บทที่ 3

โครงสร้างตัวแปลภาษาวอตฟัฟ

1. องค์ประกอบในการแปลโปรแกรมของตัวแปลภาษาวอตฟัฟ

เครื่องคอมพิวเตอร์ที่ติดตั้งที่ศูนย์คอมพิวเตอร์ต่างๆโดยเฉพาะในสถาบันการศึกษาต่างๆ เวลาส่วนหนึ่งได้ใช้ไปในงานที่เกี่ยวกับการพัฒนาโปรแกรม เช่นงานที่เกี่ยวกับการฝึกหัดเขียนโปรแกรมของนักเรียน และใช้ภาษาฟอร์แทรนเป็นภาษาหลัก ลักษณะโปรแกรมมักเป็นโปรแกรมเล็กๆ ซึ่งมีความผิดพลาดทางไวยากรณ์เป็นส่วนใหญ่ ดังนั้นโปรแกรมเหล่านี้จะสิ้นสุดการทำงานแค่การแปล ซึ่งตัวแปลภาษาที่มีอยู่ไม่สามารถทำการแปลได้เร็วเท่าที่ควร ถ้ามีตัวแปลภาษาที่สามารถแปลได้เร็วแล้วจะทำให้ประหยัดเวลาเครื่องลงได้อย่างมาก

จากปัญหาดังกล่าวจึงได้มีการพัฒนาตัวแปลภาษาฟอร์แทรนขึ้นมาใหม่ ที่เรียกว่า "วอตฟัฟ" เพื่อใช้กับงานการพัฒนาโปรแกรมโดยมีจุดประสงค์เพื่อให้

1) มีการอธิบายข้อผิดพลาดต่างๆที่ชัดเจนทั้งช่วงการแปล (compile) และช่วงทำงานจริง (execution)

2) สามารถแปลได้เร็ว

3) ทำงานในลักษณะประมวลผลแบบกลุ่ม (Batch Processing)

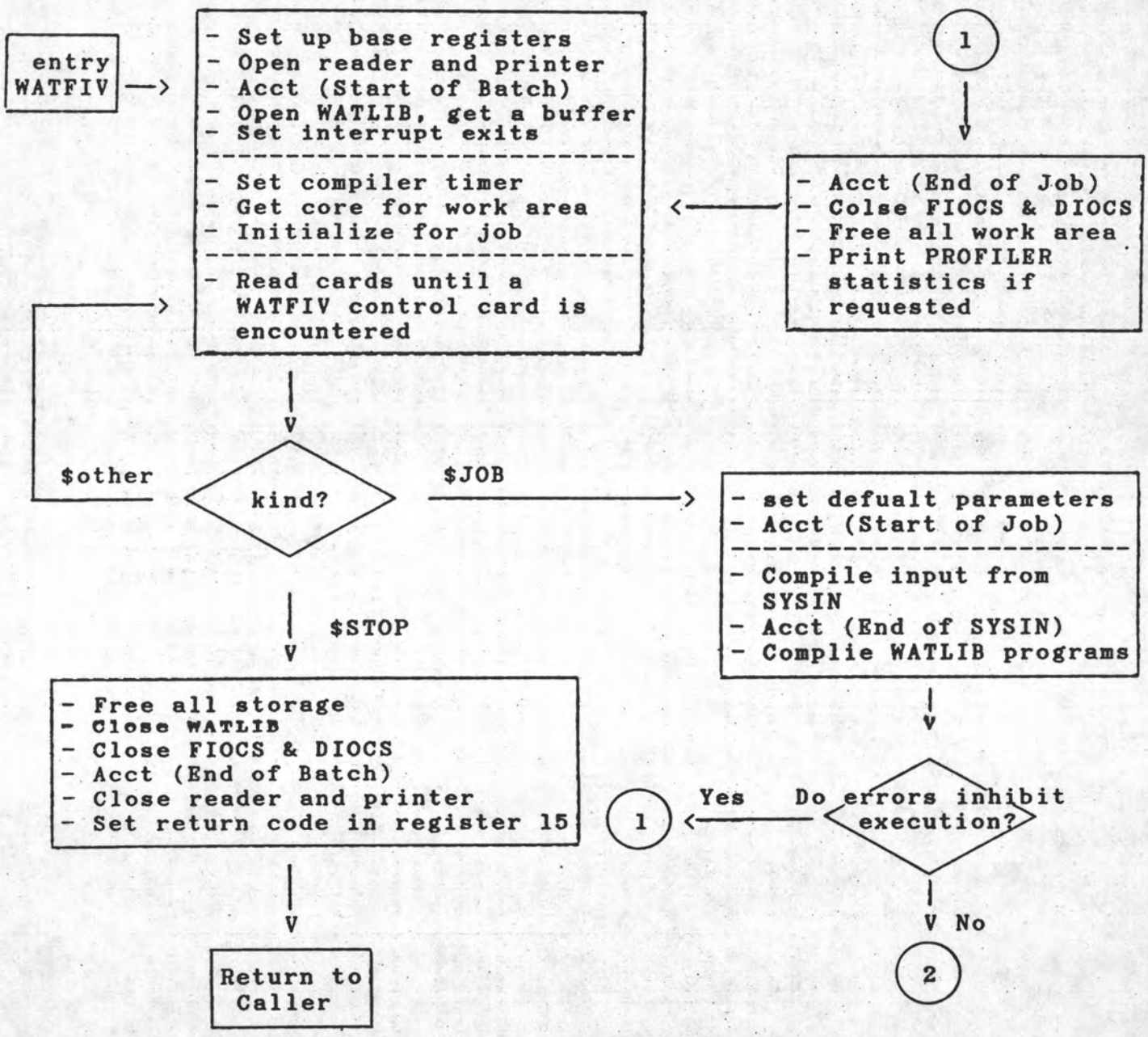
4) สามารถรับโปรแกรมภาษาฟอร์แทรนที่ใช้กับเครื่อง IBM ได้

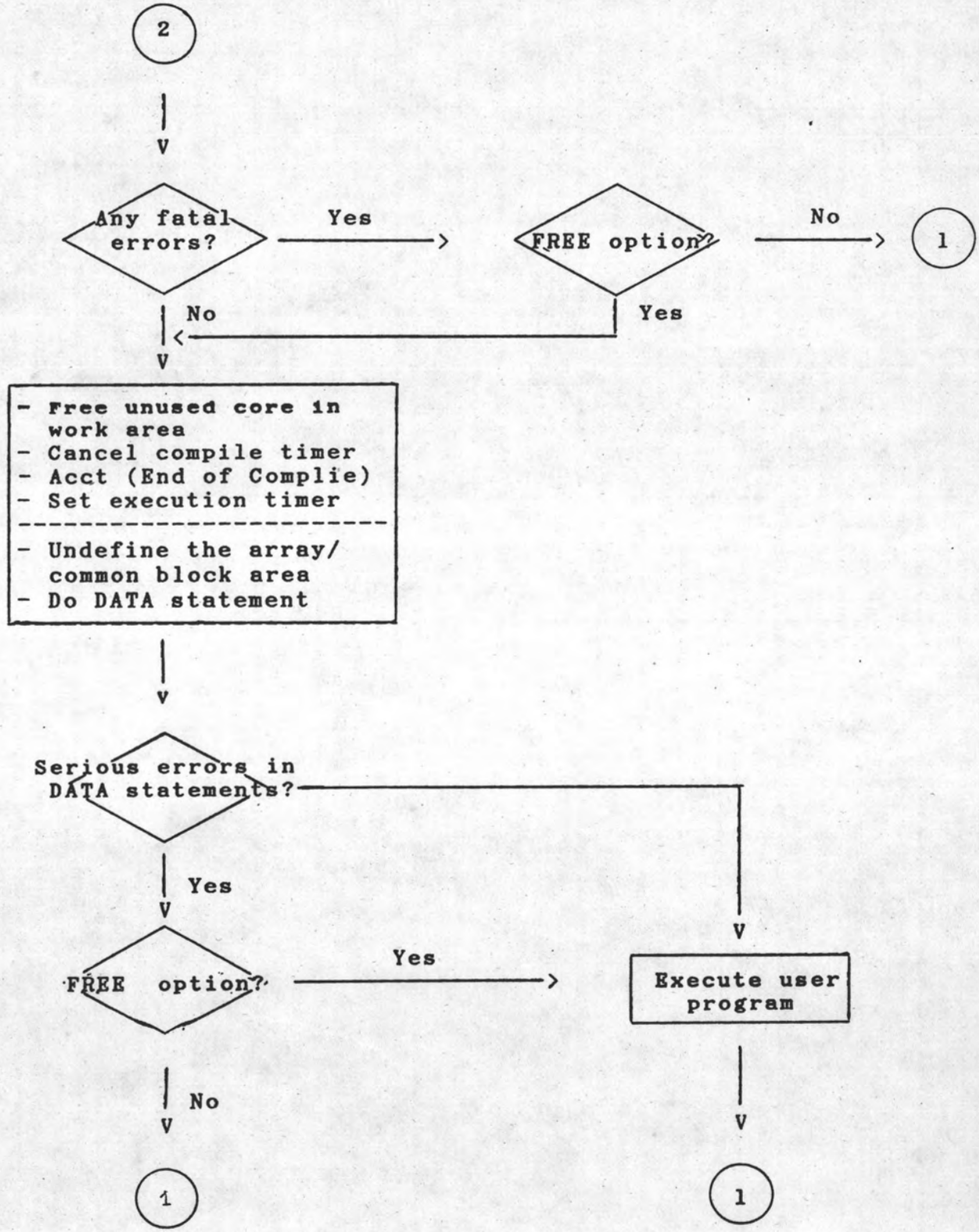
ตัวแปลภาษาวอตฟัฟ ออกแบบมาเพื่อใช้กับระบบควบคุมการทำงานระบบดอส (DOS), ดอส/วีเอส (DOS/VS) และโอเอส (OS) ดังนั้นรูปแบบของแฟ้มข้อมูลต่างๆที่ใช้จึงต้องเป็นไปตามมาตรฐานของระบบนี้ ตัวแปลภาษาวอตฟัฟนี้สามารถแปลโปรแกรมภาษาฟอร์แทรนโมได้ทุกมาร์ติชั้นที่เรียกใช้ และสามารถจัดสรรหน่วยส่วนต่างๆของการทำงานได้ด้วยตัวเอง

การทำงานของตัวแปลภาษาวอศฟิฟจะต้องมีหน่วยความจำอย่างน้อย 128 กิโลไบท์ และจะเก็บไว้ใน CIL (Core Image Library) เมื่อมีการเรียกใช้จะโลดโปรแกรมของตัวแปลภาษาวอศฟิฟทั้งหมดตามที่ระบุตอนติดตั้ง (Implementation) ตัวแปลภาษาวอศฟิฟทำงานแบบ in core, one pass, load and go ซึ่งจะรับโปรแกรมภาษาฟอร์แทรนไปทำการแปล และทำงานจริงต่อเนื่องกันไป

2. โครงสร้างของตัวแปลภาษาวอศฟิฟ

2.1 ผังงานการทำงานของตัวแปลภาษาวอศฟิฟ


entry WATFIV
- Set up base registers
- Open reader and printer
- Acct (Start of Batch)
- Open WATLIB, get a buffer
- Set interrupt exits
- Set compiler timer
- Get core for work area
- Initialize for job
- Read cards until a WATFIV control card is encountered
kind?
$other
$JOB
$STOP
- set defualt parameters
- Acct (Start of Job)
- Compile input from SYSIN
- Acct (End of SYSIN)
- Complie WATLIB programs
Do errors inhibit execution?
Yes
No
1
2
- Free all storage
- Close WATLIB
- Close FIOCS & DIOCS
- Acct (End of Batch)
- Close reader and printer
- Set return code in register 15
Return to Caller
1
- Acct (End of Job)
- Colse FIOCS & DIOCS
- Free all work area
- Print PROFILER statistics if requested



รูป 3.1 ผังงานของตัวแปลภาษาวอตฟิฟ

2.2 โมดูลหลักของตัวแปลภาษาวอตฟัฟ

ตัวแปลภาษาวอตฟัฟประกอบด้วยโมดูลหลัก 6 โมดูล ดังนี้คือ

1. การจัดข้อมูลเข้า/ข้อมูลออก (INPUT/OUTPUT)
2. การจัดการหน่วยความจำ (CORE MANAGEMENT)
3. การขัดจังหวะ (INTERRUPT HANDLING)
4. การคิดบัญชี (JOB ACCOUNTING)
5. ฟังชั่นก์ของระบบควบคุม (OPERATING SYSTEM FUNCTIONS)
6. โปรแกรมประกอบ (SUBPROGRAM FACILITIES)

1 การจัดการข้อมูลนำเข้า/ข้อมูลออก

ตัวแปลภาษาวอตฟิฟ เป็นตัวแปลภาษาที่ทำงานในลักษณะเป็นกลุ่มก้อน โดยจะใช้ชุดข้อมูลซึ่งรับมาจากเครื่องอ่านทั้งในช่วงการแปลและในช่วงการทำงานจริง ส่วนออกของช่วงการแปลและช่วงการทำงานจริง ก็จะแสดงออกทางเครื่องพิมพ์ WATIO จะเป็นโปรแกรมย่อยของตัวแปลภาษาวอตฟิฟ ที่มีหน้าที่รับผิดชอบในการกระทำดังกล่าว

ในการนำข้อมูลเข้า/ข้อมูลออก ของตัวแปลภาษาวอตฟิฟผ่านทางเครื่องอ่าน และเครื่องพิมพ์นั้น จะต้องดำเนินการต่อไปนี้ :

1. ในตอนเริ่มต้นกลุ่มงานของตัวแปลภาษาวอตฟิฟ จะเป็นการเปิดชุดข้อมูลนำเข้าและข้อมูลออก เนื่องจากวอตฟิฟใช้ข้อมูลชุดเดียวกันทั้งช่วงการแปลและช่วงการทำงานจริง

2. เมื่อมีการนำข้อมูลเข้า/ข้อมูลออกใดๆ WATIO จะตรวจสอบดูหมายเลขประจำหน่วย ถ้าพบว่าเป็นเครื่องอ่าน และเครื่องพิมพ์ ก็จะดำเนินงานดังต่อไปนี้ภายใน WATIO คือถ้าเป็นการอ่านบัตรก็จะมีการตรวจสอบดูตัวอักษรควบคุม (control character) ในคอลัมน์ที่ 1 ถึง 2 (คือ @CONTROL)

ถ้าเป็นการพิมพ์ข้อมูลออก ก็จะมีการตรวจสอบดูตัวอักษรควบคุมการขึ้นแครี่พิมพ์ โดยใช้คำสั่งแม็กโคร CCTABLE ใน OPTIONS และจะเพิ่มค่าให้กับเครื่องนับจำนวนบรรทัดและหน้า ถ้าเกินหน้าที่กำหนดมา ก็ถือว่าเป็นข้อผิดพลาดอย่างหนึ่ง และจะส่งการควบคุมกลับไปให้โปรแกรมผู้เรียก (caller)

