```
from PyQt5.QtGui import QFont, QRegExpValidator, QTextCursor, QColor
from PyQt5.QtCore import QRegExp, Qt, QTimer
from PyQt5.QtWidgets import *
from QProgramWindow import Ui_MainWindow
from QDialogSpeicherzellen import Ui_Dialog_Speicher
from QDialogInsSet import Ui_QDialog_inset
from QDialogDefaultInsSet import Ui_Dialog
from REGISTER import INPUT, OUTPUT
from CACHE import CACHE
from ALU import ALU
from CU import CU, encode_instruction
import re, sys, os

app = QApplication(sys.argv)
main_window = QMainWindow()
ui = Ui_MainWindow()
ui.setupUi(main_window)

# Fenstergröße / Position: Zentriert und 2/3 der Bildschirmgröße
width = int(app.primaryScreen().size().width() / 3 * 2)
height = int(app.primaryScreen().size().height() / 3 * 2)
x_pos = int((app.primaryScreen().size().width() - width) / 2)
y_pos = int((app.primaryScreen().size().height() - height) / 2)
main_window.setGeometry(x_pos, y_pos, width, height)

# globale Variablen
global instruction_set, instruction_set_text, saved, current_file, prev_current_file, interval, timer, cycle, cu, prev_programm_counter
global offset, missing_op, code_factorial, code_digit_sum, code_sum_a_to_b_with_loop, code_sum_a_to_b_without_loop, code_mean_value_3_numbers
global code_mean_value_arbitrary, cell_number
instruction_set = []
instruction_set_text = "ld\nst\nin\nout\nadd\nsub\nmul\ndiv\nmod\ncmp\njmp\njlt\njeq\njgt\nend"
saved = True
current_file = os.path.join(os.path.join(os.environ["USERPROFILE"]), "Desktop")
prev_current_file = ""
interval = 0
timer = QTimer()
cycle = 0
cu = None
prev_programm_counter = 0
offset = 0
missing_op = False
cell_number = 64


# ================================================================================ #
# ============================== methods ========================================= #

```

```
48
49  def get_instruction_set(source: str):
50      """
51      Den aktuellen Befehlssatz aus der Textdatei (source) laden.
52      """
53      try:
54          with open(source, "r") as f:
55              global instruction_set_text
56              instruction_set_text = f.read()
57
58          with open(source, "r") as f:
59              global instruction_set
60              instruction_set = []
61              for l in f.readlines():
62                  instruction_set.append(l.strip())
63      except:
64          message_box_errors(["Fehler beim Laden des Befehlssatzes!"])
65
66
67  def load(source_code: str):
68      """
69      Kompiliert den Quellcode aus dem Code Editor, speichert die kodierten Befehle im Speicher und visualisiert gleichzeitig das Speicherwerk.
70      """
71      setup_ram(cell_number)  # Die Anzahl der Speicherzellen setzen
72      get_instruction_set("instruction_set.txt")  # Den aktuellen Befehlssatz laden
73      global cu, offset
74      cu = CU(ui, instruction_set, timer, CACHE(), ALU(), INPUT(), OUTPUT())
75
76      end = 0
77      cell = 0
78      variables = dict()
79      errors = []
80      offset = 0
81      line_number = 0
82
83      # Den Quellcode durchgehen, Fehler erkennen und Kompilation
84      for line in source_code.splitlines(False):
85          line_number += 1
86          line = line.split("#")[0]  # Kommentare ausschließen
87          global missing_op
88          missing_op = False
89
90          if not line.isspace() and line != "":
91              c = re.sub(" +", " ", line.replace("\n","").replace("  ", " ")).strip().split(" ")
92
93              if len(c) < 2 and c[0] in instruction_set[:14]:
94                  errors.append(str(line_number) + ": Fehlender Operand")
95                  missing_op = True
```

```
96
97          if c[0].strip() == "def" and not missing_op:
98              def_line = re.sub(" +", " ", " ".join(c).replace("def ", "").replace("=", "  ")).split(" ")
99              if len(def_line) != 2 or "=" not in "".join(c):
100                 errors.append(str(line_number) + ": Inkorrekte Initialisierung einer Variable " + " ".join(c))
101             elif def_line[0] != "" and def_line[0][0].isalpha():
102                 if def_line[-1].startswith("@") and def_line[-1].replace("@", "").replace(".", "").replace("-", "").isdigit():
103                     variables[def_line[0].strip()] = "@" + str(abs(int(float(def_line[-1].replace("@", "").strip()))))
104                 elif def_line[-1].replace(".", "").replace("-", "").isdigit():
105                     variables[def_line[0].strip()] = str(int(float(def_line[-1].strip())))
106                 else:
107                     errors.append(str(line_number) + ": Inkorrekte Initialisierung einer Variable " + " ".join(c))
108             else:
109                 errors.append(str(line_number) + ": Das erste Zeichen eines Variablennamens muss ein Buchstabe sein!")
