ปีการศึกษา 2532

การออกแบบสร้าง PROCESSER โดยใช้ BIT SLICE

(A DESIGN PROCESSER BY BIT SLICE)

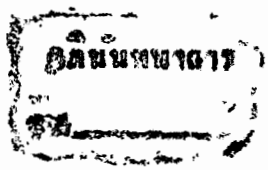
โดย

สุวัจ เวชพันธุ์

อาจารย์ที่ปรึกษา

ดร. บุญวัฒน์ อัตชู

ผศ. นิกร สุขุตมดันติ



ปริญญานิพนธ์ปีการศึกษา ...2532.....

ภาควิชา ...เทคนิคอุตสาหกรรม.....

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง...การออกแบบ PROCESSER โดยใช้ BIT - SLICE.....

ผู้จัดทำ ...สุวัจ เวชพันธุ์.....

..... อาจารย์ที่ปรึกษา

(.....)

..... อาจารย์ที่ปรึกษา

(.....)

..... อาจารย์ที่ปรึกษา

(.....)

การออกแบบสร้าง PROCESSER โดยใช้ BIT-SLICE

(A DESIGN PROCESSER BY BIT-SLICE)

สุวัจ เวชพันธุ์

ดร. บุญวัฒน์ อัตชู อ.ที่ปรึกษา

ผศ. นิกร สุขุตมตันติ อ.ที่ปรึกษา

ปีการศึกษา 2532

บทคัดย่อ.

โครงงานนี้เป็นการออกแบบสร้าง PROCESSER โดยใช้ BIT-SLICE (IC.BIT-SLICE 2901) มีวัตถุประสงค์ในการทำเพื่อศึกษาความสัมพันธ์ระหว่างชุดคำสั่ง (MACRO INSTRUCTION) และโครงสร้างสถาปัตยกรรมภายในตัว PROCESSER ซึ่งเป็น PROCESSER แบบที่มี MICRO PROGRAM ในการ CONTROL SEQUENCE ของการ EXECUTE คำสั่ง PROCESSER แบบนี้จะมีความเร็ว(คำสั่งที่ถูก EXECUTE ใน 1 วินาที) สูงมากเพราะ IC BIT - SLICE เป็น IC. ตระกูล BIPOLAR และ MICROPROGRAM เองก็สามารถเขียนให้ชุดคำสั่งมีความอ่อนตัวได้.

ในโครงงานนี้จะได้กล่าวถึงขั้นตอนของการออกแบบ PROCESSER และชุดคำสั่ง (MACRO INSTRUCTION),กล่าวถึงการทำงานของ PROCESSER ที่ได้ออกแบบไว้ดัง BLOCK DIAGRAM ที่แนบมา,และกล่าวถึงการเขียน MICRO PROGRAM ในการ CONTROL SEQENCE ของINSTRUCTION (MACRO INSTRUCTION)...

สารบัญ ...

PART 1.

INTRODUCTION..

==========

PART 2.

HARD WARE..

==========

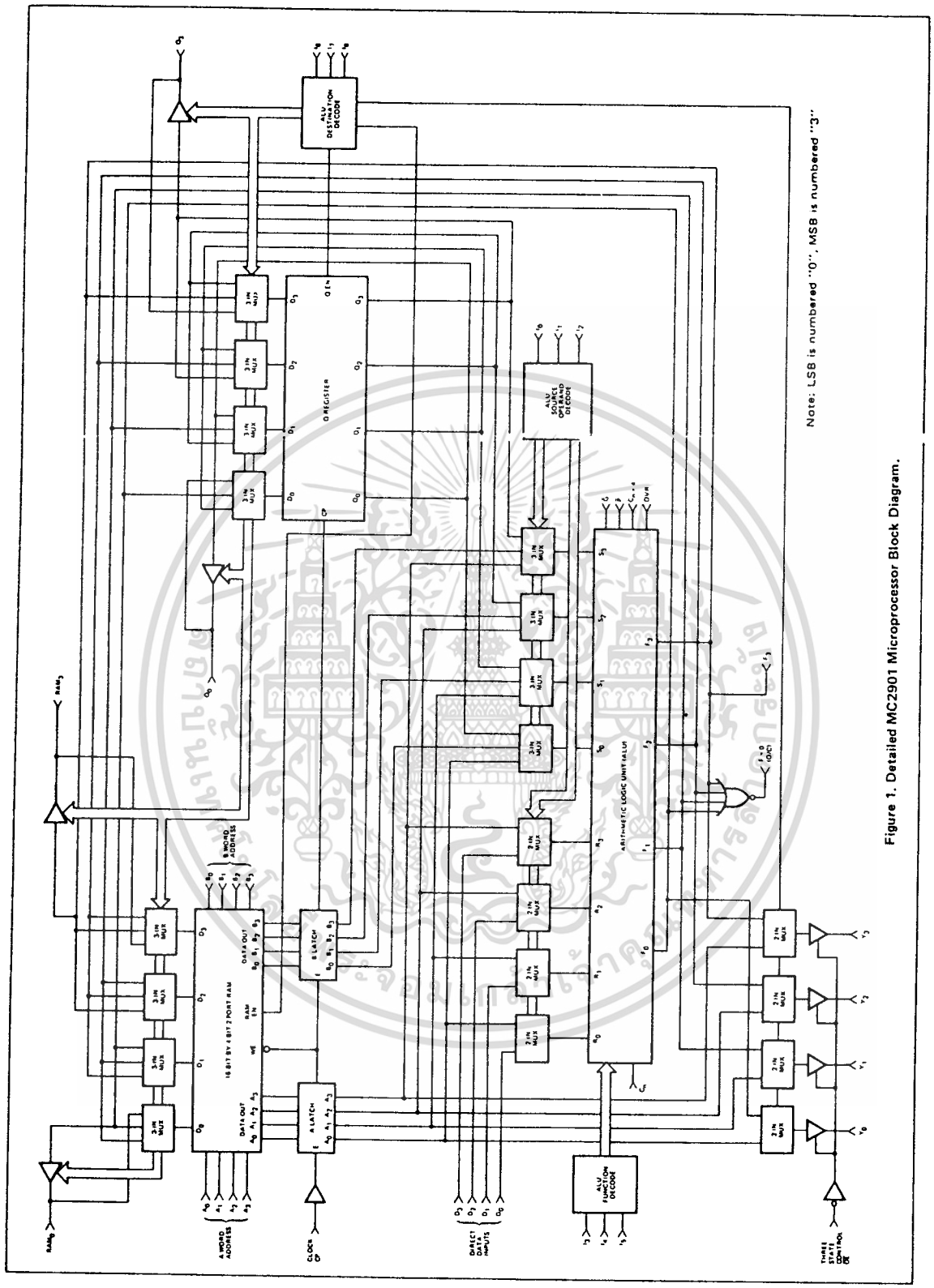


Figure 1. Detailed MC2901 Microprocessor Block Diagram.

MICRO CODE				ALU SOURCE OPERANDS	
I_2	I_1	I_0	Octal Code	R	S
L	L	L	0	A	Q
L	L	H	1	A	B
L	H	L	2	0	Q
L	H	H	3	0	B
H	L	L	4	0	A
H	L	H	5	D	A
H	H	L	6	D	Q
H	H	H	7	D	0

Figure 2. ALU Source Operand Control.

MICRO CODE				ALU Function	Symbol
I_5	I_4	I_3	Octal Code		
L	L	L	0	R Plus S	R + S
L	L	H	1	S Minus R	S − R
L	H	L	2	R Minus S	R − S
L	H	H	3	R OR S	R ∨ S
H	L	L	4	R AND S	R ∧ S
H	L	H	5	$\overline{R}$ AND S	$\overline{R} \wedge S$
H	H	L	6	R EX-OR S	R ∀ S
H	H	H	7	R EX-NOR S	$\overline{R \forall S}$

Figure 3. ALU Function Control.

MICRO CODE				RAM FUNCTION		Q-REG. FUNCTION		Y OUTPUT	RAM SHIFTER		Q SHIFTER	
I_8	I_7	I_6	Octal Code	Shift	Load	Shift	Load		RAM_0	RAM_3	Q_0	Q_3
L	L	L	0	X	NONE	NONE	F → Q	F	X	X	X	X
L	L	H	1	X	NONE	X	NONE	F	X	X	X	X
L	H	L	2	NONE	F → B	X	NONE	A	X	X	X	X
L	H	H	3	NONE	F → B	X	NONE	F	X	X	X	X
H	L	L	4	DOWN	F/2 → B	DOWN	Q/2 → Q	F	F_0	IN_3	Q_0	IN_3
H	L	H	5	DOWN	F/2 → B	X	NONE	F	F_0	IN_3	Q_0	X
H	H	L	6	UP	2F → B	UP	2Q → Q	F	IN_0	F_3	IN_0	Q_3
H	H	H	7	UP	2F → B	X	NONE	F	IN_0	F_3	X	Q_3

X= Don't care. Electrically, the shift pin is a TTL input internally connected to a three-state output which is in the high-impedance state.
B = Register Addressed by B inputs.
Up is toward MSB, Down is toward LSB.

Figure 4. ALU Destination Control.

OCTAL I_{210}	ALU Source / ALU Function (I_{543})	0	1	2	3	4	5	6	7
	OCTAL	A, Q	A, B	0, Q	0, B	0, A	D, A	D, Q	D, 0
0	C_n = L R Plus S	A+Q	A+B	Q	B	A	D+A	D+Q	D
	C_n = H	A+Q+1	A+B+1	Q+1	B+1	A+1	D+A+1	D+Q+1	D+1
1	C_n = L S Minus R	Q−A−1	B−A−1	Q−1	B−1	A−1	A−D−1	Q−D−1	−D−1
	C_n = H	Q−A	B−A	Q	B	A	A−D	Q−D	−D
2	C_n = L R Minus S	A−Q−1	A−B−1	−Q−1	−B−1	−A−1	D−A−1	D−Q−1	D−1
	C_n = H	A−Q	A−B	−Q	−B	−A	D−A	D−Q	D
3	R OR S	A∨Q	A∨B	Q	B	A	D∨A	D∨Q	D
4	R AND S	A∧Q	A∧B	0	0	0	D∧A	D∧Q	0
5	$\overline{R}$ AND S	$\overline{A} \wedge Q$	$\overline{A} \wedge B$	0	B	A	$\overline{D} \wedge A$	$\overline{D} \wedge Q$	0
6	R EX-OR S	A∀Q	A∀B	Q	B	A	D∀A	D∀Q	D
7	R EX-NOR S	$\overline{A \forall Q}$	$\overline{A \forall B}$	$\overline{Q}$	$\overline{B}$	$\overline{A}$	$\overline{D \forall A}$	$\overline{D \forall Q}$	$\overline{D}$

+ = Plus; − = Minus; ∨ = OR; ∧ = AND; ∀ = EX-OR

Figure 5. Source Operand and ALU Function Matrix.

ALU เป็น I6,I7,I8 (ดูจากตาราง ALU DESTINATION CONTROL) OUTPUT ของ ALU จะมีลักษณะเป็น TRI - STATE โดยมีขา OE ในการ CONTROL. MICROCODE I6,I7,I8 จะใช้ในการ CONTROL OUTPUT ว่าจะเป็น OUTPUT จาก A PORT หรือจาก ALU OUTPUT [F] ที่MUX. แบบ 3 INPUT จะมีผลต่อ OUTPUT ของ ALU ในเรื่องการ SHIFT UP/DOWN หรือไม่ SHIFT ซึ่งการ SHIFT นี้จะมี RAM0 และ RAM3 เป็นตัวกำหนด ซึ่งทั้ง 2 ตัวนี้จะเป็น PORT ที่ประกอบด้วย BUFEER DRIVER แบบ TRI STATE และมี INPUT ไปยัง MUX. ดังนั้นใน MODE SHIFT UP. RAM3 BUFFER จะถูก ENABLE และ RAM0 MUX. จะถูก ENABLE (BUFFER RAM0 ไม่ทำงาน) ทำนองเดียวกันใน MODE SHIFT DOWN ที่ RAM0 BUFFER และ RAM3 INPUT จะถูก ENABLE. ในกรณีไม่ SHIFT PORT ทั้งสองจะอยู่ในสถานะ HIGH IMPEDANCE..

การขยายจำนวน BIT ของ 2901..

IC.2901 เป็น IC. 4 BIT SLICE ซึ่งสามารถนำมาต่อกันได้ในรูป 12,16,24 BIT ได้ การนำมาต่อขยายอาจต่อได้ในลักษณะของ RIPPLE CARRY อย่างเช่นในการทดลองครั้งนี้ หรือในกรณีต่อ 2601 หลายตัวก็จะใช้การต่อแบบ CARRY LOOKAH ED และ CARRY OUT ที่ออกจาก MSB ของ 2901 (ขา CN+4)จะถูกใช้เป็นCARRY FLAG ของ 2901 ในการเชื่อมต่อ 2901 แบบ RIPPLE CARRY นั้นขา CN+4 ของ 2901 แต่ละตัวจะต่ออยู่กับขา CN ของ 2901 ตัวต่อไปที่มีนัยสำคัญสูงกว่า..

ADDRESS	Y0 -Y7
0	0000 0001
2	0000 1001
4	0000 1000
6	0010 0000

ซึ่งค่าในOUTPUT ที่อ่านได้ก็คือค่าในREGISTER [B] นั่นเอง.

3) กำหนด MICROCODE เป็น 327 เพื่อทำการ DECREMENT [DEC] ค่า DIRECT DATA [D] แล้วเก็บใน REGISTER [B] ทดลองป้อนค่าตามADDRESS ที่กำหนดและเช่นเดียวกับการทดลอง INC ค่า Y0 -Y7 จะแสดงค่าที่ถูก DEC มาให้เลย

4) ทดลอง READ ค่าใน REGISTER [B] ออกดูโดยกำหนด MICRO CODE เป็น 103 ให้ CN เป็น 0 จะเห็นว่าค่าที่อ่านได้นั้นจะเป็นค่าที่ถูก DEC แล้วเช่นเดียวกับการทดลองที่ 2

สรุปการทดลองที่ 2..

ในขณะทำการ READ ค่าใน REGISTER [B] ออกดูถ้าอ่านผ่าน ADDRESS [B] แล้วให้ CN เป็น 1 OUTPUT Y0 -Y7 ก็จะถูกเพิ่มอีก 1 เช่นจากการทดลองที่ 2ที่ADDRESS 0 จะถูกเพิ่มจาก 0000 0001 เป็น 0000 0010 ซึ่งก็คือการ INC ค่าใน [B]..

แต่ในการ DEC ค่าใน REGISTER [B] จะต้องเปลี่ยน MICRO CODE เป็น 317 แล้วทำเหมือนการทดลอง INC..

การทดลองที่ 3.. ทดลอง FUNCTION ของ IC.BIT-SLICE..

[3.1] ทดลองการบวกกันระหว่าง DIRECT DATA [D]กับ DATA ใน REGISTER [A]

ลำดับการทดลอง..

1. กำหนด ตัวตั้งและตัวบวกโดยให้ MICRO CODE เป็น 307 แล้วป้อนค่าตาม ADDRESS ที่กำหนด

ADDRESS	DATA
0000	1111 0101 [F5]
0001	0000 1010 [0A]

* ป้อนค่านี้แล้วไม่ต้องใส่ CLOCK ให้ SET MICRO CODE ได้เลย

2. กำหนด MICRO CODE เป็น 305 เพื่อนำตัวตั้งและตัวบวกจาก 1.มาบวกกันแล้วให้ผลลัพธ์เก็บใน ADDRESS 0010 ซึ่งผลบวกจะเป็น 1111 1111 [FF].

[3.2] ทดลองลบกันระหว่าง DIRECT DATA กับDATA ใน REG [A]

ลำดับการทดลอง..

1. กำหนดตัวตั้งและตัวลบโดยให้ MICROCODE เป็น 307 แล้วป้อนค่าตาม ADDRESS ที่กำหนด..

ADDRESS	DATA
0000	0000 1010 [0A]
0001	1111 0101 [F5]

2. กำหนด MICROCODE เป็น 325 โดยให้ CN เป็น 1 เพื่อทำการลบกันระหว่าง DIRECT DATA กับ DATA ใน REG.[A] โดยค่า 1111 0101 จะเก็บใน [D] และ 0000 1010 จะอยู่ใน REG.[A] ผลลัพธ์จะเก็บใน ADDRESS 0010 ซึ่งมีผลลัพธ์เป็น 1110 1011 และที่ CARRY FLAG จะถูก SET เป็น 1 และที่ OVERFLOW FLAG จะถูก SET เป็น 0 (ไม่คิดเครื่องหมาย) และหากให้ ADDRESS 0000 เป็น F5 และที่ ADDRESS 0001 เป็น 0A โดยให้ MICROCODE เป็น 325 และ CN ยังเป็น 1 แลวผลลัพธ์ที่ได้จะเป็นค่า 2'COMPLEMENT ของ 11101011 ซึ่งก็คือค่าลบของ 1110 1011 [EB] การที่จะกลับเป็นค่าบวกนั้นทำโดยเปลี่ยน MICROCODE เป็น 315 ซึ่งก็หมายถึงการนำค่าใน REG.[A] ลบกับค่า DIRECT DATA และที่ OVR. FLAG ก็ยังคงเป็น 0..

