사용자가 마이크로 프로그램을 할 수 있는 컴퓨터 설계

논 문
26~1~2

Design of A User Microprogrammable Computer

조 정 완*·우 남 성**
(Jung Wan Cho, Nam Sung Woo)

Abstrakt

It has been expected that the 4th generation computers will be characterized for their problem adaptability. There are few techniques of implementing such a characteristic. One of the techniques that one have considered in this paper the user microprogrammable computer architecture. There are two different computer architectures that support user microprogramming. One uses the writeable control storage and another uses the main memory.

The concept of utilizing writeable control storage for microprogramming was developed in 1950's and since then the most of the user microprogrammable computers produced belong to such category. The concept of utilizing the main memory for user microprogramming was first introduced by Thomas in 1973. This architecture has a strong advantage in the aspect of the system cost.

In this paper, we have developed a user microprogrammable computer. The computer utilizes the main memory for user microprograms. It employs a 32 bit micro-instruction word in the form of the little encoded. The performance of the developed machine will be evaluated in the hardware cost, programming easiness. and the running time.

1. Introduction

During the last decade, significant progress in the computer hardware development technolgy has been wittnessed. Among the various areas of the technology, the architectural enhancement resulted in more flexible and more powerful computers. Such architectural improvements was possible due to indirectly the improvement of the semiconductor processing techniques and due directly to the utilization of the large scale integration (LSI) memory and random logic, and to the adoption of microgram.

Concept of microprogram [1,2] was first developed in the early 1950's However its use in the control sections of the commercially produced computers was delayed until the mid 1960's. This is because the instruction sets of the most of the computers manufactured before the mid 1960's were not required to be powerful and therefore the cost of microprogrammed control section is greater than that of the control section implemented by the random logic [3].

Microprogrammed computers can be classified into two classes. One is the group of host machines which are microprogrammed to the equivalent target machines and another is the host machines which can be microprogrammed to implement different target machines. The term equivalent target machine, in the above, is used in the sense of a group of machines which have the processing elements that look alike to the aoolication programmer. One of the typical example of the former class of machines is the various models of IBM System/360 [4,5]. The computers in the later class are called user

* 正會員：韓國科學院 敎授(工博)
** 〃 〃 在學

microprogrammable or dynamically microprogrammable computers. Some of the examples of such microprogrammable computers are QM-1 [6], META 4 [7], MLP 900 [8], and EMMY [9].

In this paper we propose a user microprogrammable computer, CRIMM (Control Resides In Memory), that is being designed and implemented in Korea Advanced Institute of Science.

2. User Microprogrammable Computer Architecture

User microprogrammable computer architecture can be characterized as an architecture that allows the user to customize the computer for the best possible way for his application. Therefore it must provide the user the user the facilities of defining his target machine architecture including the instruction set. When a host machine provide complete freedom to every user such facilities to emulate every possible target machine it is called an universal host [9], Design of a virtually universal host machine is a very difficult task because the performance of the target machine being emulated is decided by the scope and width of the data paths and the execution units. For instance, if a user wants to emulate a 32-bit target machine in a host machine that has an 8-bit adder, the performance of the emulated target machine will be poor when it is compared to an ordinary 32-bit machine. Therefore in designing a host machine a compromise muss be made between the hardware cost and the performances of the emulated target machines.

One of the factors to be considered in designing a user microproprogrammable host machine is whether the host machine shall have instruction set or not. When the host machine does not have any machine instructions, each user must generate his own instruction set via microprogramming. In the macbines which have aninstruction set a class of applications are defined for the host so that the intersection of the requirements of every application is microprogrammed in advance. Therefore in such systems, duplicate efforts can be avoided. The instructions in the intersection are called the base machine instruction set, Base machine instruction set, therefore, must include such instructions as load, store, simple arithmetic, branch instructions. What each user has to do is microprogram his own special functions to compensate the base machine instruction set. In this paper, we shall only consider the later type of architecture further.

It is apparent that in order to support user microprograms, host machine must have some form of user microprogram storage. Such storage element must also provide dynamic read/write as the conventional main memory does. Depending on the position of the user microprogram storage, user microprogrammable organizations can be classified into two different classes. Ooe is the organization that employs the writeble control storage(CS), and anothor is lhat employs the main memory(MM) for user microprogram. The later class of machines can further be grouped into three different types [10].

