Technische Universität Dresden
Fakultät Elektrotechnik und Informationstechnik
Institut für Nachrichtentechnik

Lehrstuhl Telekommunikation
Prof. Dr.-Ing. R. Lehnert
Oktober 2005

Praktikum zur Lehrveranstaltung

Computertechnik I

Versuch

Von-Neumann-Simulator

Inhaltsverzeichnis:

1. Zielstellung
2. Ablauf und Bewertung
3. Nutzungshinweise für den Simulator COMPI16
3.1. Aufruf des Simulators
3.2. Eingabe des Dateinamens
3.3. Bildschirmaufbau
3.4. Bedienung des COMPI16-Modellrechners
4. Befehle und Befehlsausführung
4.1. Adressierungsarten
4.2. Befehlsformat
4.3. Transportbefehle
4.4. Arithmetische und logische Befehle
4.5. Sprungbefehle
4.6. Ein-/ Ausgabebefehle
4.7. Steuerbefehle
5. Assembler
5.1. Format von Assembleranweisungen
5.2. Maschinenbefehle
5.3. Pseudobefehle
5.4. Programmbeispiel
5.5. Bedienung
6. Programmverbinder
7. Beispielaufgabe
8. Kolloquiumsschwerpunkte
9. Literaturhinweise
10. Arbeits- und Brandschutzhinweise

1. Zielstellung:

Das Praktikum soll dazu dienen, das Verständnis für die prinzipielle Arbeitsweise von Digitalrechnern zu vertiefen.
Hauptinhalt ist die Demonstration interner Arbeitsabläufe in einem Computer. Dazu gehört das Zusammenwirken von Prozessor, Speicher und Peripherie und vor allem der Mechanismus der zyklischen Befehlsbearbeitung (zentrale Steuerschleife). Auf dem Monitor des Praktikumsrechners wird dazu eine Computerarchitektur nachgebildet. Einzelne grafische Gestaltungsmittel erlauben es, die notwendigen Abläufe bei der Abarbeitung der Befehle zu verfolgen. Grundlage dafür ist ein von der Praktikumsgruppe zu erstellendes Programm in Assemblersprache, dessen schrittweise Verarbeitung bis hin zum Maschinenprogramm ebenfalls veranschaulicht wird.
In Vorbereitung auf die Praktikumsdurchführung ist der Algorithmus zur konkreten Praktikumsaufgabe zu formulieren und als ASSEMBLER-Quelltext zum Praktikumstermin mitzubringen.

2. Ablauf und Bewertung:

Der Versuch „Von-Neumann-Simulator“ wird als erster der drei Praktikumsversuche zur Lehrveranstaltung „Computertechnik I“ absolviert. Mit ihm sollen die Voraussetzungen für die Versuche „Programmierung in ASSEMBLER“ und „Programmierung in C“ verbessert werden. In Vorbereitung auf das Praktikum sind sowohl die zuvor behandelten Stoffgebiete aus den Vorlesungen „Informatik“ und „Computertechnik I“ zu wiederholen, als auch die Arbeiten zum Programmentwurf für die konkrete Praktikumsaufgabe zu erledigen.
Während des eigentlichen Praktikums am Computer soll die Gruppe sich mit den Möglichkeiten des Simulationstools vertraut machen, die Abläufe bei der Arbeit eines Digitalrechners an Hand des Modellcomputers COMPI16 studieren, die Implementierung des vorbereiteten Quelltextes vornehmen und das Programm zur fehlerfreien Arbeit bringen.
Während des Praktikums findet ein Kolloquium statt, in dem der Betreuer die Vorbereitung des einzelnen Teilnehmers überprüft. Nach dem Praktikum ist von der Gruppe ein gemeinsames Protokoll anzufertigen. Es soll den Programmentwurf und den Prozeß der Implementierung widerspiegeln und muß somit die genaue Aufgabendefinition, eine geeignete Algorithmusnotation, den ausführlich kommentierten Quelltext sowie eine Diskussion der aufgetretenen Fehler und Ergebnisse enthalten. Außerdem ist ein besonders interessanter Programmausschnitt von 10 ... 15 Zeilen Länge (z.B. Ausschnitt mit Sprungbefehlen oder Befehlen, die indirekte Adressierung nutzen) im Schrittbetrieb abzuarbeiten und in folgender Weise zu protokollieren:

$<BZ>_{alt}$	$<<BZ>>_{alt}$	$<BZ>_{neu}$	$<AC>_{bin}$	$<Flags>_{bin}$	Bemerkungen
hexadez	mnemonisch hexadezimal	hexadez	nach Abarbeitung von $<<BZ>>_{alt}$		
Bsp.:					
1F	SUB 012E	20	010011	0011	Korr. Wert2
20	JMP 15	15	010011		Eing. Schleife

Beim Protokollieren des Akkumulatorinhalts brauchen führende Nullen nicht notiert zu werden. Die Bemerkungen sollen das Problem kommentieren, nicht den vorn stehenden Befehl.

Die Leistung im Praktikum wird mit maximal 10 Punkten bewertet. Davon entfallen 6 Punkte auf das Kolloquium, 2 Punkte auf die Praktikumsdurchführung und 2 Punkte auf das Protokoll.

