論 文

論文 76-13-2-5

Microprocessor에 의한 NOVA의 Emulator 設計

(Design of NOVA Emulator by Microprocessor)

宋 榮 宰*

(Song, Young-Jae)

要 約

최근, Microprocessor가 염가로 入手可能함에 따라 廣範圍한 分野에 걸쳐서 사용되고 있다. microprocessor를 有効하게 사용하면 컴퓨터 시스템의 設計가 용이해진다.

이 論文에서는 MMI-6701을 사용하여 NOVA의 Emulator를 설계 하였으며 硏究의 成果로서는 NOVA를 Microprocessor로 대치하므로서 IC의 數가 約 3分의 1로 대치할 수 있었고, 또 Emulator의 効率화를 위하여 PROM의 Micro命令은 32bit로서 구성하여 4종류의 命令形式으로 設計 되였다.

Abstract

In recent years, Microprocessor have the use of extended wide field because of obtainable to low price. Design of computer system be easy to do by this microprocessor apply validly.

This Papers: NOVA Emulator designed by use of MMI-6701.

As a result of this studies, quantity of IC changed with about a third part by NOVA exchanged with Microprocessor. Micro Instruction of PROM consist of 32bit that designed Instruction Format of four kinds.

1. 序 論

Micropocessor技術의 急激한 발전에 따라 Computer System의 Architecture에 새로운 轉換點을 이루고 있다. 즉, Microcomuter는 超小型, 低價格과 더불어 시스템 設計에서는 實裝의 容易性, 仕樣變更에 대한 융통성과 LSI化에 따른 신뢰성의 向上등 많은 이점을 가지고 있으므로 Microprocessor를 効果的으로 이용 함으로서 컴퓨터의 Architecture에 많은 革新을 가져 올 것으로 期待되고 있다. 그러나 이 경우에 가장 커다란 作業은 旣存컴퓨터의 프로그람을 새로운 컴퓨터上에 사용할 수 있어야 한다. 이 작업을 보통 Program Conversion이라 한다. 여기에는 현재 다음 2가지의 기술이 가장 많이 사용되고 있다[1].

첫째는 기존 컴퓨터의 機械語를 새로운 컴퓨터로 Simulate하는 것(즉 Simulation)으로서 기존 프로그람을 그대로 새로운 시스템에서 사용할 수 있는 것이다.

그러나 이 方法은 Architecture의 類似性이 없는 경우, 기존 컴퓨터의 한개의 命令이 새로운 컴퓨터에서는 한개의 Subroutine으로 處理되기 때문에 處理速度가 大幅低下한다. 이 方法은 본래의 속도보다 1/10~1/100의 속도가 되는 경우가 많다.

둘째는 위에서와 같은 문제점에 대하여 Emulation이라는 技術이 採用되었다. 이것은 Simulator를 Hardware의 support에 의하여 高速化하는 方法으로서 Microprogram方式이 기초가 되고 있다.

여기에서 Emulator를 생각할 수 있다.

Emulator라는 것은 어떤 컴퓨터 시스템에 의해서 Architecture가 틀린 시스템의 프로그람 처리를 가능하게 하는 기구라고 정의 하였다[2][3].

Emulator의 設計를 성공시키기 위해서는 다음 3가지 문제에 대한 効果的인 해답을 필요로 한다[3].

* 正會員, 慶熙大學校 電子工學科
Dept. of Electronic Engg.
Kyung Hee University

接受日字: 1976年 5月 29日

i) Target Machine의 思想을 Host Machine에 對應시킬 필요가 있다. 여기에는 다음과 같은 것이 요구된다.

a) Main Memory의 對應

b) 입출력 시스템의 對應

c) Register, Counter, Accumulator등의 對應.

d) Source Program으로 Address 指定이 가능한 Target Machine의 모든 Resource.

ii) Emulator를 設計할때 커다란 문제로서, Host Machine의 命令Set를 증가시킬 목적으로 거기에 첨가할 小數의 機械語命令을 선택하는 작업이 있다 이 特殊命令은 Software Simultor의속도를 올리기 위하여 선택된다.

iii) Host Machine의 動作Mode와 解釋動作Mode를 有効하게 연결시킬 필요가 있다.

本 硏究에서는 Microprocessor MMI-6701(Monolithic Memories Incorporated)을 Host Machine으로 選定하고, Target Machine에는 NOVA를 사용하여 Emulator의 設計를 하였다. 그 結果 Hostmachine의 試作에 있어서 NOVA 1200 System을 MMI-6701을 中核으로 할때 약 40개의 IC로서 대치할 수 있다는 것을 기술한다.

