Course wide

Week 1 - Notes

Course wide

Binary 10 = 2 decimal (Number)

* Relay/Binary transistor USED ON/current = 1
  OFF/No current = 0

* Petaflop = 10^{15} (10 to the Power 15) Floating Point Operations Per Second

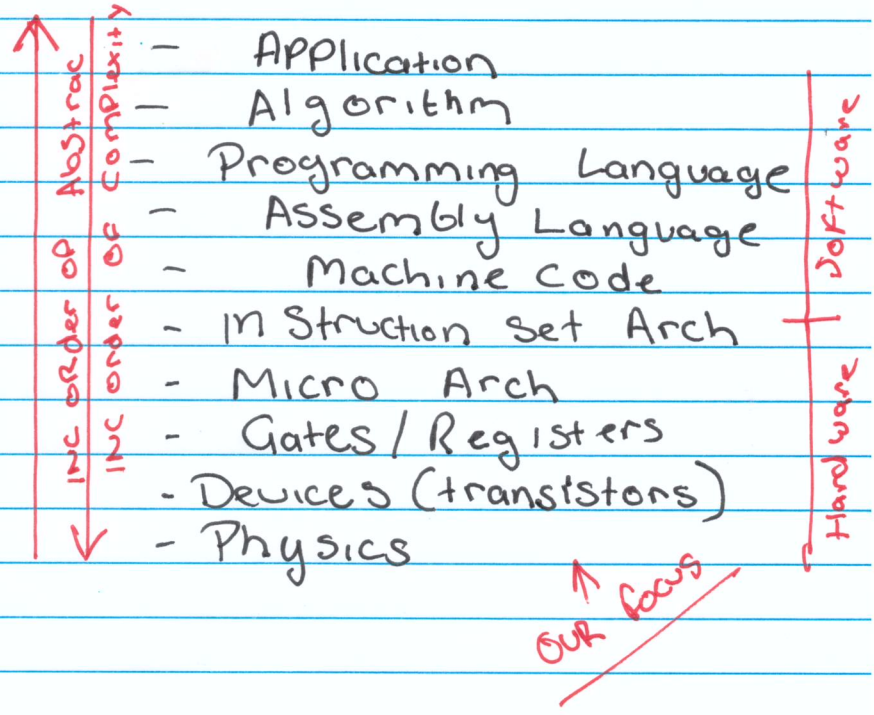
* ARM = Advanced Risc Machine - Side Note
  ↳ Reduced Instruction Set Computer
* CISC = Complex Instruction Set Computer

* Moore's Law = the number of transistors increase exponentially in every generation
  Soon → *No Longer Really Applicable

* Computer Systems - Abstraction

USER
Programs
Operating Systems
Hardware

Computer Layerd as a sys



* 1's and 0's = Bits
→ at the lowest levels of digital computing hardware everything is either 0 or 1

0 1 0 0 1 0 1 0 = J → ASCII (7 Bit code) often Extended to 8

→ Humans dont like ~~trang~~ binary so we use
Octal, decimal & Hexadecimal
↓ ↓ ↓
Base 8 Base 10 Base 16

→ Binary = Base 2 number system

Consider base 5 number: 410_5

$410_5 = 4 \times 5^2 + 1 \times 5^1 + 0 \times 5^0$

→ in Base 5, each Place indicates a power of 5
→ 410_5 in Base 5 = 105 in base 10

Binary to decimal

Consider binary: 1101001:

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

$1 \times (2 \times 2 \times 2 \times 2 \times 2 \times 2)$ (6)

$0_1 = 0$

$= 1$?

$= 64 + 32 + 0 + 8 + 0 + 0 + 1$

$= 105_{10}$ ← As Decimal ☺

MSB = Most-Left = Most-Significant Bit 1001
LSB = Most-Right = Least- " " Bit 1001

Decimal to Binary

*Consider: 178_{10}

* Devide 178 By 2 until Result = 0
Results with decimal Place (IE) 1.1 = 1
Whole number Results = 0 (Work Backwards)

	Quotient	Remainder	
178/2	89	0	(LSB)
89/2	44 (44.4)	1	
44/2	22	0	
22/2	11	0	
11/2	5 (5.5)	1	
5/2	2 (2.5)	1	
2/2	1	0	
1/2	0 (0.5) STOP Bit	1	(MSB)

Read Up for Answer

$178_{10} = 10110010_2$

* Binary to Octal Equivalent *

$10110010_2 = 010/110/010_2$ (Add)

