

ภาคผนวก E

การทดลองที่ 5 การพัฒนาโปรแกรมภาษา C บน ลินุกซ์

การทดลองนี้คัดว่าผู้อ่านผ่านหัวข้อที่ 3.2 และมีประสบการณ์การเขียนหรือพัฒนาโปรแกรมด้วยภาษา C มาแล้ว ผู้อ่านอาจมีความคุ้นเคยกับ IDE (Integrated Development Environment) จากการพัฒนาโปรแกรมและการดีบักโปรแกรมด้วยภาษา C/C++ ดังนั้น การทดลองมีวัตถุประสงค์เหล่านี้

- เพื่อให้เข้าใจการพัฒนาซอฟต์แวร์ด้วย IDE ชื่อ CodeBlocks บนระบบปฏิบัติการ Raspberry Pi OS/Linux/Unix
- เพื่อให้สามารถสร้าง Makefile เพื่อพัฒนาศักยภาพการทำงานเป็นนักพัฒนาอาชีพ
- เพื่อให้เข้าใจความแตกต่างระหว่างการพัฒนาโปรแกรมภาษา C ด้วย IDE และ Makefile

E.1 การพัฒนาโดยใช้ IDE

โปรแกรมหรือแอปพลิเคชัน IDE ย่อมาจาก Integrated Development Environment ทำหน้าที่ช่วยเหลือโปรแกรมเมอร์ พัฒนา ทดสอบ และอาจรวมถึงควบคุมซอฟต์แวร์สโคล์ดให้เป็นปัจจุบัน ขั้นตอนการทดลองนี้เริ่มต้นโดย

- ตรวจสอบภายในเครื่องว่ามีโปรแกรมชื่อ CodeBlocks ติดตั้งแล้วหรือไม่ โดยพิมพ์คำสั่งเหล่านี้ลงบนโปรแกรม Terminal

```
$ codeblocks
```

- หากติดตั้งแล้ว ให้ผู้อ่านข้ามไปข้อที่ 4 ได้ หากไม่มีโปรแกรม ผู้อ่านจะต้องติดตั้ง CodeBlocks โดยพิมพ์คำสั่งเหล่านี้ลงบนโปรแกรม Terminal

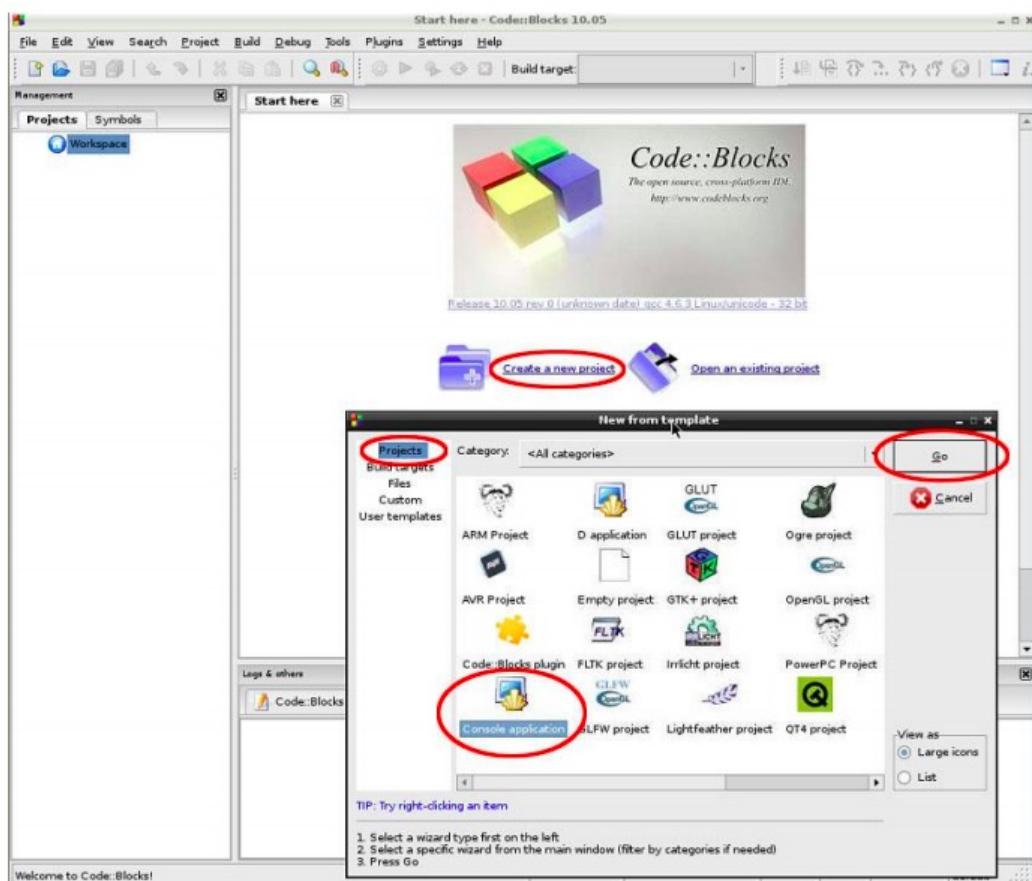
```
$ sudo apt-get install codeblocks
```

