Dann McCleary

AN 8080 SIMULATOR
for the 6502

KIM-1 VERSION
AN 8080 SIMULATOR FOR THE 6502
USER'S MANUAL
KIM - 1 VERSION

COPYRIGHT (C) 1978 BY DANN MCCREARY

THE 8080 SIMULATOR ENABLES A 6502 MICROPROCESSOR TO EXECUTE THE 8080 INSTRUCTION SET. THE SIMULATOR DOES THIS BY INTERPRETING 8080 INSTRUCTIONS IN A NORMAL PROGRAM SEQUENCE, ALL INTERNAL 8080 REGISTERS ARE AVAILABLE FOR EXAMINATION AT ANY TIME. THEY MAY BE VIEWED STATICLY IN A SINGLE-STEP MODE OR DYNAMICLY IN A TRACE MODE, ALL 8080 SOFTWARE FEATURES ARE PRESENTLY SUPPORTED WITH THE EXCEPTION OF DIRECT MEMORY ACCESS (DMA).

THIS SIMULATOR VERSION RUNS ON A BASIC KIM - 1, IN ITS MINIMUM CONFIGURATION, THE 8080 SIMULATOR PROVIDES SINGLE STEP AND FULL SPEED MODES AND LEAVES 227 BYTFS OF MEMORY OPEN FOR 8080 PROGRAMS, ALTERNATIVE RUN MODE FEATURES TAKE UP ADDITIONAL (OPTIONAL) MEMORY SPACE.

THE CASSETTE TAPE INCLUDED IN THIS PACKAGE IS RECORDED IN STANDARD KIM - 1 FORMAT, THE PURCHASER OF THIS PACKAGE IS HERewith GRANTED PERMISSION TO MAKE ONE COPY FOR HIS/HER OWN PERSONAL USE, RETAINING THE ORIGINAL AS A BACKUP.

THE AUTHOR WELCOMES ANY FEEDBACK FROM SIMULATOR USERS AND WILL BE PLEASED TO RESPOND TO SPECIFIC QUESTIONS WHICH ARE ACCOMPANIED BY A STAMPED, SELF-ADDRESSED ENVELOPE.

ADDITIONAL COPIES ARE AVAILABLE, CONSISTING OF A KIM - 1 FORMAT CASSETTE TAPE, A USER MANUAL AND A COMPLETE, WELL COMMENTED ASSEMBLY LEVEL SOURCE/OBJECT LISTING, PRICED AT $20.00 + $1.50 POSTAGE & HANDLING (CALIFORNIA RESIDENTS PLEASE ADD 6% SALES TAX). THEY MAY BE ORDERED FROM

DANN MCCREARY
P.O. BOX 16435-8
SAN DIEGO, CA 92116

AS AN ADEQUATE TREATMENT OF THE INSTRUCTION SET AND PROGRAMING TECHNIQUES FOR THE 8080 MICROPROCESSOR IS BEYOND THE SCOPE OF THIS MANUAL, IT IS SUGGESTED THAT THE USER AQUIRE THE "8080/8085 ASSEMBLY LANGUAGE PROGRAMMING MANUAL" FROM INTEL.

ACKNOWLEDGEMENT: THANKS TO GARY DAVIS FOR GENEROUS SUPPORT IN THE FORM OF ACCESS TO HIS 8080 SYSTEM AND HIS ASSISTANCE IN RUNNING COMPARISON TESTS,
INTRODUCTION

WHY IMITATE ONE MICROPROCESSOR WITH ANOTHER? YOU PROBABLY PURCHASED THIS 8080 SIMULATOR PACKAGE TO DO ONE OR MORE OF THE FOLLOWING:

- RUN EXISTING 8080 SOFTWARE ON YOUR 6502
- WRITE, TEST AND DEBUG YOUR OWN 8080 SOFTWARE
  WITHOUT HAVING TO PURCHASE A COMPLETE
  8080 BASED SYSTEM
- LEARN SOMETHING ABOUT THE ARCHITECTURE
  AND INSTRUCTION SET OF THE 8080
  VIA HANDS-ON EXPERIENCE

BY SPENDING A LITTLE TIME WITH THIS MANUAL AND GOING THROUGH THE STEP BY STEP "FIRST TIME AROUND" PROCEDURE, YOU WILL SOON BE DOING ALL THIS AND MORE, LATER, WHEN YOU KNOW THE CAPABILITIES AND LIMITATIONS OF THE SIMULATOR, YOU HAVE THE OPTION OF RELOCATING IT PERMANENTLY IN ROM.

THE ESSENTIAL PART OF THE 8080 SIMULATOR IS THE SUBROUTINE CALLED "SIM80", EACH TIME THIS SUBROUTINE IS CALLED, ANOTHER 8080 INSTRUCTION IS EXECUTED. PROGRAMS OF VARYING COMPLEXITY MAY CALL AND USE "SIM80", DEPENDING ON YOUR OBJECTIVES. A TYPICAL EXAMPLE OF SUCH A PROGRAM IS INCLUDED ON PAGE THREE OF THE SIMULATOR PROGRAM LISTINGS, IN CONJUNCTION WITH A SIMPLE SET OF SWITCHES, IT PROVIDES FOR EASY SELECTION OF RUN, TRACE OR SINGLE STEP MODES AND BREAKPOINT OPERATION, ALTERNATIVELY, THE SIMULATOR MAY BE RUN IN A MINIMUM MODE IN ORDER TO LEAVE A GREATER AMOUNT OF MEMORY FOR 8080 PROGRAMS.

AFTER STUDYING THESE CALLING PROGRAMS YOU MAY WISH TO DESIGN YOUR OWN SPECIAL PURPOSE CALLING PROGRAM, FOR EXAMPLE, TRY A "SUPER BREAKPOINT" PROGRAM, ALL SIMULATED 8080 REGISTERS ARE MAINTAINED IN RAM, YOU CAN MONITOR THEM BETWEEN 8080 INSTRUCTIONS AND FORCE A BREAKPOINT ON THE BASIS OF A PARTICULAR PATTERN.

I/O HANDLING IS FACILITATED BY A SPECIAL INSTRUCTION DESIGNED INTO THE SIMULATOR, THE "C65" INSTRUCTION ENABLES YOU TO CALL 6502 CODED SUBROUTINES FROM YOUR 8080 PROGRAM, WITH SLIGHT PRECAUTIONS, THEN, YOU CAN USE ALL YOUR SYSTEM MONITOR'S SUBROUTINES.

INCLUDED IN THE SIMULATOR LISTINGS IS AN 8080 TIME-OF-DAY CLOCK PROGRAM, BY USING THE SIMULATOR TO RUN THIS PROGRAM, YOU WILL SEE BY EXAMPLE ALL THE VARIOUS MODES AND FEATURES PROVIDED BY THE 8080 SIMULATOR, SO, WHY NOT GET STARTED?
TAPE LOADING INSTRUCTIONS

THE 8080 SIMULATOR KIM - 1 FORMAT CASSETTE TAPE CONSISTS OF 4 BLOCKS OF PROGRAMS AND DATA.

BLOCK 1 IS THE 8080 REGISTER AREA AND INTERPRETER ROUTINES, ADDRESSES #$00E3 TO #$03FF.

BLOCK 2 IS THE MAIN CONTROL LOOP, ADDRESSES #$1780 TO #$17E6.

BLOCK 3 IS THE OPTIONAL EXECUTIVE AND INTERRUPT AREA, ADDRESSES #$0076 TO #$00E2.

BLOCK 4 IS A SAMPLE 8080 PROGRAM, ADDRESSES #$0000 TO #$0075.

THE KIM DATA BLOCK I.D. FOR ALL BLOCKS IS #$80.

LOADING PROCEDURE

<table>
<thead>
<tr>
<th>ACTION</th>
<th>SEE DISPLAYED</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) <em>POWER ON KIM &amp; CASSETTE INTERFACE</em></td>
<td></td>
</tr>
<tr>
<td>(2) <em>REWIND CASSETTE</em></td>
<td></td>
</tr>
<tr>
<td>(3) PUSH [RS1, IAD] 00F1</td>
<td>00F1 XX</td>
</tr>
<tr>
<td>(4) PUSH [DA], 00</td>
<td>00F1 00</td>
</tr>
<tr>
<td>(5) PUSH [AD], 17F9</td>
<td>17F9 XX</td>
</tr>
<tr>
<td>(6) PUSH [DA], 00</td>
<td>17F9 00</td>
</tr>
<tr>
<td>(7) <em>START CASSETTE PLAYER</em></td>
<td></td>
</tr>
<tr>
<td>(8) PUSH [AD], 1873</td>
<td>1873 A9</td>
</tr>
<tr>
<td>(9) PUSH [GO]</td>
<td>BLANK DISPLAY</td>
</tr>
<tr>
<td>(10) <em>DISPLAY RELIGHTS</em></td>
<td>0000 XX</td>
</tr>
<tr>
<td>(14) <em>REPEAT STEPS 8 - 10 FOR REMAINING BLOCKS</em></td>
<td></td>
</tr>
<tr>
<td>(15) PUSH [AD], 00F1</td>
<td>00F1 00</td>
</tr>
<tr>
<td>(16) PUSH [DA], FF</td>
<td>00F1 FF</td>
</tr>
<tr>
<td>(17) PUSH [AD], 17FA</td>
<td>17FA XX</td>
</tr>
<tr>
<td>(18) PUSH [DA], 00</td>
<td>17FA 00</td>
</tr>
<tr>
<td>(19) PUSH [♦]</td>
<td>17FB XX</td>
</tr>
<tr>
<td>(20) PUSH 17</td>
<td>17FB 17</td>
</tr>
</tbody>
</table>
OPERATING INSTRUCTIONS

TO FAMILIARIZE YOURSELF WITH 8080 SIMULATOR OPERATION LOAD TAPE BLOCKS 1 - 4 AND WORK YOUR WAY THROUGH THE FOLLOWING INSTRUCTIONS.

ONCE THE TAPE IS LOADED, BE SURE TO PUT #$FF AT ADDRESS #$00F1, NOW LOAD THE MINIMUM CONFIGURATION INTERRUPT VECTORS: #$80 INTO LOCATION #$17FA AND #$17 INTO #$17FB, THIS IS THE NMI VECTOR WHICH IS ACTIVATED BY THE [ST] KEY ON KIM, LOAD #$22 INTO #$17FE & #$1C INTO #$17FF FOR THE IRQ VECTOR.

