Wesley Vo

ESE 356 Project 1 Phase 2

ALU Updated Control Signals

Instr	Imm	Unsigned	2's comp	Carry	ADD	MUL	AND	OR	XOR	MOV	Shift	Read from RF Rdest	Read from RF Rsrc	Read from DM	Write to DM	Write to RF rdest
ADD					1							1	1			1
ADDI	1				1							1				1
ADDU		1			1							1	1			1
ADDUI	1	1			1							1				1
ADDC				1	1							1	1			1
ADDCI	1			1	1							1				1
MUL						1						1	1			1
MULI	1					1						1				1
SUB			1		1							1	1			1
SUBI	1		1		1							1				1
SUBC			1	1	1							1	1			1
SUBCI	1		1	1	1							1				1
CMP			1		1							1	1			

Wesley Vo
ESE 356 Project 1 Phase 2

CMPI	1		1		1							1				
AND							1					1	1			1
ANDI	1						1					1				1
OR								1				1	1			1
ORI	1							1				1				1
XOR									1			1	1			1
XORI	1								1			1				1
MOV										1		1	1			1
MOV I	1									1		1				1
LSH		1									1	1	1			1
LSHI	1	1									1	1				1
ASH											1	1	1			1
ASHI	1										1	1				1
LUI	1										1	1				1
LOA D												1	1	1		1
STO R												1	1		1	
Brnch			1	1	1								1			
Jump													1			

Wesley Vo
ESE 356 Project 1 Phase 2

Controller Comments

- The inputs of the controller module are
    - The retrieved instruction from the PM module
    - The PSR flags from the ALU
    - The target address from the ALU to be updated into the PC
- Based on the above inputs, the SC_METHOD process was designed to be sensitive only to the retrieved instruction “pm_instr”
    - Not sure if this is correct but this is what makes sense to me
- Because of the sensitivity list, the program cannot have consecutive NOP instructions since this will not trigger the process and therefor not update the PC
    - i.e.: the NOP and another type of NOP instruction, 0x0000 and 0x0400, were used to help trigger the sensitivity list and were placed at the beginning as the first 5 NOP instructions to have the programs start at address 10 and at the end after the ‘last instruction’ to help keep the pipeline running for the last instruction’s 5 stages
- The outputs of the controller module are (with connection to each module)
    - PM module
        - The current PC
    - RF module
        - RF addresses 1 and 2 decoded from the received instruction, along with a signal to its read enable signal
        - RF address 3 used for the writeback stage along with an appropriate write enable signal
    - ALU module
        - Various control signals along with its enable signal
        - The immediate value to be used
    - DM module
        - Appropriate read and write enable signals
- The PC increments by 1 each clock cycle. In case of any branch/jump instructions the Rtarget address will be divided by 2 before being updated into the PC since the program description indicates that the addresses increase by 2.

