論 文

論文 77-14-4-1

미니컴퓨터의 에뮤레이숀에 關한 硏究
(A Study on Emulation of Minicomputer)

宋 榮 宰*·鄭 然 澤**
(Song, Young Jae *and* Chung, Yon Tack)

要 約

마이크로프로그램을 CPU chip에 固定시키지 않고, user에게 開放한 마이크로프로셋서가 등장하고 있다. 이 論文에서는, μCOM-16에 NOVA-01의 機能을 에뮤레이트 시키기 위하여, mapping array部와 마이크로프로그램을 設計하였다.

Abstract

Microprocessor which is opened to the user, not fixed in CPU chip, has been appeared. In this paper, Microprogram and Mapping Array designed to make functions of NOVA-01 emulate to μCOM-16.

1. 序 論

최근, user microprogrammable 마이크로프로셋서가 등장하고 있다. 즉, 마이크로프로그램을 시스템에 따라 자유로히 바꾸어 넣을수 있는, 이른바 dynamic microprogram을 이용 한다면 旣存시스템의 에뮤레이숀이 간단히 이루어진다[1),2)].

따라서, 이와같은 傾向에 대하여 汎用에뮤레이숀을 考慮한 마이크로프로그램 計算機가 몇개 登場하고 있다.

1) QM—1[3),4)]

이것은 2-level의 dynamic microprogramming을 使用하여 各種시스템의 일반적인 에뮤레이숀을 目的으로 한 硏究用計算機이다. CPU는 6種類의 記憶(Main Store, Control Store, Local Store, External Register, Nano Store, F-Store)을 가지고, 그 內容에 따라 CPU의 細部하드웨어 까지 制御 할수 있도록 되어 있다.

2) B—1700[5)]

컴퓨터의 아키텍춰를 dynamic 하게 문제에 適應시키기 위하여 word size, data 形式, 固有의 構造를 전혀 가지고 있지 않다. 記憶은 빗트當 어드레스가 붙어 있어서 field의 길이가 任意로 되어 있다.

이 以外에 MLP-900[6)], The Interpreter[7)]等이 있다.

이 論文에서는, 마이크로프로그램을 CPC chip內에 固定시키지 않고 user에게 開放한 μCOM-16 마이크로프로셋서를 host machine으로 選定하고, target machine에는 NOVA-01을 選定하여 mapping array部와 microprogram을 設計에서 에뮤레이숀을 한것이다.

2. Host Machine의 槪要[8)]

日本電氣의 μCOM-16 CPU는 마이크로프로그램을 store하는 micro-instruction ROM (MIROM)과 마이크로프로그램의 sequence를 制御하는 control chip (CTL), 레지스터演算 chip (RALU)의 3部分으로 되어 있고, 그림 1과 같이 utility bus(U-BUS)라고 불

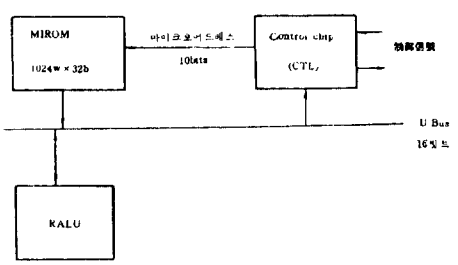


그림 1. μCOM-16 CPU시스템構成
Fig. 1. System configuration of μCOM-16 CPU

리우는 16빗트 雙方向性버스로 접속되어 있다.

CTL chip가운데에서 汎用레지스터(GR)는 마이크로命令으로 어드레스可能한 16빗트의 레지스터로서, GR_0

*正會員, 慶熙大學校 電子工學科
Dept. of Electronics Eng. Kyung Hee Univ.
**正會員, 明知大學 電氣工學科
Dept. of Electrical Eng. Myung Ji Univ.
接受日字 : 1977年 7月 25日

~GR_{15}의 16개가 있다.

mapping array는 user命令의 FORMAT에 맞추어 프로그램 시킨수 있다.

MIROM은 user가 自由로히 마이크로프로그램 할수 있도록 되어 있다.

