**Computer Architecture**

**Lab 09**

Build a Multi-cycle implementation of a processor core -KUSTAR



**2013210128 변민재**

**1. Code 및 주석**

|  |  |
| --- | --- |
| **control**  **\_units** |  |
| **kustar** |  |
| **regfile** |  |
| **mem** |  |
| **mux2to1** |  |
| **alu** |  |
| **components** |  |

**2. 결과 waveform**

(별도 첨부)

* kustar

**3. 결과 waveform 원인 분석**

1) control\_units

kustar에서는 읽어온 데이터의 Operation에 따라 state가 달라지고 그에 따라 mcu에 속하는 값들 또한 달라진다. state machine은 다음과 같다.

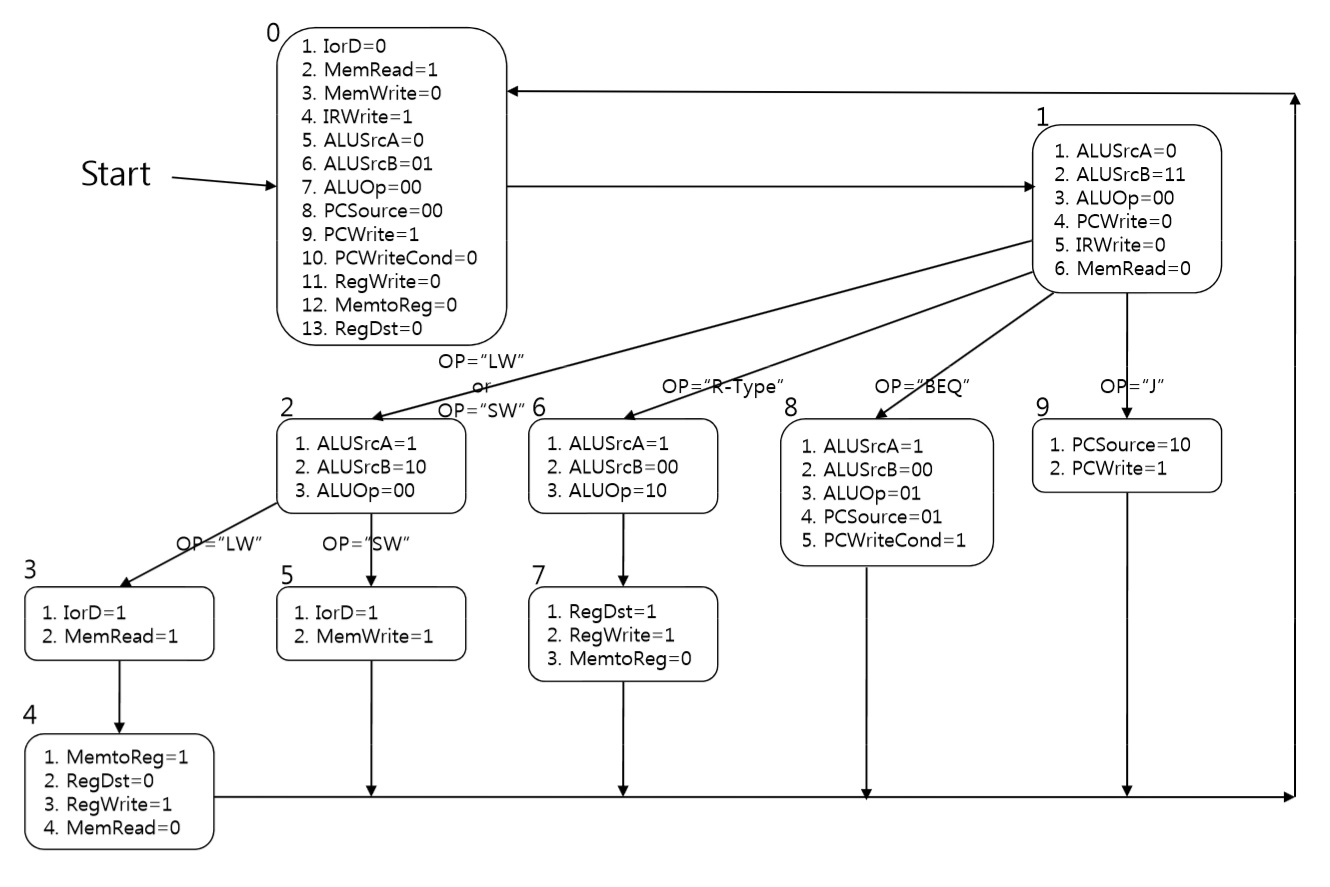


표 1 시간에 따른 state

|  |  |  |  |
| --- | --- | --- | --- |
| **명령어** | **Operation** | **time** | **state** |
| 8d42005a | LW R2, 90(R10) | 150ns~250ns | 0 |
| 250ns~350ns | 1 |
| 350ns~450ns | 2 |
| 450ns~550ns | 3 |
| 550ns~650ns | 4 |
| 8d43005e | LW R3, 94(R10) | 650ns~750ns | 0 |
| 750ns~850ns | 1 |
| 850ns~950ns | 2 |
| 950ns~1050ns | 3 |
| 1050ns~1150ns | 4 |
| 00432022 | SUB R4, R2, R3 | 1150ns~1250ns | 0 |
| 1250ns~1350ns | 1 |
| 1350ns~1450ns | 6 |
| 1450ns~1550ns | 7 |
| ad44005a | SW R4, 90(R10) | 1550ns~1650ns | 0 |
| 1650ns~1750ns | 1 |
| 1750ns~1850ns | 2 |
| 1850 ns~1950 ns | 5 |
| 014c2820 | ADD R5, R10, R12 | 1950 ns~2050 ns | 0 |
| 2050 ns~2150 ns | 1 |
| 2150 ns~2250ns | 6 |
| 2250ns~2350ns | 7 |
| 012d3020 | ADD R6, R9, R13 | 2350ns~2450ns | 0 |
| 2450ns~2550ns | 1 |
| 2550ns~2650ns | 6 |
| 2650ns~2750ns | 7 |
| 10a60005 | BEQ R5, R6, 5 | 2750ns~2850ns | 0 |
| 2850ns~2950ns | 1 |
| 2950ns~3050ns | 8 |
| 08000000 | J 0 | 3050ns~3150ns | 0 |
| 3150ns~3250ns | 1 |
| 3250ns~3350ns | 9 |
|  |  | 3350ns~3450ns | 0 |

