Appendix F

การทดลองที่ 6 การพัฒนาโปรแกรมภาษาแอสเซมบลี

การทดลองนี้คาดว่าผู้อ่านเคยเรียนการเขียนหรือพัฒนาโปรแกรมด้วยภาษา C ในการทดลองที่ 5 ภาคผนวก E แล้ว และมีความคุ้นเคยกับ IDE จากพัฒนาโปรแกรมและการดีบักโปรแกรมด้วยภาษา C/C++ ดังนั้น การทดลองมีวัตถุประสงค์เหล่านี้

- เพื่อให้เข้าใจการพัฒนาและดีบัก (Debug) โปรแกรมภาษาแอสเซมบลีด้วย IDE ชื่อ Code::Blocks บนระบบปฏิบัติการ Raspbian/Linux/Unix
- เพื่อให้เข้าใจความแตกต่างระหว่างการพัฒนาโปรแกรมภาษาแอสเซมบลีด้วย IDE และ Makefile

F.1 การพัฒนาโดยใช้ IDE

1. พิมพ์คำสั่งนี้ในโปรแกรม Terminal เพื่อเริ่มต้นใช้งาน Code::Blocks

   ```
   $ codeblocks
   ```

2. หน้าต่างหลักจะปรากฏขึ้น หลังจากนั้น ผู้อ่านสามารถสร้างโปรเจ็คท์ใหม่โดยเลือก "Create a new project" ในช่องด้านซ้าย แล้วเลือก "Console application" ในช่องด้านขวาเพื่อสร้างโปรแกรม
3. กรอกชื่อโปรเจ็คท์ใหม่ชื่อ Lab6 ในช่อง Project title: และกรอกชื่อโฟลเดอร์ /home/pi/asm/ ในช่อง Folder to create project in: โปรดสังเกตข้อความในช่อง Project filename: ว่าตรงกับ Lab6.cbp ใช่หรือไม่
4. โปรแกรม code blocks จะสร้างโฟลเดอร์ต่างๆ ภายใต้โฟลเดอร์ชื่อ /home/pi/asm/Lab6/
5. กดปุ่ม "Next>" เพื่อดำเนินการต่อและสุดท้ายจะเป็นขั้นตอนการเลือกคอนฟิกกูเรชัน (Configuration) สำหรับคอมไพเลอร์ เลือกออพชัน Debug เหมาะสำหรับการเริ่มต้นและแก้ไขข้อผิดพลาด แล้วจึงกดปุ่ม "Finish" เมื่อเสร็จสิ้น

6. กดชื่อ Workspace ในหน้าต่างด้านซ้ายเพื่อขยายโครงสร้างโปรเจ็คท์เพื่อค้นหาไฟล์ ”main.c” คลิกขวาบนชื่อไฟล์ แล้วเลือกเมนู ”Remove file from project” ตามรูปที่ F.1