3. ในตอนท้ายของกลุ่มงาน จะเป็นการปิดชุดข้อมูลทั้งหมด และแม็กโคร FREEPOOL จะปล่อยเนื้อที่ตรงนี้ให้เป็นอิสระ

ถ้าเป็นการนำข้อมูลเข้า/แสดงข้อมูลออกในช่วงการทำงาน ที่ทำงานหน่วยอื่นที่ไม่ใช่เครื่องอ่าน และ เครื่องพิมพ์ ก็จะใช้โปรแกรมย่อย IHCSIOSH และ IHCDIOSE ของไอบีเอ็มทำหน้าที่ข้อมูลเข้า/ข้อมูลออก ประเภทที่สามารถเข้าถึงข้อมูลได้โดยตรงตามลำดับ โปรแกรมย่อยเหล่านี้อาจจะเป็นส่วนหนึ่งใน WATFIV หรือ อาจจะถูกโหลดโดยอัตโนมัติเข้ามา

ในเนื้อที่ทำงาน (work area) เมื่อโปรแกรมของผู้ใช้ต้องการ

การเชื่อมโยงโปรแกรมย่อยๆเข้าด้วยกัน เป็นดังนี้คือ

1. ถ้าเป็นการนำข้อมูลเข้า/ข้อมูลออก แบบเรียงตามลำดับ ก็จะมีการส่งการควบคุมไปให้กับ WATIO ซึ่งจะตรวจสอบดูว่ามาจากหน่วยเครื่องอ่าน หรือ เครื่องพิมพ์ ถ้าพบว่าไม่ได้มาจากหน่วยทั้งสอง ก็จะส่งการควบคุมต่อให้กับ FIOCS หลังจากเสร็จสิ้นการนำข้อมูลเข้า/แสดงข้อมูลออก FIOCS ก็จะส่งคืนการควบคุมให้กับ WATIO แล้ว WATIO ก็จะส่งคืนการควบคุมให้กับโปรแกรมผู้เรียกต่ออีกที

2. ถ้าเป็นการนำข้อมูลเข้า/แสดงข้อมูลออกแบบโดยตรง ก็จะส่งการควบคุมไปให้กับหนึ่งในสองจุดเข้าพิเศษของ WATIO คือ จุดเข้าชื่อ WATDFILE สำหรับคำสั่ง DEFINE FILE และจุดเข้าชื่อ WATDIOCS ถ้าเป็นคำสั่ง READ , WRITE และ FIND ซึ่งจะผ่านต่อการควบคุมไปยังจุดเข้า DIOCS หลังจากสิ้นสุดการนำข้อมูลเข้า/ข้อมูลออกแล้ว การควบคุมก็จะถูกส่งกลับคืนให้กับ WATIO ซึ่งจะผ่านการควบคุมกลับไปให้โปรแกรมผู้เรียกต่อไป

3. WATIO มีรหัสสำหรับจัดการกับข้อผิดพลาดต่างๆที่เกิดขึ้น อันเป็นผลเนื่องมาจากการนำข้อมูลเข้า/ข้อมูลออก ถ้าข้อผิดพลาดเกิดขึ้นใน FIOCS หรือ DIOCS การควบคุมก็จะถูกส่งไปให้กับโปรแกรมย่อยใน WATFIV ที่เรียกว่า IBCOM เพื่อแปลงรหัสข้อผิดพลาดของ IBM ให้เป็นของ WATFIV และส่งการควบคุมกลับไปยัง WATIO ถ้าพบว่ามีรหัสข้อผิดพลาดเกิดเนื่องจากหน่วยเครื่องอ่าน และ/หรือ เครื่องพิมพ์ ก็จะส่งผ่านการควบคุมโดยตรงไปยังกลุ่มโปรแกรมที่จัดการกับข้อผิดพลาด (error handling routine) ใน WATIO ตัวอย่างเช่น การสิ้นสุดของแฟ้มข้อมูล ก็จะถือว่าเป็นข้อผิดพลาด ถ้าผู้ใช้ไม่ระบุ END= ลงไปในคำสั่ง READ ในโปรแกรม WATIO จะจัดการกำหนดรหัสข้อผิดพลาดและส่งต่อการควบคุมไปยัง ERROR# ใน STARTB (เป็นส่วนหนึ่งของกลุ่มโปรแกรม MAIN) โปรแกรมย่อย ERROR# จะกระทำการต่อไปนี้ ซึ่งขึ้นอยู่กับประเภทของข้อผิดพลาดคือ

- ส่งคืนการควบคุมกลับไปให้กับผู้ใช้ ถ้าพบว่าผู้ใช้ระบุ END= และ/หรือ ERR=

ในกรณีที่รหัสข้อผิดพลาดเป็นข้อผิดพลาดประเภทนี้

- แสดงรายละเอียดประเภทของข้อผิดพลาด แล้วก็จะยุติการทำงาน

4. เมื่อ SYSIN พบการสิ้นสุดของแฟ้มข้อมูล ก็จะนำ $STOP (หรือทางคูมือคอมพิวเตอร์นั้นๆจะกำหนด) ไปยังบริเวณที่มีไว้สำหรับเก็บข้อมูลเข้าตอนช่วงการแปลคือ XCARD ในโปรแกรม STARTA และก็กำหนดค่า "ได้เกิด CARD-SWITCH ขึ้น" ใน CIHGACRD ของโปรแกรม COMMR ซึ่งจะเป็นการยุติการทำงานตามโปรแกรมผู้ใช้ (execution of job) และกลุ่มงาน หลังจากนั้นส่งคืนการควบคุมกลับไปให้กับ MAIN เพื่อคิดบัญชีการใช้งานของงานนั้น ข้อมูลเข้าในช่วงการแปล จะถูกอ่านมาจากหน่วยเครื่องอ่าน หรือกลุ่มโปรแกรมภาษาต่างๆ (SOURCE) หรือภาษาเครื่อง (object code) ที่มีเก็บไว้แล้ว โปรแกรมย่อย CREAD ที่มีอยู่ในโปรแกรม COMMR (ซึ่งเป็นส่วนหนึ่งในชุดโปรแกรม MAIN) จะเป็นผู้อ่านข้อมูลเข้ามาทั้งหมดในช่วงการแปล ซึ่งมีลำดับการเรียกดังนี้ :

```
BAL     R14,CREAD
B       LABEL1      การคืนกลับไปยังบัตรควบคุม
B       LABEL2      การคืนกลับไปยังบัตรต่อเนื่อง
B       LABEL3      การคืนกลับไปยังคำสั่งงานใหม่
B       LABEL4      การคืนกลับไปยังกลุ่มภาษาเครื่อง
B       LABEL5      การคืนกลับไปยังบัตรรายละเอียด
```

CREAD จะตรวจสอบดูค่าของ CIHGACRD ว่าเป็น "ได้เกิด CARD-SWITCH ขึ้น" เพื่อดูว่าจะต้องการข้อมูลเข้าอีกหรือไม่ ข้อมูลในรูปเดิมคนบัตรได้ถูกเก็บอยู่แล้วในบริเวณข้อมูลเข้าในรูปบัตร (XCARD) แล้วก็จะตรวจสอบโดยใช้ลิงก์ CSUBRDS เพื่อดูว่าข้อมูลเข้านั้นมาจากหน่วยนำข้อมูลเข้าเครื่องอ่าน หรือจากกลุ่มโปรแกรมที่มีเก็บไว้แล้ว หลังจากนั้นก็จะเรียก WATIO หรือ JSUBRDS ตามลำดับเพื่อนำข้อมูลเข้าในรูปบัตรไปเก็บใน XCARD ถ้าพบว่าเป็นบัตรต่อเนื่อง (CONTINUATION CARD) หรือบัตรคำสั่งงานใหม่ (New Statement) ก็จะ

แทนที่คอลัมภ์ 73 ด้วย X'FF' ซึ่งเป็นเครื่องหมายแสดงการสิ้นสุดคำสั่งงาน ส่วนตำแหน่งที่อยู่ (address) ของที่เก็บบัตรชั่วคราว (XCARD) จะเก็บในรีจิสเตอร์ 1 และความยาวคือ 80 ตัวอักษรก็จะถูกเก็บไว้ในรีจิสเตอร์หมายเลข 2 แล้วก็จะกำหนด 'ได้เกิด CARD-SWITCH ขึ้น' และส่งการควบคุมกลับไปให้ผู้เรียก

2 การจัดการหน่วยความจำ

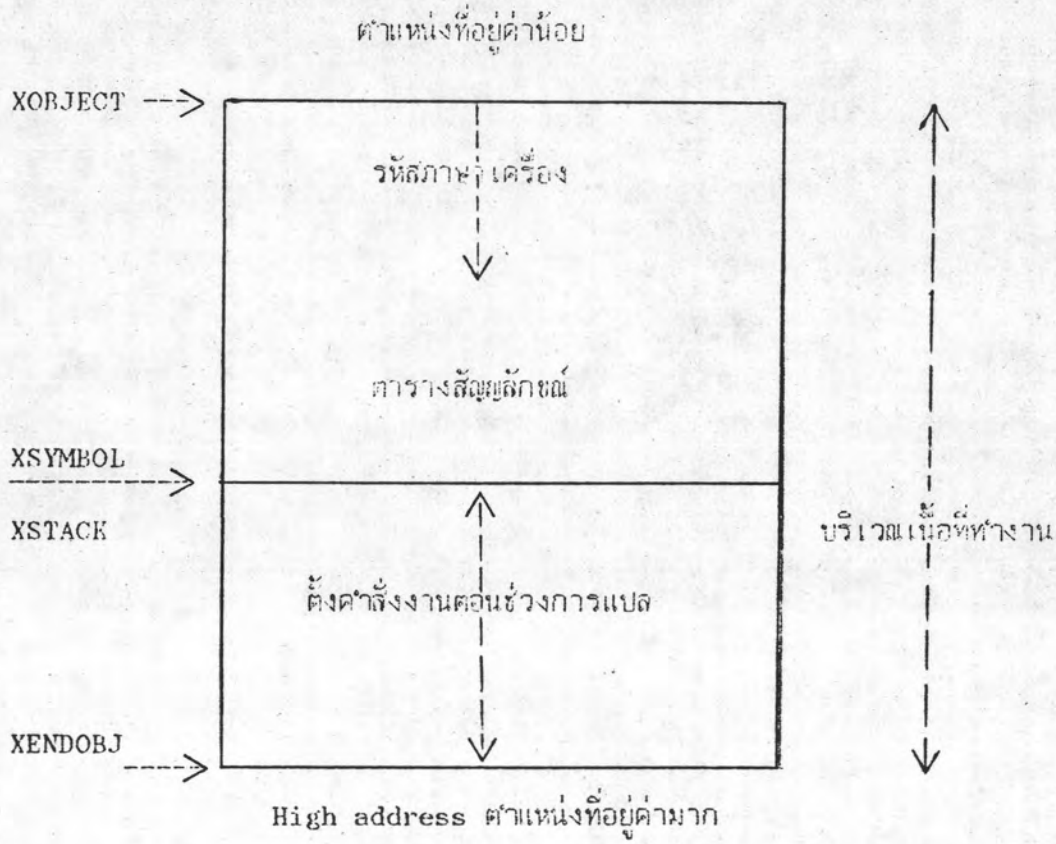
การจัดแบ่งเนื้อที่ทำงานโดย WATFIV อาจจะเป็นการจัดแบ่งได้ 2 แบบ คือ แบบเปลี่ยนแปลงได้ (dynamic) และแบบคงที่ (static)

ขั้นตอนการจัดการหน่วยความจำแบบเปลี่ยนแปลงได้ เมื่อกำหนดให้ &MENTYPE เป็น "MAX"

1. ในตอนเริ่มงานของแต่ละกลุ่ม WATIO จะเปิดชุดข้อมูลสำหรับเครื่องอ่านบัตร และเครื่องพิมพ์ บัฟเฟอร์ (buffer) จะเป็นที่เก็บข้อมูลเหล่านั้น

2. ถ้ามีการเรียกโปรแกรมในกลุ่มโปรแกรมที่เก็บรวบรวมไว้แล้วเพื่อให้เรียกมาใช้งานได้โดยตรง (direct-access library feature) โปรแกรมย่อย TIOT จะตรวจดูว่ามีโปรแกรมชื่อ WATLIB ถ้ามี WATLIB ก็จะถูกเปิดและโปรแกรมย่อย GETMAIN จะกันเนื้อที่ไว้ให้

3. ในตอนเริ่มต้นของงานแต่ละงานในกลุ่ม WATFIV โปรแกรมย่อย GETMAIN จะพยายามหาเนื้อที่ว่างๆที่ใหญ่ที่สุดในพาติชั่นนั้นๆ และส่วนหนึ่งของเนื้อที่นี้จะต้องถูกใช้โดยตัวควบคุมระบบ (Operating System) จะมีการตั้งตัวชี้ทั้งหลายที่จำเป็นสำหรับตัวแปลภาษาขึ้นมา ดังแสดงในภาพต่อไปนี้



รูป 3.2 แสดงการจัดแบ่งหน่วยความจำ

ตั้งคำสั่งงาน (stack for source statement) ภาษาต่างๆในช่วงการแปลนั้นจะเก็บอยู่ในเนื้อที่ทำงานนี้ และมีขนาดคงที่อยู่กับจำนวนบัตรต่อเนื่องที่ระบุไว้ตอนแรก จะมีข้อความเตือนออกมาถ้าจำนวนบัตรต่อเนื่องมากกว่าตัวกำหนด &NOCCRDS แต่การแปลจะดำเนินต่อไปจนกระทั่งตั้งนี้เต็ม ซึ่งจะมีข้อความผิดพลาดชนิดร้ายแรงแสดงออกมา กลุ่มโปรแกรมผู้ป้อนภาษาเครื่องของ WATFIV (WATFIV's Object deck loader) ก็ใช้เนื้อที่ในตั้งนี้สำหรับเก็บตารางสัญญลักษณ์ต่างๆ (Symbol Table) ที่ใช้ในโปรแกรมแต่ไม่ได้กำหนดความหมาย ตารางนี้จะเก็บสัญญลักษณ์ต่างๆได้ทั้งหมดเป็นจำนวน 256 ตัว และแต่ละสัญญลักษณ์จะมีขนาด

เป็น 4 ไบท์ ดังนั้นขนาดเล็กสุดของตั้งนี้ที่ WATFIV ต้องใช้ คือสามารถบรรจุบัตรต่อเนื่องได้ 5 บัตร ถ้าศูนย์คอมพิวเตอร์นั้นๆ ระบุว่าให้ใช้น้อยกว่า 5 บัตร เนื้อที่ตั้งนี้ก็สามารถเก็บได้ 5 บัตร

