Problem 1) Given x in r1, y in r2, z in r3, and a in r4 convert the C code to albaCore assembly and then albaCore machine encoding. You may use r5 and r6 for temporary registers if necessary.

```
x = 5;

y = -6;

z = 4;

x = y + z;

a = x + y – z;
```

Solutions:

albaCore assembly

```
ldi r1, 5

ldi r2, -6

ldi r3, 4

add r1, r2, r3

add r5, r1, r2

sub r4, r5, r3
```

albaCore machine encoding

```
0x7105

0x72FA

0x7304

0x0123

0x0512

0x1453
```

Problem 2) Given a variable x in memory at address 0xab4e, convert the C code to albaCore assembly and then albaCore machine encoding. The address of x is stored in r1, a is in r2. and b is in r3. Use r4 and r5 for temp registers if necessary

```
int* x = 0xab4e;

int a = 5;

int b = -1;

*x = a – b;
```

Solutions:

albaCore assembly

```
ldi r1, 0xab

ldi r0, 8

shl r1, r1, r0

ldi r0, 0x4e

or  r1, r1, r0

ldi r2, 5

ldi r3, -1

sub r4, r2, r3

st  r4, r1, 0
```

albaCore machine encoding

```
71ab   # Only have 8 bits to store!

       # r1 now equals 0x00ab

7008   # Need to shift r1 left by 8

5110   # Use ALU to shift r1 8-bit

       # r1 now equals 0xab00

70cd   # r0 now equals 0x004e

       # r1 now equals 0xab4e

7105   # r2 now equals 5

711f   # r3 now equals -1

1423   # r4 now equals 4

9041   # M[r4+0] ← r1
```

Problem 3) Given that x is stored in r1, y is in r2. and z is in r3, convert the C code to albaCore assembly and then albaCore machine encoding. Use r4, r5, and r6 for temp registers if necessary.

```
if (x == 10)
      y = x + z;
else
      y = x - z;
```

Solutions:

albaCore assembly

```
ldi r4, 10
sub r5, r1, r4
bn  r5, IF_BLOCK
add r2, r1, r3
br  RICK
IF_BLOCK:
sub r2, r1, r3
RICK:
```

albaCore machine encoding

```
740A   # load 10 into reg r4
1514   # r5 = r1 – r4
C503   # Branch to IF_BLOCK if r5 < 0
0213   # y = x + z if not taken
A02X   # Go to RICK, X means don’t care

1213   # y = x – z if taken
```