3. Nutzungshinweise für den Simulator COMPI16

3.1. Aufruf des Simulators

Der mit dem Programm COMPI16.EXE simulierte Computer entspricht in seinem Aufbau einer „von-Neumann“-Rechnerarchitektur. Er besitzt die typischen Komponenten eines derartigen Rechnersystems, nämlich Hauptspeicher - HS, Steuereinheit - SE und Verarbeitungseinheit - VE (zusammengefaßt als ZVE oder CPU) sowie Bussystem. Die simulierte Peripherie besteht aus Tastatur und Bildschirm. Das Simulationsprogramm COMPI16.EXE wurde vom Institut für Datenverarbeitung der TU Wien erstellt und ist unter dem Betriebssystem MS-DOS lauffähig. Es wird durch die Eingabe von COMPI16 <ENTER> gestartet. Dabei muß sich die Datei COMPI16.SCR im aktuellen Verzeichnis befinden.

3.2. Eingabe des Dateinamens

Nach dem Aufruf von COMPI16.EXE erscheint auf dem Bildschirm ein Titelbild sowie die Frage nach der abzuarbeitenden Datei. Durch die Eingabe des korrekten Dateinamens und Bestätigen mit <ENTER> wird die entsprechende Datei in den Speicher des Modellcomputers geladen. Wird nur die <ENTER>-Taste betätigt, bleibt der Speicher leer.
Folgende Punkte sind bei der Eingabe zu beachten:

Der Dateiname muß unter MS-DOS gültig sein.

Der Dateiname darf nur die Namenserweiterung „.LAD“ aufweisen, sie kann aber auch

vollständig entfallen.

Entspricht der Dateiname nicht den Regeln von MS-DOS oder ist die Datei nicht vorhanden, wird eine entsprechende Fehlermeldung ausgegeben.

3.3. Bildschirmaufbau

Nach der Eingabe des Dateinamens erscheint das untenstehende Bild des Modellcomputers auf dem Bildschirm (Bild 1). Die letzten zwei Zeilen dienen als Hilfestellung und zeigen die Belegung der zehn Funktionstasten.

Die Komponenten des Modellcomputers haben folgende Bedeutung:

Hauptspeicher (HS, MEM)
Hier ist ausschnittsweise der Hauptspeicher des Modellcomputers dargestellt. Der Speicher des COMPI16 ist wortweise organisiert. Die Wortlänge beträgt 16 Bit.

Adreßregister
Dieses Register enthält die Adresse der aktuellen Speicherzelle, auf die sich Speicherzugriffe beziehen.

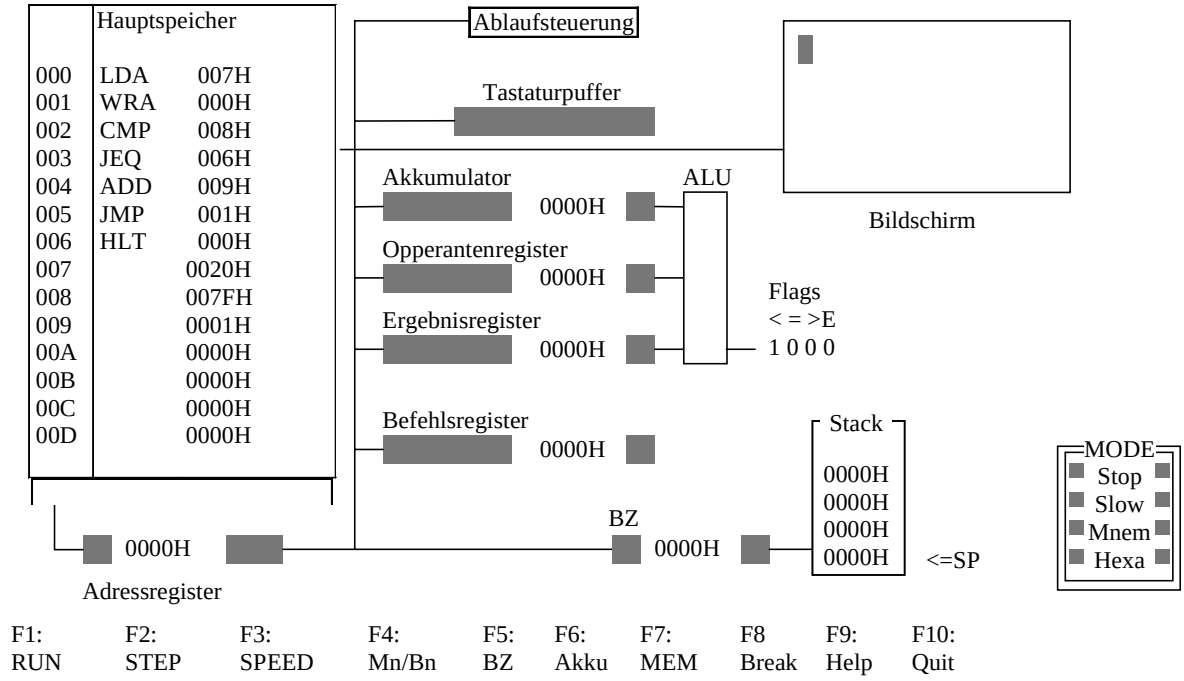

Bild 1: Komponenten des Modellcomputers

Ablaufsteuerung
Hier ist stilisiert die komplexe Ablaufsteuerung aller Prozesse im Modellcomputer zusammengefaßt dargestellt.