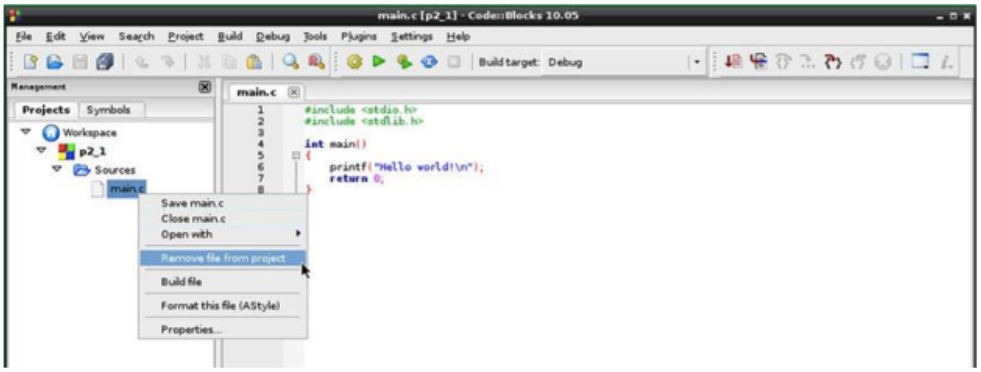


Figure F.1: การย้ายไฟล์ main.c ออกจากโปรเจ็คท์

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

Figure F.2: การเพิ่มไฟล์ใหม่ลงในโปรเจ็คท์

8. กดปุ่ม ”Yes” เพื่อยืนยันในรูปที่ F.3

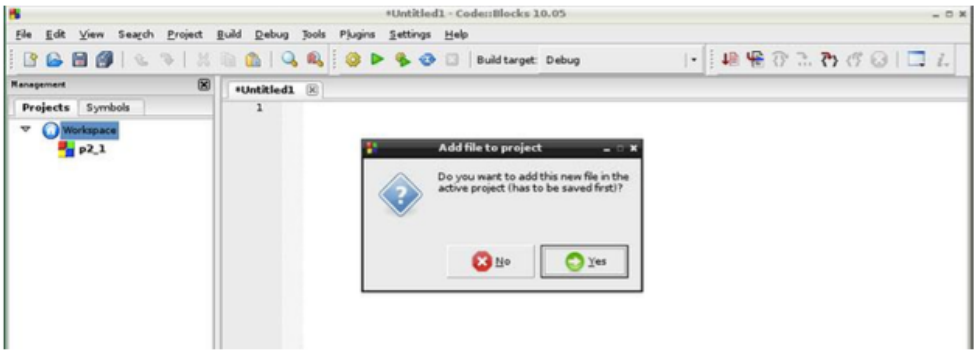


Figure F.3: หน้าต่างกดปุ่ม ”Yes” เพื่อยืนยัน

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

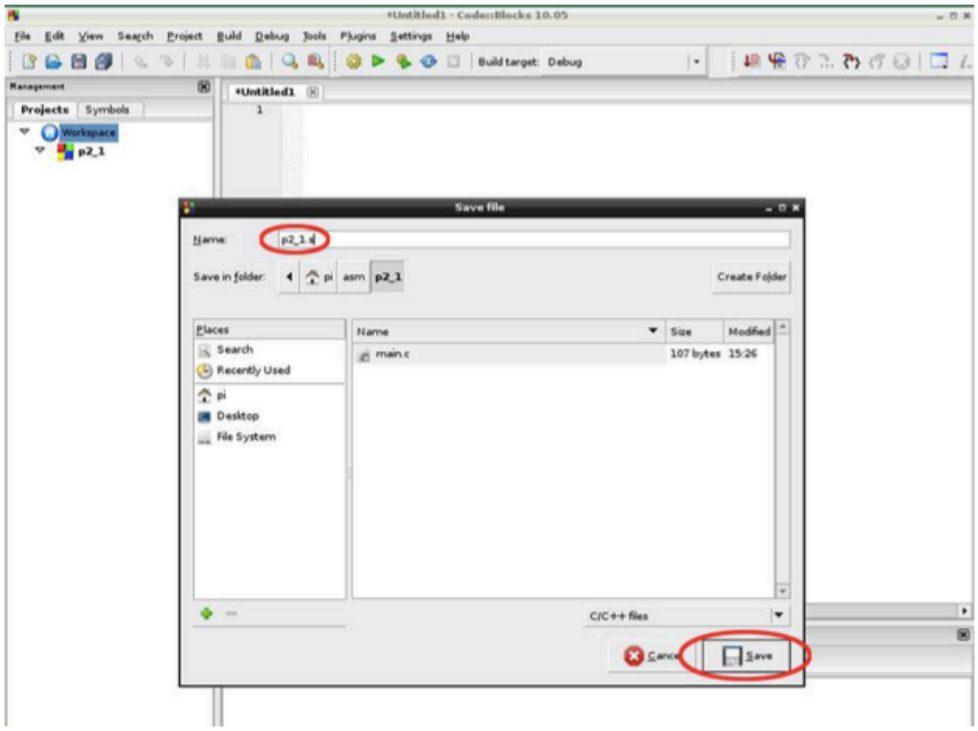


Figure F.4: หน้าต่าง Save File ชื่อไฟล์ว่า main.s

10. เพิ่ม (Add) ไฟล์ main.s เข้าไปในโปรเจ็คท์
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 เพื่อแปลโปรแกรมที่เขียนให้เป็นโปรแกรมภาษาเครื่อง
13. เลือกเมนู Build->Run เพื่อรันโปรแกรม
14. อ่านและบันทึกประโยคที่เกิดขึ้นในหน้าต่าง Terminal ที่โผล่ขึ้นมา

