전산274

결과 보고서

8조

전자계산학과	전자계산학과
9725002	9725018
강준명	오태호

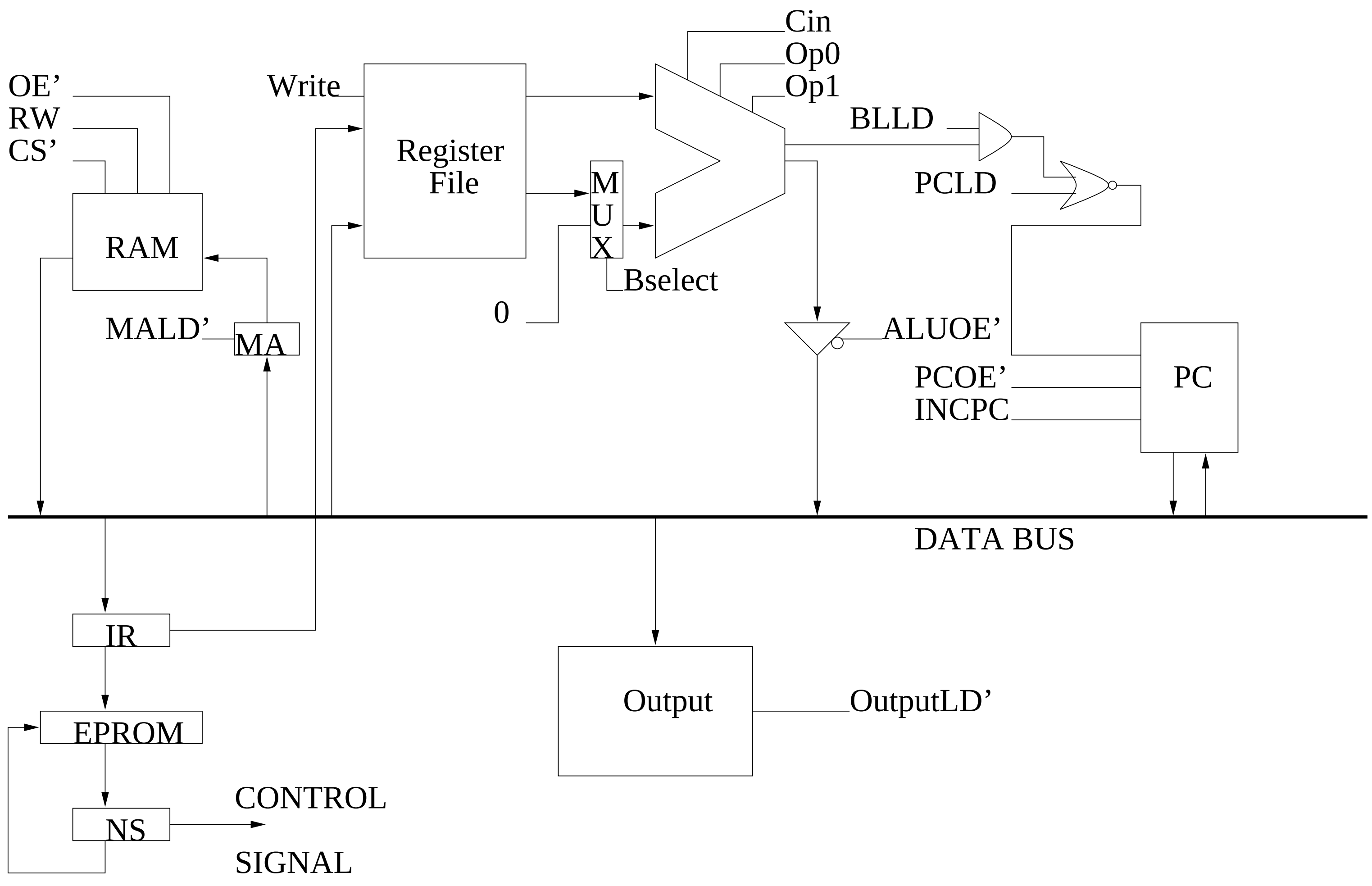
1998년 6월 22일

1 Instruction Set

$0001xxR_s$	OUT	OUT ← A
$0010R_dR_s$	ADDAB	A ← A + B
$0011R_dR_s$	SUBAB	A ← A − B
$0101R_dxx$	INCA	A ← A + 1
$0110R_dxx$	DECA	A ← A − 1
$0111R_dR_s$	ANDAB	A ← A and B
$1000R_dR_s$	ORAB	A ← A or B
$1001R_dR_d$	NOTA	A ← not A
$1011xxxx$ $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$	JUMP	PC ← $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$
$1100R_dR_s$ $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$	BL	IF A < B THEN PC ← $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$
$1101R_dR_s$	MOVA2B	B ← A
$1110R_dxx$ $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$	LOADA	A ← $MEMY_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$
$1111xxR_s$ $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$	STORA	A → $MEMY_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$

IN, COMP, XORAB Instruction Set은 처음에 시작할 때 넣을 계획이었지만 구현에 어려움이 있을 것 같아서 뺐다.