[3.3] ทดลอง AND ระหว่าง DIRECT DATA กับ DATA ใน REG.[A]

ลำดับการทดลอง..

1. กำหนด DATA ที่จะ AND โดยให้ MICROCODE เป็น 307 แล้วป้อนค่าตาม ADDRESS ที่กำหนด

ADDRESS	DATA	FLAG
0000	10100101	0100
0001	10010110	0100

2. กำหนด MICROCODE เป็น 345 (CN ไม่มีผลในกรณีนี้ เพื่อทำการ AND กันแล้วให้ผลลัพธ์อยู่ใน ADDRESS 0010 ซึ่งผลลัพธ์ที่ได้จะเป็น 1000 0100 และที่ CARRY FLAG , OVERFLOW จะถูกSET เป็น 1..

[3.4] ทดลอง EX-OR ระหว่าง DIRECT DATA กับค่าใน REG. [A]

ลำดับการทดลอง..

1. กำหนด DATA ที่จะ EX-OR ลงใน [D]และREG.[A] โดยให้ MICROCODE เป็น 307 ป้อนค่าตาม ADDRESS ที่กำหนด

ADDRESS	DATA
0000	1100 0011
0001	1010 0101

2. กำหนด MICROCODE เป็น 365 (CN ก็ยังไม่มีผล)
เพื่อทำการ EX-OR ให้ผลลัพธ์อยู่ในADD. 0010

ADDRESS	DATA	FLAG
0010	0110 0110	0000

3. ทำการทดลองใหม่โดยเปลี่ยนค่าใน [D]และ [A]ให้
MICROCODE เป็น 307 เช่นเดิม

ADDRESS	DATA
0000	1001 1100
0001	0011 1100

4. ทำการ EX-OR กันอีกครั้งโดยให้ MICROCODE เป็น 365 ผลลัพธ์เก็บใน ADDRESS 0010

ADDRESS	DATA	FLAG
0010	1010 0000	0101

* สังเกตุที่ OVR.FLAG จะถูก set ให้เป็น 1 ซึ่งในการทดลองที่ 2 จะเป็น 0..

[3.5] ทดลองการ EX-OR ระหว่าง DATA ใน REG.[A]กับ [B]

ลำดับการทดลอง

1. กำหนดค่าที่จะ EX-OR ลงใน [A]และ [B] โดยให้ MICROCODE เป็น 307 แล้วป้อนค่าตาม ADDRESS ที่กำหนด

ADDRESS	DATA
0000	1100 0011
0001	1010 0101

2. กำหนด MICROCODE เป็น 361 เพื่อทำการ EX-OR กัน แต่ผลลัพธ์ก็ยังคงอยู่ใน ADDRESS 0001 ไม่สามารถเก็บใน ADDRESS ที่ 0010 ได้เหมือนการทดลองที่แล้วมา. ซึ่งผลลัพธ์ของการ EX-OR ก็จะเป็น 0110 0110 ซึ่งตรงกับการทดลองที่[3.6] ข้อ2 แต่จะต่างกันที่ FLAG ในกรณีนี้ FLAG CN+4 จะถูก SET เป็น 1.

3. ทดลองใหม่อีกครั้งโดยเปลี่ยนค่าใน [A] และ[B]ให้ MICROCODE เป็น 307 เช่นเดิม

ADDRESS	DATA
0000	0011 1100
0001	1000 0001

4. ทำการ EX-OR กันอีกครั้งให้ MICROCODE เป็น 361 ผลลัพธ์ก็ยังอยู่ใน ADDRESS 0001

ADDRESS	DATA	FLAG
0001	1011 1101	0101

* สังเกตุที่OVR.FLAG จะถูกSET เป็น 1 (ในข้อ 2เป็น 0)

สรุปการทดลองที่3..

-ในการทดลองที่ 3.2 การลบกันระหว่าง DIRECT DATA กับค่าใน REG.[A] นั้นค่าของตัวตั้งจะเก็บไว้ใน ADDRESSที่สูงกว่าส่วน ADDRESSที่ต่ำกว่านั้น 1ตำแหน่งจะไว้เก็บตัวลบ และในการทดลองนี้ไม่มี OVER FLOWแม้ว่าจะทดลองกลับค่าตัวตั้งกับตัวลบก็ตามค่าที่ได้จะเป็นค่า 2' COMPLEMENT ของค่าเดิม

- ในการทดลองที่ 3.4 การ EX-OR ระหว่าง DIRECT DATA กับค่าใน REG.[A] การทดลองนี้ทำ 2ครั้งให้ค่า [A]และ[D] ต่างกันจะพบว่าที่ OVR.FLAG จะต่างกัน.

การทดลองที่ 4. การ SHIFT-UP/DOWN ของ BIT-SLICE

[4.1] ทดลองการSHIFT-UP โดยให้ RAM3 เป็น 1 และ RAM0 เป็น 0

ลำดับการทดลอง

1. กำหนด MICROCODE เป็น 707 เพื่อทำการSHIFT UPค่าจากการบวกกันระหว่าง DIRECT DATA กับค่า

หลักการทำงานของ MICROPROGRAM SEQUENCER

พิจารณา BLOCK DIAGRAM จะเห็นว่า MICROPROGRAM SEQUENCER เป็น IC ที่ประกอบด้วย MULTIPLEXER แบบ 4 INPUT สำหรับเลือกค่าจาก DIRECT IN PUT,ADDRESS REGISTER,MICROPROGRAM COUNTER หรือ FILE [STACK] เพื่อเป็น ADDRESS ของ MICROPROGRAM MEMORY. MUX. ที่กล่าวถึงนี้จะถูก CONTROL โดยขา S0 และ S1 ใน ADDRESS REGISTER จะประกอบด้วย FLIP-FLOP แบบ [D] 4 ชุดที่ใช้ CLOCK ร่วมกัน. และใน MICROPROGRAM COUNTER จะประกอบด้วย INCREMENTER ขนาด 4 BIT ซึ่งในนั้นจะมี CARRY IN [CN] และ CARRY OUT [CN+4] เพื่อใช้สำหรับต่อเพิ่มจำนวน WORD ให้ยาวขึ้น IC.2909 มี OUTPUTเป็นแบบ TRI STATE..

MC2909

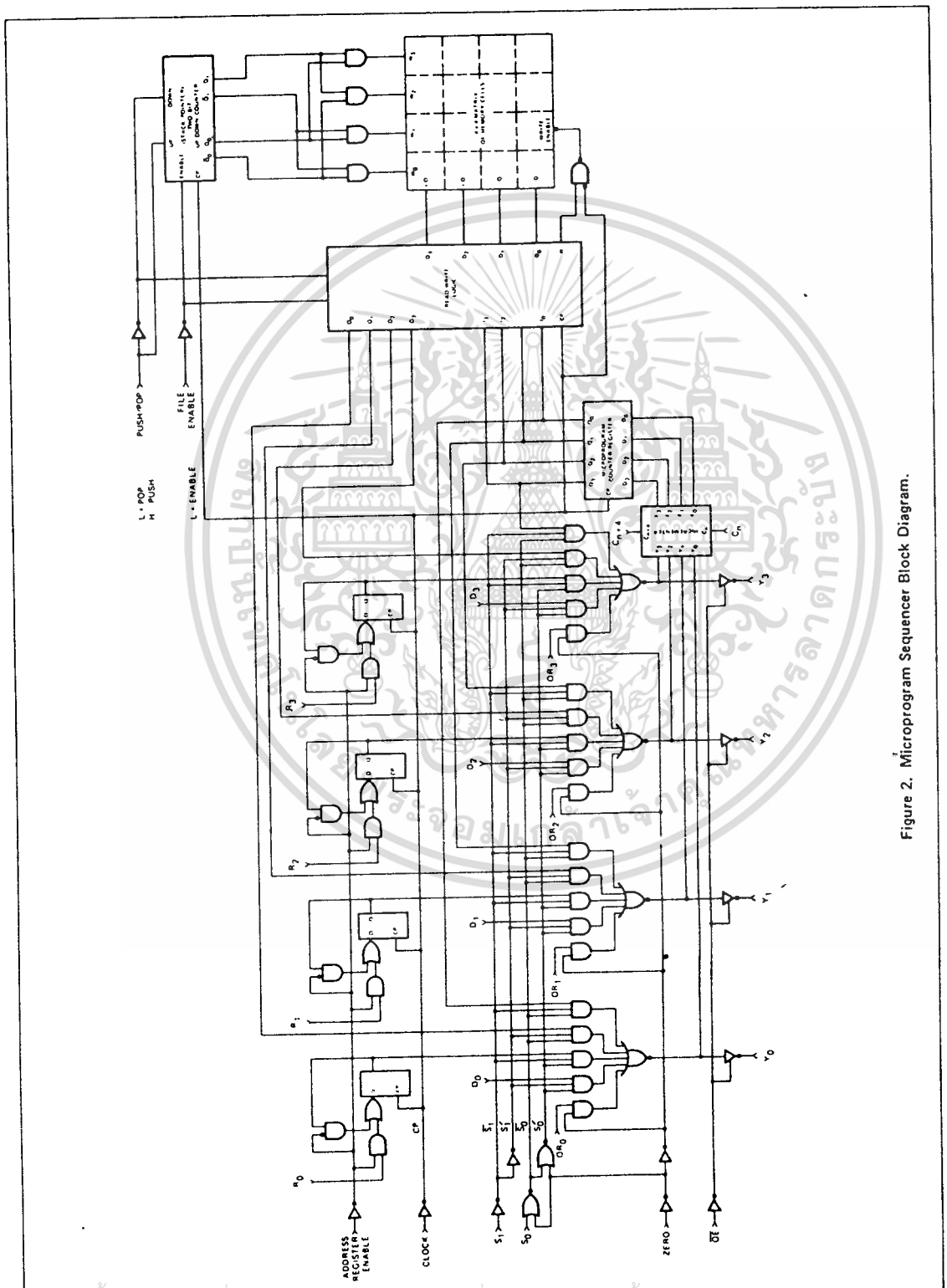


Figure 2. Microprogram Sequencer Block Diagram.

DEFINITION OF TERMS

A set of symbols is used in this data sheet to represent various internal and external registers and signals used with the MC2909. Since its principle application is as a controller for a microprogram store, it is necessary to define some signals associated with the microcode itself. Figure 3 illustrates the basic interconnection of MC2909, memory, and microinstruction register. The definitions here apply to this architecture.

Inputs to MC2909

S_1, S_0	Control lines for address source selection
$\overline{FE}$, PUP	Control lines for push/pop stack
$\overline{RE}$	Enable line for internal address register
OR_i	Logic OR inputs on each address output line
$\overline{ZERO}$	Logic AND input on the output lines
$\overline{OE}$	Output Enable. When $\overline{OE}$ is HIGH, the Y outputs are OFF (high impedance)
C_n	Carry-in to the incrementer
R_i	Inputs to the internal address register
D_i	Direct inputs to the multiplexer
CP	Clock input to the AR and μPC register and Push-Pop stack

Outputs from the MC2909

Y_i	Address outputs from MC2909. (Address inputs to control memory.)
C_{n+4}	Carry out from the incrementer

Internal Signals

μPC	Contents of the microprogram counter
AR	Contents of the address register
STK0-STK3	Contents of the push/pop stack. By definition, the word in the four-by-four file, addressed by the stack pointer is STK0. Conceptually data is pushed into the stack at STK0; a subsequent push moves STK0 to STK1; a pop implies STK3 → STK2 → STK1 → STK0. Physically, only the stack pointer changes when a push or pop is performed. The data does not move. I/O occurs at STK0.
SP	Contents of the stack pointer

External to the MC2909

A	Address to the control memory
I(A)	Instruction in control memory at address A
μWR	Contents of the microword register (at output of control memory). The microword register contains the instruction currently being executed.
T_n	Time period (cycle) n

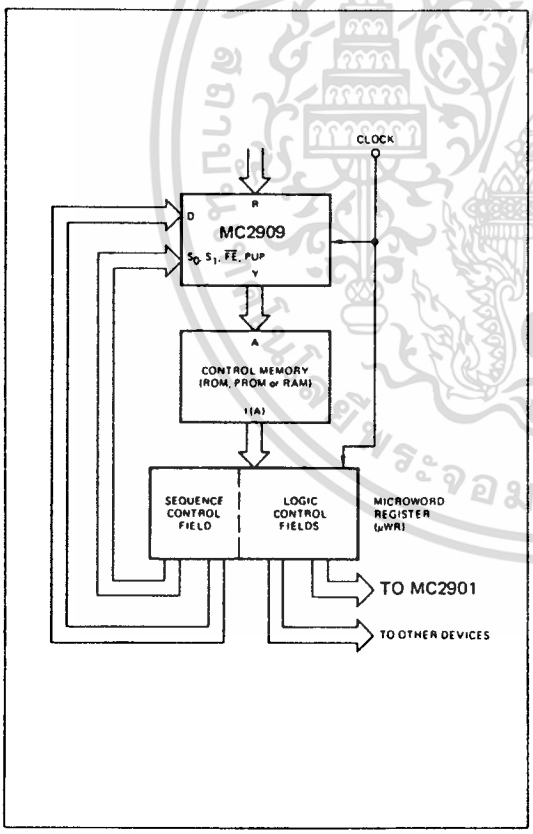


Figure 3. Microprogram Sequencer Control.

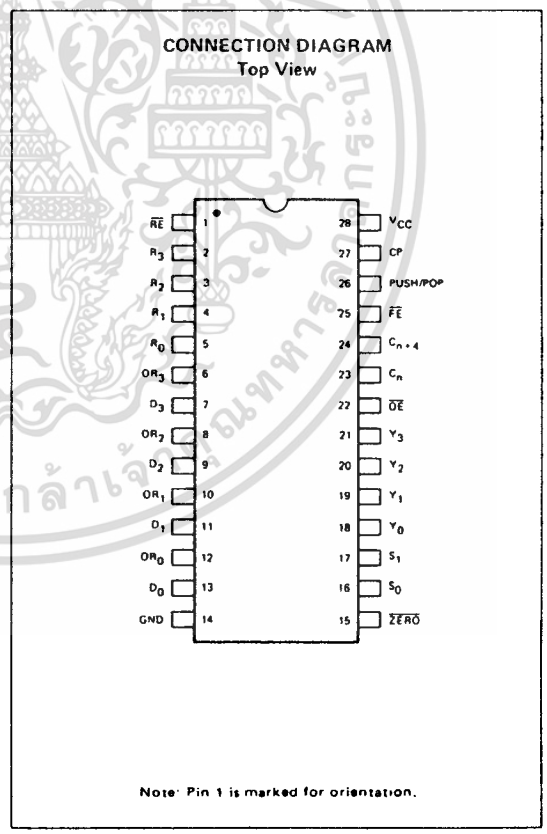
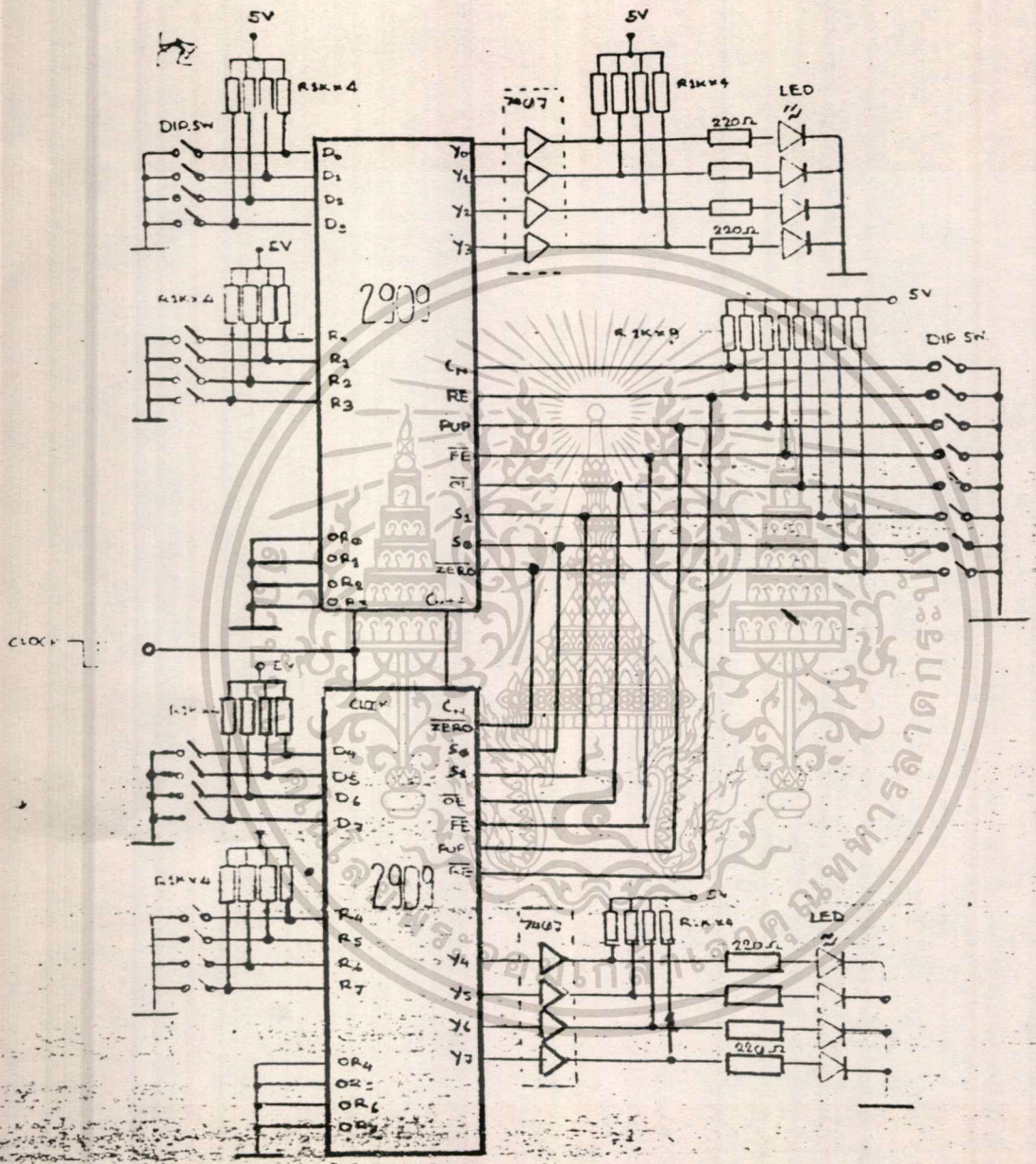


