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

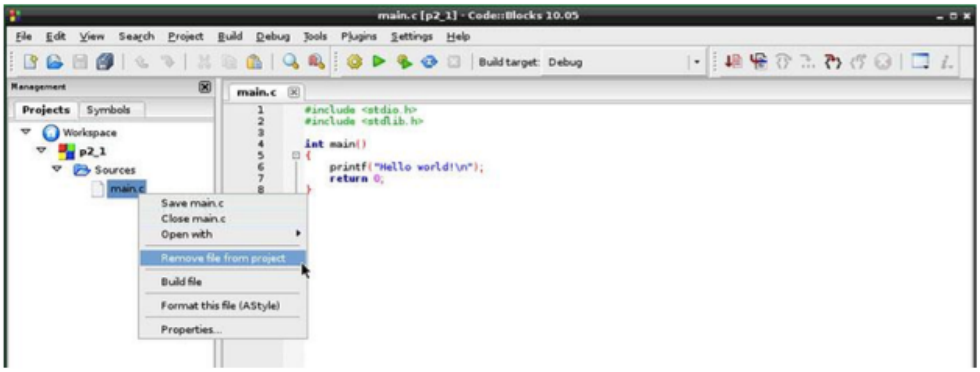
F.1 การพัฒนาโดยใช้ IDE

1. พิมพ์คำสั่งนี้ในโปรแกรม Terminal เพื่อเริ่มต้นใช้งาน CodeBlocks

   ```
   $ codeblocks
   ```

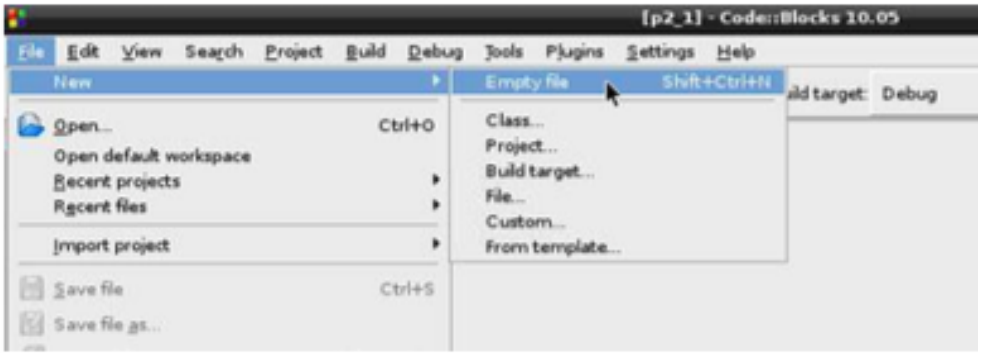
2. หน้าต่างหลักจะปรากฏขึ้น หลังจากนั้น ผู้อ่านสามารถสร้างโปรเจกต์ใหม่โดยเลือก "Create a new project" ในช่องด้านซ้าย แล้วเลือก "Console application" ในช่องด้านขวาเพื่อสร้างโปรแกรม
3. กรอกชื่อโปรเจ็กต์ใหม่ชื่อ Lab6 ในช่อง Project title: และกรอกชื่อไดเรกทอรี **/home/pi/asm** ในช่อง Folder to create project in: โปรดสังเกตข้อความในช่อง Project filename: ว่าตรงกับ Lab6.cbp ใช่หรือไม่ แล้วจึงกด Next>
4. โปรแกรม CodeBlocks จะสร้างไดเรกทอรีต่าง ๆ ภายใต้ไดเรกทอรีชื่อ **/home/pi/asm/Lab6/**
5. กดปุ่ม "Next>" เพื่อดำเนินการต่อและสุดท้ายจะเป็นขั้นตอนการเลือกคอนฟิกกูเรชัน (Configuration) สำหรับคอมไพเลอร์ เลือกออปชัน Debug เหมาะสำหรับการเริ่มต้นและแก้ไขข้อผิดพลาด แล้วจึงกดปุ่ม "Finish" เมื่อเสร็จสิ้น

