SCELBI-8H USER'S MANUAL

AUTHOR: NAT WADSWORTH

THE BASIC OPERATION OF A SCELBI-8H MINI-COMPUTER

THERE HAVE BEEN NUMEROUS EXAMPLES PUT FORTH OVER THE YEARS TO ILLUSTRATE THE BASIC SCHEME BEHIND THE OPERATION OF COMPUTERS. THE SCHEME IS DECEPTIVELY SIMPLE AND INCREDIBLY POWERFUL. THE POWER COMES FROM THE SPEED WITH WHICH THE MACHINES CAN PERFORM THE SIMPLE OPERATIONS. THE FUNDAMENTAL CONCEPT OF THE COMPUTER IS THAT IT IS A MACHINE THAT IS CAPABLE OF DOING TWO FUNDAMENTAL OPERATIONS AT VERY HIGH SPEED: FIRST IT IS ABLE TO OBTAIN A PIECE OF INFORMATION FROM A STORAGE AREA AND PERFORM A FUNCTION AS DIRECTED BY THE INFORMATION IT OBTAINS, AND SECONDLY, BASED ON ITS CURRENT STATUS, IT IS ABLE TO ASCERTAIN WHERE TO OBTAIN THE NEXT PIECE OF INFORMATION THAT WILL GIVE IT FURTHER "DIRECTIONS." THIS FUNDAMENTAL CONCEPT IS THE KEY TO THE OPERATION OF ALL DIGITAL COMPUTERS AND WHILE IT IS A SIMPLE CONCEPT, IT CAN BE BUILT UPON TO ARRIVE AT ALL THE COMPLEX OPERATIONS COMPUTERS OF TODAY CAN PERFORM. HOW THIS IS DONE IS WHAT THIS BOOK IS ABOUT.

ONE OF THE BEST ANALOGIES FOR DESCRIBING A COMPUTER'S BASIC OPERATIONS IS TO CONSIDER A BANK OF BOXES, SIMILAR TO A BANK OF POST OFFICE MAIL BOXES. A PIECE OF PAPER CONTAINING "DIRECTIONS" CAN BE PLACED IN EACH BOX. A PERSON IS DIRECTED TO GO TO THE BANK OF BOXES, AND AFTER STARTING AT A GIVEN PLACE, TO OPEN EACH BOX, WITHDRAW THE PIECE OF PAPER AND FOLLOW THE DIRECTIONS THERE-ON. THE BOXES ARE LABELED IN AN ORDERLY FASHION, AND THE PERSON IS ALSO TOLD, THAT UNLESS A PIECE OF PAPER IN A BOX DIRECTS OTHERWISE, WHEN THE PERSON IS FINISHED PERFORMING THE TASK DIRECTED THEY ARE TO REPLACE THE PAPER IN THE BOX AND PROCEED TO OPEN THE NEXT BOX. NOTE, HOWEVER, THAT A PIECE OF PAPER MAY GIVE DIRECTIONS TO ALTER THE SEQUENCE IN WHICH THE PERSON IS TO OPEN BOXES.

FIGURE 1 ON THE NEXT PAGE SHOWS A PICTURE OF A SET OF SUCH BOXES. EACH BOX IS LABELED FOR IDENTIFICATION.

TO PRESENT A VIEW OF A COMPUTERS OPERATION, ASSUME A PERSON HAS BEEN TOLD TO START AT BOX A1 AND TO FOLLOW THE DIRECTIONS CONTAINED ON THE PIECES OF PAPER IN THE BOXES UNTIL A PIECE OF PAPER CONTAINING THE DIRECTION "STOP" IS FOUND IN ONE OF THE BOXES. IN THIS EXAMPLE THE PERSON FINDS THE FOLLOWING "INSTRUCTIONS."

IN BOX A1 IS THE MESSAGE: "TAKE THE MATHEMATICAL VALUE OF 1 AND WRITE IT DOWN ON A SCRATCH PAD."

SINCE THE "INSTRUCTION" IN BOX A1 ONLY PERTAINED TO SOME FUNCTION THAT THE PERSON WAS TO PERFORM, AND DID NOT DIRECT THE PERSON TO GO TO SOME SPECIFIC BOX, THEN THE PERSON WILL SIMPLY GO ON TO THE NEXT BOX IN THE ROW. BOX A2 CONTAINS THE INFORMATION:

"ADD THE NUMBER 2 TO ANY VALUE ALREADY PRESENT ON YOUR SCRATCH PAD."

THE PERSON WILL AT THIS POINT PERFORM AN ADDITION AND HAVE A TOTAL "ACCUMULATED" VALUE ON THE PAD OF SCRATCH PAPER. THE ACCUMULATED VALUE WOULD BE 3. SINCE THERE ARE NO OTHER DIRECTIONS IN BOX A2, THE OPERATOR WOULD CONTINUE ON TO OPEN BOX A3 WHICH HAS THE FOLLOWING MESSAGE:

"PLACE ANY ACCUMULATED MATHEMATICAL VALUE YOU HAVE ON YOUR SCRATCH PAD INTO BOX H8."

THUS THE PERSON WOULD TEAR THE CURRENT SHEET OFF THE "SCRATCH PAD" AND PLACE IT - CONTAINING THE VALUE "3" - INTO BOX H8. NOTE THOUGH, THAT WHILE THE PERSON WAS DIRECTED TO PLACE THE ACCUMULATED VALUE ON THE SCRATCH PAD INTO BOX H8, THE PERSON WAS NOT DIRECTED TO ALTER THE SEQUENCE IN WHICH TO OBTAIN NEW "INSTRUCTIONS" SO THE PERSON WOULD PROCEED TO OPEN BOX A4 WHICH CONTAINS THE DIRECTIVE:

"TAKE THE MATHEMATICAL VALUE OF 6 AND PLACE IT ON YOUR SCRATCH PAD."

A1	A2	A3	A4	A5	A6	A7	A8
B1	B2	B3	B4	B5	B6	B7	B8
C1	C2	C3	C4	C5	C6	C7	C8
D1	D2	D3	D4	D5	D6	D7	D8
E1	E2	E3	E4	E5	E6	E7	E8
F1	F2	F3	F4	F5	F6	F7	F8
G1	G2	G3	G4	G5	G6	G7	G8
H1	H2	H3	H4	H5	H6	H7	H8

FIGURE 1

GOING ON TO BOX A5 THE PERSON FINDS:

"ADD 3 TO THE PRESENT VALUE ON YOUR SCRATCH PAD."

THIS IS OBVIOUSLY JUST A "DATA WORD." THE OPERATOR ADDS THE VALUE 6 FROM THE PREVIOUS BOX TO THE NUMBER 3, NOTING THE CALCULATION ON THE SCRATCH PAD AND PROCEEDS TO OPEN BOX A6:

"PLACE ANY ACCUMULATED VALUE YOU HAVE ON YOUR SCRATCH PAD INTO BOX H7."

THE PERSON THUS WOULD PUT THE VALUE "9" ON A PIECE OF PAPER (FROM THE SCRATCH PAD) INTO THE DESIGNATED BOX AND PROCEED TO OPEN BOX A7:

"GET THE VALUE PRESENTLY STORED IN BOX H8 AND SAVE THE VALUE ON YOUR SCRATCH PAD."

THIS IS A SIMPLE OPERATION AND THE PERSON PROCEEDS TO OPEN UP BOX A8:

"FETCH THE VALUE IN BOX H7. SUBTRACT THE VALUE ON YOUR SCRATCH PAD FROM THE VALUE FOUND IN BOX H7. LEAVE THE RESULT ON YOUR SCRATCH PAD."

WHEN THE OPERATOR HAS PERFORMED THIS OPERATION THE OPERATOR WILL HAVE FINISHED THE "A" ROW AND WILL THEN CONTINUE OBTAINING "INSTRUCT-IONS" BY GOING TO THE "B" ROW AND OPENING BOX B1 WHERE MORE DIREC-TIONS ARE FOUND:

"IF THE PRESENT VALUE ON YOUR SCRATCH PAD IS NOT ZERO GO TO BOX B3."

AT THIS TIME IF THE PERSON CHECKS THE SCRATCH PAD IT WILL BE FOUND THAT THE VALUE ON THE SCRATCH PAD IS INDEED NON-ZERO AS THE LAST CALCULATION PERFORMED ON THE SCRATCH PAD WAS TO SUBTRACT THE VALUE IN BOX H8 FROM THE VALUE IN BOX H7. IN THIS EXAMPLE THAT WOULD BE:

$$9 - 3 = 6$$

THEREFORE THE DIRECTIONS IN BOX B1 FOR THIS PARTICULAR CASE WILL TELL THE OPERATOR TO "JUMP OVER" BOX B2 AND GO TO BOX B3. FOR THE SAKE OF COMPLETENESS, HOWEVER, BOX B2 DOES CONTAIN AN INSTRUCTION, FOR HAD THE VALUE ON THE SCRATCH PAD BEEN ZERO THE OPERATOR WOULD NOT HAVE "JUMPED OVER" BOX B2 AND WOULD HAVE FOUND THE FOLLOWING MESSAGE INSIDE BOX B2:

"THE VALUES IN BOX H7 AND H8 ARE OF EQUAL VALUE. STOP!"

HOWEVER, FOR THE VALUES USED IN THIS EXAMPLE, THE PERSON WOULD HAVE "JUMPED" TO BOX B3 WHERE THE FOLLOWING DIRECTIVE WOULD BE FOUND:

"IF THE PRESENT VALUE ON YOUR SCRATCH PAD IS A "NEGATIVE NUMBER" JUMP TO BOX B5."

SINCE THIS IS NOT CURRENTLY THE CASE THE PERSON WILL NOT "JUMP" TO BOX B5, BUT WILL SIMPLY CONTINUE TO OPEN BOX B4 WHICH CONTAINS:

"THE VALUE IN BOX H7 IS LARGER THAN THE VALUE IN BOX H8. STOP!"

AT THIS POINT THE PERSON HAS COMPLETED THE "INSTRUCTION SEQUENCE" FOR THIS EXAMPLE. IT SHOULD BE NOTED, HOWEVER, THAT BOX B5 DID CON-TAIN THE MESSAGE:

"THE VALUE IN BOX H7 IS SMALLER THAN THE VALUE IN BOX H8. STOP!"

THIS LITTLE EXAMPLE OF A PERSON OPENING UP BOXES AND FOLLOWING THE DIRECTIONS CONTAINED IN EACH ONE IS VERY SIMILAR TO THE CONCEPT USED BY A COMPUTER. NOTE THAT EACH "INSTRUCTION" IS VERY SHORT AND SPECIFIC. ALSO NOTE, THAT THE COMBINATION OF ALL THE INSTRUCTIONS IN THE EXAMPLE WILL RESULT IN THE PERSON BEING DIRECTED TO SOLVE THE PROBLEM:

IS 1 + X GREATER THAN, LESS THAN, OR EQUAL TO: 6 + Y ?

FOR, THE READER CAN NOTE, IF THE "DATA WORDS" CONTAINED IN BOXES A2 AND A5 FOR THE EXAMPLE WERE CHANGED, THE SEQUENCE OF "INSTRUCTIONS"

WOULD STILL RESULT IN THE PERSON BEING TOLD TO "STOP" AT THE BOX THAT CONTAINED THE CORRECT ANSWER. THE READER CAN VERIFY THIS BY SIMPLY ASSUMING THAT DIFFERENT NUMBERS THAN THOSE USED IN THE EXAMPLE ARE IN BOXES A2 AND A5 AND GOING THROUGH THE INSTRUCTION SEQUENCE UNTIL TOLD TO "STOP."

THE EXAMPLE ILLUSTRATES HOW A CAREFULLY PLANNED SET OF DIRECTIONS, ARRANGED SUCH THAT THEY ARE PERFORMED IN A PRECISE SEQUENCE, CAN BE USED TO SOLVE A PROBLEM EVEN THOUGH THE "VARIABLES" (DATA) IN THE PROBLEM MAY VARY. SUCH A SET OF "INSTRUCTIONS" IS OFTEN TERMED AN "ALGORITHM" BY THOSE IN THE MINI-COMPUTER FIELD. THE EXAMPLE SOLVED A MATHEMATICAL PROBLEM USING THE "ALGORITHM," BUT THE READER WILL FIND THAT "ALGORITHMS" CAN BE DEVISED TO SOLVE MANY PROBLEMS ON A COMPUTER THAT ARE NOT STRICTLY MATHEMATICAL!

ANY PERSON LEARNING A NEW SKILL MUST OF NECESSITY LEARN THE VOCABULARY OF THE FIELD IN ORDER TO PROCEED TO ANY GREAT EXTENT. YOU MIGHT THINK THAT IT WOULD BE EASIER IF EVERYTHING WAS WRITTEN IN PLAIN EVERYDAY WORDS, BUT THE TRUTH OF THE MATTER IS THAT SPECIALIZED VOCABULARIES DO SERVE SEVERAL USEFUL FUNCTIONS. FOR ONE THING, THEY CAN GREATLY SHORTEN THE TIME THAT IT TAKES TO COMMUNICATE IDEAS OR CONCEPTS. IN TODAY'S FAST-MOVING WORLD THAT IS OF SIGNIFICANCE IN ITSELF. IN ADDITION, THE LIMITATIONS OF THE ENGLISH LANGUAGE OFTEN RESULT IN A GIVEN WORD HAVING A SPECIAL MEANING WHEN IT IS USED IN THE CONTEXT OF A PARTICULAR SUBJECT. ONE MUST KNOW THE NEW MEANING WHEN IT IS USED IN SUCH A MANNER. THROUGH-OUT THIS BOOK THE MEANINGS OF VARIOUS WORDS USED BY THOSE IN THE MINI-COMPUTER FIELD WILL BE POINTED OUT. FORTUNATLY, MUCH OF THE MINI-COMPUTER FIELD VOCABULARY IS VERY LOGICALLY NAMED. THIS IS PROBABLY DUE PARTLY TO THE FACT THAT COMPUTERS ARE OF NECESSITY EXTREMELY DEPENDENT ON LOGIC, AND HENCE MANY PERSONS WHO HELPED CREATE THE FIELD - AND BY THAT FACT WERE RATHER LOGICALLY ORIENTED THEMSELVES - SEEM TO HAVE HAD THE LOGICAL SENSE TO HAVE NAMED MANY OF THE PARTS AND SYSTEMS OF COMPUTERS AND COMPUTER PROGRAMS, IN A LOGICAL MANNER.

ON THE NEXT FEW PAGES ARE TWO DIAGRAMS (FIGURES 2A AND 2B) WHICH SHALL BE USED TO DEMONSTRATE THE ANALOGY BETWEEN THE PERSON TAKING "INSTRUCTIONS" FROM A GROUP OF MAIL BOXES AND THE BASIC OPERATION OF A REAL MINI-COMPUTER.

FIGURE 2A SHOWS THE POST OFFICE BOXES; A FIGURE REPRESENTATION OF A PERSON WHO IS ABLE TO "FETCH" AND RETURN THE "INSTRUCTIONS" OR "DATA" FROM AND TO THE BOXES, AND A "SCRATCH PAD" ON WHICH THE PERSON CAN MAKE TEMPORARY CALCULATIONS WHEN DIRECTED TO DO SO.

IN FIGURE 2B ARE THREE INTER-CONNECTED BOXES WHICH FORM A "BLOCK-DIAGRAM" OF A BASIC MINI-COMPUTER. THE UPPER-MOST PORTION OF THE "BLOCK-DIAGRAM" IS LABELED THE "MEMORY." THE MIDDLE PORTION IS LABELED THE "CENTRAL PROCESSOR UNIT" OR "CPU" FOR SHORT. THE LOWER PART OF THE DIAGRAM DEPICTS AN "ACCUMULATOR."

THE CORRELATION BETWEEN THE TWO PICTURES IS EXTREMELY SIMPLE. THE "POST OFFICE BOXES" CORRESPOND TO THE "MEMORY" PORTION OF A REAL MINI-COMPUTER. THE "MEMORY" IS A STORAGE PLACE. A LOCATION WHERE INSTRUCTIONS AND DATA CAN BE STORED FOR LONG LENGTHS OF TIME. THE "MEMORY" CAN BE "ACCESSED." "INSTRUCTIONS" AND/OR "DATA" CAN BE TAKEN OUT OF MEMORY, OPERATED ON, AND REPLACED. NEW "DATA" CAN BE PUT INTO THE "MEMORY." A "MEMORY" THAT CAN BE "READ FROM" AS WELL AS "WRITTEN INTO" IS CALLED A "READ AND WRITE MEMORY." A "READ AND WRITE MEMORY" IS OFTEN REFERRED TO AS A "RAM" AS AN ABBREVIATION. MANY TIMES IT IS FEASIBLE TO HAVE A "MEMORY" THAT IS ONLY "READ FROM."

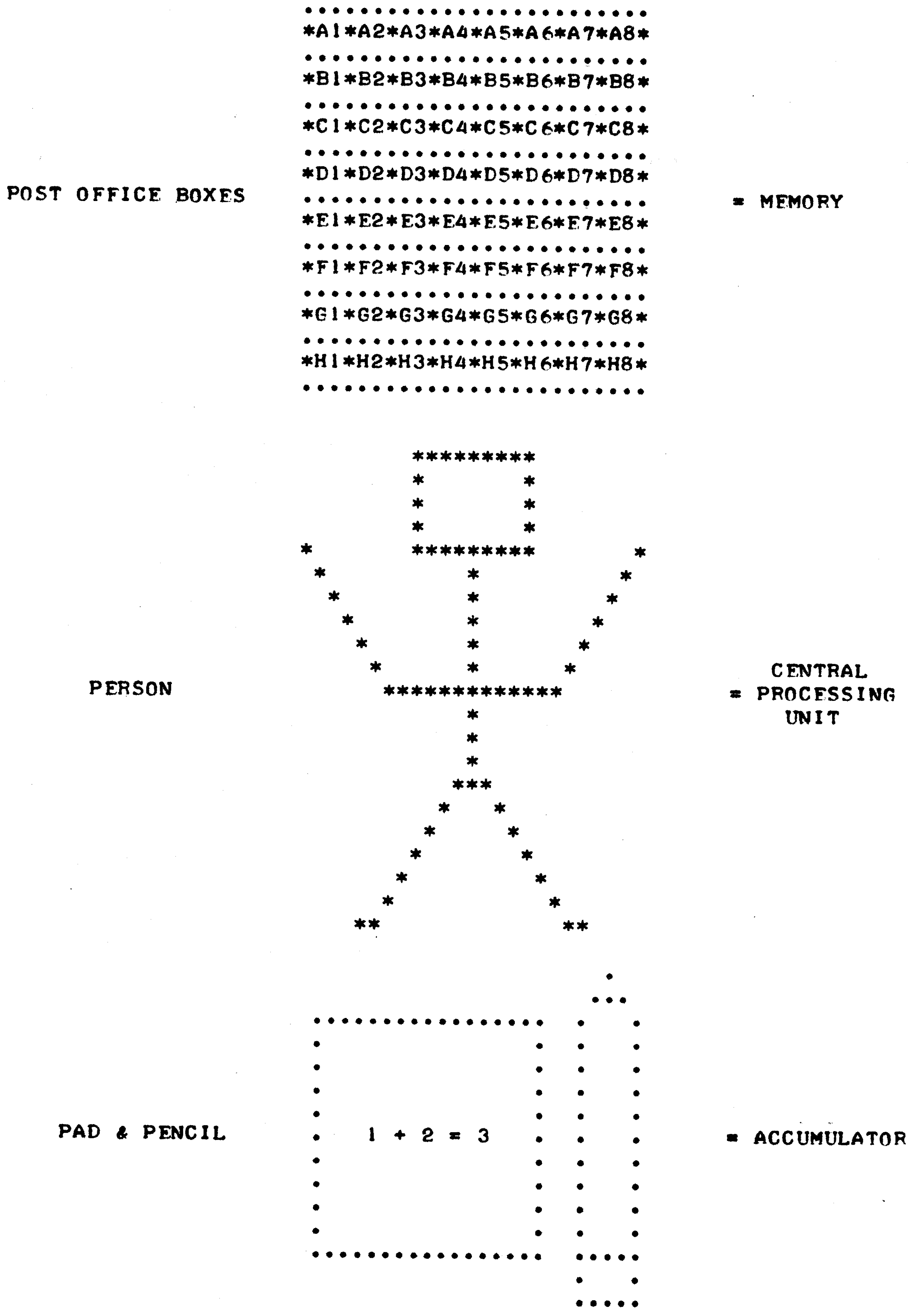

*A1*A2*A3*A4*A5*A6*A7*A8*
*B1*B2*B3*B4*B5*B6*B7*B8*
*C1*C2*C3*C4*C5*C6*C7*C8*
*D1*D2*D3*D4*D5*D6*D7*D8*
*E1*E2*E3*E4*E5*E6*E7*E8*
*F1*F2*F3*F4*F5*F6*F7*F8*
*G1*G2*G3*G4*G5*G6*G7*G8*
*H1*H2*H3*H4*H5*H6*H7*H8*
POST OFFICE BOXES
= MEMORY
PERSON
= CENTRAL PROCESSING UNIT
PAD & PENCIL
1 + 2 = 3
= ACCUMULATOR

FIGURE 2A

THE ANALAGOUS STRUCTURE OF A COMPUTER AS GIVEN IN THE EXAMPLE DISCUSSED IN THE TEXT

```
************************
*                      *
*                      *
*                      *
*                      *
*        MEMORY        *
*                      *
*                      *
*                      *
*                      *
************************
         .        .
         .        .
         .        .
         .        .
         .        .
************************
*                      *
*       CENTRAL        *
*      PROCESSOR       *
*        UNIT          *
*        (CPU)         *
*                      *
************************
         .        .
         .        .
         .        .
         .        .
         .        .
************************
*                      *
*     ACCUMULATOR      *
*                      *
************************
```

FIGURE 2B

BLOCK DIAGRAM OF THE FUNDAMENTAL COMPONENTS
OF A MINI-COMPUTER

A MEMORY THAT IS NEVER "WRITTEN INTO," BUT IS ONLY USED TO "READ FROM," IS TERMED A "READ ONLY MEMORY" AND IS ABBREVIATED AS A "ROM." FOR THE PRESENT DISCUSSION THE TERM "MEMORY" WILL REFER TO A "READ AND WRITE MEMORY" ("RAM"). THE UTILIZATION OF "READ ONLY MEMORIES," AS A GENERAL RULE, REQUIRES A MORE SOPHISTICATED "CENTRAL PROCESSOR UNIT" AND MORE SOPHISTICATED PROGRAMMING TECHNIQUES, THAN THAT ILLUSTRATED IN THE PRESENT DISCUSSION, AND HENCE THEIR USE WILL NOT BE DISCUSSED AT THIS TIME.

THE FIGURE OF A PERSON IN PICTURE 2A CORRESPONDS TO THE CENTRAL PROCESSOR UNIT IN FIGURE 2B. THE CENTRAL PROCESSOR UNIT IN A COMPUTER IS THE SECTION THAT "CONTROLS" THE OVER-ALL OPERATION OF THE MACHINE. THE "CPU" CAN RECEIVE (FETCH) "INSTRUCTIONS" OR "DATA" FROM THE MEMORY. IT IS ABLE TO "INTERPRET" THE "INSTRUCTIONS" IT FETCHES FROM THE MEMORY. IT IS ALSO ABLE TO PERFORM VARIOUS TYPES OF MATHEMATICAL OPERATIONS. IT CAN ALSO "RETURN" INFORMATION TO THE MEMORY - FOR INSTANCE MAKE DEPOSITS OF "DATA" INTO THE MEMORY. THE "CPU" ALSO CONTAINS CONTROL SECTIONS THAT ENABLE IT TO SEQUENTIALLY "ACCESS" THE "NEXT" LOCATION IN MEMORY WHEN IT HAS FINISHED PERFORMING AN OPERATION, OR, IF IT IS DIRECTED TO DO SO, TO "ACCESS" THE MEMORY AT A SPECIFIED LOCATION, OR

TO "JUMP" TO A NEW AREA IN MEMORY FROM WHICH TO CONTINUE FETCHING "INSTRUCTIONS."

THE PAD OF PAPER AND PENCIL IN FIGURE 2A CORRESPONDS TO THE BLOCK TITLED "ACCUMULATOR" IN PICTURE 2B. THE "ACCUMULATOR" IS A TEMPORARY "REGISTER" OR "MANIPULATING AREA" WHICH IS USED BY THE CPU WHEN IT IS PERFORMING OPERATIONS SUCH AS ADDING TWO NUMBERS. ONE NUMBER, OR PIECE OF INFORMATION CAN BE TEMPORARILY HELD IN IT WHILE THE CENTRAL PROCESSOR UNIT GOES ON TO OBTAIN ADDITIONAL INSTRUCTIONS OR DATA FROM MEMORY. IT IS AN ELECTRONIC "SCRATCH PAD" FOR THE CPU.

THE THREE FUNDAMENTAL UNITS: THE MEMORY, CENTRAL PROCESSOR UNIT, AND THE ACCUMULATOR ARE AT THE HEART OF EVERY DIGITAL COMPUTER SYSTEM. OF COURSE, THERE ARE OTHER PARTS WHICH WILL BE ADDED IN AND EXPLAINED LATER, BUT THESE FUNDAMENTAL PORTIONS CAN BE USED TO EXPLAIN THE BASIC OPERATION OF A DIGITAL MINI-COMPUTER WHICH IS THE PURPOSE OF THIS CHAPTER.

THE READER SHOULD LEARN THE NAMES OF THE BASIC PARTS OF THE MINI-COMPUTER AS THEY ARE PRESENTED. NOTE HOW EASY IT IS TO RE-MEMBER THE PORTIONS THAT HAVE BEEN SHOWN. THE "REMEMBERING" ELEMENT IS A "MEMORY." THE PORTION THAT DOES THE "WORK" OR PRO-CESSING IS SIMPLY TERMED THE "CENTRAL PROCESSOR UNIT," AND THE PART THAT IS USED TO ACCUMULATE INFORMATION TEMPORARILY IS APTLY CALLED THE "ACCUMULATOR!"

THE READER SHOULD NOW HAVE A CONCEPTUAL VIEW OF THE CONCEPT BEHIND A COMPUTER'S OPERATION AND AN UNDERSTANDING OF THE MACHINE'S MOST BASIC ORGANIZATION. IT IS SIMPLY A MACHINE THAT CAN FETCH INFORMATION FROM A MEMORY, INTERPRET THE INFORMATION AS AN INSTRUC-TION OR DATA, PERFORM A VERY SMALL OPERATION, AND CONTINUE ON TO DETERMINE THE NEXT OPERATION THAT IS TO BE PERFORMED. EACH OPERATION IT IS CAPABLE OF DOING IS VERY TINY BY ITSELF, BUT WHEN THE MANY OPERATIONS OF A TYPICAL "PROGRAM" ARE PERFORMED IN SEQ-UENCE, THE SOLUTIONS TO VERY COMPLEX PROBLEMS CAN BE OBTAINED. IT IS IMPORTANT TO REMEMBER THAT THE MINI-COMPUTER CAN PERFORM EACH LITTLE OPERATION IN JUST A FEW MILLIONTHS OF A SECOND! THUS, A PROGRAM THAT MIGHT SEEM VERY LARGE TO A PERSON - SAY ONE WITH MANY THOUSANDS OF INDIVIDUAL INSTRUCTIONS - WOULD ONLY TAKE A DIGITAL MINI-COMPUTER A FEW THOUSANDTH'S OF A SECOND TO PERFORM. THE SPEED WITH WHICH THE COMPUTER CAN EXECUTE INDIVIDUAL INSTRUCTIONS IS WHAT GIVES THE MINI-COMPUTER ITS FANTASTIC CAPABILITY.

IT IS NOW TIME TO START DELVING INTO THE ACTUAL PHYSICAL MANNER IN WHICH A MINI-COMPUTER OPERATES. HOW CAN A MACHINE BE CON-STRUCTED SO THAT IT IS ABLE TO PERFORM THE PROCESSES OF THE CENTRAL PROCESSOR UNIT? WHILE IT WILL REQUIRE A NUMBER OF PAGES OF TEXT TO EXPLAIN THE PROCEDURE - IT IS NOT NEARLY AS DIFFICULT TO UNDERSTAND AS MANY PEOPLE MIGHT SUSPECT. THE COMPLEXITY OF A COMPUTER WHEN FIRST VIEWED BY A PERSON IS CAUSED BY THE FACT THAT IT APPEARS TO CONSIST OF MANY HUNDREDS OF PARTS. IT BECOMES MUCH SIMPLER WHEN ONE UNDERSTANDS THAT THE HUNDREDS OF PARTS ARE REALLY MADE UP FROM A FEW DOZEN SIMILAR PARTS AND THEY ARE CAREFULLY ORGANIZED INTO JUST A FEW MAJOR OPERATING PORTIONS. THE READER IS ALREADY FAMILIAR WITH THE MOST FUNDAMENTAL PORTIONS.

AS FANTASTIC AS IT MAY SOUND AT FIRST, A DIGITAL MINI-COMPUTER CAN BE THOUGHT OF AS REALLY NOTHING MORE THAN A HIGHLY ORGANIZED COLLECTION OF "ON OR OFF" SWITCHES! YES, COMPUTERS ARE CONSTRUCTED FROM ELECTRONIC DEVICES THAT CAN ONLY ASSUME ONE OF TWO POSS-

IBLE STATES! THE ELECTRONIC SWITCHES CAN BE CONSTRUCTED IN A VARIETY OF WAYS. FOR INSTANCE, THE SWITCH CAN BE MADE SO THAT THE VOLTAGE AT A GIVEN POINT IS EITHER HIGH OR LOW, OR CURRENT THROUGH A DEVICE IS EITHER FLOWING OR NOT FLOWING, OR FLOWING IN ONE DIRECTION, AND THEN THE OTHER DIRECTION. BUT, REGARDLESS OF HOW THE ELECTRONIC SWITCH IS CONSTRUCTED, ITS STATUS CAN ALWAYS BE REPRESENTED AS BEING EITHER "ON" OR "OFF." THIS "ON" OR "OFF" STATUS CAN BE MATHEMATICALLY SYMBOLIZED MOST SUITABLY BY A MATHEMATICAL SYSTEM BASED ON "BINARY" NOTATION.

SOME PEOPLE TEND TO THINK THAT COMPUTERS ARE VERY DIFFICULT TO UNDERSTAND BECAUSE THEY HAVE HEARD OF "STRANGE" TYPES OF MATHEMATICS THAT ARE OFTEN REFERRED TO IN CONJUNCTION WITH COMPUTERS. IN ACTUALITY MUCH OF THE MATHEMATICS THAT ARE DEALT WITH IN COMPUTER TECHNOLOGY ARE MUCH EASIER TO UNDERSTAND AND DEAL WITH THAN THE DECIMAL SYSTEM THAT THE AVERAGE PERSON IS FAMILIAR WITH. IN THE DECIMAL NUMBERING SYSTEM A PERSON MUST LEARN 10 DIFFERENT SYMBOLS AND IN ORDER TO MANIPULATE THOSE SYMBOLS THEY MUST MEMORIZE A LOT OF INFORMATION. FOR INSTANCE, LOOK AT HOW STUDENTS ARE TAUGHT TO MULTIPLY. THE LEARNING PROCESS ACTUALLY INVOLVES THE STUDENT HAVING TO MEMORIZE A RATHER LARGE NUMBER OF FACTS. BECAUSE OF THE WAY IT IS TYPICALLY TAUGHT, MOST STUDENTS NEVER REALIZE HOW MUCH WORK THEY HAVE TO GO THROUGH JUST TO LEARN THE MULTIPLICATION TABLES! THE TEACHER DOES NOT STAND UP AND SAY; "O.K., NOW YOU ARE GOING TO MEMORIZE ABOUT 100 FACTS." INSTEAD, OVER A PERIOD OF A FEW WEEKS OR SO, THE STUDENT IS MADE TO MEMORIZE THE 100 OR SO FACTS - A FEW AT A TIME. THE STUDENT MUST LEARN THE VALUE OF EACH DIGIT MULTIPLIED BY ALL THE OTHER DIGITS IN THE DECIMAL NUMBERING SYSTEM. THE DECIMAL NUMBERING SYSTEM IS FAR MORE COMPLICATED FOR THE BEGINNER THAN LEARNING THE BINARY NUMBERING SYSTEM - AND THE BINARY NUMBERING SYSTEM IS THE ONE UTILIZED BY COMPUTERS AT THEIR MOST BASIC FUNCTIONING LEVEL. THE REASON THE COMPUTER USES THE BINARY SYSTEM IS BECAUSE IT IS THE SIMPLEST SYSTEM AROUND AND HENCE THE EASIEST ONE WITH WHICH TO CONSTRUCT A COMPUTING MACHINE!

READERS KNOW THE WORD "BINARY" INDICATES "TWO." COMPUTERS ARE BUILT UP OF ELECTRONIC SWITCHES THAT CAN ONLY HAVE TWO POSSIBLE STATES. THE SWITCHES ARE BINARY DEVICES. THE STATUS OF THE SWITCHES CAN BE REPRESENTED MATHEMATICALLY UTILIZING THE "BINARY" NUMBERING SYSTEM. THE BINARY NUMBERING SYSTEM ONLY HAS TWO DIGITS IN IT! THEY ARE ZERO (0) AND ONE (1). A SWITCH CAN THUS BE MATHEMATICALLY SYMBOLIZED, FOR INSTANCE, BY A ZERO WHEN IT IS "OFF" AND A ONE WHEN IT IS "ON." THE OPPOSITE RELATIONSHIP COULD ALSO BE ESTABLISHED, A ONE COULD BE USED TO REPRESENT A SWITCH BEING "OFF" AND A ZERO USED TO REPRESENT A SWITCH AS "ON." IT WOULD MAKE NO DIFFERENCE MATHEMATICALLY WHICH CONVENTION WAS USED AS LONG AS ONE WAS CONSISTENT. FOR THE PURPOSES OF THE PRESENT DISCUSSION THE READER CAN ASSUME THAT THE FIRST CONVENTION (SWITCH OFF = 0, SWITCH ON = 1) WILL BE USED.

IT SHOULD BE IMMEDIATELY APPARENT THAT WORKING WITH A NUMBERING SYSTEM BASED ON ONLY TWO INTEGERS WILL BE A LOT EASIER THAN WORKING WITH ONE HAVING 10 INTEGER SYMBOLS. IN FACT, MOST PROBLEMS FOR PEOPLE LEARNING THE BINARY SYTEM, COME ABOUT BECAUSE THEY TEND TO FORGET HOW SIMPLE IT IS AND THEY TEND TO KEEP GOING TOWARDS A DECIMAL SOLUTION OUT OF HABIT WHEN THEY ARE WORKING WITH THE BINARY SYSTEM. FOR INSTANCE, WHEN ONE STARTS TO ADD BINARY NUMBERS, AS SOON AS THE VALUE "1" IS EXCEEDED, A "CARRY" TO THE NEXT COLUMN MUST BE MADE. THE VALUE OF THE ADDITION OF "1 + 1" IN THE BINARY SYSTEM IS: 10. IT IS NOT 2! THERE IS NO SUCH INTEGER AS "2" IN THE BINARY NUMBERING SYSTEM. HOWEVER, WHEN A PERSON WHO HAS WORKED WITH THE DECIMAL SYSTEM FOR YEARS FIRST STARTS WORKING WITH THE BIN-

ARY SYSTEM, OLD DECIMAL HABITS TEND TO GET IN THE WAY. THE READER WILL HAVE TO BEWARE!

TO FORMERLY INTRODUCE THE BINARY MATHEMATICAL SYSTEM ONE CAN START BY STATING THAT IT USES TWO INTEGERS ZERO (0) AND ONE (1) AND NO OTHERS. A BINARY NUMBER HAS A VALUE DETERMINED BY THE VALUE OF THE INTEGERS THAT MAKE UP THE NUMBER, AND THE POSITION OF THE DIGITS.

IN THE DECIMAL NUMBERING SYSTEM THE READER IS FAMILIAR WITH THE LOCATION OF A DIGIT HAVING A "WEIGHTED" VALUE AS FOLLOWS; A THREE DIGIT NUMBER HAS A VALUE DETERMINED BY THE UNIT VALUE OF THE DIGIT IN THE RIGHT-MOST COLUMN PLUS THE VALUE OF THE DIGIT TO THE LEFT OF IT MULTIPLIED BY 10, PLUS THE VALUE OF THE THIRD DIGIT MULTIPLIED BY ONE HUNDRED AS ILLUSTRATED IN THE FOLLOWING EXAMPLE:

```
                    THE DECIMAL NUMBER  345

                    IS EQUAL TO:

                    5 UNITS         =      5

          PLUS (+)  4 TIMES 10      =     40

          PLUS (+)  3 TIMES 100     =    300
```

IN OTHER WORDS, AFTER THE RIGHT HAND MOST COLUMN (WHICH HAS THE VALUE OF THE DIGIT), EACH COLUMN TO THE LEFT IS GIVEN A WEIGHTING FACTOR WHICH INCREASES AS A POWER OF THE TOTAL NUMBER OF DIGITS UTILIZED BY THE NUMBERING SYSTEM. NOTE THAT IN THE ABOVE EXAMPLE THE 4 REPRESENTING 40 UNITS IS EQUAL TO 4 TIMES THE NUMBER OF INTEGER SYMBOLS IN THE DECIMAL SYSTEM (10) BECAUSE IT IS LOCATED IN THE SECOND COLUMN FROM THE RIGHT. THE NUMBER 3 REPRESENTING 300 UNITS IS EQUAL TO 3 TIMES THE NUMBER OF INTEGER SYMBOLS IN THE DECIMAL SYSTEM SQUARED BECAUSE IT IS LOCATED IN THE THIRD COLUMN FROM THE RIGHT. FOR MATHEMATICIANS, THIS RELATIONSHIP OF THE WEIGHTED VALUE OF THE DIGITS BASED ON THEIR POSITION CAN BE DESCRIBED IN MATHEMATICAL SHORT-HAND AS FOLLOWS:

IF THE NUMBER OF DIFFERENT INTEGER SYMBOLS IN THE NUMBERING SYSTEM IS U (FOR THE DECIMAL SYTEM U=10)

AND THE COLUMN WHOSE WEIGHTED VALUE IS TO BE DETERMINED IS COLUMN NUMBER M (STARTING WITH THE RIGHT MOST COLUMN AND COUNTING TO THE LEFT)

AND ANY DIGIT IS REPRESENTED BY THE SYMBOL X

THEN THE WEIGHTED VALUE OF A DIGIT IN COLUMN M IS EXPRESSED AS:

X TIMES U RAISED TO THE POWER (M-1) OR X*((U↑(M-1))

NOTE: IN THIS BOOK THE ASTERISK (*) WILL BE USED TO INDICATE MULTIPLICATION AND THE UP-ARROW (↑) WILL BE USED TO SIGNIFY THE RAISING OF A NUMBER TO A POWER.

THE READER CAN EASILY VERIFY THAT THE ABOVE FORMULA APPLIES TO THE DECIMAL NUMBERING SYSTEM. HOWEVER, THE ABOVE FORMULA IS A GENERAL FORMULA THAT CAN BE USED TO DETERMINE THE WEIGHTED POSITIONAL VALUE OF ANY NUMBERING SYSTEM. IT WILL BE USED TO DETERMINE THE WEIGHTED

POSITIONAL VALUES OF NUMBERS IN THE BINARY NUMBERING SYSTEM.

IN THE BINARY NUMBERING SYSTEM THERE ARE JUST TWO DIFFERENT INTEGER SYMBOLS (Ø AND 1). THUS U IN THE ABOVE FORMULA IS EQUAL TO 2. FOR ILLUSTRATIVE PURPOSES ASSUME THE FOLLOWING BINARY NUMBER IS TO BE ANALYZED:

1 Ø 1

AND IT IS DESIRED TO DETERMINE ITS VALUE IN TERMS OF DECIMAL NUMBERS. (REMEMBER ITS BINARY VALUE IS JUST: 1 Ø 1). USING THE ABOVE FORMULA; FOR THE DIGIT IN THE RIGHT-MOST COLUMN: M IS EQUAL TO 1, THUS (M-1) IS EQUAL TO Ø, AND WITH X = 1:

```
      WEIGHTED VALUE = X*((U↑(M-1)) = 1*((2↑(Ø)) = 1*1 = 1
```

(REMEMBER THAT ANY NUMBER RAISED TO THE ZERO POWER IS EQUAL TO 1.) GOING ON TO THE NEXT DIGIT IT CAN BE SEEN THAT THE WEIGHTED VALUE IS SIMPLY Ø! FINALLY, THE DIGIT IN THE THIRD COLUMN FROM THE RIGHT HAS THE WEIGHTED VALUE BECAUSE OF ITS POSITION:

```
      WEIGHTED VALUE = 1*((2↑(3-1)) = 1*((2↑2)) = 1*4 = 4
```

THEN, BY ADDING UP THE SUM OF THE WEIGHTED VALUES (SIMILAR TO THAT DONE FOR THE DECIMAL EXAMPLE EARLIER) ONE CAN SEE THAT THE DECIMAL EQUIVELANT OF 1 Ø 1 BINARY IS 5:

```
                    THE BINARY NUMBER  1Ø1

                    IS EQUAL TO:

                    1 UNITS          =        1

          PLUS (+) Ø TIMES 2        =        Ø

          PLUS (+) 1 TIMES 4        =        4
```

AND THUS 1 Ø 1 IN THE BINARY NUMBERING SYSTEM IS THE SAME AS 5 IN THE DECIMAL NUMBERING SYSTEM.

THERE WILL BE MORE TO LEARN ABOUT THE BINARY NUMBERING SYSTEM. HOWEVER, THE BRIEF INFORMATION GIVEN WILL BE ENOUGH TO CONTINUE ON WITH THE DISCUSSION THAT THIS SECTION IS PRIMARILY CONCERNED WITH - THE BASIC OPERATION OF A MINI-COMPUTER. SINCE THE READER IS NOW AWARE THAT A COMPUTER IS COMPOSED OF NUMEROUS ELECTRONIC SWITCHES AND KNOWS THAT ONE CAN USE A MATHEMATICAL SHORT-HAND TO REPRESENT THE STATUS OF THE SWITCHES (WHETHER THEY ARE "ON" OR "OFF"), AND IS ALSO AWARE OF THE FUNDAMENTAL CONCEPT BEHIND A COMPUTER'S OPERATION; IT IS NOW POSSIBLE TO PROCEED TO SHOW HOW ELECTRONIC SWITCHES CAN BE ARRANGED TO BUILD A FUNCTIONAL COMPUTER. THAT IS, HOW THE ELECTRONIC SWITCHES CAN BE ARRANGED AND INTERCONNECTED IN A FASHION THAT WILL ALLOW A MACHINE TO "FETCH" A PIECE OF INFORMATION FROM A "MEMORY" SECTION, DECODE THE INFORMATION SO AS TO DETERMINE AN "INSTRUCTION," AND ALSO DETERMINE WHERE TO OBTAIN THE NEXT INSTRUCTION OR ADDITIONAL "DATA."