Figure 4.



8 BIT MICROPROGRAM SEQUENCER...

การทดลอง MC 2909 MICROPROGRAM SEQUENCER..

การทดลองที่ 1. การ PUSH - POP ค่าจาก DIRECT INPUT

ลำดับการทดลอง.

1.ให้ขา S1 และ S0เป็น 1 ทั้งคู่ และให้ขา PUP เป็น 1 (เพื่อทำการ PUSH

2.ป้อน INPUT ที่ขา D0 - D7 ครั้งที่ 1 ด้วยค่า 00011000

3.ป้อน CLOCK ที่ขา CLOCK ของ 2909 ครั้งที่ 1 หนึ่งลูก

4.ป้อน INPUT ที่ขา D0 - D7 ครั้งที่ 2 ด้วยค่า 00111100

5.ป้อน CLOCK ที่ขา CLOCK ของ 2909 ครั้งที่ 2 หนึ่งลูก

6.ป้อน INPUT ที่ขา D0 - D7 ครั้งที่ 3 ด้วยค่า 01111110

7.ป้อน CLOCK ที่ขา CLOCK ของ 2909 ครั้งที่ 3 หนึ่งลูก

8.ป้อน INPUT ที่ขา D0 - D7 ครั้งที่ 4 ด้วยค่า 11111111

9.ป้อน CLOCK ที่ขา CLOCK ของ 2909 ครั้งที่ 4 หนึ่งลูก

10.ป้อน CLOCK ที่ขา CLOCK ของ 2909 ครั้งที่ 5 อีกครั้ง

การทดลองทั้ง 10 ขั้นตอนนี้เป็นการ PUSH ค่าจาก DIRECT INPUT ลง STACK ของ 2909 ต่อไปจะทดลอง POP ค่าเหล่านั้นออกดู โดยให้ขา CONTROL S1 และ S0 เป็น 1 และ 0 ตามลำดับ และให้ขาPUPเป็น 0 จากนั้นก็ป้อน CLOCK ทีละลูก

CLOCK NO.	OUTPUT DISPLAY
0	1111 1111
1	0111 1110
2	0011 1100
3	0001 1000
4	1111 1111

* ทดลองทำหลายๆครั้ง ด้วยค่า[D0 - D7] หลายๆค่าก็ให้ค่าที่ถูกต้องตามลักษณะของ STACK (เข้าที่หลัง ออกก่อน)

การทดลองที่ 2. การ PUSH - POP จาก ADRESS REGISTER

ลำดับการทดลอง..

1.ให้ S1 และ S0 เป็น 0 และ 1 ตามลำดับ และให้ขา PUP เป็น 1 เพื่อทำการ PUSH

2.ป้อน INPUT ที่ขา R0 -R7 ครั้งที่ 1 ด้วยค่า 0001 1000

3.ป้อน CLOCK ที่ขา CLOCK ของ 2909 ครั้งที่ 1

4.ป้อน INPUT ที่ขา R0 -R7 ครั้งที่ 2 ด้วยค่า 0011 1100

5.ป้อน CLOCK เข้า 2909 ครั้งที่ 2

6.ป้อน INPUT ที่ขา R0 -R7 ครั้งที่ 3 ด้วยค่า 0111 1110

7.ป้อน CLOCK เข้า 2909 ครั้งที่ 3

8.ป้อน INPUT ที่ขา R0 -R7 ครั้งที่ 4 ด้วยค่า 1111 1111

9.ป้อน CLOCK เข้า 2909 ครั้งที่ 4

10.ป้อน CLOCK เข้า 2909 ครั้งที่ 5

11.ให้ S1 และ S0 เป็น 0 ทั้งคู่ และขา PUP เป็น 1

12.ป้อน CLOCK เข้า 2909 ครั้งที่ 6

ทั้ง 12 ขั้นตอนนี้เป็นการทดลองการ PUSH ค่าจาก ADDRESS REGISTER ลงใน STACK ของ 2909 ต่อไปจะทดลอง POP ค่าเหล่านี้ออกดู โดยให้ขา S1 และ S0 เป็น 1 และ0 ตามลำดับ และให้ขา PUP เป็น 0 (ทำการ POP) จากนั้นก็ให้ CLOCK ทีละลูก

CLOCK NO.	OUTPUT DISPLAY
0	1111 1111
1	0111 1110
2	0011 1100
3	0001 1000
4	1111 1111

สรุปผลการทดลอง..

ค่า Y. OUTPUT นั้นมาจาก SOURCE 4 แห่ง ซึ่จากการทดลองทั้ง 2 ที่ผ่านมานั้นจะใช้ SOURCE ทั้ง4 ครบ. การทดลองอันแรก (PUSH - POP จาก DIRECT INPUT) เมื่อให้ขา S1 และS0 เป็น 1 ทั้งคู่ เมื่อป้อน INPUT D0 -D7 เป็นอะไรที่ LED. DISPLAY จะแสดงค่านั้นทันที. CLOCK ลูกที่ 1 ที่ป้อนเข้าไปนั้นเมื่อดูจาก BLOCK DIAGRAM ของ 2909 จะเห็นว่าค่าที่ OUTPUT [O0 -03] ของ MICROPROGRAM COUNTER REGISTER ซึ่งขณะนั้นเป็นอะไรก็ไม่รู้จะถูก PUSH ลง STACK.พอ CLOCK ลูกที่ 2 เข้ามาค่าที่ป้อนผ่าน D0 -D7 ครั้งแรก (0001 1000) ถึงจะถูก PUSH ลง STACK. CLOCK ลูกที่ 3 ก็จะทำให้ค่าที่ป้อนผ่าน D0 - D7 ครั้งที่ 2 (0011 1100) ถูก PUSH ลง STACK. ดังนั้น CLOCK ลูกที่ 5จึงจะทำให้ค่าที่ป้อนผ่าน D0 -D7 ครั้งที่ 4 (1111 1111) ถูก PUSH ลง STACK. ในขณะทำการ PUSH นั้นจะทำให้ที่ STACK POINTER ทำการ COUNT UP ขึ้น 1 ครั้ง ก็เป็นอันเสร็จขบวนการ PUSH

ส่วนการ POP นั้นก็เป็นไปตามลักษณะของ STACK คือค่าที่ PUSH เข้าที่หลังจะถูก POP ออกมาก่อน นั่นคือค่า 1111 1111 จะถูก POP ออกมาก่อน ซึ่งทำได้โดยให้ S1 และ S0 เป็น 1 และ 0 ซึ่งก็คือการที่ค่า Y. OUTPUT รับ SOURCE จาก STACK ซึ่งก็คือการเอาค่าจาก STACK ออกมา (POP) เมื่อป้อน CLOCK เข้ามา 1 ลูกจะทำให้ STACK POINTER ทำการ COUNT DOWN 1 ครั้ง ดังนั้นค่าใน STACK ในตำแหน่งที่ 2 ซึ่งก็คือค่าที่ PUSH ลงไปเป็นครั้งที่ 3 (0111 1110) จะถูก POP ออกมา. เมื่อป้อน CLOCK ลูกที่ 2 จะทำให้ที่ LED.

DISPLAY แสดงค่าที่ถูก PUSH ลงไปเป็นครั้งที่ 2 (0011 1100) หรือกล่าวว่าค่า 0011 1100 ถูก POP ออกมา

หากต้องการ PUSH และ POP ค่าผ่านทาง D0 -D7 เพียงค่าเดียว ก็ทำได้โดยเมื่อป้อนINPUT เสร็จแล้วก็ป้อน CLOCK ให้ 2 ครั้งจากนั้น ก็ทดลอง POP ออกดูด้วยวิธีที่กล่าวมาแล้ว. พอป้อน CLOCK เข้าไป ค่าที่ถูก PUSH จะถูกนำมาให้เห็นทาง LED. DISPLAY

ลักษณะของ STACK POINTER จะเป็นวงจรนับ UP/DOWN นับแบบ CIRCULATE (นับวน) เป็นวงจรนับ 4 . ในขณะทำการ POP เมื่อป้อน CLOCK ลูกที่ 5 ที่ LED. DISPLAY จึงแสดงค่า 1111 1111 ซึ่งค่านี้ อยู่ใน STACK ตำแหน่งที่ 1 นั่นเอง.

จากการทดลองอันที่ 2 (PUSH POP จาก ADDRESS REGISTER) เมื่อให้ขา S1 และ S0 เป็น 0 และ 1 ตามลำดับ เมื่อป้อน INPUT R0 -R7 จะเห็นว่าที่ LED. DISPLAY จะยังไม่แสดงค่าที่ป้อนนั้นจน กว่าจะมี CLOCK ให้ 1 ลูก ในขณะทำการ PUSH ค่าลง STACK นั้น ก็จะยังคงใช้ CLOCK 5 ลูกเหมือนการทดลองครั้งที่แล้ว แต่มีขั้นตอน เพิ่มขึ้นอีกคือ เมื่อป้อน INPUT เสร็จแล้วจะต้องทำให้ที่ S1 และ S0 เป็น 0 ทั้งคู่ (ให้ Y. OUTPUT รับ SOURCE จาก MICROPROGRAM COUNTER) แล้วจากนั้นก็ให้ CLOCK ตามอีก 1 ลูก ถ้าหากไม่มี 2ขั้น ตอนนี้ เวลา POP ออกมาแล้วจะได้ว่า ที่ LED. DISPLAY จะแสดง ค่า 0111 1110 ทันที

CLOCK NO.	OUTPUT DISPLAY
1	0011 1100
2	0001 1000
3	1111 1111
4	0111 1110
7	1111 1111

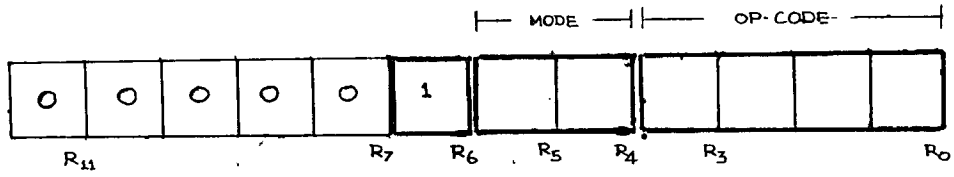
จบการทดลองเรื่อง MC 2909 MICROPROGRAM SEQUENCER

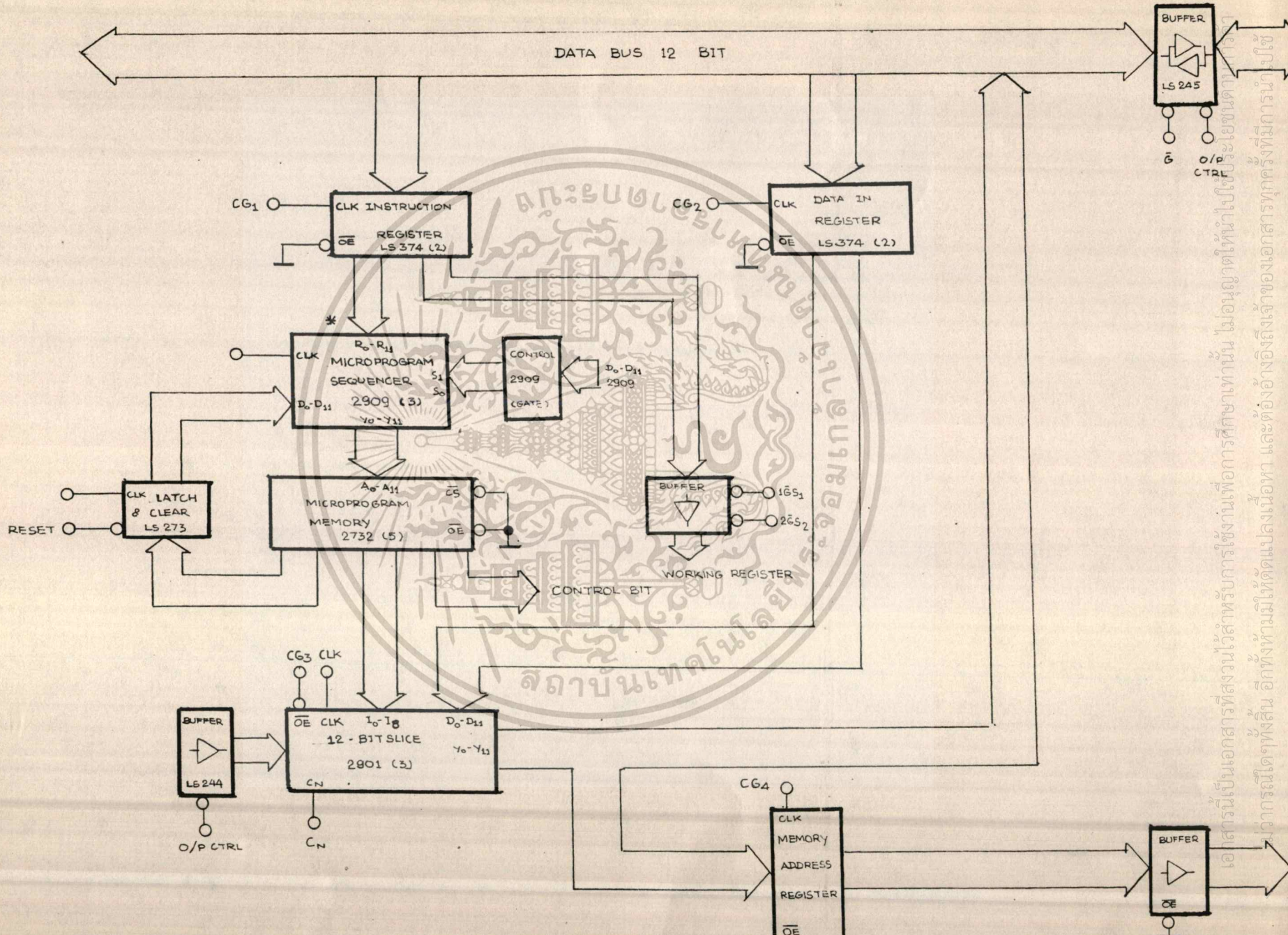
HARD WARE..

