลังจากผู้เขียนเขียนโปรแกรมกดปุ่ม Compile แล้ว โปรแกรมต้นฉบับจะถูกนำไปผ่านกระบวนการแปลภาษา โดยเริ่มจากขั้นตอนตัววิเคราะห์ศัพท์ (lexical analyzer) ทำให้ได้ผลลัพธ์ออกมาเป็นโทเค็นประเภทต่างๆ อันจะเป็นข้อมูลเข้าสำหรับขั้นตอนถัดไปคือขั้นตอนตัววิเคราะห์เชิงวากยสัมพันธ์ (parser) และทำให้ได้ผลลัพธ์เป็นรหัสระหว่างกลางซึ่งก็คือต้นไม้วิเคราะห์กระจาย(parse tree) นั่นเอง จากนั้นจึงผ่านขั้นตอนตัววิเคราะห์ความหมาย (semantic analyzer) จะได้ผลลัพธ์คือรหัสระหว่างกลางซึ่งเป็นต้นไม้วิเคราะห์กระจายที่มีลักษณะเฉพาะต่างๆกำกับไว้เพื่อให้เป็นไปตามไวยากรณ์ของภาษา และผ่านขั้นตอนสุดท้ายคือขั้นตอนตัวก่อกำเนิดรหัสอันจะทำให้ได้ผลลัพธ์เป็นโปรแกรมเป้าหมายที่เป็นภาษาแอสเซมบลี จากนั้นจึงแสดงโปรแกรมเป้าหมายใน Text area ส่วนของการแสดงผลการแปลภาษาบนเว็บ

การออกแบบตัวแปลภาษาอาร์แซดสามารถแบ่งเป็นขั้นตอนย่อยๆได้ดังภาพที่ 3-2

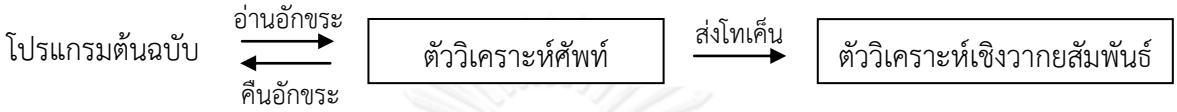


ภาพที่ 3-2 กระบวนการแปลภาษาของตัวแปลภาษาอาร์แซด

โดยการออกแบบแต่ละขั้นตอนสามารถอธิบายได้ดังนี้

3.1 การออกแบบตัววิเคราะห์ศัพท์

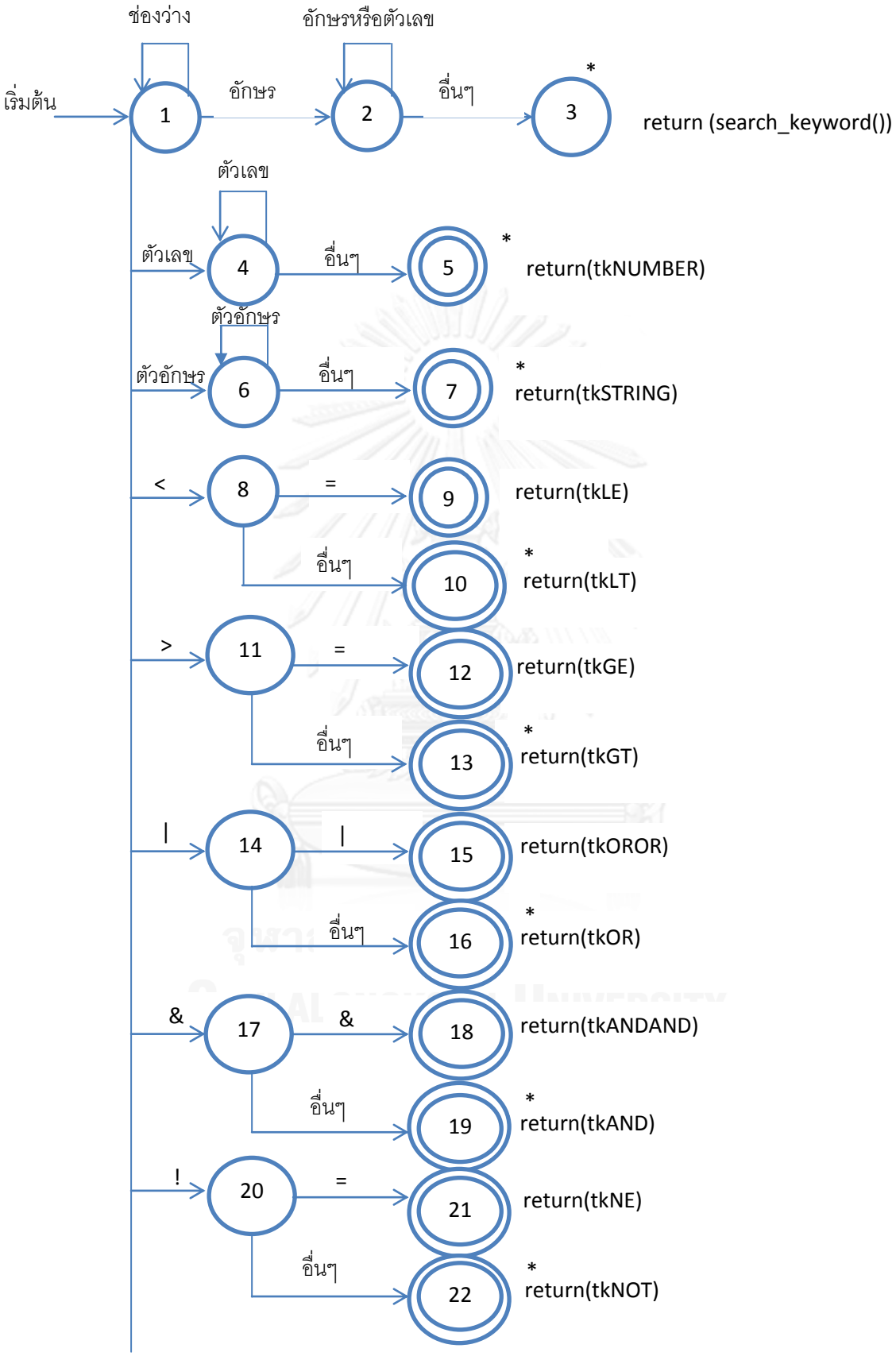
ตัววิเคราะห์ศัพท์ทำหน้าที่อ่านอักขระทีละอักขระจากโปรแกรมต้นฉบับ ขณะที่อ่านจะจดจำตำแหน่งสดมภ์และบรรทัดของอักขระนั้น เมื่อพบโทเค็นจะทำการส่งโทเค็นไปยังขั้นตอนถัดไป กรณีที่อ่านอักขระเกินมาจะคืนอักขระนั้นกลับไป ดังแสดงในภาพที่ 3-3



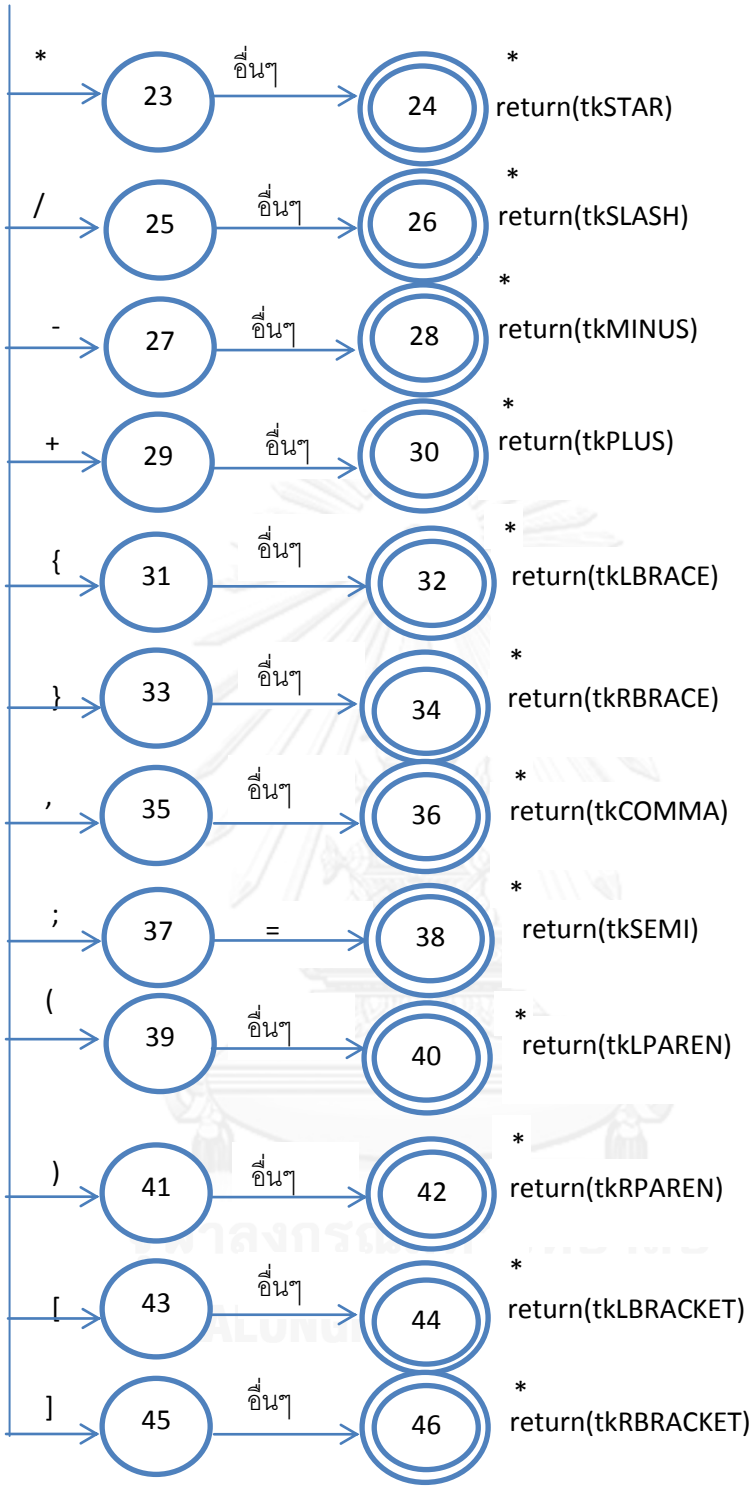
ภาพที่ 3-3 ขั้นตอนของตัววิเคราะห์ศัพท์

เมื่อมีช่องว่างและหมายเหตุจะอ่านทิ้งไปเพราะไม่ถือว่าเป็นโทเค็น ช่องว่าง ได้แก่ อักขระว่าง (space) อักขระตั้งระยะ (tab) และอักขระบรรทัดใหม่ (newline) นอกจากนี้จะมีสัญลักษณ์สิ้นสุดข้อมูลเข้า เพื่อบอกให้รู้ว่าสิ้นสุดโปรแกรมต้นฉบับและเป็นการบอกให้ขั้นตอนต่อไปทราบ การเขียนโทเค็นผิดสามารถตรวจจับได้ในขั้นตอนนี้ เช่น การเขียนอักขระที่ไม่ได้อยู่ในกลุ่มของอักขระของภาษาอาร์แซด เป็นต้น

การวิเคราะห์ศัพท์เพื่อแยกโทเค็น สามารถแสดงด้วยแผนภาพแทรนซิชัน (transition diagram) ดังภาพที่ 3-4 และ 3-5



ภาพที่ 3-4 แผนภาพทรานซิชันแสดงการแยกโทเค็น



ภาพที่ 3-5 แผนภาพทรานซิชันแสดงการแยกโทเค็น (ต่อ)

จากแผนภาพทรานซิชัน การทำงานจะเริ่มจากสถานะเริ่มต้นเป็นสถานะปัจจุบัน และจะอ่านข้อมูลจากโปรแกรมต้นฉบับมาหนึ่งอักขระ ถ้าอักขระนั้นตรงกับป้ายของเส้นเชื่อมใด ก็จะเปลี่ยนสถานะปัจจุบันให้เป็นสถานะที่เส้นเชื่อมนั้นชี้ไป และรับอักขระเข้ามาอีกหนึ่งตัว ทำการตรวจสอบว่าตรงกับอักขระที่ต้องการหรือไม่ ถ้าตรงตัววิเคราะห์ศัพท์จะยอมรับว่าพบโทเค็น โดยมันจะคืนประเภทของโทเค็นนั้นๆกลับมา แต่ถ้าอักขระที่อ่านมาไม่ตรงกับป้ายของเส้นเชื่อมใดเลย แสดงว่าไม่พบโทเค็น และศัพท์ของโทเค็นคืออักขระทั้งหมดที่อ่านมาตั้งแต่สถานะเริ่มต้น

จากแผนภาพทรานซิชันจะพบว่ามีโทเค็นทั้งหมด 24 โทเค็น และยังมีโทเค็นอื่นๆ ดังแสดงในตารางที่ 3-1

ตารางที่ 3-1 ตารางแสดงชื่อและรายละเอียดของโทเค็นแต่ละประเภท

ชื่อโทเค็น	**รายละเอียด**
tkNUMBER	โทเค็นตัวเลข
tkLE	โทเค็นเครื่องหมายน้อยกว่าเท่ากับ (<=)
tkLT	โทเค็นเครื่องหมายน้อยกว่า (<)
tkGE	โทเค็นเครื่องหมายมากกว่าเท่ากับ (>=)
tkGT	โทเค็นเครื่องหมายมากกว่า (>)
tkOROR	โทเค็นเครื่องหมาย ||
tkOR	โทเค็นเครื่องหมาย |
tkANDAND	โทเค็นเครื่องหมาย &&
tkAND	โทเค็นเครื่องหมาย &
tkNE	โทเค็นเครื่องหมายไม่เท่ากับ (!=)
tkNOT	โทเค็นเครื่องหมายไม่เท่า (!)
tkSTAR	โทเค็นเครื่องหมายคูณ (*)
tkSLASH	โทเค็นเครื่องหมายหาร (/)
tkMINUS	โทเค็นเครื่องหมายลบ (-)
tkPLUS	โทเค็นเครื่องหมายบวก (+)
tkCOMMA	โทเค็นเครื่องหมายวรรคตอน (,)
tkSEMI	โทเค็นเคื่องหมายเซมิโคลอน (;)
tkLPAREN	โทเค็นเครื่องหมายวงเล็บทางซ้าย (()
tkRPAREN	โทเค็นเครื่องหมายวงเล็บทางขวา ())

tkLBRACKET	โทเค็นเครื่องหมายวงเล็บเหลี่ยมทางซ้าย ([)
tkRBRACKET	โทเค็นเครื่องหมายวงเล็บเหลี่ยมทางขวา (])
tkLBRACE	โทเค็นเครื่องหมายปีกกาเปิดทางซ้าย ({)
tkRBRACE	โทเค็นเครื่องหมายปีกกาเปิดทางขวา (})
tkIDEN	โทเค็นตัวระบุ
tkSTRING	โทเค็นสตริง
tkIF	โทเค็นคำสงวน if
tkELSE	โทเค็นคำสงวน else
tkWHILE	โทเค็นคำสงวน while
tkRETURN	โทเค็นคำสงวน return
tkEOF	โทเค็นสัญลักษณ์สิ้นสุด

ยกตัวอย่างข้อมูลเข้าเป็น main จากแผนภาพทรานซิชันในภาพที่ 3-4 และ 3-5 สถานะเริ่มต้นจะอยู่ที่ 1 รับอักขระตัวแรกคือ m หลังจากนั้นเปลี่ยนสถานะใหม่เป็นสถานะที่ 2 รับอักขระตัวถัดไปคือ a ซึ่งตรงตามเงื่อนไข จากนั้นยังอยู่ที่สถานะที่ 2 เหมือนเดิมและรับอักขระตัวถัดไปเรื่อยๆ คือ i และ n ตามลำดับ เมื่อไม่มีอักขระตัวถัดไปแล้ว สถานะปัจจุบันไปอยู่ที่สถานะ 3 และทำการตรวจสอบว่าเป็นคำสงวนหรือไม่ ซึ่งคำสงวนของโปรแกรมภาษาอาร์แซด ได้แก่ if, else, while, return ถ้าไม่ใช่จะคืนค่าเป็นโทเค็นตัวระบุที่เป็นคำว่า main

3.2 การออกแบบตัววิเคราะห์เชิงวากยสัมพันธ์

ตัววิเคราะห์เชิงวากยสัมพันธ์ทำหน้าที่กำหนดไวยากรณ์หรือโครงสร้างของโปรแกรม ด้วยเหตุนี้จึงถูกเรียกอีกชื่อว่าขั้นตอนวิเคราะห์ไวยากรณ์ของภาษา (syntax analysis) ไวยากรณ์ของภาษาโปรแกรมมักจะถูกกำหนดโดยกฎไวยากรณ์ของไวยากรณ์ไม่พึ่งบริบท (context-free grammar) เนื่องจากไวยากรณ์ไม่พึ่งบริบทสามารถอธิบายวากยสัมพันธ์ส่วนใหญ่ของภาษาระดับสูงได้ อัลกอริทึมที่ใช้ในการตรวจสอบไวยากรณ์ของภาษาค่อนข้างแตกต่างจากอัลกอริทึมที่ใช้ตรวจสอบโทเค็นในขั้นตอนตัววิเคราะห์ศัพท์ เนื่องจากต้องใช้การเรียกซ้ำ (recursive calls) และการจัดการเกี่ยวกับกองแยก (stack) โครงสร้างข้อมูลที่ใช้เป็นตัวแทนโครงสร้างไวยากรณ์ของภาษาคือต้นไม้วิเคราะห์กระจาย (parse tree หรือ syntax tree)

ไวยากรณ์เขียนแทนด้วยสัญกรณ์ต่างๆ ดังตารางที่ 3-2

ตารางที่ 3-2 ตารางแสดงชื่อและรายละเอียดของสัญกรณ์ต่างๆ

สัญกรณ์	คำอธิบายสัญกรณ์
pass	เริ่มต้นวิเคราะห์เชิงวากยสัมพันธ์
dcl, dcl2, dcl3, formal, formals, stmt0, stmts, stmt1, stmt2, term, term1, expr, exprs	ฟังก์ชันที่ใช้ตรวจสอบไวยากรณ์ต่างๆของภาษา
block	ตรวจสอบการใช้โทเค็นเครื่องหมายปีกกาเปิดและโทเค็นเครื่องหมายปีกกาปิดตามไวยากรณ์ของภาษา
elset	ตรวจสอบการใช้โทเค็นคำสงวน else ตามไวยากรณ์ของภาษา
returnst	ตรวจสอบการใช้โทเค็นคำสงวน return ตามไวยากรณ์ของภาษา
prlist, prlists	ตรวจสอบการใช้โทเค็นคำสงวน print ตามไวยากรณ์ของภาษา
bop	ตรวจสอบการใช้โทเค็นเครื่องหมายดำเนินการต่างๆ เช่น <=, >= เป็นต้น
mod, param, params	ตรวจสอบการรับพารามิเตอร์
index	ตรวจสอบการกำหนดค่าในอาร์เรย์
eof	สิ้นสุดข้อมูลเข้า
ID	โทเค็นตัวระบุ
num	โทเค็นตัวเลข
,	โทเค็นเครื่องหมายวรรคตอน
;	โทเค็นเครื่องหมายเซมิโคลอน
(	โทเค็นเครื่องหมายวงเล็บทางซ้าย
)	โทเค็นเครื่องหมายวงเล็บทางขวา
[	โทเค็นเครื่องหมายวงเล็บเหลี่ยมทางซ้าย
]	โทเค็นเครื่องหมายวงเล็บเหลี่ยมทางขวา

