27.07.20 12:40

CSCI 150
Introduction to Digital and Computer System Design
Lecture 5: Registers II

Jetic Gū

2020 Summer Semester (S2)

Overview

- Focus: Fundamentals of Complex Digital Circuit Design
- Architecture: von Neumann
- Textbook v4: Ch7 7.6; v5: Ch6 6.6
- Core Ideas:
    1. Implementation of Register Microoperations

P0 Review

Register Transfer Operations

	Operator	Example
Assignment	<=	ax **<=** 12h
Reg. Transfer	<=	ax **<=** bx
Addition	+	ax **+** bx
Subtraction	-	ax **-** bx
Shift Left	sll	ax **sll** 2
Shift Right	srl	ax **srl** 2

	Operator	Example
Bitwise AND	and	ax **and** bx
Bitwise OR	or	ax **or** bx
Bitwise NOT	not	**not** ax
Bitwise XOR	xor	ax **xor** bx
Vectors	ax(3 down to 0)	ax(3 down to 0)
Concatenate	&	ax(7 down to 4) **&**ax(3 down to 0)

Review

Implementation of Datapath I

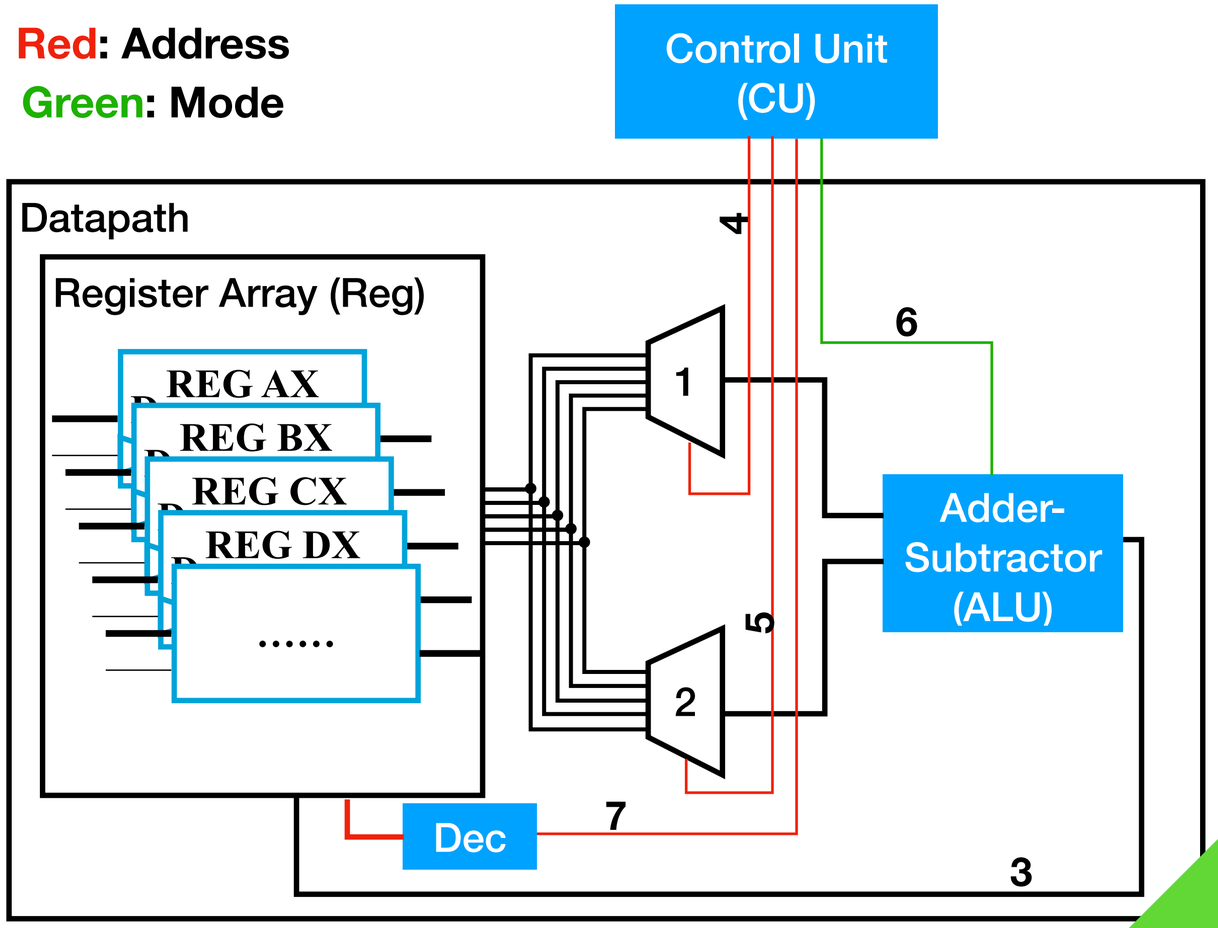
Architecture

Summary

P1
Datapath

Datapath and Control Unit

1. `ALU` connected to `Reg` for 1st input
2. `ALU` connected to `Reg` for 2nd input
3. `ALU` connected to `Reg` to store result
4. `CU` tells `ALU` which register to take as 1st input
5. `CU` tells `ALU` which register to take as 2st input
6. `CU` tells `ALU` which operation to do
7. `CU` tells which `Reg` to store result in
