計算機アーキテクチャ　100－6
（1993．6．11）

工業高専におけるマイクロプログラミング教育

今井 慈郎[1]　本田 道夫[1]　小島 洋一[2]　富田 眞治[3]
宮武 明義[4]　田嶋 眞一[4]　國井 洋大臣[4]

[1]香川大学 経済学部　[2]日本 AMD (株)
[3]京都大学 工学部　[4]詫間電波工業高等専門学校

マイクロプログラミングを実践的に教育するための教育環境において核として使用できる教育システムを開発した。このマイクロプログラミング実験システムは、ボード型マイクロプログラム制御計算機、制御用ワンボート・マイコン、およびパソコンから構成されている。実験支援のためのユーティリティとして、アセンブラとデバッガを装備し、パソコンから会話的に操作できる環境となっている。

工学実験での利用形態や学生からのレポートなどを基に実験システムを含む教育環境全体の簡単な評価を行った。その結果、工業高等専門学校での計算機システム教育を補完し、実践的な情報処理教育を行う上で効果があったと思われる。また、企業等における新人教育においても利用可能であると考えられる。

An Education System of Microprogramming
– An Example for National College of Technology –

Yoshiro IMAI[1]　Michio HONDA[1]　Yoichi KOJIMA[2]　Shinji TOMITA[3]
Akiyoshi MIYATAKE[4]　Shin-ichi TASHIMA[4]　Hiro-omi KUNII[4]

[1]Kagawa University, Takamatsu-shi, Kagawa, 760 Japan
[2]Advanced Micro Devices Japan ltd., Shinjuku-ku, Tokyo, 160 Japan
[3]Kyoto University, Sakyo-ku, Kyoto, 606 Japan
[4]Takuma National College of Technology, Mitoyo-gun, Kagawa, 769-11 Japan

We propose an education system to facilitate practical instruction of microprogramming. Our education system consists of three major components, namely a microprogrammable CPU (implemented on a VME print board), a one-board type microcomputer to control and manage the whole system, and a personal computer for man-machine interaction. In order to utilize an environment of student experiment, the system prepares microprogram assembler/linker and microprogram debugger. They can be used through personal computer with conversational mode.

Brief evaluation has been carried out about our system, by means of analyzing experiment effects and reports from students. Our system can play a supplemental role for students actually to understand computer architecture of register-transfer level. And it will be one of the flexible and powerful tools which can support to educate practically some kinds of subjects, such as 'computer system', 'switching theory', or 'systems programming'.

1 Introduction

Information processing education is , nowadays, necessary in not only engineering fields but other science fields including social science. Many schools, which have departments related with information engineering, want to accomplish that education from hardware to software using more advanced and systematic environments. But not a few of those schools have suffered from lack of education tools/systems to instruct computer system practically.

Those schools are looking forward to an efficient education system for computer architecture and system softwares. Such a system can provide comprehensible methods and an effective environment to teach beginners practical examples about a computer system. We have focused on microprogramming[1], because it is one of the basic technologies of computer organization, and it is situated between hardware and software. Understanding register-transfer level mechanism of computer system can bridge a gap of lecture between 'switching theory' and 'computer system'. Unfortunately, however, microprogramming used to be introduced almost always absolutely in lectures.

We have developed a new education system to instruct microprogramming in order to teach a wide scope of computer technologies from hardware level to software one. The education system is designed to be used not only as a microprogramming environment but as a reference model to construct hardware and system software. On the other hand, our system is both a tool, with which description and operation of microprograms can be performed, and a teaching material of which students follow an example to construct another version in the experiment.

The education system is organized with a microprogrammable CPU, one board microcomputer, and a personal computer. Such a organization may be almost sufficient for microprogramming education system to explain the details of computer system and system softwares practically.

This paper describes microprogramming education system at first, illustrates details of our system construction secondarily, and presents brief evaluation of the education system finally.