ถ้าตัวแปร &LIBCORE และ/หรือ @LIBSTACK มีค่าเป็น "USE" และถ้าตัวแปร @OPSYS มีค่าไม่ใช่ "DOS" หรือ "DOSVS" แล้วแต่ละแห่งในตั้งนี้ที่เกิดจะมีขนาดเป็น 12 ไบท์ โดยใช้ 8 ไบท์เก็บชื่อ และ 4 ไบท์ เป็นตำแหน่งที่อยู่ TTR ตอนช่วงการแปลจะเก็บรหัสภาษาเครื่อง เริ่มจากตำแหน่งที่อยู่ต่ำน้อยไปตำแหน่งที่อยู่ต่ำมากและตารางสัญญลักษณ์เริ่มจากตำแหน่งที่อยู่ต่ำมากไปตำแหน่งที่อยู่ต่ำน้อย ส่วนตารางสัญญลักษณ์ที่ถูกใช้จะส่งคืนให้กับ WATFIV ซึ่งกระบวนการนี้จะทำให้สามารถแปลโปรแกรมขนาดใหญ่ๆได้โดยโปรแกรมนั้นจะต้องถูกแบ่งย่อยเป็นส่วนๆ และเก็บในเนื้อที่ทำงานขนาดเดียวกัน

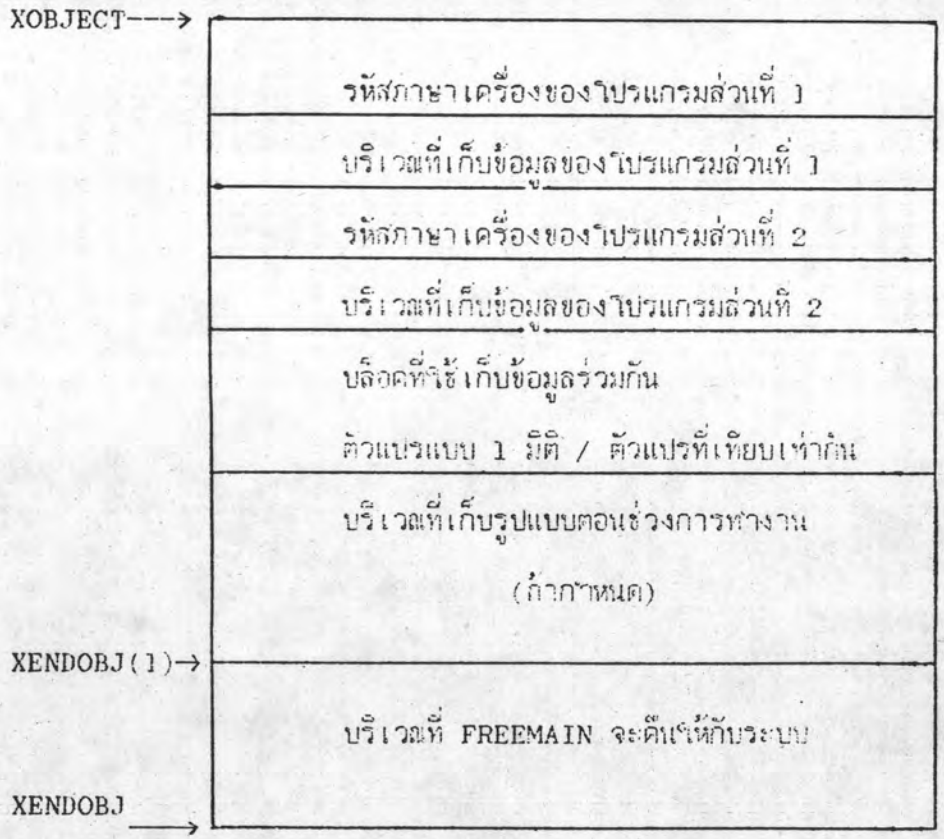
ถ้าตารางสัญญลักษณ์มาพบกับรหัสภาษาเครื่องจะแสดงข้อผิดพลาดออกมา ตัวชี้ (Pointer) ของรหัสภาษาเครื่องจะถูกตั้งใหม่ โดยชี้ไปที่ส่วนท้ายของเนื้อที่ทำงาน และหยุดการสร้างรหัสภาษาเครื่อง ซึ่งทำให้สามารถเพิ่มขนาดตารางสัญญลักษณ์ได้กระบวนการแปลก็จะดำเนินต่อไปจนกว่าตารางสัญญลักษณ์จะกินเนื้อที่ตั้งที่สุดของเนื้อที่ทำงาน ซึ่งถ้าเกิดขึ้นเมื่อใดก็จะแสดงข้อผิดพลาดอย่างมหันต์ (fatal error message) ออกมา และการแปลจะยุติลง

4. เมื่อแปลคำสั่งโปรแกรมภาษาต่างๆ (source statement) ที่เข้ามาทางหน่วยนำข้อมูลเข้าหมดแล้ว โปรแกรมย่อย MLIBR ในกลุ่มโปรแกรม MAIN จะตรวจสอบดูว่ายังมีสัญญลักษณ์ที่ถูกอ้างอิงในตารางสัญญลักษณ์ใหม่บางที่ยังเป็นปัญหาอยู่ สำหรับแต่ละสัญญลักษณ์ที่ยังเป็นปัญหาอยู่ ซึ่งไม่มีชื่อปรากฏอยู่ในคำสั่ง EXTERNAL สิ่งแรกที่จะทำคือ จะมีการค้นดูรายชื่อของโปรแกรมย่อยทั้งหลายที่เก็บไว้ในกลุ่มฟังชั่นก์ (FUNCTION) ถ้าค้นพบก็จะนำตำแหน่งที่อยู่ของจุดเข้าโปรแกรมย่อยนั้นมาใช้ แต่ถ้าค้นไม่พบจะค้นที่ตั้งคำสั่งที่ถูกสร้างขึ้นแล้ว (stack generated) โดยโปรแกรมย่อย STACKERS และ/หรือ ที่แม็กโคร STACK (ถ้า @LIBCORE และ/หรือ @LIBSTACK ถูกกำหนดว่า 'use') ถ้าพบโปรแกรมย่อย FIND จะนำตำแหน่งที่อยู่

ของโปรแกรมย่อยนั้นๆมาใช้ แต่ถ้าไม่พบก็จะค้นหาในชุดโปรแกรม WATLIB ถ้าพบก็จะตั้งสวิทช์เพื่อให้อ่านข้อมูลเข้ามาให้กับตัวแปลภาษาจากแหล่งที่เหมาะสม กลุ่มแกรมที่ถูกดึงออกมาในนี้จะถูกเพิ่มเข้าไปในเนื้อที่ที่ใช้เก็บรหัสภาษาเครื่องและตารางสัญญลักษณ์

5. หลังจากอ่านกลุ่มโปรแกรมเข้าไปจนครบและทำการแปลเสร็จเรียบร้อยแล้ว ก็จะมีการจัดสรรเนื้อที่ทางานส่วนที่อยู่เหนือที่เก็บรหัสภาษาเครื่องให้ใช้เก็บตัวแปรแบบ หนึ่งมิติ (arrays) ทั้งหลายและบล็อคที่ใช้ร่วมกัน (common blocks) ตัวแปลภาษาก็จะดูว่ามีการอ้างอิงถึงรูปแบบของช่วงการทางาน (execution-time formats) ในโปรแกรมที่ถูกแปลหรือไม่ ถ้ามีก็กันเนื้อที่ส่วนที่อยู่เหนือเนื้อที่ที่ใช้เก็บตัวแปรแบบหนึ่งมิติไว้ในปริมาณที่เหมาะสม

6. WATFIV จะคืนเนื้อที่ที่ไม่มีการใช้งานดังแสดงในรูป 3.5



รูป 3.3 Free area before Execution time

(1) จะถูกแก้ไขให้ไปชี้ที่จุดสูงสุดของเนื้อที่ทางานที่ถูกใช้งานจริง

7. เมื่อสิ้นสุดการทำงาน WATFIV จะปิด FIOCS และคืนเนื้อที่ในหน่วยทรงจำที่ใช้เก็บโปรแกรมย่อย 2 อันนี้ และคืนเนื้อที่ทั้งหมดที่ใช้ในการทำงานและจะกลับไปขั้นตอนที่ 3 อีก ถ้าในกลุ่มยังมีงานอีก หรือไปทำขั้นตอนที่ 8 ในกรณีที่หมดงานของกลุ่มแล้ว

8. ในตอนสิ้นสุดการทำงานจริง เนื้อที่ทั้งหมดที่ถูกใช้โดย WATFIV รวมทั้งเนื้อที่สำหรับ EREAD และ EPRINT และชุดข้อมูล WATLIB จะถูกปล่อยให้เป็นอิสระ

ถ้า WATFIV มี EMEMTYPE เป็น "FIXED" เนื้อที่ทำงานจะถูกแบ่งย่อยลงไปอีก เพื่อให้ตัวแปลภาษาใช้ในลักษณะวิธีเดียวกับที่ได้อธิบายมาแล้วสำหรับแบบเปลี่ยนแปลง

3 การจัดกับการถูกขัดจังหวะ

หัวข้อนี้จะพูดถึงวิธีการที่ WATFIV จัดการ กับการขัดจังหวะโปรแกรมที่กำลังทำงานอยู่ โดยจะพิจารณารายละเอียดในหัวข้อต่อไปนี้

- การขัดจังหวะที่ WATFIV ยอมรับให้เกิดขึ้นได้
- หลักตรรกวิทยาของโปรแกรมที่มีหน้าที่จัดการกับการขัดจังหวะ
- การยุติการทำงานของตัวแปลภาษาเนื่องจากความผิดพลาดเกิดขึ้น
- จุดอ่อนของโปรแกรมย่อยที่หน้าที่จัดการกับการขัดจังหวะ

รหัสทั้งหมดที่มีหน้าที่จัดการกับการขัดจังหวะ จะเก็บไว้เป็นส่วนหนึ่งของโปรแกรม JSYS ซึ่งจะรวมแม็กโคร SPIE (ซึ่งจะมีชื่อ JMSPIE ในโปรแกรม MAIN) เมื่อเกิดการขัดจังหวะโปรแกรมที่กำลังทำงานอยู่ โปรแกรมควบคุมก็จะส่งต่อการควบคุมไปให้กับ XRUPT ซึ่งจะตรวจว่าการทำงานของ WATFIV อยู่ในช่วงการแปลหรือช่วงการทำงานจริง

1. ถ้าเป็นช่วงการแปล :- การขัดจังหวะที่เกิดขึ้นทั้งหมดยกเว้นที่เกิดจากกรณีที่ค่าทศนิยมที่เล็กเกินไป (13) และที่ใหญ่เกินไป (12) จะคิดเป็นข้อผิดพลาดของตัวแปลภาษา (compiler error) และแสดงข้อผิดพลาดออกมา ส่วนกรณียกเว้น 2 กรณีนี้ จะป้อนค่า 0 เข้าไปเก็บในรีจิสเตอร์ของค่าทศนิยม (Floating register) สำหรับกรณีค่าทศนิยมที่น้อย

เกินไป (underflow) หรือจะป้อนค่าบวกที่ใหญ่ที่สุดเข้าไปเก็บ สำหรับกรณีค่าทศนิยมที่ใหญ่เกินกำหนด (overflow) โปรแกรมย่อย COLCONST (ใน COMMR) ซึ่งเป็นโปรแกรมย่อยเดียวที่จะดำเนินการทางเลขคณิตในช่วงการแปลกับจำนวนเลขทศนิยม แล้วก็จะแสดงข้อความเกี่ยวกับข้อผิดพลาดนี้ออกมา เพื่อเป็นการแจ้งให้ทราบว่าได้เกิดกรณีค่าใหญ่เกินไปหรือเล็กเกินไปขึ้นในขณะที่ดำเนินการรวมวิธีกับค่าคงที่ในคำสั่งงาน (SOURCE STATEMENT) หลังจากนั้นก็จะกลับคืนการควบคุมมาให้ระบบ แล้วจึงกลับคืนมา COLCONST

2. ถ้าเป็นช่วงการทำงานจริงตามคำสั่งในโปรแกรม :- การขัดจังหวะมีสาเหตุมาจาก

- การปฏิบัติงาน (1) : operation
- การปฏิบัติงานด้วยคำสั่งพิเศษ (2) : privileged operation
- การทำงานงานจริงตามโปรแกรม (3) : execution
- ข้อมูล (7) : data
- ค่าทศนิยมที่ใหญ่เกินไป (10) : decimal overflow
- การหาร (11) : divide

จะเป็นข้อผิดพลาดของตัวแปลภาษา และใช้ทางออกที่มีไว้ให้กับข้อผิดพลาดประเภทนี้ ส่วนการขัดจังหวะเนื่องมาจาก

- การหารจำนวนที่มีทศนิยมคงที่ (9) : fixed divide
- การหารจำนวนที่มีทศนิยมยกกำลัง (15) : floating divide
- ตัวยกกำลังที่ใหญ่เกินกำหนด (12) : exponent overflow
- ตัวยกกำลังที่เล็กเกินกำหนด (13) : exponent underflow

จะคล้ายๆกัน โดยที่ตัวนับของการขัดจังหวะแต่ละประเภทจะถูกลดค่าไปทีละ 1 เมื่อใดที่ได้ผลลัพธ์เป็น 0 ก็จะมีข้อความเกี่ยวกับข้อผิดพลาดแสดงออกมา และตามด้วยการยุติการทำงาน พร้อมกับมีรายละเอียดบอกถึงสาเหตุที่ผิดพลาดนั้น ซึ่งจะทำให้ผู้ใช้สามารถย้อนกลับไปดูและ

แก้ไขข้อผิดพลาดนั้นได้ ตัวนับเหล่านี้ถ้าไม่มีการกำหนดค่ามาให้ก็จะถูกกำหนดให้มีค่าเป็น 1 โดยอัตโนมัติเลยในโปรแกรม MAIN ในส่วนเริ่มการทำงาน แต่อาจจะกำหนดค่าให้กับตัวนับนั้นได้ใหม่ โดยโปรแกรมที่ได้รับการแปลเรียบร้อยแล้ว โดยใช้โปรแกรมย่อยชื่อ TRAPS ในกรณีที่ผลลัพธ์ในตัวนับไม่เป็น 0 ก็จะมีการตั้งสวิทช์ของค่าที่ใหญ่เกินไป (แล้วแต่ว่าอันไหนเหมาะสม) ตามกฎเกณฑ์ของโปรแกรมย่อย DVCHK , OVERFL แล้วจึงจะส่งคืนการควบคุมไปให้กับระบบ นั่นก็คือกลับไปยังโปรแกรมภาษาเครื่องที่ถูกขัดจังหวะการทำงาน ถ้าเป็นกรณีด้วยกกำลังใหญ่เกินกำหนดหรือเล็กเกินกำหนด ก็ต้องเพิ่มการดำเนินการ ทั้งนี้ก็เพื่อจะจัดการกำหนดค่าให้กับริจิสเตอร์ที่เกี่ยวข้องด้วย โดยค่านั้นอาจจะเป็นจำนวนใหญ่ที่สุดพร้อมเครื่องหมายที่ถูกต้อง หรืออาจจะเป็นค่าศูนย์ ก็แล้วแต่กรณีที่เกิดขึ้นตามลำดับ แล้วจึงกลับคืนมายังโปรแกรม

3. การดำเนินการกับการขัดจังหวะอันมีสาเหตุเกี่ยวข้องกับ

- การกำหนดตำแหน่งที่อยู่ (addressing : 5)

- การป้องกันการประมวณข้อมูลต่างๆ (processing of protection : 4)

