204324 ปฏิบัติการระบบคอมพิวเตอร์

ปฏิบัติการที่ 3: สร้าง single-cycle MIPS CPU บน Logisim

ในปฏิบัติการนี้เราจะสร้าง CPU เพื่อประมวลผลคำสั่ง MIPS assembly ที่เราได้ใช้ในการโปรแกรมในสองปฏิบัติการที่ผ่านมา การได้ฝึกปฏิบัติในครั้งนี้จะทำให้นิสิตเข้าใจกลไกการทำงานของฮาร์ดแวร์ในระดับล่างอย่างถ่องแท้ และจะเป็นพื้นฐานที่ติดตัวไปจนกว่าคอมพิวเตอร์แบบควอนตัมจะเข้ามาแทนที่คอมพิวเตอร์ปัจจุบันที่ใช้สถาปัตยกรรม Von Neumann

สิ่งแรกที่จะต้องทำ

- ดาวน์โหลดไฟล์ Lab3.zip ที่ภายในจะมี Logisim เวอร์ชั่นล่าสุด (รุ่น evolution โดย Window ให้ใช้ jar ไฟล์ ส่วน Mac ให้ใช้ zip ไฟล์)
- ทบทวนการใช้งาน Logisim โดยเฉพาะอย่างยิ่ง 1) การสร้าง sub-circuit 2) การใช้งาน Combinational Analysis (ในเมนู Window)
- ศึกษาฟอร์แมทของคำสั่ง MIPS ในระดับ machine code โดยเฉพาะในส่วน opcode และ function ให้ทำความเข้าใจ Table 1 และ Table 2 ต่อไปนี้ให้เข้าใจอย่างถ่องแท้

Table 1: MIPS32 Encoding of the Opcode Field

opcode		bits 28..26 →							
		0	1	2	3	4	5	6	7
↓ bits 31..29		000	001	010	011	100	101	110	111
0	000	*SPECIAL δ*	*REGIMM δ*	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	001	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	010	*COP0 δ*	*COP1 δ*	*COP2 θδ*	*COP3 θδ*	*BEQL φ*	*BNEL φ*	*BLEZL φ*	*BGTZL Φ*
3	011	β	β	β	β	*SPECIAL2 δ*	*JALX ε*	ε	*
4	100	LB	LH	LWL	LW	LBU	LHU	LWR	β
5	101	SB	SH	SWL	SW	β	β	SWR	CACHE
6	110	LL	LWC1	LWC2 θ	PREF	β	LDC1	LDC2 θ	β
7	111	SC	SWC1	SWC2 θ	*	β	SDC1	SDC2 θ	β

Table 2: MIPS32 *SPECIAL* Opcode Encoding of the Function Field

function		bits 2..0 →							
		0	1	2	3	4	5	6	7
↓ bits 5..3		000	001	010	011	100	101	110	111
0	000	SLL	*MOVCI δ*	SRL	SRA	SLLV	*	SRLV	SRAV
1	001	JR	JALR	MOVZ	MOVN	SYSCALL	BREAK	*	SYNC
2	010	MFHI	MTHI	MFLO	MTLO	β	*	β	β
3	011	MULT	MULTU	DIV	DIVU	β	β	β	β
4	100	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
5	101	*	*	SLT	SLTU	β	β	β	β
6	110	TGE	TGEU	TLT	TLTU	TEQ	*	TNE	*
7	111	β	*	β	β	β	*	β	β

เราจะสร้าง CPU เพื่อประมวลผลคำสั่ง MIPS ใดบ้าง

เราจะไม่ได้สร้าง CPU เพื่อประมวลผลทุกคำสั่งใน MIPS แต่เฉพาะคำสั่งที่ใช้งานมากดังจัดกลุ่มตาม Table A และ B ด้านล่างนี้

