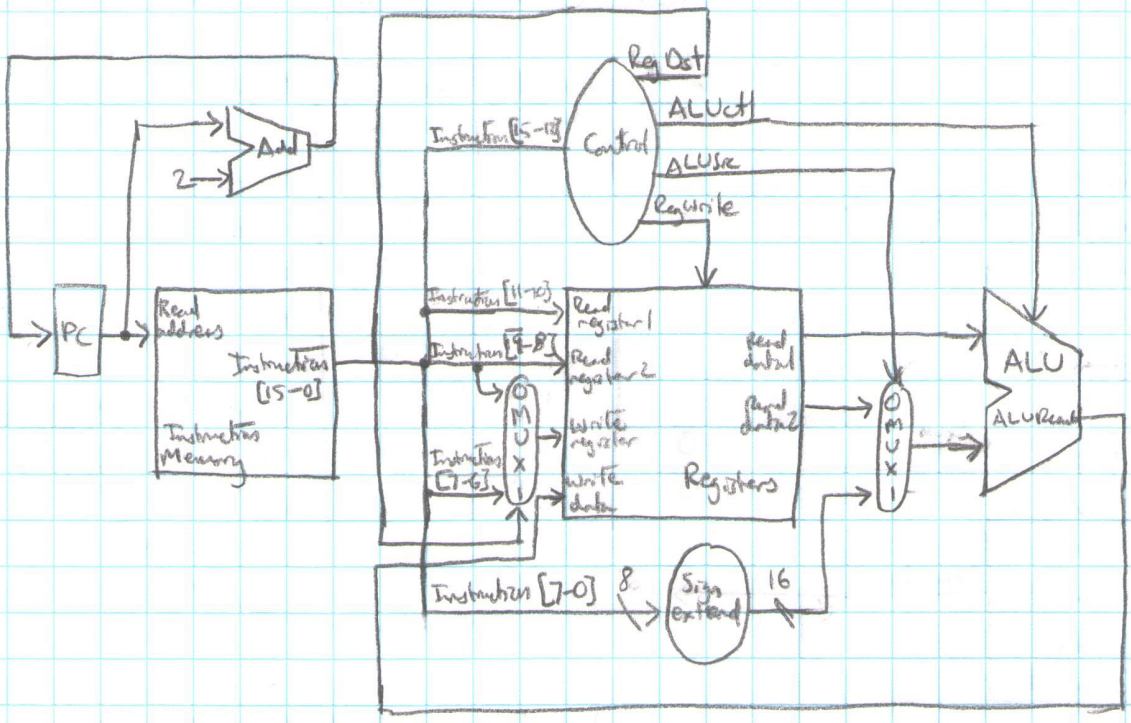
Rumman Shahzad
Shah Qureshi

Progress Report #1 - 2/29/2024

Overview: Simplified cycle datapath for R-type & addi instructions (16-bit MIPS)



Reg Array Implementation:
- PC
- Instruction memory

Behavioral:
- Main Control
- Register file
- Sign-Extend/Adder

Gate-Level:
- ALU
- Multiplexers

→ Logic diagrams only for ALU & multiplexers are shown, not for HA/FA.

Instruction Set Architecture/Control Unit Truth Table:

Instr	Opcode	RegDst	ALUSrc	RegWrite	ALUctl	
add	0000	1	0	1	0010	R-type
sub	0001	1	0	1	0110	
and	0010	1	0	1	0000	
or	0011	1	0	1	0001	
nor	0100	1	0	1	1100	
nand	0101	1	0	1	1101	
slt	0110	1	0	1	0111	
addi	0111	0	1	1	0010	→ I-type

input (Opcode) | output (RegDst, ALUSrc, RegWrite, ALUctl)

Instruction Formats:

R-format (add, sub, and, or, nor, nand, slt)

op	rs	rt	rd	unused
4	2	2	2	6

I-type (addi)

