CPT160 –

Intro to Computer Systems

Assignment 1

Channon Harper
S3871491

Contents

Task 1 Number Systems

a) Binary

student number s3871491

Division by 2	Quotient	Remainder
3871491	1935745	1
1935745	967872	1
967872	483936	0
483936	241968	0
241968	120984	0
120984	60492	0
60492	30246	0
30246	15123	0
15123	7561	1
7561	3780	1
3780	1890	0
1890	745	0
945	472	1
472	236	0
236	118	0
118	59	0
59	29	1
29	14	1
14	7	0
7	3	1
3	1	1
1	0	1

Binary = 1110110001001100000011_2

b) i) Octal

Division by 8	Quotient	Remainder Digit
3871491	483936	3
483936	60492	0
60492	7561	4
7561	945	1
945	118	1
118	14	6
14	1	6
1	0	1

octal = 16611403_8

ii) Hexadecimal

Division by 16	Quotient	Remainder Digit
3871491	241968	3
241968	15123	0
15123	945	3
945	59	1
59	3	11
3	0	3

11 in hex table is = B therefore:

hex = $3B1303_{16}$

c) Base 13

Division by 13	Quotient	Remainder Digit
3871491	297807	0
297807	22908	3
22908	1762	2
1762	135	7
135	10	5
10	0	10

10 in table is equal to a therefore base 13=

$A57230_{13}$

d) add to RMIT number 39_{10}

New number 3871530

Division by 13	Quotient	Remainder Digit
3871530	297810	0
297810	22908	6
22908	1762	2
1762	135	7
135	10	5
10	0	10

10 in table is equal to a therefore base 13=

$A57260_{13}$

There is 1 digit different this being the 3 changing to a 6, this is easy to calculate as 39_{10} is divisible by 13 the base we are working with no further numbers changed 39/13 = 3 so would expect the final number to have an additional 3. Since my original student number was also divisible by 13 made my table easier than others I would assume.

e) base 26 with alphabet

First 2 Letters First CH

First 3 Letters Last HAR

CH_{26} + HAR_{26}

CH_{26} = (26*2) + 7 = 59

HAR_{26} = (676*7) + (26*0) + 17 = 4732 + 0 + 17 = 4749

59 + 4749 = 4808

Division by 26	Quotient	Remainder digit
4808	184	24
184	7	2
7	0	7

Sum is = HCY_{26}

Working check HCY_{26}

*= (676*7) + (26*2) + 24 = 4732 + 52 + 24*

= 4808

Task 2 Binary Addition and Subtraction

3871491

Therefore A = 1 and B = 9

a)

Division by 2	Quotient	Remainder
1	0	1

Division by 2	Quotient	Remainder
9	4	1
4	2	0
2	1	0
1	0	1

1 = 0001

9 = 1001

Addition

 0001
+1001
=1010 this is valid to a 4-bit arithmetic as 4 bit can go to 15 and the value of my addition is 10

b) 5 bit 2's compliment 1 = 00001 9 = 01001

I and II) subtract (a – b) or (1-9)

find 2's compliment of B
flip first then convert by adding 1 9 is 01001 flipped 10110.

10110 +1
10111

The 9 is now 10111 or in 2's compliment or decimal -9 so can now add 1 to minus 9 which is the same as 1 – 9.

Carry 01110
00001 +10111
11000

I am unsure of I need to show a conversion procedure as is stated with 2's the 5 bit we are using would go -16 8 4 2 1 which is pretty straight forward to convert in my opinion.

Answer is 11000 which in 2's compliment is (-16+8) = -8$_{10}$

In case I am reading it wrong as the brackets on assignment say (-A-B) here is second conversion to suite that maths.

A= 1 B = 9 the sum reads (-1-9)

Convert a to minus by flipping and adding 1 for 2’s 00001 is 1 flip is 11110.

11110 +1
11111

Now to work with number provided previously for -9 (see above working) we would have -1+-9.

