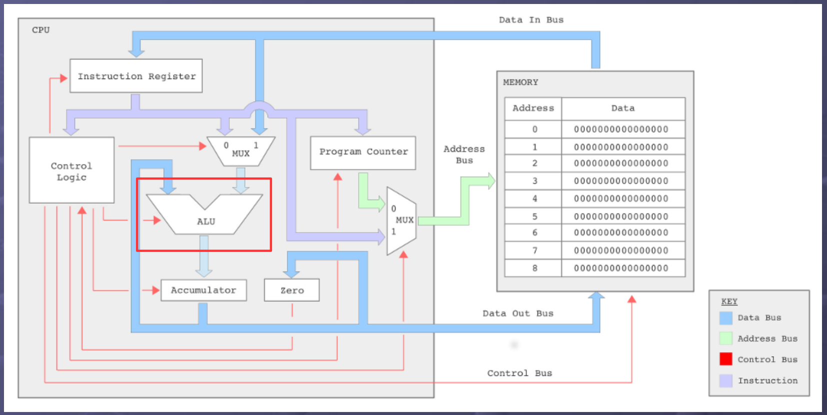
Systems and Devices 1
Lec 3b :
Combinatorial Logic

SimpleCPU_v1a

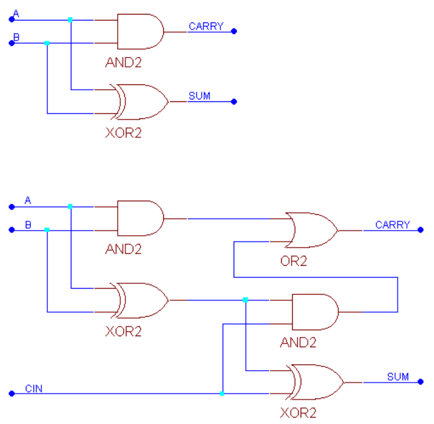


- Block diagram
  - ALU : a core requirement of any computer is to process data i.e. the Arithmetic and Logic unit, the ADDER the heart of any CPU.

Binary addition

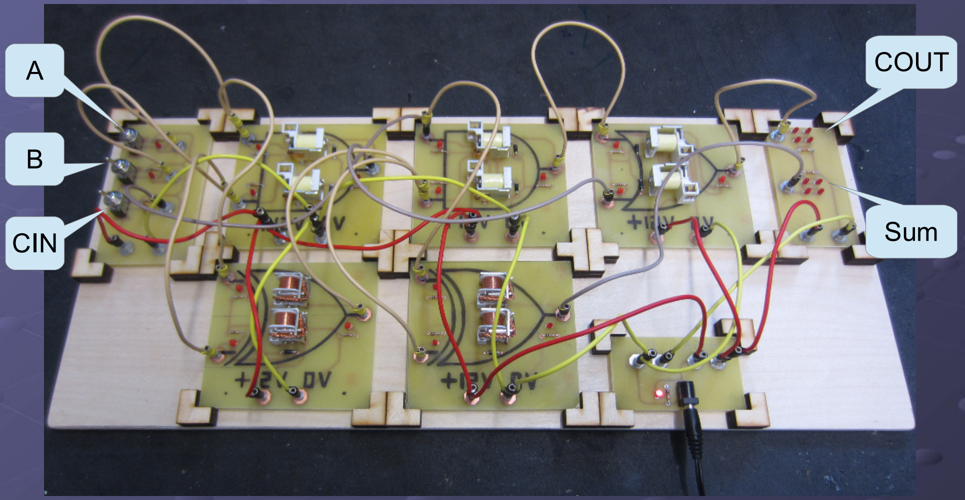
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

A	B	Cin	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



- Half and full adder
  - Basic components can be combined into larger circuits

Demo : relay logic

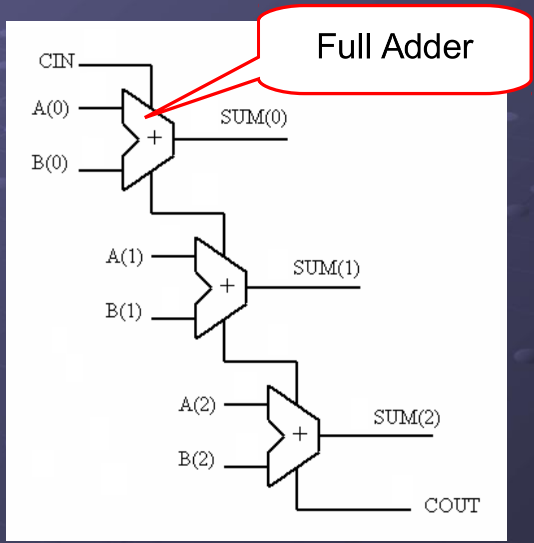


- Full adder

Slide 5

Binary addition

- Ripple adder
  - Replicated full adders
    - Three FA, producing a 3 bit adder
  - LSB carry in (CIN) is set to zero
  - Carry out (COUT) feeds carry signal to next full adder stage

