디지탈시스템과 마이크로 프로세서 설계(Ⅲ)

金 明 恒 *

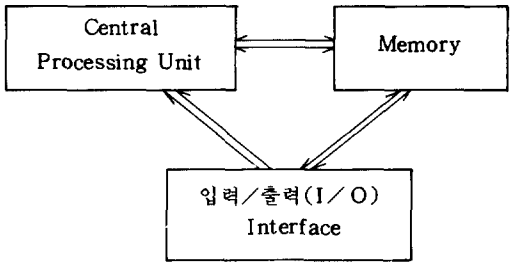
요 약

소형 컴퓨터의 구조를 공부하고 체계적 설계방법을 응용해서 소형 컴퓨터를 설계한다. 컴퓨터의 3 소자는 중앙처리장치(CPU)와 memory 와 입력/출력 Interface 이며 컴퓨터 Operation은 instruction을 써서 시행하는 것을 설명한다. 소형 CPU 구조와 설계를 공부하기 위해서 주어진 16 Assembly Language Instruction을 시행할 수 있는 CPU 설계에 체계적인 설계 방법을 응용한다.

1. 서 론

컴퓨터는 많은 분야에서 데이타처리와 계산에 있어서 여러 가지 중요한 역할을 하고 있다. 컴퓨터는 네일 잘 알려져 있는 디지탈 시스템이다. 컴퓨터의 특징은 INSTRUCTION으로 되어있는 PROGRAM을 주어진 데이타에 시행하는 것이다. 따라서 사용자가 필요에 따라서 PROGRAM과 데이타를 변경할 수 있어서 수없이 많은 정보처리의 일을 할 수 있다.

이 체계적 설계 방법을 컴퓨터 설계에 응용할 수 있다. 컴퓨터를 아래 그림과 같이 세부 시스템으로 나눌 수 있다.



正會員 : 美國 Cornell 大 工大 電氣工學科 教授 · 工博

Central Processing Unit (CPU)는 PROGRAM에 요구되는 계산과 데이타 처리 일을 한다. Memory는 INSTRUCTION으로 되어있는 PROGRAM, 입력과 출력 데이타나 결과 등을 저장한다. 사용자가 준비한 PROGRAM과 데이타를 I/O interface 에 연결되어 있는 punch-card read 나 CRT Terminal같은 입력장치를 써서 memory에 넣는다. I/O interface에 연결되어 있는 Printer 나 CRT Terminal 같은 출력장치는 CPU가 PROGRAM에 따라서 계산이나 데이타 처리한 결과를 받아서 사용자에게 보여준다. I/O 장치는 전자회로로 제어기계장치로 되어있는 특정디지탈 시스템일 수도 있다.

모든 컴퓨터는 PROGRAM에 쓰이는 INSTRUCTION SET가 있다. INSTRUCTION은 bit로 되어 있는 code로서 컴퓨터가 어떠한 operation을 하는 것을 지시한다. CPU가 PROGRAM으로 저장되어 있는 INSTRUCTION을 순서로 하나씩 읽어서 지정된 operation을 시행한다.

CPU도 체계적 설계 방법에서 취급하는 디지탈 시스템 같이 제어부분과 data flow 부분으로 되어 있다. 또, 각 INSTRUCTION을 시행하는 operation은 microoperation으로 형성된다. Data Flow부분은 Arithmetic and Logic Unit (ALU)와 register과 Condition Register (CR)와 Instruction Regist-

따라서 operand이 CPU register에만 있을 수 있고 또는 memory에 있을 수 있다. Instruction에 시행하는데 operand가 Accumulator register에 있는 것을 implied mode라 한다. 이 컴퓨터의 첫 여섯 instruction이 implied mode이고 이러한 instruction은 1 word instruction이다. Operand가 memory에 있을 경우는 direct addressing mode라 한다. 이 컴퓨터에서는 주 memory가 256 word이므로 address에 8 bits이 필요하다. 그러므로서 direct addressing mode는 instruction이 3 word이며 두번째 word는 낮은 4 bit와 세번째 word는 높은 4 bit로 addressing을 한다.