TO BEGIN THIS PART OF THE DISCUSSION IT WILL BE BENEFICIAL FOR THE READER TO PICTURE A GROUP OF CELLS (SIMILAR TO THE POST-OFFICE BOXES SHOWN EARLIER) ARRANGED IN ORDERLY ROWS AS SHOWN IN FIGURE 3. THIS TIME, INSTEAD OF EACH CELL HOLDING A COMPLETE INSTRUCTION, IT CAN BE UNDERSTOOD THAT EACH CELL ONLY REPRESENTS PART OF AN INSTRUCTION AND THAT IT TAKES A WHOLE ROW OF CELLS TO MAKE UP AN INSTRUCTION. FURTHER-

MORE, EACH CELL MAY ONLY CONTAIN THE MATHEMATICAL SYMBOL FOR A ONE (1) OR A ZERO (0) - OR, IN OTHER WORDS, ITS CONTENTS REPRESENT THE STATUS OF AN ELECTRONIC SWITCH!

```
                  .................................
  WORD #1         * 1 * 0 * 1 * 0 * 1 * 0 * 1 * 0 *
                  .................................
  WORD #2         * 0 * 1 * 0 * 1 * 0 * 1 * 0 * 1 *
                  .................................
  WORD #3         * 1 * 1 * 0 * 0 * 1 * 1 * 0 * 0 *
                  .................................
  WORD #4         * 0 * 0 * 1 * 1 * 0 * 0 * 1 * 1 *
                  .................................
  WORD #5         * 1 * 1 * 1 * 1 * 0 * 0 * 0 * 0 *
                  .................................
  WORD #6         * 0 * 0 * 0 * 0 * 1 * 1 * 1 * 1 *
                  .................................
  WORD #7         * 1 * 1 * 1 * 1 * 1 * 1 * 1 * 1 *
                  .................................
  WORD #8         * 0 * 0 * 0 * 0 * 0 * 0 * 0 * 0 *
                  .................................
```

FIGURE 3

AT THIS TIME A FEW MORE COMPUTER TECHNOLOGY DEFINITIONS WILL BE ILLUSTRATED. IN FIGURE 3, EACH BOX CONTAINING A BINARY 1 OR 0 REPRESENTS WHAT IS CALLED A "BIT" OF INFORMATION. WHILE EACH CELL MAY ONLY CONTAIN ONE PIECE OF INFORMATION AT A TIME, A CELL CAN ACTUALLY REPRESENT ONE OF TWO POSSIBLE STATES OF INFORMATION. THIS IS BECAUSE THE CELL CAN BE IN TWO POSSIBLE STATES - IT EITHER CONTAINS A ZERO OR A ONE. IF ONE STARTS ASSIGNING POSITIONAL VALUES TO THE CELLS IN A ROW, IT CAN BE SEEN THAT THE TOTAL NUMBER OF POSSIBLE STATES IN ONE ROW WILL INCREASE RAPIDLY. FOR INSTANCE, TWO CELLS IN A ROW CAN REPRESENT UP TO 4 STATES OF INFORMATION. THIS IS BECAUSE TWO CELLS SIDE-BY-SIDE, CONTAINING EITHER A 0 OR 1 IN EACH CELL CAN HAVE ONE OF THE FOLLOWING FOUR STATES AT A PARTICULAR MOMENT IN TIME: 1 0, 0 1, 1 1, OR 0 0. THREE CELLS IN A ROW CAN REPRESENT UP TO EIGHT STATES OF INFORMATION AS THE POSSIBLE STATES OF THREE CELLS SIDE-BY-SIDE ARE: 0 0 0, 0 0 1, 0 1 0, 0 1 1, 1 0 0, 1 0 1, 1 1 0, 1 1 1. IN FACT, WHEN EACH CELL CAN REPRESENT A BINARY NUMBER, THE TOTAL NUMBER OF STATES OF INFORMATION THAT A ROW OF "N" CELLS CAN REPRESENT IS: 2 TO THE N'TH POWER (2↑N). THUS, A ROW OF EIGHT BINARY CELLS CAN REPRESENT 2 TO THE EIGHTH (256) STATES OF INFORMATION! THAT IS, THE COMBINATION OF THE EIGHT CELLS CAN BE FILLED WITH ZEROS AND ONES IN 256 DIFFERENT PATTERNS!

A GROUP (ROW) OF CELLS IN A COMPUTER'S MEMORY IS OFTEN REFERRED TO AS A "WORD." A "WORD" IN A COMPUTER'S MEMORY IS A FIXED SIZE GROUP OF CELLS THAT ARE "ACCESSED" OR MANIPULATED DURING ONE OPERATIONAL CYCLE OF THE CENTRAL PROCESSING UNIT (CPU). THE CPU WILL EFFECTIVELY HANDLE ALL THE CELLS IN A "WORD" IN MEMORY SIMULTAINEOUSLY WHENEVER IT PROCESSES INFORMATION IN THE MEMORY. DIGITAL COMPUTERS CAN HAVE VARYING "WORD LENGTHS" DEPENDING ON HOW THEY ARE ENGINEERED. THE SCELBI-8H SERIES OF MINI-COMPUTERS HAVE A MEMORY WORD SIZE CONSISTING OF EIGHT CELLS. THE NUMBER OF CELLS IN A WORD, AND THE NUMBER OF WORDS IN A COMPUTER'S MEMORY HAVE A LOT TO DO WITH THE MACHINE'S OVER-ALL CAPABILITY. THE SCELBI-8H SERIES OF MINI-COMPUTERS ALLOW THE MEMORY SIZE OF THE MACHINE TO BE EASILY EXPANDED TO INCREASE OVER-ALL COMPUTING POWER. TYPICAL SCELBI-8H SYSTEMS HAVE FROM 256 TO 4,096 "WORDS" IN MEMORY. IN

SPECIAL CASES THIS SIZE CAN BE INCREASED TO 16,384 WORDS. SINCE EACH WORD ACTUALLY CONTAINS EIGHT CELLS, THE NUMBER OF CELLS IN A SCELBI-8H SYSTEM IS EIGHT TIMES THE NUMBER OF WORDS IN THE MEMORY. THUS TYPICAL SYSTEMS HAVE FROM 2,048 TO 32,768 CELLS AND SPECIAL SYSTEMS CAN HAVE UP TO 131,072 CELLS IN MEMORY. THUS, A LARGE AMOUNT OF INFORMATION CAN BE "STORED" IN THE COMPUTER'S MEMORY AT ANY ONE TIME.

THE ASTUTE READER MAY HAVE ALREADY FIGURED OUT A VERY SPECIAL REASON FOR GROUPING CELLS INTO "WORDS" IN MEMORY. IT WAS POINTED OUT EARLIER THAT A ROW OF EIGHT CELLS COULD REPRESENT UP TO 256 DIFFERENT PATTERNS. NOW, IF EACH POSSIBLE PATTERN COULD BE "DECODED" BY ELECTRONIC MEANS SO THAT A PARTICULAR PATTERN COULD SPECIFY A PRECISE "INSTRUCTION" FOR THE CENTRAL PROCESSOR UNIT, THEN A LARGE GROUP OF "INSTRUCTIONS" WOULD BE AVAILABLE FOR USE BY THE MACHINE. THAT IS EXACTLY THE CONCEPT USED IN A DIGITAL COMPUTER. PATTERNS OF ONES AND ZEROS, ORGANIZED INTO A COMPUTER "WORD" ARE STORED IN MEMORY. THE CPU IS ABLE TO EXAMINE A WORD IN MEMORY AND DECODE THE PATTERN CONTAINED THERE-IN TO DETERMINE THE PRECISE OPERATION THAT IT IS TO PERFORM. THE SCELBI-8H, BECAUSE OF TECHNICAL REASONS, DOES NOT DECODE EVERY ONE OF THE POSSIBLE 256 PATTERNS THAT CAN BE HELD IN A ROW OF EIGHT CELLS AS AN INSTRUCTION. IT DOES, HOWEVER, HAVE AN "INSTRUCTION SET" OF ABOUT 170 "INSTRUCTIONS" WHICH ARE REPRESENTED BY DIFFERENT PATTERNS OF ONES AND ZEROS IN AN EIGHT CELL MEMORY "WORD." EACH PATTERN THAT REPRESENTS AN "INSTRUCTION" CAN BE DECODED BY THE CPU AND WILL CAUSE THE CPU TO PERFORM A SPECIFIC FUNCTION. ALL OF THE FUNCTIONS WILL BE EXPLAINED IN DETAIL LATER IN THIS MANUAL.

THERE IS ANOTHER INGREDIENT NECESSARY FOR MAKING THE MACHINE "AUTOMATIC" IN OPERATION. THAT IS THAT THE CPU MUST "KNOW" WHERE TO OBTAIN THE NEXT "INSTRUCTION" IN MEMORY AFTER IT COMPLETES AN OPERATION. THAT FUNCTION IS GREATLY AIDED BY HAVING THE MEMORY CELLS GROUPED AS "WORDS." THE READER SHOULD NOTE THAT IN FIGURE 3 EACH GROUP OF CELLS REPRESENTING A WORD WAS LABELED AS: "WORD #1," "WORD #2," ETC.. THERE IS A SPECIAL PORTION OF THE CENTRAL PROCESSOR UNIT THAT IS USED TO CONTROL WHERE THE NEXT WORD CONTAINING AN INSTRUCTION IN MEMORY IS LOCATED. THIS SPECIAL PART IS COMMONLY REFERRED TO AS THE "PROGRAM COUNTER." ONE REASON IT WAS GIVEN THE NAME "PROGRAM COUNTER" IS BECAUSE MOST OF THE TIME ALL IT DOES IS COUNT! IT COUNTS MEMORY WORDS! EACH WORD IN MEMORY IS CONSIDERED TO HAVE AN "ADDRESS." IN FIGURE 3 EACH WORD WAS GIVEN AN "ADDRESS" BY SIMPLY DESIGNATING EACH WORD WITH A NUMBER. WORD #1 HAS AN "ADDRESS" OF 1. WORD #2 HAS AN ADDRESS OF 2, ETC.. THE "PROGRAM COUNTER" PORTION OF THE CPU KEEPS TABS ON WHERE THE CPU SHOULD OBTAIN THE NEXT INSTRUCTION BY MAINTAINING AN "ADDRESS" OF THE WORD IN MEMORY THAT IS TO BE PROCESSED! ABOUT 90% OF THE TIME ALL THE PROGRAM COUNTER DOES IS "INCREMENT" THE VALUE IT HAS EACH TIME THE CPU FINISHES DOING AN OPERATION. THUS, IF THE COMPUTER WERE TO START EXECUTING A SIMPLE PROGRAM THAT BEGAN BY ITS PERFORMING THE INSTRUCTION CONTAINED IN "WORD #1" IN MEMORY - THE VERY PROCESS OF HAVING THE MACHINE START THE PROGRAM AT THAT LOCATION IN MEMORY WOULD CAUSE THE PROGRAM COUNTER TO ASSUME A VALUE OF 1. AS SOON AS THE CPU HAD PERFORMED THE FUNCTION THE "PROGRAM COUNTER" WOULD INCREMENT ITS VALUE TO 2. THE CPU WOULD THEN LOOK AT THE PROGRAM COUNTER AND SEE THAT ITS NEXT INSTRUCTION WAS LOCATED IN WORD #2 IN MEMORY. WHEN THE INSTRUCTION IN WORD #2 HAD BEEN PROCESSED THE "PROGRAM COUNTER" WOULD INCREMENT ITS VALUE TO 3. THIS PROCESS MIGHT CONTINUE UNINTERRUPTED UNTIL THE CPU FOUND AN INSTRUCTION THAT TOLD IT TO "STOP."

A SHARP READER MIGHT BE STARTING TO ASK "WHY HAVE A PROGRAM COUNTER" IF EACH INSTRUCTION FOLLOWS THE NEXT?" THE ANSWER IS SIMPLY THAT THE AVAILABILITY OF A "PROGRAM COUNTER" GIVES THE FREEDOM OF NOT HAVING TO ALWAYS TAKE THE INSTRUCTION AT THE NEXT "ADDRESS" IN MEMORY. THIS

IS BECAUSE THE CONTENTS OF THE "PROGRAM COUNTER" CAN ITSELF BE CHANGED WHEN THE CPU DETECTS AN "INSTRUCTION" THAT DIRECTS IT TO DO SO! THIS ENABLES THE COMPUTER TO BE ABLE TO "JUMP" AROUND TO DIFFERENT SECTIONS IN MEMORY, AND AS WILL BECOME APPARENT LATER, GREATLY INCREASES THE CAPABILITY OF THE MACHINE. THE "PROGRAM COUNTER" IN THE SCELBI-8H SERIES OF MINI-COMPUTERS HAS SOME VERY SPECIAL CAPABILITIES WHICH ADD EVEN MORE POWER TO THE MACHINE AS IT CAN "REMEMBER" A WHOLE GROUP OF MEMORY ADDRESSES WHICH ENABLES THE MACHINE TO PERFORM VERY COMPLEX OPERATIONS REFERRED TO AS "SUBROUTINING." THESE OPERATIONS WILL BE EXPLAINED IN DETAIL FURTHER ON IN THIS MANUAL.

THE "PROGRAM COUNTER" IS ACTUALLY JUST A GROUP OF CELLS IN THE CPU THAT MAY CONTAIN EITHER A BINARY ZERO OR ONE. THE BINARY VALUE IN THE ROW OF CELLS THAT CONSTITUTE THE PROGRAM COUNTER DETERMINES THE "ADDRESS" OF A WORD IN MEMORY. SINCE THE NUMBER OF WORDS IN MEMORY CAN BE VERY LARGE, AND SINCE THE PROGRAM COUNTER MUST BE CAPABLE OF HOLDING THE ADDRESS OF ANY POSSIBLE LOCATION IN MEMORY, THE NUMBER OF CELLS IN A ROW IN THE PROGRAM COUNTER IS LARGER THAN THE NUMBER OF CELLS IN A WORD IN MEMORY. IN THE SCELBI-8H SERIES OF MINI-COMPUTERS THE NUMBER OF CELLS IN THE PROGRAM COUNTER IS 14. SINCE 2 TO THE 14TH POWER IS 16,384, THE PROGRAM COUNTER CAN PRESENT UP TO 16,384 DIFFERENT PATTERNS. EACH PATTERN CAN BE USED TO REPRESENT THE "ADDRESS" OF A WORD IN MEMORY. FIGURE 4 ILLUSTRATES WHAT THE CONTENTS OF THE PROGRAM COUNTER WOULD LOOK LIKE WHEN IT CONTAINED THE ADDRESS FOR A SPECIFIC WORD IN MEMORY. THE ADDRESS THE EXAMPLE DISPLAYS IS "ADDRESS 0" WHICH CAN BE CONSIDERED THE FIRST WORD IN MEMORY. THE READER SHOULD NOTE THAT AN ADDRESS OF ZERO CAN ACTUALLY REPRESENT A WORD IN MEMORY!

```
*********************************************************
*   *   *   *   *   *   '   *   *   *   *   *   *   *   *
* 0 * 0 * 0 * 0 * 0 * 0 ' 0 * 0 * 0 * 0 * 0 * 0 * 0 * 0 *
*   *   *   *   *   *   '   *   *   *   *   *   *   *   *
*********************************************************
```

FIGURE 4

EARLIER IT WAS STATED THAT SOME "INSTRUCTIONS" CAN ACTUALLY CHANGE THE VALUE OF THE PROGRAM COUNTER AND THUS ALLOW A PROGRAM TO "JUMP" TO DIFFERENT SECTIONS IN MEMORY. HOWEVER, THE READER NOW KNOWS THAT A WORD IN MEMORY ONLY CONTAINS EIGHT CELLS, AND YET THE PROGRAM COUNTER CONTAINS 14 CELLS. IN ORDER TO CHANGE THE ENTIRE CONTENTS OF THE PROGRAM COUNTER (BY BRINGING IN WORDS FROM MEMORY) IT IS NECESSARY TO USE MORE THAN ONE MEMORY WORD! THIS CAN BE DONE IF THE PROGRAM COUNTER IS CONSIDERED TO ACTUALLY BE TWO GROUPS OF CELLS CONNECTED TOGETHER. ONE GROUP CONTAINS EIGHT CELLS, AND THE OTHER SIX. IN ORDER TO CHANGE THE CONTENTS OF THE ENTIRE PROGRAM COUNTER, ONE WHOLE EIGHT CELL WORD COULD BE READ FROM A MEMORY LOCATION AND PLACED IN THE RIGHT HAND GROUP OF EIGHT CELLS OF THE PROGRAM COUNTER. THEN ANOTHER EIGHT CELL WORD COULD BE READ FROM MEMORY. SINCE ONLY SIX MORE CELLS ARE NEEDED TO FINISH FILLING THE PROGRAM COUNTER, THE INFORMATION IN TWO OF THE EIGHT CELLS FROM THE SECOND WORD BROUGHT IN FROM MEMORY COULD BE "DISCARDED." IF THE INFORMATION IN THE TWO LEFT MOST CELLS OF THE WORD IN MEMORY WERE THROWN AWAY THEN THE REMAINING SIX CELLS WOULD CONTAIN INFORMATION THAT COULD BE PLACED IN THE SIX UNFILLED LOCATIONS IN THE PROGRAM COUNTER. THAT IS PRECISELY THE PROCEDURE UTILIZED IN THE SCELBI-8H MINI-COMPUTER.

IN ORDER TO MAKE IT EASIER FOR A PERSON WORKING WITH THE MACHINE TO REMEMBER "ADDRESSES" OF WORDS IN MEMORY, A CONCEPT REFERRED TO BY COMPUTER TECHNOLOGISTS AS "PAGING" IS UTILIZED. "PAGING" IS THE ARBITRARY

ASSIGNMENT OF "BLOCKS" OF MEMORY WORDS INTO SECTIONS THAT ARE REFERRED TO FIGURATIVELY AS "PAGES." THE READER SHOULD REALIZE THAT THE ACTUAL PHYSICAL MEMORY UNIT CONSIST OF ALL THE WORDS IN MEMORY - WITH EACH WORD ASSIGNED A NUMERICAL ADDRESS THAT THE MACHINE UTILIZES. AS FAR AS THE MACHINE IS CONCERNED, THE WORDS IN MEMORY ARE ASSIGNED CONSECUTIVE ADDRESSES FROM WORD #0 ON UP TO THE HIGHEST WORD # CONTAINED IN THE MEMORY. HOWEVER, PEOPLE USING COMPUTERS HAVE FOUND IT EASIER TO WORK WITH ADDRESSES BY ARBITRARILY GROUPING "BLOCKS" OF WORDS INTO PAGES. IN THE SCELBI-8H "PAGES" ARE CONSIDERED TO BE "BLOCKS" OF 256 MEMORY WORDS. THE FIRST MEMORY WORD ADDRESS IN A SCELBI-8H MINI-COMPUTER IS AT ADDRESS ZERO (0). PROGRAMMERS COULD REFER TO THIS WORD AS WORD #0 ON PAGE #0. THE 256TH WORD IN MEMORY AS FAR AS THE COMPUTER IS CONCERNED HAS AN ADDRESS OF 255. (NOTE: SINCE THE ADDRESS OF 0 IS ACTUALLY ASSIGNED FOR THE FIRST PHYSICAL WORD IN MEMORY, ALL SUCCEEDING WORDS HAVE AN ADDRESS THAT IS ONE LESS THAN THE PHYSICAL QUANTITY!) A PROGRAMMER COULD REFER TO THIS WORD AS WORD #255 ON PAGE #0. THE 257TH WORD IN MEMORY HAS AN ABSOLUTE ADDRESS OF 256 ("N"TH WORD MINUS ONE SINCE LOCATION 0 CONTAINS A MEMORY WORD) AS FAR AS THE MACHINE IS CONCERNED, BUT A PROGRAMMER COULD REFER TO THAT WORD LOCATION AS BEING ON PAGE #1 AT LOCATION 0! SIMILARLY, THE 513TH WORD IN MEMORY, WHEN THE PAGING CONCEPT IS USED, BECOMES WORD #0 ON PAGE #2 FOR A PROGRAMMER - BUT IT IS JUST 512 AS FAR AS THE MACHINE IS CONCERNED.

THE READER MIGHT HAVE NOTED A NICE COINCIDENCE IN REGARDS TO THE ASSIGNMENT OF "PAGING" FOR THE SCELBI-8H. EACH "PAGE" REFERS TO A "BLOCK" OF MEMORY WORDS THAT CONTAINS 256 LOCATIONS (0 - 255). THE READER WILL RECALL THAT THAT IS EXACTLY THE NUMBER OF DIFFERENT PATTERNS THAT CAN BE SPECIFIED BY A GROUP OF EIGHT BINARY CELLS, AND THERE ARE EIGHT BINARY CELLS IN A MEMORY "WORD." THE RELATIONSHIP IS MORE THAN COINCIDENTAL! NOTE THAT NOW ONE HAS DEVISED A CONVENIENT WAY FOR A PERSON TO BE ABLE TO THINK OF MEMORY ADDRESSES AND AT THE SAME TIME BE ABLE TO SPECIFY A NEW ADDRESS TO THE PROGRAM COUNTER THAT WILL STILL RESULT IN IT CONTAINING AN "ABSOLUTE" ADDRESS THAT THE MACHINE CAN USE. FOR INSTANCE, IF IT WAS DESIRED TO CHANGE THE CONTENTS OF THE 14 CELL PROGRAM COUNTER FROM AN ABSOLUTE ADDRESS OF WORD #0, SAY TO WORD #511, THE FOLLOWING PROCEDURE COULD BE USED: THE PROGRAMMER WOULD FIRST SPECIFY AN INSTRUCTION THAT THE CPU WOULD DECODE AS MEANING "CHANGE THE VALUE IN THE PROGRAM COUNTER." (SUCH AN INSTRUCTION MIGHT BE A "JUMP" INSTRUCTION IN THE SCELBI-8H INSTRUCTION SET.) FOLLOWING THAT INSTRUCTION WOULD BE A WORD THAT HELD THE DESIRED VALUE OF THE "LOW ORDER ADDRESS" OR WORD # WITHIN A "PAGE." SINCE A MEMORY WORD ONLY HAS EIGHT CELLS, SINCE EIGHT CELLS CAN ONLY REPRESENT 256 DIFFERENT PATTERNS, AND SINCE ONE OF THE PATTERNS IS EQUIVALENT TO A VALUE OF ZERO, THEN THE LARGEST NUMBER THE EIGHT CELLS CAN REPRESENT IS 255. HOWEVER, THIS IS THE LARGEST WORD # THAT IS CONTAINED ON A PAGE. THIS VALUE CAN BE PLACED IN THE RIGHT-MOST EIGHT CELLS OF THE PROGRAM COUNTER. NOW IT IS NECESSARY TO COMPLETE THE ADDRESS BY GETTING THE CONTENTS OF ANOTHER WORD FROM MEMORY. THUS, IMMEDIATELY FOLLOWING THE WORD THAT CONTAINED THE "LOW ADDRESS" WOULD BE ANOTHER WORD THAT CONTAINED THE "PAGE #" OF THE ADDRESS THAT THE PROGRAM COUNTER WAS TO CONTAIN. IN THIS CASE THE PAGE NUMBER WOULD BE 1. WHEN THIS VALUE IS PLACED IN THE LEFT SIX CELLS OF THE PROGRAM COUNTER THE PROGRAM COUNTER WOULD CONTAIN THE FOLLOWING PATTERN:

```
*********************************************************
*   *   *   *   *   *   !   *   *   *   *   *   *   *   *
* 0 * 0 * 0 * 0 * 0 * 1 ! 1 * 1 * 1 * 1 * 1 * 1 * 1 * 1 *
*   *   *   *   *   *   !   *   *   *   *   *   *   *   *
*********************************************************
```

FIGURE 5

IF DESIRED, THE READER CAN VERIFY BY USING THE FORMULA PRESENTED PREVIOUSLY FOR DETERMINING THE DECIMAL VALUE OF A BINARY NUMBER, THAT THE PATTERN PRESENTED IN FIGURE 5 CORRESPONDS TO 511, AND THUS, BY USING THE "PAGE #" AND "WORD # ON THE PAGE," EACH OF WHICH WILL FIT IN AN EIGHT CELL MEMORY WORD, A METHOD HAS BEEN DEMONSTRATED THAT WILL RESULT IN THE PROGRAM COUNTER BEING SET TO AN ABSOLUTE ADDRESS FOR A WORD IN MEMORY. FIGURE 6A AND 6B PROVIDE SOME EXAMPLES AS A SUMMARY.

```
          PAGE #0                   WORD #0
            ..                        ..
           .  .                      .  .
......      ......   .........      .........
.                .   .                      .    1ST PHYSICAL WORD IN
0  0  0  0  0  0     0  0  0  0  0  0  0  0      MEMORY HAS AN ABSOLUTE
.                                           .    ADDRESS OF:  0
..................       ....................
                     .  .
                      ..
  ABSOLUTE ADDRESS IN THE PROGRAM COUNTER
```

```
          PAGE #0                  WORD #255
            ..                        ..
           .  .                      .  .
......      ......   .........      .........
.                .   .                      .    256TH PHYSICAL WORD IN
0  0  0  0  0  0     1  1  1  1  1  1  1  1      MEMORY HAS AN ABSOLUTE
.                                           .    ADDRESS OF:  255
..................       ....................
                     .  .
                      ..
  ABSOLUTE ADDRESS IN THE PROGRAM COUNTER
```

```
          PAGE #1                   WORD #0
            ..                        ..
           .  .                      .  .
......      ......   .........      .........
.                .   .                      .    257TH PHYSICAL WORD IN
0  0  0  0  0  1     0  0  0  0  0  0  0  0      MEMORY HAS AN ABSOLUTE
.                                           .    ADDRESS OF:  256
..................       ....................
                     .  .
                      ..
  ABSOLUTE ADDRESS IN THE PROGRAM COUNTER
```

```
          PAGE #1                   WORD #1
            ..                        ..
           .  .                      .  .
......      ......   .........      .........
.                .   .                      .    258TH PHYSICAL WORD IN
0  0  0  0  0  1     0  0  0  0  0  0  0  1      MEMORY HAS AN ABSOLUTE
.                                           .    ADDRESS OF:  257
..................       ....................
                     .  .
                      ..
  ABSOLUTE ADDRESS IN THE PROGRAM COUNTER
```

FIGURE 6A

```
     PAGE #1                   WORD #2
        ..                        ..
       .  .                      .  .
......      ......   .........      .........
.                .   .                      .    259TH PHYSICAL WORD IN
Ø  Ø  Ø  Ø  Ø  1     Ø  Ø  Ø  Ø  Ø  Ø  1  Ø     MEMORY HAS AN ABSOLUTE
.                                           .    ADDRESS OF:  258
...................       ..................
                   .  .
                    ..
 ABSOLUTE ADDRESS IN THE PROGRAM COUNTER
```

```
     PAGE #1                 WORD #255
        ..                        ..
       .  .                      .  .
......      ......   .........      .........
.                .   .                      .    512TH PHYSICAL WORD IN
Ø  Ø  Ø  Ø  Ø  1     1  1  1  1  1  1  1  1     MEMORY HAS AN ABSOLUTE
.                                           .    ADDRESS OF:  511
...................       ..................
                   .  .
                    ..
 ABSOLUTE ADDRESS IN THE PROGRAM COUNTER
```

```
     PAGE #2                   WORD #Ø
        ..                        ..
       .  .                      .  .
......      ......   .........      .........
.                .   .                      .    513TH PHYSICAL WORD IN
Ø  Ø  Ø  Ø  1  Ø     Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø     MEMORY HAS AN ABSOLUTE
.                                           .    ADDRESS OF:  512
...................       ..................
                   .  .
                    ..
 ABSOLUTE ADDRESS IN THE PROGRAM COUNTER
```

```
     PAGE #3                 WORD #255
        ..                        ..
       .  .                      .  .
......      ......   .........      .........
.                .   .                      .    1Ø24TH PHYSICAL WORD IN
Ø  Ø  Ø  Ø  1  1     1  1  1  1  1  1  1  1     MEMORY HAS AN ABSOLUTE
.                                           .    ADDRESS OF:  1Ø23
...................       ..................
                   .  .
                    ..
 ABSOLUTE ADDRESS IN THE PROGRAM COUNTER
```

FIGURE 6B

BY NOW THE READER SHOULD HAVE A PRETTY GOOD UNDERSTANDING OF THE CONCEPTS REGARDING THE ORGANIZATION OF MEMORY INTO ELECTRICAL CELLS WHICH CAN BE IN ONE OF TWO POSSIBLE STATES; THE GROUPING OF THESE CELLS INTO "WORDS" WHICH CAN HOLD PATTERNS WHICH THE CPU CAN RECOGNIZE AS SPECIFYING PARTICULAR OPERATIONS; AND THE OPERATION OF A "PROGRAM COUNTER" WHICH IS ABLE TO HOLD THE "ADDRESS" OF A WORD IN MEMORY FROM WHICH THE CPU IS TO OBTAIN AN INSTRUCTION.

IT IS NOW TIME TO DISCUSS THE OPERATION OF THE "SCRATCH PAD" AREA FOR A COMPUTER - THE ACCUMULATOR (AND SOME ADDITIONAL "MANIPULATING REGISTERS" IN THE SCELBI-8H MINI-COMPUTER.)

AS WAS POINTED OUT EARLIER IN THIS CHAPTER THERE IS A SECTION OF A COMPUTER THAT IS USED TO PERFORM CALCULATIONS IN AND WHICH CAN HOLD INFORMATION WHILE THE CPU IS IN THE PROCESS OF "FETCHING" ANOTHER INSTRUCTION FROM THE MEMORY. THE PORTION WAS TERMED AN "ACCUMULATOR" BECAUSE IT COULD "ACCUMULATE" INFORMATION OBTAINED FROM THE CPU PERFORMING A SERIES OF INSTRUCTIONS UNTIL SUCH TIME AS THE CPU WAS DIRECTED TO TRANSFER THE INFORMATION ELSEWHERE (OR DISCARD IT.) THE ACCUMULATOR IS ALSO CONSIDERED TO BE THE PRIMARY "MATHEMATICAL" CENTER FOR COMPUTER OPERATIONS FOR IT IS THE PLACE WHERE ADDITIONS, SUBTRACTIONS, AND VARIOUS OTHER MATHEMATICALLY ORIENTED OPERATIONS (SUCH AS BØØLEAN ALGEBRA) ARE GENERALLY PERFORMED UNDER PROGRAM CONTROL.

THE CONCEPT OF AN "ACCUMULATOR" IS NOT DIFFICULT TO UNDERSTAND AND ITS PHYSICAL STRUCTURE CAN BE READILY EXPLAINED. THE ACTUAL CONTROL OF AN ACCUMULATOR BY THE CPU CAN BE QUITE COMPLEX, BUT THESE COMPLEX ELECTRONIC MANIPULATIONS DO NOT HAVE TO BE UNDERSTOOD BY THE COMPUTER USER. IT IS ONLY NECESSARY TO KNOW THE "END RESULTS" OF THE VARIOUS OPERATIONS THAT CAN BE PERFORMED WITHIN AN ACCUMULATOR. THE PHYSICAL STRUCTURE AS WELL AS THE KINDS OF OPERATIONS THAT CAN BE PERFORMED IN THE ACCUMULATOR WILL BE DISCUSSED IN THIS SECTION, AND IN ADDITION THE READER WILL LEARN ABOUT SOME ADDITIONAL "PARTIAL ACCUMULATORS" WHICH ARE IN THE SCELBI-8H SERIES OF MINI-COMPUTERS AND WHICH SERVE MANY VALUABLE PURPOSES.

THE ACCUMULATOR IN A SCELBI-8H MACHINE CAN BE CONSIDERED AS A GROUP OF EIGHT "MEMORY CELLS" SIMILAR TO A "WORD" IN MEMORY EXCEPT THAT THE INFORMATION IN THE CELLS CAN BE MANIPULATED IN MANY WAYS THAT ARE NOT DIRECTLY POSSIBLE IN A WORD IN MEMORY.

```
    B 7    B 6    B 5    B 4    B 3    B 2    B 1    B Ø

*************************************************
*      *      *      *      *      *      *      *      *
*  1   *  Ø   *  1   *  Ø   *  1   *  Ø   *  1   *  Ø   *
*      *      *      *      *      *      *      *      *
*************************************************
```

FIGURE 7

FIGURE 7 SHOWS A COLLECTION OF EIGHT BINARY CELLS CONTAINING ONES AND ZEROS TO REPRESENT AN ACCUMULATOR. THE CELLS ARE NUMBERED FROM LEFT TO RIGHT STARTING WITH "B7" DOWN TO "BØ." THE DESIGNATIONS REFER TO "BIT POSITIONS" WITHIN THE ACCUMULATOR. NOTE THAT THE RIGHT-MOST CELL IS DESIGNATED B Ø AND THE EIGHTH CELL (LEFT MOST CELL) IS DESIGNATED B 7. THE READER SHOULD BECOME THOROUGHLY FAMILIAR WITH THE CONCEPT OF ASSIGNING THE REFERENCE OF "ZERO" TO THE RIGHT-MOST BIT POSITION IN A ROW OF CELLS (SIMILAR TO THE CONCEPT OF ASSIGNING A REFERENCE OF ZERO TO THE FIRST ADDRESS OF A WORD ON A PAGE IN MEMORY) AS THE CONVENTION IS FREQUENTLY USED BY COMPUTER TECHNOLOGISTS. THE CONVENTION CAN BE CONFUSING FOR THE BEGINNER WHO FAILS TO REMEMBER THAT THE PHYSICAL QUANTITY IS ONE MORE THAN THE REFERENCE DESIGNATION. THE CONVENTION OF LABELING THE FIRST PHYSICAL POSITION AS ZERO MAKES MUCH MORE SENSE ONCE THE READER LEARNS TO THINK IN TERMS OF THE BINARY NUMBERING SYSTEM AND THOROUGHLY REALIZES THAT THE "ZERO" REFERRED TO SO FREQUENTLY IN COMPUTER WORK WHEN DISCUSSING ACTUAL OPERATIONS ACTUALLY REPRESENTS A PHYSICAL STATE (THE STATUS OF AN ELECTRONIC SWITCH) AND

DOES NOT NECESSARILY IMPLY THE MATHEMATICAL NOTION OF "NOTHING." THE CONCEPT OF ASSIGNING A BIT DESIGNATION TO THE POSITIONS OF THE CELLS WITHIN THE ACCUMULATOR WILL ALLOW THE READER TO FOLLOW EXPLANATIONS OF VARIOUS ACCUMULATOR OPERATIONS.

ONE OF THE MOST FUNDAMENTAL AND MOST OFTEN USED OPERATIONS OF AN ACCUMULATOR IS FOR IT TO SIMPLY HOLD A NUMBER WHILE THE CPU OBTAINS A SECOND OPERATOR. IN THE SCELBI-8H THE ACCUMULATOR CAN BE "LOADED" WITH A VALUE OBTAINED FROM A LOCATION IN MEMORY OR ONE OF THE "PARTIAL ACCUMULATORS." IT CAN THEN HOLD THIS VALUE UNTIL IT IS TIME TO PERFORM SOME OTHER OPERATION WITH THE ACCUMULATOR. (IT WILL BECOME APPARENT LATER THAT THE ACCUMULATOR IN THE SCELBI-8H CAN ALSO RECEIVE INFORMATION FROM EXTERNAL DEVICES.)

PERHAPS THE SECOND MOST OFTEN USED OPERATION OF AN ACCUMULATOR IS TO HAVE IT PERFORM MATHEMATICAL OPERATIONS SUCH AS ADDITION OR SUBTRACTION WITH THE VALUE IT CONTAINS AT THE TIME THE FUNCTION IS PERFORMED AND THE CONTENTS OF A MEMORY LOCATION OR ONE OF THE "PARTIAL ACCUMULATORS." THUS IF THE ACCUMULATOR CONTAINED THE BINARY EQUIVALENT OF THE DECIMAL NUMBER 5, AND AN INSTRUCTION TO ADD THE CONTENTS OF A SPECIFIC MEMORY LOCATION WHICH CONTAINED THE BINARY EQUIVALENT OF THE DECIMAL NUMBER 3 WAS ENCOUNTERED, THE ACCUMULATOR WOULD END UP WITH THE VALUE OF 8 IN BINARY FORM AS SHOWN IN FIGURE 8.

```
   B 7    B 6    B 5    B 4    B 3    B 2    B 1    B 0

*************************************************
*      *      *      *      *      *      *      *      *     ORIGINAL CONTENTS
*  0   *  0   *  0   *  0   *  0   *  1   *  0   *  1   *          OF THE
*      *      *      *      *      *      *      *      *        ACCUMULATOR
*************************************************

*************************************************
*      *      *      *      *      *      *      *      *      CONTENTS OF THE
*  0   *  0   *  0   *  0   *  0   *  0   *  1   *  1   *     SPECIFIED WORD IN
*      *      *      *      *      *      *      *      *           MEMORY
*************************************************

*************************************************
*      *      *      *      *      *      *      *      *        FINAL RESULTS
*  0   *  0   *  0   *  0   *  1   *  0   *  0   *  0   *     AFTER THE ADDITION
*      *      *      *      *      *      *      *      *     IN THE ACCUMULATOR
*************************************************
```

FIGURE 8

PERHAPS THE NEXT MOST FREQUENTLY USED GROUP OF OPERATIONS FOR THE ACCUMULATOR IS FOR IT TO PERFORM "BOOLEAN" MATHEMATICAL OPERATIONS BETWEEN ITSELF AND/OR OTHER "PARTIAL ACCUMULATORS" OR WORDS IN MEMORY. THESE OPERATIONS IN THE SCELBI-8H INCLUDE THE LOGICAL "AND," "OR," AND "EXCLUSIVE OR" OPERATIONS. THE USE OF THESE "BOOLEAN" MATHEMATICAL FUNCTIONS WILL BE DESCRIBED IN MORE DETAIL IN OTHER PARTS OF THIS PUBLICATION.

ANOTHER IMPORTANT CAPABILITY OF THE ACCUMULATOR IS ITS ABILITY TO "ROTATE" ITS CONTENTS. IN THE SCELBI-8H THE CONTENTS OF THE ACCUMULATOR CAN BE ROTATED EITHER TO THE RIGHT OR LEFT. THIS CAPABILITY HAS MANY USEFUL FUNCTIONS - AND IS ONE METHOD BY WHICH MATHEMATICAL MULTIPLICATION OR DIVISION CAN BE PERFORMED. FIGURE 9 ILLUSTRATES THE CONCEPT

OF "ROTATING" THE CONTENTS OF THE ACCUMULATOR.

```
  B 7  B 6  B 5  B 4  B 3  B 2  B 1  B Ø

*****************************************
*    *    *    *    *    *    *    *    *    ORIGINAL CONTENTS
*  Ø *  Ø *  Ø *  Ø *  Ø *  Ø *  1 *  Ø *    OF THE ACCUMULATOR
*    *    *    *    *    *    *    *    *    (EQUAL TO DECIMAL 2)
*****************************************

*****************************************       RESULT WHEN THE
*    *    *    *    *    *    *    *    *         ACCUMULATOR
*  Ø *  Ø *  Ø *  Ø *  Ø *  1 *  Ø *  Ø *      IS ROTATED TO THE
*    *    *    *    *    *    *    *    *        LEFT ONE TIME
*****************************************    (VALUE NOW EQUAL 4)

*****************************************       RESULT WHEN THE
*    *    *    *    *    *    *    *    *     ACCUMULATOR IS NOW
*  Ø *  Ø *  Ø *  Ø *  Ø *  Ø *  Ø *  1 *    ROTATED TO THE RIGHT
*    *    *    *    *    *    *    *    *          TWO TIMES
*****************************************    (VALUE NOW EQUAL 1)

*****************************************       NOTE THAT IF A
*    *    *    *    *    *    *    *    *     ROTATE RIGHT COMMAND
*  1 *  Ø *  Ø *  Ø *  Ø *  Ø *  Ø *  Ø *   IS DONE AGAIN THAT THE
*    *    *    *    *    *    *    *    *     "1" IN POSITION B Ø
*****************************************    WILL APPEAR AT B 7 !!

*****************************************       AND THAT NOW A
*    *    *    *    *    *    *    *    *     ROTATE LEFT COMMAND
*  Ø *  Ø *  Ø *  Ø *  Ø *  Ø *  Ø *  1 *      WOULD RESTORE THE
*    *    *    *    *    *    *    *    *     "1" IN POSITION B 7
*****************************************         BACK TO B Ø !
```

FIGURE 9

THE ASTUTE READER MAY NOTICE THAT THE ACCUMULATOR ROTATE CAPABILITY ALSO ENABLES THE ACCUMULATOR TO EMULATE A "SHIFT REGISTER" WHICH CAN BE A VALUABLE FUNCTION IN MANY PRACTICAL APPLICATIONS OF THE COMPUTER.

THE ACCUMULATOR IN THE SCELBI-8H SERVES ANOTHER EXTREMELY POWERFUL FUNCTION. WHEN CERTAIN OPERATIONS ARE PERFORMED WITH THE ACCUMULATOR THE SCELBI-8H MINI-COMPUTER IS CAPABLE OF EXAMINING THE RESULTS AND WILL THEN "SET" OR "CLEAR" A SPECIAL GROUP OF "FLAGS." OTHER INSTRUCTIONS CAN THEN TEST THE STATUS OF THE SPECIAL "FLAGS" AND PERFORM OPERATIONS BASED ON THE PARTICULAR SETTING(S) OF THE "FLAGS." IN THIS MANNER THE MACHINE IS CAPABLE OF "MODIFYING" ITS BEHAVIOR WHEN IT PERFORMS OPERATIONS DEPENDING ON THE RESULTS IT OBTAINS AT THE TIME THE OPERATION IS PERFORMED!

THERE ARE FOUR SPECIAL FLAGS IN THE SCELBI-8H MINI-COMPUTER WHICH ARE MANIPULATED BY THE RESULTS OF OPERATIONS WITH THE ACCUMULATOR (AND IN SEVERAL SPECIAL CASES BY OPERATIONS WITH "PARTIAL ACCUMULATORS). THESE FOUR FLAGS ARE DESCRIBED IN DETAIL BELOW.

THE "CARRY FLAG" CAN BE CONSIDERED AS A ONE BIT (CELL) EXTENSION OF THE ACCUMULATOR REGISTER. THIS FLAG IS CHANGED IF THE CONTENTS OF THE ACCUMULATOR SHOULD "OVERFLOW" DURING AN ADDITION OPERATION (OR "UNDERFLOW" DURING A SUBTRACTION OPERATION). ALSO, THE "CARRY BIT" CAN BE

UTILIZED AS AN EXTENSION OF THE ACCUMULATOR FOR CERTAIN TYPES OF "ROTATE" COMMANDS.

THE "SIGN FLAG" IS SET TO A LOGIC STATE OF "1" WHEN THE MOST SIGNIFICANT BIT (MSB) OF THE ACCUMULATOR (OR PARTIAL ACCUMULATOR) IS A "1" AFTER CERTAIN TYPES OF INSTRUCTIONS HAVE BEEN PERFORMED. THE NAME OF THIS FLAG DERIVES FROM THE CONCEPT OF USING TWO'S COMPLEMENT ARITHMETIC IN A REGISTER WHERE THE MSB IS USED TO DESIGNATE THE SIGN OF THE NUMBER IN THE REMAINING BIT POSITIONS OF THE REGISTER - CONVENTIONALLY, A "1" IN THE MSB DESIGNATES THE NUMBER AS A "NEGATIVE" NUMBER. IF THE MSB OF THE ACCUMULATOR (OR PARTIAL ACCUMULATOR) IS "Ø" AFTER CERTAIN OPERATIONS THEN THE "SIGN FLAG" IS ZERO (INDICATING THAT THE NUMBER IN THE REGISTER IS A POSITIVE NUMBER BY TWO'S COMPLEMENT CONVENTION.)