마이크로命令은 32 bits로 構成되어 있고, Branch命令(BR), 레지스터 오페레이숀 命令(RO), data output 命令(DO), data Input命令(DI)의 4종류가 있다. 마이크로命令의 FORMAT는 表1과 같다.

表 1. 마이크로命令의 FORMAT

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR	FB		0 0		CDF				BTF				BRIF												BR2F							
RO			0 1		AF				BF				SBF				S T B	H S B	L S B	B M B	S S B		ALF		IEF	EIB		DMF				
DO			1 0										MSB	ECF											LIF(K)							
DI			1 1																						DIB	CSB	EIB	DMF				

3. Mapping Array의 設計

host machine은 mask 할때에 設定可能한, AND-OR 回路로 構成된 mapping array를 制御 chip에 가지고 있다. 이것은 마이크로命令의 AF휠드의 指定에 의하여 100種의 마이크로 storage address를 發生시킬 수 있도록 되어 있어, user FORMAT에 맞추어 프로 그램시킬 수 있다.

mapping array에의 入力情報는 MAR (Mapping Array Register)의 16빗트와 마이크로命令의 AF휠드의 4빗트로서 array의 內部는 12브록(0100~1111)으로 分類된다.

이번硏究에서는 그림 2와 같이 off-chip에 附加回路를 追加하여 mapping array를 自由로 設計할 수 있도록 하였다. 여기에서 사용한 PLA는 Intersil의 IM5200 FPLA[9] (14入力, 8出力, 48 product term)를 사용하여 10빗트의 入力데이타를 마이크로命令의 AF휠드로 指定하여 48種의 ROM어드레스로 變換시킨다.

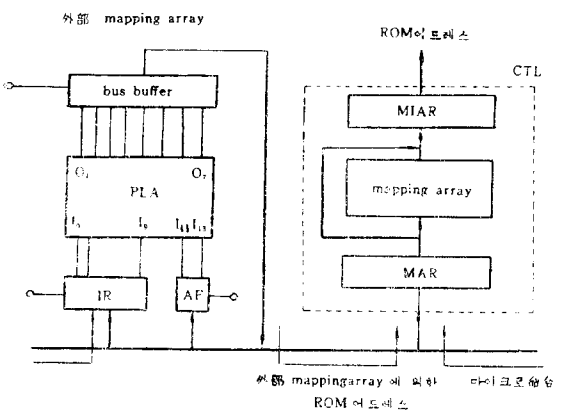


그림 2. 外部 mapping array의 追加

Fig.2. An addition of external mapping array

外部 PLA의 出力을 마이크로 命令으로 CTL의 MAR에 보내어 MIAR(micro instruction address register)에 轉送함에 의하여 內部 mapping array와 똑같은 기능을 시킬수가 있었다. 外部 PLA를 사용 하기 위하여 필요한 마이크로 step數의 增加는 거의 無視할 程度이다. 왜냐하면, 分岐가 필요하기 前의 마이크로 스텝에서 外部 PLA의 出力을 MAR에 집어 넣을수 있기 때문이다. 그림 3에 NOVA-01用으로 設計한 mapping array의 coding例를 표시한다(그림 가운데 X는 don't care bit이다.)

4. 에뮤레이숀[10, 11, 12]

일반적으로 컴퓨터의 基本動作은 "인터라프트判定→마이크로命令 fetch→마크로命令 decode→마크로命令의 實行"을 反復함에 의하여 이루어진다. 이들 動作을 마이크로프로그램의 루우틴으로 實現하는 것에 의하여 에뮤레이숀이 이루어진다. 따라서 各마크로命令이 반드시 거쳐야 할 處理를 어떻게 效率좋게 실현할까 가 문제이다. 특히 에뮤레이숀 할때에 문제가 되는것은마크로 命令의 operation code와 addressing mode의 decode效率이다. 오페레이숀 코드의 decode를 위하여 ROM을 사용하는 手法도 있지만, 마이크로命令의 OP-code가 16빗트 전체에 걸쳐서 있는것은 64kw의 ROM을 필요로 하여 不經濟的이다. 또, OP코드가 上位빗트에 集中되어 있는 경우에는 OP코드를 마이크로 루

