**Introduction to Computer Systems and Platform Technologies**

Study Period 3, 2021 – CPT160

Assessment 1

|  |  |
| --- | --- |
| **Student Name:** | Adam Mutimer |
| **Student Number:** | S3875753 |

# Task 1 – Number Systems

Student Number: s3875753  
X = 5753

1. Convert X from decimal to binary

|  |  |  |  |
| --- | --- | --- | --- |
| Division by 2 | Quotient | Remainder | Bit# |
| 5753/2 | 2876 | 1 | 0 |
| 2876/2 | 1438 | 0 | 1 |
| 1438/2 | 719 | 0 | 2 |
| 719/2 | 359 | 1 | 3 |
| 359/2 | 179 | 1 | 4 |
| 179/2 | 89 | 1 | 5 |
| 89/2 | 44 | 1 | 6 |
| 44/2 | 22 | 0 | 7 |
| 22/2 | 11 | 0 | 8 |
| 11/2 | 5 | 1 | 9 |
| 5/2 | 2 | 1 | 10 |
| 2/2 | 1 | 0 | 11 |
| 1/2 | 0 | 1 | 12 |

X = 0001 0110 0111 1001

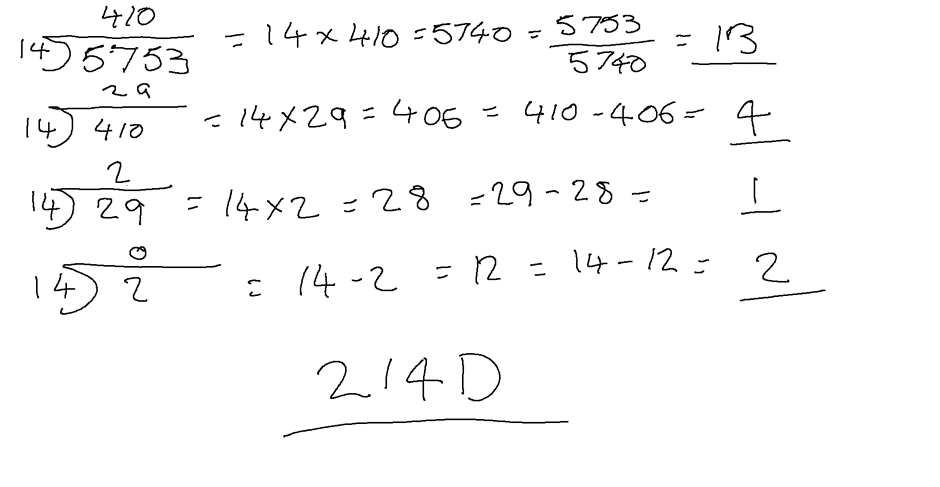
1. Convert the binary string obtained from your answer to (a) into an octal and hexadecimal
   1. Octal Conversion

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Octal Digit Value** | **Binary Equivalent** | | 0 | 000 | | 1 | 001 | | 2 | 010 | | 3 | 011 | | 4 | 100 | | 5 | 101 | | 6 | 110 | | 7 | 111 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 001 | 011 | 001 | 111 | 001 | | 1 | 3 | 1 | 7 | 1 |   Octal = 13171 |

