COMS12200
Introduction to Computer Architecture

Dr. Cian O'Donnell

cian.odonnell@bristol.ac.uk

Topic 5: State machines and decoding

Topics

1. Data, Control and Instructions
2. Memory
3. Execution cycle
4. Processor control flow
5. State machines and decoding
6. Machine types
7. Memory paradigms

4. Processor control flow

Summary from previous lecture

- We looked at branching
- And its use in:
  - `for` loops
  - `if/else` statements
  - calling sub-routines (functions)
- How they are implemented in hardware using the link register and the stack.

Topics

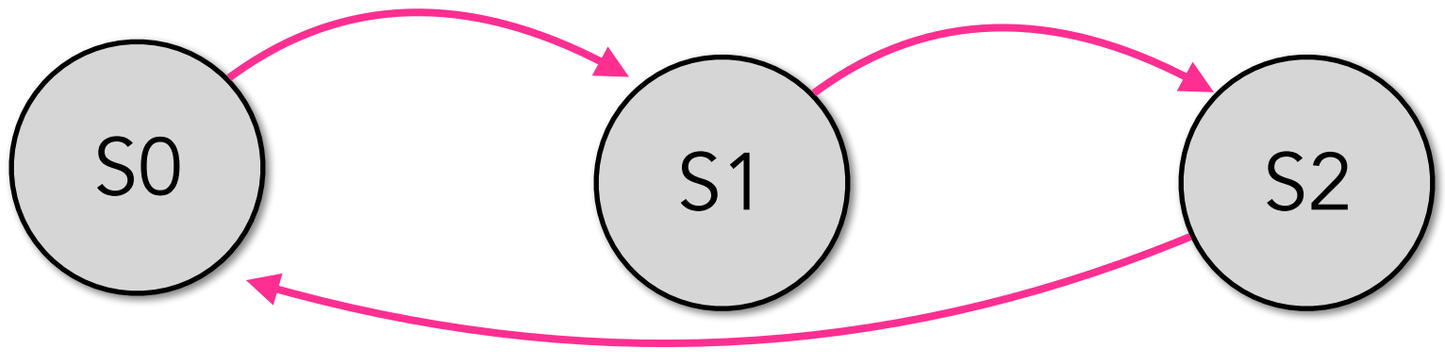
1. Data, Control and Instructions
2. Memory
3. Execution cycle
4. Processor control flow
5. State machines and decoding
6. Machine types
7. Memory paradigms

State machine recap

- A state machine is one with a finite set of defined states.

- Transitions are made between states.

- Transitions can be gated or ungated.



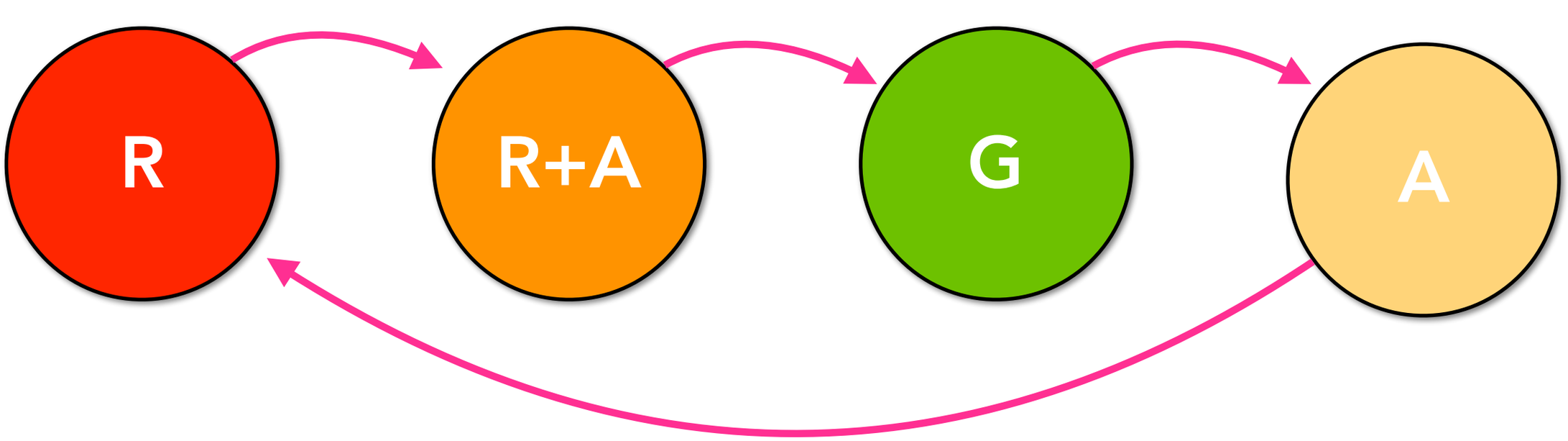
State machine recap

Each state is uniquely labeled and transitions can also be expressed in a table.