	AF	MAR	MIAR
	3 2 1 0	F E D C B A 9 8 7 6 5 4 3 2 1 0	
1	0 1 0 0	0 0 0 0 0 X X X X X X X X X X X	JMP
2	0 1 0 0	0 0 0 0 1 X X X X X X X X X X X	JMS
3	0 1 0 0	0 0 0 1 0 X X X X X X X X X X X	ISZ
4	0 1 0 0	0 0 0 1 1 X X X X X X X X X X X	DSZ
.	.	.	.
.	.	.	.
17	0 1 0 0	1 X X X X 0 0 1 X X X X 0 X X X	NEG
18	0 1 0 0	1 X X X X 0 1 0 X X X X 0 X X X	MOV
.	.	.	.
.	.	.	.
24	0 1 0 1	X X X X X X 0 0 X X X X X X X X	ZERO PAGE
25	0 1 0 1	X X X X X X 0 1 X X X X X X X X	RELATIVE
26	0 1 0 1	X X X X X X 1 0 X X X X X X X X	INDEX 1
27	0 1 0 1	X X X X X X 1 1 X X X X X X X X	INDEX 2
.	.	.	.
.	.	.	.
45	1 0 1 0	0 0 0 0 0 0 0 0 0 0 0 1 0 X X X	$20-27_8$
46	1 0 1 0	0 0 0 0 0 0 0 0 0 0 0 1 1 X X X	$30-37_8$
47	1 0 1 1	X X X X X X X X X X X X X X X X	무조건 分岐
.	.	.	.
.	.	.	.

그림 3. Target machine의 mapping array例

Fig.3. Example of mapping array for target machine

우틴의 先頭어드레스로 可能하지만 마크로命令을 처리하는 마이크로 루우틴의 先頭를 일정 간격으로 늘어놓게 되어 마이크로프로그램 가운데에 불필요한 分岐가 많아지게 된다. 따라서 여기에서는 PLA를 사용함에 의하여 任意필드의 bit pattern에 對應하는 出力데이타를 발생 할수 있도록 設計하였다.

target machine은 4개의 汎用 Accumlator를 가지고 있고, 그 가운데 2개는 Index Register로서 사용 할 수 있다. 또 8進表現의 20~27, 30~37번지가 間接指定으로 사용되면, 그內容이 각각 Increment, Decrement 시키는 기능을 가지고 있다.

host machine에 target machine을 실현 시키기 위하여 μCOM-16의 16개의 레지스터를 NOVA의 4개의 Accumulator 및 다른 하드웨어 레지스터用으로 割當하고, 나머지 레지스터는 에뮤레이숀 프로그램을 위하여 사용한다. GR_{15}는 Status Register로 사용한다.

그림 4는 에뮤레이숀을 行하기 위한 하드웨어 構成을 표시한다. 이것은 그림 5와 같은 타이밍으로 T_1~T_8까지의 信號에 의하여 이루어 진다. 1 machine cycle은 T_1~T_4까지의 타이밍으로 이루어 진다(=1.4 μs). 보통, 1마이크로命令은 1 machine cycle로 실행 되지만, 그림 5와 같이 1 machine cycle가운데에는 마이크로 命令 實行과 다음의 마이크로命令 fetch를 overlap 시키고 있다. status情報를 轉送하기 위하여 STB가 1일때 만은 2 machine cycle을 필요로 한다(T_1~T_8). 마이크로命令은 T_3時點에서 다음 마이크로 命令의 어드레스 내용이 MIROM에 보내져서, 다음의 machine cycle T_1에서 MIROM에서 읽어낸 마이크로 命令의 최초의 16bits가, 다음의 T_2에서 나머지 16 bits가 U버스를 통하여 CTL및 RALU에 보내진다. CTL에서는 T_1, T_2사이에 마이크로命令의 順序制御필드 指定에 의하여 mapping array를 動作시켜 다음의 어드레스를 결정하여 MIROM에 設定한다. 主記憶 가운데의 마이크로命令은 마이크로프로그램으로 구성된 fetch루우틴을 실행함에 의하여 CTL의 MAR에 들어가게 된다. 그 코드를 mapping array로 各種의 마이크로프로그램 루우틴에 mapping 하면서 마크로 命令의 기능을 마이크로프로그램으로 실행한다. 그림 6에 에뮤레이숀 프로그램의 flowchart를 표시한다. 이것은 CTL에 reset信號가 들어오면, 마이크로프로그램은 0番地로 돌아가 console panel써비스 루우틴을 實行하도록 되어있다. 콘솔로 부터 start 要求가 있으면