สามารถเขียนไวยากรณ์ทั้งหมดของภาษาอาร์แซดด้วยไวยากรณ์ไม่พึ่งบริบทดังนี้

pass ⟶ dcl pass | eof

dcl ⟶ ID dcl2

dcl2 ⟶ (formal) stmt0 | [num] dcl3 | , | ;

dcl3 ⟶ , | ;

formal ⟶ ID formals

formals ⟶ , formal

stmt0 ⟶ block | stmt1

stmts ⟶ stmt1

block ⟶ { stmts }

stmt1 ⟶ ; | if (expr) stmt0 elsest | while (expr) stmt0 | return returnst | print (prlist) ; | ID = expr ; | ID stmt2

elsest ⟶ else stmt0

returnst ⟶ ; | expr ;

prlist ⟶ string prlists | expr prlists

prlists ⟶ , prlist

stmt2 ⟶ = expr ; | [expr] = expr ; | (param) ;

param ⟶ expr params

params ⟶ , param

expr ⟶ term exprs

exprs ⟶ loop term

bop ⟶ || | && | < | <= | == | != | >= | > | + | - | * | /

term ⟶ - term1 | ! term1 | * term1 | & ID index | term1

term1 ⟶ ID mod | num | string | (expr)

mod ⟶ (param) | index

index ⟶ [expr]

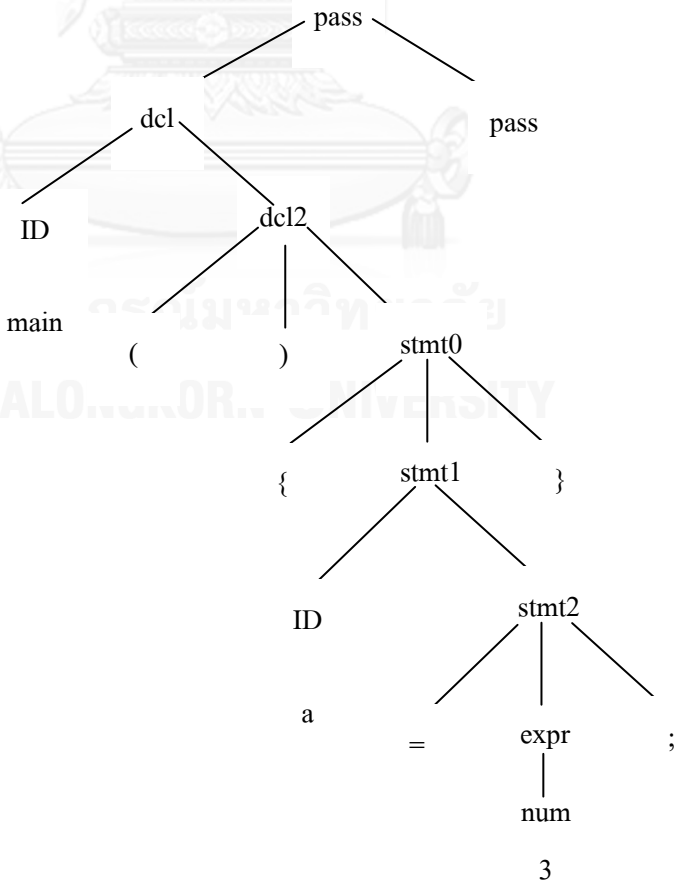
ตัววิเคราะห์เชิงวากยสัมพันธ์ (parser) จะตรวจสอบการเป็นสมาชิกของภาษาที่อธิบายด้วยไวยากรณ์ จากไวยากรณ์ไม่พึ่งบริบท การวิเคราะห์การกระจายเป็นแบบบนลงล่าง (top-down parsing) เพื่อตรวจสอบข้อมูลเข้าว่าเป็นประโยคของภาษาที่ผลิตโดยไวยากรณ์หรือไม่ โดยเริ่มจากสัญลักษณ์เริ่มต้น แล้วใช้กฎในการแทนที่สายอักขระย่อย (substring) เรียกว่า การอนุพัทธ์ (derivation) ถ้าสายอักขระที่ได้มีสัญลักษณ์ไม่สิ้นสุด หากฎมาแทนสัญลักษณ์ไม่สิ้นสุดดังกล่าว ทำเช่นนี้ซ้ำๆ จนได้ประโยคตามที่ต้องการและได้ผลลัพธ์เป็นต้นไม้วิเคราะห์กระจาย แต่ถ้าไม่ได้แสดงว่าข้อมูลเข้านั้นไม่เป็นประโยค

ตัวอย่างโปรแกรมภาษาอาร์แซด ดังภาพที่ 3-6

```
main () {
    a = 3;
}
```

ภาพที่ 3-6 ตัวอย่างโปรแกรมภาษาอาร์แซด

เมื่อผ่านตัววิเคราะห์การกระจายที่ตรวจสอบการเป็นสมาชิกของภาษาที่อธิบายด้วยไวยากรณ์ไม่พึ่งบริบท จะได้ต้นไม้วิเคราะห์กระจาย (parse tree) ดังภาพที่ 3-7



ภาพที่ 3-7 ต้นไม้วิเคราะห์กระจายของโปรแกรมในภาพที่ 3-6

3.3 การออกแบบตัววิเคราะห์ความหมาย

การวิเคราะห์ความหมายและการผลิตรหัสสามารถทำไปพร้อมๆกันได้ โดยมีการกำหนดลักษณะเฉพาะ (attribute) ให้กับสัญลักษณ์ในกฎ โดยลักษณะเฉพาะสามารถเป็น ชนิดข้อมูลของตัวแปร ค่าของนิพจน์ ตำแหน่งที่อยู่ของตัวแปรในหน่วยความจำ หรือค่าตัวเลข ลักษณะเฉพาะดังกล่าวจะใช้เป็นข้อมูลในการตรวจสอบความหมายและผลิตรหัสในขั้นตอนตัวก่อกำเนิดรหัส

ลักษณะเฉพาะแบ่งออกเป็น 2 แบบ คือ ลักษณะเฉพาะสังเคราะห์ (synthesized attribute) และลักษณะเฉพาะถ่ายทอด (inherited attribute) โดยลักษณะเฉพาะสังเคราะห์ คือการคำนวณลักษณะเฉพาะที่โหนดลูกและส่งผลการคำนวณมายังโหนดแม่ของต้นไม้วิเคราะห์กระจาย (parse tree) ส่วนลักษณะเฉพาะถ่ายทอด คือการคำนวณลักษณะเฉพาะที่โหนดแม่หรือโหนดพี่น้อง (sibling) และส่งผลการคำนวณมายังโหนดลูก ซึ่งการคำนวณของแต่ละลักษณะเฉพาะต้องเป็นไปตามกฎไวยากรณ์ของภาษาอาร์แซด ซึ่งไวยากรณ์ที่มีลักษณะเฉพาะและการคำนวณเป็นไปตามกฎเรียกว่า ไวยากรณ์ลักษณะเฉพาะ (attribute grammar)

การคำนวณลักษณะเฉพาะเป็นดังนี้

1. ลักษณะเฉพาะถ่ายทอดของสัญลักษณ์ที่อยู่ในสายอักขระทางขวาของกฎ คำนวณมาจากลักษณะเฉพาะของสัญลักษณ์ในกฎ
2. ลักษณะเฉพาะสังเคราะห์ของสัญลักษณ์ที่อยู่ในสายอักขระทางซ้ายของกฎ คำนวณมาจากลักษณะเฉพาะของสัญลักษณ์ในกฎ

พิจารณาไวยากรณ์พื้นฐานสำหรับเลขไม่มีสัญลักษณ์ดังต่อไปนี้

number ⟶ number digit | digit

digit ⟶ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

กฎไวยากรณ์ digit ⟶ 0 หมายถึง digit มีค่าเป็น 0

สามารถแสดงการคำนวณลักษณะเฉพาะได้เป็น

digit.val = 0

ถ้าจำนวนประกอบด้วยเลขตัวเดียว สามารถใช้กฎ

number ⟶ digit

แสดงการคำนวณลักษณะเฉพาะได้เป็น

number.val = digit.val

ถ้าจำนวนประกอบด้วยเลขมากกว่า 1 ตัว สามารถใช้กฎ

number ⟶ number digit

แต่ number ทางขวามือกับ number ทางซ้ายมือมีค่าแตกต่างกัน ดังนั้นเราจึงต้องใช้เลขกำกับ number แต่ละตัว เขียนเป็นกฎใหม่ได้ดังนี้

number1 ⟶ number2 digit

แสดงการคำนวณลักษณะเฉพาะได้เป็น

number1.val = number2.val * 10 + digit.val

ยกตัวอย่างกฎไวยากรณ์ (grammar rule) ในภาษาอาร์แซด

exp1 ⟶ exp2 + term

exp1 ⟶ exp2 – term

exp ⟶ term

term1 ⟶ term2 * factor

term ⟶ factor

factor ⟶ (exp)

factor ⟶ number

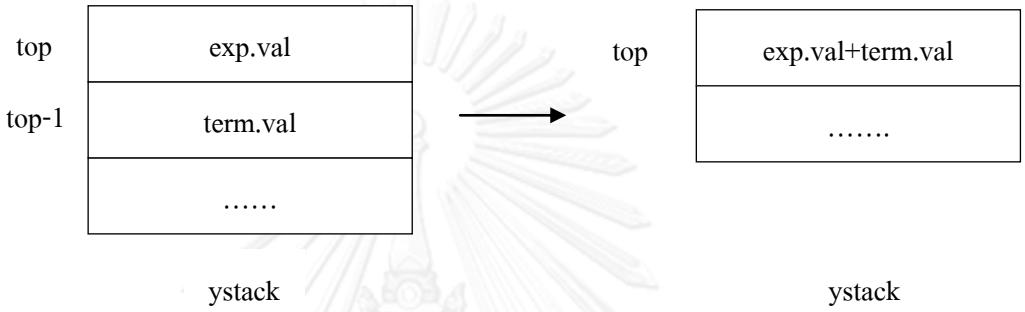
จากกฎไวยากรณ์ข้างบน สามารถเขียนเป็นกฎไวยากรณ์ที่มีความหมายกำกับ (semantic rule) ดังนี้

exp1.val	=	exp2.val + term.val
exp1.val	=	exp2.val – term.val
exp.val	=	term.val
term1.val	=	term2.val * factor.val
term.val	=	factor.val
factor.val	=	exp.val
factor.val	=	number.val

ลักษณะเฉพาะที่คำนวณได้จะถูกเก็บไว้ในกองซ้อน (stack)

ถ้าให้ ystack เป็นแถวลำดับแทนกองซ้อน และ top เป็นดรรชนีที่ยอดกองซ้อน โดยเริ่มต้นกองซ้อนว่างเปล่า จากสายอักขระทางขวาของกฎเรียงจากซ้ายไปขวา สัญลักษณ์ที่มีลักษณะเฉพาะจะ

อยู่ที่ ystack[top] และ ystack[top-1] ตามลำดับ ยกตัวอย่างกฎ exp ⟶ exp + term ลักษณะเฉพาะ exp.val จะอยู่ที่ ystack[top] ส่วนสัญลักษณ์ + ไม่มีลักษณะเฉพาะกำกับจึงไม่เก็บลงกองซ้อน ดังนั้นลักษณะเฉพาะ term.val จะอยู่ที่ ystack[top-1] การคำนวณลักษณะเฉพาะ exp.val + term.val เขียนได้เป็น ystack[top] + ystack[top-1] และนำค่านี้ไปเก็บเป็นลักษณะเฉพาะของสัญลักษณ์ทางซ้ายของกฎ นั่นก็คือ exp.val ดังนั้นที่ยอดกองซ้อนจะมีค่าลักษณะเฉพาะของ exp.val + term.val แสดงดังภาพที่ 3-8



ภาพที่ 3-8 ลักษณะของกองซ้อนขณะที่มีการคำนวณและเก็บลักษณะเฉพาะ

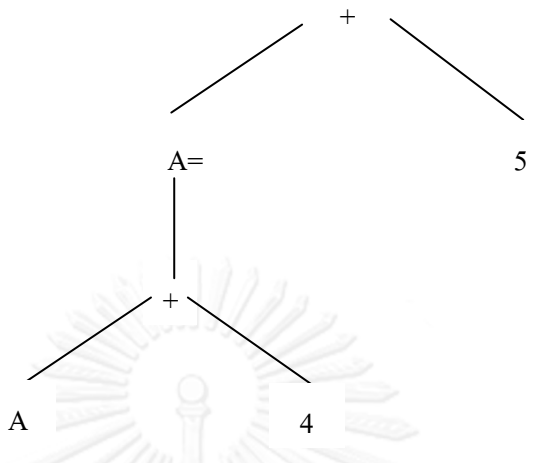
นอกจากนี้มีการเก็บข้อมูลต่างๆเกี่ยวตัวแปรเฉพาะที่ (local variable) และตัวแปรส่วนกลาง (global variable) ไว้ในตารางสัญลักษณ์ (symbol table) โดยจะกล่าวรายละเอียดเกี่ยวกับข้อมูลต่างๆที่เก็บในตารางสัญลักษณ์ในหัวข้อการออกแบบตารางสัญลักษณ์

3.4 การออกแบบตัวก่อกำเนิดรหัส

ขั้นตอนการออกแบบตัวก่อกำเนิดรหัสถือเป็นขั้นตอนที่ซับซ้อนที่สุดของการออกแบบขั้นตอนต่างๆในกระบวนการแปลภาษาของตัวแปลภาษา เนื่องจากการออกแบบตัวก่อกำเนิดรหัสไม่ได้คำนึงถึงอักขระต่างๆของโปรแกรมภาษาต้นฉบับเพียงอย่างเดียว แต่ยังคำนึงถึงสถาปัตยกรรมของภาษาเป้าหมาย โครงสร้างสภาพแวดล้อมในการทำงาน เช่น ขนาดของชนิดข้อมูล ตำแหน่งที่อยู่ของตัวแปร จำนวนรีจิสเตอร์ที่มีให้ใช้ และระบบปฏิบัติการในการทำงานของภาษาเป้าหมาย เป็นต้น โดยในงานวิจัยนี้ เราทำการออกแบบขั้นตอนตัววิเคราะห์ความหมายและขั้นตอนตัวก่อกำเนิดรหัสให้สามารถทำงานไปพร้อมกันได้ และรหัสระหว่างกลางที่ตัวก่อกำเนิดรหัสนำมาใช้ผลิตภาษาเป้าหมายซึ่งในงานวิจัยนี้คือภาษาแอสเซมบลี คือ ต้นไม้วิเคราะห์กระจาย

จากขั้นตอนตัววิเคราะห์ความหมาย ผลลัพธ์ที่ได้คือต้นไม้วิเคราะห์กระจายที่มีลักษณะเฉพาะ (attribute) กำกับเพื่อใช้เป็นข้อมูลในการตรวจสอบความหมายและผลิตรหัสต่อไป

ตัวอย่างต้นไม้วิเคราะห์กระจายดังภาพที่ 3-9



ภาพที่ 3-9 ต้นไม้วิเคราะห์กระจาย

จากตัวอย่างต้นไม้วิเคราะห์กระจายดังภาพที่ 3-9 เราสามารถเขียนขั้นตอนการผลิตรหัสโค้ดเทียมแบบเรียกซ้ำ (recursive call)

การแวะผ่านต้นไม้แบบเรียกซ้ำ (recursive tree traversal) อาจเป็นการแวะผ่านก่อนลำดับ (preorder traversal) หมายถึงการแวะผ่านโดยให้ความสำคัญรากก่อนแล้วจึงแวะผ่านต้นไม้ย่อยของลูกจากซ้ายไปขวา หรือการแวะผ่านตามลำดับ (inorder traversal) หมายถึงการแวะผ่านโดยให้ความสำคัญต้นไม้ย่อยของลูกซ้ายก่อน และกลับมาแวะที่ราก แล้วจึงแวะผ่านต้นไม้ย่อยทางขวา และกลับมาแวะที่ราก เช่นนี้ไปเรื่อยๆสลับกันไป จนถึงต้นไม้ย่อยของลูกสุดท้าย และการแวะผ่านหลังลำดับ (postorder traversal) หมายถึงการแวะผ่านต้นไม้ย่อยของลูกเรียงจากซ้ายไปขวา แล้วจึงแวะผ่านรากทีหลัง โดยถือว่าแต่ละโหนดของต้นไม้วิเคราะห์กระจายมีโหนดลูกมากสุด 2 โหนด คือ โหนดซ้ายและโหนดขวา

จำนวนรีจิสเตอร์ที่ใช้สำหรับการออกแบบตัวก่อกำเนิดรหัสคือ 32 ตัว

ตัวอย่างโปรแกรมดังภาพที่ 3-10

```
main(){
      c = a + b;
}
```

ภาพที่ 3-10 ตัวอย่างโปรแกรมภาษาอาร์แซด

หลังจากประมวลผลแล้วจะได้ผลลัพธ์เป็นภาษาแอสเซมบลีซึ่งจำเป็นต้องมีการใช้รีจิสเตอร์ดังภาพที่ 3-11

```
ld adr(a) r1
ld adr(b) r2
add r3 r1 r2
str adr(c) r3
```

ภาพที่ 3-11 รหัสแอสเซมบลี

โดย r1, r2, r3 คือ รีจิสเตอร์ตัวที่ 1 ตัวที่ 2 และตัวที่ 3 ตามลำดับ

ld adr(a) r1 คือคำสั่งโหลดค่า a ไปเก็บไว้ที่รีจิสเตอร์ตัวที่ 1

ld adr(b) r2 คือคำสั่งโหลดค่า b ไปเก็บไว้ที่รีจิสเตอร์ตัวที่ 2

add r3 r1 r2 คือคำสั่งนำค่าในรีจิสเตอร์ตัวที่ 1 และตัวที่ 2 มาบวกกัน แล้วนำไปเก็บไว้ที่รีจิสเตอร์ตัวที่ 3

str adr(c) r3 คือคำสั่งนำค่าจากรีจิสเตอร์ตัวที่ 3 ไปเก็บไว้ในค่า c

นอกจากนี้จะใช้ฟังก์ชันมาตรฐานของภาษาซี คือ sprintf เพื่อนำผลลัพธ์ที่อยู่ในรูปแบบต่างๆ เปลี่ยนเป็นตัวแปรสตริงและสามารถแสดงค่าได้ ซึ่งจะนำมาใช้ในการผลิตโปรแกรมภาษาเป้าหมายที่เป็นภาษาแอสเซมบลี โดยผู้วิจัยได้เขียนฟังก์ชัน sprintf ขึ้นใหม่ด้วยภาษาจาวาสคริปต์เพื่อให้สามารถทำงานบนเว็บได้

