Lecture

- Last Time: Combinational Circuits
  - Circuits → Boolean Algebra -> Circuits
  - Building up larger comments
    - half and full adders
    - sum of products
  - 1-bit → N-bit operations
  - Binary Addition and Subtraction
  - Binary Coded Decimal (BCD) Addition

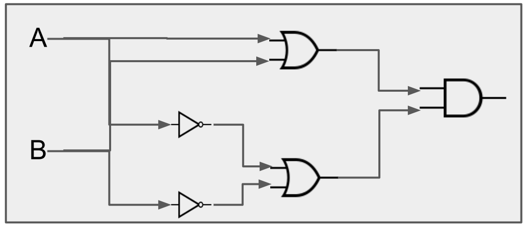
- Today:
  - On-Off Switch: Not Data, but a Control Line
  - Decoders and Multiplexers
  - 1-bit Arithmetic and Logic Unit (ALU)
  - Schematic of the CPU
  - Schematic of Main Memory

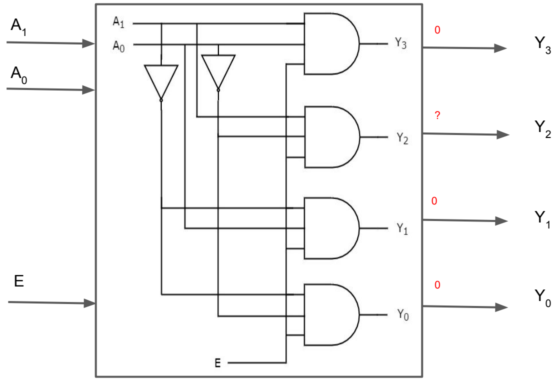
- Nextime: State and MicroArchitecture
  - Sequential Circuits: clocks, latches, and flip/flops:
  - CPU Control: PC/IR
  - MIPS pipeline architecture

Data Lines and Control Lines

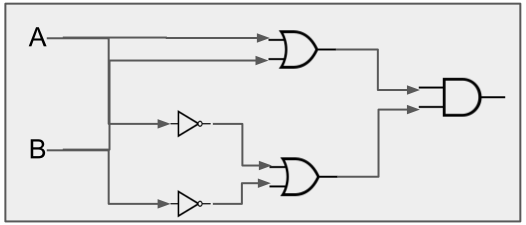
- All lines (wires) carry either a 0 or a 1
    - In the XOR circuit, we perceive these values to be data
    - In other circuits, e.g., decoder, some lines are for control

- On-Off Switch:
    - Use to turn on or turn off circuits

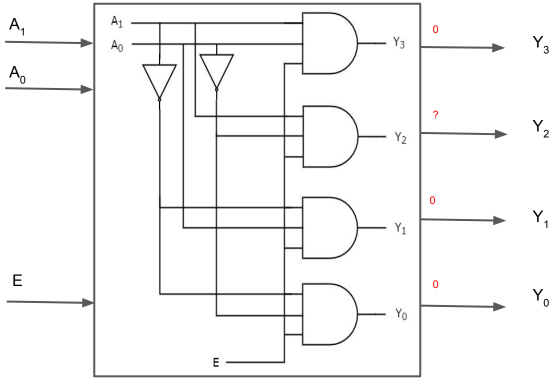




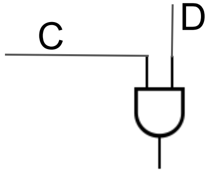
Data Lines and Control Lines



- All lines (wires) carry either a 0 or a 1
  - In the XOR circuit, we perceive these values to be data
  - In other circuits, e.g., decoder, some lines are for control

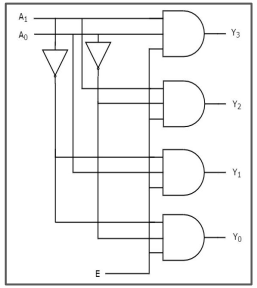


- On-Off Switch:
  - Use to turn on or turn off circuits
  - C is a control line, D is a data line

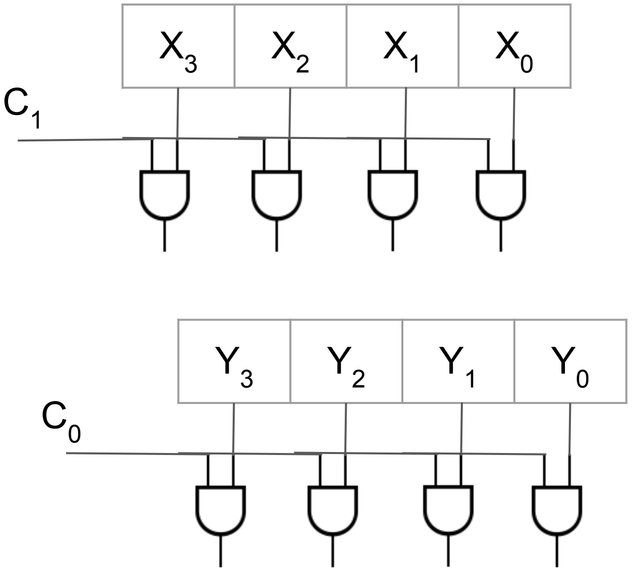


D	**C**	**Output (and)**
0	$\boxed{0}$ (Off)	0 (Off)
$\boxed{0}$	1 (On)	$\boxed{0}$
1	$\boxed{0}$ (Off)	0 (Off)
$\boxed{1}$	1 (On)	$\boxed{1}$

