

# Inhaltsverzeichnis

|  |   |
|--|---|
| Beschreibung der grafischen Benutzerschnittstelle..... | 2 |
| Menüleiste.....  | 3 |
| Toolbar.....   | 3 |
| Micro Controller (Input / Output).....                 | 3 |
| Memory Bank.....                                       | 3 |
| Programmablauf Sequenz.....                            | 3 |
| Statusregister und W-Register.....                     | 3 |
| Programmzyklen.....                                    | 3 |
| Auszüge der Programmlogik.....                         | 4 |
| MOVLW.....   | 4 |
| Struktogramm.....                                      | 4 |
| Beschreibung.....                                      | 4 |
| Code Snippet.....                                      | 4 |
| MOVWF.....   | 5 |
| Struktogramm.....                                      | 5 |
| Beschreibung.....                                      | 5 |
| Code Snippet.....                                      | 5 |
| ANDWF.....   | 6 |
| Struktogramm.....                                      | 6 |
| Beschreibung.....                                      | 6 |
| Code Snippet.....                                      | 6 |
| Fazit.....   | 7 |

## Beschreibung der grafischen Benutzerschnittstelle

The screenshot displays the PIC16F8X Simulator interface, which is divided into several main sections:

- Menüleiste (Menu Bar):** Located at the top, it includes 'File', 'About', and 'Help' menus.
- Toolbar:** Positioned below the menu bar, it contains buttons for 'Start', 'Step', 'Stop', and 'Restart'.
- Micro Controller (Input / Output):** A central panel showing the internal state of the PIC. It includes:
  - Registers:** A list of registers (RA2, RA3, RA4/T0CKI, MCLR, Vss, RBO/INT, RB1, RB2, RB3) with their current values and directions (e.g., RA2 <--> RA1).
  - Memory Bank:** A table showing memory addresses (00h to 07h) and their corresponding values (e.g., 00h: 0x0, 01h: 0xff).
  - Program Counter:** A section showing the current instruction being executed, including the command code and its arguments (e.g., 'commandCode=MOVWF, commandArg=17, line=23, label='movwf 11h, in W steht nun 11h, DC=?; C=?, Z=?').
- Status- und W-Register:** A table showing the status of various flags and registers (IRP, RP1, RP0, T0, PD, Z, DC, C, W-Regi..., Cycles) with their current values (e.g., IRP: 0, RP1: 0, RP0: 0, T0: 0, PD: 0, Z: 0, DC: 0, C: 0, W-Regi...: 0x00, Cycles: 0).
- Current File:** A text field at the bottom right showing the path to the current file being simulated (e.g., 'C:/Users/Philipp/Documents/production/hso.ra.ja...').

## Menüleiste

|              |   |
|--------------|---|
| File -> Open | hier können die jeweiligen Testprogramme des Micro Controllers geladen werden                 |
| About        | Hier können Sie Informationen über die Entwickler erhalten                                    |
| Help         | Hier können zum einen das Datenblatt und die Dokumentation der Java Software geöffnet werden. |

## Toolbar

|         |   |
|---------|---|
| Run     | Das Programm wird ausgeführt.                                       |
| Step    | Ermöglicht es dem Benutzer das Programm in Schritten zu durchlaufen |
| Stop    | Das Programm wird gestoppt  |
| Restart | Die Werte des Programms werden zurückgesetzt                        |

## Micro Controller (Input / Output)

Über das Micro Controller I/O können, bei anklicken der einzelnen Pins, die Eingänge und Ausgänge gesteuert werden.

## Memory Bank

Zur Veranschaulichung werden in einer Tabelle die beiden Speicherbänke (Memory Bank Zero und Memory Bank One) angezeigt. Dort können die jeweiligen Register Werte der Speicherbänke abgelesen werden.

Die Werte der Speicherbänke werden in Echtzeit aktualisiert. Programmablauf Tabelle

## Programmablauf

In der Programmablauf Tabelle werden alle einzelnen Programmbefehle dargestellt, dabei wird der aktuelle ausgeführte Befehl farblich markiert.

## Statusregister und W-Register

Im Statusregister werden alle 8-Bits einzeln mit deren Bedeutung veranschaulicht.

Daneben befindet sich das W-Register mit dem aktuellen enthaltenen Wert.

Beide Register werden in Echtzeit aktualisiert.

## Programmzyklen

Der Programmzyklus gibt an, wie viele Zyklen der Micro Controller bereits durchlaufen hat.

## Auszüge der Programmlogik

### MOVLW

#### Struktogramm

|  |
|--|
| Das Befehlsargument wird dem W-Register zugewiesen |
| Überprüfung des ZeroFlags                          |
| Befehlszyklen hochzählen                           |
| Programmcouter wird aktualisiert                   |

#### Beschreibung

Der Wert des Befehls wird in dem W-Register abgespeichert.

Falls dieser 0 ist muss das Zero Flag gesetzt werden, andernfalls wird es zurückgesetzt.

