30.03.20 08:47

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

Jetic Gū

2020 Winter Semester (S1)

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

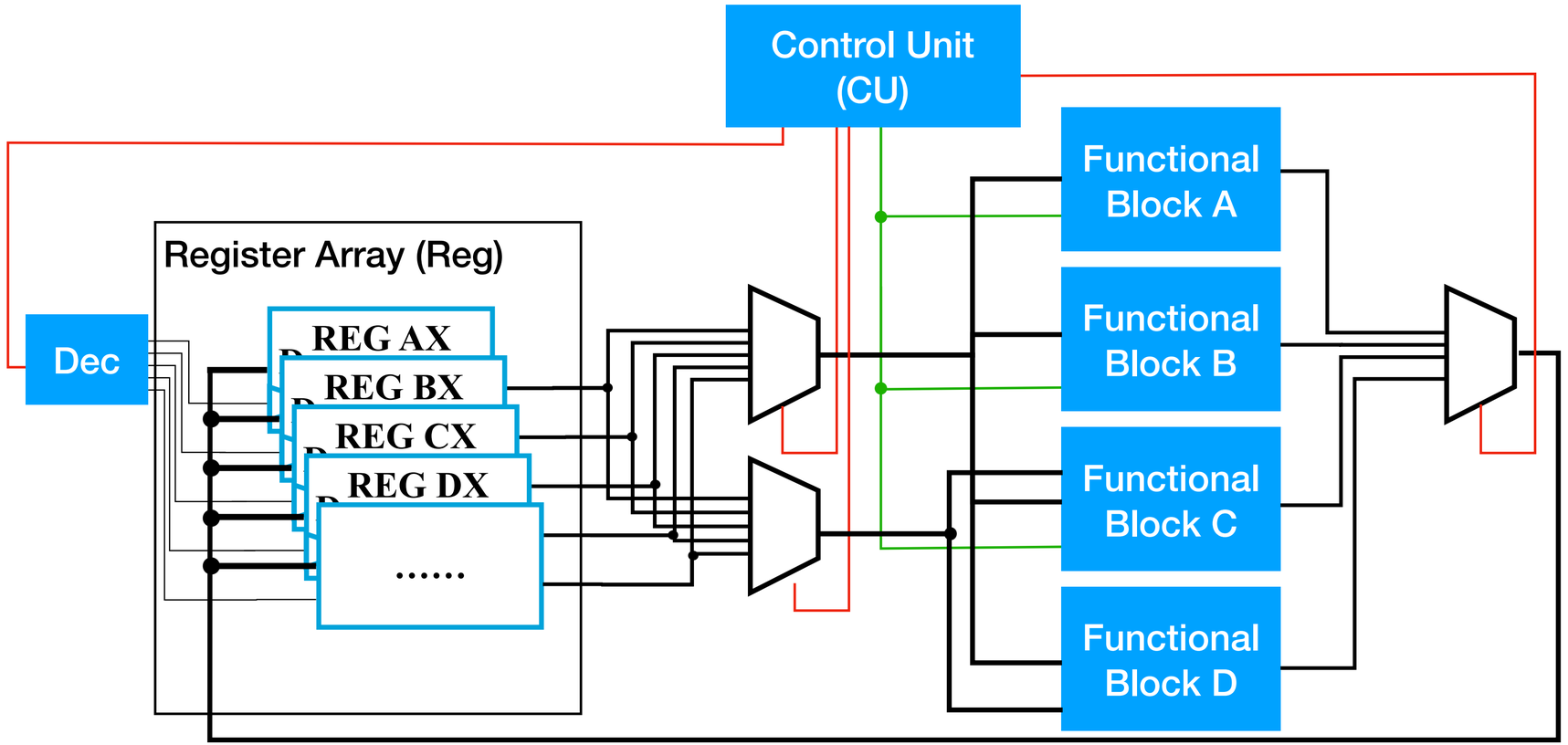
	Operator	Example
Assignment	<=	ax **<=** 12h
Reg. Transfer	<=	ax **<=** bx
Addition	+	ax **+** bx
Subtraction	-	ax **-** bx
Shift Left	sll	ax **sll** 2
Shift Right	srl	ax **srl** 2

	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

P0
Review

Example Datapath Architecture



Review

Implementation of Datapath II

Individual Functional Blocks

Summary

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)

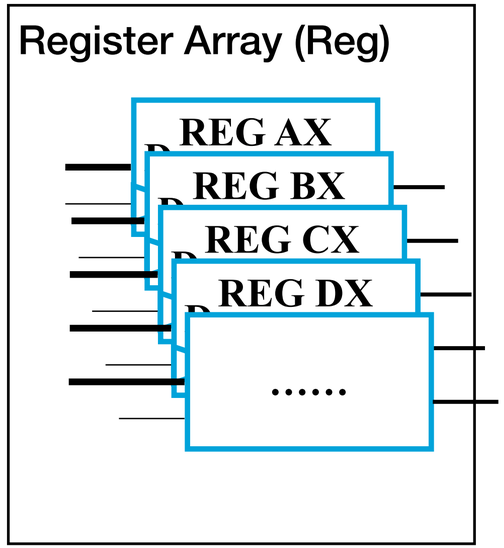
Concept

P1
Datapath

Single Register Microoperations

Red: Address
Green: Mode

Control Unit
(CU)



Functional
Block 1

...

Functional
Block N

Review

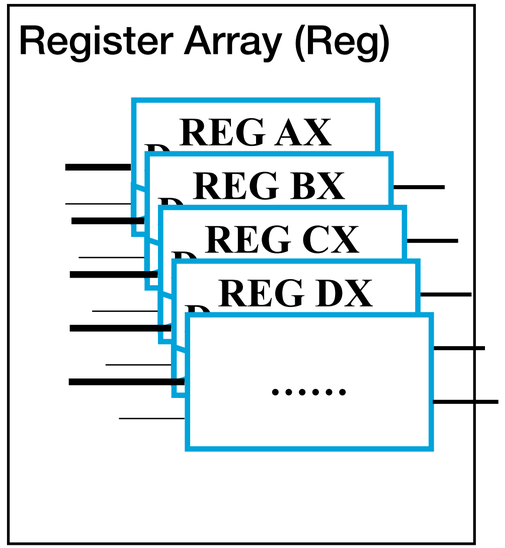
P1
Datapath

Single Register Microoperations

Red: Address
Green: Mode

- Select Input `REG` to be fed into Processing Blocks

Control Unit (CU)



Functional Block 1

...