   Using decoder and `EN` on each Register



Review

Single Register Microoperations

- In reality, a single circuit is designed to do multiple stuff
  - a CPU can perform thousands/millions of different types of instructions
- How to design it?
- Let’s start with single register Microoperations
  - Input: one of the registers; Output: another one of the registers;

Concept

Single Register Microoperations

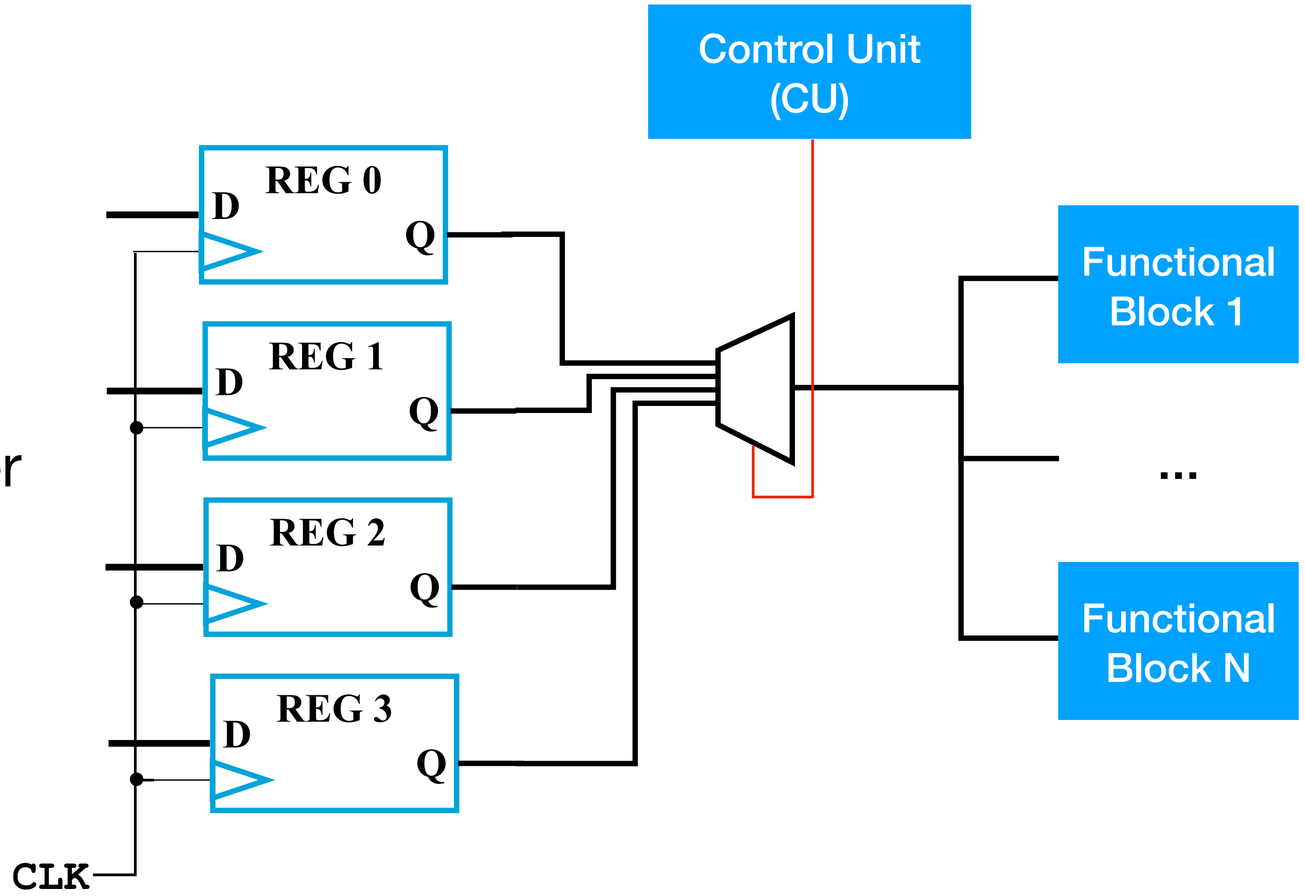
	Operator	Example
Assignment	<=	ax <= 12h
Reg. Transfer	<=	ax <= bx
Addition	+	ax + bx
Subtraction	-	ax - bx
Shift Left	sll	ax sll 2
Shift Right	srl	ax srl 2

