(12/14)

**컴퓨터 구조 디지털 프로세서 설계 대회**

분반 : 1

이름 : 김민석

학번 : 20195124

소속 : 소프트웨어융합 대학

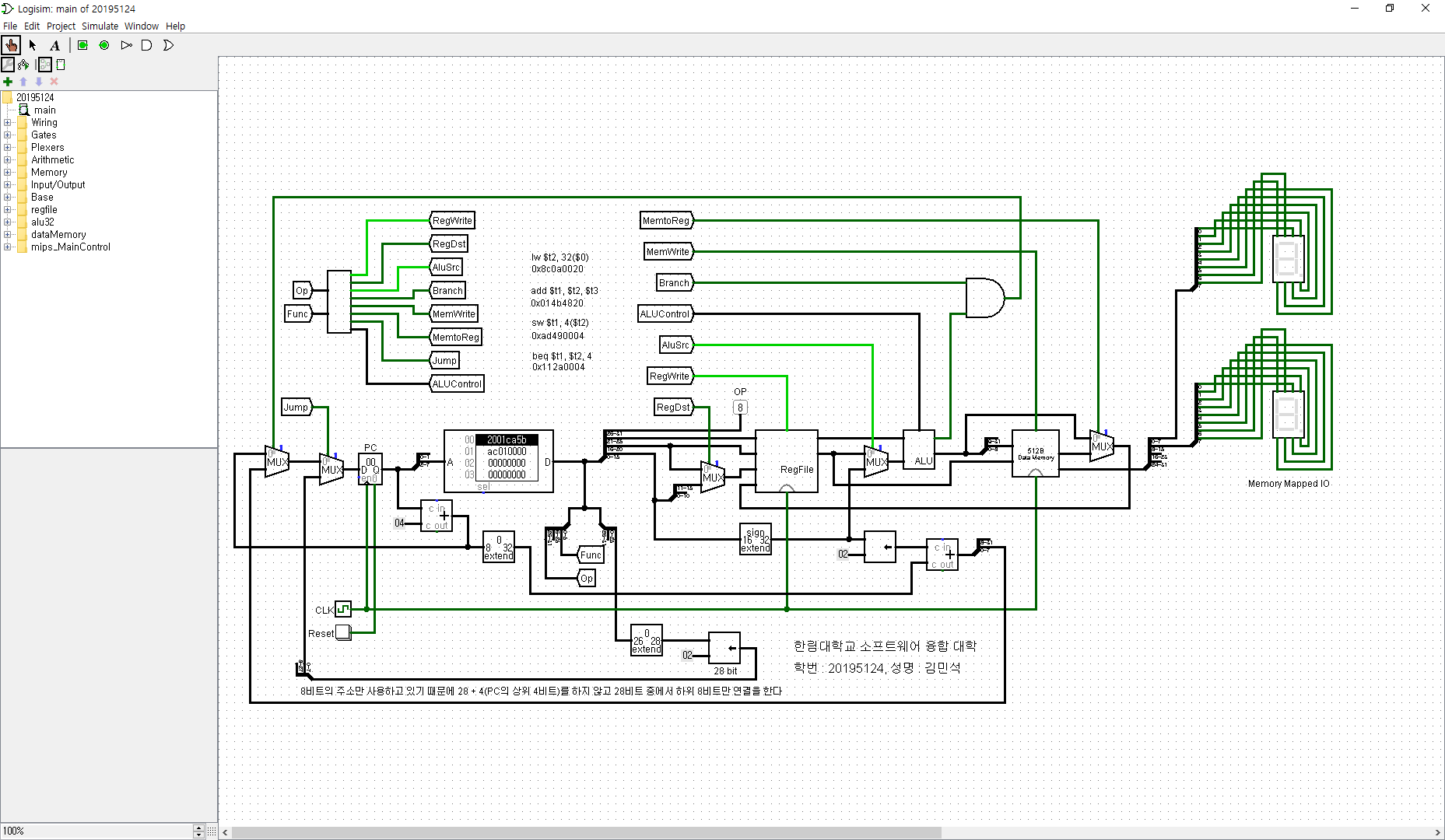
학과 : 컨텐츠 IT

시험 기간이 같이 껴있기 때문에 시간이 날 때마다 명령어를 추가하여 매번 제출하겠습니다!