이 외에도 operand가 instruction 내에 있을 때는 Immediate Addressing mode라 하며 operand가 op code 다음에 오므로 [illegible] word이면이 Instruction은 2-word Instruction이다. Operand가 CPU에 있는 register 있을 때 register mode라 하며 이 Instruction을 1 word Instruction이다. Operand가 memory에 있으며 이 operand 주소가 CPU에 있는 register에 기억되여 있을 때 이 Instruction은 register indirect mode라 한다. 이 Instruction은 1 word Instruction이다. Indexed Addressing mode는 Instruction이 3 word Instruction이며 나중 2 word는 address를 표시한다. Operand이 있는 memory address를 결정하기 위해서 Instruction의 주소 부분인 나중 2 word에다 Index register에 있는 수를 더해 준다.

우리가 설계하려는 컴퓨터 Instruction에서 memory reference instruction은 direct addressing mode이며 3 word로 되어 있으며 나머지 Instruction은 1 word로 된다. 표 2와 같이 Instruction의 Binary code를 정할 수 있다.

첫 단계로 CPU 구조정의 선언(declara[illegible]
[illegible] 같이 할 수 있다.
[illegible] : IR[4], AC[4], TMP[4], MD

표 1. 설계할 컴퓨터 Instruction의 정의

Register Reference Instructions

NOP	no operation
SKZ	skip next instruction if accumulator, AC, equals zero
SKL	skip next instruction if link, ℓ, equals zero
CL	clear link
RAR	right rotate AC thru the link bit
COM	complement the accumulator
HLT	halt
XXX	your choice

Memory Reference Instructions

LAD, L, H	AC ← M(H, L)
AND, L, H	AC ← AC ∧ M(H, L)
ADD, L, H	ℓ, AC ← AC + M(H, L)
DSZ, L, H	M(MA) ← DEC(M(MA) decrement the content of the memory location and skip next instruction if the result is zero
J, L, H	unconditional program jump
STD, L, H	M(MA) ← AC
XXX	your choice
XXX	your choice

표 2. Instruction의 Binary Code

1-Word Instructions

IR_3	IR_2	IR_1	IR_0	Instruction
0	0	0	0	NOP
0	0	0	1	SKZ
0	0	1	0	SKL
0	0	1	1	CL
0	1	0	0	RAR
0	1	0	1	COM
0	1	1	0	Optional
0	1	1	1	HLT

3-Word Instructions

IR_3	IR_2	IR_1	IR_0	Instruction
1	0	0	0	LAD
1	0	0	1	AND
1	0	1	0	ADD
1	0	1	1	DSZ
1	1	0	0	J
1	1	0	1	STD
1	1	1	0	Optional
1	1	1	1	Optional

