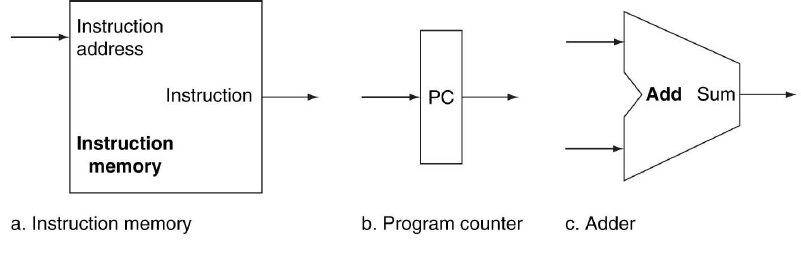
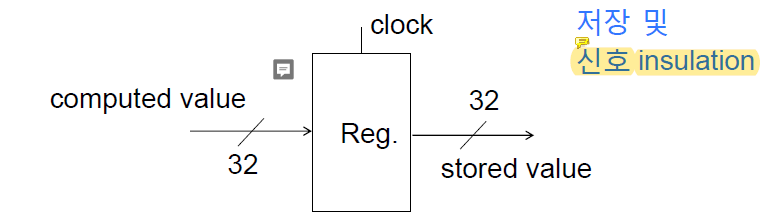
#HW11 2017069598 박상지

the processor – single cycle design

building blocks for IF

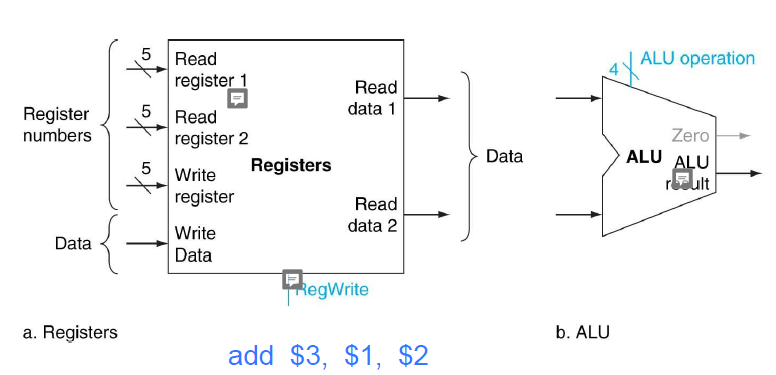


instruction fetch에서는 PC가 가리키는 Instruction 주소를 주어서 해당 instruction에 대한 fetch가 일어난다. (only read) PC가 가리키는 주소를 읽으면 PC값은 자동적으로 +4를 시킨다.

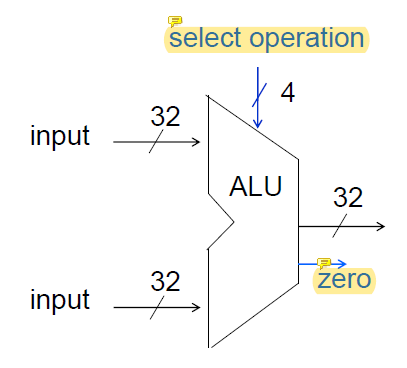


Register에서는 32-bit machine을 사용하면 32-bit 데이터가 input으로 들어오는데, clock이 올라올 때 input 값을 한 주기동안 저장한다. clock이 다시 올라오면 해당 input값에 따라 데이터를 저장한다. 즉 주기가 끝나기 전에 input이 들어와도 레지스터의 값은 바뀌지 않는다. 주기가 끝날 때까지 input가 output이 영향을 주지 않게 해당 신호를 차단을 할 수 있으며 insulation이라 한다.

