

# THE COSMAC

by Brian Kapitan

The COSMAC is gaining fans in the hobby market. Here the author describes its salient features. Read on and evaluate them for yourself.

The COSMAC microprocessor is an 8-bit register-oriented central-processing unit designed for general purpose computing or for use as part of a control system. It features:

- Static silicon-gate CMOS circuitry
- Instruction fetch-execute time of 2.5 to 3.75 microseconds at 10V
- Single voltage supply
- No minimum clock frequency
- Low power
- TTL compatible
- Any combination of RAM and ROM
- Memory addressing up to 64 K bytes
- Programmed I/O mode
- On-chip DMA
- Four I/O flag inputs directly tested by branch instructions
- Programmable output port
- 91 instructions

## REGISTERS

This microprocessor features sixteen 16-bit registers. Individual registers are selected by a 4-bit binary code from one of the three 4-bit registers (N, P and X). The contents of the registers can be directed:

- 1) to external memory
- 2) to the D register (either high or low bytes)
- 3) to the increment/decrement circuit

COSMAC instructions usually consist of two 8-clock pulse machine cycles. The first is the fetch cycle and the second is an execute cycle. During a fetch, the 4-bits in the P designator select one of the 16 registers as the program counter. When the instruction is read out of memory, the high order bits are loaded into the I register and the lower 4 bits into the N register. Then, the program counter is incremented by one, causing R(P) to be pointing to the next byte in memory.

The X designator selects one of the 16 registers to find in that register an operand or section of data to be used in an ALU or input-output operation.

The N designator can perform many different operations. One of these is to designate one of the 16 registers to be acted upon during a register operation. It may also designate a command-code or device-selection code for peripherals. It is also used to indicate a specific operation. Another, and the most used of this register, is to load a value into the P or into the X designator.

## PROGRAM COUNTERS

The purpose of the P designator is to indicate the program counter. The 4-bit binary code of the register whose purpose is to be the program counter is held in the P register. Other register can be loaded with a specific address in a program, and then the P designator can be changed to that register there by causing a call to a subroutine. When interrupts are being used, the R1 register is the program counter.

## DATA POINTERS

The R registers may also be used as data pointers. That is that they may point to a location in memory. The

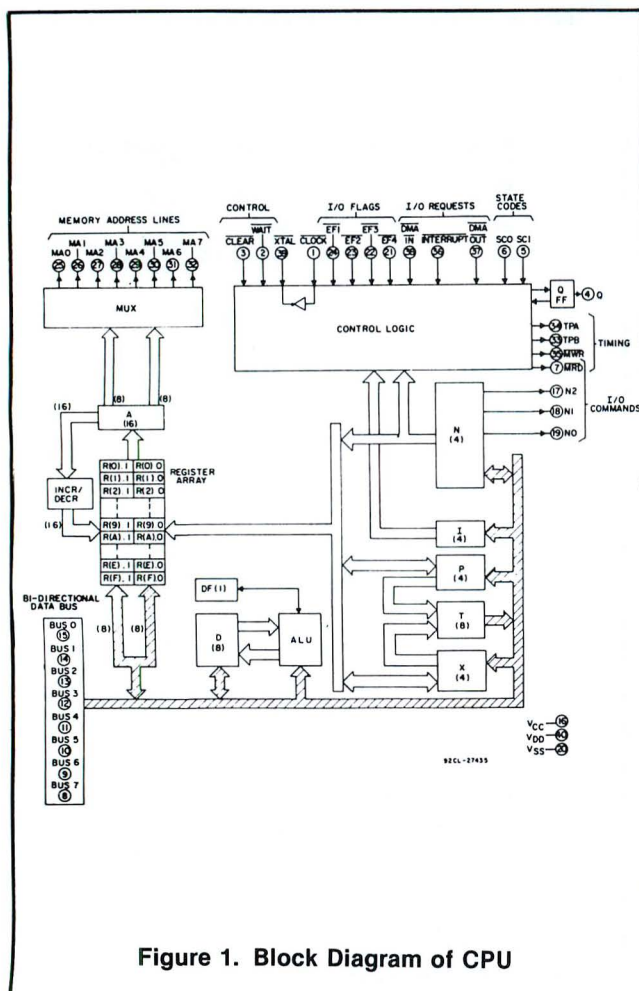


Figure 1. Block Diagram of CPU



# MICROPROCESSOR

X designator points to a register for an ALU operation, for input and output instructions and for other control and miscellaneous operations. The N register is also used as a data pointer. It is used in many memory loading operations into the D register.

