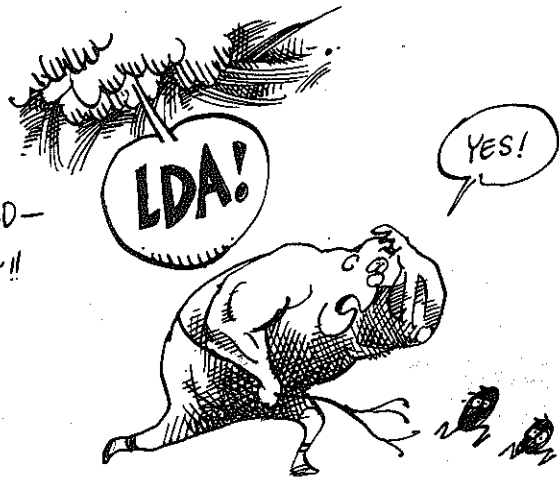


SO YOU SEE,
CONTROL IS
NO TYRANT
AT ALL. IT
ONLY DOES
WHAT IT'S TOLD—
COMPLETELY
AUTOMATICALLY!!



IF YOU REALLY WANT TO IMAGINE THE CONTROL SECTION'S
PERSONALITY, THINK OF A PERFECTLY EFFICIENT
BUREAUCRAT, ACTING IN STRICT OBEDIENCE TO THE
COMPUTER'S REAL BOSS: THE **PROGRAM!**



PART III

SOFTWARE



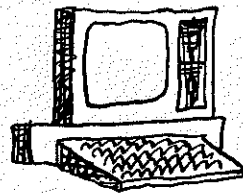
IF PROGRAMS REALLY RULE
THE COMPUTER, THEY DESERVE
A PROPER SCIENTIFIC NAME...
SOMETHING IN GREEK OR
LATIN, PREFERABLY...



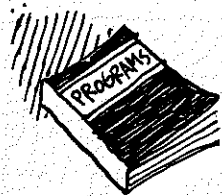
TECHNICALCULUS?
REGULA RATIONOCERUS?
CEPHALONEURALGIA?

* * * * *

BUT THAT'S NOT HOW IT IS IN COMPUTER SCIENCE...
INSTEAD, PROGRAMS IN GENERAL ARE CALLED **SOFTWARE**,
TO DISTINGUISH THEM FROM THE CIRCUIT BOARDS, CATHODE
RAY MONITORS, DISK DRIVES, KEYBOARDS, AND OTHER
ITEMS OF COMPUTER **HARDWARE**.



HARDWARE



SOFTWARE



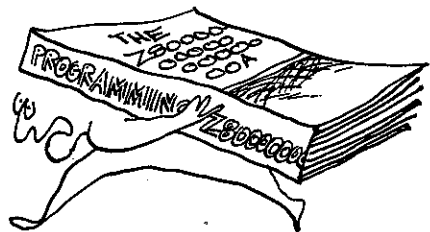
TUPPERWARE

WHAT'S REALLY FUNNY ABOUT THE NAME IS THAT SOFTWARE IS ONE OF THE HARDEST THINGS ABOUT COMPUTING!



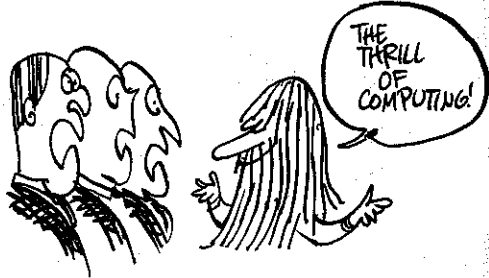
WHILE HARDWARE HAS BEEN DROPPING IN PRICE AND GROWING IN POWER, SOFTWARE ONLY GETS MORE HORRENDOUSLY COMPLEX!

GET ME A HAND TRUCK!



⇒ WE SEE SMALLER AND SMALLER CHIPS WITH BIGGER + BIGGER MANUALS!

IT'S OFTEN IMPOSSIBLE TO ESTIMATE HOW MUCH TIME, MONEY, AND AGONY A GIVEN SOFTWARE PROBLEM WILL COST TO SOLVE... WHAT A WAY TO RUN A BUSINESS!



LIKEWISE THERE'S A DIFFERENCE BETWEEN THE IMAGE OF HARDWARE AND SOFTWARE WORKERS —



HARDWARE TYPES ARE ENGINEERS... INTO GADGETS... MOSTLY MEN... BOUND BY THE LAWS OF PHYSICS...

PROGRAMMERS HAVE NO TOOL BUT THEIR BRAINS... THEY'RE MORE OFTEN WOMEN... SUPPOSED TO BE SOLITARY DREAMERS WHOSE IDEAS HAVE NOTHING TO DO WITH THE LAWS OF PHYSICS!!



PROGRAMS THESE DAYS ARE SO COMPLEX THAT NO ONE PERSON CAN UNDERSTAND THEM — SO THESE LONERS HAVE TO WORK IN TEAMS — A SPECTACLE I LEAVE TO THE READER'S IMAGINATION...



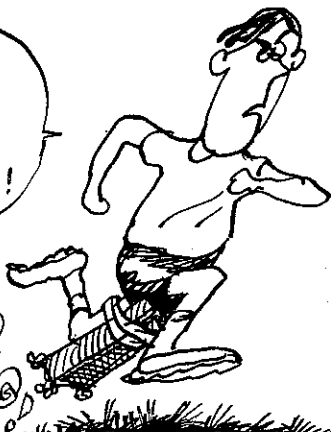
WHILE ADA LOVELACE WAS THE ORIGINAL PROGRAMMER, THE FIRST PERSON TO PROVE THE FULL POWER OF SOFTWARE WAS

ALAN TURING
(1912-1954)



TURING, WHO ENJOYED LONG-DISTANCE RUNNING BACK WHEN THAT WAS CONSIDERED WEIRD, PROBABLY WENT INTO COMPUTERS TO SHRINK THE SIZE OF HIS JOGGING CLOCK.

IT'S MAKING MY STRIDE LOPSIDED!



IN 1936 HE DREAMED UP THE TURING MACHINE...

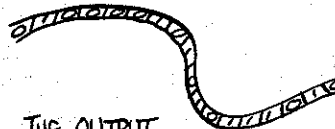
BONG!
BONG!

TURING MACHINES AREN'T REAL MACHINES... THEY'RE ABSTRACT MACHINES, EXISTING ONLY IN THEORY...



A SOFTWARE ENGINEER'S DREAM — NO HARDWARE!

ROUGHLY SPEAKING, A TURING MACHINE IS AN INPUT-OUTPUT DEVICE: A BLACK BOX THAT READS A SEQUENCE OF 0'S AND 1'S.



THE OUTPUT DEPENDS ONLY ON THE PRESENT INPUT (0 OR 1) AND THE PREVIOUS OUTPUT.

THE NATURE OF THE OUTPUT IS UNIMPORTANT.

THE MAIN THING IS THAT THE CHANGES FROM ONE OUTPUT STATE TO THE NEXT ARE GIVEN BY DEFINITE RULES, CALLED THE **TRANSITION RULES**.

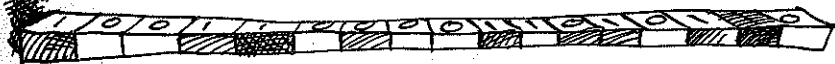
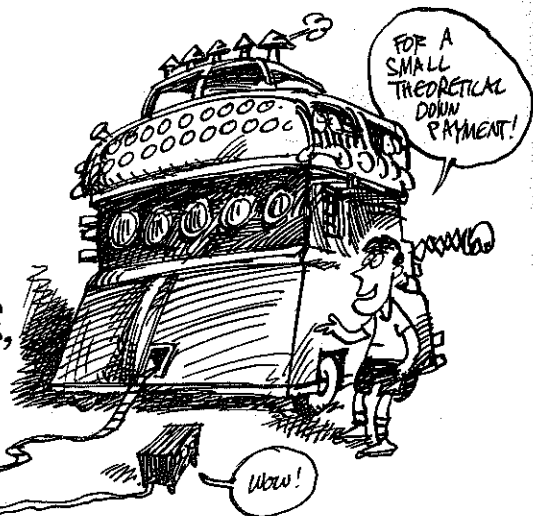
THE REASON TURING MACHINES ARE IMPORTANT IS THAT THEY ARE A WAY OF THINKING PHYSICALLY ABOUT LOGIC. ANY WELL-DEFINED, STEP-BY-STEP LOGICAL PROCEDURE CAN BE EMBODIED IN SOME TURING MACHINE.



THERE'S A TURING MACHINE THAT CAN ADD!

*FOR DETAILS, SEE J. WEIZENBUM'S *COMPUTER POWER AND HUMAN REASON*, CHAPTER 2.

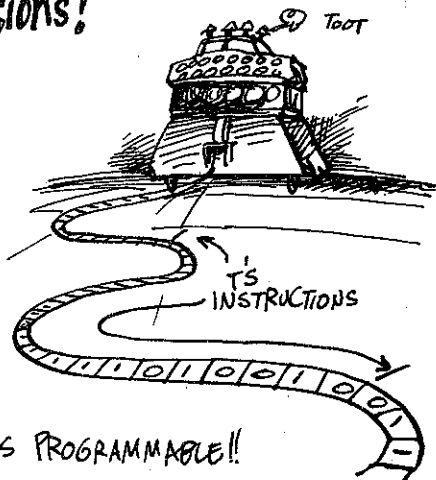
WHAT TURING PROVED:
IT'S THEORETICALLY
POSSIBLE TO
CONSTRUCT A
SINGLE TURING
MACHINE, THE
**UNIVERSAL
TURING MACHINE,**
WHICH CAN IMITATE
ALL OTHER
TURING MACHINES!!!



THE TRICK IS THAT THE UNIVERSAL TURING MACHINE CAN...

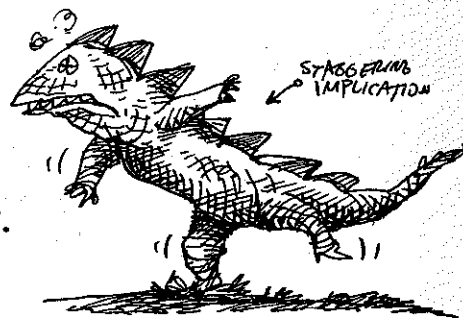
read instructions!

THAT IS, TO MAKE THE
UNIVERSAL TURING MACHINE
(U) ACT LIKE MACHINE T,
YOU ENCODE T'S
TRANSITION RULES ONTO
U'S TAPE. AT EACH
STEP, U OBSERVES ITS
OWN INPUT, THEN
REFERS TO T'S
TRANSITION RULES TO
SEE WHAT TO DO.



⇒ IN OTHER WORDS, U IS PROGRAMMABLE!!

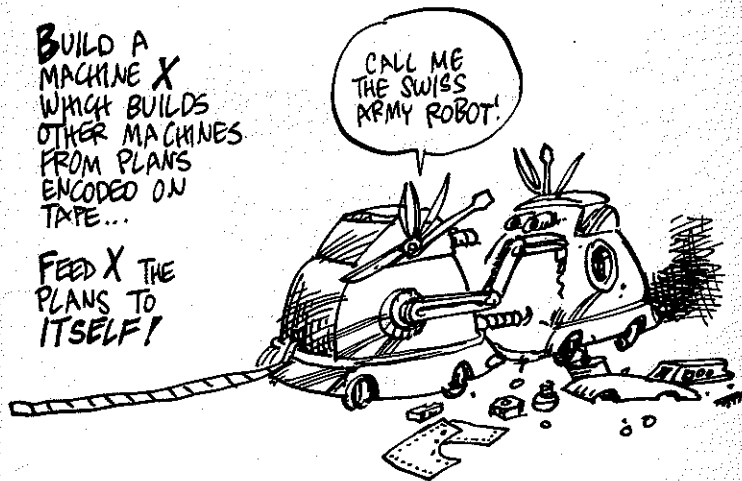
THE IMPLICATIONS
ARE STAGGERING:
A SINGLE,
PROGRAMMABLE
MACHINE CAN
PERFORM ANY
WELL-DEFINED,
STEP-BY-STEP
LOGICAL PROCEDURE.
(REMEMBER, TURING
SAW THIS TEN YEARS
BEFORE A REAL
COMPUTER WAS BUILT.)



JOHN VON NEUMANN CARRIED TURING'S IDEAS A STEP
FURTHER. VON NEUMANN REALIZED THAT ONE COULD:

BUILD A
MACHINE X
WHICH BUILDS
OTHER MACHINES
FROM PLANS
ENCODED ON
TAPE...

FEED X THE
PLANS TO
ITSELF!