3.5 การออกแบบตารางสัญลักษณ์

ตารางสัญลักษณ์มีหน้าที่สำหรับเก็บข้อมูลต่างๆของตัวระบุ เพื่อใช้ตรวจสอบวากยสัมพันธ์ส่วนที่ไวยากรณ์ไม่พึ่งบริบทไม่สามารถอธิบายได้ เช่น จำนวนและแบบชนิดข้อมูลของพารามิเตอร์ในตอนเรียกใช้ต้องตรงกับตอนที่ประกาศไว้ หรือการประกาศตัวระบุก่อนการอ้างถึง เป็นต้น นอกจากนี้ยังใช้ตรวจสอบส่วนที่เป็นความหมายของตัวระบุ เช่น นิพจน์ x * y มีแบบชนิดข้อมูลของ x และ y ที่สามารถนำมาคำนวณด้วยตัวดำเนินการ * ได้หรือไม่ เพราะบางครั้งอาจต้องมีการเปลี่ยนแบบชนิดข้อมูลก่อนคำนวณ ถ้าไม่สามารถคำนวณได้จะรายงานความผิดพลาดให้ทราบ แต่ถ้าคำนวณได้ จะผลิตรหัสสำหรับการคูณระหว่าง x กับ y

โครงสร้างข้อมูลของตารางสัญลักษณ์สำหรับงานวิจัยนี้ ออกแบบเป็นโครงสร้างข้อมูลแบบออบเจกต์ โดยตัวระบุต่างๆจะถูกเก็บเป็นออบเจกต์ และแต่ละออบเจกต์จะมีคุณสมบัติ (properties) ต่างๆที่ถูกเก็บไว้ ได้แก่ ชื่อของตัวระบุ ชนิดของตัวระบุ เช่น ตัวแปรเฉพาะที่ (local variable) ตัวแปรส่วนกลาง (global variable) หรือชื่อฟังก์ชัน เป็นต้น นอกจากนี้ยังเก็บที่อยู่ของตัวระบุ ลักษณะเฉพาะของพารามิเตอร์ที่รับมา และลักษณะเฉพาะของพารามิเตอร์และตัวระบุทั้งหมดในฟังก์ชัน

การดำเนินการกับตารางสัญลักษณ์คือ การเพิ่ม และการค้นหา สำหรับภาษาอาร์แซดต้องมีการประกาศตัวระบุก่อนการอ้างถึงกรณีที่เป็นตัวแปรส่วนกลาง ดังนั้นจึงต้องมีการเพิ่มตัวระบุเข้าไปในตารางสัญลักษณ์ในตอนประกาศ และการค้นหาจะทำเมื่อมีการอ้างถึงตัวระบุนี้ หรือเมื่อต้องการตรวจสอบการตั้งชื่อซ้ำกันของตัวระบุ กรณีที่มีการอ้างถึงตัวระบุและนำชื่อของตัวระบุไปค้นหาในตารางสัญลักษณ์แล้วไม่พบ แสดงว่าเป็นตัวระบุใหม่ที่ต้องเพิ่มเข้าไปในตารางสัญลักษณ์

ตัวอย่างโปรแกรมดังภาพที่ 3-12

```
ax[20];
main(){
  max = ax[0];
  i = 1;
  while(i < 10){
     if(ax[i] > max){
        max = ax[i];
     }
     i = i + 1;
  }
```


ภาพที่ 3-12 ตัวอย่างโปรแกรมภาษาอาร์แซด หาจำนวนที่มากที่สุดในอาร์เรย์

สามารถแสดงตารางสัญลักษณ์มีที่โครงสร้างแบบออบเจกต์ได้ดังตารางที่ 3-3

ตารางที่ 3-3 ตารางแสดงข้อมูลที่เก็บในตารางสัญลักษณ์

ดรรชนีกำกับ	ชื่อตัวระบุ	ประเภทตัวระบุ	ที่อยู่ของตัวระบุ	ลักษณะเฉพาะของพารามิเตอร์	ลักษณะเฉพาะของพารามิเตอร์และภายในฟังก์ชัน
1	ax	ตัวแปรส่วนกลาง	0		อาร์เรย์ 1 มิติ
2	main	ชื่อฟังก์ชัน	1		
3	max	ตัวแปรเฉพาะที่	2		สเกล่าร์
4	i	ตัวแปรเฉพาะที่	3		สเกล่าร์

3.6 การออกแบบรายงานความผิดพลาด

สำหรับการรายงานความผิดพลาดจะเริ่มตั้งแต่ขั้นตอนแรก คือ ตัววิเคราะห์ศัพท์ โดยมีการตรวจสอบว่าผู้เขียนโปรแกรมได้มีการพิมพ์โปรแกรมต้นฉบับลงไปก่อนทำการประมวลผลหรือไม่ และขนาดของข้อมูลเข้าของโปรแกรมต้นฉบับต้องไม่เกินที่กำหนดไว้ คือ 40,000 ไบท์ นอกจากนี้จำนวนอักขระของแต่ละโทเค็นต้องไม่เกิน 256 อักขระ

ขั้นตอนตัววิเคราะห์เชิงวากยสัมพันธ์ มีการตรวจสอบการเรียงของโทเค็นว่าถูกต้องตามไวยากรณ์ของภาษาหรือไม่ เช่น ตัวอย่างโปรแกรมดังภาพที่ 3-13

```
main(){
      a=a+3
}
```

ภาพที่ 3-13 ตัวอย่างโปรแกรมภาษาอาร์แซดที่ผิด

เมื่อสังเกตโปรแกรมจากภาพที่ 3-13 จะเห็นว่าข้างหลัง a+3 ไม่มีเครื่องหมายเซมิโคลอน ซึ่งถือว่าผิดตามไวยากรณ์ของภาษาอาร์แซด ดังนั้นจะมีการรายงานความผิดพลาดว่าตรวจไม่พบเครื่องหมายเซมิโคลอนภายหลังจากการประมวลผลโปรแกรมเสร็จสิ้นแล้ว

ขั้นตอนตัววิเคราะห์ความหมาย มีการตรวจสอบขนาดของกองซ้อนที่เก็บข้อมูลประเภทสตริงว่ามีการเก็บข้อมูลจนเกินขนาดของกองซ้อนหรือไม่ และกรณีดึงข้อมูลสตริงออกจากกองซ้อน มีการ

ดึงข้อมูลที่น้อยเกินเก็บหรือไม่ ถ้ามีจะรายงานความผิดพลาดเช่นกัน นอกจากนี้มีการตรวจสอบชนิดของข้อมูลว่าถูกต้องตามไวยากรณ์ของภาษาหรือไม่ และการตรวจสอบการประกาศตัวแปรส่วนกลางซ้ำ ตรวจสอบการเขียนพารามิเตอร์ ตรวจสอบการเรียกใช้งานกองซ้อนสำหรับเก็บโทเค็นประเภทตัวดำเนินการ และตรวจสอบการใช้งานตารางสัญลักษณ์เพื่อไม่ให้มีข้อมูลมากเกินขนาดของตารางสัญลักษณ์

สำหรับขั้นตอนตัวก่อกำเนิดรหัส เมื่อมาถึงขั้นตอนนี้โปรแกรมต้นฉบับจะถือว่าได้รับการตรวจสอบความผิดพลาดต่างๆมาเกือบสมบูรณ์แล้ว โดยขั้นตอนนี้จะมีการตรวจสอบอะตอมหรือโหนดต่างๆในต้นไม้วิเคราะห์กระจายว่าเป็นประเภทที่ถูกต้องหรือไม่ และตรวจสอบโทเค็นเพื่อผลิตรหัสให้ถูกต้องตามประเภทโทเค็น ถ้าโทเค็นที่รับมาไม่เข้าเงื่อนไขจะรายงานความผิดพลาดให้ทราบ

บทที่ 4
การพัฒนาตัวแปลภาษาอาร์แซดบนเว็บที่ทำงานบนเครื่องผู้รับบริการ

ในบทนี้จะกล่าวถึงการพัฒนาตัวแปลภาษาอาร์แซดบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB ซึ่งเขียนด้วยภาษาจาวาสคริปต์ (JavaScript language) พร้อมทั้งกล่าวถึงการพัฒนาเว็บอินเตอร์เฟซสำหรับเป็นส่วนติดต่อผู้ใช้เพื่อการใช้งานตัวแปลภาษาดังกล่าว ซึ่งเขียนด้วยภาษา HTML

4.1 ข้อกำหนดของการพัฒนาตัวแปลภาษาบนเว็บ

การพัฒนาตัวแปลภาษาให้สามารถทำงานบนเว็บได้นั้น ภาษาที่ใช้ในการพัฒนาต้องเป็นภาษาที่เว็บเบราว์เซอร์ทั่วไปรองรับและสามารถประมวลผลข้อมูลที่ฝั่งของเครื่องผู้รับบริการได้ เพื่อให้การเขียนโปรแกรมบนเว็บเป็นไปได้และสามารถประมวลผลบนเครื่องผู้รับบริการได้ทันที ภาษาที่ผู้วิจัยเลือกนำมาใช้ในการพัฒนาคือภาษาจาวาสคริปต์ เว็บเบราว์เซอร์โดยทั่วไปจะมีจาวาสคริปต์เอนจิน (Javascript Engine) เป็นของตนเอง ซึ่งจะทำหน้าที่แปลภาษาจาวาสคริปต์และประมวลผลโดยทันที

ในปัจจุบันเว็บเบราว์เซอร์จำนวนมากได้ทำการวิจัยและปรับปรุงจาวาสคริปต์เอนจินของตนเองให้มีประสิทธิภาพที่สูงยิ่งขึ้น เช่นค่าย Opera เปิดตัว “Carakan” ในOpera 10.5 ซึ่งสามารถประมวลผลเร็วกว่า Chrome และ Safari ถ้าเทียบเวอร์ชันบนแมคด้วยกัน หรือเว็บเบราว์เซอร์ของค่าย Mozilla มีการพัฒนาจาวาสคริปต์เอนจินมาหลายเวอร์ชัน เริ่มจาก SpiderMonkey ที่ใช้มาตั้งแต่สมัย Netscape, TraceMonkey เริ่มใช้ใน Firefox 3.1/3.5, JägerMonkey ใน Firefox 4 และพัฒนามาเป็น IonMonkey ในปัจจุบัน ซึ่งต่างไปจาก TraceMonkey และ JägerMonkey อยู่พอสมควร เพราะเอนจินก่อนหน้านี้ใช้แนวทางแปลงโค้ดจาวาสคริปต์เป็นภาษาเครื่องโดยตรง แต่ IonMonkey ใช้แนวทางคล้ายกับ Java/.NET คือแปลงเป็นภาษาระหว่างกลางก่อนแล้วค่อยแปลงเป็นภาษาเครื่องอีกที นอกจากนี้ยังมีค่ายอื่นๆ อาทิ Chrome ที่มีการพัฒนาประสิทธิภาพของจาวาสคริปต์เอนจินของตนเองเช่นกัน แนวโน้มในอนาคตของการประมวลผลภาษาจาวาสคริปต์โดยจาวาสคริปต์เอนจินของทุกเว็บเบราว์เซอร์จึงมีประสิทธิภาพที่ดีขึ้นเรื่อยๆ และนั่นทำให้การทำงานของตัวแปลภาษาอาร์แซดบนเว็บที่ทำงานบนเครื่องผู้รับบริการมีประสิทธิภาพที่ดีขึ้นเช่นกัน

4.2 การสร้างเว็บอินเตอร์เฟส

เริ่มจากการสร้างหน้าเว็บโดยใช้ภาษา HTML ซึ่งในส่วนของหน้าเว็บมีการสร้าง Text area สำหรับให้ผู้เขียนโปรแกรมทำการเขียนโปรแกรมลงไปใน Text area นั้น และสร้างปุ่มสำหรับการกดเพื่อประมวลผลโปรแกรม โดยผลลัพธ์ของการประมวลผลโปรแกรมที่ได้จะเป็นภาษาแอสเซมบลี และจะแสดงผลไปยัง Text area ที่เป็นส่วนแสดงผลลัพธ์โปรแกรมโดยเฉพาะ

ตัวอย่างโค้ด HTML ของเว็บอินเตอร์เฟสดังภาพที่ 4-1

```
<html>

<head>

<title></title>

</head>

<body>

<form name="checkForm" method="post" action=""
onSubmit="return check()">

<p>sourcecode : <textarea name="detail" rows="25"
style="width: 100%"></textarea></p>

<p> <input type="button" value="Compile"
onclick="main();" /></p>

<p>output : <textarea id="output" name="output"
rows="25"style="width:100%"></textarea></p>

     <script type="text/javascript"

          นำเข้าไฟล์จาวาสคริปต์ทั้งหมดที่ใช้ในกระบวนการแปลภาษา

     </script>

</form>

</body>

</html>
```

ภาพที่ 4-1 ส่วนโปรแกรมของเว็บอินเตอร์เฟส

ฟังก์ชัน main() คือฟังก์ชันสำหรับเริ่มการประมวลผลเมื่อผู้เขียนโปรแกรมคลิกปุ่ม Compile

4.3 การสร้างตัววิเคราะห์ศัพท์

สร้างฟังก์ชันในการรับอักขระจาก Text area ชื่อ readTextarea() ซึ่งฟังก์ชันนี้จะทำการรับอักขระจาก Text area มาทีละอักขระ และเริ่มทำการวิเคราะห์ศัพท์โดยโปรแกรมวิเคราะห์ศัพท์ที่เป็นไปตามแผนภาพไดอะแกรมภาพที่ 3-4 และ 3-5

เมื่อการรับอักขระที่เข้ามาเป็นไปตามเงื่อนไขของโทเค็นใดๆ โปรแกรมวิเคราะห์ศัพท์จะทำการกำหนดอักขระหรือกลุ่มของอักขระให้เป็นโทเค็นประเภทนั้นๆ เช่นเมื่อรับอักขระ “*” จะกำหนดให้อักขระนี้เป็นโทเค็นประเภทเครื่องหมายคูณ (tkSTAR) หรือถ้ารับกลุ่มของอักขระมา เช่น “main” จะกำหนดให้อักขระนี้เป็นโทเค็นประเภทตัวระบุ (tkIDEN) ซึ่งโทเค็นตัวระบุยังมีการกำหนดเป็นตัวระบุประเภทต่างๆ เช่น ตัวระบุประเภทสเกล่าร์ ตัวระบุประเภทอาร์เรย์ และตัวระบุประเภทฟังก์ชัน เป็นต้น ในขั้นตอนตัววิเคราะห์ความหมาย

4.4 การสร้างตัววิเคราะห์เชิงวากยสัมพันธ์

การกำหนดไวยากรณ์หรือโครงสร้างของโปรแกรมภาษาอาร์แซด เริ่มจากการรับโทเค็นจากขั้นตอนแรกเข้ามาและมีการวิเคราะห์โทเค็นที่รับเข้ามาว่าเป็นไปตามกฎไวยากรณ์ของภาษาหรือไม่ และมีการตรวจสอบการสิ้นสุดของโทเค็นที่รับเข้ามา ดังโปรแกรมข้างล่าง

```
function pass(){
      while(dcl()){
      }
      if(tok != tkEOF){
            mylex();
            return true;
      }
      return false;
}
```

ภาพที่ 4-2 ส่วนหนึ่งของโปรแกรมตัววิเคราะห์เชิงวากยสัมพันธ์

ฟังก์ชัน dcl() คือฟังก์ชันแรกที่กำหนดไวยากรณ์ของภาษา และฟังก์ชัน mylex() คือการวิเคราะห์ศัพท์ออกมาเป็นโทเค็นแต่ละประเภท ถ้าพบว่าเป็นโทเค็นสิ้นสุดข้อมูล (tkEOF) จะสิ้นสุดกระบวนการวิเคราะห์เชิงวากยสัมพันธ์

จากไวยากรณ์ไม่พึ่งบริบทในบทที่ 3 หัวข้อ 3.2 การออกแบบตัววิเคราะห์เชิงวากยสัมพันธ์ สามารถเขียนโปรแกรมสำหรับกำหนดไวยากรณ์ของภาษาอาร์แซดโดยเริ่มจากฟังก์ชัน dcl() ได้ดังนี้

```
function dcl(){
     if( tok == tkIDEN ){
     pusht(tokstring);
          mylex();
          commit(dcl2());
          return true;
     }
     return false;
}
```

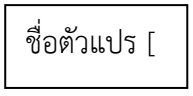
ภาพที่ 4-3 ส่วนหนึ่งของโปรแกรมตัววิเคราะห์เชิงวากยสัมพันธ์

เริ่มจากรับโทเค็นตัวระบุเข้ามา เก็บค่าโทเค็นตัวระบุนั้นลงในกองซ้อนและมีการรับโทเค็นตัวถัดไป โดยฟังก์ชัน dcl2() จะตรวจสอบโทเค็นตัวถัดไปที่มีสิทธิ์เป็นไปได้ 4 โทเค็น คือ

1. โทเค็นเครื่องหมายวงเล็บทางซ้าย ในกรณีที่โทเค็นตัวแรกที่รับเข้ามาเป็นโทเค็นตัวระบุที่เป็นชื่อฟังก์ชัน ตามไวยากรณ์ภาษา ถัดจากชื่อฟังก์ชันจึงควรเป็นโทเค็นเครื่องหมายวงเล็บทางซ้าย ดังภาพที่ 4-4

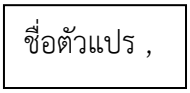
ภาพที่ 4-4 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

2. โทเค็นเครื่องหมายวงเล็บเหลี่ยมทางซ้าย ในกรณีที่โทเค็นตัวแรกรับเข้ามาเป็นโทเค็นตัวระบุที่เป็นชื่อตัวแปรส่วนกลางหรือตัวแปรเฉพาะที่ กรณีที่เป็นอาร์เรย์ถัดจากชื่อตัวแปรจึงควรเป็นโทเค็นเครื่องหมายวงเล็บเหลี่ยมทางซ้าย ดังภาพที่ 4-5



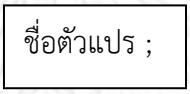
ภาพที่ 4-5 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

3. โทเค็นเครื่องหมายวรรคตอน ในกรณีที่โทเค็นตัวแรกรับเข้ามาเป็นโทเค็นตัวระบุที่เป็นชื่อตัวแปรส่วนกลางหรือตัวแปรเฉพาะที่ โทเค็นถัดไปอาจเป็นโทเค็นเครื่องหมายวรรคตอน ดังภาพที่ 4-6