The last use of the R register as a data pointer is in the DMA function. Register R0 is always used as the data pointer for memory in a DMA operation. Data is written in or read out using the register as the pointer. The best thing about the DMA-input is that the user can load programs into memory directly without the need of a bootstrap loader.

## DATA REGISTERS

Another purpose of the R register is the storing of data. These data may be read into the D register. Since the D register is only eight bits and the R registers are 16 bits, only parts of the R registers can be loaded into the D register. The R register is divided into two parts: high and low bits. The high order bits of any register is R.1 and the low order is R.0. Also, the register may be used as loop counters through the use of the increment and/or decrement operations.

Register	Number of bits maximum	Purpose
D	8	data register—accumulator
DF	1	flag for ALU carry/borrow
R	16	scratchpad registers
P	4	designates program counter
X	4	designates data pointer
N	4	holds low order instruction bit
I	4	holds high order instruction bit
T	8	holds old X,P after interrupt
IE	1	interrupt enable
Q	1	output flip-flop

## THE Q FLIP-FLOP

The Q flip-flop is an internal flip-flop that can be set or reset and can be sensed in a branch instruction and can also be used for output.

## MICROPROCESSOR INSTRUCTIONS

$M(R(X)) + D : DF, D$

means that the memory byte pointed to by R(X) is added to D and the result is placed in D. If the total of the two is greater than FF (hex 255) then DF (data flag) is set for 1. Otherwise it is kept at zero.

OP	MNEMONIC	OPERATION
00	IDL	idle; wait for DMA or interrupt, $M(R(O)) : BUS$
C4	NOP	no operation; continue
DN	SEP	set P; let P equal N
EN	SEX	set X; let X equal N
7B	SEQ	1 : Q
7A	REQ	0 : Q
78	SAV	save temporary storage T : $M(R(X))$
79	MARK	push X,P to stack (X,P) : T, (X,P) : $M(R(2))$ , P:X R(2)-1
70	RET	return $M(R(X)) : (X,P)$ , $R(X) + 1$ , 1:IE
71	DIS	disable interrupt same as above but: 0:IE
0N	LDN	load via N $M(R(N)):D$ N cannot equal 0
4N	LDA	load and advance $M(R(N)):D$ R(N) + 1
F0	LDX	load via X $M(R(X)):D$

