NAME

pep-8 − Beschreibung der pep-8 Architektur

BESCHREIBUNG

Der **pep−8** (Practical Education Processor based on pdp-8) ist von der **DEC pdp-8** abgeleitet und soll der Vermittlung grundlegender Konzepte von Rechnerarchitekturen und Assemblerprogrammierung dienen.

Der Prozessor lehnt sich am historischen Prozessor der *DEC* **pdp-8** an, vereinfacht aber an vielen Stellen und nimt keine Rücksticht auf die tatsächliche Implementierung in Hardware. Für Fortgeschrittenen werden Ein-/Ausgabegeräte und Unterbrechungsverarbeitung unterstützt.

PROGRAMMIERMODELL

Der extrem einfach aufgebaute pep-8 Prozessor besteht aus einem Programmzähler (Program Counter, PC) und einem Akkumulator (Accumulator, AC), die zwölf Bit breit sind sowie dem Verbindungsbit (Link Bit, L). Der Speicher ist ebenfalls 12 Bit breit ist und wird mit 12 Bit Adressen angesprochen. Er umfasst also 4096 12−Bit Worte. Andere Einheiten als ein 12−Bit Wort können nicht adressiert werden. Die Bits eines Wortes werden von null bis elf durchnummeriert, wobei Bit null das signifikateste Bit ist:

0	1	2	3	4	5	6	7	8	9	10	11

ADRESSIERUNGSARTEN

Bei allen Befehlen, die einen Hauptspeicheroperanden haben, wird dessen Adresse in Bit fünf bis elf codiert. Der Speicher ist logisch in Seiten zu 128 Worten aufgeteilt und eine Operandenadresse ist immer relativ zum Beginn der Speicherseite, in der sich der Befehl befindet, oder der Seite null ("Zero Page"). Welche dieser Alternativen gewählt wird enscheidet das "Zero Page" Bit, das im Befehl in Bit vier gespeichert wird. Ist es null, wird relativ zur Seite null adressert, sonst relativ zur laufenden Seite. Wird das Indirekt−Bit gesetzt, das sich in Bit 3 befindet, so wird der Operand als 12−Bit Zeiger auf ein Speicherwort betrachtet, auf das sich der Befehl dann bezieht. In den Bits 0 bis 2 wird der Befehl kodiert.

B0	B1	B2	I	ZP	A0	A1	A2	A3	A4	A5	A6
0	1	2	3	4	5	6	7	8	9	10	11

Das ergibt die folgenden Adressierungsarten:

Name	I	ZP	Bits	Erreichbare Worte
Seitenrelativ	0	1	7 Bit	In der aktuellen Seite
Zero Page	0	0	7 Bit	In der nullten Seite
Indirekt	1	1	12 Bit	Im gesamten Speicher, Zeiger in der aktuellen Seite
Indirekt Zero Page	1	0	12 Bit	Im gesamten Speicher, Zeiger in der nullten Seite

Die effektive Adresse für jeden Befehl mit einem Hauptspeicheroperanden berrechnet sich aus der Adresse im Befehl (7 Bit, A0-7) und der effektiven Seite (effective page, P0-4). Die effektive Seite ist Null wenn das Zero−Page−Bit null ist, und entspricht sonst den fünf signifikatesten Bits des Programmzählers.

P0	P1	P2	P3	P4	A0	A1	A2	A3	A4	A5	A6
0	1	2	3	4	5	6	7	8	9	10	11

Ist das Indirekt−Bit gesetzt wird dieser Wert noch dereferenziert, d.h. der Inhalt des durch ihn adressierten Speicherwortes bildet die effektive Adresse. Wird nicht indirekt adressiert, bildet der Wert aus effektiver Seite und Adresse im Befehlswort direkt die effektive Adresse.

BEFEHLE

Der pep-8 Prozessor verfügt über sieben Befehle die auf dem Hauptspeicher operieren. Bei diesen wird der Operationscode in den Bits null bis drei abgelegt. Zu jeden Befehl sind die Mnemonische Abkürzung, der englische Begriff aus den sich diese ableitet, sowie der Operationscode in Binärdarstellung aufgeführt.