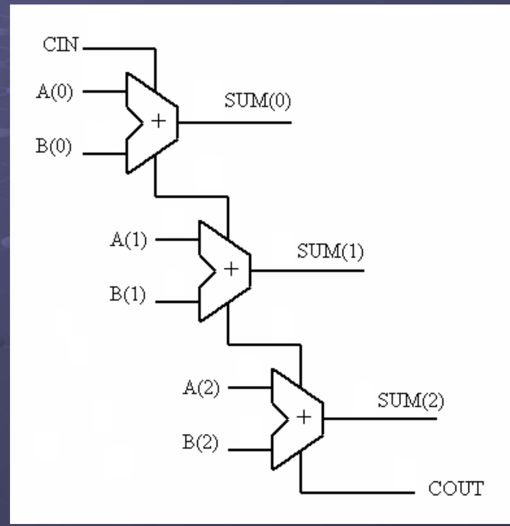


Slide 6

Binary addition

- Ripple adder
  - Add : 7 + 1

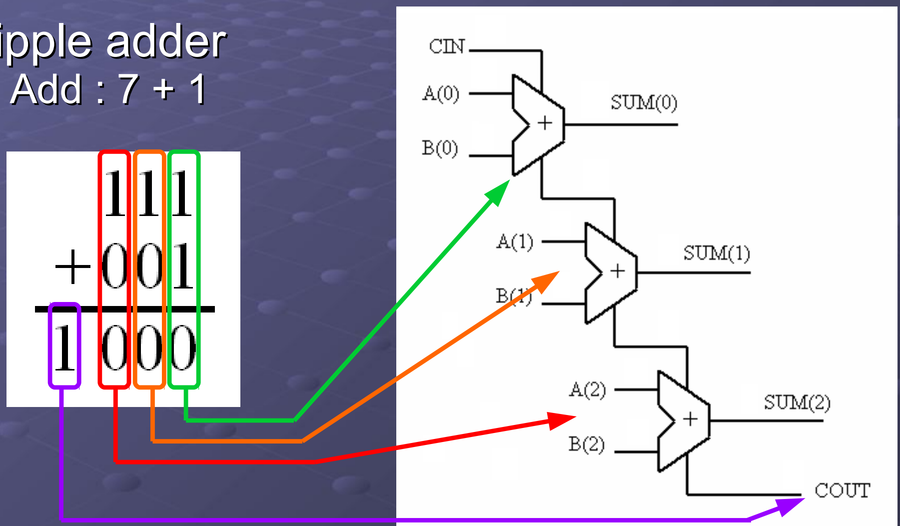
$$\begin{array}{r} 111 \\ +001 \\ \hline 1\,000 \end{array}$$



