Son of Motorola (or, the $20 CPU Chip)

Would you believe – another microprocessor? You bet. The calculator firm, MOS Technology of Norristown, Pennsylvania, has just recently announced a new microprocessor which combines plug in compatibility with the Motorola 6800 and a new instruction set to come out with yet another option for microprocessor users – but at a price of $20 in single quantities. Here comes the under $200 processor kit? Not quite yet, but maybe within a year or two. (It's already to the point where the sheet metal and transformer iron of a home computer often cost more than all the silicon products which make it work . . . this new low on CPU prices just compounds the problem.) It may be three to six months before you see one of these new MCS6501 processors designed into a kit, so Dan Fylstra in his article covers quite a few details of the Motorola 6800 by way of comparison with "Son of Motorola."

by
Daniel Fylstra
Associate Editor, BYTE
25 Hancock St.
Somerville MA 02144

We thought that the "age of the affordable computer" had arrived when you could buy a microprocessor chip for $150. But the potent combination of new technology and free enterprise has brought about developments beyond our wildest expectations.

So now you can buy your microprocessor brand new, in single quantities, for $20. The new offering is from MOS Technology, Inc., and is pin-compatible, but software-incompatible with the Motorola 6800 microprocessor. Although it will be a while before the new chip finds its way into ready-to-build kits for the hobbyist (after all, the first Motorola 6800 kits have just been announced), the news should be of interest to nearly every home brew computer experimenter. So here's a comparison of the Motorola 6800 and the MOS Technology 6500 series, based on the information presently available. If you aren't already familiar with the Motorola microprocessor, don't worry – we'll cover its major features in the course of the comparison.

Hardware Comparison

Both the Motorola 6800 and the MOS Technology chip are TTL-compatible devices, operating from a single five volt power supply. Like earlier microcomputers, such as the Intel 8008, 8080 and National PACE, these processors make use of a bidirectional data bus, to which both memory and input/output devices may be connected. However there are no special input/output instructions in the instruction repertoire of either the Motorola or MOS Technology microprocessors. Output of a character, for example, is accomplished by storing a value into a certain memory location, which is in reality a special register inside an external I/O interface chip, connected to the data bus just like any other RAM or ROM chip.

Motorola supplies a Peripheral Interface Adapter (PIA) chip which connects to the data bus for 8-bit parallel I/O, and an Asynchronous Communications Interface Adapter (ACIA) for bit-serial input/output. (The ACIA is simply a type of UART, as discussed in Don Lancaster's September article on serial interfaces. It may be used to connect a teletype or CRT terminal to the microcomputer system.) MOS Technology plans to supply a similar set of chips.

Most of the time, data is being transmitted between the microprocessor and the memory chips over the data bus. But the processor can also disconnect itself from the bus, enabling, for example, a data transfer to take place directly between an I/O device and memory. Both the Motorola 6800 and the MOS Technology chip have three-state buffers for the eight data lines, enabling them to disconnect from the bus in this fashion. But the Motorola also has three-state buffers on its 16 address lines, whereas the MOS Technology chips do not.

This would be used, for example, in a floppy disk controller which is capable of transferring a whole block of many bytes of data in response to a single command from the CPU. The controller would present a series of addresses on the 16 address lines, and data bytes on the data lines, causing the bytes to be stored in a series of locations in some RAM chip on the bus; all this would take place in the intervals when the CPU itself was disconnected from the bus.

As a practical matter, however, small systems do not require this kind of direct memory access (DMA) capability, and larger systems with more devices on the bus will require buffers on the

Ready or not, here I come: 6800 to 6501.

address lines to supply the necessary power – and these buffers may as well have three-state outputs.

The other major hardware difference between the Motorola 6800 and the MOS Technology 6500 series is that the MOS Technology chip has an 8080-style Ready line, whereas the Motorola 6800 does not. The Ready line is used to make the microprocessor wait for a variable length of time before going on with the execution of an instruction. This feature makes it easy to use the less expensive memory chips, especially for Programmable or Erasable Read-Only Memory (PROM or EROM) which are not as fast as the CPU itself. It is possible to use such devices with the Motorola 6800, of course, by stretching out one of the clock phases to as long as five microseconds. But the availability of the Ready line on the MOS Technology chip is certainly a convenience, and allows you to use extremely slow memories if you wish.

The MCS6501, first in the MOS Technology 6500 series, requires the same type of external clock as the Motorola 6800. But for $25 you can have the MCS6502, which includes an on-the-chip clock, driven by an external single phase clock or an RC or crystal time base input. As the manufacturer suggests, it is probably cheaper in an original design to use the MCS6502 than to provide the external logic to generate the two-phase clock.