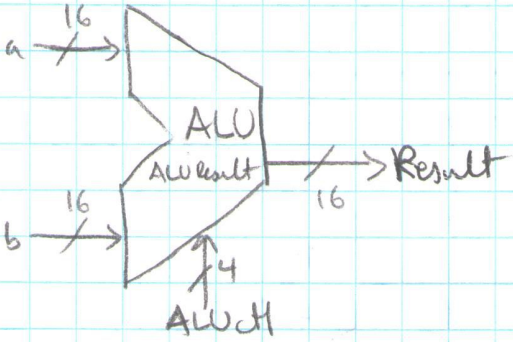
op	rs	rt	addr/value
4	2	2	8

Emmanuel Shahzad
Shah Qureshi

Progress Report #1 (cont'd) – 2/29/2024

No Zero or Overflow

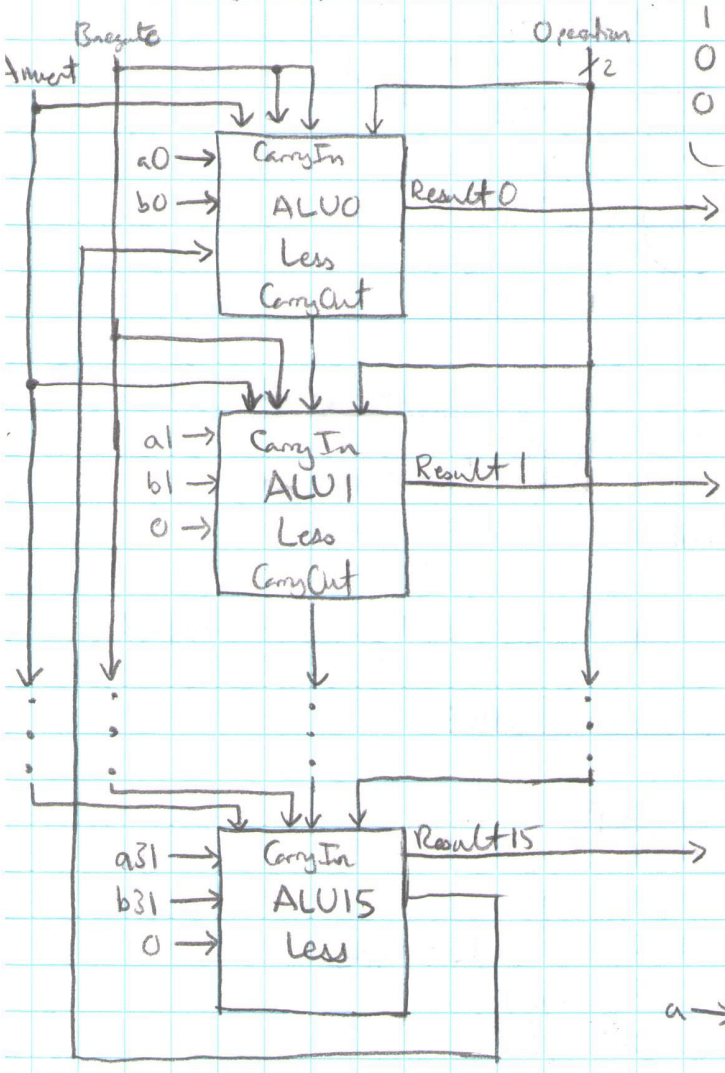
16-bit ALU:



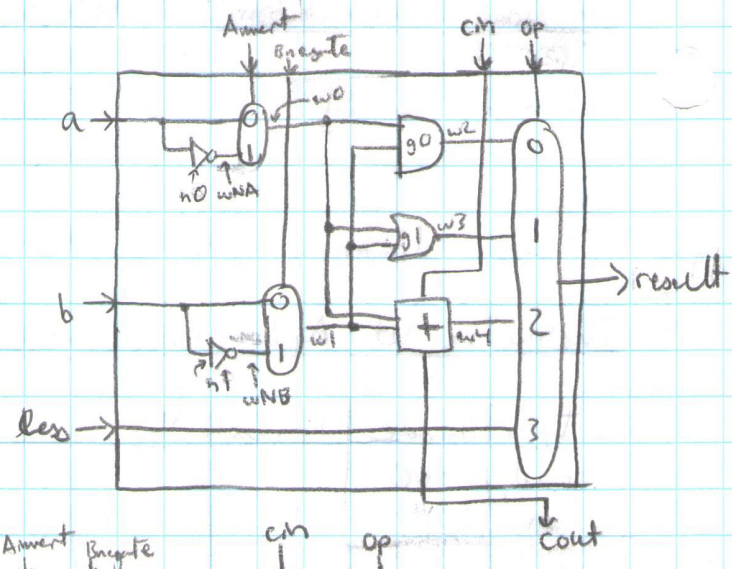
ALU Control Lines:

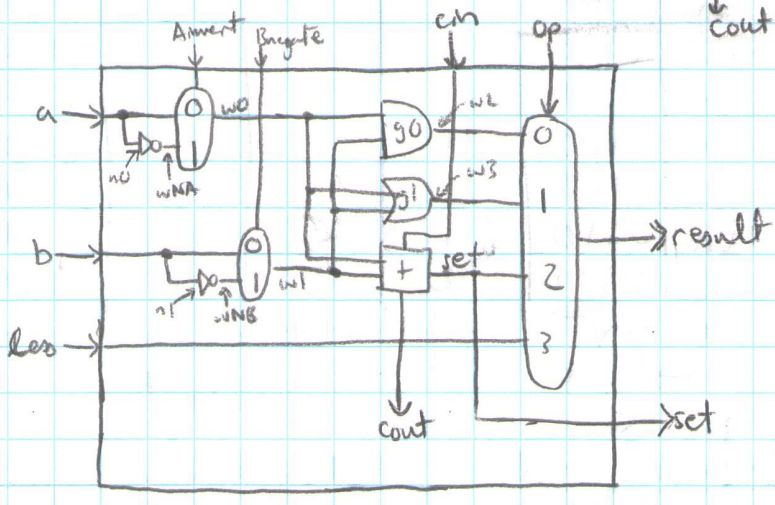
Ainvert ALUctl[3]	Bnegate ALUctl[2]	Operation ALUctl[1:0]	Instr.
0	0	10	add
0	1	10	sub
0	0	00	and
0	0	01	or
1	1	00	nor
1	1	01	nand
0	1	11	slt
0	0	10	addi