THE "ZERO FLAG" IS SET TO A LOGIC STATE OF "1" IF ALL THE BITS IN THE ACCUMULATOR (OR PARTIAL ACCUMULATOR) ARE SET TO ZERO AFTER CERTAIN TYPES OF OPERATIONS HAVE BEEN EXECUTED. IT IS SET TO "Ø" IF ANY ONE OF THE BITS IS A LOGIC ONE AFTER THESE SAME OPERATIONS. THUS THE "ZERO FLAG" CAN BE UTILIZED TO DETERMINE WHEN THE VALUE IN A PARTICULAR REGISTER IS ZERO.

THE "PARITY FLAG" IS SET TO A "1" AFTER CERTAIN TYPES OF OPERATIONS WITH THE ACCUMULATOR (OR PARTIAL ACCUMULATORS) WHEN THE NUMBER OF BITS IN THE REGISTER THAT ARE A LOGIC ONE IS AN EVEN VALUE (WITHOUT REGARD TO THE POSITIONS OF THE BITS). THE "PARITY FLAG" IS SET TO "Ø" AFTER THESE SAME OPERATIONS IF THE NUMBER OF BITS IN THE REGISTER THAT ARE A LOGIC ONE IS AN ODD VALUE (1, 3, 5 OR 7). THE "PARITY FLAG" CAN BE ESPECIALLY VALUABLE WHEN DATA FROM EXTERNAL DEVICES IS BEING RECEIVED BY THE SCELBI-8H MINI-COMPUTER TO TEST FOR CERTAIN TYPES OF "TRANSMISSION ERRORS" ON THE INFORMATION BEING RECEIVED.

THERE WILL BE DETAILED DISCUSSION ON THE USE OF THE FOUR FLAGS IN OTHER SECTIONS OF THIS PUBLICATION. THEIR INTRODUCTION AT THIS TIME IS SIMPLY TO INFORM THE READER OF THEIR AVAILABILITY AND TO INDICATE THEIR RELATION TO THE ACCUMULATOR (AND PARTIAL ACCUMULATORS) AS THEIR FUNCTIONS GREATLY INCREASE THE CAPABILITIES OF THE MACHINE - AND THEY ARE FUNCTIONALLY CONNECTED WITH THE ACCUMULATOR (AND PARTIAL ACCUMULATORS.)

IN ADDITION TO THE FULL ACCUMULATOR PREVIOUSLY DISCUSSED THERE ARE SIX OTHER 8 BIT REGISTERS IN THE SCELBI-8H MINI-COMPUTER REFERRED TO AS "PARTIAL ACCUMULATORS" BECAUSE THEY ARE CAPABLE OF PERFORMING TWO SPECIAL FUNCTIONS NORMALLY ASSOCIATED WITH AN ACCUMULATOR (IN ADDITION TO SIMPLY SERVING AS TEMPORARY STORAGE REGISTERS). THE FULL ACCUMULATOR WILL OFTEN BE ABBREVIATED IN THIS MANUAL AS "ACC" OR "REGISTER A." THE SIX "PARTIAL ACCUMULATORS" WILL BE REFERRED TO AS "REGISTERS B, C, D, E, H AND L."

REGISTERS B, C, D, E, H AND L ARE ALL CAPABLE, UPON BEING DIRECTED TO DO SO BY A SPECIFIC INSTRUCTION, OF EITHER INCREMENTING OR DECREMENTING THEIR CONTENTS BY ONE. THIS CAPABILITY ALLOWS THEM TO BE USED AS "COUNTERS" AND "POINTERS" WHICH ARE OFTEN OF TREMENDOUS VALUE IN COMPUTER PROGRAMS. WHAT MAKES THEM ESPECIALLY VALUABLE IN THE SCELBI-8H MACHINE IS THAT WHEN THEIR CONTENTS ARE INCREMENTED OR DECREMENTED THE IMMEDIATE RESULTS OF THAT REGISTER WILL AFFECT THE STATUS OF THE "ZERO," "SIGN," AND "PARITY" FLAGS DISCUSSED ABOVE. THUS IT IS POSSIBLE FOR THE PARTICULAR CONTENTS OF THESE REGISTERS TO AFFECT THE OPERATION OF THE COMPUTER DURING THE COURSE OF A PROGRAMS OPERATION AND THEY CAN BE USED TO "GUIDE" OR MODIFY A SEQUENCE OF OPERATIONS BASED ON CONDITIONS FOUND AT THE ACTUAL TIME A PROGRAM IS EXECUTED.

IT SHOULD BE NOTED THAT REGISTERS B, C, D, E, H AND L ARE CAPABLE OF BEING INCREMENTED AND DECREMENTED - BUT THE FULL ACCUMULATOR - REGISTER A - CANNOT PERFORM THOSE TWO FUNCTIONS IN THE SAME PRECISE MANNER. (THE FULL ACCUMULATOR CAN BE INCREMENTED OR DECREMENTED BY ANY VALUE BY SIMPLY ADDING OR SUBTRACTING THE DESIRED VALUE. THERE IS NOT, HOWEVER, A SIMPLE INCREMENT OR DECREMENT BY ONE INSTRUCTION FOR USE WITH THE FULL ACCUMULATOR!)

TWO OF THE PARTIAL ACCUMULATORS, REGISTERS H AND L, SERVE AN ADDITIONAL PURPOSE IN THE SCELBI-8H MINI-COMPUTER. THESE TWO REGISTERS CAN BE USED TO DIRECTLY "POINT" TO A SPECIFIC WORD IN MEMORY SO THAT THE COMPUTER MAY OBTAIN OR DEPOSIT INFORMATION IN A DIFFERENT PART OF MEMORY THAN THAT IN WHICH A PROGRAM IS ACTUALLY BEING EXECUTED. THE READER SHOULD RECALL THAT A SPECIAL PART OF THE CENTRAL PROCESSOR UNIT (CPU) TERMED THE PROGRAM COUNTER IS USED TO TELL THE COMPUTER WHERE TO OBTAIN THE NEXT INSTRUCTION WHILE EXECUTING A PROGRAM. THE PROGRAM COUNTER WAS EFFECTIVELY A "DOUBLE WORD LENGTH" REGISTER THAT COULD HOLD THE VALUE OF ANY POSSIBLE ADDRESS IN MEMORY. THE PROGRAM COUNTER IS ALWAYS USED TO TELL THE MACHINE WHERE TO OBTAIN THE NEXT INSTRUCTION. HOWEVER, IT IS OFTEN DESIRABLE TO HAVE THE MACHINE OBTAIN SOME INFORMATION - SUCH AS A "DATA WORD" - FROM A LOCATION IN MEMORY THAT IS NOT CONNECTED WITH WHERE THE NEXT INSTRUCTION TO BE PERFORMED IS LOCATED. THIS CAN BE ACCOMPLISHED BY SIMPLY LOADING "REGISTER H" WITH THE "HIGH ADDRESS" (PAGE) PORTION OF AN ADDRESS IN MEMORY, THEN LOADING "REGISTER L" WITH THE "LOW ADDRESS" PORTION OF AN ADDRESS IN MEMORY, AND THEN UTILIZING ONE OF A CLASS OF COMMANDS THAT WILL DIRECT THE CPU TO FETCH INFORMATION FROM OR DEPOSIT INFORMATION INTO THE LOCATION IN MEMORY THAT IS SPECIFIED ("POINTED TO") BY THE "H" AND "L" REGISTER CONTENTS. THIS INFORMATION FLOW CAN BE FROM/TO THE LOCATION SPECIFIED IN MEMORY AND ANY OF THE CPU REGISTERS.

AT THIS TIME IT WOULD BE BENEFICIAL FOR THE READER TO STUDY FIGURE 10 SHOWN ON THE NEXT PAGE. FIGURE 10 IS AN EXPANDED BLOCK DIAGRAM OF FIGURE 2B AND SHOWS THE UNITS OF THE SCELBI-8H MINI-COMPUTER WHICH HAVE BEEN PRESENTED IN THE PREVIOUS SEVERAL PAGES.

UNTIL NOW NO MENTION HAS BEEN MADE OF HOW INFORMATION IS PUT INTO OR RECEIVED FROM A COMPUTER. NATURALLY, THIS IF A VERY VITAL PART OF A COMPUTER BECAUSE THE MACHINE WOULD BE RATHER USELESS IF PEOPLE COULD NOT PUT INFORMATION INTO THE MACHINE UPON WHICH CALCULATIONS OR PROCESSING COULD BE DONE, AND RECEIVE INFORMATION BACK FROM THE MACHINE WHEN THE OPERATION(S) HAD BEEN PERFORMED!

COMMUNICATIONS BETWEEN THE COMPUTER AND EXTERNAL DEVICES - WHETHER THOSE DEVICES BE SIMPLE SWITCHES, OR TRANSDUCERS, OR TELETYPE MACHINES, OR CATHODE-RAY-TUBE DISPLAY UNITS, OR KEYBOARDS, OR "MAG-TAPE" AND "DISK" SYSTEMS - OR WHATEVER, ARE COMMONLY REFERRED TO AS INPUT/OUTPUT OPERATIONS AND ARE COLLECTIVELY REFERRED TO IN ABBREVIATED FORM AS "I/O" TRANSFERS.

IN THE SCELBI-8H SERIES OF MINI-COMPUTERS ALL "I/O" TRANSFERS ARE MADE BETWEEN EXTERNAL "I/O PORTS" (WHICH CONNECT TO EXTERNAL DEVICES V.I.A. APPROPRIATE ELECTRONIC CONNECTIONS) AND THE FULL ACCUMULATOR IN THE COMPUTER. THE STANDARD SCELBI-8H CAN RECEIVE INFORMATION FROM SIX DIFFERENT "INPUT PORTS" (THE INFORMATION WILL ARRIVE AT THE FULL ACCUMULATOR) OR SEND INFORMATION FROM REGISTER A (THE FULL ACCUMULATOR) TO ANY ONE OF EIGHT DIFFERENT "OUTPUT PORTS." (SPECIAL SYSTEMS CAN HAVE GREATLY EXPANDED I/O CAPABILITY). THIS I/O STRUCTURE MEANS THAT A WHOLE GROUP OF DEVICES CAN BE SIMULTAINEOUSLY HOOKED UP TO THE MINI-COMPUTER AND THE COMPUTER USED TO RECEIVE INFORMATION FROM OR TRANSMIT INFORMATION TO A VARIETY OF DEVICES AS DIRECTED BY A "PROGRAM." A SPECIAL SET

```
*********************************
*          M E M O R Y          *
*********************************
* 0 * 1 * 0 * 0 * 0 * 1 * 0 * 0 *    WORD #1 AT PAGE 0 LOC 0
*********************************
* 0 * 0 * 0 * 0 * 0 * 0 * 0 * 0 *    WORD #2 AT PAGE 0 LOC 1
*********************************
* 0 * 0 * 0 * 0 * 0 * 0 * 0 * 1 *    WORD #3 AT PAGE 0 LOC 2
*********************************
* 1 * 1 * 0 * 0 * 0 * 1 * 1 * 1 *                .
*********************************
* 0 * 0 * 1 * 1 * 0 * 0 * 0 * 1 *                .
*********************************
* 1 * 1 * 1 * 1 * 1 * 0 * 0 * 0 *                .
*********************************
* 1 * 1 * 1 * 1 * 1 * 1 * 1 * 1 *    WORD #X AT PAGE Y LOC Z
*********************************
        .             .
        .             .
        .             .
        .             .
*********************************
*        PROGRAM COUNTER        *
*********************************
*                               *
*            CENTRAL            *
*                               *            CPU WITH
*           PROCESSOR           *        PROGRAM COUNTER
*                               *    AND REGISTER STATUS FLAGS
*             UNIT              *
*                               *
*********************************
*  FLAGS: "C," "Z," "S," & "P"  *
*********************************
        .             .
        .             .
        .             .
        .             .
*********************************
*         REGISTER  "A"         *   =    FULL ACCUMULATOR
*********************************
*         REGISTER  "B"         *   =  PARTIAL ACCUMULATOR
*********************************
*         REGISTER  "C"         *   =  PARTIAL ACCUMULATOR
*********************************
*         REGISTER  "D"         *   =  PARTIAL ACCUMULATOR
*********************************
*         REGISTER  "E"         *   =  PARTIAL ACCUMULATOR
*********************************
* REG "H" & MEMORY PAGE POINTER *   =  PARTIAL ACCUMULATOR
*********************************
* REG "L" & LOW ADDRESS POINTER *   =  PARTIAL ACCUMULATOR
*********************************
```

FIGURE 10

OF COMMANDS ARE USED TO INSTRUCT THE COMPUTER AS TO WHICH "I/O PORT" IS TO BE OPERATED AT ANY PARTICULAR INSTANT. WITH APPROPRIATE PROGRAMMING IT IS THEN POSSIBLE TO HAVE THE COMPUTER "COMMUNICATE" WITH A LARGE VARIETY OF DEVICES IN AN ESSENTIALLY "AUTOMATIC" MODE - FOR INSTANCE RECEIVING INFORMATION FROM A DIGITAL MULTI-METER AT SPECIFIED TIMES, THEN POSSIBLY PERFORMING SOME AVERAGING CALCULATIONS, AND THEN OUTPUTTING RESULTS TO A TELETYPE MACHINE WITHOUT HUMAN INTERVENTION. OR, IN OTHER APPLICATIONS - INFORMATION FROM A HUMAN OPERATOR CAN BE TYPED IN TO THE MACHINE USING A TYPE-WRITER-LIKE KEYBOARD. IN ITS SIMPLIST FORM A GROUP OF SWITCHES CAN BE USED AS AN INPUT DEVICE AND A GROUP OF LAMPS USED AS AN OUTPUT DEVICE FOR THE SCELBI-8H MINI-COMPUTER!

HOWEVER, A MORE SOPHISTICATED SYSTEM USED IN MANY APPLICATIONS WOULD BE TO USE A TELETYPE MACHINE OR A COMBINATION OF A KEYBOARD AND A CATHODE-RAY-TUBE (CRT) DISPLAY ATTACHED TO INPUT AND OUTPUT PORTS TO SERVE AS THE PRIMARY MEANS OF I/O. A PERSON CAN THUS TYPE INFORMATION ON THE KEYBOARD WHICH WILL PASS IT INTO THE COMPUTER, AND THE COMPUTER CAN DISPLAY THE RESULTS OF ITS OPERATIONS ON THE CRT DISPLAY (WHICH CAN INCIDENTLY, BE MADE FROM AN ORDINARY OSCILLOSCOPE AND A SPECIAL SCELBI CRT INTERFACE UNIT!)

PERHAPS THE MOST WONDERFUL AND EXCITING ASPECT ABOUT A DIGITAL COMPUTER IS ITS TREMENDOUS VERSATILITY. IT HAS BEEN SAID THAT THE COMPUTER IS THE MOST VERSATILE MACHINE IN EXISTENCE AND THAT ITS APPLICATIONS ARE LIMITED ONLY BY MAN'S ABILITY TO DEVELOP PROGRAMS THAT DIRECT THE OPERATION OF THE MACHINE. IT IS UNDOUBTABLY ONE OF THE BEST MACHINES FOR ALLOWING MAN TO EXERCISE AND TEST HIS CREATIVE POWERS THROUGH THE DEVELOPMENT OF PROGRAMS THAT DIRECT THE MACHINE TO PERFORM COMPLEX OPERATIONS THAT CAN NOT ONLY CONTROL OTHER MACHINES, OR PERFORM CALCULATIONS MANY TIMES FASTER THAN HUMANLY POSSIBLE - BUT BECAUSE IT CAN BE USED TO "SIMULATE" OR "MODEL" OTHER SYSTEMS THAT IT MIGHT BE IMPRACTICAL TO BUILD FOR PURELY EXPERIMENTAL PURPOSES. THUS MAN CAN CREATE A "MODEL" IN A COMPUTER PROGRAM AND ACTUALLY "PLAY" WITH THE SYNTHETIC MODEL WITHOUT ACTUALLY BUILDING THE PHYSICAL DEVICE!

THE COMPUTER'S GREAT VERSATILITY COMES ABOUT BECAUSE THE MACHINE IS CAPABLE OF EXECUTING A LARGE GROUP OF INSTRUCTIONS IN AN ESSENTIALLY LIMITLESS SERIES OF COMBINATIONS - THESE SERIES OF INSTRUCTIONS ARE STORED IN THE MEMORY BANK(S) OF THE COMPUTER - AND A NEW SERIES OF INSTRUCTIONS CAN BE PLACED IN THE MEMORY BANK(S) WHENEVER DESIRED. IN FACT, THE MEMORY BANK(S) CAN OFTEN HOLD SEVERAL COMPLETLY UNRELATED "PROGRAMS" IN DIFFERENT SECTIONS AND THUS ONE CAN HAVE A MACHINE THAT PERFORMS TOTALLY UNRELATED TASKS SIMPLY BY PUSHING A FEW BUTTONS AND THEREBY DIRECTING THE MACHINE TO START EXECUTING A NEW PROGRAM IN A DIFFERENT SECTION OF MEMORY!

THE DIGITAL MINI-COMPUTER IS CAPABLE OF PROVIDING SERVICES TO PEOPLE FROM ALL WALKS OF LIFE! A PERSON NEED ONLY CHOOSE (OR DEVELOP) PROGRAMS AND CONNECT EXTERNAL INSTRUMENTS THAT WILL PROVIDE THE CAPABILITIES DESIRED.

FOR INSTANCE, A SCIENTIST MIGHT PUT A MATHEMATICAL CALCULATOR PROGRAM INTO THE COMPUTER'S MEMORY AND USE THE MINI-COMPUTER AS A SOPHISTICATED ELECTRONIC CALCULATOR BY USING A CALCULATOR TYPE KEYBOARD AS AN INPUT DEVICE AND A CRT DISPLAY AS AN OUTPUT DEVICE ON WHICH TO RECEIVE THE ANSWERS TO COMPLEX MATHEMATICAL CALCULATIONS WHICH THE COMPUTER PERFORMS. AFTER USING THE COMPUTER AS A CALCULATOR FOR A PERIOD OF TIME THE SCIENTIST MIGHT DECIDE TO UTILIZE THE SAME COMPUTER TO AUTOMATICALLY RECORD DATA FROM INSTRUMENTS DURING AN EXPERIMENT. BY SIMPLY PUTTING A DIFFERENT PROGRAM IN THE COMPUTER'S MEMORY AND PLUGGING IN SOME PERIPHERAL MEASURING INSTRUMENTS INTO THE COMPUTER'S I/O PORTS, THE

SCIENTIST COULD HAVE THE COMPUTER PERIODICALLY MAKE MEASUREMENTS WHILE HE WENT OUT TO LUNCH AND SAVE THE RESULTS IN IT'S MEMORY. AFTER LUNCH THE SCIENTIST COULD HAVE THE COMPUTER TABULATE AND PRESENT THE DATA OBTAINED FROM THE EXPERIMENT IN COMPACT FORM. THEN - BY MERELY PUTTING A DIFFERENT PROGRAM IN THE MEMORY, THE SCIENTIST COULD HAVE THE COMPUTER HELP HIM SET UP AND ARRANGE A "REFERENCE FILE" ALL SORTED INTO ALPHABETICAL ORDER OR ANY MANNER THAT WOULD ENABLE HIM TO USE THE COMPUTER TO EXTRACT INFORMATION FAR FASTER THAN A MANUALLY OPERATED "PAPER FILE CARD" SYSTEM.

SO THE SCELBI-8H MINI-COMPUTER CAN BE A VALUABLE TOOL FOR A SCIENTIST - BUT, THE EXACT SAME MACHINE WITH A DIFFERENT PROGRAM IN ITS MEMORY (AND POSSIBLY DIFFERENT PERIPHERAL DEVICES) COULD BE USED TO CONTROL A COMPLEX MANUFACTURING OPERATION SUCH AS A PLASTIC INJECTION MOLDING MACHINE. IN SUCH A CASE I/O UNITS THAT COUPLED TO TRANSDUCERS ON THE INJECTION MOLDING MACHINE MIGHT BE USED TO RELAY INFORMATION TO THE COMPUTER ON A VARIETY OF PARAMETERS SUCH AS TEMPERATURE OF THE PLASTIC IN THE FEED BARREL, AMOUNT OF FEED MATERIAL IN THE HOPPER AND INJECTION BARREL, AVAILABLE PRESSURE TO THE MOLD JAWS AND FEED BARREL, VACANCY OR FILLED STATUS OF THE MOLD AND OTHER USEFUL PARAMETERS. THE COMPUTER COULD BE PROGRAMMED TO ANALYZE THIS INFORMATION AND SEND BACK SIGNALS TO CONTROL THE OPERATION OF HEATERS, PRESSURE VALVES, THE FEED RATE OF RAW MATERIALS, WHEN TO INJECT PLASTIC INTO THE MOLD, WHEN TO EMPTY THE MOLD, AND OTHER OPERATIONS TO ENABLE THE PLASTIC INJECTION SYSTEM TO OPERATE IN AN ESSENTIALLY AUTOMATIC MODE.

OR, A BUSINESSMAN COULD USE THE SAME COMPUTER CONNECTED TO AN ELECTRIC TYPEWRITER, WITH A SUITABLE PROGRAM IN MEMORY, TO COMPOSE, EDIT AND THEN TYPE OUT "PERSONALIZED FORM LETTERS" BY DIRECTING THE COMPUTER TO INSERT PARAGRAPHS FROM A "BANK OF STANDARD PARAGRAPHS" SO AS TO FORM A PERSONALIZED CUSTOMER ANSWERING SYSTEM THAT WOULD HANDLE ROUTINE INQUIRIES IN A FRACTION OF THE TIME (AND COST) THAT IT WOULD TAKE A SECRETARY TO PREPARE SUCH LETTERS. OR, THE BUSINESSMAN MIGHT UTILIZE THE COMPUTER TO HELP HIM CONTROL HIS INVENTORY, OR SPEED UP HIS ACCOUNTING OPERATIONS.

HOWEVER, A MINI-COMPUTER THAT COSTS AS LITTLE AS A SCELBI-8H DOES NOT HAVE TO BE RESTRICTED TO A BUSINESS OR SCIENTIFIC ENVIRONMENT. THE SCELBI-8H THAT CAN DO ALL THE TYPES OF TASKS MENTIONED ABOVE CAN ALSO BE USED TO HAVE FUN WITH - OR TO PERFORM VALUABLE SERVICES - TO PRIVATE INDIVIDUALS.

THE COMPUTER CAN BE USED AS A SOPHISTICATED ELECTRONIC CALCULATOR BY ALMOST ANYONE. IT CAN BE USED TO COMPOSE LETTERS (USING AN EDITOR PROGRAM) BY VIRTUALLY ANYONE. PROGRAMS THAT SORT DATA ALPHABETICALLY OR IN VARIOUS OTHER CATEGORIES CAN BE OF VALUABLE SERVICE TO PEOPLE IN MANY APPLICATIONS. THE COMPUTER CAN BE USED TO MONITOR AND CONTROL MANY HOUSEHOLD ITEMS, SERVE AS A SECURITY MONITORING SYSTEM, BE CONNECTED TO DEVICES THAT WILL DIAL TELEPHONES, AND DO THOUSANDS OF OTHER TASKS.

THE ELECTRONIC HOBBYIST CAN BE KEPT OCCUPIED FOR YEARS WITH A DIGITAL COMPUTER. FOR INSTANCE, ONE CAN BUILD A LITTLE TEST INSTRUMENT THAT PLUGS INTO A FEW I/O PORTS ON THE SCELBI-8H, THEN LOAD PROGRAMS INTO MEMORY THAT WILL DIRECT THE COMPUTER TO AUTOMATICALLY TEST ELECTRONIC COMPONENTS (SUCH AS COMPLEX TTL INTEGRATED CIRCUITS) IN A FRACTION OF A SECOND! (BUSINESSES CAN DO THIS TOO!)

OR A HAM RADIO OPERATOR CAN PUT A PROGRAM INTO MEMORY THAT WILL ENABLE THE SCELBI-8H MINI-COMPUTER TO RECEIVE MESSAGES TYPED IN FROM A KEYBOARD, CONVERT THE MESSAGES TO MORSE CODE, AND THEN ACTUATE AN OSCILLATOR V.I.A. AN OUTPUT PORT TO SEND PERFECTLY TIMED MORSE CODE.

IN ADDITION, FOR INSTANCE, THE HAM RADIO OPERATOR MIGHT USE THE COMPUTER WITH AN APPROPRIATE PROGRAM TO SERVE AS A "CONTEST LOGGING AID." THE "LOGGING AID" WOULD SERVE AS AN INSTANT REFERENCE FILE WHEREBY THE OPERATOR COULD ENTER THE CALLS OF STATIONS AS THEY WERE WORKED AND HAVE THE COMPUTER VERIFY IF THE CONTACT WAS A DUPLICATE - THE COMPUTER COULD DO OTHER TASKS TOO, SUCH AS RECORD THE TIME OF THE CONTACT BY CHECKING AN EXTERNAL DIGITAL CLOCK (OR BY UTILIZING A PROGRAM THAT WOULD ENABLE THE COMPUTER TO BE USED AS A CLOCK WITHIN ITSELF!)

AND, THE COMPUTER CAN BE USED TO PLAY NUMEROUS GAMES WITH, SUCH AS TIC-TAC-TOE, CHECKERS, WORD GAMES, CARD GAMES - AND A LARGE VARIETY OF OTHER TYPES OF GAMES THAT ONE CAN PROGRAM A COMPUTER TO PERFORM.

AND PERHAPS MOST IMPORTANT - FOR THE STUDENT, HOBBYIST, SCIENTIST, BUSINESSMAN, OR ANYONE INTERESTED IN THE EXCITING POSSIBILITIES OF ITS APPLICATIONS, THE SCELBI-8H MINI-COMPUTER OFFERS UNLIMITED POSSIBILITIES FOR THE EXPRESSION OF INDIVIDUAL CREATIVITY. FOR, THE DEVELOPMENT OF COMPUTER PROGRAMS CAN BE AN EXTREMELY CREATIVE, EXCITING, AND PERSONALLY REWARDING PASTIME AND OFFERS ESSENTIALLY LIMITLESS WAYS TO EXERCISE ONE'S CREATIVE CAPABILITIES IN DEVELOPING "ALGORITHMS" THAT WILL ENABLE THE MACHINE TO PERFORM DESIRED TASKS!

THE REMAINDER OF THIS BOOK IS AIMED AT SHOWING THE READERS HOW TO UTILIZE THE SCELBI-8H MINI-COMPUTER TO SERVE THEIR PERSONAL NEEDS. THIS BOOK WILL PROVIDE INFORMATION ON HOW TO OPERATE THE SCELBI-8H MINI-COMPUTER, HOW TO OPERATE VARIOUS TYPES OF PROGRAMS THAT ARE AVAILABLE FOR THE SCELBI-8H AS STANDARD PROGRAMS, HOW TO DEVELOPE NEW PROGRAMS IN "MACHINE LANGUAGE," AND HOW TO IMPLEMENT PERIPHERAL INTERFACES SO THAT THE COMPUTER CAN COMMUNICATE WITH EXTERNAL DEVICES THAT THE READER MAY WISH TO HAVE CONNECTED TO THE COMPUTER.

AT THIS POINT, HAVING READ THIS CHAPTER, THE READER ALREADY KNOWS THE BASIC STRUCTURE OF THE MACHINE. THE NEXT CHAPTER WILL PRESENT THE "INSTRUCTION SET," THAT IS THE PRECISE TYPES OF INSTRUCTIONS THAT THE MACHINE IS ABLE TO PERFORM, IN DETAIL. PRIOR TO BEGINNING THE NEXT CHAPTER IT WOULD BE BENEFICIAL FOR THE READER TO BECOME FAMILIAR WITH A COMMONLY USED "SHORTHAND" FOR REPRESENTING STRINGS OF BINARY NUMBERS. THIS SHORTHAND IS SIMPLY TO GROUP THREE BINARY "BITS" (CELLS) OF INFORMATION AND TAKE THE VALUE OF THE THREE BITS AS AN OCTAL NUMBER! A FEW DIAGRAMS WILL CLARIFY THE PROCESS.

FIGURE 11 REPRESENTS AN EIGHT CELL REGISTER SIMILAR TO A WORD OF MEMORY IN A SCELBI-8H MINI-COMPUTER. THE EIGHT CELLS ARE FILLED WITH ONES AND ZEROS WHICH REPRESENT THE TWO POSSIBLE STATES OF THE ELECTRONIC SWITCHES THAT ARE USED BY THE COMPUTER. THE CONTENTS OF THIS REGISTER ACTUALLY REPRESENTS A PATTERN THAT WOULD BE INTERPRETED BY THE COMPUTER AS SPECIFYING A SPECIFIC INSTRUCTION. IN THE NEXT CHAPTER THE READER WILL SEE THAT THE MACHINE CAN RECOGNIZE ABOUT 170 (OF THE 256 POSSIBLE PATTERNS THAT CAN BE PUT IN AN EIGHT CELL REGISTER) AND THEN PERFORM A SPECIFIC FUNCTION AS A RESULT OF RECOGNIZING A SPECIFIC PATTERN. EACH PATTERN THUS REPRESENTS A COMPUTER "INSTRUCTION." IN ORDER FOR HUMANS TO OPERATE AND PROGRAM THE MACHINE IT IS OFTEN NECESSARY FOR THEM TO ALSO BE ABLE TO RECOGNIZE AND "DECODE" THE PATTERNS OF ONES AND ZEROS. PEOPLE DO THIS BY REMEMBERING THE PATTERNS IN THEIR MINDS. HOWEVER, PEOPLE HAVE FOUND THAT IT CAN BE RATHER DIFFICULT TO REMEMBER STRINGS OF BINARY DIGITS. ESPECIALLY WHEN THEY MUST LEARN A LOT OF DIFFERENT PATTERNS. PEOPLE HAVE LEARNED THAT THE HUMAN MIND CAN HANDLE THE TASK MUCH EASIER IF THE BINARY DIGITS ARE GROUPED IN SETS OF THREE CELLS AND THE PATTERN REPRESENTED BY THE GROUPS OF THREE BITS CONVERTED TO AN OCTAL DIGIT!

NOTE THAT IN FIGURE 11 THAT THE LEFT MOST TWO BITS IN THE EIGHT CELL REGISTER CAN STILL BE ASSIGNED AN OCTAL EQUIVELANT BY ASSUMING THAT A THIRD CELL EXISTS WITH A BINARY VALUE OF ZERO (ILLUSTRATED BY THE DOTTED CELL) SO THAT THE OCTAL VALUE OF THE LEFT MOST GROUP CAN NOT EXCEED THREE. IT SHOULD BE APPARENT THAT THE OCTAL NUMBERING SYSTEM USES THE DIGITS Ø THROUGH 7. NOTE HOW IT IS MUCH EASIER TO REMEMBER THE OCTAL NUMBERS: 1 Ø 4 THAN IT IS TO REMEMBER THE BINARY STRING OF NUMBERS: Ø 1 Ø Ø Ø 1 Ø Ø .

```
                    EIGHT CELL REGISTER
                              .
                             . .
      .......................   .......................
      .                                               .
......*************************************************
.     *     *     ↑     *     *     ↑     *     *     *
.  Ø  *  Ø  *  1  ↑  Ø  *  Ø  *  Ø  ↑  1  *  Ø  *  Ø  *
.     *     *     ↑     *     *     ↑     *     *     *
......*************************************************
.                 .                 .                 .
........   ...............   ...............   ........
        . .               . .               . .
         .                 .                 .
         1                 Ø                 4
```

FIGURE 11

CONVERTING AN EIGHT BIT REGISTER FROM BINARY TO OCTAL NUMBERS SO THAT THE PATTERN CAN BE MORE EASILY REMEMBERED BY PROGRAMMERS.

FIGURE 12 ILLUSTRATES THE RELATIONSHIP BETWEEN THE BINARY AND OCTAL SYSTEMS FOR ALL THE POSSIBLE PATTERNS WITH THREE CELLS AS AN AID TO LEARNING THE CONVERSION TECHNIQUE.

BINARY PATTERN	REPRESENTATIVE OCTAL #
Ø Ø Ø	Ø
Ø Ø 1	1
Ø 1 Ø	2
Ø 1 1	3
1 Ø Ø	4
1 Ø 1	5
1 1 Ø	6
1 1 1	7

FIGURE 12

THE REMAINDER OF THIS BOOK ASSUMES THAT THE READER UNDERSTANDS THE BINARY TO OCTAL CONVERSION PROCESS (AND VICE-VERSA) AND IT IS RECOMMENDED THAT THE READER TAKE WHATEVER TIME IS NECESSARY TO THOROUGHLY UNDERSTAND THE RELATIONSHIP BEFORE PROCEEDING TO READ FURTHER IN THIS BOOK.

SCELBI-8H PROGRAMMING INSTRUCTION SET

THIS MINI-COMPUTER HAS QUITE A COMPREHENSIVE INSTRUCTION SET THAT CONSISTS OF 48 BASIC INSTRUCTIONS, WHICH, WHEN THE POSSIBLE PERMUTATIONS ARE CONSIDERED, RESULT IN A TOTAL SET OF ABOUT 170 INSTRUCTIONS.

THE INSTRUCTION SET ALLOWS THE USER TO DIRECT THE COMPUTER TO PERFORM OPERATIONS WITH MEMORY, WITH THE 7 BASIC REGISTERS IN THE CPU, AND WITH INPUT AND OUTPUT PORTS.

IT SHOULD BE POINTED OUT THAT THE 7 BASIC REGISTERS IN THE CPU CONSIST OF ONE ACCUMULATOR - THAT IS A REGISTER THAT CAN PERFORM ADDITIONS, SUBTRACTIONS, ROTATES ETC., AND 6 ADDITIONAL REGISTERS WHICH WHILE NOT HAVING THE FULL CAPABILITY OF THE ACCUMULATOR, CAN PERFORM CERTAIN OPERATIONS (INCREMENT AND DECREMENT), CAN STORE DATA, AND CAN OPERATE WITH THE ACCUMULATOR. TWO OF THE SIX REGISTERS HAVE SPECIAL SIGNIFICANCE BECAUSE THEY MAY BE USED TO "POINT" TO AN ADDRESS IN MEMORY.

THE SEVEN CPU REGISTERS HAVE ARBITRARILY BEEN GIVEN SYMBOLS SO THAT WE MAY REFER TO THEM IN A COMMON LANGUAGE. THE FIRST REGISTER IS DESIGNATED BY THE SYMBOL "A" IN THE FOLLOWING DISCUSSION AND WILL BE CONSIDERED THE ACCUMULATOR REGISTER. THE NEXT FOUR REGISTERS WILL BE REFERRED TO AS THE "B," "C," "D," AND "E," REGISTERS, AND THE REMAINING TWO SPECIAL MEMORY POINTING REGISTERS SHALL BE DESIGNATED THE "H" (FOR THE HIGH PORTION OF A MEMORY ADDRESS) AND THE "L" (FOR THE LOW PORTION OF A MEMORY ADDRESS) REGISTERS.

THE CPU ALSO HAS SEVERAL FLIP-FLOPS WHICH SHALL BE REFERRED TO AS "FLAGS." THESE FLIP-FLOPS ARE SET AS THE RESULT OF CERTAIN OPERATIONS AND ARE IMPORTANT BECAUSE THEY CAN BE "TESTED" BY MANY OF THE INSTRUCTIONS AND THE INSTRUCTION'S MEANING CHANGED AS A CONSEQUENCE OF THE FLAGS PARTICULAR STATUS AT THE TIME IT IS TESTED. THERE ARE FOUR BASIC FLAGS WHICH WILL BE REFERRED TO IN THIS MANUAL DESIGNATED AS FOLLOWS:

> THE "C" FLAG REFERS TO THE CARRY BIT STATUS. THE CARRY BIT IS A 1 UNIT REGISTER WHICH CHANGES STATE WHEN THE ACCUMULATOR OVER-FLOWS OR UNDER-FLOWS. THIS BIT CAN ALSO BE SET TO A KNOWN CONDITION BY CERTAIN TYPES OF INSTRUCTIONS. THIS IS IMPORTANT TO REMEMBER WHEN DEVELOPING A PROGRAM BECAUSE QUITE OFTEN A PROGRAM WILL HAVE A LONG STRING OF INSTRUCTIONS WHICH DO NOT UTILIZE THE CARRY BIT OR CARE ABOUT ITS STATUS, BUT WHICH WILL BE CAUSING THE CARRY BIT TO CHANGE ITS STATUS FROM TIME TO TIME. THUS, WHEN ONE PREPARES TO DO A SERIES OF OPERATIONS THAT WILL RELY ON THE CARRY BIT, ONE OFTEN DESIRES TO SET THE CARRY BIT TO A KNOWN STATE.
>
> THE "Z" FOR ZERO FLAG REFERS TO A 1 UNIT REGISTER THAT WHEN DESIRED WILL INDICATE WHETHER THE VALUE OF THE ACCUMULATOR IS EXACTLY EQUAL TO ZERO. IN ADDITION, IMMEDIATELY AFTER AN INCREMENT OR DECREMENT OF THE B, C, D, E, H OR L REGISTERS, THIS FLAG WILL ALSO INDICATE WHETHER THE INCREMENT OR DECREMENT CAUSED THAT PARTICULAR REGISTER TO GO TO ZERO.
>
> THE "S" FOR SIGN FLAG REFERS TO A 1 UNIT REGISTER THAT INDICATES WHETHER THE VALUE IN THE ACCUMULATOR IS A POSITIVE OR NEGATIVE VALUE (BASED ON TWO'S COMPLEMENT NOMENCLATURE.) ESSENTIALLY, THIS FLAG MONITORS THE MOST SIGNIFICANT BIT IN THE ACCUMULATOR AND IS "SET" WHEN IT IS A ONE.

THE "P" FLAG REFERS TO THE LAST FLAG IN THE GROUP WHICH IS FOR INDICATING WHEN THE ACCUMULATOR CONTAINS A VALUE WHICH HAS EVEN PARITY. PARITY IS USEFUL FOR A NUMBER OF REASONS AND IS USUALLY USED IN CONJUNCTION WITH TESTING FOR ERROR CONDITIONS ON WORDS OF DATA PARTICULARLY WHEN INPUTTING DATA FROM EXTERNAL SOURCES. EVEN PARITY OCCURS WHEN THE NUMBER OF BITS THAT ARE A "1" IN THE ACCUMULATOR (OUT OF THE EIGHT POSSIBLE) IS AN EVEN VALUE, I.E., 2, 4, 6, OR 8; REGARDLESS OF WHAT ORDER THEY MAY BE IN THE ACCUMULATOR REGISTER.

IT IS IMPORTANT TO NOTE THAT THE "Z," "S," AND "P" FLAGS (AS WELL AS THE PREVIOUSLY MENTIONED "C" FLAG) CAN ALL BE SET TO KNOWN STATES BY CERTAIN INSTRUCTIONS. IT IS ALSO IMPORTANT TO NOTE THAT SOME INSTRUCTIONS DO NOT RESULT IN THE FLAGS BEING SET SO THAT IF THE PROGRAMMER DESIRES TO HAVE THE PROGRAM MAKE "DECISIONS" BASED ON THE STATUS OF FLAGS, THE PROGRAMMER SHOULD ENSURE THAT THE PROPER INSTRUCTION, OR SEQUENCE OF INSTRUCTIONS IS UTILIZED. IT IS PARTICULARLY IMPORTANT TO NOTE THAT "LOAD REGISTER" INSTRUCTIONS DO NOT BY THEMSELVES SET THE FLAGS. SINCE IT IS OFTEN DESIRABLE TO OBTAIN A DATA WORD (I.E. LOAD IT INTO THE ACCUMULATOR) AND TEST ITS STATUS FOR SUCH PARAMETERS AS WHETHER OR NOT THE VALUE IS ZERO, OR A NEGATIVE NUMBER ETC., THE PROGRAMMER MUST REMEMBER TO FOLLOW A LOAD INSTRUCTION BY A LOGICAL INSTRUCTION (SUCH AS THE NDA - "AND THE ACCUMULATOR") IN ORDER TO SET THE FLAGS BEFORE USING AN INSTRUCTION THAT IS CONDITIONAL IN REGARDS TO THE FLAG STATUS.

THE DESCRIPTION OF THE VARIOUS TYPES OF INSTRUCTIONS AVAILABLE ON THE SCELBI-8H WHICH FOLLOWS WILL PROVIDE BOTH THE MACHINE LANGUAGE CODE FOR THE INSTRUCTION GIVEN AS 3 OCTAL DIGITS, AND ALSO A MNEMONIC NAME SUITABLE FOR WRITING PROGRAMS IN SYMBOLIC TYPE LANGUAGE WHICH IS USUALLY EASIER THAN TRYING TO REMEMBER OCTAL CODES! IT MAY BE NOTED THAT THE SYMBOLIC LANGUAGE USED IS THE SAME AS THAT UTILIZED BY INTEL CORPORATION WHICH DEVELOPED THE 8008 (RTM) "CPU-ON-A-CHIP" WHICH IS AT THE HEART OF THE SCELBI-8H, AND HENCE THE USE OF THIS MNEMONIC LANGUAGE WILL ALLOW PROGRAMS DEVELOPED FOR THE SCELBI-8H TO BE COMPATIBLE WITH OTHER TYPES OF COMPUTING SYSTEMS WHICH UTILIZE THE INTEL 8008 (RTM) DEVICE. IF THE PROGRAMMER IS NOT ALREADY AWARE OF IT, THE USE OF MNEMONICS FACILITATES WORKING WITH AN "ASSEMBLER" PROGRAM WHEN IT IS DESIRED TO DEVELOPE RELATIVELY LARGE PROGRAMS. THUS THE PROGRAMMER IS URGED TO CONCENTRATE ON LEARNING THE MNEMONICS FOR THE INSTRUCTIONS AND NOT WASTE TIME MEMORIZING THE OCTAL CODES. AFTER A PROGRAM HAS BEEN WRITTEN USING THE MNEMONIC CODES, THE PROGRAMMER CAN ALWAYS USE A LOOKUP TABLE TO CONVERT TO THE MACHINE CODE IF AN ASSEMBLER PROGRAM IS NOT AVAILABLE. ITS A LOT EASIER TECHNIQUE (AND LESS SUBJECT TO ERROR) THAN TRYING TO MEMORIZE THE 170 OR SO 3 DIGIT COMBINATIONS WHICH MAKE UP THE MACHINE INSTRUCTION CODE SET!

THE PROGRAMMER MUST ALSO BE AWARE, THAT IN THIS MACHINE, SOME INSTRUCTIONS REQUIRE MORE THAN ONE "WORD" IN MEMORY. "IMMEDIATE" TYPE COMMANDS REQUIRE TWO CONSECUTIVE WORDS AND JUMP AND CALL COMMANDS REQUIRE THREE CONSECUTIVE WORDS. THE REMAINING TYPES OF INSTRUCTIONS ONLY REQUIRE ONE WORD. THIS WILL BE PRESENTED IN DETAIL IN THE DESCRIPTION FOR EACH TYPE OF INSTRUCTION.

THE FIRST GROUP OF INSTRUCTIONS TO BE PRESENTED ARE THOSE THAT ARE USED TO "LOAD" DATA FROM ONE CPU REGISTER TO ANOTHER, OR FROM A CPU REGISTER TO A WORD IN MEMORY, OR VICE-VERSA. THIS GROUP OF INSTRUCTIONS REQUIRES JUST ONE WORD OF MEMORY. IT IS IMPORTANT TO NOTE THAT NONE OF THE INSTRUCTIONS IN THIS GROUP AFFECT THE "FLAGS."

