ISA Assignment A04: Pico-Assembler

อ่าน Patterson and Hennessy's Computer Organization & Design ภาคผนวก Assemblers, Linkers, and the SPIM Simulator (ภาคผนวก A ใน 2nd และ 3rd Edition แต่เป็น ภาคผนวก B ใน 4th Edition)

แอสเซมเบลอร์ ทำหน้าที่ (1) อ่านข้อความ (ที่เขียนด้วยภาษาแอสเซมบลี) ในไฟล์ (2) แปลงคำสั่งภาษาแอสเซมบลีไปเป็นคำสั่งภาษาเครื่องที่เป็นเลขฐานสอง และ (3) เขียนคำสั่งภาษาเครื่องนั้นลงในอีกไฟล์หนึ่ง

งานนี้เราจะเน้นที่การแปลงคำสั่งภาษาแอสเซมบลีไปเป็นคำสั่งภาษาเครื่อง ซึ่งมีขั้นตอนย่อยที่สำคัญคือ
(i) การระบุแอสเดรส (address) ของหน่วยความจำ ที่เขียนเป็นฉลาก (label) เช่น

```
myasm: .asciiz      "asm \njr $ra\n ISA \njr $ra\nMIPS32;"

            .globl main
            .text
main:
      la $s0, myasm            # $s0 <- base address of myasm
            :
```

แอสเซมบลี อาจทำงานโดย การอ่านโปรแกรมไปก่อนหนึ่งครั้ง และ สร้างตารางเชื่อม (mapping table) ระหว่างฉลากและแอดเดรส เช่น

myasm	0x 1000 0000
main	0x 0040 0000
:	:

คำถามทบทวนเนื้อหา ทำไม myasm ถูกกำหนดให้ 0x 1000 0000 เป็นแต่ main ถูกกำหนดให้เป็น 0x 0040 0000 ทำไมไม่ต่อกัน และทำไมไม่เริ่มต้นที่ 0x 0000 0000 ?
หมายเหตุ จริงๆแล้ว main ถูกกำหนดเป็น 0x 0040 0024 คุณว่าเป็นเพราะอะไร

(ii) การแปลงคำสั่งภาษาแอสเซมบลีแต่ละคำสั่ง ไปเป็น เลขฐานสอง ซึ่งประกอบไปด้วย ตัวเลขของ opcode, ตัวเลขที่แทนรีจีสเตอร์ (register), ตัวเลขที่แทนฉลาก, ตัวเลขของค่าตรงๆ (immediate) ในลำดับที่ถูกต้อง
เช่น addi $s6, $s6, 3 จะถูกแปลงเป็น 0x22d60003.

A04pico.asm

โปรแกรม A04pico.asm เป็น โปรแกรมแอสเซมเบลอร์อย่างง่ายทำหน้าที่แปลงคำสั่งแค่คำสั่งเดียว คือ `addi $s6, $s6, 3` ไปเป็นคำสั่งภาษาเครื่องที่ถูกต้อง (ซึ่งคือ 0x22d60003) และแปลงอย่างอื่นเป็น 0x00000000.

1. ลองโปรแกรม A04pico.asm ทำความเข้าใจการทำงานของมัน

2. ตอบคำถาม/ทำ

(a) หาลำดับของตัวอักษรที่ทำให้ A04pico.asm ทำงานพลาดจากที่มันควรจะทำ (ซึ่ง คือ แปลง `addi $s6, $s6, 3` ไปเป็น `0x22d60003` และแปลงอย่างอื่นเป็น `0x00000000`)

(b) white space มีผลกับการทำงานของ A04pico.asm หรือไม่ อย่างไร

(c) ปรับปรุง A04pico.asm ให้สามารถจัดการกับ white space ได้

(bonus) เพิ่มความสามารถของ A04pico.asm เพื่อให้รับคำสั่งอื่นได้ (เช่น `jr $ra`) หรือ operand อื่นได้ (เช่น `addi $s0, $s1, 28`)

ข้อควรระวัง

การใช้งานหน่วยความจำในโหมด word ของ SPIM (เช่น `lw`) แอดเดรสต้องลงท้ายด้วย `0x0`, `0x4`, `0x8` หรือ `0xC`

ตัวอย่าง รันคำสั่ง `lw $t0, 0($s0)`

ถ้า `$s0 = 0x10010000` ทำงานได้

ถ้า `$s0 = 0x10010004` ทำงานได้

ถ้า `$s0 = 0x10010008` ทำงานได้

ถ้า `$s0 = 0x1001000C` ทำงานได้

แต่ถ้า `$s0 = 0x1001000A` ทำงานไม่ได้ (error)