Slide 7

Binary addition

- Ripple adder
  - Add : 7 + 1

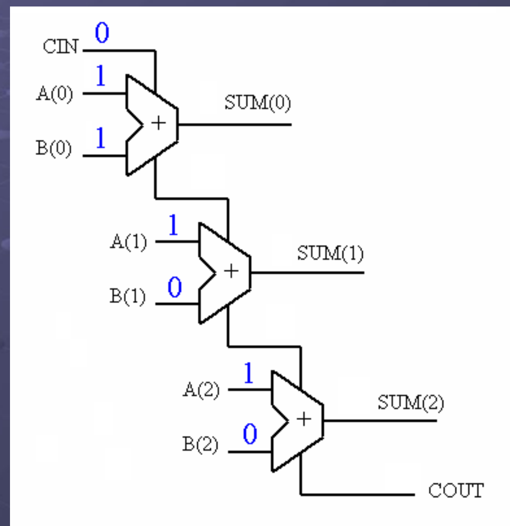


Slide 8

Binary addition

- Ripple adder
  - Add : 7 + 1

$$\begin{array}{r} 111 \\ +001 \\ \hline 1\,000 \end{array}$$

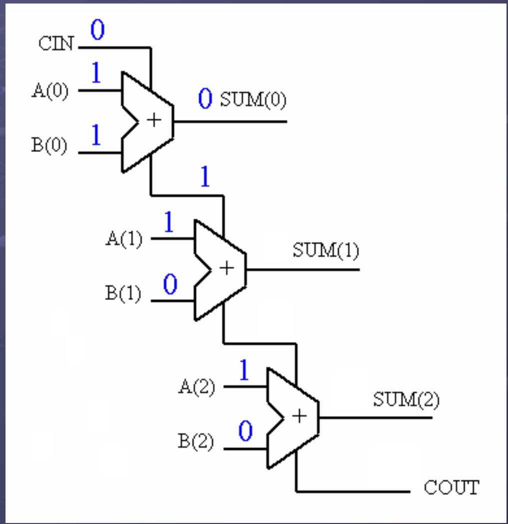


Slide 9

Binary addition

- Ripple adder
  - Add : 7 + 1

111
+001
1 000

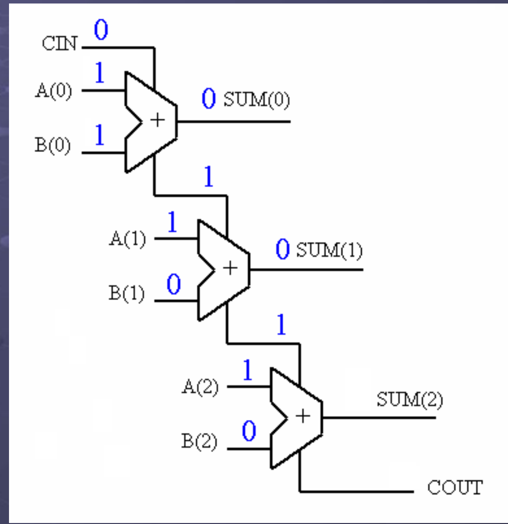


Slide 10

Binary addition

- Ripple adder
  - Add : 7 + 1

111
+001
1 000



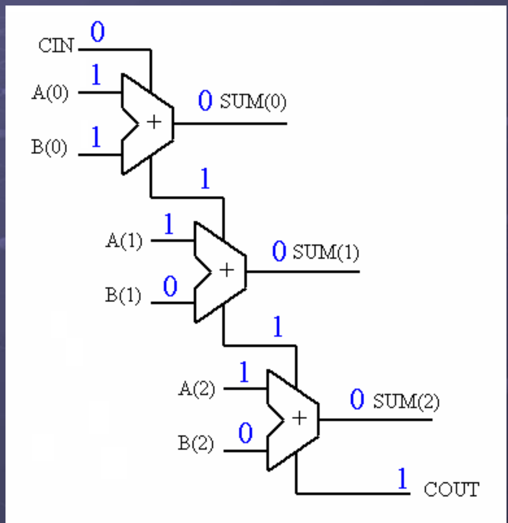
Slide 11

Binary addition

- Ripple adder
  - Add : 7 + 1

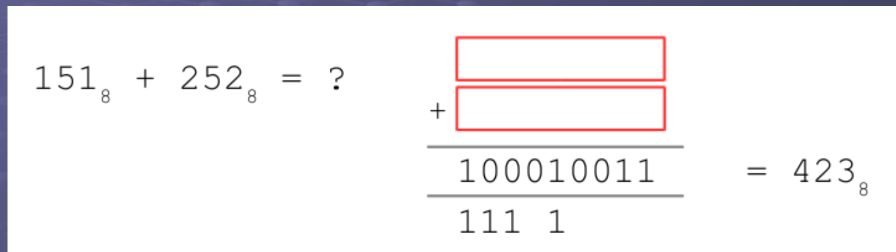
111
+001
1 000

  - Result 8_{10}, 1000_2



Slide 12

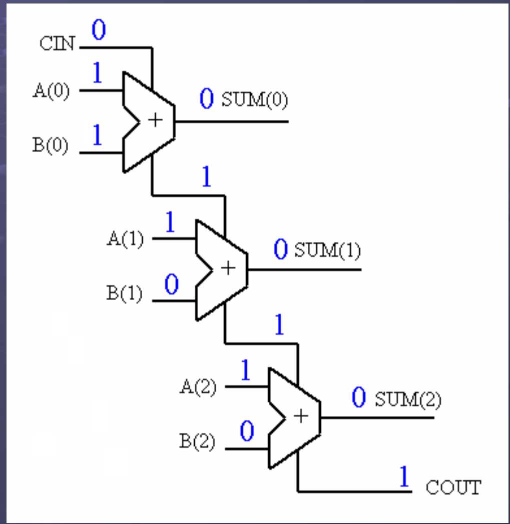
Binary addition



- Quick Quizzz
  - Convert the following values into binary then confirm the result of the binary addition.
  - Is the conversion of the binary result to octal correct?

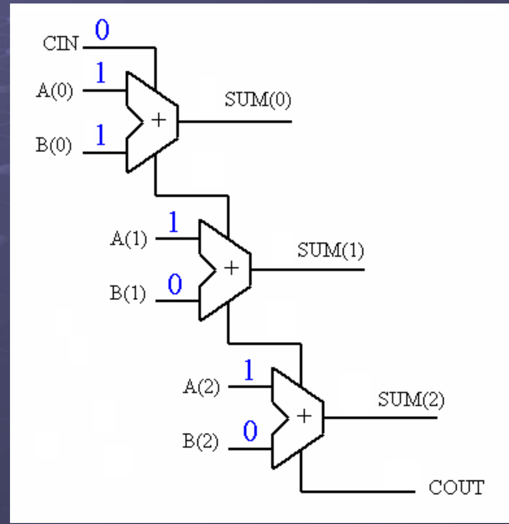
Binary addition

- Ripple adder
  - MSB Carry Out
    - Can be passed to additional full adder stage to allow larger adders to be constructed.
    - Can be used to indicate that the result has exceeded the maximum bit representation i.e. an ***overflow*** has occurred.
  - **Important**, will use these ideas when writing assembly code.



Binary addition

- Ripple adder
  - Remember that hardware is not software i.e. each full adder will operate in parallel.
  - The result will go through a series of states before it settles down to the final value.