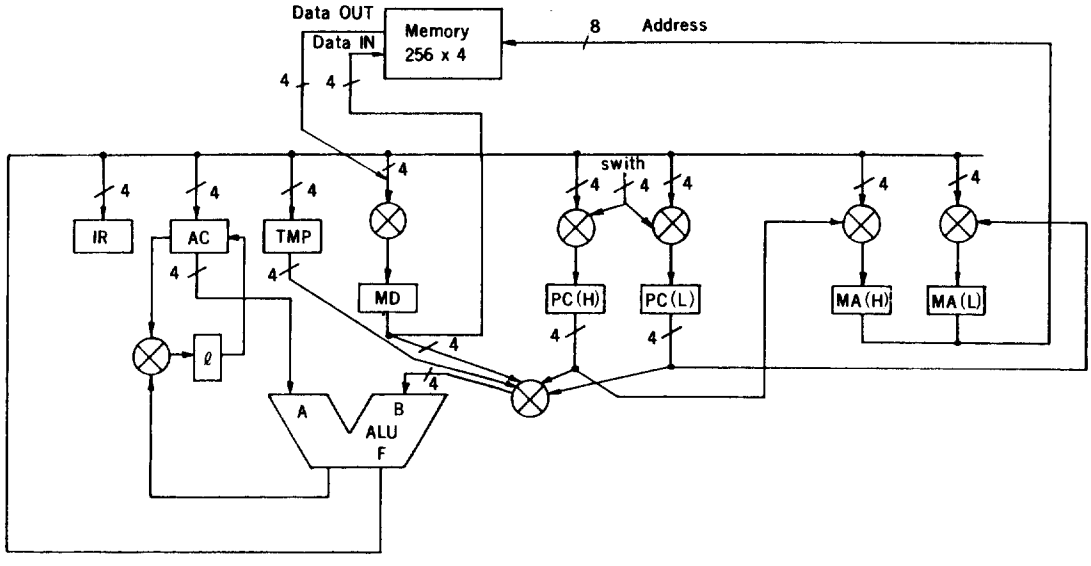


그림 1. 4bit Computer CPU Block Diagram

〔4〕, PC〔8〕, MA〔8〕, 1〔1〕

ALU

또 action은 표 1에 표시되어 있다. 기계조직에서 데이타 부분의 block diagram을 그림1과 같이 정할 수 있다.

2.1 컴퓨터의 Control Sequence

이 컴퓨터 설계에서 optional instruction을 뺀 나머지 INSTRUCTION을 위한 control sequence 를 표 2의 Binary Code 와 그림 1의 Block Diagram을 참작해서 다음과 같이 결정할 수 있다.

1. → ($\overline{\text{start}}$, start)/(1, 2) ;
2. MA ← PC ; ;
3. MD ← M(MA) ; PC ← PC + 1 ; ;
4. IR ← MD ; ;
5. → ($\overline{IR_3}$, IR_3)/(6, 23) ;
6. → (7, 8, 9, 16, 18, 20, 22, 22) ⊥ (IR_2, IR_1, IR_0) ;

(NOP) 7. → 2 ; ;

(SKZ) 8. → (V/AC, $\overline{V/AC}$)/(2, 10) ;

(SKL) 9. → (ℓ, $\overline{\ell}$)/(2, 10) ;

10. MA ← PC ; ;
11. MD ← M(MA) ; PC ← PC + 1 ; ;
12. → ($\overline{MD_3}$, MD_3)/(2, 13) ;
13. PC ← PC + 1 ; ;
14. PC ← PC + 1 ; ;
15. → 2 ; ;

(CL) 16. ℓ ← 0 ; ;

17. → 2 ; ;

(RAR) 18. ℓ, A ← right rotate (ℓ, A) ; ;

19. → 2 ; ;

(COM) 20. AC ← $\overline{AC}$; ;

21. → 2 ; ;

(HLT) 22. HALT (run ← 0) ; ;

23. MA ← PC ; ;
24. MD ← M(MA) ; PC ← PC + 1 ; ;
25. TMP ← MD ; MA ← PC ; ;
26. MD ← M(MA) ; PC ← PC + 1 ; ;
27. → ($IR_2\overline{IR_1} \cdot \overline{IR_0}$, $\overline{IR_2\,IR_1\,IR_0}$)/(28, 31) ;

(J) 28. PC(L) ← TMP ; ;

