Int. J. Engng Ed. Vol. 15, No. 5, pp. 353–357, 1999
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 1999 TEMPUS Publications.

Use of Simulator in Teaching Introductory Computer Engineering*

DAVID EDWARDS
School of Engineering, Griffith University, Gold Coast Campus, PMB 50, Gold Coast Mail Centre, Qld 9726 Australia. E-mail: d.edwards@mailbox.gu.edu.au

Microprocessor development systems are used in the teaching of computer engineering classes. As standalone devices they have relatively high capital costs and complex modes of operation. Both these features tend to limit their use in introductory level classes. It is in these classes that students probably have most need of detailed assistance as they begin their studies of machine and assembly language operations. Implementation of a software analogue of the system, a simulation, means that each student has easy access to their own development system. Enhancements in software can be added that aid the learning process. The use of a simulator allows students to undertake more complex and real world related exercises than would otherwise be the case. To date a simulator has been developed for the Motorola 8-bit microprocessor and development system currently in use in the undergraduate laboratory. This is being extended to include the Motorola 68HC11 8-bit microcontroller in the MIT Handyboard environment. Investigations have shown that the teaching and learning process is enhanced by the added features: the ability to see register contents after each operation, the ability to single-step through a program, the ability to execute instructions considerably more slowly than with a real hardware system, the inclusion of integrated debugging and trace facilities, and the inclusion of an integrated editor assembler. Evaluation of the present tool has been undertaken. Students have responded favourably to the use of the simulator. Suggestions for enhancements have been made.

INTRODUCTION

COMPUTER ENGINEERING is introduced in the first semester of the Bachelor of Information Technology (BIT) as well as being in the second year of the microelectronics strand of Bachelor of Engineering (B.Eng.) through a subject entitled 'Microprocessors'. This subject has an enrolment of 75. Within the subject students are introduced to the concepts of a microcomputer and a microprocessor development system.

The subject 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 of assembly language programming introduces students to the need to carefully plan and document programming tasks.

The laboratory component of the subject is based on the Motorola MC6802 8-bit microprocessor in the Motorola D3 Development System. The Development System is memory sparse. It includes only 256 bytes of user memory, a hex keypad for program and data entry, and a multiplexed eight-character seven-segment LED display for output. The system comes with a monitor routine D3BUG in 2K of ROM. The monitor includes subroutines that may be accessed to allow user programs to read from the keyboard and write to the display.

EaSim (*E*ditor/*a*ssembler and *Sim*ulator) is an integrated assembly language editor/assembler and simulator for the MC6802 microprocessor running in the D3 development system environment.

WHY EASIM?

The assessment for the subject includes the requirement for the students to develop, implement and demonstrate programmes that read data from the keyboard, output to the display and respond to interrupts.

A simple hardware microprocessor development system is a relatively unfriendly environment for software development. The lack of any feedback as the programme is running makes debugging extremely difficult for novices.

A decision was taken to implement a computer simulation 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 the project was started there was no simulator for the MC6802 available. Today there are many microprocessor simulators available.

The special feature of EaSim, differentiating it from other simulators, is that it simulates the operation of the microprocessor in a real hardware

* Accepted 20 May 1999.

environment. The commercial simulator for the 68HC11 from Vmdesign also does this [2].

To give these students 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 enrolled in the microprocessors subject are permitted to make a copy of the programme for use on their own PC. This considerably reduces the demand for computer laboratory access. For the last three weeks of the subject students are given access to the hardware development system to test the operation of their assigned tasks under the hardware timing restrictions.

STUDENT ASSESSMENT

The assessment for this subject includes examinations covering the theory plus assignments involving assembly language programming. The main assignment involves designing and implementing a programme to run on the D3 hardware.

In 1997 for example, the main assignment task was to organise the display of the message ⟨CAFÉ⟩ on the eight 7-segment LED displays. The message was to appear from the left and slide across the display at a specified rate. Once the message had disappeared it was to reappear from the left. When an interrupt button was pushed the message was to flash on and off next time it was centred. The number of times it was to flash was entered from the keyboard at start-up. A second press of the interrupt button was to cause the direction of 'rotation' of the message to reverse.