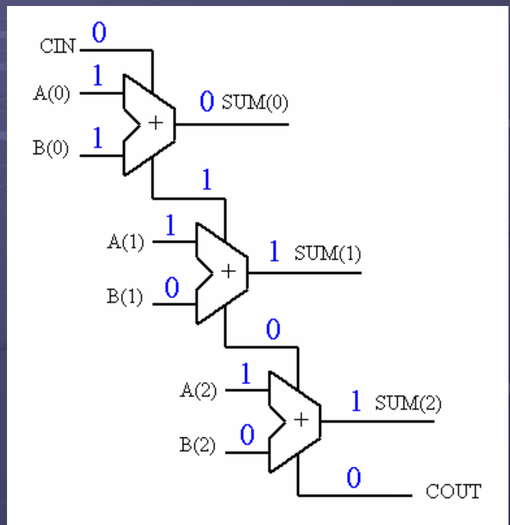


Binary addition

- Ripple adder
  - Add : 7 + 1
  - Step 1

```
  111
+001
0 110
```

  - Result 6_{10},0110_2

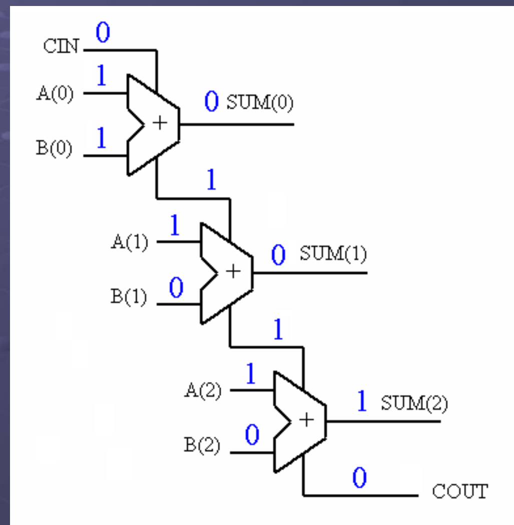


Binary addition

- Ripple adder
  - Add : 7 + 1
  - Step 2

```
  111
+001
0 100
```

  - Result 4_{10},0100_2



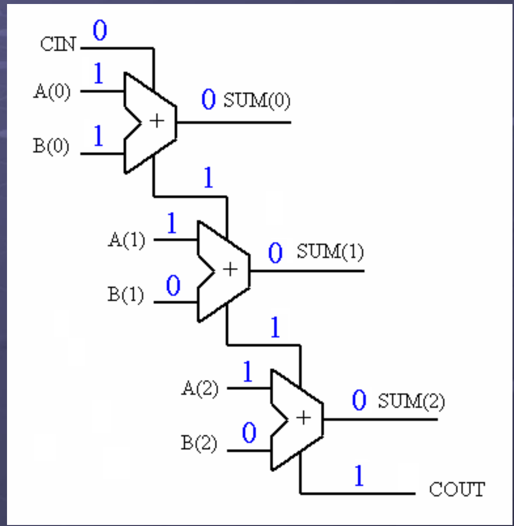
Slide 17

Binary addition

- Ripple adder
  - Add : 7 + 1
  - Step 3

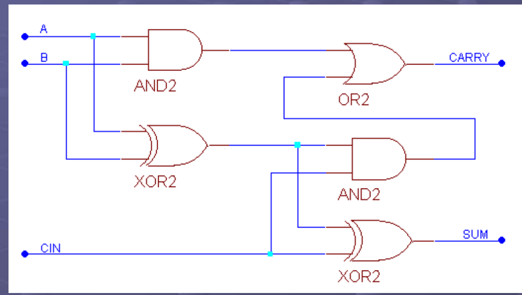
$$\begin{array}{r} 111 \\ +001 \\ \hline 1\,000 \end{array}$$