2. 시스템의 設計思想

Hardware와 Software을 効果的으로 分擔시켜 低價格, 性能比를 좋게 하기위한 시스템을 만든다.

Traget Machine에는 Minicomputer라고 할 수 없을만큼 强力한 Software의 제공이 풍부한 NOVA를 選定한다.

Host Machine에는 高速 Bipolar Microprocessor로서 Emulation에 適合한 Microprogram制御가 가능한 4bit MMI-6701을 사용하여, 4개로서 16bit 並列處理 Microprocessor(CPU)를 만들어 off-chip 制御回路에 의해 Emulator를 구성하여 NOVA를 Emulate하였다. 이 Bipolar Microcontroller는 지금까지의 Microcomputer와 같이 1chip上에 完全한 processor의 기능을 集積한 것이 아니고, data-flow制御를 포함한 CPU의 一部分을 1chip化한 것이다. 여기에 프로그람의 순서를 control하는 制御回路를 chip에 組合하면 完全한 Microcomputer를 구성할 수 있게 된다.

3. MMI-6701의 構造

MMI-6701은 基本 Cycle Time은 200ns, 2 Address

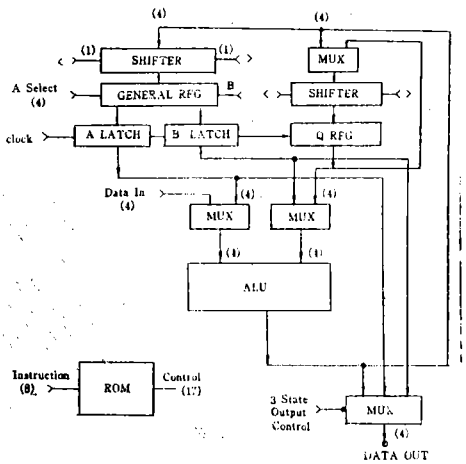

그림 1. MMI-6701의 브록圖
Fig. 1. Block Diagram of MMI-6701

方式, Instruction Set는 36命令으로 되어 있다. 그림 1에 MMI-6701의 Block Diagram을 圖示한다[4].

主要구성은 RAM(범용 Register, 16개, 4bit), Q Register(확장 Accumulator), 4bit 並列處理의 ALU, Shift Multiplexer(RAM과 Q Register에 부가되어 있음), ROM(32×9bit와 8×8bit), Latch, 出力用 Multiplexer로 구성되어 있다. RAM에는 複數의Ports가 있어서 A Address와 B Address를 同時에, 獨立的으로 Read할 수 있다. 단, Write는 B Address에서 지정한 番地만 가능하다. ROM에는 外部로 부터 命令을 주기 위하여 8個의 Instruction Line과 명령을 실행할때 필요한 Path를 開閉하는 17個의 on-chip control line이 있다.

4. Emulator의 設計

NOVA와 互換性을 만들기 위하여 MMI-6701을 4個直列로 접속해서 16bit로 한다. MMI-6701의 RAM

00	AC 0	10	WR 4
01	AC 1	11	WR 5
02	AC 2	12	WR 6
03	AC 3	13	WR 7
04	WR 0	14	EAR
05	WR 1	15	AR
06	WR 2	16	SBR
07	WR 3	17	PC

