Name: [redacted] ID: [redacted]

85

COMP5201 Assignment 2 Fall 2020

Issued: September 29, 2020 Due: October 20, 2020

Submit electronically. No extension will be granted.

1. [24 marks] Digital Logic.

a) 'X' is the ternary connective such that 'Xpqr' is logically equivalent to '$p \wedge (p + q + r)$'. '+' is 'xor'. 'F' and 'T' denote the 0-ary connectives 'false' and true, respectively. Whenever possible, put letters in alphabetical order, and put letters before any 0-place connectives.

Using {'X', 'F', 'T'}, synthesize: ~p |= =| X(T p F)

p	~p	X(TpF)
0	1	X(100) = 1
1	0	X(110) = 0

Using {'X', 'T'}, synthesize: $p \wedge q$ |= =| X(p q T)

p	q	p and q	X(pqT)
0	0	0	X(001)=0
0	1	0	X(011)=0
1	0	0	X(101)=0
1	1	1	X(111)=0

Using {'X', '~', 'T'}, synthesize: $p \vee q$ |= =| ~X(~p ~q T)

p	q	q or p	~X(~p~qT)
0	0	0	X(111) = 0
0	1	1	X(101) = 1
1	0	1	X(011) = 1
1	1	1	X(011) = 1

b) 'Y' is the ternary connective such that 'Ypqr' is logically equivalent to 'p /\ (q <--> r)'. 'p <--> q' is true iff 'p' and 'q' have the same truth value. 'F' and 'T' denote the two 0-ary connectives 'false' and 'true', respectively. Whenever possible, put letters in alphabetical order, and put letters before any 0-place connectives.

Using {'Y', 'F', 'T'}, synthesize: ~p |= =| Y(TpF)

p	~p	Y(pqT)
0	1	Y(100) = 1
1	0	Y(110) = 0

Using {'Y', 'F', 'T'}, synthesize p /\ q |= =| Y(pqT)

p	q	p and q	Y(pqT)
0	0	0	Y(001)=0
0	1	0	Y(011)=0
1	0	0	Y(101)=0
1	1	1	Y(111)=0

Using {'Y', '~', 'F'}, synthesize: p \/ q |= =| ~Y(~pqF)

p	q	p and q	Y(~pqF)
0	0	0	Y(100)=0
0	1	1	X(110)=1
1	0	1	X(000)=1
1	1	1	X(010)=1

2. [16 marks] Binary and Hexadecimal Numbers.

Convert each of the following six binary or hexadecimal natural numbers into decimal. Show work. For the last number, show either scientific notation with six decimal places, or the exact integer.

a) Binary numbers: 1111, 1111 1110, 1111 1110 1101 0111

signed:	-7	-126	-32 471
unsigned:	15	254	65 239

1111:

each digit represents an increasing power of 2:

4 bits: 2^4 -> We can represent up to 16 values

The values go from (n/2)-1 to (n/2)-1 with n = 16: [-7:7]

Most significant bit is 1 so if it's signed, it's a negative value.

The other bits are 111, so it's the highest negative value: -7

If it's an unsigned number (no negative values) it's n-1: 15

1111 1110:

8 bits: 2^8 -> 256 values:

if signed, values are in: [-127:127]

if unsigned, values are in [0:255]

First bit is 1, and all but the least significant bit are 1:

it's the -[n-1] value where n is the lowest value possible: -(127-1) = -126

if it's usigned, it's the n-1 higest value possible: 255-1: 254

1111 1110 1101 0111:

16 bits: 2^16 -> the values are in [0:65 535]

1111 1110 1101 0111 = 1111 1111 1111 1111 - 0000 0001 0010 1000

0000 0001 0010 1000 // from previous question we know that 1 0000 0000 = 256

= 256 + 0010 1000

= 256 + 32 + 8

= 296

----> 1111 1111 1111 1111-0000 0001 0010 1000

is equal to 65 535 - 296 = 65 239

if signed values:

1111 1111 1111 1111 = (65 536/2)-1 = -32 767

- (32 767 - 296) = -32 471

____

b) Hexadecimal numbers: af, afba, afba 51de

____

af:

a = 10 * (16 ^ 1) = 160

f = 15 * (16 ^ 0) = 15

af = a + f = 160 + 15 = 175

afba:

a = 10 * (16 ^ 3) = 40 960

f = 15 * (16 ^ 2) = 3 840

b = 11 * (16 ^ 1) =   176

a = 10 * (16 ^ 0) =   10

afba = a + f + b + a = 44986

afba 51de:

a = 10 * (16 ^ 7) = 2 684 354 560

f = 15 * (16 ^ 6) =   251 658 240

b = 11 * (16 ^ 5) =   11 534 336

a = 10 * (16 ^ 4) =   655 360

afba 0000 =   2 948 202 496

5'= 5 * (16 ^ 3) =   20 480

1'= 1 * (16 ^ 2) =   256

d = 13 * (16 ^ 1) =   208

e = 14 * (16 ^ 0) =   14

51de =   20 958

afba51de = 2 948 202 496 + 20 958 = 2 948 223 454

3. [25 marks] Fractional Numbers and Blackboard Notation.

