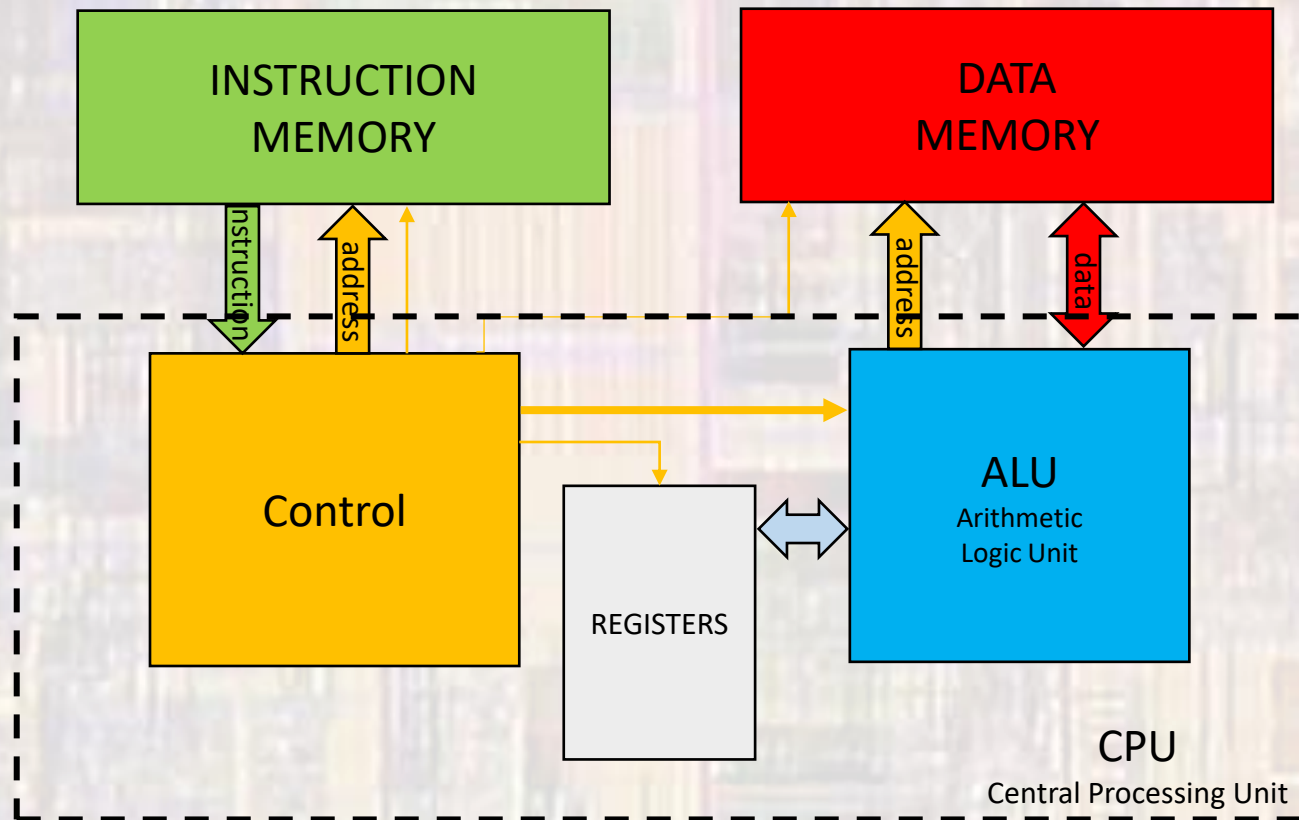


Single Cycle Processor ALU

Last updated 4/14/21

Single Cycle Processor - ALU

- Generalized Structure

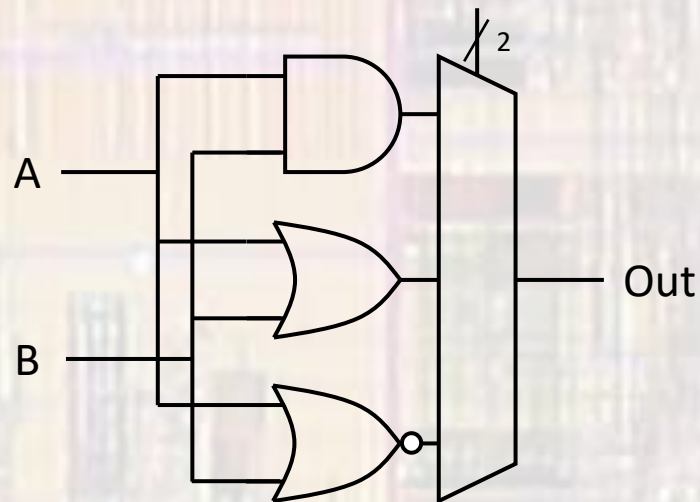


Single Cycle Processor - ALU

- Instruction Set
 - Instruction Set Architecture - ISA
 - Arithmetic Instructions
 - Add
 - Subtract
 - Less Than
 - Logical Instructions
 - AND
 - OR
 - NOR

Single Cycle Processor - ALU