คำสั่ง sudo นำหน้าคำสั่งได้ ๆ นี้จะเป็นการเรียกใช้งานคำสั่งนั้นด้วยสิทธิ์ระดับ SuperUser การติดตั้งจะดาวน์โหลดโปรแกรมผ่านทางเครือข่ายอินเทอร์เน็ต และจำเป็นต้องใช้สิทธิ์ระดับสูงสุดนี้

- เมื่อติดตั้งเสร็จสิ้น พิมพ์คำสั่งนี้เพื่อเริ่มต้นใช้งาน CodeBlocks

```
$ codeblocks
```

- การใช้งาน CodeBlocks ครั้งแรกจะเป็นการติดตั้งค่า compiler plug-ins เป็น GCC หรือ GNU C Compiler.
- หน้าต่างหลักจะปรากฏขึ้น หลังจากนั้น ผู้อ่านควรกด "Create a new project" เพื่อสร้างโปรเจกต์ใหม่ในหน้าต่าง "New from template"



รูปที่ E.1: หน้าต่างเลือกชนิดโปรเจกต์ที่จะพัฒนาเป็นชนิด "Console application"

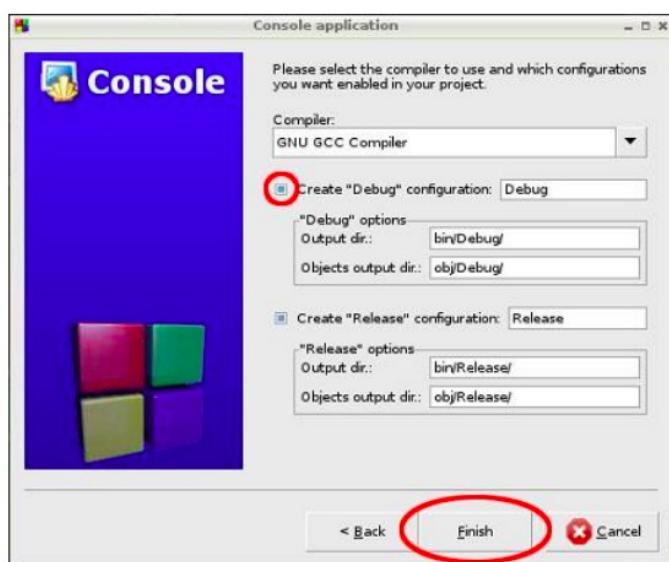
- เลือก "New Projects" ในช่องด้านซ้าย แล้วเลือก "Console application" ในช่องด้านขวาเพื่อสร้างโปรแกรมในรูปแบบเทกซ์mode (Text Mode) กดปุ่ม "Go" ตามรูปที่ E.1
- กดปุ่ม Next> เพื่อดำเนินการต่อ

8. หน้าต่าง "Console application" จะปรากฏขึ้น กดเลือกภาษา "C" เพื่อพัฒนาโปรแกรมแล้วกดปุ่ม "Next >" ตามรูปที่ E.2)



