Title	A general-purpose experimental computer system characterized by its architecture changeability
Sub Title	
Author	Okada, Kenichi(Kitagawa, Misao) 北川, 節
Publisher	慶応義塾大学工学部
Publication year	1980
Jtitle	Keio engineering reports Vol.33, No.5 (1980. 5) ,p.49- 66
JaLC DOI	
Abstract	This system is composed of both a microprocessor whose control memory is completely opened to the user, and a minicomputer which stores the supporting software used for developing the microprogram for the microprocessor and aims at simplifying its development, testing, and updating. The purpose is accomplished conversationally on the disk base and the system can be utilized as an adaptive computer system to every objective of the users.
Notes	
Genre	Departmental Bulletin Paper
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO50001004-00330005-0049

KEIO ENGINEERING REPORTS
VOL. 33, NO. 5, pp. 49-66, 1980

A GENERAL-PURPOSE EXPERIMENTAL COMPUTER SYSTEM CHARACTERIZED BY ITS ARCHITECTURE CHANGEABILITY

KEN'ICHI OKADA AND MISAO KITAGAWA

Dept. of Instrumentation Engineering, Keio University, Yokohama 223, Japan

(Received October 4, 1979)

ABSTRACT

This system is composed of both a microprocessor whose control memory is completely opened to the user, and a minicomputer which stores the supporting software used for developing the microprogram for the microprocessor and aims at simplifying its development, testing, and updating. The purpose is accomplished conversationally on the disk base and the system can be utilized as an adaptive computer system to every objective of the users.

1. Introduction

The microprogramming technique has such an essential feature that it gives flexibility to the hardware construction of a processing unit, and thus is widely utilized in various fields such as emulation, high-level language processing, special-purpose computer, and so forth.[4] On the other hand, the LSI technology has recently been made a remarkable progress, and consequently various kinds of microprocessors, low-price, high-speed ICs, and large capacity IC memories have now been provided. Furthermore, since the microprogram control system is extensively introduced to the mini- and micro-computers, they have taken great strides in their progress of application.[1]

In the microprogram control computers that are on the market, there are two types; one is such that its control memory is opened to the users, and the other is not to them. When trying to study the utility and capability of the microprogram system, a computer as an objective should have such characteristics that its control memory is opened to the user, and is feasible to alter the contents. In order to alter or modify the contents of the control memory, two ways of static and dynamic alterations are considered; the former is the case where after temporarily interrupting and halting the computer operation, they are altered manually, and in the latter they can be changed by program during the computer operation. Naturally, this is richer in the flexibility and adaptability than that.[8] Though there are several techniques of constructing the con-

trol memory, the present main current of it is such that it (1) should completely be independent of the main memory, (2) adopts a high-speed IC memory, and (3) comprises a combination of non-volatile ROM and volatile RAM and thus provides us with the flexibility, adaptability, and ease of usage.

In this type of computers, a dynamic microprogramming is realized in such a way that an elementary instruction set is stored in the unalterable ROM beforehand after having been programmed, and by using those instructions (normally those for input/output), the microprograms are written into the RAM.

Some supporting software of considerable amount of functions is required in order to develop the various kinds of such microprograms to be written into the RAM of the control memory which can dynamically alter the computer characteristics according to the user's objective and also to make programming of the elementary instruction set to be written into the ROM.

In order to develop a microprogram in a single microcomputer only, some supporting software should be made by using the instruction set written into the ROM that is developed beforehand by the crossassembler or the like. Consequently, for the development and checking of a microprogram, swapping the control memory for the secondary one will become necessary for frequently changing the contents of its own, accordingly the amount of the microprograms that should be resident in the control memory will be increased, and thus it will be difficult to make the control memory adaptive to the individual objectives only of the users.

On the contrary, in case where more than two persons are to use a single computer according to their individual objectives, the number of input and output units and their utilization will become a problem. The larger the number of usable I/O units becomes, the better the situation will be, but owing to such constraints as of the price, developing time, and usage, those will be a bottleneck against the design of a general-purpose experimental computer. When data for trial are needed during the development of such program, a higher intelligent I/O unit will be able to make an efficient and effective checking of the program.

Summarizing the above statements, as to the design of a general-purpose experimental computer system adaptable to the individual objectives of the users, it should be noted that the following must be satisfied:

(1) The control memory of the microprogram control computer is opened to the user.
(2) The contents of the control memory can freely and easily be altered by statical and dynamical methods of manipulation.
(3) A strong supporting software is prepared for developing a microprogram.
(4) Adaptive I/O units can be utilized freely.

This system aims at an experimental computer that satisfies the above conditions by decentralizing the functions through the combination of the HITAC-10 minicomputer and the mCOM-16 microprocessor of microprogram control. Namely, the HITAC-10 is used for developing the microprograms and for dynamically controlling the I/O services as well as the system operation and the mCOM-16 is for the execution of the user's program.