Functional Block N

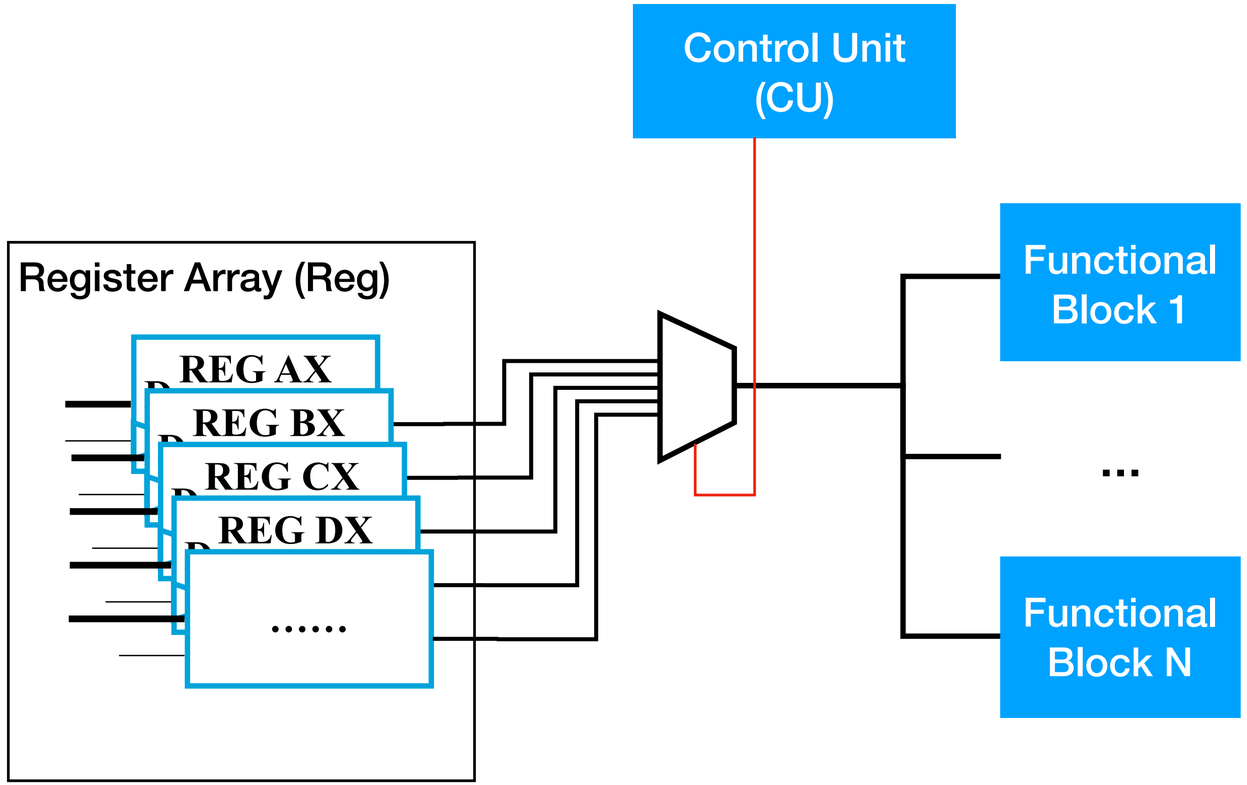
Review

P1
Datapath

Single Register Microoperations

Red: Address
Green: Mode

- Select Input `REG` to be fed into Processing Blocks



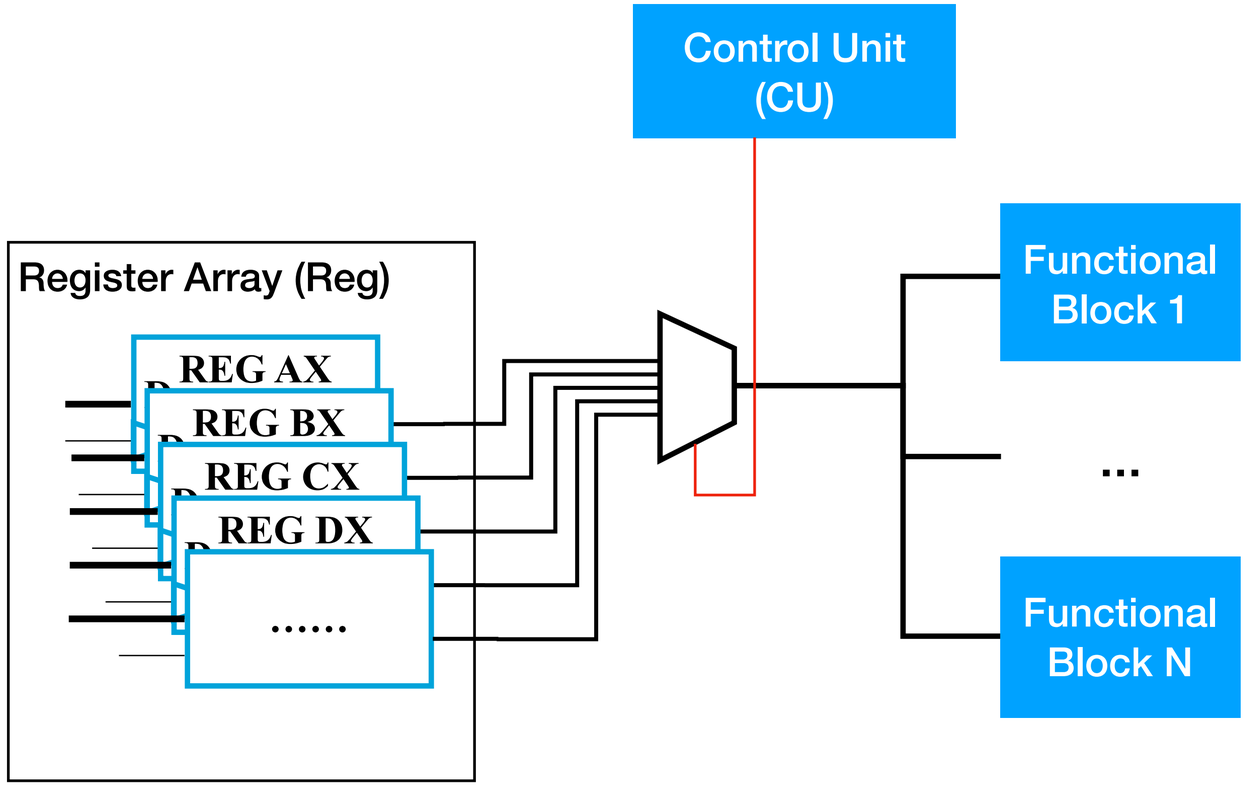
Review

P1
Datapath

Single Register Microoperations

Red: Address
Green: Mode

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



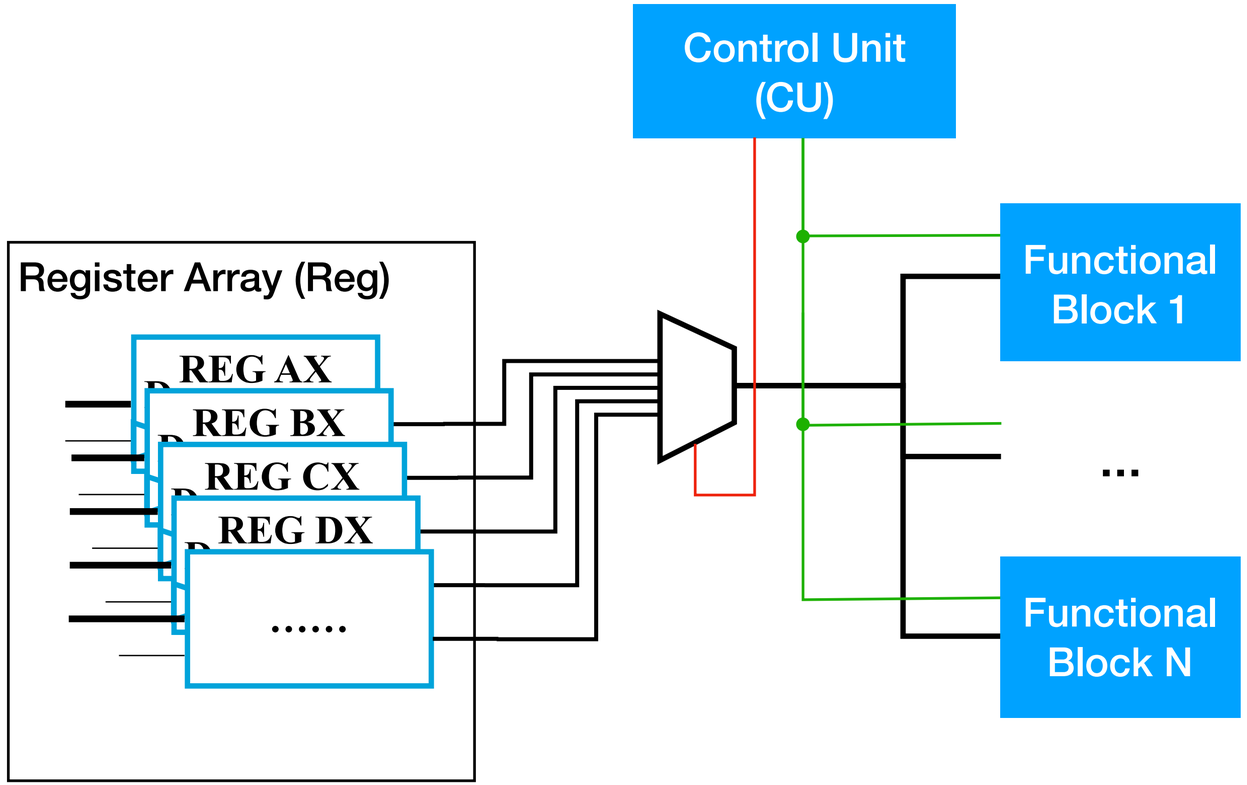