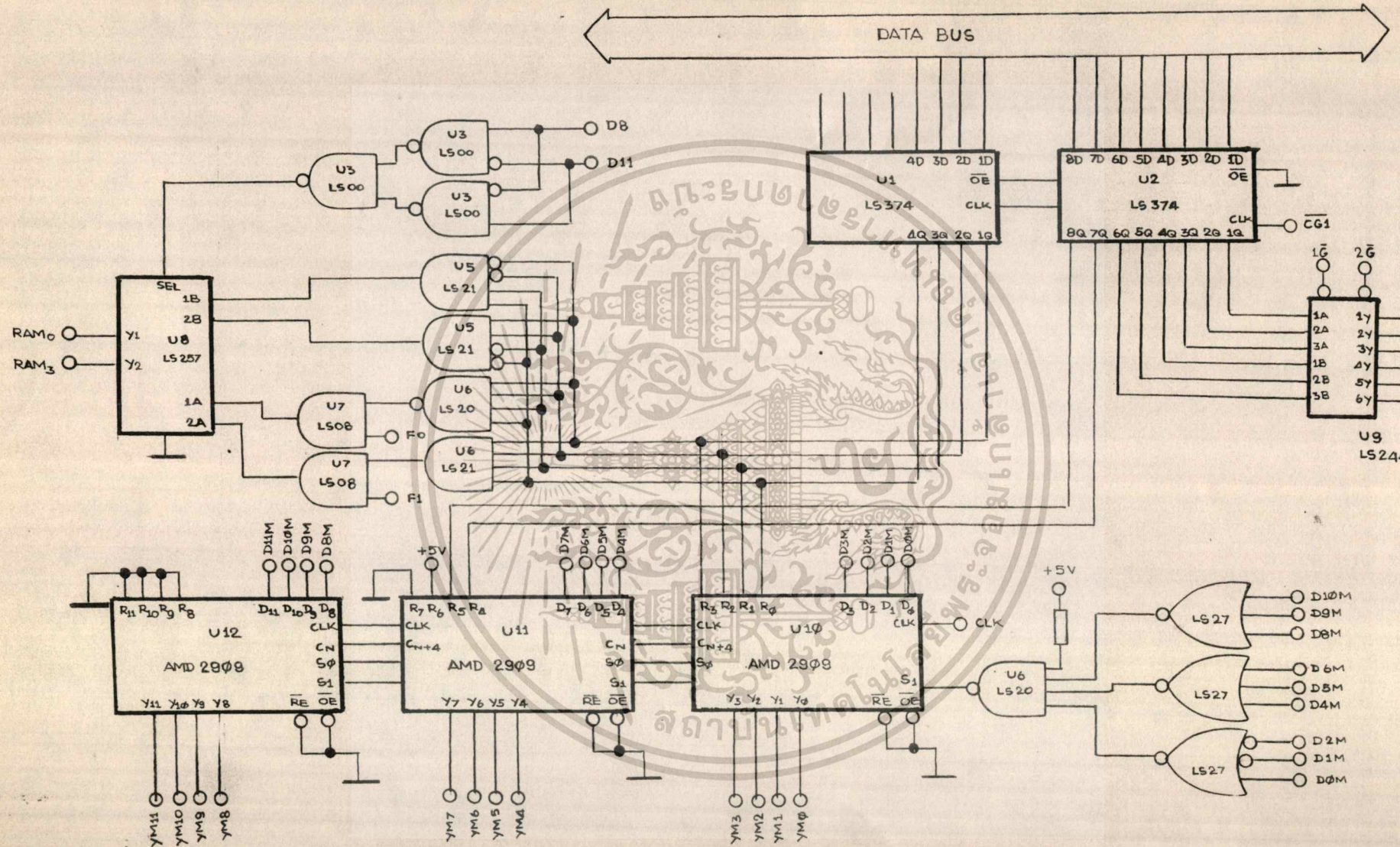
พิจารณาจาก BLOCK DIADRAM การทำงานจะเริ่มจากการที่ PROCESSER ส่ง ADDRESS ออกไปยัง MEMORY เพื่อไปเอา OP - CODE ขนาด 12 BIT เข้ามาซึ่งจะเข้ามาทาง INSTRUCTION REGISTER ซึ่งอยู่ใน BOARD #1 จากนั้น OP - CODE จะถูกแบ่งเป็น 2 ส่วน คือส่วน OP - CODE AND MODE กับส่วน OPERAND ในส่วน OP - CODE AND MODE นั้นจะเป็น IN PUT ให้ MICRO PROGRAM SEQUENCER 2909 เพื่อทำการ DECODE. ลักษณะการ DECODE ของ PROCESSER นี้จะนำเอา OP - CODE มาเป็น ADDRESS ให้ MICROPROGRAM MEMORY ซึ่งเป็น MEMORY ขนาด 35 BIT ใช้ CONTROL ส่วนต่างๆของ HARD WARE ของ SYSTEM. ส่วนที่เป็น OPER AND ใน BOARD #1 จะถูกส่งไป ALU..(BIT-SLICE) ใน BOARD #3 โดยผ่านทาง MULTIPLEXER..

IR.AND MICRO SEQUENCE BOARD (BOARD #1)

ใน BOARD นี้จะประกอบด้วย INSTRUCTION REGISTER ขนาด 12BIT ใช้ 74LS374 และ BUFFER สำหรับส่วนที่เป็น OPERAND. นอกจากนั้นก็ยังมี MICROPROGRAM SEQUENCER 2909 ซึ่งใช้ในการ DECODE OP - CODE ที่มาจาก INSTRUCTION REG. ที่ 2909 OP - CODE จะถูก FORMAT ใน รูป..




DATA BUS 12 BIT
BUFFER
LS 245
$\bar{G}$
O/P CTRL
CG_1
CLK INSTRUCTION
$\overline{OE}$
REGISTER
LS 374 (2)
CG_2
CLK
DATA IN
REGISTER
$\overline{OE}$
LS 374 (2)
CLK
R_0-R_{11}
MICROPROGRAM
SEQUENCER
2909 (3)
D_0-D_{11}
Y_0-Y_{11}
S_1
S_0
CONTROL
2909
(GATE)
D_0-D_{11}
2909
CLK
LATCH
& CLEAR
LS 273
RESET
A_0-A_{11}
MICROPROGRAM
MEMORY
2732 (5)
$\overline{CS}$
$\overline{OE}$
BUFFER
$1\bar{G}S_1$
$2\bar{G}S_2$
WORKING REGISTER
CONTROL BIT
CG_3
CLK
BUFFER
LS 244
O/P CTRL
$\overline{OE}$
CLK
I_0-I_8
D_0-D_{11}
12 - BIT SLICE
2901 (3)
Y_0-Y_{11}
C_N
C_N
CG_4
CLK
MEMORY
ADDRESS
REGISTER
$\overline{OE}$
BUFFER
$\overline{OE}$


DATA BUS
U1
LS374
4D 3D 2D 1D
OE
CLK
4Q 3Q 2Q 1Q
U2
LS374
8D 7D 6D 5D 4D 3D 2D 1D
OE
CLK
CG1
8Q 7Q 6Q 5Q 4Q 3Q 2Q 1Q
U3
LS00
D8
D11
U5
LS21
U6
LS20
U7
LS08
F0
F1
SEL
1B
2B
U8
LS257
1A
2A
Y1
Y2
RAM0
RAM3
1G
2G
1A
2A
3A
1B
2B
3B
1Y
2Y
3Y
4Y
5Y
6Y
S20
S21
S22
S10
S11
S12
U9
LS244
+5V
U12
U11
U10
AMD 2909
CLK
CN+4
CN
S0
S1
OE
RE
D11M
D10M
D9M
D8M
D7M
D6M
D5M
D4M
D3M
D2M
D1M
D0M
YM11
YM10
YM9
YM8
YM7
YM6
YM5
YM4
YM3
YM2
YM1
YM0
LS27
D10M
D9M
D8M
D6M
D5M
D4M
D2M
D1M
D0M

โดยที่ D0 - D11 จะถูกป้อนเข้า INPUT R0 - R11 ของ2909 ดังนั้นที่ D7 - D11 จึงถูกกำหนดให้เป็น 00 ทั้งหมดและ D6 เป็น 1 D0 - D5 ถูกป้อนเข้า R0 - R5 แต่จากลักษณะของ 2909 จะมี 2 INPUT คือ R.INPUT และ DIRECT INPUT ซึ่งจะเป็น INPUT ที่ถูกป้อนกลับมาจาก MICROPROGRAM MEMORY เพื่อเป็น NEXT ADDRESS ของ MICROPROGRAM ดังนั้น 2909จึงต้องมีขา CONTROL (S1) ว่าที่ OUTPUT Y0 - Y11 จะมาจาก INPUT ตัวใด. และสัญญาน S1 นี้มาจากการ DECODE ของ DIRECT INPUT. OP CODE ที่มาจาก INSTRUCTION REG. นี้จะถูกต่อไปยังส่วน CONTROL การ SHIFT ด้วย เพื่อทำการแยกคำสั่ง SHIFT และ ROTATE เพราะใน BIT-SLICE นั้นคำสั่ง SHIFT และ ROTATE มี MICRO CODE เดียวกัน นอกจากนี้ในส่วน SHIFTING CONTROL นี้จะทำการแยก SHIFT เป็น LEFT และ RIGHT ด้วย และไม่ว่า SHIFT LEFT , RIGHT หรือ ROTATE ก็คือการส่งค่า 1 หรือ 0 ไปยัง RAM 0 และ RAM 3 ของ BIT SLICE

MICROPROGRAM MEMORY BOARD (BOARD #2)

ใน BOARD นี้จะประกอบด้วย EPROM ขนาด 8 BIT จำนวน 5 ตัว.เพื่อใช้ CONTROL ส่วนต่างๆของ HARD WARE ในขณะทำการ EXECUTE อันได้แก่

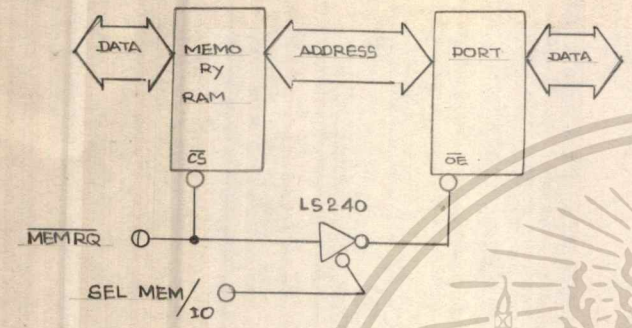
1.) ส่วนที่ป้อนกลับมาเป็น NEXT ADDRESS เข้าที่ 2909

2.) เป็น MICRO CODE ให้ BIT SLICE (I0 - I8)

3.) เปิด ปิด REGISTER และ BUFFER ต่างๆในระบบ

4.) กำเนิดสัญญาน MEMRQ และ IORQ

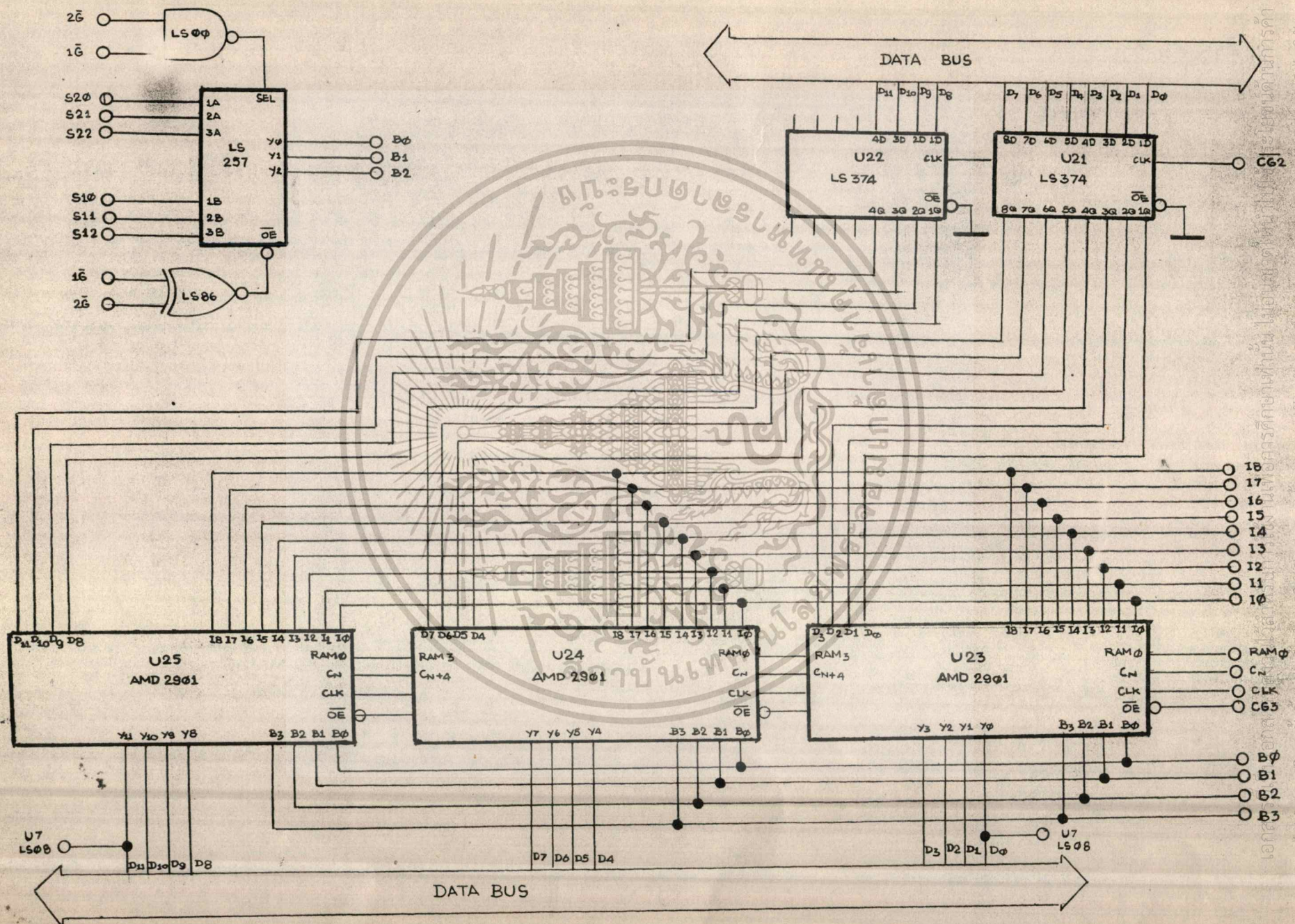
ในส่วนที่ใช้กำเนิดสัญญาน MEMRQ และ IORQ นั้น เนื่องจากสัญญานทั้งสองจะไม่เกิดพร้อมกัน ดังนั้นจึงใช้ GATE ช่วยดังรูป..