2 Block Diagram



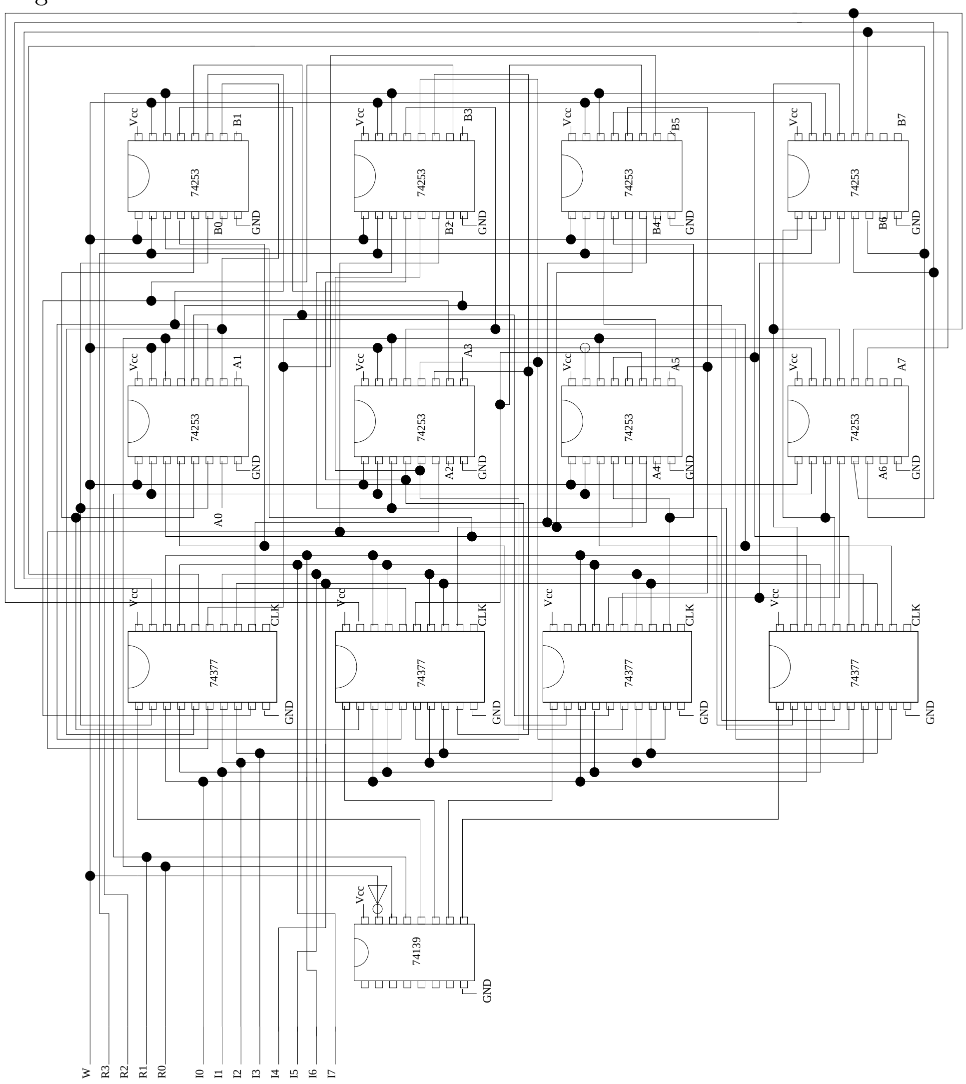
다른 자세한 것은 각각의 설명을 참고하면 되고 각각의 설명에 나와있지 않은 것은 다음과 같다.

- Bselect
  1을 더하거나 1을 뺄 때 B0~B7까지 0을 넣어 주어야 되는데 이때 사용한다.
- ALUOE'
  ALU의 출력을 Bus에 내보낼 것인지 내보내지 않을 것인지 결정할 때 사용한다.
- PCLD
  PCLD가 1이 되면 PC에 Bus로부터 입력받은 데이터를 Load한다.
- BLLD
  ALU에서 나온 결과가 음수이고 BLLD가 1이면 PC에 Bus로부터 입력받은 데이터를 Load한다.
- OutputLD'
  OutputLD'이 0이면 Output Register에 Bus에 있는 값을 Load한다.

3 회로도

아래의 회로도는 실제 구현된 회로의 회로도이다.