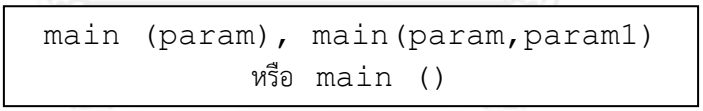
ภาพที่ 4-6 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

4. โทเค็นเครื่องหมายเซมิโคลอน ในกรณีที่โทเค็นตัวแรกรับเข้ามาเป็นโทเค็นตัวระบุที่เป็นชื่อตัวแปรส่วนกลางหรือตัวแปรเฉพาะที่ โทเค็นถัดไปอาจเป็นโทเค็นเครื่องหมายเซมิโคลอน ดังภาพที่ 4-7



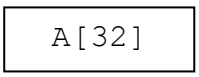
ภาพที่ 4-7 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

หลังโทเค็นเครื่องหมายวงเล็บทางซ้าย คาดหวังว่าโทเค็นต่อไปควรเป็นโทเค็นตัวระบุที่เป็นพารามิเตอร์ของฟังก์ชันแล้วตามด้วยโทเค็นเครื่องหมายวงเล็บทางขวา หรือโทเค็นตัวระบุที่เป็นพารามิเตอร์ของฟังก์ชัน ตามด้วยเครื่องหมายวรรคตอนแล้วตามด้วยโทเค็นตัวระบุที่เป็นพารามิเตอร์ของฟังก์ชัน หรือเป็นโทเค็นเครื่องหมายวงเล็บทางขวา ดังภาพที่ 4-8



ภาพที่ 4-8 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

หลังโทเค็นเครื่องหมายวงเล็บเหลี่ยมทางซ้าย คาดหวังว่าโทเค็นต่อไปควรเป็นโทเค็นตัวเลขและตามด้วยโทเค็นเครื่องหมายวงเล็บเหลี่ยมทางขวา ซึ่งเป็นการระบุขนาดของอาร์เรย์ 1 มิติดังภาพที่ 4-9



ภาพที่ 4-9 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

หลังโทเค็นเครื่องหมายวงเล็บทางขวา คาดหวังว่าโทเค็นต่อไปควรเป็นโทเค็นเครื่องหมายปีกกาทางซ้าย ซึ่งถือว่าเป็นการเริ่มเขียนการดำเนินการต่างๆภายในฟังก์ชัน ซึ่งเมื่อการดำเนินการภายในฟังก์ชันเสร็จแล้วต้องตามด้วยโทเค็นเครื่องหมายปีกกาทางขวา เพื่อเป็นการบอกว่าสิ้นสุดการทำงานฟังก์ชันนั้น ดังภาพที่ 4-10

```
main() {
        การดำเนินการภายในฟังก์ชัน
}
```

ภาพที่ 4-10 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

การดำเนินการต่างๆภายในฟังก์ชันอาจเริ่มจากการประกาศตัวแปรเฉพาะที่ภายในฟังก์ชันหรือนำตัวแปรส่วนกลางที่ประกาศภายนอกฟังก์ชันมาใช้ได้ เนื่องจากภาษาอาร์แซดมีชนิดข้อมูลชนิดเดียวคือจำนวนเต็ม ดังนั้นการดำเนินการต่างๆส่วนใหญ่จึงเป็นการดำเนินการทางคณิตศาสตร์ ได้แก่ การบวก ลบ คูณ หารกันของตัวแปร และเก็บผลลัพธ์ไว้ในตัวแปรค่าหนึ่ง หรืออาจมีการเปรียบเทียบค่าทางซ้ายและทางขวา

เมื่อมีการประกาศตัวแปรเฉพาะที่หรือนำตัวแปรส่วนกลางมาใช้ซึ่งถือว่าเป็นโทเค็นตัวระบุทั้งคู่ โทเค็นตัวถัดไปที่เป็นไปได้ได้แก่ โทเค็นเครื่องหมายเท่ากับ โทเค็นเครื่องหมายบวก โทเค็นเครื่องหมายลบ โทเค็นเครื่องหมายคูณ โทเค็นเครื่องหมายหาร และเมื่อเสร็จการดำเนินการในแต่ละบรรทัด จะต้องตามด้วยโทเค็นเครื่องหมายเซมิโคลอนเสมอ แสดงดังภาพที่ 4-11

```
a = b;
a = b + c;
a = b - c;
a = b * c;
a = b / c;
```

ภาพที่ 4-11 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

นอกจากนี้คำสงวนต่างๆของภาษาอาร์แซด ได้แก่ if, else, while, print และreturn มีกฎไวยากรณ์ในการใช้ดังนี้

1. หลังโทเค็น if ควรตามด้วยโทเค็นเครื่องหมายวงเล็บทางซ้ายและตามด้วยเงื่อนไขที่ต้องการ จากนั้นจึงตามด้วยโทเค็นเครื่องหมายวงเล็บทางขวา และเป็นโทเค็นเครื่องหมายปีกกาทางซ้ายเพื่อเริ่มการดำเนินการเมื่อเป็นไปตามเงื่อนไขที่ระบุในวงเล็บ จากนั้นเมื่อสิ้นสุดการดำเนินการจึงปิดท้ายด้วยโทเค็นเครื่องหมายปีกกาทางขวา หรือ

หลังเครื่องหมายวงเล็บทางซ้ายสามารถเขียนการดำเนินการได้เลยโดยไม่จำเป็นต้องตามด้วยเครื่องหมายปีกกาทางซ้าย โดยเงื่อนไขภายในวงเล็บหลังโทเค็น if นั้นเป็นการดำเนินการที่เป็นการเปรียบเทียบค่าทางซ้ายและทางขวา ซึ่งต้องเริ่มจากโทเค็นตัวระบุที่เป็นตัวแปรเฉพาะที่ก่อน จากนั้นจึงเป็นโทเค็นเครื่องหมายเปรียบเทียบ ได้แก่ โทเค็นเครื่องหมายน้อยกว่าเท่ากับ โทเค็นเครื่องหมายมากกว่าเท่ากับ โทเค็นเครื่องหมายน้อยกว่า โทเค็นเครื่องหมายเท่ากับ โทเค็นเครื่องหมายเท่ากับเท่ากับ โทเค็นเครื่องหมายมากกว่า และโทเค็นเครื่องหมายไม่เท่ากับ แสดงดังภาพที่ 4-12

```
if (a = b) a = a+1;
if (a != b){a = a+1;}
if (a == b) a = a+1;
if (a <= b) {a = a+1;}
if (a >= b) a = a+1;
if (a < b) {a = a+1;}
if (a > b) a = a+1;
```

ภาพที่ 4-12 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

2. หลังโทเค็น else ควรตามด้วยโทเค็นเครื่องหมายปีกกาทางซ้าย และตามด้วยการดำเนินการ จากนั้นปิดด้วยโทเค็นเครื่องหมายปีกกาทางขวา ซึ่งการใช้ else ต้องใช้คู่กับ if เสมอ เมื่อเงื่อนไขหลัง if ไม่ตรงตามที่ต้องการ โปรแกรมก็จะดำเนินการตามการดำเนินการหลัง else ทันที ดังภาพที่ 4-13

```
if (a != b) a = a+1;
else { a = a-1; }
```

ภาพที่ 4-13 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

3. หลังโทเค็น while ควรตามด้วยโทเค็นเครื่องหมายวงเล็บทางซ้าย และตามด้วยเงื่อนไขที่ต้องการ จากนั้นจึงตามด้วยโทเค็นเครื่องหมายวงเล็บทางขวา และเป็นโทเค็นเครื่องหมายปีกกาทางซ้ายเพื่อเริ่มการดำเนินการเมื่อเป็นไปตามเงื่อนไขที่ระบุในวงเล็บ จากนั้นเมื่อสิ้นสุดการดำเนินการจึงปิดท้ายด้วยโทเค็นเครื่องหมายปีกกาทางขวา ดังภาพที่ 4-14

```
while(i<10){
      i = i + 1;
}
```

ภาพที่ 4-14 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

4. หลังโทเค็น print ควรตามด้วยโทเค็นเครื่องหมายวงเล็บทางซ้าย และตามด้วยโทเค็นตัวระบุ ซึ่งก็คือตัวแปรที่ต้องการพิมพ์ค่าออกมา จากนั้นตามด้วยโทเค็นเครื่องหมายวงเล็บทางขวา และเครื่องหมายเซมิโคลอนปิดท้ายเพื่อแสดงว่าสิ้นสุดการดำเนินการ ดังภาพที่ 4-15

```
print (max);
```

ภาพที่ 4-15 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

5. หลังโทเค็น return ควรตามด้วยโทเค็นตัวระบุ ซึ่งก็คือตัวแปรที่ต้องการส่งค่ากลับไปยังฟังก์ชันก่อนหน้านี้ที่ทำการเรียกฟังก์ชันปัจจุบัน และเครื่องหมายเซมิโคลอนปิดท้ายเพื่อแสดงว่าสิ้นสุดการดำเนินการ ดังภาพที่ 4-16

```
return max;
```

ภาพที่ 4-16 ภาพแสดงการเรียงโทเค็นที่ถูกต้อง

4.5 การสร้างตัววิเคราะห์ความหมาย

ในขั้นตอนนี้มีการยุ่งเกี่ยวกับตารางสัญลักษณ์ โดยเริ่มจากการรับโทเค็นตัวระบุแล้วนำไปเก็บไว้ในตารางสัญลักษณ์ โดยถ้าตัวระบุนั้นเป็นชื่อฟังก์ชัน การเก็บข้อมูลตัวระบุในตารางสัญลักษณ์จะมีการกำหนดข้อมูลนั้นเป็นสัญลักษณ์ประเภทฟังก์ชันทันที เพื่อให้สามารถตรวจสอบชนิดของสัญลักษณ์ของโทเค็นตัวระบุว่าเป็นสัญลักษณ์ที่ถูกต้องหรือไม่ในภายหลังได้ โดยสามารถเขียนโปรแกรมได้ดังภาพที่ 4-17

```
function chktype(type1, type2) {
     if( type1 != type2 )
     seterror("identifier type     mismatch");
}

function installFunc(name, pv){
     var a, found, type, ref, arg;
     a = installGlobal(name,tyFUNCTION,0,pv,found);

     if ( found )   {
          type = getType(a);
          chktype(type,tyFUNCTION);
     }
     return a;
}
```

ภาพที่ 4-17 ส่วนหนึ่งของโปรแกรมตัววิเคราะห์ความหมาย

ฟังก์ชัน chktype คือฟังก์ชันตรวจสอบความถูกต้องของสัญลักษณ์ตัวระบุ และฟังก์ชัน installGlobal คือฟังก์ชันสำหรับการเก็บข้อมูลตัวระบุในตารางสัญลักษณ์ ส่วนฟังก์ชัน getType คือฟังก์ชันรับค่าสัญลักษณ์หรือประเภทของตัวระบุ

นอกจากนี้ตัวระบุประเภทตัวแปรส่วนรวม ซึ่งไม่อาจใช้ไวยากรณ์ไม่พึ่งบริบทในการจัดการได้ จะถูกนำไปเก็บในตารางสัญลักษณ์เช่นกัน โดยขณะเก็บจะมีการระบุชนิดสัญลักษณ์ของตัวแปรว่าเป็นสเกล่าร์คือตัวแปรทั่วไป หรือประเภทเวกเตอร์คือตัวแปรอาร์เรย์ ซึ่งมีการระบุขนาดของอาร์เรย์ด้วย โดยสามารถเขียนโปรแกรมได้ดังภาพที่ 4-18

```
function putvec(name, size){
     putSym(name,tyVECTOR,DS,size);
     DS += size;
}
function putvar(name){
     putSym(name,tySCALAR,DS,1);
     DS++;
}
```

ภาพที่ 4-18 ส่วนหนึ่งของโปรแกรมตัววิเคราะห์ความหมาย

ตัวระบุประเภทตัวแปรเฉพาะที่ก็ถูกจัดเก็บลงในตารางสัญลักษณ์เช่นกัน แต่สำหรับตัวแปรเฉพาะที่จะมีการระบุชนิดของสัญลักษณ์เป็นสเกล่าร์ เนื่องจากไวยากรณ์ภาษาอาร์แซดอนุญาตให้ตัวแปรเฉพาะที่เป็นตัวแปรสเกล่าร์เท่านั้น

สำหรับการระบุตัวแปรเฉพาะที่ใดๆในฟังก์ชัน จะมีการตรวจสอบว่าตัวแปรนั้นได้เคยถูกใช้มาแล้วหรือไม่ ซึ่งจะทำการตรวจสอบจากตารางสัญลักษณ์ที่มีการเก็บข้อมูลตัวระบุทั้งหมดไว้ ถ้าไม่เคยถูกใช้เป็นตัวแปรเฉพาะที่มาก่อน ก็จะตรวจสอบต่อไปว่าเคยถูกใช้เป็นตัวแปรส่วนรวมมาก่อนหรือไม่ ถ้าไม่พบก็จะทำการเก็บค่าตัวแปรเฉพาะที่นั้นลงในตารางสัญลักษณ์

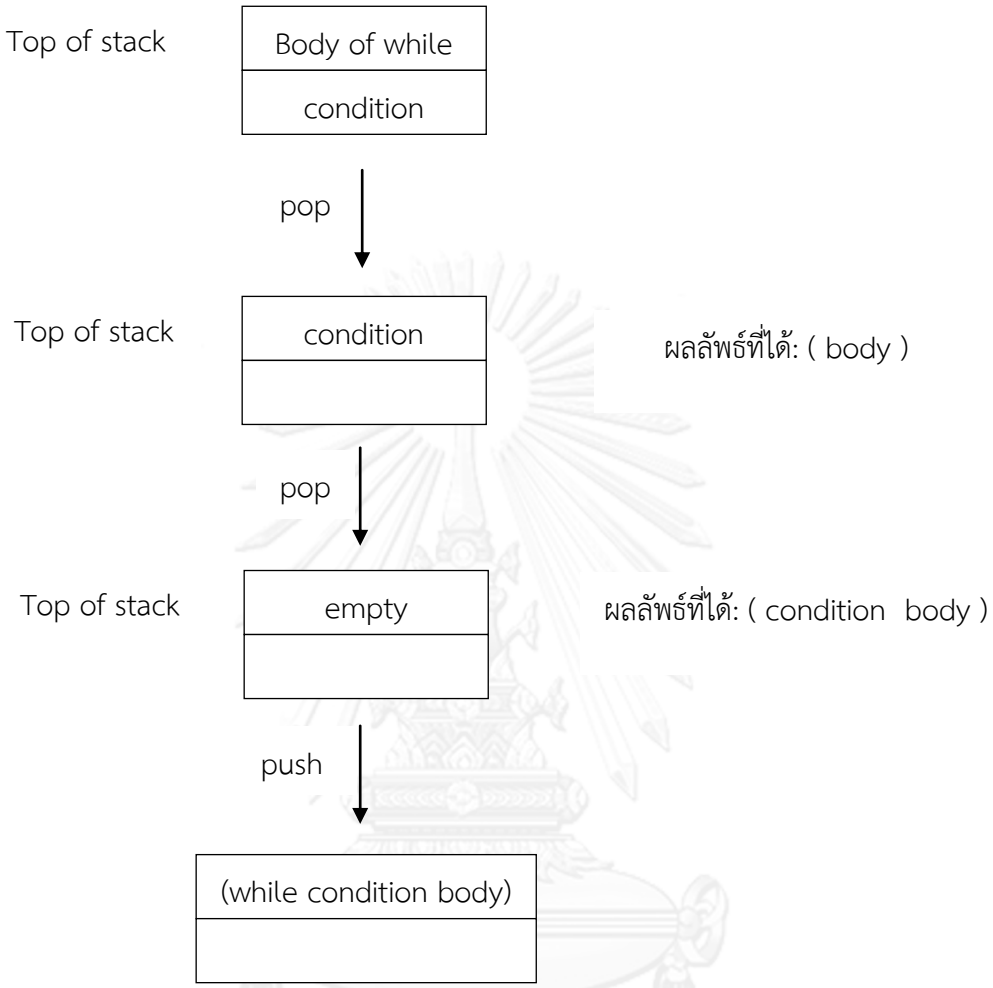
สำหรับการทำลูปของ while สามารถเขียนโปรแกรมได้ดังภาพที่ 4-19

```
function dowhile(){
    var e;
     e = ypop();
     e = cons(ypop(), list(e));
     ypush(cons(newatom(OPER, tkWHILE),e));
}
```

ภาพที่ 4-19 ส่วนหนึ่งของโปรแกรมตัววิเคราะห์ความหมาย

ฟังก์ชัน ypop() ทำการดึงโครงสร้างโปรแกรมของ while (body) ออกจากกองซ้อน ฟังก์ชัน cons() ทำการดึงเงื่อนไขของ while (condition) ออกจากกองซ้อน และมาใส่วงเล็บให้กับ body และ condition เพื่อสร้างต้นไม้วิเคราะห์กระจาย (parse tree) จากนั้นจึงใส่โทเค็น while ลงไปในกองซ้อนพร้อมกับ body และ condition

สามารถแสดงรูปภาพของขั้นตอนการทำงานของฟังก์ชัน dowhile() ได้ดังภาพที่ 4-20



ภาพที่ 4-20 ภาพแสดงขั้นตอนการทำงานของฟังก์ชัน dowhile()

การเขียนโปรแกรมสำหรับโครงสร้างอื่นๆจะใช้หลักการเดียวกับการเขียนโปรแกรมลูป while สำหรับการทำโครงสร้างของ if สามารถเขียนโปรแกรมได้ดังภาพที่ 4-21

```
function doif(){
     var e;
     e = ypop();
     e = cons(ypop(),list(e));
     ypush(cons(newatom(OPER,tkIF),e));
}
```

ภาพที่ 4-21 ส่วนหนึ่งของโปรแกรมตัววิเคราะห์ความหมาย

สำหรับการทำโครงสร้างของ ifelse สามารถเขียนโปรแกรมได้ดังภาพที่ 4-22

```
function doifelse(){
     var e1, e2, e;
     e2 = ypop();
     e1 = ypop();
     e = cons(ypop(), cons(e1,list(e2)));
     ypush(cons(newatom(OPER, tkELSE),e));
}
```

ภาพที่ 4-22 ส่วนหนึ่งของโปรแกรมตัววิเคราะห์ความหมาย

สำหรับการทำคำสั่ง return สามารถเขียนโปรแกรมได้ดังภาพที่ 4-23

```
function doreturn(){
      ypush(cons(newatom(OPER,tkRETURN)
      ,list(ypop())));
}
```

ภาพที่ 4-23 ส่วนหนึ่งของโปรแกรมตัววิเคราะห์ความหมาย

สำหรับการทำคำสั่ง print สามารถเขียนโปรแกรมได้ดังภาพที่ 4-24

```
function doprint(){
     var e, t;
     e = NIL;
     while(ytos() != MARK)
     e=cons(ypop(),e);
     t = ypop();
     ypush(cons(newatom(OPER,tkPRINT),e));
}
```

