นนทิชา สุขเจริญ 63010484
นภสร ชลานุมาศ 63010492
SEC 102

ภาคผนวก F

การทดลองที่ 6 การพัฒนาโปรแกรมภาษาแอสเซมบลี

การทดลองนี้คาดว่าผู้อ่านผ่านการเขียนหรือพัฒนาโปรแกรมด้วยภาษา C ในการทดลองที่ 5 ภาคผนวก E แล้ว และมีความคุ้นเคยกับ IDE จากการพัฒนาโปรแกรมและการดีบักโปรแกรมด้วยภาษา C/C++ ด้วย CodeBlocks ดังนั้น การทดลองมีวัตถุประสงค์เหล่านี้

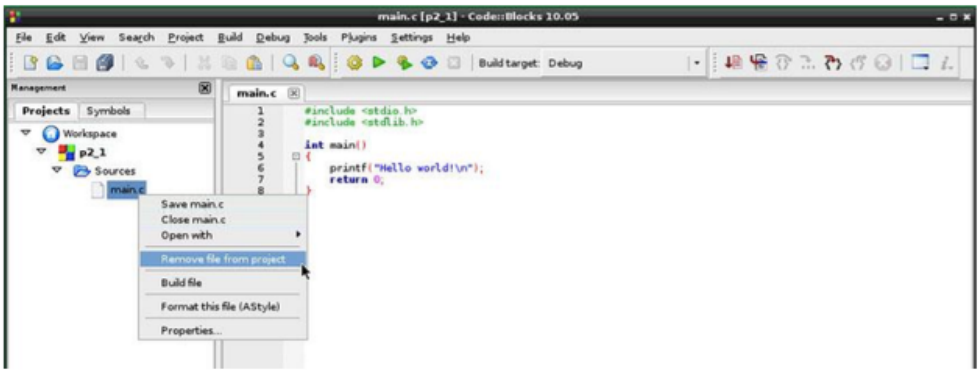
- เพื่อให้เข้าใจการพัฒนาและ**ดีบัก** (Debug) โปรแกรมภาษาแอสเซมบลีด้วย IDE ชื่อ CodeBlocks บนระบบปฏิบัติการตระกูลยูนิกซ์
- เพื่อให้เข้าใจความแตกต่างระหว่างการพัฒนาโปรแกรมภาษาแอสเซมบลีด้วย IDE และ Makefile

F.1 การพัฒนาโดยใช้ IDE

1. พิมพ์คำสั่งนี้ในโปรแกรม Terminal เพื่อเริ่มต้นใช้งาน CodeBlocks

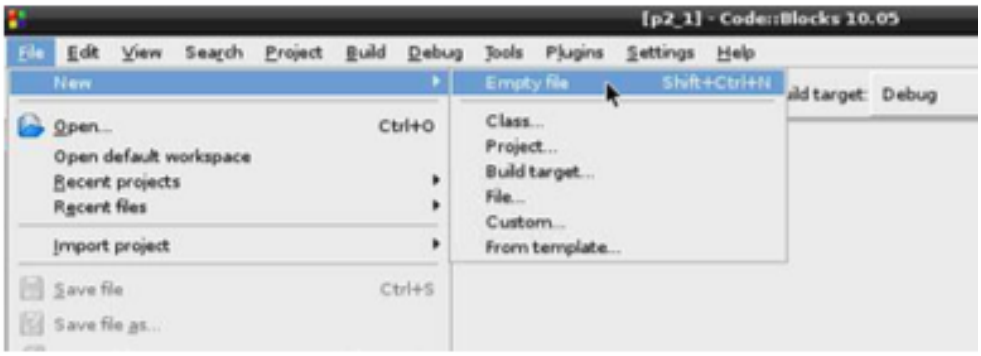
```
$ codeblocks
```

2. หน้าต่างหลักจะปรากฏขึ้น หลังจากนั้น ผู้อ่านสามารถสร้างโปรเจ็คท์ใหม่โดยเลือก ”Create a new project” ในช่องด้านซ้าย แล้วเลือก ”Console application” ในช่องด้านขวาเพื่อสร้างโปรแกรม
3. กรอกชื่อโปรเจ็ทค์ใหม่ชื่อ Lab6 ในช่อง Project title: และกรอกชื่อไดเรกทอรี **/home/pi/asm** ในช่อง Folder to create project in: โปรดสังเกตข้อความในช่อง Project filename: ว่าตรงกับ Lab6.cbp ใช่หรือไม่ แล้วจึงกด Next>
4. โปรแกรม CodeBlocks จะสร้างไดเรกทอรีต่างๆ ภายใต้ไดเรกทอรีชื่อ **/home/pi/asm/Lab6/**
5. กดปุ่ม ”Next>” เพื่อดำเนินการต่อและสุดท้ายจะเป็นขั้นตอนการเลือกคอนฟิกูเรชัน (Configuration) สำหรับคอมไพเลอร์ เลือกออพชัน Debug เหมาะสำหรับการเริ่มต้นและแก้ไขข้อผิดพลาด แล้วจึงกดปุ่ม ”Finish” เมื่อเสร็จสิ้น
6. คลิกชื่อ Workspace ในหน้าต่างด้านซ้ายเพื่อขยายโครงสร้างโปรเจ็คท์แล้วค้นหาไฟล์ชื่อ ”main.c” คลิกขวาบนชื่อไฟล์ แล้วเลือกเมนู ”Remove file from project” ตามรูปที่ F.1