MINIMUM CONFIGURATION

SINGLE STEP MODE

CHANGE LOCATIONS #$178E - #$1790 TO #$4C, #$22, #$1C, THE PROGRAM COUNTER SHOULD CONTAIN #$0000 (#$00 IN LOCATION #$00EF AND #$00 IN LOCATION #$00F0), IT POINTS TO THE FIRST BYTE OF 8080 PROGRAM AREA WHICH NOW CONTAINS A SAMPLE 8080 PROGRAM.

LOOK AT LOCATION #$00EF, THE LOW ORDER BYTE OF THE PROGRAM COUNTER, PUSH THE [ST] KEY AND YOU WILL SEE IT CHANGE FROM #$00 TO #$03, EACH SUBSEQUENT OPERATION OF THE [ST] KEY WILL ADVANCE THE SIMULATOR THROUGH ANOTHER PROGRAM STEP.

WITH THE DEMO PROGRAM LISTING IN HAND YOU CAN VERIFY THAT THE PROGRAM IS FOLLOWING ITS PROPER SEQUENCE, BETWEEN EACH PROGRAM STEP CONTROL IS RETURNED TO THE KIM MONITOR PROGRAM, THIS MAKES IT POSSIBLE TO VIEW OR CHANGE THE CONTENTS OF ANY SIMULATED 8080 REGISTER, FOR INSTANCE, LOOK AT LOCATION #$00E5, THIS IS THE 8080 ACCUMULATOR, NOW CONTINUE PRESSING THE [ST] KEY AND YOU CAN WATCH DATA MOVING INTO THE ACCUMULATOR, BEING INCREMENTED AND SO ON, BY USING THE [DA] AND HEX KEYS BETWEEN PROGRAM STEPS YOU CAN MODIFY THAT DATA TO SEE WHAT EFFECT THAT MAY HAVE ON PROGRAM EXECUTION, NOTE: THIS PROGRAM INTERACTS WITH THE KIM DISPLAY BUFFER, SO BE PREPARED FOR THE DISPLAY TO CHANGE AFTER THE PROGRAM REACHES ADDRESS #$002B.

PC SINGLE STEP

THE 8080 PROGRAM COUNTER IS LOCATED ON PAGE ZERO IN THE SAME PLACE KIM MONITOR ROUTINES STORE THE 6502 PROGRAM COUNTER, IF YOU PUSH THE [PC] KEY BETWEEN INSTRUCTION STEPS, YOU WILL SEE THE CURRENT 8080 PROGRAM ADDRESS AND OP-CODE DISPLAYED, BY PLACING #$4C, #$0C, #$1C AT ADDRESS #$178E - #$1790, THIS WILL BE DONE FOR YOU BY THE KIM MONITOR.

RUN MODE

TO RUN THE SAMPLE PROGRAM AT FULL SPEED JUST CHANGE THE DATA AT LOCATION #$178E - #$1790 TO #$58, #$58, #$58, NOW WHEN YOU PUSH [ST] THE SIMULATOR WILL START AND CONTINUE RUNNING UNTIL YOU PUSH [RS] OR ACTIVATE THE IRQ, USING THE IRQ IS RECOMMENDED, AS [RS] WILL INTERRUPT THE SIMULATOR, LEAVING ITS REGISTERS IN AN UNKNOWN STATE, NOTE: MINIMUM CONFIGURATION DOES NOT SUPPORT INTERRUPTS BUT IT DOES EXECUTE INTERRUPT-RELATED INSTRUCTIONS.
ILLEGAL OP-CODES & BREAKPOINTS

When the 8080 Simulator encounters an illegal op-code, a jump will be forced to the system monitor. In the minimum configuration, this may be used to advantage by inserting an illegal op-code (try #$12) in place of any op-code in your 8080 program. This will act as a breakpoint, to continue operation after a breakpoint, replace the original op-code, subtract 1 from the 8080 program counter, and continue normally. (Note: do not use #$CB as a break.)

OPTIONAL FEATURES

Using the simulator's optional features requires the illustrated switching arrangement connected to KIM port PB, in particular, do not attempt to use the interrupt subroutine without the appropriate setup. Omitting the mode control switches however will only result in a default to the single step mode, with switches connected, put #$FD, #$00 into #$17FA & #$17FB, put #$58, #$60, #$EA back into #$178F - #$1790.

REGISTER SINGLE STEP MODE

Set mode control switches (left to right) to register, step & step, this position is identical in operation to the minimum configuration single step mode.

REGISTER TRACE MODE

Set control switches to register, trace & step, when you activate the [ST] key the simulator will begin executing 8080 instructions at a rate determined by the time-delay value stored in "speed," address #$00E2.

PROGRAM COUNTER TRACE MODE

Set control switches to PC, trace & step, now the display will trace the progress of the 8080 program counter and the 8080 program op-codes.

PROGRAM COUNTER SINGLE STEP MODE

Set control switches to PC, step & step, now pushing the [ST] key is required to advance the program counter.

TRACE SPEED

Trace speed may be set by inserting a value in "speed," #$00E2, a value of #$00 gives the slowest trace, and #$81 the fastest.

RUN MODE

Move the run switch to "run," push [ST] and your program will run at full speed. Try setting the demo program clock by keying in the time in rapid succession, move the run switch back to revert to trace or single step.
INTERRUPT ACTION

INSERT #$20, #$76, #$00 AT #$179F - #$17A1 TO USE INTERRUPTS, WHEN PB7 IS BROUGHT HIGH AN 8080 INTERRUPT WILL OCCUR (NOTE: THE PB7 INTERRUPT LINE MUST BE HELD HIGH THROUGH AT LEAST ONE SIMULATOR CYCLE FOR AN INTERRUPT TO TAKE PLACE), NO INTERRUPT ACKNOWLEDGE LINE IS PROVIDED - IF ONE IS REQUIRED, INSERT AN OUTPUT INSTRUCTION JUST BEFORE "NOINT" IN THE INTERRUPT LOGIC ON PAGE 2 OF THE LISTINGS.

THE "INTF" INTERRUPT FLAG (ADDRESS #$00E3) IS OPPOSITE IN SENSE FROM THE 8080 INTERRUPT FLAG - i.e., IT DISABLES INTERRUPTS WHEN SET TO ONE AND ENABLES THEM WHEN SET TO ZERO, IT ALSO GOES THROUGH A TRANSITION STATE OF #$FF AFTER AN "EI" INSTRUCTION IS EXECUTED, THE RESULTANT ACTION, HOWEVER, IS IDENTICAL TO THAT OF AN ACTUAL 8080, THE INTERRUPT VECTOR IS SET BY SWITCHES AT PB 3, 4, & 5.

THE 8080 "HALT" INSTRUCTION

#$76 WILL CAUSE AN 8080 HALT, IN AN ACTUAL 8080, THE PROCESSOR HALTS WITH THE PROGRAM COUNTER POINTING TO THE NEXT INSTRUCTION IN SEQUENCE, ON THE SIMULATOR, HOWEVER, IN ORDER TO MAINTAIN CONTROL AND TO KEEP MONITORING FOR SIMULATED INTERRUPTS, A #$76 WILL RESULT IN NO ADVANCEMENT OF THE PROGRAM COUNTER - RATHER, THE #$76 WILL BE REPEATEDLY EXECUTED UNTIL AN INTERRUPT IS DETECTED OR UNTIL THE SIMULATOR IS HALTED EXTERNALLY, AFTER AN INTERRUPT IS SERVICED, EXECUTION WILL CONTINUE WITH THE INSTRUCTION IMMEDIATELY FOLLOWING THE HALT,

BREAKPOINT OPERATION

THE 8080 SIMULATOR ALLOWS MULTIPLE BREAKPOINTS, STORE THESE BREAKPOINT ADDRESSES IN "BKTBL", A RAM AREA, START PROGRAM EXECUTION IN THE NORMAL MANNER, WHEN THE PROGRAM REACHES THE BREAKPOINT A JUMP WILL BE FORCED, RETURNING CONTROL TO THE KIM MONITOR, YOU MAY NOW EXAMINE OR MODIFY ANY 8080 REGISTERS, TO CONTINUE OPERATION, ALL THAT IS REQUIRED IS TO PUSH [ST], THE SIMULATOR WILL RUN UNTIL THE NEXT BREAK IS ENCOUNTERED, "BKTBL" MAY BE EXPANDED UP TO 128 ENTRIES.

SIMULATOR I/O HANDLING

NORMAL SIMULATOR I/O INSTRUCTIONS ARE HANDLED VIA "IOTBL", THIS IS A TABLE OF ADDRESSES OF INPUT/OUTPUT PORTS, NOTE THAT THESE I/O ADDRESSES ARE STORED IN NORMAL ORDER, i.e., MOST SIGNIFICANT ADDRESS FIRST, THE SIMULATOR ASSUMES THAT THERE IS A DATA DIRECTION REGISTER AT THE PORT ADDRESS +1, IF YOU ASSIGN RAM LOCATIONS AS SIMULATED I/O PORT ADDRESSES, 2 LOCATIONS MUST BE ALLOWED FOR EACH PORT.

THE FIRST ENTRY IN "IOTBL" WILL BE 8080 PORT 0, THE SECOND ENTRY WILL BE 8080 PORT 1, AND SO ON UP TO A MAXIMUM OF PORT 127.

THIS VERSION OF THE SIMULATOR ASSIGNS ALL I/O INSTRUCTIONS TO KIM PORTS A & SA, PORT SA IS ALSO SHARED BY THE KIM KEYPAD, SO TO AVOID INTERFERENCE DO NOT USE THE KEYBOARD (EXCEPT FOR [ST]) WHILE INPUT INSTRUCTIONS ARE BEING USED.
THE 8080 "RST" INSTRUCTION

THE SIMULATOR SENDS RST CALLS TO PAGE ZERO, AS DOES THE 8080, IF HOWEVER YOUR PAGE ZERO IS TAKEN UP WITH MONITOR ROUTINES, ETC., THE RESET PAGE MAY BE CHANGED BY CHANGING "RSTHI" FROM #$00 TO WHATEVER PAGE YOU WISH, IT IS LOCATED AT ADDRESS #$0189 ON PAGE 9 OF THE SIMULATOR LISTINGS.

