컴퓨터구조

이성원교수님

Project #0

학과 : 컴퓨터 정보 공학부

학번 : 2019202103

이름 : 이은비

제출 날짜 : 2023.04.05

<문제의 해석 및 해결방향>

<문제의 해석 및 기본 설명>

전체적인 프로세스 과정은 다음과 같습니다. Fetch -> Decode -> Exec 과정으로 프로젝트에서 제시한 MU0는 Processor로 16bit를 구현하고 있습니다. 즉, Memory(RAM)와 Processor내부의 레지스터가 16bit로 구성되어 있다는 것이고, RAM에서 reset과 clk신호에 따라 clk이 rising edge일 때 reset신호에 따라 opcode가 4bit이고 address는 12bit인 data를 Processor의 PC가 메모리(RAM)의 0번째 주소부터 데이터를 load하여 그 데이터를 IR로 전달합니다. 그리고 IR에서 splitter를 이용하여 그 데이터의 opcode(상위 4bit)를 불러내고 opcode를 분석하는 decode과정을 거쳐 각 Register에 일정한 signal을 주는 과정을 거치게 됩니다. 이후 일정한 signal을 입력 받은 Register는 그 signal에 맞는 일을 처리함으로써 Instruction이 Processor 내부에서 실행되는 execute과정을 거칩니다.

그리고 구체적으로 각 세부 Register에서 하는 일들을 간략하게 서술하면 다음과 같습니다.

IR은 Instruction Register의 약자로서 즉 명령어 레지스터입니다. 메모리에서 읽어온 데이터가 이곳에 저장되며 이곳에서 데이터의 opcode를 분석하여 명령어를 구분합니다. PC는 Program Counter Register의 약자로서 메모리에서 데이터를 address 순으로 순차적(PC+4로 업데이트 하는 과정)으로 불러오거나, 명령어에 따라서 명령어가 가리키는 주소의 값을 불러오기도 합니다. ACC는 Accumulator 의 약자로서 우리나라 말로는 누산기 입니다. 이것은 메모리에서 불러온 데이터 값이나 주소 값을 일시적으로 저장하여 다음 명령어가 실행되면 ACC에 저장된 데이터를 이용하여 산술이나 논리적인 연산을 할 수 있게 합니다. ALU는 Arithmetic Logic Unit의 약자로서 산술 논리 연산 장치라 불립니다. Accumulator에 저장된 값과 새로운 데이터 값을 받아서 그 두 값을 산술 논리적으로 연산하는 장치이며, PC에 저장된 주소의 값을 증가시키는 등 모든 opcode의 연산을 수행합니다. MUX : Multiplexer의 약자로서 두 개의 입력 데이터 값 중에서 한 가지만 출력합니다. Control Unit은 데이터에서 불러온 opcode에 따라서 각 Register에서 수행 해야 할 신호를 정해서 출력하는 소자입니다...

그리고 프로젝트 제안서에 있는 각 opcode에 대한 간단한 설명과 opcode에 따른 동작을 truth table과 함께 설명하면 다음과 같습니다,

LDA (Load) : 메모리 주소 S에 있는 데이터를 누산기(ACC 레지스터)에 적재한다.
STO (Store) : Accumulator의 값을 메모리에 저장한다.
ADD (Addition) : 메모리의 값과 누산기의 값을 더한 후 누산기에 저장한다.
SUB (Subtraction) : '누산기의 값 - 메모리'의 결과 값을 누산기에 저장한다.
JMP (Jump) : 다음에 처리할 명령어의 주소 값을 S로 점프(이동)시킨다.
JGE (Jump if Greater or Equal) : ACC값이 0보다 크거나 같다면 PC에 S를 대입시켜 현재 명령어 처리 주소값을 S로 점프한다.
JNE (Jump if Not Equal) : ACC가 0이 아니라면 PC에 S를 대입한다.
STP (Stop) : 중지한다.

MEMrq: Memory Request 신호는 메모리에 접근할 필요가 있을 때 수행됩니다. PC에 있는 다음 명령어 레지스터를 불러와 실행하기위해 상태가 fetch일 때 '1'이 됩니다. 즉, LDA, STO, ADD. SUB를 수행할 때 메모리를 접근하므로 '1'이 됩니다. 나머지 메모리 접근을 하지 않을 때 '0'이 됩니다..

RnW (Read. if Negative, Write)는 '1'이면 memory Read, '0' 이면 memory Write가 수행됩니다.

