PFSIM - eine Simulationsplattform für den fiktiven Prozessor P/F
- Kurzbeschreibung -

Anwendung:
Editieren und Ausführen kleinerer Programme zu Einarbeitungs-, Lehr- und Demonstrationszwecken.

Wirkungsweise:
Direkte Interpretation des Assembler-Quelltextes (es gibt keinen Maschinencode).

Implementierung des Simulators:
Als Delphi-Projekt.

Ausführungsgeschwindigkeit:
Ziemlich gering. Für die vorgesehenen Zwecke auch weitgehend bedeutungslos (zumeist ohnehin Befehlsabarbeitung im Einzelschrittbetrieb).

Simulierter Speicher:

1. Befehlsspeicher: direkt aus Quelltext. (Schließt Selbstmodifikation von Programmen aus). Maximale Länge ca. 40 kBytes.
2. Datenspeicher: 256 Worte zu 32 Bits, wahlweise als Liste oder Tabelle. Anzeige über PRAGMA-Befehl zurückschaltbar auf 16 Bytes (z. B. für Speichertestprogramme).
3. Bildspeicher des Übungsbildschirms. 480 Worte zu 32 Bits. Belegung der niedrigstwertigen Bytes wird auf Übungsbildschim als Zeichen dargestellt.

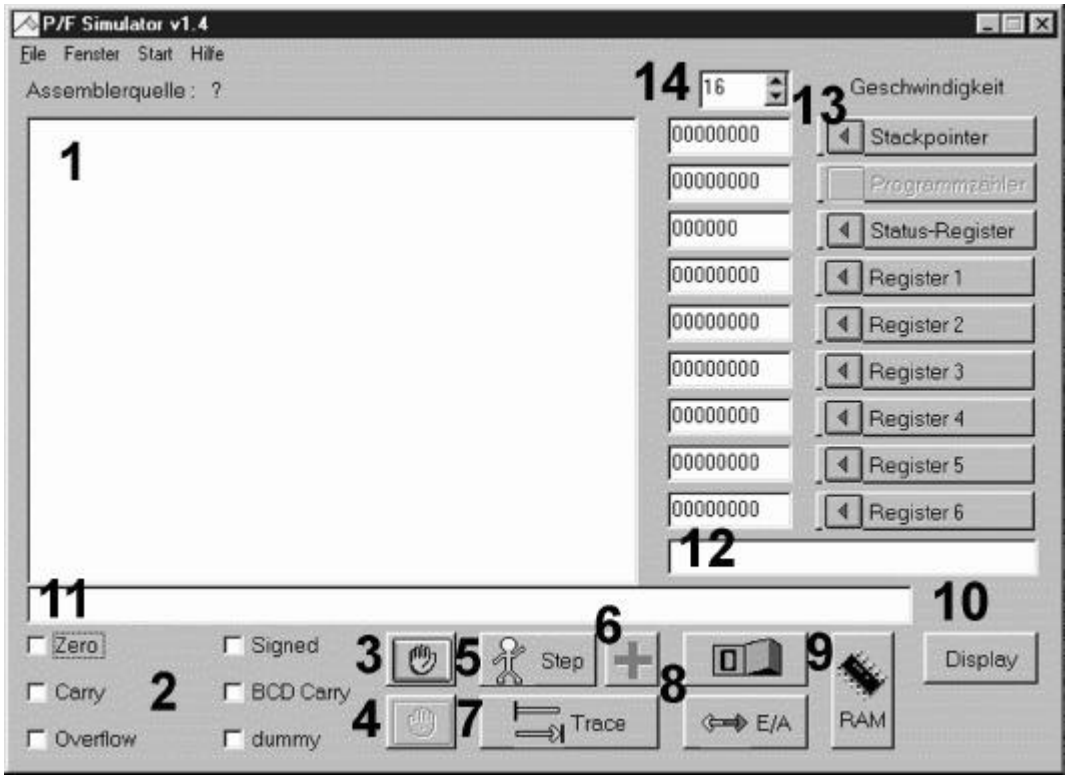
Simulierte Ein- und Ausgabe:

1. 18 E-A-Ports zu einem Byte (gleichbedeutend: 18 32-Bit-Ports, in denen nur die Bits 7...0 belegt sind).
2. Eingabemittel (1): 19 Kippschalter, die wahlweise mit jedem Anschluß (Pin) der 18 Ports verbunden werden können.
3. Eingabemittel (2): die Tastatur des PCs (über Port 18), falls Übungsbildschirm den Fokus hat.
4. Ausgabemittel: binäre Portanzeige in gesonderter Box (E/A-Schnittstellenansicht).
5. Vereinfachung: Befehl PRAGMA IO bewirkt, daß die Schalter 1...16 mit den Ports 1 und 2 verbunden und daß die Ports 3...6 sowohl binär als auch hexadezimal in der Kommentarzeile angezeigt werden.
6. Richtungssteuerung: gibt es nicht. Ausgabe des Prozessors dominiert über Eingabe.

Anwendungsprogrammbezeichner: PFSIM.EXE

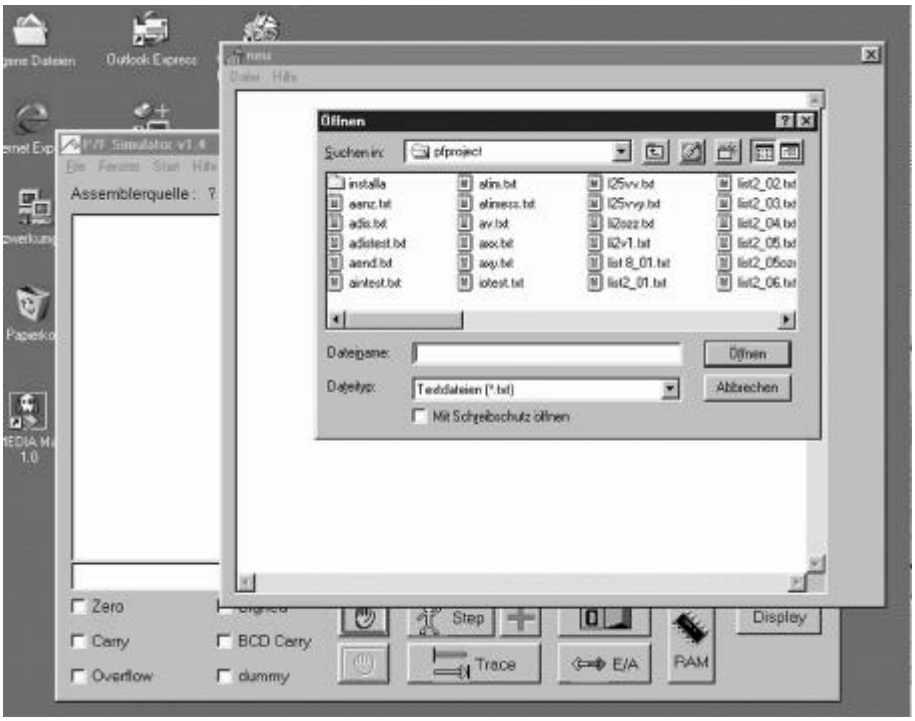
Assemblerdateien: ACSII-Textdateien.

