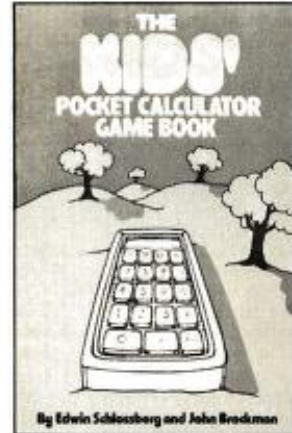
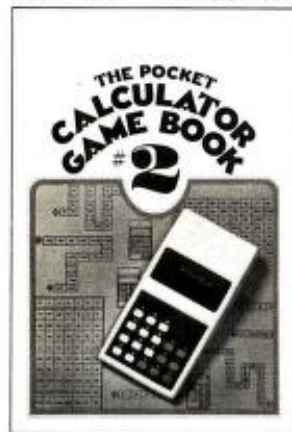


IT ALL ADDS UP TO EDUCATIONAL FUN

The creators of the original Pocket Calculator GameBook now present two fun-filled new game books for use with this incredible machine that has found a place in almost every home.



THE KIDS' POCKET CALCULATOR GAME BOOK
by Edwin Schlossberg and John Brockman
A quick trip through elementary mathematics—fun and games with real purpose. The first book of its kind for kids from kindergarten through college. Illustrated with line drawings and cartoons.
\$6.95 hardcover \$3.95 paperback



THE POCKET CALCULATOR GAME BOOK #2
by Edwin Schlossberg and John Brockman
Even more popular in approach than its famous predecessor, this book is simpler, more accessible, and its games are more mathematically basic. Illustrated with line drawings and cartoons.
\$6.95 hardcover \$3.95 paperback

WILLIAM MORROW

POP @Rn (Pop indirect)

The low order ACC byte is loaded from the memory location whose address resides in Rn after Rn is decremented by 1 and the high order ACC byte is cleared. Branch conditions reflect the final 2 byte ACC contents which will always be positive and never minus 1. The carry is cleared. Because Rn is decremented prior to loading the ACC, single byte stacks may be implemented with the ST @Rn and POP @Rn operations (Rn is the stack pointer).

Example:

15 34 A0	SET	R5, A034	Init stack pointer.
10 04 00	SET	R0, 4	Load 4 into ACC.
35	ST	@R5	Push 4 onto stack.
10 05 00	SET	R0, 5	Load 5 into ACC.
35	ST	@R5	Push 5 onto stack.
10 06 00	SET	R0, 6	Load 6 into ACC.
35	ST	@R5	Push 6 onto stack.
85	POP	@R5	Pop 6 off stack into ACC.
85	POP	@R5	Pop 5 off stack.
85	POP	@R5	Pop 4 off stack.

STP @Rn (Store pop indirect)

The low order ACC byte is stored into the memory location whose address resides in Rn after Rn is decremented by 1. Then the high order ACC byte is stored into the memory location whose address resides in Rn after Rn is again decremented by 1. Branch conditions will reflect the 2 byte ACC contents which are not modified. STP @Rn and PLA @Rn are used together to move data blocks beginning at the greatest address and working down. Additionally, single byte stacks may be implemented with the STP @Rn and LDA @Rn ops.

Example:

14 34 A0	SET	R4, A034	Init pointers.
15 22 90	SET	R5, 9022	
84	POP	@R4	Move byte from A033 to R021.
84	POP	@R4	Move byte from A032 to R020.
95	STP	@R5	
95	STP	@R5	

ADD Rn (Add)

The contents of Rn are added to the contents of the ACC (R0) and the low order 16 bits of the sum are stored in ACC. The 17th sum bit becomes the carry and other branch conditions reflect the final ACC contents.

Example:

10 34 76	SET	R0, 7634	Init R0 (ACC) and R1.
11 27 42	SET	R1, 4227	
A1	ADD	R1	Add R1 (sum = 885B, carry clear)
A0	ADD	R0	Double ACC (R0) to 7086 with carry set.

SUB Rn (Subtract)

The contents of Rn are subtracted from the ACC contents by performing a two's complement addition:

$ACC = ACC + \overline{Rn} + 1$

The low order 16 bits of the subtraction are stored in the ACC. The 17th sum bit becomes the carry and other branch conditions reflect the final ACC contents. If the 16 bit unsigned ACC contents are greater than or equal to the 16 bit unsigned Rn contents then the carry is set, otherwise it is cleared. Rn is not disturbed.

Example:

10 34 76	SET	R0, 7634	Init R0 (ACC) and R1.
11 27 42	SET	R1, 4227	
A1	SUB	R1	Subtract R1 (diff = 340D with carry set)
A0	SUB	R0	Clears ACC (R0)

POPD @Rn (IPOP Double byte indirect)

Rn is decremented by 1 and the high order ACC byte is loaded from the memory location whose address now resides in Rn. Then Rn is again decremented by 1 and the low order ACC byte is loaded from the corresponding memory location. Branch conditions reflect the final ACC contents. The carry is cleared. Because Rn is decremented prior to loading each of the ACC halves, double byte stacks may be implemented with the STD @Rn and POPD @Rn operations. (Rn is the stack pointer).

