Explanation of pipelined ARM processor implement

1. Overview of the pipelined ARM processor  
   과제 파일로 주어진 armreduced module과 ctrlSig.v파일을 기반으로 수정하여 구현했다. Instruction을 IF-ID-EXE-MEM-WB의 5단계로 구분하여 각 단계 사이에 control signal과 중간 연산결과를 저장하는 register를 두어 pipeline을 구현하였다.
2. Explanation about control signals  
   multicycle processor에서는 각 step별로 control signal을 설정하였었다. 그러나 pipleline에서는 instruction이 계속해서 들어오고 control signal이 각 단계에서 register를 통해 전달되기 때문에 step별로 signal을 할당할 필요가 없어졌다. 따라서 18bit의 일차원 벡터로 시그널을 정의했다. 여기서 이전과 다른 점은 매번 Instruction fecth를 하므로 IRwrite시그널이 필요가 없어졌다.  
   branch=s[17]; 이 signal은 해당 instruction이 branch 명령어라면 1 아니면 0.  
   Mwrite=s[16]; 해당 instruction이 memory write를 하면 1 아니면 0.  
   Mread=s[15]; 해당 instruction이 memory read를 하면 1 아니면 0  
   regwrite=s[14]; 해당 instruction이 register write를 하면 1 아니면 0e  
   regdst=s[13]; register write시에 write할 register를 지정(0이면 rd., 1이면 lr)  
   regsrc=s[12:10]; register write시에 register에 복사할 data를 지정(0이면 mdr, 1이면 aluout, 2이면 aluresult, 3이면 b, 4이면 pc)  
   단, 여기서 aluout은 MEM단계에서, aluresult값은 EXE단계에서, pc는 WB단계에서의 값을 말한다.  
   ALUsrcA=s[9]; alu의 첫번째 피연산자를 지정(0이면 A, 1이면 0)  
   ALUsrcB=s[8]; alu의 두번째 피연산자를 지정(0이면 imm, 1이면 b<< inst[12:7])  
   ALUop=s[7:4]; // ALU opcode  
   NZCVwrite=s[3]; //update NZCV flags  
   immsrc=s[2:1]; //select how extend immediate value   
   regbdst=s[0]; //register file 에서 두번째 register를 지정(0이면 inst[3:0], 1이면 inst[15:12])
3. Handling data hazard  
   data hazard는 stall로 구현했다. 매 단계에 추가한 register마다 stall값을 저장하여 해당 instruction이 datahazard가 포함된 instruction이라면(stall==1) memory write, memory read, pc update, register write를 안하도록 설정했다.
4. Handling Control hazard  
   control hazard 역시 stall로 구현했다. branch하도록 결정됐다면 결정된 순간에 pipeline에 들어있던 instruction들의 stall value를 모두 1로 설정하여 해당 instruction들이 무효화되도록 설정했다.
5. Limitation and more idea  
   1) stall되는 동안 alu, register 들은 멈추지 않고 계속 동작을 하기 때문에 발열이나 전력소비 측면에서 손해일 것이다.  
   2) control signal을 18비트 보다 줄일 수 있을 것 같았지만 코드의 변경을 최소화하느라 줄이지 못했다. 특히, multicycle코드에선 alu가 pc의 다음주소를 결정했기 때문에 alu의 피연산자를 정하는 signal의 크기가 컸다.