Grundbild der Simulationsplattform:



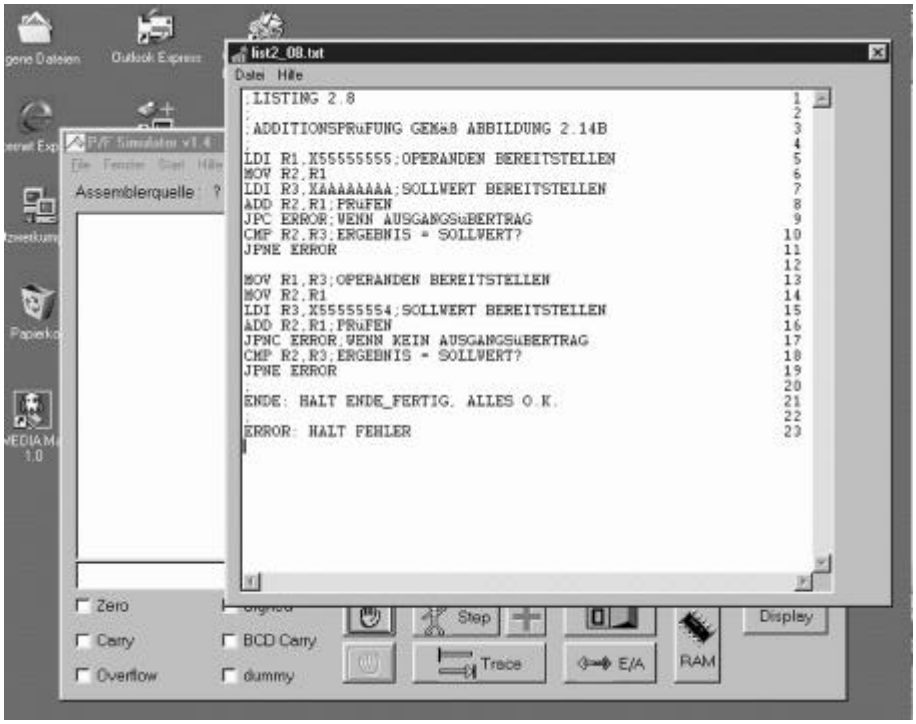
Erklärung:
1 - Quelltextfenster; 2 - Flagbits; 3 - Starttaste; 4 - Rücksetztaste; 5 - Betriebsartenwahl Schrittbetrieb (= Anhalten des Programmablaufs) 6 - Schrittaste (Single Step); 7 - Start der bereichsweisen Programmabarbeitung; 8 - Auswahl der E-A-Anzeige; 9 - Auswahl der Speicheranzeige; 10 - Auswahl des Übungsbildschirms; 11 - Kommentarzeile; 12 - Nachrichtenzeile; 13 - Registeranzeige und -auswahl; 14 - Steuerung der Ablaufgeschwindigkeit.

Dateidialog:



Dieser Dialog erscheint über *File - Öffnen* aus dem Grundbild. Bei “Abbrechen” erscheint ein leeres Editierfenste. Bei “Öffnen” wird die ausgewählte Datei geladen.

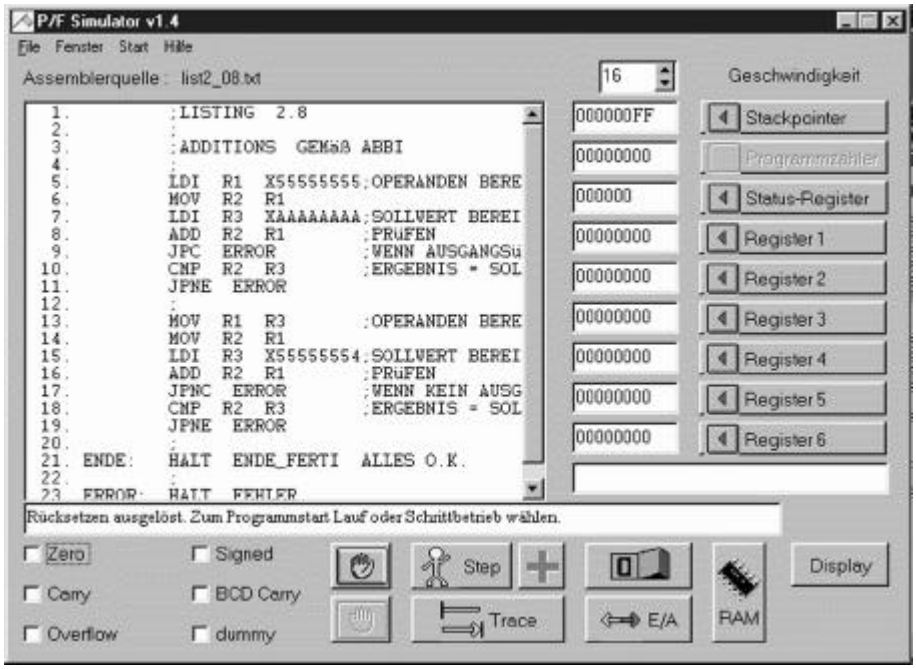
Eine Textdatei im Editierfenster:



Zur Eingabesyntax:

1. Befehlsmnemonic: gemäß Architekturbeschreibung.
2. Groß- und Kleinschreibung: ist beliebig. Kleinbuchstaben werden nach dem Schließen bzw. vor dem Abspeichern in Großbuchstaben gewandelt.
3. Abstände zwischen Sprungmarken (Labels), Befehlen und Kommentaren: können entfallen. Sie werden nach dem Schließen automatisch eingefügt.
4. Zeilennummern: werden automatisch eingefügt bzw. vor dem Abspeichern entfernt (gespeicherte Dateien enthalten keine Zeilennummern). Ersteingabe ohne Zeilennummern.
5. Anzeige der Zeilennummern im Editierfenster: ist über Punkt “Zeilennummer” im Dateidialog steuerbar (Ein- und Ausschalten). Nach dem Öffnen oder Speichern wird die Zeilennummer automatisch angezeigt.
6. Achtung: die Anzeige der Zeilennummern im Editierfenster ist eine Komfortfunktion. Die gesamte Zeile einschließlich der Zeilennummer wird als *eine* Zeichenkette betrachtet. Zeilennummern sind Zeichen wie andere auch. Geraten diese beim Editieren zu weit nach links, so kann die entstehende Zeichenkette während der Programmausführung als fehlerhaft erkannt werden.
7. Aus- und Wiedereinschalten der Zeilennummern: ist während des Editierens möglich (Position „Zeilennummer“ im Dateidialog).
8. Speicherung der editierten Dateien: ohne Zeilennummern. Jede Zeile hat nir ihre minimale Länge (Leerzeichen rechts vom Text werden entfernt). Um die Zeilenanordnung zu erhalten (wichtig, weil es keine ORG-Anweisungen gibt), wird anstelle jeder Leerzeile eine Zeile mit einem Semikolon (;) gespeichert.
9. Kommentarkennung: Am Zeilenanfang bzw. nach dem Befehl durch Semikolon (;).
10. Alternative Kommentarkennung: (für Pseudo-Befehle): -- (Ada-Stil). -- vor der Befehls-Mnemonic macht den Befehl zum NOP.
11. Leerzeilen: werden durch automatisches Einfügen von Semikolons zu Kommentarzeilen = NOPs. Diese Semikolons erscheinen aber nicht im Editierfenster und werden auch nicht mitgespeichert.
12. Direktwerteingabe dezimal: nur Ziffern. Größer Wert: 2 147 483 647; kleinster Wert: - 2 147 483 647.
13. Direktwerteingabe HEX: durch vorangestelltes x.
14. Sprungmarke: am Zeilenanfang. Mit Doppelpunkt (:) abschließen. Maximale Länge: 6 Zeichen zuzüglich Doppelpunkt.
15. Achtung: Sprungmarke ohne Befehl ist unzulässig! (Zulässig ist aber MARKE:; bzw. MARKE:--).
16. Achtung: PFSIM ist ein Interpreter und kein echter Assembler. Es wird nicht erkannt, wenn Sprungmarken mehr als einmal vorkommen. Sprungziel ist dann stets die in der Reihenfolge der Zeilen erste dieser Marken. Nicht als Marken vorhandene Sprungziele und manche weiteren Inkorrektheiten werden erst während der Ausführung erkannt. Es gibt auch keine ORG- und EQ-Anweisungen.

Programm fertig zur Ausführung:



Durch Schließen des Editierfensters wird das Programm zwecks Ausführung im Quelltextfenster bereitgestellt.

Programmausführung:

1. Lauf (kontinuierliche Ausführung gemäß eingestellter Geschwindigkeit): Betätigen der “grünen Hand”.
2. Schrittbetrieb (befehlsweise Ausführung): durch Betätigen von *Step*; dann weiter mit “Plustaste”.
3. Wechsel zwischen Lauf- und Schrittbetrieb sowie Rücksetzen (“rote Hand”) ist jederzeit möglich.
4. Anhalten eines laufenden Programms: durch Anklicken von „Step“.
5. Anklicken einer Befehlszeile im Schrittbetrieb bewirkt Übergang zu diesem Befehl.
6. Ausführungsgeschwindigkeit: über Geschwindigkeitswahlfenster einstellbar (Hoch-Runter-Prinzip). 1...20 sind einzelne Schrittgeschwindigkeiten.
7. Programmablauf: ist am Wandern der Zeilenmarkierung (kennzeichnet zur Ausführung jeweils anstehenden Befehl) im Quelltextfenster sowie am Schalten der Registerbelegungen und Flags erkennbar. Befehlswirkung wird jeweils in der Kommentarzeile beschrieben.
8. Schnelldurchlauf: Schrittgeschwindigkeit 0. Dabei entfällt das Bewegen der Zeilenmarkierung sowie das Beschreiben der Befehle in der Kommentarzeile.
9. Kommentarzeilen: werden nicht übergangen, sondern wie NOPs durchlaufen.

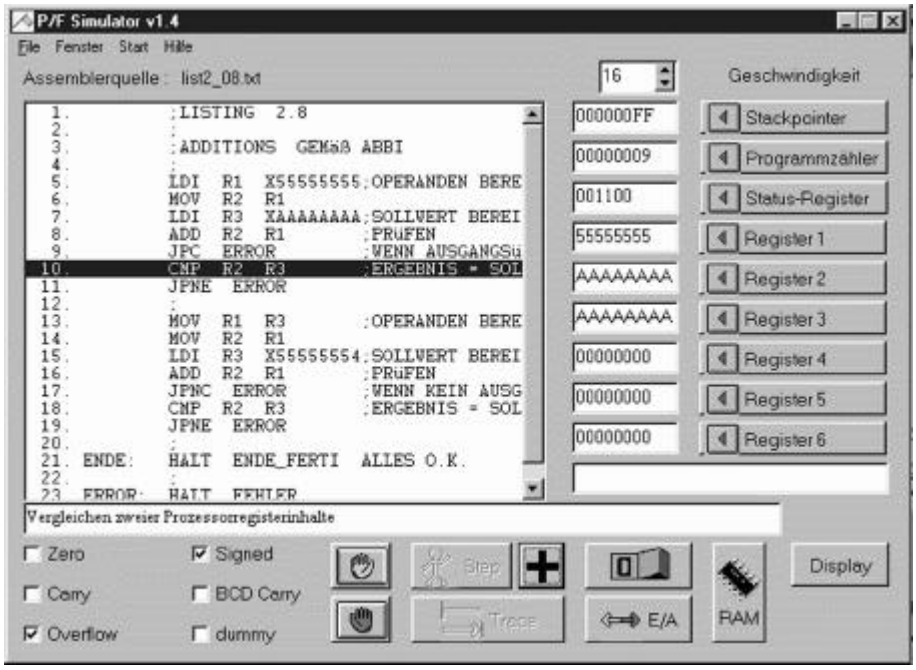
Eingriffsmöglichkeiten:

