EASIM – A SIMULATOR FOR THE 68HC11 BASED HANDYBOARD

David Edwards[1]

Abstract *¾ Microprocessor development systems are used in the teaching of computer engineering but have relatively high capital costs and complex modes of operation which tends to limit their use in introductory level classes. A software simulator, EaSim, is described that illustrates both the Motorola 68HC11 microprocessor and the MIT developed HandyBoard development system. Simulation giving each student ready access to their own development system. EaSim features enhance the learning process, allowing the undertaking of more complex, and real world related, exercises. These include: the viewing of register and memory contents after each operation, single stepping through a program, executing instructions considerably more slowly than with a real hardware system, integrated debugging and trace facilities, and an integrated editor-assembler. EaSim also simulates the operation of external hardware devices, such as the Bilby 'maze mouse', when connected to the HandyBoard. Evaluation of the present package has been undertaken with students responding favorably to the use of the simulator.*

Index Terms ¾ Computer engineering, HandyBoard, Simulation.

INTRODUCTION

Computer engineering is introduced in the first semester of the Bachelor of Information Technology (BIT) and the Bachelor of Engineering in Electronic Engineering (BEng) degrees at Griffith University through a course entitled 'Microprocessors'. This course has an enrolment of 185. Within the course students are introduced to the concepts of a microcomputer and a microprocessor development system.

The course forms a foundation for further computer engineering studies for the BEng students. For the BIT students the use of simple development system monitor routines introduces the functional requirements of operating systems; the use of a simple assembler introduces concepts of compilers used in later courses. The constraints associated with assembly language programming introduces students to the need to carefully plan and document programming tasks.

The laboratory component of the course is based on the Motorola MC68HC11 8-bit microcontroller in the HandyBoard Development System. The HandyBoard [1] was designed by Massachusetts Institute of Technology (MIT) for controlling small robots. The HandyBoard now has its own website for support [2].

FIGURE. 1
HANDYBOARD DEVELOPMENT SYSTEM.

The board includes analog inputs, digital I/O including motor drivers, a 32 character LCD screen, 32kB of user RAM, and a serial link. Although the HandyBoard comes with an inbuilt C language interpreter, this is not used with this course. A colleague, Charles Hacker, has written a work-alike version of the Buffalo [3] monitor program supplied with Motorola 68HC11 development boards [4].

EaSim (Editor/assembler and Simulator) is an integrated assembly language editor/assembler and simulator for the MC68HC11 microcontroller running in the HandyBoard environment. Most of the features of the HandyBoard are also simulated.

WHY SIMULATE

The assessment for the course includes the requirement for the students to develop, implement and demonstrate programs that interact with a hardware device - a small robot

To give students in the course adequate development time on a hardware based system would require an expensive outlay for development systems and laboratories to house them. Using a simulator allows tutorial/laboratory classes to be taken in a computer laboratory. Students can access the simulator program from any of the campus computer labs or Learning Centres. In addition, students enrolled in the Microprocessors course are permitted to make a copy of the program for use on their own PC. This considerably reduces the demand for computer laboratory access. For the last three weeks of the course students are given access to hardware development systems to test the operation of their assigned tasks under the hardware timing restrictions.

[1] David Edwards, Griffith University, School of Engineering, PMB 50, Gold Coast Mail Centre, 9726 Queensland, Australia d.edwards@mailbox.gu.edu.au

A simple hardware microprocessor development system like the HandyBoard is a relatively unfriendly environment for software development. The lack of any feedback as the program is running makes debugging extremely difficult for novices. A simulator program can provide enhanced feedback opportunities.

A decision was taken to implement a computer simulation, called EaSim, of the development system that would enhance the software system development process as well as lead to a better understanding of the operation of a microcomputer. At the time there was no suitable simulator available. Today there are many microprocessor simulators available [5], mainly for the Intel family of microprocessors.

The special feature of EaSim, differentiating it from many other simulators, is that it simulates the operation of the microprocessor in a real hardware environment. The commercial simulator for the 68HC11 from vmdesign also does this to a lesser extent [6].

STUDENT ASSESSMENT

The assessment for the Microprocessors course includes assignments involving assembly language programming. The main assignment involves designing and implementing a program to run on the actual HandyBoard to control the movement of a small stepper motor driven robot using infra red sensors to follow a painted track.

Two earlier assignments are wholly simulator based involving sub-tasks of this assignment. The first is a 68HC11 programming task. The second involves programming the 68HC11 in the simulated HandyBoard environment. As students are given all three assignments tasks at the beginning of the course, they can see the relevance of the simplified early tasks.