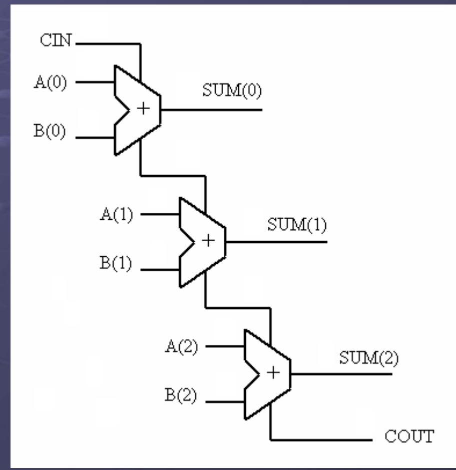
  - Result 8_{10}, 1000_2



Slide 18

Binary addition

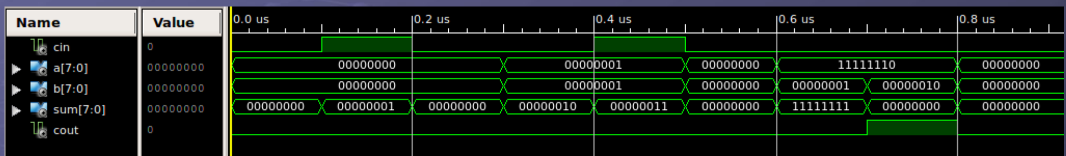


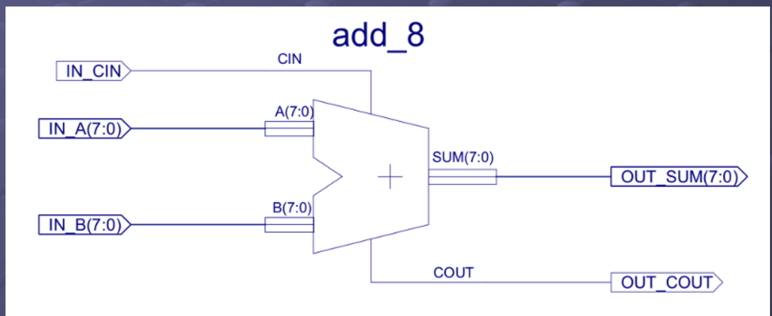


- Quick Quizz
  - If each logic gate takes 10 ns to process a signal, what is the critical path delay of this ripple adder?

Slide 19

Example : adder_8.zip

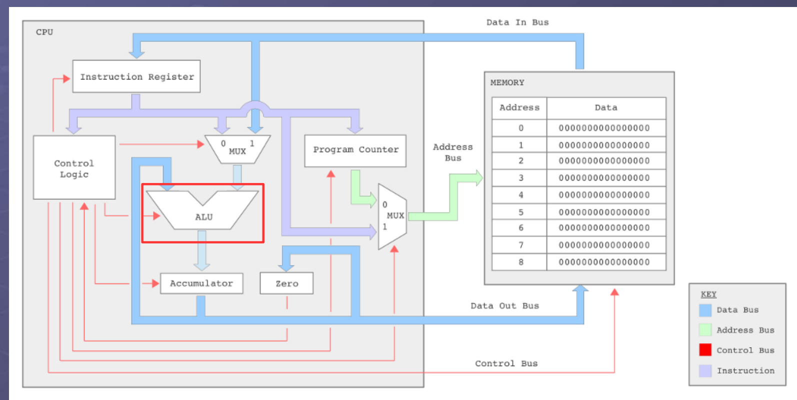




- 8 bit ripple adder

Slide 20

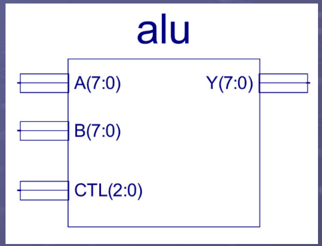
SimpleCPU_v1a



- Block diagram
  - Q : how do we control the ALU's function e.g. pass through, add, subtract and bitwise AND functions, as defined in the instruction set? How do we implement these functions?

Slide 21

ALU



ALU CTL2	ALU CTL1	ALU CTL0	OP
0	0	0	ADD
0	0	1	SUB
0	1	0	AND
0	1	1	NU
1	0	0	PASS
1	0	1	NU
1	1	0	NU
1	1	1	NU

- ALU interface and control (CTL) signals
  - A(7:0) – 8bit input, driven by ACC
  - B(7:0) – 8bit input, driven by Data MUX, IR(7:0) or DIN(7:0)
  - CTL(2:0) – 3bit input, function select, driven by control logic
  - Y(7:0) – 8bit output, result of selected function, Y <= A op B.
- Pass through = multiplexer, addition = ripple adder
- How do we perform subtraction and bitwise AND?

Slide 22

Key skills : working in base 2

```
 10010110     150
-00101100     -44
_________     ___

_________     ___
```

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Slide 23

Key skills : working in base 2

```
 10010110     150
-00101100     -44
_________     ___
        0
_________     ___
```

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Slide 24

Key skills : working in base 2

```
 10010110     150
-00101100     -44
_________     ___
       10
_________     ___
```

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Slide 25

Key skills : working in base 2

10010110	150
-00101100	-44
010	

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Slide 26

Key skills : working in base 2

10010110	150
-00101100	-44
010	

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Slide 27

Key skills : working in base 2

- Borrow case
  - Look to the left until the first 1 is found, this defining a block i.e. 10…0
  - Write a 1 in the result and update block to 01….1
  - Continue subtraction
- Alternatively, another way to think of it
  - Borrow '2' from the left column
    - Same process you would perform in base 10, but rather than borrowing 10 you borrow 2

Slide 28

Key skills : working in base 2

10010110	150
-00101100	-44
010	

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Slide 29

Key skills : working in base 2

```
  10001110        150
 -00101100        -44
 ---------        ---
      1010
 ---------        ---
```

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Slide 30

Key skills : working in base 2

```
  10001110        150
 -00101100        -44
 ---------        ---
     01010
 ---------        ---
```

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Slide 31

Key skills : working in base 2

```
  10010110        150
 -00101100        -44
 ---------        ---
     01010
 ---------        ---
```

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Slide 32

Key skills : working in base 2

```
  01110110        150
 -00101100        -44
 ---------        ---
    101010
 ---------        ---
```

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Key skills : working in base 2

```
 01110110     150
-00101100     -44
---------    ----
  1101010
---------    ----
```

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Key skills : working in base 2

```
 01110110     150
-00101100     -44
---------    ----
 01101010
---------    ----
```