To sum up, both the Motorola 6800 and the MOS Technology have comparable features with some differences. In terms of hardware differences, the MOS Technology Ready line is probably more valuable than the three-state address line buffers available on the Motorola 6800.

A final hardware advantage possessed by the MOS Technology chip is speed. The Motorola 6800 cycle time is one microsecond (1 MHz clock rate), and a typical instruction takes about three clock cycles. While the cycle time of the MOS Technology chip is nominally the same, the company has hinted broadly that the chip can be run at clock rates of 2 or even 3 MHz. Of course, one would have to use faster and more expensive memory chips to take advantage of this increased speed.

In addition, certain critical instructions take fewer cycles on the MOS Technology chip. An STA (store accumulator) instruction referencing an arbitrary location takes 4 cycles, versus 5 for the Motorola, and a JSR (jump to subroutine) instruction requires 6 cycles, as opposed to 9 on the 6800. Conditional branches take 4 cycles on the Motorola microprocessor, while they take 2 cycles if the condition is false and 3 if it is true on the MOS Technology chip. Because these instructions are so frequently executed in most programs, the 6500 series should enjoy a performance edge over the Motorola 6800 even at the same clock rate.

Table I. Functionally equivalent instructions for both the Motorola 6800 and MOS Technology MCS6501 microprocessors. The mnemonics are Motorola's. Of course, these instructions operate on the A accumulator only in the MCS6501, but can address either accumulator in the Motorola 6800. The BIT instruction (*) has a different effect on the V and N processor flags in the MCS6501.

ADC	DEX
AND	EOR
ASL	INC
ASR	INX
BCC	JMP
BCS	JSR
BEQ	LDA
BIT*	LDX
BMI	LSR
BNE	NOP
BPL	ORA
BVC	PSH
BVS	PUL
CLC	ROL
CLI	RTI
CLV	RTS
CMP	SBC
CPX	SEC
DEC	SEI
	STA
	STX
	TSX
	TXS

We thought that the "age of the affordable computer" had arrived when you could buy a microprocessor chip for $150. But the potent combination of new technology and free enterprise has brought about developments beyond our wildest expectations.

Software Comparison

We can treat the instruction set architecture of the two processors in two stages, first considering the facilities for manipulating *data* and then dealing with the facilities for manipulating *addresses*. Both features are important to the overall effectiveness of the processor design.

Data Manipulation

The instructions for manipulating data are quite similar on the two processors. There are two major differences: First, the Motorola 6800 has two 8-bit accumulators, A and B, while the MOS Technology chip has only one accumulator, A. Second, in addition to conditional branches for unsigned comparisons, the Motorola 6800 has special branch instructions for signed comparisons, but the MOS Technology chip does not. (The signed comparisons treat the two values as positive or negative numbers in two's complement notation, in the range –128 to +127. For example, –1 is represented as 2^8-1 = 11111111. An unsigned comparison would treat this quantity as the largest possible (8-bit) value, whereas a signed comparison would treat it as smaller than, say, zero.)

Table I lists the instructions which are the same for both processors, while Table II lists instructions on the Motorola 6800 which must be replaced by more than one instruction on the 6500 series microprocessors.

Some of the instructions omitted on the MOS Technology chip are merely incidental; others are more serious. The lack of signed comparisons represents a real inconvenience in many applications. The lack of a simple ADD instruction means that an operation such as A = B + C on one-byte operands must be coded with a "Clear Carry" (CLC) as in this example:

```
CLC
LDA  B
ADC  C
STA  A
```

on the MOS Technology chip. On the other hand, a computation such as A = B + C – D could be coded as

```
CLC
LDA  B
ADC  C
SBC  D
STA  A
```

assuming that the inclusion of "carry" in both operations is indeed desired.

Less serious but still irritating are the absence of the ROR (rotate right), NEG (negate) and COM

Table II. Motorola 6800 instructions which have no direct equivalent in the MCS6501. The information in this table is taken from MOS Technology documentation on the 6500 series.