Tastaturpuffer
Dieses Pufferregister speichert die über Tastatur eingegebenen Zeichen (im ASCII-Code), bis sie vom Programm gelesen werden. Ist bei der Eingabe eines Zeichens der Puffer voll, geht das älteste Zeichen verloren (FIFO-Organisation, first in first out).

Akkumulator (Akku, AC)
Akkumulatorregister des Prozessors; Es kann maximal 15 Bit breite Daten aufnehmen. Das 16. Bit wird zur Unterscheidung von Befehlen und Daten verwendet.

Operandenregister
Es dient bei entsprechenden Operationen als Zwischenspeicher für den zweiten Operanden.

Ergebnisregister
Es dient bei entsprechenden Operationen als Zwischenspeicher für das Resultat, das dann in den Akkumulator übertragen wird (Ausnahme CMP-Befehl).

Befehlsregister (BR)
Es nimmt den abzuarbeitenden Befehl auf.

Befehlszähler (BZ)
Er enthält die Adresse des nächsten abzuarbeitenden Befehls.

Arithmetisch-logische Einheit, arithmetic logic unit (ALU)
Hier werden die arithmetischen und logischen Operationen ausgeführt.

Flags < = > E
Diese Komponente stellt das Flag-Register des Prozessors dar.
< = > : Wenn das entsprechende zugeordnete Bit auf „1“ gesetzt ist, so war das Ergebnis der letzten arithmetischen Operation kleiner als Null, gleich Null oder größer als Null bzw. das Ergebnis des letzten CMP-Befehls „Kleiner“, „Gleich“ oder „Größer“.
E: Extensions-Flag für Verschiebeoperationen

Stack und Stackpointer (SP)
Kellerspeicher mit entsprechendem Kellerzeiger; Das Fassungsvermögen des Stacks beträgt 4 Worte.

Bildschirm
Display des Modellcomputers;

Mode
Anzeige des aktuellen Arbeitszustandes des Modellcomputers;

3.4. Bedienung des COMPI16-Modellrechners

Die Bedienung des Modellcomputers erfolgt im wesentlichen mit den Funktionstasten F1 bis F10 des PC. Ihre Bedeutung wird am unteren Bildschirmrand angezeigt.

F1 : RUN (bzw. STOP)	F2 : STEP	F3 : Speed Slow/Fast)
F4 : Mn/Bn (Mnemonic/Binary)	F5 : BZ	F6 : Akku
F7 : MEM	F8 : Break	F9 : Help
F10: Quit		

RUN (F1)
Mit der RUN-Taste wird der Programmablauf gestartet. Befindet sich COMPI16 im Slow-Modus, werden alle Aktionen wie Registertransfers, ALU-Operationen usw. ausführlich dargestellt. Im Fast-Modus wird nur der Inhalt des Akkumulators und des Befehlszählers gezeigt.
Die Programmausführung kann wie folgt unterbrochen werden:

* Erreichen eines HLT-Befehls während der Programmabarbeitung;
* Drücken der Taste STOP (diese wird nur während des Programmablaufs sichtbar); Der laufende Befehl wird danach noch vollständig zu Ende geführt.
* Erreichen eines Haltepunktes (Breakpoint, siehe Beschreibung F8);
* Auftreten eines Laufzeitfehlers; mögliche Ursachen:
    - Daten (statt Befehl) werden in BR geladen
    - Befehl (statt Daten) wird in Akkumulator oder Operandenregister geladen
    - Befehl im Speicher wird durch Daten überschrieben
    - Bei indirekter Adressierung wird die Adresse aus einem Speicherplatz entnommen, in dem ein Befehl steht.

  Das Erkennen dieser Laufzeitfehler ist möglich, da bei diesem Modellcomputer Befehle und Daten unterschieden werden können. Dies ist jedoch nicht typisch für die „J. von Neumann“–Architektur, sondern nur ein Merkmal dieses Modellcomputers. Ein Befehl ist dadurch gekennzeichnet, daß das höchstwertige Bit (Bit 15) auf logisch „1“ gesetzt ist. Zur Kennzeichnung von Daten ist es immer logisch „0“.

Bei allen Laufzeitfehlern wird der aktuelle Befehl abgebrochen, der Prozessor befindet sich im HALT-Zustand und der Fehler wird durch eine Meldung angezeigt. Vom Nutzer ist die ESC-Taste zu betätigen, und COMPI16 behebt den Fehler:
- Bit 15 von BR wird gesetzt (Befehl)
- Bit 15 von Akkumulator und Operandenregister werden gelöscht (Daten)
- Der alte Speicherinhalt wird wiederhergestellt.

Danach ist das Programm zu überprüfen, um die Fehlerursache zu klären.

STEP (F2)
Bei Betätigen dieser Taste wird nur ein einzelner Befehl ausgeführt, er kann nicht unterbrochen werden.