2 Education System

Design concept of our system is introduced here, which consists of the following keywords;

Not Simulation but Actual Hardware : A simulation-based method is very efficient to build up such a system. From the viewpoint of educating practical microprogramming, however, they are very important procedures to design microprogrammable CPU, implement that hardware using CAD, and create related software utilities. So we think that microprogramming education system must be actualized with not simulation but real hardware.

Personal Use : In multiplexed system, availability is better than single-user system. Blocked is system expandability, however, such as interfacing several peripheral devices or tuning total system for some applications. The system should be constructed for personal use so that it can permit some requests to reorganize even its architecture.

System Customization : Softwares have useful characteristics to exchange their shapes by means of easier efforts. In the similar manner, microprogramming has tailorability for applications. So whole our system must be completely customizable. Generally speaking, ROM-based systems intend to be solid for any requests. We employ system configuration to ensure that applications can be down-loaded and executed like other native functions. A

new function can be easily appended to our system. Reorganization, namely system customization, is a characteristic feature of our system.

There may be two types of experiments as an application of our system into actual microprogramming education.

One type is writing several kinds of microprograms between multiplication/division microroutines and emulator for virtual machine, for example, "COMET" which is a well-known conceptual computer designed for information processing education. And some mechanical-electronic systems can be constructed with our system, such as micro-mouse control or hobby car's radio control.

The other is employing the system itself as reference model for construction of simple computer system including system softwares. Students are allowed to implement another microprogram assembler/linker or microprogram debugger, referring to the original softwares written in C. If cheap CAD services and low-cost print board development are available, our microprogrammable CPU will be a reference model for students to actualize a new hardware during experiment.

In general, few students of colleges seize opportunities to write microprograms and operate microprogrammable computers. So the student, given chances to use a microprogrammable computer, make an advantage of it to understand register transfer level behavior of computer actually.

3 System Configuration

This section explains system overview, structure of microprogrammable CPU, microprogram description language, its language processor, and microprogram debugger.

3.1 Overview

Our microprogramming education system is built up of the three parts: a microprogrammable CPU,a microcomputer for system management, and a personal computer.

An education-oriented microprogrammable CPU has been developed by Advanced Micro Devices Japan Ltd. and Takuma National College. The CPU is designed as a microprogrammable computing engine for microprogramming education system[2]. This CPU is called M1. It does not has main memory, input-output devices, or any equipments for system maintenance. It is implemented into a double-height VME-bus size print board. Some conventional components, therefore, can be utilized to build up a total computer system with the microprogrammable CPU.

An M68000-based microcomputer board, we call Supervisor Processor, has a VME-bus connection with externally-accessible memory. It is connected with M1,and provides its shared memory as the main memory of both that CPU and itself. It may also work as an input-output controller of the total computer system.

Supervisor Processor plays the two following important roles: One of them is a manager of our microprogrammable CPU, which provides some facilities such as microcode loading, microprogram-level debugging, main memory managing, and so on. The other is an input-output processor for M1, which offers flexible interface operations such as serial communication with users. Even parallel data communication may be available with other digital system. The former role is played in the passive mode of our microprogrammable CPU, where it can be controlled step-by-step by means of Supervisor Processor. The later one is done in the active mode of our CPU, where it can work autonomously. A microprogram-level debugger, called M1DEB, has been im-

plemented on Supervisor Processor. M1DEB supplies an interactive debugging environment such as internal registers dumping or exchanging, step-by-step tracing, and microcode loading.

With VME-based Bus Interconnection, M1and Supervisor Processor are semi-tightly connected, where M1 can gain access to the main memory. It is also available to add a specific VME board and expand system configuration for practical applications.

A personal computer, for example IBM-PC or NEC-PC, is loosely connected with Supervisor Processor through an RS-232C serial communication channel. The personal computer facilitates essential functions of our system such as terminal emulation, file management, and microcode generation.If you need only the function of terminal emulation, a dumb terminal could be connected with Supervisor Processor. In such a dumb terminal, however, the other functions cannot be equipped. We decide to use a personal computer to manage source, object, and executable files about microprograms efficiently. In the personal computer are also available many utilities such as screen editor, cross assembler/compiler/linker for Supervisor Processor, native compiler/linker, and so on.