6. คลิก ชื่อ Workspace ใน หน้าต่าง ด้าน ซ้าย เพื่อ ขยาย โครงสร้าง โปร เจ กต์ แล้ว ค้นหา ไฟล์ ชื่อ ”main.c” คลิกขวาบนชื่อไฟล์ แล้วเลือกเมนู ”Remove file from project” ตามรูปที่ F.1



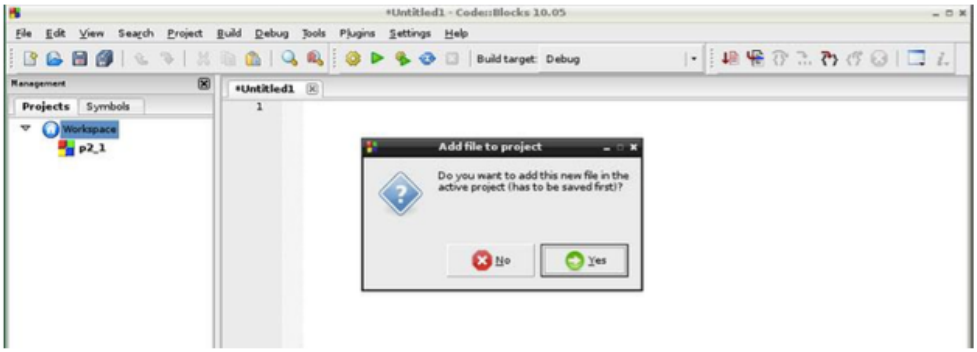
รูปที่ F.1: การย้ายไฟล์ main.c ออกจากโปรเจกต์

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



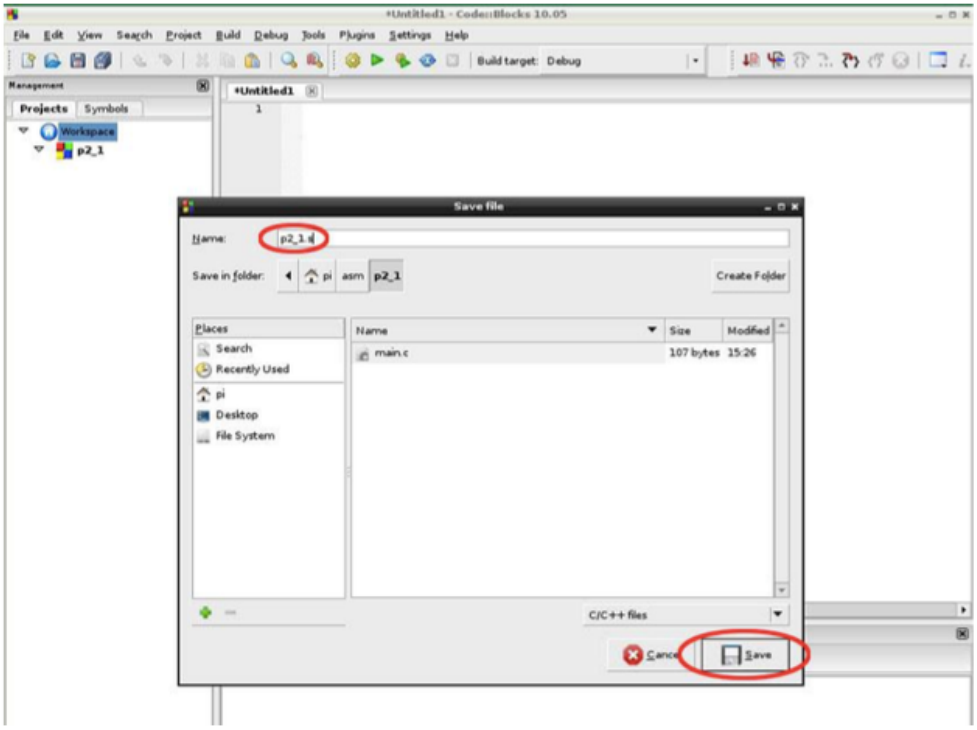
รูปที่ F.2: การเพิ่มไฟล์ใหม่ลงในโปรเจกต์