Review

P1
Datapath

Single Register Microoperations

Red: Address
Green: Mode

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



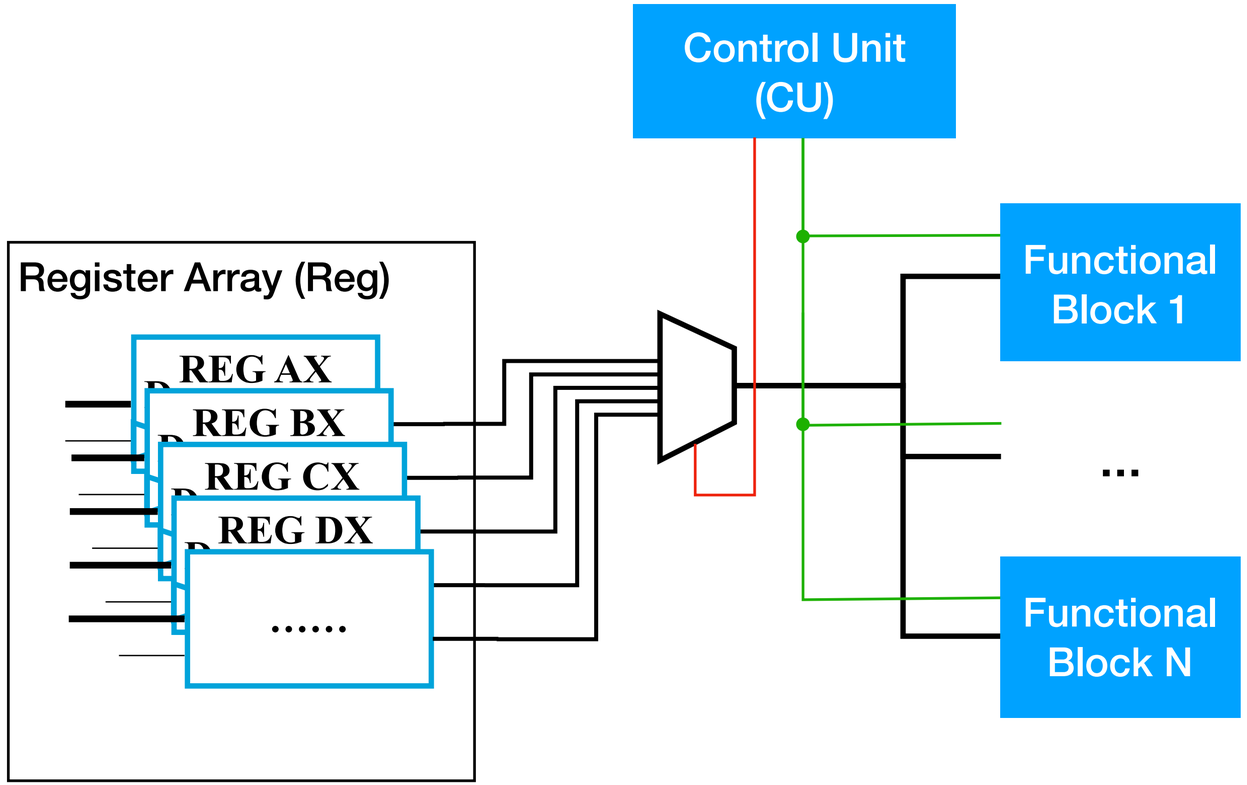
Review

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`



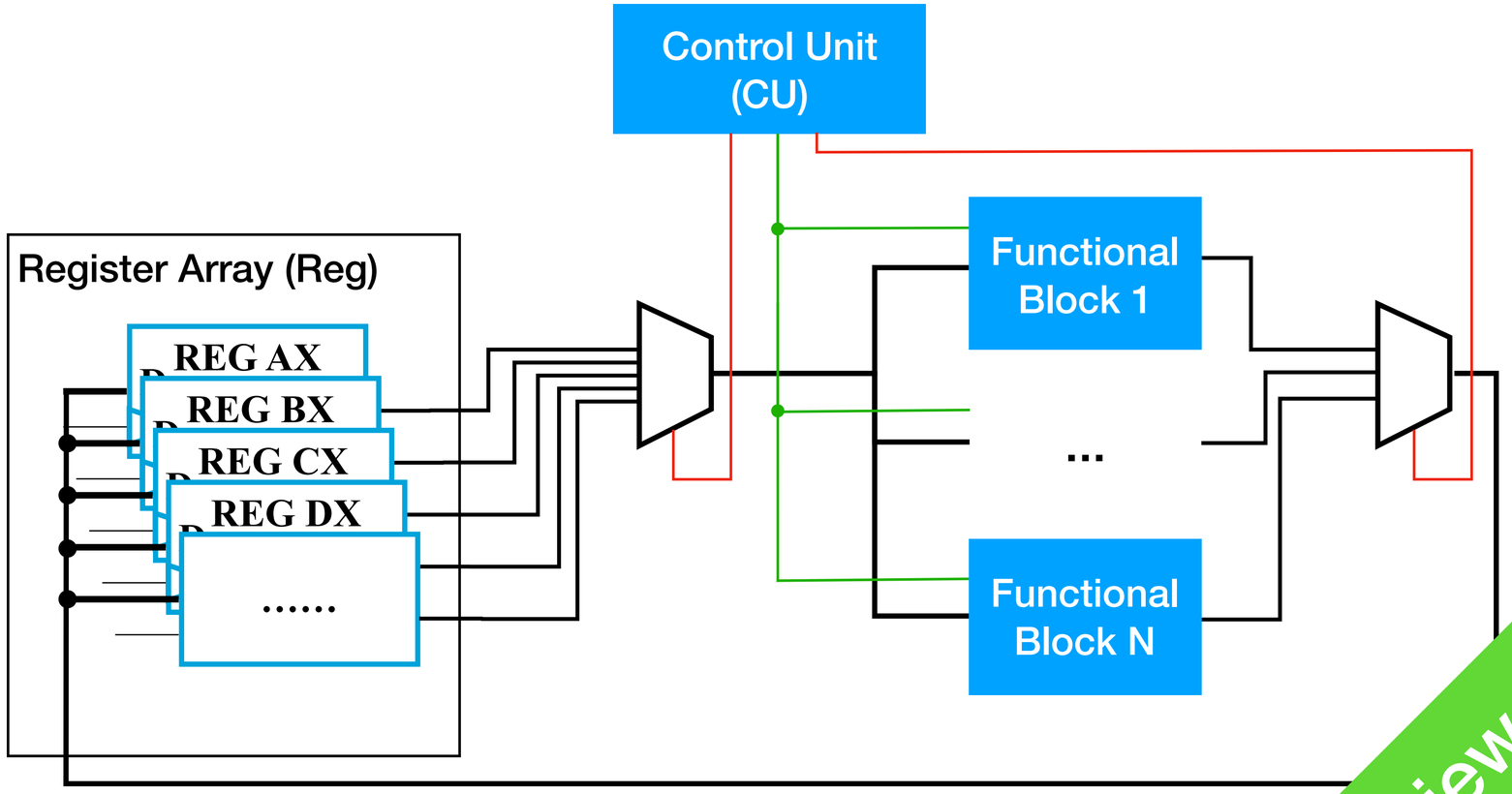
Review

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`



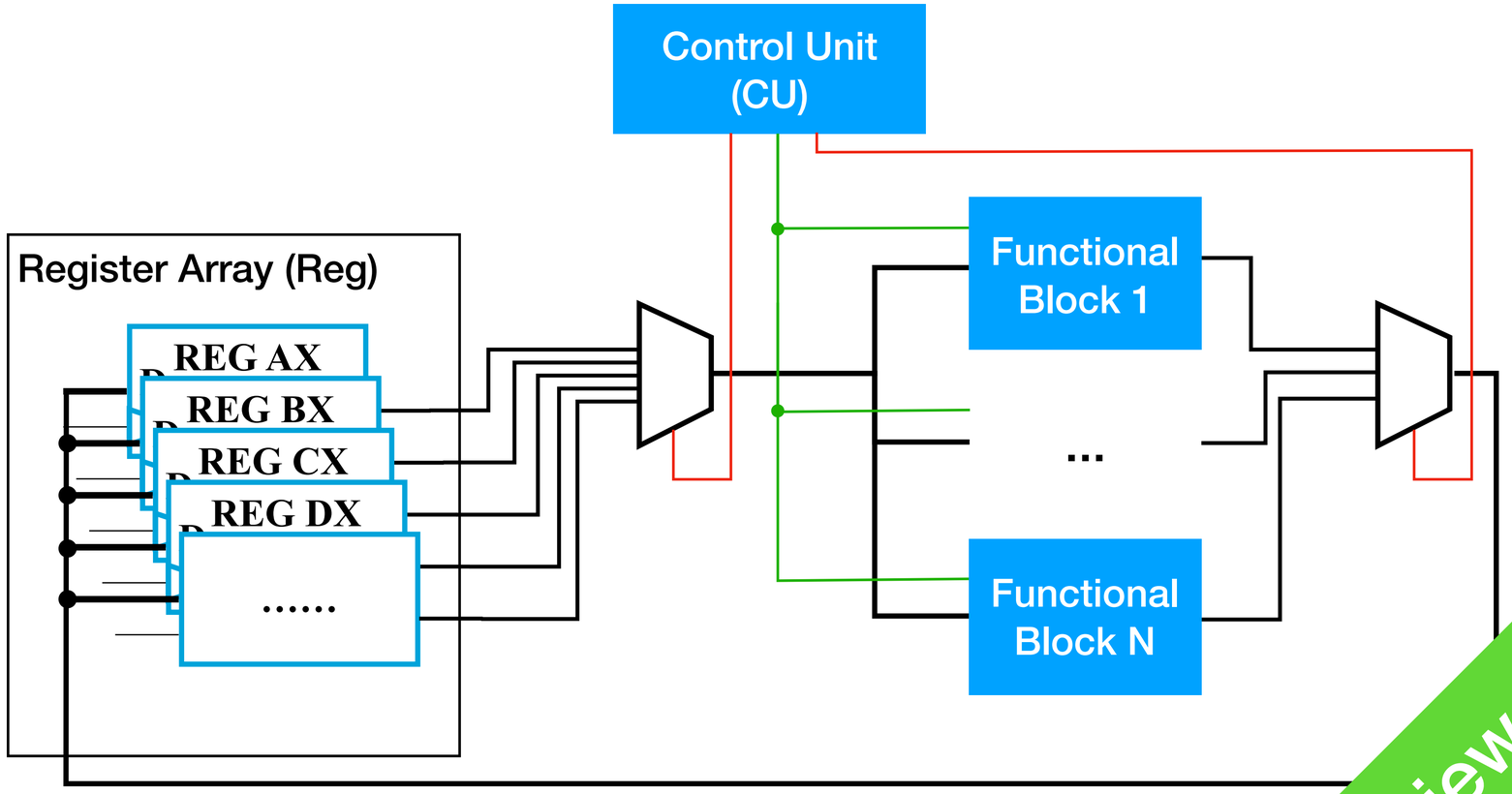
Review