Type I machine organization is shown in Fig. 1. In this organizalion, CS contains the base machine control informations and MM contains the user's program as well as the user's microprogram. Therfore, CS needs not to be dynamically writeble. Type II machine organization is shown in Fig. 2. In this type of machines, CS and MM are both in the same address space therefore, they use the same addressing logic, In a typical Type II machine organization, memory is organized in such a way that the lower address spaces are assigned to the CS and the higher address spaces are assigned to the user microprogram and the user program. In this organization, since MM and CS are in the same dynamically writeable memory space, base machine protection is one of the important problem. Type III machine organizationsis shown in Fig. 3. In this type of machines, MM and CS are in the separate dynamically wrileable storage. In a typical Type III machine organization, the CS cycle time is faster than

Fig. 1. Type I machine organization (memory)

Fig. 2. Type II machine or ganization (memory)

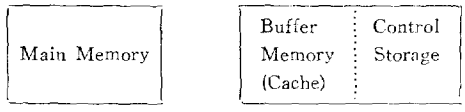

Fig. 3. Type III machine organization (memory)

the MM so that the user microprogram in the MM is loaded into the cache, which is the same address space as CS, and executed out of cache. MM contains the user's program and user's microprogram. When the desired microprogram is in the cache, the performance of the Type III machine organization is identical to the Type II machine organization. However, the performance of the Type III machine organization depends on the buffering algorithm and the actual iob being executed.

3. User Microprogrammable Computer-CRIMM

In this paper, a user microprogrammable computer, CRIMM, that employs the MM for user's microprogram is designed and constructed. CRIMM is a 16-bit general purpose minicomputer designed with small scale integrated(SSI) and medium scale integrated(MSI) circuits. The central processing unit(CPU) is constructed around three 16-bit data buses-input bus 1, input bus 2, and output bus-, a 16-bit parallel adder/shifter, and 12 16-bit general purpose register file that includes 4 accumulators, 3 special purpose registers, and 3 input/output buffers. The organization of CRIMM is shown in Fig. 4. As can be seen in Fig. 4, all the data transfers are made through the arithmetic and logical(ALU) no motter which register the data originates. In other words, data transfer must occur in input bus 1-ALU-output bus or, alternatively, input bus 2-ALU-output bus.

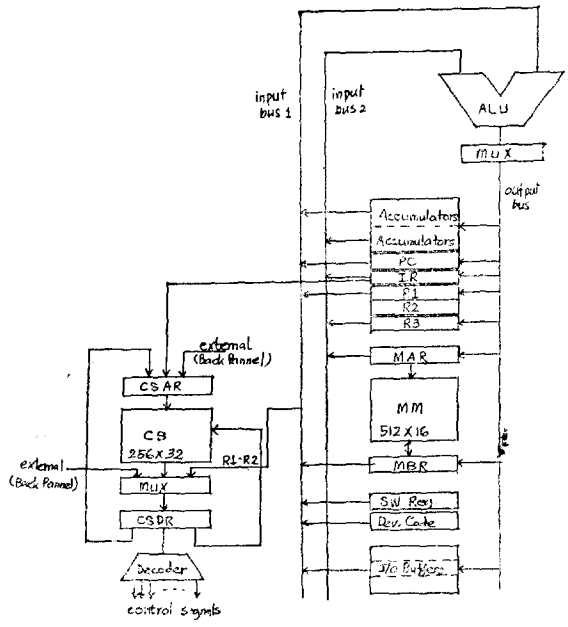

Fig. 4. Organization of CRIMM.

Special Purpose Registers

There are three special purpose registers -R1, R2, and R3 -in the genral purpose register file. These registers are transparent to the ordinary machine language programmers. They exist only for user micropramming support. Concatenation of R1 and R2 contains the user's micro-instruction fetched from user's microprogram that resides in the MM. Therfore, in order to execute the user's microprogram (micro-instruction), the control storage data regiter(CSDR) must be loaded with the information contained in the concatenation of R1 and R2. R3 is the validity counter register that contains the number of user defined instructions, i.e. the number of user's microroutines. The purpose of R3, therefore, is to checi the vaidity of user's operation code used in his program.

Base Machine Instruction Set

Base machine instruction is 1-address type and the instruction word length 16-bit. The control sepuence for all the operation codes in the base

machine instruction set are microprogrammed and slored in the CS.