LOAD DATA FROM ONE CPU REGISTER TO ANOTHER CPU REGISTER

MNEMONIC	MACHINE CODE
LAA	3 Ø Ø
LBA	3 1 Ø
•	•
•	•
LAB	3 Ø 1

THE LOAD REGISTER GROUP OF INSTRUCTIONS ALLOWS THE PROGRAMMER TO MOVE THE CONTENTS OF ONE CPU REGISTER INTO ANOTHER CPU REGISTER. THE CONTENTS OF THE ORIGINATING (FROM) REGISTER IS NOT CHANGED. THE CONTENTS OF THE DESTINATION (TO) REGISTER BECOMES THE SAME AS THE ORIGINATING REGISTER. ANY CPU REGISTER CAN BE LOADED INTO ANY CPU REGISTER. NOTE THAT FOR INSTANCE LOADING REGISTER "A" INTO REGISTER "A" IS ESSENTIALLY A "NOP" (NO OPERATION) COMMAND. WHEN USING MNEMONICS THE LOAD SYMBOL IS THE LETTER "L" FOLLOWED BY THE "TO" REGISTER AND THEN THE "FROM" REGISTER. THE MNEMONIC "LBA" MEANS THE THE CONTENTS OF REGISTER "A" (THE ACCUMULATOR) IS TO BE LOADED INTO REGISTER "B." THE MNEMONIC "LAB" STATES THAT REGISTER "B" IS TO HAVE ITS CONTENTS LOADED INTO REGISTER "A." IT CAN BE SEEN THAT THIS BASIC INSTRUCTION HAS MANY VARIATIONS. THE MACHINE LANGUAGE CODING FOR THIS INSTRUCTION IS IN THE SAME FORMAT AS THE MNEMONIC CODE EXCEPT THAT THE LETTERS USED TO REPRESENT THE REGISTERS ARE REPLACED BY NUMBERS THAT THE MACHINE CAN USE. USING OCTAL CODE, THE 7 CPU REGISTERS ARE CODED AS FOLLOWS:

REG "A" =	Ø
REG "B" =	1
REG "C" =	2
REG "D" =	3
REG "E" =	4
REG "H" =	5
REG "L" =	6

ALSO SINCE THE MACHINE CAN ONLY UTILIZE NUMBERS, THE OCTAL NUMBER 3 IN THE MOST SIGNIFICANT LOCATION OF A WORD SIGNIFIES THAT THE COMPUTER IS TO PERFORM A "LOAD" OPERATION. THUS, IN MACHINE CODING, THE INSTRUCTION FOR LOADING REGISTER "B" WITH THE CONTENTS OF REGISTER "A" BECOMES: 3 1 Ø (IN OCTAL FORM) OR, IF ONE WANTED TO GET VERY DETAILED, THE ACTUAL BINARY CODING FOR THE 8 BITS OF INFORMATION IN THE INSTRUCTION WORD WOULD BE: 1 1 Ø Ø 1 Ø Ø Ø. IT IS IMPORTANT TO NOTE THAT THE LOAD INSTRUCTIONS DO NOT AFFECT ANY OF THE "FLAGS."

LOAD DATA FROM ANY CPU REGISTER TO A LOCATION IN MEMORY

LMA	3 7 Ø
LMB	3 7 1
LMC	3 7 2
LMD	3 7 3
LME	3 7 4
LMH	3 7 5
LML	3 7 6

THIS INSTRUCTION IS VERY SIMILAR TO THE PREVIOUS GROUP OF INSTRUCTIONS EXCEPT THAT NOW THE CONTENTS OF A CPU REGISTER WILL BE LOADED INTO A SPECIFIED MEMORY LOCATION. THE MEMORY LOCATION THAT WILL RECEIVE THE CONTENTS OF THE PARTICULAR CPU REGISTER IS THAT WHOSE ADDRESS IS SPECIFIED BY THE CONTENTS OF THE CPU "H" AND "L" REGISTERS AT THE TIME THE INSTRUCTION IS EXECUTED. THE "H" CPU REGISTER SPECIFIES THE "HIGH" PORTION OF THE ADDRESS DESIRED, AND THE "L" CPU REGISTER SPECIFIES THE "LOW" PORTION OF THE ADDRESS

INTO WHICH DATA FROM THE SELECTED CPU REGISTER IS TO BE LOADED. NOTE THAT THERE ARE 7 DIFFERENT INSTRUCTIONS IN THIS GROUP AS ANY CPU REGISTER CAN HAVE ITS CONTENTS LOADED INTO ANY LOCATION IN MEMORY. THIS GROUP OF INSTRUCTIONS DOES NOT AFFECT ANY OF THE "FLAGS."

LOAD DATA FROM A MEMORY LOCATION TO ANY CPU REGISTER

LAM	3 0 7
LBM	3 1 7
LCM	3 2 7
LDM	3 3 7
LEM	3 4 7
LHM	3 5 7
LLM	3 6 7

THIS GROUP OF INSTRUCTIONS CAN BE CONSIDERED THE OPPOSITE OF THE PREVIOUS GROUP. NOW, THE CONTENTS OF THE WORD IN MEMORY WHOSE ADDRESS IS SPECIFIED BY THE "H" (FOR THE HIGH PORTION OF THE ADDRESS) AND "L" (LOW PORTION OF THE ADDRESS) REGISTERS WILL BE LOADED INTO THE CPU REGISTER SPECIFIED BY THE INSTRUCTION. ONCE AGAIN, THIS GROUP OF INSTRUCTIONS HAS NO AFFECT ON THE STATUS OF THE "FLAGS."

LOAD "IMMEDIATE" DATA INTO A CPU REGISTER

LAI	0 0 6
LBI	0 1 6
LCI	0 2 6
LDI	0 3 6
LEI	0 4 6
LHI	0 5 6
LLI	0 6 6

AN "IMMEDIATE" TYPE OF INSTRUCTION REQUIRES TWO WORDS IN ORDER TO BE COMPLETELY SPECIFIED. THE FIRST WORD IS THE INSTRUCTION ITSELF, THE SECOND WORD, OR "IMMEDIATELY FOLLOWING" WORD, MUST CONTAIN THE DATA UPON WHICH IMMEDIATE ACTION IS TAKEN. THUS, A LOAD "IMMEDIATE" INSTRUCTION IN THIS GROUP MEANS THAT THE CONTENTS OF THE WORD IMMEDIATELY FOLLOWING THE INSTRUCTION WORD IS TO BE LOADED INTO THE SPECIFIED REGISTER. FOR EXAMPLE, A TYPICAL LOAD IMMEDIATE INSTRUCTION WOULD BE: LAI 001. THIS WOULD RESULT IN THE VALUE 001 BEING PLACED IN THE "A" REGISTER WHEN THE INSTRUCTION WAS EXECUTED. IT IS IMPORTANT TO REMEMBER THAT ALL "IMMEDIATE" TYPE INSTRUCTIONS MUST BE FOLLOWED BY A DATA WORD. AN INSTRUCTION SUCH AS LDI ALONE WOULD RESULT IN IMPROPER OPERATION BECAUSE THE COMPUTER WOULD ASSUME THE NEXT WORD CONTAINED DATA, AND IF THE PROGRAMMER HAS MISTAKENLY LEFT OUT THE DATA WORD, AND IN ITS PLACE HAD ANOTHER INSTRUCTION, THE COMPUTER WOULD NOT REALIZE THE OPERATORS "MISTAKE" AND HENCE THE PROGRAM WOULD BE "FOULED-UP!" NOTE TOO, THAT THE LOAD "IMMEDIATE" GROUP OF INSTRUCTIONS DOES NOT AFFECT THE "FLAGS."

LOAD "IMMEDIATE" DATA INTO A MEMORY LOCATION

LMI	0 7 6

THIS INSTRUCTION IS ESSENTIALLY THE SAME AS THE LOAD IMMEDIATE INTO THE CPU REGISTER GROUP EXCEPT THAT NOW, USING THE CONTENTS OF

THE "H" AND "L" REGISTERS AS "POINTERS" TO THE DESIRED ADDRESS IN MEMORY, THE CONTENTS OF THE "IMMEDIATELY FOLLOWING WORD" WILL BE PLACED IN THE MEMORY LOCATION SPECIFIED. THIS INSTRUCTION DOES NOT AFFECT THE STATUS OF THE "FLAGS."

THE ABOVE RATHER LARGE GROUP OF "LOAD" INSTRUCTIONS PERMIT THE PROGRAMMER TO DIRECT THE COMPUTER TO MOVE DATA ABOUT. THEY ARE USED TO BRING IN DATA FROM MEMORY WHERE IT CAN BE OPERATED ON BY THE CPU, OR TO TEMPORARILY STORE INTERMEDIATE RESULTS IN THE CPU REGISTER DURING COMPLICATED AND EXTENDED CALCULATIONS, AND OF COURSE ALLOW DATA, SUCH AS RESULTS ETC., TO BE PLACED BACK INTO MEMORY FOR LONG TERM STORAGE. SINCE NONE OF THEM WILL ALTER THE CONTENTS OF THE FOUR CPU FLAGS, THESE INSTRUCTIONS CAN BE CALLED UPON TO, FOR EXAMPLE, SET UP DATA, BEFORE INSTRUCTIONS THAT MAY AFFECT OR UTILIZE THE FLAGS' STATUS ARE EXECUTED. THE PROGRAMMER WILL USE INSTRUCTIONS FROM THIS SET FREQUENTLY. THE MNEMONIC NAMES FOR THE INSTRUCTIONS ARE EASY TO REMEMBER AS THEY ARE WELL ORDERED. THE MOST IMPORTANT ITEM TO REMEMBER ABOUT THE MNEMONICS IS THAT THE "TO" REGISTER IS ALWAYS INDICATED FIRST IN THE MNEMONIC, AND THEN THE "FROM" REGISTER. THUS "LBA" = "LOAD TO REGISTER "B" FROM REGISTER "A."

INCREMENT THE VALUE OF A CPU REGISTER BY 1

INB	0 1 0
INC	0 2 0
IND	0 3 0
INE	0 4 0
INH	0 5 0
INL	0 6 0

THIS GROUP OF INSTRUCTIONS ALLOWS THE PROGRAMMER TO "ADD 1" TO THE PRESENT VALUE OF ANY OF THE CPU REGISTERS EXCEPT THE ACCUMULATOR. (NOTE CAREFULLY THAT THE ACCUMULATOR CAN NOT BE INCREMENTED BY THIS TYPE OF INSTRUCTION. IN ORDER TO "ADD 1" TO THE ACCUMULATOR A MATH-EMATICAL ADDITION INSTRUCTION, DESCRIBED LATER, MUST BE USED.) THIS INSTRUCTION FOR INCREMENTING THE DEFINED CPU REGISTERS IS VERY VAL-UABLE IN A NUMBER OF APPLICATIONS. FOR ONE THING, IT IS AN EASY WAY TO HAVE THE "L" REGISTER SUCCESSIVELY "POINT" TO A STRING OF LOC-ATIONS IN MEMORY. A FEATURE THAT MAKES THIS TYPE OF INSTRUCTION EVEN MORE POWERFUL, IS THAT THE RESULT OF THE INCREMENTED REGISTER WILL AFFECT THE "Z," "S," AND "P" FLAGS. (IT WILL NOT CHANGE THE "C" OR "CARRY" FLAG.) THUS, AFTER A CPU REGISTER HAS BEEN INCREMENTED BY THIS INSTRUCTION, ONE CAN UTILIZE A "FLAG TEST" INSTRUCTION (SUCH AS THE JUMP AND CALL INSTRUCTIONS TO BE DESCRIBED LATER) TO DETERMINE WHETHER THAT PARTICULAR REGISTER HAS A VALUE OF ZERO ("Z" FLAG), OR IF IT IS A NEGATIVE NUMBER ("S" FLAG), OR EVEN PARITY ("P" FLAG.) IT IS IMPORTANT TO NOTE THAT THIS GROUP OF INSTRUCTIONS, AND THE DECREMENT GROUP (DESCRIBED IN THE NEXT PARAGRAPH) ARE THE ONLY INSTR-UCTIONS WHICH ALLOW THE "FLAGS" TO BE MANIPULATED BY OPERATIONS THAT ARE NOT CONCERNED WITH THE ACCUMULATOR ("A") REGISTER.

DECREMENT THE VALUE OF A CPU REGISTER BY 1

DCB	0 1 1
DCC	0 2 1
DCD	0 3 1
DCE	0 4 1
DCH	0 5 1
DCL	0 6 1

THE DECREMENT GROUP OF INSTRUCTIONS IS SIMILAR TO THE INCREMENT GROUP EXCEPT THAT NOW THE VALUE 1 WILL BE SUBTRACTED FROM THE SPECIFIED CPU REGISTER. THIS INSTRUCTION WILL NOT AFFECT THE "C" FLAG BUT IT DOES AFFECT THE "Z," "S," AND "P" FLAGS. IT SHOULD ALSO BE NOTED THAT THIS GROUP, AS WITH THE INCREMENT GROUP, DOES NOT INCLUDE THE ACCUMULATOR REGISTER. A SEPARATE MATHEMATICAL INSTRUCTION MUST BE USED TO SUBTRACT 1 FROM THE ACCUMULATOR.

ARITHMETIC INSTRUCTIONS USING THE ACCUMULATOR

THE FOLLOWING GROUP OF INSTRUCTIONS ALLOW THE PROGRAMMER TO DIRECT THE COMPUTER TO PERFORM ARITHMETIC OPERATIONS BETWEEN OTHER CPU REGISTERS AND THE ACCUMULATOR, OR BETWEEN THE CONTENTS OF WORDS IN MEMORY AND THE ACCUMULATOR. ALL OF THE OPERATIONS FOR THE DESCRIBED ADDITION, SUBTRACTION, AND COMPARE INSTRUCTIONS AFFECT THE STATUS OF THE "FLAGS."

ADD THE CONTENTS OF A CPU REGISTER TO THE ACCUMULATOR

ADA	2 Ø Ø
ADB	2 Ø 1
ADC	2 Ø 2
ADD	2 Ø 3
ADE	2 Ø 4
ADH	2 Ø 5
ADL	2 Ø 6

THIS GROUP OF INSTRUCTIONS WILL SIMPLY ADD THE PRESENT CONTENTS OF THE ACCUMULATOR REGISTER TO THE PRESENT VALUE OF THE SPECIFIED CPU REGISTER AND LEAVE THE RESULT IN THE ACCUMULATOR. THE VALUE OF THE SPECIFIED REGISTER IS UNCHANGED EXCEPT IN THE CASE OF THE "ADA" INSTRUCTION. NOTE THAT THE "ADA" INSTRUCTION ESSENTIALLY ALLOWS THE PROGRAMMER TO DOUBLE THE VALUE OF THE ACCUMULATOR (WHICH IS THE "A" REGISTER!) IF THE ADDITION CAUSES AN "OVER-FLOW" OR "UNDER-FLOW" THEN THE "CARRY" ("C" FLAG) WILL BE AFFECTED.

ADD THE CONTENTS OF A CPU REGISTER PLUS THE VALUE OF THE CARRY FLAG TO THE ACCUMULATOR

ACA	2 1 Ø
ACB	2 1 1
ACC	2 1 2
ACD	2 1 3
ACE	2 1 4
ACH	2 1 5
ACL	2 1 6

THIS GROUP IS IDENTICAL TO THE PREVIOUS GROUP EXCEPT THAT NOW THE CONTENT OF THE CARRY FLAG IS CONSIDERED AS AN ADDITIONAL BIT (MSB) IN THE SPECIFIED CPU REGISTER AND THE COMBINED VALUE OF THE CARRY BIT PLUS THE CONTENTS OF THE SPECIFIED CPU REGISTER ARE ADDED TO THE VALUE IN THE ACCUMULATOR. THE RESULTS ARE LEFT IN THE ACCUMULATOR. AGAIN, WITH THE EXCEPTION OF THE "ACA" INSTRUCTION, THE CONTENTS OF THE SPECIFIED CPU REGISTER IS LEFT UNCHANGED. AGAIN TOO, THE CARRY BIT ("C" FLAG) WILL BE AFFECTED BY THE RESULTS OF THE OPERATION.

SUBTRACT THE CONTENTS OF A CPU REGISTER FROM THE ACCUMULATOR

SUA	2 2 0
SUB	2 2 1
SUC	2 2 2
SUD	2 2 3
SUE	2 2 4
SUH	2 2 5
SUL	2 2 6

THIS GROUP OF INSTRUCTIONS WILL CAUSE THE PRESENT VALUE OF THE SPECIFIED CPU REGISTER TO BE SUBTRACTED FROM THE VALUE IN THE ACCUMULATOR. THE VALUE OF THE SPECIFIED REGISTER IS NOT CHANGED EXCEPT IN THE CASE OF THE "SUA" INSTRUCTION. (NOTE THAT THE "SUA" INSTRUCTION IS A CONVENIENT INSTRUCTION WITH WHICH TO "CLEAR" THE ACCUMULATOR.) THE CARRY FLAG WILL BE AFFECTED BY THE RESULTS OF A SUBTRACT INSTRUCTION.

SUBTRACT THE CONTENTS OF A CPU REGISTER AND THE VALUE OF THE CARRY FLAG FROM THE ACCUMULATOR

SBA	2 3 0
SBB	2 3 1
SBC	2 3 2
SBD	2 3 3
SBE	2 3 4
SBH	2 3 5
SBL	2 3 6

THIS GROUP IS IDENTICAL TO THE PREVIOUS GROUP EXCEPT THAT NOW THE CONTENT OF THE CARRY FLAG IS CONSIDERED AS AN ADDITIONAL BIT (MSB) IN THE SPECIFIED CPU REGISTER AND THE COMBINED VALUE OF THE CARRY BIT PLUS THE CONTENTS OF THE SPECIFIED CPU REGISTER ARE SUBTRACTED FROM THE VALUE IN THE ACCUMULATOR. THE RESULTS ARE LEFT IN THE ACCUMULATOR, AND THE CARRY BIT ("C" FLAG) IS AFFECTED BY THE RESULT OF THE OPERATION. WITH THE EXCEPTION OF THE "SBA" INSTRUCTION THE CONTENTS OF THE SPECIFIED CPU REGISTER IS LEFT UNCHANGED.

COMPARE THE VALUE IN THE ACCUMULATOR AGAINST THE CONTENTS OF A CPU REGISTER

CPA	2 7 0
CPB	2 7 1
CPC	2 7 2
CPD	2 7 3
CPE	2 7 4
CPH	2 7 5
CPL	2 7 6

THE "COMPARE" GROUP OF INSTRUCTIONS ARE A VERY POWERFUL AND SOMEWHAT UNIQUE SET OF INSTRUCTIONS. THEY DIRECT THE COMPUTER TO COMPARE THE CONTENTS OF THE ACCUMULATOR AGAINST ANOTHER REGISTER AND TO SET THE "FLAGS" AS A RESULT OF THE COMPARING OPERATION. IT IS ESSENTIALLY A SUBTRACTION OPERATION WITH THE VALUE OF THE SPECIFIED REGISTER BEING SUBTRACTED FROM THE VALUE OF THE ACCUMULATOR EXCEPT THAT THE VALUE OF THE ACCUMULATOR IS NOT ACTUALLY ALTERED BY THE OPERATION. HOWEVER, THE "FLAGS" ARE SET IN THE SAME MANNER AS THOUGH AN ACTUAL SUBTRACTION OPERATION HAD OCCURED. THUS, BY SUBSEQUENTLY TESTING THE STATUS OF THE VARIOUS FLAGS AFTER A COM-

PARE INSTRUCTION HAS BEEN EXECUTED, THE PROGRAM CAN DETERMINE WHETHER THE "COMPARE" OPERATION RESULTED IN A MATCH, OR NON-MATCH, AND IN THE CASE OF A NON-MATCH WHETHER THE COMPARED REGISTER CONTAINED A VALUE GREATER OR LESS THAN THAT IN THE ACCUMULATOR. THIS WOULD BE ACCOMPLISHED BY TESTING THE "Z" FLAG AND "C" FLAG RESPECTIVELY UTILIZING A "JUMP" OR "CALL" FLAG TESTING INSTRUCTION (WHICH WILL BE DESCRIBED LATER.)

ADDITION, SUBTRACTION, AND COMPARE INSTRUCTIONS THAT USE WORDS IN MEMORY AS OPERANDS

THE FIVE TYPES OF MATHEMATICAL OPERATIONS: ADD, ADD WITH CARRY, SUBTRACT, SUBTRACT WITH CARRY, AND THE COMPARE; WHICH HAVE JUST BEEN PRESENTED FOR PERFORMING THE OPERATIONS WITH THE CONTENTS OF THE CPU REGISTERS, CAN ALL ALSO BE PERFORMED WITH WORDS THAT ARE IN MEMORY. AS WITH THE "LOAD" INSTRUCTIONS WITH MEMORY, THE "H" AND "L" REGISTERS MUST CONTAIN THE ADDRESS OF THE WORD IN MEMORY THAT IT IS DESIRED TO ADD, SUBTRACT, OR COMPARE TO THE ACCUMULATOR. THE SAME CONDITIONS FOR THE OPERATIONS AS WAS DETAILED WHEN USING THE CPU REGISTERS APPLY. THUS, FOR MATHEMATICAL OPERATIONS WITH A WORD IN MEMORY, THE FOLLOWING INSTRUCTIONS ARE USED:

ADD THE CONTENTS OF A MEMORY WORD TO THE ACCUMULATOR

```
ADM         2 0 7
```

ADD THE CONTENTS OF A MEMORY WORD PLUS THE VALUE OF THE CARRY FLAG TO THE ACCUMULATOR

```
ACM         2 1 7
```

SUBTRACT THE CONTENTS OF A MEMORY WORD FROM THE ACCUMULATOR

```
SUM         2 2 7
```

SUBTRACT THE CONTENTS OF A MEMORY WORD AND THE VALUE OF THE CARRY FLAG FROM THE ACCUMULATOR

```
SBM         2 3 7
```

COMPARE THE VALUE IN THE ACCUMULATOR AGAINST THE CONTENTS OF A MEMORY WORD

```
CPM         2 7 7
```

"IMMEDIATE" TYPE ADDITIONS, SUBTRACTIONS, AND COMPARE INSTRUCTIONS

THE 5 TYPES OF MATHEMATICAL OPERATIONS DISCUSSED CAN ALSO BE PERFORMED WITH THE OPERAND BEING THE WORD OF DATA IMMEDIATELY AFTER THE INSTRUCTION. THIS GROUP OF INSTRUCTIONS IS SIMILAR IN FORMAT TO THE PREVIOUSLY DESCRIBED "LOAD IMMEDIATE" INSTRUCTIONS. THE SAME CONDITIONS FOR THE MATHEMATIC OPERATIONS AS DISCUSSED FOR THE OPERATIONS WITH THE CPU REGISTERS APPLY.

ADD "IMMEDIATE"

ADI	Ø Ø 4

ADD WITH CARRY "IMMEDIATE"

ACI	Ø 1 4

SUBTRACT "IMMEDIATE"

SUI	Ø 2 4

SUBTRACT WITH CARRY "IMMEDIATE"

SBI	Ø 3 4

COMPARE "IMMEDIATE"

CPI	Ø 7 4

LOGICAL INSTRUCTIONS WITH THE ACCUMULATOR

THERE ARE SEVERAL GROUPS OF INSTRUCTIONS WHICH ALLOW BOOLEAN LOGIC OPERATIONS TO BE PERFORMED BETWEEN THE CONTENTS OF THE CPU REGISTERS AND THE "A" OR ACCUMULATOR REGISTER, AS WELL AS BETWEEN CONTENTS OF LOCATIONS IN MEMORY AND THE "A" REGISTER. IN ADDITION THERE ARE LOGIC "IMMEDIATE" TYPE INSTRUCTIONS. THE BOOLEAN LOGIC OPERATIONS ARE VALUABLE IN A NUMBER OF PROGRAMMING APPLICATIONS. THE INSTRUCTION SET ALLOWS THREE BASIC BOOLEAN OPERATIONS TO BE PERFORMED. THESE ARE THE: "LOGICAL AND;" "LOGICAL OR;" AND "EXCLUSIVE OR" OPERATIONS. EACH TYPE OF LOGIC OPERATION IS PERFORMED ON A "BIT-BY-BIT" BASIS BETWEEN THE ACCUMULATOR REGISTER AND THE CPU REGISTER OR MEMORY LOCATION SPECIFIED BY THE INSTRUCTION. A DETAILED EXPLANATION OF EACH TYPE OF LOGIC OPERATION, AND THE APPROPRIATE INSTRUCTIONS FOR EACH TYPE IS PRESENTED BELOW. THE LOGIC INSTRUCTION SET IS ALSO VALUABLE BECAUSE ALL OF THEM WILL CAUSE THE CARRY ("C") FLAG TO BE SET TO THE "Ø" CONDITION. THIS IS IMPORTANT IF ONE IS GOING TO PERFORM A SEQUENCE OF INSTRUCTIONS THAT WILL EVENTUALLY USE THE STATUS OF THE "C" FLAG TO ARRIVE AT A DECISION AS IT ALLOWS THE PROGRAMMER TO SET THE "C" FLAG TO A KNOWN STATE AT THE START OF THE SEQUENCE. ALL OTHER "FLAGS" ARE SET IN ACCORDANCE WITH RESULT OF THE LOGIC OPERATION AND HENCE THE GROUP OFTEN HAS VALUE WHEN THE PROGRAMMER DESIRES TO DETERMINE THE CONTENTS OF A REGISTER THAT HAS JUST BEEN "LOADED" INTO A REGISTER (SINCE THE "LOAD" INSTRUCTIONS DO NOT AFFECT THE STATE OF THE "FLAGS.")

THE BOOLEAN "AND" OPERATION AND INSTRUCTION SET

WHEN THE BOOLEAN "AND" INSTRUCTION IS EXECUTED, EACH BIT OF THE ACCUMULATOR WILL BE COMPARED WITH THE CORRESPONDING BIT IN THE REGISTER OR MEMORY LOCATION SPECIFIED BY THE INSTRUCTION. AS EACH BIT IS COMPARED A LOGIC RESULT WILL BE PLACED IN THE ACCUMULATOR FOR EACH BIT COMPARISON. THE LOGIC RESULT IS DETERMINED AS FOLLOWS: IF BOTH THE BIT IN THE ACCUMULATOR AND THE BIT IN THE REGISTER WITH WHICH THE OPERATION IS BEING PERFORMED ARE A "1," THEN THE ACCUMULATOR BIT

WILL BE LEFT AS A "1." FOR ALL OTHER POSSIBLE COMBINATIONS (I.E., THE ACCUMULATOR BIT = Ø AND THE OTHER REGISTER'S BIT = 1, OR IF THE ACCUMULATOR BIT = 1 AND THE OTHER REGISTER'S BIT = Ø, OR IF BOTH THE ACCUMULATOR AND THE OTHER REGISTER HAVE THE PARTICULAR BIT = Ø), THEN THE ACCUMULATOR BIT WILL BE SET TO "Ø." AN EXAMPLE WILL ILLUSTRATE THE LOGICAL "AND" OPERATION:

```
        INITIAL STATE OF THE ACCUMULATOR:     1 Ø 1 Ø 1 Ø 1 Ø

        CONTENTS OF OPERAND REGISTER:         1 1 Ø Ø 1 1 Ø Ø

        FINAL STATE OF THE ACCUMULATOR:       1 Ø Ø Ø 1 Ø Ø Ø
```

THERE ARE 7 LOGICAL "AND" INSTRUCTIONS THAT ALLOW ANY CPU REGISTER TO BE USED AS THE "AND" OPERAND. THEY ARE AS FOLLOWS:

NDA	2 4 Ø
NDB	2 4 1
NDC	2 4 2
NDD	2 4 3
NDE	2 4 4
NDH	2 4 5
NDL	2 4 6

THE CONTENTS OF THE OPERAND REGISTER IS NOT ALTERED BY AN "AND" LOGICAL INSTRUCTION.

THERE IS ALSO A LOGICAL "AND" INSTRUCTION THAT ALLOWS A WORD IN MEMORY TO BE USED AS AN OPERAND. THE ADDRESS OF THE WORD IN MEMORY THAT WILL BE USED IS "POINTED TO" BY THE CONTENTS OF THE "H" AND "L" CPU REGISTERS.

```
                        NDM          2 4 7
```

AND FINALLY THERE IS ALSO A LOGICAL "AND" "IMMEDIATE" TYPE OF INSTRUCTION THAT WILL USE THE CONTENTS OF THE WORD IMMEDIATELY FOLLOWING THE INSTRUCTION AS THE OPERAND.

```
                        NDI          Ø 4 4
```

THE NEXT GROUP OF BOOLEAN LOGIC INSTRUCTIONS DIRECT THE COMPUTER TO PERFORM THE LOGICAL "OR" OPERATION ON A "BIT-BY-BIT" BASIS WITH THE ACCUMULATOR AND THE CONTENTS OF A CPU REGISTER OR A WORD IN MEMORY. THE LOGICAL "OR" OPERATION WILL RESULT IN THE ACCUMULATOR HAVING A BIT SET TO "1" IF EITHER THAT BIT IN THE ACCUMULATOR, OR THE CORRESPONDING BIT IN THE OPERAND REGISTER (IS A "1." SINCE THE CASE WHERE BOTH THE ACCUMULATOR BIT AND THE OPERAND BIT IS A "1" ALSO SATISFIES THE RELATIONSHIP, THAT CONDITION WILL ALSO RESULT IN THE ACCUMULATOR BIT BEING A "1." IF NEITHER REGISTER HAS A ONE IN THE BIT POSITION, THEN THE ACCUMULATOR BIT REMAINS "Ø." AN EXAMPLE ILLUSTRATES THE RESULTS OF A LOGICAL "OR" OPERATION:

```
        INITIAL STATE OF THE ACCUMULATOR:     1 Ø 1 Ø 1 Ø 1 Ø

        CONTENTS OF THE OPERAND REGISTER:     1 1 Ø Ø 1 1 Ø Ø

        FINAL STATE OF THE ACCUMULATOR:       1 1 1 Ø 1 1 1 Ø
```

THERE ARE 7 LOGICAL "OR" INSTRUCTIONS THAT ALLOW ANY CPU REGISTER TO BE USED AS THE "OR" OPERAND. THEY ARE:

ORA	2 6 Ø
ORB	2 6 1
ORC	2 6 2
ORD	2 6 3
ORE	2 6 4
ORH	2 6 5
ORL	2 6 6

AND, BY USING THE "H" AND "L" REGISTERS AS "POINTERS" ONE CAN ALSO USE A WORD IN MEMORY AS AN "OR" OPERAND:

ORM	2 6 7

THERE IS ALSO THE LOGICAL "OR" "IMMEDIATE" INSTRUCTION:

ORI	Ø 6 4

AS WITH THE LOGICAL "AND" GROUP OF INSTRUCTIONS, THE LOGICAL "OR" INSTRUCTION DOES NOT ALTER THE CONTENTS OF THE OPERAND REGISTER.

THE LAST GROUP OF BOOLEAN LOGIC INSTRUCTIONS IS A VARIATION OF THE LOGIC "OR." THE VARIATION IS TERMED THE LOGICAL "EXCLUSIVE OR." THE "EXCLUSIVE OR" OPERATION IS SIMILAR TO THE "OR" EXCEPT THAT WHEN THE CORRESPONDING BITS IN BOTH THE ACCUMULATOR AND THE OPERAND REGISTER ARE A "1" THEN THE ACCUMULATOR BIT WILL BE SET TO "Ø." THUS, THE ACCUMULATOR BIT WILL BE A "1" AFTER THE OPERATION ONLY IF JUST ONE OF THE REGISTERS (ACCUMULATOR REGISTER OR OPERAND REGISTER) HAS A "1" IN THE BIT POSITION. (AGAIN, THE OPERATION IS PERFORMED ON A BIT-BY-BIT BASIS.) AN EXAMPLE PROVIDES CLARIFICATION:

INITIAL STATE OF THE ACCUMULATOR:	1 Ø 1 Ø 1 Ø 1 Ø
CONTENTS OF THE OPERAND REGISTER:	1 1 Ø Ø 1 1 Ø Ø
FINAL STATE OF THE ACCUMULATOR:	Ø 1 1 Ø Ø 1 1 Ø

THE 7 INSTRUCTIONS THAT ALLOW THE CPU REGISTERS TO BE USED AS OPERANDS ARE:

XRA	2 5 Ø
XRB	2 5 1
XRC	2 5 2
XRD	2 5 3
XRE	2 5 4
XRH	2 5 5
XRL	2 5 6

THE INSTRUCTION THAT USES REGISTERS "H" AND "L" AS POINTERS TO A MEMORY LOCATION IS:

XRM	2 5 7

AND THE "EXCLUSIVE OR" "IMMEDIATE" TYPE INSTRUCTION IS:

XRI	Ø 5 4

AS IN THE CASE OF THE LOGICAL "OR" OPERATION, THE OPERAND REGISTER IS NOT ALTERED EXCEPT FOR THE SPECIAL CASE WHEN THE "XRA" INSTRUCTION IS USED. THIS INSTRUCTION, WHICH DIRECTS THE COMPUTER TO "EXCLUSIVE OR" THE ACCUMULATOR (CPU REGISTER "A") WITH ITSELF, WILL CAUSE THE OPERAND REGISTER - SINCE IT IS ALSO THE ACCUMULATOR, TO HAVE ITS CONTENTS ALTERED (UNLESS IT IS ZERO AT THE TIME THE INSTRUCTION IS ISSUED.) THIS IS BECAUSE, REGARDLESS OF WHAT VALUE IS IN THE ACCUMULATOR, IF IT IS "EXCLUSIVE-ORED" WITH ITSELF, THE RESULT WILL ALWAYS BE ZERO! THE EXAMPLE ILLUSTRATES:

```
ORIGINAL VALUE OF THE ACCUMULATOR:     1 0 1 0 1 0 1 0

"EXCLUSIVE OR" WITH ITSELF:            1 0 1 0 1 0 1 0

FINAL VALUE OF THE ACCUMULATOR:        0 0 0 0 0 0 0 0
```

THIS ONLY OCCURS WHEN THE LOGICAL "EXCLUSIVE OR" IS PERFORMED ON THE ACCUMULATOR ITSELF. IT CAN BE SHOWN THAT THE RESULTS OF PERFORMING THE LOGICAL "OR" OR LOGICAL "AND" BETWEEN THE ACCUMULATOR AND ITSELF WILL RESULT IN THE ORIGINAL ACCUMULATOR VALUE BEING RETAINED.

INSTRUCTIONS FOR ROTATING THE CONTENTS OF THE ACCUMULATOR

IT IS OFTEN DESIRABLE TO BE ABLE TO "SHIFT" THE CONTENTS OF THE ACCUMULATOR EITHER RIGHT OR LEFT. IN A FIXED LENGTH REGISTER, A SIMPLE SHIFT OPERATION WOULD RESULT IN SOME INFORMATION BEING LOST BECAUSE WHAT WAS IN THE MSB OR LSB (DEPENDING ON IN WHICH DIRECTION THE SHIFT OCCURED) WOULD JUST BE SHIFTED RIGHT OUT OF THE REGISTER! THEREFORE, INSTEAD OF JUST SHIFTING THE CONTENTS OF A REGISTER, AN OPERATION TERMED "ROTATING" IS UTILIZED. NOW, INSTEAD OF JUST SHIFTING A BIT OFF THE END OF THE REGISTER, THE BIT IS BROUGHT AROUND TO THE OTHER END OF THE REGISTER. FOR INSTANCE, IF THE REGISTER IS "ROTATED" TO THE RIGHT, THE LSB (LEAST SIGNIFICANT BIT) WOULD BE BROUGHT AROUND TO THE POSITION OF THE MSB (MOST SIGNIFICANT BIT) IN THE REGISTER WHICH WOULD HAVE BEEN VACATED BY THE SHIFTING OF ITS ORIGINAL CONTENTS TO THE RIGHT. OR, IN THE CASE OF A SHIFT TO THE LEFT, THE MSB WOULD BE BROUGHT AROUND TO THE POSITION OF THE LSB.

SINCE THE CARRY BIT (CARRY OR "C" FLAG) CAN BE CONSIDERED AS AN EXTENSION OF THE ACCUMULATOR REGISTER, IT IS OFTEN DESIRED THAT THE CARRY BIT BE CONSIDERED AS PART OF THE ACCUMULATOR (THE MSB) DURING A ROTATE OPERATION. THE INSTRUCTION SET FOR THIS MACHINE ALLOWS TWO TYPES OF ROTATE INSTRUCTIONS. ONE CONSIDERS THE CARRY BIT TO BE PART OF THE ACCUMULATOR REGISTER FOR THE ROTATE OPERATION, AND THE OTHER TYPE DOES NOT. IN ADDITION, EACH TYPE OF ROTATE CAN BE DONE EITHER TO THE RIGHT, OR TO THE LEFT.

IT SHOULD BE NOTED THAT THE ROTATE OPERATIONS ARE PARTICULARLY VALUABLE WHEN IT IS DESIRED TO MULTIPLY A NUMBER BECAUSE SHIFTING THE CONTENTS OF A REGISTER TO THE LEFT IS A QUICK WAY TO MULTIPLY A BINARY NUMBER BY POWERS OF TWO, AND SHIFTING TO THE RIGHT PROVIDES THE INVERSE OPERATION.

ROTATING THE ACCUMULATOR LEFT

```
RLC        0 0 2
```

ROTATATING THE ACCUMULATOR LEFT WITH THE "RLC" INSTRUCTION MEANS THE MSB OF THE ACCUMULATOR WILL BE BROUGHT AROUND TO THE LSB POSITION AND ALL OTHER BITS ARE SHIFTED ONE POSITION TO THE LEFT. WHILE THIS INSTRUCTION DOES NOT SHIFT THROUGH THE CARRY BIT, THE CARRY BIT WILL BE SET BY THE STATUS OF THE MSB OF THE ACCUMULATOR AT THE START OF THE ROTATE OPERATION. (THIS FEATURE ALLOWS THE PROGRAMMER TO DETERMINE WHAT THE MSB WAS PRIOR TO THE SHIFTING OPERATION BY TESTING THE "C" FLAG AFTER THE ROTATE INSTRUCTION HAS BEEN EXECUTED.)

ROTATING THE ACCUMULATOR LEFT THROUGH THE CARRY BIT

RAL Ø 2 2

THE "RAL" INSTRUCTION WILL CAUSE THE MSB OF THE ACCUMULATOR TO GO INTO THE CARRY BIT. THE INITIAL VALUE OF THE CARRY BIT WILL BE SHIFTED AROUND TO THE LSB OF THE ACCUMULATOR. ALL OTHER BITS ARE SHIFTED ONE POSITION TO THE LEFT.

ROTATING THE ACCUMULATOR RIGHT

RRC Ø 1 2

THE "RRC" INSTRUCTION IS SIMILAR TO THE "RLC" INSTRUCTION EXCEPT THAT NOW THE LSB OF THE ACCUMULATOR IS PLACED IN THE MSB OF THE ACCUMULATOR AND ALL OTHER BITS ARE SHIFTED ONE POSITION TO THE RIGHT. ALSO, THE CARRY BIT WILL BE SET TO THE INITIAL VALUE OF THE LSB OF THE ACCUMULATOR AT THE START OF THE OPERATION.

ROTATING THE ACCUMULATOR RIGHT THROUGH THE CARRY BIT

RAR Ø 3 2

HERE, THE LSB OF THE ACCUMULATOR IS BROUGHT AROUND TO THE CARRY BIT AND THE INITIAL VALUE OF THE CARRY BIT IS SHIFTED TO THE MSB OF THE ACCUMULATOR. ALL OTHER BITS ARE SHIFTED A POSITION TO THE RIGHT.

IT SHOULD BY NOTED THAT THE "C" FLAG IS THE ONLY FLAG THAT CAN BE ALTERED BY A ROTATE INSTRUCTION. ALL OTHER FLAGS REMAIN UNCHANGED.

JUMP INSTRUCTIONS

THE INSTRUCTIONS DISCUSSED SO FAR HAVE ALL BEEN SORT OF "DIRECT ACTION" INSTRUCTIONS. THE PROGRAMMER ARRANGES A SEQUENCE OF THESE TYPES OF INSTRUCTIONS IN MEMORY AND WHEN THE PROGRAM IS STARTED THE COMPUTER PROCEEDS TO EXECUTE THE INSTRUCTIONS IN THE ORDER IN WHICH THEY ARE ENCOUNTERED. THE COMPUTER AUTOMATICALLY READS THE CONTENTS OF A MEMORY LOCATION, EXECUTES THE INSTRUCTION IT FINDS THERE, AND THEN AUTOMATICALLY INCREMENTS A SPECIAL ADDRESS REGISTER CALLED A "PROGRAM COUNTER" THAT WILL RESULT IN THE MACHINE READING THE INFORMATION CONTAINED IN THE NEXT SEQUENTIAL MEMORY LOCATION. HOWEVER, IT IS OFTEN DESIRABLE TO PERFORM A SERIES OF INSTRUCTIONS LOCATED IN ONE SECTION OF MEMORY, AND THEN SKIP OVER A GROUP OF MEMORY LOCATIONS AND START EXECUTING INSTRUCTIONS IN ANOTHER SECTION OF MEMORY. THIS ACTION CAN BE ACCOMPLISHED BY A GROUP OF INSTRUCTIONS THAT WILL CAUSE A NEW ADDRESS VALUE TO BE PLACED IN THE "PROGRAM COUNTER." THIS WILL CAUSE THE COMPUTER TO GO TO A NEW SECTION OF MEMORY AND TO CONTINUE EXECUTING INSTRUCTIONS SEQUENTIALLY FROM THE NEW MEMORY LOCATION.

THE "JUMP" INSTRUCTIONS IN THIS COMPUTER ADD CONSIDERABLE POWER TO THE MACHINE'S CAPABILITIES BECAUSE THERE ARE A SERIES OF "CONDITIONAL" JUMP INSTRUCTIONS AVAILABLE. THAT IS, THE COMPUTER CAN BE DIRECTED TO TEST THE STATUS OF A PARTICULAR FLAG ("C," "Z," "S," OR "P") AND IF THE STATUS OF THE FLAG IS THE DESIRED ONE, THEN A "JUMP" WILL BE PERFORMED. IF IT IS NOT, THE MACHINE WILL CONTINUE TO EXECUTE THE NEXT INSTRUCTION IN THE CURRENT SEQUENCE. THIS CAPABILITY PROVIDES A MEANS FOR THE COMPUTER TO "MAKE DECISIONS" AND TO MODIFY ITS OPERATION AS A FUNCTION OF THE STATUS OF THE VARIOUS FLAGS AT THE TIME THAT THE PROGRAM IS BEING EXECUTED.

IN A MANNER SIMILAR TO "IMMEDIATE" TYPES OF INSTRUCTIONS, THE "JUMP" INSTRUCTIONS REQUIRE MORE THAN ONE WORD OF MEMORY. A JUMP INSTRUCTION REQUIRES THREE WORDS TO BE PROPERLY DEFINED. (REMEMBER THAT "IMMEDIATE" TYPE INSTRUCTIONS REQUIRED TWO WORDS.) THE "JUMP" INSTRUCTION ITSELF IS THE FIRST WORD. THE SECOND WORD MUST CONTAIN THE "LOW ADDRESS" PORTION OF THE ADDRESS OF THE WORD IN MEMORY THAT THE "PROGRAM COUNTER" IS TO BE SET FOR - IN OTHER WORDS, THE NEW LOCATION FROM WHICH THE NEXT INSTRUCTION IS TO BE TAKEN. THE THIRD WORD MUST CONTAIN THE "HIGH ADDRESS" (PAGE) OF THE MEMORY ADDRESS THAT THE "PROGRAM COUNTER" WILL BE SET TO, HENCE, THE "PAGE" OR HIGH ORDER PORTION OF THE ADDRESS THAT THE COMPUTER WILL "JUMP TO" TO OBTAIN ITS NEXT INSTRUCTION.