Carry 111110
11111 +10111
110110

The sum should answer -10

So, ignoring the 6th carry answer is 10110 or (-16+4+2) with a decimal of -10_{10}

Task 3 Bitwise Operations

a) M = 0100 0001 operator OR

Bit num	7	6	5	4	3	2	1	0
A (any Byte)	x	x	x	x	x	x	x	x
M	0	1	0	0	0	0	0	1
A or M	x	1	x	x	x	x	x	1

b) M1 = 1111 1111 operator OR then M2 = 1101 0111 with AND

Bit num	7	6	5	4	3	2	1	0
A (any Byte)	x	x	x	x	x	x	x	x
M1	1	1	1	1	1	1	1	1
A or M1	1	1	1	1	1	1	1	1

Bit num	7	6	5	4	3	2	1	0
A or M1	1	1	1	1	1	1	1	1
M1	1	1	0	1	0	1	1	1
(A or M1) AND M2	1	1	0	1	0	1	1	1

c) M1 = 1110 0111 operator XOR then M2 = 1110 0111 with AND

Bit num	7	6	5	4	3	2	1	0
A(any Byte)	x	x	x	x	x	x	x	x
M1	1	1	1	0	0	1	1	1
A XOR M1	/x	/x	/x	x	x	/x	/x	/x

Bit num	7	6	5	4	3	2	1	0
A XOR M1	/X	/x	/x	x	x	/x	/x	/x
M2	1	1	1	0	0	1	1	1
(A XOR M1) AND M2	/x	/x	/x	0	0	/x	/x	/x

Task 4 Logic Circuits and Truth Tables

Note: could not find symbol and was not wasting time.

Circuit 1 O=/(/(A.B)./(A.C)) XOR (B+/C)

Circuit 2 O= (/A.B) + /C

a	b	c	A.B	/A.B	A.C	/A.C	/(A.B)./(A.C)	/(/(A.B)./(A.C))	/C	B+/C	/(/(A.B)./(A.C)) XOR (B+/C)
1	1	1	1	0	1	0	0	1	0	1	0
1	1	0	1	0	0	1	0	1	1	1	0
1	0	1	0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	1	1	0	1	1	1
0	1	1	0	1	0	1	1	0	0	1	1
0	1	0	0	1	0	1	1	0	1	1	1
0	0	1	0	1	0	1	1	0	0	0	0
0	0	0	0	1	0	1	1	0	1	1	1

a	b	c	/A	/A.B	/C	(/A.B) + /C
1	1	1	0	0	0	0
1	1	0	0	0	1	1
1	0	1	0	0	0	0
1	0	0	0	0	1	1
0	1	1	1	1	0	1
0	1	0	1	1	1	1
0	0	1	1	0	0	0
0	0	0	1	0	1	1

c) Given the output columns these circuits do not match therefore not giving the same output. Unless they are meant to then I have misread the diagrams and I apologise.

Task 5 Pipelining

Non pipelined prepare times.

Read 7min 1	**7**			**7**			**7**			**7**		
Prepare 10min 6		**10**			**10**			**10**			**10**	
Cook 15min 2			**15**			**15**			**15**			**15**

Total time is (7+10+15)x4 = **128 minutes**

Pipelined

Read 7min 1	**7**	**7**	**7**	**7**		
Prepare 10min 6		**10**	**10**	**10**	**10**	
Cook 15min 2			**15**	**15**	**15**	**15**

Total time pipelined although not easily distinguished from graph above (more intricate graph in next page) = **54 minutes**

b) Pipelining improves performance by allowing multiple tasks to be running in a more efficient manner. As seen above each task prior to pipelining would run at set stages, with the introduction of the extra resources of multiple knives and cooktops pipelining these to overlap decreases time taken by more than half. A further description to why is put into part c.

c) As seen, it reduces latency of all the processes as can read the book one after each other rather than waiting for whole process to complete the same goes for the other 2 processes although this can reduce the overall time of all 4 it does not however change the overall time for 1 task to be complete henceforth overall latency is improved individually, singularly it has increased times of 2 of the 4 increasing these. The addition resources help to overlap to have all stages working in sync could theoretically get away with having 2 knives. As seen in the more elaborate table below there are gaps in the process of preparing and cooking with the 2nd group to coming through there are unused slots for preparation with the 2 knives of the 6 being available for use at these stages; alongside this the 3rd and 4th cooker must wait 1 minute prior to having a cooker available.