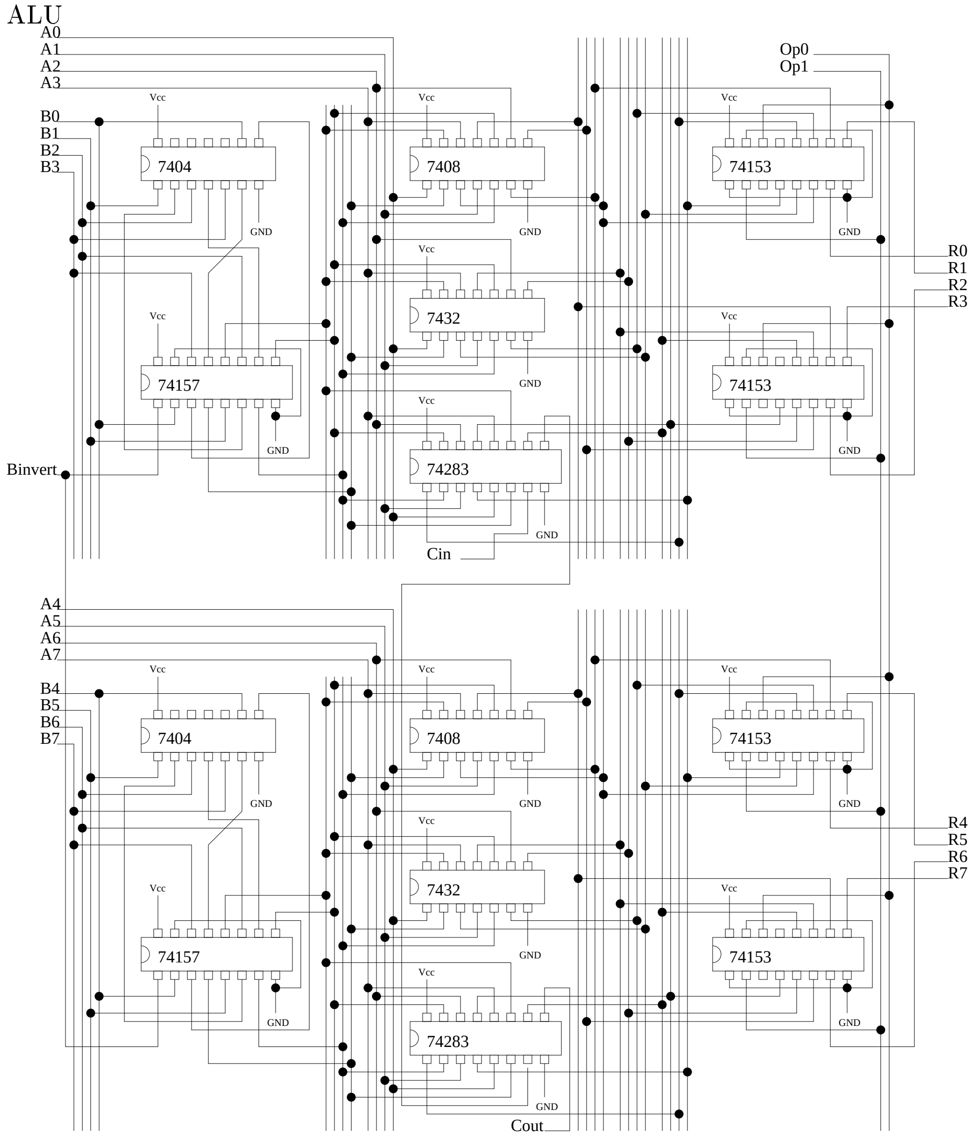
1. Register File



[1]

[1] 74139에 들어가는 input중에 보면 inverter를 하나 볼 수 있다. 이것은 회로도를 설계하면서 실수해서 read와 write가 동시에 일어나 이를 막기 위해 넣은 것이다. 실제로는 7404를 inverter로 사용했다.