THE UNCONDITIONAL JUMP INSTRUCTION

JMP	1 X 4

NOTE: THE MACHINE CODE 1 X 4 INDICATES THAT ANY CODE FOR THE SECOND OCTAL DIGIT OF THE MACHINE CODE IS VALID. IT IS RECOMMENDED AS A STANDARD PRACTICE THAT THE CODE Ø BE USED THUS THE TYPICAL MACHINE CODE WOULD BE 1 Ø 4.

REMEMBER, THE JUMP INSTRUCTION MUST BE FOLLOWED BY TWO MORE WORDS WHICH CONTAIN THE LOW, AND THEN THE HIGH (PAGE) PORTION OF THE ADDRESS THAT THE PROGRAM IS TO "JUMP" TO!

JUMP IF THE DESIGNATED FLAG IS TRUE (CONDITIONAL JUMP)

JTC	1 4 Ø
JTZ	1 5 Ø
JTS	1 6 Ø
JTP	1 7 Ø

AS WITH THE UNCONDITIONAL JUMP INSTRUCTION, THE CONDITIONAL JUMP INSTRUCTIONS MUST BE FOLLOWED BY TWO WORDS OF INFORMATION - THE LOW PORTION, THEN THE HIGH PORTION, OF THE ADDRESS THAT PROGRAM EXECUTION IS TO CONTINUE FROM IF THE JUMP IS EXECUTED. THE "JUMP IF TRUE" GROUP OF INSTRUCTIONS WILL ONLY JUMP TO THE DESIGNATED ADDRESS IF THE CONDITION OF THE APPROPRIATE FLAG IS TRUE (LOGICAL "1"). THUS THE "JTC" INSTRUCTION STATES THAT IF THE CARRY FLAG ("C") IS A LOGICAL "1" (TRUE) THEN THE JUMP IS TO BE EXECUTED. IF IT IS A LOGICAL "Ø" (FALSE) THEN PROGRAM EXECUTION IS TO CONTINUE WITH THE NEXT INSTRUCTION IN THE CURRENT SEQUENCE OF INSTRUCTIONS. IN A SIMILAR MANNER THE "JTZ" INSTRUCTION STATES THAT IF THE ZERO FLAG IS TRUE THEN THE JUMP IS TO BE PERFORMED. OTHERWISE THE NEXT INSTRUCTION IN THE PRESENT SEQUENCE IS EXECUTED. LIKEWISE FOR THE "JTS" AND "JTP" INSTRUCTIONS.

JUMP IF THE DESIGNATED FLAG IS FALSE (CONDITIONAL JUMP)

JFC	1 0 0
JFZ	1 1 0
JFS	1 2 0
JFP	1 3 0

AS WITH ALL JUMP INSTRUCTIONS THESE INSTRUCTIONS MUST BE FOLLOWED BY THE LOW ADDRESS THEN HIGH ADDRESS OF THE MEMORY LOCATION THAT PROGRAM EXECUTION IS TO CONTINUE FROM IF THE JUMP IS EXECUTED. THIS GROUP OF INSTRUCTIONS IS THE OPPOSITE OF THE JUMP IF THE FLAG IS TRUE GROUP. FOR INSTANCE THE "JFC" INSTRUCTION COMMANDS THE COMPUTER TO TEST THE STATUS OF THE CARRY ("C") FLAG. IF THE FLAG IS "FALSE," I.E. A LOGIC "0," THEN THE JUMP IS TO BE PERFORMED. IF IT IS "TRUE" THEN PROGRAM EXECUTION IS TO CONTINUE WITH THE NEXT INSTRUCTION IN THE CURRENT SEQUENCE OF INSTRUCTIONS. THE SAME PROCEDURE HOLDS FOR THE "JFZ," "JFS," AND "JFP" INSTRUCTIONS.

SUBROUTINE CALLING INSTRUCTIONS

QUITE OFTEN WHEN A PROGRAMMER IS DEVELOPING COMPUTER PROGRAMS THE PROGRAMMER WILL FIND THAT A PARTICULAR ALGORITHM (SEQEUNCE OF INSTRUCTIONS FOR PERFORMING A FUNCTION) CAN BE USED MANY TIMES IN DIFFERENT PARTS OF THE PROGRAM. RATHER THAN HAVE TO KEEP ENTERING THE SAME SEQUENCE OF INSTRUCTIONS AT DIFFERENT LOCATIONS IN MEMORY - WHICH WOULD NOT ONLY CONSUME THE TIME OF THE PROGRAMMER BUT WOULD ALSO RESULT IN A LOT OF MEMORY BEING USED TO PERFORM ONE PARTICULAR FUNCTION, IT IS DESIRABLE TO BE ABLE TO PUT AN OFTEN USED SEQUENCE OF COMMANDS IN ONE LOCATION IN MEMORY. THEN, WHENEVER THE PARTICULAR ALGORITHM IS REQUIRED BY ANOTHER PART OF THE PROGRAM, IT WOULD BE CONVENIENT TO "JUMP" TO THE SECTION THAT CONTAINED THE OFTEN USED ALGORITHM, PERFORM THE SEQUENCE OF INSTRUCTIONS, AND THEN RETURN BACK TO THE "MAIN" PART OF THE PROGRAM. THIS IS A STANDARD PRACTICE IN COMPUTER OPERATIONS. THE FREQUENTLY USED ALGORITHM CAN BE DESIGNATED AS A "SUBROUTINE." A SPECIAL SET OF INSTRUCTIONS ALLOWS THE PROGRAMMER TO "CALL" - IN OTHER WORDS SPECIFY A SPECIAL TYPE OF "JUMP TO," A SUBROUTINE. A SECOND TYPE OF INSTRUCTION IS USED TO TERMINATE A SEQUENCE OF INSTRUCTIONS THAT IS TO BE CONSIDERED A SUBROUTINE. THIS SPECIAL TERMINATOR WILL CAUSE THE PROGRAM OPERATION TO REVERT BACK TO THE NEXT SEQUENTIAL LOCATION IN MEMORY FOLLOWING THE INSTRUCTION THAT "CALLED" THE "SUBROUTINE." A GREAT DEAL OF COMPUTER POWER IS PROVIDED BY THE INSTRUCTION SET IN THIS MACHINE FOR "CALLING" AND "RETURNING" FROM SUBROUTINES. THIS IS BECAUSE, IN A MANNER SIMILAR TO THE CONDITIONAL JUMP INSTRUCTIONS, THERE ARE A NUMBER OF "CONDITIONAL CALLING" COMMANDS AND A NUMBER OF "CONDITIONAL RETURN" COMMANDS IN THE INSTRUCTION SET.

LIKE THE "JUMP" INSTRUCTIONS, THE "CALL" INSTRUCTIONS ALL REQUIRE THREE WORDS IN ORDER TO BE FULLY SPECIFIED. THE FIRST WORD IS THE "CALL" INSTRUCTION ITSELF. THE NEXT TWO WORDS MUST CONTAIN THE LOW AND HIGH PORTIONS OF THE STARTING ADDRESS OF THE SUBROUTINE THAT IS BEING "CALLED."

WHEN A "CALL" INSTRUCTION IS ENCOUNTERED BY THE COMPUTER, THE "CPU" WILL ACTUALLY SAVE THE CURRENT VALUE OF ITS PROGRAM COUNTER BY STORING IT IN A SPECIAL "PROGRAM COUNTER PUSH-DOWN STACK." THIS STACK IS CAPABLE OF HOLDING 7 ADDRESSES PLUS THE CURRENT OPERATING ADDRESS. WHAT THIS MEANS IS THAT THE MACHINE IS CAPABLE OF "NESTING" UP TO 7 SUBROUTINES AT ANY ONE TIME. THUS ONE CAN HAVE A SUBROUTINE, THAT IN TURN CALLS ANOTHER SUBROUTINE - THAT IN TURN CALLS ANOTHER ETC., UP TO 7 LEVELS AND THE MACHINE WILL BE ABLE TO "RETURN" TO THE INITIAL

LOCATION. THE PROGRAMMER MUST ENSURE THAT SUBROUTINES ARE NOT "NEST-ED" AT MORE THAN 7 LEVELS OTHERWISE THE "PROGRAM COUNTER PUSH-DOWN STACK" WILL "PUSH" THE ORIGINAL CALLING ADDRESS(ES) COMPLETELY OUT OF THE "PUSH-DOWN STACK" AND THE PROGRAM COULD NO LONGER AUTOMATICALLY RETURN TO THE INITIAL "CALLING" ROUTINE.

THE "RETURN" INSTRUCTION WHICH TERMINATES A SUBROUTINE ONLY RE-QUIRES ONE WORD. WHEN THE CPU ENCOUNTERS A "RETURN" INSTRUCTION IT CAUSES THE "PROGRAM COUNTER PUSH-DOWN STACK" TO "POP" UP ONE LEVEL. THIS EFFECTIVELY CAUSES THE ADDRESS "SAVED" IN THE STACK BY THE CALLING ROUTINE TO BE TAKEN AS THE NEW "PROGRAM COUNTER" AND HENCE PROGRAM EXECUTION RETURNS TO THE CALLING ROUTINE.

THE UNCONDITIONAL CALL INSTRUCTION

CAL	1 X 6

THIS INSTRUCTION FOLLOWED BY TWO WORDS CONTAINING THE LOW AND THEN THE HIGH ORDER OF THE STARTING ADDRESS OF THE SUBROUTINE THAT IS TO BE EXECUTED IS AN UNCONDITIONAL "CALL." THE SUBROUTINE WILL BE EXECUTED REGARDLESS OF THE STATUS OF THE "FLAGS." THE NEXT SEQUENTIAL ADDRESS AFTER THE "CAL" INSTRUCTION IS SAVED IN THE "PROGRAM COUNTER PUSH-DOWN STACK."

THE UNCONDITIONAL RETURN INSTRUCTION

RET	Ø X 7

THIS INSTRUCTION DIRECTS THE CPU TO UNCONDITIONALLY "POP" THE "PROGRAM COUNTER PUSH-DOWN STACK" UP ONE LEVEL. THUS PROGRAM EXECU-TION WILL CONTINUE FROM THE ADDRESS SAVED BY THE SUBROUTINE CALLING INSTRUCTION.

CALL A SUBROUTINE IF THE DESIGNATED FLAG IS TRUE

CTC	1 4 2
CTZ	1 5 2
CTS	1 6 2
CTP	1 7 2

IN A MANNER SIMILAR TO THE CONDITIONAL "JUMP IF TRUE" INSTRUCTIONS THESE INSTRUCTIONS (WHICH MUST ALL BE FOLLOWED BY THE LOW AND HIGH PORTIONS OF THE CALLED SUBROUTINE'S STARTING ADDRESS) WILL ONLY PER-FORM THE "CALL" IF THE DESIGNATED FLAG IS IN THE TRUE (LOGICAL "1") STATE. IF THE DESIGNATED FLAG IS FALSE THEN THE "CALL" INSTRUCTION IS IGNORED AND PROGRAM EXECUTION CONTINUES WITH THE NEXT SEQUENTIAL IN-STRUCTION.

RETURN FROM A SUBROUTINE IF THE DESIGNATED FLAG IS TRUE

RTC	Ø 4 3
RTZ	Ø 5 3
RTS	Ø 6 3
RTP	Ø 7 3

THESE ONE WORD INSTRUCTIONS WILL CAUSE A SUBROUTINE TO BE TERMI-NATED ONLY IF THE DESIGNATED FLAG IS IN THE LOGICAL "1" (TRUE) STATE.

CALL A SUBROUTINE IF THE DESIGNATED FLAG IS FALSE

CFC	1 Ø 2
CFZ	1 1 2
CFS	1 2 2
CFP	1 3 2

THESE INSTRUCTIONS ARE THE OPPOSITE OF THE PREVIOUS GROUP OF CALLING COMMANDS. THE SUBROUTINE IS CALLED ONLY IF THE DESIGNATED FLAG IS IN THE FALSE (LOGICAL Ø) CONDITION. REMEMBER, THESE INSTRUCTIONS MUST BE FOLLOWED BY TWO WORDS WHICH CONTAIN THE LOW AND THEN HIGH PART OF THE STARTING ADDRESS OF THE SUBROUTINE THAT IS TO BE EXECUTED IF THE DESIGNATED FLAG IS FALSE. IF THE FLAG IS TRUE, THE SUBROUTINE WILL NOT BE CALLED AND PROGRAM OPERATION WILL CONTINUE WITH THE NEXT INSTRUCTION IN THE CURRENT SEQUENCE.

RETURN FROM A SUBROUTINE IF THE DESIGNATED FLAG IS FALSE

RFC	Ø Ø 3
RFZ	Ø 1 3
RFS	Ø 2 3
RFP	Ø 3 3

THESE ONE WORD INSTRUCTIONS WILL TERMINATE A SUBROUTINE (POP THE "PROGRAM COUNTER STACK" UP ONE LEVEL) IF THE DESIGNATED FLAG IS FALSE. OTHERWISE THE INSTRUCTION IS IGNORED AND PROGRAM OPERATION IS CONTINUED WITH THE NEXT INSTRUCTION IN THE SUBROUTINE.

THE SPECIAL "RESTART" SUBROUTINE CALL INSTRUCTIONS

THERE IS A SPECIAL PURPOSE INSTRUCTION AVAILABLE THAT EFFECTIVELY SERVES AS A ONE WORD SUBROUTINE CALL (REMEMBER THAT IT NORMALLY REQUIRES THREE WORDS TO SPECIFY A SUBROUTINE CALL.) THIS SPECIAL INSTRUCTION ALLOWS THE PROGRAMMER TO CALL A SUBROUTINE THAT STARTS AT ANY ONE OF EIGHT SPECIALLY DESIGNATED MEMORY LOCATIONS. THE EIGHT SPECIAL MEMORY LOCATIONS ARE AT LOCATIONS: ØØØ, Ø1Ø, Ø2Ø, Ø3Ø, Ø4Ø, Ø5Ø, Ø6Ø AND Ø7Ø ON PAGE ZERO. THERE ARE EIGHT VARIATIONS OF THE RESTART INSTRUCTION - ONE FOR EACH OF THE ABOVE ADDRESSES. THUS, THE ONE WORD INSTRUCTION CAN SERVE TO "CALL" A SUBROUTINE AT THE SPECIFIED STARTING LOCATION (INSTEAD OF HAVING TWO ADDITIONAL WORDS TO SPECIFY THE STARTING ADDRESS OF THE SUBROUTINE.) IT IS OFTEN CONVENIENT TO UTILIZE A RESTART COMMAND AS A QUICK "CALL" TO AN OFTEN USED SUBROUTINE, OR AS AN EASY WAY TO CALL SHORT "STARTING" ROUTINES FOR LARGE PROGRAMS - HENCE THE NAME FOR THE TYPE OF INSTRUCTION. THE EIGHT RESTART INSTRUCTIONS - ALONG WITH THE STARTING ADDRESS OF THE SUBROUTINE THAT EACH WILL AUTOMATICALLY "CALL" IS AS FOLLOWS:

INSTRUCTION (MNEMONIC)	MACHINE CODE	SUBROUTINE STARTING ADDRESS
RST Ø	Ø Ø 5	ØØØ ØØØ
RST 1	Ø 1 5	ØØØ Ø1Ø
RST 2	Ø 2 5	ØØØ Ø2Ø
RST 3	Ø 3 5	ØØØ Ø3Ø
RST 4	Ø 4 5	ØØØ Ø4Ø
RST 5	Ø 5 5	ØØØ Ø5Ø
RST 6	Ø 6 5	ØØØ Ø6Ø
RST 7	Ø 7 5	ØØØ Ø7Ø

INPUT INSTRUCTIONS

IN ORDER TO RECEIVE INFORMATION FROM AN EXTERNAL DEVICE THE COMPUTER MUST UTILIZE A GROUP OF SPECIAL SIGNAL LINES. THE SCELBI-8H COMPUTER IS DESIGNED TO HANDLE UP TO SIX GROUPS (EACH GROUP HAVING EIGHT SIGNAL LINES) OF INPUT SIGNALS. A GROUP OF SIGNALS IS ACCEPTED AT THE COMPUTER BY WHAT IS REFERRED TO AS AN "INPUT PORT." THE COMPUTER CONTROLS THE OPERATION OF THE "INPUT PORTS." UNDER PROGRAM CONTROL, THE COMPUTER CAN BE DIRECTED TO OBTAIN THE INFORMATION THAT IS ON THE GROUP OF LINES COMING IN TO ANY "INPUT PORT" AND BRING IT INTO THE ACCUMULATOR. VARIOUS TYPES OF EXTERNAL EQUIPMENT - SUCH AS A KEYBOARD - CAN BE CONNECTED TO THE INPUT PORT(S). WHEN IT IS DESIRED TO HAVE INFORMATION OBTAINED FROM A SPECIFIC "INPUT PORT" AN INPUT INSTRUCTION MUST BE USED. THE INPUT INSTRUCTION SIMPLY IDENTIFIES WHICH INPUT PORT IS TO BE OPERATED AND WHEN EXECUTED CAUSES THE SIGNAL LEVELS ON THE SELECTED INPUT PORT TO BE BROUGHT INTO THE "A" CPU REGISTER (ACCUMULATOR). THE SIX STANDARD INPUT PORTS ON THE SCELBI-8H MINI COMPUTER ARE DESIGNATED AS INPUT PORTS Ø - 5.

INP Ø	1 Ø 1
INP 1	1 Ø 3
INP 2	1 Ø 5
INP 3	1 Ø 7
INP 4	1 1 1
INP 5	1 1 3

AN INPUT INSTRUCTION ONLY REQUIRES ONE MACHINE CODE WORD. IT IS ALSO IMPORTANT TO NOTE THAT AN INPUT INSTRUCTION - WHICH BRINGS NEW DATA INTO THE ACCUMULATOR - DOES NOT AFFECT THE STATUS OF ANY OF THE CPU FLAGS.

OUTPUT INSTRUCTIONS

IN ORDER TO OUTPUT INFORMATION TO AN EXTERNAL DEVICE THE COMPUTER UTILIZES ANOTHER GROUP OF SIGNAL LINES WHICH ARE REFERRED TO AS "OUTPUT PORTS." THE SCELBI-8H IS STANDARDLY EQUIPPED TO SERVICE UP TO EIGHT "OUTPUT PORTS." (EACH OUTPUT PORT ACTUALLY CONSIST OF EIGHT SIGNAL LINES.) AN OUTPUT INSTRUCTION CAUSES THE CONTENTS OF THE CPU "A" REGISTER (ACCUMULATOR) TO BE TRANSFERRED TO THE SIGNAL LINES OF THE DESIGNATED OUTPUT PORT. THE STANDARLY EQUIPPED OUTPUT PORTS ARE DESIGNATED AS OUTPUT PORTS 1Ø - 17.

OUT 1Ø	1 2 1
OUT 11	1 2 3
OUT 12	1 2 5
OUT 13	1 2 7
OUT 14	1 3 1
OUT 15	1 3 3
OUT 16	1 3 5
OUT 17	1 3 7

AN OUTPUT INSTRUCTION ONLY REQUIRES ONE MACHINE CODE WORD. IT DOES NOT AFFECT THE STATUS OF ANY OF THE CPU FLAGS. OUTPUT PORT(S) ARE CONNECTED TO EXTERNAL DEVICES - SUCH AS AN OSCILLOSCOPE DISPLAY SYSTEM, AND PROVIDE CAPABILITY FOR THE COMPUTER TO DISPLAY INFORMATION OR OTHERWISE CONTROL THE OPERATION OF EXTERNAL DEVICES.

THE HALT INSTRUCTION

THERE IS ONE MORE INSTRUCTION FOR THE COMPUTER'S INSTRUCTION SET. THIS INSTRUCTION DIRECTS THE CPU TO STOP ALL OPERATIONS AND TO REMAIN IN THAT STATE UNTIL AN "INTERRUPT" SIGNAL IS RECEIVED. IN THE STANDARD SCELBI-8H AN "INTERRUPT" SIGNAL MUST BE GENERATED BY THE OPERATOR PRESSING A SWITCH ON THE FRONT PANEL OF THE COMPUTER. THIS INSTRUCTION IS NORMALLY USED WHEN THE PROGRAMMER DESIRES TO HAVE A PROGRAM BE TERMINATED, OR WHEN IT IS DESIRED TO HAVE THE MACHINE WAIT FOR AN OPERATOR TO SET UP EXTERNAL CONDITIONS ETC.. THERE ARE THREE MACHINE CODE INSTRUCTIONS THAT MAY BE USED FOR THE HALT COMMAND:

```
HLT        0 0 0
HLT        0 0 1
HLT        3 7 7
```

THE HALT INSTRUCTION DOES NOT AFFECT THE STATUS OF THE CPU FLAGS. IT IS A ONE WORD INSTRUCTION.

INFORMATION ON INSTRUCTION EXECUTION TIMES

WHEN PROGRAMMING FOR REAL TIME APPLICATIONS IT IS IMPORTANT TO KNOW HOW MUCH TIME EACH TYPE OF INSTRUCTION REQUIRES TO BE EXECUTED. WITH THIS INFORMATION THE PROGRAMMER CAN DEVELOPE "TIMING LOOPS" OR DETERMINE WITH SUBSTANTUAL ACCURACY HOW MUCH TIME IT TAKES TO PERFORM A PARTICULAR SERIES OF INSTRUCTIONS. THIS INFORMATION IS ESPECIALLY IMPORTANT WHEN DEALING WITH PROGRAMS THAT CONTROL THE OPERATION OF EXTERNAL DEVICES WHICH REQUIRE EVENTS TO OCCUR AT SPECIFIC TIMES.

THE FOLLOWING TABLE PROVIDES THE NOMINAL INSTRUCTION EXECUTION TIME FOR EACH CATEGORY OF INSTRUCTION USED IN A STANDARD SCELBI-8H SYSTEM. THE MASTER CLOCK IN A SCELBI-8H MINI-COMPUTER HAS AN ACCURACY RATED AT PLUS OR MINUS 2% OF THE NOMINAL VALUE. THE TABLE SHOWS THE NUMBER OF "CYCLE STATES" REQUIRED BY THE TYPE OF INSTRUCTION FOLLOWED BY THE NOMINAL TIME REQUIRED TO PERFORM THE ENTIRE INSTRUCTION. SINCE EACH STATE EXECUTES IN 4 MICROSECONDS (U'SECS) THE TOTAL TIME REQUIRED TO PERFORM THE INSTRUCTION AS SHOWN IN THE TABLE IS OBTAINED BY MULTIPLYING THE NUMBER OS STATES BY 4 MICROSECONDS. BY KNOWING THE NUMBER OF STATES REQUIRED FOR EACH INSTRUCTION THE PROGRAMMER CAN OFTEN REARRANGE AN ALGORITHM OR SUBSTITUTE DIFFERENT TYPES OF INSTRUCTIONS TO PROVIDE PROGRAMS THAT HAVE SPECIFIC EVENTS OCCURRING AT PRECISELY TIMED INTERVALS.

INSTRUCTION EXECUTION TIME TABLE

TYPE OF INSTRUCTION	# OF STATES	TOTAL EXECUTION TIME
LOAD DATA FROM ONE CPU REGISTER TO ANOTHER CPU REGISTER	5	20 U'SECS
LOAD DATA FROM A CPU REGISTER TO A LOCATION IN MEMORY	7	28 U'SECS

INSTRUCTION EXECUTION TIME TABLE

TYPE OF INSTRUCTION	# OF STATES	TOTAL EXECUTION TIME
LOAD DATA FROM A LOCATION IN MEMORY TO A CPU REGISTER	8	32 U'SECS
LOAD "IMMEDIATE" DATA INTO A CPU REGISTER	8	32 U'SECS
LOAD "IMMEDIATE" DATA INTO A LOCATION IN MEMORY	9	36 U'SECS
INCREMENT OR DECREMENT A CPU REGISTER	5	20 U'SECS
ARITHMETIC INSTRUCTION BETWEEN THE ACCUMULATOR AND A CPU REGISTER	5	20 U'SECS
COMPARE BETWEEN THE ACCUMULATOR AND A CPU REGISTER	5	20 U'SECS
ARITHMETIC OR COMPARE INSTRUCTION BETWEEN THE ACCUMULATOR AND A WORD IN MEMORY	8	32 U'SECS
"IMMEDIATE" TYPE ARITHMETIC AND COMPARE INSTRUCTIONS	8	32 U'SECS
BOOLEAN MATH OPERATIONS BETWEEN ACCUMULATOR AND CPU REGISTERS	5	20 U'SECS

INSTRUCTION EXECUTION TIME TABLE

TYPE OF INSTRUCTION	# OF STATES	TOTAL EXECUTION TIME
BOOLEAN MATH OPERATIONS BETWEEN ACCUMULATOR AND A LOCATION IN MEMORY	8	32 U'SECS
BOOLEAN "IMMEDIATE" INSTRUCTIONS	8	32 U'SECS
ACCUMULATOR ROTATE INSTRUCTIONS	5	20 U'SECS
UNCONDITIONAL JUMP OR CALL INSTRUCTIONS	11	44 U'SECS
CONDITIONAL JUMP OR CALL INSTRUCTIONS WHEN CONDITION IS NOT SATISFIED	9	36 U'SECS
AND CONDITIONAL JUMP OR CALL INSTRUCTIONS WHEN CONDITION IS SATISFIED	11	44 U'SECS
UNCONDITIONAL RETURN INSTRUCTION	5	20 U'SECS
CONDITIONAL RETURN INSTRUCTION WHEN CONDITION IS NOT SATISFIED	3	12 U'SECS
CONDITIONAL RETURN INSTRUCTION WHEN CONDITION IS SATISFIED	5	20 U'SECS
RESTART INSTRUCTION	5	20 U'SECS
OUTPUT INSTRUCTION	6	24 U'SECS
INPUT INSTRUCTION	8	32 U'SECS
HALT INSTRUCTION	4	16 U'SECS

SCELBI-8H OPERATING INFORMATION

THE STANDARD SCELBI-8H MINI-COMPUTER IS OPERATED THROUGH THE USE OF 11 CHASSIS PANEL SWITCHES. THE STATUS OF THE COMPUTER AND THE RESULTS OF VARIOUS OPERATIONS CAN BE OBSERVED ON THE SCELBI 1104-FRONT PANEL CARD.

THE CHASSIS PANEL SWITCHES, FROM LEFT TO RIGHT, CONSIST OF THREE MOMENTARY PUSH BUTTON SWITCHES AND EIGHT TOGGLE SWITCHES.

THE LEFT-MOST PUSH BUTTON SWITCH IS THE "INTERRUPT" BUTTON. WHENEVER THIS BUTTON IS DEPRESSED THE CENTRAL PROCESSOR UNIT (CPU) WILL RECEIVE A SIGNAL THAT INDICATES IT IS TO INTERRUPT THE PROCESS IT IS ENGAGED IN AND PREPARE TO RECEIVE AN INSTRUCTION FROM THE CHASSIS PANEL TOGGLE SWITCHES. UPON RECEIPT OF THIS SIGNAL THE CPU WILL FINISH PERFORMING ANY INSTRUCTION IT MIGHT BE EXECUTING AND THEN ACKNOWLEDGE RECEIPT OF THE INTERRUPT COMMAND BY LIGHTING THE "INT" LAMP ON THE SCELBI 1104- FRONT PANEL CARD. NOTE THAT THE LENGTH OF TIME BEFORE THE "INT" LAMP ON THE FRONT PANEL CARD COMES ON IS A FUNCTION OF THE STATE OF THE MACHINE AT THE TIME THE INTERRUPT BUTTON WAS DEPRESSED. THE POSSIBLE CONDITIONS CAN BE SUMMARIZED AS FOLLOWS:

1. IF THE SCELBI-8H WAS IN THE "RUN" MODE THE "INT" LAMP WOULD APPEAR TO COME ON INSTANTANEOUSLY AND THE "RUN" LAMP ON THE FRONT PANEL CARD WOULD SIMULTANEOUSLY EXTINGUISH.

2. IF THE MACHINE WAS IN THE "STOP" STATE THE "INT" LIGHT WOULD IMMEDIATELY TURN ON AND THE "STOP" LAMP WOULD TURN OFF.

3. IF THE MACHINE WAS OPERATING IN THE "STEP" MODE (BY USE OF THE "STEP" PUSH BUTTON SWITCH) THEN THE "INT" LAMP WOULD NOT LIGHT UP UNTIL THE OPERATOR FINISHED STEPPING THROUGH THE CURRENT INSTRUCTION BEING EXECUTED BY THE MACHINE.

4. IF THE "INTERRUPT" BUTTON WAS DEPRESSED WHILE THE "INT" LAMP WAS ALREADY ON (INDICATING AN INSTRUCTION WAS CURRENTLY BEING EXECUTED IN THE "INTERRUPT" MODE) THEN THE "INT" LAMP WOULD REMAIN ON AFTER THE FIRST INTERRUPT INSTRUCTION WAS EXECUTED. THE END OF THE PREVIOUS INTERRUPT COMMAND AND THE START OF THE NEW INTERRUPT COMMAND WOULD HAVE TO BE DISCERNED BY CAREFUL OBSERVATION OF THE CYCLE "STATUS" LAMPS ON THE FRONT PANEL CARD WHOSE SIGNIFICANCE WILL BE EXPLAINED IN DETAIL LATER IN THIS CHAPTER.

THE NEXT PUSH BUTTON IS THE "STEP" BUTTON. THIS BUTTON ALLOWS THE OPERATOR TO EXECUTE INSTRUCTIONS IN SINGLE STEPS. WHEN ENTERING A MULTI-WORD INSTRUCTION IN THE "INTERRUPT" MODE THIS BUTTON ALLOWS THE MACHINE TO PAUSE AT EACH WORD SO THAT THE CHASSIS TOGGLE SWITCHES CAN BE SET UP FOR THE NEXT WORD OF THE INSTRUCTION. THIS BUTTON WILL ALSO ALLOW AN OPERATOR TO STEP SLOWLY THROUGH EACH INSTRUCTION IN A PROGRAM WHILE THE FRONT PANEL CARD LIGHTS ARE OBSERVED AS AN AID TO PROGRAM VERIFICATION OR MONITORING.

THE THIRD PUSH BUTTON IS THE "RUN" BUTTON. DEPRESSING THIS BUTTON CAUSES THE COMPUTER TO RESUME EXECUTING A PROGRAM IN MEMORY AT THE NORMAL AUTOMATIC RATE. PROGRAM EXECUTION WILL CONTINUE FROM THE MEMORY WORD LOCATION SPECIFIED BY THE CURRENT CONTENTS OF THE CPU'S PROGRAM COUNTER.

THE EIGHT TOGGLE SWITCHES ON THE CHASSIS PANEL ALLOW INSTRUCTIONS AND DATA TO BE FED TO THE COMPUTER BY THE OPERATOR IN CONJUNCTION WITH THE "INTERRUPT" AND "STEP" PUSH BUTTONS. THE LEFT-MOST TOGGLE SWITCH CONTROLS THE INPUT OF BIT B7 (THE MOST SIGNIFICANT BIT) AND THE SWITCHES ARE ARRANGED IN DESCENDING BIT ORDER DOWN TO THE RIGHT-MOST TOGGLE SWITCH WHICH IS FOR BIT B0 (THE LEAST SIGNIFICANT BIT.)

THE 11 CHASSIS PANEL SWITCHES ALLOW THE OPERATOR: TO LOAD PROGRAMS INTO THE MEMORY OR ALTER MEMORY CONTENTS. TO EXAMINE THE CONTENTS OF WORDS IN MEMORY OR THE CONTENTS OF CPU REGISTERS (USING THE FRONT PANEL CARD INDICATORS). TO ALTER THE CONTENTS OF CPU REGISTERS. TO PERFORM INPUT/OUTPUT (I/O) OPERATIONS. TO START AND STOP EXECUTION OF PROGRAMS IN MEMORY. TO INSERT OR "JAM" IN INSTRUCTIONS VIA THE "INTERRUPT" FACILITY IN BETWEEN INSTRUCTIONS THAT ARE BEING EXECUTED FROM MEMORY AND THUS ALTER THE RESULTS OF A COMPUTATION OR ENTER NEW DATA INTO A PROGRAM WITHOUT ACTUALLY CHANGING THE PROGRAM STORED IN MEMORY. IN SUMMARY, THE SWITCHES ALLOW THE OPERATOR TO MANUALLY CONTROL THE COMPLETE OPERATION OF A SCELBI-8H MINI-COMPUTER.

THE SCELBI 1104- FRONT PANEL CARD MAKES OPERATING THE SCELBI-8H MINI-COMPUTER A REAL PLEASURE. A CLEVERLY DESIGNED DISPLAY SYSTEM ON THE 1104- CARD ALLOWS THE OPERATOR TO ASCERTAIN THE STATUS OF THE MACHINE AT ALL TIMES. THE LAMPS MAY BE USED: TO DISPLAY THE CONTENTS OF SPECIFIC MEMORY LOCATIONS OR CPU REGISTERS. TO OBSERVE THE TRANSFER OF INFORMATION FROM AND TO EXTERNAL (I/O) DEVICES. TO OBSERVE THE STEP-BY-STEP EXECUTION OF PROGRAMS STORED IN MEMORY. TO ASCERTAIN THE GENERAL TYPE OF OPERATIONS (STATUS) BEING PERFORMED BY THE MACHINE AS WELL AS ITS MODE OF OPERATION (RUN, STOP, OR INTERRUPT). THE CARD ALSO HAS LAMPS THAT VERIFY THE PRESENCE OF NORMAL POWER SUPPLY VOLTAGES TO THE SCELBI-8H MINI-COMPUTER.

THE LIGHTS ON THE FRONT PANEL CARD ARE ARRANGED IN THE FOLLOWING PATTERNS.

ALONG THE TOP ROW: THE FIRST TWO LAMPS ON THE LEFT SIDE OF THE ROW ARE USED TO INDICATE THE PRESENCE OF THE +5 VOLT AND -9 VOLT POWER SUPPLY VOLTAGES. WHENEVER THE COMPUTER IS ON BOTH OF THESE LAMPS SHOULD GLOW BRIGHTLY AND STEADILY. IF ONE OR BOTH OF THESE LAMPS SHOULD FAIL TO COME ON WHEN POWER IS INITIALLY APPLIED TO THE COMPUTER, OR SHOULD SUDDENLY GO OUT, THEN ALL POWER TO THE COMPUTER SHOULD BE IMMEDIATLY DISCONNECTED AS IT INDICATES THAT THE RESPECTIVE POWER SUPPLY VOLTAGE IS NOT PRESENT. NO ATTEMPT SHOULD BE MADE TO OPERATE THE COMPUTER IF EITHER POWER SUPPLY VOLTAGE IS ABSENT!

THE NEXT SIX LAMPS ON THE TOP ROW OF THE LEFT HAND SIDE OF THE CARD ARE ARRANGED IN TWO GROUPS OF THREE LAMPS. THESE LAMPS ARE USED TO INDICATE SEVERAL TYPES OF INFORMATION DEPENDING ON THE STATUS OF THE MACHINE. FOR THE MAJORITY OF INSTRUCTIONS EXECUTED BY THE MACHINE THESE LAMPS WILL INDICATE WHICH "PAGE" (HIGH ORDER PORTION OF A MEMORY ADDRESS) WILL NEXT BE ACCESSED BY THE COMPUTER. EACH GROUP OF LAMPS CAN REPRESENT AN OCTAL NUMBER FROM 0 TO 7 DEPENDING ON WHICH LAMPS ARE LIT. THUS THE TWO GROUPS OF THREE LAMPS CAN DENOTE ALL THE POSSIBLE PAGES OF MEMORY THAT A SCELBI-8H COULD ACCESS. THAT IS PAGE 00 TO PAGE 77 (OCTAL). FOR A SMALL GROUP OF INSTRUCTIONS, NOTABLY THOSE RELATED TO I/O OPERATIONS, THESE LAMPS WILL DISPLAY OTHER INFORMATION WHICH IS EXPLAINED LATER IN THIS CHAPTER. HOWEVER, SINCE THEY ARE USED PRIMARILY TO DENOTE THE "PAGE" ADDRESS THEY ARE APPROPRIATELY LABELED AS THE "PAGE" INDICATORS.

FIGURE 1 ILLUSTRATES THE LIGHTS THAT ARE ON THE LEFT HAND TOP ROW

ON THE FRONT PANEL CARD AND THEIR LABELS.

```
+5V -9V          P A G E

 0   0    0   0   0    0   0   0
```

FIGURE 1

THE TOP ROW OF LIGHTS ON THE RIGHT HAND SIDE OF THE CARD ARE REPRESENTED PICTORIALLY IN FIGURE 2. THESE LIGHTS GENERALLY ARE USED TO DISPLAY THE LOW ORDER ADDRESS (LOCATION ON A PAGE) OF THE NEXT WORD IN MEMORY THAT WILL BE ACCESSED BY THE COMPUTER. HOWEVER, FOR A FEW INSTRUCTIONS, SUCH AS I/O OPERATIONS, THEY DISPLAY INFORMATION RELATED TO THE I/O TRANSFER. THESE LIGHTS ARE LABELED "MEMORY ADDRESS" IN KEEPING WITH THE INFORMATION THAT THEY USUALLY DISPLAY.

```
        M E M O R Y   A D D R E S S

 0   0    0   0   0    0   0   0      .
```

FIGURE 2

THE MEMORY ADDRESS LAMPS ARE GROUPED TO ALLOW EASY REPRESENTATION OF OCTAL NUMBERS. THE LAMPS AS GROUPED CAN INDICATE THE OCTAL NUMBERS FROM ØØØ TO 377 WHICH ARE ALL THE POSSIBLE ADDRESSES ON A "PAGE" IN MEMORY.

THE SECOND ROW OF LIGHTS ON THE RIGHT HAND SIDE OF THE FRONT PANEL CARD ARE ARRANGED AND LABELED AS FOLLOWS.

```
 0   0    0   0   0    0   0   0

    M E M O R Y   C O N T E N T S
```

FIGURE 3

THESE LIGHTS SERVE PRIMARILY TO SHOW THE CONTENTS OF THE LAST WORD IN MEMORY THAT WAS ACCESSED BY THE COMPUTER. THE USER IS CAUTIONED TO NOTE THAT THE MEMORY CONTENTS BEING DISPLAYED ARE GENERALLY THOSE OF THE WORD IN MEMORY WHOSE ADDRESS IS ONE LESS THAN THAT CURRENTLY DISPLAYED BY THE "PAGE" AND "MEMORY ADDRESS" LAMPS BECAUSE THOSE LAMPS GENERALLY SHOW THE NEXT WORD IN MEMORY THAT WILL BE ACCESSED WHILE THE "MEMORY CONTENTS" LAMPS SHOW THE CONTENTS OF THE LAST MEMORY LOCATION ACCESSED BY THE COMPUTER. THIS CONVENTION IS EASILY LEARNED BY THE OPERATOR AND IS OF PARTICULAR VALUE WHEN THE MACHINE IS PERFORMING "JMP" (JUMP) AND "CAL" (CALL) INSTRUCTIONS AS THE OPERATOR CAN EASILY SEE THAT THE COMPUTER IS GOING TO A NEW SECTION IN MEMORY. AN EXCEPTION TO THE CONVENTION OCCURS WHEN A "HLT" (HALT) INSTRUCTION HAS BEEN EXECUTED BY THE COMPUTER. THE SPECIFIC RELATIONSHIP BETWEEN THE "MEMORY CONTENTS" LAMPS AND THE ADDRESSING LAMPS FOR EACH TYPE OF INSTRUCTION IS PRESENTED IN A COMPREHENSIVE TABLE LATER IN THIS CHAPTER.

IN ADDITION TO SHOWING THE CONTENTS OF THE LAST MEMORY LOCATION ACCESSED BY THE COMPUTER, THESE LAMPS ALSO SERVE TO PRESENT THE CONTENTS OF CPU REGISTERS AND CAN PROVIDE OTHER TYPES OF INFORMATION WHICH WILL BE PRESENTED LATER IN THIS CHAPTER. THEY ARE LABELED THE "MEMORY CONTENTS" LAMPS AS A REMINDER OF THE TYPE OF INFORMATION THEY MOST OFTEN DISPLAY.

THE SECOND ROW OF LIGHTS ON THE LEFT HAND SIDE OF THE CARD ARE ARRANGED AND LABELED AS SHOWN IN FIGURE 4.

```
 0       0     0     0       0

RUN     INT    STATUS      STOP
```

FIGURE 4

THE LAMP LABELED "RUN" IS LIT WHEN THE COMPUTER IS OPERATING UNDER PROGRAM CONTROL AT THE NORMAL MACHINE EXECUTION RATE.

THE LAMP LABELED "STOP" LIGHTS TO SIGNIFY THAT THE COMPUTER HAS ENCOUNTERED A "HLT" (HALT) INSTRUCTION AND THAT THE COMPUTER IS IN THE "STOPPED" CONDITION.

- N O T I C E -

WHENEVER THE SCELBI-8H ENCOUNTERS A "HLT" (HALT) INSTRUCTION THE "STOP" LAMP WILL LIGHT. IN ORDER TO RESUME OPERATION THE OPERATOR MUST DEPRESS THE "INT" PUSH BUTTON SWITCH ON THE CHASSIS AND CAUSE AT LEAST ONE INSTRUCTION TO BE EXECUTED VIA THE INTERRUPT MODE. QUITE OFTEN IT WILL BE DESIRABLE TO ISSUE A "JMP" (JUMP) COMMAND OR OTHER SPECIFIC INSTRUCTION AT SUCH A POINT. HOWEVER, IN THE EVENT THE USER DESIRES TO SIMPLY CONTINUE EXECUTING A PROGRAM (WITH THE INSTRUCTION IN MEMORY IMMEDIATELY FOLLOWING THE ONE THAT CAUSED THE COMPUTER TO HALT) IT IS RECOMMENDED THAT A "NO OPERATION" (NOP) TYPE OF INSTRUCTION BE USED AS THE INTERRUPT COMMAND. A GOOD "NO OPERATION" INSTRUCTION TO USE IS THE "LAA" (LOAD REGISTER A TO REGISTER A) INSTRUCTION. THIS INSTRUCTION WILL NOT DISTURB THE RESULTS OF ANY PROGRAM THAT THE MACHINE MIGHT BE PROCESSING AND WILL SATISFY THE REQUIREMENT OF ISSUEING AN "INTERRUPT" COMMAND TO MOVE THE MACHINE OUT OF THE "STOPED" CONDITION.