ALUctl



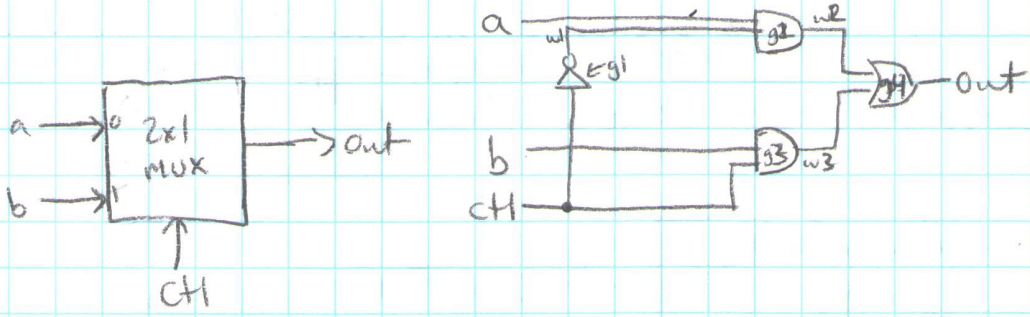
ALU0 to ALU14: 1-bit ALU, cascaded.



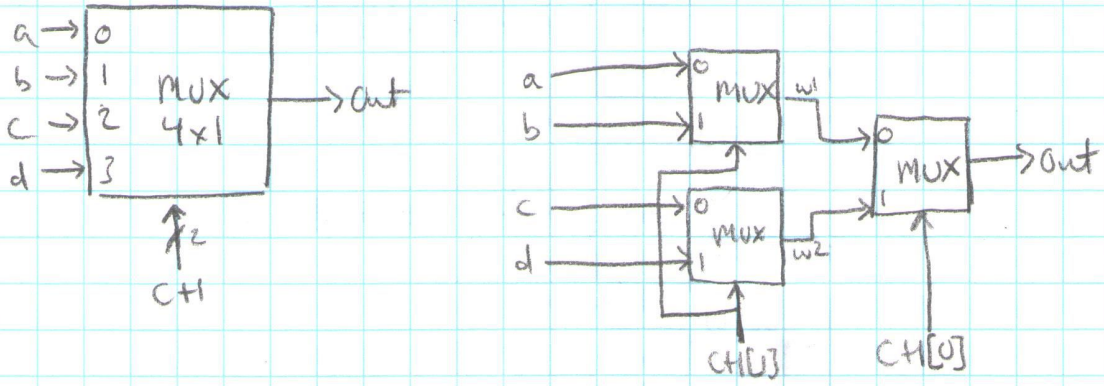


ALU 15: 1-bit ALU, last of the cascaded chain of ALU's. Slightly different

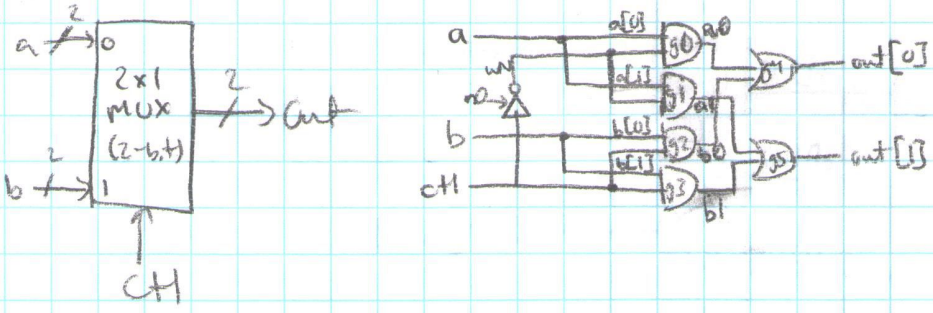
1-bit 2x1 Multiplexer:



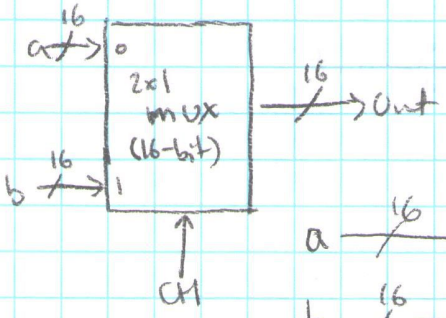
1-bit 4x1 Multiplexer: Implemented by cascading 2x1 multiplexers.



2-bit 2x1 Multiplexer: Multiplexes 2 inputs of 2-bit data



16-bit 2x1 multiplexer: Multiplexes 2 inputs of 16-bit data.



* Implemented by cascading 8 2x1 multiplexers for 2-bits, as shown below.*