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`



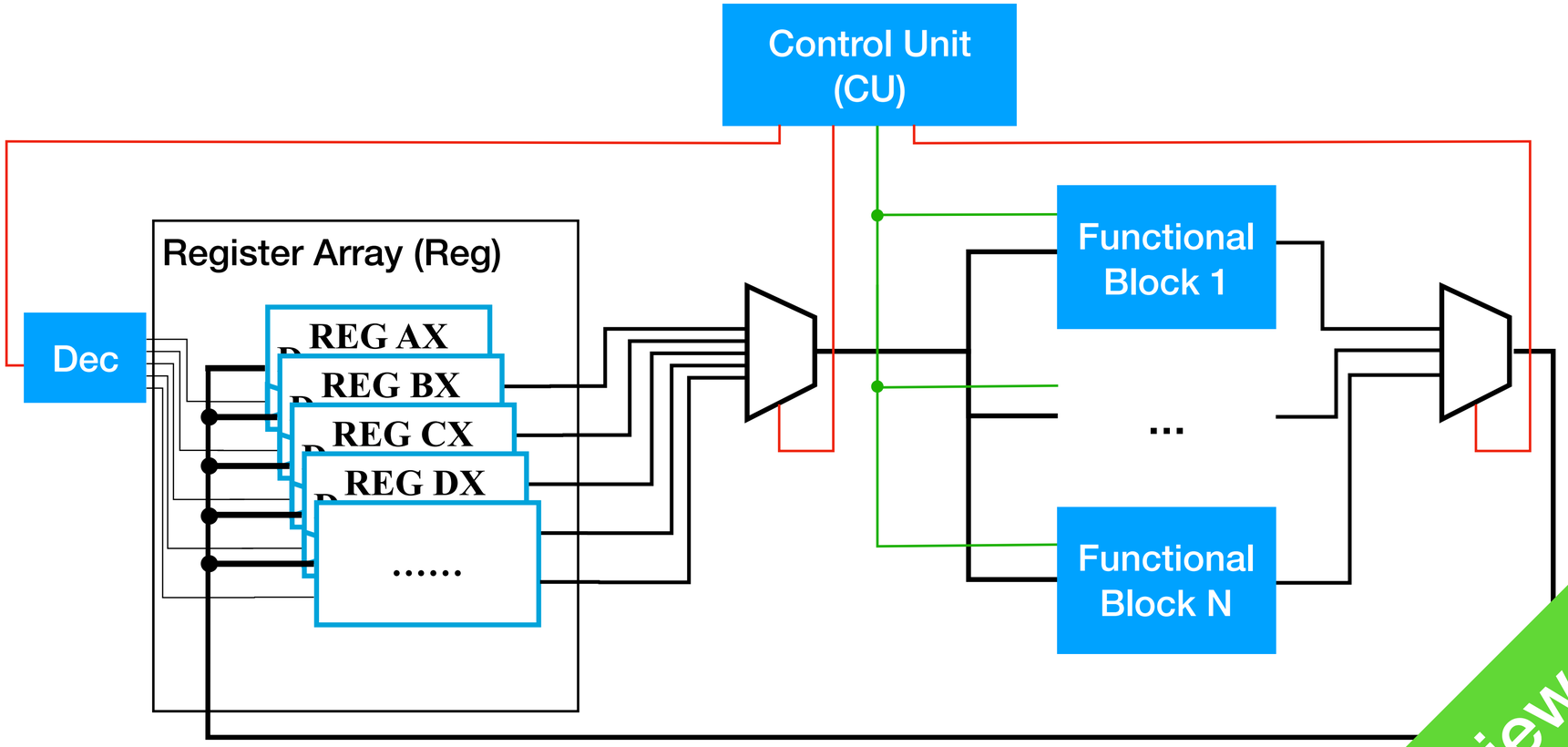
Review

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`



Review

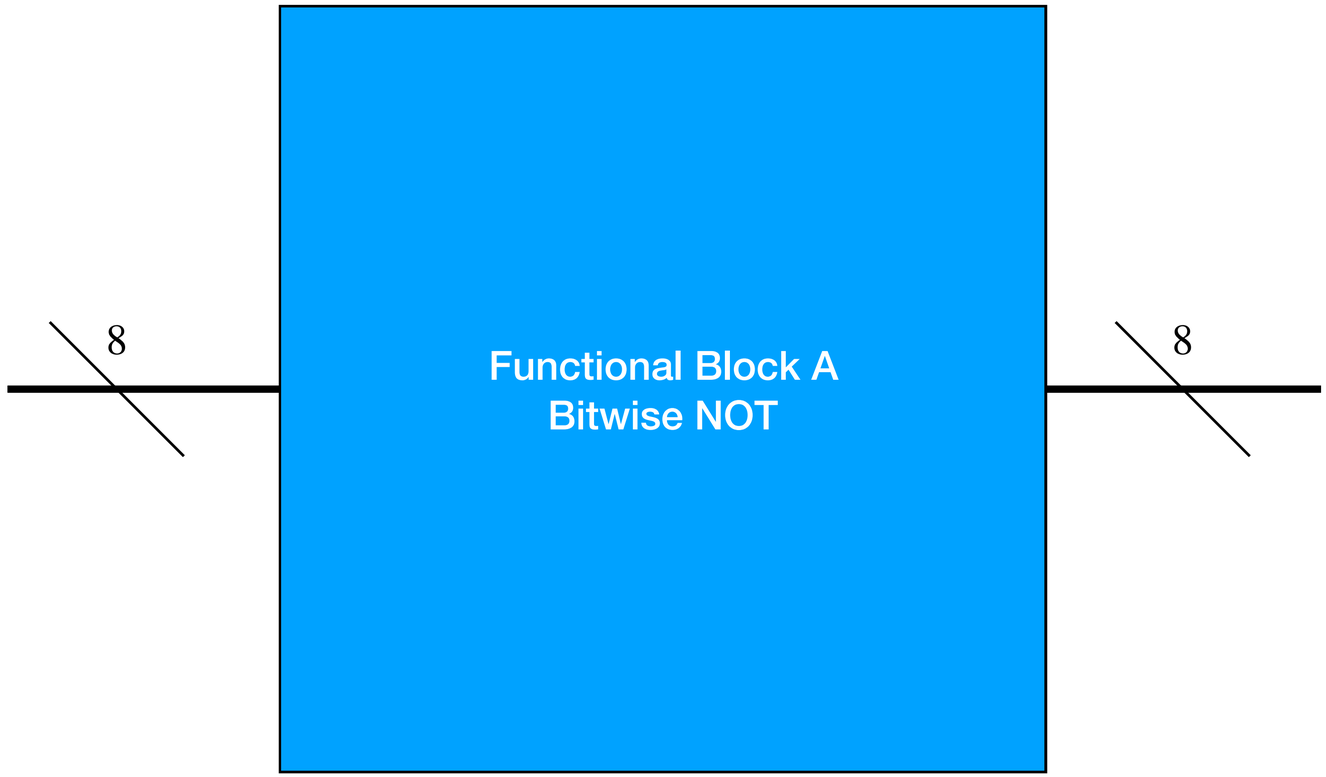
Single Register Microoperations

- Each individual functional block design identical to the functional blocks we've covered in Lecture 3
- For operations completed in a single CLK tick, use the 5 step design procedures

