PIC-Simulator: Dokumentation

Projekt im Rahmen der Vorlesung Rechnertechnik an der Dualen Hochschule Karlsruhe von

Jan Beck und Daniel Morgenstern

Diese Dokumentation beschreibt, wie wir die Vorgaben der Aufgabenstellung und die Spezifikation des PIC16F84 in unserem Simulator umgesetzt haben

# Inhalt

* Allgemeines
* Realisierung der Vorgaben und Spezifikationen
* Zusammenfassung

# Allgemeines

Arbeitsweise eines Simulators:

Ein Simulator bildet die Arbeitsweise eines Systems virtuell nach. Dazu müssen die ablaufe des Systems erfasst und in der virtuellen Umgebung modelliert werden. Eine Simulation kann dementsprechende nur so genau sein, wie die Kenntnis über das System. Abgesehen vom System selbst kann oder muss auch ein Teil der Umwelt simuliert werden, wenn diese relevante Einflüsse auf das System haben (z. B. externe Steuerung).

Vor- und Nachteile:

Ein Simulator kann genutzt werden um ein System zu testen, auch mit dem expliziten Ziel das System an seine Grenzen zu bringen, ohne die echte Hardware zu beschädigen. Außerdem können durch eine Anpassung der Simulationsgeschwindigkeit oder das Anhalten der Simulation an bestimmten Zeitpunkten einzelne Schritte eines Ablaufs genauer untersucht oder getestet werden. Des weiteren ist es oft einfacher und schneller Modifikationen eines Systems in einem Simulator zu testen. Tests im Simulator sind trotzdem nicht ausreichend, weil Ablaufe im Simulator abstrahiert werden und so nicht 100% exakt abgebildet werden können und weil nicht alle Umwelteinflüsse (z. B. Temperatur) simuliert werden können.

Handhabung

Die Handhabung des Simulators ist im ebenfalls beiliegenden Benutzerhandbuch beschrieben.

## Realisierung

Unser Simulator ist in C++ geschrieben, weil wir beide mit dieser Sprache vertraut sind. Er besteht aus zwei Teile: Dem Frontend, das die Steuerung der GUI übernimmt, und dem Backend, das Dateien mit Assemblerbefehlen einließt und ausführt. Beiden Teile sind jeweils eine eigene Klasse, GUI und Backend. Von beiden Klassen wird jeweils nur ein Objekt angelegt. Das GUI-Objekt wird direkt in der Main-Funktion angelegt, das Backend-Objekt ist eine private Variable der GUI und wird dementsprechend im Konstruktor der GUI erstellt. Für die Benutzeroberfläche wird die Bibliothek FLTK verwendet. Das Backend verwendet keine externen Bibliotheken.

FLTK übernimmt am Ende der Initialisierung den Hauptthread der Anwendung und ruft die Funktionen der GUI auf, wenn es nötig ist. Das Backend kann nicht direkt etwas an der Benutzeroberfläche ändern. Stattdessen ruft es die Funktionen int\_update() oder int\_updateAll() auf. Diese aktualisieren die Werte, die die Benutzeroberfläche anzeigen soll, und teilen FLTK dann mit, welche Teile der Benutzeroberfläche neu gezeichnet werden müssen. Die Interaktion mit dem Nutzer erfolgt über Callbacks. Das sind Funktionen, die bei bestimmten Teilen der Benutzeroberfläche hinterlegt sind und von FLTK aufgerufen werden, wenn der Nutzer mit diesem Teil interagiert, meist indem er darauf klickt. Diese Funktionen rufen dann meistens eine entsprechende Schnittstelle im Backend auf, um eine bestimmte Aktion durchzuführen, z. B. ruft der beim "Start"-Button hinterlegte Callback die Funktion "Start()" des Backend auf und die Funktion, die ausgelöst wird, wenn man auf den "Ändern" Button beim W-Register drückt, verwendet die Funktion "SetW()". Die Position und Größe der einzelnen Objekte in der Benutzeroberfläche hängt von der Schriftgröße ab, die verwendet wird. Diese wird am Programmstart anhand der verfügbaren Fensterbreite festgelegt, sodass alle wichtigen Teile sichtbar sind.

Das Backend stellt der GUI eine Reihe an spezifischen Schnittstellen zur Verfügung, mit der die GUI das Backend steuern kann. So gibt es zum Beispiel die Funktion LoadProgramm(char\* c), die eine neue Assemblerdatei einließt. Diese wird aufgerufen, wenn über den "Lade Datei"-Dialog eine Datei ausgewählt wurde.

Zur Ausführung der Assemblerbefehle erzeugt das backend einen eigenen Thread, in dem solange Assemblerbefehle ausgeführt werden, bis entweder ein externer Zugriff die Ausführung beendet (Funktion Stop() ) oder ein Breakpoint erreicht wird.