<u>Computer Architecture Exercise 6</u>

1. We will translate each line into formal representation:

ADDI R2, R0, #0x180	REGS[R2] ← REGS[R0] + 0x180
LW R3, 10(R2)	REGS[R3] ← MEM[REGS[R2] + 1]
SUBI R3, R3, #0x020	REGS[R2] ← REGS[R3] – 0x020
XORI R3, R3, #0x040	REGS[R3] ← (0xFFFF – REGS[R3])*0x040 + REGS[RG]*(0xFFFF – 0x040)
AND R3, R3, #0x0000FFFF	REGS[R3] ← REGS[R3]*0x0000FFFF
SW 0(R2), R3	MEM[0 +REGS[R2]] ← REGS[R3]
SUBI R2, R2, #0x4	REGS[R2] ← REGS[R2] – 0x4
BNEZ R2, 0xFFE4	If REGS[R2] != 0 then PC ← PC + 0xFFFFFFE4
ADD R3, R8, R9	REGS[R3] ← REGS[R8] + REGS[R9]

2. The loop starts with the LOOP label and ends with the branch command.
The offset 0xFFE4 is an offset with a sign (that is, a negative offset). Therefore, it expands to 0xFFFFFFE4 indicates the number -0x1C in HEX base, and in a decimal base it indicates the number -28.
Each command is 4 bytes. Therefore, the -28 offset in the decimal base takes us -28/4 = 7 commands back starting from the last line - ADD, that is, in total, the addition of the offset in the branch command takes us back to the LOOP label (in other words, to the beginning of the loop).

3. The branch command BNEZ ran with the register R2, so we will stop running in the loop when we get that R2 = 0. R2 is initialized in the first line to 0x180, and in each transition we subtract from R2 0x4, Therefore, in total we will perform: 0x180/0x4 = 0x60 and in decimal base **96** passes in the loop.

4. The first row and the last row are not in the body of the loop. Therefore, there are 2 commands that will be executed once.
There are 7 lines in the body of the loop, where the number of transitions in each loop will be 96, so in total we will get that the number of commands that will be executed in the program run: 2+7*96 = 674.
IC = **674**.

5. We will calculate how many commands will be executed in the program run by division into ALU, load, store and branch commands.
<u>ALU commands:</u> There are 2 ALU instructions outside the loop body.
In the body of the loop there are 4 more commands: ALU – SUBI twice, XORI and AND. Therefore, we get: ALU instructions = 2 + 4*96 = 386 → $CPI_{ALU} = \frac{386}{674} = \mathbf{0.5727}$.
<u>Store commands:</u> There is one storage command in the body of the loop - SW, therefore we will get: Store instructions = 1*96 = 96 →$CPI_{Store} = \frac{96}{674} = \mathbf{0.1424}$.
<u>Load commands:</u> There is one loading command in the body of the loop - LW, therefore we will get: Load instructions = 1*96 = 96 →$CPI_{Load} = \frac{96}{674} = \mathbf{0.1424}$.
<u>Branch commands:</u> There is one Branch command in the body of the loop - BNEZ,

therefore we will get: Branch instructions = 1*96 = 96 →$CPI_{\text{Branch}} = \frac{96}{674} = \mathbf{0.1424}$.

6. We will calculate the general CPI according to the CPI_i for four types of orders in version 1 of the DLX:

$\text{CPI} = \frac{386}{674} \times 4(\text{ALU}) + \frac{96}{674} \times 4(\text{Store}) + \frac{96}{674} \times 5(\text{Load}) + \frac{96}{674} \times 4(\text{Branch}) = \mathbf{4.1424}$.