MIPS assembly code for computation of “d = (a+b)*(b-c)”

00000000 1C090001 // LW $9,1($0)

00000001 1C0A0002 // LW $10,2($0)

00000002 1C0B0003 // LW $11,3($0)

00000003 012A6000 // ADD $12,$9,$10

00000004 054B6800 // SUB $13,$10,$11

00000005 158D4000 // MUL $8,$12,$13

00000006 20080004 // SW $8,4($0)

MIPS assembly code for computation of “d = (a+b)*(b-c)” including NOPs

00000000 00000000 // NOP

00000001 1C090001 // LW $9,1($0)

00000002 1C0A0002 // LW $10,2($0)

00000003 1C0B0003 // LW $11,3($0)

00000004 00000000 // NOP

00000005 012A6000 // ADD $12,$9,$10

00000006 054B6800 // SUB $13,$10,$11

00000007 00000000 // NOP

00000008 00000000 // NOP

00000009 158D4000 // MUL $8,$12,$13

0000000A 00000000 // NOP

0000000B 00000000 // NOP

0000000C 20080004 // SW $8,4($0)

Firstly, we identify the instructions that may result in data dependencies. These are the instructions that may result in Read-After-Write dependency (RAW):

- Between “LW $10,2($0)” and “ADD $12,$9,$10”
- Between “SUB $13,$10,$11” and “MUL $8,$12,$13”
- Between “MUL $8,$12,$13” and “SW $8,4($0)”

To eliminate data hazards due to RAW dependency, we have to include NOPs into our MIPS assembly code. The instructions identified above have to be stalled till the WriteBack stage of the previous instruction is on the same cycle as the Decode stage of current instruction that have RAW dependency.

1	LW $9,1($0)	Fetch	Decode	Execute	Memory	WriteBack										
2	LW $10,2($0)		Fetch	Decode	Execute	Memory	WriteBack									
3	LW $11,3($0)			Fetch	Decode	Execute	Memory	WriteBack								
4	ADD $12,$9,$10				NOP	Fetch	Decode	Execute	x	WriteBack						
5	SUB $13,$10,$11						Fetch	Decode	Execute	x	WriteBack					
6	MUL $8,$12,$13							NOP	NOP	Fetch	Decode	Execute	x	WriteBack		
7	SW $8,4($0)										NOP	NOP	Fetch	Decode	Execute	Memory	WriteBack

Snapshot of instruction and data-memory from ISIM simulation window

Snapshot of instruction-memory from ISIM simulation window:

	0	1	2	3
0x0	00000000	1C090001	1C0A0002	1C0B0003
0x4	00000000	012A6000	054B6800	00000000
0x8	00000000	158D4000	00000000	00000000
0xC	20080004	XXXXXXXX	XXXXXXXX	XXXXXXXX
0x10	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX

Snapshot of data-memory from ISIM simulation window:

	0	1	2	3
0x0	XXXXXXXX	0000000C	0000000B	00000003
0x4	000000B8	XXXXXXXX	XXXXXXXX	XXXXXXXX
0x8	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
0xC	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
0x10	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX

Workflow of the five stage pipeline for both LW and SW instruction

1. LW instruction workflow on the five stage pipeline

LW $9,1($0)

This is the first instruction of the set of MIPS assembly code written for the computation of “d = (a+b)*(b-c)”. Data in address 00000001 (hexadecimal) of data memory will be store in register $9. It is assumed that each stage of the pipeline takes one cycle each.

1st cycle: In the Fetch stage, PC provides the read address of 00000001 (hexadecimal) to the instruction memory. This can be seen at PCOUT[31:0] between the time of 165ns and 195ns. Based on the read address of 00000001 (hexadecimal) given by the PC, content of 1c090001 (hexadecimal) in address of 00000001 (hexadecimal) in instruction memory will be read and transferred to register file. However, due to delay of reading the content of INST[31:0], the value of INST[31:0] is only reflected in the 2nd cycle which is in Decode stage.

2nd cycle: In the Decode stage, read address of 00000 (binary) from instruction value of 1c090001 (hexadecimal) will be used to obtain the data of 00000000 (hexadecimal) from the register file. 0001 (hexadecimal) from the instruction value of 1c090001 (hexadecimal) will be sign extended to 00000001 (hexadecimal).Write address to register file, 09 (hexadecimal), is obtained from the instruction value of 1c090001 (hexadecimal) which will be used during the 5th cycle, WriteBack stage. Finally, the values will be stored in the ID_EXE pipeline register.