THE SIMULATED HARDWARE

The MC68HC11 is an industry standard 8-bit microcontroller running at a 2MHz clock speed. It has two 8-bit data accumulators (which can be treated as a combined 16 bit register), an 8-bit status register, and four 16-bit address registers: program counter, stack pointer and index registers. Incorporated are serial communications, digital I/O and analog input ports.

The HandyBoard environment provides a number of I/O capabilities. These include a serial link to a terminal, a two line 32 character LCD screen, two push buttons and 6 digital input lines, 8 digital output lines linked to 8 LEDs in parallel with DC motor drivers, a buzzer, and a rotary knob input.

The HandyBoard provides 32kB of RAM at the top of the memory map. This is divided in our application into 1kB of user RAM (for program development) the remainder being used for the Monitor program. The limited amount of user RAM for data, program and stack is seen as an advantage for an introductory course. It forces students to be concise in their code, making them very aware of the need to structure data, code and stack space. This in turn encourages the student to undertake very careful design and desk checking before coding.

Also simulated is the small mouse-like robot, called a Bilby [7], used in the final assignment. Students can check their program operation in this simulated environment before transferring to the real hardware.

SIMULATOR HISTORY

The simulator is a Windows based program, the development of which has been a semi-continuous process over the last 16-17 years. The original version was written as DASM, an editor/assembler for the MC6802, and SIMUL-02, the D3 simulator program, while the author was with the University of Southern Queensland. These programs were written in Turbo Pascal Version 2 to run in a CP/M-86 environment. They were later modified to run under DOS on IBM PCs. In later years a screen editor was added and the two programs combined into the one package.

For 1997 the simulator was completely rewritten in Delphi as EaSim, a Windows package. For 1999 the D3 development boards were replaced with the 68HC11 based HandyBoards. EaSim was updated to simulate the new processor and hardware environment. Each year EaSim is updated to simulate the particular 'add-on' hardware being used for the major assignment.

EASIM FEATURES

EaSim consists of an integrated Editor/Assembler as well as the Simulator for the operation of a 68HC11 microcontroller operating in the HandyBoard hardware environment with external hardware devices connected. The simulator has been designed to enhance the learning process. Right from the earliest trial, it was decided to make visible information such as memory and register contents that is not normally readily viewable in a development system to help students understand the operation of a microcomputer.

The decision was taken to display all data in hexadecimal as this is the format used by the HandyBoard system for keyboard entry through a terminal.

Editor/Assembler

EaSim contains an integrated editor and assembler for the 68HC11. This integrated editor and assembler allows programs for the HandyBoard to be developed as assembly language programs. These programs can either be tested in the simulated environment or an S19 object code file is produced for downloading to the 'real' hardware for testing.

Features of the Editor/Assembler section of EaSim include:

- Warning messages being produced if reference is made

to addresses outside the User section of the HandyBoard memory map. These warnings can be disabled.

- Extensive context sensitive on-screen help. The availability of such help is a feature of the whole of EaSim. The on-screen help with the editor includes help on the 68HC11 instruction set.
- The instruction set of the 68HC111 can be called up at any time.
- There is on-line help for using the Editor/Assembler.
- If the assembler reports an error, the on-screen help suggests the most likely cause of that error.
- All files including object code files and handled as plain .txt files. This means other, more sophisticated editors can be used for source code writing. Students may import these files into word processors for enhancing for assignment documentation.

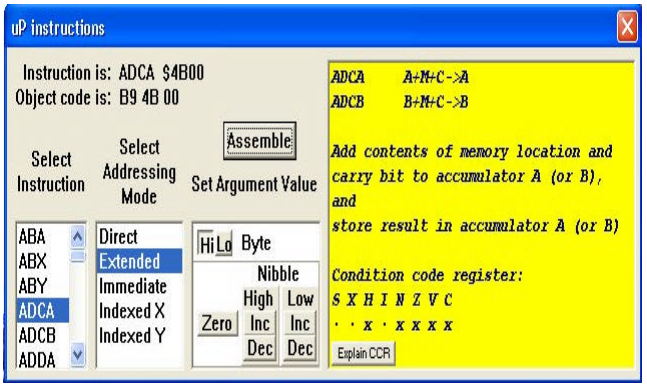

FIGURE. 2
ASSEMBLER INSTRUCTION HELP SCREEN.