จะเป็นการทดสอบเพื่อดูว่า การขัดจังหวะมีสาเหตุมาจากการใช้ ADCON ที่ตัวแปลภาษาสร้างขึ้นมาโดยเฉพาะ ในการพยายามที่จะดักข้อผิดพลาดของการเขียนโปรแกรมทั้งหลาย ดังแสดงจากตัวอย่างต่อไปนี้ ซึ่งมีการเรียกค่าคงที่ตามตำแหน่งที่เก็บมาใช้งาน ซึ่ง ณ จุดที่ทำการเรียกใช้ ก็ยังไม่ได้มีการกำหนดตำแหน่งที่เก็บเลย

```
CALL  RTN                    CALL  RTN2 (Y)

  .                            .
  .                            .
  .                            .

  END                          END

SUBROUTINE RTN1 (X)          SUBROUTINE RTN (/X/)
DIMENSION X(10)              ENTRY RTN2 (A)
ENTRY RTN2 (A)               A = X
A = X(1)
```

- ถ้ามีการทดสอบระบุถึงข้อผิดพลาดประเภทนี้ ก็จะยุติการทำงานนั้นพร้อมกับพิมพ์รายละเอียดเกี่ยวกับข้อผิดพลาดออกมา เพื่อให้ผู้ใช้สามารถย้อนกลับไปดูได้

- ถ้าการทดสอบล้มเหลวก็ให้ถือว่าเป็นข้อผิดพลาดของตัวแปลภาษา และให้ใช้ทางออกสำหรับข้อผิดพลาดประเภทนี้

4. การดำเนินการกับการขัดจังหวะที่เกิดจากข้อกำหนดรายละเอียด (Processing of specification : b) จะเป็นการตรวจดูว่า คำสั่งงานอันนี้มีความยาวเป็น 4 ไบท์หรือไม่ และเก็บอยู่ที่ตำแหน่งที่อยู่เลขคู่ (even address) หรือไม่

- ถ้าไม่ผ่านการทดสอบ ก็ให้ใช้ทางออกที่มีไว้ให้กับข้อผิดพลาดอันเกิดจากตัวแปลภาษา

- ถ้าผ่านการทดสอบนั้นได้ ก็ถือว่าการขัดจังหวะเป็นผลของการที่คำสั่งที่อ้างอิงถึงตัวถูกกระทำ (operand) ถูกบังคับให้ไปยังเขตที่ไม่ถูกต้องโดยคำสั่งงาน COMMON และ/หรือ EQUIVALENCE ในกรณีนี้จะต้องมีการจัดการแก้ไขกำหนดวางขอบเขตใหม่ ซึ่งทำได้โดยการเคลื่อนเขตข้อมูลที่เก็บตัวถูกกระทำแบบชิดซ้าย (operand left justified) ซึ่งมีความเขตเป็น 8 ไบท์ ไปยังคำที่มีขนาดเป็น 2 เท่า (double word) ของปกติ แล้วก็จะจัดการ

แก้ไขคำสั่งงานใหม่ โดยจะอ้างถึงคำนี้ที่มีขนาดใหญ่เป็น 2 เท่า และเคลื่อนย้ายคำนี้กลับที่ เพื่อจะให้เห็นการเปลี่ยนแปลงที่อาจเป็นผลของการกระทำนี้ จะใส่รหัสแสดงสภาพการณ์ (condition code) เข้าไปในรีจิสเตอร์ที่เก็บข้อมูลสภาพการณ์เก่าของโปรแกรม ซึ่งเก็บอยู่ใน PIE เพื่อจะจองไว้ให้กับโปรแกรมภาษาเครื่องที่ถูกขัดจังหวะ แล้วจึงกลับคืนมาสู่ระบบ ซึ่งก็คือการกลับมายังโปรแกรมภาษาเครื่องที่ถูกขัดจังหวะ

5. โปรแกรมย่อยทางออกของข้อผิดพลาดที่เกิดจากตัวแปลภาษา จะพิมพ์รายละเอียดของข้อผิดพลาดที่เกิดขึ้น แล้วแต่ว่าจะเป็นข้อผิดพลาดที่เกิดในช่วงการแปล หรือช่วงการทำงานจริงตามคำสั่งในโปรแกรม แล้วก็พยายามปิดแฟ้มข้อมูลทั้งหมดที่ถูกเปิด และยุติการทำงานของกลุ่มเลย โดยวิธีการต่อไปนี้ : บิทที่ 2 ของ PIE จะถูกกำหนดให้มีค่าปิด อันจะทำให้สามารถขัดจังหวะได้ในช่วงขณะที่ยังอยู่ในช่วงการทำงานของโปรแกรมย่อย ที่จัดการกับการขัดจังหวะนี้ สวิทช์นี้จะตรวจสอบเพื่อดูว่าเป็นการยุติการทำงานครั้งที่ 2 (The Second execution of this termination) ของโปรแกรมหรือไม่ ถ้าใช่ก็จะส่งคืนการควบคุมให้กับระบบทันทีพร้อมกับกำหนดค่ารหัสแสดงภาพเป็น "8" เก็บไว้ในรีจิสเตอร์ 15 ซึ่งหมายความว่าการขัดจังหวะเกิดขึ้นในขณะที่กำลังปิดแฟ้มข้อมูล ดังนั้นจึงอาจจะยังมีแฟ้มข้อมูลบางแฟ้มที่ยังไม่ได้ถูกปิด ซึ่งก็หมายความว่าที่เก็บข้อมูลชั่วคราวเหล่านี้ก็ยังไม่ถูกปล่อยเป็นอิสระ แต่ถ้าตรวจสอบพบว่าเป็นการทำงานครั้งแรก ก็จะมีการตั้งสวิทช์ และจะเลือกทางออกไปที่โปรแกรมย่อยที่จัดการกับการสิ้นสุดของกลุ่มงานชื่อ "MSTOP" ซึ่งเป็นส่วนหนึ่งของโปรแกรม MAIN เพื่อปิดแฟ้มข้อมูลทั้งหมด และจะส่งผลให้กลับคืนการควบคุมมาให้กับระบบ พร้อมกับกำหนดรหัสแสดงสภาพเป็น "5" ไปเก็บไว้ในรีจิสเตอร์ 15 (MSTOP จะนำค่า 0 ป้อนเข้าไปเก็บในรีจิสเตอร์ 15 ถ้าเป็นการลบข่าวสารในรีจิสเตอร์ในตอนสิ้นสุดกลุ่มอย่างปกติ)

6. ข้อบกพร่องของโปรแกรม XRUPT มีดังนี้

1) เมื่อ XRUPT เริ่มทำงาน ก็จะป้อนตำแหน่งที่อยู่โปรแกรม STARTA เข้าไป เก็บไว้ในรีจิสเตอร์ 12 เพื่อจะได้สามารถรู้ตำแหน่งที่อยู่ของ STARTA ได้ ค่าเดิมที่

เก็บอยู่ในรีจิสเตอร์ 12 ตอนเกิดการขัดจังหวะจะไม่ถูกเก็บไว้ หรือไม่ถูกนำกลับมาเก็บในรีจิสเตอร์ 12 ใหม่ตอนออกจาก XRUPT นี้ นั่นคือโปรแกรมย่อยที่ถูกขัดจังหวะการทำงานอาจจะสูญเสียค่าที่เก็บอยู่เดิมในรีจิสเตอร์ 12 ไปเลย (โปรแกรมต่างๆที่ WATFIV แปล มักใช้รีจิสเตอร์ 12 นี้เป็นฐานของ STARTA ตลอดช่วงการทำงานจริงตามคำสั่งในโปรแกรม)

2) จะไม่มีการกำหนดรหัสแสดงสภาพให้กับโปรแกรมย่อย ที่ถูกขัดจังหวะอันมีสาเหตุเนื่องจากตัวยกกำลังที่มีค่าใหญ่เกินกำหนด และค่าเล็กกว่ากำหนด

3) โปรแกรมย่อยที่มีหน้าที่จัดการแก้ไขขอบเขตใหม่ จะใช้รีจิสเตอร์ 14 เป็นรีจิสเตอร์ทำงาน ดังนั้นถ้าคำสั่งงานที่ถูกขัดจังหวะเกิดมีการไปเปลี่ยนแปลงแก้ไขค่าในรีจิสเตอร์นั้นๆ ก็จะทำให้การทำงานของโปรแกรมย่อยนี้ไม่ประสบความสำเร็จ

4) ถ้าพบว่าตัวแปลภาษาเข้าสู่โปรแกรมย่อย ที่จัดการกับการขัดจังหวะเป็นครั้งที่ 2 และถ้าคำสั่งงานที่เป็นสาเหตุของการแก้ไขขอบเขตใหม่เป็นสาเหตุที่ทำให้เกิดมีการขัดจังหวะขึ้น เช่นการหารด้วยจำนวน 0 ก็จะยุติการทำงานของตัวแปลภาษาทั้งๆ ที่ยังทำงานไม่เสร็จ

5) โปรแกรมย่อยที่จัดการกับการออกจากข้อผิดพลาด อันเกิดเนื่องจากตัวแปลภาษา จะมีการไปดำเนินการอันไม่พึงกระทำกับสวิทช์จุดเข้าอันที่ 2 อันเป็นความลับของระบบ

4 การคิดบัญชี

WATFIV จะรวบรวมหน้าที่เกี่ยวกับบัญชีต่างๆมาไว้ที่หน่วยโปรแกรม ACCT ซึ่งจะถูกลอกลงเป็นส่วนหนึ่งของ MAIN หน้าที่เหล่านี้จะค่อนข้างจำกัดเนื่องจากว่ามันจะตรวจสอบดูพารามิเตอร์ต่างๆ ที่ปรากฏอยู่บนบัตร $JOB หรือ ($OPTION) ในตอนเริ่มต้นของแต่ละงาน และเวลาที่ใช้ในการพิมพ์ และปริมาณเนื้อที่ที่เก็บข้อมูลในตอนสิ้นสุดของแต่ละงาน อย่างไรก็ดีได้มีการสอดแทรกจุดเรียกต่างๆ เข้าไปในที่ที่เหมาะสมในตัวแปลภาษา สำหรับการดำเนินการ

อย่างอื่น ซึ่งคู่นย์คอมพิวเตอร์นั้นๆอาจจะประสงค์จะทำ การติดตั้งตัวแปลภาษาที่ครบสมบูรณ์ได้ตราบเท่าที่รวมจุดเข้า (entry points) ทั้งหลายที่เหมาะสมไว้ด้วย ชื่อของจุดเข้าต่างๆ และหน้าที่การทำงานก็ได้แก่ :

หน้าที่	ชื่อของจุดเข้า
เริ่มต้นกลุ่มงาน	BSTART
เริ่มต้นงาน	BBEGJOB
สิ้นสุดโปรแกรมภาษาต้นฉบับ	BENDSRS
สิ้นสุดการแปล	BENDCOMP
สิ้นสุดการทำงาน	BENDJOB
สิ้นสุดกลุ่มงาน	BSTOP

รูป 3.4 Entry points of Job Accouting

4.1 การทำงานของโปรแกรมการคิดบัญชี : -

1. หน่วยโปรแกรมทางการบัญชีต้องการข่าวสารที่ตัวแปลภาษาเก็บไว้ ซึ่งส่วนใหญ่แล้วข่าวสารข้อมูลเหล่านี้จะเก็บอยู่ในหนึ่งในสองของบริเวณที่ใช้ในการติดต่อสื่อสารข้อมูลของโปรแกรม COMMR และ STARTA ซึ่งตำแหน่งที่อยู่จะเก็บอยู่ในรีจิสเตอร์หมายเลข 10 และ 12 ตามลำดับ การกำหนดค่าในรีจิสเตอร์เหล่านี้จะกระทำใน MAIN ตั้งแต่ก่อนจะมีการเรียกใช้ ACCT
2. โปรแกรมย่อย CENT ใน COMMR จะถูกเรียก ณ แต่ละจุดเข้าใน ACCT โดยใช้คำสั่งแม็กโคร CENT เพื่อจะกำหนดให้รีจิสเตอร์หมายเลข 11 เก็บ

รีจิสเตอร์ฐานสำหรับโปรแกรมย่อยนั้นๆ

หน้าที่ของโปรแกรมย่อย CENT คือ

- เก็บค่าที่ปรากฏอยู่ในรีจิสเตอร์ต่างๆของโปรแกรมที่เรียกไว้
- นำค่าไปเก็บในรีจิสเตอร์ 13 เพื่อให้ชี้ไปที่บริเวณที่เก็บ ACCT คือ BSAVE
- ส่งคืนการควบคุมให้กับ ACCT

หลังจากนั้น ACCT ก็จะคืนการควบคุมกลับมาที่จุดเรียกโดยใช้คำสั่งแม็กโคร CRET ซึ่งจะเป็นการทำให้รีจิสเตอร์ 11, 13 และ 14 ของโปรแกรมที่เรียกกลับมามีค่าเดิมก่อนเกิดการเรียก การทำเช่นนี้จะเป็นไปตามข้อกำหนดของ MAIN แต่ให้สังเกตว่ารหัสใดก็ตามที่เราใส่เพิ่มเข้าไปใน ACCT ไม่ควรจะไปเปลี่ยนแปลงแก้ไขค่าในรีจิสเตอร์ 10 และ 12 เพราะทั้ง 2 รีจิสเตอร์ จะเก็บค่าตำแหน่งที่อยู่บริเวณที่จะใช้ในการติดต่อสื่อสารทั้งสองโปรแกรม คือ COMMR และ STARTA

3. ถ้ามีการเปลี่ยน $JOB หรือ C$OPTIONS ในส่วนที่ทำการวิเคราะห์พารามิเตอร์ของ ACCT ก็ควรจะระมัดระวังเป็นพิเศษในการจะกำหนดค่าให้กับพารามิเตอร์ทั้งหลาย และ สวิทช์ต่างๆที่ตัวแปลภาษาต้องใช้ให้ถูกต้อง ควรใช้โปรแกรมย่อย BBEGJOB เป็นหลักในการพิจารณาว่าต้องการพารามิเตอร์และสวิทช์ต่างๆอะไรบ้าง ให้สังเกตว่าก่อนจะเรียก BBEGJOB จะมีการกำหนดค่าให้โดยอัตโนมัติอยู่แล้วใน MAIN
4. สำหรับบรรทัดสุดท้ายเกี่ยวกับบัญชี จะเป็นการบอกวันที่และเวลาของวันที่ทำงานนั้นโดยได้มาจากการเรียกหน่วยโปรแกรม ZMONTH

4.2 โปรแกรมย่อยต่างๆที่ใช้ใน ACCT

1 โปรแกรมย่อยที่รับผิดชอบต่อการเริ่มต้นของกลุ่มงาน (BSTART)

โปรแกรมย่อยนี้จะเก็บตำแหน่งที่อยู่ของพารามิเตอร์ในรีจิสเตอร์ 1 ถ้าพบว่าความยาวของพารามิเตอร์เป็นศูนย์ ก็จะส่งคืนการควบคุมกลับไปผู้เรียก ถ้าพบว่าพารามิเตอร์เป็น 'DEBUG' ก็จะกำหนดค่าของพารามิเตอร์เป็นแบบงานนักเรียน

