LOGBOOK FOR COMPUTER ARCHITURE

Neil Abraham

UNIVERSITY OF WEST LONDON Student Number:21299930

Contents

Week 1 – Chapter 1

Exercises:

1. Explain the idea that 'all computers can do exactly the same things'.

This idea can be explained by defining the word "computer". The definition of computer is "an electronic device which is capable of receiving information (data) in a particular form and of performing a sequence of operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a result in the form of information or signals." (Oxford dictionaries 2016)

2. Name one of the characteristic of natural languages that prevents them from being used as programming languages.

Natural languages e.g. English can be ambiguous.

3. Write a statement in a natural language and offer two different interpretations of that statement.

Statement: Harry met his friends at Starbucks.

Two different interpretations of statement above:

Harry's friends met Harry at Starbucks.

Harry met his best friends near Starbucks.

4. Two computers, A and B are identical except for the fact that A has a subtract instruction and B does not. Both have added instructions. Both have instructions that can take a value and produce the negative of that value. Which computer is able to solve more problems, A or B? Prove your results.

Both A and B are computers so both computers can do same things. However, computer A has more number of instructions than computer B, so computer A is able to solve more problems.

Computer A	Computer B
subtraction and addition	Only addition

5. Identify one advantage of programming in a higher-level language compared to a lower-level language. Identify one disadvantage.

Advantage	Disadvantage
High level programming languages are independent to the machine on which the programs will execute.	High level programming languages are easier for humans to understand but high level programs cannot be understood by a Computer hence low level programming language is needed.

6. List the level of transformation and name an example for each level.

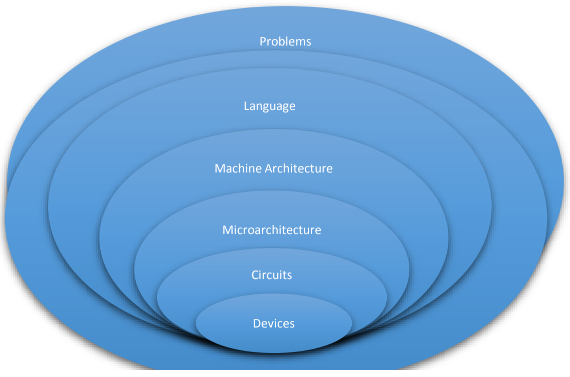


Figure 1 List of transformation

Week 2 – Chapter 2

Exercises:

2.1 Given n bits, how many distinct combinations of the n bits exists?

2^n possible combinations for a collection of n bits.

2.2 There are 26 characters in the alphabet we use for writing English. What is the least number of bits needed to give each character a unique bit pattern? How many bits would we need to distinguish between upper and lowercase versions of all 26 characters?

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Minimum bits needed is $n = 5$ $which\ would\ give\ 32\ characters$ for just upper case. We need $52\ characters\ for\ upper\ and\ lower\ case\ characters.\ so\ the\ n = 6\ since\ 52\ is\ less\ than\ 64.$

2.3.a. Assume that there are about 400 students in your class. If every student is to be assigned a unique bit pattern, what is the minimum number of bits required to do this?

$9\ bits\ is\ required\ as\ there\ are\ 400\ students\ in\ the\ class.$

2.3.b. How many more students can be admitted to the class without requiring additional bits for each student's unique bit pattern?

$512 - 400 = 112$ more students can be admitted to the class without requiring additional bits.

2.4 Given n bits, how many unsigned integers can be represented with the n bits? What is the range of these integers?

$With\ n\ bits, 2^n\ intergers\ ranging\ from\ 0\ to\ 2^n - 1.$

2.5 Using 5 bits to represent each number, write the representations of 7 and -7 in 1's complement, signed magnitude, and 2's complement integers.

N-bits	Signed magnitude	1's Complement	2's Complement
7	00111	00111	00111
-7	10111	11000	11001

2.6 Write the 6-bit 2's complement representation of —32.

$32 = 00100000\ so - 32\ would\ be\ 11100000.$

2.7 Create a table showing the decimal values of all 4-bit 2's complement numbers.

Decimal	Binary
-9	0110
-8	100
-7	1001
-6	1010
-5	1011
-4	1100
-3	1100
-2	1010
-1	1000
0	0000
1	0001
2	0010
3	0011
4	0100

5	0101
6	0110
7	0111
8	0111
9	0111

2.8. a. What is the largest positive number one can represent in an 8-bit 2's complement code? Write your result in binary and decimal.

$8\ bit\ 2's\ complement\ code\ is\ 0111111 = 27 - 1 = 127$

2.8.b. What is the greatest magnitude negative number one can represent in an 8-bit 2's complement code? Write your result in binary and decimal.

$Signed\ 8\ bit\ 2's\ complement\ code\ is\ 10000000 = -(27) = -128$

2.9 Convert the following 2's complement binary numbers to decimal.

a. 1010 = 10

b. 01011010 = 90

c. 1 1 1 1 1 1 1 0 = 1

d. 0011100111010011 = 14803

2.10 Convert these decimal numbers to 8-bit 2's complement binary numbers.

a. 102 = 01100110

b. 64 = 01000000

c. 33 = 0100001

d. -128 = 1000000

127 = 01111111

Week 3

Exercises:

3.1 Add the following bit patterns. Leave your results in binary form.

a. 1011 + 0001 = 1100

b. 0000 + 1010 = 1010

c. 1100 + 0011 = 1111

d. 0101 + 0110 = 1011

e. 1111 + 0001 = 0000

3.2 When is the output of an AND operation equal to 1?

AND operation has an output equal to 1 when there is no change.

3.3 Fill in the following truth table for a one-bit AND operation

x	y	X AND y

0	0	0
0	1	0
1	0	0
1	1	1

X AND y = 0001.