The control memory of the mCOM-16 is completely opened to the user, and thus is constructed according to the user's objectives only. Furthermore, its contents can be altered by the microprogram of the mCOM-16 as well as the

program in the HITAC-10.

The fundamental configuration of this system was developed in 1978.[6] Since then an attempt has been made to improve the software for developing the firmware of it and it has been utilized as an exclusive experimental machine adaptive to the user's objectives.[7]

2. Hardware Configuration

Figure 1 illustrates the hardware configuration of the mCOM-16 microcomputer system. All devices are connected through a bilateral bus of 16 bits called the U-bus. The microprocessor mCOM-16 is of N-channel MOS, and is composed of such two chips as CTL (control logic) and RALU (register, arithmetic and logic unit). One microinstruction is executed in 1.4 microsec.[8] The following are the explanation of this system:

(1) CTL (control logic). The function of the mPD756 of N-channel MOS is partitioned into two; one is the address control of the microinstructions, and the other the state control. Figure 2 shows the block diagram of the chip. In the address control the outputs such as increment, unconditional jump, test jump, subroutine return by address stack register, and mapping array composed of the PLA defined till the time of masking are included. In the state control, there are interrupt, hold, and reset.

(2) RALU (register, arithmetic and logic unit). This is composed of the mPD755 of N-channel MOS, whose block diagram is depicted in Figure 3. This chip comprises 15 registers, a status register, ALU, and data manipulator.

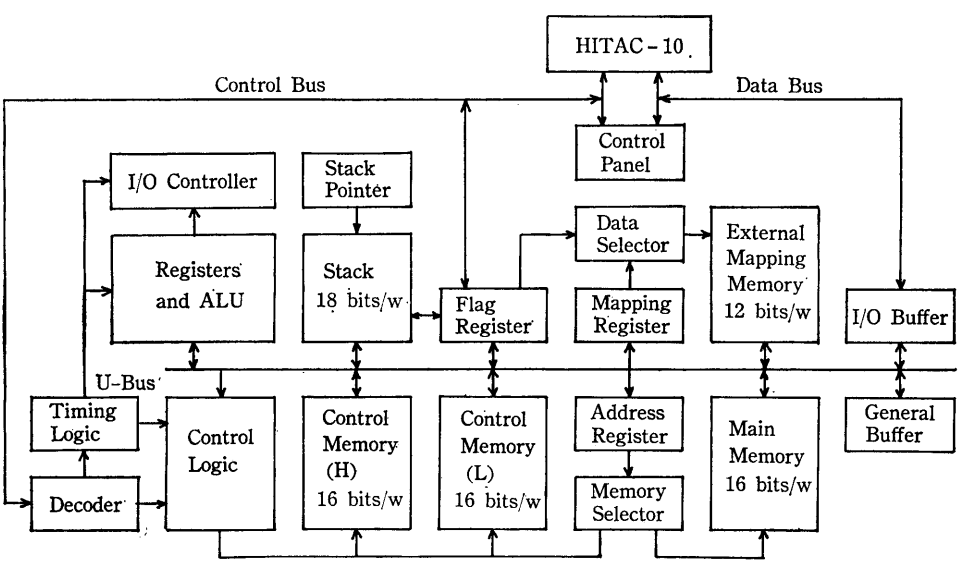


Figure 1. Hardware configuration of the mCOM-16 microcomputer system.

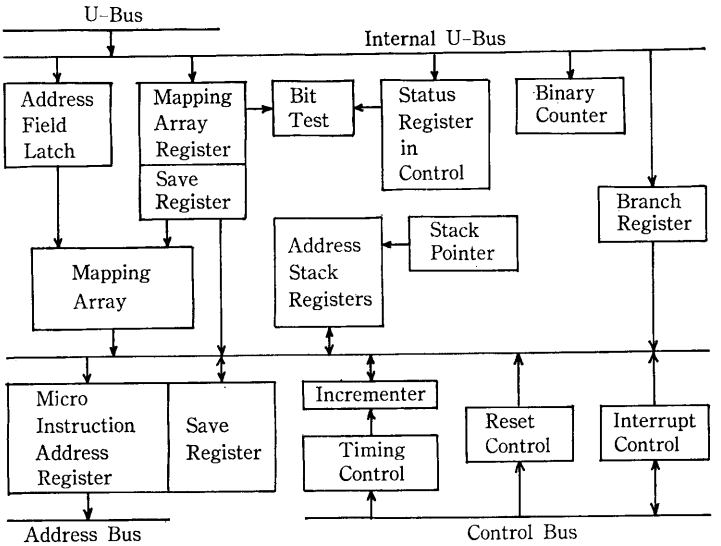


Figure 2. Block diagram of the CTL.