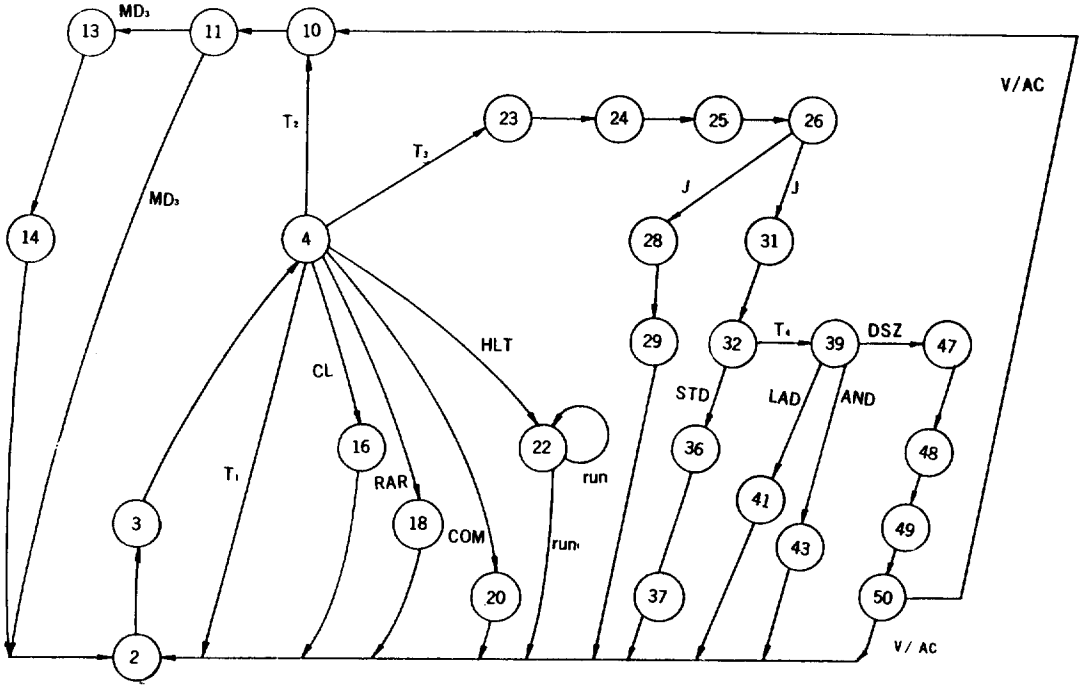
29. PC(H) ← MD ; ;
30. → 2 ; ;
31. MA(L) ← MD ; ;
32. MA(H) ← MD ; ;
33. → (IR_2, $\overline{IR_2}$)/(34, 39) ;
34. → ($\overline{IR_1}$, IR_1)/(36, 35) ;
35. HALT (Optional instruction not implemented) ; ;

(STD) 36. MD ← AC ; ;
37. M(MA) ← MD; ;
38. → 2 ; ;
39. MD ← M(MA) ; ;
40. (41, 43, 45, 47)⊥(IR_1, IR_0) ;
(LAD) 41. AC ← MD ; ;
42. → 2 ; ;
43. AC ← AC Λ MD ; ;
44. → 2 ; ;
(ADD) 45. AC ← AC + MD ; ;
46. → 2 ; ;
47. AC ← MD ; ;
(DSZ) 48. AC ← AC − 1 ; ;
49. MD ← AC ; ;
50. M(MA) ← MD ; ;
51. → 8 ; ;

여기서 microoperation 6은 $IR_2IR_1IR_0$의 이진수에 따라서 8 microoperation으로 branch하는 것이 결정된다는 것이다. 즉 $IR_2IR_1IR_0$ = 000이면 첫 microoperation 7로 가고 001이면 8, 010이면 9로 가라는 것이다. 여기서 $IR_2IR_1IR_0$ = 110과 111이면 다 microoperation 22로 가는데 110인 경우는 optional instruction인데 여기서 22 Halt로 임시 가도록 한 것이다. Microoperation 8에서 V/AC가 나오는데 이것은 Accumulator (AC)의 모든 bit을 OR하는 결과를 말하는 것이다. 즉 AC = (AC_3, AC_2, AC_1, AC_0)이면 V/AC = AC_3 V AC_2 V AC_1 V AC_0이다. 이 control sequence에 해당하는 state diagram은 그림 2와 같다.