Speed (F3)
Mit dieser Taste kann zwischen den zwei möglichen Arbeitsgeschwindigkeiten (Slow/Fast) umgeschaltet werden. Bei langsamer Betriebsart wird die Arbeit im Detail dargestellt, bei schnellem Betrieb sind nur die Ergebnisse der Instruktionen sichtbar. Der jeweils aktive Modus wird im MODE-Fenster angezeigt.

Mn/Bn (F4)
Mit dieser Taste (Mnemonic/Binary) kann zwischen zwei Darstellungsarten umgeschaltet werden. Im Modus „Binary“ werden alle Register sowie der Speicher binär dargestellt, im „Mnemonic“-Modus erfolgt die Befehlsdarstellung mnemonisch und die Daten- und Adreßdarstellung dezimal oder hexadezimal (umschalten mit CTRL+H). Der jeweils aktive Modus wird im MODE-Fenster angezeigt.

BZ (F5)
Mit dieser Taste kann ein Wert in den Befehlszähler BZ binär eingegeben werden.

Akku (F6)
Mit dieser Taste kann ein Wert in den Akkumulator binär eingegeben werden. Bit 15 bleibt dabei immer Null (Kennzeichnung von Daten).

MEM (F7)
Mit dieser Taste kann ein Wert in eine HS-Speicherzelle binär eingegeben werden. Verändert werden kann nur die Speicherzelle, die im Speicherfenster neben dem Datenweg zum Speicher steht. Dazu muß ggf. die zu editierende Speicherzelle mittels Cursortasten an die gewünschte Position gebracht werden. Diese Verschiebungen des Speicherfensters sind für den eigentlichen Programmablauf ohne Bedeutung und brauchen nicht rückgängig gemacht zu werden.

Break (F8)
Das Breakpointregister ist ein Parallelregister zum Befehlszähler und dient beim Testen von Programmen zum Setzen eines Unterbrechungspunktes. Ist ein solcher Breakpoint gesetzt, so wird das Programm unterbrochen, wenn der BZ den im Brekpointregister angegebenen Wert erreicht. Damit kann das Programm zu Testzwecken an einer bestimmter Stell angehalten werden, aber nur, wenn das Programm diese Stelle auch wirklich erreicht (!). Soll ein Breakpoint verwendet werden, muß der gewünschte Adreßwert in das Breakpointregister eingetragen und anschließend der Breakpoint aktiviert werden (INS- bzw. EINFG-Taste). Nach dem Drücken von F8 erscheint auf dem Bildschirm das Breakpointregister statt des BZ, und es kann ein Wert binär eingegeben werden. Bei aktivem Breakpoint steht über dem Bit 0 des BZ ein großes „B“. Dieses „B“ blinkt, wenn der Breakpoint im Programmablauf erreicht

wird. Das Blinken wird erst durch Deaktivieren (DEL- bzw. Entf-Taste) des Breakpoints oder erneutes Starten des Programms beendet.

Help (F9)
Auf dem Bildschirm erscheint eine kurze Erklärung der Bedientasten. Durch Drücken der Leertaste wird die Hilfefunktion verlassen.

Quit (F10)
Der COMPI16-Simulator wird beendet.

Außer den genannten Funktionstasten werden noch folgende Tasten verwendet:

Cursortasten „↓“ und „↑“ **Seitentasten „Bild ↑“ und „Bild ↓“**
Diese Tasten verschieben das angezeigte Speicherfenster.

Einfg (INS) bzw. Entf (DEL)
Diese Tasten aktivieren bzw. deaktivieren den Breakpoint.

Strg+T
Diese Tastenkombination löscht den Tastaturpuffer.

Strg+B
Diese Tastenkombination löscht den Bildschirm des Modellcomputers.

Strg+A
Diese Tastenkombination faßt Strg+T und Strg+B zusammen.

Strg+H
Umschalten der Darstellung von Daten und Adressen zwischen dezimal und hexadezimal; Im Modus „Binär/Binary“ können nur die HS-Adressen hexadezimal dargestellt werden.

Alle anderen Tasten dienen der normalen Zeicheneingabe, das Zeichen der betätigten Taste gelangt dabei in den Tastaturpuffer. Dabei sind nur die druckbaren Zeichen (ohne Umlaute) und die Steuerzeichen CR (carriage return) und LF (line feed) aus dem ASCII-Zeichensatz zulässig. Andere Tasten bleiben wirkungslos. Das Steuerzeichen LF (line feed) wird über Strg+J erzeugt.

4. Befehle und Befehlsausführung

Der simulierte Prozessor kann 32 verschiedene Befehle ausführen, die sich in folgende Befehlsgruppen untergliedern lassen:

- Transportbefehle
- arithmetische und logische Befehle
- Sprungbefehle
- Ein-/ Ausgabebefehle
- Steuerbefehle

Jeder Befehl belegt genau ein Speicherwort, wobei zur Unterscheidung von Daten und Adressen das höchstwertige Bit (Bit 15) den Wert „1“ hat.

4.1. Adressierungsarten

Der Prozessor unterscheidet zwei Adressierungsarten → direkte und indirekte Adressierung. Bei der direkten Adressierung steht die Operandenadresse im Befehlswort. Bei der indirekten Adressierung (im engeren Sinne) befindet sich die Adresse einer Speicherzelle, in der die Operandenadresse zu finden ist, im Befehl.