⇒ **SELF-REPRODUCING
MACHINES ARE POSSIBLE!!**

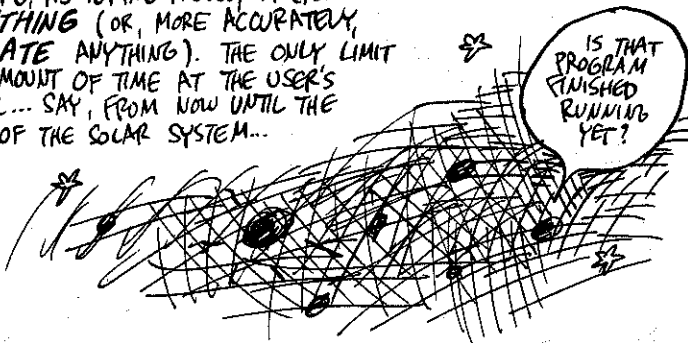
THE DIGITAL COMPUTER IS A FANCY UNIVERSAL TURING MACHINE COME TO LIFE.

IF YOU CALL THAT "LIFE..."



THEREFORE, AS TURING PROVED, IT CAN DO ANYTHING (OR, MORE ACCURATELY, SIMULATE ANYTHING). THE ONLY LIMIT IS THE AMOUNT OF TIME AT THE USER'S DISPOSAL... SAY, FROM NOW UNTIL THE DEATH OF THE SOLAR SYSTEM...

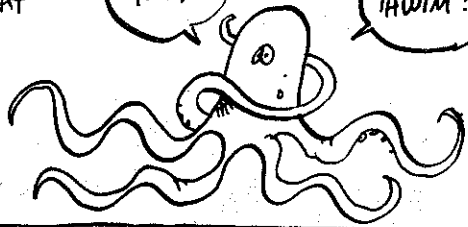
IS THAT PROGRAM FINISHED RUNNING YET?



TO BE PERFECTLY HONEST, THERE ARE A COUPLE OF OTHER QUALIFICATIONS ON THAT "ANYTHING." WHAT KIND OF "ANYTHING" CAN A COMPUTER DO?

CAN IT THINK?

CAN IT TALK?



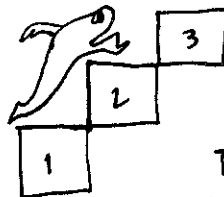
IN A WORD, COMPUTERS DO

ALGORITHMS

FROM AL-KHWARIZMI, REMEMBER?

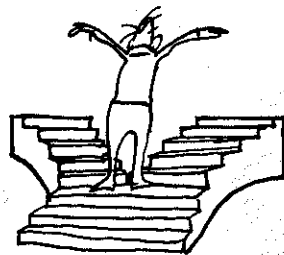


AN ALGORITHM IS SIMPLY ANY WELL DEFINED, STEP-BY-STEP PROCEDURE: A RECIPE, IF YOU WILL!



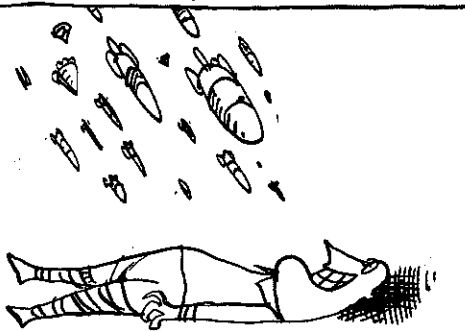
STEP-BY-STEP MEANING EACH STEP IS COMPLETED BEFORE THE NEXT IS BEGUN.

WELL DEFINED, MEANING EACH STEP IS COMPLETELY DETERMINED BY CURRENT INPUT AND THE RESULTS OF PREVIOUS STEPS. NO AMBIGUITY ALLOWED!



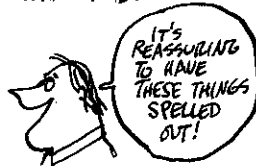
EXAMPLES OF ALGORITHMS:

"IF NUCLEAR WARHEADS ARE FALLING LIKE HAILSTONES, I WILL LIE DOWN AND TRY TO ENJOY IT. OTHERWISE, I WILL GO TO WORK AS USUAL."



IT'S AN ALGORITHM BECAUSE I ALWAYS KNOW WHAT TO DO:

1. CHECK TO SEE IF WARHEADS ARE FALLING
2. IF YES, LIE DOWN + ENJOY!
3. IF NO, GO TO WORK.



LIKEWISE, ALGEBRAIC FORMULAS REPRESENT ALGORITHMS

$y = x^2 + 2x + 10$ MEANS -

- (1) INPUT A NUMBER x
- (2) MULTIPLY x TIMES ITSELF
- (3) MULTIPLY x TIMES 2
- (4) ADD THE RESULTS OF (2) AND (3)
- (5) ADD 10 TO THE RESULT OF (4)

IF YOU UNDERSTAND, LIE DOWN AND ENJOY YOURSELF!



EXAMPLES OF NON-ALGORITHMS:

"IF NUCLEAR WARHEADS ARE FALLING LIKE HAILSTONES, LIE DOWN AND TRY TO ENJOY IT."



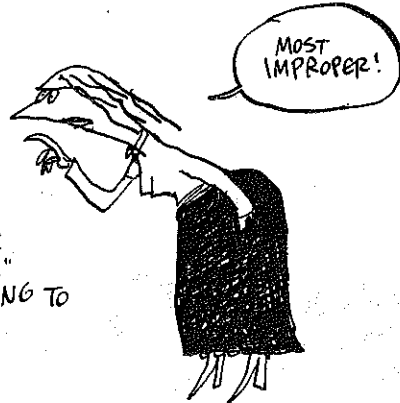
THIS FAILS TO TELL YOU WHAT TO DO IF NO WARHEADS ARE FALLING... SO IT'S NOT WELL DEFINED.

ANOTHER?

HOW ABOUT

$y = x^2 ++ 2x - 10$?

THIS IS NO ALGORITHM BECAUSE IT'S NOT EXPRESSED IN PROPER "ALGEBRAIC GRAMMAR." WE ASSIGN NO MEANING TO THE SYMBOLS "++".



IF YOU TRY TO MAKE A COMPUTER DO A NON-ALGORITHM, IT WILL JUST SIT THERE FLASHING ERROR MESSAGES!

SOME STANDARD SYMBOLS ARE USED TO MAKE ALGORITHMS EASIER TO FOLLOW. EACH STEP IS REPRESENTED BY A SPECIALLY SHAPED — WHAT ELSE? — BOX. THE SHAPE INDICATES WHAT TYPE OF STEP IS TO BE EXECUTED:

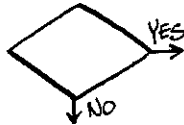
THERE ARE 4 POSSIBILITIES:



BEGIN OR END



PERFORM A PROCEDURE (ADD, SUBTRACT, ETC)



CONDITIONAL BRANCH



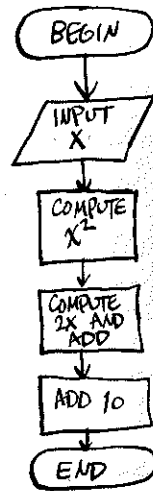
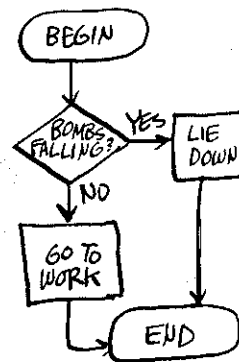
INPUT OR OUTPUT

THE "FLOW" OF THE ALGORITHM IS REPRESENTED BY ARROWS \rightarrow , AND WHEN ALL THE SYMBOLS ARE COMBINED, IT'S A

FLOW CHART

Go WITH THE FLOW...

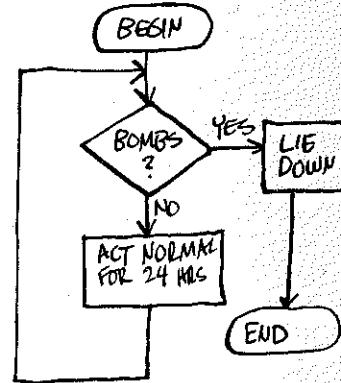
HERE ARE THE FLOW CHARTS OF THE ALGORITHMS FROM A COUPLE OF PAGES BACK:



IN BOTH ALGORITHMS, THE FLOW PROCEEDS IN ONE DIRECTION, FROM START TO FINISH.

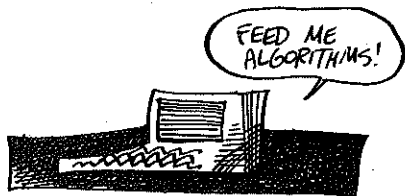
IT'S ALSO POSSIBLE FOR THE FLOW OF ALGORITHMS TO JUMP FORWARD OR BACKWARD. FOR EXAMPLE, LET'S REWRITE THAT FIRST ALGORITHM:

1. IF BOMBS ARE FALLING, GO TO STEP 2. OTHERWISE, GO TO STEP 4.
2. LIE DOWN AND ENJOY!
3. GO TO STEP 6.
4. LEAD A NORMAL LIFE FOR 24 HOURS
5. GO TO STEP 1
6. END



YOU MAY FIND THE FLOW CHART EASIER TO GRASP THAN THE WRITTEN "PROGRAM." NOTE THAT IT MAY CONTINUE INDEFINITELY!!

FLOW CHARTS ARE USEFUL
IN HELPING TO DESIGN
ALGORITHMS — SIMPLE ONES,
ANYWAY — AND DESIGNING
ALGORITHMS IS WHAT
COMPUTER PROGRAMMING
IS ALL ABOUT !!



THE FIRST STEP IN WRITING ANY PROGRAM IS TO
ANALYZE THE JOB TO BE DONE, AND SEE
HOW TO DO IT ALGORITHMICALLY!

FAILURE TO
THINK
ALGORITHMICALLY
HAS CAUSED
MANY
SOFTWARE
NIGHTMARES !!
MOST
SOFTWARE
DESIGNERS
HAVE
HORROR STORIES
ABOUT
CUSTOMERS
WHO DIDN'T
KNOW
EXACTLY
WHAT THEY
WANTED !!

YES...
THAT INFORMATION
SHOULD GO INTO
MY FILES... OR
MAYBE NOT... THE
VICE PRESIDENT'S ARE
JUST AS GOOD...
OR MAYBE THE
TREASURER'S...

NO!
NO!

LET'S TRY A COUPLE MORE EXAMPLES...
A LITTLE MORE LIKE WHAT A COMPUTER
MIGHT ACTUALLY BE ASKED TO DO...

"ROOMMATE RECEIPTS"

TWO ROOMMATES, LISA AND SOPHIE,
SHARE THEIR MEALS. THEY
BOTH SHOP FOR FOOD AND SAVE
THEIR RECEIPTS. AT THE END OF
THE MONTH, THEY WANT TO
KNOW WHO OWES WHOM HOW
MUCH.

"MULTIPLE PLUG-INS"

THIS ONE ASKS THE
COMPUTER TO
EVALUATE THE
EXPRESSION

$$x^2 + 2x + 10$$

NOT JUST AT ONE
VALUE OF X, BUT
FOR MANY VALUES,
NAMELY

X=0, 0.1, 0.2, 0.3,
... AND SO ON...
UP TO 2.0.



FOR "ROOMMATE RECEIPTS" WE REASON LIKE SO:

LET S = SOPHIE'S EXPENSES
 L = LISA'S EXPENSES

THEN THE TOTAL EXPENSE IS $S+L$, AND EACH ROOMMATE'S SHARE IS

$$\frac{1}{2}(S+L).$$

IF LISA OUTSPENT SOPHIE, SO $L > S^*$, THEN SOPHIE OWES LISA

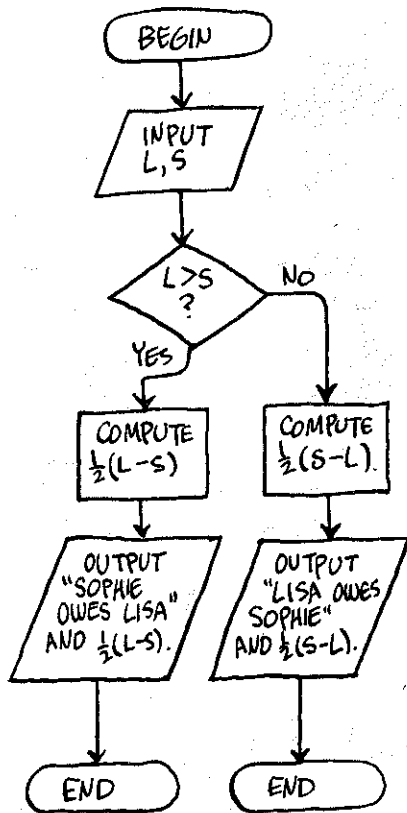
$$\frac{1}{2}(S+L) - S, \text{ OR}$$

$$\frac{1}{2}(L-S).$$

OTHERWISE (WHEN $S \geq L^*$), LISA OWES SOPHIE

$$\frac{1}{2}(S-L).$$

THE ALGORITHM'S OUTPUT IS TO TELL US WHO OWES WHOM AND HOW MUCH.



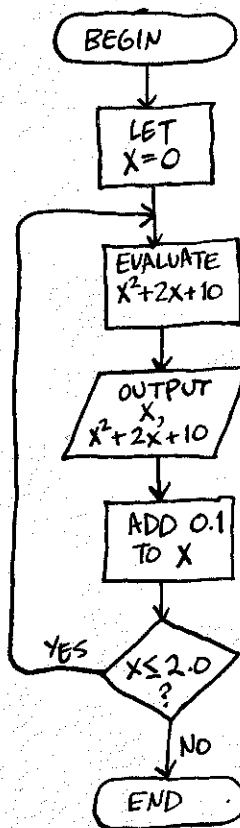
* $>$ MEANS "IS GREATER THAN"; \geq MEANS "IS GREATER THAN OR EQUAL TO";
 $<$ MEANS "IS LESS THAN"; \leq MEANS "IS LESS THAN OR EQUAL TO".