THE "CALL6502", OR "C65" OP-CODE

THE SIMULATOR APPROPRIATES ONE OF THE 8080'S UNIMPLEMENTED OP-CODES FOR SPECIAL USE, IT IS #$CB, AND TAKES THE FORM "CB XXXX", WHERE XXXX IS THE ADDRESS -1 OF ANY 6502 SUBROUTINE. SUBROUTINES WHICH DON'T REQUIRE ANY PARAMETERS MAY BE CALLED DIRECTLY, SUBROUTINES WHICH REQUIRE SOME DATA PASSED TO OR FROM THEM MUST BE PREFACED OR ENDED WITH CODE TO MOVE THE REQUIRED DATA INTO OR OUT OF APPROPRIATE 8080 REGISTERS, SEE AS AN EXAMPLE THE "DSKBD" SUBROUTINE ON PAGE 20 OF THE PROGRAM LISTINGS. IT DOESN'T NEED ANY DATA PASSED TO IT, BUT MUST RETURN A KEY VALUE IN A, RE SURF TO SPECIFY THE ADDRESS MINUS ONE (ADDR,-1) WHEN USING "C65".

RELOCATION INFORMATION

THE 8080 SIMULATOR IN ITS MINIMUM CONFIGURATION DOES NOT RELY ON ANY KIM MONITOR SUBROUTINES PER-SE, RATHER, CONTROL IS PASSED RACK AND FORTH BETWEEN THE SIMULATOR AND THE KIM MONITOR, IT IS EASILY RELOCATED WITHIN A KIM SYSTEM AND EASILY ADAPTED TO OTHER NON-KIM 6502 SYSTEMS, THE SIMULATOR CAN BE RELOCATED IN ROM WITH THE EXCEPTION OF THE REGISTER AREA AND OTHER ZERO-PAGE VARIABLES, IF YOU DON'T HAVE ACCESS TO AN ASSEMBLER BUT WOULD LIKE TO PUT THE SIMULATOR IN HIGH MEMORY, LEAVE THE RELATIVE POSITIONS OF THE MAJOR ROUTINES (#$0100 -> #$03FF) THE SAME, AND JUST BE SURE TO CHANGE ALL THE HIGH ORDER ADDRESSES (I.E., THE THIRD BYTE OF ANY THREE BYTE INSTRUCTION.)

TO USE THE 8080 SIMULATOR IN A NON-KIM 6502 SYSTEM ADDITIONAL MODIFICATIONS ARE NECESSARY, ALL PROGRAM LINES WHICH REFERENCE KIM MONITOR-RESIDENT DATA HAVE THE WORD "KIM" SOMEWHERE IN THE COMMENT AREA, THESE INCLUDE I/O ADDRESSES AND SUBROUTINE CALLS, I/O ADDRESSES SHOULD BE CHANGED TO AVAILABLE PORTS IN YOUR SYSTEM.

ONE SUBROUTINE CALL IS TO SCAND, SCAND IS A MONITOR SUBROUTINE WHICH DISPLAYS THE ADDRESS SPECIFIED BY "POIN1" AND "POINH" ALONG WITH THE DATA BYTE AT THAT ADDRESS, SUBSTITUTE A SIMILAR DISPLAY ROUTINE FROM YOUR SYSTEM MONITOR, GETKEY READS A ONE BYTE VALUE FROM KIM'S HEXADECIMAL KEYPAD, ANY COMPARABLE INPUT ROUTINE IN YOUR MONITOR WILL DO,
GENERAL OPERATING CONSIDERATIONS

6502 ADDRESSING MODES MAKE HEAVY USE OF ZERO-PAGE RESOURCES AND SINCE PAGE ONE IS USED FOR A STACK, 8080 PROGRAMS MUST BYPASS THESE AREAS, IF THE 8080 APPLICATION PROGRAM YOU WISH TO RUN USES #$0000 - #$00FF OR #$0100 TO #$01FF, RELOCATE THAT PORTION OF THE PROGRAM IN ANOTHER AREA AND PROVIDE JUMPS TO AND FROM THE NEW AREA.

SIMULATOR PROGRAMS RUN CONSIDERABLY SLOWER THAN THE SAME PROGRAM WOULD RUN ON ACTUAL 8080 HARDWARE. THIS IS, OF COURSE, BECAUSE THE SIMULATOR MUST EXECUTE MANY 6502 INSTRUCTIONS IN THE COURSE OF EXECUTING ONE 8080 INSTRUCTION. DON'T EXPECT BLINDING SPEED ON THE OTHER HAND, TIME DEPENDENT EVENTS OF MODERATE FREQUENCY (SUCH AS THE INCLUDED CLOCK PROGRAM) CAN BE HANDLED REASONABLY WELL, IF YOU INTEND TO TRANSFER PROGRAMS TO AN ACTUAL 8080 BASED SYSTEM, BE SURE TO MAKE ALLOWANCES IN YOUR TIME DELAY ROUTINES, FOR HIGHLY TIME DEPENDENT EVENTS, USE THE "C65" INSTRUCTION TO HANDLE THEM IN 6502 CODE.

UNLIKE AN ACTUAL 8080 PROCESSOR, SIMULATOR REGISTERS ARE MAINTAINED IN MEMORY, THIS REQUIRES CARE THAT YOUR 8080 PROGRAM NOT ACCESS OR MODIFY THESE REGISTERS, THE SAME ALSO HOLDS TRUE FOR THE SIMULATOR PROGRAM ITSELF, AVOID REFERENCING ANY SIMULATOR PROGRAM AREA FROM YOUR 8080 PROGRAM, IF YOU SUSPECT THIS MAY HAVE HAPPENED, RELOAD THE SIMULATOR FROM TAPE.

AN EASY TRAP FOR 8080 PROGRAMMERS TO FALL INTO IS "ACCIDENTAL INITIALIZATION" OF REGISTERS, THESE ARE THE SYMPTOMS: YOU HAVE COMPLETED WORK ON AN 8080 PROGRAM, IT RUNS PERFECTLY ON THE SIMULATOR, YOU LOAD IT INTO THE TARGET SYSTEM, HIT "GO" AND NOTHING HAPPENS, WHY?

TO OPERATE PROPERLY YOUR 8080 PROGRAM MAY REQUIRE THAT ONE OR MORE REGISTERS BE INITIALIZED TO CERTAIN VALUES OR RANGES OF VALUES, THE SIMULATOR REGISTERS MAY BE PRE-SET TO THOSE CONDITIONS FROM PRIOR OPERATION, THUS YOUR PROGRAM MAY RUN ON THE SIMULATOR EVEN THOUGH IT WAS NO INSTRUCTIONS TO PROPERLY INITIATE THOSE REGISTERS, THIS MAY EVEN HAPPEN ON YOUR TARGET SYSTEM, DUE TO RANDOM VARIATIONS FROM ONE PROCESSOR CHIP TO ANOTHER, EACH PROCESSOR MAY "COME UP" WITH DIFFERENT RANDOM VALUES IN UNINITIALIZED REGISTERS, THE RESULT, YOUR PROGRAM WORKS FINE WITH SOME PROCESSORS BUT NOT AT ALL WITH OTHERS.

TO AVOID THIS PITFALL, TAKE THESE STEPS: BEFORE EACH SERIOUS EVALUATION OF AN 8080 PROGRAM RELOAD THE 8080 REGISTER AREA TO A POWER-ON CONFIGURATION WITH RANDOM DATA IN THE REGISTERS, TRY SEVERAL PROGRAM RUNS WITH DIFFERENT RANDOM DATA IN THE REGISTERS AT STARTUP, THIS WILL ASSURE YOU THAT YOUR INITIALIZATION SEQUENCES ARE WORKING AS YOU EXPECT,
*************** AN 8080 SIMULATOR FOR THE 6502 ***************
*************** KIM=1 VERSION ***************

* THIS PROGRAM SIMULATES THE OPERATION OF 8080 MICRO-
* PROCESSOR SOFTWARE ON A KIM - 1 6502 MICROCOMPUTER,
* IN ORDER TO SIMPLIFY RELOCATION TO ANOTHER NON -
* KIM 6502 SYSTEM, LINES DEPENDENT ON KIM MONITOR
* RESIDENT DATA ARE FLAGGED WITH THE WORD *KIM* IN
* THE COMMENT FIELD.

*************** COPYRIGHT (C) 1978 BY DANN MCCREARY ***************

*************** SYMBOl DEFINITIONS ***************

GETKEY EQU #$1F6A J*KIM* KEYPAD SUBROUTINE
SCAN EQU #$1F19 J*KIM* DISPLAY SUBROUTINE
SCANDS EQU #$1F1F J*KIM* DISPLAY SUBROUTINE
PADD EQU #$1701 J*KIM* PORT A DATA DIR, REG,
PA EQU #$1700 J*KIM* PORT A DATA
SADD EQU #$1741 J*KIM* PORT SA DATA DIR, REG,
SA EQU #$1740 J*KIM* PORT SA DATA
PBDD EQU #$1703 J*KIM* PORT B DATA DIR, REG,
PB EQU #$1702 J*KIM* PORT B DATA
MONITOR EQU #$1C22 J*KIM* RESET ENTRY TO MONITOR
PCCMD EQU #$1CDC J*KIM* PC SINGLE STEP ENTRY
REGS EQU A 18080 REG, BASE ADDR,
HL EQU L JML REGISTER PAIR
PC EQU PCL 18080 PROGRAM COUNTER
INST EQU SCR JCURRENT 8080 INSTRUCTION
(HL) EQU HL-REGS JINDEXES TO REGISTERS, ETC.,
(DE) EQU DE-REGS
(SP) EQU SP-REGS
(PC) EQU PC-REGS
(DECIT) EQU DECIT-REGS
(INCIT) EQU INCIT-REGS
(PNT) EQU PNT-REGS
(S) EQU SCR-REGS
SMASK EQU #$80 JISOLATES SIGN BIT FROM PSW
PHMASK EQU #$84 JISOLATES PARITY BIT
CMASK EQU #$81 JISOLATES CARRY BIT
ZMASK EQU #$40 JISOLATES ZERO BIT
SINC EQU #$81 JFOR INR/DCR
MINS PD EQU #$20 JSETS MINIMUM TRACE SPEED
HALT EQU #$7F 15000 HALT OP-CODE
BKTBLN EQU ENDBK-BKTBL JLENGTH OF BRKPNT TABLE
NORRK EQU #$FFFF JDUMMY BREAKPOINT ENTRY
RSTHI EQU #$80 JHIGH ORDER 18080 RST VECTOR
INPG2 EQU #$02 JSECOND PG OF INTERP, ROUTINES
INTDIS EQU #$01 JDISABLES INTERRUPTS
********KIM-1 INTERRUPT VECTORS********