In the ID_EXE pipeline register:

rdata1_ID_EXE[31:0] = 00000000 (hexadecimal) (From register file)

rdata2_ID_EXE[31:0] = 00000000 (hexadecimal) (From register file)

imm_ID_EXE[31:0] = 00000001 (hexadecimal) (Sign extended immediate value)

aluop_ID_EXE[2:0] = 0 (hexadecimal) (Opcode for ALU)

waddr_ID_EXE[4:0] = 09 (hexadecimal) (For write address of register file)

However, in the ISIM windows, the values from the ID_EXE register will only be seen in the 3rd stage which is the Execute stage as the values in the wire from the ID_EXE register is only read when it is retrieving from the ID_EXE register. Retrieving of values from the ID_EXE register only occurs in the 3rd cycle which is the Execute stage.

3rd cycle: In the Execute stage, values of 00000000 (hexadecimal) from register and 00000001 (hexadecimal) of signed extended immediate values are obtained from the ID_EXE pipeline register and added through the ALU. All the values are stored in the EXE_MEM pipeline register.

In the EXE_MEM register:

waddr_EXE_MEM[4:0] = 09 (hexadecimal) (For write address of register file)

aluout_EXE_MEM[31:0] = 00000001 (hexadecimal) (From the output of ALU)

rdata2_EXE_MEM[31:0] = 00000000 (hexadecimal) (From ID_EXE pipeline register)

However, in the ISIM windows, the values from the EXE_MEM register will only be seen in the 4th stage which is the Memory stage as the values in the wire from the EXE_MEM register is only read when it is retrieving from the EXE_MEM register. Retrieving of values from the EXE_MEM register only occurs in the 4th cycle which is the Memory stage.

4th cycle: In the Memory stage, value of 00000001 (hexadecimal) from the EXE_MEM pipeline register which is from the output of ALU, is used as the read address of data memory. A value of 0000000c (hexadecimal) is obtained from the data memory from the address of 00000001 (hexadecimal) which can be seen in dmemdata[31:0]. The values are stored in the MEM_WB register.

In the MEM_WB register:

waddr_MEM_WB[4:0] = 09 (hexadecimal) (For write address of register file)

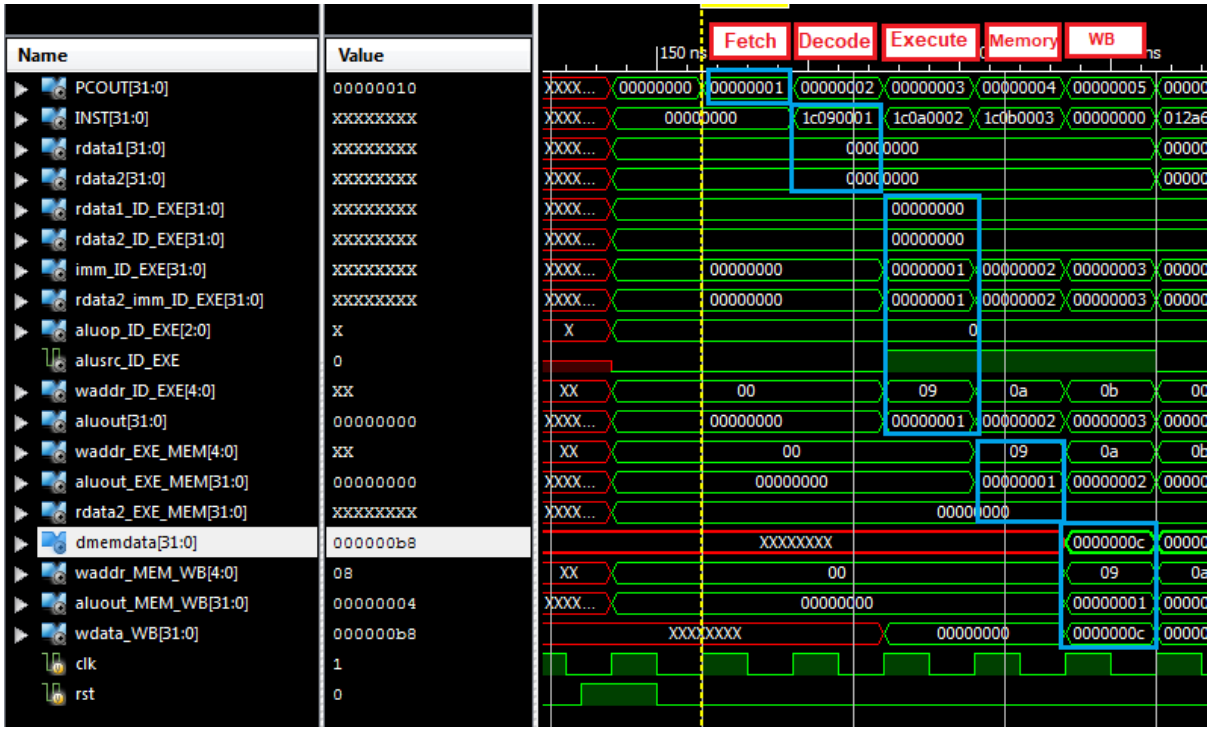
aluout_MEM_WB [31:0] = 00000001 (hexadecimal) (From the output of ALU)

dmemdata[31:0] = 0000000c (hexadecimal) (From data memory)