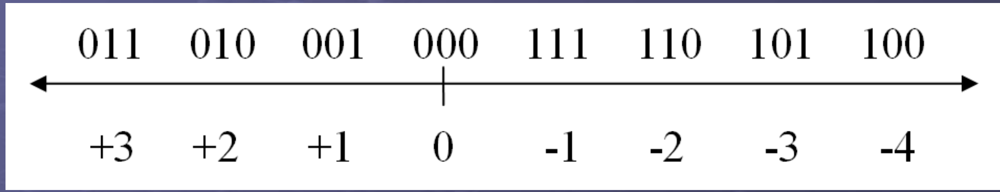
- Subtract two binary numbers : 150 - 44
  - Positive, integer

Key skills : working in base 2

```
 10010110     150
-00101100     -44
---------    ----
 01101010     106
---------    ----
```

- Subtract two binary numbers : 150 - 44
  - Positive, integer

Method of Complements



- Q : How do we represent negative numbers?
  - Using the complement of a number e.g. 2s complement
  - MSB represents the sign: 0 = +num, 1 = –num
    - Max positive sign bit = 0, MSB –1 to LSB = 1
    - Max negative sign bit = 1, MSB –1 to LSB = 0
- To convert to a Two's complement representation
  - Invert each bit position (one's complement) 0→1, 1→0
  - Add 1 (carry ignored)

Slide 37

2s Complement

$1_{10} = 00000001_2$

One's Complement : 11111110
Add one : 11111111

$-1_{10} = 11111111_2$

$100_{10} = 01100100_2$

One's Complement : 10011011
Add one : 10011100

$-100_{10} = 10011100_2$

$200_{10} = 11001000_2$

One's Complement : 00110111
Add one : 00111000

$-200_{10} = 00111000_2$???

- Examples
  - MSB represents the number's sign i.e. a signed number.
  - Maximum value that can be represented is halved compared to an unsigned representation

Slide 38

2s Complement

$-100_{10} = 10011100_2$

One's Complement : 01100011
Add one : 01100100

$100_{10} = 01100100_2$

- To determine the absolute value of a negative binary number
  - Take the Two's complement again

Eight bit signed number	Sixteen bit signed number
$-100_{10} = 10011100_2$	$-100_{10} = 1111111110011100_2$
$100_{10} = 01100100_2$	$100_{10} = 0000000001100100_2$

- Note, when changing the size of a number don't forgot to sign extend.

Slide 39

Binary subtraction

0x2A − 0x6C = ?

[ ]
− [ ]
10111110 = 0xBE

0x2A
− 0x6C
0xBE

42_{10} − [ ] =

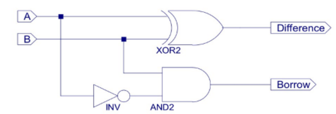
10111110
01000001
01000010 = -66_{10}

- Quick Quizzz
  - Convert the following values into binary then confirm the result of the binary subtraction.
  - Is the conversion of the binary result to hexadecimal correct?

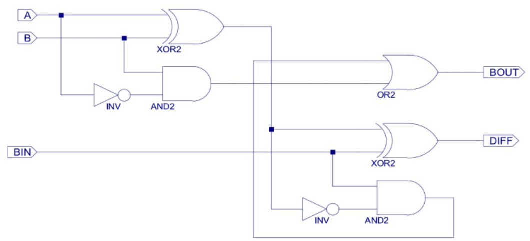
Slide 40

Binary subtraction

A	B	BORROW	DIFFERENCE
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0



A	B	BIN	BOUT	DIFF
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



- We could implement the subtraction operation using half and full subtractors, but …

Slide 41

Key skills : working in base 2

010010110	150
-000101100	-44

- Using the Two's complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 42

Key skills : working in base 2

010010110	150
+111010100	-44

- Using the Two's complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 43

Key skills : working in base 2

010010110	150
+111010100	-44
0	

- Using the Two's complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 44

Key skills : working in base 2

010010110	150
+111010100	-44
10	

- Using the Two's complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 45

Key skills : working in base 2

```
 010010110     150
+111010100     -44
----------    ----
       010
----------    ----
      1
```

- Using the Two's complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 46

Key skills : working in base 2

```
 010010110     150
+111010100     -44
----------    ----
      1010
----------    ----
      1
```

- Using the Two's complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 47

Key skills : working in base 2

```
 010010110     150
+111010100     -44
----------    ----
     01010
----------    ----
    1 1
```

- Using the Two's complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 48

Key skills : working in base 2

```
 010010110     150
+111010100     -44
----------    ----
    101010
----------    ----
    1 1
```

- Using the Two's complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 49

Key skills : working in base 2

```
  010010110      150
 +111010100      -44
 ----------      ---
    1101010
 ----------      ---
     1  1
```

- Using the Two’s complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 50

Key skills : working in base 2

```
  010010110      150
 +111010100      -44
 ----------      ---
   01101010
 ----------      ---
  1   1  1
```

- Using the Two’s complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 51

Key skills : working in base 2

```
  010010110      150
 +111010100      -44
 ----------      ---
  001101010      106
 ----------      ---
 11   1  1
```