3.4 Compute the following. Write your results in binary.

a. 0 1 0 1 0 1 1 1 AND 11010111 = 01010111

b. 1 0 1 AND 110 = 100

c. 1 1 1 0 0 0 0 0 AND 10110100 = 10100000

d. 0 0 0 1 1 1 1 1 AND 10110100 = 00010100

e. (0 0 1 1 AND 0 1 1 0) AND 1 1 0 1 = 0000

x	y	z	(V) x AND y	V AND z
0	1	0	0	0
0	1	0	1	0
1	0	1	0	0
1	1	1	0	0

f. 0 0 1 1 AND (0 1 1 0 AND 1 1 0 1) = 0000

x	y	z	(V) x AND y	V and z
0	1	0	0	0
1	1	0	1	0
1	0	1	0	0
0	1	1	0	0

3.5 When is the output of an OR operation equal to 1?

The output of an OR operation equal to 1 when there is change.

3.6 Fill in the following truth table for a one-bit OR operation.

x	y	X OR y
0	0	0
0	1	1
1	0	1
1	1	1

3.7 Compute the following:

a. 01010111 OR 11010111 = 11010111

b. 101 OR 110 = 111

c. 11100000 OR 10110100 = 11110100

d. 00011111 OR 10110100 = 10111111

e. (0101 OR 1 1 0 0) OR 1101 = 1101

x	y	z	(V) x OR y	V OR z
0	1	1	1	1
1	1	1	1	1

0	0	0	0	0
1	0	1	1	1

f. 0101 OR (1100 OR 1101) = 1101

x	y	z	(V) y OR z	V OR z
0	1	1	1	1
1	1	1	1	1
0	0	0	0	0
1	0	1	1	1

3.8 Compute the following:

a. NOT (1 0 1 1) OR NOT (1100) = 0111

X	y	NOT x	NOT y	NOT x OR NOT y
1	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	0	1	1

b. NOT (1 0 0 0 AND (1100 OR 0101)) = 0111

x	y	z	V (y OR z)	A (x AND V)	NOT A
1	1	0	1	1	0
0	1	1	1	0	1
0	0	0	0	0	1
0	0	1	1	0	1

C. NOT (NOT (1 1 0 1)) = 1101

d. (0110 OR 0 0 0 0) AND 1111 = 0110

x	y	z	U (x OR y)	U AND z
0	0	1	0	0
1	0	1	1	1
1	0	1	1	1
0	0	1	0	0

3.9 Convert the following unsigned binary numbers to hexadecimal.

a. 1101 0001 1010 1111 = D 1 A F

b. 001 1111 = 1 F

c. 1 = 1

d. 1110 1101 1011 0010 = E D B 2

3.10 Convert the following hexadecimal numbers to binary.

a. x10 = 0000 0001 0000

b. x801 = 0000 1000 0000 0001

c. xF731 = 0000 1111 0111 0011 0001

d. x0F1E2D = 0000 0000 1111 0001 1110 0010 1101

e. xBCAD = 0000 1011 1100 1010 1101

Week 4 - Part A

Week 4 – Recap Exercises:

1. Conversion

(a) Convert the binary number 00100101 to decimal.

$$0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 37$$

(b) Convert the decimal number 21 to an 8-bit unsigned binary representation.

$$21_{10} \rightarrow \frac{21}{2} \rightarrow\ 10^{remander(r)=\,1}, 5^{r=0}, 2^{r=1}, 1^{r=0}, 0^{r=1} = 00010101$$

(c) Convert the 8-bit 2's complement binary number 11011010 to decimal.

$$11011010 = 218$$

(d) Convert the decimal number -111 to an 8-bit 2's complement binary representation.

$$111\ \rightarrow \frac{111}{2} \rightarrow 55^{r=1}, 27^{r=1}, 13^{r=1}, 6^{r=1}, 3^{r=0}, 1^{r=1}, 0^{r=1} \rightarrow 11110110\ so - 111 = 10010001$$

(e) Convert the 8-bit unsigned binary number 11010010 to hexadecimal.

D 2

(f) Convert the unsigned hexadecimal number 29 to unsigned 8-bit binary

29 $\Rightarrow$ 0010 1001

2. Binary Arithmetic and Logical Operations. Let A = 00100101 and B = 11111011 be 2's complement integers. Compute the following. Assume a fixed width of 8 bits (i.e., your answers must be 8 bits).

(a) A +B: 0010 0000

$$\begin{array}{r} 0010\,0101 \\ \underline{+\,1111\,1011} \\ 0010\,0000 \end{array}$$

(b) A OR B: 1111 1011

A	B	A OR B	A AND B	NOT B
0	1	1	0	0
0	1	1	0	0
1	1	1	1	0
0	1	1	0	0
1	1	1	0	0
0	0	0	0	1
0	1	1	0	0
1	1	1	1	0

(c) A AND B: 0010 0001

(d) B – A: 1101 0101

$$\begin{array}{l} 1111\ 1011 \\ \underline{-\ 0010\ 0101} \\ 1111\ 1011 \\ \underline{+\ 1101\ 1010} \\ 1\ 1101\ 01001 \end{array}$$

(e) A – B: 0010 1001

(f) A +(NOT B) + 1: 0 0100 1010

$$\begin{array}{r} 00100101 \\ +\ 00000100 \\ \underline{+\qquad\quad 1} \\ 001001010 \end{array}$$

3. Logical Operations. Complete the following truth tables.

(a)

A	$\neg A$	**A OR** $\neg A$	**A AND** $\neg A$
0	1	1	0
0	1	1	0

(b)

A	B	C	(A OR B) AND C	(A AND C) OR (B AND C)	A AND C	B AND C	A OR B
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	1
0	1	1	1	1	0	1	1
1	0	0	0	1	1	0	1
1	0	1	0	1	1	0	1
1	1	0	0	0	0	0	1
1	1	1	1	1	1	1	1

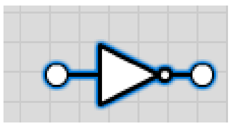
(c)

A	B	$\neg A\ AND\ \neg B$	$\neg(A\ OR\ B)$	$\neg A$	$\neg B$	A OR B
0	0	1	1	1	1	0
0	1	0	0	1	0	1
1	0	0	0	0	1	1
1	1	0	0	0	0	1

Week 4 - Logic Gates

NOT gate:

The NOT gate also known as inverter. It is an electronic circuit symbol that produces opposite binary values of input at its output. This can be shown in the example below.



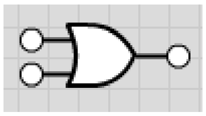
The truth table for NOT gate:

If the input is 1 the output of the circuit will have inverted of input which is 0.