ภาพที่ 4-24 ส่วนหนึ่งของโปรแกรมตัววิเคราะห์ความหมาย

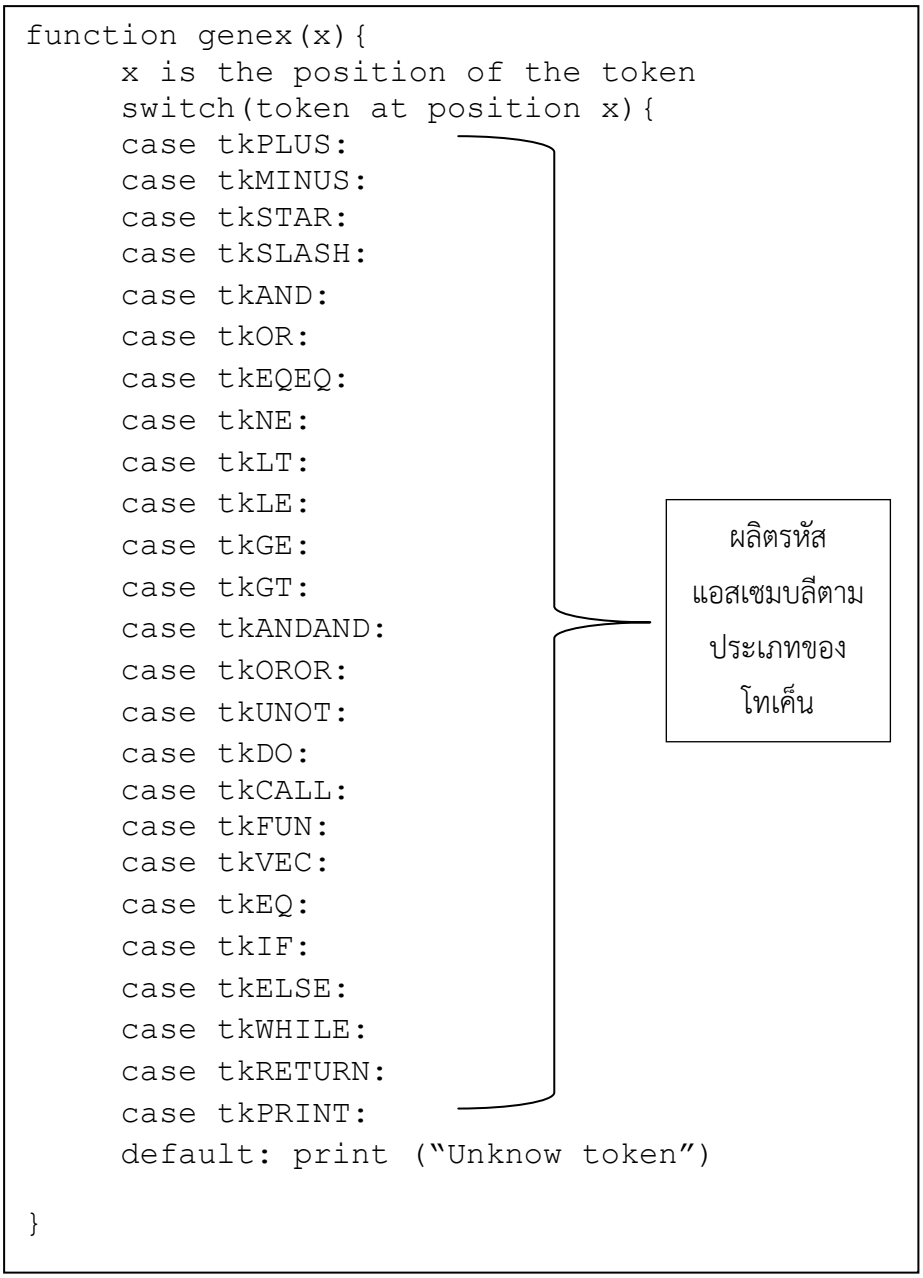
4.6 การสร้างตัวก่อกำเนิดรหัส

เริ่มจากการจัดสรรพื้นที่สำหรับการใช้รีจิสเตอร์ (register allocation) โดยมีรีจิสเตอร์ทั้งหมด 32 ตัว และการสร้างบัฟเฟอร์สำหรับเก็บการดำเนินการต่างๆ ที่จะแสดงค่าออกมาในรูปแบบภาษาแอสเซมบลี โดยเมื่อเริ่มกระบวนการแปลภาษาจนมาถึงขั้นตอนตัววิเคราะห์ความหมายจะสามารถทำงานไปพร้อมๆกับขั้นตอนตัวก่อกำเนิดรหัสได้ โดยโทเค็นแต่ละประเภทจะมีกระบวนการในการผลิตรหัสแตกต่างกัน เช่นโทเค็น if เขียนรหัสเทียมของโปรแกรมผลิตรหัสได้ดังภาพที่ 4-25

```
function gnIf(e){

     get the condition of if

     generate assembly code of
     condition of if

     set new label of e

     get the body of if

     generate assembly code of
     body of if

     return label to the end of e

}
```

ภาพที่ 4-25 ส่วนหนึ่งของโปรแกรมตัวก่อกำเนิดรหัส

ฟังก์ชัน gnIf จะถูกเรียกใช้งานเมื่อตรวจพบโทเค็น if โดยพารามิเตอร์ e คือตำแหน่งที่อยู่ของโทเค็น if และผลิตรหัสของโทเค็นให้เป็นไปตามไวยากรณ์ของภาษาอาร์แซด โดยฟังก์ชันหลักที่ทำหน้าที่ผลิตรหัสแอสเซมบลีทั้งหมดคือ ฟังก์ชัน genex ซึ่งภายในฟังก์ชันจะมีการผลิตรหัสตามประเภทของโทเค็น ดังภาพที่ 4-26



ภาพที่ 4-26 โปรแกรมการผลิตรหัสของแต่ละโทเค็นให้เป็นไปตามไวยากรณ์ภาษาอาร์แซด

เมื่อเป็นโทเค็น if ก็มีการเรียกฟังก์ชัน gnIf จากภาพที่ 4-25 เพื่อทำการผลิตรหัสแอสเซมบลีของ โทเค็น if ให้เป็นไปตามไวยากรณ์ของภาษาอาร์แซด

4.7 การสร้างตารางสัญลักษณ์

ตารางสัญลักษณ์ถูกพัฒนาเป็นโครงสร้างแบบออบเจ็กต์ เพื่อให้การเก็บข้อมูลต่างๆเกี่ยวกับตัวระบุทำได้ง่ายขึ้นและง่ายต่อการเรียกใช้งาน เนื่องจากข้อมูลแบบออบเจ็กต์จะมีพร็อพเพอตี้ (property) ต่างๆที่เกี่ยวข้องกัน โปรแกรมสำหรับการสร้างตารางสัญลักษณ์ที่เป็นโครงสร้างออบเจ็กต์ดังภาพที่ 4-27

```
function
sym_entry(name,type,ref,arg,fs) {
     this.name=name;
     this.type=type;
     this.ref=ref;
     this.arg=arg;
     this.fs=fs;
}
var symtab = new Array();
for (var i=0; i<TABLESIZES; i++){
    symtab[i] = new sym_entry();
}
```

ภาพที่ 4-27 โปรแกรมสำหรับการสร้างตารางสัญลักษณ์ที่เป็นโครงสร้างออบเจ็กต์

ฟังก์ชัน sym_entry คือการสร้างออบเจ็กต์ใหม่ โดยมีการรับพารามิเตอร์ต่างๆ ได้แก่ ชื่อตัวระบุ ชนิดตัวระบุ ตำแหน่งที่อยู่ตัวระบุ ลักษณะเฉพาะของพารามิเตอร์ที่รับมา และลักษณะเฉพาะของพารามิเตอร์และตัวระบุทั้งหมดในฟังก์ชันตามลำดับ ซึ่งพารามิเตอร์ที่รับเข้ามาก็คือกลุ่มพร็อพเพอตี้ของออบเจ็กต์ประเภท sym_entry นั่นเอง สำหรับการสร้างตารางสัญลักษณ์ที่มีโครงสร้างเป็นออบเจ็กต์ประเภท sym_entry นั้น เริ่มจากประกาศตัวแปรประเภทอาร์เรย์ชื่อ symtab แทนตารางสัญลักษณ์ จากนั้นใช้ลูป for ในการทำให้ตัวแปร symtab ในแต่ละช่องอาร์เรย์เป็นโครงสร้างแบบออบเจ็กต์

สำหรับการเก็บข้อมูลตัวระบุลงในตารางสัญลักษณ์ มีการเก็บตัวแปรเฉพาะที่ ตัวแปรส่วนกลาง ชื่อฟังก์ชัน ลักษณะเฉพาะของพารามิเตอร์ที่รับมา และลักษณะเฉพาะของพารามิเตอร์และตัวระบุทั้งหมดในฟังก์ชัน โดยในการเก็บแต่ละครั้งจะมีการกำหนดสัญลักษณ์ให้กับตัวระบุนั้นๆได้แก่ ตัวระบุชนิดตัวแปรส่วนกลางแบบอาร์เรย์ (tyVECTOR), ตัวระบุชนิดตัวแปรส่วนกลาง (tySCALAR), ตัวระบุชนิดชื่อฟังก์ชัน (tyFUNCTION) และตัวระบุชนิดตัวแปรเฉพาะที่ (tyLOCAL) โดยมีการกำหนดขนาดของตารางสัญลักษณ์ด้วย

4.8 การสร้างรายงานความผิดพลาด

เริ่มตั้งแต่ขั้นตอนแรก คือ ตัววิเคราะห์ศัพท์ โดยมีการตรวจสอบว่าผู้เขียนโปรแกรมได้มีการพิมพ์โปรแกรมต้นฉบับลงไปก่อนทำการประมวลผลหรือไม่ และจำนวนอักขระของแต่ละโทเค็นต้องไม่เกิน 256 อักขระ ดังตัวอย่างของโปรแกรมภาพที่ 4-28 และภาพที่ 4-29

```
if( str == ''''){
          alert('Please enter some remarks');
          return false;
     }
```

ภาพที่ 4-28 โปรแกรมตรวจสอบอักขระเข้าว่ามีหรือไม่

```
if( len <= 0 ) {
     alert("lex error token length = 0");
     break;
}
     if( len > 255 ) {
     alert("error token too long
     >256 char");
     break;
     }
```

ภาพที่ 4-29 โปรแกรมตรวจสอบอักขระเข้าว่ายาวหรือสั้นเกินไปหรือไม่

ขั้นตอนตัววิเคราะห์เชิงวากยสัมพันธ์ มีการตรวจสอบการเรียงของโทเค็นว่าถูกต้องตามไวยากรณ์ของภาษาหรือไม่ ดังตัวอย่างโปรแกรมในภาพที่ 4-30

```
case tkLBRACKET:
     mylex();
     expect(tkNUMBER,"missing tkNUMBER");
```

ภาพที่ 4-30 โปรแกรมตรวจสอบการเรียงของโทเค็นให้ถูกต้องตามไวยากรณ์ภาษา

จากภาพที่ 4-30 เมื่อโทเค็นที่รับมาเป็นโทเค็นเครื่องหมายวงเล็บเหลี่ยมทางซ้าย ซึ่งเครื่องหมายนี้ใช้ในการประกาศตัวแปรอาร์เรย์ เพื่อให้เป็นไปตามกฎไวยากรณ์ของภาษาจึงควรตามด้วยโทเค็นตัวเลข ดังนั้นจึงมีฟังก์ชัน expect โดยพารามิเตอร์ของฟังก์ชันนี้คือ (tkNUMBER, "missing tkNUMBER") ซึ่งหมายความว่า ถ้าโทเค็นตัวถัดไปไม่ใช่โทเค็นตัวเลข ก็จะรายงานความผิดพลาดโดยแสดงข้อความ "missing tkNUMBER"

ขั้นตอนตัววิเคราะห์ความหมาย มีการตรวจสอบขนาดของกองซ้อนที่เก็บข้อมูลประเภทสตริง โดยตรวจสอบเมื่อมีการเก็บข้อมูลและการดึงข้อมูลสตริงออกจากกองซ้อน ดังตัวอย่างโปรแกรมในภาพที่ 4-31

```
function pusht(nm){
      strstkp++;
      if(strstkp > MAXSTK)
      seterror("string stack overflow");
      strstk[strstkp] = nm;
}
function popt(){
      if(strstkp <= 0)
      seterror("string stack underflow");
      return strstk[strstkp--];
}
```

ภาพที่ 4-31 โปรแกรมตรวจสอบขนาดของกองซ้อนที่เก็บข้อมูลประเภทสตริง

ฟังก์ชัน pusht คือ การเก็บข้อมูลสตริงลงในกองซ้อน ถ้าข้อมูลที่เก็บมีจำนวนมากเกินขนาดของกองซ้อน จะรายงานความผิดพลาดโดยแสดงข้อความ "string stack overflow" ฟังก์ชัน popt คือ การดึงข้อมูลสตริงออกจากกองซ้อน ถ้าข้อมูลที่ดึงจากกองซ้อนเป็นการดึงข้อมูลที่น้อยเกินเก็บ จะรายงานความผิดพลาดโดยแสดงข้อความ "string stack underflow"

นอกจากนี้ยังมีการตรวจสอบขนาดของกองซ้อนที่เก็บข้อมูลประเภทอื่นๆ ได้แก่ ข้อมูลจากตัววิเคราะห์เชิงวากยสัมพันธ์ (parser) และข้อมูลตัวดำเนินการ (operator)

ขั้นตอนตัววิเคราะห์ความหมาย มีการตรวจสอบชนิดของข้อมูลว่าถูกต้องตามไวยากรณ์ของภาษาหรือไม่ ถ้าชนิดข้อมูลไม่ถูกต้องจะแสดงข้อความ "identifier type mismatch" ดังโปรแกรมภาพที่ 4-32

```
function chktype(type1, type2) {
     if( type1 != type2 )
     seterror("identifier type mismatch");
}
```

ภาพที่ 4-32 โปรแกรมตรวจสอบชนิดของข้อมูลให้ถูกต้องตามไวยากรณ์ภาษา

การตรวจสอบการประกาศตัวแปรส่วนกลางซ้ำ ดังโปรแกรมภาพที่ 4-33

```
function putSym(name, type, ref, arg){
     var a, found;
     a =  installGlobal(name,type,ref,arg,found);
     if(found) seterror("redefine global name");
     return a;
}
```

ภาพที่ 4-33 โปรแกรมตรวจสอบการประกาศตัวแปรส่วนกลางซ้ำ

จากภาพที่ 4-33 ถ้าเงื่อนไขของ if เป็นจริง นั่นหมายความว่ามีการพบว่าตัวแปรส่วนรวมชื่อนี้ได้ถูกประกาศและเก็บไว้ในตารางสัญลักษณ์แล้ว ดังนั้นจึงแสดงข้อความ "redefine global name" นอกจากนี้มีการตรวจสอบการเขียนพารามิเตอร์ และตรวจสอบการใช้งานตารางสัญลักษณ์เพื่อไม่ให้มีข้อมูลมากเกินขนาดของตารางสัญลักษณ์

ขั้นตอนตัวก่อกำเนิดรหัส จะมีการตรวจสอบอะตอมหรือโหนดในต้นไม้วิเคราะห์กระจายว่าเป็นประเภทที่ถูกต้องหรือไม่ ดังบางส่วนของโปรแกรมในภาพที่ 4-34

```
if(car(a) != OPER)
     seterror("genex: expect operator");
```

ภาพที่ 4-34 โปรแกรมตรวจสอบอะตอมในต้นไม้วิเคราะห์กระจายให้เป็นประเภทที่ถูกต้อง

จากภาพที่ 4-34 อะตอมที่ถูกต้องควรเป็นประเภท OPER (ตัวดำเนินการ) ดังนั้นถ้าอะตอมที่รับมาไม่ใช่ จะรายงานความผิดพลาดโดยแสดงข้อความ “genex: expect operator”

ส่วนการตรวจสอบโทเค็นเพื่อผลิตรหัสให้ถูกต้องตามประเภทโทเค็น ถ้าโทเค็นที่รับมาไม่เข้าเงื่อนไขจะรายงานความผิดพลาดให้ทราบ โดยแสดงข้อความว่าไม่รู้จักโทเค็นนั้น

บทที่ 5
การทดสอบและผลลัพธ์

ในบทนี้นำเสนอโปรแกรมที่ใช้ในการทดลอบและผลลัพธ์ที่ได้จากการทดลอบซึ่งแสดงถึงประสิทธิภาพของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB โดยโปรแกรมทดสอบที่ใช้กับงานวิจัยนี้ ประกอบด้วยโปรแกรม 4 โปรแกรม ดังแสดงในตารางที่ 5-1

ตารางที่ 5-1 ตารางแสดงรายละเอียดโปรแกรมที่ใช้ในการทดสอบ

ชื่อโปรแกรม	รายละเอียดโปรแกรม
Sum	หาผลรวมของจำนวน 10 จำนวนในอาร์เรย์
Findmax	หาจำนวนที่มีค่ามากที่สุดในอาร์เรย์
Bubble	Bubble sort ข้อมูลจำนวน 10 ตัว ข้อมูลเริ่มต้นเรียงจากมากไปน้อย
Matrix Multiplication	หาผลลัพธ์การคูณกันของเมทริกซ์มิติ 4×4

5.1 รายละเอียดของโปรแกรมที่นำเข้า

ในส่วนนี้จะนำเสนอรายละเอียดของโปรแกรมที่ใช้ในการทดสอบประสิทธิภาพของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB โดยมีรายละเอียดดังนี้

- โปรแกรม Sum เป็นโปรแกรมหาผลรวมของจำนวนในอาร์เรย์ทั้งหมด 10 จำนวน โดยเริ่มจากกำหนดค่าของจำนวนในอาร์เรย์ 10 จำนวน จากนั้นจึงทำการหาผลรวมของจำนวนทั้งหมดในอาร์เรย์
- โปรแกรม Findmax เป็นโปรแกรมหาจำนวนที่มีค่ามากที่สุดในอาร์เรย์จากทั้งหมด 10 จำนวน โดยเริ่มจากกำหนดค่าของจำนวนในอาร์เรย์ 10 จำนวน จากนั้นจึงทำการหาจำนวนที่มีค่ามากที่สุดในอาร์เรย์
- โปรแกรม Bubble เป็นโปรแกรม bubble sort จำนวนทั้งหมด 10 จำนวน โดยเริ่มจากกำหนดค่าของจำนวนในอาร์เรย์ 10 จำนวนที่เรียงจากมากไปน้อย จากนั้นจึงทำการ sort จำนวนทั้งหมดให้เรียงจากน้อยไปมาก
- โปรแกรม Matrix Multiplication เป็นโปรแกรมหาผลลัพธ์การคูณกันของเมทริกซ์มิติ 4*4 โดยเริ่มจากกำหนดค่าของเมทริกซ์มิติ 4*4 จำนวน 2 เมทริกซ์ จากนั้นจึงนำเมทริกซ์มิติ 4*4 ทั้งสองเมทริกซ์มาคูณกันเพื่อหาผลลัพธ์