Table A

Immediate arithmetic	ADDIU, ANDI, ORI, XORI, SLTI, SLTIU
Register arithmetic	ADDU, SUBU, AND, OR, XOR, NOR, SLT, SLTU
Shifts (constant)	SLL, SRL, SRA

Table B

Jumps	J, JR, JAL
Branches	BEQ, BNE
Memory Load/Store	LW, SW

Component ที่เลือกมาใช้งานได้

เลือก component จาก CS3410 Components มาได้อย่างละหนึ่งชิ้นจาก component ต่อไปนี้

- Register File
- MIPS Program ROM
- MIPS ALU
- MIPS RAM
- Incrementer

เลือก component จากแฟ้มอื่นๆได้เต็มที่ ไม่มีข้อจำกัด แต่ สัญญาณนาฬิกา (clock) มีได้เพียงหนึ่งเดียวเท่านั้น โดยรายละเอียดของ component ใน CS3410 Components มีอยู่ในลิงค์ต่อไปนี้

http://www.cs.cornell.edu/courses/cs3410/2018fa/logisim/components.html

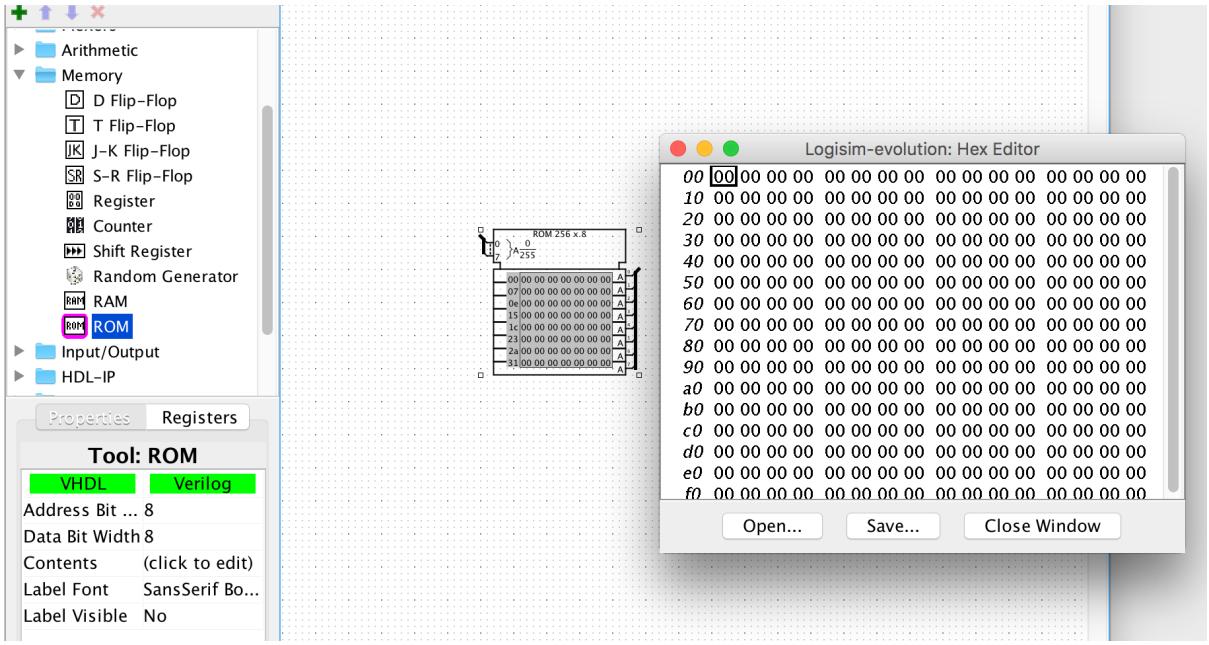
รายละเอียดของ component อื่นๆใน Logisim อยู่ที่ Help->Library Reference