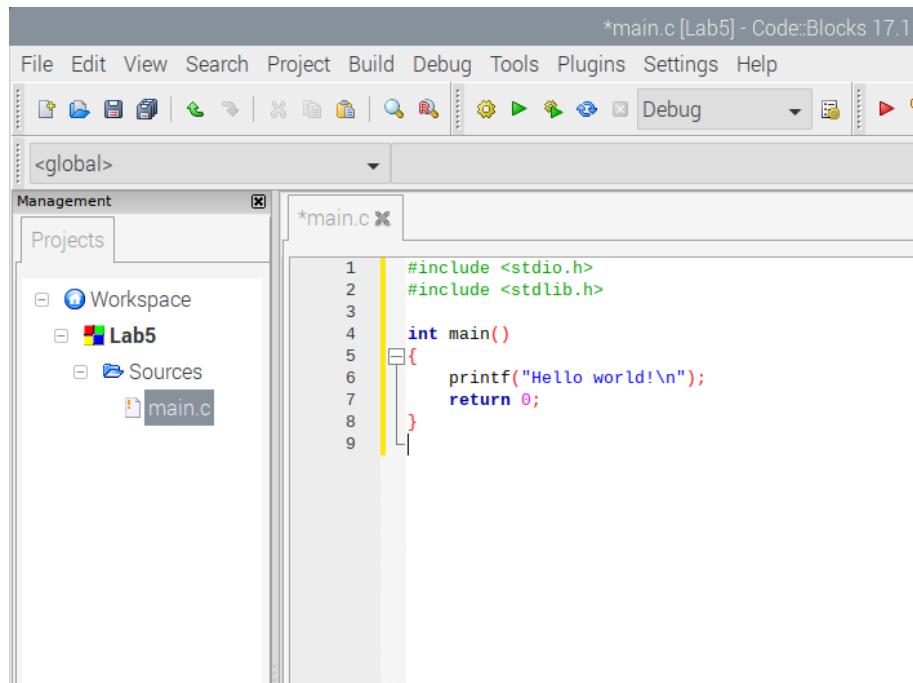
รูปที่ E.2: หน้าต่างเลือกภาษา C หรือ C++ สำหรับโปรเจกต์ที่จะพัฒนา

9. กรอกชื่อโปรเจกต์ใหม่ชื่อ Lab5 ในช่อง Project title: และกรอกชื่อไดเรกทอรี /home/pi/asm/ ในช่อง Folder to create project in: โปรดสังเกตข้อความในช่อง Project filename: ว่าตรงกับ Lab5.cbp หรือไม่
10. กดปุ่ม "Next >" เพื่อดำเนินการต่อ และสุดท้ายจะเป็นขั้นตอนการเลือกคอนฟิกกูเรชัน (Configuration) สำหรับคอมไพล์เตอร์ในรูปที่ E.3 โดย Debug เหมาะสำหรับการเริ่มต้นและแก้ไขข้อผิดพลาด แล้วจึงกดปุ่ม "Finish" เมื่อเสร็จสิ้นการติดตั้ง



รูปที่ E.3: การเลือกคอนฟิกกูเรชัน (Configuration) Debug สำหรับคอมไпал์เตอร์ GNU GCC ในโปรเจกต์ Lab5

11. คลิกช้ายบันชื่อ Lab5 ในหน้าต่าง Management/Workspace ด้านซ้ายมือ เพื่อขยายไดเรกทอรี Sources แล้วจึงคลิกบนไฟล์ไอคอนชื่อ main.c



รูปที่ E.4: การเปิดอ่านไฟล์ main.c ภายใต้โปรเจกต์ Lab5 ที่สร้างขึ้น

คำสั่งเริ่มต้นที่ CodeBlocks สร้างไว้อัตโนมัติในไฟล์ main.c คือ Hello world!

12. ป้อนโปรแกรมนี้แทนที่ของเดิมในไฟล์ main.c

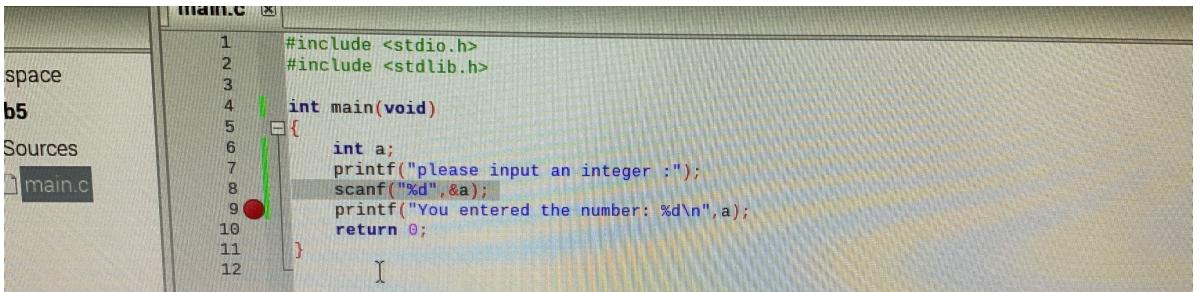
```
#include <stdio.h>
int main(void)
{
    int a;
    printf("Please input an integer: ");
    scanf("%d", &a);
    printf("You entered the number: %d\n", a);
    return 0;
}
```

