if-else Statement

Convert the following C++ code to LEGv8 Assembly code. Assume the variables f, g, h, i, and j correspond to five registers X19, X20, X21, X22, and X23.

```
if (i == j)
        f = g + h;
else
        f = g - h;
```

c++

i ≠ j

B. NE → Not equal.

B. EQ →

f → X19   g → X20   h → X21   i → X22   j → X23.

```
        i     j
CMP   X22, X23.          // compare i and j
B.NE     else.           // if i ≠ j go to else
ADD   X19, X20, X21      // f = g+h
B exit                   // Go to exit

else: SUB  X19, X20, X21 // f = g-h.
           f    g    h

exit:
      CMP   X22, X23
      B.EQ  L1
      SUB   X19, X20, X21
      B     exit
L1:   ADD   X19, X20, X21
exit:
```

i = i+1

ADDI X22, X22, #1

While loop

Convert the following C++ code to LEGv8 Assembly code. Assume the variables i and k correspond to registers X22 and X24.

```
While (i==k)
{  i = i+1;
}
```

$i \downarrow X22$ $\quad K \downarrow X24$

```
Loop:   CMP   X22, X24          // compare i and J
        B.NE  exit              // if i ≠ K, go to exi
        ADDI  X22, X22, #1      // i=i+1
  →     B     Loop
exit:
```

Convert the following C++ code to LEGv8 Assembly code. Assume the variables i and k correspond to registers X22 and X24. The base address of the array save is in X25.

```
while (save[i]==k)
{
    i=i+1;
}
```

save[i]

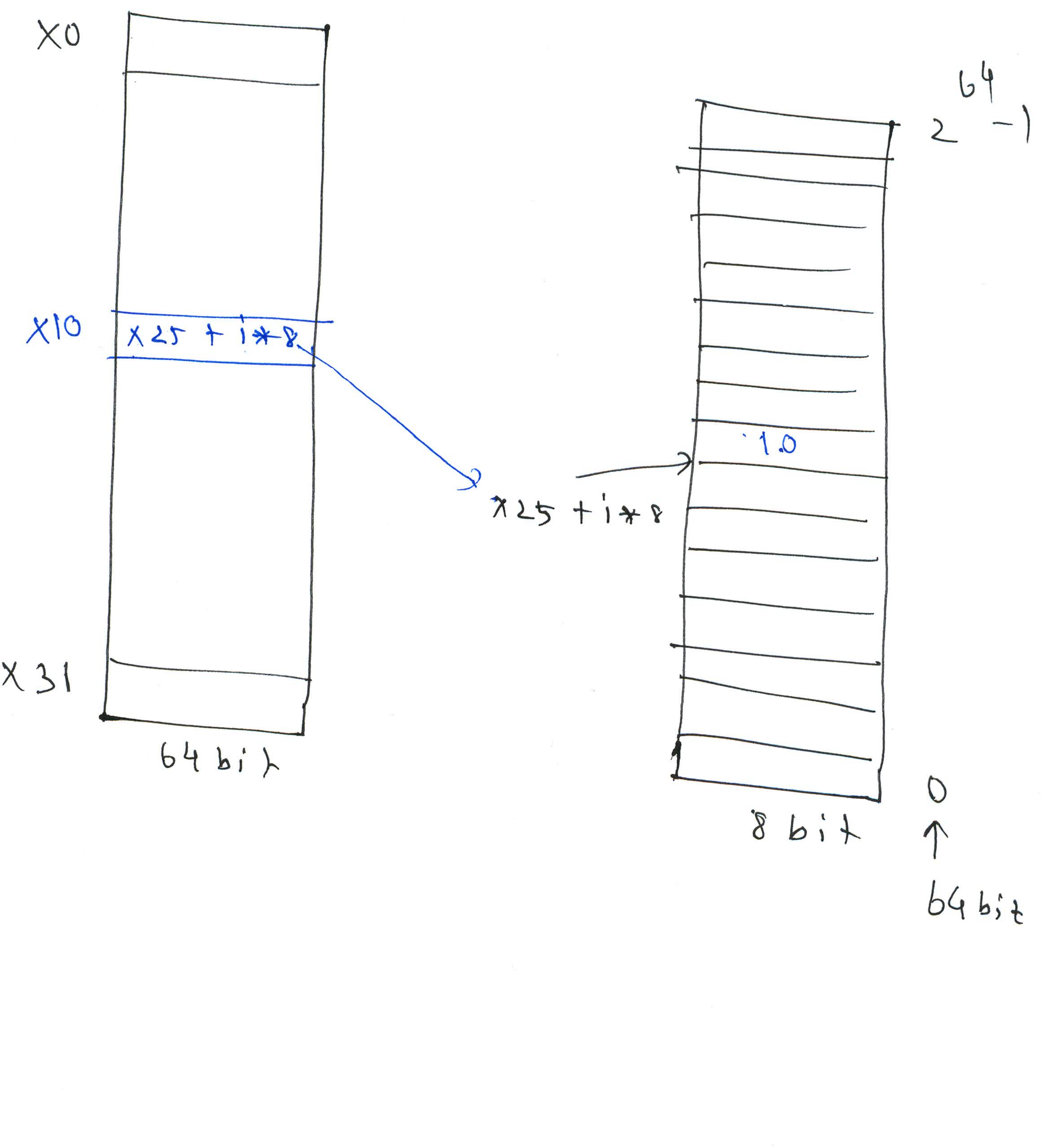
$= \underbrace{X25 + i * 8}$

i → X22   K → X24   save → X25

```
Loop:   LSL   X10, X22, #3       // X10 = i*2^3 = i*8
        ADD   X10, X10, X25      // X10 = X25 + i*8
        LDUR  X9, [X10, #0]      // X9 = save[i]
        ~~SUB~~
        CMP   X9, X24            // compare save[i] and K
        B.NE  Exit
        ADDI  X22, X22, #1       // i = i+1
        B Loop

Exit:
```

$\textcircled{9}$


X0
X10
x25 + i*8
X31
64 bit
$2^{64}-1$
1.0
x25 + i*8
0
8 bit
64 bit

Address of b[i] = X23 + i*8

MOVI X9, #0 // i=0

for Loop

Convert the following C++ code to LEGv8 Assembly code. Assume the variable a is in X22 and base address of array b is in X23.

i >= a   C++

```
for(i=0,i<a,i++)
{
    b[i] = a + i;
}
```

a ↓ X22   b ↓ X23.   i ↓ X9.

~~MOVI X9, #0~~

zero ~~register = 0~~

```
                                  i=0
      ADDI  X9, XZR, #0        // X9 = 0+0 = 0
loop: CMP   X9, X22            // compare i and a
            i   a
      B.GE  Exit               // if i>=a, Go to exit
      ADD   X10, X22, X9       // X10 = a+i
                 a    i
      LSL   X11, X9, #3        // X11 = i*2^3 = i*8
                 i
      ADD   X11, X23, X11      // X11 = X23 + i*8
      ~~ADDI~~
      STUR  X10, [X11, #0]     // b[i] = a+i
      ADDI  X9, X9, #1         // i++
            i   i
      B     Loop
Exit:
                               ~~STUR X~~
                               STUR X10, [X23, X11]
```