4.2. Befehlsformat

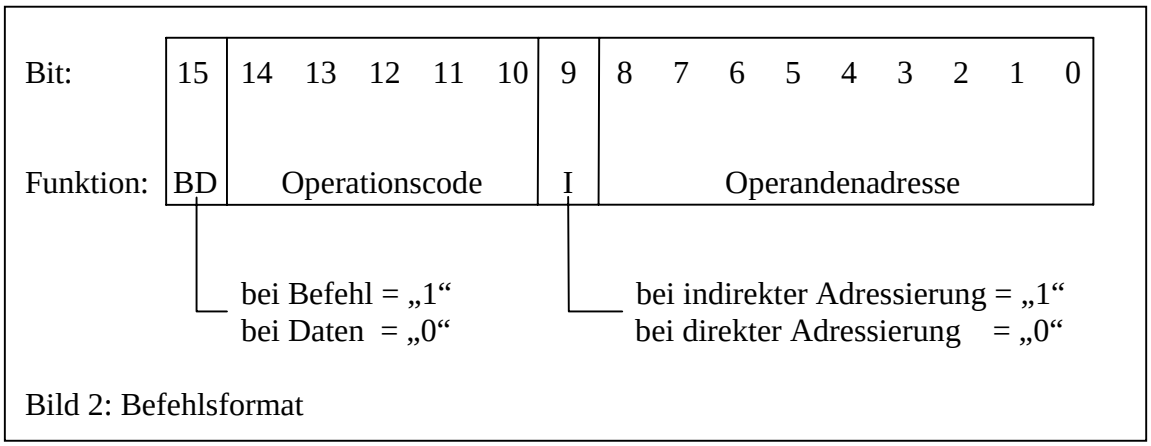

Bild 2: Befehlsformat

Bei Operationen mit zwei Operanden befindet sich der erste Operand im Akkumulator und der zweite im Speicher. Auch das Ergebnis steht im Akkumulator. COMPI16 ist also eine typische Einadreßmaschine. Bei Befehlen ohne notwendige Adressangabe sind die Bits 0-8 bedeutungslos.
In den nachfolgenden Erläuterungen der einzelnen verfügbaren Maschinenbefehle steht „Adr.“ entweder für eine direkte oder für eine indirekte Adresse. Die Unterscheidung erfolgt intern durch Auswertung des Bits 9. In der mnemonischen Darstellung wird bei indirekter Adressierung dem Operationscode die Zeichenkombination „ ,I “ angehängt.

4.3. Transportbefehle

LDA Adr. **l**oa**d** **a**ccu (<AC> := <Adr.>)
STA Adr. **st**ore **a**ccu (<Adr.> := <AC>)

4.4. Arithmetische und logische Befehle

ADD Adr. **add** to accu (addiert Datenwort zum Akkumulatorinhalt, Operationsergebnis beeinflußt die Flags)
SUB Adr. **sub** from accu (subtrahiert Datenwort vom Akkumulatorinhalt, Operationsergebnis beeinflußt die Flags)
CMP Adr. **c**o**mp**are with accu (Datenwort wird mit Akkumulatorinhalt verglichen, Ergebnis der Operation = Akkumulatorinhalt minus Datenwort beeinflußt nur die Flags)
AND Adr. logic **and** with accu (logische UND-Verknüpfung von Datenwort mit Akkumulator, beeinflußt Flags nicht)
IOR Adr. logic **or** with accu (logische ODER-Verknüpfung von Datenwort mit Akkumulator, beeinflußt Flags nicht)
XOR Adr. logic e**x**clusive **or** with accu (logische EXCLUSIV-ODER-Verknüpfung von Datenwort mit Akkumulator, beeinflußt Flags nicht)

SHR **sh**ift **r**ight (Verschiebung des Akkumulatorinhalts um eine Bitstelle nach rechts, beeinflußt Flags nicht)
SRE **s**hift **r**ight with **e**xtension (Rotieren des Akkumulatorinhalts um eine Bitstelle nach rechts durch das Extensions-Flag)
SHL **sh**ift **l**eft (wie SHR, nur links)
SLE **s**hift **l**eft with **e**xtension (wie SRE, nur links)

4.5. Sprungbefehle

unbedingter Sprung:
JMP Adr. **j**u**mp** (Sprung zur angegebenen Adresse)

bedingte Sprünge:

		Sprung zur angegebenen Adresse, wenn folgende Flagbelegung:						
	jump if	<	=	>		<	=	>
JLT Adr.	**l**ess **t**hen	1	0	0				
JLE Adr.	**l**ess or **e**qual	1	0	0	oder	0	1	0
JEQ Adr.	**eq**ual	0	1	0				
JGE Adr.	**g**reater or **e**qual	0	1	0	oder	0	0	1
JGT Adr.	**g**reater **t**hen	0	0	1				
JNE Adr.	**n**ot **e**qual	1	0	0	oder	0	0	1

Unterprogrammsprung und Rückkehr:
JSB Adr. **j**ump to s**ub**routine (Retten des Befehlszählerinhaltes in den Stack und Sprung zur angegebenen Adresse: <<SP>> := <BZ>
<BZ> := Adr.)
RET **ret**urn from subroutine (Wiederherstellen des Befehlszählerinhaltes aus dem Stack: <BZ> := <<SP>>)

