บทที่ 5
การพัฒนาโปรแกรม
(Program development)

การพัฒนาโปรแกรมแบ่งเป็น 4 ขั้นตอนดังนี้

สร้าง source program ไฟล์

การคอมไพล์ source program ไฟล์ให้เป็น object module ไฟล์

การลิงค์ (linking) object module ไฟล์เพื่อสร้าง an image

การ execute และการแก้ไขที่ผิด (debugging) ในโปรแกรม

5.1 การสร้างโปรแกรม (Creating the program)

ในการ run โปรแกรมนั้น เริ่มต้นด้วยการสร้างแฟ้มข้อมูลของ source program, การ default ชนิดของแฟ้มข้อมูลขึ้นอยู่กับภาษาที่ใช้เขียนโปรแกรมสำหรับเครื่อง VAX-11 การ default ชนิดของแฟ้มข้อมูล ที่เป็น source program จะเป็นดังนี้

ชนิดของไฟล์ (File type)	ความหมาย (Content)
BAS	Input source file สำหรับ BASIC compiler
B32	Input source file สำหรับ BLISS-32 compiler
C	Input source file สำหรับ C compiler
COB	Input source file สำหรับ COBOL compiler
COB	Input source file สำหรับ COBOL-74 compiler
COR	Input source file สำหรับ CORAL-66 compiler
FOR	Input source file สำหรับ FORTRAN compiler
MAR	Input source file สำหรับ MACRO assembler
PAS	Input source file สำหรับ PASCAL compiler
PLI	Input source file สำหรับ PL/I compiler

5.2 การคอมไพล์โปรแกรม (Compiling or assembling the program)

การเตรียม source program สำหรับนำไป execute โดยคอมพิวเตอร์ โปรแกรมที่เราเขียนนั้น จะถูกคอมไพล์ (compile) ให้อยู่ในรูปแบบที่คอมพิวเตอร์สามารถเข้าใจได้ นั่นคือ จะต้องมีโปรแกรมอีกชุดหนึ่งเรียกว่าคอมไพล์เลอร์ (compiler) หรือ assembler ซึ่งจะแปล source program ให้เป็นภาษาเครื่อง (binary machine code) ซึ่งเป็นภาษาที่คอมพิวเตอร์สามารถแปลความหมายได้

ภาษา assembly ปกติจะออกแบบมาเพื่อใช้กับคอมพิวเตอร์เครื่องใดเครื่องหนึ่งโดยเฉพาะ และถูกแปลบรรทัดต่อบรรทัดให้เป็นภาษาเครื่อง แต่ภาษาระดับสูงส่วนใหญ่ กลับตรงกันข้าม เพราะถูกออกแบบมาเพื่อใช้ทั่วไป และหนึ่งบรรทัดของ source code จะถูกแปลให้เป็น machine code หลายบรรทัด ถ้า source program เขียนด้วยภาษา Assembly (หมายถึง VAX-11 MACRO) เราต้องเรียก VAX-11 MACRO assembler เพื่อเอามาแปลโปรแกรมให้เป็นภาษาเครื่อง ถ้า source program เขียนด้วยภาษาระดับสูง เช่น BASIC, C, COBOL, FORTRAN, PASCAL, หรือ PL/I เราก็ต้องเรียกคอมไพล์เลอร์ของภาษานั้นเพื่อมาคอมไพล์ (compile) โปรแกรม

ต่อไปนี้เป็นคำสั่ง DCL เพื่อเรียกคอมไพล์เลอร์ของแต่ละภาษา

คำสั่ง (Command)	เรียก (Invokes)
BASIC	VAX-11 BASIC compiler
BLISS	VAX-11 BLISS-32 compiler
CC	VAX-11 C compiler
COBOL	VAX-11 COBOL compiler
COBOL/C74	VAX-11 COBOL-74 compiler
CORAL	VAX-11 CORAL-66 compiler
FORTRAN	VAX-11 FORTRAN compiler
MACRO	VAX-11 MACRO assembler
PASCAL	VAX-11 PASCAL compiler
PLI	VAX-11 PL/I compiler