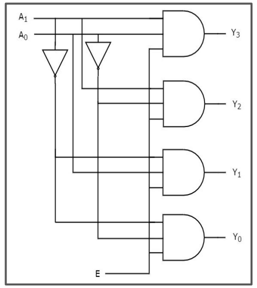
Decoder Motivation



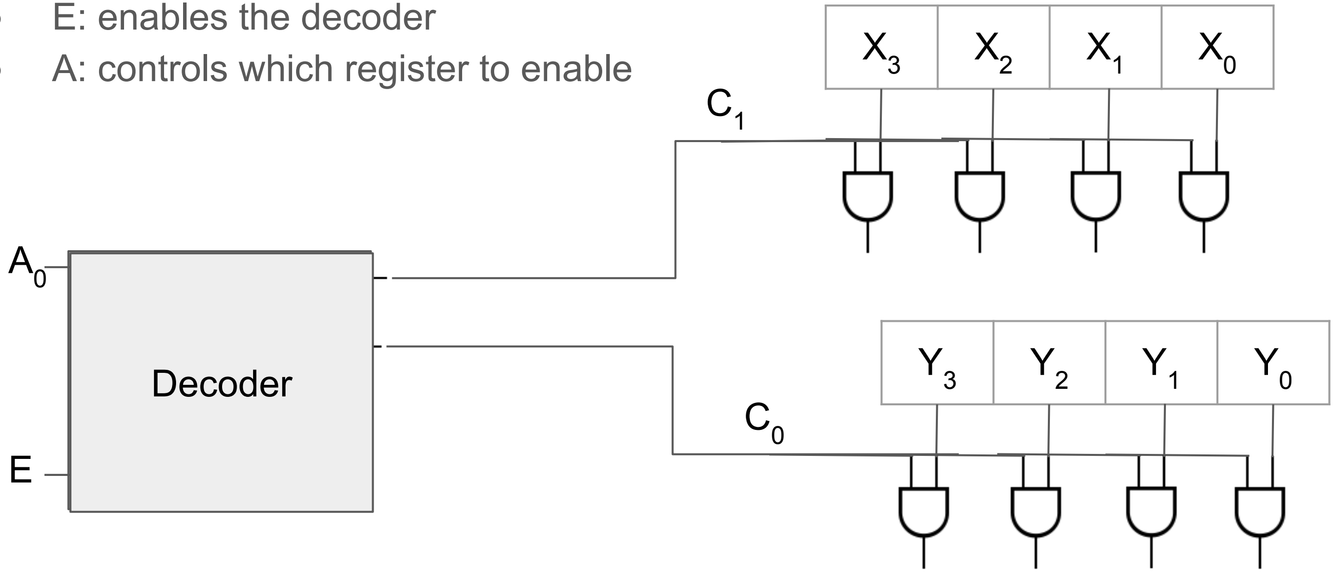
- Consider a *two* 4-bit registers
- Use an On-Off switch to enable all the data lines from a register
    - C_0 and C_1 controls whether or not the register is selected



Decoder Motivation



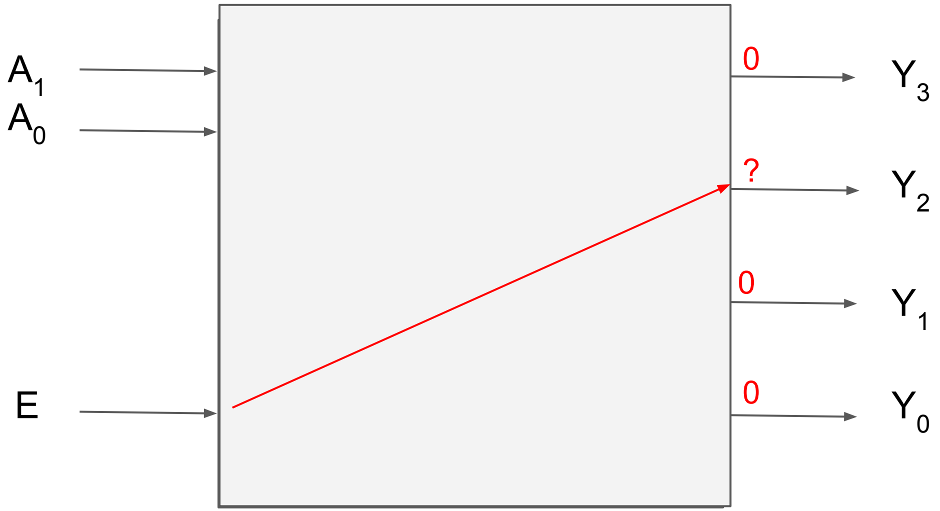
- Consider a *two* 4-bit registers
- Use an On-Off switch to enable all the data lines from a register
- Use 1-to-2 Decoder to select which register:
  - E: enables the decoder
  - A: controls which register to enable



Decoder: 2-to-4

- N+1 Inputs:
  - E: An "enable" line to active the circuit
  - A: Think of it as a binary number
- 2^n Outputs: A output line is set

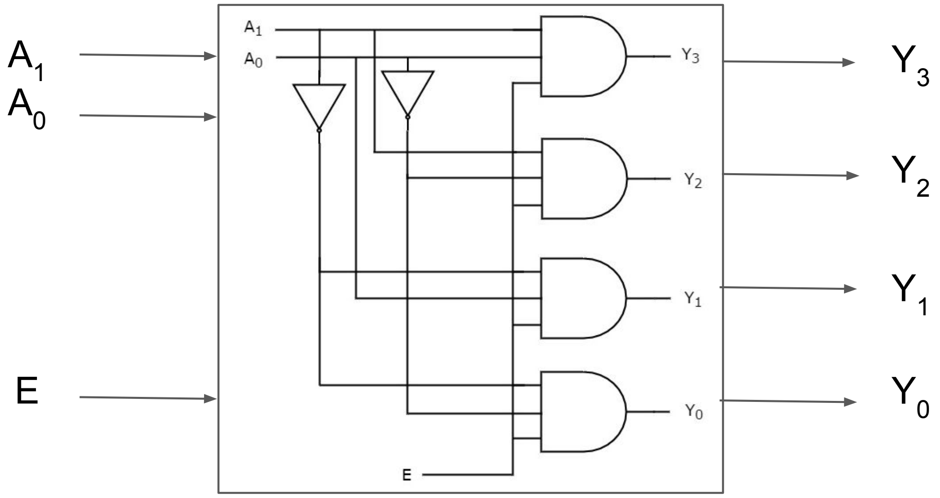
E	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0				1
1	0	1			1	
1	1	0		1		
1	1	1	1			

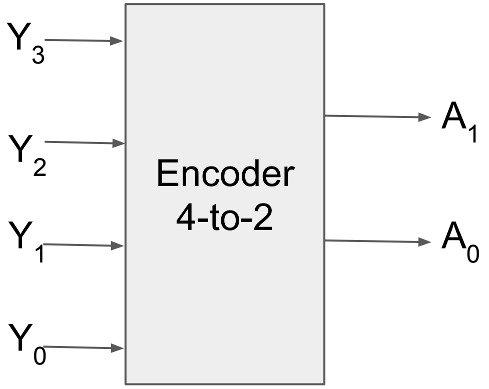


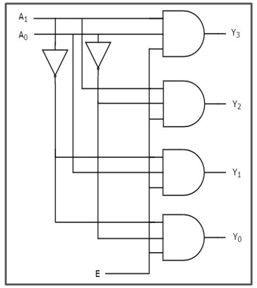
Decoder: 2-to-4, 3-to-8, 4-to-16, 5-to-32

- N+1 Inputs:
  - E: An "enable" line to active the circuit
  - A: Think of it as a binary number
- 2^n Outputs: A output line is set

E	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0				1
1	0	1			1	
1	1	0		1		
1	1	1	1			