Two earlier assignments were wholly simulator based involving sub-tasks of this assignment. As students are given all three assignments tasks at the beginning of the subject, they could see the relevance of the simplified early tasks.

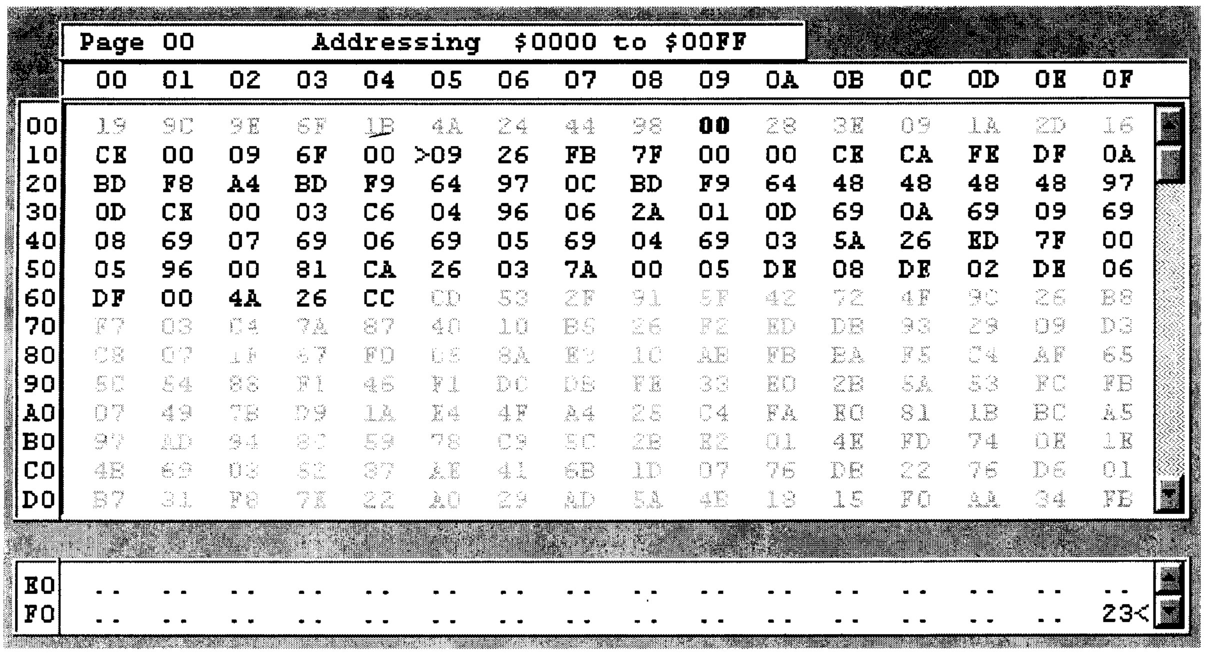
All assignments are assessed in two stages. The running program is demonstrated and marked for achievement of specifications in lab/tutorial time, then the write up is assessed separately. The use of a printed marking sheet/marking scheme, means the 80 students can be assessed for an assignment in 4 hours of lab/tutorial time and about 3 hours of report marking time.

A teaching problem with microprocessor programming assignments is to keep the level of complexity low enough to allow all students to achieve the task while making it non trivial for the better students. In the subject this is achieved by having a base task with bonus features to be implemented if students wish to achieve higher grades. The subdivision of the task into three consecutive assignments allows early assignments to be set with simple requirements which build into the more complex final task.

THE HARDWARE SIMULATED

The MC6802 is an 8-bit microprocessor running at a 1 MHz clock speed. It is an early generation device in a microprocessor family that includes the 68HC11 microcontroller. It has two 8-bit data accumulators, an 8-bit status register, and three 16-bit address registers: program counter, stack pointer and index register.