IN "MULTIPLE PLUG-INS," WE WANT TO EVALUATE A SINGLE EXPRESSION, $x^2 + 2x + 10$, REPEATEDLY AT DIFFERENT VALUES OF x (NAMELY 0.0, 0.1, 0.2, ..., 1.9, 2.0)

THE CORE OF THE ALGORITHM WILL BE THIS LOOP:

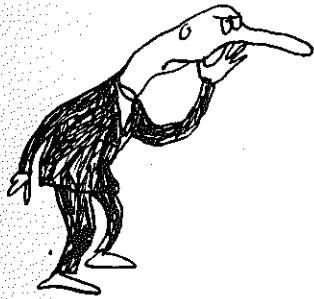
1. PLUG THE CURRENT VALUE OF x INTO $x^2 + 2x + 10$
2. PRINT THE RESULT
3. NEXT x
4. RETURN TO STEP 1.

WE ALSO HAVE TO SPECIFY WHAT x TO START WITH, WHEN TO STOP, AND HOW TO COMPUTE "NEXT x ."



NOTE HOW THE FLOW CHART SHOWS HOW THE PROGRAM LOOPS BACK, PLUGGING IN SUCCESSIVE VALUES OF x UNTIL x EXCEEDS 2.

NOW THE #738 QUESTION:
(\$64 AFTER INFLATION):



HOW DO YOU WRITE
AN ALGORITHM
THAT'S
INTELLIGIBLE TO
A COMPUTER?

HUH?



IN OTHER WORDS, HOW DO YOU PROGRAM A COMPUTER?

UNFORTUNATELY,
YOU HAVE TO
SPEAK THE
COMPUTER'S
LANGUAGE—
BECAUSE THE
COMPUTER IS
STILL TOO
STUPID TO
UNDERSTAND
YOURS!



IT MAY BE
FAST, BUT
IT'S THICK!

WHAT LANGUAGE DOES THE COMPUTER
UNDERSTAND?

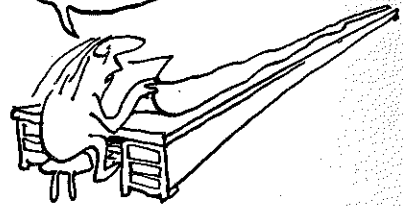
AT THE VERY BEGINNING, PROGRAMMERS WROTE DIRECTLY
IN "MACHINE LANGUAGE"—BINARY CODE. THIS WAS
OBVIOUSLY A HEADACHE!

WE'LL NEED
A COMPUTER
JUST TO FIGURE
OUR ASPIRIN
BILL!



SOON THEY
SWITCHED TO
ASSEMBLY LANGUAGE
(SEE P. 174), AIDED
BY AUTOMATIC
"ASSEMBLERS,"
WHICH TRANSLATED
ASSEMBLY LANGUAGE
MNEMONICS INTO
MACHINE CODE.
STILL SOMETHING MORE
WAS NEEDED!

YES...
MILE-LONG
DESKS...



AND
FINALLY,

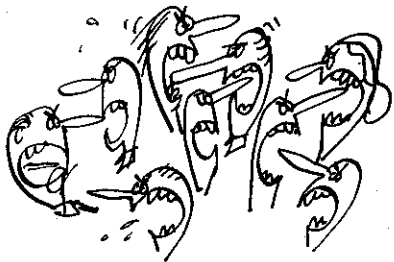
THE **HIGHER-LEVEL** PROGRAMMING LANGUAGES
WERE INVENTED. THESE CONTAIN
FAMILIAR ENGLISH-LIKE COMMANDS,
SUCH AS "PRINT," "READ," AND "DO,"
WHICH ARE TRANSLATED INTO MACHINE
LANGUAGE BY COMPLEX PROGRAMS
CALLED COMPILERS OR INTERPRETERS.
HIGHER-LEVEL PROGRAMS ARE
SOMETIMES CALLED "SOURCE CODE,"
AND THE MACHINE-LANGUAGE TRANSLATION
IS CALLED "OBJECT CODE."

SOURCE CODE

↓
COMPILER OR
INTERPRETER

↓
OBJECT CODE

THE FIRST HIGHER-LEVEL LANGUAGE WAS FORTRAN ("FORMULA TRANSLATOR"), WHICH MADE ITS DEBUT IN THE EARLY 1950'S. SINCE THEN, LITERALLY HUNDREDS OF LANGUAGES HAVE BEEN WRITTEN, EACH WITH ITS OWN ARMY OF RABID DEVOTEES!

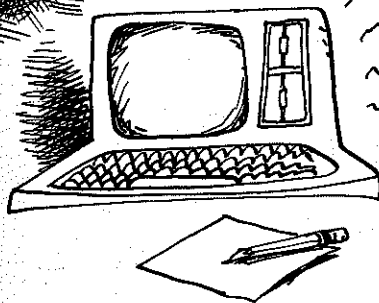


WE'RE GOING TO TAKE A QUICK LOOK AT BASIC — BEGINNER'S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE. BASIC IS EASY TO LEARN AND WIDELY USED, DESPITE CRITICISM (ESPECIALLY BY PASCAL ADMIRERS) THAT IT PROMOTES "BAD PROGRAMMING HABITS."



WITH APOLOGIES TO PASCAL, THEN, HERE'S A LITTLE BASIC...

BASIC BASIC



THERE ARE TWO WAYS TO WRITE A BASIC PROGRAM: WITH PENCIL AND PAPER, OR DIRECTLY AT THE COMPUTER.

IT'S GOOD PRACTICE TO PLAN PROGRAMS ON PAPER FIRST, TO WORK OUT THE ESSENTIAL IDEAS AND STRUCTURE, BUT EVENTUALLY YOU MUST SIT DOWN AT THAT KEYBOARD!



SOME MACHINES ARE READY FOR BASIC AS SOON AS YOU TURN THEM ON. OTHERS ONLY BRING IT UP ON COMMAND. IF IN DOUBT, ASK!



WHEN THE COMPUTER IS READY, IT GIVES YOU A "PROMPT" OF SOME KIND: THE WORD "READY" OR JUST THE SIGN ">".



THE COMPUTER KEYBOARD RESEMBLES A STANDARD TYPEWRITER'S "QWERTY" KEYBOARD... EXCEPT THAT AS YOU TYPE, CHARACTERS APPEAR ON THE CRT (CATHODE RAY TUBE) SCREEN, INSTEAD OF ON PAPER. TO GO TO THE NEXT LINE, HIT THE **RETURN** (Z) KEY. HERE'S A SIMPLE BASIC PROGRAM:

```
10 REM BASIC MULTIPLICATION
20 READ A, B
30 DATA 5.6, 1.1
40 LET C=A*B
50 PRINT "THE PRODUCT IS"; C
60 END
```

BASIC MATH:
 $A+B$ } AS USUAL
 $A-B$ }
 $A*B$... A TIMES B
 A/B ... A DIVIDED BY B
 A^B ... A TO THE BTH POWER

THE PROGRAM IS NOW STORED IN MEMORY. TO RUN IT, TYPE "RUN", FOLLOWED BY THE RETURN KEY. THE SCREEN DISPLAYS:

```
RUN
THE PRODUCT IS 6.16
```



A FEW POINTS TO NOTE:



> EVERY LINE BEGINS WITH A **LINE NUMBER** (10, 20, ...). EVERY LINE OF A BASIC PROGRAM MUST HAVE A NUMBER! IT'S WISE TO COUNT BY TENS, SO YOU CAN INSERT LINES LATER.

> THE FIRST LINE (10) IS A **REMARK**. REMARKS EXPLAIN THE PROGRAM BUT AREN'T EXECUTED BY THE COMPUTER. THE PREFIX "REM" IDENTIFIES REMARKS. WE MIGHT INSERT ONE HERE:

```
20 READ A, B
25 REM THESE ARE THE #S TO BE MULT'D
30 DATA 5.6, 1.1
```

> PROGRAM **STATEMENTS** CONSIST OF INSTRUCTIONS ("LET", ETC), NUMBERS (5.6, 1.1), VARIABLES (A, B, C), **TEXT** ("THE PRODUCT IS"), AND PUNCTUATION.

```
50 PRINT "THE PRODUCT IS"; C
```

↑ QUOTES ↑ SPACES ↑ SEMICOLON

> EACH OF THESE HAS A PRECISE MEANING!

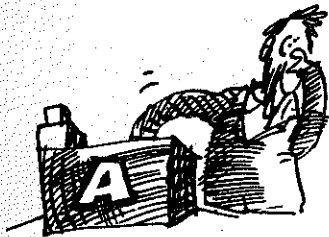
(NUMERICAL) VARIABLES

THINK OF A VARIABLE AS A LABELED BOX IN MEMORY!

A NUMERICAL VARIABLE IN BASIC IS LIKE A VARIABLE IN ALGEBRA. IT ASSUMES A NUMERICAL VALUE, WHICH MAY VARY (BUT IT HAS ONLY ONE VALUE AT A TIME!). ONLY THESE SYMBOLS CAN BE USED AS VARIABLES:

A, B, C, D, Z
A0, B0, Z0
A1, B1, Z1
...	...
A9, B9, Z9

AND EVERYTHING IN BETWEEN!



THERE ARE SEVERAL WAYS TO ASSIGN A VALUE TO A VARIABLE. ONE IS THE **READ = DATA** STATEMENT:

```
20 READ A, B
30 DATA 5.6, 1.1
```

COMMAS ARE ESSENTIAL!

THIS INSTRUCTS THE COMPUTER TO ASSIGN THE NUMERICAL VALUES IN THE **DATA** STATEMENT - IN ORDER - TO THE VARIABLES IN THE **READ** STATEMENT.

```
20 READ A, B, C
30 DATA 5.6, 1.1
```

THIS IS A BUG!

DISGUSTING!



ANOTHER WAY TO ASSIGN VALUES TO VARIABLES IS WITH

LET.

```
10 LET Q = 6.5
20 LET R = 2 * Q
30 LET S = Q * 2 + R + 10
```

MAKES R = 13

MAKES S = (6.5) * 2 + 13 + 10 = 65.25

THE LET STATEMENT ASSIGNS THE VALUE ON THE RIGHT OF THE EQUALITY SIGN, "=", TO THE VARIABLE ON THE LEFT. THE RIGHT-HAND SIDE MAY BE A NUMBER, OR SOME MATHEMATICAL EXPRESSION INVOLVING OTHER VARIABLES - AS LONG AS THEY ALREADY HAVE VALUES!!

```
10 LET Q = 6.5
20 LET Q = 0.5 * R
30 LET S = Q * 2 + R + 10
```



HERE STATEMENT 20 DOES NOT ASSIGN ANY VALUE TO R, BECAUSE R IS NOT ON THE LEFT SIDE OF "=". IN FACT, IF R HASN'T BEEN ASSIGNED SOME VALUE EARLIER IN THE PROGRAM, THEN STATEMENT 20 GIVES Q AN INDETERMINATE VALUE! BUT -

```
10 LET M = 0
20 LET M = M + 1
30 LET M = M + 1
```

MAKES M = 1

MAKES M = 2

THESE STRANGE-LOOKING STATEMENTS ARE PERFECTLY O.K! "LET M = M + 1" MEANS "ASSIGN TO THE VARIABLE M A VALUE EQUAL TO ITS CURRENT VALUE PLUS 1."

PRINT

THIS IS AN OUTPUT COMMAND, MEANING "DISPLAY ON THE SCREEN," NOT "PRINT ON PAPER."

WHAT CAN BE
PRINTED?



YOU CAN PRINT ANY TEXT:

```
10 PRINT "ANY NUKES TODAY?"  
RUN  
ANY NUKES TODAY?
```

QUOTATION MARKS
ESSENTIAL!

QUOTATION MARKS
REMOVED

PRINT A VARIABLE AND YOU GET
ITS VALUE:

```
10 LET X=77001  
20 PRINT X  
RUN  
77001
```

BUT—

```
10 LET X=77001  
20 PRINT "X"  
RUN  
X
```

QUOTATION MARKS MAKE
THE COMPUTER TREAT X
AS A TEXT.

