

ภาคผนวก F

การทดลองที่ 6 การพัฒนาโปรแกรมภาษาแอสเซมบลี

การทดลองนี้คาดว่าผู้อ่านผ่านการเขียนหรือพัฒนาโปรแกรมด้วยภาษา C ในการทดลองที่ 5 ภาคผนวก E แล้ว และมีความคุ้นเคยกับ IDE จากการพัฒนาโปรแกรมและการดีบั๊กโปรแกรมด้วยภาษา C/C++ ด้วย CodeBlocks ดังนั้น การทดลองนี้มีวัตถุประสงค์เหล่านี้

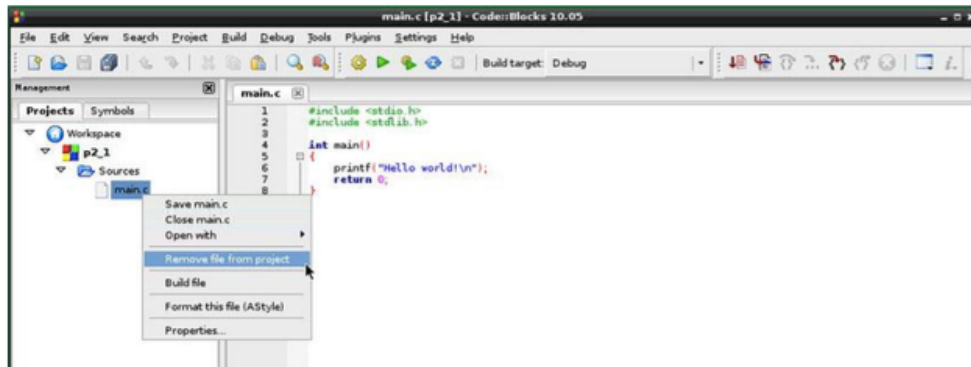
- เพื่อให้เข้าใจการพัฒนาและดีบั๊ก (Debug) โปรแกรมภาษาแอสเซมบลีด้วย IDE ชื่อ CodeBlocks บนระบบปฏิบัติการตระกูลยูนิกซ์
- เพื่อให้เข้าใจความแตกต่างระหว่างการพัฒนาโปรแกรมภาษาแอสเซมบลีด้วย IDE และ Makefile