- Using the Two’s complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

Slide 52

Key skills : working in base 2

```
  010010110      150
 +111010100      -44
 ----------      ---
  001101010      106
 ----------      ---
 11   1  1
```

- Using the Two’s complement representation simplifies binary subtraction i.e. can be performed using addition
  - A - B = A + (-B)

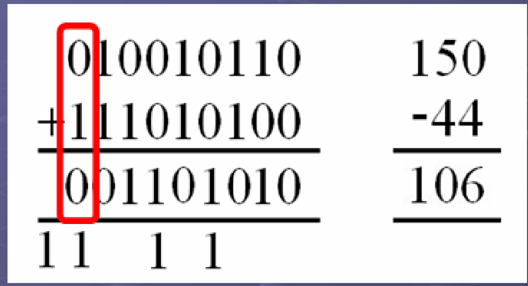
Slide 53

Addition of negative numbers

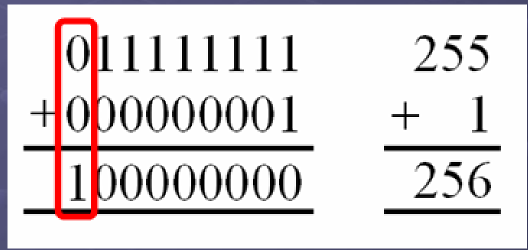
- When using Two's complement representation the carry bit can no longer be used to indicate an overflow.
  - Oveflow - number (result) can not be represented by the maximum number of bits within a memory location or register i.e. need more bits, can not be stored.
- Overflow is determined by these rules
  - If operand sign bits are equal then result sign bit must equal operand sign bit
    - E.g. (A + B) or (-A + -B) magnitude always bigger
  - If operand sign bit are not equal then overflow can not occur
    - E.g. (A – B) or (-A + B) magnitude always smaller

Slide 54

Addition of negative numbers



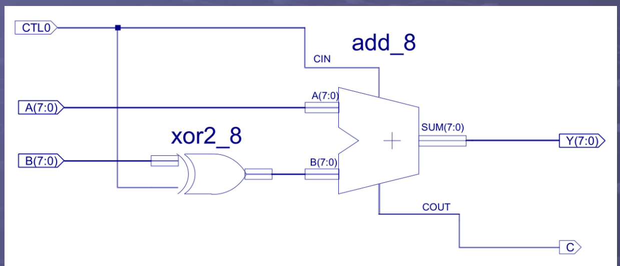
- Different sign bits
  - Overflow can not occur

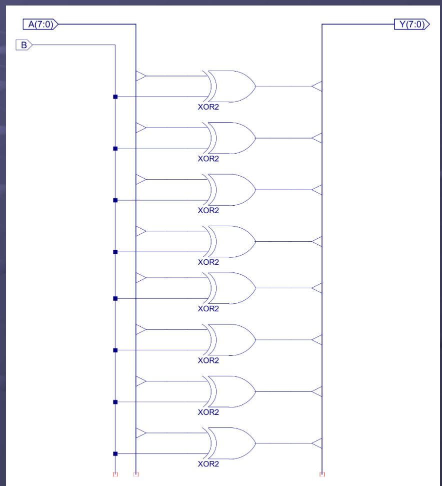


- Matching sign bits
  - Overflow may occur

Slide 55

Adder / Subtractor unit

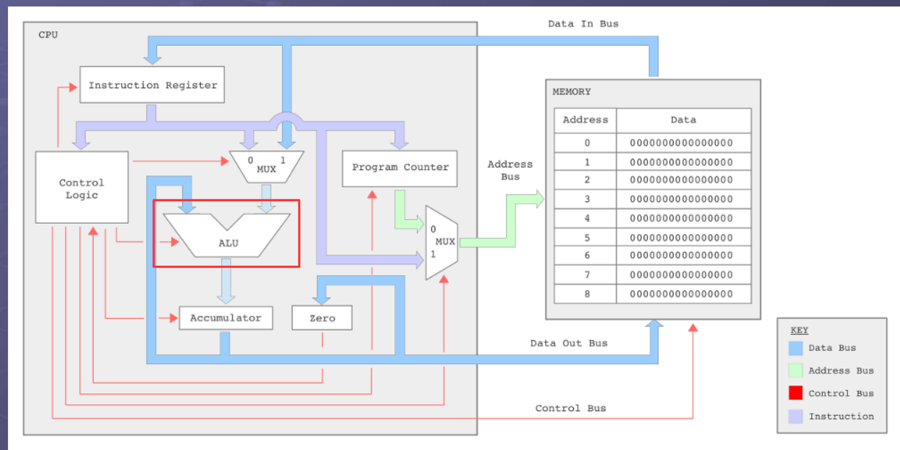




- How do we perform 2's complement in hardware?
  - ADD_SUB_8
    - Ripple adder
    - XOR array