PRINT A MATHEMATICAL EXPRESSION AND YOU GET ITS VALUE.

```
10 LET Z=1.5  
20 PRINT Z↑2 + 2*Z + 10  
RUN  
15.25
```

BECAUSE
 $(1.5)^2 + 2 \times 1.5 + 10$
 $= 2.25 + 3.0 + 10 = 15.25$

SEMICOLON (;)

A SEMICOLON AFTER A PRINT STATEMENT CAUSES THE
NEXT PRINT STATEMENT TO DISPLAY ITS OUTPUT ON THE
SAME LINE AND DIRECTLY AFTER THE FIRST ONE'S:

```
10 LET A=1  
20 PRINT "INFINITY IS MORE THAN";  
30 PRINT A  
RUN  
INFINITY IS MORE THAN 1
```

IT'S O.K. TO ABBREVIATE THIS:

```
10 LET A=1  
20 PRINT "INFINITY IS MORE THAN"; A  
RUN  
INFINITY IS MORE THAN 1
```

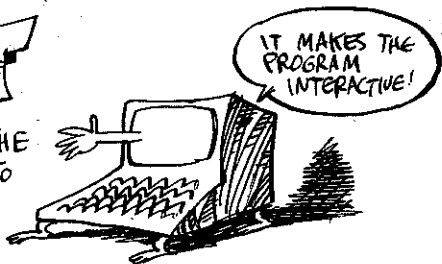
FOR EXAMPLE, WE COULD REWRITE THE PROGRAM ON P. 208.

```
10 REM BASIC MULTIPLICATION  
20 READ A, B  
30 DATA 5.6, 1.1  
40 LET C=A*B  
50 PRINT "THE PRODUCT OF"; A; "AND"; B; "IS"; C; "."  
60 END  
RUN  
THE PRODUCT OF 5.6 AND 1.1 IS 6.16.
```

➤ THERE ARE ALSO SOME NIFTY TRICKS USING THE **COMMA**
AND PRINT, BUT WE WON'T GET INTO IT...

INPUT

THIS STATEMENT ALLOWS THE USER TO ASSIGN VALUES TO VARIABLES WHILE THE PROGRAM IS RUNNING.



THE FORM OF THE STATEMENT:

INPUT A

WHEN THE PROGRAM RUNS AND REACHES AN INPUT STATEMENT, THE SCREEN DISPLAYS:

?

THIS INDICATES THAT THE PROGRAM HAS HALTED, WANTING INPUT. YOU TYPE SOME NUMBER (FOLLOWED BY "RETURN," AS ALWAYS!).

5.6

AND THE PROGRAM CONTINUES RUNNING.

"INPUT" AND "PRINT" CAN BE USED IN COMBINATION TO LET YOU KNOW WHAT SORT OF INPUT IS EXPECTED:

```
10 BASIC DIVISION
20 PRINT "TYPE THE NUMERATOR."
30 INPUT N
40 PRINT "TYPE THE NON-ZERO DENOMINATOR."
50 INPUT D
60 PRINT N; "/"; D; "="; N/D
70 END
```

RUN
TYPE THE NUMERATOR.

? 5
TYPE THE NON-ZERO DENOMINATOR.

? 8
5/8 = 0.625

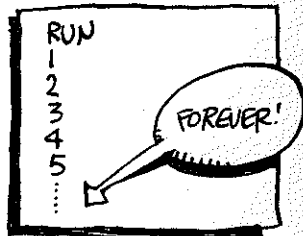
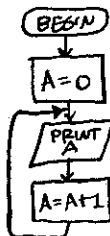
TYPED BY THE USER.

GO TO

THIS IS THE UNCONDITIONAL BRANCHING INSTRUCTION.

"GO TO (LINE NUMBER)" TRANSFERS CONTROL TO A LINE OTHER THAN THE NEXT. THE PROGRAM THEN CONTINUES FROM THERE, AS IN THIS ENDLESS LOOP:

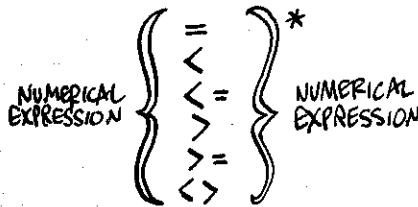
```
10 LET A=0
20 PRINT A
30 LET A=A+1
40 GO TO 20
```



IF-THEN

IS THE "SMART," CONDITIONAL JUMP.

IT HAS THE GENERAL FORM IF (CONDITION) THEN (LINE NUMBER). THE CONDITION HAS THE FORM:



AS IN **IF A <= B THEN 30**

THIS ALWAYS INCLUDES THE UNSTATED INSTRUCTION, "OTHERWISE, GO TO THE NEXT LINE."

* < LESS THAN, <= LESS THAN OR EQUAL TO, > GREATER THAN, >= GREATER THAN OR EQUAL TO, <> DOES NOT EQUAL.

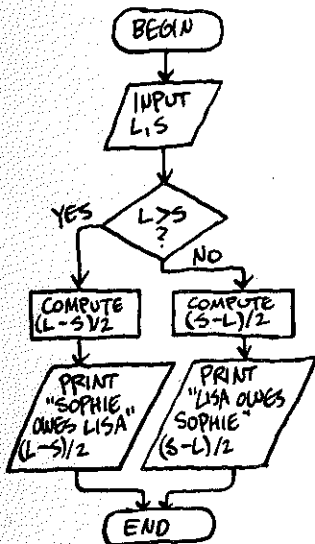
```
10 LET A=0
20 PRINT A
30 LET A=A+1
40 IF A<=2 THEN 20
50 END
RUN
0
1
2
```

"OTHERWISE, NEXT LINE!"

THIS IS ENOUGH TO WRITE BASIC PROGRAMS FOR THE TWO ALGORITHMS FROM P. 201:

ROOMMATE RECEIPTS

THE FLOW CHART:



THE PROGRAM:

```

10 PRINT "LISA SPENT"
20 INPUT L
30 PRINT "SOPHIE SPENT"
40 INPUT S
50 IF L > S THEN 80
60 PRINT "LISA OWES SOPHIE"; (S-L)/2
70 GO TO 90
80 PRINT "SOPHIE OWES LISA"; (L-S)/2
90 END
  
```

SEE HOW "IF-THEN" AND "GO TO" ARE USED? IF $L > S$, THEN LINES 60 AND 70 ARE NOT EXECUTED. OTHERWISE, THEY ARE EXECUTED, AND LINE 70 ENSURES THAT LINE 80 IS SKIPPED.

IF THE PROGRAM IS RUN:

```

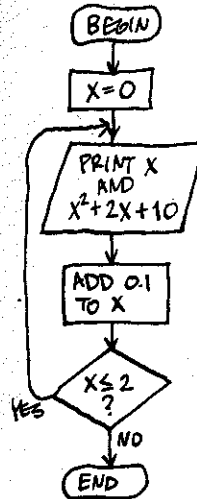
RUN
LISA SPENT
? 93.75
SOPHIE SPENT
? 77.38
SOPHIE OWES LISA 8.185
  
```

NOW WE NEED A PROGRAM TO ROUND OFF THE HALF PENNY!



MULTIPLE PLUG-LINES

THE FLOW CHART:



THE PROGRAM:

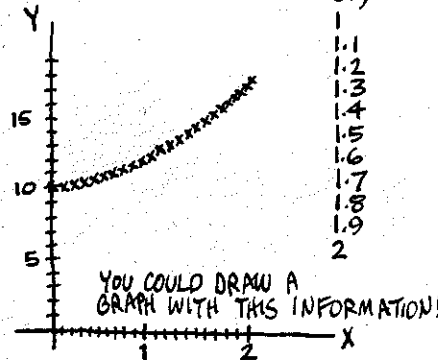
```

10 REM LINE 20 PRINTS A HEADING
20 PRINT "X      Y"
30 LET X=0
40 LET Y=X^2+2*X+10
50 PRINT X;"      ";Y
60 LET X=X+0.1
70 IF X<=2 THEN 40
80 END
  
```

RUN

X	Y
0	10.21
0.1	10.44
0.2	10.69
0.3	10.96
0.4	11.25
0.5	11.56
0.6	11.89
0.7	12.24
0.8	12.61
0.9	13
1	13.41
1.1	13.84
1.2	14.29
1.3	14.76
1.4	15.25
1.5	15.76
1.6	16.29
1.7	16.84
1.8	17.41
1.9	18
2	

GAPS A RESULT OF NOT USING ALL OF BASIC'S FORMATTING ABILITIES!



THE "MULTIPLE PLUG-INS" LOOP IS SO TYPICAL THAT ALL PROGRAMMING LANGUAGES HAVE SPECIAL COMMANDS JUST FOR SUCH REPETITIONS. IN BASIC, IT'S →

FOR NEXT

THIS REPLACES THESE THREE LINES:

```
30 LET X=0
...
60 LET X=X+0.1
70 IF X<=2 THEN 30
...
```

WITH THESE TWO:

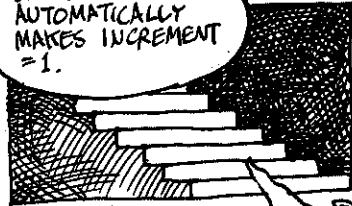
```
30 FOR X=0 TO 2 STEP 0.1
...
60 NEXT X
```

THE STATEMENT INITIALLY SETS THE VARIABLE EQUAL TO THE LOWER LIMIT, EXECUTES THE LINES UP TO "NEXT," INCREMENTS THE VARIABLE BY THE AMOUNT "STEP," AND REPEATS THE LOOP UNTIL THE UPPER LIMIT IS EXCEEDED.

A SIMPLE EXAMPLE:

```
10 FOR I=1 TO 4
20 PRINT I*I
30 NEXT I
40 END
RUN
1
4
9
16
```

OMITTING "STEP" AUTOMATICALLY MAKES INCREMENT = 1.



NEXT PAGE!

PROBLEMS

PROBLEMS?
WHO HAS PROBLEMS?



1. WHAT DOES THIS PROGRAM DO?

```
10 INPUT N
20 FOR I=1 TO N
30 PRINT I*I
40 NEXT I
50 END
```

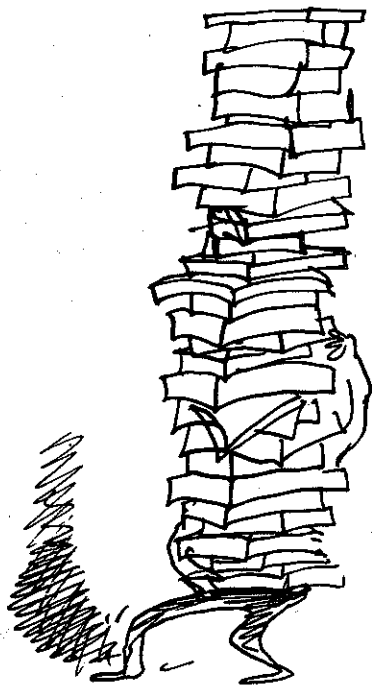
2. REWRITE THE "MULTIPLE PLUG-INS" PROGRAM USING THE "FOR NEXT" STATEMENT.

3. WRITE A PROGRAM WHICH ADDS THE INTEGERS (WHOLE NUMBERS) FROM 1 TO 1,000,000. DITTO FROM 1 TO N, FOR ANY N.

4. IN THE FIBONACCI SEQUENCE 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... EACH NUMBER IS THE SUM OF THE PREVIOUS TWO NUMBERS. WRITE A PROGRAM WHICH GENERATES THIS SEQUENCE.

5. READ ENOUGH OF A BASIC TEXTBOOK TO WRITE A "ROOMMATE RECEIPTS" PROGRAM FOR ANY NUMBER OF ROOMMATES.

THERE ARE PLENTY OF OTHER BASIC FEATURES, ENOUGH TO FILL ENTIRE BOOKS — AND IN FACT TONS OF BOOKS ON BASIC HAVE BEEN PUBLISHED.



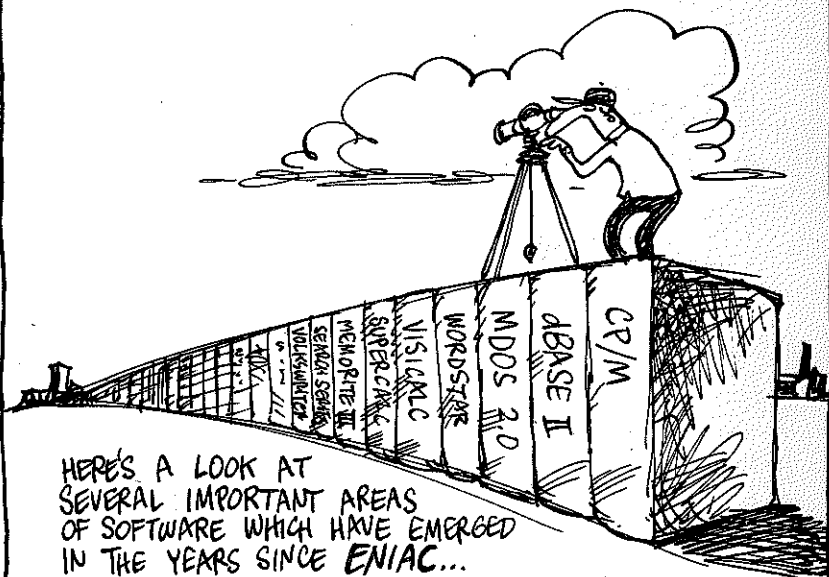
BASIC
BUILDS
BICEPS!

SO... IF YOU'RE INTERESTED IN DISCOVERING STRING VARIABLES, SUBROUTINES, FUNCTIONS, ARRAYS, NESTED LOOPS, HOW TO DEAL WITH DISKS AND AVOID BUGS, ETC ETC ETC, THEN **GO TO** YOUR LOCAL LIBRARY OR BOOKSTORE AND GET STARTED !!