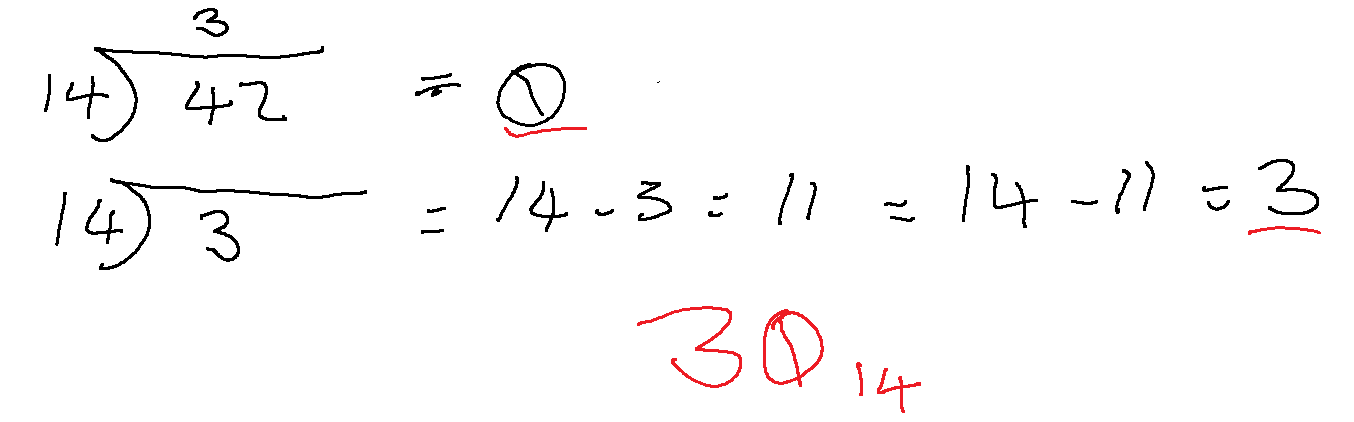
* 1. Hexadecimal Conversion

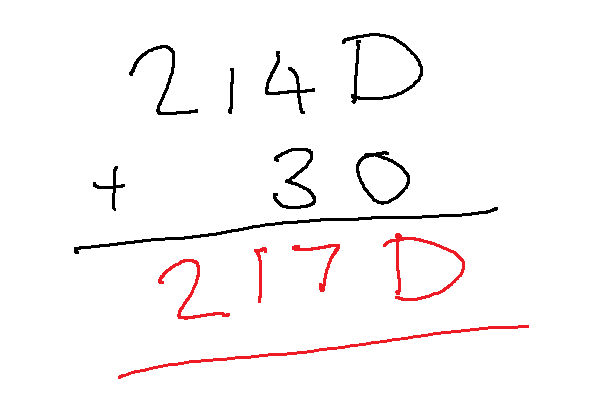
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Hex Value** | **Binary Equivalent** | | 0 | 0000 | | 1 | 0001 | | 2 | 0010 | | 3 | 0011 | | 4 | 0100 | | 5 | 0101 | | 6 | 0110 | | 7 | 0111 | | 8 | 1000 | | 9 | 1001 | | A | 1010 | | B | 1011 | | C | 1100 | | D | 1101 | | |  |  |  |  | | --- | --- | --- | --- | | 0001 | 0110 | 0111 | 1001 | | 1 | 6 | 7 | 9 |   Hexadecimal = 1679 |