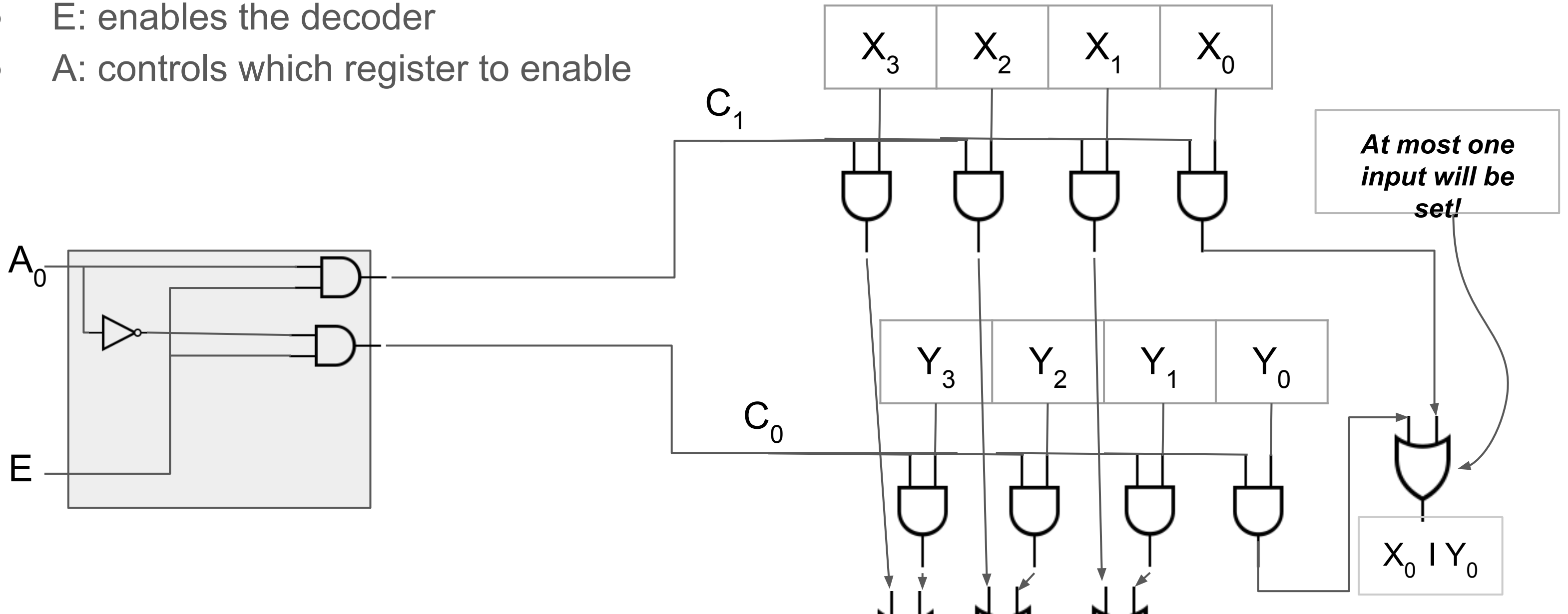






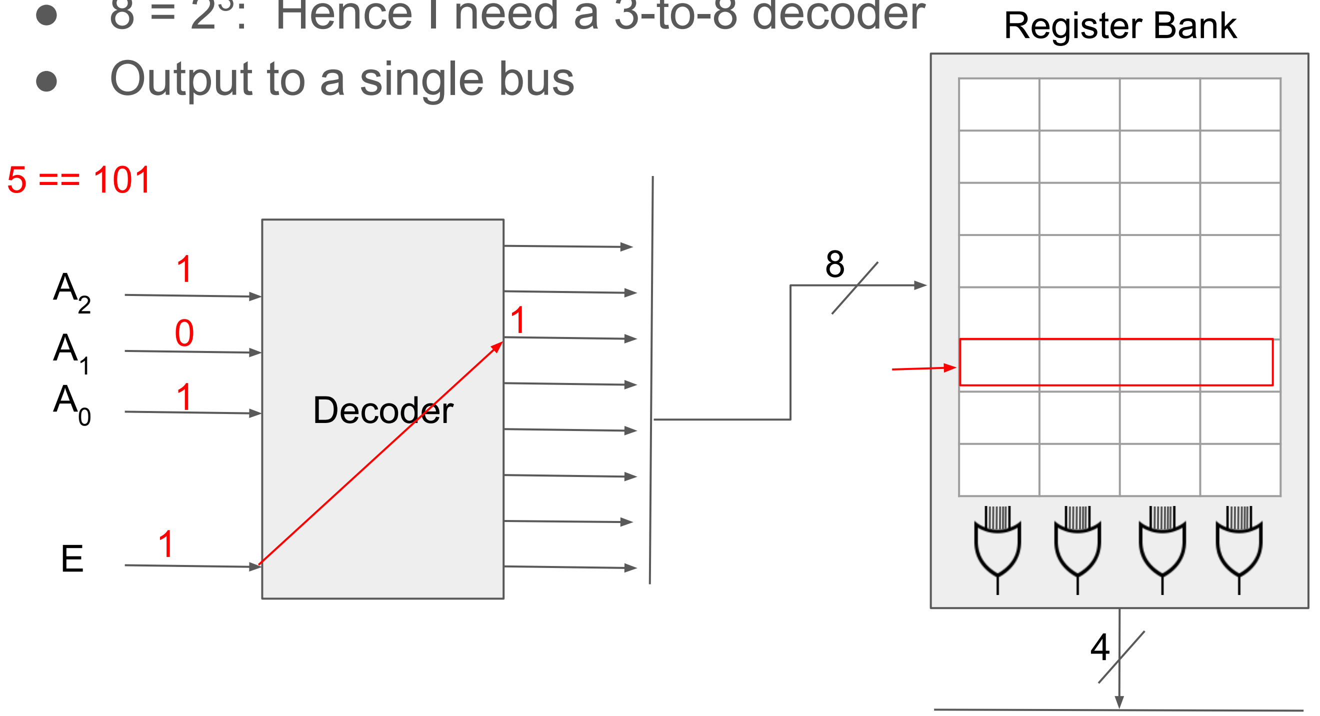
Decoder and Register Selection

- Consider a *two* 4-bit registers
- Use an On-Off switch to enable all the data lines from a register
- Use 1-to-2 Decoder to select which register:
  - E: enables the decoder
  - A: controls which register to enable



