**Computer Architecture Milestone3**

by kkh

#1

이번에도 어김없이 찾아온 milestone 팁 강좌입니다 라기보다는 그냥 가이드입니다. 여기에 답이나 코딩하는 법을 쓰는 순간 저는 올바른 전공도우미가 아니라고 생각하기에.. 하하. 정말 재밌는 과목이니깐 꼭 여러분의 힘으로 끝까지 다 하시고, 드높은 성취감을 얻으시길 바랍니다..

일단 시작하기에 앞서 막막하네요. 이번엔 꽤 길 것 같은데...언제쓰지..,하지만…해야죠… 말보단 글이 진심을 전하기 쉽고 결국 사랑을 이룰 수 있는 것처럼… 헛소리고요 자! 한번 해보겠습니다.

#2

여기부턴 이제 여러분이 **verilog 코딩**을 하셔야되구요. **모델심 웨이브폼**을 보실 줄 알아야 합니다.

(**verilog 코딩)**

verilog는 ~~.v 이 파일들이죠. milestone 2 하시다보면 꽤 많은 종류의 ~~.v와 module들을 보셨을텐데 이것들을 이용해 코딩하는 겁니다. 우리는 소프트웨어 프로그래밍을 하는게 아니라 하드웨어 디자인을 하고 있기 때문에, 기존에 있는거에 부품을 추가해 나가는 개념으로 생각하시면 됩니다. 그럼 본인의 역량에 따라 module들을 자유롭게 불러오면 되겠네요. 가령 내가 지금 beq를 만드는데 도선 어디부분에 당장 mux가 필요해요(mux란 input2개중에 s가 0이면 앞에 것을, 1이면 2번째 것을 y로 넘기는 애입니다.). 그런 경우 코드 아무데나에다

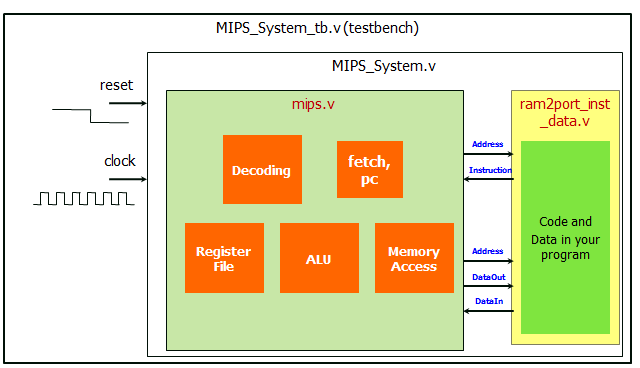
**mux2 #(32) 불라불라( .d0 (input1), .d1 (input2), .s (control), .y (output));**

이렇게 하나 선언해주고 wire들만 잘 이어주면 되겠죠. 기존에 이어지던 wire를 끊어서 wire가 2개가 되었으니, wire하나 더 선언해서 변수명 다르게 해 주시는 것 까먹으면 안돼요..ㅎㅎ 무튼 이런 식으로 하드웨어를 추가적으로 달아서 디자인 하는 겁니다. 물론 최소의 하드웨어로 디자인을 한다면 performance나 경제적으로 좋겠지만 그런 것 까지 프로젝트에서 따지진 않습니다. 여러분은 그냥 무슨 짓을 해서든 그걸 되게만 만들면 되는거죠.

(**모델심 웨이브폼)**

자 두 번째, 모델심은 뭐냐면 그러니깐 Mips\_system안에 엄청나게 많은 모든!!! 변수들이 시시각각,,, instruction이 점점 진행됨에 따라 어떻게 변하는지 보여주는 툴입니다.

밑에 그림을 한번 보시죠.



Milestone 1을 하신 분은 알겠지만 우리가 짠 c코드를 cross 컴파일을 했고, 그것 중에 나오는 결과물 inst\_data.mif를 이용해 memory 초기 값 세팅을 했지요. 그리고 **Mips\_system.v**를 DE0 Board에 올려서 돌렸습니다. 그런데 그림을 보세요. Mips\_system 밖에 tb라고 써있는 애가 하나 더 있죠. 이 녀석은 뭐하는 애일까요.