Input	Output
1	0

OR gate:

The OR gate gives a high input of 1 if one or more inputs are on at 1. The electronic circuit symbol is shown below.

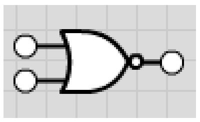


The truth table for OR gate:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

NOR gate:

The NOR gate is the inverted OR gate. It is also an OR gate followed by a Not gate. The outputs of a NOR gate are 0 if any of the inputs are 1. The electronic symbol for the NOR gate is shown below.

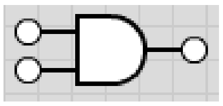


The truth table for NOR gate:

A	B	$\neg(A\ OR\ B)$
0	0	1
0	1	0
1	0	0
1	1	0

AND gate:

The AND gate give a high output of 1 if all its input are 1. The electronic symbol for an AND gate is shown below.

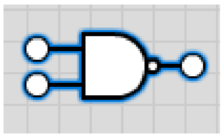


The truth table for AND gate:

A	B	$A\ AND\ B$
0	0	0
0	1	0
1	0	0
1	1	1

NAND gate:

The NAND gate is an AND gate followed by an NOT gate. The outputs of NAND gates are 1 when any of the inputs are 0. The electronic symbol of NAND gate is shown below.

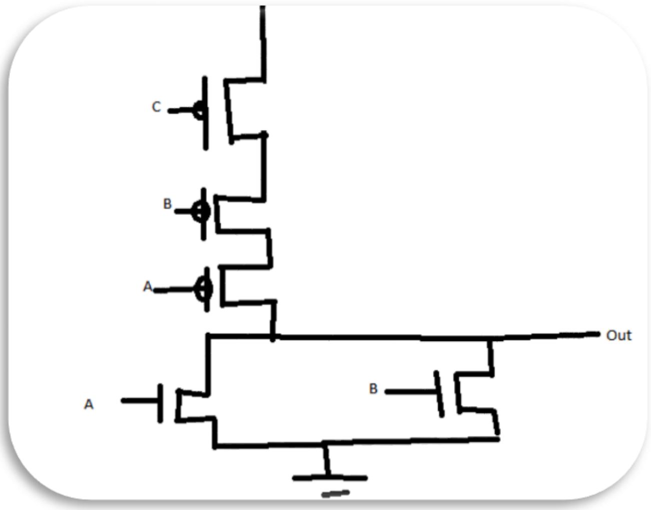


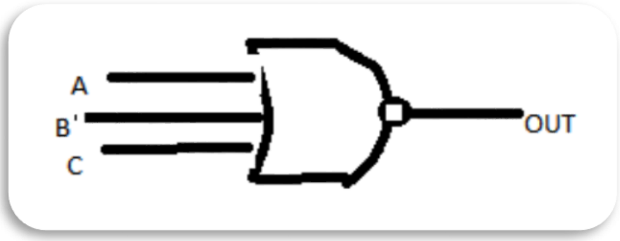
The truth table for NAND gate:

A	B	$A\ NAND\ B$
0	0	1
0	1	1
1	0	1
1	1	0

Implement a 3-input NOR gate with CMOS:

The image on the left shows a 3-input NOR gate with CMOS. The image below shows 3-input NOR gate with electrical symbol.





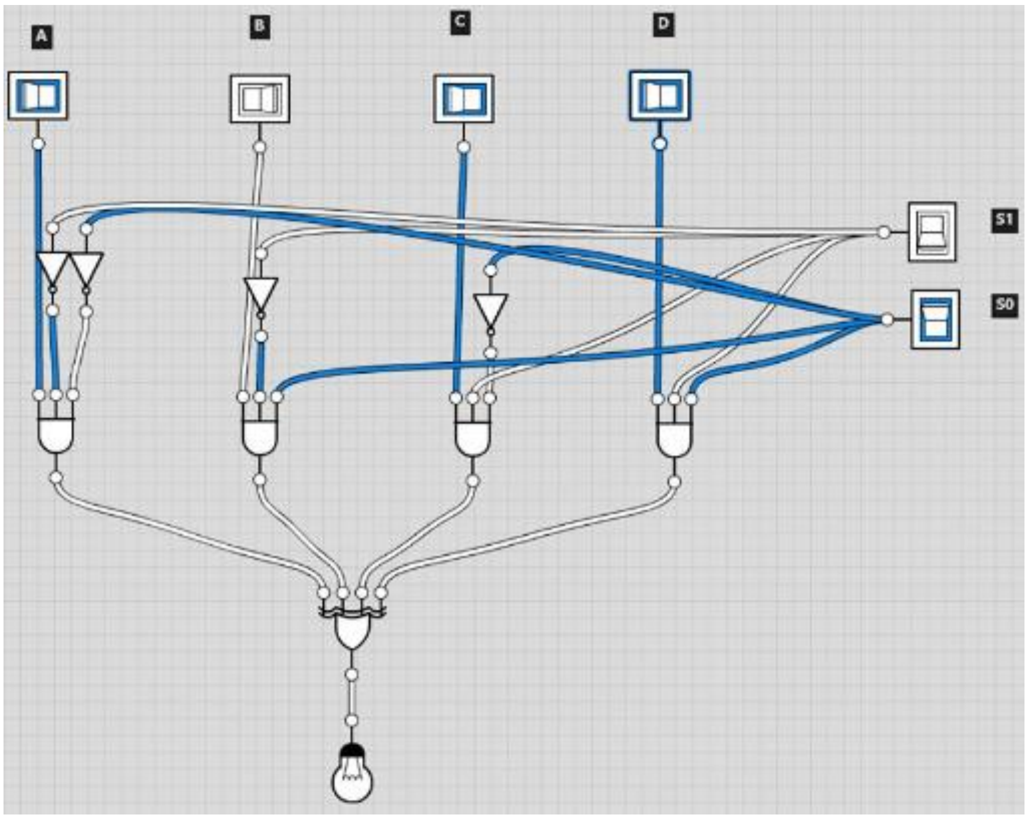
The truth table for 3-input NOR gate is shown below:

A	B	C	OUT $\neg(A+B+C)$
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0
1	1	0	0
1	0	1	0
1	1	1	0

