Xsim68k

68000 Microprocessor Simulator
for the *X Window System*

b y
Jay Lloyd

Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, N.C. 1991.

INTRODUCTION

The software package, Xsim68k, contains an assembler and simulator for the Motorola 68000 microprocessor. This package allows the student to create an assembly language file, assemble the file into an object module, then monitor execution of the program using the simulator.

Features of the simulator include:

- Graphical user interface via the X Window System
- Interactive access to the microprocessor's programmers model (ie. registers, stack pointer, program counter)
- Interactive access to simulator's memory during execution
- Commands to trace, single step, set break points and dump simulator contents.

The purpose of this document is to describe the Xsim68k software package and to explain the procedures and commands needed to run a simulation.

DESCRIPTION

Assembler

The assembler, invoked by the command 'asm', is a two pass assembler which is used similar to other UNIX compilers and assemblers. The assembler supports most standard assembly language directives such as: DC, DS, EQU, ORG, END, and labels. The assembler does not support the XREF, XDEF, and MACRO directives. Notational conventions include: '$' for hexadecimal numbers and '#' for constants. When invoked with the -l option, asm will produce a listing file in addition to the object file. The object file format is ascii representation of Motorola S-records.

Simulator

The simulator, invoked by the command 'sim', opens a window displaying the programmer's model for the 68000 microprocessor. Commands are now available to load an object file into memory, alter the contents of registers or memory locations, set break points, and run, step, or trace through the execution of a program.

OPERATION

Assembler

The assembler operation is similar to many UNIX system compilers and assemblers. The first step is to create an assembly language file in your favorite UNIX text editor. This file should have the format:

LABEL OPCODE OPERANDS COMMENTS

The LABEL field is optional. Fields need only to be separated by a space but tabs are recommended. Tabs align the fields to make the program more readable. Listing 1.1 provides an example assembly language file which uses assembler directives.

```
        ORG     $2000       Start at location 2000 hex
START   CLR.W   SUM         Clear variable SUM
        MOVE.W  ICNT,D0     Load value for CNT
ALOOP   MOVE.W  D0,CNT      Save value of CNT
        ADD.W   SUM,D0      Add to SUM
        MOVE.W  CNT,D0      Update CNT
        ADD.W   #-1,D0
        BNE     ALOOP       Loop if CNT not zero
        JMP     SYSA        Go to operating system when done
SYSA    EQU     $8008       Return address to operating system
CNT     DS.W    1           Reserve one word for CNT
SUM     DS.W    1           Reserve one word for SUM
IVAL    EQU     17          Initial value for CNT
ICNT    DC.W    IVAL        Store initial value
        END     START
```

Listing 1.1: Sample assembly language file.

Once the file has been edited, use the command: **asm** <filename> to assemble the file. The assembler will indicate if any errors were found. If no errors were found then the assembler creates an object file with the suffix .H68. If errors were found, to get a better idea of the errors assemble the file using the -l option: asm -l <filename> . This option tells the assembler to produce a listing file with a .LIS suffix in addition to the .H68 file. A sample program listing is provided in listing 1.2.

```
00002000                        1          ORG       $2000
00002000  4279 0000202C       2   START  CLR.W     SUM
00002006  3039 0000202E       3          MOVE.W    ICNT,D0
0000200C  33C0 0000202A       4   ALOOP  MOVE.W    D0,CNT
00002012  D079 0000202C       5          ADD.W     SUM,D0
00002018  3039 0000202A       6          MOVE.W    CNT,D0
0000201E  0640 FFFF           7          ADD.W     #-1,D0
00002022  66E8                8          BNE       ALOOP
00002024  4EF9 00008008       9          JMP       SYSA
0000202A  =00000000           10  SYSA   EQU       $8008
0000202A                      11  CNT    DS.W      1
0000202C                      12  SUM    DS.W      1
0000202E  =00000000           13  IVAL   EQU       17
0000202E  0011                14  ICNT   DC.W      IVAL
00002030                      15         END       START


No errors detected
No warnings generated
```

Listing 1.2: Assembler listing.

Once the file has been assembled with no errors, the object file can be loaded into the simulator for execution.

Simulator

The simulator must be run from an X Window terminal. To start the simulator type the command: **sim**. Wait for the simulator window to appear. This may take up to thirty seconds or more so be patient. All simulator operations require some delay due to the processing required of the X Window System. The simulator window can not be resized but may be moved around the screen. Figure 1.1 shows a partial view of the simulator window.