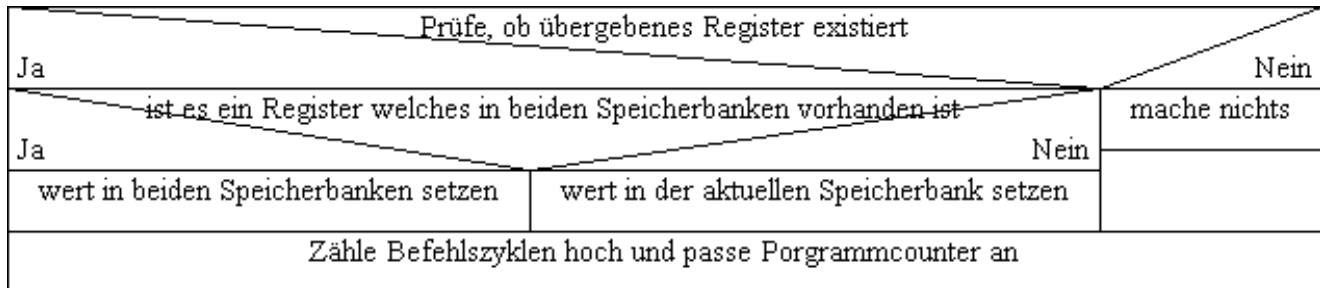
Anschließend werden die Befehlszyklen erhöht und der Programmzähler wird aktualisiert.

#### Code Snippet

```
case MOVLW:
    registerW = command.getCommandArg();
    checkZeroFlag(registerW);
    cycle++;
    programcounterInc();
    break;
```

## MOVWF

### Struktogramm



### Beschreibung

Zunächst wird überprüft, ob das übergebene Register existiert.

Falls dies nicht der Fall ist, werden nur die Befehlszyklen erhöht und der Programmzähler wird aktualisiert.

Falls das Register existiert, wird überprüft, ob dieses auf beiden Speicherbänken vorhanden ist.

Sofern es in beiden Speicherbänken existiert werden die Änderungen auf diese geschrieben.

Andernfalls wird es nur auf die ausgewählte Speicherbank geschrieben.

### Code Snippet

```
case MOVWF:
    if (command.getCommandArg() < 0x50) {
        if (Arrays.stream(equalRegister).filter(r -> r == command.getCommandArg()).count() > 0) {
            setRegisterValue(registerW, command.getCommandArg(), bankZero, areFlagsTargeted: false);
            setRegisterValue(registerW, command.getCommandArg(), bankOne, areFlagsTargeted: false);
        } else {
            setRegisterValue(registerW, command.getCommandArg(), getCurrentBank(), areFlagsTargeted: false);
        }
    }
    cycle++;
    programcounterInc();
    break;
```

## ANDWF

### Struktogramm

|  |
|--|
| W-Register UND-Verknüpft mit Befehlsargument |
| Zeroflag checken                             |
| Wert richtig abspeichern                     |
| Befehlszyklen und Programmcounter anpassen   |

### Beschreibung

Zunächst wird das W-Register UND-Verknüpft mit dem Befehlsargument.

Dieser Wert wird auf 0 überprüft und auf Basis dieses Ergebnisses wird das Zero Flag gesetzt oder gelöscht. Der Wert wird abhängig vom gesetzten Ziel Bit abgespeichert.

- Ziel Bit = 1. Der Wert wird in das Register gespeichert.
- Ziel Bit = 0. Der Wert wird in das W-Register gespeichert

Zum Schluss werden die Befehlszyklen erhöht und der Programmzähler aktualisiert.

### Code Snippet

```
case ANDWF:

    result = registerW & getRegisterValue((command.getCommandArg() & 0x7F));
    checkZeroFlag(result);

    safeValueInRegister(command, result, b: true, noWRegister: false);
    cycle++;
    programmcounterInc();
    break;
```

## Fazit

Das Projekt hat uns im Gesamten sehr gut gefallen, dies hat verschiedene Gründe.

Jedoch fangen wir erst mit den Kritikpunkten an.

Das Projekt hat einen hohen Zeitaufwand, welcher die vorgesehene Studienzeit dieses Kurses überschreiten könnte.

Durch das Abklären von Micro Controller spezifischen Fragen während des Labors und consequenten Arbeiten an dem Projekt ist es dennoch gut möglich dieses in vorgegebener Zeit umzusetzen.

Nun wenden wir uns den positiven Aspekten dieses Projekts zu.

Zum einen bestand dieses Labor nicht aus kleinen Teilaufgaben, sondern aus einem größerem Projekt.

Dadurch konnte man eigene Ideen in die Umsetzung einbringen.

Im Rahmen des Projekts konnte man viele Programmiertechniken anwenden, wie beispielsweise die Verwendung von Threads, der Umgang mit logischen Bit Operationen und das gestalten einer passenden grafischen Benutzeroberfläche.