✕F.1 การพัฒนาโดยใช้ IDE

1. พิมพ์คำสั่งนี้ในโปรแกรม Terminal เพื่อเริ่มต้นใช้งาน CodeBlocks

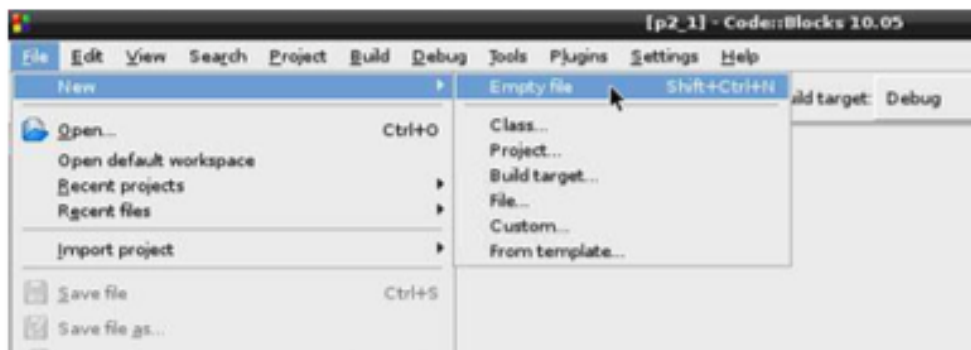
```
$ codeblocks
```

2. หน้าต่างหลักจะปรากฏขึ้น หลังจากนั้น ผู้อ่านสามารถสร้างโปรเจกต์ใหม่โดยเลือก "Create a new project" ในช่องด้านซ้าย แล้วเลือก "Console application" ในช่องด้านขวาเพื่อสร้างโปรแกรม
3. กรอกชื่อโปรเจกต์ใหม่ชื่อ Lab6 ในช่อง Project title: และกรอกชื่อไดเรกทอรี `/home/pi/asm` ในช่อง Folder to create project in: โปรดสังเกตข้อความในช่อง Project filename: ว่าตรงกับ Lab6.cbpp ใช่หรือไม่ แล้วจึงกด Next>
4. โปรแกรม CodeBlocks จะสร้างไดเรกทอรีต่างๆ ภายใต้ไดเรกทอรีชื่อ `/home/pi/asm/Lab6/`
5. กดปุ่ม "Next>" เพื่อดำเนินการต่อและสุดท้ายจะเป็นขั้นตอนการเลือกคอนฟิกูเรชัน (Configuration) สำหรับคอมไพเลอร์ เลือกออปชัน Debug เหมาะสำหรับการเริ่มต้นและแก้ไขข้อผิดพลาด แล้วจึงกดปุ่ม "Finish" เมื่อเสร็จสิ้น
6. คลิกชื่อ Workspace ในหน้าต่างด้านซ้ายเพื่อขยายโครงสร้างโปรเจกต์แล้วค้นหาไฟล์ชื่อ "main.c" คลิกขวาบนชื่อไฟล์ แล้วเลือกเมนู "Remove file from project" ตามรูปที่ [F.1](#)



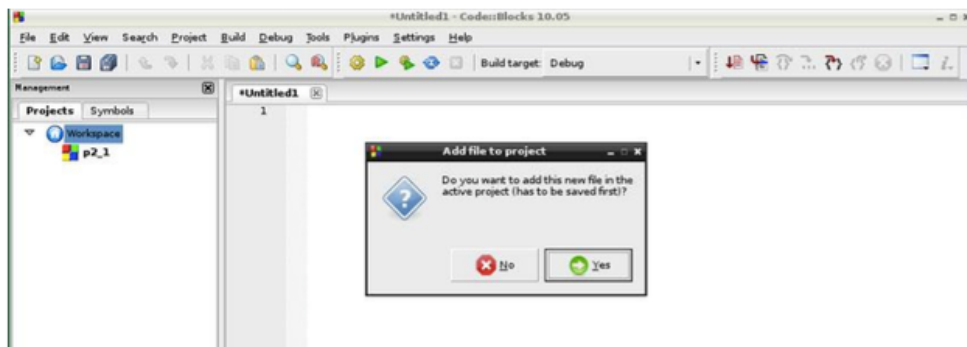
รูปที่ F.1: การย้ายไฟล์ main.c ออกจากโปรเจกต์

7. เพิ่มไฟล์ใหม่ลงในโปรเจกต์โดยกดเมนู File->New->Empty file ตามรูปที่ F.2



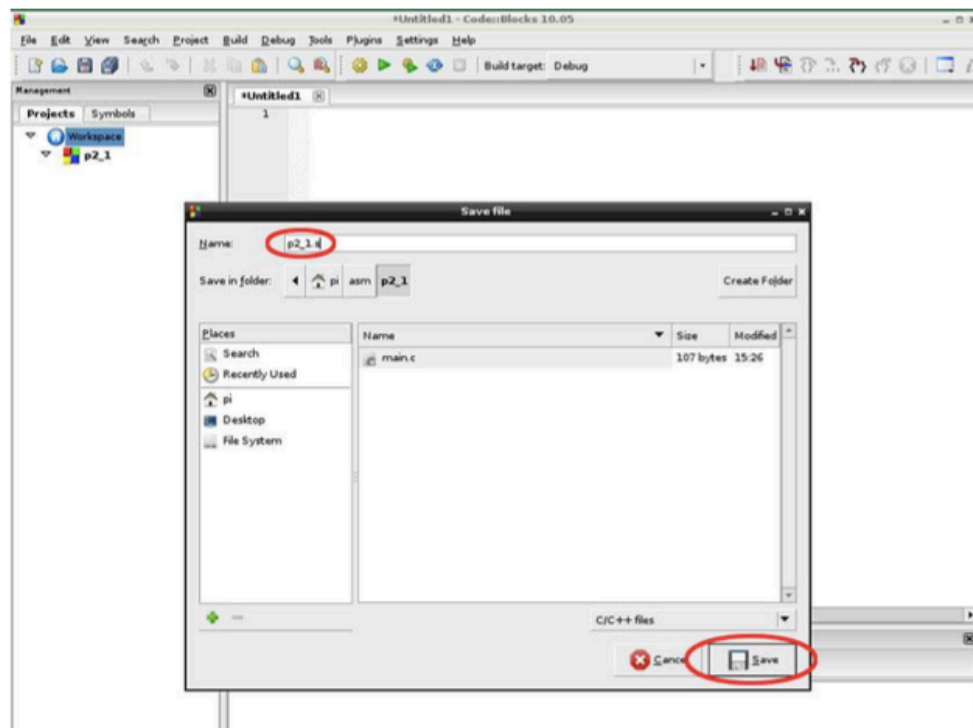
รูปที่ F.2: การเพิ่มไฟล์ใหม่ลงในโปรเจกต์