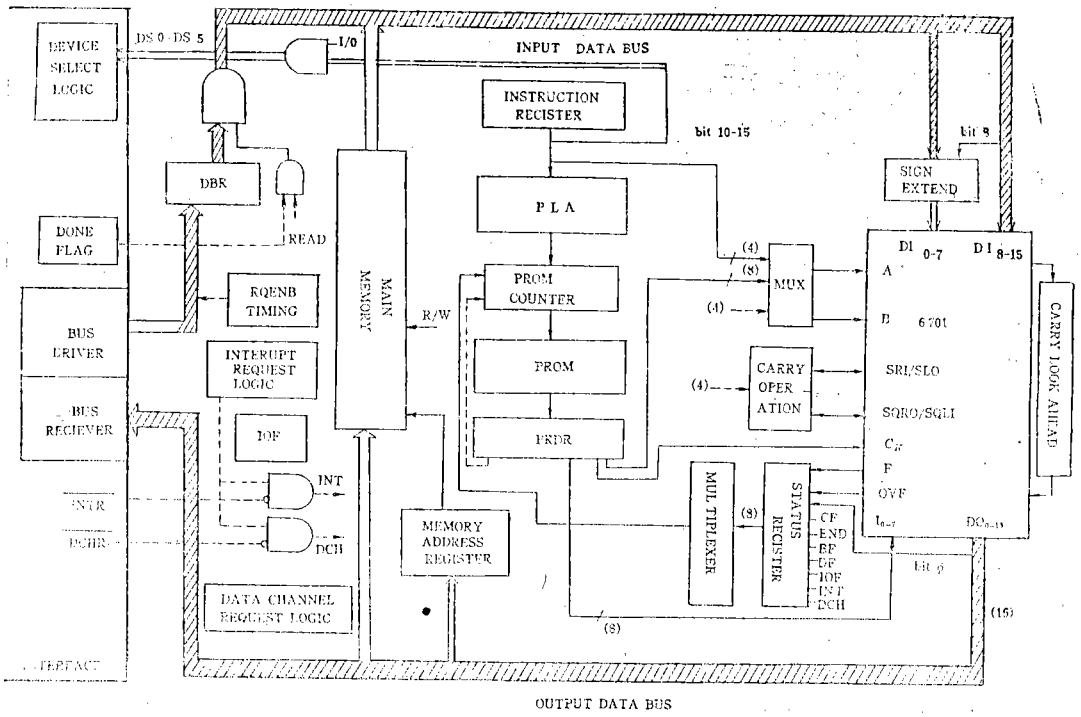

그림 2. NOVA에뮤레이터의 브록圖
Fig. 2. Block Dagram of NOVA Emulator

에는 汎用 Registor가 16개 있으므로 이것을 NOVA의 CPU에 있는 Register와 대치하기 위하여 다음과 같이 制當한다.

그림 2에 本硏究에서 設計한 NOVA Emulator의 Block圖를 표시한다.

(1) PLA:Instruction Register에 들어온 命令을 decode하기 위한 회로이다[5][6]. 本 硏究에서는 이 PLA를 사용하여 OP-Code가 해독되어 해당하는 Micro命令을 處理하는 Micro-Routne에 점프하여 마이크로命令에 應한 處理가 개시된다. 이 관계는 그림 3과 같다.

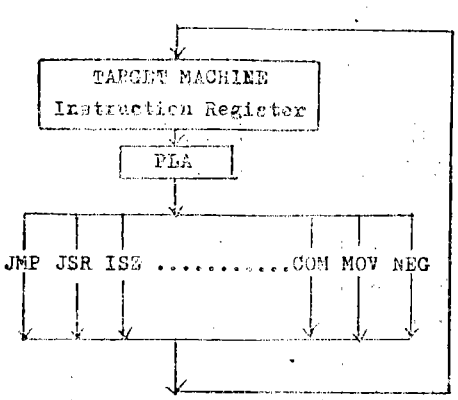

그림 3. NOVA의 IR과 PLA의 관계
Fig. 3. Relation of PLA and NiOVA's IR

(2) PROM Counter: 다음에 실행해야 될 마이크로 命令의 番地를 Microprograim Counter를 하나씩 증가 시킴으로서 實行된다.

(3) PRDR: 여기에 PROM의 命令을 하나씩 꺼집어 내어 制御信號를 발생시켜 命令을 실행시킨다.

(4) STR:Flag등을 포함하여 시스템의 여러 가지 정보를 모은 레지스터이다. 이것을 Multiplexer로서 狀態를 任意로 선택하여 PROM Counter에 信號를 보내어 命命을 실행한다.

(5) A/B Address 選擇回路 : MMI-6701의 汎用레지스터를 A Port, B Port로 부터 指定한 경우, Control ROM으로 부터 지정할 경우와 Instruction Register로

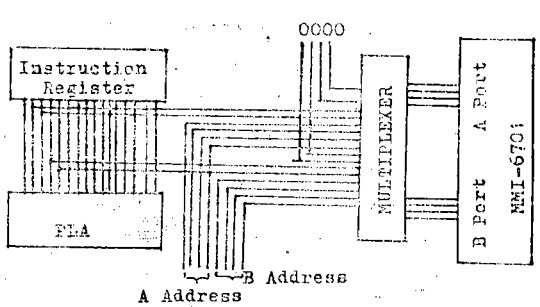

그림 4. A/B番地 선택회로
Fig. 4. A/B Address Select Circuit

부터 Accumulator를 지정하는 경우가 있다. 이 選擇은 그림 4와 같이 Multiplexer로 행한다.