1. Register R1...R6: bei Anklicken des jeweiligen Feldes erscheint ein Eingabefenster. Eingabe: dezimal (nur Ziffern) oder hexadezimal mit x als erstem Zeichen.
2. Programmzähler (Befehlszähler): Eingabefenster erscheint nur bei laufendem Programm (nicht im Rücksetzzustand). Eingabe der Zeilennummer (nur dezimal).
3. Stackpointer: wie Register R1...R6.
4. Statusregister: keine direkte Eingabe. Flags sind aber einzeln zu setzen.

Wirkung des Rücksetzens:

1. Register R1...R6 und Programmzähler: => Inhalt 0.
2. Stackpointer => Inhalt FFH.
3. Speicherinhalt: wird unbestimmt (-).
4. Portinhalt: wird Null.
5. Schalterzuordnung: bleibt erhalten.

Programm in Ausführung:



Ersichtlich sind:

- der aktuelle Befehl,
- die Flagbelegung,
- die Registerinhalte,
- die Befehlsbeschreibung in der Kommentarzeile.

Sondervorkehrungen:

1. Auswahl: Menue-Punkt *Start* im Grundbild.
2. Adreßvergleichsstop: über Dialog Haltepunkt. Zeilennummer eingeben, an der die Befehlsabarbeitung angehalten werden soll. Es kann jeweils nur 1 Haltepunkt (Hardware-Breakpoint) gesetzt werden). Ausschalten eines Haltepunkts: durch Eingabe einer Programmzeile, die nie erreicht wird (auch: Zeile 0).
3. Trace-Funktion: Ausführung der Befehle zwischen einer einzugebenden Start- und Stopmarkierung (jeweils die Nummer der Programmzeile). Nach Eingabe von Start- und Stopmarkierung kann Abarbeitung durch Anklicken von “Trace” gestartet werden. Verläßt der Programmablauf den so ausgewählten Bereich, wird die Programmausführung beendet.

Befehle zur generellen Programmsteuerung und zur Fehlersuchunterstützung:

1. HALT *Message*. Die dem Befehl nachfiolgende Zeichenkette *Message* wird in der Nachrichtenzeile dargestellt. Der Programmablauf wird beendet.
2. CHECKPOINT (Kurzform CHK) Message. Die dem Befehl nachfolgende Zeichenkette *Message* wird in der Nachrichtenzeile dargestellt. Der Programmablauf wird nicht beeinflußt.
3. BREAKPOINT (Kurzform BRK). Bei Erreichen eines Breakpoints wird der Programmablauf zunächst angehalten. Es erscheint ein Dialog, der es ermöglicht, zwischen Fortsetzen (Laufzustand) und Übergang in den Schrittbetrieb zu wählen.
4. PRAGMA *Message*. Die dem Befehl nachfolgende Zeichenkette *Message* steuert bestimme Einzelheiten der Simulationsumgebung. PRAGMA-Befehle dürfen mitten im Programm gegeben werden.

Hinweise:

1. Ist keine *Message* angegeben, so wird die Anzeige in der Nachrichtenzeile gelöscht.
2. Die Message endet jeweils mit dem folgenden Leerzeichen.

Messages des PRAGMA-Befehls:

IO	-	Einfachversion der E-A-Darstellung. Schalter 1...16 werden mit den Bits 7...0 der Ports 1 und 2 verbunden. Die Schalter werden dabei auf Null zurückgestellt. Die Ports 3, 4, 5, 6 werden binär und hexadezimal in der Kommentarzeile angezeigt. (Die Befehlskommentierung entfällt.) Es ist möglich, die Schalterzuordnung manuell zu ändern.
IOOFF	-	Vollversion der E-A-Darstellung. Alle Schalter können freizügig verbunden werden. Portanzeige in Box *Portinhalte*.

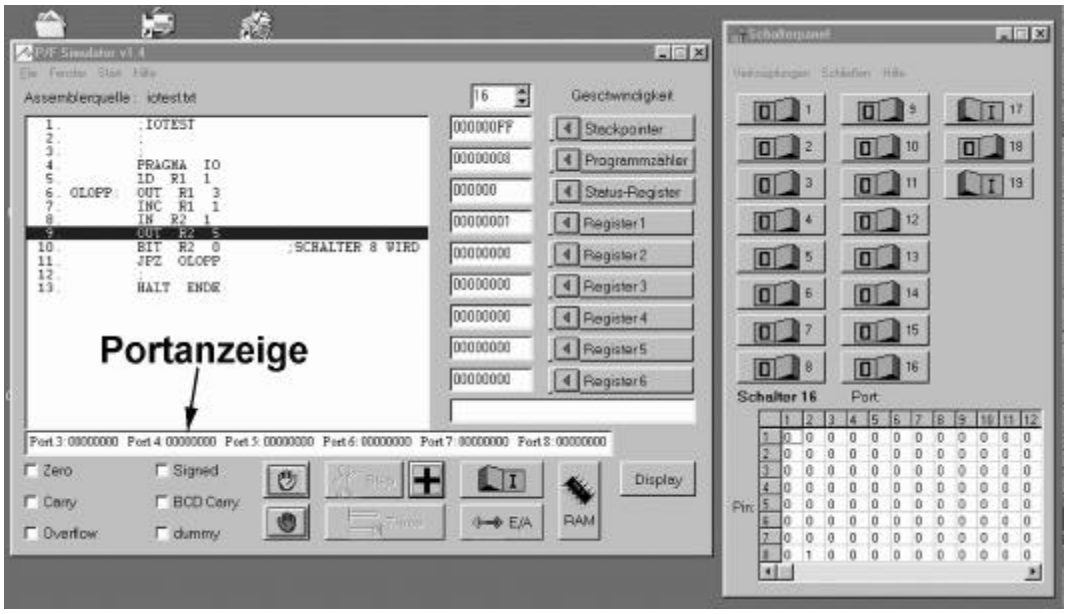
MEMINIT - werden nicht initialisierte Speicherinhalte gelesen, so wird dies in der Nachrichtenzeile angezeigt. (Die Nachricht bleibt bis zum nächsten Befehl sichtbar, der in die Nachrichtenzeile schreibt. Programmiertip: Löschen z. B. mit Befehl CHK.)