Minutes	Read	Prepare Knife 1	Prepare Knife 2	Cooker 1	Cooker 2
1	🟨				
2	🟨				
3	🟨				
4	🟨				
5	🟨				
6	🟨				
7	🟨				
8	🟦	🟨			
9	🟦	🟨			
10	🟦	🟨			
11	🟦	🟨			
12	🟦	🟨			
13	🟦	🟨			
14	🟦	🟨			
15	🟧	🟨	🟦		
16	🟧	🟨	🟦		
17	🟧	🟨	🟦		
18	🟧		🟦	🟨	
19	🟧		🟦	🟨	
20	🟧		🟦	🟨	
21	🟧		🟦	🟨	
22	🟩	🟧	🟦	🟨	
23	🟩	🟧	🟦	🟨	
24	🟩	🟧	🟦	🟨	
25	🟩	🟧		🟨	🟦
26	🟩	🟧		🟨	🟦
27	🟩	🟧		🟨	🟦
28	🟩	🟧		🟨	🟦
29		🟧	🟩	🟨	🟦
30		🟧	🟩	🟨	🟦
31		🟧	🟩	🟨	🟦
32			🟩	🟨	🟦
33			🟩	🟧	🟦
34			🟩	🟧	🟦
35			🟩	🟧	🟦
36			🟩	🟧	🟦
37			🟩	🟧	🟦
38			🟩	🟧	🟦
39				🟧	🟦
40				🟧	🟩
41				🟧	🟩
42				🟧	🟩
43				🟧	🟩
44				🟧	🟩
45				🟧	🟩
46				🟧	🟩
47				🟧	🟩
48					🟩
49					🟩
50					🟩
51					🟩
52					🟩
53					🟩
54					🟩

Task 6 CPU Architecture

Modern GPU and CPU selected for this part are AMD Ryzen 9 5900X and AMD Radeon RX 6900 XT (ref 1 and ref 2)

	AMD Ryzen 9 5900X	AMD Radeon RX 6900 XT
Number of Processor cores	12	80 compute units= 5120units
Number of Registers	16	16000? These varied chose the number most repetitive.
Amount and Number Cache	Total L2 Cache 6MB Total L3 Cache 64MB	Infinity Cache 128 MB

b) Threads are used to split the process into multiple tasks allowing for quicker processing. So if you were to allocate a process to a multi thread CPU it could split these into multiple smaller sections allowing for the execution to occur in a quick manner. Typical configuration will see 2 threads per core, the more threads a unit has will allow for larger segments to be completed a good example of this is the AMD Ryzen thread ripper 3990x which has 64 cores and 128 threads. A breakdown of this on Linus tech tips shows the performance increase this can have when rendering videos a very heavy computing task. (Ref 3)

c) A warp is the execution of a compute unit on the GPU, it is a collection of threads that are executed simultaneously. To my understanding the main constraint of a warp is that each will execute the same command not allowing threads to be optimized with other commands if the entirety of the warp is not needed for that process also known as Warp divergence.

Task 7 Memory

a)

Going off research to find clock speeds of GDDR5 and GDDR6 I am hoping my understanding of these equations are correct:
GDDR5 – 8Gbps transfer speed, gddr5 is Quad so the effective clock speed would be 8000/4 which is equal to 2000MHz
GDDR6 – 14-16Gbps transfer speed, gddr6 is quad then doubled. Working off the max value of 16Gbps with the equation 16000/4/2 which is equal to 2000Mhz.
So looking at the figures above we can see the maximum clock speeds on both are equivalent the only difference being gddr6 is dual channel allowing for the 16Gbps speeds. Once again finding the relevant information about clock speeds on GDDR was not in terms easy to find so my research relied heavily on a message board that but it in these terms. (Ref 4)

b)

Although asking for the throughput information on this was carrying depending on the GPU at which the GDDR was installed so henceforth I will give maximum bandwidth as you can defer from this the throughput will be less that these. Working on a bus width of 256 for gddr5 and for gddr6 384-bit bus. Given the equation in a we can surmise that (2 *256)/8*4 = 256GB/s for gddr5 and (2*512)/8*8 = 768GB/s for gddr6. Note that the multiplier for dggr6 is 8 since it is dual channel.

c)

i) Bus size of DDR5 6400MHz is 64 (ref 5) while as stated previously the gddr6 comes in 384 bits.

ii) From my understanding the difference in these are due the bank structures within the groupings with the ddr5 having 32-bank structure while gddr6 has 16-bank structures. I guess to point out the obvious ddr5 is system and gddr6 is graphics.

d)

The difference between HBM and gddr6 is the clock speeds and the bus, to elaborate on this HBM operates at 500Mhz as opposed to the 2000Mhz but to make up for the low clock to keep up with the lower bandwidth has a tremendously bigger bus usually 4096 bits as opposed to the 384bit of gddr6 even with a 1024-bit HBM the bandwidth is >100GB/s. Is this an advantage? I do not know since will have to wait and find out since most current GPUs do not use it at this stage currently not as it is only advantageous if it is put into practice.