17FA

**ORG #17FA
17FA 8D00 NMI DC 2, START *KIM* NON-MASKABLE INTERRUPT
17FC FFFF RESET DC 2, #FFFF *KIM* RESET VECTOR

*******MINIMUM CONFIGURATION VECTORS*******

*17FA 8017 NMI DC 2, MNSTRT *KIM* NON-MASKABLE INTERRUPT
*17FC FFFF RESET DC 2, #FFFF *KIM* RESET
*17FE 221C IRQ DC 2, MNITOR *KIM* IRQ VECTOR

******************************************************************************

0076

**ORG #0076

*******8080 INTERRUPT LOGIC*******

0076 AA INT TAX JSAVE HALT DATA
0077 A9 01 LOAD #01 JSAVE INTERRUPT PORT
0079 BD 0317 STA P800 *KIM*
007C A8 TAY JUSE TO SET INTE
007D AD 0217 LDA PB *KIM*
0080 25 E3 ANDZ INTE JINT=FF IF ENABLED
0082 10 07 BPL NOINT JINTERRUPT?
0084 84 E3 STYZ INTE JYES, DISABLE FURTHER INTERRUPTS
0086 09 C7 ORA #C7 JFORM RST INSTRUCTION
0088 85 FE STA I INST JSAVE NEW INSTRUCTION
008A 88 DEY JSAVE INTERRUPT TAKEN
008B A5 E3 NOINT LDAZ INTE
008D DD 02 BNE NINT JENABLE INTERRUPT
008F C6 E3 DECZ INTE
0091 8A NINT TXA
0092 DD 01 BNE NONLY IAWAS THIS A HALT?
0094 88 DEY IYES
0095 98 NONLY TYA ISPECIAL RETURN?
0096 DD 05 BNE NRMRET IYES, GO RETURN
0098 86 PLA INO, CHANGE RETURN TO
0099 10 CLC
009A 69 03 ADCTM #03 IPOINT PAST JSR INCPC
009C 48 PHA
009D 60 NRMRET RTS
*******OPTIONAL EXECUTIVE LOOP*******

**********ENTER AT "START"**********

<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>009E</td>
<td>A5 E2</td>
<td>TRACE</td>
<td>LDAZ</td>
</tr>
<tr>
<td>00A0</td>
<td>85 FE</td>
<td>SPEED</td>
<td>STAZ</td>
</tr>
<tr>
<td>00A2</td>
<td>20 191F</td>
<td>TRA</td>
<td>JSR</td>
</tr>
<tr>
<td>00A5</td>
<td>C0 FE</td>
<td>SCR</td>
<td>DECZ</td>
</tr>
<tr>
<td>00A7</td>
<td>08 F9</td>
<td>SCR</td>
<td>BNE, TRA</td>
</tr>
<tr>
<td>00A9</td>
<td>A2 02</td>
<td>BRK</td>
<td>LDXIIM</td>
</tr>
<tr>
<td>00AA</td>
<td>A5 EF</td>
<td>BKLP</td>
<td>LDAZ</td>
</tr>
<tr>
<td>00AD</td>
<td>DD DF00</td>
<td>CMPX</td>
<td>BKTBL-2</td>
</tr>
<tr>
<td>00B0</td>
<td>08 07</td>
<td>BNE</td>
<td>NXRK</td>
</tr>
<tr>
<td>00B2</td>
<td>A5 F0</td>
<td>LDAZ</td>
<td>PCH</td>
</tr>
<tr>
<td>00B4</td>
<td>DD DF00</td>
<td>CMPX</td>
<td>BKTBL-1</td>
</tr>
<tr>
<td>00B7</td>
<td>F0 24</td>
<td>Req</td>
<td>MONIT</td>
</tr>
<tr>
<td>00B9</td>
<td>CA</td>
<td>NXRK</td>
<td>DEX</td>
</tr>
<tr>
<td>00BA</td>
<td>CA</td>
<td>DEX</td>
<td>BNE</td>
</tr>
<tr>
<td>00BD</td>
<td>A2 FF</td>
<td>START</td>
<td>LDXIIM</td>
</tr>
<tr>
<td>00BF</td>
<td>9A</td>
<td>TXS</td>
<td>JSR</td>
</tr>
<tr>
<td>00C0</td>
<td>20 0317</td>
<td>SIM80</td>
<td>JSR</td>
</tr>
<tr>
<td>00C3</td>
<td>A9 00</td>
<td>LDAIM</td>
<td>INSET</td>
</tr>
<tr>
<td>00C5</td>
<td>0D 0317</td>
<td>STA</td>
<td>PBDD</td>
</tr>
<tr>
<td>00CD</td>
<td>AD 0217</td>
<td>LOA</td>
<td>PB</td>
</tr>
<tr>
<td>00CB</td>
<td>29 07</td>
<td>ANDIM</td>
<td>#507</td>
</tr>
<tr>
<td>00CD</td>
<td>4A</td>
<td>LSRA</td>
<td>PCL</td>
</tr>
<tr>
<td>00CE</td>
<td>90 09</td>
<td>BRK</td>
<td>YES, GO RUN</td>
</tr>
<tr>
<td>00D0</td>
<td>4A</td>
<td>LSRA</td>
<td>MD, REGISTER MODE?</td>
</tr>
<tr>
<td>00D1</td>
<td>08 08</td>
<td>BNE</td>
<td>REG</td>
</tr>
<tr>
<td>00D3</td>
<td>A5 EF</td>
<td>LDAZ</td>
<td>PCL</td>
</tr>
<tr>
<td>00D5</td>
<td>85 FA</td>
<td>STA</td>
<td>PointL</td>
</tr>
<tr>
<td>00D7</td>
<td>A5 F0</td>
<td>LDAZ</td>
<td>PCH</td>
</tr>
<tr>
<td>00D9</td>
<td>85 FB</td>
<td>STA</td>
<td>PointH</td>
</tr>
<tr>
<td>00DB</td>
<td>90 C1</td>
<td>REG</td>
<td>BCC</td>
</tr>
<tr>
<td>00DD</td>
<td>4C 221C</td>
<td>MONIT</td>
<td>JMP</td>
</tr>
</tbody>
</table>

*NOTE: *KIM* stands for "Keyboard Input Mode."
**BREAKPOINT ADDRESS TABLE**

<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00E0</td>
<td>FF FF</td>
<td>BREAKPOINT ADDRESS TABLE</td>
</tr>
<tr>
<td>00E2</td>
<td>00</td>
<td>SPEED DC MIN SPD SETS TRACE SPEED</td>
</tr>
</tbody>
</table>

*THE FOLLOWING ITEMS ON THIS PAGE MUST BE LOCATED IN PAGE 0 RAM MEMORY. THEY MUST REMAIN IN ORDER AND BEGIN ON AN ODD BOUNDARY.*

**8080 REGISTERS**

<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00E3</td>
<td>01</td>
<td>INTE DC INTDIS INTERRUPT ENABLE FLAG</td>
</tr>
<tr>
<td>00E4</td>
<td>02</td>
<td>PSW DC #02 18080 PROCESSOR STATUS</td>
</tr>
<tr>
<td>00E5</td>
<td>00</td>
<td>A DC #00 18080 ACCUMULATOR</td>
</tr>
<tr>
<td>00E6</td>
<td>00</td>
<td>M DC #00 DUMMY MEMORY REGISTER</td>
</tr>
<tr>
<td>00E7</td>
<td>00</td>
<td>L DC #00 JL REGISTER PAIR</td>
</tr>
<tr>
<td>00E8</td>
<td>00</td>
<td>H DC #00 IDE REGISTER PAIR</td>
</tr>
<tr>
<td>00E9</td>
<td>00</td>
<td>E DC #00 IDE REGISTER PAIR</td>
</tr>
<tr>
<td>00EA</td>
<td>00</td>
<td>D DC #00 JBC REGISTER PAIR</td>
</tr>
<tr>
<td>00EB</td>
<td>00</td>
<td>C DC #00 JBC REGISTER PAIR</td>
</tr>
<tr>
<td>00EC</td>
<td>00</td>
<td>B DC #00 JBC REGISTER PAIR</td>
</tr>
<tr>
<td>00ED</td>
<td>74</td>
<td>SPL DC #74 18080 STACK POINTER</td>
</tr>
<tr>
<td>00EE</td>
<td>00</td>
<td>SPH DC #00</td>
</tr>
<tr>
<td>00EF</td>
<td>00</td>
<td>PCL DC #00 18080 PROGRAM COUNTER</td>
</tr>
<tr>
<td>00F0</td>
<td>00</td>
<td>PCH DC #00</td>
</tr>
</tbody>
</table>

**CONSTANTS AND DATA AREA**

*NOTE: #S00F1 MUST BE SET TO #300 WHEN READING OR WRITING.*
*TAPES AND MANUALLY SET TO #SFF BEFORE SIMULATOR OPERATION.*

<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00F1</td>
<td>FFFF</td>
<td>DECIT DC 2 DOUBLE PRECISION #1</td>
</tr>
<tr>
<td>00F3</td>
<td>0100</td>
<td>INCIT DC 2 DOUBLE PRECISION 1</td>
</tr>
<tr>
<td>00F5</td>
<td>00</td>
<td>FLAG DC #00 USED BY SIMULATOR</td>
</tr>
<tr>
<td>00F6</td>
<td>00</td>
<td>DEST DC #00 DESTINATION INDEX</td>
</tr>
<tr>
<td>00F7</td>
<td>00</td>
<td>DEST DC #00 DESTINATION DATA</td>
</tr>
<tr>
<td>00F8</td>
<td>00</td>
<td>SRC DC #00 SOURCE INDEX</td>
</tr>
<tr>
<td>00F9</td>
<td>00</td>
<td>DC #00 #KIM* DISPLAY BUFFER</td>
</tr>
<tr>
<td>00FA</td>
<td>FA</td>
<td>POINTL DC #5A #KIM* DISPLAY POINTER</td>
</tr>
<tr>
<td>00FB</td>
<td>09</td>
<td>POINTH DC #00 #KIM*</td>
</tr>
<tr>
<td>00FC</td>
<td>0100</td>
<td>PNT DC #00 DATA POINTER</td>
</tr>
<tr>
<td>00FE</td>
<td>0001</td>
<td>SCR DC #00 SCRATCHPAD REGISTER</td>
</tr>
</tbody>
</table>
**MAIN CONTROL LOOP**