THE MIDDLE GROUP OF THREE LAMPS LABELED "INT" AND "STATUS" ARE VERY IMPORTANT INDICATORS FOR THE COMPUTER OPERATOR. THE "INT" LIGHT IS TURNED ON WHENEVER THE COMPUTER IS PROCESSING AN "INTERRUPT" COMMAND VIA THE CHASSIS SWITCHES. THE NEXT TWO LAMPS LABELED "STATUS" ARE USED TO INFORM THE OPERATOR JUST WHAT TYPE OF OPERATION THE COMPUTER IS CURRENTLY PERFORMING. THESE LAMPS ARE PARTICULARLY IMPORTANT DURING MULTI-WORD INSTRUCTIONS AS THEIR CONDITION IS USED TO INFORM THE OPERATOR WHEN TO SET UP THE CHASSIS TOGGLE SWITCHES TO PROVIDE NEW CODES FOR THE NEXT WORD OF A MULTI-WORD OPERATION, SUCH AS WHEN "IMMEDIATE" OR "JMP" INSTRUCTIONS ARE BEING INSERTED VIA THE CHASSIS SWITCHES. THE STATUS LAMPS ARE ALSO VALUABLE AS PROGRAM VERIFICATION AIDS AS THEY INFORM THE OPERATOR AS TO JUST WHAT TYPE OF OPERATION IS BEING PERFORMED BY THE CPU AT EACH STEP IN THE EXECUTION OF A PROGRAM.

THE MEANINGS OF THE STATUS LAMPS SHOULD BE MEMORIZED BY THE USER. THE TWO LAMPS CAN REPRESENT FOUR POSSIBLE CONDITIONS AS PRESENTED IN THE ILLUSTRATION BELOW.

CONDITION OF STATUS LAMPS	TYPE OF COMPUTER PROCESS
0 0 BOTH LAMPS OFF	THE CPU IS PERFORMING A ONE WORD INSTRUCTION OR READING THE FIRST WORD OF A MULTI-WORD INSTRUCTION.
* 0 LEFT STATUS LAMP IS ON RIGHT STATUS LAMP IS OFF	THE CPU IS READING THE NEXT WORD OF A MULTI-WORD INSTRUCTION OR PERFORMING THE LATTER PART OF A MULTI-WORD COMMAND.
0 * RIGHT STATUS LAMP IS ON LEFT STATUS LAMP IS OFF	COMPUTER IS PERFORMING AN I/O (INPUT/OUTPUT) OPERATION.
* * BOTH STATUS LAMPS ARE ON	COMPUTER IS PERFORMING A MEMORY WRITE OPERATION.

FIGURE 5

INITIALIZING THE SCELBI-8H FOLLOWING POWER TURN-ON

WHEN POWER IS INITIALLY APPLIED TO THE STANDARD SCELBI-8H MINI-COMPUTER THERE WILL NOT BE ANY PROGRAM IN THE MACHINE'S MEMORY (UNLESS IT IS EQUIPPED WITH SPECIAL READ-ONLY-MEMORY (ROM) ELEMENTS), AS THE STANDARD SEMI-CONDUCTOR "RAM" MEMORY ELEMENTS CANNOT RETAIN INFORMATION WHEN POWER IS NOT APPLIED. IN ADDITION, WHEN THE COMPUTER IS FIRST TURNED ON THE CIRCUITS IN THE CPU WILL BE IN VARIOUS RANDOM STATES. IT IS THUS NECESSARY TO PERFORM A SEQUENCE OF OPERATIONS VIA THE CHASSIS SWITCHES TO BRING THE COMPUTER TO A SET OF CONDITIONS FROM WHICH NORMAL OPERATIONS CAN PROCEED. THE RECOMMENDED PROCEDURE TO BE PERFORMED IMMEDIATELY FOLLOWING POWER TURN-ON IS DETAILED BELOW. IT CONSISTS OF USING THE INTERRUPT CAPABILITY TO "INSERT" AN INSTRUCTION INTO THE CPU DIRECTING IT TO JUMP TO A NON-EXISTANT ADDRESS IN MEMORY. IN STANDARD SCELBI-8H SYSTEMS THIS IS READILY ACCOMPLISHED BY USING AN ADDRESS ON PAGE 77 (OCTAL) BECAUSE THERE WILL NOT BE ANY MEMORY ELEMENTS ASSIGNED TO SUCH A HIGH ADDRESS. THIS PROCEDURE WILL CAUSE THE PROGRAM COUNTER IN THE CPU TO BE SET TO AN ADDRESS THAT DOES NOT ACTUALLY CONTAIN ANY MEMORY CELLS. IN THE SCELBI-8H, IF THE CPU ATTEMPTS TO READ INFORMATION FROM A NON-EXISTANT AREA OF MEMORY IT WILL RECEIVE THE CODE 377 (OCTAL) WHICH IT INTERPRETS AS A "HLT" INSTRUCTION. THUS, PERFORMING THIS PROCEDURE WHEN POWER IS INITIALLY APPLIED (OR WHENEVER IT IS DESIRED TO LOAD PROGRAMS OR PERFORM OTHER EXTENSIVE OPERATIONS UNDER MANUAL CONTROL) IS RECOMMENDED BECAUSE OF THE SPECIAL WAY IN WHICH THE SCELBI-8H INTERRUPT FACILITY OPERATES. THAT IS THE FACT THAT AN INSTRUCTION ISSUED TO THE COMPUTER IN THE INTERRUPT MODE ACTUALLY CAUSES THE INSTRUCTION TO BE "INSERTED" VIA THE CHASSIS TOGGLE SWITCHES DIRECTLY INTO THE CPU WHILE IT IS IN BETWEEN THE PROCESS OF EXECUTING INSTRUCTIONS FROM MEMORY. THUS, AS SOON AS THE INSTRUCTION THAT HAS BEEN INSERTED THROUGH THE INTERRUPT PROCESS HAS BEEN PERFORMED, THE CPU WILL RESUME OPERATIONS BY EXECUTING THE NEXT INSTRUCTION AT THE ADDRESS SPECIFIED BY THE CONTENTS OF THE PROGRAM COUNTER. THIS FEATURE OF HAVING THE "INTERRUPT" FACILITY ACTUALLY INSERT INSTRUCTIONS TO THE CPU IN BETWEEN THE CPU'S PROCESS OF EXECUTING INSTRUCTIONS FROM MEMORY IS EXTREMELY VALUABLE AT CERTAIN TIMES, SUCH AS WHEN IT IS DESIRED TO MANUALLY ALTER A PROGRAM'S OPERATION WITHOUT ACTUALLY PLACING A NEW INSTRUCTION IN MEMORY. BUT, IT CAN ALSO CAUSE DIFFICULTIES IF ONE IS NOT FULLY FAMILIAR WITH WHAT IS ACTUALLY OCCURRING WHEN ONE DESIRES TO PERFORM EXTENSIVE MANUAL OPERATIONS. IF PRECAUTIONS ARE NOT TAKEN AT SUCH TIMES, THE CPU MAY PERFORM INSTRUCTIONS FROM MEMORY (IN BETWEEN THE RECEIPT OF COMMANDS FROM THE INTERRUPT FACILITY) THAT COULD INTERFERE WITH THE MANUAL PROCEDURES. THIS POTENTIAL PROBLEM IS READILY AVOIDED IF THE PROGRAM COUNTER HAS BEEN SET TO AN ADDRESS THAT DOES NOT CONTAIN ANY ACTUAL MEMORY ELEMENTS BECAUSE THEN ANY COMMAND IT RECEIVES FROM SUCH A MEMORY ADDRESS WILL SIMPLY BE A "HLT" INSTRUCTION - WHICH WILL NOT INTERFERE WITH THE MANUAL PROCESS. THE PRECISE PROCEDURE TO ESTABLISH THIS DESIRED CONDITION WHEN POWER IS FIRST APPLIED TO THE COMPUTER, OR WHEN A PROGRAM IS BEING MANUALLY LOADED, OR WHEN OTHER TYPES OF EXTENSIVE MANUAL OPERATIONS ARE TO BE PERFORMED, IS SHOWN BELOW.

1. AFTER INITIALLY TURNING THE POWER ON, OR WHENEVER IT IS DESIRED TO DO EXTENSIVE OPERATIONS WITH THE CHASSIS SWITCHES - FIRST SET THE CHASSIS TOGGLE SWITCHES TO REPRESENT THE OCTAL CODE 1 Ø 4. (SWITCHES B6 AND B2 UP, ALL OTHERS DOWN.) 1 Ø 4 IS THE MACHINE LANGUAGE CODE FOR A "JMP" (JUMP) INSTRUCTION.

2. DEPRESS THE "INT" PUSH BUTTON SWITCH ON THE CHASSIS.

3. NOW DEPRESS THE "STEP" SWITCH ONE OR MORE TIMES UNTIL THE "INT" LIGHT ON THE FRONT PANEL CARD LIGHTS UP AND ACKNOWLEDGES RECEIPT OF THE INTERRUPT COMMAND. (THIS MAY REQUIRE SEVERAL OPERATIONS OF THE "STEP" BUTTON AS THE COMPUTER MIGHT BE IN THE PROCESS OF EXECUTING A MULTI-WORD INSTRUCTION AT THE TIME THE INTERRUPT COMMAND IS RECEIVED. THE CPU WILL FINISH PERFORMING THE ENTIRE INSTRUCTION IT WAS OPERATING ON BEFORE ACKNOWLEDGING THE INTERRUPT COMMAND.)

4. AFTER THE INTERRUPT LIGHT COMES ON DEPRESS THE STEP SWITCH ONCE MORE. THIS WILL CAUSE THE LEFT "STATUS" LAMP TO LIGHT INDICATING THAT THE COMPUTER IS READY TO ACCEPT THE NEXT WORD OF THE MULTI-WORD "JMP" INSTRUCTION THAT IS BEING INSERTED.

5. NOW CHANGE THE CHASSIS TOGGLE SWITCHES TO REPRESENT Ø77 OCTAL. (B7 AND B6 DOWN, B5 THROUGH BØ UP.)

6. DEPRESS THE STEP SWITCH ONE TIME. THIS WILL CAUSE THE CODE Ø77 TO BE RECEIVED FROM THE CHASSIS SWITCHES AS THE LOW ORDER ADDRESS PORTION OF THE "JMP" INSTRUCTION. THERE WILL BE NO CHANGES ON THE FRONT PANEL CARD LAMPS.

7. DEPRESS THE "STEP" SWITCH AGAIN. AT THIS TIME THE "INT" AND THE LEFT STATUS LAMP WILL EXTINGUISH SIGNIFYING THE COMPLETION OF THE INTERRUPT COMMAND INSTRUCTION. IN ADDITION IT CAN BE OBSERVED THAT THE MEMORY ADDRESS (MA) LAMPS ON THE TOP ROW OF THE FRONT PANEL CARD SHOW AN ADDRESS OF PAGE 77 LOCATION 077. IF THE OPERATOR WERE TO PRESS THE "STEP" BUTTON AGAIN (IT IS NOT NECESSARY TO DO SO BUT IT MAY BE DONE AS AN ILLUSTRATION) THE OPERATOR WOULD SEE ALL THE MEMORY CONTENTS (MC) LAMPS LIGHT AND THE "STOP" LAMP TURN ON INDICATING THAT THE COMPUTER TRIED TO OBTAIN AN INSTRUCTION FROM MEMORY (AFTER COMPLETION OF THE "JMP" INSTRUCTION ISSUED BY THE INTERRUPT COMMAND) AT THE ADDRESS PAGE 77 LOCATION 077. SINCE A BASIC SCELBI-8H DOES NOT HAVE THE 16,000 WORDS OF MEMORY NECESSARY TO UTILIZE SUCH A HIGH ADDRESS, THE MACHINE WOULD OBTAIN THE 377 "HLT" CODE FROM THAT ADDRESS WHICH IT INTERPRETS AS A HALT INSTRUCTION.

TYPICAL MANUAL OPERATIONS AFTER INITIALIZATION PROCEDURES

ONCE THE PROGRAM COUNTER HAS BEEN SET TO AN "OFF-MEMORY" ADDRESS NUMEROUS TYPES OF PROCEDURES CAN BE PERFORMED VIA THE CHASSIS TOGGLE SWITCHES - SUCH AS MANUALLY LOADING A PROGRAM INTO A SECTION OF MEMORY. THE BASIC PROCEDURE TO MANUALLY LOAD A PROGRAM INTO MEMORY IS DETAILED BELOW.

FIRST THE "H" REGISTER MUST BE SET TO THE PAGE IN MEMORY WHERE THE PROGRAM (OR PORTION OF A PROGRAM) IS TO RESIDE. NEXT THE "L" REGISTER IS SET TO THE LOCATION ON THE PAGE WHERE IT IS DESIRED TO START LOADING THE PROGRAM. THEN A LOAD MEMORY IMMEDIATE INSTRUCTION IS USED TO PLACE AN INSTRUCTION (OR DATA) INTO THE MEMORY LOCATION CURRENTLY SPECIFIED BY THE CONTENTS OF THE "H" AND "L" REGISTERS. THEN REGISTER "L" IS INCREMENTED BY ONE TO PREPARE FOR LOADING INFORMATION INTO THE NEXT SEQUENTIAL LOCATION IN MEMORY. THIS IS FOLLOWED BY ANOTHER "LMI" TYPE INSTRUCTION. THE SEQUENCE OF USING "LMI" INSTRUCTIONS AND THEN INCREMENTING REGISTER "L" TO POINT TO THE NEXT ADDRESS IN MEMORY IS USED UNTIL ONE HAS LOADED THE DESIRED PROGRAM. (SOMETIMES, IF A PROGRAM EXTENDS OVER MORE THAN ONE "PAGE" IT WILL BE NECESSARY TO ALSO INCREMENT REGISTER "H" AT SOME TIME DURING THE LOADING PROCEDURE.)

TO ILLUSTRATE THE PROCESS A SIMPLE ONE INSTRUCTION PROGRAM WILL BE DEMONSTRATED. THE INSTRUCTION "JUMP TO LOCATION 000 ON PAGE 00" WILL BE PLACED IN MEMORY BEGINNING AT LOCATION 000 ON PAGE 00. WHEN THIS INSTRUCTION IS EXECUTED THE COMPUTER WILL SIMPLY BE PLACED IN A "LOOP" AND WILL REPEATEDLY EXECUTE THE SAME INSTRUCTION. IT IS AN INTERESTING LITTLE ONE INSTRUCTION PROGRAM THAT CAN EVEN SERVE A PURPOSE BEYOND THAT OF A SIMPLE DEMONSTRATION - ITS OPERATION IS A QUICK CHECK ON THE COMPUTER'S GENERAL OPERABILITY!

THE "JMP" INSTRUCTION (REFER TO CHAPTER TWO WHEN NECESSARY) IS AN INSTRUCTION THAT REQUIRES THREE CONSECUTIVE WORDS IN MEMORY. THE FIRST WORD CONTAINS THE ACTUAL MACHINE CODE FOR THE JUMP COMMAND. THE NEXT WORD MUST THEN CONTAIN THE "LOW ADDRESS" (LOCATION ON A PAGE) OF WHERE THE PROGRAM IS TO GO. THE THIRD WORD CONTAINS THE PAGE NUMBER OF THE MEMORY LOCATION WHERE THE MACHINE IS TO GET ITS NEXT INSTRUCTION. (REMEMBER THAT "JMP" AND "CAL" INSTRUCTIONS ACTUALLY LOAD THE ADDRESS WORDS INTO THE CPU'S PROGRAM COUNTER TO AFFECT THE ADDRESSING OPERATION OF THE MACHINE.)

THE OCTAL MACHINE CODE FOR THE "JMP" INSTRUCTION IS 1 0 4. THE FULL THREE WORD INSTRUCTION TO JUMP TO LOCATION 000 ON PAGE 00 WOULD APPEAR

IN THREE CONSECUTIVE WORDS AS SHOWN IN FIGURE 6.

```
1 0 4

0 0 0

0 0 0
```

FIGURE 6

WHILE THIS PARTICULAR INSTRUCTION COULD HAVE BEEN PLACED AT ANY LOCATION IN MEMORY - IN ORDER TO HAVE THE COMPUTER OPERATE ON IT AND FORM A "LOOP TO ITSELF" IT IS NECESSARY TO PLACE IT IN THE SPECIFIC MEMORY ADDRESS LOCATION STARTING AT LOCATION 000 ON PAGE 00 AS IS ILLUSTRATED IN FIGURE 7.

ADDRESS		MEMORY
PAGE	LOCATION	CONTENTS
00	000	104
00	001	000
00	002	000

FIGURF 7

IN ORDER TO START LOADING THE INSTRUCTION INTO MEMORY IT WILL FIRST BE NECESSARY TO SET CPU REGISTER "H" TO 000 BY USING THE INTERRUPT FACILITY TO INSERT THE INSTRUCTION "LOAD REGISTER H IMMEDIATE WITH 000" (LHI 000) AS DETAILED BELOW.

1. SET THE CHASSIS TOGGLE SWITCHES TO 056 (OCTAL). NOW DEPRESS THE "STEP" BUTTON UNTIL THE "INT" LIGHT ON THE FRONT PANEL CARD COMES ON.

2. PRESS THE "STEP" BUTTON ONCE MORE. THE LEFT STATUS LAMP LIGHT WILL TURN ON INDICATING THAT THE COMPUTER IS READY FOR THE NEXT WORD OF THE TWO WORD INSTRUCTION.

3. NOW SET THE CHASSIS TOGGLE SWITCHES TO 000.

4. PRESS THE "STEP" BUTTON AGAIN. THE "INT" AND LEFT STATUS LAMP WILL EXTINGUISH. REGISTER "H" IN THE CPU HAS NOW BEEN SET TO THE OCTAL VALUE 000.

NEXT REGISTER "L" MUST BE SET TO 000 IN ORDER TO SPECIFY THE LOCATION ON THE MEMORY PAGE. THIS IS ACCOMPLISHED IN A SIMILAR MANNER BY INSERTING A "LOAD REGISTER L IMMEDIATE WITH 000" (LMI 000) INSTRUCTION VIA. THE INTERRUPT MODE. THE PROCEDURE IS EXACTLY THE SAME AS THAT FOR THE "LHI 000" INSTRUCTION ILLUSTRATED ABOVE EXCEPT THAT THE CHASSIS SWITCHES ARE SET TO 066 - THE OCTAL MACHINE CODE FOR AN "LLI" COMMAND - IN STEP NUMBER 1 OF THE ABOVE PROCEDURE.

CPU REGISTERS "H" AND "L" WILL NOW BE SET TO POINT TO THE DESIRED LOCATION IN MEMORY WHERE THE FIRST WORD OF THE "JMP" INSTRUCTION IS TO BE PLACED. IT IS NOW AN EASY MATTER TO USE A "LOAD MEMORY IMMEDIATE" TYPE INSTRUCTION VIA THE INTERRUPT MODE TO ACTUALLY LOAD THE OCTAL CODE 1 Ø 4 (THE FIRST WORD FOR A "JMP" INSTRUCTION) INTO THE COMPUTER'S MEMORY.

1. SET THE CHASSIS TOGGLE SWITCHES TO Ø76.

2. DEPRESS THE "INT" PUSH BUTTON SWITCH AND THEN ADVANCE THE PROCESSOR WITH THE "STEP" BUTTON UNTIL THE "INT" LAMP LIGHTS.

3. PUSH THE "STEP" BUTTON ONCE MORE. THE LEFT STATUS LAMP WILL LIGHT. THE COMPUTER IS NOW READY TO ACCEPT THE "IMMEDIATE" PART OF THE "LMI" INSTRUCTION.

4. SET THE TOGGLE SWITCHES TO 1 Ø 4.

5. DEPRESS THE "STEP" SWITCH. AT THIS TIME THE ADDRESS LAMPS WILL SHOW AN ADDRESS OF PAGE ØØ LOCATION ØØØ (THUS VERIFYING THAT THE "H" AND "L" REGISTERS HAVE BEEN SET TO THE PROPER ADDRESS.) BOTH STATUS LAMPS WILL NOW BE LIT INDICATING THAT THE MACHINE IS READY TO WRITE A WORD INTO MEMORY (AT THE ADDRESS SHOWN BY THE ADDRESS LAMPS.)

6. DEPRESS THE "STEP" SWITCH ONE MORE TIME. AT THIS POINT THE MEMORY CONTENTS WILL SHOW 1 Ø 4 THUS VERIFYING THAT THE CODE 1 Ø 4 WAS SENT TO THE COMPUTER'S MEMORY. (NOTE. AT THIS POINT THE MEMORY ADDRESS LAMPS WILL NO LONGER SHOW THE ADDRESS OF THE LOCATION THAT WAS WRITTEN INTO - THEIR INDICATIONS MAY BE IGNORED AT THIS STEP.) ALSO, AT THIS TIME THE "INT" LAMP AND THE STATUS LAMPS WILL EXTINGUISH INDICATING THAT THE INSTRUCTION HAS BEEN COMPLETED.

IN ORDER TO LOAD THE NEXT WORD IN MEMORY IT IS NECESSARY TO ADVANCE THE "L" REGISTER SO THAT IT POINTS TO LOCATION ØØ1 ON THE CURRENT PAGE (WHICH IS STILL PAGE ØØ.) TO DO THIS AN "INCREMENT CPU REGISTER L" INSTRUCTION IS GIVEN VIA THE INTERRUPT PROCESS.

1. SET THE CHASSIS SWITCHES TO THE CODE FOR AN "INL" INSTRUCTION WHICH IS Ø 6 Ø.

2. DEPRESS THE "INT" BUTTON AND THEN STEP THE COMPUTER UNTIL THE "INT" LIGHT COMES ON.

3. DEPRESS THE "STEP" SWITCH ONCE MORE. THE "INT" LIGHT WILL EXTINGUISH INDICATING THAT THE COMMAND HAS BEEN EXECUTED. (NOTE: SINCE THE "INL" INSTRUCTION IS A SIMPLE ONE WORD INSTRUCTION NEITHER ONE OF THE "STATUS" LAMPS WILL LIGHT DURING THE PROCESS.)

NOW THE CODE Ø Ø Ø (FOR THE SECOND WORD, I.E., THE LOW ADDRESS PORTION OF THE "JMP" INSTRUCTION) NEEDS TO BE LOADED INTO MEMORY. THE "LMI ØØØ" INSTRUCTION IS USED. THE PROCEDURE FOR AN "LMI" INSTRUCTION WAS DETAILED ABOVE AND THE ONLY CHANGE NECESSARY IN THE ABOVE PROCEDURE FOR INSERTING THE "LMI" COMMAND IS TO CHANGE STEP NUMBER FOUR OF THE ILLUSTRATION. AT STEP NUMBER FOUR THE CHASSIS SWITCHES MUST NOW BE SET TO THE OCTAL CODE Ø Ø Ø. NOTE NOW AT STEP NUMBER FIVE THAT THE ADDRESS

LAMPS WILL INDICATE PAGE 00 AT LOCATION 001. ALSO AT STEP NUMBER SIX THE MEMORY CONTENTS (MC) LAMPS WILL SHOW 0 0 0 TO REFLECT THE NEW INFORMATION LOADED INTO THE WORD IN MEMORY.

NOW TO LOAD THE NEXT WORD AND THUS COMPLETE THE "JMP" INSTRUCTION (AS WORD NUMBER THREE CONTAINS THE PAGE ADDRESS FOR THE "JMP" INSTRUCTION) IT IS AGAIN NECESSARY TO ADVANCE THE "L" REGISTER TO POINT TO LOCATION 002 ON THE CURRENT PAGE. THE SEQUENCE FOR INCREMENTING THE "L" REGISTER ("INL") IS PERFORMED EXACTLY AS PREVIOUSLY DETAILED.

FINALLY ANOTHER "LMI 000" INSTRUCTION IS ISSUED AS DESCRIBED ABOVE TO LOAD 0 0 0 INTO WORD NUMBER 002 ON PAGE 00.

AT THIS TIME THE COMPUTER'S MEMORY SHOULD CONTAIN THE THREE WORD "JUMP TO LOCATION 000 ON PAGE 00" INSTRUCTION STARTING AT LOCATION 000 ON PAGE 00. THE INSTRUCTION WAS LOADED UTILIZING THE INTERRUPT FACILITY TO MANUALLY LOAD EACH WORD. THE VARIOUS LAMPS ON THE FRONT PANEL CARD WERE USED TO MONITOR THE LOADING PROCESS. THE PROCESS OF USING THE INTERRUPT FACILITY TO ACTUALLY LOAD THE PROGRAM INTO THE COMPUTER'S MEMORY IS A DISTINCT AND SEPARATE PROCESS FROM THAT OF HAVING THE COMPUTER EXECUTE A PROGRAM THAT RESIDES IN MEMORY. HOWEVER, IF THE READER HAS PERFORMED THE LOADING OPERATIONS JUST DESCRIBED THEN THE READER CAN PROCEED TO HAVE THE COMPUTER ACTUALLY OPERATE AND PERFORM THE INSTRUCTION IN THE NORMAL "PROGRAMMED OPERATION" MODE. WHEN THIS IS DONE THE COMPUTER WILL START TO READ INSTRUCTIONS IN MEMORY AND AUTOMATICALLY PERFORM THE OPERATIONS DICTATED BY THE PROGRAM.

TO HAVE THE MACHINE START EXECUTING THE PROGRAM JUST LOADED IS AN EASY MATTER. NOW THE INTERRUPT FACILITY IS SIMPLY USED TO INSERT A "JUMP TO LOCATION 000 ON PAGE 00 TO THE CPU. (NOTE THAT NOW AN ACTUAL "JMP" INSTRUCTION IS BEING DIRECTED TO THE CPU. IT IS NOT BEING LOADED INTO MEMORY.) THIS COMMAND WILL RESULT IN THE PROGRAM COUNTER BEING SET TO LOCATION 000 ON PAGE 00. REMEMBER THAT AT THE BEGINNING OF THE LOADING PROCESS THE USER WAS DIRECTED TO USE A "JMP" INSTRUCTION TO SET THE PROGRAM COUNTER TO AN ADDRESS THAT DID NOT CONTAIN MEMORY CELLS. NOW THE PROGRAM COUNTER WILL BE SET BACK TO AN AREA WHERE MEMORY CELLS ARE PRESENT. WHEN THE COMPUTER IS NOT IN THE INTERRUPT MODE IT WILL AUTOMATICALLY PROCESS INSTRUCTIONS FROM MEMORY AT THE LOCATION SPECIFIED BY THE CONTENTS OF THE PROGRAM COUNTER. HENCE, AT THE COMPLETION OF THE INTERRUPT CYCLE ABOUT TO BE ENTERED THE COMPUTER WILL BE READY TO START EXECUTING THE PROGRAM AT LOCATION 000 ON PAGE 00.

TO INSERT THE JUMP TO LOCATION 000 ON PAGE 00 INSTRUCTION VIA THE INTERRUPT MODE THE USER SIMPLY REPEATS THE PROCEDURE ILLUSTRATED SEVERAL PAGES EARLIER WHEN THE INTERRUPT FACILITY WAS USED TO INSERT THE DIRECTIVE TO JUMP TO LOCATION 077 ON PAGE 77 WITH THE FOLLOWING APPROPRIATE CHANGES:

A. THE CHASSIS TOGGLE SWITCHES SHOULD BE SET TO THE ADDRESS 0 0 0 FOR STEP NUMBER FIVE.

B. AT THE COMPLETION OF STEP NUMBER SEVEN THE MEMORY ADDRESS (MA) LAMPS WILL DENOTE THE ADDRESS PAGE 00 LOCATION 000.

NOW THE "STEP" BUTTON MAY BE USED TO EXECUTE THE STORED PROGRAM ONE WORD AT TIME.

NOW, WHEN THE COMPUTER IS EXECUTING A PROGRAM IN MEMORY THE ADDRESS LAMPS WILL SHOW THE ADDRESS OF THE NEXT LOCATION IN MEMORY THAT WILL BE ACCESSED BY THE COMPUTER (EXCEPT FOR SEVERAL SPECIAL CASES WHICH ARE EXPLAINED LATER.) THE MEMORY CONTENTS LAMPS WILL SHOW THE CONTENTS OF

THE MEMORY LOCATION JUST PREVIOUSLY ACCESSED (AGAIN EXCEPT FOR A FEW SPECIAL CASES WHICH ARE COVERED LATER.) THIS DISPLAY ARRANGEMENT ALLOWS THE USER TO CHECK THE OPERATION OF A PROGRAM ON A STEP-BY-STEP BASIS.

IN ADDITION TO THE INFORMATION PROVIDED BY THE MEMORY ADDRESS (MA) AND MEMORY CONTENTS (MC) LAMPS, THE STATUS LAMPS WILL PROVIDE THE OPERATOR WITH ADDITIONAL INFORMATION AS TO WHAT TYPE OF OPERATION IS BEING PERFORMED DURING THE EXECUTION OF INSTRUCTIONS FROM MEMORY (AS THEY DO WHEN THE MACHINE IS IN THE INTERRUPT MODE.)

WITH THIS INFORMATION THE READER CAN PROCEED TO ADVANCE THE COMPUTER THROUGH THE TINY "JUMP TO ITSELF" PROGRAM PREVIOUSLY DESCRIBED IF THE USER HAS FOLLOWED THE DIRECTIONS AND PLACED THE PROGRAM IN MEMORY, AND HAS ALSO SET THE PROGRAM COUNTER TO LOCATION 000 ON PAGE 00 WITH A "JMP" COMMAND.

IF THE "STEP" BUTTON IS DEPRESSED NOW, THE MEMORY ADDRESS LAMPS WILL SHOW PAGE 00 LOCATION 001. THE MEMORY CONTENTS LAMPS SHOULD DISPLAY 104 - WHICH IS THE CONTENTS OF MEMORY ADDRESS PAGE 00 LOCATION 000.

PRESSING THE "STEP" BUTTON AGAIN WILL RESULT IN THE MA LAMPS CHANGING TO THE ADDRESS PAGE 00 LOCATION 002. THE MC LAMPS WILL THEN SHOW 000 (THE CONTENTS OF THE MEMORY ADDRESS THAT IS ONE LESS THAN THAT SHOWN BY THE MA LAMPS.

AND PRESSING THE "STEP" BUTTON ONCE MORE WILL RESULT IN THE MA LAMPS "JUMPING" TO AN ADDRESS OF PAGE 00 LOCATION 000 TO REFLECT THE EXECUTION OF THE "JMP" INSTRUCTION. THE MC LAMPS WILL SHOW 000 WHICH IS THE CONTENTS OF THE LAST MEMORY LOCATION ACCESSED - WHICH WAS AT PAGE 00 LOCATION 002.

THE USER COULD CONTINUE THROUGH THE TINY PROGRAM IN THE STEP MODE AS LONG AS DESIRED. THE COMPUTER WILL CONTINUE TO PERFORM THE "JUMP TO ITSELF" PROGRAM REPEATEDLY. HOWEVER, AT THIS TIME, PROVIDED THAT THE PROGRAM OPERATED CORRECTLY IN THE STEP MODE, THE COMPUTER CAN BE PLACED IN THE "RUN" MODE BY SIMPLY DEPRESSING THE "RUN" PUSH BUTTON SWITCH ON THE CHASSIS. WHEN THIS IS DONE THE "RUN" LAMP ON THE FRONT PANEL CARD WILL TURN ON. THE COMPUTER WILL NOW PERFORM THE PROGRAM AUTOMATICALLY AT A RATE OF MANY THOUSANDS OF TIMES PER SECOND. ALL THE FRONT PANEL LAMPS WILL CONTINUE TO OPERATE BUT SINCE THEY ARE FLASHING AT THOUSANDS OF TIMES PER SECOND THE OPERATOR CAN NOT DISCERN INDIVIDUAL INSTRUCTIONS. HOWEVER, IT IS OFTEN STILL POSSIBLE TO OBSERVE WHAT GENERAL AREA OF MEMORY THE COMPUTER IS OPERATING IN BY OBSERVING THE MA LAMPS WHICH WILL TEND TO GLOW STRONGEST AT THE ADDRESSES WHERE THE COMPUTER IS PERFORMING THE MOST INSTRUCTIONS.

THE "JUMP TO ITSELF" PROGRAM JUST DESCRIBED HAS LITTLE VALUE BEYOND SERVING AS A DEMONSTRATION PROGRAM FOR THE USER TO USE TO BECOME ACQUAINTED WITH THE OPERATION OF A SCELBI-8H MINI-COMPUTER - OR TO SERVE AS A SIMPLE TEST OF THE COMPUTERS FUNCTIONAL INTEGRITY. WHILE IT WILL UNDOUBTABLY TAKE THE OPERATOR SEVERAL MINUTES TO CAREFULLY SET UP AND EXECUTE THE PROGRAM THE FIRST TIME AROUND, IT SHOULD BE POINTED OUT THAT WITH A LITTLE PRACTICE THE OPERATOR WILL FIND THAT ONE IS ABLE TO LOAD AND EXECUTE A TINY PROGRAM OF THIS SIZE IN JUST A FEW SECONDS. IN OPERATING A COMPUTER, EXPERIENCE IS THE BEST TEACHER. WITH A SCELBI-8H THE LEARNING PROCESS CAN BE A LOT OF FUN!

PROCEDURE FOR INSERTING ANY TYPE OF INSTRUCTION VIA THE INTERRUPT FACILITY

THE READER HAS ALREADY BEEN SHOWN HOW TO OPERATE THE SCELBI-8H MINI-COMPUTER IN THE INTERRUPT MODE FOR SOME OF THE MOST OFTEN USED TYPES OF INSTRUCTIONS. PARTICULARLY THOSE NECESSARY WHEN MANUALLY LOADING A PROGRAM INTO MEMORY. HOWEVER, EACH AND EVERY TYPE OF INSTRUCTION THAT THE SCELBI-8H CAN PERFORM (REFER TO CHAPTER 2) MAY BE COMMANDED IN THE INTERRUPT MODE. THE FOLLOWING TABLE DETAILS THE STEP-BY-STEP PROCEDURE FOR EACH CLASS OF INSTRUCTION AND ALSO PRESENTS THE INFORMATION THAT WILL BE DISPLAYED BY THE LIGHTS ON THE FRONT PANEL CARD AS EACH STEP IS PERFORMED. THE ORDER OF PRESENTATION WILL BE SIMILAR TO THAT USED TO PRESENT THE INSTRUCTION SET IN CHAPTER 2 SO THAT INFORMATION IN THE CHAPTER MAY BE REFERENCED TO OBTAIN THE SPECIFIC MACHINE LANGUAGE CODES FOR THE VARIOUS TYPES OF INSTRUCTIONS.

THE USER IS REMINDED THAT THE INTERRUPT MODE CAN BE USED TO "INSERT" AN INSTRUCTION INTO A SEQUENCE OF INSTRUCTIONS THAT IS BEING EXECUTED IM MEMORY WITHOUT ACTUALLY CAUSING THE INSTRUCTION TO BE PLACED IN MEMORY. OR, AS ILLUSTRATED EARLIER, BY USING APPROPRIATE COMBINATIONS OF INSTRUCTIONS, THE MODE MAY BE USED TO LOAD INSTRUCTIONS OR DATA INTO MEMORY. OR, THE MODE MAY BE USED TO PERFORM A HOST OF OPERATIONS COMPLETELY INDEPENDENT OF MEMORY. THE USER IS ALSO REMINDED OF THE SUGGESTION TO SET THE PROGRAM COUNTER TO AN ADDRESS OUTSIDE THE RANGE OF PHYSICAL MEMORY FOR THE USER'S SYSTEM WHENEVER IT IS DESIRED TO PERFORM EXTENSIVE OPERATIONS IN THE INTERRUPT MODE. THIS WILL PREVENT THE COMPUTER FROM CONTINUALLY SWITCHING BACK TO "PROGRAMMED OPERATION" DURING EXTENSIVE INTERRUPT OPERATIONS.

IN THE FOLLOWING TABLE IT IS ASSUMED THAT THE OPERATOR WILL FIRST SET THE CHASSIS SWITCHES TO THE SPECIFIC MACHINE LANGUAGE CODE FOR THE PARTICULAR KIND OF INSTRUCTION DESIRED. NEXT THE OPERATOR WILL PRESS AND RELEASE THE "INT" PUSH BUTTON, AND THEN USE THE "STEP" SWITCH TO STEP THE COMPUTER UNTIL THE "INT" LAMP ON THE FRONT PANEL CARD TURNS ON. THE TABLE THEN LISTS THE ACTION(S) NECESSARY TO COMPLETE THE INSTRUCTION AND DEFINES THE MEANINGS OF THE INDICATORS ON THE FRONT PANEL CARD FOR EACH STEP.

SEVERAL ABBREVIATIONS WILL BE USED IN THE TABLE. THESE INCLUDE:

NRI	=	NOT RELATED TO THE CURRENT INTERRUPT PROCESS.
HA	=	HIGH ADDRESS (PAGE)
LA	=	LOW ADDRESS (LOCATION ON A PAGE)
H & L	=	REFERS TO THE ADDRESS CONTAINED IN CPU REGISTERS "H" & "L"

IN ADDITION THE WORD "CODE" IN THE CHASSIS SWITCH POSITION COLUMN MEANS THAT THE CHASSIS TOGGLE SWITCHES SHOULD BE SET TO THE OCTAL MACHINE LANGUAGE CODE FOR THE TYPE OF INSTRUCTION BEING EXECUTED AND THE WORD "DATA" INDICATES THAT THE CHASSIS TOGGLE SWITCHES SHOULD BE SET TO THE DESIRED DATA.

TABLE 1

STEP #	CHASSIS SWITCHES	STATUS LAMPS	MEMORY ADDRESS LAMPS	MEMORY CONTENTS LAMPS
LOAD DATA FROM ONE CPU REGISTER TO ANOTHER CPU REGISTER (LAA, LBA...)				
1	CODE	0 0	NRI	NRI
LOAD DATA FROM A CPU REGISTER TO A LOCATION IN MEMORY				
1	CODE	* *	H & L	NRI
2	XXX	0 0	NRI	CONTENTS OF THE CPU REGISTER
LOAD DATA FROM A LOCATION IN MEMORY TO A CPU REGISTER				
1	CODE	* 0	H & L	NRI
2	XXX	0 0	NRI	CONTENTS OF THE MEMORY LOCATION
LOAD "IMMEDIATE" DATA INTO A CPU REGISTER				
1	CODE	* 0	NRI	NRI
2	DATA	0 0	NRI	NRI
LOAD IMMEDIATE DATA INTO A MEMORY LOCATION				
1	CODE	* 0	NRI	NRI
2	DATA	* *	H & L	NRI
3	XXX	0 0	NRI	DATA LOADED

TABLE 1

STEP #	CHASSIS SWITCHES	STATUS LAMPS	MEMORY ADDRESS LAMPS	MEMORY CONTENTS LAMPS
INCREMENT OR DECREMENT THE VALUE OF A CPU REGISTER				
1	CODE	0 0	NRI	NRI
ARITHMETIC INSTRUCTIONS BETWEEN ACCUMULATOR AND A CPU REGISTER				
1	CODE	0 0	NRI	NRI
COMPARE INSTRUCTIONS BETWEEN ACCUMULATOR AND A CPU REGISTER				
1	CODE	0 0	NRI	NRI
ARITHMETIC AND COMPARE INSTRUCTIONS THAT UTILIZE A WORD IN MEMORY AS AN OPERAND				
1	CODE	* 0	H & L	NRI
2	XXX	0 0	NRI	CONTENTS OF THE WORD IN MEMORY
"IMMEDIATE" TYPE ARITHMETIC AND COMPARE INSTRUCTIONS				
1	CODE	* 0	NRI	NRI
2	DATA	0 0	NRI	NRI
BOOLEAN MATH OPERATIONS BETWEEN THE ACCUMULATOR AND A CPU REGISTER				
1	CODE	0 0	NRI	NRI

TABLE 1

STEP #	CHASSIS SWITCHES	STATUS LAMPS	MEMORY ADDRESS LAMPS	MEMORY CONTENTS LAMPS
BOOLEAN MATH OPERATIONS BETWEEN THE ACCUMULATOR AND A WORD IN MEMORY				
1	CODE	* 0	H & L	NRI
2	XXX	0 0	NRI	CONTENTS OF THE WORD IN MEMORY
BOOLEAN "IMMEDIATE" INSTRUCTIONS				
1	CODE	* 0	NRI	NRI
2	DATA	0 0	NRI	NRI
ACCUMULATOR ROTATE INSTRUCTIONS				
1	CODE	0 0	NRI	NRI
UNCONDITIONAL JUMP OR CALL INSTRUCTIONS				
1	CODE	* 0	NRI	NRI
2	LA	* 0	NRI	NRI
3	HA	0 0	HA & LA	NRI
CONDITIONAL JUMP OR CALL INSTRUCTIONS				
1	CODE	* 0	NRI	NRI
2	LA	* 0	NRI	NRI
3	HA	0 0	HA & LA ONLY IF CONDX SATISFIED NRI OTHERWISE	NRI

TABLE 1

STEP #	CHASSIS SWITCHES	STATUS LAMPS	MEMORY ADDRESS LAMPS	MEMORY CONTENTS LAMPS
			UNCONDITIONAL RETURN INSTRUCTION	
1	CODE	0 0	NEW VALUE OF PROGRAM COUNTER	NRI
			CONDITIONAL RETURN INSTRUCTION	
1	CODE	0 0	NEW VALUE OF PROGRAM COUNTER IF CONDX SATISFIED NRI OTHERWISE	NRI
			RESTART INSTRUCTION	
1	CODE	0 0	HA & LA	NRI
			OUTPUT INSTRUCTION	
1	CODE	0 *	PAGE LAMPS = PORT SELECTED MA LAMPS = DATA OUTPUTTED	NRI
2	XXX	0 0	NRI	NRI
			INPUT INSTRUCTION	
1	CODE	0 *	PAGE LAMPS = PORT SELECTED MA LAMPS = NRI	NRI
2	XXX	0 0	NRI	NRI
			HALT INSTRUCTION	
1	CODE	0 0	NRI	NRI

** NOTE: "STOP" LAMP WILL TURN ON **

INFORMATION PRESENTED BY THE FRONT PANEL CARD LIGHTS DURING STEPPED PROGRAM EXECUTION.

THE NEXT TABLE PRESENTS THE TYPE OF INFORMATION THAT CAN BE OBTAINED FROM OBSERVING THE FRONT PANEL CARD LAMPS WHEN THE VARIOUS CLASSES OF INSTRUCTIONS ARE BEING EXECUTED VIA A PROGRAM IN MEMORY ONE STEP AT A TIME USING THE CHASSIS "STEP" BUTTON.

AGAIN, THE TABLE IS ORGANIZED TO PRESENT THE VARIOUS TYPES OF INSTRUCTIONS IN THE ORDER GIVEN IN CHAPTER 2. APPLICABLE ABBREVIATIONS ARE THE SAME AS USED FOR THE PREVIOUS TABLE WITH THE ADDITION OF THE WORD "ADDR + 1" WHICH IS AN ABBREVIATION TO INDICATE THE "ADDRESS WHERE THE NEXT INSTRUCTION WILL BE OBTAINED BY THE CPU."