Asel, Bsel 은 MUX의 Sellect 즉 enable기능을 하는 신호이다. '0'이면 0의 입력신호를 출력시키고, '1'이면 1의 신호를 출력시킨다.

IRce, PCce, ACCce 는 각각 IR,PC,ACC의 register의 Enable 신호입니다.

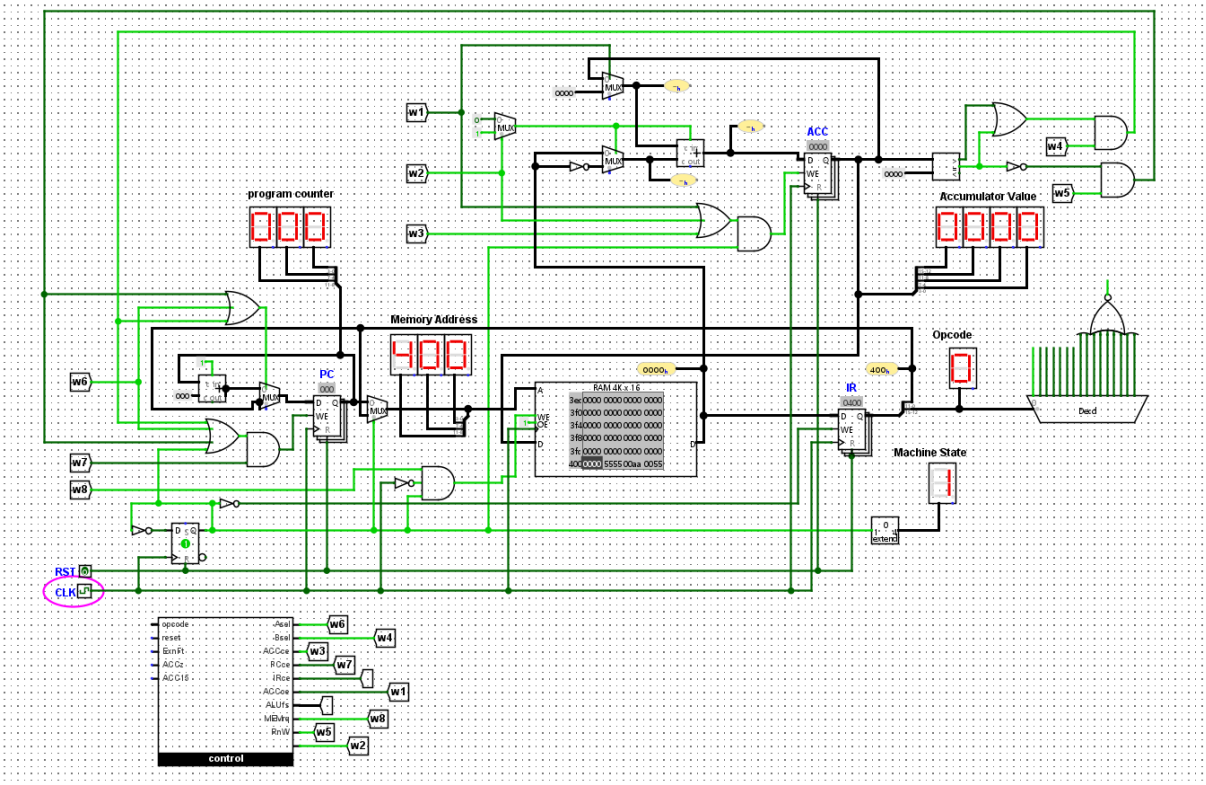
ALUop[1:0] (ALU Function Signal)는 ALU의 add, sub 등의 동작을 결정하는 신호입니다.

아래표는 input값(opcode,...)에 따른 output값은 다음 괄호 안의 값들을 가진 (Asel,Bsel,...) truth table입니다.