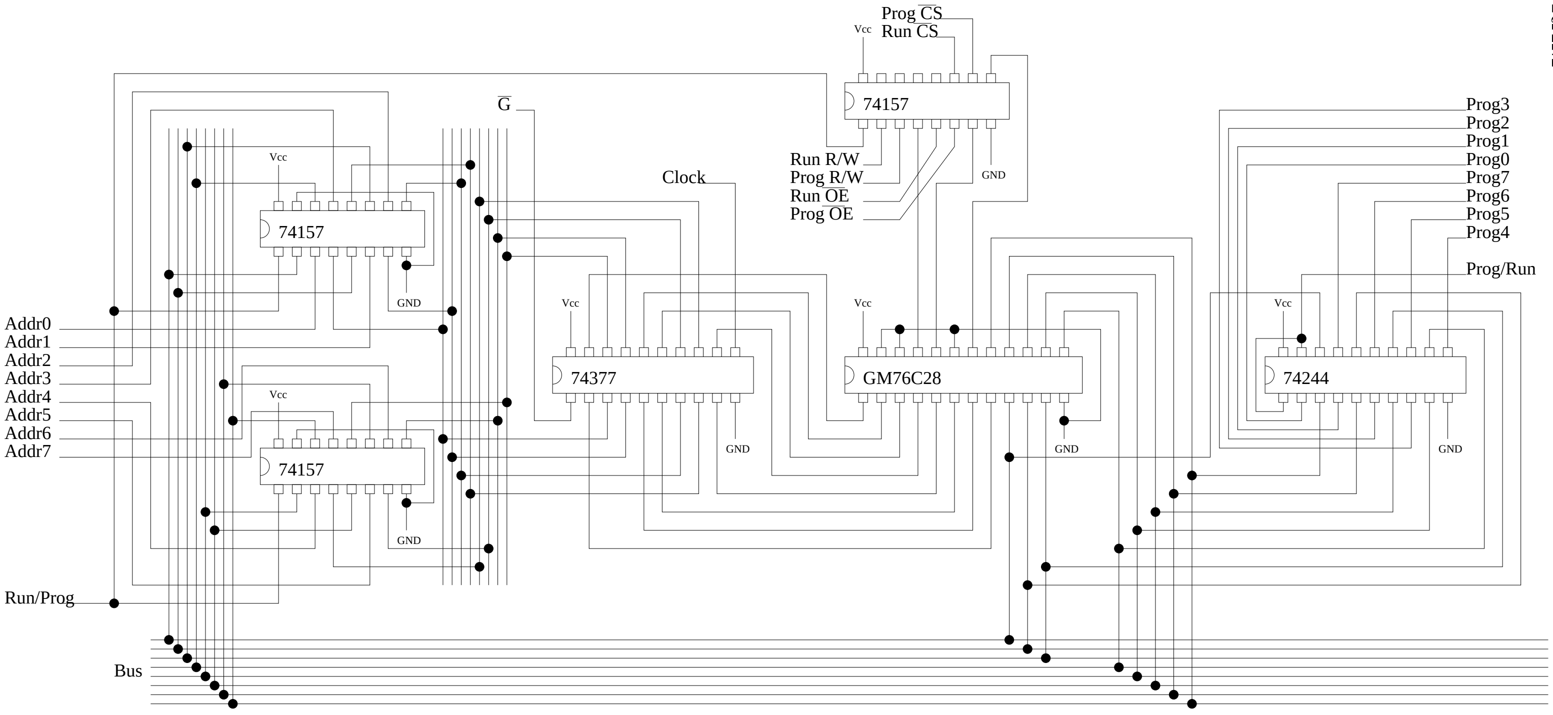
2. ALU



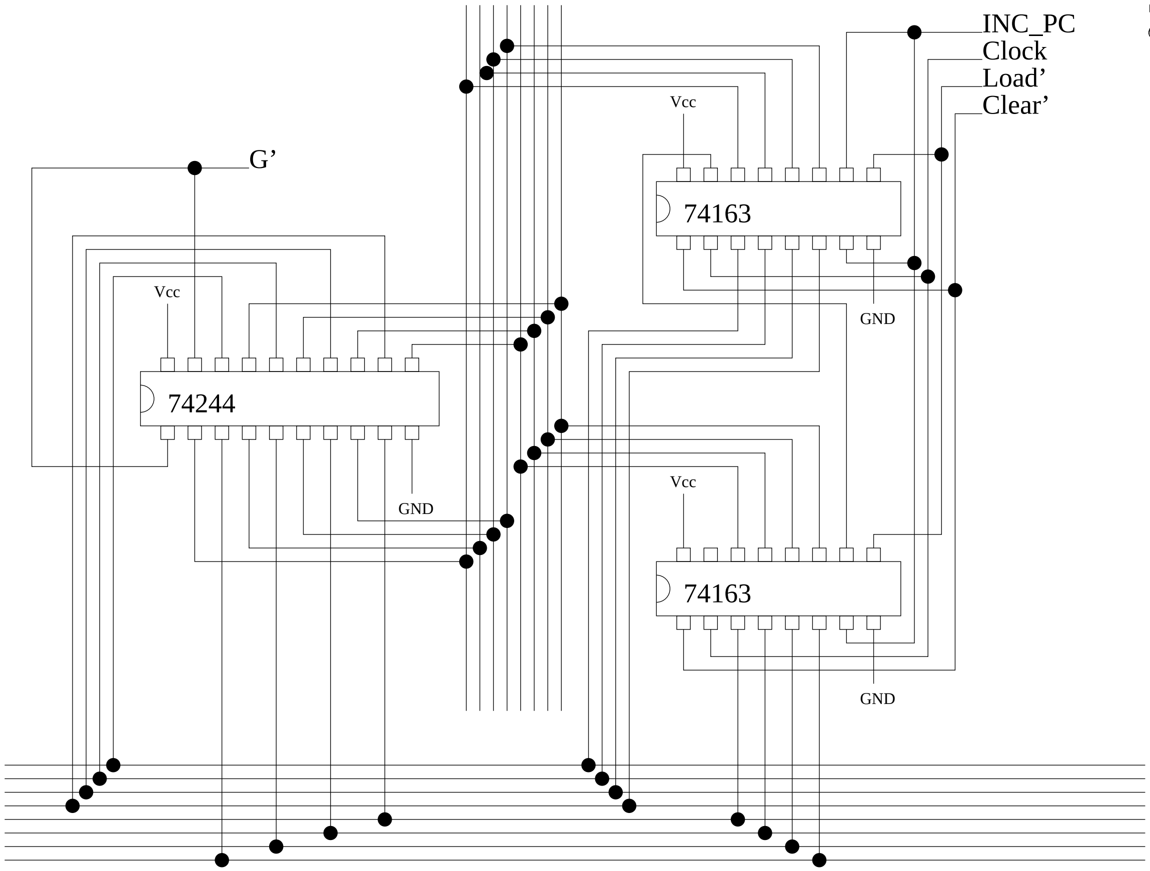
[2]

[2] ALU를 구현할 때 Op0와 Op1 둘 다 1일때는 don't care로 처리했지만 CLU에서는 Op0와 Op1이 둘 다 1일때 B0~B7의 값을 그대로 출력한다고 생각하고 만들었다. 그러므로 정상적으로 작동하게 하려면 4:1 MUX의 마지막에 B0~B7을 넣어주어야 한다.

