Tufts COMP 140 - Advanced Architecture Summer 2014

MIPS ALU Practice

Goals:

- **Use Logisim to become familiar with the MIPS ALU**

Getting Started

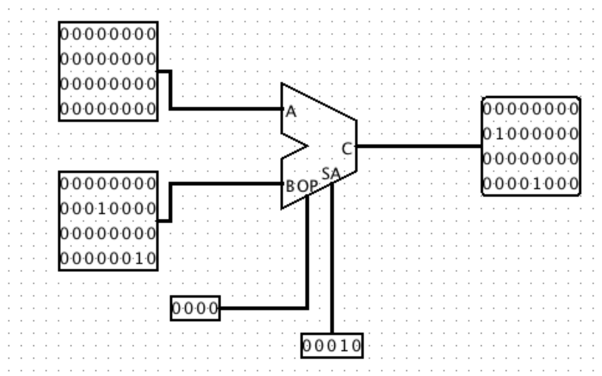
Step 1. Download the cs3410.jar file from the class website (thanks to Cornell's CS 3410 class for designing this):

- http://www.cs.tufts.edu/comp/140/labs/lab3/cs3410.jar

Step 2. Open Logisim, and load the jar file

- Project→Load Library→JAR Library

Build the following simple circuit



This circuit consists of an ALU from the cs3410 library, two 32-bit inputs (A and B), an OP input, a Shift Amount input, and a 32-bit output (C).

Test the inputs and outputs with real numbers, based on the MIPS ALU operations below:

```
MIPS ALU. Computes a result as follows:

Op     name                          C
---- -----------------------      --------------
000x   shift left                 C = B << Sa;
001x   add                        C = A + B
0100   shift right logical        C = B >>> Sa
0101   shift right arithmetic     C = B >> Sa
011x   subtract                   C = A - B
1000   and                        C = A & B
1010   or                         C = A | B
1100   xor                        C = A ^ B
1110   nor                        C = ~(A | B)
1001   eq                         C = (A == B) ? 1 : 0
1011   ne                         C = (A != B) ? 1 : 0
1101   gtz                        C = (A > 0) ? 1 : 0
1111   lez                        C = (A ≤ 0) ? 1 : 0
```

Determine the mapping of Opcode bits to ALU OP bits.

We will be implementing the Opcodes in white.

Notice that some of the opcodes will use the ALU, and some will not. Of the ones that will use the ALU, we need to map four of the bits to the ALU OP input so it will provide the correct output when the Opcode is found in our instruction.

Your job is to figure out a particular mapping that will work, using the bits we have. Notice the overlaps in the operations that have the SPECIAL opcode and those that don't: this will be the key to figuring out how to go about producing the correct translation.

For this lab, you only need to figure out a mapping for the functions that the ALU supports (in other words, don't worry about mapping the Branch, Jump, Move, Load, Store, or Set instructions.

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

Table 3: MIPS32 *SPECIAL* REGIMM Encoding of the rt Field

rt		*bits 18..16* →							
		0	1	2	3	4	5	6	7
↓ bits 20..19		000	001	010	011	100	101	110	111
0	00	BLTZ	BGEZ	BLTZL	BGEZL	*	*	*	*
1	01	TGEI	TGEIU	TLTI	TLTIU	TEQI	*	TNEI	*
2	10	BLTZAL	BGETAL	BLTZALL	BGETALL	*	*	*	*
3	11	*	*	*	*	*	*	*	*