SEL MEM/IO	MEMRQ	IORQ	COMMENT
0	1	0	IORQ
0	0	1	
1	1	Z	NO USE
1	0	Z	MEMRQ

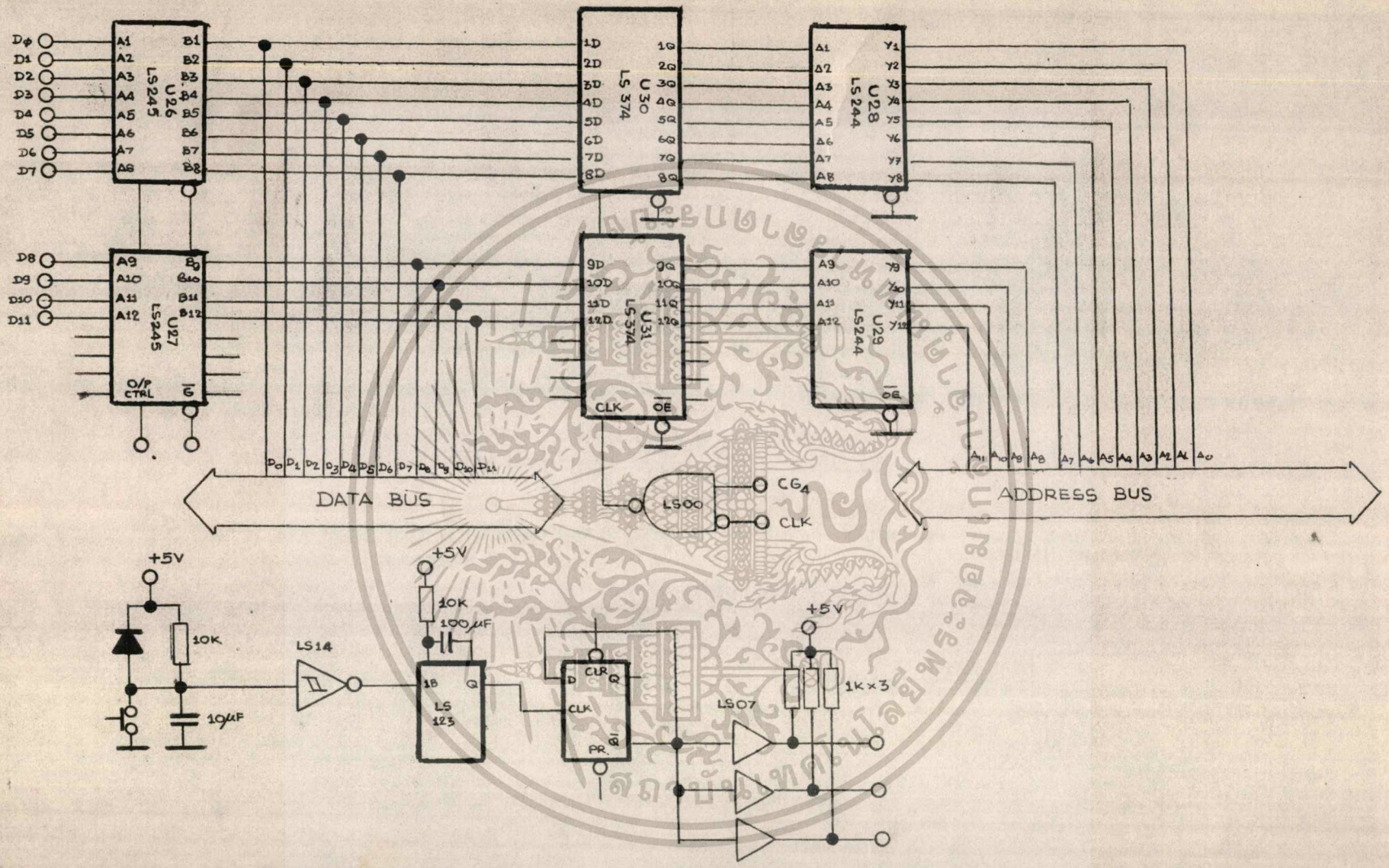
TRUTH TABLE แสดงการ SELECT MEM/IO

พิจารณาจาก TRUTH TABLE จะได้ว่าเมื่อ SEL MEM/IO เป็น 0 จะเป็นสัญญาน IORQ เนื่องจากสัญญานนี้ ACTIVE ที่ LOW ดังนั้น สัญญาน MEMRQ จะต้องเป็น HIGH ทำให้ PORT ถูก SELECT เพียงอย่างเดียว และเมื่อ SEL MEM/IO เป็น 1 จะเป็นสัญญาน MEMRQ ที่มาจาก EPROM แต่ถ้า MEMRQ เป็น 1 แล้วจัเป็นกรณี NO USE คือไม่มีการใช้งาน MEMORY และ PORT .. ADDRESS ของ MICROPROGRAM นี้ถูกส่งมาจาก OUTPUT Y0 - Y11 ของ MICROPROGRAM SEQUENCER 2909


2Ḡ
1Ḡ
LSΦΦ
S2Φ
S21
S22
1A
2A
3A
SEL
LS
257
YΦ
Y1
Y2
BΦ
B1
B2
S1Φ
S11
S12
1B
2B
3B
OE
1Ḡ
2Ḡ
LS86
DATA BUS
D11 D10 D9 D8
D7 D6 D5 D4 D3 D2 D1 DΦ
4D 3D 2D 1D
U22
LS 374
CLK
OE
4Q 3Q 2Q 1Q
8D 7D 6D 5D 4D 3D 2D 1D
U21
LS 374
CLK
OE
8Q 7Q 6Q 5Q 4Q 3Q 2Q 1Q
CG2
I8
I7
I6
I5
I4
I3
I2
I1
IΦ
U25
AMD 29Φ1
U24
AMD 29Φ1
U23
AMD 29Φ1
RAMΦ
CN
CLK
OE
RAM3
CN+4
Y11 Y10 Y9 Y8
B3 B2 B1 BΦ
Y7 Y6 Y5 Y4
Y3 Y2 Y1 YΦ
CG3
U7
LSΦ8
DATA BUS

ALU.. BOARD (BOARD #3)

ใน BOARD นี้จะทำหน้าที่เป็น ALU (ARITHMATIC LOGIC UNIT) เหมือน PROCESSER ทั่วๆไป จะต่างกันก็ตรงลักษณะของการเป็น BIT SLICE คือจะมี FUNCTION ที่ช่วยในการ SHIFT ได้ ซึ่งแตกต่างจาก ALU ทั่วๆไป และใน IC. BIT SLICE เองจะมี 2-PORT RAM 1 ชุดทำหน้าที่เป็น WORKING REGISTER จำนวน 8 REGISTER (REG0 - REG7) ADDRESS ของ 2-PORT RAM จะมาจาก MULTIPLEXER ซึ่ง SELECT ค่า ADDRESS มาจากส่วนที่เป็น OPERAND ของ INSTRUCTION REGISTER (S1 และ S2). ผลลัพธ์ของการ EXECUTE จะถูกนำออกที่ OUTPUT ของ BIT SLICE ลง DATA BUS ไปยังส่วนต่างๆของ SYSTEM..


U26 LS245
U30 LS 374
U28 LS244
U27 LS245
U31 LS 374
U29 LS244
O/P CTRL
CLK
OE
DATA BUS
ADDRESS BUS
LS00
CG4
CLK
+5V
10K
100µF
LS14
LS 123
CLR
PR
LS07
1K x 3
10µF

MAR. AND BUFFER BOARD (BOARD #4)

ใน BOARD นี้จะมีส่วนที่เป็น CLOCK GENERATE ซึ่งเป็น SINGLE STEP จ่ายสัญญาน CLOCK ไปยังส่วนต่างๆของ SYSTEM. และยังมี MEMORY ADD RESS REGISTER (MAR) ใช้ 74LS374 ซึ่งทำหน้าที่ส่ง ADDRESS ไปยังหน่วยความจำผ่านทาง ADDRESS BUS โดยมี BUFFER 1 ตัวเพื่อขยายสัญญาน. และที่ BOARD นี้จะมี BUFFER อีก 1 ตัวสำหรับรับส่งข้อมูลระหว่างหน่วยความจำภายนอกกับ CPU โดยผ่านทาง DATA BUS . และ BUFFER ในส่วนของ DATA นี้จะใช้ TTL เบอร์ 74LS245 ซึ่งเป็น BI - DIRECT IONAL BUFFER.

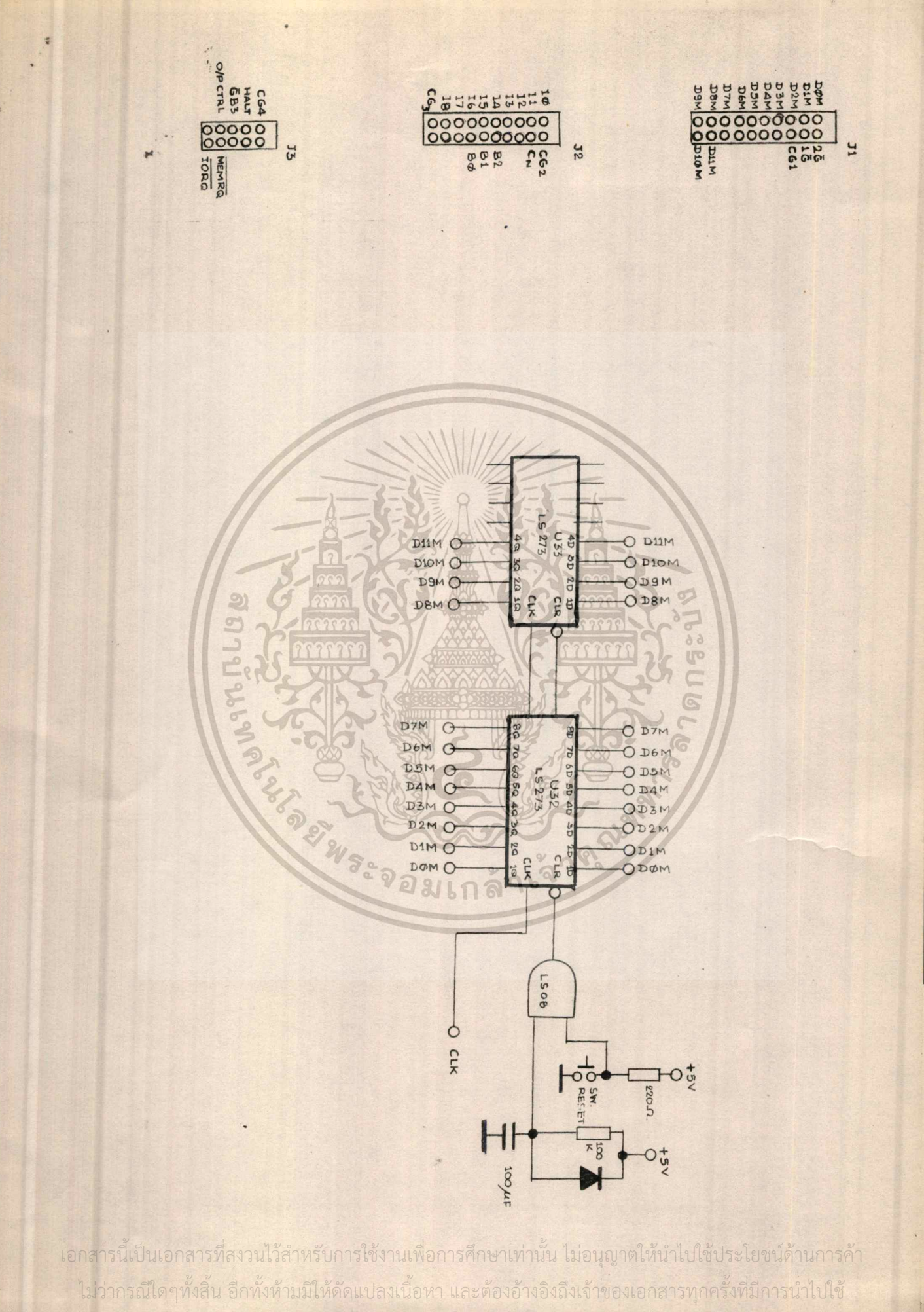
LATCH AND RESET BOARD (BOARD #5)

เนื่องจาก ADDRESS ของ MICCRO PROGRAM (NEXT ADDRESS) มาจากการป้อนกลับจากตัวมันเอง. ดังนั้นจึงต้องมีการ LATCH เพื่อให้ MICROPROGRAM ทำคำสั่ง MICROINSTRUCTION ของ ADDRESS เก่าให้เสร็จก่อน. จากนั้นจึงส่ง ADDRESS ต่อไปให้ MICROPROGRAM SEQUENCER

การ RESET ของ PROCESSER นี้จะมี 2 วิธีดังนี้..

1. POWER ON RESET เป็นการ RESET ตัวเองเมื่อ ON POWER ครั้งแรก
2. RESET BY SWITCH .

สัญญาน RESET ทั้ง 2 นี้จะถูกนำไปทำการ AND กัน. ผลของการ AND จะถูกส่งไปเข้า INPUT CLEAR ของ 74LS273 ซึ่งทำหน้าที่ LATCH NEXT ADDRESS ด้วย. เมื่อ 74LS273 ถูก CLEAR จะทำให้ ADDRESS ของ MICROPROGRAM เป็น 000H



การออกแบบชุดคำสั่ง (MACRO INSTRUCTION).

ชุดคำสั่งที่จะใช้ใน PROCESSER นี้จะแบ่งออกได้เป็น 4 ประเภทด้วยกันคือ...

1. คำสั่งเกี่ยวกับการ LOAD
2. คำสั่งทางคณิตศาสตร์และ LOGIC
3. คำสั่งการเคลื่อนย้ายข้อมูลเป็นวงรอบและการ SHIFT
4. คำสั่งเกี่ยวกับ INPUT / OUTPUT

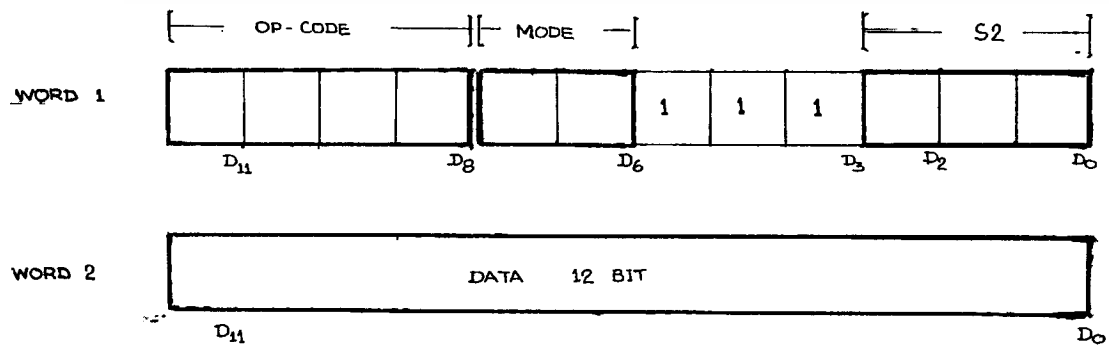
ในคำสั่งทั้ง 4 ประเภทนี้สามารถกระทำได้โดยใช้ REGISTER ต่างๆที่อยู่ใน PROCESSER หรือกระทำระหว่าง REGISTER กับหน่วยความจำ หรือจาก หน่วยความจำเข้าหา REGISTER ก็ได้. คำสั่งที่เกี่ยวกับ INPUT/OUTPUT จะมี 2 คำสั่ง คือ IN S2 กับ OUT S2...

ADDRESSING MODE..

เนื่องจาก PROCESSER จะต้องทำการติดต่อกับหน่วยความจำและอุปกรณ์ INPUT/OUTPUT เพราะฉะนั้น PROCESSER จำเป็นต้องมีการเรียก ADDRESS ที่ต้องการ [ADDRESSING] ซึ่งมีด้วยกัน 4 แบบ (MODE)

1) IMMEDIATE ADDRESSING.[MEMORY TO REGISTER]

ใน MODE นี้จะประกอบด้วย OP-CODE 1 WORD และตามด้วย OPERAND อีก 1 WORD.. ลักษณะคำสั่งจะเป็นดังนี้



MACRO INSTRUCTION SET

ขั้นตอนต่อไปของการออกแบบชุดคำสั่งก็คือการกำหนด OP-CODE และ OPERATION ของแต่ละคำสั่งในแต่ละ ADDRESSING MODE.[INSTRUCTION TYPE]

[1] IMMEDIATE INSTRUCTION TYPE.

เป็นชุดคำสั่งใน IMMEDIATE ADDRESSING MODE มีอยู่ 14 คำสั่ง

OP-CODE	MODE	OPERATION	MNEMONIC
0000	00	S2= DATA LEFT ROTATE	RL S2,#DATA
0001	00	S2= DATA + 1	INC S2,#DATA
0010	00	S2= DATA - 1	DEC S2,#DATA
0011	00	S2= DATA LEFT SHIFT	SL S2,#DATA
0100	00	S2= S2 - DATA	SUB S2,#DATA
0101	00	S2= S2 - DATA - 1	SUBC S2,#DATA
0110	00	S2= S2 + DATA	ADD S2,#DATA
0111	00	S2= S2 + DATA + 1	ADDC S2,#DATA
1000	00	S2= DATA	LD S2,#DATA
1001	00	S2= S2 XOR DATA	XOR S2,#DATA
1010	00	S2= S2 OR DATA	OR S2,#DATA
1011	00	S2= S2 AND DATA	AND S2,#DATA
1100	00	S2= DATA RIGH SHIFT	SR S2,#DATA
1111	00	S2= DATA RIGH ROTATE	RR S2,#DATA

[2] MEMORY TO REGISTER INSTRUCTION TYPE

เป็นชุดคำสั่งใน INDIRECT OPERAND ADDRESSING MODE

OP-CODE	MODE	OPERATION	MNEMONIC
0000	01	S2= DATA LEFT ROTATE	RL S2,(S1)
0001	01	S2= DATA + 1	INC S2,(S1)
0010	01	S2= DATA - 1	DEC S2,(S1)
0011	01	S2= DATA LEFT SHIFT	SL S2,(S1)
0100	01	S2= S2 - DATA	SUB S2,(S1)
0101	01	s2= s2 - DATA - 1	SUBC S2,(S1)
0110	01	S2= S2 + DATA	ADD S2,(S1)
0111	01	S2= S2 + DATA + 1	ADDC S2,(S1)
1000	01	S2= DATA [MEMORY]	LD S2,(S1)
1001	01	S2= S2 XOR DATA	XOR S2,(S1)
1010	01	S2= S2 OR DATA	OR S2,(S1)
1011	01	S2= S2 AND DATA	AND S2,(S1)
1100	01	S2= DATA RIGH SHIFT	SR S2,(S1)
1101	01	S2=DATA [I/O PORT]	IN S2,(S1)
1111	01	S2= DATA RIGH ROTATE	RR S2,(S1)

---

* DATA ในที่นี้หมายถึง DATA จาก MEMORY ที่กำหนดโดย S1

[3] REGISTER TO MEMORY INSTRUCTION TYPE.