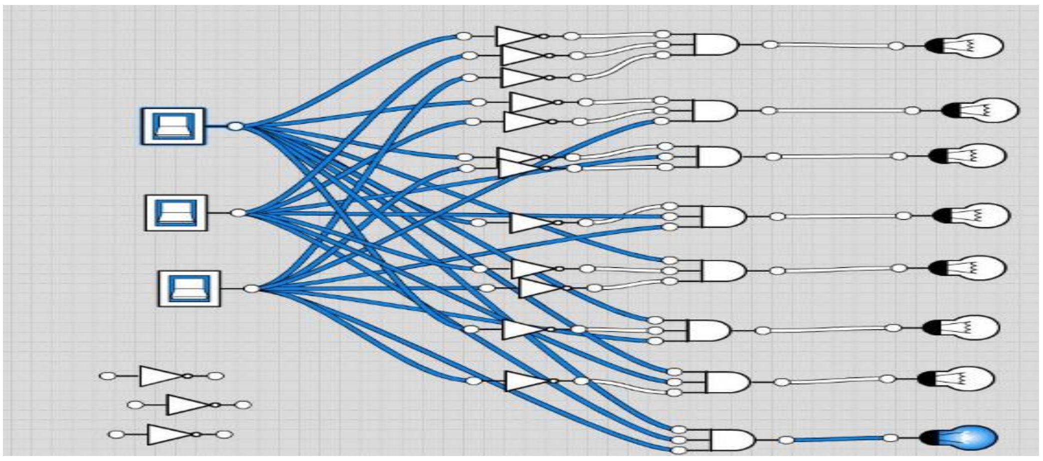
Week 4 – Part B

(a) A two – input decoder:

Property of a decoder is that one of decoder's output is 1 and the rest of the outputs are 0. Decoders have n inputs and 2^n outputs. It is useful as it can explain how to interpret a bit pattern. The image below shows a two input decoder.

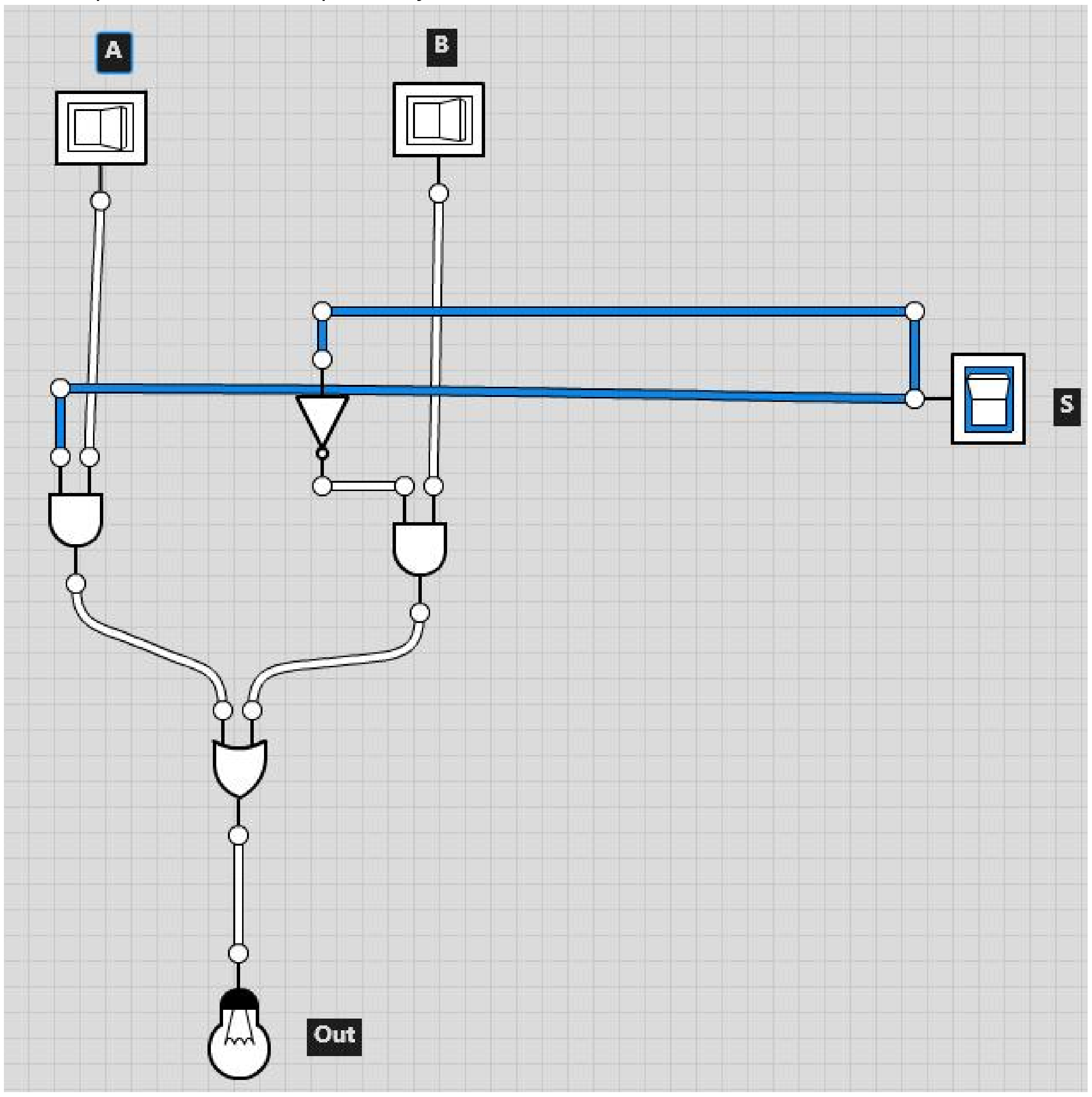


(b) A three – input decoder:

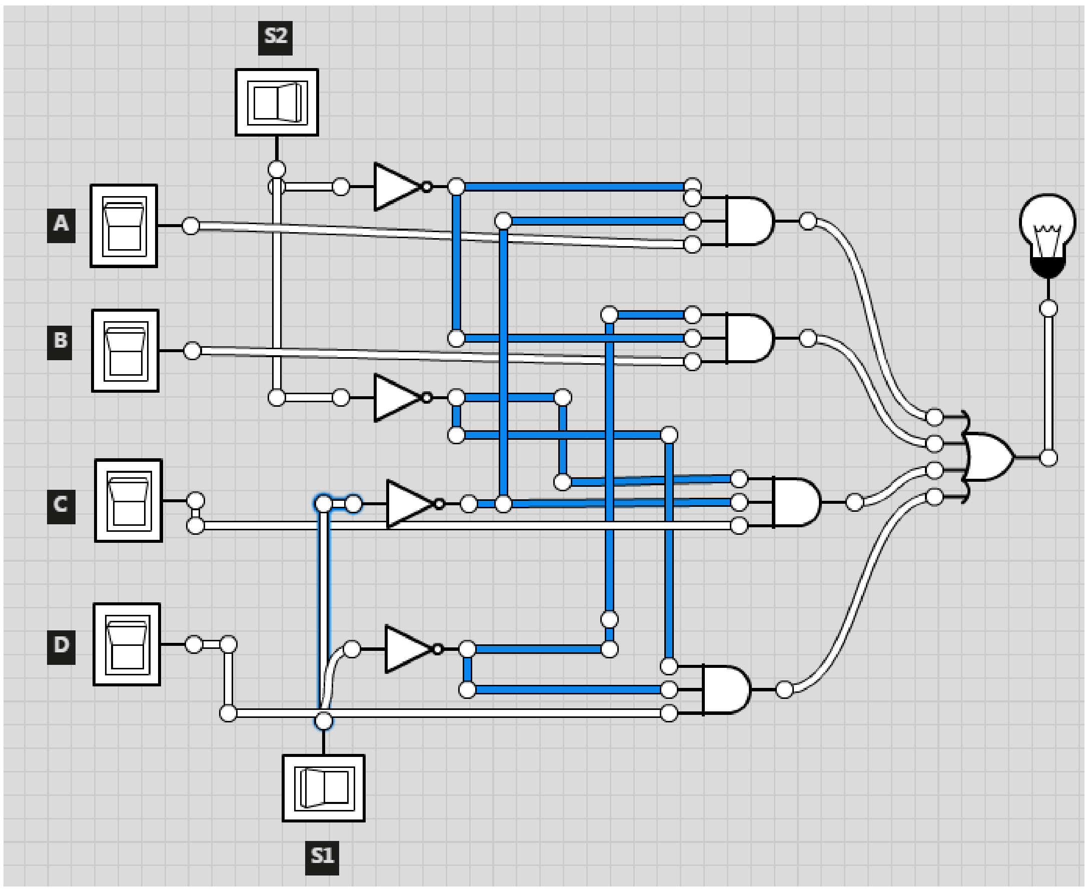


(c) A 2 – to -1 Mux

Mux is known as multiplexer. The main function of a mux is to select one of n number of inputs to have an output of 1 just like a switch.

