

Kapitel 1 – Grundlagen

1. Mathematische Grundlagen

2. **Beispielrechner ReTI**

Albert-Ludwigs-Universität Freiburg

Prof. Dr. Christoph Scholl

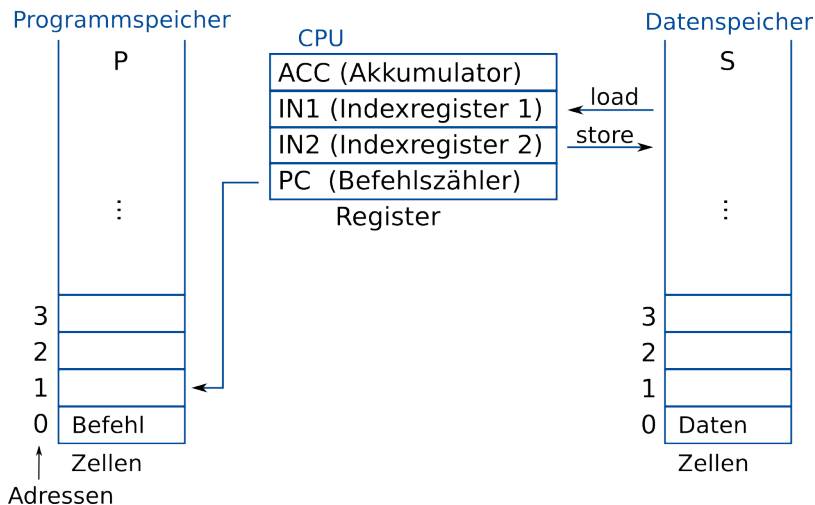
Institut für Informatik

WS 2015/16

- Ursprünglich eingeführt in [*Keller, Paul*] unter dem Namen ReSa.
- Hier wird ReTI zunächst abstrakt eingeführt.
 - Alle Speicher bestehen aus unendlich vielen Speicherzellen, die beliebig große ganze Zahlen aufnehmen können.
- Später wird die tatsächliche Implementierung von ReTI unter realistischen Annahmen thematisiert.

- Zwei unendlich große Speicher
 - **Datenspeicher S** für Daten (beliebig große Zahlen).
 - $S(i)$ = Inhalt von Zelle i des Datenspeichers, $i \in \mathbb{N}$ **Adresse**.
 - **Programmspeicher P** für Maschinenbefehle.
 - Lade-/Speicher-, Rechen, Sprungbefehle - siehe später.
 - $P(i)$ = Inhalt von Zelle i des Programmspeichers.
- **Zentraleinheit CPU** (Central Processing Unit)
 - Vier für Benutzer sichtbare Register.
 - **PC** = Befehlszähler (Program Counter).
 - **ACC** = Akkumulator.
 - **$IN1$, $IN2$** = Indexregister 1 und 2.

Aufbau von ReTI



- Programme bzw. Daten stehen beim Start der Maschine in P bzw. S .
- Programm beginnt bei Zelle 0 von P .
- Inhalt von P wird nicht geändert.
- Maschine arbeitet in Schritten $t = 1, 2, \dots$
In jedem Schritt t :
 - Ausführung eines Befehls: $P(PC)$ wird als Befehl interpretiert und in Schritt t ausgeführt.
 - PC erhält neuen Wert (abhängig von Befehl).
- Bei Programmstart ist $PC = 0$.

- **Load/Store**: Laden von Werten aus dem Datenspeicher S bzw. Schreiben von Werten in S .
- **Compute**: Berechnungen (hier zunächst Addition und Subtraktion).
 - Mit Werten im Datenspeicher S .
 - Mit Absolutwerten (Immediate).
- **Indexregister**: Indirekte Speicheradressierung (siehe unten).
- **Sprungbefehle**: Bedingte und unbedingte Sprünge.

Transport von Daten zwischen *ACC* und *Datenspeicher*.

- **LOAD i :**
Lädt *Inhalt $S(i)$* von Speicherzelle i in Akkumulator *ACC* und erhöht *PC* um 1.
- **STORE i :**
Speichert den *Inhalt von ACC* in $S(i)$ und erhöht *PC* um 1.

Load/Store: Übersicht

Befehl	Wirkung
LOAD i	$ACC := S(i) \quad PC := PC + 1$
STORE i	$S(i) := ACC \quad PC := PC + 1$

Beispielprogramm

Ein Programm, das Inhalte von Speicherzelle $S(0)$ ($= x$) und $S(1)$ ($= y$) vertauscht.