2. 2 Control Sequencer

이 CPU의 control sequence를 시행하는 데 필요한 timing 펄스를 내는 control sequencer를 calculator 설계때 [1]과 같이 세가지 방법으로 설계할 수 있다. 한 state에 대해서 한 Flip-Flop 방법을 써서 설계한 회로가 그림 3 A와 3 B에 표시



T_1 = NOP V (SKZ Λ V/AC) V (SKL Λ ℓ)
T_2 = (SKZ Λ $\overline{V/AC}$) V (SKL Λ $\overline{\ell}$)
T_3 = LAD V AND V ADP V J V STD V (Optional Instruction 2 or 3)
T_4 = LAD V AND V ADD V DSZ

그림 2. CPU State Diagram

er(IR)로 형성된다고 볼 수 있다.

제어부분은 Hardwired 제어나 microprogramed 제어로 실현할 수 있다. 컴퓨터 INSTRUCTION의 FORMAT은 다음과 같이 표시할 수 있다.

OP Code	Operand 주 소	Operand 주 소	결과 주소	다음 Instruction 주소

OP(Operation) Code는 제일 기본적 부분이다. OP Code가 이 INSTRUCTION의 operation을 표시한다. 이 code가 합하나 빼나 곱하나 나누나 shift나 complement 같은 것을 표시한다. 이 operation을 시행하는데 보통 데이타가 operand으로 나타난다. 두 operand의 memory 주소를 instruction이 포함할 때가 있으며 또 결과를 어디로 저장하는 주소와 이 INSTRUCTION의 시행이 끝나면 다음 INSTRUCTION의 주소를 지시한다.

그러나 이렇게 긴 INSTRUCTION FORMAT를 쓰는 것이 비능률적이므로 Program Counter (PC)와 Accumulator register를 써서 format을 짧게 한다. PC는 항상 memory에 있는 다음 INSTRUCTION의 주소를 지시한다. 따라서 PC가 있으므로 다음 INSTRUCTION의 주소를 format에서 없앨 수 있다. 또 Accumulator register에 항상 operand 하나를 넣고 INSTRUCTION 결과를 이 register에 집어 넣으므로서 operand 주소 하나와 결과주소를 생략할 수 있다. 그러므로서 다음과 같은 format을 갖게 된다.

OP Code	Operand 주소나 또는 Operand

컴퓨터 INSTRUCTION이나 데이타를 표시하는데 몇개의 bit를 기본으로 삼아 WORD라고 부른다. 예로 INTEL 8080이나 Zilog Z-80나 motorola 6800은 8 bits이 WORD로 되어 byte라고 부른다. 즉 3 bytes라 하면 8 bits짜리 WORD가 셋이라는 것이다. 어느 컴퓨터는 4 bits이나 16 bits이 기본적 WORD로 쓴다.

컴퓨터의 register는 두 가지로, operation을 위한 것과 저장용으로 구분할 수 있다. Operation 용 register는 flip-flop와 gate로 되어 저장뿐만 아니라 데이타 처리까지 할 수 있다. 저장용 register는 memory 속에 있어 이진수 데이타를 저장역할만 한다.

Memory는 word를 기본으로 해서 데이타를 저축해 둔다. Memory는 두가지 Random Access Memory (RAM)과 Read Only Memory (ROM) 으로 구별할 수 있다. RAM은 저장용 register로 형성되어 있으며 이에 따른 논리회로와 함께 register에 다(이진수) 데이타를 기억(write)시키거나, register에 있는 데이타를 읽을 수(read) 있다. RAM은 WORD를 한 단위로서 memory에 기억해 둔다.

RAM과 외부와의 통신은 제어선들과 주소선들과 데이타선들을 통해서 한다. ROM은 읽는 operation만 가능하며 ROM에다 컴퓨터 operation중에는 데이타를 저장하는 기억 operation을 못한다. ROM은 제작시에 저장될 WORD를 기억하게 하며, 이러므로서 컴퓨터가 operation하는 동안은 저장될 WORD를 변경 못한다.