RCL (Recall, 000)
: Läd das im Hauptspeicher adressierte Wort in den Akkumulator.

STO (Store, 001)
: Speichert den Akkumulator im adressierten Speicherwort.

AND (And accumulator, 010)
: Der Akkumulator wird mit dem adressierten Wort bitweise und-verknüpft. Das Ergebnis wird in den Akkumulator zurückgeschrieben.

ORA (Or accumulator, 011)
: Der Akkumulator wird mit dem adressierten Wort bitweise oder-verknüpft. Das Ergebnis wird in den Akkumulator zurückgeschrieben.

TAD (Two's Complement Add, 100)
: Der Operand wird zu dem 13−Bit Wert der duch das Verbindungsbit und Akkumulator gebildet wird (L, AC) addiert. Negative Zahlen werden im Zweierkomplement repräsentiert.

JMP (Jump, 101)
: Das adressierte Hauptspeicherwort wird in den Programmzähler geladen. Es wird also zur angegebenen Adresse gesprungen.

JMS (Jump to Subroutine, 110)
: Der Programmzähler wird im adressierten Hauptspeicherwort gespeichert. Dann wird die Adresse des darauf folgenden Speicherwortes in den Programmzähler geladen. Es wird also tatsächlich zu der Adresse gesprungen die auf die angegebene Adresse folgt. Ein Rücksprung erfolgt dann über einen indirekten Sprung mit der Startadresse der Routine als Argument.

Die verbleibende Bitkombination *111* (OPR, Operate) identifiziert einen der restlichen Befehle des pep-8, die keinen Hauptspeicheroperanden benötigen und deshalb die Bits drei bis elf zur Bestimmung des Befehls heranziehen können. Zu jedem Befehl wird hier der vollständige 12−Bit Befehlscode als Oktalzahl angegeben.

Befehle die den internen Zustand ändern

Diese Befehle können in einem Befehlswort kombiniert werden. Bei diesen Befehlen ist Bit 3 auf null. Den Bits 4 bis 11 ist jeweils eine Operation zugeordnet.

				CLA		CMA		RAR		DAC	
1	1	1	0		CLL		CML		RAL		IAC
0	1	2	3	4	5	6	7	8	9	10	11

Ist keines dieser Bits gesetzt, wird auch keine Operation ausgeführt (*No Operation*). Die Operationen werden schrittweise abgearbeitet, beginnend mit Bit 4 bis Bit 11 fortschreitend. So haben die verschiedenen Bitkombinationen eine definierte Bedeutung (CLA CMA löscht erst den Akkumulator und invertiert ihn dann).

NOP (No Operation, 7000)
: Nichtstun. Kann mit keinem anderen Befehl kombiniert werden.

CLA (Clear accumulator, 7200)
: Akkumulator auf Null setzen.

CLL (Clear Link Bit, 7100)
: Verbindungsbit auf Null setzen.

CMA (Complement accumulator, 7040)
: Einerkomplement des Akkumulators bilden und dort ablegen.

CML (Complement Link, 7020)
Verbindungsbit invertieren.

RAR (Rotate right Link and accumulator, 7010)
Den 13−Bit Wert Verbindungsbit-Akkumulator (L+AC) nach rechts rotieren.

RAL (Rotate left Link and accumulator, 7004)
Den 13−Bit Wert Verbindungsbit-Akkumulator (L+AC) nach links rotieren.

DAC (Decrement accumulator, 7002)
Akkumulator um eins vermindern, Verbindungsbit wird bei Unterlauf invertiert.

IAC (Increment accumulator, 7001)
Akkumulator um eins erhöhen, Verbindungbit wird bei Überlauf invertiert.

Kombinationen aus Befehlen denen Mnemonische Abkürzungen zugeordnet sind:

STA (Set accumulator, 7240, CLA CMA)
Alle Bits des Akkumulators auf Eins setzen.

STL (Set Link Bit, 7120, CLL CML)
Verbindungsbit auf Eins setzen.

CIA (Complement and increment accumulator, 7041, CMA IAC)
Bildet das Zweierkomplement des Akkumulators.