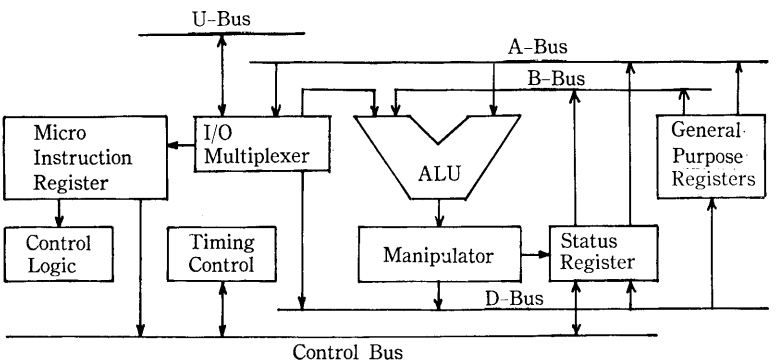


Figure 3. Block diagram of the RALU.

(3) Main and control memories. The main memory is a RAM of 16 kilowords of 16 bits/word. The control memory is composed of a RAM of 1 kiloword of 32 bits/word, and fetched to the CTL and RALU by each one of two 16 bits read out from a 32 bits. A microaddress is usually output from the CTL, but in case the contents of the control memory is to be altered during the execution of a microprogram, the memory selector of address register is switched to use its output. Namely, two machine cycles are needed; the first is to fetch a microinstruction of the microaddress put out of the CTL and write the data into the buffer register, if it is an alter instruction of the control memory. In the next cycle, a microinstruction is not fetched, but the contents of the buffer

register is written into the control memory addressed by the memory address register. The microinstruction fetched in the first cycle designates whether they should be written into the upper or lower 16 bits of the control memory. This way permits the alteration of the contents of the control memory by a microprogram, although the execution cycle of a microinstruction and the fetch cycle of the next one are normally overlapped in order to increase the execution speed as a whole. Further, the control memory can be extended up to 16 kilowords by the page address system using the 4 bits of the status register in the RALU chip.

(3) External mapping memory. This memory, 256 words, 12 bits/word, is used to generate a microaddress corresponding to the pertinent results that are decoded from the designated bits of the macroinstructions as well as the data on the U-bus, and behaves just the same way as the mapping array in the CTL, although the latter is set at the time of masking and thus impossible to alter, while the former can be altered even during the execution of a microinstruction. The data on the U-bus is latched to the mapping register, and its address is generated by

Table 1. Function of the flag register.

Bit	Function
0	Bit 16 of the stack
1	Bit 17 of the stack
2	Bit 0 of the mapping selector
3	Bit 1 of the mapping selector
4	Bit 2 of the mapping selector
5	HITAC-10 interrupt
6	Interrupt mask
7	Software interrupt
8	Address error
9	Memory parity error
10	Auxiliary
11	Interrupt request
12	Underflow of the stack
13	Overflow of the stack
14	Output buffer flag
15	Input buffer flag

selecting the bit position of the register to be decoded through the data selector by using the 3-bit value in the flag register. The output of the mapping memory is fetched to the CTL and the microaddress is determined. The mapping memory is also used as usual decoders as well as the generation of a microaddress.

(5) Stack. This is a ring stack (last-in-first-out, LIFO) of 1 kiloword, 18 bits/word, and can store 16 bits of the U-bus and 2 bits of the flag register. The information of the stack, full or empty, is set into the flag register.

(6) Flag register. This register contains 16 flags collectively, the function of which is given in Table 1. Bit positions 0 to 7 can be reset or set according to the microinstructions. Bit positions 8 to 15 are set by hardware, while bits 8 to 11 are reset by the microinstruction and 12 to 15 reset by hardware.

The execution control of the mCOM-16 system and the data transfer can be performed by the manual operation on the control panel, but the same function can also be carried out by the input/output instructions of the HITAC-10 as shown in Table 2. Control instructions are analyzed by the decoder and the

Table 2. Control instructions out of the HITAC-10.

Instruction	Function
SRT	Start
HLT	Halt
RES	Reset
INT	Interrupt
SIO	Set I/O flag
LIB	Load input buffer
KIB	Skip on input buffer flag
ROB	Read output buffer flag
KOB	Skip on output buffer flag
LMA	Load memory address register
WCH	Write control memory (H)
WCL	Write control memory (L)
WMM	Write main memory
RCH	Read control memory (H)
RCL	Read control memory (L)
RMM	Read main memory

generated control signals are given to every device on the U-bus. The access to the memory and the memory address register are only effective in case the mCOM-16 is being halted, but since the instructions of the HITAC-10 can control the operation of the mCOM-16, the HITAC-10 can completely control the mCOM-16 by its program. Data can be transferred through the I/O buffer, and two bits in the flag register are used for its control.