Motorola 6800 Instruction	Equivalent 6500 Series Sequence
ABA	No B accumulator
ADD	CLC, ADC
BGE loc	BMI *+6, BVC loc, BVS *+4, BVS loc
BGT loc	BMI *+6, BVC *+6, BVS *+6, BVC *+4, BNE loc
BHI loc	BCS *+4, BNE loc
BLE loc	BEQ loc, BMI *+6, BVS loc, BVC *+4, BVC loc
BLS loc	BCS loc, BEQ loc
BLT loc	BMI *+6, BVS loc, BVC *+4, BVC loc
BRA	JMP
BSR	JSR
CBA	No B accumulator
CLR [loc]	LDA #0, [STA loc]
COM [loc]	[LDA loc], EOR #$FF, [STA loc]
DAA	Replaced by SED
DES	Use PHA
INS	Use PLA
LDS loc	LDX loc, TXS
NEG [loc]	EOR #$FF, ADC #1 [or LDA #0, SBC loc]
ROR [loc]	[LDA loc], PHP, LSR, PLP, BCC *+4, ORA #$80, [STA loc]
SBA	No B accumulator
SEV	LDA #1, LSR
STS loc	TSX, STX loc
SUB	CLC, SBC
SWI	BRK saves state without transferring control
TAB	No B accumulator
TAP	PHA, PLP
TBA	No B accumulator
TPA	PHP, PLA
TST	BIT #0
WAI	JMP *
op disp, X [indexed addressing mode]	LDY #disp, op @loc, Y [indirect indexed addressing mode]

(complement) instructions, as well as single-byte instructions to increment and decrement the accumulator. Probably the least significant difference is the omission of the B accumulator on the MOS Technology chip. This is more than made up for by the availability of an extra index register (see below).

All in all, the Motorola 6800 comes out ahead when considering facilities for manipulating data, the most important point in its favor being the availability of the signed comparisons. Generally speaking, however, the basic instructions available on the two processors are quite similar.

Address Manipulation

The greatest architectural differences between the two processors lie in their facilities for manipulating addresses, or their "addressing modes" – and here the MOS Technology chip has much more to offer.

The two microprocessors are the same in one respect: both have special "short forms" of most instructions for referencing the first 256 bytes of memory. This is called "direct addressing" on the Motorola 6800, and "zero page addressing" on the MOS Technology chip. As an example, the most general LDA (load accumulator) instruction is three bytes long; the second and third bytes form the effective address (0-65535), which can reference any byte in memory. The short form of the LDA instruction, however, is two bytes long; the second byte forms the effective address (0-255) of a byte in the first "page" of memory. The "short form" instructions generally take one fewer clock cycle to execute, since only two rather than three instruction bytes must be fetched from memory.

The major differences between the two processors lie in the important area of indexed addressing. The Motorola 6800 has a single 16-bit index register, called X. Essentially all instructions have an indexed addressing form, in which a one-byte displacement (0-255) is added to the address in the index register to form the effective address. The MOS Technology chip, on the other hand, has two 8-bit index registers, called X and Y. All of the computational instructions have indexed addressing forms in which either a one- or two-byte base address is added to the contents of either the X or the Y register to form the effective address.

Which approach is the better one? For the purpose of accessing elements of arrays, or tables of many identical elements, the MOS Technology chip comes out way ahead. This is partly due to the lack of certain critical instructions on the Motorola 6800, such as an instruction to add the contents of an accumulator to the index register, or even to transfer the value in the accumulators to the index register.

Suppose that we wish to add the Ith element of an array, S_I, to another variable, T. In general, the array may be located anywhere in memory, and the subscript I may be the result of some calculation done in the accumulators. Letting S denote the address of the zeroth element (the base address) of the array, and assuming that the value of the subscript I is already in the A accumulator, consider the instructions necessary to accomplish this operation on the two processors.

The biggest difference is in the area of addressing modes, an area where the 6500 series devices far outshine the Motorola 6800.

On the Motorola 6800, our first try yields the following:

```
SHI   EQU S/256*256
      CLR B          ┐
      ADD A #S-SHI   │ Calculate the
      ADC B #S/256   │   indexed address
      STA A TEMP+1   │
      STA B TEMP     │
      LDX TEMP       ┘
      LDA A 0, X     ┐ Perform desired
      ADD A T        │   computation
      STA A T        ┘
```

This instruction sequence requires 19 bytes, counting the two-byte temporary TEMP and assuming that TEMP and T are located in the first 256 bytes of memory. Since the array S could be anywhere in memory, we were unable to use the displacement field of an instruction with indexed addressing for the array base address, and instead we had to add the array base to the index (in double precision), store the result in memory, load it into the index register, and finally reference the array element S_I.

We can improve on this with the aid of a little lateral thinking. Noticing that the 6800 is actually capable of adding a one-byte quantity to a two-byte address, but only in a storage reference with indexed addressing, we will split up the base address into two parts to arrive at a better solution:

```
SHI    EQU S/256*256
       STA A TEMP+1
       LDX TEMP
       LDA A S-SHI, X
       ADD A T
       STA A T
         .
         .
         .
TEMP   FDB SHI
```

This instruction sequence requires only 12 bytes, under the same assumptions.

