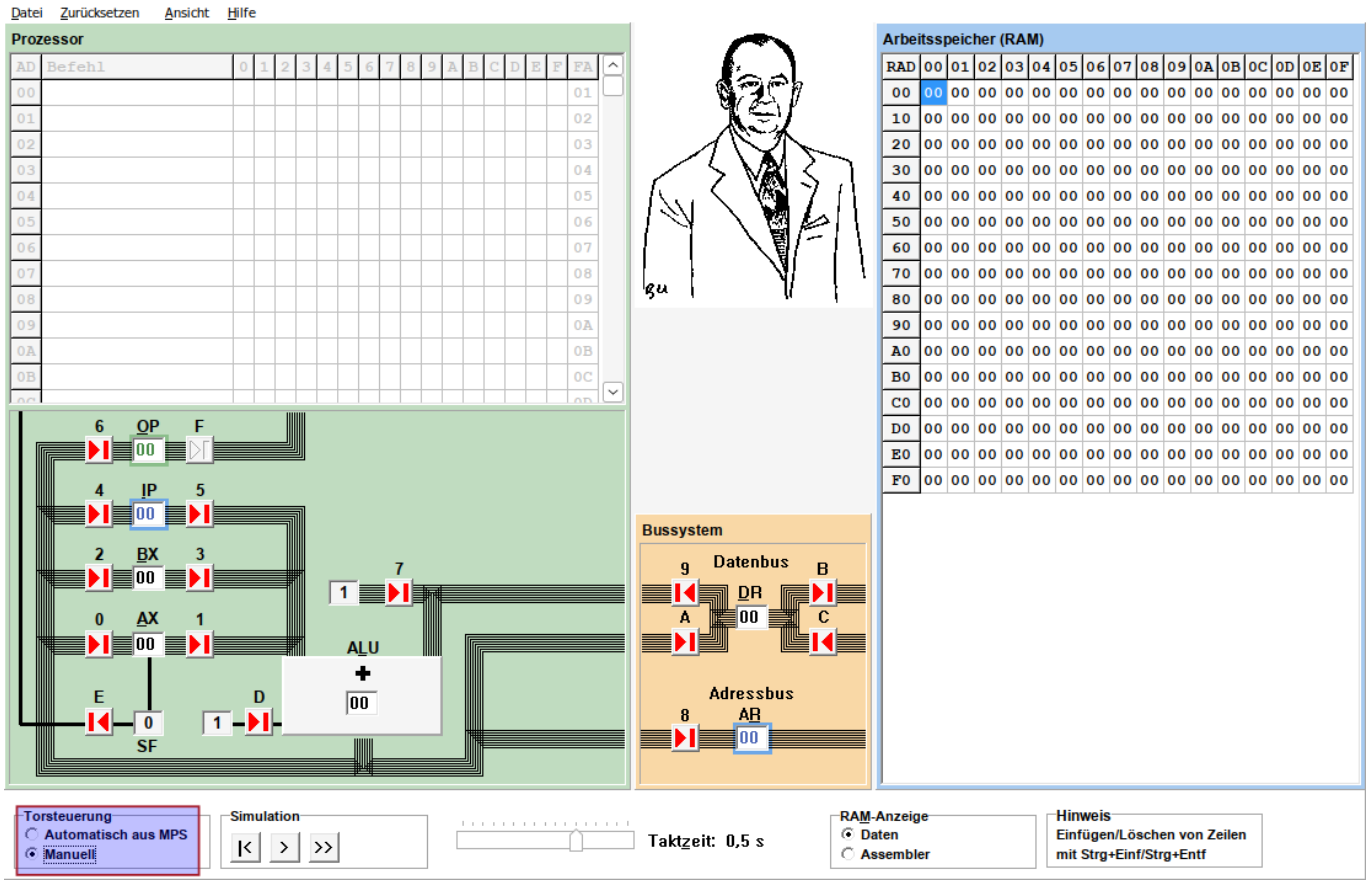
Einführung in MikrosimD

Das Bild zeigt das Hauptfenster von MikrosimD:

- Unten links im befindet sich die ALU (Arithmetisch-Logische Einheit) und eine Reihe von Registern (1-Byte-Speicher). Alle Bitfolgen sind hexadezimal angegeben, die zwei Hexadezimalzahlen, die in jedes der Register geschrieben werden können entsprechen also einem Byte.
- Rechts ist der Arbeitsspeicher (das RAM) mit 256 Byte Speicherplatz (von 00h bis FFh)
- Zwischen den Registern, der ALU und dem Arbeitsspeicher stehen "Tore" (von 0h bis Fh), die man durch Klicken öffnen kann.

Wenn ein Tor geöffnet ist, kann im nächsten Bus-Takt der im Register gespeicherte Wert über den Bus transportiert werden. Ein Bus-Takt wird durch den Button > unten links ausgelöst.



Erste Schritte

- Stelle zunächst die Torsteuerung unten links auf `Manuell`.
- Vollziehe die folgenden Beispiele nach

Einführungsbeispiele 1

a)

Trage im Register AX den Wert 17h ein, dieser soll in die Register BX und DR kopiert werden.

- Öffne die Tore 1, 2 und A
- Löse nacheinander mehrere Taktschritte aus, indem du auf die Schaltfläche > klickst und beobachte, was geschieht.
- Setze anschließend die Simulation zurück (`Zurücksetzen` → `Register löschen` oder F9)

b)

In den Registern AX, BX und DR steht zunächst die Zahl 00. in beiden Registern soll die Zahl 01 eingetragen werden.

- Öffne die Tore 0, 2, 7 und A
- Löse nacheinander mehrere Taktschritte aus, indem du auf die Schaltfläche > klickst und beobachte, was geschieht.
- Setze die Simulation zurück

c)

Im Register AX steht der Wert 08h, im Register DR steht der Wert 03h. In BX soll die Summe der in den Registern von AX und DR gespeicherten Werte stehen (also 0Bh).

Findest du selbst heraus, welche Tore geöffnet werden müssen?

Lösung

- Öffne die Tore 1, 2 und 9
- Löse nacheinander mehrere Taktschritte aus, indem du auf die Schaltfläche > klickst und beobachte, was geschieht.

d)

Im Register AX steht der Wert 08h, im Register DR steht der Wert 03h. In AX und DR soll die Differenz der beiden Registerinhalte eingetragen werden (also AX-DR, 05h).

Findest du selbst heraus, welche Tore geöffnet werden müssen? Überlege, welche Funktion das Tor D haben könnte.

Lösung

- Öffne die Tore 0, 1, 9, A und D

- Löse nacheinander mehrere Taktschritte aus, indem du auf die Schaltfläche > klickst und beobachte, was geschieht.

Einführungsbeispiele 2

a)

Wie setzt man alle Register auf 00?

Lösung

Tore 0, 2, 4, 6, 8 und A

b)

Kann man die Tore 1 und 3 gleichzeitig öffnen? Warum geht das nicht?

Lösung

Die ALU hat zwei Eingänge und einen Ausgang. Im Ersten Taktschritt werden Werte in die ALU geschrieben und verarbeitet, im zweiten Taktschritt das Ergebnis am Ausgang ausgegeben. Man kann über einen Eingang nicht im selben Taktschritt mehr als einen Wert verarbeiten.

c)

Eine besondere Aufgabe hat das Register AR (Adress-Register). Es legt fest, auf welche Adresse des Arbeitsspeichers zugegriffen werden soll. Die Tore B und C legen fest, ob der Wert des Datenregisters DR in den Arbeitsspeicher geschrieben wir, oder ob der Wert, welcher an der in AR gegebenen Adresse gespeichert ist nach DR ausgegeben wird.

Teste die Funktionalität:

- Schreibe den Wert AF von DR aus an die Adresse EA im RAM.
- Trage den Wert FF an der Adresse 1D im RAM ein. Lies den Wert ins Register DR aus.