(6) Shift操作, Carry조작 : Shift조작의 경우, 各 chip의 RAM, Q Register에 붙어 있는 Shifter에 각각 쌍방향성의 Shift I/O pin이 붙어 있으므로 上位 chip의 下位側 shift pin과 下位chip의 上位側 shift pin을 접속하면 된다. 그림 5의 (a)는 NOVA의 shift 方式이고 (b)는 本研究의 Emulator의 shift方式이다. 이 Emulator에서는 off-chip의 Carry Flag와 汎用레지스터의 Shifter와 Q Register의 Shifter를 그림 5의 (b)와 같이 접속해 33bit를 環狀으로 shift시킨다. 이 경우 연산시킨 16bit가 汎用레지스터의 Shifter와 Q Register와 Shifter에 同時에 들어가 shift된다.

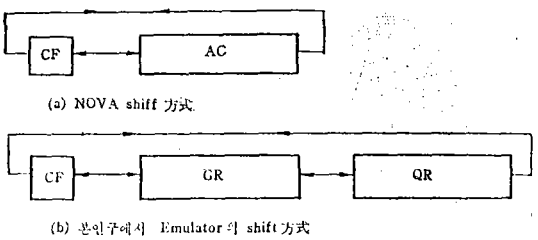

그림 5. NOVA와 에뮤레이터의 Shift方式
Fig. 5. Shift Method of NOVA and Emulator

(7) Device Selection發生回路 : 命令이 I/O에 관한 것 인때 Device를 지정하기 위해 $\overline{DS0}$~$\overline{DS5}$의 6개의 信號를 발생한다.

(8) $\overline{RQENB}$ 基本Timing發生回路 : Processor Cycle 마다 device의 Interface에 Interrupt와 Data Channel 轉送의 요구가 있는지 없는지를 묻는 pulse.

(9) Interrupt요구 판정 회로 : Device로 부터 Interrupt 요구가 오면 Interrupt가 可能한 상태인가를 판정한다.

(10) Interrupt On Flag: Interrupt가 가능한가, 어떤가를 表示하는 Flag.

(11) Data Channel Mode判定回路 : Device로 부터 요구된 DMA轉送이 어떤 Mode인가를 판정한다.

(12) Interface/Memory 선택 회로 : 入力 Data Bus에 들어오는 Data를 Main Memory로 부터 받을까, Interface側에서 받을까를 선택한다.

(13) Carry Look Ahead: Microcontroller의 各段으로 부터 나오는 Carry를 기다리지 않고, 入力의 Operand에 의하여 Carry가 있는가, 없는가를 豫測하기 위한 회로.

(14) Data Bus: MMI-6701의 Data Input Pin側에 접속 시키는 Data Input Bus와 Data Output Pin側에 접속 시키는 Data Output Bus의 2個의 Bus를 사용한다.

(15) Clock Generator: Emulator의 Cycle Time을 300ns로 하기 위한 안정발진기로서 13.34MHz를 사용한다.

以上은 NOVA를 Emulator와 Compatible하게 하기 위하여 設計한 Hardware이다.

6. Microprogram의 設計[7)8)]

위에서 設計된 시스템에 대한 마이크로프로그람은 다음과 같이 設計하였다.

6.1 Control ROM의 役割

CPU의 制御는 모두 PROM에 格納되어 있는 Microprogram에 의하여 이루어진다. 命令實行은 Instruction Fetch로 부터 시작된다. 여기에서는 命令語를 Memory로 부터 Read하여 Instruction Register에 Load함과 동시에 命令語의 8 bit 符號付 Displacement를 16bit로 확장하여 MMI-6701의 Q Register에 Load한다. 계속해서 命令語를 PLA로 해독하여 分岐先의 PROM Address를 PROM Counter로 표시하고, PROM에는 命令實行을 위하여 Routine이 格納되어 있는 PROM Counter에서 지시하는 Address의 내용을 실행한다. 命令語에서 지정 시킨 명령 실행에 있어서는 필요에 따라 Address계산 Operand Fetch를 行하고, 계속해서 각종 명령의 Sequence를 실행한다. 조건부 브란취는 브란취에 필요한 정보를 Status Register에 모아서 브란취命令에 의하여 브란취先의 PROM Address에 점프 한다. 각종 命令 실행의 최후 micro-step에서는 Interrupt 요구의 검출을 행한다.

Interrupt요구의 검출은 브란취命令으로 한다. Interrupt요구가 있으면 Interrupt처리를 行하고, 없으면 다음 命令의 Instruction Fetch를 한다.