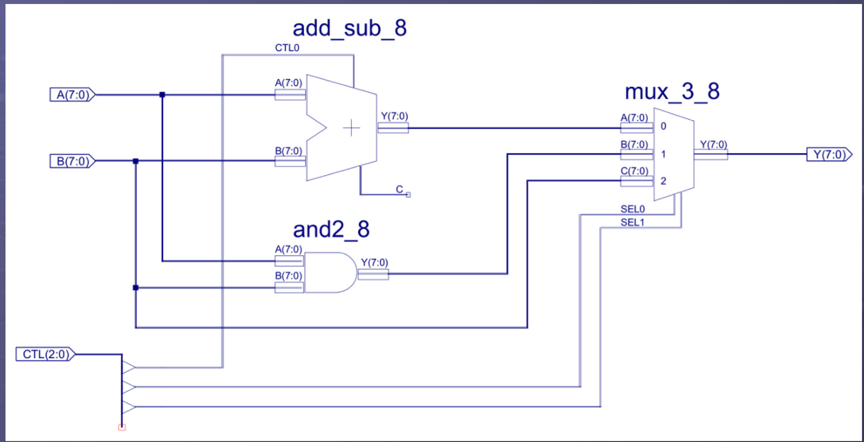
Slide 56

SimpleCPU_v1a



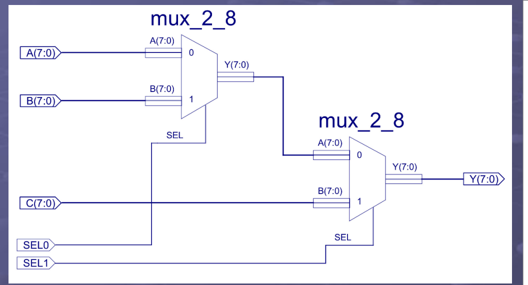
- Block diagram
  - Q : what else is inside the ALU?

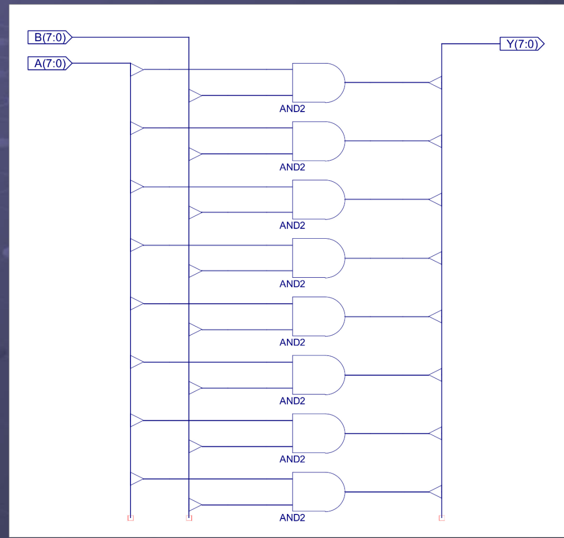
ALU



- A : not a lot, a simple ALU for a simple instruction set
  - Q : what will happen to the CPD if we have more, "complex" (multiply, divide, square root) instructions?

ALU

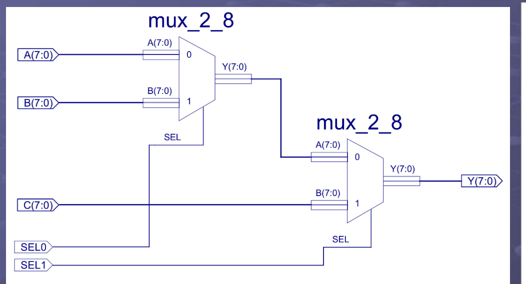


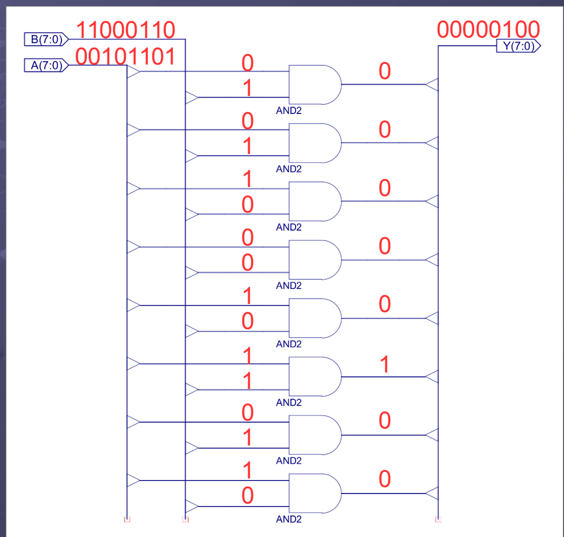


- Bitwise AND

A	00101101	45
B	11000110	198 or -58
	00000100	4

ALU





- Bitwise AND

A	00101101	45
B	11000110	198 or -58
	00000100	4

Summary

- Key concepts :
  - Binary arithmetic
    - Half subtractor, Full subtractor and Ripple subtractor
    - Representing negative numbers - 2's complement
    - Subtraction using addition of a negative number
      - Detecting overflows when using signed data
  - Data processing : Arithmetic & Logic Unit (ALU)
    - Pass, Add, Subtract and Bitwise AND
    - More complex functions can be construction in software using these basic binary operators e.g. multiply, divide etc.