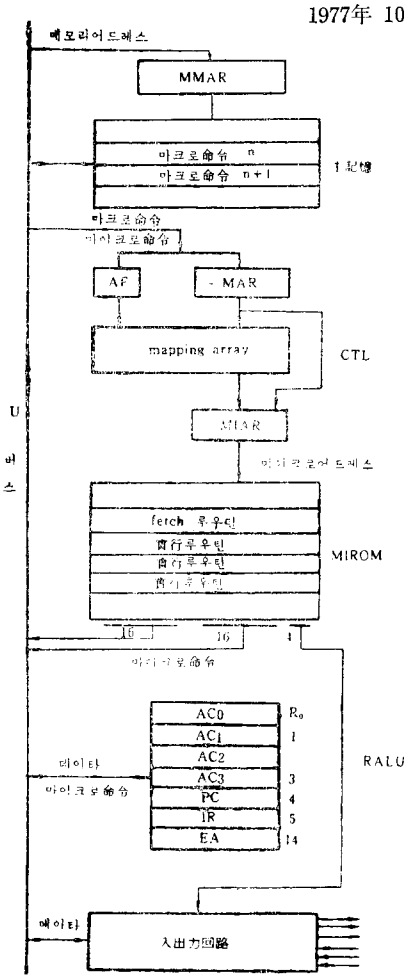


그림 4. 에뮤레이숀을 위한 시스템 構成
Fig.4. System configuration for emulation

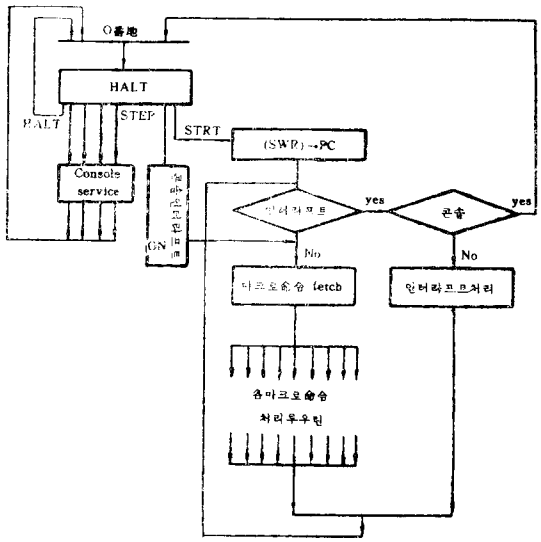


그림 6. 에뮤레이숀 프로그램의 flowchart概要
Fig.6. The flowchart summary of emulation program

Switch Register(SWR)에 設定시켜 있는 start 番地를 마이크로命令으로 프로그램카운터(PC)에 셋트 한다. 다음에 인터라프트 flag의 상태를 파악하여, 인터라프트가 있으면 콘솔 패널 써비스 또는 인터라프트의 處理를 行한다. 인터라프트가 없으면, PC의 내용을 主記憶의 address register에 보내어 마이크로命令을 fetch 한다. 마이크로命令은 fetch할때에 CTL의 MAR