110             cell -= 2
111             offset += 1
112
113         elif c[0].strip() not in instruction_set:
114             errors.append(str(line_number) + ": unbekannter Befehl: " + c[0])
115
116         elif instruction_set[14] == c[0]:
117             cu.CACHE.write(end, c[0].strip())
118             write_ui_ram(end, c[0].strip(), "0")
119
120         elif c[0].strip() in instruction_set[10:13] and not missing_op:
121             cu.CACHE.write(cell, c[0].strip())
122             cu.CACHE.write(cell + 1, str(int(float(c[1].strip()) - offset)))
123             write_ui_ram(cell, c[0].strip(), str(int(float(c[1].strip()) - offset)))
124             end += 2
125
126         elif not missing_op:
127             cu.CACHE.write(cell, c[0].strip())
128             if c[1].startswith("@"):
129                 cu.CACHE.write(cell + 1, "@" + str(abs(int(float(c[1].replace("@", "").strip())))))
130                 write_ui_ram(cell, c[0].strip(), "@" + str(abs(int(float(c[1].replace("@", "").strip())))))
131             else:
132                 try:
133                     cu.CACHE.write(cell + 1, str(int(float(c[1].strip()))))
134                     write_ui_ram(cell, c[0].strip(), str(int(float(c[1].strip()))))
135                 except:
136                     if c[1] in variables.keys():
137                         if variables[c[1].strip()].startswith("@"):
138                             cu.CACHE.write(cell + 1, "@" + str(abs(int(float(variables[c[1].strip()].replace("@", "").strip())))))
139                             write_ui_ram(cell, c[0].strip(), "@" + str(abs(int(float(variables[c[1].strip()].replace("@", "").strip())))))
140                         else:
141                             cu.CACHE.write(cell + 1, str(int(float(variables[c[1].strip()]))))
142                             write_ui_ram(cell, c[0].strip(), str(int(float(variables[c[1].strip()]))))
143                     else:
```

```
144                         errors.append(str(line_number) + ": Variable '" + c[1] + "' wurde noch nicht initialisiert!")
145
146                 end += 2
147
148             cell += 2
149
150         else:
151             offset += 1
152
153     if len(errors) > 0:
154         message_box_errors(errors)
155         return False
156
157     ui.actionFakult_t.setEnabled(False)
158     ui.actionQuersumme.setEnabled(False)
159     ui.actionSumme_a_bis_b_mit_Schleife.setEnabled(False)
160     ui.actionSumme_a_bis_b_ohne_Schleife.setEnabled(False)
161     ui.actionDurchschnitt_dreier_Zahlen_floor.setEnabled(False)
162     ui.actionBefehlssatz_ver_ndern.setEnabled(False)
163     ui.actionZahl_der_angezeigten_Speicherzellen.setEnabled(False)
164     ui.actionDurchschnitt_beliebig_vieler_Zahlen.setEnabled(False)
165     return True
166
167
168 def message_box_errors(errors: list):
169     """
170     Fenster, das alle Elemente aus errors anzeigt
171     """
172     info = QMessageBox()
173     info.setIcon(QMessageBox.Critical)
174     info.setWindowFlags(Qt.WindowTitleHint | Qt.CustomizeWindowHint | Qt.MSWindowsFixedSizeDialogHint)
175     font = QFont()
176     font.setPointSize(10)
177     info.setFont(font)
178     info.setText("\n".join(errors))
179     info.setWindowTitle("Fehler")
180     info.exec_()
181
182
183 def run_vn():
184     """