Wesley Vo
ESE 356 Project 1 Phase 2

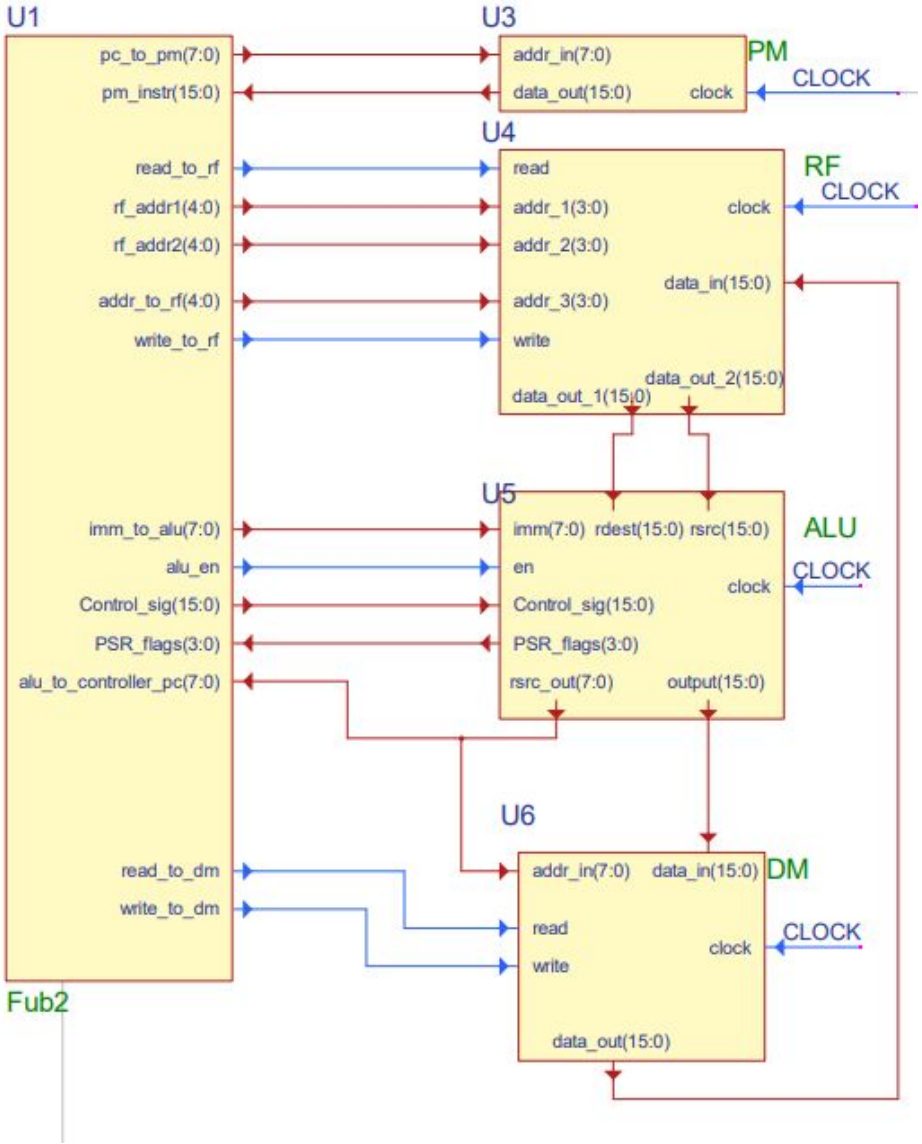

Figure of Controller and Datapath. The controller is the left module and the datapath is the right

Wesley Vo
ESE 356 Project 1 Phase 2

Program 1 Comments

- Program 1 instruction address 26 is “ADD R8, R2” and expected to be 26, however our program had an output of 65
    - However, the previous two instructions affect these two registers and therefore the data to be written into R8 and R2 have not updated yet (data has not reached WB stage yet)
    - Therefore, R8 is still a value of 29 and R2 is still a value of 36 (29+36 = 65)
- Program 1’s last WB stage occurs at 195ns
- Last DM writing stage occurs at 210ns
- The final RF register values and DM values are shown below; ones not shown are still 0

RF address	RF Data	DM address	DM Data
0	0	10	36
2	65	12	12
3	46	14	-24
4	12	46	14
5	12		
6	-24		
7	14		
8	14		

Program 2/3 Notes

- Program 2 and 3 are extremely similar to each other so the final register values should not be any different (output.txt are pretty much the same)
- The only different instruction is the CMPI instruction, but both programs will trigger the JNE instruction
- The first four initial ADD instructions all have an RDEST of R1 which will cause data hazards since data forwarding was not implemented

Wesley Vo
ESE 356 Project 1 Phase 2

- The ISA for ADD is ADD Rsrc, Rdest and all the ADD instructions have R1 as Rdest, therefore data hazards will definitely occur
- The first ADD instruction will execute on the operands -25 and 39 and the result is 14, but is not written back into R1 yet (written at 105ns)
  - R2 = -25 and R1 = 39
- The second ADD instruction will execute on the operands 14 and 39 and the result is 53, but is not written back into R1 yet (written at 115ns)
  - R3 = 14 and R1 = 39
- The third ADD instruction will execute on the operands -50 and 39 and the result is -11, but is not written back into R1 yet (written at 125ns)
  - R4 = -50 and R1 = 39
- The fourth ADD instruction will execute on the operands 70 and 14 and the result is 84, but is not written back into R1 yet (written at 135ns)
  - R5 = 70
  - R1 = 14 → this was finally updated from the first ADD instruction

- Jumping occurs at 160ns during the MEM ACCESS stage and will jump to pc =16 which is address 32
  - I designed the PC to increment by 1 so to account for this the Rtarget address was divided by 2 before jumping
- The final RF register values are shown below; ones not shown are still 0

RF address	RF Data
1	99
2	-25
3	14
4	-50
5	70
6	6

Wesley Vo
ESE 356 Project 1 Phase 2

7	7
8	8
10	32