Although the microprocessor is old technology, it is upwardly software compatible with the



```
Page 00          Addressing  $0000 to $00FF
    00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
00  19  9C  9E  6F  1B  4A  24  44  98  00  28  3E  09  1A  2D  16
10  CE  00  09  6F  00 >09  26  FB  7F  00  00  CE  CA  FE  DF  0A
20  BD  F8  A4  BD  F9  64  97  0C  BD  F9  64  48  48  48  48  97
30  0D  CE  00  03  C6  04  96  06  2A  01  0D  69  0A  69  09  69
40  08  69  07  69  06  69  05  69  04  69  03  5A  26  ED  7F  00
50  05  96  00  81  CA  26  03  7A  00  05  DE  08  DE  02  DE  06
60  DF  00  4A  26  CC  CD  53  2F  91  5F  42  72  4F  9C  26  B8
70  F7  03  C4  7A  87  40  10  B5  26  F2  ED  DB  93  29  09  D3
80  C8  07  1F  67  FD  C5  8A  E2  1C  AB  FB  EA  F5  C4  AF  65
90  5C  54  88  F1  46  F1  DC  DB  FE  33  E0  2B  5A  53  FC  FB
A0  07  49  7B  D9  1A  E4  4F  A4  25  C4  FA  E0  81  1B  BC  A5
B0  97  AD  94  8C  59  78  C9  5C  2B  E2  01  4E  FD  74  0E  1E
C0  4B  69  03  52  37  AE  41  6B  1D  07  76  DB  22  76  D6  01
D0  B7  31  F8  7E  22  A0  29  AD  5A  4B  13  15  F0  AA  34  FB

E0  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
F0  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  23<
```

Fig. 1. Memory Display showing PC location > and SP location <. Memory contents shown in feint are random, switch-on values, and have not been set by the user. Memory contents shown in bold have been affected by current instruction.

industry standard 68HC11 microcontroller. 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, and makes 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.

The display is time multiplexed; with each run through of the display routine a single digit is flashed. To stop the LED display from going out, the display routine must be run through at least $8 \times 25 = 200$ times a second. This need to service hardware requirements is a common requirement in a real embedded application.

SIMULATOR DEVELOPMENT

The simulator is a PC-based program, the development of which has been a semi-continuous process over the last 13–14 years. The original version was written as DASM, an editor/assembler for the MC6802, and SIMUL-02, the D3 simulator programme, while the author was with the University of Southern Queensland. These programmes were originally 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 programmes combined into the one package.

For 1997, the Griffith University's general access computer laboratories, where tutorials and early laboratory classes were taken, stopped supporting DOS applications. For that year, the simulator was completely rewritten as EaSim, a Windows package in Delphi.

SIMULATOR FEATURES

The simulator has always been designed to enhance the learning process. Right from the earliest trial, it was decided to display information that would help students understand the operation of a microcomputer. All data is shown in hexadecimal. This is the format used by the D3 system for keyboard entry and for LED display.

In addition to showing the contents of each memory location in RAM, the microprocessors register contents are always on screen. Facilities exist for executing a programme step by step or letting it run. When a programme step, instruction, is executed, the microprocessor register contents are all updated appropriately. Any registers that have changed are highlighted. Any data that is written to memory by the instruction is shown highlighted.

EaSim simulates the operation of the 6802 microprocessor in the D3 environment. All of the 6802 instruction set, the D3Bug monitor routines that deal with input and output, the operation of the keyboard and the LED display are simulated.

In addition to the simulator for the D3 environment, EaSim contains an integrated editor and assembler for the 6802. The integrated editor and assembler allows programs for the D3 to be developed as assembly language programs and tested in the simulated D3 environment. Machine code can be directly entered into the simulated D3 environment if desired.

A download feature allows developed code from the simulator to be loaded into the D3 development systems for real time testing.