โค้ดต้นฉบับของโปรแกรมที่ใช้ในการทดสอบทั้ง 4 โปรแกรมระบุอยู่ในภาคผนวก ก

5.2 การประเมินผล

การประเมินผลในการวิจัยนี้จะเปรียบเทียบประสิทธิภาพของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB กับการประมวลผลแบบดั้งเดิมของตัวแปลภาษาบนระบบปฏิบัติการดอส โดยมีการประเมินความถูกต้องในการแปลภาษา และเปรียบเทียบความเร็วในการทำงาน

5.2.1 ความถูกต้องในการแปลภาษา

การประเมินความถูกต้องในการแปลภาษาในงานวิจัยนี้ใช้โปรแกรมในการประเมินทั้งหมด 4 โปรแกรม โดยรายละเอียดของโปรแกรมอยู่ในหัวข้อ 5.1และโค้ดต้นฉบับของโปรแกรมอยู่ในภาคผนวก ก การประเมินนี้จะคำนึงถึงเฉพาะความถูกต้องของผลลัพธ์ในกระบวนการแปลภาษาเท่านั้น

การทำงานของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการจะได้ผลลัพธ์เป็นภาษาแอสเซมบลี ซึ่งผลลัพธ์ที่ได้จะถูกนำมาผ่านกระบวนการแปลภาษาอีกครั้งโดยแอสเซมเบลอร์ซึ่งเป็นตัวแปลภาษาของภาษาแอสเซมบลีให้ได้ผลลัพธ์เป็นภาษาเครื่อง จากนั้นจึงนำภาษาเครื่องที่ได้ไปรันบน virtual machine เพื่อแสดงผลลัพธ์ของโปรแกรม โดยแอสเซมเบลอร์ที่ใช้จะเป็นแอสเซมเบลอร์ของตัวแปลภาษาบนระบบปฏิบัติการดอส และ virtual machine ที่ใช้สำหรับการรันภาษาเครื่อง คือ virtual machine ที่มีให้บริการบนเว็บ [9] โดยผลลัพธ์ที่ได้จากการทำงานของตัวแปลภาษาบนเว็บจะนำมาเปรียบเทียบกับผลลัพธ์ที่ได้จากการทำงานของตัวแปลภาษาบนระบบปฏิบัติการดอสเพื่อใช้ในการประเมินความถูกต้อง พร้อมทั้งวิเคราะห์ผลในการประเมินความถูกต้อง อันจะกล่าวเป็นลำดับถัดไปในหัวข้อที่ 5.3

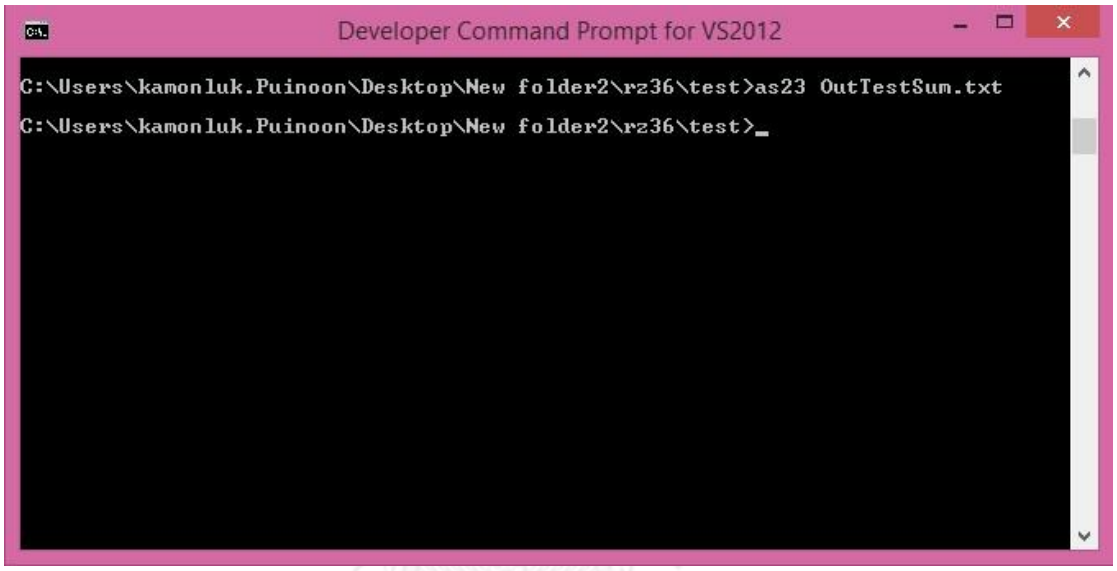
5.2.2 ความเร็วในการทำงาน

ความเร็วในการทำงานจะใช้เปรียบเทียบประสิทธิภาพของตัวแปลภาษาทั้งสองแบบ คือ ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ และตัวแปลภาษาที่ทำงานบนระบบปฏิบัติการดอส เมื่อการแปลภาษาจากทั้งสองแบบมีการแปลภาษาจากโปรแกรมเดียวกันและมีความถูกต้องในการแปลภาษา

การเปรียบเทียบความเร็วในการทำงานนี้ พิจารณาจากจำนวนโปรแกรมที่ใช้ ขั้นตอนการทำงานในกระบวนการของการได้ผลลัพธ์เป็นภาษาแอสเซมบลี และเวลาโดยเฉลี่ยที่ใช้ในกระบวนการแปลภาษาของตัวแปลภาษาทั้งสองแบบ อันจะกล่าวเป็นลำดับถัดไปในหัวข้อที่ 5.4

5.3 ผลการทดสอบและวิเคราะห์ผลในการประเมินผลความถูกต้องในการแปลภาษา

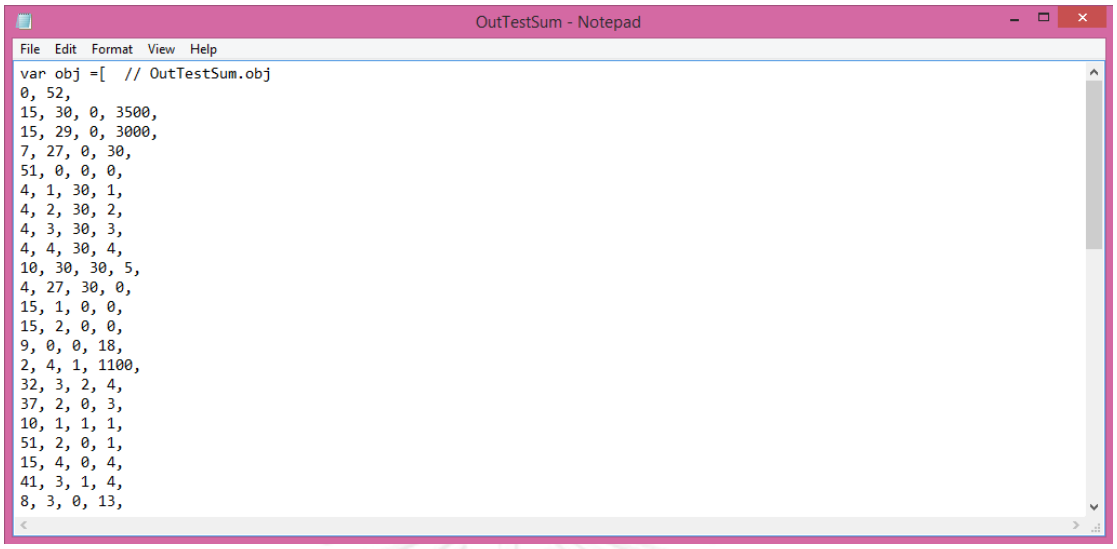
ผลลัพธ์ภาษาแอสเซมบลีของการประมวลผลโปรแกรม Sum ของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB จะถูกนำไปผ่านกระบวนการแปลภาษาโดยแอสเซมเบลอร์ของตัวแปลภาษาบนระบบปฏิบัติการดอสเพื่อให้ได้ผลลัพธ์ที่เป็นภาษาเครื่อง แสดงดังภาพที่ 5-1



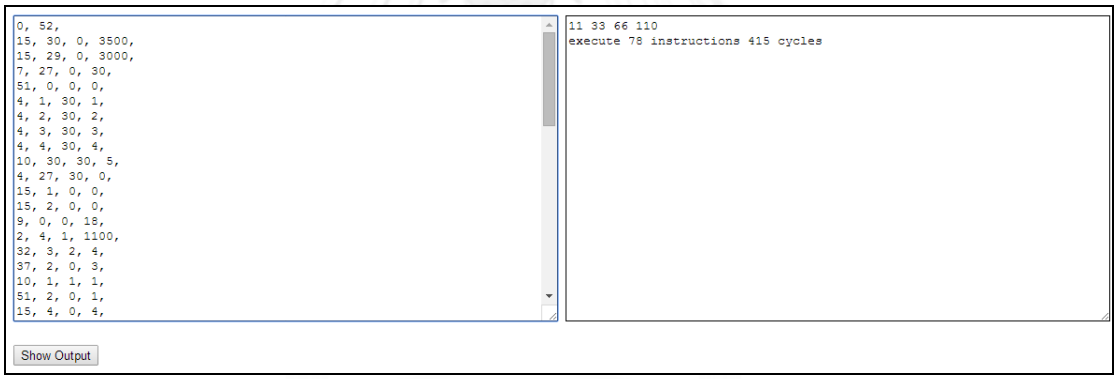
ภาพที่ 5-1 ภาพแสดงการใช้แอสเซมเบลอร์แปลภาษาบนระบบปฏิบัติการดอส

จากภาพที่ 5-1 as23 คือแอสเซมเบลอร์ของตัวแปลภาษาบนระบบปฏิบัติการดอส โดยแอสเซมเบลอร์จะทำการแปลภาษาแอสเซมบลีในไฟล์ OutTestSum.txt ซึ่งเป็นภาษาแอสเซมบลีที่ได้จากตัวแปลภาษาบนเว็บในระบบ RZ-WEB โดยแอสเซมเบลอร์จะทำการสร้างไฟล์ OutTestSum.obj ไว้ในไดเร็กทอรี่เดียวกับที่เก็บไฟล์ OutTestSum.txt เพื่อเก็บผลลัพธ์ที่ได้จากการแปลภาษาซึ่งเป็นภาษาเครื่องไว้ในไฟล์นั้น

เมื่อเปิดไฟล์ OutTestSum.obj จะได้ภาษาเครื่องของโปรแกรม Sum จากนั้นจึงคัดลอกภาษาเครื่องไปใส่ใน virtual machine ที่ให้บริการบนเว็บ และคลิกปุ่ม Show Output เพื่อทำการรันภาษาเครื่องบน virtual machine แสดงดังภาพที่ 5-2



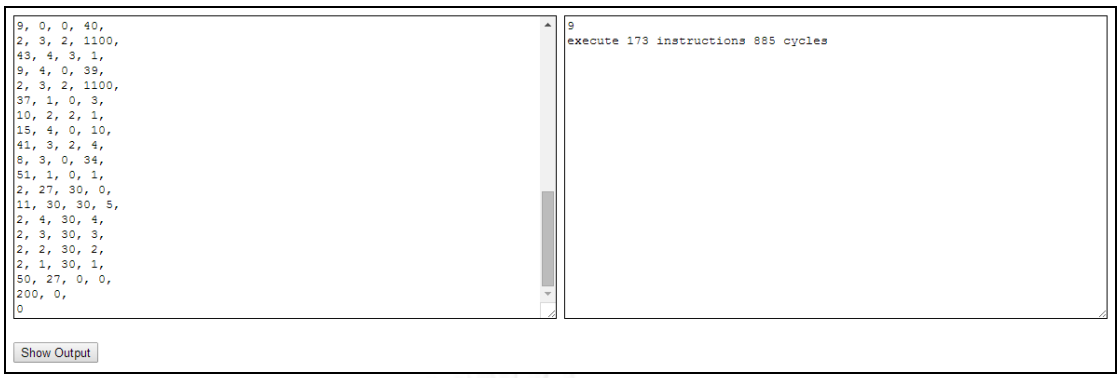
ภาพที่ 5-2 ผลลัพธ์การแปลภาษาของแอสเซมเบลอร์ที่เก็บไว้ในไฟล์ชนิด obj



ภาพที่ 5-3 การคัดลอกภาษาเครื่องจากไฟล์ชนิด obj มารันบน virtual machine ที่ให้บริการบนเว็บ

จากภาพที่ 5-3 ผลลัพธ์ของการรันภาษาเครื่องบน virtual machine ที่ให้บริการบนเว็บ ให้ผลลัพธ์การทำงานของโปรแกรม Sum ที่ถูกต้อง ดังนั้นตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการสามารถทำงานได้อย่างถูกต้องในโปรแกรม Sum

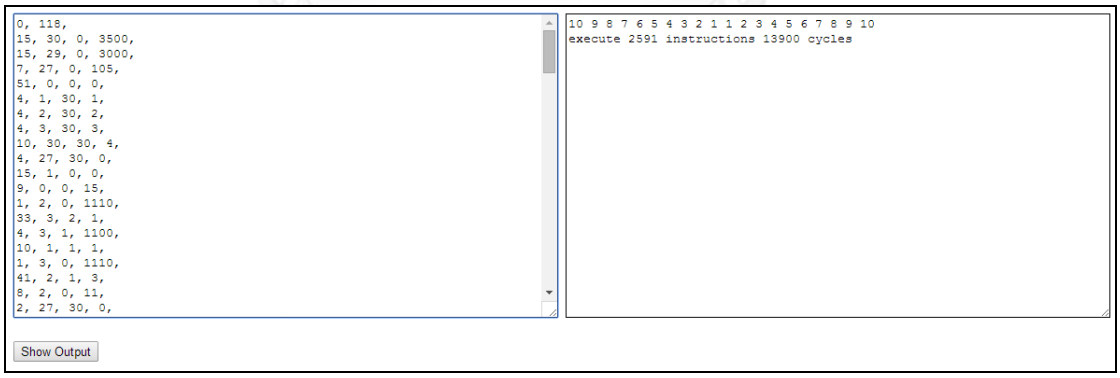
ผลลัพธ์ภาษาแอสเซมบลีของการประมวลผลโปรแกรม Findmax ของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB จะถูกนำไปผ่านกระบวนการแปลภาษาโดยแอสเซมเบลอร์ของตัวแปลภาษาบนระบบปฏิบัติการดอส จากนั้นจึงคัดลอกภาษาเครื่องที่ได้ไปใส่ใน virtual machine ที่ให้บริการบนเว็บ และคลิกปุ่ม Show Output เพื่อทำการรันภาษาเครื่องบน virtual machine ผลลัพธ์ของโปรแกรม Findmax ที่ได้ แสดงดังภาพที่ 5-4



ภาพที่ 5-4 การคัดลอกภาษาเครื่องจากไฟล์ชนิด obj มารันบน virtual machine ที่ให้บริการบนเว็บ

จากภาพที่ 5-4 ผลลัพธ์ของการรันภาษาเครื่องบน virtual machine ที่ให้บริการบนเว็บ ให้ผลลัพธ์การทำงานของโปรแกรม Findmax ที่ถูกต้อง ดังนั้นตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการสามารถทำงานได้อย่างถูกต้องในโปรแกรม Findmax

ผลลัพธ์ภาษาแอสเซมบลีของการประมวลผลโปรแกรม Bubble ของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB จะถูกนำไปผ่านกระบวนการแปลภาษาโดยแอสเซมเบลอร์ของตัวแปลภาษาบนระบบปฏิบัติการดอส จากนั้นจึงคัดลอกภาษาเครื่องที่ได้ไปใส่ใน virtual machine ที่ให้บริการบนเว็บ และคลิกปุ่ม Show Output เพื่อทำการรันภาษาเครื่องบน virtual machine ผลลัพธ์ของโปรแกรม Bubble ที่ได้ แสดงดังภาพที่ 5-5



ภาพที่ 5-5 การคัดลอกภาษาเครื่องจากไฟล์ชนิด obj มารันบน virtual machine ที่ให้บริการบนเว็บ

จากภาพที่ 5-5 ผลลัพธ์ของการรันภาษาเครื่องบน virtual machine ที่ให้บริการบนเว็บ ให้ผลลัพธ์การทำงานของโปรแกรม Bubble ที่ถูกต้อง ดังนั้นตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการสามารถทำงานได้อย่างถูกต้องในโปรแกรม Bubble

ผลลัพธ์ภาษาแอสเซมบลีของการประมวลผลโปรแกรม Matrix multiplication ของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB จะถูกนำไปผ่านกระบวนการแปลภาษาโดยแอสเซมเบลอร์ของตัวแปลภาษาบนระบบปฏิบัติการดอส จากนั้นจึงคัดลอกภาษาเครื่องที่ได้ไปใส่ใน virtual machine ที่ให้บริการบนเว็บ และคลิกปุ่ม Show Output เพื่อทำการรันภาษาเครื่องบน virtual machine ผลลัพธ์ของโปรแกรม Matrix multiplication ที่ได้ แสดงดังภาพที่ 5-6

ภาพที่ 5-6 การคัดลอกภาษาเครื่องจากไฟล์ชนิด obj มารันบน virtual machine ที่ให้บริการบนเว็บ

จากภาพที่ 5-6 ผลลัพธ์ของการรันภาษาเครื่องบน virtual machine ที่ให้บริการบนเว็บ ให้ผลลัพธ์การทำงานของโปรแกรม Matrix multiplication ที่ถูกต้อง ดังนั้นตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการสามารถทำงานได้อย่างถูกต้องในโปรแกรม Matrix multiplication

<u>วิเคราะห์ผลการประเมินความถูกต้องของการแปลภาษา</u>

จากการทดสอบสามารถวิเคราะห์ผลการประเมินความถูกต้องของการแปลภาษาได้ดังต่อ ไปนี้ สำหรับโปรแกรม Matrix multiplication, โปรแกรม Sum, โปรแกรม Findmax และโปรแกรม Bubble เมื่อผ่านกระบวนการแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ และนำผลลัพธ์ภาษาแอสเซมบลีที่ได้ไปผ่านกระบวนการแปลภาษาโดยแอสเซมเบลอร์ของตัวแปลภาษาบนระบบปฏิบัติการดอส จากนั้นจึงนำภาษาเครื่องที่ได้ไปรันบน virtual machine ที่ให้บริการบนเว็บ ปรากฏว่าผลลัพธ์การทำงานของโปรแกรมถูกต้อง จึงสรุปได้ว่าตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB สามารถทำงานได้อย่างถูกต้อง