185     Führt immer die nächste VN-Phase aus (Fetch, Decode oder Execute) und setzt die Labels in der GUI.
186     """
187     global cycle, timer, cu
188
189     if cycle >= 3:
190         cycle = 0
191
```

```
192     try:
193         if cycle == 0:
194             # Bei neuem Zyklus wird im Code Editor und im Speicherwerk eine neue Zeile matkiert
195             cursor = QTextCursor(ui.Editor.document().findBlockByLineNumber(int(cu.program_counter / 2) + offset))
196             ui.Editor.setTextCursor(cursor)
197             if int(cu.program_counter / 2) < ui.Speicher.rowCount():
198                 global prev_programm_counter
199                 for j in range(ui.Speicher.columnCount()):
200                     try:
201                         if int(prev_programm_counter / 2) % 2 == 0:
202                             ui.Speicher.item(int(prev_programm_counter / 2), j).setBackground(QColor("#DCDCDC"))
203                         else:
204                             ui.Speicher.item(int(prev_programm_counter / 2), j).setBackground(QColor(Qt.white))
205                         ui.Speicher.item(int(cu.program_counter / 2), j).setBackground(QColor(Qt.green).lighter(150))
206                     except:
207                         pass
208                 prev_programm_counter = cu.program_counter
209 
210             ui.label_which_cycle.setText("FETCH")
211             cu.fetch()
212         elif cycle == 1:
213             ui.label_which_cycle.setText("DECODE")
214             cu.decode()
215         elif cycle == 2:
216             ui.label_which_cycle.setText("EXECUTE")
217             cu.execute()
218 
219         ui.label_operand.setText(cu.ALU.operand)
220         ui.label_operation.setText(cu.ALU.operation)
221         ui.label_accu.setText(cu.ALU.acc)
222         ui.label_result.setText(cu.ALU.result)
223 
224         cycle += 1
225     except Exception as e:
226         stop()
227         if type(e) == ZeroDivisionError:
228             message_box_errors([str(int(cu.program_counter / 2) + offset) + ": Division durch 0"])
229         else:
230             message_box_errors([str(int(cu.program_counter / 2) + offset + 1) + ": Ein Fehler ist aufgetreten!"])
231 
232     if cu.program_counter == -1:
233         stop()
234 
235 
236 def start():
237     """
238     Wird ausgeführt, wenn "Start" gedrückt wird. Setzt die GUI zurück und startet den Timer.
239     """
```

```
240     if not load(ui.Editor.toPlainText()):
241         return
242 
243     ui.label_which_cycle.setText("IDLE")
244     ui.output_reg.setText("")
245     ui.input_reg.setText("")
246     ui.input_reg.setPlaceholderText("")
247     ui.input_reg.setEnabled(False)
248     ui.label_befehlszaehler.setText("0")
249     ui.label_befehlsreg_1.setText("0")
250     ui.label_befehlsreg_2.setText("0")
251     ui.label_dekodierer_1.setText("0")
252     ui.label_dekodierer_2.setText("0")
253     ui.label_accu.setText("0")
254     ui.label_operand.setText("0")
255     ui.label_operation.setText(" ")
256     ui.label_result.setText("0")
257 
258     if ui.actionVNPhasen.isChecked():
259         ui.button_weiter.setEnabled(True)
260     else:
261         ui.button_weiter.setEnabled(False)
262 
263     ui.button_start.setEnabled(False)
264     ui.button_stop.setEnabled(True)
265 
266     global timer, cycle, interval, cu
267 
268     timer = QTimer()
269     cycle = 0
270 
271     timer.setInterval(int(interval * 1000 / 3))
272     ui.Editor.setTextInteractionFlags(Qt.TextSelectableByKeyboard)
273     ui.Editor.setTextCursor(QTextCursor(ui.Editor.document().findBlockByLineNumber(offset)))
274     if interval >= 0:
275         timer.timeout.connect(lambda: run_vn())
276     timer.start()
277     cu.timer = timer
278     run_vn()
279 
280 
281 def stop():
282     """
283     Wird ausgeführt, wenn "Beenden" gedrückt wird. Setzt die GUI zurück und beendet den Timer.
284     """
285     timer.stop()
286     ui.button_weiter.setEnabled(False)
287     ui.button_start.setEnabled(True)
```

```
    ui.button_stop.setEnabled(False)
    ui.Editor.setTextInteractionFlags(Qt.TextSelectableByMouse | Qt.TextSelectableByKeyboard | Qt.TextEditable)
    ui.input_reg.setPlaceholderText("")
    ui.input_reg.setText("")
    ui.input_reg.setEnabled(False)
    ui.Editor.setFocus()

    ui.actionFakult_t.setEnabled(True)
    ui.actionQuersumme.setEnabled(True)
    ui.actionSumme_a_bis_b_mit_Schleife.setEnabled(True)
    ui.actionSumme_a_bis_b_ohne_Schleife.setEnabled(True)
    ui.actionDurchschnitt_beliebig_vieler_Zahlen.setEnabled(True)
    ui.actionDurchschnitt_dreier_Zahlen_floor.setEnabled(True)
    ui.actionBefehlssatz_ver_ndern.setEnabled(True)
    ui.actionZahl_der_angezeigten_Speicherzellen.setEnabled(True)

    if int(cu.program_counter / 2) < ui.Speicher.rowCount():
        global prev_programm_counter
        for j in range(ui.Speicher.columnCount()):
            try:
                if int(prev_programm_counter / 2) % 2 == 0:
                    ui.Speicher.item(int(prev_programm_counter / 2),j).setBackground(QColor(QColor("#DCDCDC")))
                else:
                    ui.Speicher.item(int(prev_programm_counter / 2),j).setBackground(QColor(Qt.white))
            except:
                pass


def write_ui_ram(adr: str, instruction: str, val: str):
    """
    Trägt einen Befehl und den dazugehörigen Wert (kodiert) in das Speicherwerk (GUI) an die Adresse ein.