2 โปรแกรมย่อยที่รับผิดชอบต่อการเริ่มต้นของงาน (BBEGJOB)

โปรแกรมย่อยนี้จะพิเคราะห์ดูพารามิเตอร์ทั้งหมดที่ปรากฏอยู่บนบัตร $JOB และกำหนดค่าต่างๆ ที่เหมาะสมสำหรับการทำงานของตัวแปลภาษาจะมีการมอบหมายค่าให้ (ดังที่ได้ระบุไว้ใน OPTIONS) โดยอัตโนมัติในกรณีที่ไม่มีการกำหนดค่ามาในโปรแกรม โดยค่าที่จะกำหนดโดยอัตโนมัตินี้ได้จัดการกำหนดมาเรียบร้อยแล้วในกลุ่ม MAIN ก่อนที่จะมีการเรียกไปยัง BBEGJOB สามารถอธิบายการทำงานของโปรแกรมย่อยนี้ได้ดังต่อไปนี้ :

- พิมพ์บัตร $JOB ออกมา (พร้อมกับกระโดดข้ามไปขึ้นหน้าใหม่)
- เริ่มต้นการพิเคราะห์หาพารามิเตอร์ โดยพารามิเตอร์ EPARMCOL และ EIDENT จะระบุถึงจุดเริ่มต้นของพารามิเตอร์ต่างๆเหล่านั้นบนบัตร ควรสังเกตว่าถ้าพบข้อผิดพลาดในหลักไวยากรณ์ของพารามิเตอร์ จะมีข้อความเตือนแสดงออกมาสำหรับพารามิเตอร์ตัวนั้น และการพิเคราะห์ก็จะดำเนินการต่อไป เขตข้อมูลอันแรกที่พบที่ไม่มีอะไรเก็บอยู่จะเป็นการระบุถึงการสิ้นสุดรายชื่อพารามิเตอร์
- ค่าดงที่บางตัวจะถูกกำหนดค่ามาสำหรับตัวแปลภาษา ทั้งนี้ก็ขึ้นอยู่กับพารามิเตอร์ตัวที่กำลังถูกดำเนินกรรมวิธีอยู่ในขณะนั้น รหัสใน ACCT จะค่อนข้างตรงไปตรงมา และอาจพิจารณาสถานการณ์แวดล้อมต่างๆ ได้จากหน่วย ACCT นี้

3 โปรแกรมย่อยที่รับผิดชอบต่อการสิ้นสุดโปรแกรมภาษา หรือ สิ้นสุดของโปรแกรมที่ถูกแปลในการกระจาย ACCT จะมีข้อความ 'NO ACTION' เก็บไว้ด้วยกัน เพราะเน้นการกลับคืนไปหาผู้เรียกเท่านั้น

4 โปรแกรมที่รับผิดชอบต่อการสิ้นสุดของงาน (BENDJOB)

โปรแกรมย่อยนี้จะพิมพ์บรรทัดทั้งหลายที่เกี่ยวข้องกับบัญชีออกมา ซึ่งจะประกอบด้วยจำนวนทั้งหมดของคำสั่งงานที่ถูกกระทำ เวลาที่ใช้การแปล และ การทำงานจริงตามโปรแกรม โดยมีหน่วยเป็นวินาที รวมทั้งข่าวสารเกี่ยวกับที่เก็บข้อมูลที่ใช้ไปโดยโปรแกรมภาษา FORTRAN อันจะประกอบด้วยเนื้อที่ทั้งหมดของที่เก็บข้อมูลที่ใช้เก็บตัวแปรแบบ 1 มิติ ทั้งหลายบล็อคต่างๆที่ใช้เก็บข้อมูลร่วมกัน และตัวแปรทั้งหลายที่เก็บค่าเทียบเท่ากัน (ดังที่เรียกว่า " บริเวณเก็บตัวแปรแบบ 1 มิติ ") นอกจากนี้แล้ว ยังมีการแจ้งให้เจ้าหน้าที่เขียนโปรแกรมทราบ ปริมาณที่เก็บข้อมูลทั้งหมดที่มีไว้ให้ใช้สำหรับการทำงานอันนั้น

จะใช้ค่าต่อไปนี้ในการพิจารณาดูข่าวสารทางบัญชี :

XSTMTCNT - จำนวนคำสั่งทั้งหมดที่ใช้ในการทำงานจริง

BTIME CMP - เวลาที่ใช้แปล

XOBJECT - ตำแหน่งที่อยู่เริ่มต้นของบริเวณทำงาน

XBEGDATA - ตำแหน่งที่อยู่เริ่มต้นของบริเวณที่ใช้เก็บตัวแปรแบบ 1มิติ

CENDARAY - ตำแหน่งที่อยู่สุดท้ายของบริเวณที่ใช้เก็บตัวแปรแบบ 1มิติ

MVAIUESG - ปริมาณเนื้อที่ในหน่วยความจำหลักที่ใช้เป็นบริเวณทำงาน

MVALUESS - ปริมาณเนื้อที่ในหน่วยความจำหลัก ที่ส่งคืนให้โปรแกรมควบคุม

หน่วย ZMONTH จะถูกเรียกมาเพื่อให้ข้อมูลเกี่ยวกับวันที่และเวลาของวัน

5 โปรแกรมย่อยที่รับผิดชอบต่อการสิ้นสุดการทำงานของกลุ่มงาน (BSTOP)
โปรแกรมย่อยนี้จะไม่มีบทบาทการทำงานเลยใน ACCT รุ่นนี้

5 ฟังชั่นก์ของระบบควบคุม

ได้พยายามรวบรวมฟังชั่นก์ทั้งหลายของโปรแกรมควบคุมให้มาอยู่ใกล้ๆกัน และให้การเรียกใช้โปรแกรมต่างๆ โดยโปรแกรมควบคุมจำกัดอยู่ภายในวงไม่กว้างนัก

หน่วยโปรแกรม WATIO, FIOCS และ DIOCS จะรวมตัวเชื่อมสำคัญของการนำข้อมูลเข้า/แสดงข้อมูลออกไปยังโปรแกรมที่มีหน้าที่จัดการของการ นำข้อมูลเข้า/แสดงข้อมูลออกไปยังโปรแกรมที่มีหน้าที่จัดการเกี่ยวกับการดึงเอาข้อมูลมาใช้งาน ที่ใช้กันทั้งแบบเรียงตามลำดับ และแบบโดยตรง

การเรียกอื่นๆ และฟังชั่นก์ทั้งหลายที่อยู่ภายใต้การควบคุมของโปรแกรมควบคุมจะรวมมาไว้ในหน่วยเดียวกันชื่อ SYS. ซึ่งจะถูกรวบรวมเป็นส่วนหนึ่งของ MAIN รายละเอียดสิ้นๆของแต่ละฟังชั่นก์มีดังต่อไปนี้ :

1. การขัดจังหวะของระบบ
โปรแกรมย่อยที่จะเป็นผู้ออกคำสั่งแม็กโคร STXIT ของตัวแปลภาษา และโปรแกรมย่อยที่ควบคุมติดตามดูการขัดจังหวะ ชื่อ ARUPT จะเป็น 2 โปรแกรมย่อยใน SYS รายละเอียดวิธีการจัดการกับการขัดจังหวะดังที่ได้กล่าวมาแล้วในหัวข้อ 3

2. การดึงโปรแกรมย่อยจากชุดโปรแกรมที่รวบรวมไว้แล้ว
โปรแกรมย่อยชุดนี้จะมีหน้าที่นำข้อมูลเข้ามาจากชุดโปรแกรม WATLIB และ

ชุดโปรแกรม WATLIB จะถูกเปิดโดยโปรแกรม JOPENLIB และปิดโดย JCLOSLIB

- โปรแกรมย่อย JFIND จะตรวจค้นหาชื่อที่ถูกระบุมาในรายชื่อชุดโปรแกรม WATLIB มีความยาว 6 ตัวอักษร และ ข้อมูลออกจากการค้นจะเป็นรหัสที่จะระบุว่าได้พบ หรือ ไม่พบชื่อนั้น หรือ เกิดมีข้อผิดพลาดในข้อมูลเข้าขึ้น

- โปรแกรมสุดท้ายในกลุ่มนี้ ชื่อ JSUBRRD มีหน้าที่จัดการดึงกลุ่มโปรแกรมออกมาจากชุดโปรแกรมที่ระบุมา แล้วทำการแยกบล็อคข้อมูล และขยายข้อมูลที่เก็บในแบบอัดแน่น ซึ่งโปรแกรมย่อยที่มีหน้าที่ให้บริการกับชุดโปรแกรมที่รวบรวมไว้เป็นระเบียบแล้วนี้เป็นผู้สร้างขึ้น การเชื่อมโยงเข้ากับโปรแกรมย่อยนี้จะกระทำผ่าน CREAO ใน (COMMR) ซึ่งเป็นโปรแกรมย่อยที่มีหน้าที่อ่านในช่วงการแปล CREAO จะตรวจสอบสวิทช์เพื่อดูว่าข้อมูลนำเข้า จะมาจากชุดโปรแกรมที่รวบรวมไว้เป็นระเบียบแล้วในระบบ หรือมาจากหน่วยนำข้อมูลเข้า ถ้าข้อมูลเข้าที่ต้องการมาจากชุดโปรแกรมที่รวบรวมไว้แล้วในระบบ คำสั่งงานต่อไปนี้จะถูกสั่งออกมาหลังจากได้ทำการค้นหาข้อมูลเข้านั้นพบแล้ว ในชุดโปรแกรมที่รวบรวมไว้แล้วในระบบ (ซึ่งการสั่งนี้ โปรแกรมย่อยที่มีหน้าที่ตรวจค้นในชุดโปรแกรมที่รวบรวมไว้แล้วในระบบชื่อ MLIBR ใน MAIN จะเป็นผู้สั่ง)

```
MVI     CSUBRDS,    ON
MVC     CPOINT (4), CBUFFE
```

คำสั่งงานคำสั่งแรก จะเป็นการตั้งสวิทช์ขึ้นก่อน CREAD

คำสั่งงานที่สอง จะไปบังคับให้โปรแกรมย่อยที่มีหน้าที่อ่านชุดโปรแกรมที่รวบรวมไว้แล้วในระบบ ให้ทำการอ่านข้อมูลเข้ามาในครั้งแรกที่มีการเรียกโปรแกรมย่อยนั้น

หลังจากแปลโปรแกรมประกอบเรียบร้อยแล้ว คำสั่งงาน

```
MVI     CSUBRDS,    OFF
```

จะมีผลไปตั้งสวิทช์ใหม่เพื่อให้รับข้อมูลจากหน่วย EREAD

ถ้าพบเครื่องหมายแสดงการสิ้นสุดของแฟ้มข้อมูลนั้น ก็จะเก็บตัว $ENTRY (หรือเทียบเท่าแล้วแต่ศูนย์คอมพิวเตอร์จะเป็นผู้กำหนดขึ้นมา) ขึ้นมาในบริเวณเก็บข้อมูลเข้าของตัวแปลภาษา ซึ่งจะถูกใช้โดยโปรแกรมย่อย SCAN เพื่อกลับคืนไปยังโปรแกรมย่อย ที่มีหน้าที่ดำเนินการวิธีกับชุดโปรแกรมในระบบที่มีรวบรวมไว้เป็นระเบียบแล้ว

ถ้าข้อมูลนำเข้าเกิดมีข้อผิดพลาดขึ้น การประมวลผลโปรแกรมที่กำลังดำเนินการอยู่ในขณะนั้นก็จะยุติลง และตัวแปลภาษาก็จะดำเนินการแปลโปรแกรมอันดับต่อไปในกลุ่มนั้น

ข้อสังเกต :

แต่ละส่วนในชุดโปรแกรมภาษาต้นฉบับที่มีรวบรวมไว้ อาจจะตามหรือไม่ตามด้วยบัตร ($ ENTRY) (หรือเทียบเท่า แล้วแต่ศูนย์คอมพิวเตอร์นั้นๆ) ก็ได้ แต่ถ้าตามด้วย ก็จะเป็นการช่วยเร่งการประมวลผลชุดโปรแกรมนั้นให้เร็วขึ้น เพราะว่า จะข้ามการอ่านเครื่องหมายแสดงการสิ้นสุดของแฟ้มข้อมูลไปได้เลย และข้ามการประมวลผลที่เกี่ยวข้องทั้งหมดไปด้วย

3. โปรแกรมย่อยที่มีหน้าที่เกี่ยวกับเวลา

จะใช้โปรแกรมย่อย 2 อัน คือ JSTIMER และ JTCANCEL เพื่อตั้งและเลิกล้มตัวบอกเวลาเป็นช่วงๆ WATFIV จะยอมให้ใช้เวลาแปลได้นานที่สุด 15 ช.ม. และเวลาที่ระบุมาในบัตร $JOB (หรือถ้าไม่มีระบุ ก็ให้ใช้ค่าอัตในมัติที่ทางศูนย์คอมพิวเตอร์กำหนดให้) เป็นเวลาที่กำหนดสำหรับใช้ในช่วงการทำงานจริงตามคำสั่งในโปรแกรม โปรแกรมย่อยที่มีหน้าที่ขัดจังหวะเครื่องจับเวลาชื่อ YTIMINT จะเป็นผู้ตั้งสวิทช์อันจะแสดงถึงการได้เกิดการขัดจังหวะเครื่องจับเวลาขึ้น โดยรหัสภาษาเครื่องอันแรกสุดของทุกๆคำสั่งทำงานจะเป็นผู้ทดสอบดูสวิทช์อันนั้น โดยเฉพาะเจาะจงลงไปเลยรหัสภาษาเครื่องอันแรกสุดที่เกิดขึ้นสำหรับทุกๆคำสั่งทำงาน ก็คือ

```
BALR        R11, R12
DC          AL2 (ISN)
```

ตัวอย่าง ถ้าคำสั่งงาน I = J อยู่ในบรรทัดที่ (หรือ ISN) 5 ใน โปรแกรมอันหนึ่ง รหัสภาษาเครื่องของคำสั่งงานนี้จะเป็น

```
BALR        R11, R12
DC          RL2 (5)
L           RO, I
ST          RO, J
```

ในการปฏิบัติงานของ WATFIV จะใช้ R12 เก็บตำแหน่งที่อยู่ของโปรแกรมย่อยชื่อ XISNRTN ซึ่งจะเป็นผู้ทดสอบสวิทช์ทั้งหลายที่เกี่ยวข้องกับการขัดจังหวะเครื่องจับเวลา โปรแกรมย่อยทั้ง 2 อัน คือ YTIMENT และ XISNRTN นี้จะอยู่ตอนต้นของ CSUT STARTA ซึ่งจะถูกลอกลงไปเป็นส่วนหนึ่งของ MAIN โดยคำสั่งแม็กโคร FUNCREFS .