A. Memory Reference Instruction

Various memory addressing methods such as indexing, direct addressing, reference instructions. Especially, unlimited number of levels of levels of indirect addressing is permitted. The memory reference instruction group consists of load, store, simple arithmetic and logical operations to the memory contents and branch operations. The memory erference instruction format is shown is shown in Fig. 5.

B. Arithmelic and Logical Instruction

The arithmetic and logical instructions are register type instructions. The instruction specifies the source and destination accumulators. There are two optional fields in the instruction word. Skip field specifies the test and skip conditions for the ALU operation result. The test can be specified to the result of the ALU operation or to carry. Load/No Load field specifies whether the result of the ALU operation is to be loaded or not to the destination register. Instructions in this group consists of simple ALU operations such as the arithmetic, logical, rotating and swap byte operations. The format of the arithmetic and logical instructions is shown in Fig. 6.

C. Immediate Addressing Instruction

Instructions in this group consist of load and add. Load instructions are to be used to load the accumulator with the immediate data specified in the operand field. Add instructions are to be used to edd the immediate data specified in the operand field to the accumulator leaving the result in the accumumulator. The format of the immediate addressing instructions is shown in Fig. 7.

D. Input/Output Instruction

Input/output (I/O) instructions are divided into two categories. One is a set of strict I/O instructions and another is a set of special CPU instructions. I/O instructions consist of the control of asynchrounous I/O control flip flops, namely busy and done flip flops, and their test as well as the programmed data transfer type of I/O data move and I/O skip. Special CPU instructions include the interrupt associated instructions, halt, control pannel switch read. The format of I/O instructions is shown in Fig. 8.

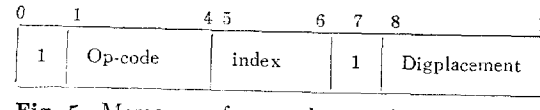

Fig. 5. Memory reference instruction format.

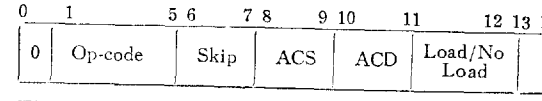

Fig. 6. Arithmetic and logical instruction format.

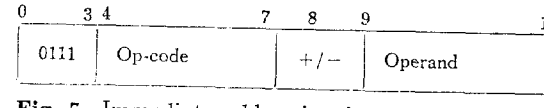

Fig. 7. Immediate addressing instruction format.

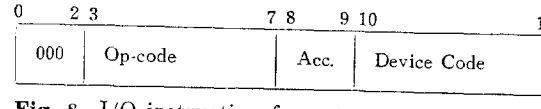

Fig. 8. I/O instruction format.

Micro-Instruction

CRIMM employs a poly-phase, encoded horizontal type micro-instruction. The length of the micro-instruction is 32-bit. Therfore, user's micro-instructions must be stored in a pair of adiacent MM locations. The format of micro-instruction is shown in Fig. 9. As shown in Fig. 9. input bus 1, input bus 2 and output bus fields are encoded so that when a pair of input and output

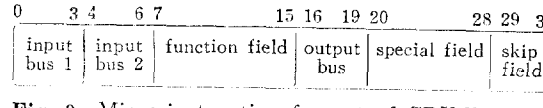

Fig. 9. Micro-instruction format of CRIMM.

buses are activated a register to register data path is opened for a registef to register load or for an ALU operation when one of the ALU function is also activated in the function field. The function field is a horizontal format so that one of nine direct control signal emanates to one of the control points of the ALU. It may be possible to reduce the micro-instruction word length to 27 bits by encoding the function field using only 4 bits. However, CRIMM cannot

afford extra time penalty to decode the function field to generate the ALU control signals. The special field generates the control signals for MM read/write, I/O, and status checks.

Since CRIMM used high speed bipolar semiconductor memory for MM, read operand from MM, execution and storage of the result to the MM can be done in one micro-instruction cycle. Thus, operand read must be issued early in the micro-instruction cycle, i.e. we cannot afford delay for ecoding to generate the MM read sigdal. In case of I/O, since I/O operation is inherently slow, device code musn be sent out to the device controller as soon as possible. Therfore separate bits, one bit each, are assigned to the MM read and to discriminate the I/O instructions from the rest of the instructinos. The purpose of the skip field is micro-instruction sequencing. It generates necessary control signals for the conditional as well as unconditional micro-instruction cranching. Conditional branch operations deal with the handling of the indirect addressing and the non-base machine operation code. Unconditional branch operations deal with the transfer to the specified addresses such as the fetch routine address.