- ALU - Implementation
 - Logical Instructions
 - AND, OR, NOR
 - 2 inputs A and B
 - 1 output



Single Cycle Processor - ALU

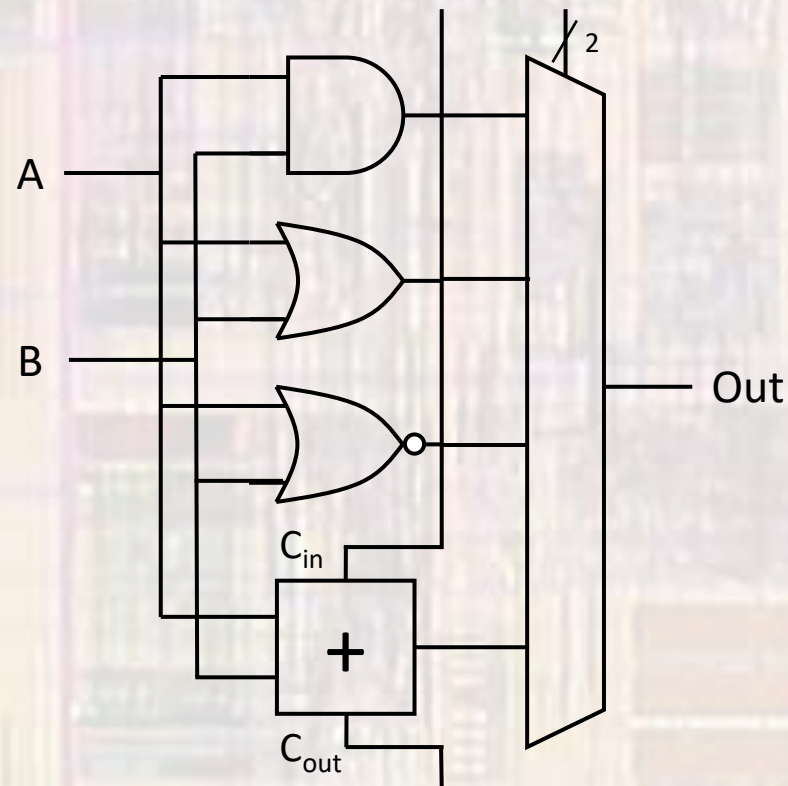
- ALU – Implementation

- Arithmetic Instructions

- ADD

- Inputs: A, B, C_{in}

- Outputs: Out, C_{out}



Single Cycle Processor - ALU

- ALU – Implementation

- Arithmetic Instructions

- SUB (2's complement)