    """
    try:
        enc = encode_instruction(instruction_set, instruction, val)
        ui.Speicher.setItem(int(adr / 2), 1, QTableWidgetItem(enc[0]))
        ui.Speicher.setItem(int(adr / 2), 3, QTableWidgetItem(enc[1]))
    except:
        pass


def setup_ram(cell_count: int):
    """
    Setup des Speicherwerks (GUI)
    """
    ui.Speicher.setColumnCount(4)
    ui.Speicher.setRowCount(round(cell_count / 2 + 0.2))
    ui.Speicher.setHorizontalHeaderLabels(["Adr", "Wert", "Adr", "Wert"])
    font = QFont()
```

```
    font.setPointSize(15)
    font.setBold(True)
    header = ui.Speicher.horizontalHeader()
    header.setSectionResizeMode(0, QHeaderView.ResizeToContents)
    header.setSectionResizeMode(1, QHeaderView.Stretch)
    header.setSectionResizeMode(2, QHeaderView.ResizeToContents)
    header.setSectionResizeMode(3, QHeaderView.Stretch)
    ui.Speicher.setHorizontalHeader(header)
    i = 0
    while i < ui.Speicher.rowCount():
        it1 = QTableWidgetItem(str(i * 2))
        it1.setFont(font)
        it2 = QTableWidgetItem(str(i * 2 + 1))
        it2.setFont(font)
        ui.Speicher.setItem(i, 0, it1)
        ui.Speicher.setItem(i, 1, QTableWidgetItem("0"))
        ui.Speicher.setItem(i, 2, it2)
        ui.Speicher.setItem(i, 3, QTableWidgetItem("0"))
        i += 1


def setup_demo():
    """
    Lädt die Demos mit dem aktuellen Befehlssatz neu.
    """
    global code_factorial, code_digit_sum, code_sum_a_to_b_with_loop, code_sum_a_to_b_without_loop
    global code_mean_value_3_numbers, code_mean_value_arbitrary

    code_factorial = "# Fakultät\n\ndef a = @30\ndef b = @31\n\n" + instruction_set[2] + "  a\n" + instruction_set[0] + "  1\n" + \
                     instruction_set[1] + "  b\n" + instruction_set[0] + "  a\n" + instruction_set[9] + " 2\n" + \
                     instruction_set[11] + " 18\n  " + instruction_set[6] + " b\n  " + instruction_set[1] + "  b\n  " + \
                     instruction_set[0] + "  a\n  " + instruction_set[5] + " 1\n  " + instruction_set[1] + "  a\n" + \
                     instruction_set[10] + " 9\n" + instruction_set[3] + " b\n" + instruction_set[14]
    code_digit_sum = "# Quersumme\n\ndef a = @28\ndef b = @29\n\n" + instruction_set[2] + "  a\n" + instruction_set[0] + "  a\n" + \
                     instruction_set[9] + " 0\n" + instruction_set[12] + " 17\n  " + instruction_set[8] + " 10\n  " + \
                     instruction_set[4] + " b\n  " + instruction_set[1] + "  b\n  " + instruction_set[0] + "  a\n  " + \
                     instruction_set[7] + " 10\n  " + instruction_set[1] + "  a\n" + instruction_set[10] + " 7\n" + \
                     instruction_set[3] + " b\n" + instruction_set[14]
    code_sum_a_to_b_with_loop = "# Summe a bis b\n# mit Schleife\n\ndef a = @34\ndef b = @35\ndef c = @36\n\n" + instruction_set[2] + "  a\n" + \
                                instruction_set[2] + "  b\n" + instruction_set[0] + "  b\n" + instruction_set[4] + " 1\n" + \
                                instruction_set[1] + "  b\n" + instruction_set[0] + "  a\n" + instruction_set[9] + " b\n" + \
                                instruction_set[12] + " 22\n  " + instruction_set[4] + " c\n  " + instruction_set[1] + "  c\n  " + \
                                instruction_set[0] + "  a\n  " + instruction_set[4] + " 1\n  " + instruction_set[1] + "  a\n" + \
                                instruction_set[10] + " 13\n" + instruction_set[3] + " c\n" + instruction_set[14]
    code_sum_a_to_b_without_loop = "# Summe a bis b\n# ohne Schleife\n\ndef a = @38\ndef b = @39\n\n" + instruction_set[2] + "  a\n" + \