เป็นชุดคำสั่งใน INDIRECT RESULT ADDRESSING MODE

OP-CODE	MODE	OPERATION	MNEMONIC
0000	10	(S2)= DATA LEFT ROTATE	RL [S2]
0001	10	(S2)= DATA + 1	INC [S2]
0010	10	(S2)= DATA - 1	DEC [S2]
0011	10	(S2)= DATA LEFT SHIFT	SL [S2]
1000	10	(S2)= DATA	LD [S2],S1
1100	10	(S2)= DATA RIGH SHIFT	SR [S2]
1110	10	(S2)= DATA (FROM S1)	OUT [S2],S1
1111	10	(S2)= DATA RIGH ROTATE	RR [S2]

* DATA ในที่นี้หมายถึง DATA จาก REGISTER [S1] ในกรณีคำสั่ง LD (S2),S1 จะหมายถึง ข้อมูลที่อยู่ใน S1 จะถูกนำไปเก็บใน MEMORY ในตำแหน่งที่ชี้โดย OPERAND [S2] แต่ในกรณีคำสั่ง OUT (S2),S1 นั้น ข้อมูลใน REGISTER [S1] จะถูกนำไปเก็บที่ PORT ในตำแหน่งที่ชี้โดย OPERAND [S2]..

[4] REGISTER TO REGISTER INSTRUCTION TYPE.

เป็นชุดคำสั่งที่อยู่ใน REGISTER ADDRESSING MODE

OP-CODE	MODE	OPERATION	MNEMONIC
0000	11	S2= S1 LEFT ROTATE	RL S2,S1
0001	11	S2= S1 + 1	INC S2,S1
0010	11	S2= S1 - 1	DEC S2,S1
0011	11	S2= S1 LEFT SHIFT	SL S2,S1
0100	11	S2= S2 - S1	SUB S2,S1
0101	11	S2= S2 - S1 - 1	SUBC S2,S1
0110	11	S2= S2 + S1	ADD S2,S1
0111	11	S2= S2 + S1 + 1	ADDC S2,S1
1000	11	S2= S1	LD S2,S1
1001	11	S2= S2 XOR S1	XOR S2,S1
1010	11	S2= S2 OR S1	OR S2,S1
1011	11	S2= S2 AND S1	AND S2,S1
1100	11	S2= S1 RIGH SHIFT	SR S2,S1
1111	11	S2= S1 RIGH ROTATE	RR S2,S1

1

การเขียน ALGORITHM

ALGORITHM จะเป็นการแสดงขั้นตอนของการ EXECUTE MACRO INSTR UCTION ซึ่งในแต่ละ MODE ของ ชุดคำสั่งก็จะมีขั้นตอนการ EXECUTE แตกต่างกันไป ดังรายละเอียดที่จะกล่าวถึงต่อไปนี้..

ALGORITHM ของการ FETCH..

1. PC0 --- MAR ; MAR = MEMORY ADDRESS REGISTER
   PC0 = PROGRAM COUNTER ตัวที่ 0
2. ส่งสัญญาน RD , MEMRQ ; MEMRQ = MEMORY REQUEST
3. PC0 + 1 --- PC1
4. OP - CODE --- IR ; IR = INSTRUCTION REGISTER
5. DECODE..

ใน ALGORITHM ทั้ง 5 ขั้นตอนนี้จะถูกนำไปเขียน MICRO PROGRAM ซึ่งจะกล่าวถึงภายหลัง..

ALGORITHM ของการ EXECUTE MACRO INSTRUCTION ใน MODE ต่างๆ

IMMDIAT INSTRUCTION TYPE MODE..

- คำสั่งเกี่ยวกับการ ROTATE

RL S2 , #DATA และ RR S2 , #DATA

1. PC1 --- MAR
2. ส่งสัญญาน RD , MEMRQ
3. PC1 + 1 --- PC2
4. DATA ที่จะ ROTATE --- S2 ; S2 = REGISTER 1 TO 6
5. ส่งสัญญาน CONTROL 2901

.....

MACRO INSTRUCTION ตัวอื่นๆนอกจาก ROTATE แล้วจะมี ALGORITHM เหมือนกันหมดต่างกันที่ MICRO CODE ที่ใช้ CONTROL BIT SLICE ซึ่งจะมี ALGORITHM ดังนี้..

1. PC1 --- MAR
2. ส่งสัญญาน RD , MEMRQ
3. PC1 +1 --- PC2
4. ส่งสัญญาน CONTROL 2901 ให้ทำคำสั่งนั้นๆ

.....

MEMMORY TO REGISTER INSTRUCTION TYPE MODE..

- คำสั่งเกี่ยวกับการ ROTATE

RL S2 , (S1) และ RR S2 , (S1)

1. S1 --- MAR ; S1 = REGISTER 1 TO 6
2. ส่งสัญญาน RD , MEMRQ
3. DATA จาก MEMORY ที่ชี้โดย S2 --- DATA IN REGISTER
4. DATA ที่จะทำ ROTATE --- S2
5. ส่งสัญญาน CONTROL 2901

.....

- คำสั่ง INPUT

IN S2 , (S1)

1. S1 --- MAR ; ADDRESS ของ PORT
2. ส่งสัญญาน IORQ ; IORQ = I/O. REQUEST
3. DATA จาก I/O ที่ชี้โดย S1 --- DATA IN REGISTER
4. ส่งสัญญาน CONTROL 2901

.....

MACRO INSTRUCTION ตัวอื่นๆนอกจาก INPUT และ ROTATE แล้ว จะมี ALGORITHM เหมือนกันหมด ต่างกันที่ MICRO CODE ที่ใช้ CONTROL BIT SLICE 2901. ซึ่งมี ALGORITHM ดังนี้..

1. S1 --- MAR
2. ส่งสัญญาน RD , MEMRQ
3. DATA จาก MEMORY ที่ชี้โดย S1 --- DATA IN REGISTER
4. ส่งสัญญาน CONTROL 2901 ให้ทำคำสั่งนั้นๆ

.....

REGISTER TO MEMORY INSTRUCTION TYPE MODE..

- คำสั่งเกี่ยวกับการ ROTATE

RL (S2) , RR (S2)

1. S2 --- MAR
2. ส่งสัญญาน RD , MEMRQ
3. DATA ที่จะทำ ROTATE --- S2
4. ส่งสัญญาน CONTROL 2901

5. ส่งสัญญาน WR , MEMRQ

6. ผลลัพธ์การ ROTATE --- MEMORY

.....

- คำสั่งเกี่ยวกับการ LOAD

LD (S2) , S1

1. S2 --- MAR

2. S1 --- B0 - B3 ของ BIT SLICE ; B0 -B3 = ADDRESS ของ REGISTER

3. ส่งสัญญาน CONTROL 2901

4. ส่งสัญญาน WR , MEMRQ

5. ผลลัพธ์การ LOAD --- MEMORY

.....

- คำสั่ง OUTPUT

OUT (S2) , S1

1. S2 --- MAR ; ไป ADDRESS ของ PORT

2. S1 --- B0 - B3 ของ BIT SLICE

3. ส่งสัญญาน CONTROL 2901

4. ส่งสัญญาน WR , IORQ

5. ผลลัพธ์การ OUT --- PORT

.....

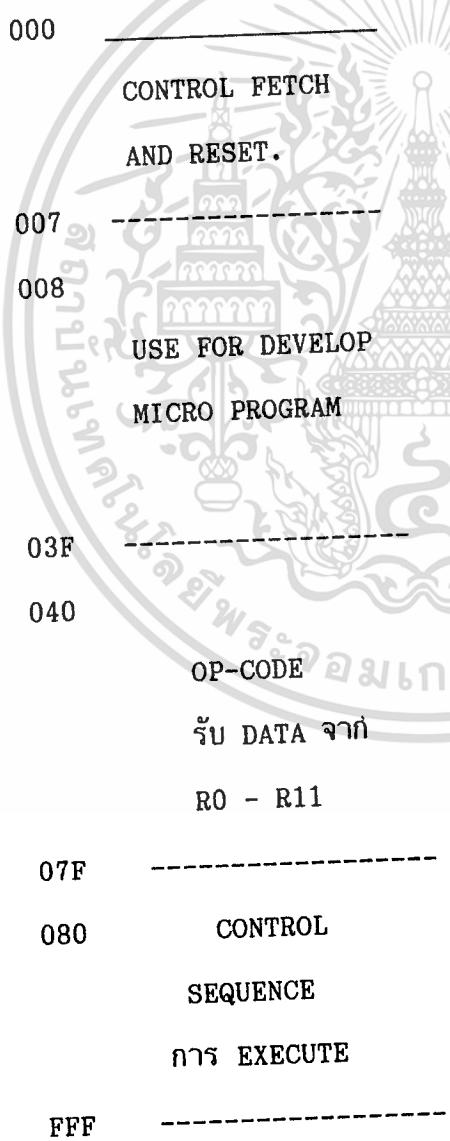
MACRO INSTRUCTION ตัวอื่นๆนอกจาก ROTATE , LOAD และ OUTPUT แล้ว จะมี ALGORITHM เหมือนกัน จะต่างกันที่ MICRO CODE ที่ใช้ CONTROL BIT SLICE. ซึ่งจะมี ALGORITHM ดังนี้..

1. S2 ---> MAR
2. ส่งสัญญาน RD , MEMRQ
3. ส่งสัญญาน CONTROL 2901 ให้ทำคำสั่งนั้นๆ
4. ส่งสัญญาน WR , MEMRQ
5. ส่งผลการ EXECUTE คำสั่งนั้นๆ --- MEMORY

.....

MICRO PROGRAMING..

จาก ALGORITHM ของการ EXECUTE คำสั่งต่างๆในหัวข้อที่แล้วต่อไปจะได้นำ ALGORITHM นั้นมาเขียน MICRO PROGRAM เพื่อ CONTROL การ EXECUTE ในแต่ละ MACRO INSTRUCTION ซึ่งขนาดของ MICRO PROGRAM จะขึ้นอยู่กับ HARD WARE การจัดเนื้อที่ของหน่วยความจำของ MICRO PROGRAM ของโครงงานนี้จะเป็นดังรูป..



ADDRESS MICRO PROGRAM	2901 CONTROL				CONTROL BIT													NEXT ADDRESS OF MICRO PROGRAM				OPERATION
	I_8 I_7 I_6	I_5 I_4 I_3	I_2 I_1 I_0		$\overline{15}$ 2951	$\overline{25}$ 2952	MEMRQ	MAR CG4	IR CG1	2901 CG3	DATA I/P CG2	C_N 2901	$\overline{OE}$ BUFF 2901	$\overline{E}$ DATA BUF=245	O/P CTRL DATA BUF	HALT	SELECT MEM/IO	$D_{11}D_{10}$ D_9 D_8	D_7 D_6 D_5 D_4	D_3 D_2 D_1 D_0	
RESET																						
000	0 1 1	0 0 0	0 1 1		1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	0 0 0 0	0 0 1 0	CLEAR PC [illegible]
FETCH																						
001	0 1 1	0 0 0	0 1 1		1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	0 0 0 0	0 0 1 0	(B+Φ)→B [illegible]
002	X X X	X X X	X X X		1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 1 1	DATA [illegible]
003	X X X	X X X	X X X		1	1	0	0	0	1	1	0	1	0	0	0	1	0 0 0 0	0 0 0 0	0 1 0 0	[illegible] RD, MEMRQ
004	0 1 1	0 0 0	0 1 1		1	1	0	1	0	1	1	1	0	0	0	0	1	0 0 0 0	0 0 0 0	0 1 0 1	$PC_0+1 \rightarrow PC_1$ .. OP-CODE [illegible] IR
005	X X X	X X X	X X X		1	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 1 1 0	OP-CODE → IR [illegible]
006	X X X	X X X	X X X		1	1	1	1	1	1	1	0	1	1	X	0	1	X X X X	X X X X	X X X X	[illegible]
RLS S_2, # DATA																						
040	0 1 1	0 0 0	0 1 1		1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	1 0 0 0	0 0 0 0	(B+Φ)→B [illegible]
048	X X X	X X X	X X X		1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 0 0	0 0 0 1	ADDR [illegible]
041	X X X	X X X	X X X		1	1	0	1	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 0 0	0 0 1 0	[illegible] RD, MEMRQ
042	0 1 1	0 0 0	0 1 1		1	1	1	1	1	0	1	0	0	0	0	1	0 0 0 0	1 0 0 0	0 0 1 1	$PC_0+1 \rightarrow PC_2$ .. [illegible]
043	0 1 1	0 0 0	1 1 1		1	0	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 0 0	0 1 0 0	[illegible] ROTATE → S_2
044	1 1 1	0 0 0	0 1 1		1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 0	1 0 0 0	0 1 0 1	CONTROL 2901
045	X X X	X X X	X X X		1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	[illegible] FETCH [illegible] ADDR 001
INC S_2, # DATA																						
041	0 1 1	0 0 0	0 1 1		1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	1 0 0 0	0 1 1 0	(B+Φ)→B [illegible]
046	X X X	X X X	X X X		1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 0 0	0 1 1 1	ADDR [illegible]
047	X X X	X X X	X X X		1	1	0	1	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 0 0	1 0 0 0	[illegible] RD, MEMRQ
048	0 1 1	0 0 0	0 1 1		1	1	1	1	1	1	0	1	0	0	0	0	1	0 0 0 0	1 0 0 0	1 0 0 1	$PC_1+1 \rightarrow PC_2$ .. [illegible]
049	0 1 1	0 0 0	1 1 1		1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 0	1 0 0 0	1 0 1 0	CONTROL 2901
04A	X X X	X X X	X X X		1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	[illegible] FETCH [illegible]
DEC S_2, # DATA																						
042	0 1 1	0 0 0	0 1 1		1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	1 0 0 1	0 0 0 0	(B+Φ)→B [illegible]
04C	X X X	X X X	X X X		1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 0 1	0 0 0 1	ADDR [illegible]
041	X X X	X X X	X X X		1	1	0	1	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 0 1	0 0 1 0	[illegible] RD, MEMRQ
042	0 1 1	0 0 0	0 1 1		1	1	1	1	1	1	0	1	0	0	0	0	1	0 0 0 0	1 0 0 1	0 0 1 1	$PC_1+1 \rightarrow PC_2$ .. [illegible]
043	0 1 1	0 1 0	1 1 1		1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 0	1 0 0 1	0 1 0 0	CONTROL 2901
044	X X X	X X X	X X X		1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	[illegible] FETCH [illegible]