- 1 0 Collapse Duplicated Rows Show All Rows

256 of 256 rows shown

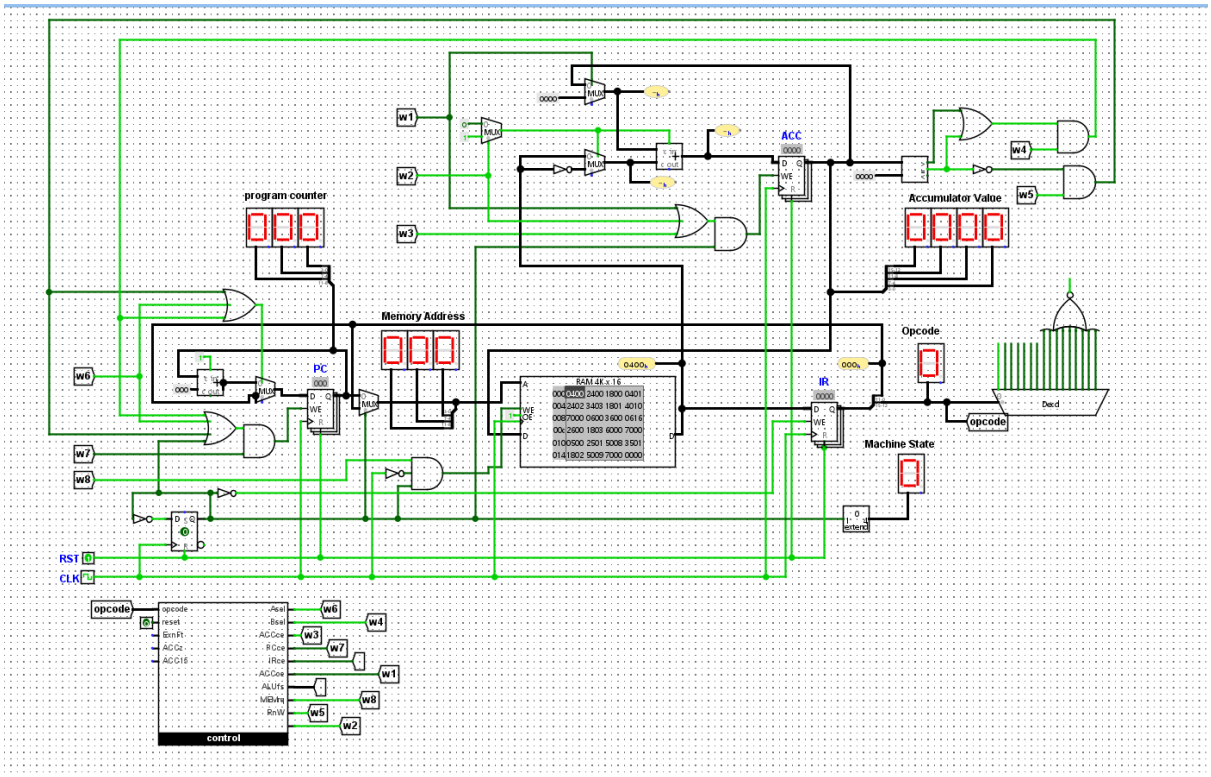
opcode[3..0]	reset	ExnFt	ACCz	ACC15	Asel	Bsel	ACCce	PCce	IRce	ACCoe	ALUfs[1..0]	MEMrq	RnW	exnft
0 0 0 0	0	0	0	0	1	1	1	0	0	0	0 0	1	1	1
0 0 0 0	0	0	0	1	1	1	1	0	0	0	0 0	1	1	1
0 0 0 0	0	0	1	0	1	1	1	0	0	0	0 0	1	1	1
0 0 0 0	0	0	1	1	1	1	1	0	0	0	0 0	1	1	1
0 0 0 0	0	1	0	0	0	0	0	1	1	0	0 0	1	1	0
0 0 0 0	0	1	0	1	0	0	0	1	1	0	0 0	1	1	0
0 0 0 0	0	1	1	0	0	0	0	1	1	0	0 0	1	1	0
0 0 0 0	0	1	1	1	0	0	0	1	1	0	0 0	1	1	0
0 0 0 0	1	0	0	0	0	0	1	1	1	0	0 0	1	1	0
0 0 0 0	1	0	0	1	0	0	1	1	1	0	0 0	1	1	0
0 0 0 0	1	0	1	0	0	0	1	1	1	0	0 0	1	1	0
0 0 0 0	1	0	1	1	0	0	1	1	1	0	0 0	1	1	0
0 0 0 0	1	1	0	0	0	0	1	1	1	0	0 0	1	1	0
0 0 0 0	1	1	0	1	0	0	1	1	1	0	0 0	1	1	0
0 0 0 0	1	1	1	0	0	0	1	1	1	0	0 0	1	1	0
0 0 0 0	1	1	1	1	0	0	1	1	1	0	0 0	1	1	0
0 0 0 1	0	0	0	0	1	-	0	0	0	1	0 0	1	0	1
0 0 0 1	0	0	0	1	1	-	0	0	0	1	0 0	1	0	1
0 0 0 1	0	0	1	0	1	-	0	0	0	1	0 0	1	0	1
0 0 0 1	0	0	1	1	1	-	0	0	0	1	0 0	1	0	1
0 0 0 1	0	1	0	0	0	0	0	1	1	0	0 0	1	1	0



