VON–NEUMANN–RECHNER auf dem Bildschirm
CPU–Simulation mit dem Didaktischen Computer DC

1 'Hardware'–Beschreibung

1.1 Zentraleinheit

Das Programm DC simuliert auf dem Bildschirm eine einfache Zentraleinheit (CPU = **c**entral **p**rocessing **u**nit) eines Computers nach dem *Von–Neumann–Prinzip* mit

- Rechenwerk (ALU = **a**rithmetic **l**ogical **u**nit),
- Steuerwerk bzw. Kontrolleinheit,
- Speicher mit Schreib–Lese–Zugriff (RAM = **r**andom **a**ccess **m**emory), und
- Ein–Ausgabe–Einheit (I/O = Input/Output).

Die simulierte Zentraleinheit hat einen Arbeitsspeicher von 128 Wörtern zu je 13 Bit, in dem das Programm und die Daten untergebracht sind. Durch die Beschränkung der Wortlänge auf 13 Bit sind als Daten nur ganze Zahlen x mit $-4096 \leq x \leq 4095$ zulässig. Eine Speicherstelle kann eine ganze Zahl oder einen Befehl einschließlich einer Speicheradresse enthalten. Mit 6 der 13 Bit lässt sich ein Befehl darstellen, d.h. es sind 64 verschiedene Befehle möglich. Mit den restlichen 7 Bit ist eine Speicheradresse (0 – 127) darstellbar. In diesem Punkt weicht der Simulationsrechner von realen Rechnerarchitekturen ab, die meist den Befehl und den Operand an aufeinanderfolgenden Speicherstellen ablegen.

1.2 Register

Im Mikroprozessor stehen verschiedene Register zur internen Zwischenspeicherung verschiedener Daten, Ablaufkontrolle und zum Rechnen zur Verfügung:

1.2.1 Befehlsregister (IR = instruction register)

Vor der Ausführung eines Befehls muss das entsprechende Befehlswort in das Befehlsregister geladen werden. Das Befehlswort (13 Bit) wird vom Steuerwerk (CONTROL) in den eigentlichen Befehl und den Operanden zerlegt. Am Bildschirm ist die Aufteilung in Befehlsteil und Adresse zu erkennen. Bei realen Mikroprozessoren wird an dieser Stelle der Operand mit einem weiteren Speicherzugriff geladen. Das Steuerwerk hat intern für jeden Befehl ein kleines Programm gespeichert (Mikrocode). Ein solches Programm wird nach der Decodierung des geladenen, binär codierten Befehls gestartet. Diverse Register werden von hier bedient bzw. aktiviert. Die Synchronisation aller Vorgänge durch einen Taktgenerator wird bei diesem Rechnermodell nicht thematisiert.

1.2.2 Befehlszählregister (PC = program counter)

In diesem Register steht immer die Speicheradresse des nächsten auszuführenden Befehls. Vor jeder Befehlsausführung wird dieses Register um 1 inkrementiert. Bei Sprungbefehlen wird dieses Register mit der Zieladresse, die vom Befehlsregister geliefert wird, geladen. Dieses Register kann im Direktmodus manipuliert werden.

1.2.3 Adressregister (AR)

Bei jedem Schreib- und Lesevorgang im Speicher muss in diesem Register die Adresse der anzusprechenden Speicherstelle stehen. Dadurch wird die entsprechende Speicherstelle zugänglich. Das Adressregister kann nicht direkt manipuliert werden, sondern wird jeweils vom Steuerwerk mit der benötigten Adresse bedient.

1.2.4 Datenregister (DR)

Das Datenregister nimmt den zu schreibenden oder den gelesenen Wert auf. Alle Werte vom und zum Speicher gehen über dieses Register. Es kann wie das Adressregister nicht direkt manipuliert werden.

1.2.5 Akkumulator (AC)

Der Akkumulator ist das zentrale Register im Mikroprozessor. In Verbindung mit der ALU (arithmetic logical unit) ist er für das Rechnen zuständig. Die ALU kann z.B. einen Wert aus einer Speicherstelle zu einem Wert im Akkumulator addieren. Das Ergebnis steht dann im Akkumulator. Im Akkumulator können also zwei Werte durch eine Rechenoperation 'zusammengeführt' werden. Die vorliegende ALU kennt allerdings nur Addition, Subtraktion und Negation als Operationen. Das Inkrementieren und Dekrementieren um 1 kann direkt ohne Zugriff auf den Speicher geschehen. Reale Computer besitzen meist mehrere Register.

1.3 Datenübertragung

Zur Übertragung von Daten sind Leitungen zwischen den einzelnen Komponenten des Mikroprozessors bzw. zwischen Mikroprozessor, Speicher und I/O–Einheit notwendig:

1.3.1 Adressbus

Über den Adressbus wird über das Adressregister im Speicher eine Speicherstelle 'geöffnet'. Im vorliegenden Modell wären also nur 7 Leitungen nötig, da mit 7 Bit 128 Speicheradressen angesprochen werden können. Der Adressbus hat hier also eine Busbreite von 7 Bit. Gängige Werte für Adressbusbreiten in realen Rechnern sind 8 Bit, 16 Bit (beide veraltet), 20 Bit, 24 Bit, 32 Bit und 64 Bit.