Infinite binary expansions of rational numbers are either pure recurring or mixed recurring depending on whether the cycle starts immediately after the point.

This exercise is done assuming 6 4/9 means - 6 is the integer part

and 4/9 the fractional part

---

a) [math] Show the infinite binary expansion of 6 4/9 without normalization.

---

6 in binary is 110

6 4/9 is equal to 58/9 which is an endless series of 4 after the .

An approximation of 6 4/9 equal like 6+2^-2+2^-3+2^-4+2^-8+2^-9+2^-10+2^-14+2^-15+2^-16

gives: 6.4444427490234375 in decimal and 110.0111000111000111 in binary

There is a regularity in the alternation between the 1s and 0s

and since the decimal expression of 58/9 has no end, so does its binary expression

---

b) [math] Show this infinite binary expansion in hexadecimal without normalization.

With the longer decimal expression:

6+2^-2+2^-3+2^-4+2^-8+2^-9+2^-10+2^-14+2^-15+2^-16+2^-20+2^-21+2^-22+2^-26+2^-27+2^-28+2^-32+2^-33+2^-34+2^-35

we get a better approximation: 6.444444444438

Let's convert the integer part to hexadecimal:

6 -> 6

Let's convert the fractional part to hexadecimal: int part -> hex

0.444444444438	x	16	=	7.111111111008	7
0.111111111008	x	16	=	1.777777776128	1
0.777777776128	x	16	=	12.444444418048	C
0.444444418048	x	16	=	7.111110688768	7
0.111110688768	x	16	=	1.777771020288	1
0.777771020288	x	16	=	12.444336324608	C
0.444336324608	x	16	=	7.109381193728	7
0.109381193728	x	16	=	1.750099099648	1
0.750099099648	x	16	=	12.001585594368	C

......

We get 6.444444444438 is ~6.71C71C71C in hexadecimal.

Just like in the previous question we can see a repetition pattern for an infinite expansion

---

c) [math] Show the infinite binary expansion of 6 4/9 with normalization. Do not forget the scale factor. (This is a _scaled_ infinite binary expansion).

-2

(110.0111000111000111) * 2^0 <identity> // for comparison

(1.100111000111000111) * 2^2 <normalized>

---

d) [math] Show this infinite binary expansion in hexadecimal. Again, do not forget the scale factor, which may be shown in decimal.

-5

1.8<71C> 2^2 <normalized>

e) Show the normalized (binary) blackboard notation that best approximates 6 4/9. The fractional field is 16 bits. Show all 16 bits. Now, show just the 16-bit (4-hexit) fractional field in hexadecimal.

____

in 16bits:

1.10<0111000111000111> [2] <"lazy">   // for comparison

1.10<0111000111000111> 2^2 <normalized>

Fractional field in hexadecimal:

0111 0001 1100 0111 -> <71C7>

4. [15 marks] Integer Multiplication I.

a) Multiply the following 10-bit binary natural numbers. The multiplicand is 10011 11100 (27c hex) and the multiplier is 11010 (1a hex). Show, in hexadecimal, i) the initial value of the accumulator, and ii) each term added to the accumulator, and the partial sum after the addition. The last addition yields the final value.

____

```
          carry

          a---->      +1     +4  +7

          1---->     +0   +0  +0

 addition---->     +1   +1

                    ----------------------

sum       =      2   5   7


                      16^3 16^2 16^1 16^0

                              2   7   c                1) a*c = 10*12 = 120 = 7*16+8

                              *   1   a                2) a*7 = 10*7  =  70 = 4*16+6

                    ----------------------             3) a*2 = 10*2  =  20 = 1*16+4

          a * 27c =           4   6   8                4) 1*c = 13 = c

          1 * 27c =    2   7   c              5) 1*7 =  7

                    ----------------------             6) 1*2 =  2

           result =    2   b   2   8          7) 6+c = 18 = 1*16+2

                                                       8) 4+7 = 11 = b

         acc      =    2   5   7              9) 2+7 = 9

          ------------------------------          10) b+5 = 1*16 + 0

2b28 + acc    =    4   0   9   8        11) 2+2 = 4
```

b) Redo the multiplication steps exactly as in question 4 a), but initialize the accumulator to s = 11011 (1b hex) instead of 0. Show the same intermediate and final values. (This is called "fused multiply-add").