13. Build->Compile โปรแกรม จนไม่มีข้อผิดพลาด โดยสังเกตจากหน้าต่างย่อยด้านล่างสุด

14. รันโปรแกรมเพื่อทดสอบการทำงาน

E.2 การดีบัก (Debugging) โดยใช้ IDE

การดีบักโปรแกรม คือ การตรวจสอบการทำงานของโปรแกรมอย่างละเอียด CodeBlocks รองรับการดีบัก ผ่านเมนู Debug ผู้อ่านสามารถเริ่มต้นโดย

1. กด Debug บนเมนูแถบบนสุด เลือก Active Debuggers GDB/CDB Debugger เป็นค่าดีฟอลท์ (Target's Default)
2. เลื่อนเมาส์ซอร์ (Cursor) ไปยังบรรทัดที่ต้องการศึกษา กดปุ่ม F5 เพื่อตั้งเบรกพอยน์ (Break Point) ตรงบรรทัดปัจจุบันของเมาส์ซอร์ โปรดสังเกตต้นประโยคด้านซ้ายสุดจะมีวงกลมสีแดงปรากฏขึ้น และเมื่อกด F5 อีกครั้งวงกลมสีแดงจะหายไป เรียกว่า การท็อคเกิล (Toggle) เบรกพอยน์ กด F5 อีกครั้งเพื่อสร้างวงกลมสีแดงตรงบรรทัดที่สนใจเพียงจุดเดียวเท่านั้น จับภาพหน้าต่างที่ได้วางไว้ได้คำสั่งนี้เพื่อให้ตรวจสอบ
 
3. กดปุ่ม F8 (Start/Continue) บนคีย์บอร์ดเพื่อรันโปรแกรมอีกรอบ โปรแกรมจะรันไปจนหยุดตรงประโยคที่มีวงกลมสีแดงนั้น โปรดสังเกตสัญลักษณ์สามเหลี่ยมสีเหลืองซ้อนทับกันอยู่ หลังจากนั้น กดปุ่ม F7 (Next line) เพื่อดำเนินการต่อทีละบรรทัด
4. เลื่อนเมาส์ซอร์ไปยังประโยคที่มีวงกลมสีแดง กดปุ่ม F5 บนคีย์บอร์ดเพื่อปลดวงกลมสีแดงออก หรือยกเลิกเบรกพอยน์
5. เริ่มต้นการดีบักใหม่เพื่อศึกษาการทำงานของปุ่ม F4 (Run to cursor) โดยเลื่อนเมาส์ซอร์ไปวางบนประโยคที่สนใจ กดปุ่ม F4 และสังเกตว่าสามเหลี่ยมสีเหลืองจะปรากฏหน้าประโยค เพื่อระบุว่า เครื่องรันนามาถึงประโยคนี้แล้ว
6. กดปุ่ม F8 เพื่อรันต่อไป จนสิ้นสุดการทำงานของโปรแกรม
7. ใช้โปรแกรมไฟล์เมเนเจอร์ค้นหาในไดเรกทอรี `/home/pi/asm/Lab5` ว่าไฟล์ `main.o` ที่เป็นไฟล์ อ็อบเจกต์อยู่ในไดเรกทอรีใด **`/home/pi/asm/Lab5/obj/Debug`**
8. ใช้โปรแกรมไฟล์เมเนเจอร์ค้นหาในไดเรกทอรี `/home/pi/asm/Lab5` ว่าไฟล์ `Lab5` ที่เป็นไฟล์ โปรแกรมหรือไฟล์ Executable อยู่ในไดเรกทอรีใด **`/home/pi/asm/Lab5/bin/Debug`**

E.3 การพัฒนาโดยใช้ประโยชน์คำสั่ง (Command Line)

ผู้อ่านควรเข้าใจคำสั่งพื้นฐานในการแปลงโปรแกรมภาษา C ที่สร้างขึ้นใน CodeBlocks ก่อนหน้านี้ ตามขั้นตอนต่อไปนี้

1. เปิดโปรแกรม Terminal หน้าต่างใหม่ แล้วป้อนโค้ดเรกทอรีไปยัง `/home/pi/asm/Lab5` โดยใช้คำสั่ง `cd`
2. ทำการคอมไพล์ (Compile) ไฟล์ซอร์ฟโค้ดให้เป็นไฟล์อ็อบเจกต์ (.o) โดยเรียกใช้คอมไเพเลอร์ชื่อ `gcc` ดังนี้

```
$ gcc -c main.c
```

