Dylan Thornton

1.

A. addi x5,x6,3

$$\overbrace{000000000011}^{3}\ \overbrace{00110}^{x6}\ \overbrace{000}^{funct3}\ \overbrace{00101}^{x5}\ \overbrace{0010011}^{addi}$$

$\boxed{0x00330293}$

B. There are 12 bits for the Immediate

$2^{12} = 4096$

The range is 0-4095, 4096 different immediates
or -2048 to 2047

2,

A.

```
lui x1, 0x1aa0
xori x2,x1,0xffff
```

leading bits allows this to work for bigger immediates too.

B.

assume I did the .data thing

```
la x20, variable
variable:
    .word 0x deadball
```

deadball is too big to be loaded immediately. I put it in mem and loaded from there. This worked in the assembler

You could also do this, but mem loading is faster (to type):

```
lui x21, 0x911
SRLi x21,x21,12
lui x20, 0xdeadb
add x20,x20,x21
```

and then add xori x20,x20,0x4fe to make 0xdeadbeef

C.

0xdeadball = 3735927313
0xdeadbeef = 3735928559

```
la x20, variable
xori x20,x20,0x4fe
variable:
    .word 0xdeadball
```

I believe this would work. The RISC-V resources we were given do not always work so I couldn't prove it, but w/ the leading zeros aspect, it should work

3.

a. #1: addi x8, x0, 3

#2: addi x9, x0, 10 //0xA if in hex

#3: add x8, x8, x8

#4: blt: x8, x9, 4094 // ← I initially got this but when I plugged into an interpreter, it said -4

so its that or this

$\boxed{\text{blt x8, x9, -4}}$

b. just one (#4)

c.

#5: slli x20, x8, 2

x8 = 0 + 3 = 3

x9 = 0 + 10 = 10

x8 = 3 + 3 = 6

blt x9 < 4094

So it branches

0 b/c we branch before shifting by two.

However → would be 24 or 0x18

If the answer is blt x8, x9, -4 then we get sent back an instruction as 6 < 10. x8 now = 12 and is bigger than 10 and thus would move on. It gets shifted to the left twice (multiplied by 2^2) to now have x20 = $12 \cdot 2^2 = 48$.