Bisherige Erkenntnisse

RAM Zugriff

- Das Adressregister AR enthält die Adresse, die beim Zugriff auf das RAM gelesen oder geschrieben werden soll.
- Wenn also der Inhalt von AR z.B. 09 ist, dann wird
  - mit dem Tor C der Inhalt der RAM-Zelle 09 ins Datenregister DR gelesen
  - mit dem Tor B der Inhalt des Datenregisters DR in die RAM-Zelle 09 geschrieben
- Wenn auf das RAM zugegriffen wird, sind alle anderen Tore blockiert. Der Grund hierfür ist, dass in der Realität der Speicherzugriff mehr als 200 mal so lange dauern kann wie ein Registerzugriff.

Taktzyklen

- Ein ALU-Takt besteht aus zwei Bus-Takten
  - Im ersten Bustakt werden die Werte aus den Registern über die geöffneten Tore in die ALU geschickt.
  - Im zweiten Bustakt werden die Ergebnisse aus der ALU in die Register geschrieben.
- Der mittlere Button (>) führt je einen Bustakt aus, der rechte Button (») führt den gesamten ALU-Takt aus.
- Speicherzugriffe benötigen nur nur einen Takt.

Ein Wort zu Latenzen

Die **Latenzzeit** ist ein Maß für die Wartezeit. Damit kann z.B. die Zeit gemeint sein, die benötigt wird, um einen bestimmten Vorgang abzuschließen, z. B. die Anforderung des Inhalts einer Speicherzelle oder eines Registers, aber auch die Zeit, die eine Datenbankabfrage benötigt um abgeschlossen zu werden.

Die Latenzzeit kann z.B. auch die Zeit ausdrücken, die eine Website benötigt, um vollständig geladen zu werden, vom Anklicken des Links bis zur Darstellung auf dem Bildschirm.

Heutige CPUs arbeiten sehr schnell, mit Taktfrequenzen von einigen GHz. Der Zugriff auf Inhalte des Hauptspeichers dauert jedoch um ein Vielfaches länger - wenn der Prozessor also Daten aus dem Hauptspeicher verarbeiten möchte, muss er warten bis dieser Zugriff abgeschlossen ist und die Daten zur Verfügung stehen - es entsteht eine Latenzzeit.

Wenn man Aussagen trifft wie: "*Der Zugriff auf den Hauptspeicher dauert ein vielfaches länger, als der Zugriff auf die Register oder den Cache des Prozessors*" kann man sich unter dem Begriff "ein vielfaches länger" jedoch meist wenig vorstellen.

Der Autor Brendan Gregg hat die Latenzen eines Computersystems in seinem Buch "Systems Performance - Enterprise and the Cloud" sehr schön in einer Tabelle veranschaulicht. Er geht von einem CPU Takt aus, der etwa 0,3ns dauert ($0,3\cdot10^{-9}s$), wenn man von einer Taktfrequenz von etwa 3GHz bei modernen Prozessoren ausgeht.

Diese Zeitdauer skaliert der Autor in einer Tabelle auf 1 Sekunde hoch, und rechnet auf dieser Basis aus, wie lange andere Vorgänge in Rechnersystemen bei dieser Skalierung dauern würden. Dadurch erhält man ein besseres Gefühl, des Ausdrucks "ein Vielfaches länger. Im Falle des Hauptspeicherzugriffs kann man der Tabelle entnehmen, dass dieser etwa 360 mal so lange dauert wie ein CPU Takt und 120 mal so lange wie der Zugriff auf den Level 1 Cache im Prozessor.

Vorgang	**Latenz**	**Skaliert**
1 CPU Takt	0,3ns	1s
Level 1 Cache Zugriff	0,9ns	3s
Level 2 Cache Zugriff	2,8ns	9s
Level 3 Cache Zugriff	12.9ns	43s
Hauptspeicherzugriff (DRAM, von der CPU aus)	120ns	6min
SSD Zugriff (Solid State Festplatte)	50-150µs	2-6 Tage