                                   instruction_set[2] + "  b\n" + instruction_set[0] + "  a\n" + instruction_set[5] + " 1\n" + \
                                   instruction_set[6] + " a\n" + instruction_set[1] + "  a\n" + instruction_set[7] + " 2\n" + \
                                   instruction_set[1] + "  a\n" + instruction_set[0] + "  b\n" + instruction_set[4] + " 1\n" + \
```

```
                        instruction_set[6] + " b\n" + instruction_set[1] + "  b\n" + instruction_set[7] + " 2\n" + \
                        instruction_set[1] + "  b\n" + instruction_set[5] + " a\n" + instruction_set[1] + "  a\n" + \
                        instruction_set[3] + " a\n" + instruction_set[14]
    code_mean_value_3_numbers = "# Durchschnitt\n# dreier Zahlen\n\ndef a = @30\ndef b = @31\n\n" + instruction_set[2] + "  a\n" + \
                        instruction_set[2] + "  b\n" + instruction_set[0] + "  a\n" + instruction_set[4] + " b\n" + \
                        instruction_set[1] + "  a\n" + instruction_set[2] + "  b\n" + instruction_set[0] + "  a\n" + \
                        instruction_set[4] + " b\n" + instruction_set[1] + "  a\n" + instruction_set[0] + "  a\n" + \
                        instruction_set[7] + " 3\n" + instruction_set[1] + "  a\n" + instruction_set[3] + " a\n" + instruction_set[14]
    code_mean_value_arbitrary = "# Durchschnitt\n# beliebig\n# vieler Zahlen\n\ndef a = @32\ndef b = @33\ndef c = @34\ndef anzahl = 5\n\n" + \
                        instruction_set[0] + "  c\n" + instruction_set[9] + " anzahl\n" + instruction_set[12] + " 20\n  " + \
                        instruction_set[2] + "  a\n  " + instruction_set[4] + " b\n  " + instruction_set[1] + "  b\n  " + \
                        instruction_set[0] + "  c\n  " + instruction_set[4] + " 1\n  " + instruction_set[1] + "  c\n" + \
                        instruction_set[10] + " 10\n" + instruction_set[0] + "  b\n" + instruction_set[7] + " anzahl\n" + \
                        instruction_set[1] + "  b\n" + instruction_set[3] + " b\n" + instruction_set[14]


def show_dialog_ram():
    """
    Fenster, um auszuwählen, wie viele Speicherzellen im Speicherwerk (GUI) angezeigt werden.
    """
    global cell_number
    dialog_speicher = QDialog(flags=Qt.WindowTitleHint | Qt.CustomizeWindowHint | Qt.MSWindowsFixedSizeDialogHint)
    ui2 = Ui_Dialog_Speicher()
    ui2.setupUi(dialog_speicher)
    ui2.buttonBox.button(QDialogButtonBox.Cancel).setText("Abbrechen")
    ui2.spinBox.setValue(cell_number)
    dialog_speicher.show()
    if dialog_speicher.exec_() == 1 and 1 < int(ui2.spinBox.text()) < ui2.spinBox.maximum() + 1:
        cell_number = int(ui2.spinBox.text())
        font = QFont()
        font.setPointSize(14)
        font.setBold(True)
        if ui.Speicher.rowCount() * 2 < cell_number:
            while ui.Speicher.rowCount() * 2 < cell_number:
                ui.Speicher.insertRow(ui.Speicher.rowCount())
                it1 = QTableWidgetItem(str((ui.Speicher.rowCount() - 1) * 2))
                it1.setFont(font)
                it2 = QTableWidgetItem(str((ui.Speicher.rowCount() - 1) * 2 + 1))
                it2.setFont(font)
                ui.Speicher.setItem(ui.Speicher.rowCount() - 1, 0, it1)
                ui.Speicher.setItem(ui.Speicher.rowCount() - 1, 1, QTableWidgetItem("0"))
                ui.Speicher.setItem(ui.Speicher.rowCount() - 1, 2, it2)
                ui.Speicher.setItem(ui.Speicher.rowCount() - 1, 3, QTableWidgetItem("0"))

        elif ui.Speicher.rowCount() * 2 > cell_number:
            while ui.Speicher.rowCount() * 2 > cell_number:
                ui.Speicher.removeRow(ui.Speicher.rowCount() - 1)

```

```
def show_dialog_info():