4.6. Ein-/ Ausgabebefehle

RDA **r**ea**d** into **a**ccu (transportiert ASCII-Code des ersten Zeichens aus dem Tastaturpuffer in den Akkumulator. Ist der Tastaturpuffer leer, so wird 0000H in den Akkumulator geladen.)
WRA **wr**ite **a**ccu onto screen (schreibt Zeichen aus dem Akkumulator auf die aktuelle Cursorposition des Modellcomputerbildschirms. Der Bildschirm zeigt ASCII-codierte Zeichen an, z.B. 41H ergibt „A“)

4.7. Steuerbefehle

HLT **h**a**lt** (Modellprozessor geht in Halt-Zustand)
NOP **n**o **op**eration (keine Operation wird ausgeführt)

Von den 32 möglichen Befehlen werden nur 25 genutzt. Die restlichen 7 Bitkombinationen haben dieselbe Wirkung wie der Befehl HLT.

5. Assembler

Der Assembler ist ein Übersetzungsprogramm für maschinenorientierte Programmiersprachen. Er liefert ein relativ adressiertes (noch verschiebbares)

Objektprogramm. Die Übersetzung erfolgt in zwei Arbeitsgängen (Pässen). Während im ersten Pass im wesentlichen das Adreßbuch erstellt wird, ist die Aufgabe des zweiten Passes die eigentliche Überführung des symbolischen Operationscodes und der Adressen in den Interncode.
Das Objektprogramm muß noch mit dem Linker in ein abarbeitbares Programm (Lademodul) übergeführt werden.

5.1. Format von Assembleranweisungen

Jede Anweisung gliedert sich in vier Felder:

[Label] Mnemocode[,I] [Operandenadresse] [;Kommentar]

Im Label- und Operandenadreßfeld stehen symbolische (keine numerischen) Adressen, die maximal sieben Zeichen lang sein dürfen. Ein Label ist dadurch gekennzeichnet, daß es in der ersten Spalte der Zeile beginnt. Steht in der ersten Spalte ein Leerzeichen, so nimmt der Assembler ein leeres Labelfeld an, d.h. ein einzelner Befehl ohne Label beginnt frühestens in der zweiten Spalte. Die einzelnen Felder sind durch wenigstens ein Leerzeichen getrennt. Der Zusatz „ ,I “ zum mnemonischen Operationscode wird ohne Zwischenraum angefügt und kennzeichnet indirekte Adressierung.
Label, Mnemocode und Operandenadresse sind in GROSSBUCHSTABEN zu schreiben, und es dürfen keine Tabulatorensprünge verwendet werden. Am Ende des Quellfiles ist eine Leerzeile einzugeben.

Beispiel:
```
ADR1        SUB,I  ADR2        ;das ist ein Kommentar
```

Als Assembleranweisungen können auftreten:
- Maschinenbefehle in symbolischer Form
- Pseudobefehle (Direktiven)

5.2. Maschinenbefehle

Symbolische Form der intern verarbeitbaren Befehle;
Siehe 4.3. ... 4.7.!

5.3. Pseudobefehle

Reservierung von Speicherplatz: Ein Speicherplatz besteht aus 16 Bit (ein Wort).

[Label] DEC n [;Kommentar]
Reservieren eines Speicherplatzes, der die Dezimalzahl n enthält, unter der symbolischen Adresse „Label“;

[Label] HEX h [;Kommentar]
Reservieren eines Speicherplatzes, der die Hexadezimalzahl h enthält, unter der symbolischen Adresse „Label“;

[Label] RST n [;Kommentar]
Reservieren von n aufeinanderfolgenden Speicherplätzen mit dem Inhalt Null unter der symbolischen Adresse „Label“;

[Label1] DEF [Label2] [;Kommentar]
Reservieren eines Speicherplatzes, der den Wert der angegebenen symbolischen Adresse „Label2“ enthält, unter der symbolischen Adresse „Label1;

Externe Bezüge:

EXT Label
Dem Assembler wird mitgeteilt, daß die symbolische Adresse „Label“ in einem anderen Quellmodul definiert ist.

ENT Label
Dem Assembler wird mitgeteilt, daß die in diesem Quellmodul definierte symbolische Adresse auch für andere Module zur Verfügung gestellt werden soll (Pendant zu EXT).

Die Anweisungen EXT und ENT müssen am Anfang des Quellmoduls angegeben werden.