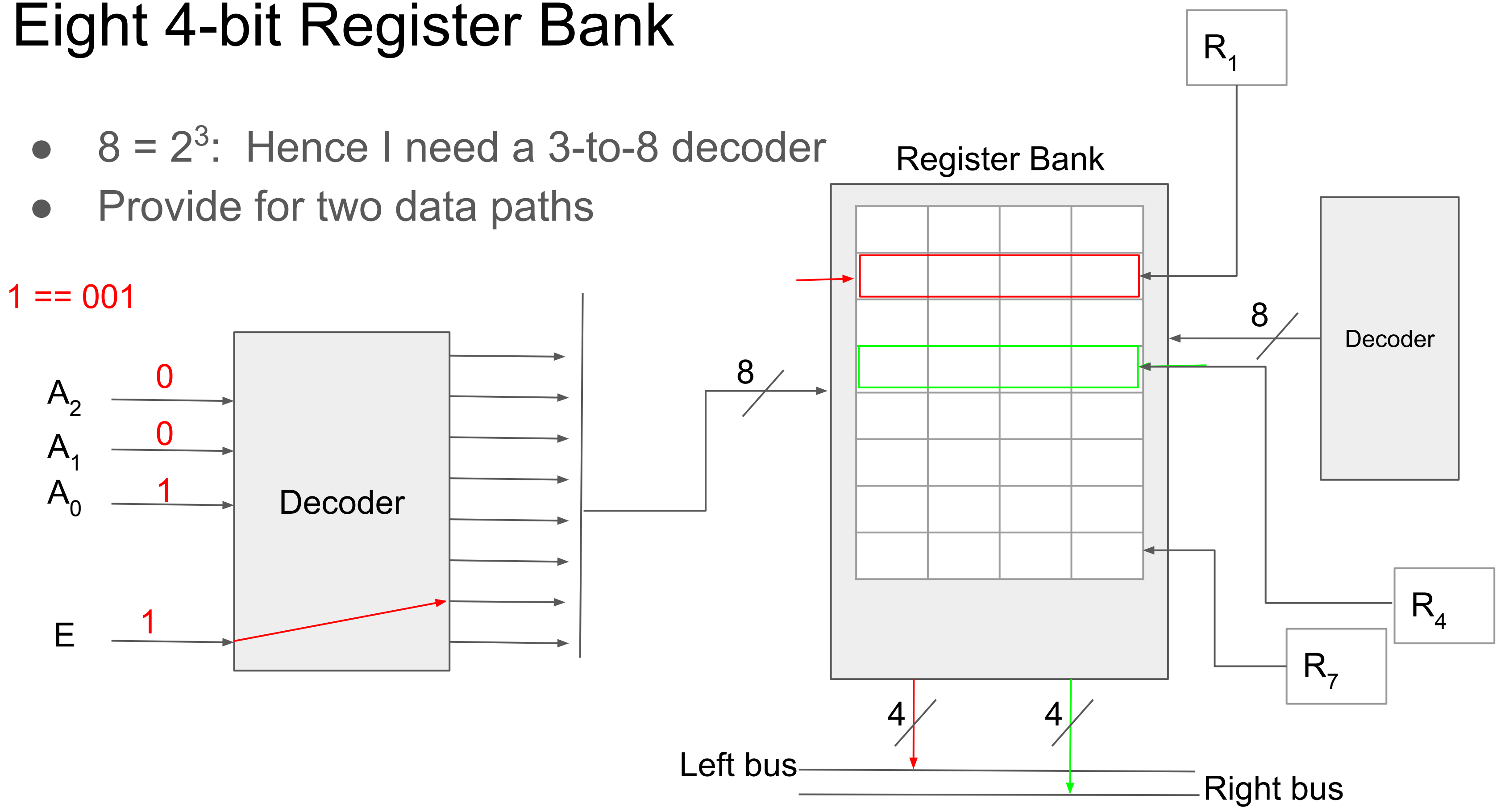
Eight 4-bit Register Bank

- $8 = 2^3$: Hence I need a 3-to-8 decoder
- Output to a single bus



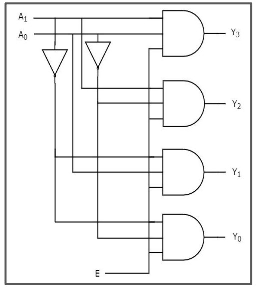
Eight 4-bit Register Bank

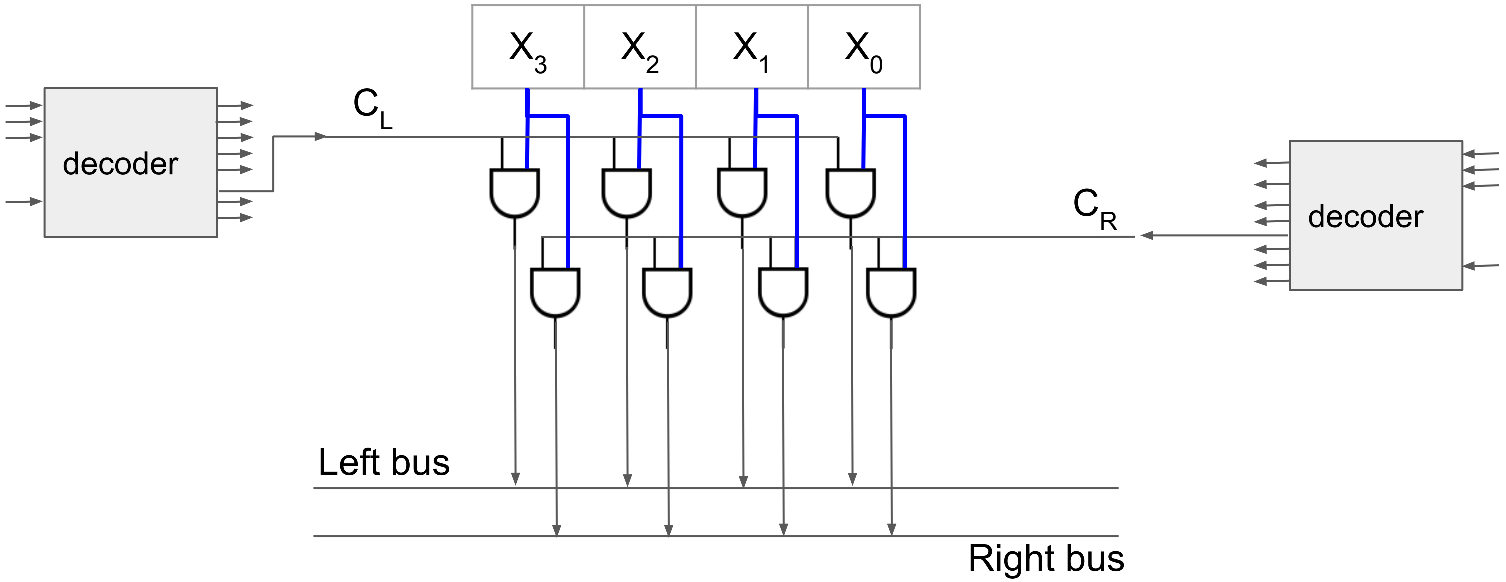
- $8 = 2^3$: Hence I need a 3-to-8 decoder
- Provide for two data paths