____

carry

initialized--> +1 +b

a----> +1 +4 +7 | Same as

1----> +0 +0 +0 | previous

addition----> +1 +1 | question

----

sum = 2 5 8 b

16^3 16^2 16^1 16^0

2 7 c 1) a*c = 10*12 = 120 = 7*16+8

* 1 a 2) a*7 = 10*7 = 70 = 4*16+6

---------------------- 3) a*2 = 10*2 = 20 = 1*16+4

a * 27c = 4 6 8 4) 1*c = 13 = c

1 * 27c = 2 7 c 5) 1*7 = 7

---------------------- 6) 1*2 = 2

result = 2 b 2 8 7) 6+c = 18 = 1*16+2

8) 4+7 = 11 = b

accumulator = 2 5 8 b 9) 2+7 = 9

------------------------------- 10) b+5 = 1*16 + 0

2b28 + acc = 4 0 B 3 11) 2+2 = 4

12) b+8 = 19 = 1*16 + 3 (accumulator)

13) 2+8+1 (from 12) = 11 = B

14) b+5 = 16 = 1*16 + 0

15) 2+2 = 4

____

5. [20 marks] Integer Multiplication II.

a) Show that, regardless of the initial n-bit value of the accumulator, the fused multiply-add result of two n-bit natural-number operands is always representable in 2n bits. Now, suppose n = 16. Starting from the largest possible FMA result, what is the hexadecimal representation of the largest n = 16-bit number that can _still_ be added without producing overflow?

____

If we're looking for the largest possible accumulator value that can be used without producing overflow:

For a 16 bits bit register:

1111 1111 1111 1111 // 65 535

*1111 1111 1111 1111 // 65 535

= 1111 1111 1111 1110 0000 0000 0000 0001 // 4 294 836 225

The value x must be such that x < 2^n-1 to avoid overflow, the total value must be less

than 2^32-1: 4 294 967 295

The difference between the max value and the value we can have by multiplying 6535*6535 is:

4 294 967 295 - 4 294 836 225 = 131 070

131070 > 2^16 => The largest 16 bits number still fit without producing overflow

=> 1111 1111 1111 1111 (binary) or FFFF (hexadecimal)

____

b) A modular-adder device 'M' operates with 16-bit registers. You give it two 16-bit natural numbers 'a' and 'b'. It adds them, divides by 2^16, keeps the quotient 'q' a secret, and publishes the remainder 'r'. Hint: Before answering, experiment with small addition tables.

i) If a = 31,465 and r = 53,576, what are 'b' and 'q'?

____

to binary:

a -> 0111 1010 1110 1001

r -> 1101 0001 0100 1000

solving: {

(31465 + b)/(2^16) = q

(31465 + b)%(2^16) = 53576

}

solution: {

z = |53576 - 31465| // absolute value

b = z + n*65536

q is a function of b

}

assuming n = 0 we have:

b = r - a <==> 1101 0001 0100 1000 - 0111 1010 1110 1001

b -> 0101 0110 0101 1111

b = 22 111

a + b < 2^16 => q = 0

assuming n = 1 we have:

b = 53576 - 31465 + 65536 = 87 647

b -> 1010 1011 0010 1111 1 --> doesn't fit 16 bits address

===> We can't have any other solution than b = 22 111 so q = 0

____

ii) If a = 35,492 and r = 11,087, what are 'b' and 'q'?

____

to binary:

a -> 1000 1010 1010 0100

r -> 0010 1011 0100 1111

solving: {

(35492 + b)/(2^16) = q

(35492 + b)%(2^16) = 11087

}

solution: {

b = 41 131 + n*65536

q is a function of b

}

assuming n = 0 we have:

b = 41 131

b -> 1010 0000 1010 1011

a+b > 2^16

(a+b)/(2^16) = 1 // division

(a+b)%(2^16) = 11087

-3

b is indeed 41131

assuming n = 1 we have:

b = 41 131 + 65 536 = 106 667

b -> 1101 0000 0101 0101 1 --> doesn't fit 16 bits address

===> We can't have any other solution than b = 41 131