3. RAM


Prog CS
Run CS
Vcc
74157
GND
Run R/W
Prog R/W
Run OE
Prog OE
G
Clock
74157
Vcc
GND
Addr0
Addr1
Addr2
Addr3
Addr4
Addr5
Addr6
Addr7
74157
Vcc
GND
74377
GM76C28
74244
Prog3
Prog2
Prog1
Prog0
Prog7
Prog6
Prog5
Prog4
Prog/Run
Run/Prog
Bus

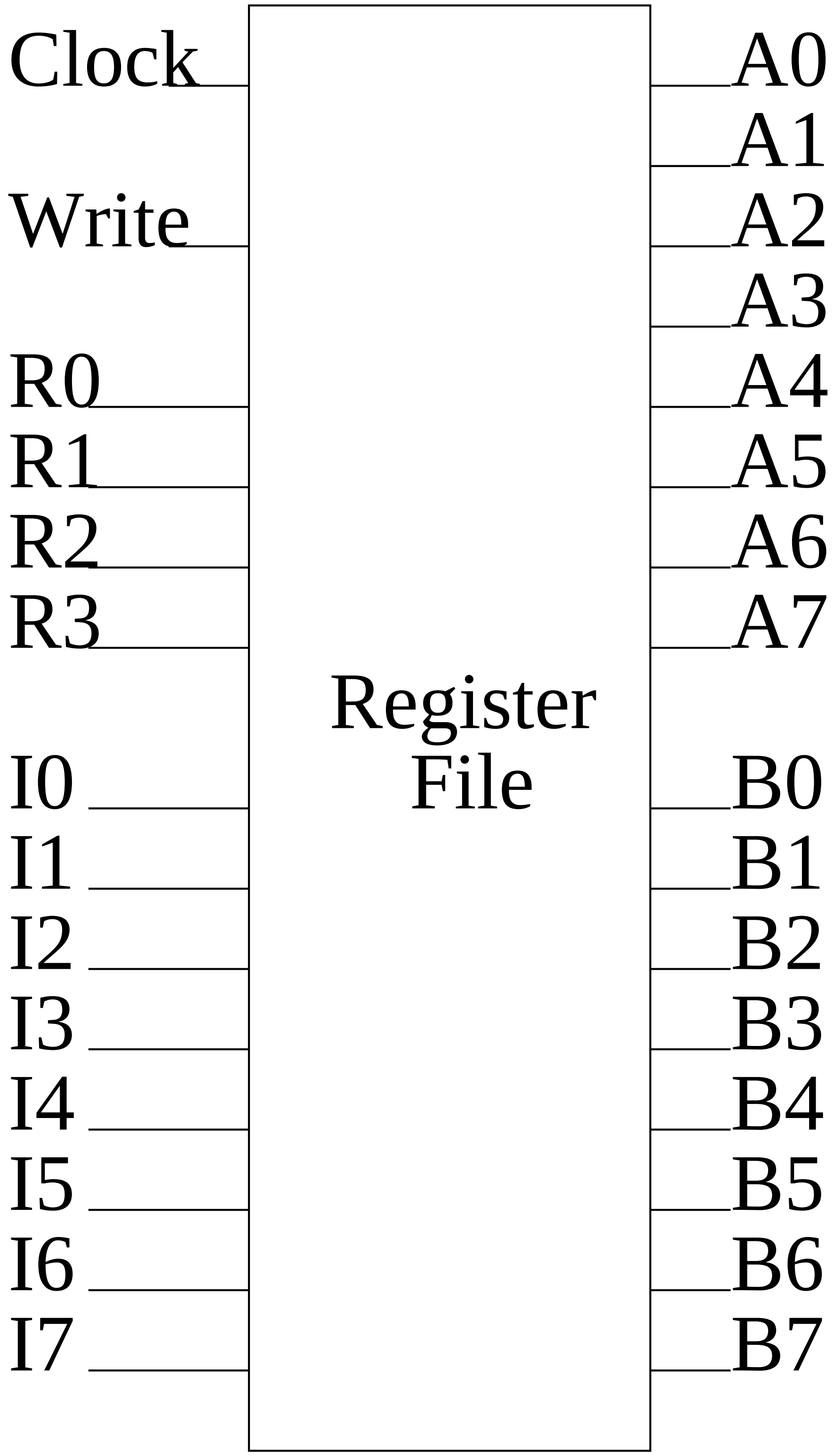
4. PC



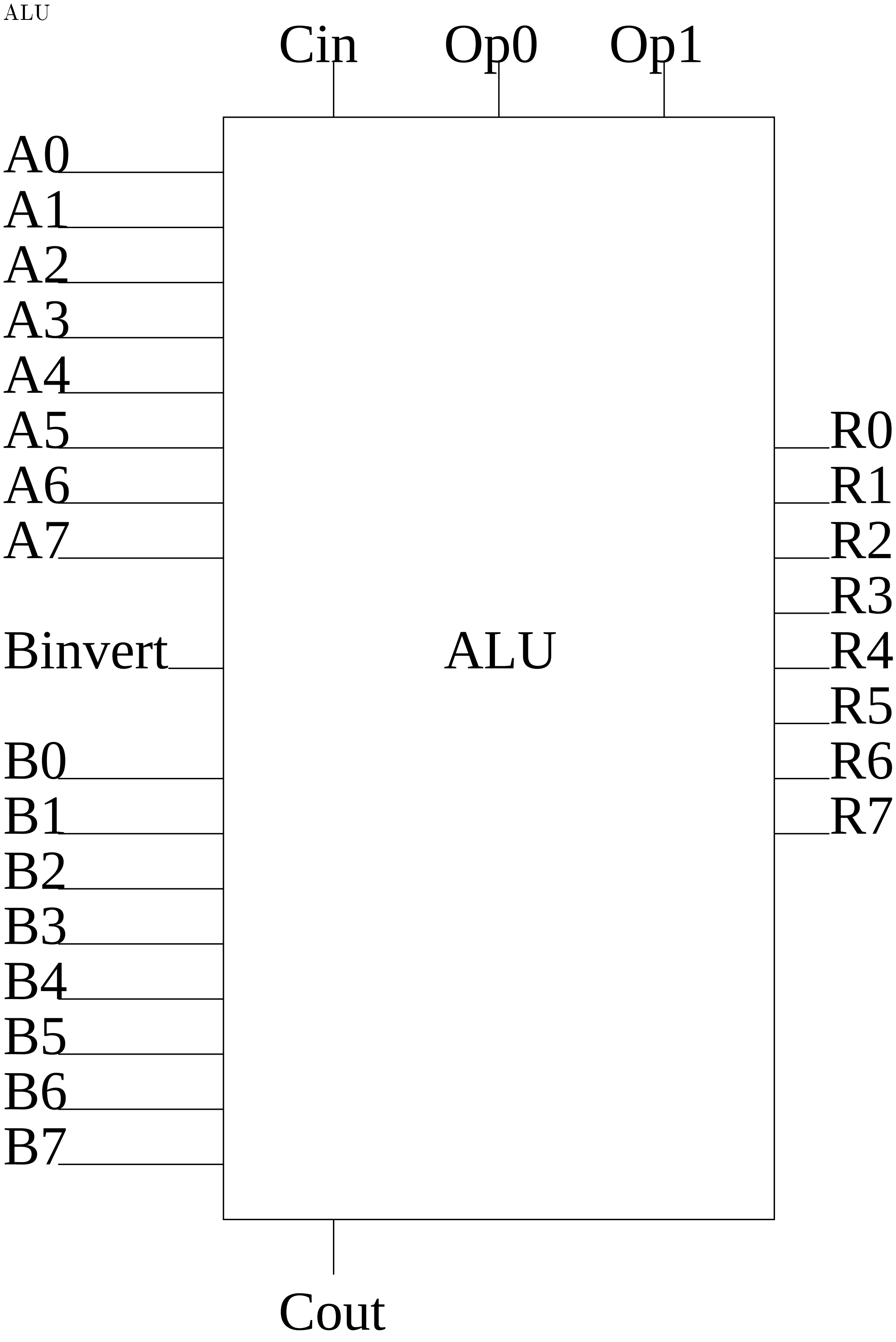
INC_PC
Clock
Load’
Clear’
G’
Vcc
74163
GND
Vcc
74244
GND
Vcc
74163
GND

4 각 Component의 모식도

1. Register File



- Write
  0일때 Read, 1일때 Write한다. (Clock에 의에 동기화된다.)
- R0,R1
  Write가 0일때는 A0~A7에 출력될 Register를 선택한다. Write가 1일때는 I0~I7으로부터 입력된 데이터를 Write할 Register를 선택한다.
- R2,R3
  Write가 0일때는 B0~B7에 출력될 Register를 선택한다. Write가 1일때는 don't care이다.
- I0~I7
  데이터를 입력한다.
- A0~A7
  데이터를 출력한다.
- B0~B7
  데이터를 출력한다.


Cin
Op0
Op1
A0
A1
A2
A3
A4
A5
A6
A7
Binvert
B0
B1
B2
B3
B4
B5
B6
B7
ALU
R0
R1
R2
R3
R4
R5
R6
R7
Cout

- Cin

  Add연산을 할 때 Carry in을 설정한다.

- Op0,Op1

Op0	Op1	연산
0	0	AND
0	1	ADD
1	0	OR
1	1	B

- Binvert

  B0˜B7의 값을 invert한다.

- A0˜A7

  데이터를 입력한다.

- B0˜B7

  데이터를 입력한다.