Befehlskomnination ermöglicht kompakte Subtraktion, die ja keinen eigenen Befehl hat. Man läd den Subtrahenden in den Akkumulator und wendet den kombinierten Befehl **CMA IAC** bzw. **CIA** (7041) an, der das Zweierkomplement bildet. Anschließend wird mit **TAD** der Minuend addiert und die Differenz steht im Akkumulator. Wenn nötig kann im ersten Schritt zusätzlich noch das Verbindungsbit gelöscht werden (CLL) womit sich dann das Bitmuster 7141 (**CLL CMA IAC**) ergibt.

Befehle die den Programmfluß verändern

Bei diesen Befehlen ist Bit 3 gesetzt und Bit 4 auf null. Diese Befehle können in einem Befehlswort kombiniert werden, solange die Befehle zum gleichen Kombinationstyp gehören. Die Sprungbedingungen entsprechend den Bits 6 bis 8 werden oder-verknüpft, wenn Bit 5 des Befehlswortes null ist (*Oder-Typ*).

Ist Bit 5 eins, wird die Bedingung invertiert, d.h. der Sprung wird nicht genommen, wenn eine der durch Bit 6 bis 8 selektierten Bedingungen wahr ist. Dadurch ergeben sich drei neue Sprungbedingungen, die und-verknüft werden, um zu bestimmen ob der Sprung genommen wird (*Und-Typ*).

Ist keines der Bedingungsbits 6 bis 8 gesetzt wird niemals gesprungen (*Skip never*), effektiv ein (zweites) NOP. Wird dies mit mit Bit 5 invertiert, wird immer gesprungen (*Skip always*).

						SMA		SNL			
1	1	1	1	0	Inv		SZA				
0	1	2	3	4	5	6	7	8	9	10	11

SKN (Skip never, no operation, 7400)
Nicht springen - nichtstun.

SNL (Skip on non-zero Link, 7410, Oder-Typ)
Wenn das Verbindungsbit nicht null ist, den Programmzähler erhöhen.

SZA (Skip on zero accumulator, 7420, Oder-Typ)
Wenn der Akkumulator null ist, den Programmzähler erhöhen.

SMA (Skip on minus accumulator, 7440, Oder-Typ)
Bei negativem Akkumulator den Programmzähler erhöhen.

SKP (Skip always, 7500)
Den Programmzähler erhöhen. (ohne Bedingung)

SZL (Skip on zero Link, 7510, Und-Typ)
Wenn das Verbindungsbit null ist, den Programmzähler erhöhen.

SNA (Skip on non-zero accumulator, 7520, Und-Typ)
Wenn der Akkumulator nicht null ist, den Programmzähler erhöhen.

SPA (Skip on plus accumulator, 7540, Und-Typ)
Bei positivem Akkumulator den Programmzähler erhöhen.

Befehle zum Ansprechen von Ein-/Ausgabegeräten

Mit diesen Befehlen werden Ein-/Ausgabegeräte angesteuert. Der pep-8 Prozessor unterstützt bis zu 16 Geräte (0..15) die mehrere Untereinheiten unterstützen können. Bei diesen Befehlen sind Bit 3 und Bit 4 gesetzt. Das jeweils angesprochene Gerät wird im Befehlswort in den Bits 8 bis 11 kodiert. In den Bits 5 bis 7 ist der E/A Kode, der die Operation bestimmt gespeichert.

1	1	1	1	1	I0	I1	I2	D0	D1	D2	D3
0	1	2	3	4	5	6	7	8	9	10	11

Die angegeben Mnemonischen Abkürzungen gelten für zeichenorientierte Geräte. Andere Geräte können die Bits 5 bis 7 unter Umständen anders verwenden. Diese Befehle können nicht kombiniert werden.

SRI (Skip on ready for input, 7600)
Den Programmzähler erhöhen wenn der Eingabekanal bereit ist.

SRO (Skip on ready for output, 7620)
Den Programmzähler erhöhen wenn der Ausgabekanal bereit ist.

DGA (Device get word to accumulator, 7640)
Den Akkumulator mit dem aktuellen Wert des Eingabekanals laden.

DPA (Device put word from accumulator, 7660)
Den Wert des Akkumulators in den Ausgabekanal schreiben.