Task 8 Hamming & SECDED Code

a)

4 bits has the equivalent of 15 total which is determined by 2^4-1 for every bit we need 1 for hamming so 15-4 will give us a total of 11 bits which can be protected.

b)

$BE4_{16}$ to decimal =
($11x16^2$) + ($14x16^1$) + ($4*16^0$)=
2816 + 224 + 4 =
3044 to binary=

Div. by 2	Quotient	Remainder digit
3044	1522	0
1522	761	0
761	380	1
380	190	0
190	95	0
95	47	1
47	23	1
23	11	1
11	5	1
5	2	1
2	1	0
1	0	1

Binary number is 1011 1110 0100

11	10	9	8	7	6	5	4	3	2	1	0	position
D7	D6	D5	P8	D4	D3	D2	P4	D1	P2	P1	P0	Data/Parity
1	0	1	1	1	1	1	0	0	1	0	0	Data
1	0	1	1	1	1	1	0	0	1	0	0	P0
1		1		1		1		0		0		P1
1	0			1	1			0	1			P2
				1	1	1	0					P4
1	0	1	1									P8

P0 all bits = 7 so is odd
P1 = 4 so is even

P2 = 4 so is even
P4 = 3 so is odd
P8 = 3 so is odd

With the previous working out we can see that there was indeed an error not only has P0 reported as error since we are indeed using even parity; furthermore both P3 and P4 had errors.

11	10	9	8	7	6	5	4	3	2	1	0	position
D7	D6	D5	P8	D4	D3	D2	P4	D1	P2	P1	P0	Data/Parity
1	0	1	0	1	1	1	1	0	1	0	1	Data
1	0	1	0	1	1	1	1	0	1	0	0	P0
1		1		1		1		0		0		P1
1	0			1	1			0	1			P2
				1	1	1	1					P4
1	0	1	0									P8

The following corrections were made P4 was made 0 as the additional did not need to be made, P3 was changed as it needed 1 to be added in order to be even, thus forth changed P0 to suite the new command line giving us 101011110101. These were on the stipulation that all overlapping data was correct but did not punch out the correct parity. Or if this is not allowed the data cannot be fixed as there are no common bits the error shares, I have tried the methods showing the P1P2P4P8 = in this case 1010 or the 10th bit if we flip this we still have an error on P4 total so fixing 1 error would only have more therefore is not possibly fixable.

The ASCII character for this being unable to get as it is 12 long? However if I were to take this back to data only we would have 1011110 which is / in ASCII/UTF-8 or ^ in ASCII.

I am aware the previous may be wrong this was my understanding of the study at the time, time permitting I would look more into this however that is not the case.

Task 9.1 Half-precision Floating-point format

a)

Since the usual single precision of 32 bit is typically an 8 bit exponent and 23 bit mantissa with 1 bit as point mark I have decided that my half precision would be approx. half of these values with a 5 bit exponent and 10 bit mantissa allowing 1 bit for the sign 5 bits for value and 10 bits for fraction. The following table depicts the format in which I will use.

Bit	Depiction	Value
1	+ or -	0 = + , 1 = -
2	2^4	0 = 0, 1 = 16
3	2^3	0 = 0, 1 = 8
4	2^2	0 = 0, 1 = 4
5	2^1	0 = 0, 1 = 2
6	2^0	0 = 0, 1 =1
7	$2^{1/2(1)}$	0=0, 1=0.5
8	$2^{1/2(2)}$	0=0, 1=0.25
9	$2^{1/2(3)}$	0=0, 1 = 0.125
10	$2^{1/2(4)}$	0=0, 1 = 0.0625
11	$2^{1/2(5)}$	0=0, 1 = 0.03125
12	$2^{1/2(6)}$	0=0, 1 = 0.015625
13	$2^{1/2(7)}$	0=0, 1 = 0.0078125
14	$2^{1/2(8)}$	0=0, 1 = 0.00390625
15	$2^{1/2(9)}$	0=0, 1 = 0.001953125
16	$2^{1/2(10)}$	0=0, 1 = 0.0009765625