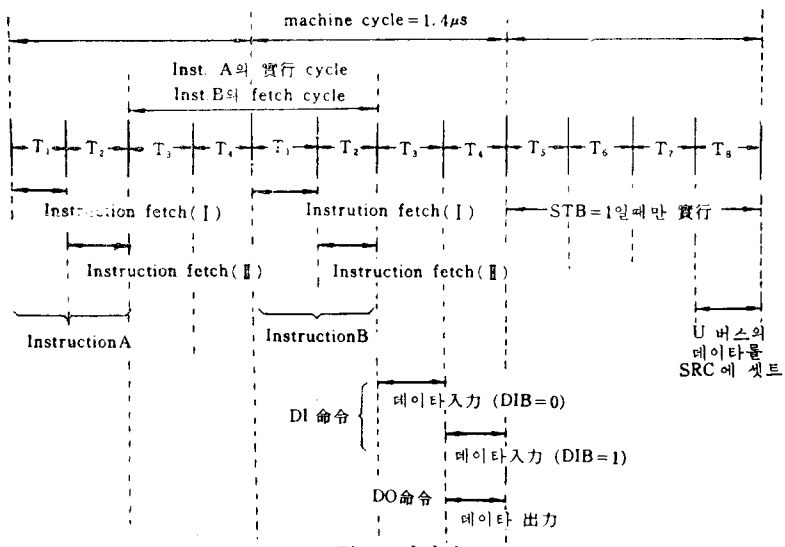


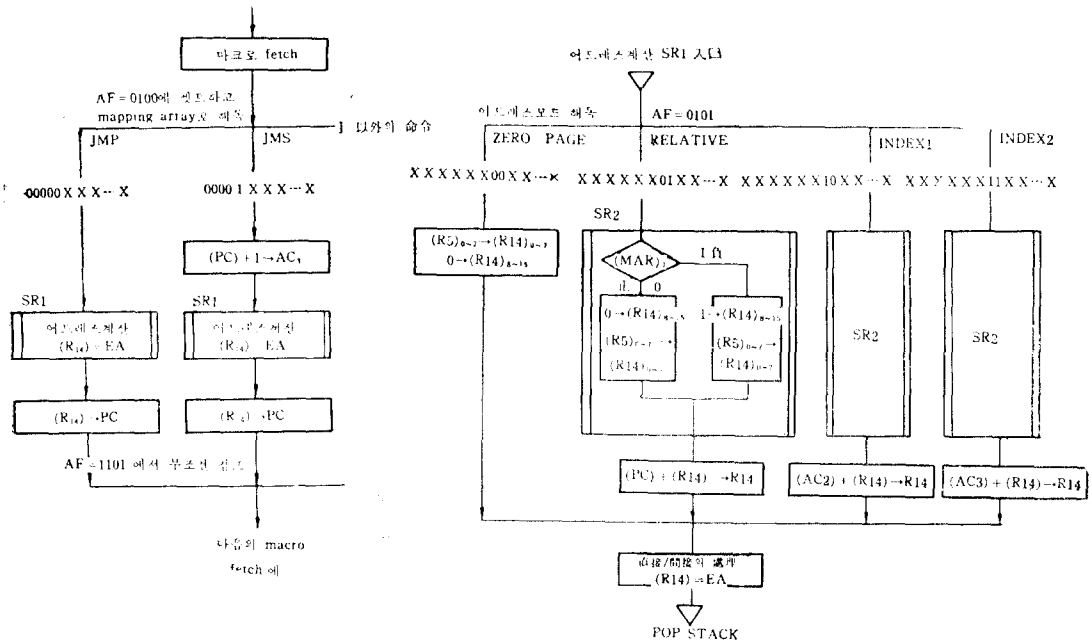
그림 5. 타이밍
Fig.5. Timing

과 RALU의 R5(Instruction Register)에 同時에 읽어 두고, 각각 命令의 解讀및 어드레스計算의 目的에 利用한다. 마크로命令을 MAR에 넣고나서 부터 마이크로命令의 AF휠드를 mapping array의 operation decode에 相當하는 값으로 한다. 이것에 의하여 다음의 마이크로 싸이클에서, mapping array에 의하여 OP코드가 해독되어 該當하는 마이크로命令을 처리하는 마이크로루우틴에 점프하여 마이크로命令에 따라 처리가 개시된다. 各處理루우틴의 실행과정에서 addressing모드等 마크로命令의 部分휠드의 解讀이 필요한 경우에는, R5의 내용은 그대로 保存시켜 있으므로 解讀시킬 마크로命令의 빗트 휠드에 對應한 mapping array의 브록을 AF로 指定함에 의하여 處理루우틴에 分岐가 이루어진다.

5. 마이크로프로그램의 設計

마이크로프로그램을 記述하는 ROM은 최종적으로 1K×8 bits의 ROM을 4개 사용하나, 試作段階에서는 마이크로프로그램의 debugging을 考慮하여 static RAM을 사용한다.