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

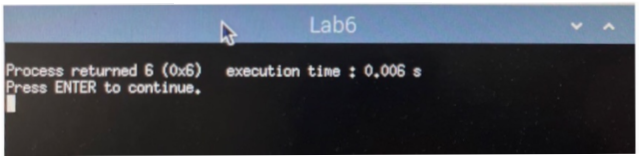


รูปที่ F.4: หน้าต่าง Save File ชื่อไฟล์ว่า main.s

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. อ่านและบันทึกประโยคที่เกิดขึ้นในหน้าต่าง Terminal ที่ปรากฏขึ้นมา



return 6 → $R_1 = 0010$, $R_2 = 0100$

$R_0 = R_1 \mid R_2$

$= 0010 \mid 0100$

$= 0110$

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 $R_0 = 0\times0$ $PC = 0\times103dc <main+12>$
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 $R_0 = 0\times6$ $PC = 0\times103e0 <main+16>$ ∴ มีค่าไม่เท่ากับ ข้อ 4
7. อธิบายว่าเกิดอะไรขึ้นกับค่าของรีจิสเตอร์ R0 และ PC

R_0 เปลี่ยนจาก 0 → 6 เพราะเก็บค่า address ที่ 0 แล้วทำการคำนวณ R_1 OR กับ R_2 แล้วเก็บค่าใน R_0
PC เพิ่มมา 4 bytes เพราะ 1 คำสั่งใช้พื้นที่ 4 bytes

F.3 การพัฒนาโดยใช้ประโยคคำสั่ง (Command Line)

ผู้อ่านควรเข้าใจคำสั่งพื้นฐานในการแปลโปรแกรมภาษาแอสเซมบลีที่สร้างขึ้นใน CodeBlocks ก่อนหน้านี้ ตามขั้นตอนต่อไปนี้

1. ใช้โปรแกรมไฟล์เมเนเจอร์เพื่อเบราส์ไฟล์ในไดเรกทอรี **/home/pi/asm/Lab6**
2. ดับเบิ้ล คลิก บน ชื่อ ไฟล์ **main.s** เพื่อ เปิด อ่าน ไฟล์ และ เปรียบ เทียบ กับ ไฟล์ ที่ เขียน ใน โปรแกรม CodeBlocks
3. เปิดโปรแกรม Terminal หน้าต่างใหม่ แล้วย้ายไดเรกทอรีปัจจุบัน (cd: change directory) ไปยัง **/home/pi/asm/Lab6** โดยใช้คำสั่ง

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

4. แปลไฟล์ซอร์สโค้ดให้เป็นไฟล์อ็อบเจกต์ โดยเรียกใช้คำสั่ง as (assembler) ดังนี้

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

5. ใช้คำสั่ง ls -la ใน Terminal เพื่อค้นหาไฟล์อ็อบเจกต์ชื่อ main.o ว่ามีจริงหรือไม่ มีจริง

6. ทำการลิงก์และแปลงไฟล์อ็อบเจกต์เป็นไฟล์โปรแกรมโดย

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

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

8. เรียกโปรแกรม Lab6 โดยพิมพ์

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

9. เปรียบเทียบหมายเลขที่ปรากฏขึ้นว่าตรงกับผลการรันใน IDE หรือไม่ อย่างไร ตรงกัน 6 = 6

F.4 การพัฒนาโดยใช้ Makefile

การใช้ **makefile** สำหรับพัฒนาโปรแกรมภาษาแอสเซมบลีคล้ายกับกับการทดลองที่ 5 ในภาคผนวก E ก่อนหน้านี้