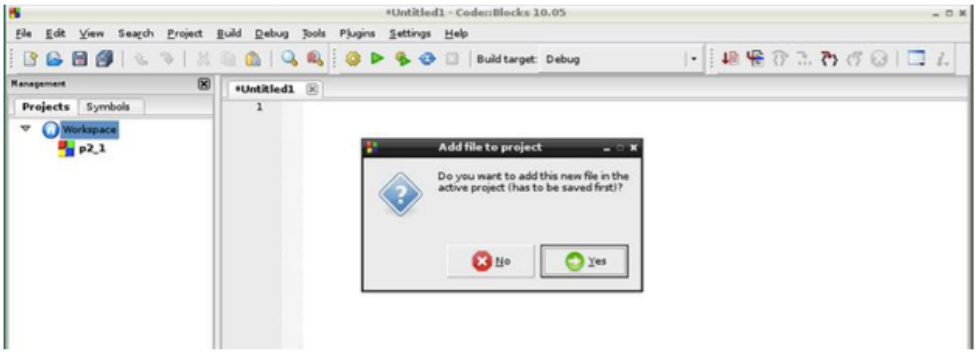
รูปที่ F.1: การย้ายไฟล์ main.c ออกจากโปรเจ็คท์

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



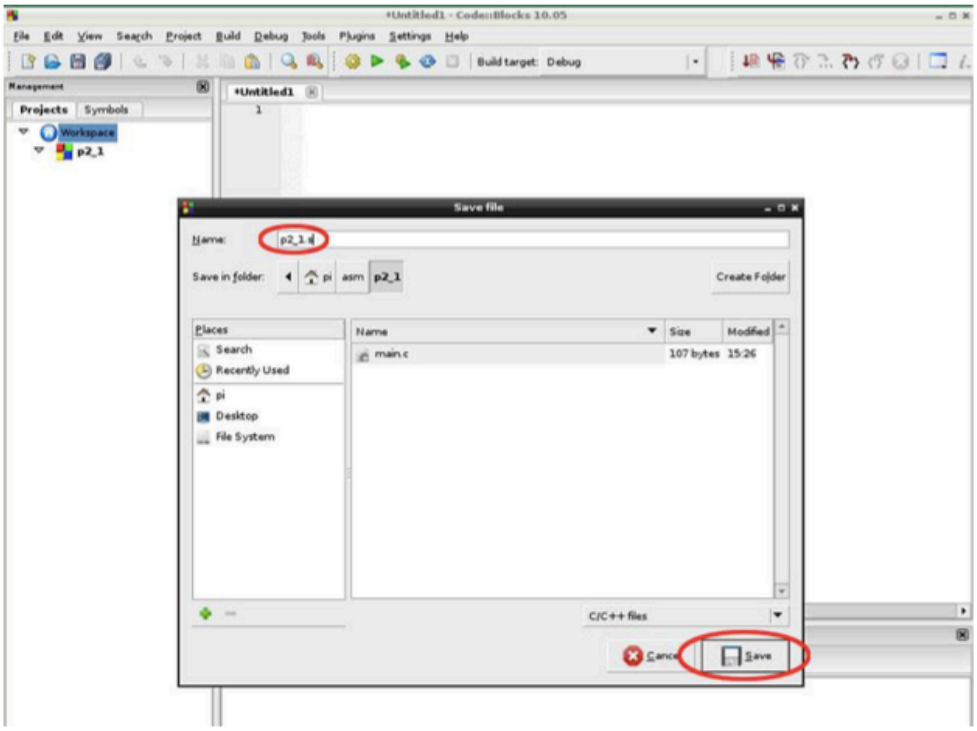
รูปที่ F.2: การเพิ่มไฟล์ใหม่ลงในโปรเจ็คท์

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 เข้าไปในโปรเจ็คท์
11. ป้อนคำสั่งเหล่านี้ในไฟล์ main.s

```
$ nano 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. อ่านและบันทึกประโยคที่เกิดขึ้นในหน้าต่าง Terminal ที่ปรากฏขึ้นมา

F.2 การดีบักโปรแกรมโดยใช้ 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 ผู้อ่านสามารถเริ่มต้นการดีบักโดยกดเมนู Debug->Start/Continue หรือกดปุ่ม F8 โปรแกรมจะเริ่มต้นทำงานตั้งแต่ประโยคแรกจนหยุดที่คำสั่ง 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. ใช้โปรแกรมไฟล์เมเนเจอร์เพื่อเบราส์ไฟล์ในไดเรกทอรี **/home/pi/asm/Lab6**
2. ดับเบิ้ลคลิกบนชื่อไฟล์ **main.s** เพื่อเปิดอ่านไฟล์และเปรียบเทียบกับไฟล์ที่เขียนในโปรแกรม Code-Blocks
3. เปิดโปรแกรม Terminal หน้าต่างใหม่ แล้วย้ายไดเรกทอรีปัจจุบัน (cd: change directory) ไปยัง **/home/pi/asm/Lab6** โดยใช้คำสั่ง

เปลี่ยนเป็น ts3010xxx ๆ

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

4. แปลไฟล์ซอร์สโค้ดให้เป็นไฟล์อ็อบเจกต์ โดยเรียกใช้คำสั่ง as (assembler) ดังนี้

แปลง

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

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

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

7. ใช้คำสั่ง ls -la ใน Terminal เพื่อค้นหาไฟล์โปรแกรมชื่อ Lab6 ว่ามีจริงหรือไม่ มี

8. เรียกโปรแกรม Lab6 โดยพิมพ์