1780

**ORO**

$1780

****MAIN CONTROL LOOP****

1780 BFF MNSTR TXIX $FF

1782 9A

TXIX

1783 20 9117 SIM80

JSR MAIN

1786 A5 E4

LDAX PSW

1788 29 D7

ANDIM $07

178A 09 02

ORAIM $02

178C 05 E4

STAZ PSW

178D 58

EXIT

178F 60

RTS

1790 EA

NOP

****VARIATIONS FOR MINIMUM CONFIGURATION*******

*178E 4C 221C

JMP MONITOR

**KIM** REGISTER SINGLE STEP

*178E 4C D1C

JMP PCCMD

**KIM** PC SINGLE STEP

*178E 58

CLI

**FULL SPEED RUN**

*178F 0E EF

BNE MNSTR

****VARIATION FOR INTERRUPT ACTION*******

*179F 20 1760

JSR INT

**GO CHECK FOR INTERRUPT**

****VARIATION FOR INTERRUPT ACTION*******

17A2 20 5303

JSR INCPC

**NO, INCREMENT PC**

17A5 B2 02

ITR

17A7 A5 FE

LDAX INST

**RECOVER INSTRUCTION**
17A9 48 DIV PHA  **ITEM TEMP SAVE**
17AA 29 07 ANDIM #307 **IMASK FOR LS3 BITS**
17AC 49 07 EORIM #307 **REVERSE 'EM**
17AE 95 F6 STA ZX DEST+1 **SAVE 'EM**
17B0 68 PLA **RECOVER TEMP**
17B1 4A LSRA **SHIFT TO NEXT FIELD**
17B2 4A LSRA
17B3 4A LSRA
17B4 CA DEX **IDEC COUNTER/INDEX**
17B5 D0 F2 RNE DIV **LOOP TILL DONE**
17B7 86 F5 STXZ FLAG **CLEAR FLAG**
17B9 A8 02 LDYIM INPG2 **SET FOR 2ND INTERP PAGE**
17BB 4A LSRA **TEST FOR QUADRANT**
17BC 80 06 BCS IMI
17BE F0 0E BEQ 1ST
17C0 A9 11 TMARL LDAIM #311 **NO, ARILOG**
17C2 65 F7 ADCZ DEST
17C4 AA TAX
17C5 A9 00 LDAIM (A) **SET DEST = A**
17C7 85 F7 STA ZX DEST
17C9 F0 07 IMI BEQ MOVE
17CB BB 2ND DEY **CHANGE PAGES**
17CC A9 07 LDAIM #307 **OFFSET FOR 2ND**
17CE 65 F8 1ST ADCZ SRC **IXFER TO INDEX**
17D0 AA TAX
17D1 E8 INX
17D2 98 MOVE TYA **SET HIGH ADDRESS**
17D3 48 PHA
17D4 BD E303 LOAX XFRBL **GET LO ADDR**
17D7 48 PHA **SET LO ADDRESS**
17D8 A6 F8 LDIZ SRC **GET SOURCE INDEX**
17DA B4 E5 LDIZX REGS **GET SOURCE DATA IN Y**
17DC A6 F7 LDIXZ DEST **GET DEST INDEX**
17DE B5 E5 LDIZX REGS
17E0 85 F6 STAZ DSTDATA **SAVE DESTINATION DATA**
17E2 A9 4A LDAIM #344 **SET 6502 STATUS**
17E4 48 PHA
17E5 8A TXA **DEST IND. IN A & X**
17E6 40 RTI **IGO INTERPRET**
Org $0100

0100 4A IN L3RA
0101 F0 06 BEQ XCHG
0103 90 33 BCC INPUT
0105 A8 TAY
0106 4C 7A03 JMP CALL65 CALL 6502 SUBROUTINE
0109 20 6303 XCHG JSR RP/SCR MOVE HL TO SCRATCH
010C A2 04 LDXIM DE POINT TO DE
010E A0 02 LDYIM HL AND HL
0110 20 6503 JSR RP/HP MOVE DE TO HL
0113 A2 04 LDXIM DF POINT TO DE
0115 4C 7003 JMP SCR/HP MOVE SCRATCH TO DE

********I/O, ETC, ENTRY********

0127 4A I/O LSRA
0128 DB 03 BNE LBL
012A 86 E3 STXZ INTE
012C 60 RTS

012D A2 02 LBL LDXIM HL POINT TO HL
012F 90 CF RCC IN IS IT INPUT?
0131 4A LSRA NO, IS IT XTHL?
0132 F0 E4 REQ XTHL YES
0134 RB 58 RCS JUMPNO NO, JUMP?
0136 C6 F5 OUTPUT DECZ FLAG NO, FLAG AN OUTPUT
0138 20 AF03 INPUT JSR IMM GET PORT #
013B A5 FE LOAZ SCR
013D 9A ASLA
013E AA TAX
013F B0 FC03 IOPLP LDAX IOTBL GET I/O ADDRESS
0142 99 F800 STAY PNT-1 STORE IN POINTER
0145 E8 INX BUMP INDEX
0146 88 DEY DECREMENT COUNTER
0147 DB F5 BNE IOPLP
0149 C8 INY
014A A5 F5 LOAZ FLAG GET INPUT OR OUTPUT DATA
014C 91 FC STA0Y PNT SET DATA DIR, REGISTER
014E 88 DEY POINT TO PORT
014F 4A LSRA INPUT?
0150 98 06 BCC IOIN YES, GO DO IT
0152 A5 E5 LOAZ A NO, OUTPUT A
0154 91 FC STA0Y PNT
0156 80 04 BCS IOEX
0158 B1 FC IOIN LOAIY PNT INPUT FROM PORT
015A 85 E8 STAZ A
015C 60 IOEX RTS
******ARILOG IMMEDIATE ENTRY******

015D 20 AB03 ARIM JSR IMM GET IMMEDIATE BYTE
0160 18  CLC
0161 4C C017 JMP IMARL GO DO ARITH ROUTINE

******PUSH ENTRY******

0164 4A 4A  PUSH LSRA IS THIS A PUSH?
0165 B0 07  BCS SKP YES, GO PUSH
0167 C9 03  CMPIM #03 NO, IS THIS A CALL?
0169 F0 27  BEQ CALLUN YES, GO CALL
016B 4C 1102  JMP UNDEF NO, UNDEFINED

016E 9A 9A  SKP ASLA MAKE AN INDEX
016F AA  AA  TAX IS IT PUSH PSW?
0170 00 01  BNE PUSHMT INO, KEEP INDEX
0172 CA CA  DEX YES, ADJUST INDEX
0173 8A 8A  PUSHMT TXA JTEMP SAVE
0174 48 48  PHA
0175 20 9703  JSR DECSS DECUREMENT STACK POINTER
0178 20 7603  JSR SP/PNT IT BECOMES POINTER
017B 68 68  PLA IRECOVER TEMP SAVE
017C AA  AA  TAX
017D 40 01  RPLP/LDYILovIM #91 ICLEAR INDEX
017F 85 E6  RPLP/LDAZX REGS+1 IGET NEXT RP DATA
0181 91 FC  STAIY PNT ISTORE IN MEMORY
0183 CA CA  DEX
0184 88 88  DEY
0185 F0 F8  BEQ RPLP
0187 60 60  RTS RTS

******JUMP ENTRY******

0188 88 88  JMP CLV INDICATE A JUMP

******CALL ENTRY******

0189 20 0002 CALL JSR CONDIT ITEST CONDITION
018C F0 04  BEQ CALLUN IMET
018E 4C 8003  JMP DBLINC INOT MET

0191 88 88  JUMPUN CLV INDICATE A JUMP
0192 20 5003 CALLUN JSR PC/PNT IGET NEXT 2 BYTES
0195 A2 19  LOXIM (SCR) INTO SCRATCH
0197 20 A003  JSR MEM/RP
019A 50 08  BVC JM JJUMP
019C 20 8003  JSR DBLINC JBUMP PC

019F A2 0A  SVR LOXIM (PC) ISAVE RETURN
01A1 20 7301  JSR PUSHMT JON STACK
01A4 A2 8A JM LOXIM (PC) JPOINT TO PC
01A6 4C 7003  JMP SCR/RP IGIVE IT NEW ADDRESS
********RETURN ENTRY*******

01A9 20 0002 RETURN JSR CONDIT TEST CONDITION
01AC D0 D9 BNE RTS NOT MET
01AE A2 0A RETURN LDXIM (PC) POP RETURN OFF
01B0 D0 14 BNE POPIT STACK INTO PC

********RESET ENTRY*******

01B2 A5 FE PST LOAD INST CONVERT INSTRUCTION
01B4 29 38 ANDIM $38 TO RESET VECTOR
01B6 85 FE STA SI STORE IT IN SCRATCH
01B8 A9 00 LDAIM RSTHI HIGH RESET VECTOR
01BA 85 FF STA SI SCR+1
01BC 90 E1 BCC SVR

********POP ENTRY*******

01BE 4A POP LSRA $POP?
01BF 90 17 BCC OTH $NO
01C1 0A ASLA
01C2 AA TAX $POP PSW?
01C3 D0 01 BNE POPIT $NO, KEEP INDEX
01C5 CA OEX YES, ADJUST INDEX
01C6 8A POPIT TXA $TEMP SAVE INDEX
01C7 48 PHA
01C8 20 7603 JSR 3P/PNT $SP IS POINTER
01C9 68 PLA $RECOVER TEMP
01CC AA TAX
01CD 20 A003 JSR MEM/RP $PULL OUT STACK DATA

01D0 20 0301 INCSP JSR ISP $BUMP SP
01D3 A2 08 ISP LDXIM (SP)
01D5 4C 8503 JMP INC
01D8 A2 02 OTH LDXIM (HL) $POINT TO HL
01DA A0 08 LDYIM (SP) $ASSUME SPHL
01DC 0A ASLA
01DD 00 BEQ SPHL YES, GO DO IT
01DF C9 04 CMPIM #$04 $NO, WHAT IS IT?
01E1 F0 2E BEQ UNDEF UNDEFINED
01E3 10 C9 BPL RETURN $RETURN
01E5 A0 0A LDYIM (PC) $PCHL
01E7 4C 6503 SPHL JMP RP/RP