	Operator	Example
Bitwise AND	and	ax and bx
Bitwise OR	or	ax or bx
Bitwise NOT	not	not ax
Bitwise XOR	xor	ax xor bx
Vectors	ax(3 down to 0)	ax(3 down to 0)
Concatenate	&	ax(7 down to 4) &ax(3 down to 0)

Concept

Select Registers for Input

- Suppose we have 4 registers, we want to select one to provide input to a functional block
- Suppose 2^n registers, each 8-bit
- What kind of multiplexer should be used?
- How many bits on the red wire?



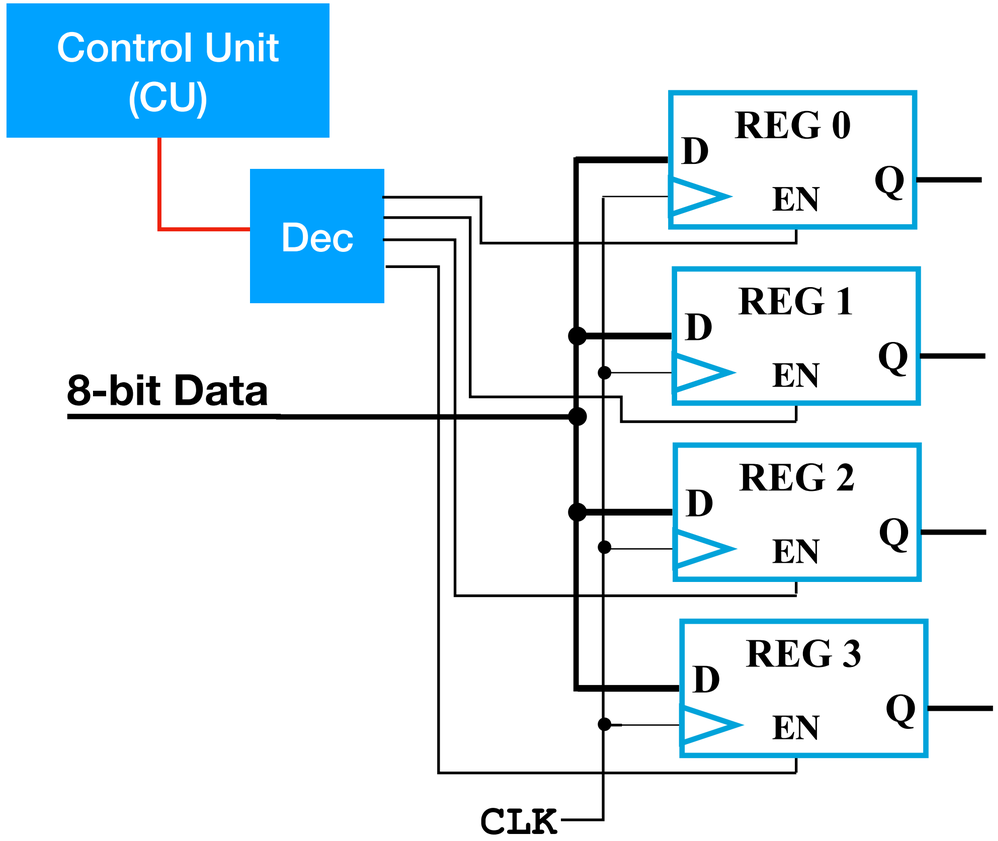
Concept

P1
Datapath

Select Registers to Output

- Suppose we have 4 registers, we want to select one to store our output

- Suppose 2^n registers, each 8-bit
- What kind of decoder should be used?
- How many bits on the red wire?



Concept

P1
Datapath

Single Register Microoperations

- Assignment
  e.g. `ax <= 12h`
- Register Transfer
  e.g. `ax <= bx`
- Shift left by $X < 8$
  e.g. `ax sll` X
- Shift right by $X < 8$
  e.g. `ax srl` X
- Bitwise NOT
  e.g. `not ax`
- Vector by
  $0 \leq X \leq Y < 8$
  e.g. `ax(`Y `down to` X`)`
- Rotate left by $X < 8$
  e.g. `ax rll` X
- Rotate right by $X < 8$
  e.g. `ax rrl` X

Concept

P1
Datapath

Single Register Microoperations

0. Assignment
   e.g. `ax <= 12h`

1. Register Transfer
   e.g. `ax <= bx`

2. Shift left by $X < 8$
   e.g. `ax sll` X

2. Shift right by $X < 8$
   e.g. `ax srl` X

3. Bitwise NOT
   e.g. `not ax`

4. Vector by
   $0 \leq X \leq Y < 8$
   e.g. `ax(`Y `down to` X`)`

5. Rotate left by $X < 8$
   e.g. `ax rll` X

5. Rotate right by $X < 8$
   e.g. `ax rrl` X

Functional Block 1

...

