TECHNICAL REPORT NO. 7

AN INTERACTIVE VISUAL COMPUTER SIMULATOR

STUART C. SHAPIRO

NOVEMBER 1, 1973

An Interactive Visual Computer Simulator

Stuart C. Shapiro
Computer Science Department
Indiana University

It is often felt desirable to include in an introductory computing course a small section of material on machine language, even when the course will mostly use a language such as BASIC, FORTRAN or ALGOL. The purpose of the section on machine language is to give the students an understanding of what a programmable, digital, sequential computer is, what its basic operations are, and how a program, made up of very basic operations can effect a fairly complicated calculation. To accomplish this purpose, it is not necessary to use a real computer as an example. Indeed, the complexities of a modern computer would obscure the points to be made. Therefore, various "hypothetical" computers have been invented that consist of a basic set of operations and registers. Students generally write programs for these hypothetical computers and find out if their programs are correct either from a human instructor or by executing them using a simulator in a batch environment. In either case, it is difficult for them to understand the sequential operation of their programs or the effect of any bugs that might have been in them.

In order to provide a better way for introductory computing students to gain this basic understanding, we have written an interactive simulator of one hypothetical computer. This

simulator displays all the storage and active registers of the hypothetical computer on the screen of a CRT terminal, allows the student to load and patch programs, load data, and execute the program. The simulator traces the program by changing the information on the screen at a slow enough rate that the student can watch how his program works. He can then make any modifications he feels are necessary and see the effect of those changes.

The simulator, HYCOMP1, was written in SNOBOL4[1] to simulate the HYCOMP computer of Terry Walker's introductory text.[2] It runs under the KRONOS Time Sharing System on a CDC 6600, using an Applied Digital Data Systems, Inc. ADDS Consul 880 terminal.

Walker's HYCOMP has 1000 words of memory, each holding a sign plus five decimal digits. Its instruction code ignores the sign, uses the two high order digits for an operation code and the remaining three digits for an address. In order to display all of memory on the CRT, we implemented only 100 words for HYCOMP1, and use the two low order digits for the address, ignoring the middle digit. HYCOMP and HYCOMP1 have the following active registers: a sign plus 5 digit arithmetic register (REG); a 5 digit control unit (CU); a program instruction counter (IC) which is three digits in HYCOMP and two digits on HYCOMP1. They

---

[1] Griswold, R.E., Poage, J.F., Polonsky, I.P., The SNOBOL4 Programming Language, second edition, Prentice-Hall, Englewood Cliffs, N.J., 1971.

[2] Walker, Terry M., Introduction to Computer Science: An Interdisciplinary Approach, Allyn and Bacon, Inc., Boston, 1972.

also have an overflow (OF) switch, underflow (UF) switch, and and end-of-file switch (EOF). Floating point numbers are stored using the two high order digits as a biased exponent. There are forty nine alphanumeric characters coded two digits per character and stored two characters per word in the low order four digits. The 24 instructions include numeric and alphanumeric I/O, integer and floating point arithmetic and test and branch instructions. HYCOMP does not have index registers, but in the future, they may be added to HYCOMP1 using the middle digit of instructions for the index register field.

When the student executes HYCOMP1, the screen is cleared and written on as shown in fig. 1. He may then load his program (only machine language is accepted by HYCOMP1) and his data and run the program. He may then patch or change his program and continue trying and changing it until he is satisfied or tired. The HYCOMP1 simulator gives the student the same options he would have if he were sitting at a HYCOMP1 console, but in addition, he can observe the execution of his program.

There are two commands with which the student can execute his HYCOMP1 program - RUN and CYCLE. With RUN, the student specifies the starting address of his program, and it is executed until it halts or produces an execution error. If the student uses CYCLE, one machine cycle is executed so that he can study the results of each instruction at his own speed.

```
00:         01:         02:         03:         04:         05:         06:
07:         08:         09:         10:         11:         12:         13:
14:         15:         16:         17:         18:         19:         20:
21:         22:         23:         24:         25:         26:         27:
28:         29:         30:         31:         32:         33:         34:
35:         36:         37:         38:         39:         40:         41:
42:         43:         44:         45:         46:         47:         48:
49:         50:         51:         52:         53:         54:         55:
56:         57:         58:         59:         60:         61:         62:
63:         64:         65:         66:         67:         68:         69:
70:         71:         72:         73:         74:         75:         76:
77:         78:         79:         80:         81:         82:         83:
84:         85:         86:         87:         88:         89:         90:
91:         92:         93:         94:         95:         96:         97:
98:         99:           OF=     UF=     IC=         CU=             REG=
                         EOF=
TO SEE A LIST OF AVAILABLE COMMANDS, TYPE HELP
TO PROCEED AFTER AN ERROR MESSAGE APPEARS, HIT  NEW LINE
TO LEAVE A MODE, JUST TYPE NEW COMMAND.  ? MEANS READY
INPUT APPEARS ON LINE 22, OUTPUT ON LINE 23
?_
```

Fig. 1. CRT screen after initialization

A machine cycle consists of the following steps:

1a. The CRT cursor underlines the contents of the word whose address is in the IC.

b. That instruction is copied into the CU and displayed there on the screen.

2. IC is incremented by 1, the new value being displayed properly on the screen.

3. The instruction in the CU is executed. This may cause various things to happen on the screen, for example:

a. A test instruction causes the displayed contents of REG to blink for a short time.

b. A branch instruction changes the IC.

c. Whenever data is read, either from REG, a memory word or on the Input line, the cursor underlines the data.

d. Data may be written into REG, a memory word or the Output line.

e. A floating point instruction may set up UF or OF.

f. An input instruction may set EOF.

If an execution error occurs, an appropriate message is written on the screen. When the student is ready, he can clear this message and enter additional commands.

The valid commands are:

±nnnnn - The signed five digit number is put into the word whose address is the current value of IC and IC is incremented by 1. If the sign is omitted, + is assumed.

LOAD nn - IC is set to nn, OF, UF, and EOF are initialized to 0, and the input line is positioned at the first "card" of data. If nn is omitted, 00 is assumed.

RUN nn - IC is set to nn and the program is run to termination or until an execution error occurs. If nn is omitted, the current value of IC is retained.

CY - One machine cycle is executed.

QUIT - The HYCOMP1 simulator is terminated.

DATA - Allows data to be entered, one "card" at a time. A card with # in column 1 signals the end of data entry and serves as the EOF flag.

DATA ADD - Allows data to be added to the end of the existing data "deck". The existing EOF card is automatically removed.

HELP - A list and brief description of the commands is displayed.

HYCOMP1 has not yet had extensive classroom use. We did have several students from an introductory programming class use it during the latter stages of its development. Earlier in the semester, these students had studied HYCOMP and had written several HYCOMP programs which were graded by a human grader. Their reaction to HYCOMP1 was extremely favorable and they reported that it would have been very helpful in their study of machine language. One student finally understood that there is no inherent difference between instructions and data. This kind of insight, if not gained intellectually, could hardly be gained by any means other than an interactive, visual simulation.

An interactive, visual simulator for a pedagogically typical computer has been described. This simulator is to be used during a brief introduction to machine language programming within an introductory programming course to give the students a "feel" for how the machine works at that level. It was written in SNOBOL4 and runs interactively using a CRT terminal.

The author thankfully acknowledges David A. Grace, who did the programming for HYCOMP1.