แต่ละคำสั่งจะเรียกคอมไพล์เลอร์ภาษานั้นเพื่อแปล source program ที่เราตั้งชื่อแฟ้มข้อมูลไว้ตามหลังคำสั่งนี้ ถึงแม้ว่าแต่ละคำสั่งจะแตกต่างกันบ้างเล็กน้อยเกี่ยวกับ parameters และ qualifiers แต่รูปแบบของคำสั่งก็ยังเหมือนกัน

ตัวอย่าง

$ *cobol myfile*

คำสั่งนี้เรียกคอมไพล์เลอร์ภาษา COBOL เพื่อคอมไพล์แฟ้มข้อมูล MYFILE ให้เป็นภาษาเครื่องได้เป็น output file เรียกว่า object module เนื่องจากไม่ได้บอกชนิดของแฟ้มข้อมูลไว้, คอมไพล์เลอร์จึง default ชนิดของแฟ้มข้อมูลเป็น COB

5.3 การลิงก์ object module

object module นั้นโดยตัวมันเองยังนำไป execute ไม่ได้ คือ ส่วนนี้ยังมีการอ้างถึงโปรแกรมอื่น หรือ routine อื่น ซึ่งจะต้องนำมารวม (combined) กับ object module ก่อนที่จะถูก execute ซึ่งเป็นหน้าที่ของ ลิงค์เกอร์ (linker) ที่จะทำการรวมเข้าด้วยกัน

คำสั่ง LINK เรียก VAX-11 linker, ลิงค์เกอร์จะทำการค้นหา system libraries เพื่อ resolve การอ้างถึง routine หรือสัญลักษณ์ ซึ่งไม่ได้ defined ไว้ใน object module ที่มันกำลังลิงค์อยู่ เราสามารถสั่งให้ลิงค์เกอร์อ้างถึง (include) object module ที่เป็น input ได้มากกว่าหนึ่งชุด หรือเราอาจจะระบุ libraries ของ object module ของเราเอง ให้มันค้นหาก็ได้รูปแบบของคำสั่ง LINK มีดังนี้

$ *link myfile*

เนื่องจากไม่ได้ระบุชนิดของแฟ้มข้อมูล, ลิงค์เกอร์จึง default ชนิดของแฟ้มข้อมูลเป็น OBJ สำหรับ object module

ลิงค์เกอร์จะสร้าง an image ซึ่งเป็นแฟ้มข้อมูล หมายถึง โปรแกรมที่เราเขียนและอยู่ในรูปแบบที่นำไป execute ได้