바로 Simulation 돌릴 때 쓰는 testbench 라고 합니다. 프로젝트에서 쓰는 테스트벤치는 클락(콘센트 꼽은거라고 생각하시면 될 듯)을 생성하고 reset(스위치)을 1(꺼짐)에서 0(켜짐)으로 만들어 주기만 하는 무지 짧은 코드입니다. 하지만 저 2개만 있으면 Mips\_system은 당연히 알아서 돌아가겠죠?

즉 모델심에서는 testbench 파일을 실행시켜서 시뮬레이션을 하는 것이고, 이것은 MIPS\_System\_tb.v로 프로젝트 폴더 안에 예쁘게 들어가 있으니 한번 보시기 바랍니다. 무튼 모델심 조작법은 esca에서 **Synthesis & Simulation**에 다 나와있으니 해보셨으리라 생각하고,, 이런건 다 필요없고 중요한건 나오는 웨이브폼을 해석하는건데.. 그건 이따가 설명을 해보고 일단 지금은 이 정도만 알고 넘어가시죠.

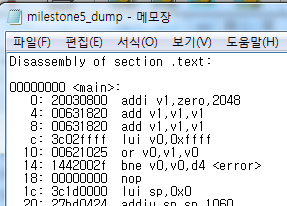
#3 **(verilog 코딩)**

일단 뭘 해야 되는지 좀 부터 보겠습니다. milestone3을 웹페이지에서 보시죠.

뭔가 되게 긴데 **결국 숫자 5와 A를 계속 번갈아가며 표시하는 프로그램을 줄테니 얘가 돌아가도록 해라!** 가 핵심입니다. C프로그램을 줄 테니 얘가 돌아가도록 하드웨어를 바꾸는 거죠. 그런데 하드웨어는 어셈블리어(기계어)로 돌아가고, 모든 Application은 결국 컴파일러가 어셈블리어로 만드니깐, 저 c프로그램을 우리는 일단 컴파일을 해서 얘가 어떤 instruction 들로 바뀌는지를 먼저 봐야할 겁니다. 어차피 mif 파일 만들어서 메모리에 넣어야 하기 떄문에 겸사겸사 하도록 하죠.

(컴파일 중..)

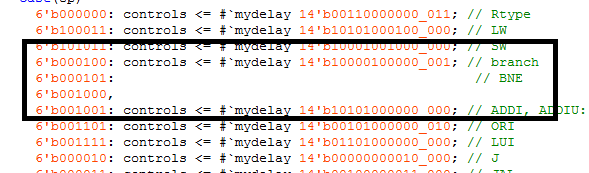
짠! 꽤 많은 파일이 나옵니다. 그중 dump파일이 있죠. 열어보신분은 알겠지만 그 프로그램이 어셈블리어로 어떻게 구성되어있고, 각 instruction은 몇 번지 인지까지 사람이 알아보기 쉽게 잘 정리된 파일입니다. 대충 이런거죠.



이걸 쭈욱 봅니다. 과연 지금 내 깡통 cpu가 실행하지 못하는 instruction이 뭐인가..

만약 add sub sw lw 이런 것으로만 되어있으면 처음부터 실행이 가능할겁니다. 아! 역시,,우리는 알고 있는 거지만, mips\_cpu.v 에는 없는 instruction이 몇 개 있네요. 만들어주면 되겠죠?

수정할 때 감을 못 잡으실 분이 있을 것 같아 말씀드리자면 그 중 하나 bne가 있습니다, 가볍게 예시로 제가 말로만 설명해 볼게요. bne 이전까지 instruction은 주욱주욱 잘 실행될겁니다. 한 clk에 하나씩 instruction fetch decode execution 하면서요. 그런데 bne가 딱 메모리에서 나와서 이제 cpu로 들어가는 장면을 상상해보세요. bne instruction은 분해가 되어 opcode는 opcode 대로 register는 register 대로, immediate는 immediate대로 잘려서 나갑니다. 근데 opcode에 없네요??? 만들어줘야겠죠!