Example:

15 34 A0	SET	R5, A034	Init stack pointer.
10 12 AA	SET	R0, AA12	Load AA12 into ACC.
75	STD	@R5	Push AA12 onto stack.
10 34 BB	SET	R0, BB34	Load BB34 into ACC.
75	STD	@R5	Push BB34 onto stack.
10 56 CC	SET	R0, CC56	Load CC56 into ACC.
75	STD	@R5	
C5	POPD	@R5	Pop CC56 off stack.
C5	POPD	@R5	Pop BB34 off stack.
C5	POPD	@R5	Pop AA12 off stack.

CPR Rn (Compare)

The ACC (R0) contents are compared to Rn by performing the 16 bit binary subtraction ACC-Rn and storing the low order 16 difference bits in R13 for subsequent branch tests. If the 16 bit unsigned ACC contents are greater than or equal to the 16 bit unsigned Rn contents then the carry is set, otherwise it is cleared. No other registers, including ACC and Rn, are disturbed.

Example:

15 34 A0	SET	R5, A034	Pointer to memory.
16 BF A0	SET	R6, A0BF	Limit address.
10 00 00	LOOP	SET R0, 0	Zero data.
75	STD	@R5	Clear 2 locs, incr R5 by 2.
25	LD	R5	Compare pointer R5 to limit R6.
D6	CPR	R6	
02 F8	BNC	LOOP	Loop if carry clear.

INR Rn (Increment)

The contents of Rn are incremented by 1. The carry is cleared and other branch conditions reflect the incremented value.

Example:

15 34 A0	SET	R5, A034	Init R5 (pointer)
10 00 00	SET	R0, 0	Zero to R0.
55	ST	@R5	Clears loc A034 and incrs R5 to A035.
E5	INR	R5	Inc R5 to A036.
55	ST	@R5	Clears loc A036 (not A035).

DCR Rn (Decrement)

The contents of Rn are decremented by 1. The carry is cleared and other branch conditions reflect the decremented value.

Example: (Clear nine bytes beginning at loc A034)

15 34 A0	SET	R5, A034	Init pointer.
14 09 00	SET	R4, 9	Init count.
10 00 00	SET	R0, 0	Zero ACC.
55	LOOP	ST @R5	Clear a mem byte.
F4	DCR	R4	Decr count.
07 FC	BNZ	LOOP	Loop until zero.

SWEET16 Nonregister Instructions

RTN (Return to 6502 mode)

Control is returned to the 6502 and program execution continues at the location immediately following the RTN instruction. The 6502 registers and status conditions are restored to their original contents (prior entering SWEET16 mode).



Shopping for a computer at the ByteShop is almost as much fun as building one.

Computers are fun. And affordable. Thousands of people are already using personal computers for TV games, video color graphics, digital music and lots of things nobody ever dreamed of—till now.

Until we came along the toughest part about getting started with computers was shopping for one. Now you can visit a ByteShop and put your hands on a wide variety of personal, hobby and business computers.

- | | |
|---|--|
| <p>Arizona
Phoenix—East
813 N. Scottsdale Rd.
Phoenix, Ariz.
12654 N. 28th Drive
Tucson
2012 J. Broadway
California
Berkeley
1514 University Ave.
Berkeley
1812 W. Barbican Blvd.
Cupertino
2530 Elmer Ave.
Berkeley
2909 N. Main St.
Fremont
119 Oak Street
Folsom
3139 L. McKinley Ave.
Hayward
1122 "B" Street
Los Angeles
9001 W. Olympic Blvd.
Lynwood
16508 Hawthorne Blvd.
Long Beach
5431 E. Swann St.
Mojave Del Rio
4658 E
Admiralty Way
Mountain View
1062 W. El Camino Real
Palo Alto
2231 El Camino Real
Pomona
496 W. Lake Ave.
Pleasanton
123 E. Yuba Linda
Sacramento
6041 Greenback Lane
San Diego
8250 Wilshire
San Fernando Valley
18424 Ventura Blvd.
San Francisco
321 Pacific Ave.
Santa Barbara
4 West Mission
Stockton
7910 N. 9th Street
Thousand Oaks
2107 Thousand Oaks Blvd.
Van Nuys
1555 Alhambra Ave.
Westminster
14300 Beach Blvd.
Colorado
Arapahoe County
3464 S. Arroyo St.</p> | <p>Bohler
2040 30th St.
Florida
Gainesville
1325 N. Atlantic Ave., Suite 4
11. Lakeland
1044 E. Oakland Park
Miami
2025 Bird Road
Indiana
Indianapolis North
5947 E. 82nd St.
Kansas
Mission
5815 Johnson Drive
Minnesota
Tucson
1434 Yankee Doodle Rd.
Montana
Billings
1201 Grand Ave., Suite 3
New York
Circleville
2721 Hempstead Turnpike
Rocky River
264 Park Avenue
Ohio
Rocky River
19524 Center Ridge Rd.
Oregon
Beaverton
3482 SW Cedar Hills Blvd.
Portland
2033 SW 4th
Pennsylvania
Bryn Mawr
1065 W. Lancaster Ave.
North Carolina
Raleigh
1213 Hillsborough Street
South Carolina
Columbia
2018 Green St.
Utah
Salt Lake City
161 S. State St.
Washington
Bellevue
14701 NE 20th Ave.
Canada
Vancouver
2151 Banard St.
Winnipeg
665 Century St.
Japan
Tokyo
Tama Bldg., 1-5-9
Saitama</p> |
|---|--|

BYTE SHOP
the affordable computer store