**목차**

1. **수업 시간에 의해 만들어진 최종 MIPS 구조**
2. **추가한 명령어**
3. **쉬프트 명령어(sll, srl, sra)가 추가된 MIPS 구조**
   1. **원리**
   2. **과정**
4. **쉬프트 명령어(sllv, srlv, srav)가 추가된 MIPS 구조**
   1. **원리**
   2. **과정**
   3. **추가**
5. **아쉬운점**

**수업 시간에 의해 만들어진 최종 MIPS 구조**

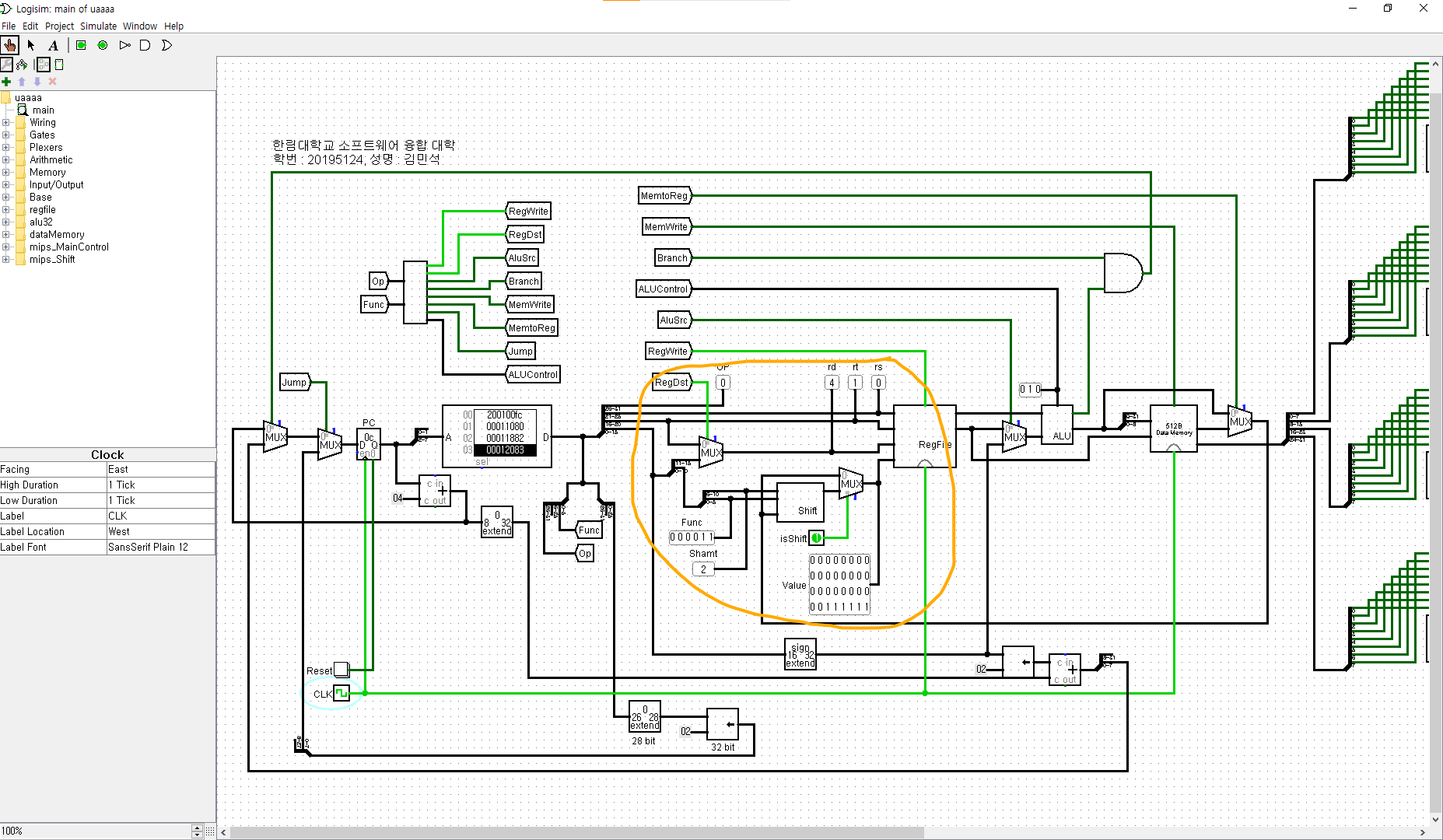


**추가한 명령어**

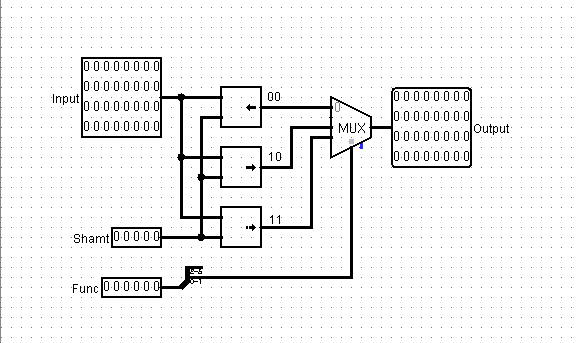
**12/13: sll(shift left logical), srl(shift right logical), sra(shift right arithmetic)**

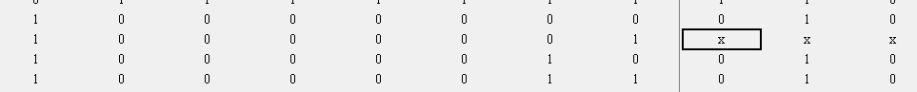
**12/14: sllv(shift left logical variable), srlv(shift right logical variable), srav(shift right arithmetic variable)**

**쉬프트 명령어(sll, srl, sra)가 추가된 MIPS 구조**

****

**원리**



Shift 명령어들은 rs레지스터를 사용하지 않는 R-type명령어입니다. 이때 func값이 각각 sll(0), srl(2), sra(3)인 것을 구글링을 통해 알게된 지식을 통해 func값을 받아 하위 2비트가 00, 10, 11일 경우에 따라서 Input과 shift 양(shamt)을 받아서 원하는 값을 얻을 수 있게 했습니다. 

(ALU Control 자동으로 만들어주는 곳에서 쉬프트 명령어들은 R-type이므로(ALUOP: 10일때), ALU가 덧셈을 하도록 지정)

**과정**

먼저 shift 명령어가 흐르는 방향을 알고 값이 바뀌어야 할 곳을 찾은 후에 가장 간단한 sll 명령어부터 직접 넣어서 테스트를 해보았습니다.