    """
    Fenster, in dem Infos über das Programm stehen
    """
    info = QMessageBox()
    info.setIcon(QMessageBox.Information)
    info.setWindowFlags(Qt.WindowTitleHint | Qt.CustomizeWindowHint | Qt.MSWindowsFixedSizeDialogHint)
    font = QFont()
    font.setPointSize(10)
    info.setFont(font)
    info.setText("Dieses Programm ist im Rahmen meiner Seminararbeit entstanden.\n"
                 "Die Inspiration bzw. Vorlage stammt vom Modellrechner MOPS.\n"
                 "Die Umsetzung stammt allerdings von mir selbst.\nSteve Heilenz")
    info.setWindowTitle("Info")
    info.setInformativeText("Sprache:     Python 3.7\nIDE:            PyCharm (Community Edition)\n"
                            "für die GUI: PyQt5 und Qt Designer (Open Source Lizenz)\n")
    info.exec_()


def show_dialog_ins_set():
    """
    Fenster, um den Befehlssatz zu verändern
    """
    dialog_inset = QDialog(flags=Qt.WindowTitleHint | Qt.CustomizeWindowHint | Qt.MSWindowsFixedSizeDialogHint)
    ui3 = Ui_QDialog_inset()
    ui3.setupUi(dialog_inset)
    get_instruction_set("instruction_set.txt")
    ui3.textEdit.insertPlainText(instruction_set_text)
    ui3.buttonBox.button(QDialogButtonBox.Cancel).setText("Abbrechen")
    ui3.pushButton.clicked.connect(lambda: instruction_set_default())
    dialog_inset.show()

    def instruction_set_default():
        """
        Fenster, um zu bestätigen, dass der Standardbefehlssatz wiederhergestellt werden soll
        """
        dialog_default = QDialog(flags=Qt.WindowTitleHint | Qt.CustomizeWindowHint | Qt.MSWindowsFixedSizeDialogHint)
        ui4 = Ui_Dialog()
        ui4.setupUi(dialog_default)
        ui4.buttonBox.button(QDialogButtonBox.Yes).setText("Ja")
        ui4.buttonBox.button(QDialogButtonBox.No).setText("Nein")
        dialog_default.show()
        if dialog_default.exec_() == 1:
            ui3.textEdit.clear()
            ui3.textEdit.setHtml("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict."
                                 "dtd\">\n<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\np, li "
                                 "{ white-space: pre-wrap; }\n</style></head><body style=\" font-family:\'Consolas\'; font-size:"
```

```
480                         "14pt; font-weight:600; font-style:normal;\">\n<p style=\"-qt-paragraph-type:empty; margin-top:"
481                         "0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px"
482                         "; line-height:114%;\"><br /></p></body></html>")
483         ui3.textEdit.insertPlainText("ld\nst\nin\nout\nadd\nsub\nmul\ndiv\nmod\ncmp\njmp\njlt\njeq\njgt\nend")
484
485     if dialog_inset.exec_() == 1:
486         set_lines = set()
487         for line in ui3.textEdit.toPlainText().splitlines(False):
488             line = line.replace("\n", "")
489             if not line.isspace() and line != "" and not line.strip().__contains__(" ") and not line.strip().__contains__("\t"):
490                 set_lines.add(line.strip())
491
492         if len(set_lines) == 15:
493             with open("instruction_set.txt", "w") as f:
494                 f.write(ui3.textEdit.toPlainText().replace(" ", ""))
495             get_instruction_set("instruction_set.txt")
496             setup_demo()
497
498         else:
499             info = QMessageBox()
500             info.setIcon(QMessageBox.Information)
501             info.setWindowFlags(Qt.WindowTitleHint | Qt.CustomizeWindowHint | Qt.MSWindowsFixedSizeDialogHint)
502             font = QFont()
503             font.setPointSize(10)
504             info.setFont(font)
505             info.setText("Der Befehlssatz wurde nicht geändert!")