Functional Block N

Example

P1
Datapath

Single Register Microoperations

0. Assignment
   e.g. `ax <= 12h`

1. Register Transfer
   e.g. `ax <= bx`

2. Shift left by $X < 8$
   e.g. `ax sll` X

2. Shift right by $X < 8$
   e.g. `ax srl` X

3. Bitwise NOT
   e.g. `not ax`

4. Vector by
   $0 \leq X \leq Y < 8$
   e.g. `ax(`Y `down to` X`)`

5. Rotate left by $X < 8$
   e.g. `ax rll` X

5. Rotate right by $X < 8$
   e.g. `ax rrl` X

0. Assign.

1. Transfer

2. Shift L/R

3. NOT

4. Vector

5. Rotate L/R

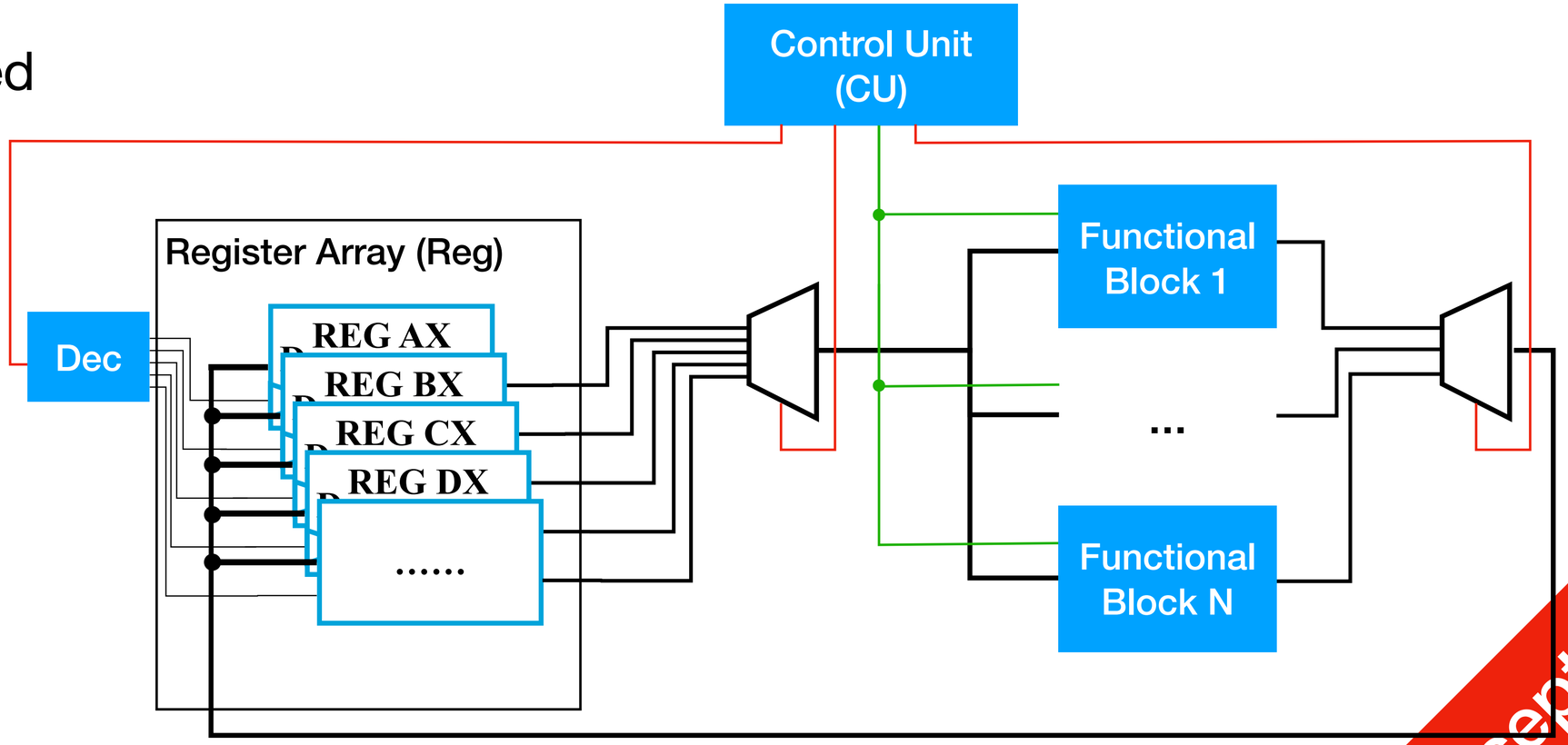
Example

P1
Datapath

Single Register Microoperations

Red: Address
Green: Mode

- Select Input `REG` to be fed into Processing Blocks
- Select mode for Processing Blocks
- Select Processing Block for output to `REG`
- Select Receiving `REG`