마크로命令의 處理를 어떻게하여 마이크로 프로그램으로 處理하는가를 표시하기 위하여 NOVA-01의 JMS (Jump Subroutine)命令을 例로 들어 論하기로 한다. 그림 7, 그림 8에 著者가 設計한 마이크로프로그램의 flowchart와 coding例를 표시한다.



(a) macro命令의 에뮤레이숀 (b) 어드레스計算 마이크로 서부루우틴

그림 7. 마이크로프로그램의 flowchart
(JMS 마크로命令의 解讀과 그 處理에 關한 部分)
Fig. 7. The flowchart of microprogram
(Decode of JMS micro-instruction and a part of the treatment)

마크로命令을 마이크로의 DI命令으로 MAR과 R5에 동시에 보낸다(그림 8의 ⓐ). AF휠드를 0100에 셋트한 마이크로 命令(ⓑ)을 주므로서 다음의 마이크로싸이클에서는 mapping array의 0100브록이 有效하게 되어 OP decoder로서 動作한다. 따라서 MAR 內容의 上位 5빗트 (지금의 경우 JMS의 코드 00001)에 의하여 JMS 처리의 마이크로루우틴에 점프한다(ⓒ). JMS처리루우틴 가운데의 어드레스計算 處理의 部分은 메모리참조命令에서 共通으로 사용 할수 있도록 서브루우틴化하여 사용한다. JMS命令에서는 실행중의 PC의 내용을 +1하여 AC3에 格納할 필요가 있기 때문에 JMS 명령 처리의 先頭의 2마이크로 스텝으로 이處理를 끝내고 나서(ⓒ,ⓓ), 다음에 BO 命令으로 어드레스 계산 루우틴을 call한다(ⓔ). 서브루우틴을 부르는 경우에는

MICROPROGRAM EXAMPLE
MACRO FETCH→OPDECODE→JMS EXECUTE→
BACK TO FETCH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FB1	FB0	0	0	CDF				BTF				BRIF												BR2F							
		0	1	AF				SAF				SBF				STB	HSB	LSB	BMB	SSB	ALF	IEF		EIB	DMF						
		1	0									MSB	ECF									LIF (K)									
		1	1																			DIB		CSB	EIB	DMF					

마이크로 어드레스 (8進)									
010	0110	0001	0100	0000	00000	000	00	0	00000
011	1011	0001	0101	1000	00000	000	10	0	00000ⓐ
012	0001	0100	0000	0000	00000	000	00	0	00000ⓑ
016	0010	0001	0100	0000	01110	110	00	0	00001ⓒ
017	0001	0001	0011	0100	01100	110	00	0	00000ⓓ
020	0000	0011	0000	0000	00000	000	01	0	01000ⓔ
021	0001	1011	0100	1110	01100	000	00	0	00000ⓕ
110	0001	0101	0000	0000	00000	000	00	0	00000ⓖ
111	0001	1100	1110	0101	01110	000	00	0	00000ⓗ
112	0000	0011	0000	0000	00000	000	01	0	11110
113	0001	1100	1110	0100	01100	110	00	0	00000ⓘ
114	0000	0000	0000	0000	00000	000	01	0	11110
115	0001	1100	1110	0010	01100	110	00	0	00000ⓙ
116	0000	0011	0000	0000	00000	000	01	0	11110
117	0001	1100	1100	0011	01100	110	00	0	00000ⓚ
120	0000	1101	1010	0100	11010	101	00	0	11001ⓛ
135	0001	0010	0000	0000	00000	000	00	0	00000
136									ⓜ

```
FTCH: DO PC (R4)→ADR, FB=01 (READ)
    : DI MACRO→MAR, IR(R5)
    : RO NOP OPCODE DECODE                    AF=0100
JMS : DO  (PC)+1→PC
    : RO (PC)→AC3 (R3)
    : BO CALL ADRS ROUTIN (PUSHSTACK)
    : RO EA(R14)→PC                           AF=1011
ADRS: RO NOP ADDRESS MODE DECODE              AF=0101
ZPAG: RO(R5)0~7→(R14)0~7, 0→(R14)8~13         AF=1100
RELA: BO CALL SR2 (DISPLACEMENT)
    : RO (PC)+(R14)→R14                       AF=1100
IDX1: BO CALL SR2
    : RO AC2 (R2)+(R14)→R14                   AF=1100
IDX2: BO CALL SR2
    : RO AC3 (R3)+(R14)→R14                   AF=1100
D/I : BO TEST D/I
        INDIRECT처리
    : RO RETURN (POP STACK)
SR2 :
```