Enhanced features of the simulator

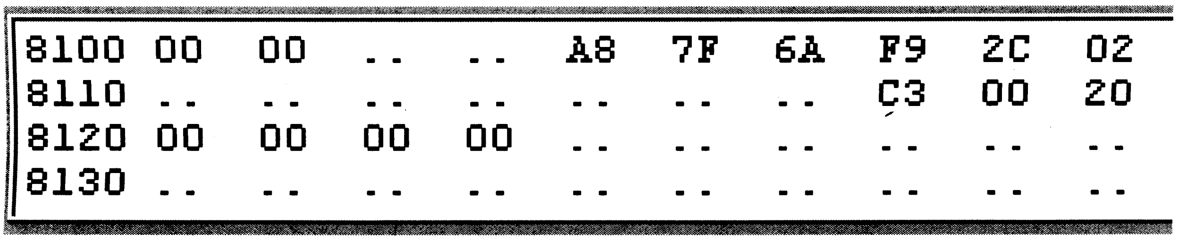
The EaSim simulator has a number of enhancements over the actual hardware D3 Development System.

A feature of the **display** is that the hexadecimal contents of all 256 bytes of user RAM are always visible. Values set by the user stand out from initial switch-on values. As a program is executed, the contents of any memory location written to by an instruction are temporarily highlighted.

As shown in Fig. 1, 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 <.

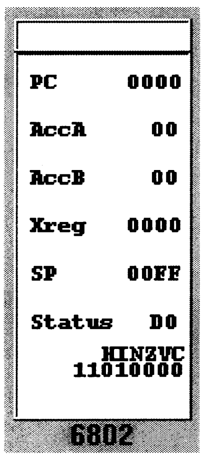
An option allows the display of the contents of the system RAM buffers used by the I/O monitor routines, these include the keyboard input buffer, the hexadecimal data display buffer, the 7-segment display codes output buffer, and the interrupt vectors. Only those system RAM locations students should be accessing are displayed. The hidden locations are shown as '..'.

The **microprocessor register** contents are continuously displayed. The status register is decoded to the six status flags and either of the two 8-bit accumulators can be displayed in binary as well as hex. As the program runs, the status



```
8100 00  00  ..  ..  A8  7F  6A  F9  2C  02
8110 ..  ..  ..  ..  ..  ..  ..  C3  00  20
8120 00  00  00  00  ..  ..  ..  ..  ..  ..
8130 ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
```

Fig. 2. Part of system RAM area display.



```
PC        0000
AccA        00
AccB        00
Xreg      0000
SP        00FF
Status      D0
        HINZVC
      11010000
      6802
```

Fig. 3. Microprocessor register display.

flags that are affected by an instruction are highlighted, as are the contents of any register affected by the instruction.

A number of options to facilitate **debugging** are included. As each instruction is executed, the disassembly of the instruction is shown.

A debug option relates the microprocessor operation back to the assembly language source. As each instruction is executed the instruction is highlighted in the on-screen assembler listing.

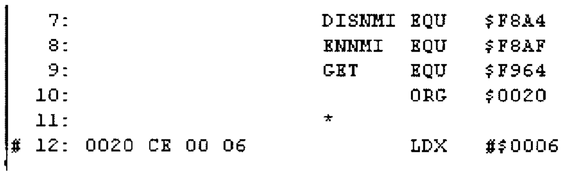
A trace facility allows the viewing of the most recent instructions executed. This trace can be printed out.

The **execution speed** of a machine language program can be varied. The program can be stepped through one instruction at a time. Alternatively 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.

There is extensive **on-line help** for the assembler and simulator parts of the programme. This includes the full microprocessor instruction set, a description of hardware of a D3 system, and a description of the operation of the I/O monitor routines.

STUDENT FEEDBACK

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



```
  7:                      DISNMI  EQU   $F8A4
  8:                      ENNMI   EQU   $F8AF
  9:                      GET     EQU   $F964
 10:                              ORG   $0020
 11:                      *
# 12: 0020 CE 00 06               LDX   #$0006
```

Fig. 4. Current instruction linked back to assembler list file.

The introduction in 1997 of the Windows version of the programme was used as an opportunity to undertake a more formal evaluation. Students were surveyed by questionnaire at the midpoint and the end of the subject. A summary of relevant student responses is included in the Appendix.

Both evaluations gave a very positive response to the use of the simulator programme. The major negative response related to the speed of operation of the simulator. The restriction to 20 instructions per second was felt to be too slow for the later assignments and they took too long to execute. An option was added to the speed control to allow the programme to run at a platform-limited speed.