3. Supporting Software

The microprogram to be executed in the mCOM-16 is developed on the HITAC-10 minicomputer and then transferred back to the former. The supporting software to develop the microprogram is largely partitioned into crossassembler, editor, and file-controller and is under the control of the operating system, FDOS, in the low-price floppy disk memory. Various kinds of microprogram file developed by the individual users are stored in the floppy disk and used separately or combined with each other.

3.1 Cross-Microassembler

Generally speaking, a microinstruction has a longer word length and is partitioned into various fields. In order to make a microprogram effectively, a cross-microassembler has been generated to assemble the microinstructions written by the mnemonic codes through the HITAC-10, and to transfer them to the control memory of the mCOM-16. The features of this assembler are as follows:

(1) The microinstruction has so many fields that such fields that are considered unnecessary for the moment may be omitted, but can be supplemented by the system as the most occurrable pattern of the instruction referring to the other fields, except in case of error.

(2) Editor is being linked with the file-controller, capable of easily developing and correcting microprograms, and can completely process them in a disk base.

(3) When an error occurred, its source program will be corrected by the editor and re-assembled and the program in a micro-object level can be corrected by the error address presented by the TTY in a conversational mode.

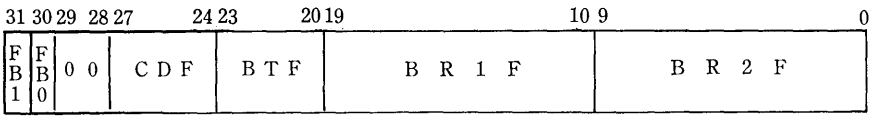
(4) This crossassembler can also describe and transfer simultaneously the contents of the mapping memory that generates a microaddress.

The microinstruction of the mCOM-16 is composed of 32 bits, which is partitioned into 4 kinds of instruction formats such as branch, register, data output, and data input according to the contents of the operation field. Figure 4 shows the formats of the individual instructions. The function of those instructions is as follows:

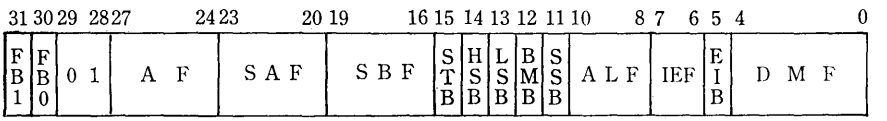
(1) Branch instruction will do address jump after the bit test of the MAR and SRC in the CTL chip or after checking the zero of BC.

(2) Register instruction has a function of operating the contents of the two general-purpose registers and of loading the result to the one of them after some data manipulations.

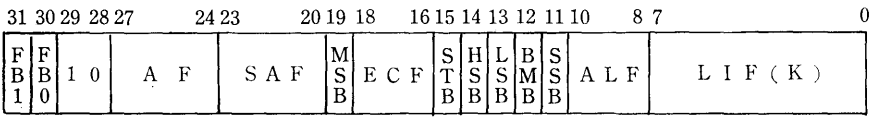
(1) Branch Operation



(2) Register Operation



(3) Data Output Operation



(4) Data Input Operation

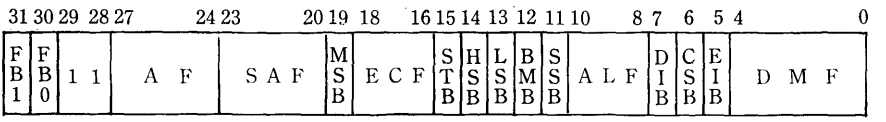


Figure 4. Format of the microinstructions.

(3) Data output instruction will put out the contents of the general-purpose register on the U-bus and at the same time loads the result of operation of the literal data with them to the general-purpose register.

Table 3 gives the format of the crossassembler against each microinstruction and the correspondence between both fields of crossassembler and microinstruction. Bracketed part, [ ], may be omitted. The microprogram written by the mnemonics will be assembled by the crossassembler to transfer it to the control memory of the mCOM-16 through the input/output instructions of the HITAC-10.

In order to decode the necessary bits of data for generating the corresponding microaddress, the mCOM-16 system is provided with two mapping circuits; one is a mapping array provided in the CTL chip, and has been set when masking so that it could not be altered but can generate a microaddress in one machine cycle. The other is an external mapping memory provided for giving the flexibility to the system, and its contents can freely be altered but it takes 3 machine cycles to generate a microaddress. The crossassembler also has the facility of stating and transferring the contents of the external mapping memory. The generation of the microaddresses by mapping circuits is called a multiple-

Table 3(a). Format of the Cross-Microassembler.