기본적 컴퓨터 구조에 CPU와 memory 외에 입력과 출력(I/O) interface가 있다. 컴퓨터는 이 interface를 통해서 I/O operation을 시행한다. 즉, I/O interface를 통해서 컴퓨터는 외부의 세계와 통신을 한다.

기본적 컴퓨터의 구조와 설계를 간단한 소형 CPU (microprocessor)를 설계함으로서 공부하기로 한다. 소형 CPU도 체계적 설계 방법인 구조정의, 기계조직, 제어부분, 데이타 부분으로 나누어 설계할 수 있다.

2. 소형 CPU 설계문제

보통 소형 CPU는 instruction수가 50이 넘는다. 그러나 우리가 설계하려는 소형 CPU는 독자들이 비교적 단시간에 완성하기 위해서 표 1과 같이 16 instruction를 택한다. 여기서 register reference instruction은 시행에 쓰이는 operand가 register에만 있는 instruction이다. Memory reference instruction은 시행하는 operand가 memory에 있는 instruction이다.

이 컴퓨터 설계에 있어서 아래와 같은 specification을 주어졌다.

1. WORD 길이는 4 bits
2. 주 memory는 256 words
3. INSTRUCTION 속에 input나 output instruction을 넣을 것.
4. CPU는 TTL Chip으로서 실현할 것.

Instruction의 Addressing mode가 operand이 어디있냐 하는 것을 결정한다. Instruction 에

되여 있다. Control sequencer 회로는 그림 3A와 3B에 있는 Ⓐ점을 연결해주고 또 Ⓑ점을 연결하므로 완전한 회로가 형성된다. 여기에 D Flip-Flop을 썼으며 표시는 되어있지 않지만 모든 flip-flop에 control clock를 넣어 준다.

그림 2에 있는 이 CPU의 state diagram를 써서 설계하면 flip-flop이 다섯이면 되므로 절약이 된다. 또 microprogramming 방법으로 설계할 수 있다.

각 Register와 ALU의 mode control을 위해서 control sequencer에서 내는 timing 펄스를 써서 적합하게 각 소자의 mode terminal에 줄 수 있다. ALU와 Memory Address register (MA)의 mode control table를 표 3과 4같이 구할 수 있다. 또 다른 register 들의 mode control table도 똑같은 방법으로 구할 수 있다.

Data Flow 부분의 설계는 각 register와 ALU의 Data Path Connection table를 그림 1의 Block Diagram와 Control Sequence를 써서 구할 수 있다. 또 Data Path Wiring도 Data Path Connection Table과 Control Sequencer의 timing 펄스를 써서 구할 수 있다. 이것으로서 이 CPU 설계는 끝난다.

다음은 Stack과 Input/Output operation에 대해서 설명한다.

3. 결 론

이 강좌에서 소형 CPU를 체계적 방법을 응용해서 설계하는 것을 공부했다. 예로서 설계한 CPU는 16 Instruction이지만 이 Instruction에는 세가지 기본적 Instruction형이 다 들어가 있다. 이 세가지 형은 첫째, 메이타를 움직이는 MOVE Instruction으로 이 CPU에서는 LAD와 STD이다. 둘째, 연산(Arithmetic and Logic) operation으로 CL, RAR, COM, AND, ADD이다. 세째, Branch Instruction으로 SKZ, SKL, DSZ, J이다. 이 CPU의 Instruction set에 필요한 다른 Instruction을 가할 경우도 이 체계적 설계방법을 써서 설계할 수 있다.