Concept

Multiple Register Microoperations

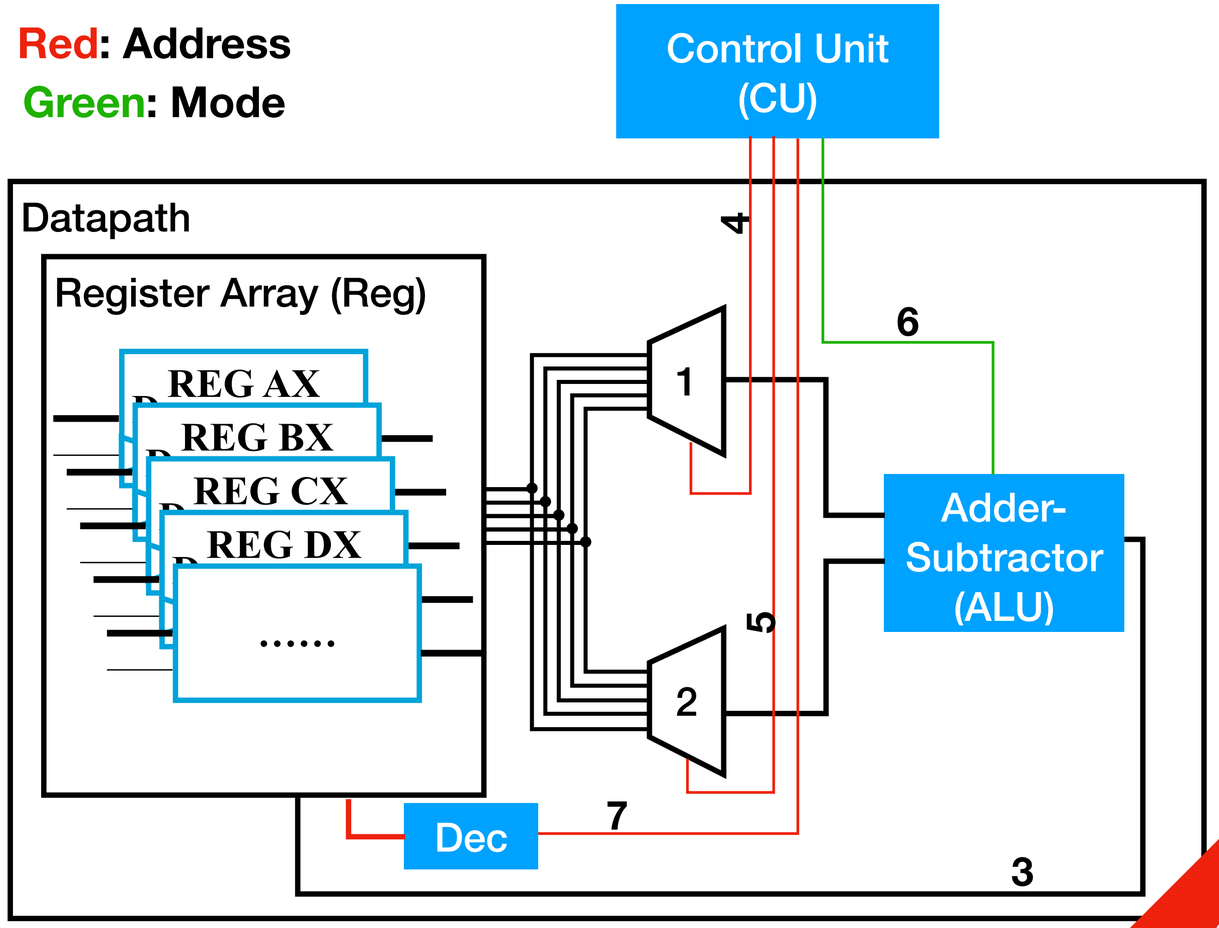
	Operator	Example
Assignment	<=	ax **<=** 12h
Reg. Transfer	<=	ax **<=** bx
Addition	+	ax **+** bx
Subtraction	-	ax **-** bx
Shift Left	sll	ax **sll** 2
Shift Right	srl	ax **srl** 2

	Operator	Example
Bitwise AND	and	ax **and** bx
Bitwise OR	or	ax **or** bx
Bitwise NOT	not	**not** ax
Bitwise XOR	xor	ax **xor** bx
Vectors	ax(3 down to 0)	ax(3 down to 0)
Concatenate	&	ax(7 down to 4) **&**ax(3 down to 0)

Concept

Multiple Register Microoperations

- Extension: how to incorporate more multiple register functional blocks?
- Extension: how to incorporate more single register functional blocks?