MEMINITOFF - auf das Lesen nicht initialisierter Speicherinhalte wird nicht reagiert. *Hinweis:* Nich initialisierter Speicherinhalte werden als Nullen gelesen.

4 - Speicherdarstellung als 4·4-Matrix (die ersten 16 Zellen - Adressen 0...15 - werden angezeigt).

16 - Speicherdarstellung als 16·16-Matrix (256 Zellen; Adressen 0...255 (FFH)).

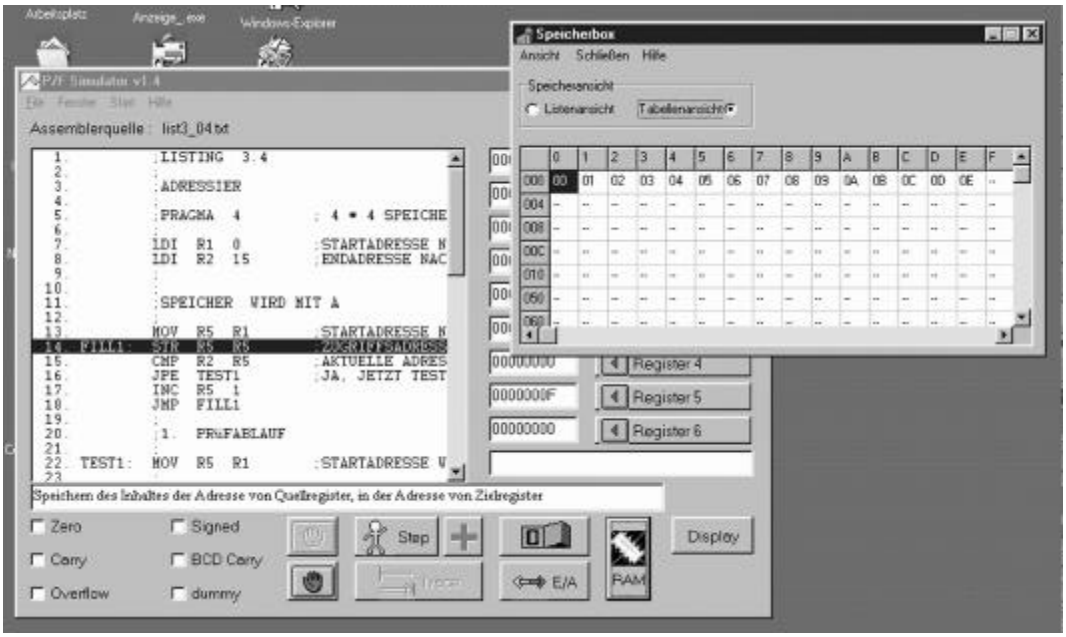
Schalteranzeige:
Hier ist zusätzlich die Portanzeige in der Kommentarzeile dargestellt (Wirkung von PRAGMA IO):



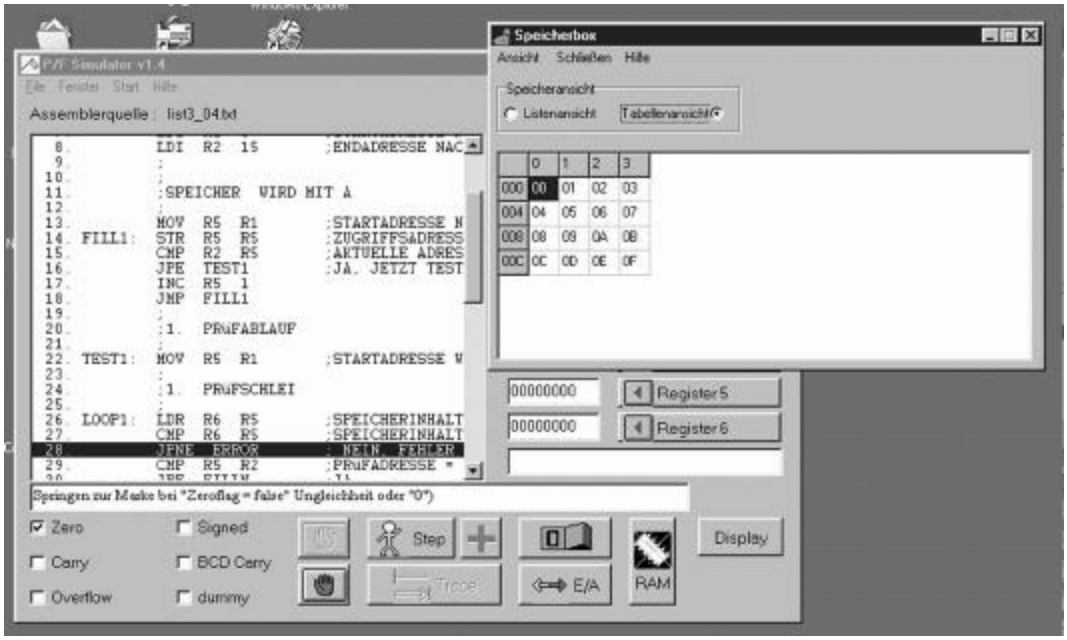
Schalterauswahl in der Verbindungsmatrix (im Schalterfeld unten):

1. Gewünschten Schalter anklicken.
2. Die Matrix muß Nullen und Einsen enthalten.
3. Matrixinhalt: in jeder Zelle nur 0 oder 1.
4. Matrixinhalt ändern: durch wechselseitiges Anklicken. Tastatureingaben sind möglich, aber wirkungslos!
5. Alternative Zuordung der Port-Anschlüsse zu den Schaltern: über die Verknüpfungstabelle in der E-A-Schnittstellenansicht.