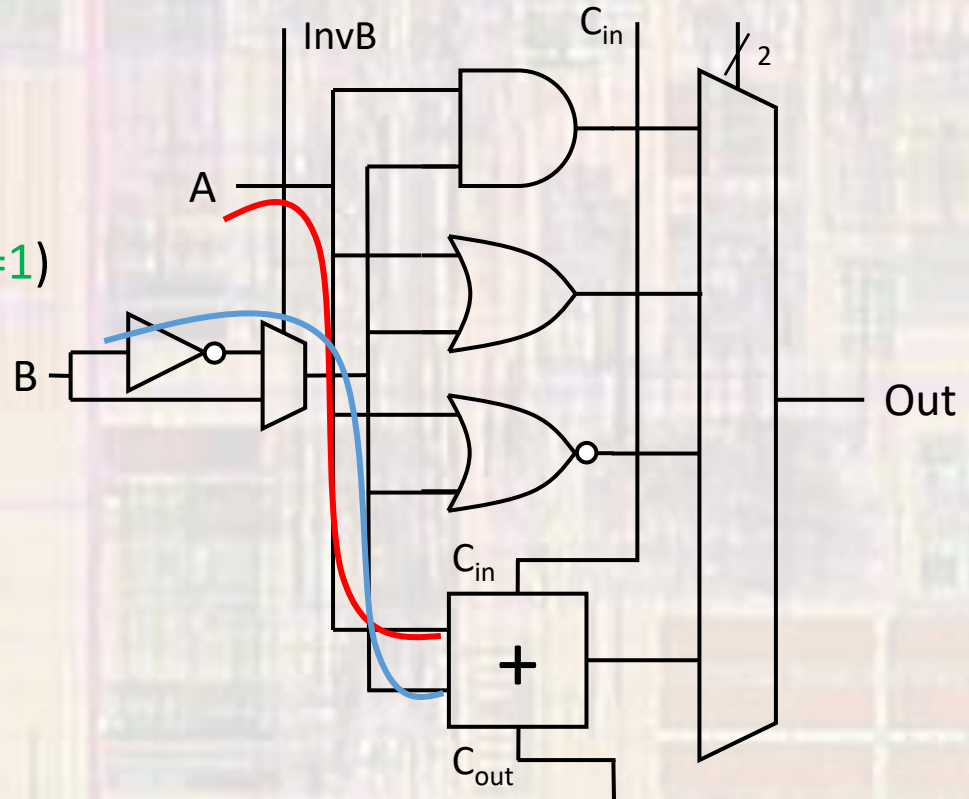
- $A - B = A + (-B)$

- $= A + (\overline{B} + 1)$

- Invert B and add 1 ($C_{inBO}=1$)

