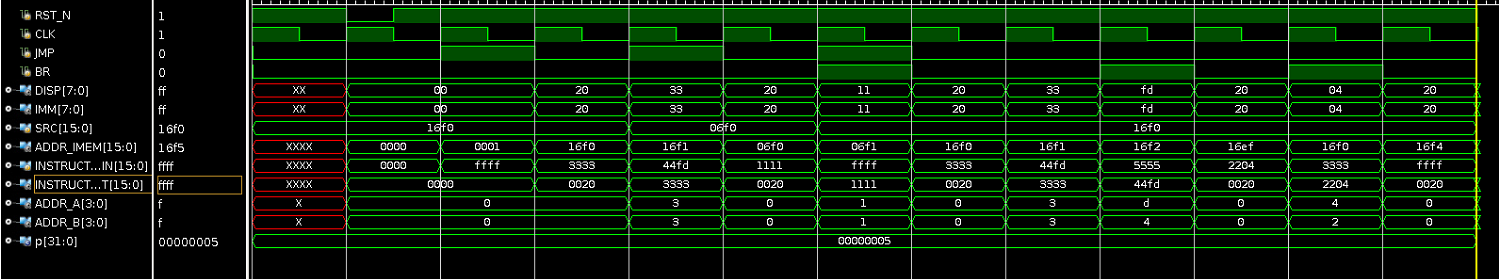
**Digital System Design Lab 05 Report**

**2015-18525 김세훈**

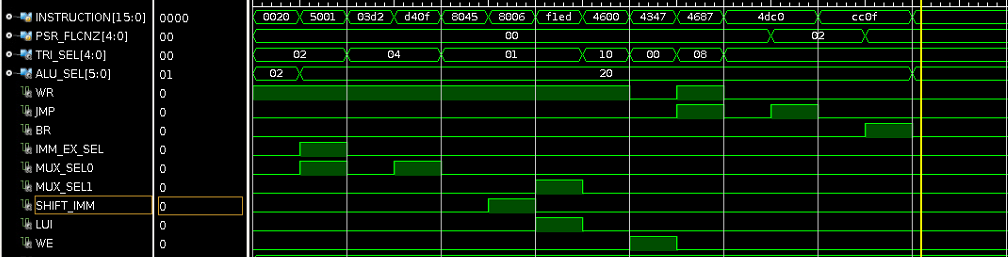
**1. PC & IR 분석**



@1 @2 @3 @4 @5 @6 @7 @8

1. 첫 번째 빨간 상자 (@1→@2) : @1의 JMP와 BR가 모두 0이기 때문에 pc(addr\_imem)는 16f0에서 16f1로 단순히 1이 더해진다. 또한, instruction\_out 역시 @1의 fetch state에서의 instruction\_in이 그대로 전달되어 3333이 된다.
2. 첫 번째 빨간 상자 (@2→@3) : @2에 JMP가 1로 되어있기 때문에 pc는 +1이 되지 않고 src로 들어온 값으로 바뀌게 된다. 즉, @2에서 src로 06f0이 들어오고 있기 때문에 @3에서의 pc는 16f2가 아닌 06f0이 된다. 또한 이 경우 fetch가 되어 있던 @2의 instruction\_in 값인 44fd가 wrong-path에 의한 것이므로 flush가 되어야 하며, 따라서 @3에는 nop인 0020이 instruction\_out으로 출력된다.
3. 두 번째 빨간 상자 (@4→@5) : @4는 @2에서와 동일한 pc(06f1)이지만 이번에는 JMP와 BR가 모두 0이기 때문에 @5에서의 pc는 16f1에서 1 더해진 16f2가 된다. 또한 instruction\_out 역시 flush됮 않고 @4의 fetch state에서의 instruction\_in 값이 그대로 전달되어 44fd가 된다.
4. 두 번째 빨간 상자 (@5→@6) : @5의 BR가 1로 되어있기 때문에 pc는 +1이 되지 않고 disp(의 sign extension) 값이 더해진다. @5에서의 disp 값은 fd(-3)이기 때문에 @6에서의 pc는 16f2(16f2+1)이 아닌 16ef(16f2-3)이 된다. 이 경우 역시 fetch가 되어 있던 @2의 instruction\_in 값인 5555가 wrong-path에 의한 것이므로 flush가 되어야 하며, 따라서 @6에는 nop인 0020이 instruction\_out으로 출력된다.
5. 세 번째 빨간 상자 (@7→@8) : *@8의 pc=16f5, instruction\_out=ffff가 잘려서 그림에 나타나지 않음.* @7의 JMP와 BR가 모두 0이기 때문에 pc는 16f4에서 16f5로 단순히 1이 더해진다. 또한, instruction\_out 역시 @1의 fetch state에서의 instruction\_in이 그대로 전달되어 ffff가 된다.