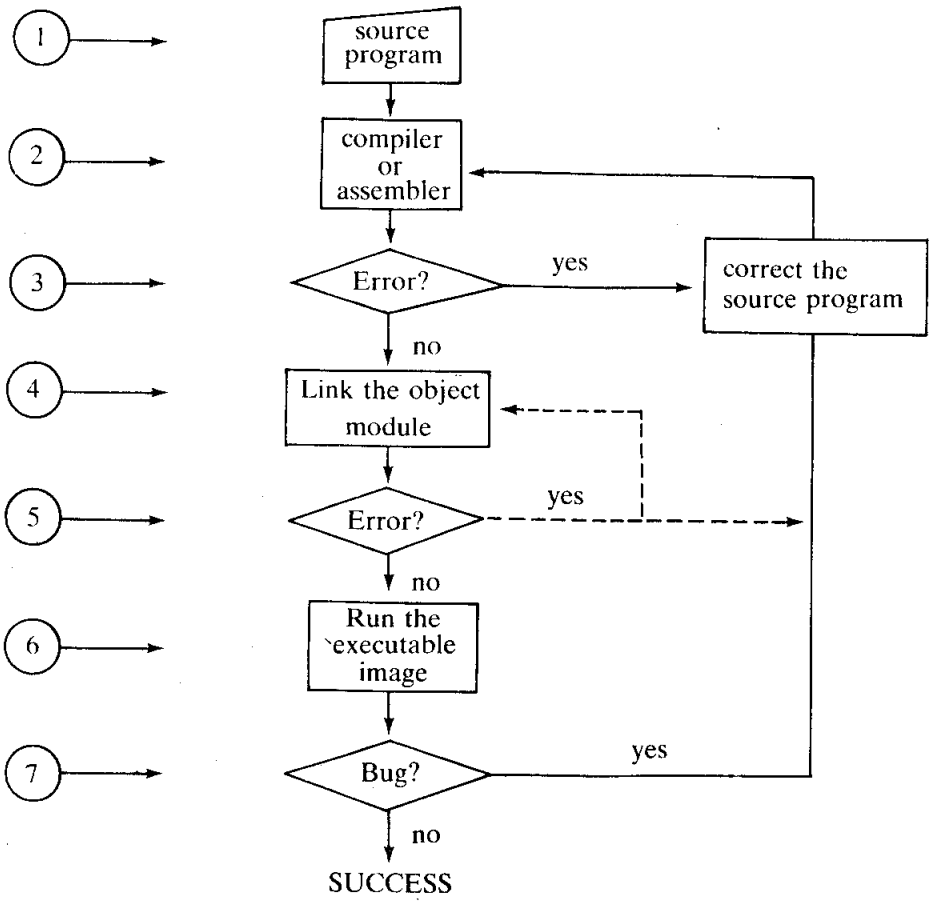
5.4 การ execute โปรแกรม

คำสั่ง RUN ใช้ execute an image นั่นคือมันจะนำ image ซึ่งสร้างด้วยลิงค์เกอร์ไปไว้ในหน่วยความจำเพื่อที่จะ run, รูปแบบของคำสั่ง RUN มีดังนี้

$ *run myfile*

เนื่องจากไม่ได้ระบุชนิดของแฟ้มข้อมูล คำสั่ง RUN จึง default ชนิดของแฟ้มข้อมูลเป็น EXE เพื่อ execute image

ครั้งแรกที่เรา run โปรแกรม บางทียัง execute ไม่ได้ ถ้ามีที่ผิด (bug) หรือผิดพลาดเนื่องจากการเขียนโปรแกรม เราต้องหาสาเหตุของที่ผิดโดยการตรวจสอบ output จากโปรแกรม เมื่อพบแล้วก็แก้ไขที่ผิด แล้วทำขั้นตอนซ้ำอีกครั้งหนึ่ง คือ compile, link, และ run เพื่อทดสอบผลลัพธ์ ดังรูปที่แสดงข้างล่างนี้



ขั้นที่ 1 ใช้อีดิเตอร์สร้าง source program ไฟล์แล้วเก็บไว้ในดิสค์ เมื่อเรียกคอมไพล์เลอร์หรือ assembler ให้ระบุชื่อแฟ้มข้อมูล

ขั้นที่ 2 ใช้คำสั่งเรียกคอมไพล์เลอร์ภาษานั้น ซึ่งจะตรวจสอบ syntax แล้วสร้าง object module และถ้าต้องการ listing ของโปรแกรมก็ให้ระบุไว้ด้วย

ขั้นที่ 3 ถ้าคอมไพล์เลอร์ให้สัญญาณว่ามีที่ผิด ให้ใช้อีดิเตอร์แก้ไข source program

ขั้นที่ 4 ลิงค์เกอร์จะค้นหา syntem libraries เพื่อแก้ไข (resolve) การอ้างถึง subroutine ใน object module แล้วสร้าง executable image ถ้าเราต้องการระบุ libraries ส่วนตัวเพื่อต้องการค้นหา ก็สามารถทำได้ และถ้าต้องการให้ลิงค์เกอร์ สร้าง storage map ของโปรแกรมก็สามารถทำได้เช่นกัน

ขั้นที่ 5 ถ้า object module อ้างถึง subroutines หรือ symbol ซึ่งไม่มี หรือ หาไม่ได้ (undefined) ลิงค์เกอร์ก็จะให้ข้อความบอกที่ผิด ถ้าลิงค์เกอร์ไม่สามารถเรียก subroutine มาใช้ได้ เราต้องใช้คำสั่ง LINK อีกครั้งหนึ่ง ระบุชื่อ module หรือ libraries ที่ต้องการและถ้า symbol นั้นหาไม่ได้ เราจำเป็นต้องแก้ไข source program