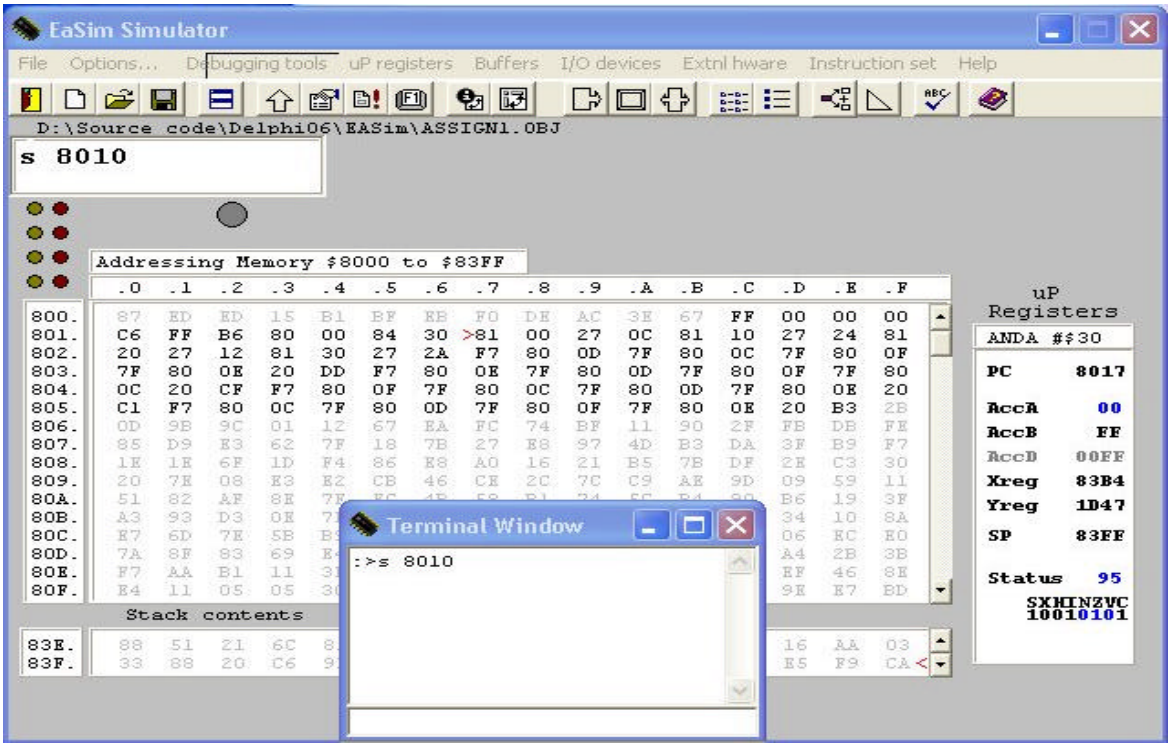

FIGURE. 3
MAIN EASIM SIMULATOR SCREEN.

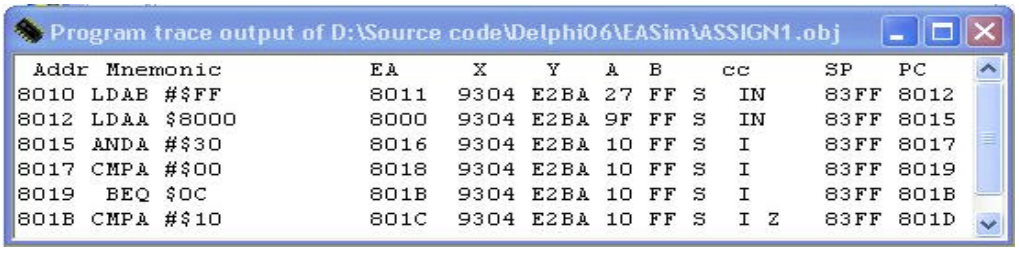

FIGURE. 4
PROGRAM TRACE SCREEN.

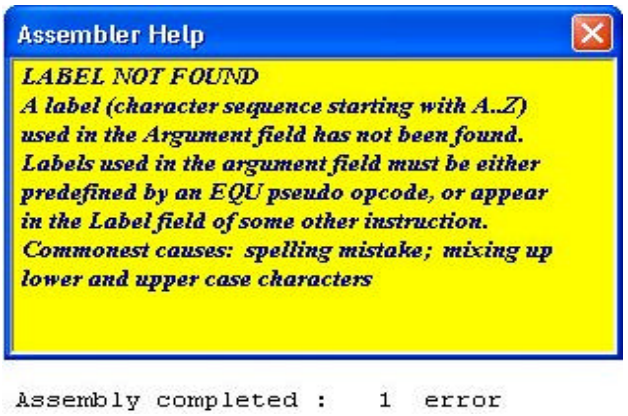

FIGURE. 5
ON-SCREEN HELP FOR ERROR.

