Rechnerstrukturen

7. Assembler

Inhalt

- Assemblerprogrammierung
- SML-CPU
- SML-Assembler
  - Adressierungsarten
  - Instruktionssatz
  - Assembler-Direktiven
- Binden

7.2

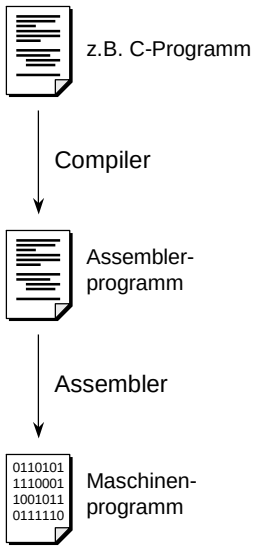
Motivation

- Maschinensprache
  - Instruktion = Bitkombination
  - Für Menschen ungeeignet
- Assembler
  - Symbolische Instruktionen (Mnemonic)
  - 1:1-Abbildung auf Maschinensprache (Assembler)
- Eigenschaften
  - Einfache Instruktionen
  - Einfache Datentypen
    - Bit, Byte, Wort, Langwort, ...
    - Integer, Float, Boolean
    - Zeiger
  - Unmittelbarer Zugriff auf Hardware
- Hochsprachen
  - Symbolisch
  - Komplexe Sprachelemente
  - Übersetzung (Compiler)
- Eigenschaften
  - Vielfältige Sprachelemente
  - Einfache Datentypen
    - Integer, Float, Boolean, ...
    - Zeiger (sprachabhängig)
  - Zusammengesetzte Typen
    - Record, Feld, ...
  - Eigene Datentypen (z.B. OO)
  - Höhere Abstraktionen verbergen Hardware
    - nicht alle Instruktionen verwendbar
    - eingeschränkter HW-Zugriff

7.3

Assembler

- Compiler erzeugen meist Assemblerquellen
- Nachfolgender Assemblerlauf produziert Maschinenprogramm
  - Modul
  - Bibliothek
  - ausführbares Programm
- Explizite Assemblerprogrammierung selten
  - Systemsoftware
    - Zugriff auf Prozessorzustände
    - Interruptbehandlung
    - Virtuelle Speicherverwaltung
    - Zugriff auf E/A-Controller
    - ...
  - Optimierung
    - Compiler sind mittlerweile schwer zu schlagen



7.4

Bestandteile eines Assemblerprogramms

- Instruktionsfolge(n)
  - sichtbarer Prozessorzustand (Register, ...)
  - Instruktionssatz
  - Adressierungsarten
- Datenstrukturen
  - Globale Datenbereiche
- Speicherlayout
  - Positionierung der Instruktionen
    - RAM- und ROM-Bereiche der Zielhardware?
    - Instruktionslänge auf Assemblerebene unbekannt
    - Zieladresse bei Sprüngen?
  - Positionierung der Datenbereiche
  - Benennung ausgezeichneter Positionen durch Symbole

7.5

Eigeninitiative gefragt ...

- Prozeduren und Funktionen
  - Übergabemechanismus Argumente
  - Prozedurlokale Variablen
    - Rekursion!
  - Übergabemechanismus Rückgabewert
  - Hilfen
    - eventuell spezielle Instruktionen
- Komplexe Datentypen
  - Speicheranordnung
  - Zugriffsfunktionen
  - Hilfen
    - eventuell spezielle Instruktionen

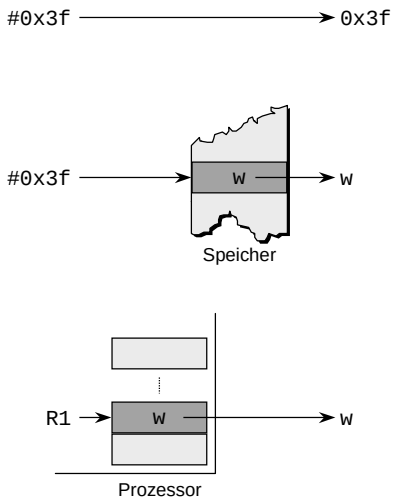
7.6

Adressierungsarten

- Bestimmung des (der) Operanden
- Anforderungen
  - Konstanten
  - Register
  - Direkter Zugriff auf Speicher
  - Adreßberechnung (indirekter Zugriff)
- Wünschenswerte Adressierungsarten
  - Zugriff auf Recordelemente
  - Indizierter Feldzugriff
  - ...
- Implizite Adressierungsarten
  - Adressierungsart beim Operandenzugriff implizit in Instruktion enthalten