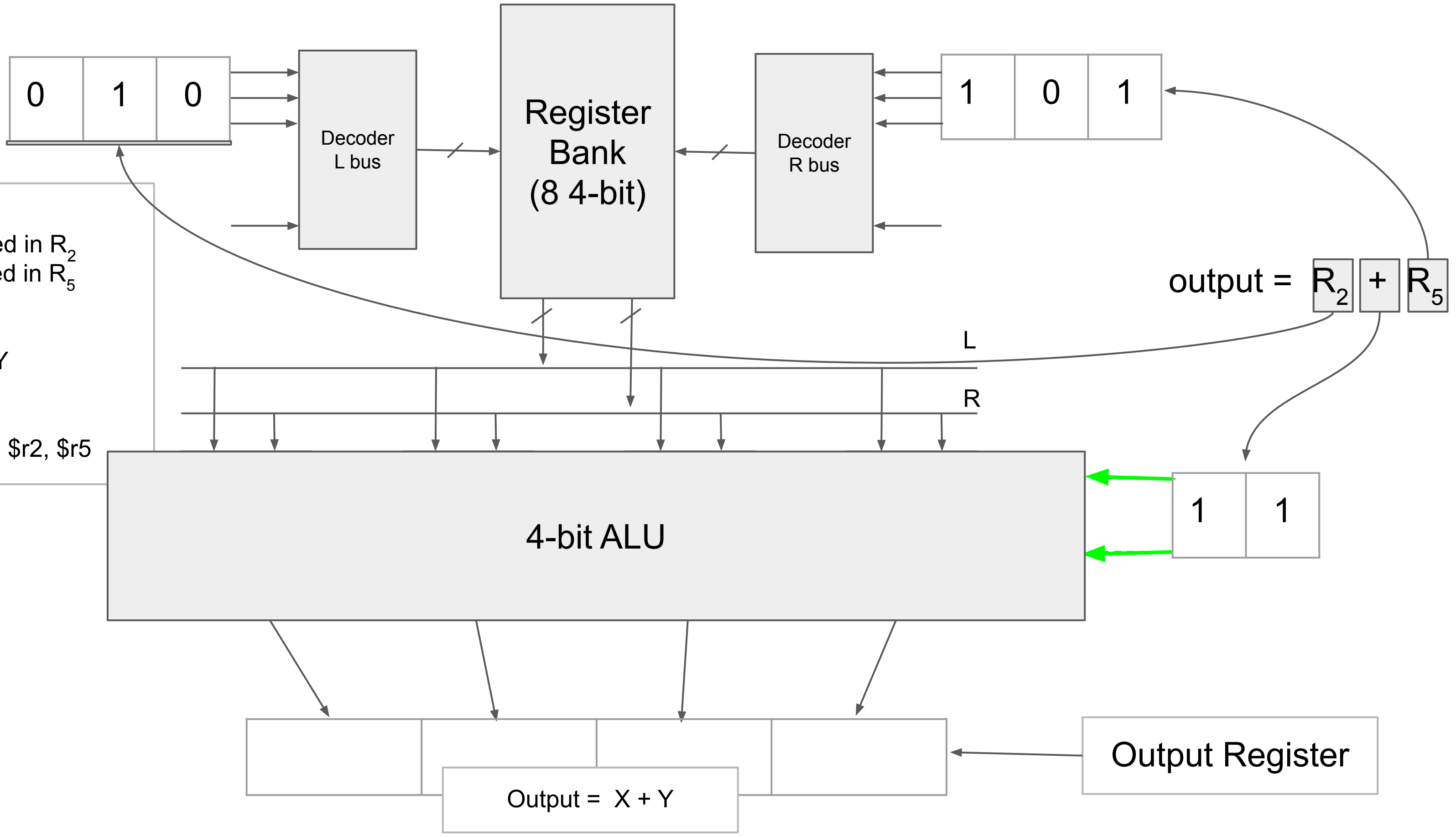
Two Data Paths: revisited

●





CPU: Almost!



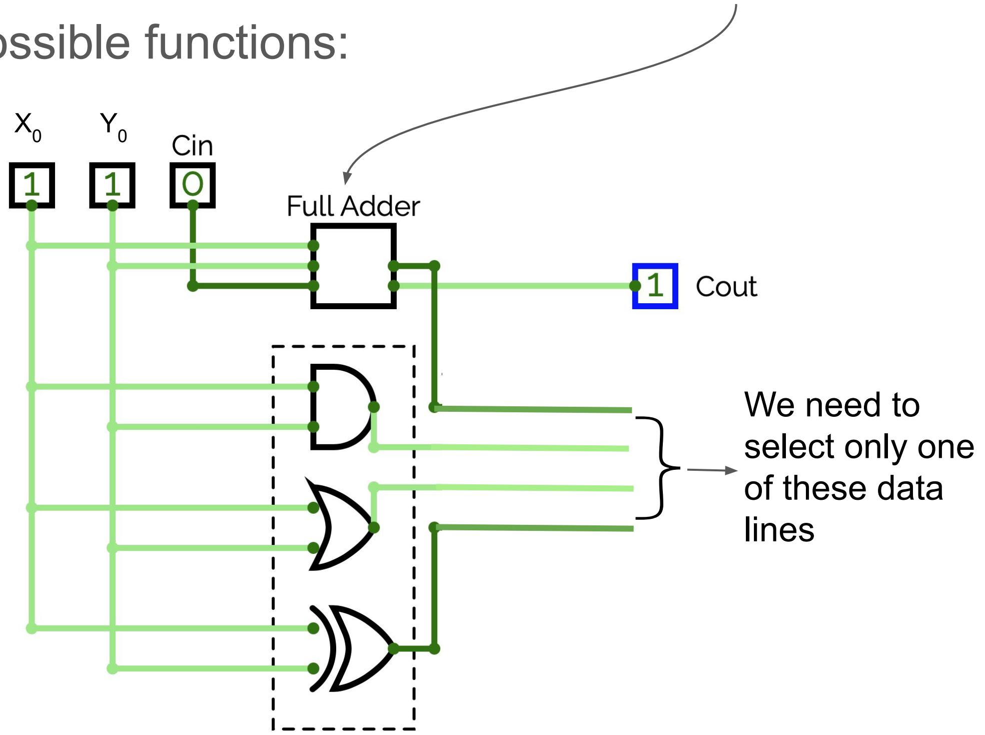
Assume:
 X is stored in R_2
 Y is stored in R_5

Java
 O = X + Y

MIPS:
 ADD \$r1, \$r2, \$r5

Arithmetic and Logical Unit (ALU)

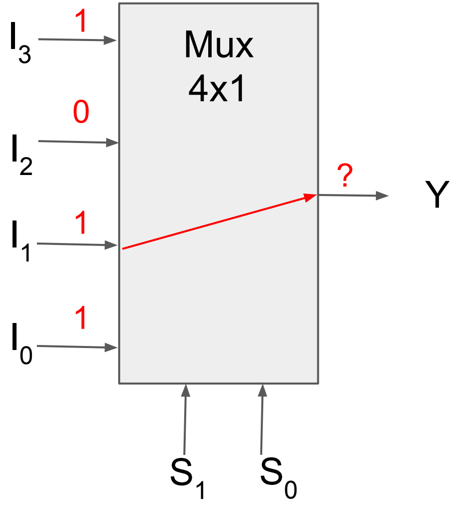
- Let's build an 1-bit ALU with 4 possible functions:
  - $X_0 + X_0$: Binary Addition
  - X_0 & Y_0: Bitwise AND
  - $X_0 \mid Y_0$: Bitwise OR
  - X_0 ^ Y_0: Bitwise XOR


All four functions are computed in parallel
X0
Y0
Cin
1
1
0
Full Adder
1
Cout
We need to select only one of these data lines

Multiplexer (Data Selector)