72	LDXA	load via X and advance $M(R(X)):D$ R(X) + 1
F8	LDI	load immediate $M(R(P)):D$ R(P) + 1
5N	STR	store via N $D:M(R(N))$
73	STXD	store via X and decrement $D:M(R(X))$ R(X)-1
1N	INC	increment register R(N) + 1
2N	DEC	decrement register R(N)-1
60	IRX	increment register R(X) + 1
8N	GLO	move low N register to D $R(N).0:D$
AN	PLO	reverse of above $D:R(N).0$
9N	GHI	move high N register to D $R(N).1:D$
BN	PHI	reverse of above $D:R(N).1$
F1	OR	$M(R(X))$ or D:D
F9	ORI	or immediate $M(R(P))$ or D:D $R(P) + 1$
F3	XOR	exclusive or $M(R(X))$ xor D:D
FB	XRI	exclusive or immediate $M(R(P))$ xor D:D $R(P) + 1$
F2	AND	$M(R(X))$ and D:D
FA	ANI	and immediate $M(R(P))$ and D:D $R(P) + 1$
F6	SHR	shift right D ; least significant bit D:DF msb set to zero
76	SHRC	shift right D; $Isb(D):DF$ ; $DF:msb(D)$
	RSHR	
FE	SHL	shift left D msb(D):DF $Isb$ D set to zero
7E	SHLC	shift left D msb(D):DF $DF:Isb(D)$
	RSHL	
F4	ADD	add; $M(R(X)) + D:DF,D$
FC	ADI	add immediate; $M(R(P)) + D:DF,D$ $R(P) + 1$
74	ADC	add with carry; $M(R(X)) + D + DF:DF,D$
7C	ADCI	add with carry, immediate; $M(R(P)) + D + DF:DF,D$ $R(P) + 1$
F5	SD	subtract D; $M(R(X))-D:DF,D$
FD	SDI	subtract immediate; $M(R(P))-D:DF,D$ $R(P) + 1$
75	SDB	subtract with borrow; $M(R(X))-D-(not DF):DF,D$
7D	SDBI	subtract with borrow, immediate $M(R(P))-D-(not DF):DF,D$ $R(P) + 1$
F7	SM	subtract memory; $D-M(R(X)):DF,D$
FF	SMI	subtract memory, immediate; $D-M(R(P)):DF,D$ $R(P) + 1$
77	SMB	subtract memory with borrow; $D-M(R(X))-(not DF):DF,D$
7F	SMBI	subtract memory with borrow, immediate $D-M(R(P))-(not DF):DF,D$ $R(P) + 1$
30	BR	short branch $M(R(P)):R(P).0$
38	NBR	no short branch $R(P) + 1$
32	BZ	short branch if D = zero $M(R(P)):R(P).0$ else $R(P) + 1$
3A	BNZ	branch on no zero $M(R(P)):R(P).0$ else $R(P) + 1$
33	BDF	if $DF = 1$ $M(R(P)):R(P).0$ else $R(P) + 1$
	BPZ	if positive or zero (same as above)
	BGE	if greater or equal (same as above)
3B	BNF	if $DF = 0$ $M(R(P)):R(P).0$ else $R(P) + 1$
	BM	if minus (same as above)
	BL	if less than (same as above)
31	BQ	if $Q = 1$ $M(R(P)):R(P).0$ else $R(P) + 1$
39	BNQ	if $Q = 0$ $M(R(P)):R(P).0$ else $R(P) + 1$
34	B1	if $EF1 = 1$ do process $M(R(P)):R(P).0$ else $R(P) + 1$
3C	BN1	if $EF1 = 0$ do above process
35	B2	if $EF2 = 1$ do above process
3D	BN2	if $EF2 = 0$ do above process
36	B3	if $EF3 = 1$ do above process
3E	BN3	if $EF3 = 0$ do above process
37	B4	if $EF4 = 1$ do above process



3F BN4 if EF4 = 0 do above process

Note: the following are along branch instructions. See notes below for explanation.

C0 LBR long branch  $M(R(P)):R(P).1, M(R(P) + 1):R(P).0$   
 C8 NLBR no long branch  $R(P) + 2$   
 C2 LBZ branch if  $D = 0$   $M(R(P)):R(P).1, M(R(P) + 1):R(P).0$   
 else  $R(P) + 2$   
 CA LBNZ branch if D not zero (process same as above)  
 C3 LBDF branch if  $DF = 1$  (process same as above)  
 CB LBNF branch if  $DF = 0$  (process same as above)  
 C1 LBQ branch if  $Q = 1$  (process same as above)  
 C9 LBNQ branch if  $Q = 0$  (process same as above)  
 38 SKP short skip  $R(P) + 1$   
 C8 LSKP long skip  $R(P) + 2$   
 CE LSZ long skip if  $D = 0$   $R(P) + 2$  else continue  
 C6 LSNZ long skip if D not zero  $R(P) + 2$  else continue  
 CF LSDF long skip if  $DF = 1$   $R(P) + 2$  else continue  
 C7 LSNF long skip if  $DF = 0$   $R(P) + 2$  else continue  
 CD LSQ long skip if  $Q = 1$   $R(P) + 2$  else continue  
 C5 LSNQ long skip if  $Q = 0$   $R(P) + 2$  else continue  
 CC LSIE long skip if  $IE = 1$   $R(P) + 2$  else continue

#### INPUT-OUTPUT

6N OUT where  $N = 1$  to  $7$   $M(R(X)):BUS$   $R(X) + 1$   
 6N IN where  $N = 9$  to  $F$   $BUS:M(R(X)), BUS:D$

#### NOTES ON THE INSTRUCTION SET

1. the N in the op code stands for the digit in the N register.
2. the : in the operation means moved to.
3. in the input and output, the N is for the device address line.
4. the branch instructions are not for subroutines. They are goto's.
5. the short branch is used to branch to another memory location in the same page (256-byte) of memory. The long branch involves three bytes. The first is the instruction. The second and third are the branching address. In a short branch, there is only two bytes—one for the instruction itself, and the other for the current page branching address.
6. The skip instructions are used to 'skip' the next or next two instructions. Take for example instruction CE. This will skip the next two

instructions if  $D = 0$ . If D is some other number than zero, it will continue.