**2. Decoder 분석**



**# ALU operation group:** TRI\_SEL=00010이며, control flow operation이 아니기 때문에 BR, JMP는 0이다. 또한 register에 값을 작성해야 하므로 WR(reg\_write)는 enable, WE(mem\_write)는 disable되며, ALU를 사용해야 하므로 적절한 ALU\_SEL signal이 생성되어야 한다.

1. 0020 : OR R0, R0

2번째 source가 register이므로 MUX\_SEL0는 0이며, ALU\_SEL는 OR에 해당되는 000010이 생성된다.

1. 5001 : ADDI R0, x01

2번째 source가 immediate이므로 MUX\_SEL0은 1이며, arithmetic 연산이기 때문에 immediate value가 sign-extended 되어야 한다. 따라서 IMM\_EX\_SEL로 1을 넣어준다. 또한 ALU\_SEL는 ADD에 해당되는 100000이 생성된다.

**# Move operation group:** TRI\_SEL=00100이며, control flow operation이 아니기 때문에 BR, JMP는 0이다. 또한 register에 값을 작성해야 하므로 WR(reg\_write)는 enable, WE(mem\_write)는 disable 된다.

1. 03D2 : MOV R3, R2

2번째 source가 register이기 때문에 MUX\_SEL0은 0이다.

1. D40F : MOVI R4, x0F

2번째 source가 immediate이기 때문에 MUX\_SEL0은 1이며, 이 경우 arithmetic 연산이 아니기 때문에 굳이 immediate value를 sign-extension시킬 필요가 없다. 따라서 zero-extension에 해당되는 IMM\_EX\_SEL=0을 생성한다.

**#Shift operation group:** TRI\_SEL=00001이 되며 control flow operation이 아니기 때문에 BR, JMP는 0이다. 또한 register에 값을 작성해야 하므로 WR(reg\_write)는 enable되며, WE(mem\_write)는 disable 된다. Shifter unit을 사용해야 하기 때문에 적절한 lui signal을 생성하여 넘겨준다.

1. 8045 : LSH R0(src, dest), R5(amount)

Source가 register(R0)이기 때문에 MUX\_SEL0은 0이다. 또한 shift amount가 register(R5)로 제공되기 때문에 SHIFT\_IMM가 0으로 설정된다. Lui는 당연히 0으로 설정한다.

1. 8006 : LSHI R0(src, dest), x6 (left shift amount)

Source가 register(R0)이기 때문에 MUX\_SEL0은 0으로 주어진다. 하지만 이번에는 shift amount가 immediate value로 주어지기 때문에 SHIFT\_IMM는 1이다. Lui는 당연히 0으로 설정한다.

1. F1ED : LUI R1(dest), xED(src)

이 경우는 source가 immediate로 주어져 있기 때문에 MUX\_SEL0은 1이 되어야 한다. 또한 lui operation이 실행되어야 하므로 lui는 1로 설정해 주며, 이 경우 shift unit에서 shift amount는 무시되므로 don’t care로 해도 무방하다.

**#Memory(L/S) operation group:** control flow operation이 아니기 때문에 BR, JMP는 0이다.

1. 4600 : LOAD R6(dest), R0(addr)

Register에 값을 작성해야 하므로 WR(reg\_write)는 enable되며, WE(mem\_write)는 disable 된다. TRI\_SEL=10000이 된다.

1. 4347 : STOR R3(src), R7(addr)

Memory에 값을 작성해야 하므로 WR(reg\_write)는 disable, WE(mem\_write)는 enable 되며, TRI\_SEL=0이 되어 register에 들어가는 data를 차단시켜 준다.

**#Control flow operation group:**

1. 4687 : JAL R6(link), R7(target)

이 경우 다른 control flow operation들과 다르게 register를 작성해야 한다. 따라서 TRI\_SEL= 01000으로 설정하며, WR(reg\_write)는 enable, WE(mem\_write)는 disable시켜준다. 또한 무조건적인 jump operation이기 때문에 JMP=1, BR=0으로 설정한다.

1. 4DC0 : Jcond GE R0(target)

TRI\_SEL, WR, WE를 모두 0으로 설정하여 register와 memory 모두 write을 disable해준다. GE 조건을 비교해야 하기 때문에 N=1 or Z=1인 상황에만 JMP가 1이 된다. 따라서 앞 cycle에서는 FLCNZ=00000이므로 JMP=0이며, 뒤 cycle에서는 FLCNZ=00010이기 때문에 JMP=0이다. BR는 두 경우 모두 0이다.

1. CC0F : Bcond LT x0F(disp)

TRI\_SEL, WR, WE를 모두 0으로 설정하여 register와 memory 모두 write을 disable해준다. LT 조건을 비교해야 하기 때문에 N=0 and Z=0인 상황에만 BR가 1이 된다. 따라서 앞 cycle에서는 FLCNZ=00010이므로 BR=0이며, 뒤 cycle에서는 FLCNZ=00000이기 때문에 BR=0이다. JMP는 두 경우 모두 0이다.