6. The Sequencer (CU) controls the CPU and understands the following machine language instructions:

6.1. ADD ADD Rx, Ry (the result is stored in Rx)

6.2. SUB → SUB Rx, Ty (the result is stored in Rx)

6.3. STORE STR Rx, Address (the contents of Rx is put into Data Cache at Address)

6.4. LOAD LD Ry, Address (the contents at Address from Data Cache is put into Ry)

6.5. Branch if Zero BZ Rx, Address (if Rx is zero then go to Instruction Cache Address)

6.6. Branch if not zero → BNZ Rx, Address (If Rx is not zero then go to Instruction Cache Address)

6.7. PRINT → PRT Rx (output contents of Rx to 1-digit digital display buffer)

6.8. READ → INP Rx (store data to Rx from keypad input buffer)

6.9. STOP RETURN (mysteriously, the program stops running)

6.10. MULT Rx, Rx. The result is stored in Rx. (bonus)

You must supply your own binary definition for these instructions (i.e. its op-code,

parameters, etc.)

Rx and Ry can be: R0, R1

R0 and R1 can be used for indirect addressing

Not sure yet what the instruction format will be, but the ALU uses:

ADD = 00

SUB = 01

MULT = 10

to select which operation to perform

In the CU instruction format the same mapping is preserved.

The instruction set is divided in 4 instruction types:

1. **Address operations**: if instructions starts with a 1, then it is an address operation.

The constraint was using only 8bits - Len(address)- Len(register) = 8 - 4 - 1= 3 bits

1. **ALU Operations**: 000 + 1 + ALUOPCODE + Rx + Ry
2. **I/O Operations**: 000 + 0 + IOOPCODE + Rn + 0

Where IOOPCODE is 0 for PRT and 1 for INP

1. **RETURN**: “00000011”

This instruction code was chosen to avoid any ambiguity.