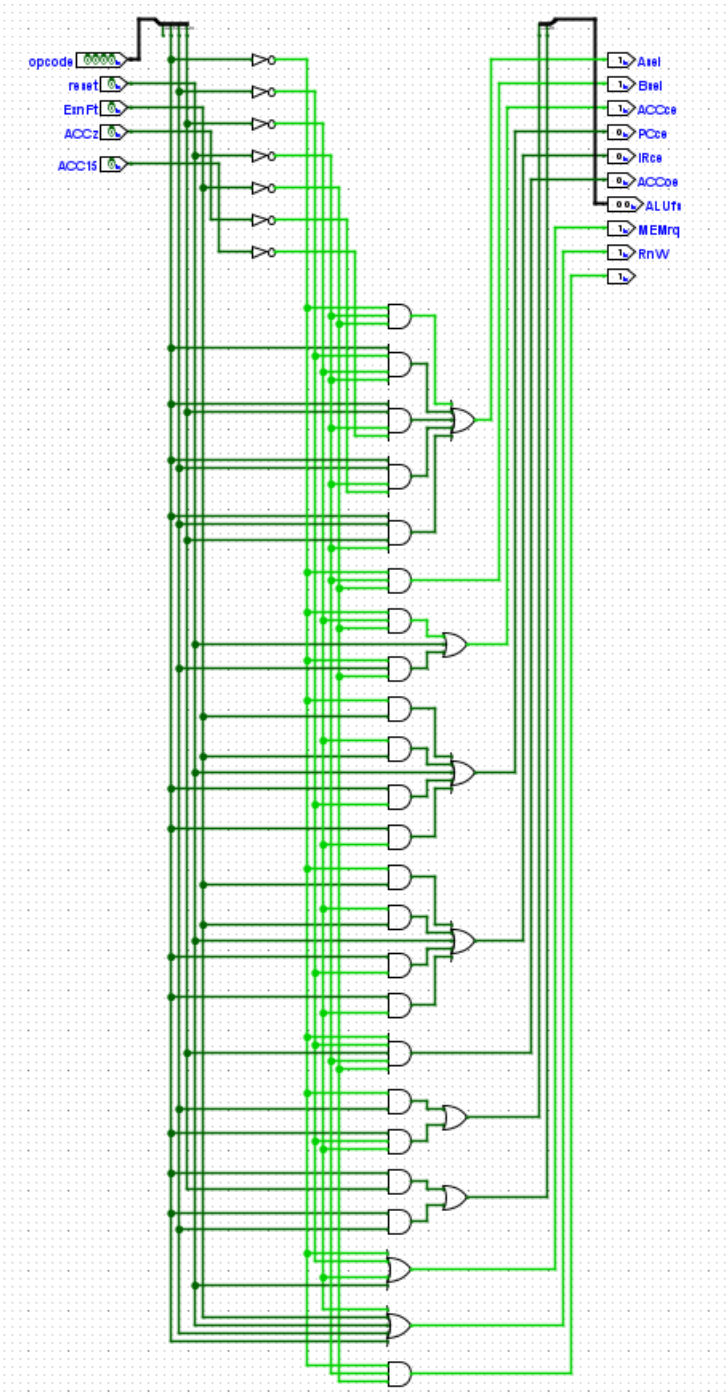
각각의 wire(tunnel)의 연결은 w1~w8까지 각각 ACCce, Ex/Ft, ACCce, Bsel, RnW, Asel, RCce, Memrq 으로 연결되었습니다. 근데 결과 opcode의 결과 까지는 하지 못한 것이 Machine state, Memory Address의 7 segmentation에서 확인한 것으로 Ram에 접근하여 처음data를 read하고 fetch상태가 된 것을 확인하였으나 program counter와 Accumulator Value 와 Opcode의 7 segmentation의 변화는 없어서 실행이 제대로 수행되었다고 판단할 수 없었습니다. 이때의 문제점을 생각해보면 현재 tunnel로 연결한 control unit의 output, 예를 들어 ALUop1, ALUop0와 같은 bits를 받아 ALU의 sub,add와 같은 opcode를 구분하여 수행하는 과정을 수행하는 unit이 있어야 할 것 같고, decode가 수행하는 것이 무엇인지 정확히 몰라서 연결을 마무리 짓지 못한부분, 그리고 w4,w5가 각각 JGE,JNE로 서로 나누어지는 상황 같은데 그에 wire 즉 tunnel연결을 제대로 수행되지 않은 것 같습니다.

