```python
from PyQt5.QtWidgets import QTableWidgetItem, QMessageBox
from PyQt5.QtGui import QFont
from PyQt5.QtCore import Qt

def message_box_errors(errors: list):
    """
    Fenster, das alle Elemente aus errors anzeigt
    """
    info = QMessageBox()
    info.setIcon(QMessageBox.Critical)
    info.setWindowFlags(Qt.WindowTitleHint | Qt.CustomizeWindowHint | Qt.MSWindowsFixedSizeDialogHint)
    font = QFont()
    font.setPointSize(10)
    info.setFont(font)
    info.setText("\n".join(errors))
    info.setWindowTitle("Fehler")
    info.exec_()

class CU:  # Control Unit = Steuerwerk
    def __init__(self, ui, instruction_set, timer, cache, alu, input_register, output_register):
        self.ui = ui
        self.CACHE = cache
        self.ALU = alu
        self.input_register = input_register
        self.timer = timer
        self.output_register = output_register
        self.register = ["?", "0"]
        self.program_counter = 0
        self.instruction_set = instruction_set

    def fetch(self):
        if self.program_counter == -1:
            self.ui.label_which_cycle.setText("IDLE")
            return

        self.register[0] = self.CACHE.read(self.program_counter)
        self.register[1] = self.CACHE.read(self.program_counter + 1)

        self.ui.label_befehlszaehler.setText(str(self.program_counter))
        enc = encode_instruction(self.instruction_set, str(self.register[0]), str(self.register[1]))
        self.ui.label_befehlsreg_1.setText(enc[0])
        self.ui.label_befehlsreg_2.setText(enc[1])

    def decode(self):
        if self.program_counter == -1:
            self.ui.label_which_cycle.setText("IDLE")
            return
        elif self.register[0] not in self.instruction_set:
            self.program_counter = -1
            self.ui.label_which_cycle.setText("IDLE")
            return

        self.ui.label_dekodierer_1.setText(str(self.register[0]))
        self.ui.label_dekodierer_2.setText(str(self.register[1]))

        if self.register[0] == self.instruction_set[0]:  # ld
            if self.register[1].startswith("@"):
                self.ALU.acc = self.CACHE.read(int(self.register[1].replace("@", "")))
            else:
                self.ALU.acc = self.register[1]

        elif self.register[0] == self.instruction_set[1]:  # st
            if self.register[1].startswith("@"):
                self.CACHE.write(int(self.register[1].replace("@", "")), self.ALU.acc)
                try:
                    if int(self.register[1].replace("@", "")) % 2 == 0:
                        self.ui.Speicher.setItem(int(int(self.register[1].replace("@", "")) / 2), 1, QTableWidgetItem(self.ALU.acc))
                    else:
                        self.ui.Speicher.setItem(int(int(self.register[1].replace("@", "")) / 2), 3, QTableWidgetItem(self.ALU.acc))
                except Exception as e:
                    print(e.__reduce_ex__(0))
                    message_box_errors(["Ein Fehler ist aufgetreten!"])
            else:
                message_box_errors(["Ein Fehler ist aufgetreten!"])

        elif self.register[0] == self.instruction_set[2]:  # in
            if self.register[1].startswith("@"):
                self.ui.input_reg.setEnabled(True)
                self.ui.button_weiter.setEnabled(False)
                self.ui.input_reg.setPlaceholderText(self.register[1])
                self.ui.input_reg.setFocus()
                self.timer.stop()  # input handled in Program.input_pressed
            else:
                message_box_errors(["Ein Fehler ist aufgetreten!"])

        elif self.register[0] == self.instruction_set[3]:  # out
            if self.register[1].startswith("@"):
                self.output_register.text = self.CACHE.read(int(self.register[1].replace("@", "")))
                self.ui.output_reg.setText(str(self.output_register.text))
            else:
                message_box_errors(["Ein Fehler ist aufgetreten!"])