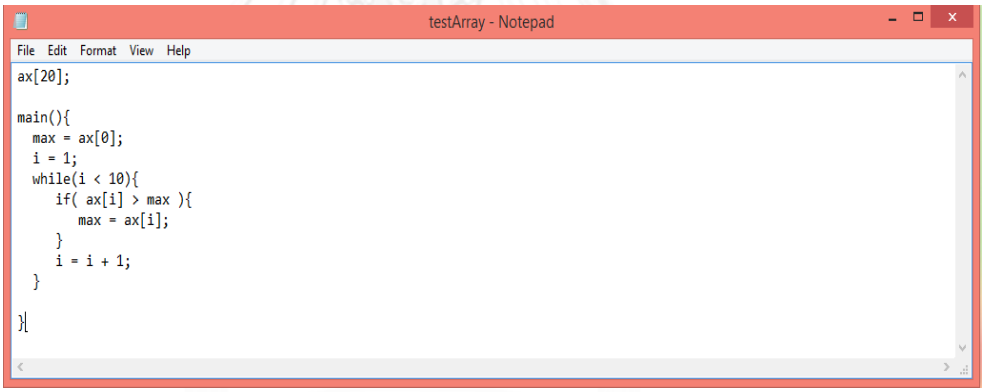
5.4 ผลการทดสอบและวิเคราะห์ผลในการประเมินความเร็วในการแปลภาษา

จำนวนโปรแกรมที่ใช้

เวลาที่ใช้ในการทำงานจะมากหรือน้อยอาจขึ้นอยู่กับจำนวนโปรแกรมที่ใช้ นั่นคือถ้าจำนวนโปรแกรมที่ต้องใช้งานมีมาก เวลาที่ใช้ในการทำงานจะมากตามไปด้วย เนื่องจากต้องใช้เวลาในการทำงานของแต่ละโปรแกรมเพิ่มมากขึ้นกว่าการทำงานเพียงโปรแกรมเดียว โดยที่ผลลัพธ์ของการทำงานทั้งสองแบบเหมือนกัน

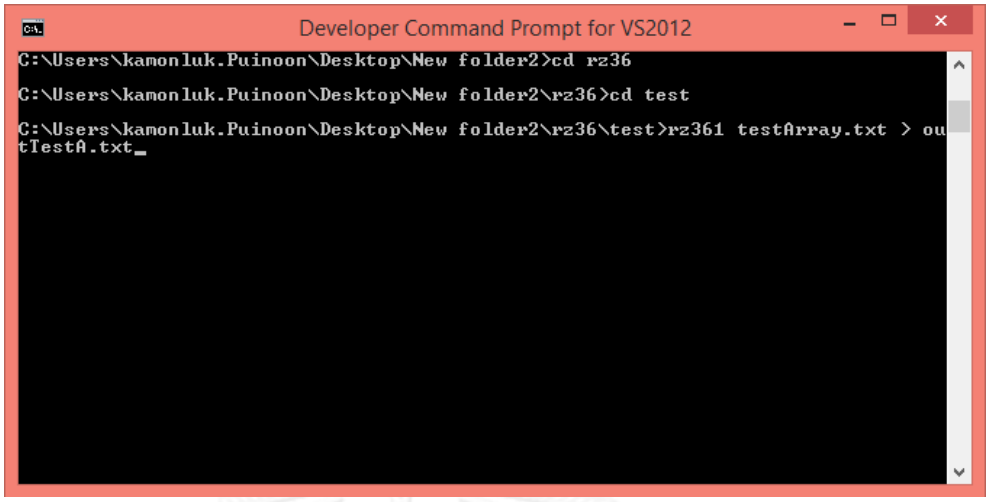
การทำงานของตัวแปลภาษาบนระบบปฏิบัติการดอส เพื่อให้ได้ผลลัพธ์ของการแปลภาษาเป็นภาษาแอสเซมบลีต้องใช้โปรแกรม Text Editor ซึ่งเป็นโปรแกรมสำหรับเขียนโค้ดต้นฉบับภาษาอาร์แซด และโปรแกรม Command Prompt เพื่อเปิดใช้งานระบบปฏิบัติการดอส โดยสามารถอธิบายรายละเอียดได้ดังนี้

เขียนโปรแกรมภาษาอาร์แซดลงในโปรแกรม Text Editor เช่น โปรแกรม NotePad แสดงดังภาพที่ 5-1



ภาพที่ 5-7 ภาพแสดงการเขียนโปรแกรมภาษาอาร์แซดลงในโปรแกรม Text Editor

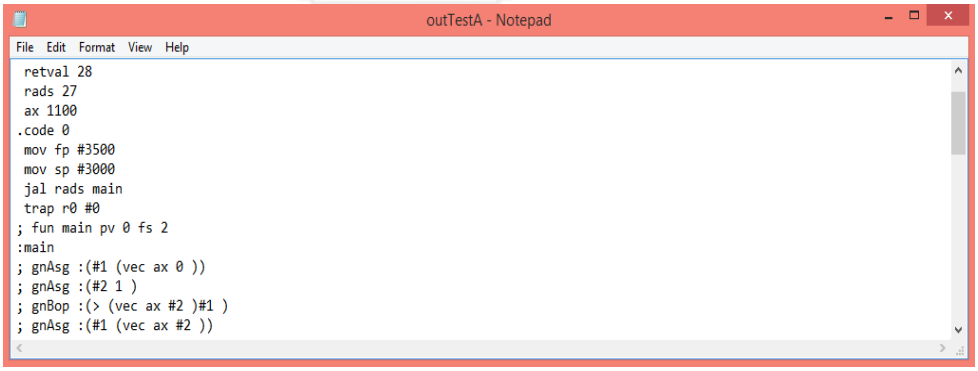
ทำการบันทึกโปรแกรมลงในไดเร็กทอรีเดียวกับตัวแปลภาษา แล้วเปิดโปรแกรม Command Prompt และพิมพ์ไดเร็กทอรีของตัวแปลภาษา จากนั้นพิมพ์ชื่อตัวแปลภาษาแล้วตามด้วยชื่อไฟล์ของโปรแกรม.txt ตามด้วยสัญลักษณ์ > และพิมพ์ชื่อไฟล์ที่ต้องการเก็บผลลัพธ์จากการแปลภาษา.txt ซึ่งจะทำให้โปรแกรมจากภาพที่ 5-7 ถูกแปลเป็นภาษาแอสเซมบลี และถูกบันทึกลงในไฟล์ที่เก็บผลลัพธ์ แสดงดังภาพที่ 5-8



ภาพที่ 5-8 การใช้โปรแกรม command prompt ในการแปลภาษา

จากภาพที่ 5-8 โปรแกรมต้นฉบับคือไฟล์ testArray.txt และ โปรแกรมผลลัพธ์คือไฟล์ outTestA.txt โดยตัวแปลภาษาชื่อ rz361

เปิดไฟล์ outTestA.txt เพื่อดูผลลัพธ์ที่เป็นภาษาแอสเซมบลี แสดงรูปบางส่วนของผลลัพธ์ที่เป็นภาษาแอสเซมบลีดังภาพที่ 5-9



ภาพที่ 5-9 ผลลัพธ์ภาษาแอสเซมบลีบนไฟล์

การทำงานของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ เพื่อให้ได้ผลลัพธ์ของการแปลภาษาเป็นภาษาแอสเซมบลีต้องใช้โปรแกรมเว็บเบราว์เซอร์ เช่น Mozilla Firefox, Opera, Safari, Google Chrome เป็นต้น สำหรับเขียนโค้ดต้นฉบับภาษาอาร์แซด โดยโปรแกรมต้นฉบับจะผ่านกระบวนการแปลภาษาโดยตัวแปลภาษาที่ฝังอยู่บนเว็บเบราว์เซอร์ในทันทีที่ผู้เขียนโปรแกรมคลิกปุ่ม Compile และแสดงผลลัพธ์ของการแปลภาษาเป็นภาษาแอสเซมบลีบนหน้าเว็บ รูปภาพแสดงขั้นตอนการทำงานของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการแสดงในภาคผนวก ข

ขั้นตอนการทำงานในกระบวนการของการได้ผลลัพธ์เป็นภาษาแอสเซมบลี

ขั้นตอนการทำงานของตัวแปลภาษาบนระบบปฏิบัติการดอส ค่อนข้างซับซ้อนกว่าขั้นตอนการทำงานของตัวแปลภาษาในงานวิจัยนี้ เนื่องจากจำนวนโปรแกรมที่ใช้งานมีมากกว่า และการทำงานด้วย Command Prompt ค่อนข้างซับซ้อน เนื่องจากผู้ใช้จำเป็นต้องจำคำสั่งสำหรับการทำงานจึงทำให้การทำงานไม่ค่อยสะดวก ดังแสดงในภาพที่ 5-7 ถึง ภาพที่ 5-9 ซึ่งนั่นเป็นเหตุผลที่ทำให้การทำงานของตัวแปลภาษาบนเว็บมีความสะดวกกว่าในส่วนของการติดต่อผู้ใช้ นอกจากนี้ตัวแปลภาษาที่ฝังอยู่บนเว็บเบราว์เซอร์ที่สามารถทำงานได้ในทันที ทำให้ผู้เขียนโปรแกรมไม่ต้องเสียเวลาในการบันทึกผลลัพธ์ภาษาแอสเซมบลีลงไฟล์ และเปิดไฟล์นั้นเพื่อดูผลลัพธ์

เนื่องจากจำนวนโปรแกรมที่ใช้ในงานวิจัยนี้มีจำนวนน้อยกว่า และขั้นตอนการทำงานในกระบวนการแปลภาษาเป็นภาษาแอสเซมบลีซับซ้อนน้อยกว่า เมื่อเปรียบเทียบกับจำนวนโปรแกรมและขั้นตอนการทำงานของตัวแปลภาษาดั้งเดิม ดังนั้นประสิทธิภาพทางด้านเวลาที่ใช้ในการทำงานในกระบวนการของการได้ผลลัพธ์เป็นภาษาแอสเซมบลีของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการจึงดีกว่าตัวแปลภาษาที่ทำงานบนระบบปฏิบัติการดอส เมื่อการแปลภาษาจากทั้งสองแบบมีการแปลภาษาจากโปรแกรมเดียวกันและมีความถูกต้องในการแปลภาษา

เวลาโดยเฉลี่ยที่ใช้ในกระบวนการแปลภาษาของตัวแปลภาษาทั้งสองแบบ

ในส่วนนี้จะกล่าวถึงเวลาโดยเฉลี่ยที่ใช้ในกระบวนการแปลภาษาของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ และตัวแปลภาษาที่ทำงานบนระบบปฏิบัติการดอส ซึ่งเริ่มตั้งแต่ขั้นตอนตัววิเคราะห์ศัพท์จนเสร็จสิ้นที่ขั้นตอนตัวก่อกำเนิดรหัสและได้ผลลัพธ์เป็นภาษาแอสเซมบลี แสดงดังตารางที่ 5-2 โดยแมชชีน (machine) ที่ใช้สำหรับการรันโปรแกรมคือ CPU Intel Core i5-3317U (1.70 GHz, 3 MB L3 Cache, up to 2.60 GHz) และ Chipset คือ Mobile Intel HM76 Express Chipset

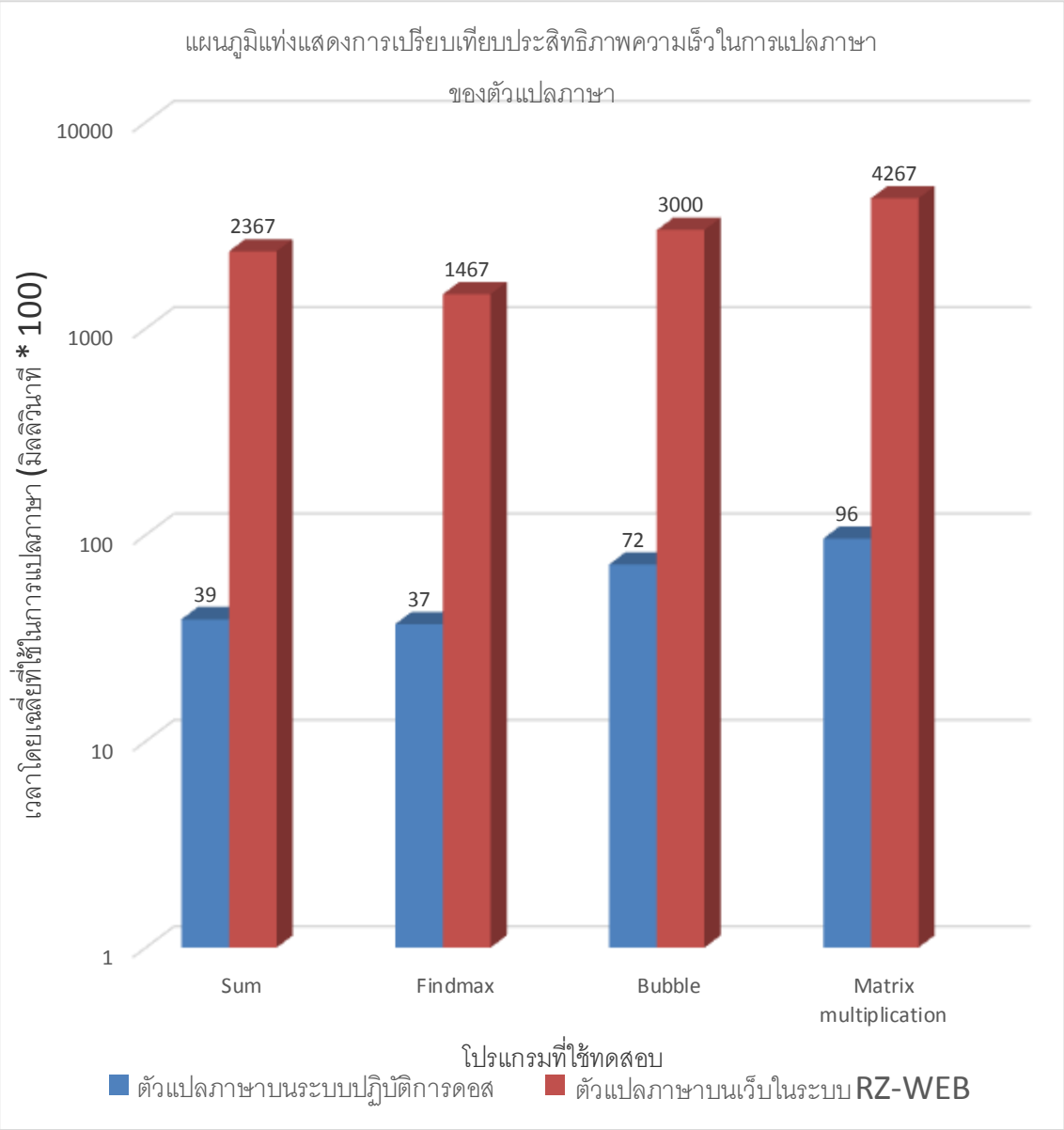
ตารางที่ 5-2 ตารางแสดงการเปรียบเทียบเวลาโดยเฉลี่ยที่ใช้ในกระบวนการแปลภาษาของตัวแปลภาษาบนระบบปฏิบัติการดอสกับตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ

โปรแกรมที่ใช้ทดสอบ	**เวลาโดยเฉลี่ยที่ใช้ในกระบวนการแปลภาษา (มิลลิวินาที)**	
	ตัวแปลภาษาบนระบบปฏิบัติการดอส	**ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการ**
Sum	0.39	23.67
Findmax	0.37	14.67
Bubble	0.72	30
Matrix multiplication	0.96	42.67

จากตารางที่ 5-2 เป็นการเปรียบเทียบประสิทธิภาพความเร็วของตัวแปลภาษาระหว่างตัวแปลภาษาแบบดั้งเดิมบนระบบปฏิบัติการดอส และตัวแปลภาษาบนเว็บในระบบ RZ-WEB โดยจะวิเคราะห์ประสิทธิภาพความเร็วของตัวแปลภาษาทั้งสองแบบในแต่ละโปรแกรมที่ใช้ทดสอบ ดังนี้

- โปรแกรม Sum เมื่อถูกแปลภาษาด้วยตัวแปลภาษาบนระบบปฏิบัติการดอสใช้เวลาโดยเฉลี่ยในการแปลภาษาทั้งสิ้น 0.39 มิลลิวินาที ขณะที่ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการใช้เวลาโดยเฉลี่ยถึง 23.67 มิลลิวินาที เมื่อนำมาเปรียบเทียบพบว่าตัวแปลภาษาบนระบบปฏิบัติการดอสใช้เวลาในการแปลภาษาเร็วกว่าตัวแปลภาษาบนเว็บประมาณ 60.69 เท่า
- โปรแกรม Findmax เมื่อถูกแปลภาษาด้วยตัวแปลภาษาบนระบบปฏิบัติการดอสใช้เวลาโดยเฉลี่ยในการแปลภาษาทั้งสิ้น 0.37 มิลลิวินาที ขณะที่ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการใช้เวลาโดยเฉลี่ยถึง 14.67 มิลลิวินาที เมื่อนำมาเปรียบเทียบพบว่าตัวแปลภาษาบนระบบปฏิบัติการดอสใช้เวลาในการแปลภาษาเร็วกว่าตัวแปลภาษาบนเว็บประมาณ 39.65 เท่า

- โปรแกรม Bubble เมื่อถูกแปลภาษาด้วยตัวแปลภาษาบนระบบปฏิบัติการดอสใช้เวลาโดยเฉลี่ยในการแปลภาษาทั้งสิ้น 0.72 มิลลิวินาที ขณะที่ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการใช้เวลาโดยเฉลี่ยถึง 30 มิลลิวินาที เมื่อนำมาเปรียบเทียบพบว่าตัวแปลภาษาบนระบบปฏิบัติการดอสใช้เวลาในการแปลภาษาเร็วกว่าตัวแปลภาษาบนเว็บประมาณ 41.67 เท่า
- โปรแกรม Matrix multiplication เมื่อถูกแปลภาษาด้วยตัวแปลภาษาบนระบบปฏิบัติการดอสใช้เวลาโดยเฉลี่ยในการแปลภาษาทั้งสิ้น 0.96 มิลลิวินาที ขณะที่ตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการใช้เวลาโดยเฉลี่ยถึง 42.67 มิลลิวินาที เมื่อนำมาเปรียบเทียบพบว่าตัวแปลภาษาบนระบบปฏิบัติการดอสใช้เวลาในการแปลภาษาเร็วกว่าตัวแปลภาษาบนเว็บประมาณ 44.45 เท่า



ภาพที่ 5-10 แผนภูมิแท่งแสดงการเปรียบเทียบประสิทธิภาพความเร็วในการแปลภาษา
ของตัวแปลภาษา