ไฟล์ผลลัพธ์ ชื่อ `main.o` จะปรากฏขึ้น ผู้อ่านต้องตรวจสอบโดยใช้คำสั่ง `ls -la` เพื่อตรวจสอบวันที่และขนาดของไฟล์ เปรียบเทียบการใช้งานกับรูปที่ 3.10 จับภาพหน้าต่างที่ได้วางไว้ใต้คำสั่งนี้เพื่อให้ตรวจสอบ

```
pi@raspberrypi:~/asm/Lab5 $ ls -la
total 32
drwxr-xr-x 4 pi pi 4096 Jan 30 22:57 .
drwxr-xr-x 5 pi pi 4096 Jan 30 22:47 ..
drwxr-xr-x 3 pi pi 4096 Jan 30 22:49 bin
(-rw-r--r-- 1 pi pi 998 Jan 30 22:47 Lab5.cbp
(-rw-r--r-- 1 pi pi 98 Jan 30 22:49 Lab5.depend
(-rw-r--r-- 1 pi pi 191 Jan 30 22:49 main.c
(-rw-r--r-- 1 pi pi 1200 Jan 30 22:57 main.o
drwxr-xr-x 3 pi pi 4096 Jan 30 22:49 obj
pi@raspberrypi:~/asm/Lab5 $ ls -la
```

3. ทำการลิงก์ (Link) โดยใช้ `gcc` ทำหน้าที่เป็นลิงก์เกอร์ (Linker) และแปลงไฟล์อ็อบเจกต์เป็นไฟล์โปรแกรม (Executable File) โดย

```
$ gcc main.o -o Lab5
```

ไฟล์โปรแกรม ชื่อ `Lab5` จะปรากฏขึ้น ผู้อ่านต้องตรวจสอบโดยใช้คำสั่ง `ls -la` เพื่อตรวจสอบวันที่และขนาดของไฟล์เพื่อเปรียบเทียบกับ `man.o` จับภาพหน้าต่างที่ได้วางไว้ใต้คำสั่งนี้เพื่อให้ตรวจสอบ

```
e pi@raspberrypi:~/asm/Lab5 $ ls -la
total 40
drwxr-xr-x 4 pi pi 4096 Jan 30 22:59 .
drwxr-xr-x 5 pi pi 4096 Jan 30 22:47 ..
drwxr-xr-x 3 pi pi 4096 Jan 30 22:49 bin
-rw-r--r-- 1 pi pi 8116 Jan 30 22:59 Lab5
(-rw-r--r-- 1 pi pi 998 Jan 30 22:47 Lab5.cbp
(-rw-r--r-- 1 pi pi 98 Jan 30 22:49 Lab5.depend
(-rw-r--r-- 1 pi pi 191 Jan 30 22:49 main.c
(-rw-r--r-- 1 pi pi 1200 Jan 30 22:57 main.o
drwxr-xr-x 3 pi pi 4096 Jan 30 22:49 obj
pi@raspberrypi:~/asm/Lab5 $
```

4. รัน (Run) โปรแกรม `Lab5` โดยพิมพ์

```
$ ./Lab5
```

5. เปรียบเทียบ ผลลัพธ์ ที่ ปรากฏขึ้น ว่า ตรง กับ ผล การ รัน ใน CodeBlocks หรือ ไม่
..... เพราะเหตุใด ..
..... **เมื่อเวลาไปรัน แคปท์ ก็กรอก main.io ไม่ถูกต้อง**

E.4 โครงสร้างของ Makefile

นอกเหนือจากการพัฒนาโปรแกรมด้วย IDE แล้ว การพัฒนาด้วย Makefile จะช่วยให้นักพัฒนา มี สมัครเล่นและมืออาชีพดำเนินการได้ถูกต้องและรวดเร็ว ไฟล์ซึ่ง Makefile เป็นไฟล์อักษรหรือเทกซ์ไฟล์ (text file) ง่าย ๆ ที่อธิบายความสัมพันธ์ระหว่างไฟล์ซอฟต์แวร์ต่าง ๆ ไฟล์อืบเจกต์ และไฟล์โปรแกรม แต่ละบรรทัดจะมีโครงสร้างดังนี้

```
target : prerequisites ...
<tab>recipe
<tab>    ...
<tab>...
```