* * * * * * * * * * *

TABLE 2

TYPE OF INSTRUCTION	STEP #	STATUS LAMPS	MA LAMPS	MC LAMPS
LOAD DATA FROM ONE CPU REGISTER TO ANOTHER CPU REGISTER.	1	0 0	ADDR + 1	CODE
LOAD DATA FROM A CPU REGISTER TO A LOCATION IN MEMORY.	1	* *	H & L	CODE
	2	0 0	ADDR + 1	DATA LOADED INTO MEMORY
LOAD DATA FROM A LOCATION IN MEMORY TO A CPU REGISTER.	1	* 0	H & L	CODE
	2	0 0	ADDR + 1	DATA LOADED FROM MEMORY
LOAD "IMMEDIATE" DATA INTO A CPU REGISTER.	1	* 0	ADDR + 1	CODE
	2	0 0	ADDR + 1	DATA

TABLE 2

TYPE OF INSTRUCTION	STEP #	STATUS LAMPS	MA LAMPS	MC LAMPS
LOAD "IMMEDIATE" DATA INTO A LOCATION IN MEMORY.	1	* 0	ADDR + 1	CODE
	2	* *	H & L	DATA
	3	0 0	ADDR + 1	DATA LOADED INTO MEMORY
INCREMENT OR DECREMENT A CPU REGISTER.	1	0 0	ADDR + 1	CODE
ARITHMETIC INSTRUCTION BETWEEN THE ACCUMULATOR AND A CPU REGISTER.	1	0 0	ADDR + 1	CODE
COMPARE BETWEEN THE ACCUMULATOR AND A CPU REGISTER.	1	0 0	ADDR + 1	CODE
ARITHMETIC OR COMPARE INSTRUCTION BETWEEN THE ACCUMULATOR AND A WORD IN MEMORY.	1	* 0	H & L	CODE
	2	0 0	ADDR + 1	CONTENTS OF THE REFERRED WORD IN MEMORY
"IMMEDIATE" TYPE ARITHMETIC AND COMPARE INSTRUCTIONS.	1	* 0	ADDR + 1	CODE
	2	0 0	ADDR + 1	DATA

TABLE 2

TYPE OF INSTRUCTION	STEP #	STATUS LAMPS	MA LAMPS	MC LAMPS
BOOLEAN MATH OPERATIONS BETWEEN ACCUMULATOR AND CPU REGISTERS.	1	0 0	ADDR + 1	CODE
BOOLEAN MATH OPERATIONS BETWEEN ACCUMULATOR AND A LOCATION IN MEMORY.	1	* 0	H & L	CODE
	1	0 0	ADDR + 1	CONTENTS OF THE REFERRED WORD IN MEMORY
BOOLEAN "IMMEDIATE" INSTRUCTIONS.	1	* 0	ADDR + 1	CODE
	2	0 0	ADDR + 1	DATA
ACCUMULATOR ROTATE INSTRUCTIONS.	1	0 0	ADDR + 1	CODE
UNCONDITIONAL JUMP OR CALL INSTRUCTIONS.	1	* 0	ADDR + 1	CODE
	2	* 0	ADDR + 1	LA
	3	0 0	NEW ADDR	HA
CONDITIONAL JUMP OR CALL INSTRUCTIONS.	1	* 0	ADDR + 1	CODE
	2	* 0	ADDR + 1	LA
	3	0 0	NEW ADDR IF CONDX SATISFIED OTHERWISE ADDR + 1	HA

TABLE 2

TYPE OF INSTRUCTION	STEP #	STATUS LAMPS	MA LAMPS	MC LAMPS
UNCONDITIONAL RETURN INSTRUCTION.	1	0 0	NEW ADDR	CODE
CONDITIONAL RETURN INSTRUCTION.	1	0 0	NEW ADDR IF CONDX SATISFIED OTHERWISE ADDR + 1	CODE
RESTART INSTRUCTION.	1	0 0	NEW ADDR	CODE
OUTPUT INSTRUCTION.	1	0 *	SEE NOTE #1	CODE
	2	0 0	ADDR + 1	IGNORE
INPUT INSTRUCTION.	1	0 *	SEE NOTE #2	CODE
	2	0 0	ADDR + 1	IGNORE
HALT INSTRUCTION.	1	0 0	ADDR	CODE

NOTE #1: PAGE LAMPS SHOW OUPUT PORT SELECTED. MEMORY ADDRESS LAMPS SHOW DATA OUTPUTTED.

NOTE #2: PAGE LAMPS SHOW INPUT PORT SELECTED. MEMORY ADDRESS LAMPS SHOW ACCUMULATOR CONTENTS PRIOR TO RECEIVING NEW INPUT.

** NOTE: "STOP" LAMP WILL TURN ON **

THE INFORMATION PRESENTED IN TABLES 1 AND 2 WILL ALLOW THE USER TO INTERPRET THE LAMPS AND TO HAVE COMPLETE CONTROL OVER THE MANUAL OPERATION OF A SCELBI-8H MINI-COMPUTER. THE DATA IN THE TABLES MIGHT APPEAR COMPLICATED AT FIRST GLANCE, HOWEVER, WITH A LITTLE "HANDS-ON" EXPERIENCE A PERSON QUICKLY BECOMES FAMILIAR WITH THE LIGHTS AND SWITCHES AND

PROCEED TO LOAD, EXAMINE AND "DEBUG," AND THEN EXECUTE SOPHISTICATED PROGRAMS IN A SHORT AMOUNT OF TIME. UNTIL ONE IS FULLY AT EASE WITH THE SCELBI-8H CONTROLS AND INDICATORS, THE TABLES JUST PRESEFTED WILL SERVE AS A VALUABLE AFD CGEPACT REFEREFCE&

SOME SAMPLE PROGRAMS AND DEMONSTRATIONS OF PROGRAMMING METHODS

AT THIS POINT IT MIGHT SERVE AS A VALUABLE EXERCISE FOR THE USER TO APPLY WHAT HAS BEEN PRESENTED BY LOADING A TRULY PRACTICAL PROGRAM INTO MEMORY AND THEN HAVING THE SCELBI-8H MINI-COMPUTER EXECUTE THE PROGRAM. THE PROGRAM THAT IS PRESENTED NEXT IS ONE THAT WILL PERFORM A VERY BASIC TEST ON A SECTION OF THE COMPUTER'S MEMORY BANKS. THE PROGRAM PERFORMS THE FOLLOWING OPERATIONS.

FIRST IT WILL WRITE ZEROS INTO EVERY BIT POSITION OF A WORD IN MEMORY. THEN IT WILL READ THE CONTENTS OF THE SAME WORD AND CHECK TO SEE THAT THE WORD DOES INDEED CONTAIN A ZERO IN EVERY BIT POSITION. IF THE CHECK SHOULD FIND ANY ONES IN THE WORD THEN THE COMPUTER WILL HALT IN A MANNER THAT CAUSES ALL THE MEMORY CONTENTS LAMPS TO BE TURNED ON AS AN INDICATOR TO THE OPERATOR THAN AN ERROR HAS BEEN NOTED. HOWEVER, IF THE CHECK IS CORRECT THEN THE PROGRAM WILL PROCEED TO WRITE ALL ONES INTO EVERY BIT POSITION OF THE SAME WORD AND THEN AGAIN READ THE CONTENTS OF THE WORD. NOW THE MEMORY ELEMENTS IN THE WORD SHOULD ALL CONTAIN A LOGIC "1." IF THIS TEST IS PASSED THE PROGRAM WILL GO TO THE ADDRESS OF THE NEXT WORD IN MEMORY AND REPEAT THE TEST PROCEDURE. (IF THE TEST SHOULD FAIL THE PROGRAM WILL HALT WITH THE MEMORY CONTENTS LAMPS TURNED ON.) THIS PROCESS WILL CONTINUE UNTIL EVERY WORD ON A "PAGE" IN MEMORY HAS BEEN TESTED. SINCE THE COMPUTER OPERATES SO FAST THAT THE TIME TO PERFORM THE TEST ON ALL 256 (DECIMAL) WORDS IN A PAGE OF MEMORY REQUIRES ONLY A FRACTION OF A SECOND, WHICH IS BARELY ENOUGH TIME TO DETECT THAT THE PROGRAM IS OPERATIONAL, A PROGRAM "LOOP" HAS BEEN ADDED TO THE BASIC PROGRAM SO THAT THE COMPUTER WILL TEST EVERY WORD ON THE PAGE A LARGE NUMBER OF TIMES BEFORE ENDING. TO DO THIS, A CPU REGISTER IS SET UP TO SERVE AS A "LOOP COUNTER." EVERY TIME THE PROGRAM FINISHES TESTING ALL THE LOCATIONS ON A "PAGE" IN MEMORY THE PROGRAM WILL "DECREMENT" THE VALUE OF THE "LOOP COUNTER" AND THEN AN INSTRUCTION IS USED TO TEST THE VALUE OF THE "LOOP COUNTER" TO SEE IF IT HAS REACHED A VALUE OF ZERO. IF IT HAS NOT, THE PROGRAM WILL CYCLE THROUGH THE ENTIRE TEST OF EVERY WORD ON THE "PAGE" AGAIN. WHEN THE "LOOP COUNTER" DOES REACH A VALUE OF ZERO, THE TEST IS CONCLUDED BY THE COMPUTER COMING TO A HALT WITH ALL THE MEMORY CONTENTS LAMPS TURNED OFF TO SIGNIFY TO THE OPERATOR THAT A SATISFACTORY TEST HAS BEEN COMPLETED.

THE PROGRAM THUS SERVES AS A "DIAGNOSTIC" PROGRAM. IT IS A PROGRAM THAT ACTUALLY ENABLES THE COMPUTER TO TEST A PORTION OF ITS OWN MEMORY. NATURALLY, THE PROGRAM ITSELF MUST BE PLACED IN A SECTION OF MEMORY THAT IS OPERATIONAL, AND THE CPU PART OF THE COMPUTER MUST BE OPERATIONAL IN ORDER TO PERFORM THE TEST. HOWEVER, THE PROGRAM IS A VERY PRACTICAL ONE BECAUSE IF A PROBLEM SHOULD ARISE IN THE MEMORY PORTION OF THE COMPUTER IT IS MOST LIKELY TO BE RESTRICTED TO JUST ONE MEMORY ELEMENT - THE REST OF THE COMPUTER REMAINS FULLY FUNCTIONAL. IN THE EXAMPLE ILLUSTRATED HERE THE PROGRAM HAS BEEN WRITTEN TO RESIDE ON PAGE 00. THEN, THE "H" REGISTER IN THE CPU IS SET (PRIOR TO STARTING THE PROGRAM) TO THE PAGE OF MEMORY THAT IS TO BE TESTED. IT IS IMPORTANT TO NOTE HERE THAT IF CPU REGISTER "H" WAS SET TO PAGE 00 WHERE THE ACTUAL TESTING PROGRAM RESIDES, THAT THE PROGRAM WOULD LITERALLY "DESTROY" A PORTION OF ITSELF WHEN IT WAS DIRECTED TO WRITE INTO A

WORD THAT ACTUALLY CONTAINED AN INSTRUCTION USED BY THE PROGRAM! THE USER MUST THUS MAKE SURE THAT REGISTER "H" DOES NOT "POINT" TO PAGE 00 WHEN THE PROGRAM IS EXECUTED. SINCE THE PROGRAM ITSELF IS ON PAGE 00, AND SINCE THE PROGRAM MUST PERFORM THE TEST ON SOME PAGE OTHER THAN PAGE 00, THEN THE MINIMUM AMOUNT OF MEMORY NEEDED IN A SCELBI-8H SYSTEM IN ORDER TO UTILIZE THIS PROGRAM IS TWO PAGES (512 DECIMAL WORDS) OF MEMORY.

THE PROGRAM IS A GOOD ONE FOR THE PROGRAMMER TO STUDY CAREFULLY AT THIS TIME AS IT CONTAINS QUITE A FEW DIFFERENT TYPES OF COMMONLY USED INSTRUCTIONS INCLUDING "JMP" (JUMP) AND "CAL" (CALL) COMMANDS. BY STUDYING THIS PROGRAM THE USER CAN START TO GET IDEAS ON UTILIZING VARIOUS PROGRAMMING TECHNIQUES.

THE FIRST LISTING OF THIS PROGRAM WAS DONE USING THE "MNEMONICS" (GIVEN IN CHAPTER 2) FOR THE VARIOUS INSTRUCTIONS. "MNEMONICS" ARE A "SYMBOLIC LANGUAGE" REPRESENTATION OF THE MACHINE LANGUAGE CODES USED BY THE COMPUTER. IT IS GENERALLY MUCH EASIER TO FIRST WRITE A PROGRAM USING THE INSTRUCTION "MNEMONICS." THEN, WHEN THE PROGRAM IS IN SATISFACTORY ORDER, IT IS AN EASY MATTER TO ASSIGN THE MEMORY ADDRESSES TO EACH INSTRUCTION AND CONVERT THE "MNEMONICS" TO THE ACTUAL MACHINE LANGUAGE CODES. THE CONVERSION OF "MNEMONICS" TO ACTUAL MACHINE CODES AND THE ASSIGNMENT OF MEMORY ADDRESSES CAN ALSO BE DONE BY AN "ASSEMBLER PROGRAM" WHICH SAVES A LOT OF WORK FOR THE PROGRAMMER WHEN A LARGE PROGRAM IS BEING DEVELOPED. HOWEVER, IN THE EXAMPLES PROVIDED HERE THE CONVERSION PROCESS WILL BE ILLUSTRATED FOR THE MANUAL METHOD.

IN THE FIRST LISTING OF THE PROGRAM PRESENTED BELOW THE PROGRAMMER NUMBERED EACH INSTRUCTION AND ALSO KEPT A COUNT ON HOW MANY WORDS OF MEMORY EACH TYPE OF INSTRUCTION REQUIRED. THIS IS A GOOD PRACTICE AS IT ALLOWS THE PROGRAMMER TO BE ABLE TO QUICKLY DETERMINE HOW MANY WORDS IN MEMORY WILL BE REQUIRED BY THE PROGRAM. ALSO, DURING DEVELOPMENT OF THE PROGRAM THE PROGRAMMER ASSIGNED "LABELS" TO KEY INSTRUCTIONS THAT MIGHT BE REFERRED TO BY OTHER INSTRUCTIONS. A "LABEL" CAN BE CONSIDERED AS A SYMBOLIC REPRESENTATION OF AN "ADDRESS." WHEN THE PROGRAMMER DECIDES WHERE IN MEMORY THE PROGRAM WILL RESIDE, THE "LABELS" CAN BE CONVERTED TO THE ACTUAL ADDRESSES IN MEMORY OF THE INSTRUCTIONS THAT HAVE BEEN "LABELED."

MEMORY TEST PROGRAM AS ORIGINALLY DEVELOPED IN MNEMONIC FORM

INSTRUCTION #	NUMBER MEMORY WORDS REQD	MNEMONIC FORM	PROGRAMMERS COMMENTS
1	1	AGAIN, XRA	/SET ACCUMULATOR TO ZEROS
2	1	LMA	/DEPOSIT ACCUMULATOR INTO MEMORY
3	1	LAM	/NOW READ BACK FROM SAME LOCATION
4	1	NDA	/BOOLEAN OP SETS UP FLAGS AFTER LOAD
5	3	JFZ STOP	/ACCUMULATOR SHOULD BE ALL 0'S
6	2	LMI 377	/NOW LOAD MEMORY WITH ALL 1'S
7	1	LAM	/AND THEN READ IT BACK

INSTRUC- TION #	NUMBER MEMORY WORDS REQD	MNEMONIC FORM	PROGRAMMERS COMMENTS
8	2	ADI 001	/IF ADD 1 TO 377 HAVE 000!
9	1	RTZ	/END SUBROUTINE IF O.K.
10	1	STOP, 377	/OTHERWISE HALT WITH MC LAMPS ON
11	1	NEWTES, XRA	/PROGRAM STARTS HERE - CLEAR ACC
12	2	LDI 200	/SET UP LOOP CNTR IN CPU REG "D"
13	3	GO, CAL AGAIN	/CAL SUBRTN TO WRITE 0'S AND 1'S
14	1	INL	/SET POINTER TO NEXT MEMORY LOC
15	3	JFZ GO	/REPEAT TEST IF NOT THROUGH PAGE
16	1	DCD	/DECR LOOP CNTR IF THROUGH PAGE
17	3	JFZ GO	/CONTINUE TEST IF CNTR IS NOT ZERO
18	1	ALDONE, 000	/TEST DONE O.K. - MC LAMPS OFF
19	3	JMP NEWTES	/ALLOW EASY RESTART

NOW THAT THE PROGRAM HAS BEEN WRITTEN IN MNEMONIC FORM THE PROGRAMMER CAN COUNT UP THE NUMBER OF MEMORY WORDS REQUIRED TO STORE THE PROGRAM AND THEN DECIDE WHERE TO PLACE THE PROGRAM IN MEMORY. IN THIS CASE THE PROGRAMMER DECIDED TO PLACE THE PROGRAM ON PAGE 00 STARTING AT LOCATION 000. THE NEXT LISTING SHOWS THE RESULTS OF CONVERTING THE MNEMONIC SYMBOLS OVER TO THE ACTUAL MACHINE LANGUAGE CODES AND THE PROCESS OF ASSIGNING THE CODES TO SPECIFIC LOCATIONS IN MEMORY.

MEMORY TEST PROGRAM IN MACHINE CODE FORM

MEMORY ADDRESS PAGE	 LOC	MACHINE CODES	ORIGINAL MNEMONICS	COMMENTS
00	000	250	AGAIN, XRA	/SET ACCUMULATOR TO ZEROS
00	001	370	LMA	/DEPOSIT ACCUMULATOR INTO MEMORY
00	002	307	LAM	/NOW READ BACK FROM SAME LOCATION
00	003	240	NDA	/BOOLEAN OP SETS UP FLAGS
00	004	110	JFZ STOP	/ACCUMULATOR SHOULD BE ALL 0'S
00	005	015		
00	006	000		

```
  MEMORY     MACHINE    ORIGINAL
 ADDRESS      CODES    MNEMONICS                  COMMENTS
PAGE  LOC
--------------------------------------------------------------------------


 00   007      076      LMI 377       /NOW LOAD MEMORY WITH ALL 1'S
 00   010      377

 00   011      307      LAM           /AND THEN READ IT BACK

 00   012      004      ADI 001       /IF ADD 1 TO 377 HAVE 000!
 00   013      001

 00   014      053      RTZ           /END SUBROUTINE IF O.K.

 00   015      377      STOP, 377     /OTHERWISE HALT WITH MC LAMPS ON

 00   016      250      NEWTES, XRA   /PROGRAM STARTS HERE - CLEAR ACC

 00   017      036      LDI 200       /SET UP LOOP CNTR IN CPU REG "D"
 00   020      200

 00   021      106      GO, CAL AGAIN /CAL SUBRTN TO WRITE 0'S AND 1'S
 00   022      000
 00   023      000

 00   024      060      INL           /SET POINTER TO NEXT MEMORY LOC

 00   025      110      JFZ GO        /REPEAT TEST IF NOT THROUGH PAGE
 00   026      021
 00   027      000

 00   030      031      DCD           /DECR LOOP CNTR IF THROUGH PAGE

 00   031      110      JFZ GO        /CONTINUE TEST IF CNTR IS NOT ZERO
 00   032      021
 00   033      000

 00   034      000      ALDONE, 000   /TEST DONE O.K. - MC LAMPS OFF

 00   035      104      JMP NEWTES    /ALLOW EASY RESTART
 00   036      016
 00   037      000
```

THE READER SHOULD STUDY THE PROGRAM, REFERRING TO CHAPTER TWO WHEN NECESSARY, UNTIL THE PROGRAM'S OPERATION, AND THE TECHNIQUE USED TO DEVELOP THE PROGRAM AND CONVERT IT TO MACHINE LANGUAGE IS UNDERSTOOD. SEVERAL SALIENT FEATURES OF THE PROGRAM AND COMMENTS ON THE PROGRAM'S DEVELOPMENT ARE DISCUSSED BELOW.

FOR INSTANCE, THE VERY FIRST INSTRUCTION USED (XRA = EXCLUSIVE "OR" THE CONTENTS OF THE ACCUMULATOR WITH ITSELF) IS A LITTLE PROGRAMMING TRICK USED AS AN EASY WAY TO "CLEAR" THE ACCUMULATOR TO A VALUE OF ZERO. ANOTHER WAY TO HAVE PERFORMED THE SAME FUNCTION WOULD HAVE BEEN TO USE A "LAI 000" (LOAD THE ACCUMULATOR IMMEDIATE WITH 000) INSTRUCTION. NOTE HOWEVER THAT THE LATTER METHOD REQUIRES TWO WORDS OF MEMORY WHERE-AS THE "XRA" COMMAND ONLY REQUIRES ONE. A GOOD PROGRAMMER SOON GETS IN THE HABIT OF TRYING TO USE THE SMALLEST NUMBER OF LOCATIONS IN MEMORY POSSIBLE TO PERFORM A FUNCTION IN ORDER TO CONSERVE MEMORY

SPACE. THE MORE MEMORY IN A MACHINE, THE MORE EXPENSIVE THE MACHINE. IT IS THUS WISE TO TRY AND GET MAXIMUM USE OUT OF AVAILABLE MEMORY BY USING GOOD PROGRAMMING TECHNIQUES.

AN INTERESTING SEQUENCE OF INSTRUCTIONS STARTS WITH THE "LAM" (LOAD MEMORY TO ACCUMULATOR) AT PAGE 00 LOCATION 011. AT THAT POINT THE ACCUMULATOR SHOULD CONTAIN ALL ONES PROVIDED THAT THE MEMORY WORD BEING TESTED HAS PROPERLY RETAINED THE ONES WHICH WERE PREVIOUSLY WRITTEN INTO THE WORD. IN ORDER TO TEST THAT THE ACCUMULATOR DOES ACTUALLY CONTAIN ALL ONES THE MATHEMATICL VALUE "1" IS ADDED TO THE CONTENTS IN THE ACCUMULATOR USING THE "ADD IMMEDIATE 001" (ADI 001) INSTRUCTION. WHEN 001 (OCTAL) IS ADDED TO AN EIGHT BIT CLOSED REGISTER THAT HAS ALL ITS BITS ALREADY SET TO "1," (377 OCTAL) THE REGISTER WILL "OVER-FLOW" AND RESULT IN THE REGISTER HAVING A VALUE OF 000 OCTAL. HOWEVER, IF THE REGISTER HAD A "0" IN ANY ONE OF ITS BIT POSITIONS, THEN ADDING THE VALUE "1" WOULD NOT CAUSE THE REGISTER TO "OVER-FLOW" AND THE REGISTER WOULD HAVE SOME NON-ZERO VALUE. THE PROCESS IS DETAILED IN FIGURE 8.

```
1 1    1 1 1    1 1 1     IF THE ACCUMULATOR IS FILLED WITH 1'S

0 0    0 0 0    0 0 1     THEN ADDING THE MATHEMATICAL VALUE OF 1
---------------------
0 0    0 0 0    0 0 0     CAUSES REGISTER TO OVER-FLOW TO ALL 0'S


1 1    0 1 1    1 1 1     BUT IF ACCUMULATOR CONTAINS ANY 0'S

0 0    0 0 0    0 0 1     THEN ADDING THE MATHEMATICAL VALUE OF 1
---------------------
1 1    1 0 0    0 0 0     WILL RESULT IN A NON-ZERO VALUE
```

FIGURE 8

THUS AFTER THE VALUE 001 (OCTAL) IS ADDED TO THE ACCUMULATOR A "RETURN IF THE ZERO FLAG IS TRUE" (RTZ) TYPE INSTRUCTION IS USED TO TEST WHETHER THE ACCUMULATOR IS ZERO AND IF SO THE PROGRAM "RETURNS" TO THE ORIGINAL CALLING ROUTINE. BUT, IF THE ZERO FLAG IS NOT TRUE (MEANING THAT THE ACCUMULATOR HAS SOME NON-ZERO VALUE) THEN THE NEXT INSTRUCTION IN THE CURRENT SEQUENCE WILL BE EXECUTED. THAT INSTRUCTION WILL CAUSE THE COMPUTER TO HALT. AGAIN, THERE IS MORE THAN ONE WAY TO PERFORM THE ABOVE FUNCTION. FOR INSTANCE, PRIOR TO DOING THE "ADI 001" INSTRUCTION AN "NDA" (BOOLEAN "AND" THE CONTENTS OF THE ACCUMULATOR WITH ITSELF) COULD HAVE BEEN PERFORMED. THE "NDA" INSTRUCTION WOULD NOT ALTER THE CONTENTS OF THE ACCUMULATOR BUT IT IS AN INSTRUCTION THAT WILL "CLEAR" THE CARRY FLAG, THUS PUTTING IT IN A "KNOWN STATE." THEN, IF THE "ADI 001" INSTRUCTION CAUSES THE ACCUMULATOR TO "OVER-FLOW" THE CARRY FLAG WOULD BE SET. THUS, A "JTC" (JUMP ON TRUE CARRY FLAG) TYPE OF INSTRUCTION COULD BE USED TO TEST THE RESULTS OF THE ADDITION. THIS METHOD WOULD HAVE REQUIRED AN ADDITIONAL MEMORY WORD OVER THE METHOD USED (FOR THE "NDA" INSTRUCTION TO GUARANTEE THAT THE CARRY FLAG WOULD BE IN A CLEARED CONDITION PRIOR TO EXECUTION OF THE "ADI 001" COMMAND.)

THE USER SHOULD NOTE HOW CPU REGISTER "D" IS USED IN THE PROGRAM AS A "LOOP COUNTER." AT LOCATION 017 AND 020 A "LOAD 'D' IMMEDIATE" (LDI 200) INSTRUCTION IS USED TO PUT THE OCTAL VALUE 200 INTO THE REGISTER. THEN, AT LOCATION 030 REGISTER "D" IS DECREMENTED EACH TIME THE PROGRAM FINISHES TESTING ALL THE WORDS ON THE PAGE IN MEMORY WHICH IS

BEING TESTED. SINCE THE DECREMENT INSTRUCTION IS IN A CLASS OF INSTRUCTIONS WHICH AUTOMATICALLY CAUSE THE ZERO, SIGN, AND PARITY FLAGS (SEE CHAPTER TWO) TO BE SET AS A FUNCTION OF THE CONTENTS OF THE REGISTER THAT WAS DECREMENTED - IMMEDIATELY AFTER THE DECREMENT OPERATION OCCURS - THEN THE "DCD" INSTRUCTION CAN BE IMMEDIATELY FOLLOWED BY A "CONDITIONAL" BRANCHING INSTRUCTION SUCH AS THE "JFZ" (JUMP IF THE ZERO FLAG IS NOT SET) COMMAND. THAT IS WHAT IS DONE AT LOCATION 031 IN THE PROGRAM. IF THE REGISTER HAS NOT REACHED ZERO AFTER IT IS DECREMENTED THEN THE "JFZ" INSTRUCTION WILL DIRECT THE PROGRAM BACK UP TO LOCATION 021 WHERE THE PROGRAM GOES THROUGH THE TESTING CYCLE AGAIN. WHEN THE "D" REGISTER DOES REACH ZERO, THEN A "JUMP" IS NOT PERFORMED BY THE "JFZ GO" INSTRUCTION AND INSTEAD THE "HLT" (HALT) INSTRUCTION IMMEDIATELY FOLLOWING THE "JFZ GO" COMMAND (AT LOCATION 034) IS ENCOUNTERED.

NOTE THAT USE IS MADE OF THE FACT THAT THERE ARE SEVERAL DIFFERENT MACHINE CODES FOR THE "HLT" INSTRUCTION. THE FACT THAT THESE CODES WILL BE DISPLAYED IN THE MEMORY CONTENTS LAMPS WHEN THE INSTRUCTION IS EXECUTED PROVIDES AN EASY WAY TO SIGNIFY TO THE OPERATOR JUST WHAT TYPE OF EVENT CAUSED THE COMPUTER TO STOP. AT LOCATION 015 THE HALT CODE 377 IS USED. IF THE PROGRAM REACHES THIS HALT INSTRUCTION THE 377 CODE WILL CAUSE ALL THE MEMORY CONTENTS LAMPS TO BE LIT. THE OPERATOR CAN THEN TELL THAT A TEST FAILURE OCCURRED. ON THE OTHER HAND, AT LOCATION 034 THE CODE 000 IS USED FOR THE HALT INSTRUCTION. THIS HALT IS REACHED WHEN THE PROGRAM HAS SUCCESSFULLY "LOOPED" THROUGH THE ENTIRE TEST CYCLE 200 OCTAL (128 DECIMAL) TIMES. THE CODE 000 CAUSES ALL THE MC LAMPS TO BE TURNED OFF. THUS, WHENEVER THE COMPUTER STOPS WHILE THE PROGRAM IS BEING USED, THE OPERATOR CAN TELL AT A GLANCE WHETHER THE TEST WAS SUCCESSFULLY COMPLETED OR IF THE MACHINE STOPPED BECAUSE A PROBLEM WAS DETECTED.

IMMEDIATELY FOLLOWING THE HALT INSTRUCTION AT LOCATION 034 IS A "JMP NEWTES" INSTRUCTION. PLACING THIS INSTRUCTION AT THAT POINT ALLOWS THE OPERATOR TO SIMPLY USE A "LAA" INSTRUCTION VIA THE INTERRUPT FACILITY TO GET THE COMPUTER OUT OF THE STOPPED STATE AND START ANOTHER TEST, RATHER THAN HAVING TO INSERT A THREE STEP "JUMP TO LOCATION 016 ON PAGE 00" WHICH WOULD REQUIRE THREE CHANGES OF THE CHASSIS TOGGLE SWITCHES.

A SIMILAR CONCEPT IS USED IN THE ORGANIZATION OF THE PROGRAM TO ALLOW EASY RESTARTING OF THE PROGRAM IN THE EVENT THE HALT AT LOCATION 015 IS ENCOUNTERED. SIMPLY INSERTING AN "INTERRUPT" NO-OPERATION COMMAND (SUCH AS "LAA") WILL CAUSE THE PROGRAM TO START OVER BEGINNING WITH THE INSTRUCTION AT LOCATION 016.

THE READER SHOULD UNDERSTAND HOW THE "CAL AGAIN" INSTRUCTION THAT IS IN LOCATIONS 021, 022 AND 023 SERVES TO ALLOW THE BASIC TEST ROUTINE TO BE USED OVER AND OVER. IT SHOULD ALSO BE NOTED THAT CPU REGISTER "L" IS INCREMENTED AFTER EACH "CALL" OF THE TEST ROUTINE IN ORDER TO CHANGE THE ADDRESS OF THE MEMORY WORD THAT IS TO BE TESTED. (REMEMBER THAT THE "H" AND "L" CPU REGISTERS "POINT" TO THE LOCATION IN MEMORY THAT IS OPERATED ON BY THE "LAM," "LMA," AND OTHER TYPES OF INSTRUCTIONS THAT REFER TO WORDS IN MEMORY.

FINALLY, THE USER IS AGAIN CAUTIONED TO MAKE SURE THAT CPU REGISTER "H" IS SET TO THE PAGE IN MEMORY THAT IS TO BE TESTED (VIA AN INTERRUPT COMMAND) PRIOR TO STARTING THE PROGRAM AND THAT IT IS NOT SET TO PAGE 00 WHERE THE PROGRAM ITSELF IS STORED!

THE READER CAN MANUALLY LOAD THE PROGRAM INTO MEMORY BY USING THE TECHNIQUE PREVIOUSLY DESCRIBED WHEN THE TINY "JUMP TO ITSELF" PROGRAM WAS ILLUSTRATED EARLIER IN THIS CHAPTER. FIRST THE PROGRAM COUNTER

SHOULD BE SET TO PAGE 77 LOCATION Ø77. THEN REGISTER "H" SET TO ØØ AND REGISTER "L" SET TO ØØØ. NEXT "LMI" AND "INL" INSTRUCTIONS ARE ALTERNATED USING THE INTERRUPT FACILITY TO LOAD THE PROGRAM INTO MEMORY. THE MACHINE CODES FOR THE SAMPLE PROGRAM MAY BE TAKEN RIGHT FROM THE LISTING OF THE PROGRAM. WHEN THE PROGRAM HAS BEEN LOADED BE SURE TO SET REGISTER "H" TO A PAGE OTHER THAN ØØ. (REGISTER "H" SHOULD BE SET TO THE PAGE WHERE MEMORY ELEMENTS ARE TO BE TESTED.) THE PROGRAM MAY THEN BE STARTED BY USING A "JUMP TO LOCATION Ø16 ON PAGE ØØ" COMMAND VIA THE INTERRUPT MODE. THE OPERATOR MAY WISH TO FIRST STEP THE PROGRAM THROUGH THE FIRST "CAL" SUBROUTINE AS A CHECK TO SEE THAT THE PROGRAM HAS BEEN CORRECTLY LOADED. PRESSING THE "RUN" BUTTON WILL CAUSE THE PROGRAM TO BE EXECUTED AUTOMATICALLY. IF THE PROGRAM HAS BEEN CORRECTLY LOADED INTO MEMORY THE OPERATOR WILL SEE THE "RUN" LAMP ON THE FRONT PANEL CARD LIGHT. THE PROGRAM WILL THEN RUN FOR A SHORT WHILE (THROUGH 2ØØ OCTAL "LOOPS") AND THEN STOP WITH THE MEMORY CONTENTS LAMPS OFF.

IF THE PROGRAM DOES NOT APPEAR TO RUN CORRECTLY THEN THE USER MAY USE VARIOUS TYPES OF "INTERRUPT MODE" COMMANDS TO DISPLAY MEMORY LOCATIONS AND MAKE NECESSARY CORRECTIONS (OR CAN STEP THROUGH THE PROGRAM LOOKING FOR INCORRECT INSTRUCTIONS.) IT SHOULD BE POINTED OUT THAT THE PROGRAM HAS BEEN OPERATED MANY HUNDREDS OF TIMES AND IS KNOWN TO BE A "GOOD" PROGRAM. THE MOST COMMON PROBLEMS ENCOUNTERED BY BEGINNERS ARE USUALLY THOSE ASSOCIATED WITH HAVING LOADED THE PROGRAM INCORRECTLY. REMEMBER - EACH AND EVERY INSTRUCTION MUST BE LOADED WITH THE CORRECT MACHINE CODE IN THE RIGHT MEMORY LOCATION. IF THEY ARE NOT - THE MACHINE WILL PERFORM WHATEVER INSTRUCTION IT ENCOUNTERS AND THE RESULTS OF ITS PERFORMING AN UNINTENDED INSTRUCTION CAN BE HARMLESS AND AMUSING, OR, NOT SO FUNNY AND EVEN DESTRUCTIVE TO THE PROGRAM (OR OTHER PROGRAMS THAT MIGHT BE ELSEWHERE IN MEMORY) DEPENDING ON THE INDIVIDUAL CIRCUMSTANCES. AS A GENERAL RULE IT IS MUCH BETTER TO TAKE CARE TO LOAD A PROGRAM CORRECTLY AND TO MAKES CHECKS THAT IT WAS LOADED PROPERLY BEFORE STARTING THE EXECUTION OF THE PROGRAM, RATHER THAN HASTILY EXECUTING A PROGRAM THAT CONTAINS INCORRECT INSTRUCTIONS. IT IS OFTEN MUCH MORE DIFFICULT (SOMETIMES IMPOSSIBLE!) TO TRY AND FIGURE OUT WHAT HAPPENED AFTER AN INCORRECTLY LOADED PROGRAM HAS BEEN STARTED AND GONE "BESERK," THAN IT IS TO SIMPLY STEP THROUGH THE PROGRAM AT THE BEGINNING TO MAKE SURE THAT IT HAS THE INSTRUCTIONS INTENDED. REMEMBER, WHEN THE COMPUTER IS IN THE "RUN" MODE IT WILL EXECUTE INSTRUCTIONS AT A RATE OF MANY THOUSANDS PER SECOND. IN A FRACTION OF A SECOND IT CAN DO JUST AS MANY "WRONG ACTIONS" IF THE PROGRAM IS INCORRECT, AS IT CAN DO "RIGHT ACTIONS" WHEN THE PROGRAM IS PERFORMING AS DESIRED.

AN ILLUSTRATIVE "LIGHT FLASHER" PROGRAM

ANOTHER PROGRAM THAT THE USER CAN USE TO FURTHER PRACTICE LOADING A PROGRAM AND OPERATING THE SCELBI-8H IS SHOWN BELOW. THIS PROGRAM IS A SIMPLE LITTLE DISPLAY PROGRAM THAT WILL CAUSE THE MEMORY ADDRESS PAGE LAMPS TO FLASH IN SEQUENCE. THE PROGRAM IS INTERESTING IN THAT IT HAS A PROGRAMMED "TIMING LOOP" THAT USES THE COMPUTER ITSELF TO CONTROL HOW FAST THE LIGHTS TURN ON AND OFF. THE LISTING ON THE NEXT PAGE IS PROVIDED WITH COMMENTS TO AID THE READER IN UNDERSTANDING THE PROGRAM'S OPERATION. ESSENTIALLY THE PROGRAM USES A "LAM" INSTRUCTION REPEATEDLY TO CAUSE THE MEMORY PAGE ADDRESS LAMPS TO LIGHT OFTEN ENOUGH TO BE VISIBLE WHEN THE COMPUTER IS IN THE "RUN" MODE. THE PAGE POINTER (CPU REGISTER "H") IS ADVANCED AT INTERVALS DETERMINED BY A "TIMER LOOP." CHANGING THE VALUE IN LOCATION Ø21 OF THE PROGRAM WILL CHANGE THE AMOUNT OF TIME IT TAKES TO ADVANCE THE PAGE POINTER. THE PROGRAM STARTS AT LOCATION Ø25 ON PAGE ØØ.

```
  MEMORY     MACHINE
 ADDRESS      CODES     MNEMONICS                   COMMENTS
PAGE  LOC
------------------------------------------------------------------------

 00    000     050      NEWPNT, INH          /ADVANCE PAGE POINTER

 00    001     305      LAH                  /PUT "H" IN ACCUMULATOR

 00    002     004      ADI 300              /SET UP FOR TEST
 00    003     300

 00    004     110      JFZ LOOP1            /IF VALUE IS 100 DO NEXT INSTR
 00    005     011
 00    006     000

 00    007     056      LHI 000              /RESET "H" TO PAGE 00
 00    010     000

 00    011     307      LOOP1, LAM           /PAGE LIGHTS LOOP

 00    012     040      INE                  /DUTY CYCLE COUNTER

 00    013     110      JFZ LOOP1            /LOOP UNTIL REG "E" = 000
 00    014     011
 00    015     000

 00    016     030      IND                  /ADVANCE TIMER FOR TIMER LOOP

 00    017     303      LAD                  /PUT VALUE OF "D" INTO ACC

 00    020     004      ADI 350              /THIS SETS SEQUENCE TIME
 00    021     350

 00    022     110      JFZ LOOP1            /KEEP DUTY CYCLE GOING
 00    023     011
 00    024     000

 00    025     036      START, LDI 000       /SET UP NEW TIMER
 00    026     000

 00    027     104      JMP NEWPNT           /CHANGE PAGE POINTER
 00    030     000
 00    031     000
```

AT THIS POINT THE READER SHOULD HAVE A GOOD FUNDAMENTAL UNDERSTANDING OF HOW TO OPERATE A SCELBI-8H MINI-COMPUTER. THE USER HAS BEEN PRESENTED WITH THE BASIC OPERATING PROCEDURES OF THE MACHINE, HAS BEEN PROVIDED WITH DETAILED TABLES DESCRIBING EVERY TYPE OF OPERATION, AND HAS SEEN SOME SAMPLE PROGRAMS. THE READER HAS HOPEFULLY SPENT SOME TIME LOADING PROGRAMS VIA THE MANUAL METHOD AND THEN HAD THE PLEASURE OF SEEING THEM EXECUTED BY THE COMPUTER.

THE USER SHOULD NOW BE IN A POSITION TO START PUTTING THE SCELBI-8H TO USE PERFORMING TASKS SPECIFICALLY DESIRED BY THE INDIVIDUAL USER. BY UTILIZING THE TABLES AND EXAMPLES IN THIS CHAPTER, AND REFERRING TO THE INFORMATION IN CHAPTER TWO, THE USERS ARE IN A POSITION TO CREATE THEIR OWN PROGRAMS, MANUALLY LOAD THEM INTO THE COMPUTER, AND THEN HAVE THE COMPUTER EXECUTE THE PROGRAMS.

COMMENTS ON OTHER PROGRAM LOADING METHODS

THE PROCESS OF MANUALLY LOADING PROGRAMS INTO MEMORY VIA THE CONSOLE SWITCHES IS SUITABLE FOR RELATIVELY SMALL PROGRAMS AND MANY APPLICATIONS ONLY REQUIRE SUCH PROGRAMS. HOWEVER, FOR LARGE SOPHISTICATED PROGRAMS IT IS GENERALLY DESIRABLE TO USE OTHER DEVICES IN CONJUNCTION WITH THE COMPUTER TO SIMPLIFY THE LOADING PROCESS FOR THE OPERATOR WHILE SIGNIFICANTLY INCREASING THE SPEED WITH WHICH PROGRAMS CAN BE LOADED INTO MEMORY.

WHILE THE NEXT CHAPTER WILL GO INTO SPECIFIC DETAILS ON CONNECTING INPUT/OUTPUT (I/O) DEVICES TO THE SCELBI-8H IT IS PERHAPS WORTH MENTIONING HERE A FEW OF THE TYPICAL DEVICES THAT CAN BE CONNECTED TO THE COMPUTER TO SPEED UP THE PROCESS OF LOADING PROGRAMS. AND, IN ADDITION, TO COMMENT ON HOW RELATIVELY SIMPLE "LOADER" PROGRAMS CAN BE USED TO ENABLE THE LOADING OF LARGER PROGRAMS.

ONE OF THE MOST COMMON DEVICES AVAILABLE TO CONNECT TO A COMPUTER IS AN ELECTRONIC KEYBOARD SIMILAR TO A TYPEWRITER. WHEN LETTERS OR NUMBERS ARE TYPED ON THE UNIT A GROUP OF ELECTRONIC SIGNALS IN A CODED FORM (SUCH AS THE ASCII CODE) ARE SENT TO THE COMPUTER. WITH A SUITABLE, RELATIVELY SMALL PROGRAM IN MEMORY, THE COMPUTER IS ABLE TO INTERPRET THE SIGNALS RECEIVED AND PERFORM OPERATIONS BASED ON THE INFORMATION RECEIVED FROM THE KEYBOARD UNIT. FOR INSTANCE, A "KEYBOARD LOADER" PROGRAM WILL ENABLE A PERSON TO TYPE IN ADDRESSES AND THEN THE DATA TO GO INTO THE MEMORY LOCATION AT THOSE ADDRESSES. THIS METHOD OF LOADING PROGRAMS IS CONSIDERABLY FASTER THAN LOADING PROGRAMS VIA THE CONSOLE SWITCHES.

ANOTHER DEVICE OFTEN CONNECTED TO A COMPUTER IS A TELETYPE MACHINE. A PERSON CAN ENTER PROGRAMS ON A TELETYPE KEYBOARD SIMILAR TO THE METHOD USED WITH AN ELECTRONIC KEYBOARD. IN ADDITION, THE TELETYPE HAS A PRINTER MECHANISM THAT CAN BE USED TO RECEIVE INFORMATION FROM THE COMPUTER. MANY TELETYPE MACHINES ARE ALSO EQUIPPED WITH PAPER TAPE READERS AND PUNCHES AND THE ADDITON OF THOSE UNITS ALLOW ONE TO LOAD PROGRAMS, OR PUNCH COPIES OF DEVELOPED PROGRAMS, WITH CONSIDERABLE EASE.

AN EVEN FASTER WAY TO LOAD PROGRAMS IS TO UTILIZE A MAGNETIC TAPE SYSTEM. IN THIS TYPE OF ARRANGEMENT A MAGNETIC TAPE RECORDER IS CONNECTED TO THE COMPUTER THROUGH A SPECIAL ELECTRONIC NETWORK GENERALLY TERMED AN "INTERFACE." WITH AN APPROPRIATE SMALL PROGRAM IN THE COMPUTER THE "MAG-TAPE" SYSTEM CAN BE USED TO RAPIDLY LOAD LARGE PROGRAMS INTO MEMORY. OR, VICE-VERSA, THE COMPUTER CAN SAVE "COPIES" OF PROGRAMS FROM ITS MEMORY ONTO CASSETTES OF MAGNETIC TAPE.