AND OTHERWISE?

I'M SORRY... THAT POSSIBILITY IS AGAINST UNIVERSITY POLICY...

SOFTWARE SURVEY



HERE'S A LOOK AT SEVERAL IMPORTANT AREAS OF SOFTWARE WHICH HAVE EMERGED IN THE YEARS SINCE ENIAC...

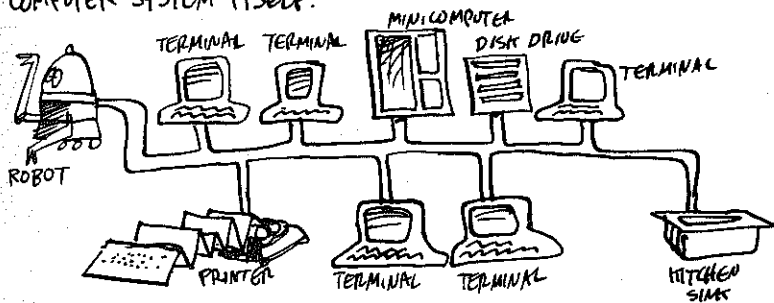
SYSTEMS SOFTWARE

PROGRAMS ARE COMMONLY DIVIDED INTO SYSTEMS SOFTWARE AND APPLICATIONS SOFTWARE.

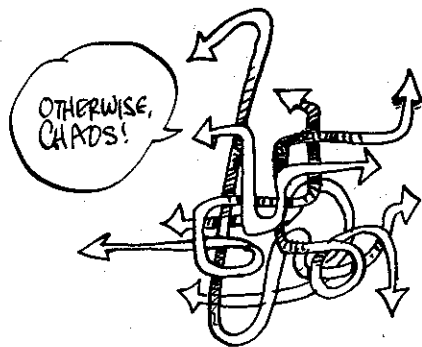


WITH SOME GRAY AREA OF OVERLAP!

APPLICATIONS SOFTWARE DOES "REAL WORLD" JOBS, WHILE SYSTEMS SOFTWARE EXISTS PURELY TO REGULATE THE COMPUTER SYSTEM ITSELF.



A SYSTEM TYPICALLY CONSISTS OF ONE OR MORE INPUT/OUTPUT DEVICES (TERMINALS, PRINTERS, CARD READERS, COMMUNICATIONS PORTS), PROCESSORS, MEMORY UNITS (MAIN AND MASS), AND WHO KNOWS WHAT ELSE. SOMETHING HAS TO COORDINATE IT ALL!



THE PROGRAM THAT DOES IT IS CALLED THE

OPERATING SYSTEM.

IF YOU THINK OF THE COMPUTER'S CORE AS A GIANT ELECTRONIC FILING CABINET (WITH A CALCULATOR ATTACHED), THEN THE OPERATING SYSTEM

★ CREATES THE STRUCTURE OF THE FILES

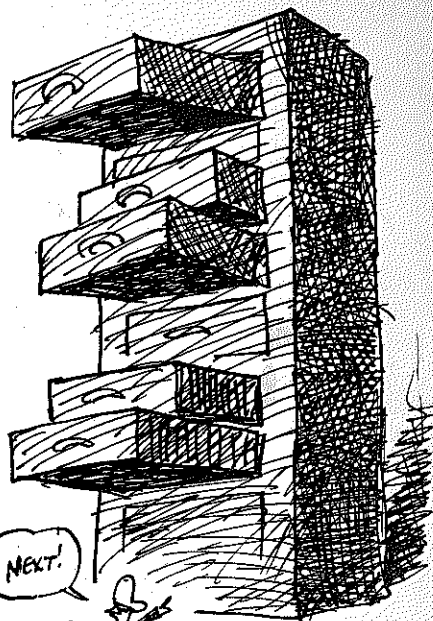
★ MANAGES MEMORY SO THAT DIFFERENT FILES DON'T BUMP INTO EACH OTHER

★ REGULATES ACCESS TO THE FILES AND THE MOVEMENT OF INFORMATION TO AND FROM OTHER PARTS OF THE SYSTEM...

ETC!

BESIDES THE OPERATING SYSTEM, SYSTEM SOFTWARE INCLUDES OTHER PROGRAMS "IN THE SYSTEM," SUCH AS LOADERS (WHICH LOAD PROGRAMS INTO MEMORY) AND COMPILERS (WHICH TRANSLATE HIGHER-LEVEL LANGUAGE INTO MACHINE CODE).

ALL INVISIBLE TO THE USER!



DATA BASE MANAGEMENT



THIS IS A BIGGIE!

A **DATA BASE** IS JUST A BIG PILE OF INFORMATION: A LIBRARY'S CARD CATALOG, A BANK'S TRANSACTION RECORDS AND ACCOUNT BALANCES, AN AIRLINE'S FLIGHT SCHEDULES AND RESERVATIONS, POLICE FILES, STOCK EXCHANGE DATA — ALL ARE DATA BASES.

➔ A DATA BASE MANAGEMENT PROGRAM ORGANIZES, UPDATES, AND PROVIDES ACCESS TO THE DATA BASE.

IN THE CASE OF AN AIRLINE, FOR EXAMPLE, THE COMPUTER HAS TO BOOK RESERVATIONS, ASSIGN SEATS, ERASE RESERVATIONS WHEN THE CUSTOMER CANCELS, MAKE REASSIGNMENTS IF A FLIGHT IS CANCELED, PRINT THE TICKETS, AND PROVIDE ALL THE FLIGHT INFORMATION TO TRAVEL AGENTS — WORLDWIDE!!



WORD PROCESSING

A "PERSONAL" USE FOR COMPUTERS...

WORD PROCESSING SOFTWARE ALLOWS YOU TO WRITE, EDIT, AND FORMAT TEXT — ALL FROM THE SAME KEYBOARD. YOU CAN GO FROM FIRST TO FINAL DRAFT ELECTRONICALLY, BEFORE EVER PRINTING A WORD.

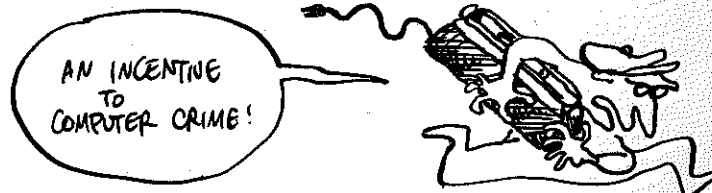


THERE ARE ALSO PROGRAMS TO CORRECT SPELLING — AND EVEN TO FIX SYNTAX AND GRAMMAR. SOON ILLITERATES WILL BE CREATING MASTERPIECES!

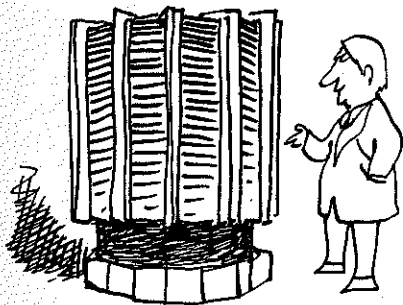
"HAMLET" BY J. FRED SHAKESPEARE...



A SMALL COMPUTER WITH WORD PROCESSING CAN BE QUITE INEXPENSIVE... THE CATCH IS THAT A "LETTER QUALITY" PRINTER CAN COST TEN TIMES THE PRICE OF A TYPEWRITER!



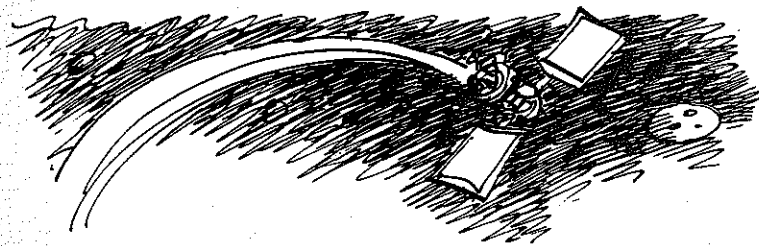
SCIENCE



CRAY-1 COMPUTER, CAPABLE OF 100 MILLION OPERATIONS PER SECOND!!

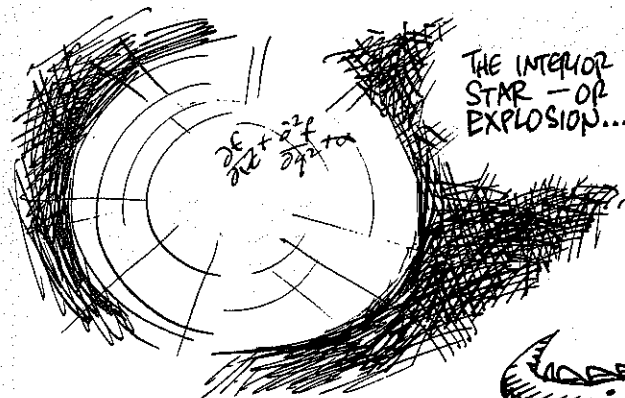
SCIENCE DEPENDS ON MATHEMATICS AND COMPUTERS ARE SUPER MATHA MACHINES. THE FASTEST, MOST POWERFUL COMPUTERS ARE MAINLY APPLIED TO SCIENTIFIC PROBLEMS.

THESE "SUPERCOMPUTERS" EXCEL AT SIMULATION. THE IDEA BEHIND SIMULATION IS TO FEED THE COMPUTER THE EQUATIONS GOVERNING A PHYSICAL SYSTEM AND THEN MATHEMATICALLY "MOVE" THE SYSTEM ACCORDING TO THOSE LAWS.



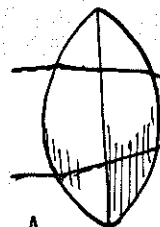
TIME SPACE TRAVEL: A COMPUTER CAN GUIDE A CRAFT TO THE MOON, BECAUSE IT CAN INTERNALLY SIMULATE THE ENTIRE FLIGHT!!

COMPUTERS CAN SIMULATE:



THE INTERIOR OF A STAR — OR A NUCLEAR EXPLOSION...

THE EVOLUTION OF AN ECOSYSTEM...



A LENS

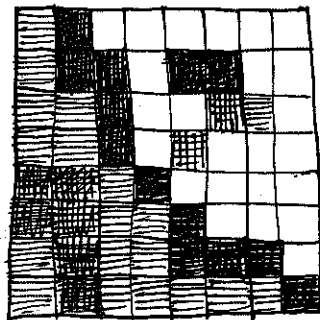


CLIMATE (ALTHOUGH EVEN THE FASTEST COMPUTER ISN'T FAST ENOUGH TO PREDICT THE WEATHER).

I'LL TELL YOU TOMORROW'S WEATHER NEXT WEEK!

GRAPHICS

FROM THE SIMPLEST "PONG" SCREEN TO THE MOST SOPHISTICATED FLIGHT SIMULATOR, THE IDEA IS THE SAME:

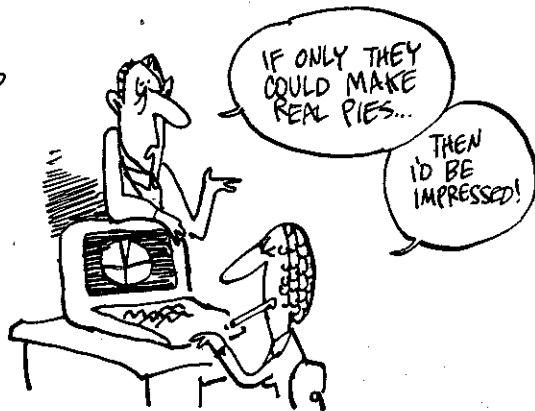


DIVIDE THE SCREEN AREA INTO A LARGE NUMBER OF TINY RECTANGLES ("PIXELS") AND ASSIGN EACH ONE A COLOR AND BRIGHTNESS.

THAT'S WHY COMPUTER PICTURES HAVE CORNERS!

BUT THERE ARE ALSO ALGORITHMS FOR SMOOTHING CORNERS!

UNFORTUNATELY, IT TAKES A LOT OF COMPUTER POWER TO DO FANCY GRAPHICS. SMALL COMPUTERS MOSTLY DO THINGS LIKE MAKE PIE CHARTS...



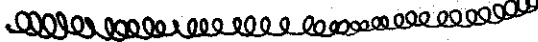
COMMUNICATION

THE BIGGEST COMPUTER SYSTEM OUTSIDE GOVERNMENT BELONGS TO THE TELEPHONE COMPANY.

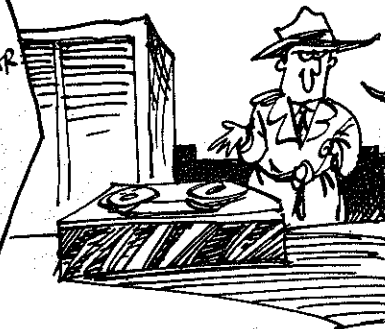
A VOICE (OR ANY OTHER SIGNAL) CAN BE DIGITALLY ENCODED, TRANSMITTED, AND DECODED.



COMPUTERS ALSO CONTROL THE ROUTING AND SWITCHING OF CALLS THROUGH THE NETWORK — AND KEEP TRACK OF EVERYONE'S BILL!