ADDRESS MICRO PROGRAM	2901 CONTROL				CONTROL BIT													NEXT ADDRESS OF MICRO PROGRAM				OPERATION
	I_8	I_7 I_6	I_5 I_4 I_3	I_2 I_1 I_0	$\overline{1G}$ 2833 S1	$\overline{2G}$ 2833 S2	$\overline{MEMRQ}$	MAR CG4	IR CG1	2901 CG3	DATA I/P CG2	C_n 2901	$\overline{OE}$ BUFF 2901	$\overline{E}$ DATA BUF-BAS	O/P CTRL DATA BUF	HALT	SELECT MEM/IO	D_{11} D_{10} D_9 D_8	D_7 D_6 D_5 D_4	D_3 D_2 D_1 D_0	
SL S_2, #DATA																						
043	0	1 1	0 0 0	0 1 1	1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	1 0 0 1	0 1 0 1		
095	X	X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 0 1	0 1 1 0		
096	X	X X	X X X	X X X	1	1	0	1	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 0 1	0 1 1 1		
097	0	1 1	0 0 0	0 1 1	1	1	1	1	1	1	0	1	0	0	0	0	1	0 0 0 0	1 0 0 1	1 0 0 0		
098	1	1 1	0 0 0	1 1 1	1	0	1	1	1	0	1	0	1	1	1	0	1	0 0 0 0	1 0 0 1	1 0 0 1		
099	X	X X	X X X	X X X	1	1	1	1	1	1	1	0	1	1	1	0	1	0 0 0 0	0 0 0 0	0 0 0 1		
SUB S_2, #DATA																						
044	0	0 1	0 0 0	0 1 1	1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	1 0 1 0	0 0 0 0		
0A0	X	X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 1 0	0 0 0 1		
0A1	X	X X	X X X	X X X	1	1	0	1	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 1 0	0 0 1 0		
0A2	0	1 1	0 0 0	0 1 1	1	1	1	1	1	1	0	1	0	0	0	0	1	0 0 0 0	1 0 1 0	0 0 1 1		
0A3	0	1 1	0 0 1	1 0 1	1	0	1	1	1	0	1	1	1	1	X	0	1	0 0 0 0	1 0 1 0	0 1 0 0		
0A4	X	X X	X X X	X X X	1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1		
SBC S_2, #DATA																						
045	0	0 1	0 0 0	0 1 1	1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	1 0 1 0	0 1 0 1		
0A5	X	X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 1 0	0 1 1 0		
0A6	X	X X	X X X	X X X	1	1	0	1	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 1 0	0 1 1 1		
0A7	0	1 1	0 0 0	0 1 1	1	1	1	1	1	1	0	1	0	0	0	0	1	0 0 0 0	1 0 1 0	1 0 0 0		
0A8	0	1 1	0 0 1	1 0 1	1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 0	1 0 1 0	1 0 0 1		
0A9	X	X X	X X X	X X X	1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1		
ADD S_2, #DATA																						
046	0	0 1	0 0 0	0 1 1	1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	1 0 1 1	0 0 0 0		
0B0	X	X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 1 1	0 0 0 1		
0B1	X	X X	X X X	X X X	1	1	0	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 1 1	0 0 1 0		
0B2	0	1 1	0 0 0	0 1 1	1	1	0	0	1	1	0	1	0	0	0	0	1	0 0 0 0	1 0 1 1	0 0 1 1		
0B3	0	1 1	0 0 0	1 0 1	1	0	1	1	1	0	0	0	1	1	X	0	1	0 0 0 0	1 0 1 1	0 1 0 0		
0B4	X	X X	X X X	X X X	1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1		
ADC S_2, #DATA																						
047	0	0 1	0 0 0	0 1 1	1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	1 0 1 1	0 1 0 1		
0B5	X	X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 1 1	0 1 1 0		
0B6	X	X X	X X X	X X X	1	1	0	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 0 1 1	0 1 1 1		
0B7	0	1 1	0 0 0	0 1 1	1	1	0	0	1	1	0	1	0	0	0	0	1	0 0 0 0	1 0 1 1	1 0 0 0		
0B8	0	1 1	0 0 1	1 0 1	1	0	1	1	1	0	0	0	1	1	X	0	1	0 0 0 0	1 0 1 1	1 0 0 1		
0B9	X	X X	X X X	X X X	1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1		

MICRO PROGRAM	2901 CONTROL I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	CONTROL BIT $1\overline{S}$ 2BUS1	$2\overline{S}$ 2BUS2	MEMRQ	MAR CG4	IR C61	2901 C63	DATA I/P C62	C_N 2901	$\overline{OE}$ BUFF 2901	8 DATA BUF-2A5	O/P CTRL DATA BUF	HALT	SELECT MEM/IO	NEXT ADDRESS OF MICRO PROGRAM D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	OPERATION
LD S_2, # DATA																																			
048	0	0	1	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0	1	X	0	1	0	0	0	0	1	1	0	0	0	0	0	0	
0C0	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	0	1	1	0	0	0	0	0	1	
0C1	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	1	0	1	1	X	0	1	0	0	0	0	1	1	0	0	0	0	1	0	
0C2	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	1	1	
0C3	0	1	1	0	0	0	1	1	1	1	0	1	1	1	0	0	1	1	X	0	1	0	0	0	0	1	1	0	0	0	0	0	0		
0C4	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1	1	X	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	
XOR S_2, # DATA																																			
049	0	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0	1	X	0	1	0	0	0	0	1	1	0	0	0	1	0	1	
0C5	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	0	1	1	0	0	0	1	1	0	
0C6	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	1	0	1	1	X	0	1	0	0	0	0	1	1	0	0	0	1	1	1	
0C7	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0	0	0	
0C8	0	1	1	1	1	0	1	0	1	1	0	1	1	1	0	0	0	1	1	X	0	1	0	0	0	0	1	1	0	0	1	0	0	1	
0C9	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	0	1	1	X	0	1	0	0	0	0	0	0	0	0	0	0	0	1	
OR S_2, # DATA																																			
04A	0	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0	1	X	0	1	0	0	0	0	1	1	0	1	0	0	0	0	
0LO	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	0	1	1	0	1	0	0	0	1		
0D1	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	1	0	1	1	X	0	1	0	0	0	0	1	1	0	1	0	0	1	0		
0D2	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	1	1	
0D3	0	1	1	0	1	1	1	0	1	1	0	1	1	1	0	0	0	1	1	X	0	1	0	0	0	0	1	1	0	1	0	1	0	0	
0D4	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	0	1	1	X	0	1	0	0	0	0	0	0	0	0	0	0	0	1		
AND S_2, # DATA																																			
04B	0	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0	1	X	0	1	0	0	0	0	1	1	0	1	0	1	0	1	
0D5	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	0	1	1	0	1	0	1	1	0		
0D6	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	1	0	1	1	X	0	1	0	0	0	0	1	1	0	1	0	1	1	1		
0D7	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	1	1	0	1	1	0	0	0	
0D8	0	1	1	1	0	0	1	0	1	1	0	1	1	1	0	0	0	1	1	X	0	1	0	0	0	0	1	1	0	1	1	0	0	1	
0D9	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	0	1	1	X	0	1	0	0	0	0	0	0	0	0	0	0	0	1		
SR S_2, # DATA																																			
0AC	0	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0	1	X	0	1	0	0	0	0	1	1	1	0	0	0	0	0	
0E0	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	0	1	1	1	0	0	0	0	1		
0E1	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	1	0	1	1	X	0	1	0	0	0	0	1	1	1	0	0	0	1	0		
0E2	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	1	1	
0E3	0	1	1	0	0	0	1	1	1	1	0	1	1	1	0	0	1	1	X	0	1	0	0	0	0	1	1	1	0	0	0	1	0		
0E4	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	0	1	1	X	0	1	0	0	0	0	0	0	0	0	0	0	0	1		

MICRO PROGRAM	2901 CONTROL				CONTROL BIT													NEXT ADDRESS OF MICRO PROGRAM			OPERATION
	I_8 I_7 I_6	I_5 I_4 I_3	I_2 I_1 I_0	16 BUS S1	16 BUS S2	MEMRQ	MAR CG4	IR CG1	2901 CG3	DATA I/P CG2	C_N 2901	OE BUFF 2901	E DATA BUF*DAS	O/P CTRL DATA BUF	HALT	EJECT MEM/IO	$D_{11}D_{10}$ D_9 D_8	D_7 D_6 D_5 D_4	D_3 D_2 D_1 D_0	
RR S_2, # DATA																					
0 4 F	0 1 1	0 0 0	0 1 1	1	1	1	1	1	0	1	0	0	1	X	0	1	0 0 0 0	1 1 1 0	0 1 0 1		
0 E 5	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 1 1 0	0 1 1 0		
0 E 6	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 1 1 0	0 1 1 1		
0 E 7	0 1 1	0 0 0	0 1 1	1	1	0	0	1	1	0	1	0	0	0	0	1	0 0 0 0	1 1 1 0	1 0 0 0		
0 E 8	0 1 1	0 0 0	1 1 1	1	0	0	1	1	1	0	0	1	1	X	0	1	0 0 0 0	1 1 1 0	1 0 0 1		
0 E 9	1 0 1	0 0 0	0 1 1	1	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 0	1 1 1 0	1 0 1 0		
0 E A	X X X	X X X	X X X	1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1		
BL S_2, (S_1)																					
0 E 0	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 0	1 1 1 1	0 0 0 0	ดึงค่าใน BUS$_1$	
0 E 0	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 0	1 1 1 1	0 0 0 1	(B+0) → B. ไปที่ DATA MEMORY	
0 E 1	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 1 1 1	0 0 1 0	DATA ที่มี S_1 → MAR	
0 E 2	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 1 1 1	0 0 1 1	สั่ง RD, MEMRQ	
0 E 3	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1	0 0 0 0	1 1 1 1	0 1 0 0	DATA ที่ MEM (ซึ่งอยู่ใน S_1) ออกทาง D_{15}	
0 E 4	0 1 1	0 0 0	1 1 1	1	0	0	1	1	1	0	0	1	0	0	0	1	0 0 0 0	1 1 1 1	0 1 0 1	ค่าที่ออกทาง ROTAIL → S_2	
0 E 5	1 1 1	0 0 0	0 1 1	1	0	1	1	1	1	0	0	1	1	X	0	1	0 0 0 0	1 1 1 1	0 1 1 0	CONTROL 2901	
0 E C	0 0 1	0 0 0	0 1 1	1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	กลับไป FETCH ใหม่	
INC S_2, (S_1)																					
0 E 1	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 0	1 1 1 1	0 1 1 1	ดึงค่าใน BUS$_1$	
0 E 7	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 0	1 1 1 1	1 0 0 0	(B+0) → B. ไปที่ DATA MEMORY	
0 E 8	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	1 1 1 1	1 0 0 1	DATA ที่มี S_1 → MAR	
0 E 9	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	0	0	0	1	0 0 0 0	1 1 1 1	1 0 1 0	สั่ง RD, MEMRQ	
0 E A	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1	0 0 0 0	1 1 1 1	1 0 1 1	DATA ที่ MEM (ซึ่งอยู่ใน S_1) ออกทาง D_{15}	
0 E B	0 1 1	0 0 0	1 1 1	1	0	1	1	1	0	0	1	1	1	X	0	1	0 0 0 0	1 1 1 1	1 1 0 0	CONTROL 2901	
0 E C	X X X	X X X	X X X	1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	กลับไป FETCH ใหม่	
DEC S_2, (S_1)																					
0 E 2	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	0 0 0 0	0 0 0 1		
1 0 0	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	0 0 0 0	0 0 1 0		
1 0 1	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	0 0 0 0	0 0 1 1		
1 0 2	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	0	0	0	1	0 0 0 1	0 0 0 0	0 1 0 0		
1 0 3	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1	0 0 0 1	0 0 0 0	0 1 0 1		
1 0 4	0 1 1	0 1 0	1 1 1	1	0	1	1	1	0	0	0	1	1	X	0	1	0 0 0 1	0 0 0 0	0 1 1 0		
1 0 5	X X X	X X X	X X X	1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 1	0 0 0 0	0 1 1 1		

MICRO PROGRAM	2901 CONTROL			CONTROL BIT														NEXT ADDRESS OF MICRO PROGRAM			OPERATION
	I_8 I_7 I_6	I_5 I_4 I_3	I_2 I_1 I_0	16 to 2901 S1	26 to 2901 S2	$\overline{\text{MEMRQ}}$	MAR CG4	IR CG1	2901 CG3	DATA I/P CG2	C_N 2901	$\overline{OE}$ BUFF 2901	$\overline{E}$ DATA BUF-245	O/P CTRL DATA BUF	HALT	SELECT MEM/IO	D_{11} D_{10} D_9 D_8	D_7 D_6 D_5 D_4	D_3 D_2 D_1 D_0	
SL S_2, (S_1)																					
053	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1		0 0 0 1	0 0 0 0	0 1 1 0	
106	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1		0 0 0 1	0 0 0 0	0 1 1 1	
107	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1		0 0 0 1	0 0 0 0	1 0 0 0	
108	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	0	0	0	1		0 0 0 1	0 0 0 0	1 0 0 1	
109	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1		0 0 0 1	0 0 0 0	1 0 1 0	
10A	1 1 1	0 0 0	1 1 1	1	0	1	1	1	1	0	0	1	1	X	0	1		0 0 0 1	0 0 0 0	1 0 1 1	
10B	0 0 1	0 0 0	0 1 1	1	0	1	1	1	0	1	0	1	1	X	0	1		0 0 0 0	0 0 0 0	0 0 0 1	
SUB S_2, (S_1)																					
054	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1		0 0 0 1	0 0 0 1	0 0 0 0	
110	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1		0 0 0 1	0 0 0 1	0 0 0 1	
111	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1		0 0 0 1	0 0 0 1	0 0 1 0	
112	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	0	0	0	1		0 0 0 1	0 0 0 1	0 0 1 1	
113	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1		0 0 0 1	0 0 0 1	0 1 0 0	
114	0 1 1	0 0 1	1 0 1	1	0	1	1	1	0	0	1	1	1	X	0	1		0 0 0 1	0 0 0 1	0 1 0 1	
115	X X X	X X X	X X X	1	0	1	1	1	1	1	0	1	1	X	0	1		0 0 0 0	0 0 0 0	0 0 0 1	
SBC S_2, (S_1)																					
055	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1		0 0 0 1	0 0 0 1	0 1 1 0	
116	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1		0 0 0 1	0 0 0 1	0 1 1 1	
117	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1		0 0 0 1	0 0 0 1	1 0 0 0	
118	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	0	0	0	1		0 0 0 1	0 0 0 1	1 0 0 1	
119	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1		0 0 0 1	0 0 0 1	1 0 1 0	
11A	0 1 1	0 0 1	1 0 1	1	0	1	1	1	0	0	0	1	1	X	0	1		0 0 0 1	0 0 0 1	1 0 1 1	
11B	X X X	X X X	X X X	1	0	1	1	1	1	1	0	1	1	X	0	1		0 0 0 0	0 0 0 0	0 0 0 1	
ADD S_2, (S_1)																					
056	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1		0 0 0 1	0 0 1 0	0 0 0 0	
120	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1		0 0 0 1	0 0 1 0	0 0 0 1	
121	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1		0 0 0 1	0 0 1 0	0 0 1 0	
122	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	0	0	0	1		0 0 0 1	0 0 1 0	0 0 1 1	
123	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1		0 0 0 1	0 0 1 0	0 1 0 0	
124	0 1 1	0 0 0	1 0 1	1	0	1	1	1	0	0	0	1	1	X	0	1		0 0 0 1	0 0 1 0	0 1 0 1	
125	X X X	X X X	X X X	1	0	1	1	1	1	1	0	1	1	X	0	1		0 0 0 0	0 0 0 0	0 0 0 1	

MICRO PROGRAM	2901 CONTROL I_8 I_7 I_6	I_5 I_4 I_3	I_2 I_1 I_0	16 2B S1	26 2B S2	MEMRQ	MAR CG4	IR CG1	2901 CG3	DATA I/P CG2	C_N 2901	OB BUFF 2901	DATA BUF-245	O/P CTRL DATA BUF	HALT	SELECT MEM/IO	NEXT ADDRESS OF MICRO PROGRAM D_{11} D_{10} D_9 D_8	D_7 D_6 D_5 D_4	D_3 D_2 D_1 D_0	OPERATION
ADC S_2, (S_1)																			
057	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	0 0 1 0	0 1 1 0	
126	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	0 0 1 0	0 1 1 1	
127	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	0 0 1 0	1 0 0 0	
128	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	0	0	0	1	0 0 0 1	0 0 1 0	1 0 0 1	
129	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1	0 0 0 1	0 0 1 0	1 0 1 0	
12A	0 1 1	0 0 0	1 0 1	1	0	1	1	1	0	0	1	1	1	X	0	1	0 0 0 1	0 0 1 0	1 0 1 1	
LD S_2, (S_1)																			
058	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	0 0 1 1	0 0 0 0	
130	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	0 0 1 1	0 0 0 1	
131	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	0 0 1 1	0 0 1 0	
132	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	0	0	0	1	0 0 0 1	0 0 1 1	0 0 1 1	
133	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1	0 0 0 1	0 0 1 1	0 1 0 0	
134	0 1 1	0 0 0	1 1 1	1	0	1	1	1	0	0	1	1	X	0	1		0 0 0 1	0 0 1 1	0 1 0 1	
135	X X X	X X X	X X X	1	0	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	
XOR S_2, (S_1)																			
059	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	0 0 1 1	0 1 1 0	
136	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	0 0 1 1	0 1 1 1	
137	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	0 0 1 1	1 0 0 0	
138	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	0	0	0	1	0 0 0 1	0 0 1 1	1 0 0 1	
139	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1	0 0 0 1	0 0 1 1	1 0 1 0	
13A	0 1 1	1 1 0	1 0 1	1	0	1	1	1	0	0	0	1	1	X	0	1	0 0 0 1	0 0 1 1	1 0 1 1	
13B	X X X	X X X	X X X	1	0	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	
OR S_2, (S_1)																			
05A	0 1 1	0 0 0	0 1 1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	0 1 0 0	0 0 0 0	
140	0 1 1	0 0 0	0 1 1	0	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	0 1 0 0	0 0 0 1	
141	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	0 1 0 0	0 0 1 0	
142	X X X	X X X	X X X	1	1	0	0	1	1	1	0	1	0	0	0	1	0 0 0 1	0 1 0 0	0 0 1 1	
143	X X X	X X X	X X X	1	1	0	0	1	1	0	0	1	0	0	0	1	0 0 0 1	0 1 0 0	0 1 0 0	
144	0 1 1	0 1 1	1 0 1	1	0	1	1	1	0	0	0	1	1	X	0	1	0 0 0 1	0 1 0 0	0 1 0 1	
145	X X X	X X X	X X X	1	0	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	