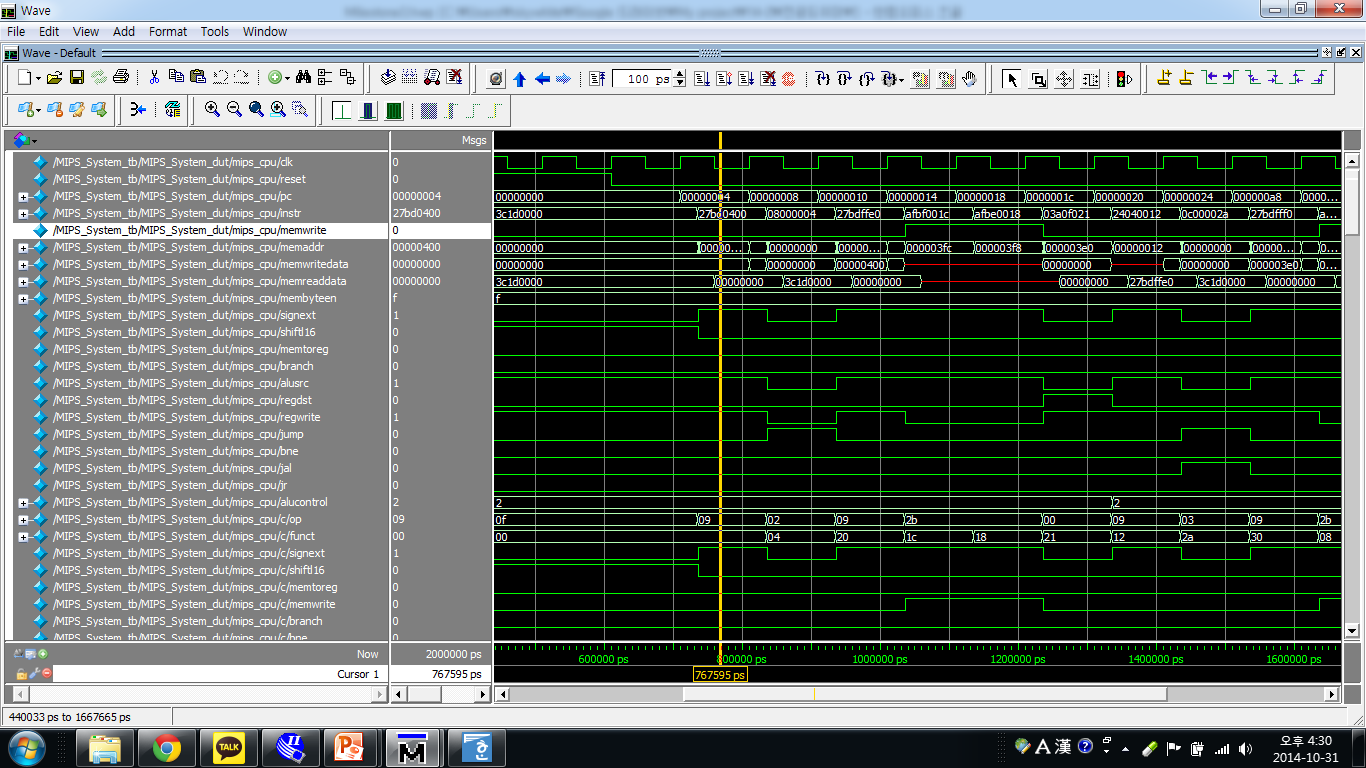
여기서 각 control signal들이 어떻게 나가는지 알아서~ 해주시면 되겠습니다. 이건 제꺼 중 일부고요ㅋㅋ 물론 control signal 외에도 해줘야할게 당연히 있겠죠?? 우리의 목표는 bne가 정상적으로 작동하는 것! pc계산은 어떻게 하는건지.. 이런거 고려하셔서 부품 더해주시고 빼주셔도 무방합니다.