COMPUTERS CAN BE PROGRAMMED TO RECOGNIZE PARTICULAR WORDS OR GROUPS OF WORDS — A CAPABILITY NOT LOST ON THE INTELLIGENCE COMMUNITY..



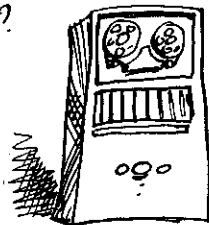
WE CAN AUTOMATICALLY RECORD ANY CONVERSATION CONTAINING WORDS I CAN'T SAY BECAUSE I DON'T WANT TO BE RECORDED...

ARTIFICIAL INTELLIGENCE

DESPITE THEIR INCREDIBLE SPEED AND ACCURACY, COMPUTERS ARE LOUSY AT PATTERN RECOGNITION, ANALYSIS, HUNCH-PLAYING, AND UNDERSTANDING HUMAN LANGUAGE!

CAN A MACHINE BE PROGRAMMED TO THINK?

ER... WELL...UM... AH-- LET ME SEE...



ACTUALLY, WE KNOW VERY LITTLE ABOUT HOW THINKING WORKS...

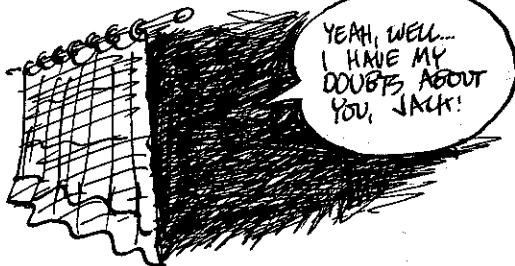
SO A BETTER QUESTION IS: HOW CAN YOU TELL IF A MACHINE IS THINKING?

ALAN TURING SUGGESTED THIS TEST: SUPPOSE YOU COULD COMMUNICATE WITH SOMETHING, OR SOMEONE, CONCEALED FROM VIEW. IF, ON THE BASIS OF THE CONVERSATION, YOU COULDN'T SAY WHETHER IT WAS MACHINE OR HUMAN, YOU WOULD HAVE TO SAY IT WAS THINKING!

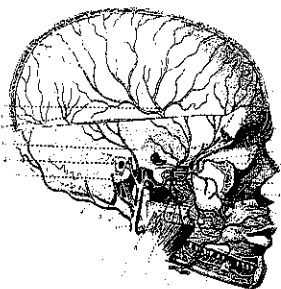
IT'S A MACHINE!



YEAH, WELL... I HAVE MY DOUBTS ABOUT YOU, JACK!



I PERSONALLY DISLIKE THIS CRITERION, ON THE GROUNDS THAT A SIMULATION ISN'T THE REAL THING...

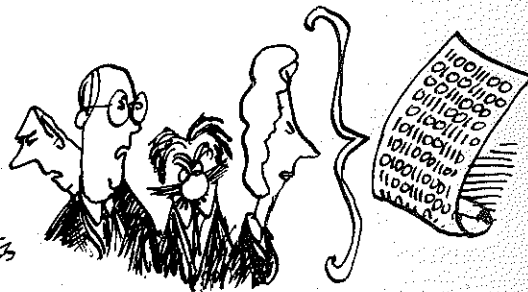


ONWARD!

THIS PHILOSOPHICAL MUDDLE HASN'T STOPPED PEOPLE FROM TRYING TO MAKE MACHINES THINK. THEY'VE HAD SOME SUCCESS WITH SO CALLED **EXPERT SYSTEMS**, WHICH MIMIC HUMAN EXPERTS IN VARIOUS FIELDS.

HOW DO YOU CREATE AN EXPERT SYSTEM?

FIRST, INTERVIEW A BUNCH OF EXPERTS — GEOLOGISTS, FOR EXAMPLE — AND FORCE THEM TO SPELL OUT THE ALGORITHMS BEHIND THEIR SKILLS, HUNCHES AND BRAINSTORMS.



THEN LOAD THE COMPUTER'S MEMORY WITH THE HUMANS' KNOWLEDGE BASE... AND THE RESULT IS (SOMETIMES) A PROGRAM WHICH CAN OUTPERFORM ANY HUMAN!!

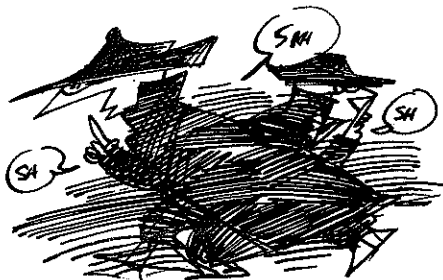
IF WE'RE SO SMART, WHY DID WE LET THIS HAPPEN?



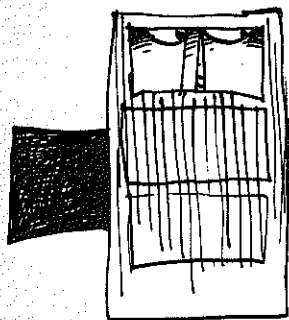
CRYPTOGRAPHY

SHHH!

THERE ARE STANDARD CODES LIKE ASCII (P. 128) FOR CONVERTING WRITTEN TEXT INTO BINARY... BUT WHAT ABOUT USING COMPUTERS FOR SECRET CODES??



SECRET CODES USED TO BE STRICTLY MILITARY AND SPY STUFF, BUT NOW MORE AND MORE SENSITIVE INFORMATION IS STORED IN COMPUTER SYSTEMS:

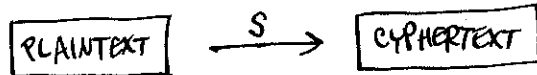


MEDICAL RECORDS,
BANK RECORDS,
CENSUS DATA
INCOME TAX
RECORDS,
GRADE TRANSCRIPTS,
CORPORATE MEMOS,
ETC. ETC.



SCRAMBLING DATA HAS BECOME AN IMPORTANT WAY OF PROTECTING PRIVACY !!

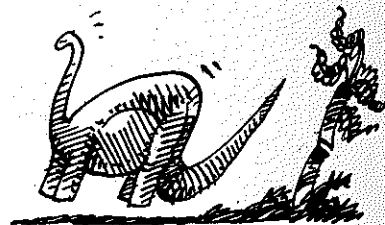
ORDINARILY, INFORMATION IS STORED AS A BINARY STRING ANY COMPUTER CAN READ: THE **PLAINTEXT**, IN CRYPTOGRAPHIC JARGON. TO ENCRYPT IT YOU APPLY SOME ALGORITHM **S**, WHICH CONVERTS IT TO A SCRAMBLED MESSAGE CALLED THE **CYPHERTEXT**.



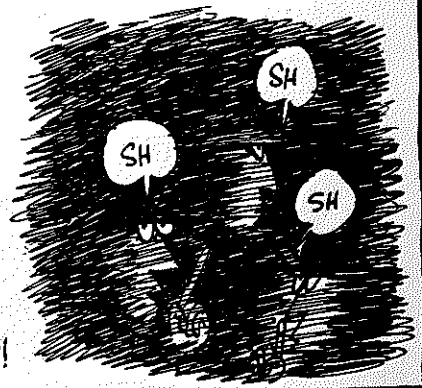
THEORETICALLY, IT'S IMPOSSIBLE TO RECONSTRUCT THE PLAINTEXT FROM THE CYPHERTEXT WITHOUT KNOWING SOMETHING ABOUT **S** ... HOWEVER, A POTENTIAL CODE-BREAKER COULD PUT A COMPUTER TO WORK SEARCHING FOR **S**.



TO BE SECURE, **S** HAS TO BE SO COMPLICATED THAT EVEN THE FASTEST COMPUTER WOULD TAKE, SAY, A FEW MILLION YEARS TO FIGURE IT OUT!



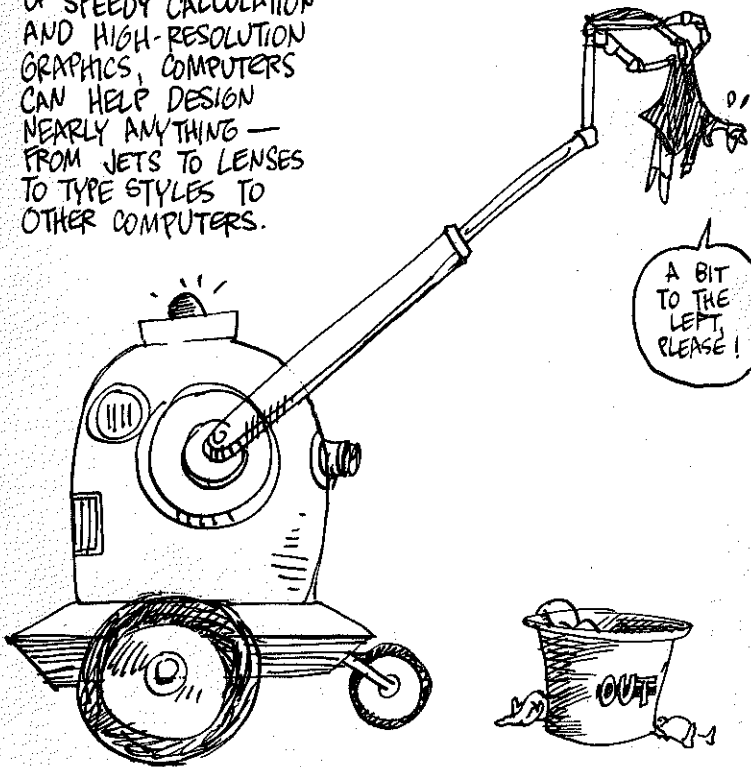
RECENTLY, THE NATIONAL BUREAU OF STANDARDS APPROVED A FAMILY OF ALGORITHMS AS A DATA ENCRYPTION STANDARD FOR THE NATION. SEVERAL SCIENTISTS SUSPECT THAT THIS STANDARD IS JUST COMPLEX ENOUGH TO STYMIE ORDINARY COMPUTERS, BUT NOT TOO TOUGH FOR THE NINE ACRES OF COMPUTERS OF THE NATIONAL SECURITY AGENCY!



CAD/CAM

○ COMPUTER-AIDED DESIGN /
○ COMPUTER-AIDED MANUFACTURE

USING A COMBINATION OF SPEEDY CALCULATION AND HIGH-RESOLUTION GRAPHICS, COMPUTERS CAN HELP DESIGN NEARLY ANYTHING — FROM JETS TO LENSES TO TYPE STYLES TO OTHER COMPUTERS.



THEN THEY CAN GO ON TO CONTROL AUTOMATIC MANUFACTURING PROCESSES AS WELL. YES, **ROBOTS** ARE ALREADY HERE!!

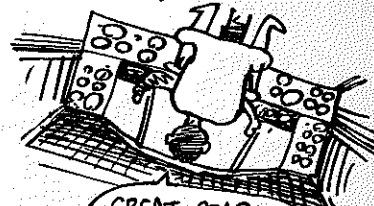
WAR

THE MILITARY CAN USE JUST ABOUT EVERY TYPE OF SOFTWARE WE'VE MENTIONED — AND THEN SOME!

ENIAC WAS BUILT FOR CALCULATING BALLISTICS... NOW WE HAVE BALLISTIC MISSILES!



FLIGHT SIMULATORS CAN TRAIN PILOTS RIGHT ON THE GROUND...



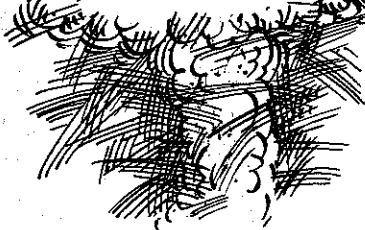
GREAT GRAPHICS ON THESE!

THEN THERE ARE THE FAMOUS "SMART" MISSILES, WHICH CAN FOLLOW A MOVING TARGET...



I'M ABOUT TO BLOW MYSELF UP... HOW SMART IS THAT?

SUPERCOMPUTERS HELP DESIGN NUKES...



...NOT TO MENTION DATA PROCESSING AND CRYPTOGRAPHY... SO GREAT IS THE DEFENSE DEPARTMENT'S SOFTWARE NEED THAT THEY HAVE THEIR OWN PROGRAMMING LANGUAGE: **ADA**, NAMED AFTER THE UNFORTUNATE LADY LOVELACE.



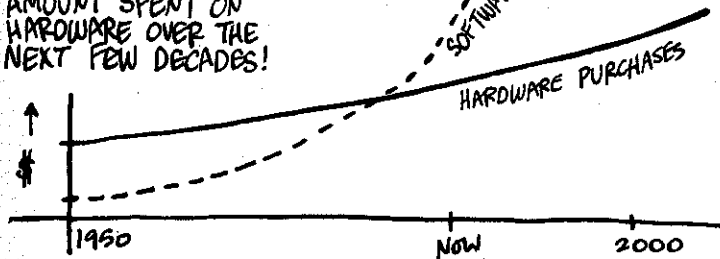
MERCY!

THIS LITTLE SURVEY ONLY BEGINS TO SUGGEST THE RANGE OF SOFTWARE CURRENTLY AVAILABLE. EVERY DAY THERE'S MORE... SOME PROGRAMS MOVE INTO NEW AREAS, WHILE OTHERS INTEGRATE EXISTING ROUTINES INTO NEW, MORE POWERFUL PACKAGES.