DUS (Device unit select, 7700)
Untereinheit gemäß dem Wert im Akkumulator auswählen.

DGS (Device get status word, 7720)
Statuswort des Geräts bzw. der Untereinheit im Akkumulator speichern.

DSM (Device sense mask, 7740)
Den Wert des Akkumulators als Meldungsauswahlmaske an das Gerät übergeben.

RSD (Reset device, 7760)
Das Gerät zurücksetzen und Untereinheit null auswählen.

UNTERBRECHUNGEN

Der **pep8**(7) unterstützt Unterbrechungen durch Ein-/Ausgabegeräte. Sind Unterbrechungen freigeschaltet und ein Gerät erzeugt eine Unterbrechungsanforderung, so wird nach Beendigung des aktuellen Befehls ein implezites **JMS I 0** durchgeführt. Es wird also die Addresse des Befehls, der ohne vorliegen einer Unterbrechungsanforderung ausgeführt worden wäre in Adresse 0 (Null) gespeichert und die Ausführung an Adresse 1 fortgesetzt. Ausserden wird die Unterbrechungsverarbeitung blockeier, so das die Rücksprungadresse nicht von einer weiteren Unterbrechung überschrieben werden kann.

Da verschiedene Geräte eine Unterbrechung anfordern können, ist es Aufgabe der Unterbrechungs-Bearbitungsroutine festzustellen welches Gerät die Unterbrechung angefordert hat. Ausserdem hat die Routine dafür zu sorgen das der Inhalt von Akkumulator und Verbindungsbit gesichert werden, damit sie beim Ende der Unterbrechungsroutine wieder in den Zustand versetzt werden können die sie bei der Unterbrechung hatten.

Am Ende der Routine können dann auch die Unterbrechungen wieder ativiert werden, bevor über ein **JMP I 0** der normale Programmablauf wieder aufgenommen wird.

Das aktivieren und deaktivieren der Unterbrechungsverarbeitung geschieht über das *Prozessor* Gerät mit der Nummer 15 mit Hilfe der Befehle **ION** *(Interrupts On)* und **IOF** *(Interrupts Off)*. Für jedes Gerät, das Unterbrechungen unterstützt, müssen diese über die Meldungsauswählmaske freigeschaltet werden. Es wird alo nur eine Unterbrechung angefordert wenn Unterbrechungen grundsätzlich über **ION** freigeschaltet sind und zusätzlich die entsprechende Gerätespezifische Unterbrechung über die Meldungsauswahlmaske freigeschaltet ist.

GERÄTE

Teletype, Gerät 0 (Null)

In Verbindung mit dem Terminalsimulator **teletype**(1) kann **pepsi**(1) Zeichen ein- und ausgeben, wenn das Gerät beim Start freigeschaltet wird. Das Teletype (TTY) unterstützt die Befehle **SRI**, **SRO**, **DGA**, **DPA** und **DSM**.

Über das Bit 0 der Auswahlmaske für **DSM** wird das Gerät veranlasst bei vorliegen eines Eingabezeichens eine Unterbrechung auszulösen. Bei freigeschalteten Unterbrechungen kann si mit **CLA CLL CML RAR** eine 1 in Bit 0 des Akkumulators geladen werden, um dann mit **DSM 0** die Unterbrechungen beim Vorliegen eines Eingabezeichens zu aktivieren.

pepsi(1) kommuniziert mit dem Terminalsimulator über einen TCP Verbindung, wobei **teletype**(1) den Server implementiert und **pepsi**(1) den Client. Entsprechend ist **teletype**(1) zuerst zu starten.

Wird der Terminalsimulator **teletype**(1) auf einem anderen Host oder Port als dem Standard localhost:4200 gestartet können Host und Port dem Gerät übergeben werden, i.e **-e 0**:**localhost**:**4200**. Wir nur ein anderer Host verwendet kann die Angabe des Ports entfallen.

Papertape Reader, Gerät 1

Dem Gerät wird beim Freischalten eine Datei zugewiesen aus der Einzelzeichen gelesen werden können. Am Dateiende geht das Grät in den nicht-bereit Zustand. Der Papertape Reader unterstützt die Befehle **SRI** und **DGA**.