- Inputs: A, B, C_{in}

- Outputs: Out, C_{out}

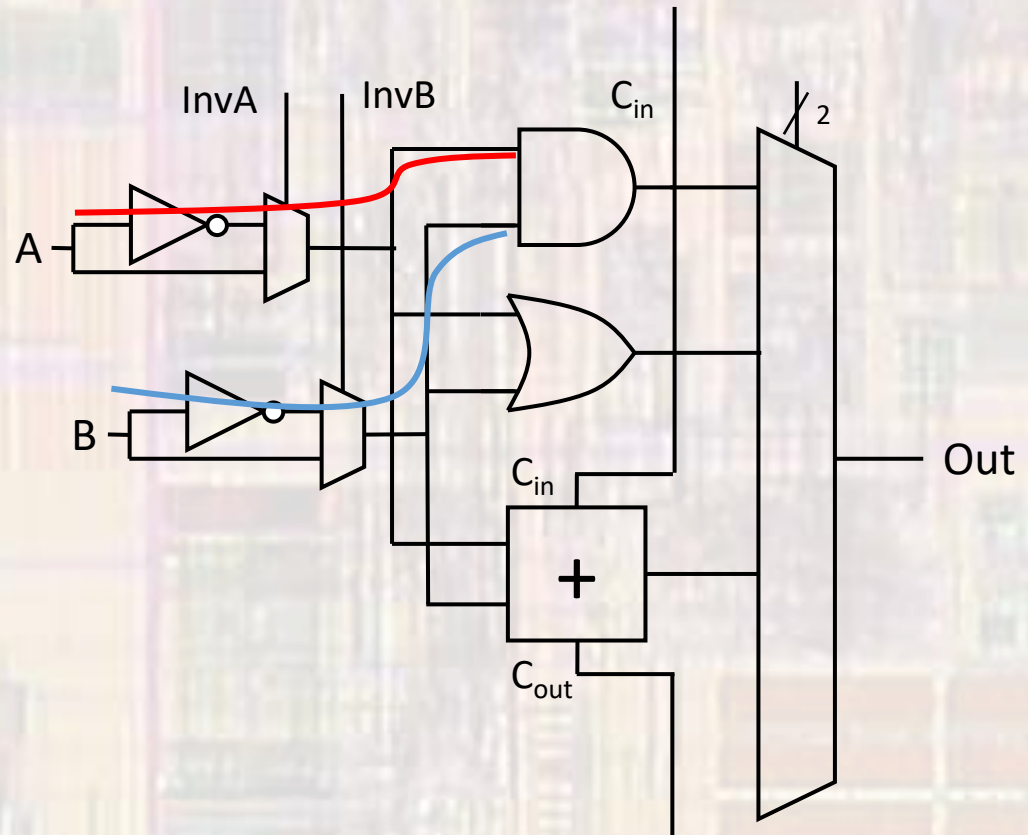


Single Cycle Processor - ALU

- ALU - Implementation

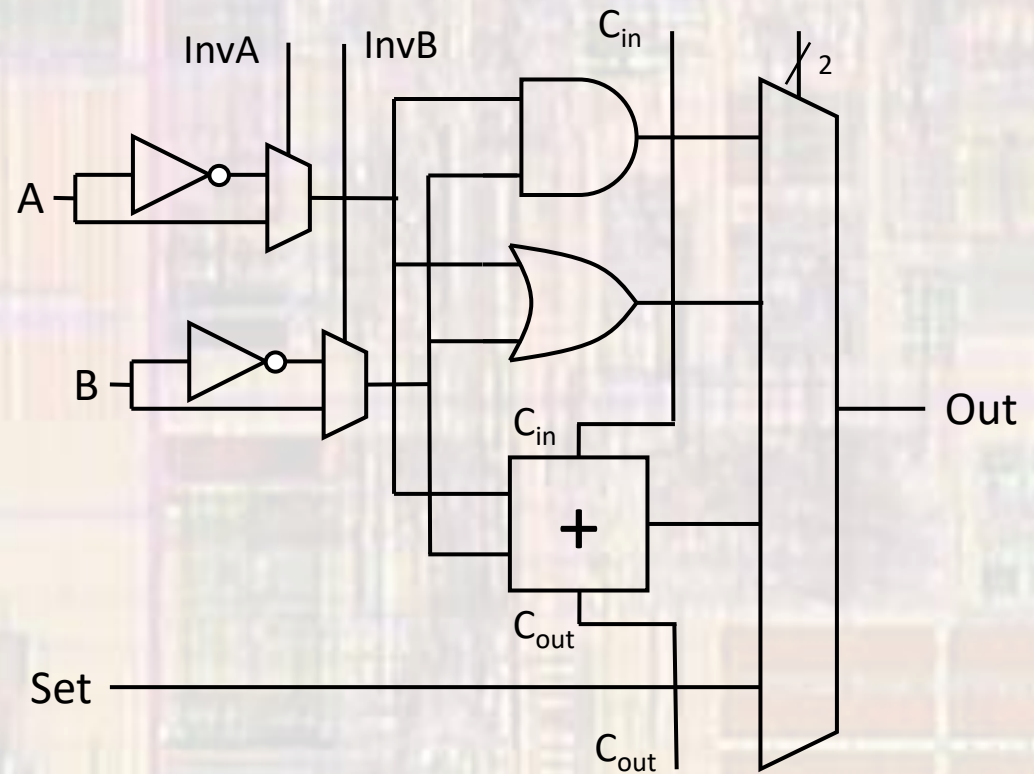
- Revisit NOR

- $\overline{A + B} = \bar{A} \bar{B}$



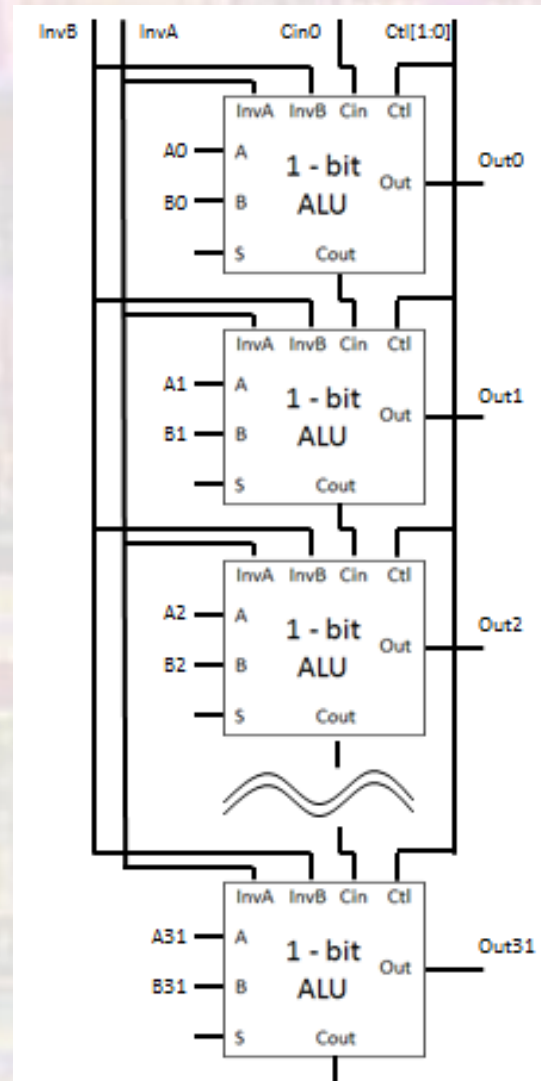
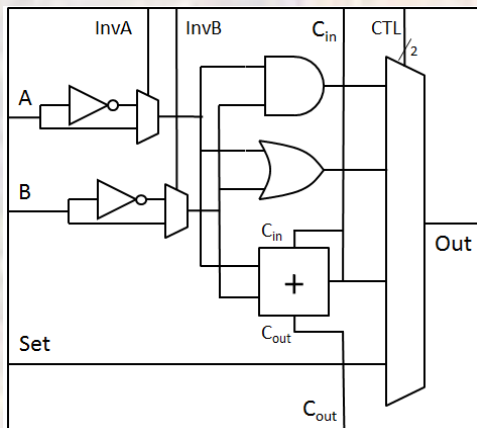
Single Cycle Processor - ALU

- ALU - Implementation
 - Pre-plan for set function



Single Cycle Processor - ALU

- ALU - Implementation
 - 32 bits



Single Cycle Processor - ALU

- ALU – Implementation
 - Determine if $A < B$
 - $A < B \rightarrow (A - B) < 0 \rightarrow$ negative answer
 - Subtraction is implemented by addition
 - $A - B \rightarrow A + \overline{B} + 1$
 - MSB indicates sign in 2's complement arithmetic
 - MSB = 1 \rightarrow negative number
 - MSB = 0 \rightarrow positive number
 - Set On Less Than instruction
 - Use **Adder MSB** for SLT signal
 - If $A < B$: Out[31:1] = 0, Out[0] = 1
 - If $A \geq B$: Out[31:0] = 0, Out[0] = 0
 - SET = 1