Current state	Next state	Condition
`S0`	`S1`	Always
`S1`	`S2`	Always
`S2`	`S0`	Always

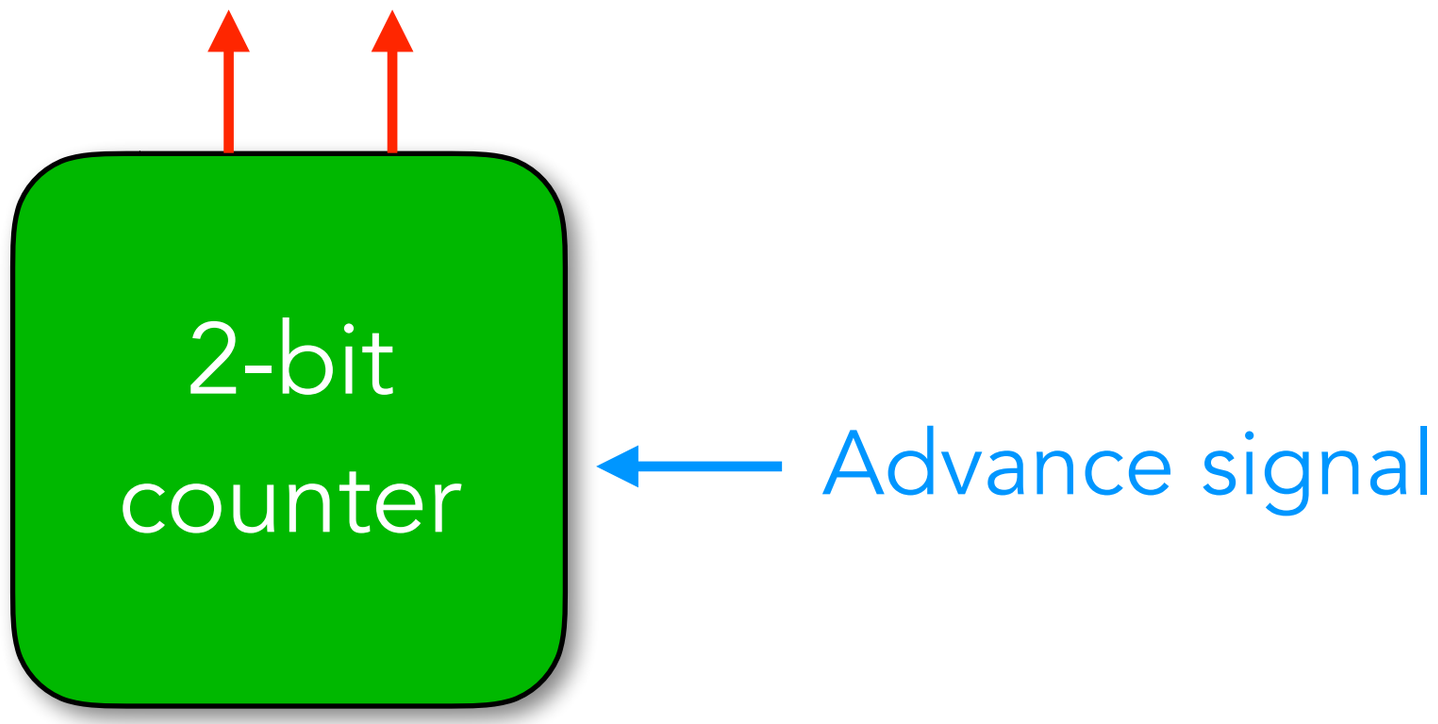
Traffic lights

A traffic light is a useful state machine with four states.