2) kustar

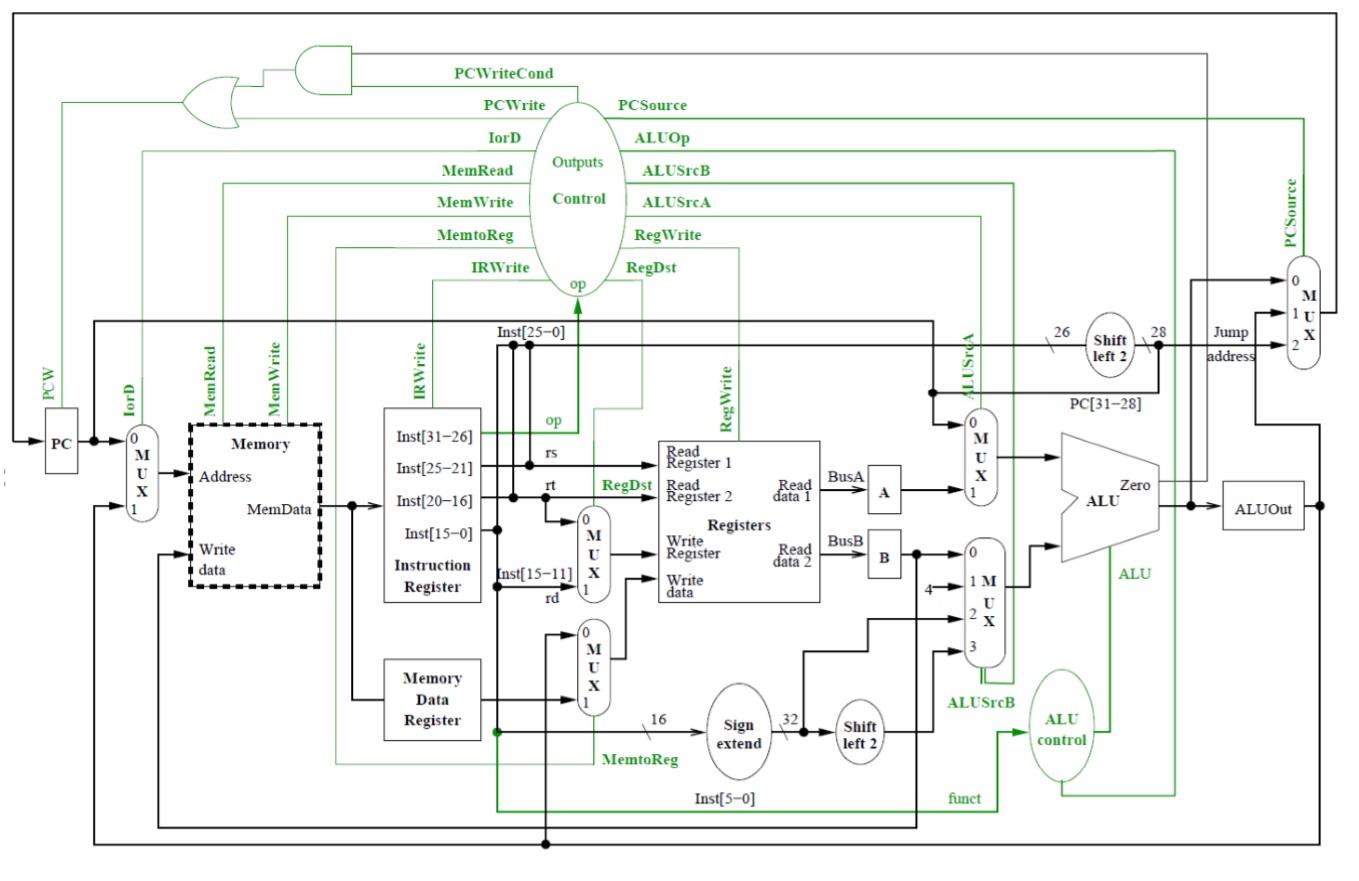


표 2 시간에 따른 path

|  |  |
| --- | --- |
| **time** | **path** |
| 0~10000ns | clock은 50ns마다 바뀐다. (0과 1 중 하나의 값을 갖는다.) clock이 50ns마다 바뀌기 때문에 clock cycle은 100ns가 된다.  clear는 초기값을 1로 주어 데이터를 reset한 다음 10ns 뒤에 clear를 0으로 바꾸어서 데이터가 reset되지 않도록 한다. |
| 0ns~150ns | state가 시작되기 전이기 때문에 아직 값을 전달받지 못한 wire나 register는 빨간색 줄로 표시된다. |
| 150ns~  250ns | start 단계로 current state는 0이고 각각의 register들이 값을 받아온다. |
| 250ns~  350ns | ①current state는 1로 바뀌고 ALUSrcB, PCWrite, IRWrite, MemRead 값이 바뀐다.  ②single register IR은 mem.dat에서 값을 읽어온다. 읽어온 값은 8d42005a이고 명령어로는 LW R2, 90(R10)를 의미한다. |
| 350ns~  450ns | ①operation이 LW이므로 current state는 2로 바뀌고 ALUSrcA, ALUSrcB 값이 바뀐다.  ②ALUSrcA와 ALUSrcB를 select로 하는 mux3과 mux4가 ALU에 값을 넘겨준다.  ③ACU에서 ALUOp가 00이기 때문에 ALUcontrol은 3’b010이 되고 ALU는 add 연산을 수행한다. |
| 450ns~  550ns | ①operation이 LW이므로 current state는 3으로 바뀌고 IorD, MemRead값이 바뀐다.  ②ALU에서 연산된 값은 single register인 ALUOut을 지나 MUX에 도달하게 된다.  ③mux0의 경우 select로 사용되고 있는 IorD값이 1이 되었기 때문에 ALUOuttoMux값을 메모리의 address값으로 전달한다.  ④이 때 MemRead 값이 1이므로 메모리에서는 전달받은 address값에 해당하는 데이터를 읽는다. |
| 550ns~  650ns | ①current state는 4가 되고 MemtoReg, RegWrite, MemRead 값이 바뀐다.  ②메모리에서 읽어온 데이터는 single register인 IR과 MDR에 전달되지만 (MemtoInstReg) IR은 WE로 사용되고 있는 IRWrite값이 0이기 때문에 값이 전달되지 않는다. MDR은 WE값이 항상 1이기 때문에 데이터가 Mux로 전달된다.(MemdatatoMux) |