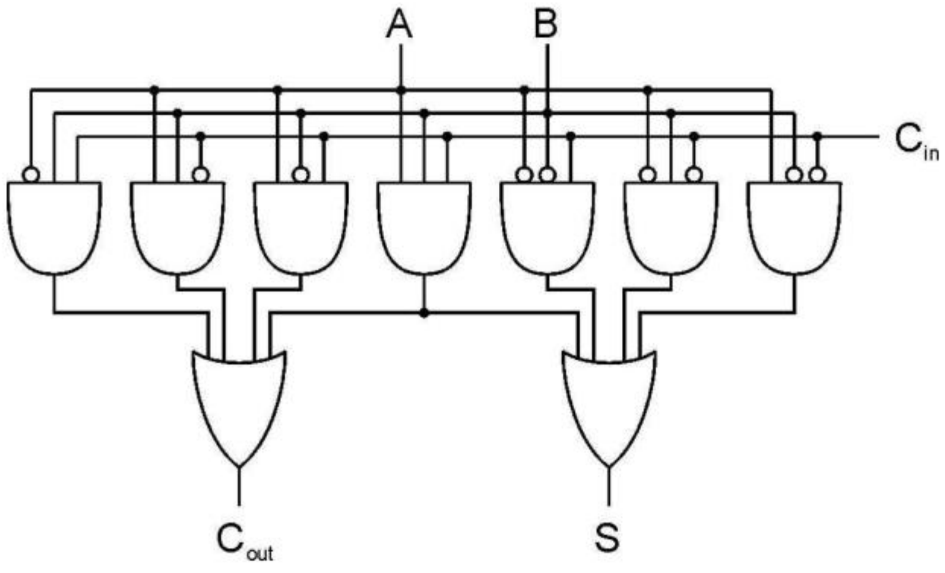


(d) A 4-to-1 mux

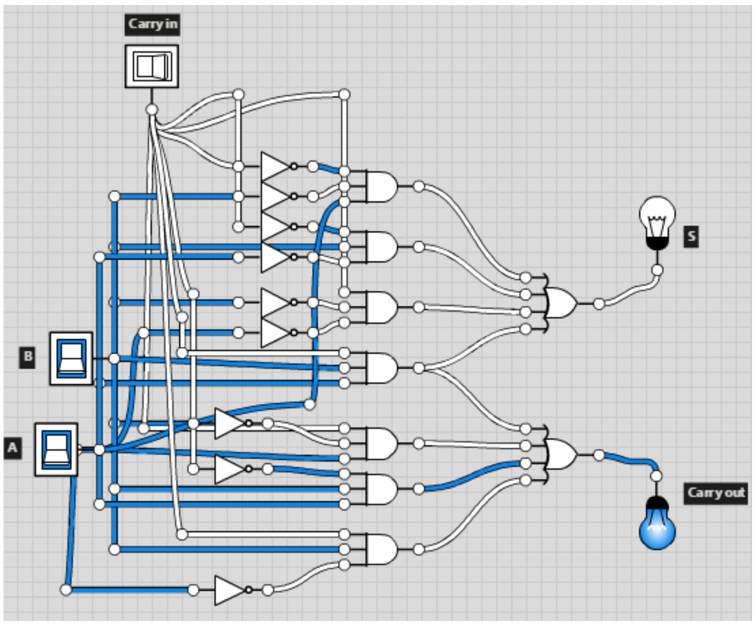


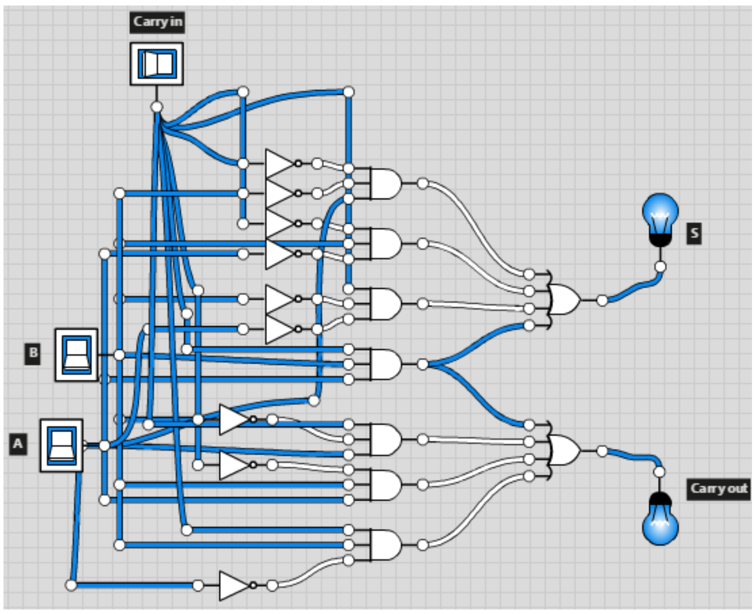
(e) A 2-bit Adder

A full adder adds two bits and carry in bits to produce a one-bit sum with a carry out.



The image above shows an example of a full 2-bit adder.

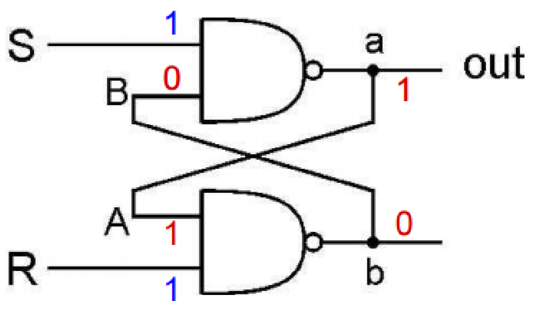


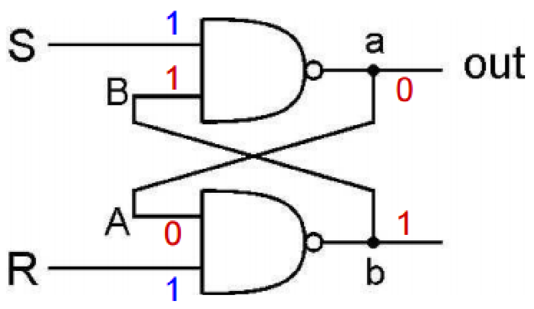


Week 5

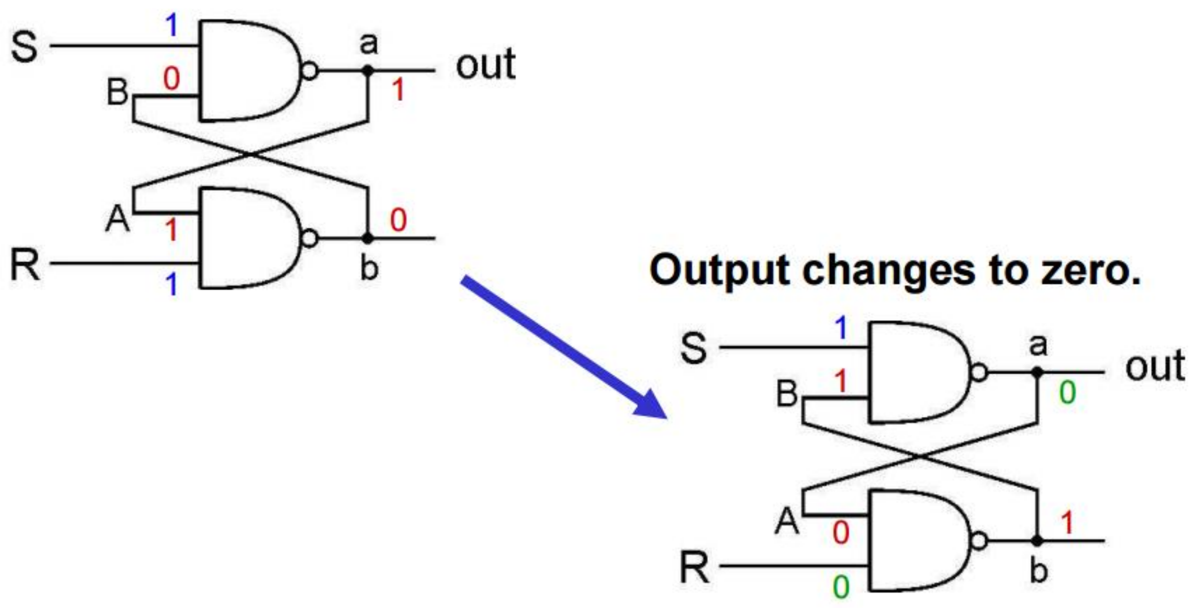
1. **Explain the difference between logic structures that include the storage of information and those that do not.**
   There are two types of circuits:
   - **Combinational circuit:**
     This type of circuit always gives you the same output for a set of given inputs.
   - **Sequential circuit:**
     This type of circuit store information. The output of the circuit depends on the stored data or state as well as the input.
     This means that a given input might produce different outputs due to stored image.
     This is useful circuit type for building memory.
2. **Using the information provided in the slides, explain how the R-S Latch, Gated D Latch, and register work.**

- **R-S Latch**:
  R (“reset”) is used to clear the element which means that R is set to 0.
  S (“set”) is used to set the element which means that S set to 1.