Even so, we can't match the simplicity of the solution to the same problem on the MOS Technology chip:

```
TAX
LDA S, X
ADD T
STA T
```

This instruction sequence requires only seven bytes. Only four bytes were needed to reference the element S_I, versus eight for the Motorola 6800.

How important is this improvement? It is certainly significant, since arrays and tables are used so frequently in programs of any size. On the other hand, in many applications it is only necessary to reference each element of an array in turn; it is not necessary to access elements randomly based on a computed subscript. In this case, we can obtain better code on the Motorola 6800 by first loading the array base address into the index register, and then referencing each element directly (i.e., with a zero indexed address displacement), incrementing the address in the index register using the INX instruction to proceed from element to element. We are therefore using the 6800's index register to hold a pointer or indirect address rather than an index.

An even more important difference between the two microprocessors in that the MOS Technology chip possesses *two* (8-bit) index registers, X and Y, whereas the Motorola 6800 has only *one* (16-bit) index register X. As we shall see, two index registers are far more valuable than two accumulators. This is because programs frequently manipulate two (or more) tables, or other indirectly addressed variables, at the same time. As an example, we will consider perhaps the simplest operation of this type, the problem of moving a string of bytes from one area of storage to another. Assume that 20 bytes, starting at the location denoted by the symbol FROM, are to be moved to the area starting at the location denoted by the symbol TO.

Two index registers are far more valuable than two accumulators.

On the Motorola 6800, we can write the following routine:

```
LOOP   LDX FRPTR  ┐ Fetch
       LDA A 0, X ┘ FROM
       LDX TOPTR  ┐ Move
       STA A 0, X ┘ TO
       INC FRPTR  ┐
       INC TOPTR  │ change
       DEC COUNT  ┘ pointers
       BNE LOOP   Test
                  continuation
         .
         .
         .
FRPTR  FDB FROM
TOPTR  FDB TO
COUNT  FCB 20
```

This routine requires 24 bytes, including the working storage locations, and executes in 820 clock cycles. This routine can move up to 256 bytes.

On the MOS Technology chip we have the following solution:

```
       LDX #0
       LDY #0
LOOP   LDA FROM, X
       STA TO, Y
       INX
       INY
       DEC COUNT
       BNE LOOP
         .
         .
         .
COUNT  FCB 20
```

This routine requires 17 bytes, and executes in 404 clock cycles. The improvement in speed clearly depends on the number of bytes to be moved; each pass through the loop in the Motorola 6800 routine takes 41 clock cycles, while each pass through the loop in the MOS Technology routine takes 20 cycles. (The MOS Technology routine is also limited to moving at most 256 bytes.)

Once again the degree of improvement is substantial, and the improvement is

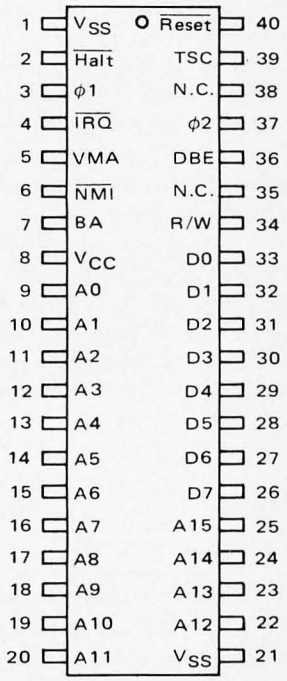

Fig. 1. The pin assignments of the Motorola 6800 (and by implication, the MOS Technology MCS6501). V_{SS} is ground (0 volts) and V_{CC} is +5 volts. The A lines are address outputs, and the D lines are bidirectional tristate data bus lines. For details see the Motorola and MOS Technology documentation of these parts.