4. โปรแกรมย่อยที่จัดการกับหน่วยความจำหลักแบบวงแหวนแม่เหล็ก

ตัวแปลภาษาจะใช้โปรแกรมย่อย 2 อัน คือ JGET และ JFREE เพื่อให้ได้มาซึ่งเนื้อที่ทำงานในหน่วยเก็บข้อมูล และเพื่อปล่อยเนื้อที่ทำงานเป็นอิสระ ข้อมูลนำเข้าจริงๆที่ต้องการ และผลลัพธ์ที่ได้ของมัน จะมีอธิบายอยู่ในโปรแกรมย่อยเหล่านี้

6 โปรแกรมประกอบ

องค์ประกอบของโปรแกรมที่ครบสมบูรณ์ที่ทำงานอยู่ภายใต้ดอสฟ จะได้มาจาก 3 แหล่งคือ :

1 กลุ่มบัตรจากสำดับชุดข้อมูลนำเข้า (SYSIN)

2 โปรแกรมย่อยทั้งหลายที่เก็บอยู่ในหน่วยความจำหลัก ในลักษณะเป็นส่วนหนึ่งของตัวแปลภาษา

3 กลุ่มบัตรที่ดึงมาจากชุดโปรแกรม ที่มีเก็บรวบรวมไว้แล้วในระบบประเภทที่สามารถเข้าถึงได้โดยตรง

กลุ่มบัตรประเภทแรก และ ประเภทที่ 3 อาจจะเป็นโปรแกรมภาษาฟอร์แทรน หรือ เป็นโปรแกรมย่อยภาษาเครื่องก็ได้

หัวข้อต้นๆ ได้กล่าวถึงวิธีการรวมโปรแกรมย่อยต่างๆที่ถูกเลือกมาเข้าไปเป็นส่วนหนึ่งของหน่วยตัวแปลภาษาประเภทที่เก็บไว้ในหน่วยความจำหลัก ต่อไปจะเป็นการพูดถึงรายละเอียดข้อกำหนดของภาษาสำหรับโปรแกรมภาษาฟอร์แทรน หัวข้อนี้จะพูดถึงกฎเกณฑ์ต่างๆ ที่ใช้ในการสร้าง และการใช้ชุดโปรแกรม ที่เก็บรวบรวมไว้แล้วในระบบแบบที่เข้าถึงได้โดยตรงเลย และ กฎเกณฑ์ของการเขียนโปรแกรมย่อยภาษาแอลเซมบลี

6.1 การสร้างชุดโปรแกรมแบบเข้าถึงได้โดยตรงเลย

โปรแกรมย่อยที่ WATFIV จะสามารถเรียกออกมาใช้งานได้โดยตรงเลย จะต้องถูกเก็บไว้เป็นส่วนๆในชุดโปรแกรมย่อยของชุดโปรแกรมคำสั่งต้นฉบับที่รวบรวมไว้ทั้งส่วนบุคคล และ/หรือ ของระบบโดยชื่อของส่วน กับชื่อของโปรแกรมประกอบประเภทฟังชั่นก์ หรือ ซับรูทีนที่เก็บจะต้องเป็นชื่อเดียวกัน ถ้าโปรแกรมประกอบใดมีจุดเข้าหลายจุด (ซึ่งกำหนดจากคำสั่ง ENTRY) ก็จะต้องมีการสร้างชื่อแฝงสำหรับจุดเข้าแต่ละจุด ทั้งนี้เพื่อให้สามารถดึงโปรแกรมย่อยนั้นมาใช้งานได้ เมื่อมีการอ้างอิงถึงไม่ว่าจะเป็นการอ้างอิงถึงไม่ว่าจะเป็นการอ้างอิงชื่อธรรมดา หรือชื่อจุดเข้าหลายๆชื่อนั้น ชื่อแฝงก็จะเป็นอีกส่วนหนึ่งต่างหาก ซึ่งชื่อของส่วนนี้ก็จะเป็นชื่อแฝง และจะประกอบขึ้นด้วยคำสั่งงาน คำสั่งงานซึ่งอาจมีรูปแบบได้ดังต่อไปนี้

คอลัมภ์ 1 – รูเจาะควบ 12-0-8 (X '88') จะเป็นการบอกให้โปรแกรมย่อยที่มีหน้าที่อ่านข้อมูลเข้า มาจากชุดโปรแกรมของ WATFIV ทราบถึงชื่อแฝง

คอลัมภ์ 2-7 – จะเป็นชื่อของส่วนซึ่งใช้ชื่อแฝงนี้ โดยชื่อนั้นจะมีความยาว 1-6 ตัวอักษร และเจาะชิดซ้าย

คอลัมภ์ 8 – ไม่มีเจาะ เว้นว่างไว้

คอลัมภ์ 9-80 – ไม่ใช้ ; อาจใช้ใส่หมายเหตุ

ตัวอย่างเช่น

1. ถ้าเราต้องการที่จะใช้ ENTRYP เป็นชื่อแฝงของโปรแกรมย่อย SUBRTN เราจะต้องนำส่งบัตรคำสั่งต่อไปนี้ให้กับ MAINT :

```
//EXEC      MAINT
  CATELS    W. ENTRYP
  BKEND     W. ENTRYP
 #SUBRTN    [ตามด้วยหมายเหตุต่างๆหรือไม่มีการใช้...]
  BKEND
/ *
```

โดยที่ '#' แทน รูเจาะควบ 12-0-8

นอกจากนี้ยังสามารถจะใช้โปรแกรมช่วยงานต่างๆของ DOS และ DOS/VS อันได้แก่ MAINT, SSERV และ CORGZ ในการสร้าง และดูแลรักษาชุดโปรแกรมในช่วงการทำงานจริงตามคำสั่งในโปรแกรมของ WATFIV

ข้อพึงสังเกต :

1. ในกรณีที่มีการเรียกโปรแกรมประกอบ B จากโปรแกรมประกอบ A และทั้ง A และ B ต่างก็ถูกเก็บอยู่ในชุดโปรแกรมที่รวบรวมไว้เป็นระเบียบแล้ว ซึ่งเป็นไปได้ที่จะเก็บกลุ่ม A และ B ไว้ในส่วนเดียวกัน ดังนั้นไม่จำเป็นที่ B จะต้องมีชื่อแฝง เพราะว่าในการดึงเอาส่วน A มาใช้งาน กลุ่มโปรแกรม B ก็จะถูกดึงออกมาด้วย การกระทำเช่นนี้จะช่วยเร่งการดึงเอาโปรแกรมในชุดโปรแกรมมาใช้งานให้เป็นไปได้เร็วขึ้น เนื่องจากว่าไม่ต้องมีการไปค้นหาดูในรายชื่อของชุดโปรแกรมเพื่อจะดูว่า B เก็บอยู่ที่ไหนอีก
2. บัตรใบสุดท้ายของแต่ละส่วน อาจจะเป็นบัตร ENTRY (หรือเทียบเท่า แล้วแต่ศูนย์คอมพิวเตอร์นั้นๆจะกำหนด) โปรแกรมย่อยที่มีหน้าที่อ่านข้อมูลเข้ามาจากชุดโปรแกรม จะใช้บัตรใบนี้เป็นสัญญาณให้ยุติ

การดึงส่วนที่กำลังกระทำอยู่ขณะนั้นมาใช้งาน การเรียกชุดโปรแกรมมาใช้งานจะเร็วขึ้นเพราะไม่ต้องมีการอ่านเครื่องหมายแสดงการสิ้นสุดของแฟ้มข้อมูล และไม่ต้องมีการดำเนินการรวมวิธีเกี่ยวข้องกับการสิ้นสุดแฟ้มข้อมูล

3. โปรแกรมย่อยที่มีหน้าที่อัดรวมคำสั่งงานหลายๆคำสั่งในกลุ่มโปรแกรมให้มาอยู่บนบัตรใบเดียวกันจะกระจายอยู่ในตัวแปลภาษา *ให้ดูเกี่ยวกับการเขียนภายใต้หัวข้อ "โปรแกรมย่อยที่มีหน้าที่อัดรวม/คล้ายการรวมคำสั่งงานต่างๆ" ในคู่มือประกอบการใช้ ของ WATFIV* ข้อดีก็คือสามารถจะประหยัดเนื้อที่เก็บในข้อมูลแบบใดยได้ และเรียกกลุ่มที่มีการอัดรวมไว้มาใช้งานก็เป็นไปได้เร็วขึ้น

4. ถ้าชุดโปรแกรมช่วงการทำงานจริงตามคำสั่งในโปรแกรม ของ WATFIV ใช้ชุดโปรแกรมคำสั่งภาษาต้นฉบับที่รวบรวมไว้ส่วนบุคคล มันจะถูกเชื่อมต่อเข้ากับชุดโปรแกรมคำสั่งภาษาต้นฉบับ ของระบบในลักษณะปาติ นั่นก็คือการค้นหาโปรแกรมจะกระทำก่อนในชุดโปรแกรมส่วนบุคคล ถ้าไม่มีการมอบหมายกำหนดมาให้ใช้ชุดโปรแกรมส่วนบุคคล ก็จะถือเอาว่า ชุดโปรแกรมช่วงการทำงานจริงตามคำสั่งในโปรแกรมของ WATFIV ใช้ชุดโปรแกรมคำสั่งภาษาต้นฉบับของระบบ

5. ถ้าจะมีการใช้ชุดโปรแกรมคำสั่งภาษาต้นฉบับส่วนบุคคล ก็ต้องมีคำสั่งควบคุมการทำงาน "// DLBL IJSYSSL" ที่เหมาะสมรวมอยู่ในลำดับงานที่ใช้เรียกกระตุ้น WATFIV หรือเก็บอยู่ในบริเวณเก็บชื่อที่ตั้งมาตรฐาน SYSRES โดยใช้ข้อเลือก STDLBL

สุดท้ายนี้ ควรจะมีการกล่าวถึงผลที่อาจเกิดจากการที่ชื่อโปรแกรมประกอบที่ผู้ใช้กำหนดเป็นชื่อเดียวกับโปรแกรมย่อยที่ฟอร์แทรนมีให้ ซึ่งเก็บอยู่ใน (ชุดโปรแกรม WATFIV.

FUNLIB ในช่วงการทำงานจริงตามคำสั่งในโปรแกรม) ตัวอย่างเช่นให้พิจารณาโปรแกรมประกอบต่อไปนี้ ซึ่งผู้ใช้ได้กำหนดฟังชั่นก์ SQRT ของเขาขึ้นมาเองในลักษณะที่มีการเรียกใช้ฟังชั่นก์ DSQRT ซึ่งฟอร์แทรนมีไว้ให้

```
      REAL  FUNCTION  SQRT X 8 (X)
      REAL X 8 X
      SQRT   =   DSQRT (X)
      RETURN
      END
```

กรณีที่ 1

สมมุติว่า

(1) โปรแกรมประกอบข้างต้นนี้ถูกอ่านเข้ามาทาง SYSIN

(2) WATFIV ในลักษณะที่มี DSQRT เก็บไว้ในหน่วยความจำหลัก จะถูกทำให้เกิดขึ้นมา

(3) อาจจะมีการระบุเอ่ยถึง หรือไม่มีการเอ่ยถึงชุดโปรแกรมที่มีรวบรวมไว้แล้วก็ได้ ดังนั้นแล้ว DQRT ที่ผู้ใช้เขียนขึ้น และ DSGRT ที่มีเก็บอยู่ในหน่วยความจำหลักแบบวงแหวนแม่เหล็กจะถูกใช้

กรณีที่ 2

สมมุติว่า

(1) โปรแกรมประกอบข้างต้นถูกอ่านเข้ามาโดย SYSIN

(2) ไม่ได้เก็บ DSGRT ไว้ในหน่วยความจำหลักแบบวงแหวนแม่เหล็ก แต่เก็บอยู่ใน (ชุดโปรแกรมในช่วงทำงานจริงตามโปรแกรมที่รวบรวม

ไว้) WATFIV.FUNLIB ซึ่งได้ระบุชุดโปรแกรมไว้แล้วหรือระบุเป็นชุดโปรแกรมแรกจะได้ผลลัพธ์ตามที่ต้องการ แต่ตัวแปลภาษาจะออกข้อความเตือนเกี่ยวกับจุดเข้า SORT ว่ามิได้ถูกกำหนดขึ้นมากกว่า 1 ทาง อันเป็นผลเนื่องมาจากว่าส่วนของ (ชุดโปรแกรม WATFIV.FUNLIB ในช่วงการทำงานจริงตามโปรแกรม) นอกจากจะเก็บโปรแกรมย่อย DSQRT แล้ว ยังเก็บโปรแกรมย่อย SQRT ที่ตัวแปลภาษามิไว้ให้ และทั้ง 2 โปรแกรมย่อยจะถูกดึงออกมาใช้งาน เราอาจจะละเลยต่อข้อความเตือนก็ได้ เนื่องจากว่าโปรแกรมย่อยอันแรก อันเป็นโปรแกรมย่อยที่ผู้ใช้เขียนขึ้นจะถูกนำมาใช้งาน

6.2 การสร้างสมาชิกในชุดโปรแกรมตอนช่วงการทำงานจริงตามคำสั่งในโปรแกรม

เนื้อหาของหัวข้อนี้จะมีความสำคัญ ต่อเจ้าหน้าที่เขียนโปรแกรมระบบของคูนย์คอมพิวเตอร์นั้นๆที่มีเจ้าหน้าที่คอยปรับปรุงแก้ไขสมาชิกต่างๆของชุดโปรแกรมตอนช่วงทำงานจริง โดยเฉพาะโปรแกรมย่อย FIOCS#, DIOCS#, IBCOM# และ UNIT@

มี 2 ขั้นตอนด้วยกันคือ

- ทำการรวบรวมโปรแกรมส่วนที่ต้องการจากชุดโปรแกรม WATFIV .SOURCE เพื่อแปลงเป็นรหัสภาษาเครื่อง
- ปรับปรุงแก้ไขชุดโปรแกรม WATFIV.FUNLIB โดยการแทนที่ส่วนที่ถูกแก้ไขเปลี่ยนแปลงไปด้วยกลุ่มรหัสภาษาเครื่องในขั้นตอนที่ 1 (นอกจากนี้การปรับปรุงแก้ไข จะต้องทำการเปลี่ยนชื่อแฝงต่างๆของส่วนนั้นในชุดโปรแกรม WATFIV.FUNLIB)

ตัวอย่าง : ในการสร้างโปรแกรมซึ่งมีจุดเข้าเป็น DARSIN, DARCOS, ARSIN และ ARCOS ให้ใช้งาน

```
//osjob   JOB   accounting
//    EXEC   WASM,NAME=DARSIN,PARM.ASM='DECK,NOLOAD',
//           LKED=IEFBR14
//ASM.SYSPUNCH  DD  SYSOUT=B
```

ใช้รหัสภาษาเครื่องทำงาน

```
//osjob    JOB   accounting
//    EXEC      PGM=IEBUPDTE
//SYSPRINT    DD  SYSOUT=A
//SYSUT1      DD  DSN=WATFIV.FUNLIB,DISP=OLD,VOL=SER=
//                volume,UNIT=disk
//SYSUT2      DD  DSN=WATFIV.FUNLIB,DISP=OLD,VOL=SER=
//                volume,UNIT=disk
//SYSIN       DD  *
./ REPL NAME=DARSIN


object deck from lst job


./ ALIAS NAME=DARCOS
./ ALIAS NAME=ARSIN
./ ALIAS NAME=ARCOS
./ ENDUP
/*
```