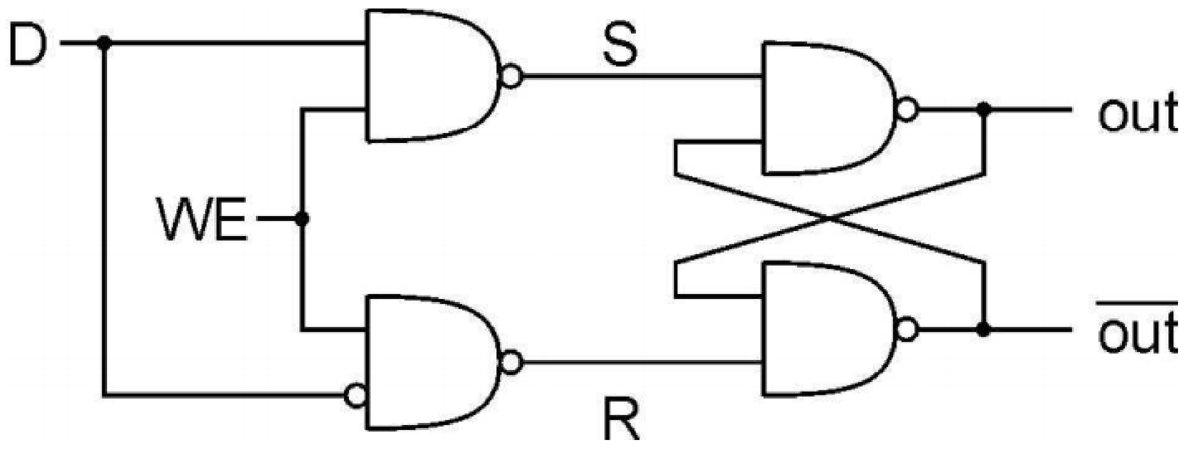




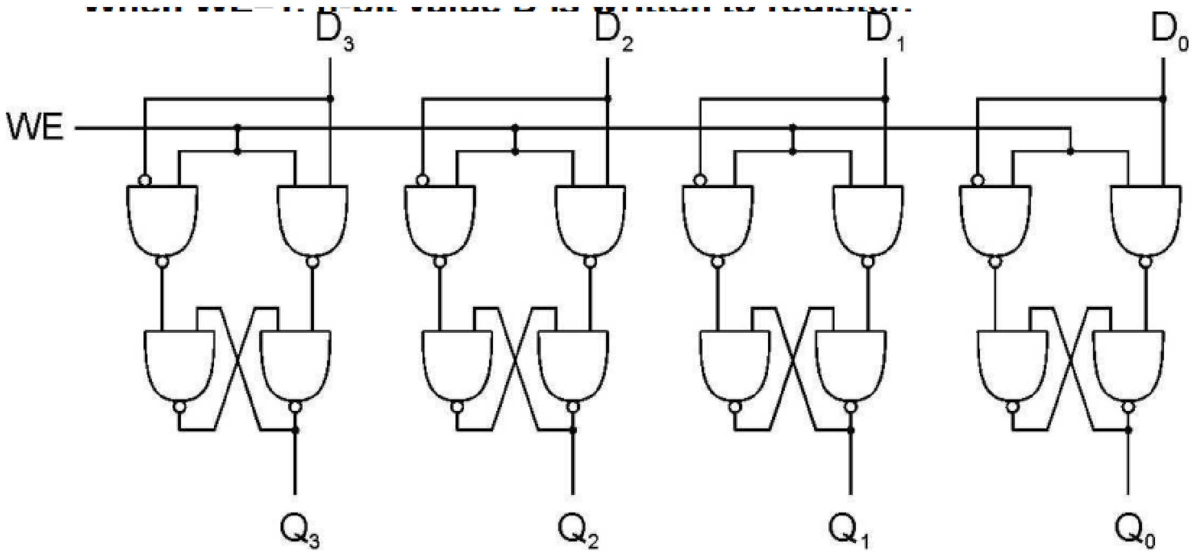
  When both the R and S are 1, the output could be either 0 or 1. The image above shows how the R-S latch is set.
  The output changes to 0 when clearing the R-S latch. The image below shows how the R-S latch is reset.



- **Gated D latch:**
  There are two types of inputs for D Latch: D which is data and WE which write enable.
  The latch is set to the value of D when WE = 1 and the latch holds previous value when WE is equal to 0.
  The image below shows the structure of D-Latch.



- **Register**
  The collection of D-Latches is controlled by a common WE is called a register.
  Register stores a multi-bit value.
  The image below shows the structure of ta register.

