Problem 1: In DLX the three instructions below, though they do very different things, are of the same type (format).

```
bnez r2, SKIP       lw r1, 1(r2)   addi r1, r2, #1
```

Because of their similarity their implementations in the diagram below shares a lot of hardware.

(a) Show how these DLX instructions are coded.

(b) Find corresponding instructions in the SPARC V9 ISA. (See the SPARC Architecture Manual V9, [SPARC Architecture Manual V9](http://www.ece.lsu.edu/ee4720/samv9.pdf)) (The DLX branch instruction will have to be replaced by two instructions, one to set the condition code registers.)

(c) Show the coding of the SPARC V9 branch, load, and add immediate instructions (but not the condition code setting instruction).

(d) Do these codings allow the same degree of hardware sharing?
**Problem 2:** Write a DLX assembly language program that determines the length of the longest run of consecutive elements in an array of words. For example, in array \(\{1, 7, 7, 1, 5, 5, 5, 7, 7\}\) the longest run is three: the three 5’s (the four 7’s are not consecutive). The comments below show how registers are initialized and where to place the longest run length.

\[
\begin{align*}
! & \ r10 \ \text{Beginning of array (of words).} \\
! & \ r11 \ \text{Number of elements.} \\
! & \ r1 \ \text{At finish, should contain length of the longest run.}
\end{align*}
\]

**Problem 3:** Small integers can be stored in a packed array to reduce the amount of storage required; the array can be unpacked into an ordinary array when the data is needed. Write a DLX assembly language program to unpack an array containing \(n\) \(b\)-bit integers stored as follows. The low \(b\) bits (bits 0 to \(b - 1\)) of the first word of the packed array contain the first integer, bits \(b\) to \(2b - 1\) contain the next, and so on. When the end of the word is reached integers continue on the second word, etc. Size \(b\) is not necessarily a factor of \(n\) and so an integer might span two words.

The diagram below shows how the first 6 integers \(i_0, i_1, \ldots, i_5\) are stored for \(b = 12\) bits and \(n \geq 6\).

\[
\begin{array}{cccccc}
\text{i}_2 & & \text{i}_1 & & \text{i}_0 \\
\text{low part} & & & & \\
31 & 24 & 23 & 12 & 11 & 0
\end{array}
\]

\[
\begin{array}{cccccc}
\text{i}_5 & & \text{i}_4 & & \text{i}_3 & & \text{i}_2 \\
\text{low part} & & & & \text{high part} \\
31 & 28 & 27 & 16 & 15 & 4 & 3 & 0
\end{array}
\]

Write DLX assembly language code to unpack such an array into an array of signed words. The packed array consists of \(n\) \(b\)-bit signed numbers, with \(b \in [1, 32]\). Initial values of registers are given below.

\[
\begin{align*}
! & \ \text{Initial values} \\
! & \ r10: \ \text{Address of start of packed array.} \\
! & \ r11: \ \text{Number of elements (n).} \\
! & \ r12: \ \text{Size of each element, in bits (b).} \\
! & \ r14: \ \text{Address of start of unpacked array.}
\end{align*}
\]