MICRO PROGRAM	2901 CONTROL I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	CONTROL 1$\bar{G}$ 2901 S_1	CONTROL 2$\bar{G}$ 2901 S_2	MEMRQ	MAR CG4	IR C61	2901 C63	DATA I/P C62	C_N 2901	$\overline{OE}$ BUFF 2901	$\bar{E}$ DATA BUF 245	O/P CTRL DATA BUF	HALT	RESET MEM/IO	NEXT ADDRESS OF MICRO PROGRAM D_{11} D_{10} D_9 D_8	D_7 D_6 D_5 D_4	D_3 D_2 D_1 D_0	OPERATION
AND S_2, C_1)																										
025B	0	1	1	0	0	0	0	1	1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	0 1 0 0	0 1 1 0	
1466	0	1	1	0	0	0	0	1	1	0	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	0 1 0 0	0 1 1 1	
1477	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	0 1 0 0	1 0 0 0	
1488	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	1	0	1	0	0	0	1	0 0 0 1	0 1 0 0	1 0 0 1	
1499	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	0	0	1	0	0	0	1	0 0 0 1	0 1 0 0	1 0 1 0	
14AA	0	1	1	1	0	0	1	0	1	1	0	1	1	1	0	0	0	1	1	X	0	1	0 0 0 1	0 1 0 0	1 0 1 1	
14BB	X	X	X	X	X	X	X	X	X	1	0	1	1	1	1	0	1	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	
SR S_2, C_1)																										
025C	0	1	1	0	0	0	0	1	1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	0 1 0 1	0 0 0 0	
1500	0	1	1	0	0	0	0	1	1	0	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	0 1 0 1	0 0 0 1	
1511	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	0 1 0 1	0 0 1 0	
1522	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	1	0	1	0	0	0	1	0 0 0 1	0 1 0 1	0 0 1 1	
1533	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	0	0	1	0	0	0	1	0 0 0 1	0 1 0 1	0 1 0 0	
1544	1	0	1	0	0	0	1	1	1	1	0	1	1	1	0	0	1	1	X	0	1	0 0 0 1	0 1 0 1	0 1 0 1		
1555	X	0	1	0	0	0	0	1	1	1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 0	0 0 0 1	0 0 0 1	
IN S_2, C_1)																										
025D	0	1	1	0	0	0	0	1	1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	0 1 0 1	0 1 1 0	ที่อยู่ 1$\bar{G}$ S_1
1566	0	1	1	0	0	0	0	1	1	0	1	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	0 1 0 1	0 1 1 1	(B+Ø)→B ที่ DATA และที่ MAR
1577	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	0 1 0 1	1 0 0 0	DATA ที่ S_1 ADR MAR → ADLR. FOR
1588	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	0	0	0	0	0 0 0 1	0 1 0 1	1 0 0 1	ที่ว่าง IORQ
1599	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	0	0	1	0	0	0	0	0 0 0 1	0 1 0 1	1 0 1 0	DATA ที่ I/O ที่อยู่ IORQ S_1 ... D_{11}
15AA	0	1	1	0	0	0	1	1	1	1	0	1	1	1	0	0	0	1	1	X	0	1	0 0 0 1	0 1 0 1	1 0 1 1	CONTROL 2901
15BB	X	X	X	X	X	X	X	X	X	1	0	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	ที่ถัดไป FETCH ที่แล้ว
RR S_2, C_1)																										
025E	0	1	1	0	0	0	0	1	1	0	1	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	0 1 1 0	0 0 0 0	
1600	0	1	1	0	0	0	0	1	1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	0 1 1 0	0 0 0 1		
1611	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	0 1 1 0	0 0 1 0		
1622	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	1	0	1	1	X	0	1	0 0 0 1	0 1 1 0	0 0 1 1		
1633	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	0	0	1	0	0	0	1	0 0 0 1	0 1 1 0	0 1 0 0		
1644	0	1	1	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	0	0	0	1	0 0 0 1	0 1 1 0	0 1 0 1	
1655	1	0	1	0	0	0	0	1	1	1	0	1	1	1	1	0	0	1	1	X	0	1	0 0 0 1	0 1 1 0	0 1 1 0	
1666	0	1	1	0	0	0	0	1	1	1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	

ADDRESS MICRO PROGRAM	2901 CONTROL								CONTROL BIT												D_0 - D_{11} to 2909 NEXT ADDRESS OF MICRO PROGRAM												OPERATION
	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	$1\bar{G}$ to 2909 S1	$2\bar{G}$ to 2909 S2	$\overline{MEMRQ}$	MAR CG4	IR CG1	2901 CG3	DATA I/P CG2	C_N 2901	$\bar{OE}$ BUFF 2901	$\bar{E}$ DATA BUF-245	O/P CTRL DATA BUF	HALT	SELECT MEM/IO	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
RL [S_2]																																	
0 6 0	0	1	1	0	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	X	0	1	0	0	0	1	0	1	1	1	0	0	0	0	ที่อยู่ $2\bar{G}$ S_2
1 7 0	0	1	1	0	0	0	0	1	1	1	0	1	1	1	0	1	0	1	1	X	0	1	0	0	0	1	0	1	1	1	0	0	0	1	$(S_2+0) \rightarrow S_2$
1 7 1	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	1	0	1	1	1	0	0	1	0	ที่อยู่ใน S_2 ถูกส่งไป MAR
1 7 2	X	X	X	X	X	X	X	X	X	1	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	1	1	ให้ RD, MEMRQ, (ค่าในตำแหน่งที่ D_0-D_{11} ชี้อยู่) ถูกส่งไป S_2
1 7 3	0	1	1	0	0	0	1	1	1	1	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	1	0	1	1	1	0	1	0	0	ค่าที่ทำ ROTATE → S_2
1 7 4	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1	X	0	1	0	0	0	1	0	1	1	1	0	1	0	1	CONTROL 2901
1 7 5	0	0	1	0	0	0	0	1	1	1	1	0	1	1	0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	1	0	1	1	0	ให้ WR, MEMRQ
1 7 6	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	1	0	0	0	1	0	1	0	0	0	1	0	1	1	1	0	1	1	1	ค่าผลการ ROTATE → MEMORY
1 7 7	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	0	0	0	0	0	0	0	0	1	ตำแหน่งไป FETCH ต่อไป
INC [S_2]																																	
0 6 1	0	1	1	0	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	X	0	1	0	0	0	1	0	1	1	1	1	0	0	0	ที่อยู่ $2\bar{G}$ S_2
1 7 8	0	1	1	0	0	0	0	1	1	1	0	1	1	1	0	1	0	1	1	X	0	1	0	0	0	1	0	1	1	1	1	0	0	1	$(S_2+0) \rightarrow S_2$
1 7 9	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	1	0	1	1	1	1	0	1	0	ค่าใน S_2 ถูกส่งไป MAR
1 7 A	X	X	X	X	X	X	X	X	X	1	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	1	0	1	1	1	1	0	1	1	ให้ RD, MEMRQ, (ค่าในตำแหน่งที่ชี้อยู่) ถูกส่งไป S_2
1 7 B	0	1	1	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	1	X	0	1	0	0	0	1	0	1	1	1	1	1	0	0	CONTROL 2901
1 7 C	X	X	X	X	X	X	X	X	X	1	1	0	1	1	1	1	0	1	0	1	0	1	0	0	0	1	0	1	1	1	1	1	0	1	ให้ WR, MEMRQ
1 7 D	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	0	1	0	1	0	0	0	1	0	1	1	1	1	1	1	0	ข้อมูลที่ EXECUTE → MEMORY
1 7 E	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	0	0	0	0	0	0	0	0	1	ตำแหน่งไป FETCH ต่อไป
DEC [S_2]																																	
0 6 2	0	1	1	0	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	X	0	1	0	0	0	1	1	0	0	0	0	0	0	0	
1 8 0	0	1	1	0	0	0	0	1	1	1	0	1	1	1	0	1	0	1	1	X	0	1	0	0	0	1	1	0	0	0	0	0	0	1	
1 8 1	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	1	1	0	0	0	0	0	1	0	
1 8 2	X	X	X	X	X	X	X	X	X	1	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	1	1	
1 8 3	0	1	1	0	1	0	1	1	1	1	0	1	0	1	0	0	0	1	1	X	0	1	0	0	0	1	1	0	0	0	0	1	0	0	
1 8 4	X	X	X	X	X	X	X	X	X	1	1	0	1	1	1	1	0	1	0	1	0	1	0	0	0	1	1	0	0	0	0	1	0	1	
1 8 5	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	0	1	0	1	0	0	0	1	1	0	0	0	0	1	1	0	
1 8 6	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	0	0	0	0	0	0	0	0	1	

MICRO PROGRAM ADDRESS	2901 CONTROL I_8 I_7 I_6	I_5 I_4 I_3	I_2 I_1 I_0	CONTROL BIT: 1$\bar{G}$ ของ S1	2$\bar{G}$ ของ S2	$\overline{MEMRQ}$	MAR CG4	IR CG1	2901 CG3	DATA I/P CG2	C_N 2901	$\overline{OE}$ BUFF 2901	$\bar{E}$ DATA BUF 245	O/P CTRL DATA BUF	HALT	SELECT MEM / IO	NEXT ADDRESS OF MICRO PROGRAM: D_{11} D_{10} D_9 D_8	D_7 D_6 D_5 D_4	D_3 D_2 D_1 D_0	OPERATION
SL (S2)																				
0 6 3	0 1 1	0 0 0	0 1 1	1	0	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	1 0 0 0	0 1 1 1	
1 8 7	0 1 1	0 0 0	0 1 1	1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	1 0 0 0	1 0 0 0	
1 8 8	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	1 0 0 0	1 0 0 1	
1 8 9	X X X	X X X	X X X	1	0	0	0	1	1	0	0	1	0	0	0	1	0 0 0 1	1 0 0 0	1 0 1 0	
1 8 A	1 1 1	0 0 0	1 1 1	1	0	1	0	1	1	0	0	1	1	X	0	1	0 0 0 1	1 0 0 0	1 0 1 1	
1 8 B	0 0 1	0 0 0	0 1 1	1	1	0	1	1	0	0	0	1	0	1	0	1	0 0 0 1	1 0 0 0	1 1 0 0	
1 8 C	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	0	1	0	1	0 0 0 1	1 0 0 0	1 1 0 1	
1 8 D	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	
LD (S2), S1																				
0 6 8	0 1 1	0 0 0	0 1 1	1	0	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	1 0 0 1	0 0 0 0	เปิด 2$\bar{G}$ S2
1 9 0	0 1 1	0 0 0	0 1 1	1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	1 0 0 1	0 0 0 1	(Ø+S2) → S2
1 9 1	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	1 0 0 1	0 0 1 0	ค่าใน S2 → MAR
1 9 2	X X X	X X X	X X X	0	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	1 0 0 1	0 0 1 1	เปิด 1$\bar{G}$ S1 ให้ [illegible] B_0 B_3
1 9 3	0 1 1	0 0 0	0 1 1	1	0	0	0	1	0	1	0	1	1	X	0	1	0 0 0 1	1 0 0 1	0 1 0 0	(Ø + S1) → S2 .. CONTROL [illegible]
1 9 4	X X X	X X X	X X X	1	1	0	1	1	1	1	0	1	0	1	0	1	0 0 0 1	1 0 0 1	0 1 0 1	ส่ง WR , MEMRQ
1 9 5	X X X	X X X	X X X	1	1	1	1	1	1	1	0	1	0	1	0	1	0 0 0 1	1 0 0 1	0 1 1 0	ส่งผลการ LOAD → MEMORY
1 9 6	X X X	X X X	X X X	1	1	1	1	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	กลับไป FETCH ใหม่..
SR (S2)																				
0 6 C	0 1 1	0 0 0	0 1 1	1	0	1	1	0	1	1	0	1	1	X	0	1	0 0 0 1	1 0 0 1	0 1 1 1	
1 9 7	0 1 1	0 0 0	0 1 1	1	0	1	1	1	0	1	0	1	1	X	0	1	0 0 0 1	1 0 0 1	1 0 0 0	
1 9 8	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 1	1 0 0 1	1 0 0 1	
1 9 9	X X X	X X X	X X X	1	0	0	0	1	1	0	0	1	0	0	0	1	0 0 0 1	1 0 0 1	1 0 1 0	
1 9 A	1 0 1	0 0 0	1 1 1	1	0	1	0	1	1	0	0	1	1	X	0	1	0 0 0 1	1 0 0 1	1 0 1 1	
1 9 B	0 0 1	0 0 0	0 1 1	1	1	0	1	1	0	0	0	1	0	1	0	1	0 0 0 1	1 0 0 1	1 1 0 0	
1 9 C	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	0	1	0	1	0 0 0 1	1 0 0 1	1 1 0 1	
1 9 D	X X X	X X X	X X X	1	1	1	0	1	1	1	0	1	1	X	0	1	0 0 0 0	0 0 0 0	0 0 0 1	

ADDRESS MICRO PROGRAM	2901 CONTROL						CONTROL BIT													NEXT ADDRESS OF MICRO PROGRAM								OPERATION
	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	1G ของ S1	2G ของ S2	$\overline{MEMRQ}$	MAR CG4	IR C61	2901 C63	DATA I/P CG2	C_N 2901	$\overline{OE}$ BUFF 2901	$\overline{E}$ DATA BUF-245	O/P CTRL DATA BUF	HALT	SELECT MEM/IO	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
OUT (S2), S1																																			
06E	0	1	1	0	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	X	0	1	0	0	0	1	1	0	1	0	0	0	0	0	ที่ตา 2G S2
1A0	0	1	1	0	0	0	0	1	1	1	0	1	1	1	0	1	0	1	1	X	0	1	0	0	0	1	1	0	1	0	0	0	0	1	(S2+Φ) → S2 ; S2 = ALR.
1A1	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	1	1	0	1	0	0	0	1	0	ที่ใน S2 ก็ MAR → ALR
1A2	X	X	X	X	X	X	X	X	X	0	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	1	1	0	1	0	0	0	1	1	ที่ตา 1G S1 ... S1 ...
1A3	0	1	1	0	0	0	0	1	1	1	0	1	0	1	0	1	0	1	1	X	0	0	0	0	0	1	1	0	1	0	0	1	0	0	(S1+Φ) → S2 .. CONTRO
1A4	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	1	1	0	1	0	0	0	1	0	ให้ WR , IORQ
1A5	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	0	1	1	0	1	0	0	1	1	0	ส่งออกทาง OUT → PORT
1A6	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	0	0	0	0	0	0	0	0	1	กลับไป FETCH ใหม่...
RR (S2)																																			
06F	0	1	1	0	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	X	0	1	0	0	0	1	1	0	1	0	0	1	1	1	
1A7	0	1	1	0	0	0	0	1	1	1	0	1	1	1	0	1	0	1	1	X	0	1	0	0	0	1	1	0	1	0	1	0	0	0	
1A8	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	1	1	0	1	0	1	0	0	1	
1A9	X	X	X	X	X	X	X	X	X	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0	1	0	
1AA	0	1	1	0	0	0	1	1	1	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0	1	
1AB	1	0	1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1	X	0	1	0	0	0	1	1	0	1	0	1	1	0	0	
1AC	0	0	1	0	0	0	0	1	1	1	1	0	1	1	0	1	0	1	0	1	0	1	0	0	0	1	1	0	1	0	1	1	0	1	
1AD	X	X	X	X	X	X	X	X	X	1	1	0	0	1	1	1	0	1	0	1	0	1	0	0	0	1	1	0	1	0	1	1	1	0	
1AE	X	X	X	X	X	X	X	X	X	1	1	1	0	1	1	1	0	1	1	X	0	1	0	0	0	0	0	0	0	0	0	0	0	1	

1. MOTOROLA SEMICONDUCTOR INC. " M2900 BIPOLAR PROCESSER FAMILY. " , 1976.