8. คลิกปุ่ม "Yes" เพื่อยืนยันในรูปที่ F.3



รูปที่ F.3: หน้าต่างกดปุ่ม "Yes" เพื่อยืนยัน

9. หน้าต่าง "Save file" จะปรากฏขึ้น กรอกชื่อไฟล์ว่า main.s แล้วจึงกดปุ่ม "Save" ดังรูปที่ F.4



รูปที่ F.4: หน้าต่าง Save File ชื่อไฟล์ว่า main.s

10. คลิกชื่อ Workspace ในหน้าต่างด้านซ้าย และคลิกขวาเพิ่ม (Add) ไฟล์ main.s เข้าไปในโปรเจกต์
11. ป้อนคำสั่งเหล่านี้ในไฟล์ main.s

```
.global main
main:
    MOV R0, #0
    MOV R1, #2
    MOV R2, #4
    ORR R0, R1, R2
    BX LR
```

12. เลือกเมนู Build->Build เพื่อแปล (Assemble) โปรแกรมที่เขียนให้เป็นโปรแกรมภาษาเครื่อง
13. เลือกเมนู Build->Run เพื่อรันโปรแกรม
14. อ่านและบันทึกประโยชน์ที่เกิดขึ้นในหน้าต่าง Terminal ที่ปรากฏขึ้นมา

X F.2 การดีบั๊กโปรแกรมโดยใช้ IDE

1. ในไฟล์ main.s เลื่อนเคอร์เซอร์ไปบรรทัดที่มีคำสั่ง ORR R0, R1, R2 คลิกเมนู Debug->breakpoint หรือกดปุ่ม F5 ผู้อ่านจะสังเกตวงกลมสีแดงปรากฏขึ้นด้านซ้ายของคำสั่ง ORR
2. กดเมนู Debug->Debugging Windows->CPU Registers เพื่อแสดงค่าของ CPU register ในหน้าต่างที่ปรากฏขึ้นมาเพิ่มเติม
3. เมื่อพร้อมแล้ว ผู้อ่านสามารถเริ่มต้นการดีบั๊กโดยกดเมนู Debug->Start/Continue หรือกดปุ่ม F8 โปรแกรมจะเริ่มทำงานตั้งแต่ประโยคแรกจนหยุดที่คำสั่ง ORR R0, R1, R2
4. อ่านและบันทึกค่าของ R0 และ PC ในหน้าต่าง CPU Registers
5. ประมวลผลคำสั่งถัดไปโดยกดเมนู Debug->Next Instruction หรือกดปุ่ม Alt+F7 พร้อมกัน
6. อ่านและบันทึกค่าของ R0 และ PC ในหน้าต่าง CPU Registers และสังเกตการเปลี่ยนแปลงที่เกิดขึ้น โดยเปรียบเทียบกับค่า R0 และ PC ในข้อ 4 กับข้อนี้
7. อธิบายว่าเกิดอะไรขึ้นกับค่าของรีจิสเตอร์ R0 และ PC

F.3 การพัฒนาโดยใช้ประโยคคำสั่งทีละขั้นตอน

ผู้อ่านควรเข้าใจคำสั่งพื้นฐานในการแปลโปรแกรมภาษาแอสเซมบลีที่สร้างขึ้นใน CodeBlocks ก่อนหน้านี้ตามขั้นตอนต่อไปนี้