3.2 Microprogrammable CPU

The characteristics of our microprogram CPU are summarized as follows: semi-horizontal type of 40-bit length microinstruction word, 16-bit parallel ALU operations, 4K words of control storage(up to 64K words max), microprogram-level subroutine nesting(stacking), and functional register read-write mechanism for microprogram-level debugging. It can control sequencer, ALU, interruption, and memory operations simultaneously.

The CPU M1 has 9 special registers. We call them Functional Registers as generics, namely two memory address registers(MARU, MARL), two memory data registers(MDRR, MDRW), 40-bit length microinstruction register(μIR), microstatus register(μSR), transfer mode register(TMR), macro-instruction register(MIR), and jump-vector table register(VTR). These registers can work asynchronously in the normal operation mode. But they are serially connected so that all their contents can be rotated from end to end. Such an operation mode is actualized by a special hardware structure, we call Serial Link Mechanism of Functional Registers.

In order to gain access to those functional registers, we prepare a particular procedure to utilize two memory data registers. In such a procedure, the memory data register for writing(MDRW) is used as an entry point to set data to be written into. The memory data register for reading(MDRR) is considered to be used as a monitoring point to dump data to be read out. If you want to write new data at some functional register, you can rotate the content of such a register at the position of MDRW with this mechanism, set that new data into MDRW, and rotate again until all the contents of registers come back at the original position. Consequently the new data has been stored into the functional register you want. While your request is to read a content of some register, it will be realized by a similar manner described above using Serial Link Mechanism of Functional Registers.

It can be thought that conceptually Read-window and Write-window are equipped to read and write content of functional registers respectively. These windows may be moved at any position in the chain of serially-linked functional registers. At the position of such windows, the content of the according register can be dumped or exchanged.

3.3 Description Language and Its Processor

It was necessary to design a microprogram description language for our microprogrammable CPU. Such languages used to be complicated so they are not readable for beginners. But readability are very important for beginners to understand microprogramming. We consider structured description of microprograms to be readable for anyone. So a microprogram description language has been designed easy to describe microprograms with several control structures such as if-else-statement, for-end loop, while-loop, and unconditional loop.

It is one of the most characteristic specifications for language design to employ module-oriented structure about describing microprograms. External references are available in every module, which makes sure that the specific labels are to be written as a destination place for many branch operations after those labels are listed by means of "import" declaration. It is assumed that such labels are externally defined elsewhere.

It is summarized that our microprogram description language has the following characteristic specifications:

- It is well-designed for users to write module-structured microprograms easily. It can express several control structures like high-level programming languages.
- Some microoperations can be represented in one statement, whose according fields are separated by delimiter(,). Free-format programming style is available because every statement is terminated by delimiter(;). It accepts strings for comments as the same as C language does, which are enclosed with (/*) and (*/).
- All the module blocks must be led by the reserved words, namely "module" or "located", and "module"-leading blocks are translated into position-independent microcodes, while "locate"-leading ones are into position-specific microcodes.
- Both kinds of the above blocks may declared external references, which are represented with label-parameter list followed by the reserved word "import".

A language processor of microprogram has been implemented in C language. Because this language is widely available from MS-DOS to Unix. It is very powerful to manipulate low-level file accesses, too. But it is the very reason that we can utilize generators for parser and scanner called Bison and Flex without special efforts. Our language processor consists of microprogram assembler and linker. Both of them include two kinds of code segments which are generated by Bison and Flex.

The first version of our microprogram assemblers is called M1ASM, which reads the source files of microprograms, translates them into machine code strings, and generates a final microcoded file to be loaded directly. It is necessary for us to use a simple-structured microprogram assembler (even though it has poor functions), because we needed to assemble some sample programs into microcodes and test a hardware of our developing microprogrammable engine with these codes. So this version is a prototype of assembler. But M1ASM performs completely, and it is still suitable enough to develop small sizes of microprograms handily. It is mainly made up of scanner, parser, table keeper, and code generator. Except of separate assembling facility, M1ASM can satisfy the specifications mentioned above.