Single Cycle Processor - ALU

- ALU – Implementation

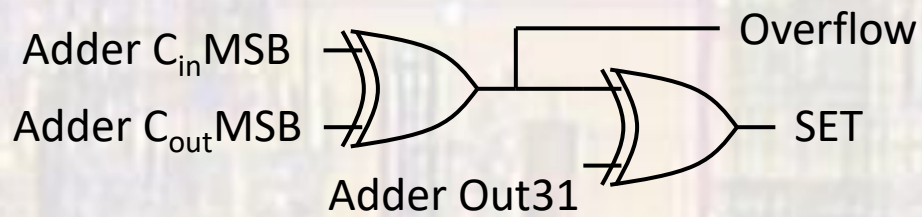
- Set On Less Than – cont'd

- MSB after subtraction indicates sign
 - MSB = 1 → negative number
 - MSB = 0 → positive number

Not implementing this function

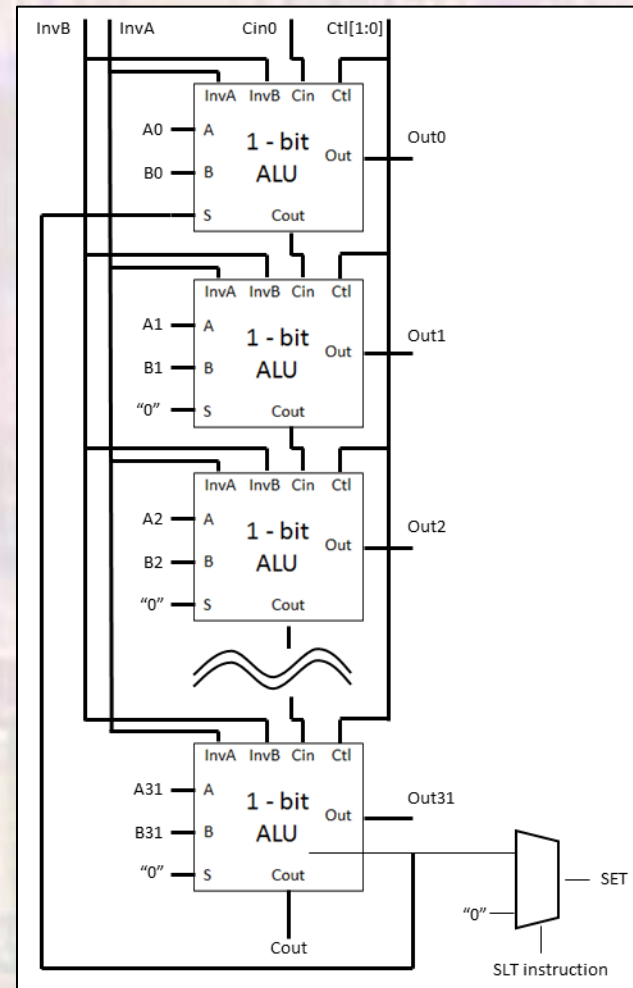
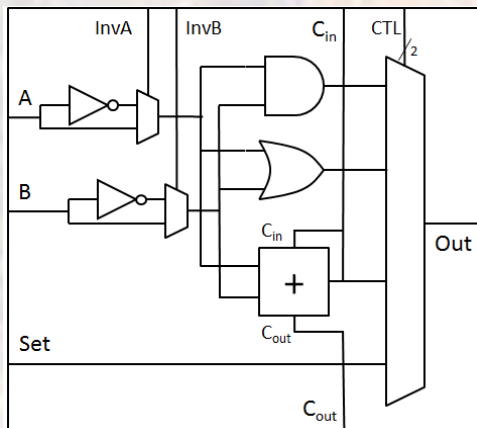
- Exception: Subtraction (addition) is not valid if overflow occurs
If overflow occurs, MSB is wrong sign