1. เปิดไดเรกทอรี **/home/pi/asm/Lab6** ด้วยโปรแกรมไฟล์เมเนเจอร์
2. กดปุ่มขวาบนเมาส์ในพื้นที่ไดเรกทอรีเพื่อสร้างไฟล์เปล่าใหม่ (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 ผู้อ่านควรตรวจสอบรายชื่อไฟล์ที่อยู่ภายในไดเรกทอรีนี้ว่ามีไฟล์อะไรบ้าง
5. ลบไฟล์อ็อบเจกต์ที่มีอยู่โดยใช้คำสั่ง

```
$ make clean
```

ในโปรแกรม Terminal เพื่อเปรียบเทียบหลังจากที่รันคำสั่ง make clean

6. ใช้คำสั่ง ls -la ใน Terminal ค้นหาไฟล์อ็อบเจกต์ main.o และ Lab6 ว่าถูกลบหรือไม่ ถูกลบ
7. ทำการแอสเซมเบิล main.s โดยใช้คำสั่ง make ในโปรแกรม Terminal และขอให้สังเกตวันเวลาของไฟล์ต่าง ๆ

```
$ make
```

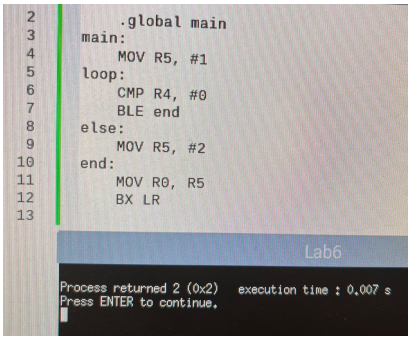
8. ใช้คำสั่ง ls -la ใน Terminal เพื่อค้นหาไฟล์ชื่อ main.o และ Lab6 ว่ามีจริงหรือไม่ มีจริง
9. เรียกโปรแกรม Lab6 โดยพิมพ์

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

F.5 กิจกรรมท้ายการทดลอง

1. จงปรับแก้คำสั่ง ORR เป็นคำสั่ง AND ในโปรแกรม main.s และตรวจสอบผลการเปลี่ยนแปลงแล้วจึงอธิบาย return 0 → ค่า R_0 เปลี่ยนเป็น 0×0 เพราะเปลี่ยนให้ R_1 AND กับ R_2 แทนการ OR ค่า pc เป็น 0×103dc
   ↳ $R_0 = R_1 \& R_2$ = 0010 & 0100 = 0000
2. จงปรับแก้โปรแกรมใน main.s เป็นดังนี้ จดบันทึกผลการทดสอบและอธิบาย

```
        .global main        returned 2 (0×2)
main:
        MOV R5, #1      → นำค่าคงที่ 1 ไปเก็บใน R5
loop:
        CMP R4, #0      → สั่ง compare เทียบกัน
        BLE end         → BLE คือ ถ้าจริงตามเงื่อนไขจะทำ end
                          แต่ข้อนี้ไม่จริงเลยลงไปทำ else
else:
        MOV R5, #2      → นำค่าคงที่ 2 ไปเก็บใน R5
end:
        MOV R0, R5      → นำค่าใน R5 (R5 = 2) เก็บลงใน R0
        BX LR           → return R0 = 2
```



3. จงปรับแก้โปรแกรมใน main.s เป็นดังนี้ จดบันทึกผลการทดสอบและอธิบาย

returned 2 (0x2)

```
    .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
```

จองพื้นที่ 4 bytes

สร้างตัวแปร var 1 2 bytes (จาก word) ที่มีค่า 1

เก็บค่าคงที่ 2 ใน R_1

เอาค่า address var 1 ใน R_2

นำค่า 2 ใน R_1 ($R_1 = 2$) เขียนทับ data ใน address R_2

นำค่า 2 ใน R_2 ($R_2 = 2$) มาเก็บใน R_0

return R0 = 2