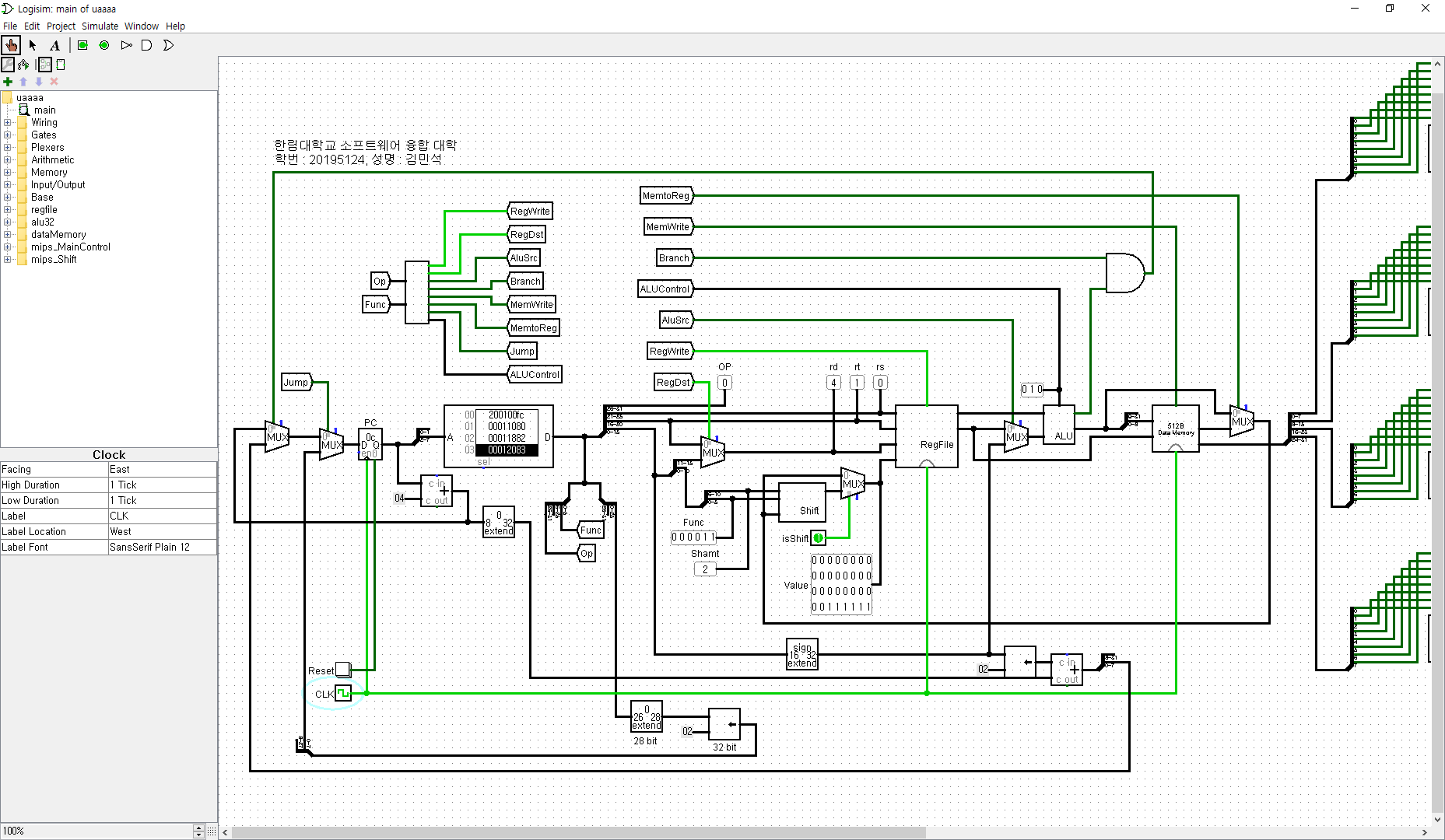
# 레지스터 1번지에 000… 0001을 집어 넣고 sll명령어로 쉬프트한 값을 레지스터 2번지에 넣었습니다.

**addi $1 $0 0x1**

**sll $2 $1 2**

**200100fc**

**00011080**



그 다음에 세 명령어를 한 번에 구분할 수 있도록 func과 shamt 양을 받아서 따로 서킷mips\_Shift를 구현했습니다.

# 레지스터 1에 저장된 0000…. 0000 1111 1100의 값을 sll은 레지스터 2, srl은 레지스터 3, sra의 값은 레지스터 4에 저장되도록 했습니다. 이때 보기 쉽게 각각의 shamt 값은 2로 하였습니다.