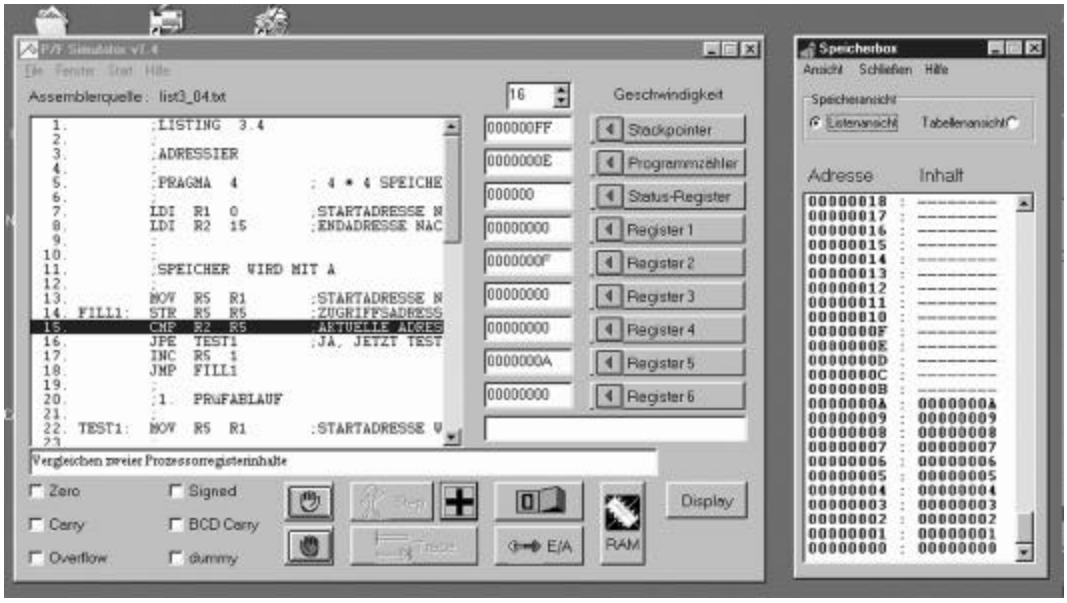
Speicheranzeige als 16·16-Matrix (Default bzw. Wirkung von PRAGMA 16):



Speicheranzeige als 4·4.Matrix (Wirkung von PRAGMA 4):



Speicheranzeige als Liste



Zur Speicheransicht:

1. Der Speicher umfaßt 256 Zellen zu 32 Bits. Adressen: von 00H bis FF H (0...255).
2. Es sind stets 256 Zellen vorhanden, auch wenn mit PRAGMA 4 die Anzeige auf die ersten 16 Zellen beschränkt wurde.
3. Der Stackpointer wird auf Adresse FFH initialisiert. Der Stack wächst in Richtung niederer Adressen.
4. Listenansicht: es wird der Speicherinhalt vollständig dargestellt (32-Bit-Worte; hexadezimal).
5. Tabellenansicht: es werden nur die Bits 3...0 der 32-Bit-Worte dargestellt (hexadezimal).
6. Tip: für Speichertestprogramme die Tabellenansicht verwenden. Flimmert nicht bei zyklischem Schreiben.

Speicheraufteilung

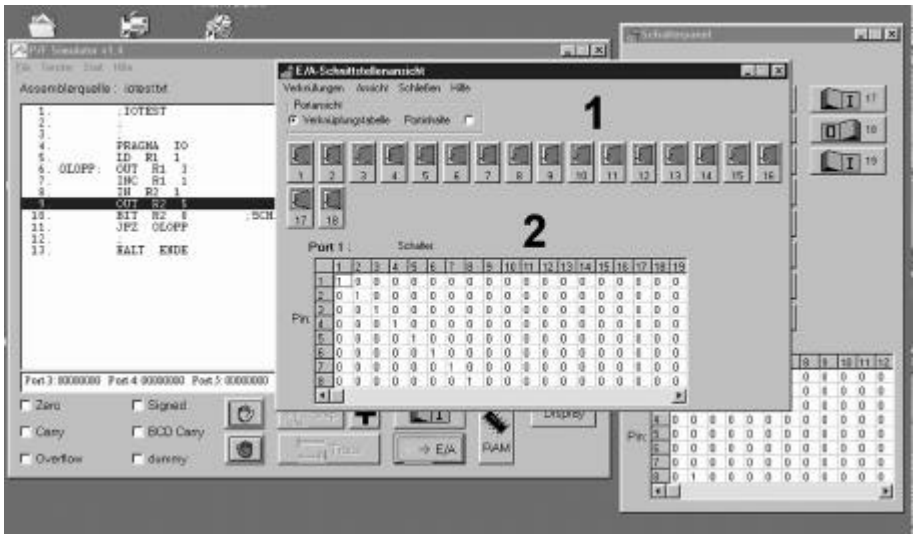
1. Arbeitsspeicher:

- Anfangsadresse: 0,
- Endadresse: 255 oder 15.