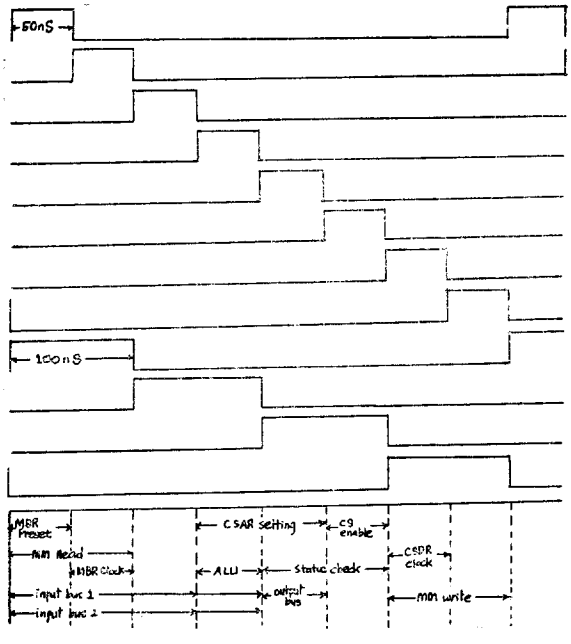

Fig. 10. Timing diagram of CRIMM

Timing

Master clock frequency is 20 MHz and microdrogram controlutilizes 12 clock phases. The timing diagram is shown in Fig. 10.

Board Organization

Basic CRIMM consists of 11 66-pin boards and a front and back control pannels. Fig. 11 shows the board organization. The back control pannel along with the back pannel control board are needed to load the micro-routines for the base machine operation codes into the CS.

Front Control Pannel

Front Pannel Control
Control Storage & Multiplexor
Skip Field, IR, CSDR, CSAR
3 Buses Fields, Function, Special Fields
Timing Control
Clock
Main Memory
Arithmetic and Logical Unit
Teletype Interface
User Microprogramming Board
Back Pannel Control

Back Control Pannel

Fig. 11. Board organization of CRIMM.

4. Conclusion

A general purpose minicomputer that provides the facility for dynamic uter microprogramming in the MM has been designed and constructed. Such a system facilitates additional flexibility and power to computers whose control section is either hardwired or microgrammed.

The proposed computer must further be improved in both architectectural and organizational levels. There are several specific areas to be investigaed. First, the width of data paths must be studied in order to facilitate both scientific and business applications. Second, base machine instruction set must be optimized. Third, addressing scheme, especially variable length operand addressing scheme, must be studied. Fouth, advanced I/O capabilities must be integrated. Fifth, micro-instruction format must be studiede furthee to incorporate more sophistied micro-instruction sequencing schemes. Lastly, present wired boards must be transformed to printed circuit boards in

order to improve the performance and reliability of the computer.

References

[1] J.W. Cho, "Microprogramming for Control Units," KIEE Review, Vol. 3, No. 1, pp. 12~17.

[2] Y.J. sSong, "Computer Design and Microprogramming," KIEE Review, Vol. 3, No. 1, pp. 18~24.

[3] S. Husson, Microprogramming Principles and Practices, Englewood Cliffs, M.J., Prentice-Hall, 1970.

[4] W.Y. Stevens, "The Structure of System/360: Part 1-System Implementations," IBM Systems Joumal, Vol. 3, No. 2, 1964, pp. 136~142.

[5] S.G. Tucker, "Microprogram Control for System/360," IBM Systems Journal, Vol. 6, No. 4, 1967, pp. 222~241.

[6] Nanodata Coration, QMI Hardware Level User's Manual, 1972.

[7] Digital Scientific Corporation, META 4 Computer System Microprogramming Reference Manual, June, 1972.

[8] H.W. Lawson, Jr. and B.K. Smith, "Functional Characticnal Characteristics of A Multilingual Processor," IEEE Trans. on Computers, Vol. C-20, No. 7, July, 1971, pp. 732~742.

[9] M.J. Flynn, C. Neuhauser and R.M. Mc Clure, "EMMY-An Emulation System for User Microprogramming", Proc. of 1975 NCC, Vol. 44, 1975, pp. 85~89.

[10] R.T. Thomas, "Main Memory for User Microprogram Residence An Analysis," Proc. of 6th Annual Workshop on Microprogramming, Sept. 1973, pp. 13~20.