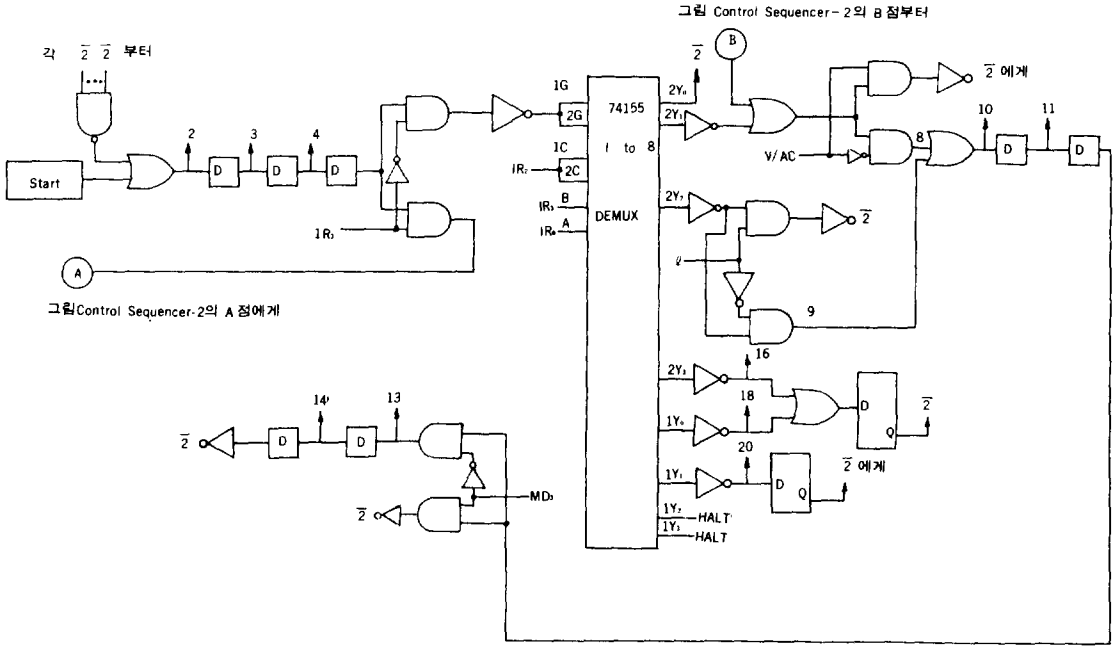


그림 3A. 컴퓨터 Control Sequencer — 1

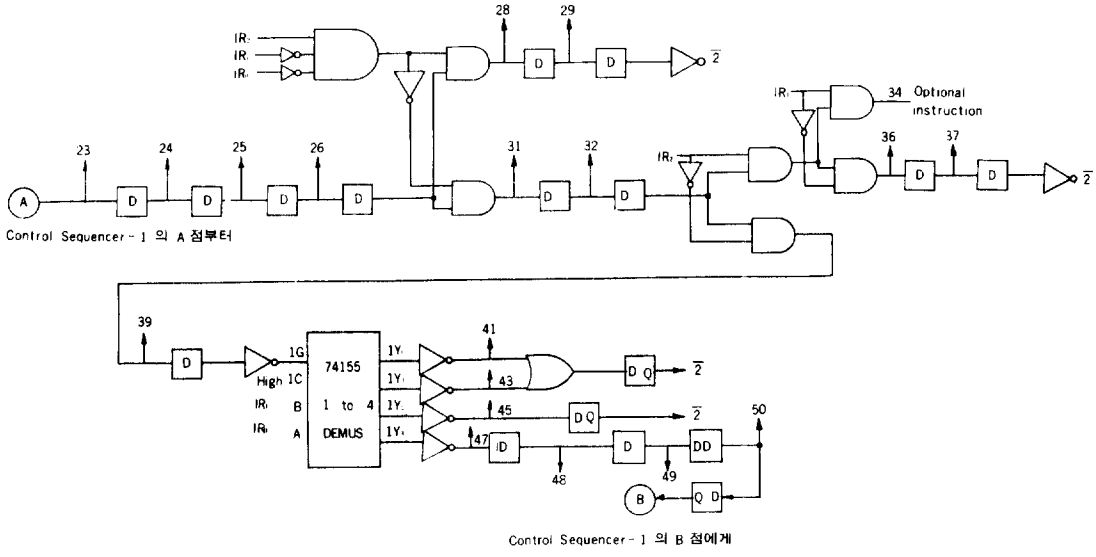


그림 3 B. 컴퓨터 Control Sequencer – 2

표 3. ALU의 Mode Control Table– C1

Micro-operation	74181 ALU Function		Mode Terminals					
			M	$\overline{C_N}$	S_3	S_2	S_1	S_0
4	IR ← MD	(F = B)	H	X	H	L	H	L
20	AC ← $\overline{AC}$	(F = $\overline{A}$)	H	X	L	L	L	L
25	TMP ← MD	(F = B)	H	X	H	L	H	L
28	PC(L) ← TMP	(F = B)	H	X	H	L	H	L
29	PC(H) ← MD	(F = B)	H	X	H	L	H	L
31	α^4/ MA ← TMP	(F = B)	H	X	H	L	H	L
32	ω^4/ MA ← MD	(F = B)	H	X	H	L	H	L
36	MD ← AC	(F = A)	H	X	H	H	H	H
41	AC ← MD	(F = B)	H	X	H	L	H	L
43	AC ← AC ∧ MD	(F = A+B)	H	X	H	L	H	H
45	AC ← ADD(AC, MD)	(F = A, PLUS B)	L	H	H	L	L	H
47	AC ← MD	(F = B)	H	X	H	L	H	L
48	AC ← SUB(A, 1)	(F = A MINUS 1)	L	H	H	H	H	H
49	MD ← AC	(F = A)	H	X	H	H	H	H

M = $\overline{45}(\overline{48})$
C_N = H
S_3 = $\overline{20}$
S_2 = 36 + 48 + 49
S_1 = $\overline{20}(\overline{45})$
S_0 = 36 + 45 + 48 + 49 + 43

표 4. Memory Address Register의 Mode Control Table

	MA(L)-74194 Function	Mode Terminals		