1. ใช้โปรแกรมไฟล์เมเนเจอร์เพื่อเบร่าสไฟล์ในไดเรกทอรี /home/pi/asm/Lab6
2. ดับเบิลคลิกบนชื่อไฟล์ main.s เพื่อเปิดอ่านไฟล์และเปรียบเทียบกับไฟล์ที่เขียนในโปรแกรม CodeBlocks
3. เปิดโปรแกรม Terminal หน้าต่างใหม่ แล้วย้ายไดเรกทอรีปัจจุบัน (cd: change directory) ไปยัง /home/pi/asm/Lab6 โดยใช้คำสั่ง

```
$ cd /home/pi/asm/Lab6
```

↳ ๓๖3๐1๐๐๖

4. แปลไฟล์ซอร์สโค้ดให้เป็นไฟล์อ็อบเจกต์ โดยเรียกใช้คำสั่ง as (assembler) ดังนี้

```
$ as -o main.o main.s
```

5. ใช้คำสั่ง ls -la ใน Terminal เพื่อค้นหาไฟล์อ็อบเจกต์ชื่อ main.o ว่ามีจริงหรือไม่ มี
6. ทำการลิงก์และแปลงไฟล์อ็อบเจกต์เป็นไฟล์โปรแกรมโดย

```
$ gcc -o Lab6 main.o
```

- ใช้คำสั่ง `ls -la` ใน Terminal เพื่อค้นหาไฟล์โปรแกรมชื่อ Lab6 ว่ามีจริงหรือไม่
- เรียกโปรแกรม Lab6 โดยพิมพ์

```
$ ./Lab6
```

↗ `echo $?` ได้ 6 , ไม่สามารถเปลี่ยนแก้ไขได้ เพราะไม่ได้รันใน IDE

- เปรียบเทียบหมายเลขที่ปรากฏขึ้นว่าตรงกับผลการรันใน IDE หรือไม่ อย่างไร

F.4 การพัฒนาโดยใช้ Makefile

การใช้ **makefile** สำหรับพัฒนาโปรแกรมภาษาแอสเซมบลีคล้ายกับการทดลองที่ 5 ในภาคผนวก E ก่อนหน้านี้

- เปิดไดเรกทอรี `/home/pi/asm/Lab6` ด้วยโปรแกรมไฟล์แมนเนเจอร์
- กดปุ่มขวามบนเมาส์ในพื้นที่ไดเรกทอรีเพื่อสร้างไฟล์เปล่าใหม่ (New Empty File) โดยกำหนดชื่อ `makefile`
- ป้อนข้อความเหล่านี้ลงในไฟล์ `makefile`:

```
Lab6: main.o
    gcc -o Lab6 main.o
main.o: main.s
    as -o main.o main.s
clean:
    rm *.o Lab6
```

- บันทึกไฟล์แล้วปิดหน้าต่างบันทึก ผู้อ่านควรตรวจสอบรายชื่อไฟล์ที่อยู่ภายในไดเรกทอรีนี้ว่ามีไฟล์อะไรบ้าง

- ลบไฟล์อ็อบเจกต์ที่มีอยู่โดยใช้คำสั่ง

```
$ make clean
```

```
t63010062@Pi432b:~/asm/Lab6 $ ls -la
total 28
drwxr-xr-x 2 t63010062 t63010062 4096 Feb 19 22:37 .
drwxr-xr-x 3 t63010062 t63010062 4096 Feb 19 22:33 ..
-rwxr-xr-x 1 t63010062 t63010062 7908 Feb 19 22:37 Lab6
-rw-r--r-- 1 t63010062 t63010062 652 Feb 19 22:37 main.o
-rw-r--r-- 1 t63010062 t63010062 79 Feb 19 22:27 main.s
-rw-r--r-- 1 t63010062 t63010062 89 Feb 19 22:25 makefile
```

ในโปรแกรม Terminal เพื่อเปรียบเทียบหลังจากที่รันคำสั่ง `make clean`

- ใช้คำสั่ง `ls -la` ใน Terminal ค้นหาไฟล์อ็อบเจกต์ `main.o` และ `Lab6` ว่าถูกลบหรือไม่

7. ทำการแอสเซมเบิล main.s โดยใช้คำสั่ง make ในโปรแกรม Terminal และขอให้สังเกตวันเวลาของไฟล์ต่างๆ