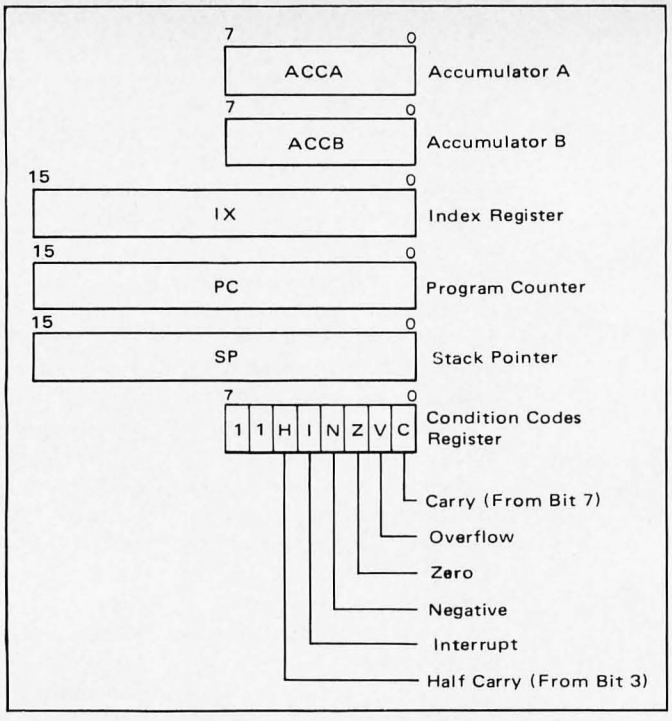

Fig. 2. The programmer's view of the 6800 CPU. This diagram, excerpted from the Motorola 6800 documentation, shows the various registers of the CPU including the processor's condition code register. Note the similarity to the MCS6501 in Fig. 3.

significant because this type of problem arises so frequently in large programs.

The MOS Technology chip has some additional addressing modes not possessed by the Motorola 6800. First, there is a "short form" for instructions with simple indexed addressing if the array base address is in the first "page" (256 locations) of memory. This feature is of somewhat limited use except in very small programs, since only a few small arrays can actually be placed in the first 256 locations. Of greater interest is the so-called "indirect indexed" addressing mode. Instructions with this type of addressing are two bytes long; the second byte specifies the address of a two-byte constant in the first page of memory. This two-byte constant then becomes the "array base address," and the contents of the Y register are added to this constant to form the effective address. This addressing mode is very useful: In a program with many references to a particular array or table which is too large to place in the first page of memory, one can *trade space for time* by placing the array base address in the first page of memory, and then referencing elements of the array using indirect indexed addressing. Each element reference takes less space (two bytes instead of three) but more time (five cycles instead of four) than would be required for ordinary indexed addressing.

There are two other addressing modes on the MOS Technology chip which are somewhat less useful. The first is called "indexed indirect" addressing: Here the contents of the X register are added to a one-byte base address to obtain the address of a two-byte constant in the first page of memory. The contents of this two-byte constant then becomes the effective address. Unfortunately this addressing mode is not available for the JMP instruction, where it would be most useful: It could be used to implement a "jump table," or a "computed GO TO" or "CASE statement" in some high-level languages.

Finally, two other addressing modes are used with branch instructions:

One unfortunate feature of the MOS Technology chip's many addressing modes is that they do not apply consistently to all instructions.

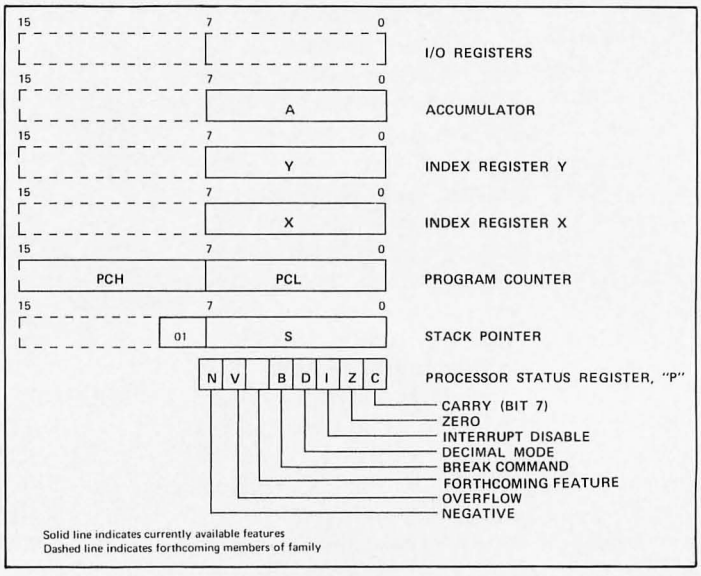

Fig. 3. The programmer's view of the MCS6501 CPU. This diagram, excerpted from the MOS Technology 6500 series preliminary documentation, shows the various registers of the CPU. Note the similarity to the Motorola 6800 diagram in Fig. 2.

Table III. Instructions, addressing modes and execution times for the Motorola 6800 processor. Execution times are in "machine cycles" which for a 1.0 MHz clock take 1.0 microsecond apiece. This table is excerpted from Motorola documentation on their processor.