5.4. Programmbeispiel

```
;Dieses Programm schreibt die Buchstaben A bis F auf den Schirm.
            LDA   ACODE     ;ASCII-Code für „A“ laden
LOOP        WRA             ;Zeichen auf Schirm ausgeben
            CMP   ECODE     ;mit Endcode vergleichen
            JEQ ENDE        ;wenn Endcode, dann Sprung zu ENDE
            ADD EINS        ;Zeichencode um 1 erhöhen
            JMP LOOP        ;Sprung zurück, alles wiederholen
ENDE        HLT             ;fertig
ACODE       DEC 65          ;ASCII-Code für „A“
ECODE       HEX 46          ;ASCII-Code für „F“
EINS        DEC 1           ;dezimale Eins zum inkrementieren
```

5.5. Bedienung

Der Assembler wird mit ASSEM.EXE aufgerufen. Anschließend wird die Eingabe des Quelldateinamens verlangt, der mit .TXT enden sollte. Mit <ENTER> kann die Abfrage des Ausgabedateinamens bestätigt werden. Die Assemblierung benötigt zwei Läufe, die jeweils mit F4, entweder im Slow- oder Fast-Modus (ähnlich COMPI16), zu starten sind. Der Assembler liefert eine verschiebbare Datei mit der Erweiterung .REL, das mit dem Linker weiter zu bearbeiten ist.

6. Linker (Binder)

Die vom Assembler erzeugten Dateien werden mit dem Linker in .LAD-Dateien umgewandelt, die vom Modellcomputer COMPI16 abgearbeitet werden können. Dazu löst der Linker externe Bezüge auf und trägt absolute Adressen ein.
Er wird mit LINKER.EXE gestartet. Anschließend wird die Eingabe der Namen aller vom Assembler übersetzten Dateien (u.U. nur eine Datei) verlangt, die zu einem einzigen ablauffähigen Programm verbunden werden sollen. Nach dem letzten Namen ist <Enter> einzugeben. Die Frage nach dem Namen der Codedatei für COMPI16 (Lademodul) kann im Anschluß mit <ENTER> bestätigt werden. Beim Linken sind drei Läufe notwendig, wobei die Bedienung ähnlich dem Assembler gestaltet ist. Mit dem entstandenen .LAD-File kann der Simulator COMPI16 gestartet werden.

7. Beispielaufgabe

Zur Unterstützung der Vorbereitung und ggf. auch der Durchführung des Praktikums wird im folgenden der Quelltext zur Realisierung der Ausgabe einer zweistelligen Dezimalzahl angegeben. Das Programmstück ist als separater Quellmodul gestaltet und kann nach der Übersetzung (===> Objektmodul) mittels des Programmverbinders (Linker) zusammen mit anderen Objektmodulen zu einem abarbeitbaren Lademodul verbunden werden. Es enthält die Ausgabe als Unterprogramm mit der Adresse UPAUS. Die auszugebende Zahl ist im Akkumulator bereitzustellen. Der Akkumulatorinhalt wird im Unterprogramm verändert.
Um die Arbeitsweise von Assembler und Linker kennenzulernen, sind beide Läufe mindestens einmal im Langsamlauf abzuarbeiten.

Quelltext des Moduls AUS.TXT

```
        ENT     UPAUS   ;symb. Adresse UPAUS auch in anderen
                        ;Modulen bekannt machen
UPAUS   STA     ZAHL    ;Akkumulator speichern
        LDA     TABMERK ;Anfangsadresse der Tabelle holen
        STA     TAB     ;und zur weiteren Berechnung ablegen
        LDA     ZAHL    ;auszugebende Zahl in den Akkumulator laden
ZYKL1   CMP     ZEHN    ;Test, ob Zahl kleiner als Zehn ist
        JLT     AUS1    ;ja, also Sprung zum Ausgeben
                        ;nein, noch größer oder gleich Zehn, also zyklisch
                        ;Zehn abziehen und dabei Tabellenzeiger erhöhen
        LDA     TAB     ;Tabellenzeiger holen
        ADD     EINS    ;Zeiger auf nächste Ziffer setzen
        STA     TAB     ;veränderten Zeiger wieder abspeichern
        LDA     ZAHL    ;Zahl holen
        SUB     ZEHN    ;Zehn abziehen
        STA     ZAHL    ;und wieder abspeichern
        JMP     ZYKL1   ;diesen Zyklus solange ausführen, bis Zahl
                        ;kleiner als Zehn ist
        ;Ausgabe
AUS1    LDA,I   TAB     ;über Tabellenzeiger auszugebenden ASCII-Code
                        ;holen
        WRA             ;ASCII-Code der Zehnerstelle ausgeben
        LDA     TABMERK ;Anfangsadresse der Tabelle holen
        ADD     ZAHL    ;Tabellenzeiger um den Wert der Einerstelle
                        ;erhöhen
        STA     TAB     ;Tabellenzeiger für indirekte Adressierung
                        ;speichern
        LDA,I   TAB     ;über Tabellenzeiger auszugebenden ASCII-Code
                        ; holen
        WRA             ;ASCII-Code der Einerstelle ausgeben
        RET             ;finales Return
        ;Definitionen
ZAHL    RST     1       ;Speicher für auszugebende Zahl
EINS    DEC     1       ;dezimale Eins für Zeigerrechnungen
ZEHN    DEC     10      ;dezimale Zehn für Abtrennung der Zehnerstelle
                        ;von Zahl
TAB     RST     1       ;Speicherzelle für Tabellenzeiger
```

```
TABMERK   DEF     TABANF   ;Ablage für Tabellenanfangsadresse
TABANF    HEX     30       ;Tabelle der ASCII-Codes der Ziffern von Null
                           ;bis Neun
          HEX     31       ;ASCII-Code von Eins
          HEX     32       ;
          HEX     33       ;          :
          HEX     34       ;          :
          HEX     35       ;          :
          HEX     36       ;
          HEX     37       ;
          HEX     38       ;
          HEX     39       ; ASCII-Code von Neun
```