SET becomes MSB xor OVERFLOW



Single Cycle Processor - ALU

- ALU - Implementation

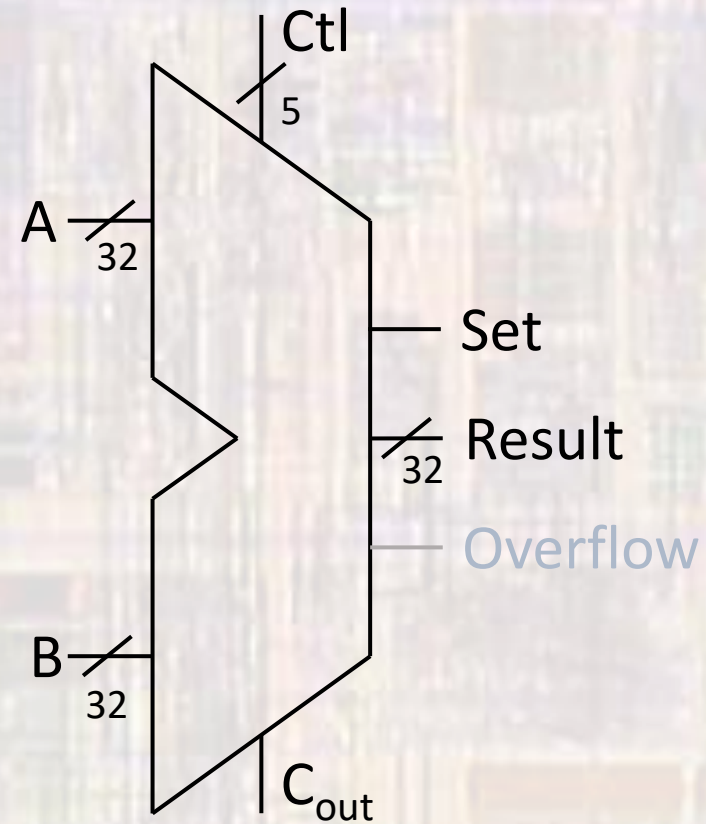
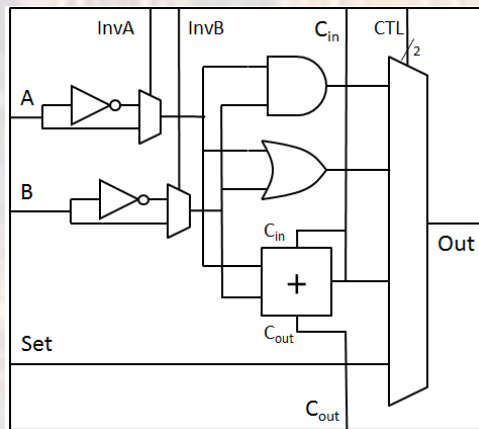


Single Cycle Processor - ALU

- ALU – Implementation

- Control

- invA
- invB
- Cin
- ctl[1:0]



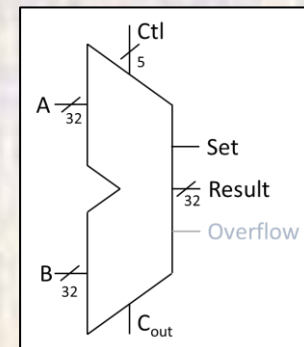
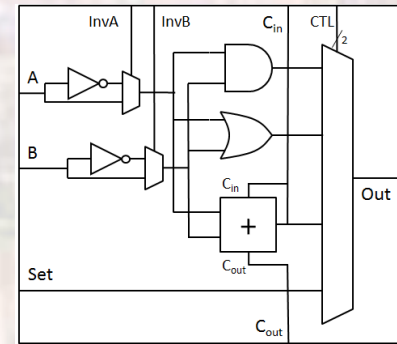
Single Cycle Processor - ALU