**addi $1 $0 0x00fc**

**sll $2 $1 2**

**srl $3 $1 2**

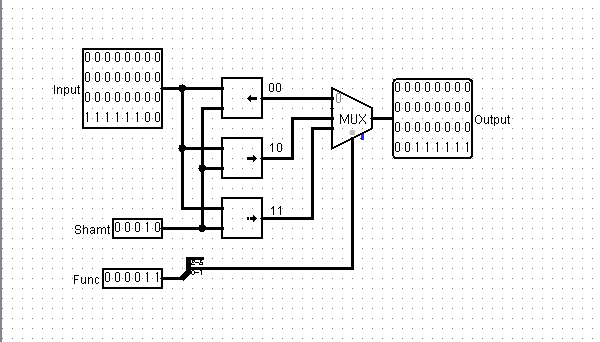
**sra $4 $1 2**

**200100fc**

**00011080**

**00011882**

**00012083**



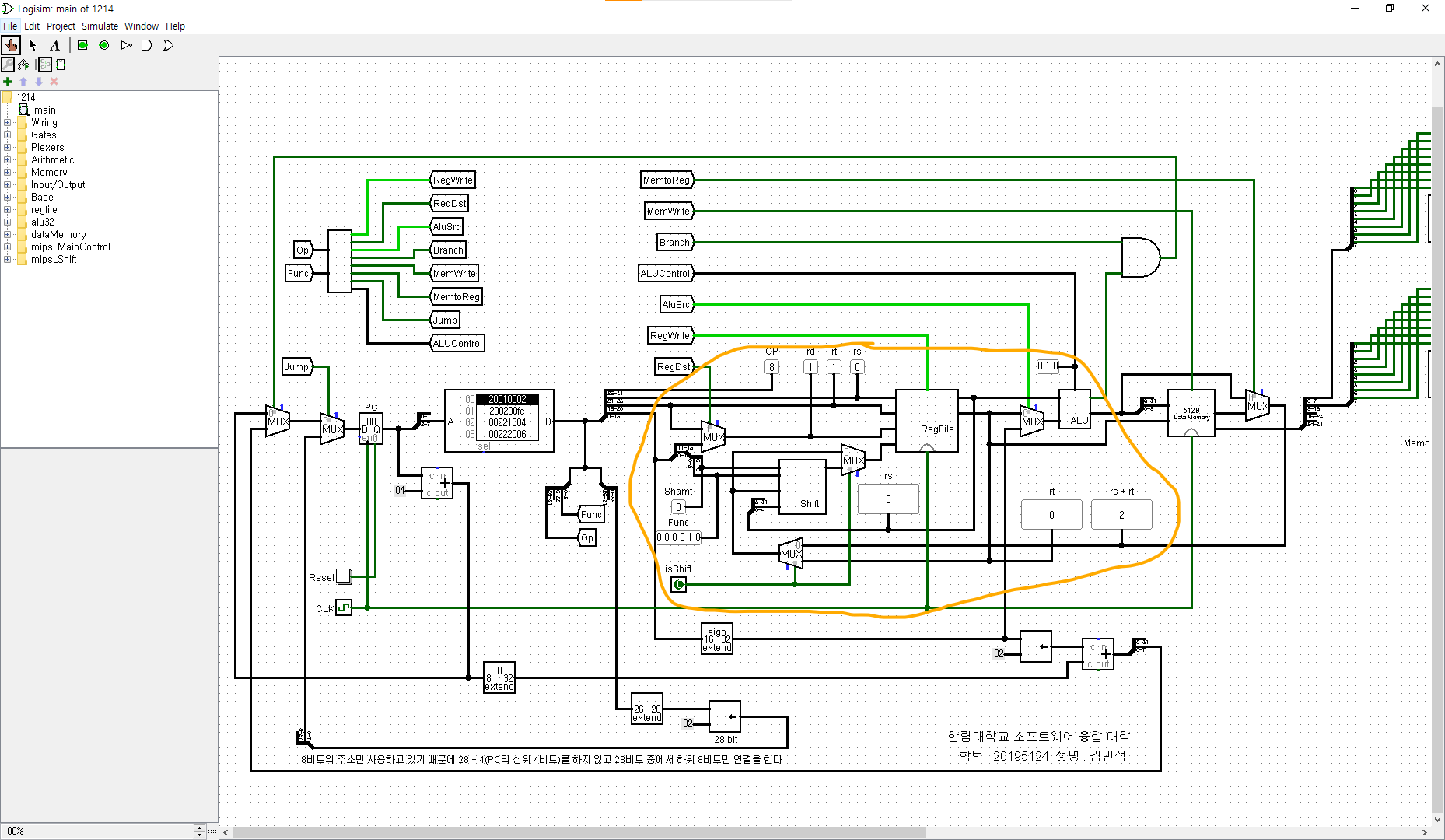
여기서 sra 명령어와 srl 명령어의 결과가 같은데, 그 이유는 레지스터 1번지로부터 값을 받아 쉬프트 명령어를 진행하는데, 1번지에 0x00fc 값을 넣었기 때문에 산술 연산 명령어(sra)와 논리 연산 명령어(srl)의 결과가 같게 나옵니다.

그런 후에 만든 mips\_Shfit 서킷을 기존 프로세서에 넣은 후 func과 shamt 값만 빼내어 보기 쉽게 Probe를 넣어 값이 제대로 나오는지 확인을 한 후에 깔끔히 보이도록 꾸몄습니다.