그림 8. 마이크로프로그램例(JMS命令部分)

Fig. 8. Example of Microprogram (a part of JMS instruction)

마이크로프로그램으로 사용할수 있는 addres stack에 實行中의 마이크로 어드레스를 push하는 方法을 採用한다. 이 stack의 利用은 마이크로프로그램을 RAM으로 실현하는 경우에는 반드시 필요한 것이다.

어드레스計算 루우틴(그림 8의 { 部分)에서는 求하여진 實効어드레스를 R14에 넣고, Address Stack을 pop하여 main routine에 돌아간다. 어드레스計算의 終了後 R14에 들어있는 實効어드레스를 PC에 옮기면 JMS命令의 처리는 끝난다 (ⓕ). 이때 마이크로명령의 AF훽드를 1011에 指定함에 의하여 mapping array를 통하여 無條件 다음의 macro命令을 fetch하는 命令 fetch루우틴에 돌아 갈수 있다. PLA을 사용한 方式에서는 共通루우틴으로 돌아 가는데에 積極的으로 이 技術을 사용한다. 일반적으로 垂直型마이크로프로그래밍에서는 모든 마이크로命令에 Next Address Field를 넣지 않으므로 처리 step數가 增加 하지만, 이번 에뮤레이숀에서는 PLA를 사용함에 의하여 이 缺點을 어느程度 카버 할수 있었다. 어드레스計算 서브루우틴 가운데에는 최초에 AF를 0101으로 하고, MAR에 남아있는 마크로命令의 X部를 해독하여 4가지의 어드레스 모드로 分岐시킨다(ⓖ).

Zero page mode 에서는 R5에 넣어둔 마크로 命令의 上位8빗트를 0로 하고, 下位 8빗트는 그대로 R14에 轉送한다(ⓗ).. 이操作은 마이크로의 RO命令으로 ALU에의 上位byte入力을 0로 하는 機能을 사용하여 1step으로 실현 할수 있다.

相對모드와 Index모드에서는 마크로 命令의 displacement部를 16빗트의 2의補數로서 취급하기 때문에 앞의 處理가 필요하였다(ⓜ).) 이 서브루우틴 SR2에서는 displacement部의 MSB(最上位빗트)가 1이면 負의 數로서 上位바이트를 모두 1로, MSB가 0이면 正의 數로서 上位바이트를 모두 0로 셋트하여 16빗트의 2의 補數表現으로 한다. 正負의 判定은 MAR 內容의 7빗트째를 마이크로명령의 BO命令으로 테스트 함에 의하여 이루어진다. (그림 6의 (b)참조). 앞의 처리가 끝났으면 다음의 處理는 간단하다. 相對모드에서는 PC와 R14(ⓘ), Index 1모드에서는 AC2와 R14(ⓙ), Index 2모 드에서는 AC3와 R14(ⓚ)의 內容을 각각 加算하는 것만으로 된다. 4가지 모드에 대한 어드레스計算에 대하여 다시 直接/間接 어드레스의 區別을 하여야 한다. 그때문에 MAR의 10빗트째를 BO命令으로 다시 한번 테스트한다(ⓛ). 0이면 直接指定이고, 지금까지 얻어진 實効어드레스를 그대로 어드레스計算루우틴에서 빼어낸다. 1이면 間接指定이므로 R14의 값을 메모리의 間接指定番地로서, 그番地의 內容을 읽어 내어서 MAR과 R14에 집어 넣는다. NOVA에서는 特定番地가 間接指定에 의한 auto Index로서 사용되므로 間接指定시킨 番地의 領域判定이 必要하다. 間接指定番地를 마이크로의 DO命令으로 버스에 태워서 어드레스레지스터에 보냄과 동시에 MAR에도 보낸다. 그 마이크로命令의 AF를 1010에 셋트함에 의하여 그 番地가 該當하는 範圍에 있을까 없을까의 判定과 同時에 處理루우틴의 分岐까지를 1micro step으로 실현할수 있었다.