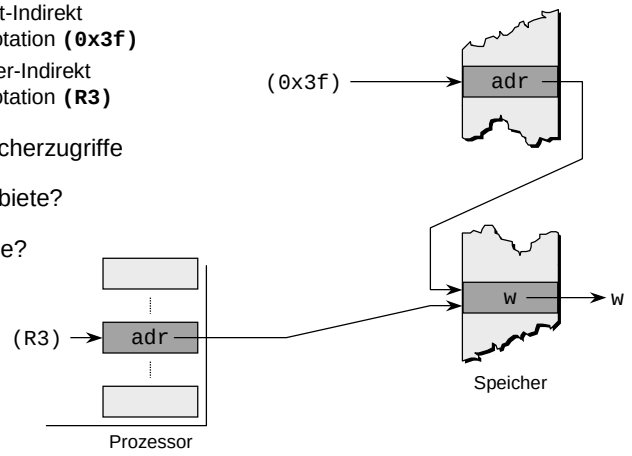
Immediate-, Absolute- und Register-Adressierung

- Immediate (Konstante)
  - Quelloperand ist unmittelbarer Bestandteil der Instruktion
  - Notation **`#0x3f`**
- Absolut
  - Ziel- oder Quelloperand befindet sich an der angegebenen Adresse im Hauptspeicher
  - Notation **`0x3f`**
- Register
  - Ziel- oder Quelloperand befindet sich im angegebenen Register
  - Notation **`R1`**



Indirekte Adressierung

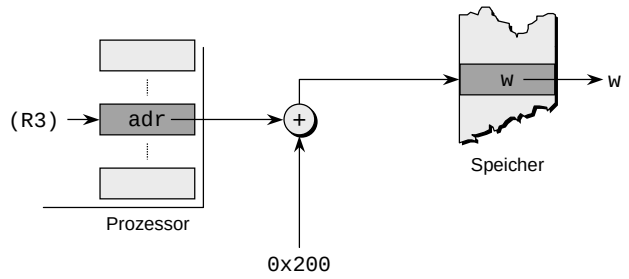
- Varianten
  - Absolut-Indirekt
    - Notation **(0x3f)**
  - Register-Indirekt
    - Notation **(R3)**
- Zwei Speicherzugriffe
- Einsatzgebiete?
- Sonderfälle?



7.9

Register-Indirekt mit Displacement

- Notation **(R3)+0x200**
- Displacement ist 2er-Komplement
- Sonderfälle
  - Instruktionen mit impliziter PC-indirekter Adressierung



7.10

Indiziert-Indirekt

- Zugriff auf Tabellen mit gleich großen Elementen
  - Adresse des Tabellenanfangs (t)
  - Index (i)
  - Adresse des Quell- oder Zieloperanden

    $$adr := t + i \cdot size$$

  - Größe ergibt sich aus der Instruktion
- Tabellenanfang Absolut oder Register
- Index meist nur Register
- Notation
  - **`0x2000[R3]`**
  - **`R4[R5]`**

7.11

Instruktionssatz

- Ausschnitt SML-Befehlssatz
- Beispiel **`MOVE`**
- Allgemeine Syntax:
  `Move.[B|W|L] <lval>, <rval>`
  - Extension B, W oder L bestimmt Operandengröße
  - <lval>: jede Adressierungsart außer Immediate
  - <rval>: jede Adressierungsart
- Semantik: <lval> := <rval>
- Beispiele
  - **`Move.B R3,#100`**
  - **`Move.L R5,R3`**
  - **`Move.W 0x2000[R5]`**

7.12

Weitere Instruktionen

- Arithmetische Operationen
  - Addition, Subtraktion, Inkrement, Dekrement
  - Realisierung Multiplikation und Division
  - Compare-Operation
- Logische Operationen
  - AND, OR, XOR, NOT
- Konvertierung
  - Byte auf Wort, Wort auf Langwort
  - Vorzeichenerweiterung (sign extend)
- Kelleroperationen
- Sprung- und Unterprogrammbefehle
- Schiebe- und Rotationsbefehle

7.13

Assembler-Direktiven

- Setzen des Positionzeigers
  - Legt jeweils die weiteren Adressen fest
  - Initial 0
  - Syntax: **`Loc <Wert>`**
- Explizites Deklarieren und Definieren von Symbolen
  - Syntax: **`Set <Symbol> [Typ] <Wert>`**
- Symbolsichtbarkeit
  - Syntax: **`Export <Symbol>`**
  - Syntax: **`Import <Symbol>`**
- Reservieren von Speicherbereichen
  - Syntax: **`Ds <Anzahl Bytes>`**
- Reservieren und Initialisieren von Speicherbereichen
  - Syntax: **`Def <Wert> {, <Wert> }`**

7.14