QUITE OFTEN THE FIRST PROGRAM USED TO ENABLE AN EXTERNAL DEVICE SUCH AS A TELETYPE OR A "MAG-TAPE" UNIT TO BE ABLE TO LOAD PROGRAMS IS CALLED A "BOOTSTRAP LOADER." THIS TERMINOLOGY ORIGINATES FROM THE OLD EXPRESSION OF "LIFTING ONESELF UP BY ONE'S OWN BOOT STRAPS." FOR A "BOOTSTRAP LOADER" IS QUITE LITERALLY A SMALL PROGRAM THAT WILL ALLOW A LARGER MORE POWERFUL PROGRAM TO BE LOADED INTO THE COMPUTER FROM SOME SORT OF EXTERNAL DEVICE.

THE SCELBI-8H MINI-COMPUTER IS CAPABLE OF OPERATING WITH ALL OF THE ABOVE TYPES OF EXTERNAL DEVICES AND "INTERFACES" AS WELL AS PROGRAMS SUCH AS "BOOTSTRAP LOADERS" FOR THOSE TYPES OF DEVICES ARE AVAILABLE. THE SPECIFIC DETAILS OF THE DEVICE "INTERFACES" AND PROGRAMS ARE COVERED IN THE DOCUMENTATION FOR THE UNITS. SUFFICE IT TO CONCLUDE THAT THERE ARE MANY METHODS AVAILABLE FOR SPEEDING UP THE PROGRAM LOADING PROCESS OVER THAT OF THE MANUAL METHOD DESCRIBED IN THIS CHAPTER.

HOWEVER, THE MANUAL METHOD IS THE MOST FUNDAMENTAL ONE, AND THE ONE THAT MUST BE USED WHEN FIRST STARTING UP A SYSTEM. THE USER SHOULD THEREFORE BECOME THOROUGHLY FAMILIAR WITH THE CONTENTS OF THIS CHAPTER.

IN CONCLUSION OF THE CHAPTER IT WILL BE POINTED OUT THAT THE PROVISIONS FOR COMPLETE MANUAL CONTROL OF THE SCELBI-8H WHICH WERE DESIGNED INTO THE MACHINE, ALLOW THE SCELBI-8H USER TO ENJOY COMPLETE CONTROL OVER THE MACHINE WITHOUT HAVING TO HAVE ANY EXTERNAL DEVICES. WHILE EXTERNAL LOADING AND OPERATING DEVICES ARE OFTEN NICE TO HAVE, A USER CAN PUT A SCELBI-8H MINI-COMPUTER TO PLENTY OF GOOD USE WITHOUT HAVING ADDITIONAL EXTERNAL PROGRAM LOADING DEVICES!

CONNECTING EXTERNAL EQUIPMENT TO THE SCELBI-8H MINI-COMPUTER

INTERFACING INPUT AND OUTPUT DEVICES TO THE SCELBI-8H IS QUITE SIMPLE. THE SCELBI-8H HAS BEEN DESIGNED SO THAT ALL OF THE I/O CONNECTIONS ARE "TTL" (TRANSISTOR-TRANSISTOR-LOGIC) COMPATIBLE. STANDARD 7400 SERIES TTL AND LOW POWER TTL DEVICES MAY BE CONNECTED DIRECTLY TO THE INPUT AND OUTPUT PORTS.

INPUT PORTS

THERE ARE SIX INPUT PORTS ON THE SCELBI-8H DESIGINATED AS INPUT PORTS Ø THROUGH 5. EACH INPUT PORT HAS EIGHT INPUT LINES ASSOCIATED WITH IT. UNDER PROGRAM CONTROL, WHEN AN INPUT PORT HAS BEEN SELECTED TO RECEIVE INFORMATION THE COMPUTER WILL SIMPLY "SAMPLE" THE CONDITION OF ALL EIGHT INPUT LINES SIMULTANEOUSLY (I.E. IN PARALLEL) AND PLACE THE LOGIC EQUIVALENT OF EACH LINES STATUS (HIGH = "1," LOW = "Ø") IN THE ACCUMULATOR. THUS, THE SIMPLEST TYPE OF INPUT DEVICE THAT ONE MIGHT CONNECT TO AN INPUT PORT COULD JUST BE MECHANICAL SWITCHES AS SHOWN IN FIGURE 1.

```
B7 <------------------------X--------*
                                     '
                                     '
B6 <------------------------X--------*
                                     '
                                     '
B5 <------------------------X--------*
                                     '
                                     '
B4 <------------------------X--------*
                                     '
                                     '
B3 <------------------------X--------*
                                     '
                                     '
B2 <------------------------X--------*
                                     '
                                     '
B1 <------------------------X--------*
                                     '
                                     '
BØ <------------------------X--------*
                                     '
                                     '
                                  ---*---
                                   -----
                                    ---
```

FIGURE 1

NOT SHOWN IN FIGURE 1 IS THE FACT THAT EACH INPUT LINE HAS A 1Ø K-OHM "PULL-UP" RESISTOR CONNECTED TO IT (ON THE SCELBI 11Ø2- INPUT CARD) AND HENCE IN THE ABOVE DIAGRAM IF A SWITCH IS "OPEN" THE COMPUTER WILL RECEIVE A LOGIC "1" INPUT WHEN THE PORT IS SAMPLED. IF THE SWITCH WAS CLOSED SO THAT THE LINE WAS CONNECTED TO SIGNAL GROUND, THEN THE INPUT

LINE WOULD BE AT A LOGIC "0."

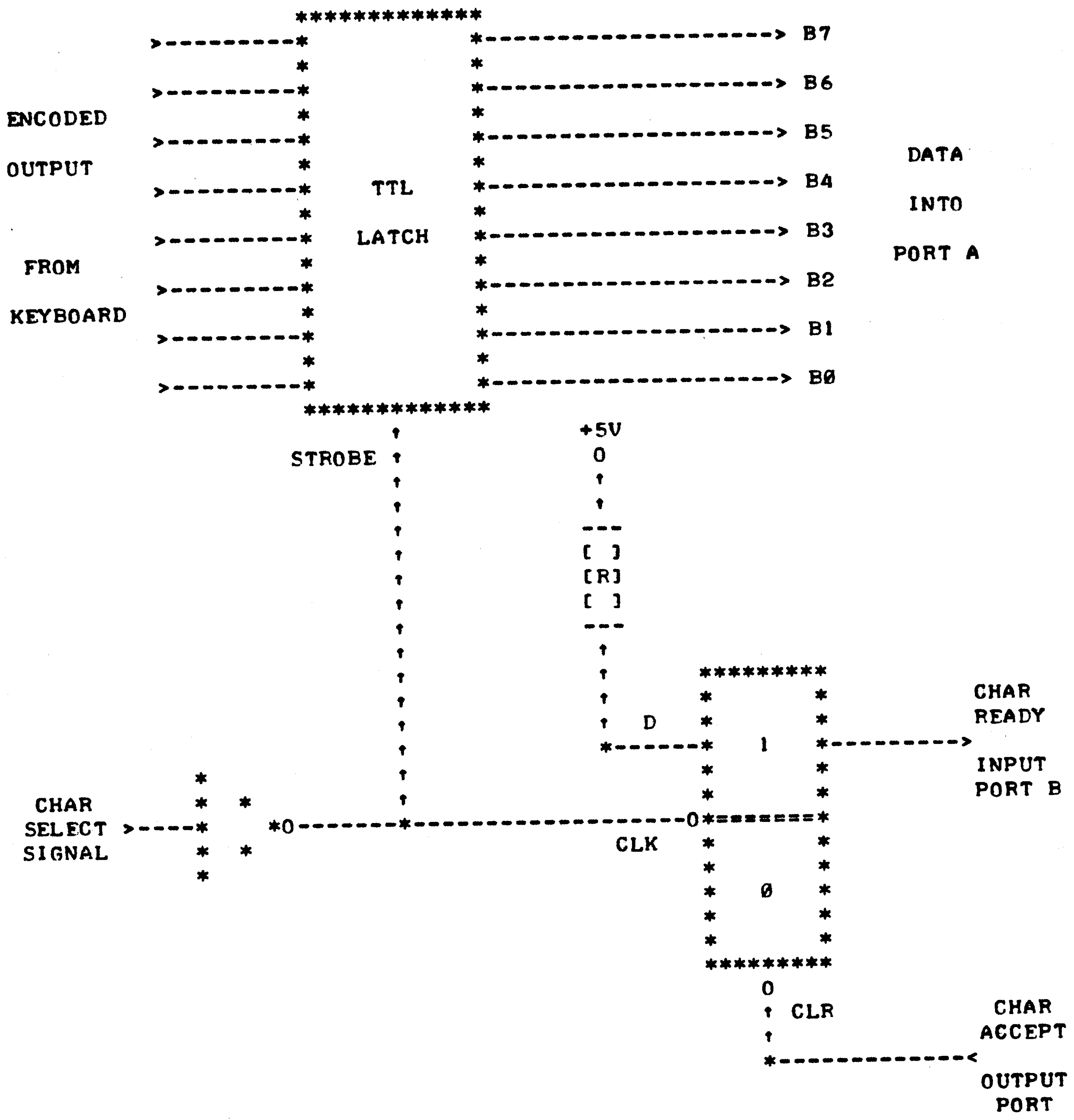
NATURALLY THE INPUT LINES CAN BE CONNECTED TO THE OUTPUTS OF TTL DEVICES INSTEAD OF MECHANICAL SWITCHES. THE 10 K-OHM "PULL-UP" RESISTORS PROVIDED ON EACH LINE ALLOW EITHER STANDARD TTL DEVICES OR "OPEN COLLECTOR" TTL DEVICES (OR EQUIVALENT CIRCUITS) TO BE CONNECTED DIRECTLY TO AN INPUT LINE. SINCE EACH LINE REPRESENTS JUST SLIGHTLY OVER ONE TTL LOAD (THE 10 K-OHM "PULL-UP" RESISTOR CONTRIBUTES A SLIGHT LOADING FACTOR) ON THE RECEIVING END IT IS EVEN POSSIBLE TO USE A LOW POWER TTL DEVICE TO DRIVE AN INPUT LINE.

THE AVAILABILITY OF EIGHT PARALLEL LINES ON EACH INPUT PORT MEANS DEVICES SUCH AS ENCODED KEYBOARDS CAN BE DIRECTLY CONNECTED TO INPUT PORTS. HOWEVER, IT IS OFTEN NECESSARY TO ADD ADDITIONAL CIRCUITRY BETWEEN A DEVICE SUCH AS AN ELECTRONIC KEYBOARD AND THE COMPUTER TO PROVIDE LATCHING OF THE DATA BETWEEN THE TIME A KEY IS DEPRESSED (AND RELEASED) AND THE TIME THE COMPUTER SAMPLES THE INPUT PORT. THE EXACT CIRCUITRY FOR INDIVIDUAL UNITS WILL DEPEND ON THE EXACT REQUIREMENTS OF THE DEVICE BEING USED BUT A GENERAL TYPE OF CIRCUIT FOR A KEYBOARD INTERFACE MIGHT APPEAR AS SHOWN IN FIGURE 2.

THE CIRCUIT SHOWN IN FIGURE 2 INCLUDES CIRCUITRY WHERE-BY THE COMPUTER CAN DETERMINE WHEN A NEW CHARACTER IS WAITING TO BE INPUTTED FROM THE KEYBOARD. AS THE DIAGRAM SHOWS, WHENEVER A KEY IS STRUCK ON THE KEYBOARD UNIT A "CHARACTER SELECTED" SIGNAL IS USED TO STROBE THE ENCODED INFORMATION FROM THE KEYBOARD INTO A LATCHING NETWORK WHERE THE INFORMATION CAN BE RETAINED AFTER THE KEY HAS BEEN RELEASED. THE "CHARACTER SELECT" SIGNAL WILL ALSO SET A "CHARACTER READY" FLIP-FLOP TO THE LOGIC "1" STATE. THE EIGHT DATA LINES FROM THE LATCHES ARE FED TO ONE INPUT PORT ON THE SCELBI-8H. THE OUTPUT OF THE "CHARACTER READY" FLIP-FLOP IS CONNECTED ON ONE LINE OF A SECOND INPUT PORT. WITH THIS ARRANGEMENT, THE COMPUTER CAN FROM TIME TO TIME SAMPLE THE "CHARACTER READY" INPUT PORT AND PERFORM A TEST ON THE BIT ASSOCIATED WITH THE INPUT LINE USED TO DETERMINE IF A NEW CHARACTER IS WAITING IN THE TTL LATCHES. IF NO NEW CHARACTER IS WAITING THEN THE PROGRAM COULD CONTINUE WITH SOME OTHER COMPUTATIONS, OR IT COULD GO INTO A "WAITING LOOP." WHENEVER THE PROGRAM DETERMINES THAT A NEW CHARACTER IS WAITING IT SIMPLY DIRECTS THE COMPUTER (VIA A BRANCHING INSTRUCTION) TO INPUT THE INFORMATION FROM THE INPUT PORT THAT IS CONNECTED TO THE DATA LINES. WHEN THIS HAS BEEN DONE THE PROGRAM CAN THEN PROCEED TO GENERATE A SIGNAL ON AN OUTPUT PORT (TO BE DISCUSSED LATER IN THIS CHAPTER) THAT WOULD "CLEAR" THE "CHARACTER READY" FLIP-FLOP.

IT SHOULD BE NOTED THAT IN THIS ARRANGEMENT ONLY ONE LINE IS USED BY THE "CHARACTER READY" SIGNAL (AND SIMILARLY ONLY ONE OUTPUT LINE IS USED TO CLEAR THE "CHARACTER READY" FLIP-FLOP.) SINCE AN INPUT (AND OUTPUT) PORT HAS EIGHT LINES AVAILABLE, THE REMAINING SEVEN LINES OF THE PORT COULD BE USED FOR SIMILAR "CONTROL" SIGNALS FROM (AND TO) A WHOLE GROUP OF DEVICES. A PROGRAM SUBROUTINE CAN THEN BE USED TO DETERMINE WHICH PARTICULAR DEVICE(S) IN A GROUP ARE READY AND SELECT THE PARTICULAR DATA PORTS FOR THE DEVICE(S) AS REQUIRED.

MENTION SHOULD BE MADE OF THE FACT THAT THOUGH THE DIAGRAM SHOWS A TTL LATCH BEING USED TO INTERFACE TO THE INPUT PORT, OTHER TYPES OF CIRCUITS COULD ALSO BE USED. FOR INSTANCE, ONE COULD INSTEAD HAVE A "SERIAL TO PARALLEL" CONVERTER THAT WOULD ACCEPT SERIAL INFORMATION FROM A DEVICE (SUCH AS A TELETYPE MACHINE) AND WHEN THE "SERIAL TO PARALLEL" CONVERTER WAS FILLED WITH DATA A STROBE SIGNAL COULD BE USED TO SET A "READY" FLIP-FLOP. THE COMPUTER COULD THEN BRING THE DATA INTO THE ACCUMULATOR FROM THE "SERIAL TO PARALLEL" CONVERTER AND ISSUE A "CLEAR" SIGNAL TO THE "READY" FLIP-FLOP.



A REPRESENTATIVE CIRCUIT FOR AN ELECTRONIC KEYBOARD INTERFACE

FIGURE 2

HOWEVER, IT IS NOT ALWAYS NECESSARY TO USE A "SERIAL TO PARALLEL" CONVERTER TO BRING INFORMATION FROM A SERIAL DEVICE INTO THE COMPUTER - ESPECIALLY IF THE DEVICE IS ASYNCHRONOUS IN OPERATION SUCH AS A TELETYPE MACHINE. IN FACT A "START AND STOP" DEVICE SUCH AS A TELETYPE MACHINE IS EVEN EASIER TO INTERFACE THAN THE ELECTRONIC KEYBOARD DISCUSSED ABOVE. IN THE SIMPLE INTERFACE FOR A TELETYPE MACHINE SHOWN IN FIGURE 3 THE INFORMATION IS BROUGHT INTO THE COMPUTER ONE BIT AT A TIME USING JUST ONE DATA LINE OF AN INPUT PORT. A PROGRAM IN THE COMPUTER IS THEN

USED TO ACCEPT THE INFORMATION RECEIVED SERIALLY BIT-BY-BIT AND FORMAT IT INTO A "CHARACTER."

IN THE CIRCUIT ILLUSTRATED THE SAME POWER SUPPLY AS THAT USED BY THE COMPUTER SUPPLIES A "LOOP CURRENT" FOR THE TELETYPE TRANSMITTER. THE TELETYPE TRANSMITTER IS ESSENTIALLY JUST A MECHANICAL SWITCH THAT OPENS AND CLOSES TO TRANSMIT A SERIES OF BITS OF INFORMATION. WHEN THE SWITCH IS CLOSED (THE TELETYPE "MARKING" CONDITION) THE TRANSISTOR IN THE CIRCUIT WILL BE TURNED ON. A TTL INVERTER IS USED FOLLOWING THE TRANSISTOR TO PROVIDE D.C. LEVEL SHIFTING AND BUFFERING PRIOR TO CONNECTING TO A DATA LINE ON AN INPUT PORT TO THE COMPUTER. THE INVERTER ALSO SERVES TO PROVIDE A LOGIC CONVENTION THAT DEFINES A "MARKING" CONDITION FROM THE TELETYPE TO BE RECEIVED AT THE COMPUTER AS A LOGIC "1" LEVEL. A TELETYPE "SPACING" CONDITION IS THEN RECEIVED AS A LOGIC "Ø" LEVEL. IN THE EXAMPLE THE SIGNAL FROM THE TELETYPE IS FED TO BIT B7 OF AN INPUT PORT SO THAT INCOMING DATA CAN BE READILY TESTED FOR A "MARKING" OR "SPACING" CONDITION BY USING A CONDITIONAL INSTRUCTION THAT DIRECTLY TESTS B7 (THAT IS AN INSTRUCTION SUCH AS A "CTS" OR A "JTS" THAT TESTS THE "SIGN" FLAG.) HOWEVER, ANY BIT POSITION COULD HAVE BEEN USED TO BRING THE DATA INTO THE ACCUMULATOR AND OTHER TYPES OF INSTRUCTIONS USED TO POSITION THE INCOMING DATA WITHIN THE ACCUMULATOR AND PERFORM TESTS TO DETERMINE THE LOGIC LEVEL OF THE INCOMING DATA.

```
                          +5V
                           O
                           †
               -------     †
 ---> >--------I  R  I-----*
 .             -------     †
 .                         †
 .                        ---
 .                        [ ]
FROM                      [R]
                          [ ]
TTY                       ---
                           †                     *
XMTR                       †        -------      *  *              INPUT
 .                         *--------I  R  I------*    *O------> B7
 .                         †        -------      *  *              PORT
 .                         †                     *
 .                      *******
 .                      *  Q  *
 ---> >-----*------*----*     *
            †      †    * NPN *
            †      †    *******
           ---     †       †
           [ ]     † C     †
           [R]   =====     †
           [ ]     †       †
           ---     †       †
            †      †       †
            †      †       †
            *------*-------*
                           †
                           †
                           O
                          -9V
```

FIGURE 3

SINCE A TELETYPE MACHINE IS AN ASYNCHRONOUS DEVICE IT IS A SIMPLE MATTER TO HAVE THE COMPUTER RECEIVE INFORMATION VIA THE SIMPLE INTERFACE. ALL THAT IS NECESSARY IS TO HAVE A PROGRAM THAT PERIODICALLY SAMPLES THE LINE ON THE INPUT PORT TO WHICH THE TELETYPE TRANSMITTER IS CONNECTED AND PERFORM A TEST TO SEE IF THE TELETYPE IS "MARKING" OR "SPACING." IF THE TELETYPE IS IN THE STEADY MARKING CONDITION THE COMPUTER CAN PERFORM SOME OTHER TASK BEFORE COMING BACK TO SAMPLE THE INPUT PORT. IF ON THE OTHER HAND A "SPACE" IS DETECTED THEN THE PROGRAM WILL KNOW THAT THE "START BIT" IS BEING SENT BY THE TELETYPE MACHINE. AT THIS TIME THE PROGRAM CAN BRANCH TO A "TIMING LOOP" THAT WILL SAMPLE THE INCOMING SERIAL DATA AT SPECIFIC TIMES (DEPENDENT ON THE OPERATING SPEED OF THE PARTICULAR MODEL OF TELETYPE MACHINE BEING USED.) THE SAMPLE POINT IS SELECTED TO FALL AT OR NEAR THE PROJECTED MID-POINT OF EACH BIT OF SERIAL DATA. AS THE DATA IS SAMPLED A LOGIC "1" (MARK) OR LOGIC "Ø" (SPACE) CAN BE STORED IN SEQUENTIAL ORDER IN A CPU REGISTER (OR WORD IN MEMORY) UNTIL ALL THE BITS FROM A "CHARACTER" HAVE BEEN RECEIVED. USING THIS METHOD THE COMPUTER EASILY CONVERTS THE SERIAL DATA TO A PARALLEL CODE (SUCH AS ASCII OR BAUDOT) AND ELIMINATES THE NEED FOR AN EXTERNAL "SERIAL TO PARALLEL" CONVERTER. WHEN ALL THE BITS OF A CHARACTER HAVE BEEN TRANSMITTED THE TELETYPE SENDS A "STOP" BIT (MARKING CONDITION.) THE COMPUTER PROGRAM CAN DETERMINE WHEN THE "STOP" BIT IS DUE BY COUNTING THE NUMBER OF BITS RECEIVED. THE COMPUTER CAN THEN USE THE TIME THAT IT TAKES FOR THE "STOP" BIT TO BE SENT BY THE TELETYPE TO PERFORM SOME OTHER TYPES OF OPERATIONS AND THEN GO BACK TO PERIODICALLY SAMPLING THE INPUT LINE TO CHECK FOR THE BEGINNING OF ANOTHER "CHARACTER."

OUTPUT PORTS

OUTPUTTING INFORMATION FROM THE SCELBI-8H MINI-COMPUTER TO AN EXTERNAL DEVICE IS JUST ABOUT AS EASY AS INPUTTING INFORMATION. EACH OUTPUT PORT IS EQUIPPED WITH EIGHT "DATA" LINES AND ONE "STROBE" LINE. THE EIGHT DATA LINES ARE ACTUALLY A BUSS ARRANGEMENT WITH THE BUSS GOING TO ALL OUTPUT PORTS. THE STROBE LINE IS USED TO SELECT THE PARTICULAR OUTPUT PORT THAT IS TO TAKE DATA FROM THE OUTPUT BUSS. THE STROBE LINE FOR THE SELECTED OUTPUT PORT IS SIMPLY PULSED BY THE COMPUTER DURING THE EXECUTION OF AN OUTPUT INSTRUCTION FOR THAT SPECIFIC PORT. THE DATA FROM AN OUTPUT PORT CAN BE STROBED INTO A TTL LATCH, OR SHIFT REGISTER, OR OTHER TYPE OF CIRCUIT THAT WILL FURTHER PROCESS THE DATA RECEIVED TO OPERATE AN EXTERNAL DEVICE. A TYPICAL CIRCUIT FOR AN OUPUT PORT IS SHOWN IN FIGURE 4.

IT SHOULD BE MENTIONED THAT THE STANDARD SCELBI-8H CHASSIS IS EQUIPPED WITH EIGHT OUTPUT PORT SOCKETS. THESE EIGHT OUTPUT PORTS ARE REFERRED TO AS OUTPUT PORTS 1Ø THROUGH 17. HOWEVER THE SCELBI 11Ø1- "DBB AND OUTPUT CARD" IS EQUIPPED WITH CIRCUITRY TO OPERATE SIXTEEN OUTPUT PORTS (THE ADDITIONAL PORTS ARE REFERRED TO AS PORTS 2Ø THROUGH 27.) THIS FACT IS MENTIONED AS SOME USERS MAY DESIRE TO HAVE A SPECIAL SYSTEM CONFIGURED OR DESIGN THEIR OWN SYSTEM (BY INSTALLING A SCELBI-8H CARD SET IN THEIR OWN CHASSIS OR OTHER SYSTEM PACKAGING ARRANGEMENT EQUIPPED WITH ADDITIONAL OUTPUT CONNECTORS) IN ORDER TO HAVE A SYSTEM WITH EXPANDED OUTPUT CAPABILITY. THE MAJORITY OF USERS, HOWEVER, WILL FIND THAT EIGHT OUTPUT PORTS PROVIDES PLENTY OF OUTPUT CAPABILITY FOR A SCELBI-8 MINI-COMPUTER SYSTEM.

SINCE THE OUTPUT DATA LINES ARE ON A BUSS STRUCTURE CARE MUST BE USED WHEN A LARGE NUMBER OF EXTERNAL DEVICES ARE CONNECTED TO THE OUTPUT PORTS. THE BUSSES ARE DESIGNED TO BE ABLE TO DRIVE UP TO FOUR STANDARD TTL LOADS. IF THE USER DESIRES TO HAVE MORE THAN FOUR OUT-

PUT PORTS IN USE (CONNECTED TO THE COMPUTER) SIMULTANEOUSLY THEN LOW POWER TTL DEVICES SHOULD BE USED TO CONNECT TO THE OUTPUT DATA BUSS. THE OUTPUT BUSS CAN READILY HANDLE UP TO 16 OUTPUT PORTS WHEN ALL PORTS ARE EQUIPPED WITH LOW POWER TTL INTERFACING CIRCUITS.

THE STROBE LINE FOR EACH OUTPUT PORT IS ABLE TO DRIVE UP TO TEN STANDARD TTL LOADS. HOWEVER, IT IS RECOMMENDED THAT A "PULL-UP" RESISTOR BE USED IF MORE THAN A FEW LOADS ARE DRIVEN BY A SINGLE STROBE LINE OR IF THE DISTANCE TO THE DEVICE BEING DRIVEN REQUIRES MORE THAN 36 INCHES OF CONNECTING WIRE.

```
                               *************
               B7 >---------------*          *---------------->
                                  *          *
               B6 >---------------*          *---------------->
                                  *          *
               B5 >---------------*          *---------------->
    FROM                          *          *                      TO
               B4 >---------------*   TTL    *---------------->
   OUTPUT                         *          *                   EXTERNAL
               B3 >---------------*  LATCH   *---------------->
    PORT                          *          *                    DEVICE
               B2 >---------------*          *---------------->
                                  *          *
               B1 >---------------*          *---------------->
                                  *          *
               B0 >---------------*          *---------------->
                               *************
                                       ↑
                                       ↑
   OUTPUT                              ↑
    PORT       >-----------------------*
   STROBE
```

FIGURE 4

QUITE OFTEN IT IS DESIRABLE TO USE TWO OR MORE OUTPUT PORTS TO CONTROL AN EXTERNAL DEVICE. FOR INSTANCE ONE PORT MIGHT BE USED PURELY FOR TRANSMITTING "DATA" TO A DEVICE AND A SECOND PORT UTILIZED FOR PASSING "CONTROL" SIGNALS TO THE MECHANISM. THE USER IS PRACTICALLY UNLIMITED IN THE NUMBER OF WAYS EXTERNAL DEVICES CAN BE CONTROLLED BY THE SCELBI-8H MINI-COMPUTER. FOR INSTANCE, THE INFORMATION TRANSMITTED FROM THE OUTPUT PORTS CAN BE TRANSLATED INTO SIGNALS THAT OPERATE RELAYS, OR ELECTRONICALLY ACTIVATED PNEUMATIC VALVES OR CYLINDERS, OR ELECTRONIC STEPPING MOTORS, OR A WHOLE HOST OF OTHER SIMILAR DEVICES AS WELL AS CONVENTIONAL ELECTRONIC CIRCUITS THAT CAN BE USED TO CONTROL THE OPERATION OF PURELY ELECTRONIC EQUIPMENT.

THEN TOO, IT IS OFTEN DESIRABLE TO USE JUST ONE OUTPUT PORT TO CONTROL A WHOLE GROUP OF DEVICES, OR TO JUST USE ONE DATA LINE OF AN OUTPUT PORT. FOR EXAMPLE, ONE CAN MAKE A VERY SIMPLE INTERFACE THAT WILL ENABLE THE COMPUTER TO DRIVE A SERIAL DEVICE SUCH AS A TELETYPE RECEIVER. IN THIS ARRANGEMENT THE COMPUTER IS USED TO CONVERT INFORMATION WITHIN THE COMPUTER FROM A PARALLEL TO A SERIAL FORMAT. AN ILLUSTRATION OF SUCH AN INTERFACE IS SHOWN IN FIGURE 5. THIS INTERFACE CAN BE USED IN CONJUNCTION WITH THE PREVIOUSLY DESCRIBED CIRCUIT THAT ACCEPTED INFORMATION FROM A TELETYPE, TO PROVIDE A SYSTEM THAT WILL TRANSMIT DATA TO A TELETYPE PRINTER, AND RECEIVE DATA FROM THE TELETYPE KEYBOARD SO THAT THE USER HAS COMPLETE INPUT/OUTPUT CAPABILITY WITH THE MACHINE.

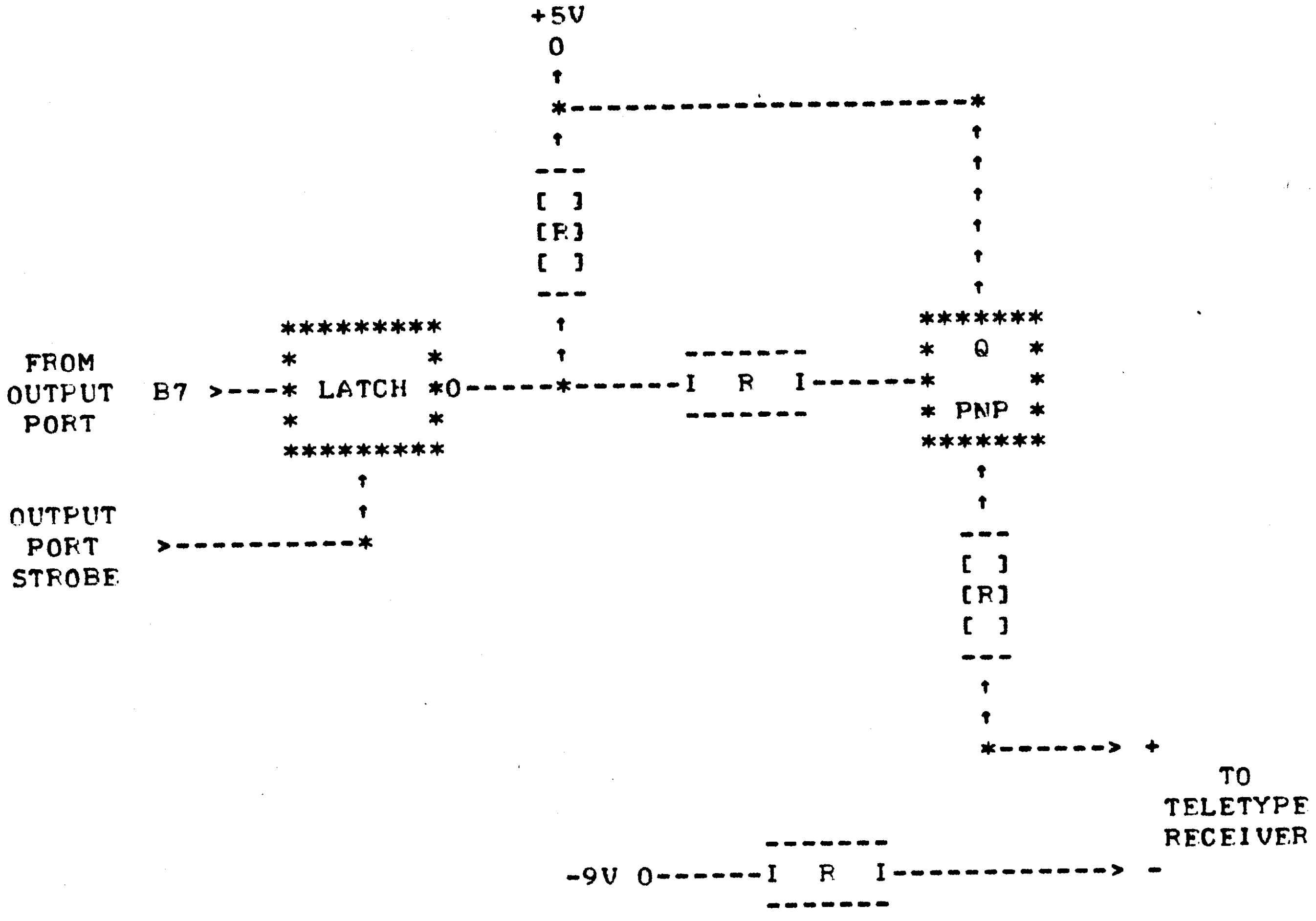


FIGURE 5

THE CIRCUIT IN FIGURE 5 OPERATES AS FOLLOWS. INFORMATION FROM THE COMPUTER IS TRANSMITTED TO THE LATCH USING BIT B7 OF AN OUTPUT PORT. SINCE THE DATA IS TO BE TRANSMITTED IN SERIAL FASHION A PROGRAM IN THE COMPUTER IS USED TO CONVERT A CHARACTER IN PARALLEL FORMAT TO A SERIAL FORMAT BY USING ROTATE INSTRUCTIONS TO HAVE EACH BIT OF THE CHARACTER POSITIONED TO BIT B7 IN THE ACCUMULATOR PRIOR TO EXECUTING THE OUTPUT INSTRUCTION FOR THE SELECTED PORT. A PROGRAMMED TIMING LOOP IS USED TO DETERMINE WHEN TO SEND THE NEXT BIT OF INFORMATION. PRIOR TO STARTING TRANSMISSION OF THE CHARACTER THE PROGRAM SENDS A "SPACE" BIT TO START THE TELETYPE AND AT THE END OF THE CHARACTER THE PROGRAM SENDS A "MARK" BIT TO PLACE THE TELETYPE IN THE "STOP" CONDITION. INFORMATION RECEIVED BY THE LATCH IN THE ABOVE CIRCUIT IS INVERTED BY USING THE INVERTING OUTPUT OF THE LATCH AND FED TO A TRANSISTOR THAT OPERATES AS A SWITCH FOR THE RECEIVING LOOP OF THE TELETYPE. THIS CONVENTION ESTABLISHES A LOGIC "1" FROM THE COMPUTER AS A "MARK" FOR THE TELETYPE AND A LOGIC "0" AS A "SPACING" CONDITION.

I/O CONNECTORS

THE STANDARD SCELBI-8H CHASSIS IS EQUIPPED WITH 14 I/O SOCKETS. SIX SOCKETS ARE FOR INPUT PORTS AND EIGHT SOCKETS ARE FOR OUTPUT PORTS. THE SOCKETS USED ARE 11 PIN AMPHENOL SERIES 78 CONNECTORS WHICH WILL MATE WITH SERIES 86 MALE PLUGS. I/O CABLES ARE AVAILABLE WHICH CONSIST OF 11 WIRE CABLES WITH A FEMALE CONNECTOR ON ONE END AND A MALE CONNECTOR ON THE OTHER END. STANDARD SCELBI SUPPLIED PERIPHERAL INTER-

FACES ARE EQUIPPED WITH THE MALE AMPHENOL SERIES 86 CONNECTORS. THE USE OF THESE TYPES OF CONNECTORS PROVIDES FOR A SIMPLE, LOW COST, AND RELIABLE METHOD FOR CONNECTING THE COMPUTER TO EXTERNAL DEVICES. THE STANDARD PIN ASSIGNMENTS ON THE I/O CONNECTORS ARE SHOWN BELOW FOR BOTH THE INPUT AND OUTPUT PORT CONNECTORS.

STANDARD INPUT PORT CONNECTOR PIN ASSIGNMENTS

PIN #1	B0
PIN #2	B1
PIN #3	B2
PIN #4	B3
PIN #5	B4
PIN #6	B5
PIN #7	B6
PIN #8	B7
PIN #9	SPARE
PIN #10	SPARE
PIN #11	SIGNAL GROUND

STANDARD OUTPUT PORT CONNECTOR PIN ASSIGNMENTS

PIN #1	B0
PIN #2	B1
PIN #3	B2
PIN #4	B3
PIN #5	B4
PIN #6	B5
PIN #7	B6
PIN #8	B7
PIN #9	STROBE
PIN #10	SPARE
PIN #11	SIGNAL GROUND

IT SHOULD BE NOTED THAT THERE ARE SPARE PIN(S) ON THE INPUT AND OUTPUT PORT CONNECTORS. THESE PINS MAY BE USED TO ROUTE SPECIAL SIGNALS (SUCH AS THE COMPUTER'S "SYNC" SIGNAL WHICH CAN SERVE AS A "CLOCK") FROM THE COMPUTER TO AN EXTERNAL DEVICE.

NOTES ON UTILIZING HARDWARE INTERRUPTS FROM EXTERNAL DEVICES

THE INTERFACES DISCUSSED PREVIOUSLY IN THIS CHAPTER HAVE RELIED ON USING "SOFTWARE" (A PROGRAM IN THE COMPUTER) TO TEST AN EXTERNAL FLIP-FLOP OR SIMILAR TYPE OF FLAG TO DETERMINE WHEN INFORMATION IS READY TO BE SENT TO THE COMPUTER. IN SOME APPLICATIONS IT MAY BE DE-SIRABLE NOT TO USE SOFTWARE TO DETERMINE WHEN A DEVICE HAS NEW INFOR-MATION TO SEND TO THE COMPUTER, BUT INSTEAD, TO "INTERRUPT" THE CPU VIA A HARDWARE SIGNAL. THIS CAN BE ACCOMPLISHED IN A STANDARD SCELBI-8H BY PARALLELING WIRES TO THE "INT" AND "RUN" PUSH BUTTON CHASSIS SWITCH-ES, AND THEN LEAVING THE CHASSIS TOGGLE SWITCHES SET TO, FOR EXAMPLE, A "RST" (RESTART) INSTRUCTION WHICH IS EFFECTIVELY A ONE WORD "CALL" TYPE COMMAND. WHEN THIS IS DONE THE EXTERNAL DEVICE CAN THEN TRIGGER THE "INTERRUPT" FACILITY WHENEVER IT HAS A REQUIREMENT FOR THE COM-PUTER'S ATTENTION. AFTER GENERATING THE "INTERRUPT" SIGNAL THE EXTER-NAL DEVICE SHOULD ISSUE A PULSE ON THE "RUN" LINE. THIS SEQUENCE WILL CAUSE THE COMPUTER TO EXECUTE THE "INTERRUPT" INSTRUCTION THAT IS SET UP ON THE CHASSIS TOGGLE SWITCHES. IF THE SWITCHES ARE SET TO A "RST" TYPE INSTRUCTION THEN A SUBROUTINE IN ONE OF THE RESTART LOCATIONS CAN BE USED TO SERVICE THE DEVICE INITIATING THE INTERRUPT. IT IS POSS-IBLE TO HAVE THE "INTERRUPT" SERVICE ROUTINE DETERMINE WHICH OF A SER-IES OF DEVICES NEEDS TO BE SERVICED (BY LOOKING FOR CONTROL SIGNALS ON AN INPUT PORT.) THE METHOD OF USING HARDWARE GENERATED INTERRUPTS IS OFTEN MORE EFFICIENT IN TERMS OF PROGRAMMING REQUIREMENTS AND CAN PRO-VIDE FASTER RESPONSE TO SERVICING EXTERNAL DEVICES IN MANY APPLICATIONS.

IF AN EVEN MORE SOPHISTICATED HARDWARE INTERRUPT SYSTEM IS REQUIRED, IT IS POSSIBLE TO PARALLEL THE CHASSIS TOGGLE SWITCHES AND HAVE AN EXTERNAL DEVICE SET UP THE DESIRED CODE FOR AN "INTERRUPT" INSTRUCTION. HOWEVER, IN THIS KIND OF APPLICATION ONE MUST TAKE EXTREME CARE TO SET ALL THE CHASSIS TOGGLE SWITCHES TO THE OPEN CONDITION WHEN THE EXTERNAL DEVICE IS CONNECTED IN PARALLEL WITH THE TOGGLE SWITCHES. SINCE SUCH AN APPLICATION IS LIKELY TO BE FOR A DEDICATED OR SPECIAL SERVICE THAT DOES NOT REQUIRE SIGNIFICANT OPERATOR ATTENTION IT WOULD BE ADVISABLE TO HAVE A SPECIAL SCELBI-8H SYSTEM ASSEMBLED THAT WOULD SPECIFICALLY MAXIMIZE THE BASIC CAPABILITY OF THE COMPUTER TO INTERFACE TO AN EXTENSIVE HARD-WARE INTERRUPT SYSTEM.

SUMMARY

THE INFORMATION PRESENTED IN THIS CHAPTER SHOWS HOW EASY IT IS TO CONNECT EXTERNAL DEVICES TO A SCELBI-8H MINI-COMPUTER. MANY USERS WILL WANT TO BUILD THEIR OWN SYSTEM INTERFACES TO CONTROL DEVICES THAT HAVE SPECIFIC AND INDIVIDUAL REQUIREMENTS. SUCH USER'S MAY WANT TO REFER TO THE DETAILED SCHEMATICS AND OTHER DETAILED ENGINEERING DRAWINGS WHICH ARE PROVIDED WITH EACH SCELBI-8H UNIT IN ORDER TO OBTAIN ADDITIONAL INFORMATION ON SUCH POSSIBILITIES AS USING SIGNALS AVAILABLE IN THE COMPUTER TO PROVIDE CLOCK SIGNALS TO EXTERNAL CIRCUITS.

THE READER IS REMINDED THAT INTERFACES FOR MANY COMMONLY USED I/O DEVICES, AND SUPPORTING PROGRAMS, ARE AVAILABLE FROM THE MANUFACTURER OF THE SCELBI-8H MINI-COMPUTER. THESE INCLUDE: AN ASCII KEYBOARD WITH AN INTERFACE, INTERFACES FOR STANDARD MODELS OF TELETYPE MACHINES, BOTH ASCII AND BAUDOT UNITS, AN INTERFACE THAT WILL CONVERT AN OSCILLOSCOPE INTO AN ALPHA-NUMERIC DISPLAY DEVICE (WITHOUT REQUIRING ANY MODIFICATION TO THE OSCILLOSCOPE UNIT), AND AN INTERFACE THAT ENABLES A LOW COST AUDIO CASSETTE TAPE RECORDER TO BE USED TO STORE DATA AND PROGRAMS FROM

THE COMPUTER'S MEMORY AND TO RELOAD DATA OR PROGRAMS BACK INTO MEMORY FROM THE TAPE UNIT (AGAIN THE INTERFACE DOES NOT REQUIRE ANY MODIFITIONS TO THE TAPE RECORDER UNIT).

IN ADDITION TO STANDARD INTERFACES THE MANUFACTURER IS OFTEN ABLE TO SUPPLY SPECIALLY DESIGNED INTERFACES AND PROGRAMS TO USERS WHO DO NOT DESIRE TO DESIGN AND/OR CONSTRUCT THEIR OWN.

MANY USERS, HOWEVER, WILL BE ABLE TO BUILD UP THEIR OWN CUSTOM TAILORED I/O SYSTEMS WITHOUT DIFFICULTY BECAUSE OF THE SIMPLICITY WITH WHICH THE SCELBI-8H MINI-COMPUTER CAN BE INTERFACED TO EXTERNAL EQUIPMENT.