	(Dual Operand)	ACCX	Immediate	Direct	Extended	Indexed	Implied	Relative
ABA		•	•	•	•	•	2	•
ADC	x	•	2	3	4	5	•	•
ADD	x	•	2	3	4	5	•	•
AND	x	•	2	3	4	5	•	•
ASL		2	•	•	6	7	•	•
ASR		2	•	•	6	7	•	•
BCC		•	•	•	•	•	•	4
BCS		•	•	•	•	•	•	4
BEA		•	•	•	•	•	•	4
BGE		•	•	•	•	•	•	4
BGT		•	•	•	•	•	•	4
BHI		•	•	•	•	•	•	4
BIT	x	•	2	3	4	5	•	•
BLE		•	•	•	•	•	•	4
BLS		•	•	•	•	•	•	4
BLT		•	•	•	•	•	•	4
BMI		•	•	•	•	•	•	4
BNE		•	•	•	•	•	•	4
BPL		•	•	•	•	•	•	4
BRA		•	•	•	•	•	•	4
BSR		•	•	•	•	•	•	8
BVC		•	•	•	•	•	•	4
BVS		•	•	•	•	•	•	4
CBA		•	•	•	•	•	2	•
CLC		•	•	•	•	•	2	•
CLI		•	•	•	•	•	2	•
CLR		2	•	•	6	7	•	•
CLV		•	•	•	•	•	2	•
CMP	x	•	2	3	4	5	•	•
COM		2	•	•	6	7	•	•
CPX		•	3	4	5	6	•	•
DAA		•	•	•	•	•	2	•
DEC		2	•	•	6	7	•	•
DES		•	•	•	•	•	4	•
DEX		•	•	•	•	•	4	•
EOR	x	•	2	3	4	5	•	•

	(Dual Operand)	ACCX	Immediate	Direct	Extended	Indexed	Implied
INC		2	•	•	6	7	•
INS		•	•	•	•	•	4
INX		•	•	•	•	•	4
JMP		•	•	•	3	4	•
JSR		•	•	•	9	8	•
LDA	x	•	2	3	4	5	•
LDS		•	3	4	5	6	•
LDX		•	3	4	5	6	•
LSR		2	•	•	6	7	•
NEG		2	•	•	6	7	•
NOP		•	•	•	•	•	2
ORA	x	•	2	3	4	5	•
PSH		•	•	•	•	•	4
PUL		•	•	•	•	•	4
ROL		2	•	•	6	7	•
ROR		2	•	•	6	7	•
RTI		•	•	•	•	•	10
RTS		•	•	•	•	•	5
SBA		•	•	•	•	•	2
SBC	x	•	2	3	4	5	•
SEC		•	•	•	•	•	2
SEI		•	•	•	•	•	2
SEV		•	•	•	•	•	2
STA	x	•	•	4	5	6	•
STS		•	•	5	6	7	•
STX		•	•	5	6	7	•
SUB	x	•	2	3	4	5	•
SWI		•	•	•	•	•	12
TAB		•	•	•	•	•	2
TAP		•	•	•	•	•	2
TBA		•	•	•	•	•	2
TPA		•	•	•	•	•	2
TST		2	•	•	6	7	•
TSX		•	•	•	•	•	4
TSX		•	•	•	•	•	4
WAI		•	•	•	•	•	9

NOTE: Interrupt time is 12 cycles from the end of the instruction being executed, except following a WAI instruction. Then it is 4 cycles.

Which processor comes out ahead overall? To a great extent it depends on your point of view: Systems programs are better on the MOS Technology machines; applications programs would tend to come out ahead on the Motorola 6800.

"Relative" addressing, available on both the Motorola and the MOS Technology processors, is used with the conditional branch instructions, which are two bytes long. The second byte of such an instruction specifies a positive or negative displacement in two's complement notation (−128 to +127). The destination address of the branch is taken to be the algebraic sum of the address of the byte immediately following the branch instruction and this displacement. Of course, this means that it is possible to branch directly to a location within only a certain limited distance from the branch itself; but, more often than not, the range of −128 to +127 bytes is adequate, and a space savings is realized in comparison to processors such as the Intel 8080 which have only three-byte branch instructions. If necessary, a conditional branch can always transfer to a three-byte unconditional JMP instruction, which can jump to any location in memory. On the MOS Technology chip, a JMP instruction can also specify "absolute indirect" addressing: In this case, the second and third bytes of the instruction specify the address of a two-byte constant anywhere in memory, and the contents of this two-byte constant becomes the destination address for the jump.