Microinstruction	Format
Branch Operation	[Symbol] Op-code [Bit] Operand-1 [Operand-2]
Register Operation	[Symbol] [Op-code] [Modify] Register-1 Register-2 [Bus] [DM] [IN]
Data Output Operation	[Symbol] [Op-code] [Modify] Register I/O-code [Bus] [Literal]
Data Input Operation	[Symbol] [Op-code] [Modify] Register I/O-code [Bus] [DM] [IN]

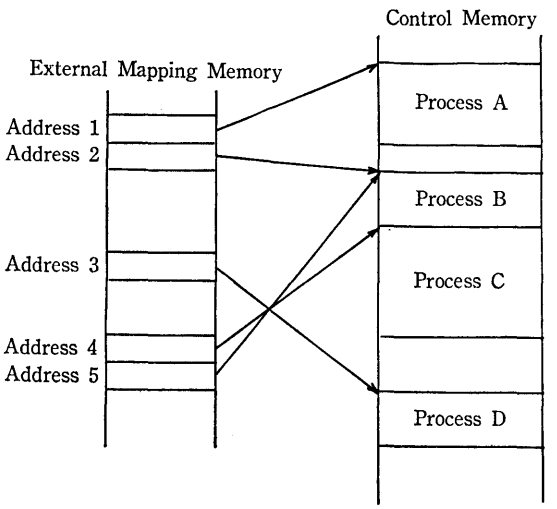
Table 3(b). Comparison between the fields of the Cross-Microassembler and the microinstruction.

Cross-Microassembler	Microinstruction
Op-Code	CDF ALF DIB
Bit	BTF
Operand-1	BR2F
Operand-2	BR1F
Modify	AF STB MSB
Register-1 Register	SAF
Register-2	SBF
Bus	HSB LSB BMB SSB
DM	DMF
IN	IEF EIB CSB
I/O-code	FB1 FB2 ECF
Literal	LIF(K)

directional jump, and provides a means of performing the various processes due to the bit patterns of the data. As shown in Figure 5, a symbol is attached to the first microinstruction of each process, and corresponds to the address of the external mapping memory which is generated by the data selector circuits.

3.2 Editor and File-Controller

In order to make a microprogram more effective by using the crossassembler, editor and file-controller have been simultaneously developed. Table 4 gives the




```
Symbol A    microinstruction A1
                  .....          [Process A]
            microinstruction Ah
Symbol B    microinstruction B1
                  .....          [Process B]
            microinstruction Bi
Symbol C    microinstruction C1
                  .....          [Process C]
            microinstruction Cj
Symbol D    microinstruction D1
                  .....          [Process D]
            microinstruction Dk
            MAP
Symbol A    Address 1
Symbol B    Address 2
            Address 5
Symbol C    Address 4
Symbol D    Address 3
```

Figure 5. Behaviour of the external mapping circuit and its description.

26 kinds of commands and their functions of the editor. The commands can be partitioned into three of compiling use, input/output, and mode transition, and put in through the data typewriter. An input character sequence is of a magic list structure,[2] which is inserted, added, exchanged, or retrieved through the manipulation of the compiling commands. Input/output commands can switch the I/O units such as TTY, PTR, or disk used for the editor and generate a microprogram effectively by using them when necessary. The mode transition command is used for the transition from the editor to the file-controller, and then to the crossassembler.

Table 4. Commands of the Editor.

Format	Function
A	Assembler
B	Bottom
C	Current pointer
D h	Delete
E	Exit
F s	Find string
G c1 s c1 c2 s' c2	Get and exchange string
H h	File-controller
I s	Insert
J h	Disk in
K s	Seek string
L h	Last
M	Memory
N h	Next
O h	Disk out
P h	Print out
Q	Quash
R s	Retype
S	Source
T h	Top
U h	Up
V h	Block transfer
W h	Write neighbour
X h	Execute
Y h	Load disk
Z	Cross-microassembler

s: string h: hexadecimal c: character

Table 5. Commands of the File-Controller.