506             info.setWindowTitle("Fehler")
507             info.exec_()
508
509
510 def show_dialog_open():
511     """
512     Fenster, um zu bestätigen, dass nicht gespeicherter Quellcode beim Öffnen verloren geht
513     """
514     dialog_o = QFileDialog()
515     try:
516         global saved, current_file
517         dialog = QDialog(flags=Qt.WindowTitleHint | Qt.CustomizeWindowHint | Qt.MSWindowsFixedSizeDialogHint)
518         ui4 = Ui_Dialog()
519         ui4.setupUi(dialog)
520         ui4.label.setText("Nicht gespeicherter Quellcode wird verloren gehen!")
521         ui4.label_2.setText("")
522         dialog.setWindowTitle("Warnung")
523         ui4.buttonBox.button(QDialogButtonBox.Yes).setText("Fortfahren")
524         ui4.buttonBox.button(QDialogButtonBox.No).setText("Abrechen")
525         dialog.show()
526         if dialog.exec_() == 1:
527             file_name, _ = QFileDialog.getOpenFileName(
```

```
528             dialog_o, "Öffnen", current_file, "Textdokument (*.txt)")
529         with open(file_name, "r") as f:
530             ui.Editor.setPlainText(f.read())
531 
532         current_file = file_name
533     except:
534         pass
535 
536 
537 def show_dialog_save_as():
538     """
539     Dialogfenster "Speichern als"
540     """
541     dialog_s = QFileDialog()
542     try:
543         global saved, current_file, prev_current_file
544         file_name, _ = QFileDialog.getSaveFileName(
545             dialog_s, "Speichern", current_file, "Textdokument (*.txt)")
546         with open(file_name, "w") as f:
547             f.write(ui.Editor.toPlainText())
548 
549         saved = True
550         current_file = file_name
551         prev_current_file = file_name
552     except:
553         pass
554 
555 
556 def save():
557     """
558     Speichert den Quellcode in einer Textdatei. Falls der aktuelle Quellcode nicht gespeichert wurde und sich
559     die Textdatei geändert hat, mit der gearbeitet wird, wird der Dialog "Speichern als" angezeigt.
560     """
561     if current_file != prev_current_file and not saved:
562         show_dialog_save_as()
563     else:
564         try:
565             with open(current_file, "w") as f:
566                 f.write(ui.Editor.toPlainText())
567         except:
568             pass
569 
570 
571 def new():
572     """
573     Erstellt eine neue Textdatei für den Quellcode.
574     """
575     global saved, current_file
```

```
    dialog = QDialog(flags=Qt.WindowTitleHint | Qt.CustomizeWindowHint | Qt.MSWindowsFixedSizeDialogHint)
    ui4 = Ui_Dialog()
    ui4.setupUi(dialog)
    ui4.label.setText("Nicht gespeicherter Quellcode wird verloren gehen!")
    ui4.label_2.setText("")
    dialog.setWindowTitle("Warnung")
    ui4.buttonBox.button(QDialogButtonBox.Yes).setText("Fortfahren")
    ui4.buttonBox.button(QDialogButtonBox.No).setText("Abrechen")
    dialog.show()
    if dialog.exec_() == 1:
        current_file = os.path.join(os.path.join(os.environ["USERPROFILE"]), "Desktop")
        saved = False
        ui.Editor.clear()


def open_doc():
    """
    Öffnet die Dokumentation in der Standard-App für .pdf-Dateien.
    """
    try:
        os.startfile("Dokumentation.pdf")
    except:
        message_box_errors(["Fehler beim Öffnen der Dokumentation!"])


def validate_action(action: QAction):
    """
    Beim Auswählen eines neuen Ablaufs wird der aktuell angewählte Ablauf abgewählt und der Timer wird entsprechend gesetzt.
    """
    if not action.isChecked():
        action.setChecked(True)
    else:
        for a in ui.menuVollst_ndig.actions():
            if a is not action:
                a.setChecked(False)
        ui.actionVNPhasen.setChecked(False)

        if action is ui.actionVNPhasen:
            for a in ui.menuVollst_ndig.actions():
                a.setChecked(False)
            ui.actionVNPhasen.setChecked(True)

        global interval
        if action is ui.actionVNPhasen:
            interval = -1
        else:
            interval = int(action.text().replace("s", ""))

```

```
624
625 def input_pressed():
626     """
627     Wird ausgeführt, wenn im Eingaberegister "ENTER" gedrückt wird. Setzt das Programm nach Warten auf die Eingabe fort.