0	LOAD 0;	$ACC := S(0) = x$
1	STORE 2;	$S(2) := ACC = x$
2	LOAD 1;	$ACC := S(1) = y$
3	STORE 0;	$S(0) := ACC = y$
4	LOAD 2;	$ACC := S(2) = x$
5	STORE 1;	$S(1) := ACC = x$

Compute-Befehle

Verknüpfe den Inhalt von *ACC* mit *S(i)* (oder mit einer Konstante) und speichere das Ergebnis in *ACC* ab.

- *ADD, SUB* = Compute *memory*-Befehle
- *ADDI, SUBI* = Compute *immediate*-Befehle
- Beides zusammen ergibt die **Compute-Befehle**.

Bei Compute memory: Interpretiere Parameter *i* direkt als Speicheradresse.

Befehl	Wirkung
<i>ADD i</i>	$ACC := ACC + S(i) \quad PC := PC + 1$
<i>SUB i</i>	$ACC := ACC - S(i) \quad PC := PC + 1$

Immediate-Befehle

Interpretiere Parameter i direkt als Konstante.

Befehl	Wirkung
LOADI i	$ACC := i$ $PC := PC + 1$
ADDI i	$ACC := ACC + i$ $PC := PC + 1$
SUBI i	$ACC := ACC - i$ $PC := PC + 1$

- Anmerkung: **ADDI** und **SUBI** sind Compute Befehle.
LOADI ist den Load-/Store-Befehlen zuzuordnen.

Indexregister-Befehle

Befehl	Wirkung
LOADINj <i>i</i>	$ACC := S(INj + i)$ $PC := PC + 1$ ($j \in \{1, 2\}$)
STOREINj <i>i</i>	$S(INj + i) := ACC$ $PC := PC + 1$ ($j \in \{1, 2\}$)
MOVE <i>S D</i>	$D := S$ $PC := PC + 1$ ($D \in \{ACC, IN1, IN2\}$, $S \in \{ACC, IN1, IN2, PC\}$)
MOVE <i>S PC</i>	$PC := S$ ($S \in \{ACC, IN1, IN2\}$)

Beispielprogramm für Indexregister-Befehle

$S(0) = x, S(1) = y$

Kopiere y in Zelle $S(x)$:

0	LOAD 0;	$ACC := S(0) = x$
1	MOVE ACC $IN1$;	$IN1 := ACC = x$
2	LOAD 1;	$ACC := S(1) = y$
3	STORE $IN1$ 0;	$S(x) = S(IN1 + 0) := ACC = y$

Sprung-Befehle

Manipulation des Befehlszählers.

- **JUMP** für *unbedingte* Sprünge,
- **JUMP_c** mit $c \in \{<, =, >, \leq, \neq, \geq\}$ für *bedingte* Sprünge.
- Mit bedingten Sprüngen kann man *Programmschleifen* und *bedingte Anweisungen* realisieren!

Befehl	Wirkung
JUMP i	$PC := PC + i \quad (i \in \mathbb{Z})$
JUMP _c i	$PC := \begin{cases} PC + i, & \text{falls } ACC \text{ } c \text{ } 0 \\ PC + 1, & \text{sonst} \end{cases}$ $(i \in \mathbb{Z}, c \in \{<, =, >, \leq, \neq, \geq\})$

Beispielprogramm

$S(0) = x; S(1) = y, y \geq 0$

0	LOADI 0;	$ACC := 0$
1	STORE 2;	$S(2) := 0$
2	LOAD 1;	$ACC := S(1)$
3	SUBI 1;	$ACC := ACC - 1$
4	STORE 1;	$S(1) := ACC$
5	JUMP < 5;	$PC := PC + 5$, falls $ACC < 0$
6	LOAD 2;	$ACC := S(2)$
7	ADD 0;	$ACC := ACC + S(0)$
8	STORE 2;	$S(2) := ACC$
9	JUMP -7;	$PC := PC - 7$

- Mathematik erlaubt es uns, reale Zusammenhänge formal zu fassen und allgemeingültige Folgerungen aus ihnen herzuleiten.
- Rechner ReTI wird uns im weiteren Verlauf der Vorlesung als Illustrator und Anwendungsbeispiel für die vorgestellten Konzepte dienen.