入出力動作은 正確한 타이밍까지 emulate 하는것은 어려웁고, 實現하려고 하면 付加 하드웨어의 負擔이 커지게 된다. 따라서 入出力動作의 에뮤레이숀은 入出力데이타와 다른 制御線上의 信號와의 相對的인 sequence 만을 마이크로命令으로 실현 하는 方法을 採用하였다. 具體的으로는 32bits의 마이크로命令을 4빗트擴張하여, 이것을 I/O펄스 制御훽드로서 使用함에 의하여 이루어진다. 入出力마이크로命令의 device code部는 主記憶에서 그것이 읽어 내어졌을때에, 專用으로 만들어 놓은 device code 레지스터에 latch하여 두고, 그 다음에 데이타를 I/O데이타 버스에 태우는 方法을 採用하였다.

주변장치의 상태후랙(status flag)을 檢出하여 skip하는 機能은 마이크로命令의 I/O function部를 解讀한 信號와 外部에서 오는 flag信號의 AND操作을 하드웨어의 게이트로 行하여, 그出力을 RALU의 EI端子에 보내고, SR에 집어넣어 bit test를 하도록 하였다. 이以外의 마이크로命令도 똑같은 方式으로 프로그램하여 간다.

6. 結 論

內部 mapping array 대신에 外部에 PLA를 追加시켜서 decode의 効率을 增加시킬수 있었고, target machine用의 마이크로프로그램을 設計하여 에뮤레이숀에 適合한 시스템을 構成할수 있었다. 에뮤레이숀에 必要한 마이크로프로그램量은 約 700step 이었다. 1micro 命令의 實行速度는 12~30μs(8~20micro step)이었다.

參 考 文 獻

1. 宋榮宰 : 컴퓨터의 建築樣式(著), 翰信文化社, pp.241~272, 1977.
2. E.G.Mallach: Emulation-Practice and Principles, Honeywell Information Systems, Oct. 1972.
3. QM-1 hardware level user's manual, 2nd edition, Nanodata Corp., March, 1973.
4. R.F. Rosin, et al: An environment for research in microprogramming and emulation,

CACM Vol. 15, No.8, pp. 748~760.
5. W.T.Wilner: Design of the Burroughs B1700. Proc. FJCC, pp.489~497, 1972.
6. H.W. Lawson and B.K.Smith: functional characteristics of a multilingual processor, IEEE Trans. C-20, No.7, pp. 732~743.
7. E. W. Reigel, et al: The interpreter-A microprogrammable building block system, Proc. SJCC, pp.705~723, 1972.
8. μCOM-16 解設書, 日本電氣, 1975年
9. FPLA IM5200CJG manual, Intersil, Oct. 1975
10. 宋榮宰 : Microprocessor에 의한 NOVA의 Emulator 設計, 電子工學會誌, Vol. 13, No.2, pp. 28~33
11. 宋榮宰 : NOVA 에뮤레이터의 시뮤레이숀에 關한 硏究, 電子工學會誌 Vol. 13, No.2, pp.34~39
12. 宋榮宰 : 컴퓨터設計의 마이크로프로그래밍, 電子工學會雜誌, Vol.3, No.1, pp.18~24
13. S.H. Fuller: Microprogramming and its relationship to Emulation and Technology, ACM micro the Seventh Annual Workshop, Sep. 30-Oct. 2, pp.151-158, 1974.
14. S.G. Tucker: Emulation of Large Systems, CACM, Vol.3, No.12, pp.753-761.
15. G.R. Allred: System/370 integratede emulation under OS and DOS, SJCC (1971), pp. 163-168.
16. R.I. Benjamin: The Spectra 70/45 Emulator for the RCA 301, CACM, Vol.3, No.12, pp. 749-752.
17. R.F. Rosin: Contemporary concepts of microprogramming and Emulation, Computing Surveys, Vol.1, No.4, pp.197-212.