return 6

F.2 การดีบักโปรแกรมโดยใช้ IDE

1. เลื่อนปุ่มเคอร์เซอร์ไปบรรทัดที่มีคำสั่ง ORR R0, R1, R2 คลิกเมนู Debug->breakpoint หรือกดปุ่ม F5 ผู้อ่านจะสังเกตวงกลมสีแดงปรากฏขึ้นด้านซ้าย
2. กดเมนู Debug->Debugging Windows->CPU Registers เพื่อแสดงค่าของ CPU register ในหน้าต่างที่ปรากฏขึ้นมาเพิ่มเติม
3. เมื่อพร้อมแล้ว ผู้อ่านสามารถเริ่มต้นการดีบักโดยกดเมนู Debug->Start/Continue หรือกดปุ่ม F8 โปรแกรมจะเริ่มต้นทำงานตั้งแต่ประโยคแรกจนหยุดที่บรรทัด ที่มีคำสั่ง ORR R0, R1, R2
4. อ่านและบันทึกค่าของ R0 และ PC ในหน้าต่าง CPU Registers
5. ประมวลผลคำสั่งถัดไปโดยกดเมนู Debug->Next Instruction หรือปุ่ม Alt+F7
6. อ่านและบันทึกค่าของ R0 และ PC ในหน้าต่าง CPU Registers และสังเกตการเปลี่ยนแปลงที่เกิดขึ้น
7. อธิบายว่าเกิดอะไรขึ้น

F.3 การพัฒนาโดยใช้ประโยคคำสั่งทีละขั้นตอน

ผู้อ่านควรเข้าใจคำสั่งพื้นฐานในการแปลโปรแกรมภาษาแอสเซมบลีที่สร้างขึ้นใน Code::Blocks ก่อนหน้านี้ ตามขั้นตอนต่อไปนี้

1. ใช้โปรแกรมไฟล์เมเนเจอร์เพื่อเบราส์ไฟล์ในโฟลเดอร์ **/home/pi/asm/Lab6**
2. ดับเบิ้ลกดบนชื่อไฟล์ **main.s** เพื่อเปิดอ่านไฟล์และเปรียบเทียบกับไฟล์ที่เขียนในโปรแกรม Code::Blocks
3. เปิดโปรแกรม Terminal หน้าต่างใหม่ แล้วย้ายโฟลเดอร์ไปยัง **/home/pi/asm/Lab6** โดยใช้คำสั่ง **cd**
4. แปลไฟล์ซอร์สโค้ดให้เป็นไฟล์อ็อบเจ็คท์ โดยเรียกใช้คำสั่ง as (assembler) ดังนี้