The simplest 4-state machine

2 bits = 2^2 = 4 states

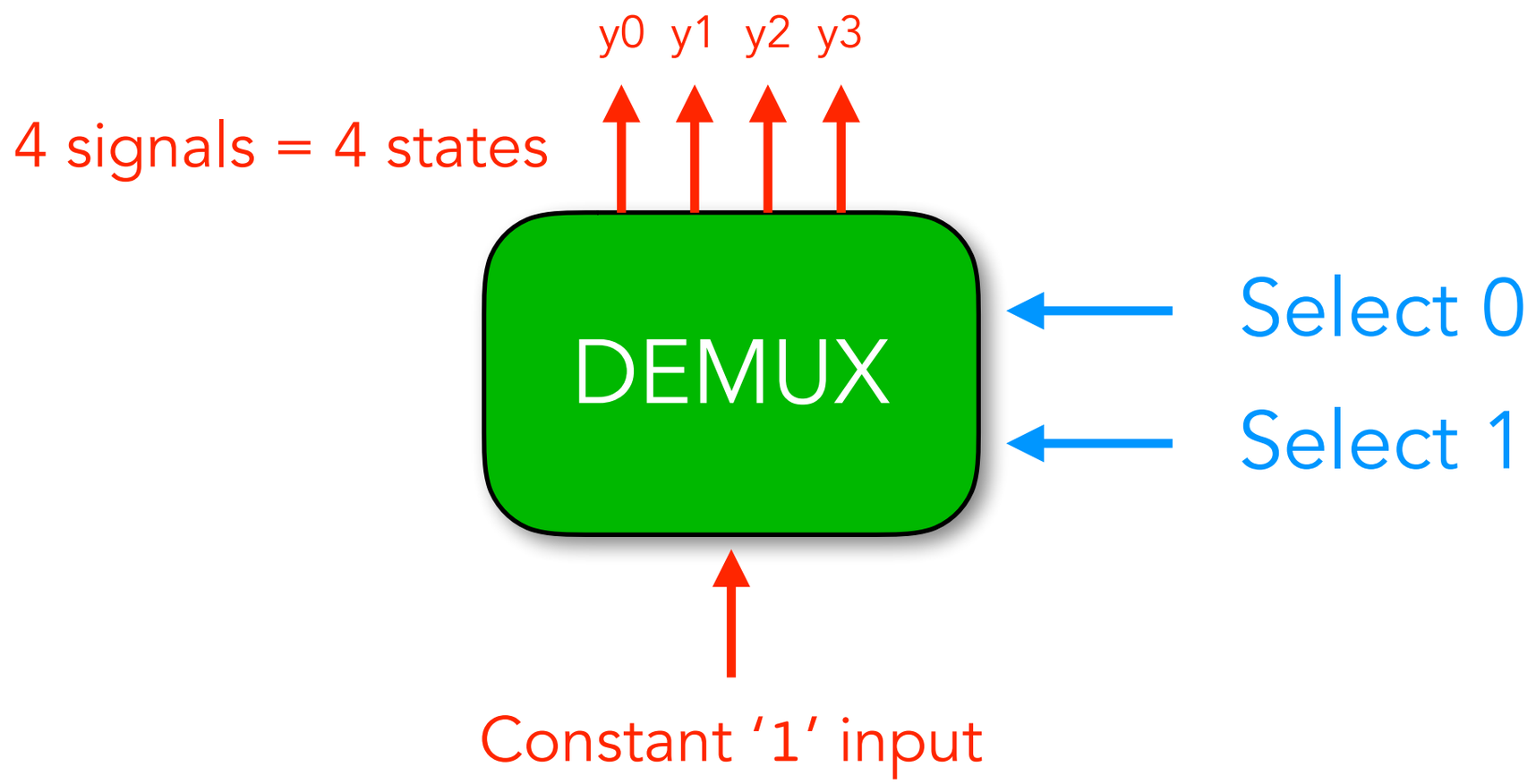


Decoding states

A four state decoder

Q: How can we uniquely decode a state from our simple machine?

A: A demultiplexor solves the problem.



From states to lights

How do we convert the decoder outputs to the R, A, G values for a simple traffic light?

From states to lights

- Observe that each of the decoder outputs has produced a minterm.

- Each minterm is unique and can be manipulated via boolean logic.

- Typically, we will combine minterms via logical gates to obtain a compound output (e.g. Red light from R and R/A minterms).