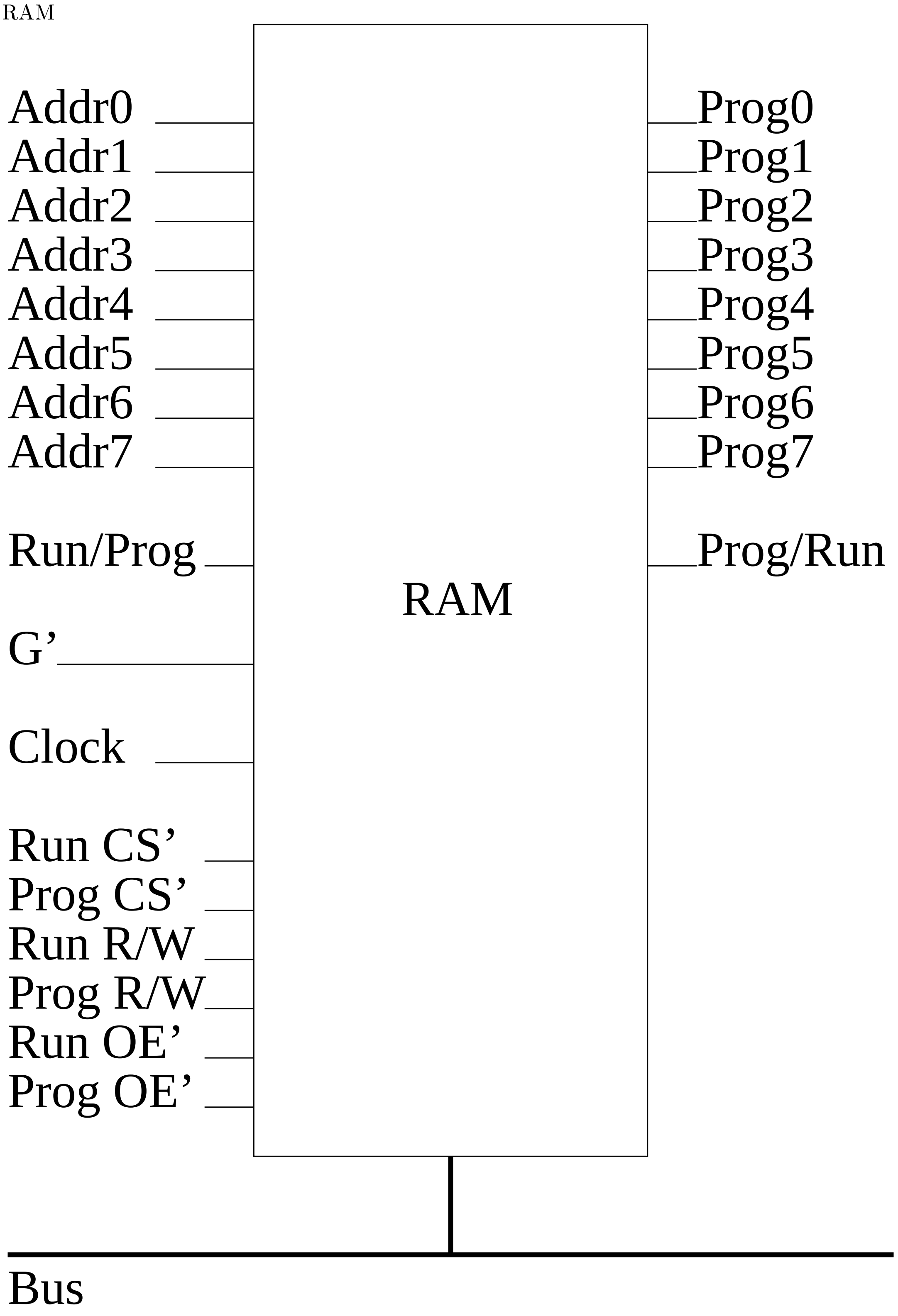
- R0˜R7

  데이터를 출력한다.

- Cout

  Add연산을 할 때 Carry out을 출력한다.

3. RAM


Addr0
Addr1
Addr2
Addr3
Addr4
Addr5
Addr6
Addr7
Run/Prog
G'
Clock
Run CS'
Prog CS'
Run R/W
Prog R/W
Run OE'
Prog OE'
RAM
Prog0
Prog1
Prog2
Prog3
Prog4
Prog5
Prog6
Prog7
Prog/Run
Bus

- Run/Prog
  Run/Prog가 0이면 주소를 Bus로부터 받아들이고 CS',R/W,OE'은 Run mode로부터 받아들인다. Run/Prog가 1이면 Addr0~Addr7로부터 주소를 받아들이고 CS',R/W,OE'은 Prog mode로부터 받아들인다.

- Prog/Run
  Prog/Run이 0이면 데이터를 Prog0~Prog1로부터 받아들이고 Prog/Run이 1이면 데이터를 Bus로부터 받아들인다.

- CS',RW,OE'

	Write	Read	NoOp
CS'	0	0	0
R/W	0	1	1
OE'	1	0	1

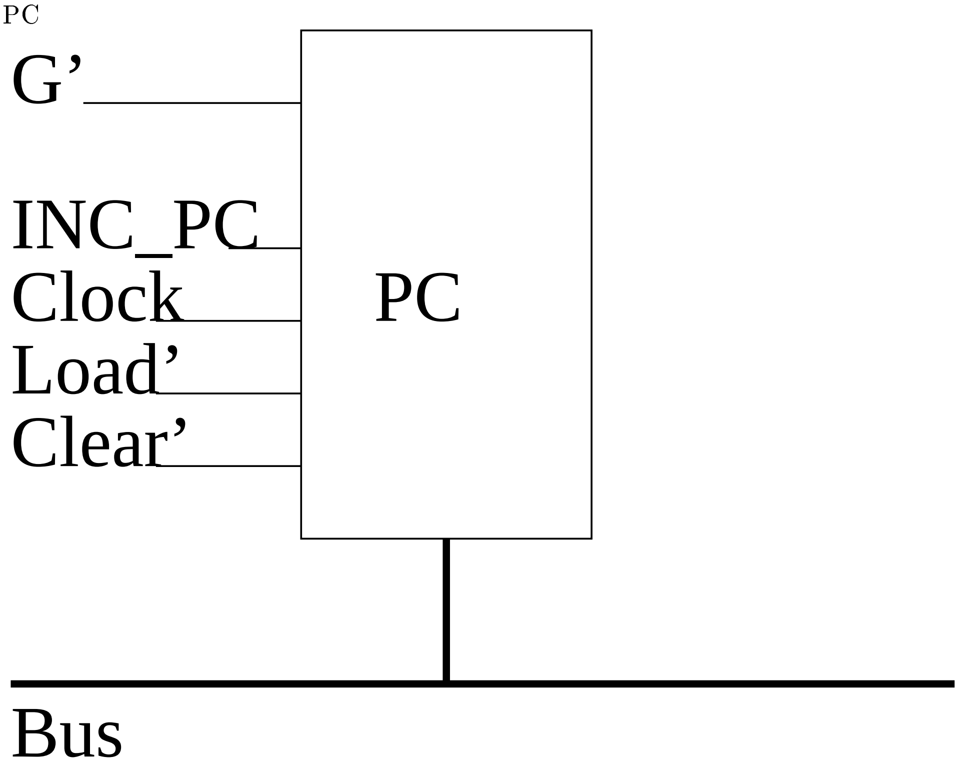
- G'[3]
  G'가 0이면 MA로 주소를 Load한다. G'가 1이면 MA로 주소를 Load하지 않는다.