```
$ ./Lab6
```
ไม่มีอะไรเกิดขึ้น

9. เปรียบเทียบหมายเลขที่ปรากฏขึ้นว่าตรงกับผลการรันใน IDE หรือไม่ อย่างไร เปรียบเทียบไม่ได้ เพราะ ไม่ได้ใช้ IDE

F.4 การพัฒนาโดยใช้ Makefile

การใช้ **makefile** สำหรับพัฒนาโปรแกรมภาษาแอสเซมบลีคล้ายกับกับการทดลองที่ 5 ในภาคผนวก E ก่อนหน้านี้

1. เปิดไดเรกทอรี **/home/pi/asm/Lab6** ด้วยโปรแกรมไฟล์เมเนเจอร์ (tc3010484)

2. กดปุ่มขวาบนเมาส์ในพื้นที่ไดเรกทอรีเพื่อสร้างไฟล์เปล่าใหม่ (New Empty File) โดยกำหนดชื่อ make-file

3. ป้อนข้อความเหล่านี้ลงในไฟล์ makefile:

$ nano

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

4. บันทึกไฟล์แล้วปิดหน้าต่างบันทึก ผู้อ่านควร**ตรวจสอบ**รายชื่อไฟล์ที่อยู่ภายในไดเรกทอรีนี้ว่ามีไฟล์อะไรบ้าง

5. ลบไฟล์อ็อบเจกต์ที่มีอยู่โดยใช้คำสั่ง

```
$ make clean
```

ในโปรแกรม Terminal เพื่อเปรียบเทียบหลังจากที่รันคำสั่ง make clean

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

7. ทำการแอสเซมเบิล main.s โดยใช้คำสั่ง make ในโปรแกรม Terminal และขอให้สังเกตวันเวลาของไฟล์ต่างๆ

```
$ make
```

8. ใช้คำสั่ง ls -la ใน Terminal เพื่อค้นหาไฟล์ชื่อ main.o และ Lab6 ว่ามีจริงหรือไม่ มี

9. เรียกโปรแกรม Lab6 โดยพิมพ์

```
$ ./Lab6
```

F.5 กิจกรรมท้ายการทดลอง อธิบายการทดลองก่อนส่ง

1. จงปรับแก้คำสั่ง ORR เป็นคำสั่ง AND ในโปรแกรม main.s และ**ตรวจสอบ**ผลการเปลี่ยนแปลงแล้วจึงอธิบาย

2. 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
```

3. 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)

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

ORR ได้ผลลัพธ์เป็น 6

```
t63010484@Pi432b:~/asm/Lab6 $ echo $?
6
```

AND ได้ผลลัพธ์เป็น 0

```
t63010484@Pi432b:~/asm/Lab6 $ echo $?
0
```

เปลี่ยนแปลง โดย R0 มีค่า = 0 เนื่องจาก $R1 = 0010_2$, $R2 = 0100_2$

เมื่อนำมา AND กันแต่ละ bit ก็คือ 0010_2 AND $0100_2 = 0000_2 = 0_{10}$

0010_2 ORR $0100_2 = 0110_2 = 6_{10}$

---

2)

ดังค่าที่ออกมาว่าเป็นค่าอะไร ⇒ ค่า R0 = 1

```
t63010484@Pi432b:~/asm/Lab6 $ echo $?
1
```

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

ค่า R0 = 1 โดยเริ่มจาก

กำหนดค่า R5 = 1

เปรียบเทียบค่าระหว่าง R4 และ 0

ถ้า R4 ≤ 0 จะไปทำงานที่บรรทัด end label

└ เนื่องจาก R4 ไม่ได้กำหนดค่าเริ่มต้น จึงมีค่าเท่ากับ 0

จึงให้ไปทำงานที่บรรทัด end label เลย

ทำการ copy ค่า R5 ไปยัง R0 ซึ่งมีค่าเท่ากับ 1

---

3)

```
t63010484@Pi432b:~/asm/Lab6 $ echo $?
2
```

ค่า R0 = 2 โดยเริ่มจาก

```
        .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
```

└ เก็บที่อยู่ของ var1

กำหนดค่า R1 = 2

โหลด address ของตัวแปร var1 ลง R2

เก็บค่า R1 ลง var1 ผ่าน address R2

โหลดค่าที่ address R2 ซึ่งเป็นตำแหน่งของ var1 ลงใน R0