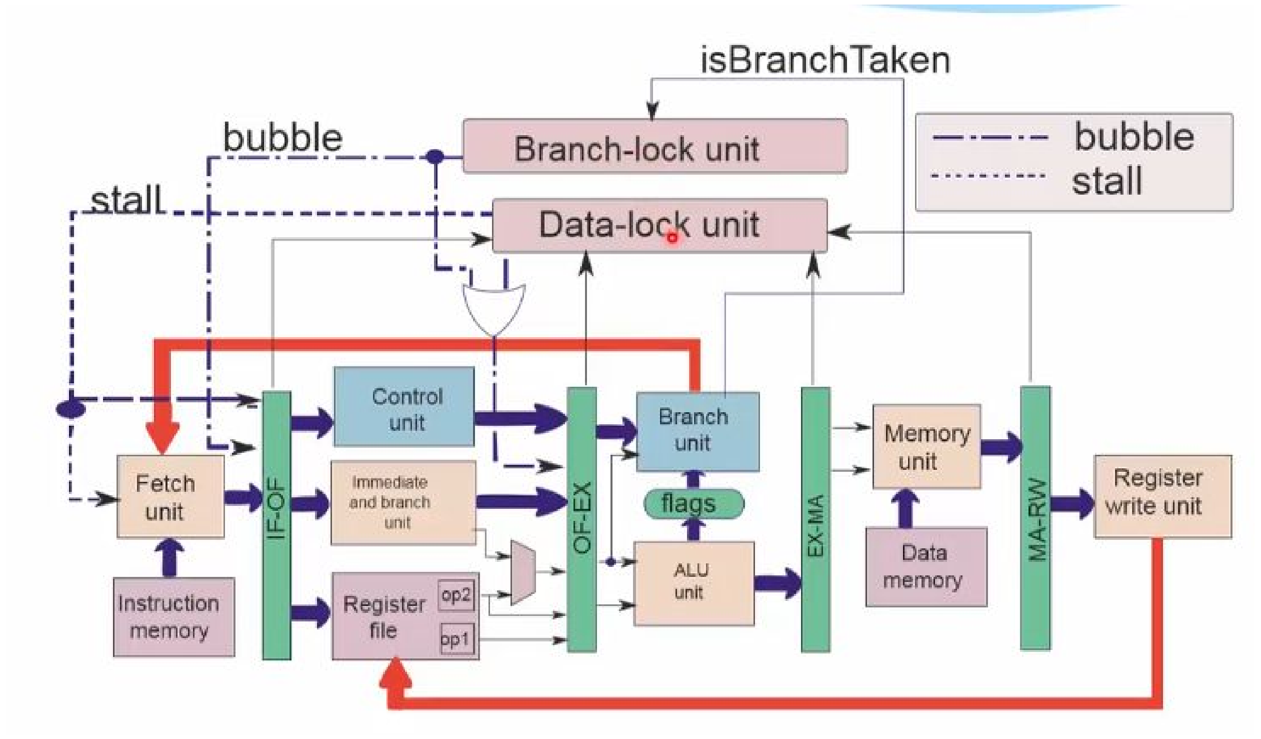
180010009 Akhilesh Bharadwaj

Computer Architecture
Assignment 7

Question 1:
The most efficient way to do with using interlocks is to let the next two instructions after the branch statement enter the pipeline and only if the **branch is taken** then only convert them to nop/bubbles.
In this way we make sure of functional correctness of code and if the branch is not taken we won't lose any time.



Using this as a compact view of the pipeline.

We can see here that when **isBranchtaken** is set to true in the Execute stage then it will send a signal to its previous stage to send a nop/bubble forward in the pipeline. Everytime when **isBranchtaken** is true we have to pay a penalty of performance.
Now we can see that when **isBranchtaken** signal is true 2 instructions already entered the pipeline so the minimum number of bubbles to introduce is 2.

Question 2:
No, there is no way to reduce performance impact introduced through inter-locking based handling of control hazards. As one way to reduce impact is use of out of order pipelining (executing instructions which are before branch inst which are unrelated to branch inst and always going to execute. Let them execute after branch inst). But we are currently using in-order pipelining so there is no way(as far as i can think of) to reduce impact caused by this.

Question 3:
There are 3 instruction
One in RW stage, //write
One in MA stage, and //write (WAW)
One in OF stage. //read (RAW)

**Assuming WAW was already solved in previous clock cycles.*

To maintain functional correctness we need to get recent updates/latest results. Last write statement as r5 being dest operand is in MA stage and will contain latest results.
Yes, Forwarding can be done.
Instruction presently in MA can be a load operation. So we should wait for another cycle to perform its load operation. Also we need a value of r5 at the start of Execute stage, so given circumstances we can wait for another cycle to complete. After one cycle we can forward value in RW to EX stage. So we can do RW->EX(2-stage) forwarding. This way we get the latest value and we don't face any data hazard.

Question 4:
P ∝ (IPC * f) / # inst
IPC: instruction per cycle = 1/CPI = 1
fp1: frequency for 1st processor
fp2: frequency for 2nd processor
# inst : num instructions

30% of instructions are branch instruction
80% of those branch instructions are branch not taken

In total we have 6% of those instructions will face penalty as we already loaded next two instructions which will be converted to nops
Effective number of instruction because of penalty will be
For processor 1: 112% = (100 + 6*(penalty=2))
For processor 1: 118% = (100 + 6*(penalty=3))

Performance1 ∝ 1*1GHz/1.12 * # inst
Performance2 ∝ 1*1.2GHz/1.18 * # inst
Performance 1: Performance 2 = 87:100 //approx.

So processor 2 is performing better than processor 1. Processor 2 gives better performance.