THIS BABY DOES WORD PROCESSING, MANAGES A GERBIL RANCH, AND DESIGNS H-BOMBS! EVERY GERBIL GETS ITS OWN DETERRENT!

IF YOU'RE LOOKING FOR OPPORTUNITY IN THE COMPUTER BUSINESS, CONSIDER THIS: THE TOTAL CONSUMPTION OF SOFTWARE, WHICH BEGAN AS A SMALL FRACTION OF COMPUTING COSTS, IS EXPECTED TO REACH MANY TIMES THE AMOUNT SPENT ON HARDWARE OVER THE NEXT FEW DECADES!

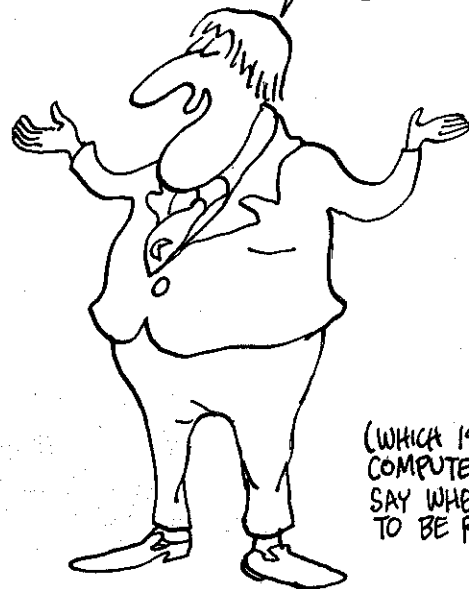


WRITE SOFTWARE!

IN CONCLUSION,

A FEW WORDS ABOUT THIS FAMILIAR SENTENCE:

COMPUTERS ONLY DO WHAT PEOPLE TELL THEM TO DO!



(WHICH IS WHAT COMPUTER SCIENTISTS SAY WHEN THEY WANT TO BE REASSURING...)

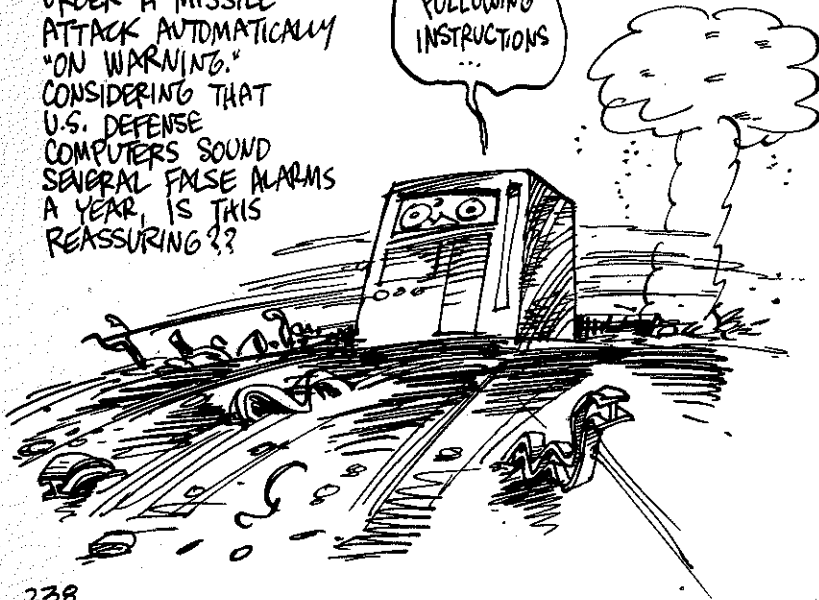
TECHNICALLY, IT'S TRUE, IN THE SENSE THAT SOFTWARE CONTROLS COMPUTERS, AND PEOPLE WRITE SOFTWARE...

BUT WHO CONTROLS PEOPLE?!!



FOR EXAMPLE, SUPPOSE A NATION'S STRATEGIC PLANNERS DECIDED TO PROGRAM THEIR COMPUTERS TO ORDER A MISSILE ATTACK AUTOMATICALLY "ON WARNING." CONSIDERING THAT U.S. DEFENSE COMPUTERS SOUND SEVERAL FALSE ALARMS A YEAR, IS THIS REASSURING??

I WAS ONLY FOLLOWING INSTRUCTIONS ...



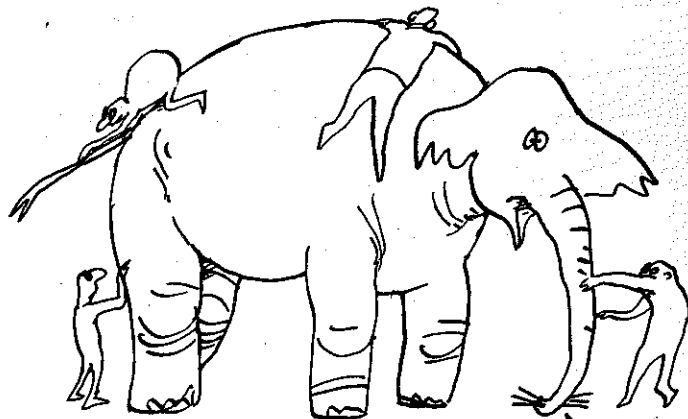
MY CALCULATOR SAYS

$2^{16} = 65,536.001$
(REALLY!)



ANOTHER PROBLEM IS THAT ALGORITHMS DON'T ALWAYS DO EXACTLY WHAT THEY ARE SUPPOSED TO.

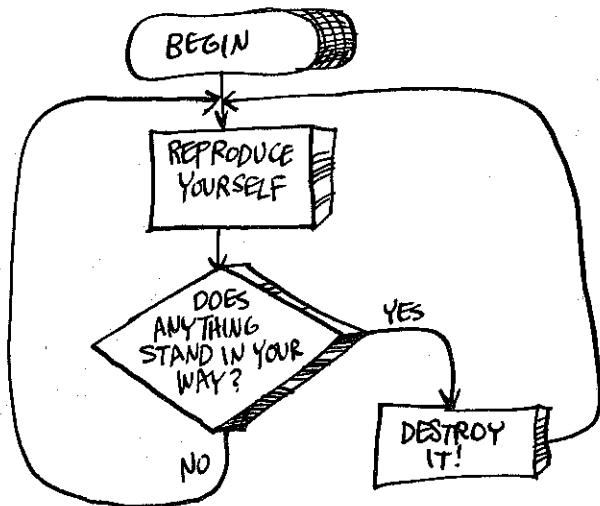
LARGE SOFTWARE SYSTEMS ARE WRITTEN BY TEAMS OF PROGRAMMERS. LIKE THE ELEPHANT, NO ONE UNDERSTANDS THE WHOLE THING!



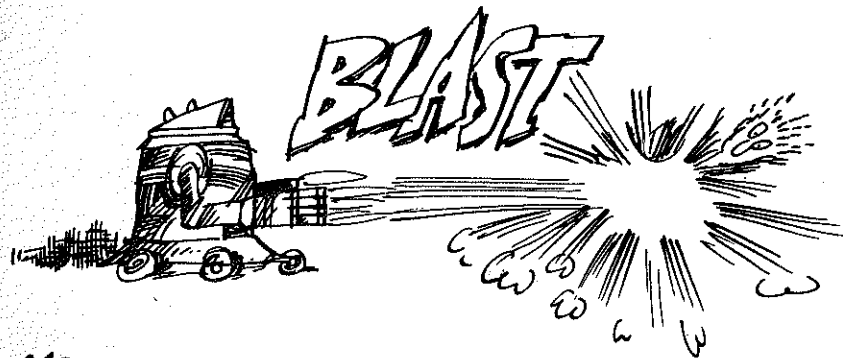
COMPUTERS ROUTINELY DO BIZARRE AND UNEXPECTED THINGS, ESPECIALLY WHEN RUNNING NEW, UNTESTED SOFTWARE!

I'M GETTING TO BE A TIRED METAPHOR!

FINALLY, CONSIDER THIS OMINOUS ALGORITHM:



WHILE NO COMPUTER IS INTELLIGENT, MOBILE, OR WELL EQUIPPED ENOUGH — YET — TO EXECUTE THESE INSTRUCTIONS, SUCH A MACHINE REMAINS A THEORETICAL POSSIBILITY. THIS PROGRAM WOULD MAKE IT SOMETHING VERY MUCH LIKE A COMPETING LIFE FORM!!!



AND IF YOU THINK THAT BECAUSE "IT'S ONLY A MACHINE," YOU CAN ALWAYS TURN IT OFF, PONDER THE WORDS OF NORBERT WIENER, A SCIENTIST WHO THOUGHT DEEPLY ABOUT THESE THINGS:



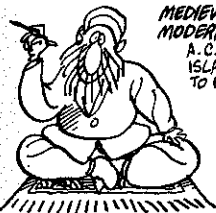
"TO TURN A MACHINE OFF EFFECTIVELY, WE MUST BE IN POSSESSION OF INFORMATION AS TO WHETHER THE DANGER POINT HAS COME. THE MERE FACT THAT WE HAVE MADE THE MACHINE DOES NOT GUARANTEE THAT WE SHALL HAVE THE PROPER INFORMATION TO DO THIS.... THE VERY SPEED OF... MODERN DIGITAL MACHINES STANDS IN THE WAY OF OUR ABILITY TO PERCEIVE AND THINK THROUGH THE INDICATIONS OF DANGER." *

* CYBERNETICS, SECOND EDITION, P. 176

SO WELCOME TO THE INFORMATION AGE, AND HAPPY COMPUTING !!



SOME FURTHER READING:

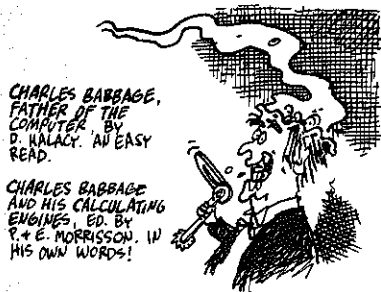


MEDIEVAL AND EARLY MODERN SCIENCE BY A.C. CROMBIE. TELLS HOW ISLAMIC SCIENCE CAME TO EUROPE.

THE MAKING OF THE MICRO BY C. EVAUS. NICE DIAGRAMS OF OLD ADDING MACHINES

HISTORY OF MATHEMATICS BY A. GITTLEMAN. DON'T MISS THE STORY OF NAPIER'S "PSYCHIC" CHICKEN!

THE COMPUTER FROM PASCAL TO VON NEUMANN BY H. GOLDSTINE. THE DEFINITIVE ACCOUNT OF ENIAC.



CHARLES BABBAGE, FATHER OF THE COMPUTER BY D. WALACE. AN EASY READ.

CHARLES BABBAGE AND HIS CALCULATING ENGINES, ED. BY P. + E. MORRISON. IN HIS OWN WORDS!

SYMBOLIC LOGIC AND THE GAME OF LOGIC BY LEWIS CARROLL. MILLIONS OF SILLY SYLLOGISMS

THE MATHEMATICAL THEORY OF COMMUNICATION BY C. SHANNON. CONTAINS TWO PARTS, ONE WITH AND ONE WITHOUT MATH

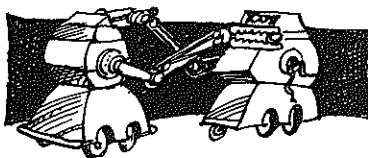
CYBERNETICS, 2ND EDITION, BY N. WEINER. THE THEORY OF AUTOMATIC CONTROL

UNDERSTANDING DIGITAL ELECTRONICS, BY D. McWHORTER. BOOLEAN CIRCUITS

UNDERSTANDING DIGITAL COMPUTERS, BY F. MING. A PERSONAL FAVORITE, BUT LOOK OUT FOR MISPRINTS!

INTRODUCTION TO MICROCOMPUTERS, BY A. OSBORNE (4 VOLUMES). VERY DETAILED

UNDERSTANDING COMPUTER SCIENCE BY R.S. WALKER. MORE ADVANCED TOPICS



ILLUSTRATING BASIC BY D. ALCOCK. A QUICK COURSE, USING QUASI-CARTOONS

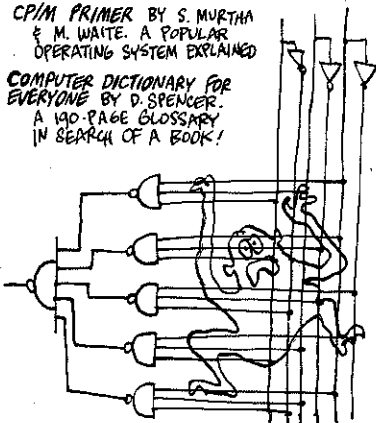
USING BASIC, BY R. DIDDAY & R. PAGE. A GENTLER, BUT WORDIER, APPROACH

PASCAL PRIMER BY D. FOX & M. WAITE. IT HELPS TO KNOW BASIC BEFORE READING THIS

FORTRAN COLORING BOOK BY R. KAUFMAN. WITTY, BORDERING ON CORNY

CP/M PRIMER BY S. MURTHA & M. WAITE. A POPULAR OPERATING SYSTEM EXPLAINED

COMPUTER DICTIONARY FOR EVERYONE BY D. SPENCER. A 140-PAGE GLOSSARY IN SEARCH OF A BOOK!



