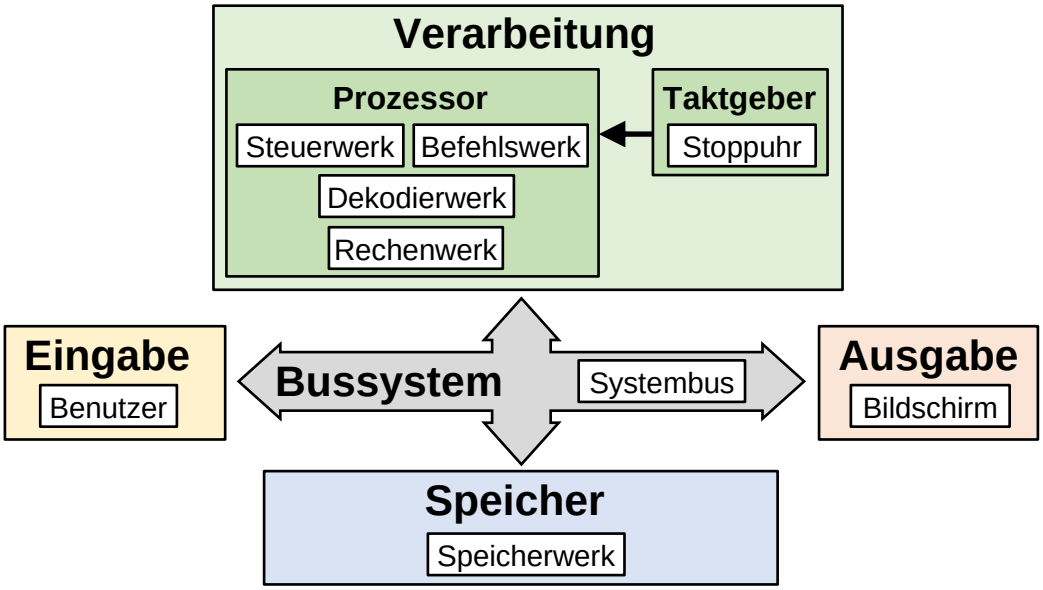
Von-Neumann-Architektur-Rollenspiel[1]

Beschreibung des Rollenspiels & Hinweise für LP

Überblick



Rollen

Vorgesehen sind 5-9 Rollen pro Rollenspiel-Gruppe, d.h. 2 bis 3 Rollenspiel-Gruppen pro Klasse. Die Rollen sind im Einzelnen:

- Steuerwerk
- Befehlswerk
- Speicherwerk
- Dekodierwerk
- Rechenwerk
- Systembus (auch passiv umsetzbar – es liegt ein Notizzettel «Systembus» in der Tischmitte, auf den alle zugreifen können)
- Benutzer (nur bei Programm 2/3; falls zu wenig Personen: Bildschirm spielt auch den Benutzer)
- Stoppuhr (auch nachträglich umsetzbar – es wird dann nach dem Rollenspiel abgeschätzt, wie lange eine Runde gebraucht hat)
- Bildschirm (auch passiv umsetzbar – es liegt dann einfach ein Bildschirm-Notizzettel auf dem Tisch)

Tipp: Für den Systembus braucht es wohl keinen Sitzplatz, die Person muss sich zwischen den Komponenten hin- und herbewegen.

[1] Adaption einer Überarbeitung von Tom Jampen der "CPU-Simulation als Rollenspiel" von Horst Gierhardt (URL der Originalversion von Horst Gierhardt: http://www.oberstufeninformatik.de/dc/). Die Überarbeitung von Tom Jampen unterliegt der CC-BY-SA 4.0 Lizenz, somit auch diese Adaption.

Vorbereitung / Materialien

- Die Rollenanleitungen, Speicher-Karten und Befehlstreifen sind im Dokument *Neumann-Rollenspiel-Druckvorlage* zum Drucken und Ausschneiden vorbereitet
- Stifte und Notizpapier für mehrere der Rollen

Programme

Für das Rollenspiel stehen vier Programme zur Verfügung:

- Programm 1:
  Es werden zwei Zahlen addiert und auf dem Bildschirm angezeigt
- Programm 2:
  Es werden zwei vom Benutzer eingegebene Zahlen addiert und auf dem Bildschirm angezeigt
- Programm 3:
  Der Benutzer muss solange eine Zahl eingeben (erraten), bis sie stimmt
- Programm 4:
  Es werden die Zahlen 02, 04, 06, 08, 10 nacheinander auf dem Bildschirm angezeigt

Vorschlag Rollenspiel-Ablauf

- Einführung in Computeraufbau, EVAS-Prinzip und von Neumann-Architektur.
- Phase 1: Beispiel 1 (ohne Kenntnis, was das Programm tut)
  - Die anwesenden Schülerinnen und Schüler werden geeignet in Gruppen eingeteilt und die Rollen werden verteilt. Alle lesen nur die eigenen Anweisungen.
  - Die Speicheradressen-Karten und die Befehlstreifen der «Basis-Sprache» bekommt das Speicherwerk resp. Dekodierwerk.
- Phase 2: Erkenntnis diskutieren
  - Was ist alles passiert? Was hat das Programm gemacht?
  - Was man bis hier verstehen sollte
    - Fetch-Decode-Execute Prinzip
    - Speicher unterteilt in Programmcode und Daten
- Phase 3: Beispiel 2
  - Speicherwerk und Dekodierwerk erhalten die Speicheradressen-Karten resp. Befehlstreifen der "Erweiterten Sprache".
- Phase 4: Abschliessende Diskussion (zentrale Punkte siehe unten)

Abschliessende Diskussion

Hier kann z.B. der Vergleich zu einer richtigen CPU gemacht werden. Die Stoppuhr hat für jede Runde notiert, wie lange diese gedauert hat (oder die Zeit wird nachträglich abgeschätzt). Eine richtige CPU arbeitet mit ca. 3 Milliarden Taktzyklen pro Sekunde.