Simulation of the 68HC11

EaSim simulates the following for the 68HC11:

- All the instruction set of the 68HC11
- The operation of hardware interrupts
- Allows object code to be loaded into user RAM and programs executed
- Shows the contents of some of the internal control registers
- Shows contents of the processor's register contents as programs are executed

FIGURE. 6
CONTENTS OF INTERNAL REGISTERS.

In addition to the above 68HC11 features, there a number of enhancements to the EaSim display:

- Context sensitive on-screen help is available.
- Each instruction is disassembled as it is executed
- Programs can be executed a single instruction at a time
- A 'Debug' option links the microprocessor instruction execution back to the assembly language source. As each instruction is executed the instruction is 'highlighted' in the on-screen assembler listing.
- A 'Trace' facility allows the viewing of the most recent (up to 100) instructions executed. This trace can be printed out. Figure 4 shows an example of a program trace report.
- The execution speed of a machine language program can be varied. The program can be set to execute until conclusion, or a breakpoint, at a speed adjustable between 0.5 and 20 instructions per second. This speed is determined by the Windows 'clock', so the speed is PC speed independent. A further option allows the program to run as fast as the PC permits.
- Allows the user the option of choosing any instruction from the instruction set, setting the arguments for that instruction and executing it with the current microprocessor register contents.
- The status register is decoded into the individual status flags.
- Register contents that are set (or cleared) by the last instruction are shown highlighted.
- The contents of any of the registers can be changed at any time.

Simulation of the HandyBoard

Features of the HandyBoard that are simulated:

- 1kB of user RAM.
- The operating system (monitor program) buffers in system RAM.
- A number of the monitor I/O subroutines including the reading of a keyboard key and the writing to the LCD screen. These can be called from a user program.
- The screen and keyboard of the terminal connected to the HandyBoard. Interaction with the HandyBoard is via a terminal connected to the serial port. The keyboard of the PC EaSim is running on acts as the terminal keyboard. A window acts as the terminal screen.
  An option, useful in the lecture environment, allows all keyboard keystrokes to be shown on screen.
- The two lines of 16 characters LCD screen.
- The 4 pairs of red and green LEDs in parallel with the DC motor drives.
- The 8 digital input lines.
- The two input push buttons (connected to the digital inputs).
- The analogue inputs.
- A rotary knob (connected via the analog inputs)

A number of enhancements intended to increase the 'user friendliness' of the development environment have been made to the EaSim display of the HandyBoard. These enhancements include:

- In addition there is on-screen help for Using the Simulator, Monitor program commands, and on the accessible Monitor program routines.
- In addition to showing the contents of each memory

location in RAM are always on screen. Contents set by the user are shown in bold, contents not set by the user are shown in feint. (All memory contents are randomized at start up).
- Any data changed by the execution of the last instruction by the microprocessor are shown highlighted.
- The current location of the program counter is shown against the memory location as '>'.
- The current location of the stack pointer is shown against the memory location as '<'.
- An option allows the display of the contents of the system RAM 'buffers' used by the I/O monitor routines, these include the keyboard input buffer, the LCD screen display buffer, and the interrupt vectors. Only those system RAM locations students should be accessing are displayed. The 'hidden' locations are shown as '..'.
- Contents of a memory location can be changed (by a right mouse button click on the memory display screen) while user programs are executing. This allows user programs to interact with changing data.

Simulation of external hardware

In addition to the HandyBoard environment, EaSim also includes a simulation of a small mouse-like robot, the Bilby. This is a two wheeled stepper motor driven device with two IR sensors for locating a track. The simulated Bilby can either be operated stand alone, so students can get the feel of its characteristics, or it can be 'connected' to the simulated HandyBoard. In this mode the stepper motors are driven from the user program and the position of the Bilby, as 'seen' by the IR sensors, in relation to its track is feedback via the digital inputs.

STUDENT FEEDBACK

Informal feedback from students and fellow staff had led to a number of refinements of the DOS based program.

The introduction in 1997 of the Windows version of the program was used as the opportunity to undertake a more formal evaluation. Students were surveyed by questionnaire at the midpoint and the end of the course. This has been followed up on an annual basis since.