A more major problem was that the simulator runs at an unrealistically slow speed compared with the real hardware. As the real hardware runs at around 250,000 instructions per second, the simulator is operating at 0.01% of that speed. This makes setting up the display on the D3 more difficult.

A further negative response was that the simulator does not simulate correctly the operation of the 7-segment display. (The multiplexed D3 display strobes a character on for 1 ms each time through the display routine. To keep the display illuminated, the display routine must be executed at least 8 times every 1/25 of a second.) Various options were tried, but there appears to be no easy solution to this problem.

In the survey, students reported that they made little use of the extensive Help that was provided in the package. As students were shown how to use the package in tutorials, it may have been a case of 'if all else fails, read the instructions'.

SIMULATOR DETAILS

EaSim was written in Borland Delphi 1. It can be run on any PC under Windows, Windows 95 or Windows NT. The program has been developed to run on as wide a range of PCs as possible. To facilitate this, the screen size is set to 640480. This facilitates running on student laptops.

Executable code size is 427 kilobytes. The Help files are a further 292 kilobytes. The Help files, written using the HDK authoring package, utilise a dynamic linked library *hdkcnts.dll* to give enhanced tables of contents.

A demonstration version of EaSim is available by e-mail from the author.

All files created by EaSim, the assembler source file, the assembler list file, the assembler symbol table, the object code, 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, there is a separate *Instructor's Version* of the program for use in lectures. This uses a larger font and rearranged screen layouts to facilitate reading from video projected images.

FURTHER DEVELOPMENT

EaSim is in the process of being updated to cover the 68HC11 microcontroller running in the MIT developed Handyboard development environment. This product is in beta test mode. It will be used in teaching the Microprocessors subject in 1999.

This version has simulated analogue and digital inputs as well as digital outputs.

CONCLUSION

EaSim is an integrated editor/assembler simulator package that has proved very useful for introducing students to microprocessors and microprocessor programming. Further development to encompass the 68HC11 will extend the usefulness to more complex tasks.

Acknowledgements—The author would like to acknowledge the assistance provided by his colleagues in the development of the EaSim package. Charles Hacker in particular developed the keyboard entry routines, the download to the D3 hardware and assisted with general debugging. Nicholas Edwards provided invaluable assistance with Delphi programming.

REFERENCES

1. M6800 Microcomputer: System Design Data, Motorola Inc., 1976.
2. http://www.vmdesign.com/universal simulator
3. http://www.samphire.demon.co.uk/8086

APPENDIX

Student evaluation summary

A formal evaluation of student perceptions of EaSim was undertaken in the year of introduction. During this year many versions were released as bugs introduced by the switch to the Windows environment were found and fixed. A number of pre-existing bugs in terms of the simulation of some 6802 opcodes were also found and fixed.

Students were asked to indicate there agreement with a number of statements on a modified Lickert 5 point scale ranging from 5 (strong agreement) to 1 (strong disagreement)

Question	SA	S	N	D	DS	Av
EaSim enhanced the learning environment	12	26	4			4.2
Simulator makes it easier to understand μP workings	13	20	7	2		4.1
EaSim makes programming assignments easier	16	18	5		2	4.1

The students also indicated that 97% of the class had access to their own computer off campus. 100% of the class said they used the simulator program off campus.

When asked to state the best feature of EaSim, the universal answer was the visibility of the memory and register contents

When also asked to state the worst feature of EaSim, two main complaints were raised. These were the lack of examples in the Help files and the poor simulation of the LED display in terms of speed.

David Edwards holds a B.Sc. in Physics from University of New South Wales, Australia and an M.Sc. in Meteorology from the University of Reading, UK. His interests lie in flexible delivery of engineering subjects and the use of multimedia to support the teaching and learning environment.

He was the foundation dean of the Faculty of Engineering and Applied Science on the Gold Coast campus of Griffith University. He is presently a senior lecturer in the School of Engineering and the Shared resource Co-ordinator for the Faculty of Engineering.