One unfortunate feature of the MOS Technology chip's many addressing modes is that they do not apply consistently to all instructions. For example, the binary arithmetic instructions are available with essentially all addressing modes, but the unary arithmetic instructions are missing the Y-register and indirect modes, and the BIT instruction is missing several others as well. This not only makes programming more difficult, since one must constantly check to see which instruction forms are legal, and program around the exceptions; it also makes the design of an assembler or compiler more complicated. A compiler, in particular, would require complex logic to determine when it could and could not take advantage of the addressing modes.

In summary, the MOS Technology chip comes out ahead when considering facilities to manipulate addresses, and in many cases the advantage realized due to the availability of the extra addressing modes is substantial. The greatest failing of the 6500 series design is the inconsistent availability of the addressing modes from instruction to instruction.

Which processor comes out ahead overall? This is very difficult to judge. It depends partly on whether the programs being executed on the microcomputer are "system" programs, such as compilers, interpreters and I/O controllers, which tend to make heavy use of address

Table IV. Instructions, addressing modes and execution times for the MOS Technology MCS6501 processor. Execution times are in "machine cycles" which for a 1.0 MHz clock take 1.0 microsecond apiece. This table is excerpted from MOS Technology documentation on their processor.

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
ADC		2	3	4		4	4*	4*			6	5*	
AND		2	3	4		4	4*	4*			6	5*	
ASL	2		5	6		6	7						
BCC										2**			
BCS										2**			
BEQ										2**			
BIT			3			4							
BMI										2**			
BNE										2**			
BPL										2**			
BRK													
BVC										2**			
BVS										2**			
CLC									2				
CLD									2				
CLI									2				
CLV									2				
CMP		2	3	4		4	4*	4*			6	5*	
CPX		2	3			4							
CPY		2	3			4							
DEC			5	6		6	7						
DEX									2				
DEY									2				
EØR		2	3	4		4	4*	4*			6	5	
INC			5	6		6	7						
INX									2				
INY									2				
JMP						3							5

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
JSR						6							
LDA		2	3	4		4	4*	4*			6	5*	
LDX		2	3		4	4		4*					
LDY		2	3	4		4	4*						
LSR	2		5	6		6	7						
NØP									2				
ØRA		2	3	4		4	4*	4*			6	5*	
PHA									3				
PHP									3				
PLA									4				
PLP									4				
RØL	2		5	6		6	7						
RTI									6				
RTS									6				
SBC		2	3	4		4	4*	4*			6	5*	
SEC									2				
SED									2				
SEI									2				
STA			3	4		4	5	5			6	6	
STX			3		4	4							
STY			3	4		4							
TAX									2				
TAY									2				
TSX									2				
TXA									2				
TXS									2				
TYA									2				

* Add one cycle if indexing across page boundary
** Add one cycle if branch is taken. Add one additional if branching operation crosses page boundary

In favor of the 6500 series are price and speed; in favor of the 6800 are availability and very good Motorola documentation.

manipulation facilities; or application programs, which make greater use of data manipulation facilities. One would expect better results in the former case with the MOS Technology chip, and in the latter case with the Motorola 6800. One would also expect the MOS Technology chip to enjoy an advantage on large programs, since larger programs inevitably tend to make use of tables, subroutines with parameters, and other forms of address manipulation.

All in all, the Motorola 6800 comes out ahead when considering facilities for manipulating data . . . but nevertheless the two processors are quite similar.

Against these factors one must weigh the availability of an excellent applications manual, proven software, and kits for the hobbyist for the Motorola 6800 microprocessor. At the same time, the MOS Technology chip's price can't be beat, and its speed advantage may be important for some purposes.

At the time that this article is being written (late August), the MOS Technology chip is just a promise: The chip should be available for purchase at the Western Electronics Conference (Wescon) in San Francisco, September 16-19. By the time you read this, the chip itself should be in the hands of at least a few hobbyists. Let's have some letters to BYTE describing initial experiences with the new microprocessor! Send your comments to the author or to the editor of BYTE. In the meantime, we'll be waiting to see what new surprises the semiconductor houses and kit manufacturers have in store for us. And BYTE will try to keep you up to date on the latest developments in the world's hottest, fastest-moving hobby – home computers!

More information on the 6500 series microprocessors is available from:

MOS Technology, Inc.
Valley Forge Corporate Center
950 Rittenhouse Rd.
Norristown PA 19401
1-215-666-7950

Information on the Motorola 6800 microprocessor is available from many local distributors, and from:

Motorola Semiconductor Products Inc.
Box 20912
Phoenix AZ 85036

Table V. MCS6501 microprocessor instructions, listed in alphabetical order by mnemonics. The instructions with asterisks are similar to the same mnemonics in the Motorola 6800 processor.