요런식으로 수정하셔가지고 결국 저 모든 instruction들을 작동하게 만들어 주시면 됩니다. 다시 한번 말씀드리자면 **wire!!! 잘 이어 주셔야 해요.** 하나의 wire를 두 군데 다 불러쓰시거나 한곳에 2개 잇거나 그러시면 안 됩니당~

#4 **(모델심 웨이브폼)**

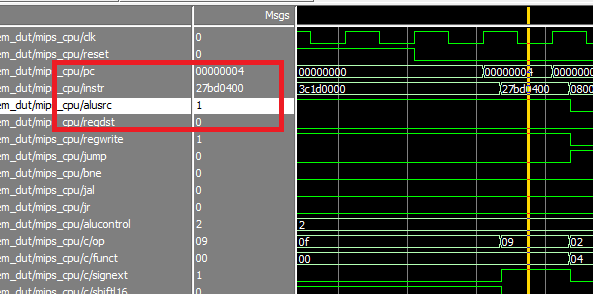
일단 걍 보드에 돌려보면되는데 이걸 왜 봐야되요? 라고 묻는 답변에 대답을 해보자면, Debugging을 위해서입니다. 어느 누구도 한방에 코딩 못합니다. 변수하나 wire선 잘못 잇는 실수가 발생하거든요. 아예 logic이 잘못되었거나..무튼 뭐가 잘못되었는가를 찾아내기 위해 시뮬레이션을 해본다고 보시면 됩니다. 보드에 돌리면 각 변수가 내가 원하던대로 잘 움직이는지 볼 수는 없잖아요. 이러쿵저러쿵..

모델심 웨이브폼 보려고 조작하는 방법은 설명 생략하겠습니다. esca에서 **Synthesis & Simulation**에 다 나와 있으니깐요.. 이제는 그래요 run 을 했을 때 이제 웨이브 폼이 뜬 상황!



여담이지만 single cycle cpu에서 waveform 보는 건 정말 쉽습니다. 한 cycle동안 cpu 모든 부품을 한 instruction이 점유하고 있으니 그냥 수직으로 선하나 긋고 그 때 변수들이 어떻게 되나~ 다 보시면 되니깐요. 이후 진행되는 pipeline cpu는 pc가 누굴 가리키면 그 앞에 3번째 있는애가 execution stage에 있어서 alu는 뭐 어찌되고 궁시렁궁시렁 이러니 지금 연습을 많이 해두시고! milestone4 하실 때 이 모든 걸 활용해서 하시면 되겠습니다..

무튼 지금 위의 그림은 pc가 2번 째 instruction을 가리키고 있어요. 2번째 instruction이 뭔지 제가 dump파일을 몰라서 뭔지는 모르겠는데 waveform 보니깐 addi같네요. signext 변수가 1이되고 29번 레지스터에 0x400이 다음 클락에 들어가거든요(뭐 add 아닐수도 있긴한데 무튼). addi니깐 레지스터가 아닌 immediate를 선택하는 mux 의 control signal alusrc 라는게 분명히 1일 겁니다. 확인해보죠. 변수검색은 Ctrl + F입니다. 타자 치면서 실시간 검색이 되기 때문에 매우 유용해요. 약간 다른 변수들 지우고 pc와 instr과 alusrc를 가까이 놓아 보았습니다.



이런식으로 변수를 찾아보며 control signal은 똑바로 나가는지, 데이터는 똑바로 들어가는지 하나하나 확인 하시는게 디버깅 되시겠습니다... ㅎㅎ 모델심 다루는 법은 정말 그냥 글로 많이 보는게 아니라 만져봐야 압니다. 머리로 어디엔 뭔 값이 있을거다 라는걸 확인하시면서 많이 연습해보세요. 보고서에 modelsim 의 코딩에 해당하는 적절한 스크린샷을 붙여야 하니 꼭이요 !

#5 마치며

지금 여러분이 milestone1 하고 계실 때 다 썼네요. 생각보다 오래 걸릴 줄 알았는데 오래 걸렸습니다ㅋㅋㅋ. 저도 1년 전 것이라서 만져보면서 하느라… 읽느라 수고하셨고요 이제 무운을 빕니다.. 파이팅 ^^\*