Simulation Report

MIPS CPU

**Appendix B**

|  |  |
| --- | --- |
| Students: | Otabek Sherman -326868775 |
|  | Lidor Cohen - 305782732 |

Answer the following questions.

B.1) What are the limitations due to the pipeline latency of the following combinations:

* lw after add where the add Rd is the lw Rs
* lw after add where the add Rd is the lw Rt
* add after lw where the lw Rt is the add Rt
* beq after lw where the lw Rt is the beq Rs

Use a similar figure to Fig.2 and Fig. 3 to demonstrate your answers. Explain your answer!

B.1.a - lw after add where the add Rd is the lw Rs [ e.g., lw $4,16($3) ]

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

add **$3**,$5,$8

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX



We will wait until the WB step is over, after that we can access the GPR and get the correct Rs value. If we will not wait we will get the incorrect value in the register because the add instruction is before lw instruction and the use the same register to write.

B.1.b - lw after add where the add Rd is the lw Rt

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

add **$3**,$5,$8

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX



First the program storing the information in the register $3, in the lw operation the information that stored in $3 will be overwritten.

The limitation is imposed by the logic of the instructions.

B.1.c - add after lw where the lw Rt is the add Rt

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

lw **$3**,16($10)

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX



First we write to the GPR the result of the lw instruction in the WB step. Next after 3 nops in the ID step the add operation opens the result from the previous lw operation, but after 3 cycles the result is already written.

B.1.d - beq after lw where the lw Rt is the beq Rs

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

lw **$3**,16($10)

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX



In lw instruction: GPR[Rt]<=DMEM[GPR[$10]+16];

In beq instruction: If GPR[Rt]=GPR[Rs] then

Offset(branch offset);

If beq instruction comes after lw instruction, then we need to wait 3 nop instruction while the information is stored in the GPR[Rt]. Then in the beq instruction we can use the correct data in GPR[Rt].

B.2) What are the limitations of all cases of B.1 after you add he Data Forwarding? . Explain your answer!

B.2.a - lw after add where the add Rd is the lw Rs

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

add **$3**,$5,$8

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

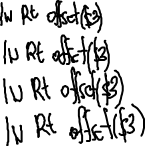
MEM

MEM

MEM

MEM

EX



In lw instruction: GPR[Rt]=DMEM[GPR[$3]+offset];

* The first lw is performed without latency of the pipeline, because we can forward and use the result of the add instruction from ALU out to the EX step of the lw instruction by comparing the Rs\_pEX with Rd\_pMEM.
* We can use the data from the WB step of the add instruction. This data is forwarded instead of the storing it in the GPR. By this way we have no latency in second and third ls instructions. In second lw instruction the data is written in the EX step, and in the third la instruction the data is written in the ID step.
* The 4th lw instruction is also performed without latency.

B.2.b - lw after add where the add Rd is the lw Rt

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

add **$3**,$5,$8

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX



As in the last question, if the data is written to the register $3 in the add instruction in the WB phase, and in the WB phase of the lw instruction.

B.2.c - add after lw where the lw Rt is the add Rt

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

lw **$3**,16($10)

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

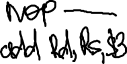
MEM

MEM

MEM

MEM

EX



If the add comes after lw, we need to forward data in the EX step of the add and to wait for 1 clock to get the correct value in register $3.

B.2.d - beq after lw where the lw Rt is the beq Rs

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

lw **$3**,16($10)

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX



In the lw instruction the data written to the Rt register. We wait 2 clocks for information being successful written to the GPR from the WB step of ls instruction, then we can use the data from Rt in the WB step of the beq instruction.

B.3) How many times do we perform the instruction following a jal instruction? Explain in detail. What are the implications? If this is a problem, what do you suggest in order to solve it?

Before the program jump, the program still fetch, decode and run the instructions in the pipeline, so we execute twice the instructions coming after instruction jal. Also we can put an instruction nop after jal instruction to avoid the execution of the same instruction twice.

B.4) How soon after jal instruction can we issue a jr $31 instruction in order to return to the right location in the code? Give the answer before data forwarding is added and then after the data forwarding is added. . Explain your answer!

No data forwarding:

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

jal routine1

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX



In case of no data forwarding, we need to wait until the data is successful stored in jal instruction. It means that we need to wait until the WB step of the jal instruction.

With data forwarding:

WB

EX

ID

IF

WB

EX

ID

IF

WB

EX

ID

IF

jal routine1

WB

EX

ID

IF

WB

EX

ID

IF

CK

MEM

MEM

MEM

MEM

MEM

EX



In case of data forwarding we will wait 2 nop instructions after jal instruction to get the correct data, then we have the data already written in register $31 in the ID step of jr instruction.