        elif self.register[0] == self.instruction_set[4]:  # add
            if self.register[1].startswith("@"):
                self.ALU.operand = self.CACHE.read(int(self.register[1].replace("@", "")))
            else:
                self.ALU.operand = self.CACHE.read(int(self.program_counter + 1))
            self.ALU.operation = "+"

        elif self.register[0] == self.instruction_set[5]:  # sub
            if self.register[1].startswith("@"):
                self.ALU.operand = self.CACHE.read(int(self.register[1].replace("@", "")))
            else:
                self.ALU.operand = self.CACHE.read(self.program_counter + 1)
            self.ALU.operation = "-"

        elif self.register[0] == self.instruction_set[6]:  # mul
            if self.register[1].startswith("@"):
                self.ALU.operand = self.CACHE.read(int(self.register[1].replace("@", "")))
            else:
                self.ALU.operand = self.CACHE.read(self.program_counter + 1)
            self.ALU.operation = "*"

        elif self.register[0] == self.instruction_set[7]:  # div
            if self.register[1].startswith("@"):
                self.ALU.operand = self.CACHE.read(int(self.register[1].replace("@", "")))
            else:
                self.ALU.operand = self.CACHE.read(self.program_counter + 1)
            self.ALU.operation = "/"

        elif self.register[0] == self.instruction_set[8]:  # mod
            if self.register[1].startswith("@"):
                self.ALU.operand = self.CACHE.read(int(self.register[1].replace("@", "")))
            else:
                self.ALU.operand = self.CACHE.read(self.program_counter + 1)
            self.ALU.operation = "%"

        elif self.register[0] == self.instruction_set[9]:  # cmp
            if self.register[1].startswith("@"):
                self.ALU.operand = self.CACHE.read(int(self.register[1].replace("@", "")))
            else:
                self.ALU.operand = self.CACHE.read(self.program_counter + 1)
            self.ALU.operation = "?"

        if self.program_counter != -1:
            self.program_counter += 2

        if self.register[0] == self.instruction_set[10]:  # jmp
            self.program_counter = int(self.CACHE.read(self.program_counter - 1)) * 2 - 2

        elif self.register[0] == self.instruction_set[11]:  # jlt
            if self.ALU.result == "-1":
                self.program_counter = int(self.CACHE.read(self.program_counter - 1)) * 2 - 2

        elif self.register[0] == self.instruction_set[12]:  # jeq
            if self.ALU.result == "0":
                self.program_counter = int(self.CACHE.read(self.program_counter - 1)) * 2 - 2

        elif self.register[0] == self.instruction_set[13]:  # jgt
            if self.ALU.result == "1":
                self.program_counter = int(self.CACHE.read(self.program_counter - 1)) * 2 - 2

        if self.register[0] == self.instruction_set[14]:  # end
            self.program_counter = -1
            self.ui.label_which_cycle.setText("IDLE")

    def execute(self):
        if self.program_counter == -1:
            self.ui.label_which_cycle.setText("IDLE")
            return

        if self.register[0] in self.instruction_set[4:10]:
            self.ALU.compute()


def encode_instruction(instruction_set: list, instruction: str, val: str):
    if len(instruction_set) != 15:
        return "0", "0"

    elif instruction == instruction_set[0]:  # ld
        return "101" if str(val).startswith("@") else "100", str(val).replace("@", "")

    elif instruction == instruction_set[1] and val.startswith("@"):  # st
        return "111", str(val.replace("@", ""))

    elif instruction == instruction_set[2] and val.startswith("@"):  # in
         return "121", str(val.replace("@", ""))

    elif instruction == instruction_set[3] and val.startswith("@"):  # out
        return "131", str(val.replace("@", ""))

    elif instruction == instruction_set[4]:  # add
        return "201" if str(val).startswith("@") else "200", str(val).replace("@", "")

    elif instruction == instruction_set[5]:  # sub
        return "211" if str(val).startswith("@") else "210", str(val).replace("@", "")

    elif instruction == instruction_set[6]:  # mul
        return "221" if str(val).startswith("@") else "220", str(val).replace("@", "")

    elif instruction == instruction_set[7]:  # div
```

```python
        return "231" if str(val).startswith("@") else "230", str(val).replace("@", "")

    elif instruction == instruction_set[8]:  # mod
        return "241" if str(val).startswith("@") else "240", str(val).replace("@", "")

    elif instruction == instruction_set[9]:  # cmp
        return "251" if str(val).startswith("@") else "250", str(val).replace("@", "")

    elif instruction == instruction_set[10] and not val.startswith("@"):  # jmp
        return "300", str(val)

    elif instruction == instruction_set[11] and not val.startswith("@"):  # jlt
        return "310", str(val)

    elif instruction == instruction_set[12] and not val.startswith("@"):  # jeq
        return "320", str(val)

    elif instruction == instruction_set[13] and not val.startswith("@"):  # jgt
        return "330", str(val)

    elif instruction == instruction_set[14] and not val.startswith("@"):  # end
        return "400", "0"

    else:
        return "0", "0"
```