*ADC	Add with Carry to Accumulator	*JSR	Jump to New Location Saving Return Address
*AND	"AND" to Accumulator	*LDA	Transfer Memory to Accumulator
*ASL	Shift Left One Bit (Memory or Accumulator)	*LDX	Transfer Memory to Index X
		LDY	Transfer Memory to Index Y
*BCC	Branch on Carry Clear	*LSR	Shift One Bit Right (Memory or Accumulator)
*BCS	Branch on Carry Set		
*BEQ	Branch on Zero Result	NØP	Do Nothing - No Operation
*BIT	Test Bits in Memory with Accumulator	*ØRA	"OR" Memory with Accumulator
*BMI	Branch on Result Minus	*PHA	Push Accumulator on Stack
*BNE	Branch on Result not Zero	PHP	Push Processor Status on Stack
*BPL	Branch on Result Plus	*PLA	Pull Accumulator from Stack
*BRK	Force an Interrupt or Break	PLP	Pull Processor Status from Stack
*BVC	Branch on Overflow Clear	*RØL	Rotate One Bit Left (Memory or Accumulator)
*BVS	Branch on Overflow Set		
*CLC	Clear Carry Flag	*RTI	Return From Interrupt
CLD	Clear Decimal Mode	*RTS	Return From Subroutine
*CLI	Clear Interrupt Disable Bit	*SBC	Subtract Memory and Carry from Accumulator
*CLV	Clear Overflow Flag	*SEC	Set Carry Flag
*CMP	Compare Memory and Accumulator	SED	Set Decimal Mode
*CPX	Compare Memory and Index X	*SEI	Set Interrupt Disable Status
CPY	Compare Memory and Index Y	*STA	Store Accumulator in Memory
*DEC	Decrement Memory by One	*STX	Store Index X in Memory
*DEX	Decrement Index X by One	STY	Store Index Y in Memory
DEY	Decrement Index Y by One	TAX	Transfer Accumulator to Index X
*EØR	Exclusive-or Memory with Accumulator	TAY	Transfer Accumulator to Index Y
*INC	Increment Memory by One	*TSX	Transfer Stack Register to Index X
*INX	Increment X by One	TXA	Transfer Index X to Accumulator
INY	Increment Y by One	*TXS	Transfer Index X to Stack Register
*JMP	Jump to New Location	TYA	Transfer Index Y to Accumulator

GLOSSARY

BYTE's Board of Resident Inexperts (BRI) has ruled the following terms to be worthy of further explanation. This list is probably not complete – readers who would like further explanation of terminology are invited to write a letter to the editor identifying terms which need such treatment.

8-Bit Bidirectional Bus – a "data bus" which simultaneously transmits eight separate signals corresponding to one byte's worth of information. The bi-directional aspect means that either tristate, open collector or similar form of output stage is used, so that multiple drivers can be tied in common with only one such driver active at any time. A given board, CPU, output terminal or other logic circuit can then interface to the bus (with some addressing and master timing control intelligence) for both sending and receiving data.

Effective Address – whenever the computer's CPU addresses memory, it must send out 16 bits (for Motorola 6800, MCS 6501 or other similar chips). The way in which these 16 bits are derived can often be a fairly elaborate procedure, as well as a simple absolute expression. Whatever the method of derivation, however, the result is a 16-bit value which is used to address memory, called the effective address because it is what actually does go out to memory regardless of the details of the internal codes of the program.

Instruction Repertoire – the repertoire of a musician is the set of all pieces he or she can play well in concert. Well, the repertoire of a computer – its instructions – is the list of all the instructions it can perform and their definitions.

Subscript – in typical high order languages, a means is provided to specify elements of arrays of data. This is done by subscripts to indicate the "nth" element for subscript "n". Use of such notation presents the problem of calculating the *effective address* of the actual data being referenced. In the context of evaluating a CPU, attention spent on the problem of calculating effective addresses from subscripts is very fundamental.

Time Base – whenever it is necessary to examine the relative timing of different signals, it is necessary to have a reference point and a scale for making the measurement. This is the "time base" of the reference.

TTL compatible – one of the largest families of integrated circuits is the line of "transistor-transistor logic" devices, TTL for short. A TTL compatible line of some non-TTL device can "drive" one or more TTL loads if it is an output, or can receive a TTL device's output if it is an input. There are various cautions to be observed – probably worthy of a BYTE article – when different types of logic are interfaced, but the phrase "TTL compatible" usually means that the compatible device can be wired directly to TTL interconnection pins safely in at least one configuration.

Unary – this term is derived from the Latin roots of "oneness." A unary operation is an operation which has but one operand, for example the complement operation of a Motorola 6800 CPU.