Tharishka Gamage

CS 2011 - HW 1

M14304830

(A)

1) 0x1C = 0000 0000 0000 0000 0000 0000 0001 1100

Two's Complement 1111 1111 1111 1110 0011 +1

= 1111 1111 1111 1111 1111 1111 1110 0100 = -28 =

0xFFFFFFE4

2) 30 = 0000 0000 0001 1110

additive inverse +1

= 1111 1111 1110 0001 + 1

1111 1111 1110 0010

= -30 = 0xFFE2

3) 0x8F = 1000 1111

= 1111 1111 1111 1111 1111 1111 1111 1111 1111

1111 1111 1111 1111 1111 1000 1111

0xFFFFFFFFFFFFFF8F

(B)

X = X & FF

Y = Y & FFFF FF00

Y += X

Return = Y

explain

adding to x by FF filters if by the last 2 hex values adding y by FFFFFF000 filters last 2 hexs of y to be 00 → if Y = x+y = y = desired value

(C)

1) Void replace_byte_inline (ussigned char x, int, usigned char) {

*(x+i) = b;

} Replace the byte by using the pointer to shift and change char x.

2) unsigned replace_byte (unsigned x, int i, unsigned char b) {
return | xd in (0xFF << (i<<3)) | (b << (i<<3)));
} Removes all the bits except the desired one, and then swap i to the char b.

(D)

1) The error is that the last 8 bits will be returned, due to the & 0xFF.

2) int extraSignedByte(unsigned word, int byte_number) {
word = ~(word ^ ((3 - byte.number) << 3)) << 24) >> 24
return, word;

E

1) num = (x<<4) + x;
2) num = x - (x<<3);
3) num = (x<<5) - (x<<2)
4) num = (x<<4) - (x<<7)

Shifting by 1 is the same as multiplying a number by a power of two, so doing multiple shifts together achieves multiplication by the desired amount.

F

1) x<y == (-x>-y) is not always 1.
if x = the minimum possible value (ie. 100...000)
the inverse of this is itself, while y would be changed to the opposite sign.

2) ~x + ~y + 1 == ~(x+y)
Always yields 1 because ~x + 1 = -x (two's comp)
=> ~(x+y) + 1 == -(x+y)
=> ~(x+y) + 1 == -x - y
=> ~(x+y) + 1 == ~x + 1 + ~y + 1
=> ~(x+y) = == ~x + ~y + 1

3) (ux - uy) == -(unsigned)(y-x)
Always evals. to 1, as the position of the cast does not matter, as the resulting numbers will still just break down to ux - uy == -(uy - ux), which is true

4) $((x >> 2) << 2) <= x$

Always evaluates to 1, as shifting right to start will only ever cut off numbers, and shifting back left does not return them. If all 1's are 2 bits from the right edge, nothing will change from $(x >> 2) << 2$, but if there are bits these, they will be int off, there decreasing the value (ie $1111 > 1100$), so x will either be equal or less after this operation

5) $((x+y) << 4) + y - x == 17.y + 15.x$

Always: as shifting left 4 is multiplication by 2^4, or 16. Then by adding 4, y is multiplied by 17, ad subtracting an x makes x multiplied by 15: Thus the operation is e and yo $17y + 15x$.

6) $0x(642b200 = 1100\ 0110\ 0100\ 0010\ 1111\ 0010\ 0000\ 0000$

$2^{31} + 2^{30} + 2^{26} + 2^{25} + 2^{22} + 2^{17} + 2^{15} + 2^{14} + 2^{12} + 2^{9}$

$= 3326279 04$ //