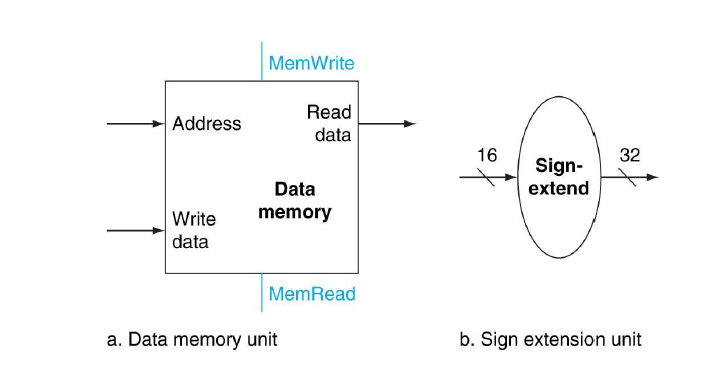
building blocks for r-types



해당 instruction을 예로 처음에 두 개의 레지스터를 읽을 수 있는 channel이 필요하다. 해당 channel은 32-to-1 mux을 사용하며, 32개의 레지스터 중에 하나의 레지스터를 선택을 한다. 더 깊게 들어가면 레지스터를 선택하기 전에 5-to-32 decoder을 통해서 신호에 따른 결과 값이 하나씩 만 있기 때문에 5-to-32 decoder는 5-bit으로 표현되는 레지스터에서 신호를 받은 해당 레지스터만 mux으로 신호를 보낸다. 두 개의 레지스터를 읽으면 Read data 1, 2에 ALU 연산으로 보내는데 ALU연산에서도 마찬가지 어떤 연산인지 확인하기 위해 select operation와 zero signal이 존재한다. select operation에서 add, sub, 등의 signal이 opcode에 따라 결정되며 zero signal은 beq에서 사용되는데 ALU에서 빼기 연산으로 zero의 값이 나오면 jump를 한다. 현재 add의 연산이기 때문에 결과값을 다시 보내기 위해 ALU result를 통해 Write Data에 데이터를 보내고 Write register에 해당한 레지스터에 값을 저장한다. add라는 instruction을 실행시키면 두 개의 레지스터를 더해서 레지스터에 저장을 해야 하는데 여기서 RegWrite 신호를 enable를 해줘야 한다. 반대로 beq는 두 개의 레지스터를 같은 지 아닌지 비교만 하기 때문에 신호를 disable 즉, 0의 값을 줘야 한다. 이것도 역시 opcode에서 결정한다.

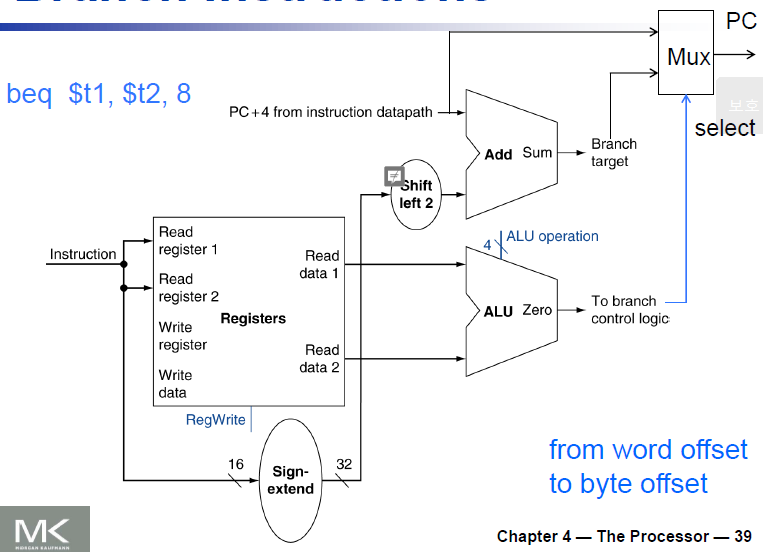


building blocks for lw/sw



load에서는 메모리를 읽기 때문에 MemRead 신호에 1을, store에서는 메모리에 쓰기를 해야 하기 때문에 MemWrite에 신호를 줘야 하고, 저장할 데이터도 받아야 한다. load/store에서는 immediate 값은 16-bit이기 때문에 ALU 연산에 보내기전에 Sign extend에서 32-bit으로 만들어 준다.

Branch Instructions



beq에서 immediate 값을 16-bit에서 32-bit으로 바꾸고 해당 immediate값은 word 단위이기 때문에 byte로 바꿔 주기 위해서 shift를 해준다. 두 개의 레지스터 같으면 ALU 연산에서 0의 값을 받아 Mux로 통해 해당 PC값을 jump를 하고 아니면 PC+4를 받는다.