<설계 의도와 방법>

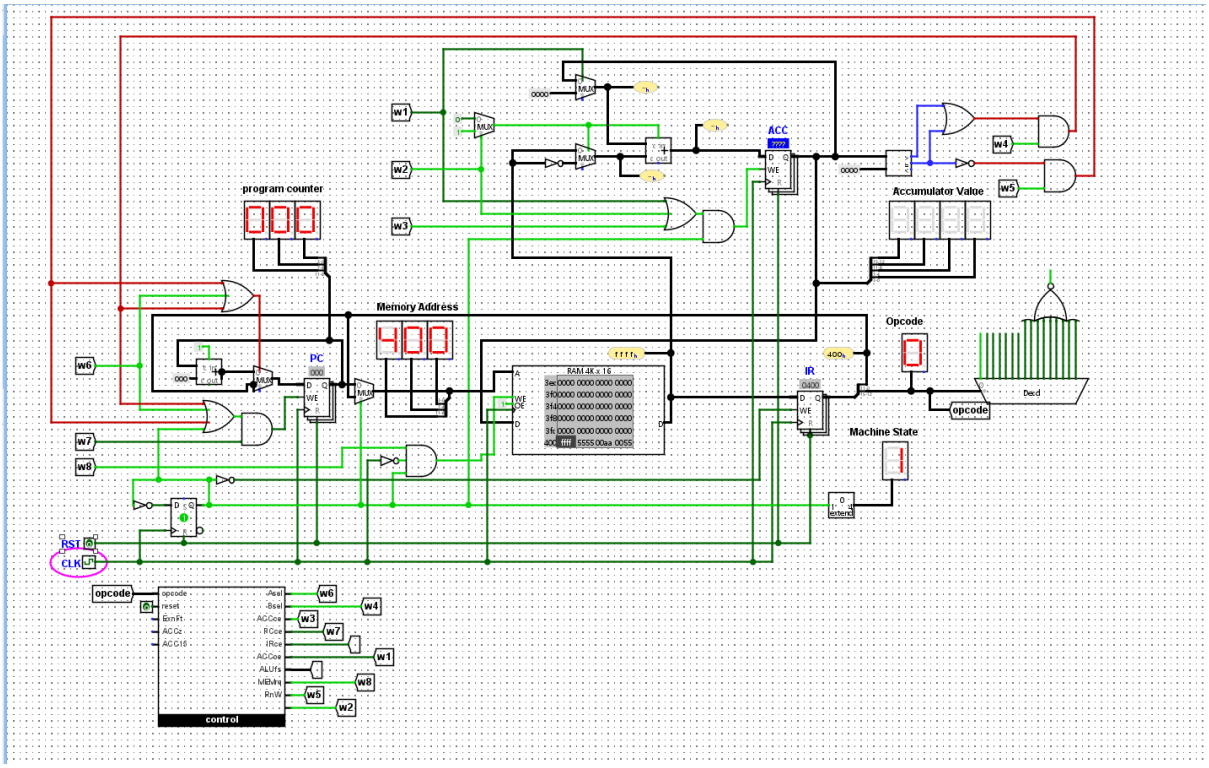
구현한 회로도는 아래와 같습니다.



Control unit의 회로도는 다음과 같습니다.



회로의 기본적인 구조는 프로젝트에서 제시한대로 수행하였으며, wire의 tunnel을 완성하기 위해서는 project의 내부에 별도의 control 파일을 만들어서 연결하였습니다, simulation방법으로는 clk과 reset의 button을 이용하여 변화를 확인하는 방식으로 진행하였습니다.



이후에 위에 4쪽에서 언급한 예상했던 문제와 같이 w4,w5와 같은 tunnel의 연결이 잘못된 건지 이후의 exec과정이 제대로 진행되지 않은 것을 확인 하였습니다.

<고찰>

Exec의 과정이 정확히 이해가 되지 않은 것과 decoder와 연결될 것들에 대해 알아내지 못한 것이 최종적으로 회로완성이 실패의 요인이지 않았을까 생각합니다. 또한 control unit과 그 결과가 다시 alu로 연결되는 과정이 필요한데 그 alu는 또 전체회로에서 어디에 구성되는지 파악되지 않았습니다. 또한 state가 연달아서 input과 output으로 맞물리게 되는데 그 연결 또한 어떤 식으로 연결 되는지 의문이 생겼던 것 같습니다.