Using carry lookahead adder to accelerate adders

**Data hazard**

1. Addi x1 x0 1 addi x2 x1 1

Solution: data forwarding, forward the data from exec to decode

Add mux at operand a and operand B

1. Addi x1(mem access) x0 1 addi x2 x0 2 addi x3 x1 2(decode)

Forward from mem access to decode

1. Addi x1 x0 1(WB) addi x2 x0 x2 (mem access) addi x3 x0 3 (exec) addi x4 x1 3(decode)

Forward WB to decode

Load use

1. Ld x1 x0 9 (exec) addi x2 x1 1(decode)

Solution: wait nop

1. Ld x1 x0 9(mem access) addi x3 x0 3 (exec) addi x2 x1 1(decode)

Load store

1. Ld x1 x0 9(WB) sd x1(used at mem access) x2 3

Can get the right x1

Data forwarding from WB to mem access , add a mux at data memory din

1. Ld x1 x0 9(mem access) sd x2 x1(used at ALU) 3

Same as load and use

Solution: stop and wait nop

**Control hazard:**

Beq:

Don’t know branch or not when fetch the instruction

Solution:

1. Stop and Wait(not good)
2. Assume no branch,if branch flush 2 instructions in the pipeline( not good for jal and jalr)
3. Get branch in advance(get earliest to decode at best) ,and assume no branch(only flush 1 instruction)
4. **Can add jal and logic to instruction fetch stage( mini decoder),jalr and branch to decoder stage**
5. **Branch prediction at IF**

PC + imm(imm>0 or imm<0)

If beq to previous instructions, assume branch(good for loop)

If beq to later instructions, assume no branch.

Solutions:

move branch logic to decode.

Mini decoder at instruction fetch.

* If branch has data hazard with register, have to wait

1. Addi x1(exec) x0 1 Beq x1 x2 1(dec) wait till finish exec
2. Ld x1 x0 9 (exec) beq x1 x2 1(dec) wait till load to mem access

Stall operations:

Feed the input signal to itself using mux and control it with a control signal.

Flush pipeline:

Reset the register.