**********TABLE OF PSW MASKS**********

01EA 00 MSKTBTL DC SMASK $SIGN BIT MASK
01EB 04 FOUR DC PMASK $PARITY BIT MASK
01EC 01 DC CHASK $SCARRY BIT MASK
01ED 40 DC ZMASK $ZERO BIT MASK

*******************END*******************

*PAGE 1 RAM MEMORY ADDRESSES $301F0 TO $301FF MUST BE RESERVED FOR STACK*
0200 4A CONDIT LSRA
0201 AA TAX
0202 RD EA01 LDAX MSKTRL GET CONDITION FLAG MASK
0205 25 E4 ANDZ PSW IS CONDITION SET?
0207 R0 03 RCS RTN LEAVE Z FLAG SET
0209 5D EA01 EORX MSKBL FOR REVERSE IT
020C 60 RTN RTS

020D A5 FE NOP LDAA INST CHECK INSTRUCTION
020F F0 FB 8EQ RTN IF NOP, JUST RETURN
0211 4C 221C UNDEF JMP *KIM* GO TO SYSTEM MONITOR

0214 4A DADLXI LSRA DROP LSB
0215 0A ASLA
0216 D0 02 NPE NSP USE SPT
0218 A9 08 LDAIM (SP) YES, POINT TO IT
021A 48 NSP PHA SAVE POINTER
021B 0A TXA GET DESTINATION INDEX
021C 4A LSRA LOAD?
021D 80 12 BCS LXI NO, GO DO LXI
021F 68 DAD PLA YES, RECOVER POINTER
0220 AB TAY
0221 A2 02 LDXIM (HL) SET HL AS DESTINATION
0223 20 0703 JSR INDFC GO ADD RP & HL

0226 A8 CARRY TAY SAVE RESULT
0227 2A ROLA BRING IN CARRY
0228 45 F5 EORZ FLAG CHG TO BORROW IF SUBTRACT
022A 46 E4 LSRZ PSW
022C 4A LSRA
022D 26 E4 ROLZ PSW RESTORE CARRY(BORROW)
022F 98 TYA RECOVER RESULT
0230 68 RTS
0231 20 5D03 LXI JSR PC/PNT PC INTO PNT
0234 60 PLA IRECOVER POINTER
0235 AA TAX USE TO INDEX DESTINATION RP
0236 20 A003 JSR MEM/RP MOVE IMMEDIATE DATA TO RP
0239 4C 6003 JMP DUBLINC BUMP PROGRAM COUNTER
**LOAD & STORE ENTRY**

<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Instruction</th>
<th>Flags</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>023C</td>
<td>2C EB01</td>
<td>LDSTR BIT</td>
<td>FOUR</td>
<td>!INDIRECT?</td>
</tr>
<tr>
<td>023F</td>
<td>F0 08</td>
<td>BEQ DIRECT</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>0241</td>
<td>4A 23</td>
<td>LSRA ACC</td>
<td>LOAX</td>
<td>!STAX?</td>
</tr>
<tr>
<td>0242</td>
<td>A5 E5</td>
<td>STAX LOAD</td>
<td>A</td>
<td>!YES, STORE A</td>
</tr>
<tr>
<td>0246</td>
<td>81 E4</td>
<td>STAIX REGS+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0248</td>
<td>60</td>
<td>RTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0249</td>
<td>40</td>
<td>DIRECT PHA</td>
<td></td>
<td>!TEMP SAVE</td>
</tr>
<tr>
<td>024A</td>
<td>20 5D03</td>
<td>JSR PC/PNT</td>
<td></td>
<td>!PC TO POINTER</td>
</tr>
<tr>
<td>024D</td>
<td>20 8903</td>
<td>JSR DBLINC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0250</td>
<td>A2 19</td>
<td>LDXIM SCR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0252</td>
<td>20 A003</td>
<td>JSR MEM/RP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0255</td>
<td>A2 17</td>
<td>LDXIM PNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0257</td>
<td>20 7003</td>
<td>JSR SCR/RP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>025A</td>
<td>68</td>
<td>PLA</td>
<td></td>
<td>!RECOVER TEMP</td>
</tr>
<tr>
<td>025B</td>
<td>4A 0E</td>
<td>LSRA BCS</td>
<td>STORE</td>
<td>!NO, STORE</td>
</tr>
<tr>
<td>025C</td>
<td>80 0E</td>
<td>STORE A</td>
<td></td>
<td>!LOAD?</td>
</tr>
<tr>
<td>025E</td>
<td>0A</td>
<td>LOAD ASLA</td>
<td>!MAKE INDEX</td>
<td></td>
</tr>
<tr>
<td>025F</td>
<td>AA</td>
<td>TAX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0260</td>
<td>F8 03</td>
<td>BEQ LDAD</td>
<td>!LOAD A DIRECT?</td>
<td>!NO, LOAD HL DIRECT</td>
</tr>
<tr>
<td>0262</td>
<td>4C A003</td>
<td>JMP MEM/RP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0265</td>
<td>A2 17</td>
<td>LDAD LDXIM</td>
<td></td>
<td>!POINT TO POINTER</td>
</tr>
<tr>
<td>0267</td>
<td>A1 E5</td>
<td>LDXAIX REGS</td>
<td>!LOAD A</td>
<td>A</td>
</tr>
<tr>
<td>0269</td>
<td>85 E5</td>
<td>STAIX REGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>026B</td>
<td>60</td>
<td>RTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>026C</td>
<td>0A</td>
<td>STORE ASLA</td>
<td>!MAKE INDEX</td>
<td></td>
</tr>
<tr>
<td>026D</td>
<td>AA</td>
<td>TAX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>026E</td>
<td>F0 03</td>
<td>BEQ STAD</td>
<td>!STORE A DIRECT?</td>
<td>!NO, STORE HL DIRECT</td>
</tr>
<tr>
<td>0270</td>
<td>4C 7D01</td>
<td>JMP RP/MEM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0273</td>
<td>A2 18</td>
<td>STAD LDXIM</td>
<td></td>
<td>!POINT TO POINTER</td>
</tr>
<tr>
<td>0275</td>
<td>DB CD</td>
<td>BNE STAX</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
******INX/DCX ENTRY******

0277 A0 0C INXDCX LDYIM (DECIT) JSET FOR DECREMENT
0279 4A LSRA
027A 90 02 BCC DRP JYES
027C A0 0F LDYIM (INCIT) JNO, SET FOR INCR
027E 0A DRP ASLA JDROP LSB
027F D0 02 BNF NOR JFOR SP?
0281 A9 08 LDAIM (SP) JYES, POINT TO SP
0283 AA NOR TAX JUSE AS INDEX
0284 4C 8703 JMP INCDFC

******DCR ENTRY******

0287 38 DCR SEC JSET FOR DECREMENT
0288 C6 F5 DECZ FLAG

******INR ENTRY******

028A A0 01 INR LDYIM SINC JSET FOR INCREMENT
028C 20 B003 BTH JSR ADDR JGO ADD
028F 4C 3803 JMP STATUS JSET STATUS, NOT CARRY

******MOVE IMMEDIATE ENTRY******

0292 20 AB03 MVI JSR IMM JGET IMMEDIATE BYTE
0295 4C D217 JMP MOVE JMOVE IT TO DESTINATION

******MOVE ENTRY******

0298 98 MOVIT TYA JGET SOURCE DATA
0299 CA MVIT1 DEX JIS DEST MEMORY?
029A D0 02 BNE NMEM JNO
029C 81 E7 STAIX HL JYES, STORE IN MEMORY
029E 95 E6 NMEM STAZX REGS+1 JSTORE IN DEST REGISTER
02A0 60 *3 RTS
**ARITHMETIC ROUTINES**

*******ADC ENTRY*******

```
02A1 88 ADC  CLV  BVC  ADD  $FLAG W/CARRY
02A2 50 88 ADC  CLV  BVC  ADD
```

*******CMP ENTRY*******

```
02A4 A2 19 CMP  LDXIM (SCR)  JCHG DEST TO SCR
02A6 D0 01 BNE  SUB  IS0 ONLY PSW IS AFFECTED
```

*******SBB ENTRY*******

```
02A8 B8 SBB  CLV  $FLAG W/BORROW
```

*******SUB ENTRY*******

```
02A9 C6 F5 SUB  DECZ  FLAG  $FLAG = FF FOR SUBTRACT
02AB 38 SEC
```

*******ADD ENTRY*******

```
02AC 4C 3203 ADD  JMP  AD/CRY
```

***LOGICAL OPERATIONS***

******ORA ENTRY*******

```
02AF 98 ORA  TYA  $GET SOURCE DATA
02B0 05 E5 ORAZ  A  $LOGICAL OR
02B2 90 03 BCC  SOM  $SET FOR A CLEAR AUX CRY
```

******XRA ENTRY*******

```
02B4 98 XRA  TYA  $GET SOURCE DATA
02B5 45 E5 EORZ  A  $LOGICAL EXCLUSIVE OR
02B7 A0 00 SOM  LDYIM #S00  $00 WON'T SET ANY FLAGS
02B9 90 08 BCC  ALL  $GO SAVE RESULT
```

******ANA ENTRY*******

```
02BB 98 ANA  TYA  $GET SOURCE DATA
02BC 48 PHA  $SAVE IT
02BD 05 E5 ORAZ  A  $LOGICAL OR OF BIT
02BF 0A A$LA  ANDIM #S10  $THREE BECOMES AUX CARRY
02C0 29 10 TAY
02C2 A8
02C3 68 PLA
02C4 25 E5 ANDZ  A  $GET SOURCE DATA
02C6 84 FF ALL  STYZ  SCR*1  $SAVE FLAG SETTING DATA
02C8 85 E5 STAZ  A  $LOGICAL RESULT INTO A
02CA 20 3803 JSR  STATUS  $SET STATUS
02CD A9 EE LDAIM #SEE  $SELECT STATUS BITS TO CLEAR
02CF 25 E4 CY&AC ANDZ  PSW  $CLEAR SELECTED BITS
02D1 05 FF ORAZ  SCR+1  $SET SELECTED BITS
02D3 85 E4 STPSW STAZ  PSW  $SAVE NEW PSW
02D5 60 RTS
```
**ROTA T E, ETC. ENTRY******

02D6  A4 E5  ROTATE  LDYZ  A  !GET ACCUMULATOR
02D8  4A  !IS THIS CM C OR STC?
02D9  D0 0A  BNE  ROCOMP  !NO, IT IS A ROTATE OR CMA