- ALU – Implementation

Operation	invA	invB	Cin	ctl[1]	ctl[0]
AND	0	0	x	1	1
OR	0	0	x	1	0
NOR	1	1	x	1	1
ADD	0	0	0	0	1
SUB	0	1	1	0	1
SLT	0	1	1	0	0

DeMorgan

Addition



What other logic function do we get for free

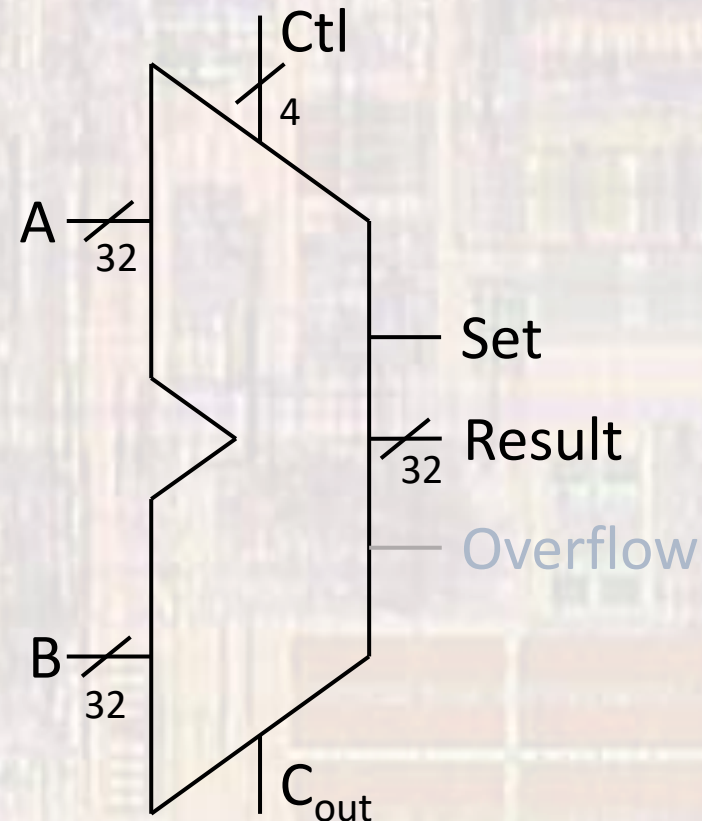
Single Cycle Processor - ALU

- ALU – Implementation

- Note: C_{in} and $invB$ can always be the same \rightarrow combine ($negB$)

Reduces control lines to 4

Operation	invA	negB	ctl[1]	ctl[0]
AND	0	0	1	1
OR	0	0	1	0
NOR	1	1	1	1
ADD	0	0	0	1
SUB	0	1	0	1
SLT	0	1	0	0



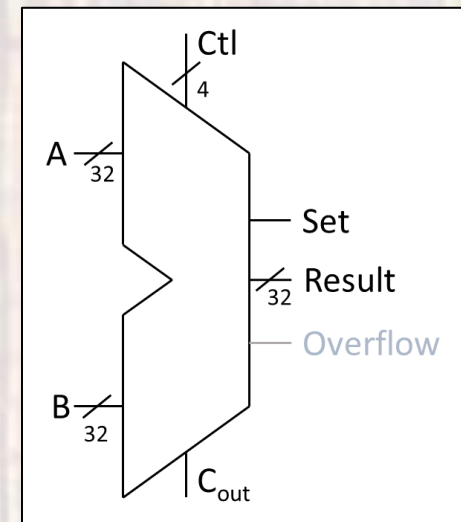
Single Cycle Processor - ALU

- ALU – Implementation
 - 3 out of 6 instructions involve addition

Current implementation is **very slow** – why?

Operation	invA	negB	ctl[1]	ctl[0]
AND	0	0	1	1
OR	0	0	1	0
NOR	1	1	1	1
ADD	0	0	0	1
SUB	0	1	0	1
SLT	0	1	0	0

Addition {



Single Cycle Processor - ALU

- Enhanced Adder