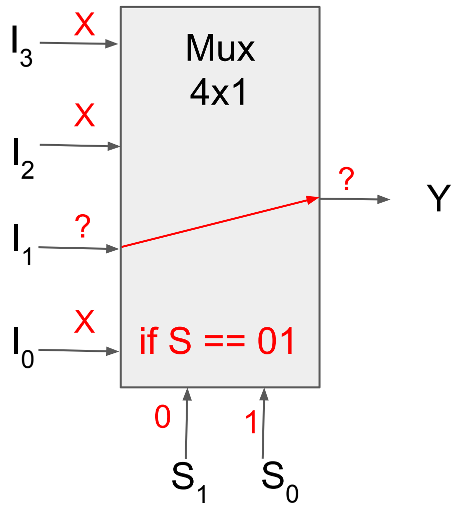
- Inputs:
  - 2^N data input lines
  - N selector lines
- Outputs:
  - 1 dataline

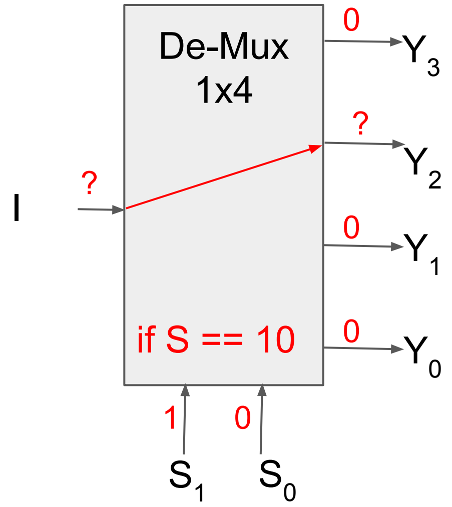
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Multiplexer (Data Selector)

- Inputs:
  - 2^N data input lines
  - N selector lines
- Outputs:
  - 1 dataline

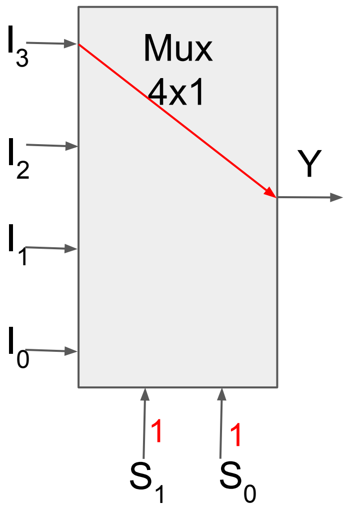


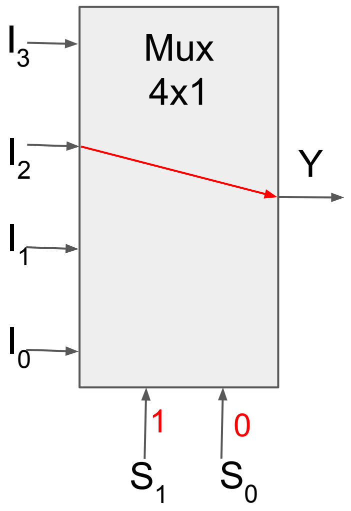


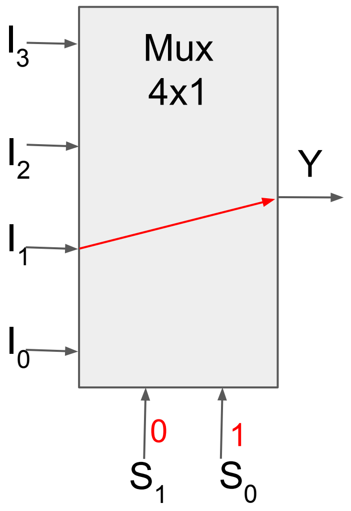
Multiplexer Revisited

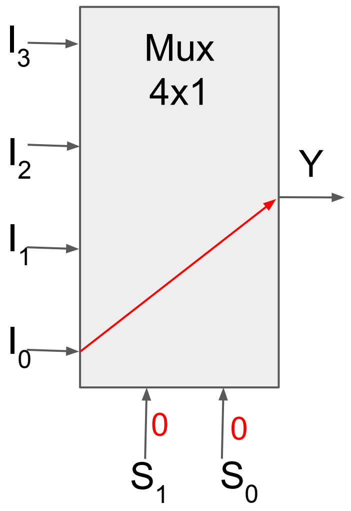
- Inputs:
  - 2^N data input lines
  - N selector lines
- Outputs:
  - 1 dataline

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3





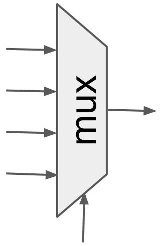


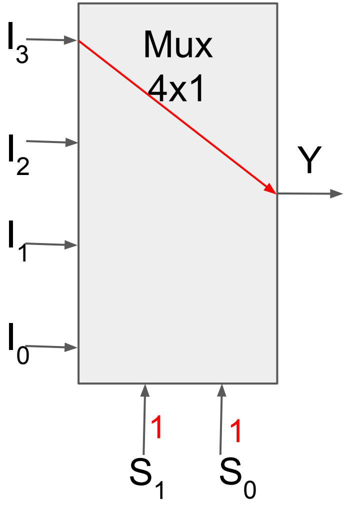


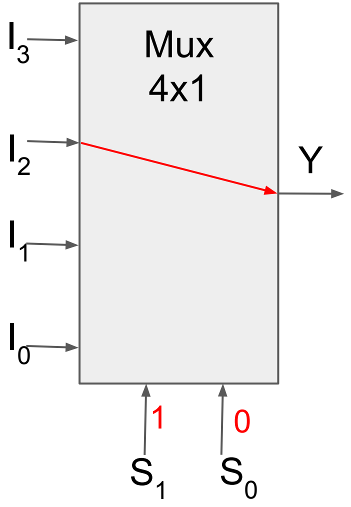
Multiplexer for my ALU

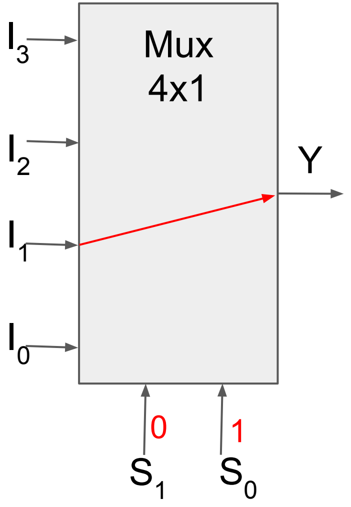
- Inputs:
  - 2^N data input lines
  - N selector lines
- Outputs:
  - 1 dataline