Die Datei muss beim Freischalten des Gerätes angegeben werden, i.e. **-e** *meinedatei.tap*

X/Y-Point Plotter (Scope), Gerät 2

Dieses Gerät simuliert ein Oszilloskop dessen X- und Y-Eingänge über zwei AD-Wandler angesteuert werden. Programmgesteuert kann - nach der Ansteuerung der korrekten Koordinaten - der Elektronenstrahl kurz verstärkt werden und so ein Punkt im (lang nachleuchtenden) Phosphor des Bildschirms erzeugt werden.

Mit **DGA** und **DPA** werden die X- bzw. die Y-Koordinate gesetzt. Mit **SSR** kann abgefragt werden ob der Strahl bereits positioniert ist, dann kann mit **SIP** der Strahl kurzzeitig verstärkt werden um den Leuchtpunkt zu erzeugen.

pepsi(1) kommuniziert mit dem X/Y-Point Plotter über einen TCP Verbindung, **scope**(1) implementiert den Server, entsprechend ist es zuerst zu starten.

Wird der X/Y-Point Plotter **scope**(1) auf auf einem anderen Host oder Port als dem Standard localhost:4321 gestartet können Host und Port dem Gerät übergeben werden, i.e **-e 0**:**localhost**:**4321**. Wir nur ein anderer Host verwendet kann die Angabe des Ports entfallen.

GPIO 7-Segment Anzeige, Gerät 3

Auf Platformen mit General-Purpose I/O (aktuell wird die Raspberry Pi Familie unterstützt) kann über zwei GPIO Pins eine 7-Segement Anzeige mit TM1637 Controller-Chip angesteuert werden. Es werden zwei aufeinander folgende Pins für **Clock** und **Data** benötigt. Die Nummer des ersten Pins (Clock) kann als Parameter übergeben werden, der Default ist Pin 12, entsprechend *-e 3:12*. Als zweiter Pin wird dann Pin 1 für Data benutzt.

Das Gerät unterstützt **RSD** zum Löschen der Anzeige, **DUS** zum Auswählen der Ziffer (wobei die zu verwendenden Werte vom Anzeigen-Modell abhängen, der Prototyp hat die Ziffer mit Index "0" ganz links) und **DPA** zum Anzeigen eines Wertes. Dabei wird der Wert des Akkumulators als Dezimalziffer angezeigt.

GPIO Ein-/Ausgabe, Gerät 4

Auf Platformen mit General-Purpose I/O (aktuell wird die Raspberry Pi Familie unterstützt) können mit diesem Gerät bis zu 12 GPIO Pins zur Ein- oder Ausgabe verwendet werden. Dabei entspricht Bit 11 (LSB) dem GPIO Pin 2 und Bit 0 (MSB) dem Pin 14.

Mit dem **DSM** Befehl wird selektiert welche Pins verwendet werden (ein gesetztes Bit selektiert einen Pin). Der**DUS** Befehl bestimmt welche Pins als Ausg abe und welche als Eingabe verwendte werden. Ein gesetztes Bit selektiert den Pin zur Ausgabe.

Mit **DPA** werden die vorher für Ausgabe selektierten Pins auf die entsprechenden Bitwerte im Akkumulator gesetzt. Die für Eingabe selektierten Pins werden mit **DGA** eingelesen und an die

entsprecheden Positionen im Akkumulator geschrieben. Nicht für die Eingabe selektierte Pins haben keinen definierten Zustand.

Prozessor, Gerät 15

Über die Gerätenummer 15 wird der Prozessor gesteuert. Dies wird verwendet um die CPU programmgesteuert mit eine RSD Befehl (E/A Code 7) anzuhalten (i.e. am Programmende).

HLT (Halt the program, 7777, RSD 0017)
Das Programm anhalten.

ION (Enable interrupte, 7637, SRO 0017)
Unterbrechungen ermöglichen.

IOF (Disable interrupts, 7657, DGA 0017)
Unterbrechungen blockieren.

SIEHE AUCH

pot(1), **pot**(5), **pepsi**(1), **teletype**(1), **scope**(1)