ให้ใช้ตารางในรูปที่ 3.6 เป็นแนวทางสำหรับการรวบรวมสมาชิกอื่นๆ โดยใช้ชื่อแรกสุดที่ปรากฏบนบรรทัดใดๆ เป็นชื่อของส่วนโปรแกรมใน WATFIV.SOURCE ชื่ออื่นๆที่ปรากฏอยู่ในบรรทัดเดียวกันจะเป็นชื่อแฝงของส่วนที่มีชื่อเดียวกัน ซึ่งอยู่ใน WATFIV.FUNLIB

6.3 การเขียนโปรแกรมประกอบภาษาแอสเซ็มบลี

กฎเกณฑ์ที่จะต้องใช้ในการเขียนโปรแกรมประกอบ (SUBPROGRAM) ภาษาแอสเซ็มบลีเพื่อมาใช้กับ WATFIV กฎเกณฑ์นี้จะคล้ายคลึงกับที่ตัวแปลภาษาฟอร์แทรนของเครื่อง IBM ใช้ จะใช้เครื่องหมายในรูปของสัญญลักษณ์แทนรีจิสเตอร์ทั้งหลาย ตลอดคำอธิบายต่อไปนี้

1. ลำดับการเรียกโปรแกรมประกอบ

สมมุติว่าคำสั่ง CALL หรือคำสั่งอ้างอิงถึงฟังชั่นก์ในคำสั่งงานภาษาฟอร์แทรน มีการอ้างถึงโปรแกรมประกอบแบบฟังชั่นก์หรือ แบบซับรูทีน ชื่อว่า 'rtn' ดังตัวอย่าง

(i) CALL rtn (arg1,arg2,arg3,...,argn)

(ii) Y=rtn (arg1,arg2,arg3,...,argn)

ลำดับการเรียกที่ถูกสร้างขึ้นโดย WATFIV สำหรับทั้ง 2 กรณีข้างต้นเป็นดังนี้

```
                        CNOP        2,4
                        LA          R14,return
                        L           R15,=V(rtn)
                        BALR        R1,R15
                       -DC          AL1(c1) ,AL3(addr1)
                       ¦ DC         AL1(c2) ,AL3(addr2)
                       ¦ .
argument               < .
list                   ¦ .
                       ¦
                       ¦
                       -DC          AL1(cm) ,AL3(addrm)
return                  EQU         *
```

ค่าชั่วคราว, พารามิเตอร์ของคำสั่ง DO, ดัชนีทั้งหลายของคำสั่ง GOTO ส่วน 4 บิท 'mmmm' จะบอกประเภทของตัวแปร ซึ่งเป็นไปตามตารางรูป 3.7

สำหรับประเภทที่มีหมายเลข 0-7 , AL3(addri) = AL3(Q) ;

สำหรับประเภทที่มีหมายเลข 8,9 , AL3(addri) = AL3(Q*)

โดยที่ Q* เป็นค่าเต็มที่มีรูปแบบ DC AL1(n) , AL3(Q)

Type	TYPE NUMBER	mmmm	s-VALUE
Logical*4	0	0000	2
Logical*1	1	0001	0
Interger*4	2	0010	2
Interger*2	3	0011	1
Real*4	4	0100	2
Real*8	5	0101	3
Complex*8	6	0110	3
Complex*16	7	0111	4
Character*n n=1	8	1000	0
Character*n n>1	9	1001	0

รูป 3.5 ตารางบิทแสดงรหัสประเภท

(ข) ตัวแปรต่างๆ, ข้อมูลที่เก็บในตัวแปรแบบ 1 มิติ - V: Ci='B1000 mmmm' สำหรับกรณีนี้ AL3(addri) = AL3(V) และ 'mmmm' ดูจากตารางรูป 3.11 ถ้าข้อมูลของตัวแปรเป็นข้อมูลที่เก็บในตัวแปรแบบ 1 มิติ ก็จะต้องมีค่าเพิ่ม 1 ค่าตามมาในรายชื่อตัวแปรเพื่อเป็นดัชนีพิเศษ อันจะมีรูปแบบดังนี้

DC X '8C', AL3 (V*)

โดยที่ V* เป็นที่เรียกว่า โปรแกรมย่อย STAR สำหรับตัวแปร 1 มิติ ซึ่งค่าของข้อมูลนั้นเป็นสมาชิกอยู่ ต่อไปนี้จะเป็นการอธิบายถึงโปรแกรมย่อย STAR

ตัวอย่างเช่น : ถ้าใช้ A(5) เป็นค่าตัวแปรในโปรแกรมประกอบ ค่าที่ตรงกันในรายชื่อตัวแปร จะต้องมีลักษณะดังต่อไปนี้

DC B'10000100',AL3 (A1+16), X'8C', AL3(A*)

ในขณะที่ใช้สัญลักษณ์ A1 แทน A(1) (สมมุติว่า A เป็นตัวแปรประเภท REAL*4)

(ค) ชื่อตัวแปรแบบ 1 มิติ - A: Ci = B'1kkkmmmm'

สำหรับกรณีนี้ kkk เป็นจำนวนมิติของ A และ AL3(addri) =AL3 (A)

(ง) ถ้า R เป็นซับรูทีน Ci = B'01010000'

แต่ถ้า R เป็นฟังชั่นก์ Ci = B'0110mmmm'

ทั้ง 2 กรณีนี้ AL3(addri) = AL3(R*) ในขณะที่ R* ก็คือ DC A(R)

(จ) เลขหมายคำสั่ง En : Ci = B'00110000'

สำหรับกรณีนี้ AL3 (addri) = AL3(n*) ในขณะที่ n เป็นตำแหน่งที่อยู่ของคำสั่งงานที่มีหมายเลขเท่ากับ n

(ฉ) เครื่องหมายแสดงการสิ้นสุดรายชื่อตัวแปร

ตัวนี้จะเป็นข้อมูลพิเศษที่มีไว้เพื่อ แสดงการสิ้นสุดของรายชื่อตัวแปร นอกจากนั้นแล้ว ข้อมูลตัวนี้ยังจะบอกให้ทราบถึงลักษณะของโปรแกรมย่อยที่ถูกเรียกด้วย ถ้าโปรแกรมย่อยที่ถูกเรียกเป็นประเภทซับรูทีน, C = B'00010000' แต่ถ้าโปรแกรมย่อยที่ถูกเรียกเป็นประเภทฟังชั่นก์, C = B'0010mmmm'

2. โปรแกรมย่อย STAR สำหรับตัวแปรแบบ 1 มิติ

การอ้างอิงทั้งหมดที่มีไปยังตัวแปรแบบ 1 มิติ (สมมุติว่าเป็น X) ในโปรแกรมที่ WATFIV. แปล จะต้องอาศัยการทดสอบดัชนีกำกับตัวแปร และโปรแกรมย่อยสำหรับหาตำแหน่งที่อยู่ (หรือโปรแกรมย่อย STAR) ของ X ซึ่ง X* ตัวแปลภาษาจะเป็นผู้สร้างโปรแกรมย่อย STAR ให้กับตัวแปรแบบ 1 มิติ ทั้งหมดที่บอกมาให้ทราบในโปรแกรมภาษาฟอร์แทรน โปรแกรมย่อย STAR แต่ละอันจะเก็บข้อมูลที่เกี่ยวกับตัวแปรแบบ 1 มิตินั้นๆ (เช่น จำนวนมิติของมัน, ตำแหน่งที่อยู่เริ่มต้น, ความยาวรวมทั้งหมด) และจะถูกใช้ในตอนช่วงทำงานจริง เพื่อจะได้เข้าไปใช้ข้อมูลของตัวแปรแบบ 1 มิตินั้นได้

- จะตรวจสอบดัชนีกำกับที่มีค่าเกินพิสัย
- จะส่งผ่านตัวแปรแบบ 1 มิติ ไปให้กับโปรแกรมประกอบทั้งหลาย

จำเป็นที่จะต้องทราบรูปแบบของโปรแกรมย่อย STAR ในกรณีที่โปรแกรมประกอบภาษาแอสเซ็มบลีจะต้องมีการรับค่า (หรือส่งผ่านค่า) ตัวแปรแบบ 1 มิติ มาจาก (ไปให้กับ) โปรแกรมย่อยภาษาฟอร์แทรน แต่สำหรับจุดมุ่งหมายของเราก็เป็นการเพียงพอแล้วที่จะพิจารณาดูโครงประกอบของโปรแกรมย่อย STAR (สมมุติว่าเป็น X*) ที่มีรูปแบบดังต่อไปนี้ :-

```
        DS    0F
X*      EQU   *-4
        DC    AL1(f) , AL3 (ข้อมูลตัวแรกที่เก็บในตัวแปรแบบ 1 มิติ)
        DC    AL1(s) , AL3 (ความยาวเป็นไบท์ของตัวแปรแบบ 1มิติ)
```

สำหรับกรณีนี้ f = 4k-4 โดย k เป็นจำนวนมิติ

s = s - ค่าตามประเภทของตัวแปรแบบ 1 มิติ (ให้ดูตารางรูป 3.11) ในโปรแกรมย่อย STAR ที่ถูกสร้างขึ้นมาให้ส่งผ่านตัวแปรแบบ 1 มิติ ไปให้กับโปรแกรมประกอบภาษาฟอร์แทรน อาจจะกำหนดค่าให้กับ f และ s เป็นศูนย์ โปรแกรมส่วนนำของโปรแกรมภาษาฟอร์แทรนจะอ้างถึง AL3 adcon เพียง 2 ตัวเท่านั้น เมื่อเริ่มกำหนดโปรแกรมย่อย STAR ให้กับตัวแปรจำลองแบบ 1 มิติ (dummy array) ซึ่งได้รับค่าส่งมาจากตัวแปรแบบ 1 มิติที่เรียกรูปแบบเต็มของโปรแกรมย่อย STAR ที่ตัวแปลภาษาเป็นผู้สร้างขึ้นมาจะมีอยู่ในหัวข้อย่อยต่อไป

3. กฎเกณฑ์อื่นๆของโปรแกรมประกอบ

(ก) โปรแกรมที่มีหน้าที่ป้อนกลุ่มรหัสภาษาเครื่องของ WATFIV จะใช้ตัวอักษร 6 ตัวแรกเท่านั้น ของชื่อ CSECT, ENTRY, EXTRN ชื่อที่มีความยาวเกิน 6 ตัวอักษรก็จะถูกตัดเหลือเพียง 6 ตัวอักษรเท่านั้น ชื่อต่างๆของจุดเข้าทั้งหลาย และ CSECT จะต้องไม่ซ้ำกัน นั่นก็คือตัวอักษร 6 ตัวแรกของชื่อจุดเข้าในกลุ่มโปรแกรมเดียวกันไม่ได้

(ข) อาจจะอ้างถึง COMMON แบบไม่มีการตั้งชื่อโดยใช้ COM ซึ่งเป็นรูปลักษณะหนึ่งของภาษาแอสเซ็มบลี ในการอ้างถึง COMMON ที่มีการตั้งชื่อให้ จะให้ค่าคงที่ของตำแหน่งที่อยู่ประเภท V ชื่อ DC V (ชื่อของ COMMON)

(ค) โปรแกรมประกอบภาษาแอสเซ็มบลีอาจจะใช้ CXD และ DXD อันเป็นรูปลักษณะของภาษาแอสเซ็มบลีได้

(ง) ฟังชั่นก์ตรรกวิทยาจะส่งคืนค่ากลับมาเก็บ ในตำแหน่งที่อยู่ค่าต่ำของ R0 โดย ถ้าเป็นค่าจริง (.TRUE.) จะตรงกับ X 'FF'

และ ค่าเท็จ (.FALSE.) จะตรงกับ X'00'

WATFIV จะเก็บค่าของตัวแปรตรรกวิทยาไว้ในตำแหน่งที่อยู่ค่ามากของ R0

(จ) ในการจะจำลองคำสั่งกลับหลายๆแห่ง (RETURN i) โปรแกรมประกอบที่ถูกเรียกจะต้องไปหาดูตัวแปรที่เป็นหมายเลขคำสั่งต่อที่ i ในรายชื่อตัวแปร ตำแหน่งที่อยู่ที่ควรจะส่งคืนกลับการควบคุม สามารถจะพิจารณาได้จากข้อมูลในรายชื่อตัวแปรนี้

(ฉ) ในการเรียกโปรแกรมประกอบภาษาฟอร์แทรนจากโปรแกรมประกอบภาษาแอสเซ็มบลิให้กระทำดังต่อไปนี้ :-

- จำลองการเรียกที่ WATFIV เป็นผู้ทำขึ้นดังกล่าวถึงในหัวข้อย่อย 6.3.1 ซึ่งการกระทำดังกล่าวนี้จะทำให้รายชื่อตัวแปรและโครงประกอบโปรแกรมย่อย STAR ทั้งหลายถูกสร้างขึ้นมาอย่างถูกต้อง สำหรับตำแหน่งที่อยู่บริเวณที่ใช้เก็บค่าเดิมต่างๆก่อนการเรียกจะเก็บไว้ใน R13

- ส่งผ่านค่าต่างๆที่เก็บใน R12 ที่ถูกส่งมาให้กับโปรแกรมประกอบภาษาแอสเซ็มบลิจากโปรแกรมย่อยภาษาฟอร์แทรนที่อยู่ในลำดับขั้นที่สูงกว่า

- ตรวจดูให้แน่ว่า F6 มีค่าเป็นศูนย์

- หลังจากกลับคืนมาจากโปรแกรมประกอบภาษาฟอร์แทรน ให้นำค่าเดิมของโปรแกรมประกอบภาษาแอสเซ็มเบลอร์ ที่เก็บอยู่ในรีจิสเตอร์ต่างๆคือ R0 ถึง R4, R12, R15 ก่อนที่จะมีการเรียกโปรแกรมประกอบภาษาฟอร์แทรนกลับมาเก็บไว้ในที่เดิมตามต้องการ (โดยค่าเดิมเหล่านี้เก็บไว้ในบริเวณที่ใช้เก็บโดยเฉพาะ) epilogue ของโปรแกรมประกอบภาษาฟอร์แทรนจะนำค่าเดิมกลับมาเก็บใน R5-R11, R13-R14 เท่านั้น (ส่วนค่าของฟังชั่นก์จะคืนกลับมาใน R0, F0 หรือ F0-F2) ให้ระมัดระวังการใช้รีจิสเตอร์ R12 และ F6 เป็นพิเศษ เพราะ WATFIV จะถือว่าค่าในรีจิสเตอร์ 2 ตัวนี้คงที่ตลอดการทำงานจริง โดยจะ

ใช้ F6 ในการเปลี่ยนแปรค่าที่มีความเที่ยงตรงเท่าเดียวมาเป็นค่าที่มีความเที่ยงตรงเป็น 2 เท่า ส่วน R12 จะใช้เป็นรีจิสเตอร์ฐานสำหรับหลายๆโปรแกรมย่อย ภายในที่ใช้ในช่วงการทำงานจริงที่เก็บอยู่ใน STARTA ของตัวแปลภาษา