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

5. ทำการลิงค์และแปลงไฟล์อ็อบเจ็คท์เป็นไฟล์โปรแกรมโดย

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

6. เรียกโปรแกรม Lab6 โดยพิมพ์

```
$ ./Lab6
$ echo $?
```

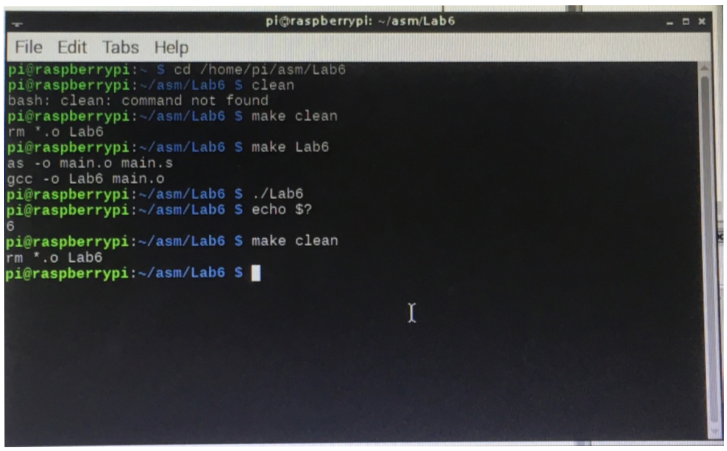
7. เปรียบเทียบหมายเลขที่ปรากฏขึ้นว่าตรงกับผลการรันใน IDE หรือไม่ อย่างไร

F.4 การพัฒนาโดยใช้ Makefile

การใช้ **makefile** สำหรับพัฒนาโปรแกรมภาษาแอสเซมบลี มีความเหมือนกับการทดลองที่ 5 ในภาคผนวก E

1. เปิดไปยังโฟลเดอร์ /home/pi/asm/Lab6 ด้วยโปรแกรมไฟล์เมเนเจอร์
2. กดปุ่มขวาบนเมาส์ในพื้นที่โฟลเดอร์เพื่อสร้างไฟล์เปล่าใหม่ (New Empty File) โดยกำหนดชื่อ makefile
3. ป้อนข้อความเหล่านี้ลงในไฟล์ makefile:

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

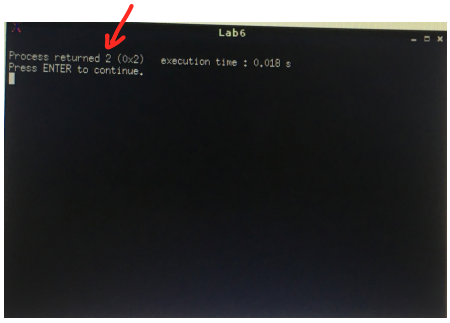


4. บันทึกไฟล์แล้วปิดหน้าต่างบันทึก
5. ลบไฟล์อ็อบเจ็คท์ที่มีอยู่โดยใช้คำสั่ง clean ในโปรแกรม Terminal
6. ทำการแปลโดยใช้คำสั่ง make ในโปรแกรม Terminal
7. เรียกโปรแกรม Lab6 โดยพิมพ์

```
$ ./Lab6
$ echo $?
```

F.5 กิจกรรมท้ายการทดลอง

1. จงปรับแก้คำสั่ง ORR เป็นคำสั่ง AND ในโปรแกรม main.s ทดสอบเพื่อตรวจสอบการเปลี่ยนแปลง อธิบายผลการทดสอบ



2. จงปรับแก้โปรแกรมใน main.s ให้รีเทิร์นค่าอื่นแทน

3. จงปรับแก้โปรแกรมใน main.s เป็นดังนี้:

```
        .global main
main:
        MOV R5, #1 -> เอาเลข 1 ใส่เข้าไปใน R5
loop:
        CMP R4, #0 -> เทียบค่า R0 กับ R4
        BLE end    -> ถ้า R4 ≤ R0 แล้วเป็นจริง ให้ไปที่ end
else:
        MOV R5, #2 -> ถ้า R4 > R0 จะทำบรรทัดนี้
end:
        MOV R0, R5 -> เอา R5 ใส่เข้าไปใน R0
        BX LR
```

จดบันทึกผลการทดสอบและอธิบาย.

4. จงปรับแก้โปรแกรมใน main.s เป็นดังนี้:

```
    .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
```

สร้างตัวแปรใน Memory

เอาค่า 2 ใส่ register R1
Load Address มาจาก เอามาใส่ R2
เอา data R1 เก็บใน Mem Address R2
เอา data จาก Memory Address R2 มาใส่ R0
return R0

จดบันทึกผลการทดสอบและอธิบาย

1. ได้ผลลัพธ์เป็น 0
2. แก้ R2 ให้เป็นเลข 2 จะทำให้ผลลัพธ์เป็น 2