		CLEAR	S_1	S_0
2	MA ← PC	H	H	H
10	MA ← PC Clear = H	H	H	H
23	MA ← PC	H	H	H
25	MA ← PC $S_1=S_0=2+10+23+25+31$	H	H	H
31	MA(L) ← TMP	H	H	H
All Others	no operation	H	L	L

	MA(H)-74194			
2	MA ← PC Clear = H	H	H	H
10	MA ← PC $S_1=S_0=2+10+23+25+32$	H	H	H
23	MA ← PC	H	H	H
25	MA ← PC	H	H	H
32	MA(H) ← MD	H	H	H
All Others	no operation	H	L	L

참 고 문 헌

[1] 김명환 ; "디지탈 시스템과 마이크로 프로세서 설계 : Ⅱ, 체계적 설계 - 2 ", 전기학회지, 제 31 권, 제 8 호 1982년 8월

[2] Lecture Note, EE 675 Computer Structure Cornell University.

[3] F. Hill and G. Peterson; "DIGITAL SYSTEM: Hardware Organization and Design", 2nd Ed., John Wiley & Sons, Inc., 1978, pp. 121-402.

[4] M. Nano; "Digital Logic and Computer Design", Prentice-Hall, Inc., Englewood Clifts, N.J., 1979, Chapter 8.

[5] Mano, M.M.; "Computer System Architecture" Prentice Hall, Inc., Englewood Clifts, N.J., 1976.

[6] Kline, M.M.; "Digital Computer Design", Prentice-Hall, Inc., Englewood Clifts, N.J., 1977.

[7] Chu, Y.; "Computer Organization and Microprogramming", Prentice-Hall Inc., Englewood Clifts, N.J., 1972.