Format	Function
MAP.	Print map
PUT. h f	Put file
GET. h f	Get file
DEFB. b f1 f2 ·····	Define block
KILL. n	Kill file or block
EXEC. n	Make execute table
FIN.	Finish

h : hexadecimal
f : file name
n : file or block name
b : block name

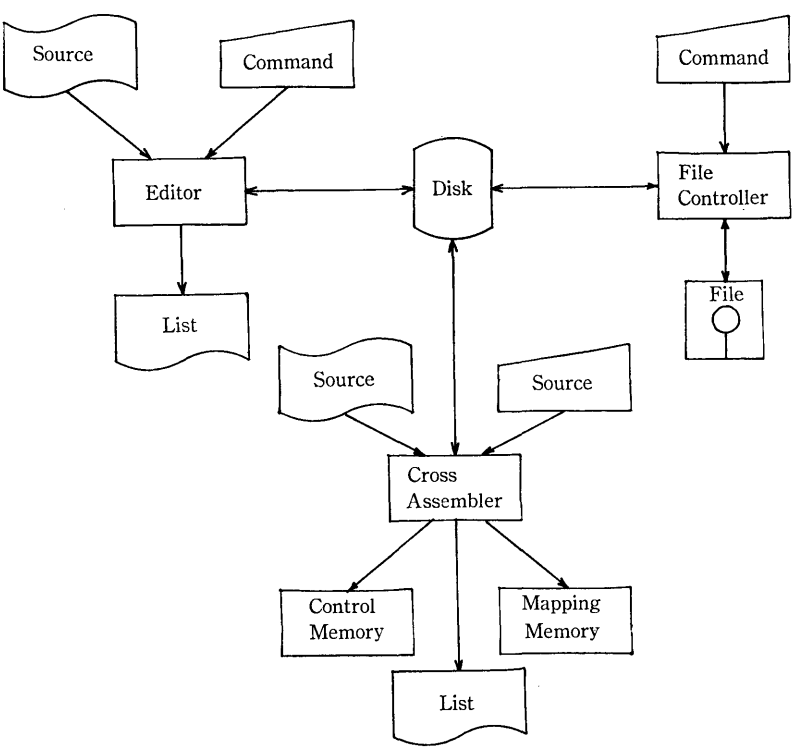


Figure 6. Information flow of the supporting software.

Table 5 gives the 7 kinds of commands of the file-controller and their functions. Considering from the fact that the mCOM-16 is aimed at a general-purpose experimental computer system, the contents of the control memory will largely be altered according to the individual objectives of the users. The file-controller can register a developed microprogram on the floppy disk as a file in a form of source or object one. Floppy disk of considerably low-price can be used exclusively for every user to develop his research program and the like effectively. Both editor and file-controller can manipulate not only the microprogram but also all of the usual character sequences, so that it can be used for the usual macroprograms.

3.3 Mode Transition

In order to develop a microprogram in practice, the procedure is as follows, the explanation of which is given in Figure 6.

(1) To make a micro-source program according to the format of the crossassembler by using the editor.
(2) To register the micro-source program on a floppy disk as a file by using the file-controller.
(3) To convert the micro-source program into a micro-object one through the crossassembler.
(4) To register the micro-object program on the floppy disk.

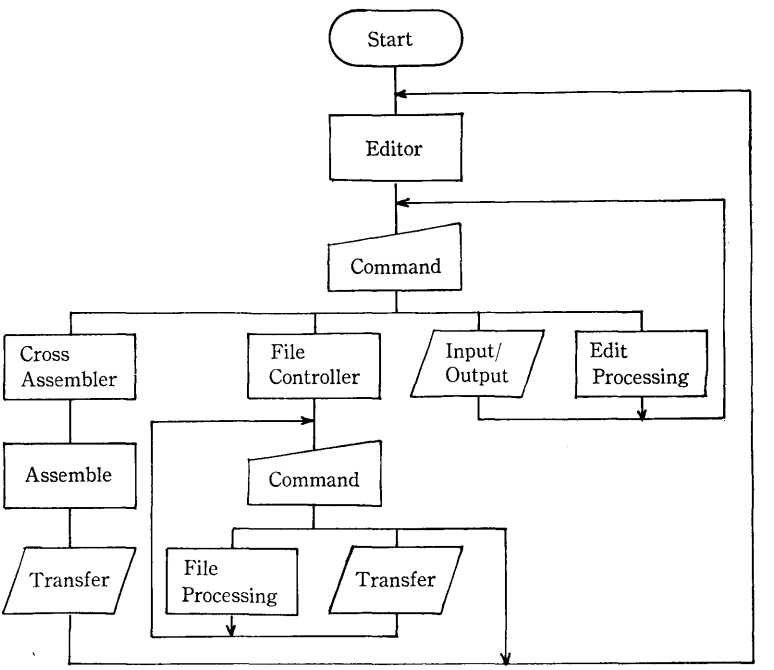


Figure 7. Mode transition of the supporting software.

(5) To transfer the micro-object program to either the control memory or the external mapping one by way of the main memory if there is a map control instruction.

In case of using the microprogram that is registered as a file on the disk, several micro-source programs are read by the file-controller first, then compiled, altered, and supplemented by the editor, and then assembled by the crossassembler. In case of using the registered micro-object program as its own state, it should be read and transferred to the control memory. Figure 7 illustrates the mode transition of the supporting software.