Variable control signals

Why vary control?

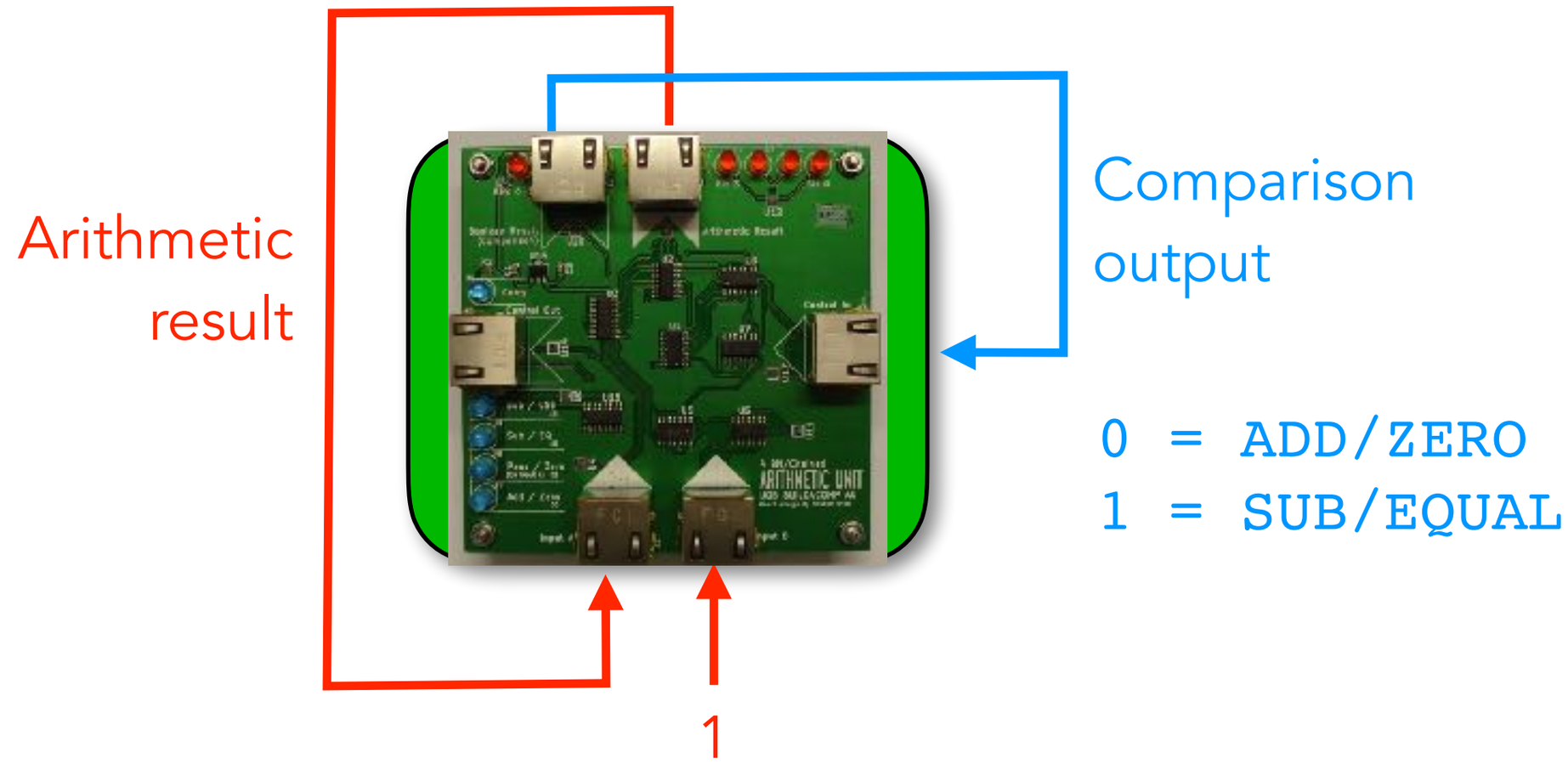
- In your labs so far, you have made static systems: ones that do the same task again and again (and again)

- To change their behaviour, you have used switches.

- This is great for simple tasks, but we'd really like to be hands off when computing more complex things.