7. In the instruction set, you may have noticed one op code with more than one mnemonic. The reason for this is it depends which instruction is before it. The same action will occur, but it is worded different to make its action clear.

#### SIGNALS IN THE 1802

BUS 0 to BUS 7 8-bit directional DATA BUS lines. These lines are used for transfer of data between the microprocessor, the memory and the I/O devices.  
 N0 to N2 Issued by an I/O instruction for the I/O control logic of data transfer. These lines can be used to issue commands or device selection codes to the I/O devices. The N bits are low except when an I/O instruction is being executed. During this time, their state equals that of their corresponding bits in the N register. The direction of data flow is defined by the N3 bit, and indicated by the level of the MRD signal.  
 EF1 to EF4 These are flag lines that can be tested in the program. These flag lines can have a variety of uses such as input and output for sensing of a certain condition or counting a certain number of objects.  
 INTERRUPT Interrupt: X,P stored in T. X is set to 2 and P is set for 1 and IE is set to zero.  
 DMA-IN/OUT Dma in/out: R(0) points to memory location for input or output of memory. After input or output of data, the R(0) is incremented.  
 SC0, SC1 These lines indicate the action of the CPU.  

process	SC1	SC0
fetch	low	low
execute	low	high
DMA	high	low
interrupt	high	high

 TPA,TPB Timing pulses that occur once each machine cycle  
 MA0-MA7 these are the 8 memory address lines. The high bits appear on the line and are put into the external address latches by TPA. Then, the low order bits are placed on the lines after the TPA is completed.  
 MWR This is the memory write pulse that appears after the address lines have been stabilized.  
 MRD This is the memory read level. It can be used to control 3-state outputs from the addressed memory which may have a common data input output bus.  
 Q This is a single bit output line from the CPU which can be set or reset under program control.  
 CLOCK This is the input line for externally generated single-phase clock. The clock is counted down at the rate of 8 pulses/machine cycle.  
 XTAL purpose is to provide for external crystal for timing.  
 WAIT CLEAR Provide control modes as follows:  

CLEAR	WAIT	MODE
low	low	load
low	high	reset
high	low	pause
high	high	run

 Load: holds the CPU in idle and allows for an I/O device to load the memory without the need of a bootstrap loader.  
 Reset: I,N,Q are reset, IE set, and zeros are placed on the data bus.  
 Pause: stops the internal CPU timing generator on the first negative high to low transition of the input clock.  
 Run: starts a fetch from 0000 in memory.

#### OTHER COMMANDS

!Maaaa xx change memory at aaaa to xx  
 ?Maaaa hhhh list memory at aaaa for hhhh bytes  
 \$Paaaa begin program execution at aaaa with p = zero

#### MONITOR BOARD COMMANDS

!Rn hhhh set Rn to hhhh  
 !Xn set X to n  
 !Pn set P to n  
 !Dhh set D to hh  
 !Fb set D flag to b = 0 or 1  
 !BPaaaa set a breakpoint at aaaa  
 !BR remove the breakpoint  
 ?R display the registers  
 ?X dis X  
 ?P dis P  
 ?D dis D  
 ?F dis F  
 \$P resume program execution  
 \$Nhhhh execute the next hhhh instructions  
 aaaa = address xx is a hex digit pair n is a reg number  
 h is a hex digit

# BITS N BYTES

SPECIALTIES:

Business Packages

Industrial Systems

Entrepreneur Systems

Hardware Designs

Service

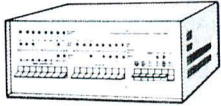
We Stock Most Major Micro Manufacturers

\*\*\* Hobbyists Welcome \*\*\*

Our Representative In San Diego!

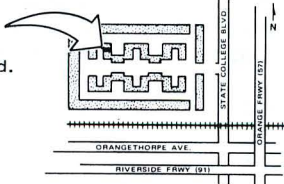
Jim Farthing

(714) 421-1041



**BITS N BYTES**  
 College Business Park  
 679 "D" S. State College Blvd.  
 Fullerton, Calif. 92631  
 (714) 879-8386

**HOURS:** 12-7 P.M. M-F  
 12-5 P.M. Sat.



CIRCLE INQUIRY NO. 62