The following format will give me a range of + 31.9990234375 to – 31.9990234375 I opted for the sign bit over having an extra range as this would allow for extra precision on lower numbers if for example, I would have 2^5 rather than a sign bit I would have in equivalence the exact same precision points but would not be able to depict a negative number. The range and the precision is depicted by the size of the mantissa which is why I have chosen the previously stated 5 and 10 if for example I were to switch the exponent and the mantissa I would only have half the precision but double the range. Depending on what is more important at the time would equate to how you chose to do the floating point if more range is needed you would shorten the mantissa if accuracy is more important you would extent the mantissa this is viable for any bit length you use be it 16 32 or 64 also if minus values are required the sign bit should be implemented but if no negative is required can extend the positive range theoretically; this does not change the actual range however as the sign bit will have the same range.

b)

Student number is 3871491

A = 1 B = 9

X = A+ (B/10)

= 1+(9/10)

X = 1.9

y = B/(a*10)

= 9/(1*10)

= 9/10

= 0.9

X = 1.9 y = 0.9

Conversion	Divide 2	Remainder	
1	0	1	
Conversion	Times 2	Whole	Remainder
0.9	1.8	1	0.8
0.8	1.6	1	0.6
0.6	1.2	1	0.2
0.2	0.4	0	0.4
0.4	0.8	0	0.8
0.8	1.6	1	1.6

Can see from the above the 0.9 is now repeating so can conclude the that the coding will repeat from this process since x and y both have 0.9 can conclude the binary for both will have the same process therefore using the 16 bit I had previously stated and has 5 bits before repeating since the mantissa is 10 long the binary will repeat twice therefore.

X = 000001.1110011100
which would make x
(+,
$0*2^{4},0*2^{3},0*2^{2},0*2^{1},1*2^{0}.,1*^{2(1/2)1},1*2^{(1/2)2},1*2^{(1/2)3},0*2^{(1/2)4},0*2^{(1/2)5},1*2^{(1/2)6},1*2^{(1/2)7},1*2^{(1/2)8},0*2^{(1/2)9},0*2^{(1/2)10}$)

X = + + 0 + 0 + 0 + 0 + 1 + 0.5 + 0.25 + 0.125 + 0 + 0 + 0.015625 + 0.007815 + 0.00390625 + 0 + 0

X = 1.90234375

Y= same as previous state without the 1 so 0000001110011100

Y = 0.90234375

Addition in binary

 0000011110011100

+0000001110011100

Value 1	Value 2	Total and carry
0	0	1
1	0	2 = 0 bin with 1 carry
1	1	3 = 1 bin 1 carry
1	1	3 = 1 bin 1 carry
1	1	2 = 0 bin 1 carry
0	0	0
0	0	1
1	1	3 = 1 bin 1 carry
1	1	3 = 1 bin 1 carry
1	1	2 = 0 bin 1 carry
0	0	0
0	0	0

Addition is 10.1100111000

So 10 is 2

1100111000 is $1^{*2(1/2)1}$,$1*2^{(1/2)2}$,$1*2^{(1/2)3}$,$0*2^{(1/2)4}$,$1*2^{(1/2)5}$,$1*2^{(1/2)6}$,$1*2^{(1/2)7}$,$0*2^{(1/2)8}$,$0*2^{(1/2)9}$,$0*2^{(1/2)10}$

= 0.5+0.25+0+0+0.0325+0.015625+0.0078125+0+0+0

= 2.8059375

Missed the step but worked out in head to determine the binary to the decimal we must find the exponent bias in this case we have 1 sign value 5 exp and 10 mantissa. Bias is 2^{5-1}-1 = 16-1 = 15_{10}

I am pretty sure I have missed a few steps here however I believe my working our is still sound conversion of a floating to pure binary eludes me at this stage and have run out of time.

(iv)

Original precession

X= 1.9 floating point = 1.90234375

Y = 0.9 floating point = 0.90234375

Original difference 0.00234375 and 0.00234375 if we add both these the original 2 numbers were out by 0.0046875.

Addition of these numbers result in 2.8059375.