INDEX

Abacus, 32-34, 43
 Abbreviations, mnemonic, 175
 Abstract symbol-manipulation, 42
 Accumulator, 173
 Ada programming language, 235
 Adder, 123
 1-bit, 125-126
 Adding machines, 59
 Address register, 173
 Addresses, 155
 possible, 156
 Aiken, Howard, 72
 Algebra, 40
 Boolean, 101-105
 Algorithm, 41-42
 Algorithms, 41, 195
 examples of, 196
 examples of non-algorithms, 197
 flow of, 198
 Al-Khwarismi, 40
 Alphabet, 30-31
 Alphabetical order, 31
 Alphanumeric information, 130
 ALU (arithmetic logic unit), 130-132
 American Standard Code for Information Interchange (ASCII), 128
 Analysis, 200
 Analytical Engine, 53-55
 AND-gate, 107
 multiple-input, III
 seatbelt buzzer in, 109
 AND logical operator, 103
 Applications software, 222
 Arabs, 40-43
 Arithmetic logic unit (ALU), 130-132
 Arithmetic on paper, 39
 Arithmetic table, Chinese, 29
 Artificial intelligence, 230-231
 ASCII (American Standard Code for Information Interchange), 128

Assemblers, 205
 Assembly language, 174-176
 Assembly language statements, 175
 Asynchronous ripple counter, 148
 Automated type-setter, 54
 Automatic switches, 106-109

 B register, 173
 Babbage, Charles, 51-56, 58
 Babbage's Law, 58
 Ballistic tables, 74
 Ballistics, 73
 BASIC language, 162, 206
 basic, 207-209
 Begin box, 198
 Binary calculation, 121
 Binary code, 205
 Binary coded decimal, 127
 Binary numbers, 115
 adding two, 121
 counting in, 120
 multiplying, 122
 subtracting, 122
 translated into decimal numbers, 119
 Binary strings, 130
 Bits, 123
 carry, 125
 Bone, tally, 20, 23
 Boole, George, 101
 Boolean algebra, 101-105
 Boxes, specially shaped, 198
 Branch instructions, 183
 Bubble memories, 166
 Bugs, 210, 211
 Bus architecture, 170
 Bytes, 123

 C register, 173
 CAD/CAM, 234

Calculation, 34
 binary, 121
Calculators, 36
 mechanical, 59
Calculus, 34
Cards, punched, *see* Punched cards
Carroll, Lewis, 100
Carry bit, 125
Cash registers, 59
Census tabulators, 60
Charge-coupled devices, 166
Chinese arithmetic table, 29
Chinese number system, 27-29
Chinese writing, 22
Circuits
 electrical, completing, 62
 integrated, 84
Clocks, 142-143
Codes, secret, 232-233
Combinational logic, 142
Communication, 229
Compilers, 205, 223
Computer-aided design/computer-aided
 manufacture, 234
Computers, 5-6
 clocks and, 142-143
 control of, 237-241
 Cray-1, 226
 describing, 89
 evolution of, 14
 IBM Personal, 96
 information and, 6
 problems with, 237-241
 research in, 153
 size of, 152
Conditional branch box, 198
Conditional jumps, 57, 183
 "smart," 215
Control, 170
 transfer of, 183
Control bus, 170
Control flow, 93-94
Control unit, 92
Core memories, 158
Counters, 146-148
 asynchronous ripple, 148
 synchronous, 148
Counting
 binary and decimal, 120
 digital, 19
Cray-1 computer, 226
Crusaders, 44
Cryptography, 232-233
Cycles, 142
Cyphertext, 233
Data base management, 224
Data bases, 224
Data processing, 63
Decimal system, 115
 binary numbers translated into, 119
Decoders, 129
Deductive logic, 99
"Difference Engine, The," 51-52
Digital counting, 19
Disallowed input, 137
Disks, magnetic and floppy, 166-167
DNA, 12
DNA-protein system, 13
DNA technology, recombinant, 86
Duns Scotus, 99
EBCDIC, 128
Eckert, J. Presper, 74
Egyptian number system, 26
Egyptians, 22
Electric "mouse," 7
Electrical circuits, completing, 62
Electricity, 65
Electromechanical memories, 154
Electromechanical switches, 71
Electronic memories, 154
Electronic Numerical Integrator and
 Calculator (ENIAC), 75-76
Encoders, 129
Encoding instructions, 79
End box, 198
ENIAC (Electronic Numerical Integrator
 and Calculator), 75-76
EPROM, 161
Expert systems, 231
External storage of information, 20
Facts, 7
Fetching instructions, 178
Fibonacci sequence, 219
Fields, 176
Fives, counting by, 23
Flight simulators, 235
Flip-flop inputs, 137

Flip-flops, 133-137
 master-slave, 144
Floating point representation, 127
Floppy disks, 166-167
Flow charts, 198-200
 examples of, 199
 for multiple plug-ins, 203
 for roommate receipts, 202
For-next commands, 218
FORTRAN, 206
Gating network, 140
Gene, 12
Glitches, 143
Go-to statement, 215
Grammar, laws of, 18
Graphics, 228
Gravitation, theory of, 45-46
Greek mathematicians, 33
Handfuls, counting by, 24-25
Hardware, 187
Hertz (one cycle per second), 142
Hexadecimal numerals, 157
Higher-level programming languages,
 205
Hindus, 37-38
Hollerith, Herman, 60, 64
IBM, 64
IBM Personal Computer, 96
If-then statement, 215
Incas, 22
Increments, 146
Inductive logic, 99
Industrial Revolution, 49
Information
 ages of, 1-86
 alphanumeric, 130
 computers and, 6
 defined, 7-8
 excess, 3-5, 86
 external storage of, 20
 forms of, 8-9
 power of, 12
 stored, 10
Information flow, 93-94
Information processing, 11
 understanding, 90
Information theory, 7

Input, 48, 92, 95
 card-reading device, 54
 disallowed, 137
 flip-flop, 137
Input box, 198
Input-output (I/O) tables, 110, 112-113
Input statement, 214
Input wire, 106
Instruction register, 173
Instruction set, Motorola 6800, 182
Instructions, 48
 branch or jump, 183
 8-bit, 156
 encoding, 79
 fetching, 178
 machine, 176
 microinstructions, 178
 to mill, 53
Integers, 127
Integrated circuits, 84
Integration, large-scale and very
 large-scale, 84
Intelligence, artificial, 230-231
Internal memory, 155
Interpreters, 205
Inverters, 108
I/O (input-output) tables, 110, 112-113
Jacquard, Joseph Marie, 50
Japanese calculation of π , 29
Jump, conditional, *see* Conditional jumps
Jump instructions, 183
K (kilo), 163
Language
 assembly, *see* Assembly language
 entries
 BASIC, *see* BASIC language
 expressive, 17
 higher-level programming, 205
 machine, 177
Large-scale integration (LSI), 84
Latches, 138
 gated, 140
Leibniz, Gottfried Wilhelm, 47
Let statement, 211
Life form, 13
 competing, 240
Line numbers, 209

Loaders, 223
Logic, 99
 combinational, 142
 laws of, 18
 sequential, 142
 simple, 150
 symbolic, 101
Logic gates, multiple-input, 111
Logic unit, arithmetic (ALU), 130-132
Logical operations, 98
Logical operators, 103-104
Logical spaghetti, 87-184
Loom, Jacquard, 50
Looping, value of, 57
Lovell, Ada, 56-58

Machine instruction, 176
Machine language, 177
Magnetic disks, 166
Magnetic tape, 165
Mainframes, 85
Mark I, 72
Mass storage, 165
 uses of, 168
Master-slave flip-flop, 144
Mauchly, John, 74
Mechanical calculators, 59
Megaflops, 85
Memory, 94, 95
 bubble, 166
 core, 158
 electromechanical, 154
 electronic, 154
 internal, 155
 random access (RAM), 159
 read-only, *see* Read-only memory
Memory unit, 54
Merge program, 81
Messages, form of, 16
Messenger RNA, 12
"Method of the Celestial Element, The,"
 29
Microcomputer, 85
Microinstructions, 178
Microprogram, 181
Military software, 235
Mill of the Analytical Engine, 53-55
Minicomputer, 85
Mnemonic abbreviations, 175

Modem, 96
Motorola 6800 instruction set, 182
"Mouse," electric, 7
Multiple-input logic gates, 111
Multiplication, binary, 122
Music, 9

NAND-gate, 134
Napier, John, 47
"Napier's bones," 47
Newton, Isaac, 45
Nibbles, 124
NOR-gate, 138
NOT logical operator, 104
Nucleotide pairs, 12
Number system
 Chinese, 27-29
 Egyptian, 26
Numbers, 18
 binary, *see* Binary numbers
 hexadecimal, 157
Numerical variables, 210-211

Object code, 205
Op-code, 176
Operand, 175
Operating system, 223
Operations, logical, 98
Operator, 175
 logical, 103-104
Optic nerve, 8
Optical disks, 166
OR-gate, 108
 multiple-input, 111
OR logical operator, 103
Order, alphabetical, 31
Output, 54, 92, 95
Output box, 198
Output wire, 106

Paper, 37
 arithmetic on, 39
Paper tape, 165
Papermaking, 43
Parallel registers, 141
Pascal, 206
Pascal, Blaise, 47
Personal Computer, IBM, 96
Pi, Japanese calculation of, 29

Pictograms, 30-31
Pictures, 8
Pixels, 228
Plaintext, 233
Powers of two, 118
Print statement, 212
Printer, 96
Procedure box, 198
Processing unit, 92, 95, 97
Program counter, 173
Programmable ROMs (PROMS), 161
Programmer, first, 57
Programming languages, higher-level,
 205
Programs, *see also* Software
 microprogram, 181
 for multiple plug-ins, 217
 for roommate receipts, 216
 self-modification by, 80
 sort and merge, 81
 stored, 78-80, 82
Prompts, 208
PROMS (programmable ROMs), 161
Punched cards, 49-50, 165
 functions of, 55
 input device for, 54
 responses on, 61
Punctuation, 209
Pushbutton switches, 67

Random access memory (RAM), 158-160
Read-data statement, 210
Read-only memory (ROM), 158-159, 161
 programmable (PROMS), 161
 uses of, 162
Recombinant DNA technology, 86
Recorded signals, 10
Registers, 139
 parallel, 141
 shift, 144
Relay
 telephone, 69
Relay, automatic, 68
Remarks, 209
Renaissance, 44
Return key, 208
Rings on floppy disks, 167
Ripple counter, asynchronous, 148
RNA, messenger, 12

Robots, 234
ROM, *see* Read-only memory
Romans, 34-35
Rotary switches, 67
Run statement, 208

Schickard, Wilhelm, 47
Scientific problems, 226-227
Secret codes, 232-233
Sectors on floppy disks, 167
Self-modification, program, 80
Self-reproducing machines, 193
Semicolons, 213
Semiconductors, 83
Senses, 15
Sensory impressions, 15
Sequential logic, 142
Shannon, Claude, 7
Shift register, 145
Signals, 8-9
 recorded, 10
Simulations, 226-227
Software, 185-236
 applications, 222
 defined, 187
 growth of, 236
 military, 235
 range of, 236
 survey of, 221-236
 systems, 222
Sort and merge program, 81
Source code, 205
Spaghetti, logical, 87-184
Statements, 209
Storage
 external, of information, 20
 mass, *see* Mass storage
Store, memory, 54
Stored information, 10
Stored programs, 78-80, 82
Subroutines, 57
Subtraction, binary, 122
Sumerians, 21-22
Supercomputers, 85
Superminicomputers, 86
Swan-pan, Chinese arithmetic table, 29
Switchboard, 68
Switches, 66-68
 automatic, 106-109

Switches (*cont'd*)
 electromechanical, 71
 patterns of, 70
Symbol-manipulation, abstract, 42
Symbolic logic, 100
Synchronous counters, 148
Systems, 222
 expert, 231
 operating, 223
Systems software, 222

Tabulators, census, 60
Tally bone, 20, 23
Tape, paper and magnetic, 165
Tartaglia, Niccolo, 45
Telephone company, 229
Telephone relay, 69
Telephones, 68
Ten, 116-117
Tens, counting by, 23
Three-body problem, 46
Timing, 142
Toggle switches, 67
Toggling, 147
Transfer of control, 183
Transistors, 83-84
Transition rules, 191
Truth-values, 102

Tube, vacuum, 69
Turing, Alan, 190, 230
Turing machines, 191-192
Two, powers of, 118
"Two's complement" method, 122
Type-setter, automated, 54

Unconditional branching statement, 215
Universal Turing machine, 192

Vacuum tube, 69
Variables, 210
 numerical, 210-211
Very large-scale integration (VLSI), 84
von Neumann, John, 77, 193

Wiener, Norbert, 241
Word processing, 225
Words, 8
World War II, 72
Writing, 21-22
 Chinese, 22
Written zero, 37-38

Zero, 27-28
 written, 37-38
Zuse, Konrad, 71