ขั้นที่ 6 คำสั่ง RUN จะ excute program image ขณะที่กำลัง run โปรแกรมนั้น system อาจจะพบที่ผิดและให้ข้อความออกมา ถ้าต้องการตรวจสอบที่ผิดให้ดูที่ output

ขั้นที่ 7 ถ้ามีสิ่งที่ไม่ต้องการ (bug) ในโปรแกรมให้หาสาเหตุที่ผิดแล้วแก้ไข source program

5.5 โปรแกรมภาษาโคบอล (A cobol program)

ขั้นตอนที่แสดงด้วยรูปภาพข้างล่างนี้ เป็นการ run โปรแกรมภาษาโคบอลด้วยเครื่อง VAX/VMS

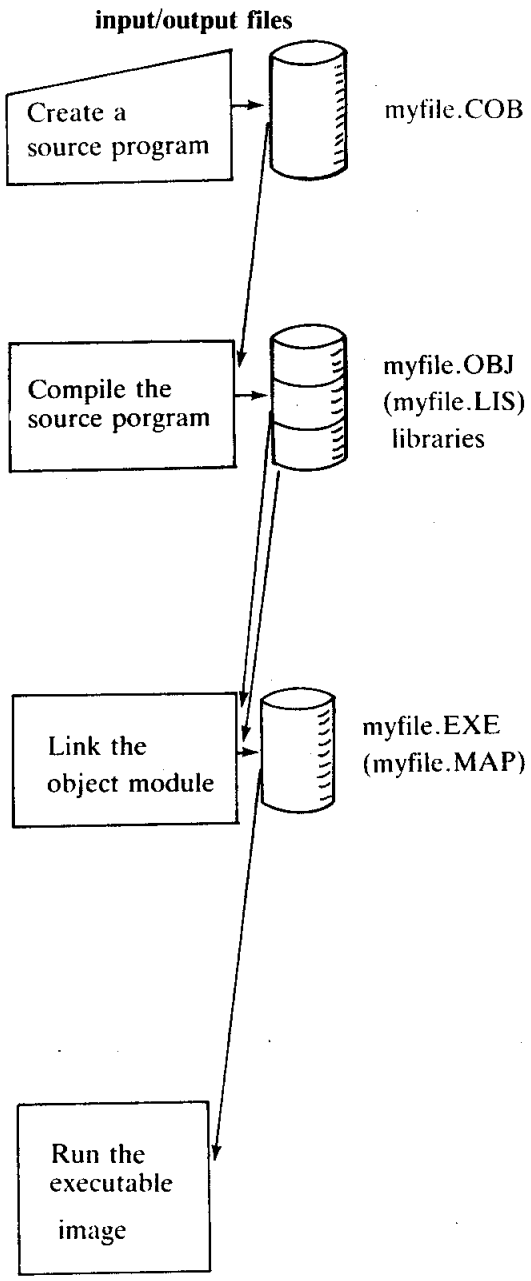
คำสั่ง

$ EDIT/EDT myfile. COB
ในที่นี้ชนิดของแฟ้มข้อมูลคือ COB
แสดงว่าไฟล์นี้เป็นโปรแกรม
ภาษาโคบอล

$ COBOL/LIS myfile
$ COBOL myfile [.COB]
คำสั่ง COBOL เป็นการ
บอกว่า input file มี
ชนิดของแฟ้มข้อมูล เป็น COB
(ถ้าเราใช้ qualifier
/LIST, คอมไพล์เลอร์
จะสร้าง listing file ให้อีกหนึ่งชุด)

$ LINK myfile
คำสั่ง LINK เป็นการบอกว่า
input file มีชนิดของแฟ้มข้อมูลเป็น OBJ
(ถ้าเราใช้ qualifier/MAP
ลิงค์เกอร์จะสร้าง map file ให้อีกหนึ่งชุด)