Concept

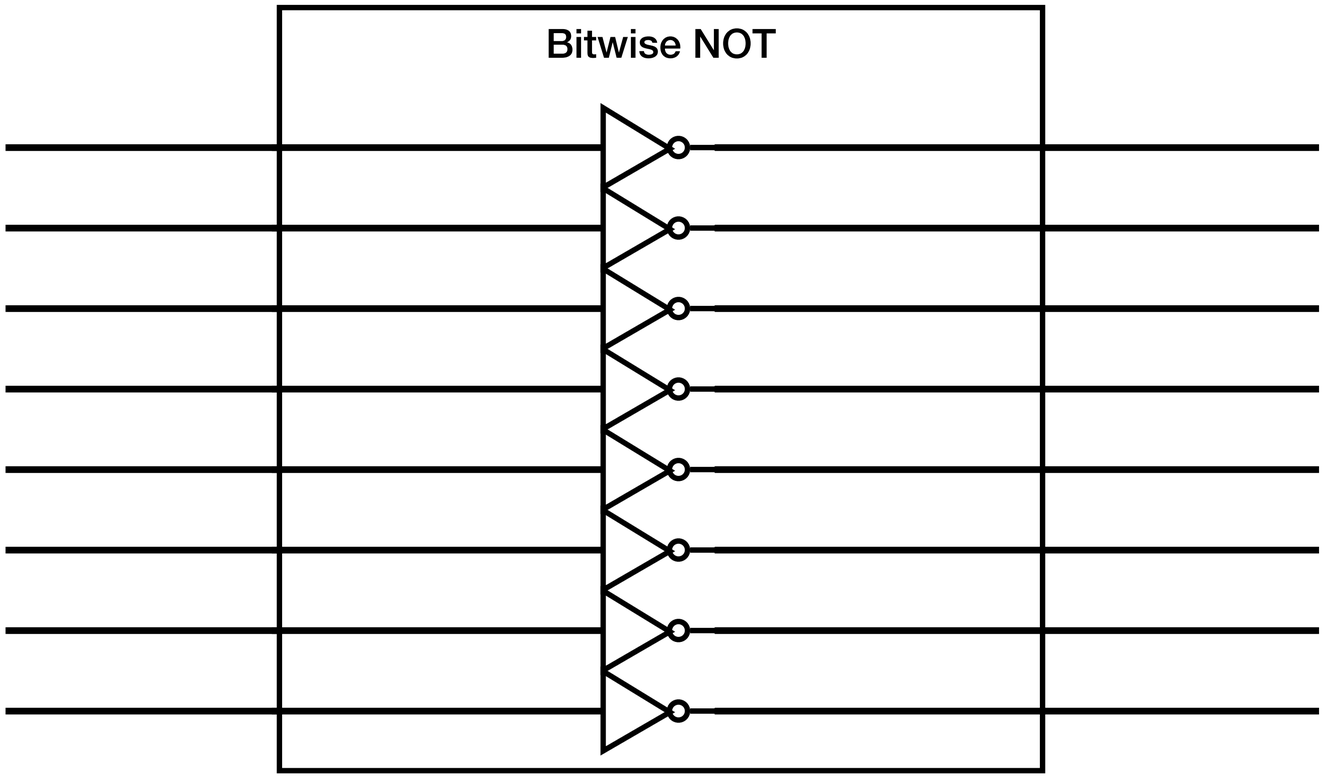
Single Register Microoperations

- Bitwise NOT



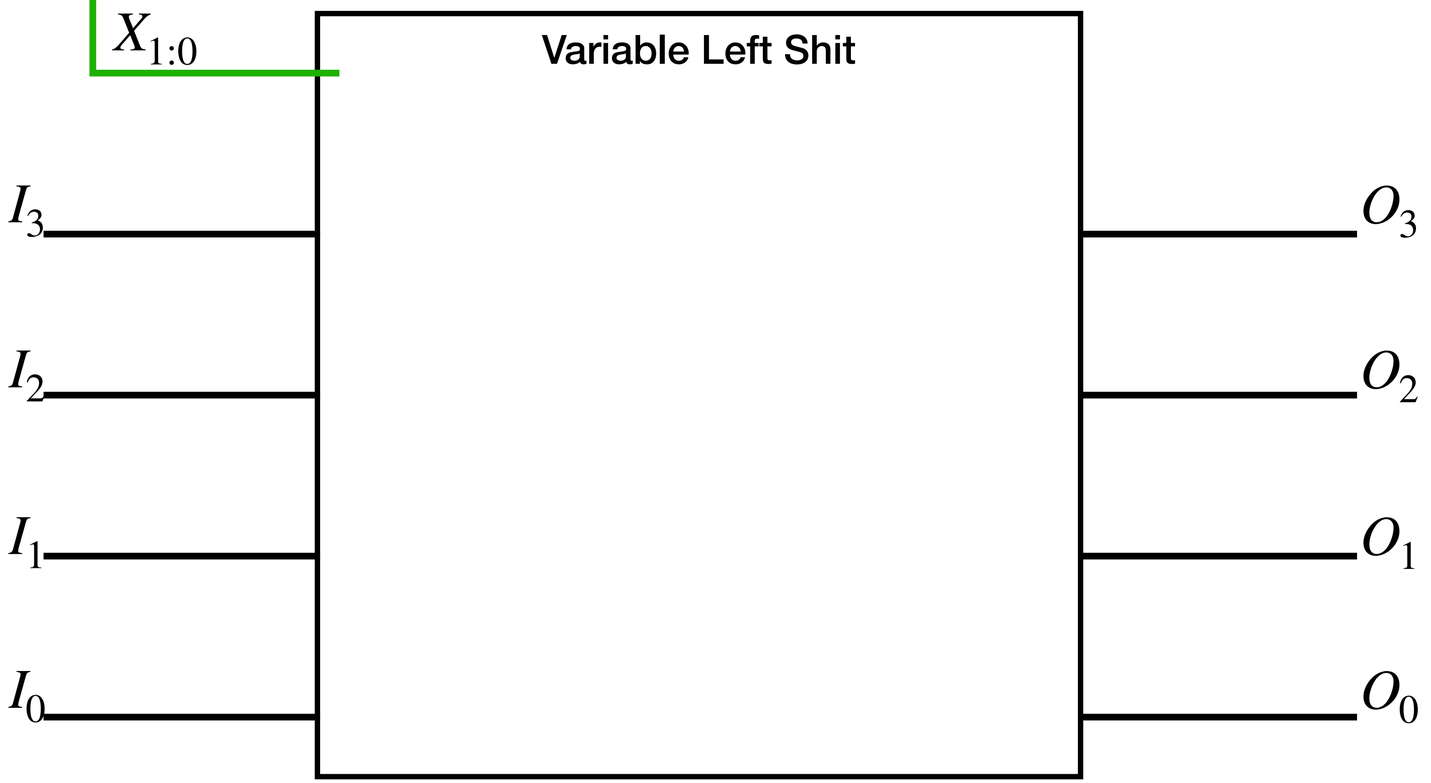
Single Register Microoperations

- Bitwise NOT



Single Register Microoperations

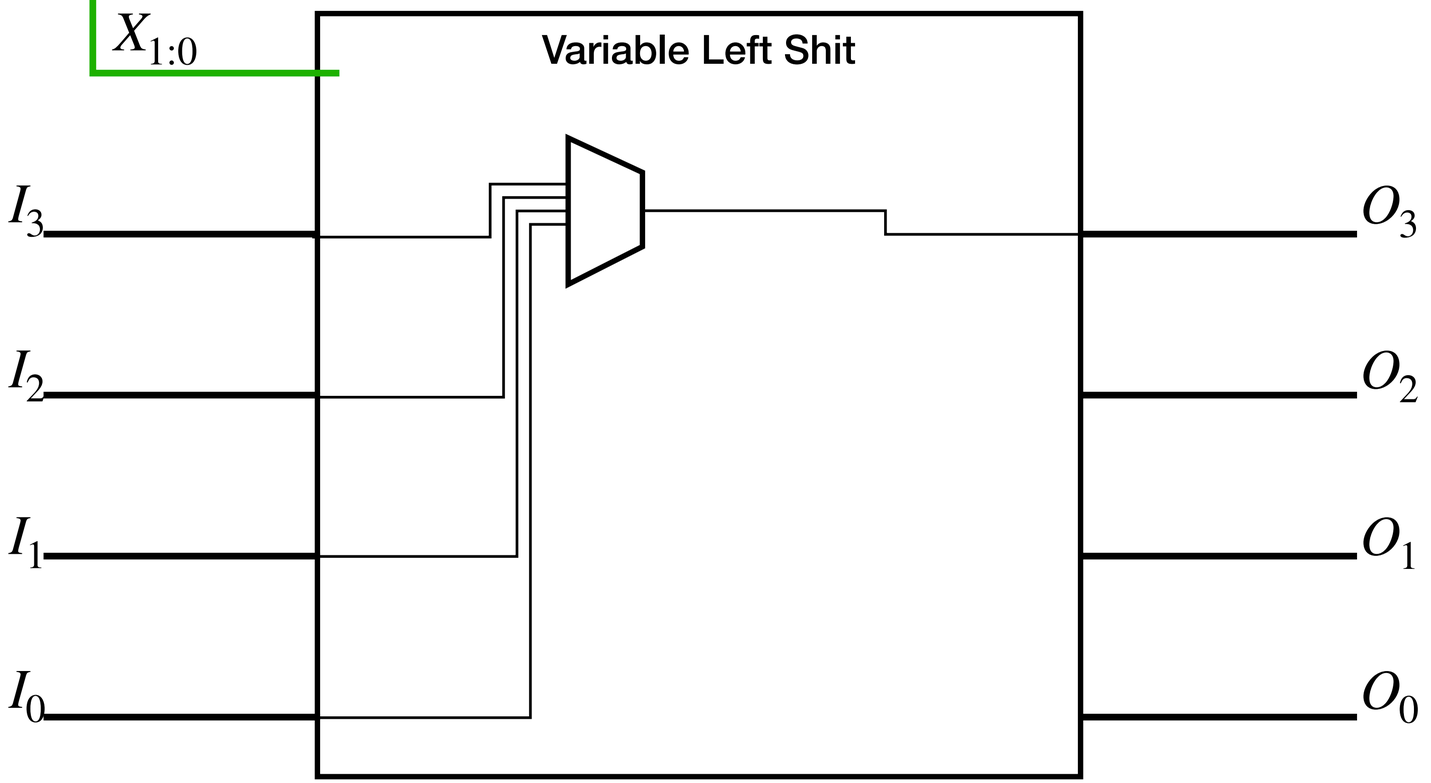
- Left Shifter: variable shifts
  Assuming 4-bit registers, another input $X_{1:0}$ controls the number of shifts



Exercise

Single Register Microoperations

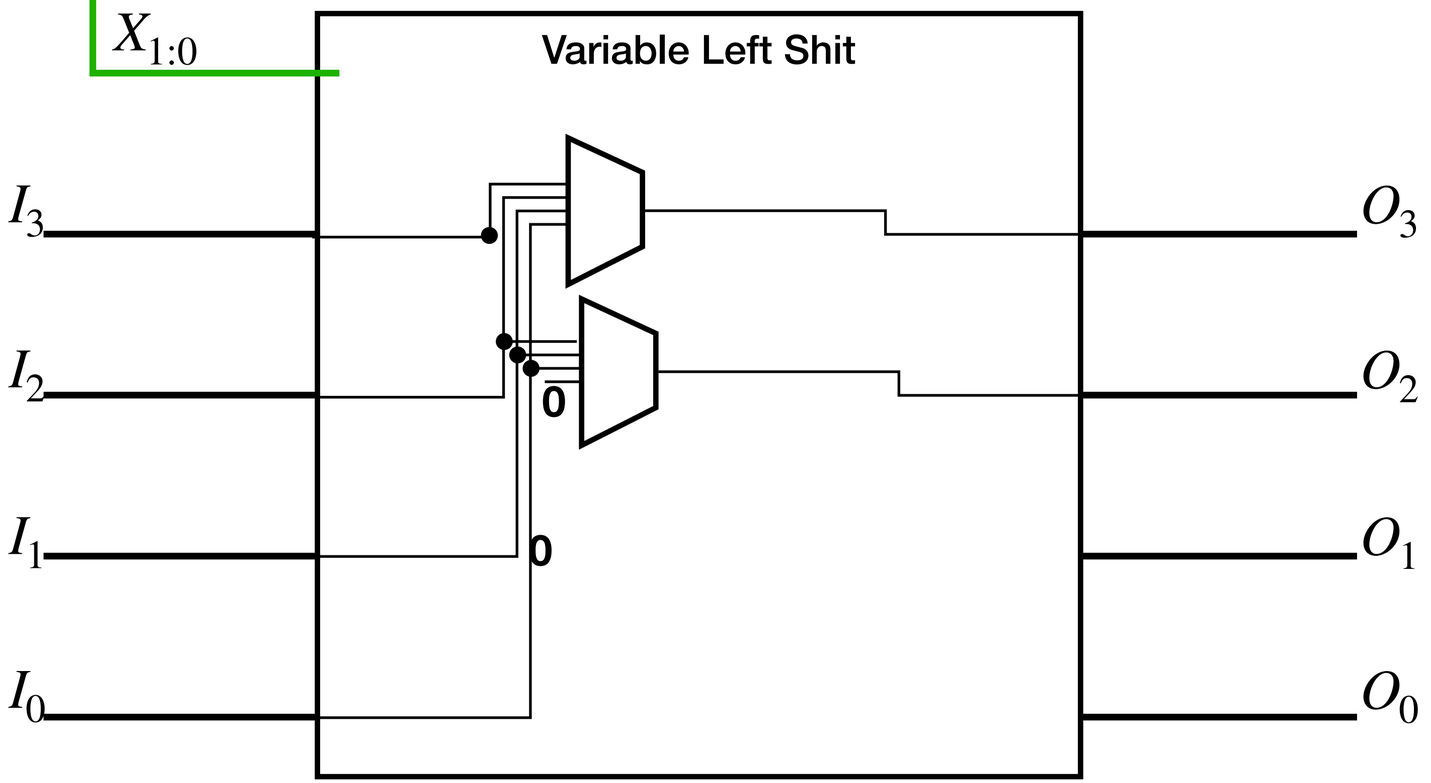
- Left Shifter: variable shifts
  Assuming 4-bit registers, another input $X_{1:0}$ controls the number of shifts