ต้องทำอะไรจริงๆในปฏิบัติการนี้

- สร้าง datapath เพื่อเชื่อม component ต่างๆที่ได้เลือกมา
- สร้าง control เพื่อควบคุมการประมวลผล การทำงานของ ALU และ component อื่นๆ
- สร้างไฟล์ขึ้นมาเพื่อทดสอบความถูกต้องของ CPU ที่สร้างขึ้น

MIPS ALU จาก CS3410 Components ไม่มีขา overflow ออกมา ดังนั้นในเพื่อรองรับคำสั่งพวก SLT ให้นำ comparator จากโฟลเดอร์ของ Arithmetic มาใช้งาน (ถ้า ALU มี overflow งานของเราจะง่ายลงมากเพราะการเช็ค A < B เราแค่เลือก A - B แล้วเช็ค signed bit XOR overflow ก็จะบอกได้ว่า A < B หรือไม่)

งานที่ท้าทายสำหรับปฏิบัติการนี้จริงๆก็คือการสร้าง control โดยแนวทางที่เราแนะนำให้นิสิตจัดการกับการผลิตสัญญาณ control คือ ใช้ ROM component ภายใน ROM แต่ละแถวจะบรรจุสัญญาณ control ของแต่ละคำสั่ง และ ROM นี้จะใช้ opcode (และ/หรือ function) เป็น address เพื่อเข้าถึงสัญญาณ control สำหรับคำสั่งนั้นๆ การเลือกใช้งาน ROM และเข้าแก้ไข content ใน ROM ดูจากภาพประกอบด้านล่าง



เพื่อให้แน่ใจว่านิสิตเข้าใจเรื่องสัญญาณ control ใน MIPS CPU จริงๆ ก่อนจะลงมือสร้าง MIPS CPU อย่างจริงจัง แต่ละกลุ่มจะต้องทำแบบฝึกหัดที่เกี่ยวข้องที่แนบมากับปฏิบัติการนี้เสียก่อน

การทดสอบ

2 ไฟล์แรกมีมาให้แล้วคือ test0.s และ test1.s

MIPS Program ROM (ไม่ใช่ control ROM ที่เราได้พูดถึงก่อนหน้านี้) มี assembler มาให้แต่เป็น assembler ที่มีข้อจำกัดพอควร ดังนั้นจึงต้องเขียนไฟล์ assembly ให้เป็นไปในลักษณะตัวอย่างที่ให้มา อย่าให้มี directive พวก .text .data .word .space ใดๆปรากฏออกมา และอย่าให้ต้องมีการเรียกใช้ syscall

ถ้าไม่สามารถ load โปรแกรมลง ROM เพราะมี error จะต้องกลับไปแก้ไขที่ไฟล์ assembly เพื่อให้ผ่านกระบวนการ assembler แล้วนำ machine code ที่เป็น binary โหลดลง ROM ได้

ขอให้นิสิตสร้างโปรแกรมง่ายๆมาทดสอบคำสั่งทุกๆคำสั่งที่ระบุไว้ใน Table A และ B ให้ครบถ้วน ต้องเขียนอธิบายด้วยว่า กรณีทดสอบของเราทำอะไร ดูผลลัพธ์ได้จากค่า register หรือ memory ที่ตำแหน่งใด

เมื่อมั่นใจว่า CPU ทำงานได้อย่างถูกต้องแล้ว ให้ทดสอบกับโปรแกรมที่มีความซับซ้อน 2 โปรแกรมต่อไปนี้

1)

```
int r_hailstone(int n) {
    if (n == 1) return 0;
    else if ((n % 2) == 0) return 1 + r_hailstone(n/2);
    else return 1 + r_hailstone(3*n+1);
}
```

2)

```
int i_hailstone(int n) {
    int i = 0;
    while (n != 1) {
        i = i + 1;
        if ((n % 2) == 0)
            n = n/2;
        else
            n = 3*n+1;
    }
    return i;
}
```