However, in the ISIM windows, the values from the MEM_WB register will only be seen in the 5th stage which is the WriteBack stage as the values in the wire from the MEM_WB register is only read when it is retrieving from the MEM_WB register. Retrieving of values from the MEM_WB register only occurs in the 5th cycle which is the WriteBack stage. Output of dmemdata[31:0] will only be reflected on the 5th cycle which is the WriteBack stage as stated in the lab manual that the output of DMEM is provided after one clock cycle.

5th cycle: In the WriteBack stage, 0000000c (hexadecimal), as shown in wdata_WB[31:0], is written to address of 09 (hexadecimal) which was obtain from the MEM_WB pipeline register.

Below the blue boxes is the workflow for LW $9,1($0):



2. SW instruction workflow on the five stage pipeline

SW $8,4($0)

This is the first instruction of the set of MIPS assembly code written for the computation of “d = (a+b)*(b-c)”. It will store the content in register $8 into address 000000004 (hexadecimal) in data memory. It is assumed that each stage of the pipeline takes one cycle each.

1st cycle: In the Fetch stage, PC provides the read address of 0000000c (hexadecimal) to the instruction memory. This can be seen at PCOUT[31:0] between the time of 495ns and 525ns. Based on the read address of 0000000c (hexadecimal) given by the PC, content of 20080004 (hexadecimal) in address of 0000000c (hexadecimal) in instruction memory will be read and transferred to register file. However, due to delay of reading the content of INST[31:0], the value of INST[31:0] is only reflected in the 2nd cycle which is in Decode stage.

2nd cycle: In the Decode stage, read address of 00000 (binary) from instruction value of 20080004 (hexadecimal) will be used to obtain the content of address 000000 (binary) from the register file. Read address of 00008 (binary) from instruction value of 20080004 (hexadecimal) will be used to obtain the data of 000000b8 (hexadecimal) from the register file. 0004 (hexadecimal) from the instruction value of 20080004 (hexadecimal) will be sign extended to 00000004 (hexadecimal). Finally, the values will be stored in the ID_EXE pipeline register.

In the ID_EXE pipeline register:

rdata1_ID_EXE[31:0] = 00000000 (hexadecimal) (From register file)

rdata2_ID_EXE[31:0] = 000000b8 (hexadecimal) (From register file)

imm_ID_EXE[31:0] = 00000004 (hexadecimal) (Sign extended immediate value)

aluop_ID_EXE[2:0] = 0 (hexadecimal) (Opcode for ALU)

waddr_ID_EXE[4:0] = 08 (hexadecimal) (For write address of register file)

However, in the ISIM windows, the values from the ID_EXE register will only be seen in the 3rd stage which is the Execute stage as the values in the wire from the ID_EXE register is only read when it is retrieving from the ID_EXE register. Retrieving of values from the ID_EXE register only occurs in the 3rd cycle which is the Execute stage.

3rd cycle: In the Execute stage, values of 00000000 (hexadecimal) from register and 00000004 (hexadecimal) of signed extended immediate values are obtained from the ID_EXE pipeline register and added through the ALU. All the values are stored in the EXE_MEM pipeline register.

In the EXE_MEM register:

waddr_EXE_MEM[4:0] = 08 (hexadecimal) (For write address of register file)

aluout_ EXE_MEM[31:0] = 00000004 (hexadecimal) (From the output of ALU)

rdata2_EXE_MEM[31:0] = 0000s00b8 (hexadecimal) (From ID_EXE pipeline register)

However, in the ISIM windows, the values from the EXE_MEM register will only be seen in the 4th stage which is the Memory stage as the values in the wire from the EXE_MEM register is only read when it is retrieving from the EXE_MEM register. Retrieving of values from the EXE_MEM register only occurs in the 4th cycle which is the Memory stage.

4th cycle: In the Memory stage, value of 00000004 (hexadecimal) from the EXE_MEM pipeline register which is from the output of ALU, is used as the write address of data memory. A value of 000000b8 (hexadecimal) is written to the data memory from the address of 00000004 (hexadecimal) which can be seen in dmemdata[31:0]. The values are stored in the MEM_WB register.