628     """
629     text = ui.input_reg.text()
630     adr = int(ui.input_reg.placeholderText().replace("@", ""))
631     cu.CACHE.write(adr, text)
632     cu.ALU.acc = text
633
634     try:
635         column = 1 if int(cu.register[1].replace("@", "")) % 2 == 0 else 3
636         ui.Speicher.setItem(int(adr / 2), column, QTableWidgetItem(text))
637     except:
638         pass
639
640     if interval < 0:
641         ui.button_weiter.setEnabled(True)
642         ui.button_weiter.setFocus()
643     else:
644         timer.setInterval(int(interval * 1000 / 3))
645
646     timer.start()
647     ui.input_reg.setPlaceholderText("")
648     ui.input_reg.setText("")
649     ui.input_reg.setEnabled(False)
650
651
652 def set_saved(new_val: bool):
653     global saved
654     saved = new_val
655
656
657 # ================================================================================== #
658 # ================================ setup ============================================ #
659
660
661 get_instruction_set("instruction_set.txt")
662 setup_ram(64)
663 setup_demo()
664
665 # === Alle Buttons bzw. klickbare Felder werden mit den dazugehörigen Methoden verknüpft. === #
666
667 # Hilfe
668 ui.actionDokumentation.triggered.connect(lambda: open_doc())
669 ui.actionInfo.triggered.connect(lambda: show_dialog_info())
670
671 # Optionen
```

```
ui.actionBefehlssatz_anpassen.triggered.connect(lambda: show_dialog_ins_set())
ui.actionZahl_der_angezeigten_Speicherzellen_2.triggered.connect(lambda: show_dialog_ram())

# Demo
ui.actionFakult_t.triggered.connect(lambda: ui.Editor.setPlainText(code_factorial))
ui.actionFakult_t.triggered.connect(lambda: set_saved(False))
ui.actionQuersumme.triggered.connect(lambda: ui.Editor.setPlainText(code_digit_sum))
ui.actionQuersumme.triggered.connect(lambda: set_saved(False))
ui.actionSumme_a_bis_b_mit_Schleife.triggered.connect(lambda: ui.Editor.setPlainText(code_sum_a_to_b_with_loop))
ui.actionSumme_a_bis_b_mit_Schleife.triggered.connect(lambda: set_saved(False))
ui.actionSumme_a_bis_b_ohne_Schleife.triggered.connect(lambda: ui.Editor.setPlainText(code_sum_a_to_b_without_loop))
ui.actionSumme_a_bis_b_ohne_Schleife.triggered.connect(lambda: set_saved(False))
ui.actionDurchschnitt_dreier_Zahlen_floor.triggered.connect(lambda: ui.Editor.setPlainText(code_mean_value_3_numbers))
ui.actionDurchschnitt_dreier_Zahlen_floor.triggered.connect(lambda: set_saved(False))
ui.actionDurchschnitt_beliebig_vieler_Zahlen.triggered.connect(lambda: ui.Editor.setPlainText(code_mean_value_arbitrary))
ui.actionDurchschnitt_beliebig_vieler_Zahlen.triggered.connect(lambda: set_saved(False))

# Datei
ui.actionNeu.triggered.connect(lambda: new())
ui.action_ffnen.triggered.connect(lambda: show_dialog_open())
ui.actionSpeichern.triggered.connect(lambda: save())
ui.actionSpeichernAls.triggered.connect(lambda: show_dialog_save_as())

# Ablauf
ui.action0s.triggered.connect(lambda: validate_action(ui.action0s))
ui.action1s.triggered.connect(lambda: validate_action(ui.action1s))
ui.action2s.triggered.connect(lambda: validate_action(ui.action2s))
ui.action3s.triggered.connect(lambda: validate_action(ui.action3s))
ui.action4s.triggered.connect(lambda: validate_action(ui.action4s))
ui.action5s.triggered.connect(lambda: validate_action(ui.action5s))
ui.action10s.triggered.connect(lambda: validate_action(ui.action10s))
ui.actionVNPhasen.triggered.connect(lambda: validate_action(ui.actionVNPhasen))

# Sonstiges
ui.Editor.textChanged.connect(lambda: set_saved(False))
ui.input_reg.setValidator(QRegExpValidator(QRegExp("^-?\d*\d+$")))
ui.input_reg.returnPressed.connect(lambda: input_pressed())
ui.button_start.clicked.connect(lambda: start())
ui.button_stop.clicked.connect(lambda: stop())
ui.button_weiter.clicked.connect(lambda: run_vn())
ui.Speicher.setStyleSheet("alternate-background-color: white;background-color: gainsboro;")

ui.Editor.setFocus()
main_window.show()
sys.exit(app.exec_())
```