6.2. Micro命令의 FORMAT

命令을 실행할때 동시에 실행하는 일이 없는 명령은 MMI-6701을 제어하는 命令, PLA에 관한 명령, 조건부 브란취명령의 셋으로 나눈다. 이 가운데 MMI-6701을 제어하는 명령은 MMI-6701만을 제어 하는 명령과 MMI-6701과 입출력기구를 동시에 제어 하는 명령으로 나눈다. 여기에서 4종류의 命令을 가지고 4가지

종류의 Format를 設定한다. 이 4종류의 Format를 다음과 같이 이름을 붙인다.

1) ALU Micro命令 : MMI-6701만을 制御한다.

2) Branch Micro命令 : Status Register의 內容을 검출하여, 지정시킨 PROM Address에 브란취할지의 여부를 판단한다.

3) PLA Micro命令 : PLA의 命令語 中에서 브란취에 필요한 부분을 해독 시킨다.

4) I/O Micro命令 : MMI-6701과 입출력기구의 양쪽을 제어 한다.

위에서 설명 한바와 같이 Format를 4종류로 하려면 이것을 분류 하는데 2bit가 필요하게 된다.

즉, bit 0, 1의 내용이

00 : ALU Micro命令

01 : Branch Micro命令

10 : PLA Micro命令

11 : I/O Micro命令으로 나눈다.

(i) ALU Micro命令

ALU Micro命令의 Format를 그림 6과 같이 設計하였다.

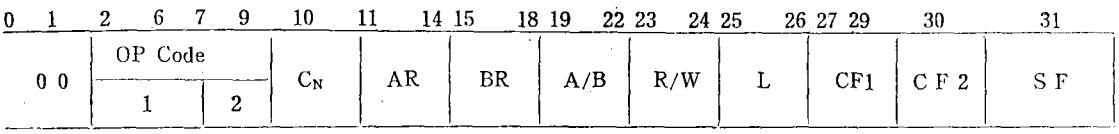

그림 6. ALU 마이크로 명령
Fig. 6. ALU Micro-Instruction.

bit 0~1 : ALU命令임을 표시한다.

bit 2~6 : MMI-6701의 연산명령을 표시한다.

bit 7~9 : MMI-6701이 Load, Shift, Data Out Control의 命令을 표시

bit 10 : Carry Input상태

bit 11~14 : A Port로 부터 General Register를 지정한다.

bit 15~18 : B Port로 부터 General Register를 지정한다.

bit 19~22 : A, B Port로 부터의 레지스터 지정은 Control ROM으로 하지만, Instruction Register에서 行하는 경우도 있으므로 이것의 선택에 사용하는 bit. Mnemonic는 Table 1과 같다.

Table. 1. Mnemonic of Register appointment,
表 1. 레지스터지정 니모닉크

bit 19~22	Mnemonic
0001	RB
0010	RA
0100	IB
1000	IA
1100	IA IB
0011	RA RB
0110	RA IB
1001	IA RB
그外	NOP

Table 1에서

IA : IR의 bit 1~2에서 지정

IB : IR의 bit 3~4에서 지정

RA : ROM의 AR Field에서 지정

RB : ROM의 BR Field에서 지정

NOP : No Operation

bit 23~24 : Read/Write信號

00 : —, NOP

01 : R, Read

10 : W, Write

11 : —, NOP

bit 25~26 : Bus上의 Data를 MAR, IR에 Load하는 命令

bit 27~29 : Carry조작 命令(Ⅰ)

000 : CZ, Carry Flag에 Zero (0)을 넣는다.

001 : CO, Carry Flag에 One (1)을 넣는다.

010 : CC, CarryFlag에 현재의 値의 補數를넣는다

011 : CF, MMI-6701로 부터 Shift Out시킨 값을 넣는다.

100, 101, 110, 111 : —, NOP

bit 30 : Carry조작 명령(Ⅱ)

0 : ↑, Carry Out

1 : —, NOP

bit 31 : Shift Field

0 : R, Shift Right

1 : L, Shift Left

(ii) Branch Micro命令

Branch Micro命令을 그림 7과 같이 설계하였다.

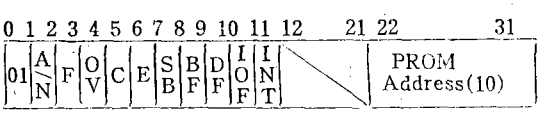

그림 7. 分岐마이크로 명령
Fig. 7. Branch Micro-Instruction

bit 0~1 : Branch명령임을 표시한다.

bit 2 : Branch명령조건의 肯定, 否定을 표시.

bit 3~11 : Branch조건을 표시한다.