In order to develop large scales of microprograms, it is necessary to employ separate assembling facility. So we have upgraded M1ASM into a new microprogram assembler called M1ASM2. It is one of the most different

points between the first version and the second one that the former generates the absolute microcodes, while the latter outputs the relocatable intermediate microcodes. M1ASM2 assembles many source files of microprograms into the according intermediate object files. Such files are assumed to be processed by microprogram linker. Same as the previous version, a new one also consists of scanner, parser, table keeper, and intermediate-code generator.

The microprogram linker is designed to bind the intermediate microcoded object files into one executable binary file. A style of the binary file is called "M1-format" and is like S-record format of Motorola. This style is suitable for such binary files to be transferred and stored into WCS of the microprogrammable CPU through RS-232C channel of the supervisor processor.

3.4 Microprogram debugger

A microprogram debugger is developed to load microcodes into control storage of M1, to check whether loaded microcodes correctly work or not, and to maintain the whole computer system. It is implemented to provide an interactive environment of microprogramming education system. This debugger called M1DEB is written in C language, translated into ROM-based machine codes, and available on M68000-based microcomputers.

M1DEB provides several facilities, with which users can utilize the microprogramming system efficiently. Those facilities includes microcode loading into control storage, stepwise or continuous microcode tracing, and dumping or exchanging contents of many registers of M1.

The hardware of M1 can be manipulated by means of setting Control Register with several control patterns. According to those patterns, all the contents of functional registers are rotated simultaneously, the hardware works step-by-step, or M1 itself can operate autonomously. Status Register is prepared for monitoring M1's internal condition. So the debugger on Supervisor Processor can examine M1's current status from the status register, and handle the detail of M1's hardware by means of activating Control Register.

In debugging mode, it is very important to get precise information what instruction is executed or what next instruction will be done. Even short-length machine instructions might be not easily understood from only their bit patterns. With long-instruction words, therefore, a disassembler is necessary, which deciphers machine-readable bit patterns into human-readable mnemonic sequence. As you know, microcodes must be usually stored in control storage of CPU. Such microcodes are not accessible directly so that we cannot dump nor exchange current content of the microinstruction register of M1 without Serial Link Mechanism of Functional Registers.

Disassembler for WCS-loaded microcodes is one of subroutines included in our debugger. This routine does not usually stand alone but almost always cooperate with a trace function. The disassembler scans 40-bit length bit pattern, divides it into some fields, which are for sequence control, ALU control, source/destination control, and miscellaneous control, and then translates each field into an according assembly symbol.

M1DEB also provides a kind of function for microcode loading. But a function for microcode generation do not belong to the debugger. Another part of our education system must be due to such a function. First of all, users of the education system describe programs with the microprogram description language for M1. On a personal computer, they are stored as files of MS-DOS. With M1ASM, the programs are translated into executable binaries, one of whose characteristics is a man-readable ASCII string. Second, the binaries

can be transferred from personal computer to Supervisor Processor through an RS-232C communication channel. They are stored in a suitable area, called buffer for microcodes, of the main memory. Finally, M1DEB reads out the pair of address and its data(microcode) from buffer, sets them VTR and μIR respectively, and writes stepwisely the content of μIR into the location of WCS specified by VTR using Serial Link Mechanism.

A command processor of M1DEB is equipped to improve user interface, which is called M1-shell. It can accept dump or exchange request for registers, and let the according command routine display or replace the content of those registers. M1-shell can also accept a trace request for the specified sequence of microcodes. For trace request, M1-shell invokes the according trace function to perform stepwise execution, display the contents from Functional Registers to many internal registers of ALU, disassemble the content of μIR into microassembly mnemonics.