Schematisch ist die Programmerstellung für COMPII16 zur Beispielaufgabe folgendermaßen darstellbar (Bild 3):

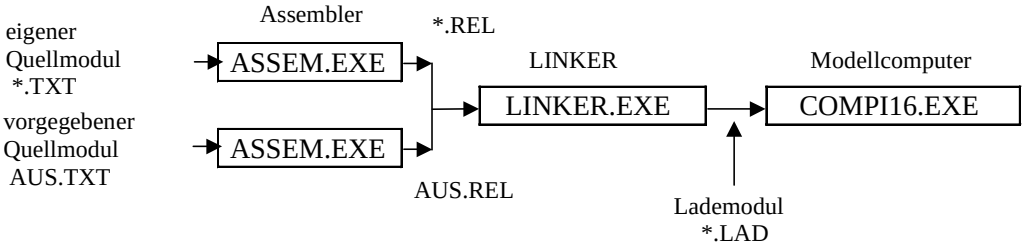

Bild 3: Schema der Programmerstellung

8. Kolloquiumsschwerpunkte

- Aufbau und Wirkungsweise eines Digitalrechners
- Zentrale Steuerschleife
- Informationsdarstellung
- Arbeitsweise einer Einadreßmaschine
- Befehlsformate, Operandentypen, Adressierungsarten und Adreßraum
- Stufen der Informationsverarbeitung mittels eines Digitalrechners
- Arbeitsweise eines Assemblers
- Befehle und Pseudobefehle
- Makros und Unterprogramme
- Wirkungsweise des Programms zur Praktikumsaufgabe

9. Literaturhinweise

/1/ Rieger, P.: Vorlesung „Computertechnik I“

/2/ Schulze, R.: Vorlesung „Informatik“

/3/ Link, W.: Assembler-Programmierung
Franzis‘ Verlag GmbH, Poing, 1999

/4/ Backer, R.: Programmiersprache ASSEMBLER – Eine strukturierte Einführung
Rowohlt Taschenbuch Verlag GmbH, Reinbek bei Hamburg, 1998

10. Arbeits- und Brandschutzhinweise

Vorbeugende Maßnahmen:

- Die Praktikumsteilnehmer haben sich so zu verhalten, daß Gefahrensituationen und Unfälle vermieden werden.
- Die Befugnis zum Bedienen und Nutzen von Geräten ist auf den zugewiesenen Praktikumsplatz beschränkt.
- Eingriffe in die zum Praktikumsaufbau gehörenden Geräte sind nicht erlaubt.
- Der Anschluß und der Betrieb privater Geräte in den Praktikumsräumen ist verboten.
- Defekte an Geräten oder Gebäudeeinrichtungen sind unverzüglich dem Betreuer mitzuteilen. Betroffene Geräte sind außer Betrieb zu nehmen. Andere Personen sind vor Gefahren zu warnen.
- Den Anweisungen der Praktikumsbetreuer bzw. anderer aufsichtsführender Personen ist unbedingt Folge zu leisten.
- Betriebsfremde dürfen sich nur mit Erlaubnis des Praktikumsbetreuers in den Praktikumsräumen aufhalten.
- Rauchen und Umgang mit offenem Feuer ist nicht gestattet.
- Nach Ende des Praktikums ist der Arbeitsplatz in sauberem und aufgeräumtem Zustand zu hinterlassen.
- Außergewöhnliche Ereignisse bzw. besondere Vorkommnisse sind umgehend dem Betreuer oder dem diensthabenden Assistenten zu melden.

Verhalten im Falle eines Brandes:

- Beachten der richtigen Reihenfolge: MELDEN – RETTEN – LÖSCHEN

- FEUER MELDEN:
  * Telephonische Brandmeldung:
    Notruf 112 der Feuerwehr (von jedem Telephon aus möglich)
    Notruf HA 34515 der Technischen Leitzentrale der TUD
  * Deutliche, genaue und vollständige Angaben:
    Wo brennt es?
    Was brennt?
    Angaben zu verletzten oder gefährdeten **Personen**
    Wer meldet?
- PERSONEN RETTEN:
  * Erste Hilfe leisten
  * Weitere Hilfe organisieren, medizinische Hilfe anfordern
  * Gefahrenbereich räumen; Fluchtwege benutzen, keine Aufzüge
  * Andere Personen warnen, Sammelplatz (Platz vor Turmeingang zum Barkhausenbau) aufsuchen
  * Behinderten und älteren Personen helfen
- LÖSCHVERSUCH UNTERNEHMEN
  * Feuerlöscher verwenden (Standorte: Gänge des Barkhausenbaues), dabei nicht selbst gefährden
  * Fenster und Türen schließen, aber nicht abschließen
  * Möglichst elektrische Verbraucher abschalten

Rufnummern für Notfälle:

Rettungsdienst:	112	Polizei:	110
TUD – Notruf:	HA 34515	Betriebsärztin Dr. Hentschel:	HA 36199 oder 36255
Klinikum Friedrichstadt:	0 4800 (Amtsanschluß)		