FIT9137 Applied Session
Week 1

Topics:
- Introduce yourself. Build a community within your session
- Get to know your tutor. Get to know where to get help
- Introduction to Number Systems and Boolean Logic

Instructions:
- One of the main purposes of an applied session is to build the learning community, create connections and include the learners. The other goal is to give and receive feedback from your peers and or your tutors.
- Form groups of 2 students (peers) to work through the exercises. If you meet a problem, try to solve it by asking direct questions to your peers. If the issue was not solved within peers, ask your tutor. If you did not get a chance to solve the problem during your applied session with your peer or tutor, jump into one of many consultation hours and ask any of the tutors to help you. Please visit the "Teaching Team and Unit Resources" tile in the FIT9137 Moodle site.

1. Introductions

Get to know your group partner! Perhaps find out about the following things:

- Name
- Country of origin
- Degree they're enrolled in
- What their previous degree was
- One fun fact about them

Then it's your turn to **introduce your group partner** to the whole group.

2. Expectations

Let's start with a few "ground rules" for this unit. Have a quick discussion about whether the following rules are okay with you, and perhaps add more:

1. We encourage everyone to participate in discussions. We understand that learning means making mistakes and expressing ideas that are not fully thought through.

2. We are going to be polite and respectful. Any threatening, bullying, harassing behaviour has absolutely no place at Monash.

3. We acknowledge that our classmates (and teachers) come from a wide range of backgrounds: different cultures, languages, opinions, skills, and different previous

knowledge about IT. We will keep this in mind during discussions and when working in groups.

4. We will make an effort to use English in class whenever possible (even if that's not our main language and we're talking to our friends).

5. We respect each others' right to privacy and keep any personal information shared during lab sessions to ourselves.

Regarding point 3: Of course **you are allowed** to talk in your native language, we just suggest that you use the opportunity to improve your English skills. This will make it easier to interact with your classmates, to complete your assignments, and to express your thoughts in the exam.

Regarding point 5: Of course **you are allowed** (and encouraged) to report any antisocial, bullying or harassing behaviour.

3. Bits, Bytes and Numbers

3.1 Bit, Byte and Word

Construct a bit, a byte, and a word.

3.2 Build a Table of Number Systems

Build a table of Number Systems consisting of- Base 10, Base 8, Base 16 & Base 2 numbers. This table will contain 16 numbers only (decimal numbers 0-15). Represent Base-2 numbers using 4 bits.

Use appropriate notations to differentiate between different number systems:

(a) subscripts "10" for Decimal Numbers- 45_{10},
(b) subscripts "8" for Octal Numbers- 45_{8},
(c) subscripts "16" for Hexadecimal Numbers- 45_{16},
(d) subscripts "2" for Binary Numbers- 1010_{2}.

4. Number System Conversions

4.1: Binary to Decimal and Decimal to Binary

(i) Convert the Decimal number $\mathbf{165_{10}}$ to a Binary number using Division & Remainder operations.

(ii) Convert the Binary number $\mathbf{11000101_{2}}$ to a Decimal number using step (or place) value and multiplication process.

4.2: Binary to Hexadecimal and Hexadecimal to Binary

(i) Convert the Binary number $\mathbf{11101010_{2}}$ to a Hexadecimal number using the Number Systems table.

(ii) Convert the Hexadecimal number **$FA01_{16}$** to a Binary number using the Number Systems table.

4.3: Hexadecimal to Decimal and Decimal to Hexadecimal

(i) Convert the Hexadecimal number **$1AF_{16}$** to a decimal number using place value and multiplication operations.

(ii) Convert the Decimal number **151_{10}** to a Hexadecimal number using division and remainder operations.

5. Representation of Characters (using ASCII) and Numbers

In our computers, characters are represented using ASCII/Unicode and numbers are represented using 2's complement representation.

(Note: Using MARIE we can show how the computer keyboard characters are represented in computer memory.)

5.1: Representation of Characters.

If your computer uses 7-bit ASCII, how would the computer represent the string "FA5"?

5.2: Integer Numbers (4-bit) Representing Unsigned, Signed, 1's & 2's complement representations.

Create a table with all possible 4-bit binary values in one column and the (equivalent) decimal they represent in different columns using the following different notations:

(i) unsigned integer,
(ii) sign-magnitude notation,
(iii) 1's complement representation,
(iv) 2's complement representation.