1. Convert X from decimal to base 14, Where A, B,C and D correspond to 10, 11, 12 and 13 respectively.

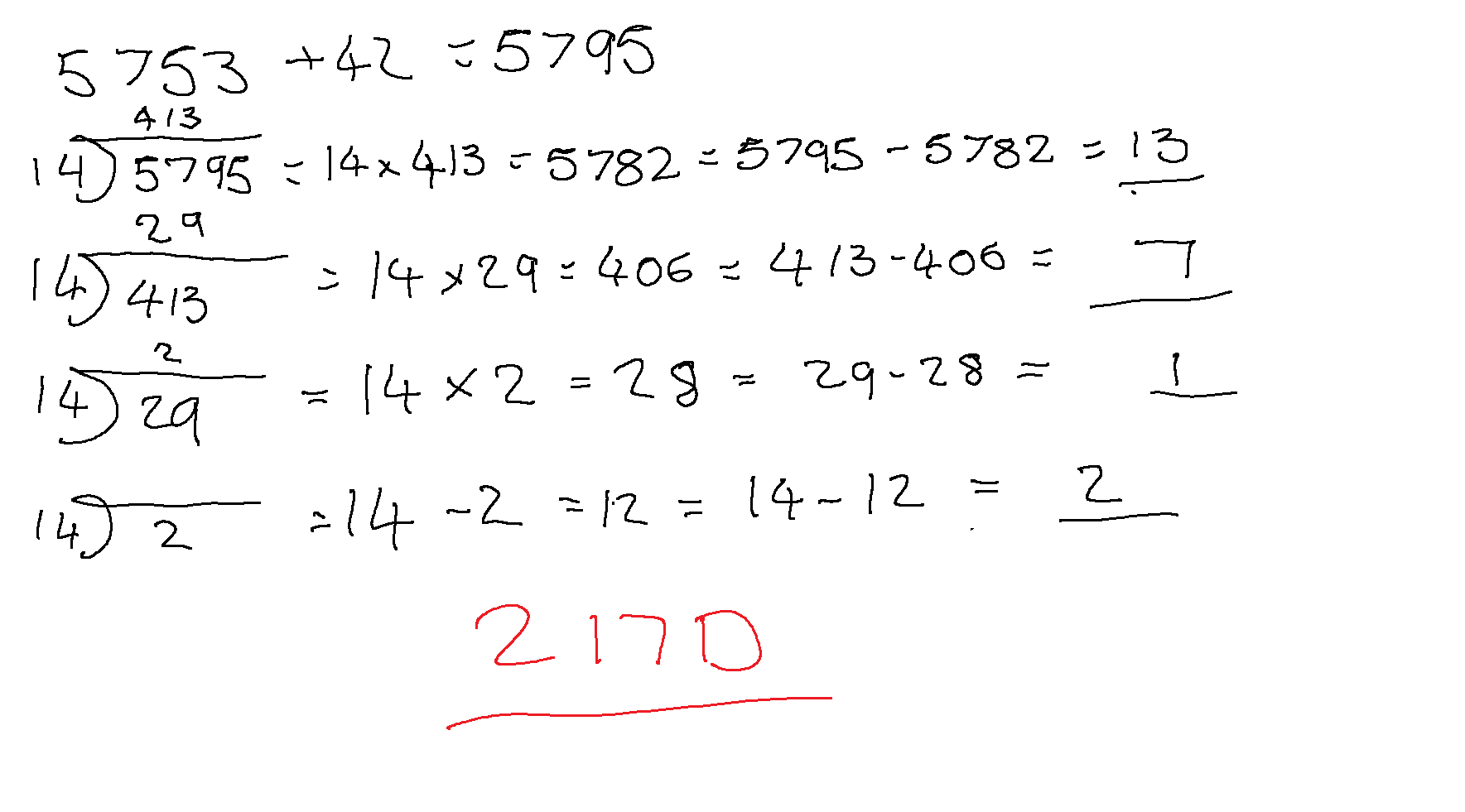


1. Now add (42 in decimal) to X and calculate the sum in base 14. Consider the following two calculations:
2. Conversion (base 14 to decimal) before addition (in base 14): convert in to base 14, then add the two base 14 numbers.





1. Addition (in decimal) before conversion (decimal to base 14): add to X in decimal, then convert the decimal sum into base 14.



1. Which calculation is simpler? Please explain you answer. How many digits are different from your answer to (c)?

In this instance I found the conversion method simpler, however I do prefer the addition before conversion as its easy to follow when going back to review the work as it makes more sense to me personally. Only 1 digit was different from my answer to (c) 214D is now 217D, 4 being the single digit that has changed to a 7.

1. Consider a base 26 number system wherein the letters of the alphabet are the digits.

*That is, A=0, B=1, C=2, ….Z=25 in base 10.*

Use the first three letters of your given name as a number in the base 26 system, and the first three letters of your surname as another number in the base 26 system.

Add these numbers together to obtain the sum in base 26.

ADA = A=0, D=3, A=0  
MUT = M=12, U=20, T=19

|  |  |  |  |
| --- | --- | --- | --- |
| Base 26 | A | D | A |
| * Base10 | 0 | 3 | 0 |
| Base 26 | M | U | T |
| * Base10 | 12 | 20 | 19 |
| + (Addition Base10) | 12 | 23 | 19 |
| = (SUM Base26) | M | X | T |

SUM = MXT

# Task 2 – Binary Addition and Subtraction

1. Convert the decimal numbers A and B to 4-bit binary numbers. (A=3 & B=5). Show hoe to add together these two 4-bit binary numbers and state whether the answer is valid to 4-bit arithmetic.

|  |  |  |  |
| --- | --- | --- | --- |
| **Convert A () to Binary** | | | |
| Division by 2 | Quotient | Remainder | Bit# |
| 3/2 | 1 | 1 | 0 |
| 1/2 | 0 | 1 | 1 |
| 0/2 | 0 | 0 | 2 |