02DB  A5 E4  LDAZ  PSW  !YES, CHANGE CARRY
02DD  49 01  FORIM  #$01  !CMC
02DF  90 02  BCC  STPS  !STC
02E1  09 01  ORAAM  #$01  !STC
02E3  70 EE  STPS  RVS  STPSW

02E5  B0 13  ROCOMP  BCS  LEFT  !LEFT OR RIGHT?
02E7  4A  LSRA  !CMC?
02EB  D0 06  BNE  ROT  !NO, ROTATE
02E9  90  TYA  !YES, COMPLEMENT A
02EE  49 FF  FORIM  #$FF  !DOESN'T SET STATUS
02F0  6A  RTS

02F1  B0 02  BCS  RRC  !GET 8080 CARRY
02F3  A5 E4  LDAZ  PSW  !GET 8080 CARRY
02F5  90  TYA  !GET ACCUMULATOR
02F6  6A  RORA  !ROTATE RIGHT
02F7  70 00  BVS  JCRIY  !GO STORESET CARRY

02FA  4A  LEFT  LSRA  !DECIMAL ADJUST?
02FB  F0 0F  BEQ  DAA  !YES, GO DO IT
02FC  98  TYA  !NO, GET ACCUMULATOR
02FE  B0 04  BCS  RLC  !RLC?
0300  A5 E4  LDAZ  PSW  !NO, GET 8080 CARRY
0302  6A  RORA  !MOVE IT INTO MSNIB
0303  6A  RORA
0304  9A  RLC  ASLA  !MOVE IT INTO CARRY
0305  98  TYA  !GET ACCUMULATOR
0306  2A  ROLA  !MOVE IT LEFT
0307  85 E5  JCRIY  STAIZ  A  !SAVE IT
0309  4C 26 02  JMP  CARRY  !GO SET STATUS

030C  18  DAA  CLC  !PRESERVE STATUS
030D  08  PHP  !GET SOURCE DATA
030E  98  TYA  !GET SOURCE DATA
030F  EA  NOP
0310  88 F6  STAIZ  DESTDA  !PREP FOR ADD
0312  29 0F  ANIIM  #$0F  !LOOK AT LSNIB
0314  69 06  ADIM  #$06  IF IF 28A WILL CAUSE AUX CRY
0316  05 E4  ORAAM  PSW  !YES, IF AC IS ALREADY SET
0318  29 10  ANIIM  #$10  !NEITHER SET?
031A  F0 02  BEQ  NOSIX  !NO, DON'T ADJUST LSNIB
031C  A9 06  LDAIM  #$06  !YES, ADJUST IT
031E  A0  NOSIX  TAY
PAGE 15 8080 SIMULATOR (C) 1978 BY DANN MCCREARY

031F 65 F6 ADCZ DESTDA #GET SOURCE
0321 80 08 BCS SXY #GET CARRY
0323 69 60 ADCIM #$60 #IS MSNIB NOW == A0?
0325 2A ROLA #IF SO, CARRY IS SET
0326 05 E4 ORAZ PSW #IS EITHER SET?
0328 4A LSR #YES, ADJUST MSNIB
0329 90 04 BCC DA #NO, DON'T ADJUST MSNIB
032B 98 SXY TYA #$5F + CARRY = 60
032C 69 5F ADCIM #$5F #DESTINATION IS A
032E A8 TAY
032F A2 00 DA LDXIM #$00 #DESTINATION IS A
0331 28 PLP #RESTORE STATUS
0332 20 8D03 AD/Cry JSR ADDR
0335 20 2602 JSR CARRY

*******STATUS ROUTINE*******
*SETS ZERO, SIGN AND PARITY, LEAVES CARRY AND AC*

0338 A8 STATUS TAY #SAVE RESULT
0339 26 E4 ROLZ PSW #CLEAR PSW SIGN BIT
033B 0A ASLA #INPUT NEW SIGN IN CARRY
033C 85 F5 STAZ FLAG #CLEAR LSB OF FLAG
033E A5 E4 LDZ PSW #INPUT NEW SIGN IN PSW
0340 6A RORA
0341 49 46 SGW ORAIM #$46 #PRESERVE Z, P & PRESET

0343 AA TAX #SAVE IN X
0344 98 TYA #RECOVER WORD
0345 F0 13 BEQ DONE #IF ZERO, ALL DONE

0347 E6 F5 FLIP INCZ FLAG #FLIP FLAG
0349 4A PAR LSR #TEST EACH BIT
034A F0 02 BEQ ALL #NO MORE BITS
034C 90 FB BCC PAR #INPUT NEW SIGN IN CARRY
034E 80 F7 ALL BCS FLIP

0350 46 F5 LSRZ FLAG #TEST FLAG
0352 8A TXA #RECOVER PSW
0353 29 BF ANDIM #$BF #CLEAR Z
0355 B0 02 BCS REC #PARITY EVENT?
0357 29 F8 ANDIM #$F8 #NO, CLEAR P
0359 AA REC TXA #BACK TO X

035A 86 E4 DONE STXZ PSW #STORE AS PSW
035C 60 RTS
****SUBROUTINES****

***MOVE REGISTER PAIRS***

0350 A2 0A PC/PNT LDIMX (PC) \(\text{SOURCE IS PC}\)

035F A0 17 RP/PNT LDYIM (PNT) \(\text{DESTINATION IS PNT}\)

0361 D0 02 BNE RP/RP JGO TRANSFER

0363 A0 19 RP/SCR LDYIM (SCR) \(\text{DESTINATION IS SCR}\)

0365 B5 E5 RP/RP LOAZX REGS \(\text{GET LOW ORDER SOURCE}\)

0367 99 E500 STAY REGS \(\text{STORE IN LOW DESTINATION}\)

036A B5 E6 LDYIM (PNT) \(\text{DESTINATION IS PNT}\)

036C 99 E600 STAY REGS+1 \(\text{STORE IN HI DESTINATION}\)

036F 60 RTS

0370 A8 SCR/RP TXA \(\text{RP IS DESTINATION}\)

0371 A8 TAY

0372 A2 19 LDYIM (SCR) \(\text{SCR IS SOURCE}\)

0374 D0 EP BNE RP/RP

0376 A2 08 SP/PNT LDYIM (SP) \(\text{SP IS SOURCE}\)

0378 D0 E5 BNE RP/PNT \(\text{UNCONDITIONAL BRANCH}\)

******CALL 6502 SUBROUTINE******

***CALL SUBROUTINE ADDRESS -1***

037A B1 EF CALL65 LDAIY PC \(\text{GET ADDRESS}\)

037C 48 PHA

037D 88 DEY \(\text{POINT TO LO CALL}\)

037E 10 FA BPL CALL65 \(\text{DONE?}\)

******INC, DEC, OR ADD REGISTER PAIRS******

0380 20 0303 DBCINC JSR INCPC \(\text{INC PC TWICE}\)

0383 A2 0A INCPC LDIMX (PC) \(\text{INC PC}\)

0385 A0 0E INC LDYIM (INCIT) \(\text{POINT TO INC DATA}\)

0387 18 INCDEC CLC

0388 B5 E5 INDE LOAZX REGS \(\text{GET}\)

038A 79 E500 A0CY REGS \(\text{ADD}\)

038D 95 E5 STAIX REGS \(\text{SAVE}\)

038F C8 INY \(\text{BUMP INDEXES}\)

0390 E8 INX

0391 98 TYA \(\text{CHECK INDEX}\)

0392 29 01 ANDIM #001 \(\text{LAST PASS}\)

0394 D0 F2 BNE INDE JNO

0396 60 RTS \(\text{YES}\)

0397 20 9A03 DCSAP JSR DSP \(\text{DEC SP TWICE}\)

039A A2 08 DSP LDIMX (SP) \(\text{DEC SP}\)

039C A0 0C DEC LDYIM (DECIT) \(\text{POINT TO DEC DATA}\)

039E D0 E7 BNE INCDEC \(\text{GO DO IT}\)
**MOVE MEMORY TO REG PAIR**

03A0 A0 01 MEM-RP LDYIM #$01 MOVE DATA IN MEMORY
03A2 R1 FC MRLP LOAIX PNT 1GET NEXT BYTE
03A4 95 E6 STAIX REGS+1 1STORE IN DESTINATION
03A6 CA DEX 1BUMP INDICES
03A7 88 DEY
03A8 F0 F0 BEQ MRLP
03AA 60 RTS

**FETCH IMMEDIATE BYTE**

03AB A2 00 IMM LDYIM #$00 1GET BYTE POINTED AT
03AD A1 EF LOAIX PCL 1BY PROGRAM COUNTER
03AF 85 FE STAIX SCR 1SAVE IT IN SCR
03B1 A9 19 LOAIX (SCR) 1MAKE SCR THE SOURCE INDEX
03B3 85 FB STAIX SRC
03B5 20 B303 JSR INCPC 1BUMP PC
03B8 A2 00 LOAIM #$00
03BA A0 02 LOAIM INPG2 1POINT TO INTERP PAGE 2
03BC 60 RTS

**ADD CONTENTS OF Y & DESTDA**

03BD 98 ADDR TYA 1GET SOURCE DATA
03BE 45 F5 EORZ FLAG 1CHANGE IF SUBTRACT
03C0 A8 TAY
03C1 29 0F ANDIM #$0F 1SAVE LSNib
03C3 85 FE STAIX SCR
03C5 2A ROL A 1GET CARRY IN A
03C6 70 02 BV3 W/OCRY 1ADD/SUB W/CARRY?
03C8 45 E4 EORZ PSW 1YES, CHG PER 8080 CARRY
03CA 48 W/OCRY PHA 1SAVE FOR LATER
03CB 4A LSRA 1GET CARRY.
03CC A5 F6 LOAZ DESTDA 1GET ACCUM DATA
03CE 29 0F ANDIM #$0F 1LOOK AT LSNib
03D0 65 FE ADCZ SCR 1ADD TO SOURCE LSNib
03D2 29 18 ANDIM #$18 1WAS THERE AN AC?
03D4 85 FF STAIX SCR+1 1SAVE AC
03D6 A9 EF LOAIX #$EF 1CLEAR AC
03DB 20 CF02 JSR CY&AC
03DB 68 PLA 1GET CARRY BACK
03DC 4A LSRA
03DD 98 TYA 1GET SOURCE DATA
03DE 65 F6 ADCZ DESTDA 1ADD TO ACCUMULATOR
03EE 4C 9902 JMP MVIT1 1GO PUT RESULT IN DESTINATION
**ENTRY ADDRESS TABLE FOR TRANSFER**
**TO INSTRUCTION INTERPRETATION**

