204324 ปฏิบัติการระบบคอมพิวเตอร์

ปฏิบัติการที่ 5: การผลิตโค๊ดสำหรับ expression

ปฏิบัติการนี้เราจะผลิตโค๊ด MIPS assembly อัตโนมัติ เพื่อคำนวณผลลัพธ์ของ expression ทางคณิตศาสตร์ที่รับเข้ามาในรูปแบบ infix เช่น (5 + 31) * 4 จากนั้นให้นำโค๊ด assembly ที่ผลิตได้ไปรันใน SPIM simulator ที่เราได้ใช้งานในปฏิบัติการก่อนหน้านี้เพื่อตรวจสอบความถูกต้องในขั้นสุดท้าย

เริ่มต้นจากโค๊ดการสร้าง expression tree กำหนดให้ operand ของ expression ที่รับเข้ามามีเพียงค่าคงที่ที่เป็นจำนวนเต็มเท่านั้น (ไม่ให้มีตัวแปร) ในการผลิตโค๊ด MIPS assembly จาก expression tree ให้ยึดสถาปัตยกรรมแบบ 1-register stack machine ตามเอกสารประกอบปฏิบัติการนี้ โดยขั้นตอนการผลิตโค๊ดมีแนวคิดหลักโดยใช้ recursion ดังต่อไปนี้

1. เราจะ traverse sub-tree ทางด้านซ้ายแบบ recursive เพื่อผลิตโค๊ดทางฝั่งนี้ให้เรียบร้อย จนได้ผลลัพธ์สุดท้ายของการคำนวณค่า expression ทาง sub-tree ฝั่งนี้อยู่ที่ accumulator จากนั้นเราจะ push ค่าใน accumulator ลงบน stack
2. ต่อมาเราจะ traverse sub-tree ทางด้านขวาแบบ recursive เช่นเดียวกับที่เราทำในขั้นตอนแรก จนได้ผลลัพธ์สุดท้ายของการคำนวณค่า expression ทาง sub-tree ฝั่งนี้อยู่ที่ accumulator แต่ในครั้งนี้เราไม่ต้อง push ค่านี้ลง stack
3. จากนั้นเราดูว่า operator ของ node ที่เราเริ่มต้น traverse ไปที่ sub-tree ทั้งสองฝั่งคือ operator อะไร แล้วเราก็จะนำ operand ที่ 1 (operand ฝั่งซ้ายของ operator นี้) ที่อยู่ที่ top-of-stack และ operand ที่ 2 (operand ฝั่งขวาของ operator นี้) ที่อยู่ที่ accumulator มาประมวลผลตาม operator นั้น

โค๊ด MIPS assembly ที่ผลิตได้ ให้นิสิตเขียนลงไฟล์ที่มีนามสกุล .asm และไฟล์นี้จะต้องมีโครงสร้างดังต่อไปนี้

```
.text # text section
.globl main # call main by SPIM
main:
# ใส่โค๊ด MIPS assembly ที่ผลิตได้จากการ traverse expression tree บริเวณนี้
li $v0, 1 # for printing an integer result
syscall  # for printing an integer result
li $v0, 10 # for correct exit (or termination)
syscall    # for correct exit (or termination)
```

เซฟโปรแกรมที่ผลิตโค๊ด MIPS assembly ไว้ในไฟล์ชื่อ **expr_compiler.c** จากนั้นทดสอบโปรแกรมกับตัวอย่างทดสอบต่อไปนี้ว่า โค๊ดที่ผลิตออกมาให้ผลลัพธ์ถูกต้อง โดยนำโค๊ดนี้ไปรันใน SPIM

- 12 * 3 / (2 + 7) - 5 % 2
- (20 - 10) * (30 / 4 + 40) % (5 + 6)
- 11 + 22 * 33 % 12 - 44 / 3 - 7

นอกจากนั้นให้สร้าง expression compiler สำหรับ stack machine ที่แท้จริง โดยแทนที่จะผลิตโค๊ด MIPS แบบ 1-register stack machine ให้ผลิตโค๊ดสำหรับ CPU ที่ใช้สถาปัตยกรรมแบบ stack อย่างแท้จริง กล่าวคือใช้เพียงคำสั่ง push pop add mul div และ mod เท่านั้น เซฟโปรแกรมนี้ในไฟล์ชื่อ **expr_stack_compiler.c**

สิ่งที่ต้องส่งในชั่วโมง

- แบบฝึกหัดที่ทำในชั้นเรียน

สิ่งที่ต้องสิ่งหลังชั่วโมงปฏิบัติการ

- ไฟล์ **expr_compiler.c** และไฟล์ **expr_stack_compiler.c**
- ไฟล์ README เพื่อระบุสถานะว่าทำเสร็จถึงส่วนใด มีข้อผิดพลาดอะไรบ้าง กรณีที่ทำไม่เสร็จให้ชี้แจงเหตุผลด้วย

การส่งงาน:

- **นำงานที่ต้องส่งหลังจากชั่วโมงปฏิบัติการใส่ไว้ที่โฟลเดอร์ studentID1_firstname1_studentID2_firstname2_lab5 โดย studentID และ firstname คือเลขประจำตัวและชื่อแรกของสมาชิกที่ทำปฏิบัติการร่วมกัน จากนั้น zip โฟลเดอร์นี้แล้วส่ง zip ไฟล์มาที่ Google Classroom ก่อนกำหนดส่ง**