(ช) ถ้าต้องการให้ข้อความเกี่ยวกับข้อผิดพลาดที่เกิดขึ้น แสดงข้อมูลย้อนหลัง และชื่อของโปรแกรมประกอบด้วย ให้ใช้คำสั่ง :-

```
$ ERROR (NOAC, XX, ENTreg), XRETRACE
```

โดยที่ XX, เป็นหลายเลขข้อความเกี่ยวกับข้อผิดพลาดที่เกิดขึ้น reg เป็นรีจิสเตอร์ที่ชี้ไปยังชื่อของซับรูทีน

ตัวอย่างเช่น :- ในการจะให้ SR-4 ปรากฏในซับรูทีน SUBPRO ให้ใช้:

```
# ERROR (NOAC, SR, 4, ENTR11), XRETRACE
```

โดยที่ก่อนหน้านี้ในโปรแกรมประกอบ R11 จะต้องป้อนต่อไปนี้เข้าไปเก็บใน

```
LA  R11,NAME
```

และ NAME ถูกกำหนดว่า

```
NAME  DS  0H
      DC  H '0'
      DC  CL6 'SUBPRO'
```

ต่อไปนี้เป็นตัวอย่างโปรแกรม WATFIV เพื่อจะแสดงให้เห็นถึงหลักเกณฑ์บางอันที่ใช้ เมื่อมีการเขียนโปรแกรมประกอบเป็นภาษาแอสเซ็มเบลอร์ เพื่อมาใช้งานกับ WATFIV

โปรแกรมหลักจะเรียกโปรแกรมประกอบภาษาแอสเซ็มเบลอร์ RTN และ RTN ก็จะเรียกโปรแกรมประกอบที่เขียนโดย WATFIV ชื่อ NEXT RTN จะส่งผ่านค่า 12-25 ไปให้กับ XX เพื่อใช้ใน NEXT NEXT จะกำหนดค่าเริ่มต้นให้กับตัวแปรแบบ 1 มิติ ชื่อ B และตัวแปร Y และก็จะส่งคืนการควบคุมกลับมาให้กับ RTN RTN ก็จะส่งผ่านต่อค่ามาให้กับ X ในโปรแกรมหลัก

```
      COMMON I
      DIMENSION A (10)
      CALL RTN (A,X)
      PRINT,A,X,I
      STOP
      END
      SUBROUTINE NEXT (B,Y)
      COMMON J
      DIMENSION B (5)
      PRINT, 'HELLO FROM NEXT',' Y=',Y
      DO 1 J=1,5
    1 B (J)=J
      Y=-17.5
      PRINT, 'GOOD-BYE FROM NEXT',' Y=',Y
      RETURN
      END
```

```
      START
      ENTRY   RTN
      USING   RTN,15
RTN   STM     14,12,12(13)    เก็บค่าเดิมที่เก็บอยู่ในรีจิสเตอร์ที่โปรแกรมผู้เรียกใช้
      LR      2,13            ตำแหน่งที่อยู่บริเวณที่ใช้เก็บค่าเดิมต่างๆของโปรแกรม
                              ผู้เรียก
```

```
         LA      13,SAVE         บริเวณที่เก็บอันใหม่
         ST      13,8 (2)        เชื่อม 2 บริเวณที่เก็บเข้าด้วยกัน
         ST      2,SAVE+4        บริเวณทั้งหลายที่ใช้เก็บ
         L       2,0 (1)         ตำแหน่งที่อยู่ของโปรแกรมย่อย 'STAR' ของ A
         L       3,4 (2)         ตำแหน่งที่อยู่ของข้อมูลตัวแรกของ 'A'
         LA      3,0 (3)         กำจัดไบท์ที่เป็นรหัส
         ST      3,ASTAR+4       เก็บไว้ในโปรแกรมย่อยจำลอง 'STAR'
         L       3,8 (2)         ความยาวของตัวแปรแบบ 1 มิติ 'A'
         ST      3,ASTAR+8       เก็บในโปรแกรมย่อยจำลอง 'STAR'
         L       2,4 (1)         ตำแหน่งที่อยู่ของตัวแปรที่สอง 'X'
         ST      2,XADDR         เก็บไว้สำหรับภายหลัง
         LA      1,ARGLIST       ตำแหน่งที่อยู่ของรายชื่อตัวแปรสำหรับการเรียก
         L       15,=V (NEXT)    ไปยังโปรแกรมย่อยภาษาฟอร์แทรน 'NEXT'
         BALR    14,15           ข้ามไปทำยังโปรแกรมย่อย 'NEXT'
         DROP    15
         USING   *,14
         L       2,XADDR         ตำแหน่งที่อยู่ของ'X' ที่ถูกส่งผ่านมาจากโปรแกรมหลัก
         MVC     0(4,2), XX      ค่าของ X ที่จะถูกส่งกลับไปให้กับโปรแกรมหลัก
         L       13,SAVE+4       ดัชนีชี้ไปยังบริเวณเก่าที่ใช้เก็บค่าต่างๆ
         LM      14,12,12(13)    รีจิสเตอร์ต่างๆที่โปรแกรมหลักใช้
         BR      14              กลับคืนไปโปรแกรมหลัก
XX       DC      E'12.25'        ถูกเปลี่ยนโดย 'Y = 17.5' ใน 'NEXT'
*        ARGUMENT LIST FOR THE CALL TO 'NEXT' I.E. CALL NEXT(A,XX)
```

```
ARGLIST   DC    B'10010100',AL3(ASTAR) ตัวชี้ไปยังโปรแกรมย่อย STAR ของ'A'
          DC    B'10000100',AL3(XX)    ตัวชี้ไปยัง 'XX'
          DC    B'00010000',AL3(0)     สิ้นสุดรายชื่อตัวชี้ต่างๆ
XADDR     DC    A (*-*)                เก็บตำแหน่งที่อยู่ของ 'X' ไว้ที่นี่
SAVE      DS    18F                    บริเวณที่เก็บค่าต่างๆ
ASTAR     EQU   *-4                    นี่เป็นโปรแกรมย่อย STAR ที่จะส่งผ่าน'A'ไป
          DC    2A(*-*)                ให้กับโปรแกรมย่อย 'NEXT'
          END
```

4. โปรแกรมย่อย STAR ทั้งหลายที่ตัวแปลภาษาทำให้เกิดขึ้นมา

รูปแบบเต็มของโปรแกรมย่อย STAR ที่ตัวแปลภาษาสร้างขึ้นมาสำหรับตัวแปรแบบ 1 มิติ A จะเป็นดังต่อไปนี้ :-

```
          CNOP      2,4
          DC        C16'A'
  A*      BAL       R15,XATEST      See note 1
          DC        AL1(f) ,AL3(1st element in array)
          DC        AL1(s) ,AL3(length,in bytes,of array) See note 2
          DC        A(n)            See note 3
          DC        A(d1)-
          DC        A(d2) 1
                          1         only k present
          .               .
          .               1
          .               1
                          1
```

```
DC      A(dk)-
DC      B'C0C1C2C3C4C5C6C7' ,AL3 (a1)-
DC      B'C00000000' ,AL3 (a2)          1
DC      B'C00000000' ,AL3 (a3)          1
.                                       >   See note 4
.                                       1
.                                       1
                                        1
DC      B'C00000000' ,AL3 (aj)          -
.
```

ในที่นี้ k เป็นจำนวนมิติของ A

j เป็นจำนวนมิติที่เปลี่ยนแปรได้ (ได้อย่างมากที่สุด = 7)

d ,d เหล่านี้เป็นต้น เป็นจำนวนมิติที่จะใช้ในการคำนวณตำแหน่งของข้อมูลแต่ละตัวในตัวแปรแบบ 1 มิติ

f = 4k-4

s = s - ค่าจากตารางรูปที่ 3.11

หมายเหตุ :

1. XATEST เป็นชื่อของโปรแกรมย่อยที่มีหน้าที่คำนวณค่าของดัชนีกำกับ ซึ่งเป็นโปรแกรมย่อยภายในตัวแปลภาษา และเก็บอยู่ในตัวแปลภาษา STARTA ซึ่งตำแหน่งที่เก็บอยู่ใน R12 ถ้าเป็นตัวแปรแบบ 1 มิตินี้เป็นประเภท CHARACTER*N โดย n>1 คำสั่งงานจะใช้ R12 เป็นตัวเก็บดัชนีแทนที่จะเป็นฐาน นั่นคือ XATEST คือ d(R12) แทนที่จะเป็น d(,R12) ซึ่งผู้ใช้ข้อมูลนี้คือโปรแกรมย่อยที่มีหน้าที่จัดการเกี่ยวกับข้อผิดพลาด ชื่อ ERROR

2. ถ้าเป็นตัวแปรแบบ 1 มิติ เป็นตัวแปรที่ใช้ในโปรแกรมประกอบ เมื่อมีการเรียกโปรแกรมประกอบ โปรแกรมตัวนำทางจะเดียวกัน โปรแกรมย่อยตัวนำนี้ก็จะคำนวณ และนำความยาวไปใส่ใน adcon ในกรณีที่ขนาดของตัวแปรแบบ 1 มิติ ในโปรแกรมประกอบนั้นเปลี่ยนแปรได้

3. จะมีค่านี้ปรากฏอยู่ในกรณีที่ตัวแปรแบบ 1 มิติเป็นประเภท CHARACTER*n โดยที่ n>1 เท่านั้น ซึ่งในกรณี f=4k ความจริงแล้วตัวแปลภาษาจะถือเหมือนกับว่า ตัวแปรแบบ 1 มิติ โดยที่จำนวนมิติจริงๆแล้วเป็นแค่ n เท่านั้น

4. ค่าแรกนี้จะมีปรากฏอยู่ถ้าตัวแปรแบบ 1 มิติ เป็นตัวแปรแบบ 1 มิติ ที่จะใช้ในโปรแกรมประกอบหนึ่งๆ บิท C1 ถึง C7 จะระบุถึงขนาดมิติตัวที่เปลี่ยนแปรได้ ถ้ามีอยู่ด้วย โดยที่

C_1 จะเป็น 1 ถ้าขนาดมิติตัวสุดท้ายเปลี่ยนแปรได้ไม่คงที่

C_2 จะเป็น 1 ถ้าขนาดมิติตัวรองสุดท้ายเปลี่ยนแปรได้

เป็นดังนี้ไปเรื่อยๆ ถ้าตั้งแต่ C1 ถึง C7 ไม่มีบิทตัวใดมีค่าเป็น 1 เลย ดังนั้นทั้ง C0 และ a1 จะต้องเป็น 0 มิฉะนั้นแล้วก็ C0 และ a1 จะระบุถึงตำแหน่งที่เก็บของขนาดมิติตัวสุดท้ายที่เปลี่ยนแปรได้ ดังต่อไปนี้

ถ้า C0 = 0 แล้ว AL3(a1) = AL3 (ขนาดมิติที่เปลี่ยนแปรได้)

ถ้า C0 = 1 แล้ว AL3(a1) = AL3 (V) โดยที่ V ถูกกำหนดมาโดย

DC A (ขนาดมิติที่เปลี่ยนแปรได้)

บิท C0 จะเป็น 1 ถ้าขนาดมิติที่เปลี่ยนแปรได้อยู่ใน COMMON หรือ เป็นพารามิเตอร์ประเภทที่คูตามตำแหน่งที่เก็บของโปรแกรมประกอบ ค่าที่ 2,3 ไปเรื่อยๆจะมีปรากฏอยู่ถ้ามี 2,3 ไปเรื่อยๆ ขนาดมิติที่เปลี่ยนแปรได้ โดยค่าที่ 2 จะใช้หาขนาดมิติตัวรองสุดท้ายที่เปลี่ยนแปรได้ ค่าที่ 3 จะใช้หาขนาดมิติตัวถัดจากรองสุดท้ายที่เปลี่ยนแปรได้ เป็นดังนี้ไปเรื่อยๆ สำหรับค่าเหล่านี้คือ C0 และ a2, a3,....., ก็จะถูกแปลความหมายทำนองเดียวกับที่กล่าวมาแล้วข้างต้น

กรณีที่การบอกถึงตัวแปรแบบ 1 มิติที่ใช้ในโปรแกรมประกอบมีขนาดมิติสุดท้ายเป็น 1 จะถูกจัดการดังต่อไปนี้ บิทที่เหมาะสม (C1 ถึง C7) จะถูกกำหนดให้มีค่าเป็น 1 เพื่อระบุถึงขนาดมิติที่เปลี่ยนแปรได้ และตำแหน่งที่อยู่ที่เกี่ยวข้องกัน (A1 ถึง A7) จะถูกกำหนดให้มีค่าเป็น 0 ใน LANDR ณ เวลาทำงานจริง จะมีการตรวจสภาพนี้ และจำนวนมิติที่เปลี่ยนแปรได้ราวกับของจริงคือ 1 จะถูกคำนวณใหม่ เพื่อว่าขนาดของตัวแปรแบบ 1 มิติในโปรแกรมประกอบจะได้เท่ากับขนาดของตัวแปรแบบ 1 มิติ ดังที่กล่าวมาแล้วข้างต้น

1. โปรแกรมประกอบตัวนำจะใช้กลุ่มคำที่ได้อธิบายมาให้หมายเหตุ 3 ตอนทำงานจริงเพื่อกำหนดค่าที่ถูกต้องสัมพันธ์กับขนาดมิติในรายชื่อที่มีมาได้ และเพื่อคำนวณผลรวบรวมความยาวทั้งหมดของตัวแปรแบบ 1 มิติ ตัวนำนี้จะกำหนดค่าให้กับค่าตัวแปรแบบ 1 มิติที่ใช้ในโปรแกรมย่อย STAR ต่อเมื่อได้ผ่านค่าตัวแปรต่างๆไปหมดแล้ว

ตัวอย่าง:

```
      SUBROUTINE RTN (ALPHA,X,/M/,N)
      COMMON K
      DIMENSION ALPHA (10,K,N,5,M,12)
```

the compiler constructs the following the STAR routine for ALPHA.

```
         CNOP     2,4
         DC       C16 'ALPHA'
ALPHA*   BAL      R15,XAN
         DC       AL1(20),AL3(*-*) 1st element; filled by prologue
         DC       AL1(2),AL3(*-*)  length; filled by prologue
         DC       A (10)
```

```
DC      A (*-*)          filled by prologue from K
DC      A (*-*)          filled by prologue from N
DC      A (5)
DC      A (* *)          filled by prologue from M
DC      A (12)
DC      B'10101100',AL3 (M)
DC      B'00000000',AL3 (N)
DC      B'10000000',AL3 (K)
```

The compiler also constructs M and K as

```
K       DC      A (K)
M       DC      A (*-*)  filled in by prologue.
```

โปรแกรมย่อย STAR จะถูกเก็บไว้ในบริเวณเดียวกับบริเวณที่เก็บข้อมูลของส่วนโปรแกรมนั้นที่จะต้องใช้โปรแกรมย่อยที่เก็บตัวแปรแบบ 1 มิติ ทั้งนี้ตัวแปรที่ใช้ในโปรแกรมประกอบจะถูกจัดให้ใช้ในบริเวณที่เก็บตัวแปรแบบ 1 มิติ