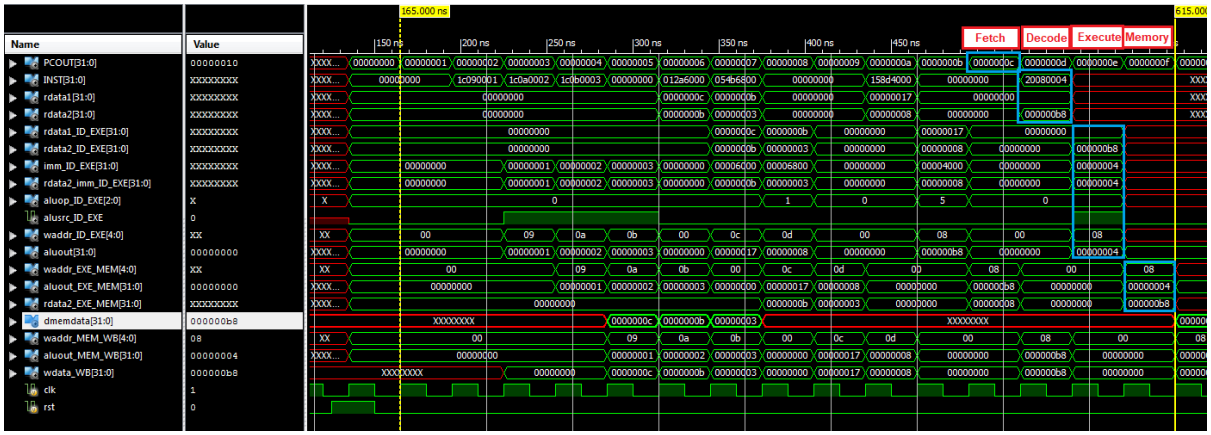
In the MEM_WB register:

waddr_MEM_WB[4:0] = 08 (hexadecimal) (For write address of register file)

aluout_MEM_WB [31:0] = 00000004 (hexadecimal) (From the output of ALU)

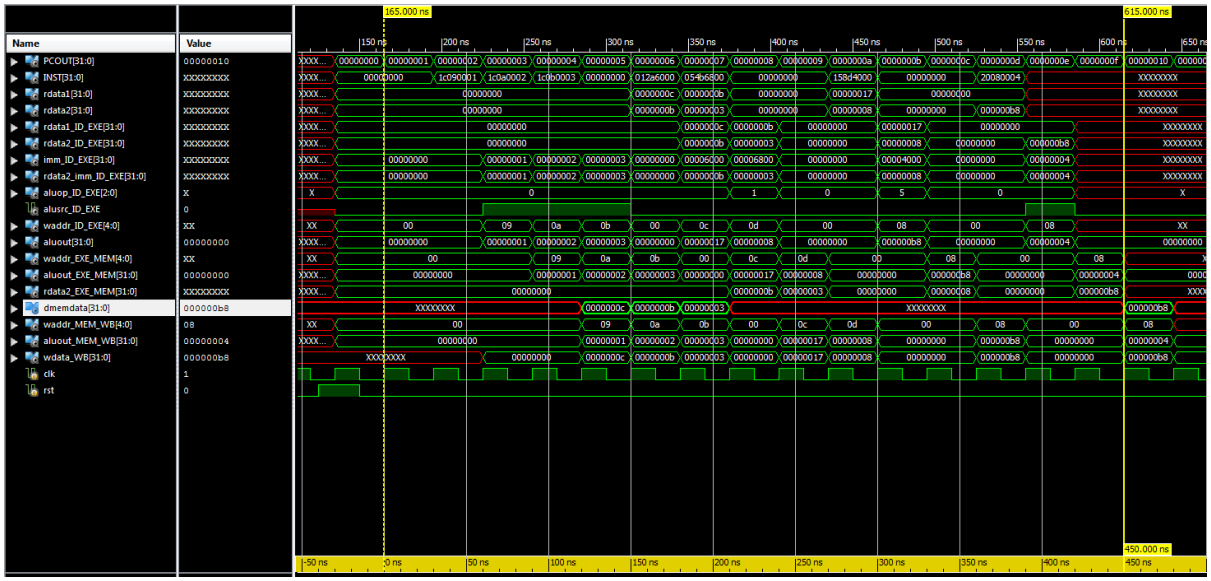
However, in the ISIM windows, the values from the MEM_WB register will only be seen in the 5th stage which is the WriteBack stage as the values in the wire from the MEM_WB register is only read when it is retrieving from the MEM_WB register. Retrieving of values from the MEM_WB register only occurs in the 5th cycle which is the WriteBack stage. Output of dmemdata[31:0] will only be reflected on the 5th cycle which is the WriteBack stage as stated in the lab manual that the output of DMEM is provided after one clock cycle.

Below the blue boxes is the workflow for SW $8,4($0):



Execution time for running the program

The execution time for the running program is 450ns.



The steady state CPI of the code

Steady state CPI = (Number of instructions + number of stalls) / Number of instructions

= (7+5) / 7

= 1.7142857143

≈ 1.71 (3s.f)