<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>03E3</td>
<td>98</td>
<td>XFRTB, MOVIT ENTRIES ON 2ND INTRP, PAGE</td>
</tr>
<tr>
<td>03E4</td>
<td>08</td>
<td>DC ROTATE</td>
</tr>
<tr>
<td>03E5</td>
<td>92</td>
<td>DC MVI</td>
</tr>
<tr>
<td>03E6</td>
<td>87</td>
<td>DC DCR</td>
</tr>
<tr>
<td>03E7</td>
<td>8A</td>
<td>DC INR</td>
</tr>
<tr>
<td>03E8</td>
<td>77</td>
<td>DC INXDCX</td>
</tr>
<tr>
<td>03E9</td>
<td>3C</td>
<td>DC LDSTR</td>
</tr>
<tr>
<td>03EA</td>
<td>14</td>
<td>DC DADLXI</td>
</tr>
<tr>
<td>03EB</td>
<td>00</td>
<td>DC NOP</td>
</tr>
<tr>
<td>03EC</td>
<td>02</td>
<td>DC RST ENTRIES ON 1ST INTRP, PAGE</td>
</tr>
<tr>
<td>03ED</td>
<td>5D</td>
<td>DC ARIM</td>
</tr>
<tr>
<td>03EE</td>
<td>64</td>
<td>DC PUSH</td>
</tr>
<tr>
<td>03EF</td>
<td>89</td>
<td>DC CALL</td>
</tr>
<tr>
<td>03F0</td>
<td>27</td>
<td>DC I/O</td>
</tr>
<tr>
<td>03F1</td>
<td>88</td>
<td>DC JMP</td>
</tr>
<tr>
<td>03F2</td>
<td>8E</td>
<td>DC POP</td>
</tr>
<tr>
<td>03F3</td>
<td>A9</td>
<td>DC RETURN</td>
</tr>
<tr>
<td>03F4</td>
<td>A4</td>
<td>DC CMP ENTRIES ON 2ND INTRP, PAGE</td>
</tr>
<tr>
<td>03F5</td>
<td>AF</td>
<td>DC ORA</td>
</tr>
<tr>
<td>03F6</td>
<td>84</td>
<td>DC XRA</td>
</tr>
<tr>
<td>03F7</td>
<td>BB</td>
<td>DC ANA</td>
</tr>
<tr>
<td>03F8</td>
<td>AB</td>
<td>DC SBB</td>
</tr>
<tr>
<td>03F9</td>
<td>A9</td>
<td>DC SUB</td>
</tr>
<tr>
<td>03FA</td>
<td>A1</td>
<td>DC ADC</td>
</tr>
<tr>
<td>03FB</td>
<td>AC</td>
<td>DC ADD</td>
</tr>
</tbody>
</table>

**INPUT/OUTPUT PORT ADDRESS**

**DEFINITION TABLE**

*ENTRIES MUST BE IN NORMAL ORDER, IE, MOST SIGNIFICANT ADDRESS BYTE FIRST*

<table>
<thead>
<tr>
<th>Address</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>03FC</td>
<td>1700</td>
<td>IOTBL DC 2, #30017 18080 PORT 0 - KIM PORT A</td>
</tr>
<tr>
<td>03FE</td>
<td>1740</td>
<td>DC 2, #44017 18080 PORT 1 - KIM PORT SA</td>
</tr>
</tbody>
</table>
***8080 DEMONSTRATION PROGRAM***

*******TIME-OF-DAY CLOCK*******

```
0000 31 7400 DEMO LXI SP #$0074 SET STACK POINTER
0003 F3 DI DISABLE INTERRUPTS
0004 00 NOP ING OPERATION
0005 C3 0900 JMP TIME
0008 F1 INTIME POP PSW POP UNUSED RETURN OFF STACK
0009 B7 TIME ORA A CLEAR CARRY
000A 79 MOV A,C UPDATE SECONDS
000C 3C INR A
000D 27 DAA
000E 4F MOV C,A
0010 FA 2B00 CPI #$60 MINUTE UPDATE?
0013 0E 00 MOV A,C IUPDATE SECONDS
0015 78 MOV A,E UPDATE MINUTES
0016 3C INR A
0017 27 DAA
0018 5F MOV E,A
0019 FE 60 CPI #$60 HOUR UPDATE?
001B FA 2B00 JMP SHOW NOT NOW
001E 1E 00 MVI E,#$30 YES, CLEAR MINUTES
0020 7A MOV A,D UPDATE HOURS
0021 3C INR A
0022 27 DAA
0023 57 MOV D,A
0024 FA 13 CPI #$13 USE #$24 FOR 24 HOUR CLOCK
0025 FA 2B00 JMP SHOW NOT NOW
0029 16 01 MVI D,#$81 USE #$80 FOR 24 HOUR CLOCK
002B 2A 8000 SHOW LXI H,POINTH *KIM*
002E 72 MOV H,D MOVE TIME TO DISP, BUFFER
002F 28 DCX H
0030 73 MOV M,E
0031 28 DCX H
0032 71 MOV M,C
0033 66 8B NXTKY MVI B,#$80 SET TIME DELAY VALUE

*******ALTERNATE FOR INTERRUPT TIMEBASE*******

*0033 06 37 NXTKY MVI B,#$37 IOEBOUNCE INTERRUPT*

***************************************************************************
**PAGE 20 8080 SIMULATOR (C) 1978 BY DAN MCREARY**

0035  CB 6500  WAIT  C65  DSKBD=1  16502 DISPLAY&KEYBOARD SUB
0038  FE 0A  CPI  #50A  INUMERIC KEY PRESSED?
003A  FA 4500  JM  SET  YES, GO SET TIME
003D  05  DCR  B  NO, COUNT DOWN DELAY
003E  C2 3500  JNZ  WAIT  KEEP WAITING
0041  FB  EI  ENABLE INTERRUPTS
0042  C3 0900  JMP  TIME  GO SET TIME

*******ALTERNATE FOR INTERRUPT TIMEBASE*******

*0042  C3 3500  JMP  WAIT  KEEP WAITING FOR INTERRUPT*

********************

0045  47  SET  MOV  B,A  SAVE INPUT IN B
0046  FB  XCHG  SHIFT THE TIME ONE DIGIT LEFT
0047  29  DAD  H  BY ADDING HL TO ITSELF
0048  29  DAD  H
0049  29  DAD  H
004A  29  DAD  H
004B  EB  XCHG

004C  79  MOV  A,C  AND ROTATING THE SECONDS LEFT
004D  97  RLC
004E  97  RLC
004F  97  RLC
0050  97  RLC

0051  4F  MOV  C,A
0052  E6 0F  ANI  #0F
0054  B3  ORA  E
0055  5F  MOV  E,A

0056  79  MOV  A,C
0057  E6 F0  ANI  #F0
0059  B0  ORA  B  INSERT NEW DIGIT
005A  4F  MOV  C,A

005B  CB 6500  DBCN  C65  DSKBD=1  WAIT FOR KEY RELEASE
005E  FE 15  CPI  #15  RELEASED?
0060  FA 5B00  JM  DBCN  NOT YET
0063  C3 2B00  JMP  SHOW  YES, GO SHOW TIME

*******6502 SUBROUTINE CALLED FROM 8080 PROGRAM*******
*******(CALL MUST BE TO SUBROUTINE ADDRESS = 1)*******

0066  20 1F1F  DSKBD  JSR  SCANDS  *KIM* DISPLAY BUFFER
0069  20 6A1F  JSR  GETKEY  *KIM* KEYPAD
006C  85 E5  STA  A  RESULT TO 8080 ACCUMULATOR
006E  69  RTS  RETURN TO SIMULATOR

*******8080 STACK #3866F=#3874*******
1. SPECIAL MINIMUM CONFIGURATION PROCEDURE

This procedure does not require the control configuration hardware on KIM port PB or the use of the NMI interrupt (KIM's ST key). It is limited to running 3080 programs (i.e., no single-step or other features).

1. Load the Simulator programs from tape as specified in steps 1 - 15 of the Tape Loading Instructions of the User Manual.
2. Be absolutely certain to put $FF in location $C0F1!
3. Change $173F from $36C to $5D0.
4. Change $1790 from $8A to $3EF.
5. Examine location $1730 - it contains $A2.
6. Push the GO key.
7. The demonstration 3080 Time Of Day Clock program is now running. You should see a Hz count begin in the display.
8. The clock may be set by entering the time from the keypad in rapid sequence, less than 1 second between digits. The time digits will shift in from right to left. If you make a mistake, just begin again and the new time will be entered.
9. When restarting the program or another 3080 program, be sure to set the 3080 program counter to zero, or to whatever your 3080 program starting address may be.

2. TELETYPETE NOTES

The following routines may be called from your 3080 program to communicate with your teletype via KIM teletype routines. They may be located anywhere in memory. Call them from your 3080 program with the special "C65" opcode in the format, CB XXXX, where XXXX is the address of GET30-1 or OUT30-1, least significant byte first.

Read a character from the teletype:

<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$20 5A1E</td>
<td>GET30</td>
<td>JSR GETCH</td>
</tr>
<tr>
<td>$85 E5</td>
<td>STAZ A</td>
<td></td>
</tr>
<tr>
<td>$60</td>
<td>RTS</td>
<td></td>
</tr>
</tbody>
</table>

Write a character to the teletype:

<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$45 E5</td>
<td>OUT30</td>
<td>LDAZ A</td>
</tr>
<tr>
<td>$20 A01E</td>
<td>JSR OUTCH</td>
<td></td>
</tr>
<tr>
<td>$60</td>
<td>RTS</td>
<td></td>
</tr>
</tbody>
</table>

3. ADDITIONAL RELOCATION REQUIREMENTS

1. Don't change the third byte of three byte references to page zero. (i.e., Addresses 0142, 0367, 036C, 0384)
2. Change references to INFG2 (at 17B9 and 03BA) from $02 to the most significant byte of the page to which you are moving page two data.
3. Don't attempt to relocate the Simulator at an arbitrary address. To do so would require modifying the page select logic on page 6 of the Simulator listings. For simplicity sake keep the code in the same position relative to the page boundary.

If arbitrary relocation is a necessity, change XFRTBL to two byte entries, including the most significant byte of address with each entry. Change the logic on page 6 to extract both bytes from the table and push them onto the stack.