A = 011 🡪 4-Bit =

|  |  |  |  |
| --- | --- | --- | --- |
| **Convert A () to Binary** | | | |
| Division by 2 | Quotient | Remainder | Bit# |
| 5/2 | 2 | 1 | 0 |
| 2/2 | 1 | 0 | 1 |
| 1/2 | 0 | 1 | 2 |

B = 101 🡪 4-Bit =

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Carry*** | 1 | 1 | 1 |  |
|  | 0 | 0 | 1 | 1 |
| **+** | 0 | 1 | 0 | 1 |
| **=** | 1 | 0 | 0 | 0 |

SUM = this is answer is valid to 4-bit arithmetic

1. Convert the decimal numbers A and B to 5-bit binary numbers. Using two’s compliment representation, show how to:

A = 0011 B= 0101

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **16’s** | **8’s** | **4’s** | **2’s** | **1’s** | **TASK** |
|  | *0* | *0* | *1* | *1* | *4-bit* |
|  |  |  |  |  |  |
| *0 (Positive Number)* | *0* | *0* | *1* | *1* | *5-Bit* |
| *0* | *1* | *1* | *0* | *0* | *1’s Compl.* |
| *1 (Flipped this to become a negative)* | *1* | *1* | *0* | *1* | *2’s Compl.* |

**To confirm 11101 = -3:***-16 + 8 + 4 +1 = -3*

* 1. Subtract the two 5-bit binary numbers (-A-B)

*We need to make (A) a negative number, so we flip bits and add 1 = A=11101*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **16’s** | **8’s** | **4’s** | **2’s** | **1’s** |
| **CARRY** | 1 | 1 |  | 1 |  |
|  | 0 | 0 | 1 | 0 | 1 |
| **+** | 1 | 1 | 1 | 0 | 1 |
| **1 (Overflow Ignore)** | 0 | 0 | 0 | 1 | 0 |

***00010 = 2* to confirm this:**

*0 (16’s) + 0 (8’s) + 0 (4’s) + 1 (2’s) + 0 (1’s) = 2*

**To confirm:**

As a decimal A= -3 and B = 5  
*-3 + 5 (5 -3) = 2*

SUM = 00010

* 1. How to translate the binary result back to decimal.

(Note if your solution is negative, you must use 2’s compliment to show the positive equivalent)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#4** | **#3** | **#2** | **#1** | **#0** |
| 0 | 0 | 0 | 1 | 0 |
|  |  |  |  |  |
| 0 | 0 | 0 | 2 | 0 |

*0 + 0 + 0 + 2 + 0 = 2*

SUM = 2

# Task 3 – Bitwise Operations

**A AND M =**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **AND Truth Table** | | | | Input | Mask | Result | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 | | |  |  |  | | --- | --- | --- | | **OR Truth Table** | | | | Input | Mask | Result | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 | | |  |  |  | | --- | --- | --- | | **XOR Truth Table** | | | | Input | Mask | Result | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 | |
|  | |  |  |  | | --- | --- | --- | | **NOT AND  Truth Table** | | | | Input | Mask | Result | | 0 | 0 | 1 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 | |  |

1. Reset but 0, bit 7 and leave the reset untouched

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit Position** | ***7*** | ***6*** | ***5*** | ***4*** | ***3*** | ***2*** | ***1*** | ***0*** |
| **Input** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| **Mask** | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| **Result of** AND **Operation** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Result:

1. Make sure that bit 2 and bit 6, and only these, are reset, the others are set

*My understanding of this question is that bit 2 and 6 need to be reset to 0 while all other bits are set to 1*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit Position** | ***7*** | ***6*** | ***5*** | ***4*** | ***3*** | ***2*** | ***1*** | ***0*** |
| **Input** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| **Mask** | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| **Result of XOR** **Operation** | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |

Result:

1. Toggle the values of the middle 4 bits (the opposite of what thewy are currently) and set the 2bits on each side.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit Position** | ***7*** | ***6*** | ***5*** | ***4*** | ***3*** | ***2*** | ***1*** | ***0*** |
| **Input** | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| **Mask** | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| **Result of** NAND **Operation** | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

Result:

# Task 4 – Logic Circuits and Truth Tables

# Task 5 – Pipelining

# Task 6 – CPU Architecture

# Task 7 – Memory

# Task 8 – Hamming & SECDED Code

# Advanced Questions