4. Behaviour of The System

When a microprogram developed on the HITAC-10 is transferred to the control memory of the mCOM-16, and it starts the execution through the input/output instructions of the HITAC-10 or the manual operation, the HITAC-10 comes to play a role of general-purpose I/O units of a high intelligence. The mCOM-16 has no I/O units of an exclusive use, but can utilize the I/O units attached to the HITAC-10 through the program in it. This situation can be considered that there is a buffer of the HITAC-10 between the mCOM-16 and the I/O units, so that there takes place either a problem that the execution time of programs will slow down for some of their kind or a problem that not only the program of the mCOM-16 itself but also the I/O processing programs of the HITAC-10 should be developed simultaneously. Nevertheless, the following are the distinctive features of this system as an experimental tool.

(1) Already installed I/O units can be used without any modification.
(2) Data that are exchanged between the mCOM-16 and the I/O units can be converted in an adaptable form to the user.
(3) By the program control, the I/O units that are not attached at all can be considered as virtual I/O units and test data can be transferred to the system.

On the other hand, the HITAC-10 considers the mCOM-16 as an I/O unit of its own and its input/output instructions can control the operation and data transfer of the mCOM-16 system, so that when a program that controls the operation of the mCOM-16 is carried out on the HITAC-10, the contents of the control memory as well as the main memory and the program flow of the mCOM-16 can all be changed dynamically according to either the request from the mCOM-16, the user's request, or the program control of the HITAC-10.

Figure 8 illustrates the operation of the two computers when the developed microprogram starts its execution, as explained below.

(1) Microprogram, macroprogram, and map information are transferred from the disk to the HITAC-10.
(2) The HITAC-10 transfers the information accepted in (1) to the control, main, and mapping memories.
(3) The control instruction from the HITAC-10 starts the execution of the mCOM-16.

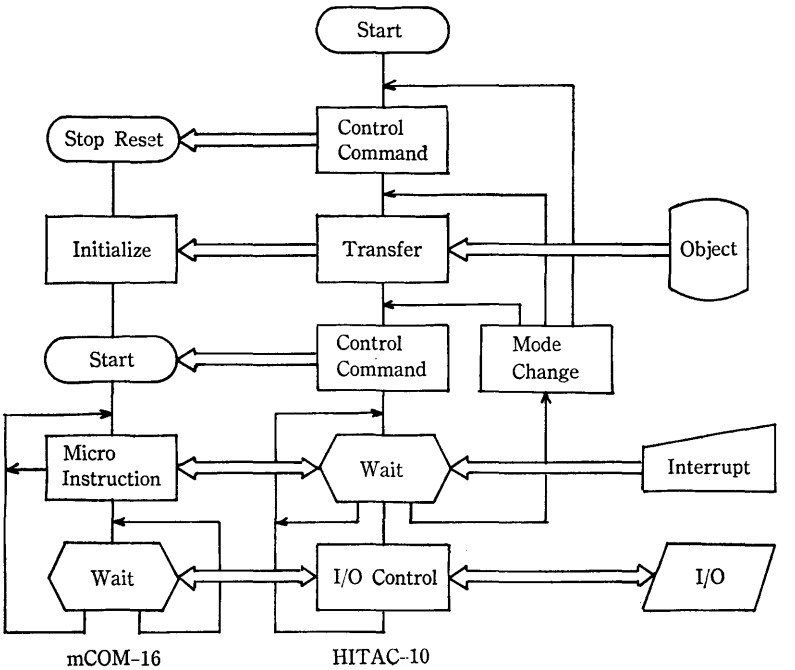


Figure 8. Information flow of this system.

(4) The microprogram on the mCOM-16 is carried out sequentially and the HITAC-10 will be set in a waiting state.
(5) According to the request from the mCOM-16 or the contents of the user's interruption, the next turn goes to (6) or (7).
(6) In the state that the mCOM-16 requires an I/O service, it will be set in a waiting state during the I/O processing of the HITAC-10. When the I/O processing is completed, the next turn goes back to (4).
(7) At the time of changing the state of the mCOM-16, the next goes back to (1) or (2), or goes to (8).
(8) The HITAC-10 will be set in a wait state after sending the control instruction to the mCOM-16, and then the next returns to (5).

The above stated is an example of the operation transitions and this system can be utilized according to the user's objectives depending on the contents of either the program in the HITAC-10 or the microprogram in the mCOM-16.