- target หมายถึง ชื่อไฟล์ที่จะถูกสร้างขึ้น โดยอาศัยไฟล์ต่าง ๆ จากส่วนที่เรียกว่า prerequisites นอกจากชื่อไฟล์แล้ว คำสั่ง 'clean' สามารถใช้เป็น target ได้ จึงนิยมใช้สำหรับลบไฟล์ต่าง ๆ ที่ไม่ต้องการ
- recipe หมายถึง คำสั่งหรือการกระทำที่จะใช้รายชื่อไฟล์ใน prerequisites นั้นมาสร้างไฟล์ target ได้สำเร็จ โดยแต่ละบรรทัดจะต้องเริ่มต้นด้วยปุ่ม tab เสมอ

E.5 การพัฒนาโดยใช้ Makefile

ตัวอย่างนี้เป็นการสร้าง Makefile เพื่อใช้คอมไฟล์และลิงก์โปรแกรมเดิมที่เรามีอยู่ ผู้อ่านจะได้เข้าใจ กลไกการทำงานที่ง่ายที่สุด ก่อน หลังจากนั้นผู้อ่านสามารถศึกษาเพิ่มเติมด้วยตนเองได้จากเว็บไซต์หรือ ตัวอย่างโปรแกรมโอเพนซอร์สที่ซับซ้อนขึ้นเรื่อย ๆ ต่อไป

1. ในโปรแกรม Terminal ย้ายไดเรกทอรีปัจจุบันไปที่ /home/pi/asm/Lab5
2. เรียกใช้โปรแกรม nano ในหน้าต่าง Terminal

```
$ nano
```

กรอกข้อความเหล่านี้ในไฟล์เปล่าโดยใช้ nano

Lab5: main.o

```
gcc main.o -o Lab5
```

main.o: main.c

```
gcc -c main.c
```

clean:

```
rm *.o
```

3. เมื่อกรอกเสร็จแล้ว ให้ทำการบันทึก หรือ save โดยตั้งชื่อไฟล์ว่า Makefile หรือ makefile อย่างใด อย่างหนึ่งโดยไม่มีนามสกุล หลังจากนั้น และบันทึกในไดเรกทอรี /home/pi/asm/Lab5 แล้วปิดโปรแกรม nano

4. พิมพ์คำสั่งนี้ใน Terminal

```
$ make clean
```

เพื่อเรียกใช้คำสั่ง rm *.o ผ่านทาง Makefile เพื่อลบ (Remove) ไฟล์ที่มีนามสกุล .o ทั้งหมด

5. พิมพ์คำสั่งนี้ใน Terminal

```
$ make Lab5
```

เพื่อเรียกใช้คำสั่ง gcc -c main.c และ gcc -g main.c -o Lab5 เพื่อสร้างไฟล์คำสั่ง Lab5 ที่จะทำงานตามชอร์สโค้ด main.c ที่กรอกไป โดยไฟล์ Lab5 ที่เกิดขึ้นใหม่จะมีโครงสร้างรูปแบบ ELF

6. พิมพ์คำสั่งนี้ใน Terminal

```
$ ls -la
```

เพื่ออ่านค่าเวลาที่ไฟล์ Lab5 ที่เพิ่งถูกสร้าง โปรดสังเกตสิ่งของข้อไฟล์ต่าง ๆ ว่ามีสีอะไรบ้าง และบ่งบอกอะไรมาตามสีนั้น ๆ

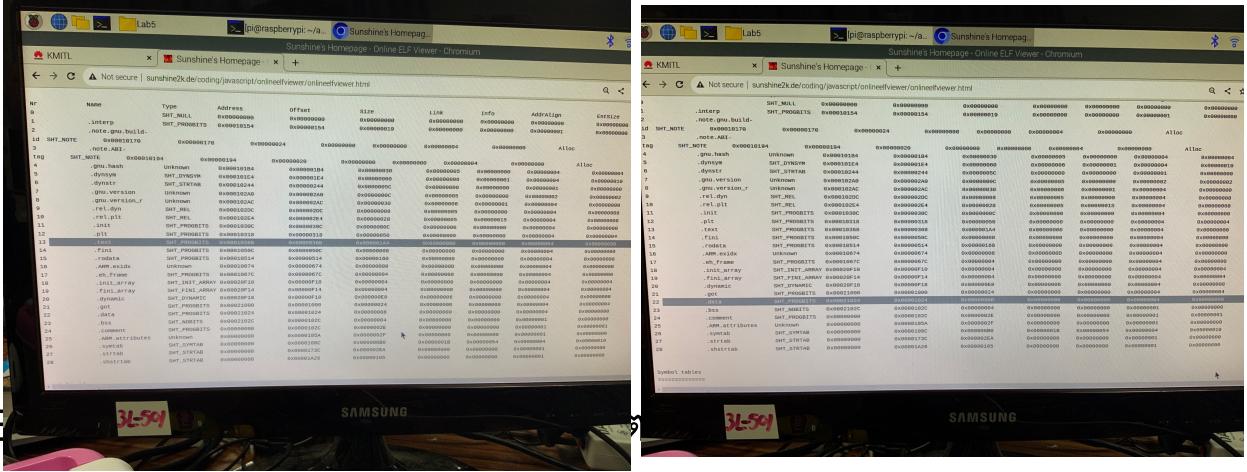
7. พิมพ์คำสั่งนี้ใน Terminal

```
$ ./Lab5
```