Zentrale Punkte:

- *Ein Computer kann nur wenige, simple Dinge, dafür in horrendem Tempo.*
- Ein Programm besteht auch nur aus Zahlen (braucht Dekoder).
- Fetch-Decode-Execute-Prinzip
- Ein Programm wird aus unzähligen einfachen Anweisungen aufgebaut.
- Programmiersprachen mit ihren Konstrukten vereinfachen uns die Arbeit und abstrahieren die benutzte Hardware.

Musterprogrammablauf

Programm 1

Speicheradresse	Speicherinhalt	Bedeutung
1	1A	Hole Inhalt von Speicheradresse A (23) und übergib diesen dem Rechenwerk.
2	3B	Hole Inhalt von Speicheradresse B (42) und übergib diesen dem Rechenwerk zum Addieren (65).
3	2C	Speichere das Resultat (65) in Speicheradresse C
4	4C	Hole den Inhalt der Speicheradresse C (65) und übergib diesen dem Bildschirm
5	50	Beende das Programm
6	00	
7	00	
8	00	
9	00	
A	23	[Datenspeicher: Zahl 1]
B	42	[Datenspeicher: Zahl 2]
C	00	[Datenspeicher: Resultat]

Programm 2

Speicheradresse	Speicherinhalt	Bedeutung
1	6A	Speichere Eingabe vom Benutzer in Speicherstelle A.
2	6B	Speichere Eingabe vom Benutzer in Speicherstelle B.
3	1A	Hole Inhalt von Speicheradresse A (Eingabe 1) und übergib diesen dem Rechenwerk.
4	3B	Hole Inhalt von Speicheradresse B (Eingabe 2) und übergib diesen dem Rechenwerk zum Addieren.
5	2C	Speichere das Resultat in Speicheradresse C
6	4C	Hole den Inhalt der Speicheradresse C und übergib diesen dem Bildschirm
7	50	Beende das Programm.
8	00	
9	00	
A	23	[Datenspeicher: Inhalt egal, wird überschrieben]
B	42	[Datenspeicher: Inhalt egal, wird überschrieben]
C	65	[Datenspeicher: Inhalt egal, wird nicht verwendet]

Programm 3

Speicheradresse	Speicherinhalt	Bedeutung
1	6A	Speichere Eingabe vom Benutzer in Speicherstelle A.
2	1A	Hole Inhalt von Speicheradresse A (Eingabe) und übergib diesen dem Rechenwerk.
3	7C	Hole Inhalt der Speicheradresse C (32) und übergib diese dem Rechenwerk zum Vergleichen
4	2B	Speichere das Resultat in Speicheradresse B
5	4B	Hole den Inhalt der Speicheradresse B (Zahl richtig=01 oder falsch=00) und übergib ihn dem Bildschirm
6	80	Hole Zahl beim Rechenwerk → Falls 00 (geratene Zahl falsch), fängt das Programm wieder bei Speicheradresse 1 an. → Falls 01 (geratene Zahl richtig), dann fährt das Programm weiter.
7	50	Beende das Programm.
A	16	[Datenspeicher: Inhalt egal, wird überschrieben]
B	73	[Datenspeicher: Inhalt egal, wird überschrieben]
C	32	[Datenspeicher: zu erratende Zahl]

Programm 4

Speicheradresse	Speicherinhalt	Bedeutung
1	1A	Hole Inhalt von Speicheradresse A (00) und übergib diesen dem Rechenwerk.
2	3C	Hole Inhalt von Speicheradresse C (02) und übergib diesen dem Rechenwerk zum Addieren (02).
3	2A	Speichere das Resultat (02) in Speicheradresse A
4	4A	Hole den Inhalt der Speicheradresse A (02) und übergib ihn dem Bildschirm
5	7B	Hole Inhalt der Speicheradresse B (10) und übergib diese dem Rechenwerk zum Vergleichen
6	80	Hole Zahl beim Rechenwerk (00), Befehlswerk setzte Speicheradresse auf 2. Ziffer des aktuellen Befehls (0) → d.h. das Programm fängt wieder bei Speicheradresse 1 an, jetzt aber mit dem Wert 02 an Speicheradresse A. Somit wird diese Zahl in jedem Durchgang erhöht bis auch dort die Zahl 10 steht und der Vergleich der beiden Zahlen in 01 resultiert; dann fährt das Programm weiter.
7	50	Beende das Programm
8	00	
9	00	
A	00	[Datenspeicher: Startzahl]
B	10	[Datenspeicher: Endzahl]
C	02	[Datenspeicher: Inkrement]