All evaluations have shown a very positive response to the use of the simulator program. Students also have the opportunity to suggest improvements to EaSim. It is interesting to note that as improvements have been made to EaSim in response to this feedback, different 'problems' will be thrown up the following year.

The major negative responses have always relatde to the speed of operation of the simulator. The restriction to 20 instructions per second was felt to be too slow for the major assignments making programs too slow to execute. An option was added to the execution speed control to allow the program to run at a platform limited speed. (On a 1GHz Pentium 4 the execution speed is approximately 200 instructions per second. This relatively slow speed is thought to be caused by the extensive changes that are made to the visual display during the execution of each instruction.)

A more major problem than the time it takes large programs to execute is that the simulator runs at an unrealistically slow speed compared with the real hardware. As the real hardware runs at around 500,000 instructions per second, the simulator is operating at only 0.04% of that speed. This causes problems in the transition from a simulated environment to the hardware environment.

In previous surveys, students had reported that they made little use of the extensive Help that was provided in the package. For 2002 the separate Help files were incorporated into a built-in context sensitive 'On-screen' help feature.

SIMULATOR DETAILS

Originally developed using Borland Delphi 1, recently EaSim has been ported to Borland Delphi 6. This 32bit environment removes many of the restrictions on file size previously in EaSim. EaSim will be updated for 2003 to take advantage of the removal of these restrictions.

EaSim can be run on any PC under Windows 95 or later. However, due to the highly visual nature of EaSim, the program works best on a PC with a Pentium processor. The program has been developed to run on as wide a range of PCs as possible. To facilitate this, the main screen size is set to 640x480. This facilitates running on student laptops. The executable code size is 1112 kilobytes.

All files created by EaSim, the assembler source file, the assembler list file, and the object code file are plain text files. Students can either print these out or import them into a word processor for 'polishing' for assignments. The simulator screen and the trace file can also be printed out.

As there is quite a lot of detail on the screens, a 'Lecture Theatre' option within the program increase the font size of the main text based information. This uses larger fonts and rearranged screen layouts to facilitate viewing from video projected images.

EaSim is available in three versions which all have the same features except where noted:
- The student version which has an expiry date at the end of the semester. The student version defaults to having the on-screen help always visible.
- A demonstration version which does not allow the saving of files created in the Editor/Assembler and does not have context sensitive help. This version of EaSim is available for downloading from the Griffith University School of Engineering website (http://www.gu.edu.au/school/eng/mmt/MMTdownlds.html).
- A limited life version which is fully featured and can use for suitability evaluation.

FUTURE DEVELOPMENT

Development underway with EaSim includes the removal of the 400 lines of source code limit imposed by the '64kB block limit' of the original 16bit development environment. Minor changes are being made to make the simulated Bilby behave more exactly like the real hardware.

An interface to the parallel printer port is planned (limited to running under Windows 98 or earlier) which will have a transfer box connected to the printer port behave as the external hardware connection to the HandyBoard. This, fro example, would allow the hardware Bilby to be directly driven by EaSim. This addition would also simplify future HandyBoard hardware extensions development.

A 'stripped down' option is planned. This would turn off many of the debugging features to reduce changes to the display. Such an option should allow an increase in the speed at which EaSim can execute a 68HC11 instruction at the cost of less frequent screen updates.

CONCLUSION

EaSim has proved to be a very effective tool for introducing students to microprocessor operations and assembly language programming. Having an integrated editor-assembler-simulator allows students to move seamlessly from code development to program testing. Incorporating the simulation of a development system environment as well as add-on hardware devices has greatly increased flexibility. Without such a simulation package, the costs of development systems and laboratory time would prohibit allowing students to undertake 'hands-on' assignments.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the generous assistance provided by his colleagues in the development of the EaSim package. Charles Hacker developed the keyboard entry routines, the Terminal Emulation package used to link EaSim to the real HandyBoard and the Buffalo work-alike that runs as the HandyBoard monitor program.

REFERENCES

[1] http://lcs.www.media.mit.edu/groups/el/projects/handy-board/

[2] http://handyboard.com/

[3] Miller, G.H, *Microcomputer Engineering*, Prentice Hall, 1993

[4] http://www.gu.edu.au/school/eng/mmt/HBoard.html

[5] http://www.samphire.demon.co.uk/

[6] http://www.vmdesign.com/

[7] http://www.usq.edu.au/users/phythian/BILBY.HTM