เพื่อรันโปรแกรม Lab5 ให้ชีพิญ ปฏิบัติ ตาม โดย . หมายถึง Current directory / หมายถึง เพื่อใน directory นั้นๆ วางแผนหน้าต่างที่ได้วางไว้ให้คำสั่งนี้เพื่อให้ตรวจสอบ

8. คลิกบนลิงก์ต่อไปนี้ sunshine2k.de เพื่อเปิดเบราว์เซอร์และอัปโหลดไฟล์ Lab5 ที่ได้จากการคอมไพล์และลิงก์ก่อนหน้านี้ ตรวจสอบค่า Status: File successfully loaded หรือไม่

9. เปิดเบราว์เซอร์ให้เต็ม จอแล้วเลื่อนหน้าจอแสดงผลขึ้นเพื่อเปรียบเทียบกับโครงสร้างของไฟล์ ELF ในรูปที่ 3.14 แคปเจอร์หน้าจอบริเวณแท็กเมนต์และดาต้าแท็กเมนต์ของ Lab5 วางภาพ



E.6.1 เลขจำนวนเต็มฐานสองไม่มีเครื่องหมาย

หัวข้อที่ 2.3.1 กล่าวถึงการบวกเลขจำนวนเต็มชนิดไม่มีเครื่องหมาย 2 จำนวนผลลัพธ์ที่ได้จะไม่มีเครื่องหมายด้วยเช่นกัน แต่การบวกเลขขนาดใหญ่ที่เข้าใกล้ค่าสูงสุด สามารถเกิดความผิดพลาดได้ เรียกว่า การเกิดโอเวอร์โฟล์ (Overflow) ในสมการที่ (2.52) ซึ่งเป็นผลสืบเนื่องจากวงจรดิจิทัลที่สามารถประมวลผลได้จำกัด ตามจำนวนบิตข้อมูลสูงสุดที่ทำได้ ในตัวอย่างการแปลงเลขฐานสองเป็นฐานสิบที่ได้แสดงไปแล้ว ยกตัวอย่างเช่น การวนรอบหรือวนลูป (Loop) เพิ่มค่าอย่างต่อเนื่องโดยไม่ระวัง ตามประโยชน์ในภาษา C/C++ ตามการทดลองต่อไปนี้

1. ทดสอบโปรแกรมภาษา C ต่อไปนี้

```
#include <stdio.h>

int main()
{
    unsigned int i=1;

    while (i>0) {
        i=i+1;
        if (i==0) {
            printf("i was %10u before\n", i-1);
            printf("i is %10u now\n", i);
        }
    }
    return 0;
}
```

}

2. อธิบายว่า 프로그래มตัวแปรจึงตั้งค่าเริ่มต้น unsigned int i=1; ... หมายเหตุ i=0 จะไม่เป็น loop while >0

โปรแกรมซึ่งมีปัญหา

3. การวนลูปเพิ่มค่า i=i+1 จน i มีค่าเป็นศูนย์แล้วแสดงผลค่าของ i มาทางหน้าจอ เมื่อเกิดเหตุการณ์ อะไร เพราะเหตุใด ... ก็ต Overflow เนื่องจากตัวบันทึกค่า unsigned int เก็บได้ 32 บิต กว้างกว่า 32 บิต

4. จับภาพ

```
*main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     unsigned int i=1;
7     while (i>0){
8         i=i+1;
9         if (i==0){
10             printf("i was %10u before\n",i-1);
11             printf("i is %10u now\n",i);
12         }
13     }
14     return 0;
15 }
16 
```

test

```
i was 4294967295 before
i is 0 now
Process returned 0 (0x0) execution time : 35.354 s
Press ENTER to continue.
```