$ RUN myfile
คำสั่ง RUN เป็นการบอกว่า image
มีชนิดของแฟ้มข้อมูลเป็น EXE



ขั้นตอนในการพัฒนาโปรแกรม (Program development)

1. สร้างแฟ้มข้อมูลของ source โปรแกรมภาษาโคบอลโดยใช้คำสั่ง $ EDIT SAMPLE1.COB

2. คอมไพล์ source โปรแกรมให้เป็นแฟ้มข้อมูลของ object module ใช้คำสั่ง

```
$ COBOL SAMPLE1
```

หรือ

```
$ COBOL/LIST SAMPLE1
```

หรือ `$ COBOL SAMPLE/LIS`

จะได้ output สองไฟล์คือ

SAMPLE1.OBJ

และ SAMPLE1.LIS จะให้ line number ออกมาด้วย

ในขั้นตอนนี้หลังจากคอมไพล์แล้วถ้ามีผิด นักศึกษาตรวจสอบได้โดยสั่งให้เครื่องแสดงแฟ้มข้อมูล SAMPLE.LIS ออกมาทางจอ แล้วเรียกโปรแกรมไฟล์ SAMPLE1.COB มาแก้ไข (edit) ทุกแห่งที่ผิดก่อน เมื่อเก็บ (save) แฟ้มข้อมูลแล้วให้ย้อนกลับไปทำข้อ 2 ซ้ำอีกครั้งหนึ่ง เมื่อคอมไพล์แล้วไม่มีที่ผิดจึงจะทำขั้นที่ 3 ต่อไป

3. การลิงค์ object module ไฟล์ให้เป็น an image ใช้คำสั่ง

```
$ LINK SAMPLE1
```

หรือ `$ LINK SAMPLE1, SUBTHAI [.OBJ], SHIFT [.OBJ]`

จะได้ไฟล์ SAMPLE1.EXE

```
:
CALL "SUBTHAI"
CALL "SHIFT"
```

4. การ execute ไฟล์ SAMPLE.EXE ใช้คำสั่ง

```
$ RUN SAMPLE1
```

ถ้า output ออกมาไม่ถูกต้องให้ตรวจสอบ logic ของโปรแกรมแล้วแก้ไขที่ผิดในโปรแกรม จากนั้นจึงกลับไปทำข้อ 2

สรุป

a) คำสั่ง DCL ที่ใช้พัฒนาโปรแกรมภาษาโคบอล

```
$ EDIT SAMPLE1.COB
$ COBOL/LIS SAMPLE1 [.COB]
$ LINK SAMPLE1 [.OBJ]
```

$ RUN SAMPLE1 [.EXE]

b) ถ้าต้องการให้เครื่องแสดง (display) แฟ้มข้อมูลออกมาทางจอ ใช้คำสั่ง

$ TYPE SAMPLE1.COB*

c) ถ้าต้องการให้เครื่องพิมพ์ตัวโปรแกรมออกมาทาง printer ใช้คำสั่ง

$ PRINT/QUE = LPA0:SAMPLE1.LIS,SAMPLE1.DAT/NOFLAG.
SAMPLE1.LST/NOFLAG RET

---

* *ถ้าใช้คำสั่ง $ TYPE SAMPLE1 เครื่องจะ default ชนิดของไฟล์ (file type) เป็น DAT เสมอ*

แบบฝึกหัด

1. จงเขียน flowchart และอธิบายการพัฒนาโปรแกรมภาษาโคบอลด้วยคำสั่ง DCL
2. ให้ใส่คำสั่ง DCL หรือผลลัพธ์ลงในตารางข้างล่างนี้ ซึ่งเป็น 4 ขั้นตอนของการพัฒนาโปรแกรมภาษาโคบอล ชื่อ PROG

ต้องการ	ใช้คำสั่ง	Input ไฟล์ คือ	Output ไฟล์ คือ
สร้าง source program ไฟล์			
คอมไพล์ source program ได้ an object module ไฟล์			
ลิงค์ object module ไฟล์ ได้ an image			
run the executable image			