Split into groups of 3 starting at LSB
add 0's to MSB to Gain group of 3

$1 \times 2^8 + 0 \times 2^8$

=

1) 010 2) 110 3) 010

* Binary to Octal - CONT

1) 010

$= 0 \times 2^{2} + 1 \times 2^{1} + 0 \times 2^{0} = 2$

$0 \quad + \quad 2 \quad + \quad 0$

2) 110

$= 1 \times 2^{2} + 1 \times 2^{1} + 0 \times 2^{0} = 6$

$4 \quad + \quad 2 \quad + \quad 0$

3) 010

$= 0 \times 2^{2} + 1 \times 2^{1} + 0 \times 2^{0}$

$0 \times (2 \times 2) + 1 \times (2) + 0 \times (0)$

$0 \quad + \quad 2 \quad + \quad 0 = 2$

$10110010_{2} = 262_{8}$

Binary Numbers

- With 4 bits we can represent:

- $0000_2 = 0_{10}$
- $0001_2 = 1_{10}$
- $0010_2 = 2_{10}$
- $0011_2 = 3_{10}$
- $0100_2 = 4_{10}$
- $0101_2 = 5_{10}$
- $0110_2 = 6_{10}$
- $0111_2 = 7_{10}$
- $1000_2 = 8_{10}$
- $1001_2 = 9_{10}$
- $1010_2 = 10_{10}$ A
- $1011_2 = 11_{10}$ B
- $1100_2 = 12_{10}$ C
- $1101_2 = 13_{10}$ D
- $1110_2 = 14_{10}$ E
- $1111_2 = 15_{10}$ F

Hexadecimal

U1

* Binary to Hexadecimal

$1011\ 0010_2$

* Devide in to groups of 4 called a Nibble Starting from Right (LSB). add extra zero's infront of MSB as Required

$1011\ \ 0010_2 = B2_{16}$

* Binary Addition

1) $0101\ 0101_2 + 0001\ 0010_2$

CARRY 1

\+ 0 1 0 1 0 1 0 1

\+ 0 0 0 1 0 0 1 0

Result 0 1 1 0 0 1 1 1

=2 so Carry 1, Mark AS 0

Valid to 8 Bits

~~Lso F~~

2) $1111\ 0101_2 + 1001\ 0110_2$

+1 +1 +1 +1

1 1 1 1 0 1 0 1

\+ 1 0 0 1 0 1 1 0

= 1 0 0 0 1 0 1 1

Carry Condition INValid to 8 Bits

+1 +1 +1 +1 +1

1 1 1 1 0 1 0 1

\+ 1 0 0 1 0 1 1 0

= 1 0 0 0 1 0 1 1

7 →

Binary Addition

			+1					
	0	1	0	1	0	1	0	1
+	0	0	0	1	0	0	1	0
Valid to 8 bits	0	1	1	0	0	1	1	1

+1	+1	+1	+1		+1			
	1	1	1	1	0	1	0	1
+	1	0	0	1	0	1	1	0
Carry conditi on-invalid to 8 bits	1	0	0	0	1	0	1	1

Carry Condition

- Computers can only allocate a finite number of bits to represent a number
- Assume only 8 bits are used for each integer

+1	+1	+1	+1	+1	+1	+1	+1	
	1	1	1	1	1	1	1	1
+	0	0	0	0	0	0	0	1
	0	0	0	0	0	0	0	0

This bit is lost, this error is known as a carry condition

Note that $1111\ 1111_2 = 255_{10}$

* Positive Binary is UnSigned
* Negative Binary is Signed

* there are other binary coding systems for Signed Integers such as

  → two's complement
  → one's complement

* there are other binary coding systems for Representing Floating Point numbers

→ 1's complement (only flipping Bits)

$+5_{10} = 0101_2$ & $-5_{10} = 1010_2$

$+2_{10} = 0010_2$ & $-2_{10} = 1101_2$

* Problem with 1's complement is it has 2x zero's (0000_2 & 1111_2) so we always need to check for Both which is inefficient

→ 2's complement

* to get 2's complement use 1's complement the add +1

* 4 Bits arithmetic

$0101_2 (5_{10}) \Rightarrow$ 1's complement 1010_2 +1 (using Binary ADD) $\Rightarrow 1011_2$

1011_2

↗ Tell us it's a Negative in 2's complement

* Overflow in 2's Complement is ignored

Negative Integers- 2's Complement

- 1011 (-5_{10})
- 0100 =>1's complement

  +1 add 1
- 0101 We get back the original number=> $+5_{10}$

- Now consider 0 =0000=> 2's complement

+1	+1	+1	+1		
	1	1	1	1	1's complement
			+	1	Add 1
	0	0	0	0	

In 2's complement you **ignore the carry bit** if you are using 4 bit arithmetic

- **So there is only one representation of 0 in 2's complement**

Characters- ASCII Table

- ASCII is a 7 bit code (often extended to 8 bits)
- Note: 1 is not 1 !!
- Some ASCII characters:

01000001 = 'A'
01000010 = 'B'
01000011 = 'C'
...
01011010 = 'Z'
..
01100001 = 'a'
01100010 = 'b'

00110000 = '0'
00110001 = '1'
00110010 = '2 '
00100000 = ' '
0 00100001 =' ! '
00100100 = '$'
00000111 = ring bell !!