<u>วิเคราะห์ผลการประเมินความเร็วในการทำงาน</u>

จากการวิเคราะห์ประสิทธิภาพความเร็วในการแปลภาษาของตัวแปลภาษาทั้งสองแบบพบว่า ตัวแปลภาษาบนระบบปฏิบัติการดอสใช้เวลาโดยเฉลี่ยในกระบวนการแปลภาษาน้อยกว่าตัวแปลภาษาบนเว็บ เนื่องจากตัวแปลภาษาบนระบบปฏิบัติการดอสพัฒนาโดยภาษาซี ซึ่งตัวแปลภาษาของภาษาซีทำงานอยู่บนระบบปฏิบัติการดอสได้อย่างรวดเร็วเนื่องจากกระบวนการแปลภาษาจะทำการอ่านโปรแกรมภาษาซีทั้งหมดตั้งแต่ต้นจนจบ แล้วทำการแปลผลรอบเดียว การแปลผลลักษณะนี้เรียกว่าคอมไพล์ (compile) ในขณะที่ตัวแปลภาษาบนเว็บในระบบ RZ-WEB ที่พัฒนาโดยภาษาจาวาสคริปต์ ซึ่งตัวแปลภาษาของภาษาจาวาสคริปต์ที่ทำงานอยู่บนเว็บเบราว์เซอร์ทำงานได้ช้ากว่าตัวแปลภาษาของภาษาซีบนระบบปฏิบัติการดอส เนื่องจากตัวแปลภาษาของภาษาจาวาสคริปต์จะทำการอ่านและแปลโปรแกรมทีละบรรทัด เมื่อแปลผลบรรทัดหนึ่งเสร็จก็จะทำงานตามคำสั่งในบรรทัดนั้น แล้วจึงทำการแปลผลตามคำสั่งในบรรทัดถัดไป การแปลผลลักษณะนี้เรียกว่าอินเตอร์เพรต (interpret) การทำงานของตัวแปลภาษาของภาษาซีและภาษาจาวาสคริปต์ที่แตกต่างกันดังกล่าวเป็นสาเหตุสำคัญที่ทำให้ตัวแปลภาษาบนเว็บในระบบ RZ-WEB ทำงานช้ากว่าตัวแปลภาษาบนระบบปฏิบัติการดอส อย่างไรก็ตามเมื่อวิเคราะห์ประสิทธิภาพความเร็วในการทำงาน เนื่องจากจำนวนโปรแกรมที่ใช้สำหรับตัวแปลภาษาบนเว็บมีจำนวนน้อยกว่า และขั้นตอนการทำงานในกระบวนการแปลภาษาเป็นภาษาแอสเซมบลีซับซ้อนน้อยกว่า เมื่อเปรียบเทียบกับจำนวนโปรแกรมและขั้นตอนการทำงานของตัวแปลภาษาดั้งเดิม ดังนั้นประสิทธิภาพด้านเวลาที่ใช้ในการทำงานในกระบวนการได้ผลลัพธ์เป็นภาษาแอสเซมบลีของตัวแปลภาษาบนเว็บในระบบ RZ-WEB จึงดีกว่าตัวแปลภาษาที่ทำงานบนระบบปฏิบัติการดอส เมื่อการแปลภาษาจากทั้งสองแบบมีการแปลภาษาจากโปรแกรมเดียวกันและมีความถูกต้องในการแปลภาษา

บทที่ 6
สรุปผลการวิจัยและแนวทางการวิจัยในอนาคต

ในบทนี้จะกล่าวถึงการสรุปผลการวิจัย รวมทั้งข้อเสนอแนะเพื่อจะเป็นแนวทางในการพัฒนางานวิจัยต่อไป

6.1 สรุปผลการวิจัย

การเขียนโปรแกรมบนเว็บถือเป็นทางเลือกใหม่สำหรับผู้เขียนโปรแกรมที่กำลังได้รับความนิยมในปัจจุบัน เนื่องจากผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้สะดวกยิ่งขึ้นเมื่อเปรียบเทียบกับการเขียนโปรแกรมแบบดั้งเดิม ทั้งในแง่ของอุปกรณ์และโปรแกรมที่ใช้ในการเขียนโปรแกรม รวมไปถึงการทดสอบโปรแกรมที่สามารถทำได้อย่างรวดเร็วและได้ผลลัพธ์เหมือนกับการเขียนโปรแกรมบนคอมพิวเตอร์ อันเป็นสาเหตุสำคัญที่ทำให้การเขียนโปรแกรมบนเว็บได้กลายมาเป็นคู่แข่งที่สำคัญของการเขียนโปรแกรมบนคอมพิวเตอร์ในแง่ของคุณภาพและความสะดวกในการใช้งาน

งานวิจัยจำนวนมากมีการพัฒนาแนวคิดและเครื่องมือสำหรับการเขียนโปรแกรมบนเว็บแบบออนไลน์โดยมีกระบวนการแปลภาษาจากภาษาระดับสูงไปเป็นภาษาเครื่องที่ฝั่งเครื่องผู้ให้บริการ (server) และนำผลลัพธ์ของโปรแกรมกลับมาแสดงยังฝั่งเครื่องผู้รับบริการ (client) ผ่านทางหน้าเว็บ อย่างไรก็ตามการใช้บริการเครื่องผู้ให้บริการจำเป็นต้องมีผู้เชี่ยวชาญทางด้านคอมพิวเตอร์ในการดูแลระบบ และการประมวลผลโปรแกรมที่ฝั่งเครื่องผู้ให้บริการอาจทำให้ระบบมีความเป็นส่วนตัว (privacy) และความปลอดภัย (security) ในการทำงานที่ต่ำ เนื่องจากโค้ดต้นฉบับอาจถูกคัดลอกระหว่างการส่งไปประมวลผลยังฝั่งเครื่องผู้ให้บริการและผลลัพธ์โปรแกรมที่ส่งกลับมายังเครื่องผู้รับบริการอาจไม่ใช่ผลลัพธ์ของโค้ดเดิม หรืออาจมีการแฝงสคริปต์กลับมาโจมตีเครื่องผู้รับบริการอันก่อให้เกิดปัญหาในการใช้งานตามมา

ในงานวิจัยนี้มุ่งเน้นการพัฒนาแนวคิดและเครื่องมือสำหรับการเขียนโปรแกรมบนเว็บโดยไม่อาศัยเครื่องผู้ให้บริการ เพื่อลดปัญหาความซับซ้อนในการดูแลระบบ รวมถึงทำให้ระบบมีความเป็นส่วนตัวและความปลอดภัยในการทำงานที่สูง ระบบที่พัฒนาในงานวิจัยนี้คือ ระบบ RZ-WEB เป็นระบบที่นำภาษาอาร์แซดมาเป็นภาษาต้นแบบในการประยุกต์ใช้สำหรับพัฒนาตัวแปลภาษาบนเว็บที่สามารถทำงานบนเครื่องผู้รับบริการได้ทันที อันจะเป็นทางเลือกใหม่สำหรับการเรียนรู้การเขียนโปรแกรมและการสร้างโปรเจกต์บนเว็บเบราว์เซอร์ ตัวแปลภาษาดังกล่าวจะใช้ภาษาจาวาสคริปต์ในการพัฒนา เนื่องจากเว็บเบราว์เซอร์ทั้งหมดมีจาวาสคริปต์เอนจินซึ่งเป็นตัวแปลภาษาที่ทำหน้าที่แปลภาษาจาวาสคริปต์และประมวลผลโดยทันที จึงทำให้ภาษาจาวาสคริปต์เป็นภาษามาตรฐานของทุกๆเว็บเบราว์เซอร์และสามารถรันได้กับทุกแพลตฟอร์มที่รองรับเว็บเบราว์เซอร์ ได้แก่ คอมพิวเตอร์ตั้งโต๊ะ โน๊ตบุ๊ค แท็บเล็ต และสมาร์ทโฟน อันทำให้ระบบ RZ-WEB สามารถทำงานได้กับอุปกรณ์ดังกล่าวโดยมีความเป็นส่วนตัวและความปลอดภัยในการเขียนโปรแกรมบนเว็บที่สูง

กระบวนการแปลภาษาของระบบ RZ-WEB ประกอบด้วยขั้นตอนหลัก 4 ขั้นตอน ได้แก่ ขั้นตอนตัววิเคราะห์ศัพท์ (lexical analyzer), ขั้นตอนตัววิเคราะห์เชิงวากยสัมพันธ์ (parser), ขั้นตอนตัววิเคราะห์ความหมาย (semantic analyzer) และขั้นตอนตัวก่อกำเนิดรหัส (code generator) นอกจากนี้ยังมีขั้นตอนที่เสริมการทำงานของขั้นตอนหลักอีก 2 ขั้นตอน คือ ตารางสัญลักษณ์ (symbol table) และรายงานความผิดพลาด (error handle) สำหรับการประเมินประสิทธิภาพของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB มีการประเมินความถูกต้องในการแปลภาษาและเปรียบเทียบความเร็วในการแปลภาษา โดยเปรียบเทียบกับการประมวลผลแบบดั้งเดิมของตัวแปลภาษาบนระบบปฏิบัติการดอส เมื่อการแปลภาษาจากทั้งสองแบบเป็นการแปลภาษาจากโปรแกรมเดียวกัน ซึ่งโปรแกรมที่ใช้ในการทดสอบมี 4 โปรแกรม ได้แก่ โปรแกรม Sum, โปรแกรม Findmax, โปรแกรม Bubble และโปรแกรม Matrix multiplication จากการทดสอบการทำงานของระบบ RZ-WEB ในโปรแกรมดังกล่าว เมื่อนำผลลัพธ์ภาษาแอสเซมบลีที่ได้ไปผ่านกระบวนการแปลภาษาโดยแอสเซมเบลอร์ของตัวแปลภาษาบนระบบปฏิบัติการดอส และนำภาษาเครื่องที่ได้ไปรันบน virtual machine ที่ให้บริการบนเว็บ ปรากฏว่าผลลัพธ์การทำงานของโปรแกรมถูกต้อง จึงสรุปได้ว่าตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการในระบบ RZ-WEB สามารถทำงานได้อย่างถูกต้อง และเมื่อวิเคราะห์ประสิทธิภาพความเร็วในการแปลภาษาของตัวแปลภาษาทั้งสองแบบพบว่าตัวแปลภาษาบนระบบปฏิบัติการดอสใช้เวลาโดยเฉลี่ยในกระบวนการแปลภาษาน้อยกว่าตัวแปลภาษาบนเว็บเนื่องจากการทำงานของตัวแปลภาษาของภาษาซีที่เร็วกว่าการทำงานของตัวแปลภาษาของภาษาจาวาสคริปต์ อย่างไรก็ตามเมื่อวิเคราะห์ประสิทธิภาพของเวลาที่ใช้ในการทำงานในกระบวนการแปลภาษาจนได้ผลลัพธ์เป็นภาษาแอสเซมบลี ระบบ RZ-WEB สามารถทำงานได้รวดเร็วกว่ากระบวนการแปลภาษาบนระบบปฏิบัติการดอส เนื่องจากจำนวนโปรแกรมที่ใช้และขั้นตอนการทำงานในกระบวนการของการได้ผลลัพธ์เป็นภาษาแอสเซมบลีมีจำนวนน้อยกว่า

6.2 แนวทางการวิจัยในอนาคต

งานวิจัยนี้ยังมีโทเค็นบางประเภทที่ยังไม่สามารถสร้างภาษาเป้าหมายที่เป็นภาษาแอสเซมบลีได้อย่างเต็มประสิทธิภาพ นั่นคือโทเค็น print ซึ่งเป็นคำสั่งสำหรับการแสดงผลลัพธ์ทางหน้าจอ เนื่องจากโทเค็น print ยังไม่สามารถรองรับการแสดงผลที่เป็นข้อความประเภทสตริงได้ โดยสามารถรองรับการแสดงผลเฉพาะตัวเลขเท่านั้น

ผลลัพธ์ของตัวแปลภาษาบนเว็บที่ทำงานบนเครื่องผู้รับบริการยังเป็นภาษาแอสเซมบลีเท่านั้น จึงยังไม่สามารถแสดงผลลัพธ์ของโปรแกรมออกมาได้ เพื่อให้สามารถแสดงผลลัพธ์ของโปรแกรมได้ สามารถทำได้โดยการเขียนตัวก่อกำเนิดรหัสใหม่ให้ได้ผลลัพธ์ออกมาเป็นภาษาเครื่อง และนำภาษาเครื่องที่ได้ไปรันบน Virtual Machine ที่ให้บริการบนเว็บเพื่อแสดงผลลัพธ์ของโปรแกรมออกมา

รายการอ้างอิง

1. Zakai, A. Emscripten: An LLVM-to-JavaScript compiler. in SPLASH'11 Compilation - Proceedings of OOPSLA'11, Onward! 2011, GPCE'11, DLS'11, and SPLASH'11 Companion. 2011.
2. Ansari, A.N., et al. Online C/C++ compiler using cloud computing. in Multimedia Technology (ICMT), 2011 International Conference on. 2011.
3. Kuan-Cheng, L. An On-line Instruction/Learning Environment for Supporting Individualized Learning in Java Programming. in Parallel Processing Workshops, 2007. ICPPW 2007. International Conference on. 2007.
4. Muchnick, S.S., Advanced compiler design implementation. 1997: Morgan Kaufmann.
5. Louden, K.C., Compiler construction. 1997: PWS Publ.
6. Hwang, W.Y., et al., A web-based programming learning environment to support cognitive development. Interacting with Computers, 2008. 20(6): p. 524-534.
7. More, A., J. Kumar, and V.G. Renumol. Web based programming assistance tool for novices. in Proceedings - IEEE International Conference on Technology for Education, T4E 2011. 2011.
8. Chongstitvatana, P. Rz language. 2010 2013, July 13; Available from: http://www.cp.eng.chula.ac.th/~piak/project/rz3/index-rz3.htm.
9. S2 version 3 simulator. 2014; Available from: http://www.cp.eng.chula.ac.th/~piak/project/s2/js/s2-sim.htm.

ภาคผนวก

ภาคผนวก ก
โค้ดต้นฉบับของโปรแกรมที่ใช้ในการทดสอบประสิทธิภาพของตัวแปลภาษา

ในส่วนนี้แสดงรายละเอียดเพิ่มเติมจากเนื้อหาในบทที่5 เกี่ยวกับโค้ดต้นฉบับของโปรแกรมที่ใช้ในการทดสอบทั้ง 4 โปรแกรม ได้แก่ โปรแกรม Sum, โปรแกรม Findmax, โปรแกรม Bubble และโปรแกรม Matrix multiplication

โค้ดตันฉบับของโปรแกรม Sum ดังภาพที่ ก-1

```
ax[10];
sum(){
      i = 0;
      s = 0;
      while( i < 4 ){
            s = s + ax[i];
            i = i + 1;
            print(s);
      }
      return s;
}
main(){
      ax[0] = 11;
      ax[1] = 22;
      ax[2] = 33;
      ax[3] = 44;
      sum();
}
```

ภาพที่ ก-1 โค้ดต้นฉบับของโปรแกรม Sum

โค้ดต้นฉบับของโปรแกรม Findmax ดังภาพที่ ก-2

```
ax[10];
init(){
     i=0;
     while(i<10){
          ax[i] = i;
          i = i+1;
     }
}
main(){
     init();
     max = ax[0];
     i = 1;
     while(i < 10){
          if( ax[i] > max ){
               max = ax[i];
          }
          i = i + 1;
     }
     print(max);
}
```

ภาพที่ ก-2 โค้ดต้นฉบับของโปรแกรม Findmax

โค้ดต้นฉบับของโปรแกรม Bubble ดังภาพที่ ก-3 และ ก-4

```
data[10], maxdata;
init(){
  i = 0;
     while(i < maxdata){
          data[i] = maxdata - i;
          i = i+1;
     }
}
show(){
  i = 0;
     while(i < maxdata){
          print(data[i]);
          i = i+1;
     }
}
swap(a,b){
     t = data[a];
     data[a] = data[b];
     data[b] = t;
}
```

ภาพที่ ก-3 โค้ดต้นฉบับของโปรแกรม Bubble

```
sort(){
     i = 0;
     while(i < maxdata){
          j = 0;
          while(j < maxdata-1){
               if(data[j+1] < data[j])swap(j,j+1);
               j = j+1;
          }
          i = i+1;
     }
}
main(){
     maxdata = 10;
     init();
     show();
     sort();
     show();
}
```

ภาพที่ ก-4 โค้ดต้นฉบับของโปรแกรม Bubble (ต่อ)

โค้ดต้นฉบับของโปรแกรม Matrix multiplication ดังภาพที่ ก-5 และ ก-6

```
N, a[16], b[16], c[16];
inita(){
     i = 0;
     while( i < N ){
          j = 0;
          while( j < N ){
                    a[i*N+j] = i;
                    j = j + 1;
          }
          i = i + 1;
     }
}
initb(){
     i = 0;
     while( i < N ){
          j = 0;
          while( j < N ){
                    b[i*N+j] = j;
                    j = j + 1;
          }
          i = i + 1;
     }
}
```

ภาพที่ ก-5 โค้ดต้นฉบับของโปรแกรม Matrix multiplication

```
matmul(){
      i = 0;
      while( i < N ){
            j = 0;
            while( j < N ){
                  s = 0;
                  k = 0;
                  while( k < N ){
                        s = s + a[i*N+k] *
                        b[k*N+j];

                              k = k + 1;
                  }
                        c[i*N+j] = s;
                  j = j + 1;
            }
            i = i + 1;
      }
}
show(){
      i = 0;
            while( i < N ){
                  j = 0;
                  while( j < N ){

                        print(c[i*N+j]);

                        j = j + 1;
                  }
                  i = i + 1;
            }
}
main(){
      N = 4;
      inita();
      initb();
      matmul();
      show();
}
```

ภาพที่ ก-6 โค้ดต้นฉบับของโปรแกรม Matrix multiplication (ต่อ)

ภาคผนวก ข
รูปภาพแสดงขั้นตอนการทำงานของระบบ RZ-WEB

ในส่วนนี้แสดงรายละเอียดเพิ่มเติมจากเนื้อหาในบทที่5 เกี่ยวกับรูปภาพแสดงขั้นตอนการทำงานทั้งหมดของระบบ RZ-WEB โดยเริ่มตั้งแต่การเขียนโปรแกรมบนเว็บไปจนถึงผลลัพธ์ของการแปลภาษาของระบบออกมาเป็นภาษาแอสเซมบลี รวมถึงแสดงรูปภาพตัวอย่างการเขียนโปรแกรมที่ผิดและระบบทำการรายงานความผิดพลาดที่เกิดขึ้น

ภาพที่ ข-1 หน้าตาของเว็บอินเตอร์เฟซสำหรับการเขียนโปรแกรมบนเว็บ

ภาพที่ ข-2 เขียนโปรแกรมภาษาอาร์แซดในส่วนของ Source code

ภาพที่ ข-3 ผลลัพธ์ภาษาแอสเซมบลีที่ได้หลังจากคลิกปุ่ม Compile

ภาพที่ ข-4 ตัวอย่างการเขียนโปรแกรมที่ผิด โดยบรรทัดแรกไม่มีเครื่องหมาย “;”

ภาพที่ ข-5 การแสดงรายงานความผิดพลาดของโปรแกรมให้ผู้เขียนโปรแกรมทราบ

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวกมลลักษณ์ สุขเสน เกิดวันที่ 23 เมษายน พ.ศ. 2533 ที่จังหวัดลำปาง สำเร็จการศึกษาระดับปริญญาตรีวิศวกรรมศาสตร์บัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2555 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2556