- Addr0~Addr7
  주소를 입력한다.

- Prog0~Prog7
  데이터를 입력한다.

[3]Block Diagram에서는 MALD'이라고 표기했다.

4. PC



- G'
  G'가 0이면 Count값이 출력되고 G'가 1이면 출력이 되지 않는다.(하이 임피던스)
- INC_PC
  INC_PC가 0이면 Count를 멈추고 INC_PC가 1이면 Count값을 1증가시킨다.
- Load'
  Load'가 0이면 Bus로부터 Count값을 Load하고 Load'가 1이면 Load하지 않는다.
- Clear'
  Clear'가 0이면 Count를 0으로 초기화하고 Clear'가 1이면 Count를 초기화하지 않는다.

5 Instruction time event table

INSTRUCTION	T0	T1
ALL	MA←PC	IR←M[MA] PC←PC+1

INSTRUCTION	T2	T3	T4	T5	T6	T7	T8
0001 OUT	OUT←A						
0010 ADDAB	A←A+B						
0011 SUBAB	A←A−B						
0101 INCA	A←A+1						
0110 DECA	A←A−1						
0111 ANDAB	A←A∧B						
1000 ORAB	A←A∨B						
1001 NOTA	A←A’						
1011 JUMP	MA←PC	PC←M[MA]					
1100 BL	A←PC	A←A−B PC←PC+1	if A<0 PC←M[MA]				
1101 MOVE2B	B←A						
1110 LOADA	A←PC	MA←M[MA] PC←PC+1	A←M[MA]				
1111 STORA	A←PC	MA←M[MA] PC←PC+1	M[MA]←A				

6 State transition table

	Write	Cin	Op0	Op1	Binvert	Bselect	ALUOE'	MALD'	CS'	R/W	OE'	PCOE'	INCPC	PCLD	BLLD	OutputLD'
T0	0	X	X	X	X	X	1	0	0	1	1	0	0	0	0	1
T1	0	X	X	X	X	X	1	1	0	1	0	1	1	0	0	1
T2OUT	0	X	X	X	X	X	1	1	0	1	1	1	0	0	0	0
T2ADDAB	1	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1
T2SUBAB	1	1	0	1	1	0	0	1	0	1	1	1	0	0	0	1
T2INCA	1	1	0	1	0	1	0	1	0	1	1	1	0	0	0	1
T2DECA	1	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1
T2ANDAB	1	X	0	0	0	0	0	1	0	1	1	1	0	0	0	1
T2ORAB	1	X	1	0	0	0	0	1	0	1	1	1	0	0	0	1
T2NOTA	1	X	1	1	1	0	0	1	0	1	1	1	0	0	0	1
T2JUMP	0	X	X	X	X	X	1	0	0	1	1	0	0	0	0	1
T3JUMP	0	X	X	X	X	X	1	1	0	1	0	1	0	1	X	1
T2BL	0	X	X	X	X	X	1	0	0	1	1	0	0	0	0	1
T3BL	1	1	0	1	1	0	0	0	0	1	1	1	1	0	0	1
T4BL	0	X	X	X	X	X	1	1	0	1	0	1	0	0	1	1
T2MOVE2B	1	X	1	1	0	0	0	1	0	1	1	1	0	0	0	1
T2LOADA	0	X	X	X	X	X	1	0	0	1	1	0	0	0	0	1
T3LOADA	0	X	X	X	X	X	1	0	0	1	0	1	1	0	0	1
T4LOADA	1	X	X	X	X	X	1	1	0	1	0	1	0	0	0	1
T2STORA	0	X	X	X	X	X	1	0	0	1	1	0	0	0	0	1
T3STORA	0	X	X	X	X	X	1	0	0	1	0	1	1	0	0	1
T4STORA	1	X	1	1	0	0	0	1	0	0	1	1	0	0	0	1

7 Test example

- 1부터 10까지 더하는 프로그램 소스코드

```
LOAD R0,Zero
LOAD R1,Zero
LOAD R2,Ten
LOOP:
INC R1
ADD R0,R1
MOVE R3,R1
BL R3,R2 LOOP
JUMP EXIT
EXIT:
OUT R0
```

- 메모리에 저장되어 있는 형식

Address	Value
00000000	11100000
00000001	00001101
00000010	11100100
00000011	00001101
00000100	11101000
00000101	00001110
00000110	01010100
00000111	00100001
00001000	11011101
00001001	11001110
00001010	00000101
00001011	10110000
00001100	00001111
00001101	00000000
00001110	00001010
00001111	00010000

8 결론

이번 프로젝트를 하면서 컴퓨터의 구조에 대해 많은 것을 알게 되었다.