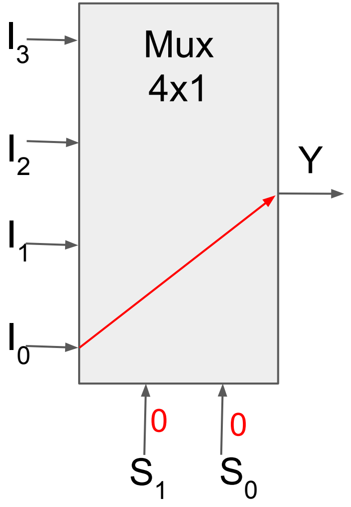
Op	S_1	S_0	Y
^	0	0	I_0
|	0	1	I_1
&	1	0	I_2
+	1	1	I_3







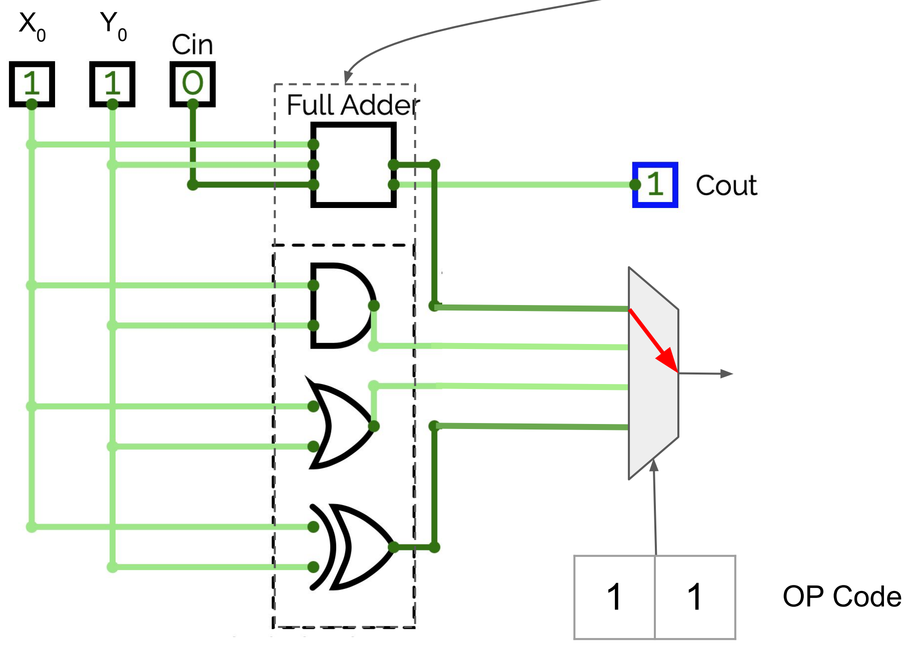




1-bit Arithmetic and Logical Unit (ALU)

- Let's build an 1-bit ALU with 4 possible functions:
  - $X_0 + Y_0$: (3) Binary Addition
  - X_0 & Y_0: (2) Bitwise And
  - $X_0 \mid Y_0$: (1) Bitwise OR
  - X_0 ^ Y_0: (0) Bitwise XOR

All four functions are computed in parallel

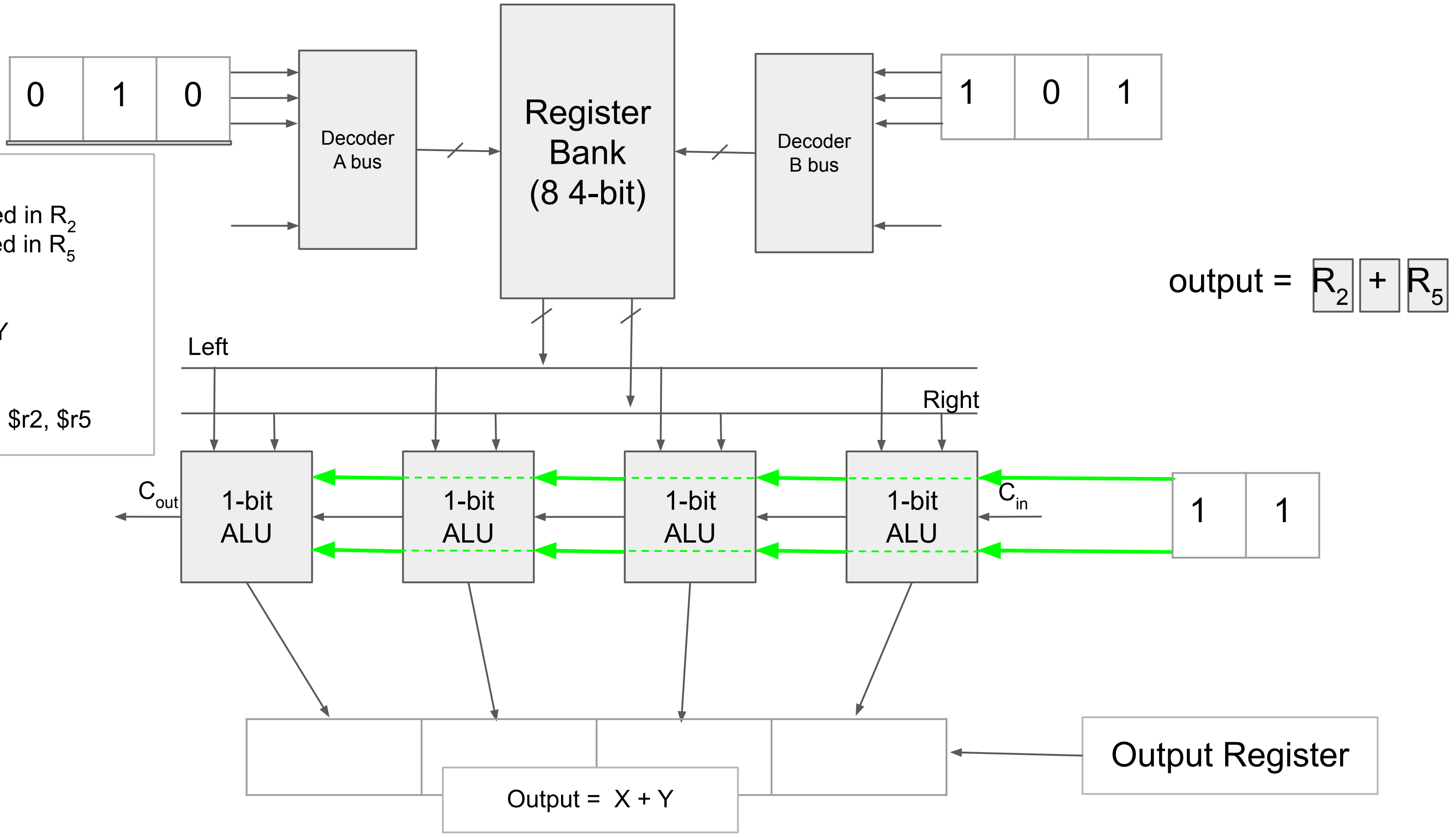


CPU: Almost!