Customization is one of the most attractive characteristics of our debugger. If users wants to develop their own debugger, they can describe another version of microprogram debugger. Additionally it is able to be downloaded onto Supervisor Processor, and executed as same as the original debugger is invoked. This characteristic is educative because users can design and implement their own microprogram debuggers as microprogramming education practices.

4 Courseware and Evaluation

This section introduces our courseware of microprogramming education included in lectures and student experiments.

At first, subjects on 'switching theory' and 'computer system' are lectured to a whole class of students of Information Engineering Department of Takuma College. We teach them various logical circuits and register transfer level behavior of computer, referring our microprogrammable CPU, M1, as practical examples. Next year, the students can choose four themes from hardware to software among seven ones prepared for the student experiment. From five to seven students are assigned to one theme. It takes seven weeks for students to be explained about the principle and objective for each theme, exercise everything, resolve their problems, and report own results during that experiment.

The schedule of our microprogramming experiment are summarized as follows:

1. explanation about microprogramming education system (1 week)
2. exercises of microprogram coding (2 weeks) including how to use softwares and how to operate education system
3. applications construction (4 weeks)
   choice among the below sub themes
   - coding a virtual machine emulator
   - reorganization of system softwares
   - controlling a micro-mouse/radio-controlled car
   - (design another version of microprogrammable CPU)

The first and second sub themes used to be selected by the students who make it their concerns to develop system softwares. The third sub themes are always the most favorite themes of all. Students will want to make any motion system rather than abstractive or static system. But the fourth sub theme, we want to realize soon, is difficult to carried out in our experiment. Once a student tried to design a new microprogrammable CPU. Although he get a high score of 'computer sys-

tem' and he was willing to inspect many documents about computer architecture, he was forced into only desk-top design of CPU because of lack for support equipment such as CAD system and Logic Simulator. That is why we can not realize the fourth sub theme.

From a few reports, we analyze all the students, who choose this theme, can visualize the details of microprogramming and understand the register transfer level behavior of computer practically. Simple microprograms can be written by every student during the experiment. Motion system construction as well as development of graphic applications is one of the high evaluated themes from students. There is not quantitative analysis, we think microprogramming education will be easily comprehensible for students to utilize our education system.

Currently, we have a serious problem to be solved. That problem is that there are two sets of our education systems and only one can work fully correctly. Personal Use is one of the concepts of our system, but some students have to use the system in multiplexed mode. Advanced education systems will be accomplished for each student, if any sponsor allows us to design and implement a new microprogrammable CPU,

5 Conclusion

This paper describes a education system of microprogramming. The system is organized simple-structured microprogrammable CPU, M68000-based one board microcomputer for system management, and a personal computer for multi-purpose applications such as terminal emulation, file management, microcode generation, software development and so on.

It also presents the system configuration, including CPU structure, microprogram description language, its language processor, and ROM-based but customizable microprogram debugger. These facilities are not only designed as components of education system but also developed as teaching materials to fill the gap of some lectures about computer system. They can allow students to understand the practical examples about computer hardware and system software.

Brief evaluation indicates that this education system is effective for users to comprehend the actual mechanism of microprogramming. Especially, it is useful to construct any motion system in the experiment. We think that our system will be useful to accomplish a practical information processing education not only for students of technical college but also for newfaces of company.

Acknowledgements

The authors express special thanks to Hidetomo Shibamura and Toshio Fukui for their contributions to develop microprogram assembler/linker and microprogram debugger respectively. They thank to Haruo Abeof Advanced Micro Devices Japan ltd. for his constructive support to actualize a prototype of microprogrammable CPU. They are grateful to our students of Takuma College for their comments and reports during the experiments.

This work was supported in part by grant 6375365 in aid for scientific research from the Ministry of Education, Science and Culture. It was also sponsored in part by Advanced Micro Devices Japan Ltd.

References

[1] H. Hagiwara, "Microprogramming", Sangyoutosho, 1977 (in Japanese).

[2] Y. Imai et al., "A Practical Education System for Microprogramming", InfoJapan '90, pp. 357 - 364, 1990.