1.9 + 0.9 should equal 2.8 with the floating point we can see we are out by 0.0059375 a further 0.001286 away from the original number so we have after addition resulted in being further away and less precise. This all comes down to the original numbers since we could not represent them in full and had extra when changed you would expect to see extra of the extra due to addition rounding to 1 decimal place the numbers are still sound however this shows a downside to 16-point precision.

c)

When adding a large volume of floating-point numbers, it is best to work from the smallest to the largest. Since adding numbers will always prove a variance as previous stated adding in this way will provide the smallest deviation since you are adding the more precise numbers first giving a more precise answer. For example, using my table we will use the numbers 11.9 + 1.9 + 0.9 + 0.09 with the answer of 14.79 to be desired. The working for the codes were done on by hand refer to references (ref 6) for image.

To do them in this way would result in 0001011.1110011100 + 000001.1110011100 + 000000.1110011100 + 000000.0001011100.

So in hindsight we have 11.90234375+1.90234375+0.90234375+0.08984375

To add in binary, we would achieve largest to smallest

First	Second	Total
001011.1110011100	000001.1110011100	001101.1100111100
001101.1100111100	000000.1110011100	001110.1011011000
001110.1011011000	000000.0001011100	001110.1100110100

Answer being 001110.1100110100 or
8+4+2+0.5+0.25+0.03125+0.015625+0.00390625
= 14.80078125

Difference in this case from desired is 0.01078125

Alternatively smallest to largest

First	Second	Total
000000.0001011100	000000.1110011100	000000.1111111000
000000.1111111000	000001.1110011100	000010.1110010100
000010.1110010100	001011.1110011100	001110.1100110000

Answer being 001110.1100110000 or
8+4+2+0.5+0.25+0.03125+0.015625
= 14.796875

Difference in this case from desired is 0.006875

From the following we can see by adding from smallest to largest has resulted in being 0.00390625 more precise to the actual number in which we original required.

Task 9.2 Logic Simplification

After consideration I am hoping I am correct in my assessment that task 9 was to complete either part for the original out of 250 mark as apposed to the rubric out of 300 as the rubric was released late the time was not viable. I have put in this section as I was going to try however a lot more research into this as apposed to time available is required to put forth any real attempt. I am hoping my previous work is sufficient to see this assignment through and on a further note without knowing if my Logic circuit equation is correct, I also find this hard to achieve I would hate to produce work based on an equation that is incorrect to begin with making all further work incorrect. To be submitted by due date I must miss this step however study will go into these as experience for future development when the timing permits. If both tasks in question 9 were meant to be complete I apologise however this could be made more clear in the future.

Bibliography

Ref 1

Database, G. and Specs, R., 2021. *AMD Radeon RX 6900 XT Specs*. [online] TechPowerUp. Available at: <https://www.techpowerup.com/gpu-specs/radeon-rx-6900-xt.c3481> [Accessed 17 July 2021].

Ref 2

2021. [online] Available at: <https://www.amd.com/en/products/ryzen-threadripper> [Accessed 17 July 2021].

Ref 3

Sebastion, L., 2021. *It's hard to watch, but I can't look away - Threadripper 3990X*. [online] Youtube.com. Available at: <https://www.youtube.com/watch?v=1LaKH5etJoE> [Accessed 17 July 2021].