Supporting variable control

Here's a very simple example of how we could vary control to do something useful:




```
0 = ADD/ZERO
1 = SUB/EQUAL
```

Feedback

- Feedback was the key thing that made the previous example do something interesting.

- Feedback is when a previous output from a system is used to alter its current behaviour:

$$S_{n+1} = f(S_n, \text{inputs})$$

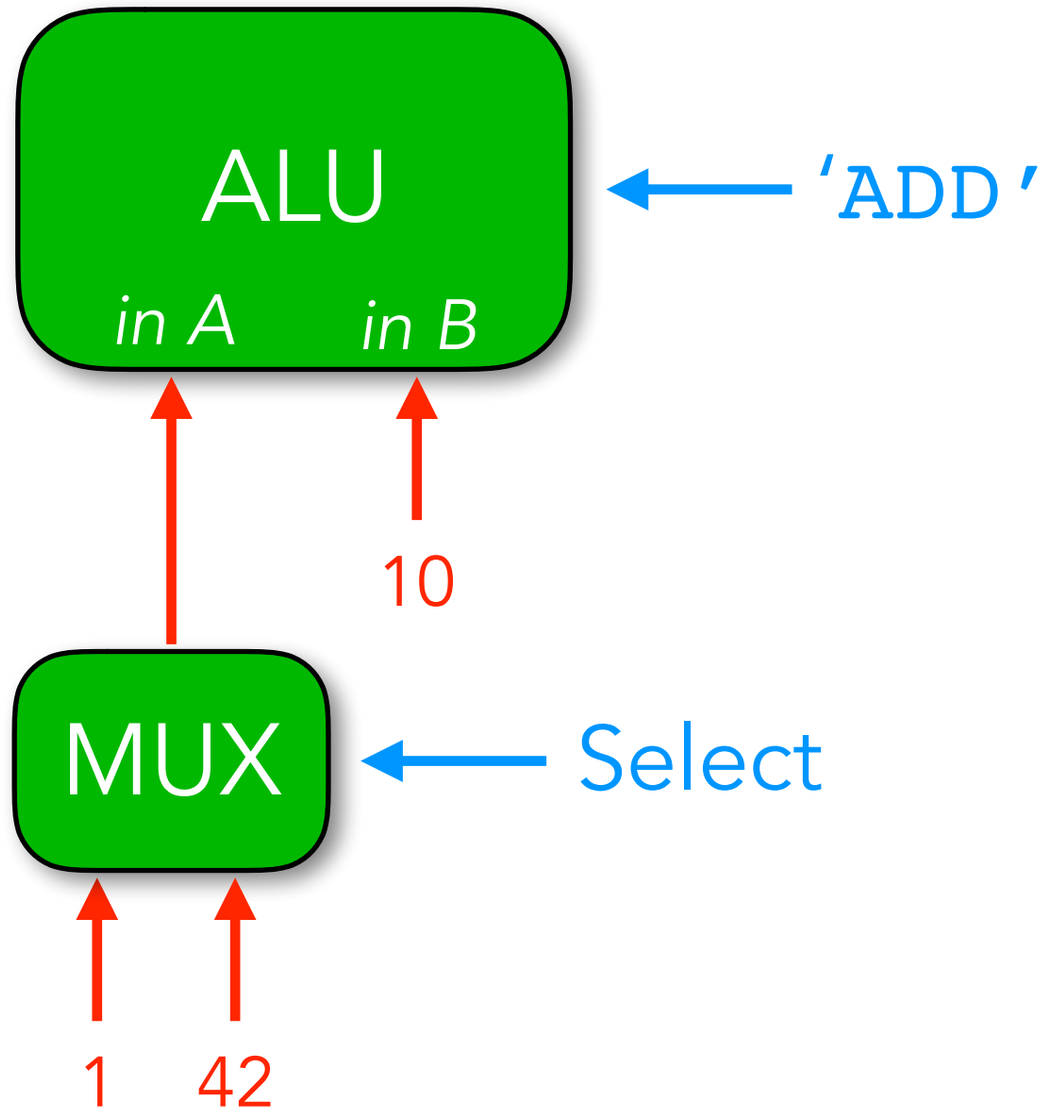
Feedback to change operations

- We can use feedback to alter both the data inputs and/or the function of a unit.