E.6.2 เลขจำนวนเต็มฐานสองมีเครื่องหมายชนิด 2's Complement

หัวข้อที่ 2.3.2 กล่าวถึง การ บวก เลขจำนวนเต็ม ชนิด มี เครื่องหมาย 2 จำนวน ผลลัพธ์ที่ได้ จะ มี เครื่องหมายด้วย เช่น กัน แต่ การบวกเลขขนาดใหญ่ที่เข้าใกล้ค่าสูงสุด สามารถเกิดความผิดพลาดได้ เรียกว่า การเกิดโอเวอร์โฟล์ว (Overflow) ในสมการที่ (2.56) ซึ่งเป็นผลสืบเนื่องจากการตัดต่อของตัวเลขที่สามารถประมวลผลได้จำกัด ตามจำนวนบิตข้อมูลสูงสุดที่ทำได้ ในตัวอย่างการแปลงเลขฐานสองเป็นฐานสิบที่ได้แสดงไปแล้ว ยกตัวอย่าง เช่น การวนรอบหรือวนลูป (Loop) เพิ่มค่าอย่างต่อเนื่องโดยไม่ระวาง ตามประ予以ค ในภาษา C/C++ ตามการทดลองต่อไปนี้

1. ทดสอบโปรแกรมภาษา C ต่อไปนี้

```
#include <stdio.h>
int main()
{
    int i=1;
    while (i>0) {
        i=i+1;
        if (i<0) {
            printf("i was %10d before\n", i-1);
            printf("i is %10d now\n", i);
            break;
        }
    }
}
```

```

        }
    }

    return 0;
}

```

2. อธิบายว่า ประการตัวแปรจึงตั้งค่าเริ่มต้น int i=1; เมื่อรันที่ i=0 จะเกิดอะไรขึ้น while (i > 0) ทำอะไรบ้าง

3. การวนลูปเพิ่มค่า i=i+1 จน i มีค่าน้อยกว่าศูนย์แล้วแสดงผลค่าของ i มาทางหน้าจอ เมื่อเกิดเหตุการณ์อะไร เพราะเหตุใด ... เกิด Overflow ทราบพาร์ทห์ เกี่ยวกับ ค่า int เป็นไปได้ดังนี้ในบรรทัด ทางข้างล่าง
ที่แสดงตัวอย่าง int

4. จับภาพหน้าต่างที่ได้วางไว้ใต้คำสั่งนี้เพื่อให้ตรวจสอบ

```

main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     int i=1;
7     while (i>0){
8         i=i+1;
9         if (i<0){
10            printf("i was %10u before\n",i-1);
11            printf("i is %10u now\n",i);
12            break;
13        }
14    }
15    return 0;
16 }
17

```

1. จะเปรียบเทียบฟล๊อว์การพัฒนาโปรแกรมในภาคผนวกนี้กับรูปที่ 3.9

2. ใน Terminal จะบัญใจเรก thor ปัจจุบันไปที่ /home/pi/asm/Lab5

\$ ls -la

เพื่ออ่านรหัสสีของชื่อไฟล์ต่าง ๆ

3. จะพัฒนาโปรแกรมภาษา C โดยประการตัวแปรและตั้งค่าเริ่มต้น int i=-1 และให้วนลูปลดค่า i=-1 จน i มีค่าเป็นบวกแล้วแสดงผลค่าของ i ออกมาทางหน้าจอ โดยใช้โปรแกรมในหัวข้อที่ E.6.2 เป็นต้นแบบ
4. จะพัฒนาโปรแกรมภาษา C ให้สามารถอ่านไฟล์ Makefile เพื่อแสดงตัวอักษรในไฟล์ทีละตัวและค่ารหัส ASCII ฐานสิบหกของตัวอักษรนั้นบนหน้าจอ แล้วปิดไฟล์เมื่อเสร็จสิ้น

5. จะพัฒนาโปรแกรมภาษา C เพื่อสั่งพิมพ์เลขอนุกรม Fibonacci โดยรับค่าเลขเป้าหมาย n ซึ่งเกิดจาก $n = (n-1) + (n-2)$ และรายละเอียดเพิ่มเติมได้จาก [wikipedia](#) ตัวอย่างต่อไปนี้ n=5 และพิมพ์ผลลัพธ์ดังนี้

1 1 2 3 5

```

main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     int i=-1;
7     while (i<0){
8         i=i-1;
9         if (i>0){
10            printf("i was %10u before\n",i-1);
11            printf("i is %10u now\n",i);
12            break;
13        }
14    }
15    return 0;
16 }
17

```