โดยโค้ด assembly ของฟังก์ชั่นทั้งสองอันนี้ (เขียนโดยพี่ TA พรมนัส หอมเกษร) ได้ให้รวมมากับไฟล์อื่นๆที่ต้องใช้ในปฏิบัติการในไฟล์ Lab3.zip แล้ว ลองศึกษาดูว่าเมื่อเราทดสอบด้วยค่า n ใดๆแล้ว ผลลัพธ์สุดท้ายจะไปอยู่ที่ไหน

ต้องเขียนอธิบายด้วยว่า กรณีทดสอบของเราทำอะไร ดูผลลัพธ์ได้จากค่า register หรือ memory ที่ตำแหน่งใด

สิ่งที่ต้องส่งในสัปดาห์แรก

- คำตอบของแบบฝึกหัดที่เกี่ยวข้องกับ control และ datapath ที่แนบมากับปฏิบัติการนี้
- Datapath และ control ของ MIPS CPU ที่ทำให้คำสั่ง addiu และ addu ใดๆ สามารถประมวลผลได้ถูกต้อง (ให้ใช้ชื่อไฟล์ MIPS_add.circ) ดูตัวอย่างการเขียนกรณีทดสอบนี้ได้จากไฟล์ test0.s
- โปรแกรมทดสอบคำสั่งทุกๆคำสั่งที่ระบุไว้ในตาราง A และ B และไฟล์ README_test_programs.pdf ที่อธิบายโปรแกรมทดสอบทุกโปรแกรมว่าทำอะไร ดูผลลัพธ์ได้จากค่า register หรือ memory ที่ตำแหน่งใด **โปรแกรมทดสอบเหล่านี้ต้องผ่านการ assembler โดย MIPS Program ROM** (เรากำลังส่งเสริมให้นิสิตทำ Test-Driven Development กล่าวคือก่อนจะไปสร้างฮาร์ดแวร์จริง เราสร้างกรณีทดสอบมาก่อน)

สิ่งที่ต้องส่งในสัปดาห์ที่สอง

- Single-cycle MIPS ที่สมบูรณ์ในไฟล์ mips-single.circ (จะต้องรัน r_hailstone และ i_hailstone ผ่าน)
- ไฟล์ assembly ที่ใช้ในการทดสอบทั้งหมด

- วิดิโอแสดงการทำงานของวงจรใน mips-single.circ ตั้งแต่เริ่มต้นจนจบการประมวลผล r_hailstone และ i_hailstone พร้อมชี้จุด input และ output ที่ถูกต้องให้ชัดเจน

การส่งงาน:

- **นำงานที่ต้องส่งในสัปดาห์ที่หนึ่งใส่ไว้ในโฟลเดอร์ชื่อ**

studentID1_firstname1_studentID2_firstname2_lab3_part_1

โดย studentID และ firstname คือเลขประจำตัวและชื่อแรกของสมาชิกที่ทำปฏิบัติการร่วมกัน จากนั้น zip โฟลเดอร์นี้แล้วส่ง zip ไฟล์มาที่ Google Classroom ของวิชาก่อนกำหนดส่ง

- **นำงานที่ต้องส่งในสัปดาห์ที่สองใส่ไว้ในโฟลเดอร์ชื่อ**

studentID1_firstname1_studentID2_firstname2_lab3_part_2

โดย studentID และ firstname คือเลขประจำตัวและชื่อแรกของสมาชิกที่ทำปฏิบัติการร่วมกัน จากนั้น zip โฟลเดอร์นี้แล้วส่ง zip ไฟล์มาที่ Google Classroom ของวิชาก่อนกำหนดส่ง

ถ้าถึงกำหนดส่งแล้วนิสิตยังทำปฏิบัติการไม่สมบูรณ์ ขอให้ชี้แจงอุปสรรค บั๊กต่างๆที่พบมาในไฟล์ README.pdf แล้วใส่ไฟล์นี้มาในโฟลเดอร์เดียวกับงานอื่นๆที่ต้องส่ง