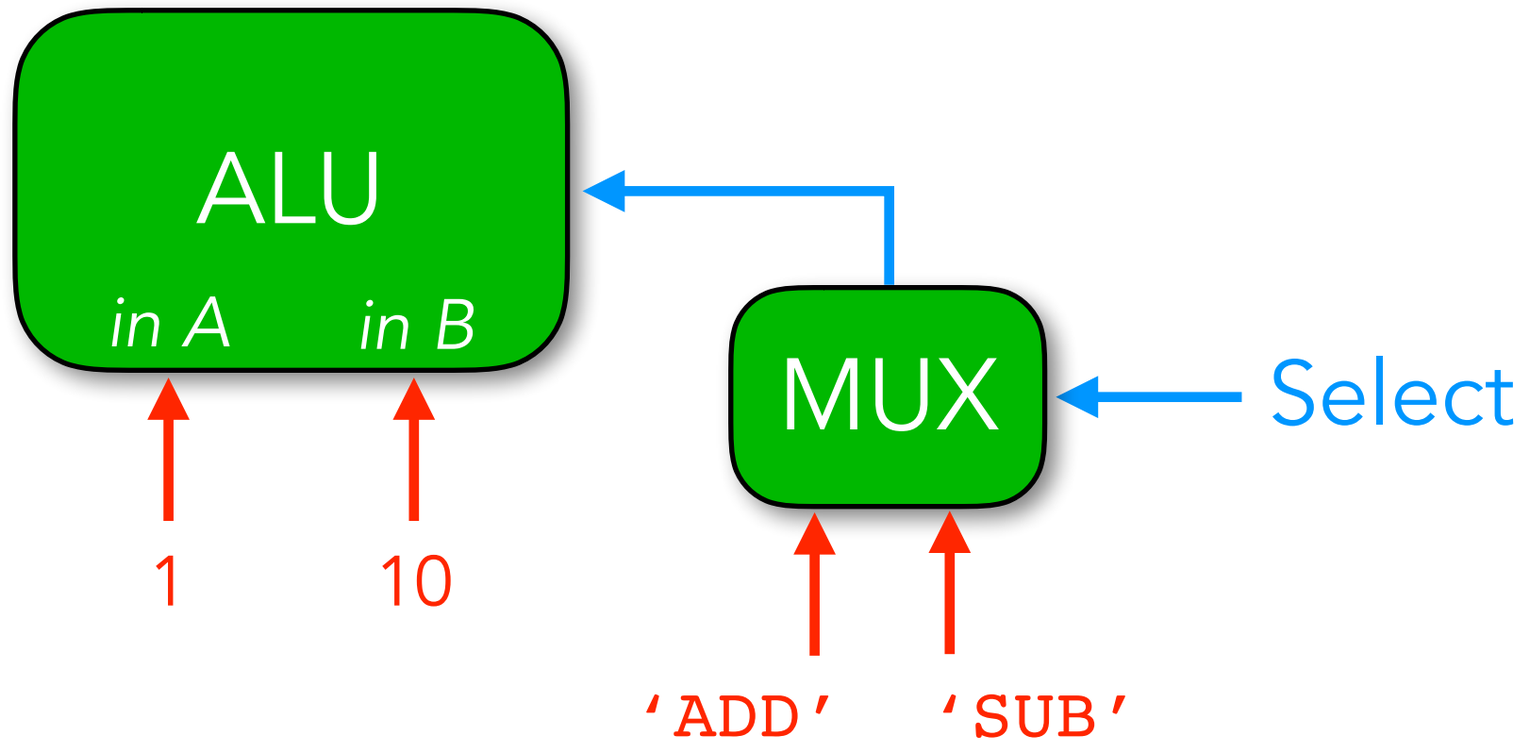
- There are several useful implementation methodologies.

- Now we'll look at one simple one

Changing the data inputs



Changing the control inputs



Selecting the input

- Where do the select signals come from?

  - They could come from instructions
    - Instructions will dictate behaviour
- They could come from feedback
- Data values will dictate behaviour

Notes

Here are three useful guidelines when creating control flow for state machines:

1. MUXs are useful for selecting inputs
2. DEMUXs are useful for decoding outputs
3. OR gates are useful for combining states and producing feedback signals.

Week 11 Labs

This week you will:

- Build a series of different traffic lights circuits using ideas from today's lecture (state machines and decoding).