Exercise

Single Register Microoperations

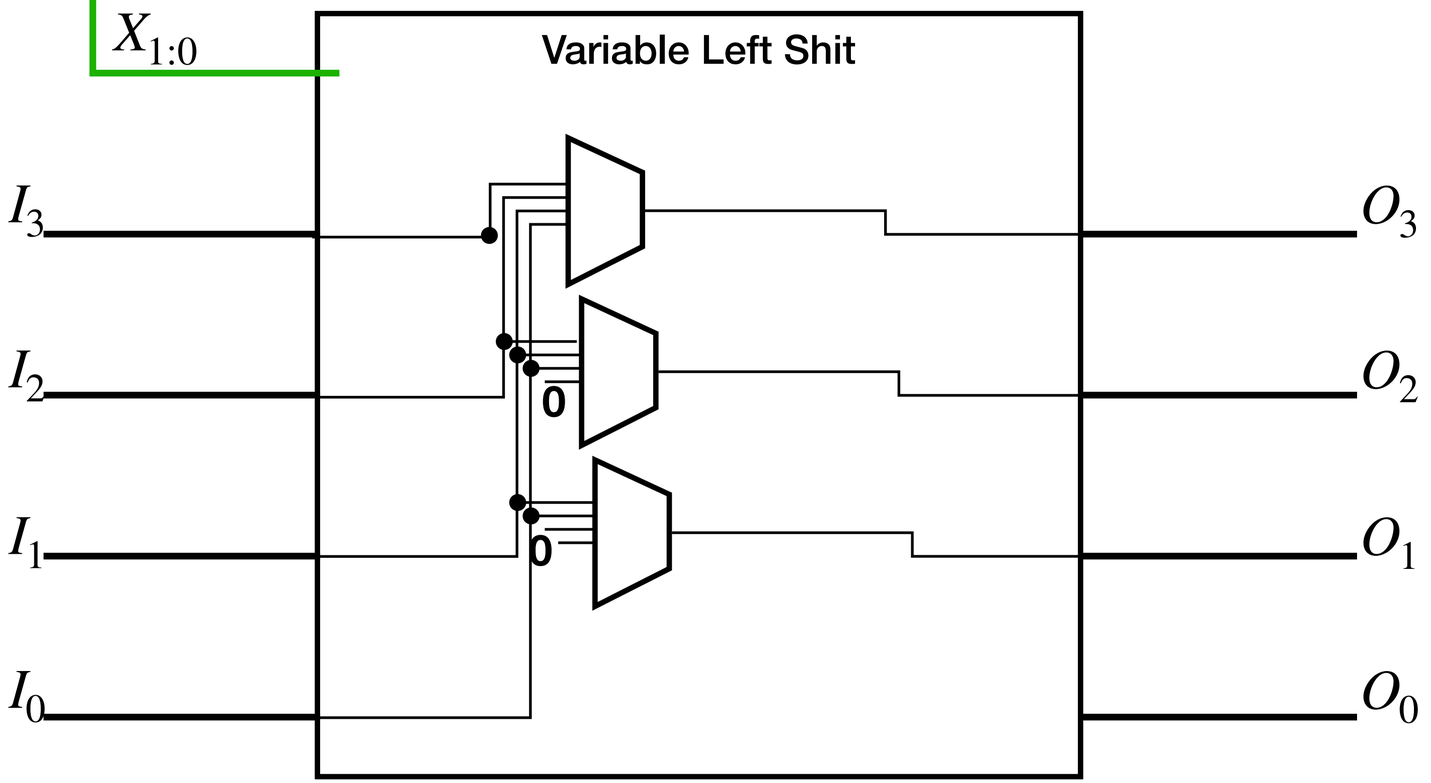
- Left Shifter: variable shifts
  Assuming 4-bit registers, another input $X_{1:0}$ controls the number of shifts



Exercise

Single Register Microoperations

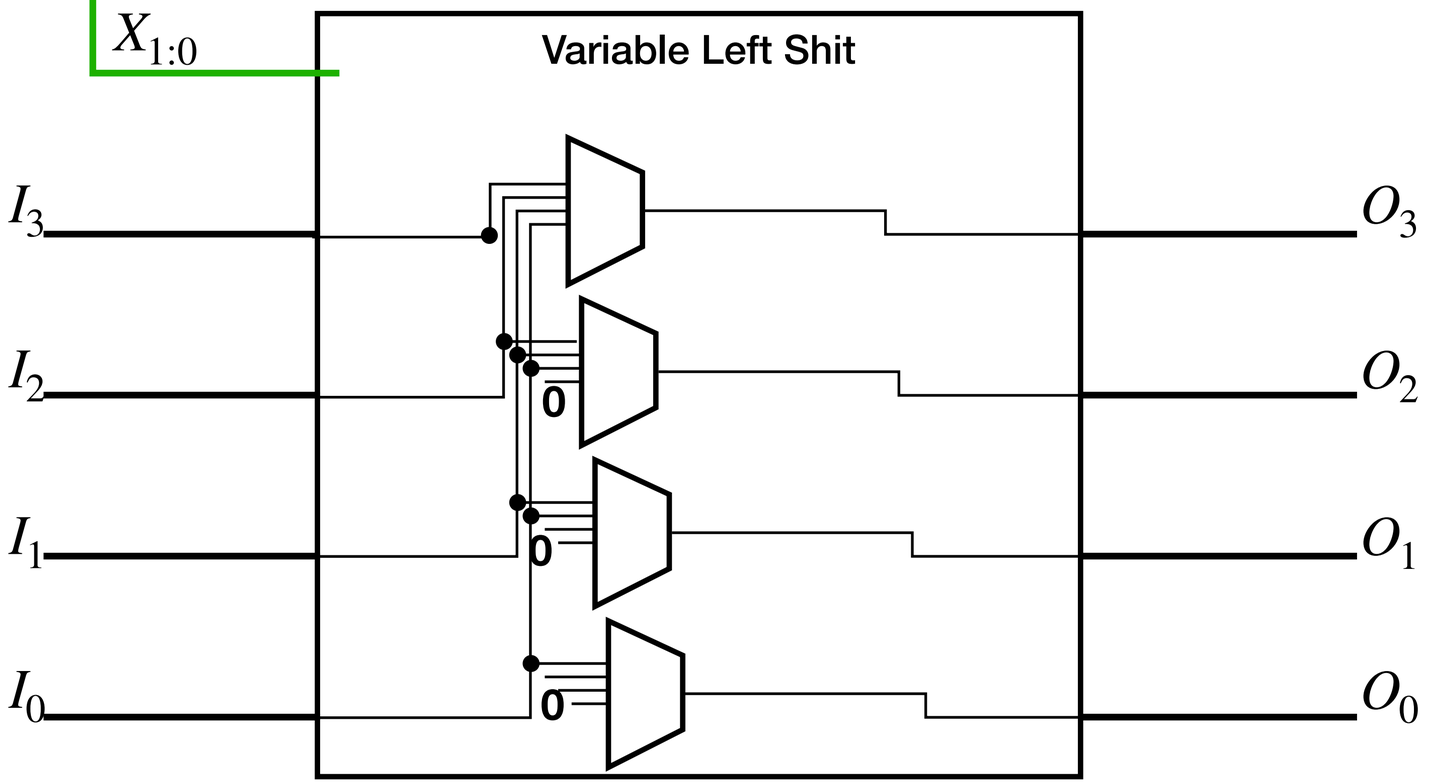
- Left Shifter: variable shifts
  Assuming 4-bit registers, another input $X_{1:0}$ controls the number of shifts