\$ make



```
rm *.o Lab6
t63010062@Pi432b:~/asm/Lab6 $ make
as -o main.o main.s
gcc -o Lab6 main.o
t63010062@Pi432b:~/asm/Lab6 $
```

8. ใช้คำสั่ง ls -la ใน Terminal เพื่อค้นหาไฟล์ชื่อ main.o และ Lab6 ว่ามีจริงหรือไม่ **จ**
9. เรียกโปรแกรม Lab6 โดยพิมพ์

\$./Lab6

F.5 กิจกรรมท้ายการทดลอง

- จงปรับแก้คำสั่ง ORR เป็นคำสั่ง AND ในโปรแกรม main.s และ**ตรวจสอบ**ผลการเปลี่ยนแปลงแล้วจึงอธิบาย
- จงปรับแก้โปรแกรมใน main.s เป็นดังนี้ จดบันทึกผลการทดสอบและอธิบาย

```
.global main
main:
    MOV R5, #1
loop:
    CMP R4, #0
    BLE end
else:
    MOV R5, #2
end:
    MOV R0, R5
    BX LR
```

- จงปรับแก้โปรแกรมใน main.s เป็นดังนี้ จดบันทึกผลการทดสอบและอธิบาย

```
.data
.balign 4
var1: .word 1
.text
.global main
main:
    MOV R1, #2
```

```

LDR R2, var1addr
STR R1, [R2]
LDR R0, [R2]
BX LR
var1addr: .word var1

```

①

```

GNU nano 3.2 main.s
.global main
main:
MOV R0, #0
MOV R1, #2
MOV R2, #4
ORR R0, R1, R2
BX LR

```

```

t63010062@Pi432b:~/asm/Lab6 $ echo $?
6

```

```

GNU nano 3.2 main.s
.global main
main:
MOV R0, #0
MOV R1, #2
MOV R2, #4
AND R0, R1, R2
BX LR

```

```

t63010062@Pi432b:~/asm/Lab6 $ echo $?
0

```

เมื่อเราเปลี่ยนแปลงเมื่อเปลี่ยน ORR ($R0 = R1 \text{ OR } R2$) ที่ echo \$? เราได้ค่าคือ 6 เป็น AND ($R0 = R1 \text{ AND } R2$) ซึ่ง echo \$? เราได้ค่าเป็น 0

②

```

GNU nano 3.2 main.s
.global main
main:
MOV R5, #1
loop:
CMP R4, #0
BLE end
else:
MOV R5, #2
end:
MOV R0, R5
BX LR

```

```

t63010062@Pi432b:~/asm/Lab6 $ echo $?
1

```

เมื่อ echo \$? ออกมาคือค่าเป็น 1 เนื่องจากโปรแกรมที่เราทำนั้น $R5 = 1$ แล้วไปอยู่ใน Loop: ซึ่ง CMP $R4, \#0$ BLE end (if $R4 \leq 0$ then end) ซึ่ง $R4 \leq 0$ เป็นจริงจึงผ่านไปทำงานที่ end คือ MOV $R0, R5$ ใน $R0$ มีค่า $= R5 = 1$ จึง echo \$? ออกมาจึงได้ 1

└─ $R4 = 0$

③

```
GNU nano 3.2 main.s
|.data
.balign 4
var1: .word 1
      .text
      .global main
main:
      MOV R1, #2
      LDR R2, var1addr
      STR R1, [R2]
      LDR R0, [R2]
      BX LR
var1addr: .word var1
```

```
t63010062@Pi432b:~/asm/Lab6 $ echo $?
2
```

เมื่อ echo \$? ออกมาค่าเป็น 2 ในโปรแกรมมีคำสั่ง MOV R1, #2 คือให้ R1 = 2 แล้ว LDR R2, var1addr คือหาค่า address ที่เก็บค่า R1 ที่ซ่อนไว้ใน R2 ต่อมาคือ STR R1, [R2] จะนำค่าจาก R1 ไปยัง Address R2 แล้วใช้คำสั่ง LDR R0, [R2] คือนำค่าจาก Address R2 ไปยัง R0 จึงทำให้อ echo \$? ออกมาค่าเป็น 2