To estimate the functional capability of this system, an emulator as a concrete application is implemented that considers the HITAC-10 as the target machine.[6] The result is that the microprogram required to emulate in the mCOM-16 has 288 words, the external mapping memory 40 words, and I/O processing program of the HITAC-10 40 words. Table 6 lists the number of steps of the fetch, addressing, and execution of every macroinstruction, and the execution time of the target machine. In order to compare the execution time of the

Table 6. Number of steps of the microprogram to emulate the HITAC-10.

		mCOM-16 Microprogram		HITAC-10 Execution time	
Fetch		5 - 7	steps	1.4	microsec.
Modify	Z	6	steps	0	microsec.
	M	9		0	
	IZ	9		1.4	
	IM	12		1.4	
	IZX	11		1.4	
	IMX	14		1.4	
Execute	L	5	steps	1.4	microsec.
	A	2		1.4	
	S	2		1.4	
	N	5		1.4	
	X	5		1.4	
	O	5		1.4	
	ST	2		1.4	
	B	2		0	
	BAL	5		1.4	
	KCT	9		2.8	
	SE	2		0	
	SC	9 - 28		2.8	
	IOC	10 - 14		2.8	
	SRL	8+2*n		1.4*[n/3]	
	SLL	5+2*n		1.4*[n/3]	
	SRA	8+2*n		1.4*[n/3]	
	SLA	12+4*n		1.4*[n/3]	
	SRDL	8+3*n		1.4*[n/3]	
	SLDL	5+3*n		1.4*[n/3]	
	SRDA	8+3*n		1.4*[n/3]	
	SLDA	12+5*n		1.4*[n/3]	
	LE	2		1.4	
	LD	8		2.8	
	AD	6		2.8	
	SD	6		2.8	
	M	137		8.4	
	D	156		9.8	
	STE	2		1.4	
	STD	5		2.8	

Table 7. Mean execution time of the emulator and the target machine.

	Probability	Emulator	Target machine
Load, Store	0.40	26.9 microsec.	3.28 microsec.
Add, Subtract	0.08	27.3	3.78
Branch	0.20	25.2	2.38
Shift	0.08	66.9	6.14
Logical	0.04	28.7	3.08
Multiply	0.02	213.5	10.08
Divide	0.02	240.1	11.48
Status control	0.10	44.1	4.34
I/O control	0.06	38.5	4.20
	1.00	40.2 microsec.	3.82 microsec.

emulator with that of the target machine, the instructions of the HITAC-10 are classified into 9, for every one of which the occurring probability and the rate of addressing modes are estimated from some typical software so far developed, and the Gibson mix is calculated.

The result in Table 7 shows that the mean execution time of one macro-instruction is 3.82 microsec. for the target machine, and 40.2 microsec. for the emulator; thus the latter takes 10.5 times than the former. It is said that an emulator will generally take the execution time 3 to 10 times when emulated by a similar level of execution time. The mCOM-16 is a N-channel MOS microprocessor and its machine cycle time is 1.4 microsec., which is the same as that of the target machine. Judging from the above mentioned, the value 10.5 can be considered as an accepted one. It took a much less time for the development of the emulator owing to the supporting software and this system has been demonstrated as a useful one.

5. Conclusion

Considering from the fact that this system is not provided with the multipler, shifter, and the hardware for the floating-point arithmetic, and further that the microprogram in it is executed by the N-channel MOS microprocessor, it is not suitable for the high-speed processing, but rather adaptable to some special-purpose tools such as debuggers, and list processing, and also to the education use for the microprogramming technology.

As an example of this system utilized in practice, an interractive debugger

has been developed for the HITAC-10 assembler programs.[7] This is effectively generated by the developing and supporting software, lately made its version up, and now under operation.

REFERENCES

[1] CHU, Y., (1976): Forword and Introduction, IEEE Trans. on Computers, Vol. C-25, No. 10, 961-962.
[2] FURUKAWA, K., (1971): Magic List (in Japanese), *Jōhōshori* (Jour. of the Information Processing Society of Japan), Vol. 12, No. 4, 248.
[3] HAGIWARA, H., (1977): Microprogramming (book in Japanese), *Sangyō-tosho.*
[4] HUSSON, S. S., (1970): Microprogramming (book), Prentice-Hall, Inc.
[5] MALLACH, E. G., (1975): Emulator Architecture, Computer, Vol. 8, No. 8, 24-32.
[6] OKADA, K., YOKOYAMA, T., and KITAGAWA M., (1978): A Microcomputer System Having an Alterable Control Memory Under the Control of a Minicomputer (in Japanese), Proc. of the Annual Conference of IEE of Japan, 1474.
[7] OKADA, K., YOKOYAMA, T. and KITAGAWA, M., (1979): A Dynamic Debugging System by Multiple processors (in Japanese), Technical Reports of Information Processing, IEE of Japan, IP-79-45, 47-56.
[8] mCOM-16 Users Manual (1976), IEM-550, *Nippon Denki*, Co. Ltd.