Exercise

Single Register Microoperations

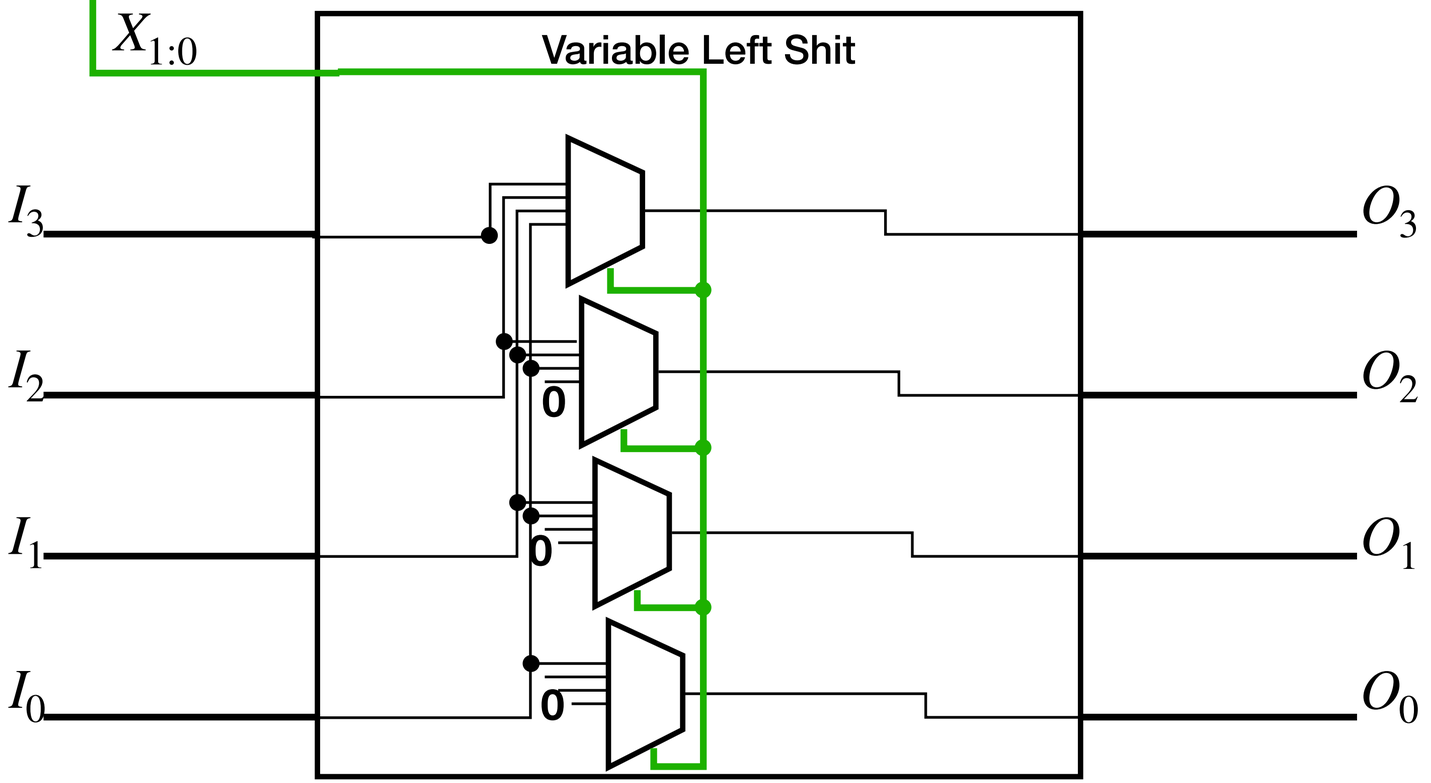
- Left Shifter: variable shifts
  Assuming 4-bit registers, another input $X_{1:0}$ controls the number of shifts



Exercise

Single Register Microoperations

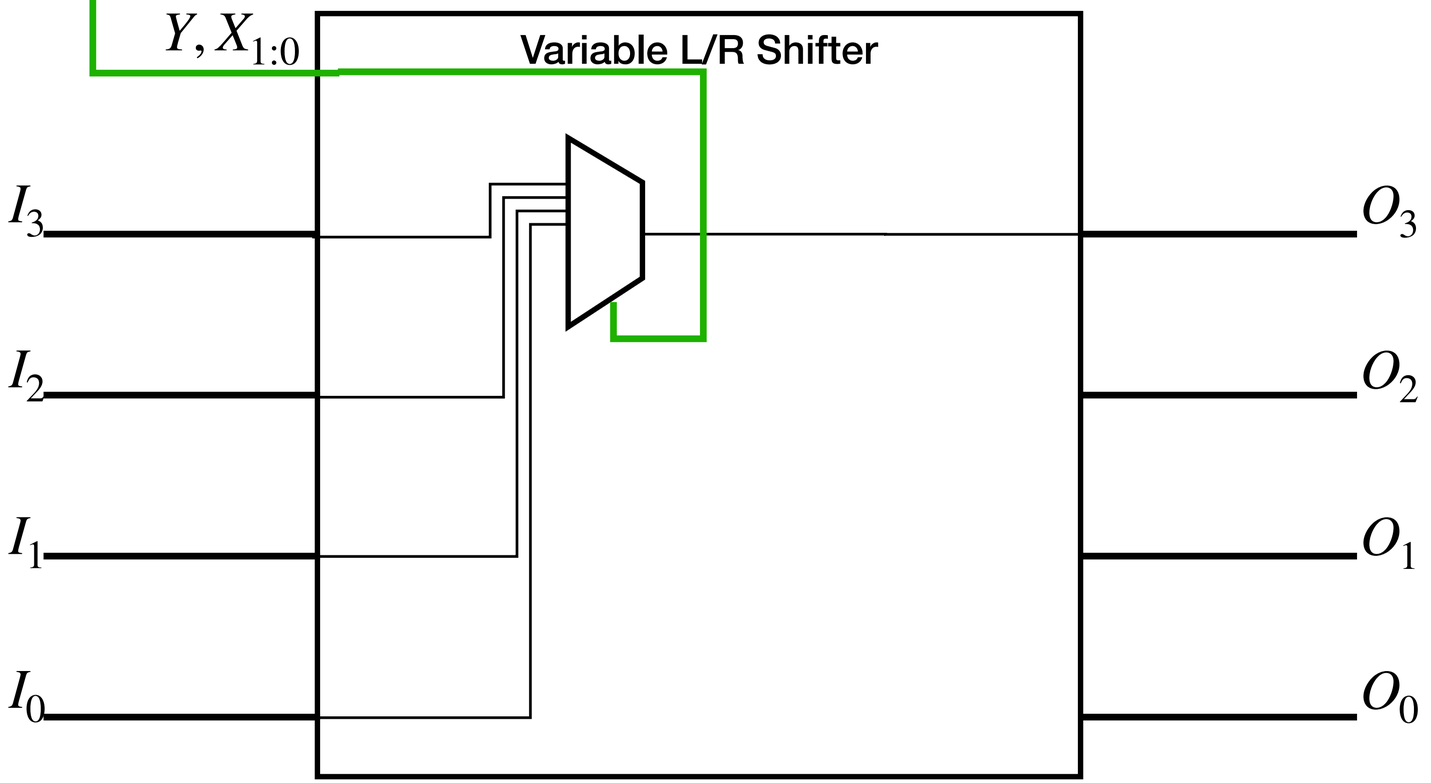
- Left Shifter: variable shifts
  Assuming 4-bit registers, another input $X_{1:0}$ controls the number of shifts



Exercise

Single Register Microoperations

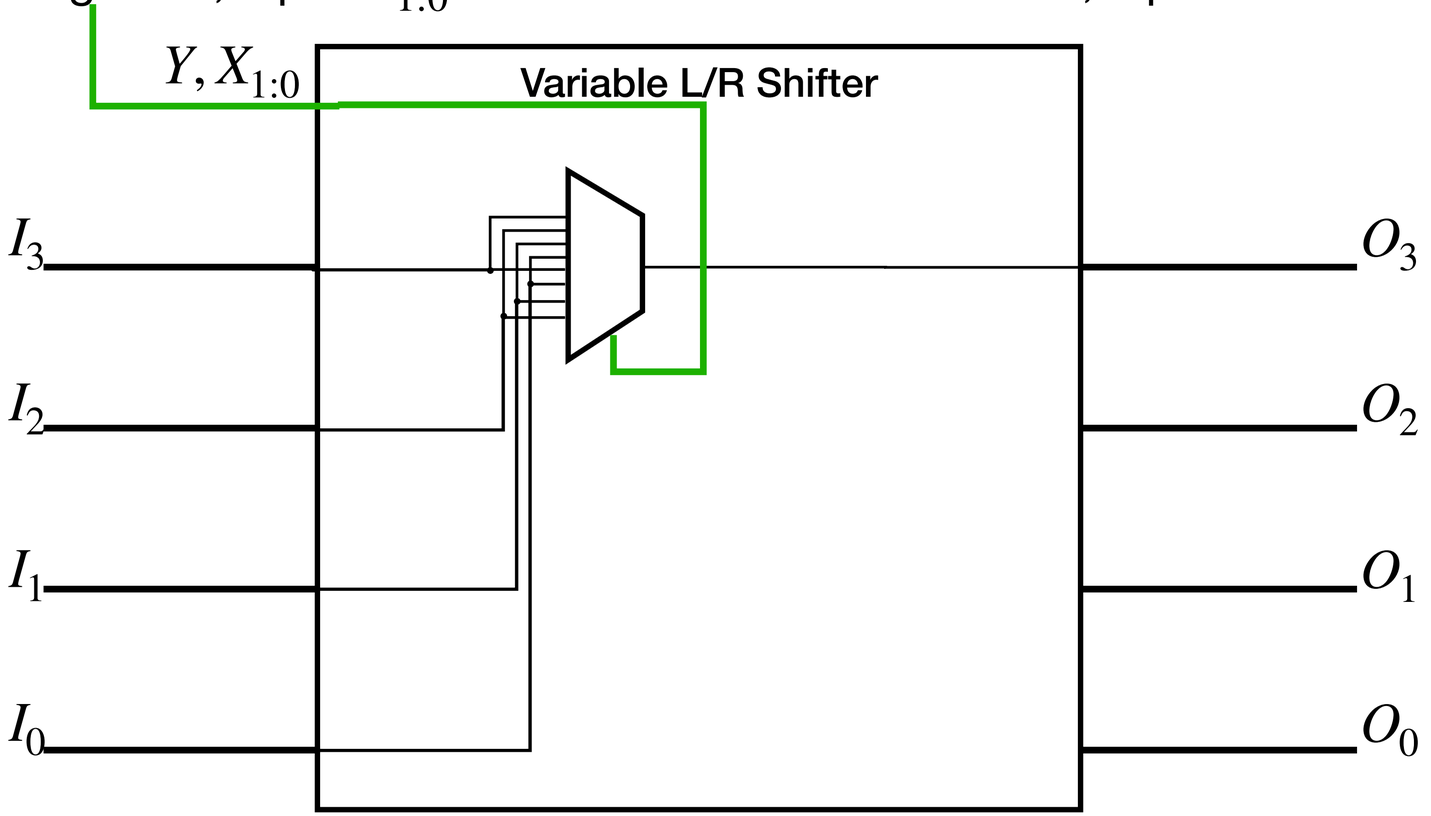
- Left/Right Shifter: variable shifts
  Assuming 4-bit registers, input $X_{1:0}$ controls the number of shifts, input Y for direction (0:L; 1:R)