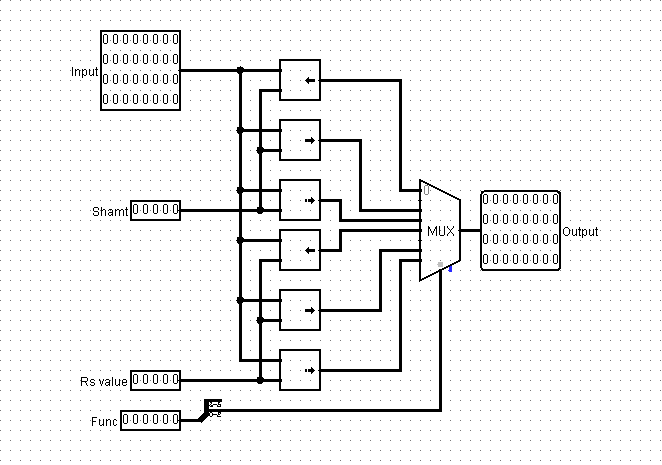


(shift 명령어를 확인하기 위한 addi, sll, srl, sra Hex Code, 교수님께서 알려주신 <http://kurtm.net/mipsasm/index.cgi> 사이트를 이용하였습니다.)

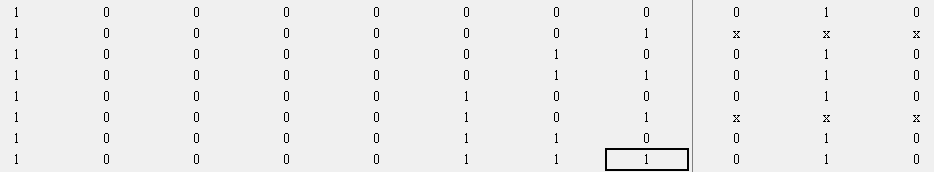
**쉬프트 명령어(sllv, srlv, srav)가 추가된 MIPS 구조**

****

**원리**

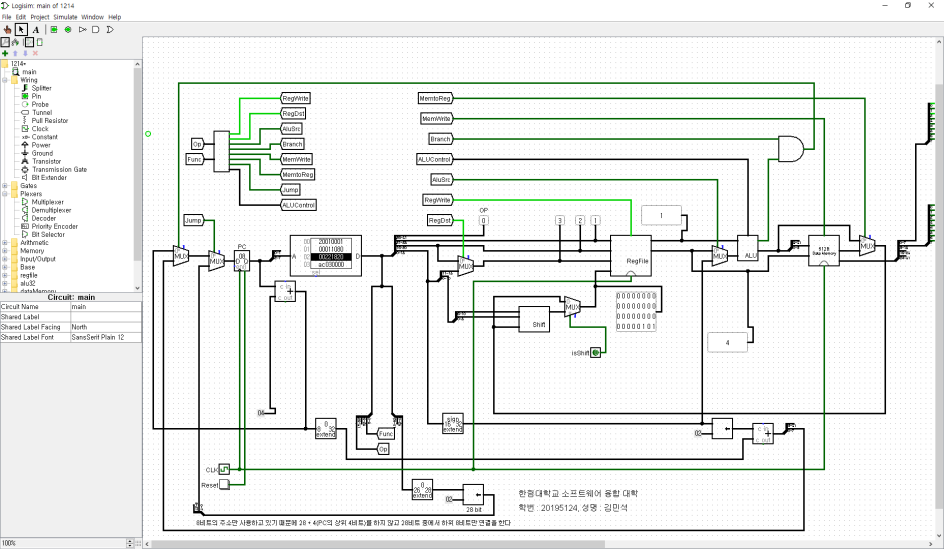


우선 sll 명령어와 sllv 의 명령어의 차이는 shamt를 쓰는지, 레지스터를 쓰는지에 있다는 것을 알게 되었습니다. 이때 쉬프트 명령어를 쓸 때 shamt와 레지스터 중 하나는 0의 값을 가지는 것을 알 수 있었습니다. 그래서 func에 맞춰서 어느 것을 사용할지 조작을 해주면 될 것 같아서 select bit가 3인 MUX를 사용하여 func 6비트 중 하위 3비트가 000(sll), 010(srl), 011(sra), 100(sllv), 110(srlv), 111(srav) 일 때 각각에 맞는 shift를 할 수 있도록 하였습니다.