Assume:
X is stored in R_2
Y is stored in R_5

Java
O = X + Y

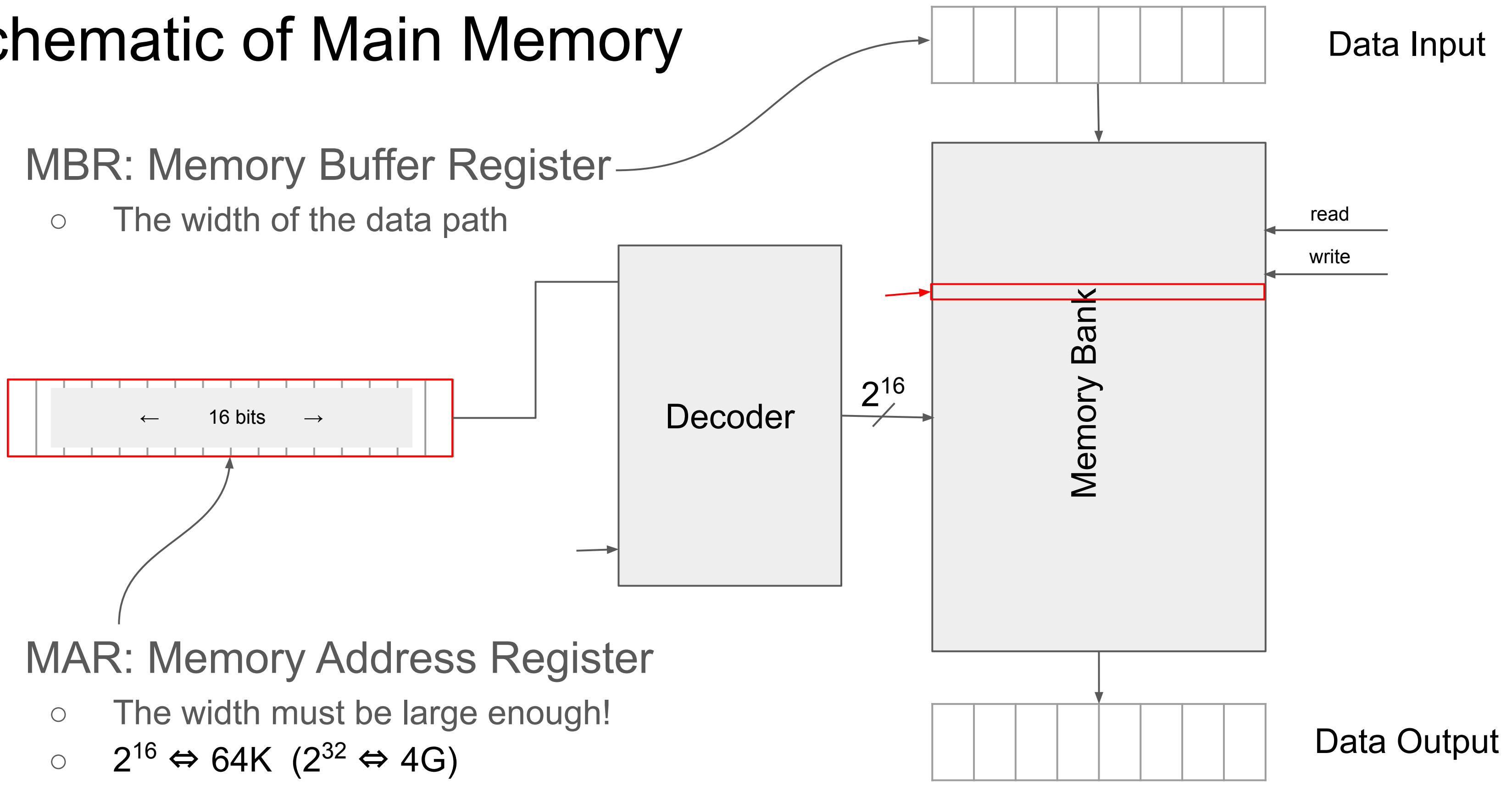
MIPS:
ADD \$r1, \$r2, \$r5



Schematic of Main Memory

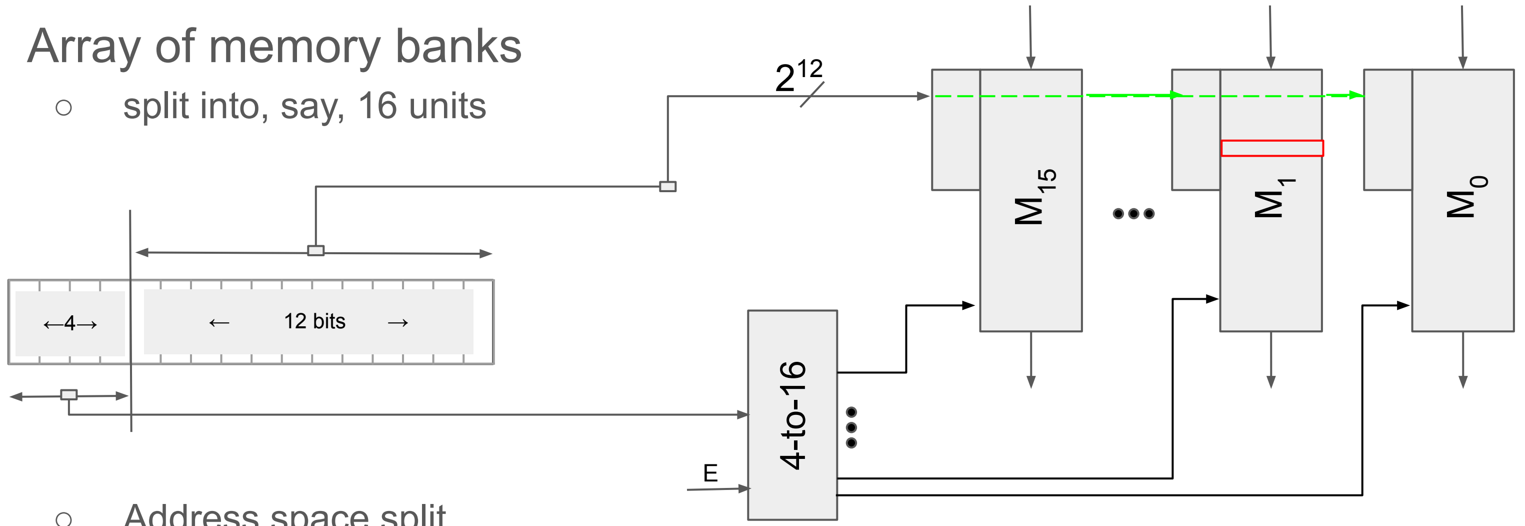
- MBR: Memory Buffer Register
  - The width of the data path

- MAR: Memory Address Register
  - The width must be large enough!
  - $2^{16} \Leftrightarrow$ 64K ($2^{32} \Leftrightarrow$ 4G)



Other Memory Organizations

- Array of memory banks
  - split into, say, 16 units



  - Address space split
    - 4 msb (most significant bits) determine which unit to use: a total of 4 bits : 15..12
    - 12 lsb (least significant bits) provide the internal address: a total of 12 bits : 11..0