Register Transfer Notation (RTN)

- language used to describe hardware computation
- describes a flow of data from register (Q) to register (D)
  - from Q outputs of registers, or from constants
  - to D inputs of registers

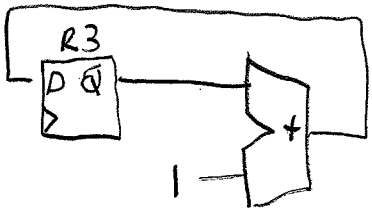
Purpose: to determine the D values captured on next ↑ clk

Examples

$$\underbrace{R0}_{\text{destination register}} \leftarrow \underbrace{\$00}_{\text{source values \& operators}}$$

assignment

$$R3 \leftarrow R3 + 1$$



$$R2 \leftarrow \underbrace{R2 + R3}_{\text{can get complex}}$$

Complex operations must be broken down into several steps or micro-operations (μ-ops)

- use RTN to describe μ-ops

complex operation

- each μ-op "fits" onto your datapath circuit, taking 1 clock cycle

Implementing (mapping) RTN onto a Specific Datapath Circuit

Rules:
1. data flows from Q to D
2. each wire takes on only 1 value in a clock cycle
3. each (simple) μ-op takes only 1 cycle

Examples

1. R0 ← $00

2. R3 ← $01

3. R3 ← R3 + $01 } desired operation

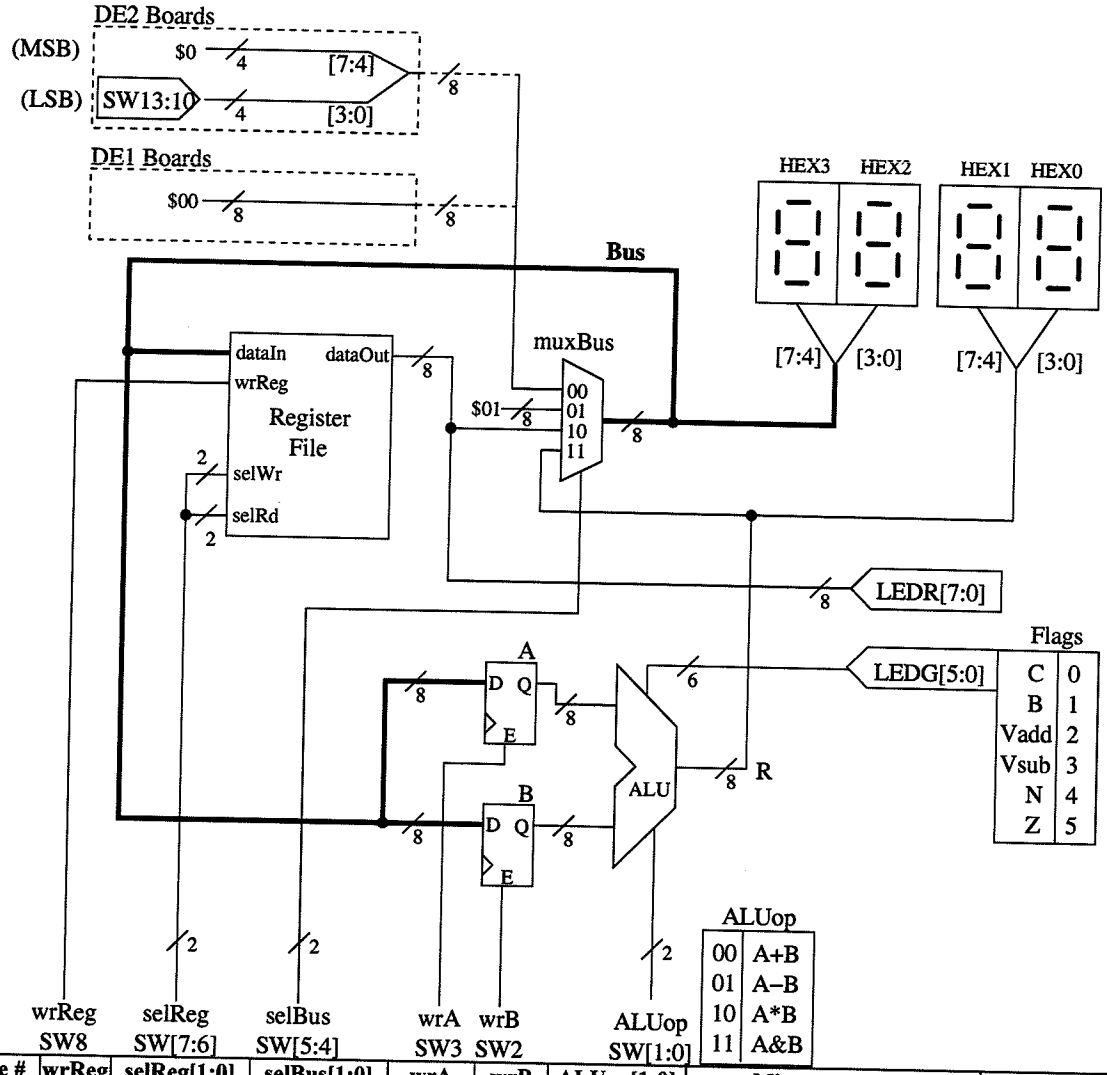
   i) A ← $01
   ii) B ← R3
   iii) R3 ← A+B } required μ-ops

   to find these, start with the original operation "R3 ←" and see that it requires a complex value on the Bus "R3+ $01" which must come from the ALU. then, you notice the ALU can only operate on A and B, so these registers must first take on their needed values (constant $01 and R3)

4. R2 ← R2 + R2

   i) A ← R2 ; B ← R2
   ii) R2 ← A+B } μ-ops

   we can do 2 at once!



Cycle #	wrReg	selReg[1:0]	selBus[1:0]	wrA	wrB	ALUop[1:0]	Micro-operation	Overall Operation
1.	1	00	00	0	0	XX	R0 ← $00	R0 ← $00
2.								
3.	1	11	01	0	0	XX	R3 ← $01	R3 ← $01
4.								
5.	0	XX	01	1	0	XX	A ← $01	R3 ← R3 + 1
6.	0	11	10	0	1	XX	B ← R3	
7.	1	11	11	0	0	00	R3 ← A+B	
8.								
9.								R2 ← R2 + R2
10.								
11.								
12.								
13.								
14.								
15.								
16.								
17.								
18.								
19.								
20.								