Exercise

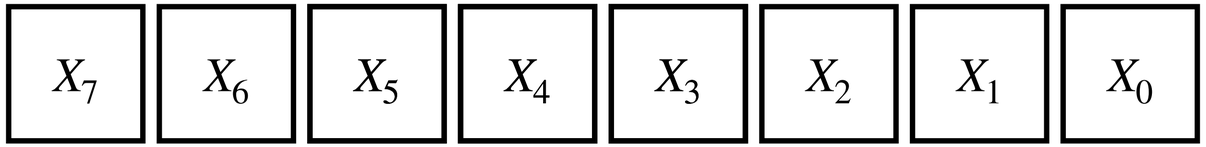
Single Register Microoperations

- Left/Right Shifter: variable shifts
  Assuming 4-bit registers, input $X_{1:0}$ controls the number of shifts, input Y for direction (0:L; 1:R)



Exercise

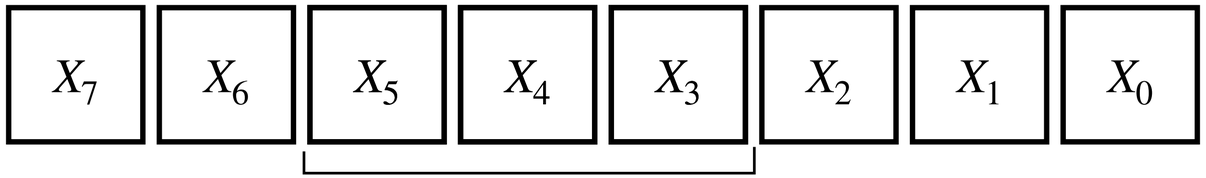
Single Register Microoperations



- Vector: I down to J (e.g. `ax(5 down to 3)`: $I = 5, J = 3$)

Exercise

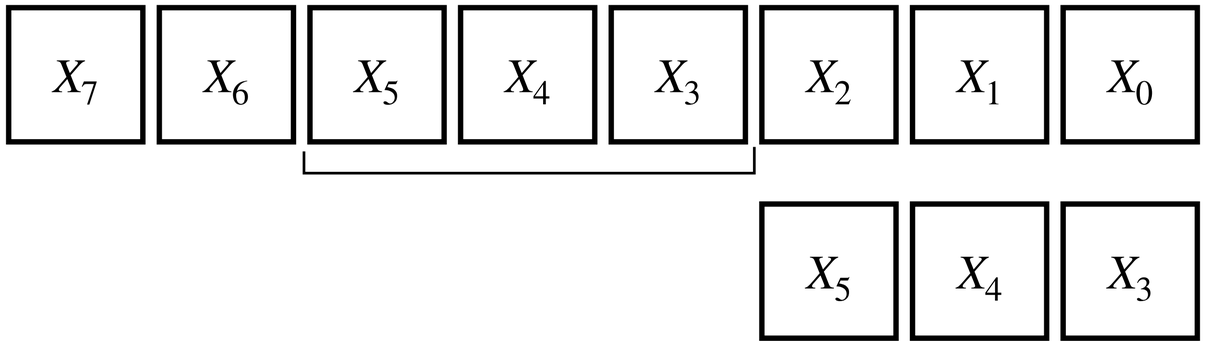
Single Register Microoperations



- Vector: I down to J (e.g. `ax(5 down to 3)`: $I = 5, J = 3$)

Exercise

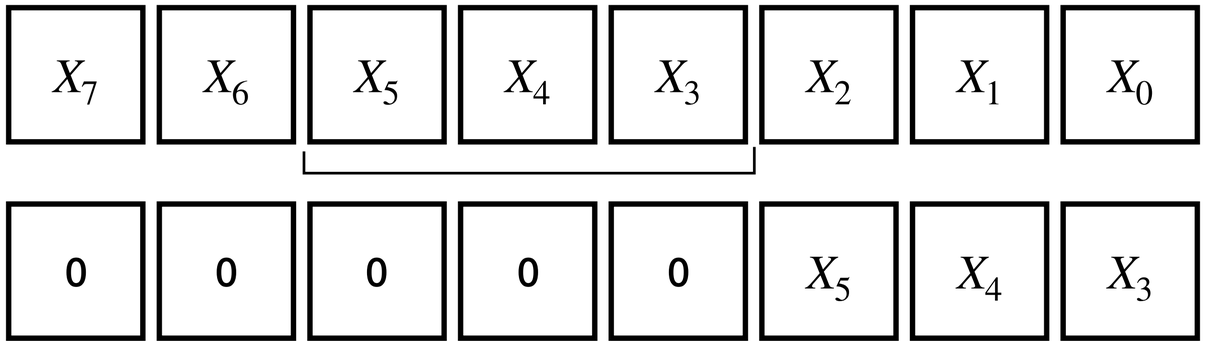
Single Register Microoperations



- Vector: I down to J (e.g. `ax(5 down to 3)`: $I = 5, J = 3$)

Exercise

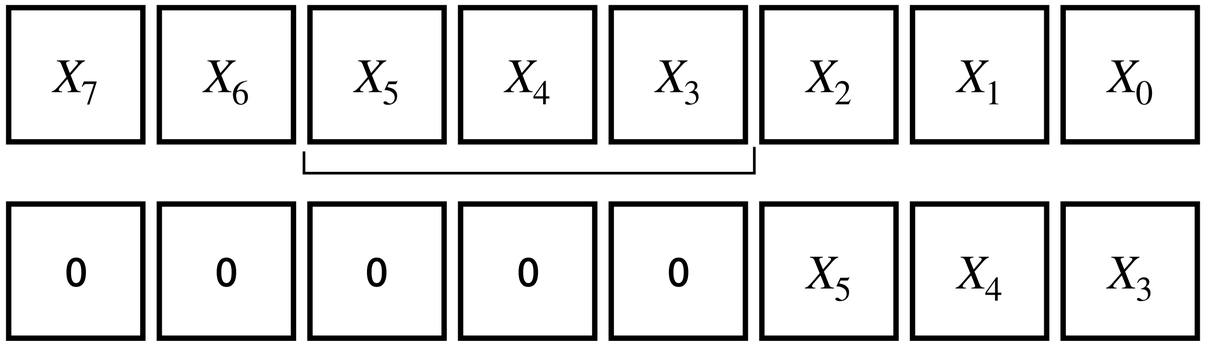
Single Register Microoperations



- Vector: I down to J (e.g. `ax(5 down to 3)`: $I = 5, J = 3$)

Exercise

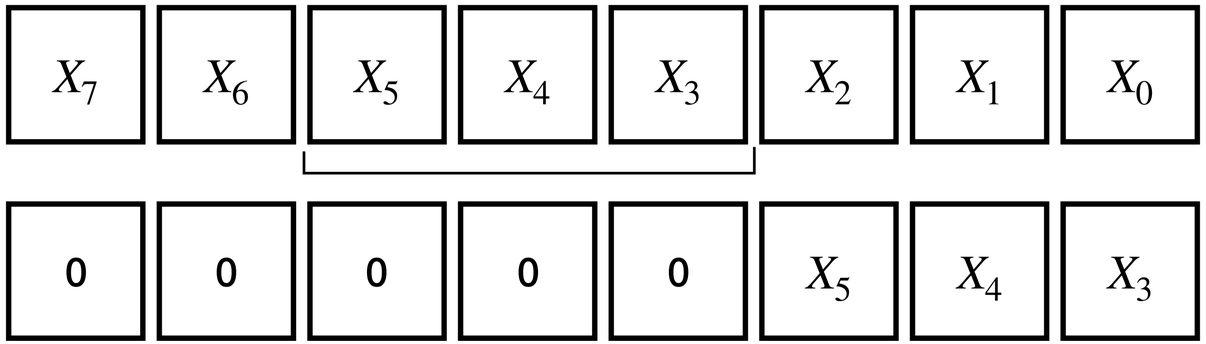
Single Register Microoperations



- Vector: I down to J (e.g. `ax(5 down to 3)`: $I = 5, J = 3$)
  - Decompose the problem

Exercise

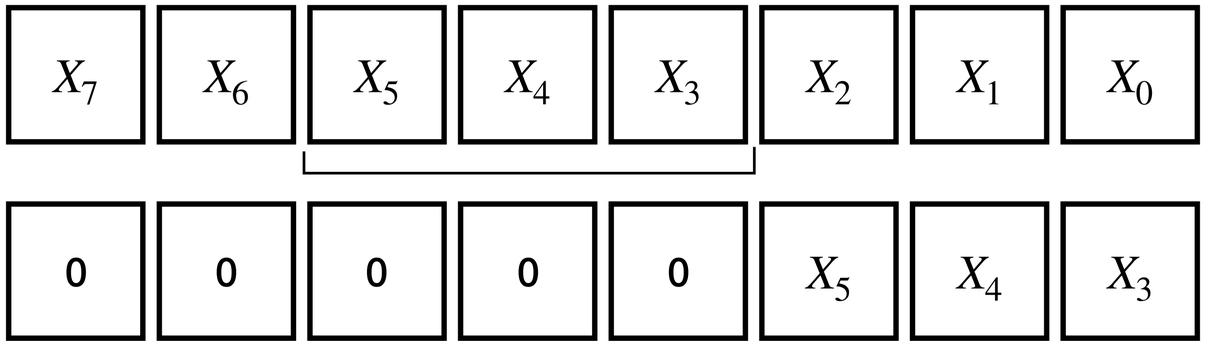
Single Register Microoperations



- Vector: I down to J (e.g. `ax(5 down to 3)`: $I = 5, J = 3$)
  - Decompose the problem
  - J: the number of right shifts to perform

Exercise

Single Register Microoperations



- Vector: I down to J (e.g. `ax(5 down to 3)`: $I = 5, J = 3$)
  - Decompose the problem
  - J: the number of right shifts to perform
  - I: masking out everything before position I to 0

Exercise

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