F : MMI-6701의 F단자가 "0"인가

OV : Over-flow가 있는가

C : Carry Flag가 zero(0)인가

SB : Sign Bit(bit 0)가 "1"인가

E : Read/Write의 End信號가 있는가

BF : Busy Flag가 "1"인가

DF : Done Flag가 "1"인가

IOF : IOF가 "1"인가

INT : Interrupt 요구가 있는가를 표시 한다.

(iii) PLA Micro命令

PLA명령을 그림 8과 같이 설계하였다.

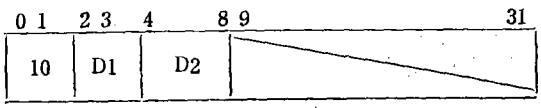

그림 8. PLA마이크로 명령
Fig.8. Micro-Instruction

bit 0~1 : PLA命令임을 표시

bit 2~3 : 명령을 다음 4종류로 분류한다.

00 : Memory참조명령에 대하여 해독한다.

01 : 연산명령에 관하여 Decode한다.

10 : I/O命令에 관하여 Decode한다.

11 : 임의의 명령이 Memory참조명령, 연산명령, I/O命令을 Decode한다.

bit 4~8의 내용은 bit 2~3의 내용에 따라서 각각의 Field를 해독한다.

(iv) I/O Micro命令

I/O Micro명령을 그림 9와 같이 설계하였다.

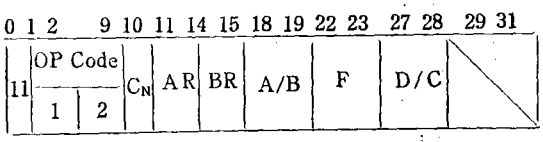

그림 9. I/O마이크로 명령
Fig. 9. I/O Micro-Instruction.

bit 0~1 : I/O명령임을 표시한다.

bit 2~22 : ALU마이크로명령과 똑같은 기능을 가진다.

bit 23~27 : NOVA Instruction에 대한 I/O명령의 Function과똑같은 Format의 기능을 가진다.

bit 28 : Device에 대한 명령, CPU에 대한 명령의 분류를 표시한다.

以上과 같이 設計한 Emulator를 效率的으로 동작시키기 위하여 마이크로명령을 32bit로 設計 하였다.

7. 結 論

設計한 Emulator와 NOVA를 서로 對應 시킬수가 있었다. 그 결과, Host machine의 試作에 있어서 NOVA 1200 System을 MMI-6701등을 포함한 約 40개의 IC로서 대치가능했다. 이것은 NOVA 1200 System보다 IC의 數가 약 3분의 1에 해당한다. 따라서, 장치가 간결 하여 지고가격면에서 현저히 低廉해진다. 또, PROM의 Microprogram은 32 bit로 구성하여 2 binary에 의해 ALU마이크로명령, Branch 마이크로명령, PLA마이크로명령, I/O마이크로명령으로 設計하였다.

參考文獻

1. S.G. Tucker: Emulation of larges ystems, Comm. ACM, Vol.8, No.12, pp.753~761, Dec., 1965.
2. S.S. Husson: Microprogramming;Principles and Pratices, Prentice-Hall, 1970.
3. E.G. Mallach:Emulator Architecture, COMPUTER, Aug., 1975, pp.24~32.
4. 4-Bit Expandable Bipolar Microcontroller;Monolithic Memories Incorporated, Aug. 1974.
5. Fleisler H.:An intoduction to Array Logic, IBM J. Res. Develop., March, pp.98~109. 1975.
6. Logue J.L., et al.:Hardware Implementation of a small system in Programmable Logic Arrays' IBM J.Res. Develop., March, 1975, pp.120~126.
7. J.Green: Microprogramming, emulators and programming languages, Comm. ACM, Vol.9, No.3, pp.230~232, March, 1966.
8. J.A. Clapp:The application of microporogramming tchnology, Sigmicro Newslett., Vol.3, No.1, pp.8~47, Jan., 1972.
9. 相磯秀夫 : ミニコンおよびマイクロプロセッサの應用, 情報處理學會, Vol.15, No.4, pp.234~239, Apr., 1974.
10. Sidney Davis:A fresh view of Mini and Microcomputers, Computer Design, May, 1974, 67~79.