2. Bildspeicher (Übungsbildschirm):

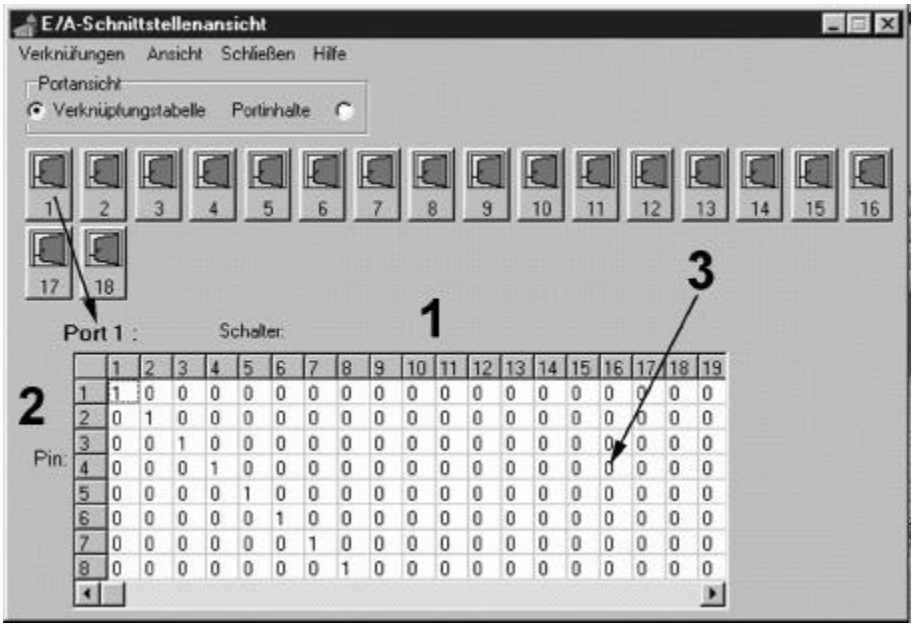
- Anfangsadresse: 8192,
- Endadresse: 8672,
- Größe: 480 Zellen (= 32-Bit-Worte = Zeichenpositionen),
- Anzeigeformat: 12 Zeilen zu 40 Zeichen. Das niedrigstwertige Byte des jeweiligen 32-Bit-Wortes wird als Zeichen angezeigt.

E-A-Schnittstellenansicht (Verknüpfungstabelle):



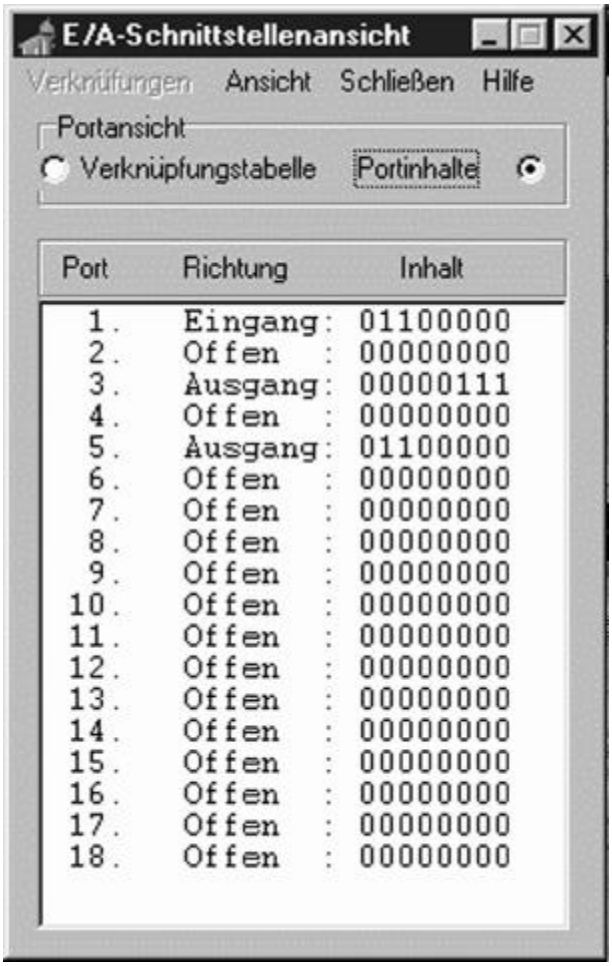
Erklärung:
1 - Portauswahl; 2 - Verknüpfungstabelle. Für jeden der 18 Ports gibt es eine solche Tabelle, über die jede der 8 Bitpositionen des Ports (Bits 7...0) mit jedem der 19 Schalter verbunden werden kann.

Einzelheit Verknüpfungstabelle:

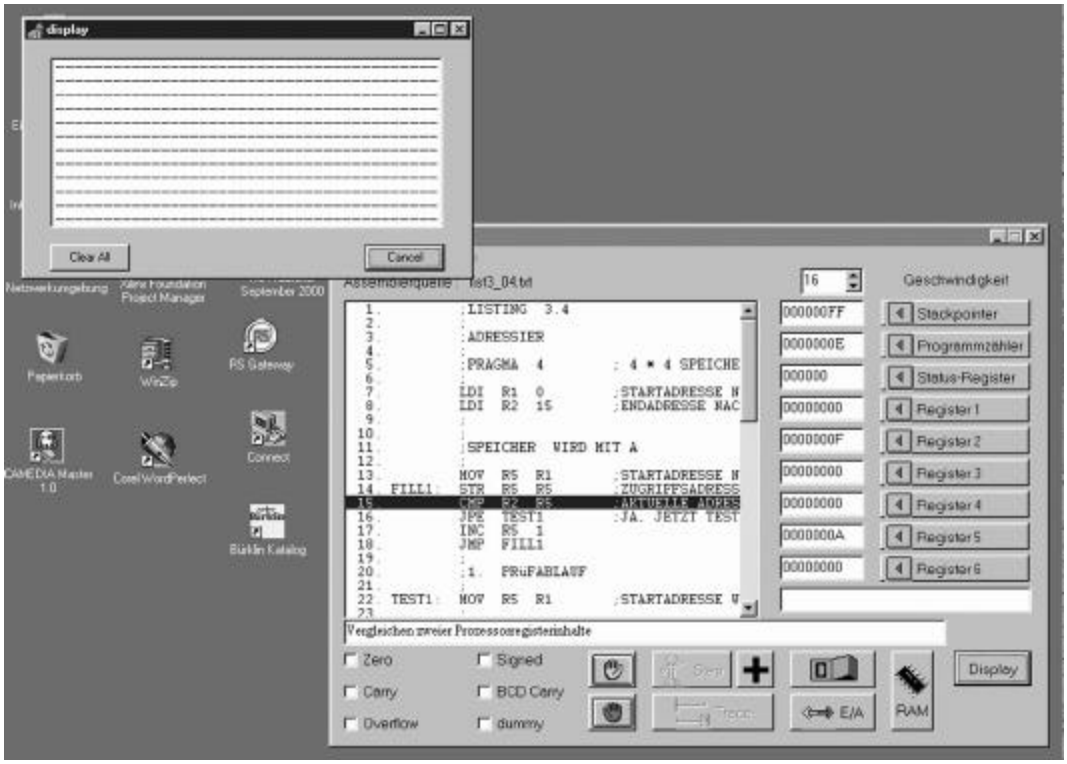


Erklärung:
Hier wurde die Verknüpfungstabelle des Ports 1 ausgewählt (Pfeil). 1 - Schalter 1...19; 2 - die Pins 1...8 entsprechen den Bitpositionen 0...7; 3 - wird diese Position angeklickt, so wird Pin 4 mit Schalter 16 verbunden. Siehe auch S. 8 unten (Schalterauswahl).