(ALU Control 자동으로 만들어주는 곳에서 쉬프트 명령어들은 R-type이므로(ALUOP: 10일때), ALU가 덧셈을 하도록 지정)

**과정**



먼저 addi와 add 명령어를 사용해 레지스터의 값이 잘 이동하는지 확인하였습니다.

# 기존에 만들어둔 sll 명령어를 이용하여 add를 해서 segment로 값을 확인하였습니다.

**addi $1 $0 0x1**

**sll $2 $1 2**

**add $3 $1 $2**

**sw $3 0($0)**

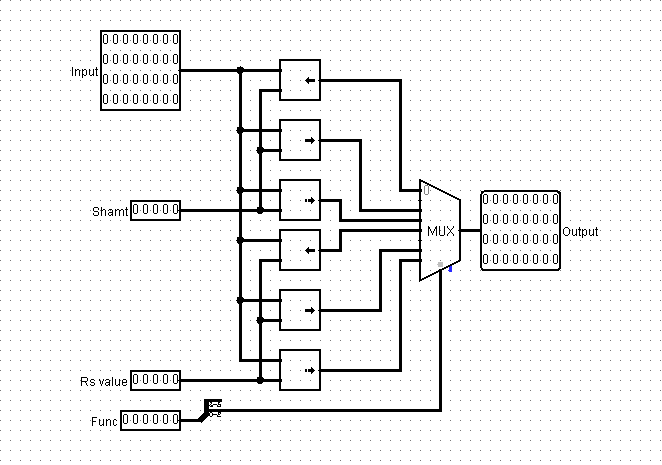
**20010001**

**00011080**

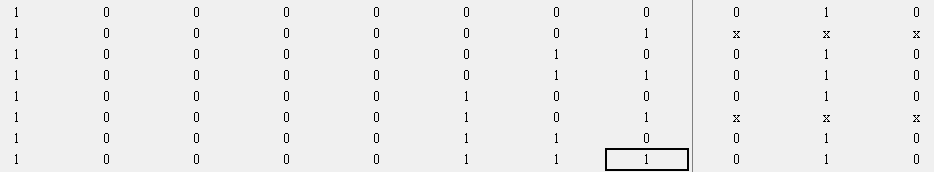
**00221820**

**ac030000**

그런 다음에 Shift 서킷에 레지스터에 대한 입력을 추가하여 항상 rs필드의 값을 받아두었다가 쉬프트 명령어 중 레지스터를 이용하여 shift를 하는 경우에만 사용할 수 있도록 MUX를 구현했습니다.



그리고 나서 sll 때와 같이 ALU Control에서의 Func에 따른 ALU 명령어를 바꿔주었습니다.



