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.

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. 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.

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

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. 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.

BSW (Byte Swap, 7002)
Die linke (Bits 0 bis 5) und die rechte (Bits 6-11) Hälfte des Akkumulators vertauschen.

IAC (Increment accumulator, 7001)
Akkumulator um eins erhöhen.

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 9 des Befehlswortes null ist (*Oder-Typ*). Ist Bit 9 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 ohne eine Bedingung auszuwählen, die Bedeutung mit Bit 9 invertiert, wird immer gesprungen (*Skip always*). Ist Bit 11 gesetzt wird der Prozessor angehalten. Bit 5 ist ungenutzt.

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

HLT (Halt the program, 7401)
Das Programm anhalten.

SKP (Skip always, 7404)
Den Programmzähler erhöhen. (ohne Bedingung)

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.

SZL (Skip on zero Link, 7414, Und-Typ)
Wenn das Verbindungsbit null ist, den Programmzähler erhöhen.

SNA (Skip on non-zero accumulator, 7424, Und-Typ)
Wenn der Akkumulator nicht null ist, den Programmzähler erhöhen.

SPA (Skip on plus accumulator, 7444, 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 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. 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, 7460)
Den Programmzähler erhöhen wenn der Eingabekanal bereit ist.

SRO (Skip on ready for output, 7462)
Den Programmzähler erhöhen wenn der Ausgabekanal bereit ist.

DGA (Device get word to accumulator, 7564)
Den Akkumulator mit dem aktuellen Wert des Eingabekanals laden.

DPA (Device put word from accumulator, 7566)
Den Wert des Akkumulators in den Ausgabekanal schreiben.

DUS (Device unit select, 7670)
Untereinheit gemäß dem Wert im Akkumulator auswählen.

DGS (Device get status word, 7672)
Statuswort des Geräts bzw. der Untereinheit im Akkumulator speichern.

DSM (Device sense mask, 7774)
Den Wert des Akkumulators als Meldungsauswahlmaske an das Gerät übergeben.

RSD (Reset device, 7776)
Das Gerät zurücksetzen und Untereinheit null auswählen.

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** und **DPA**.

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**.

X/Y-Point Plotter (Scope), Gerät 2

Diese 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.

SIEHE AUCH

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