1.3.2 Datenbus

Über den Datenbus wird über das Datenregister ein Wert in eine 'geöffnete' Speicherstelle geschrieben oder daraus gelesen. Im vorliegenden Modell wären 13 Leitungen nötig, da die Speicherworte eine Breite von 13 Bit besitzen. Der Datenbus hat hier also eine Breite von 13 Bit. Gängige Werte für Datenbusbreiten in realen Rechnern sind 8 Bit, 16 Bit, 32 Bit und 64 Bit. Ein PC mit Intel-8086-Mikroprozessor besitzt eine Adressbusbreite von 20 Bit und eine Datenbusbreite von 16 Bit. Intern verarbeitet er 16 Bit. Entsprechend wird er als 16–Bit–Computer klassifiziert. Rechner mit dem Intel–80386/486 haben eine Daten- und Adressbusbreite von je 32 Bit und können intern (z.B. im Akkumulator) 32 Bit mit einem Befehl verarbeiten.
Über den Datenbus können auch Werte an die Ausgabeeinheit (Output) übergeben oder von der Eingabeeinheit (Input) übernommen werden.

1.3.3 Steuerleitungen

Zwischen Steuerwerk und Speicher liegt zusätzlich die Schreib/Leseleitung. Über sie wird die Information weitergegeben, ob bei einem Speicherzugriff geschrieben oder gelesen werden soll. Im vorliegenden Modell ist diese Leitung nicht dargestellt.

2 Befehlssatz

2.1 Grundbefehle

Mnemo	**Bedeutung**
LDA	LOAD INTO ACCUMULATOR — Lade den Wert der angegebenen Speicherstelle in den Akkumulator.
STA	STORE ACCUMULATOR TO MEMORY — Speichere den Akkuinhalt an der angegebenen Speicherstelle ab.
ADD	ADD TO ACCUMULATOR — Addiere den Wert der angegebenen Speicherstelle zum Akkuinhalt.
SUB	SUBTRACT FROM ACCUMULATOR — Subtrahiere den Wert der angegebenen Speicherstelle vom Akkuinhalt.
NEG	NEGATE ACCUMULATOR — Negiere Akkuinhalt.
INC	INCREMENT ACCUMULATOR — Erhöhe Akkuinhalt um 1.
DEC	DECREMENT ACCUMULATOR — Erniedrige Akkuinhalt um 1.
OUT	OUTPUT MEMORY — Gib den Wert der angegebenen Speicherstelle an die Output-Einheit.
INM	INPUT TO MEMORY — Speichere die von der Input–Einheit gelesene Zahl an der angegebenen Adresse ab. Das Programm hält bei diesem Befehl an und wartet auf die Eingabe einer Zahl.
END	ENDE — Programm beenden.
DEF	DEFINE word — Mit 34 DEF 3 erhält die Speicherstelle mit der Adresse 34 den Wert 3 zugewiesen. Dies ist keine vom Mikroprozessor ausführbare Anweisung, sondern dient nur der Wertbelegung von Speicherstellen beim Einlesen eines DC–Programmes oder im Direktmodus.

2.2 Sprungbefehle

Mnemo	**Bedeutung**
JMP	JUMP — Unbedingter Sprung. Springe zur angegeb. Speicherstelle und fahre mit dem dort stehenden Befehl fort.
JMS	JUMP IF MINUS — Springe zur angegeb. Speicherstelle und fahre mit dem dort stehenden Befehl fort, wenn der Akkuinhalt negativ ist. Wenn nicht, dann fahre mit dem nächsten Befehl fort.
JPL	JUMP IF PLUS — Sprung, wenn Akkuinhalt > 0.
JZE	JUMP IF ZERO — Sprung, wenn Akkuinhalt $= 0$.
JNM	JUMP IF NOT MINUS — Sprung, wenn Akkuinhalt ≥ 0.
JNP	JUMP IF NOT PLUS — Sprung, wenn Akkuinhalt ≤ 0.
JNZ	JUMP IF NOT ZERO — Sprung, wenn Akkuinhalt $\neq 0$.
JSR	JUMP TO SUBROUTINE — Springe zum Unterprogramm an der angegebenen Adresse und fahre nach dem Rücksprung mit dem nächsten Befehl fort.

Mnemo	**Bedeutung**
RTN	RETURN FROM SUBROUTINE — Kehre vom Unterprogramm zurück zum Befehl nach der Aufrufstelle.

3 Programmsteuerung

Eingabe	**Merkform**	**Bedeutung**
H	Help	Hilfetext anzeigen.
C	Clear	Alles löschen (DC–Reset). Der Speicherinhalt wird gelöscht, und alle Register werden auf ihren Startwert gesetzt.
W	Wait	Nach jeder Phase der Befehlsausführung auf Taste warten.
D	Delay	Nur eine kleine Pause nach jeder Phase.
N	No Wait	Keine Pausen, nicht auf Taste warten. Einzelschrittmodus ausschalten
R	Run	Programmausführung starten.
G x	Go x	Programm ab Adresse x ausführen.
CR	RETURN	Einzelausführung (CR = carriage return). Ein Befehl wird komplett ausgeführt.
ESC	ESCAPE	Programm anhalten.
V x	View x	Speicherseite mit Adresse x anzeigen.
ED	Edit	Editor (intern) aufrufen.
ASS	Assemble	Mini–Assembler aufrufen.
Q	Quit	DC verlassen.
L name	Load name	DC-Programm aus Datei ‘NAME.DC’ laden. Kommentare werden abgeschnitten.
Adr. Code Argument		Befehl direkt in Speicher laden.
Adr. DEF Konstante		Konstante direkt in den Speicher laden.
PC Adresse		PC–Register mit Adresse laden.