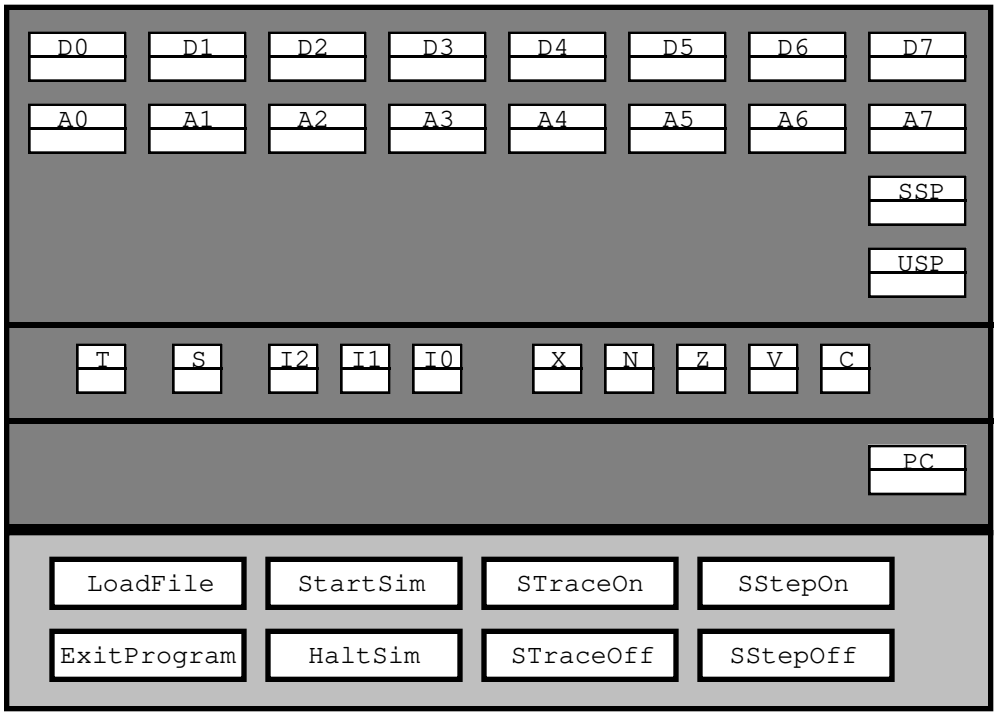

Figure 1.1: Simulator Window.

General Operation

Mouse. The mouse may take three different shapes: 1) X-man, 2) pointed-finger, 3) up-arrow, depending on its screen location. The X-man shape indicates no function or feature is available at the current mouse position. The pointed finger indicates the mouse is over a button which can be selected by clicking the left mouse button. The up-arrow indicates the mouse is over an item which can be edited by clicking the left mouse button. Editing is performed from the keyboard and is terminated by clicking the left mouse button or pressing the return key.

Buttons. A button will be highlighted to indicate selection. Once a button is selected other buttons may change from gray to white to indicate their operation is available. A grayed button's operation is not available. The buttons perform the following operations:

- **LoadFile**: displays a dialog box for entering the name of an object file to be loaded
- **ExitProgram**: halts sim and disposes of the window

- **StartSim**: begins instruction execution at the current pc location
- **HaltSim**: halts instruction execution
- **STraceOn**: updates screen after each instruction is executed
- **SStepOn**: sets run mode to execute one instruction at a time
- **TakeStep**: executes the next instruction
- **ViewMemory**: opens the memory display window
- **ClearSim**: resets all simulator values to zero
- **Dump**: creates a file containing simulator contents
- **Interrupt**: allows simulation of an interrupt by entering an infinite loop, then pressing the button.

Memory Window. The memory window can be used to set and clear break points, view memory contents, and edit memory locations. Different blocks of memory can be displayed by clicking on the addresses along the left of the window and typing the start address of the new block. Memory locations are selected for editing or break points by clicking on the actual location value.

Procedure. Once the window is open click LoadFile to load the object module. Open the memory window and set break points. Next set the pc to the starting address of the program. Set initial values for any of the registers as needed. To single step, select SStepOn which automatically selects screen tracing. Now click StartSim to begin simulation and then click TakeStep to execute the next instruction. For normal execution select StartSim which will execute until a break point is reached.

Acknowledgements

This package was developed as part of a senior project. I included the 68000 execution model. The other pieces were already developed.

The X Window Interface is based on a general microprocessor simulation environment called the SI Toolkit. The assembler was written by a previous senior project.

SI Toolkit. Tan Phan
Department of Electrical and Computer Engineering, North Carolina State University, 1989.

asm. Paul McKee
Department of Electrical and Computer Engineering, North Carolina State University, 1986.