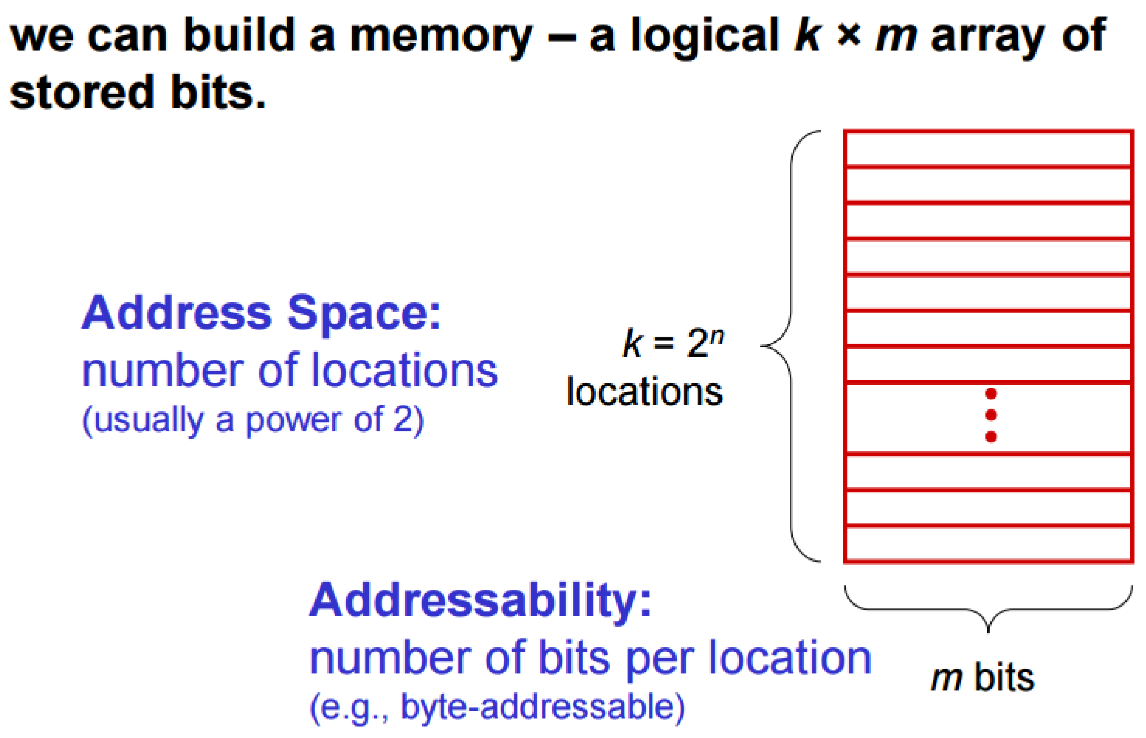


3. **Explain the concept of Memory, Address space and Addressability using current technologies and architectures as examples.**
   - **The concept of memory:**
     Memory is made up of large number of locations. Each location is identifiable and have the ability to store a value.
     <u>Two keywords that consists of:</u>
     - Address Space is the total number of locations.
     - Addressability is the number of bits stored in each memory locations (address space).

     <u>There are two basic kinds of Random Access Memory (RAM):</u>

     - Static RAM is fast and maintains data without power.
     - Dynamic RAM is slower and much denser. The bit storage must be periodically updated in other words it needs to refreshed.

- Modern types of memory include ROM, PROM, flash and etc.



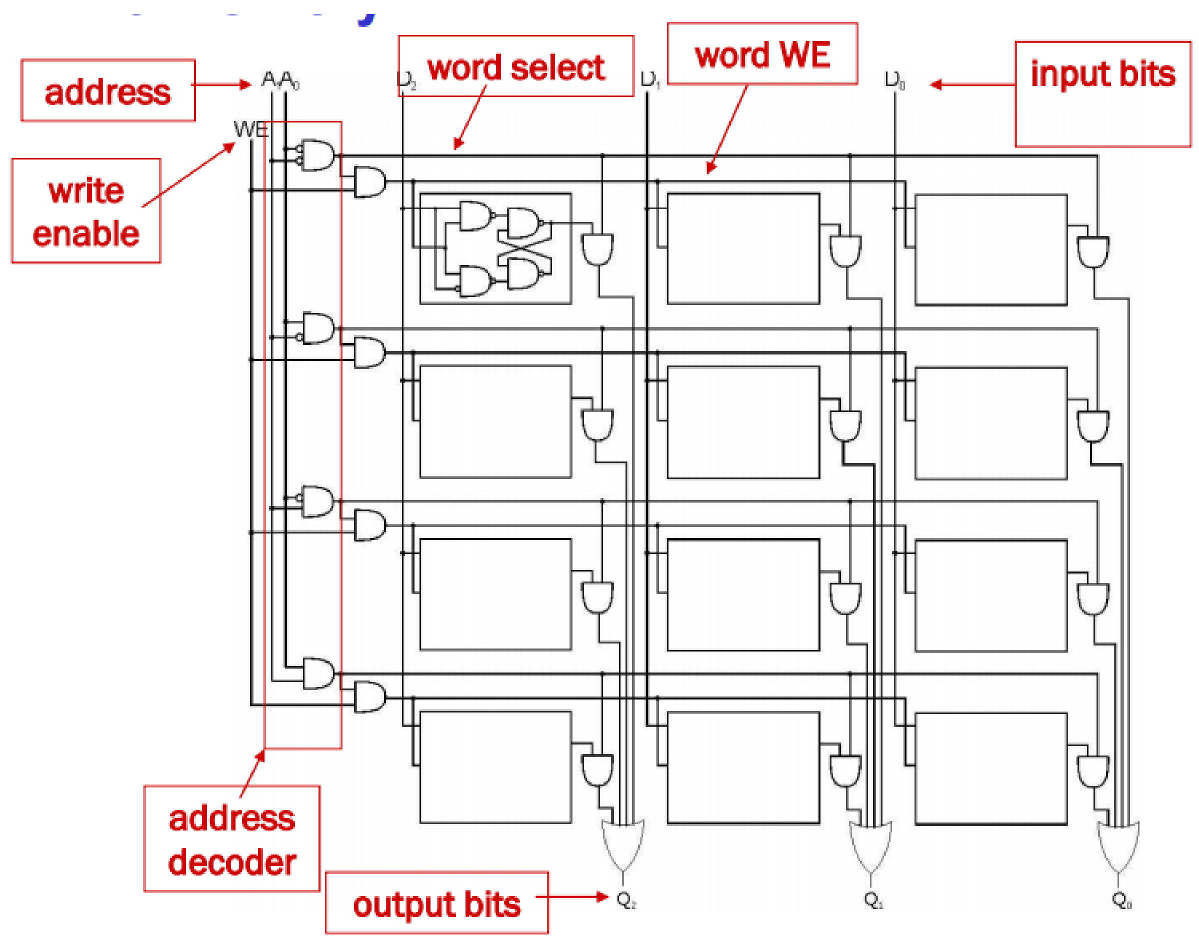
we can build a memory – a logical *k* × *m* array of stored bits.

Address Space:
number of locations
(usually a power of 2)

Addressability:
number of bits per location
(e.g., byte-addressable)

The image above shows how to calculate the amount of memory.

The image below shows the structure of $2^2 \times 3$ memory.



References

A. Patel, Patt (2004) *Introduction to Computing Systems from bits and gates to C and beyond.* Second Edition: McGraw Hill Education Private Limited

B. Dariush Kheirkhahzadeh, A (2016) “Computer Architecture lecture slides” [PowerPoint presentations]. *Week 1 to Week 5.* Available at https://online.uwl.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_82939_1&content_id=_1730544_1 (Accessed: 4 November 2016).

C. Oxford Dictionaries (2016) *Definitions of Computer.* Available at: https://en.oxforddictionaries.com/definition/computer (Accessed: 27 September 2016)