# 1번지에 2(shift amount), 2번지에 0000… 0000 1111 1100을 넣고, 변환 사이트에서 작업이 안되어 sllv와 srlv 그리고 srav는 직접 변환을 해서 실행했습니다.

**addi $1 $0 0x2**

**addi $2 $0 0x00fc**

**sllv $3 $2 $1**

**srlv $4 $2 $1**

**srav $5 $2 $1**

**20010002**

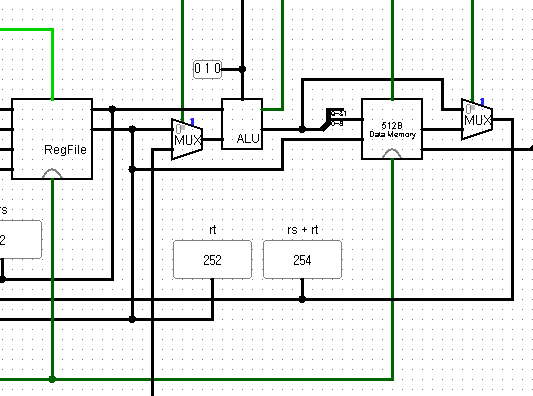
**200200fc**

**00221804**

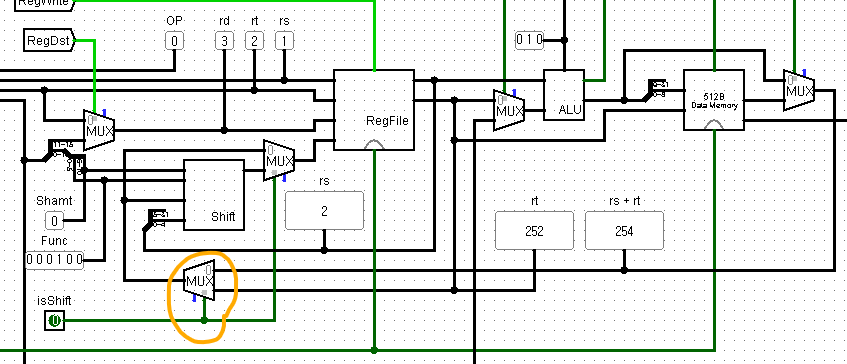
**00222006**

**00222807**

위의 명령어를 넣어서 테스트를 하는데,

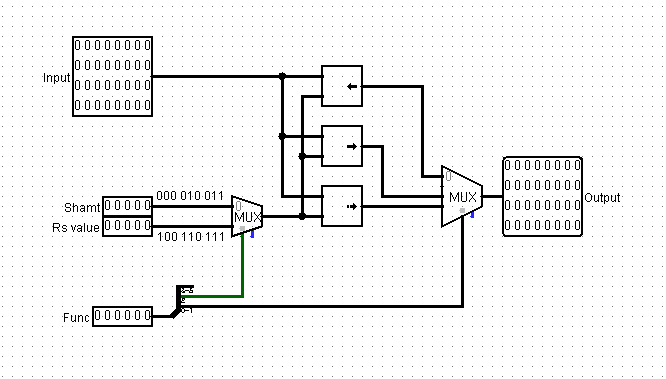


요기 부분에서 rs 값과 rt 값을 더하여(R-type ALU 명령은 더하기이므로) 그 값을 rs(예시에선 2)만큼 shift를 하길래



Shift 명령어일 때만 rt 값에 rs 값만큼 shift 할 수 있도록 MUX를 추가하였습니다.

**추가**



워드를 작성하다가 보니 쉬프트 명령어들의 Func 하위 2비트의 구조가 sll과 sllv 명령어는 00, srl과 srlv 명령어는 10, sra와 srav 명령어는 11이고, Func 하위에서 3번째 비트가 sll, srl, sra는 0 그리고 sllv, srlv, srav는 1이라는 것을 알게 되었습니다.

그래서 Func 필드의 하위에서 3번째 비트가 0이면 shamt를, 1이면 rs 필드 값을 사용할 수 있도록 바꿨습니다.

그 결과 select bit가 3인 MUX의 사용하지 않는 부분(001, 101)이 하나로 줄어들었으며, select bit도 두개, 그리고 Shifter도 세개만 사용하는 등의 간략화를 볼 수 있었습니다.

**아쉬운 점**

(12/13) 제출 기한 연장 전

원래 참여 점수라도 받으려고 했었는데 여러 일정이 겹쳐 뒤늦게 제출 당일 시작하게되었습니다.. 그래서 급하더라도 내부를 바꾸지 않으면 010 -> 110 -> 111로 바뀌던 ALU의 작업을 ALU Control에서 Analize 서킷?의 결과의 일부를 바꿔서 해결했습니다..

그리고 제출 시에 예시 모습과 깔끔하게 명령 줄들만 놔둔 것도 같이 올리려고 했는데 바로 수정할 시간이 부족하네요..

(12/14) 제출 기한 연장 후

우선 연장해주셔서 감사합니다. 단순히 참여 점수만 받아도 되지만 기간이 더 생긴 만큼 공부도 할 겸 궁금한 명령어들을 계속 추가할 수 있다는 점이 좋은 것 같습니다. 감사합니다. ╰(\*°▽°\*)╯

ALU Control에 대한 analize를 통한 자동 서킷 생성에서 테이블 값을 바꾸게 되면서 mips\_Decoder의 테이블 값도 바꾸고 싶었는데 isShift 를 넣었다가 수정해야할 것들이 시험기간에 영향을 줄 것 같아서 밖에서 조작을 할 수 있게 했습니다.

시험기간이라 많은 명령어를 추가할 수 없던 것이 아쉽다..