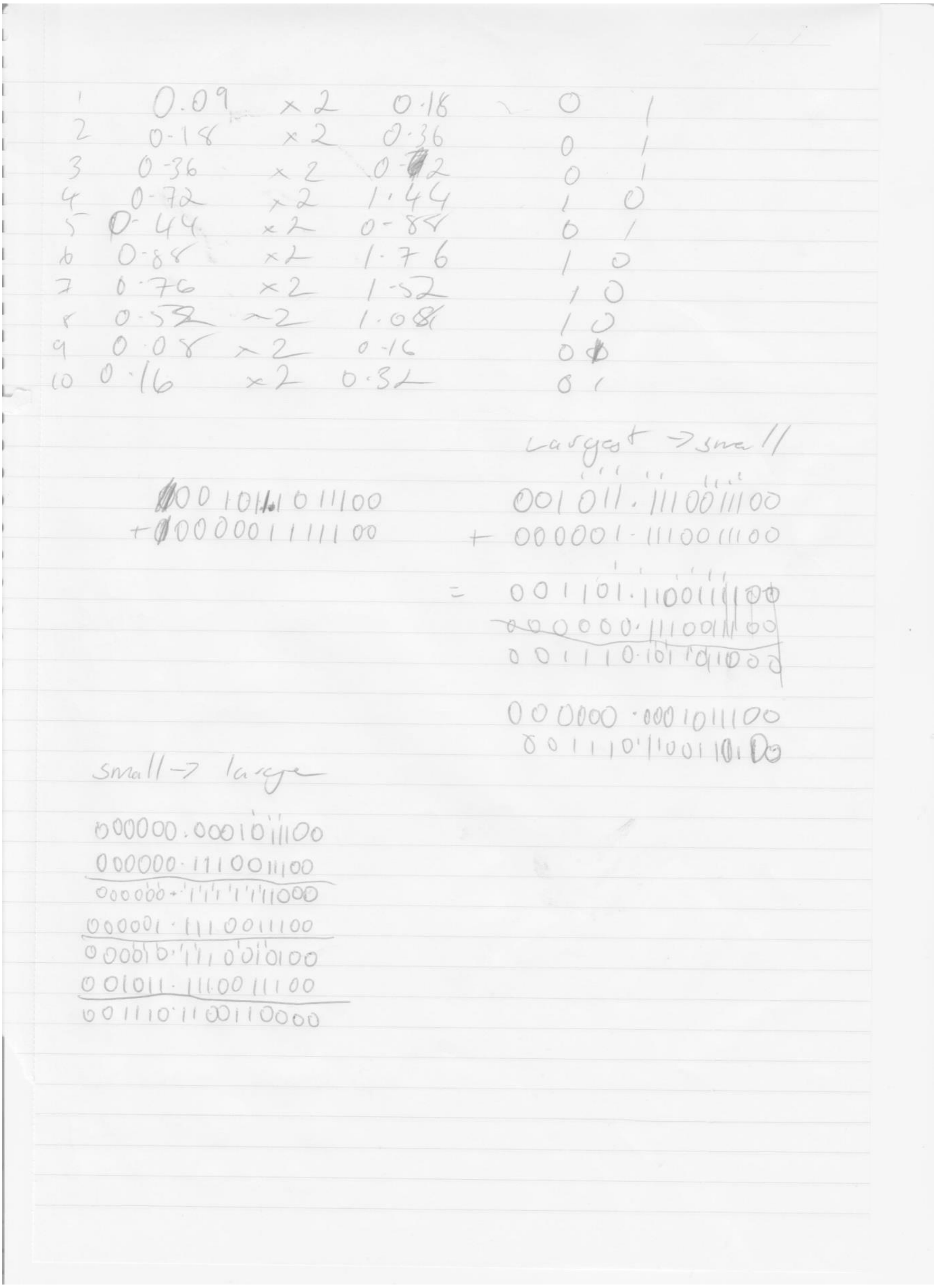
Ref 4

Cards, G., Software, P., Messiah., M., Messiah., M., Software, P., Messiah., M., Software, P. and Messiah., M., 2021. *[SOLVED] - Effective memory clock speed confusions*. [online] Tom's Hardware Forum. Available at: <https://forums.tomshardware.com/threads/effective-memory-clock-speed-confusions.3518637/> [Accessed 17 July 2021].

Ref 5

Smith, R., 2021. *DDR5 Memory Specification Released: Setting the Stage for DDR5-6400 And Beyond*. [online] Anandtech.com. Available at: <https://www.anandtech.com/show/15912/ddr5-specification-released-setting-the-stage-for-ddr56400-and-beyond> [Accessed 17 July 2021].

Ref 6



1	0.09	×2	0·18	0	1
2	0·18	×2	0·36	0	1
3	0·36	×2	0·72	0	1
4	0·72	×2	1·44	1	0
5	0·44	×2	0·88	0	1
6	0·88	×2	1·76	1	0
7	0·76	×2	1·52	1	0
8	0·52	×2	1·08	1	0
9	0·08	×2	0·16	0	0
10	0·16	×2	0·32	0	1

Largest → small

000 1011·1 011100
+ 000000111111 00

001011.1110011100
+ 000001·1110011100
= 001101.1100111100
000000·1110011100
001110·101010000

000000 ·0001011100
001110·1100110100

small → large

000000.0001011100
000000·1110011100
000000·1111111000
000001·1110011100
000010·1110010100
001011·1110011100
001110·1100110000