	4-bit Binary Number 16 Numbers	Unsigned Representation 16 Numbers [0-15]	Signed Representation 16 Numbers [-7 -0][+0 +7]	1's Complement Representation 16 Numbers [-7 -0][+0 +7]	2's Complement Representation 16 Numbers [-8 -1][0 +7]
1	0000	0	+0	+0	0
2	0001	1	+1	+1	+1
3	0010	2			
4	0011	3			
5	0100	4			
6	0101	5			
7	0110	6			

8	0111	7			
9	1000	8			
10	1001	9			
11	1010	10			
12	1011	11			
13	1100	12			
14	1101	13			
15	1110	14			
16	1111	15			

5.3: Integer Numbers (8-bit)

(a) Assume you have 8 bits to represent binary numbers. What decimal value does the binary number 10110101_2 have in the different notations?
 (i) unsigned binary number
 (ii) sign-magnitude notation
 (iii) 1's complement
 (iv) 2's complement

(b) Represent the number -92_{10} in
 (i) 8-bit signed magnitude
 (ii) 8-bit 2's complement

5.4: Addition and subtraction of signed numbers.

(a) Perform the following tasks using binary arithmetic with 8-bit 2's complement representation:
 (i) 33 + 92
 (ii) 33 - 92

(b) Convert the following numbers to 6-bit 2's complement notation and add them:
 (i) -16+11
 (ii) 16-11

5.5: Concept of "Ranges of Numbers and Overflow".

(a) What are the ranges of numbers that can be represented in a computer if it is using:
 (i) 4-bit 2's complement representation
 (ii) 6-bit 2's complement representation
 (iii) 8-bit 2's complement representation

(b) What happens if you try to calculate 92+92 (using binary arithmetic with 8-bit 2's complement representation)? Can the result of this operation be accepted? How would you explain this operation referring to the "ranges of numbers" in 8-bit 2's complement representation?

6. Boolean Logic and Logic Gates

6.1: Using the basic logic gates (AND, OR and NOT)

Make yourself familiar with the basic logic gates by doing a few very easy tasks:

a. Draw a logic circuit using an AND gate (2 inputs and 1 output). Construct the associated truth table with all possible input combinations.
b. Repeat the task (a) using an OR gate.

c. Draw a logic circuit using a NOT gate (1 input and 1 output). Construct the associated truth table with all possible input combinations.
d. Try some combinations of gates, such as "AND & NOT" and "OR & NOT". Construct the associated truth table with all possible input combinations.

6.2: Build logic circuits using a combination of AND, OR and NOT gates

In addition to AND, OR and NOT, there are a number of other Boolean logic operators. Examples are XOR (the exclusive OR) operation. Its truth table is shown in the table below.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

(a) XOR

A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

(b) NAND

Build XOR logic circuit only using AND, OR, and NOT gates. Trace the inputs to the output to test your circuit and match with the XOR truth-table.

7. Boolean Equation Simplification (using Boolean Laws)

Given a Boolean Expression:

$$F(X,Y,Z) = Y(\overline{X}Z + \overline{Z}) + \overline{Y}(XZ + \overline{Z})$$

- (i) Write down the truth table,
- (ii) Draw a logic circuit using the original Boolean expression,
- (iii) Simplify the equation,
- (iv) Draw the logic circuit using the simplified Boolean expression.

==

ASCII Table (7-bit)

Bit pattern	Character	Decimal value	Bit pattern	Character	Decimal value
0100000	SPACE	32	0100001	!	33
0100010	"	34	0100011	#	35

0100100	$	36	0100101	%	37
0100110	&	38	0100111	'	39
0101000	(	40	0101001	)	41
0101010	*	42	0101011	+	43
0101100	,	44	0101101	-	45
0101110	.	46	0101111	/	47
0110000	0	48	0110001	1	49
0110010	2	50	0110011	3	51
0110100	4	52	0110101	5	53
0110110	6	54	0110111	7	55
0111000	8	56	0111001	9	57
0111010	:	58	0111011	;	59
0111100	<	60	0111101	=	61
0111110	>	62	0111111	?	63
1000000	@	64	1000001	A	65
1000010	B	66	1000011	C	67
1000100	D	68	1000101	E	69
1000110	F	70	1000111	G	71
1001000	H	72	1001001	I	73
1001010	J	74	1001011	K	75
1001100	L	76	1001101	M	77
1001110	N	78	1001111	O	79
1010000	P	80	1010001	Q	81
1010010	R	82	1010011	S	83
1010100	T	84	1010101	U	85
1010110	V	86	1010111	W	87
1011000	X	88	1011001	Y	89
1011010	Z	90	1011011	[	91
1011100	\	92	1011101	]	93
1011110	^	94	1011111	_	95
1100000	`	96	1100001	a	97
1100010	b	98	1100011	c	99
1100100	d	100	1100101	e	101
1100110	f	102	1100111	g	103
1101000	h	104	1101001	i	105
1101010	j	106	1101011	k	107

1101100	l	108	1101101	m	109
1101110	n	110	1101111	o	111
1110000	p	112	1110001	q	113
1110010	r	114	1110011	s	115
1110100	t	116	1110101	u	117
1110110	v	118	1110111	w	119
1111000	x	120	1111001	y	121
1111010	z	122	1111011	{	123
1111100	|	124	1111101	}	125