Ansicht Portinhalte:



Der Übungsbildschirm:



Tastatureingabe: Über E-A-Port 18. Unbetätigte Tastatur liefert 00000000H zurück.

Wirkung der Umschalttasten:

zusätzlich SHIFT: Tastencode + 100H,
zusätzlich CTRL: tastencode + 200H,
zusätzlich ALT: Tastencode + 400H.

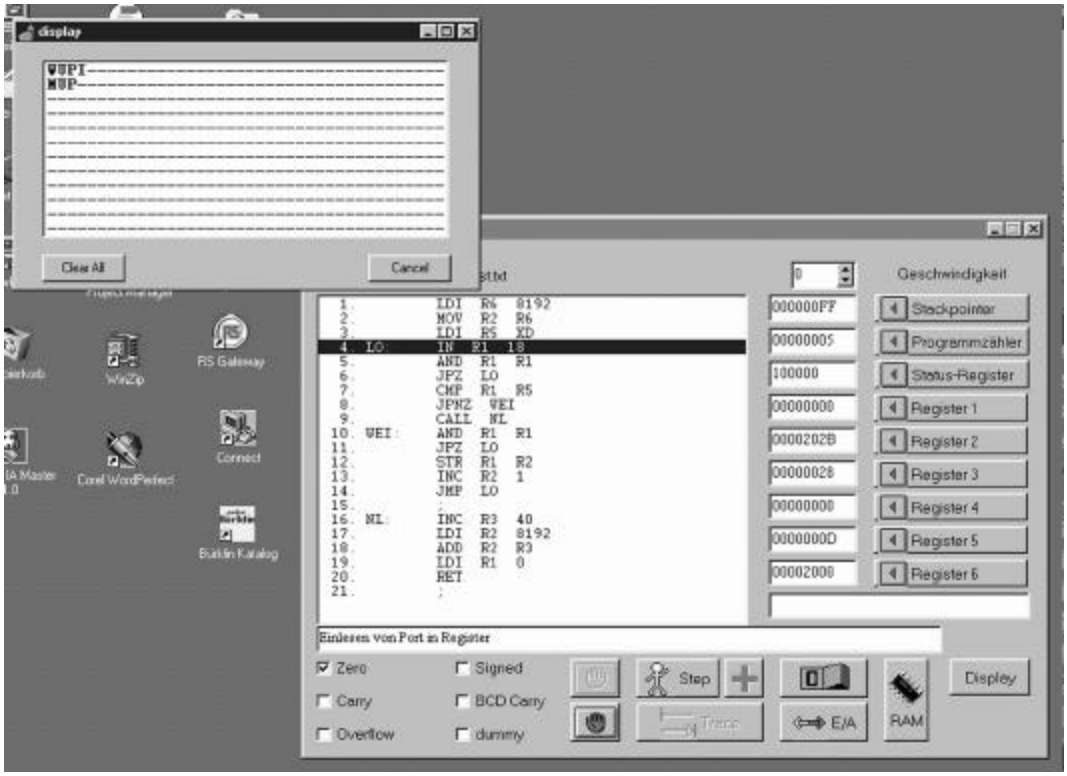
Codes von Funktionstasten:

Cursor Left:	25H	Enter:	0DH	Bild hoch:	21H	F2:	71H
Cursor Right:	27H	Einf:	2DH	Bild runter:	22H	F9:	78H
Cursor Up:	26H	Entf:	2EH	Ende:	23H	F11:	7AH
Cursor Down:	28H	Pos1:	24H,			F12:	7BH

F1: keine Wirkung,
F10: Umschalten (Windows-Funktion)

Achtung:
Das Displayfenster muß den Fokus haben, damit alle Tasten richtig wirken. Ggf. auf die Bildschirmfläche des Displayfensters klicken.

Anzeige auf Übungsbildschirm:



Dienstvorschrift zum Installieren:

1. Komplettes Verzeichnis PFSIMEX4 auf Festplatte kopieren.
2. Starten durch Doppelklick auf PFSIM.EXE.

PFSIM.EXE kann auch direkt von der CD gestartet werden.

Entinstallieren
Die Software kopiert sich nirgendwo hin. Deshalb genügt es, das Verzeichnis PFSIMEX4 zu löschen.

Quellcode
Dieser befindet sich im Verzeichnis PFDEV1_4. Bei Bedarf auf die Festplatte kopieren.

Auf Festplatte übernommene Dateien
Falls sie geändert werden sollen (Überschreiben): *Schreibschutz entfernen* (Windows Explorer - Datei wählen - Datei - Eigenschaften).

Dienstvorschrift zum Starten des ersten Programms:

1. PFSIM starten.
2. *File* anklicken.
3. *Öffnen* anklicken.
4. Editierfenster mit Dateidialog erscheint.
5. Passendes Programm anklicken oder Programmname eintippen. Beispiel: Bildein*).
6. Editierfenster schließen (Kreuz rechts oben oder *Beenden* im Dateidialog).
7. Lauf (grüne Hand) oder Schrittbetrieb (Step) wählen.
8. Bei Lauf warten, bis Programm fertig ist. Bei Schrittbetriebdurch Anklicken der Plustaste (+) Befehl für Befehl vorantasten.
9. Ablaufgeschwindigkeit: kann durch Anklicken der Hoch-Runter-Pfeile im Geschwindigkeitsfeld verändert werden. 1 = langsamster, 20 = schnellster Durchlauf (mit vollständiger Anzeige). 0 = Schnelldurchlauf ohne Ablaufanzeige und Kommentierung.
10. Rest: erklärt sich von selbst. Die Übung bringt‘s.

*): ein einfaches Bildschirmeingabeprogramm - als Grundlage, um es nach und nach zu verbessern (Cursordarstellung, Cursorbewegung usw.)...

Viel Erfolg!