| 650ns~  750ns | ①값을 전달받은 mux2는 MemtoReg가 1이기 때문에 값을 RegisterFile로 전달한다.(MuxtoWriteData) RegisterFile에서는 RegWrite가 1이기 때문에 전달받은 값을 저장하고 current state는 다시 0이 된다. |
| mem.dat에서 다음 명령어를 읽어와서 OP에 따라 다음의 state path를 반복한다.   * LW : 0->1->2->3->4->0 * SW : 0->1->2->5->0 * R-Type : 0->1->6->7->0 * BEQ : 0->1->8->0 * J : 0->1->9->0   Jump 연산에서는 항상 주소값 0을 가리키므로 @000에서부터 다시 연산을 무한반복한다. | |

**5. 구현시 어려웠던 점**

1) WE값이 표기되지 않은 single register가 있어서 WE값을 어떻게 주어야 하는지 몰랐는데 WE값을 1로 주면 항상 쓰기가 가능하기 때문에 WE값이 표기되지 않은 single register는 WE값을 1로 설정해주었다.

2) 주어진 wire와 register로 한정시켜 생각하다 보니 생각과 다른 이름이면 찾기가 힘들었는데 wire의 크기를 보고 판단했더니 해당되는 wire를 좀 더 쉽게 찾을 수 있었다.

3) 결과 waveform에서 처음에는 값이 주어지지 않기 때문에 빨간줄이 나오는 것이 정상인데 분석하지 않고 틀렸다고 생각해서 코드를 한참 검토했다. 어떤 값들은 값을 전달받기까지 약간의 시간이 필요하다.

4) wire의 크기와 할당하려는 값의 크기를 맞추어주는 것이 어려웠다. 그래서 컴파일할 때 warning이 뜨거나 시뮬레이션했을 때 일부 비트는 값을 받아오지 못해 파란줄이 뜨기도 했다.

5) 데이터 파일을 읽어오지 못해서 코드에 문제가 있는 줄 알았는데 프로젝트 파일과 데이터 파일의 경로를 같게 해주었더니 값을 제대로 읽어왔다.

6) Store 연산을 한 후 Add연산을 할 때 R-Type임에도 불구하고 state path가 1에서 2로 가서 그 후의 연산이 제대로 수행되지 않았다. MCU에서 control 값을 주는 과정에서 약간의 시간차 때문에 일어난 일로 MemWrite가 1일 때 약간의 delay를 주었더니 해결되었다.