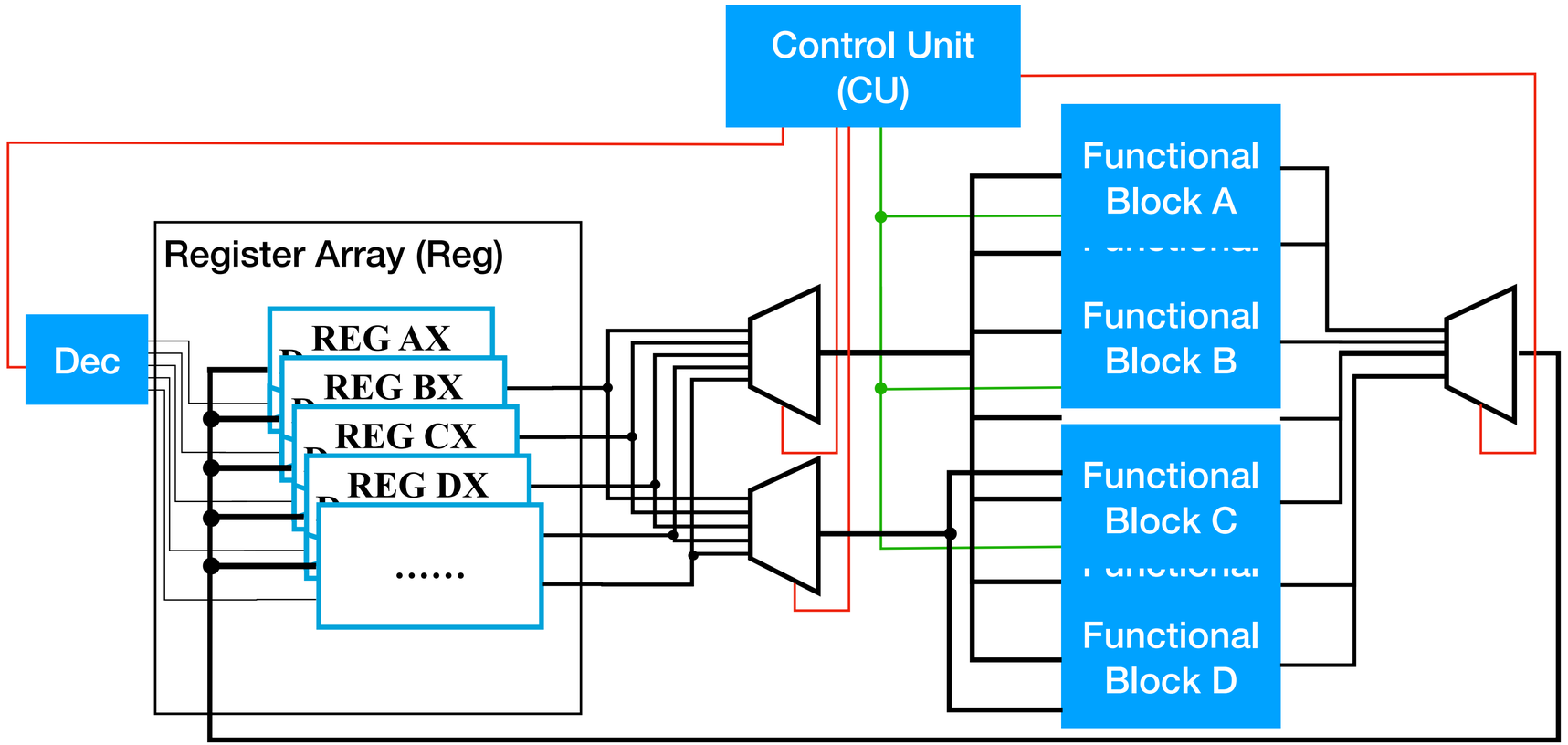
Concept

Example Datapath Question 1

- The control unit selects between 4 operations A, B, C, D.
- A, B, C, D each takes 1/1/2/2 register(s) as input and outputs to 1 register
- A, B, C, D doesn’t have additional mode selections
- Specify the control unit interface to the datapath
- Draw the circuit diagram of such datapath with a register array

Exercise

Example Datapath Question 1



Exercise