

ปฏิบัติการที่ 3: สร้าง single-cycle MIPS CPU บน Logisim

ในปฏิบัติการนี้เราจะสร้าง CPU เพื่อประมวลผลคำสั่ง MIPS assembly ที่เราได้ใช้ในการโปรแกรมในสองปฏิบัติการที่ผ่านมา การได้ฝึกปฏิบัติในครั้งนี้จะทำให้เราเข้าใจกลไกการทำงานของฮาร์ดแวร์ในระดับล่างอย่างถ่องแท้ และจะเป็นพื้นฐานที่ติดตัวไปจนกว่าคอมพิวเตอร์แบบควอนตัมจะเข้ามาแทนที่คอมพิวเตอร์ปัจจุบันที่ใช้สถาปัตยกรรม Von Neumann

สิ่งแรกที่จะต้องทำ

- ศึกษาฟอร์แมตของคำสั่ง MIPS ในระดับ machine code
- ดาวน์โหลด Logisim เวอร์ชันล่าสุด (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)

เราจะสร้าง CPU เพื่อประมวลผลคำสั่ง MIPS ไດบ้าง

เราจะได้สร้าง CPU เพื่อประมวลผลทุกคำสั่งใน MIPS แต่เฉพาะคำสั่งที่ใช้งานมากดังจัดกลุ่มตามตาราง A และ B ด้านล่างนี้

Table A	
Immediate arithmetic	ADDIU, ANDI, ORI, XORI, SLTI, SLTIU
Register arithmetic	ADDU, SUBU, AND, OR, XOR, NOR, SLT, SLTU
Move	MOVN, MOVZ
Shifts (constant and variable)	SLL, SRL, SRA, SLLV, SRLV, SRAV
Immediate load	LUI

Table B	
Jumps	J, JR, JAL,
Branches	BEQ, BNE
Memory Load/Store (little endian)	LW, LB, LBU, SW, SB

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 $\theta\delta$	COP3 $\theta\delta$	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

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

การทดสอบ

2 ไฟล์แรกมีมาให้แล้วคือ test0.s และ test1.s

MIPS Program ROM 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

ขอให้นิสิตสร้างโปรแกรมง่ายๆมาทดสอบคำสั่งต่างๆคำสั่งที่ระบุไว้ในตาราง A และ B ให้ครบถ้วน

เมื่อมั่นใจว่า CPU ทำงานได้อย่างถูกต้องแล้ว ให้ทดสอบกับโปรแกรมที่มีความซับซ้อน 2 โปรแกรมต่อไปนี้ (ต้องเขียน assembly เพื่อโหลดลง ROM ให้ได้ก่อน)

```
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;
    }
```

ต้องเขียนอธิบายด้วยว่า กรณีทดสอบของเราทำอะไร ดูผลลัพธ์ได้จากค่า register หรือ memory ที่ตำแหน่งใด

สิ่งที่ต้องส่งในชั่วโมง

- Datapath ของ CPU ที่สมบูรณ์ที่ทำให้คำสั่ง addiu ใดๆ สามารถประมวลผลได้ถูกต้อง (ให้ใช้ชื่อไฟล์ addiu.circ)
- แผนการดำเนินการเพื่อให้ส่งปฏิบัติการนี้ได้ทันภายในหนึ่งสัปดาห์หลังจบปฏิบัติการในวันนี้

สิ่งที่ต้องส่งหลังชั่วโมงปฏิบัติการ

- Single-cycle MIPS ที่สมบูรณ์ในไฟล์ mips-single.circ
- ไฟล์ assembly ที่ใช้ในการทดสอบทั้งหมด
- วิดีโอแสดงการทำงานของวงจรใน mips-single.circ ตั้งแต่เริ่มต้นจนจบการประมวลผล r_hailstone และ i_hailstone พร้อมชี้จุด input และ output ที่ถูกต้องให้ชัดเจน

การส่งงาน:

- นำงานที่ต้องส่งในชั่วโมงปฏิบัติการใส่ไว้ในโฟลเดอร์ชื่อ studentID_firstname_lastname_lab3_part1 จากนั้น zip โฟลเดอร์นี้แล้วส่ง zip ไฟล์มาที่ Google Classroom ของวิชา *ภายในชั่วโมงปฏิบัติการ*
- นำงานที่ต้องส่งหลังจากชั่วโมงปฏิบัติการใส่ไว้ในโฟลเดอร์ studentID_firstname_lastname_lab3_part2 จากนั้น zip โฟลเดอร์นี้แล้วส่ง zip ไฟล์มาที่ Google Classroom ก่อนกำหนดส่ง